Apache NiFi Developer Guide

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache NiFi Developer Guide.....4

Apache NiFi Developer Guide

For more information, see the Apache NiFi Developer Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi Developer's Guide

Apache NiFi Expression Language Guide

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache NiFi Expression Language Guide..... 4

Apache NiFi Expression Language Guide

For more information, see the Apache NiFi Expression Language Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi Expression Language Guide

Cloudera Flow Management 2.1.6

Apache NiFi RecordPath Guide

Date published: 2019-06-26
Date modified: 2022-12-13

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache NiFi RecordPath Guide.....4

Apache NiFi RecordPath Guide

For more information, see the Apache NiFi RecordPath Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi RecordPath Guide

Apache NiFi Registry REST API

Date published:
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache NiFi Registry REST API..... 4

Apache NiFi Registry REST API

For more information, see the Apache NiFi Registry REST API Guide.

Important:

The Apache NiFi Registry documentation is available both online and within the Registry NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi Registry version. If you are using an earlier NiFi Registryversion, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi Registry version directly from the NiFi Registry UI:

1. Launch the NiFi Registry UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi Registry REST API documentation

Apache NiFi Registry System Administrator Guide

Date published:
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache NiFi Registry System Administrator's Guide..... 4

Apache NiFi Registry System Administrator's Guide

For more information, see the Apache NiFi Registry System Administrator's Guide.

Important:

The Apache NiFi Registry documentation is available both online and within the Registry NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi Registry version. If you are using an earlier NiFi Registryversion, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi Registry version directly from the NiFi Registry UI:

1. Launch the NiFi Registry UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi Registry System Administrator’s Guide

Cloudera Flow Management 2.1.6

Apache NiFi REST API

Date published: 2019-06-26
Date modified: 2022-12-13

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache NiFi REST API Reference.....4

Apache NiFi REST API Reference

For more information, see the Apache NiFi REST API Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi REST API documentation

Apache NiFi System Administrator Guide

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache NiFi System Administrator's Guide..... 4

Apache NiFi System Administrator's Guide

For more information, see the Apache NiFi System Administrator's Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi System Administrator's Guide

Cloudera DataFlow for Data Hub 7.3.1

Cloudera Flow Management NiFi Reference

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Getting started with Apache NiFi.....4

Using Apache NiFi.....4

Apache NiFi Expression Language Guide.....4

Apache NiFi RecordPath Guide.....4

Apache NiFi System Administrator's Guide.....5

Using Apache NiFi Toolkit.....5

Apache NiFi Developer's Guide.....5

Getting started with Apache NiFi

For more information, see the Apache NiFi Getting Started Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Using Apache NiFi

For more information, see the Apache NiFi User Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Apache NiFi Expression Language Guide

For more information, see the Apache NiFi Expression Language Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Apache NiFi RecordPath Guide

For more information, see the Apache NiFi RecordPath Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Apache NiFi System Administrator's Guide

For more information, see the Apache NiFi System Administrator's Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Using Apache NiFi Toolkit

For more information, see the Apache NiFi Toolkit Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Apache NiFi Developer's Guide

For more information, see the Apache NiFi Developer's Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Cloudera DataFlow for Data Hub 7.3.1

Cloudera Flow Management NiFi Registry Reference

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Getting started with Apache NiFi Registry..... 4

Using Apache NiFi Registry..... 4

Apache NiFi Registry System Administrator's Guide..... 4

Getting started with Apache NiFi Registry

For more information, see the Apache NiFi Registry Getting Started Guide.

Important:

The Apache NiFi Registry documentation is available both online and within the Registry NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi Registry version. If you are using an earlier NiFi Registryversion, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi Registry version directly from the NiFi Registry UI:

1. Launch the NiFi Registry UI.
2. From the Global Menu, select Help Documentation .

Using Apache NiFi Registry

For more information, see the Apache NiFi Registry User Guide.

Important:

The Apache NiFi Registry documentation is available both online and within the Registry NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi Registry version. If you are using an earlier NiFi Registryversion, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi Registry version directly from the NiFi Registry UI:

1. Launch the NiFi Registry UI.
2. From the Global Menu, select Help Documentation .

Apache NiFi Registry System Administrator's Guide

For more information, see the Apache NiFi Registry System Administrator's Guide.

Important:

The Apache NiFi Registry documentation is available both online and within the Registry NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi Registry version. If you are using an earlier NiFi Registryversion, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi Registry version directly from the NiFi Registry UI:

1. Launch the NiFi Registry UI.
2. From the Global Menu, select Help Documentation .

Cloudera DataFlow for Data Hub 7.3.1

Connecting Kafka Clients to Data Hub Provisioned Clusters

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Connecting Kafka clients to Data Hub provisioned clusters.....4

Connecting Kafka clients to Data Hub provisioned clusters

Learn how to connect Kafka clients to clusters provisioned with Data Hub.

About this task

Use the following steps to connect Kafka clients to clusters provisioned with Data Hub. Configuration examples provided in this list of steps assume that the cluster you are connecting to was provisioned with a Streams Messaging cluster definition.

Note:

These instructions are for connecting Kafka clients to Data Hub provisioned Kafka clusters. For information on connecting NiFi to Kafka within the same CDP Public Cloud environment, see Ingesting data into Apache Kafka.

Before you begin

- If you are connecting your clients from outside of your virtual network (VPC or Vnet) verify that both inbound and outbound traffic is enabled on the port used by Kafka brokers for secure communication. The default port is 9093. For more information, see the following resources:
  - AWS: Security Groups for Your VPC
  - Azure: How to open ports to a virtual machine with the Azure portal
- If you are connecting your clients over the internet, verify that your virtual network (VPC or Vnet) is assigned a public IP address. For more information, see the following resources:
  - AWS: IP Addressing in Your VPC
  - Azure: Associate a public IP address to a virtual machine
- Clients connecting to Data Hub provisioned clusters require a CDP user account that provides access to the required CDP resources. Verify that a CDP user account with the required roles and permissions is available for use. If not, create one. Any type of CDP user account can be used. If you are creating a new account to be used by Kafka clients, Cloudera recommends that you create a machine user account. For more information, see User Management in the Cloudera Management Console documentation.
- In addition to the CDP user account having access to the required CDP resources, the user account also needs to have the correct policies assigned to it in Ranger. Otherwise, the client cannot perform tasks on Kafka resources. These policies are specified within the Ranger instance that provides authorization to the Kafka service you want to connect to. For more information, see the Cloudera Runtime documentation on Apache Ranger and the Kafka specific Ranger documentation.

Procedure

1. Obtain the FreeIPA certificate of your environment:
   a) From the CDP Home Page navigate to Management Console Environments .
   b) Locate and select your environment from the list of available environments.
   c) Go to the FreeIPA tab.
   d) Click Get FreeIPA Certificate.
   The FreeIPA certificate file, *[***ENVIRONMENT NAME***]*.crt, is downloaded to your computer.

2. Add the FreeIPA certificate to the truststore of the client.

   The certificate needs to be added for all clients that you want to connect to the Data Hub provisioned cluster. The exact steps of adding the certificate to the truststore depends on the platform and key management software used. For example, you can use the Java Keytool command line tool:

```
keytool -import -keystore [***CLIENT TRUSTSTORE.JKS***] -al
ias [***ALIAS***] -file [***FREEIPA CERTIFICATE***]
```

   Tip: This command creates a new truststore file if the file specified with the -keystore option does not exist.

3. Obtain CDP workload credentials:

   A valid workload username and password has to be provided to the client, otherwise it cannot connect to the cluster. Credentials can be obtained from Management Console.

   a) From the CDP Home Page navigate to Management Console User Management.
   b) Locate and select the user account you want to use from the list of available accounts.
      The user details page displays information about the user.
   c) Find the username found in the Workload Username entry and note it down.
   d) Find the Workload Password entry and click Set Workload Password.
   e) In the dialog box that appears, enter a new workload password, confirm the password and note it down.
   f) Fill out the Environment text box.
   g) Click Set Workload Password and wait for the process to finish.
   h) Click close.

4. Configure clients.

   In order for clients to be able to connect to Kafka brokers, all required security related properties have to be added to the client's properties file. The following example configuration lists the default properties that are needed when connecting clients to a cluster provisioned by Data Hub with a Streams Messaging cluster definition. If you made changes to the security configuration of the brokers, or provisioned a custom cluster with non-default Kafka security settings, make sure to change the appropriate parameters in the client configuration as well.

```
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
ssl.truststore.location=[***CLIENT TRUSTSTORE.JKS***]
ssl.truststore.password=[***TRUSTSTORE PASSWORD***]
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
 required \
    username="[***USERNAME***]" \
    password="[***PASSWORD***]";
```

   Replace *[***CLIENT TRUSTSTORE.JKS***]* with the path to the client's truststore file.This is the same file that you added the FreeIPA certificate to in Step 2 on page 5.

   Replace *[***TRUSTSTORE PASSWORD***]* with the password of the truststore file.

   Replace *[***USERNAME***]* and *[***PASSWORD***]* with the workload username and password obtained in Step 3 on page 5.

5. Obtain Kafka broker hostnames:

   You can obtain the Kafka broker hostnames from the Cloudera Manager UI.

   a) From the CDP Home Page navigate to Management Console Environments .
   b) Locate and select your environment from the list of available environments.
   c) Select the Data Hub cluster you want to connect to from the list of available clusters.
   d) Click the link found in the Cloudera Manger Info section.
      You are redirected to the Cloudera Manager web UI.
   e) Click Clusters and select the cluster that the Kafka service is running on.

      The default name for clusters created with a Streams Messaging Cluster definition is streams-messaging.
   f) Select the Kafka service.
   g) Go to Instances.

      The Kafka broker hostnames are listed in the Hostname column.

6. Connect clients to brokers.

   Connect the clients by supplying them with the broker hostnames obtained in step 5 on page 6. The actions you need to take differ depending on the type of client you are using.

   Custom developed Kafka Applications

   When producing or consuming messages with your own Kafka client application, you have to provide the Kafka broker hostnames within the client code.

   Kafka console producer and consumer

   When producing or consuming messages with the kafka-console-consumer or kafka-console-producer command line tools, run the producer or consumer with the appropriate hostnames. Additionally, you must also pass the client properties file containing the security related properties with --producer.config or --consumer.config. For example:

   ```
   kafka-console-producer --broker-
   list [***HOSTNAME***]:[***PORT***] --topic [***TOPIC***] --produ
   cer.config [***CLIENT PROPERTIES FILE***]
   ```

   ```
   kafka-console-consumer --bootstrap-serve
   r [***HOSTNAME***]:[***PORT***] --topic [***TOPIC***] --from-begi
   nning --consumer.config [***CLIENT PROPERTIES FILE***]
   ```

   Replace *[***CLIENT PROPERTIES FILE***]* with the path to the client's properties file. This is the same file that you updated in Step 4 on page 5.

7. To connect to Schema Registry, you must also set the following properties:

   ```
   schema.registry.url=<SCHEMA_REGISTRY_ENDPOINT>
   schema.registry.auth.username=<USERNAME>
   schema.registry.auth.password=<PASSWORD>
   ```

   - If schema.registry.url is not set, the client does not try to connect to Schema Registry.
   - If this property is set, the username and password must also be configured.
   - The Schema Registry endpoint can be found in the CDP Console, under the Endpoints tab of your DataHub Kafka cluster.

   Some applications may need to connect directly to Schema Registry to interact with it. For example, to retrieve or store a schema.

   A Schema Registry API client is also provided by Cloudera for these cases. For example, to retrieve the latest version of a schema you can use the following:

   ```
   Map<String, Object> config = new HashMap<>();
   config.put("schema.registry.url", "<SCHEMA_REGISTRY_ENDPOINT>");
   config.put("schema.registry.auth.username", "<USERNAME>");
   ```

```
config.put("schema.registry.auth.password", "<PASSWORD>");

String topicName = "my-topic";
SchemaRegistryClient client = new SchemaRegistryClient(config);
try {
    SchemaVersionInfo latest = client.getLatestSchemaVersionInfo(topicNam
e);
    System.out.println(latest.getSchemaText());
} catch (SchemaNotFoundException e) {
    LOG.info("Schema [{}] not found", topicName);
    throw e;
}
```

Results

Kafka clients and Schema Registry are configured and are able to connect to Data Hub provisioned clusters.

Cloudera DataFlow for Data Hub 7.3.1

Analyzing data with Apache Flink

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Analyzing your data with Kafka 4
Understand the use case 4
Prepare your environment 4
Assign resource roles 4
Create IDBroker mapping 5
Set workload password 5
Create your streaming clusters 6
Set Ranger policies 6
Retrieve and upload keytab file 7
Create Atlas entity type definitions 8
Running your Flink application 9
Job monitoring with Flink Dashboard 11
Metadata governance with Atlas 12

Analyzing your data with Kudu 12
Running your Flink application 12
Job monitoring with Flink Dashboard 14
Metadata governance with Atlas 14

Analyzing your data with HBase 14
Running your Flink application 14
Job monitoring with Flink Dashboard 15
Metadata governance with Atlas 16

Analyzing your data with Kafka

Understand the use case

As a basic analytical use case, you can use Streaming Analytics clusters connected to the Streams Messaging cluster to build an end-to-end streaming analytics application with Kafka as source and sink.

When choosing Kafka as a connector for a Flink application, you can create a scalable communication channel. Kafka, as a source and sink, is responsible for delivering input records to Flink, and receiving the transformed data from Flink.

This use case details the steps to connect a Streaming Analytics cluster with a Streams Messaging cluster, and to submit your Flink jobs and monitor the correct behaviour of your application in the Data Hub environment. The Stateful Flink Application Tutorial is used as an example to guide you through the basic steps to analyze your data with Kafka in CPD Public Cloud.

The Stateful Flink Application Tutorial details an inventory use case for an e-commerce site. The business logic in the application includes handling transactions and managing queries that are later summarized to monitor the state of the inventory.

Note: In the Analyzing data with Kafka use case, the steps that are used from the Stateful Flink Application Tutorial are highlighted in grey. These steps are not mandatory to use Streaming Analytics in Data Hub, but serves as an example for practice.

For more information about the tutorial, see the Stateful Flink Application Tutorial.

Related Information

Stateful Flink Application Tutorial

Prepare your environment

Before you start analyzing your data using Flink and Kafka in CDP Public Cloud, you need to register and prepare your environment so that a chosen user or group can use the clusters and services in the environment, and submit Flink jobs securely.

As a first step, an admin needs to register an AWS, Azure or GCP environment. In CDP Public Cloud, this environment is based on the virtual private network in your cloud provider account. This means that CDP can access and identify the resources in your cloud provider account which is used for CDP services, and shared between clusters in the environment. After registering your AWS, Azure or GCP environment, you can provision clusters, and set users or groups to access your environment and services.

For more information about registering an AWS, Azure or GCP environment, see the Management Console documentation.

In this use case, an administrator with AdminEnvironment resource role registers an AWS environment where flink_users group is added as EnvironmentUser. The members of flink_users have the same resource role that is set for the group.

Assign resource roles

As an administrator, you need to give permissions to users or groups to be able to access and perform tasks in your Data Hub environment.

Procedure

1. Navigate to Management Console > Environments and select your environment.

2. Click Actions > Manage Access .
3. Search for a user or group that needs access to the environment.
4. Select EnvironmentUser role from the list of Resource Roles.
5. Click Update Roles.
   The Resource Role for the selected user or group will be updated.
6. Navigate to Management Console > Environments , and select the environment where you want to create a cluster.
7. Click Actions > Synchronize Users to FreeIPA .
8. Click Synchronize Users.

   Note: There might be cases where the status of the environment is synchornized with warnings and has failed status. This does not indicate that the synchronization has failed.

   The following short video shows how you can assign the resource roles:

   ../images/csa-datahub-2-resource-role.mp4

Create IDBroker mapping

As an administrator, you must create IDBroker mapping for a user or group to access cloud storage. As a part of Knox, the IDBroker allows a user to exchange cluster authentication for temporary cloud credentials.

Procedure

1. Navigate to Management Console > Environments and select your environment.
2. Click Actions > Manage Access .
3. Click on the IDBroker Mappings tab.
4. Click Edit to add a new user or group and assign roles to have writing access for the cloud storage.
5. Search for the user or group you need to map.
6. Go to the IAM Summary page where you can find information about your cloud storage account.
7. Copy the Role ARN.
8. Go back to the IDBroker Mapping interface on the Cloudera Management Console page.
9. Paste the Role ARN to your selected user or group.
10. Click Save and Sync.

    The following short video shows how you can create the IDBroker mapping:

    ../images/csa-datahub-14-id-mapping.mp4

Set workload password

As a user, you need to set a workload password for your EnvironmentUser account to be able to access the Flink nodes through SSH connection.

Procedure

1. Navigate to Management Console > Environments and select your environment.
2. Click Actions > Manage Access .
3. Click Workload Password.
4. Give a chosen workload password for your user.
5. Confirm the given password by typing it again.
6. Click Set Workload Password.

   The following short video shows how you can set the workload password:

   ../images/csa-datahub-15-workload-password.mp4

Create your streaming clusters

As a user, you need to create the Streaming Analytics and Streams Messaging clusters. In this use case, the streaming clusters are created in the same Data Hub environment.

Procedure

1. Navigate to Management Console > Environments , and select the environment where you want to create a cluster.
2. Click Create Data Hub.
3. Select Streaming Analytics Light Duty cluster from Cluster Definition.
4. Provide a chosen cluster name.
5. Click Provision Cluster.
6. Navigate back to Management Console > Environments , and select the environment where you have created the Streaming Analytics cluster.
7. Click Create Data Hub.
8. Select Streams Messaging Light Duty cluster from Cluster Definition.
9. Provide a chosen cluster name.
10. Click Provision Cluster.

    The following short video shows how you can create the streaming clusters:

    ../images/csa-datahub-1-cluster-creation.mp4

Set Ranger policies

As an administrator, you need to set resource-based Ranger policies to give users or groups access to write and read Kafka topics.

Procedure

1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
2. Click on the Data Lake tab.
3. Select Ranger from the services.
   You are redirected to the Ranger user interface.
4. Select your Streams Messaging cluster under the Kafka folder on the Service Manager page.
5. Click Add new policy.
6. Provide the details for the topic policy.
   a) Give a chosen policy name.
   b) Select topic as Kafka resource type.
   c) Provide a prefix that is included in the name of the flink topics you need to access when writing messages.
7. Select the user or groups under Allow Conditions.
8. Click Add Permissions.
9. Select individual or all permissions for the policy.
10. Click Add.
    You are redirected to the List of Policies page.
11. Click Add new policy.
12. Provide the details for the consumer group policy.
    a) Give a chosen policy name.
    b) Select consumergroup as Kafka resource type.
    c) Provide a prefix that is included in the name of the flink topics you need to access when reading messages.
13. Select the user or groups under Allow Conditions.

14. Click Add Permissions.
15. Select individual or all permissions for the policy.
16. Click Add.

    The following short video shows how you can set Ranger policies:../images/csa-datahub-4-set-ranger.mp4

Grant permission for the ATLAS_HOOK topic

The Flink Atlas integration requires that users running Flink jobs have the privileges to write to the ATLAS_HOOK Kafka topic. Before you can use Atlas metadata management with Flink, you must grant permission to the user or user group to be able to send the Atlas lineage data through the ATLAS_HOOK Kafka topic.

Before you begin

Make sure that the atlas service user has publish privileges to the ATLAS_HOOK topic. For more information, see the Atlas documentation.

Procedure

1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
2. Click on the Data Lake tab.
3. Select Ranger from the services.
   You are redirected to the Ranger user interface.
4. Select cm_kafka under the Kafka folder on the Service Manager page.
5. Search for the ATLAS_HOOK from the Policy Name.
6. Click on Edit.
7. Add the user or group to the Select User text box under the Allow Conditions header.
8. Click Save.

Retrieve and upload keytab file

As a user, you need to retrieve the keytab file of your profile and upload it to the Streaming Analytics cluster to securely submit your Flink jobs.

Procedure

1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
2. Click on your profile name.
3. Click Profile.
4. Click Actions > Get Keytab.
5. Choose the environment where your Data Hub cluster is running.
6. Click Download.
7. Save the keytab file in a chosen location.

   The following short video shows how you can retrieve the keytab file:

   ../images/csa-datahub-5-get-keytab-user.mp4
8. Upload your keytab file to the Streaming Analytics cluster.

```
scp <location>/<your_keytab_file> <workload_username>@<manager_node_FQDN
>:.
                    Password:<your_workload_password>
```

Create Atlas entity type definitions

As an administrator, you must create the Atlas entity type definitions before submitting a Flink job, and enable Atlas in Cloudera Manager to use Atlas for metadata management.

Before you begin

The Flink Atlas integration requires that users running Flink jobs have the privileges to write to the ATLAS_HOOK Kafka topic. You need to make sure that the ATLAS_HOOK has publish privileges in Ranger. In case this is not set, the metadata collection cannot be performed. For more information, see the Atlas documentation and the Grant permission for the ATLAS_HOOK topic documentation.

Procedure

1. Copy and paste the following command:

```
curl -k -u <atlas_admin>:<atlas_admin_pwd> --location --request
```

2. Provide your workload username and password as atlas admin.
3. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
4. Click Data Lake.
5. Click Endpoints.
6. Click on the copy icon beside the Atlas endpoint.
7. Replace the atlas server URL with the copied Atlas endpoint.

```
POST '<atlas_endpoint_url>/v2/types/typedefs'
```

8. Copy and paste the entity type definitions.

```
--header 'Content-Type: application/json' \
--data-raw '{
    "enumDefs": [],
    "structDefs": [],
    "classificationDefs": [],
    "entityDefs": [
        {
            "name": "flink_application",
            "superTypes": [
                "Process"
            ],
            "serviceType": "flink",
            "typeVersion": "1.0",
            "attributeDefs": [
                {
                    "name": "id",
                    "typeName": "string",
                    "cardinality": "SINGLE",
                    "isIndexable": true,
                    "isOptional": false,
                    "isUnique": true
                },
                {
                    "name": "startTime",
                    "typeName": "date",
                    "cardinality": "SINGLE",
                    "isIndexable": false,
                    "isOptional": true,
                    "isUnique": false
                },
                {
```

```
                    "name": "endTime",
                    "typeName": "date",
                    "cardinality": "SINGLE",
                    "isIndexable": false,
                    "isOptional": true,
                    "isUnique": false
                },
                {
                    "name": "conf",
                    "typeName": "map<string,string>",
                    "cardinality": "SINGLE",
                    "isIndexable": false,
                    "isOptional": true,
                    "isUnique": false
                }
            ]
        }
    ],
    "relationshipDefs": []
}'
```

The following short video shows how you can create the Atlas entity type definitons:

../images/csa-datahub-12-configuring-atlas.mp4

9. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
10. Click on the Streaming Analytics cluster.
11. Select Cloudera Manager UI from the Services.
12. Select Flink from the list of clusters.
13. Click Configuration.
14. Search for enable atlas in the search bar.
15. Make sure that Atlas is enabled in the configurations.

    The following short video shows how you can enable Atlas for Flink in Cloudera Manager:

    ../images/csa-datahub-16-configuring-atlas-cm.mp4

Running your Flink application

After preparing your environment, you need to connect Kafka to Flink, and create Kafka topics where the input and output data is messaged. When Kafka is ready, you need to generate data to your Kafka topic and let Flink apply the computations you have added in your application design.

Procedure

1. Create your streaming Flink application with Kafka as source and sink.

   The Stateful Tutorial has a detailed guide how to create your application using Kafka as source and sink. For more information, see the Setting up Kafka inputs and outputs section.

2. Build your Flink project with maven after creating the application logic.

```
mvn clean package
```

3. Upload your Flink project to the Streaming Analytics cluster.

```
scp <location>/flink-stateful-tutorial-1.2-SNAPSHOT.jar <your_workload_u
sername>@<manager_node_FQDN>:.
Password:<your_workload_password>
```

4. Upload your keytab file to the Streaming Analytics cluster, if you have not uploaded it yet.

```
scp <location>/<your_keytab_file> <your_workload_username>@<manager_node
_FQDN>:.
Password:<your_workload_password>
```

Important: You will not be able to submit Flink jobs on a Data Hub cluster if you do not upload your keytab file to the Streaming Analytics cluster.

5. Configure the Flink job properties to securely connect to the Streams Messaging cluster.

Using the job.properties file from the Stateful Tutorial, replace your kafka brokers and trustore location. The input and output topics have "flink" as a prefix in the following example to comply with the set Ranger policies.

```
kafka.bootstrap.servers=<your_kafka_broker>:9093,<your_kafka_broker1>:90
93,<your_kafka_broker2>:9093
kafka.group.id=flink
kafka.flink.partition-discovery.interval-millis=60000
kafka.retries=3
transaction.input.topic=flink.transaction.log.1
generate.queries=false
query.input.topic=flink.query.input.log.1
query.output.topic=flink.query.output.log.1
num.items=1000000
sleep=100

kafka.security.protocol=SASL_SSL
kafka.sasl.kerberos.service.name=kafka
kafka.ssl.truststore.location=/var/lib/cloudera-scm-agent/agent-cert/cm-
auto-global_truststore.jks
```

Note: You can use the following command to get the user related truststore location and password:

```
cat /etc/flink/conf/flink-conf.yaml | grep truststore
```

After running the command, truststore location and password is displayed in the command line.

To retrieve the Kafka broker hostnames:

a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
b. Click on the Streams Messaging cluster.
c. Click Hardware.
d. Search for the Kafka brokers.
e. Click the copy icon next to the Kafka brokers to obtain the hostnames.

Note: The job.properties file shown in the video contains JAAS configuration for secure Kafka connection. You do not need to add the JAAS configuration property as it is automatically generated, but you can overwrite the default setting by adding the configuration to the job.properties file shown in the video.

6. Upload the Flink job properties file to the Streaming Analytics cluster.

```
scp <location>/config/job.properties <your_workload_username>@<manager_n
ode_FQDN>:.
Password:<your_workload_password>
```

7. Use ls command to list and check if you have every necessary file on your Streaming Analytics cluster.

After listing the files on your cluster, you should be able to see your keytab file, the Stateful Tutorial jar file and the job.properties file.

8. Create your topics in Streams Messaging Manager (SMM).
   a) Go to Management Console > Data Hub Clusters.
   b) Search for your Streams Messaging cluster.
   c) Open SMM from the list of Services.
   d) Go to Topics > Add new.
   e) Provide information for Topic Name, Partitions, Availability, and Limits.

   Note: Make sure to give the topic its name based on the prefix previously provided for the Ranger policy.

   Create three topics with 16 partitions for each. You need to name them flink.tutorial.transaction.log, flink.tutorial.query.input.log, and flink.tutorial.query.input.log to comply with the Stateful Flink Application Tutorial.

9. Start generating data to your target Kafka topic.

   Important: You must pass the Kerberos keytab file and principal name to the Flink run command in the following format:

```
-yD security.kerberos.login.keytab=<your_keytab_filename> \
-yD security.kerberos.login.principal=<csso_name> \
```

   Submit the Data Generator job from the Stateful Flink Application Tutorial.

```
flink run -yD security.kerberos.login.keytab=<your_keytab_filename> \
-yD security.kerberos.login.principal=<csso_name> \
-m yarn-cluster -d -p 2 -ys 2 -ynm Datagenerator \
-c com.cloudera.streaming.examples.flink.KafkaDataGeneratorJob \
flink-stateful-tutorial-1.2-SNAPSHOT.jar job.properties
```

10. Open SMM and check if the data is generated.

    After waiting a few minutes, you can notice the Data in count increases as the generated data arrives to the Kafka topic. You can check the incoming data in the Data explorer tab. ../images/csa-datahub-9-generatorjob-smm.mp4

11. Deploy your Flink streaming application.

    Submit the Transaction Job from the Stateful Flink Application Tutorial.

```
flink run -yD security.kerberos.login.keytab=<your_keytab_filename> \
-yD security.kerberos.login.principal=<csso_name> \
-m yarn-cluster -d -p 2 -ys 2 -ynm TransactionProcessor \
flink-stateful-tutorial-1.2-SNAPSHOT.jar job.properties
```

12. Open SMM and check the log of your application.

    You can notice how the query data results are shown in SMM. ../images/csa-datahub-11-processorjob-smm.mp4

What to do next

You have the following options to monitor and manage your Flink applications:

Job monitoring with Flink Dashboard

After submitting a Flink job, you can always use the Flink Dashboard to review if the job submission was successful. Later you can use the Flink Dashboard to monitor the history of all your submitted and completed jobs.

You can access the Flink Dashboard directly from your Data Hub cluster.

1. Go to Management Console > Data Hub Clusters.
2. Search for your Streaming Analytics cluster.
3. Select Flink Dashboard from the list of Services.

You are redirected to the Flink Dashboard user interface where you can select from the submitted jobs.

For Data Generator Job

../images/csa-datahub-8-generatorjob-dashboard.mp4

For Transaction Processor Job

../images/csa-datahub-10-processorjob-dashboard.mp4

Note: You cannot save the Completed jobs into cloud storage.

Metadata governance with Atlas

You can use Atlas to find, organize and manage different assets of data about your Flink applications and how they relate to each other. This enables a range of data stewardship and regulatory compliance use cases.

Procedure

1. Go to Management Console > Data Lakes .
2. Search for your environment from the list of available environments.
3. Select Atlas from the list of services.
4. Search and select flink_application from the Search By Type bar.
5. Click the Name of your application.
6. Select Properties, Lineage, Relationships, Classifications or Audits tabs for more information about your application.

For Data Generator Job

../images/csa-datahub-13-using-atlas.mp4

For Transaction Processor Job

../images/csa-datahub-13-using-atlas2.mp4

Related Information

Flink metadata collection using Atlas

Analyzing your data with Kudu

Running your Flink application

After preparing your environment, you need to create a Kudu table, and also connect Flink with Kudu by providing the Kudu master. You also need to choose a source to which you connect Flink in Data Hub. After generating data to your source, Flink applies the computations you have added in your application design. The results are redirected to your Kudu sink.

Before you begin

- You have a CDP Public Cloud environment.

- You have a CDP username (it can be your own CDP user or a CDP machine user) and a password set to access Data Hub clusters.

  The predefined resource role of this user is at least EnvironmentUser. This resource role provides the ability to view Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the CDP Public Cloud environment.
- You have a Streaming Analytics cluster.
- You have a Real-time Data Mart cluster in the same Data Hub environment as the Streaming Analytics cluster.
- Your CDP user has the correct permissions set up in Ranger allowing access to Kudu.
- You obtained the Kudu Master hosts:

  1. Go to Management Console > Environments .
  2. Search for your environment from the list of available environments.
  3. Select the Data Hub cluster within your environment from the list of available clusters.
  4. Select Kudu Master from the list of Services.
  5. Click Masters.
  6. Copy the host information from the list of Live Masters.

Procedure

1. Create your Kudu tables:

   - Use Impala from the Real-time Data Mart cluster to create a table in Kudu.

     For more information about how to create a Kudu table from Impala, see the official documentation.
   - Use the KuduCatalog to create a table in Kudu.

```
KuduTableInfo tableInfo = KuduTableInfo
    .forTable("ExampleTable")
    .createTableIfNotExists(
        () ->
            Lists.newArrayList(
                new ColumnSchema
                    .ColumnSchemaBuilder("first", Type.INT32)
                    .key(true)
                    .build(),
                new ColumnSchema
                    .ColumnSchemaBuilder("second", Type.STRING)
                    .build()
            ),
        () -> new CreateTableOptions()
            .setNumReplicas(1)
            .addHashPartitions(Lists.newArrayList("first"), 2));

catalog.createTable(tableInfo, false);
```

2. Choose and add a source to your Flink application.

   Note: You can create your application choosing the Streams Messaging cluster that includes Kafka, Schema Registry and Streams Messaging Manager. For more information about Kafka as a source, see the Analyzing your data with Kafka use case.

3. Add Kudu as sink to your Flink application.

   The following code example shows how to build your application logic with a Kudu sink:

```
KuduWriterConfig writerConfig = KuduWriterConfig.Builder.setMasters(KUD
U_MASTERS).build();
KuduSink<Row> sink = new KuduSink<>(
    writerConfig,
    KuduTableInfo.forTable("AlreadyExistingTable"),
```

```
    new RowOperationMapper<>(
            new String[]{"col1", "col2", "col3"},
            AbstractSingleOperationMapper.KuduOperation.UPSERT)
)
```

4. Start generating data to your source connector.
5. Deploy your Flink streaming application.
6. Open the Kudu UI to see the generated table.

What to do next

You have the following options to monitor and manage your Flink applications:

Job monitoring with Flink Dashboard

After submitting a Flink job, you can always use the Flink Dashboard to review if the job submission was successful. Later you can use the Flink Dashboard to monitor the history of all your submitted and completed jobs.

You can access the Flink Dashboard directly from your Data Hub cluster.

1. Go to Management Console > Data Hub Clusters.
2. Search for your Streaming Analytics cluster.
3. Select Flink Dashboard from the list of Services.

You are redirected to the Flink Dashboard user interface where you can select from the submitted jobs.

Note: You cannot save the Completed jobs into cloud storage.

Metadata governance with Atlas

You can use Atlas to find, organize and manage different assets of data about your Flink applications and how they relate to each other. This enables a range of data stewardship and regulatory compliance use cases.

Procedure

1. Go to Management Console > Data Lakes .
2. Search for your environment from the list of available environments.
3. Select Atlas from the list of services.
4. Search and select flink_application from the Search By Type bar.
5. Click the Name of your application.
6. Select Properties, Lineage, Relationships, Classifications or Audits tabs for more information about your application.

Analyzing your data with HBase

Running your Flink application

After preparing your environment, you need to choose a source to which you connect Flink in Data Hub. After generating data to your source, Flink applies the computations you have added in your application design. The results are redirected to your HBase sink.

Before you begin

- You have a CDP Public Cloud environment.

- You have a CDP username (it can be your own CDP user or a CDP machine user) and a password set to access Data Hub clusters.

  The predefined resource role of this user is at least EnvironmentUser. This resource role provides the ability to view Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the CDP Public Cloud environment.
- You have a Streaming Analytics cluster.
- You have an Operational Database with SQL cluster in the same Data Hub environment as the Streaming Analytics cluster.
- Your CDP user has the correct permissions set up in Ranger allowing access to HBase.

Procedure

1. Choose a source for your Flink application and add the connector to your application.

   Note: You can create your application choosing the Streams Messaging cluster with Kafka, Schema Registry and Streams Messaging Manager. For more information about Kafka as a source, see the Analyzing your data with Kafka use case.
2. Add HBase as sink to your Flink application.

   The following code example shows how to build your application logic with an HBase sink:

```
HBaseSinkFunction<QueryResult> hbaseSink = new HBaseSinkFunction<QueryR
esult>("ITEM_QUERIES") {
@Override
public void executeMutations(QueryResult qresult, Context context, Buffer
edMutator mutator) throws Exception {
  Put put = new Put(Bytes.toBytes(qresult.queryId));
  put.addColumn(Bytes.toBytes("itemId"), Bytes.toBytes("str"),   Bytes.to
Bytes(qresult.itemInfo.itemId));
  put.addColumn(Bytes.toBytes("quantity"), Bytes.toBytes("int"), Bytes.
toBytes(qresult.itemInfo.quantity));
  mutator.mutate(put);
  }
};
hbaseSink.setWriteOptions(HBaseWriteOptions.builder()
  .setBufferFlushIntervalMillis(1000)
  .build()
);
streamqueryResultStream.addSink(hbaseSink);
```

3. Start generating data to your source connector.
4. Deploy your Flink streaming application.

What to do next

You have the following options to monitor and manage your Flink applications:

Job monitoring with Flink Dashboard

After submitting a Flink job, you can always use the Flink Dashboard to review if the job submission was successful. Later you can use the Flink Dashboard to monitor the history of all your submitted and completed jobs.

You can access the Flink Dashboard directly from your Data Hub cluster.

1. Go to Management Console > Data Hub Clusters.
2. Search for your Streaming Analytics cluster.
3. Select Flink Dashboard from the list of Services.

You are redirected to the Flink Dashboard user interface where you can select from the submitted jobs.

Note: You cannot save the Completed jobs into cloud storage.

Metadata governance with Atlas

You can use Atlas to find, organize and manage different assets of data about your Flink applications and how they relate to each other. This enables a range of data stewardship and regulatory compliance use cases.

Procedure

1. Go to Management Console > Data Lakes .
2. Search for your environment from the list of available environments.
3. Select Atlas from the list of services.
4. Search and select flink_application from the Search By Type bar.
5. Click the Name of your application.
6. Select Properties, Lineage, Relationships, Classifications or Audits tabs for more information about your application.

Cloudera Flow Management 7.3.1

Flow Management Overview

Date published: 2019-06-26
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What is Apache NiFi? **4**

What is Apache NiFi Registry? **4**

Cloudera Manager integration **4**

What is Apache NiFi?

Cloudera Flow Management is a no-code data ingestion and management solution powered by Apache NiFi.

Cloudera Flow Management includes two primary components, NiFi and NiFi Registry. NiFi serves as the core data ingestion engine, with a wide range of processors, facilitating seamless connectivity, transformation, and content routing. It enables easy data ingestion and movement between Cloudera services, as well as 3rd party cloud services. NiFi has a no-code graphical user interface and offers a user-friendly environment to design and manage data flows.

For more information about the fundamental concepts behind NiFi, see the *Apache NiFi Overview* available in the Apache NiFi documentation.

Important:

The Apache NiFi documentation is available both online and within the NiFi user interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi Overview

What is Apache NiFi Registry?

Cloudera Flow Management is a no-code data ingestion and management solution powered by Apache NiFi.

Cloudera Flow Management includes two primary components, NiFi and NiFi Registry. NiFi Registry, a companion to Apache NiFi, enables DevOps style development and deployment of flow files. It also supports flow versioning, promoting flows from one environment into another, and flow deployment. It is automatically set up when you create the Flow Management cluster and it serves as a centralized hub for managing your versioned data flows.

For more information about the fundamental concepts behind NiFi Registry, see the *Getting Started with Apache NiFi Registry* guide in the Apache NiFi Registry documentation.

Important:

The Apache NiFi Registry documentation is available both online and within the NiFi Registry user interface (UI). The online documentation consistently reflects the latest NiFi Registry version. If you are using an earlier NiFi Registry version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi Registry version directly from the NiFi Registry UI:

1. Launch the NiFi Registry UI.
2. From the Global Menu, select Help Documentation .

Related Information

Getting Started with Apache NiFi Registry

Cloudera Manager integration

The NiFi service of Cloudera Flow Management is integrated with Cloudera Manager on Cloudera Private Cloud Base.

Cloudera Manager monitoring

Cloudera Manager is retrieving monitoring data from NiFi by calling the below endpoint:

```
/nifi-api/flow/metrics/json?rootFieldName=beans&sampleName=.*_jvm_.*&sampleL
abelValue=RootProcessGroup
```

Metrics and graphs for the NiFi service are available starting with version Cloudera Flow Management 2.1.4 and it requires Cloudera Manager 7.4.5 or higher.

This endpoint is called by the Cloudera Manager agent running on each NiFi node and the identity of the node is used to make the call. If Ranger is used for authorization, the calls are authorized as the nodes belong to the nifi group.

Cloudera Manager alerting

Multiple health checks are configured out of the box for the NiFi service in Cloudera Manager:

- **NiFi node connection**

  It checks if the NiFi node is disconnected or not from the NiFi cluster. This can help to detect cluster instability in case the NiFi node is up and running but the node is not part of the NiFi cluster. The check executes the NiFi CLI `get-nodes` command on each NiFi node, which calls the corresponding NiFi API. If Ranger is used for authorization, the health check isauthorized as the nodes belong to the nifi group.
- **NiFi Content repository usage**

  It is used to check how much disk space is used on the Content repository.
- **NiFi Provenance repository usage**

  It is used to check how much disk space is used on the Provenance repository.
- **NiFi FlowFile repository usage**

  It is used to check how much disk space is used on the FlowFile repository.

Cloudera DataFlow for Data Hub 7.3.1

Configuring Flow Management clusters to hot load custom NARs

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring Flow Management clusters to hot load custom NARs.....4

Configuring Flow Management clusters to hot load custom NARs

Hot loading custom NARs means that you can store the NARs for your NiFi custom processors in an external folder, and load them from that directory for use on the NiFi canvas. This makes it easy to edit custom processors and add new ones without having to restart your Flow Management cluster.

Before you begin

- Collect your Kerberos information.
  1. In Cloudera Management Console, click Environments.
  2. From the Actions drop-down, click Manage Access.
  3. Create a Machine User with the EnvironmentUser Resource Role and set its Workload password.

     Note:

     It is best practice to have a dedicated Machine User in this role. This Machine User is in charge of accessing the object store resources for NiFi, having a dedicated user ensures that there is least privilege access and separation of duties.
  4. Note down the user you create there and password, for your safety valves.
- You have created an ID Broker mapping.
  1. Access IDBroker mappings.
     a. To access IDBroker mappings in your environment, click Actions Manage Access .
     b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
  2. Add your Cloudera Machine User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

     Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
  3. Click Save and Sync.

About this task

- You can configure NiFi to hot load custom NARs in all supported cloud providers.
- You can only access resources in the object store of the cloud provider in which you environment is running. But but you can have a development and a production environment, for example, and configure both environments to access the same location to hot load custom NARs
- It is possible to define multiple locations if desired by repeating the configuration properties with a different “[**custom-name**]” for each location. But all locations should be in the object store of the underlying cloud provider of where the NiFi cluster is located.
- NiFi polls for new NARs in the configured location(s) every 5 minutes.

Procedure

1. In Cloudera Management Console, select Data Hub Clusters, then go to the Flow Management cluster, then NiFi Service.
2. Click Configuration.
3. Search for Safety Valve and then Find: NiFi Node Advanced Configuration Snippet (Safety Valve) for staging/nifi.properties.xml

4. Create the following new properties:

```
Name: nifi.nar.library.provider.[**custom-name**].implementation
Example: nifi.nar.library.provider.narstorage1.implementation
Value: org.apache.nifi.nar.hadoop.HDFSNarProvider

Path to the configuration files
Name: nifi.nar.library.provider.[**custom-name**].resources
Example: nifi.nar.library.provider.narstorage1.resources
Value, by default: /etc/hadoop/conf.cloudera.core_settings/core-site.xml
Storage location for the NARs (the bucket name in S3 for example)
Name: nifi.nar.library.provider.[**custom-name**].storage.location
Example: nifi.nar.library.provider.narstorage1.storage.location
Value: s3a://my-custom-nars-bucket

Directory library with path to the NARs relative to the storage location
Name: nifi.nar.library.provider.[**custom-name**].source.directory
Example: nifi.nar.library.provider.narstorage1.source.directory
Value: /myNARs

Workload username and password of the machine user to use to access the re
sources
Name: nifi.nar.library.provider.[**custom-name**].kerberos.principal
Example: nifi.nar.library.provider.narstorage1.kerberos.principal
Value: srv_my_s3_nar_user

Name: nifi.nar.library.provider.[**custom-name**].kerberos.password
Example: nifi.nar.library.provider.narstorage1.kerberos.password
Value: myPassword123
```

Note:

Cloudera recommends that the workload user name and password that you configure above is strictly dedicated to accessing custom NARs, and that the only permissions granted to this user are to access the custom NAR object store location.

Alternately, you can also use the keytab file. However, this file must be distributed on all NiFi nodes. This becomes problematic when scaling up a NiFi cluster in DataHub. In addition, the file permissions must be properly changed to ensure the file is readable by the `nifi` user. However, if desired, the below configuration can be used:

```
Name: nifi.nar.library.provider.[**custom-name**].kerberos.keytab
Example: nifi.nar.library.provider.narstorage1.kerberos.keytab
Value: /path/to/my/file.keytab
```

5. Restart the NiFi service in the Cloudera Manager.

Results

Refresh the NiFi UI and then drag and drop a new processor to NiFi canvas and search for your custom processor.

Cloudera DataFlow for Data Hub 7.3.1

Using Apache Flink

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Running a simple Flink application 5

Application development 6
- Flink application structure 6
  - Source, operator and sink in DataStream API 6
  - Flink application example 9
  - Testing and validating Flink applications 11
- Flink Project Template 12
- Configuring Flink applications 14
  - Setting parallelism and max parallelism 14
  - Configuring Flink application resources 14
  - Configuring state backend 16
  - Enabling checkpoints for Flink applications 19
  - Configuring PyFlink applications 20

DataStream connectors 21
- HBase sink with Flink 21
  - Creating and configuring the HBaseSinkFunction 21
- Kafka with Flink 22
  - Schema Registry with Flink 24
  - Kafka Metrics Reporter 27
- Kudu with Flink 28
- Iceberg with Flink 28

File systems 30

Job lifecycle 31
- Setting up Python for PyFlink 31
- Running a Flink job 32
- Using Flink CLI 34
- Enabling savepoints for Flink applications 35

Monitoring 35
- Enabling Flink DEBUG logging 35
- Flink Dashboard 36
- Streams Messaging Manager integration 37

Flink SQL and Table API 37
- SQL and Table API supported features 38
- DataStream API interoperability 39
  - Converting DataStreams to Tables 39
  - Converting Tables to DataStreams 40
  - Supported data types 40

SQL catalogs for Flink....41
Hive catalog....42
Kudu catalog....42
Schema Registry catalog....43
SQL connectors for Flink....43
SQL Statements in Flink....49
CREATE Statements....49
DROP Statements....51
ALTER Statements....51
INSERT Statements....51
SQL Queries in Flink....52

Flink metadata collection using Atlas....52
Atlas entities in Flink metadata collection....53
Creating Atlas entity type definitions for Flink....53
Verifying metadata collection....55

Migrating Flink jobs....55
Migrating Flink jobs without state....56
Migrating stateful Flink jobs....56
Updating Flink job dependencies....58

Reference....58
Flink Terminology....58
Cloudera Flink Tutorials....59

Running a simple Flink application

In this example, you will use the Stateless Monitoring Application from the Flink Tutorials to build your Flink project, submit a Flink job and monitor your Flink application using the Flink Dashboard in an unsecured environment.

Before you begin

- You have a Cloudera Public Cloud environment.
- You have a Cloudera username (it can be your own Cloudera user or a Cloudera machine user) and a password set to access Cloudera Data Hub clusters.

  The predefined resource role of this user is at least EnvironmentUser. This resource role provides the ability to view Cloudera Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the Cloudera Public Cloud environment.
- You have a Cloudera Streaming Analytics cluster.

Procedure

1. Clone the simple tutorial from git:

```
git clone https://github.com/cloudera/flink-tutorials.git
```

2. Access the simple tutorial folder:

```
cd flink-tutorials/flink-simple-tutorial
```

3. Build your Flink project using maven:

```
mvn clean package
```

4. Upload the Flink project to your cluster.

```
scp <location>/flink-stateful-tutorial-1.2-SNAPSHOT.jar <your_workload_u
sername>@<master_node_FQDN>:.
Password:<your_workload_password>
```

   Note: Provide your password when prompted.

5. Run the Flink application:

```
flink run -d -p 2 -ynm HeapMonitor target/flink-simple-tutorial-1.2-SNAP
SHOT.jar
```

6. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
7. Select the Cloudera Streaming Analytics cluster.
8. Click Flink Dashboard from the services.
   The Flink Dashboard opens in a new window.
9. Click Task Manager on the left side menu.
10. Monitor your Flink application under logs.

Application development

Flink application structure

You must understand the parts of application structure to build and develop a Flink streaming application. To create and run the Flink application, you need to create the application logic using the DataStream API.

A Flink application consists of the following structural parts:

- Creating the execution environment
- Loading data to a source
- Transforming the initial data
- Writing the transformed data to a sink
- Triggering the execution of the program

StreamExecutionEnvironment

The getExecutionEnvironment() static call guarantees that the pipeline always uses the correct environment based on the location it is executed on. When running from the IDE, a local execution environment, and when running from the client for cluster submission, it returns the YARN execution environment. The rest of the main class defines the application sources, processing flow and the sinks followed by the execute() call. The execute call triggers the actual execution of the pipeline either locally or on the cluster. The StreamExecutionEnvironment class is needed to configure important job parameters for maintaining the behavior of the application and to create the DataStream.

Related Information

Flink Project Template

Simple Tutorial: Application logic

Stateful Tutorial: Build a Flink streaming application

Apache Flink documentation: DataStream API overview

Source, operator and sink in DataStream API

A DataStream represents the data records and the operators. There are pre-implemented sources and sinks for Flink, and you can also use custom defined connectors to maintain the dataflow with other functions.

For Java

```
DataStream<String> source = env
        .fromSource(
                kafkaSource,
                WatermarkStrategy.noWatermarks(),
                "Kafka Source")
        .uid("kafka-source")
        .map(record -> record.getId()
                + "," + record.getName()
                + "," + record.getDescription())
        .name("To Output String")
        .uid("to-output-string");

FileSink<String> sink = FileSink
        .forRowFormat(
                new Path(params.getRequired(K_HDFS_OUTPUT)),
                new SimpleStringEncoder<String>("UTF-8"))
        .build();
```

```
source.sinkTo(sink)
  .name("FS Sink")
  .uid("fs-sink");
source.print();
```

For Python

```
source = env.from_source(source=kafka_source,
                         watermark_strategy=WatermarkStrategy.no_watermark
s(),
                         source_name='Kafka Source')
source = source.map(lambda record:
                    record.get_id() + ',' +
                    record.get_name() + ',' +
                    record.get_description(),
                    output_type=Types.STRING())
source = source.name('To Output String').uid('to-output-string')

sink = FileSink.for_row_format(
    base_path=output_path,
    encoder=Encoder.simple_string_encoder())

source.sink_to(sink).name('FS Sink').uid('fs-sink')
source.print()
```

Choosing the sources and sinks depends on the purpo se of the application. As Flink can be implemented in any kind of an environment, various connectors are available. In most cases, Kafka is used as a connector as it has streaming capabilities and can be easily integrated with other services.

Sources

Sources are where your program reads its input from. You can attach a source to your program by using StreamExecutionEnvironment.addSource(sourceFunction). Flink comes with a number of pre-implemented source functions. For the list of sources, see the Apache Flink documentation.

Cloudera Streaming Analytics supports the following sources:

- HDFS
- Kafka

Operators

Operators transform one or more DataStreams into a new DataStream. When choosing the operator, you need to decide what type of transformation you need on your data. The following are some basic transformation:

- Map

  Takes one element and produces one element.

  For Java

  ```
  dataStream.map(new MapFunction<Integer, Integer>() {
      @Override
      public Integer map(Integer value) throws Exception {
          return 2 * value;
      }
  ```

```
});
```

For Python

```
data_stream.map(lambda value: 2 * value, output_type=Types.I
NT())
```

- FlatMap

  Takes one element and produces zero, one, or more elements.

  For Java

```
dataStream.flatMap(new FlatMapFunction<String, String>() {
    @Override
    public void flatMap(String value, Collector<String> out)
        throws Exception {
        for (String word: value.split(" ")) {
            out.collect(word);
        }
   }
});
```

  For Python

```
data_stream.flat_map(lambda value: value.split(' '), output_
type=Types.STRING())
```

- Filter

  Evaluates a boolean function for each element and retains those for which the function returns true.

  For Java

```
dataStream.filter(new FilterFunction<Integer>() {
    @Override
    public boolean filter(Integer value) throws Exception {
        return value != 0;
    }
});
```

  For Python

```
data_stream.filter(lambda value: value != 0)
```

- KeyBy

  Logically partitions a stream into disjoint partitions. All records with the same key are assigned to the same partition. This transformation returns a KeyedStream

  For Java

```
dataStream.keyBy(value -> value.getSomeKey());
dataStream.keyBy(value -> value.f0);
```

  For Python

```
data_stream.key_by(lambda value: value.get_some_key())
```

```
data_stream.key_by(lambda value: value[0], key_type=Types
.STRING())
```

- Window

  Windows can be defined on already partitioned KeyedStreams. Windows group the data in each key according to some characteristic (for example, the data that arrived within the last 5 seconds).

  For Java

  ```
  dataStream
      .keyBy(value -> value.f0)
      .window(TumblingEventTimeWindows.of(Time.seconds(5)));
  ```

  For Python

  ```
  data_stream.key_by(lambda value: value[0], key_type=Types.ST
  RING()) \
              .window(TumblingEventTimeWindows.of(Time.minute
  s(1)))
  ```

For the full list of operators, see the Apache Flink documentation.

Sinks

Data sinks consume DataStreams and forward them to files, sockets, external systems, or print them. Flink comes with a variety of built-in output formats that are encapsulated behind operations on the DataStreams. For the list of sources, see the Apache Flink documentation.

Cloudera Streaming Analytics supports the following sinks:

- Kafka
- HBase
- Kudu
- HDFS

Related Information

Apache Flink documentation: Operators

Apache Flink documentation: Window operator

Apache Flink documentation: Generating watermarks

Apache Flink documentation: Working with state

Apache Flink documentation: User defined functions

Flink application example

The following is an example of a Flink application about a streaming window word count that counts worfs from a web socket in five second windows.

For Java

```
import org.apache.flink.api.common.functions.FlatMapFunction;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironme
nt;
import org.apache.flink.streaming.api.windowing.time.Time;
import org.apache.flink.streaming.api.windowing.assigners.TumblingProcess
ingTimeWindows;
import org.apache.flink.util.Collector;
```

```
public class WindowWordCount {
    public static void main(String[] args) throws Exception {

        StreamExecutionEnvironment env =
            StreamExecutionEnvironment.getExecutionEnvironment();

        DataStream<Tuple2<String, Integer>> dataStream = env
            .socketTextStream("localhost", 9999)
            .flatMap(new Splitter())
            .keyBy(value -> value.f0)
            .window(TumblingProcessingTimeWindows.of(Time.seconds(5)))
            .sum(1);

        dataStream.print();

        env.execute("Window WordCount");
    }

    public static class Splitter
        implements FlatMapFunction<String, Tuple2<String, Integer>> {

        @Override
        public void flatMap(String sentence,
                            Collector<Tuple2<String, Integer>> out)
            throws Exception {

            for (String word: sentence.split(" ")) {
                out.collect(new Tuple2<String, Integer>(word, 1));
            }
        }
    }
}
```

For Python

```
import logging
import sys

from pyflink.common import Types
from pyflink.datastream import StreamExecutionEnvironment, RuntimeExecut
ionMode

word_count_data = [
    "Lorem ipsum dolor sit amet, consectetur adipiscing elit,",
    " sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.",
    " Eget est lorem ipsum dolor sit amet.",
    " Venenatis lectus magna fringilla urna porttitor rhoncus dolor purus
non.",
    " Scelerisque felis imperdiet proin fermentum leo vel orci porta non.
",
    " A iaculis at erat pellentesque adipiscing commodo elit at.",
    " Bibendum neque egestas congue quisque egestas.",
    " Massa enim nec dui nunc. Tellus mauris a diam maecenas sed enim ut
 sem.",
    " Mauris in aliquam sem fringilla ut morbi tincidunt augue interdum.",
    " Eget sit amet tellus cras adipiscing enim. Amet porttitor eget dolor
 morbi non.",
    " Lacus suspendisse faucibus interdum posuere lorem ipsum.",
    " Diam phasellus vestibulum lorem sed risus ultricies.",
    " Nulla facilisi nullam vehicula ipsum a arcu.",
    " Diam in arcu cursus euismod quis. Tempor commodo ullamcorper a lacus
 vestibulum."]
```

```
def word_count():
    env = StreamExecutionEnvironment.get_execution_environment()
    env.set_runtime_mode(RuntimeExecutionMode.BATCH)


    ds = env.from_collection(word_count_data)

    def split(line):
        yield from line.split()
    ds = ds.flat_map(split) \
        .map(lambda i: (i, 1), output_type=Types.TUPLE([Types.STRING(), T
ypes.INT()])) \
        .key_by(lambda i: i[0]) \
        .reduce(lambda i, j: (i[0], i[1] + j[1]))
    ds.print()
    env.execute()

if __name__ == '__main__':
    logging.basicConfig(stream=sys.stdout, level=logging.INFO, format="%
(message)s")

    word_count()
```

Testing and validating Flink applications

After you have built your Flink streaming application, you can create a simple testing method to validate the correct behaviour of your application.

Pipelines can be extracted to static methods and can be easily tested with the JUnit framework.

A simple JUnit test can be written to verify the core application logic. The test is implemented in the test class and should be regarded as an integration test of the application flow.

The test mimics the application main class with only minor differences:

1. Create the StreamExecutionEnvironment the same way.
2. Use the env.fromElements(..) method to pre-populate a DataStream with some testing data.
3. Feed the testing data to the static data processing logic as before.
4. Verify the correctness once the test is finished.

```
@Test
    public void testPipeline() throws Exception {
        final String alertMask = "42";
        StreamExecutionEnvironment env = StreamExecutionEnvironment.getExec
utionEnvironment();
        HeapMetrics alert1 = testStats(0.42);
        HeapMetrics regular1 = testStats(0.452);
        HeapMetrics regular2 = testStats(0.245);
        HeapMetrics alert2 = testStats(0.9423);

        DataStreamSource<HeapMetrics> testInput = env.fromElements(alert1,
 alert2, regular1, regular2);
        HeapMonitorPipeline.computeHeapAlerts(testInput, ParameterTool.fro
mArgs(new String[]{"--alertMask", alertMask}))
                .addSink(new SinkFunction<HeapAlert>() {
                    @Override
                    public void invoke(HeapAlert value) {
                        testOutput.add(value);
                    }
                })
                .setParallelism(1);
        env.execute();
```

```
        assertEquals(Sets.newHashSet(HeapAlert.maskRatioMatch(alertMask, al
ert1),
                HeapAlert.maskRatioMatch(alertMask, alert2)), testOutput);
    }
    private HeapMetrics testStats(double ratio) {
        return new HeapMetrics(HeapMetrics.OLD_GEN, 0, 0, ratio, 0, "test
host");
    }
}
```

Related Information

Simple Tutorial: Testing the data pipeline

Stateful Tutorial: Test and validate the streaming pipeline

Flink Project Template

The Quickstart Archetype serves as a template for an Apache Flink streaming application. You can use the Archetype to add source, sink and computation to the template. Like this you can practice the development of a simple Flink application, or use the Archetype as the starting point for a more complex application including state,watermark and checkpoint.

About this task

The Flink quickstart archetype can be used to quickly build a basic Flink streaming project.

Note: You need to install the archetype locally on your host as Cloudera does not release maven archetypes to the Maven Central Repository.

Procedure

1. Perform the following commands to create the archetype locally:

```
git clone https://github.com/cloudera/flink-tutorials
cd flink-tutorials
cd flink-quickstart-archetype
git checkout CSA-1.13.0.0
mvn clean install
cd ..
```

Note: The Flink Tutorials repository contains version branches between the CDP Private Cloud Base and CDP Public Cloud deployments of CSA. The branches that use only CSA as a prefix belong to the CDP Private Cloud deployment, and the branches that use CSA-DH as a prefix belong to the CDP Public Cloud Data Hub deployment. Make sure to checkout the branch that is specific for the deployment in which CSA is used.

The following entry should be seen in your local catalog:

```
cat ~/.m2/repository/archetype-catalog.xml
 ...
 <archetypes>
    <archetype>
      <groupId>com.cloudera.flink</groupId>
      <artifactId>flink-quickstart-archetype</artifactId>
      <version>TBA</version>
      <description>flink-quickstart-archetype</description>
    </archetype>
  </archetypes>
```

2. Generate the project skeleton with the following commands:

```
mvn archetype:generate \
      -DarchetypeGroupId=com.cloudera.flink \
      -DarchetypeArtifactId=flink-quickstart-archetype \
      -DarchetypeVersion=TBA
```

3. Provide basic information about the streaming application.

   You can choose to customize configurations yourself or use automatically generated information.

   - To use automatically generated information, press Enter.
   - If you choose to customize configuration, set the following properties:

```
Define value for property 'groupId': com.cloudera.flink
                                Define value for property 'artifactId':
sample-project
                                Define value for property 'version' 1.0-
SNAPSHOT: :
                                Define value for property 'package' c
om.cloudera.flink: :
                                Confirm properties configuration:
                                groupId: com.cloudera.flink
                                artifactId: sample-project
                                version: 1.0-SNAPSHOT
                                package: com.cloudera.flink
                                Y: :
```

   The generated project will look like this:

```
sample-project
### pom.xml
### src
    ### main
        ### java
        #   ### com
        #       ### cloudera
        #           ### flink
        #               ### StreamingJob.java
        ### resources
            ### log4j.properties
```

4. Open StreamingJob.java file.

   The archetype application will look like this:

```
public class StreamingJob {
 public static void main(String[] args) throws Exception {
  final StreamExecutionEnvironment env = StreamExecutionEnvironment.getEx
ecutionEnvironment();
  DataStream<Integer> ds = env.fromElements(1,2,3,4);
  ds.printToErr();
  env.execute("Flink Streaming Java API Skeleton");
 }
}
```

5. Customize the archetype application by adding source, stream transformation and sink to the Datastream class.
6. Run the application with env.execute command.

Results

You have built your Flink streaming project.

What to do next

You can further develop your application, or you can run the Flink application archetype.

Configuring Flink applications

Cloudera Streaming Analytics includes Flink with configuration that works out of the box. It is not mandatory to configure Flink to production, but you can use the available configurations to optimize the application behavior in production. Cloudera Manager includes all the necessary configurations for Flink that can also be accessed from the flink-conf.yaml file.

Setting parallelism and max parallelism

The max parallelism is the most essential part of resource configuration for Flink applications as it defines the maximum jobs that are executed at the same time in parallel instances. However, you can optimize max parallelism in case your production goals differ from the default settings.

In a Flink application, the different tasks are split into several parallel instances for execution. The number of parallel instances for a task is called parallelism. Parallelism can be defined at the operator, client, execution environment and system level. Cloudera recommends setting parallelism to a lower value at first use, and increasing it over time if the job cannot keep up with the input rate.

To configure the max parallelism, setMaxParallelism is called as it controls the number of key-groups created by the state backends. A key-group is a partition of an operator state. The number of key-groups determines how data is going to be distributed among the parallel operators. If the key-groups are not distributed evenly, the data distribution is also uneven.

Consider the following aspects when setting the max parallelism:

- The number should be large enough to accommodate expected future load increases as this setting cannot be changed without starting from an empty state.
- If P is the selected parallelism for the job, the max parallelism should be divisible by P to get even state distribution.
- Please note that larger max parallelism settings have greater cost on the state backend side, for large scale production jobs benchmarking the size of the state based on the maximum parallelism is useful before changing this parameter.

Based on these criteria, Cloudera recommends setting the max paralellism to factorials or other numbers with a large number of divisors (120, 180, 240, 360, 720, 840, 1260), which will make parallelism tuning easier.

Table 1: Reference values

Stateless	In-memory state	RocksDB state
1 million record / sec / core	100 000 records / sec / core	10 000 records / sec / core

Configuring Flink application resources

Generally, Flink automatically identifies the required resources for an application based on the parallelism settings. However, you can adjust the configurations based on your requirements by specifying the number of task managers and their memory allocation for individual Flink applications or for the entire Flink deployment.

To control the resources of individual TaskManager processes and the amount of work allocated to them, Cloudera recommends starting the configuration with the following options:

Number of Task Slots

The number of task slots controls how many parallel pipeline/operator instances can be executed in a single TaskManager. Together with the parallelism setting, you can ultimately define how many TaskManagers will be allocated for the job. For example, if you set the job parallelism to 12 and the taskmanager.numberOfTaskSlots to 4, there will be 3 TaskManager containers for the job as the value of parallelism will be divided with the number of task slots.

You can set the number of task slots in Cloudera Manager under the Configuration tab.

TaskManager Process Memory Size

The taskmanager.memory.process.size option controls the total memory size of the TaskManager containers. For applications that store data on heap or use large state sizes, it is recommended towe should increase the process size accordingly. You can set the number of task slots in Cloudera Manager under the Configuration tab.

For more information about the TaskManager memory management, see the Apache Flink documentation. You can also check the TaskManager configuration of your running application on the Flink Dashboard to review the configured values before making adjustments.

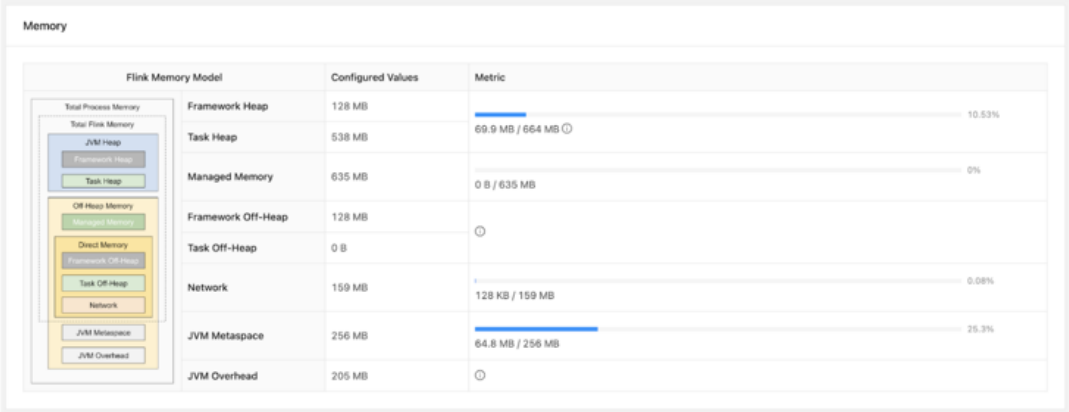

Network buffers for throughput and latency

Flink uses network buffers to transfer data from one operator to another. These buffers are filled up with data during the specified time for the timeout. In case of high data rates, the set time is usually never reached. For cases when the data rate is high, the throughput can be further increased with setting the buffer timeout to an intentionally higher value due to the characteristics of the TCP channel. However, this in turn increases the latency of the pipeline.

Yarn Related Configurations

Flink on YARN jobs are configured to tolerate a maximum number of failed containers before they terminate. You can configure the YARN maximum failed containers setting in proportion to the total parallelism and the expected lifetime of the job.

High Availability is enabled by default in Cloudera Streaming Analytics. This eliminates the JobManager as a single point of failure. You can also tune the application resilience by setting the YARN maximum application attempts, which determines how many times the application will retry in case of failures.

Furthermore, you can use a YARN queue with preemption disabled to avoid long running jobs being affected when the cluster reaches its capacity limit.

Reference values for the configurations

Configuration	Parameter	Recommended value
TM container memory	-ytm / taskmanager.heap.size	*TM HEAP + HEAP-CUTOFF*
Managed Memory Fraction	taskmanager.memory.managed.fraction	*0.4 - 0.9*
Max parallelism	pipeline.max-parallelism	*120,720,1260,5040*
Buffer timeout	execution.buffer-timeout	*1-100*
YARN queue	-yqu	*A QUEUE WITH NO PREEMPTION*
YARN max failed containers	yarn.maximum-failed-containers	*3*NUM_CONTAINERS*
YARN max AM failures	yarn.application-attempts	*3-5*

Configuring state backend

You can choose between RocksDB and Hashmap as a state backend for your Flink streaming application. While Hashmap stores data as an object on Java heap, RocksDB can be used to store a larger state that does not fit easily in memory. The RocksDB state backend uses a combination of fast in-memory cache and optimized disk based lookups to manage state.

The following state backend options are available for your Flink application:

RocksDB

By default, RocksDB is configured as the state backend for Flink. The RocksDB state backend holds in-flight data in a RocksDB database that is stored in the TaskManager local data directories and performs asynchronous snapshots. The data is stored as serialized byte arrays. The arrays are defined by the type serializer. This results in key comparisons being byte-wise instead of the hash Code() and equals() Java methods in the Hashmap state backend. The RocksDB state backend is recommended for large state and key/value states, long windows and high-availability setups. The amount of state is limited by the amount of disk space available. This also means that the maximum throughput that can be achieved will be lower as all reads/writes through the RocksDB backend have to go through deserialization and serialization.

Hashmap

The Hashmap state backend holds data internally as objects on the Java heap. The key/value state and window operators hold hash tables that store the values, triggers and so on. The Hashmap state backend is recommended for moderate state and key/value states, long windows and high-availability setups.

You can choose between Hashmap and RocksDB as the state backend based on your performance and scalability requirements. State can be accessed and updated faster with Hashmap on the Java heap, but the size is limited by the available memory in the cluster. Even though RocksDB can scale based on available disk space and is the only state backend to support incremental snapshots, accessing and updating the state requires more resources that can affect the performance compared to storing the state in-memory. For more information about the State backends in Flink, see the Apache Flink documentation.

You can configure the state backend for your streaming application using Cloudera Manager or you can also specify the state backend in your Flink application.

For Cloudera Manager

1. Open your cluster in Cloudera Manager.
2. Select Flink from the list of services.
3. Click Configuration.

4. Search for state.backend and select HASHMAP or ROCKSDB based on your requirements. ROCKSDB is set by default.

5. Click Save changes.

When configuring the state backend in Cloudera Manager, the configuration serves as a default global setting and is applied for every Flink job. You can override the default set in Cloudera Manager by specifying the state backend in your Flink application.

For Flink application

The per-job state backend can be configured by setting the env.setStateBackend to HashMapState Backend or EmbeddedRocksDBStateBackend on the StreamExecutionEnvironment of the job as shown in the following example:

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getE
xecutionEnvironment();
env.setStateBackend(new EmbeddedRocksDBStateBackend());
```

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getE
xecutionEnvironment();
env.setStateBackend(new HashMapStateBackend());
```

Note: In case you use Hashmap as state backend, it is recommended to switch to RocksDB when a job fails with the following exception:

```
Caused by: java.lang.OutOfMemoryError: GC overhead limit exceeded
```

Configuring RocksDB

When choosing RocksDB as state backend, you can also adjust how much memory RocksDB should use as a cache to increase lookup performance by setting the memory managed fraction of the TaskManagers.

The default fraction value is 0.4, but with larger cache requirements you need to increase this value together with the total memory size. For more information, see the Set up TaskManager Memory and the Memory Tuning Guide documentations:

For Cloudera Manager

1. Open your cluster in Cloudera Manager.
2. Select Flink from the list of services.
3. Click Configuration.
4. Search for taskmanager.memory.managed.fraction and change the default value as needed.

5. Click Save changes.

For Flink application

You can also configure the Taskmanager Memory when submitting the Flink job in CLI as shown in the following examples:

- Per-job mode:

```
flink run -yD taskmanager.memory.managed.fraction=<fraction_
size> -t
              <job_jar_file>
```

- Session mode:

```
flink-yarn-session -yD
          taskmanager.memory.managed.fraction=<fraction_size>
```

- Application mode:

```
flink run-application -t yarn-application
        -Dtaskmanager.memory.managed.fraction=<fraction_size>
<job_jar_file>
```

RocksDB also has predefined options that can be changed based on your system configurations:

- DEFAULT: option set by default and applicable for all settings.
- SPINNING_DISK_OPTIMIZED: option for regular spinning disks configured for better performance. The option is configured with the following values:
  - CompactionStyle=LEVEL
  - LevelCompactionDynamicLevelBytes=true
  - MaxBackgroundJobs=4
  - MaxOpenFiles=-1
- SPINNING_DISK_OPTIMIZED_HIGH_MEM: option for regular spinning hard disks configured for better performance, but with higher memory consumption. The option is configured with the following values:
  - BlockCacheSize=256 MBytes
  - BlockSize=128 KBytes
  - FilterPolicy=BloomFilter
  - LevelCompactionDynamicLevelBytes=true
  - MaxBackgroundJobs=4
  - MaxBytesForLevelBase=1 GByte
  - MaxOpenFiles=-1
  - MaxWriteBufferNumber=4
  - MinWriteBufferNumberToMerge=3
  - TargetFileSizeBase=256 MBytes
  - WriteBufferSize=64 MBytes
- FLASH_SSD_OPTIMIZED: option for flash SSDs configured for better performance. The following additional options are set:
  - MaxBackgroundJobs=4
  - MaxOpenFiles=-1

You can set the Rocksdb predefined options in the following ways:

For Cloudera Manager

1. Open your cluster in Cloudera Manager.
2. Select Flink from the list of services.
3. Click Configuration.

4. Search for state.backend.rocksdb.predefined-options and change the default value as needed.

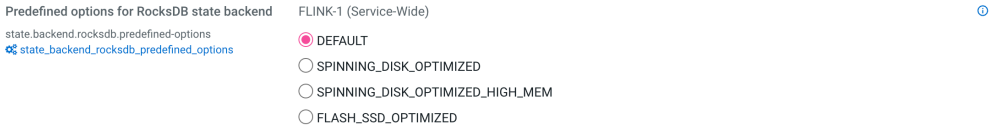

5. Click Save changes.

For Flink application

Update the EmbeddedRocksDBStateBackend for the StreamExecutionEnvironment of the Flink job as shown in the following example:

```
EmbeddedRocksDBStateBackend.setPredefinedOptions(PredefinedOptio
ns.SPINNING_DISK_OPTIMIZED_HIGH_MEM)
```

Enabling checkpoints for Flink applications

To make your Flink application fault tolerant, you need to enable automatic checkpointing. When an error or a failure occurs, Flink will automatically restarts and restores the state from the last successful checkpoint. Checkpointing is not enabled by default.

While it is possible to enable checkpointing programmatically through the StreamExecutionEnvironment, Cloudera recommends to enable checkpointing either using the configuration file for each job, or as a default configuration for all Flink applications through Cloudera Manager.

To enable checkpointing, you need to set the execution.checkpointing.interval configuration option to a value larger than 0. It is recommended to start with a checkpoint interval of 10 minutes (600000 milliseconds).

You can acces the configuration options of checkpointing in Cloudera Manager under the Configuration tab.

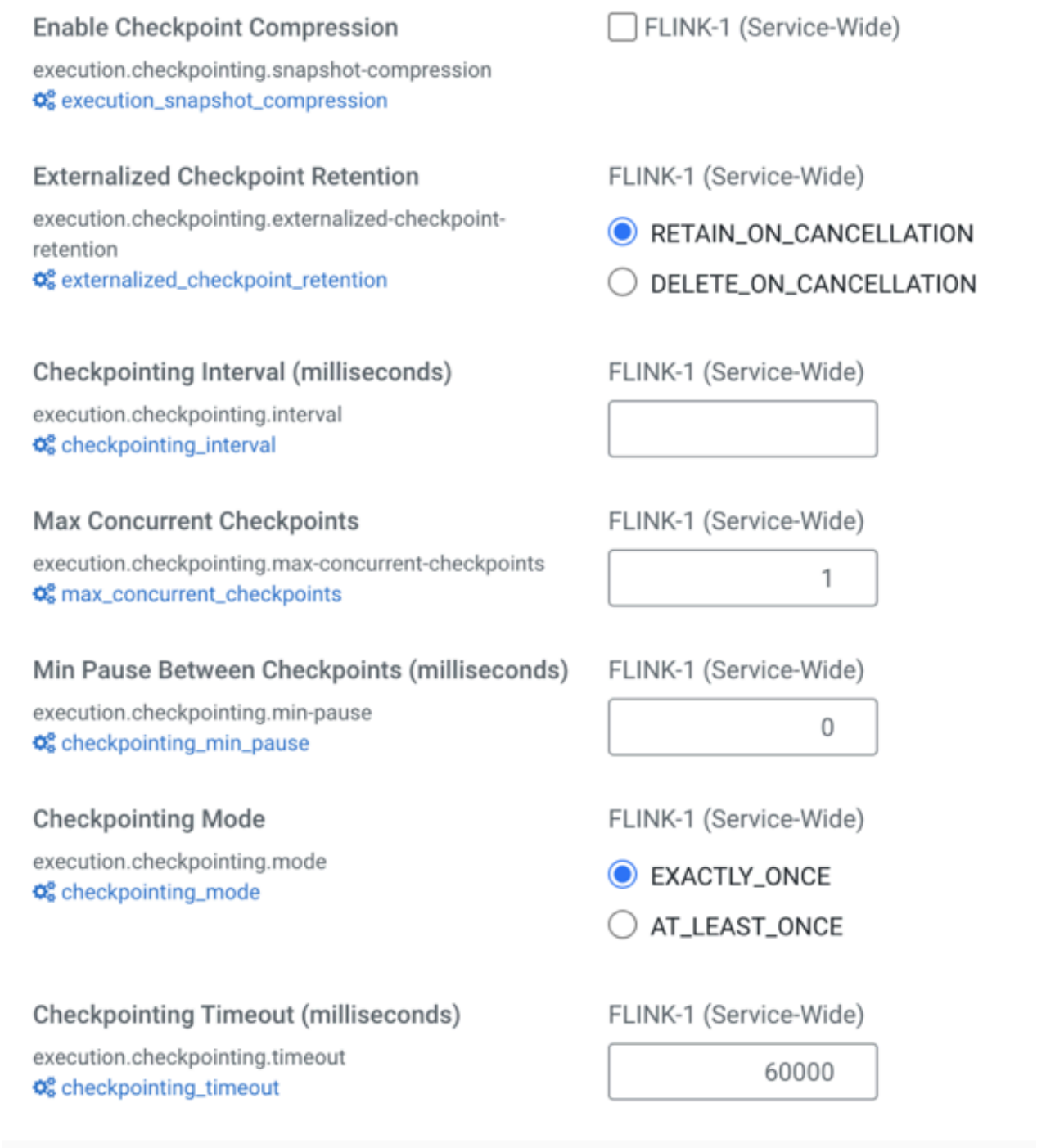

Configuring PyFlink applications

Based on the requirements of your PyFlink application, you can optimize the program with different configuration options.

You can configure the Flink application with Python API by setting the options as shown in the following example:

```
from pyflink.common import Configuration
from pyflink.datastream import StreamExecutionEnvironment

config = Configuration()
config.set_integer("python.fn-execution.bundle.size", 1000)
```

```
env = StreamExecutionEnvironment.get_execution_environment(config)
```

The configuration you set in the Flink application affects the Python runtime and how the different processes should be executed. The Flink job related configurations should be set when submiting the Flink job. For the list of Pything configurations, see the Python options section in the Apache Flink documentation. For more information about submitting a Flink job with options, see the Submitting PyFlink jobs section in the Apache Flink documentation.

DataStream connectors

HBase sink with Flink

Cloudera Streaming Analytics offers HBase connector as a sink. Like this you can store the output of a real-time processing application in HBase. You must develop your application defining HBase as sink and add HBase dependency to your project.

The HBase Streaming connector has the following key features:

- Automatic configuration on Cloudera
- High throughput buffered operations
- Customizable data-driven update/delete logic

To use the HBase integration, add one of the following dependencies to your project:

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-hbase-1.4</artifactId>
  <version>3.0-csa1.13.0.0</version>
</dependency>
```

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-hbase-2.4</artifactId>
  <version>3.0-csa1.13.0.0</version>
</dependency>
```

The general purpose HBase sink connector is implemented in the org.apache.flink.addons.hbase.HBaseSinkFunction class.

This is an abstract class that must be extended to define the interaction logic (mutations) with HBase. By using the BufferedMutator instance, you can implement arbitrary data driven interactions with HBase. While it is possible to run all mutations supported by the BufferedMutator interface, Cloudera strongly recommends that users should only use idempotent mutations: Put and Delete.

Creating and configuring the HBaseSinkFunction

You must configure the HBaseSinkFunction with Table names to have HBase as a sink. The HBase table needs to be created before the streaming job is submitted. You should also configure the operation buffering parameters to make sure that every data coming from Flink is buffered into HBase.

The HBase sink instance is always created as a subclass of the HBaseSinkFunction. When users create the subclass they have to provide required and optional parameters through the constructor of the superclass, the HBaseSinkFun ction itself.

Required parameters:

- Table name (the table itself must be created before the streaming job starts)

Optional parameters:

- Hadoop Configuration object for setting up the HBase client
- HBaseOptions for minimal connection configuration

The optional parameters are configured automatically by the Cloudera platform and should only be used for setting up custom HBase connections.

Important: The Flink Gateway node should also be an HBase Gateway node for the automatic configuration to work in the Cloudera environment.

To configure the operation buffering parameters, you need to use the HBaseSinkFunction.setWriteOptions() method. You can set the following configuration parameters using the HBaseWriteOptions object:

- setBufferFlushMaxSizeInBytes : Maximum byte size of the buffered operations before flushing
- setBufferFlushMaxRows : Maximum number of operations buffered before flushing
- setBufferFlushIntervalMillis : Maximum time interval before flushing

See the following example for setting up an HBase sink running on the Cloudera platform:

```
// Define a new HBase sink for writing to the ITEM_QUERIES table
HBaseSinkFunction<QueryResult> hbaseSink = new HBaseSinkFunction<QueryResul
t>("ITEM_QUERIES") {
@Override
public void executeMutations(QueryResult qresult, Context context, Buffer
edMutator mutator) throws Exception {
  // For each incoming query result we create a Put operation
  Put put = new Put(Bytes.toBytes(qresult.queryId));
  put.addColumn(Bytes.toBytes("itemId"), Bytes.toBytes("str"),   Bytes.toBy
tes(qresult.itemInfo.itemId));
  put.addColumn(Bytes.toBytes("quantity"), Bytes.toBytes("int"), Bytes.to
Bytes(qresult.itemInfo.quantity));
  mutator.mutate(put);
  }
};
// Configure our sink to not buffer operations for more than a second (to
reduce end-to-end latency)
hbaseSink.setWriteOptions(HBaseWriteOptions.builder()
  .setBufferFlushIntervalMillis(1000)
  .build()
);
// Add the sink to our query result streamqueryResultStream.addSink(hbaseSi
nk);
```

Kafka with Flink

Cloudera Streaming Analytics offers Kafka connector as a source and a sink to create a complete stream processing architecture with a stream messaging platform. You must develop your application defining Kafka as a source and sink, after adding Kafka dependency to your project.

About this task

In Cloudera Streaming Analytics, adding Kafka as a connector creates a scalable communication channel between your Flink application and the rest of your infrastructure. Kafka is often responsible for delivering the input records and for forwarding them as an output, creating a frame around Flink.

When Kafka is used as a connector, Cloudera offers the following integration solutions:

- Schema Registry
- Streams Messaging Manager
- Kafka Metrics Reporter

Both Kafka sources and sinks can be used with exactly once processing guarantees when checkpointing is enabled.

For more information about Apache Kafka, see the Cloudera Runtime documentation.

Procedure

1. Add the Kafka connector dependency to your Flink job.

   Example for Maven:

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-kafka</artifactId>
  <version>3.2-csa1.13.0.0</version>
</dependency>
```

2. Set KafkaSource as the source in the Flink application logic.

   For Java

```
KafkaSource<String> source = KafkaSource.<String>builder()
    .setBootstrapServers("<your_broker_url>")
    .setTopics("<your_source_topic>")
    .setGroupId("<your_group_id>")
    .setStartingOffsets(OffsetsInitializer.earliest())
    .setValueOnlyDeserializer(new SimpleStringSchema())
    .build();
```

   For Python

```
source = KafkaSource.builder() \
    .set_bootstrap_servers('<your_broker_url>') \
    .set_topics('<your_source_topic>') \
    .set_group_id('<your_group_id>') \
    .set_starting_offsets(KafkaOffsetsInitializer.earliest()) \
    .set_value_only_deserializer(SimpleStringSchema()) \
    .build()
```

3. Set KafkaSink as the sink in the Flink application logic.

   For Java

```
DataStream<String> stream = ...
KafkaSink<String> sink = KafkaSink.<String>builder()
        .setBootstrapServers("<your_broker_url>")
        .setRecordSerializer(KafkaRecordSerializationSchema.builder()
            .setTopic("<your_sink_topic>")
            .setValueSerializationSchema(new SimpleStringSchema())
            .build())
        .setDeliveryGuarantee(DeliveryGuarantee.AT_LEAST_ONCE)
        .build();

stream.sinkTo(sink);
```

   For Python

```
stream = ...
sink = KafkaSink.builder() \
   .set_bootstrap_servers(brokers) \
   .set_record_serializer(
        KafkaRecordSerializationSchema.builder()
            .set_topic('<your_sink_topic>')
```

```
                .set_value_serialization_schema(SimpleStringSchema())
                .build()) \
        .set_delivery_guarantee(DeliveryGuarantee.AT_LEAST_ONCE) \
        .build()

stream.sink_to(sink)
```

Related Information

Stateful Tutorial: Setting up Kafka inputs and outputs

Checkpointing

Schema Registry with Flink

When Kafka is chosen as source and sink for your application, you can use Cloudera Schema Registry to register and retrieve schema information of the different Kafka topics. You must add Schema Registry dependency to your project and add the appropriate schema object to your Kafka topics.

There are several reasons why you should prefer the Schema Registry instead of custom serializer implementations on both consumer and producer side:

- Offers automatic and efficient serialization/deserialization for Avro, JSON and basic types
- Guarantees that only compatible data can be written to a given topic (assuming that every producer uses the registry)
- Supports safe schema evolution on both producer and consumer side
- Offers visibility to developers on the data types and they can track schema evolution for the different Kafka topics

Add the following Maven dependency or equivalent to use the schema registry integration in your project:

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-cloudera-registry</artifactId>
    <version>${flink.version}</version>
</dependency>
```

The schema registry can be plugged directly into the KafkaSource and KafkaSink using the appropriate schema:

For Avro

```
org.apache.flink.formats.registry.cloudera.avro.ClouderaRegistryAvroKafk
aRecordSerializationSchema
org.apache.flink.formats.registry.cloudera.avro.ClouderaRegistryAvroKafka
DeserializationSchema
```

For JSON

```
org.apache.flink.formats.registry.cloudera.json.ClouderaRegistryJsonKafk
aRecordSerializationSchema
org.apache.flink.formats.registry.cloudera.json.ClouderaRegistryJsonKafka
DeserializationSchema
```

See the Apache Flink documentation for Kafka consumer and producer basics.

Supported types

Currently, the following data types are supported for producers and consumers:

- Avro Specific Record types
- Avro Generic Records
- JSON data types

- Basic Java Data types: byte[], Byte, Integer, Short, Double, Float, Long, String, Boolean

To get started with Avro schemas and generated Java objects, see the Apache Avro documentation.

Security

You need to include every SSL configuration into a Map that is passed to the Schema Registry configuration.

```
Map<String, String> sslClientConfig = new HashMap<>();
sslClientConfig.put(K_TRUSTSTORE_PATH, params.get(K_SCHEMA_REG_SSL_CLIENT
_KEY + "." + K_TRUSTSTORE_PATH));
sslClientConfig.put(K_TRUSTSTORE_PASSWORD, params.get(K_SCHEMA_REG_SSL_CLI
ENT_KEY + "." + K_TRUSTSTORE_PASSWORD));

Map<String, Object> schemaRegistryConf = new HashMap<>();
schemaRegistryConf.put(K_SCHEMA_REG_URL, params.get(K_SCHEMA_REG_URL));
schemaRegistryConf.put(K_SCHEMA_REG_SSL_CLIENT_KEY, sslClientConfig);
```

For Kerberos authentication, Flink can maintain the authentication and ticket renewal automatically. You can define an additional RegistryClient property to the security.kerberos.login.contexts parameter in Cloudera Manager.

```
security.kerberos.login.contexts=Client,KafkaClient,RegistryClient
```

Schema serialization

You can construct the schema serialization with the ClouderaRegistryAvroKafkaRecordSerializationSchema and ClouderaRegistryJsonKafkaRecordSerializationSchema objects for FlinkKafkaProducer based on the required data format. You must set the topic configuration and RegistryAddress parameter in the object.

Required settings:

- Topic configuration when creating the builder, which can be static or dynamic (extracted from the data)
- RegistryAddress parameter on the builder to establish the connection

Optional settings:

- Arbitrary SchemaRegistry client configuration using the setConfig method
- Key configuration for the produced Kafka messages
  - Specifying a KeySelector function that extracts the key from each record
  - Using a Tuple2 stream for (key, value) pairs directly
- Security configuration

For Avro

```
        KafkaSerializationSchema<ItemTransaction> schema =
                ClouderaRegistryAvroKafkaRecordSerializationSchema.<Item
Transaction>builder(topic)
                        .setRegistryAddress(registryAddress)
                        .setKey(ItemTransaction::getItemId)
                        .build();
        KafkaSink<ItemTransaction> kafkaSink =
                KafkaSink.<Row>builder()
                        .setKafkaProducerConfig(kafkaProps)
                        .setRecordSerializer(schema)
                        .setDeliveryGuarantee(DeliveryGuarantee.AT_LEAST_O
NCE)
                        .build();
```

For JSON

```
        KafkaSerializationSchema<ItemTransaction> schema =
                ClouderaRegistryJsonKafkaRecordSerializationSchema.<Item
Transaction>builder(topic)
                         .setRegistryAddress(registryAddress)
                         .setKey(ItemTransaction::getItemId)
                         .build();
        KafkaSink<ItemTransaction> kafkaSink =
                KafkaSink.<Row>builder()
                        .setKafkaProducerConfig(kafkaProps)
                        .setRecordSerializer(schema)
                        .setDeliveryGuarantee(DeliveryGuarantee.AT_LEAST_O
NCE)
                        .build();
```

Schema deserialization

You can construct the schema deserialization with the ClouderaRegistryAvroKafkaRecordDeserializationSchema and ClouderaRegistryJsonKafkaRecordDeserializationSchema objects for FlinkKafkaProducer to read the messages in the same schema from the FlinkKafkaProducer. You must set the class or schema of the input messages and the Registry Address parameter in the object.

When reading messages (and keys), you always have to specify the expected Class<T> or record Schema of the input records. This way Flink can do any necessary conversion between the raw data received from Kafka and the expected output of the deserialization.

Required settings:

- Class or Schema of the input messages depending on the data type
- RegistryAddress parameter on the builder to establish the connection

Optional settings:

- Arbitrary SchemaRegistry client configuration using the setConfig method
- Key configuration for the consumed Kafka messages (only to be specified if reading the keys into a key or value stream is necessary)
- Security configuration

For Avro

```
        KafkaDeserializationSchema<ItemTransaction> schema =
                        ClouderaRegistryAvroKafkaDeserializationSchema.bui
lder(ItemTransaction.class)
                                .setRegistryAddress(registryAddress)
                                .build();
        KafkaSource<String> transactionSource = KafkaSource.<String>build
er()
                .setBootstrapServers("<your_broker_url>")
                .setTopics(inputTopic)
                .setGroupId(groupId)
                .setStartingOffsets(OffsetsInitializer.earliest())
                .setValueOnlyDeserializer(schema)
                .build();
```

For JSON

```
        KafkaDeserializationSchema<ItemTransaction> schema =
                        ClouderaRegistryJsonKafkaDeserializationSchema.bui
lder(ItemTransaction.class)
```

```
                                .setRegistryAddress(registryAddress)
                                .build();

        KafkaSource<String> transactionSource = KafkaSource.<String>buil
der()
                .setBootstrapServers("<your_broker_url>")
                .setTopics(inputTopic)
                .setGroupId(groupId)
                .setStartingOffsets(OffsetsInitializer.earliest())
                .setValueOnlyDeserializer(schema)
                .build();
```

Kafka Metrics Reporter

In Cloudera Streaming Analytics, Kafka Metrics Reporter is available as another monitoring solution when Kafka is used as a connector within the pipeline to retrieve metrics about your streaming performance.

Flink offers a flexible Metrics Reporter API for collecting the metrics generated by your streaming pipelines. Cloudera provides an additional implementation of this, which writes metrics to Kafka with the following JSON schema:

```
{
  "timestamp" : number -> millisecond timestamp of the metric record
  "name" : string -> name of the metric
    (e.g. numBytesOut)
  "type" : string -> metric type enum: GAUGE, COUNTER, METER, HISTOGRAM
  "variables" : {string => string} -> Scope variables
    (e.g. {"<job_id>" : "123", "<host>" : "localhost"})
  "values" : {string => number} -> Metric specific values
    (e.g. {"count" : 100})
}
```

For more information about Metrics Reporter, see the Apache Flink documentation.

Configuration of Kafka Metrics Reporter

The Kafka metrics reporter can be configured similarly to other upstream metric reporters.

Required parameters

- topic: target Kafka topic where the metric records will be written at the configured intervals
- bootstrap.servers: Kafka server addresses to set up the producer

Optional parameters

- interval: reporting interval, default value is 10 seconds, format is 60 SECONDS
- log.errors: logging of metric reporting errors, value either true or false

You can configure the Kafka metrics reporter per job using the following command line properties:

```
flink run -d -p 2 -ynm HeapMonitor \
-yD metrics.reporter.kafka.class=org.apache.flink.metrics.kafka.KafkaMetric
sReporter \
-yD metrics.reporter.kafka.topic=metrics-topic.log \
-yD metrics.reporter.kafka.bootstrap.servers=<kafka_broker>:9092 \
-yD metrics.reporter.kafka.interval="60 SECONDS" \
-yD metrics.reporter.kafka.log.errors=false \
flink-simple-tutorial-1.3-SNAPSHOT.jar
```

The following is a more advanced Flink command that also contains security related configurations:

```
flink run -d -p 2 -ynm HeapMonitor \
```

```
-yD security.kerberos.login.keytab=some.keytab \
-yD security.kerberos.login.principal=some_principal \
-yD metrics.reporter.kafka.class=org.apache.flink.metrics.kafka.KafkaMetric
sReporter \
-yD metrics.reporter.kafka.topic=metrics-topic.log \
-yD metrics.reporter.kafka.bootstrap.servers=<kafka_broker>:9093 \
-yD metrics.reporter.kafka.interval="60 SECONDS" \
-yD metrics.reporter.kafka.log.errors=false \
-yD metrics.reporter.kafka.security.protocol=SASL_SSL \
-yD metrics.reporter.kafka.sasl.kerberos.service.name=kafka \
-yD metrics.reporter.kafka.ssl.truststore.location=truststore.jks \
flink-simple-tutorial-1.3-SNAPSHOT.jar
```

You can also set the metrics properties globally in Cloudera Manager using Flink Client Advanced Configuration Snippet (Safety Valve) for flink-conf-xml/flink-conf.xml.

Arbitrary Kafka producer properties

The reporter supports passing arbitrary Kafka producer properties that can be used to modify the behavior, enable security, and so on. Serializer classes should not be modified as it can lead to reporting errors.

See the following example configuration of the Kafka Metrics Reporter:

```
# Required configuration
metrics.reporter.kafka.class:
org.apache.flink.metrics.kafka.KafkaMetricsReporter
metrics.reporter.kafka.topic: metrics-topic.log
metrics.reporter.kafka.bootstrap.servers: broker1:9092,broker2:9092

# Optional configuration
metrics.reporter.kafka.interval: 60 SECONDS
metrics.reporter.kafka.log.errors: false

# Optional Kafka producer properties
metrics.reporter.kafka.security.protocol: SSL
metrics.reporter.kafka.ssl.truststore.location:
/var/private/ssl/kafka.client.truststore.jks
```

Note: Any optional property with metrics.reporter.kafka. prefix tag is processed as Kafka client configuration.

For example: metrics.reporter.kafka.property_name: property_value will be converted to property_name: p roperty_value.

Kudu with Flink

Cloudera Streaming Analytics offers Kudu connector as a sink to create analytical application solutions. Kudu is an analytic data storage manager. When using Kudu with Flink, the analyzed data is stored in Kudu tables as an output to have an analytical view of your streaming application.

You can read Kudu tables into a DataStream using the KuduCatalog with Table API or using the KuduRowInputForm at directly in the DataStream. The difference between the two methods is that when using the KuduRowInputFormat, you need to manually provide information about the table.

For more information about the Kudu sink in DataStream API, see the official documentation.

Iceberg with Flink

Apache Iceberg is an open, high-performance table format for organizing datasets that can contain petabytes of data. In Cloudera Streaming Analytics, the DataStream API enables you to use Iceberg tables in your Flink jobs.

Iceberg is integrated with Flink as a connector, and you can use the DataStream API to read and write data from a data source to an Iceberg table. Both the V1 and V2 version specifics are supported by the Flink connector. For more information, about the version changes, see the Apache Iceberg documentation.

Table 2: DataStream feature support for the Iceberg and Flink integration

Feature	Flink
Read	Supported
Append	Supported
Overwrite	Supported
Upsert	Technical Preview[1]

When using Iceberg with the DataStream API, you need to create the Flink job that includes referencing the Iceberg table at the TableLoader. The following example shows how to create your Flink job that reads or writes data to or from an Iceberg table using HDFS.

For DataStream read

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.createLocalE
nvironment();

TableLoader tableLoader = TableLoader.fromHadoopTable("hdfs://nn:8020/war
ehouse/path");

DataStream<RowData> stream = FlinkSource.forRowData()
    .env(env)
    .tableLoader(tableLoader)
    .streaming(true)
    .startSnapshotId(3821550127947089987L)
    .build();
// Print all records to stdout.
stream.print();

// Submit and execute this streaming read job.
env.execute("Test Iceberg Streaming Read");
```

For DataStream append

```
StreamExecutionEnvironment env = ...;
DataStream<RowData> input = ... ;
Configuration hadoopConf = new Configuration();
TableLoader tableLoader = TableLoader.fromHadoopTable("hdfs://nn:8020/w
arehouse/path", hadoopConf);

FlinkSink.forRowData(input)
   .tableLoader(tableLoader)
   .append();

env.execute("Test Iceberg DataStream");
```

For DataStream overwrite

```
StreamExecutionEnvironment env = ...;
```

[1] The upsert feature of the Iceberg and Flink integration is in Technical Preview and not ready for production deployment. Cloudera encourages you to explore these features in non-production environments and provide feedback on your experiences through the Cloudera Community Forums.

```
DataStream<RowData> input = ... ;
Configuration hadoopConf = new Configuration();
TableLoader tableLoader = TableLoader.fromHadoopTable("hdfs://nn:8020/wa
rehouse/path", hadoopConf);
FlinkSink.forRowData(input)
   .tableLoader(tableLoader)
   .overwrite(true)
   .append();
env.execute("Test Iceberg DataStream");
```

For DataStream upsert

```
StreamExecutionEnvironment env = ...;
DataStream<RowData> input = ... ;
Configuration hadoopConf = new Configuration();
TableLoader tableLoader = TableLoader.fromHadoopTable("hdfs://nn:8020/wa
rehouse/path", hadoopConf);
FlinkSink.forRowData(input)
   .tableLoader(tableLoader)
   .upsert(true)
   .append();
env.execute("Test Iceberg DataStream");
```

You can add the options to write when configuring the Flink Sink as shown in the following example:

```
FlinkSink.Builder builder = FlinkSink.forRow(dataStream, SimpleDataUtil.FLIN
K_SCHEMA)
   .table(table)
   .tableLoader(tableLoader)
   .set("write-format", "orc")
   .set(FlinkWriteOptions.OVERWRITE_MODE, "true");
```

File systems

You can use file systems in Flink to consume and persistently store data for application results, fault tolerance and data recovery.

The file system used for a particular file is determined by its URI scheme. For example, file:///home/user/te xt.txt refers to a file in the local file system. Flink has built-in support for the file system of the local machine, including any NFS or SAN drives mounted into that local file system. It can be used by default without additional configuration.

For all schemes where Flink cannot find a directly supported file system, it falls back to Hadoop. All Hadoop file systems are automatically available when flink-runtime and the Hadoop libraries are on the classpath. This way, Flink seamlessly supports all of Hadoop file systems implementing the org.apache.hadoop.fs.FileSystem interface, and all Hadoop-compatible file systems (HCFS).

There are also pluggable file systems that are not included by default. CSA contains the following pluggable file systems:

- Amazon S3
- Azure Data Lake Store Gen2
- Azure Blob Storage
- Google Cloud Storage

For more information about how to use the supported file systems, see the Apache Flink documentation.

Using Ozone with Flink

You can use Ozone as a sink with Flink via the implementation of Hadoop file system, but it has some limitations.

Limitations using Ozone File System (OFS) with DataStream connector

When using Ozone as a file system, you must use the OnCheckpointRollingPolicy configuration, which rolls part files on every checkpoint. This configuration is needed because in case part files traverse the checkpoint interval, upon recovery from a failure, the FileSink may use the truncate() method of the file system to discard uncommitted data from the in-progress file. This method is not supported by OFS and Flink will throw an exception.

Limitations using OFS with Table API connector

Because of the limitation present for the DataStream connector, it is required to roll the files on every checkpoint when using the Table API connector. If checkpointing is enabled and a bulk format is used, then this is done automatically. In case of a row format, automatic compaction has to be turned on.

Job lifecycle

Setting up Python for PyFlink

Before you can use Flink with the Python API, you need to create and initialize a Python virtual environment.

1. Connect to the Flink Gateway node using CLI.

   ```
   ssh <[***WORKLOAD USERNAME***]>@[***FLINK MANAGER NODE***]
   ```

   Provide your workload password when prompted.
2. Create a Python virtual environment using the following command:

   ```
   conda create --copy -y -n flink_venv python=3.8
   ```

   Important: --copy is essential so the created Python environment contains all the required files, not just symlinks.
3. Activate the newly created virtual environment:

   ```
   conda activate flink_venv
   ```
4. Install PyFlink to the flink_venv virtual environment using the following command:

   ```
   python -m pip install apache-flink==1.19.1
   ```
5. Create a ZIP archive from the flink_venv virtual environment so it can be deployed with a Flink job:

   ```
   cd path/to/flink_venv && zip -r venv.zip .
   ```

   Note: Zipping the virtual environment in its root folder makes the deployment paths shorter and more straightforward.

When the Python installation is complete, you can submit Flink application that were created using the Python API.

Running a Flink job

After developing your application, you can submit your Flink job in YARN per-job, session or application mode. To submit the Flink job, you need to run the Flink client in the command line including security parameters and other configurations with the run command.

About this task

Submitting a job means uploading the job's JAR and related dependencies to the Flink cluster and initiating the job execution.

The Flink jobs you submit to the cluster are running on YARN. Submitting a job means that the JAR file of the Flink application is uploaded to the cluster with the related dependencies. and the job execution is initiated. You have the following deployment modes in which you can run your Flink jobs:

- Per-job mode

  Per-job mode means that you run the Flink job in a dedicated YARN application. In this case each submitted Flink job has its own Flink cluster in YARN, with its own Job Manager and Task Managers. When you run Flink jobs in per-job mode, every job submission creates a new cluster. As the cluster deployment has to be created with every submission, the execution of the job can take up time.
- Session mode

  Session mode means that you run multiple Flink jobs in the same YARN sessions. In this case every Flink job shares the cluster, the allocated resources, the Job Manager and Task Managers. When you run Flink jobs in session mode, the submitted jobs are created in one cluster and are long-lived. The execution time is shorter than in per-job mode, however you need to consider that in a session mode a cluster failure affects every Flink job, and recreation from a savepoint can take up time.
- Application mode

  Regarding resource management, Application mode is much like Per-job mode. That means every deployed application has its dedicated Flink cluster. The key difference is the main() method of the deployed application is executed by the Job Manager rather than the client.

For more information about the deployment modes of Flink, see the Apache Flink documentation.

Note:

Even though it is deprecated for Apache Flink, the per-job deployment mode of Flink is still supported in Cloudera Streaming Analytics.

You can set how to run your Flink job with the execution.target setting in the Flink configuration file. By default, execution.target is set to yarn-per-job, but you can change it to yarn-application or yarn-session. Alternatively, you can add the corresponding arguments to the flink run command when submitting the Flink job.

Procedure

1. Connect to the cluster using ssh where you want to run the Flink application.

```
ssh [***YOUR WORKLOAD USERNAME***]@[***YOUR MASTER NODE FQDN***]
```

   Note: You are prompted to provide your password to the cluster.

2. Submit the Flink job using the flink run command.

   Per-job mode

   For Java

```
flink run \
```

```
  -d \
  -t yarn-per-job \
  -ynm <job_name_in_yarn> \
  -p <job_parallelism> \
  -ys <slots_per_task_manager> \
  -ytm <memory_per_container_in_mb> \
  <job_jar_file> <job_parameters>
```

For Python

```
flink run \
  -d \
  -t yarn-per-job \
  -ynm <job_name_in_yarn> \
  -p <job_parallelism> \
  -ys <slots_per_task_manager> \
  -ytm <memory_per_container_in_mb> \
  -py <python_script> <job_parameters>
```

For Python venv

```
# Assuming the content of '/tmp/shipfiles': venv.zip, word_c
ount.py
flink run -t yarn-per-job \
-Dyarn.ship-files=/tmp/shipfiles \
-pyarch /tmp/shipfiles/venv.zip \
-pyclientexec venv.zip/bin/python3 \
-pyexec venv.zip/bin/python3 \
-pyfs /tmp/shipfiles \
-pym word_count
```

Session mode

a. Start a Flink session cluster.

```
flink-yarn-session \
  -d \
  -nm <job_name_in_yarn> \
  -s <slots_per_task_manager> \
  -tm <memory_per_container_in_mb>
```

The flink-yarn-session command outputs the ID of the corresponding YARN application. You need to add the YARN application ID to the flink run command.

```
YARN ApplicationID: application_1616633166424_0024
```

b. Submit the Flink job.

For Java

```
flink run \
  -d \
  -t yarn-session \
  -yid <application_id> \
  -p <parallelism> \
  <job_jar_file> <job_parameters>
```

For Python

```
flink run \
```

```
  -d \
  -t yarn-session \
  -yid <application_id> \
  -p <parallelism> \
  -py <python_script> <job_parameters>
```

Application mode

For Java

```
flink run-application \
  -d \
  -t yarn-application \
  -p <job_parallelism> \
  -ys <slots_per_task_manager> \
  -ytm <memory_per_container_in_mb> \
  <job_jar_file> <job_parameters>
```

For Python

```
flink run-application \
  -d \
  -t yarn-application \
  -p <job_parallelism> \
  -ys <slots_per_task_manager> \
  -ytm <memory_per_container_in_mb> \
  -py <python_script> <job_parameters>
```

For more examples and PyFlink possibilities, see the Submitting PyFlink Jobs section of the Apache Flink documentation.

Note:

To run a Flink job, your HDFS Home Directory has to exist. If it does not exist, you receive an error message similar to:

```
Permission denied: user=$USER_NAME, access=WRITE, inode="/user"
```

Related Information

Simple Tutorial: Running the application from IntelliJ

Simple Tutorial: Running the application on a Cloudera cluster

Stateful Tutorial: Deploy and monitor the application

Using Flink CLI

You can use the Flink command line interface to operate, configure and maintain your Flink applications.

The Flink CLI works without requiring the user to always specify the YARN application ID when submitting commands to Flink jobs. Instead, the jobs are identified uniquely on the YARN cluster by their job IDs.

The following table summarizes the supported actions of Flink CLI:

Action	Purpose
run	You can use this action to execute jobs. It requires at least the jar containing the job. Flink- or job-related arguments can be passed if necessary.

Action	Purpose
run-application	You can use this action to execute jobs in application mode. It requires the same parameters as the run action.
info	You can use this action to print an optimized execution graph of the passed job. It requires the jar containing the job.
list	You can use this action to list all running or scheduled jobs.
savepoint	You can use this action to create or dispose savepoints for a given job. You might need to specify a savepoint directory besides the job ID if the state.savepoints.dir was not specified in Cloudera Manager.
checkpoint	You can use this action to create checkpoints for a given job. The checkpoint type (full or incremental) can be specified.
cancel	You can sue this action to cancel running jobs based on their job ID.
stop	You can use this action to stop a running job, but also create a savepoint to start from again. This action combined cancel and save point.

Enabling savepoints for Flink applications

Beside checkpointing, you are also able to create a savepoint of your executed Flink jobs. Savepoints are not automatically created, so you need to trigger them in case of upgrade or maintenance. You can also resume your applications from savepoint.

You can set the default savepoint directory in flink-conf.yaml under state.savepoints.dir property.

The following command lines can be used to maintain savepoints:

Trigger savepoint	$ bin/flink savepoint -yid <*YARNAPPID*> <*JOBID*> [targetDirectory]
Cancel job with savepoint	$ bin/flink cancel -yid <*YARNAPPID*> <*JOBID*>
Resume from savepoint	$ bin/flink run -s <*SAVEPOINTPATH*> [runArgs]
Deleting savepoint	$ bin/flink savepoint -d <*SAVEPOINTPATH*>

Monitoring

Enabling Flink DEBUG logging

You can review the log text files of the Flink jobs when an error is detected during the processes. When you set the log level of Flink to DEBUG, you can easily trace the log file for errors.

About this task

A log file is created for every Flink process that contains messages for the different events happening in the given process. You can use these log files to solve the errors and problems that can occur during Flink processes. You can access the Flink logs using the Flink Dashboard.

Procedure

1. Navigate to the **Configuration** page in Cloudera Manager.
   a) Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
   b) Select the Cloudera Streaming Analytics cluster from the list of Cloudera Data Hub clusters.
   c) Select Cloudera Manager from the services.

      You are redirected to the **Cloudera Manager** user interface.
   d) Click Clusters > Flink .
   e) Click Configuration.
2. Search for Flink Client Advanced Configuration Snippet (Safety Valve) for flink-conf/log4j.properties configuration.
3. Add the following parameters to the Safety Valve:

```
logger.flink.name = org.apache.flink
logger.flink.level = DEBUG
```

4. Click Save Changes.
5. Restart the Flink service with Action > Restart .

   After updating the Flink log configuration in Cloudera Manager, you need to access the YARN Resource Manager user interface from the Cloudera Streaming Analytics cluster page.
6. Access the YARN Resource Manager user interface to stop the YARN job of the Flink application.
   a) Navigate to Cloudera Management Console > Environments, and select the environment where you have created your cluster.
   b) Select the Cloudera Streaming Analytics cluster.
   c) Select Resource Manager from the list of Services.

      You are redirected to the **Resource Manager** user interface.
   d) Select Applications.

      The running Flink applications are displayed.
7. Select the application you need to stop.
8. Click Settings.
9. Select Kill application.

   After stopping the Flink application, you can review the log file of the Flink job by accessing the Flink Dashboard from the Cloudera Streaming Analytics cluster page.
10. Navigate to **Flink Dashboard** and review the log level for the running job.
    a) Navigate to Cloudera Management Console > Environments, and select the environment where you have created your cluster.
    b) Select the Cloudera Streaming Analytics cluster.
    c) Click Flink Dashboard from the list of services.
11. Select Task Managers from the main menu.
12. Select the previously submitted job.
13. Click Logs.

Flink Dashboard

The Flink Dashboard is a built-in monitoring interface for Flink applications in Cloudera Streaming Analytics. You can monitor your running, completed and stopped Flink jobs on the dashboard. You reach the Flink Dashboard through Cloudera Manager.

After deploying Flink and the required components, you can configure and monitor each component individually, or the whole cluster with Cloudera Manager. For the general use of Cloudera Manager, see the Cloudera Manager documentation.

The Flink Dashboard acts as a single UI for monitoring all the jobs running on the YARN cluster. It shows all the running, failed, and finished jobs.

Note: The Flink Dashboard is an updated version of the Flink HistoryServer.

You can also use the dashboard to navigate between the different Flink clusters from a central place.

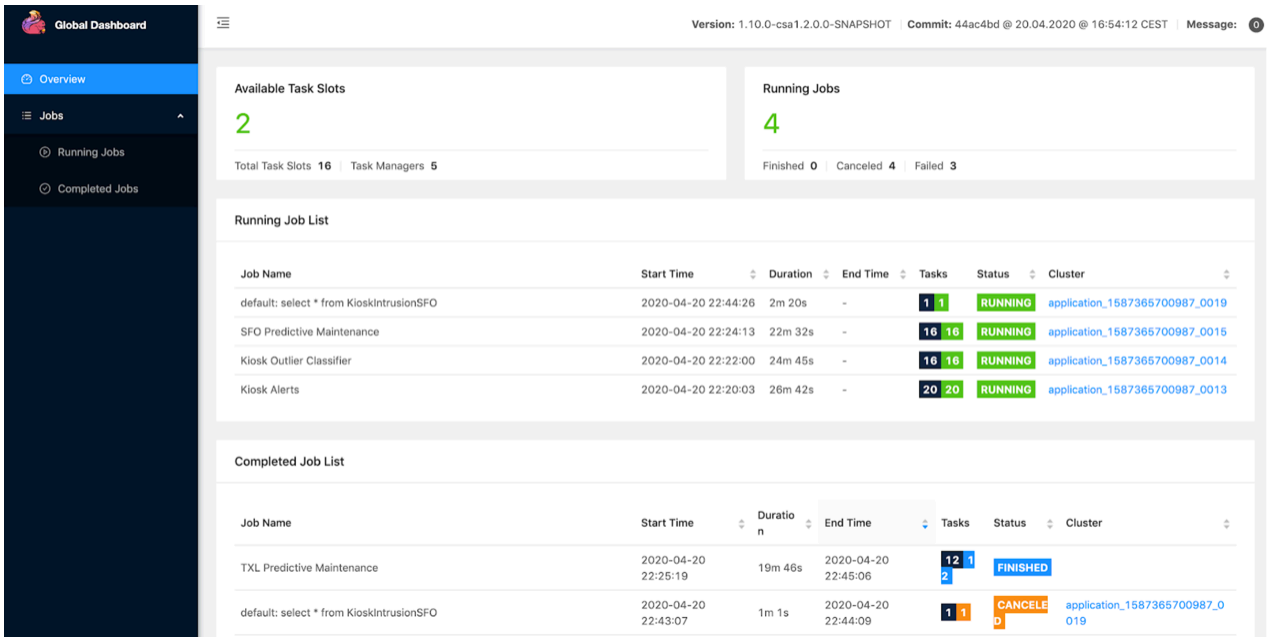

Streams Messaging Manager integration

You can use the Streams Messaging Manager (SMM) UI to monitor end-to-end latency of your Flink application when using Kafka as a datastream connector.

End-to-end latency throughout the pipeline can be monitored using SMM. To use SMM with Flink, interceptors need to be enabled for Kafka in the Flink connectors.

For more information about enabling interceptors, see the SMM documentation.

Flink SQL and Table API

In Cloudera Streaming Analytics, you can enhance your streaming application with analytical queries using Table API or SQL API. These are integrated in a joint API and can also be embedded into regular DataStream applications. The central concept of the joint API is a Table that serves as the input and output of your streaming data queries.

There are also two planners that translate Table/SQL queries to Flink jobs: the old planner and the Blink planner. Cloudera Streaming Analytics only supports the Blink planner for Table/SQL applications.

Adding the following Maven dependency to the Flink configuration file allows you to use the Table API with the Blink planner.

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-table-api-java-bridge</artifactId>
```

```
  <version>1.19.1-csa1.13.0.0</version>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-table-planner_2.12</artifactId>
  <version>1.19.1-csa1.13.0.0</version>
</dependency>
```

SQL programs in Flink follow a structure similar to regular DataStream applications:

1. Create a StreamTableEnvironment with the Blink planner.
2. Register catalogs and tables.
3. Run the queries/updates.
4. Run the StreamTableEnvironment.

You can see an example of the structure here:

```
StreamExecutionEnvironment streamEnv = StreamExecutionEnvironment.getExecuti
onEnvironment();

EnvironmentSettings tableSettings = EnvironmentSettings
     .newInstance()
     .build();

StreamTableEnvironment tableEnv = StreamTableEnvironment
     .create(streamEnv, tableSettings);

tableEnv.executeSql("CREATE TABLE ...");
Table table = tableEnv.sqlQuery("SELECT ... FROM ...");

DataStream<Row> stream = tableEnv.toDataStream(table, Row.class);
stream.print();
```

The Table API exposes different flavors of TableEnvironment to the end users that cover different feature sets. To ensure smooth interaction between other DataStream applications, Cloudera Streaming Analytics only supports using StreamTableEnvironment.

StreamTableEnvironment wraps a regular StreamExecutionEnvironment. This allows you to seamlessly go from streams to tables and back within the same pipeline.

You can create StreamTableEnvironment with the following code entry:

```
StreamExecutionEnvironment streamEnv = ...
EnvironmentSettings tableSettings = EnvironmentSettings
     .newInstance()
     .build();

StreamTableEnvironment tableEnv = StreamTableEnvironment
     .create(streamEnv, tableSettings);
```

When combining regular DataStream and Table/SQL applications, make sure to always call the .execute command on the StreamTableEnvironment instead of the regular StreamExecutionEnvironment to ensure correct execution.

SQL and Table API supported features

The following SQL and Table API features are supported in Cloudera Streaming Analytics.

StreamTableEnvironment (Blink Planner)	Catalogs

• DataStream conversions:
  • Row
  • Tuple
• Temporary views
• SQL update and Query operations
• Creating and using catalogs
• Listing catalogs, tables, databases and functions
	• In-memory - Flink default catalog
	• Hive
	• Schema Registry
	• Kudu

SQL update statements	SQL query statements

• CREATE TABLE
  • Computed columns
  • Watermark definitions
  • Table connectors
  • Kafka
  • Kudu
  • Hive (through catalog)
  • Data formats (Kafka)
  • JSON
  • Avro
  • CSV
• INSERT INTO
  • VALUES
  • SELECT queries
	• Basic operations
	  • Scan, Project, Filter
	• Basic aggregations
	  • Group By
	  • Group By Window
	  • Distinct (only windowed)
	• Joins
	  • Time-windowed stream-stream join
	• User defined scalar functions (scalar udf)

DataStream API interoperability

The DataStream API interoperability offers you new ways to build your Flink streaming application logic as you can convert the DataStreams to Tables, and the Tables back to Datastreams. This means that you can run SQL queries on your DataStreams. You can also convert the result back to other streams, or insert them into one of the supported table sinks.

The following DataStream type conversions are supported:

- DataStream<Row>
- DataStream<TupleX<...>>

Converting DataStreams to Tables

When converting DataStreams to Tables you need to define the StreamTableEnvironment for the conversion. Cloudera recommends creating the tables with names as it is easier to refer to them in SQL. You should also take the processing and event time into consideration as crucial elements of Flink streaming applications.

StreamTableEnvironment is used to convert a DataStream into a Table. You can use the fromDataStream and crea teTemporaryView methods for the conversion. Cloudera recommends that you use the createTemporaryView method as it provides a way to assign a name to the created table. Named tables can be referenced directly in SQL afterwards.

Both of these methods take an optional, but recommended, string parameter to define field name mappings. The string must contain a comma separated list of the desired column names. If the string is not specified, the column names are set to f0, f1, ...fn.

```
DataStream<Tuple2<Integer, String>> stream = ...

Table table = tableEnv.fromDataStream(stream, "col_1, col_2");

tableEnv.createTemporaryView("MyTableName", stream, "col_1, col_2");
```

You need to take into consideration the event timestamps and watermarks when converting DataStreams.

The processing time attribute must be defined as an additional (logical) column marked with the .proctime property during schema definition.

```
DataStream<Row> stream = ...

Table table = tableEnv.fromDataStream(stream, "col_1, col_2, t
s_col.proctime");
```

For more information on time handling in SQL, see the Apache Flink documentation.

Even time attributes are defined by the .rowtime property during schema definition. This can either replace an existing field or create a new one, but in either case, the field holds the event timestamp of the current record.

```
DataStream<Tuple2<Timestamp, String>> stream = …
stream.assignTimestampsAndWatermarks(...)

Table table = tableEnv.fromDataStream(stream, "event_ts.rowtime,
 col_2");
```

For more information on time handling in SQL, see the Apache Flink documentation.

Converting Tables to DataStreams

Tables are updated dynamically as the result of streaming queries. To convert them into DataStreams, you can either append them or retract them based on the SQL query you have chosen.

The Table changes as new records arrive on the query's input streams. These Tables can be converted back into Data Streams by capturing the change of the query output.

There are two modes to convert a Table into a DataStream:

- Append Mode: This mode can only be used if the dynamic Table is only modified by INSERT changes. For example, it is append-only and previously emitted results are never updated.
- Retract Mode: This mode can always be used. It encodes INSERT and DELETE changes with a boolean flag. True marks inserts, and false marks deletes.

Both toAppendStream and toRetractStream methods take the conversion class or conversion type information as parameters. For the recommended Row conversions, you need to provide the Row.class. For Tuple conversions, you need to provide the Tuple TypeInformation object manually.

```
Table table = tableEnv.sqlQuery("SELECT name, age FROM People");

DataStream<Row> appendStream = tableEnv.toAppendStream(table, Row.class);
DataStream<Tuple2<Boolean, Row>> retractStream = tableEnv.toRetractStream(ta
ble, Row.class);

DataStream<Tuple2<String, Integer>> tupleStream = tableEnv.toAppendStream(
  table,
  new TypeHint<Tuple2<String, Integer>>() {}.getTypeInfo()
);
```

Note: For a detailed discussion about dynamic tables and their properties, see the Apache Flink documentation.

Supported data types

You should review the supported data types before designing your application to have all the information regarding SQL type mappings, timestamp and date types.

Java to SQL type mappings

SQL Type	From DataStream	To DataStream
STRING	String	String
BOOLEAN	boolean/Boolean	boolean/Boolean
BYTES	byte[]	byte[]
DECIMAL(38,18)	BigDecimal	BigDecimal
TINYINT	byte/Byte	byte
SMALLINT	short/Short	short/Short
INT	int/Integer	int/Integer
BIGINT	long/Long	long/Long
FLOAT	float/Float	float/Float
DOUBLE	double/Double	double/Double
MAP	Maps of supported types using MapTypeInfo	Maps of supported types
ARRAY	primitive/object arrays	primitive/object arrays*
ROW	Row	Row

Timestamp and Date types

The Table API supports a wide variety of conversions between java.sql , java.time and SQL types. For smooth operation, it is recommended to use java.sql time classes whenever possible.

		To DataStream	
SQL Type	From DataStream	Tuple	Row
DATE	java.sql.Date	java.sql.Date	java.time.LocalDate
	java.time.LocalDate*	java.time.LocalDate*	
TIME(0)	java.sql.Time	java.sql.Time	java.time.LocalTime
	java.time.LocalTime*	java.time.LocalTime*	
TIMESTAMP(3)	java.sql.Timestamp	java.sql.Timestamp	java.time.LocalDateTime
	java.time.LocalDateTime (no .row time support)*	java.time.LocalDateTime*	
TIMESTAMP_WITH_LOCAL_TIME_ZONE	java.time.Instant*	java.time.Instant*	java.time.Instant

Note: For information about the limitations regarding DataStream conversion, see the Release Notes.

SQL catalogs for Flink

Cloudera Streaming Analytics supports Hive, Kudu and Schema Registry catalogs to provide metadata for the stored data in a database or other external systems. You can choose the SQL catalogs based on your Flink application design.

For more information about Flink Catalogs, see the Apache Flink documentation.

In-memory catalog

A generic in-memory catalog is enabled by default. However when the in-memory catalog is used, all objects are only available for the lifetime of the session.

Hive catalog

You can add Hive as a catalog in Flink SQL by adding Hive dependency to your project, registering the Hive table in Java and setting it either globally in Cloudera Manager or the custom environment file.

The Hive catalog serves two purposes:

- It is a persistent storage for pure Flink metadata
- It is an interface for reading and writing existing Hive tables

Maven Dependency

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-hive_2.12</artifactId>
  <version>1.19.1-csa1.13.0.0</version>
</dependency>
```

The following example shows how to register and use the Hive catalog from Java:

```
String HIVE = "hive";
String DB = "default";
String HIVE_CONF_DIR = "/etc/hive/conf";
String HIVE_VERSION = "3.1.3000";
HiveCatalog catalog = new HiveCatalog(HIVE, DB, HIVE_CONF_DIR, HIVE_VERSION
);
tableEnv.registerCatalog(HIVE, catalog);
tableEnv.useCatalog(HIVE);
```

Note: According to the latest recommended setup on Cloudera, the Hive service hosts only the Hive Metastore Server. Make sure that a Gateway role is installed, or the HMS itself is deployed on the node where the Flink commands are submitted.

Note: Do not use Flink to create general purpose batch tables in the Hive metastore that you expect to be used from other SQL engines. While these tables will be visible, Flink uses the additional properties extensively to describe the tables, and thus other systems might not be able to interpret them. Use Hive directly to create these tables instead.

Kudu catalog

You can add Kudu as a catalog in Flink SQL by adding Kudu dependency to your project, registering the Kudu table in Java, and enabling it in the custom environment file.

The Kudu connector comes with a catalog implementation to handle metadata about your Kudu setup and perform table management. By using the Kudu catalog, you can access all the tables already created in Kudu from Flink SQL queries.

The Kudu catalog only allows to create or access existing Kudu tables. Tables using other data sources must be defined in other catalogs, such as in-memory catalog or Hive catalog.

Maven Dependency

```
<dependency>
  <groupId>org.apache.bahir</groupId>
  <artifactId>flink-connector-kudu_2.12</artifactId>
  <version>1.1.0-csa1.13.0.0</version>
</dependency>
```

The following example shows how to register and use the Kudu catalog from Java:

```
String KUDU_MASTERS="host1:port1,host2:port2"
KuduCatalog catalog = new KuduCatalog(KUDU_MASTERS);
tableEnv.registerCatalog("kudu", catalog);
```

```
tableEnv.useCatalog("kudu");
tableEnv.listTables();
```

For more information about the Kudu connector and catalog, see the official documentation.

Note: For information about the limitations regarding Kudu catalog, see the Release Notes.

Schema Registry catalog

The Schema Registry catalog allows you to access Kafka topics with registered schemas as Flink SQL tables. You can add Schema Registry as a catalog in Flink SQL by adding the dependency to your project, registering it in Java, and enabling it in the custom environment file.

Each Kafka topic will be mapped to a table with TableSchema that matches the Avro schema.

Maven Dependency

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-cloudera-registry</artifactId>
  <version>1.0-csa1.13.0.0</version>
</dependency>
```

The following example shows how to register and use the Schema Registry catalog from Java:

```
SchemaRegistryClient client = new SchemaRegistryClient(
 ImmutableMap.of(
  SchemaRegistryClient.Configuration.SCHEMA_REGISTRY_URL.name(),
"http://<your_hostname>:7788/api/v1"
      )
 );
Map<String, String> connectorProps = new Kafka()
  .property(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
                      "<your_hostname>:9092")
  .startFromEarliest()
  .toProperties();

tableEnv.registerCatalog(
   "registry", new ClouderaRegistryCatalog("registry", client, connectorPr
ops)
 );
tableEnv.useCatalog("registry");
```

Note: The SSL related properties have to be set independently for both the SchemaRegistryClient and Kafka.

Note: For information about the limitations regarding Schema Registry, see the Release Notes.

SQL connectors for Flink

In Flink SQL, the connector describes the external system that stores the data of a table. Cloudera Streaming Analytics offers you Kafka and Kudu as SQL connectors. You need to further choose the data formats and table schema based on your connector.

Some systems support different data formats. For example, a table that is stored in Kafka can encode its rows with CSV, JSON, or Avro. The table schema defines the schema of a table that is exposed to SQL queries. It describes how sources and sinks map the data format to the table schema.

Kudu connector

The Kudu connector in Cloudera Streaming Analytics offers compatibility with other supported catalogs, and capability to convert your Kudu tables into DataStreams. You need to add the Kudu dependency to your project, set up the catalog, and you can either use SQL queries or the Kudu catalog directly to create tables.

Kafka connector

Cloudera Streaming Analytics provides Kafka as not only a DataStream connector, but also enables Kafka in the Flink SQL feature. This means if you have designed your streaming application to have Kafka as source and sink, you can retrieve your output data in tables. When using the Kafka connector, you are required to specify one of the supported message formats.

Maven dependency

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-kafka</artifactId>
  <version>3.2-csa1.13.0.0</version>
</dependency>
```

For more information about the Kafka connector, see the Apache Flink documentation.

The following example shows a CREATE TABLE statement with Kafka connector:

```
CREATE TABLE source_table (
 id  BIGINT,
 ts  BIGINT,
 itemId STRING,
 quantity  INT
) WITH (
 'connector.type'      = 'kafka',
 'connector.version'   = 'universal',
 'connector.topic'     = 'input_topic',
 'connector.startup-mode' = 'latest-offset',
 'connector.properties.bootstrap.servers' = '<hostname>:<port>',
 'connector.properties.group.id' = 'test',
 'format.type' = 'json'
);
```

On a secured environment where Kerberos and SSL is enabled, the following example can be used:

```
CREATE TABLE source_table (
  c1 STRING
) WITH (
  'connector.type'                                    = 'kafka',
   'connector.version'                                 = 'universal',
   'connector.topic'                                   = 'source_topic',
   'connector.startup-mode'                            = 'earliest-offset',
   'connector.properties.bootstrap.servers'            = '<host>:<port>',
   'connector.properties.group.id'                     = 'test',
   'connector.properties.security.protocol'            = 'SASL_SSL',
   'connector.properties.sasl.kerberos.service.name'   = 'kafka',
   'connector.properties.ssl.truststore.location'      =
 '<absolute_path_to_jks>',
   'format.type'                                       = 'csv'
)
```

Important: The Kerberos related properties should be passed to the run command when submitting your job:

```
flink run -m yarn_cluster -d \
-yD security.kerberos.login.principal=<principal> \
-yD security.kerberos.login.keytab=<local_path_to_keytab> \
-c com.cloudera.streaming.examples.flink.KafkaSecureIT \
flink-sql-tests-1.0-SNAPSHOT.jar
```

Data types for Kafka connector

When reading data using the Kafka table connector, you must specify the format of the incoming messages so that Flink can map incoming data to table columns properly.

JSON format

The JSON format enables you to read and write JSON data. You must add the JSON dependency to your project and define the format type in CREATE table to JSON.

The expected JSON schema will be derived from the table schema by default. Specifying the JSON schema manually is not supported.

Maven dependency

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-json</artifactId>
  <version>1.19.1-csa1.13.0.0</version>
</dependency>
```

For more information about the JSON format, see the Apache Flink documentation.

The following example shows the Kafka connector with JSON data type:

```
CREATE TABLE source_table (
  c1 INT,
  c2 STRING,
  c3 DECIMAL(38,18),
  c4 TIMESTAMP(3),
) WITH (
   'connector.type'      = 'kafka',
 'connector.version'   = 'universal',
 'connector.topic'     = 'input_topic',
 'connector.startup-mode' = 'latest-offset',
 'connector.properties.bootstrap.servers' = '<hostname>:<port>',
 'connector.properties.group.id' = 'test',
 'format.type' = 'json'
);
```

CSV format

The CSV format allows your applications to read data from, and write data to different external sources in CSV. You must add the CSV dependency to your project and define the format type in CREATE table to CSV.

The CSV format will derive format schema from the table schema by default. The format schema can be defined with Flink types also, but this functionality is not supported yet.

Maven dependency

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-csv</artifactId>
  <version>1.19.1-csa1.13.0.0</version>
</dependency>
```

For more information about the JSON format, see the Apache Flink documentation.

The following example shows the Kafka connector with CSV data type:

```
CREATE TABLE source_table (
  c1 INT,
  c2 STRING,
  c3 TIMESTAMP(3)
) WITH (
   'connector.type'                          = 'kafka',
   'connector.version'                       = 'universal',
   'connector.topic'                         = 'sink_topic',
   'connector.properties.bootstrap.servers'  = '<host>:<port>',
   'connector.properties.group.id'           = 'test',
   'format.type'                             = 'csv'
)
```

Avro format

The Apache Avro format enables you to read and write Avro data. You must add the Avro dependency to your project and define the format type in CREATE table to Avro. You also need to specify the fields of the Avro record within the table.

The format schema can be defined either as a fully qualified class name of an Avro specific record or as an Avro schema string. If a class name is used, the class must be available in the classpath during runtime.

When using the Avro schema string, you must specify the fields of the Avro record. The schema must correspond to the schema of the table in Flink.

For a detailed description of Avro schemas, see the Apache Avro documentation.

Maven dependency

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-avro</artifactId>
  <version>1.19.1-csa1.13.0.0</version>
</dependency>
```

The following example shows how to use an Avro schema string when creating a Kafka connector table. It is specified as a JSON object, having record as type, a name, and the specification of its fields. Note the correspondence of various data types, especially the decimal and array fields.

Example

```
CREATE TABLE source_table(
 string_field STRING,
 long_field    BIGINT,
 decimal_field  DECIMAL(38,18),
     int_arr_field  ARRAY<INT>
) WITH (
 'connector.type'                          = 'kafka',
 'connector.version'                       = 'universal',
 'connector.topic'                         = 'input_topic',
     'connector.properties.group.id'           = 'test',
 'connector.properties.bootstrap.servers'  = '<hostname>:<port>'
 'format.type'                             = 'avro',
     'format.avro-schema' =
   '{
        "type": "record",
        "name": "test",
        "fields" : [
       {"name": "string_field", "type": "string"},
       {"name": "long_field",   "type": "long"},
       {"name": "decimal_field", "type":
{"type": "bytes",
```

```
 "logicalType": "decimal",
 "precision": 38,
 "scale": 18}},
           {"name": "int_arr_field", "type":
{"type":"array",
 "items":"int"}}
      ]
    }'
)
```

Supported basic data types

Before creating your table with Avro format for the Kafka connector, you should review the supported basic data types that you can use in the application.

Avro schema type	Flink data type
string	STRING
boolean	BOOLEAN
bytes	BYTES
int	INT
long	BIGINT
float	FLOAT
double	DOUBLE

Note: Time, Date, Timestamp are not yet supported.

To specify a field with additional properties, such as the decimal or array fields in the example, the type field must be a nested object which has a type field itself, as well as the needed properties.

An example of a property that must be set this way is a field's logical type. Some types cannot be directly represented by an Avro data type, so they use one of the supported types as an underlying representation. The logical type attribute tells how it should be interpreted. For example, the decimal type – described below – is stored as bytes, while its logical type is decimal.

Decimal type

- Only 38 precision and 18 scale are supported
- Flink data type is DECIMAL(38,18)
- To specify in the Avro schema: {"name": "decimal_field", "type": {"type": "bytes", "logicalType": "decimal", "precision": 38, "scale": 18}}

Array type

Avro allows arrays of supported basic types, except:

- String
- Decimal
- Constructed types (nesting of arrays, rows, maps)

For example, defining an Array of long values:

- In the table definition: arr_field ARRAY<BIGINT>
- Avro schema:

```
{“name”:”arr_field, "type": {"type":"array",
    "items":"long"}}}
```

Row type

Flink rows can be specified as records in the Avro schema. Fields must be named both in the SQL of the table definition, as well as in the Avro schema string.

- Field names must match between the table declaration and the Avro schema's record description.
- The two name fields in the Avro schema have the following structure:
  - one on the outside is the name of the field
  - one inside is the type object, pertaining to the record definition
- Decimal fields are not supported within rows.
- Rows can be nested, Arrays are also allowed as fields of the Row.

Example table definition:

```
CREATE TABLE source(row_field ROW<f1 INT,f2 STRING,f3 BOOLEAN>)
WITH (...)
```

Corresponding Avro schema:

```
'format.avro-schema' =
     '{
      "type": "record",
      "name": "test",
      "fields" : [
        {"name": "row_field", "type": {
            "type":"record",
            "name":"row_field",
            "fields": [
            {"name":"f1", "type":"int"},
            {"name":"f2", "type":"string"},
            {"name":"f3", "type":"boolean"}]
  }}]
}'
```

Map type

Only Maps with String keys are supported. The value field can be any type of the supported ones, except decimal.

- In the table definition: map_field MAP<STRING,BIGINT>
- In the Avro schema:

```
{"name": "map_field", "type": {"type":"map",
    "values":"long"}}
```

Nullability

To set a field nullable in the Avro schema, create a union of the field's type and null. A nullable integer field would be defined as: {"name": "int_field", "type": ["int", "null"]}

Schema Registry formats

You can avoid defining the Avro and JSON schema for Kafka table sources and sinks, when the schema is stored in Cloudera Schema Registry.

Such topics are accessible through automatically generated tables from the read-only registry catalog.

Maven dependency

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-cloudera-registry</artifactId>
  <version>1.0-csa1.13.0.0</version>
```

```
</dependency>
```

If you need to define a table outside the registry catalog, the following example can be used:

```
CREATE TABLE source_table (
   id BIGINT,
   name STRING,
   description STRING
) WITH (
   'connector.type' = 'kafka',
   'connector.version' = 'universal',
   'connector.topic' = 'message',
   'connector.startup-mode' = 'latest-offset',
   'connector.properties.bootstrap.servers' = '<hostname>:<port>',
   'connector.properties.group.id' = 'test',
   'format.type' = 'registry',
   'format.registry.properties.schema.registry.url' = 'http(s)://
<hostname>:<port>/api/v1'
)
```

Cloudera Schema Registry connector for Flink stores the schema version info in the Kafka messages by default. This means that the format.registry.properties.store.schema.version.id.in.header property is set to false by default.

The schema name in the registry is usually the same as the Kafka topic name, but can be overridden by the format.registry.schema-name property.

Note: The schema used in this case must be already registered in the Schema Registry, otherwise an error will occur.

SQL Statements in Flink

You can use the CREATE / ALTER / INSERT / DROP statements to modify objects in the chosen catalog. The statements are executed with the sqlUpdate() method of the TableEnvironment.

```
...
StreamTableEnvironment tableEnv = StreamTableEnvironment.create(env, setting
s);
tableEnv.sqlUpdate("CREATE TABLE t1(c1 STRING) WITH (...)");
tableEnv.sqlUpdate("DROP TABLE t1");
...
```

CREATE Statements

You can use CREATE statements to register database, table, and function objects into catalogs. You should add the connector, the name of the table, the schema and the data format to the statement based on your application design. You can further customize your table statement with computed columns to reflect time in a Flink application.

For more information about CREATE statements, see the Apache Flink documentation.

CREATE TABLE

You can connect the Flink Table API and the Flink SQL programs with external systems to read and write streaming tables. The table declaration is similar to a SQL CREATE TABLE statement. The followings can be defined upfront for connecting to an external system:

- Name of the table
- Schema of the table
- Connector
- Data format

You can see an example of the defined parameters:

```
tableEnvironment.sqlUpdate(
    "CREATE TABLE MyTable (\n" +
    "  ...    -- declare table schema \n" +
    ") WITH (\n" +
    "  'connector.type' = '...',  -- connector specific properties\n" +
    "  ...\n" +
    "  'update-mode' = 'append',  -- declare update mode\n" +
    "  'format.type' = '...',     -- format specific properties\n" +
    "  ...\n" +
    ")");
```

Computed column and watermark

A computed column is a virtual column that is generated using the syntax “column_name AS computed_column _expression”. Computed columns are commonly used in Flink for defining time attributes in CREATE TABLE statements.

The WATERMARK defines the event time attributes of a table, and allows computed columns to calculate the watermark in the following form: WATERMARK FOR rowtime_column_name AS watermark_strategy_expres sion. The expression return type must be TIMESTAMP(3).

```
CREATE TABLE ItemTransactions (
 transactionId    BIGINT,
 ts    BIGINT,
 itemId    STRING,
 quantity INT,
 event_time AS CAST(from_unixtime(floor(ts/1000)) AS TIMESTAMP(3)),
 WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
) WITH (
 'connector.type'      = 'kafka',
 'connector.version'   = 'universal',
 'connector.topic'     = 'transaction.log.1',
 'connector.startup-mode' = 'earliest-offset',
 'connector.properties.bootstrap.servers' = '<hostname>:<port>',
 'connector.properties.group.id' = 'test',
 'format.type' = 'json'
);
```

CREATE DATABASE

```
tableEnvironment.sqlUpdate("CREATE DATABASE sample_database");
tableEnvironment.useDatabase("sample_database");
```

CREATE FUNCTION

```
package com.cloudera.udfs;
public static class HashCode extends ScalarFunction {
  public int eval(String s) {
     return s.hashCode();
  }
}
tableEnvironment.sqlUpdate("CREATE FUNCTION hashcode AS
'com.cloudera.udfs.HashCode");
```

DROP Statements

You can remove the database, table, and function objects from catalogs with DROP statements. You should also include the IF EXIST statement beside the DROP statement to avoid errors due to non existing objects.

Use IF EXISTS statement to avoid errors on non-existing objects.

For more information about DROP statements, see the Apache Flink documentation.

DROP TABLE/DATABASE/FUNCTION

```
tableEnv.sqlUpdate("DROP TABLE|DATABASE|FUNCTION [IF EXISTS] object_name");
```

ALTER Statements

You can use the ALTER statements to modify already registered databases, tables, and function definitions in the chosen catalogs. There are different limitations for databases, tables and functions when altering them.

For more information about ALTER statements, see the Apache Flink documentation.

ALTER DATABASE

Database properties can be changed, but databases cannot be renamed. You can use DROP and CREATE instead of renaming.

```
tableEnv.sqlUpdate("CREATE DATABASE sample_database");
tableEnv.sqlUpdate("ALTER DATABASE sample_database SET ('key1'='value1')");
```

ALTER TABLE

Tables can be renamed and table properties can also be changed.

```
tableEnv.sqlUpdate("CREATE TABLE sample_table(c1 STRING) WITH ('key1'='value
1')");
tableEnv.sqlUpdate("ALTER TABLE sample_table SET ('key1'='value2')");
tableEnv.sqlUpdate("ALTER TABLE sample_table RENAME TO sample_table2");
```

ALTER FUNCTION

New identifiers, which are full classpath for JAVA/SCALA objects, can be assigned to registered functions.

```
tableEnv.sqlUpdate("CREATE FUNCTION myudf AS 'com.example.MyUdf'");
tableEnv.sqlUpdate("ALTER FUNCTION myudf AS 'com.example.MyUdf'");
```

INSERT Statements

Data can be inserted into sink tables in various ways. You can use the INSERT clause to insert the query result into a table, or you can use the VALUE clause to insert data into tables directly from SQL.

You can use the INSERT clause with different source tables, or the VALUES clause. The following examples show how to use INSERT and VALUES at the same time:

```
tableEnv.sqlUpdate("CREATE TABLE source_table (c1 STRING)  WITH (...)");
tableEnv.sqlUpdate("CREATE TABLE sink_table (c1 STRING)  WITH (...)");
tableEnv.sqlUpdate("INSERT INTO source_table VALUES ('foo')");
Table elements = tableEnv.fromDataStream(env.fromElements("bar"));
tableEnv.sqlUpdate("INSERT INTO source_table SELECT f0 from " + elements);
tableEnv.sqlUpdate("INSERT INTO sink_table SELECT * from source_table");
Table table = tableEnv.sqlQuery("SELECT * FROM sink_table");
tableEnv.toAppendStream(table, Row.class).printToErr();
```

For more information about INSERT statements, see the Apache Flink documentation.

Note: The following composite types are supported:

- To construct an array in the SQL insert statement: ARRAY[1,2,3]
- To construct a row: ROW[1,'asd',1.2]
- To construct a map: MAP['key1','val1','key2','val2']
- If you get an error such as Unsupported cast from 'MAP' to 'MAP', (or ROW to ROW, ARRAY to ARRAY), it can be because the inner types are not matching. Make sure that when inserting, you properly cast the values that build up the composite type. Example: INSERT INTO t VALUES (map['www', cast(13 as bigint)])

SQL Queries in Flink

A Table can be used for subsequent SQL and Table API queries, to be converted into a DataSet or DataStream, and to be written to a TableSink. You need to specify the SELECT queries with the sqlQuery() method of the TableEnvironment to return the result of the SELECT query as a Table.

SQL and Table API queries can be seamlessly mixed, and are holistically optimized and translated into a single program.

In order to access a Table in a SQL query, it must be registered in the TableEnvironment. A Table can be registered from the following ways:

- TableSource
- Table
- CREATE TABLE statement
- DataStream
- DataSet

Alternatively, users can also register catalogs in a TableEnvironment to specify the location of the data sources.

The following is an example of SQL query in Java:

```
DataStream<Tuple2<String, Integer>> transactionStream = ...
tEnv.createTemporaryView("Transactions", transactionStream, "account, amoun
t");

Table balance = tEnv.sqlQuery(
  "SELECT account, sum(amount) as balance FROM Transactions GROUP BY accoun
t"
);

DataStream<Tuple2<Boolean, Row>> balanceStream = tEnv.toRetractStream(bal
ance, Row.class);
```

For the detailed documentation and the example code for the different query types, see the Apache Flink documentation.

Note: For information about the supported queries in SQL and Table API, see the Supported Features.

Flink metadata collection using Atlas

In Cloudera Streaming Analytics, you can use Flink with Apache Atlas to track the input and output data of your Flink jobs.

Atlas is a lineage and metadata management solution that is supported across Cloudera. This means that you can find, organize and manage different assets of data about your Flink applications and how they relate to each other. This enables a range of data stewardship and regulatory compliance use cases.

For more information about Atlas, see the Cloudera Runtime documentation.

Related Information

Create Atlas entity type definitions

Atlas entities in Flink metadata collection

In Atlas, the core concept of representing Flink applications, Kafka topics, HBase tables, and so on, is called an entity. You need to understand the relation and definition for entities in a Flink setup to enhance the metadata collection.

When submitting updates to Atlas, a Flink application describes itself and the entities it uses as sources and sinks. Atlas creates and updates the corresponding entities, and creates lineage from the collected and already available entities. Internally, the communication between the Flink client and the Atlas server is implemented using a Kafka topic. This solution is referred to as Flink hook by the Atlas community.

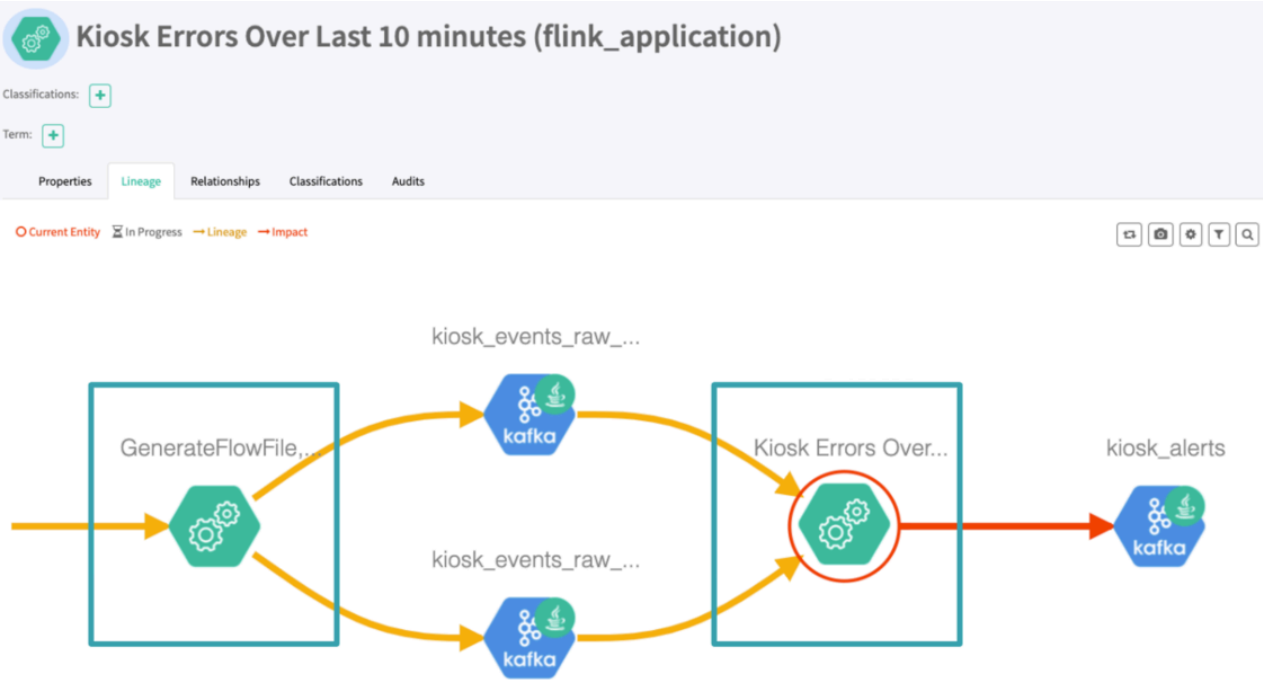

Related Information

Create Atlas entity type definitions

Creating Atlas entity type definitions for Flink

Before submitting Flink jobs to collect their metadata, you need to create Atlas entity type definitions for Flink. In the command line, you need to connect to the atlas server and add the predefined type definitions. You also need to enable Atlas for Flink in Cloudera Manager.

About this task

Atlas does not include the metadata source of Flink by default. The administrator must manually upload the entity type definitions to the cluster to be able to start the Flink metadata collection.

Note: The default ports for the Atlas admin server are 31433 and 31000 when TLS is enabled or disabled respectively.

Procedure

1. Upload the designed entity type definitions to the cluster using the Atlas REST API.

```
curl -k -u <atlas_admin>:<atlas_admin_pwd> --location --request POST 'ht
tps://<atlas_server_host>:<atlas_server_port>/api/atlas/v2/types/typedefs'
 \
--header 'Content-Type: application/json' \
--data-raw '{
    "enumDefs": [],
    "structDefs": [],
    "classificationDefs": [],
    "entityDefs": [
        {
            "name": "flink_application",
            "superTypes": [
                "Process"
            ],
            "serviceType": "flink",
            "typeVersion": "1.0",
            "attributeDefs": [
                {
                    "name": "id",
                    "typeName": "string",
                    "cardinality": "SINGLE",
                    "isIndexable": true,
                    "isOptional": false,
                    "isUnique": true
                },
                {
                    "name": "startTime",
                    "typeName": "date",
                    "cardinality": "SINGLE",
                    "isIndexable": false,
                    "isOptional": true,
                    "isUnique": false
                },
                {
                    "name": "endTime",
                    "typeName": "date",
                    "cardinality": "SINGLE",
                    "isIndexable": false,
                    "isOptional": true,
                    "isUnique": false
                },
                {
                    "name": "conf",
                    "typeName": "map<string,string>",
                    "cardinality": "SINGLE",
                    "isIndexable": false,
                    "isOptional": true,
                    "isUnique": false
                }
            ]
        }
    ],
    "relationshipDefs": []
}'
```

2. Log in to Cloudera Manager.
3. Go to Flink>Configuration.
4. Search for 'enable atlas' in the search bar.
5. Enable Atlas Metadata Collection.

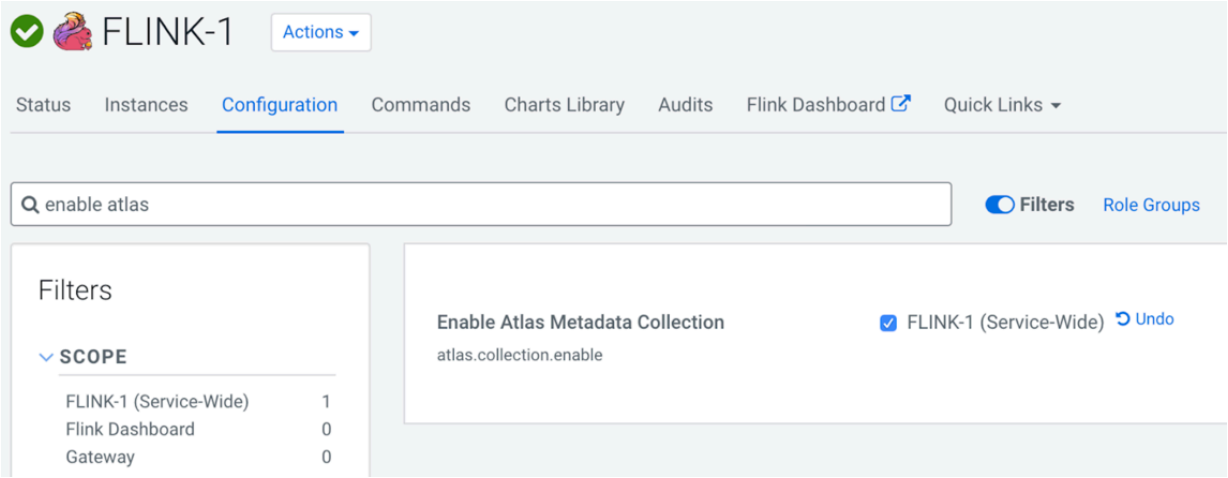

Results

The Flink client notifies Atlas about the metadata of the job on successful submission.

Verifying metadata collection

After enabling Atlas metadata collection, newly submitted Flink jobs on the cluster are also submitting their metadata to Atlas. You can verify the metadata collection with messages in the command line by requesting information regarding the Atlas hook.

To verify the metadata collection, you can run the "Streaming WordCount" example from *Running a simple Flink application*.

In the log, the following new lines appear:

```
...
20/05/13 06:28:12 INFO hook.FlinkAtlasHook: Collecting metadata for a new Fl
ink Application: Streaming WordCount
...
20/05/13 06:30:35 INFO hook.AtlasHook: <== Shutdown of Atlas Hook
```

Flink communicates with Atlas through a Kafka topic, by default the one named ATLAS_HOOK.

Migrating Flink jobs

After upgrading the version of the Data Lake in your Cloudera Public Cloud environment, you need to create Cloudera Data Hub clusters using the Cloudera Streaming Analytics cluster template that corresponds to the new version of the Data Lake. To restart your existing Flink jobs on the upgraded Cloudera Data Hub clusters, you need to migrate your Flink jobs. The migration process depends on the state of the Flink jobs.

After creating a new Cloudera Data Hub cluster using the latest Cloudera Streaming Analytics cluster template, you need to migrate your Flink jobs to the new cluster. The process of migration depends on the state of your Flink jobs: if you have stateful jobs, you need to stop your Flink applications with creating savepoints. Before starting the job migration, make sure that the job dependies are updated based on the latest version of Flink in Cloudera Public Cloud.

Migrating Flink jobs without state

When you run Flink application without state, you only need to stop your currently running Flink jobs, and restart them on your upgraded Cloudera Data Hub clusters.

Procedure

1. Connect to your host where you run your Flink jobs using ssh.

```
ssh <WORKLOAD_USERNAME>@<MASTER_HOST>
Password: <YOUR_WORKLOAD_PASSWORD>
```

2. Determine which jobs you want to stop by listing the Flink job IDs.

```
flink list
```

3. Stop your Flink jobs using command line interface (CLI).

```
flink cancel <FLINK_JOB_ID>
```

4. Resubmit the stopped Flink jobs on your new cluster.
   a) Connect to your host where you intend to run your Flink jobs using ssh.

```
ssh <WORKLOAD_USERNAME>@<MASTER_HOST>
Password: <YOUR_WORKLOAD_PASSWORD>
```

   b) Resubmit your Flink application.

```
flink run <RUN_ARGUMENTS> \
  <JAR_FILE> <APP_ARGUMENTS>
```

Migrating stateful Flink jobs

When you run Flink application with state, you must stop the Flink jobs with a savepoint, so you can restart them from the exact point on your upgraded Data Hub clusters. Based on the version of your original cluster, you can provide the savepoint path either in HDFS or S3.

About this task

The following savepoint paths are supported based on the version of your original cluster:

- If the version of the original cluster was 7.2.9 or lower, you need to provide the savepoint path in HDFS.
- If the version of the original cluster was 7.2.10 or higher, you need to provide the savepoint path in S3.

Procedure

1. Connect to your host where you run your Flink jobs using ssh.

```
ssh <WORKLOAD_USERNAME>@<MASTER_HOST>
Password: <YOUR_WORKLOAD_PASSWORD>
```

2. Determine which jobs you want to stop by listing the Flink job IDs.

```
flink list
```

3. Stop your Flink jobs with a savepoint using command line interface (CLI).

   If the version of your original cluster was 7.2.10 or higher, store the savepoint in S3:

   ```
   flink stop --savepointPath s3a://<BUCKET_NAME>/savepoints <flink_job_id>
   ```

   If the version of your original cluster was 7.2.9 or lower, store the savepoint in HDFS:

   ```
   flink stop --savepointPath hdfs:///tmp/savepoints <FLINK_JOB_ID>
   ```

4. Take note of the savepoint path in the output of the command as you must provide the path when resuming the Flink applications.
5. Copy the savepoint from HDFS to S3, only if the version of your original cluster was 7.2.9 or lower.
   a) If the savepoint size is minimal:

   ```
   hdfs dfs -mkdir -p s3a://aa-uet2/savepoints/
   hdfs dfs -cp <savepoint_path> s3a://<bucket_name>/savepoints/
   ```

   b) If the savepoint size is large (several GB or larger):

   ```
   hdfs dfs -mkdir -p s3a://aa-uet2/savepoints/
   hadoop distcp -m 10 <savepoint_path> s3a://<bucket_name>/savepoints/
   ```

6. Create the Cloudera Data Hub cluster with the new version.

   For more information, see the *Creating Cloudera Streaming Analytics cluster* section.
7. Copy the savepoint from S3 to HDFS, only if the version of your original cluster was 7.2.9 or lower.

   Make sure that you provide the same path on HDFS as the original savepoint location in the original cluster.
   a) If the savepoint size is minimal:

   ```
   hdfs dfs -cp \
     s3a://<BUCKET_NAME>/savepoints/<SAVEPOINT_FOLDER_NAME> \
     hdfs:///tmp/savepoints/
   ```

   b) If the savepoint size is large (several GB or larger):

   ```
   hadoop distcp -m 10 \
     s3a://<BUCKET_NAME>/savepoints/<SAVEPOINT_FOLDER_NAME> \
     hdfs:///tmp/savepoints/
   ```

8. Update the Flink job dependencies in your POM file, and rebuild your JAR file.

   For more information, see the *Updating Flink job dependencies* section.
9. Resume your Flink application from savepoint.

   If the version of your original cluster was 7.2.10 or higher, restore your application from S3:

   ```
   flink run <RUN_ARGUMENTS> \
     -fromSavepoint s3a://<BUCKET_NAME>/savepoints \
     <JAR_FILE> <APP_ARGUMENTS>
   ```

   If the version of your original cluster was 7.2.9 or lower, restore your application from HDFS:

   ```
   flink run <RUN_ARGUMENTS> \
     -fromSavepoint hdfs:///tmp/savepoints/<SAVEPOINT_NAME> \
     <JAR_FILE> <APP_ARGUMENTS>
   ```

Updating Flink job dependencies

When you migrate your Flink jobs to a cluster that has a new supported version of Flink, the applications need to use a new version of the artifacts provided by the Flink deployment in your cluster. To avoid incompatibilities between the packaged artifacts of your application and the artifacts provided by the Flink cluster, ensure that the POM file of the application is updated to match the Flink version of the new Data Hub cluster.

Procedure

1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
2. Select the Streaming Analytics cluster from the list of Data Hub clusters.
3. Access the latest Flink version of your cluster.
   a) Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Repository Details.
   d) Search for the **Cloudera Runtime Repository Specification** section, and review the Flink version.

      Name: FLINK

      Version: 1.13.2-csadh1.5.0.0-cdh7.2.12.0-35-17794544
4. Update the <flink.version> property of your POM file using the Flink and Cloudera Streaming Analyitcs (CSA) version of your Data Hub cluster.

   You need to copy and paste only the prefix before the 'cdh' version number: 1.13.2-csadh1.5.0.0.

```
<properties>
        ...
        <flink.version>1.13.2-csadh1.5.0.0</flink.version>
        ...
    </properties>
    ...
    <dependencies>
        ...
        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-java</artifactId>
            <version>${flink.version}</version>
        </dependency>
        ...
    <dependencies>
```

5. Rebuild your JAR file.

```
mvn clean package
```

Reference

Flink Terminology

The list of Flink terminology details the Flink specific terms that are used in the Cloudera Streaming Analytics documentation.

Event

An event is a statement about a change of the state of the domain modelled by the application. Events can be input and/or output of a stream or batch processing application. Events are special types of records.

Function

Functions are implemented by the user and encapsulate the application logic of a Flink program. Most Functions are wrapped by a corresponding Operator.

Flink Application

A Flink application is a Java Application that submits one or multiple Flink Jobs from the main() method. Submitting jobs is usually done by calling execute() on an execution environment.

Flink Job

A Flink Job is the runtime representation of a logical graph (also often called dataflow graph) that is created and submitted by calling execute() in a Flink Application.

Flink JobManager

The JobManager is the orchestrator of a Flink Cluster. It contains three distinct components: Flink Resource Manager, Flink Dispatcher and one Flink JobMaster per running Flink Job.

Logical Graph

A logical graph is a directed graph where the nodes are Operators and the edges define input/output-relationships of the operators and correspond to data streams or data sets. A logical graph is created by submitting jobs from a Flink Application. Logical graphs are also often referred to as dataflow graphs.

Operator

Node of a Logical Graph. An Operator performs a certain operation, which is usually executed by a Function. Sources and Sinks are special Operators for data ingestion and data egress.

Flink Session Cluster

A long-running Flink Cluster which accepts multiple Flink Jobs for execution. The lifetime of this Flink Cluster is not bound to the lifetime of any Flink Job.

State Backend

For stream processing programs, the State Backend of a Flink Job determines how its state is stored on each TaskManager (Java Heap of TaskManager or (embedded) RocksDB) as well as where it is written upon a checkpoint (Java Heap of JobManager or Filesystem).

Task

Node of a Physical Graph. A task is the basic unit of work, which is executed by Flink's runtime. Tasks encapsulate exactly one parallel instance of an Operator or Operator Chain.

Flink TaskManager

TaskManagers are the worker processes of a Flink Cluster. Tasks are scheduled to TaskManagers for execution. They communicate with each other to exchange data between subsequent Tasks.

Transformation

A Transformation is applied on one or more data streams or data sets and results in one or more output data streams or data sets. A transformation might change a data stream or data set on a per-record basis, but might also only change its partitioning or perform an aggregation. While Operators and Functions are the "physical" parts of Flink's API, Transformations are only an API concept. Specifically, most transformations are implemented by certain Operators.

For the complete list of Flink terminology, see the Apache documentation.

Cloudera Flink Tutorials

The Cloudera Flink Tutorials walks you through the basic steps to create a Stateless Monitoring, a Stateful Inventory and a Secure application using Flink.

For newcomers, Cloudera recommends starting with the Simple Tutorial.

The Simple Tutorial details the following steps:

- Basic structure of a Flink application
- Logging with Kafka
- Job submission
- Alerting functionality

For a more advanced application, you can use the Stateful Tutorial to get familiar with using state and windowing.

The Stateful Tutorial details the following steps:

- Using state within the application
- Windowing function
- Generating data from Kafka
- Production configuration

The Secure Tutorial serves as a first step to learn everything about the Flink security features within Cloudera.

The Secure Tutorial details the following steps:

- Securing Kafka
- Security parameters in a Flink job
- Kafka Metrics Reporter
- Integration with Schema Registry

Cloudera DataFlow for Data Hub 7.3.1

Using Cloudera SQL Stream Builder

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Running a simple SQL job 6

Project structure and development 7
- Creating a project 8
- Navigating in a project 10
- Managing member of a project 10
- Source control of a project 11
  - Masking information before using source control 13
- Setting the environment for a project 13
- Importing a project 16

Registering Data Sources in Cloudera SQL Stream Builder 17
- Adding Kafka Data Source 17
- Adding Catalogs 21
- Using auto discovery of services 28
  - Setting up the service discovery 28
  - Using the service discovery on Streaming SQL Console 31

Connectors 33
- Using connectors with templates 34
- Adding new connectors 34
- Kafka connectors 35
- CDC connectors 37
- JDBC connector 38
- Filesystem connector 38
- Datagen connector 38
- Faker connector 39
- Blackhole connector 39

Data formats 39
- Adding data formats 39

Concept of tables in Cloudera SQL Stream Builder 40
- Creating Kafka tables using wizard 41
  - Configuring Kafka tables 44
- Creating Webhook tables 53
- Creating Flink tables using Templates 55
- Creating Iceberg tables 56

SQL jobs 58
- Creating and naming SQL jobs 58
- Running SQL Stream jobs 60
- Configuring SQL job settings 62

Adjusting logging configuration in Advanced Settings....64
Configuring YARN queue for SQL jobs....68
Configuring state backend for Cloudera SQL Stream Builder....68
Managing session for SQL jobs....71
Executing SQL jobs in production mode....72

Functions....73
Creating Python User-defined Functions....73
Creating Javascript User-defined Functions....76
Developing JavaScript functions....78
Creating Java User-defined functions....79
Using System Functions....79

Introduction to Materialized Views....80
Creating Materialized Views....81
Configuring Retention Time for Materialized Views....86
Materialized View Pagination....88
Using Dynamic Materialized View Endpoints....88
Configuring Materialized View database information....91
Using Cloudera SQL Stream Builder with Cloudera Data Visualization....92

Using widgets for data visualization....93
Creating widgets....94
Choosing data sources....96
Managing data source jobs....98
Customizing visualization types....100
Configuring widget, fields and legend....100
Configuring graph style....105
Managing widgets on the Dashboard....126

Using SQL job notification....131

Using Cloudera SQL Stream Builder REST API....136

Monitoring SQL Stream jobs....139
Collecting diagnostic data....141

SSB metadata collection using Atlas....142

Flink SQL....142
Flink DDL....142
Managing time in Cloudera SQL Stream Builder....142
Flink DML....143
Flink Queries....144
Other supported statements....144
Data Types....145
Dynamic SQL Hints....145
SQL Examples....146

Enriching streaming data with join.....148
Joining streaming and bounded tables.....149
Example: joining Kafka and Kudu tables.....150
Updating SQL queries with PROCTIME function.....151

Running a simple SQL job

You can use this Getting Started use case to get familiar with the most simple form of running a SQL Stream job.

About this task

The Getting Started contains the basic steps of running a SQL Stream job. When executing the job, you do not need to select a sink as the results are displayed in the browser. Cloudera SQL Stream Builder provisions a job on your cluster to run the SQL queries. You can select the **Logs** tab to review the status of the SQL job. As data is returned, it shows up in the **Results** tab.

Note: Before using Cloudera SQL Stream Builder, review the *Cloudera Streaming Analytics* sections in the Release Notes as limitations may apply to the current version of Cloudera Streaming Analytics.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click ⋮ next to **Jobs** from the **Explorer**.
4. Select New Job > Create .

   The SQL Editor for the created job opens in a tab.
5. Select *DATAGEN* from the Templates.

   The CREATE TABLE statement is imported to the SQL window.
6. Add a SELECT query to the SQL window after the datagen template.

   ```
   SELECT * FROM <table_name>
   ```

   Note: When adding SQL statements to the SQL window, you do not need to add the semicolon (;) at the end of the statement as Cloudera SQL Stream Builder can run the command without the semicolons.
7. Click Execute.
   You can see the generated output on the Results tab.

8. Click on the Stop button to stop the job.

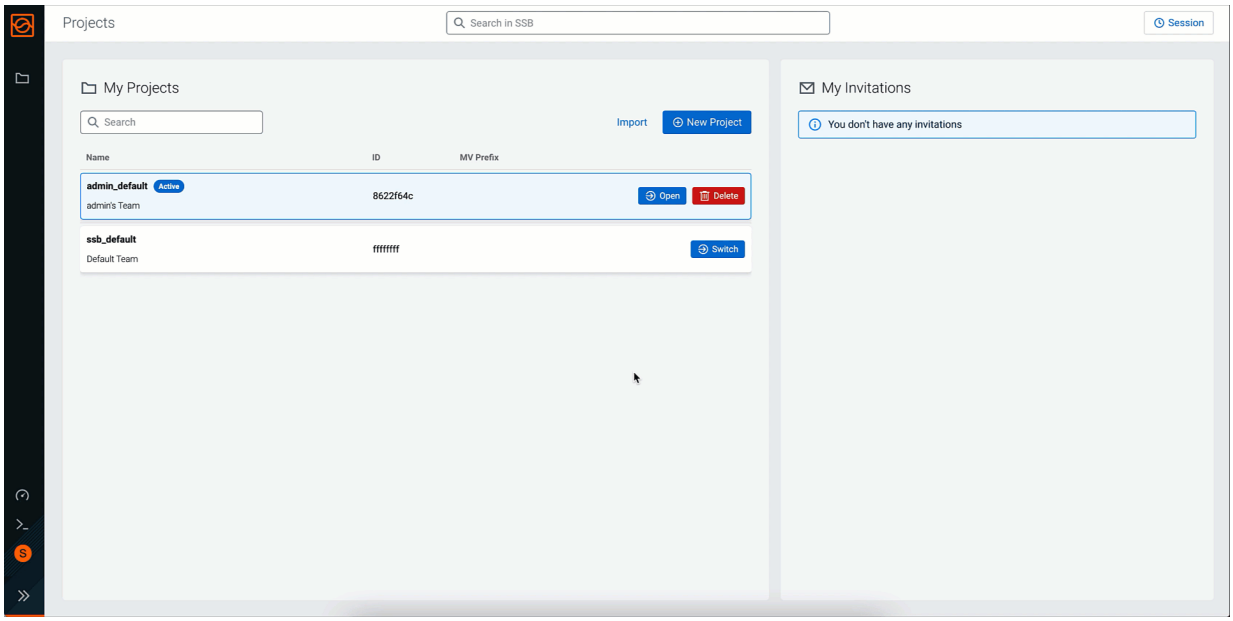

Project structure and development

Projects aim to provide a Software Development Lifecycle (SDLC) for streaming applications in Cloudera SQL Stream Builder: they allow developers to think about a task they want to solve using Cloudera SQL Stream Builder, and collect all related resources, such as job and table definitions or data sources in a central place.

Projects aim to facilitate collaboration between developers by sharing common resources among its members. Projects can be synchronized with a Git repository, allowing easy migration between different clusters. The environment concept in a project allows the templating of cluster-specific or sensitive properties.

A project is a collection of resources, static definitions of data sources, jobs with materialized views, virtual tables, user-defined functions (UDF), and materialized view API keys. These resources are called internal to a project and can be safely used by any job within the project.

Jobs can also use external resources that the logged in user has access to. These resources are defined in other projects (for example, a UDF or a virtual table in another project), or can come from a Data Source (for example, a Kudu table from the Kudu catalog). External resources are outside the scope of the project, thus they can be changed or altered by external factors. It is recommended to use only internal resources for projects that are intended to be exported.

The following table summarizes the concepts and structural elements of projects in Cloudera SQL Stream Builder:

Project specific resources	
Jobs	Cloudera SQL Stream Builder job definitions including SQL, Materialized View configuration and endpoints, Job settings such as checkpointing and parallelism. The status of the job and its associated Flink job (if any) are not part of the definition.
Functions	User-defined JavaScript functions.
Virtual Tables	Virtual tables of the project. These are stored in the *<PROJECT_NAME>* database of the ssb catalog. When exporting a project, only these tables (ssb.<project_name>.*) are exported.
Data Sources	Data Sources include Kafka providers and Catalogs in the project. While the definition of the Data Source is considered internal to the project, the tables or topics provided by them are external, as they are managed by an outside system.

Materialized Views	Materialized Views that have been defined for Jobs in the project.
API Keys	API Keys for accessing Materialized View endpoints.
Job Notifications	Job Notification actions (webhook and email notifications) as well as notification groups can be defined in a project. They may be assigned to Jobs to be triggered by failure. Job notifications are not synchronized when using the Git import/export feature.
External resources	
Virtual Tables	Virtual tables from other projects and catalogs that the logged in user has access to. Other members of the active project might not have access to these.
Connectors	Connectors available in Cloudera SQL Stream Builder for connecting to external systems. These are shared among the whole SSB instance, and are accessible to all users in all projects.
Data Formats	Data Formats available in Cloudera SQL Stream Builder that can be used by connectors when connecting to external systems. These are shared among the whole Cloudera SQL Stream Builder instance, and are accessible to all users in all projects.

When you access the Streaming SQL Console, there are already existing projects on the **My Projects** page. The ssb_default and [***USERNAME***]_default projects are automatically generated projects. The ssb_default project and its resources are visible to every user, while the [***USERNAME***]_default project and its resources are only visible to that user. Every user who created an account in Cloudera SQL Stream Builder has its own project generated by default where members can be invited.

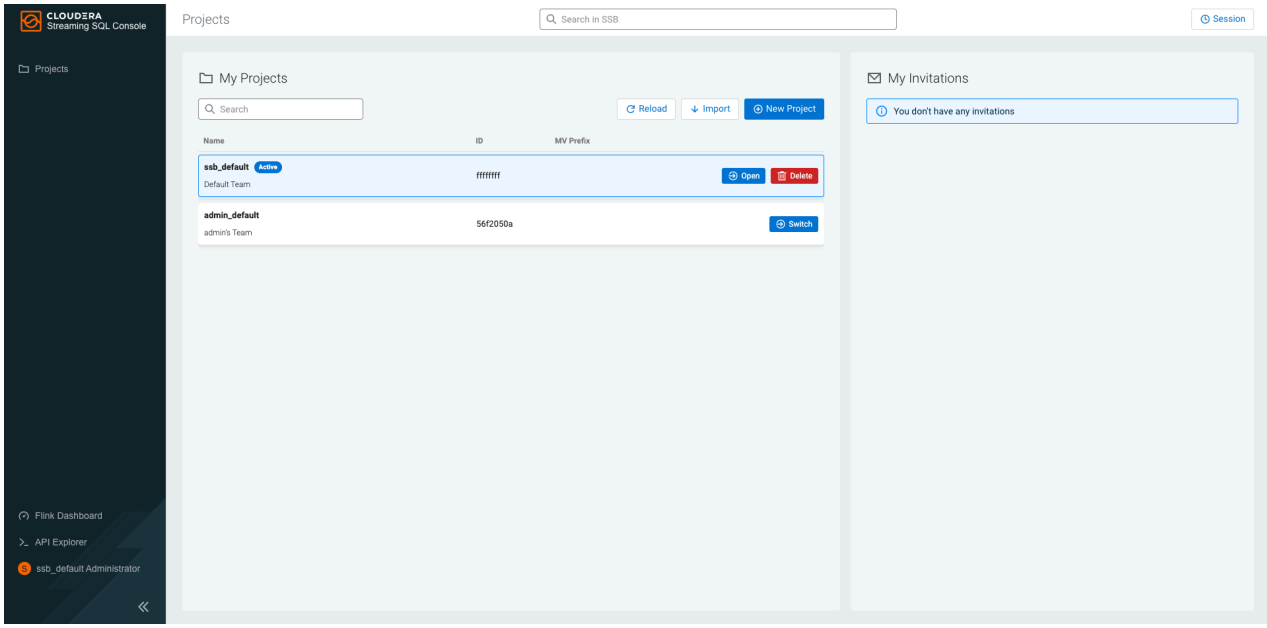

Creating a project

Before you can use Streaming SQL Console, you need to create a project where you can submit your SQL jobs and manage your project resources.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.
2. Click New Project on the **Projects** page of Streaming SQL Console.

   The **Create Project** window appears.
3. Provide a Name to the project.
4. Optional: Provide a Description to the project.
5. Optional: Provide a prefix that overrides the Materialized View table names.

   Providing a prefix to the Materialized View tables names that overrides the default one is an advanced option. The tables are prefix by default with the ID of the projects to avoid collision between projects. This configuration allows setting a custom prefix to the Materialized View tables created in the project.
6. Optional: Provide the **Source Settings** of the project.
   a) Add a remote Github repository URL to the **Clone URL** field.

      You can use the following example URLs with or without authentication:

      - To clone without authentication: https://github.com/cloudera/ssb-examples.git
      - To clone and push with HTTP basic authentication (username/password): https://github.com/cloudera/ssb-examples.git

        In case of using basic authentication, the personal access token needs to be provided as password. For more information about creating personal access tokens, see the Github documentation.
      - To clone and push with SSH authentication (SSH private key): git@github.com:cloudera/ssb-examples.git

        In case of using SSH authentication, you need to generate the private key using the following command:

        ```
        ssh-keygen -t ecdsa -b 256 -m pem -C "NAME@EXAMPLE.COM"
        ```

        As not every SSH private key is supported, you can use the following command to convert your key to the acceptable format:

        ```
        ssh-keygen -m pem -p -f [***PATH_TO_PRIVATE_KEY***]
        ```
      - To clone and push a local git repository on the filesystem of SSB host: file:///usr/src/ssb-examples/

   b) Provide the branch name or tag of the Github repository to the **Branch** field. This branch is checked out when pulling or pushing to the Github repository.

      Warning:

      When exporting a project, any sensitive information in CREATE TABLE statements, data sources, or SQL jobs will not be redacted. Ensure that usernames and passwords are hidden with the environment variables where required. For more information, see the Source control of a project documentation.
7. Enable or disable Allow deletions on import.

   Enabling the allow deletion upon import allows you to hard-reset your project to its versioned state when there are untracked resources in it. For example, the job is deleted from the project when importing the project from Github if the job does not exist in the repository. In other words, the upstream version overrides the version you have in Streaming SQL Console. If disabled, the project keeps the resources that were not pushed to the Github repository.

   The deletion affects the jobs, data sources, functions, Materialized Views and their API keys.
8. Enable or disable Authentication for the project.

What to do next

After you have created the project, you have the following options to manage the project on the **Projects** page:

1. Click Open to use the created project and start creating SQL jobs.
2. Click Switch to open another project than the active one.
3. Click Delete to remove the project from Cloudera SQL Stream Builder.

Navigating in a project

When you open a project in Streaming SQL Console, you can navigate between the Project Manager and the Tab view of the Project Resources.

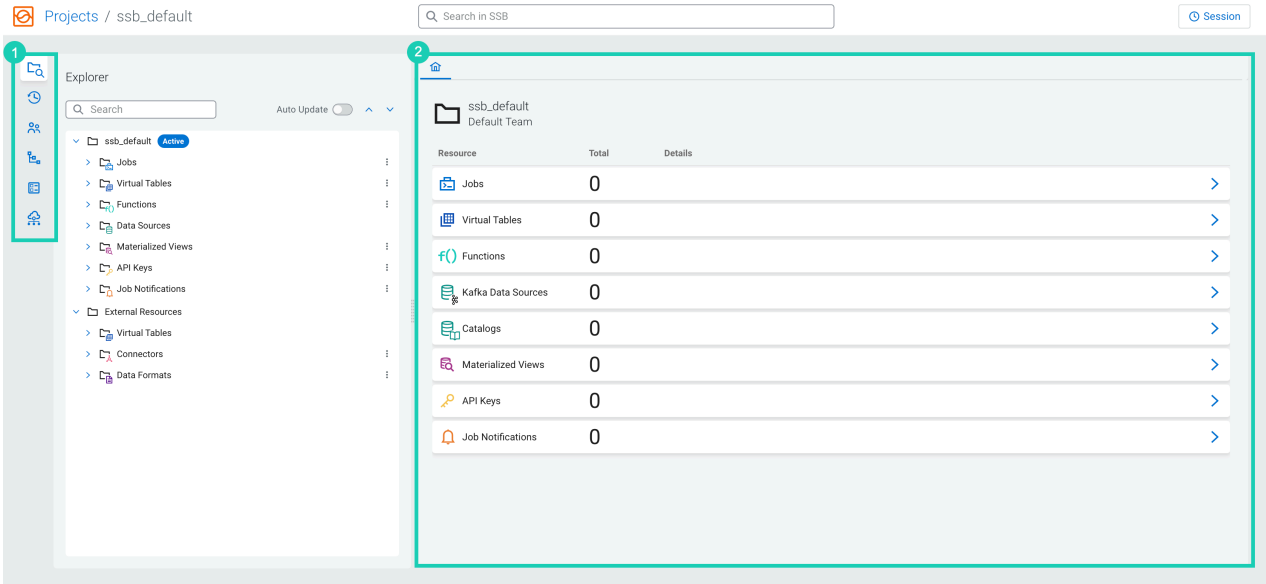

1. The **Project Manager** consists of the following views:
   - **Explorer** - tree view of the user defined and external resources of the project
   - **SQL History** - history of the queries that have been executed by the logged in user
   - **Members** - list of the users that are invited in the project
   - **Source Control** - configure the synchronization source (Git repository) for the project, import or export changes
   - **Environments** - manage environment configurations, define variables for templating
   - **Data Hub Service Discovery** - automatic discovery of Kafka, Kudu and Schema Registry services in your Cloudera Public Cloud environment
2. The **Home** tab of a project allows you to quickly access the user defined resources of a project. It also gives you a quick view about the total number of resources in each category with resource specific details. You can open the different resource tabs easily from the **Home** tab to create and manage items in a given resource category.

Managing member of a project

Created or imported projects can be shared with other users in Streaming SQL Console. You can invite members using their Streaming SQL Console username and set the access level to member or administrator.

About this task

When you create a project, you automatically become the owner of that project. You can invite Cloudera SQL Stream Builder users to your project and give them access to the jobs and resources. You can also set another user to be the admin of your project.

Table 1: Access levels in a project

Member	Creating and managing jobs and resources in a project, synchronizing the project, inviting other Members. Running jobs, stopping their own jobs.
Admin	In addition to Member privileges, Admins can stop jobs of other users, manage users of the project, invite other Admins or Members.
Owner	The priviliges of owners are identical to the Admins, but it cannot be revoked and Owners cannot be removed from the project.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click Members from the Project Manager.
4. Click Invite User.

   The **Invite Member** window appears.
5. Provide the Cloudera SQL Stream Builder Username of the member you want to invite.
6. Set the Access Level for the user.
7. Click Send Invitation.

   The sent invitation is visible on the **Projects** page of Streaming SQL Console under **My Invitations** or at the **Invitations** tab on the **Members** page.

   You can delete any members of a project that you are an owner or an admin of using button next to their username.

Source control of a project

When you provide a Git repository and branch for a project, you can push and pull your project to Git. This enables you to track your versions of your project, create a backup of your project in Git and manage the development lifecycle of your project.

You can provide the Git repository URL and branch for a project when you create one or when you import a project on the **Projects** page of Streaming SQL Console.

Note: The Source Control feature is in Technical Preview and not ready for production deployment. Cloudera encourages you to explore these features in non-production environments and provide feedback on your experiences through the *Cloudera Community Forums*.

After setting up your remote repository, you can navigate to Source Control in your project, and use the Import and Push buttons to synchronize your local state with the state in the repository.

Importing a version of project

When you import a project from Git to Cloudera SQL Stream Builder in Source Control, the configured Git repository is cloned to a temporary directory by Cloudera SQL Stream Builder, the specified branch is checked out, and the resource files found in the directory with the project's name are used to import resources to the Cloudera SQL Stream Builder project.

After pulling a version of a project, the results are listed based on the imported changes that include **UPDATED**, **DELETED**, **CREATED** or **ALREADY EXISTS**.

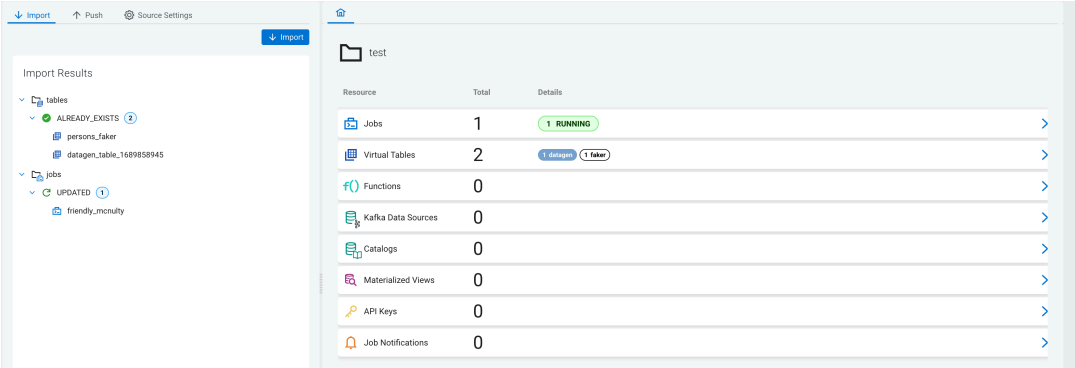

Warning: In case the imported changes affect a running job, the job must be restarted for the changes to take effect.

Pushing a version of project

When you push a project from to Git in Source Control, the configured repository is cloned to a temporary directory by SSB, and the project resources are written to files in a directory with the project's name into the temporary directory.

Changes are added and committed with the specified commit message, and pushed to the configured remote.

Changing the project settings

You can manage the source information of a project under the **Source Settings** tab. You can change the **Clone URL** and **Branch** information of the Git repository. You can enable **Allow deletions** for importing a project, and configure the **Authentication** method that is used when interacting with the repository.

Removing source settings

In case the source control settings need to be removed, you can delete the configurations for the repository using the Delete Source button.

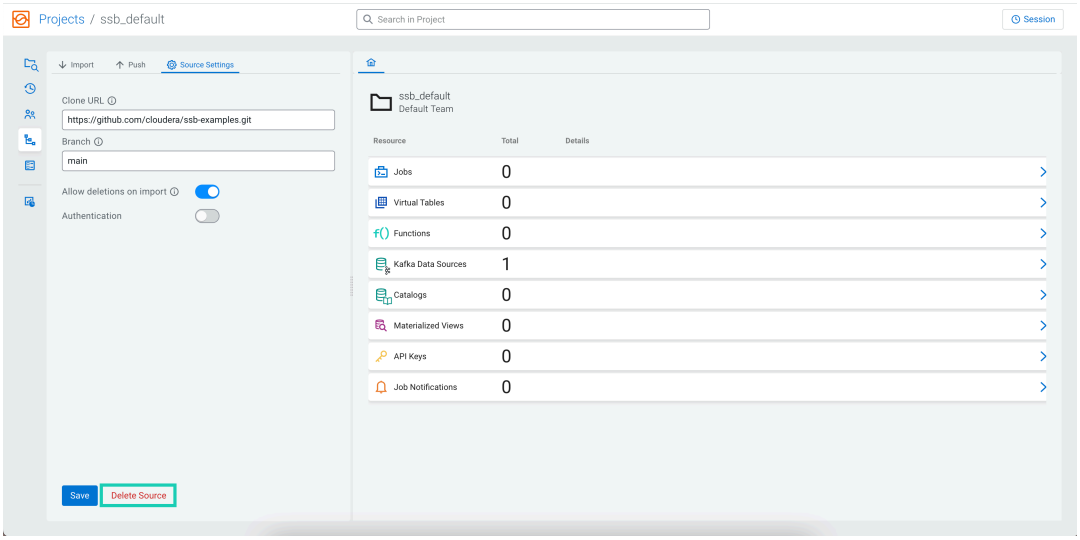

Masking information before using source control

When exporting projects from Cloudera SQL Stream Builder, possible sensitive parameters in CREATE TABLE statements, data sources, or the SQL job will not be redacted. To conceal sensitive information, environment variables can be used.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click Environments from the Project Manager.
4. Click New Environment.
5. Provide a Name for the environment.

   For example, you can name it Dev.
6. Add keys and values as needed.

   For example, you can add psql_password as key, and provide the password as value.
7. Click Activate to apply the specified information for the project.
8. When creating the table, reference the environment variable as shown in the following example:

```
CREATE TABLE  `postgres_cdc_table_1682585103` (
 `order_id` INT,
 `city` VARCHAR(2147483647),
 `street_address` VARCHAR(2147483647),
 `amount` INT,
 `order_time` TIMESTAMP(3),
 `order_status` VARCHAR(2147483647)
) WITH (
 'connector' = 'postgres-cdc',
 'PASSWORD' = '${SSB.ENV.PSQL_PASSWORD}'
 'username' = '...'
);
```

Setting the environment for a project

Environments are sets of user-defined variables that can be substituted into templates in your Cloudera SQL Stream Builder resources. Environments are like configuration files for your projects that adapt it to be used in a specific environment. They enable you to keep the common resources and logic of a project in a central repository and reuse it across different environments, different clusters.

About this task

Creating an environment file for a project means that users can create a template with variables that could be used to store environment-specific configuration.

For example, you might have a development, staging and production environment, each containing different clusters, databases, service URLs and authentication methods. Projects and environments allow you to write the logic and

create the resources once, and use template placeholders for values that need to be replaced with the environment specific parameters.

To each project, you can create multiple environments, but only one can be active at a time for a project. Environments can be exported to files, and can be imported again to be used for another project, or on another cluster.

While environments are applied to a given project, they are not part of the project. They are not synchronized to Git when exporting or importing the project. This separation is what allows the storing of environment-specific values, or configurations that you do not want to expose to the Git repository.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click Environments from the Project Manager.

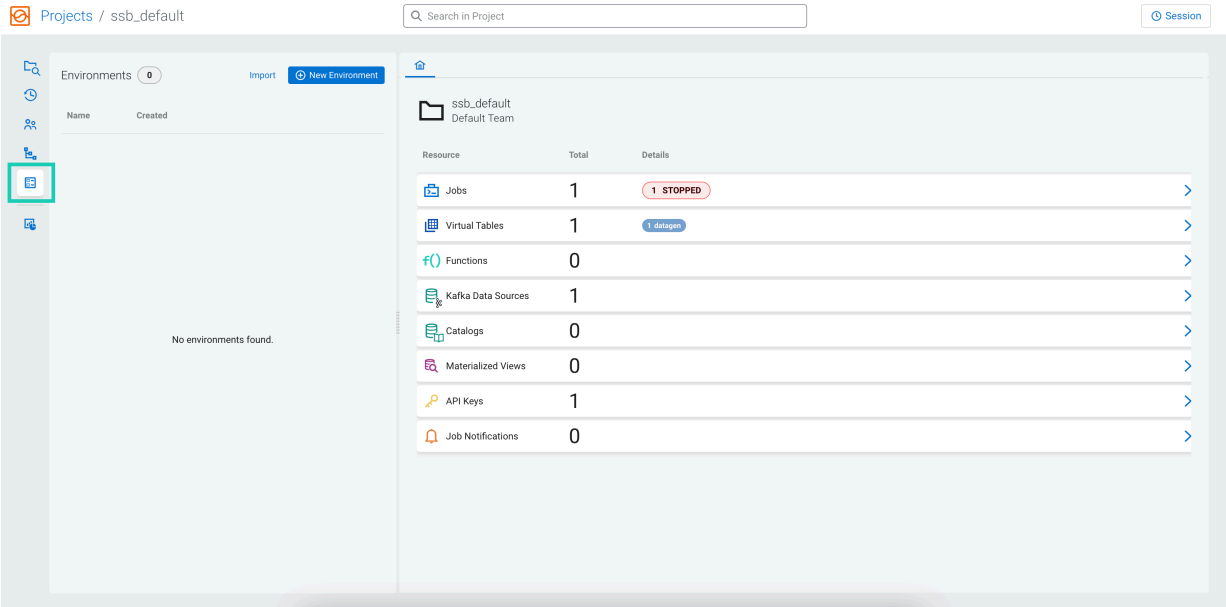

4. Click New Environment.
5. Provide a Name for the environment.
6. Add the Key and Value pairs as needed.

7. Enable or disable masking for the values using the Secure toggle.

   The Secure toggle allows you to hide sensitive information provided as values on the Streaming SQL Console as shown in the following example:

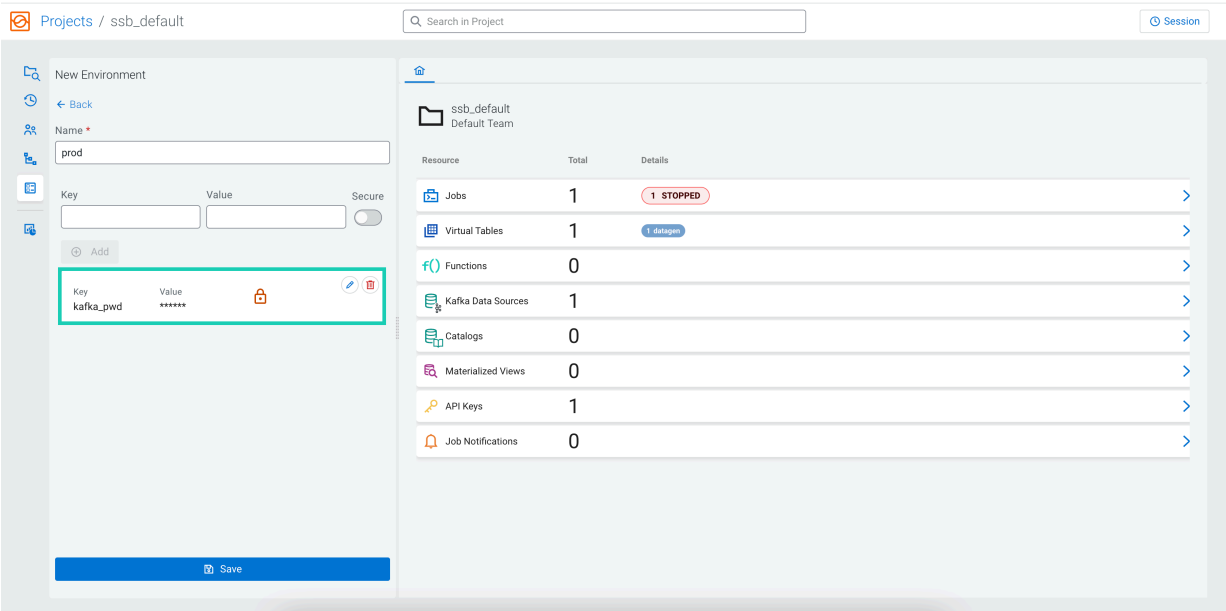

8. Click Add to save the key and value pair.
9. Click Create when you added the necessary environment variables.

   After creating an environment file, you have the following options to manage it:

   a. Click Activate to apply the specified information for the project. To deactivate an active environment file, you need to click the Deactivate button.

   b. Click ⋮ Export to export the environment file.

   c. Click Import and search for an exported environment file on your computer to import an environment file.

   d. Click ⋮ Delete to remove the environment file from the project.

What to do next

You can use a variable defined in your active environment in the following ways:

- As values in the properties section of a CREATE TABLE statement as shown in the following example:

```
CREATE TABLE (...) WITH ('key1'='${ssb.env.yourVariable}')
```

- In the properties of a Kafka data source, for example bootstrap servers, truststore location
- In the values of Catalog properties, for example Schema Registry URL, Kudu masters, Custom Catalog

To refer to an environment variable called *YOURVARIABLE*, you can use the following syntax: ${ssb.env.yourVariable}. You can use multiple substitutions or combine literals and variable substitutions in the same value string as shown in the following example:

```
'some-connector-url' = 'https://${ssb.env.connector-host}:${ssb.env.connector-port}'
```

Besides environments, it is also possible to define additional variables using the SET_VAR 'x' = 'y'; syntax in your queries. These do not become part of the environment and are not persisted, but live only in the current SQL

session. They can be used for substitutions like environment variables, but without the ssb.env prefix as shown in the following example:

```
SET_VAR 'yourVariable' = 'someValue';
CREATE TABLE (...) WITH ('key1'='${yourVariable}'))
```

You can reset a variable with the RESET_VAR 'x'; command.

Importing a project

You can use a Git repository to import a SQL Stream Builder project that was previously created and exported using source control.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.
2. Click Import Project on the **Projects** page of Streaming SQL Console.

   The **Import Project** window appears.
3. Optional: Provide a prefix that overrides the Materialized View table names.

   Providing a prefix to the Materialized View tables names that overrides the default one is an advanced option. The tables are prefix by default with the ID of the projects to avoid collision between projects. This configuration allows setting a custom prefix to the Materialized View tables created in the project.
4. Provide the **Source Settings** of the project.
   a) Add a remote Github repository URL to the **Clone URL** field.

      You can use the following example URLs with or without authentication:

      - To clone without authentication: https://github.com/cloudera/ssb-examples.git
      - To clone and push with HTTP basic authentication (username/password): https://github.com/cloudera/ssb-examples.git

        In case of using basic authentication, the personal access token needs to be provided as password. For more information about creating personal access tokens, see the Github documentation.
      - To clone and push with SSH authentication (SSH private key): git@github.com:cloudera/ssb-examples.git

        In case of using SSH authentication, you need to generate the private key using the following command:

        ```
        ssh-keygen -t ecdsa -b 256 -m pem -C "NAME@EXAMPLE.COM"
        ```

        As not every SSH private key is supported, you can use the following command to convert your key to the acceptable format:

        ```
        ssh-keygen -m pem -p -f [***PATH_TO_PRIVATE_KEY***]
        ```
      - To clone and push a local git repository on the filesystem of SSB host: file:///usr/src/ssb-examples/

   b) Provide the branch name or tag of the Github repository to the **Branch** field. This branch is checked out when pulling or pushing to the Github repository.
5. Provide the name of the project to the **Project** field.

6. Enable or disable Allow deletions on import.

   Enabling the allow deletion upon import allows you to hard-reset your project to its versioned state when there are untracked resources in it. For example, the job is deleted from the project when importing the project from Github if the job does not exist in the repository. In other words, the upstream version overrides the version you have in Streaming SQL Console. If disabled, the project keeps the resources that were not pushed to the Github repository.

   The deletion affects the jobs, data sources, functions, Materialized Views and their API keys.
7. Enable or disable Authentication for the project.
8. Click Import.

What to do next

After you have created the project, you have the following options to manage the project on the **Projects** page:

1. Click Open to use the created project and start creating SQL jobs.
2. Click Switch to open another project than the active one.
3. Click Delete to remove the project from SSB.

Registering Data Sources in Cloudera SQL Stream Builder

Data Sources are a set of data endpoints to be used as sources, sinks and catalogs. Data Sources allow you to connect to an already installed component on your cluster, then use that provider for adding tables in Cloudera SQL Stream Builder.

You can register Kafka as a data source, or Kudu, Hive, Cloudera Schema Registry and Confluent Schema Registry as a catalog and you can also add custom catalogs. When registering the components, Cloudera SQL Stream Builder can access the already existing topics from Kafka, tables from Kudu and Hive, and the schema in Cloudera Schema Registry and Confluent Schema Registry. This also means that when you update a data source, for example add new topics, tables and schemas, Cloudera SQL Stream Builder automatically detects the changes.

Adding Kafka Data Source

To create Kafka tables in Cloudera SQL Stream Builder you need to register Kafka as a Data Source, using the Streaming SQL Console.

About this task

A default Local Kafka is added to the Cloudera SQL Stream Builder data sources during installation, using a Kafka service within the same cluster as Cloudera SQL Stream Builder. This Local Kafka data source cannot be updated or delete, as it is used in the Streaming SQL Console for sampling results and cleaning up sample topics. To add your own, customizable Kafka data source instead, follow the steps in this task.

Before you begin

- Make sure that you have the right permissions set in Ranger.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.

2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Open Data Sources from the **Explorer** view.
4. Click ⋮ next to **Kafka**.
5. Select New Kafka Source.

   The Kafka Source window appears.

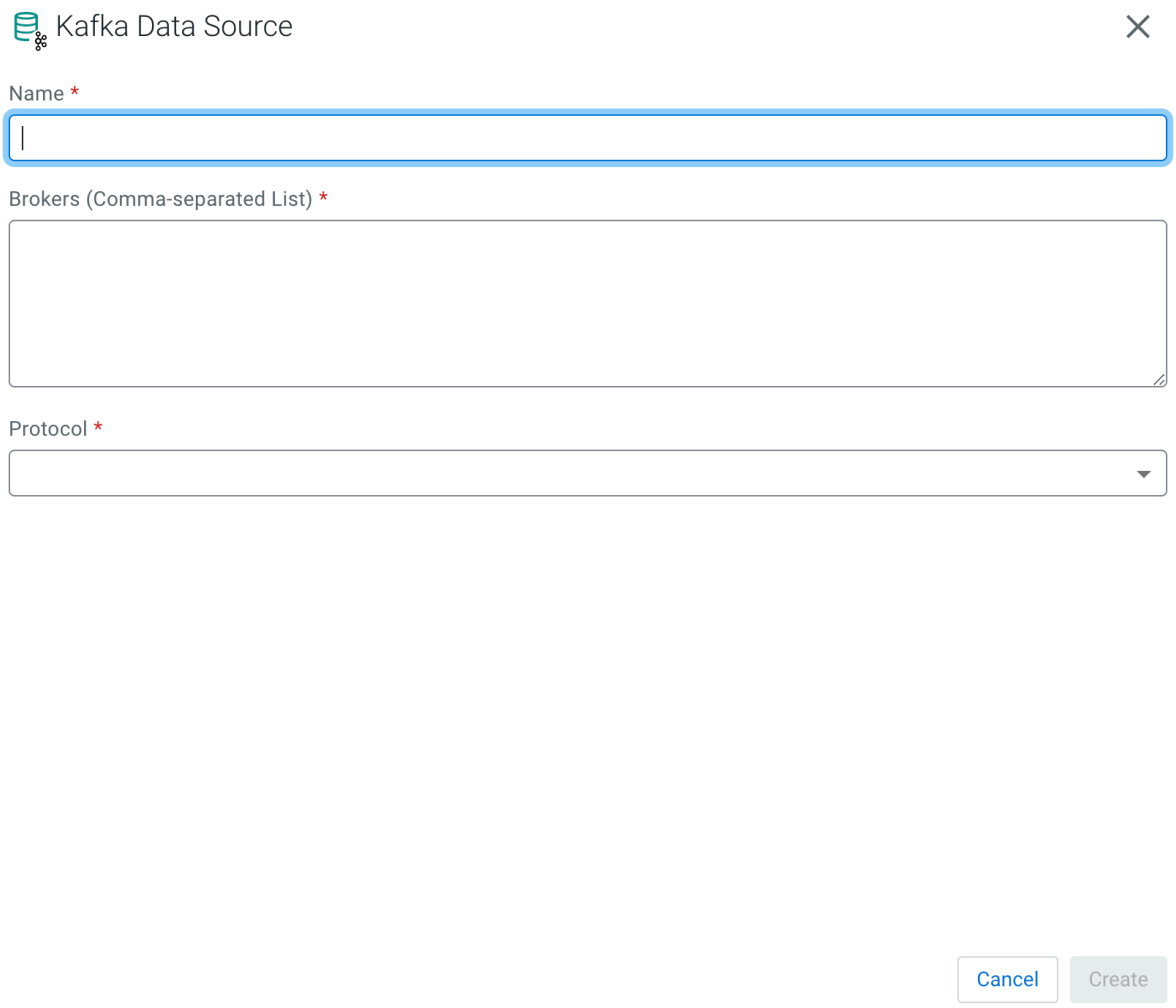

6. Add a Name to your Kafka provider.

7. Add the broker host name(s) to Brokers.

   You need to copy the Kafka broker name(s) from Cloudera Manager.

   a) Go to the Cloudera Streams Messaging cluster in your environment.
   b) Select Cloudera Manager from the list of services.
   c) Click Kafka from the list of services.
   d) Click Instances.
   e) Copy the hostname of the Kafka broker(s) you want to use.
   f) Go back to the Add Kafka Source page.
   g) Paste the broker hostname to the Brokers field.

      Note: You can add more than one broker hostname by separating them by commas.

   h) Add the default Kafka port after the hostname(s).

      Example:

```
docs-test-1.vpc.cloudera.com:9092,
docs-test-2.vpc.cloudera.com:9092
```

8. Select the security Protocol.

   The connection protocol must be the same as it is configured for the Kafka cluster in Cloudera Manager.

   You can choose from the following protocols:

   For PLAINTEXT

   a. Click Validate.
   b. Click Create after validation is successful.

   For SSL

   a. Choose your authentication method:

   By default the auto-discovery of Cloudera TrustStore is enabled. The auto-discovery of Cloudera TrustStore can be used for Kafka sources that are located in the same Cloudera Private Cloud Base cluster as Cloudera SQL Stream Builder. This means that the default TrustStore path is used for authentication, which can be customized in Cloudera Manager using local.kafka.truststore.location and local.kafka.truststore.password parameters.

   Note: If you change the default value of local.kafka.truststore.location and local.kafka.truststore.password parameters, the same value will be applied to the Local Kafka data source as well.

   If you disable the auto-discovery, you need to provide the following configurations:

   - Kafka TrustStore path
   - Kafka TrustStore Password

   You also have the option to provide the Kafka KeyStore path and KeyStore Password that belongs to the Kafka source.

   b. Click Validate.
   c. Click Create after validation is successful.

   For SASL SSL

   a. Choose your authentication method:

   By default the auto-discovery of Cloudera TrustStore is enabled. The auto-discovery of Cloudera TrustStore can be used for Kafka sources that are located in the same Cloudera Private Cloud Base cluster as Cloudera SQL Stream Builder. This means that the default TrustStore path is used for authentication,

which can be customized in Cloudera Manager using local.kafka.truststore.location and local.kafka.truststore.password parameters.

Note: If you change the default value of local.kafka.truststore.location and local.kafka.truststore.password parameters, the same value will be applied to the Local Kafka data source as well.

If you disable the auto-discovery, you need to provide the following configurations:

- Kafka TrustStore path
- Kafka TrustStore Password

You also have the option to provide the Kafka KeyStore path and KeyStore Password that belongs to the Kafka source.

b. Choose an SASL Mechanism.
c. Click Validate.
d. Click Create after validation is successful.

For SASL PLAINTEXT

a. Choose an SASL Mechanism.
b. Provide the Username for SASL.
c. Provide the Password for SASL.
d. Click Validate.
e. Click Create after validation is successful.

Results

You have registered Kafka as a data source to be able to add Kafka as a table in your SQL query. The already existing Kafka topics can be selected when adding Kafka as a table.

What to do next

After registering the Kafka data source, you can edit, duplicate, and delete it from the Streaming SQL Console:

1. Open Data Sources from the **Explorer** view.
2. Click next to **Kafka**.
3. Select Manage.

   The **Kafka Sources** tab opens where the registered Kafka providers are listed. You have the following options to manage the Kafka sources:

   - Click on one of the existing Kafka providers to edit its configurations.
   - Click to remove the Kafka provider.
   - Click to duplicate the Kafka provider with its configurations.

Related Information

Configuring Ranger policies

Adding Catalogs

You need to add Cloudera Schema Registry, Kudu, Hive, or other services as a Catalog using the Streaming SQL Console in Cloudera SQL Stream Builder to use them with Flink DDL.

Before you begin

- Make sure that you have the required service on your cluster.
- Make sure that you have the right permissions set in Ranger for Cloudera SQL Stream Builder and the services.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Open Data Sources from the **Explorer** view.
4. Click ⋮ next to **Catalogs**.

5. Select New Catalog.
   The Add Catalog window appears.

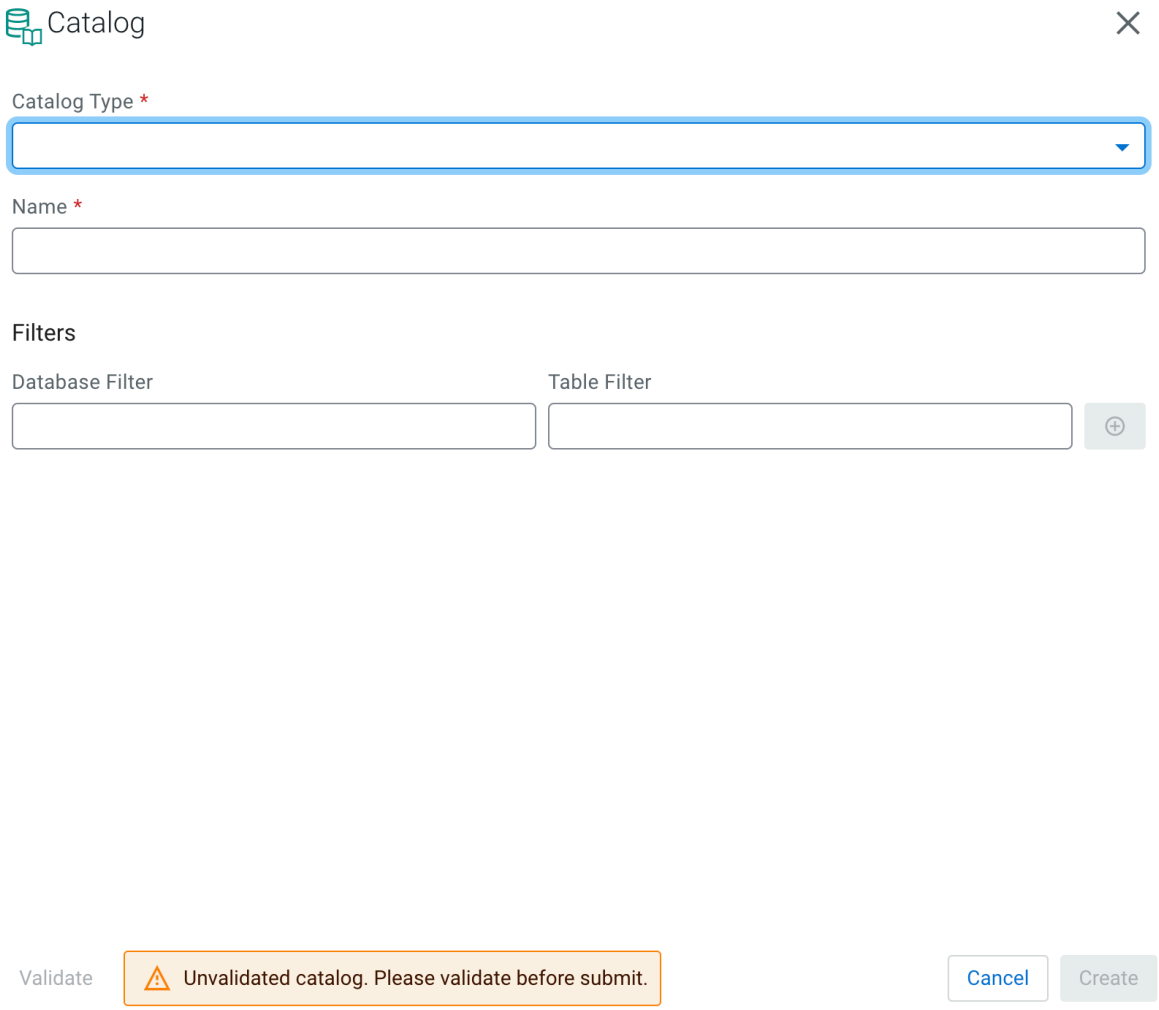

6. Select the Catalog Type from the following options:

For Schema Registry

a. Add a Name to your catalog.
b. Select Schema Registry from the **Catalog Type** drop-down.

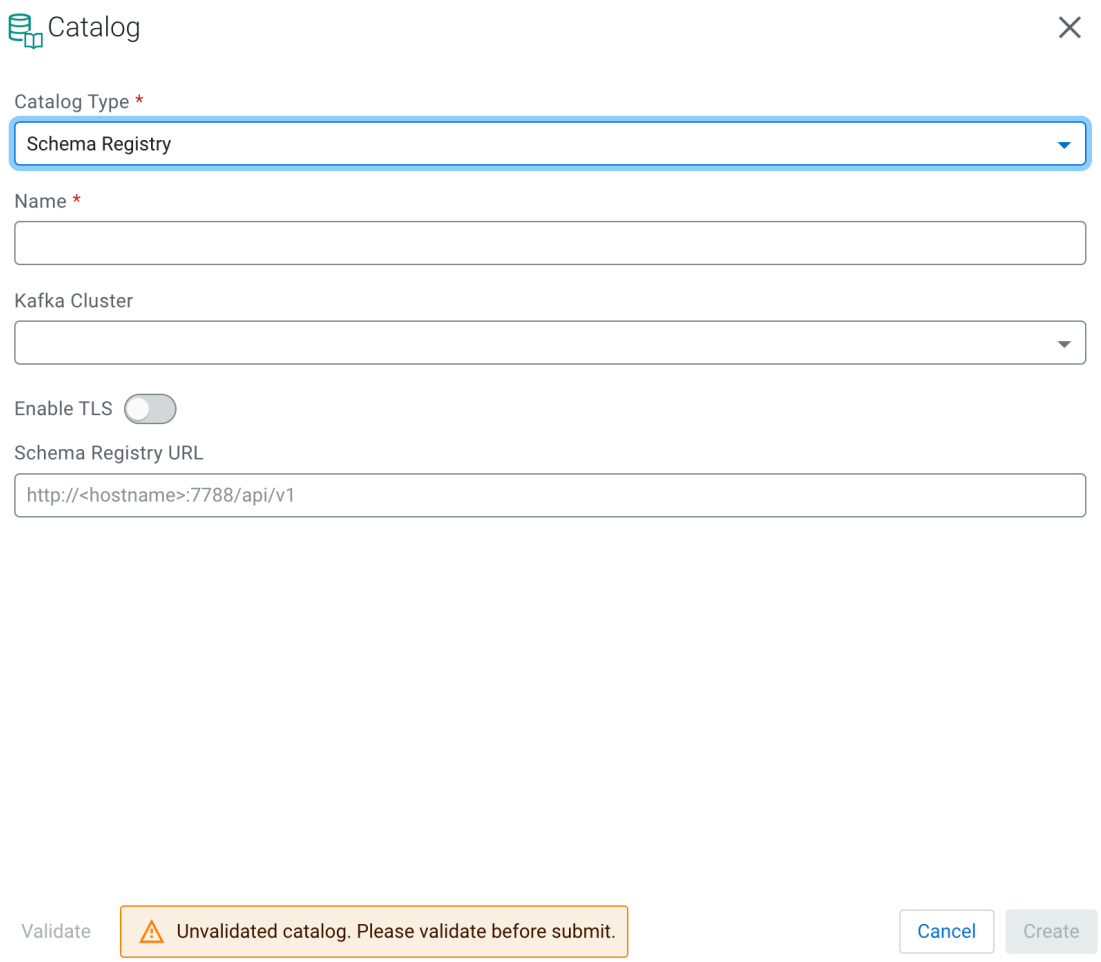

c. Select the Kafka cluster you registered as Data Source.
d. Enable TLS, if needed for the communication.
   1. If you enabled TLS, provide the Schema Registry Truststore location and password to the SR TrustStore and SR TrustStore Password field.
e. Add the Schema Registry URL.
   1. Go to the Streams Messaging cluster in your environment.
   2. Select Cloudera Manager from the list of services.
   3. Select Schema Registry from the list of services.
   4. Click on Instances.
   5. Copy the Hostname of Schema Registry.
   6. Add the default port of Schema Registry after the hostname.

      Example:

      ```
      http://docs-test-1.vpc.cloudera.com:7788/api/v1
      ```

f. Optional: Specify a Schema key suffix and a Schema value suffix.

By default, the Schema Registry catalog looks for a topic in Kafka with the same name as the schemas stored in Cloudera Schema Registry. It uses the schema to deserialize the payload of the messages in that topic and ignores the message key. If you have different schemas to deserialize message key and payload you can define suffixes to differentiate them.For example, if you configure the catalog's schema key suffix as -key and the schema value suffix as -value, and you have schemas stored in Cloudera Schema Registry named example-key and example-value. The Cloudera SQL Stream Builder catalog will automatically create a table called example and use those schemas to deserialize the key and the payload of the messages, respectively.

g. Optional: Add a Prefix for key fields in schema.

You can set a prefix for key field names to avoid table creation failure as the column names in the Flink schema are created based on the field names of the Avro schema. Table creation fails if the key and value

schemas contain overlapping fields. The provided prefix will be assigned to the key fields of the Flink table as shown in the following example:

```
CREATE TABLE … (k_a INT, k_b INT, b INT, c INT0 with (..., 'key.fiel
ds'='k_a;k_b')
```

For Kudu

a. Select Kudu from the Catalog Type drop-down.

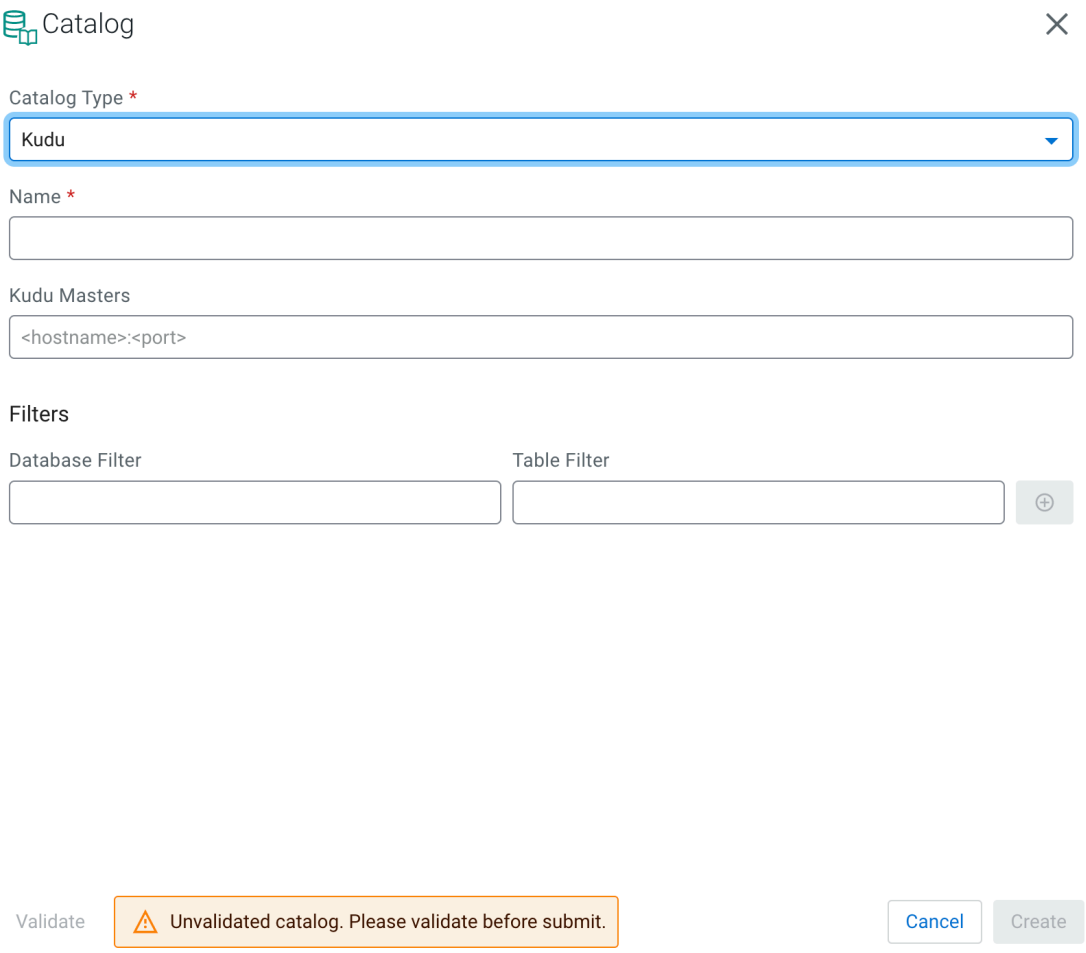

b. Add the host URL of Kudu Masters.

1. Go to the Real-Time Data Mart cluster in your environment.
2. Select Cloudera Manager from the list of services.
3. Select Kudu from the list of services.
4. Click on Instances.
5. Copy the Hostname of the Master Default Group.
6. Add the default port of Kudu after the hostname.

   Example:

   ```
   docs-test-1.vpc.cloudera.com:7051
   ```

For Hive

a. Select Hive from the Catalog Type drop-down.

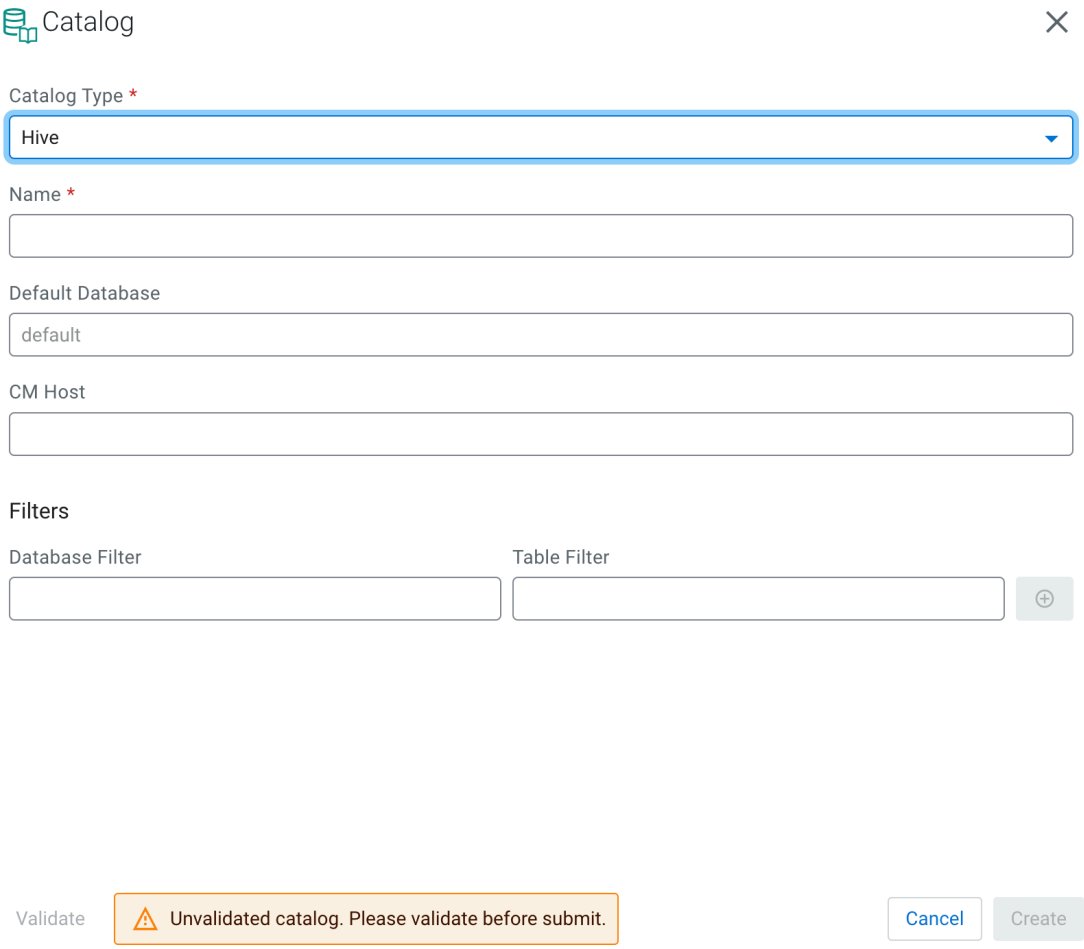

b. Provide a Name to the Hive catalog created in SSB.

c. Provide the name of the Default Database in Hive.

d. Provide the hostname of Cloudera Manager at CM Host.

You need to provide the Cloudera Manager hostname where the Hive service is also located. You can check the hostname of Cloudera Manager by clicking CM-UI from the **Services** of the Cloudera Public Cloud environment page, then access Hosts All Hosts from the Cloudera Manager homepage:

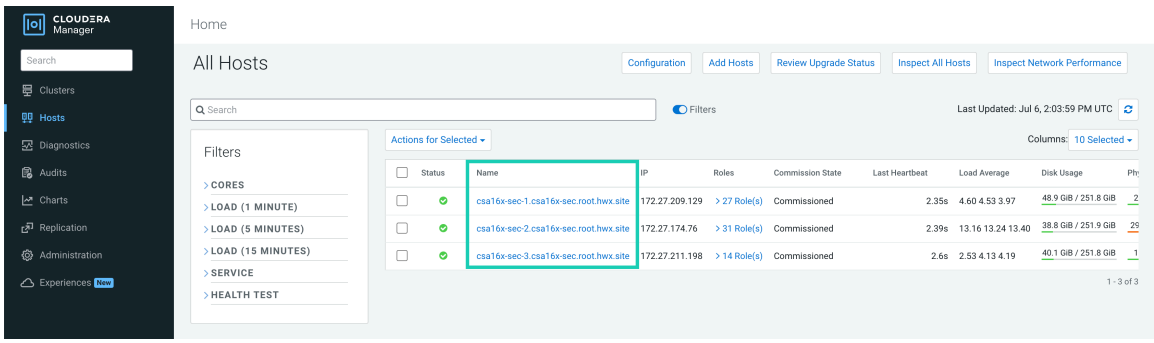

For Custom

a. Select Custom from the Catalog Type drop-down.

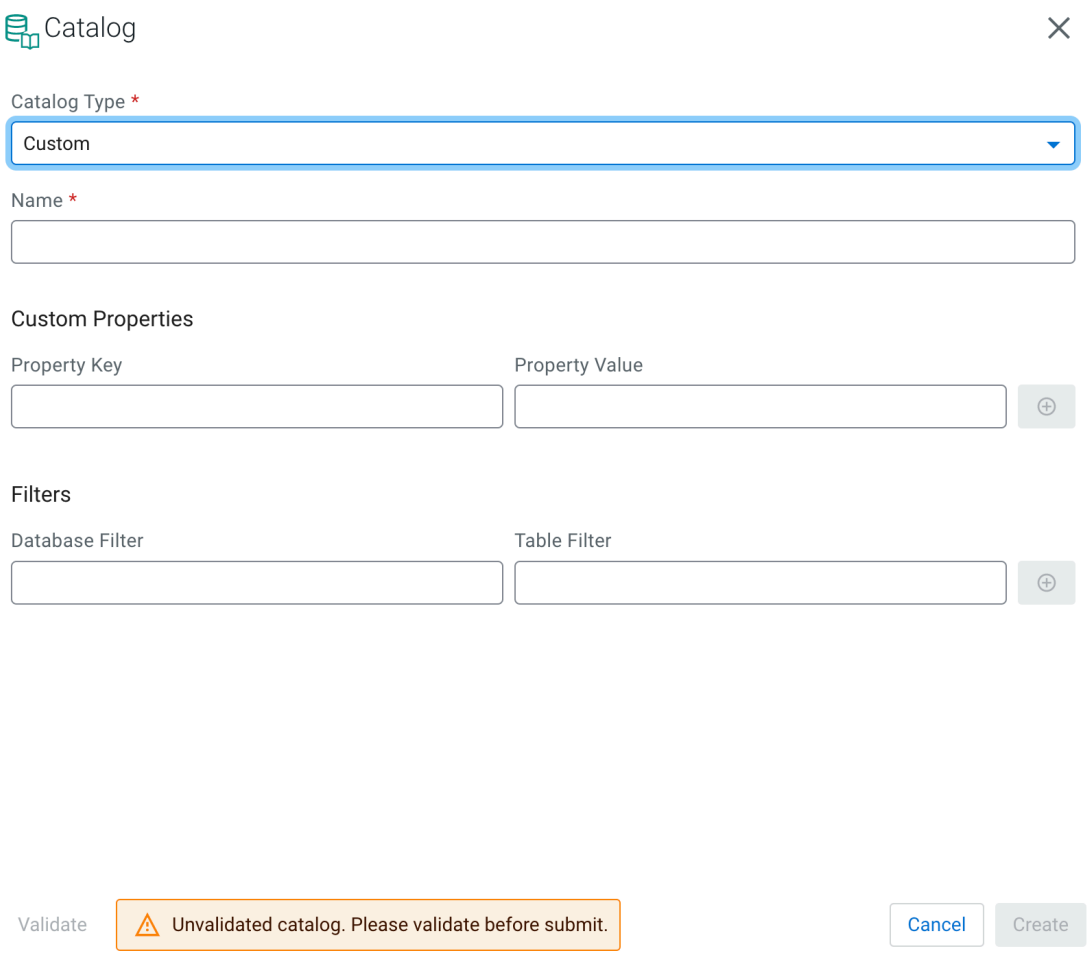

b. Provide a Property Key.

c. Proivde a Property Value.

If needed, you can specify more custom properties by using the plus icon.

7. Click on Add Filter.

a) Provide a Database and Table filter if you want to select specific tables to use from the catalog.

8. Click on Validate.

9. If the validation is successful, click Create.

Results

You are ready to use the added catalog in Cloudera SQL Stream Builder with Flink DDL. The already existing schemas in Schema Registry and Confluent Schema Registry, tables in Kudu and Hive are automatically imported to SSB.

What to do next

After registering a catalog, you can edit, duplicate and delete it from Streaming SQL Console:

1. Open Data Sources from the **Explorer** view.

2. Click ⋮ next to **Catalogs**.

3. Select Manage.

   The **Catalogs** tab opens where the registered catalogs are listed. You have the following options to manage the catalog sources:

   - Click on one of the existing catalogs to edit its configurations.
   - Click to remove the catalog.
   - Click to duplicate the catalog with its configurations.

Related Information

Configuring Ranger policies

Using auto discovery of services

Using the auto discovery of services in Cloudera SQL Stream Builder, you can easily connect to catalogs and data sources that are running on a different Cloudera Data Hub cluster in the same Cloudera Public Cloud environment.

Note: The Service Discovery feature is in Technical Preview and not ready for production deployment. Cloudera encourages you to explore these features in non-production environments and provide feedback on your experiences through the *Cloudera Community Forums*.

You can use the service discovery to automatically register Kafka, Schema Registry and Kudu with Cloudera SQL Stream Builder within the same environment. This means when you have a Cloudera Streams Messaging or Real-Time Data Mart cluster beside your Cloudera Streaming Analytics cluster in your environment, you can use the auto discovery to automatically import the tables from both services. Using the auto discovery has the same result as manually adding the catalogs and providers: you can read from and write to Kafka, Schema Registry and Kudu. However, when you have numerous Kafka, Schema Registry and Kudu services in your environment, you do not have to add them one by one, instead you can use the service discovery to import them all at once.

For more information about manually adding data sources and catalogs, see the Registering Data providers in Cloudera SQL Stream Builder section.

Note: In the Cloudera Streaming Analytics 7.2.16 cluster definition, Apache Hive is not supported for the auto discovery feature of Cloudera SQL Stream Builder.

The auto discovery feature does not check the changes you make in your environment. This means that you can register the newly added services from your environment to Cloudera SQL Stream Builder using the same process. However, when you delete the cluster from the environment, the already registered catalog and data sources are not removed from Cloudera SQL Stream Builder, you need to manually delete them from the Streaming SQL Console.

Setting up the service discovery

As an administrator, you need to make sure to set up a machine user that is used to register the services from the clusters to Cloudera SQL Stream Builder. The machine user information must be provided to the user.

About this task

For the auto discovery, you need to create a machine user, and assign the correct role so the machine user has access to the environment. The machine user also must be synchronized to the environment.

Important: Make sure that the network policies are correctly set up between the Cloudera Public Cloud environment and the cloud provider. For more information, see the cloud provider specific information under *Related information*.

Procedure

1. Navigate to Cloudera Management Console > User Management page.
2. Click Actions > Create Machine User .

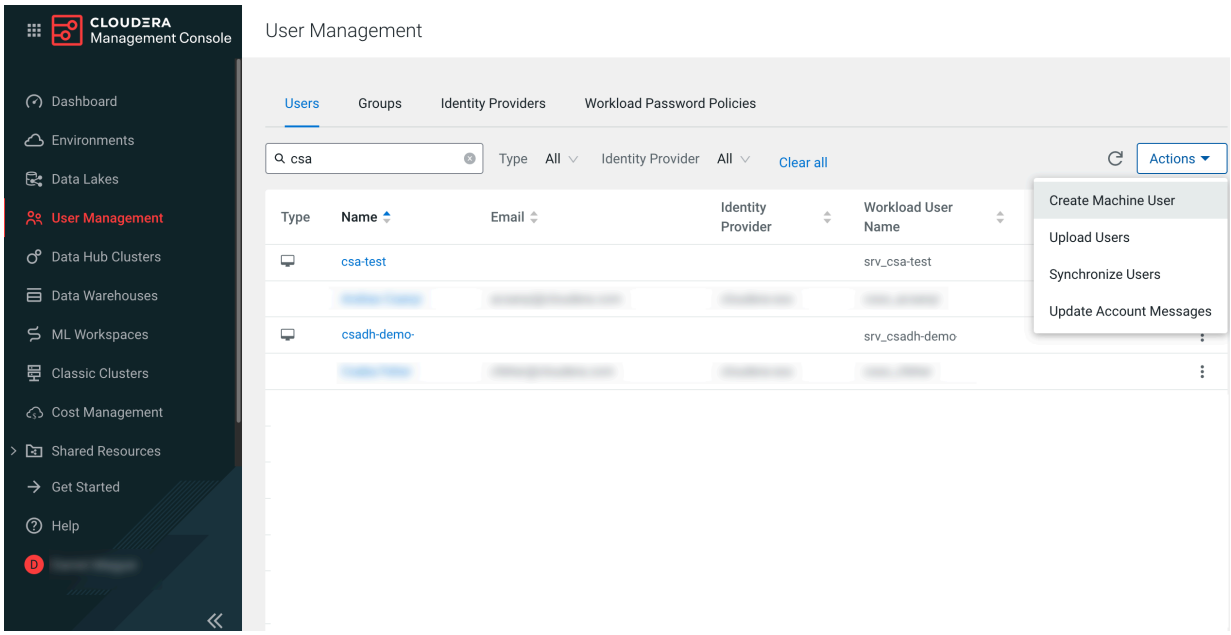

3. Provide a name to the Machine User.
4. Click Create.

   You are redirected to the **User Profile** page.

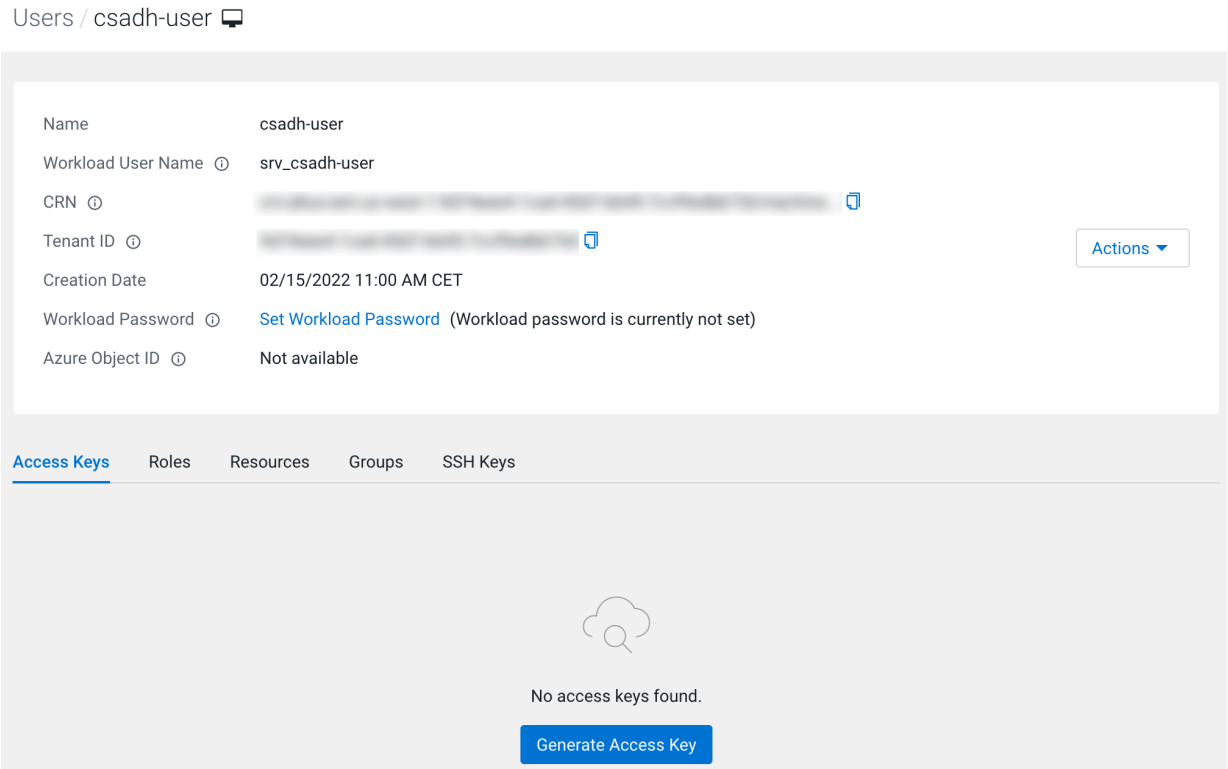

5. Click Set Workload Password.
6. Provide a Workload Password for the Machine User.
7. Click Set Workload Password.

8. Select Generate Access Key under **Access Keys** tab.

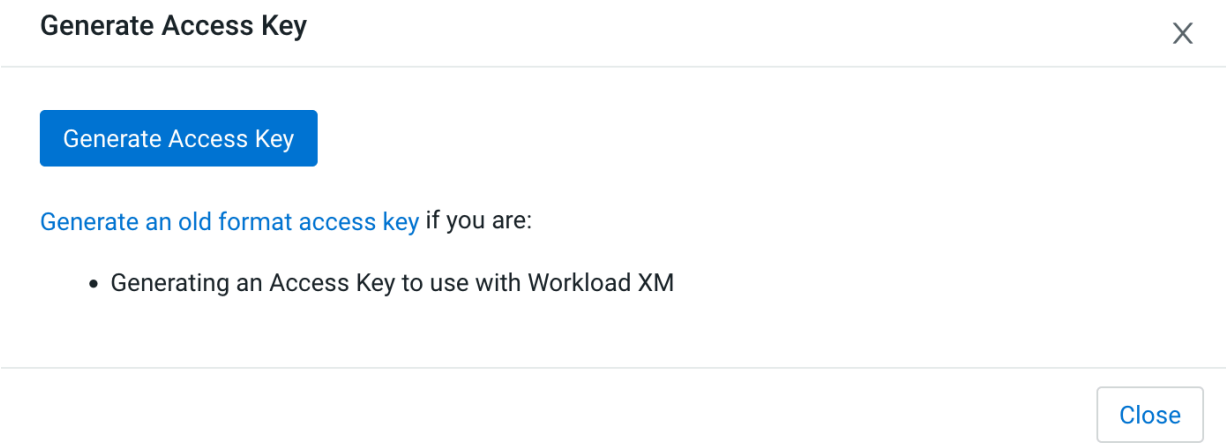

9. Click Generate Access Key.
   The **Generate Access Key** window pops up to indicate that the access key generation was successful.

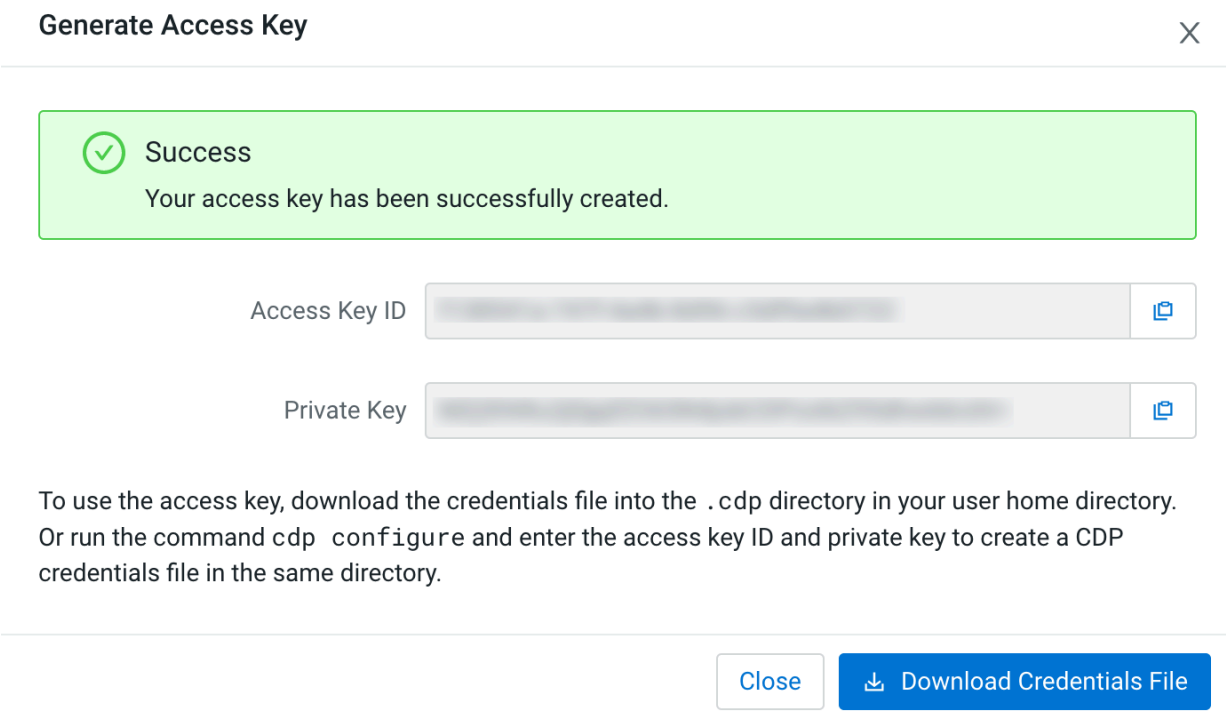

10. Take note of the Access Key ID and Private Key.
11. Click Download Credentials File.

    Note: Make sure to download the credentials file as you will not be able to retrieve the private key after closing the window.

12. Click Close.
13. Navigate to Cloudera Management Console > Environments and select your environment.
14. Click Actions > Manage Access .
15. Search for the Machine User you have created.

16. Select *ENVIRONMENTADMIN* role from the list of Resource Roles.

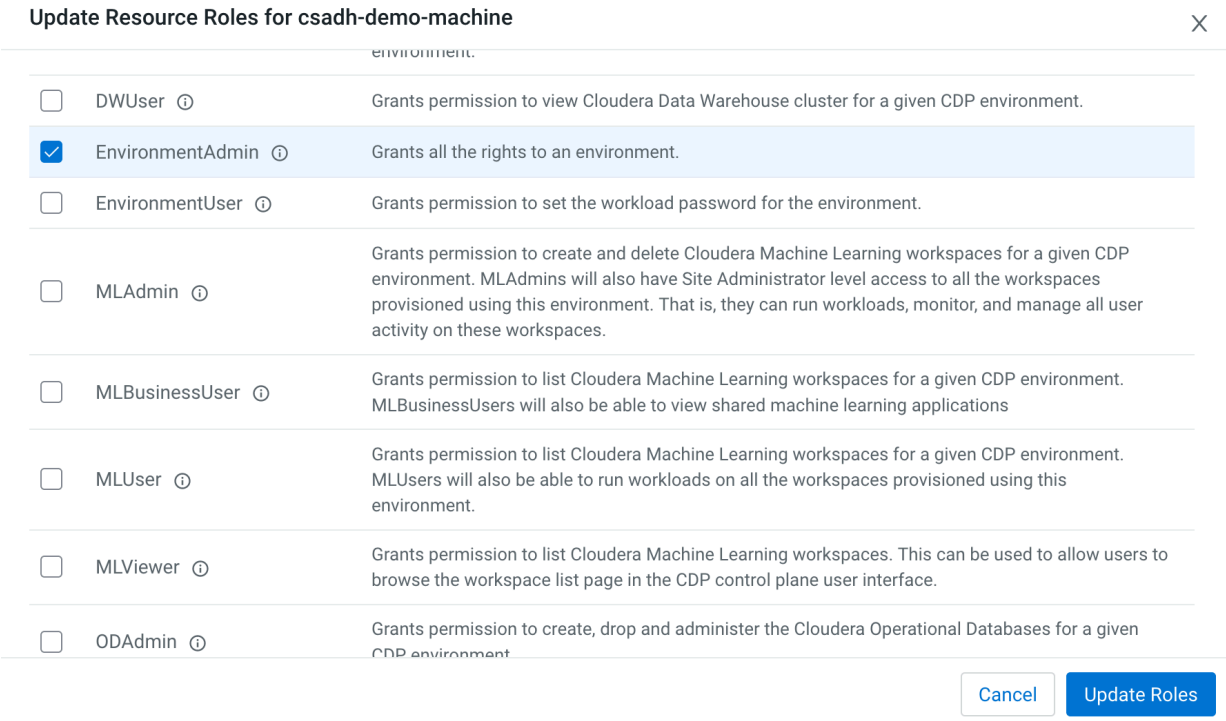

17. Click Update Roles.

The Resource Role for the selected user or group will be updated.

18. Navigate to Cloudera Management Console > Environments , and select the environment where you want to create a cluster.

19. Click Actions > Synchronize Users to FreeIPA .

20. Click Synchronize Users.

What to do next

After setting up the auto discovery feature, you need to access the Streaming SQL Console and import the data sources or catalogs from the clusters in your environment.

Using the service discovery on Streaming SQL Console

Using the Streaming SQL Console, you can import the Kafka, Schema Registry and Kudu services that already exist in your Cloudera Public Cloud environment. For the auto discovery, you need to provide the environment and machine user information that was provided by your administrator.

Procedure

1. Navigate to the Streaming SQL Console.

a) Navigate to Management Console Environments , and select the environment where you have created your cluster.

b) Select the Streaming Analytics cluster from the list of Data Hub clusters.

c) Select Streaming SQL Console from the list of services.

The **Streaming SQL Console** opens in a new window.

2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click Data Hub Service Discovery from the **Project Manager**.

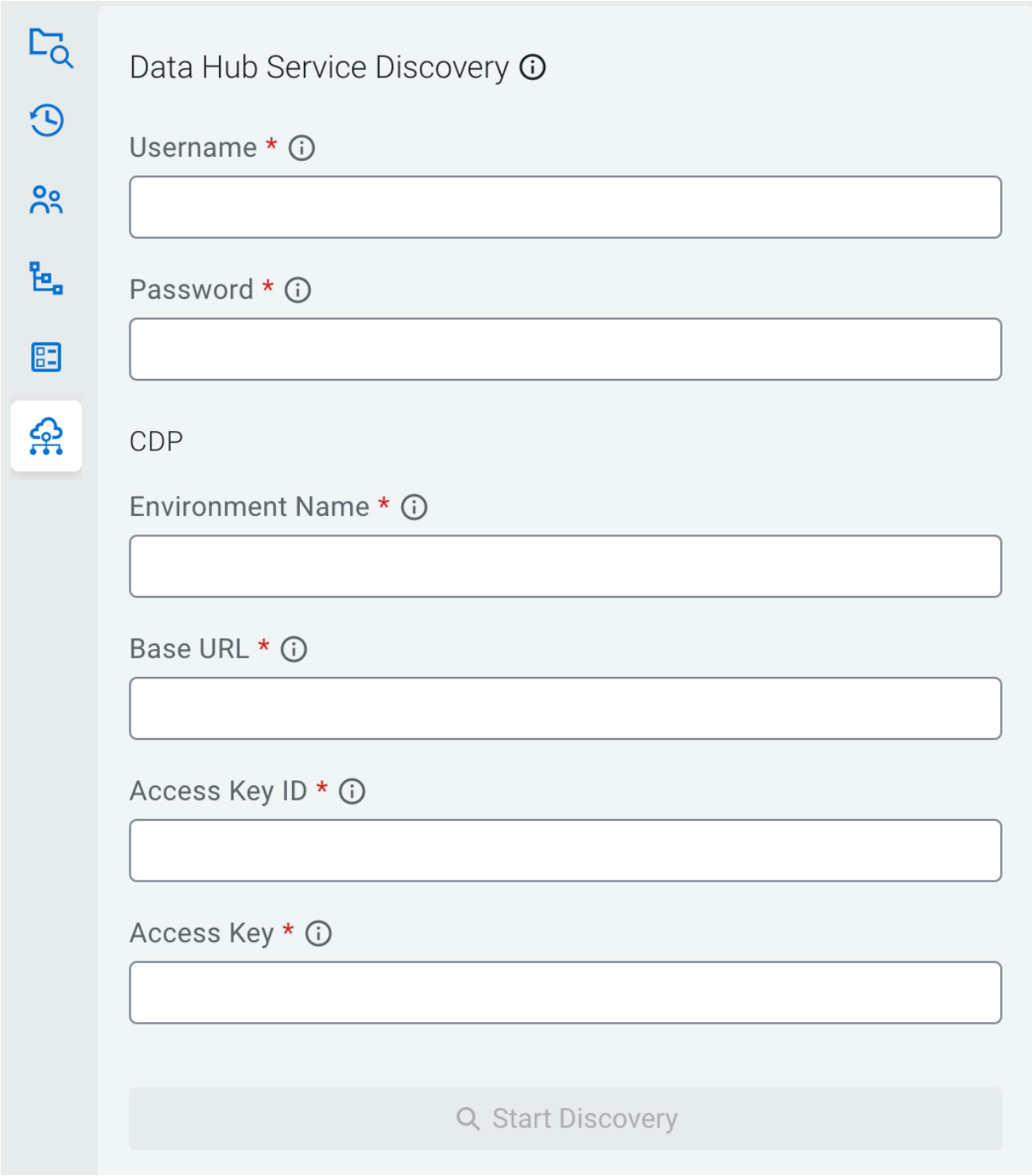

4. Provide the needed information to the corresponding field.

   Important: The environment and machine user information that is needed for the service discovery is provided by the administrator.

Field	Description	Example
Username	Workload username of the Machine User.	*CSA-TEST*
Password	Workload password of the Machine User.	*PASSWORD123!*
Environment Name	The name of the Cloudera Public Cloud environment you are using.	*CSA-TEST-ENV*
Base URL	The URL snippet from the Cloudera Management Console access URL.	*HTTPS://CONSOLE.US-WEST-1.CDP.CLOUDERA.COM*
Access Key ID	Access Key ID of the Machine User.	*00A0A0AA-00AA-00AA-00A0-0000A0A00A0*
Access Key	Private Key of the Machine User.	*Z66PXEXAMPLEKEY +Y3OIU070GEHS3KQI1EXAMPLEKEY=*

5. Click Start Discovery.

   The existing Kafka, Kudu and Schema Registry services in your environment should appear under Kafka Sources and Catalogs.

Connectors

Cloudera SQL Stream Builder supports different connector types and data formats for Flink SQL tables to ease development and access to all kinds of data sources.

The following table summarizes the supported connectors and how they can be used in Cloudera SQL Stream Builder:

Connector	Type	Description
Kafka	source/sink	Supported as exactly-once-sink
Hive	source/sink	Can be used as catalog
Kudu	source/sink	Can be used as catalog
Schema Registry	source/sink	Can be used as catalog
JDBC	source/sink	Can be used with Flink SQL. PostgreSQL, MySQL and Hive are supported.
Filesystems	source/sink	Filesystems such as HDFS, S3 and so on. Can be used with Flink SQL
Debezium CDC	source	Can be used with Flink SQL. PostgreSQL, MySQL, Oracle DB, Db2 and SQL Server are supported.
Webhook	sink	Can be used as HTTP POST/PUT with templates and headers
PostgreSQL	sink	Materialized View connection for reading views. Can be used with anything that reads PostgreSQL wire protocol
REST	sink	Materialized View connection for reading views. Can be used with anything that reads REST (such as notebooks, applications, and so on)
BlackHole	sink	Can be used with Flink SQL.

Using connectors with templates

Some of the connectors have default templates in Streaming SQL Console that allows you to create tables with them easily. These predefined templates not only contain the CREATE TABLE statement, but every mandatory and optional argument that is needed for the table.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click ⋮ next to **Jobs** from the **Explorer**.
4. Select New Job > Create .

   The SQL Editor for the created job opens in a tab.
5. Click Templates.
6. Select one of the connector templates.

   The corresponding CREATE TABLE statement is imported to the **SQL Editor**.

What to do next

After importing the template to the SQL Editor, you need to provide the mandatory properties and you can also fill out the optional arguments if needed. When you click Execute, the table of the chosen connector is created and listed under **Virtual Tables**.

Adding new connectors

When adding new connectors to Cloudera SQL Stream Builder, you need to specify the property list, data types and must upload the connector JAR file to the Console.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.

2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Open **External Resources** from the **Explorer** view.
4. Click ⋮ next to **Connectors**.
5. Click New Connector.

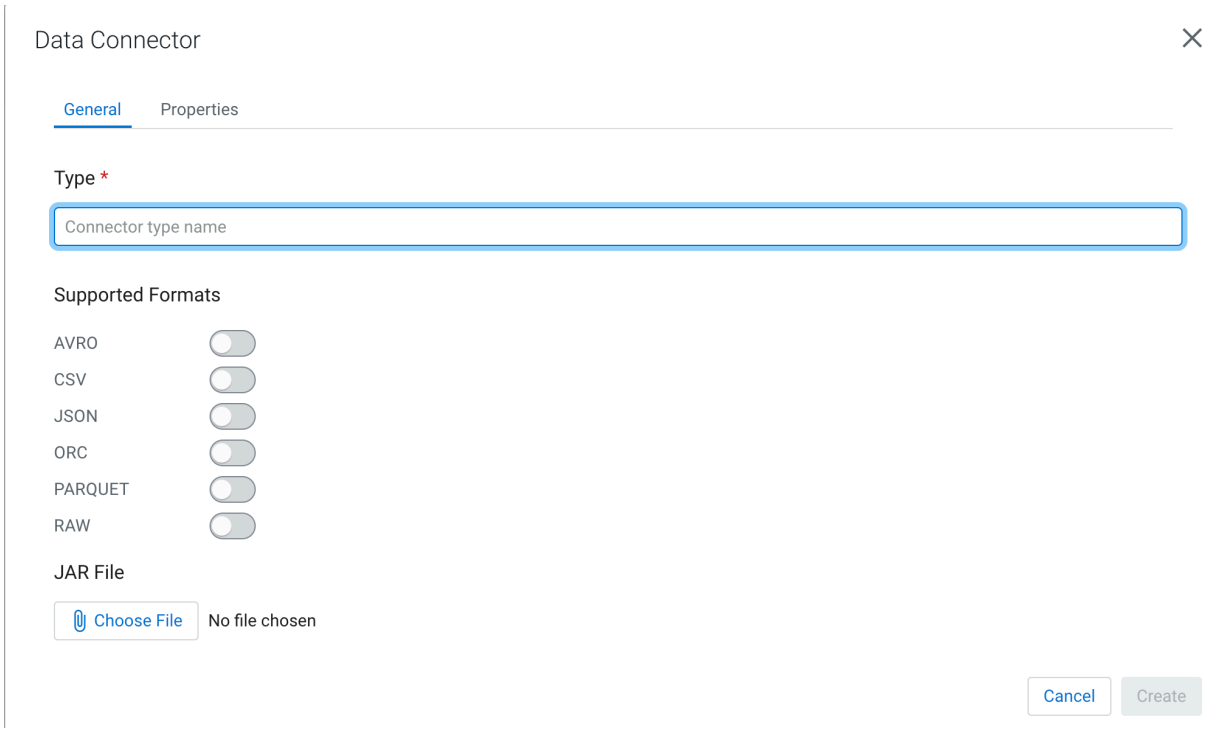

6. Provide a name for the connector as Type.
7. Select a data format to be supported for the connector from the Supported Formats.
8. Upload the connector JAR file.
9. Click Properties to add properties to the connector.
   a) Add a name to the property.
   b) Add a default value to the property.
   c) Add a description to the property.
   d) Click Required to make a property mandatory.
   e) Click Add to specify more properties.

   You can specify as many properties as needed for the connector.
10. Click Create.

    The newly added connector is listed under the **Connectors**.

Kafka connectors

When using the Kafka connector, you can choose between using an internal or external Kafka service. Based on the connector type you choose, there are mandatory fields where you must provide the correct information.

You can choose from the following Kafka connectors when creating a table in Streaming SQL Console:

Template: local-kafka

Automatically using the Kafka service that is registered in the Data Sources, and runs on the same cluster as the Cloudera SQL Stream Builder service. You can choose between JSON, Avro and CSV data types.

Type: source/sink

The following fields are mandatory to use the connector:

- scan.startup.mode: Startup mode for the Kafka consumer. group-offsets is the default value. You can choose from earliest-offset, latest-offset, timestamp and specific-offsets as startup mode.
- topic: The topic from which data is read as a source, or the topic to which data is written to. No default value is specified. You can also add a topic list in case of sources. In this case, you need to separate the topics by semicolon. You can only specify the topic-pattern or topic for the sources.
- format: The format used to deserialize and serialize the value part of Kafka messages. No default value is specified. You can use either the format or the value.format option.

Template: kafka

Using an external Kafka service as a connector. To connect to the external Kafka service, you need to speficy the Kafka brokers that are used in your deployment.

Type: source/sink

The following fields are mandatory to use the connector:

- properties.bootstrap.servers: Specifying a list of Kafka brokers that are separated by comma. No default value is specified.
- topic: The topic from which data is read as a source, or the topic to which data is written to. No default value is specified. You can also add a topic list in case of sources. In this case, you need to separate the topics by semicolon. You can only specify the topic-pattern or topic for the sources.
- format: The format used to deserialize and serialize the value part of Kafka messages. No default value is specified. You can use either the format or the value.format option.

Template: upsert-kafka

Connecting to a Kafka service in the upsert mode. This means that when using it as a source, the connector produces a changelog stream, where each data record represents an update or delete event. The value in the data records is interpreted as an update of the last value for the same key. When using the table as a sink, the connector can consume a changelog stream, and write insert/update_after data as normal Kafka message valuea. Null values are represented as delete. For more information about the upsert Kafka connector, see the Apache Flink documentation.

Type: source/sink

- properties.bootstrap.servers: Specifying a list of Kafka brokers that are separated by comma. No default value is specified.
- topic: The topic from which data is read as a soruce, or the topic to which data is written to. No default value is specified. You can also add a topic list in case of sources. In this case, you need to separate the topics by semicolon. You can only specify the topic-pattern or topic for the sources.
- key.format: The format used to deserialize and serialize the key part of Kafka messages. No default value is specified. Compared to the regular Kafka connector, the key fields are specified by the PRIMARY KEY syntax.
- value.format: The format used to deserialize and serialize the value part of Kafka messages. No default value is specified. You can use either the format or the value.format option.

Configuring deserialization policy in DDL

You can configure every supported type of Kafka connectors (local-kafka, kafka or upsert) how to handle if a message fails to deserialize which can result in job submission error. You can choose from the following configurations:

Fail

In this case an exception is thrown, and the job submission fails

Ignore

In this case the error message is ignored without any log, and the job submission is successful

Ignore and Log

In this case the error message is ignored, and the job submission is successful

Save to DLQ

In this case the error message is ignored, but you can store it in a dead-letter queue (DLQ) Kafka topic

1. Choose one of the Kafka template types from Templates.
2. Select any type of data format.

   The predefined CREATE TABLE statement is imported to the SQL Editor.
3. Fill out the Kafka template based on your requirements.
4. Search for the deserialization.failure.policy.
5. Provide the value for the error handling from the following options:

   a. ‘error’
   b. ‘ignore’
   c. ‘ignore_and_log’
   d. ‘dlq’

   If you choose the dlq option, you need to create a dedicated Kafka topic where you store the error message. In this case, you must provide the name of the created DLQ topic for the deserialization.failure.dlq.topic property.
6. Click Execute.

For more information about how to configure the error handling using the Kafka wizard, see the Deserialization tab section.

CDC connectors

You can use the Debezium Change Data Capture (CDC) connector to stream changes in real-time from MySQL, PostgreSQL, Oracle, Db2, SQL Server and feed data to Kafka, JDBC, the Webhook sink or Materialized Views using Cloudera SQL Stream Builder.

Concept of Change Data Capture

Change Data Capture (CDC) is a process to capture changes in a source system, and update the data within a downstream system or application with the changes.

The Debezium implementation offers CDC with database connectors from which real-time events are updated using Kafka and Kafka Connect. Debezium captures every row-level change in each database table of an event stream. Applications read these streams to see the change events in the same order as they occurred. The change events are routed to a Kafka topic from which Kafka Connect feeds the records to other systems and databases.

For more information about Debezium, see the official Debezium documentation.

CDC in Cloudera Streaming Analytics does not require Kafka or Kafka Connect as Debezium is implemented as a library within the Flink runtime. This means that the captured changes are propagated downstream to any connector

that Flink supports. Cloudera Streaming Analytics allows queries to be issued at change data capture time, which means filtering, grouping, joining, and so on, can be performed on the change stream as it comes from the source database.

For more information about the Flink implementation of Debezium, see the official Apache Flink documentation.

From the supported set of Debezium connectors, MySQL, PostgreSQL, Oracle, Db2, and SQL Server are supported in Cloudera Streaming Analytics.

Note: You need to configure the databases, users and permissions for the supported connectors before you are able to use them in Cloudera SQL Stream Builder. For more information about setting up the databases, see the official Debezium documentation.

JDBC connector

When using the JDBC connector, you can choose between using a PostgreSQL, MySQL, Oracle, Hive, Db2 or SQL Server databases. Based on the connector type you choose, there are mandatory fields where you must provide the correct information.

Template: jdbc

Using either PostgreSQL, MySQL, Oracle, Hive, Db2 or SQL Server as databases. When you use the JDBC connector, you must specify the JDBC database which are going to be used for the connection.

Type: source/sink

The following fields are mandatory to use the connector:

- url: The URL of the JDBC database. No default value is specified.
- table-name: The name of the JDBC table in the database that you need to connect to. No default value is specified.

Filesystem connector

When using the Filesystem connector, you can choose between HDFS, S3 and so on storage systems. Based on the connector type you choose, there are mandatory fields where you must provide the correct information.

Template: filesystem

Using either HDFS, S3 or any type of storage system. When you use the Filesystem connector, you must specify the path to the file system which are going to be used for the connection.

Type: source/sink

The following fields are mandatory to use the connector:

- path: Path to the root directory of the table data. No default value is specified.
- format: The format used for the file system. No default value is specified.

Datagen connector

The Datagen connector can be used as a source to generate random data. The Datagen connector works out of the box, no mandatory field is required to use the connector. The Datagen connector is useful when you want to experiment and try out Cloudera SQL Stream Builder or to test your SQL queries using random data.

Template: datagen

You do not need to provide any type of information when using the Datagen connector and template.

Type: source

Faker connector

The Faker connector can be used as a source to generate random data. To use the Faker connector, you need to specify the usage The Datagen connector is useful when you want to experiment and try out Cloudera SQL Stream Builder or to test your SQL queries using random data.

Template: faker

You do not need to provide any type of information when using the Datagen connector and template.

Type: source

The following fields are mandatory to use the connector:

- fields.#.expression: The Java Faker expression to generate the values for a specific field. For more information about the list and use of the Faker expressions, see the flink-faker documentation.

Blackhole connector

The Blackhole connector can be used as a sink where you can write any type of data into. The Blackhole connector works out of the box, no mandatory field is required to use the connector. The Blackhole connector is useful when you want to experiment and try out Cloudera SQL Stream Builder or to test your SQL queries using random data.

Template: blackhole

You do not need to provide any type of information when using the Blackhole connector and template.

Type: sink

Data formats

There is a set of table formats that can be used with the supported connectors of Cloudera SQL Stream Builder. The data format is a storage format that defines how to map binary data to the columns of a table. You can use the data formats based on which connector they are supported for.

Adding data formats

When adding new data formats to Cloudera SQL Stream Builder, you need to specify the property list and must upload the data format JAR file to the Console.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.

3. Open **External Resources** from the **Explorer** view.
4. Click ⋮ next to **Data Formats**.
5. Click New Connector.

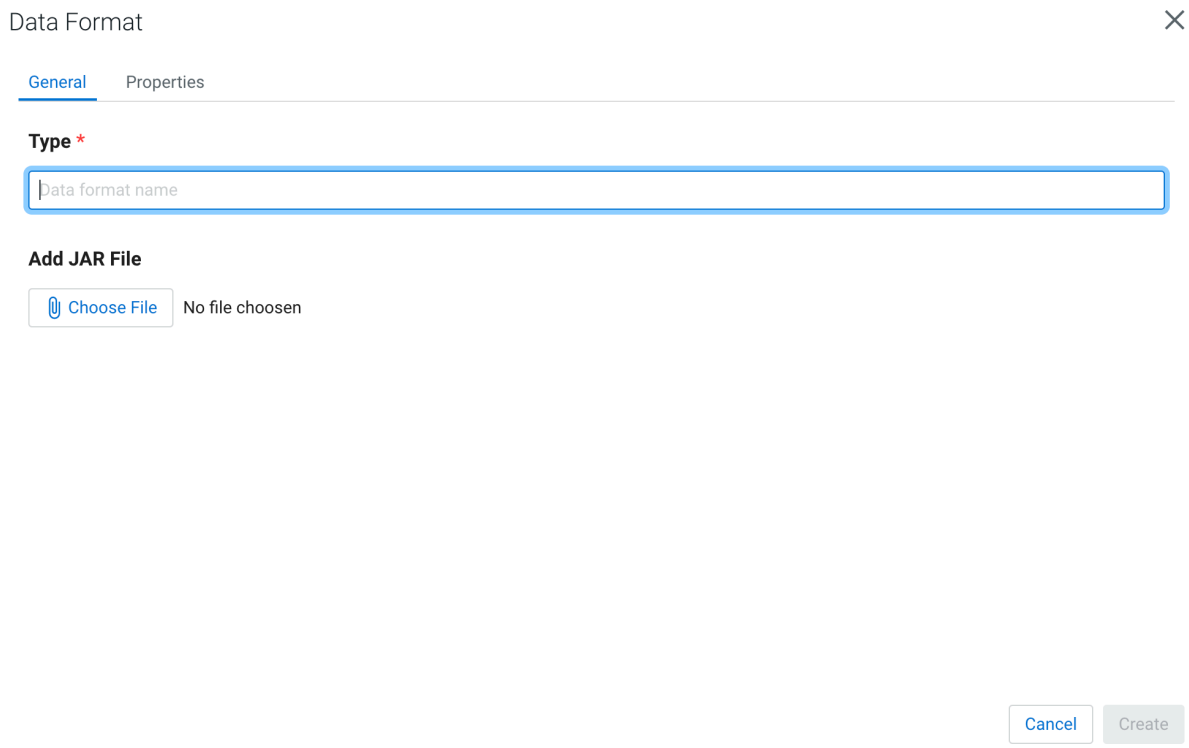

6. Provide a name for the data format as Type.
7. Upload the data format JAR file.
8. Click Properties to add properties to the data format.
   a) Add a name to the property.
   b) Add a default value to the property.
   c) Add a description to the property.
   d) Click Required to make a property mandatory.
   e) Click Add to specify more properties.

   You can specify as many properties as needed for the data format.
9. Click Create.

   The newly added data format is listed under the **Data Formats**.

Concept of tables in Cloudera SQL Stream Builder

The core abstraction for Streaming SQL is a Table which represents both inputs and outputs of the queries. Cloudera SQL Stream Builder tables are an extension of the tables used in Flink SQL to allow a bit more flexibility to the users. When creating tables in Cloudera SQL Stream Builder, you have the option to either add them manually, import them automatically or create them using Flink SQL depending on the connector you want to use.

A Table is a logical definition of the data source that includes the location and connection parameters, a schema, and any required, context specific configuration parameters. Tables can be used for both reading and writing data in most

cases. You can create and manage tables either manually or they can be automatically loaded from one of the catalogs as specified using the Data Sources section.

In SELECT queries the FROM clause defines the table sources which can be multiple tables at the same time in case of JOIN or more complex queries.

When you execute a query, the results go to the table you specify after the INSERT INTO statement in the SQL window. This allows you to create aggregations, filters, joins, and so on, and then route the results to another table. The schema for the results is the schema that you have created when you ran the query.

For example:

```
INSERT INTO air_traffic -- the name of the table sink
SELECT
lat,lon
FROM
airplanes -- the name of the table source
WHERE
icao <> 0;
```

Table types in SSB

Kafka Tables

Apache Kafka Tables represent data contained in a single Kafka topic in JSON, AVRO or CSV format. It can be defined using the Streaming SQL Console wizard or you can create Kafka tables from the pre-defined templates.

Tables from Catalogs

SSB supports Kudu, Hive and Schema Registry as catalog providers. After registering them using the Streaming SQL Console, the tables are automatically imported to SSB, and can be used in the SQL window for computations.

Note: You cannot edit the properties of the already existing tables that are automatically imported from the catalogs. To distinguish between editable and non-editable tables, in other words, user defined and catalog tables, the Edit and Delete table options are not available on the Tables page.

Flink Tables

Flink SQL tables represent tables created by the standard CREATE TABLE syntax. This supports full flexibility in defining new or derived tables and views. You can either provide the syntax by directly adding it to the SQL window or use one of the predefined DDL templates.

Webhook Tables

Webhooks can only be used as tables to write results to. When you use the Webhook Tables the result of your SQL query is sent to a specified webhook.

Iceberg Tables

You can use Apache Iceberg tables when adding Apache Hive as a catalog. The Iceberg connector of Apache Flink is implemented in Cloudera SQL Stream Builder that enables you to use Iceberg tables managed in the Hive catalog.

Creating Kafka tables using wizard

After registering a Kafka data source, you can use the Kafka table wizard in Streaming SQL Console to create a Kafka table.

About this task

You can query your streaming data using Kafka tables in Cloudera SQL Stream Builder. You have the option to use the Kafka service in your environment, or connect to an external Kafka service. When creating Kafka tables you can

use the Add Kafka wizard, the predefined templates or you can directly add a custom CREATE TABLE statement with the required properties in the SQL window.

You can also create Kafka tables using one of the Kafka templates. For more information, see the Kafka connectors and Using connectors with templates sections.

Before you begin

- Make sure that you have registered Kafka as a Data Source.
- Make sure that you have created topics in Kafka.

  Important: When creating the topic for the Kafka sink, make sure to not use log compaction as it can cause the SQL job to fail.

- Make sure there is generated data in the Kafka topic.
- Make sure that you have the right permissions set in Ranger.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click ⋮ next to **Virtual Tables**.

4. Click New Kafka Table .

   The **Kafka Table** window appears.

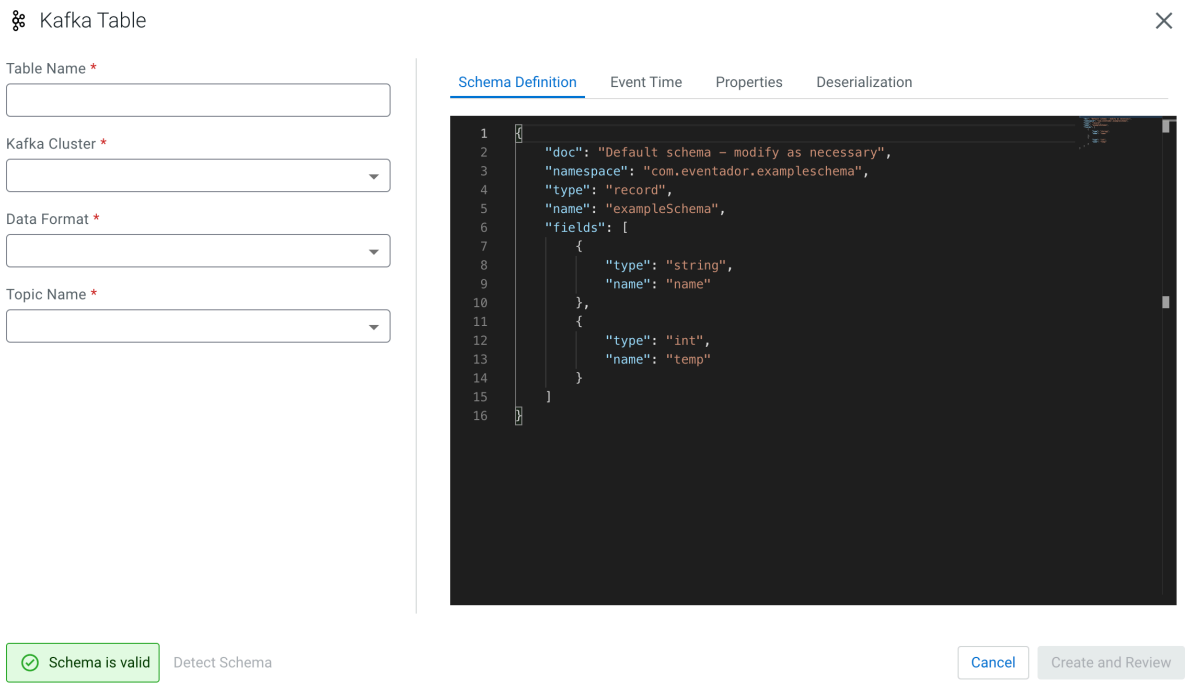

5. Provide a Table Name.

   Note: You will use this name in the FROM clause when running the SQL statement.

6. Select a registered Kafka provider as Kafka cluster.
7. Select the Data format.
   - You can select JSON as data format.
   - You can select AVRO as data format.
8. Select a Kafka topic from the list.

   Note: The automatically created topics for the websocket output is also listed here. Select the topic you want to use for the SQL job.

9. Add a customized schema to Schema Definition or click Detect Schema to read a sample of the JSON messages, and automatically infer the schema.

   Note: The Detect Schema functionality is only available for JSON data. If there are no messages in the topic, then no schema will be inferred. If your schema contains a field named timestamp, this causes a schema validation error as timestamp is a reserved word used for Kafka internal timestamps.

10. Customize your Kafka Table with the following options:
    a) Configure the Event Time if you do not want to use the default Kafka Timestamps.
    b) Configure an Input Transform on the Data Transformations tab.

       Note: The Input Transformation is only supported for JSON data formats.

    c) Configure any Kafka properties required on the Properties tab.
    d) Select a policy for deserialization errors on the Deserialization tab.

    For more information about how to configure the Kafka table, see the *Configuring Kafka tables* section.

11. Click Create and Review.

Results

The Kafka Table is ready to be used for the SQL job either at the FROM or at the INSERT INTO statements.

Configuring Kafka tables

The user-defined Kafka table can be configured based on the schema, event time, input transformations and other Kafka specific properties using either the Kafka wizard or DDL.

Schema Definition tab

When using the Add Kafka table wizard on the Streaming SQL Console, you can configure the schema under the Schema tab.

Schema is defined for a given Kafka source when the source is created. The data contained in the Kafka topic can either be in JSON or AVRO format.

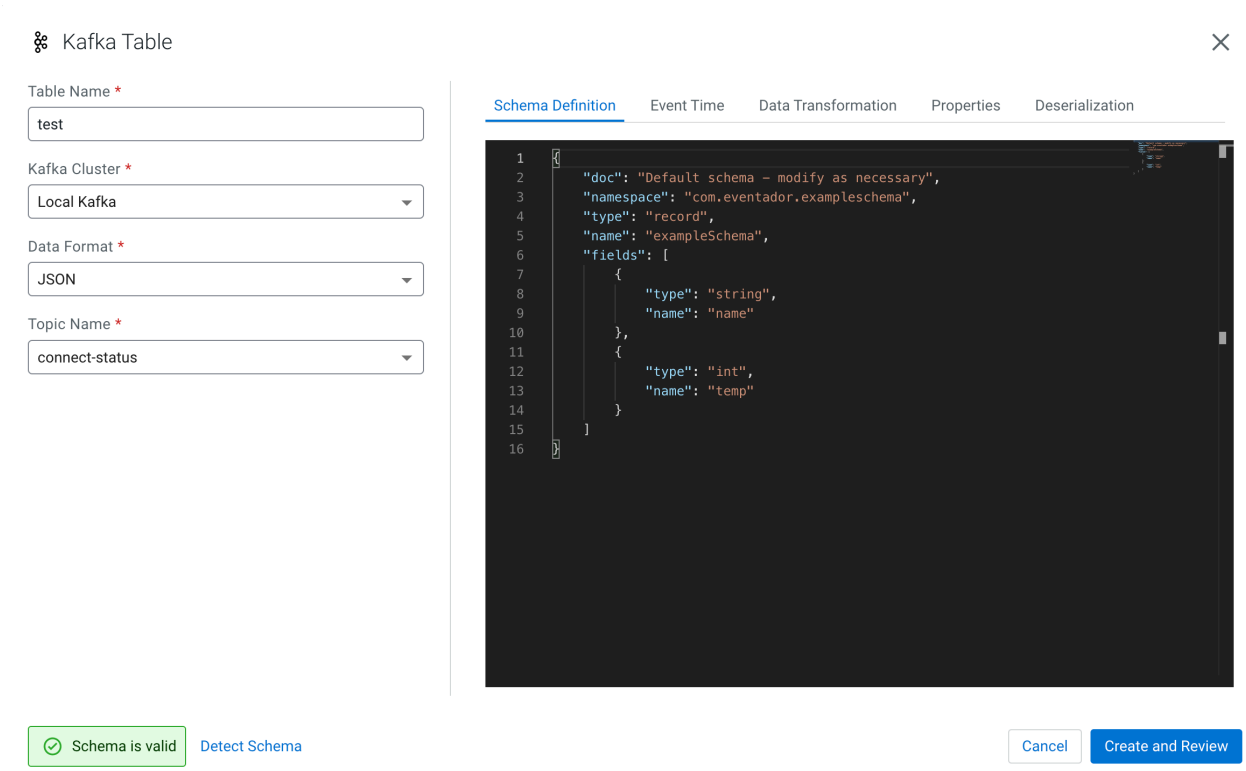

When specifying a schema you can either paste it to the Schema Definition field or click the Detect schema button to identify the schema used on the generated data. The Detect Schema functionality is only available for JSON data.

Note: If your schema contains a field named timestamp, this causes a schema validation error as timestamp is a reserved word used for Kafka internal timestamps.

Event Time tab

When using the Add Kafka table wizard on the Streaming SQL Console, you can configure the event time under the Event Time tab.

You can specify Watermark Definitions when adding a Kafka table. Watermarks use an event time attribute and have a watermark strategy, and can be used for various time-based operations. The Event Time tab provides the following properties to configure the event time field and watermark for the Kafka stream:

- Input Timestamp Column: name of the timestamp column in the Kafka topic from where the watermarks are mapped to the Event Time Column of the Kafka table
- Event Time Column: default or custom name of the resulting timestamp column where the watermarks are going to be mapped in the created Kafka table

- Watermark seconds: number of seconds used in the watermark strategy. The watermark is defined by the current event timestamp minus this value.

You have the following options to configure the watermark strategy for the Kafka tables:

- Using the default Kafka Timestamps setting
- Using the default Kafka Timestamps setting, but providing custom name for the Event Time Column
- Not using the default Kafka Timestamps setting, and providing all of the Kafka timestamp information manually
- Not using watermark strategy for the Kafka table

Using the default Kafka Timestamp setting

By default, the Use Kafka Timestamps feature is enabled when you create the Kafka table. In the Event Time Column, the new event time field is extracted from the Kafka message header with the *'EVENTTIMESTAMP'* predefined column name.

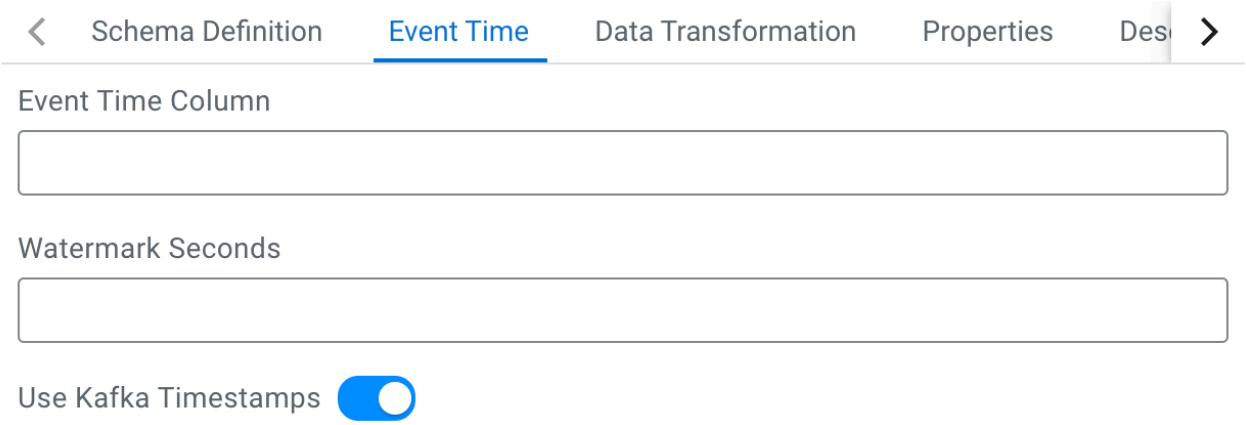

After saving your changes, you can view the created DDL syntax for the table next to the SQL Editor on the **Console** page.

The following DDL example shows the default setting of the Event Time Column and Watermark Seconds where the corresponding fields were not modified.

```
'eventTimestamp' TIMESTAMPS(3) METADATA FROM 'timestamp',
WATERMARK FOR 'eventTimestamp' AS 'eventTimestamp' - INTERVAL '3' SECOND
```

Using the default Kafka Timestamp setting with custom Event Time Column name

When you want to modify the timestamp field of the DDL from the stream itself, you must provide a custom name of the Event Time Column. You can also add a custom value to the Watermark Seconds. The following example shows that *'ETS'* is the custom name for the Event Time Column, and *'4'* is the custom value for the Watermark Seconds.

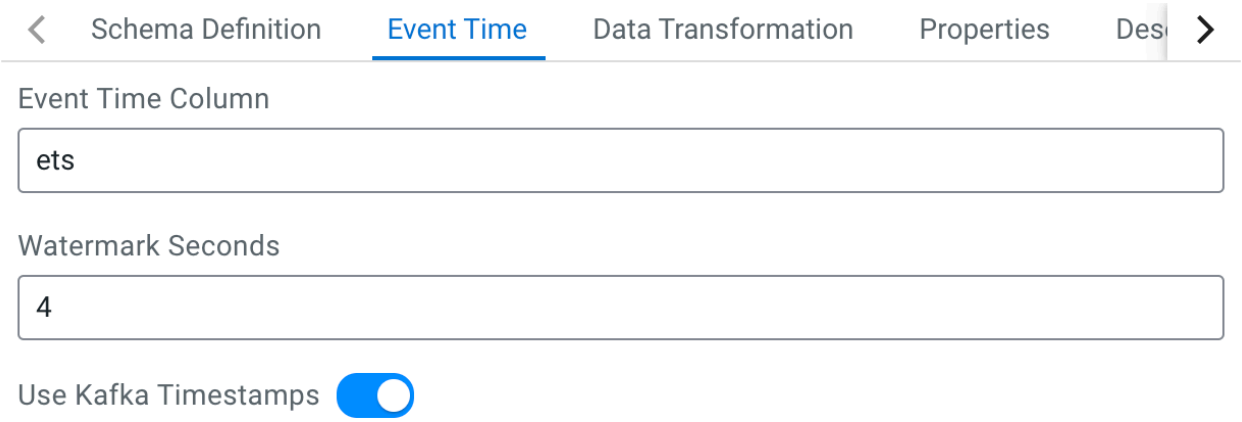

The Event Time Column can only be modified if the following requirements are met for the timestamp field of the Input Timestamp Column:

- The column type must be "long".
- The format must be in epoch (in milliseconds since January 1, 1970).

The DDL syntax should reflect the changes made for the watermark strategy as shown in the following example:

```
'ets' TIMESTAMP(3) METADATA FROM 'timestamp',
WATERMARK FOR 'ets' - INTERVAL '4' SECOND
```

Manually providing the Kafka timestamp information

When you want to manually configure the watermark strategy of the Kafka table, you can provide the timestamp column name from the Kafka source, and add a custom column name for the resulting Kafka table. Make sure that you provide the correct column name for the Input Timestamp Column that exactly matches the column name in the Kafka source data.

To manually provide information for the watermark strategy, disable the Use Kafka Timestamps feature using the toggle, and provide the following information to the column name fields:

- Input Timestamp Column: name of the timestamp field in the Kafka source
- Event Time Column: predefined *'EVENTTIMESTAMP'* name or custom column name of the timestamp field in the created Kafka table

As an example, you have a timestamp column in the source Kafka topic named *'TS'*, and want to add a new timestamp column in your Kafka table as *'EVENT_TIME'*. You provide the original timestamp column name in the Input Timestamp Column as *'TS'*, and add the custom *'EVENT_TIME'* name to the Event Time Column.

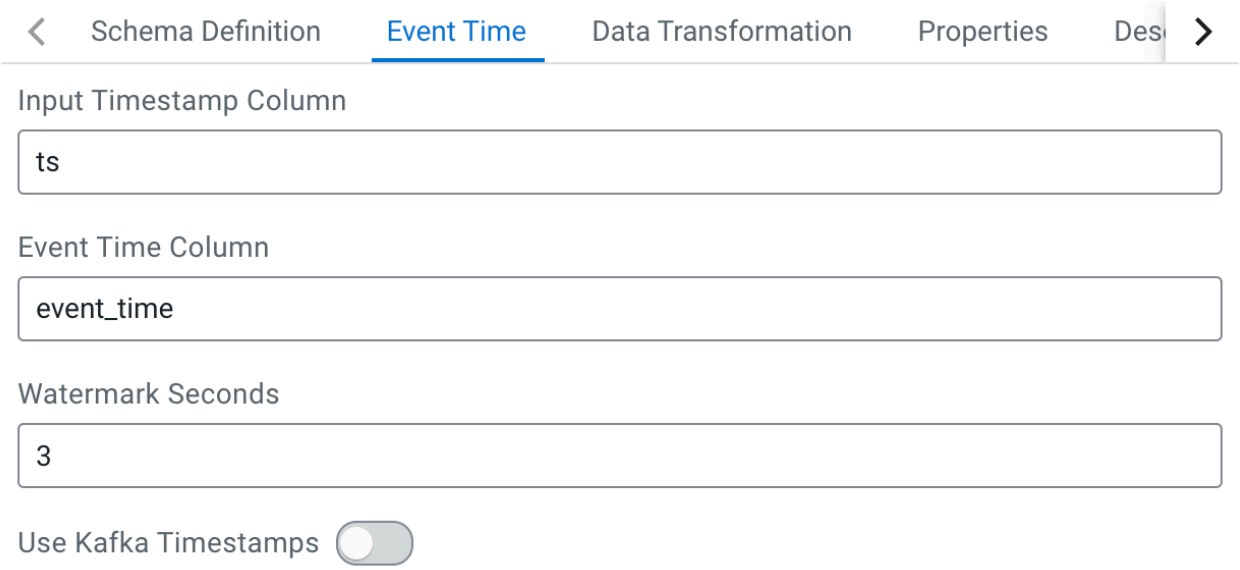

This results in that the watermarks from the *'TS'* column is going to be mapped to the *'EVENT_TIME'* column of the created Kafka table. As *'EVENT_TIME'* will become the timestamp column name in the Kafka table, you must use the custom name (in this example the *'EVENT_TIME'*) when querying the Kafka stream. This configuration of the timestamp columns is optional.

The Event Time Column can only be modified if the following requirements are met for the timestamp field of the Input Timestamp Column:

- The column type must be "long".
- The format must be in epoch (in milliseconds since January 1, 1970).

Not using watermark strategy for Kafka table

In case you do not need any watermark strategies, disable the Use Kafka Timestamps feature using the toggle, and leave the column and seconds field empty.

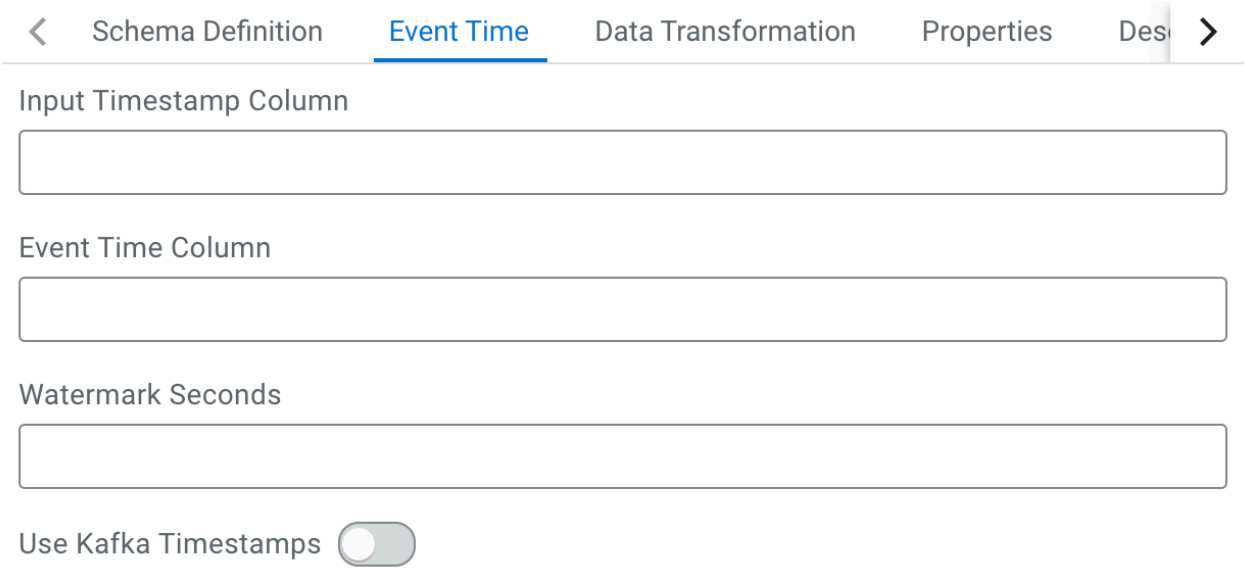

Note: Flink validates the input and output schemas of the data. You can only insert data into a Kafka topic, if the input and output data matches based on the schema defined for the topic. When customizing the timestamp column, make sure that the output data has the same schema.

Note: When configuring the timestamp for the Kafka tables, you must consider the timezone setting of your environment as it can affect the results of your query. For more information, see the Known Issues in the Release Notes.

Data Transformations tab

When using the Add Kafka table wizard on the Streaming SQL Console, you can apply input transformation under the Transformations tab. Input transformations can be used to clean or arrange the incoming data from the source using javascript functions.

Input Transforms are a powerful way to clean, modify, and arrange data that is poorly organized, has changing format, and has data that is not needed or otherwise hard to use. With the Input Transform feature of SQL Stream Builder, you can create a javascript function to transform the data after it has been consumed from a Kafk topic, and before you run SQL queries on the data.

You can use Input Transforms in the following situations:

- The source is not in your control, for example, data feed from a third-party provider
- The format is hard to change, for example, a legacy feed, other teams of feeds within your organization
- The messages are inconsistent
- The data from the sources do not have uniform keys, or without keys (like nested arrays), but are still in a valid JSON format
- The schema you want does not match the incoming topic

Note: When using Input Transforms the schema you define for the Kafka table is applied on the output of the transformed data.

•

You can use the Input Transforms on Kafka tables that have the following characteristics:

- Allows one transformation per source.
- Takes record as a JSON-formatted string input variable. The input is always named record.
- Emits the output of the last line to the calling JVM. It could be any variable name. In the following example, out and emit is used as a JSON-formatted string.

A basic input transformation looks like this:

```
var out = JSON.parse(record.value);     // record is input, parse JSON f
ormatted string to object
                                                 // add more transformatio
ns if needed
JSON.stringify(out);                             // emit JSON formatted
 string of object
```

Adding transformation to Kafka tables

When adding a Kafka table using the wizard, you can specify the input transformation on the Data Transformation tab. You have one of the following steps to apply an input transformation:

1. Add your javascript transformation code to the **Data Transformation** window.

   Make sure the output of your transform matches the Schema definition detected or defined for the Kafka table.
2. Click Install default template and schema.

   The Install Default template and schema option fills out the Data Transformation box with a template that you can use to create the Input Transform, and matches the schema with the format.

Kafka record metadata access

There are cases when it is required to access additional metadata from the Kafka record to implement the correct processing logic. SQL Stream Builder has access to this information using the Input Transforms functionality.

The following attributes are supported in the headers:

```
record.topic
record.key
record.value
record.headers
record.offset
record.partition
```

For example, an input transformation can be expressed as the following:

```
var out = JSON.parse(record);
out['topic'] = message.topic;
out['partition'] = message.partition;
JSON.stringify(out);
```

For which you define a schema manually, or use the Detect Schema feature:

```
{
  "name": "myschema",
  "type": "record",
  "namespace": "com.cloudera.test",
  "fields": [
    {
      "name": "id",
      "type": "int"
    },
    {
      "name": "topic",
      "type": "string"
    },
    {
      "name": "partition",
      "type": "string"
    }
  ]
}
```

The attribute record.headers is an array that can be iterated over:

```
var out = JSON.parse(record);
var header = JSON.parse(record.headers);
var interested_keys = ['DC'];               // should match schema defini
tion

out['topic'] = record.topic;
out['partition'] = record.partition;
Object.keys(header).forEach(function(key) {
    if (interested_keys.indexOf(key) > -1){  // if match found for schema,
set value
        out[key] = header[key];
    }
});
JSON.stringify(out);
```

For which you define a schema as follows:

```
{
  "name": "myschema",
  "type": "record",
  "namespace": "com.cloudera.test",
  "fields": [
    {
      "name": "id",
      "type": "int"
    },
    {
      "name": "topic",
      "type": "string"
    },
    {
      "name": "partition",
      "type": "string"
    },
    {
      "name": "DC",
      "type": "string"
    }
```

```
  ]
}
```

Properties tab

When using the Add Kafka table wizard on the Streaming SQL Console, you can configure the properties under the Properties tab.

You can specify certain properties to define your Kafka source in detail. You can also add customized properties additionally to the default ones. To create properties, you need to give a name to the property and provide a value for it, then click Actions.

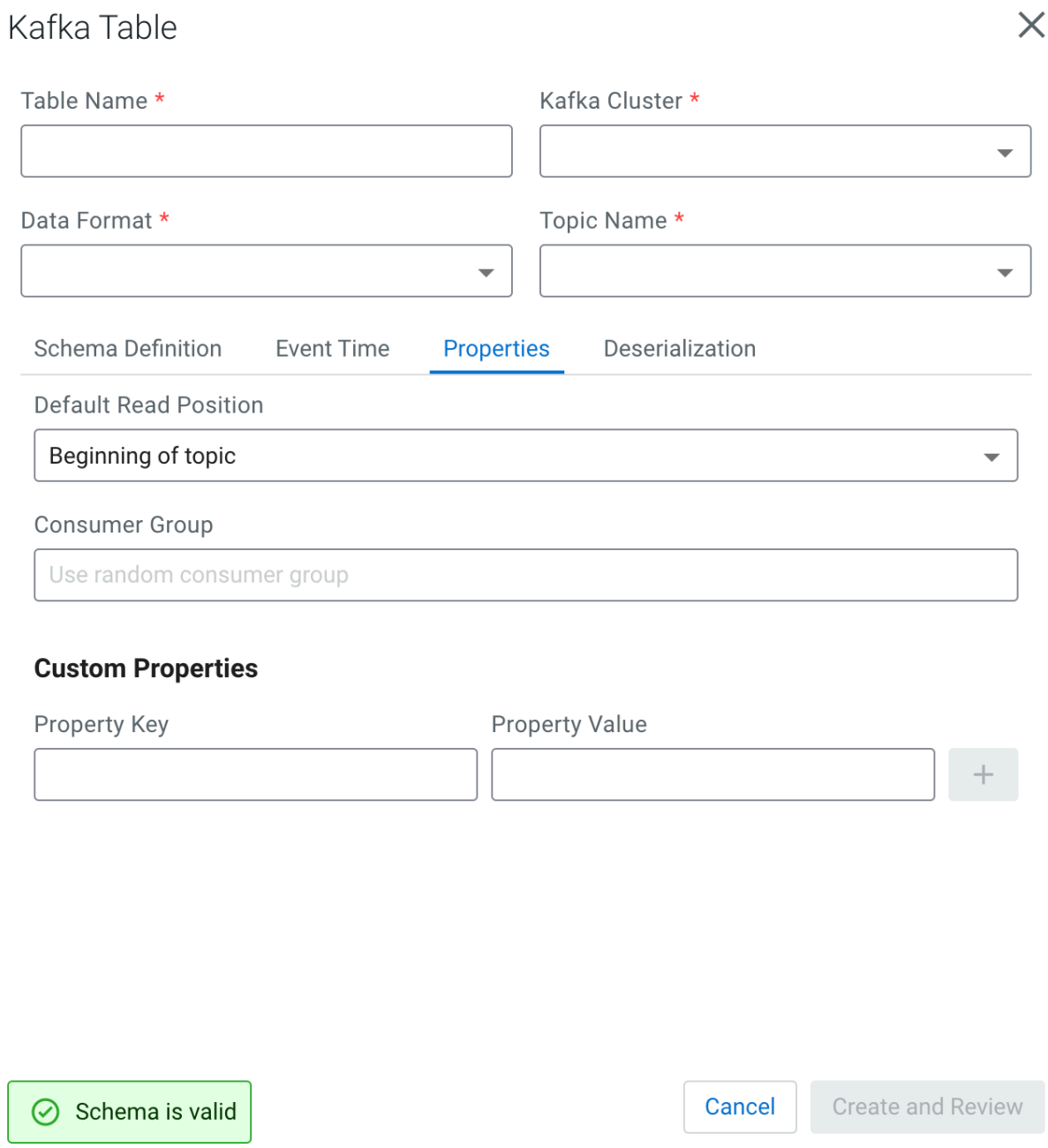

Deserialization tab

When creating a Kafka table, you can configure how to handle errors due to schema mismatch using DDL or the Kafka wizard.

You can configure every supported type of Kafka connectors (local-kafka, kafka or upsert) how to handle if a message fails to deserialize which can result in job submission error. You can choose from the following configurations:

Fail

In this case an exception is thrown, and the job submission fails

Ignore

In this case the error message is ignored without any log, and the job submission is successful

Ignore and Log

In this case the error message is ignored, and the job submission is successful

Save to DLQ

In this case the error message is ignored, but you can store it in a dead-letter queue (DLQ) Kafka topic

Using the Kafka wizard

When you create the Kafka table using the wizard on the Streaming SQL Console, you can configure the error handling with the following steps:

1. Navigate to the Streaming SQL Console.
   a. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Cloudera Streaming Analytics cluster from the list of Data Hub clusters.
   c. Select Cloudera SQL Stream Builder from the list of services.
2. Click Create Job or select a previous job on the **Getting Started** page.

   You are redirected to the **Console** page.
3. Select Add tables > Apache Kafka .

   The **Add Kafka table** window appears
4. Select Deserialization tab.

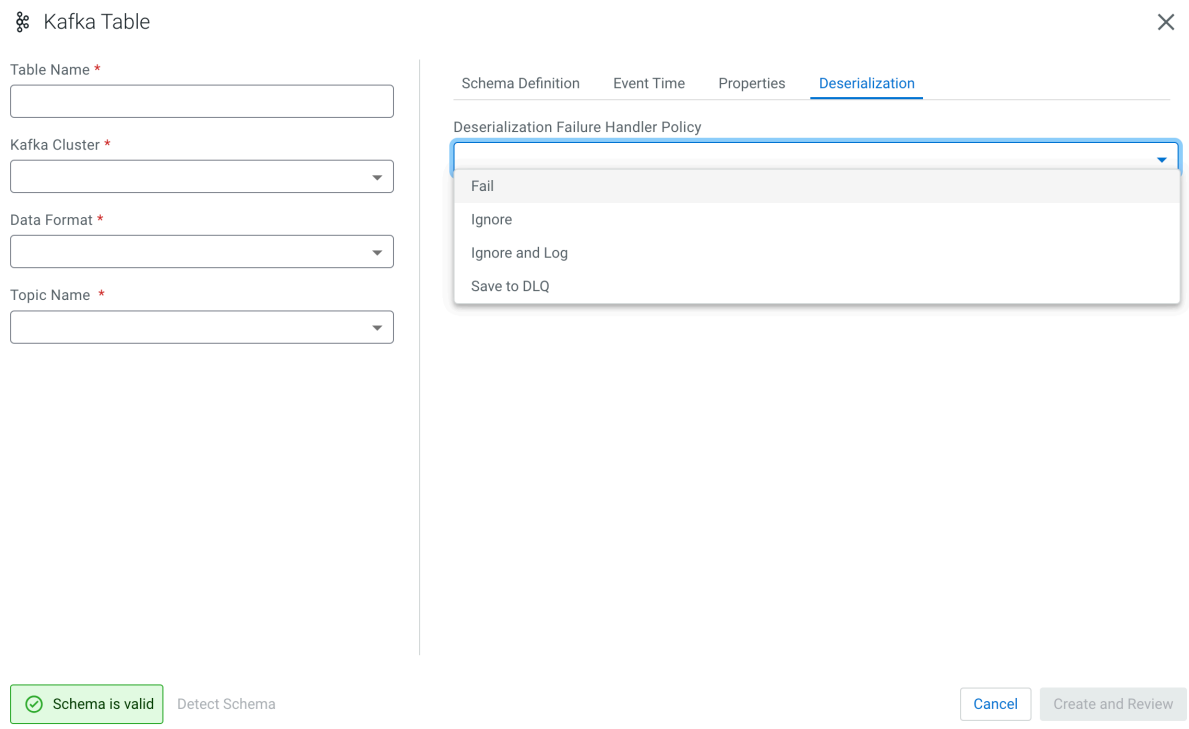

5. Choose from the following policy options under **Deserialization Policy**:
   - Fail
   - Ignore
   - Ignore and Log
   - Save to DLQ

     If you choose the Save to DLQ option, you need to create a dedicated Kafka topic where you store the error message. In this case, you must select the dedicated DLQ topic from DLQ Topic Name.

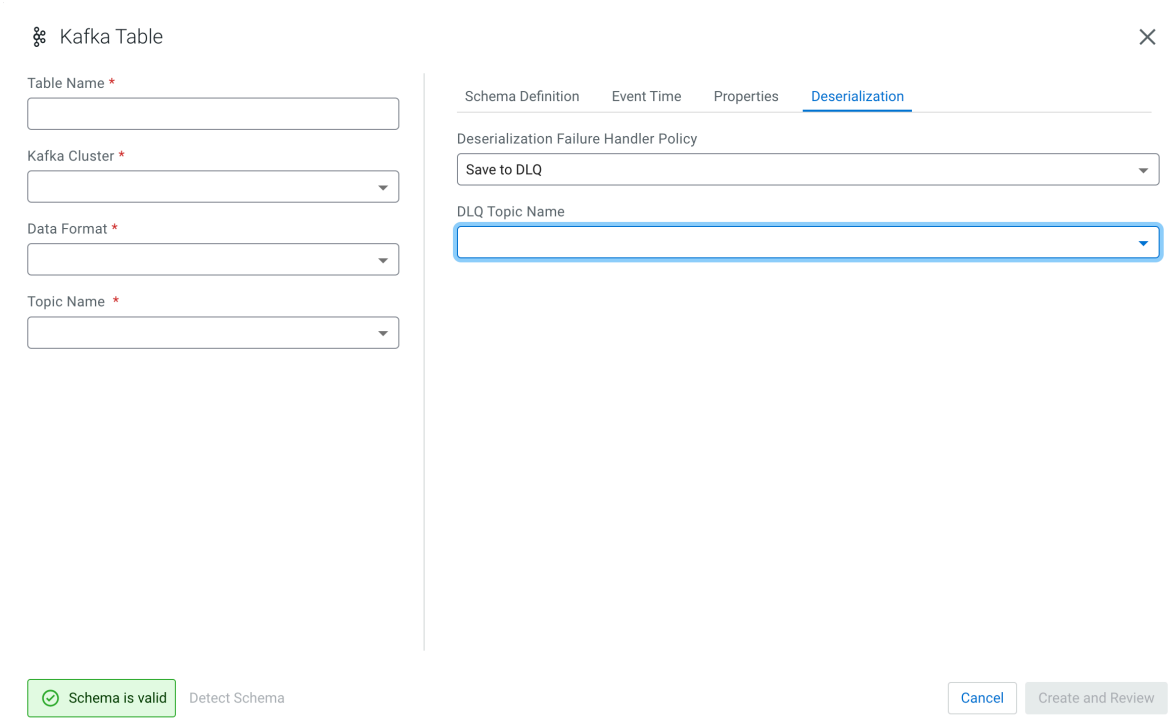

   - Click Create and Review.

For more information about how to configure the error handling using DDL, see the Kafka connectors section.

Assigning Kafka keys in streaming queries

Based on the Sticky Partitioning strategy of Kafka, when null keyed events are sent to a topic, they are randomly distributed in smaller batches within the partitions.

As the results of the SQL Stream queries by default do not include a key, when written to a Kafka table, the Sticky Partitioning strategy is used. In many cases, it is useful to have more fine-grained control over how events are distributed within the partitions. You can achieve this in SSB by configuring a custom key based on your specific workload.

For example:

```
SELECT sensor_name AS _eventKey --sensor_name becomes the key in the output
kafka topic
FROM sensors
WHERE eventTimestamp > current_timestamp;
```

To configure keys in DDL-defined tables (those that are configured using the Templates), refer to the official Flink Kafka SQL Connector documentation for more information (specifically the key.format and key.fields options).

Performance & Scalability

The Kafka and Cloudera SQL Stream Builder integration enables you to use the Kafka-specific syntax to customize your SQL queries based on your deployment and use case.

You can achieve high performance and scalability with Cloudera SQL Stream Builder, but the proper configuration and design of the source Kafka topic is critical. Cloudera SQL Stream Builder can read a maximum of one thread per Kafka partition. You can achieve the highest performance configuration when setting the Cloudera SQL Stream Builder threads equal to or higher than the number of Kafka partitions.

If the number of partitions is less than the number of Cloudera SQL Stream Builder threads, then Cloudera SQL Stream Builder has idle threads and messages show up in the logs indicating as such. For more information about Kafka partitioning, see the Kafka partitions documentation.

Creating Webhook tables

You can configure the webhook table to perform an HTTP action per message (default) or to create code that controls the frequency (for instance, every N messages). When developing webhook sinks, it is recommended to check your webhook before pointing at your true destination.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click ⋮ next to **Virtual Tables**.

4. Select New Webhook Table .

   The **Webhook Table** window appears.

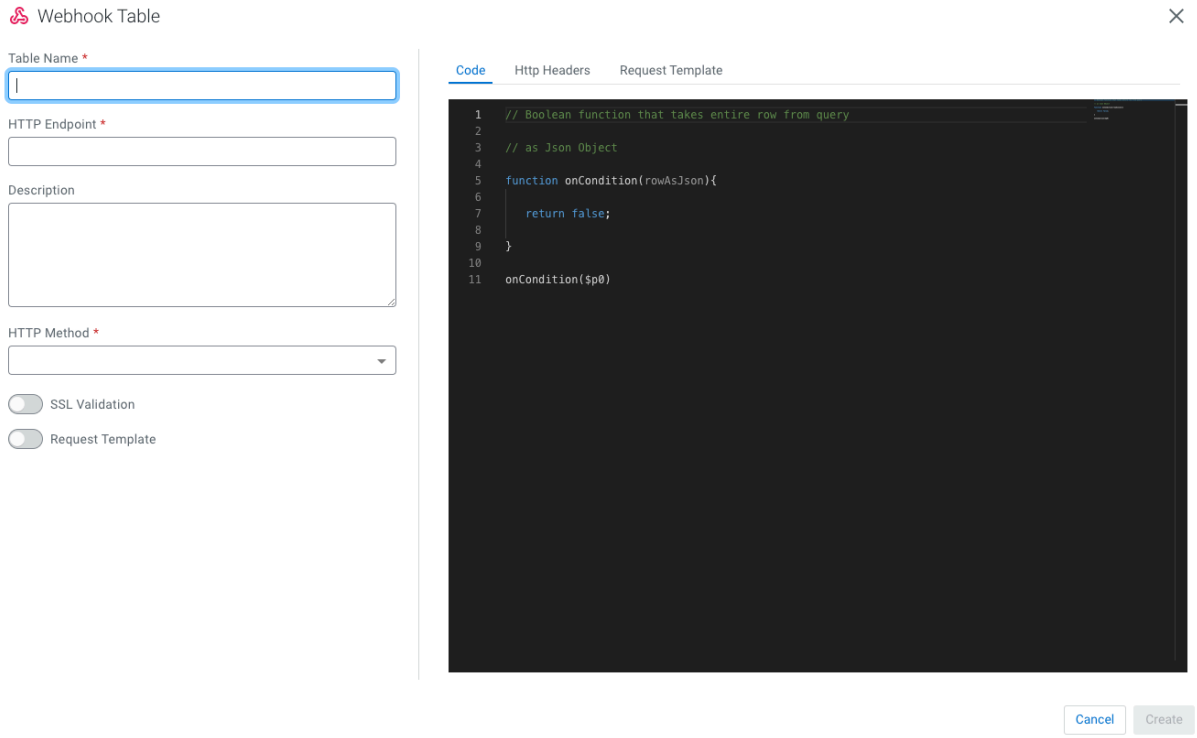

5. Provide a name to the Table.
6. Enter an HTTP endpoint. The endpoint must start with http:// or https://.

   Note: You can use hookbin for testing of the webhook sink. Paste the hookbin endpoint into the text field, and inspect the output on the hookbin site. Once you have the right output result, then point it at your final endpoint.
7. Add a Description about the webhook sink.
8. Select POST or PUT in the HTTP Method select box.
9. Choose to Disable SSL Validation, if needed.
10. Enable Request Template, if needed.
    a) If you selected Yes, then the template defined in the Request Template tab is used for output.

       This is useful if the service you are posting requires a particular data output format. The data format must be a valid JSON format, and use "${columnname}" to represent fields. For example, a template for use with Pagerduty looks like this:

```
{
   "incident":{
      "type":"incident",
      "title":"${icao} is too high!",
        "body":{
   "type":"incident_body",
         "details":"Airplane with id ${icao} has reached an altitude of
 ${altitude} meters."
 }
    }
}
```

11. In the Code editor, you can specify a code block that controls how the webhook displays the data.

    For a webhook that is called for each message the following code is used:

```
// Boolean function that takes entire row from query as Json Object
function onCondition(rowAsJson)
{return true;   // return false here for no-op, or plug in custom
        logic}
onCondition($p0)
```

    Note: The rowAsJson is the result of the SQL Stream query being run in the {"name":"value"} format.

12. Add HTTP headers using the HTTP Headers tab, if needed.

    Headers are name:value header elements. For instance, Content-Type:application/json, etc.

13. Click Create.

Results

The Webhook table is ready to be used after the INSERT INTO statement in your SQL query.

Creating Flink tables using Templates

You can use the predefined templates to create tables by choosing one of the connector templates on the Console page of Streaming SQL Console. The Flink SQL templates are predefined examples of CREATE TABLE statements which you can fill out with your job specific values.

You can create tables using the predefined templates in SQL Stream Builder. The predefined templates consist of the CREATE TABLE statement, and every connection property that is needed for the given connector. You only need to fill out the templates and execute them in the Streaming SQL Console to create the table. When filling out the templates, you can add configuration based on what is specified in the given connector template. You can also customize the table name, column names and timestamp information for a template.

You can choose from the following templates based on the connector type:

Blackhole

The BlackHole connector can be used to write all input records into. It is designed for high performance testing and UDF to output, not a substantive sink.

Datagen

The Data generator connector can be used to sample randomly generated data in SSB. You can use this connector to try out and test SQL queries as records are generated until the job is running.

Faker

The Faker connector can be used to generate fake data based on the Java faker expression. You can use this connector to try out and test SQL queries as records are generated until the job is running.

Filesystem

The filesystem connector can be used to access partitioned files in file systems, and enables reading and writing from a local or distributed file system such as local, HDFS, S3 and so on. You only need to specify the data format that is used in the file system. You can choose from the following formats:

- Avro
- JSON
- CSV
- ORC
- Parquet

JDBC

The JDBC connector enables reading data from and writing data into any relational databases with a JDBC driver. You can use PostgreSQL, MySQL and Hive as databases for the connector.

Debezium CDC

You can use the Debezium CDC connector to stream changes in real-time from MySQL, PostgreSQL, Oracle, Db2 and SQL Server into Kafka. Debezium provides a unified format schema for changelog and supports serializing messages using JSON and Avro.

You can access and import the Templates from Streaming SQL Console:

1. Navigate to the **Streaming SQL Console**.
   a. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Cloudera Streaming Analytics cluster from the list of Cloudera Data Hub clusters.
   c. Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Click Create Job or select a previous job on the **Getting Started** page.

   You are redirected to the **Console** page.
3. Click Templates at the SQL Editor.
4. Select the template you want to use.

   The template is imported to the SQL window.
5. Customize the fields of the template.
6. Click Execute.

   The table is created based on the selected template, and appears next to the SQL Editor.

For full reference on the Flink SQL DDL functionality, see the official Apache Flink documentation.

Creating Iceberg tables

Apache Iceberg is an open, high-performance table format for organizing datasets that can contain petabytes of data. Iceberg can be used to add tables to computing engines, such as Apache Hive and Apache Flink, from which the data can be queried using SQL.

As Iceberg is integrated as a connector to Flink, you can use the table format the same way for Cloudera SQL Stream Builder. Both the V1 and V2 version specifics are supported by the Flink connector.For more information, about the version changes, see the Apache Iceberg documentation.

Table 2: SQL feature support for the Iceberg and Cloudera SQL Stream Builder integration

Feature	SSB
Create catalog	Supported
Create database	Supported
Create table	Supported
Alter table	Supported **Note:** Only the altering table properties is supported, which does not inclide column and partition changes.
Drop table	Supported
Select	Supported
Insert into	Supported
Insert overwrite	Supported

Feature	SSB
Metadata tables	Supported
Rewrite files action	Supported
Upsert	Technical preview[1]
Equality delete	Not supported

You can use a Hive service that is located on the same cluster as Cloudera SQL Stream Builder, and you can use a Hive service that is available in a Cloudera Data Warehouse cluster.

Note: Currently, Flink upsert operations are not supported when writing to tables in Cloudera Data Warehouse.

Using Hive for Iceberg integration

When using the Hive service located on your cluster, you can add it as a catalog on Streaming SQL Console. Before creating the Iceberg table, ensure that you have added Hive as a catalog using the steps described in Adding catalogs documentation.

After setting up Hive for Cloudera SQL Stream Builder, you can define Iceberg as a connector in the CREATE TABLE statement as the example shows below:

```
CREATE TABLE ‘ssb’.’ssb_default’.’iceberg_hive’ (
‘column_int’ INT,
‘column_str’ STRING,
) WITH (
‘connector’ = ‘iceberg’,
‘catalog-database’ = ‘test_db’,
‘catalog-type’ = ‘hive’,
‘catalog-name’ = ‘iceberg_hive_catalog’,
‘catalog-table’ = ‘iceberg_hive_table’,
‘ssb-hive-catalog’ = ‘ssb_hive_catalog’
‘engine.hive.enabled’ = ‘true’
)
```

The following properties are mandatory when using the Iceberg connector:

Property	Example	Description
catalog-database	test_db	The Iceberg database name in the backend catalog, uses the current Flink database name by default. It will be created automatically if it does not exist when writing records into the Flink table
catalog-type	hive	Type of the catalog
catalog-name	iceberg_hive_catalog	User-specified catalog name. It is required as the connector does not have any default value.
catalog-table	iceberg_hive_table	Name of the Iceberg table in the backend catalog.
ssb-hive-catalog	ssb_hive_catalog	The name of the Hive catalog that you have registered in SSB. The configuration can be used when creating Iceberg catalogs as well.

[1] The upsert feature of the Iceberg and Cloudera SQL Stream Builder integration is in Technical Preview and not ready for production deployment. Cloudera encourages you to explore these features in non-production environments and provide feedback on your experiences through the Cloudera Community Forums.

Using the Hive service from Cloudera Data Warehouse

Before you begin

- Ensure that the ssb user has access to the all - database, table policy under the Hadoop SQL service.
- Create ID Broker mapping for the workload user.
- Create an Iceberg table using Hue in the Hive virtual warehouse on the Cloudera Data Warehouse cluster.

Steps

1. Access Cloudera Manager of the Cloudera Data Warehouse environment.
2. Click Cluster View Client Configuration URLs Hive Metastore .
3. Download the hive-conf tarball.

   The downloaded tarball needs to be uploaded to the manager node of the Flink cluster.
4. Copy the hive-conf to the manager node of the Flink cluster.

```
scp <location>/hive-conf <workload_username>@<manager_node_FQDN>:.
Password:<your_workload_password>
```

5. Extract the file under the /tmp directory that is accessible by the ssb user using the following commands:

```
mkdir /opt/hive-conf
cd /opt/hive-conf
tar -xvf /path/to/hive-conf.tar.gz
chmod a+x .
chmod -R a+r .
```

6. Access the Streaming SQL Console and create the table using the following example or select Iceberg from the **Template** drop-down and fill out the required parameters:

```
CREATE TABLE my_ssb_table (
    id   INT
) WITH (
    'connector' = 'iceberg',
    'catalog-name' = 'internal-use',
    'catalog-database' = 'default',
    'catalog-table' = 'my_iceberg_table',
    'hive-conf-dir' = '/opt/hive-conf'
);
```

SQL jobs

Creating and naming SQL jobs

You need to create or select an already existing SQL job on the Streaming SQL Console to be able to submit SQL queries.

About this task

You can manage your SQL jobs and resources using the **Explorer** view or the **Home** tab of your project. Every job and resource related action is the same for both views of Streaming SQL Console.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click ⋮ New Job on the **Explorer** view.
4. Provide a Name to your SQL job.

   You have the following options to name your SQL job:

   - Use the predefined name in the Job Name field when opening the Streaming SQL Console
   - Use the Generate Job Name button to generate a new random name for your job
   - Manually provide the name of the job to the Job Name field
5. Click Create.

   You are redirected to the SQL Editor where you can submit your SQL queries, create and manage Materialized Views and configure your SQL jobs.

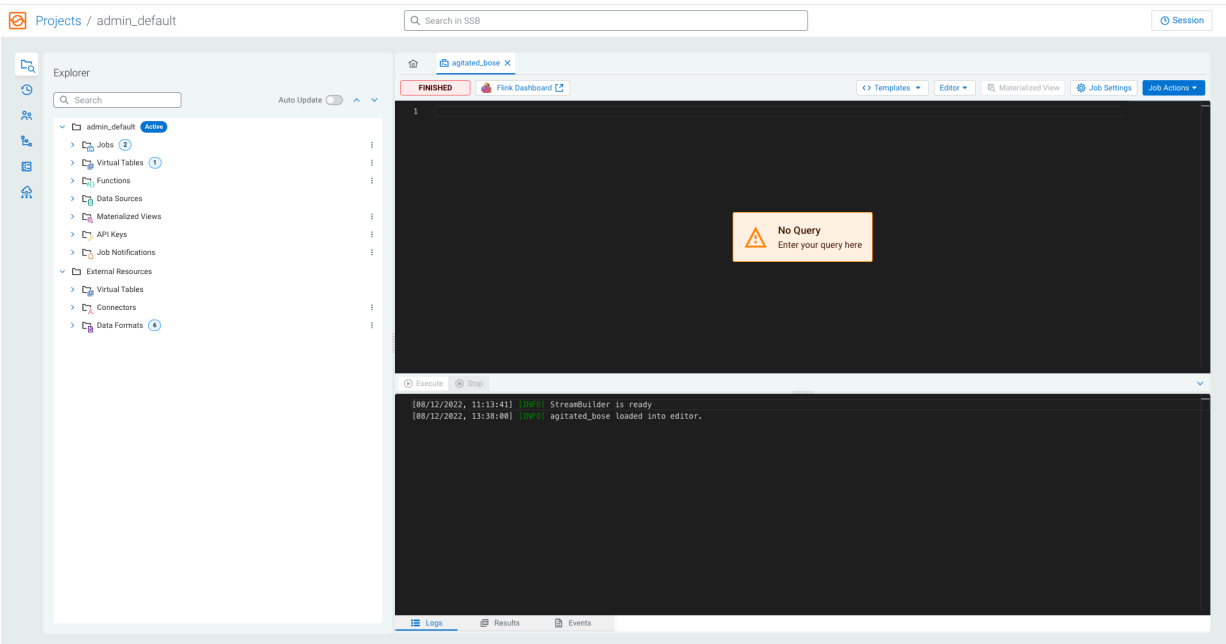

What to do next

After a new job is created, you can save the job and its settings using the Job Actions Save button. When you make a change to the SQL query, job settings or create a Materialized View, you have the option to save the changes with the job. You can also revert your changes to the latest saved version of the job using the Revert Changes button. You can also delete the job from the SQL Editor using the Delete button.

Running SQL Stream jobs

Every time you run an SQL statement in the SQL Stream console, it becomes a job and runs on the deployment as a Flink job. You can manage the running jobs using the Jobs tab on the UI.

About this task

There are two logical phases to run a job:

1. Parse: The SQL is parsed and checked for validity and then compared against the virtual table schema(s) for correct typing and key/columns.
2. Execution: If the parse phase is successful, a job is dynamically created, and runs on an open slot on your cluster. The job is a valid Flink job.

Before you begin

- Make sure that you have registered a Data Source if you use the Kafka service on your cluster.
- Make sure that you have added Kudu, Hive or Schema Registry as a catalog if you use them for your SQL job.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click ⋮ next to **Jobs** from the **Explorer**.
4. Select New Job .

   The SQL Editor for the created job opens in a tab.
5. Provide a name for the SQL job.
   a) Optionally, you can click Generate Random Name to generate a name for the SQL job.
6. Click Create.
7. Create a table using one of the following options:
   - Using the Add Kafka or Webhook table wizard.
   - Using the Templates.
   - Add your custom CREATE TABLE statement to the SQL window.

8. Add a SQL query to the SQL Editor.

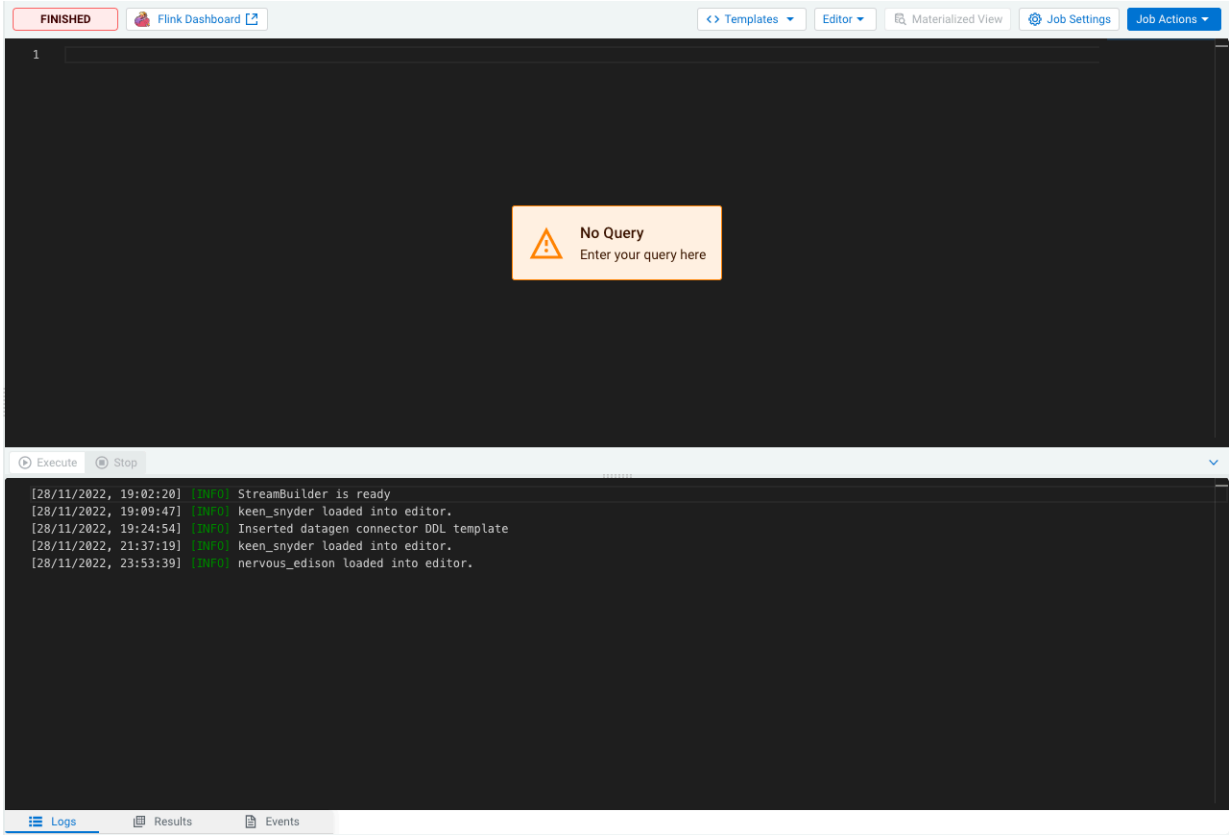

Note: You cannot start a job without adding a SQL statement in the SQL editor window. In case, there are no SQL statements provided and you click Execute , the following error message is displayed: You must provide a SQL query.

When starting a job, the number of slots consumed on the specified cluster is equal to the parallelism setting. The default is one slot. To change the parallelism setting and more job related configurations, click Job Settings.

9. Click Execute.

The Logs window updates the status of SSB.

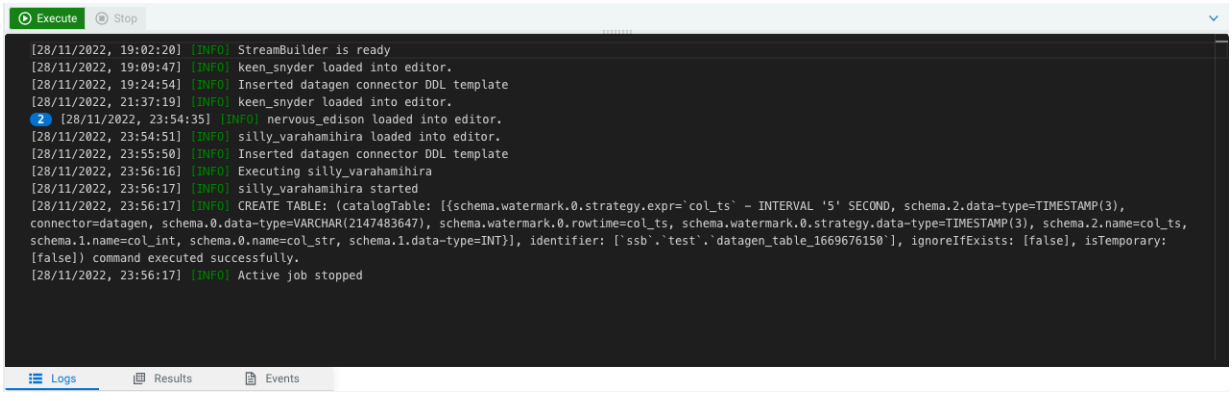

10. Click Results to check the sampled data.

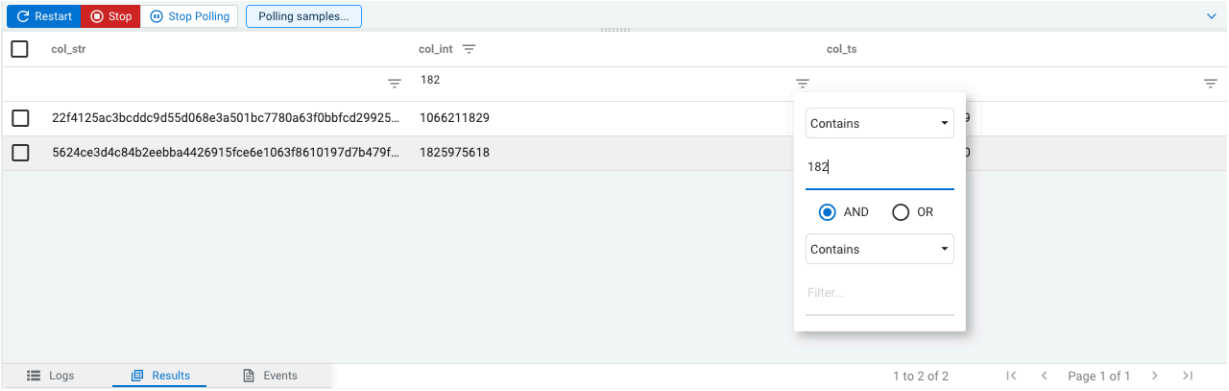

You can pause or restart the sampling of a SQL job with the Pause and Refresh buttons. The status of the polling can also be seen next to the Pause button. The Results tab also allows you to change the order of the samples, and to filter the samples using different types of conditions that you can also combine. For changing the order, click on the name of the column, the direction of the ordering is indicated next to the column name by arrows.

For filtering, you can either click the button next to the column name or under the row of the column name. The filtering value can be changed by typing the values directly to the filtering row, the conditions need to be changed using the filtering arrow.

11. Click Events to check the job related information.

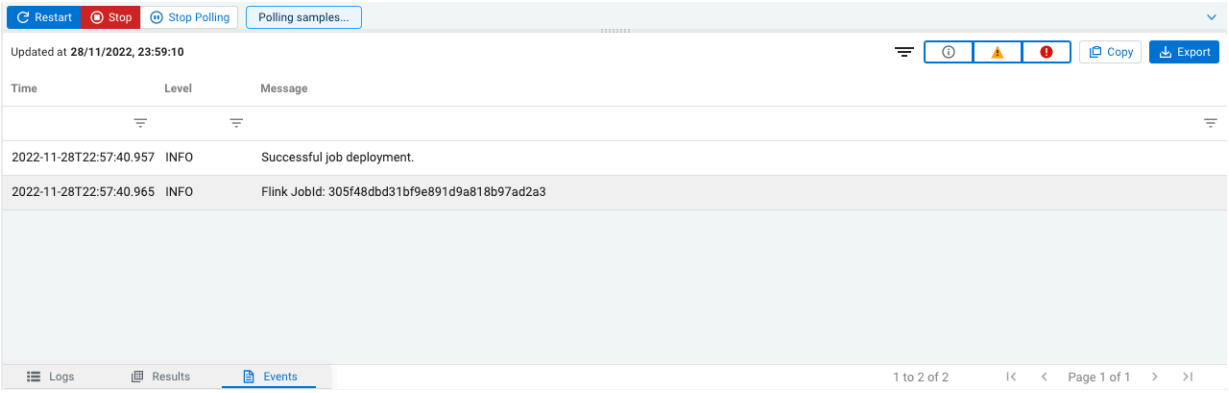

Results

A job is generated that runs the SQL continuously on the stream of data from a table, and pushes the results to a table, to the Results tab or to a Materialized View.

Configuring SQL job settings

If you need to further customize your SQL Stream job, you can add more advanced features to configure the job restarting method and time, threads for parallelism, sample behavior, exactly once processing and restoring from savepoint.

Before running a SQL query, you can configure advanced features by clicking on the Job Settings button at the SQL Editor.

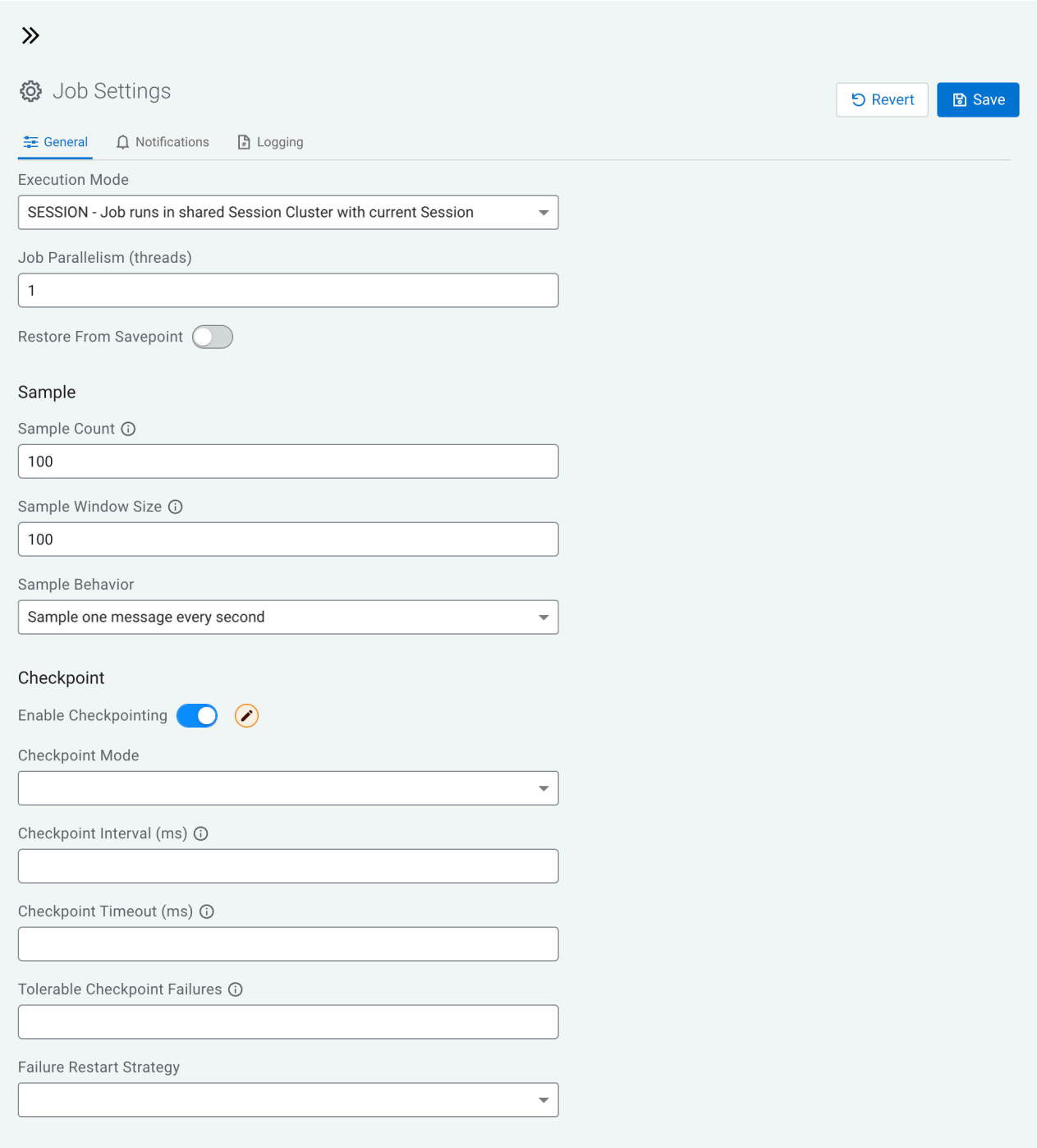

Execution Mode

You can select the execution mode for the SQL job that can be a session or a per-job mode. For more information, see the Executing SQL jobs in production mode documentation.

Job parallelism (threads)

The number of threads to start to process the job. Each thread consumes a slot on the cluster. When the Job Parallelism is set to 1, the job consumes the least resources. If the data source supports parallel reads, increasing the parallelism can raise the maximum throughput. For example, when using Kafka as a data source, setting the parallelism to the equal number as the partitions of the topic can be a starting point for performance tuning.

Restore From Savepoint

You can enable or disable restoring a SQL job from a Flink savepoint after stopping it. The savepoint is saved under hdfs:///user/flink/savepoints by default.

Sample Count

The number of sample entries to poll before sampling stops. To poll an unlimited number of sample entries, add 0 as the *SAMPLE COUNT* value.

Sample Window Size

The number of sample entries from active sampling displayed in the Results tab. To display an unlimited number of sample entries, add 0 as the *SAMPLE WINDOW SIZE* value.

If the *SAMPLE WINDOW SIZE* value is lower than the *SAMPLE COUNT* value, only the number (set in *SAMPLE WINDOW SIZE*) of the most recent of sample entries will be displayed. If the *SAMPLE WINDOW SIZE* value is higher than the value of *SAMPLE COUNT*, only the number (set in *SAMPLE COUNT*) of sample entries will be displayed.

Sample Behavior

You have the following options to choose the behavior of the sampled data under the Results tab:

- Sample all messages
- Sample one message every second
- Sample one message every five seconds

Enable Checkpointing

You can enable and disable checkpointing for a SQL job. By default the checkpointing is enabled.

Checkpoint Mode

Switching between checkpointing modes. You can choose between At Least Once or Exactly Once.

Checkpoint Interval

The time in milliseconds between checkpointing attempts.

Checkpoint Timeout

The maximum time in milliseconds until a checkpointing attempt is timed out.

Tolerable Checkpoint Failures

The number of checkpointing attempts until the job is aborted.

Failure Restart Strategy

Switching between restarting strategies when checkpointing is failed. You can choose between enabling and disabling auto recovery. By default auto recovery is enabled for checkpointing.

Adjusting logging configuration in Advanced Settings

You can customize the logging configurations for Cloudera SQL Stream Builder jobs on the Streaming SQL Console globally or at job level. Adjusting the log configuration enables you to control the log levels of all the underlying libraries: Flink, Hadoop, Kafka, Zookeeper, other common libraries, and connectors to get more or less information in your job's log.

About this task

The customization of the log configuration works differently whether it's applied globally or at job level based on the job deployment mode.

Note:

When executing a job, the order of applying configurations is as follows:

1. the job's runtime configuration,
2. the global log configuration,
3. and finally the default log configuration.

The first one found defined will be applied to the job.

Global logging configuration

Important: Only users with admin rights can access the global configuration. To grant admin rights, the user needs to be added to the ssb.admin property in Cloudera Manager.

1. Navigate to the Streaming SQL Console.
2. a. Navigate to Cloudera Management Console > Environments, and select the environment where you have created your cluster.
   b. Select the Cloudera Streaming Analytics cluster from the list of Cloudera Data Hubs.
   c. Select Streaming SQL Console from the list of services.
3. The Streaming SQL Console opens in a new window.
4. Click the Configuration tab.
5. Find the ssb.admins property.

   Tip: You can type ssb.admins into the filter bar to narrow the list of configuration properties.

6. Click the + icon on the right hand side of the last username.
7. Add the username to the new row.
8. Write a reason for change and click Save changes.
9. Restart the Cloudera SQL Stream Builder instance for the changes to be applied.

To set the global logging configuration:

1. Navigate to the Streaming SQL Console.
   a. Navigate to Cloudera Management Console > Environments, and select the environment where you have created your cluster.
   b. Select the Cloudera Streaming Analytics cluster from the list of Cloudera Data Hubs.
   c. Select Streaming SQL Console from the list of services.
2. The **Streaming SQL Console** opens in a new window.
3. In the left hand navigation click Configuration.
4. If no global configuration is present, acknowledge the system notice by clicking Set global configuration.
5. In the editor, set the configuration.
6. Click Save.

Note: Before you add a global configuration, the editor contains the default logging configuration.

Session mode

The execution.target is set to *YARN-SESSION* mode, this is the default execution mode.

Note: Settings here will overwrite the global configuration.

The log configuration is set at the start time if the Flink YARN session is applied to every job execution. For example, the current log configuration is applied if and only if the Flink YARN session is not set on the Session tab of the Compose page.

Note: The Reset Session button only resets the SSB Session, not the underlying Flink YARN session. To do that, you have to kill the YARN application that is indicated under Flink Yarn Session on the Session tab.

Per-job mode

The execution.target is set to *YARN-PER-JOB* mode.

Note: Settings here will overwrite the global configuration.

When you change the default execution mode to per-job, the currently applied log configuration is going to be used for the job. To configure the execution mode, you need to start the SQL query with the following line:

```
SET 'execution.target'='yarn-per-job';
```

Procedure

1. Click New Job or select a previous job on the **Jobs** page.
   You are redirected to the **SQL Editor** of the job.
2. Click Job Settings.
3. Click Logging.

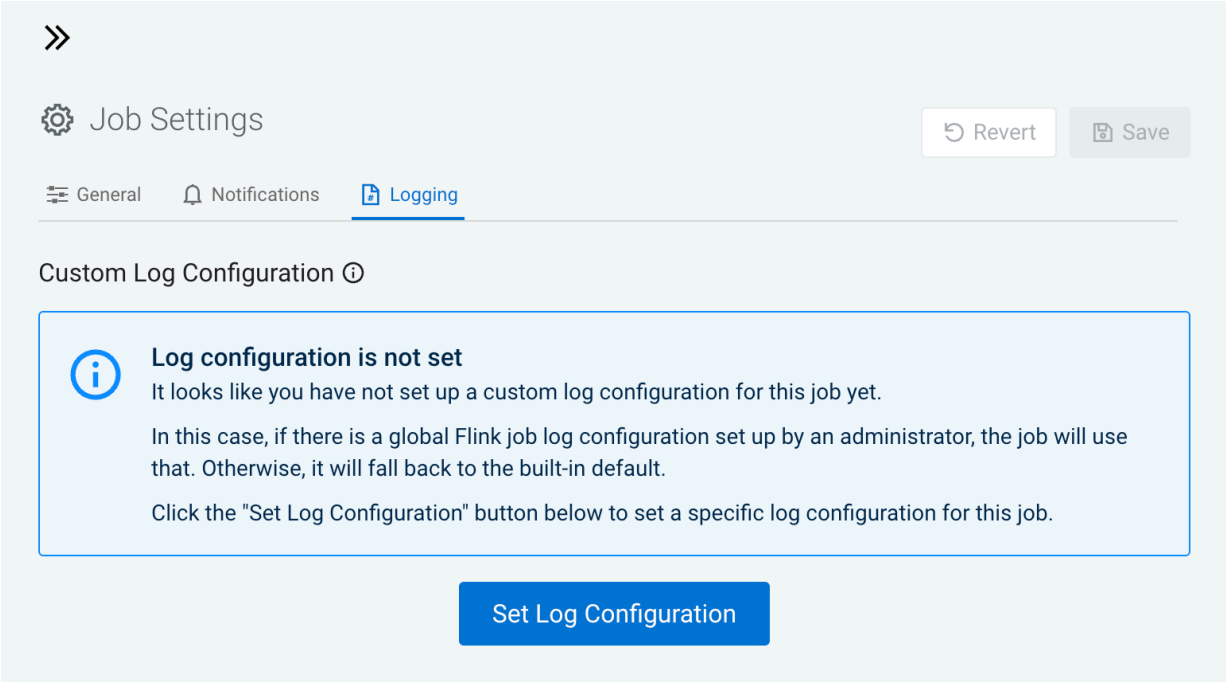

4. Click Set Log Configuration.

5. Modify the settings based on your requirements.

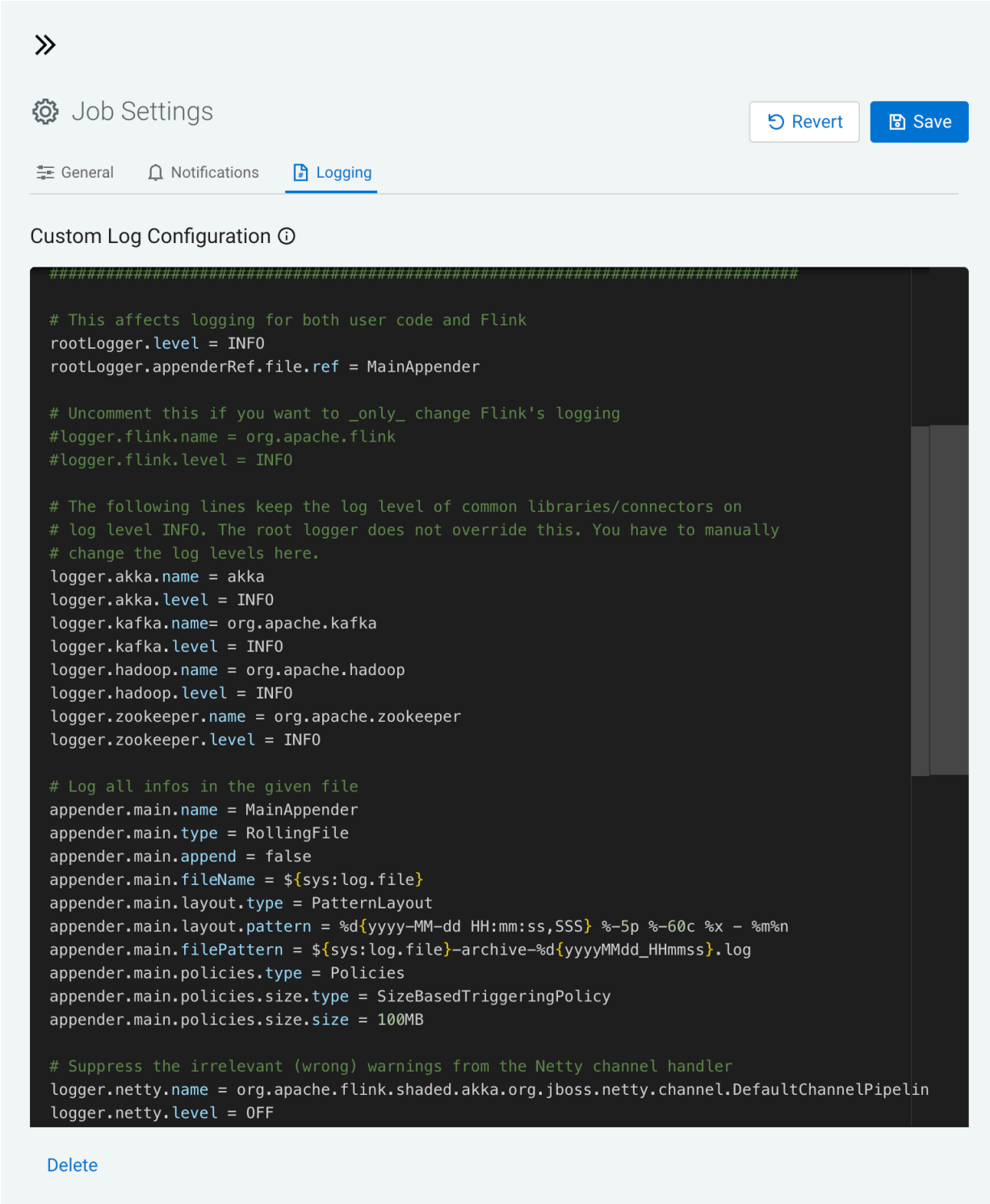

6. Click Save. (You can restore the default logging configuration through clicking Delete and Set Log Configuration again.)
7. Close the **Job Settings** window.
8. Add and execute a SQL statement.
9. Click **SQL Jobs**.
10. Search for the job you have executed previously.

11. Click **Flink Dashboard**.

    The **Flink Dashboard** opens in a new window.

12. Click Task Managers > Logs .

    The log information appears in the log window based on your custom configurations.

Configuring YARN queue for SQL jobs

You can configure the YARN application queue with a custom value for a SQL job using the Streaming SQL Console.

With YARN queues, you can deploy applications to a specific subset of nodes with separate or limited resources, according to configuration. This enables you to have a separate execution environment with limited resources to experiment with new applications without impacting any production operation.

The YARN application queue can only be configured in a per-job execution target using the production mode of SQL Stream Builder. This means that the default value of the YARN queue does not change, and you can only customize it to the specific job that is executed in the production mode.

The default value of YARN application queue can be configured in Cloudera Manager. After accessing the SQL Stream Builder service on your cluster, you need to open the configurations, and search for YARN application queue where you can change the default value. This default value is overwritten when using the SET statement in production mode for specifying the YARN queue for a SQL job.

For more information about running jobs in production mode, see the Executing SQL jobs in production mode section.

You can configure the YARN application queue using the SET statement in Streaming SQL Console as the following code example shows:

```
--PROD
set 'execution.target' = 'yarn-per-job';
set 'yarn.application.queue' = 'my-queue';
select * from datagen_table_1648801198
```

Configuring state backend for Cloudera SQL Stream Builder

The same state backend configuration is applied to the SQL jobs in Cloudera SQL Stream Builder as configured for the Flink jobs. While Hashmap stores data as object on Java heap, RocksDB can be used to store larger state that does not fit easily in memory. The RocksDB state backend uses a combination of fast in-memory cache and optimized disk based lookups to manage state.

The following state backend options are available for your Flink application:

RocksDB

By default, RocksDB is configured as the state backend for Flink. The RocksDB state backend holds in-flight data in a RocksDB database that is stored in the TaskManager local data directories and performs asynchronous snapshots. The data is stored as serialized byte arrays. The arrays are defined by the type serializer. This results in key comparisons being byte-wise instead of the hash Code() and equals() Java methods in the Hashmap state backend. The RocksDB state backend is recommended for large state and key/value states, long windows and high-availability setups. The amount of state is limited by the amount of disk space available. This also means that the maximum throughput that can be achieved will be lower as all reads/writes through the RocksDB backend have to go through deserialization and serialization.

Hashmap

The Hashmap state backend holds data internally as objects on the Java heap. The key/value state and window operators hold hash tables that store the values, triggers and so on. The Hashmap state backend is recommended for moderate state and key/value states, long windows and high-availability setups.

You can choose between Hashmap and RocksDB as the state backend based on your performance and scalability requirements. State can be accessed and updated faster with Hashmap on the Java heap, but the size is limited by the available memory in the cluster. Even though RocksDB can scale based on available disk space and is the only state backend to support incremental snapshots, accessing and updating the state requires more resources that can affect the performance compared to storing the state in-memory. For more information about the State backends in Flink, see the Apache Flink documentation.

You can configure the state backend for your streaming application using Cloudera Manager or you can also specify the state backend in the SQL editor in Streaming SQL Console.

For Cloudera Manager

1. Open your cluster in Cloudera Manager.
2. Select Flink from the list of services.
3. Click Configuration.
4. Search for state.backend and select HASHMAP or ROCKSDB based on your requirements. ROCKSDB is set by default.

5. Click Save changes.

When configuring the state backend in Cloudera Manager, the configuration serves as a default global setting and is applied for every Cloudera SQL Stream Builder job. You can override the default set in Cloudera Manager by specifying the state backend directly when submitting your Cloudera SQL Stream Builder job.

For SQL Editor

The per-job state backend can be configured by setting the state.backend.type to hashmap or rock sdb on the Streaming SQL Console when executing your SQL job in a per-job mode as shown in the following example:

```
SET 'execution.target' = 'yarn-per-job';
SET 'state.backend.type' = 'rocksdb';
```

```
SET 'execution.target' = 'yarn-per-job';
SET 'state.backend.type' = hashmap;
```

Note: In case you use Hashmap as state backend, it is recommended to switch to RocksDB when a job fails with the following exception:

```
Caused by: java.lang.OutOfMemoryError: GC overhead limit exceeded
```

Configuring RocksDB

When choosing RocksDB as state backend, you can also adjust how much memory RocksDB should use as a cache to increase lookup performance by setting the memory managed fraction of the TaskManagers.

The default fraction value is 0.4, but with larger cache requirements you need to increase this value together with the total memory size. For more information, see the Set up TaskManager Memory and the Memory Tuning Guide documentations:

For Cloudera Manager

1. Open your cluster in Cloudera Manager.
2. Select Flink from the list of services.

3. Click Configuration.
4. Search for taskmanager.memory.managed.fraction and change the default value as needed.

5. Click Save changes.

For SQL Editor

You can configure the taskmanager.memory.managed.fraction on Streaming SQL Console with executing a SET statement when submitting your SQL job in a per-job mode as shown in the following example:

```
SET 'execution.target' = 'yarn-per-job';
SET 'taskmanager.memory.managed.fraction' = '0.4';
```

RocksDB also has predefined options that can be changed based on your system configurations:

- DEFAULT: option set by default and applicable for all settings.
- SPINNING_DISK_OPTIMIZED: option for regular spinning disks configured for better performance. The option is configured with the following values:
  - CompactionStyle=LEVEL
  - LevelCompactionDynamicLevelBytes=true
  - MaxBackgroundJobs=4
  - MaxOpenFiles=-1
- SPINNING_DISK_OPTIMIZED_HIGH_MEM: option for regular spinning hard disks configured for better performance, but with higher memory consumption. The option is configured with the following values:
  - BlockCacheSize=256 MBytes
  - BlockSize=128 KBytes
  - FilterPolicy=BloomFilter
  - LevelCompactionDynamicLevelBytes=true
  - MaxBackgroundJobs=4
  - MaxBytesForLevelBase=1 GByte
  - MaxOpenFiles=-1
  - MaxWriteBufferNumber=4
  - MinWriteBufferNumberToMerge=3
  - TargetFileSizeBase=256 MBytes
  - WriteBufferSize=64 MBytes
- FLASH_SSD_OPTIMIZED: option for flash SSDs configured for better performance. The following additional options are set:
  - MaxBackgroundJobs=4
  - MaxOpenFiles=-1

You can set the Rocksdb predefined options in the following ways:

For Cloudera Manager

1. Open your cluster in Cloudera Manager.
2. Select Flink from the list of services.
3. Click Configuration.

4. Search for state.backend.rocksdb.predefined-options and change the default value as needed.

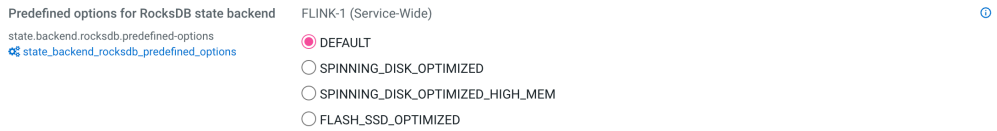

5. Click Save changes.

For SQL Editor

You can configure the state.backend.rocksdb.predefined-options on Streaming SQL Console with executing a SET statement when submitting your SQL job in a per-job mode as shown in the following example:

```
SET 'execution.target' = 'yarn-per-job';
SET 'state.backend.rocksdb.predefined-options' = 'SPINNING_DISK
_OPTIMIZED';
SET 'state.backend.rocksdb.predefined-options' = 'SPINNING_DISK
_OPTIMIZED_HIGH_MEM';
SET 'state.backend.rocksdb.predefined-options' = 'FLASH_SSD_OP
TIMIZED';
```

Managing session for SQL jobs

By default, the SQL Stream jobs are running in a session cluster. This means that multiple Flink jobs run in the same YARN session sharing the cluster, allocated resources, the Job Manager and Task Managers. The session starts when you open the Streaming SQL Console. You can reset the session, and set the properties of the session using the Streaming SQL Console.

Resetting a session

When you reset your session, every configuration, temporary table and view, default database and catalog will be lost.

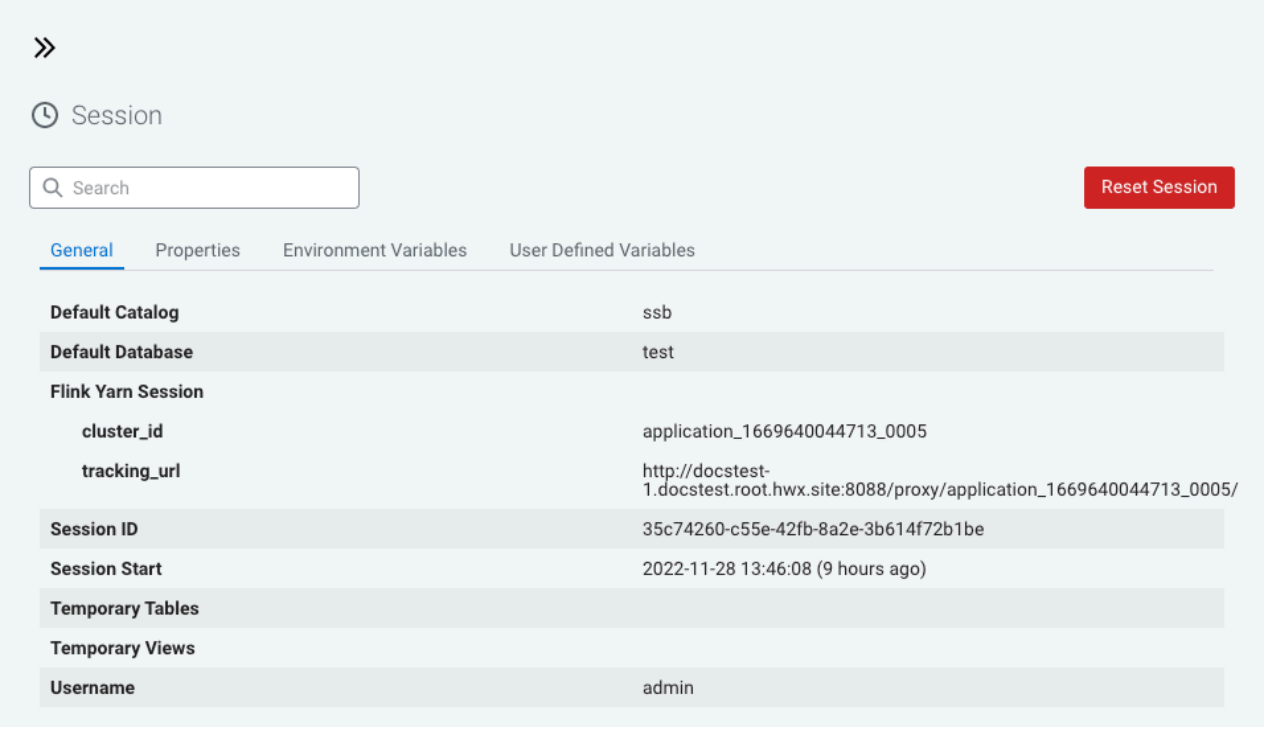

Configuring properties for a session

You can configure the session properties using the SET statement in the SQL window.

1. Use the SET statement in the **SQL Editor** to configure a session property.

   Example:

   ```
   SET state.backend=rocksdb
   ```

   You can review all of the configurable parameters on the Properties tab of the **Session** window.

   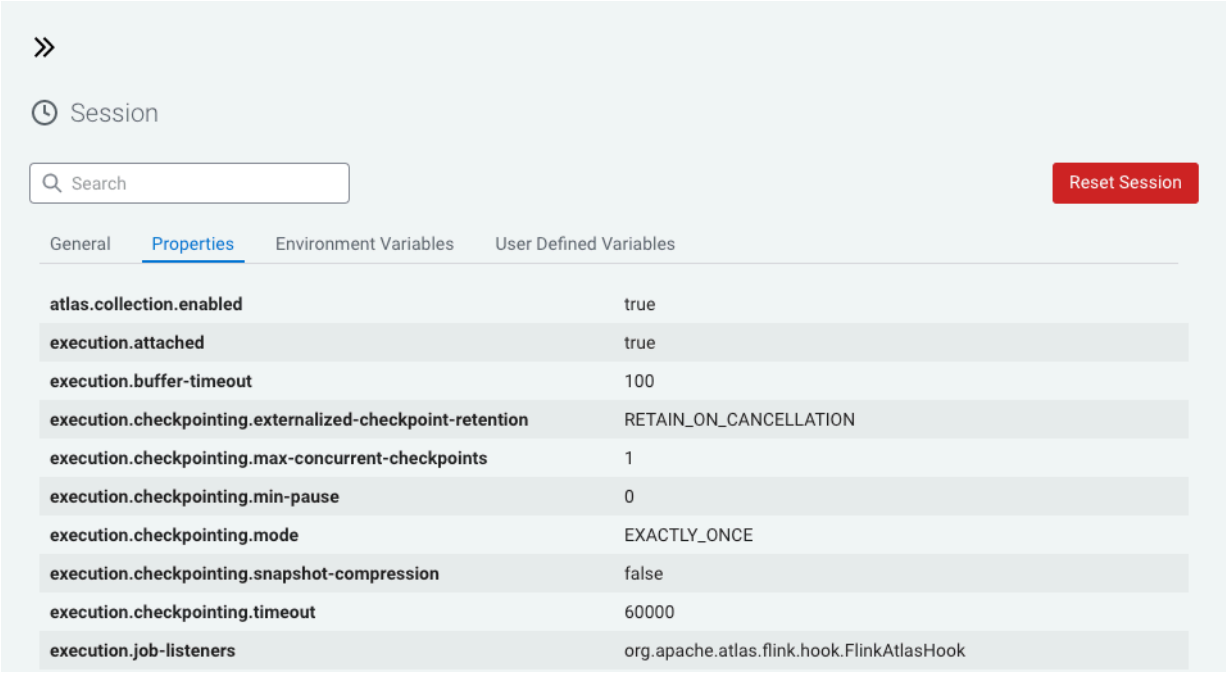

2. Click Execute.

Executing SQL jobs in production mode

As by default the SQL jobs are running in a session cluster, there is a risk in case of a cluster failure that every job is affected within that cluster. However, you can set a per-job production mode in SQL Stream Builder to create a dedicated environment for your production jobs.

Production mode means that the deployed SQL job (Flink job) runs in a separate SQL session and on a dedicated YARN cluster configured specifically for that particular job. You can set the execution mode in the **Job Settings** window, although the YARN execution target can be overwritten using a SET statement in the SQL Editor.

Setting production mode in Job Settings

To set the production mode for your SQL jobs in the **Jobs Setting** page, you need to select the execution mode with the following steps:

1. Click Job Settings at the **SQL Editor**.
2. Select PROD - Job runs in isolated Cluster at Execution Mode.
3. Add a SQL statement to the SQL Editor you want to execute.
4. Click Execute.

Setting YARN execution target in SQL Editor

To set the YARN execution mode for your SQL jobs in the **SQL Editor**, you need to use the SET statement, and execute the SQL query after defining the execution target and properties in the same window:

```
set 'execution.target' = 'yarn-per-job';
set 'logging.configuration.file' = '/tmp/log4j.properties';
select * from datagen_table_1631781644;
```

In the above example, the execution target is set to per-job to create a new YARN application for the job. Setting the execution target to per-job allows you to have an individual cluster for the specific job. The additional properties that you configure using the SET statement overwrites the properties that are configured for the running session. However, when you set properties for the production mode, the settings of the session cluster are not affected.

Note: You can override the execution mode set in the **Job Settings** window using the SET statement in the **SQL Editor**.

1. Set the execution mode to per-job.

   ```
   set 'execution.target' = 'yarn-per-job';
   ```

2. Add additional configuration to the production job.
3. Add a SQL statement you want to execute.

   Example:

   ```
   set 'execution.target' = 'yarn-per-job';
   set 'state.backend' = 'rocksdb';
   select * from faker_table_1631781644;
   ```

4. Click Execute.

Functions

Creating Python User-defined Functions

With Cloudera SQL Stream Builder, you can create powerful custom functions in Python to enhance the functionality of SQL.

About this task

User functions can be simple translation functions like Celsius to Fahrenheit, more complex business logic, or even looking up data from external sources. User functions are written in Python. When you write them, you can create a library of useful functions.

Important: Support for JavaScript UDFs was deprecated and will be removed in a future release. Python UDFs will be used as a replacement, and users should begin migrating their existing UDFs to Python.

Before you begin

Python UDFs are disabled by default. To enable them:

1. Python UDF execution requires the Python Apache Flink and dependent packages on a supported Python version installed **on all nodes**.
   a. Access your Cloudera SQL Stream Builder node's command line. (Execute the following commands as root.)
   b. Install the Python Apache Flink and dependent packages:

```
/usr/local/bin/python[***VERSION***] -m pip install apache-flink==1.19.1
```

2. Go to your cluster in Cloudera Manager.
3. Select Cloudera SQL Stream Builder from the list of services.
4. Go to the Configuration tab and set the following configuration properties:

   Tip: You can type **python** in the filter field to narrow down the configuration properties.

   a. Python Client Executable (ssb.python.client.executable): the path of the Python interpreter used to launch the Python process when submitting the Python jobs via flink run or compiling the jobs containing Python UDFs. For example /usr/bin/python3
   b. Python Executable (ssb.python.executable): the path of the Python interpreter used to execute the python UDF worker. For example /usr/bin/python3
   c. Python UDF Reaper Period (ssb.python.udf.reaper.period.seconds): the interval (in seconds) between two Python UDF Reaper runs, which deletes the Python files of the terminated jobs from the artifact storage.
   d. Python UDF Time To Live (ssb.python.udf.ttl): The minimum lifespan (in milliseconds, seconds, minutes, or hours) of a Python UDF in the artifact storage. After this the Python UDF Reaper can delete the Python files from the artifact storage.
   e. Select Enable Python UDFs in SSB and click on the checkbox.
5. Enter a reason for change and click Save Changes.
6. Navigate to Instances.
7. Select the Streaming SQL Engine from the list. (Click the checkbox on the left.)
8. Click Restart from the Actions for selected dropdown menu.

Important: The following Python versions are supported in Cloudera SQL Stream Builder:

- 3.8
- 3.9
- 3.10

To use Python UDFs:

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click ⋮ next to **Functions**.

4. Click New Function.

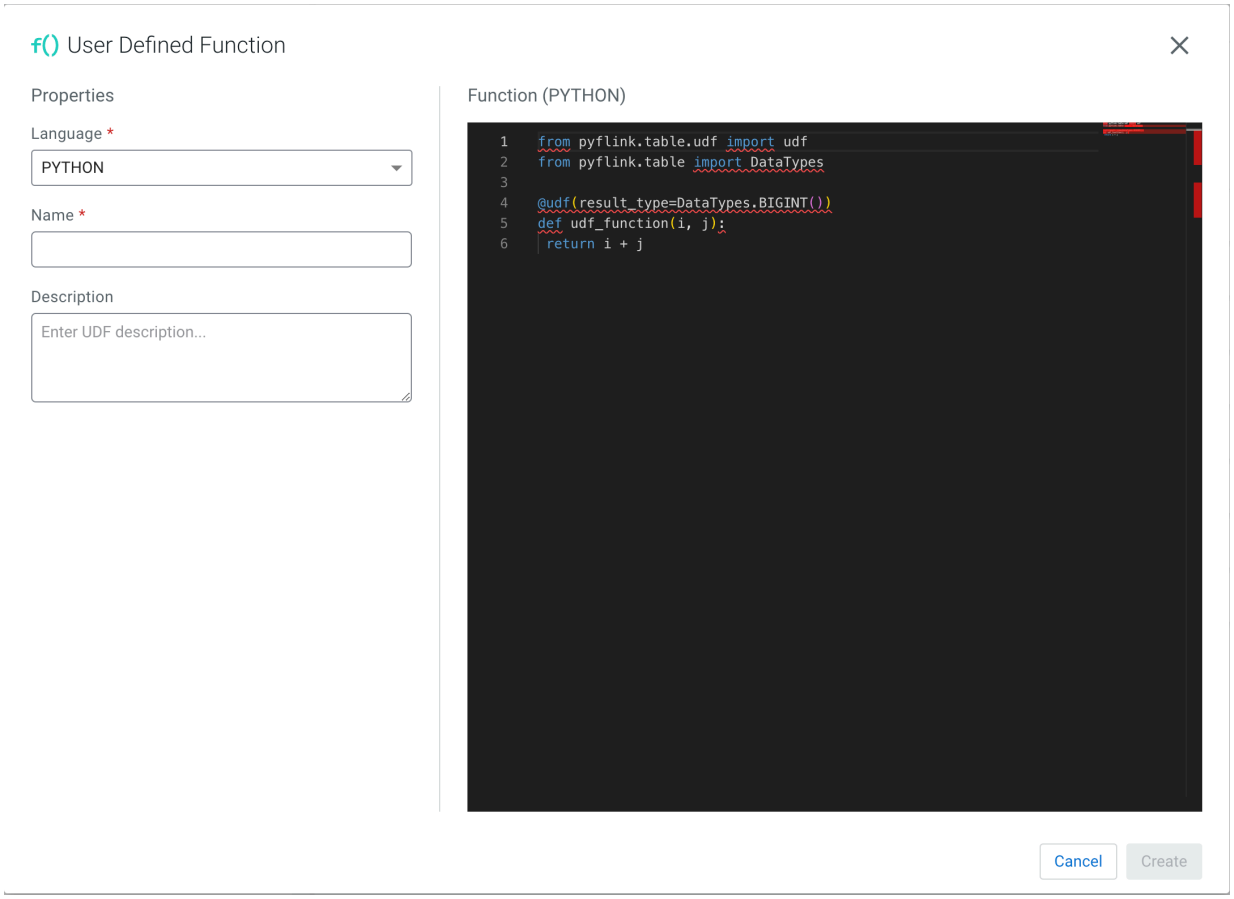

5. From the dropdown menu select PYTHON.
6. Add a Name to the UDF.
   For example, name the UDF to ADD_FUNCTION.
7. Add a Description to the UDF. (Optional.)
8. Paste the Python code to the editor.
   For example, a simple addition:

```
from pyflink.table.udf import udf
from pyflink.table import DataTypes

@udf(result_type=DataTypes.BIGINT())
def udf_function(i, j):
 return i + j
```

Important: The name of the function always has to be udf_function.

9. Click Create.
10. Once created, you can use the new User Defined Function in your SQL statement or as a computed column when creating a table.
    For example:

```
-- as a SQL statement
SELECT ADD_FUNCTION(myTable.n1, myTable.n2)

-- as a computed column
CREATE TABLE myTable (
```

```
  `number_a` BIGINT,
  `number_b` BIGINT
  `sum` AS ADD_FUNCTION(number_a, number_b)  -- evaluate expression and s
upply the result to queries
) WITH (
  'connector' = 'kafka'
  ...
);
```

Results

When a Flink job starts, Cloudera SQL Stream Builder will upload the Python file to the Artifact Storage, accessible for Flink to execute the UDF when called. The ssb.python.udf.reaper.period.seconds and ssb.python.udf.ttl configuration properties (set in Cloudera Manager) control the behavior of SSB to remove Python files associated with terminated jobs.

For more information on using Python UDFs, refer to the Apache Flink documentation.

Creating Javascript User-defined Functions

With Cloudera SQL Stream Builder, you can create user functions to write powerful functions in JavaScript that you can use to enhance the functionality of SQL.

About this task

User functions can be simple translation functions like Celsius to Fahrenheit, more complex business logic, or even looking up data from external sources. User functions are written in JavaScript. When you write them, you create a library of useful functions.

Important: Support for JavaScript UDFs was deprecated and will be removed in a future release. Python UDFs will be used as a replacement, and users should begin migrating their existing UDFs to Python.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click ⋮ next to **Functions**.

4. Click New Function.

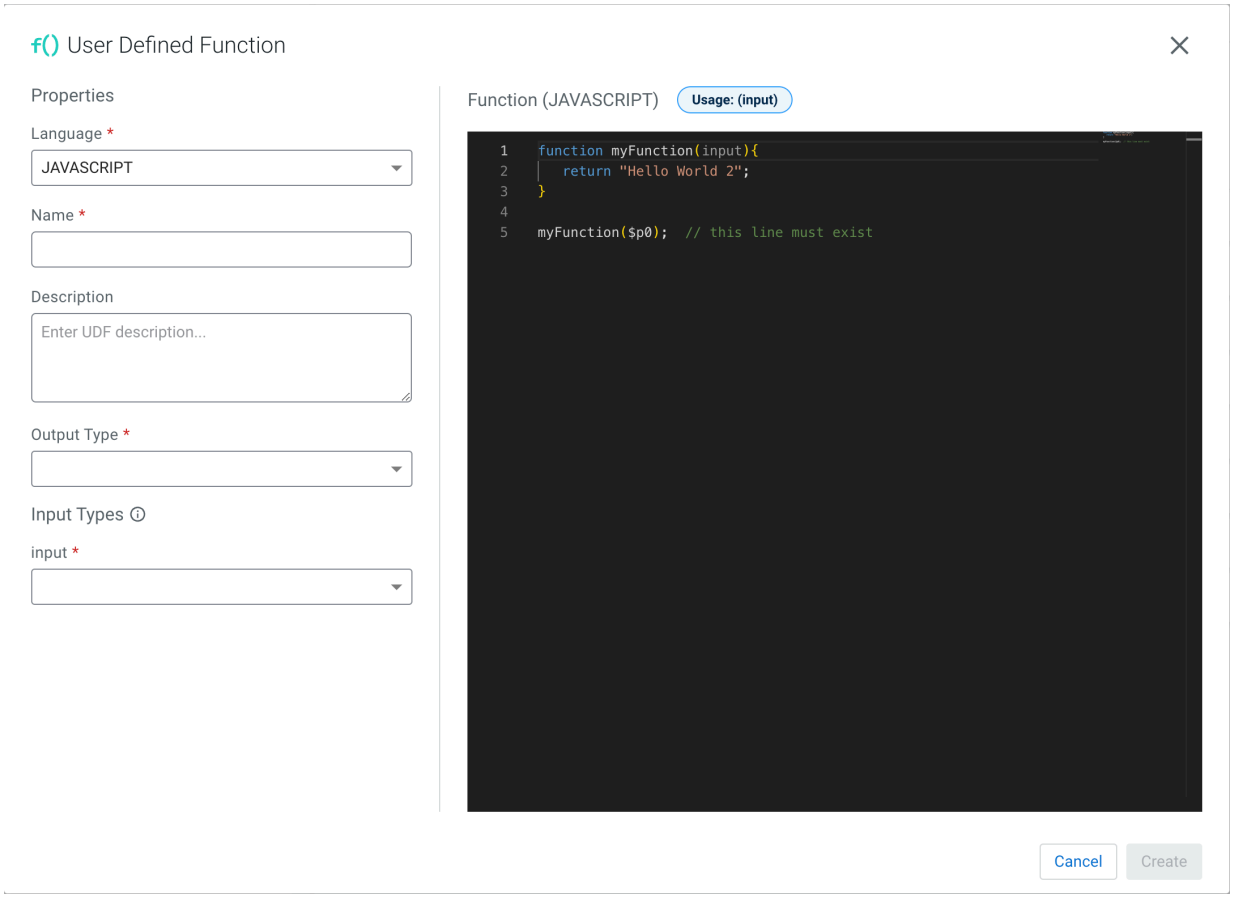

5. From the dropdown menu select JavaScript.
6. Add a Name to the UDF.

   For example, name the UDF to HELLO_WORLD.
7. Add a Description to the UDF.
8. Select the Output and Input data type.

   For example, select STRING.
9. Paste the JavaScript code to the editor.

   For example:

```
// check to see if the card is VISA
function HELLO_WORLD(card){
    var cardType = "Other";
    if (card.charAt(0) == 4){
      cardType = "Visa";
    }
    return cardType;
}
HELLO_WORLD($p0); // this line must exist
```

10. Click Create.
11. Once created, you can use a User Defined Function in your SQL statement:

    For example:

```
-- simple usage
SELECT HELLO_WORLD(card) AS IS_VISA
FROM ev_sample_fraud;
```

```
-- in the predicate
SELECT amount, card
FROM ev_sample_fraud
WHERE HELLO_WORLD(card) = "Visa";
```

Note: Valid inputs can be a field in the source virtual table or any other valid input. Functions must be in upper case.

Note: User Functions have access to the Java 8 API, this increases the overall usefulness and power. For example:

```
function GETPLANE(icao) {
  try {
    var c = new java.net.URL('http://yyyyyy.io' + icao).openConnec
tion();
    c.requestMethod='GET';
    var reader = new java.io.BufferedReader(new java.io.InputStreamRe
ader(c.inputStream));
    return reader.readLine();
  } catch(err) {
    return "Unknown: " + err;
  }
}
GETPLANE($p0);
```

Developing JavaScript functions

When developing JavaScript functions that are more complicated than just simple logic, it is recommended to use the jjs command-line utility to create and iterate while writing functions.

About this task

After the function performs the required task, migrate it to the console. Additionally these files/functions can be saved in a source code control system like git/Github.

Procedure

1. Create a file for your function.

   Note: It is recommended to name the file with the same name as that of the function.

2. Create some sample input when calling the function.
3. Call jjs on the command line to test the function.

```
$>cat TO_EPOCH.js
function TO_EPOCH(strDate) {
   var strFmt = "yyyy-MM-dd HH:ss:mm";
   var c = new java.text.SimpleDateFormat(strFmt).parse(strDate).getTime()
/1000;
   return c.toString();
}

print(TO_EPOCH("2019-02-02 22:23:13"));

then
$>jjs TO_EPOCH.js
1549167203
```

What to do next

After you have successfully developed the JavaScript code, copy and paste only the function to your code window when creating the JavaScript function in Cloudera SQL Stream Builder.

Creating Java User-defined functions

You can create User Defined Functions (UDF) using Java after manually adding the UDF function JAR file that contains the UDF class to the Flink connectors. After uploading the JAR file on the Streaming SQL Console, you can use the Java UDFs for your SQL jobs.

Procedure

1. Create a Java UDF function JAR file.

   You can use the following example as a sample:

```
package udf;
import org.apache.flink.table.functions.ScalarFunction;

public class UdfTest extends ScalarFunction {
   public String eval(String input){
       return "Hello World " + input;
   }
}
```

2. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.
3. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
4. Click ⋮ next to **Functions**.
5. Click Upload JAR.
6. Click Choose File.
7. Search and select the Java UDF function JAR file.
8. Click Create.

Results

The selected JAR file is uploaded, and listed under **Functions** and ready to be used in your SQL queries.

Using System Functions

The same set of system functions can be used for Cloudera SQL Stream Builder as for Apache Flink.

As SSB runs on Flink, the following built-in system functions can also be used for data transformations in your SQL Jobs.

- Comparison Functions
- Logical Functions
- Arithmetic Functions
- String Functions
- Temporal Functions
- Conditional Functions
- Type Conversion Functions
- Collection Functions
- JSON Functions
- Value Construction Functions
- Value Access Functions
- Grouping Functions
- Hash Functions
- Aggregate Functions
- Column Functions

For more information about the list of supported Functions, see the Apache Flink documentation.

Introduction to Materialized Views

Cloudera SQL Stream Builder has the capability to materialize results from a streaming SQL query to a persistent view of the data that can be read using REST API/endpoints. Business Intelligence tools and applications can use the Materialized View REST endpoint to query streams of data without deploying database systems. In addition to REST, Materialized Views can also be queried using JDBC/ PostgreSQL wire protocol.

Materialized Views are in synchronization with the mutating stream - they are updated by a primary key as data flows through the system. The data is updated by a given key, and it represents the latest view of the data by key. The following illustration shows the process of a Materialized View query in streaming SQL.

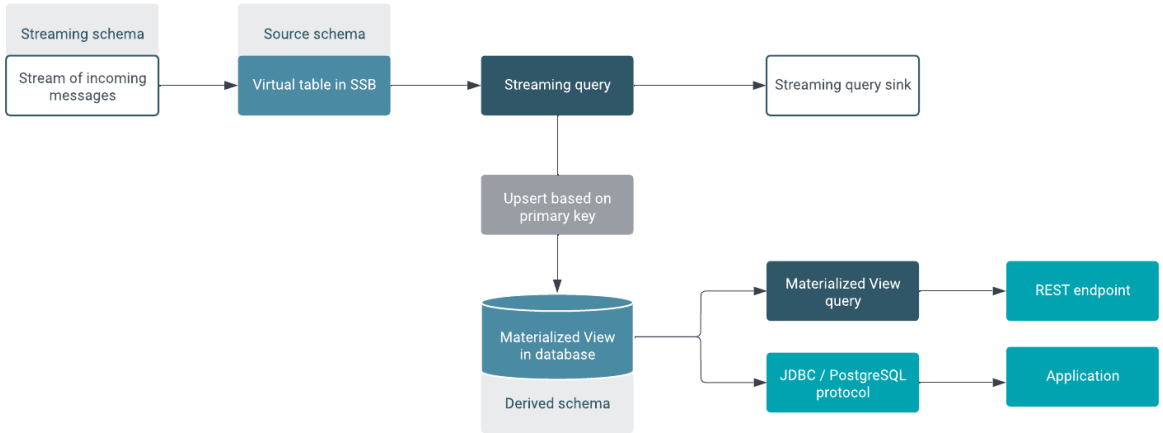

The following schemas are defined in the illustration:

- Streaming schema: refers to what Cloudera SQL Stream Builder knows from the source provider, for example Kafka
- Source schema: refers to the schema of the table defined by the catalog in Cloudera SQL Stream Builder
- Derived schema: refers to the schema of the Materialized View that is created in the database as a result of the streaming query against the catalog table

For example: vehicleID Z latest latitude and longitude is X and Y. As the vehicle moves, the latitude and longitude for the vehicleID are updated . The primary key is defined at creation time and is immutable.

Materialized Views can be created as mutating snapshots of the queried data result that is updated by a given key. The data is always the latest representation of itself by key (analogous to a primary key in most RDBMS systems).

You can query the Materialized Views using a GET request over REST, which returns a JSON response as "Content-Type: application/json". The queries are not defined at query time. Rather, they are curated, saved, and granted access through the Cloudera platform. You can configure a REST endpoint to query the Materialized View. Multiple query conditions can be created to allow various ways to query the same data. This is sometimes referred to as a 'pull query'.

Null Keys

In situations where information about a specific item is updated incrementally, for example, data about something is received in multiple messages, there may be keys missing from the incoming messages (for example, the streaming schema). In this case Cloudera SQL Stream Builder continues to consume these messages, but marks the missing key as NULL at the sink, therefore Cloudera SQL Stream Builder will also update the Materialized View with an upsert including the NULL values. To change this behavior, the “ignore null” option can be used, which tells Cloudera SQL Stream Builder to update the relevant record only with the non-null values of the incoming message. Similarly, when a key is removed from the source schema, but not from the streaming schema, Cloudera SQL Stream Builder ignores the key on the stream of incoming messages.

Creating Materialized Views

After executing a SQL Stream job, you can set up the Materialized Views to have a snapshot of your queried data. You can use the URL Pattern from the Materialized View to visualize the generated data.

Before you begin

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click ⋮ next to **Jobs** from the **Explorer**.
4. Select New Job .

   The SQL Editor for the created job opens in a tab.
5. Add a SELECT statement to the SQL Editor.

6. Click Materialized View.
   The configuration window for the Materialized View opens.

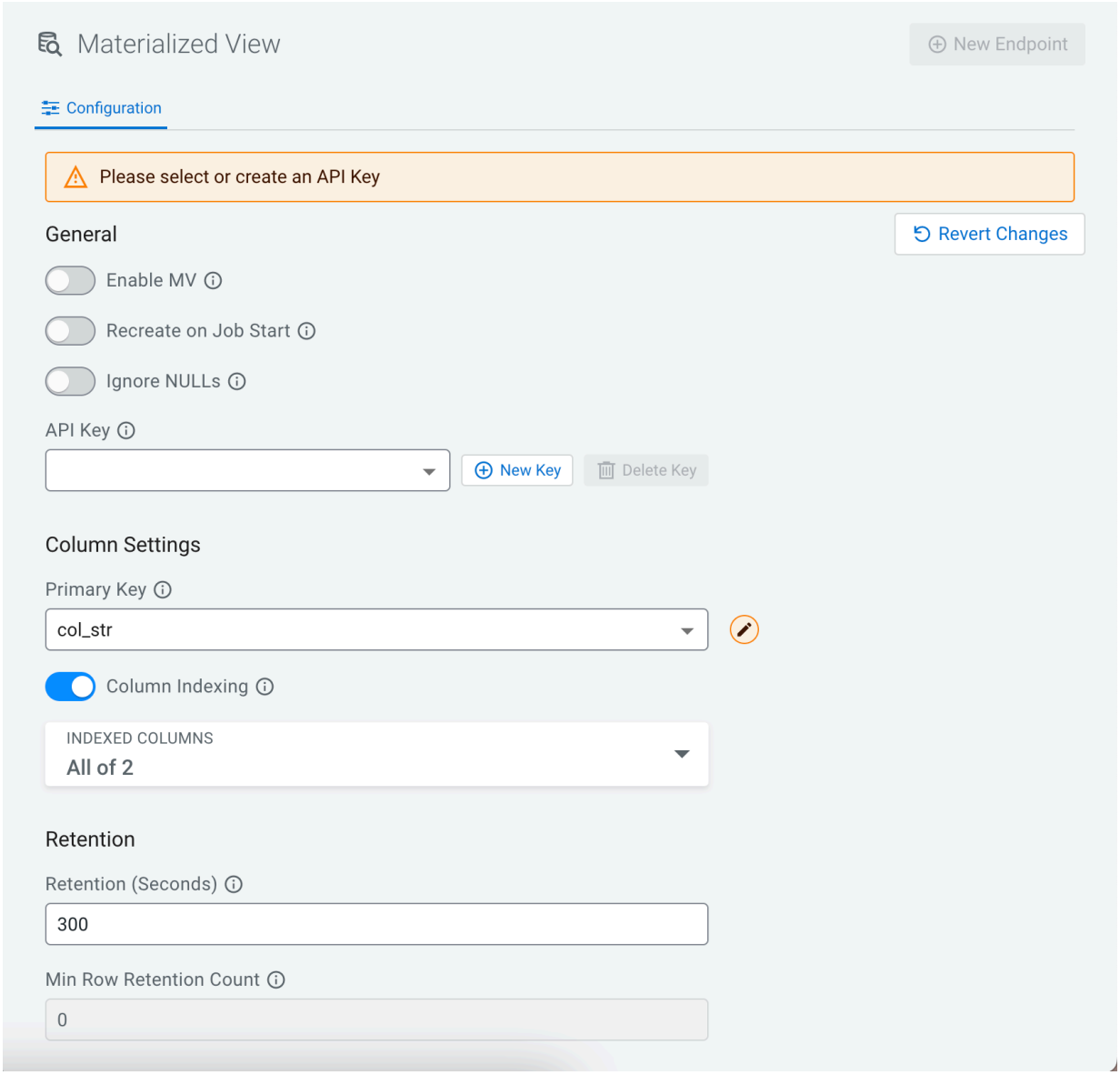

7. Switch Enable MV toggle.
8. Enable or disable Recreate on Job Start.
   If enabled, the Materialized View is deleted when a job is started or restarted.
9. Enable or disable Ignore NULLS.
   If enabled, NULL values will NOT update values that are non null - they are ignored.

10. Select an API Key.

In case there are no API Keys, click Add API key, or click Materialized Views from the main menu. The add API key window appears. Provide a name for the API key, and click Save Changes.

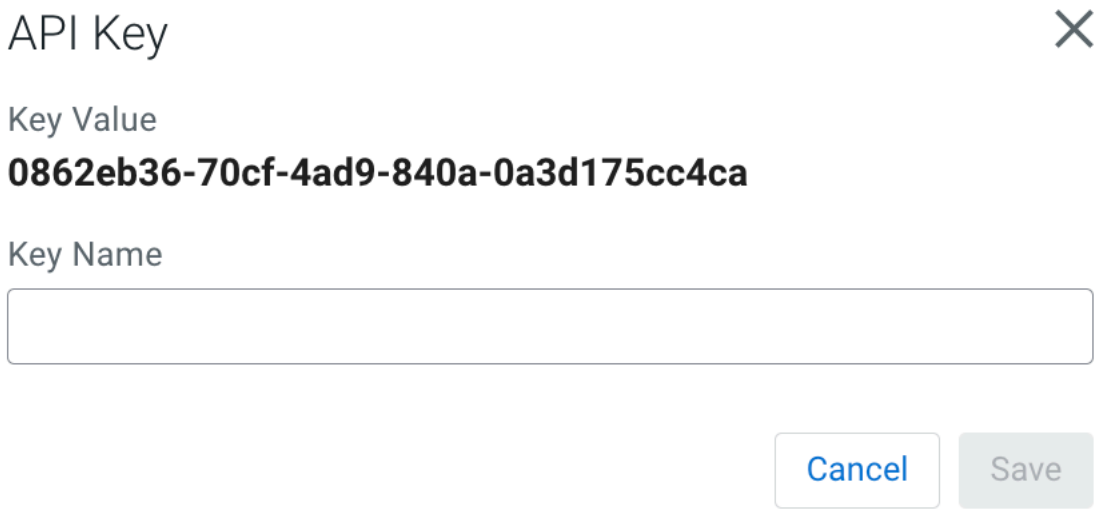

To check your created API keys:

a.

Click Manage at the **API Keys** from the Project Explorer, or click on the created API key under the **API Keys**.

11. Configure the **Column Settings** for the Materialized View.

a) Select a Primary Key.

If this list is empty, then no SQL is specified in the **SQL Editor** or the SQL query is invalid. Select a key as a primary key for the Materialized View. All data will be updated by this key.

b) Enable Column Indexing if needed, and select the column by which you need the results to be indexed.

12. Configure the retention time for the Materialized View.

For more information about how you can set the retention time, see the Configuring Retention Time for Materialized View section.

13. Click New Endpoint to create the Materialized View query.
    The **Materialized View Endpoint** window appears.

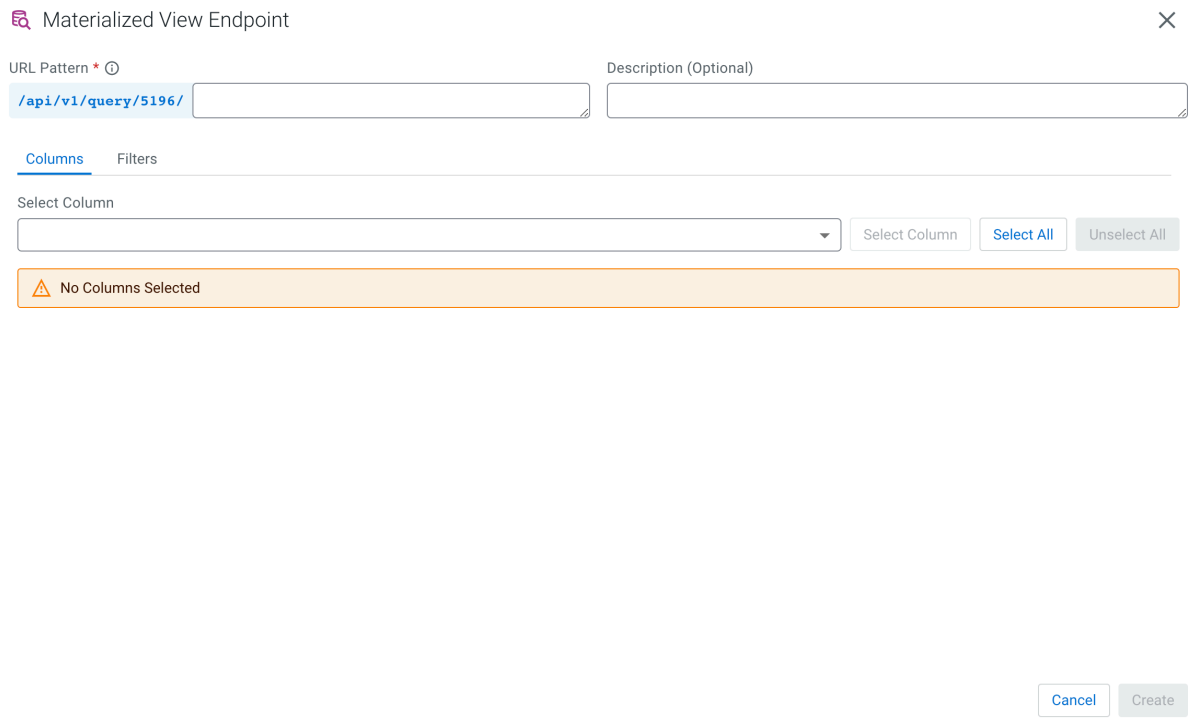

14. Provide a name in the **URL Pattern** field.
15. Provide a description of the Materialized View Query, if needed.

16. Customize the Materialized View in the **Endpoint Editor**.

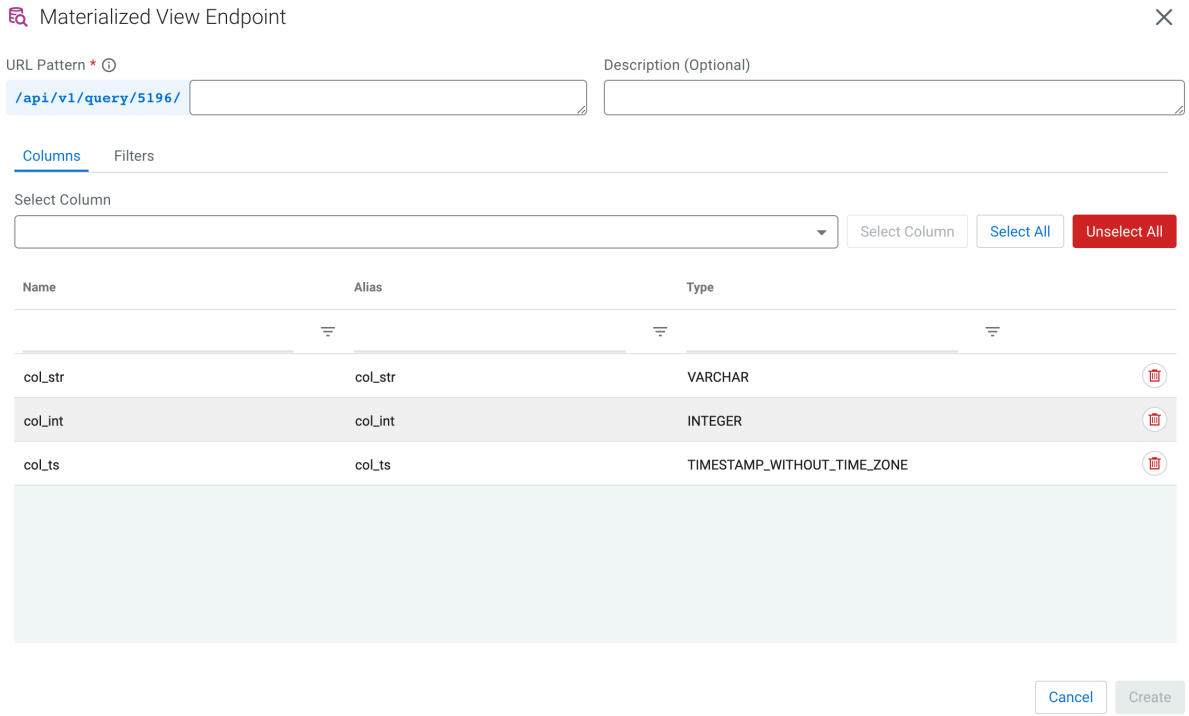

a) Select the columns of the SQL job you want to use in the Materialized View Query.
You can select every column in the table by clicking on the Select All button.

b) Click Filters tab to apply computations and further enrichment of your data.

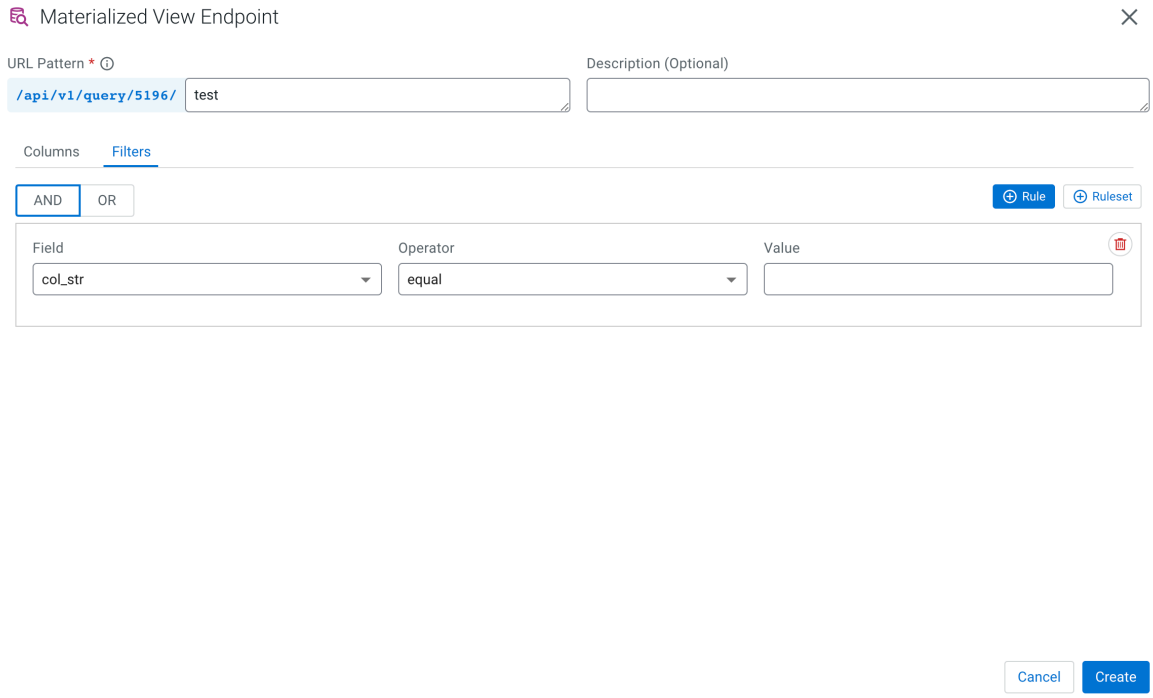

c) Provide details of the filter by selecting the Field, Operator and Value.
You can add more rules or set a ruleset for the query.

17. Click Create.

You can access the queried results by clicking ⋮ Manage at the **Materialized Views** from the Project Explorer, and by opening the URL pattern.

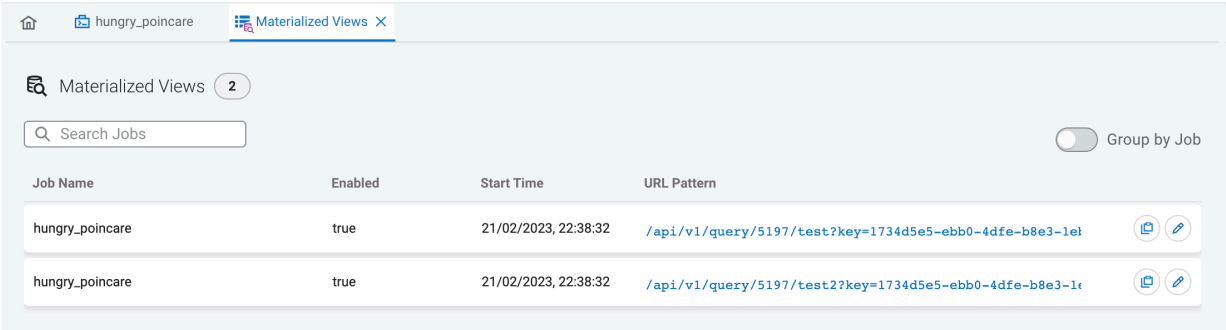

Results

You can click on the created REST endpoint to review the data, or copy it and visualize the queried data in a Business Intelligence tool, notebook, code and so on. You can also review the list of Materialized View and API keys on the Materialized View page.

Configuring Retention Time for Materialized Views

When creating Materialized Views, you can configure how the system should retain the data rows in the Materialized Views. You can either choose between retaining the data by time or the row count.

The Materialized Views configuration allows you to set one of the following configuration parameters for data retention:

- Retention Time
- Min Row Retention Count

You can specify the Retention Time and Min Row Retention Count when creating a Materialized View for a SQL job on the Compose page of the Streaming SQL Console.

Retention Time

Retention Time is specified in seconds, and it tells the system to retain data rows as old as the specified retention time. The rows that are outside of the retention time are removed from the Materialized View.

Note: You can only add a Retention Time value, if the Min Row Retention Count field is empty or set to 0.

The following example shows how to set a Retention Time of 300 seconds. This means that only those rows are included in the Materialized View that are within the 300 seconds of the job execution. The older rows are removed from the Materialized Views.

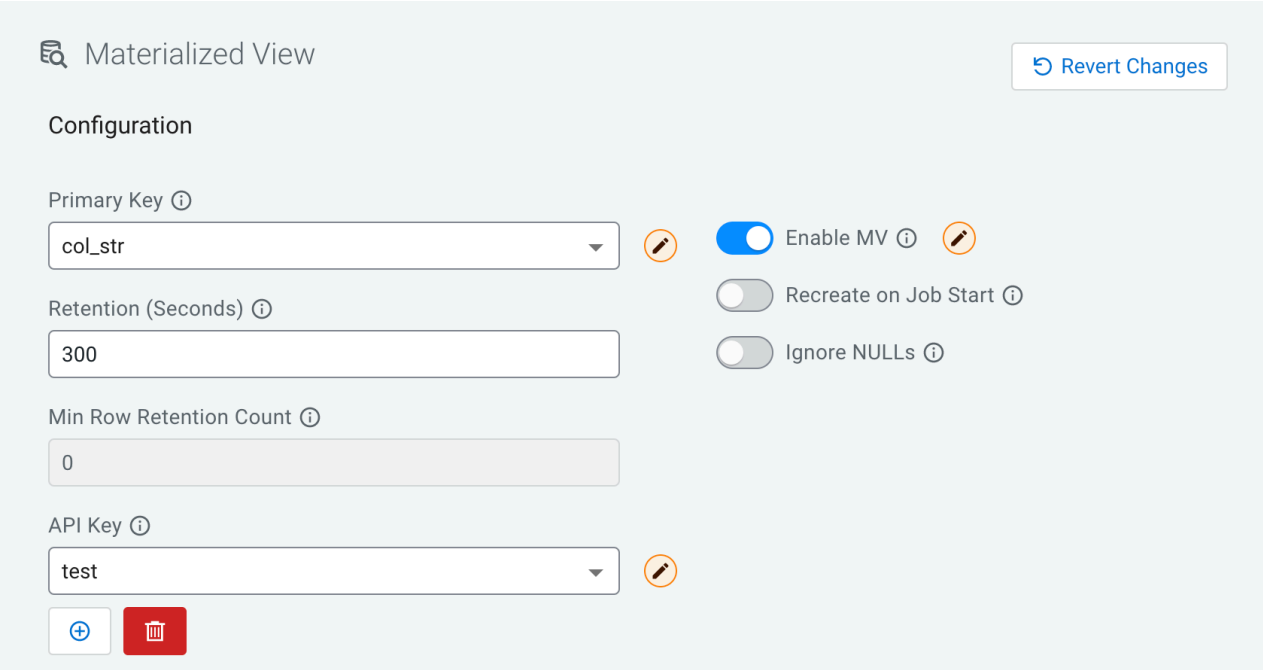

Minimum Row Retention Count

The Min Row Retention Count parameter indicated to the system to maintain a specific number of data rows in the Materialized View.

Note: You can only add a Min Row Retention Count value, if the Retention Time field is empty or set to 0.

The following example shows how to configure the system to retain the last 5000 data rows. This means that only the first 5000 data rows are included in the Materialized View, and the data rows from 5001 are removed from the Materialized View.

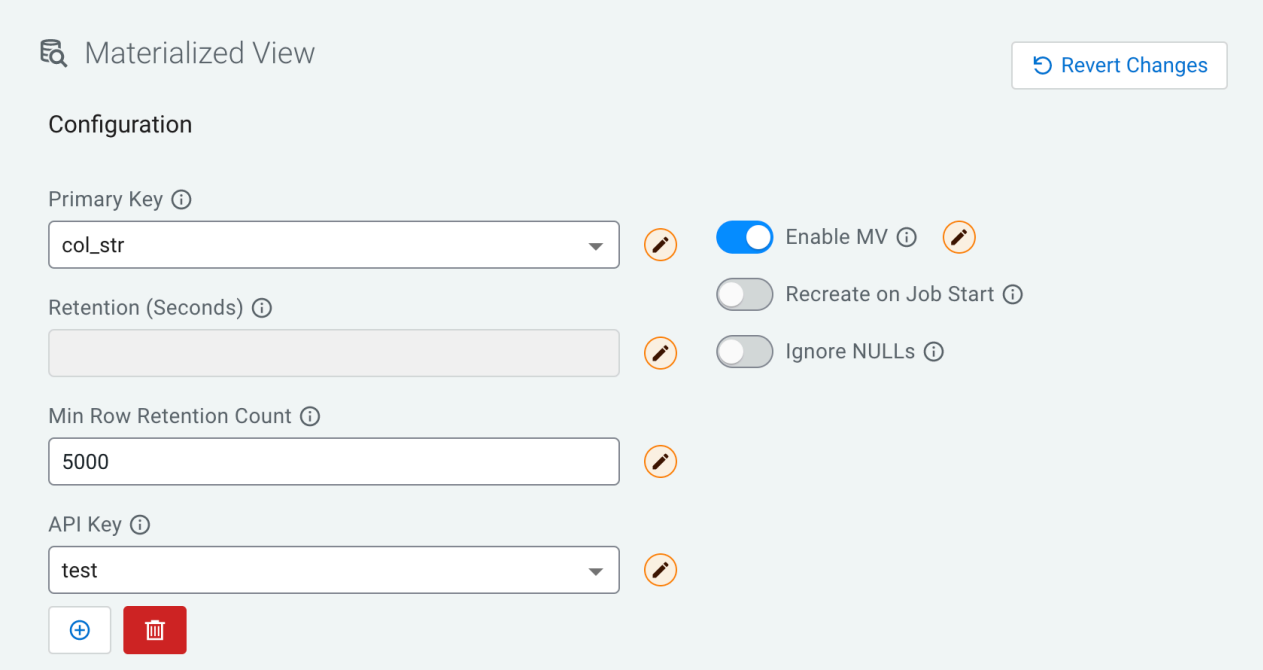

Retaining all data without limit

In order to retain all data rows, regardless of time, or number of rows in the Materialized View, both settings can be reset to zero (0). This indicates to the system that all data must be preserved without a time limit.

The following example shows how to configure the Retention parameters to keep all of the data regardless of time:

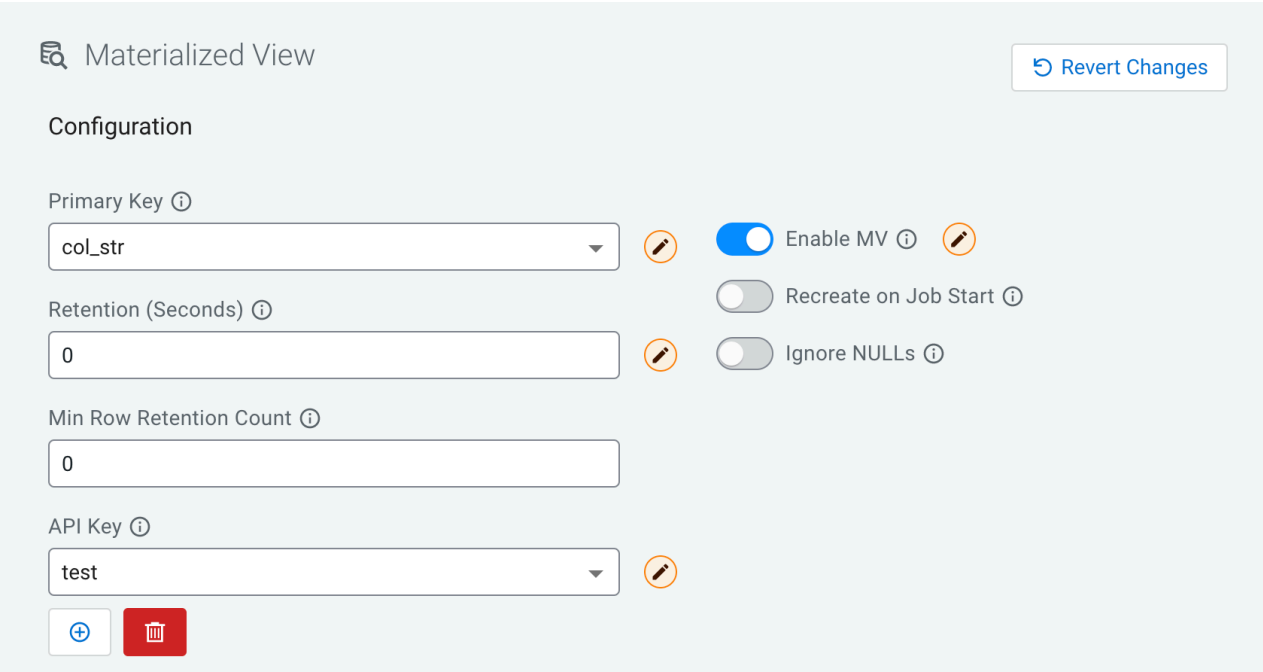

Materialized View Pagination

You can set a limit and order the results of a Materialized View query by adding values to the limit and offset configurations.

After creating a Materialized View, you can order and limit the results after opening the Materialized View. The default limit of the results is 100 entries. You can set the limit and offset using the REST URL. You need to modify the URL address in your browser by setting the value and the offset value as shown in the following example:

```
localhost:18131/api/v1/query/5275/summary_query?key=6ef33a7a-c82a-4a72-946b-
5fe0b33e033c&limit=20&offset=3
```

When you edit the Materialized View, the set limit and offset change based on the new filters set for the Materialized View query. If you set an invalid number for the limit and offset, an error message is displayed.

Using Dynamic Materialized View Endpoints

You can use static or dynamic REST endpoints when creating Materialized Views in Cloudera SQL Stream Builder. After setting filters for the Materialized View query, you can further filter down the results by using variables in the endpoint URL.

Difference between static and dynamic endpoints

When using a static endpoint, the endpoint URL is a constant string. The logic of filtering is defined statically at the endpoint configuration and the results of the endpoint call will always reflect this. A static endpoint is one that only uses filters with static values. For example, in the following static endpoint the Filter is set to age greater than 0, and the endpoint URL is always /api/v1/query/5200/*AGE*:

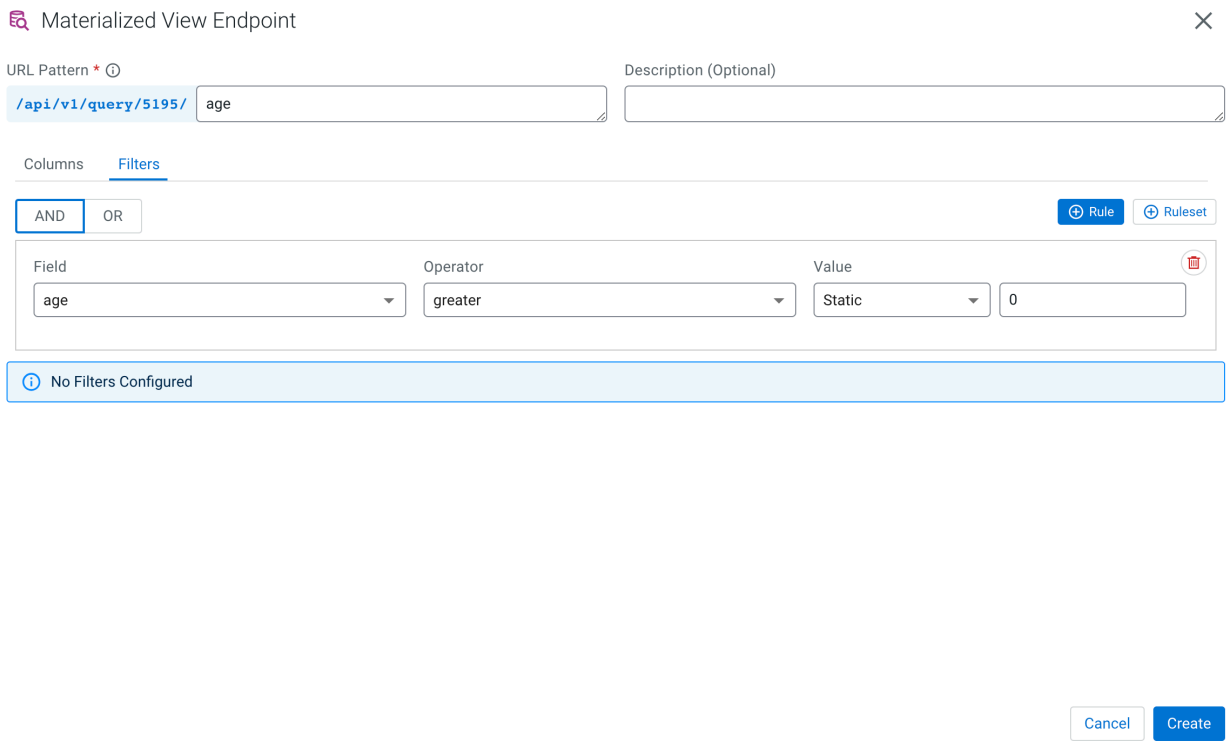

Dynamic materialized view endpoints use filters that reference parameters instead of static values. The value of these parameters are specified when the endpoints are used.

To create a dynamic filter, after selecting the Value type as Dynamic, click on the button to add a new parameter, or select the parameter from the drop-down list, if it had already been created previously. When creating a new parameter, you will be prompted to provide the name of the Dynamic Parameter.

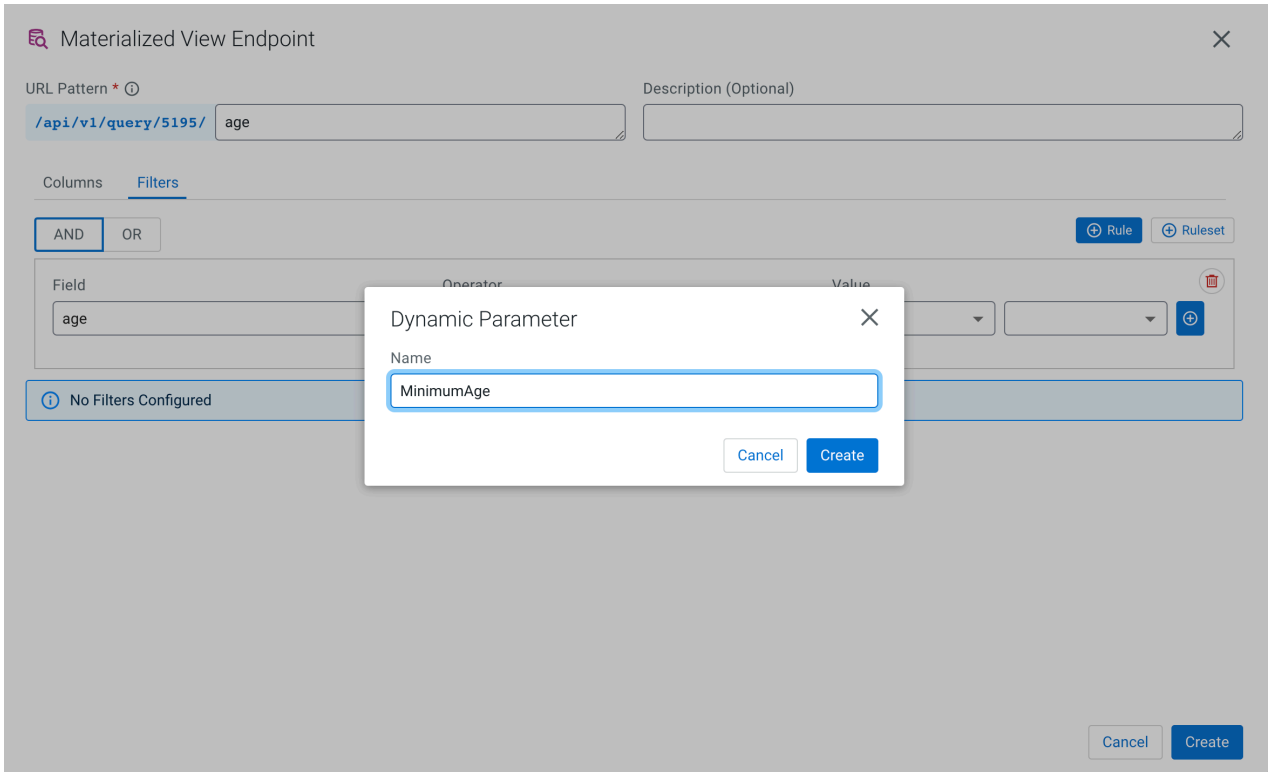

After specifying the dynamic parameter, click Create. Once the job is running, you can test the Materialized View endpoint by clicking on the endpoint URL in the endpoint page:

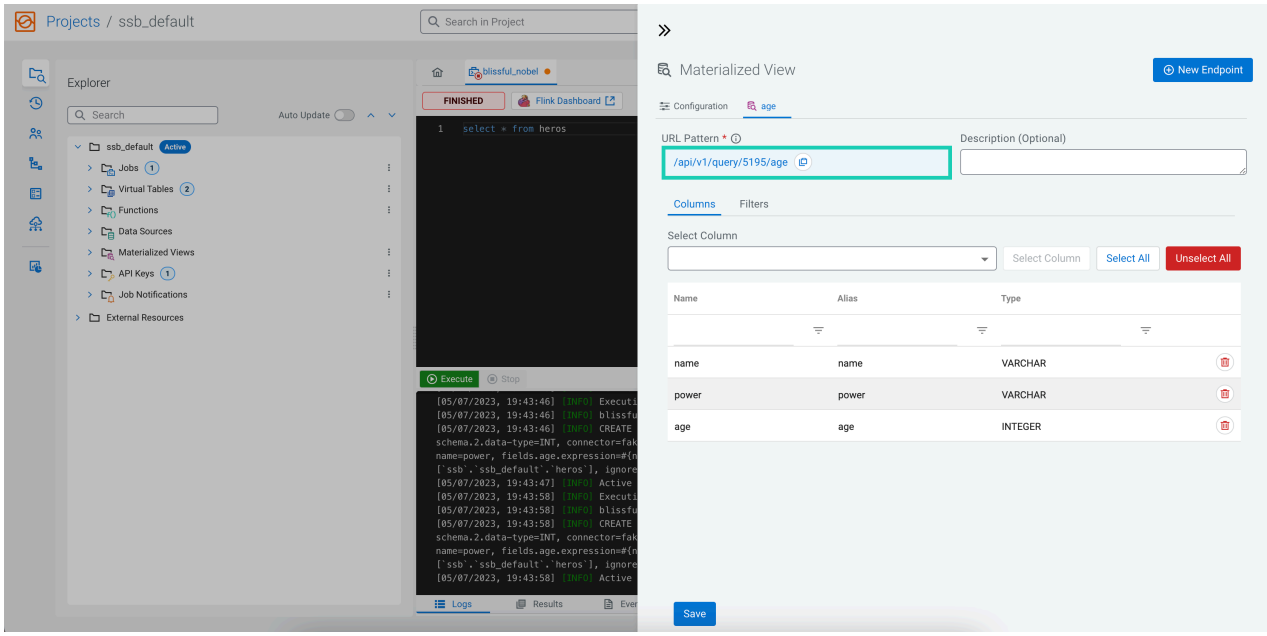

For dynamic endpoints, you will be prompted for the values of the dynamic parameters when opening the REST endpoint of the Materialized View.

The dynamic parameters are defined as URL query parameters as shown below:

```
<ENDPOINT_URL>?key=<API_KEY>&limit=<MAX_RECORDS>&param1=<VALUE1>&par
am2=<VALUE2>...
```

The advantage of a dynamic endpoint is that by providing dynamic parameters and a corresponding value, you are able to filter the queried results at runtime, returning only the values you exactly need from that dataset.

You can use multiple dynamic parameters in the endpoint and you can also use the same dynamic parameter at the different filters to create more specific filtering methods for the results of the Materialized View query. When removing a filter from a Materialized View, the Dynamic parameter is not affected, which means that if a filter is deleted from the Materialized View query, the value for Dynamic Parameter still needs to be provided when opening the REST endpoint. The Dynamic Parameter needs to be deleted separately from the filter by clicking on the Name of the Dynamic Parameter, and selecting the delete option from the drop-down menu as shown in the following example:

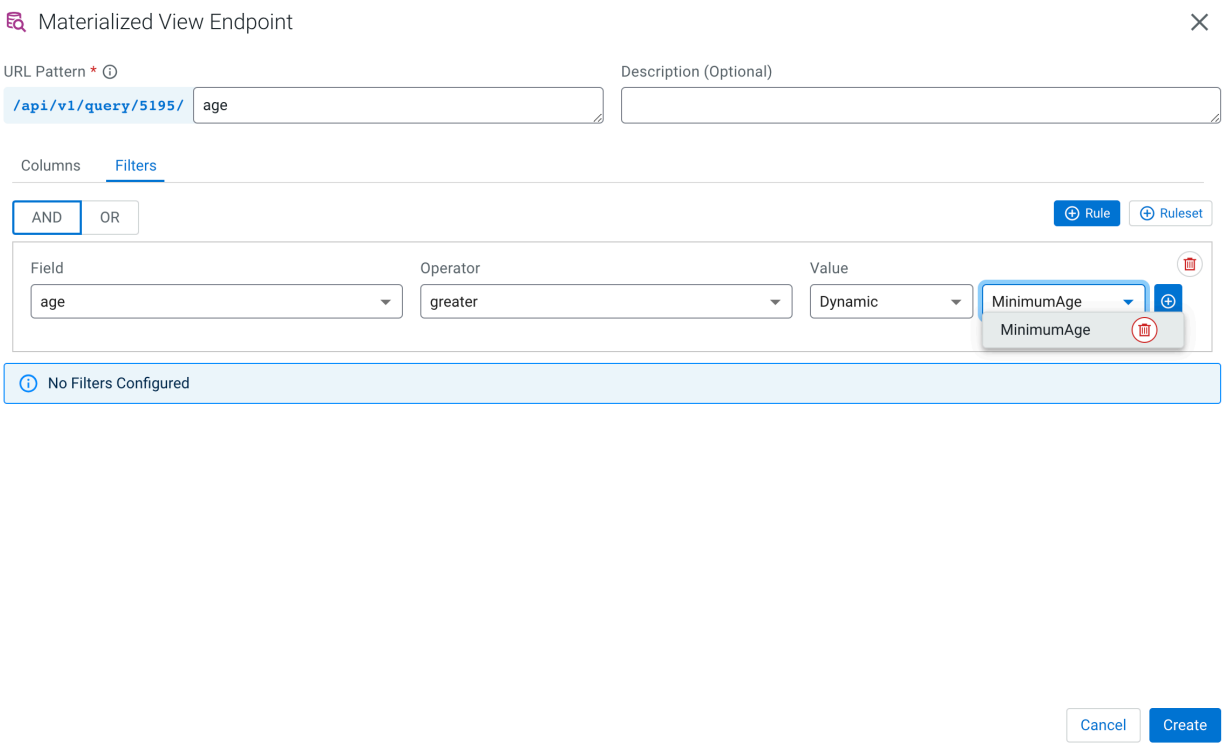

Configuring Materialized View database information

In Cloudera Public Cloud, PostgreSQL is automatically configured for Cloudera SQL Stream Builder when creating your Cloudera Data Hub cluster using the Cloudera Streaming Analytics cluster template. You can review the database configuration for Cloudera SQL Stream Builder using Cloudera Manager.

About this task

Cloudera SQL Stream Builder uses PostgreSQL database to store the queried data of the Materialized View feature. In Cloudera Public Cloud, you can only use PostgreSQL as a database for Cloudera SQL Stream Builder. In case you need to connect to the PostgreSQL database of Cloudera SQL Stream Builder, you can review the necessary information of the database in Cloudera Manager. You can also configure the user and the password of the database using CLI, and then updating the existing information in Cloudera Manager.

Where to find the PostgreSQL database information?

1. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
2. Select Cloudera Manager from the list of services.
3. Select Clusters Cloudera SQL Stream Builder .
4. Click Configuration.
5. Filter down the configuration parameters to Materialized View Engine.

   The following information is listed about the PostgreSQL database:

   - Database Name
   - Database Host
   - Database Port
   - Database User
   - Database Password

When you want to change the default configuration of PostgreSQL database, first you need to change the information in PostgreSQL and then update the configuration in Cloudera Manager for Cloudera SQL Stream Builder.

Procedure

1. Connect to the database host using ssh.

```
ssh [***WORKLOAD USERNAME***]@[***DATABASE HOST***]
Password: [***YOUR WORKLOAD PASSWORD***]
```

2. Connect to PostgreSQL.

```
sudo -U postgres psql
```

Note: As an EnvironmentUser, you will be prompted to provide the password related to the PostgreSQL client. In this case, please reach out to your administrator to provide the correct password for the PostgrSQL client connection.

3. Change the username or the password for the database.

```
ALTER USER eventador_admin WITH PASSWORD [***NEW PASSWORD***];
ALTER USER eventador_admin RENAME TO [***NEW USERNAME***];
```

Note: When changing the database username, the existing password for the username is deleted. You must change the password as well when renaming the user of the database.

After changing the password or username, you need to update the database information in Cloudera Manager.

4. Navigate to Cloudera Management Console Environments , and select the environment where you have created your cluster.
5. Select Cloudera Manager from the list of services.
6. Select Clusters Cloudera SQL Stream Builder .
7. Click Configuration.
8. Filter down the configuration parameters to Materialized View Engine.

   The list of Materialized View Engine paramaters are displayed.
9. Update the Database user or the Database password field based on the new parameters you provided in PostgreSQL.

Using Cloudera SQL Stream Builder with Cloudera Data Visualization

You can create Business Intelligence reports from the Materialized Views created in Cloudera SQL Stream Builder using Cloudera Data Visualization. To integrate Cloudera SQL Stream Builder with Cloudera Data Visualization, you need to provide the PostgreSQL database information of Cloudera SQL Stream Builder in Cloudera Data Visualization.

About this task

After creating Materialized Views of your SQL Stream job, you can create visualized reports from your queried result with Cloudera Data Visualization.

Cloudera Data Visualization enables you to explore data and communicate insights by using visual objects. You can connect to your data in Cloudera, create state-of-the-art visualizations on top of your datasets, build informative dashboards and applications, and publish them anywhere across the data lifecycle.

When connecting Cloudera SQL Stream Builder with Cloudera Data Visualization, you need to provide the PostgreSQL database information that stores the Materialized View data in the Cloudera Data Visualization web interface. For the connection, you can find the PostgreSQL database information in Cloudera Manager.

Before you begin

You have created a Materialized View query on your running SQL job.

Procedure

1. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
2. Select Cloudera Manager from the list of services.
3. Select Clusters Cloudera SQL Stream Builder .
4. Click Configuration.
5. Filter down the configuration parameters to Materialized View Engine.

   The list of Materialized View Engine paramaters are displayed.
6. Save the ssb.mve.datasource related information.

   Regarding the PostgreSQL information you need for the connection, the Database URL (JDBC) configuration parameter contains the PostgreSQL hostname, the PostgreSQL port and the PostgreSQL database name.

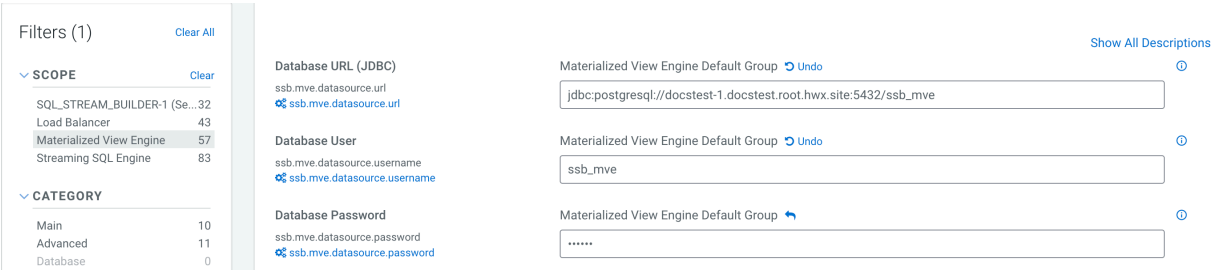

   When creating a data connection in Cloudera Data Visualization, you need to provide the connection information and credential as shown in the following example:

   - Database Name: ssb_mve
   - Database Host: docstest-1.docstest.root.hwx.site
   - Database Port: 5432
   - Database User: ssb_mve
   - Database Password: *[***POSTGRESQL DATABASE PASSWORD***]*

   The Database User and Database Password for the Materialized View Engine is configured automatically when creating your Data Hub cluster. In case you cannot access the Database information, the Administrator of the environment can help to change the configuration as described in Configuring Materialized View database information.

What to do next

After collecting the necessary information to the connection, you need to access the Cloudera Data Visualization web interface and create a connection to the PostgreSQL database of Cloudera SQL Stream Builder using the Cloudera SQL Stream Builder connector. For more information, see the Cloudera Data Visualization documentation.

Important: The Cloudera SQL Stream Builder connector is provided as a technical preview at this time in Cloudera Data Visualization. The tool is still under development and not recommended for a production environment.

Using widgets for data visualization

The built-in data visualization tool in Cloudera SQL Stream Builder enables you to present the sampling data and the results of the Materialized View query using widgets on Streaming SQL Console. As the widgets are integrated into Cloudera SQL Stream Builder, the visualization tool works out of the box without any dependencies, which offers easy access to the underlying, running jobs as data sources.

You can use widgets on Streaming SQL Console to visualize data from the samples of a running job or from the results of a Materialized View query. Various visualization types and configuration options are available to visualize data. Widgets automatically refresh to update the visualization based on the data sampled directly from the job output or from Materialized Views.

Creating widgets

Widgets can be created on Streaming SQL Console when a SQL query is submitted and the job is successfully running.

Before you begin

- Ensure that you have a running job with in progress sampling or you have a running job with a Materialized View endpoint.

  For more information, see the Running SQL Stream jobs and Creating Materialized Views documentation.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click Dashboard from the Project Manager.

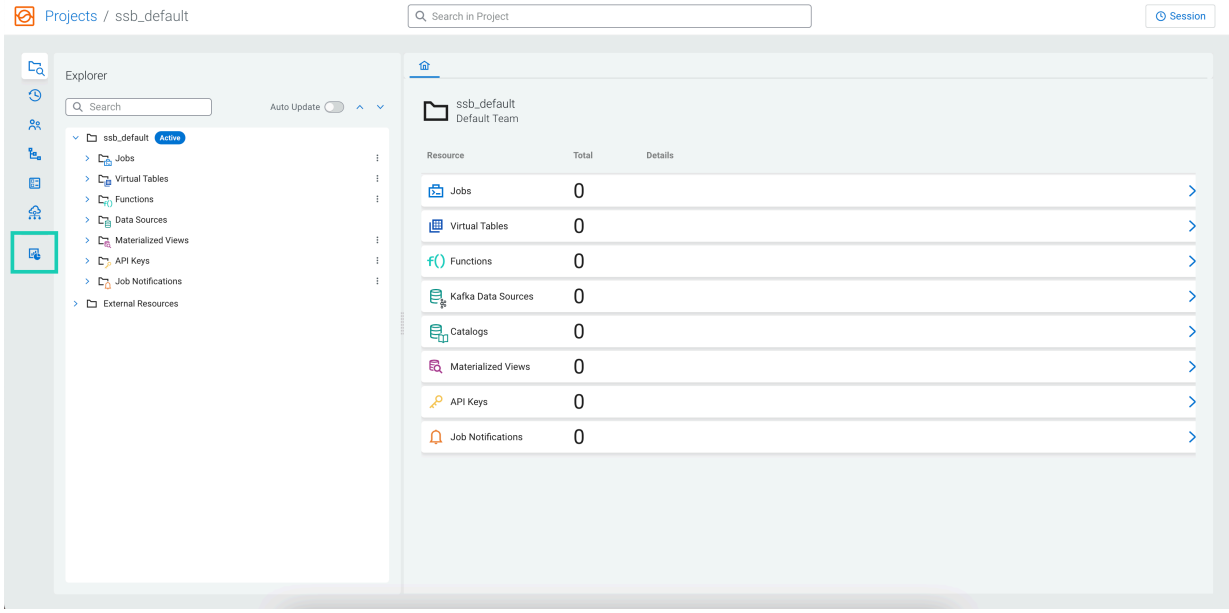

4. Click New Widgets.
   The **Widget editor** page appears.

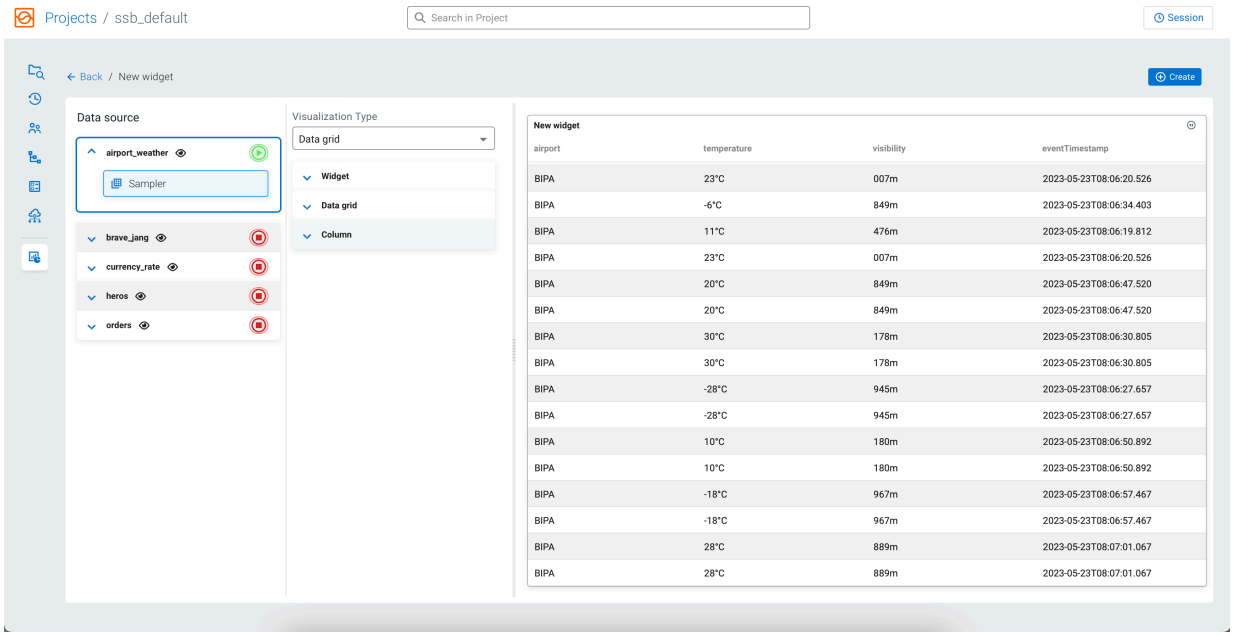

   The Widget editor page consist of the following windows:

   - **Data source**, where you can select the job and its corresponding data sources for the widget.
   - **Visualization Type** where you can provide information and customize the visualization type.
   - **Preview of the created widget**, where you can check the configurations and customization of the widget that will be added to the **Dashboard**.

5. Select a Data Source.

   The running jobs will be automatically displayed and categorized based on the Sampler or the Materialized View endpoint name of the corresponding job.For more information, see the Choosing data sources section.

6. Select a Visualization type.

   A preview is available from the selected data source and visualization type.

7. Customize the Visualization based on the available options.

   For more information, see the Customizing visualization types section.

8. Click Create.
   The created widget is added to the **Dashboard** page.

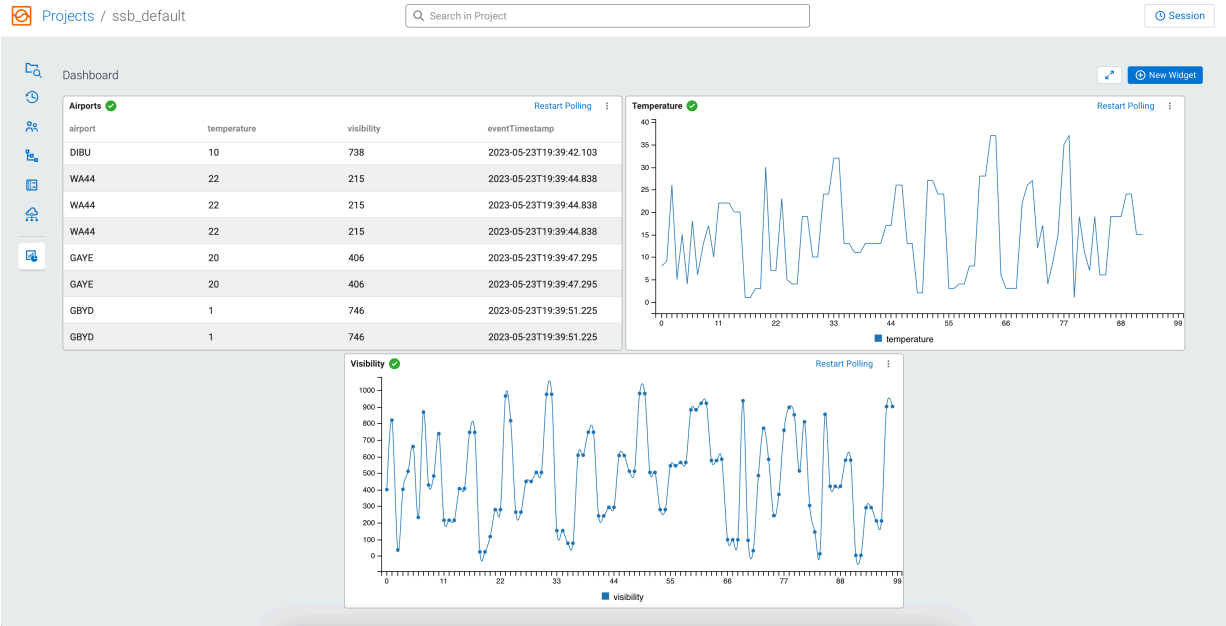

You can manage the created widget on the Dashboard by clicking ⋮, and selecting Edit to go back to the widget editor page and make changes to the visualization, or Remove to delete the widget.

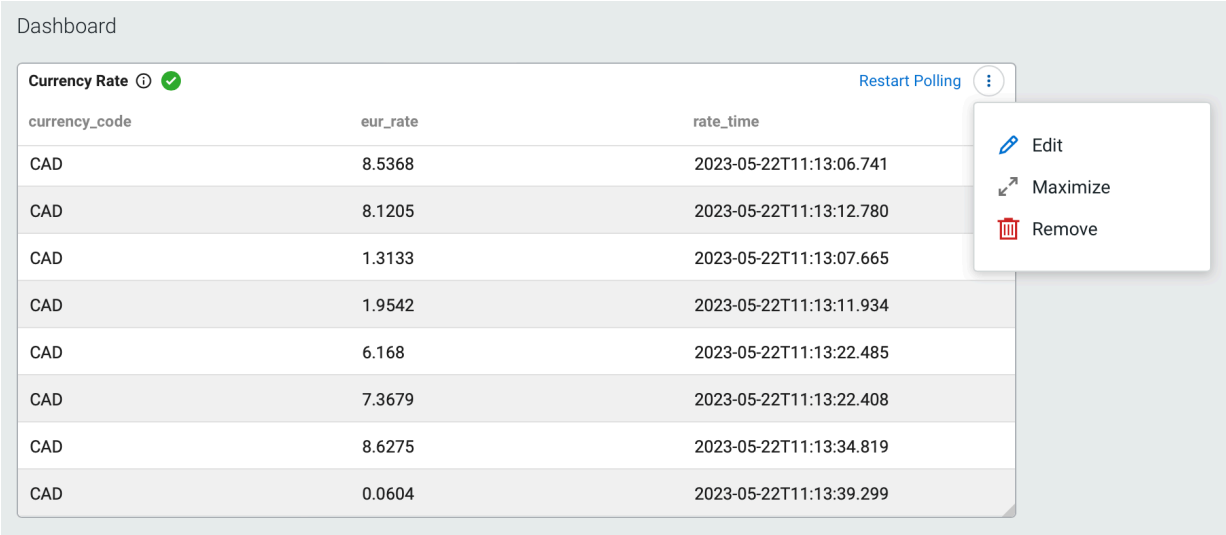

What to do next

You can create more widgets to the Dashboard page. You can resize and reorganize the widgets based on your requirements. For more information, see the Managing widgets on the Dashboard section.

Choosing data sources

The data sources are available for visualization as soon as a job sampling is in progress or a Materialized View endpoint is created. In both cases, the data is shown in the widgets when the corresponding SQL jobs are in running state.

Samples as data source

When using sampling as a data source, you must have a running job on Streaming SQL Console, which produces sampling data as the job runs. You can change the default sampling behavior at the **Job Settings** that affects the frequency, count and the sample window size of the sampled messages.

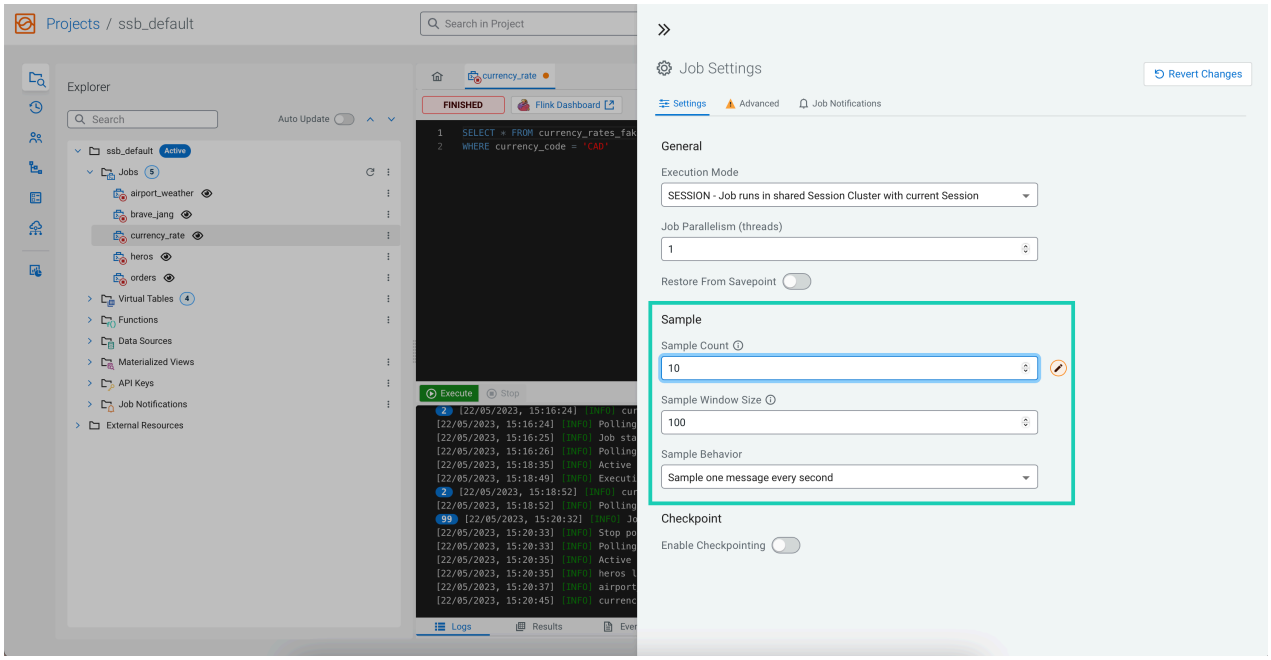

Materialized View as data source

When using Materialized View as a data source, you must have a running job with a Materialized View query. The Materialized View endpoint is called in a period of time according to the update interval and query limit configured for the widget.

Update interval

The update interval is the time the widgets waits to call from the endpoint.

Query limit

The query limit sets the maximum amount of records to be returned for each call

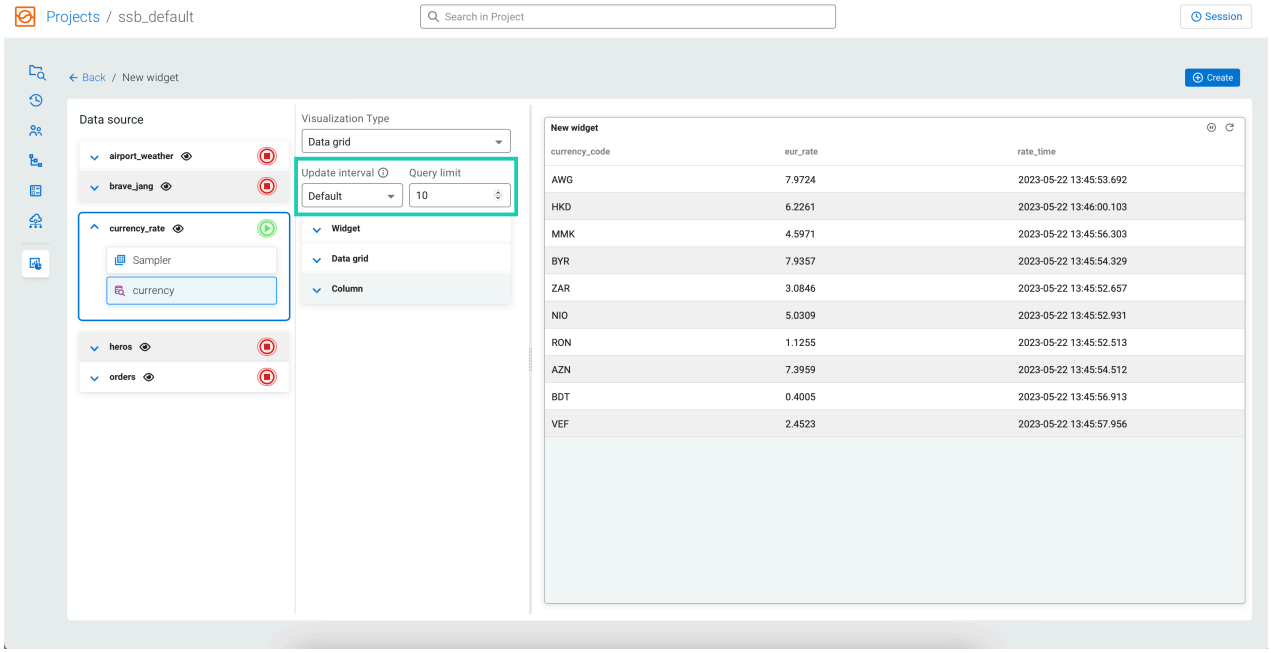

Note: The data retention configuration that is provided when creating the Materialized View affects the results of this data source. For example, when setting the retention for row count, the specified number of values will be shown on the widget.

If the update interval is set to Default, the update interval can be specified on the **Dashboard** after the widget is created. The time selected on the **Dashboard** will be applied to every widget that has **Default** configured for update interval.

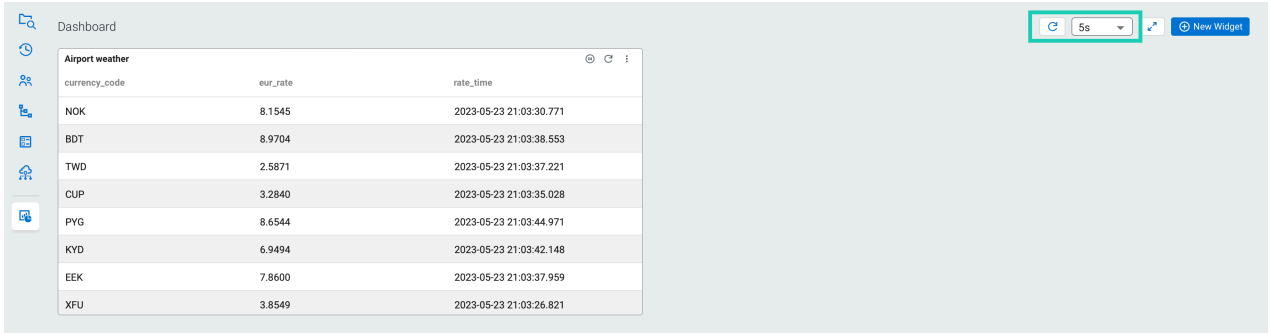

Managing data source jobs

The running jobs corresponding to the widgets can be manage from the Widget editor page.

You can manage the jobs corresponding to the data sources from the widget editor page by clicking on the job status icon at the data source as shown in the following example:

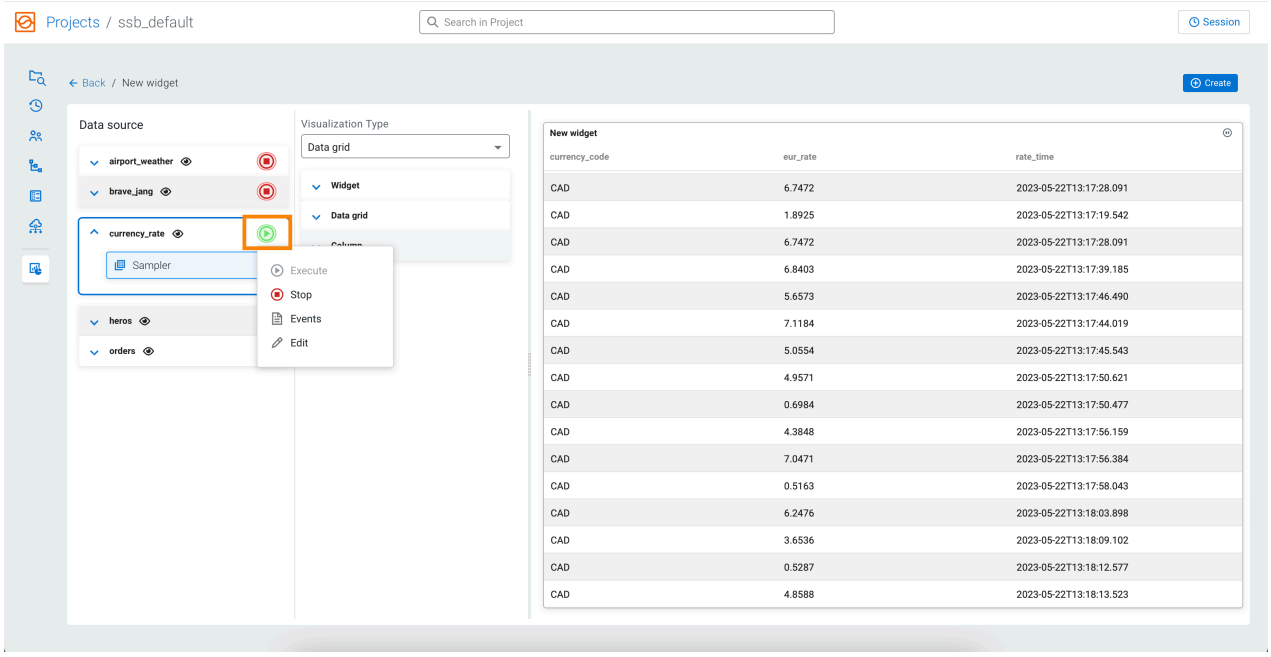

You can Execute a stopped job when you want to use it for previewing the data on a visualization, and you can also Stop the running job if you no longer need it. When using sampling as a data source, you cannot select the sampler if the job is stopped. If a Materialized View endpoint is used as a data source, you can view the data on the visualization as long as the data retention time is configured in the Materialized View settings.

The Events option can be used to check the information, warning and error messages that correspond to the selected job.

By clicking on Edit, you are redirected to the SQL editor to where you can configure the job or change the SQL query of the job. In case you have created a widget and restart the corresponding job with changes, you need to restart the polling for the widget on the **Dashboard** as well.

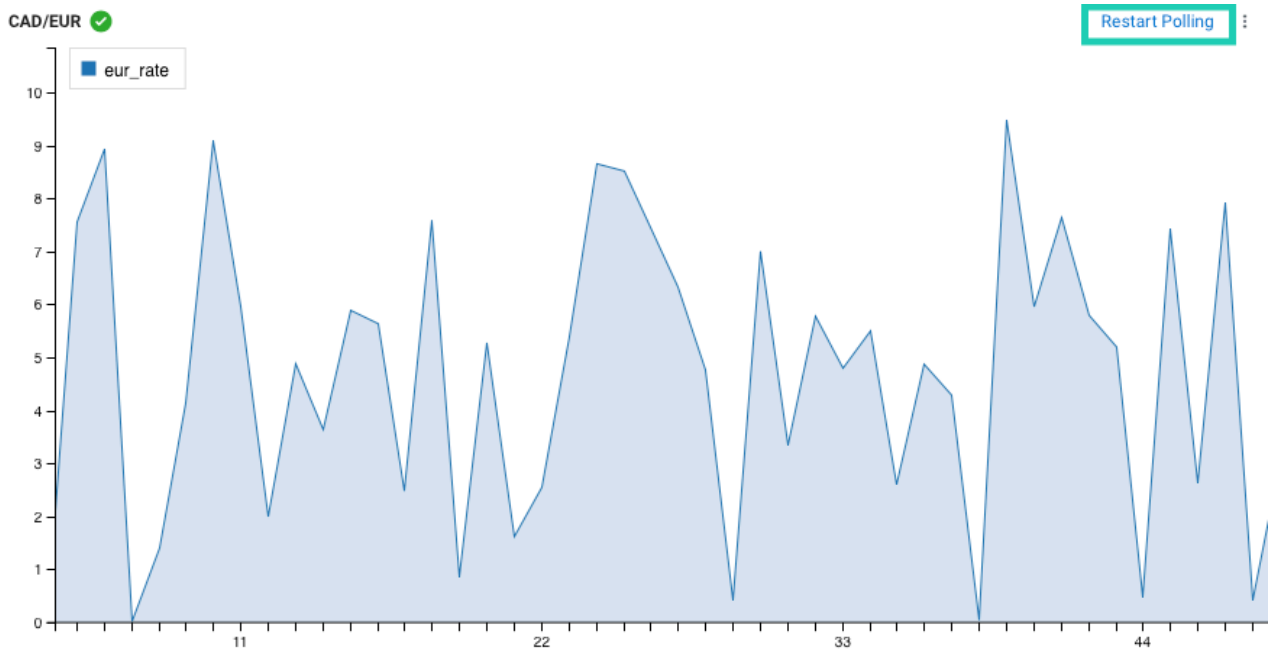

Customizing visualization types

Widgets allow you to choose from a variety of visualization types and customize them based on your requirements. Each visualization type has general configurations and its own graph style configuration that can be set when creating the widget.

Configuring widget, fields and legend

The widget, data field, label field and legend are general configurations that can be set for all of the visualization types.

Name and Description

You can provide a custom name and short description about the widget. The name of the widget appears at the right top corner of the widget. The description can be viewed if it is provided by hovering over ⓘ next to the widget name.

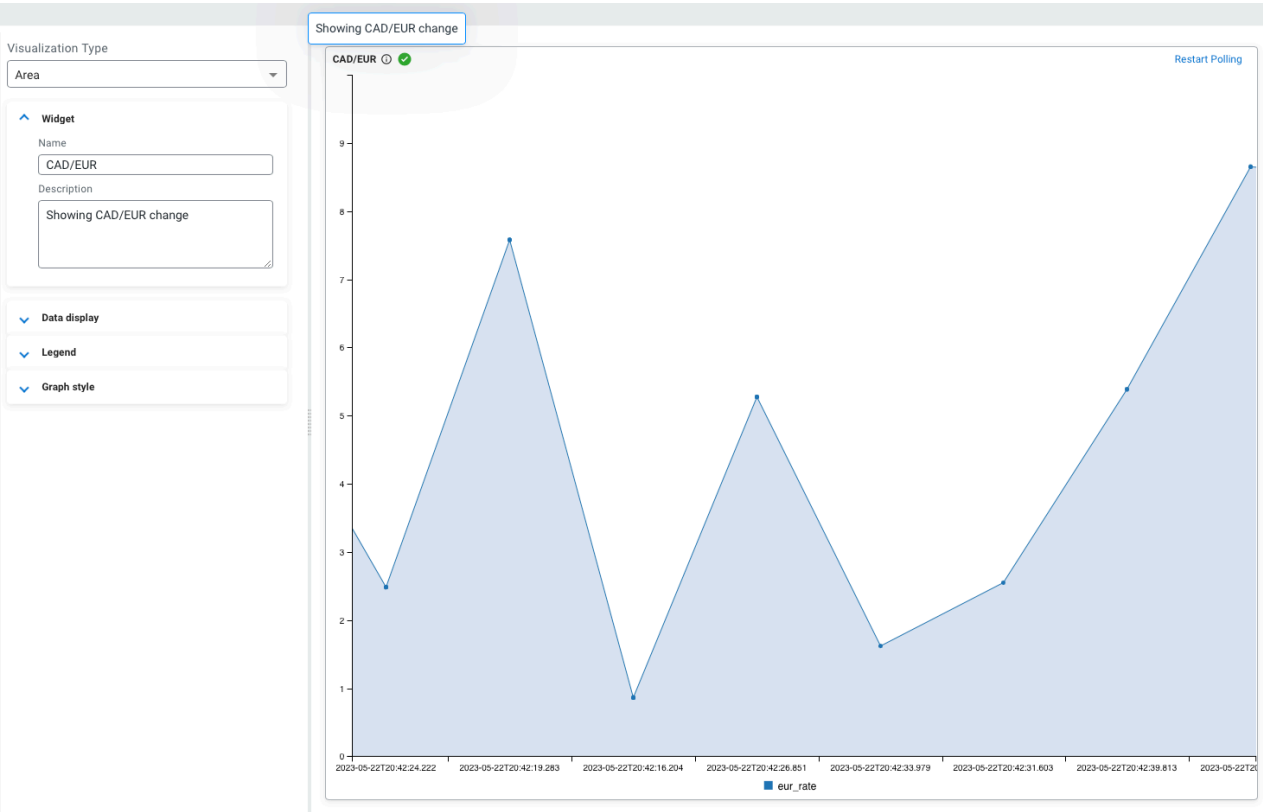

Data fields

You need to select the data fields to show on the axis of the visualization. The fields correspond to the columns of the table. You can select more than one data field to be displayed on the widget. The values of the data field are shown on the Y axis.

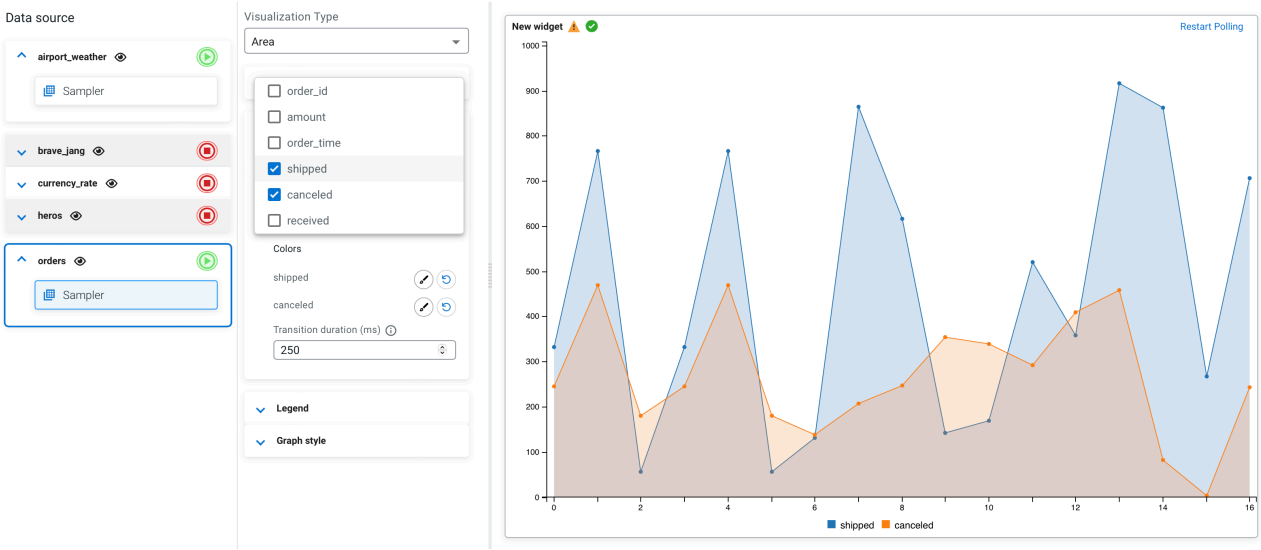

Label field

You have the option to select a field to show as labels. The fields correspond to the columns of the table. If the Label field is left empty, the number of records will be displayed as labels on the X axis.

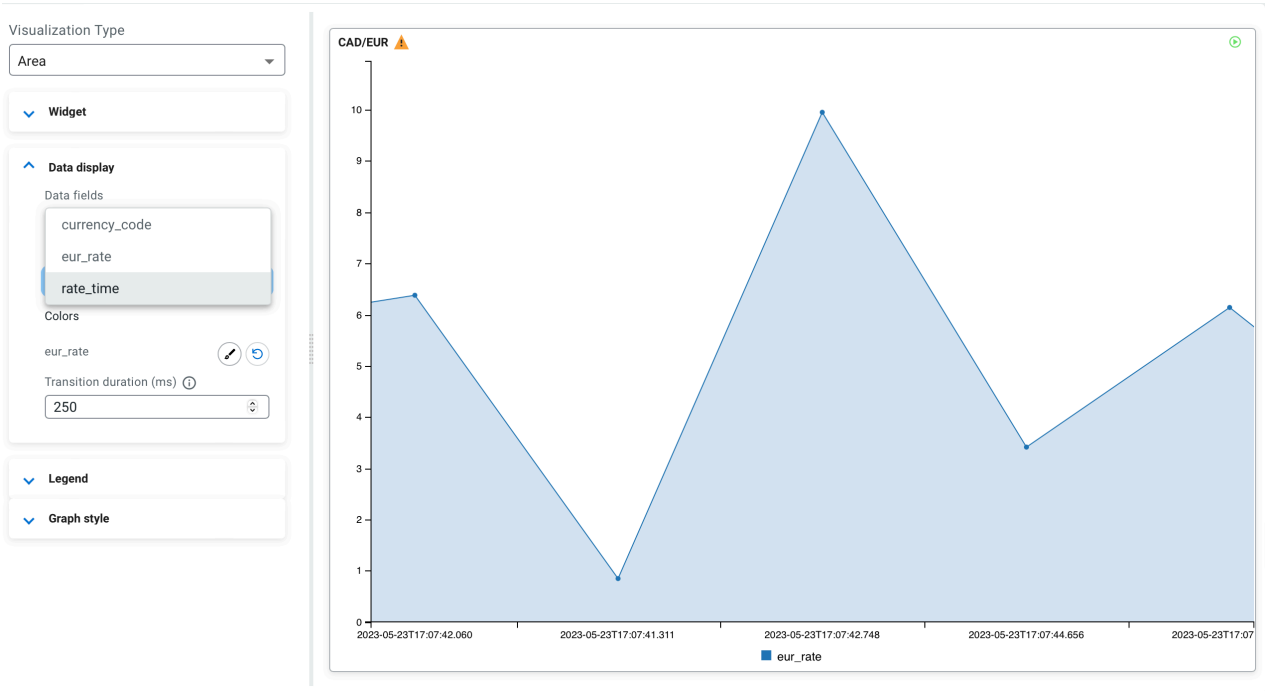

Color picker

You can change the default visualization color of the selected data fields. You can open the color picker using button, and pick a color over the palette, or provide the Hex, RGBA value to a specific color. You can also choose from one of the preset colors. After clicking on Apply, the color of the selected data field is updated.

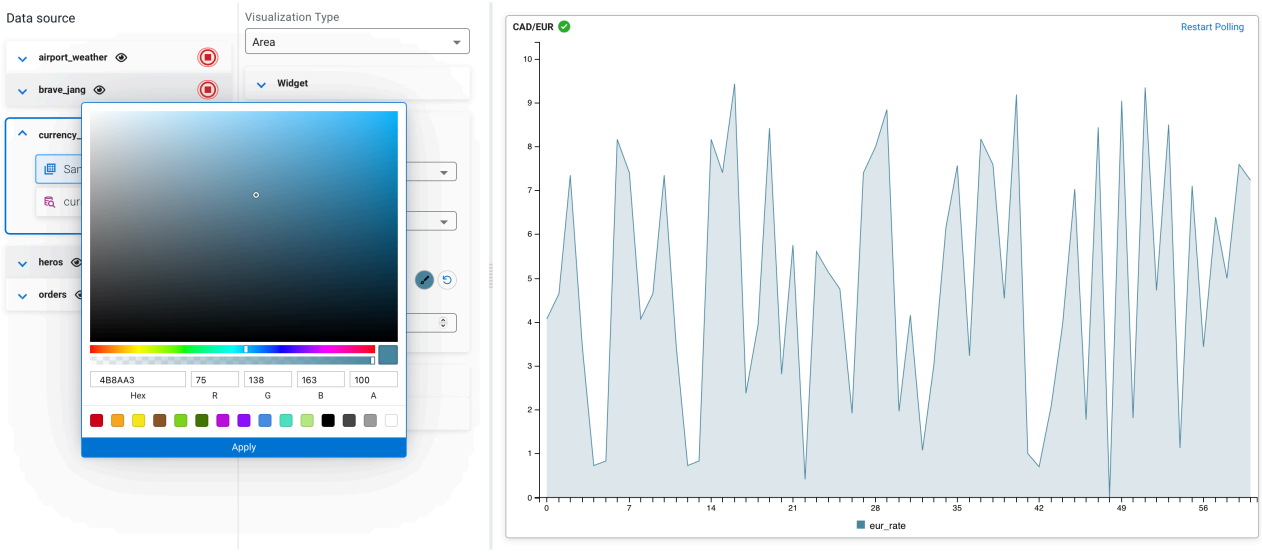

You can reset the color of the visualization using the button.

Transition duration (ms)

You can set the time for how slow or fast the data should change on the widget. The default is 250 ms. If set to 0 or left empty, the transition is skipped. The higher the time is set, the slower the visualization changes on the widget.

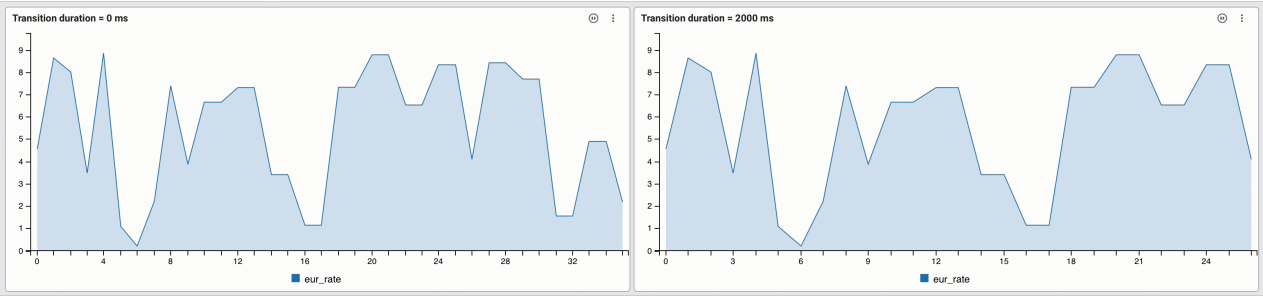

Visibility and position of Legend

You can enable or disable the legend on the visualization, and also choose to position the legend to the bottom or right side of the widget, or place it in the visualization.

For Bottom

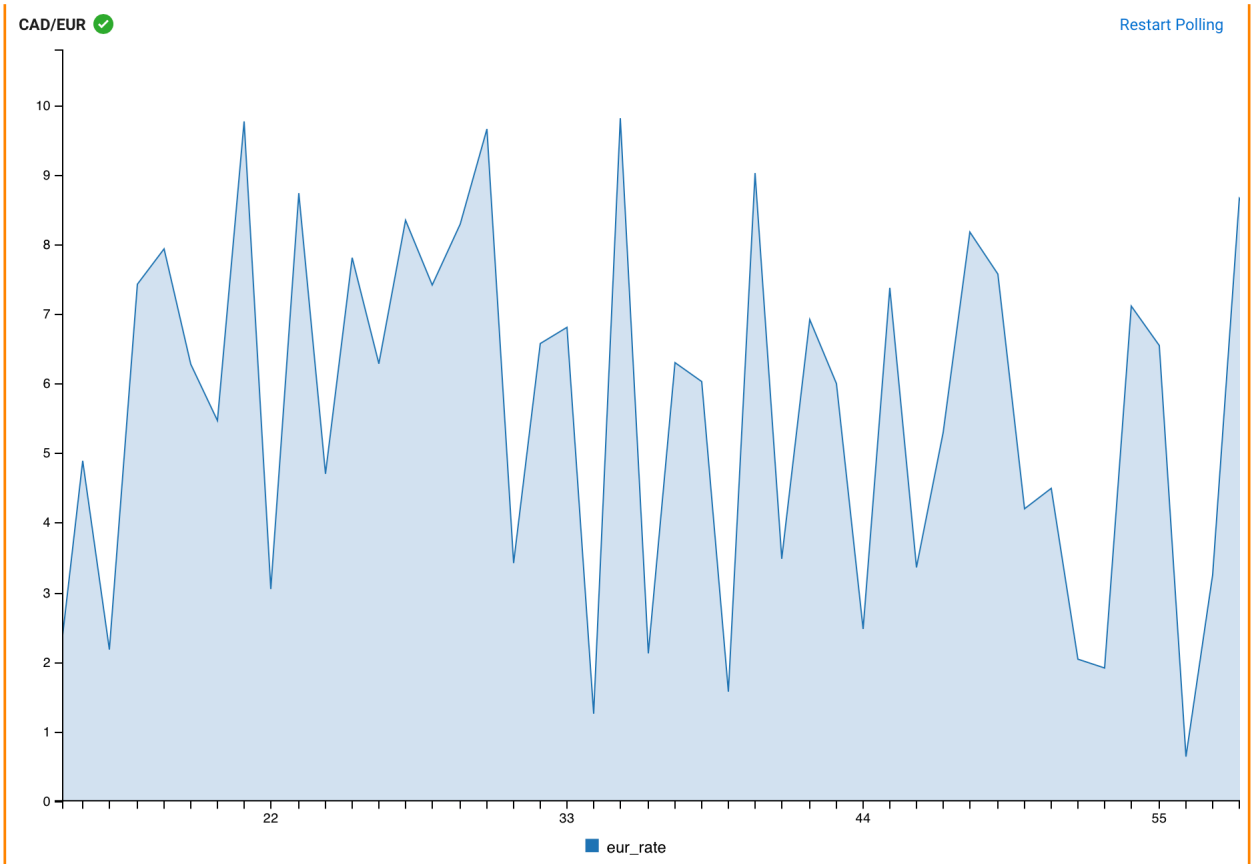

For Right side

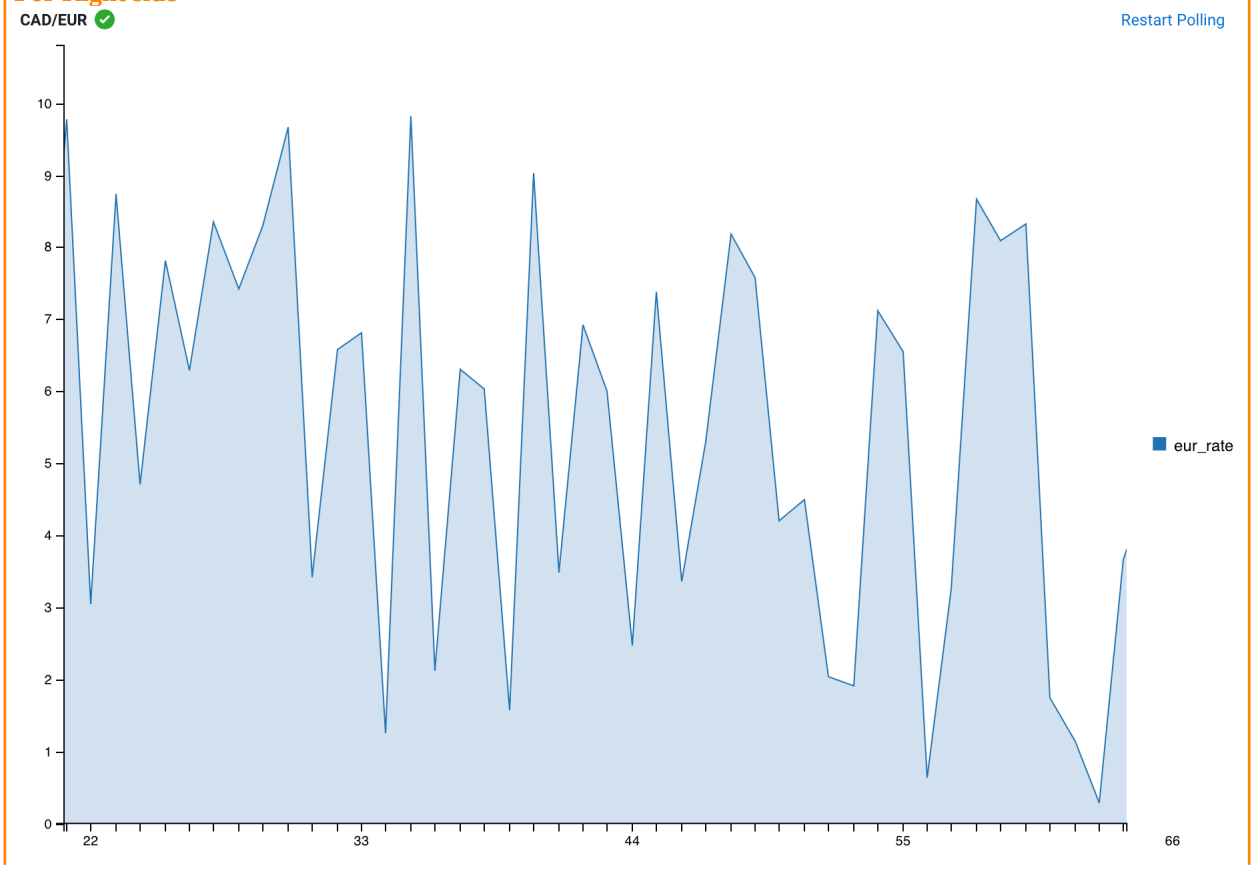

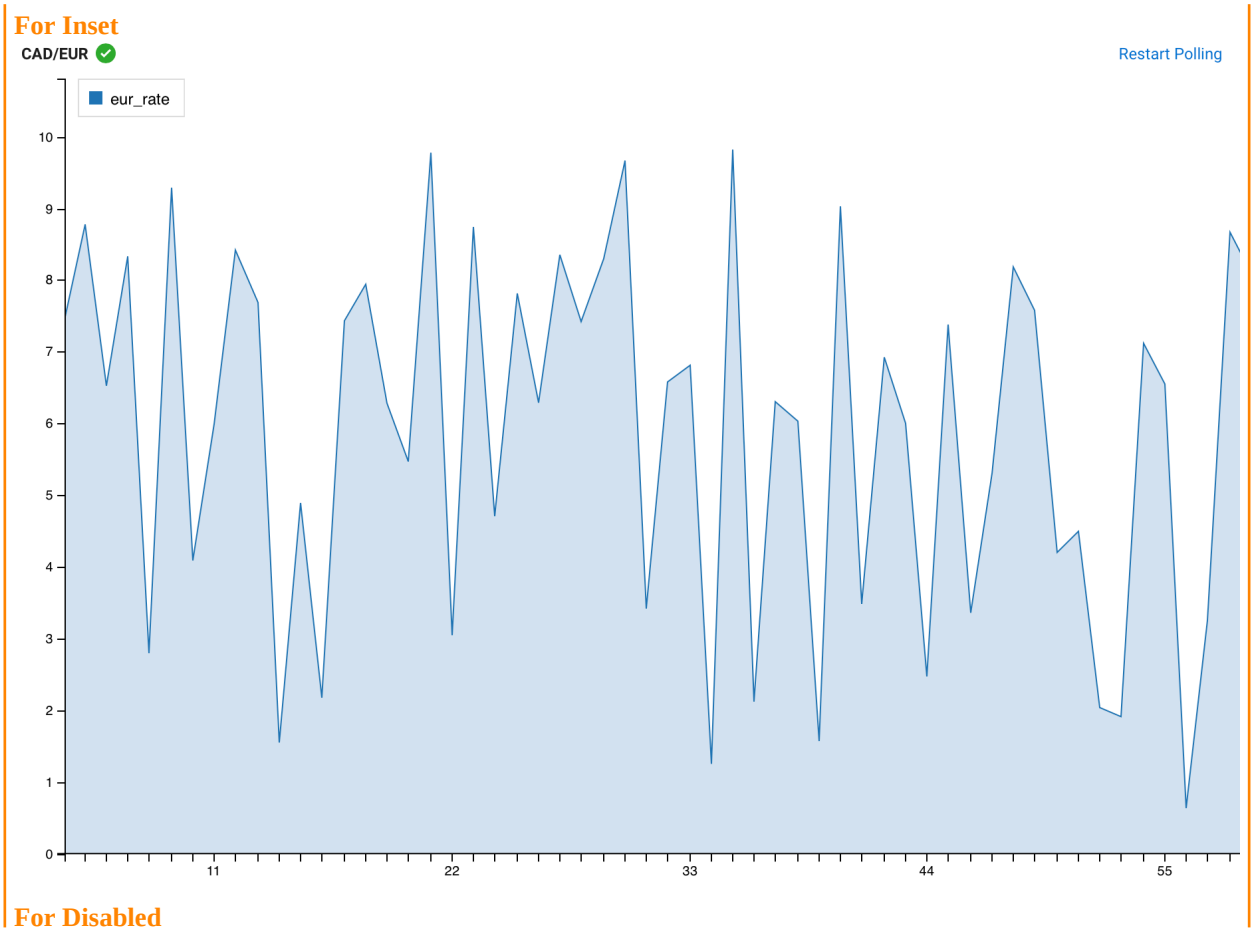
For Inset
CAD/EUR
Restart Polling
eur_rate
10
9
8
7
6
5
4
3
2
1
0
11
22
33
44
55
For Disabled

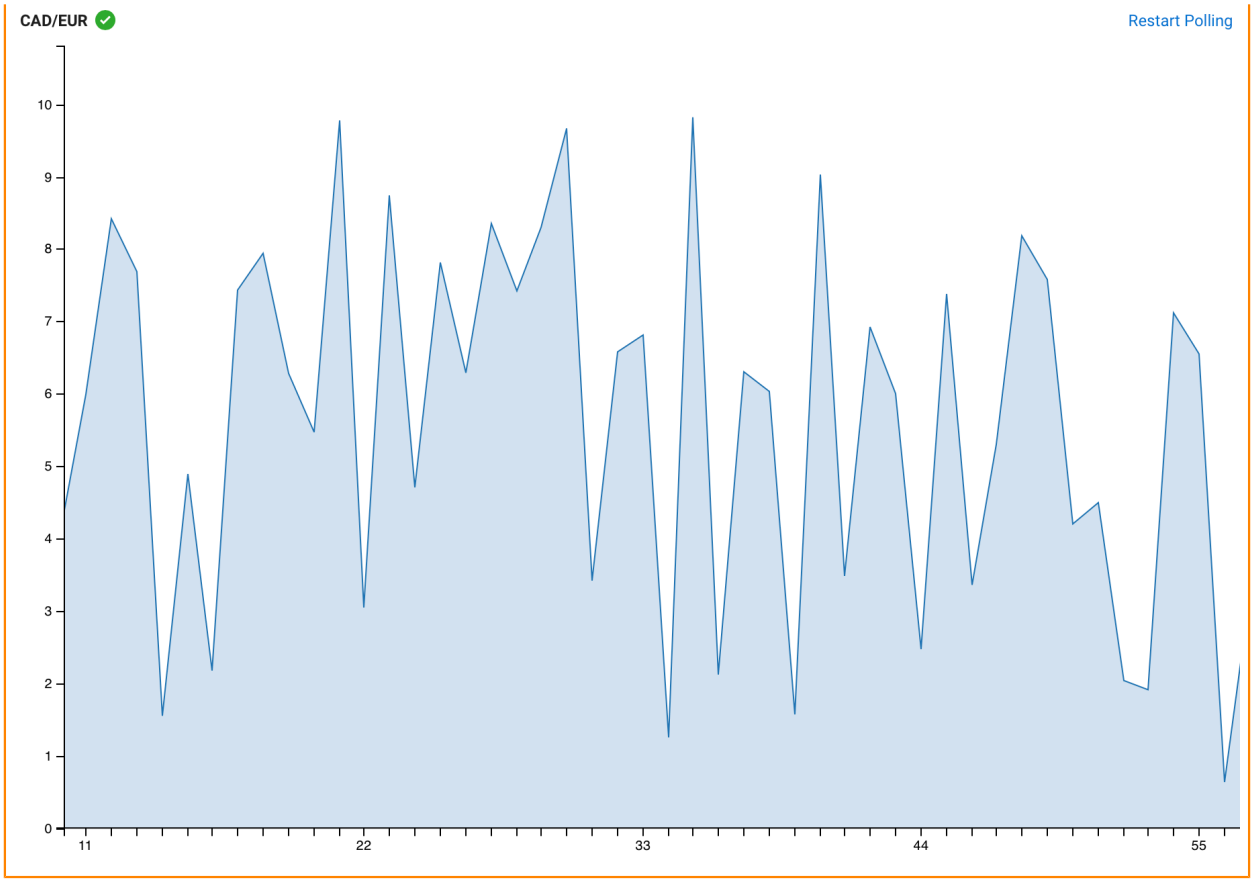

Configuring graph style

You can customize the styles of the widgets based on how you want to visualize the data.

Visualization type: Area

You can use the Area visualization type when you want to show how the values changed over a period of time.

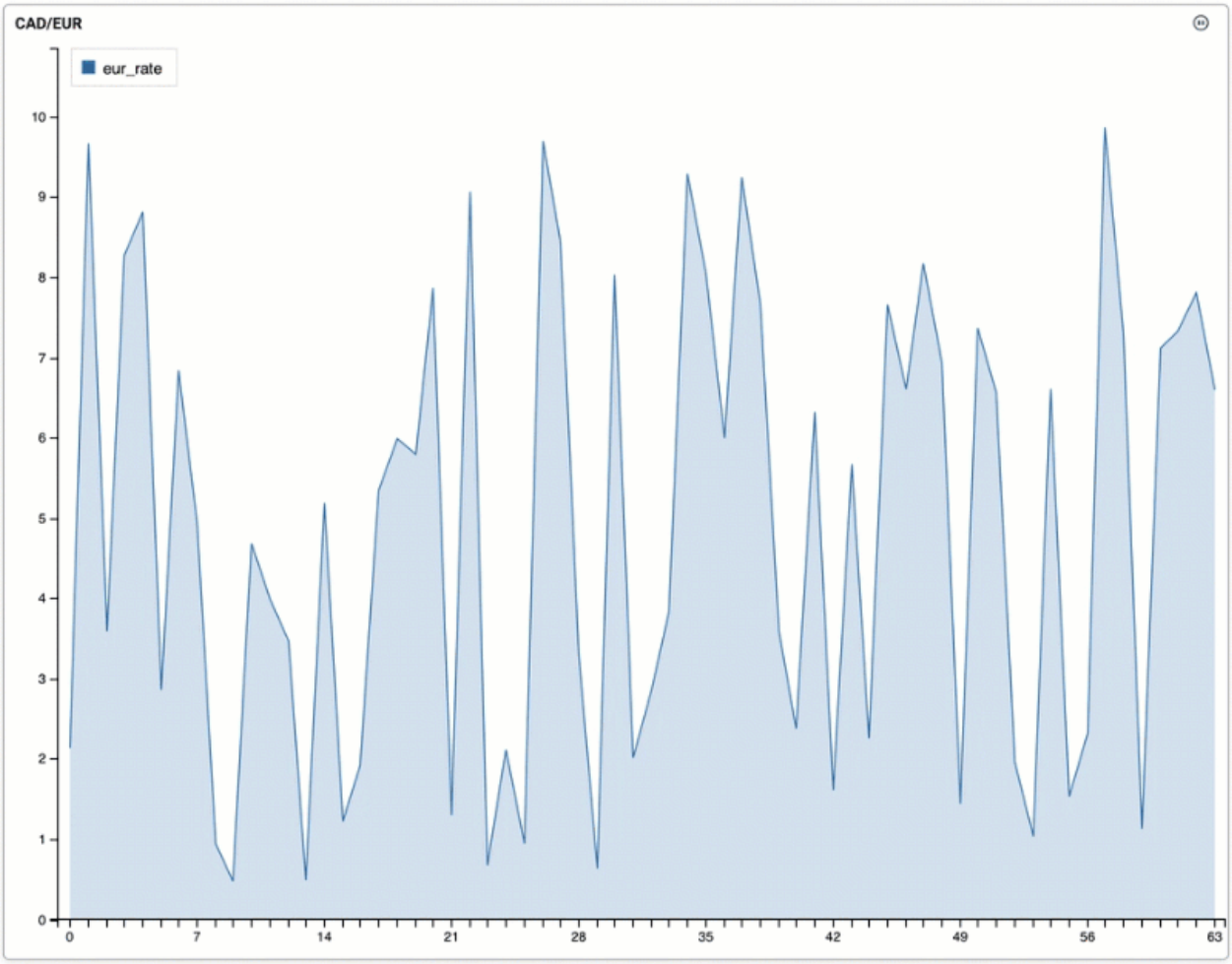

The following table and query was used for the Area widget:

```
CREATE TABLE currency_rates (
  `currency_code` VARCHAR(2147483647),
  `eur_rate` DECIMAL(6, 4),
  `rate_time` TIMESTAMP(3)
) WITH (
  'fields.currency_code.expression' = '#{Currency.code}',
  'fields.rate_time.expression' = '#{date.past ''15'',''SECONDS''}',
  'connector' = 'faker',
  'rows-per-second' = '100',
  'fields.eur_rate.expression' = '#{Number.randomDouble ''4'',''0'',''10''}'
);

SELECT * FROM currency_rates
WHERE currency_code = 'CAD'
```

The following configurations have been set for the Area widget:

Name	CAD/EUR
Data fields	eur_rate
Legend poition	Inset
Line interpolation	Area
Connect null	enabled

Show points	disabled

You can further customize the graph style of the Area widget with the following Graph style configurations:

Line interpolation

Determines the line interpolation of the visualization. You have the option to select Area-spline and Area-step for line interpolation.

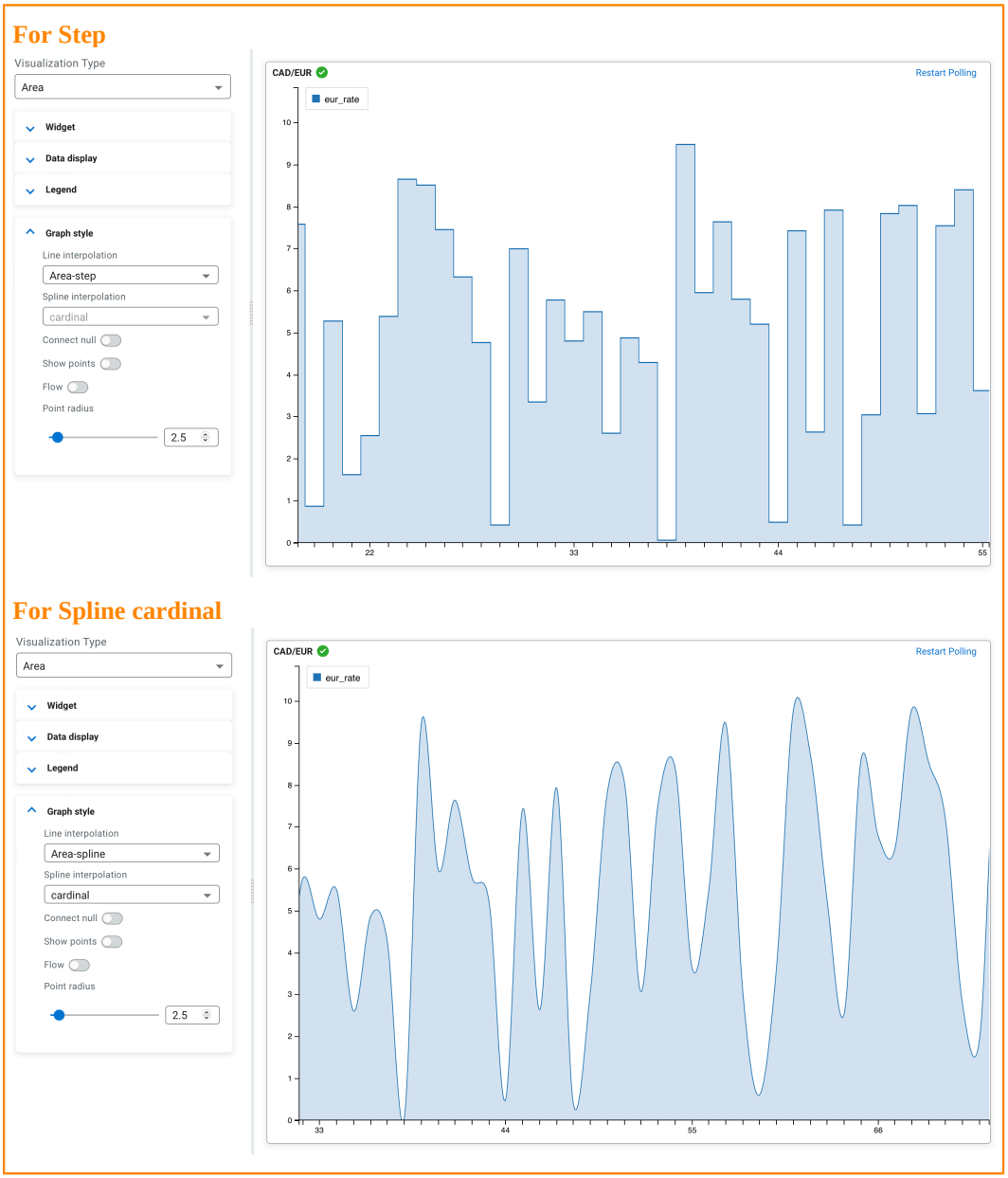

When selecting Area-spline as interpolation, you can also specify the Spline interpolation which includes the following options:

- Linear, linear-closed
- Basis, basis-open, basis-closed
- Cardinal, cardinal-open, cardinal-closed
- Monotone
- Step, step-before, step-after

Show points and Point radius

You can enable whether to show each data point in line using the Show points toggle. The size of the points can be changed using the Point radius configuration, which is set to 2.5 by default. The following example shows the enabled Show points with 4.0 Point radius:

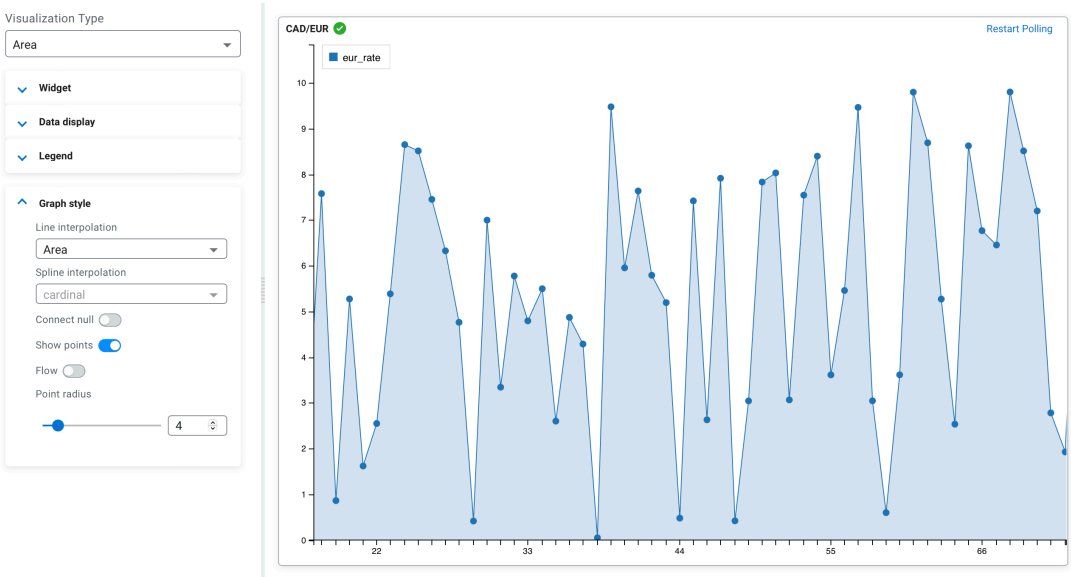

Connect null

Using the connect null configuration, you can set if the null data points are connected or not. If enabled, the region of the null data is connected without any data point. When disabled, the data points are not connected as shown in the following example:

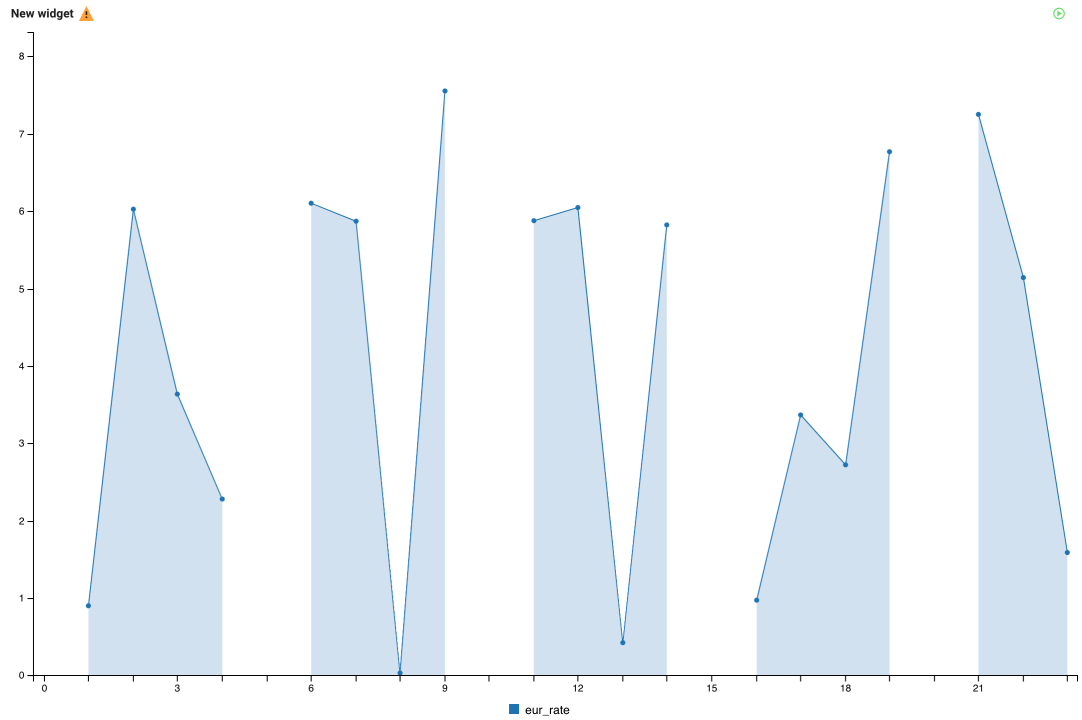

Flow

You can set the movement of the data by enabling or disabling the Flow function, which means the new data points flow to the chart from the right side compared to arriving periodically as processed on the widget.

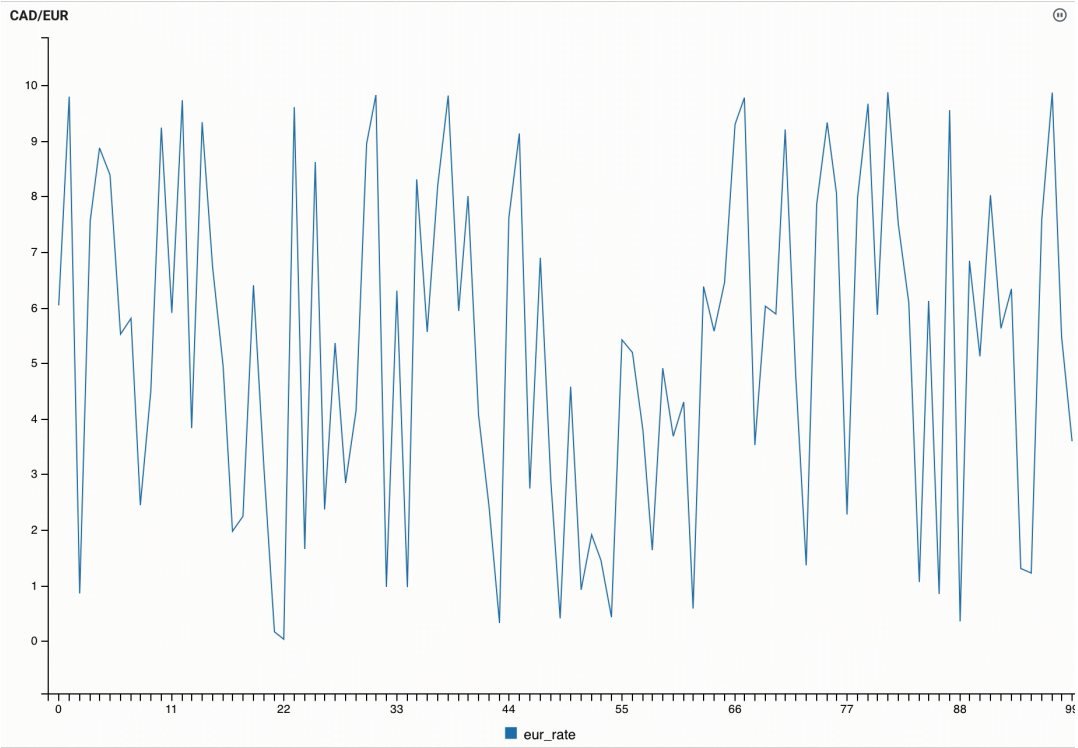

Visualization type: Bar

You can use the Bar visualization type when you want to show how the data is distributed or want to compare the data between different groups.

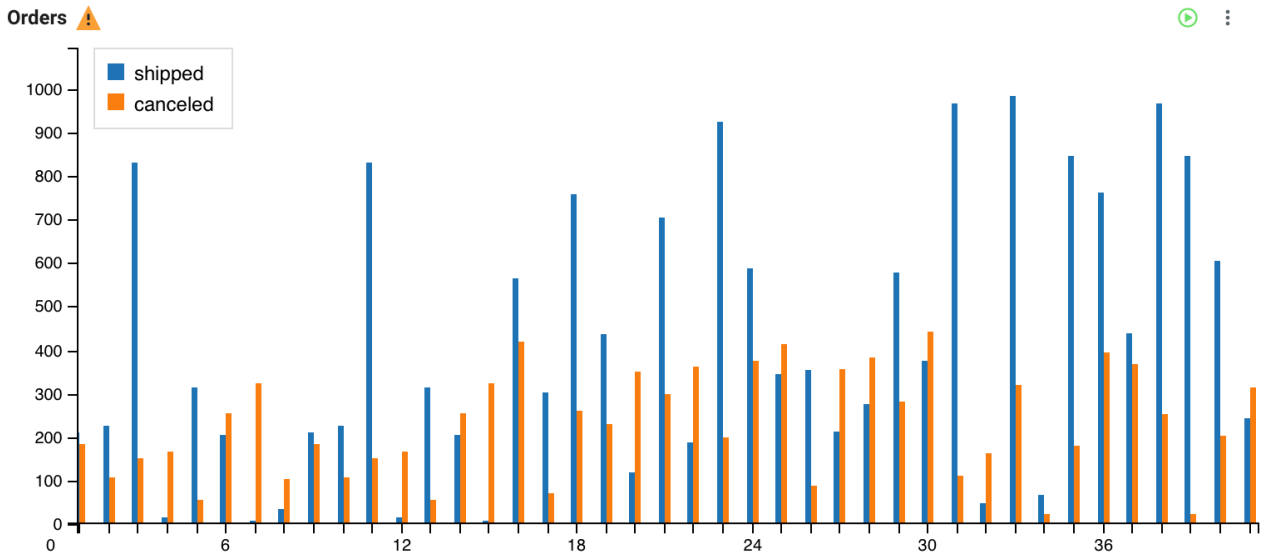

The following table and query was used for the Bar widget:

```
CREATE TABLE orders (
  `order_id` INT,
  `amount` INT,
  `order_time` TIMESTAMP(3),
  `shipped` VARCHAR(2147483647),
  `canceled` VARCHAR(2147483647),
  `received` VARCHAR(2147483647),
  WATERMARK FOR `order_time` AS `order_time` - INTERVAL '15' SECOND
) WITH (
```

```
  'fields.order_time.expression' = '#{date.past ''15'',''SECONDS''}',
  'fields.amount.expression' = '#{number.numberBetween ''0'',''100''}',
  'fields.order_id.expression' = '#{number.numberBetween ''0'',''99999999''
}',
  'connector' = 'faker',
  'fields.shipped.expression' = '#{number.numberBetween ''0'',''1000''}',
  'fields.received.expression' = '#{number.numberBetween ''0'',''800''}',
  'fields.canceled.expression' = '#{number.numberBetween ''0'',''500''}',
  'rows-per-second' = '100'
);

SELECT * FROM orders
```

The following configurations have been set for the Bar widget:

Name	Orders
Data fields	shipped, canceled
Legend poition	Inset
Bar width	0.5

You can further customize the graph style of the Bar widget with the following Graph style configurations:

Bar width

You can specify the width of the bars on the widget by changing the default value.

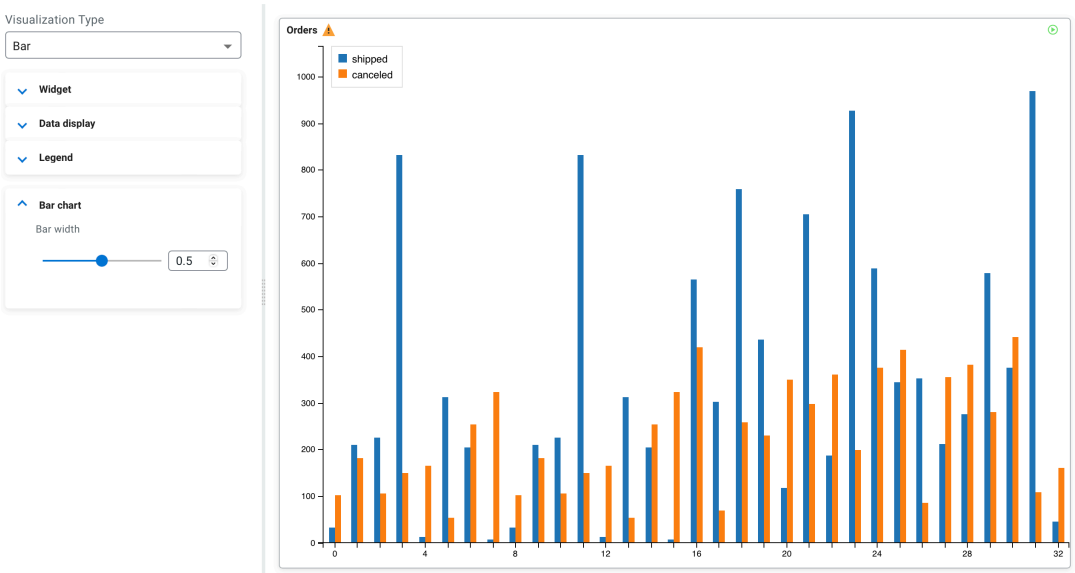

Visualization type: Donut

You can use the Donut visualization type when you want to show the proportions of data within a category where the size of the pieces represent the proportion of each category.

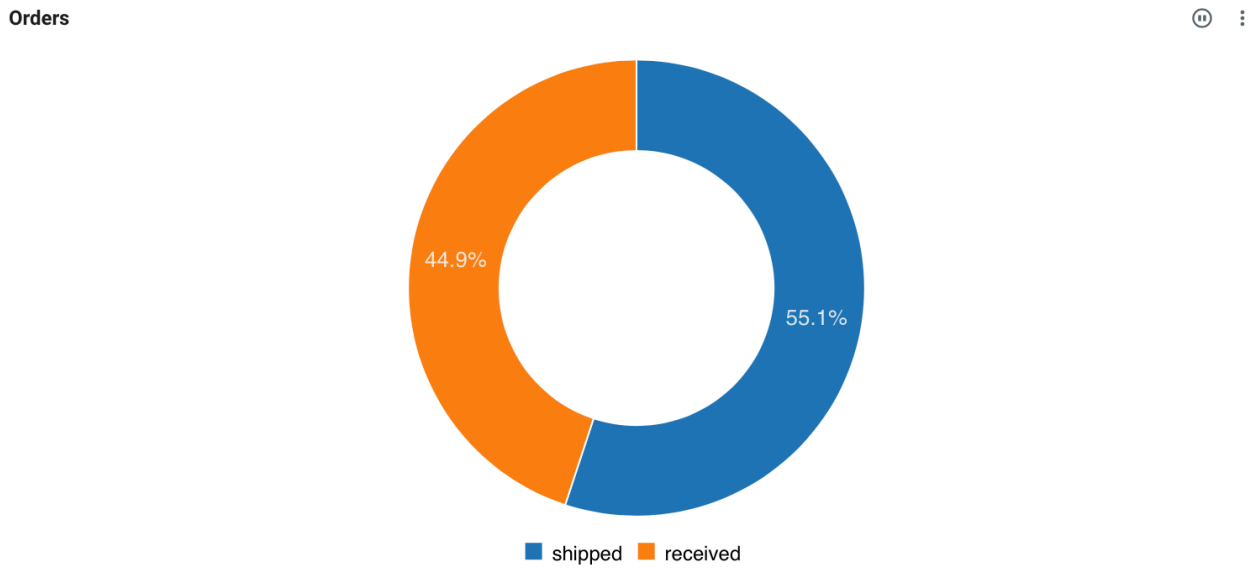

The following table and query was used for the Donut widget:

```
CREATE TABLE orders (
  `order_id` INT,
  `amount` INT,
  `order_time` TIMESTAMP(3),
  `shipped` VARCHAR(2147483647),
  `canceled` VARCHAR(2147483647),
  `received` VARCHAR(2147483647),
  WATERMARK FOR `order_time` AS `order_time` - INTERVAL '15' SECOND
) WITH (
  'fields.order_time.expression' = '#{date.past ''15'',''SECONDS''}',
  'fields.amount.expression' = '#{number.numberBetween ''0'',''100''}',
  'fields.order_id.expression' = '#{number.numberBetween ''0'',''99999999''
}',
  'connector' = 'faker',
  'fields.shipped.expression' = '#{number.numberBetween ''0'',''1000''}',
  'fields.received.expression' = '#{number.numberBetween ''0'',''800''}',
  'fields.canceled.expression' = '#{number.numberBetween ''0'',''500''}',
  'rows-per-second' = '100'
);

SELECT * FROM orders
```

The following configurations have been set for the Donut widget:

Name	Orders
Data fields	shipped, received
Legend poition	Bottom
Show label	enabled

You can further customize the graph style of the Donut widget with the following Graph style configurations:

Show label

You can enable or disable the labels on the visualization as shown in the following illustration where labels are disabled:

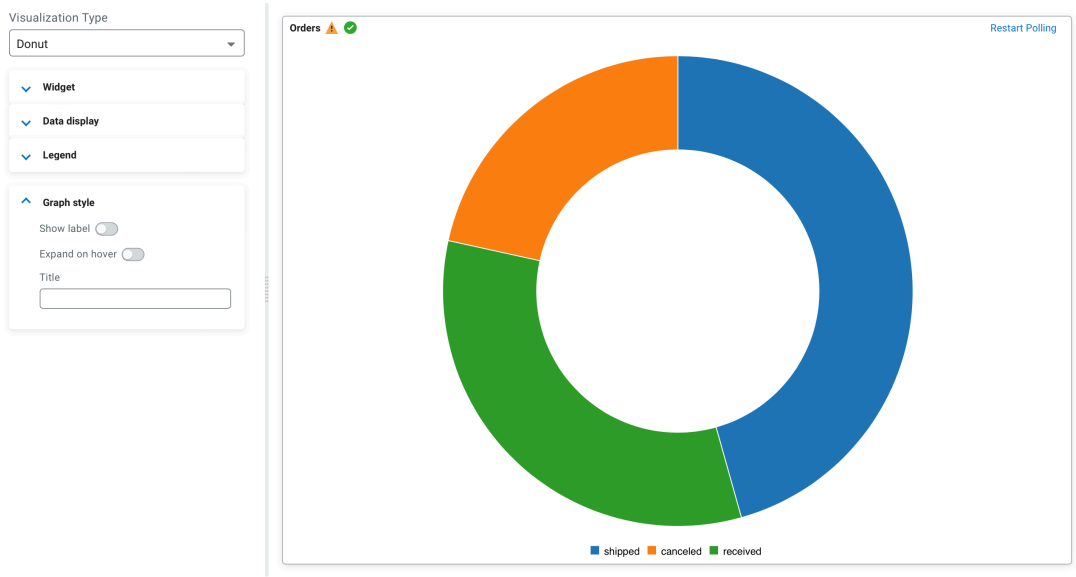

Visualization type: Gauge

You can use the Gauge visualization type when you want to show progress or amount compared to the maximum value.

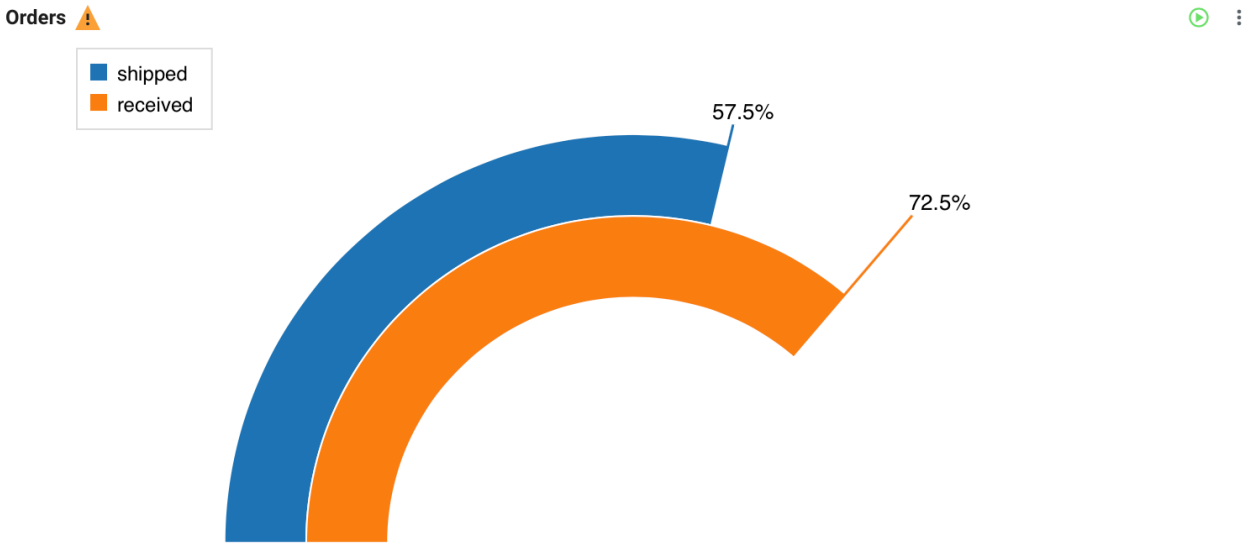

The following table and query was used for the Gauge widget:

```
CREATE TABLE orders (
  `order_id` INT,
  `amount` INT,
  `order_time` TIMESTAMP(3),
  `shipped` VARCHAR(2147483647),
  `canceled` VARCHAR(2147483647),
  `received` VARCHAR(2147483647),
  WATERMARK FOR `order_time` AS `order_time` - INTERVAL '15' SECOND
) WITH (
  'fields.order_time.expression' = '#{date.past ''15'',''SECONDS''}',
  'fields.amount.expression' = '#{number.numberBetween ''0'',''100''}',
  'fields.order_id.expression' = '#{number.numberBetween ''0'',''99999999''
}',
  'connector' = 'faker',
  'fields.shipped.expression' = '#{number.numberBetween ''0'',''1000''}',
```

```
  'fields.received.expression' = '#{number.numberBetween ''0'',''800''}',
  'fields.canceled.expression' = '#{number.numberBetween ''0'',''500''}',
  'rows-per-second' = '100'
);

SELECT * FROM orders
```

The following configurations have been set for the Gauge widget:

Name	Orders
Data fields	shipped, received
Legend poition	Inset
Reducer	Last
Show label	enabled
Min	100
Max	300

You can further customize the graph style of the Gauge widget with the following Graph style configurations:

Reducer

A reducer function that will be used to reduce many fields to a single value. You have the following options to choose from:

- Last
- First
- Min
- Max
- Total
- All

Min, Max

You can specify the minimum and maximum value of the gauge chart.

Flip color

You can enable color switching based on the values set for minimum and maximum. When the values are larger than 75% of Max then the color of the widget changes to green. When the values are smaller than 25% of Min then the color of the widget changes to red.

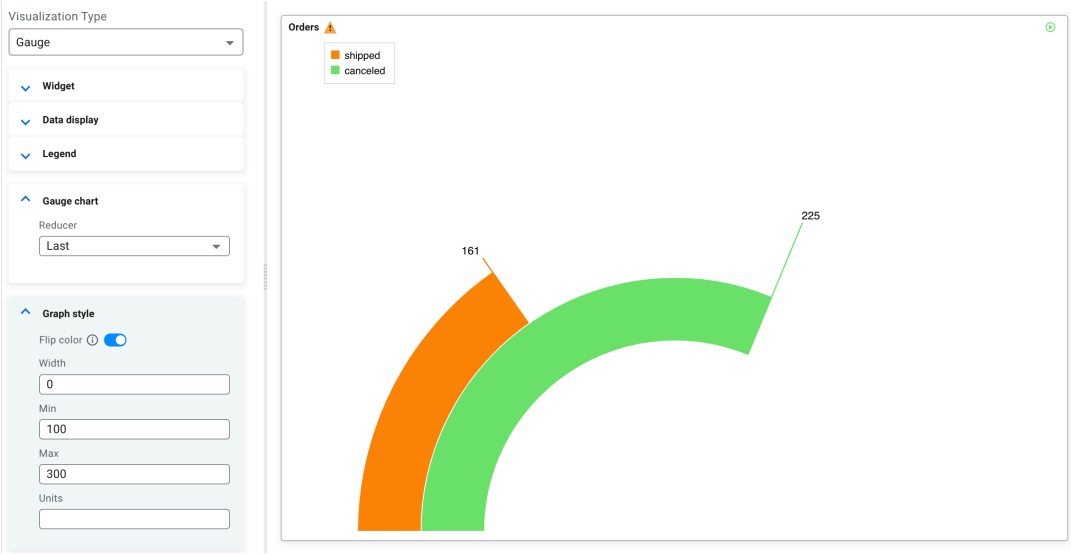

Width

You can increase or decrease the width of the gauges on the widget.

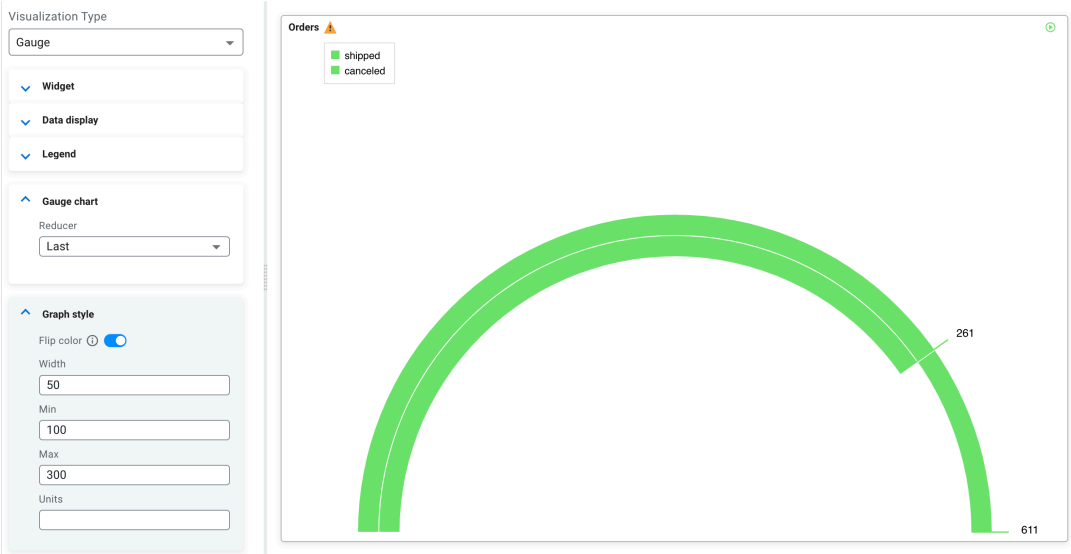

Units

You can specify a unit for the displayed values on the widget.

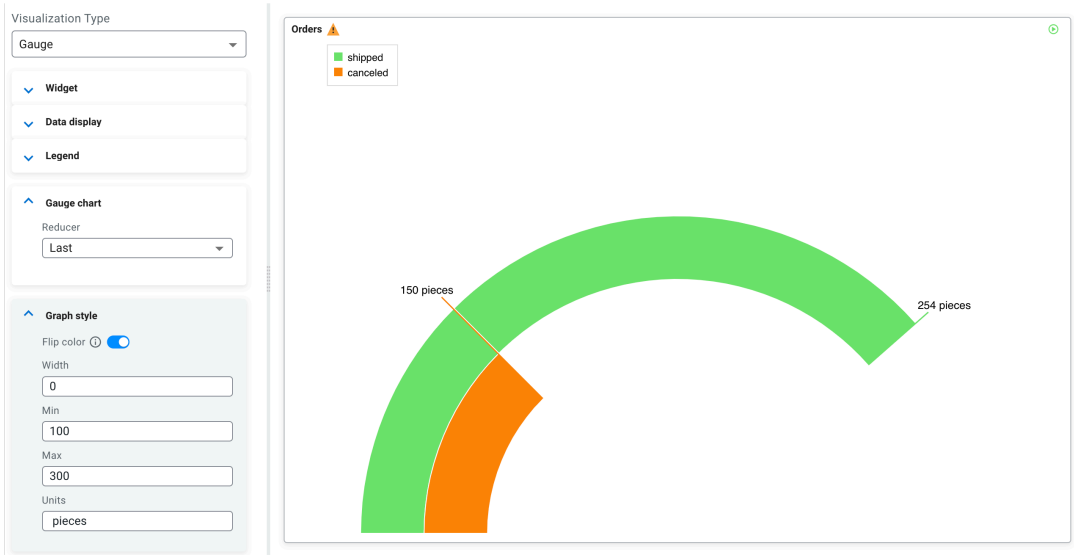

Visualization type: Data grid

You can use the Data grid visualization type when you want to show data in a table format.

Airport weather Restart Polling

airport	temperature	visibility	eventTimestamp
FAPA	-7°C	191m	2023-05-22T22:41:36.088
FAPA	-7°C	191m	2023-05-22T22:41:36.088
SAAG	-23°C	876m	2023-05-22T22:41:49.908
SAAG	-23°C	876m	2023-05-22T22:41:49.908
DIBU	29°C	766m	2023-05-22T22:41:44.142
DIBU	29°C	766m	2023-05-22T22:41:44.142
AGAT	-6°C	783m	2023-05-22T22:41:48.407
AGAT	-6°C	783m	2023-05-22T22:41:48.407
USRR	-3°C	131m	2023-05-22T22:41:38.508
USRR	-3°C	131m	2023-05-22T22:41:38.508

91 to 100 of 100 Page 10 of 10

The following table and query was used for the Data grid widget:

```
CREATE TABLE airport_weather (
  `airport` VARCHAR(2147483647) NOT NULL,
  `temperature` VARCHAR(2147483647) NOT NULL,
  `visibility` VARCHAR(2147483647) NOT NULL,
  `eventTimestamp` TIMESTAMP(3),
  WATERMARK FOR `eventTimestamp` AS `eventTimestamp` - INTERVAL '30' SECOND,
  CONSTRAINT `PK_-991666966` PRIMARY KEY (`airport`) NOT ENFORCED
) WITH (
  'fields.eventTimestamp.expression' = '#{date.past ''30'',''SECONDS''}',
  'connector' = 'faker',
  'fields.temperature.expression' = '#{Weather.temperatureCelsius}',
  'fields.airport.expression' = '#{Aviation.airport}',
```

```
  'rows-per-second' = '100',
  'fields.visibility.expression' = '#{numerify ''###m''}'
);

SELECT * FROM airport_weather
```

The following configurations have been set for the Data grid widget:

Name	Airport weather
Pagination	enabled
Page size	10
Row height	30
Header height	36
Sortable	disabled

You can further customize the graph style of the Data grid and Column configurations:

Pagination and pagination page size

You can enable and disable the pagination of the data on the visualization where the page size provides the number of rows displayed.

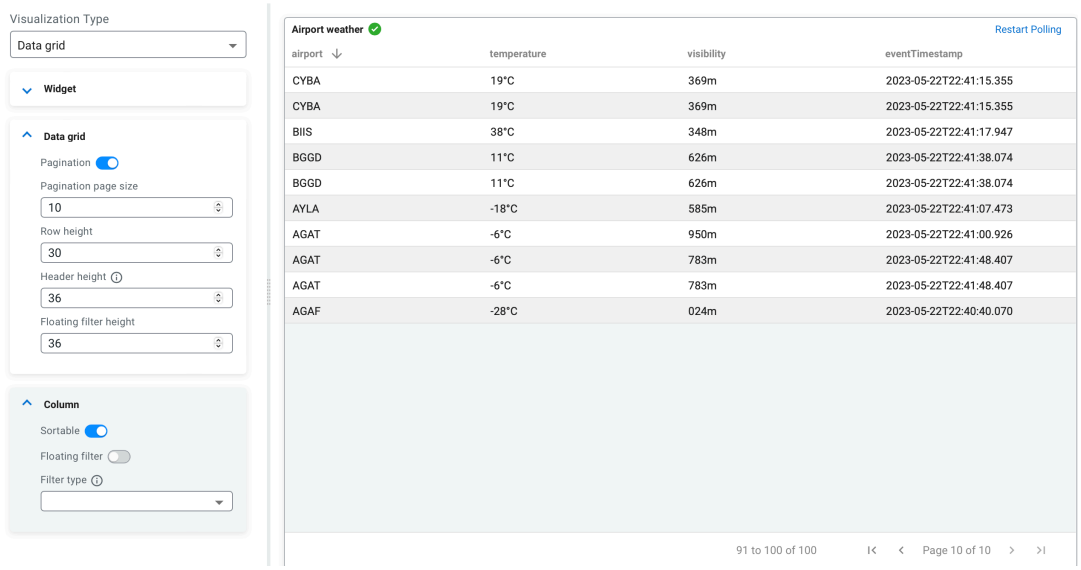

Floating filter and floating filter height:

You can add floating filter to the data grid which enables text, number or date based filtering of the columns. For example, you can set the floating filter to text and filter the data to start with M:

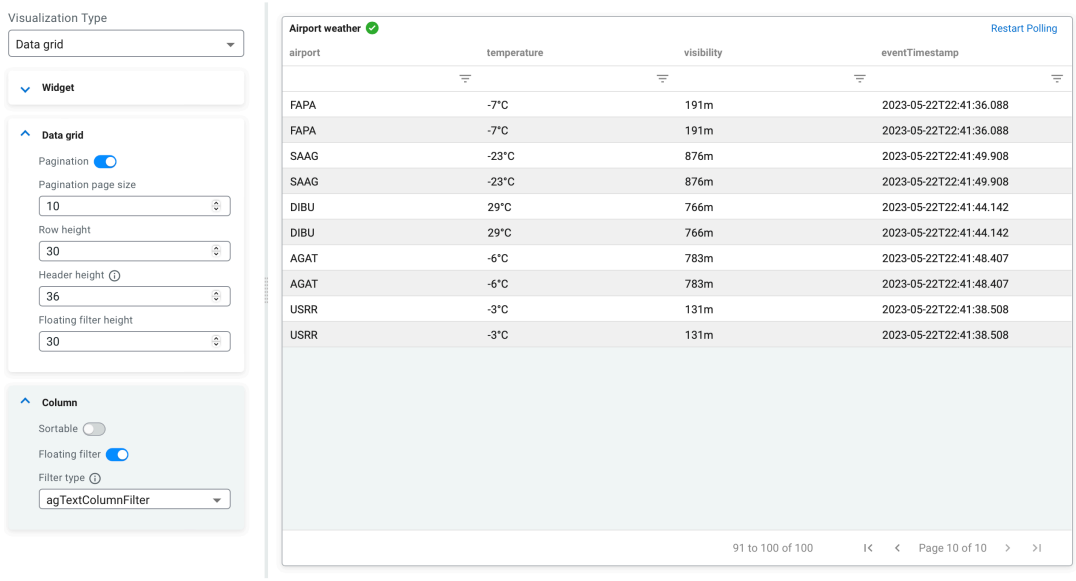

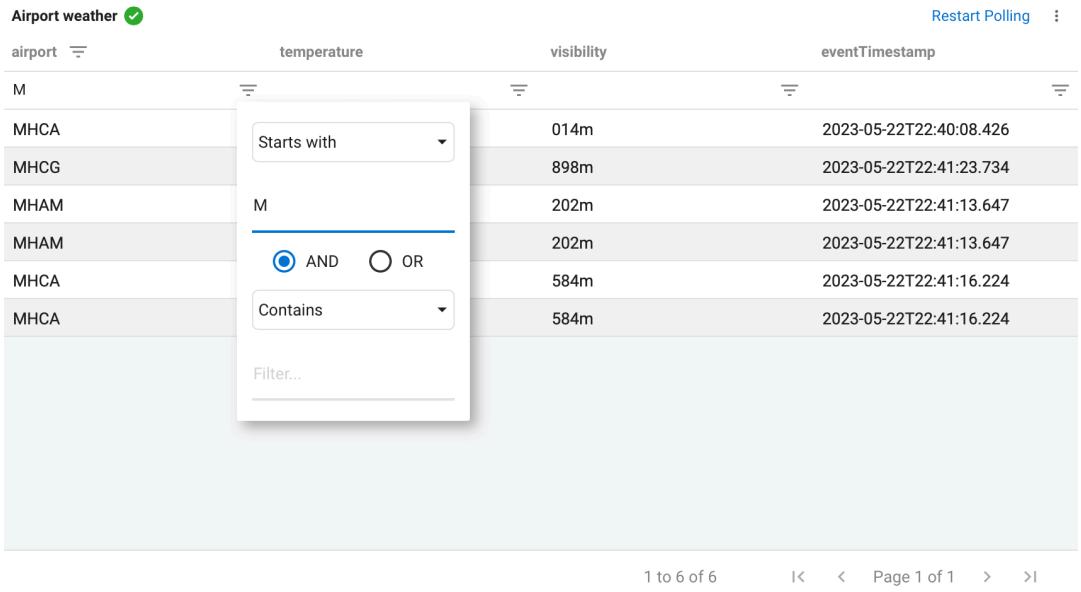

Sortable

You can enable or disable sortable columns for the data grid. You can sort the data in the filtered column by clicking on the column name so the values are in increasing or decreasing order.

Airport weather

Restart Polling

airport	temperature	visibility	eventTimestamp
NFCI	36°C	865m	2023-05-22T22:41:15.471
SGAS	37°C	379m	2023-05-22T22:40:04.549
DAAD	37°C	219m	2023-05-22T22:40:58.494
BIIS	38°C	348m	2023-05-22T22:41:17.947
FAMW	5°C	374m	2023-05-22T22:41:26.716
FAMW	5°C	374m	2023-05-22T22:41:26.716
GANR	5°C	130m	2023-05-22T22:41:28.521
GANR	5°C	130m	2023-05-22T22:41:28.521
GCFV	7°C	071m	2023-05-22T22:40:31.594
USRR	9°C	951m	2023-05-22T22:39:57.255

91 to 100 of 100 Page 10 of 10

Visualization type: Line

You can use the Line visualization type when you want to show how the values changed over a period of time

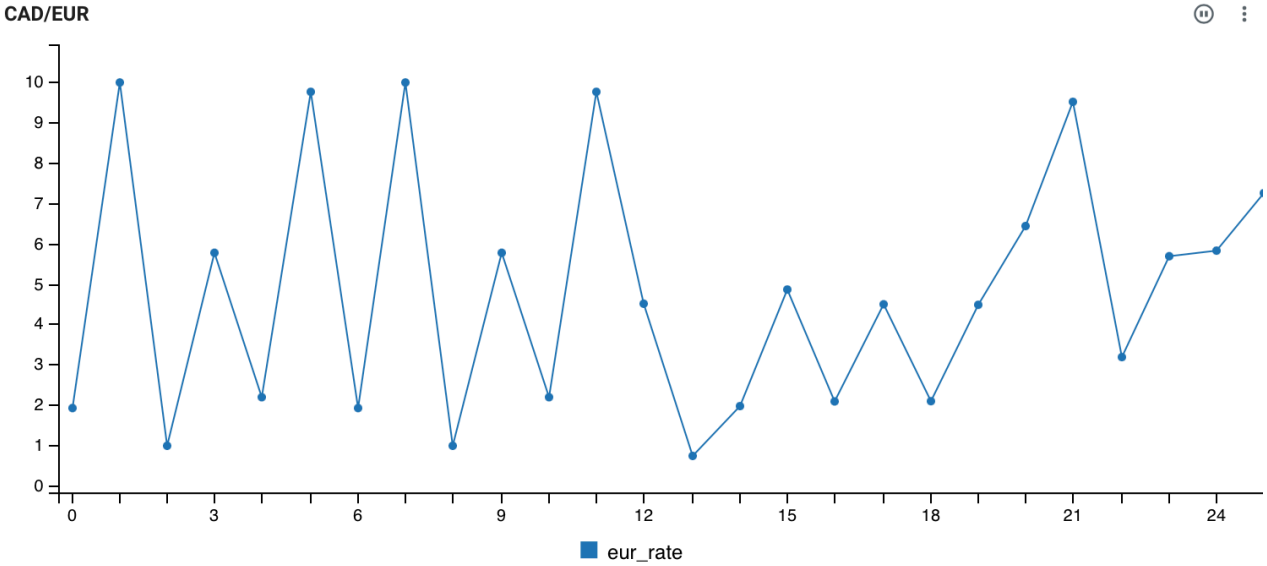

The following table and query was used for the Line widget:

```
CREATE TABLE currency_rates (
  `currency_code` VARCHAR(2147483647),
  `eur_rate` DECIMAL(6, 4),
  `rate_time` TIMESTAMP(3)
) WITH (
  'fields.currency_code.expression' = '#{Currency.code}',
  'fields.rate_time.expression' = '#{date.past ''15'',''SECONDS''}',
  'connector' = 'faker',
  'rows-per-second' = '100',
  'fields.eur_rate.expression' = '#{Number.randomDouble ''4'',''0'',''10''}'
);

SELECT * FROM currency_rates
WHERE currency_code = 'CAD'
```

The following configurations have been set for the Line widget:

Name	CAD/EUR
Data fields	eur_rate
Legend poition	Bottom
Line interpolation	Line
Point radius	2.5

You can further customize the graph style of the Line widget with the following Graph style configurations:

Line interpolation

Determines the line interpolation of the visualization. You have the option to select Step as well as the interpolation as shown in the following example:

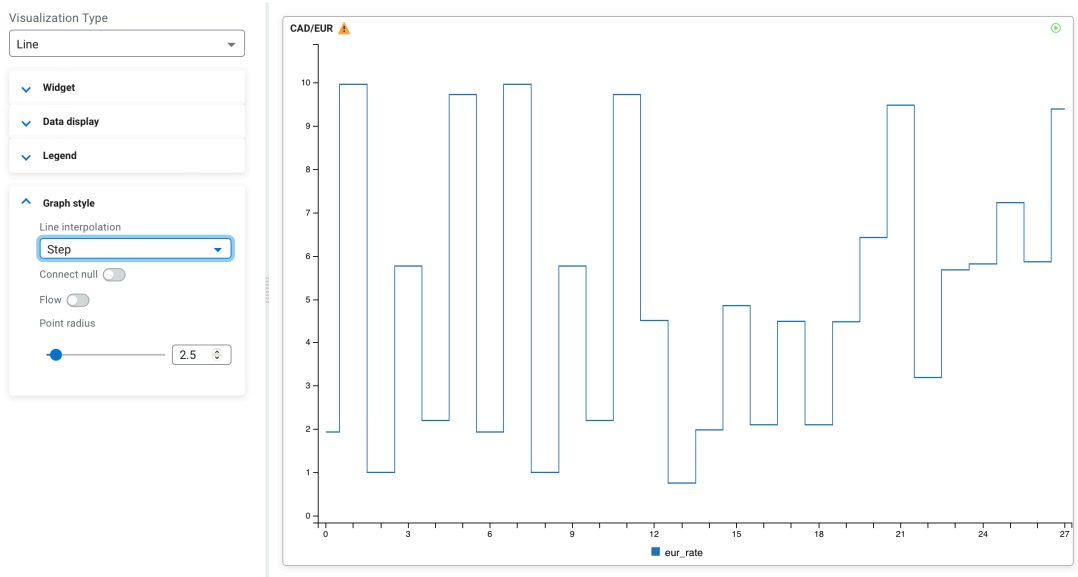

Point radius

You can enable whether to show each data point in line using the Show points toggle. The size of the points can be changed using the Point radius configuration, which is set to 2.5 by default. The following example shows the enabled Show points with 4.0 Point radius:

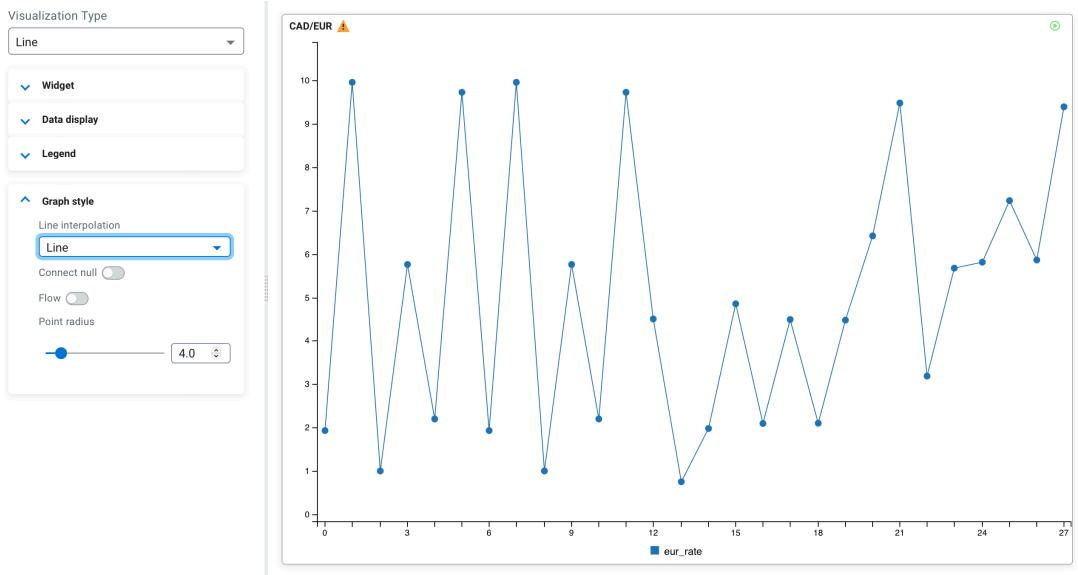

Connect null

Using the connect null configuration, you can set if the null data points are connected or not. If enabled, the region of the null data is connected without any data point. When disabled, the data points are not connected as shown in the following example:

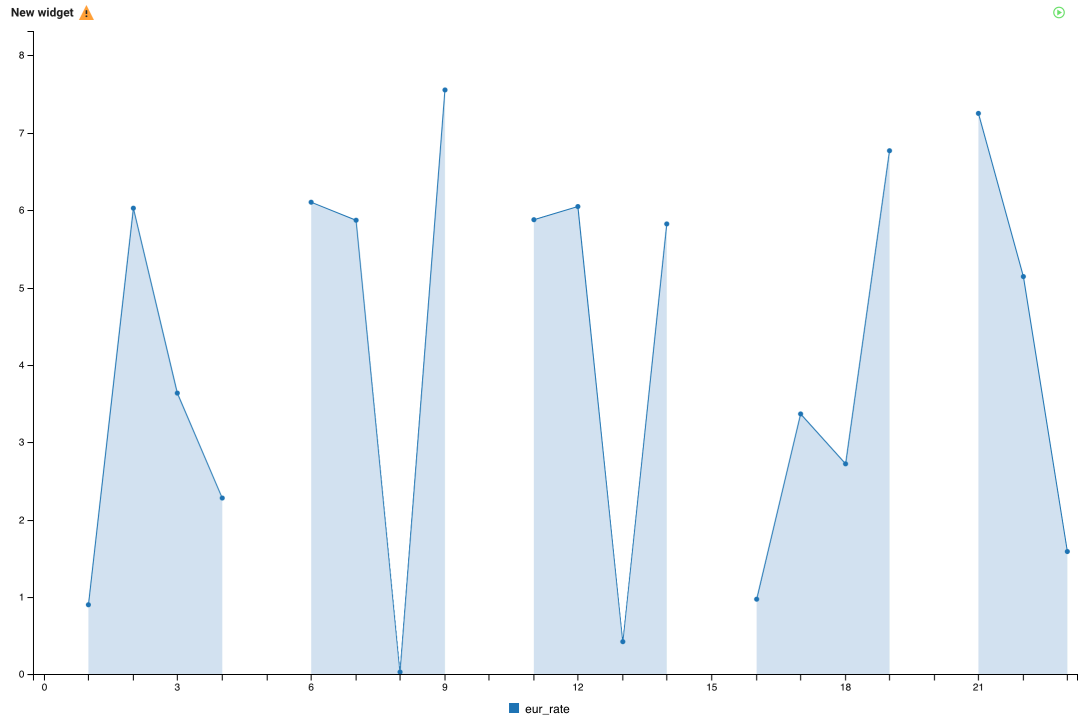

Flow

You can set the movement of the data by enabling or disabling the Flow function, which means the new data points flow to the chart from the right side compared to arriving periodically as processed on the widget.

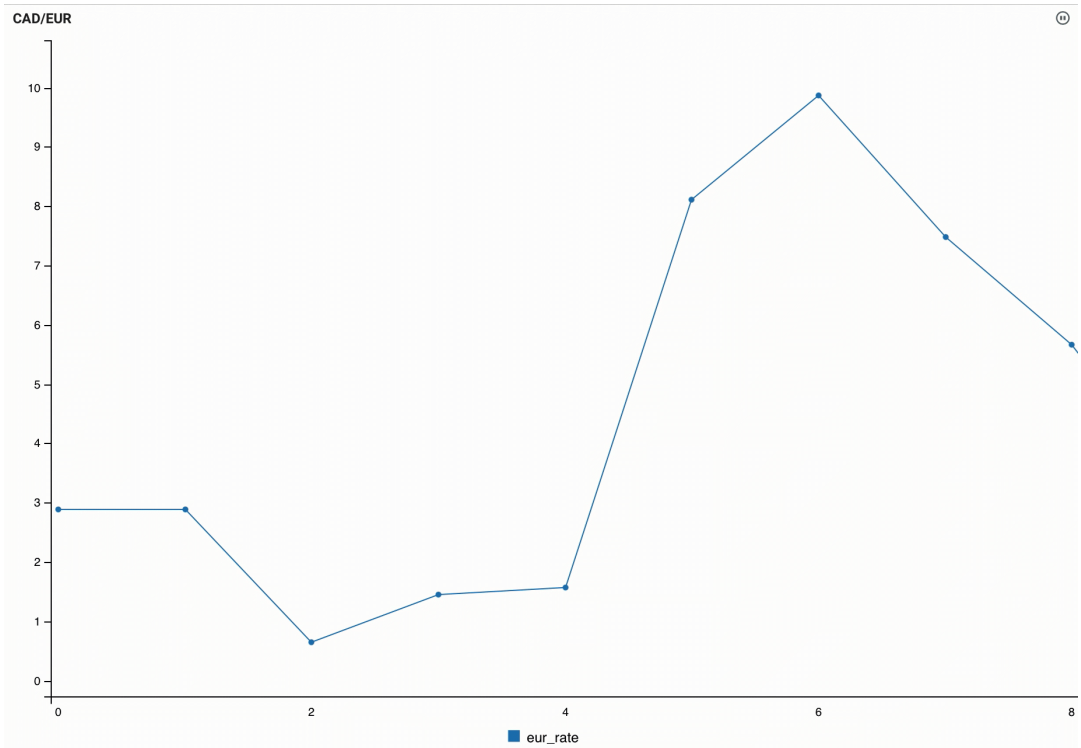

Visualization type: Pie

You can use the Pie visualization type when you want to show data as a percentage compared to the whole dataset where each slice of the pie represents a category of data.

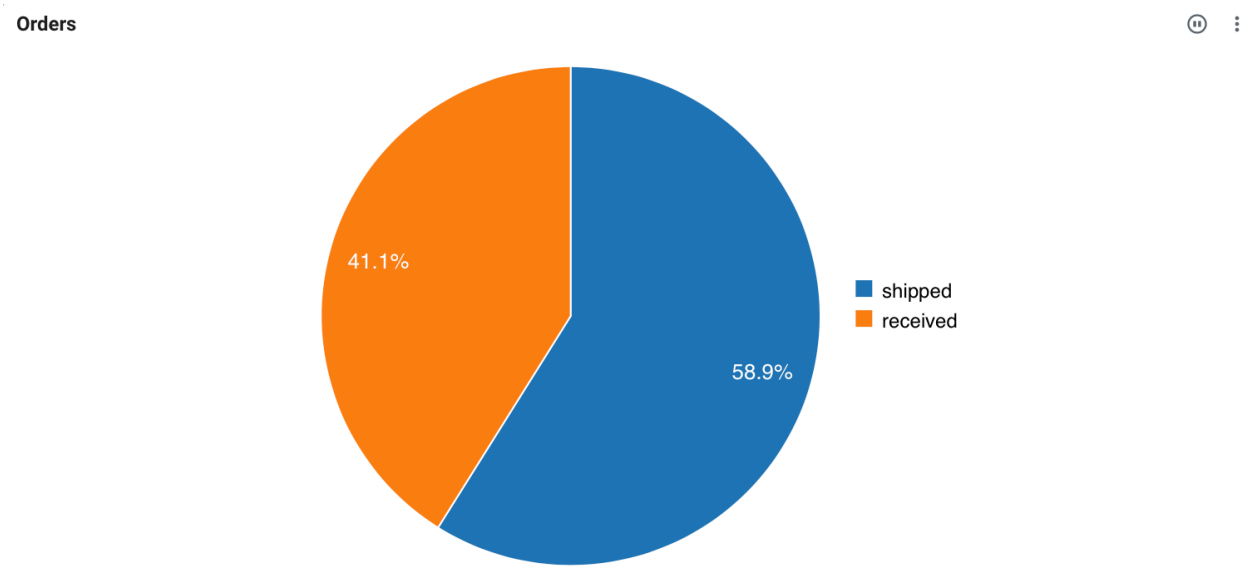

The following table and query was used for the Pie widget:

```
CREATE TABLE orders (
  `order_id` INT,
  `amount` INT,
  `order_time` TIMESTAMP(3),
  `shipped` VARCHAR(2147483647),
  `canceled` VARCHAR(2147483647),
  `received` VARCHAR(2147483647),
  WATERMARK FOR `order_time` AS `order_time` - INTERVAL '15' SECOND
) WITH (
  'fields.order_time.expression' = '#{date.past ''15'',''SECONDS''}',
  'fields.amount.expression' = '#{number.numberBetween ''0'',''100''}',
  'fields.order_id.expression' = '#{number.numberBetween ''0'',''99999999''
}',
  'connector' = 'faker',
  'fields.shipped.expression' = '#{number.numberBetween ''0'',''1000''}',
  'fields.received.expression' = '#{number.numberBetween ''0'',''800''}',
  'fields.canceled.expression' = '#{number.numberBetween ''0'',''500''}',
  'rows-per-second' = '100'
);

SELECT * FROM orders
```

The following configurations have been set for the Pie widget:

Name	Orders
Data fields	shipped, received
Legend poition	Right
Show label	enabled

You can further customize the graph style of the Pie widget with the following Graph style configurations:

Show label

You can enable or disable the labels on the visualization as shown in the following illustration where labels are disabled:

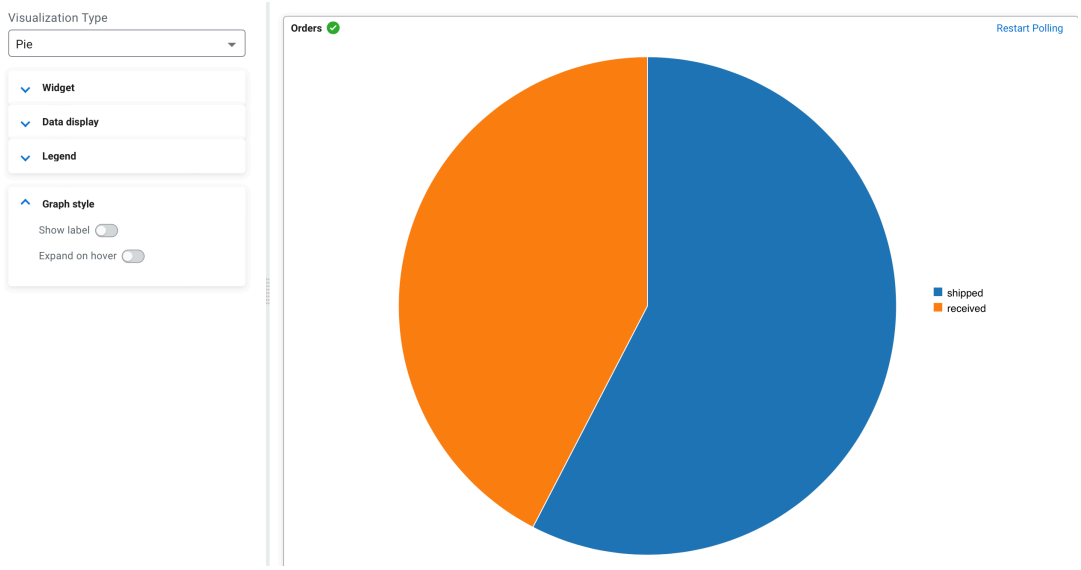

Visualization type: Scatter

You can use the Scatter visualization type when you want to show the correlation between different datasets.

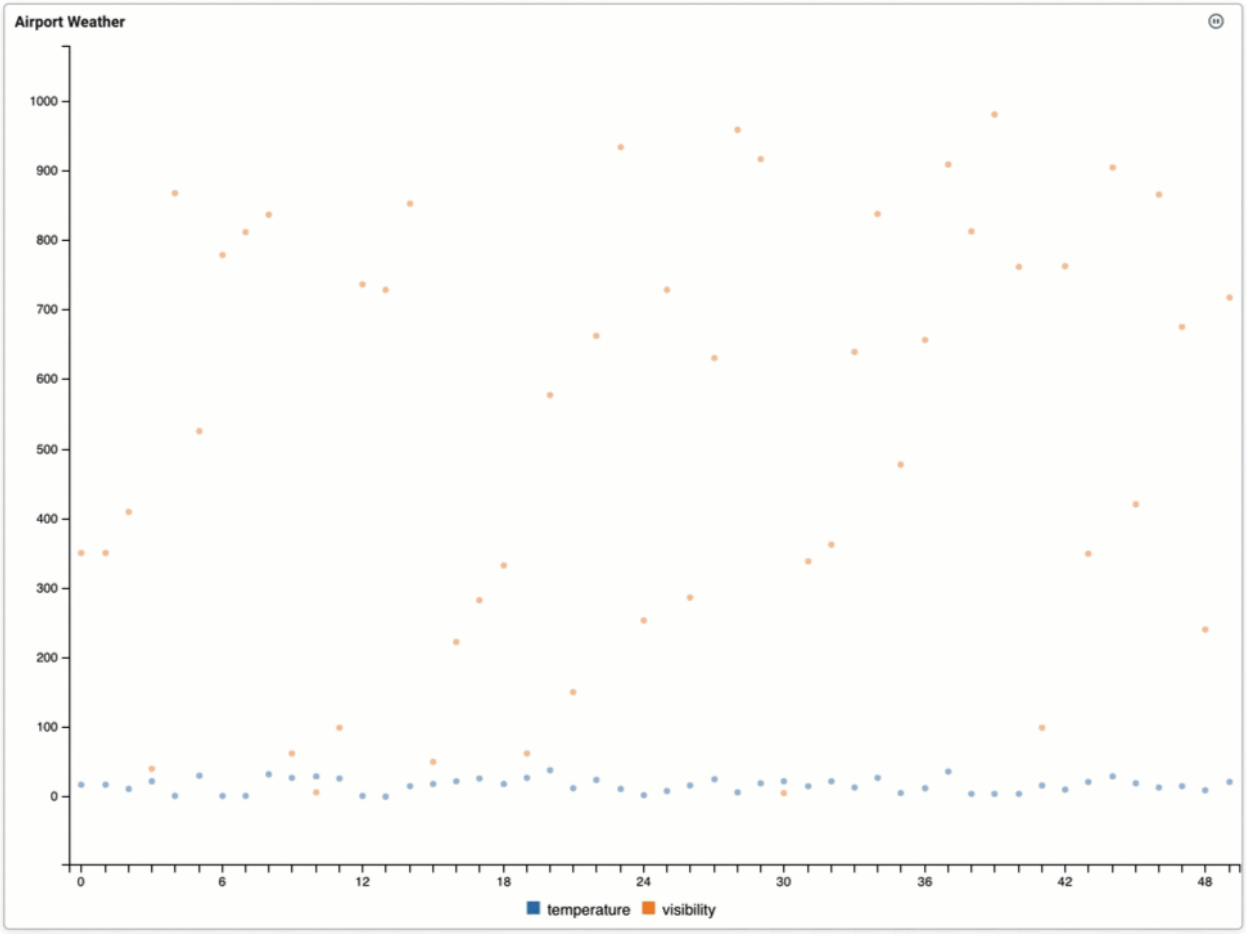

The following table and query was used for the Scatter widget:

```
CREATE TABLE airport_weather (
  `airport` VARCHAR(2147483647) NOT NULL,
  `temperature` VARCHAR(2147483647) NOT NULL,
```

```
  `visibility` VARCHAR(2147483647) NOT NULL,
  `eventTimestamp` TIMESTAMP(3),
  WATERMARK FOR `eventTimestamp` AS `eventTimestamp` - INTERVAL '30' SECOND,
  CONSTRAINT `PK_-991666966` PRIMARY KEY (`airport`) NOT ENFORCED
) WITH (
  'fields.eventTimestamp.expression' = '#{date.past ''30'',''SECONDS''}',
  'connector' = 'faker',
  'fields.temperature.expression' = '#{Weather.temperatureCelsius}',
  'fields.airport.expression' = '#{Aviation.airport}',
  'rows-per-second' = '100',
  'fields.visibility.expression' = '#{numerify ''###m''}'
);

SELECT * FROM airport_weather
```

The following configurations have been set for the Scatter widget:

Name	Airport weather
Data fields	temperature, visibility
Legend poition	Bottom
Point radius	2.5

You can further customize the graph style of the Scatter widget with the following Graph style configurations:

Point radius

You can enable whether to show each data point in line using the Show points toggle. The size of the points can be changed using the Point radius configuration, which is set to 2.5 by default. The following example shows the enabled Show points with 4.0 Point radius:

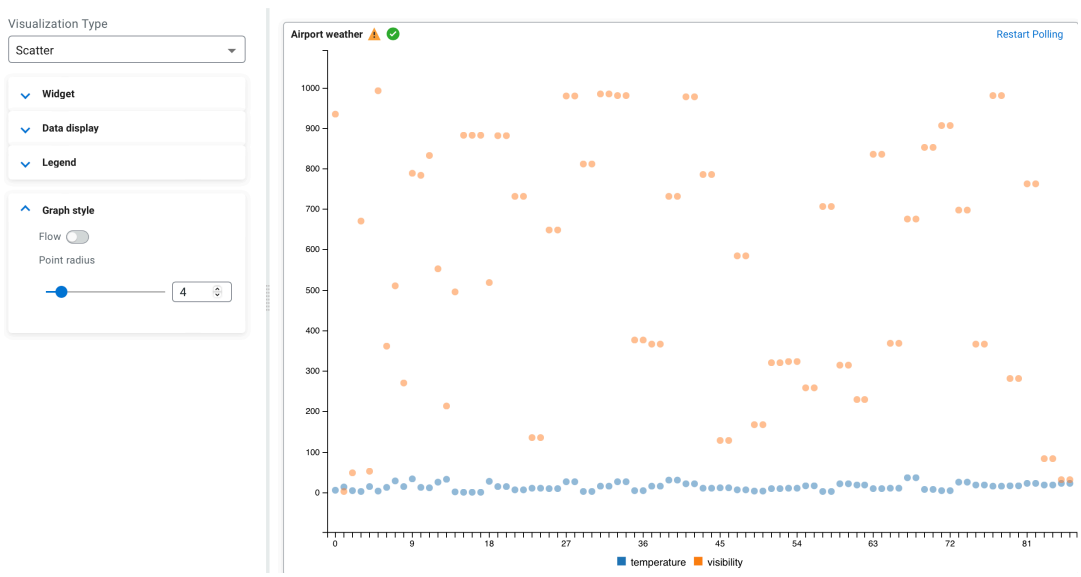

Flow

You can set the movement of the data by enabling or disabling the Flow function, which means the new data points flow to the chart from the right side compared to arriving periodically as processed on the widget.

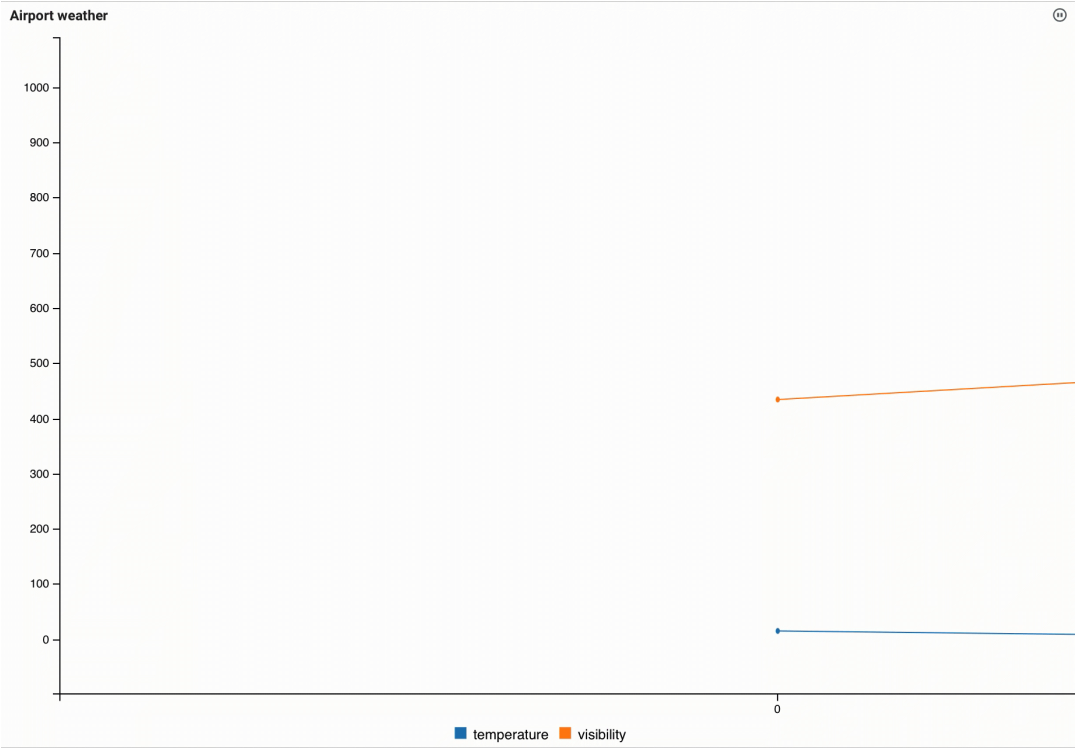

Visualization type: Spline

You can use the Spline visualization type when you want to show how the data changed over a period of time where the trends of change are emphasized.

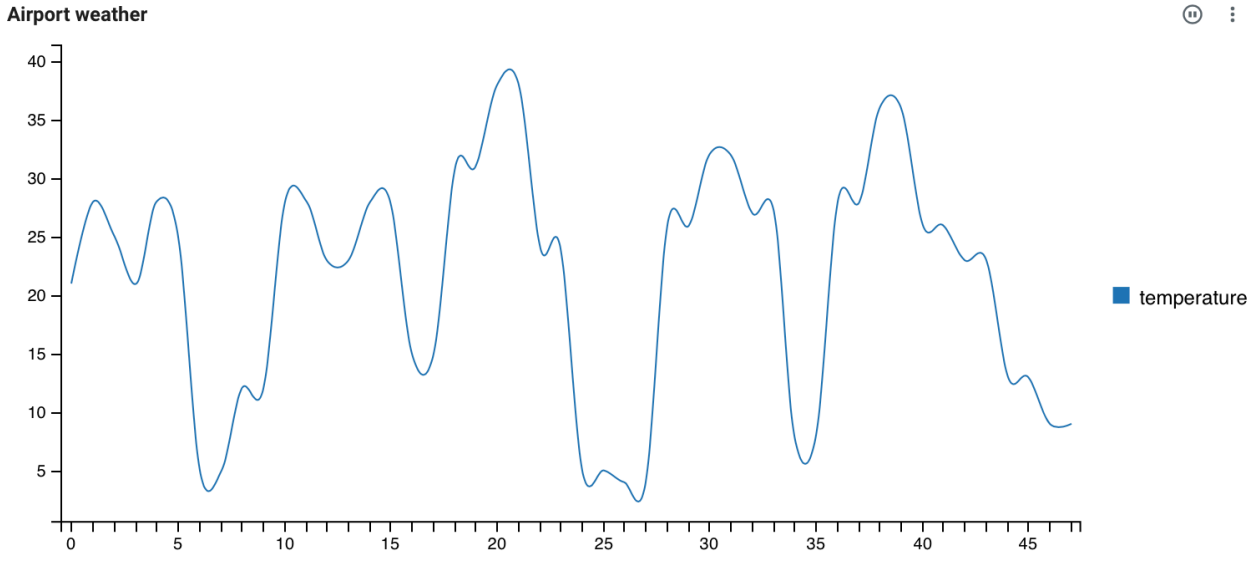

The following table and query was used for the Spline widget:

```
CREATE TABLE airport_weather (
  `airport` VARCHAR(2147483647) NOT NULL,
  `temperature` VARCHAR(2147483647) NOT NULL,
  `visibility` VARCHAR(2147483647) NOT NULL,
  `eventTimestamp` TIMESTAMP(3),
  WATERMARK FOR `eventTimestamp` AS `eventTimestamp` - INTERVAL '30' SECOND,
  CONSTRAINT `PK_-991666966` PRIMARY KEY (`airport`) NOT ENFORCED
) WITH (
  'fields.eventTimestamp.expression' = '#{date.past ''30'',''SECONDS''}',
```

```
  'connector' = 'faker',
  'fields.temperature.expression' = '#{Weather.temperatureCelsius}',
  'fields.airport.expression' = '#{Aviation.airport}',
  'rows-per-second' = '100',
  'fields.visibility.expression' = '#{numerify ''###m''}'
);

SELECT * FROM airport_weather
WHERE airport = 'BIPA'
```

The following configurations have been set for the Spline widget:

Name	Airport weather
Data fields	temperature
Legend poition	Right
Spline interpolation	cardinal
Point radius	0

You can further customize the graph style of the Spline widget with the following Graph style configurations:

Line interpolation

You can specify the Spline interpolation which includes the following list of options:

- Linear, linear-closed
- Basis, basis-open, basis-closed
- Cardinal, cardinal-open, cardinal-closed
- Monotone
- Step, step-before, step-after

Point radius

You can enable whether to show each data point in line using the Show points toggle. The size of the points can be changed using the Point radius configuration, which is set to 2.5 by default. The following example shows the enabled Show points with 4.0 Point radius:

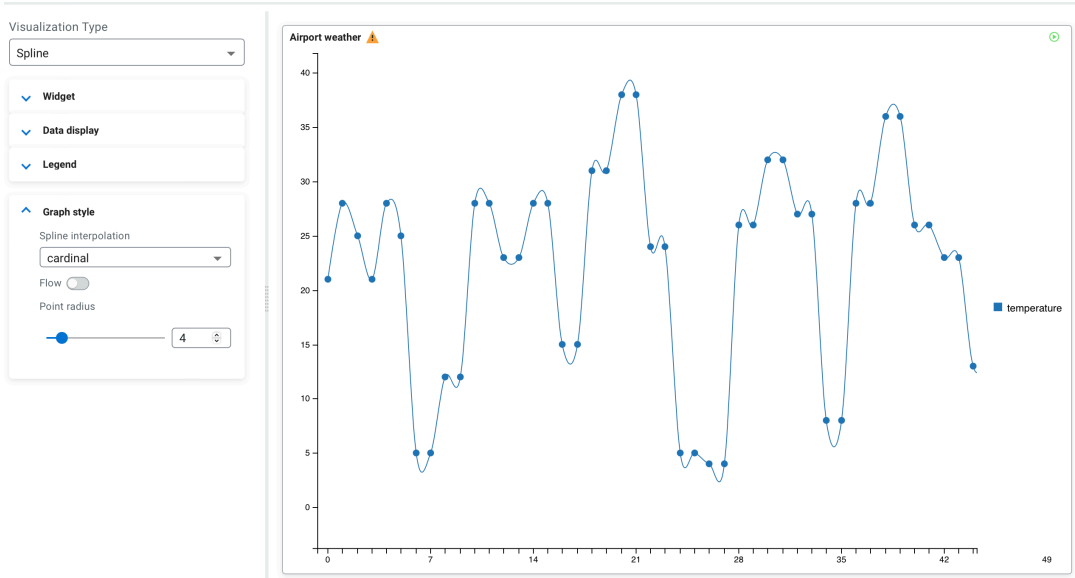

Flow

You can set the movement of the data by enabling or disabling the Flow function, which means the new data points flow to the chart from the right side compared to arriving periodically as processed on the widget.

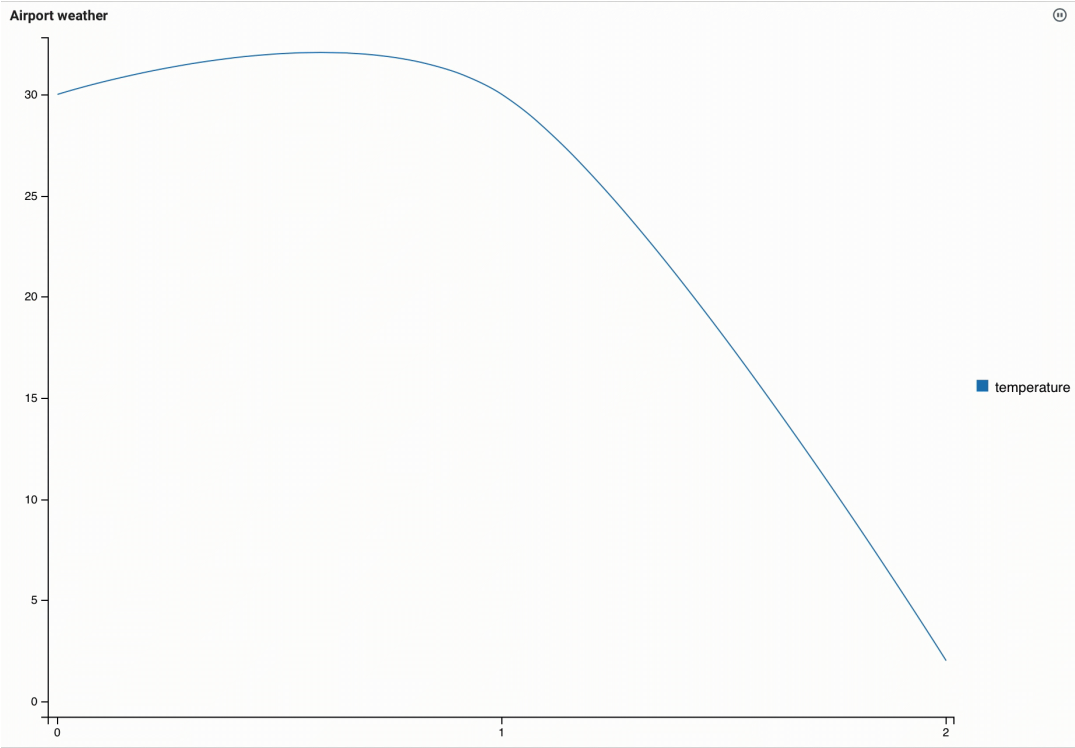

Managing widgets on the Dashboard

After a widget is created, you can customize the layout and control the frequency of the data update from the Dashboard page.

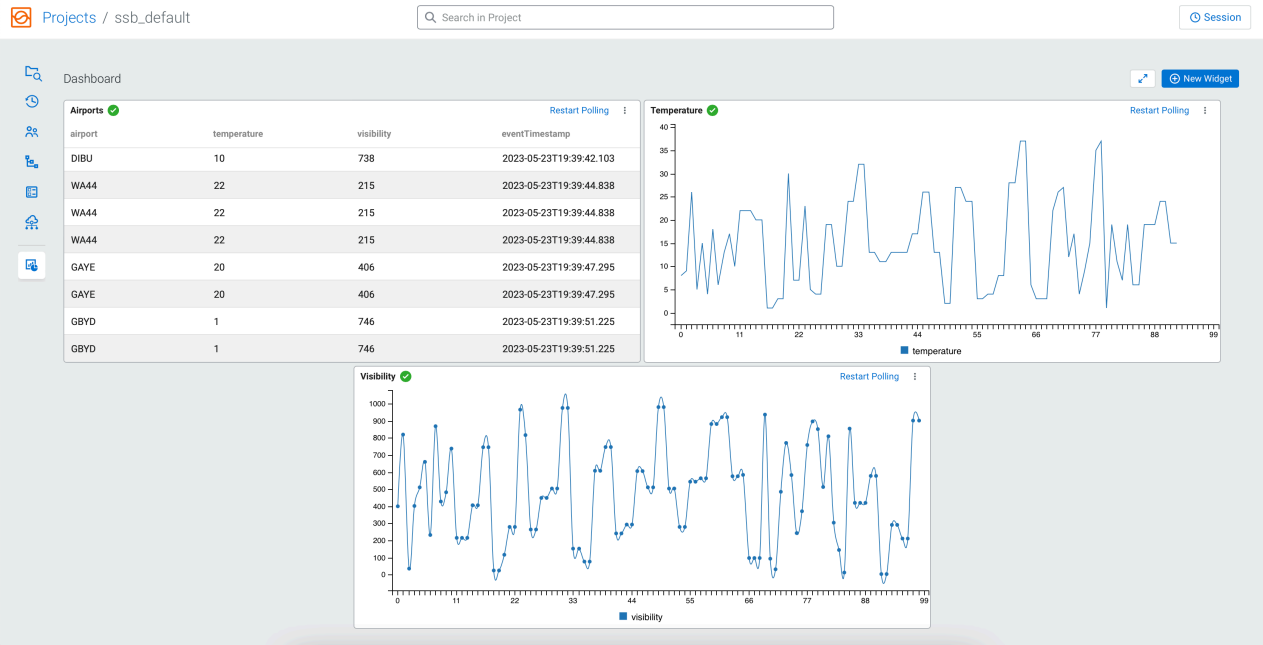

Layout

You can resize and reorganize the widgets on the **Dashboard** page based on your preferences. You can drag and drop the widgets to position them to different places on the **Dashboard**. To resize a widget, you need to click and drag to the preferred size the right bottom corner of the widget.

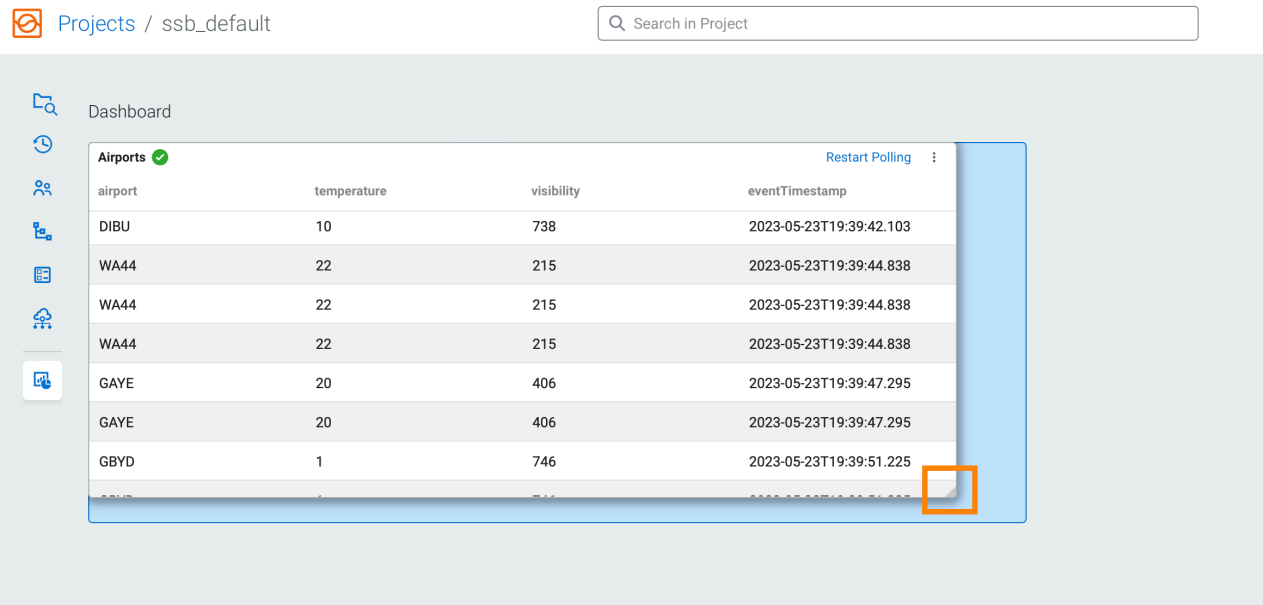

You can also maximize the widget window size to fill out the empty spaces of the Dashboard when only one widget is required for data visualization using the ⋮ Maximize option.

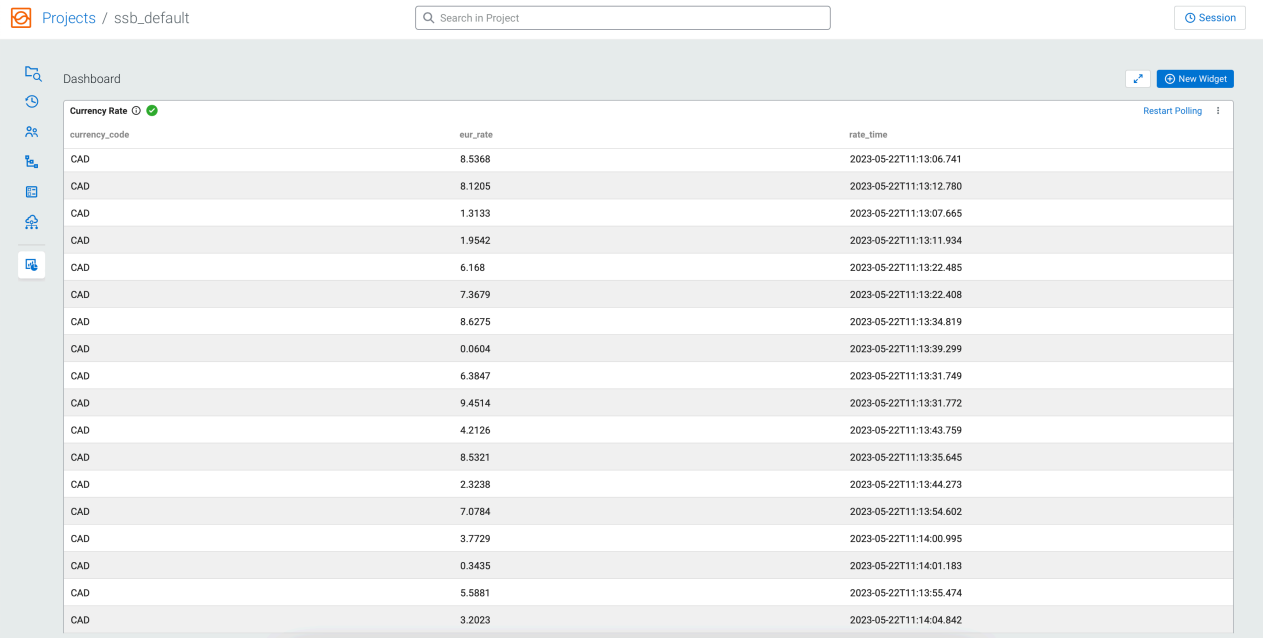

When the original size is required, the size can be reverted back by clicking ⋮ Revert.

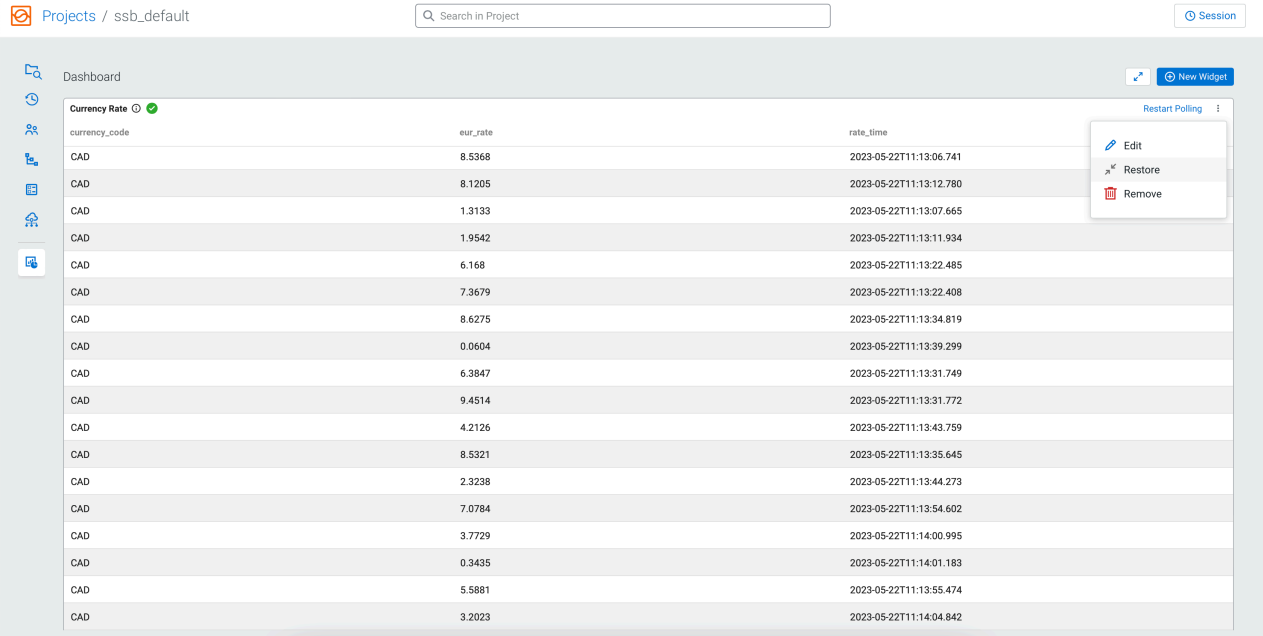

Not only the widgets, but you can also set the **Dashboard** page to a full screen mode using button to present the widgets without the Project Manager and Search bar on the screen.

Data update

You can control the frequency of the data update for a widget by either refreshing it manually between the update cycles, or you can temporarily pause the automatic data update to have a snapshot of the streaming data in that

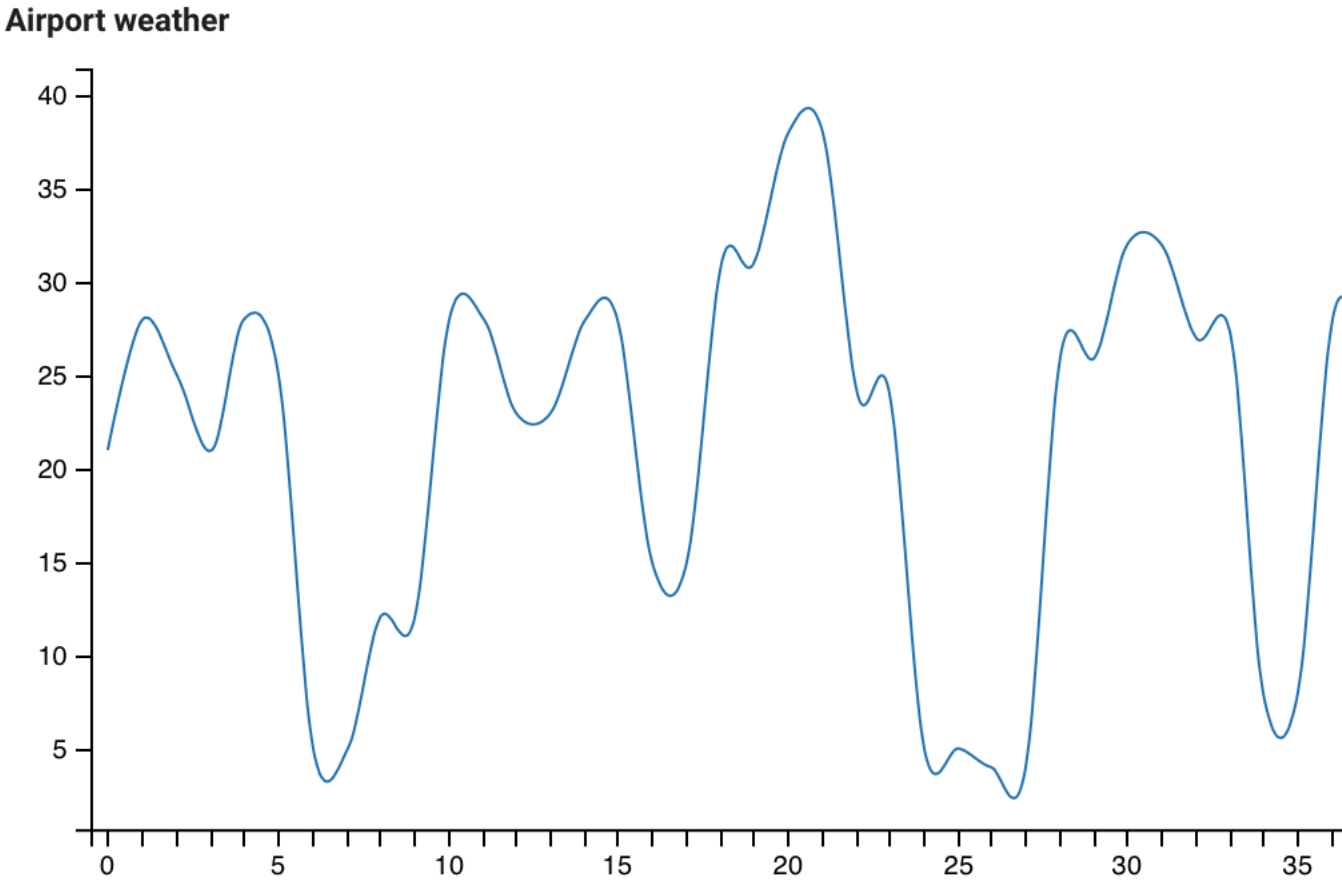

point.

When sampling is in progress, you can use the button to pause the auto update. When the auto update is paused, icon appears next to the name of the widget to indicate that the sampling is stopped. You can resume the update of data by clicking on button.When polling the samples are finished as the configured amount of messages are processed by the widget, icon appears next to the name of the widget. You can restart the polling by clicking Restart Polling.

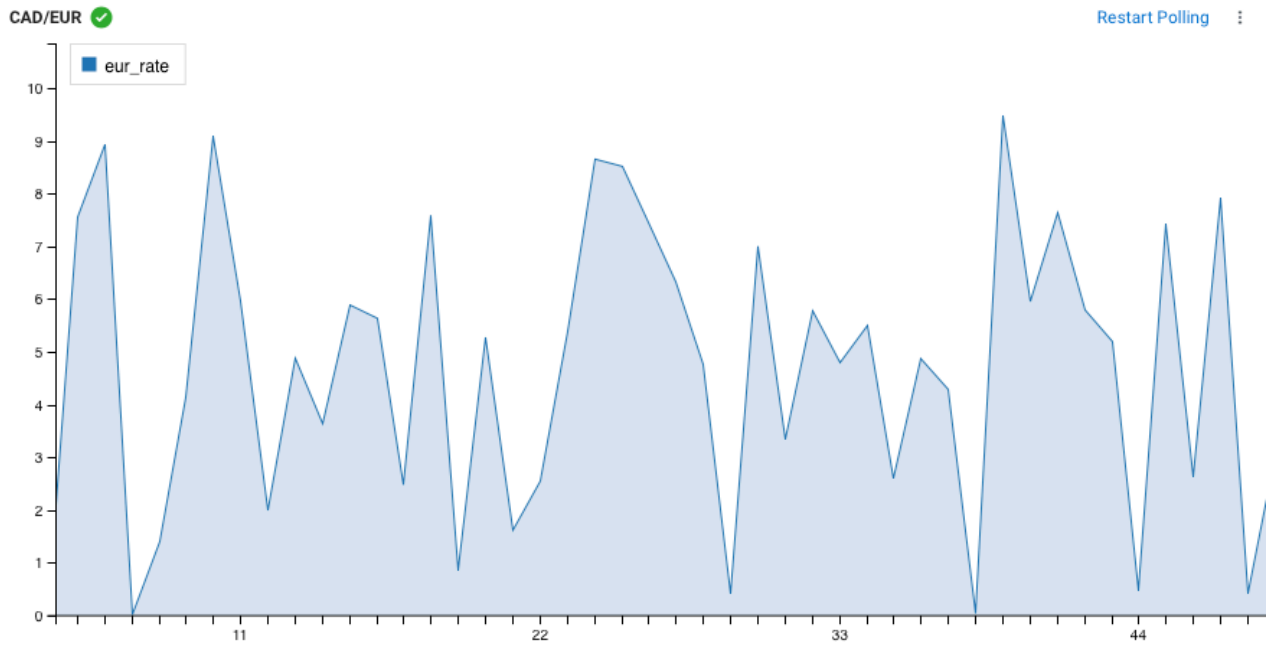

When a Materialized View endpoint is used as a data source, you can manually update the data on the visualization by clicking button individually on the widget.

You can also update every Materialized View widget using the refresh button on the

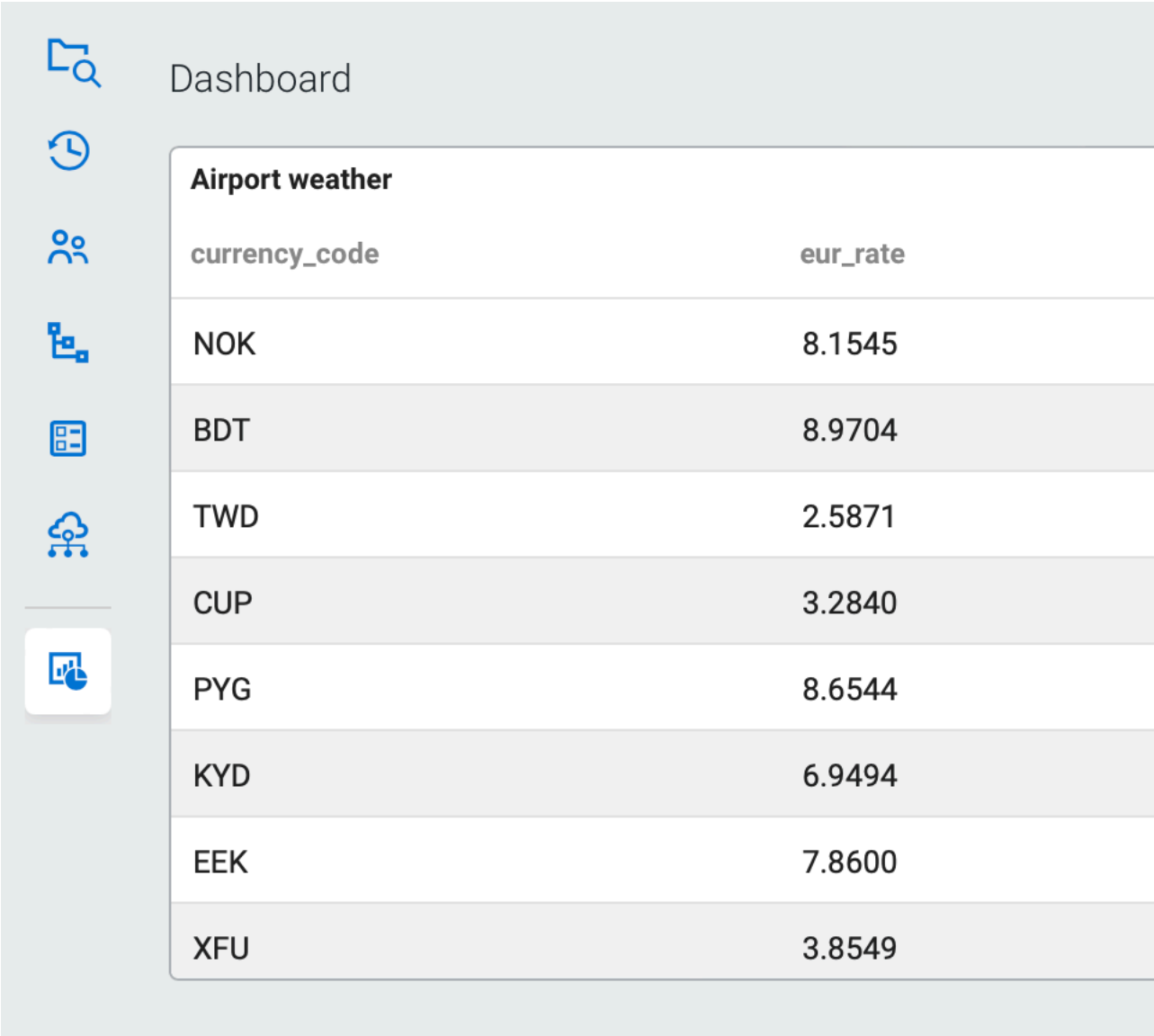

Dashboard.

If the Update Interval of the widget is set to Default, the interval can be specified using the time drop-down and selecting an interval.

Storing configuration

Widgets and Dashboard settings are stored in the local storage of the browser, which enables you to always access the created widgets within a project.

Note: Widgets are not included when exporting a project.

Using SQL job notification

You can set up notifications for SQL jobs using Streaming SQL Console to alert individuals and teams about job failures. The notifications include email and webhook alerts, which you can also group together for easier organization.

You can add job notifications to SQL jobs to send alerts and information when the status of the job is changed. The information in the notification can include the job name, job ID, cluster ID, last exception, and Flink job ID.

You can create the following notification actions for SQL jobs:

Email

Sending notification email to the provided email address

Webhook

Sending notification to a webhook address using POST and PUT

You can also collect the Email and Webhook notifications into a **Group** to organize the notifications. These groups do not trigger any action, but serve as an organizational element.

After creating and adding the notification to a SQL job, anytime a change occurs for that job, a notification is sent to the specified address.

Enabling job notifications

Before you are able to use the job notifications, you need to enable it in Cloudera Manager and based on the notification type configure the email and webhook parameters.

1. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
2. Select Cloudera Manager from the list of services.
3. Select Cloudera SQL Stream Builder from the list of services.
4. Click Configuration.
5. Search for *ENABLE JOB NOTIFICATIONS FUNCTIONALITY*.
6. Enable the job notifications by checking the checkbox.

The following parameters also need to be configured for the Cloudera SQL Stream Builder service in Cloudera Manager based on the type of the notification:

Configuration	Description
Job notifications monitoring interval	Sets the interval of the job monitoring in seconds.
Mail server host for job notifications	The host of the SMTP server for job failure notifications.
Mail server username for job notifications	The username to access the SMTP server for job failure notifications.
Mail server password for job notifications	The password to access the SMTP server for job failure notifications.
SMTP authentication for job notifications	Enable SMTP authentication for job notifications.
StartTLS for job notifications	Use the StartTLS command to establish a secure connection to the SMTP server for job notifications.
Job notifications sender mail address	Sender mail address for job notifications.
Mail server port for job notifications	The port of the SMTP server for job failure notifications.
Job notifications webhook sender parallelism	Number of threads used by the job notification task to call user-specified webhooks when notifying about a failed or missing job.

Note: In case the YARN cluster of a failed job no longer exists, Cloudera SQL Stream Builder attempts to fetch the information about the failed job from the Flink History Server. If information cannot be fetched by the first attempt, Cloudera SQL Stream Builder attempts to fetch the information again based on the interval configured for the History server fetch retry time parameter.

Creating job notifications

After enabling the job notification function, you can create email, webhook and group notifications with Streaming SQL Console.

1. Navigate to Streaming SQL Console.
   a. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Cloudera Streaming Analytics cluster from the list of Data Hub clusters.
   c. Select Cloudera SQL Stream Builder from the list of services.
   d. Click Streaming SQL Console.

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page.
   a. Select an already existing project from the list by clicking the Open button or Switch button.
   b. Create a new project by clicking the New Project button.
   c. Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click ⋮ next to **Job Notifications**.

4. Select from the following options based on which notification type you want to use:

For Email

a. Click New Email Notification .

The **Create Email Notification** window appears.

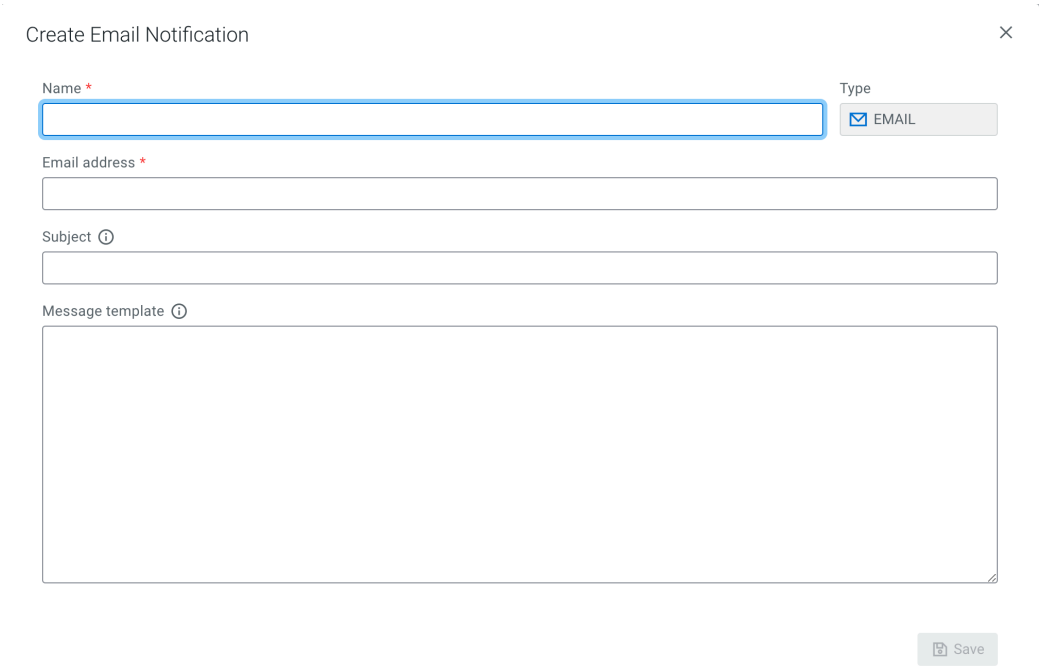

b. Provide a Name for the notification.
c. Add the Email address you want to send the notification.
d. Provide a Subject.
e. Provide a Message.

The message of the Webhook notification must be a valid JSON.

You can add placeholder items to the payload that will be converted into the specific job information. You can add the placeholders in ${placeholder} format, and the following placeholders are available:

- jobName
- jobStatus
- jobStatusDescription
- ssbJobId
- flinkJobId
- clusterId
- lastException

f. Click Save.

For Webhook

a. Click New Webhook Notification .

The **Create Webhook Notification** window appears.

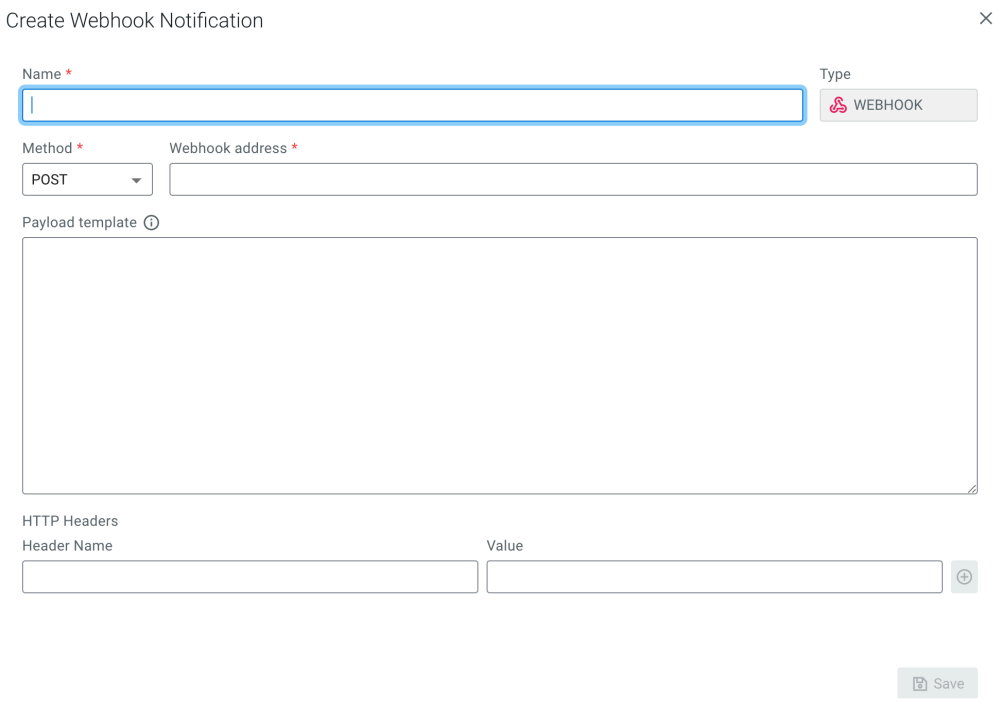

b. Provide a Name for the notification.
c. Select between POST or PUT method.
d. Provide a Webhook address.
e. Provide a Payload to the webhook.

Warning: The payload must be a valid JSON object.

You can add placeholder items to the payload that will be converted into the specific job information. You can add the placeholders in ${placeholder} format, and the following placeholders are available:

- jobName
- jobStatus
- jobStatusDescription
- ssbJobId
- flinkJobId
- clusterId
- lastException

f. Add **HTTP Headers** if needed by specifying the Header Name and Value.
g. Click Save.

For Groups

a. Click New Group Notification .

The **Create Group Notification** window appears.

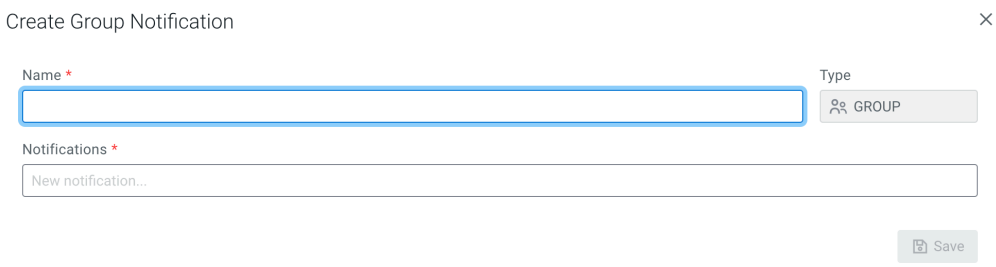

b. Select from the already created Email and Webhook notifications that you would like to add to this group.

c. Click Save.

Enabling notifications for a SQL job

After creating a job notification, you can enable it for a selected job from the **Job Settings** window.

1. Select Job Settings from the **SQL Editor**.
2. Click Job Notifications.
3. Search for the name that you have provided when creating the notification.
4. Click on the notification.

   The selected notification is added to the job and the notification is sent with the information provided in the message template.

Using Cloudera SQL Stream Builder REST API

You can use the REST API to monitor, manage and configure the SQL Stream jobs with GET and POST HTTP methods. You can use theCloudera SQL Stream Builder REST API in the command line or use a REST API Tool.

The following HTTP methods are available for SCloudera SQL Stream Builder:

- GET to query information about the specified endpoint
- POST to create resources for the specified endpoint
- PUT to update existing resources for the specified endpoint as a whole
- PATCH to partially update existing resources for the specified endpoint
- DELETE to remove objects from the specified endpoint

The REST API Reference document contains the available endpoints for Cloudera SQL Stream Builder. However, you can also reach the Cloudera SQL Stream Builder REST API reference document from on the main menu of Streaming SQL Console.

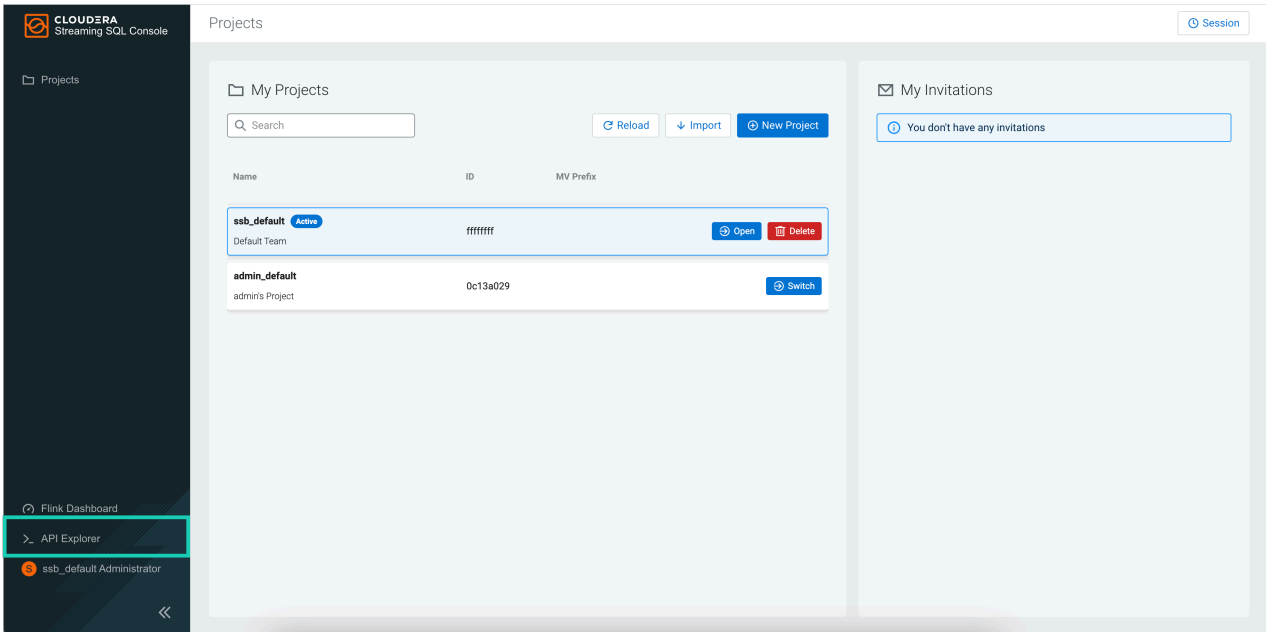

You can use the Cloudera SQL Stream Builder REST API with Command Line Interface (CLI), but you can also use a REST API Tool, for example Postman.

The Streaming SQL Engine API details the following operations for SQL Stream Builder and Flink:

- Heartbeat
- Admin Operations
- User Operations
- User Keytab Operations
- Data Source Operations
- Table Operations
- Project Operations
- Project Environment Operations
- Project Sync Operations
- Project Invitation Operations
- Project Permission Operations
- SQL Operations
- SSB Session Operations
- Job Operations
- Sampling Operations
- Flink Job Operations
- Flink Session Cluster Operations
- Artifact Operations
- UDF Artifact Operations
- UDF Operations
- API Key Operations
- Connector Operations
- Data Format Operations
- Diagnostic Operations

The Cloudera SQL Stream Builder project-related endpoints can be used with specifying which project the request is submitted to by adding the project ID parameter to the endpoint. This means that even though a project is selected as the active project on Streaming SQL Console, you can submit any request to a different project to which you have access to. The REST API user permissions are the same for a project as it is configured on the Members page in

a project. The requests that are submitted using a specific project ID are submitted as a new session that does not interfere with the already existing user sessions in SSB.

Accessing the project ID

You can find the project ID on the homepage of Streaming SQL Console:

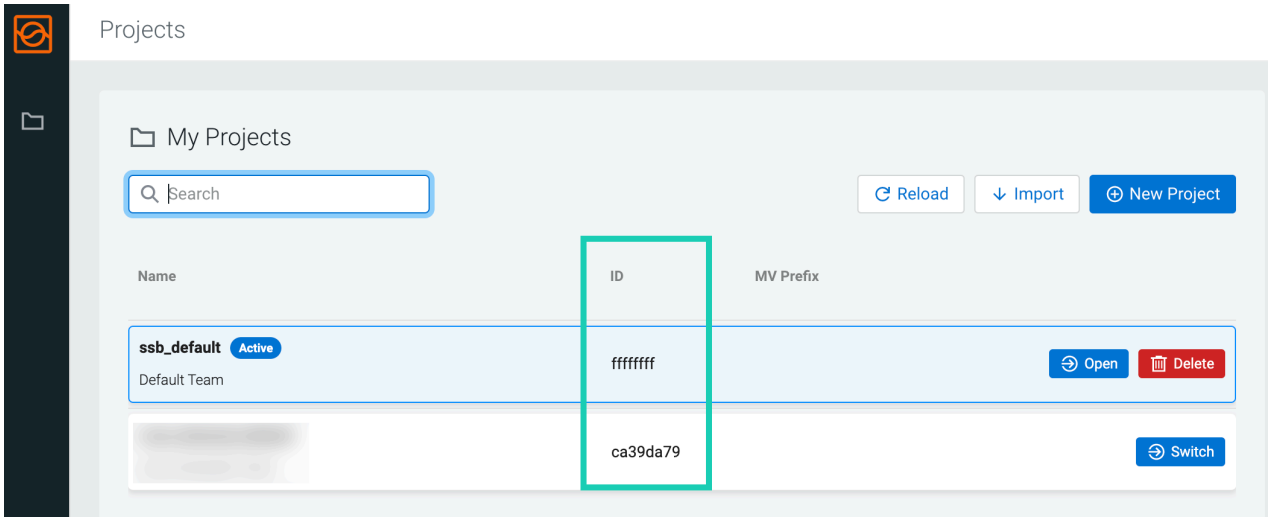

You can also use the GET /api/v2/projects call in CLI to list all projects that the user has access to.

Obtaining endpoint URLs

You need to obtain the Endpoint URL of the Streaming SQL Engine API that is needed for the API calls using the following steps:

1. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
2. Select the Cloudera Streaming Analytics cluster from the list of Cloudera Data Hub clusters.
3. Click Endpoints on the Cloudera Data Hub cluster details page.
4. Copy and note down the Streaming SQL Engine API endpoint.

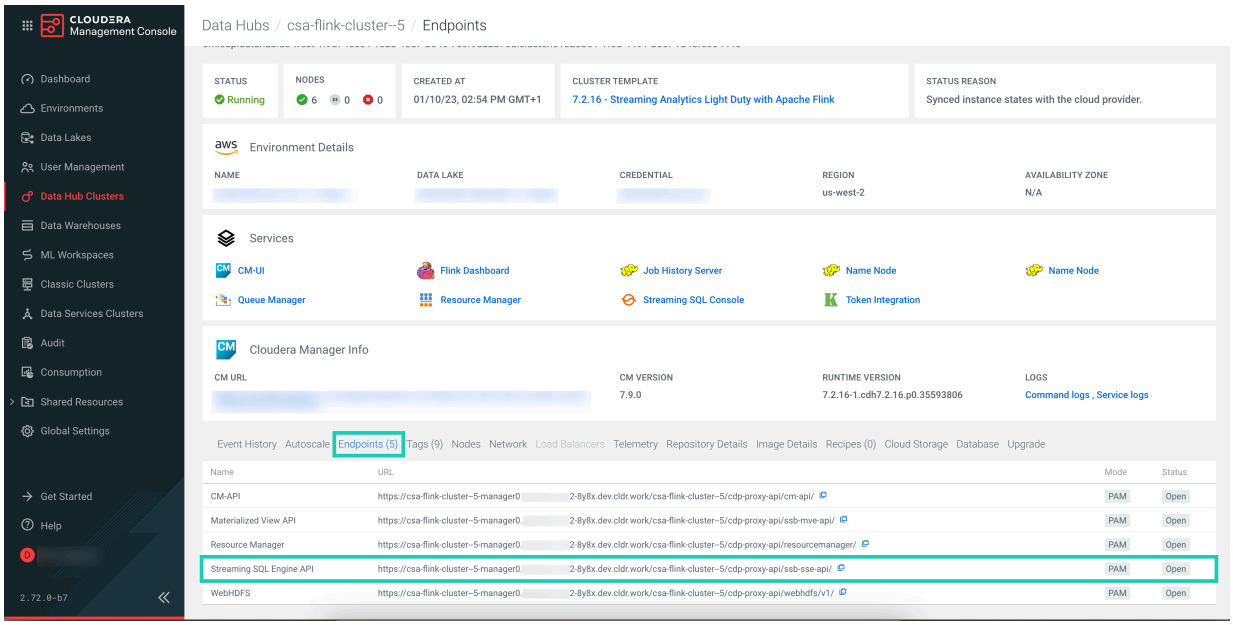

Using REST API with CLI

When using the Cloudera SQL Stream Builder REST API with CLI Tool, you need to create the POST or GET commands with curl, and also include the Manager host FQDN of the Cloudera SQL Stream Builder cluster, the name of the Cloudera Data Hub cluster, and the required operation for the endpoint. If necessary, you can include the Streaming SQL Engine port. The submitted commands return the information, or complete the process you have requested and display the status in the CLI.

The following examples show a secured GET method to list the SQL jobs:

```
$ curl -u: --negotiate '[***WORKLOAD_USERNAME***]:[***WORKLOAD_PASSWORD***]'
 \
    '[***ENDPOINT_BASE_URL***]/api/v1/jobs'
```

Using REST API Tool

When using REST API Tools, make sure that the base URL in the HTTP request is similar to the following example:

```
[***ENDPOINT_BASE_URL***]/api/v1/jobs
```

The base path has to be the same for all endpoints. In terms of authentication, make sure you set “Basic Auth”, and use your workload username and password as credentials.

Monitoring SQL Stream jobs

You can use the Streaming SQL Console to review the status, properties and log of your SQL Stream jobs executed in Cloudera SQL Stream Builder. Using the Flink Dashboard, you can also monitor the Flink job that is submitted when you execute a SQL query.

Using the Streaming SQL Console

When using the **Jobs** tab in Streaming SQL Console to monitor your SQL jobs, you can review the ID, the Name, the Start time, the State of the submitted jobs, and the User who submitted the SQL job.

1. Click ⋮ next to **Jobs**.
2. Select Manage.

   By default, all of the jobs are displayed on the **Jobs** tab.

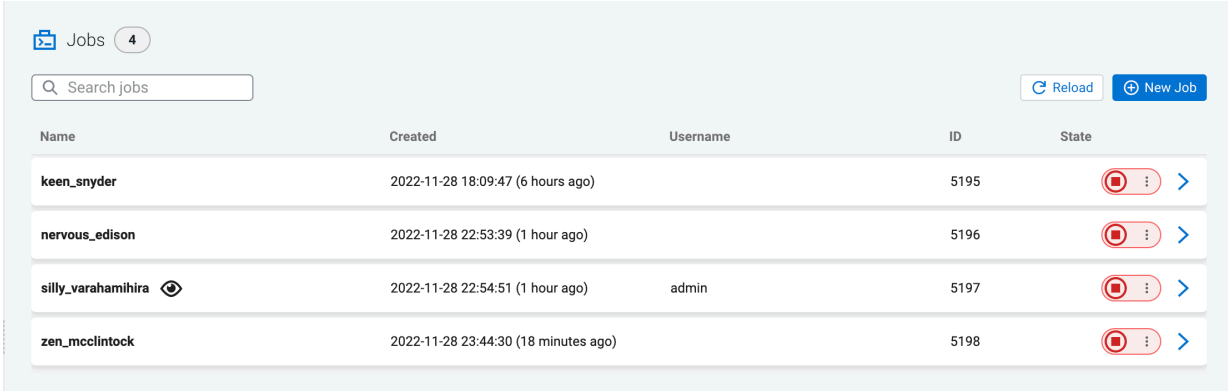

3. Click ⋮.

4. Select Events.

   The **Events** window appears detailing the Flink job information that can be further filtered based on the log level settings.

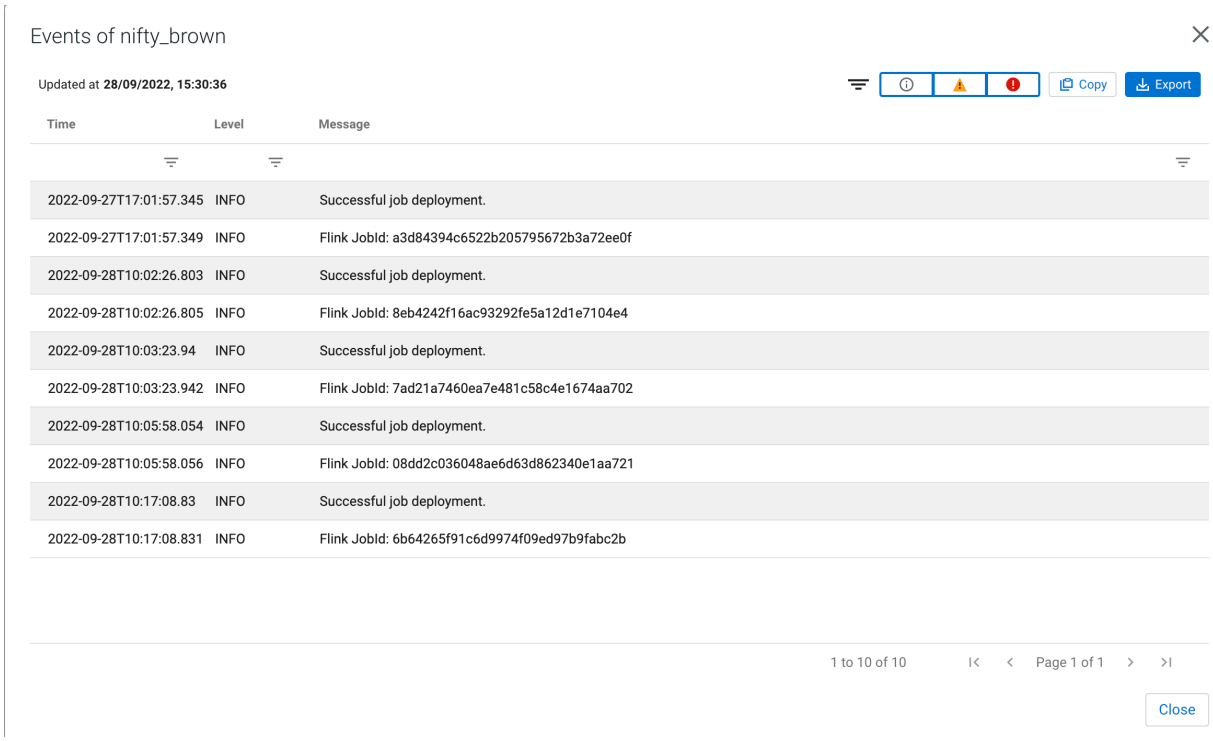

History of SQL queries

You can review and reuse the SQL queries that were previosuly executed on the **SQL History** view of the Project Manager. When you click on one of the SQL queries, it is automatically imported to the SQL window for execution. You can filter the SQL queries by the time they were last run or by the user who run them.

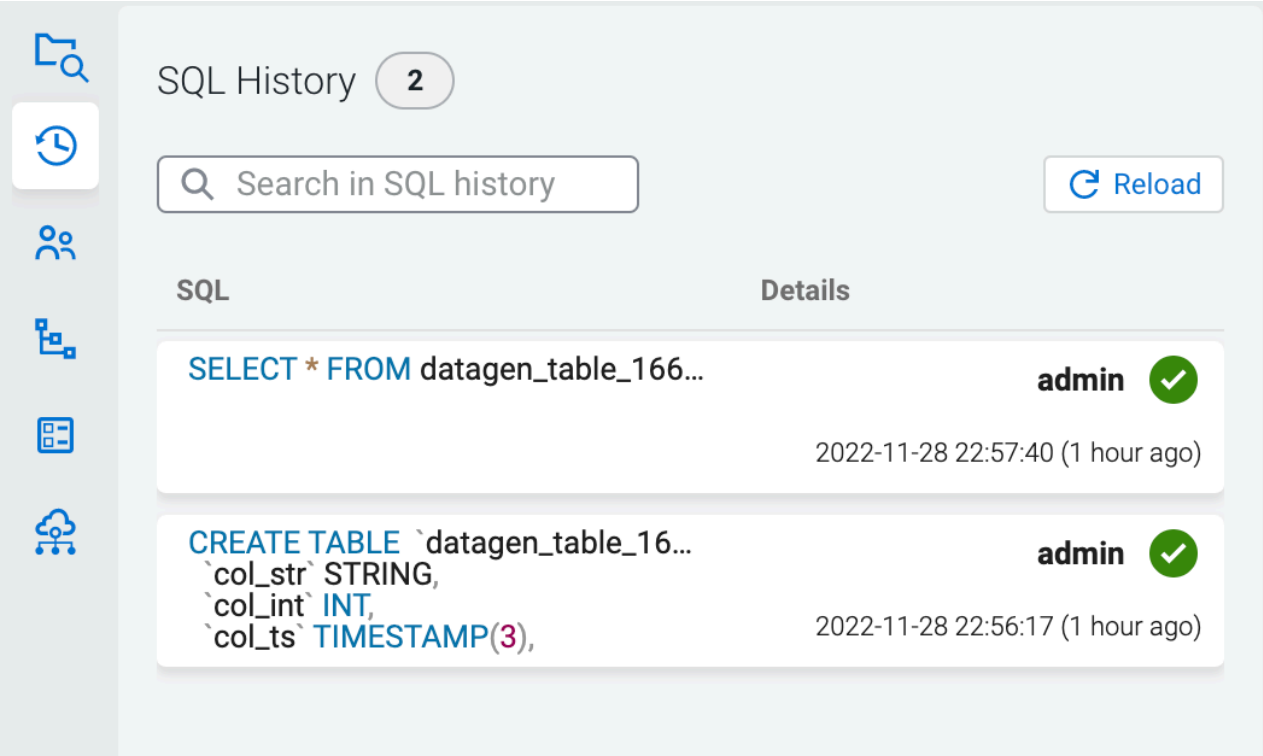

Using the Flink Dashboard

You can also monitor your running SQL jobs using the Flink Dashboard. You can easily reach the Flink Dashboard from the main menu of Streaming SQL Console or from the SQL Editor.

Collecting diagnostic data

You can collect diagnostic data from Cloudera SQL Stream Builder and Flink using Cloudera Manager, and send it to Cloudera Support for easier troubleshooting.

When using Cloudera SQL Stream Builder and Flink, Cloudera Manager automatically collects the usage and diagnostic data from the services. The collected data is analyzed to further improve the software, and to help you in support cases when you encounter an error. You can configure the frequency of data collection, and you can also manually trigger sending the diagnostic data to Cloudera.

For more information about the diagnostic data collection and configuration, see the Cloudera Manager documentation.

The following telemetry data is collected when using Cloudera SQL Stream Builder and Flink:

- Number of users
- Number of jobs
- Number of registered data sources and connectors
- Number of Materialized Views
- Number of API keys
- Number of tables
- Number of catalogs
- Total tasks slots in use
- Log files from Streaming Engine and Materialized View Engine
- Log files from YARN to collect Flink Job and Task Manager processes

SSB metadata collection using Atlas

As Cloudera SQL Stream Builder is built on Flink, you can use the Apache Atlas and Flink integration, and collect the metadata of your Flink jobs submitted from it. Using Atlas, you can find, organize and manage different assets of data about your Flink applications and how they relate to each other.

To use Atlas with Cloudera SQL Stream Builder, you need to create entity type definitions for Flink in Atlas, and enable the metadata collection in Cloudera Manager for Flink.

For more information, see the Flink metadata collection using Atlas.

Flink SQL

As Flink SQL is used in Cloudera SQL Stream Builder, you can execute the supported Flink DDL, DML and query statements directly from the SQL window in Streaming SQL Console.

Flink DDL

Flink SQL supports Data Definition Language (DDL) statements to create, modify and remove objects within a data structure.

The following table summarizes the supported DDL statements in SQL Stream Builder:

DDL	Description	Option
CREATE TABLE	Creating table for the SQL query. You can create tables based on the supported connectors.	• Adding WATERMARK and PRIMARY KEY information • Creating table with LIKE clause
CREATE VIEW	Creating custom views using columns from tables. There is no physical data behind a view.	• Adding queries, expressions and joins
CREATE FUNCTION	Creating User Defined Function (UDF) for a query.	• Using Javascript or Java language • Using Functions tab on Streaming SQL Console
DROP TABLE	Deleting a table, view or function.	• Using Streaming SQL Console functionality to delete table, view or function
DROP VIEW		
DROP FUNCTIONS		
ALTER TABLE	Modifying table, view or function properties.	• Using RENAME TO or SET for table • Using RENAME TO or AS for view and function
ALTER VIEW		
ALTER FUNCTION		

For more information about Flink SQL, see the Apache Flink documentation.

Managing time in Cloudera SQL Stream Builder

Time attributes define how streams behave for time based operations such as window aggregations or joins. For Kafka tables you can use the Event Time tab to create source provided or user provided timestamp and watermarks. For other tables you can define time related attributes in the Flink DDL or directly in the SQL query. You can use timestamps that are already provided in the source or you can use custom timestamps.

Source-provided timestamps

Source-provided timestamps are inserted directly into the data stream by the source connector. This query uses the source-provided order_time field to perform a temporal join on multiple Kafka topics:

```
-- Table of orders
CREATE TABLE orders (
    order_id    STRING,
    price       DECIMAL(32,2),
    currency    STRING,
    order_time  TIMESTAMP(3),
    WATERMARK FOR order_time AS order_time
) WITH (/* ... */);

-- Table of currency rates
CREATE TABLE currency_rates (
    currency STRING,
    conversion_rate DECIMAL(32, 2),
    update_time TIMESTAMP(3),
    WATERMARK FOR update_time AS update_time
) WITH (/* ... */);
-- Event time temporal join to enrich orders with currencies
SELECT
     order_id,
     price,
     currency,
     conversion_rate,
     order_time,
FROM orders
LEFT JOIN currency_rates FOR SYSTEM TIME AS OF orders.order_time
ON orders.currency = currency_rates.currency
```

User-provided timestamps

You can also specify timestamps contained in the data stream itself. For example, if your schema includes a field called "order_time", it is possible to construct a query such as:

```
-- Table of orders
-- Converts order_time_string field to timestamp
CREATE TABLE orders (
    order_id    STRING,
    price       DECIMAL(32,2),
    currency    STRING,
    order_time_string STRING,
    order_time as to_timestamp(order_time_string),
    WATERMARK FOR order_time AS order_time
) WITH (/* ... */);
```

When an invalid timestamp is found in the stream (for example, NaN), the timestamp of the message is going to be 0. This way the message is excluded from the current window.

When your data does not include a timestamp in a suitable format, it is possible to compute a new timestamp column from another existing column using the Input Transform feature of SSB.

Flink DML

Flink SQL supports Data Manipulation Language (DML) statements to manipulate the data itself with adding, deleting or modifying.

The following table summarizes the supported DML statements in SQL Stream Builder:

DML	Description	Option
INSERT INTO	Inserting query results into a specified table.	• Inserting columns or values into a table • Adding OVERWRITE to overwrite the existing data in the table

For more information about Flink SQL, see the Apache Flink documentation.

Flink Queries

Flink SQL supports querying data with SELECT statements and using different types of operations.

You can query data from a table using the SELECT statement.

Query	Description	Option
SELECT	Querying data from a table using different operations.	• Selecting all data in a table • Selecting data using different types of operations

The following table summarizes the supported operations for SELECT statement in SQL Stream Builder:

Operation	Description	Option
WHERE	Querying data based on adding filters.	• Using boolean expression
JOIN	Joining data from tables based on an equivalent column.	• Using temporal joins based on event time or processing time • Using lookup join
GROUP BY	Grouping results using built-in or user defined functions.	• Using with streaming table provides updated results
ORDER BY	Ordering results to be sorted based on a specified expression.	• Using with streaming table the primary sorting key needs to be time

For more information about Flink SQL, see the Apache Flink documentation.

Other supported statements

Beside the supported DDL, DML and SELECT, the supported statements also include DESCRIBE, SHOW and SET that you can use in SQL Stream Builder.

The following table summarizes the supported statements for in SQL Stream Builder:

Operation	Description
DESCRIBE TABLES	Describing the schema of a table
SHOW TABLES	Showing a list of existing tables, views, functions, databases or catalogs
SHOW VIEWS	
SHOW FUNCTIONS	
SHOW DATABASES	
SHOW CATALOGS	
SET	Setting properties of session

For more information about Flink SQL, see the Apache Flink documentation.

Data Types

The logical type of a value to declare input and output types of operations in a table ecosystem is described by data types. Flink support a set of pre-defined data types that can be also used in Cloudera SQL Stream Builder.

The following list summarizes the pre-defined data types in Flink and SQL Stream Builder:

- CHAR
- VARCHAR
- STRING
- BOOLEAN
- BINARY
- VARBINARY
- BYTES
- DECIMAL - Supports fixed precision and scale.
- TINYINT
- SMALLINT
- INTEGER
- BIGINT
- FLOAT
- DOUBLE
- DATE
- TIME - Only supports a precision of 0.
- TIMESTAMP
- TIMPESTAMP_LTZ
- INTERVAL - Only supports interval of MONTH and SECOND(3).
- ARRAY
- MULTISET
- MAP
- ROW
- RAW
- Structured types - Only exposed in user-defined functions.

For more information about Data Types in Flink SQL, see the Apache Flink documentation.

Dynamic SQL Hints

SQL hints are supported for SQL Stream Builder (SSB) that allows you to use the dynamic table options. With the dynamic table options, you can alter any option of a table on a query level.

The dynamic table options of Flink SQL allows you to specify and override options for a table in a SQL query. The hint is formatted as a SQL comment containing an OPTIONS() clause, for example:

```
/*+ OPTIONS('key1'='value1', 'key2'='value2') */
```

You need to place the clause directly afte the table name where you need to change the options. You can use separate clauses for separate tables within a SQL query.

For example, you can use the following SQL hint to dynamically configure a Kafka consumer group at run time:

```
SELECT t1.column_a, t2.column_b
FROM tableName_A /*+ OPTIONS('properties.group.id'='my_consumer_group') */ A
S t1
JOIN tableName_B /*+ OPTIONS('properties.group.id'='my_other_consumer_group
') */ AS t2
```

```
ON t1.column_a = t2.column_b
```

Any option of a table that is accessible in the DDL of a connector can be overridden using the SQL hints. The options of a connector can be viewed on the **Connector** page of Streaming SQL Console.

For more information about the SQL Hints, see the official Apache Flink documentation.

SQL Examples

You can use the SQL examples for frequently used functions, syntax and techniques in Cloudera SQL Stream Builder. Cloudera SQL Stream Builder uses Calcite Compatible SQL, but to include the functionality of Flink you need to customize certain SQL commands.

Metadata commands

```
-- show all tables
SHOW tables;
-- describe or show schema for table
DESCRIBE payments;
DESC payments;
```

Timestamps, intervals and time

```
-- eventTimestamp is the Kafka timestamp
-- as unix timestamp. Magically added to every schema.
SELECT max(eventTimestamp) FROM solar_inputs;

-- make it human readable
SELECT CAST(max(eventTimestamp) AS varchar) as TS FROM solar_inputs;

-- date math with interval
SELECT * FROM payments
WHERE eventTimestamp > CURRENT_TIMESTAMP-interval '10' second;
```

Aggregation

```
-- hourly payment volume

SELECT SUM(CAST(amount AS numeric)) AS payment_volume,
CAST(TUMBLE_END(eventTimestamp, interval '1' hour) AS varchar) AS ts
FROM payments
GROUP BY TUMBLE(eventTimestamp, interval '1' hour);
```

```
-- detect multiple auths in a short window and
-- send to lock account topic/microservice

SELECT card,
MAX(amount) as theamount,
TUMBLE_END(eventTimestamp, interval '5' minute) as ts
FROM payments
WHERE lat IS NOT NULL
AND lon IS NOT NULL
GROUP BY card, TUMBLE(eventTimestamp, interval '5' minute)
HAVING COUNT(*) > 4 -- >4==fraud
```

Working with arrays

```
-- unnest each array element as separate row
SELECT b.*, u.*
 FROM bgp_avro b,
 UNNEST(b.path) AS u(pathitem)
```

Note: Arrays start at 1 not 0.

Union ALL

```
-- union two different tables
SELECT * FROM clickstream
WHERE useragent = 'Chrome/62.0.3202.84 Mobile Safari/537.36'
UNION ALL
SELECT * FROM clickstream
WHERE useragent = 'Version/4.0 Chrome/58.0.3029.83 Mobile Safari/537.36'
```

Math

```
-- simple math
SELECT 42+1 FROM mylogs;
-- inline
SELECT (amount+10)*upcharge AS total_amount
FROM payments
WHERE account_type = 'merchant'
```

```
-- convert C to F
SELECT (temp-32)/1.8 AS temp_fahrenheit
FROM reactor_core_sensors;
```

```
-- daily miles accumulator, 100:1
-- send to persistent storage microservice
-- for upsert of miles tally
SELECT card,
SUM(amount)/100 AS miles,
TUMBLE_END(eventTimestamp, interval '1' day)
FROM payments
GROUP BY card, TUMBLE(eventTimestamp, interval '1' day);
```

Joins

```
-- join multiple streams
SELECT o.name,
       sum(d.clicks),
       hop_end(r.eventTimestamp, interval '20' second, interval '40' second)
 FROM click_stream o join orgs r on o.org_id = r.org_id
                join models d on d.org_id = r.org_id
 GROUP BY o.name,
        hop(r.eventTimestamp, interval '20' second, interval '40' second)
```

```
-- join with temporal table where LatestRates is a temporal table
SELECT
  o.amount, o.currency, r.rate, o.amount * r.rate
FROM
  Orders AS o
```

```
  JOIN LatestRates FOR SYSTEM_TIME AS OF o.proctime AS r
  ON r.currency = o.currency
```

Hyperjoins

Joins are considered "hyperjoins" because SQL Stream Builder has the ability to join multiple tables in a single query, and because the Kafka table is created from a data provider, these joins can span multiple clusters/connect strings, but also multiple types of sources (join Kafka and a database for instance).

```
SELECT us_west.user_score+ap_south.user_score
FROM kafka_in_zone_us_west us_west
FULL OUTER JOIN kafka_in_zone_ap_south ap_south
ON us_west.user_id = ap_south.user_id;
```

Misc SQL tricks

```
-- concatenation
SELECT 'testme_'||name FROM logs;
```

```
-- select the datatype of the field
SELECT eventTimestamp, TYPEOF(eventTimestamp) as mytype FROM airplanes;
```

Escaping and quoting

Typical escaping and quoting is supported.

- Nested columns

  ```
  SELECT foo.`bar` FROM table; -- must quote nested column
  ```

- Literals

  ```
  SELECT "some string literal" FROM mytable; -- a literal
  ```

Built In Functions

```
-- convert EPOCH time to timestamp
select EPOCH_TO_TIMESTAMP(1593718981) from ev_sample_fraud;

-- convert EPOCH milliseconds to timestamp
select EPOCHMILLIS_TO_TIMESTAMP(1593718838150) from ev_sample_fraud;
```

Enriching streaming data with join

In Cloudera SQL Stream Builder, you can enrich your streaming data with values from a slowly changing dataset using join statements.

Regular join

Join statements in SQL serve to combine columns and rows from two or more tables based on a shared column. When you join tables from a slowly changing source such as HDFS, Kudu, Hive and so on, you can simply use the regular

JOIN syntax of SQL. The following example shows a regular INNER JOIN where the *ORDERS* table is joined with *PRODUCT* table based on the *PRODUCTID*:

```
SELECT * FROM Orders
INNER JOIN Product
ON Orders.productId = Product.id
```

A regular join can only be used with bounded tables. In a streaming context data is produced continuously, and with a regular join both sides of the join would need to be buffered indefinitely to store all of the events that would match with the result of the SQL query. To get results from a given amount of time and to join streaming tables, a time boundary needs to be specified. This means the tables not only need to be joined by a key or column, but also on a time attribute.

Interval join

When joining streaming tables, the time attribute can be defined in the SQL syntax using BETWEEN and an interval value:

```
SELECT *
FROM Orders o, Shipments s
WHERE o.id = s.order_id
AND o.order_time BETWEEN s.ship_time - INTERVAL '4' HOUR AND s.ship_time
```

In this case, *ORDERS* table is joined with the *SHIPMENTS* table and the results are going to be generated based on the *ID* column as long as the order time and shipment time is within four hours of each other. The condition of this scenario is that the events in the streams happen almost at the same time with minimal delay, so the time boundary can be defined between an approximate interval.

When you want to join a streaming table with a slowly changing table, time attributes can differ as one of the tables stores data over a long period of time, while the streaming table receives new data continuously. To join these types of tables, a time needs to be defined that can serve as a reference point for both types of tables.

Joining streaming and bounded tables

Beside regular join and interval join, in Flink SQL you are able to join a streaming table and a slowly changing dimension table for enrichment. In this case, you need to use a temporal join where the streaming table is joined with a versioned table based on a key, and the processing or event time.

A versioned table is a table that contains a time attribute, and reflects the records from a table at a specific point of time. When you use append-only or regularly updated sources, the values related to a key are updated over a long period of time. For example, a table can contain the currency rates since last month. At every change of the currency rate, a new value is added to the stream, therefore to the table. With creating a versioned table of the currency rate, you can specify which rate you need at an exact point of time: use the currency rates from 12:00.

For more information about Version Tables, see the official Apache Flink documentation.

Temporal join

After determining the version of the bounded table, you also need to define a time for the streaming table. In Flink, event time and processing time can be specified. When using event time, you need to create a temporal join.

Event time is the time that each individual event occurred on its producing device. To have an event time attribute in your SQL query, you need to define a timestamp column with a watermark definition column when creating the table:

```
CREATE TABLE orders (
    order_id    STRING,
    price       DECIMAL(32,2),
    currency    STRING,
    order_time  TIMESTAMP(3),
```

```
    proc_time AS PROCTIME(),
    WATERMARK FOR order_time AS order_time
) WITH (
  ...
)
```

When you want to use event time in a JOIN, you need to refer to the event time column as defined for the table in the *FOR SYSTEM_TIME AS OF* part of the SQL query:

```
SELECT
     order_id,
     price,
     currency,
     conversion_rate,
     order_time,
FROM orders
LEFT JOIN currency_rates FOR SYSTEM_TIME AS OF orders.order_time
ON orders.currency = currency_rates.currency
```

Lookup join

As a special case of temporal join, you can use the processing time as a time attribute. In Flink, processing time is the system time of the machine, also known as “wall-clock time”. When you use the processing time in a JOIN SQL syntax, Flink translates into a lookup join and uses the latest version of the bounded table. The following example shows the join syntax that needs to be used for enriching streaming data:

```
SELECT o.order_id, o.total, c.country, c.zip
FROM Orders AS o
  JOIN Customers FOR SYSTEM_TIME AS OF o.proc_time AS k
    ON o.customer_id = c.id
```

In the above example, *CUSTOMERS* serves as the lookup table. The FOR SYSTEM_TIME AS OF o.proc_time syntax indicates that you always want to look up in the latest version of the table. By including the processing time in the SQL syntax, you can query the latest version of a lookup table, and enrich your streaming data with the corresponding value.

In Cloudera SQL Stream Builder, the following connectors are supported as lookup tables:

- Kudu
- Hive
- JDBC

Example: joining Kafka and Kudu tables

Using lookup join, you can join Kafka and Kudu tables to enrich the streaming data of Kafka with information from the Kudu tables. In the this example, Orders of a Kafka streaming table are enriched with metadata information from a Kudu table.

As a prerequisite for the example, the following steps were completed in SQL Stream Builder:

- Registering Kafka as a Data Source.
- Registering Kudu as a Catalog.
- Creating Orders Kafka table.
- Creating ItemMeta Kudu table.
- Generating data for Kafka and Kudu tables.

The tables in the example contain the following information:

Tables	Columns
Kafka - Orders	• order_number • price • order_time • item_id
Kudu - ItemMeta	• id • info

In the scope of this example, the *ORDERS* table will be joined with latest version of the *ITEMMETA* table based on the item *ID*, and the selected information is sampled under the **Results** tab of the Streaming SQL Console:

```
SELECT order_time, item_id, info, price
FROM Orders
JOIN kudu.default_database.ItemMeta FOR SYSTEM_TIME AS OF o.proc_time AS k
ON item_id = id
```

After running the SQL query, the results are continuously sampled to the Streaming SQL Console, the rows of *ORDER_TIME*, *ITEM_ID* and *PRICE* are enriched with *INFO* column from the *ITEMMETA* Kudu table.

Updating SQL queries with PROCTIME function

After upgrading to Cloudera Public Cloud 7.2.18 , you need to migrate your SQL queries using PROCTIME() function in lookup joins, because the PROCTIME() function is no longer avaialable in CSA.

As the PROCTIME() shortcut function was removed from Cloudera Streaming Analytics, the SQL syntax changes for lookup joins. For example, let's say you have a streaming_table for continuously incoming data (for example, a Kafka source table), and a kudu_table to enrich that streaming data, and you want to join them on an id column. With PROC TIME(), you would use the following syntax:

```
SELECT * FROM `streaming_table` AS s
  JOIN `kudu_table` FOR SYSTEM_TIME AS OF PROCTIME() AS k
    ON s.id = k.id;
```

But as PROCTIME() is no longer available, the syntax should look like the following example:

```
SELECT * FROM `streaming_table` AS s
  JOIN `kudu_table` FOR SYSTEM_TIME AS OF s.proc_time AS k
    ON s.id = k.id;
```

The difference is that the left side table is required to have a calculated column that represents the processing time attribute. The name of the column can be anything, but it is required to use that column for the FOR SYSTEM_TIME AS OF ... expression instead of the raw PROCTIME() function.

You can update the existing SQL queries that use PROCTIME() to avoid SQL jobs to fail using one of the following methods:

- Adding the computed PROCTIME() column to the existing table:

  ```
  ALTER TABLE `streaming_table` ADD `proc_time` AS PROCTIME();
  ```

- Creating a new table through copying the original DDL:

  ```
  CREATE TABLE `streaming_table_with_proc_time` (
    -- Add proc_time column.
  `proc_time` AS PROCTIME()
  ) WITH (
    -- No changes here.
  ```

```
) LIKE `streaming_table`;
```

You can also create the new table using the LIKE clause:

```
CREATE TABLE `streaming_table_with_proc_time` (
  ... -- The existing table schema.
`proc_time` AS PROCTIME()
) WITH (
  ... -- The existing table properties.
) LIKE `streaming_table`;
```

Cloudera DataFlow for Data Hub 7.3.1

Processing mainframe / EBCDIC data in Apache NiFi

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Use case overview **4**

Before you begin **4**

Configuring EBCDICRecordReader **4**

Building your dataflow **6**

Use case overview

Learn how to use NiFi to process Mainframe data encoded in the Extended Binary Coded Decimal Interchange Code (EBCDIC) scheme while building dataflows in CDP Public Cloud.

Cloudera Flow Management offers a Cloudera exclusive NiFi component designed to process Mainframe / EBCDIC encoded data. This Record Reader component can be used to convert Mainframe data into other structured formats such as JSON or Avro.

Note:

The EBCDICRecordReader component is currently in Technical Preview and will transition to General Availability based on customer feedback.

This controller service component operates under the assumption that the data has already been offloaded from the Mainframe and NiFi would retrieve the binary data files and convert them using a processor like ConvertRecord.

The Mainframe / EBCDIC data processing workflow involves:

1. Retrieving binary files exported from the Mainframe.
2. Converting the EBCDIC encoded data into a readable format, such as JSON.
3. Storing or further processing the transformed data.

This use case walks you through creating a dataflow that receives files using GetFile, converts the data using the ConvertRecord processor with the help of the EBCDICRecordReader, and writes the converted data with PutFile. It gets you started with creating a basic conversion dataflow. However, if you require a different data source or target, you have various processor options in NiFi.

Before you begin

Before setting up a NiFi dataflow to process Mainframe data encoded in the Extended Binary Coded Decimal Interchange Code (EBCDIC) scheme between moving the data between your source and target, you must meet the following prerequisites.

- Ensure that Cobol copybooks describing the schemas of the tables are available in NiFi.
- Ensure that binary files exported from the Mainframe are available in NiFi.

Configuring EBCDICRecordReader

Learn how to create and configure the EBCDICRecordReader controller service for your conversion dataflow.

About this task

Controller services provide shared services to be used by the processors in your dataflow. Configure the EBCDICRecordReader before building the NiFi dataflow to ensure that it is available during processor configuration.

The EBCDICRecordReader reads Extended Binary Coded Decimal Interchange Code (EBCDIC) encoded input data and outputs a record representation. It relies on a COBOL copybook schema that needs to be provided using the Copybook Path property. One instance of this controller service can handle one type of data described by the corresponding copybook.

Note:

The EBCDICRecordReader component is currently in Technical Preview and will transition to General Availability based on customer feedback.

Configure the EBCDICRecordReader controller service with the path to the COBOL copybook, allowing NiFi to interpret the Mainframe data correctly.

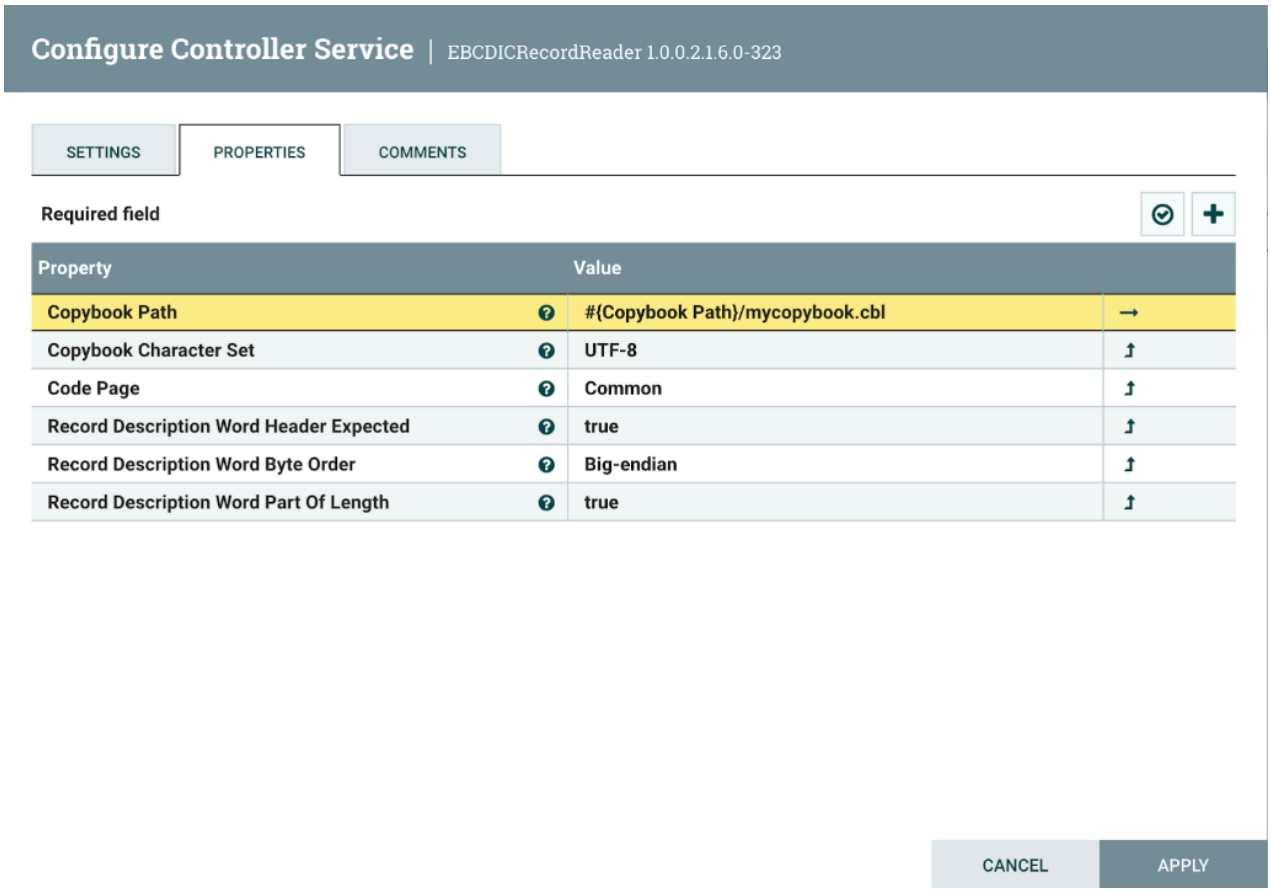

Example

Example COBOL copybook file structure:

```
        01  MONTH-END-EXT-BORR-ACCT-REC.
            02 EBORR-DEM-DATA.
              04 EBORR-ACCT-NUM.
                06 EBORR-SSN              PIC X(9).
                06 EBORR-SSN-SUFFIX       PIC X.
              04 FILLER                   PIC X(288).                    00670829
              04 EBORR-NUM-OF-ABCDS       PIC 99.
            02 MONTH-END-EXT-ABCD-DATA  OCCURS 1 TO 20 TIMES
                                        DEPENDING ON EBORR-NUM-OF-ABCDS.
              04 EABCD-RECORD.
                 06 EABCD-BINT-FLG        PIC X(1).
                 06 EABCD-BINT-PEND-AMT   PIC S9(7)V9(2) COMP-3.
                 06 EABCD-BINT-OUTS-AMT   PIC S9(7)V9(2) COMP-3.
                 06 EABCD-BINT-PT-PD-AMT PIC S9(7)V9(2) COMP-3.
                 06 EABCD-CS-BKD-PRM-AMT PIC S9(5)V9(2) COMP-3.
                 06 EABCD-NC-BKD-PRM-AMT PIC S9(5)V9(2) COMP-3.
                 06 EABCD-BKD-PRM-RT-AMT PIC S9(2)V9(3) COMP-3.
                 06 EABCD-FTD-PRM-RT-AMT PIC S9(2)V9(3) COMP-3.
                 06 EABCD-APP-YR          PIC 9(4).
                 06 EABCD-EBR-FLG         PIC X(1).
                 06 EABCD-ORG-1ST-PMT-DT.
                    08 EABCD-ORG-1ST-PMT-YY PIC X(4).
                    08 EABCD-ORG-1ST-PMT-MM PIC X(2).
                    08 EABCD-ORG-1ST-PMT-DD PIC X(2).
  L30753         06 EABCD-MPN-SEQ-NUM        PIC S9(9) COMP.
  L30753         06 EABCD-MPN-TYP-CD         PIC X(1).
  L30753         06 EABCD-MPN-STA-CD         PIC X(1).
  L30753         06 EABCD-ORIG-LDR-ID        PIC X(6).
  L30753         06 EABCD-ORIG-LDR-BR-ID     PIC X(4).
  L30753         06 EABCD-PR-OWN-ID          PIC X(6).
  L30753         06 EABCD-PR-OWN-BR-ID       PIC X(4).
```

Building your dataflow

Learn how to create a NiFi dataflow that processes Mainframe / EBCDIC encoded data.

About this task

The basic steps of setting up the elements of your dataflow in NiFi involve opening NiFi in a CDP Public Cloud Flow Management cluster, adding processors to your NiFi canvas, and connecting the processors.

Procedure

1. Open NiFi
   in
   CDP Public Cloud.
   a) To access the NiFi service in your Flow Management cluster, navigate to Management Console service Data Hub Clusters .
   b) Click the tile representing the Flow Management Data Hub cluster that you want to work with.
   c) Click the NiFi icon in the Services section of the cluster overview page to access the NiFi UI.

      You will be logged into NiFi automatically with your CDP credentials.

2. Add and configure the `GetFile` processor for data input.

   This processor creates FlowFiles from files in a directory.

   a) Drag and drop the processor icon into the canvas.

      This displays a dialog that allows you to choose the processor you want to add.

   b) Use the Add Processor filter box to search for the `GetFile` processor and click Add.

   You can use other processors (for example SFTP processors) to retrieve offloaded binary files from a given location.

3. Add and configure the `ConvertRecord` processor for data conversion.

   This processor converts records from one data format to another using Record Reader and Record Write Controller Services. You can use it to transform your EBCDIC encoded data into JSON or other structured formats. Configure it with the EBCDICRecordReader previously described and a default JSON writer.

4. Add and configure the `PutFile` processor for data output.

   You can replace PutFile with any other sink processor appropriate for your use case (for example ingesting data into Iceberg, CDW, object store).

5. Connect the processors to create the flow.

   a) Drag the connection icon from the one processor, and drop it on the next processor.

   b) Configure the connection.

   c) Click Add to close the dialog box and add the connection to your flow.

Results

Your dataflow may resemble the following:

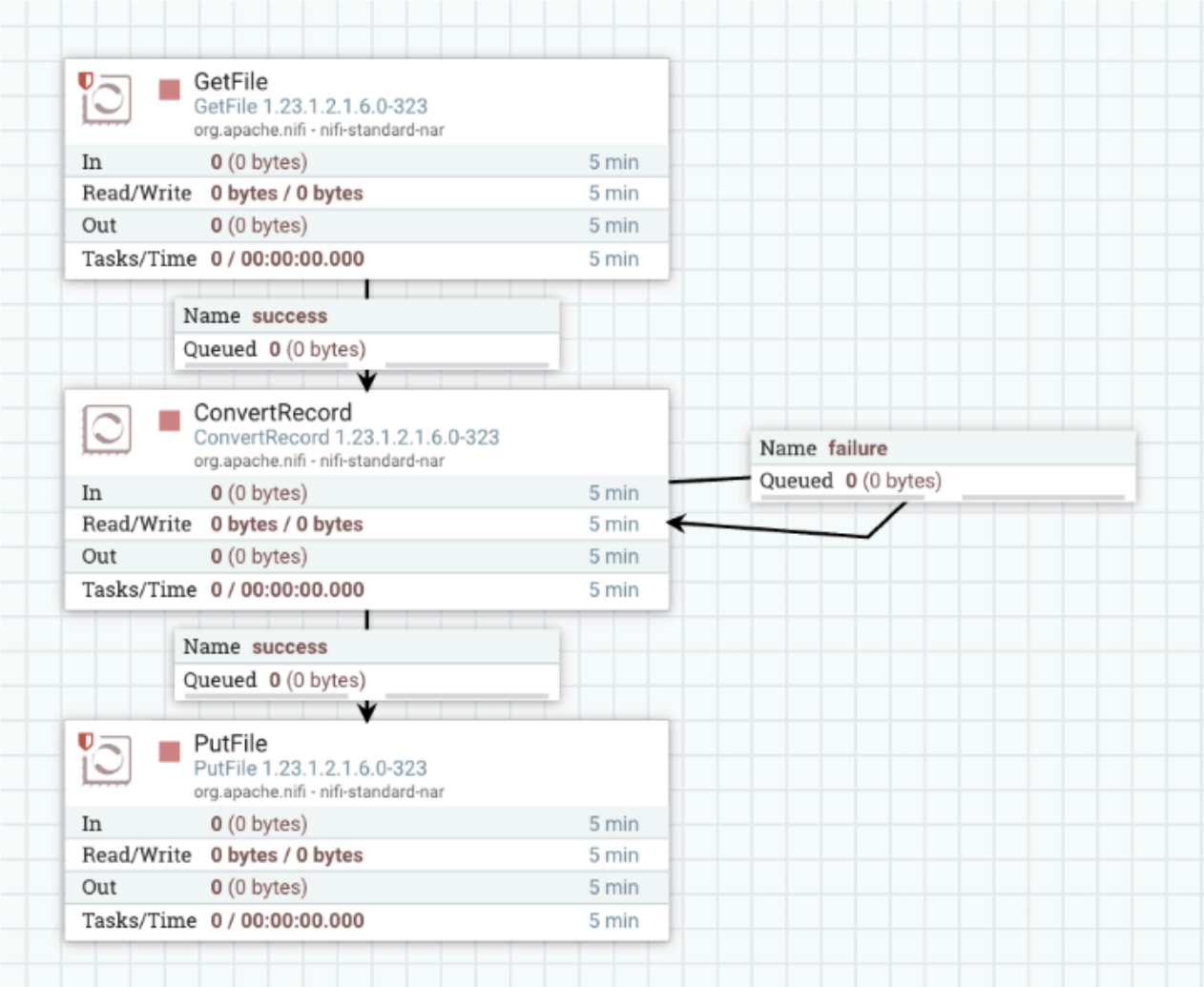

When this flow is running and receives a file from the Mainframe, it converts the data to JSON format, with a result that would look like the following:

View as: original


```
[ {
  "EBORR_DEM_DATA" : {
    "EBORR_ACCT_NUM" : {
      "EBORR_SSN" : "001030007",
      "EBORR_SSN_SUFFIX" : "1"
    },
    "FILLER__0" : null,
    "EBORR_NUM_OF_ABCDS" : 8
  },
  "MONTH_END_EXT_ABCD_DATA" : [ {
    "EABCD_RECORD" : {
      "EABCD_BINT_FLG" : " ",
      "EABCD_BINT_PEND_AMT" : 0.00,
      "EABCD_BINT_OUTS_AMT" : 0.00,
      "EABCD_BINT_PT_PD_AMT" : 0.00,
      "EABCD_CS_BKD_PRM_AMT" : 0.00,
      "EABCD_NC_BKD_PRM_AMT" : 0.00,
      "EABCD_BKD_PRM_RT_AMT" : 0.000,
      "EABCD_FTD_PRM_RT_AMT" : 0.000,
      "EABCD_APP_YR" : 0,
      "EABCD_EBR_FLG" : " ",
      "EABCD_ORG_1ST_PMT_DT" : {
        "EABCD_ORG_1ST_PMT_YY" : "2007",
```

Cloudera DataFlow for Data Hub 7.3.1

Automatic access to new components and fixes

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Automatic access to new components and fixes without upgrading..... 4

Automatic access to new components and fixes without upgrading

NiFi in Flow Management DataHub clusters is configured out of the box with a feature allowing it to fetch new components and fixes published by Cloudera.

The objective of this feature is to distribute new components as well as fixes on existing components like processors, controller services, or reporting tasks, without the need to upgrade to a new version. To do this, the NiFi nodes check for new NAR files published on the Cloudera repository.

When the feature is enabled, NiFi checks every 24 hours if NARs corresponding to the currently used NiFi versions are made available by Cloudera in the repository. If NARs are available, the NiFi nodes download the NARs through the internet and deploy the files in the hot loading directory so that the new components are automatically made available in NiFi without requiring any restart.

Note:

If a fix is being pushed for a component like a processor, you will have multiple versions of the same component in NiFi and you will need to switch to the new version of the component using the corresponding API or directly in the UI, in case of a processor, for example, by right-clicking it.

When Cloudera publishes new components or fixes using this feature, the release notes will be updated in the documentation to detail the changes being made available.

To turn off this feature, clear the nifi.nar.hotfix.provider.enabled property in the Cloudera Manager instance managing your DataHub cluster.

You can also turn off the feature during provisioning by overriding the nifi.nar.hotfix.provider.enabled property. For details, see *Cluster template overrides*.

Related Information

Cluster template overrides

Cloudera DataFlow for Data Hub 7.3.1

Ingesting Data into Amazon S3

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ingesting data into Amazon S3 4
Understand the use case 4
Meet the prerequisites 4
Build the data flow 5
Set up AWS for your ingest data flow 7
Create IDBroker mapping 8
Create controller services for your data flow 10
Configure the processor for your data source 13
Configure the processor for merging records 15
Configure the processor for your data target 16
Start the data flow 17
Verify data flow operation 18

Monitoring your data flow 18

Next steps 19

Ingesting data into Amazon S3

You can use an Apache NiFi data flow to ingest data into Amazon S3 object stores in CDP Public Cloud by following these steps.

Understand the use case

Learn how you can use a Flow Management cluster connected to a Streams Messaging cluster to build an end-to-end flow that ingests data to Amazon S3 storage. This example use case shows you how to use Apache NiFi to move data from Kafka to S3 buckets.

Why move data to object stores?

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost-effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Apache NiFi for cloud ingestion

As a part of Cloudera Data Flow in CDP Public Cloud, Flow Management clusters run Apache NiFi. You can use NiFi to move data from a range of locations into object stores. NiFi supports data transfer from nearly any type of data source to cloud storage solutions in a secure and governed manner. It also offers a scalable solution for managing data flows with guaranteed delivery, data buffering / pressure release, and prioritized queuing.

NiFi helps you to build a cloud ingestion data flow in a quick and user-friendly way, by simply dragging and dropping a series of processors on the NiFi user interface. When the data flow is ready, you can visually monitor and control the pipeline you have built.

This use case walks you through the steps associated with creating an ingest-focused data flow from Apache Kafka into Amazon S3 object stores. If you are moving data from a location other than Kafka, see the *Getting Started with Apache NiFi* for information about how to build a data flow, and about other data get and consume processor options.

Related Information

Getting Started with Apache NiFi
Ingesting Data into Apache Kafka
Ingesting Data into Apache HBase in CDP Public Cloud
Ingesting data into Apache Hive in CDP Public Cloud
Ingesting Data into Apache Kudu in CDP Public Cloud
Ingesting Data into Azure Data Lake Storage

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a CDP Public Cloud environment.
- You have a CDP username and password set to access Data Hub clusters. The predefined resource role of this user is at least EnvironmentUser. This resource role provides the ability to view Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the CDP Public Cloud environment.
- You have a Flow Management Data Hub cluster running in your CDP Public Cloud environment.
- You have a Streams Messaging Data Hub cluster in the same CDP environment as the Flow Management cluster.

- Your CDP user has been added to the appropriate pre-defined Ranger access policies to allow access to the NiFi UI.
- You have set up a consumer group in Ranger, and you have a Ranger policy for Kafka allowing your CDP user to access this consumer group.
- You have a Ranger policy on your Kafka topic you will use as a data source allowing the CDP user to consume from the topic.
- You have created a target folder on your AWS Management Console for the data to be moved to the cloud.
- You have created an AWS role with policies attached allowing write access to the S3 bucket you want to use in your data flow.

Related Information

Understanding roles and resource roles

Setting up your Flow Management cluster

Configuring Ranger policies for Apache NiFi

IDBroker

Amazon S3 Console User Guide

Build the data flow

Learn how you can create an ingest data flow to move data from Kafka to S3 buckets. This involves opening Apache NiFi in your Flow Management cluster, adding processors and other data flow objects to your canvas, and connecting your data flow elements.

About this task

You can use the `PutHDFS` or `PutS3Object` processors to build your AWS ingest data flows. Regardless of the type of flow you are building, the first steps in building your data flow are generally the same. Open NiFi, add your processors to the canvas, and connect the processors to create the flow.

Procedure

1. Open NiFi in CDP Public Cloud.
   a) To access the NiFi service in your Flow Management Data Hub cluster, navigate to Management Console service Data Hub Clusters .
   b) Click the tile representing the Flow Management Data Hub cluster you want to work with.

c) Click Nifi in the Services section of the cluster overview page to access the NiFi UI.

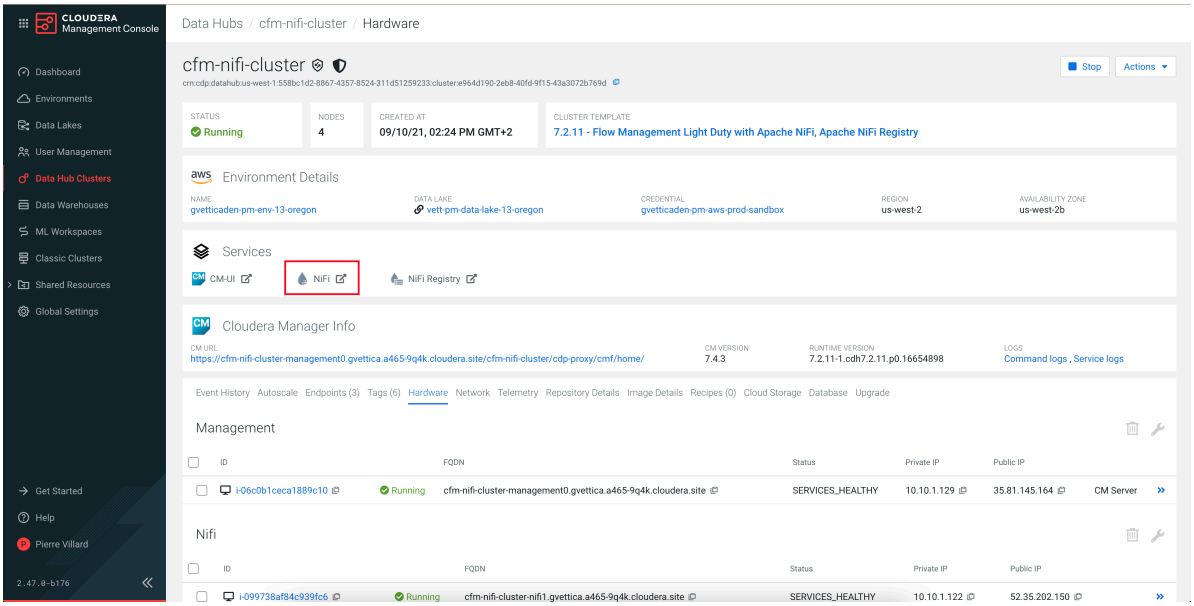

You will be logged into NiFi automatically with your CDP credentials.

2. Add the `ConsumeKafkaRecord_2_0` processor for data input in your data flow.

   a) Drag and drop the processor icon into the canvas.

   This displays a dialog that allows you to choose the processor you want to add.

   b) Select the `ConsumeKafkaRecord_2_0` processor from the list.

   c) Click Add or double-click the required processor type to add it to the canvas.

   Note: This processor consumes messages from Kafka specifically built against the Kafka 2.0 Consumer API.

3. Add the `MergeRecord` processor.

   Note: This processor merges together multiple record-oriented flowfiles into a single flowfile that contains all of the records of the input flowfiles. It works by creating bins and then adding flowfiles to these bins until they are full. Once a bin is full, all of the flowfiles are combined into a single output flowfile, and that flowfile is routed to the **merged** relationship.

   When using the `ConsumeKafkaRecord_2_0` processor, you are pulling small-sized records, so it is practical to merge them into larger files before writing them to S3.

4. Add a processor for writing data to S3.

   You have two options:

   - `PutHDFS` processor: The HDFS client writes to S3 through the S3A API. This solution leverages centralized CDP security. You can use the usernames and passwords you set up in CDP for central authentication, and all requests go through IDBroker.
   - `PutS3Object` processor: This is an S3-specific processor that interacts with the Amazon S3 object store while still leveraging the authentication mechanism provided through IDBroker in CDP Public Cloud.

   a) Drag and drop the processor icon into the canvas.

   In the dialog box you can choose which processor you want to add.

   b) Select the processor of your choice from the list.

   c) Click Add or double-click the required processor type to add it to the canvas.

5. Connect the processors to create the data flow by clicking the connection icon in the first processor, and dragging and dropping it on the second processor.

   A Create Connection dialog appears with two tabs: Details and Settings. You can configure the connection's name, flowfile expiration time period, thresholds for back pressure, load balance strategy and prioritization.

   a) Connect `ConsumeKafkaRecord_2_0` with MergeRecord.
   b) Add the **success** flow of the `ConsumeKafkaRecord_2_0` processor to the `MergeRecord` processor.
   c) Click Add to close the dialog box and add the connection to your data flow.
   d) Connect `MergeRecord` with your target data processor (`PutHDFS` / `PutS3Object`).
   e) Add the **merged** flow of the `MergeRecord` processor to the target data processor.
   f) Click Add to close the dialog box and add the connection to your data flow.
6. Optionally, you can add funnels to your flow.
   a) `ConsumeKafkaRecord_2_0`: If any of the Kafka messages are pulled but cannot be parsed, the continents of the message will be written to a separate flowfile and that flowfile will be transferred to the parse.failure relationship. You can connect the failure queue coming from the processor to a failure funnel for parse failures.
   b) `MergeRecord`: For merge failures, you can connect the failure queue coming from the processor to the failure funnel you used for `ConsumeKafkaRecord_2_0`.
   c) `PutHDFS` / `PutS3Object`: You can add success and failure funnels at the end of the data flow and connect them with your target data processor. These funnels help you see where flow files are being routed when your flow is running.

   If you want to know more about working with funnels, see the *Apache NiFi User Guide*.

Results

This example data flow has been created using the `PutHDFS` processor.

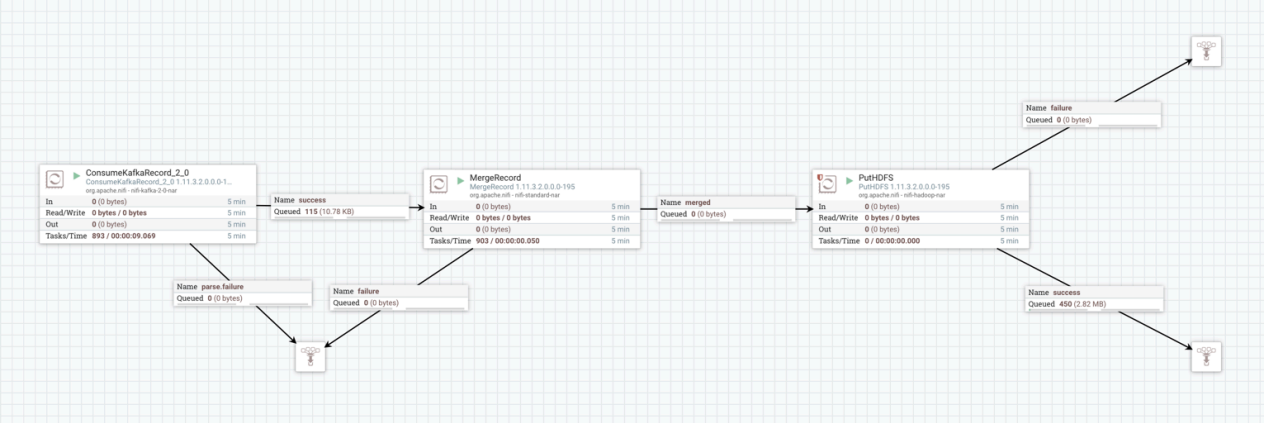

What to do next

Configure IDBroker mapping authorization.

Related Information

Apache NiFi Documentation

Set up AWS for your ingest data flow

Learn what you need to set on the AWS side to prepare for data ingestion.

Procedure

1. Create an S3 bucket that will be the target folder for ingesting data.

   For instructions, see *Create your first S3 bucket*.
2. Create a policy to allow writing into your bucket.

   For instructions, see *Creating IAM policies*.
3. Create an AWS IAM role.

   You will use this AWS identity to delegate access to your S3 bucket.
4. Apply the policy you have created to grant access to your S3 bucket to the newly create IAM role.
5. Create a trust relationship between the AWS IAM role and the CDP IDBroker role that you create when you registered your environment to allow IDBroker to assume the AWS role.

   For more information, see *Onboarding CDP users and groups for cloud storage (no RAZ)*.

What to do next

You are ready with preparing your AWS environment. Switch back to CDP and create an IDBroker mapping.

Related Information

Create your first S3 bucket

Creating IAM policies

Onboarding CDP users and groups for cloud storage (no RAZ)

Create IDBroker mapping

Learn how you can create the IDBroker mapping for the target data processors (`PutS3Object` and `PutHDFS`) of your data flow. To enable your CDP user to utilize the central authentication features CDP provides and to exchange credentials for AWS access tokens, you have to map your CDP user to the correct IAM role.

About this task

The option to add / modify the mappings is available from the Management Console in your CDP environment.

Procedure

1. Access IDBroker mappings in the Management Console.
   a) To access IDBroker mappings in your environment, click Actions Manage Access .

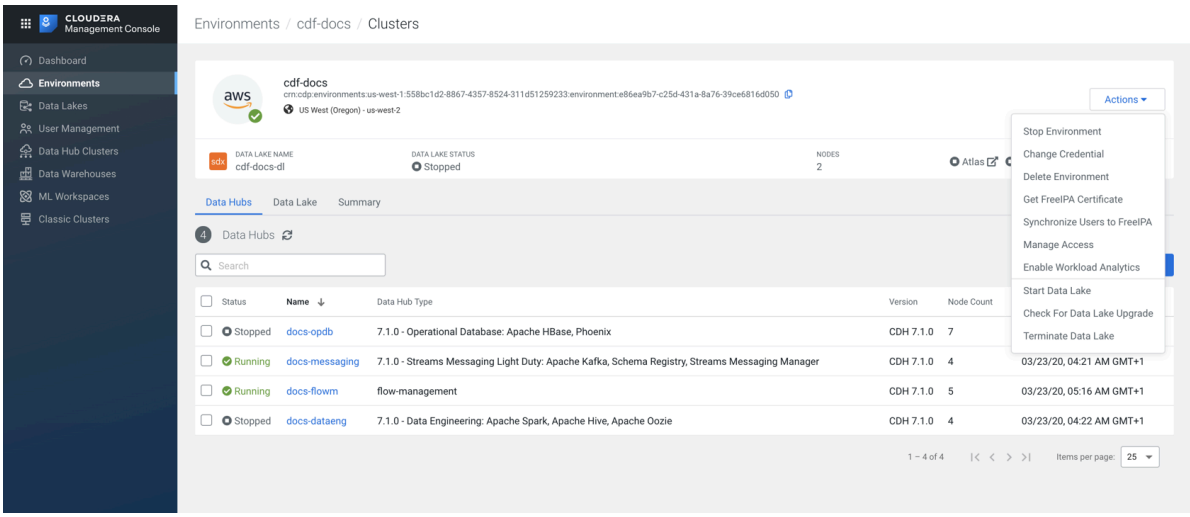

   b) Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.

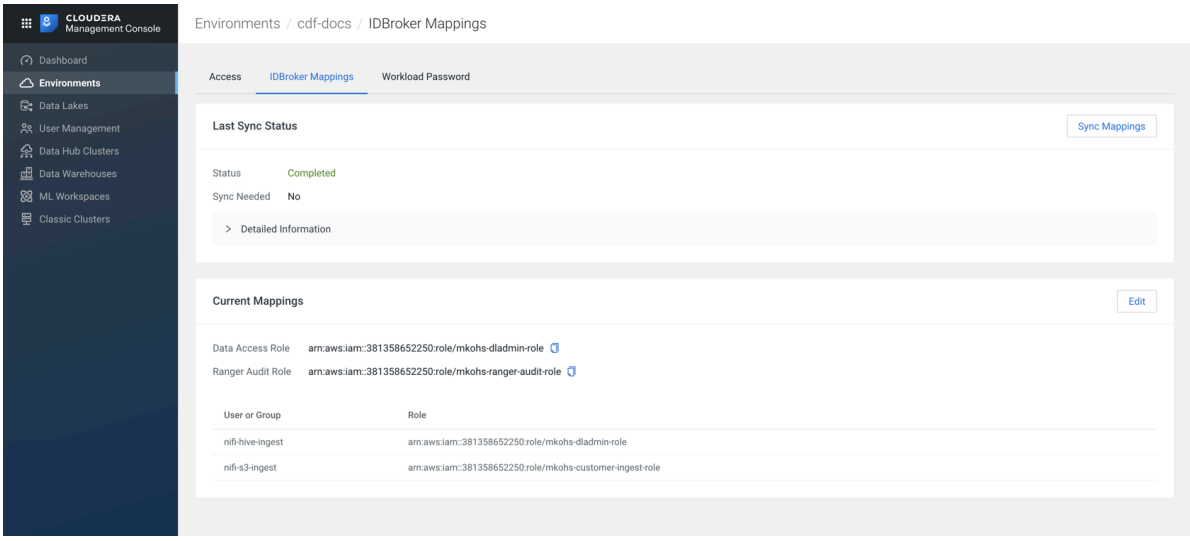

2. Add your CDP user account, a service account, or a user group and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

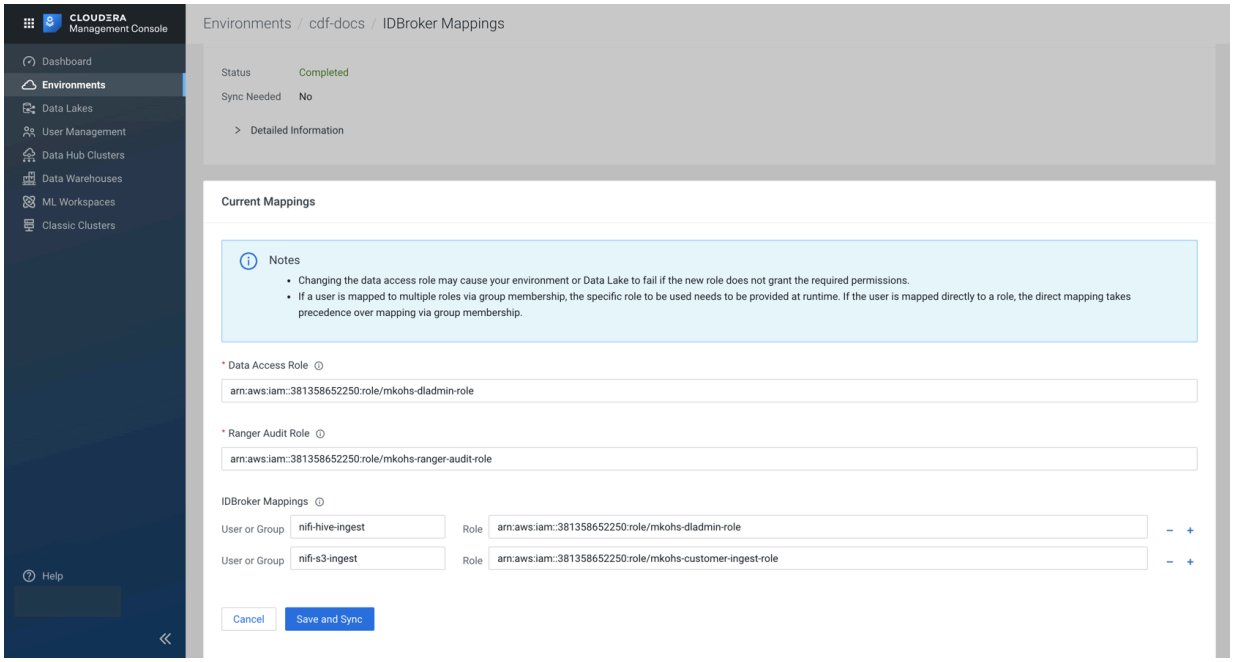

Note: You can get the AWS IAM role Amazon Resource Name (ARN) from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.

3. Click Save and Sync.

What to do next

Configure controller services for your data flow.

Related Information

IDBroker

Onboarding CDP users and groups for cloud storage

Create a provisioning credential for AWS

Create controller services for your data flow

Learn how you can create and configure controller services for an S3 ingest data flow in CDP Public Cloud. Controller services provide shared services that can be used by the processors in your data flow. You will use these Controller Services later when you configure your processors.

Procedure

1. To add a Controller Service to your flow, right-click on the canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.

3. Click the + button to display the Add Controller Service dialog.

4. Select the required Controller Service and click Add.

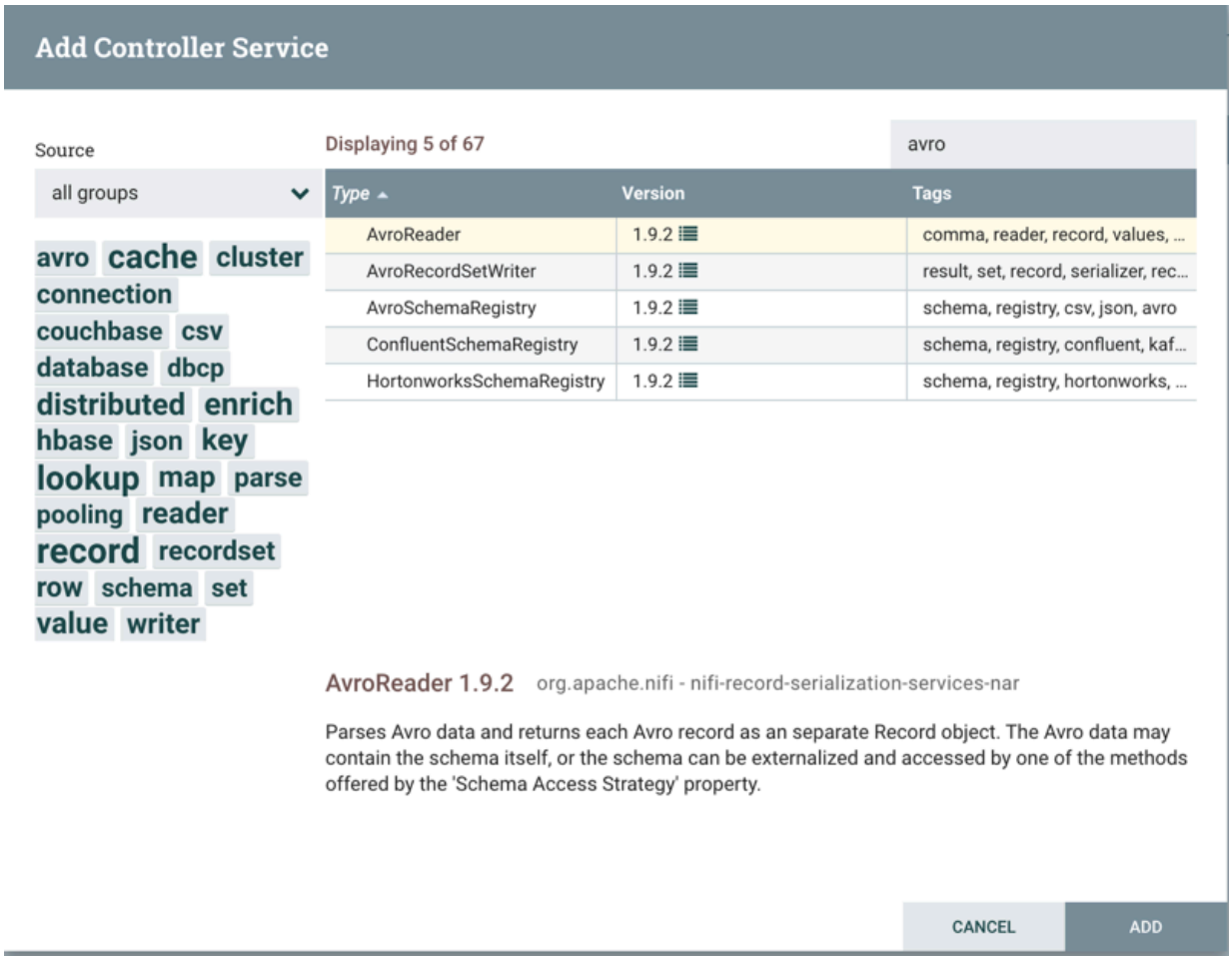

5. Perform any necessary Controller Service configuration tasks by clicking the Configure icon in the right-hand column.

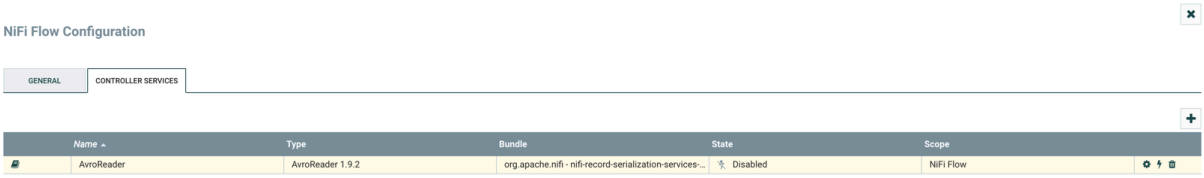

6. When you have finished configuring the options you need, save the changes by clicking the Apply button.

7. Enable the Controller Service by clicking the Enable button (flash) in the far-right column of the Controller Services tab.

Example

In this example the following controller services are used:

- AvroReader Controller Service
- CSVReader Controller Service
- CSVRecordSetWriter Controller Service
- AWSIDBrokerCloudCredentialsProvider Controller Service

You will use the record reader and record writer controller services when configuring the data source and merge record processors in your data flow.

You will use the AWS controller service when configuring the PutS3Object data target processor.

AvroReader Controller Service

This controller service parses Avro data and returns each Avro record as a separate record object.

Table 1: AvroReader Controller Service properties

Property	Description	Example value for ingest data flow
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	HWX Content-Encoded Schema Reference
Schema Registry	Specify the Controller Service to use for the Schema Registry.	CDPSchemaRegistry
Schema Name	Specify the name of the schema to look up in the Schema Registry property.	customer

CSVReader Controller Service

This controller service parses your CSV-formatted data, returning each row in the CSV file as a separate record.

Table 2: CSVReader Controller Service Properties

Property	Description	Example value for ingest data flow
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	Use String Fields From Header
Treat First Line as Header	Specify whether the first line of CSV should be considered a header or a record.	true

CSVRecordSetWriter Controller Service

This controller service writes the contents of a record set as CSV data.

Table 3: CSVRecordSetWriter Controller Service Properties

Property	Description	Example value for ingest data flow
Schema Write Strategy	Specify how the schema for a record should be added to the data.	Do Not Write Schema
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	Use 'Schema Name' Property
Schema Name	Specify the name of the schema to look up in the Schema Registry property.	customer

AWSIDBrokerCloudCredentialsProvider Controller Service

This controller service defines credentials for the PutS3Object processor.

Table 4: CSVRecordSetWriter Controller Service Properties

Property	Description	Example value for ingest data flow
Configuration Resources	Provide the path to the core site configuration file that contains IDBroker-related configuration to. The path is set as default. The only thing you have to configure is username and password. **Note:** The core-site.xml file is present on every Flow Management cluster. Cloudera Manager stores the right core-site.xml file in the same /etc directory for every cluster.	/etc/hadoop/conf.cloudera.core_settings/core-site.xml
Username	Specify your username.	csso_jsmith
Password	Specify your password. If you use your own username, provide the CDP workload password associated with your username.	CDP workload password

What to do next

Configure the processors in your data flow.

Related Information

Adding Controller Services for data flows

Apache NiFi Documentation

Configure the processor for your data source

Learn how you can configure the `ConsumeKafkaRecord_2_0` data source processor for your S3 ingest data flow. You can set up a data flow to move data to Amazon S3 from many different locations. This example assumes that you are streaming data from Kafka and shows you the configuration for the relevant data source processor.

Procedure

1. Launch the Configure Processor window, by right clicking the `ConsumeKafkaRecord_2_0` processor and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.

3. When you have finished configuring the options you need, save the changes by clicking the Apply button.

   Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

   In this example data flow, the data source is Kafka. You can create the modified Kafka broker URLs using the broker hostnames and adding port :9093 to the end of each FQDN. You can find the hostnames on the Streams Messaging cluster overview page when selecting the Hardware tab.

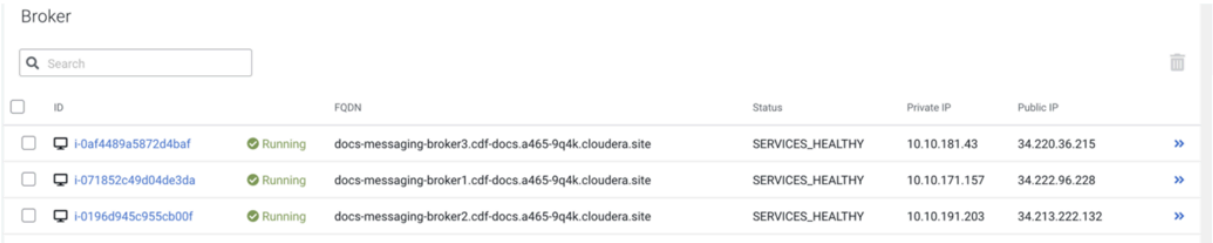

   Note: Property values can be parameterized. For example, you can create a parameter context to hold Kafka connection information and apply it to the Kafka Brokers property instead of adding the broker URLs individually.

   The following table includes a description and example values for the properties required to configure the example ingest data flow. For a complete list of `ConsumeKafkaRecord_2_0` properties, see the *Apache Nifi Documentation*.

Table 5: ConsumeKafkaRecord_2_0 processor properties

Property	Description	Example value for ingest data flow
Kafka Brokers	Provide a comma-separated list of known Kafka Brokers. In the format: <host>:<port>	Docs-messaging-broker1.cdf-docs.a465-9q4k.cloudera.site:9093,docs-messaging-broker2.cdf-docs.a465-9q4k.cloudera.site:9093,docs-messaging-broker3.cdf-docs.a465-9q4k.cloudera.site:9093
Topic Name(s)	Provide the name of the Kafka Topic(s) to pull from.	Customer
Record Reader	Specify the Record Reader to use for incoming FlowFiles.	ReadCustomerAvroRoot
Record Writer	Specify the Record Writer to use in order to serialize the data before sending to Kafka.	CSVRecordSetWriter
Security Protocol	Specify the protocol used to communicate with Kafka brokers.	SASL_SSL
SASL Mechanism	Specify the SASL mechanism to use for authentication.	PLAIN
Username	Use your CDP workload username to set this Authentication property.	srv_nifi-kafka-ingest
Password	Use your CDP workload password to set this Authentication property.	password
SSL Context Service	Specify the SSL Context Service to use for communicating with Kafka. Use the pre-configured SSLContextProvider.	Default NiFi SSL Context Service

Property	Description	Example value for ingest data flow
Group ID	Provide the consumer group ID to identify consumers that are within the same consumer group.	nifi-s3-ingest

Note: If you want to move data to Amazon S3 from a location other than Kafka, see the *Apache NiFi Getting Started* for information about other data ingest processor options.

What to do next

Configure the processor for merging your records.

Related Information

Configuring a processor

Apache NiFi Documentation

Getting Started with Apache NiFi

Configure the processor for merging records

Learn how you can configure the `MergeRecord` processor for your S3 ingest data flow. You can use it to merge together multiple record-oriented flow files into a large flow file that contains all records of your Kafka data input.

Procedure

1. Launch the Configure Processor window, by right clicking the `MergeRecord` processor and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.
3. When you have finished configuring the options you need, save the changes by clicking the Apply button.

   Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

   In this example the following settings and properties are used:

Table 6: MergeRecord processor scheduling

Scheduling	Description	Example value for ingest data flow
Automatically Terminate Relationships		original

Table 7: MergeRecord processor properties

Property	Description	Example value for ingest data flow
RecordReader	Specify the Controller Service to use for reading incoming data.	ReadCustomerCSVRoot
RecordWriter	Specify the Controller Service to use for writing out the records.	CSVRecordSetWriter
Merge Strategy	Specify the algorithm used to merge records. The Bin-Packing Algorithm generates a FlowFile populated by arbitrarily chosen FlowFiles.	Bin-Packing Algorithm
Minimum Number of Records	Specify the minimum number of records to include in a bin.	900

Property	Description	Example value for ingest data flow
Maximum Number of Records	Specify the maximum number of Records to include in a bin.	1000

For a complete list of `MergeRecord` properties, see the *processor documentation*.

What to do next

Configure the processor for your data target.

Related Information

Configuring a processor

Apache NiFi Documentation

Getting Started with Apache NiFi

Configure the processor for your data target

Learn how you can configure the data target processor for your S3 ingest data flow. This example assumes that you are moving data to AWS S3 and shows you how to configure the corresponding processors.

Procedure

1. Launch the Configure Processor window by right clicking the processor you added for writing data to S3 (`PutHDFS`or `PutS3Object`) and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.

   Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

   Examples:

   - You can use the following properties for `PutS3Object`:

     Table 8: PutS3Object processor properties

Property	Description	Example value for ingest data flow
Object key		${filename}
Bucket	Provide the name of your target bucket in AWS. This is not the bucket related to your data lake(s). You have to create it specifically for the data ingest.	ingest-idbroker-test

Property	Description	Example value for ingest data flow
AWS Credentials Provider Service	Provide here the controller service that is used to obtain AWS credentials provider.	AWSIDBrokerCloudCredentialsProviderControllerService

You can leave all other properties as default configurations.

For a complete list of `PutS3Object` properties, see the *processor documentation*.

- You can use the following properties for `PutHDFS`:

Table 9: PutHDFS processor properties

Property	Description	Example value for ingest data flow
Hadoop Configuration Resources	Specify the path to the core-site.xml configuration file. Make sure that the default file system (fs, default.FS) points to the S3 bucket you are writing to. **Note:** The core-site.xml file used for the Hadoop Configuration Resources property is present on every Flow Management cluster. Cloudera Manager stores the right core-site.xml file in the same /etc directory for every cluster.	/etc/hadoop/conf.cloudera.core_settings/core-site.xml
Kerberos Principal	Specify the Kerberos principal (your username) to authenticate against CDP.	srv_nifi-s3-ingest
Kerberos Password	Provide the password that should be used for authenticating with Kerberos.	password
Directory	Provide the path to your target directory in AWS expressed in an S3A compatible path.	s3a://your path/customer

You can leave all other properties as default configurations.

For a complete list of `PutHDFS` properties, see the *processor documentation*.

3. When you have finished configuring the options you need, save the changes by clicking Apply.

   If you want to move data to a different location, review the other use cases in the *Cloudera Data Flow for Data Hub library*.

What to do next

Your data flow is ready to ingest data into AWS S3. Start the flow.

Related Information

Configuring a processor

Apache NiFi Documentation

Data ingest use cases in Cloudera Data Flow for Data Hub

Start the data flow

When your flow is ready, you can begin ingesting data into Amazon S3 buckets. Learn how to start your S3 ingest data flow.

Procedure

1. To initiate your data flow, select all the data flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternatively, right-click a single component and choose Start from the context menu. Data should be read from Kafka and it should be written to the defined folder of your AWS S3 bucket

Results

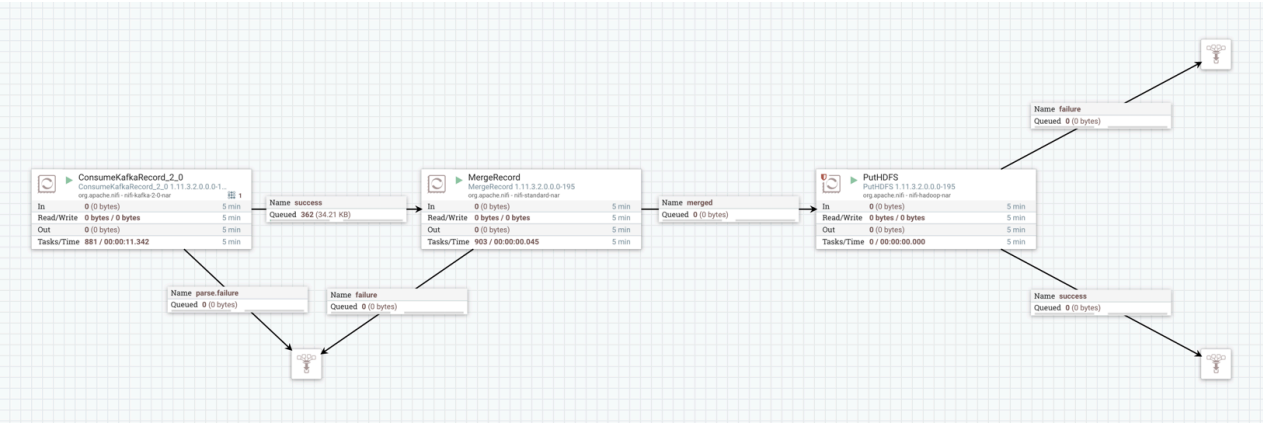

What to do next

It is useful to check that data is running through the flow you have created.

Verify data flow operation

Learn how you can verify the operation of your S3 ingest data flow.

About this task

There are a number of ways you can check that data is running through the flow you have built.

Procedure

1. You can verify that NiFi processors are not producing errors.
2. You can look at the processors in the UI, where you can see the amount of data that has gone through them. You can also right click on the processors, or on connections to view status history.
3. You can check that the data generated appears in your AWS S3 bucket. To do this, return to the dedicated folder of your S3 bucket, where you should see your files listed.

   You may have to refresh the page depending on your browser/settings.

Monitoring your data flow

Learn about the different monitoring options for your Amazon S3 ingest data flow in CDP Public Cloud.

You can monitor your data flow for information about health, status, and details about the operation of processors and connections. NiFi records and indexes data provenance information, so you can conduct troubleshooting in real time.

Data statistics are rolled up on a summary screen (the little table icon on the top right toolbar which lists all the processors). You can use the `MonitorActivity` processor to alert you, if for example you have not received any data in your flow for a specified amount of time.

If you are worried about data being queued up, you can check how much data is currently queued. Process groups also conveniently show the totals for any queues within them. This can often indicate if there is a bottleneck in your flow somewhere, and how far the data has got through that pipeline.

Another option to check that data has fully passed through your flow is to check out data provenance to see the full history of your data.

Next steps

Learn about the different options that you have after building a simple Amazon S3 ingest data flow in CDP Public Cloud.

Moving data to the cloud is one of the cornerstones of any cloud migration. Cloud environments offer numerous deployment options and services. This example data flow provides you with a model to design more complex data flows for moving and processing data as part of cloud migration efforts.

You can build a combination of on-premise and public cloud data storage. You can use this solution as a path to migrate your entire data to the cloud over time—eventually transitioning to a fully cloud-native solution or to extend your existing on-premise storage infrastructure, for example for a disaster recovery scenario. Cloud storage can provide secure, durable, and extremely low-cost options for data archiving and long-term backup for on-premise datasets.

You can also use cloud services without storing your data in the cloud. In this case you would continue to use your legacy on-premise data storage infrastructure, and work with on-demand, cloud-based services for data transformation, processing and analytics with the best performance, reliability and cost efficiency for your needs. This way you can manage demand peaks, provide higher processing power, and sophisticated tools without the need to permanently invest in computer hardware.

Cloudera DataFlow for Data Hub 7.3.1

Ingesting Data into Azure Data Lake Storage

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ingesting data into Azure Data Lake Storage.....4
Understand the use case.....4
Meet the prerequisites.....4
Build the data flow.....5
Create IDBroker mapping.....7
Create controller services for your data flow.....8
Configure the processor for your data source.....10
Configure the processor for merging records.....12
Configure the processor for your data target.....13
Start the data flow.....15
Verify data flow operation.....16

Monitoring your data flow.....16

Next steps.....17

Ingesting data into Azure Data Lake Storage

You can use an Apache NiFi data flow to ingest data into Azure Data Lake Storage (ADLS) in CDP Public Cloud by following these steps.

Understand the use case

Learn how you can use a Flow Management cluster connected to a Streams Messaging cluster to build an end-to-end flow that ingests data to Azure Data Lake Storage (ADLS). This example use case shows you how to use Apache NiFi to move data from Kafka to ADLS.

Why move data to object stores?

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost/effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Apache NiFi for cloud ingestion

You can use Apache NiFi to move data from a range of locations into object stores in CDP Public Cloud.

As a part of Cloudera Data Flow, NiFi offers a scalable solution for managing data flows with guaranteed delivery, data buffering / pressure release, and prioritized queuing. It supports data transfer from nearly any type of data source to cloud storage solutions in a secure and governed manner.

You can use NiFi to move data from various locations into Azure Data Lake Storage (ADLS) in a quick and user-friendly way, by simply dragging and dropping a series of processors on the NiFi user interface. When the data flow is ready, you can visually monitor and control the pipeline you have built.

This use case walks you through the steps associated with creating an ingest-focused data flow from Apache Kafka into ADLS folders. If you are moving data from a location other than Kafka, see the Getting Started with Apache NiFi for information about how to build a data flow, and about other data get and consume processor options.

Related Information

Getting Started with Apache NiFi
Ingesting Data into Apache Kafka
Ingesting Data into Apache HBase in CDP Public Cloud
Ingesting data into Apache Hive in CDP Public Cloud
Ingesting Data into Apache Kudu in CDP Public Cloud
Ingesting data into Amazon S3

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a CDP Public Cloud environment.
- You have a CDP username and password set to access Data Hub clusters.

  The predefined resource role of this user is at least EnvironmentUser. This resource role provides the ability to view Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the CDP Public Cloud environment.
- You have a Flow Management Data Hub cluster running in your CDP Public Cloud environment.

- You have a Streams Messaging Data Hub cluster in the same CDP environment as the Flow Management cluster.
- Your CDP user has been added to the appropriate pre-defined Ranger access policies to allow access to the NiFi UI.
- You have set up a consumer group in Ranger, and you have a Ranger policy for Kafka allowing your CDP user to access this consumer group.
- You have a Ranger policy on your Kafka topic you will use as a data source allowing the CDP user to consume from the topic.
- You have created a target folder in your Azure Data Lake Storage (ADLS) account for the data to be moved to the cloud.
- You have created a Managed Service Identity in Azure with permissions to write to the ADLS target folder.

Related Information

Understanding roles and resource roles

Setting up your Flow Management cluster

Configuring Ranger policies for Apache NiFi

IDBroker

Build the data flow

Learn how you can create an ingest data flow to move data from Kafka to ADLS. This involves opening Apache NiFi in your Flow Management cluster, adding processors and other data flow objects to your canvas, and connecting your data flow elements.

About this task

You can use the `PutHDFS` or `PutAzureDataLakeStorage` processors to build your ADLS ingest data flows. Regardless of the type of flow you are building, the first steps in building your data flow are generally the same. Open NiFi, add your processors to the canvas, and connect the processors to create the flow.

Procedure

1. Open NiFi
   in
   Data Hub.
   a) To access the NiFi service in your Flow Management
      Data Hub cluster, navigate to Management Console serviceData Hub Clusters.
   b) Click the tile representing the Flow Management Data Hub cluster you want to work with.
   c) Click Nifi in the Services section of the cluster overview page to access the NiFi UI.

   You will be logged into NiFi automatically with your CDP credentials.
2. Add the `ConsumeKafkaRecord_2_0` processor for data input in your data flow.
   a) Drag and drop the processor icon into the canvas.

      This displays a dialog that allows you to choose the processor you want to add.
   b) Select the `ConsumeKafkaRecord_2_0` processor from the list.
   c) Click Add or double-click the required processor type to add it to the canvas.

   Note: This processor consumes messages from Kafka specifically built against the Kafka 2.0 Consumer API.

3. Add the `MergeRecord` processor.

   This processor merges together multiple record-oriented flowfiles into a single flowfile that contains all of the records of the input flowfiles. It works by creating bins and adding flowfiles to these bins until they are full.

   Once a bin is full, all of the flowfiles are combined into a single output flowfile, and that flowfile is routed to the merged relationship.

   When using the `ConsumeKafkaRecord_2_0` processor, you are pulling small-sized records, so it is practical to merge them into larger files before writing them to ADLS.

4. Add a processor for writing data to ADLS.

Option	
`PutHDFS` processor	The HDFS client writes to ADLS through the Azure Data Lake Storage REST API. This solution leverages centralized CDP security. You can use CDP usernames and passwords for central authentication, and all requests go through IDBroker.
`PutAzureDataLakeStorage` processor	It is an ADLS-specific processor that provides built-in support for Azure Data Lake Storage Gen 2. This processor is not integrated into CDP's authentication and authorization frameworks, it uses Azure's built-in authentication technology. If you use it for data ingestion, you need to provide the Azure access credentials file / information when configuring the processor. You can do this by providing your storage account name and storage account key.

   a) Drag and drop the processor icon into the canvas.

      This displays a dialog that allows you to choose the processor you want to add.
   b) Select the processor of your choice from the list.
   c) Click Add or double-click the required processor type to add it to the canvas.

5. Connect the processors to create the data flow by clicking the connection icon in the first processor, and dragging and dropping it on the second processor.

   A Create Connection dialog appears with two tabs: Details and Settings. You can configure the connection's name, flowfile expiration time period, thresholds for back pressure, load balance strategy and prioritization.

   a) Connect `ConsumeKafkaRecord_2_0` with MergeRecord.
   b) Add the success flow of the `ConsumeKafkaRecord_2_0` processor to the `MergeRecord` processor.
   c) Click Add to close the dialog box and add the connection to your data flow.
   d) Connect `MergeRecord` with your target data processor (`PutHDFS` / `PutAzureDataLakeStorage`).
   e) Add the merged flow of the `MergeRecord` processor to the target data processor.
   f) Click Add to close the dialog box and add the connection to your data flow.

6. Optionally, you can add funnels to your flow.

   a) `ConsumeKafkaRecord_2_0`: If any of the Kafka messages are pulled but cannot be parsed, the continents of the message will be written to a separate flowfile and that flowfile will be transferred to the parse.failure relationship. You can connect the failure queue coming from the processor to a failure funnel for parse failures.
   b) `MergeRecord`: For merge failures, you can connect the failure queue coming from the processor to the failure funnel you used for ConsumeKafkaRecord_2_0.
   c) `PutHDFS` / `PutAzureDataLakeStorage`: You can add success and failure funnels at the end of the data flow and connect them with your target data processor. These funnels help you see where flow files are being routed when your flow is running.

   If you want to know more about working with funnels, see the Apache NiFi User Guide.

Results

This example data flow has been created using the `PutHDFS` processor.

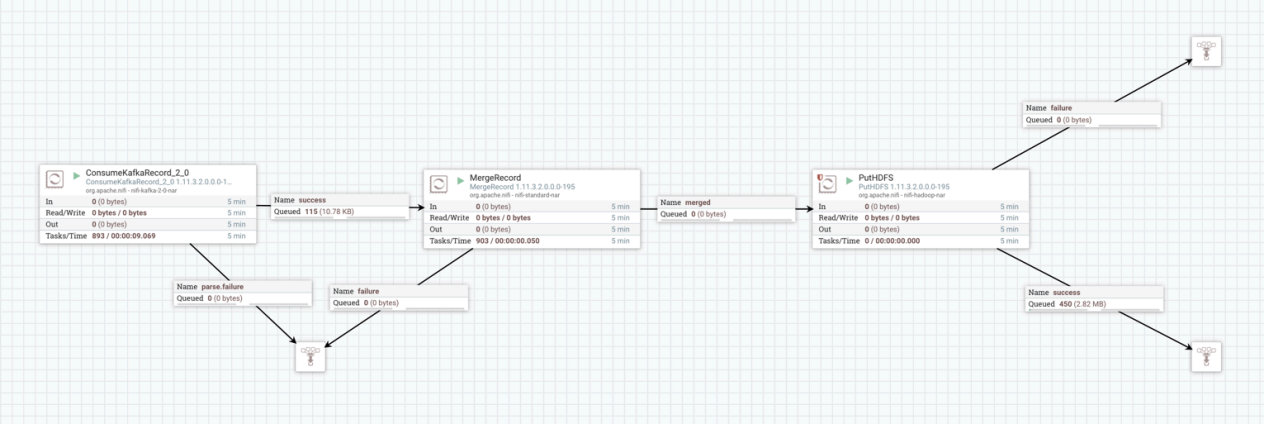

What to do next

If you are using the `PutHDFS` processor, configure IDBroker mapping authorization.

If you are using the `PutAzureDataLakeStorage`processor, you do not need IDBroker mapping, so proceed to configuring your processors in your data flow.

Related Information

Apache NiFi Documentation

Create IDBroker mapping

Learn how you can create the IDBroker mapping for the `PutHDFS` processor for your data flow. To enable your CDP user to utilize the central authentication features CDP provides and to exchange credentials for AWS access tokens, you have to map your CDP user to the correct IAM role.

About this task

This task is only needed if you use the `PutHDFS` processor for accessing your ADLS folder.

The option to add / modify these mappings is available from the Management Console in your CDP environment.

Procedure

1. Access IDBroker Mappings.
   a) To access IDBroker Mappings in your environment, click ActionsManage Access.
   b) Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.

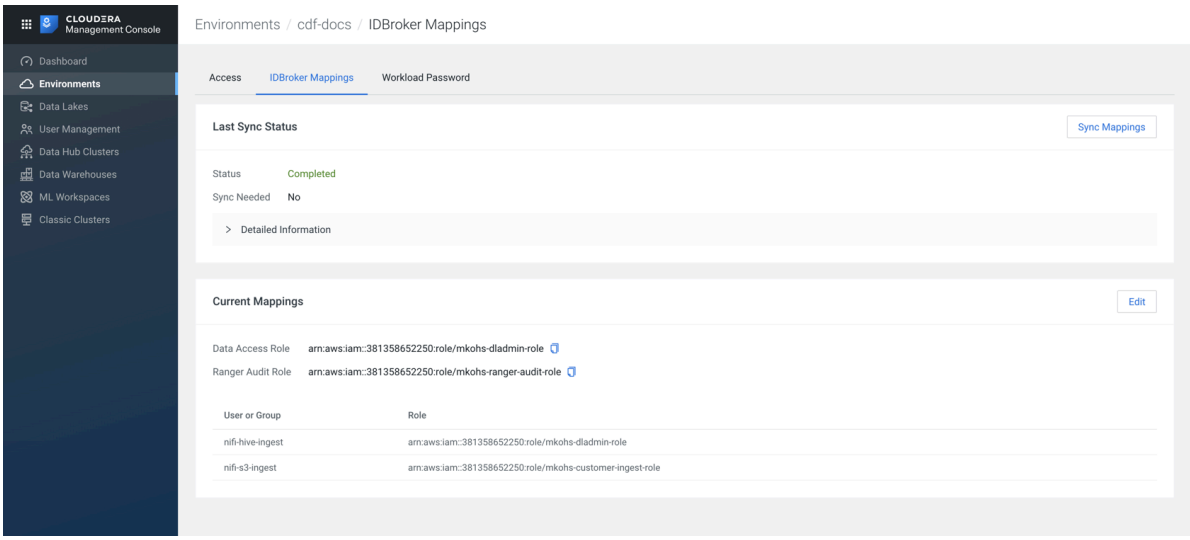

2. Add your CDP user or group and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

   You can get the Azure Managed Identity Resource ID in your ADLS Storage Explorer by navigating to Managed Identities Your Managed IdentityPropertiesResource ID. The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
3. Click Save and Sync.

What to do next

Configure controller services for your data flow.

Related Information

IDBroker

Onboarding CDP users and groups for cloud storage

Create a provisioning credential for Azure

Create controller services for your data flow

Learn how you can create and configure controller services for an ADLS ingest data flow in CDP Public Cloud. Controller services provide shared services that can be used by the processors in your data flow. You will use these Controller Services later when you configure your processors.

Procedure

1. To add a Controller Service to your flow, right-click on the Canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.

3. Click the + button to display the Add Controller Service dialog.

4. Select the required Controller Service and click Add.

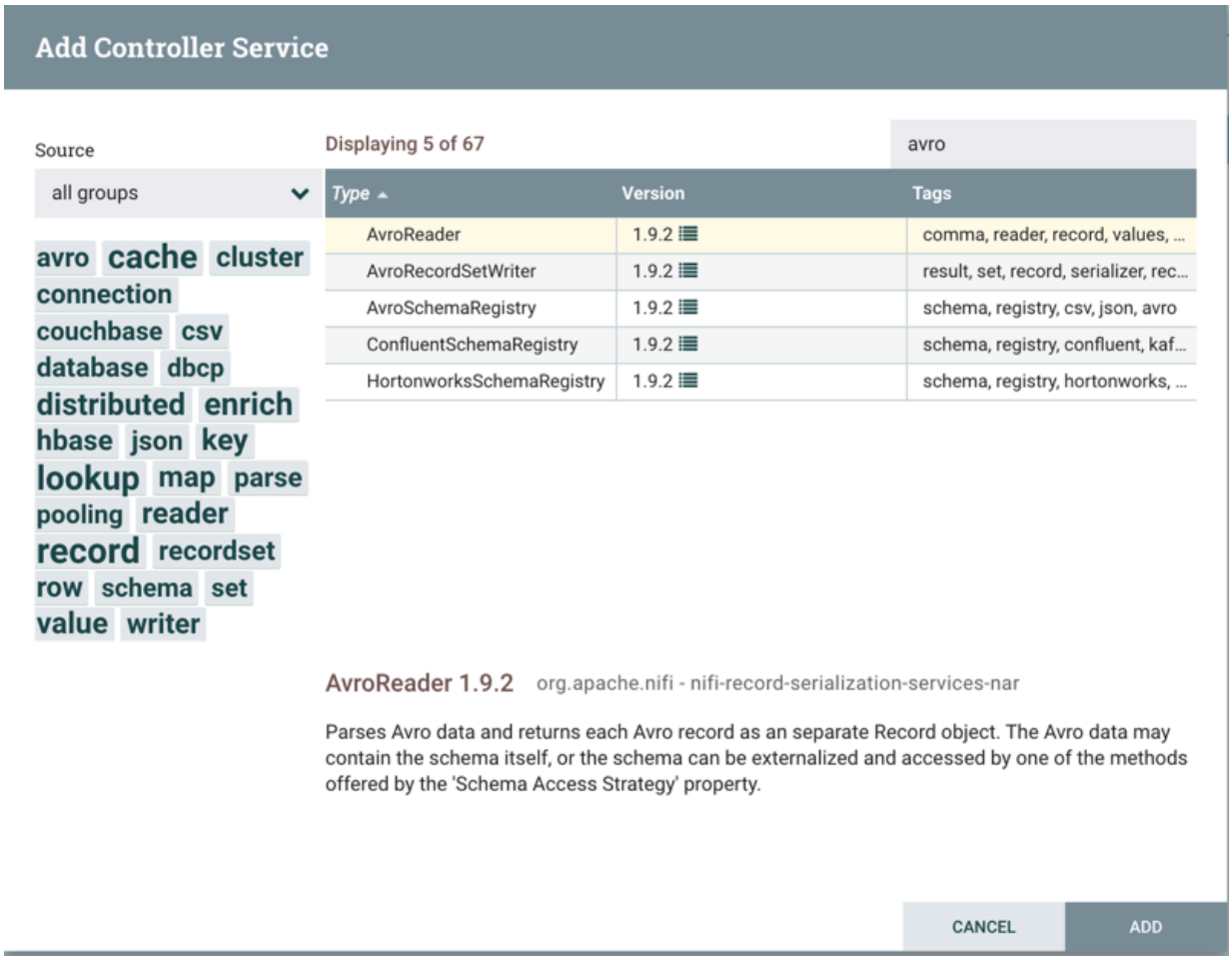

5. Perform any necessary Controller Service configuration tasks by clicking the Configure icon in the right-hand column.

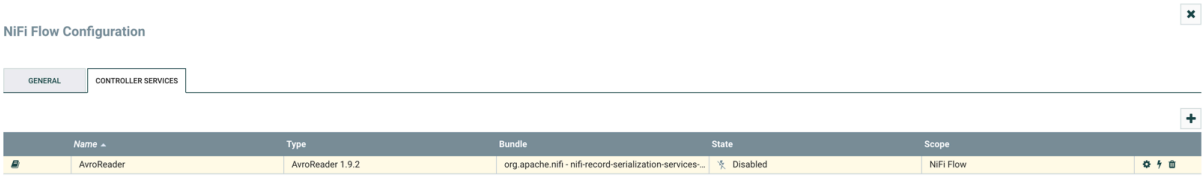

6. When you have finished configuring the options you need, save the changes by clicking the Apply button.

7. Enable the Controller Service by clicking the Enable button (flash) in the far-right column of the Controller Services tab.

Example

In this example we are using the following controller services:

AvroReader Controller Service

Table 1: AvroReader Controller Service properties

Property	Description	Example value for ingest data flow
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	HWX Content-Encoded Schema Reference
Schema Registry	Specify the Controller Service to use for the Schema Registry.	CDPSchemaRegistry
Schema Name	Specify the name of the schema to look up in the Schema Registry property.	customer

CSVReader Controller Service

Table 2: CSVReader Controller Service Properties

Property	Description	Example value for ingest data flow
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	Use String Fields From Header
Treat First Line as Header	Specify whether the first line of CSV should be considered a header or a record.	true

CSVRecordSetWriter Controller Service

Table 3: CSVRecordSetWriter Controller Service Properties

Property	Description	Example value for ingest data flow
Schema Write Strategy	Specify how the schema for a record should be added to the data.	Do Not Write Schema
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	Use 'Schema Name' Property
Schema Name	Specify the name of the schema to look up in the Schema Registry property. See the Appendix for an example schema.	customer

What to do next

Configure the processors in your data flow.

Related Information

Adding Controller Services for data flows

Apache NiFi Documentation

Configure the processor for your data source

Learn how you can configure the `ConsumeKafkaRecord_2_0` data source processor for your ADLS ingest data flow. You can set up a data flow to move data to Azure Data Lake Storage from many different locations. This

example assumes that you are streaming data from Kafka and shows you the configuration for the relevant data source processor.

Procedure

1. Launch the Configure Processor window, by right-clicking the `ConsumeKafkaRecord_2_0` processor and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.
3. When you have finished configuring the options you need, save the changes by clicking the Apply button.

   Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

Example

In this example data flow, the data source is Kafka. You can create the modified Kafka broker URLs using the broker hostnames and adding port :9093 to the end of each FQDN. You can find the hostnames on the Streams Messaging cluster overview page when selecting the Hardware tab.

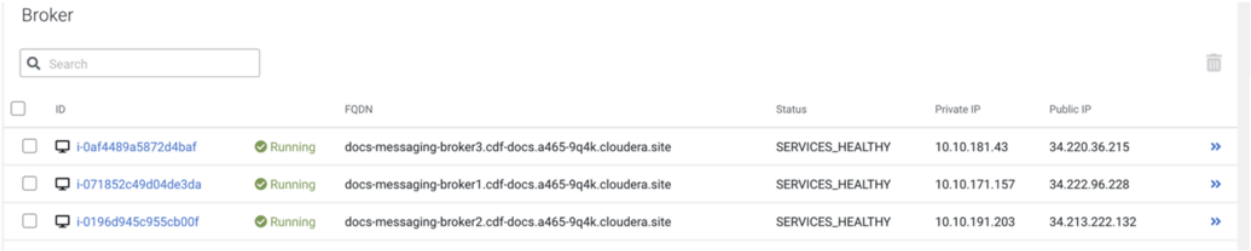

Note: Property values can be parameterized. For example, you can create a parameter context to hold Kafka connection information and apply it to the Kafka Brokers property instead of adding the broker URLs individually.

The following table includes a description and example values for the properties required to configure the example ingest data flow. For a complete list of `ConsumeKafkaRecord_2_0` properties, see the *Apache Nifi Documentation*.

Table 4: ConsumeKafkaRecord_2_0 processor properties

Property	Description	Example value for ingest data flow
Kafka Brokers	Provide a comma-separated list of known Kafka Brokers. In the format: <host>:<port>	Docs-messaging-broker1.cdf-docs.a465-9q4k.cloudera.site:9093,docs-messaging-broker2.cdf-docs.a465-9q4k.cloudera.site:9093,docs-messaging-broker3.cdf-docs.a465-9q4k.cloudera.site:9093
Topic Name(s)	Provide the name of the Kafka Topic(s) to pull from.	Customer
Record Reader	Specify the Record Reader to use for incoming FlowFiles.	AvroReader
Record Writer	Specify the Record Writer to use in order to serialize the data before sending to Kafka.	CSVRecordSetWriter
Security Protocol	Specify the protocol used to communicate with Kafka brokers.	SASL_SSL
SASL Mechanism	Specify the SASL mechanism to use for authentication.	PLAIN

Property	Description	Example value for ingest data flow
Username	Use your CDP workload username to set this Authentication property.	srv_nifi-kafka-ingest
Password	Use your CDP workload password to set this Authentication property.	password
SSL Context Service	Specify the SSL Context Service to use for communicating with Kafka. Use the pre-configured SSLContextProvider.	Default NiFi SSL Context Service
Group ID	Provide the consumer group ID to identify consumers that are within the same consumer group.	nifi-adls-ingest

Note: If you want to move data to ADLS from a location other than Kafka, see the *Apache NiFi Getting Started* for information about other data ingest processor options.

What to do next

Configure the processor for merging your records.

Related Information

Configuring a Processor

Apache NiFi Documentation

Getting Started with Apache NiFi

Configure the processor for merging records

Learn how you can configure the `MergeRecord` processor for your ADLS ingest data flow. You can use it to merge together multiple record-oriented flow files into a large flow file that contains all records of your Kafka data input.

Procedure

1. Launch the Configure Processor window, by right clicking the `MergeRecord` processor and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.
3. When you have finished configuring the options you need, save the changes by clicking Apply.

   Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

Example

In this example the following settings and properties are used:

Table 5: MergeRecord processor scheduling

Scheduling	Description	Example value for ingest data flow
Automatically Terminate Relationships		original

Table 6: MergeRecord processor properties

Property	Description	Example value for ingest data flow
RecordReader	Specify the Controller Service to use for reading incoming data.	CSVReader

Property	Description	Example value for ingest data flow
RecordWriter	Specify the Controller Service to use for writing out the records.	CSVRecordSetWriter
Merge Strategy	Specify the algorithm used to merge records. The Bin-Packing Algorithm generates a FlowFile populated by arbitrarily chosen FlowFiles.	Bin-Packing Algorithm
Minimum Number of Records	Specify the minimum number of records to include in a bin.	900
Maximum Number of Records	Specify the maximum number of Records to include in a bin.	1000

What to do next

Configure the processor for your data target.

Related Information

Configuring a Processor

Apache NiFi Documentation

Getting Started with Apache NiFi

Configure the processor for your data target

Learn how you can configure the data target processor for your ADLS ingest data flow. This example assumes that you are moving data to Azure Data Lake Storage and shows you how to configure the corresponding processors.

Procedure

1. Launch the Configure Processor window by right clicking the processor you added for writing data to ADLS (`PutHDFS` or `PutAzureDataLakeStorage`) and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.

   Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

3. When you have finished configuring the options you need, save the changes by clicking the Apply button.

   Note: Cloudera recommends using the `PutHDFS` processor for writing data to ADLS in CDP Public Cloud as it leverages the CDP authentication framework.

   In this example, the following properties are used for `PutHDFS`:

Table 7: PutHDFS processor properties

Property	Description	Example value for ingest data flow
Hadoop Configuration Resources	Specify the path to the core-site.xml configuration file. Make sure that the default file system (fs, default.FS) points to the ADLS bucket you are writing to. **Note:** The core-site.xml file is present on every Flow Management cluster. Cloudera Manager stores the right core-site.xml file in the same /etc directory for every cluster.	/etc/hadoop/conf.cloudera.core_settings/core-site.xml
Kerberos Principal	Specify the Kerberos principal (your CDP username) to authenticate against CDP.	srv_nifi-adls-ingest
Kerberos Password	Provide the password that should be used for authenticating with Kerberos.	password

Property	Description	Example value for ingest data flow
Directory	Provide the path to your target directory in Azure expressed in an abfs compatible path format.	abfs://<YourFileSystem>@<YourStorageAccount>.dfs.core.windows.net/<TargetPathWithinFileSystem>

You can leave all other properties as default configurations.

For a complete list of `PutHDFS` properties, see the *processor documentation*.

If you want to use the `PutAzureDataLakeStorage` processor to store the data in ADLS, you have to configure your Azure connection in a secure way by providing:

Note:

`PutAzureDataLakeStorage` has several limitations:

- You can add files to read-only buckets
- There is no check for file overwriting. It is possible to overwrite data.
- To add files to a bucket root level, set the destination with an empty string, rather than " / ".

- Storage Account Name - the name of your Azure storage account that holds the containers where you want to write to)
- Storage Account Key or your SAS Token - the authentication key that allows you to write data to the Azure storage account

  Note: By default, all resources in Azure Storage are secured, and are available only to the account owner. You can grant clients general access to your storage account sharing the storage account key, or you can give more granular access for certain users to specific resources in your storage account using the SAS Token.
- Filesystem Name - the name of your Azure storage account file system where you want to write to
- Directory Name - the name of the folder within your filesystem where you want to write to

Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

What to do next

Your data flow is ready to ingest data into ADLS. Start the flow.

Related Information

Configuring a processor

Apache NiFi Documentation

Data ingest use cases in Cloudera Data Flow for Data Hub

Start the data flow

When your flow is ready, you can begin ingesting data into Azure Data Lake Storage folders. Learn how to start your ADLS ingest data flow.

Procedure

1. To get your flow to work, select all the data flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternatively, right-click a single component and choose Start from the context menu. Data should be read from Kafka and it should be written to your predefined Azure ADLS folder.

Results

This example data flow has been created using the `PutHDFS` processor.

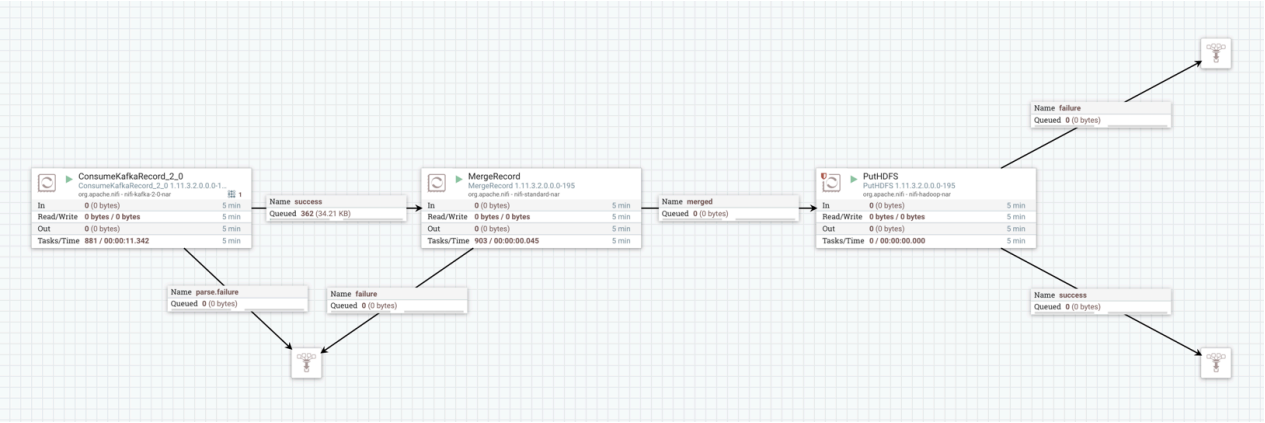

What to do next

It is useful to check that data is running through the flow you have created without any errors.

Verify data flow operation

Learn how you can verify the operation of your ADLS ingest data flow.

About this task

There are a number of ways you can check that data is running through the flow you have built.

Procedure

1. You can verify that NiFi processors are not producing errors.
2. You can look at the processors in the UI, where you can see the amount of data that has gone through them. You can also right click on the processors, or on connections to view status history.
3. To verify that the data generated appears in ADLS folder specified in the processor, return to your ADLS Storage Explorer, where you should see your files listed in the target folder of your file system.

   You may have to refresh the page depending on your browser/settings.

Monitoring your data flow

Learn about the different monitoring options for your ADLS ingest data flow in CDP Public Cloud.

You can monitor your data flow for information about health, status, and details about the operation of processors and connections. NiFi records and indexes data provenance information, so you can conduct troubleshooting in real time.

Data statistics are rolled up on a summary screen (the little table icon on the top right toolbar which lists all the processors). You can use the `MonitorActivity` processor to alert you, if for example you have not received any data in your flow for a specified amount of time.

If you are worried about data being queued up, you can check how much data is currently queued. Process groups also conveniently show the totals for any queues within them. This can often indicate if there is a bottleneck in your flow somewhere, and how far the data has got through that pipeline.

Another option to check that data has fully passed through your flow is to check out data provenance to see the full history of your data.

Next steps

Learn about the different options that you have after building a simple ADLS ingest data flow in CDP Public Cloud.

Moving data to the cloud is one of the cornerstones of any cloud migration. Cloud environments offer numerous deployment options and services. This example data flow enables you to easily design more complex data flows for moving and processing data as part of cloud migration efforts.

You can build a combination of on-premise and public cloud data storage. You can use this solution as a path to migrate your entire data to the cloud over time—eventually transitioning to a fully cloud-native solution or to extend your existing on-premise storage infrastructure, for example for a disaster recovery scenario. Cloud storage can provide secure, durable, and extremely low-cost options for data archiving and long-term backup for on-premise datasets.

You can also use cloud services without storing your data in the cloud. In this case you would continue to use your legacy on-premise data storage infrastructure, and work with on-demand, cloud-based services for data transformation, processing and analytics with the best performance, reliability and cost efficiency for your needs. This way you can manage demand peaks, provide higher processing power, and sophisticated tools without the need to permanently invest in computer hardware.

Cloudera DataFlow for Data Hub 7.3.1

Ingesting Data into Cloudera Data Warehouse using Iceberg Table Format

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ingesting data into Cloudera Data Warehouse using Iceberg table format 4

Understand the use case 4
Meet the prerequisites 4
Create the Iceberg target table in Cloudera Data Warehouse 4
Build the data flow 5
Create and configure controller services for your data flow 6
Configure the processor for your data source 7
Configure the processor for your data target 8
Start the data flow 9

Ingesting data into Cloudera Data Warehouse using Iceberg table format

You can use an Apache NiFi data flow to ingest data in Iceberg table format into Cloudera Data Warehouse in Cloudera Public Cloud by following these steps.

Understand the use case

Learn how to use NiFi to move data to an Iceberg table of a data warehouse in Cloudera Public Cloud.

Cloudera Data Warehouse can be configured to use Iceberg as the table format for querying data with Impala and/or Hive. You can use Apache NiFi to move data from a range of locations into a Data Warehouse cluster running Apache Hive or Impala in Cloudera Public Cloud.

This use case walks you through creating a data flow that generates FlowFiles with random CSV data and writes this data into an Iceberg table in Hive or Impala.This gets you started with creating an Iceberg ingest data flow. If you want to use a different data source, you can check out the other ingest documents for get and consume data processor options.

For more information on Iceberg table format, see *Apache Iceberg features*.

Related Information

Apache Iceberg features

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a Cloudera Public Cloud environment.
- You have a workload username and password set to access Cloudera Data Hub clusters. The predefined resource role of this user is at least "EnvironmentUser". This resource role provides the ability to view Cloudera Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the Cloudera Public Cloud environment.
- You have a Flow Management cluster running Apache NiFi.
- You have a running Data Warehouse cluster in the same Cloudera Public Cloud environment properly configured to support Iceberg tables.

Create the Iceberg target table in Cloudera Data Warehouse

Learn how to create an Iceberg table for your data flow.

About this task

Before you can ingest data into your Cloudera Data Warehouse target in Cloudera Public Cloud, you need to create the table where you want to send data with NiFi. The following example shows how to create a table with Impala:

```
CREATE TABLE customer (id int, name string, created_at timestamp)
PARTITIONED BY (country_code string)
STORED BY ICEBERG;
```

Build the data flow

Learn how to create an ingest data flow to move data to an Iceberg table in Hive or Impala.

About this task

Building a data flow to move data to an Iceberg table in Hive or Impala involves using NiFi in your Flow Management cluster. The example data flow generates FlowFiles with random CSV data and writes this data into an Iceberg table.

Procedure

1. Open NiFi
   in
   Data Hub.
   a) To access the NiFi service in your Flow Management
      Data Hub cluster, navigate to Management Console service Data Hub Clusters .
   b) Click the tile representing the Flow Management Data Hub cluster you want to work with.
   c) Click the NiFi icon in the Services section of the cluster overview page to access the NiFi UI.

      You are logged into NiFi automatically with your Cloudera credentials.
2. Add the `GenerateFlowFile` processor for data input. This processor creates FlowFiles with random data or custom content.
   a) Drag and drop the processor icon into the canvas. This displays a dialog that allows you to choose the processor you want to add.
   b) Select the `GenerateFlowFile` processor from the list.
   c) Click Add or double-click the required processor type to add it to the canvas.

   You will configure the `GenerateFlowFile` processor to define how to create the sample data in *Configure the processor for your data source*.
3. Add the `PutIceberg` processor for data output.

   You will configure the `PutIceberg` processor in *Configure the processor for your data target*.
4. Connect the two processors to create a flow.
   a) Drag the connection icon from the first processor, and drop it on the second processor.

      A Create Connection dialog appears with two tabs: Details and Settings.
   b) Configure the connection.

      You can configure the connection's name, flowfile expiration time period, thresholds for back pressure, load balance strategy and prioritization.
   c) Click Add to close the dialog box and add the connection to your data flow.
5. You can add success and failure funnels to your data flow, which help you see where flow files are being routed when your flow is running. Connect the `PutIceberg` to these funnels with success and failure connections.

   If you want to know more about working with funnels, see the *Apache NiFi User Guide*.

Results

Your data flow will look similar to the following:

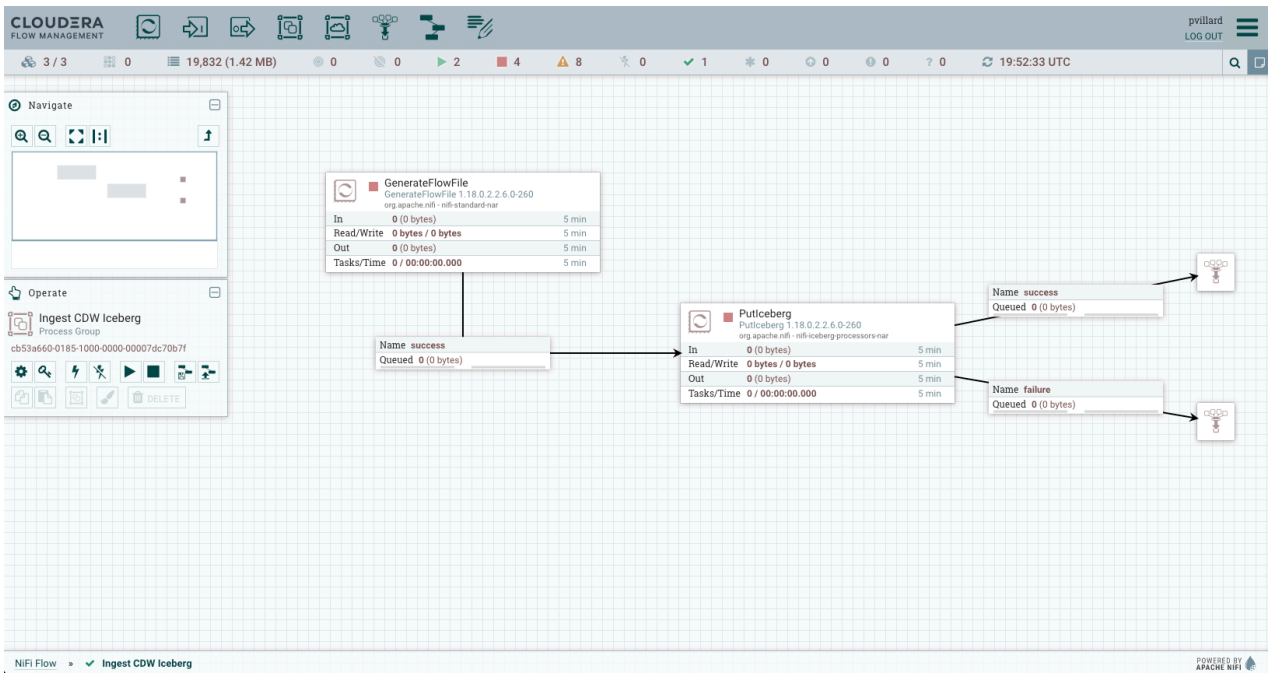

What to do next

Create controller services for your data flow. You will need these services when configuring the `PutIceberg` processor.

Create and configure controller services for your data flow

Learn how to create and configure controller services for the Cloudera Data Warehouse Iceberg ingest data flow.

About this task

You can add controller services to provide shared services to be used by the processors in your data flow. You will use these controller services later when you configure your processors.

Procedure

1. To add a controller service to your flow, right-click on the canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.
3. Click the + button to display the Add Controller Service dialog.
4. Select the required controller service and click Add.
5. Click the Configure icon in the right-hand column and configure the options that you need.
6. When you have finished the configuration, click Apply to save the changes.
7. Enable the controller service by clicking the Enable button (flash) in the far-right column of the Controller Services tab.

Example

The following controller services are used in this Cloudera Data Warehouse Iceberg ingest data flow:

- Hive Catalog Controller Service
- Kerberos User Service

Configure the Hive Catalog Controller Service

In the context of Cloudera Data Warehouse, the Hive Metastore URI is not exposed publicly. Instead, you need to provide the core-site.xml and hive-site.xml files of the Cloudera environment. In Flow Management DataHub clusters, these files are made available automatically on every node but the paths must be retrieved. If you use the Cloudera DataFlow data service and its Flow Designer, you can use the #{CDPEnvironment} parameter.

Note: If you want to push data into Iceberg tables in Data Engineering Data Hub clusters, you can use the Hive Metastore URI instead.

Table 1: Hive Catalog Controller Service properties

Property	Description	Example value for ingest data flow
Hive Metastore URI	Provide the URI of the metastore location.	
Default Warehouse Location	Provide the default warehouse location in the HDFS file system.	
Hadoop Configuration Resources	Add a comma-separated list of Hadoop Configuration files, such as hive-site.xml and core-site.xml for kerberos. Include full paths to the files so that NiFi can refer to those configuration resources from your specified path.	/etc/hive/conf.cloudera.data_context_connector-975b/hive-site.xml,/etc/hadoop/conf.cloudera.stub_dfs/core-site.xml

Configure the Kerberos User Service

Use the Kerberos Password User Service so that you do not need to distribute a keytab file across the NiFi nodes of the cluster.

It is best practice to have a dedicated Machine User created in the control plane for your specific use case so that you can configure specific policies in Ranger and have better control in case of multi-tenancy with many use cases.

Table 2: Kerberos User Service properties

Property	Description	Example value for ingest data flow
Kerberos Principal	Specify the user name that should be used for authenticating with Kerberos. Use your Cloudera workload username to set this Authentication property.	srv_nifi_to_iceberg
Kerberos Password	Provide the password that should be used for authenticating with Kerberos. Use your Cloudera workload password to set this Authentication property.	password (sensitive value)

What to do next

Configure the processor for your data source.

Configure the processor for your data source

Learn how to configure the GenerateFlowFile data source processor for the Cloudera Data Warehouse Iceberg ingest data flow.

About this task

You can set up a data flow to move data in Iceberg table format into Cloudera Data Warehouse from many different locations. This example assumes that you are using sample data generated by the GenerateFlowFile processor.

Procedure

1. Launch the Configure Processor window by right clicking the `GenerateFlowFile` processor and selecting Configure. A configuration dialog with the following tabs is displayed: Settings, Scheduling, Properties, and Comments.
2. Configure the processor according to the behavior you expect in your data flow.

   The `GenerateFlowFile` processor can create many FlowFiles very quickly. Setting the run schedule to a reasonable value is important so that the flow does not overwhelm the system.
3. When you have finished configuring the options you need, save the changes by clicking the Apply button.

   Make sure that you set all required properties, because you cannot start the processor until all mandatory properties have been configured.

Example

The following settings and properties are used in this example:

Table 3: GenerateFlowFile processor scheduling

Scheduling	Description	Example value for ingest data flow
Run Schedule	Run schedule dictates how often the processor should be scheduled to run. The valid values for this field depend on the selected Scheduling Strategy.	500 ms

Table 4: GenerateFlowFile processor properties

	Description	Example value for ingest data flow
Custom text	If Data Format is text and if Unique FlowFiles is false, you can provide custom to be used as the content of the generated FlowFiles. The expression statement in the example value generates a random ID between 1 and 10 000, with random last names assigned.	`customer_id, customer_name ${random():mod(10000):plus(1)}, Smith ${random():mod(10000):plus(1)}, Johnson ${random():mod(10000):plus(1)}, Steward`

What to do next

Configure the processor for your data target.

Configure the processor for your data target

Learn how to configure the target processor of the Cloudera Data Warehouse Iceberg ingest data flow.

About this task

You can set up a data flow to move data to many locations. This example assumes that you are moving data to Cloudera Data Warehouse. Once your data is collected and available in your flow, you can use the PutIceberg processor to send the data in the Cloudera Data Warehouse table.

Procedure

1. Launch the Configure Processor window, by right-clicking the `PutIceberg` processor and selecting Configure.

   A configuration dialog with the following tabs is displayed: Settings, Scheduling, Properties, and Comments.

2. Configure the processor according to the behavior you expect in your data flow.

   With the `PutIceberg` processor, you can ingest data using Iceberg table format.
3. When you have finished configuring the options you need, click Apply to save the changes.

   Make sure that you set all required properties, because you cannot start the processor until all mandatory properties are configured.

Example

The following settings and properties are used in this example:

Table 5: PutIceberg processor properties

Property	Description	Example value for ingest data flow
Record Reader	Select the Record Reader of your choice based on the format of your incoming data that should be pushed in the table.	JSON Reader
Catalog Service	Add the Hive Catalog Controller Service you have created.	Hive Catalog Service
Catalog Namespace	Set the Catalog Namespace property to the name of the database where you created the destination table.	default
Table Name	Set the Table Name property to the name of the table previously created.	customer
Kerberos User Service	Add the Kerberos User Service of your choice (based on username/password or based on username/keytab) that you have created.	Kerberos Password User Service

What to do next

Your data flow is ready to ingest data into an Iceberg table in Cloudera Data Warehouse. You can now start the flow.

Start the data flow

Learn how run the flow and confirm that data is successfully sent to the Iceberg table.

Procedure

1. Select all the data flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternatively, right-click a single component and choose Start from the context menu.

Results

Your flow should be running without any errors. Data should be generated in the form of FlowFiles, and the files should be written to your designated Iceberg table.

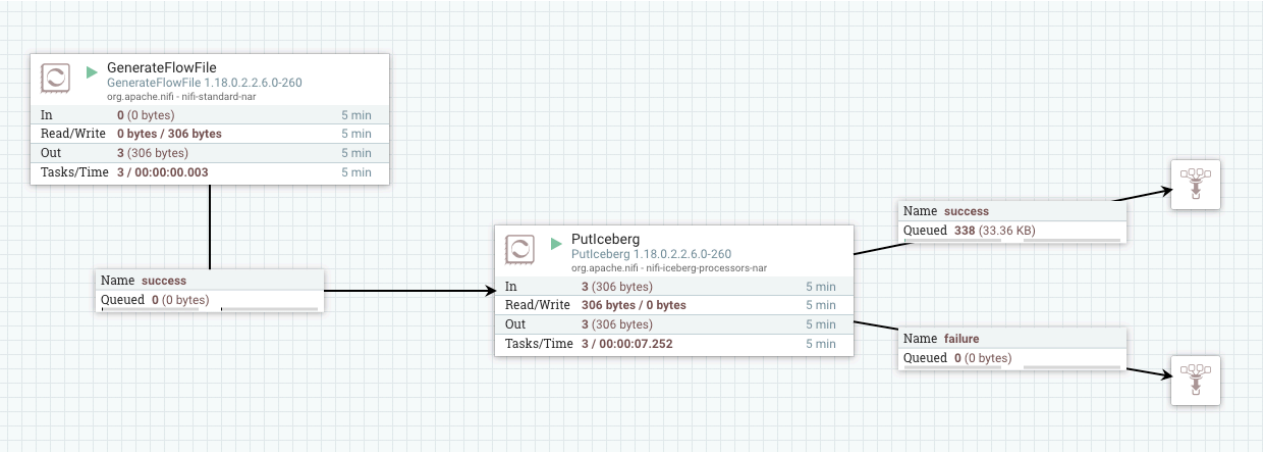

You can verify the operation of your Iceberg ingest data flow by checking that data is running through the flow that you have built and it actually appears in the target table.

- You can verify that NiFi processors are not producing errors.
- You can look at the processors in the UI to see the amount of data that has gone through them. You can also right-click on the processors, or on connections to view status history.
- You can query the data in Hue:

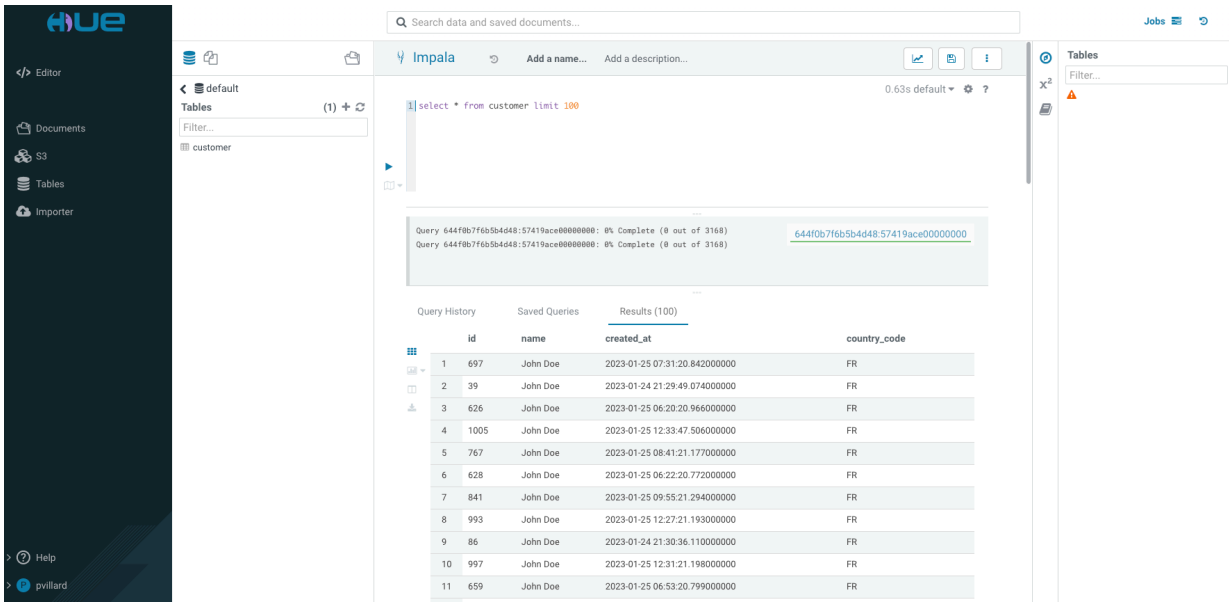

Cloudera DataFlow for Data Hub 7.3.1

Ingesting Data into Google Cloud Storage

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ingesting data into Google Cloud Storage.....4
Understand the use case.....4
Meet the prerequisites.....4
Build the data flow.....5
Create IDBroker mapping.....7
Create controller services for your data flow.....7
Configure the processor for your data source.....8
Configure the processor for merging records.....9
Configure the processor for your data target.....9
Start the data flow.....12
Verify data flow operation.....13

Monitoring your data flow.....13

Viewing data lineage in Apache Atlas.....13

Next steps.....14

Ingesting data into Google Cloud Storage

You can use an Apache NiFi data flow to ingest data into Google Cloud Storage (GCS) object stores in CDP Public Cloud by following these steps.

Understand the use case

Learn how you can use a Flow Management cluster in CDP Public Cloud to build a flow that ingests data to Google Cloud Storage (GCS). This example use case shows you how to use Apache NiFi to move data into GCS buckets.

Why move data to object stores?

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost-effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Apache NiFi for cloud ingestion

As a part of Cloudera Data Flow in CDP Public Cloud, Flow Management clusters run Apache NiFi. You can use NiFi to move data from a range of locations into object stores. NiFi supports data transfer from nearly any type of data source to cloud storage solutions in a secure and governed manner. It also offers a scalable solution for managing data flows with guaranteed delivery, data buffering / pressure release, and prioritized queuing.

NiFi helps you to build a cloud ingestion data flow in a quick and user-friendly way, by simply dragging and dropping a series of processors on the NiFi user interface. When the data flow is ready, you can visually monitor and control the pipeline you have built.

This use case walks you through the steps associated with creating an ingest-focused data flow by tailing the NiFi log files and sending the data into a GCS bucket. If you are moving data from a different source, see the *Getting Started with Apache NiFi* for information about how to build a data flow, and about other data get and consume processor options.

Related Information

Getting Started with Apache NiFi
Ingesting Data into Apache Kafka
Ingesting Data into Apache HBase
Ingesting data into Apache Hive
Ingesting Data into Apache Kudu
Ingesting Data into Azure Data Lake Storage
Ingesting Data into Amazon S3

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a CDP Public Cloud environment.
- You have a CDP username and password set to access Data Hub clusters. The predefined resource role of this user is at least EnvironmentUser. This resource role provides the ability to view Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the CDP Public Cloud environment.
- You have a Flow Management Data Hub cluster running in your CDP Public Cloud environment.

- Your CDP user has been added to the appropriate pre-defined Ranger access policies to allow access to the NiFi UI.
- You have created a target folder in your GCS bucket through the Google Cloud console for the data to be moved to the cloud.
- You have created a GCP service account with the right role and policies to write into the GCS bucket you want to use in your data flow.

Related Information

Understanding roles and resource roles

Setting up your Flow Management cluster

IDBroker - Cloud identity federation

Google Cloud Console User Guide

Build the data flow

Learn how you can create an ingest data flow to move data to Google Cloud Storage (GCS) buckets. This involves opening Apache NiFi in your Flow Management cluster, adding processors and other data flow objects to your canvas, and connecting your data flow elements.

About this task

You can use the `PutHDFS` or `PutGCSObject` processors to build your Google Cloud ingest data flows. Regardless of the type of flow you are building, the first steps in building your data flow are generally the same. Open NiFi, add your processors to the canvas, and connect the processors to create the flow.

Procedure

1. Open NiFi in Data Hub.
   a) To access the NiFi service in your Flow Management Data Hub cluster, navigate to Management Console serviceData Hub Clusters.
   b) Click the tile representing the Flow Management Data Hub cluster you want to work with.
   c) Click Nifi in the Services section of the cluster overview page to access the NiFi UI.

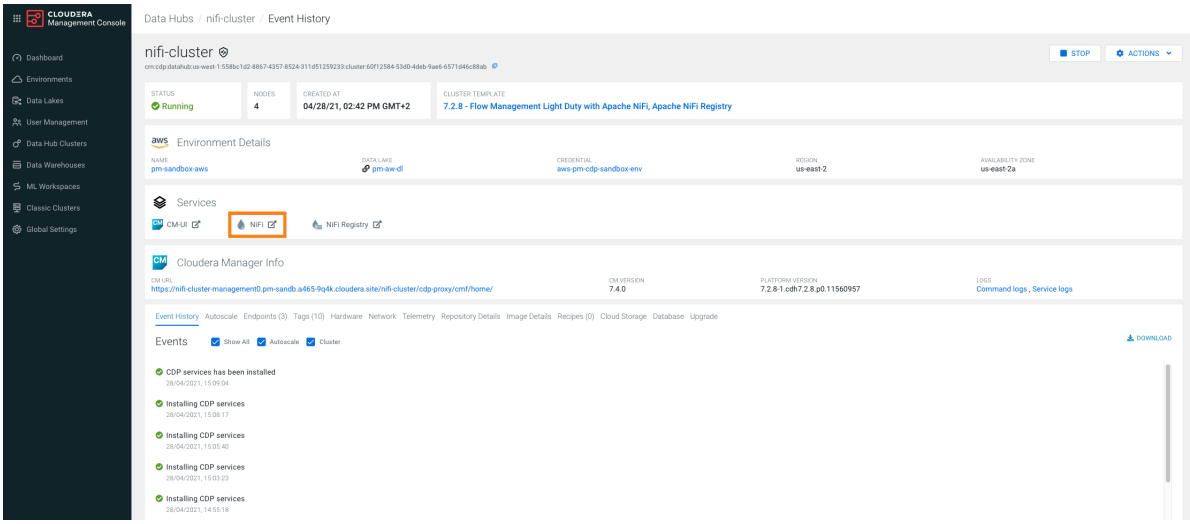

You will be logged into NiFi automatically with your CDP credentials.

2. Add the `TailFile` processor for data input in your data flow.
   a) Drag and drop the processor icon into the canvas.

      This displays a dialog that allows you to choose the processor you want to add.
   b) Select the `TailFile` processor from the list.
   c) Click Add or double-click the required processor type to add it to the canvas.

   Note: This processor continuously tails a file to get its data as flow files.

3. Add the `MergeContent` processor.

   Note: This processor merges a group of flow files together based on a user-defined strategy and packages them into a single flow file. It updates the mime.type attribute as appropriate.

   When using the `TailFile` processor, you are pulling small-sized flow files, so it is practical to merge them into larger files before writing them to GCS.
4. Add a processor for writing data to GCS.

   You have two options:

   - `PutHDFS` processor: The HDFS client writes to GCS through the GCS API. This solution leverages centralized CDP security. You can use the usernames and passwords you set up in CDP for central authentication, and all requests go through IDBroker.
   - `PutGCSObject` processor: This is a GCS-specific processor that directly interacts with the Google Cloud Storage API. It is not integrated into CDP's authentication and authorization frameworks. If you use it for data ingestion, the authorization process is handled in GCP, and you need to provide the Google Cloud access credentials file / information when configuring the processor. You can do this by using the GCPCredentialsControllerService controller service in your data flow.

   a) Drag and drop the processor icon into the canvas.

      In the dialog box you can choose which processor you want to add.
   b) Select the processor of your choice from the list.
   c) Click Add or double-click the required processor type to add it to the canvas.
5. Connect the processors to create the data flow by clicking the connection icon in the first processor, and dragging and dropping it on the second processor.

   A Create Connection dialog appears with two tabs: Details and Settings. You can configure the connection's name, flowfile expiration time period, thresholds for back pressure, load balance strategy and prioritization.

   a) Connect `TailFile` with `MergeContent`.
   b) Add the success flow of the `TailFile` processor to the `MergeContent` processor.
   c) Click Add to close the dialog box and add the connection to your data flow.
   d) Connect `MergeContent` with your target data processor (`PutHDFS` / `PutGCSObject`).
   e) Add the merged flow of the `MergeContent` processor to the target data processor.
   f) Click Add to close the dialog box and add the connection to your data flow.
6. Optionally, you can add funnels to your flow.

   If you want to know more about working with funnels, see the *Apache NiFi User Guide*.

Results

This example data flow has been created using the `PutHDFS` processor.

What to do next

If you are using the `PutHDFS` processor, configure IDBroker mapping authorization.

If you are using the `PutGCSObject`processor, you do not need IDBroker mapping, so proceed to configuring controller services for the processors in your data flow.

Related Information

Apache NiFi Documentation

Create IDBroker mapping

You must create IDBroker mapping for a user or group to access cloud storage. As a part of Knox, the IDBroker allows a user to exchange cluster authentication for temporary cloud credentials. To enable your CDP user to utilize the central authentication features CDP provides and to exchange credentials for Google Cloud access tokens, you have to map your CDP user to the correct IAM role. Learn how you can create the IDBroker mapping for the `PutHDFS` processor for your data flow.

About this task

This task is only needed if you use the `PutHDFS` processor for accessing your GCS bucket.

You have to set the mapping between the CDP user that will be used in your NiFi flow and the GCP service account you created. In the Technical Preview, you can manage ID Broker mappings only using the CDP CLI.

Use the cdp environments set-id-broker-mappings command to set the mappings.

What to do next

Configure controller services for your data flow.

Related Information

IDBroker - Cloud identity federation

Onboarding CDP users and groups for cloud storage

Using CDP CLI

Create controller services for your data flow

Controller services provide shared services that can be used by the processors in your data flow. You will use these Controller Services later when you configure your processors. Learn how you can create and configure the controller services for a Google Cloud ingest data flow in CDP Public Cloud.

Before you begin

This task is only needed if you use the `PutGCSObject` processor for accessing your GCS bucket.

Procedure

1. To add a Controller Service to your flow, right-click on the canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.
3. Click the + button to display the Add Controller Service dialog.

4. Select the required controller service and click Add.

   In this case, add the GCPCredentialsControllerService controller service and configure it by providing the credentials file of the GCP service account you created.

   Note: The credentials file is usually a JSON file.

5. Perform any necessary Controller Service configuration tasks by clicking the Configure icon in the right-hand column.

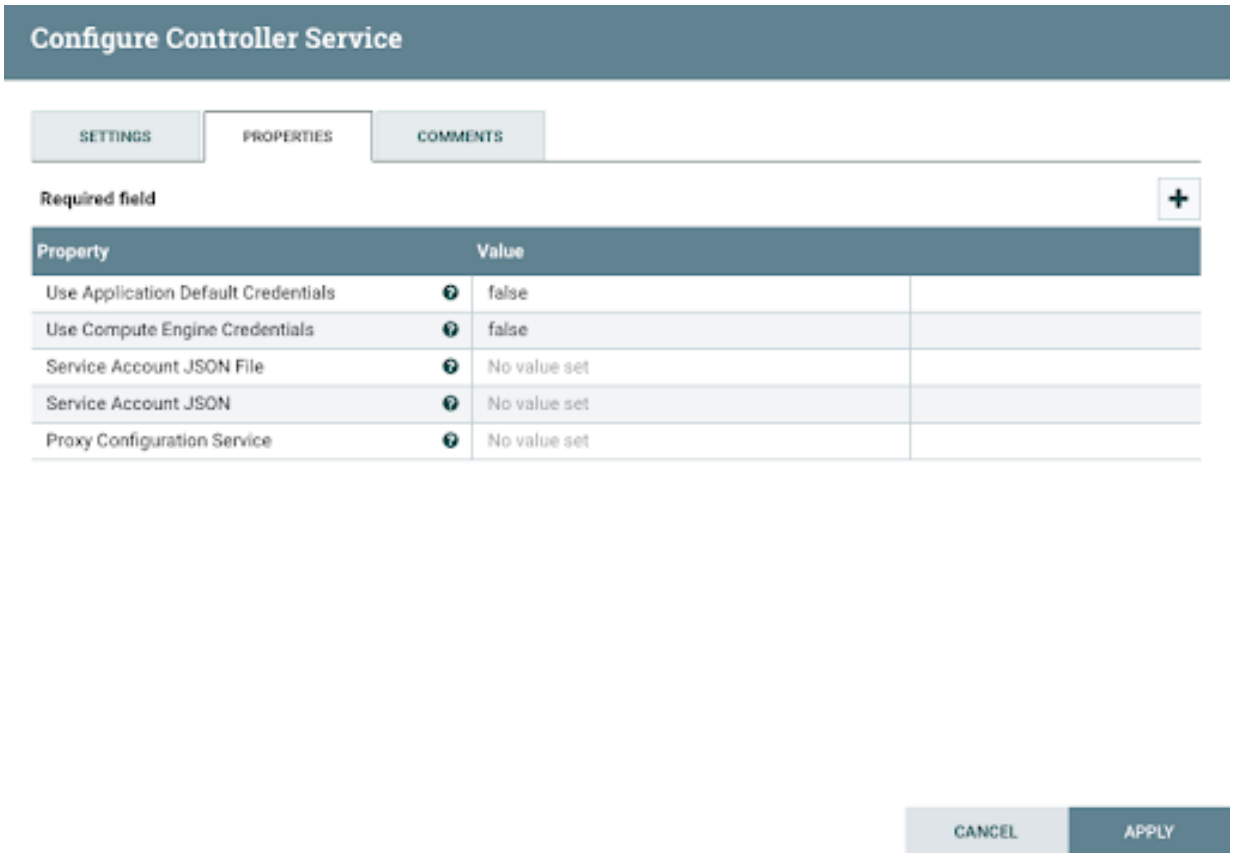

6. When you have finished configuring the options you need, save the changes by clicking Apply.
7. Enable the controller service by clicking the Enable button (flash) in the far-right column of the Controller Services tab.

What to do next

Configure the processors in your data flow.

Related Information

Adding Controller Services for data flows

Apache NiFi Documentation

Configure the processor for your data source

You can set up a data flow to move data to Google Cloud Storage (GCS) from many different locations. This example assumes that you are streaming the data that you are tailing from a local file on the NiFi host. Learn how you can configure the `TailFile` data source processor for your GCS ingest data flow.

Procedure

1. Launch the Configure Processor window, by right clicking the `TailFile` processor and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.

   You can provide the path to the nifi-app.log file that contains the log of the NiFi service running on the host.
3. When you have finished configuring the options you need, save the changes by clicking Apply.

   Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

What to do next

Configure the processor for merging your records.

Related Information

Configuring a processor

Apache NiFi Documentation

Getting Started with Apache NiFi

Configure the processor for merging records

You can use merge together multiple record-oriented flow files into a large flow file that contains all records of your data input. Learn how you can configure the `MergeContent` processor for your Google Cloud Storage (GCS) ingest data flow.

Procedure

1. Launch the Configure Processor window, by right clicking the `MergeContent` processor and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.
3. When you have finished configuring the options you need, save the changes by clicking Apply.

   Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

   For a complete list of `MergeContent` properties, see the *processor documentation*.

What to do next

Configure the processor for your data target.

Related Information

Configuring a processor

Apache NiFi Documentation

Getting Started with Apache NiFi

Configure the processor for your data target

This example use case shows you two options for ingesting data to Google Cloud Storage (GCS). Learn how you can configure these data NiFi processors for your GCS ingest flow.

Procedure

1. Launch the Configure Processor window by right clicking the processor you added for writing data to GCS (`PutHDFS`or `PutGCSObject`) and selecting Configure.

   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.

   Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.
3. When you have finished configuring the options you need, save the changes by clicking Apply.

   The following properties are used for `PutHDFS`:

Table 1: PutHDFS processor properties

Property	Description	Example value for ingest data flow
Hadoop Configuration Resources	Specify the path to the core-site.xml configuration file. Make sure that the default file system (fs, default.FS) points to the GCS bucket you are writing to. **Note:** The core-site.xml file used for the Hadoop Configuration Resources property is present on every Flow Management cluster. Cloudera Manager stores the right core-site.xml file in the same /etc directory for every cluster.	/etc/hadoop/conf.cloudera.core_settings/core-site.xml
Kerberos Principal	Specify the Kerberos principal (your username) to authenticate against CDP.	srv_nifi-gcs-ingest
Kerberos Password	Provide the password that should be used for authenticating with Kerberos.	password

Property	Description	Example value for ingest data flow
Directory	Provide the path to your target directory in Google Cloud expressed in a GCS compatible path. It can also be relative to the path specified in the core-site.xml file.	gs://your path/customer

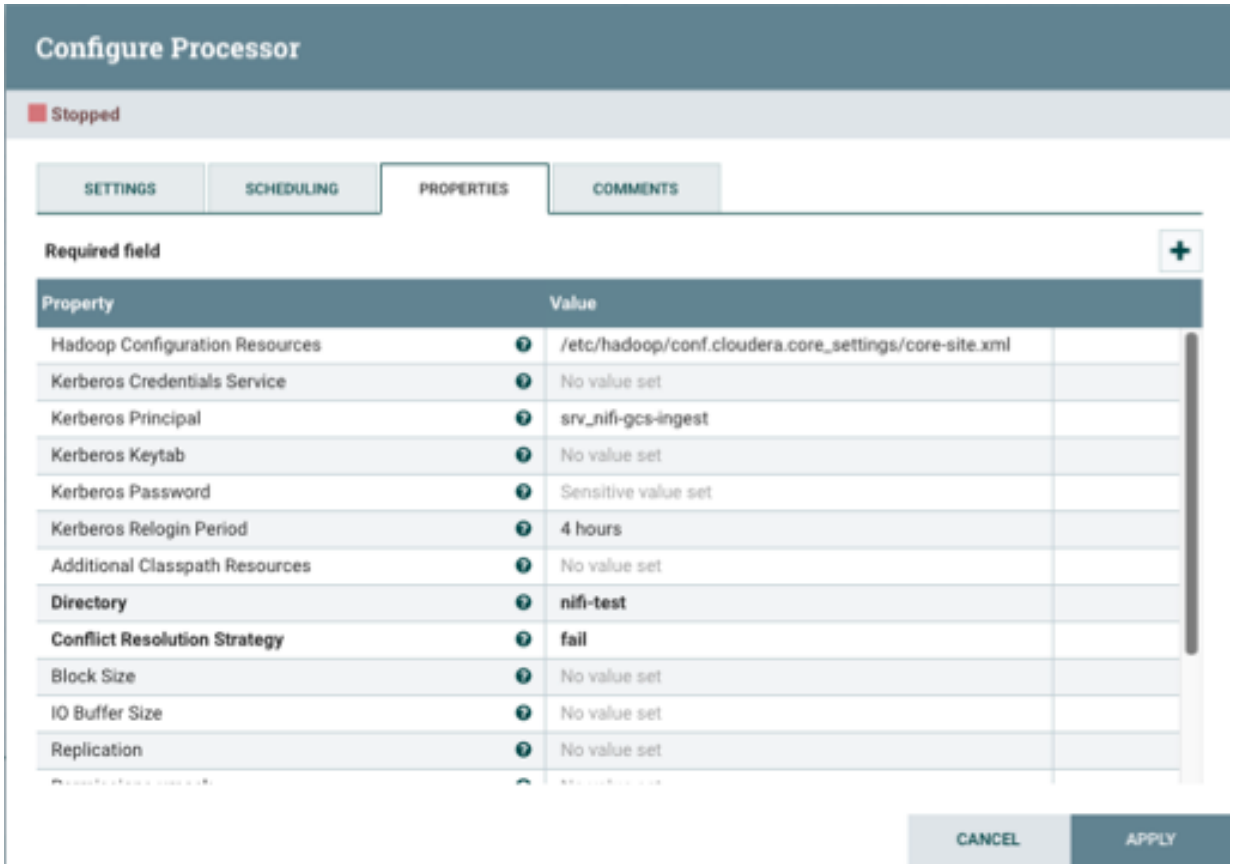

You can leave all other properties as default configurations.

For a complete list of `PutHDFS` properties, see the *processor documentation*.

If you want to use the `PutGCSObject` processor to store the data in Google Cloud Storage, you have to configure your processor to reference the controller service you configured before:

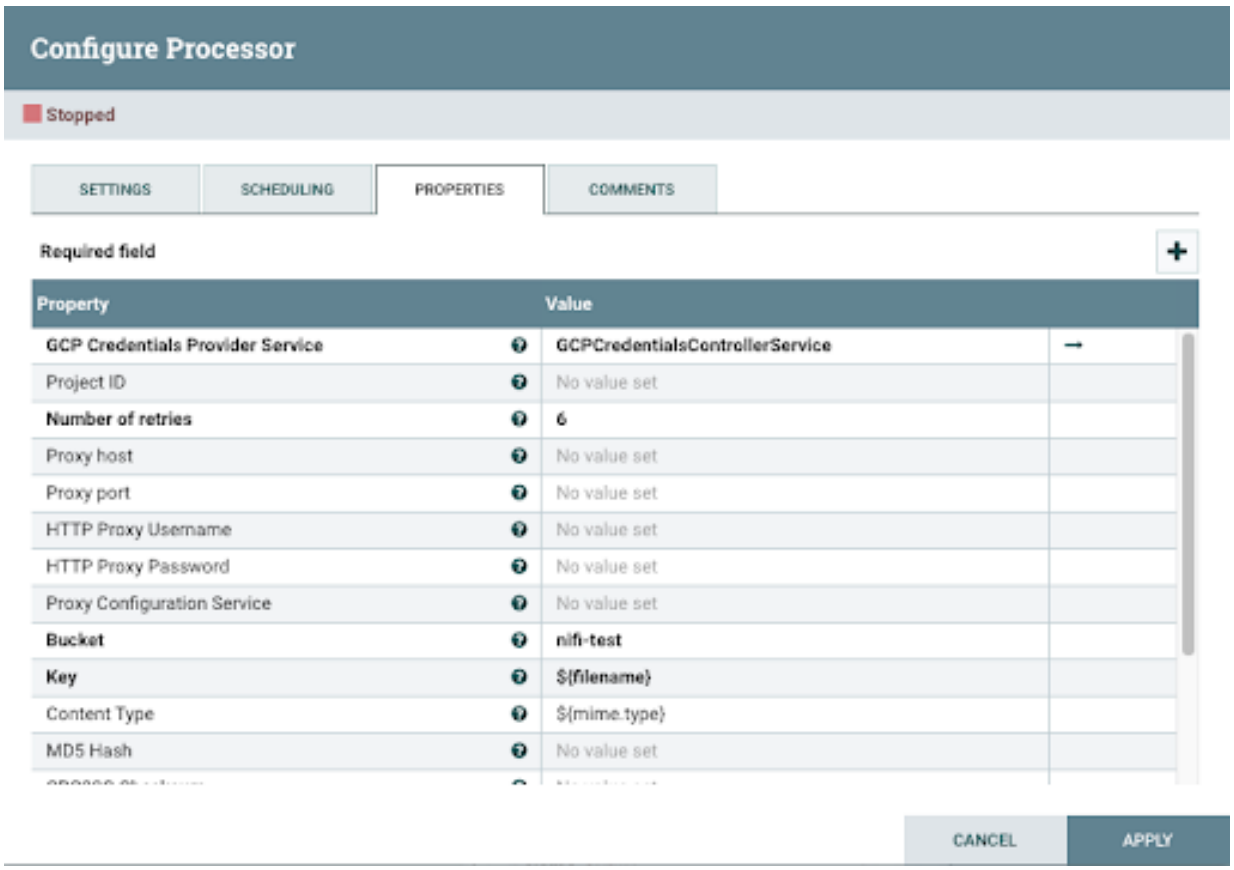

If you want to move data to a different location, review the other use cases in the *Cloudera Data Flow for Data Hub library*.

What to do next

Your data flow is ready to ingest data into Google Cloud Storage. Start the flow.

Related Information

Configuring a processor

Apache NiFi Documentation

Data ingest use cases in Cloudera Data Flow for Data Hub

Start the data flow

When your flow is ready, you can begin ingesting data into Google Cloud Storage buckets. Learn how to start your GCS ingest data flow.

Procedure

1. To initiate your data flow, select all the data flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternatively, right-click a single component and choose Start from the context menu. Data should be read from Kafka and it should be written to the defined folder of your GCS bucket.

Results

What to do next

It is useful to check that data is running through the flow you have created.

Verify data flow operation

Learn how you can verify the operation of your Google Cloud Storage (GCS) ingest data flow.

About this task

There are a number of ways you can check that data is running through the flow you have built.

Procedure

- You can verify that NiFi processors are not producing errors.
- You can look at the processors in the UI, where you can see the amount of data that has gone through them. You can also right click on the processors, or on connections to view status history.
- You can check that the data generated appears in your Google Cloud Storage bucket. To do this, return to the dedicated folder of your GCS bucket in the Google Cloud console, where you should see your files listed. You may have to refresh the page depending on your browser/settings.

Monitoring your data flow

Learn about the different monitoring options for your Google Cloud Storage (GCS) ingest data flow in CDP Public Cloud.

You can monitor your data flow for information about health, status, and details about the operation of processors and connections. NiFi records and indexes data provenance information, so you can conduct troubleshooting in real time.

Data statistics are rolled up on a summary screen (the little table icon on the top right toolbar which lists all the processors). You can use the `MonitorActivity` processor to alert you, if for example you have not received any data in your flow for a specified amount of time.

If you are worried about data being queued up, you can check how much data is currently queued. Process groups also conveniently show the totals for any queues within them. This can often indicate if there is a bottleneck in your flow somewhere, and how far the data has got through that pipeline.

Another option to check that data has fully passed through your flow is to check out data provenance to see the full history of your data.

Viewing data lineage in Apache Atlas

Learn how you can follow your data lifecycle conveying the origin of data and where it moves over time. You may view dataset level lineage graphs in the Atlas UI.

When deploying a NiFi cluster in CDP Public Cloud, a reporting task is automatically configured that sends lineage information about your flows into Apache Atlas. In the case of the example described in this simple use case, you can confirm in the Atlas UI that the data is correctly collected.

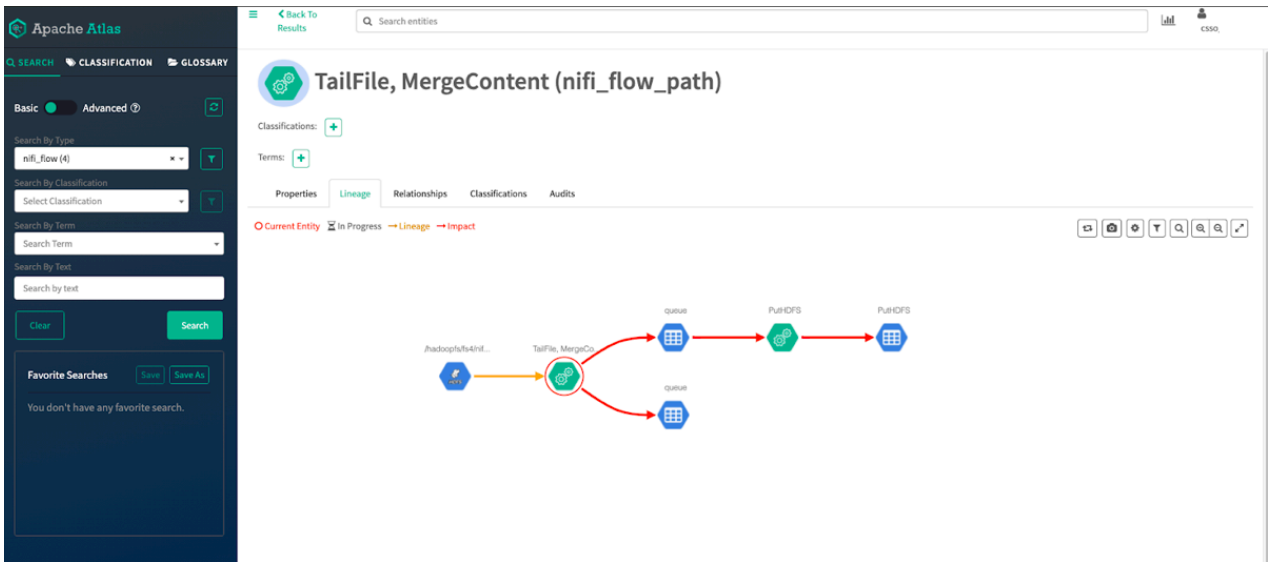

Related Information

Viewing data lineage in Apache Atlas

Next steps

Learn about the different options that you have after building a simple Google Cloud Storage ingest data flow in CDP Public Cloud.

Moving data to the cloud is one of the cornerstones of any cloud migration. Cloud environments offer numerous deployment options and services. This example data flow provides you with a model to design more complex data flows for moving and processing data as part of cloud migration efforts.

You can build a combination of on-premise and public cloud data storage. You can use this solution as a path to migrate your entire data to the cloud over time—eventually transitioning to a fully cloud-native solution or to extend your existing on-premise storage infrastructure, for example for a disaster recovery scenario. Cloud storage can provide secure, durable, and extremely low-cost options for data archiving and long-term backup for on-premise datasets.

You can also use cloud services without storing your data in the cloud. In this case you would continue to use your legacy on-premise data storage infrastructure, and work with on-demand, cloud-based services for data transformation, processing and analytics with the best performance, reliability and cost efficiency for your needs. This way you can manage demand peaks, provide higher processing power, and sophisticated tools without the need to permanently invest in computer hardware.

Getting started with Apache NiFi

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Getting started with Apache NiFi.....4

Getting started with Apache NiFi

For more information, see the Apache NiFi Getting Started Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Related Information

Getting Started with Apache NiFi

Cloudera DataFlow for Data Hub 7.3.1

Ingesting data into Apache HBase in Cloudera Public Cloud

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ingesting Data into HBase **4**
Understand the use case 4
Meet the prerequisites 4
Create the HBase target table 5
Add Ranger policies 5
Obtain HBase connection details 6
Build the data flow 7
Configure the HBase client service 8
Configure the processor for your data source 10
Configure the processor for your data target 12
Start your data flow 12
Verify your data flow 13

Next steps **13**

Ingesting Data into HBase

You can use an Apache NiFi data flow to ingest data into Apache HBase in the Cloudera Public Cloud by following these steps.

Understand the use case

You can use Apache NiFi to move data from a range of locations into an Operational Database cluster running Apache HBase in Cloudera Public Cloud.

Today's scalable web applications for use cases like hotel or flight bookings as well as mobile banking applications are relying on an equally scalable database which can serve data at a very low latency. Cloudera's Operational Database in Cloudera Data Hub is powered by Apache HBase and provides application developers with everything they need to build scalable applications on top of it.

You can use Apache NiFi data flows into Apache HBase in a Cloudera Public Cloud Operational Database cluster to make sure that the applications you build on top always have access to the latest data.

This use case walks you through the steps associated with creating an ingest-focused data flow from Apache Kafka into HBase. If you are moving data from a location other than Kafka, review the *Getting Started with Apache NiFi* for information about how to build a data flow, and about other data `get` and `consume` processor options. You can also find instructions on building a data flow for ingest into other Cloudera components.

Related Information

Getting Started with Apache NiFi

Ingesting data into Apache Kafka in Cloudera Public Cloud

Ingesting data into Apache Hive in Cloudera Public Cloud

Ingesting data into Apache Kudu in Cloudera Public Cloud

Ingesting Data into Amazong S3 Buckets

Ingesting Data into Azure Data Lake Storage

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a Cloudera Public Cloud environment.
- You have a workload username and password set to access Cloudera Data Hub clusters. The predefined resource role of this user is at least "EnvironmentUser". This resource role provides the ability to view Cloudera Data Hub clusters and set the FreeIPA password for the environment.
- Your Cloudera user is synchronized to the Cloudera Public Cloud environment.
- You have a Flow Management cluster running Apache NiFi.
- You have a Streams Messaging cluster running Apache Kafka, and have Kafka data available for consumption.
- You have an Operational Database cluster running Apache HBase and Phoenix.
- Your Cloudera user has the correct permissions set up in Ranger allowing access to the NiFi UI.

Related Information

AWS Onboarding Quickstart

Azure Onboarding Quickstart

Understanding roles and resources

Setting up your Flow Management cluster

Create the HBase target table

Before you can ingest data into Apache HBase in Cloudera Public Cloud, ensure that you have an HBase target table. These steps walk you through creating a simple table with one column family. Modify these instructions based on your ingest target table needs.

Before you begin

Your Cloudera user has the necessary Ranger permissions to create an HBase table.

Procedure

1. Ssh into one of your HBase nodes, entering your Cloudera user name and password when prompted.

```
ssh <cdp-user.name><HBase.node.name>
```

2. Use klist to verify your Kerberos ticket, to ensure that the Ranger permissions you have setup to create HBase tables will work with your Cloudera user.
3. Open the HBase shell.

```
hbase shell
```

4. Create a new table.

   For example, to create a new table called customer, with one column family called customer_data, enter:

```
create 'customer', 'customer_data'
```

5. Verify that you have created your table.

```
list

TABLE
customer
1 row(s)
```

6. Create a Ranger policy that allows the Cloudera user to ingest data into this HBase table. You will use this Cloudera user later, when specifying credentials for your NiFi data flow.

Add Ranger policies

Add Ranger policies to ensure that you have write access to your HBase tables.

Procedure

1. Go to Ranger in your Data Lake, and select the cm_hbase service.
2. From the Ranger cm_hbase service in your Data Lake, create a new policy called customer-ingest that allows the `nifi-hbase-ingest` user to access the HBase table into which you want to ingest data.

   The policy should ensure that the specified user has access to:

   - The HBase table for data ingest
   - The HBase Column family for data ingest into the specified table
   - All columns within the specified column family
   - Allows read and write permissions

Obtain HBase connection details

To enable an Apache NiFi data flow to communicate with HBase, you must obtain the HBase connection details by downloading several client configuration files. The NiFi processors require these files to parse the configuration values and use those values to communicate with HBase.

Procedure

1. From the Services pane in the Operational Database cluster that contains the HBase instance to which you want to ingest data, click CM-UI.
2. Click HBase, located on the Cloudera Manager Clusters pane.
3. From the HBase view, click Actions | Download Client Configurations to download the client configuration files as a zip file.

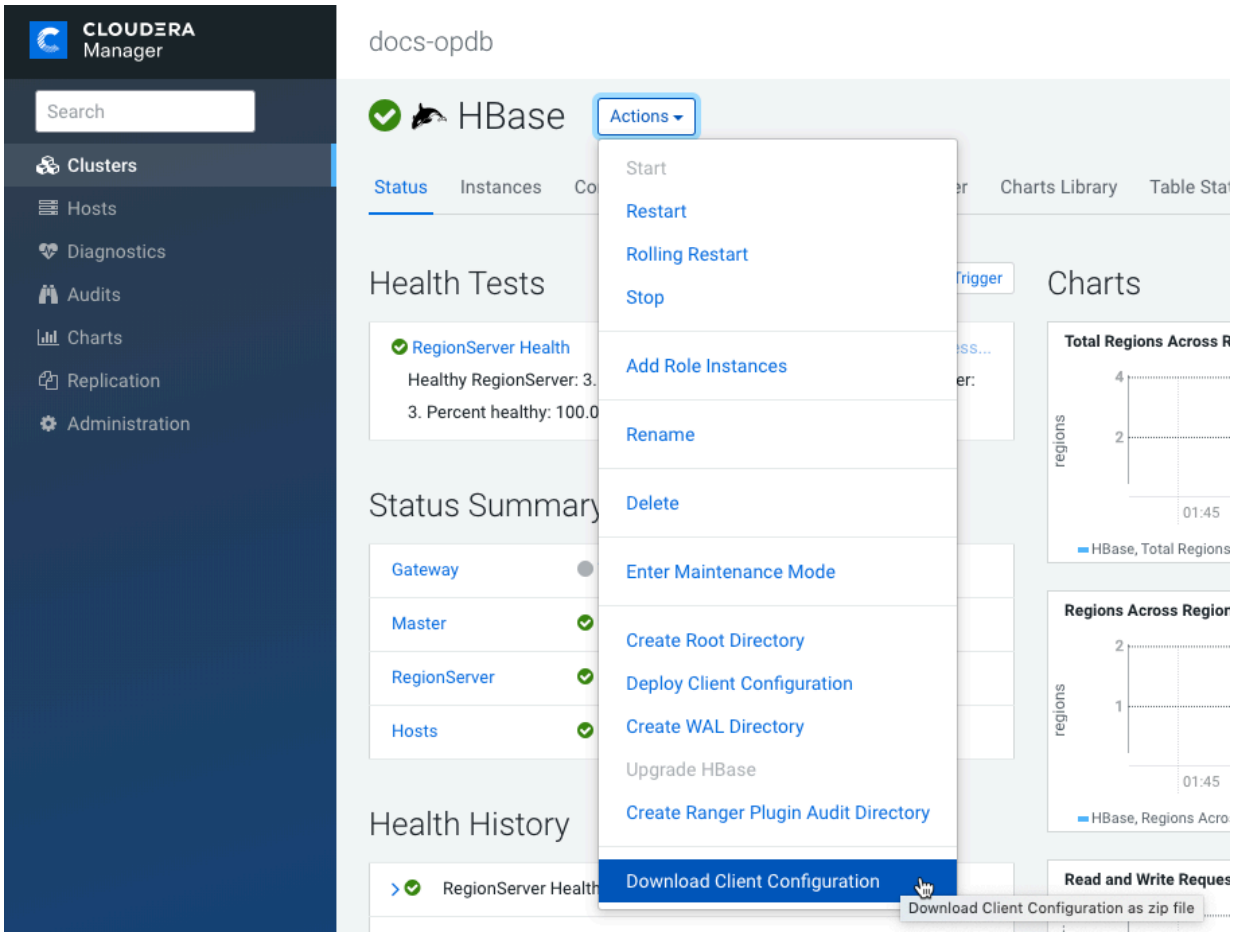

4. Expand the zip file and upload the following files to each of the NiFi nodes in your Flow Management cluster:
   - hbase-site.xml
   - core-site.xml

   For example:

   ```
   scp hbase-site.xml cdp_username@publiclp:/tmp
   ```

   You may also copy these files to a more permanent location later.

5. Adjust the file permissions to ensure that the NiFi processors can read them.

   For example:

```
chmod 755 tmp/*-site.xml
```

Build the data flow

From the Apache NiFi canvas, set up the elements of your data flow. This involves opening NiFi in Cloudera Public Cloud, adding processors to your NiFi canvas, and connecting the processors.

About this task

You should use the `PutHBaseRecord` processor to build your HBase ingest data flows.

Procedure

1. Open NiFi in Cloudera Public Cloud.
   a) To access the NiFi service in your Flow Management cluster, navigate to Management Console service > Data Hub Clusters.
   b) Click the tile representing the Flow Management cluster with which you want to work.
   c) Click the NiFi icon in the Services section of the Cluster overview page to access the NiFi UI.

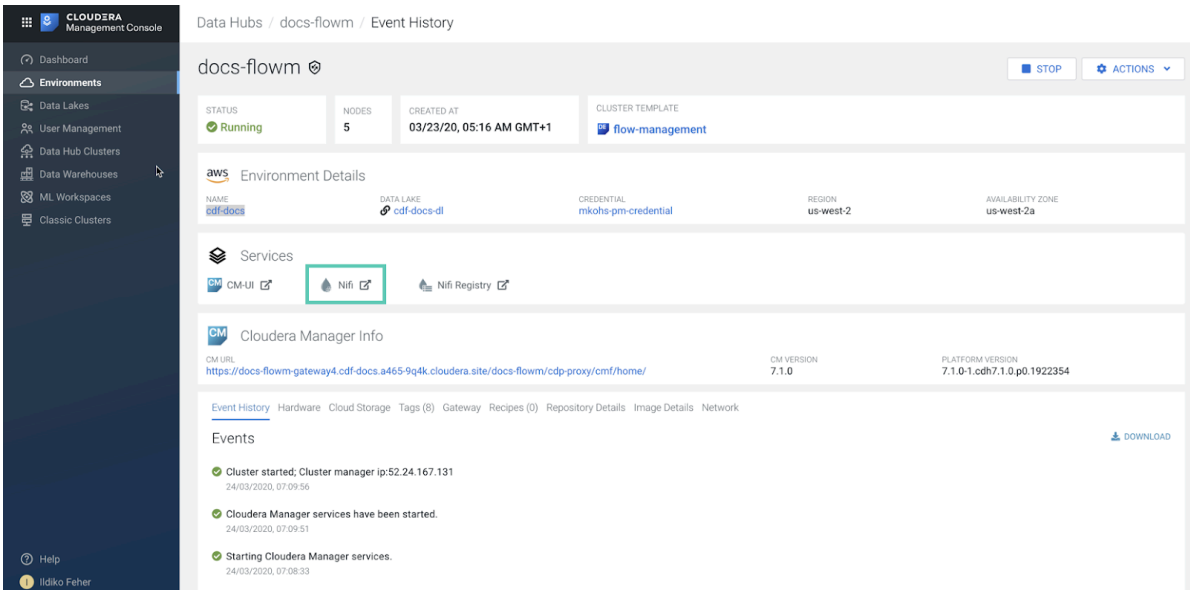

2. Add the NiFi Processors to your canvas.
   a) Select the Processor icon from the Cloudera Flow Management actions pane, and drag a processor to the Canvas.
   b) Use the Add Processor filter box to search for the processor you want to add, and then click Add.
   c) Add each of the processors you want to use for your data flow.
3. Connect the two processors to create a flow.
   a) Click the connection icon in the first processor, and drag it to the second processor.
   b) A Create Connection dialog displays. It has Details and Settings tabs. You can configure the connection's name, FlowFile expiration time period, thresholds for back pressure, load balance strategy, and prioritization.
   c) Click Add to close the dialog box and add the connection to your flow.

      Optionally, you can add success and failure funnels to your data flow, which help you see where flow files are routed when your data flow is running.

Results

Your data flow may look similar to the following:

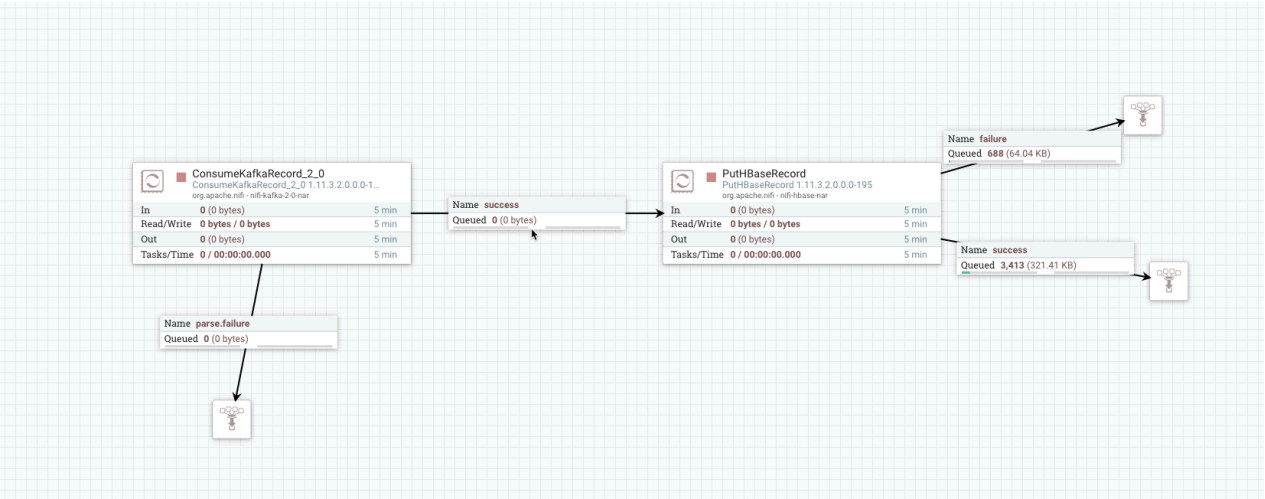

What to do next

Create the Controller service for your data flow. You will need these services later on as you configure your data flow target processor.

Related Information

Ingesting data into Apache Kafka in Cloudera Public Cloud

ConsumeKafkaRecord_2_0

PutHBaseRecord

Building a data flow

Configure the HBase client service

You can add Controller Services to provide shared services to be used by the processors in your data flow. In the case of data ingest into HBase, use a Controller Service to configure the HBase client service.

About this task

When ingesting data into HBase in Cloudera Public Cloud, use the `HBase_2_ClientService` Controller Service.

You must define the Controller Services for the processors in your data flow in the configuration of the root process group where they will be used.

Procedure

1. To add a Controller Service to your flow, right-click on the Canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.

3. Click the + button to display the Add Controller Service dialog.

4. Select the required Controller Service and click Add.

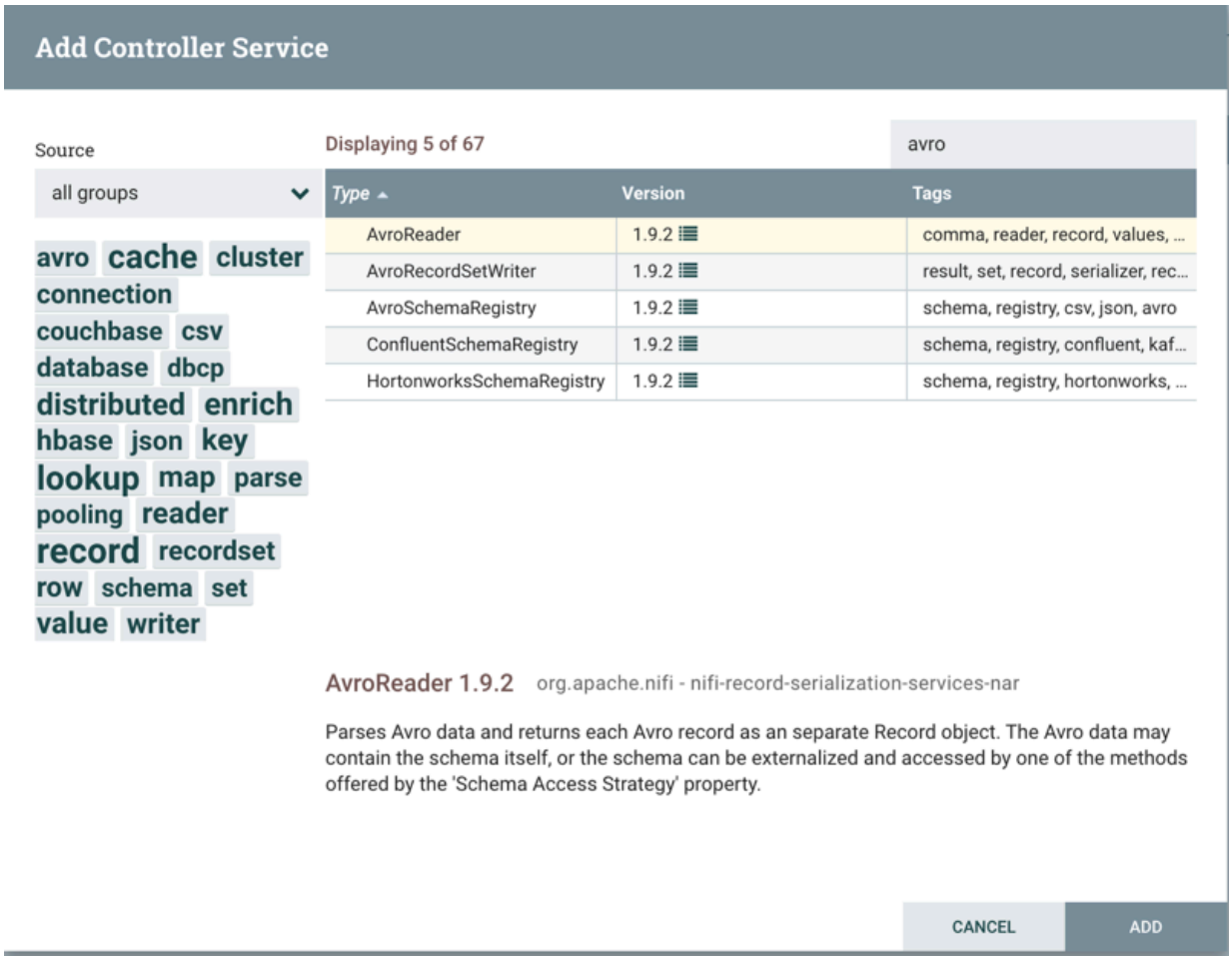

5. Perform any necessary Controller Service configuration tasks by clicking the Configure icon in the right-hand column.

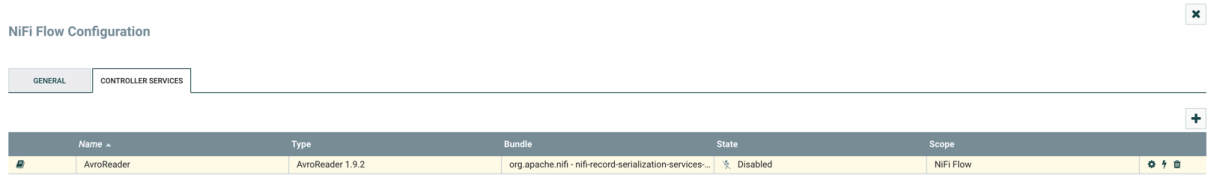

6. When you have finished configuring the options you need, save the changes by clicking the Apply button.

7. Enable the Controller Service by clicking the Enable button (flash) in the far-right column of the Controller Services tab.

Example

In this example, we are ingesting data into HBase, and will configure the `HBase_2_ClientService` Controller Service to create the HBase client service. For ingest data flows configure `HBase_2_ClientService` with the following required information. For a complete list of configuration properties, see the *Controller Service documentation*.

Property	Description	Example value for ingest data flow
Kerberos Principle	Specify the user name that should be used for authenticating with Kerberos. Use your Cloudera workload username to set this Authentication property.	srv-hbase-ingest-user
Kerberos Password	Provide the password that should be used for authenticating with Kerberos. Use your Cloudera workload password to set this Authentication property.	password
Hadoop Configuration Files	Comma-separated list of Hadoop Configuration files, such as hbase-site.xml and core-site.xml for kerberos, including full paths to the files.	These are the files you downloaded in *Obtain the HBase connection details*.

Note: You can also use the KerberosPasswordUserService controller service instead of entering the Kerberos principle and password.

Related Information

HBase_2_ClientService

Configure the processor for your data source

You can set up a data flow to move data from many locations into Apache HBase. This example assumes that you are configuring `ConsumeKafkaRecord_2_0`. If you are moving data from a location other than Kafka, review *Getting Started with Apache NiFi* for information about how to build a data flow, and about other data consumption processor options.

Before you begin

- Ensure that you have the Ranger policies required to access the Kafka consumer group.
- This use case demonstrates a data flow using data in Kafka, and moving it to HBase. To review how to ingest data to Kafka, see *Ingesting Data to Apache Kafka*.

Procedure

1. Launch the Configure Processor window, by right-clicking the processor and selecting Configure.
2. Click the Properties tab.

3. Configure `ConsumeKafkaRecord_2_0` with the required values.

   The following table includes a description and example values for the properties required to configure an ingest data flow. For a complete list of `ConsumeKafkaRecord_2_0`, see the *processor documentation*.

Property	Description	Value for ingest to HBase data flow
Kafka Brokers	Provide a comma-separated list of known Kafka Brokers in the format <host>:<port>.	`Docs-messaging-broker1.cdf-docs.a465-9q4k.cloudera.site:9093, docs-messaging-broker2.cdf-docs.a465-9q4k.cloudera.site:9093, docs-messaging-broker3.cdf-docs.a465-9q4k.cloudera.site:9093`
Topic Name(s)	Enter the name of the Kafka topic from which you want to get data.	*CUSTOMER*
Topic Name Format	Specify whether the topics are a comma separated list of topic names, or a single regular expression.	*NAMES*
Record Reader	Specifies the record reader to use for incoming Flow Files. This can be a range of data formats.	*AVROREADER*
Record Writer	Specifies the format you want to use in order to serialize data before for sending it to the data target. Note: Ensure that the source record writer and the target record reader are using the same Schema.	*CSVRECORDSETWRITER*
Honor Transactions	Specifies whether or not NiFi should honor transactional guarantees when communicating with Kafka.	*FALSE*
Security Protocol	Specifies the protocol used to communicate with brokers. This value corresponds to Kafka's 'security.protocol' property.	*SASL_SSL*
SASL Mechanism	The SASL mechanism to use for authentication. Corresponds to Kafka's 'sasl.mechanism' property.	*PLAIN*
Username	Specify the username when the SASL Mechanism is PLAIN or SCRAM-SHA-256.	*SRV_NIFI-HBASE-INGEST*
Password	Specify the password for the given username when the SASL Mechanism is PLAIN or SCRAM-SHA-256.	*PASSWORD1!*
SSL Context Service	Specifies the SSL Context Service to use for communicating with Kafka.	*DEFAULT*
Group ID	A Group ID is used to identify consumers that are within the same consumer group. Corresponds to Kafka's 'group.id' property.	*NIFI-HBASE-INGEST*

What to do next

Configure the processor for your data target.

Related Information

ConsumeKafka_2_0

Building a data flow

Ingesting data into Apache Kafka in Cloudera Public Cloud

Configure the processor for your data target

You can set up a data flow to move data into many locations. This example assumes that you are moving data into Apache HBase using `PutHBaseRecord`. If you are moving data into another location, review *Getting Started with Apache NiFi* for information about how to build a data flow, and about other data ingest processor options.

About this task

If you are moving data into HBase, `PutHBaseRecord` is the preferred HBase processor to use.

Procedure

1. Launch the Configure Processor window, by right-clicking the processor and selecting Configure.
2. Click the Properties tab.
3. Configure `PutHBaseRecord` with the required values.

   The following table includes a description and example values for the properties required to configure an ingest data flow into HBase. For a complete list of `PutHBaseRecord`, see the *processor documentation*.

Property	Description	Value for ingest to HBase data flow
Table name	The name of the HBase table into which you want to put data. The format namespace:table should be used to get the correct data lineage in Atlas.	default:customer
Column family	The column family to use when inserting data into HBase.	customer_data
Row Identifier Field Name	Specifies the name of a record field whose value should be used as the row id for the given record.	customer_id
Record Reader	Specifies the Controller Service to use for parsing incoming data and determining the data's schema	RecordCustomerCSVReader
HBase Client Service	Specifies the Controller Service to use for accessing HBase.	Select the Controller Service you configured in *Configure the HBase client service*.

What to do next

Start your data flow.

Related Information

PutHBaseRecord

Start your data flow

Start your data flow to verify that you have created a working dataflow and to begin your data ingest process.

Procedure

1. To initiate your data flow, select all the flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternately, right-click a single component and choose Start from the context menu.

Verify your data flow

Once you have configured and started your data flow, you can verify that you are successfully ingesting data into HBase.

Before you begin

You have confirmed that NiFi processors are not producing errors.

Procedure

1. Open the HBase shell.

```
hbase shell
```

2. List your available HBase tables. You should see the customer table you created at the beginning of this workflow.

   For example:

```
list

TABLE
customer
1 row(s)
```

3. Scan the customer table you created.

   For example:

```
scan 'customer'
```

Results

A successful data flow results in data moving into the customer table.

Next steps

Provides information on what to do once you have moved data into HBase in Cloudera Public Cloud.

You have built a simple data flow for an easy way to move data to HBase. This example data flow enables you to easily design more complex data flows for moving and processing data in HBase. Here are some ideas for next steps:

- Review *NiFi documentation* to learning more about building and managing data flows.
- Review *NiFi Registry documentation* for more information about versioning data flows.
- Review the *Cloudera Runtime HBase documentation* for information about working with HBase.

Related Information

NiFi documentation

NiFi Registry documentation

Cloudera Runtime HBase documentation

Cloudera DataFlow for Data Hub 7.3.1

Ingesting data into Apache Hive in Cloudera Public Cloud

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ingesting data into Hive....4
Understand the use case....4
Meet the prerequisites....4
Configure the service account....5
Create IDBroker mapping....6
Create the Hive target table....8
Add Ranger policies....8
Obtain Hive connection details....9
Build the data flow....10
Configure the controller services....12
Configure the processor for your data source....14
Configure the processor for your data target....16
Start your data flow....17
Verify your data flow....17

Next steps....17

Ingesting data into Hive

Understand the use case

You can use Apache NiFi to move data from a range of locations into a Data Engineering cluster running Apache Hive in Cloudera Public Cloud.

This use case walks you through the process of creating a data flow focused on ingesting data from Apache Kafka into Apache Hive. The PutHive3Streaming processor is used to ingest data into a Hive table. This processor interacts with the Hive Metastore to determine the storage location of the Hive table and writes data files directly to that location, eliminating the need for a running HiveServer2. After the delta files are written, the NiFi processor updates the table metadata in the Hive Metastore, ensuring that future queries include the newly ingested data.

Note: Although this process does not require a running HiveServer2 or a cluster for executing Hive queries, Cloudera strongly recommends having a Data Engineering cluster available in the same environment so that it can run Hive's compaction background processes which reconcile the delta files created by NiFi into base files. Without compaction, an excessive number of delta files can slow down Hive queries, as the engine has to reconcile all the delta files during query execution.

If you are moving data from a source other than Kafka, review the *Getting Started with Apache NiFi* for information on building data flows and exploring other data ingestion options. You can also find instructions on building a flow for ingesting data into other Cloudera components.

Related Information

Getting Started with Apache NiFi

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a Cloudera Public Cloud environment.
- You have a workload username and password set to access Cloudera Data Hub clusters. The predefined resource role of this user is at least "EnvironmentUser". This resource role provides the ability to view Cloudera Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the Cloudera Public Cloud environment.
- You have a Flow Management cluster running Apache NiFi.
- You have a running Data Engineering cluster in the same Cloudera environment.

  Note: Data can be ingested into Hive without an active Data Engineering cluster. However, for production workloads, it is strongly recommended to provision a Data Engineering cluster, which ensures that Hive’s automated compaction jobs can run concurrently with NiFi data ingestion, optimizing query performance. For more information, see *Understand the use case*.
- You have created a new database and transactional table for Hive data ingest.
- You have a policy in Ranger that allows a user to write data to your Hive table.

Related Information

Understand the use case

AWS Onboarding Quickstart

Azure Onboarding Quickstart

Understanding roles and resource roles

Cloudera workload user

Setting up your Flow Management cluster

Configure the service account

Configure the Service Account you will use to ingest data into Hive.

Procedure

1. Navigate to the Cloudera Management Console UI.
2. From the User Management tab, create a new machine user.

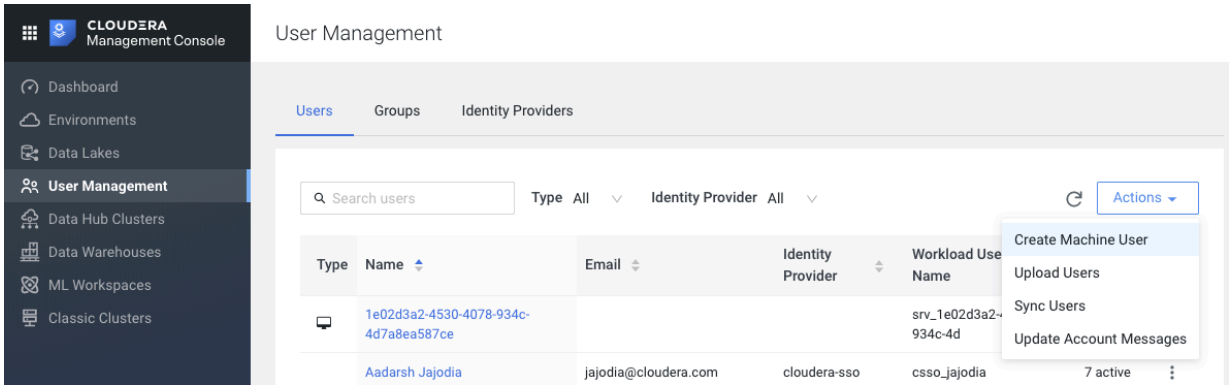

3. Once the user is created, set a workload password and synchronize the changes to the required environments.

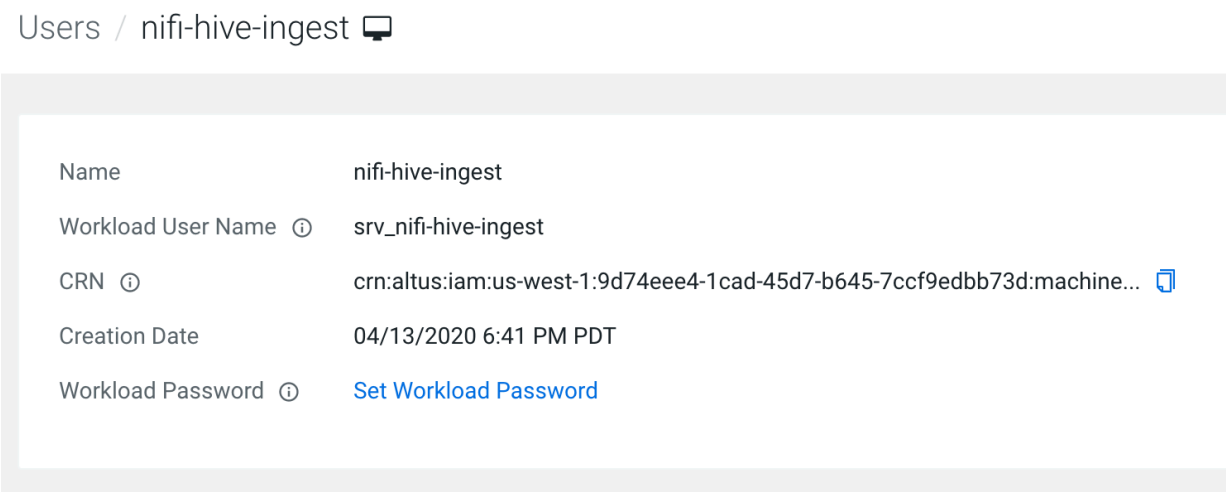

4. From the Access tab, add the machine user as an Environment User to the environment.

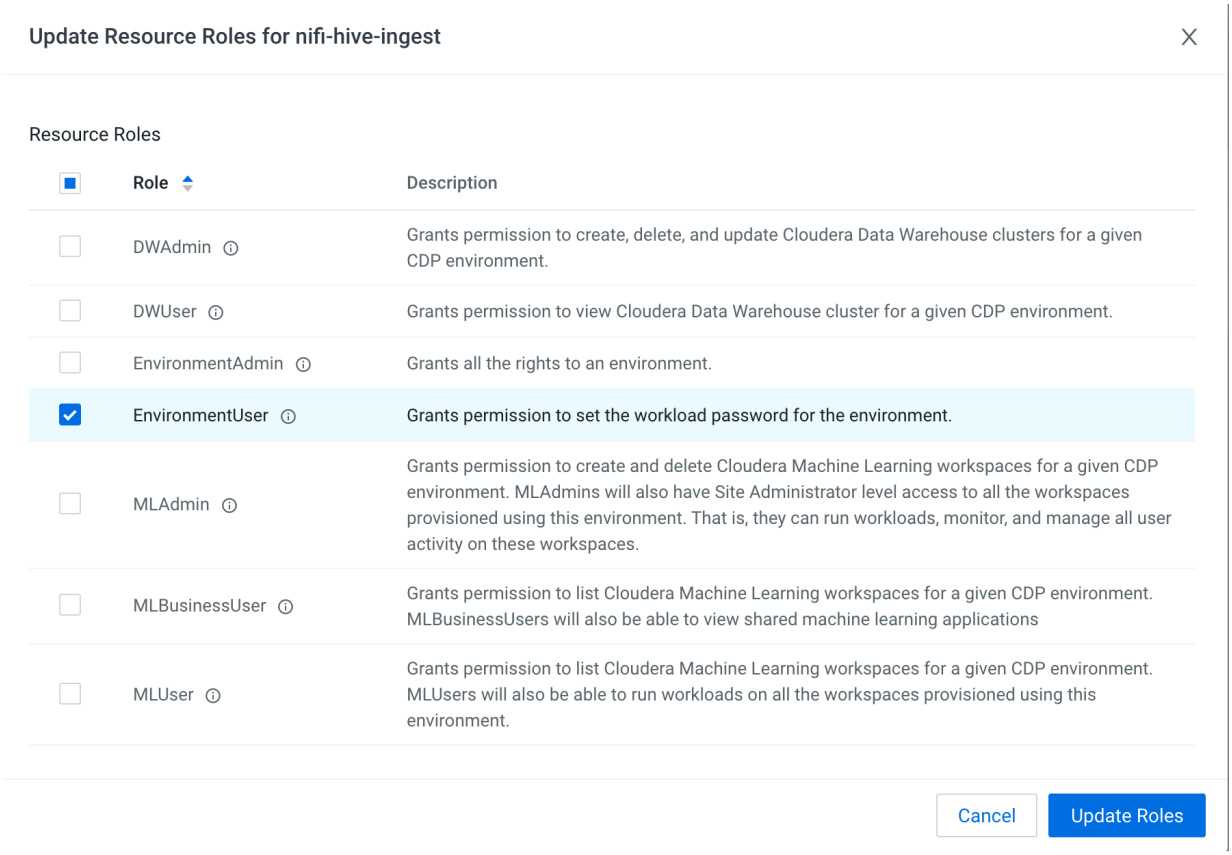

5. Go back to the environments view and click Actions - Synchronize Users to Free IPA.

Results

Once the synchronization is complete, the required user is present in the environment and has the right permissions to access the object store.

Create IDBroker mapping

To enable your Cloudera user to utilize the central authentication features Cloudera provides and to exchange credentials for AWS or Azure access tokens, you have to map this Cloudera user to the correct IAM role or Azure Managed Service Identity (MSI). The option to add/modify these mappings is available from the Cloudera Management Console in your Cloudera environment.

Procedure

1. Go to the environment in which your Flow Management and Data Engineering clusters are running.
2. To access IDBroker Mappings, click Actions | Manage Access and select the IDBroker Mapping tab in the next screen, where you can provide mappings for users or groups.
3. Click Edit.

4. Add a new mapping for your service user, mapping the user to an existing IAM role or Azure Managed Identity Resource ID that has access to the underlying storage which is used by the target Hive table.

   For example:

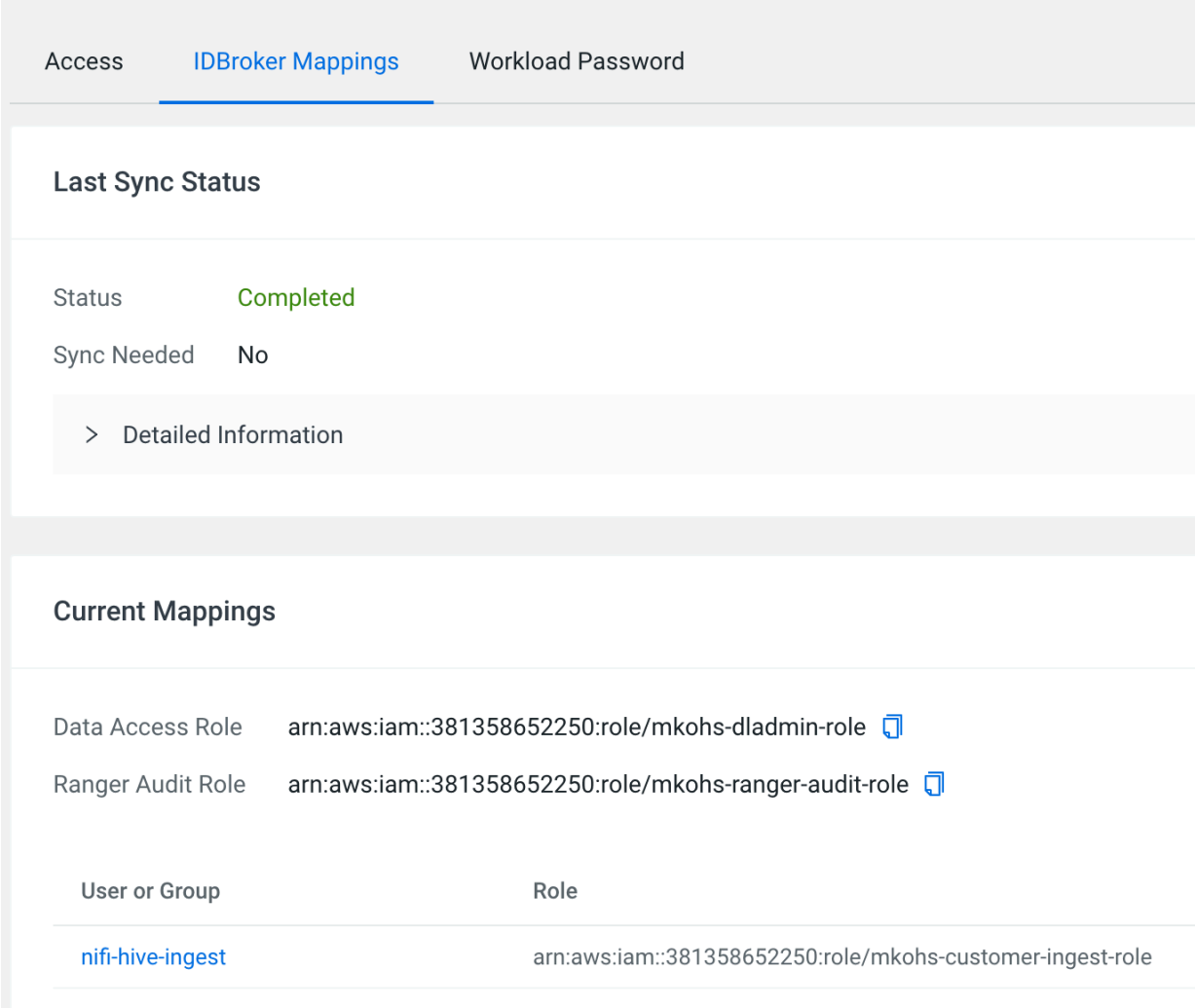

5. Add your Cloudera user and the corresponding AWS or Azure role that provides write access to your folder in your S3 bucket or ADLS folder to the Current Mappings section.

   Note:

   You can get the AWS IAM role Amazon Resource Name (ARN) from the Roles Summary page in AWS and can copy into the IDBroker role field.

   You can get the Azure Managed Identity Resource ID by navigating to Managed Identities->Your Managed Identity->Properties->Resource ID.

6. Click Save and Sync.
7. Ensure that your IDBroker mapping change is synchronized to the environment successfully.

What to do next

Create a Hive table and add Ranger policies that allow your machine user write access to your Hive table.

Related Information

IDBroker

Onboarding Cloudera users and groups for AWS cloud storage

Onboarding Cloudera users and groups for Azure cloud storage

Create the Hive target table

Before you can ingest data into Apache Hive in Cloudera Public Cloud, ensure that you have a Hive target table. These steps walk you through creating a simple table. Modify these instructions based on your data ingest target table needs.

About this task

You require a transactional Hive table. This is the type of table created by default.

Procedure

1. In the Cloudera Management Console, go to your Data Engineering cluster.
2. From Services, click Data Analytics Studio, and select Compose from the left-navigation pane.
3. Create a new target database.

```
create DATABASE retail
```

4. Create a new target table.

```
create table retail.customer(customer_id int, customer_name string)
```

What to do next

Add policies to Ranger so that the Cloudera user you are using to create this data flow can ingest data into the Hive target table.

Add Ranger policies

Add Ranger policies to ensure that you have write access to your Hive tables.

Procedure

1. From the Ranger cm_hive service in your Data Lake, create a new policy called Database access, and allow select permissions on the database in which your Hive target table is stored.

2. Create another policy to allow select, update, and Write permissions to ingest into the Hive target table.
   For example:

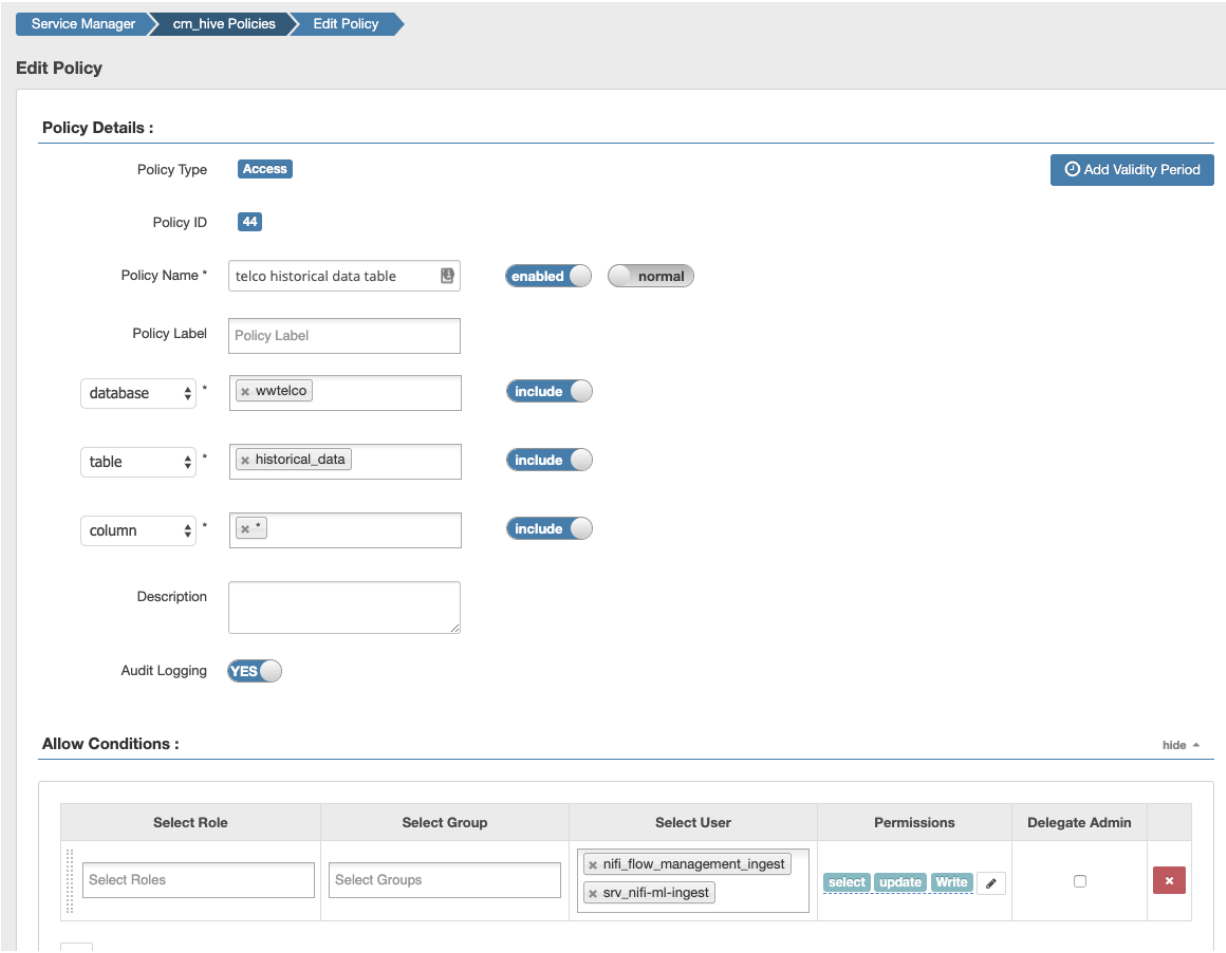

Obtain Hive connection details

To enable an Apache NiFi data flow to communicate with Hive, you must obtain the Hive connection details by downloading several client configuration files. The NiFi processors require these files to parse the configuration values and use those values to communicate with Hive.

Procedure

1. From the Services pane in the Data Engineering cluster that contains the Hive instance to which you want to ingest data, click CM-UI.
2. Click Hive Metastore, located on the Cloudera Manager Clusters pane.

3. From the Hive Metastore view, click Actions Download Client Configuration to download the client configuration files as a zip file.

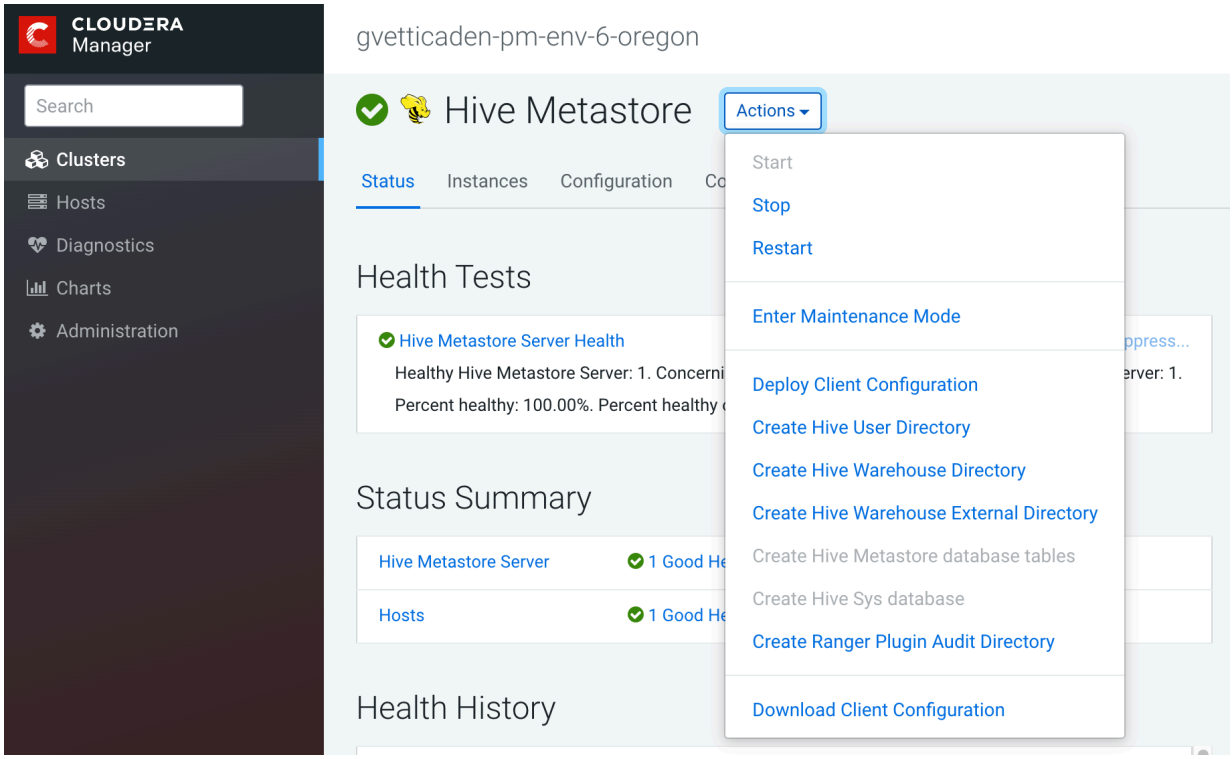

4. Expand the zip file and upload the following files to each of the NiFi nodes in your Flow Management cluster:
   - hdfs-site.xml
   - hive-site.xml
   - core-site.xml

   For example:

   ```
   scp hdfs-site.xml cdp_username@publiclp:/tmp
   ```

   You may also copy these files to a more permanent location later.
5. Adjust the file permissions to ensure that the NiFi processors can read them.

   For example:

   ```
   chmod 755 /tmp/*-site.xml
   ```

Build the data flow

From the Apache NiFi canvas, set up the elements of your data flow. This involves opening NiFi in Cloudera Public Cloud, adding processors to your NiFi canvas, and connecting the processors.

About this task

When you are building a data flow to ingest data into Apache Hive following are the preferred Hive processors

- `Select_Hive_QL_3` – To get data out of Hive tables.
- `PutHive3Streaming` – To write data to managed and external Hive tables.

Procedure

1. Open NiFi in Cloudera Public Cloud.
   a) To aces the NiFi service in your Flow Management cluster, navigate to the Cloudera Management Console and click Data Hub Clusters.
   b) Click the tile representing the Flow Management cluster with which you want to work.
   c) Click the NiFi icon in the Services section of the Cluster overview page to access the NiFi UI.

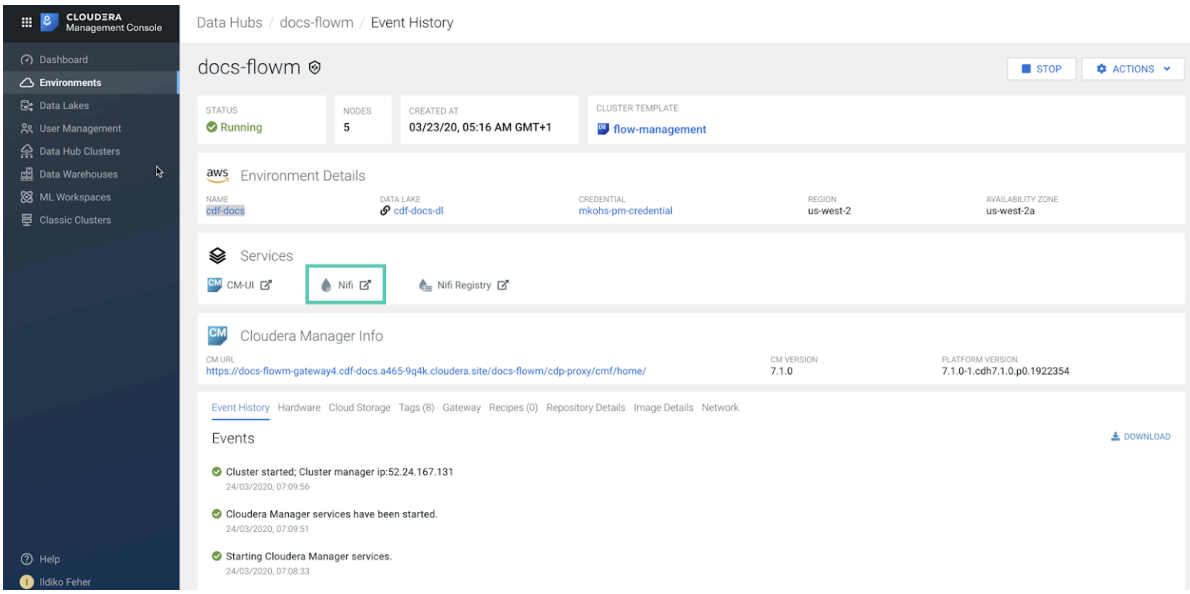

2. Add the NiFi Processors to your canvas.
   a) Select the Processor icon from the Cloudera Flow Management actions pane, and drag a processor to the Canvas.
   b) Use the Add Processor filter box to search for the processor you want to add, and then click Add.
   c) Add each of the processors you want to use for your data flow.
3. Connect the two processors to create a flow.
   a) Click the connection icon in the first processor, and drag it to the second processor.
   b) A Create Connection dialog displays. It has Details and Settings tabs. You can configure the connection's name, FlowFile expiration time period, thresholds for back pressure, load balance strategy, and prioritization.
   c) Click Add to close the dialog box and add the connection to your flow.

   Optionally, you can add success and failure funnels to your data flow, which help you see where flow files are routed when your data flow is running.

Results

Your data flow may look similar to the following:

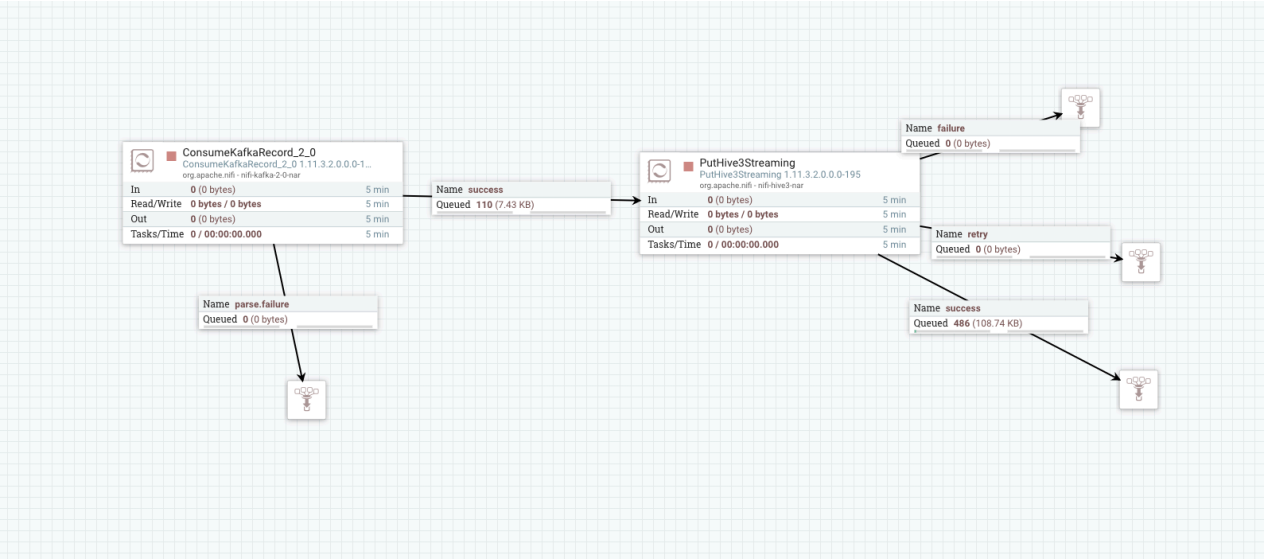

What to do next

Create the Controller service for your data flow. You will need these services later on as you configure your data flow target processor.

Related Information

Ingesting data into Apache Kafka in Cloudera Public Cloud
Consume KafkaRecord_2_0
Building a data flow

Configure the controller services

You can add Controller Services to provide shared services to be used by the processors in your data flow. You will use these Controller Services later when you configure your processors.

About this task

You must define the Controller Services for the processors in your data flow in the configuration of the root process group where they will be used.

Procedure

1. To add a Controller Service to your flow, right-click on the Canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.
3. Click the + button to display the Add Controller Service dialog.

4. Select the required Controller Service and click Add.

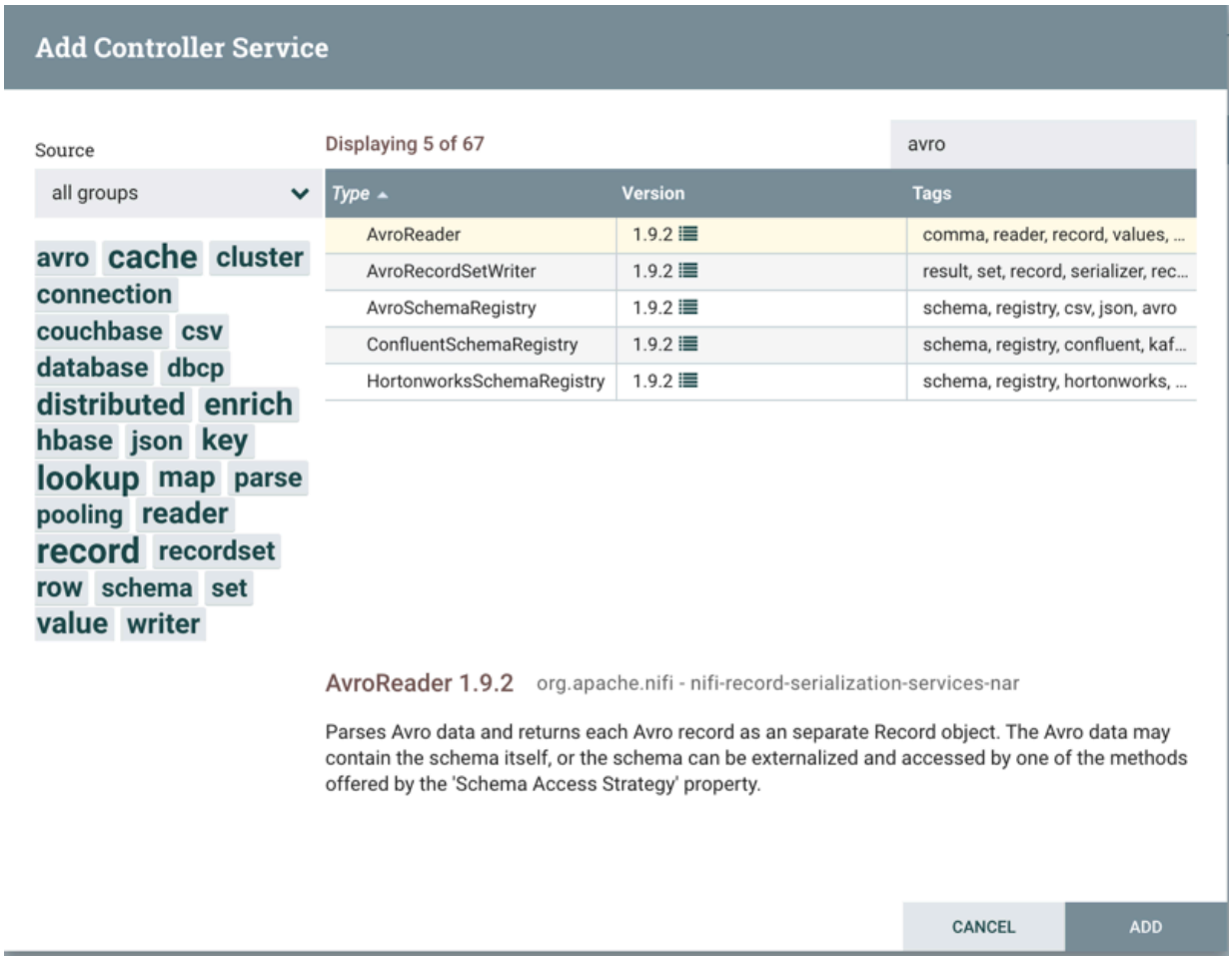

5. Perform any necessary Controller Service configuration tasks by clicking the Configure icon in the right-hand column.

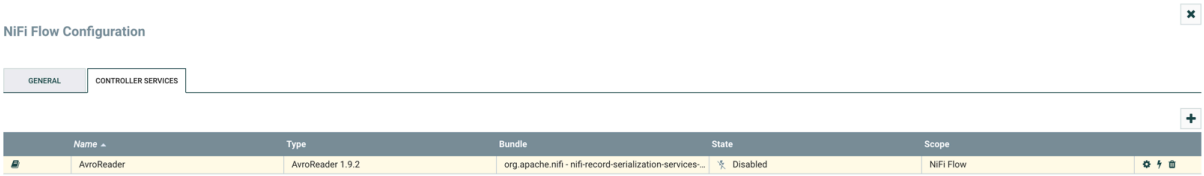

6. When you have finished configuring the options you need, save the changes by clicking the Apply button.
7. Enable the Controller Service by clicking the Enable button (flash) in the far-right column of the Controller Services tab.

Example

In this example, configure the following Controller Services:

- AvroReader
- AvroWriter
- HortonworksSchemaRegistry

AvroReader Controller Service

Table 1: AvroReader Controller Service properties

Property	Description	Example value for ingest data flow
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	HWX Content-Encoded Schema Reference
Schema Registry	Specify the Controller Service to use for the Schema Registry.	CDPSchemaRegistry
Schema Name	Specify the name of the schema to look up in the Schema Registry property.	customer

WriteAvro Controller Servie

Table 2: WriteAvro Controller Service properties

Property	Description	Example value for ingest data flow
Schema Write Strategy	Specifies how the schema for a Record should be added to the data.	HWX Content Encoded Schema Reference
Schema Access Strategy	Specifies how to obtain the schema that is to be used for interpreting the data.	Use Schema Name Property
Schema Registry	Specifies the Controller Service to use for the Schema Registry.	HortonworksSchemaRegistry
Schema name	Specifies the name of the schema to lookup in the Schema Registry property.	*CUSTOMER*

HortonworksSchema Registry

Table 3: HortonworksSchemaRegistry Controller Service properties

Property	Description	Example value for ingest data flow
Schema Registry URL	Provide the URL of the schema registry that this Controller Service should connect to, including version. In the format: https://host :7790/api/v1	https://docs-messaging-master0.cdf-docs.a465 -9q4k.cloudera.site:7790/api/v1
SSL Context Service	Specify the SSL Context Service to use for communicating with Schema Registry. Use the pre-configured SSLContextProvider.	Default NiFi SSL Context Service
Kerberos Principal	Specify the user name that should be used for authenticating with Kerberos. Use your Cloudera workload username to set this Authentication property.	srv_nifi-hive-ingest
Kerberos Password	Provide the password that should be used for authenticating with Kerberos. Use your Cloudera workload password to set this Authentication property.	password

Configure the processor for your data source

You can set up a data flow to move data from many locations into Apache Hive. This example assumes that you are configuring `ConsumeKafkaRecord_2_0`. If you are moving data from a location other than Kafka, review

Getting Started with Apache NiFi for information about how to build a data flow, and about other data consumption processor options.

Before you begin

- Ensure that you have the Ranger policies required to access the Kafka consumer group.
- This use case demonstrates a data flow using data in Kafka, and moving it to Hive. To review how to ingest data to Kafka, see *Ingesting Data to Apache Kafka*.

Procedure

1. Launch the Configure Processor window, by right-clicking the processor and selecting Configure.
2. Click the Properties tab.
3. Configure `ConsumeKafkaRecord_2_0` with the required values.

   The following table includes a description and example values for the properties required to configure an ingest data flow. For a complete list of `ConsumeKafkaRecord_2_0`, see the *processor documentation*.

Property	Description	Value for ingest to Hive data flow
Kafka Brokers	Provide a comma-separated list of known Kafka Brokers in the format <host>:<port>.	`Docs-messaging-broker1.cdf-docs.a465-9q4k.cloudera.site:9093, docs-messaging-broker2.cdf-docs.a465-9q4k.cloudera.site:9093, docs-messaging-broker3.cdf-docs.a465-9q4k.cloudera.site:9093`
Topic Name(s)	Enter the name of the Kafka topic from which you want to get data.	*CUSTOMER*
Topic Name Format	Specify whether the topics are a comma separated list of topic names, or a single regular expression.	*NAMES*
Record Reader	Specifies the record reader to use for incoming Flow Files. This can be a range of data formats.	*AVROREADER*
Record Writer	Specifies the format you want to use in order to serialize data before for sending it to the data target. Note: Ensure that the source record writer and the target record reader are using the same Schema.	*CSVRECORDSETWRITER*
Honor Transactions	Specifies whether or not NiFi should honor transactional guarantees when communicating with Kafka.	*FALSE*
Security Protocol	Specifies the protocol used to communicate with brokers. This value corresponds to Kafka's 'security.protocol' property.	*SASL_SSL*
SASL Mechanism	The SASL mechanism to use for authentication. Corresponds to Kafka's 'sasl.mechanism' property.	*PLAIN*
Username	Specify the username when the SASL Mechanism is PLAIN or SCRAM-SHA-256.	*SRV_NIFI-HIVE-INGEST*

Property	Description	Value for ingest to Hive data flow
Password	Specify the password for the given username when the SASL Mechanism is PLAIN or SCRAM-SHA-256.	*PASSWORD1!*
SSL Context Service	Specifies the SSL Context Service to use for communicating with Kafka.	*DEFAULT*
Offset Reset	Allows you to manage the condition when there is no initial offset in Kafka or if the current offset does not exist any more on the server (e.g. because that data has been deleted). Corresponds to Kafka's `auto.offset.reset` property.	*LATEST*
Group ID	A Group ID is used to identify consumers that are within the same consumer group. Corresponds to Kafka's 'group.id' property.	*NIFI-HIVE-INGEST*

What to do next

Configure the processor for your data target.

Related Information

ConsumeKafka_2_0
Building a data flow
Ingesting data into Apache Kafka in Cloudera Public Cloud

Configure the processor for your data target

You can set up a data flow to move data into many locations. This example assumes that you are moving data into Apache Hive using `PutHive3Streaming`. If you are moving data into another location, review *Getting Started with Apache NiFi* for information about how to build a data flow, and about other data ingest processor options.

Procedure

1. Launch the Configure Processor window, by right-clicking the processor and selecting Configure.
2. Click the Properties tab.
3. Configure `PutHive3Streaming` with the required values.

   The following table includes a description and example values for the properties required to configure an ingest data flow into Hive. For a complete list of `PutHive3Streaming`, see the *processor documentation*.

Property	Description	Value for ingest to Hive data flow
Database name	The name of the database in which to put the data.	retail
Table name	The name of the database table in which to put the data.	customer
Record Reader		ReadAvro
Hive Metastore URI	The URI location for the Hive Metastore. Note that this is not the location of the Hive Server. The default port for the Hive metastore is 9083.	*THIFT://<DATALAKE-MASTER-HOST-NAME:9083*

Property	Description	Value for ingest to Hive data flow
Hive Configuration Resources	A file or comma separated list of files which contains the Hive configuration.	Replace with the file paths for the three files you downloaded in *Obtain the Hive connection details*: • hdfs-site.xml • hive-site.xml • core-site.xml
Kerberos Principal	Specify your Cloudera machine user name.	*SRV_NIFI-HIVE-INGEST*
Kerberos Password	Specify your Cloudera machine user password.	*PASSWORD1!*

Start your data flow

Start your data flow to verify that you have created a working dataflow and to begin your data ingest process.

Procedure

1. To initiate your data flow, select all the flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternately, right-click a single component and choose Start from the context menu.

Verify your data flow

Learn how you can verify the operation of your Hive ingest data flow.

About this task

There are a number of ways to check that data is running through the flow you have built and it actually appears in Hive.

Procedure

1. Review the Hive ingest data flow on the NiFi canvas and confirm that the NiFi processors are not producing errors.
2. You can look at the processors in the UI, where you can see the amount of data that has gone through them. You can also right click on the processors, or on connections to view status history.
3. From the Data Analytics Studio UI, in the Compose tab, confirm that your Hive table is consuming content:

```
select * from retail.customer
```

Results

You are able to see data flowing into your retail.customer table.

Next steps

Provides information on what to do once you have moved data into Hive in Cloudera Public Cloud.

You have built a simple data flow for an easy way to move data to Hive. This example data flow enables you to easily design more complex data flows for moving and processing data in Hive. Here are some ideas for next steps:

- Review *NiFi documentation* to learning more about building and managing data flows.
- Review *NiFi Registry documentation* for more information about versioning data flows.

- Review the *Cloudera Runtime data access documentation* for information about working with Hive, Impala, and Hue.

Related Information

NiFi documentation

NiFi Registry documentation

Cloudera Runtime data access documentation

Cloudera DataFlow for Data Hub 7.3.1

Ingesting data into Apache Kafka in Cloudera Public Cloud

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ingesting data into Kafka **4**
Understand the use case 4
Meet the prerequisites 4
Build the data flow 5
Create controller services for your data flow 6
Configure the processor for your data source 8
Configure the processor for your data target 9
Start the data flow 11
Verify data flow operation 11
Monitoring end to end latency for Kafka topic 14

Monitoring your data flow **18**

Next steps **19**

Appendix – Schema example **19**

Ingesting data into Kafka

You can use an Apache NiFi data flow to ingest data into Apache Kafka by following these steps.

Understand the use case

Learn how to use NiFi to move data from a range of locations into Kafka in Cloudera Public Cloud.

Apache NiFi as a part of Cloudera Data Flow offers a scalable way of managing data flows with guaranteed delivery, data buffering / pressure release, and prioritized queuing.

You can use it to ingest data into Apache Kafka by simply dragging and dropping a series of processors on the NiFi user interface. In addition to facilitating the data flow into Kafka, the processors provide an out-of-the-box integration with Streams Messaging Manager by configuring the metrics interceptors.

When the data flow is ready, you can visually monitor and control the pipeline you have built.

This use case walks you through the steps of creating a data flow that generates FlowFiles with random CSV data and writes this data to Apache Kafka inCloudera Public Cloud. This gets you started with creating a Kafka ingest data flow. If you want to use a data source other than generated FlowFiles, see the *Getting Started with Apache NiFi* for information about how to build a data flow, and about get and consume data processor options.

Related Information

Getting Started with Apache NiFi
Ingesting Data into Apache HBase
Ingesting data into Apache Hive
Ingesting Data into Apache Kudu
Ingesting Data into Amazong S3 Buckets
Ingesting Data into Azure Data Lake Storage

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a Cloudera Public Cloud environment.
- You have a Cloudera username (it can be your own Cloudera user or a Cloudera machine user) and a password set to access Data Hub clusters.

  The predefined resource role of this user is at least EnvironmentUser. This resource role provides the ability to view Cloudera Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the Cloudera Public Cloud environment.
- Your Cloudera user has the correct permissions set up in Ranger allowing access to NiFi and Kafka.
- You have a Flow Management Data Hub cluster in your Cloudera environment.
- You have a Streams Messaging Data Hub cluster in the same Cloudera environment as the Flow Management cluster.
- You have created a Kafka topic in your Streams Messaging cluster to send data to.
- You have created a sample schema in your Streams Messaging cluster you want to use for writing data to Kafka.

  See the *Appendix* for an example schema.

Related Information

Understanding roles and resource roles
Setting up your Flow Management cluster
Setting up your Streams Messaging cluster

Creating a Kafka topic
Creating a new schema in Schema Registry
Authorizing Flow Management cluster access in Cloudera Public Cloud
Appendix – Schema example

Build the data flow

Learn how to create NiFi data flows easily with a number of processors and flow objectives to choose from.

About this task

You can use the `PublishKafka2RecordCDP` processor to build your Kafka ingest data flow. Regardless of the type of flow you are building, the basic steps in building your data flow are to open NiFi, add your processors to the canvas, and connect the processors to create the flow.

Procedure

1. Open NiFi in Data Hub.
   a) To access the NiFi service in your Flow Management Data Hub cluster, navigate to theCloudera Management Console page and select Data Hub Clusters.
   b) Click the tile representing the Flow Management Data Hub cluster you want to work with.
   c) Click the NiFi icon in the Services section of the cluster overview page to access the NiFi UI.

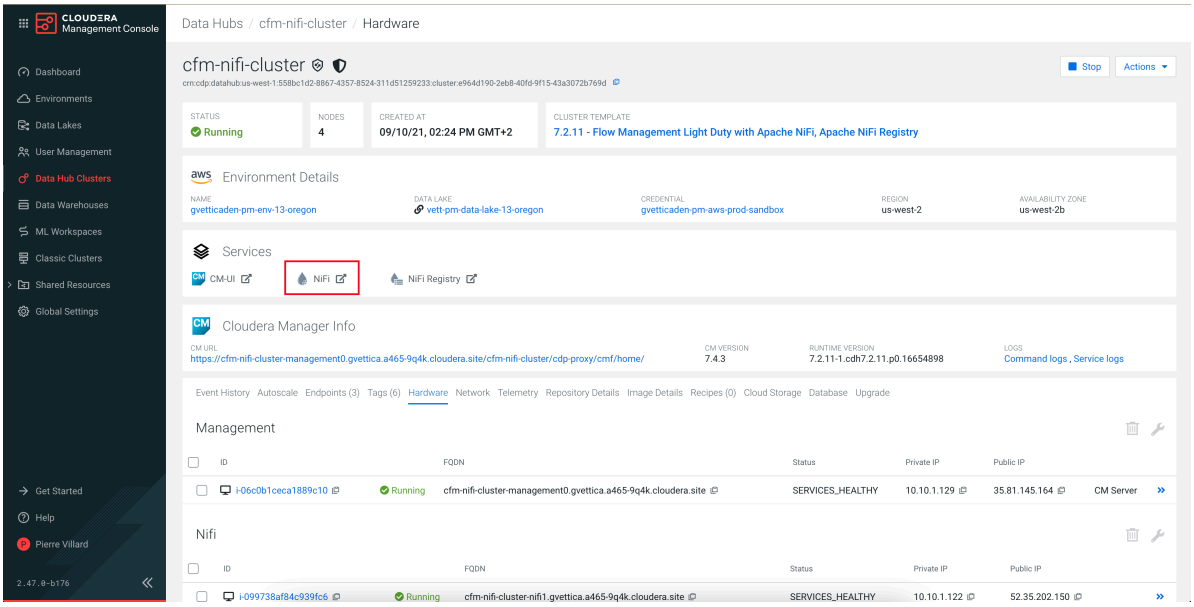

   You will be logged into NiFi automatically with your Cloudera credentials.

2. Add the `GenerateFlowFile` processor for data input.

   This processor creates FlowFiles with random data or custom content.

   a) Drag and drop the processor icon into the canvas. This displays a dialog that allows you to choose the processor you want to add.
   b) Select the `GenerateFlowFile` processor from the list.
   c) Click Add or double-click the required processor type to add it to the canvas.

   You will configure the `GenerateFlowFile` processor to define how to create the sample data in *Configure the processor for your data source*.

3. Add the `PublishKafka2RecordCDP` processor for data output.

   Note: There are various processors in NiFi for ingesting data into Kafka. `PublishKafka2RecordCDP` uses a configured record reader to read the incoming flow files as records, and then it uses a configured record writer to serialize each record for publishing to Kafka. As Streams Messaging clusters in Data Hub include Kafka version 2.x, use `PublishKafka2RecordCDP` in your data flow.

   You will configure the `PublishKafka2RecordCDP` processor in *Configure the processor for your data target*.
4. Connect the two processors to create a flow.
   a) Drag the connection icon from the first processor, and drop it on the second processor.

      A Create Connection dialog appears with two tabs: Details and Settings.
   b) Configure the connection.

      You can configure the connection's name, flowfile expiration time period, thresholds for back pressure, load balance strategy and prioritization.
   c) Click Add to close the dialog box and add the connection to your data flow.
5. Optionally, you can add success and failure funnels to your data flow, which help you see where flow files are being routed when your flow is running. Connect the `PublishKafka2RecordCDP` to these funnels with success and failure connections.

   If you want to know more about working with funnels, see the *Apache NiFi User Guide*.

Results

Your data flow may look similar to the following:

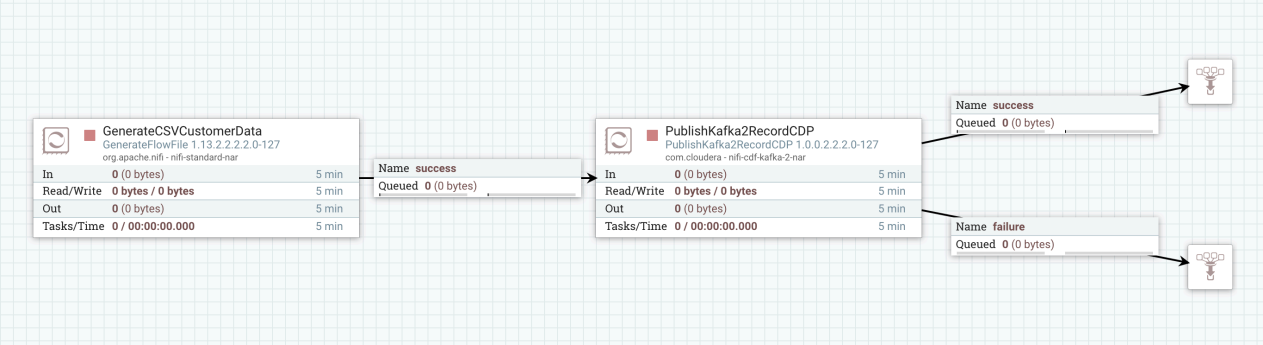

What to do next

Create controller services for your data flow. You will need these services later on, when configuring the `PublishKafka2RecordCDP` processor.

Related Information

Building a NiFi data flow

Apache NiFi User Guide

Create controller services for your data flow

Learn how to create and configure controller services for the Kafka ingest data flow.

About this task

You can add controller services that can provide shared services to be used by the processors in your data flow.

Procedure

1. To add a Controller Service to your flow, right-click on the canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.
3. Click the + button to display the Add Controller Service dialog.
4. Select the required Controller Service and click Add.
5. Click the Configure icon in the right-hand column and configure the necessary options.
6. When you have finished configuring the options you need, click Apply to save the changes.
7. Click the Enable button (flash) in the far-right column of the Controller Services tab to enable the Controller Service.

Example

The following controller services are used in this Kafka ingest example:

- HortonworksSchemaRegistry Controller Service
- CSVReader Controller Service
- AvroRecordSetWriter Controller Service

See below for property details.

HortonworksSchemaRegistry Controller Service

In this data flow, the data source is Kafka. To define the Schema Registry URL property of the HortonworksSchemaRegistry Controller Service, provide the URL of the schema registry that you want to connect to. You can find the master hostname on the Streams Messaging cluster overview page when selecting the Hardware tab.

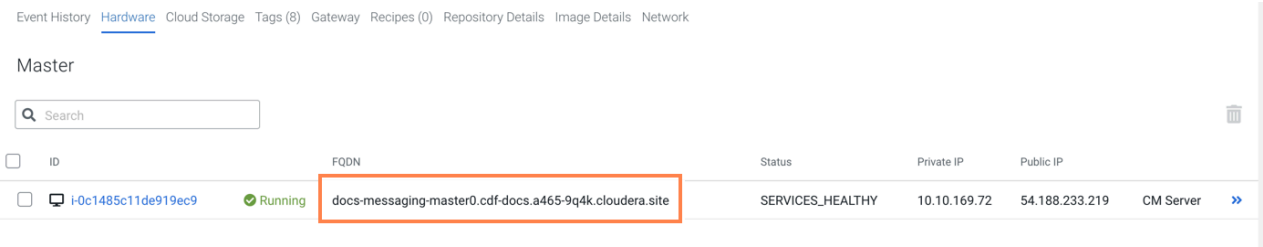

Table 1: HortonworksSchemaRegistry Controller Service properties

Property	Description	Example value for ingest data flow
Schema Registry URL	Provide the URL of the schema registry that this Controller Service should connect to, including version. In the format: https://host :7790/api/v1	https://docs-messaging-master0.cdf-docs.a465 -9q4k.cloudera.site:7790/api/v1
SSL Context Service	Specify the SSL Context Service to use for communicating with Schema Registry. Use the pre-configured SSLContextProvider.	Default NiFi SSL Context Service
Kerberos Principal	Specify the user name that should be used for authenticating with Kerberos. Use your Cloudera workload username to set this Authentication property.	srv_nifi-kafka-ingest
Kerberos Password	Provide the password that should be used for authenticating with Kerberos. Use your Cloudera workload password to set this Authentication property.	password

CSVReader Controller Service

Table 2: CSVReader Controller Service Properties

Property	Description	Example value for ingest data flow
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	Use String Fields From Header
Treat First Line as Header	Specify whether or not the first line of CSV should be considered a Header or a record.	true

AvroRecordSetWriter Controller Service

Table 3: AvroRecordSetWriter Controller Service Properties

Property	Description	Example value for ingest data flow
Schema Write Strategy	Specify how the schema for a Record should be added to the data.	HWX Content-Encoded Schema Reference
Schema Access Strategy	Specify how to obtain the schema to be used for interpreting the data.	Use 'Schema Name' Property
Schema Registry	Specify the Controller Service to use for the Schema Registry.	CDPSchemaRegistry
Schema Name	Specify the name of the schema to look up in the Schema Registry property. See the *Appendix* for an example schema.	customer

What to do next

Configure the processor for your data source.

Related Information

Adding Controller Services for data flows

Apache NiFi Documentation

Appendix – Schema example

Configure the processor for your data source

Learn how to configure a data source processor for the Kafka ingest data flow.

About this task

You can set up a data flow to move data into Apache Kafka from many different locations. This example assumes that you are using sample data generated by the `GenerateFlowFile` processor. If you are moving data from a certain location, see the *Apache NiFi Getting Started* for information on how to build a data flow, and about other data ingest processor options.

Procedure

1. Launch the Configure Processor window, by right clicking the `GenerateFlowFile` processor and selecting Configure. A configuration dialog with the following tabs is displayed: Settings, Scheduling, Properties, and Comments.
2. Configure the processor according to the behavior you expect in your data flow.

   The `GenerateFlowFile` processor can create many FlowFiles very quickly. Setting the run schedule to a reasonable value is important so that the flow does not overwhelm the system.

3. When you have finished configuring the options you need, save the changes by clicking the Apply button.

   Make sure that you set all required properties, because you cannot start the processor until all mandatory properties have been configured.

Example

The following settings and properties are used in this example:

Table 4: GenerateFlowFile processor scheduling

Scheduling	Description	Example value for ingest data flow
Run Schedule	Run schedule dictates how often the processor should be scheduled to run. The valid values for this field depend on the selected Scheduling Strategy.	500 ms

Table 5: GenerateFlowFile processor properties

	Description	Example value for ingest data flow
Custom text	If Data Format is text and if Unique FlowFiles is false, you can provide custom to be used as the content of the generated FlowFiles. The expression statement in the example value generates a random ID between 1 and 10 000, with random last names assigned.	`customer_id, customer_name ${random():mod(10000):plus(1)}, Smith ${random():mod(10000):plus(1)}, Johnson ${random():mod(10000):plus(1)}, Steward`

What to do next

Configure the processor for your data target.

Related Information

Configuring a processor

Getting started with Apache NiFi

Apache NiFi Documentation

Configure the processor for your data target

Learn how to configure a data target processor for the Kafka ingest data flow.

About this task

You can set up a data flow to move data to many locations. This example assumes that you are moving data to Apache Kafka. If you want to move data to a different location, see the other use cases in the *Cloudera DataFlow for Data Hub* library.

Procedure

1. Launch the Configure Processor window, by right-clicking the `PublishKafka2RecordCDP` processor and selecting Configure.

   A configuration dialog with the following tabs is displayed: Settings, Scheduling, Properties, and Comments.
2. Configure the processor according to the behavior you expect in your data flow.

3. When you have finished configuring the options you need, click Apply to save the changes.

   Make sure that you set all required properties, because you cannot start the processor until all mandatory properties are configured.

Example

In this example data flow, we are writing data to Kafka. You can create the modified Kafka broker URLs using the broker hostnames and adding port :9093 to the end of each FQDN. You can find the hostnames on the Streams Messaging cluster overview page when selecting the Hardware tab.

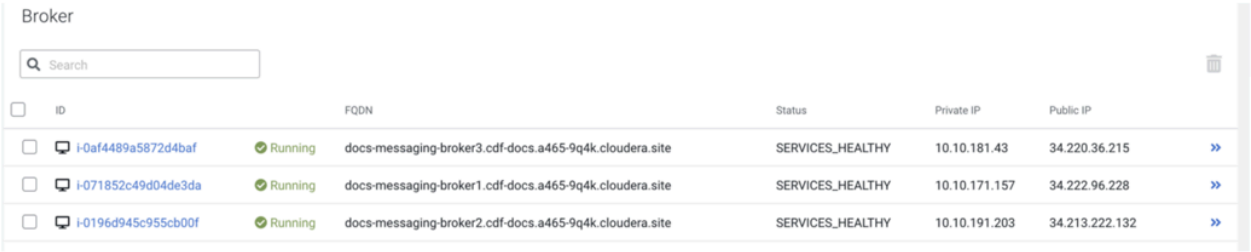

Note: Property values can be parameterized. For example, you can create a parameter context to hold Kafka connection information and apply it to the Kafka Brokers property instead of adding the broker URLs individually.

The following properties are used for the `PublishKafka2RecordCDP` processor:

Table 6: PublishKafka2RecordCDP processor properties

Property	Description	Example value for ingest data flow
Kafka Brokers	Provide a comma-separated list of known Kafka Brokers. In the format <host>:<port>.	Docs-messaging-broker1.cdf-docs.a465-9q4k.cloudera.site:9093,docs-messaging-broker2.cdf-docs.a465-9q4k.cloudera.site:9093,docs-messaging-broker3.cdf-docs.a465-9q4k.cloudera.site:9093
Topic Name	Provide the name of the Kafka Topic to publish to.	Customer
Record Reader	Specify the Record Reader to use for incoming FlowFiles.	ReadCustomerCSV
Record Writer	Specify the Record Writer to use in order to serialize the data before sending to Kafka.	WriteCustomerAvro
Use Transactions	Specify whether or not NiFi should provide Transactional guarantees when communicating with Kafka.	false
Delivery Guarantee	Specify the requirement for guaranteeing that a message is sent to Kafka.	Guarantee Single Node Delivery
Security Protocol	Specify the protocol used to communicate with Kafka brokers.	SASL_SSL
SASL Mechanism	Specify the SASL mechanism to use for authentication.	PLAIN
Username	Use your Cloudera workload username to set this Authentication property.	srv_nifi-kafka-ingest

Property	Description	Example value for ingest data flow
Password	Use your Cloudera workload password to set this Authentication property. This will use the specified Cloudera user to authenticate against Kafka. Make sure that this user has the correct permissions to write to the specified topic.	password
SSL Context Service	Specify the SSL Context Service to use for communicating with Kafka. Use the pre-configured SSLContextProvider.	Default NiFi SSL Context Service

What to do next

Your data flow is ready to ingest data into Kafka. You can now start the data flow.

Related Information

Configuring a processor

Apache NiFi User Guide

Data ingest use cases in Cloudera DataFlow for Data Hub

Start the data flow

Learn how to start your Kafka ingest data flow.

Procedure

1. Select all the data flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternatively, right-click a single component and choose Start from the context menu.

Results

Your flow should be running without any errors. Data should be generated in the form of FlowFiles, and the files should be written to your designated Kafka topic.

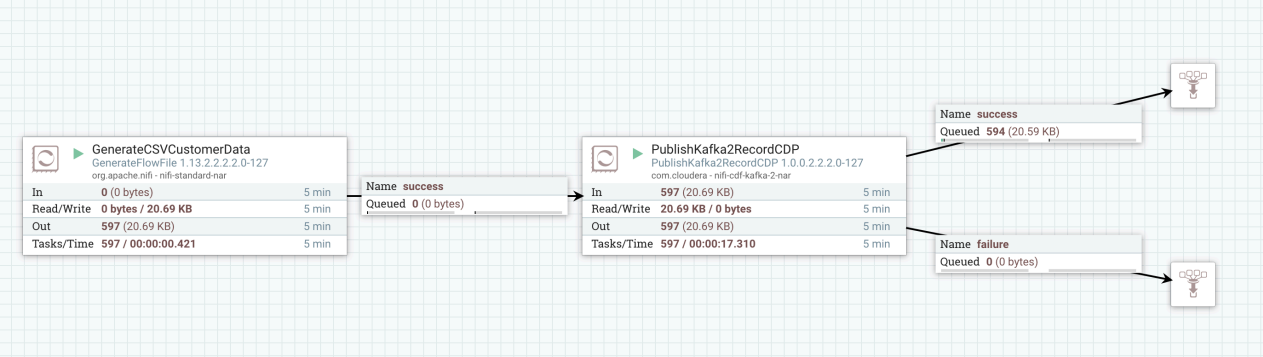

What to do next

It is useful to check that data is running through your flow without any errors.

Related Information

Apache NiFi User Guide

Verify data flow operation

Learn how you can verify the operation of your Kafka ingest data flow.

About this task

There are a number of ways to check that data is running through the flow you have built and it actually appears in Kafka.

Procedure

- You can verify that NiFi processors are not producing errors.
- You can look at the processors in the UI to see the amount of data that has gone through them. You can also right-click on the processors, or on connections to view status history.

- You can verify in Streams Messaging Manager that messages appear in the defined Kafka topic.

  To check this:

  a) Navigate to your Streams Messaging Data Hub cluster.

  b) Click Streams Messaging Manager in the Services section of the Cluster overview page.

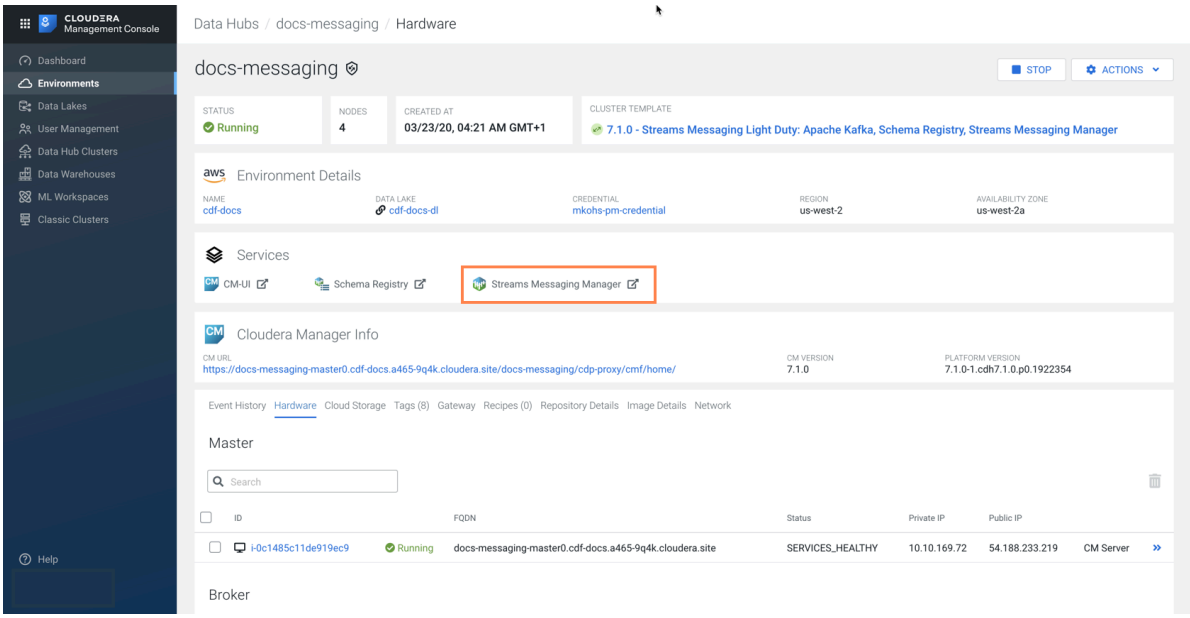

  c) Search for your previously created topic and verify that your NiFi flow is visible, and it shows the producer and consumer as well.

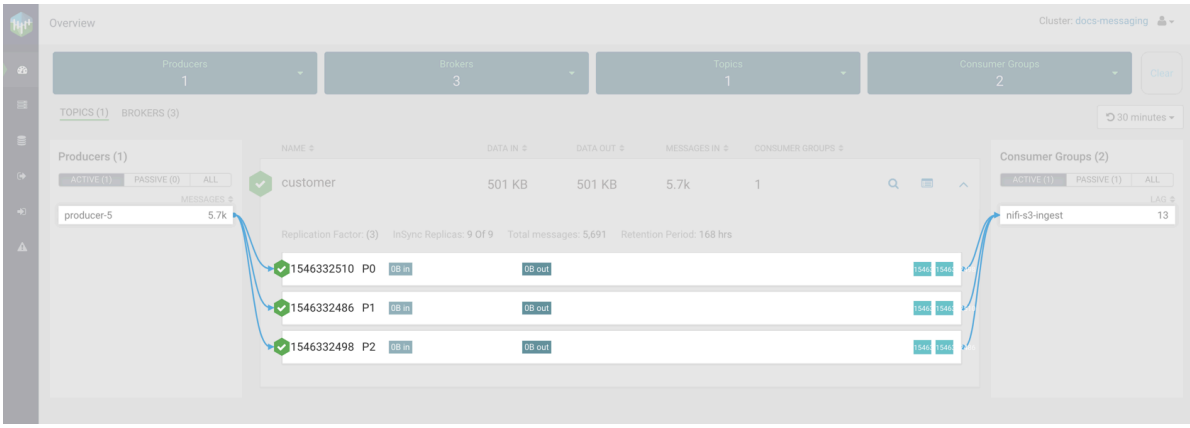

  d) To check the messages written to the Kafka topic, click the blue Profile icon at the end of the Topic row and choose the Data Explorer tab.

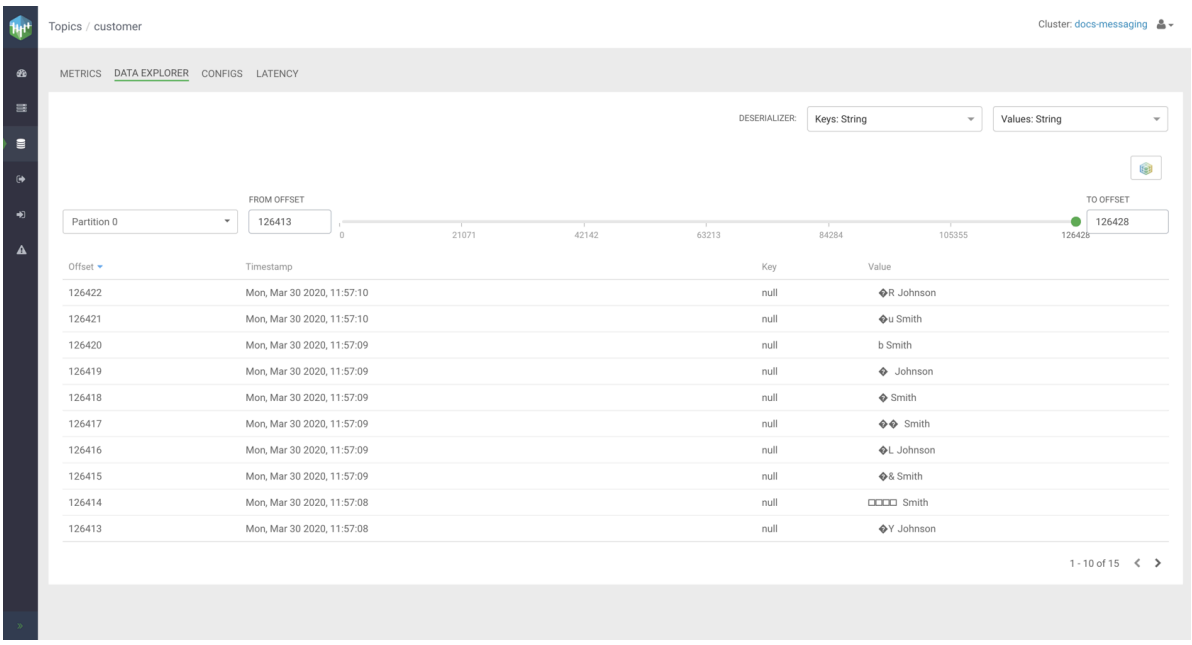

e) Since we are using an Avro schema in this example, the messages are encoded and not entirely readable. You can select the Avro Deserializer (Avro in both Key and Value fields) and make the data readable.

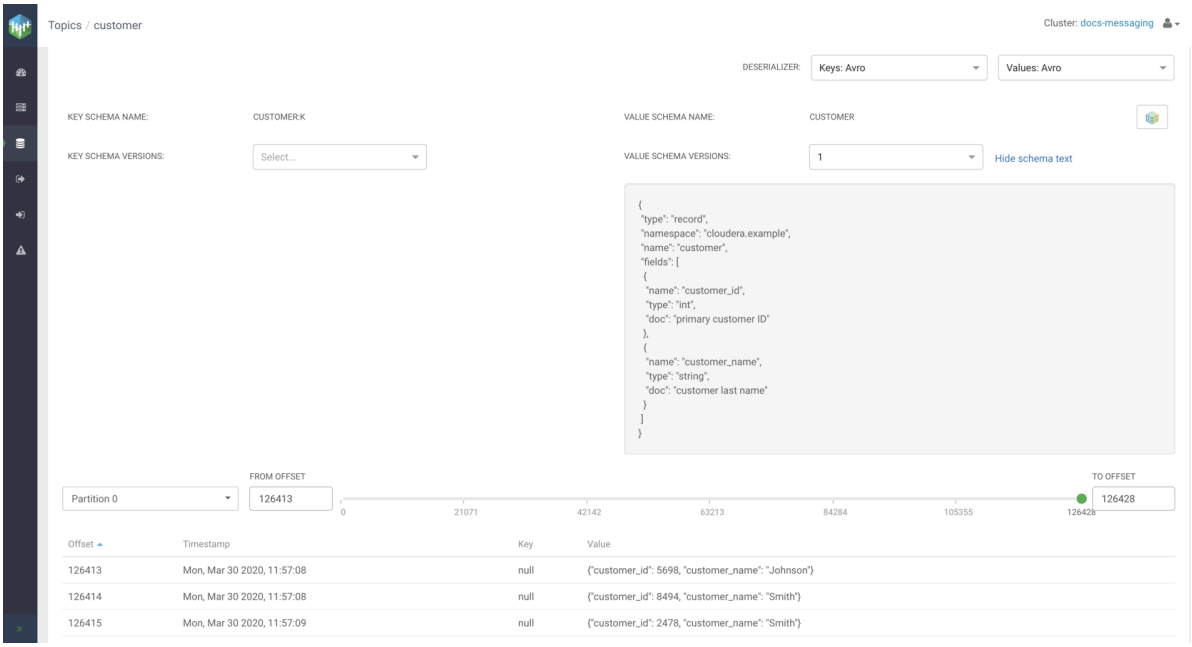

Related Information

Apache NiFi User Guide

Monitoring end to end latency for Kafka topic

To monitor end-to-end latency of Kafka topics, you can monitor the count of consumed messages across all Kafka consumer groups and the time taken by all Kafka consumer groups to consume messages that are produced in a Kafka topic, in a graphical way. You can also monitor the same for each Kafka consumer group, each client in a Kafka consumer group, and each partition in a Kafka topic.

About this task

Perform the following steps to monitor end-to-end latency in the Streams Messaging Manager UI:

Procedure

1. Go to Topics in the Streams Messaging Manager UI.
2. Select the topic you want to verify the details about.
3. Click the Profile icon beside the topic you select.

   This takes you to the Metrics page where you can find the Messages Consumed and End-to-end Latency graphs along with other topic details. On the Metrics page, these two graphs provide you an aggregated result of latency and count of consumed messages across all consumer groups.
4. To verify the details by individual consumer groups, clients, and partitions, go to the Latency tab.

   The **Latency** page appears as shown in the following image:

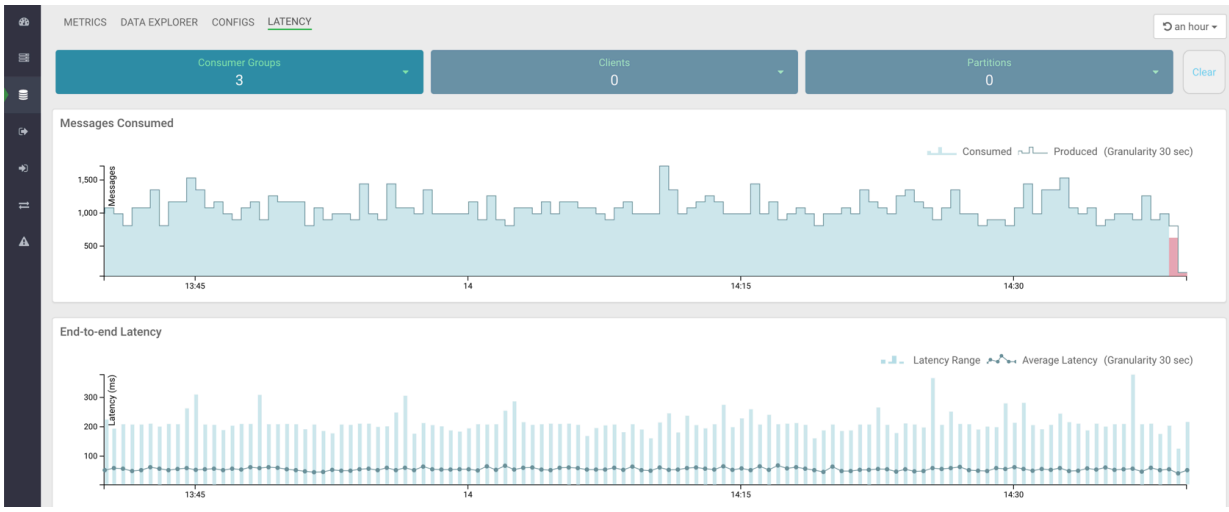

   The Latency view gives you a powerful snapshot of the end-to-end latency scenario: number of consumer groups for a topic, number of clients inside a particular consumer group, and number of partitions in a topic along with the Messages Consumed and End-to-end Latency graphs.

5. Select any consumer group from the Consumer Groups drop-down, as shown in the following image:

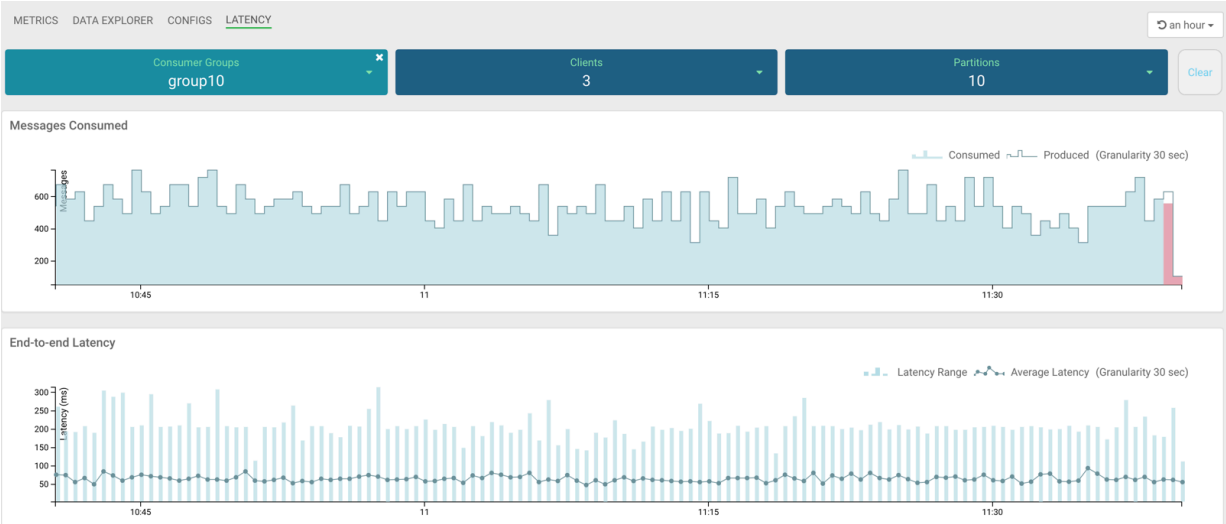

In the image, group10 consumer group is selected. The Latency tab displays that there are 3 clients in the group10 consumer group, and 10 partitions are available in the topic.

6. Select any client from the Clients drop-down, as shown in the following image:

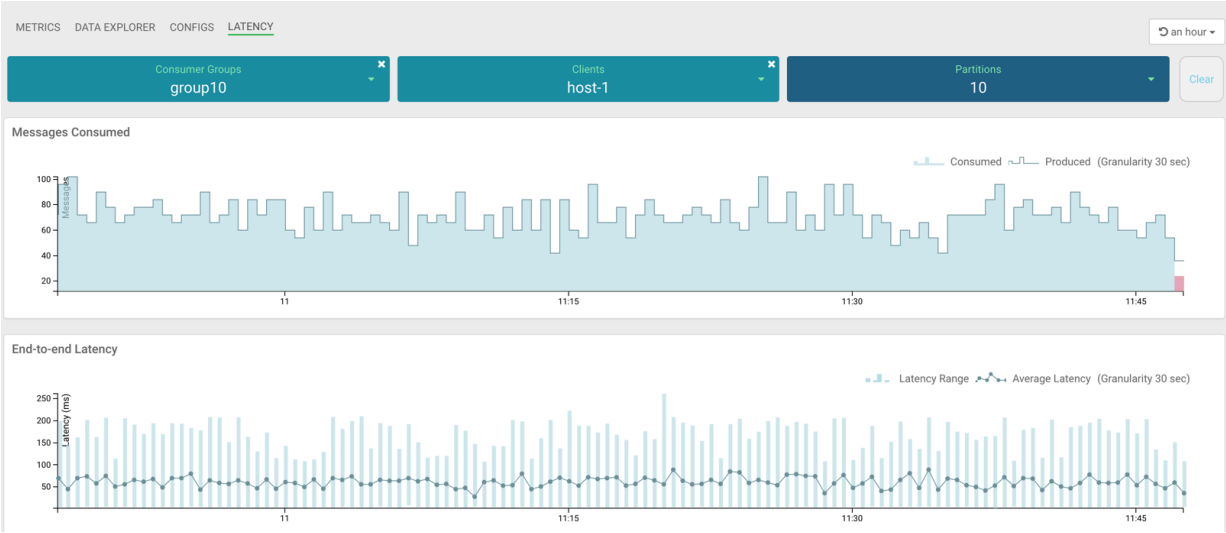

In the image, host-1 client is selected. At this instance, the Messages Consumed and End-to-end Latency graphs display data for only the host-1 client. Here you can monitor the number of messages produced, number of messages consumed, latency range, and average latency for host-1 only. Hover your mouse over the graphs and get data at any point of time in the selected time range. You can see in the Messages Consumed graph that host-1 consumed all the messages that are produced and actively consuming data at the latest time. You can see in the End-to-end Latency graph that the latency range and average latency are under 250 milliseconds.

7. To get details about the partitions from where host-1 is consuming data, click Partitions.

   The list of partitions in the topic appears, as shown in the following image:

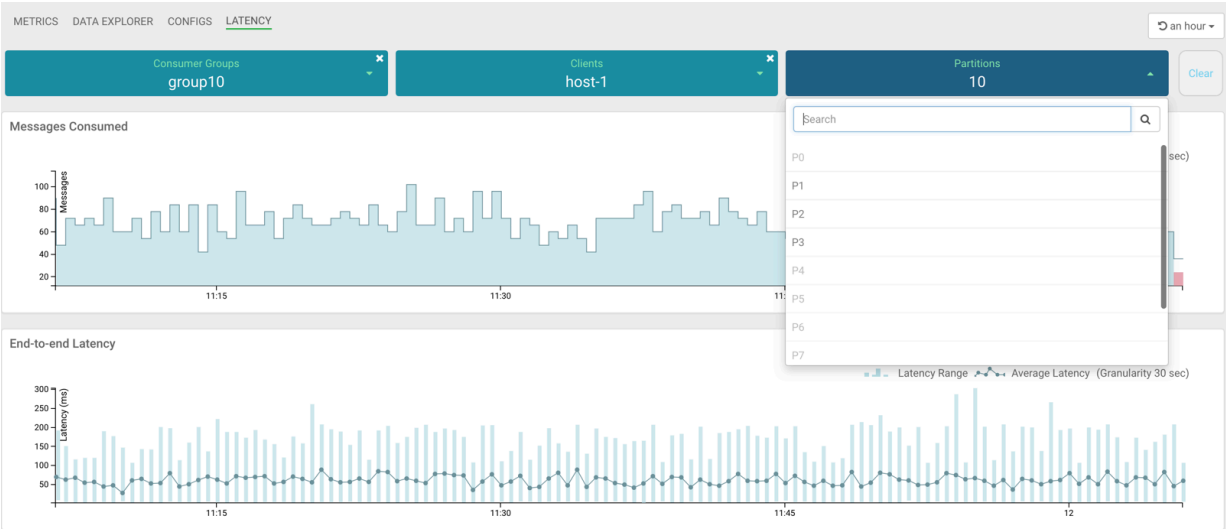

   In the image, you can see that host-1 is consuming data from 3 partitions: P1, P2, and P3. Other partitions are inactive for host-1.

8. Select any active partition from the list.

   The Latency tab displays the transaction details between host-1 and the selected partition (for example, P1), as shown in the following image:

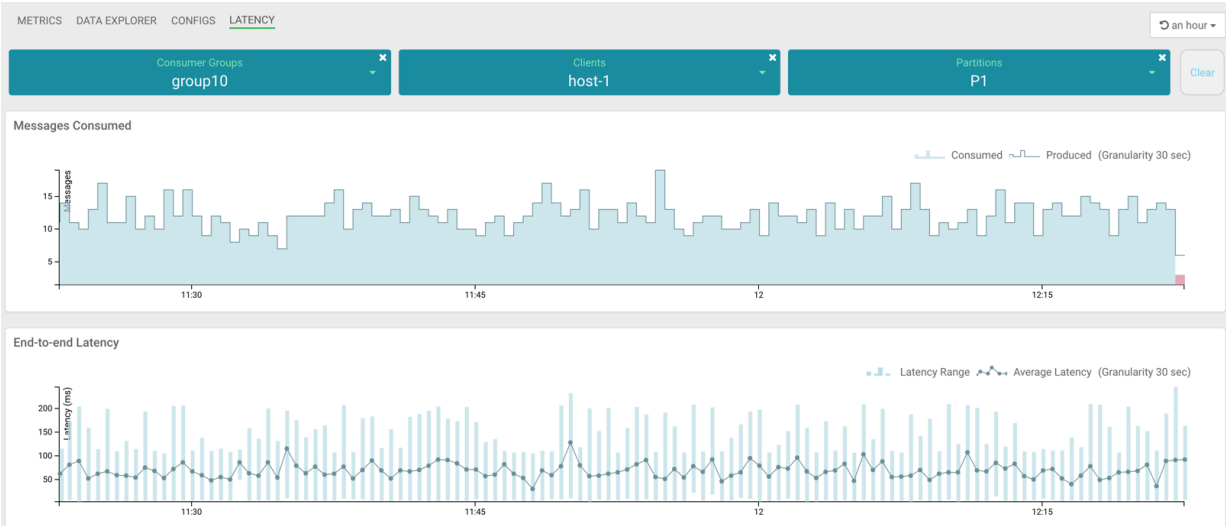

   Now you have got details for host-1 client. Likewise, you can fetch details for other clients.

9. Follow steps 6 through 8 to fetch data for all other clients.

10. Follow steps 5 through 8 to fetch data for all other consumer groups.

- To clear all the selections at once, click the Clear button at the top-right corner of the page.
- To clear selection for Consumer Groups, Clients, or Partitions, click the delete icon located on each drop-down.
- To select a different time range, use the Time Range and Quick Ranges options at the top-right corner of the page, as shown in the following image:

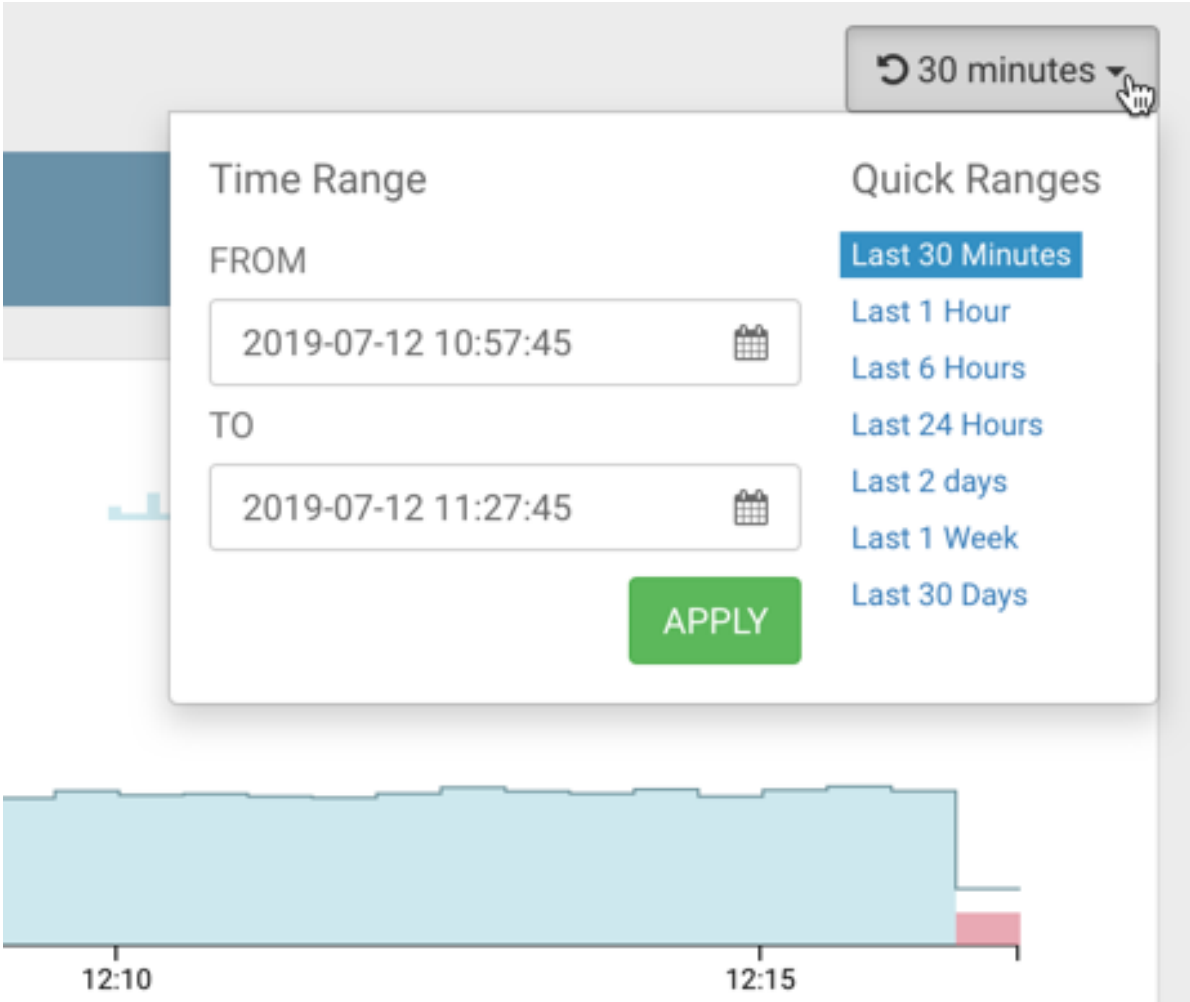

Monitoring your data flow

Learn about the different monitoring options for your Kafka ingest data flow.

You can monitor your data flow for information about health annd status, and for details about the operation of processors and connections. NiFi records and indexes data provenance information, so you can troubleshoot any issues in real time.

- Data statistics are rolled up on a summary screen (the little table icon on the top right toolbar which lists all the processors). You can use the `MonitorActivity` processor to alert you, if for example you have not received any data in your flow for a specified period of time.
- If you are worried about data being queued up, you can check how much data is currently queued. Process groups show the totals for all queues within them. This can often indicate if there is a bottleneck in your flow, and how far the data has got through in that pipeline.
- Another option to confirm that data has fully passed through your flow is to check out data provenance to see the full history of your data.

Related Information

Monitoring a Data Flow

Next steps

Learn some ideas on how to build on top of the Kafka ingest flow management use case.

You have built a data flow that generates sample data and moves this data into the Kafka topic. The data moved to Kafka can be picked up by any external services, or you can build another NiFi data flow to ingest these Kafka messages to for example HBase or Hive.

Related Information

Data ingest use cases in ClouderaData Flow for Data Hub

Appendix – Schema example

This is a sample schema for the Kafka ingest data flow presented in this use case.

customer.json

```
{
 "type": "record",
 "namespace": "cloudera.example",
 "name": "customer",
 "fields": [
  {
   "name": "customer_id",
   "type": "int",
   "doc": "primary customer ID"
  },
  {
   "name": "customer_name",
   "type": "string",
   "doc": "customer last name"
  }
 ]
}
```

Cloudera DataFlow for Data Hub 7.3.1

Ingesting data into Apache Kudu in Cloudera Public Cloud

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ingesting data into Kudu **4**
Understand the use case 4
Meet the prerequisites 4
Create the Kudu target table 5
Build the data flow 6
Configure the Controller Service 8
Configure the processor for your data source 9
Configure the processor for your data target 11
Start your data flow 12
Verify that you can write data to Kudu 12

Next steps **13**

Ingesting data into Kudu

You can use an Apache NiFi data flow to ingest data into Apache Kudu in the Cloudera Public Cloud following these steps.

Understand the use case

You can use Apache NiFi to move data from a range of locations, into Kudu in your Real-time Data Mart cluster in Cloudera Public Cloud.

Time series use cases analyse data obtained during specified intervals, and enable you to improve performance based on available data. Examples include:

- Optimizing yield or yield quality in a manufacturing plant
- Dynamically optimizing network capacity during peak load of better telecommunications uptime and services

These use cases require that you store events at a high frequency, while providing ad-hoc query and record update abilities. You can use Apache NiFi data flows into Apache Kudu and Apache Impala in a Cloudera Public Cloud Real-time Data Mart cluster to accomplish these goals.

Time Series use cases also need flexible ingest, at scale. You can use NiFi in a Data Hub Flow Management cluster, in the same environment, to build a data flow that reads your source data, modifies it to match the Kudu schema, and ingests into your Real-time Data Mart cluster for fast analytics and dashboard serving. Once you have successfully built the data flow in NiFi, you can query the data stored in Kudu by running Impala queries in Hue on the Real-time Data Mart cluster.

This use case walks you through the steps associated with creating an ingest-focused data flow from Apache Kafka in a Streaming cluster in Cloudera Public Cloud, into Apache Kudu in a Real Time Data Mart cluster, in the same Cloudera Public Cloud environment. This will get you started with creating a data ingest data flow into Kudu. If you are moving data from a location other than Kafka, review the *Getting Started with Apache NiFi* for information about how to build a data flow, and about other data ingest processor options.

Related Information

Getting Started with Apache NiFi

Ingesting data into Apache Kafka in Cloudera Public Cloud

Ingesting data into Apache Hive in Cloudera Public Cloud

Ingesting data into Apache HBase in Cloudera Public Cloud

Ingesting Data into Amazon S3 Buckets

Ingesting Data into Azure Data Lake Storage

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a Cloudera Public Cloud environment.
- You have a workload username and password set to access Cloudera Data Hub clusters. The predefined resource role of this user is at least "EnvironmentUser". This resource role provides the ability to view Cloudera Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the Cloudera Public Cloud environment.
- You have a Flow Management cluster running Apache NiFi.
- You have a Real-Time Data Mart cluster running Kudu, Impala, and Hue. This cluster must be in the same Cloudera environment as the Flow Management cluster.

- You have Kudu master detail from the “Hardware” tab in your Real Time Data Mart cluster, so that you can configure the `PutKudu` NiFi processor. For example:

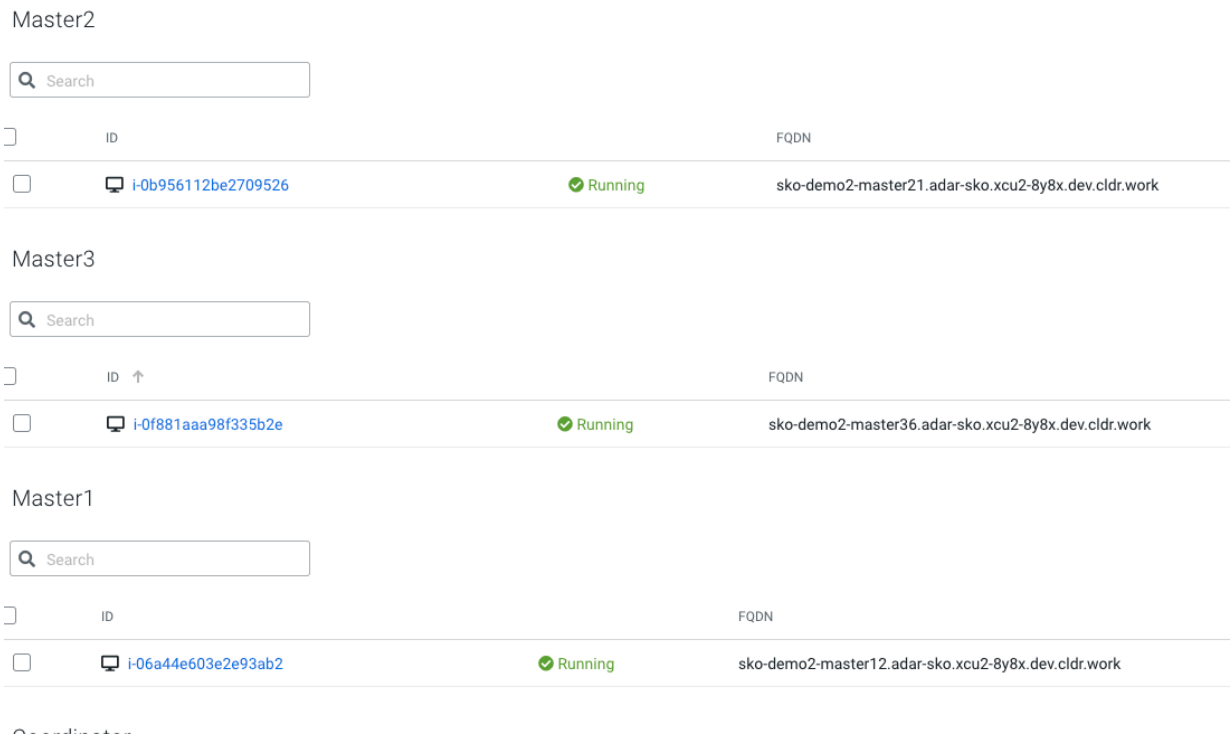

Related Information

AWS Onboarding Quickstart
Azure Onboarding Quickstart
Understanding roles and resource roles
CDP workload user
Setting up your Flow Management cluster

Create the Kudu target table

Before you can ingest data into Kudu, you need a target Kudu table prepared to receive the data flow.

Procedure

1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
2. Click the Tables icon on the left pane.
3. Select the default database, and click + New to create a new table.
4. In the Type field, select Manually and click Next.
5. Provide the table Name, Format, Primary keys, and any partitions.
6. Click Submit.

Results

The newly created table displays in the default database Tables pane.

Example

In this example, you create a table with the following values:

Field name	Value
Name	customer
Format	Kudu
Primary keys	customer_id
Partitions	No specified partitions
+ Add Field to include custom field valuecustomer_id	Type equals int
+ Add Field to include a custom field value customer_name	Type equals string

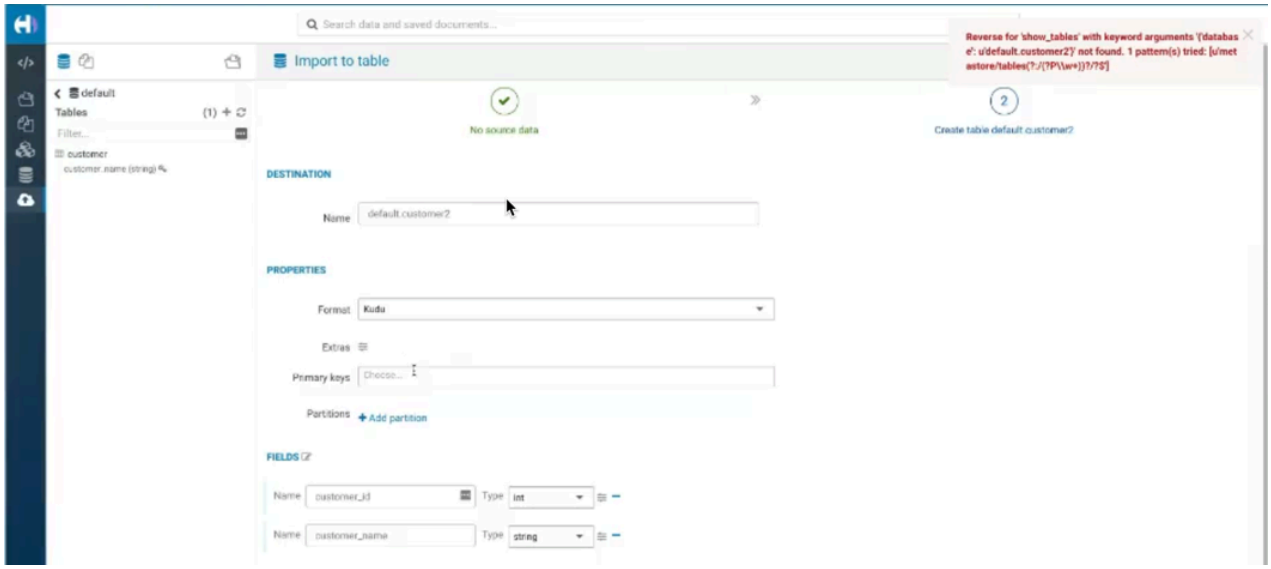

What to do next

Check the Kudu UI Tables tab, for the name of the table you created. You will need this table name later on, when configuring the NiFi `PutKudu` processor.

Depending on how you created it, it has one of the following formats.

- impala::default.customer
- default.customer

Related Information

Creating Kudu tables

Build the data flow

From the Apache NiFi canvas, set up the elements of your data flow. This involves opening NiFi in CDP Public Cloud, adding processors to your NiFi canvas, and connecting the processors.

About this task

You should use the `PutKudu` processor to build your Kudu ingest data flows.

Procedure

1. Open NiFi in Cloudera Public Cloud.
   a) To access the NiFi service in your Flow Management cluster, navigate to the Cloudera Management Console and click Data Hub Clusters.
   b) Click the tile representing the Flow Management cluster with which you want to work.
   c) Click the NiFi icon in the Services section of the Cluster overview page to access the NiFi UI.

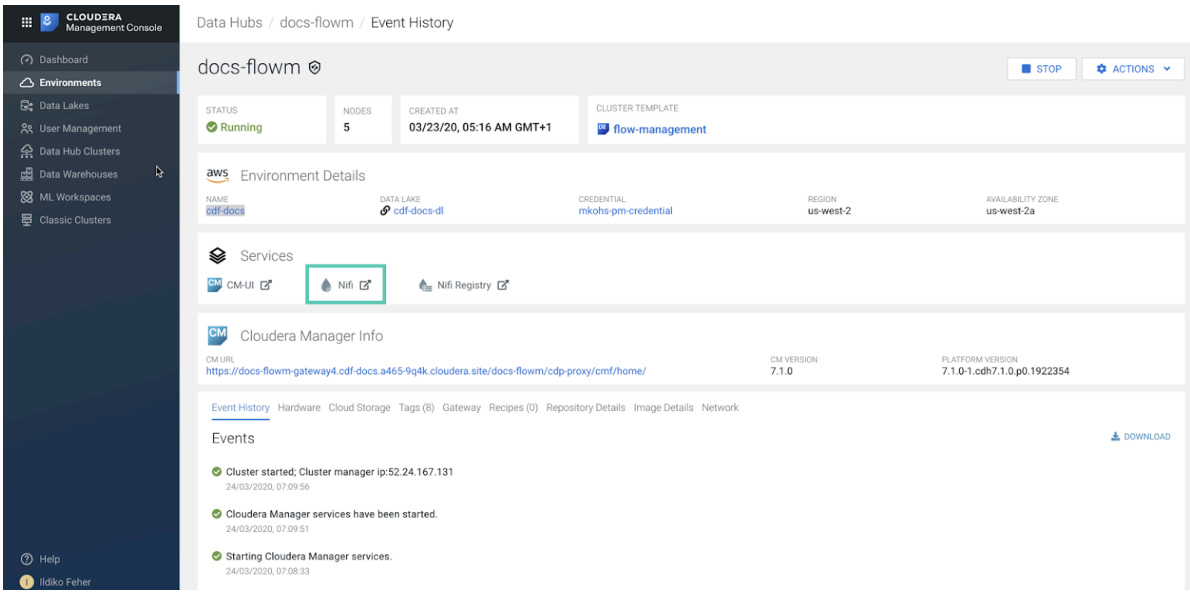

2. Add the NiFi Processors to your canvas.
   a) Select the Processor icon from the Cloudera Flow Management actions pane, and drag a processor to the Canvas.
   b) Use the Add Processor filter box to search for the processor you want to add, and then click Add.
   c) Add each of the processors you want to use for your data flow.
3. Connect the two processors to create a flow.
   a) Click the connection icon in the first processor, and drag it to the second processor.
   b) A Create Connection dialog displays. It has Details and Settings tabs. You can configure the connection's name, FlowFile expiration time period, thresholds for back pressure, load balance strategy, and prioritization.
   c) Click Add to close the dialog box and add the connection to your flow.

   Optionally, you can add success and failure funnels to your data flow, which help you see where flow files are routed when your data flow is running.

Results

Your data flow may look similar to the following:

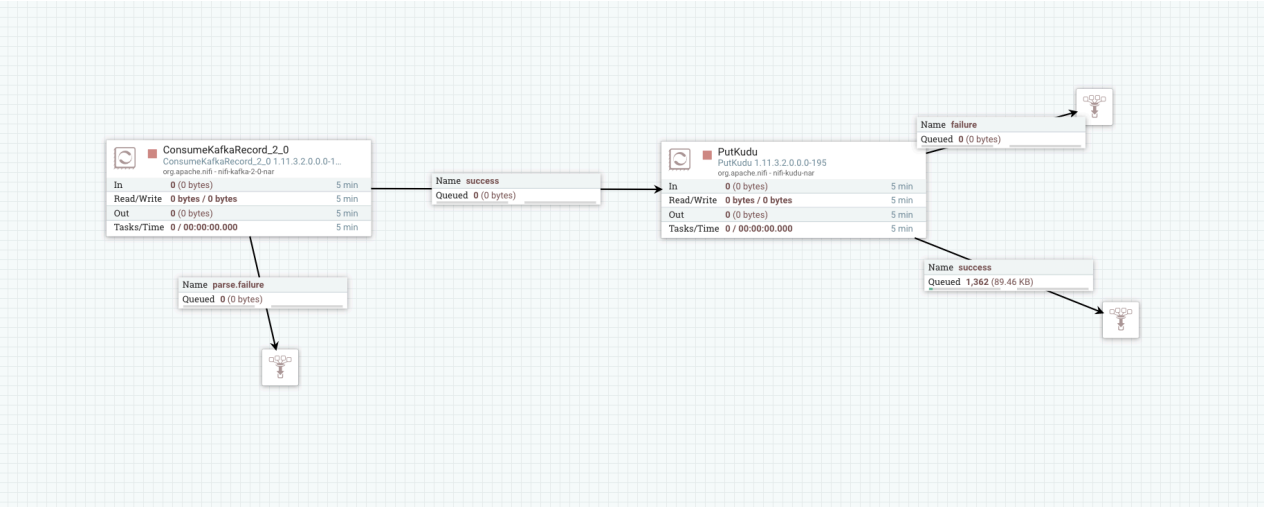

What to do next

Create the Controller Service for your data flow. You will need these services later on as you configure your data flow target processor.

Related Information

Ingesting data into Apache Kafka in Cloudera Public Cloud

Consume KafkaRecord_2_0

PutKudu

Building a data flow

Configure the Controller Service

You can add Controller Services to provide shared services to be used by the processors in your data flow. You will use these Controller Services later when you configure your processors.

About this task

You must define the Controller Services for the processors in your data flow in the onfiguration of the root process group where they will be used.

Procedure

1. To add a Controller Service to your flow, right-click on the Canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.
3. Click the + button to display the Add Controller Service dialog.

4. Select the required Controller Service and click Add.

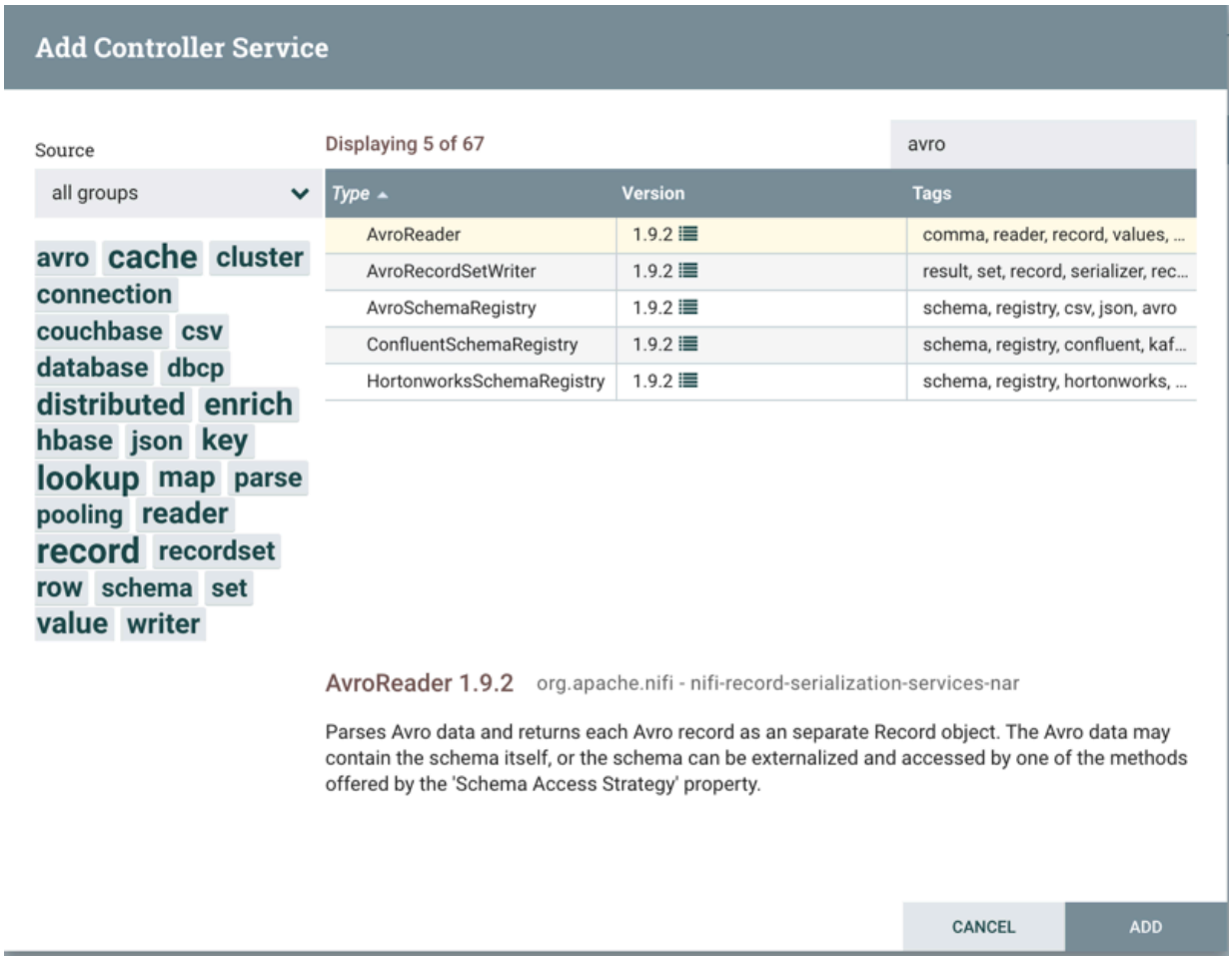

5. Perform any necessary Controller Service configuration tasks by clicking the Configure icon in the right-hand column.

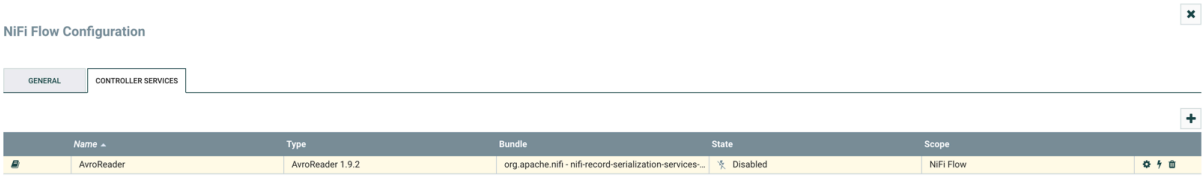

6. When you have finished configuring the options you need, save the changes by clicking the Apply button.
7. Enable the Controller Service by clicking the Enable button (flash) in the far-right column of the Controller Services tab.

Configure the processor for your data source

You can set up a data flow to move data from many locations into Apache Kudu. This example assumes that you are configuring `ConsumeKafkaRecord_2_0`. If you are moving data from a location other than Kafka, review *Getting Started with Apache NiFi* for information about how to build a data flow, and about other data consumption processor options.

Before you begin

- Ensure that you have the Ranger policies required to access the Kafka consumer group.
- This use case demonstrates a data flow using data in Kafka, and moving it to Kudu. To review how to ingest data to Kafka, see *Ingesting Data to Apache Kafka*.

Procedure

1. Launch the Configure Processor window, by right-clicking the processor and selecting Configure.
2. Click the Properties tab.
3. Configure `ConsumeKafkaRecord_2_0` with the required values.

   The following table includes a description and example values for the properties required to configure an ingest data flow. For a complete list of `ConsumeKafkaRecord_2_0`, see the *processor documentation*.

Property	Description	Value for ingest to Kudu data flow
Kafka Brokers	Provide a comma-separated list of known Kafka Brokers in the format <host>:<port>.	`Docs-messaging-broker1.cdf-docs.a465-9q4k.cloudera.site:9093, docs-messaging-broker2.cdf-docs.a465-9q4k.cloudera.site:9093, docs-messaging-broker3.cdf-docs.a465-9q4k.cloudera.site:9093`
Topic Name(s)	Enter the name of the Kafka topic from which you want to get data.	*CUSTOMER*
Topic Name Format	Specify whether the topics are a comma separated list of topic names, or a single regular expression.	*NAMES*
Record Reader	Specifies the record reader to use for incoming Flow Files. This can be a range of data formats.	*AVROREADER*
Record Writer	Specifies the format you want to use in order to serialize data before for sending it to the data target. Note: Ensure that the source record writer and the target record reader are using the same Schema.	*AVRORECORDSETWRITER*
Honor Transactions	Specifies whether or not NiFi should honor transactional guarantees when communicating with Kafka.	*FALSE*
Security Protocol	Specifies the protocol used to communicate with brokers. This value corresponds to Kafka's 'security.protocol' property.	*SASL_SSL*
SASL Mechanism	The SASL mechanism to use for authentication. Corresponds to Kafka's 'sasl.mechanism' property.	*PLAIN*
Username	Specify the username when the SASL Mechanism is PLAIN or SCRAM-SHA-256.	*SRV_NIFI-KUDU-INGEST*
Password	Specify the password for the given username when the SASL Mechanism is PLAIN or SCRAM-SHA-256.	*PASSWORD1!*
SSL Context Service	Specifies the SSL Context Service to use for communicating with Kafka.	*DEFAULT*

Property	Description	Value for ingest to Kudu data flow
Group ID	A Group ID is used to identify consumers that are within the same consumer group. Corresponds to Kafka's 'group.id' property.	*NIFI-KUDU-INGEST*

What to do next

Configure the processor for your data target.

Related Information

Getting Started with Apache NiFi
Building a data flow
ConsumeKafkaRecord_2_0
Ingesting data into Apache Kafka in Cloudera Public Cloud

Configure the processor for your data target

You can set up a data flow to move data into many locations. This example assumes that you are moving data into Apache Kudu using `PutKudu`. If you are moving data into another location, review *Getting Started with Apache NiFi* for information about how to build a data flow, and about other data ingest processor options.

About this task

You can set up a data flow to move data into many locations. This example assumes that you are moving data into Apache Kudu in a Real-time Data Mart cluster, using `PutKudu`. If you are moving data into another location, review *Getting Started with Apache NiFi* for information about how to build a data flow, and about other data ingest processor options.

Before you begin

You have created a machine user in Cloudera User Management and synchronized this user to your Cloudera Environment.

Procedure

1. Launch the Configure Processor window, by right-clicking the processor and selecting Configure.
2. Click the Properties tab.
3. Configure `PutKudu` with the required values.

   The following table includes a description and example values for the properties required to configure an ingest data flow into Kudu. For a complete list of `PutKudu`, see the *processor documentation*.

Property	Description	Value for ingest to Kudu data flow
Kudu masters	Comma separated list of the fully qualified domain names (FQDNs) of the Kudu masters to connect to.	These are the values you obtained in *Meet the prerequisites* before you began to build your data flow.

Property	Description	Value for ingest to Kudu data flow
Table name	The name of the Kudu table into which you want to ingest data.	This is the table you created in *Create the Kudu target table*. **Note:** It is important that the specified Kudu table name match the format of the table you created. Depending on how you created it, it has one of the following formats. • impala::default.customer • default.customer
Kerberos Principal	Specify the Cloudera User name you are using to perform this workflow.	Specify the Cloudera User Name you created and synced with your Cloudera Environment in *Meet the prerequisites*.
Kerberos Password	Specify the password for the CDP User you are using to perform this workflow.	For example: Password1!
RecordReader	The service for reading records from incoming flow files.	AvroReader
Kudu Operation Type	Specify operationType for this processor. Valid values are: • INSERT • INSERT_IGNORE • UPSERT • UPDATE • DELETE	UPSERT

Related Information

Getting Started with Apache NiFi

PutKudu

Start your data flow

Start your data flow to verify that you have created a working dataflow and to begin your data ingest process.

Procedure

1. To initiate your data flow, select all the flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternately, right-click a single component and choose Start from the context menu.

Verify that you can write data to Kudu

Once you have configured and started your data flow, you can verify that you are successfully ingesting data into Kudu.

About this task

There are a number of ways to check that data is running through the flow you have built and it actually appears in Hive.

Procedure

1. From your Real-time Data Mart cluster, go to the Impala UI.

2. Search for your data ingest target table.

   For example, in *Create the Kudu target table*, you created a table called default.customer.

```
describe formatted detail.customer;
select * from default.customer;
```

Results

You are able to see data flowing into your default.customer table, with customer_id and customer_name entries.

Next steps

Provides information on what to do once you have moved data into Kudu in Cloudera.

You have built a simple data flow for an easy way to move data to Kudu. This example data flow enables you to easily design more complex data flows for moving and processing data in Kudu. Here are some ideas for next steps:

- Review *NiFi documentation* to learning more about building and managing data flows.
- Review *NiFi Registry documentation* for more information about versioning data flows.
- Review the *Cloudera Runtime data storage information about Apache Kudu* for information about using Kudu in Cloudera.

Related Information

NiFi documentation

NiFi Registry documentation

Cloudera Runtime data storage information about Apache Kudu

Cloudera Runtime data storage information About Apache Kudu

CDF for Data Hub 7.2.11

Ingesting data into Cloud Object Stores with RAZ Authorizations

Date published: 2021-09-20
Date modified: 2021-10-04

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ingesting data into CDP Object Stores with RAZ authorization.....4
Understand the use case.....4
Meet the prerequisites.....4
Build the data flow.....5
Configure each object store processor.....6
Set permissions in Ranger.....12
Start the data flow.....15
Verify data flow operation.....15

Monitoring your data flow.....16

Next steps.....16

Ingesting data into CDP Object Stores with RAZ authorization

You can use an Apache NiFi data flow to ingest data into your object stores in a CDP Public Cloud environment that is enabled for Ranger Authorization Service (RAZ). Follow these steps:

Understand the use case

Learn how you can use object store processors to build an end-to-end flow that ingests data into Cloud object stores in a CDP environment that is enabled for fine-grained access control through RAZ.

The object store processors are:

- `ListCDPObjectStore`: Lists the FlowFiles in the source bucket that you want to ingest.
- `FetchCDPObjectStore`: Fetches the FlowFiles from the source bucket. If the files are compressed, this processor uncompresses the files.
- `PutCDPObjectStore`: Writes the contents of a FlowFile to the specified directory in the target bucket.
- `DeleteCDPObjectStore`: Deletes the ingested data from the source bucket.

These object store processors have the following advantages:

- The configuration of these processors is simple and easy.
- These processors can access the underlying object store of the cloud provider where the NiFi cluster is running, If you are running NiFi in Azure, these processors can be used to access ADLS. If you are running NiFi in AWS, these processors can be used to access S3. If you are running NiFi in Google Cloud, these processors can be used to access Google Cloud Storage.
- These processors are cloud-agnostic. If you have a multi-cloud architecture with, for example, a cluster running in AWS and another running in Azure, you can move this flow from one cloud provider to another with minimal change such as the input bucket.
- These processors are integrated with RAZ. With RAZ you can define policies in Ranger to specify who has access to what in the object store.
- Unlike HDFS processors that can only access the buckets that is configured for the data lake for your CDP environment, the object store processors can access any location in the underlying object store.

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a CDP username and password set to access Data Hub clusters. The predefined resource role of this user is at least EnvironmentUser. This resource role provides the ability to view Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the CDP Public Cloud environment.
- You have a Flow Management Data Hub cluster running in your CDP Public Cloud environment.
- Your CDP user has been added to the appropriate pre-defined Ranger access policies to allow access to the NiFi UI.
- You have a source bucket on your object store from which you want to fetch data.
- You have created a target bucket on your object store for the processed data to be moved into.
- You have created a role with policies attached allowing read and write access to the object store you want to use in your data flow.

Note: To leverage the RAZ capabilities for Ranger defined policies that determine fine-grained access control to the object store, enable your CDP Public Cloud environment for RAZ.

If the environment is not RAZ enabled, the processors will not integrate with RAZ but will still work using IDBroker and the corresponding mappings.

Build the data flow

Learn how you can build a data flow using the object store processors to fetch data from one bucket, and after data transformation, ingest the data into another bucket. This involves opening Apache NiFi in your Flow Management cluster, adding processors and other data flow objects to your canvas, and connecting your data flow elements.

About this task

Use the `ListCDPObjectStore` and `FetchCDPObjectStore` processors to list and fetch data from your source bucket. Then, after transforming the data, use the `PutCDPObjectStore` to push the transformed data into the target bucket. Use the `DeleteCDPObjectStore` to delete your data in the source bucket.

Regardless of the type of flow you are building, the first steps in building your data flow are generally the same. Open NiFi, add your processors to the canvas, and connect the processors to create the flow.

Procedure

1. Open NiFi
   in
   Data Hub.
   a) To access the NiFi service in your Flow Management
      Data Hub cluster, navigate to Management Console service Data Hub Clusters .
   b) Click the tile representing the Flow Management Data Hub cluster you want to work with.
   c) Click NiFi in the Services section of the cluster overview page to access the NiFi UI.

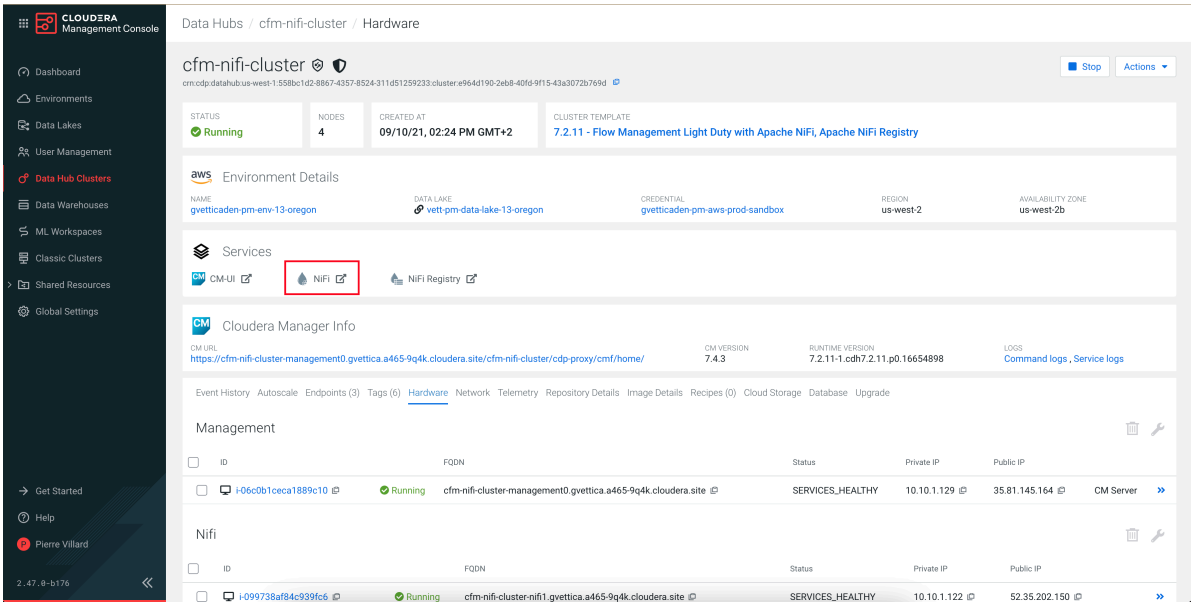

You will be logged into NiFi automatically with your CDP credentials.

2. Create the data flow by adding the `ListCDPObjectStore` processors onto the canvas.
   a) Drag and drop the processor icon into the canvas.
      In the dialog box you can choose which processor you want to add.
   b) Select the processor of your choice from the list.
   c) Click Add or double-click the required processor type to add it to the canvas.
3. Add the `FetchCDPObjectStore` processor.
4. Add any processors you require to transform your data.
5. Add the `PutCDPObjectStore` processor to write data to the target bucket.
6. Add the `DeleteCDPObjectStore` processor to write data to the target bucket.
7. Connect the processors to create the data flow by clicking the connection icon in the first processor, and dragging and dropping it on the second processor.

   A Create Connection dialog appears with two tabs: Details and Settings. You can configure the connection's name, flowfile expiration time period, thresholds for back pressure, load balance strategy and prioritization.

Results

This is an example data flow with the `ListCDPObjectStore`, `FetchCDPObjectStore`, `PutCDPObjectStore`, and `DeleteCDPObjectStore` processors:

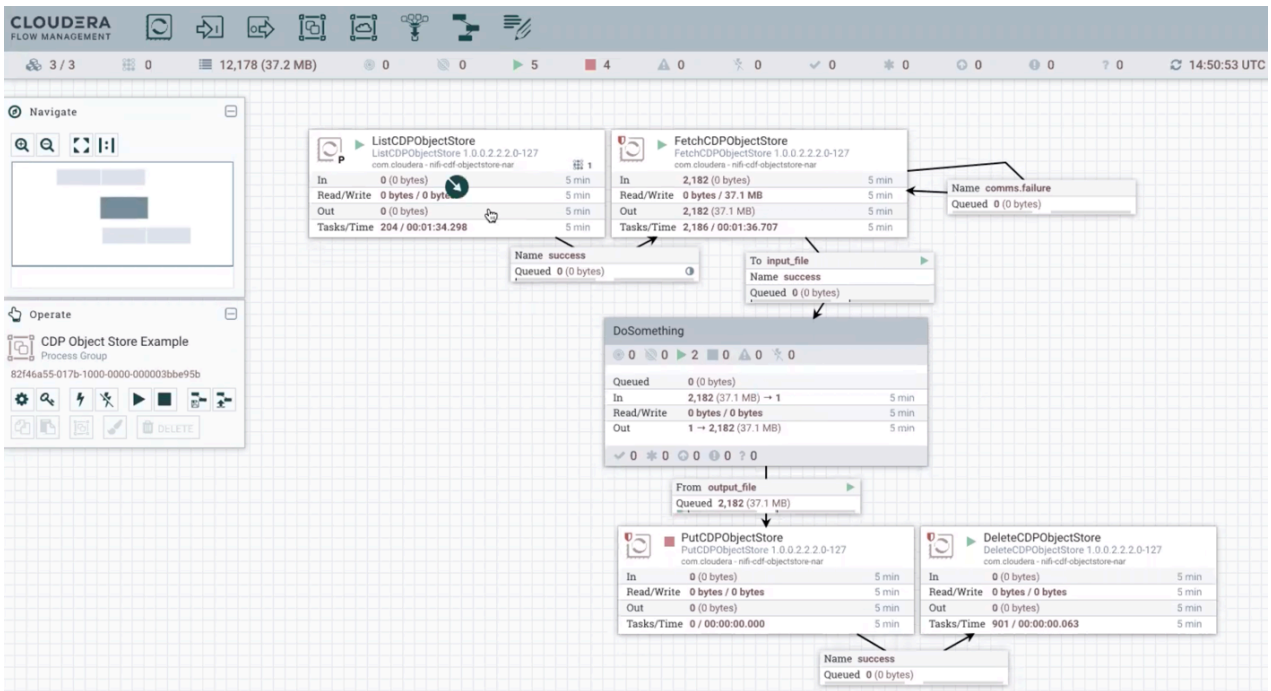

What to do next

Configure each object store processor in your data flow.

Configure each object store processor

Learn how you can configure the object store processors for your ingest data flow.

About this task

Configure the processor according to the behavior you expect in your data flow. Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured. Property values can be parameterized.

Procedure

1. Launch the Configure Processor window, by right clicking the `ListCDPObjectStore` processor and selecting Configure.
   This gives you a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the `ListCDPObjectStore` processor properties and click Apply.

Figure 1: ListCDPObjectStore processor properties

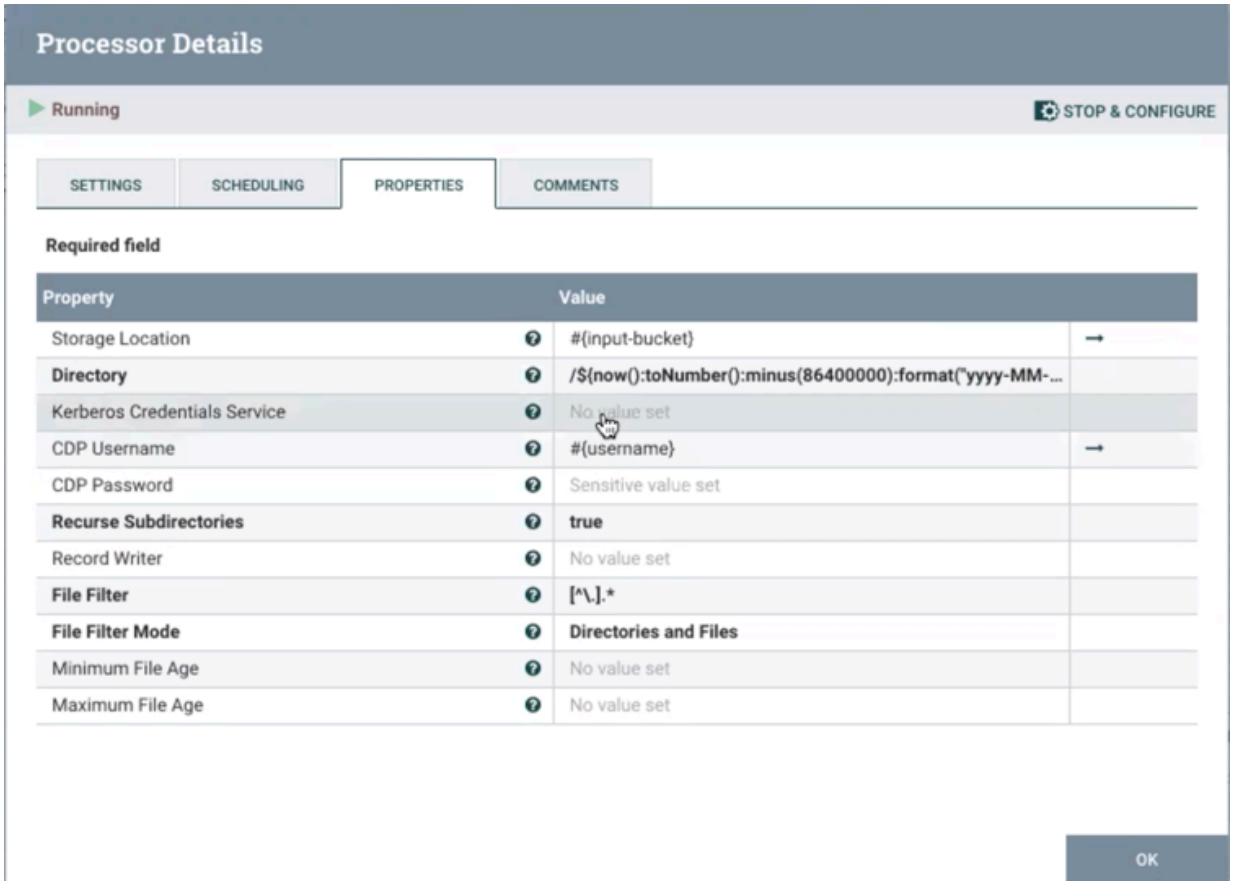

Table 1: ListCDPObjectStore processor properties

Property	Description	Example value for ingest data flow
Storage Location	The input bucket that contains the data that you want to ingest. You can create a parameter for the input bucket. **Important:** If you do not set the Storage Location property, the processor will set the storage location to the data lake.	s3a://my-input-bucket-nifi
Directory	Path to the data files in the source bucket.	/$(now():toNumber():minus(86400000):format("yyy-MM-dd", "GMT")
CDP Username	Your CDP account user name. You can create a parameter for the CDP account credentials.	srv_nifi-logs-ingest

Property	Description	Example value for ingest data flow
CDP Password	Your CDP account password. You can create a parameter for the CDP account credentials.	
Recurse Subdirectories	Set to true if you want to include subdirectories in the data ingest operation.	true

3. Configure the `FetchCDPObjectStore` processor properties and click Apply.

Figure 2: FetchCDPObjectStore processor properties

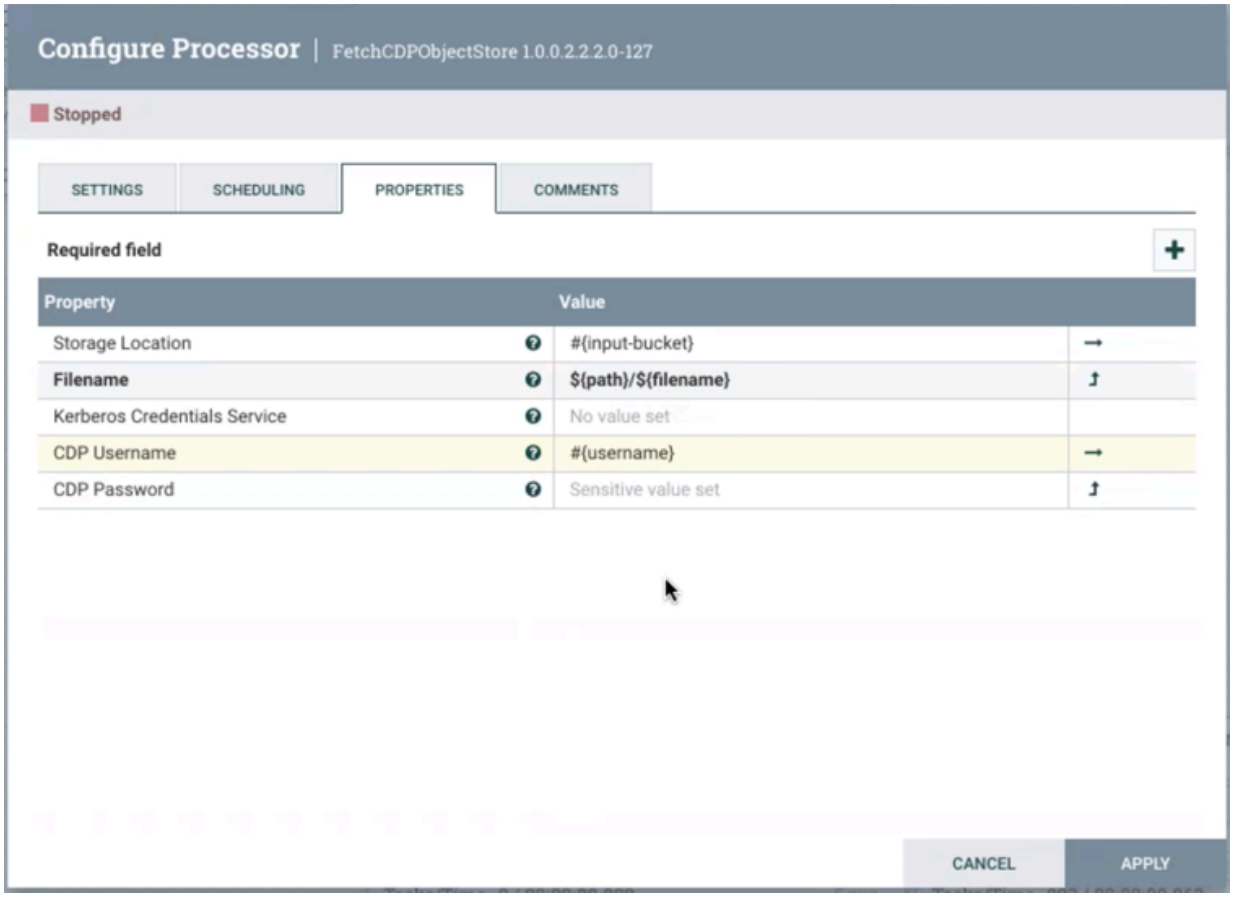

Table 2: FetchCDPObjectStore processor properties

Property	Description	Example value for ingest data flow
Storage Location	The input bucket that contains the data that you want to ingest. You can create a parameter for the input bucket and provide your workload account credentials to access the bucket. **Important:** If you do not set the Storage Location property, the processor will set the storage location to the data lake.	s3a://my-input-bucket-nifi

Property	Description	Example value for ingest data flow
Filename	The fully-qualified filename of the file to fetch from the source bucket. You do not need to set this property because the default value will use the attributes of the flow files created by the `ListCDPObjectStore` processor.	$(path)/$(filename)
CDP Username	Your CDP account user name. You can create a parameter for the CDP account credentials.	srv_nifi-logs-ingest
CDP Password	Your CDP account password. You can create a parameter for the CDP account credentials.	

4. Configure the `PutCDPObjectStore` processor properties and click Apply.

Figure 3: PutCDPObjectStore processor properties

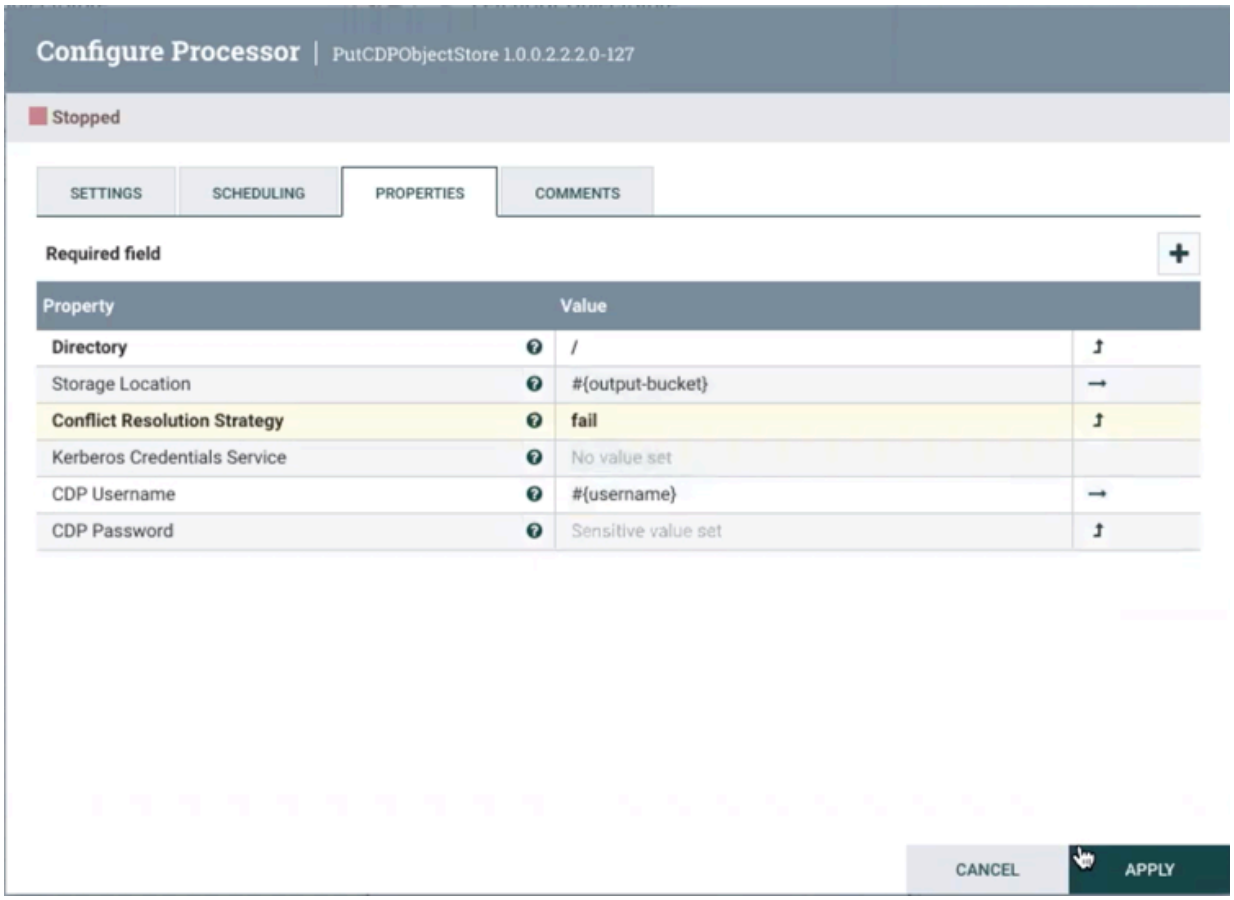

Table 3: PutCDPObjectStore processor properties

Property	Description	Example value for ingest data flow
Directory	The directory to which files should be written.	/
Storage Location	The target bucket. You can create a parameter for the target bucket.	s3a://my-output-bucket-nifi

Property	Description	Example value for ingest data flow
Conflict Resolution Strategy	Indicates what should happen when a file with the same name already exists in the target directory.	You can enter one of the following values: • fail • ignore • replace
CDP Username	Your CDP account user name. You can create a parameter for the CDP account credentials.	srv_nifi-logs-ingest
CDP Password	Your CDP account password. You can create a parameter for the CDP account credentials.	

5. Configure the `DeleteCDPObjectStore` processor properties and click Apply.

Figure 4: DeleteCDPObjectStore processor properties

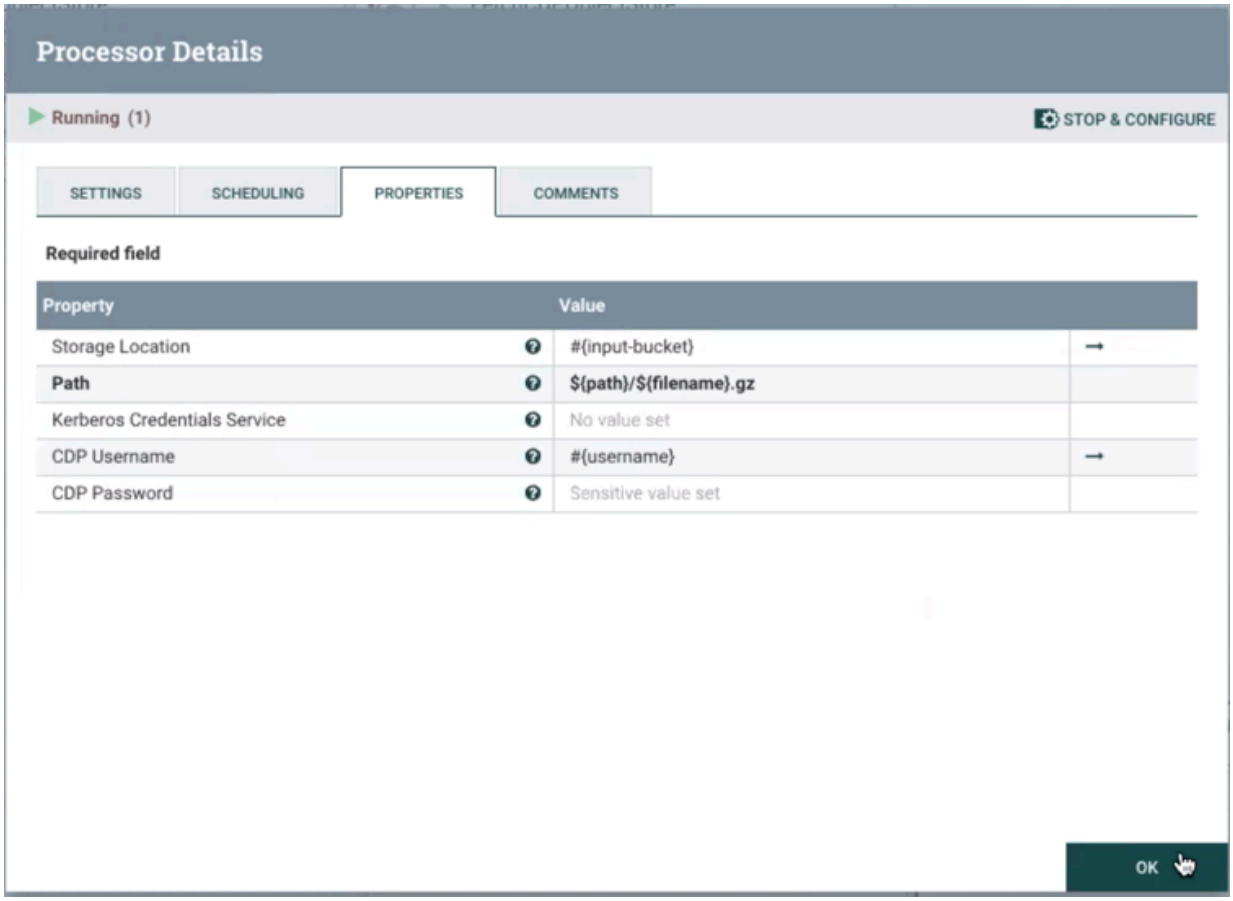

Table 4: DeleteCDPObjectStore processor properties

Property	Description	Example value for ingest data flow
Storage Location	The source bucket that contains the data that you have ingested and now want to delete. You can create a parameter for the input bucket. **Important:** If you do not set the Storage Location property, the processor will set the storage location to the data lake.	s3a://my-input-bucket-nifi
Path	Path to the data files in the source bucket that you want to delete.	$(path)/$(filename).gz
CDP Username	Your CDP account user name. You can create a parameter for the CDP account credentials.	srv_nifi-logs-ingest
CDP Password	Your CDP account password. You can create a parameter for the CDP account credentials.	

What to do next

Create policies in Ranger to enable the CDP user to access the source and target buckets.

Set permissions in Ranger

Create custom Ranger policies to enable the CDP user to read and write to the source and target buckets.

About this task

From your RAZ-enabled environment, access the Ranger service for your cloud provider. Then create a policy give the CDP user read and write access to the source bucket. Create another policy to give the CDP user read and write access to the target bucket.

Procedure

1. From your Data Hub cluster, select Ranger from the list of services and log into Ranger.

   The Ranger **Service Manager** page displays.
2. Select your cluster from your Cloud provider service folder.

   The following image shows an Amazon S3 service folder with a Data Hub cluster.

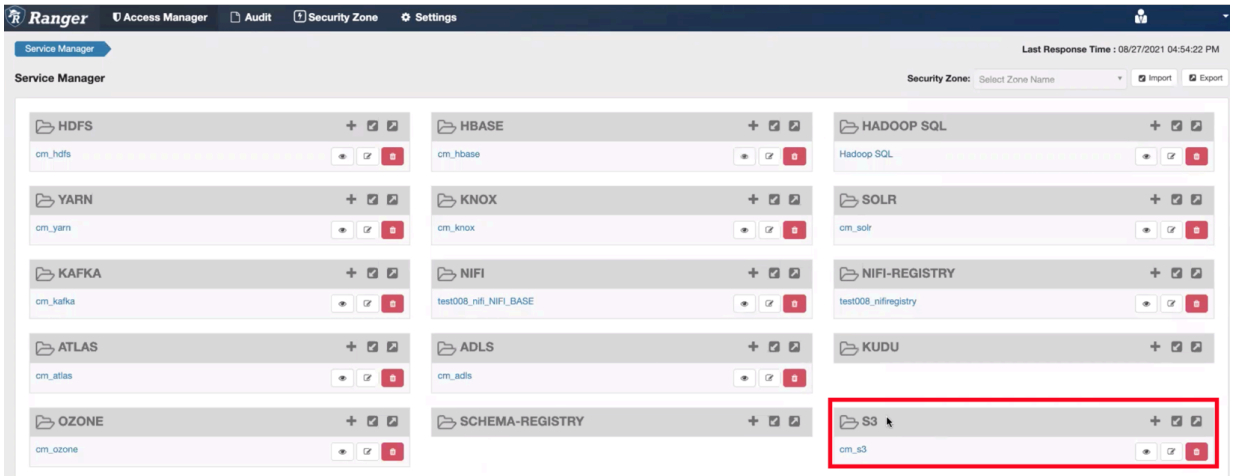

   The **List of Policies** page appears.
3. Click **Add New Policy**.

   The **Create Policy** page appears.

4. Add the following details to allow the user to access the source bucket:
   a) Enter a unique name for the policy. For example, Logs input.
   b) Specify your source bucket name. For example, s3a://my-input-bucket-nifi.
   c) In the Path field, specify the path to a specific directory or file. Or, to indicate any path, enter /.
   d) In the Allow Condition section, specify the CDP user name in the Select User field. For example, srv_nifi-log s-ingest.
   e) In the Permissions fields, enter read and write.
   f) Click Add to save the policy.

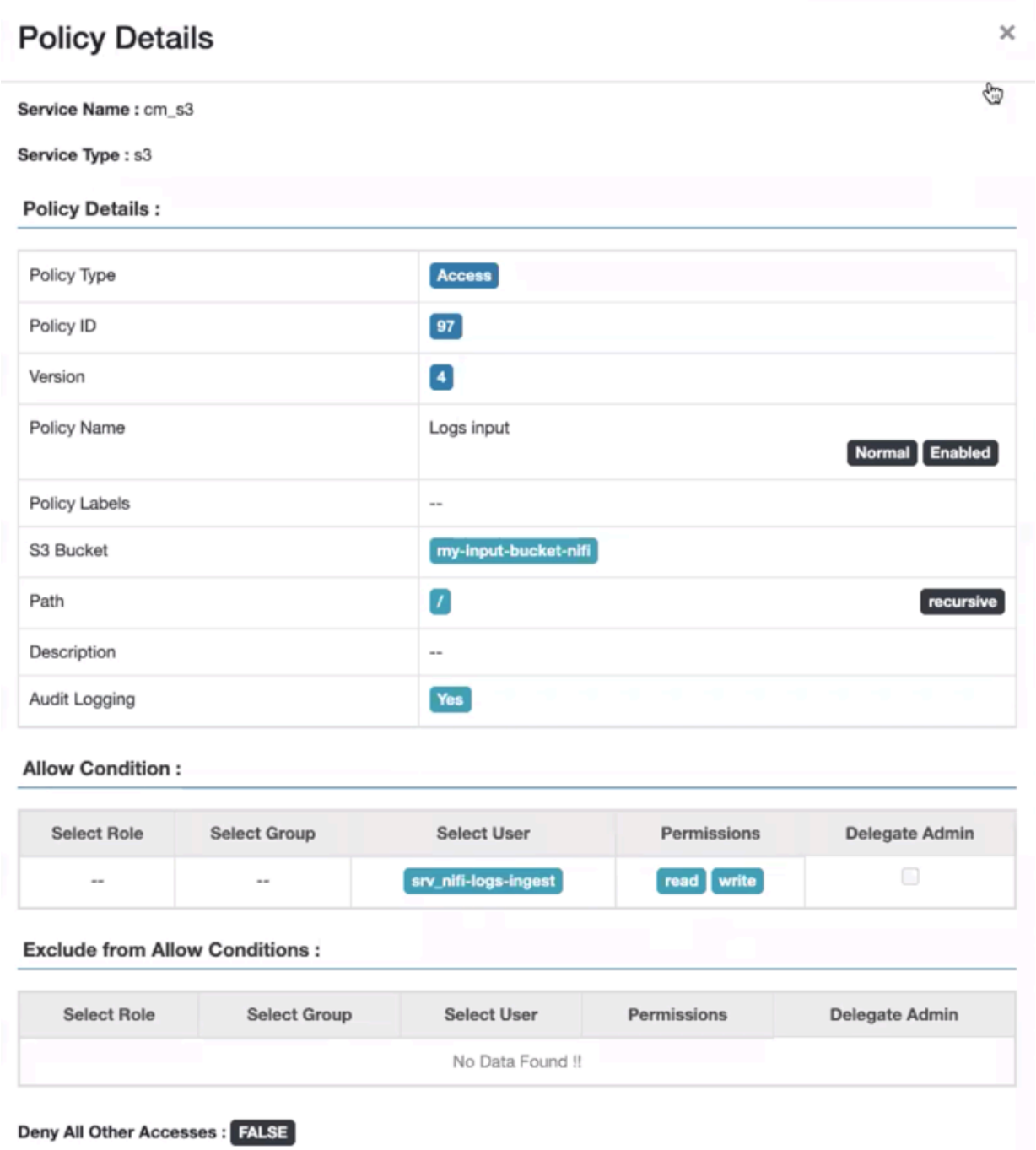

5. Add the following details to allow the user to access the target bucket:
   a) Enter a unique name for the policy. For example, Logs output.
   b) Specify your target bucket name. For example, s3a://my-output-bucket-nifi.
   c) In the Path field, specify the path to a specific directory or file. Or, to indicate any path, enter /.
   d) In the Allow Condition section, specify the CDP user name in the Select User field. For example, srv_nifi-log s-ingest.
   e) In the Permissions fields, enter read and write.
   f) Click Add to save the policy.

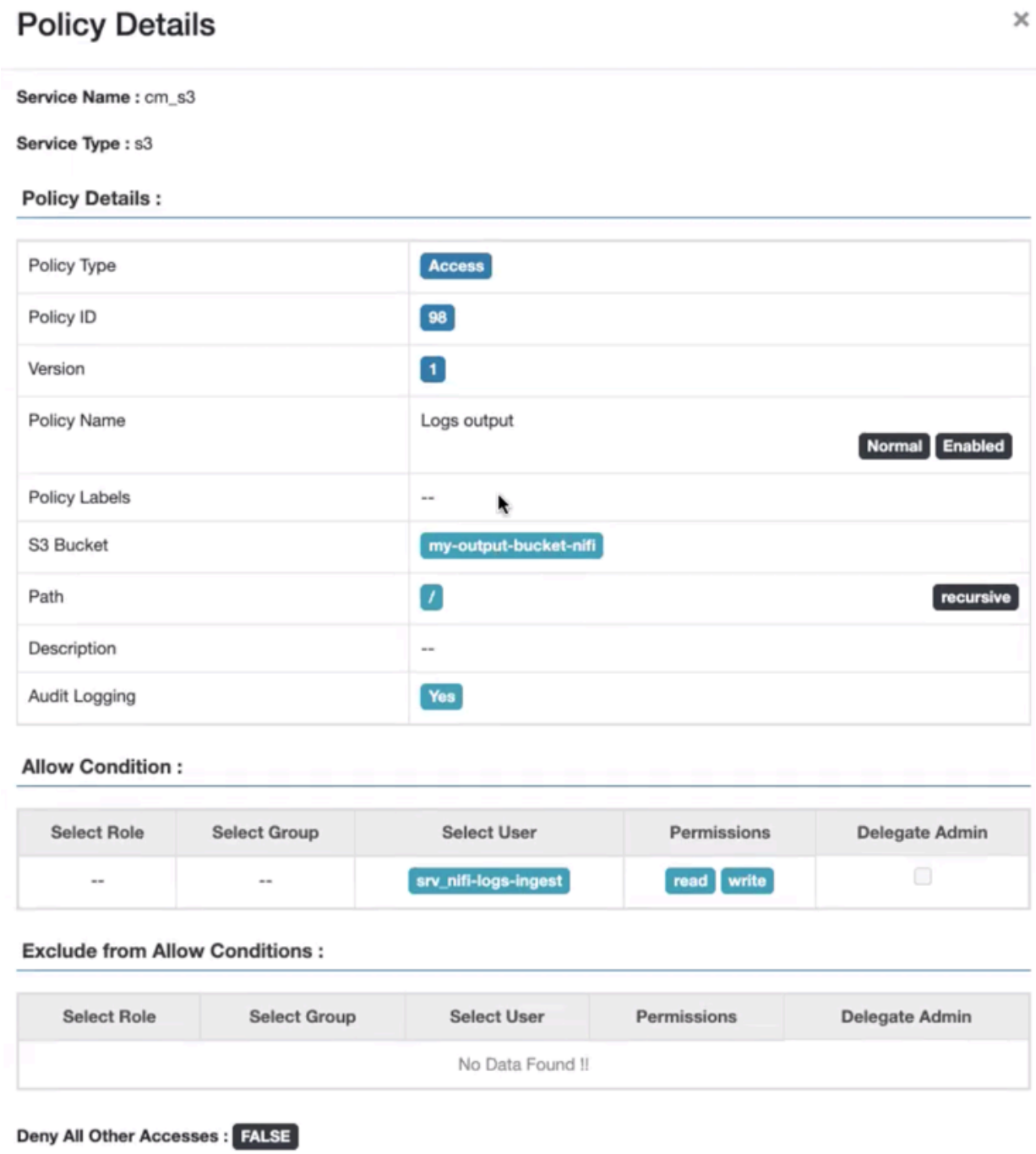

Results

When you start the data flow, the processors using the CDP user credentials can list and fetch from the source bucket and put and delete in the target bucket.

What to do next

Start the data flow.

Start the data flow

When your flow is ready, you can begin ingesting data into your object store. Learn how to start your object store ingest data flow.

Procedure

1. To initiate your data flow, select all the data flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternatively, right-click a single component and choose Start from the context menu. Data should be read from the input bucket and it should be written to the defined folder of your object store.

Results

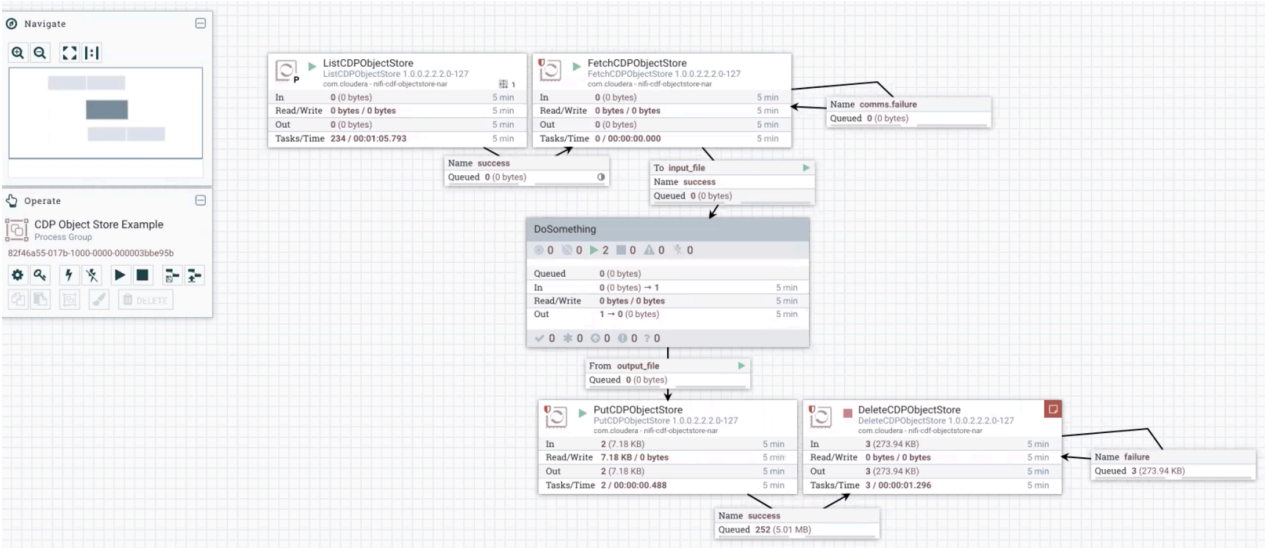

What to do next

It is useful to check that data is running through the flow you have created.

Verify data flow operation

Learn how you can verify the operation of your object store ingest data flow.

About this task

There are a number of ways you can check that data is running through the flow you have built.

Procedure

1. You can verify that NiFi processors are not producing errors.

2. You can look at the processors in the UI, where you can see the amount of data that has gone through them. You can also right click on the processors, or on connections to view status history.
3. You can check that the data generated appears in your target bucket. To do this, return to the dedicated folder of your object store bucket, where you should see your files listed.

   You may have to refresh the page depending on your browser/settings.

Monitoring your data flow

Learn about the different monitoring options for your object store ingest data flow in CDP Public Cloud.

You can monitor your data flow for information about health, status, and details about the operation of processors and connections. NiFi records and indexes data provenance information, so you can conduct troubleshooting in real time.

Data statistics are rolled up on a summary screen (the little table icon on the top right toolbar which lists all the processors). You can use the `MonitorActivity` processor to alert you, if for example you have not received any data in your flow for a specified amount of time.

If you are worried about data being queued up, you can check how much data is currently queued. Process groups also conveniently show the totals for any queues within them. This can often indicate if there is a bottleneck in your flow somewhere, and how far the data has got through that pipeline.

Another option to check that data has fully passed through your flow is to check out data provenance to see the full history of your data.

Next steps

Learn about the different options that you have after building a simple object store ingest data flow in CDP Public Cloud.

Moving data to the cloud is one of the cornerstones of any cloud migration. Cloud environments offer numerous deployment options and services. This example data flow provides you with a model to design more complex data flows for moving and processing data as part of cloud migration efforts.

You can build a combination of on-premise and public cloud data storage. You can use this solution as a path to migrate your entire data to the cloud over time—eventually transitioning to a fully cloud-native solution or to extend your existing on-premise storage infrastructure, for example for a disaster recovery scenario. Cloud storage can provide secure, durable, and extremely low-cost options for data archiving and long-term backup for on-premise datasets.

You can also use cloud services without storing your data in the cloud. In this case you would continue to use your legacy on-premise data storage infrastructure, and work with on-demand, cloud-based services for data transformation, processing and analytics with the best performance, reliability and cost efficiency for your needs. This way you can manage demand peaks, provide higher processing power, and sophisticated tools without the need to permanently invest in computer hardware.

Cloudera Flow Management 7.3.1

Using Parameter Context inheritance

Date published: 2019-06-26
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What is Parameter Context inheritance?.....4

Example for configuring Parameter Context inheritance.....4

Creating the basic Parameter Contexts.....4
Setting up Parameter Context inheritance.....6
Parameter overriding.....8

What is Parameter Context inheritance?

With Inherited Parameter Contexts, you can make granular updates to Parameter Contexts and have the changes propagate to all Contexts that inherit from them.

Parameter Context is a named set of parameters with applied access policies in Apache NiFi. Using Parameter Contexts is a powerful way to make flows more portable. Paired with Process Groups, Parameter Contexts allow different values to be supplied to the same basic flow in multiple NiFi instances, or even within the same overall NiFi flow in a single instance. However, as the number of parameters grows, Parameter Contexts can become more difficult to maintain and can result in a monolithic Parameter Context. If you collect all parameters in a root Parameter Context, you are parameterizing your Process Groups with more parameters than they need, as some of the parameters included in the Parameter Context are applicable only to certain Process Groups.

The goal is to avoid parameter duplication and yet still compose groups of parameters as needed. Starting with Apache NiFi 1.15.0, Parameter Contexts support inheritance, so you can add inherited Parameter Contexts to an existing Parameter Context. This feature brings more flexibility and maintainability to NiFi's already powerful Parameter Context framework. Using inheritance makes it easier to manage common sets of parameters used across many flows and adjust specific parameters for a given instance as needed. Creating Parameter Contexts through inheritance provides the opportunity to use a structure of smaller, logically grouped Parameter Contexts and design your flows with these smaller, composable Contexts. Once you set up the inheritance hierarchy, you can make granular updates and have the changes propagate to all other Contexts that inherit from it.

For more information about Process Groups, see *Configuring a Process Group in the Apache NiFi User Guide*.

For more information about Parameter Contexts, see *Parameter Contexts in the Apache NiFi User Guide*.

Related Information

Configuring a Process Group, Apache NiFi User Guide

Parameter Contexts, Apache NiFi User Guide

Example for configuring Parameter Context inheritance

In this example you can see how Parameter Context inheritance is configured for two existing Process Groups (Source ABC Kafka to S3 and Source DEF Kafka to GCS) to avoid repeating the Kafka brokers or the common site-specific parameters, and yet still composing groups of parameters as needed. The small, composable Parameter Contexts are logically grouped through inheritance.

Creating the basic Parameter Contexts

Follow these steps to set up your individual Parameter Contexts and add the relevant parameters to them.

Procedure

1. Create a Parameter Context for the common Kafka settings:
   a) Select Parameter Contexts from the top-right Global menu in the NiFi UI.

      The NiFi Parameter Contexts dialog opens.
   b) Click the (+) button in the top-right corner of the NiFi Parameter Contexts dialog to add a new Parameter Context.
   c) Add a name for your Parameter Context on the SETTINGS tab.

      In this example the name is Kafka.
   d) Switch to the PARAMETERS tab.
   e) Click the (+) button to add the parameters you need for configuring your data flow.
   f) Provide a name and a value in the Add Parameter dialog. You can also add a description of the parameter, if you want to.

      In this example, the following parameters are added:

      - Name: Kafka Brokers / Value: localhost:9092
      - Name: Kafka Group ID / Value: MyGroup
   g) Click APPLY to save the Parameter Context.
2. Create the remaining Parameter Contexts by repeating sub-steps a to g of Step 1. Use the following names and values:

Parameter Context	Parameter Name	Parameter Value
GCP Cloud Storage	GCP Project ID	my-project
	GCP Bucket	my-bucket
S3	S3 Bucket	my-bucket
	S3 Region	us-west-2
Site Properties	Site Identifier	1234
	Site Data Manager	MyDM
Source ABC	Kafka Topic	source-abc-topic
Source DEF	Kafka Topic	source-def-topic

Results

You are ready with the individual Parameter Contexts that you will use to create your inheritance hierarchy. Your Parameter Contexts should look like this:

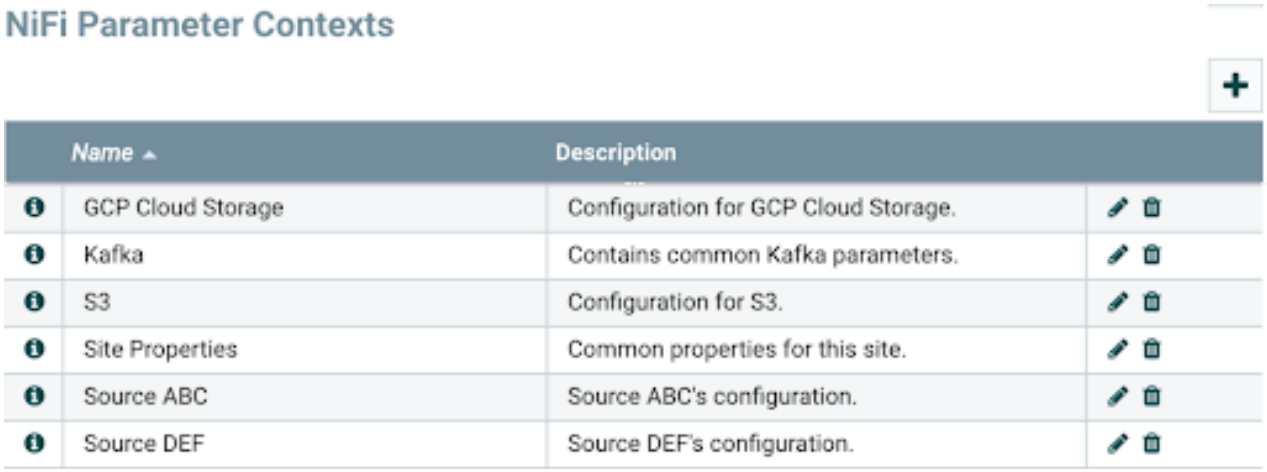

Setting up Parameter Context inheritance

Follow these steps to configure inherited Parameter Contexts.

About this task

Now that you have the individual Parameter Contexts set up, you can begin grouping them together, and create higher-level Parameter Contexts to build out your inheritance hierarchy and use them with the two Process Groups (Source ABC Kafka to S3 and Source DEF Kafka to GCS). The goal is to have one Parameter Context for each Process Group, composed of smaller units.

In this example, site properties and Kafka parameters are needed for both Process Groups. Additionally, each Process Group will have a source-related and a destination-related Parameter Context.

Before you begin

You have created two Process Groups in Apache NiFi:

- Source ABC Kafka to S3 that needs parameters for your Kafka brokers, Source ABC's topic, and for the S3 bucket
- Source DEF Kafka to GCS that needs parameters for the same Kafka brokers, Source DEF's topic, and for the GCP bucket
- In both Process Groups, you would like to use some common parameters tagging flowfiles with site-specific information

Procedure

1. Design the inheritance logic.

   In this example, the following structure is used:

   - Kafka and Site - inherits from Kafka and Site Properties
   - ABC Kafka and Site - inherits from Kafka and Site and Source ABC
   - DEF Kafka and Site - inherits from Kafka and Site and Source DEF
   - ABC Kafka to S3 - inherits from ABC Kafka and Site and S3
   - DEF Kafka to GCS - inherits from DEF Kafka and Site and GCP Cloud Storage

   Based on the above structure, the hierarchy for the two top-level Parameter Contexts looks like this:

   ABC Kafka to S3 Parameter Context

   - ABC Kafka and Site Parameter Context
     - Kafka and Site Parameter Context
       - Kafka Parameter Context
         - Kafka Brokers parameter with value localhost:9092
         - Kafka Group ID parameter with value MyGroup
       - Site Properties Parameter Context
         - Site Identifier parameter with value 1234
         - Site Data Manager parameter with value MyDM
     - Source ABC Parameter Context
       - Kafka Topic parameter with value source-abc-topic
   - S3 Parameter Context
     - S3 Bucket parameter with value my-bucket
     - S3 Region parameter with value us-west-2

   DEF Kafka to GCS Parameter Context

   - DEF Kafka and Site Parameter Context
     - Kafka and Site Parameter Context
       - Kafka Parameter Context
         - Kafka Brokers parameter with value localhost:9092
         - Kafka Group ID parameter with value MyGroup
       - Site Properties Parameter Context
         - Site Identifier parameter with value 1234
         - Site Data Manager parameter with value MyDM
     - Source DEF Parameter Context
       - Kafka Topic parameter with value source-def-topic
   - GCP Cloud Storage Parameter Context
     - GCP Project ID parameter with value my-project
     - GCP Bucket parameter with value my-bucket

2. Configure Parameter Context inheritance in NiFi for ABC Kafka to S3.
   a) Create a new Parameter Context named Kafka and Site and go to the INHERITANCE tab.

      You can see all of the individual Parameter Contexts.
   b) Drag Kafka and Site Properties to the right under Selected Parameter Context.
   c) Click APPLY, and then edit the Kafka and Site Parameter Context to see the new parameters.

      All four parameters from the two Parameter Contexts are now available in the view, and each of them has an (arrow) icon. You can click one of these to navigate to the Parameter Context in which the parameter is defined and edit its value.

      Note:

      You can drag/drop from the Available Parameter Contexts to the required inherited order in the Selected Parameter Context column.

      When you remove a Parameter Context from the Selected Parameter Context list, it returns to the Available Parameter Contexts that is sorted alphabetically.
   d) Repeat steps 1-3 to create ABC Kafka and Site Parameter Context inheriting all parameters from the Kafka and Site and the Source ABC Parameter Contexts.
   e) Finally, repeat steps 1-3 to create ABC Kafka to S3 Parameter Context inheriting all parameters from the ABC Kafka and Site and the S3 Parameter Contexts.

   This is how ABC Kafka to S3 looks like after the full setup:

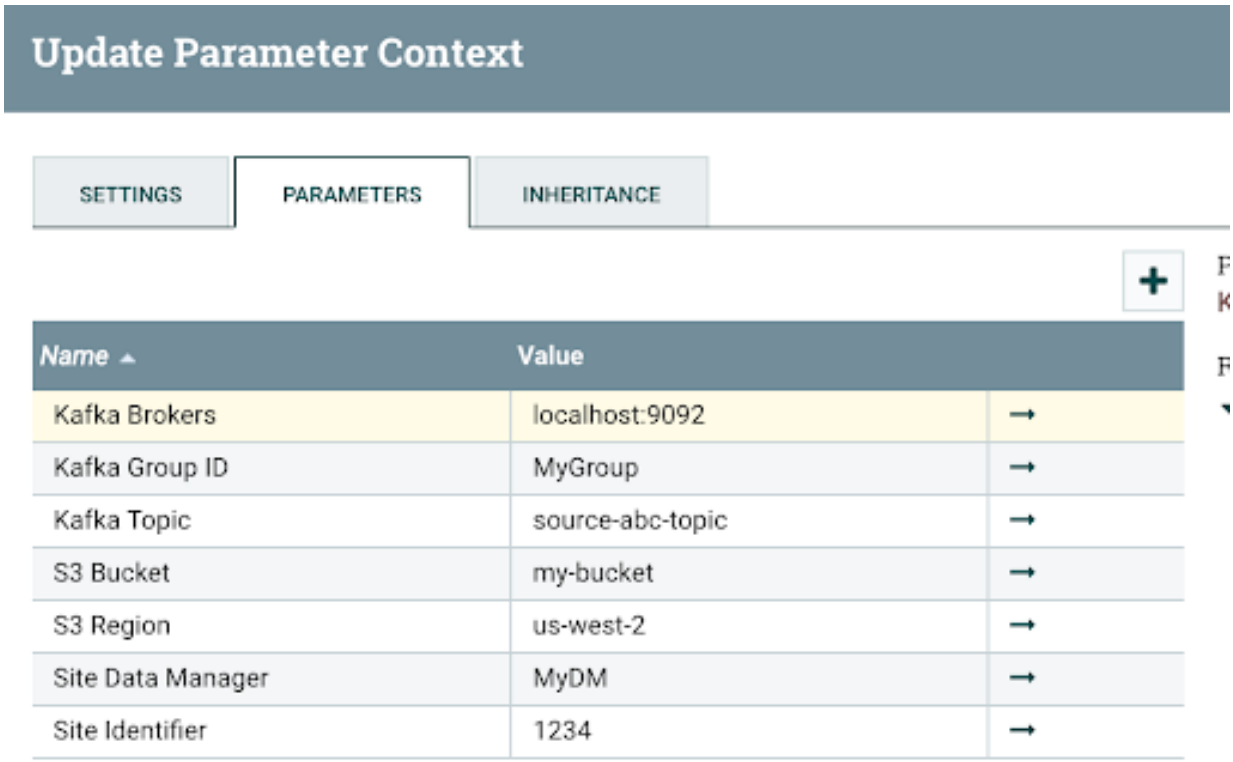

3. Configure Parameter Context inheritance for DEF Kafka to GCS. Follow the steps you completed when creating the ABC Kafka to S3 Parameter Context.

Parameter overriding

When a Parameter Context inherits parameters from other Contexts, you can still manually override parameters with a different value, if needed.

In this example, another Process Group is added that uses the same parameters as ABC Kafka to S3, but with a Kafka Group ID of MyOtherGroup.

To configure it, you can override the original parameter value by creating a new Parameter Context that inherits from ABC Kafka to S3 and adding a Kafka Group ID parameter with the MyOtherGroup value directly to that new Context:

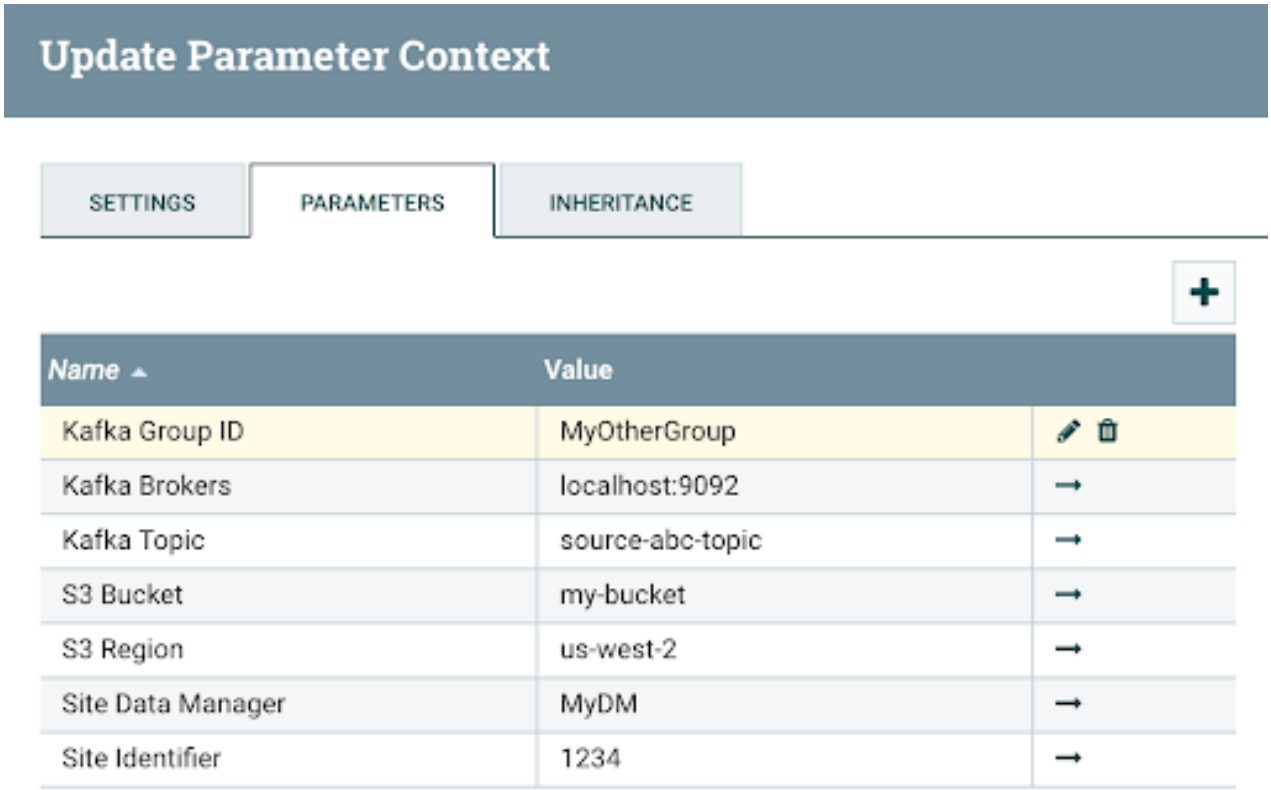

In this case the Kafka Group ID with its value MyOtherGroup appears at the top of the parameter list, and not where it should be in alphabetical order. Direct parameters always appear at the top of the list so you can see them grouped together.

You can edit the manually set parameter directly from this Parameter Context. So in this case, you are editing the parameter in the new Context and not the inherited parameter.

Parameter overriding order

- Direct parameters always take precedence.
- Parameters inside directly inherited Parameter Contexts take precedence, from top to bottom in the Selected Parameter Context column of the INHERITANCE tab.
- This recursively repeats in a depth-first manner for as many layers of inherited Parameter Contexts exist.

Cloudera Flow Management 7.3.1

Using Parameter Providers

Date published: 2019-06-26
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What are Parameter Providers? **4**

Example for using Parameter Providers **4**

Creating and configuring a Parameter Provider 4
Fetching Parameters 5
Creating a Parameter Context from a Parameter Group 6
Updating Parameter sensitivity 7
Updating Parameter Context when the external source has changed 8
Using Parameter Context inheritance to combine Parameters 8

What are Parameter Providers?

Parameter Providers is a new extension point that is added to the existing NiFi Parameter Context feature that can help to populate flow parameters on demand. Learn the features associated with Parameter Providers and how you can use them in your NiFi environment.

You can create Parameter Contexts from Parameters fetched from an external source. Using Parameter Providers allows automatic creation of Parameter Contexts from external sources, for example file-based Kubernetes secrets, environment variables, or HashiCorp Vault secrets engines.

Parameter Providers:

- Provide a feature in the Controller Settings that enable you to generate Parameter Contexts from external sources.
- Enable you to keep your provided Parameter Contexts up to date with the external source by running a Fetch Parameters operation.
- Provide an extension point for developing new custom Parameter Providers that can be deployed in a NiFi Archive (NAR).
- Provide a CLI command, nifi fetch-params, to fetch and apply parameters, allowing a scripted approach to keep Parameter Contexts up to date.

Parameter values are stored (encrypted, if sensitive) inside the flow. Parameter Providers are a mechanism that automate the creation of Parameter Contexts and facilitate keeping them updated. They do not replace the framework mechanism to resolve parameter values during Processor/Controller Service execution and they do not pull parameters directly from the external source at the time of usage in the flow.

Note: Parameter Providers do not have an automatic mechanism to refresh parameter values from the NiFi UI. Fetching and applying parameters is a potentially disruptive operation, since it can involve stopping and starting large portions of the flow. This kind of activity could be scripted using the CLI command fetch-pa rams, but should be done with the potential for flow disruption in mind.

For more information about Parameter Contexts, see *Parameter Contexts in the Apache NiFi User Guide*.

Related Information

Parameter Contexts, Apache NiFi User Guide

Example for using Parameter Providers

In this example you can see how a Parameter Provider is created and configured, how it is used to fetch Parameters from an external source, and how a Parameter Context is created and updated using the Parameters fetched from the external source.

Creating and configuring a Parameter Provider

Learn how to create and configure Parameter Providers through the NiFi Controller Settings.

Procedure

1. Select Controller Settings from the top-right Global menu in the NiFi UI.

   The **NiFi Settings** dialog opens.
2. Go to the PARAMETER PROVIDERS tab.
3. Click **+** in the top-right corner of the NiFi Settings dialog to add a new Parameter Provider.

4. Select one of the listed types, in this example, FileParameterProvider, and click Add.

   The Parameter Provider is created.

   The FileParameterProvider created in this example lets you supply parameters in key-value files inside a Parameter Group directory.

5. Click to edit it.
6. Enter the absolute path of the Parameter Groups on your file system in the Parameter Group Directories property.
7. Set the Parameter Value Encoding property to "Plain text".

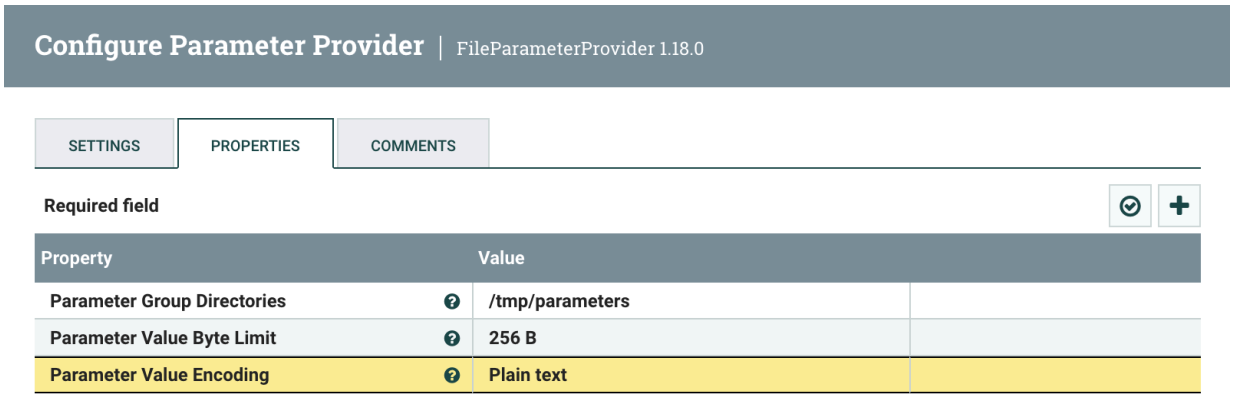

8. Click APPLY to save the Parameter Provider configuration.

   The Fetch Parameters icon becomes available if the path set for the Parameter Group Directories property is readable.

Results

The Parameter Provider, in this example, FileParameterProvider, is created and configured.

What to do next

You can now move on to fetching parameters. For details, see *Fetching parameters*.

Related Information

Fetching Parameters

Fetching Parameters

Learn how to fetch Parameters from an external source into a Parameter Group.

Before you begin

You must have created and configured a Parameter Provider. See Creating and configuring a Parameter Provider.

Procedure

1. If you do not have any files containing key-value pairs as parameters inside the directory set in the Parameter Group Directories property of FileParameterProvider, which in this example is the tmp/parameters directory, create some files and add some content.

```
$ print admin > /tmp/parameters/sys.admin.username && \
  print password > /tmp/parameters/sys.admin.password && \
```

```
print value > /tmp/parameters/sys.other
```

In this example, each file represents a Parameter in the parameters Parameter Group, and the contents of each file represents the parameter value.

2. Select Controller Settings from the top-right Global menu in the NiFi UI.

   The NiFi Settings dialog opens.
3. Go to the PARAMETER PROVIDERS tab.
4. Click the Fetch Parameters (down arrow) icon of FileParameterProvider.

   The Fetch Parameters dialog opens.

Results

A Parameter Group named parameters is listed. This is the name of the directory where the Parameter Groups are stored on your file system. This group can be used to create a Parameter Context. The fetched parameters sys.admin.username, sys.admin.password, and sys.other are displayed.

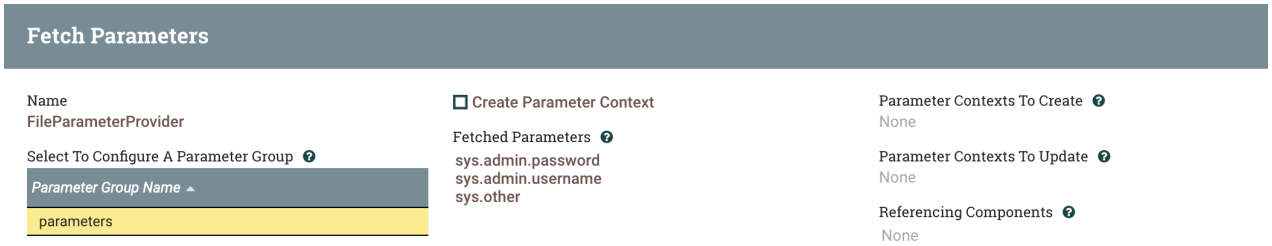

What to do next

You can now move on to *Creating a Parameter Context from a Parameter Group*.

Related Information

Creating a Parameter Context from a Parameter Group

Creating a Parameter Context from a Parameter Group

Learn how to create a Parameter Context from a Parameter Group that stores Parameters fetched from an external source using a Parameter Provider.

Before you begin

You must have fetched parameters from an external source. See Fetching Parameters.

Procedure

1. Select Controller Settings from the top-right Global menu in the NiFi UI.

   The NiFi Settings dialog opens.
2. Go to the PARAMETER PROVIDERS tab.
3. Click the Fetch Parameters (down arrow) icon of FileParameterProvider.

   The Fetch Parameters dialog opens.

4. Select Create Parameter Context.
   - A star appears next to the parameters Parameter Group. The star indicates that the Parameter Group has an associated Parameter Context.
   - Checkboxes appear next to the Parameter names.
   - The Parameter Context Name field appears.

     Enter a name for your Parameter Context. In this example, it is My Parameters.

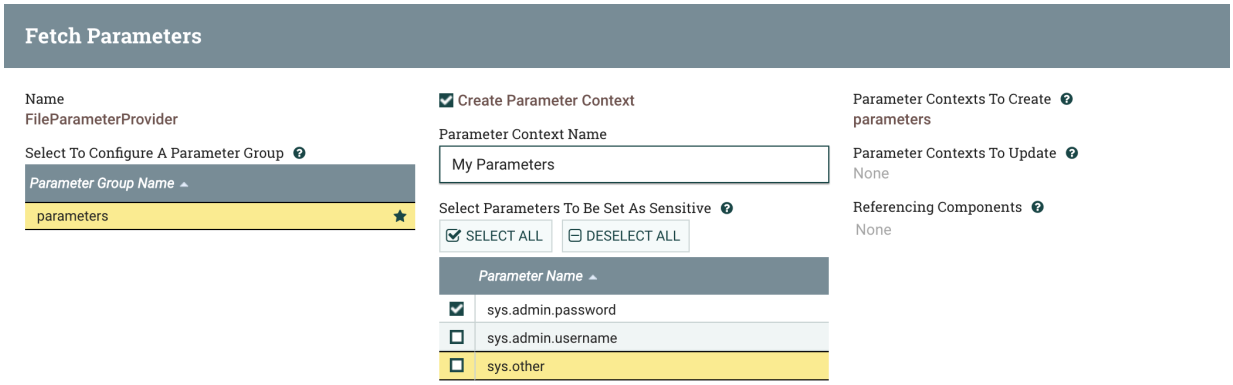

5. Select the Parameters that you want to set as sensitive.
   In this example, only the sys.admin.password Parameter needs to be set to sensitive.
6. Click APPLY to create the Parameter Context.
7. Click Close in the Fetch Parameters dialog.

Results

- The newly created Parameter Context appears under the SETTINGS tab of the Configure Parameter Provider dialog.
- The Parameters appear under the PARAMETERS tab of the Update Parameter Context dialog. Only the values of the non-sensitive Parameters can be read. The Parameters cannot be edited. The Parameter Provider can add, remove, or update Parameters in this Parameter Context.

What to do next

- You can view the new Parameter Context in the NiFi Parameter Contexts list by clicking its name.
- You can view the details of the Parameter Context in the Update Parameter Context dialog by clicking .
- You can return to the NiFi Settings by clicking the right arrow icon in the listing.

Related Information

What is Parameter Context inheritence?

Updating Parameter sensitivity

You may need to update the sensitivity setting of Parameters of a Parameter Contexts over time.

Before you begin

- You have created a Parameter Context from a Parameter Provider.
- If the Parameter Context, in this example, "My Parameters", is assigned to a group, and that group has a component that references the sys.other parameter, you cannot update its sensitivity. In order to do so, ensure that you remove any references to it.

You must have created a Parameter Context from a Parameter Provider.

Procedure

1. Select Controller Settings from the top-right Global menu in the NiFi UI.

   The NiFi Settings dialog opens.
2. Go to the PARAMETER PROVIDERS tab.
3. Click the Fetch Parameters (down arrow) icon of FileParameterProvider.

   The Fetch Parameters dialog opens.
4. Select the Parameter whose sensitivity you want to change.
5. Click APPLY to update the Parameter Context.

Updating Parameter Context when the external source has changed

The Parameters that you fetched from an external source may change in that external source over time. In such cases you need to fetch the Parameters again using the Parameter Providers.

About this task

Parameters are not automatically synchronized with the external source. To update any linked Parameter Contexts, you can fetch the parameters again, and you need to specify the sensitivity of any new parameters.

Before you begin

There must be changes to your external source. To simulate the change in the example, delete the sys.other parameter, update the value of the sys.admin.username parameter, and add a new parameter new-parameter.

```
$ rm /tmp/parameters/sys.other && \
  print admin2 > /tmp/parameters/sys.admin.username && \
  print test > /tmp/parameters/new-parameter
```

Procedure

1. Select Controller Settings from the top-right Global menu in the NiFi UI.

   The NiFi Settings dialog opens.
2. Go to the PARAMETER PROVIDERS tab.
3. Click the Fetch Parameters (down arrow) icon of FileParameterProvider.

   The Fetch Parameters dialog opens. An asterisk appears next to newly fetched and changed Parameters. To view detailed information about the change in the Parameter, hover over the asterisk.
4. Set sensitivity for the newly fetched Parameters.
5. Click APPLY to update the Parameter Context.
6. Click Close in the Fetch Parameters dialog.

Results

The changes that took place in the parameters in the external source are now reflected in the Parameter Context. In this example, sys-other parameter is removed, new-parameter is added, and the value of sys.admin.username is changed. You can view the changes by clicking the Fetch Parameters (down arrow) icon.

Using Parameter Context inheritance to combine Parameters

You can use Parameter Context inheritance to combine Parameters from different Parameter Providers within one Parameter Context.

It is good practice to create separate Parameter Providers for different sources, and then compose the Parameter Contexts through inheritance. This allows sensitive parameters to originate from a secrets manager, like HashiCorp Vault, and non-sensitive parameters to originate from other sources like Environment Variables or the file system. In the following simple example, the EnvironmentVariableParameterProvider contains non-sensitive Parameters and the FileParameterProvider contains sensitive Parameters. Both Parameter Providers are added as inherited contexts to a new Parameter Context, “Parameters”. For details on inherited Parameter Contexts, see *What is Parameter Context inheritance?*.

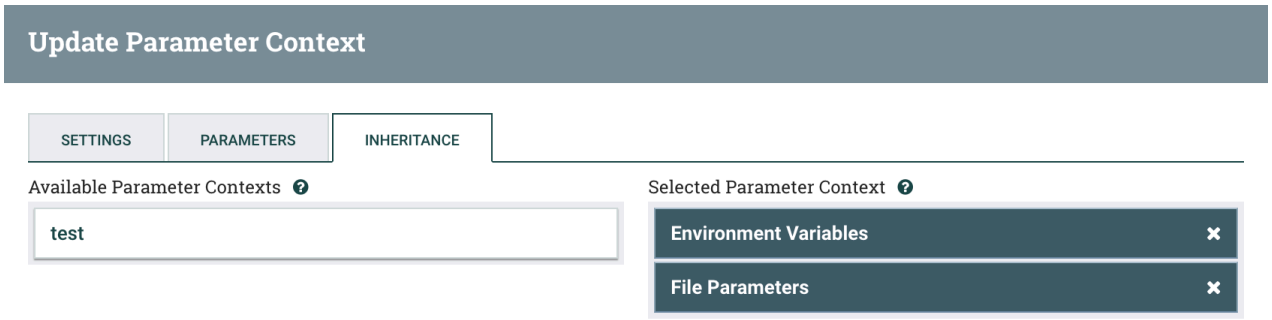

The new Parameter Context inherits Parameters from both sources:

JENV_FORCEJAVAHOME	true	→
JENV_FORCEJDKHOME	true	→
JENV_LOADED	1	→
JENV_SHELL	zsh	→
new-parameter	Sensitive value set	→
sys.admin.password	Sensitive value set	→
sys.admin.username	Sensitive value set	→

Related Information

What is Parameter Context inheritence?

Getting started with Apache NiFi Registry

Date published:
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Getting started with Apache NiFi Registry..... 4

Getting started with Apache NiFi Registry

For more information, see the Apache NiFi Registry Getting Started Guide.

Important:

The Apache NiFi Registry documentation is available both online and within the Registry NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi Registry version. If you are using an earlier NiFi Registryversion, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi Registry version directly from the NiFi Registry UI:

1. Launch the NiFi Registry UI.
2. From the Global Menu, select Help Documentation .

Related Information

Getting Started with Apache NiFi Registry Guide

Cloudera DataFlow for Data Hub 7.3.1

Scaling up or down a NiFi Cluster

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Scaling up or down a NiFi cluster.....4
Scaling up a NiFi cluster.....4
Scaling down a NiFi cluster.....5

Scaling up or down a NiFi cluster

You can scale up or down a NiFi cluster based on your requirement by adding or removing nodes respectively.

Scaling up a NiFi cluster

You have a NiFi cluster and you want to increase the throughput by adding new nodes to your NiFi cluster.

Before you begin

You have a NiFi flow which is running.

Procedure

1. Log into Cloudera Public Cloud UI.
2. Go to Cloudera Management Console Environments .
3. Select the datahub cluster where your NiFi flow is running.

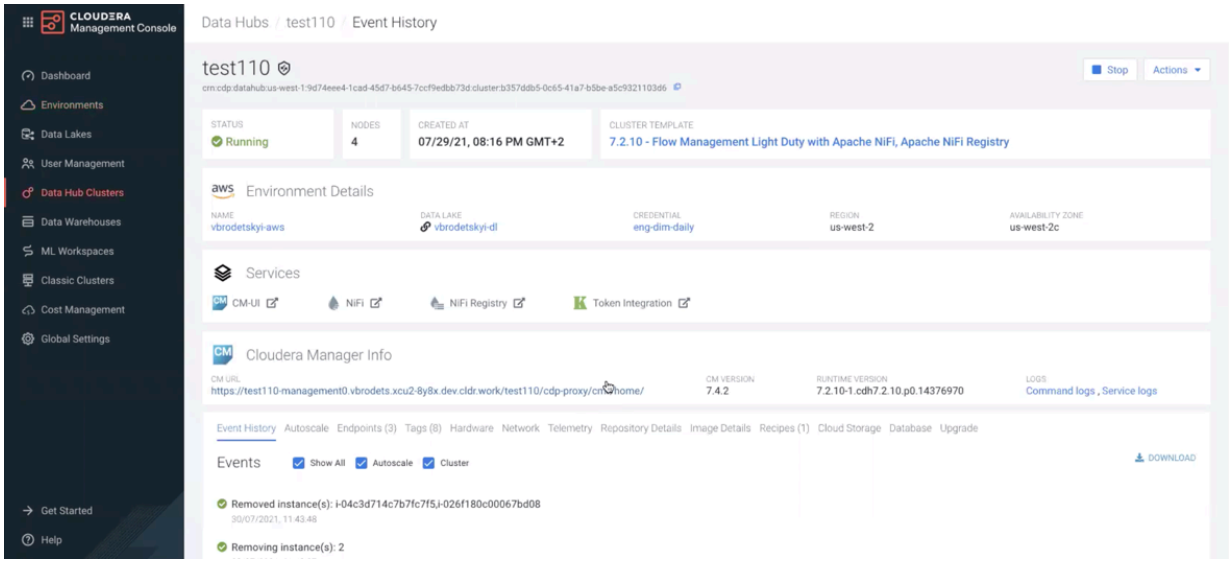

If you go to the NiFi UI by clicking NiFi, you can check the current number of NiFi nodes in your cluster, as shown in the following image:

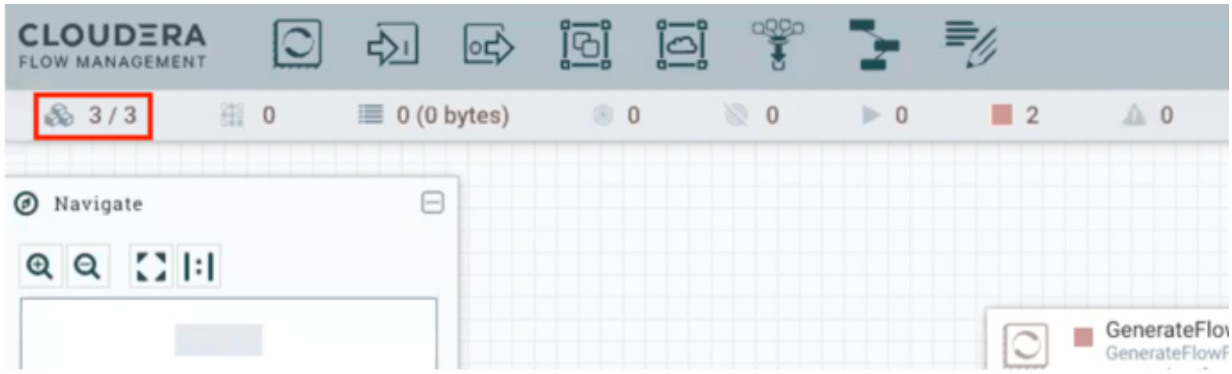

4. Click Actions, and select Resize.

   The Cluster Resize window appears.

5. Select your hostgroup from the dropdown, increase the number of nodes by clicking the + icon, and then click Resize.

Note: You need to select the hostgroup carefully. For light duty templates, the hostgroup is nifi_scaling. For heavy duty templates, the hostgroup is nifi.

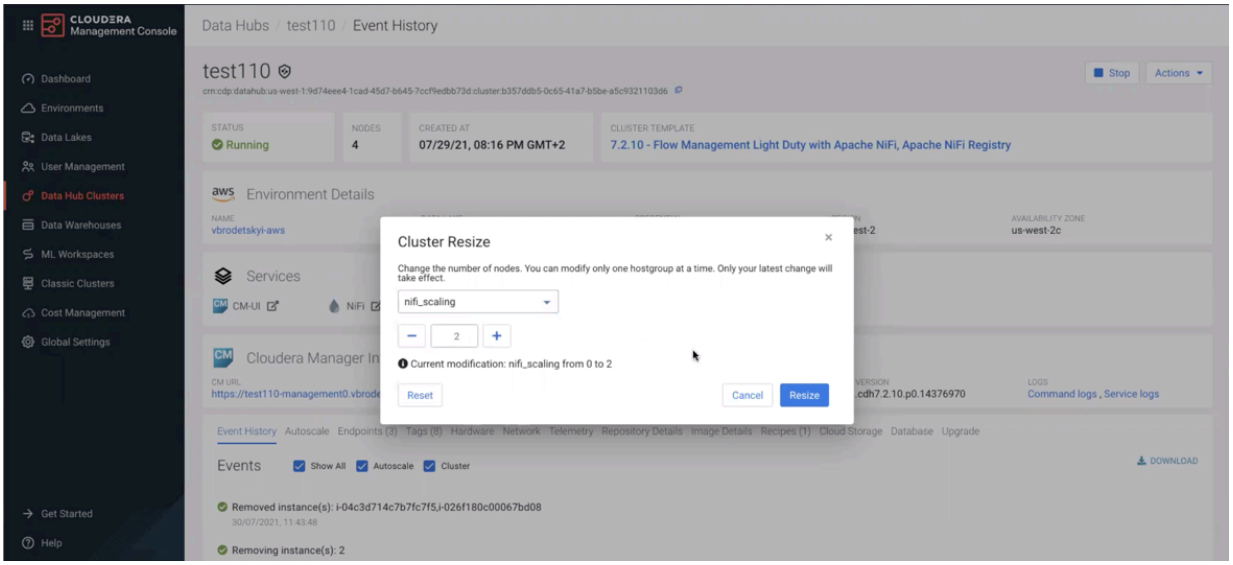

It takes some time for the change to take place. In the NiFi UI you can see when the upscale event is completed and the new nodes are added to the NiFi cluster.

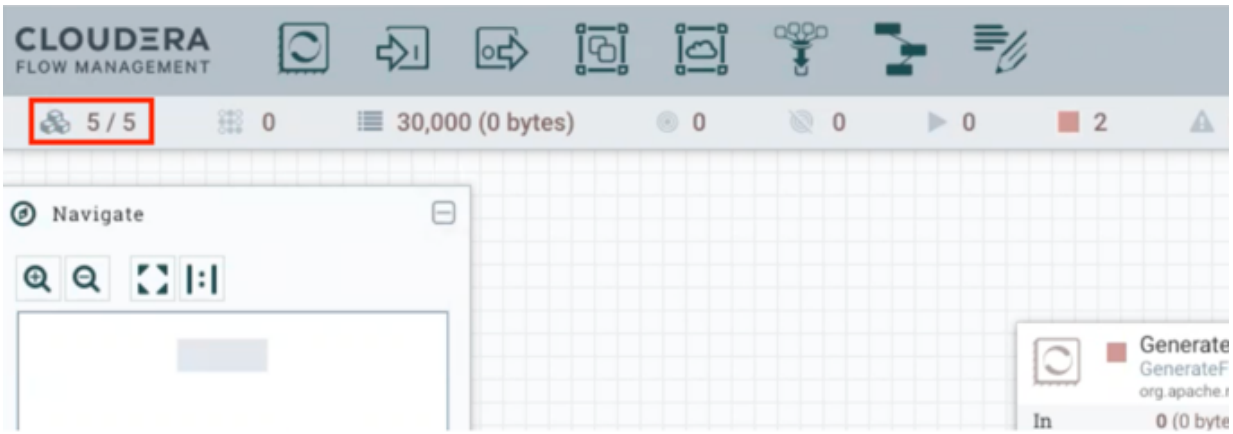

Results

You have now successfully scaled up your NiFi cluster.

Scaling down a NiFi cluster

You can also remove nodes from your NiFi cluster for cost optimization. For example, scale down the NiFi cluster when there is less data to be processed in order to save infrastructure costs.

Before you begin

You have a NiFi flow which is running.

Procedure

1. Log into Cloudera Public Cloud UI.
2. Go to Cloudera Management Console Environments .

3. Select the datahub cluster where your NiFi flow is running.

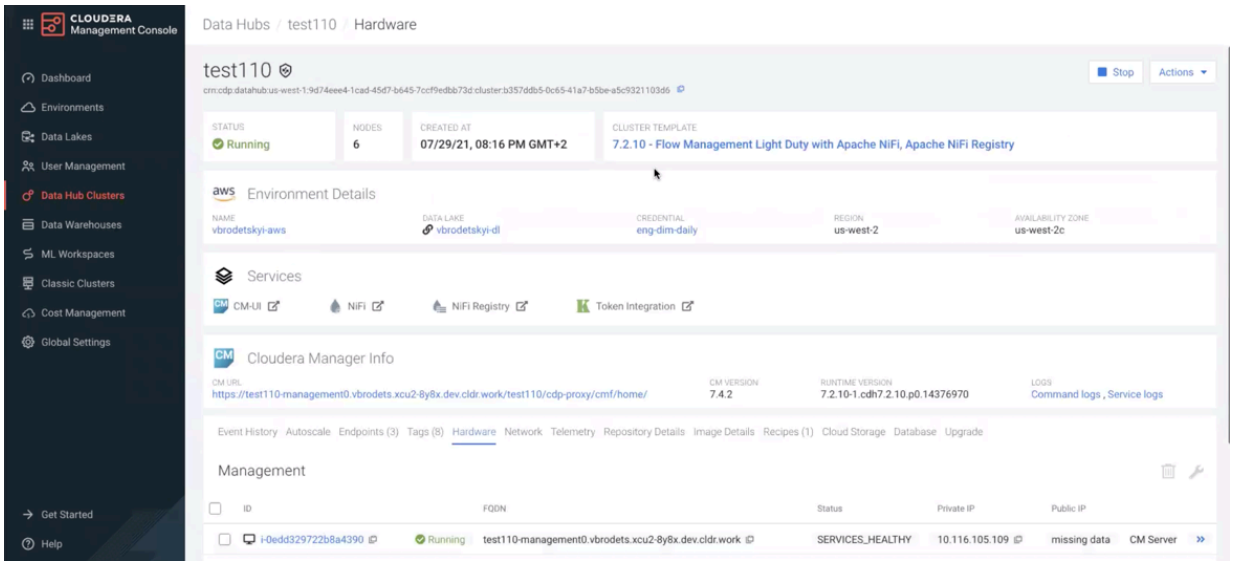

Before scaling down you can go to the NiFi UI by clicking NiFi to check the current number of NiFi nodes you have in your NiFi cluster, as shown in the following image:

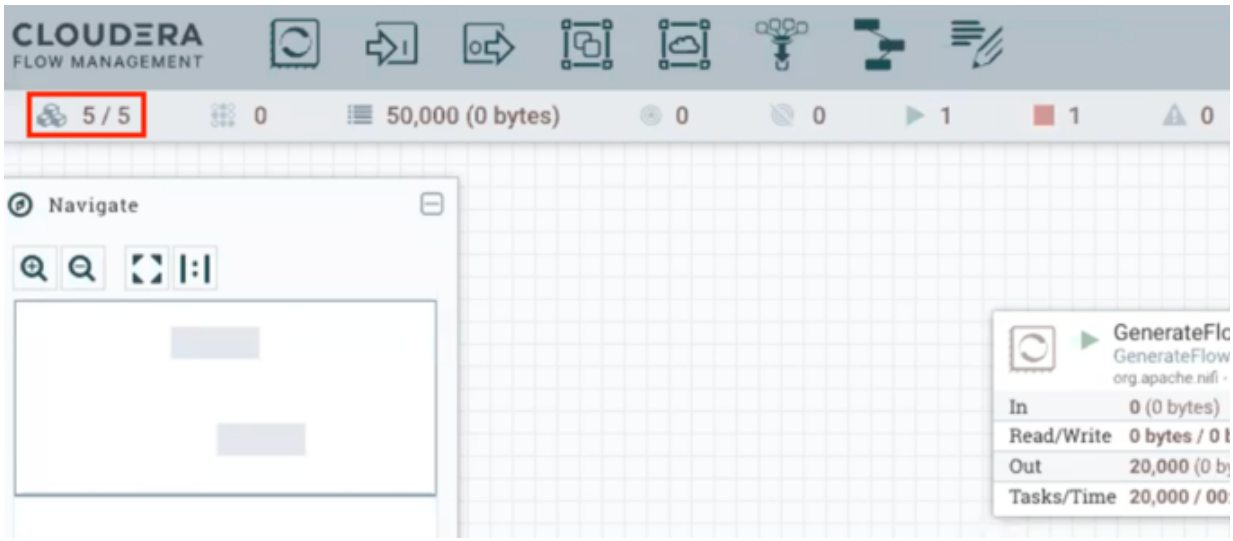

4. Click Actions, and select Resize.

   The Cluster Resize window appears.

5. Select your hostgroup from the dropdown, decrease the number of nodes by clicking the - icon, and then click Resize.

Note: You need to select the hostgroup carefully. For light duty templates, the hostgroup is nifi_scaling. For heavy duty templates, the hostgroup is nifi.

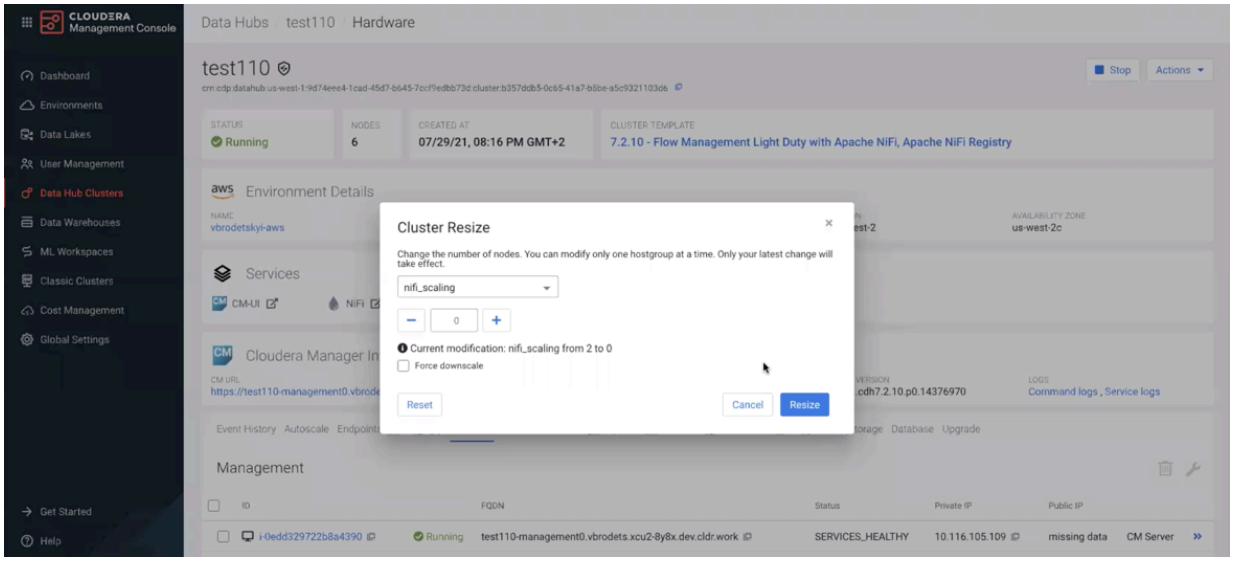

It takes some time for the change to take place. In the background, the following processes occur:

a. The nodes to be removed are disconnected from the NiFi cluster.
b. Processing of data on these nodes is stopped.
c. The data located on the nodes to be removed are moved to the remaining nodes of the cluster. This ensures no data loss during a scale down event.
d. The NiFi service is stopped on these nodes
e. The underlying virtual machines are stopped and deleted.

You can go to the NiFi UI to check that the nodes are successfully removed from the NiFi cluster:

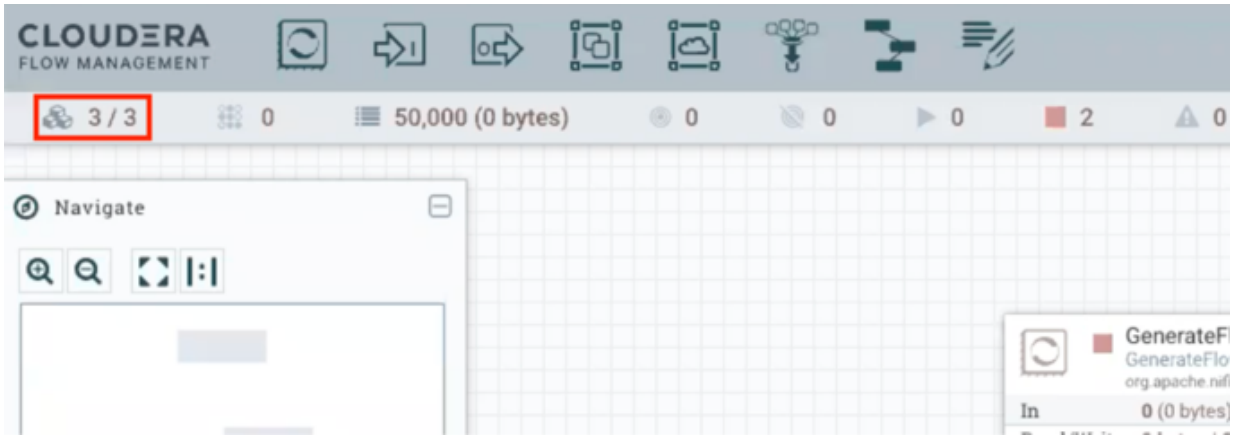

The scale down event is complete only after the nodes are no longer listed in the Hardware tab of the Data Hub Clusters page, as shown in the following image:

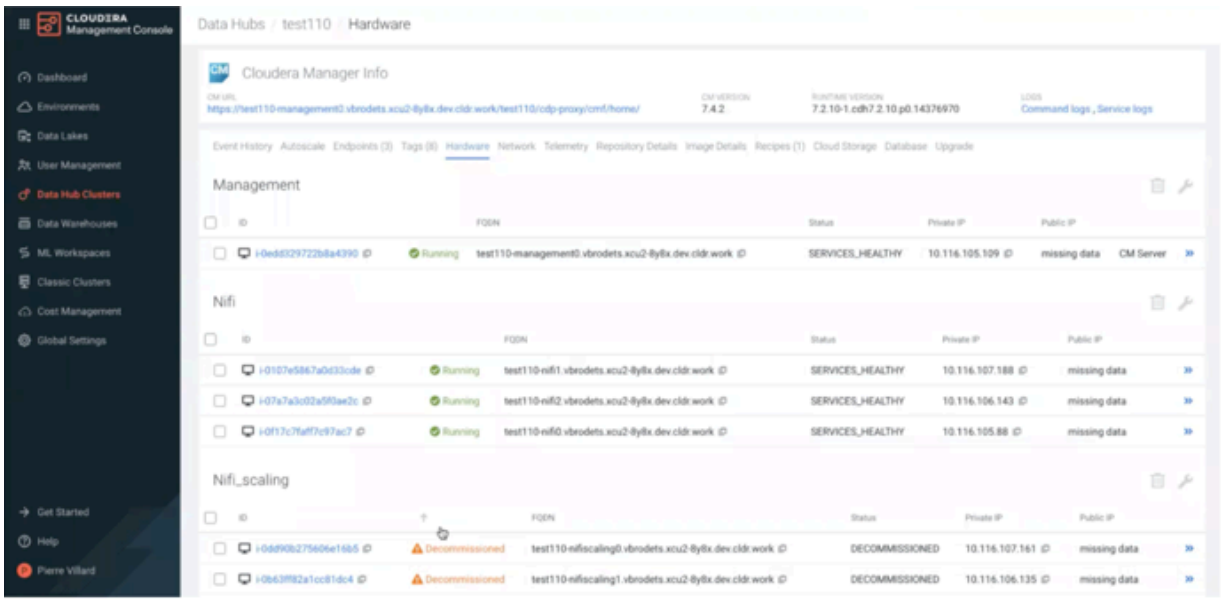

Results

You have now successfully scaled down your NiFi cluster.

Cloudera DataFlow for Data Hub 7.3.1

Securing Flow Management Clusters

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Security for Flow Management Clusters and Users in Cloudera Public Cloud 4

User Authorization 4

Authorization workflow 4
Assigning administrator level permissions 5
Assigning selective permissions to user 6
Assign the EnvironmentUser role 7
Add the user to predefined Ranger access policies 7
Create a custom access policy 9
Authorization example 14

Predefined Ranger access policies for Apache NiFi 14

Predefined Ranger access policies for Apache NiFi Registry 16

Security for Flow Management Clusters and Users in Cloudera Public Cloud

Cloudera is a secure and governed cloud service platform that offers a broad set of enterprise data cloud services with the key data analytics and artificial intelligence functionality that enterprises require.

Cloudera provides the following default security features for flow management users and clusters:

- Single-sign on (SSO) authentication with Apache Knox.
- Metadata management and governance capabilities with Apache Atlas.
- Flow versioning and management with Apache NiFi Registry.
- TLS encryption to secure communications over the network.
- Fine-grained authorization to do a specific action and/or operation with Apache Ranger.

For more information, see the identity management section in the *Cloudera Security Overview*.

This document explains how to authorize a new user to access and manage NiFi and NiFi Registry.

Related Information

Cloudera Security Overview

User Authorization

As a Cloudera administrator, after you create an environment, you must enable flow-management users to access the environment, utilize resources, and perform tasks in Cloudera. Assign the appropriate Cloudera resource role and one or more Ranger access policies based on the type of access the user requires.

A Cloudera resource role grants a user or user-group permission to access and perform tasks on a resource.

A Ranger access policy for flow management contains one or more access rights to NiFi or NiFi Registry resources.

To learn about Cloudera resource roles and Ranger policies, see the following documents:

- *Understanding roles and resource roles*
- *Ranger Policies Overview*

Related Information

Understanding roles and resource roles

Ranger Policies Overview

Authorization workflow

Before you begin, review the workflow to understand the process and options involved in authorizing users to flow management resources.

The following diagram shows the prerequisites and the workflow involved in authorizing a flow management user:

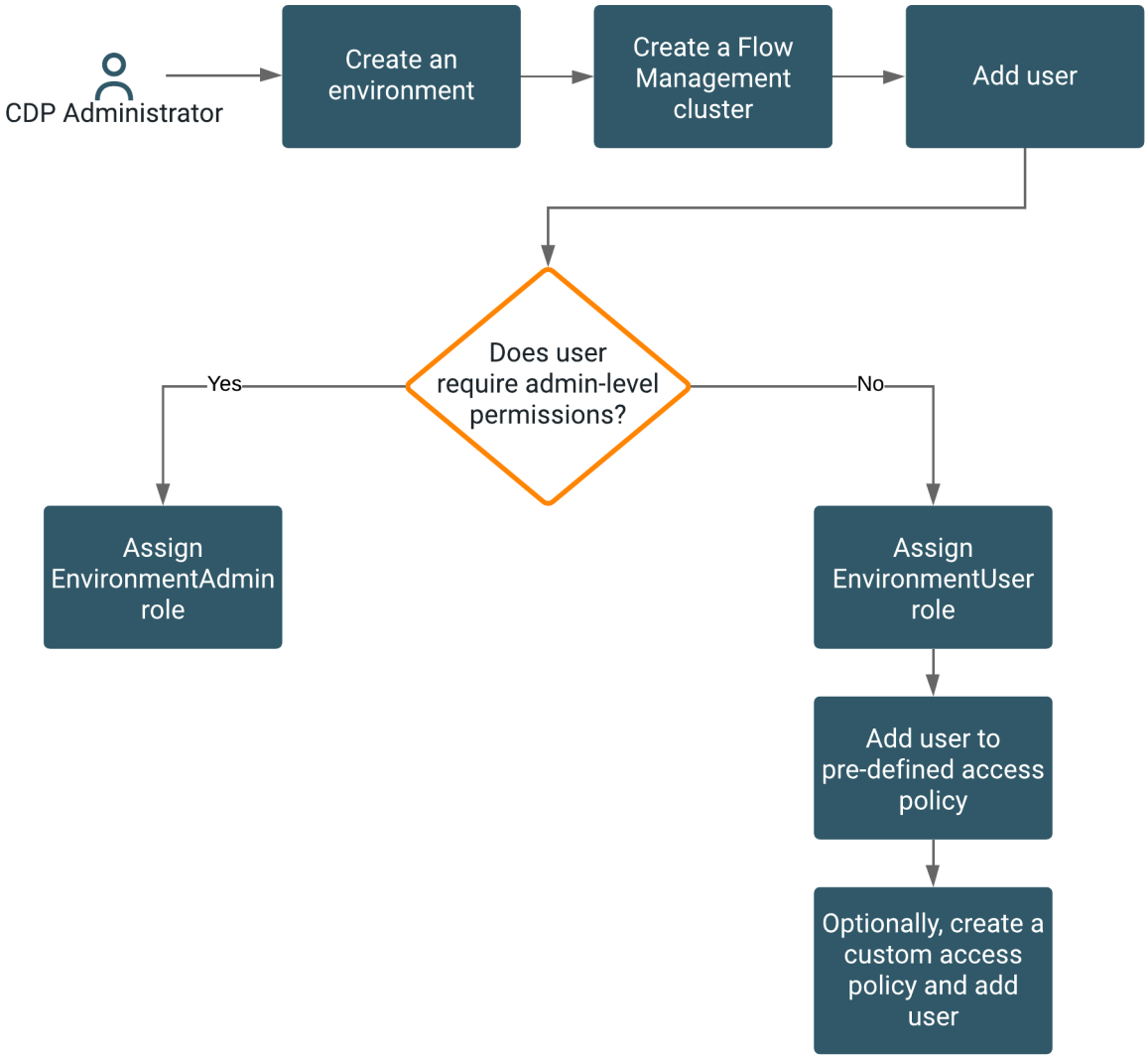

Note that you must select one of the following options based on the privileges the user is entitled to.

- If the user requires administrator-level permissions to NiFi and NiFi Registry, assign the EnvironmentAdmin role.
- If the user requires selective permissions to NiFi and NiFi Registry, assign the EnvironmentAdmin role and add the user to the appropriate Ranger policies. Optionally, create a custom policy and add the user to it.

Assigning administrator level permissions

As a Cloudera administrator, assign the EnvironmentAdmin role to enable users to have administrator-level privileges to the environment. With the EnvironmentAdmin role, the user can access and manage environments, Flow Management clusters, and NiFi and NiFi Registry resources. Users can also authorize other users to access flow management resources.

Before you begin

Ensure that you meet the following prerequisites:

- You created a Flow Management cluster.
- You determined the permission level for each user.

Procedure

1. Click the **Environments** tab.
2. Locate the environment.
3. Click the environment name.
4. Click Actions > Manage Access.
   The **Access** page appears.
5. Locate the user and click Update Roles.
   The **Update Resource Role** page for the user appears.
6. Check the EnvironmentAdmin option.
7. Click Update Roles.
8. Go back to the **Environments** tab and locate the environment.
9. Click Actions > Synchronize Users to FreeIPA.
   The **Sync Users** window appears.
10. Click Sync Users.
    This synchronizes the user to the FreeIPA identity management system to enable SSO.

Results

With the EnvironmentAdmin role and membership in the internal NiFi or NiFi Registry groups, the user has the ability to:

- Access and manage the environment and Flow Management clusters.
- Authorize users or groups by adding them to Ranger access polices.
- Modify or create conditions in predefined Ranger access policies.
- Create new Ranger access policies and create conditions that specify the desired level of access for each user or group.

Note: The EnvironmentAdmin role also gives a user the following privileges:

- Administrator rights to environments outside of NiFi.
- The right to modify predefined or custom Ranger access policies for any Ranger service in the environment.

For more information on roles, see *Understanding roles and resource roles*.

For more information about Ranger access policies, see *Ranger Policies Overview*.

To authorize flow management users who do not require administrator-level permission, add the users individually or as a group to specific Ranger access policies for selective access to NiFi and NiFi Registry resources. For more information, see *Assigning selective permissions to a user*.

Related Information

Understanding roles and resource roles

Ranger Policies Overview

Assigning selective permissions to user

Assigning selective permissions to user

As a Cloudera administrator, assign the EnvironmentUser role to enable a user to access the environment and Flow Management clusters. Then, based on the user's access requirements, add the user or a group of users to the appropriate Ranger access policies for NiFi and NiFi Registry.

Perform the following steps to authorize access for a new user:

1. Assign the EnvironmentUser role.
2. Add the user or group to the appropriate pre-defined Ranger access policies.

3. Create a custom Ranger access policy and add the user or group.

Assign the EnvironmentUser role

For flow management users who do not need administrator privileges, you must assign the EnvironmentUser role. This role enables users to set their password and access the environment.

Before you begin

Ensure that you meet the following prerequisites:

- You created a Flow Management cluster.
- You determined the permission level for each user.

Procedure

1. From the Cloudera Management console, go to the Environments tab.
2. Use the Search bar to find the environment.
3. Click the environment name.
4. Click Actions > Manage Access.
   The **Access** page appears.
5. Locate the user and click Update Roles.
   The **Update Resource Role** page for the user appears.
6. Check the EnvironmentUser option.
7. Click Update Roles.
8. Go back to the **Environments** tab and locate the environment.
9. Click Actions > Synchronize Users to FreeIPA.
   The **Sync Users** window appears.
10. Click Sync Users.
    This synchronizes the user to the FreeIPA identity management system to enable SSO.

Results

The user is added to the environment and can access the environment and Flow Management clusters. You can now add the user or a group of users to Ranger policies that allow access to NiFi and NiFi Registry resources.

What to do next

Complete the steps listed in *Add the user to predefined Ranger access policies*.

Related Information

Add the user to predefined Ranger access policies

Add the user to predefined Ranger access policies

After assigning the EnvironmentUser role to the new user, you must add the user to the appropriate predefined Ranger access policies for NiFi and NiFi Registry. These policies determine what the user can command, control, and observe in a NiFi dataflow or in the NiFi Registry.

About this task

Each predefined Ranger access policy confers specific rights to NiFi or NiFi Registry resources. When an authenticated flow-management user attempts to view or modify a NiFi or NiFi Registry resource, the system checks whether the user is associated with the specific Ranger access policy that confers the privileges to perform that action.

For more information, see:

- *Predefined Ranger access policies for NiFi resources*
- *Preefined Ranger access policies for NiFi Registry resources*

Procedure

1. From the base cluster with Ranger, click the Ranger icon.
   The **Ranger Service Manager** page appears.

   Each cluster in the environment is listed under its respective service. For example, the NiFi clusters in the environment are listed under NiFi.
2. Select a cluster from either the NiFi or NiFi Registry section.
   The following image shows the list of predefined policies for NiFi:

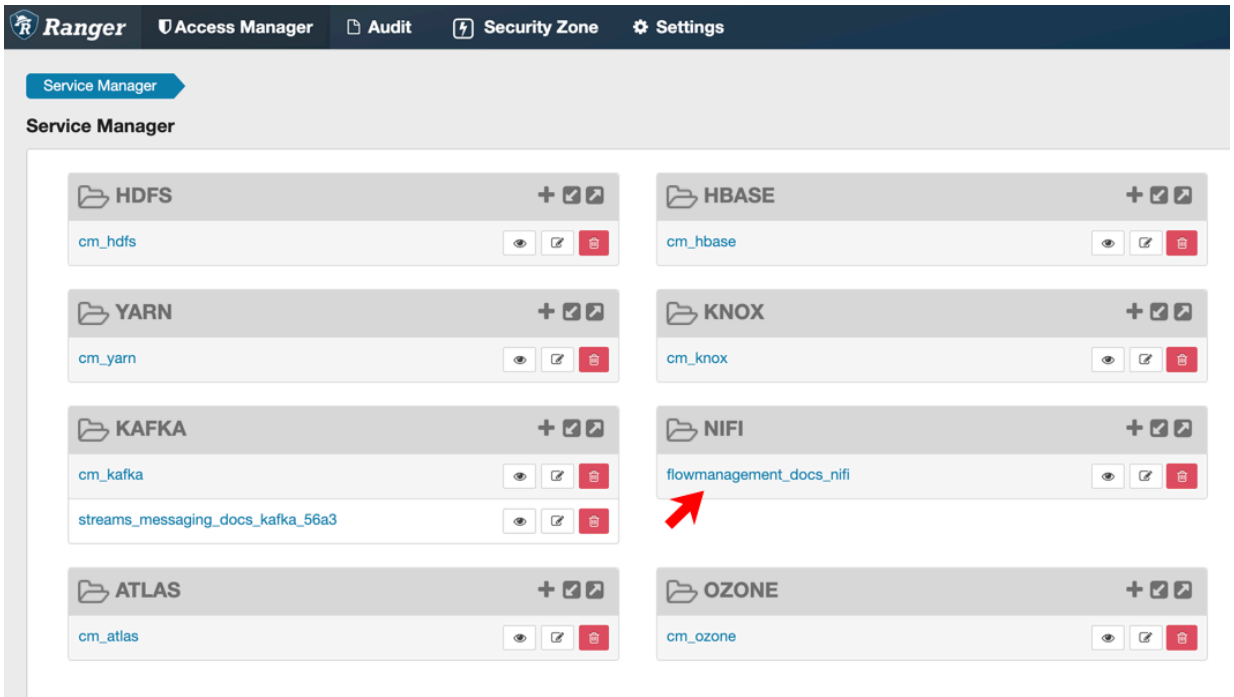

   The **List of Policies** page appears.
3. Click the ID for a policy.
   The following image shows the list of predefined policies for NiFi:

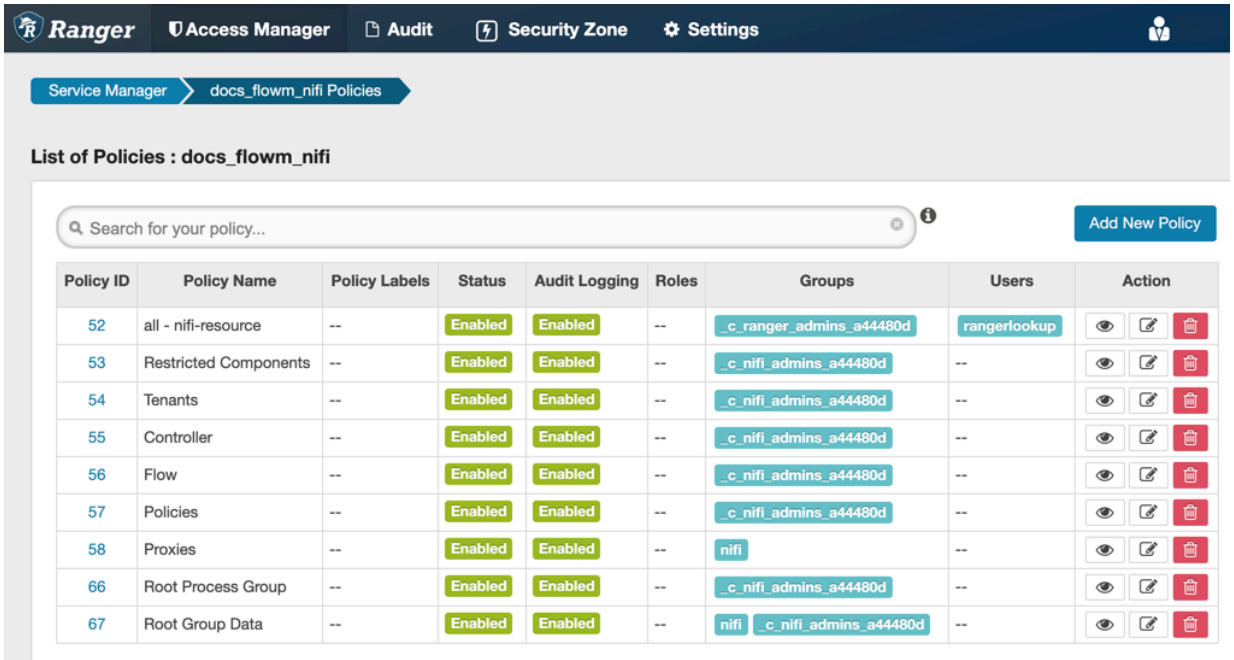

   The **Edit Policy** page appears.

4. In the Allow Conditions section, add the user or the user group to the Select User field.
5. Click Save.

Results

The user now has the NiFi and NiFi Registry rights according to the policies you added the user or user group to. These rights are inherited down the hierarchy unless there is a more specific policy on a component.

What to do next

Complete the steps listed in *Create a Custom Access Policy*.

Related Information

Predefined Ranger access policies for Apache NiFi

Predefined Ranger access policies for Apache NiFi Registry

Create a custom access policy

Create a custom access policy

If the user cannot access the component through an inherited Ranger access policy, then you must create a custom Ranger access policy for the specific component and add the user to this policy. If all the users in a group require the same access, you can add the user group to the Ranger access policy.

About this task

A user might need access to specific NiFi or NiFi Registry resources such as a processor, processor group, remote process group, funnel, label, controller service, or bucket. Each custom Ranger access policy provides access to a specific component.

First determine which NiFi or NiFi Registry components a user needs access to. Then create a new policy for each component and add the user or user group to the new policy.

When you create a new policy, you must specify the ID of the component that the user requires access to.

Note:

If a user requires permission to view or modify data for a specific component, you must create a custom data access policy and add the user and the nifi group to that policy.

The nifi group is a dynamically-managed group that exists on all Flow Management Data Hub hosts and contains the identities of NiFi and Knox nodes. When you add the nifi group to the data policy for a specific component, you authorize the nodes to access data on behalf of the user.

Procedure

1. From the NiFi canvas, copy the ID of the process group, SSL Context Service, or controller service for reporting tasks that the user needs access to.

2. To locate the ID for a process group:
   a) Click the process group.
      The ID appears in the **Operate** pane.

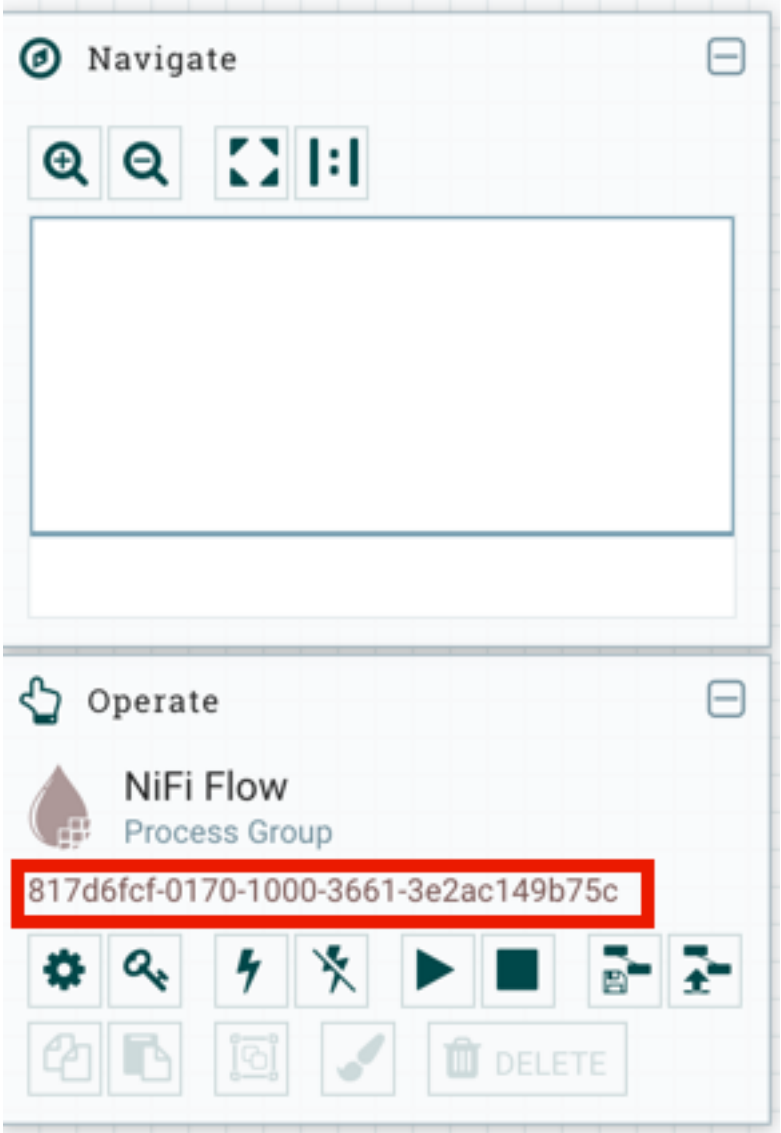

   b) Copy the ID.

3. To locate the ID of the SSL Context Service:
   a) Click the settings icon on the process group.
      The **NiFi Flow Configuration** appears.
   b) Click the **Controller Services** tab.
   c) Click the **Settings** icon for the Default NiFi SSL Context Service.
      The **Controller Service Details** window appears.
   d) From the **Settings** tab, copy the ID from the Id field.

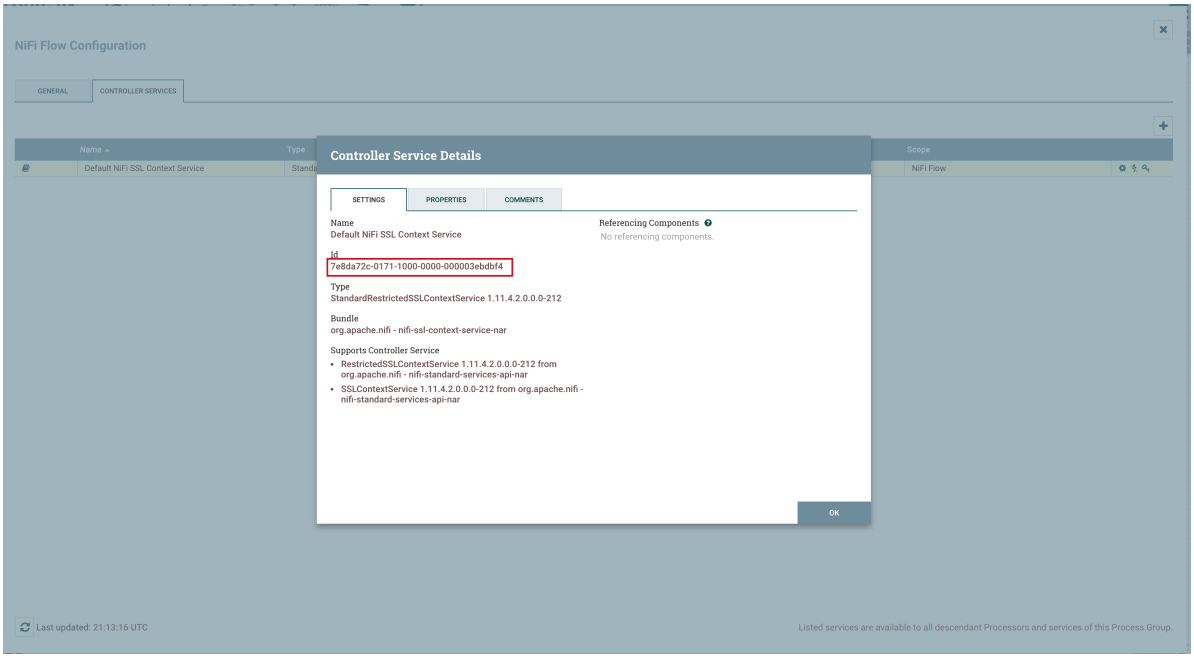

4. To locate the ID of a controller service for reporting tasks:
   a) Click the process group.
   b) Click the menu on the top right of the UI and select Controller Settings.

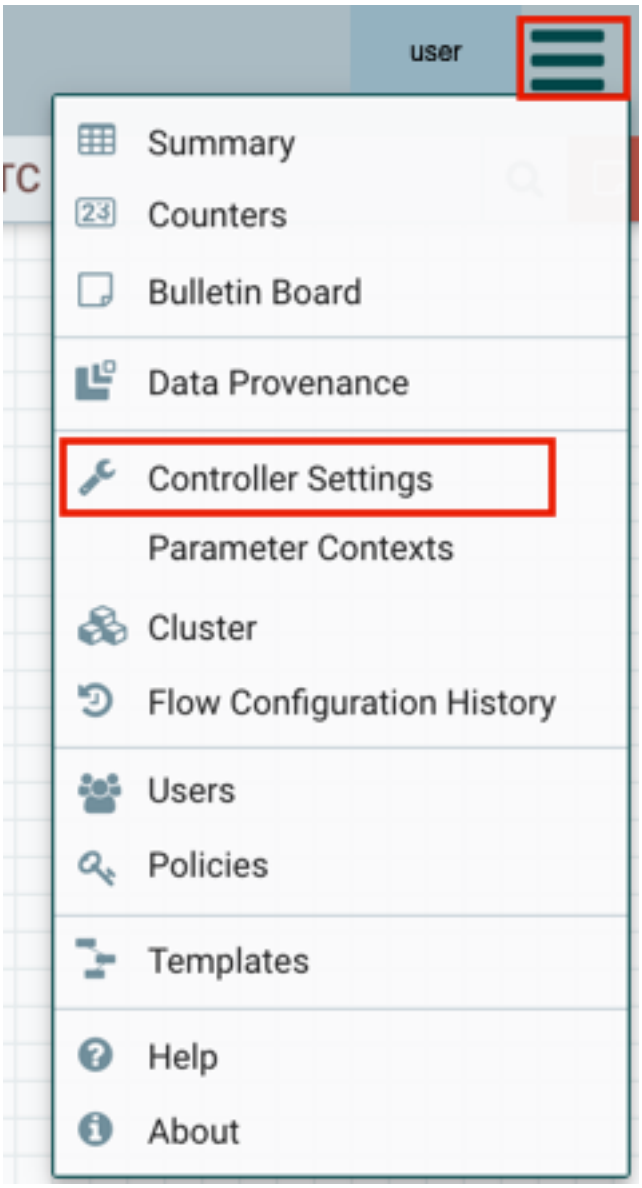

   The **NiFi Settings** page appears.
   c) Click the **Reporting Tasks Controller Services** tab.
   d) Click the Settings icon for the controller service.

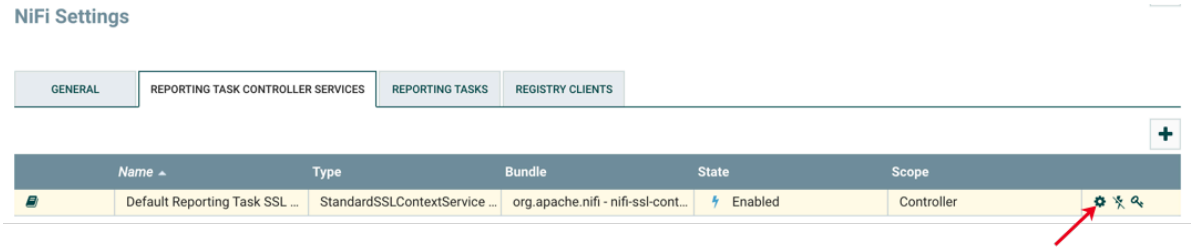

   The **Controller Service Details** page appears.
   e) From the **Settings** tab, copy the ID from the Id field.

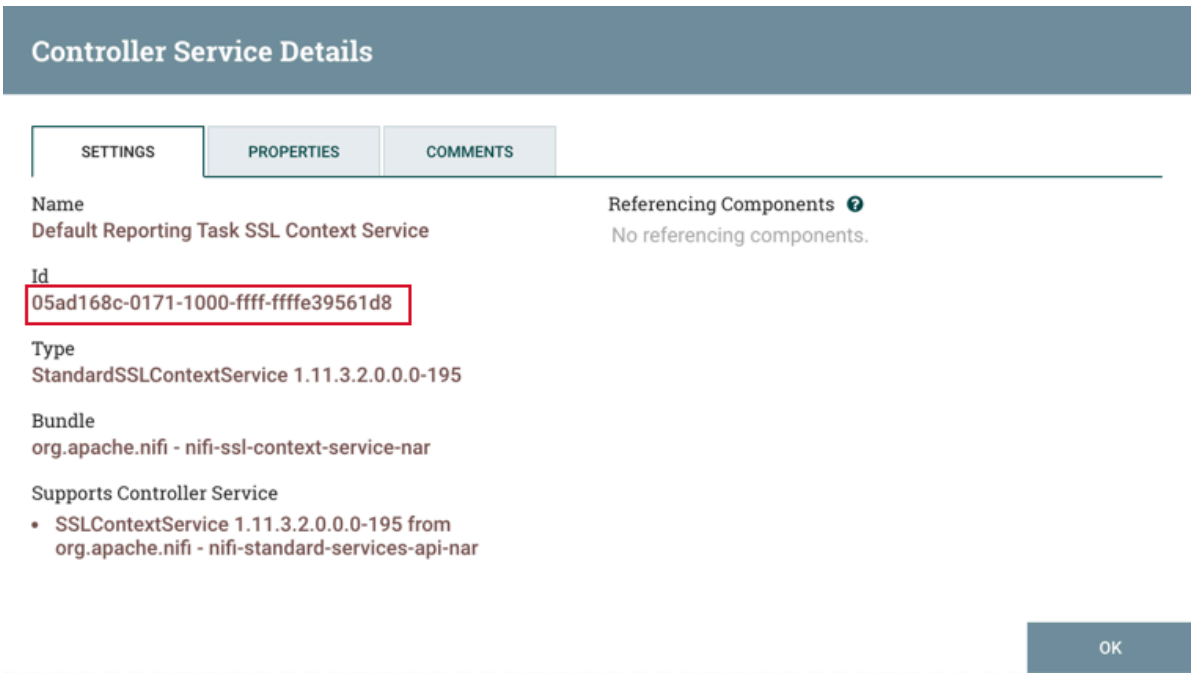

5. Go back to the **Ranger List of Policies** page.
6. Click Add New Policy.

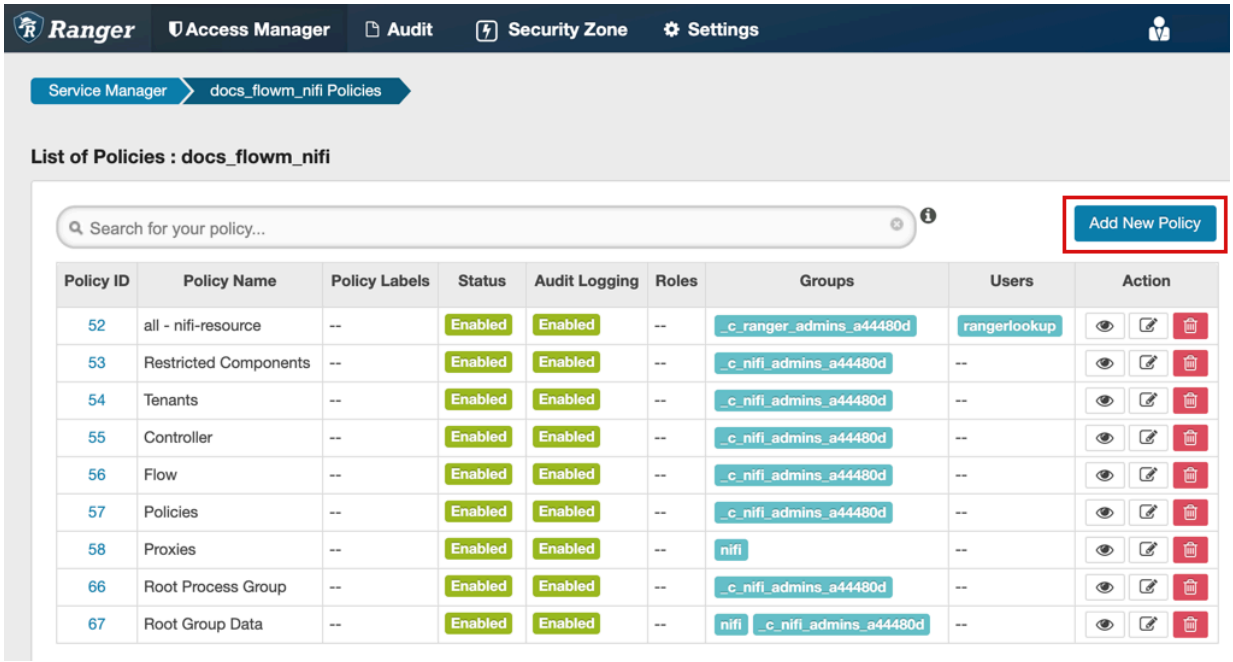

The **Create Policy** page appears.

7. Enter a unique name for the policy.
8. Optionally, enter a keyword in the Policy Label field to aid in searching for a policy.
9. Enter the resource descriptor and the resource ID in the NiFi Resource Identifier or NiFi Registry Resource Identifier field in the following format: <resource descriptor>/<resource ID>

   To determine a NiFi resource descriptor, see *Predefined Ranger access policies for Apache NiFi*.

   To determine a NiFi Registry resource descriptor, see *Predefined Ranger access policies for Apache NiFI Registry*.
10. Optionally, enter a description.

11. Add a user or a group.

> **Note:** If a user requires permission to view or modify the data for a specific component, you must create a data policy with /data/<component-type>/<component-UUID> as the resource identifier. Then add the user and the nifi group to the policy to authorize the NiFi and Knox nodes to access data on behalf of the user.

12. Set the permission level for the user or group.
13. Click Add.

Results

The user or group of users can now access the component specified in the custom policy.

Related Information

Predefined Ranger access policies for Apache NiFi

Predefined Ranger access policies for Apache NiFi Registry

Authorization example

You can review this example to understand how you can enable a flow-management user to perform specific tasks like setting up version control for a flow, by assigning the appropriate Ranger policies.

User A must be able to do the following tasks:

- Access the NiFi UI.
- Export a flow.
- View data queued in connections.
- View data flowing through.
- Use a NiFi SSLContextService to connect to SSL-enabled systems.
- Set up version control for a flow.

Complete the following steps to enable User A to perform the required tasks:

1. Add User A to the predefined Ranger access policy for NiFi, Flow. Set the permissions to Read.

   The Flow policy gives the user the right to view the NiFi UI.
2. Create a Ranger access policy for NiFi with:

   - Resource descriptor: /data/process-groups/<ID of process-group>
   - Permission: Read and Write

   Add User A to this custom policy. The policy gives the user the right to export the data, view the data that is queued and flowing through the connections.
3. Create a Ranger access policy for NiFi with:

   - Resource descriptor: /controller-service/<ID of SSL Context Service>
   - Permission: Read

   Add User A to this custom policy. The policy gives the user the right to use the specified SSLContextService in their flows to connect to SSL-enabled systems.
4. Create a Ranger access policy for NiFi Registry with:

   - Resource descriptor: /buckets/<ID of bucket>
   - Permission: Read, Write, and Delete

   Add User A to this custom policy. The policy gives the user the right to set up version control for a flow.

Predefined Ranger access policies for Apache NiFi

You can review the predefined Ranger policies for NiFi to determine the appropriate policy to assign to a user.

The following table lists the predefined Ranger access policies for NiFi. If you create a custom policy, refer to the Resource Descriptor column in this table to enter the value in the NiFi Resource Identifier field on the **New Policy** page.

Important:

Do not rename the default policies as some cluster operations rely on these policy names.

Do not select the Delegate Admin checkbox.

Note: The NiFi and Knox nodes have permission to the following Ranger policies:

- Proxies; /proxy
- Root Group Data; /data/process-groups

Ranger Policy	Description	Resource Descriptor
Controller	Allows users to view and modify the controller including Reporting Tasks, Controller Services, Parameter Contexts and Nodes in the Cluster.	/controller
Flow	Allows users to view the NiFi UI.	/flow
Policies	Allows users to view the policies for all components.	/policies
Provenance	Allows users to submit a Provenance Search and request Event Lineage.	/provenance
Proxies	Allows NiFi and Knox hosts to proxy user requests. Does not apply to users or user groups.	/proxy
Restricted Components	Allows users to create/modify restricted components assuming other permissions are sufficient. The restricted components may indicate the specific permissions that are required. Permissions can be granted for specific restrictions or be granted regardless of restrictions. If permission is granted regardless of restrictions, the user can create/modify all restricted components. Some examples of restricted components are ExecuteScript, List/FetchHDFS, and TailFile.	/restricted-components See the *NiFi Restricted Components* topic for information on the sub-policies.
Root Group Data	Allows users and the nifi group to view and delete data from the root group and down the hierarchy unless there is a more specific policy on a component. **Note:** The nifi group is a dynamically managed list of Knox and NiFi node identities. The group exists on all Data Hub Flow Management hosts.	/data/process-groups/<uuid>
Root Group Provenance Data	Allows users to view provenance data.	/provenance-data/process-groups/
Root Process Group	Allows users to view and modify the root process group including adding/removing processors to the canvas. This policy is inherited down the hierarchy unless there is a more specific policy on a component.	/process-groups/<uuid>
Tenants	Allows users to view and modify user accounts and user groups.	/tenants

Predefined Ranger access policies for Apache NiFi Registry

You can review the predefined Ranger policies for NiFi Registry to determine the appropriate policy to assign to a user.

The following table lists the pre-defined Ranger access policies for NiFi Registry. If you create a custom policy, refer to the Resource Descriptor column in this table to enter the value in the NiFi Registry Resource Identifier field on the **New Policy** page.

Important:

Do not rename the default policies as some cluster operations rely on these policy names.

Do not select the Delegate Admin checkbox.

Note: The NiFi Registry and Knox nodes have permission to the Proxies (/proxy) Ranger policy.

Ranger Policy	Description	Resource Descriptor
Actuator	Allows users to access the Spring Boot Actuator end-points.	/actuator
Buckets	Allows users to view and modify all buckets.	/buckets
Policies	Allows users to view the policies for all components.	/policies
Proxies	Allows NiFi Registry and Knox hosts to proxy user requests. Does not apply to users or user groups.	/proxy
Swagger	Allows users to access the self-hosted Swagger UI.	/swagger
Tenants	Allows users to view and modify user accounts and user groups.	/tenants

Cloudera DataFlow for Data Hub 7.3.1

Moving data in and out of Snowflake

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Pushing data to and moving data from Snowflake using NiFi.....4

Moving data out of Snowflake..... 4

Before you begin..... 4
Downloading the Snowflake JDBC driver JAR file.....4
Adding Snowflake CA certificates to NiFi truststore.....5
Building your data flow..... 6
Creating Controller Services for your data flow..... 7
Configuring your source processor..... 8
Configuring your target processor.....9
Confirming your data flow success.....10

Pushing data into Snowflake..... 10

Before you begin..... 10
Adding Snowflake CA certificates to NiFi truststore.....10
Building your dataflow.....11
Configuring your Controller Services..... 12
Configure your source processor.....14
Configuring your target processor.....15
Confirming your data flow success.....15

Next steps..... 15

Pushing data to and moving data from Snowflake using NiFi

Cloudera supports pulling data out of Snowflake and pushing data into Snowflake, using Apache NiFi. You can use Apache NiFi to move data back and forth between Snowflake and other systems including Cloudera Data Platform.

Moving data out of Snowflake

You can create a NiFi dataflow to move data out of Snowflake. To do this, you must meet some prerequisites, download the Snowflake JDBC driver jar file, update the NiFi truststore, configure your Controller Services, build the dataflow, and configure the source and target Processors.

Pushing data into Snowflake

You can create a NiFi data to push data into Snowflake. To do this, you must meet some prerequisites, configure your Controller Services, build the dataflow, and configure your source and target Processors.

Moving data out of Snowflake

You can create a NiFi data flow to move data out of Snowflake. To do this, you must meet some prerequisites, download the Snowflake JDBC driver JAR file, update the NiFi truststore, configure your controller services, build your data flow, and configure the source and target processors.

Before you begin

Before setting up a NiFi data flow to pull data from a Snowflake database table, you must meet certain minimum prerequisites.

- You have a CFM cluster running NiFi.
- You have a Snowflake account.
- You have read access to a Snowflake database table.

Related Information

Setting up your Flow Management cluster

Downloading the Snowflake JDBC driver JAR file

Before you can create a dataflow that moves data out of a Snowflake database, you must ensure that NiFi can interact with the Snowflake database using a JDBC interface. To do this, you must download the Snowflake JDBC driver JAR file, upload it to each NiFi node in your cluster, and ensure that the proper permissions are set.

About this task

You need the Snowflake driver file, when defining the DBCP Connection Pool Controller Service that is used in the NiFi processors, in the CFM dataflow that moves data out of Snowflake. This driver allows NiFi to interact with the Snowflake database through the JDBC interface.

Before you begin

- You must have reviewed and met the prerequisites.
- You must have logged in as an admin user.
- You must have created a directory on all the NiFi nodes where you will make the driver file available for NiFi

Procedure

1. Download the latest Snowflake JDBC JAR file from the Snowflake website.

   For information on how to get the Snowflake JDBC JAR file, see the Snowflake documentation in the *Related information* section below.
2. Upload the JAR file to each of the NiFi nodes in your cluster.
3. As a root user, on every NiFi node in your cluster, update the permissions for the Snowflake JDBC JAR file, and make sure this can be accessed and read by the `nifi` user:

```
chown nifi: <snowflake-jdbc>.jar
```

What to do next

After you have downloaded the Snowflake JDBC driver, proceed to the following steps.

1. Add Snowflake CA certificates to the NiFi truststore.
2. Build the dataflow.
3. Create the controller services for your dataflow.
4. Configure your source processor.
5. Configure your target processor.
6. Confirm your dataflow is successful.

Related Information

Downloading / Integrating the JDBC Driver

Adding Snowflake CA certificates to NiFi truststore

You must ensure that NiFi can communicate securely with Snowflake. To do this, configure NiFi to trust the Snowflake Certificate Authority (CA) by merging the default Snowflake JDK truststore content into the NiFi truststore.

About this task

The Snowflake endpoints have certificates signed by a Certificate Authority (CA). You must configure NiFi so that the Snowflake CAs are trusted by NiFi. The recommended approach is to follow the steps documented in *How to Add Root and Intermediate CAs to Truststore for TLS/SSL*. See the link in the *Related information* below.

Another approach is to merge the content of the default truststore of the JDK into the NiFi truststore. This approach is describe here.

Note:

If you have added the Snowflake CA into the NiFi truststore, you may skip this section. For example, this is the case if you are using Flow Management clusters in CDP Public Cloud.

Before you begin

You must have downloaded the Snowflake JDBC driver JAR file.

Procedure

1. Merge the content of the JDK truststore (represented by the file cacerts), using the below command:

```
keytool
-importkeystore
-srckeystore [***path to cacerts***]
```

```
-destkeystore [***path to NiFi truststore***]
```

2. When prompted, enter the password of the JDK truststore. The default is changeit.
3. Enter the NiFi truststore password.

What to do next

If you have changed the truststore, you must restart NiFi before the changes are taken into account.

After you have finished adding Snowflake certificates to the NiFi truststores, you may move on to the following steps:

- Build the data flow.
- Create the controller services for your data flow.
- Configure your source processor.
- Configure your target processor.
- Confirm your data flow is successful.

Related Information

How to Add Root and Intermediate CAs to Truststore for TLS/SSL

Building your data flow

Set up the elements of your NiFi data flow that enables you to move data out of Snowflake using Apache NiFi. This involves opening NiFi, adding processors to your NiFi canvas, and connecting the processors.

About this task

When you are building a data flow to move data out of Snowflake using Apache NiFi, you can consider using the following processors to build your data flow:

- ListDatabaseTables
- ExecuteSQLRecord

Before you begin

You must have added the Snowflake CA certificates to the NiFi truststore.

Procedure

1. Launch NiFi from your CDP Public Cloud or CDP Private Cloud Base cluster.
2. Add the NiFi processors to your canvas.
   a. Select the Processor icon from the Cloudera Flow Management Actions pane, and drag a processor to the canvas.
   b. Use the Add Processor filter box to search for the processor you want to add, and then click Add.
   c. Add each of the processors you want to use for your data flow.
3. Connect the two processors to create a data flow.
   a. Click the Connection icon in the first processor, and drag it to the second processor.

      A Create Connection dialog displays. It has two tabs: Details and Settings, where you can configure the connection's name, FlowFile expiration time period, thresholds for back pressure, load balance strategy, and prioritization.
   b. Click Add to close the dialog box and add the connection to your flow. Optionally, you can add success and failure funnels to your data flow, which help you see where flow files are routed when your data flow is running.

Results

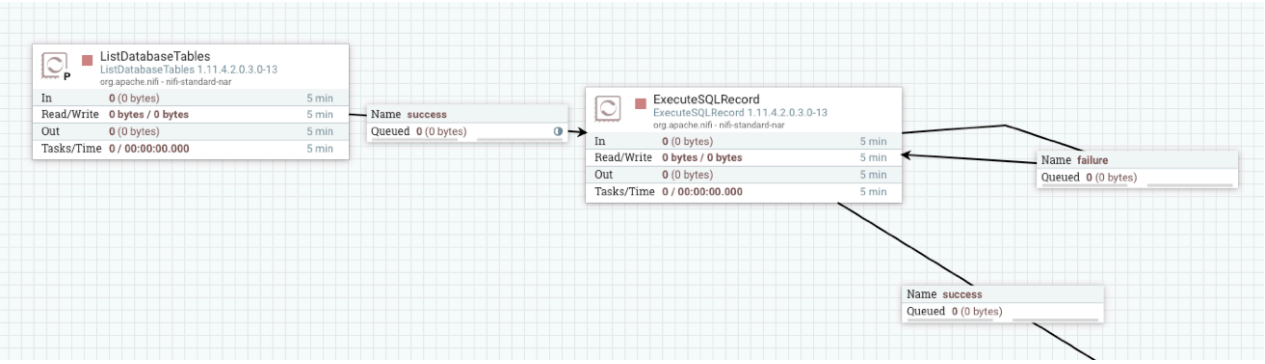

What to do next

Once you have finished building the data flow, move on to the following steps:

- Create controller services for your data flow.
- Configure your source processor.
- Configure your target processor.

Creating Controller Services for your data flow

You can add controller services that can provide shared services to be used by the processors in your data flow. Create them after you build the NiFi data flow and before you configure the processors, so that they are available when you configure your NiFi processors.

About this task

For this data flow, create two controller services.

- `AvroWriter`
- `DBCPConnectionPool`

See the *Related Information* section below, for links to complete details about these Apache NiFi controller services.

Before you begin

You must have built your data flow.

Procedure

1. To add a controller service to your flow, right-click on the canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.
3. Click the + button to display the Add Controller Service dialog.
4. Select the required controller service and click Add.
5. Click the Configure icon in the right-hand column and configure the necessary options.
6. Click Apply to save the changes.
7. Click the Enable button (flash) in the far-right column of the Controller Services tab to enable the controller service.

Example

You require the following controller services for this example of a NiFi data flow moving data out of Snowflake.

AvroRecordSetWriter

You use this controller service as the writer, when writing the data as Flow Files in NiFi. Avro is a good option in most cases but you may want to choose another writer depending on your requirements.

In general, you may accept the default configurations.

DBCPConnectionPool

This controller service allows NiFi to interact with the Snowflake database through the JDBC interface.

Property	Example value for a data flow pulling data from Snowflake
Name	SnowflakeJDBCConnectionPool
Database Connection URL	`jdbc:snowflake://[***account_name***].snowflakecomputing.com/?[***connection_params***]` Where: • account_name is similar to cb56215.europe-west2.gcp • connection_parameters is similar to db=DEMO_DB
Database Driver Class Name	net.snowflake.client.jdbc.SnowflakeDriver

What to do next

After you have finished configuring your controller services:

- Configure your source processor
- Configure your target processor
- Confirm your data flow is running

Related Information

AvroRecordSetWriter

DBCPConnectionPool

Configuring your source processor

You can use the `ListDatabaseTable` processor to get data from your Snowflake table. To do this, launch the Configure Processor window, specify the necessary configurations, and start the process to verify that you can view the Snowflake table.

About this task

Use the ListDatabaseTables processor to get data from Snowflake database tables.

Set the proper Scheduling Strategy on the processor in accordance with your use case. You use this processor to list the tables available in the remote database. You can update the value of the Refresh Interval property in combination with the Scheduling Strategy to determine at which frequency tables will be listed.

Configure this processor to run only on the primary node of your NiFi cluster.

Before you begin

- You must have built the data flow.
- You must have created your controller services.

Procedure

1. Launch the Configure Process window, by right-clicking the processor and selecting Configure.
2. Configure ListDatabaseTables according to your use case and operational objectives.

3. Configure Database Connection Pooling Service to call the database connection Controller Service you created in the preceding step.
4. You can use the Scheduling Strategy and the Refresh Interval configuration properties to determine how frequently the tables will be listed in the Snowflake database.
5. Use the Table Name Pattern value to specify whether you want to pull all the tables or just some.

   If you want to retreive a few of the tables, define a regular expression. If the property value is not set, all tables are retrieved.

Results

You can confirm that you have configured ListDatabaseTables correctly by starting the processor and confirming that you can view the Snowflake table.

What to do next

After you have finished configuring your source processor, proceed to:

- Configure your target processor.
- Confirm your data flow is running.

Related Information

ListDatabaseTables

Configuring your target processor

In a data flow that is pulling data from a Snowflake database, configure the `ExecuteSQLRecord` processor to handle data pooling from remote tables. To do this, launch the processor configuration window and provide the configurations appropriate for your use case.

About this task

Configure ExecuteSQLRecord.

The ExecuteSQLRecord processor handles data pooling from the remote tables. To load balance the execution of the SQL queries across the NiFi nodes, you should update the configuration of the relationship between the ListDatabase Tables processor and the ExecuteSQLRecord processor and define a round robin load balancing strategy. This way the flow files generated on the primary node by the ListDatabaseTables processors will be shuffled across the nodes of the NiFi cluster to distribute the workload.

Procedure

1. Right-click `ExecuteSQLRecord` and click Configure Processor.
2. For Database Connection Pooling Service, select your DBCPConnectionPool controller service you defined earlier.
3. For SQL select query, if you want to extract all the data from the remote tables, you can use the SQL query:

   ```
   SELECT * FROM ${db.table.fullname}
   ```

4. For Record Writer, select the Avro writer controller service you defined earlier.
5. For Normalize Table/Column Names, you may want to set this property to true in case some of the column names are not compatible with Avro specifications.

What to do next

After you have configured your target processor, proceed to confirming your data flow success.

Related Information

ExecuteSQLRecord

Confirming your data flow success

Confirm that you have successfully built a data flow to move data out of Snowflake database tables by starting your data flow and verifying that data is moving through it.

About this task

In a real life use case, you may want to connect your `ExecuteSQLRecord` processor with a processor that can send your data to an external system.

Procedure

1. Select all the data flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternatively, right-click a single component and choose Start from the context menu.

Results

Your flow should run without any errors. Data should be pulled from the Snowflake database tables using `ListDatabaseTables` and pooled using `ExecuteSQLRecord`.

Pushing data into Snowflake

You can create a NiFi data flow to push data into a Snowflake database table. To do this, you must meet certain prerequisites, configure your controller services, build the dataflow, and configure your source and target processors.

Before you begin

Before setting up a NiFi dataflow to push data to a Snowflake database table, you must meet certain minimum prerequisites.

- You have a CFM cluster running NiFi.
- You have a Snowflake account.
- Your Snowflake user account has write access to a Snowflake database table.
- You have et up your destination Snowflake table with the right columns and columns type.

  Table columns for this example are R_REGIONKEY, R_NAME, and R_COMMENT

Adding Snowflake CA certificates to NiFi truststore

You must ensure that NiFi can communicate securely with Snowflake. To do this, configure NiFi to trust the Snowflake Certificate Authority (CA) by merging the default Snowflake JDK truststore content into the NiFi truststore.

About this task

The Snowflake endpoints have certificates signed by a Certificate Authority (CA). You must configure NiFi so that the Snowflake CAs are trusted by NiFi. The recommended approach is to follow the steps documented in *How to Add Root and Intermediate CAs to Truststore for TLS/SSL*. See the link in the *Related information* below.

Another approach is to merge the content of the default truststore of the JDK into the NiFi truststore. This approach is described here.

Note:

If you have added the Snowflake CA into the NiFi truststore, you may skip this section. For example, this is the case if you are using Flow Management clusters in CDP Public Cloud.

Before you begin

You have reviewed and met the prerequisites

Procedure

1. Merge the content of the JDK truststore (represented by the file cacerts), you can use the below command:

```
keytool
-importkeystore
-srckeystore [***path to cacerts***]
-destkeystore [***path to NiFi truststore***]
```

2. When prompted, enter the password of the JDK truststore. The default is changeit.
3. Enter the NiFi truststore password.

What to do next

If you have changed the truststore, you must restart NiFi before the changes are taken into account.

After you have finished adding Snowflake certificates to the NiFi truststores, you may move on to the following steps.

- Build the data flow.
- Create the controller services for your data flow.
- Configure your source processor.
- Configure your target processor.
- Confirm your data flow is successful.

Related Information

How to Add Root and Intermediate CAs to Truststore for TLS/SSL

Building your dataflow

Set up the elements of your NiFi data flow that enables you to move data into Snowflake using Apache NiFi. This involves opening NiFi, adding processors to your NiFi canvas, and connecting the processors.

About this task

When you are building a flow to move data into Snowflake using Apache NiFi, you can consider using the following processors to build your data flow:

- GenerateFlowFile
- PutDatabaseRecord

Before you begin

You must have reviewed and met the prerequisites.

Procedure

1. Launch the NiFi from your CDP Public Cloud or CDP Private Cloud Base cluster.
2. Add the NiFi processors to your canvas.
   a. Select the Processor icon from the Cloudera Flow Management Actions pane, and drag a processor to the canvas.
   b. Use the Add Processor filter box to search for the processor you want to add, and then click Add.
   c. Add each of the processors you want to use for your data flow.
3. Connect the two processors to create a data flow.
   a. Click the Connection icon in the first processor, and drag it to the second processor.

      A Create Connection dialog displays. It has two tabs: Details and Settings where you can configure the connection's name, flow file expiration time period, thresholds for back pressure, load balance strategy, and prioritization.
   b. Click Add to close the dialog box and add the connection to your flow. Optionally, you can add success and failure funnels to your data flow, which help you see where flow files are routed when your data flow is running.

Results

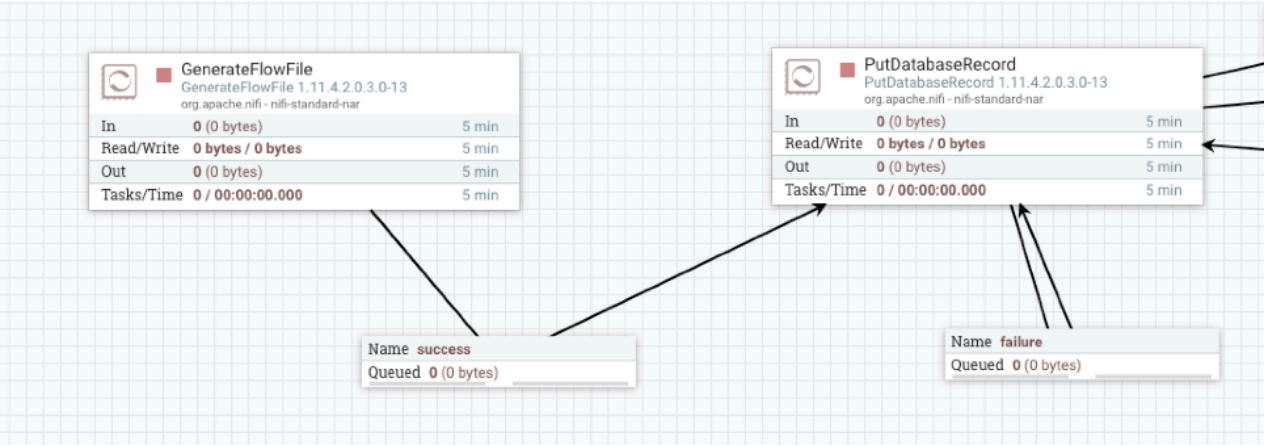

What to do next

Once you have finished building the data flow, move on to the following steps:

- Create controller services for your data flow.
- Configure your source processor.
- Configure your target processor.
- Confirm your data flow success.

Configuring your Controller Services

You can add Controller Services that can provide shared services to be used by the processors in your data flow. Create them after you build the NiFi dataflow and before you configure the processors, so that they are available when you configure your NiFi processors.

Before you begin

You have built the data flow on the NiFi canvas.

About this task

For this data flow, create two controller services.

- `DBCPConnectionPool`
- `CSVReader`

See the *Related Information* section below, for links to full details about these Apache NiFi controller services.

Procedure

1. To add a controller service to your flow, right-click on the canvas and select Configure from the pop-up menu. This displays the Controller Services Configuration window.
2. Select the Controller Services tab.
3. Click the + button to display the Add Controller Service dialog.
4. Select the required controller service and click Add.
5. Click the Configure icon in the right-hand column and configure the necessary options.
6. Click Apply to save the changes.
7. Click the Enable button (flash) in the far-right column of the Controller Services tab to enable the controller service.

Example

DBCPConnectionPool

Reuse the same `DBCPConnectionPool` controller service that you created before to interact with the Snowflake database.

Property	Example value for Snowflake ingest data flow
Name	Rename to SnowflakeJDBCConnectionPool
Database Connection URL	`jdbc:snowflake://<account_name>.snowflakecomputing.com/?<connection_params>` Where: • <account_name> is similar to cb56215.europe-west2.gcp • <connection_parameters> is similar to db=DEMO_DB
Database Driver Class Name	net.snowflake.client.jdbc.SnowflakeDriver

CSVReader

In this example, you are generating CSV data to ingest into the Snowflake remote database table. As a result, you must also configure a `CSVReader` Controller Service to parse the data. You require the CSV data to correctly construct the queries executed against the table to ingest the data.

You can accept the default configurations.

What to do next

Once you have configured your controller services, proceed to the following tasks:

- Configure your source processor.
- Configure your target processor.
- Confirm your data flow success.

Related Information

DBCPConnectionPool

CSVReader

Configure your source processor

You can set up a data flow to push data into Snowflake database tables from many different locations. To do this, start by configuring the processor for your data source by launching the Configure Processor window and specifying the necessary configurations.

About this task

Configure the `GenerateFlowFile` processor to create random data for this example data flow. `GenerateFlowFile` is useful when you are testing or creating proof of concept data flows. When you have confirmed that this data flow meets your business use case, you can replace it with a processor getting data from your actual data source.

See *Related Information* for full details on this Apache NiFi processor.

Before you begin

- You must have built the data flow.
- You must have configured your controller services.

Procedure

1. Launch the Configure Processor window, by right-clicking the `GenerateFlowFile` processor and selecting Configure.

   A configuration dialog box with the following tabs is displayed: Settings, Scheduling, Properties, and Comments.
2. Configure the processor according to the behavior you expect in your data flow.

   See the *Example* section below for recommended configuration to satisfy this example use case.
3. Save the changes by clicking Apply .

Example

The following settings and properties are used in this example:

Table 1: GenerateFlowFile processor scheduling

Scheduling	Description	Example value for ingest data flow
Run Schedule	Run schedule dictates how often the processor should be scheduled to run. The valid values for this field depend on the selected scheduling strategy.	60 s

Table 2: GenerateFlowFile processor properties

<Title>?	Description	Example value for ingest data flow

Custom text	If the value of Data Format is text and if Unique FlowFiles is set to false, you can provide custom to be used as the content of the generated FlowFiles.
	The expression statement in the example value generates a random ID between 1 and 10 000, with random last names assigned.
		`R_REGIONKEY, R_NAME, R_COMMENT`
		`100,foo1, blablabla`
		`101, foo2, blabla`
		`102, foo3, bla`

What to do next

Once you have configured your controller services, proceed to the following tasks:

- Configure your target processor.

- Confirm your data flow success.

Related Information

GenerateFlowFile

Configuring your target processor

In a data flow pushing data to a Snowflake database table, configure the `PutDatabaseRecord` processor to move data into the Snowflake database table. To do this, launch the processor configuration window and provide the configurations appropriate for your use case.

About this task

Configure `PutDatabaseRecord` Processor. See *Related Information* for complete details on this Apache NiFi processor.

Procedure

1. Right-click `PutDatabaseRecord` and click Configure Processor.
2. In the Record Reader field, specify the CSV Reader controller service you created in the earlier step.
3. In the Statement Type field, select INSERT to insert data into the remote database table.
4. In the Database Connection Pooling Service field, select the DBCP Connection Pool controller service you created previously.
5. In the Table Name field, specify the name of your remote Snowflake database table. In this example, it is NIFI TEST
6. (Optional) Depending on your requirements, you may want to auto-terminate the success relationship of the processor, and have a self-connecting loop for the failure relationship.

What to do next

After you have configured your target processor, proceed to confirming your data flow success.

Related Information

PutDatabaseRecord

Confirming your data flow success

Confirm that you have successfully built a data flow to push data into Snowflake database tables by starting your data flow and verifying that data is moving through it.

Procedure

1. Select the data flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternatively, right-click a single component and choose Start from the context menu.
3. Go to the Snowflake web interface and confirm that data is correctly ingested into your target table.

Next steps

You now have all the required information to create your NiFi flows to answer your own use cases interacting with the Snowflake data warehousing solution. You can get NiFi up and running and have flows taking care of moving data back and forth between Snowflake and other systems easily and efficiently.

Cloudera DataFlow for Data Hub 7.3.1

Ingesting Data into Apache Solr in Cloudera Public Cloud

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ingesting data into Solr....4
Understand the use case....4
Meet the prerequisites....4
Create Solr target collection....5
Build the data flow....6
Create controller services for your data flow....7
Configure the processor for your data source....8
Configure the processor for your data target....10
Start the data flow....11
Verify data flow operation....11

Monitoring your data flow....12

Next steps....12

Ingesting data into Solr

You can use an Apache NiFi data flow to ingest data into Apache Solr in the Cloudera Public Cloud following these steps.

Understand the use case

Learn how you can use Apache NiFi to move data to Apache Solr in your Data Discovery and Exploration cluster in Cloudera Public Cloud.

Time series use cases analyze data obtained during specified intervals and can improve performance based on available data. These require that you store your documents efficiently in your search engine. You can use Apache NiFi to move data from a range of locations to the Solr service in your Data Discovery and Exploration cluster in Cloudera Public Cloud.

Data Discovery and Exploration clusters provide an efficient way to have Apache Solr up and running to store and index documents that can be queried by external applications. Beyond search queries, this also allows for building insight applications or exploration services based on text or other unstructured data. For more information on how to use Data Discovery and Exploration templates in Cloudera Public Cloud, see the *Discover and Explore Data Faster with the CDP DDE Template* blog post.

This data ingest use case walks you through the steps associated with creating a data flow that moves data from Kafka in a Streams Messaging cluster in Cloudera Public Cloud into Solr in a Data Discovery and Exploration cluster, in the same Cloudera Public Cloud environment. This gets you started with creating an ingest data flow into Solr. If you want to move data from a location other than Kafka, review the *Getting Started with Apache NiFi* for general information about how to build a data flow and about additional data ingest processor options. You can also check the other Cloudera Public Cloud ingest use cases.

Related Information

Discover and explore data faster with the CDP DDE template

Getting started with Apache NiFi

Cloudera Public Cloud ingest use cases

Meet the prerequisites

Use this checklist to make sure that you meet all the requirements before you start building your data flow.

- You have a Cloudera Public Cloud environment.
- You have a workload service account with its username and password set to access Cloudera Data Hub clusters.

  The predefined resource role of this service account is at least EnvironmentUser. This resource role provides the ability to view Cloudera Data Hub clusters and set the FreeIPA password for the environment.
- Your user is synchronized to the Cloudera Public Cloud environment.
- You have a Flow Management cluster in your Cloudera Public Cloud environment running Apache NiFi.
- Your Cloudera user has been added to the appropriate pre-defined Ranger access policies to allow access to the NiFi UI.
- You have a Data Discovery and Exploration cluster running Solr and Hue. This cluster must be in the same Cloudera Public Cloud environment as the Flow Management cluster.

Related Information

Understanding roles and resource roles

Setting up your Flow Management cluster

Data Discovery and Exploration clusters

Create Solr target collection

Learn how you can create collections in Solr. Data is stored in collections, which are single indexes distributed across multiple servers.

Before you begin

Before you can ingest data into Apache Solr, you need a target Solr collection prepared to receive the data. For more information about Solr, see the *Cloudera Search Overview*.

Procedure

1. Navigate to your Data Discovery and Exploration cluster and click Hue in the Services section of the DDE cluster overview page.
2. On the Hue web UI, select Indexes+ Create index.
3. From the Type drop-down, select Manually and click Next.
4. Provide a collection name under Destination.

   In this example, we named the collection solr-nifi-demo.
5. Add the following fields, using + Add Field.

Name	Type
name	text_general
initial_release_date	date

6. Click Submit.
7. To check that the collection has been created, go to the Solr web UI by clicking the Solr Server shortcut in the Services section of the DDE cluster overview page.

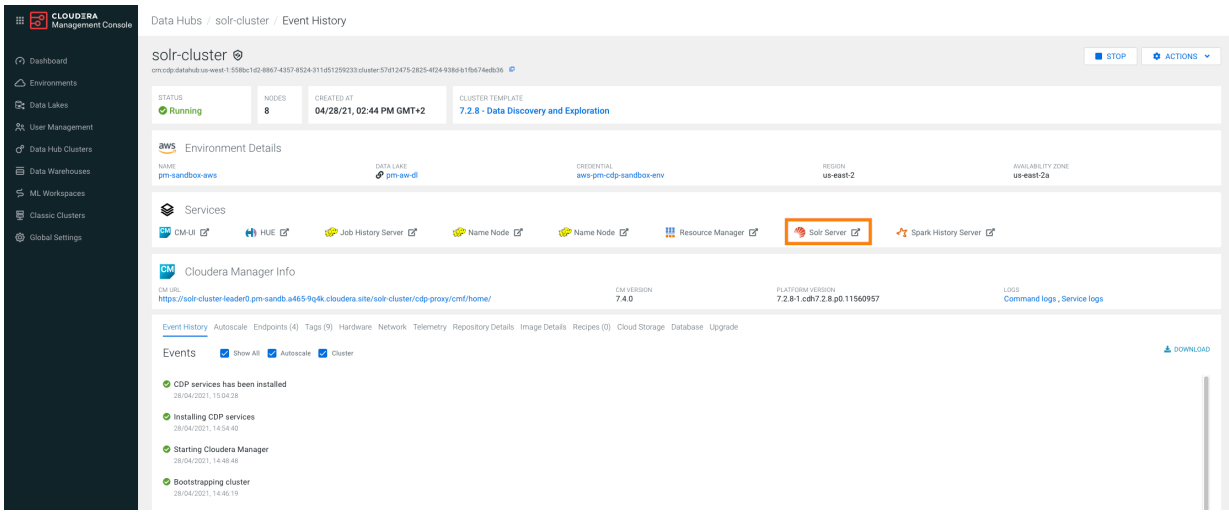

8. Execute the collection query.
   a) Click the Collections sidebar option or click Select an option.

      In the drop-down you will find the collection you have just created. In our example it is solr-nifi-demo.
   b) Click the name of the collection.
   c) Click QueryExecute Query.

      The query output should look like this:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
```

```
    "QTime":0,
    "params":{
      "q":"*:*",
      "doAs":"<querying user>",
      "_forwardedCount":"1",
      "_":"1599835760799"}},
  "response":{"numFound":0,"start":0,"docs":[]
}}
```

Results

You have successfully created an empty collection.

Related Information

Cloudera Search overview

Build the data flow

Learn how you can create an ingest data flow to move data from Apache Kafka to Apache Solr. This involves adding processors and other data flow elements to the NiFi canvas, configuring them, and connecting the elements to create the data flow.

About this task

Use the `PutSolrRecord` processor to build your Solr ingest data flow.

Procedure

1. Open NiFi
   inCloudera Public Cloud.
   a) To access the NiFi service in your Flow Management cluster, navigate to the Cloudera Management Consoleand select Data Hub Clusters.
   b) Click the tile representing the Flow Management cluster you want to work with.
   c) Click NiFi in the Services section of the cluster overview page to access the NiFi UI.

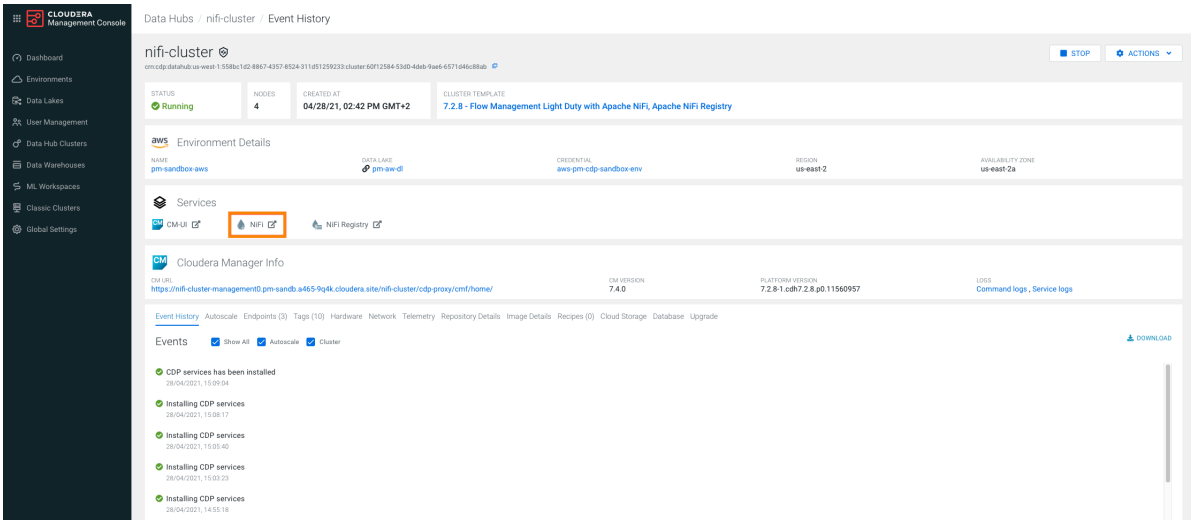

You will be logged into NiFi automatically with your Cloudera credentials.

2. Add the NiFi processors to your canvas.
   a) Select the processor icon from the Cloudera Flow Management actions pane, and drag it to the canvas.
   b) Use the Add Processor filter box to search for the processors you want to add.
   c) Select the following processor and click Add.
      - `ConsumeKafkaRecord_2_0`
      - `PutSolrRecord`
3. Connect the two processors to create the data flow.
   a) Click the connection icon in the first processor, and drag it to the second processor.

      You can see the Create Connection dialog with two tabs: Details and Settings. You can configure the connection's name, flowfile expiration time period, thresholds for back pressure, load balance strategy, and prioritization.
   b) Click Add to close the dialog box and add the connection to your flow.
   c) You can add success and failure funnels to your data flow, which helps you to see where flow files are routed when your data flow is running.

      For more information about working with funnels, see the *Apache NiFi Documentation*.

Results

Your data flow may look similar to the following:

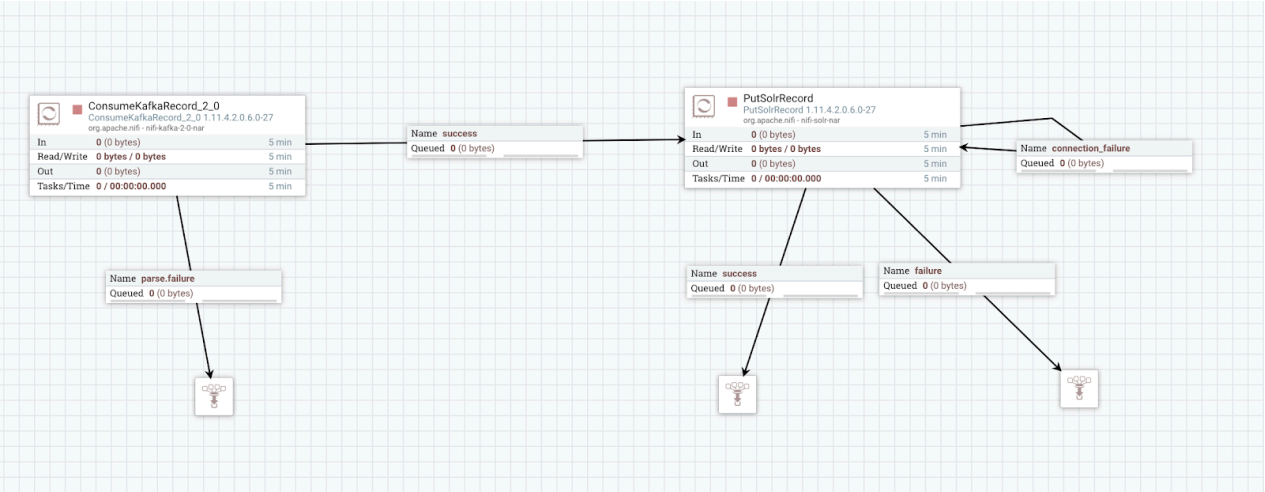

What to do next

Create a controller service for your data flow. You will need this service later as you configure your data flow target processor.

Related Information

Apache NiFi documentation

Create controller services for your data flow

Learn how you can create and configure controller services to provide shared services for the processors in your Solr ingest data flow. You will use these controller services later when you configure your processors.

About this task

Note: You must define the controller services for the processors in your data flow in the configuration of the root process group where they will be used.

For more information on controller services in NiFi, see *Adding Controller Services for Data Flows*.

Procedure

1. To add a controller service to your data flow, right-click the canvas and select Configure from the pop-up menu.

   You can see the Controller Services Configuration window for the process group you are in.
2. Select the Controller Services tab.
3. Click the + button to display the Add Controller Service dialog.
4. Select the required controller service and click Add.

   For this use case, add the RecordReader and RecordWriter controller services to support record-based processing in your data flow.
5. Perform any necessary controller service configuration tasks by clicking the Configure icon in the right-hand column.

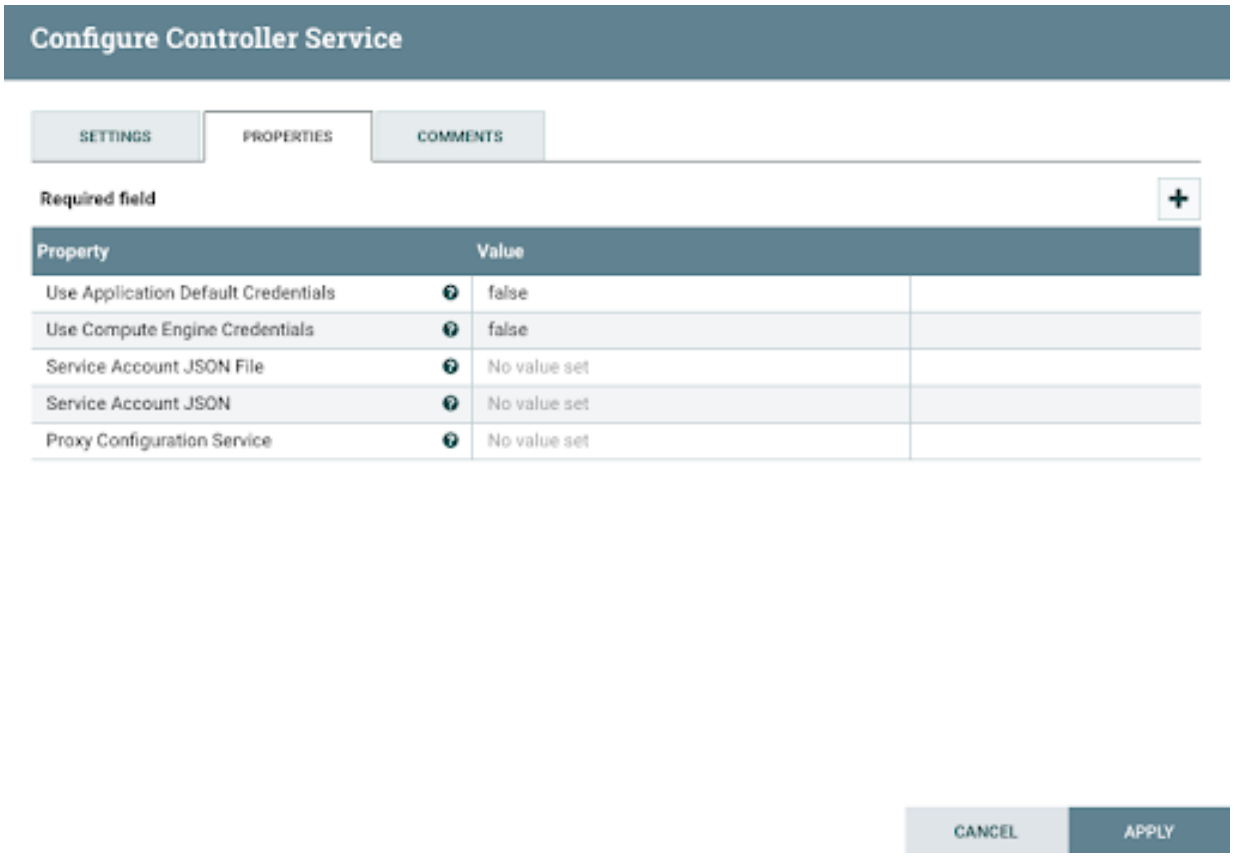

6. When you have finished configuring the options you need, save the changes by clicking Apply.
7. Enable the controller service by clicking the Enable button (flash) in the far-right column of the Controller Services tab.

What to do next

Configure the processors in your data flow.

Related Information

Adding Controller Services for Data Flows

Configure the processor for your data source

You can set up a data flow to move data to Apache Solr from many different locations. This example assumes that you are transferring data from Kafka using the `ConsumeKafkaRecord_2_0` processor.

Before you begin

- Ensure that you have the Ranger policies required to access the Kafka consumer group.
- This use case demonstrates a data flow using data in Kafka and moving it to Solr. To review how to ingest data to Kafka, see *Ingesting Data to Apache Kafka*.

Procedure

1. Launch the Configure Processor window, by right clicking the `ConsumeKafkaRecord_2_0` processor and selecting Configure.

   You can see a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.
   a) Click the Properties tab.
   b) Configure `ConsumeKafkaRecord_2_0` with the required values.

      The following table includes a description and example values for the properties required for this Kafka to Solr ingest data flow. For a complete list of `ConsumeKafkaRecord_2_0` options, see the processor documentation in *Getting Started with Apache NiFi*.

Property	Description	Value to set for ingest to Solr data flow
Kafka Brokers	Provide a comma-separated list of known Kafka Brokers in the format <host>:<port>.	Docs-messaging-broker1.cdf-docs.a465-9q4k.cloudera.site:9093, docs-messaging-broker2.cdf-docs.a465-9q4k.cloudera.site:9093, docs-messaging-broker3.cdf-docs.a465-9q4k.cloudera.site:9093
Topic Name(s)	Enter the name of the Kafka topic from which you want to get data.	customer
Topic Name Format	Specify whether the topics are a comma separated list of topic names or a single regular expression.	names
Record Reader	Specify the record reader to use for incoming flowfiles. This can be a range of data formats.	AvroReader
Record Writer	Specify the format you want to use to serialize data before for sending it to the data target. **Note:** Ensure that the source record writer and the target record reader are using the same schema.	AvroRecordSetWriter
Honor Transactions	Specify whether or not NiFi should honor transactional guarantees when communicating with Kafka.	false
Security Protocol	Specify the protocol used to communicate with brokers. This value corresponds to Kafka's 'security.protocol' property.	SASL_SSL
SASL Mechanism	Specify the SASL mechanism to be used for authentication. This value corresponds to Kafka's 'sasl.mechanism' property.	plain
Username	Specify the username when the SASL Mechanism is PLAIN or SCRAM-SHA-256.	srv_nifi-solr-ingest
Password	Specify the password for the given username when the SASL Mechanism is PLAIN or SCRAM-SHA-256.	Password1!
SSL Context Service	Specify the SSL Context Service to be used for communicating with Kafka.	default

Property	Description	Value to set for ingest to Solr data flow
Group ID	Specify the Group ID used to identify consumers that are within the same consumer group. This value corresponds to Kafka's grou p.id property.	nifi-solr-ingest

3. When you have finished configuring the options you need, save the changes by clicking Apply.

   Note: Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

   If you want to move data to Solr from a location other than Kafka, see Getting Started with Apache NiFi for general information about how to build a data flow and about other data consumption processor options. You can also check the other *Cloudera Public Cloud ingest use cases* for more information.

What to do next

Configure the processor for your data target.

Related Information

Ingesting data into Apache Kafka in Cloudera Public Cloud

Getting started with Apache NiFi

Cloudera Public Cloud ingest use cases

Configure the processor for your data target

You can set up a data flow to move data to many locations. This example assumes that you are moving data to Apache Solr in a Data Discovery and Exploration cluster in Cloudera Public Cloud, using the `PutSolrRecord` processor.

Before you begin

Create a machine user in Cloudera User Management and synchronize this user to your Cloudera environment.

Procedure

1. Launch the Configure Processor window by right clicking the `PutSolrRecord` processor and selecting Configure.

   You can see a configuration dialog with the following tabs: Settings, Scheduling, Properties, Comments.
2. Configure the processor according to the behavior you expect in your data flow.
   a) Click the Properties tab.
   b) Configure `PutSolrRecord` with the required values.

      The following table includes a description and example values for the properties required for this Kafka to Solr ingest data flow. For a complete list of `PutSolrRecord` options, see the processor documentation in *Getting Started with Apache NiFi*.

Property	Description	Value for example Solr ingest data flow
Solr Type	Provide the type of Solr instance. It can be Cloud or Standard.	Cloud
Solr Location	Specify the Solr URL for a StandardSolr type. For example:http://localhost:8984/solr/gettingstarted Specify the ZooKeeper hosts for a Cloud Solr type. For example: localhost:9983). You can find this value on the dashboard of the Solr web UI as the zkHost parameter value.	zookeeper1.cloudera.site:2181,zookeeper2 .cloudera.site:2181,zookeeper3.cloudera.site :2181/solr-dde
Collection	Specify the name of the Solr collection.	solr-nifi-demo

Property	Description	Value for example Solr ingest data flow
Solr Update Path	Provide the path in Solr where flowfile records are posted.	/update
Kerberos Principal	Specify the Cloudera user name you are using to perform this workflow.	Provide the Cloudera user name you created and synced with your Cloudera environment in *Meet the prerequisites*.
Kerberos Password	Specify the password for the CDP user you are using to perform this workflow.	Password1!
RecordReader	Specify the service for reading records from incoming flowfiles.	AvroReader
SSL Context Service	Specify the controller service to use to obtain an SSL context. This property must be set when communicating with Solr over https. Reference the default SSL context service that has been created for you.	In this example the value is: Default NiFi SSL Context Service

Make sure that you set all required properties, as you cannot start the processor until all mandatory properties have been configured.

3. When you have finished configuring the options you need, save the changes by clicking Apply.

   If you want to move data between other locations, see Getting Started with Apache NiFi for general information about how to build a data flow and about other data consumption processor options. You can also check the other *Cloudera Public Cloud ingest use cases* for more information.

What to do next

Your data flow is ready to ingest data into Solr. Start the data flow.

Related Information

Getting started with Apache NiFi

Meet the prerequisites

Start the data flow

When you have built and configured your data flow, you can begin ingesting data into Solr. Learn how to start your Solr ingest data flow.

Procedure

1. Select all the data flow components you want to start.
2. Click the Start icon in the Actions toolbar.

   Alternatively, right-click a single component and choose Start from the context menu.

Results

Data is being read from Kafka and it is written to the Solr target collection you have created.

What to do next

It is useful to check that data is running through the flow without any issues.

Verify data flow operation

Once you have configured and started your data flow, you can verify that you can write data to Solr. There are a number of ways you can check that data is running through the flow and it actually appears in Solr.

Procedure

- Review the Solr ingest data flow on the NiFi canvas and confirm that the NiFi processors are not producing errors.

- From your Data Discovery and Exploration cluster, go to the Hue web UI and search for your data ingest target collection.

  You should see data being ingested into your collection:

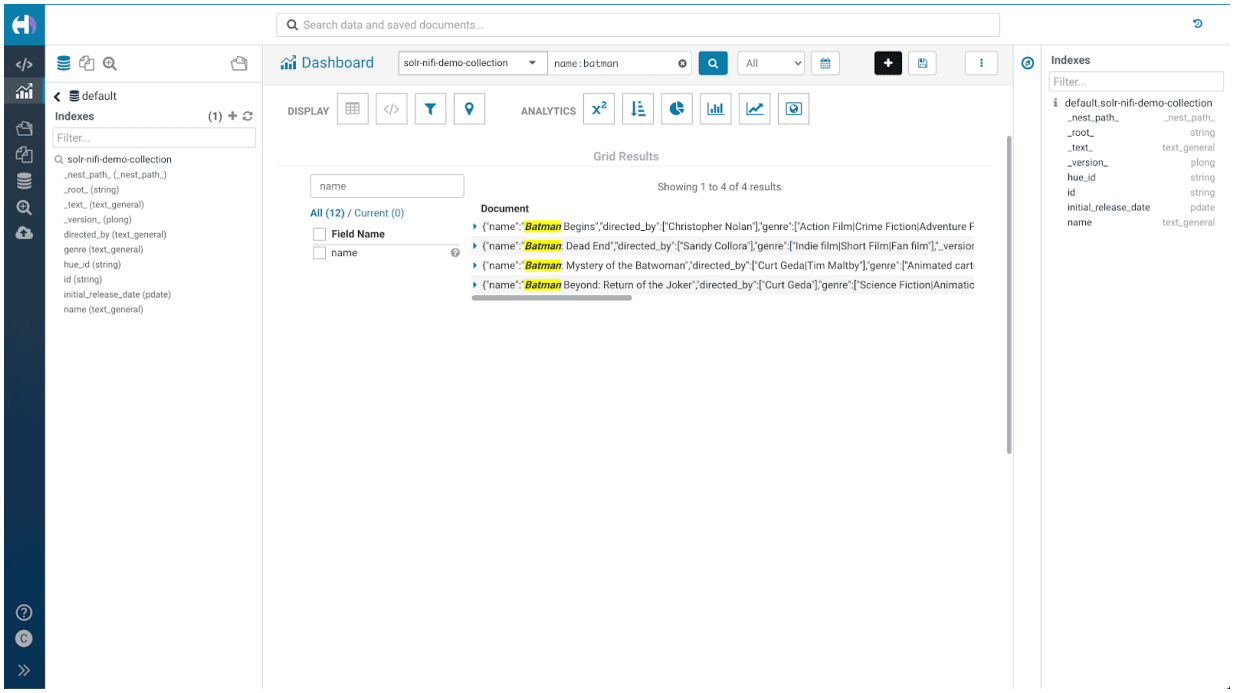

Monitoring your data flow

You can monitor your data flow for information about health, status, and details about the operation of processors and connections. NiFi records and indexes data provenance information, so you can conduct troubleshooting in real time. Learn about the different monitoring options for your Solr ingest data flow in Cloudera Public Cloud.

Data statistics are rolled up on a summary screen (the little table icon on the top right toolbar which lists all the processors). You can use the `MonitorActivity` processor to alert you, if for example you have not received any data in your flow for a specified amount of time.

If you are worried about data being queued up, you can check how much data is currently queued. Process groups also conveniently show the totals for any queues within them. This can often indicate if there is a bottleneck in your flow somewhere, and how far the data has got through that pipeline.

Another option to check that data has fully passed through your flow is to check out data provenance to see the full history of your data.

Next steps

Learn about the different options on what you can do once you have moved data into Solr in Cloudera Public Cloud.

This example data flow shows a basic setup, but it also offers endless opportunities to implement more complex solutions for moving and processing data in Solr. You can explore different data ingest and indexing options where Cloudera components collaborate with each other, while still being resource isolated and managed separately.

For more information on data flows and Solr in Cloudera Public Cloud:

- Review the *NiFi documentation* to learn more about building and managing data flows.
- Review the *NiFi Registry documentation* for more information about versioning data flows.

- Review the *Cloudera Search Overview*.

Related Information

Apache NiFi documentation
Apache NiFi Registry documentation
Cloudera Search overview

Cloudera DataFlow for Data Hub 7.3.1

Planning Your Edge Management Deployment

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Edge Management cluster definitions..... 4

Edge Management cluster layout.....4

Edge Management cluster definitions

There is one Edge Management cluster definition available to deploy Edge Flow Manager in Cloudera Public Cloud: Edge Flow Management Light Duty for AWS.

This cluster definition supports installing Edge Flow Management clusters running Edge Flow Manager. During Technical Preview, Edge Flow Manager is delivered as a 1-node deployment. No heavy duty/clustered deployment is available. Cloudera will provide a heavy duty template with a 3-node Edge Flow Manager cluster in a later release of Cloudera Public Cloud.

Edge Management cluster layout

Learn about the layout, capacity, and components of the Edge Management cluster definitions in Technical Preview.

The Cloudera Data Hub service includes a default cluster definition called Edge Flow Management Light Duty. The cluster definition is available for AWS, Azure, and GCP. You can use it in development, testing, or proof of concept scenarios.

The light duty cluster has the following specifications:

- Single node with a single component:
  - Edge Flow Manager is co-located with Cloudera Manager, running on the management node.
- Instance types for the single node:
  - AWS: m5.2xlarge
  - Azure: D8_v3
  - GCE: e2-standard-8
- Storage requirements:
  - AWS: 100 GB EBS
  - Azure: 100 GB Standard_LRS
  - GCE: 100 GB PD-Standard

You can use the Edge Flow Management: Light Duty cluster definition in development, testing, or proof of concept scenarios.

Note:

When you create an Edge Flow Management Light Duty cluster, a non-HA PostgreSQL RDS instance is provisioned and is used by the Edge Flow Manager.

Cloudera DataFlow for Data Hub 7.3.1

Planning Your Flow Management Deployment

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Flow Management cluster definitions.....4

Flow Management cluster layout.....4

Flow Management cluster definitions

There are six Flow Management cluster definitions available for deploying Apache NiFi and Apache NiFi Registry in Cloudera Public Cloud. Before you select your cluster definition, it is cruical to consider your cloud provider and operational objectives.

Available cluster definitions:

- Flow Management Light Duty for AWS
- Flow Management Light Duty for Azure
- Flow Management Light Duty for GCP
- Flow Management Heavy Duty for AWS
- Flow Management Heavy Duty for Azure
- Flow Management Heavy Duty for GCP

These cluster definitions facilitate the installation of Flow Management clusters that run Apache NiFi and Apache NiFi Registry.

With Flow Management templates, you can address crucial enterprise use cases, offering high-scale data ingestion, transformation, and management across diverse environments. It supports tasks such as data movement, continuous data ingestion, log data processing, and acquiring various streaming data types, including social, mobile, clickstream, and IoT data.

Flow Management cluster layout

The Cloudera Data Hub service provides two default Flow Management cluster definitions: Flow Management: Light Duty and Flow Management: Heavy Duty. Understanding the layout, capacity, and components of these definitions is essential for effective deployment.

Flow Management: Light Duty cluster layout

The Flow Management: Light Duty cluster definition is suitable for development, testing, or proof of concept scenarios.

Note: When you create a Flow Management Light Duty cluster, a non-HA external RDS instance is provisioned and used by NiFi Registry and Schema Registry.

Each cluster node comprises the following components:

- NiFi and ZooKeeper are co-located on all instances.
- Specifications for nodes hosting NiFi and ZooKeeper:
  - AWS: m5.2xlarge
  - Azure: D8_v3
  - GCE: e2-standard-8
- Storage requirements per NiFi node:
  - AWS: 4 x 500GB EBS ST1
  - Azure 4 x 500GB Standard SSD
  - GCE: 4 x 500GB PD-Standard

- Each NiFi node hosts:
  - FlowFile repository
  - Content repository
  - Provenance repository
  - Log and Database repository

For more information, see the *Instance types* and *Storage information* specific to your cloud provider.

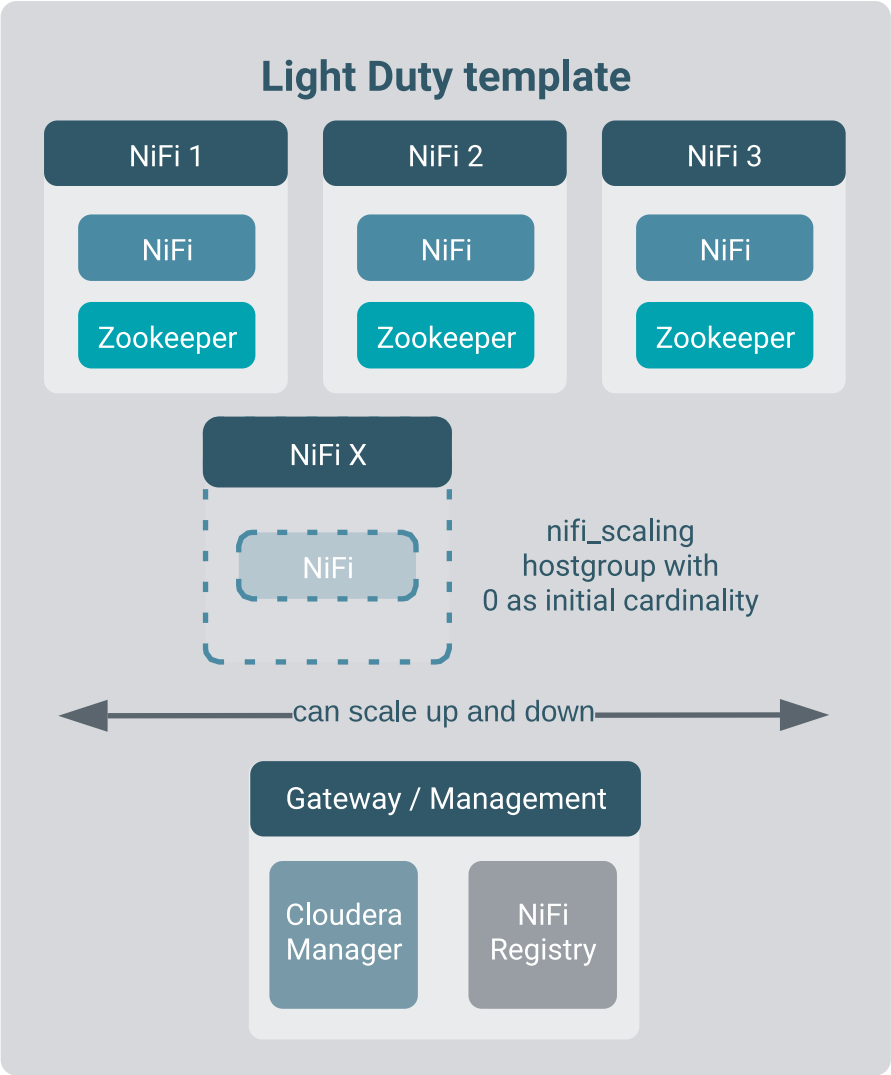

Flow Management: Heavy Duty cluster layout

The Flow Management: Heavy Duty cluster definition is intended for production scenarios.

Note: When you create a Flow Management Heavy Duty cluster, an HA external RDS instance is provisioned and used by NiFi Registry and Schema Registry.

Each cluster node comprises the following components:

- NiFi and ZooKeeper run on separate nodes.
- NiFi nodes scale independently of ZooKeeper.

- Specifications for each ZooKeeper node:
  - AWS: m5.2xlarge
  - Azure: D8_v3
  - GCE: e2-standard-8
- Specifications for each NiFi node:
  - AWS: m5.2xlarge
  - Azure: F16sv2
  - GCE: e2-standard-8
- Storage requirements per NiFi node:
  - AWS: 4x 1TB EBS GP2
  - Azure: 4x 1TB Premium SSD
  - GCE: 4x 1TB PD-SSD
- Each NiFi node hosts:
  - FlowFile repository
  - Content repository
  - Provenance repository
  - Log and Database repository

For more information, see the *Instance types* and *Storage information* specific to your cloud provider.

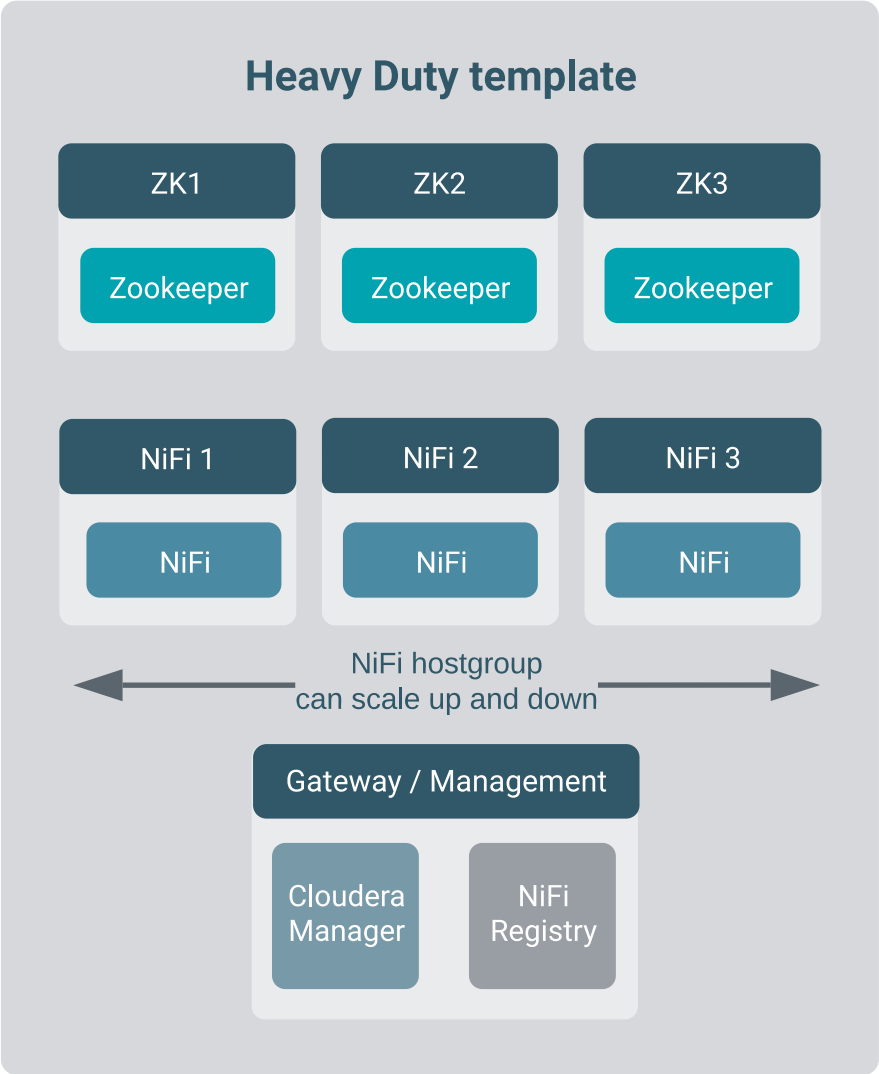

Related Information

AWS instance types
Azure instance types
AWS storage information
Azure storage information

Cloudera DataFlow for Data Hub 7.3.1

Planning your Cloudera Streaming Analytics deployment

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Streaming Analytics Cloudera Data Hub cluster definitons..... 4

Cloudera Streaming Analytics cluster layout.....4

Cloudera Streaming Analytics Cloudera Data Hub cluster definitons

There are four cluster definitions available to deploy Cloudera Streaming Analytics in Cloudera Public Cloud. You can choose from Light and Heavy duty options, and you can further select the cluster definitions depending on your cloud provider.

You can choose from the following template options based on your operational objectives:

- Cloudera Streaming Analytics Light Duty for AWS
- Cloudera Streaming Analytics Light Duty for Azure
- Cloudera Streaming Analytics Light Duty for GCP
- Cloudera Streaming Analytics Heavy Duty for AWS
- Cloudera Streaming Analytics Heavy Duty for Azure
- Cloudera Streaming Analytics Heavy Duty for GCP

Cloudera Streaming Analytics offers real-time stream processing and stream analytics with low-latency and high scaling capabilities powered by Apache Flink.

Cloudera Streaming Analytics templates include Apache Flink that works out of the box in stateless or heavy state environments. Beside Flink, the template includes its supporting services namely YARN, Zookeeper and HDFS. The Heavy Duty template comes preconfigured with RocksDB as state backend, while Light Duty clusters use the default Heap state backend. You can create your streaming application by choosing between Kafka, Kudu, and HBase as datastream connectors.

You can also use SQL to query real-time data with Cloudera SQL Stream Builder in the Cloudera Streaming Analytics template. By supporting the Cloudera SQL Stream Builder service in Cloudera Public Cloud, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. Cloudera SQL Stream Builder is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

In Cloudera Public Cloud, you can use the predefined cluster definitions within your environment to connect Flink and Cloudera SQL Stream Builder with the supported service connectors.

Table 1: Connector support with Cluster Definitions

Cluster Definition	Cloudera SQL Stream Builder Connector[1]	Flink Connector
Streams Messaging	Kafka, Schema Registry	Kafka, Schema Registry
Real-Time Data Mart	Kudu	Kudu
Data Engineering	Hive[2]	Hive
Operational Database	-	HBase

Cloudera Streaming Analytics cluster layout

The Light and Heavy Duty layouts differ from each other based on the volume of storage. You need to choose between the two options based on your operational objectives and application specifications.

[2] You can also use the available Hive service from the Data Lake of the environment.

Cloudera Streaming Analytics: Light Duty cluster layout

You can use a Cloudera Streaming Analytics: Light Duty cluster definition in development and testing scenarios. The Light Duty cluster definition can also be used in production for stateless Flink jobs or for Flink jobs with minimal state. The Light Duty cluster has the following specifications:

- Flink, Cloudera SQL Stream Builder, HDFS, YARN, Zookeeper and Kafka are co-located on all instances
- For each node hosting Flink, Cloudera SQL Stream Builder, HDFS, YARN, Zookeeper and Kafka
  - AWS: m5.2xlarge
  - Azure: Standard_D8_v3
  - GCP: e2-standard-8

For more information, see your cloud provider-specific information about instance types and storage information.

Cloudera Streaming Analytics: Heavy Duty cluster layout

You can use a Cloudera Streaming Analytics: Heavy Duty cluster definition in production for Flink jobs with large state with RockDB as state backend. The Heavy Duty cluster has the following specifications:

- Flink, Cloudera SQL Stream Builder, HDFS, YARN, Zookeeper and Kafka are co-located on all instances
- For each node hosting Flink, Cloudera SQL Stream Builder, HDFS, YARN, Zookeeper and Kafka
  - AWS: m5.2xlarge
  - Azure: Standard_D8_v3
  - GCP: e2-standard-8
- For worker nodes:
  - Storage type: SSD
  - Volume size: 1000 GB

For more information, see your cloud provider-specific information about instance types and storage information.

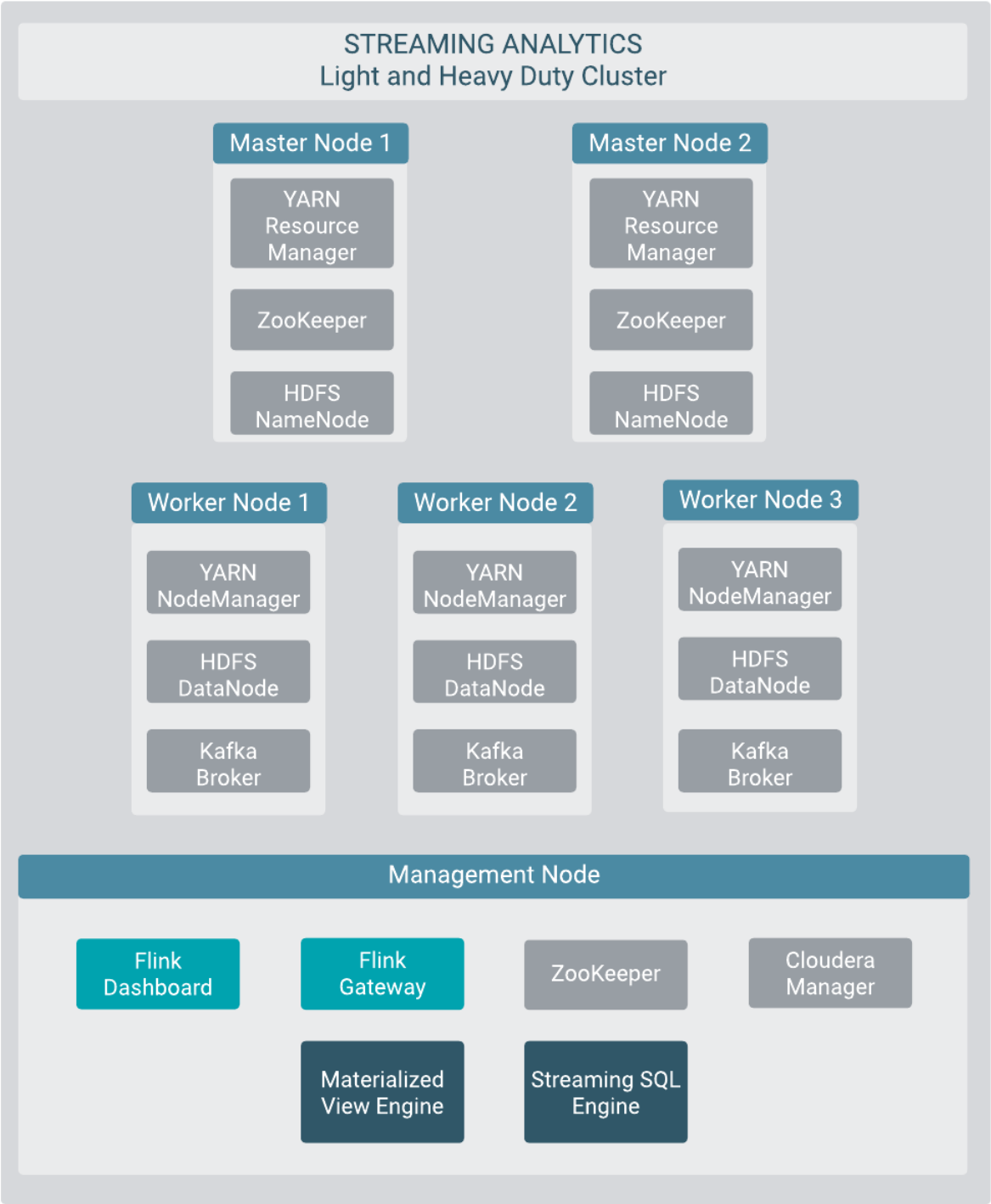

Important: In the Cloudera Streaming Analytics cluster templates, Kafka service is included by default to serve as a background service only for the websocket ouput and sampling feature of Cloudera SQL Stream Builder. The Kafka service in the Cloudera Streaming Analytics cluster template cannot be used for production, you need to use the Cloudera Streams Messaging cluster template when Kafka is needed for your deployment.

Related Information

AWS instance types

Azure instance types
GCP instance types
AWS storage information
Azure storage information
GCP storage information

Cloudera DataFlow for Data Hub 7.3.1

Planning Your Streams Messaging Deployment

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Data Hub cluster definitions 4

Streams Messaging cluster layout 4

Streams Messaging: Light Duty cluster layout 4
Streams Messaging: Heavy Duty cluster layout 7
Streams Messaging: High Availability cluster layout 9

Cloudera Data Hub cluster definitions

The Streams Messaging templates include Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, and ZooKeeper. You may choose from the following template options, depending on your operational objectives:

- Streams Messaging Heavy Duty for AWS
- Streams Messaging Light Duty for AWS
- Streams Messaging High Availability for AWS
- Streams Messaging Heavy Duty for Azure
- Streams Messaging Light Duty for Azure
- Streams Messaging High Availability for Azure (Technical Preview)
- Streams Messaging Heavy Duty for GCP
- Streams Messaging Light Duty for GCP
- Streams Messaging High Availability for GCP (Technical Preview)

Streams Messaging provides advanced messaging and real-time processing on streaming data using Apache Kafka, centralized schema management using Schema Registry, management and monitoring capabilities powered by Streams Messaging Manager, as well as cross-cluster Kafka topic replication using Streams Replication Manager and Kafka partition rebalancing with Cruise Control.

These templates set up fault-tolerant standalone deployments of Apache Kafka and supporting Cloudera components (Schema Registry. Streams Messaging Manager, Streams Replication Manager, and Cruise Control), which can be used for Kafka workloads in the cloud or as a disaster recovery instance for on-premises Kafka clusters.

Streams Messaging cluster layout

The Cloudera Data Hub service includes three default Streams Messaging cluster definitions. These are the Streams Messaging: Light Duty, Streams Messaging: Heavy Duty cluster, and Streams Messaging: High Availability definitions. Learn about the layout, capacity, and components of these definitions.

Note: Kafka KRaft is available in this version of Cloudera but is not ready for production deployment. Cloudera encourages you to explore this technical preview feature in non-production environments and provide feedback on your experiences through the Cloudera Community Forums. For more information regarding KRaft limitations and unsupported features, see Unsupported Streams Messaging features.

Streams Messaging: Light Duty cluster layout

Learn about the layout, capacity, and components of the Streams Messaging: Light Duty cluster definition.

You can use a Streams Messaging: Light Duty cluster definition in development, testing, or proof of concept scenarios. Light Duty clusters include the following nodes and components (services):

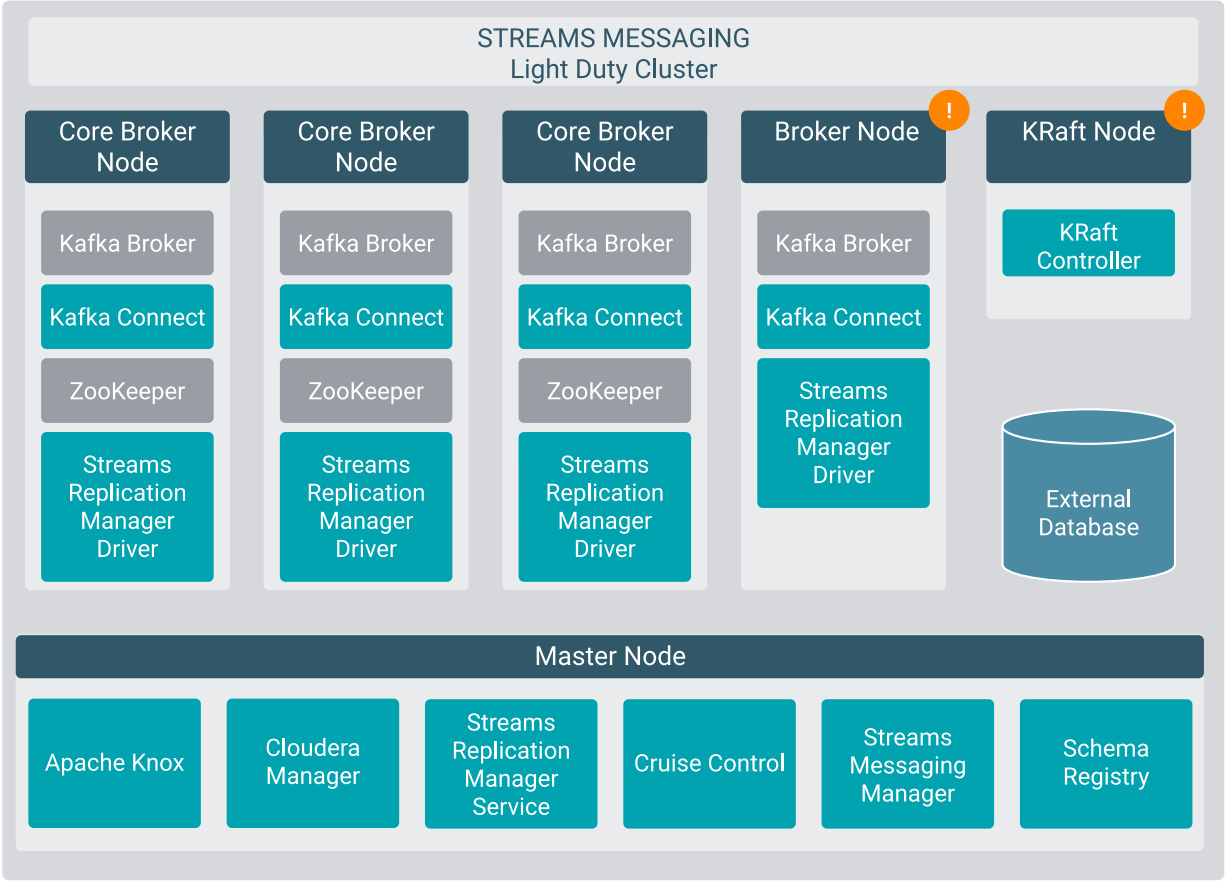

Non-default nodes and scaling

Broker and KRaft nodes are not provisioned by default. You have the option to manually set how many of these nodes are created when provisioning the cluster. After the cluster is provisioned, the number of Broker and KRaft nodes can be changed by scaling your cluster. For more information about scaling KRaft and Broker Nodes, see *Scaling Streams Messaging Clusters*.

KRaft and Zookeeper

By default, Kafka uses Zookeeper as its metadata store. If you provision KRaft nodes in the cluster, Kafka runs in KRaft mode and uses KRaft as its metadata store. In this case, ZooKeeper instances are still provisioned. However, ZooKeeper is not used by Kafka to store and manage metadata. If required, ZooKeeper can be removed from the cluster after the provisioning is finished. ZooKeeper can be removed in Cloudera Manager. For more information, see *Deleting ZooKeeper from Streams Messaging clusters*

You can only provision an odd number of KRaft nodes. This is required so that KRaft can hold a majority election for leadership. While it is possible to run Kafka in KRaft mode with a single KRaft node, Cloudera recommends that you provision a minimum of three to avoid having a single point of failure. Cluster provisioning fails if an even number of KRaft nodes are provisioned.

Broker and Core Broker volume per instance count

By default, the volume per instance count for Broker and Core Broker nodes is identical. If you customize your cluster during provisioning, Cloudera recommends that Attached Volume per Instances is set to the same value for both node types. Alternatively, if you want to provision a cluster where the number of volumes is not identical, ensure that you complete *Configure data directories for clusters with custom disk configurations* after the cluster is

provisioned. Otherwise, Kafka does not utilize all available volumes. Additionally, scaling the cluster might not be possible.

Default instance and storage configuration

The following table collects the default instance type and storage configuration of the various nodes deployed with the Streams Messaging: Light Duty cluster. For more information about the cloud provider-specific instance and storage types, see the *Related Information* section.

For Azure

Table 1: Streams Messaging: Light Duty default hardware configuration in Azure

Node	Instance type	Storage configuration
Master	Standard_E8s_v3	100 GB Standard Locally-redundant SSD storage
Core Broker	Standard_D8s_v3	1 TB Locally-redundant storage
Broker	Standard_D8_v3	1 TB Locally-redundant storage
KRaft	Standard_D8_v3	100 GB Locally-redundant storage

For AWS

Table 2: Streams Messaging: Light Duty default hardware configuration in AWS

Node	Instance type	Storage configuration
Master	r5.2xlarge	100 GB Magnetic
Core Broker	m5.2xlarge	1 TB Throughput Optimized HDD
Broker	m5.2xlarge	1 TB Throughput Optimized HDD
KRaft	m5.2xlarge	100 GB Magnetic

For GCP

Table 3: Streams Messaging: Light Duty default hardware configuration in GCP

Node	Instance type	Storage configuration
Master	e2-highmem-8	100 GB Standard Persistent Disk (HDD)
Core Broker	e2-standard-8	1 TB Standard persistent disks (HDD)
Broker	e2-standard-8	1 TB Standard persistent disks (HDD)
KRaft	e2-standard-8	100 GB Standard Persistent Disk (HDD)

Related Information

Scaling Streams Messaging Clusters
AWS instance types
Azure instance types
GCP instance types
AWS storage information
Azure storage information
GCP storage information
Configure data directories for clusters with custom disk configurations
Deleting ZooKeeper from Streams Messaging clusters

Streams Messaging: Heavy Duty cluster layout

Learn about the layout, capacity, and components of the Streams Messaging: Heavy Duty cluster definition.

You can use the Streams Messaging: Heavy Duty cluster definition in production scenarios. Heavy Duty clusters include the following nodes and components (services):

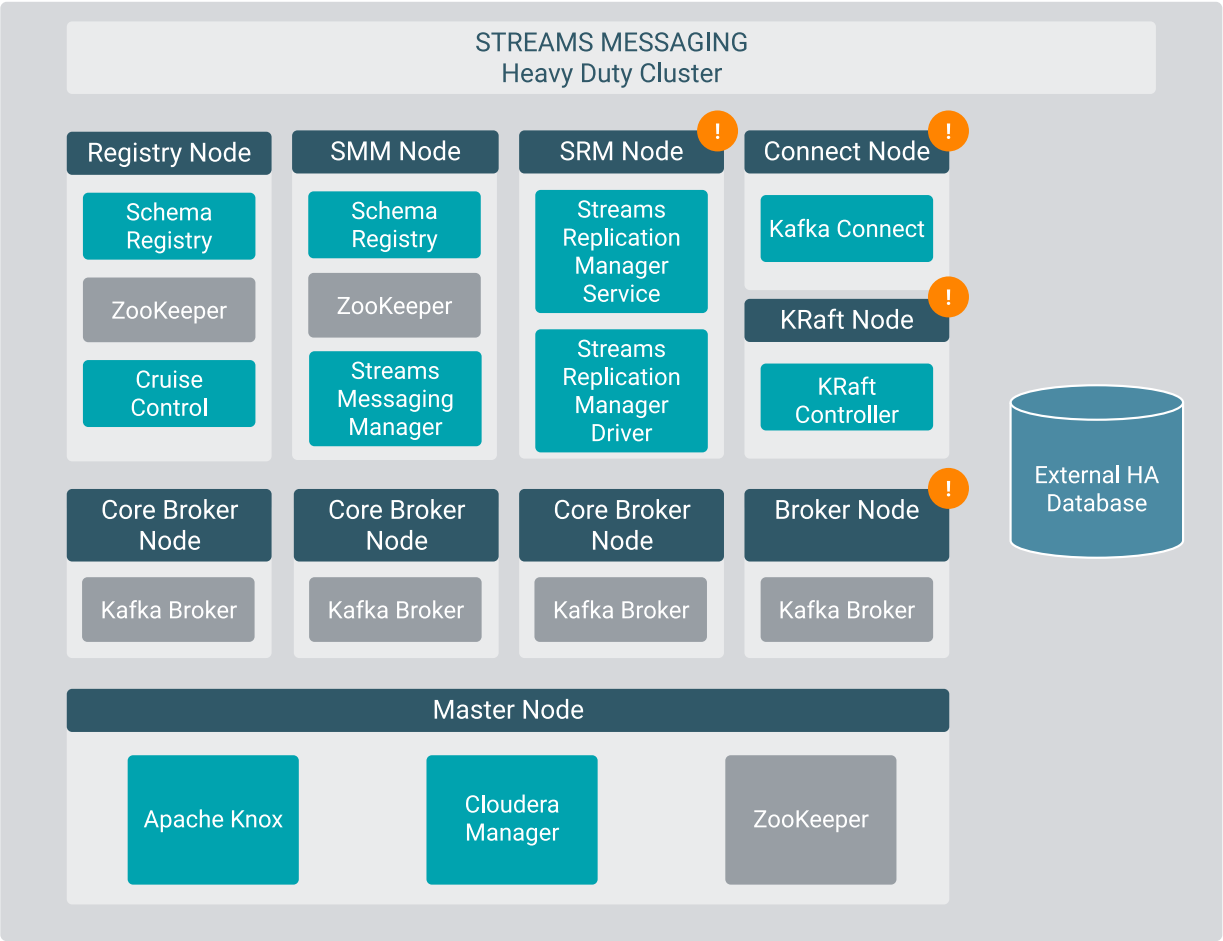

Non-default nodes and scaling

Streams Replication Manager, Broker, Connect, and KRaft nodes are not provisioned by default. If you want to have any of these services provisioned, you must manually set the instance count of the appropriate host group to at least one during cluster provisioning. Otherwise, the host group and its nodes are not provisioned. After a cluster is provisioned, you also have the option to scale these nodes. For more information on scaling, see *Scaling Streams Messaging Clusters*.

KRaft and Zookeeper

By default, Kafka uses Zookeeper as its metadata store. If you provision KRaft nodes in the cluster, Kafka runs in KRaft mode and uses KRaft as its metadata store. In this case, ZooKeeper instances are still provisioned. However, ZooKeeper is not used by Kafka to store and manage metadata. If required, ZooKeeper can be removed from the cluster after the provisioning is finished. ZooKeeper can be removed in Cloudera Manager. For more information, see *Deleting ZooKeeper from Streams Messaging clusters*

You can only provision an odd number of KRaft nodes. This is required so that KRaft can hold a majority election for leadership. While it is possible to run Kafka in KRaft mode with a single KRaft node, Cloudera recommends that you

provision a minimum of three to avoid having a single point of failure. Cluster provisioning fails if an even number of KRaft nodes are provisioned.

Broker and Core Broker volume per instance count

By default, the volume per instance count for Broker and Core Broker nodes is identical. If you customize your cluster during provisioning, Cloudera recommends that Attached Volume per Instances is set to the same value for both node types. Alternatively, if you want to provision a cluster where the number of volumes is not identical, ensure that you complete *Configure data directories for clusters with custom disk configurations* after the cluster is provisioned. Otherwise, Kafka does not utilize all available volumes. Additionally, scaling the cluster might not be possible.

Default instance and storage configuration

The following table collects the default instance type and storage configuration of the various nodes deployed with the Streams Messaging: Heavy Duty cluster. For more information about the cloud provider-specific instance and storage types, see the *Related Information* section.

For Azure

Table 4: Streams Messaging: Heavy Duty default hardware configuration in Azure

Node	Instance type	Storage configuration
Master	Standard_E8s_v3	100 GB Standard Locally-redundant SSD storage
Core Broker	Standard_D8s_v3	1 TB Premium locally-redundant storage
Broker	Standard_D8s_v3	1 TB Premium locally-redundant storage
Registry	Standard_D8_v3	100 GB Locally-redundant storage
SMM	Standard_D8_v3	100 GB Locally-redundant storage
SRM	Standard_D8_v3	100 GB Locally-redundant storage
Connect	Standard_D8_v3	100 GB Locally-redundant storage
KRaft	Standard_D8_v3	100 GB Locally-redundant storage

For AWS

Table 5: Streams Messaging: Heavy Duty default hardware configuration in AWS

Node	Instance type	Storage configuration
Master	r5.2xlarge	100 GB Magnetic
Core Broker	m5.2xlarge	1 TB General Purpose (GP3 SSD)
Broker	m5.2xlarge	1 TB General Purpose (GP3 SSD)
Registry	m5.2xlarge	100 GB Magnetic
SMM	m5.2xlarge	100 GB Magnetic
SRM	m5.2xlarge	100 GB Magnetic
Connect	m5.2xlarge	100 GB Magnetic
KRaft	m5.2xlarge	100 GB Magnetic

For GCP

Table 6: Streams Messaging: Heavy Duty default hardware configuration in GCP

Node	Instance type	Storage configuration
Master	e2-highmem-8	100 GB Standard persistent disks (HDD)
Core Broker	e2-standard-8	1 TB Solid-state persistent disks (SSD)
Broker	e2-standard-8	1 TB Solid-state persistent disks (SSD)
Registry	e2-standard-8	100 GB Standard persistent disks (HDD)
SMM	e2-standard-8	100 GB Standard persistent disks (HDD)
SRM	e2-standard-8	100 GB Standard persistent disks (HDD)
Connect	e2-standard-8	100 GB Standard persistent disks (HDD)
KRaft	e2-standard-8	100 GB Standard persistent disks (HDD)

Related Information

Scaling Streams Messaging Clusters
AWS instance types
Azure instance types
GCP instance types
AWS storage information
Azure storage information
GCP storage information
Configure data directories for clusters with custom disk configurations
Deleting ZooKeeper from Streams Messaging clusters

Streams Messaging: High Availability cluster layout

Learn about the layout, capacity, and components of the Streams Messaging: High Availability cluster definition.

Important: Multi-AZ deployments are not supported for Azure or Google Cloud. As a result, the Streams Messaging High Availability cluster definitions for Azure and Google Cloud are in technical preview. Clusters provisioned with these definitions can only be deployed in a single availability zone.

You can use the Streams Messaging: High Availability cluster definition in production scenarios where having a highly available cluster spanning multiple Availability Zones (multi-AZ) is required. High Availability clusters include the following nodes and components (services):

For V2

Important: The following is the currently available layout for High Availability clusters. This is the layout of all High Availability clusters provisioned after October 30, 2023. See the **V1** tab to view the layout that was available before October 30, 2023.

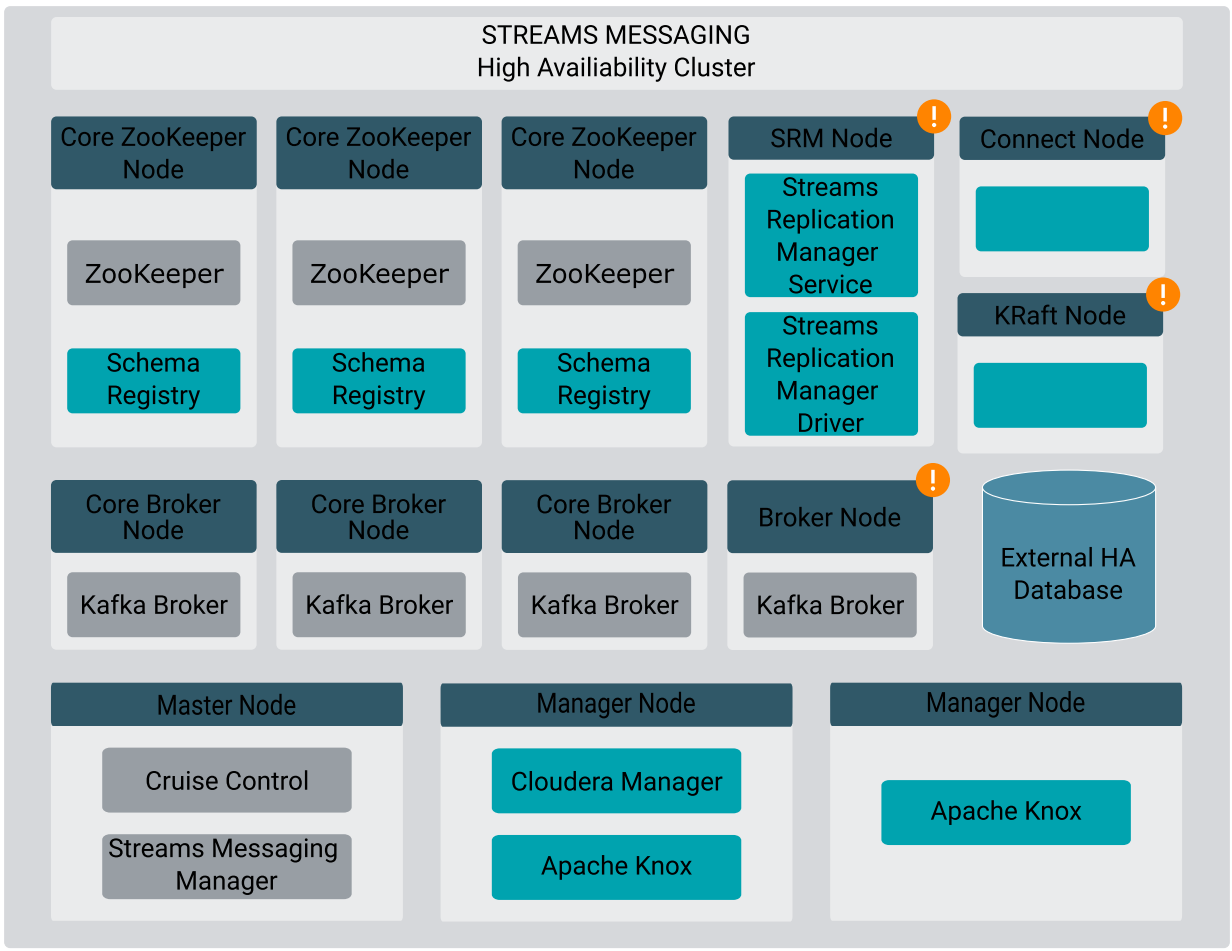

For V1

Important: The following is the layout of High Availability clusters provisioned before October 30, 2023. This layout is no longer available for provisioning. See the **V2** tab to view the currently available layout.

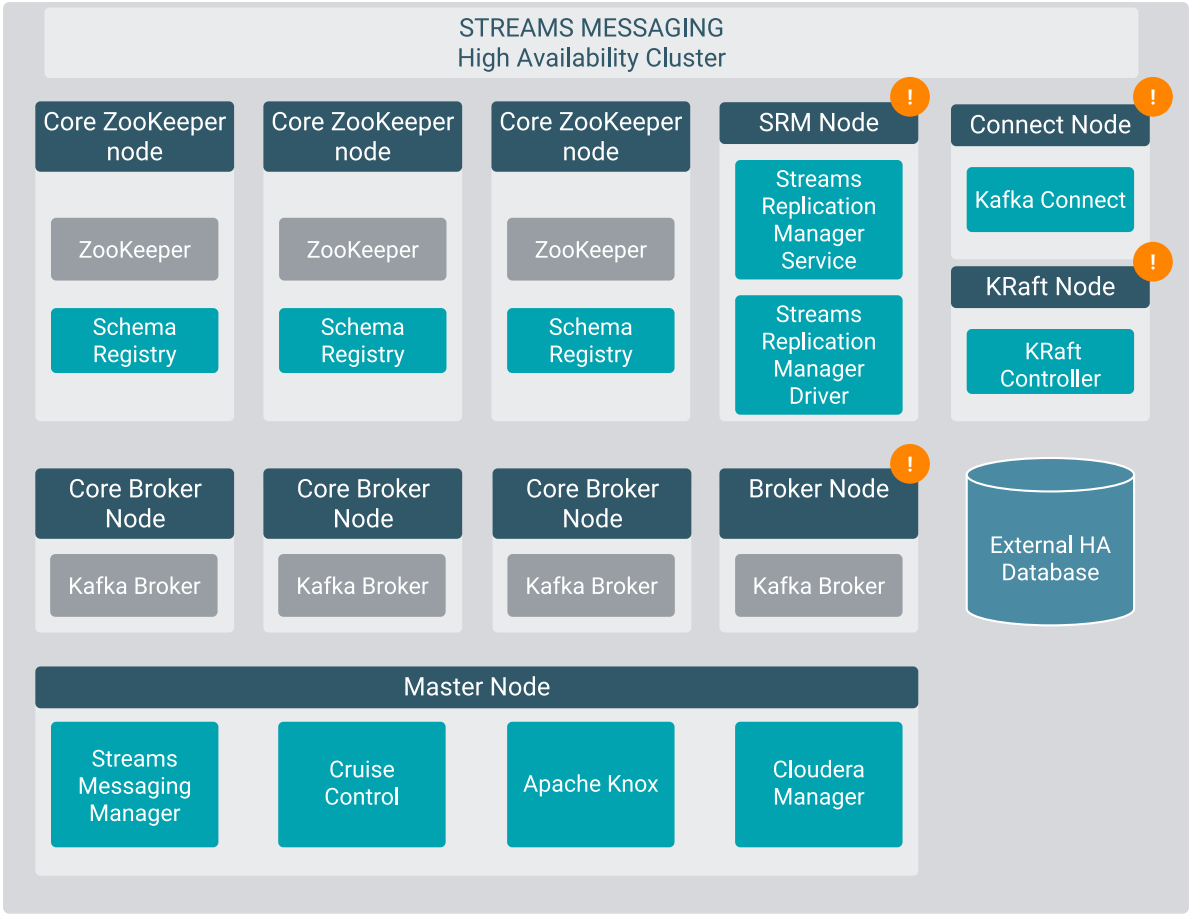

Non-default nodes and scaling

Streams Replication Manager, Broker, Connect, and KRaft nodes are not provisioned by default. If you want to have any of these services provisioned, you must manually set the instance count of the appropriate host group to at least one during cluster provisioning. Otherwise, the host group and its nodes are not provisioned. After a cluster is provisioned, you also have the option to scale these nodes. For more information on scaling, see *Scaling Streams Messaging Clusters*.

Use multiple subnets for high availability

When using the Streams Messaging High Availability definition, ensure that you select multiple subnets when provisioning the cluster. Otherwise, your cluster will not be highly available.

KRaft and Zookeeper

By default, Kafka uses Zookeeper as its metadata store. If you provision KRaft nodes in the cluster, Kafka runs in KRaft mode and uses KRaft as its metadata store. In this case, ZooKeeper instances are still provisioned. However, ZooKeeper is not used by Kafka to store and manage metadata. If required, ZooKeeper can be removed from the cluster after the provisioning is finished. ZooKeeper can be removed in Cloudera Manager. For more information, see *Deleting ZooKeeper from Streams Messaging clusters*

You can only provision an odd number of KRaft nodes. This is required so that KRaft can hold a majority election for leadership. While it is possible to run Kafka in KRaft mode with a single KRaft node, Cloudera recommends that you provision a minimum of three to avoid having a single point of failure. Cluster provisioning fails if an even number of KRaft nodes are provisioned.

Broker and Core Broker volume per instance count

By default, the volume per instance count for Broker and Core Broker nodes is identical. If you customize your cluster during provisioning, Cloudera recommends that Attached Volume per Instances is set to the same value for both node types. Alternatively, if you want to provision a cluster where the number of volumes is not identical, ensure that you complete *Configure data directories for clusters with custom disk configurations* after the cluster is provisioned. Otherwise, Kafka does not utilize all available volumes. Additionally, scaling the cluster might not be possible.

Default instance and storage configuration

The following table collects the default instance type and storage configuration of the various nodes deployed with the Streams Messaging: High Availability cluster. For more information about the cloud provider-specific instance and storage types, see the *Related Information* section.

For Azure

Table 7: Streams Messaging: High Availability default hardware configuration in Azure

Node	Instance type	Storage configuration
Master	Standard_E16s_v3	100 GB Standard Locally-redundant SSD storage
Manager [1]	Standard_D16_v3	100 GB Locally-redundant storage
Core Broker	Standard_D8s_v3	1 TB Premium locally-redundant storage
Broker	Standard_D8s_v3	1 TB Premium locally-redundant storage
Core ZooKeeper	Standard_D8s_v3	100 GB Locally-redundant storage
SRM	Standard_D8_v3	100 GB Locally-redundant storage
Connect	Standard_D8_v3	100 GB Locally-redundant storage
KRaft	Standard_D8_v3	100 GB Locally-redundant storage

For AWS

Table 8: Streams Messaging: High Availability default hardware configuration in AWS

Node	Instance type	Storage configuration
Master	r5.4xlarge	100 GB Magnetic
Manager [1]	m5.4xlarge	100 GB Magnetic
Core Broker	m5.2xlarge	1 TB General Purpose (GP3 SSD)
Broker	m5.2xlarge	1 TB General Purpose (GP3 SSD)
Core ZooKeeper	m5.2xlarge	100 GB Magnetic
SRM	m5.2xlarge	100 GB Magnetic
Connect	m5.2xlarge	100 GB Magnetic
KRaft	m5.2xlarge	100 GB Magnetic

For GCP

[1] The Manager node is only available on clusters provisioned after October 30, 2023.

Table 9: Streams Messaging: High Availability default hardware configuration in GCP

Node	Instance type	Storage configuration
Master	e2-highmem-16	100 GB Standard persistent disks (HDD)
Manager [1]	e2-standard-16	100 GB Standard persistent disks (HDD)
Core Broker	e2-standard-8	1 TB Solid-state persistent disks (SSD)
Broker	e2-standard-8	1 TB Solid-state persistent disks (SSD)
Core ZooKeeper	e2-standard-8	100 GB Standard persistent disks (HDD)
SRM	e2-standard-8	100 GB Standard persistent disks (HDD)
Connect	e2-standard-8	100 GB Standard persistent disks (HDD)
KRaft	e2-standard-8	100 GB Standard persistent disks (HDD)

Related Information

Scaling Streams Messaging Clusters
AWS instance types
Azure instance types
GCP instance types
AWS storage information
Azure storage information
GCP storage information
Configure data directories for clusters with custom disk configurations
Deleting ZooKeeper from Streams Messaging clusters

Cloudera DataFlow for Data Hub 7.3.1

Setting up your Edge Management cluster

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Checking prerequisites **4**

Creating your cluster **4**

After creating your cluster **6**

Generating certificates for MiNiFi agents 6

Adding agents to your cluster 10

Checking prerequisites

Before you start creating your Edge Flow Management Data Hub cluster, you need to ensure that you have set up the environment properly and have all the necessary accesses to use Cloudera Public Cloud. Use this checklist to verify that you meet all the requirements before you start creating the cluster.

- You have Cloudera login credentials.
- You have an available Cloudera environment.

  When you register your environment, make sure that the correct security access settings are configured. You need to enable SSH access and specify SSH key so that you can generate certificates for the agents. For more information on creating a Cloudera environment, see:

  - Working with AWS environments
  - Working with Azure environments
  - Working with GCP environments
- You have a running Data Lake. For more information on the Data Lake service in a Cloudera environment, see Introduction to Data Lakes.

  Important: Make sure that the Runtime version of the Data Lake cluster matches the Runtime version of the Cloudera Data Hub cluster that you are about to create. If these versions do not match, you may encounter warnings and/or errors.
- You have a Cloudera username and the predefined resource role of this user is EnvironmentAdmin.
- Your Cloudera user is synchronized to the Cloudera Public Cloud environment.

If you need more information about Cloudera basics, see Getting started as a user.

Creating your cluster

If you meet all prerequisites, you are ready to create a managed and secured Edge Flow Management cluster in Cloudera Public Cloud by using the prescriptive cluster definition available in Technical Preview.

Procedure

1. Log into the Cloudera Public Cloud web interface.
2. Navigate to Cloudera Management Console Environments and select the environment where you want to create a cluster.

3. Click Create Data Hub.

   The Provision Data Hub page is displayed:

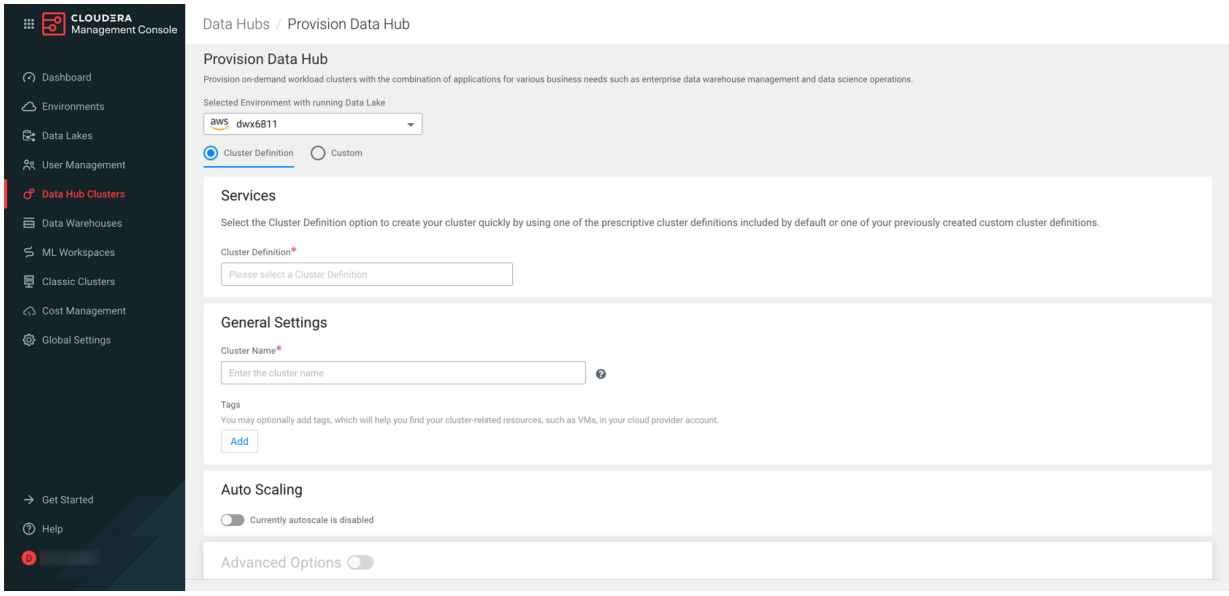

4. Select Cluster Definition.
5. Select the appropriate Edge Flow Management cluster definition from the Cluster Definition dropdown depending on which cloud privider you are using.

   There are three template options available:

   - Edge Flow Management Light Duty for AWS
   - Edge Flow Management Light Duty for Azure
   - Edge Flow Management Light Duty for GCP

   The cluster template referenced in the selected cluster definition determines which services are included in the cluster. The list of services is automatically displayed below the selected cluster definition name. It shows that the cluster definition contains the Edge Flow Manager.
6. Provide a cluster name and add tags you might need.

   Note:

   The name must be between 5 and 40 characters, it must start with a letter, and should only include lowercase letters, numbers, and hyphens.

   You can define tags that will be applied to your cluster-related resources on your cloud provider account. For more information, see *Tags*.
7. Use the Configure Advanced Options section to customize the infrastructure settings.

   For more information on these options, see the *Advanced cluster options* for your cloud environment.
8. Click Provision Cluster.

Results

The new Cloudera Data Hub cluster appears on the Data Hubs tab of the Clusters page. You can follow the status of the provisioning process in the Status column. When your cluster is ready, its status changes to Running.

After creating your cluster

The cluster you have created using the Edge Flow Management cluster definition is secured by default, and it is integrated with Knox SSO.

You can access the Edge Flow Manager UI from the Services section of the Data Hub cluster page. Click the CEM icon or the Edge Flow Manager UI link and you are redirected to the Edge Flow Manager page.

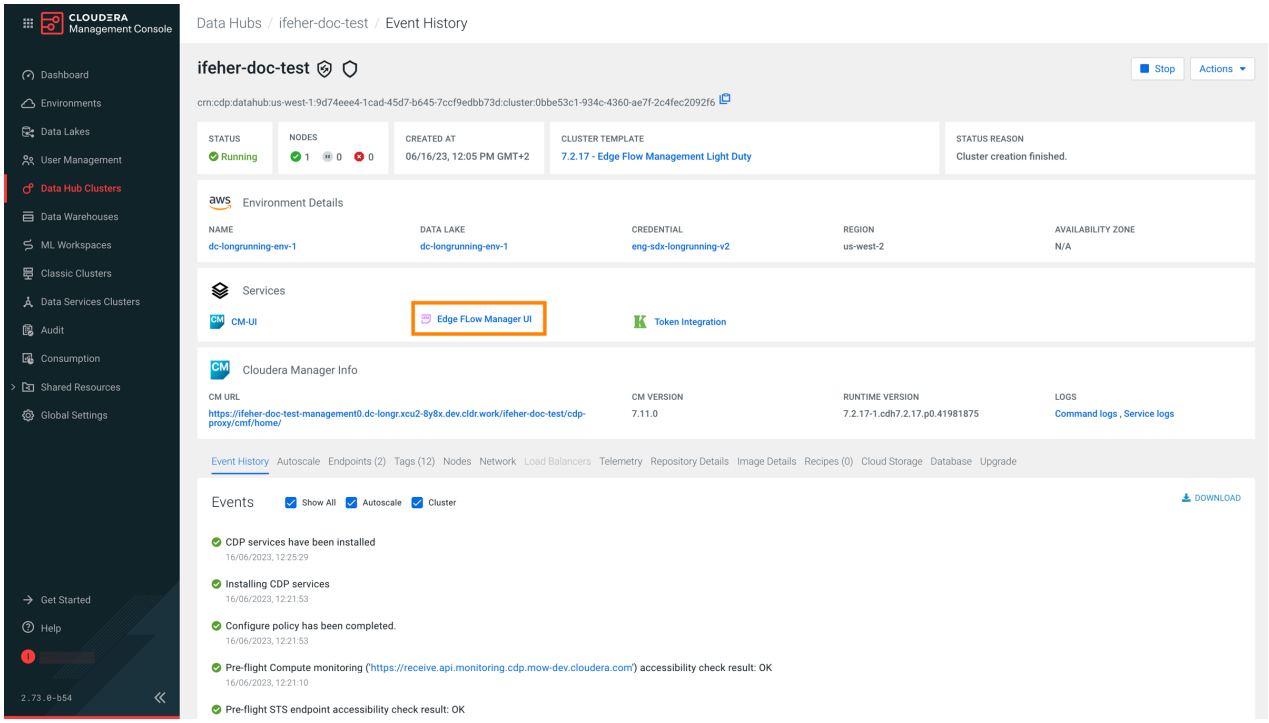

The user who creates the Cloudera Data Hub cluster is automatically designated as an administrator in Edge Flow Manager and gains immediate access to the UI.

Edge Flow Manager now integrates with Cloudera User Management and synchronizes all available users and groups. Besides the cluster creator user, all users who belong to the admin group also gain administrator rights, providing access to all Edge Flow Manager features. Users who do not belong to the admin group can still log in, but need access rights granted by an administrator before they can access the data in Edge Flow Manager.

To secure the agent-to-Edge Flow Manager communication, generate and use appropriate certificates. You also need to add the agents that you want to manage with Edge Flow Manager.

Generating certificates for MiNiFi agents

To secure the communication between agents and Edge Flow Manager, you need to generate and use proper certificates.

About this task

Edge Flow Manager is a secured application, which has to be bootstrapped with the initial admin identity. The initial admin is the person who is able to assign roles and manage permissions in Edge Flow Manager. In the Technical Preview, the initial admin is the workload user of the person who deploys the Cloudera Data Hub cluster. For more information about authentication and authorization, see *Access control bootstrapping*.

While the user traffic accessing the UI utilizes Knox, the agents running outside of the Cloudera deployment need to access Edge Flow Manager directly. To enable this, you have to open a port for the agents on the host where Edge

Flow Manager is deployed. By default, this port is 10090, used by Cloudera Edge Management components for C2 Protocol.

You do not have to generate the certificates from the agent host. You can generate them on any host that has access to the management node. When created, you can copy the certificates to the appropriate agent host.

In test environments it is not necessary to create different certificates for all agents. The same certificate can be configured for all agents. However, in production environments it is highly recommended to create a certificate for each agent.

Generating certificates with this approach is similar to adding a node to the cluster using Cloudera Manager.

Note: Agents using these certificates are considered to be the members of the cluster managed by Cloudera Manager. Use your certificates with care and protect them from illegal access.

MiNiFi agents need to set up mutual TLS (mTLS) for C2 communication to be able to communicate with Edge Flow Manager. For information on MiNiFi Java Agent authentication, see *Securing MiNiFi Java Agent*. For information on MiNiFi C++ Agent authentication, see *Securing MiNiFi C++ Agent*.

In Cloudera Public Cloud, certificates are managed by Cloudera Manager, acting as a certificate authority. All certificates are generated by Cloudera Manager, there is no option to use custom certificates.

Note:

In the Technical Preview version of Cloudera Edge Management for Cloudera Public Cloud, you have to set up agent security manually. In later versions there will be an option to set up agent security using Edge Flow Manager.

Before you begin

- You have a running Cloudera Edge Management Public Cloud cluster
- SSH access is configured to the management node of the cluster
- You have an SSH user with keypair that has sudo privileges
- You have the host name of the Edge Management cluster's management node
- An external node is available from which you are able to SSH into the Edge Management cluster's management node

Procedure

1. Create a working directory on your external node that has SSH access to your Edge Flow Management cluster.
2. Save the following script to the previously created working directory, and name it create_certs.sh.

```
#!/bin/bash
set -eo pipefail

# input parameters
SSH_USER=$1
SSH_KEY=$2
CM_HOST=$3
AGENT_FQDN=$4

EXAMPLE_USAGE="Example usage: ./create_certs.sh sshUserName ~/.ssh/userKe
y.pem host0.company.site agent-x.company.site"

[[ -z "$SSH_USER" ]] && echo "SSH User parameter is missing. $EXAMPLE_US
AGE" && exit 1
[[ -z "$SSH_KEY" ]] && echo "SSH Key parameter is missing. $EXAMPLE_USAGE"
 && exit 1
[[ -z "$CM_HOST" ]] && echo "Cloudera Manager parameter is missing. $EXA
MPLE_USAGE" && exit 1
[[ -z "$AGENT_FQDN" ]] && echo "Agent FQDN parameter is missing. $EXAMPLE_
USAGE" && exit 1
```

```
KEYSTORE_PASSWORD=$(hexdump -vn16 -e'4/4 "%08X" 1 "\n"' /dev/urandom | tr
'[:upper:]' '[:lower:]')

# constants
GENERATED_CREDENTIALS_ARCHIVE=credentials.tar
GENERATED_CREDENTIALS_REMOTE_PATH="/tmp/$GENERATED_CREDENTIALS_ARCHIVE"
CM_SITE_PACKAGES="/opt/cloudera/cm-agent/lib/python3.8/site-packages"
ORIGINAL_CERTMANAGER_BASE_DIR="/etc/cloudera-scm-server/certs"
CUSTOM_CERTMANAGER_BASE_DIR="/root/certs"
CERT_PASSWORDS_DIR="$CUSTOM_CERTMANAGER_BASE_DIR/private"
GLOBAL_KEY_PASSWORD_FILE="$CERT_PASSWORDS_DIR/.global_key_password"
GLOBAL_TRUSTSTORE_PASSWORD_FILE="$CERT_PASSWORDS_DIR/.global_truststore
_password"

rm -rf "$AGENT_FQDN"
mkdir "$AGENT_FQDN"

remote_ssh_command=$(cat << EOF
sudo \cp -n -R $ORIGINAL_CERTMANAGER_BASE_DIR $CUSTOM_CERTMANAGER_BASE_DI
R;
sudo /opt/rh/rh-python38/root/bin/python -c "import site; site.addsitedir
('$CM_SITE_PACKAGES'); import cmf.tools.cert; passwd = cmf.tools.cert.re
ad_obfuscated_password('$GLOBAL_TRUSTSTORE_PASSWORD_FILE'); print(passwd
);"
sudo rm -f $GLOBAL_KEY_PASSWORD_FILE;
sudo /opt/rh/rh-python38/root/bin/python -c "import site; site.addsitedir(
'$CM_SITE_PACKAGES'); import cmf.tools.cert; cmf.tools.cert.write_obfusc
ated_password('$GLOBAL_KEY_PASSWORD_FILE', '$KEYSTORE_PASSWORD');";
sudo /opt/cloudera/cm-agent/bin/certmanager --location "$CUSTOM_CERTMANA
GER_BASE_DIR" gen_node_cert --output "$GENERATED_CREDENTIALS_REMOTE_PATH"
 --rotate "$AGENT_FQDN";
sudo chmod 666 "$GENERATED_CREDENTIALS_REMOTE_PATH";
EOF
)

ssh -i "$SSH_KEY" -o StrictHostKeyChecking=no "$SSH_USER"@"$CM_HOST" "$rem
ote_ssh_command" > "$AGENT_FQDN/cm-auto-in_cluster_trust.pw" 2> /dev/null
scp -r -i "$SSH_KEY" -o "StrictHostKeyChecking=no" "$SSH_USER"@"$CM_HOST":
"$GENERATED_CREDENTIALS_REMOTE_PATH" "$AGENT_FQDN/" 2> /dev/null
tar -xf "$AGENT_FQDN/$GENERATED_CREDENTIALS_ARCHIVE" -C "$AGENT_FQDN"
echo "MiNiFi-Java KeyStore File":
ls -alh "$AGENT_FQDN/cm-auto-host_keystore.jks"
echo "MiNiFi-Java TrustStore File:"
ls -alh "$AGENT_FQDN/cm-auto-in_cluster_truststore.jks"
echo "MiNiFi-CPP Client certificate":
ls -alh "$AGENT_FQDN/cm-auto-host_key_cert_chain.pem"
echo "MiNiFi-CPP Client private key":
ls -alh "$AGENT_FQDN/cm-auto-host_key.pem"
echo "MiNiFi-CPP CA certificate"
ls -alh "$AGENT_FQDN/cm-auto-in_cluster_ca_cert.pem"
echo "KeyStore / HostKey Password: sensitive data, please check for it in
 $AGENT_FQDN/cm-auto-host_key.pw"
echo "TrustStore Password: sensitive data, please check for it in $AGENT_
FQDN/cm-auto-in_cluster_trust.pw"

rm -f "$AGENT_FQDN/cm-auto-global_cacerts.pem" "$AGENT_FQDN/cm-auto-globa
l_truststore.jks" "$AGENT_FQDN/$GENERATED_CREDENTIALS_ARCHIVE" "$AGENT_F
QDN/cm-auto-host_cert_chain.pem"
```

3. Make the script executable.

```
chmod +x create_certs.sh
```

4. Run the script with the following parameters:

```
./create_certs.s
h **[SSH_USER}** **[SSH_PRIVATE_KEY]** **[MANAGEMENT_NODE_HOST_NAME]** **[AGENT_FQDN]
```

For example:

```
./create_certs.sh adminuser ~/.ssh/adminuser.pem management-node.company
.site.com agent-1.company.site.com
```

The script should print a similar output:

```
credentials.tar

                                                      100%  420KB 222.0KB/s
   00:01
MiNiFi-Java KeyStore File:
-rw-------@ 1 user  group   5.2K Apr 24 13:33 agent-1.company.site.com/cm-
auto-host_keystore.jks
MiNiFi-Java TrustStore File:
-rw-r-----@ 1 user  group   2.3K Apr 24 13:19 agent-1.company.site.com/cm-
auto-in_cluster_truststore.jks
MiNiFi-CPP Client certificate:
-rw-------@ 1 user  group   7.1K Apr 24 13:33 agent-1.company.site.com/cm-
auto-host_key_cert_chain.pem
MiNiFi-CPP Client private key:
-rw-------@ 1 user  group   2.5K Apr 24 13:33 agent-1.company.site.com/cm-
auto-host_key.pem
MiNiFi-CPP CA certificate
-rw-r-----@ 1 user  group   3.0K Apr 24 13:19 agent-1.company.site.com/cm-
auto-in_cluster_ca_cert.pem
KeyStore / HostKey Password: sensitive data, please check for it in agent-
1.company.site.com/cm-auto-host_key.pw
TrustStore Password: sensitive data, please check for it in agent-1.compa
ny.site.com/cm-auto-in_cluster_trust.pw
```

A directory is created with the same name as the agent's FQDN, provided as a parameter for the script. The directory contains all the necessary keystores and certificates for configuring mTLS authentication.

The keystore and truststore passwords are not printed as they are sensitive information. You can find them in the directory that was created with the following names:

- cm-auto-host_key.pw
- cm-auto-in_cluster_trust.pw

5. Set the agent parameters.

- For the MiNiFi Java Agent:

```
c2.security.truststore.location=/path/to/cm-auto-in_cluster_truststore.j
ks
c2.security.truststore.password=<password_from_cm-auto-in_cluster_tru
st.pw>
c2.security.truststore.type=JKS
c2.security.keystore.location=/path/to/cm-auto-host_keystore.jks
c2.security.keystore.password=<password_from_cm-auto-host_key.pw>
c2.security.keystore.type=JKS
```

- For the MiNiFi C++ Agent:

```
nifi.security.client.certificate=/path/to/cm-auto-host_key_cert_chain.pe
m
nifi.security.client.private.key=/path/to/cm-auto-host_key.pem
```

```
nifi.security.client.pass.phrase=/path/to/cm-auto-host_key.pw
nifi.security.client.ca.certificate=/path/to/cm-auto-in_cluster_ca_ce
rt.pem
```

Note:

Although the parameter is called Agent FQDN, it is not mandatory to use the agent's domain name. You can use any other string. Keep in mind that the string you provide will be the common name (CN) in the generated certificate.

Adding agents to your cluster

When your cluster has been created successfully, you can add agents that you want to manage with Edge Flow Manager. Agents are deployed outside of Cloudera Public Cloud, so follow the standard agent deployment instructions:

Java Agents

Installing the MiNiFi Java Agent

C++ Agents

Installing the MiNiFi C++ Agent

Note:

Make sure that you point the agents to heartbeat to the Cloudera Data Hub Edge Flow Manager deployment.

Cloudera DataFlow for Data Hub 7.3.1

Setting up your Flow Management cluster

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Checking prerequisites 4

Creating your cluster 4

Giving access to your cluster 6

Checking prerequisites

Before you start creating your Flow Management Data Hub Cluster, you need to ensure that you have set up the environment properly and have all the necessary accesses to use Cloudera Public Cloud. Use this checklist to check that you meet all the requirements before you start creating the cluster. If you need more information about Cloudera basics, see *Getting started as a user*.

- You have Cloudera login credentials.
- You have an available Cloudera environment.

  Note:

  If you want to deploy your Flow Management Data Hub cluster on multiple availability zones (use multi-AZ support), make sure that multiple subnets have been selected during environment creation. In the Network section of the Region, Networking and Security page, select as many subnets as needed.

  For more information on creating a Cloudera environment, see:

  - *Working with AWS environments*
  - *Working with Azure environments*
  - *Working with GCP environments*
- You have a running Data Lake.

  Important: Make sure that the Runtime version of the Data Lake cluster matches the Runtime version of the Data Hub cluster that you are about to create. If these versions do not match, you may encounter warnings and/or errors.

  For more information on the Data Lake service in Cloudera environment, see *Introduction to Data Lakes*.
- You have a Cloudera username and the predefined resource role of this user is EnvironmentAdmin.
- Your Cloudera user is synchronized to the Cloudera Public Cloud environment.

Related Information

Getting started as a user

Working with AWS environments

Working with Azure environments

Working with GCP environments

Introduction to Data Lakes

Creating your cluster

Once you have met the prerequisites, you are ready to create a managed and secured Flow Management cluster in Cloudera Public Cloud by using one of the prescriptive flow management cluster definitions. You can select the cluster definition that matches your cloud provider and choose from the light and heavy duty options available.

About this task

You can create your Flow Management cluster using the Cloudera UI or CDP CLI. The UI offers a graphical interface that simplifies the cluster creation process. You can select the desired configurations for your cluster easily.

When you create a new cluster through the UI, Java 8 is used as the default JDK. The ability to deploy a new Flow Management cluster with Java 11 is available only using the command-line interface. The CDP CLI also allows you to leverage scripting and automation capabilities to create clusters programmatically or perform bulk operations.

For UI

Steps

1. Log into the Cloudera web interface.
2. Navigate to Cloudera Management Console Environments , and select the environment where you would like to create a cluster.
3. Click Create Data Hub.

   The Provision Data Hub page is displayed:

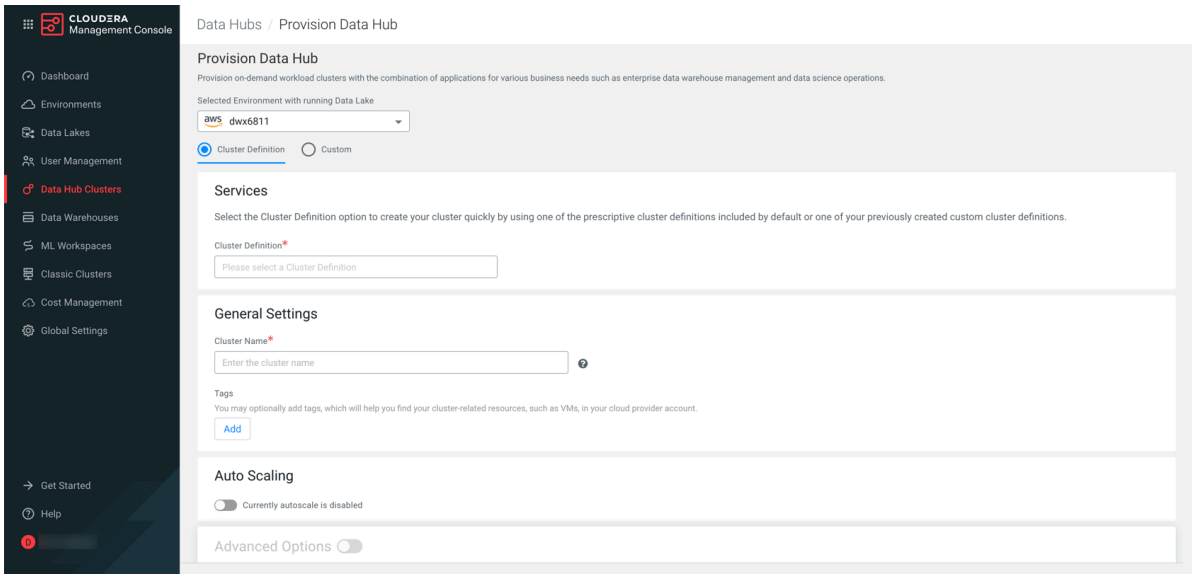

4. Select Cluster Definition.
5. Select the appropriate Flow Management cluster definition from the Cluster Definition dropdown depending on your operational objectives.

   There are six template options available:

   - Flow Management Light Duty for AWS
   - Flow Management Heavy Duty for AWS
   - Flow Management Light Duty for Azure
   - Flow Management Heavy Duty for Azure
   - Flow Management Light Duty for GCP
   - Flow Management Heavy Duty for GCP

   For more information on templates, see *Flow Management cluster definitions* and *Flow Management cluster layout*.

   The list of services is automatically shown below the selected cluster definition name.
6. Give the cluster a name and add any tags you might need.

   You can define tags that will be applied to your cluster-related resources on your cloud provider account. For more information about tags, see Tags.
7. Optional: Use the Configure Advanced Options section to customize the infrastructure settings.

   Note: If you want to deploy your Flow Management Data Hub cluster across multiple availability zones (use multi-AZ support), go to the Network And Availability section and select all desired subnets from the available options. NiFi nodes will be provisioned across the selected subnets. Currently, Multi-AZ support is only available in Amazon Web Services (AWS) environments.
8. Click Provision Cluster.

Result

You will be redirected to the Data Hub cluster dashboard, and a new tile representing your cluster will appear at the top of the page.

For CLI

Before you begin

You have installed and configured CDP CLI. For more information about setting up the CLI client, see *CDP CLI*.

Steps

Use the create-<cloud_provider>-cluster command and include the --java-version parameter to create your Flow Management cluster.

Valid inputs for the --java-version parameter are 8 and 11.

Example

```
cdp datahub create-aws-cluster \
  --cluster-name NIFI-CLUSTER \
  --environment-name EXAMPLE-SANDBOX-AWS \
  --cluster-template-name "7.2.17 - FLOW MANAGEMENT LIGHT DUTY WITH APACHE
 NIFI, APACHE NIFI REGISTRY, SCHEMA REGISTRY" \
  --instance-grou
ps NODECOUNT=3,INSTANCEGROUPNAME=NIFI,INSTANCEGROUPTYPE=CORE,INSTANCETYPE=M5.2XLARGE,
\[\{VOLUMESIZE=500,VOLUMECOUNT=4,VOLUMETYPE=STANDARD
\}\],RECOVERYMODE=MANUAL
 NODECOUNT=1,INSTANCEGROUPNAME=MANAGEMENT,INSTANCEGROUPTYPE=GATEWAY,INSTANCETYPE=M5.2
\[\{VOLUMESIZE=100,VOLUMECOUNT=1,VOLUMETYPE=STANDARD
\}\],RECOVERYMODE=MANUAL
 NODECOUNT=0,INSTANCEGROUPNAME=NIFI_SCALING,INSTANCEGROUPTYPE=CORE,INSTANCETYPE=M5.2X
\[\{VOLUMESIZE=500,VOLUMECOUNT=4,VOLUMETYPE=STANDARD
\}\],RECOVERYMODE=MANUAL \
  --image ID=6E038B51-0AEA-43CE-873A-A0AA438B54F9,CATALOGNAME=CDP-DEFAULT
 \
  --datahub-database NON_HA \
  --java-version 11 \
  --NO-ENABLE-LOAD-BALANCER
```

After you finish

After you have created your Flow Managament Data Hub cluster, you are ready to build your first data flow in Cloudera Public Cloud.

Related Information

Flow Management cluster definitions

Flow Management cluster layout

Tags

Giving access to your cluster

As an EnvironmentAdmin, you need provide users access to your environment and to the new Flow Management cluster by assigning user roles and adding users to Ranger policies.

About this task

The cluster you have created using the Flow Management cluster definition is kerberized and secured with SSL. Users can access cluster UIs and endpoints through a secure gateway powered by Apache Knox. Before you can begin developing data flows, you must give users access to the Flow Management cluster components.

Procedure

1. Assign the EnvironmentUser role to the users to grant access to the Cloudera environment and the Flow Management cluster.
2. Sync the updated user permissions to the Data Lake (Ranger) using ActionsSynchronize Users to FreeIPA.

   This synchronizes the user to the FreeIPA identity management system to enable SSO.
3. Add the user to the appropriate pre-defined Ranger policies.

   To access the NiFi canvas and view or manage NiFi resources, the user must be assigned to at least:

   - Flow
   - Root Process Group

   If needed, you can also create a custom Ranger access policy and add the user to it.

   For more information, see *Security for Flow Management Clusters and Users in Cloudera Public Cloud*.

Related Information

Security for Flow Management Clusters and Users in Cloudera Public Cloud

Cloudera DataFlow for Data Hub 7.3.1

Setting up your Streaming Analytics cluster

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Before creating your cluster **4**
Assigning resource roles 4
Creating IDBroker mapping 4
Setting workload password 5

Creating your cluster **5**

After creating your cluster **7**
Enabling Server-Side Events (SSE) for sampling in Cloudera SQL Stream Builder 7
Retrieving keytab file 8
Uploading or unlocking your keytab 8
Configuring Ranger policies for Flink and Cloudera SQL Stream Builder 11
Configuring Kafka policies 12
Configuring Schema Registry policies 14
Configuring Hive policies 15
Configuring Kudu policies 17

Before creating your cluster

Before you start creating your Cloudera Streaming Analytics Cloudera Data Hub cluster, you need to ensure that you have set up the environment properly and have all the necessary accesses to use Cloudera Public Cloud.

- You have Cloudera login credentials.
- You have an available Cloudera environment.
- You have a running Data Lake.
- You have a Cloudera username and the predefined resource role of this user is EnvironmentAdmin.
- Your Cloudera user is synchronized to the Cloudera Public Cloud environment.

Important: Ensure that the Cloudera Runtime version of the Data Lake cluster matches the version of the Cloudera Data Hub cluster that you are about to create. If these versions do not match, you may encounter warnings and/or errors.

Related Information

Getting started as a user
AWS environments
Azure environments
GCP environments
Data lakes

Assigning resource roles

As an administrator, you need to give permissions to users or groups to be able to access and perform tasks in your Cloudera Data Hub environment.

Procedure

1. Navigate to Cloudera Management Console > Environments and select your environment.
2. Click Actions > Manage Access .
3. Search for a user or group that needs access to the environment.
4. Select EnvironmentUser role from the list of Resource Roles.
5. Click Update Roles.
   The Resource Role for the selected user or group will be updated.
6. Navigate to Cloudera Management Console > Environments , and select the environment where you want to create a cluster.
7. Click Actions > Synchronize Users to FreeIPA .
8. Click Synchronize Users.

   Note: There might be cases where the status of the environment is synchronized with warnings and has failed status. This does not indicate that the synchronization has failed.

Creating IDBroker mapping

As an administrator, you must create IDBroker mapping for a user or group to access cloud storage. As a part of Knox, the IDBroker allows a user to exchange cluster authentication for temporary cloud credentials.

About this task

You must create IDBroker mapping for a user or group to have access to the S3 cloud storage. As a part of Knox, the IDBroker allows a user to exchange cluster authentication for temporary cloud credentials. The following roles are created when registering the Cloudera environment:

- idbroker-role: granting permissions to IDBroker instances associated with the Cloudera environment
- datalake-admin-role: granting access to Cloudera cloud resources
- logs-role: granting access to the logs storage location

For using Cloudera Streaming Analytics in Cloudera Public Cloud, you must make sure that the users who run Flink jobs are associated with the ARN of the datalake-admin-role as it grants access to the cloud resources required to run the Flink service.

Procedure

1. Navigate to Cloudera Management Console > Environments and select your environment.
2. Click Actions > Manage Access .
3. Click on the IDBroker Mappings tab.
4. Click Edit to add a new user or group and assign roles to have writing access for the cloud storage.
5. Search for the user or group you need to map.
6. Go to the IAM Summary page where you can find information about your cloud storage account.
7. Copy the Role ARN.
8. Go back to the IDBroker Mapping interface on the Cloudera Management Console page.
9. Paste the Role ARN to your selected user or group.
10. Click Save and Sync.

Setting workload password

As a user, you need to set a workload password for your EnvironmentUser account to be able to access the Cloudera SQL Stream Builder nodes through SSH connection.

Procedure

1. Navigate to Cloudera Management Console > Environments and select your environment.
2. Click Actions > Manage Access .
3. Click Workload Password.
4. Give a chosen workload password for your user.
5. Confirm the given password by typing it again.
6. Click Set Workload Password.

Creating your cluster

When creating your Cloudera Streaming Analytics, you must choose from the Light and Heavy duty options, and further select the cluster definition that matches your cloud provider for the environment. You also need to pay attention to the cloud storage settings where Flink saves the checkpoints and savepoints.

About this task

After you have met the prerequisites, you are ready to create your Cloudera Streaming Analytics cluster using a default cluster definition.

Procedure

1. Log into the Cloudera web interface.
2. Navigate to Cloudera Management Console > Environments , and select the environment where you want to create a cluster.
3. Click Create Data Hub.

   The following page is displayed:

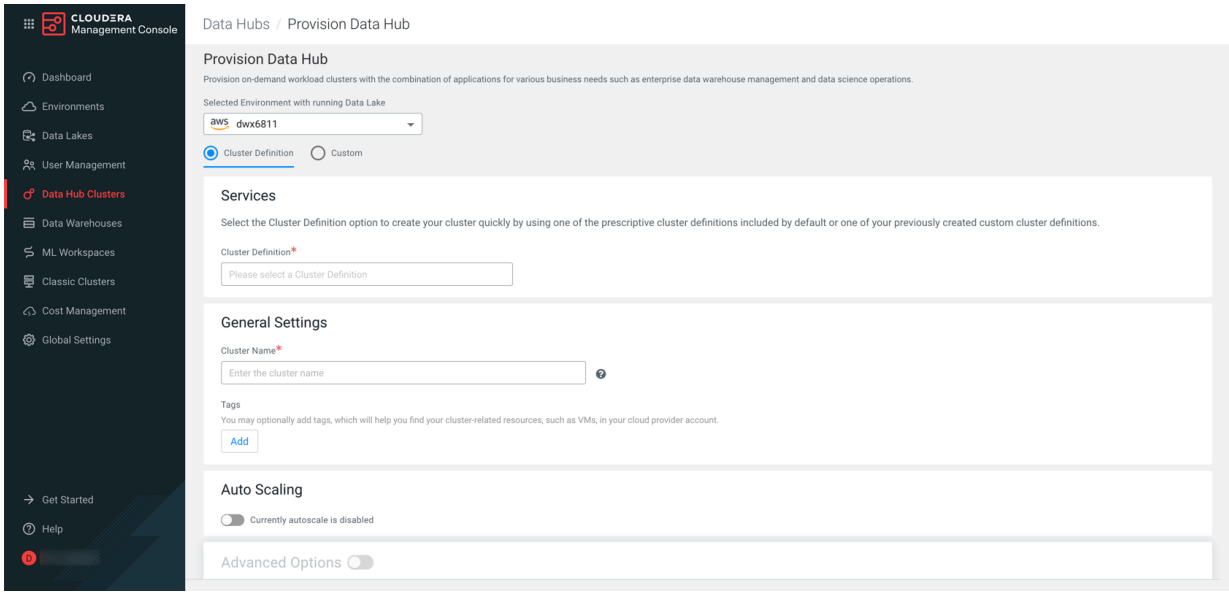

4. Select Cluster Definition.
5. Select the Cloudera Streaming Analytics cluster definition from the Cluster Definition drop-down list depending on your operational objectives.

   The following template options are available:

   - 7.3.1 - Cloudera Streaming Analytics Light Duty for AWS
   - 7.3.1 - Cloudera Streaming Analytics Light Duty for Azure
   - 7.3.1 - Cloudera Streaming Analytics Light Duty for GCP
   - 7.3.1 - Cloudera Streaming Analytics Heavy Duty for AWS
   - 7.3.1 - Cloudera Streaming Analytics Heavy Duty for Azure
   - 7.3.1 - Cloudera Streaming Analytics Heavy Duty for GCP

   For more information on templates, see *Cloudera Streaming Analytics Cloudera Data Hub cluster definitions* and *Cloudera Streaming Analytics cluster layout*.

   The list of services is automatically shown below the selected cluster definition name.
6. Provide a cluster name and add tags you might need.

   You can define tags that will be applied to your cluster- related resources on your cloud provider account. For more information about tags, see Tags.
7. Optionally, use the Configure Advanced Options section to customize the infrastructure settings.

   Note: Ensure that the right cloud storage path is given in Advanced Options > Cloud Storage . Cloudera recommends saving the checkpoints and savepoints to the S3 cloud storage to make the saved files available throughout all cluster deployments. You can also use HDFS, however Cloudera only recommends this solution for temporary storage of checkpoints and savepoints.
8. Click Provision Cluster.

Results

You are redirected to the Cloudera Data Hub cluster dashboard, and a new tile representing your cluster appears at the top of the page.

Related Information

Create a cluster from a definition on AWS
Create a cluster from a definition on Azure
Create a cluster from a definition on GCP
Tags
Streaming Analytics Data Hub cluster definitons
Cloudera Streaming Analytics cluster layout

After creating your cluster

As an EnvironmentAdmin, you need to provide access to users to your environment and to the Cloudera Streaming Analytics cluster by assigning user roles, adding users to Ranger policies, and creating IDBroker mappings.

About this task

The cluster you have created using the Cloudera Streaming Analytics cluster definition is kerberized and secured with SSL. Users can access cluster UIs and endpoints through a secure gateway powered by Apache Knox. Before you can use Flink and Cloudera SQL Stream Builder, you must provide users access to the Cloudera Streaming Analytics cluster components.

Related Information

IDBroker
Set Ranger policies for Flink
Set Ranger policies for Cloudera SQL Stream Builder

Enabling Server-Side Events (SSE) for sampling in Cloudera SQL Stream Builder

The sampling feature in Cloudera SQL Stream Builder uses Server-Side Events (SSE). Disabled by default, you need to enable the async configuration flag in KNOX.

Procedure

1. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
2. Go to KNOX serviceConfiguration.
3. Add the following property to the conf/gateway-site.xml_role_safety_valve configuration:

```
<property>
    <name>gateway.servlet.async.supported</name>
    <value>true</value>
</property>
```

4. Restart the KNOX service.

Results

Sampling through SSE is now available in Cloudera SQL Stream Builder.

Retrieving keytab file

As a user, you need to retrieve the keytab file of your profile and upload it to the Streaming SQL Console to be able to run SQL jobs.

Procedure

1. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.
2. Click on your profile name.
3. Click Profile.
4. Click Actions > Get Keytab.
5. Choose the environment where your Cloudera Data Hub cluster is running.
6. Click Download.
7. Save the keytab file in a chosen location.

Uploading or unlocking your keytab

When accessing the Streaming SQL Console for the first time in Data Hub, you must upload and unlock the keytab file corresponding with your profile before you can use SQL Stream Builder (SSB).

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.
2. Click your username on the sidebar of the Streaming SQL Console.

3. Click Manage keytab.

   The **Keytab Manager** window appears.

   You can either unlock the keytab already existing on the cluster, or you can directly upload your keytab file in the SQL Stream Builder.

   a) Unlock your keytab by providing the Principal Name and Password, and clicking Unlock Keytab. The Principal Name and Password should be the same as the workload username and password set for the Streaming Analytics cluster.

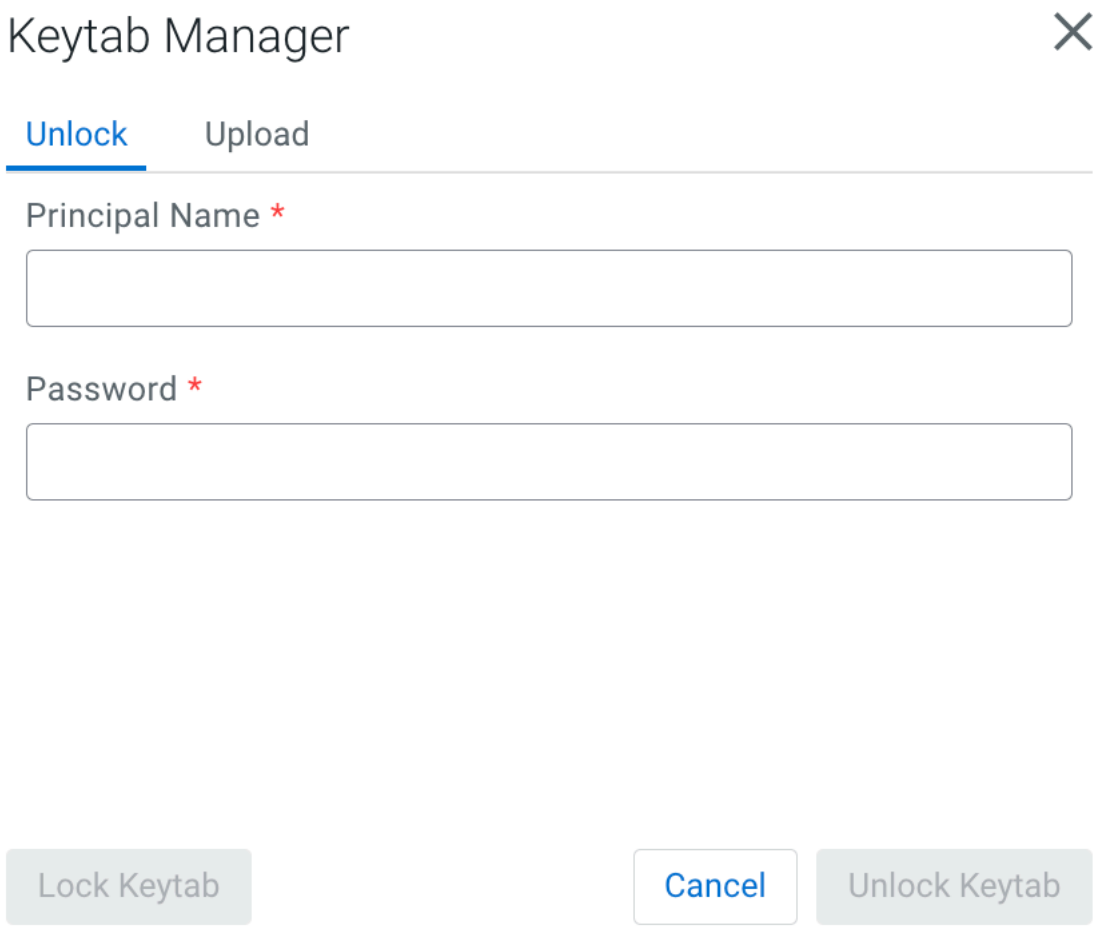

   b) Upload your keytab by clicking on the Upload tab, uploading the keytab file directly to the Console, and clicking Unlock Keytab.

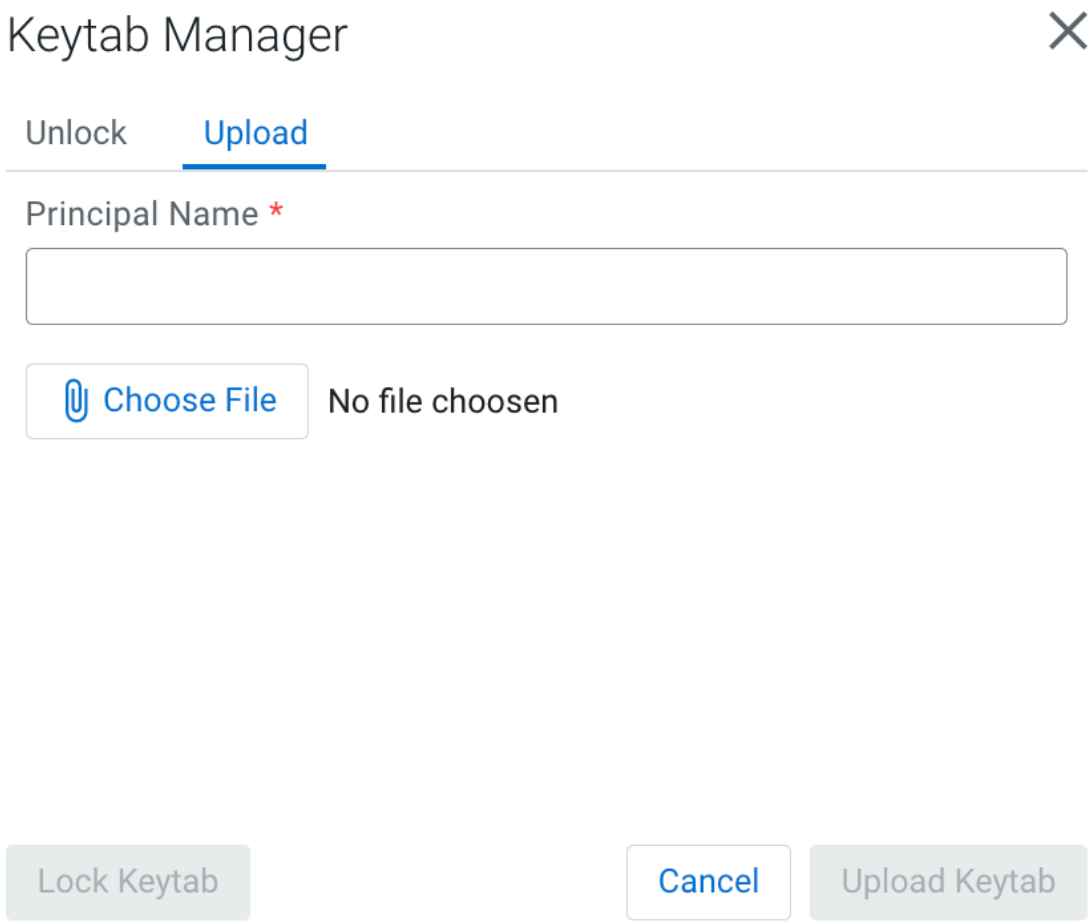

In case there is an error when unlocking your keytab, you can get more information about the issue with the following steps:

a. Retrieve your keytab file.

1. Click on your profile name in the **Management Console**.
2. Click Profile.
3. Click Actions > Get Keytab.
4. Choose the environment where your Data Hub cluster is running.
5. Click Download.
6. Save the keytab file in a chosen location.

b. Manually upload your keytab to the Streaming Analytics cluster:

```
scp <location>/<your_keytab_file> <workload_username>@<manager_node_FQDN
>:.
                    Password:<your_workload_password>
```

c. Access the manager node of your Streaming Analytics cluster:

```
ssh <workload_username>@<manager_node_FQDN>
```

```
                                Password: <workload_password>
```

d. Use kinit command to authenticate your user:

```
kinit -kt <keytab_filename>.keytab <workload_username>
```

e. Use the flink-yarn-session command to see if the authentication works properly:

```
flink-yarn-session -d \
-D security.kerberos.login.keytab=<keytab_filename>.keytab \
-D security.kerberos.login.principal=<workload_username>
```

In case the command fails, you can review the log file for further information about the issue.

Configuring Ranger policies for Flink and Cloudera SQL Stream Builder

You must add your workload username and the Cloudera SQL Stream Builder service user to the Ranger policies that are needed for Kafka, Schema Registry, Hive and Kudu to provide access to topics, schemas and tables used by the components and to be able to execute Flink jobs.

About this task

You need to provide access to users and the Cloudera SQL Stream Builder service by configuring Ranger policies for the Kafka data source and the Schema Registry, Kudu and Hive catalog services. To be able to use Flink, you need to add the workload user or users to the required policies. For Cloudera SQL Stream Builder, the ssb service user needs to be added to the same policies.

When adding more workload users, instead of adding them one by one, you can create user groups in Ranger, for example a flink_users group. This way you can assign every user who will use the Cloudera Streaming Analytics cluster into a group, and add only that one group to the Ranger policies.

Procedure

1. Navigate to Cloudera Management Console > Environments , and select the environment where you have created your cluster.

2. Click Ranger from the **Quick Links** or select Data Lake Ranger from **Services**.

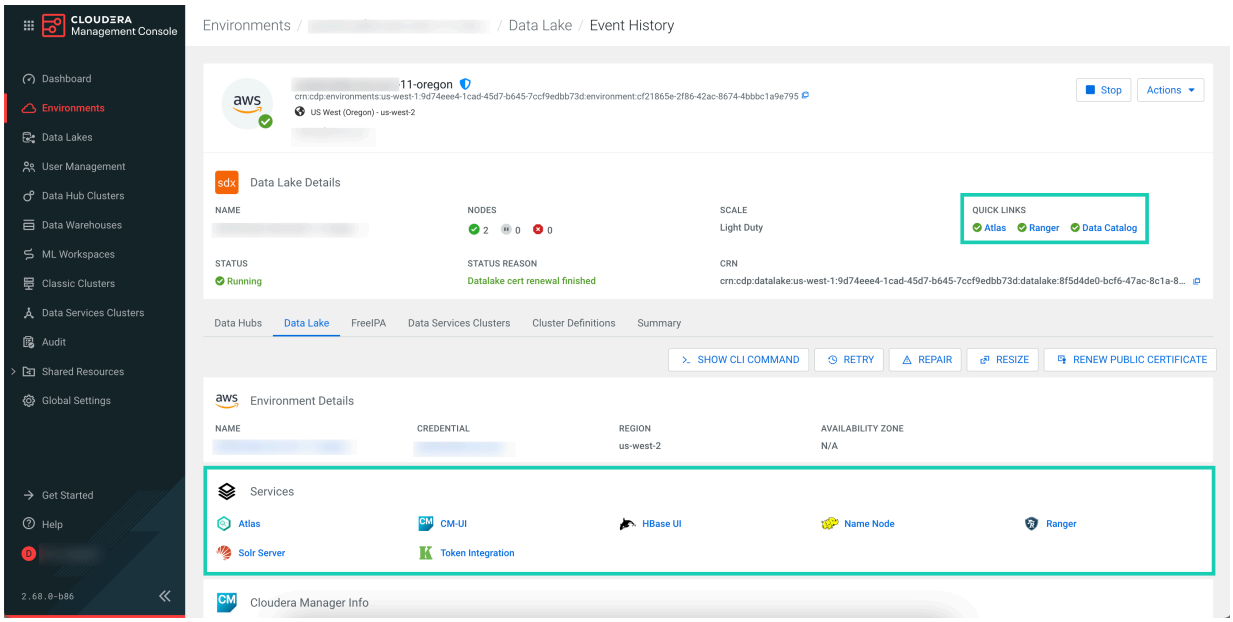

You are redirected to the Ranger Admin Web user interface (UI) where you can add the workload user and Cloudera SQL Stream Builder service user to the required policies to grant access for Flink and Cloudera SQL Stream Builder.

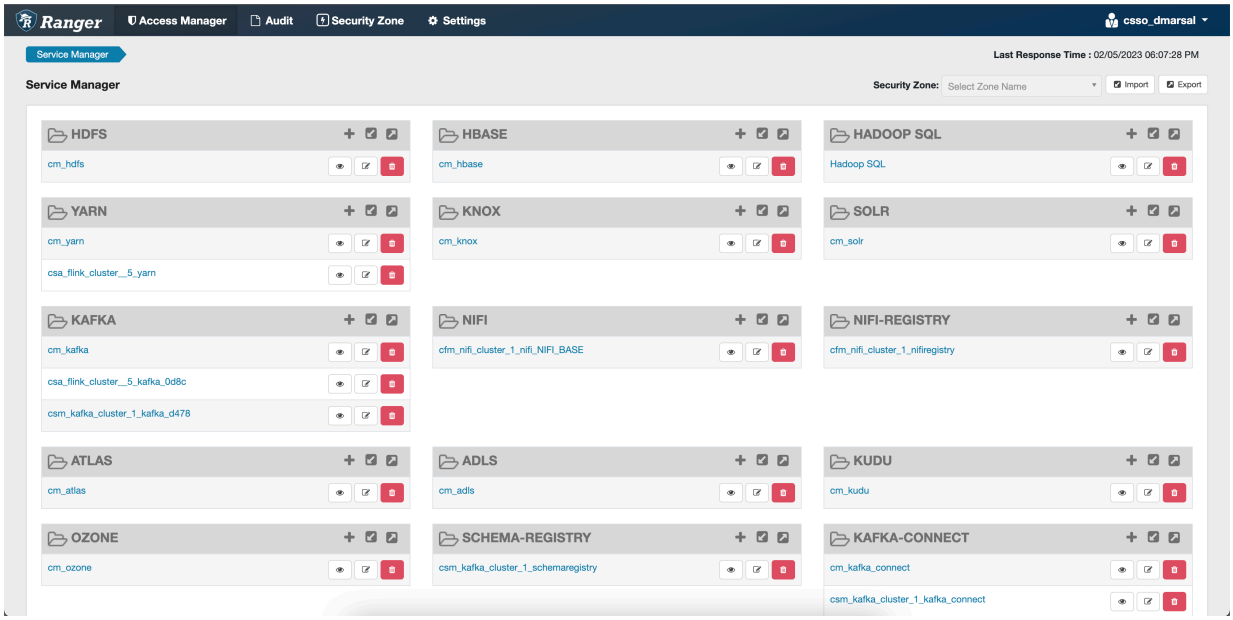

Configuring Kafka policies

After accessing the Ranger Admin Web UI, the workload username or user groups, and the Cloudera SQL Stream Builder service user needs to be added to the Kafka policies to be able to use Flink and Cloudera SQL Stream Builder.

About this task

The following resource based policies need to be configured for the Kafka service:

- all - topic: Provides access to all topics for users
- all - consumergroup: Provides access to all consumer groups for users
- all - cluster: Provides access to all clusters to users

- all - transactionalid: Provides transactionalid access to users
- all - delegationtoken: Provides delegationtoken access to users

You need to ensure that the required workload username or user group, and the ssb service user is added to the policies of the Kafka service and to the policies of the created Cloudera Streams Messaging and Cloudera Streaming Analytics clusters.

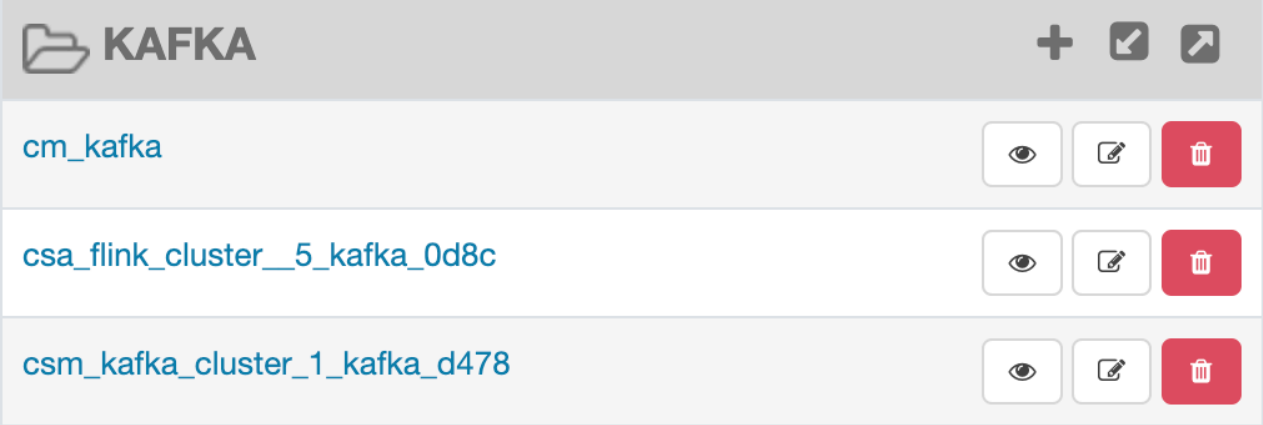

Procedure

1. Click cm_kafka under **Kafka** service on the **Service Manager** page.

   You are redirected to the **cm_kafka Policies** page.

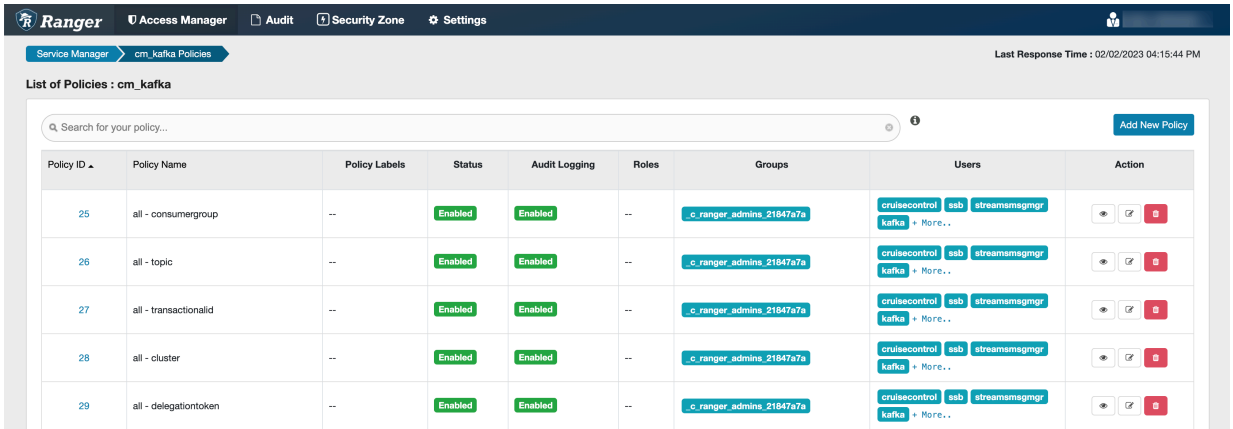

2. Click on the edit button of the *ALL-CONSUMERGROUP* policy.
3. Add the user group to the Select Group field under the **Allow Conditions** settings.
   - Alternatively, you can also add the workload usernames to the Select User field under the **Allow Conditions** setting.

4. Add the ssb service user to the Select User field under the **Allow Conditions** setting, if it is not configured by default.

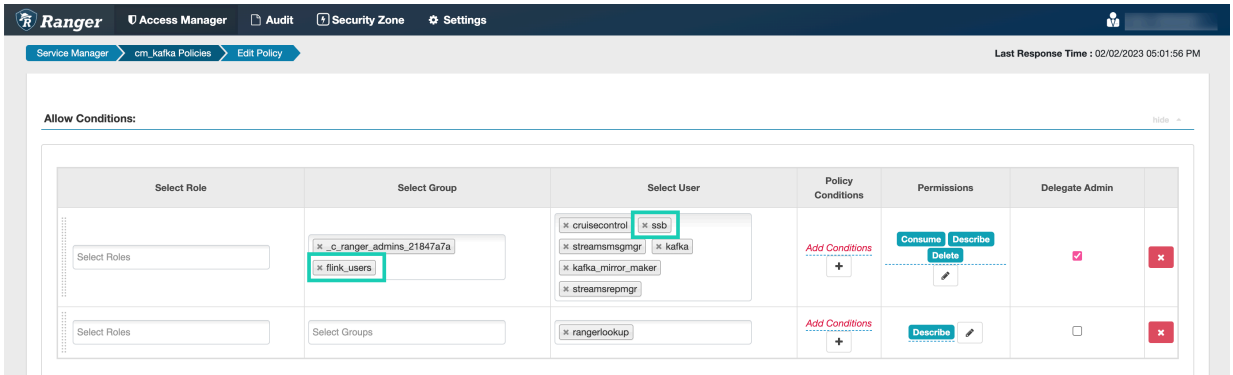

5. Click Save.

   You are redirected to the list of Kafka policies page.

6. Click on + More… to check if the needed workload user is listed under the **Users** for the edited policy.

   Repeate same steps to the remaining Kafka policies as well based on the required authorization level:

   - all - topic
   - all - transactionalid
   - all - cluster

Configuring Schema Registry policies

After accessing the Ranger Admin Web UI, the workload username or user groups, and the Cloudera SQL Stream Builder service user needs to be added to the Schema Registry policies to be able to use Flink and Cloudera SQL Stream Builder.

About this task

The following resource based policy need to be configured for the Schema Registry service:

- all - export-import: Provides import and export access for users.
- all - serde: Provides access to store metadata for the format of how the data should be read and written.
- all - schema-group, schema-metadata: Provides access to the schema groups, schema metadata, and schema branch.
- all - schema-group, schema-metadata, schema-branch: Provides access to the schema groups, schema metadata, and schema branch.
- all - schema-group, schema-metadata, schema-branch, schema-version: Provides access to schema groups, schema metadata, schema branch, and schema version.
- all - registry-service: Provides access to the schema registry service, the user can access all Schema Registry entities.

You need to ensure that the required workload username or user group, and the ssb service user is added to the policies of the created Cloudera Streams Messaging clusters.

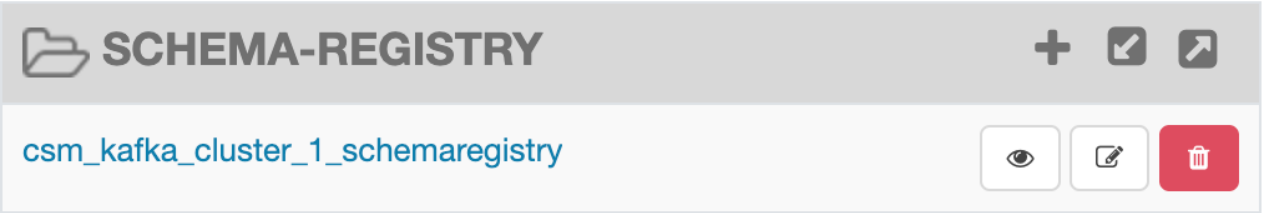

Procedure

1. Select your Cloudera Streams Messaging cluster under the **Schema Registry** service on the **Service Manager** page.

   You are redirected to the list of Schema Registry policies page.

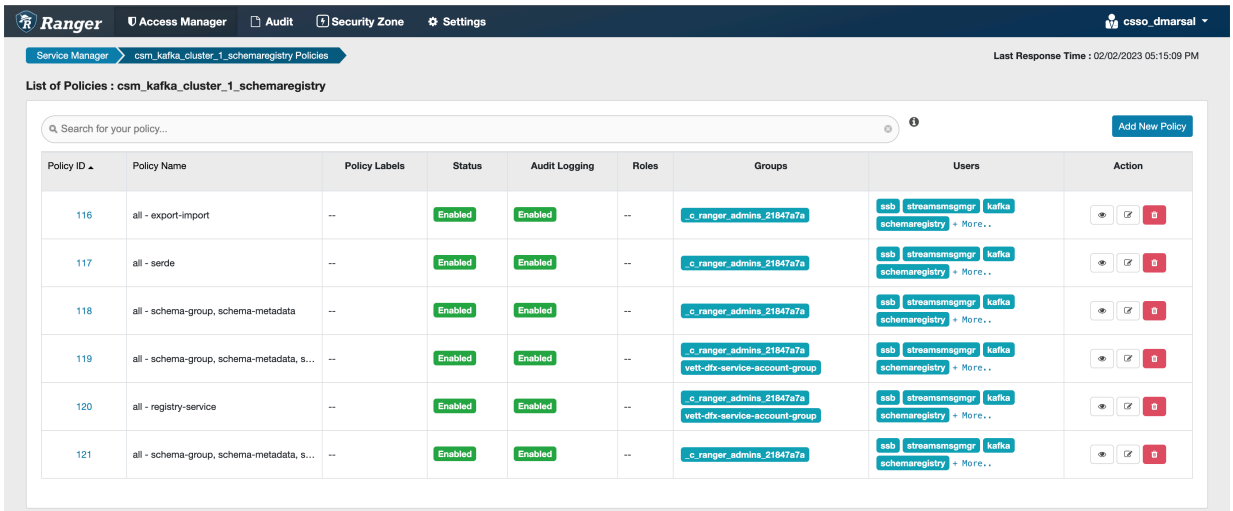

2. Click on the edit button of the *ALL-SCHEMA-GROUP*, *SCHEMA-METADATA*, *SCHEMA-BRANCH*, *SCHEMA-VERSION* policy.
3. Add the user group to the Select Group field under the **Allow Conditions** settings.
   - Alternatively, you can also add the workload usernames to the Select User field under the **Allow Conditions** setting.
4. Add the ssb service user to the Select User field under the **Allow Conditions** setting, if it is not configured by default.

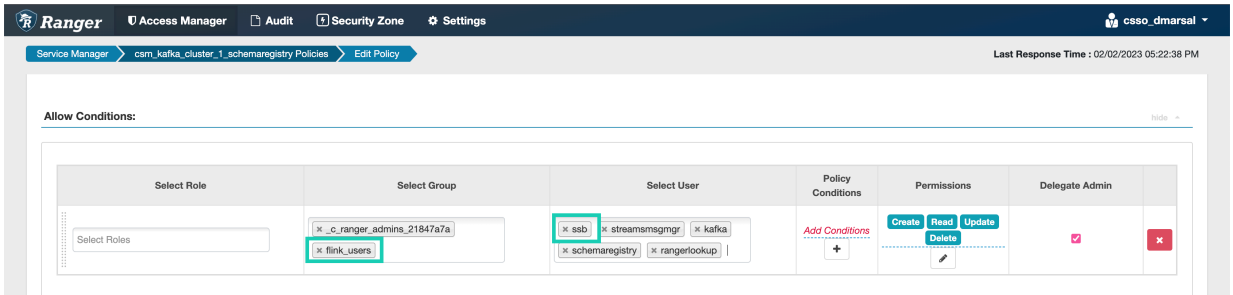

5. Click Save.

   You are redirected to the list of **Schema Registry policies** page.
6. Click on + More… to check if the needed workload user is listed under the **Users** for the edited policy.

   Repeate same steps to the remaining Schema Registry policies as well based on the required authorization level:
   - all - export-import
   - all - serde
   - all - schema-group, schema-metadata
   - all - schema-group, schema-metadata, schema-branch, schema-version
   - all - registry-service

Configuring Hive policies

After accessing the Ranger Admin Web UI, the workload username or user groups, and the Cloudera SQL Stream Builder service user needs to be added to the Hadoop SQL policies to be able to use Flink and Cloudera SQL Stream Builder with Hive.

About this task

The following resource based policy need to be configured for the Hive service:

- all - global: Provides global access to users.
- all - database, table, column: Provides access to all databases, tables, and columns for users.
- all - database, table: Provides access to all databases and tables for users.
- all - database: Provides access to all databases for users.
- all - hiveservice: Provides hiveservice access to users.
- all - database, udf: Provides database and udf access to users.
- all - url: Provides url access

You need to ensure that the required workload username or user group, and the ssb service user is added to the policies of the Hive service and the created Cloudera Operational Database clusters.

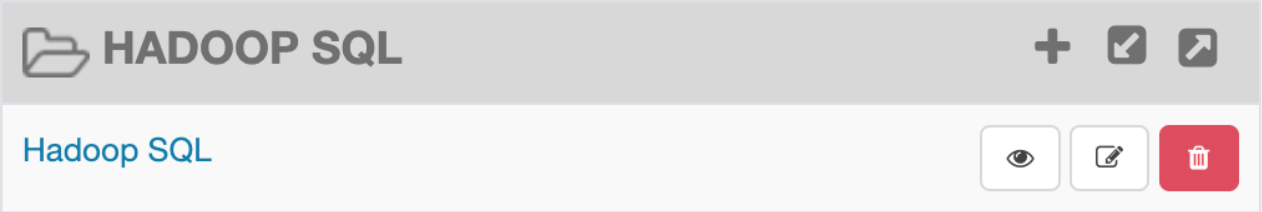

Procedure

1. Click Hadoop SQL under **Hadoop SQL** service on the **Service Manager** page.

   You are redirected to the list of **Hadoop SQL** policies page.

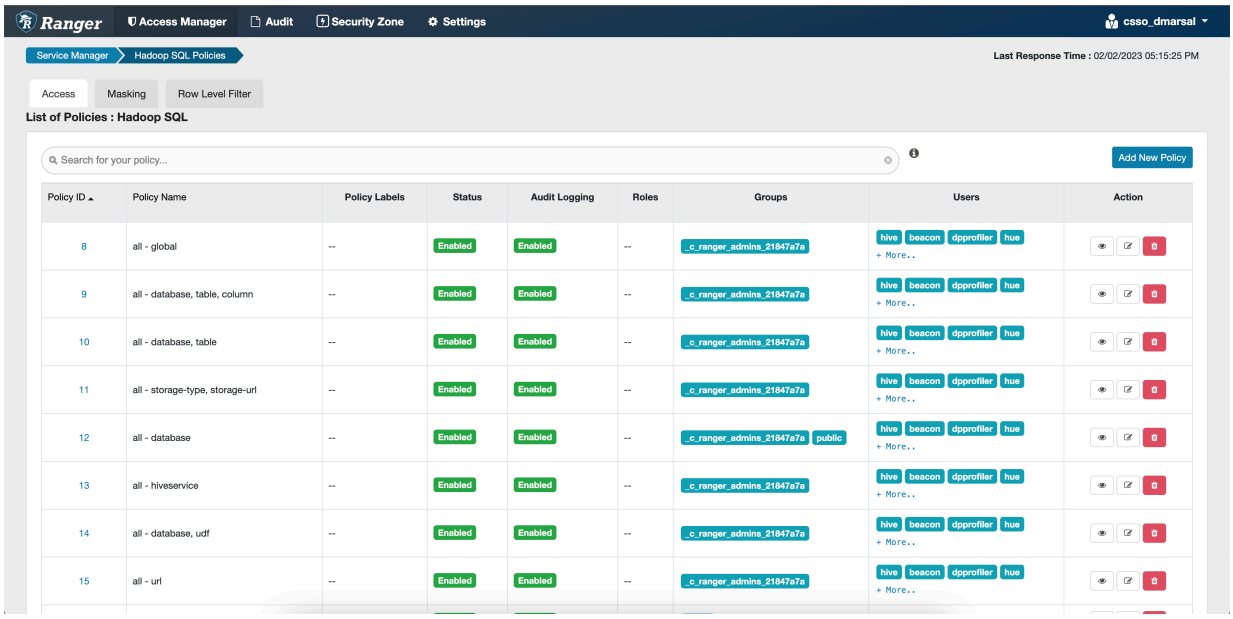

2. Click on the edit button of the *ALL-GLOBAL* policy.
3. Add the user group to the Select Group field under the **Allow Conditions** settings.
   - Alternatively, you can also add the workload usernames to the Select User field under the **Allow Conditions** setting.

4. Add the ssb service user to the Select User field under the **Allow Conditions** setting, if it is not configured by default.

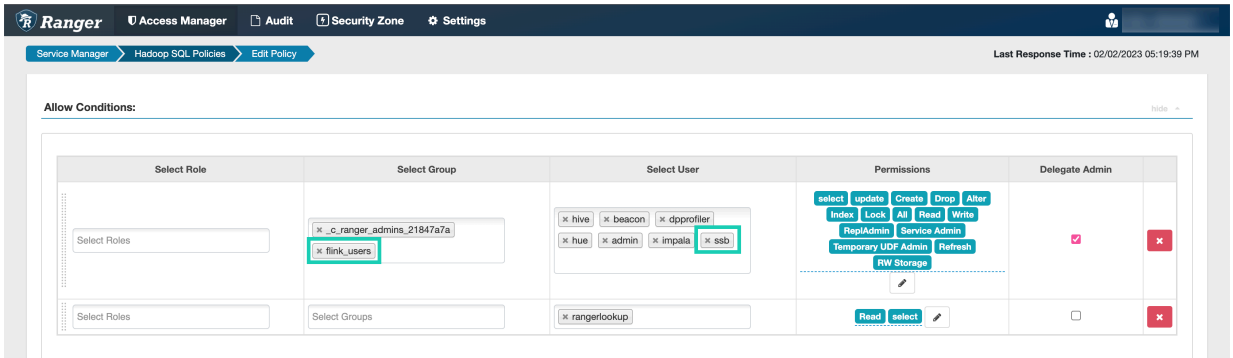

5. Click Save.

   You are redirected to the list of **Hadoop SQL policies** page.

6. Click on + More… to check if the needed workload user and the ssb are listed under the **Users** for the edited policy.

   Repeate same steps to the remaining Hadoop SQL policies as well based on the required authorization level:

   - all-database, table, column
   - all-database, table
   - all-database
   - all-database, udf
   - all-hiveservice
   - all-url

Configuring Kudu policies

After accessing the Ranger Admin Web UI, the workload username or user groups, and the Cloudera SQL Stream Builder service user needs to be added to the Kudu policies to be able to use Flink and Cloudera SQL Stream Builder.

About this task

The following resource based policy need to be configured for the Kudu service:

- all - database, table: Provides access to all databases and tables for users.
- all - database, table, column: Provides access to all databases, tables, and columns for users.
- all - database: Provides access to all databases for users.

You need to ensure that the required workload username or user group, and the ssb service user is added to the policies of the Kudu service and the created Real-Time Data Mart clusters.

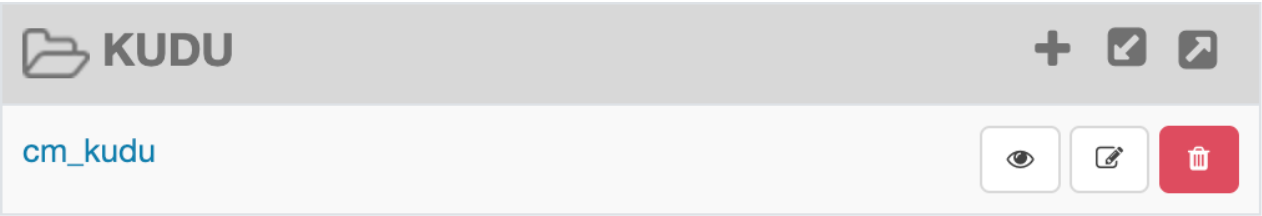

Procedure

1. Click cm_kudu under **Kudu** service on the **Service Manager** page.

   You are redirected to the **cm_kudu Policies** page.

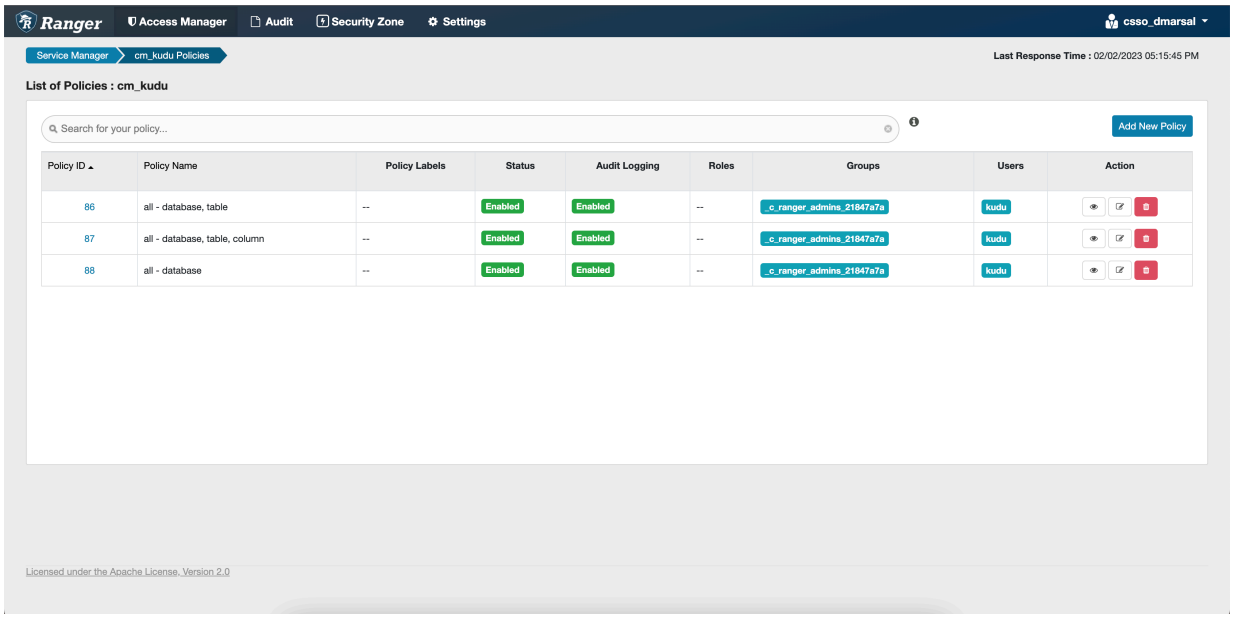

2. Click on the edit button of the *ALL-DATABASE* policy.
3. Add the user group to the Select Group field under the **Allow Conditions** settings.
   - Alternatively, you can also add the workload usernames to the Select User field under the **Allow Conditions** setting.
4. Add the ssb service user to the Select User field under the **Allow Conditions** setting, if it is not configured by default.

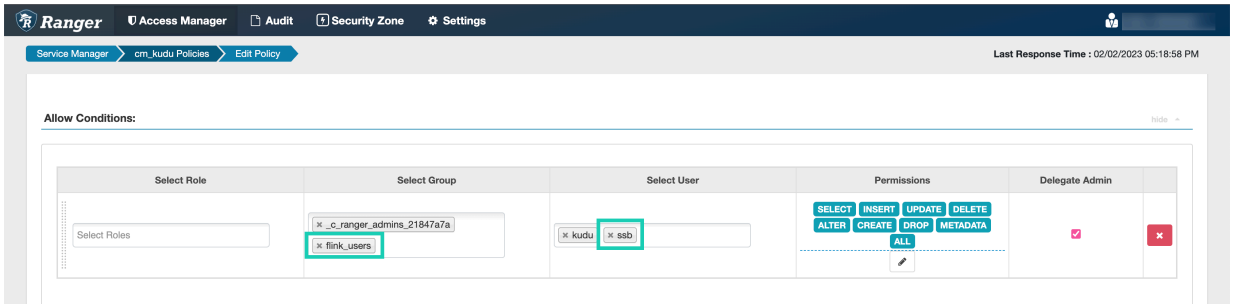

5. Click Save.

   You are redirected to the list of **Kudu policies** page.
6. Click on + More… to check if the needed workload user is listed under the **Users** for the edited policy.

   Repeate same steps to the remaining Kudu policies as well based on the required authorization level:
   - all - database, table
   - all - database, table, column

   Note: From Cloudera Public Cloud 7.2.16, Kudu environments have their own Ranger policy. Ensure that the permissions are added for every relevant Kudu cluster in the environment.

Cloudera DataFlow for Data Hub 7.3.1

Setting up your Streams Messaging cluster

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Checking prerequisites 4

Creating your cluster 4

Deleting ZooKeeper from Streams Messaging clusters 7

Configuring data directories for clusters with custom disk configurations 8

Giving access to your cluster 11

Checking prerequisites

Before you start creating your first Streams Messaging cluster, you need to ensure that you have set up the environment properly and have all the necessary accesses to use Cloudera Public Cloud. Use this checklist to make sure that you meet all the requirements before you start creating your first cluster.

- You have Cloudera login credentials.
- You have an available Cloudera environment.
- You have a running Data Lake.
- You have a Cloudera username and the predefined resource role of this user is EnvironmentAdmin.
- Your Cloudera user is synchronized to the Cloudera Public Cloud environment.
- Your Cloudera user has the correct permissions set up in Ranger allowing access to Kafka, Streams Messaging Manager and Schema Registry.

Important: Make sure that the Cloudera Runtime version of the Data Lake cluster matches the Cloudera Runtime version of the Cloudera Data Hub cluster that you are about to create. If these versions do not match, you may encounter warnings and/or errors.

Related Information

Getting started as a user

AWS environments

Azure environments

GCP environments

Data lakes

Creating your cluster

Once you have met the prerequisites, you are ready to create a managed and secured Streams Messaging cluster in Cloudera Public Cloud by using one of the prescriptive Streams Messaging cluster definitions.

Procedure

1. Log into the Cloudera web interface.
2. Navigate to Management Console Environments , and select the environment where you would like to create a cluster.

3. Click Create Data Hub.

   The following page is displayed:

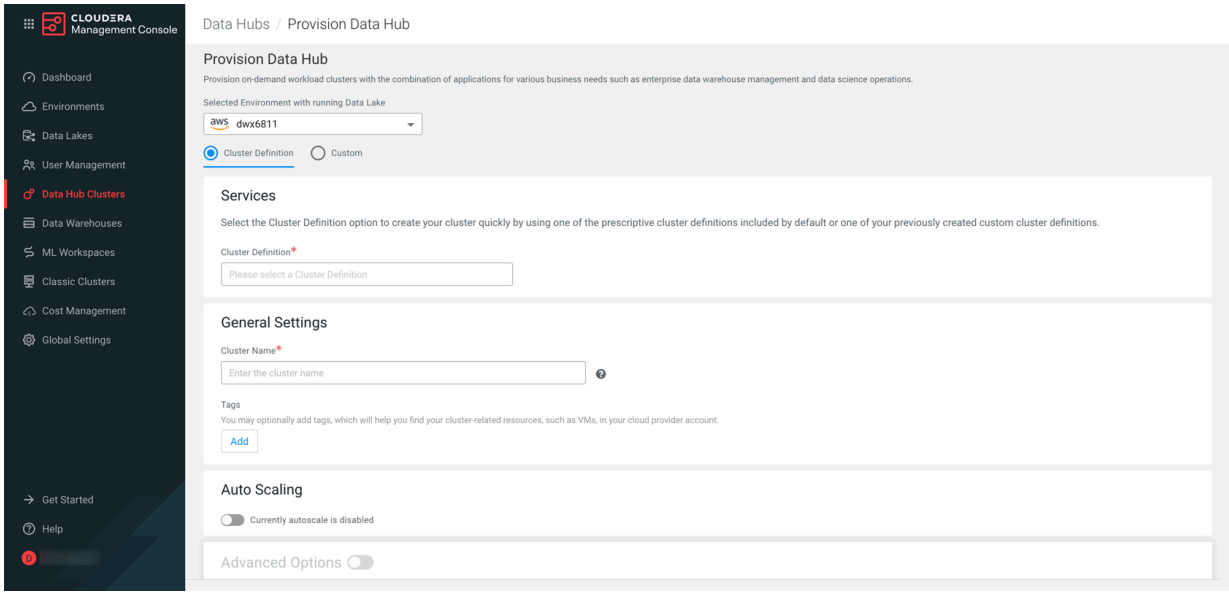

4. Select Cluster Definition.
5. Select the appropriate Streams Messaging cluster definition from the Cluster Definition dropdown depending on your operational objectives.

   The list of services is automatically shown below the selected cluster definition name. For more information on templates, see *Cloudera Data Hub cluster definitions* and *Streams Messaging cluster layout*.
6. Give the cluster a name and add any tags you might need.

   You can define tags that will be applied to your cluster-related resources on your cloud provider account. For more information about tags, see Tags.
7. If using the Streams Messaging High Availability definitions, ensure that you select multiple subnets.

   Multiple subnets can be selected by navigating to Advanced Options Network And Availability .
8. Add Kafka KRaft to your cluster.

   Note: Kafka KRaft is available in this version of Cloudera but is not ready for production deployment. Cloudera encourages you to explore this technical preview feature in non-production environments and provide feedback on your experiences through the Cloudera Community Forums. For more information regarding KRaft limitations and unsupported features, see Unsupported Streams Messaging features.

   Kraft nodes are not provisioned by default in any of the Streams Messaging cluster definitions. If you want to run Kafka in KRaft mode, you must set the **Instance Count** of the **Kraft Nodes** host group to an odd number. An odd number of nodes is required because KRaft requires an odd number of instances to function. Cluster provisioning fails if the number of KRaft nodes is set to an even number.

   Note: While it is possible to run Kafka in KRaft mode with a single KRaft node, Cloudera recommends that you provision a minimum of three nodes to avoid having a single point of failure. Additionally, note that ZooKeeper service instances are still deployed even in KRaft mode. You can optionally delete ZooKeeper after the cluster is deployed.

   a) Go to Advanced Options Hardware and Storage .
   b) Locate the **Kraft Nodes** host group and click to edit host group details.
   c) Set the Instance Count to at an odd number
   d) Click to save your changes.

9. Add Streams Replication Manager or Kafka Connect to Heavy Duty clusters.

   Streams Replication Manager and Kafka Connect nodes are not provisioned by default when using the Streams Messaging Heavy Duty definition. If you are using this definition and want to have Streams Replication Manager, Kafka Connect, or both deployed in the cluster, set the Instance Count of the Srm Nodes or Connect Nodes host group to at least one.

   a) Go to Advanced Options Hardware and Storage .
   b) Locate the appropriate host group and click to edit host group details.
   c) Set the Instance Count to at least 1.

      Important: If you want to deploy Streams Replication Manager in high availability mode, set the instance count of Srm Nodes to two or higher.

   d) Click to save your changes.

10. Use the Advanced Options section to customize the infrastructure settings.

    Important: If you customize the hardware and storage properties of the cluster, ensure that Attached Volume per Instances is set to the same value for both the **Broker** and **Core_brokers** host groups. Alternatively, if you want to provision a cluster where the number of volumes is not identical for the two host groups, ensure that you complete *Configure data directories for clusters with custom disk configurations* after the cluster is provisioned. Otherwise, Kafka does not utilize all available volumes. Additionally, scaling the cluster might also not be possible.

11. Click Provision Cluster.

Results

You will be redirected to the Cloudera Data Hub cluster dashboard, and a new tile representing your cluster will appear at the top of the page.

What to do next

What steps you take next depends on how you configured your cluster

- If you have provisioned KRaft nodes in your cluster, you can optionally choose to remove ZooKeeper from your cluster. For more information, see *Deleting ZooKeeper from Streams Messaging clusters*.
- If you configured Attached Volume per Instances and the volume count is not identical for the **Broker** and **Core_brokers** host groups, continue with *Configuring data directories for clusters with custom disk configurations*.
- If you did not customize Attached Volume per Instances or do not want to remove ZooKeeper, continue with *Give users access to your cluster*.

Related Information

Create a cluster from a definition on AWS
Create a cluster from a definition on Azure
Create a cluster from a definition on GCP
Tags
Cloudera Data Hub cluster definitions
Streams Messaging cluster layout
Configuring data directories for clusters with custom disk configurations
Give users access to your cluster
Deleting ZooKeeper from Streams Messaging clusters

Deleting ZooKeeper from Streams Messaging clusters

The ZooKeeper instances deployed in Streams Messaging clusters can be manually deleted if the cluster uses KRaft for metadata management. Learn how to delete ZooKeeper from a Streams Messaging cluster.

About this task

If you use the default cluster definitions to deploy a Streams Messaging cluster, you have the option to provision the cluster with KRaft nodes. If KRaft nodes are deployed in a cluster, Kafka runs in KRaft mode and uses KRaft for metadata management instead of ZooKeeper. By default, clusters like this still include ZooKeeper instances. However, the ZooKeeper instances are not required and can be deleted. The ZooKeeper instances can be removed using Cloudera Manager.

Before you begin

Warning: Ensure that your cluster is provisioned with KRaft nodes. If you complete the following steps in a cluster that does not include KRaft nodes, the cluster will fail.

- Ensure that the cluster, its hosts, and all its services are healthy.
- Deleting the service does not remove the service data from your hosts. It only removes the service from management by Cloudera Manager. All role groups under the service are removed from host templates.

Procedure

1. Access the Cloudera Manager instance managing the cluster.
2. Remove ZooKeeper service dependencies:
   a) On the Cloudera Manager **Home** page, go to Configuration Service Dependencies .
   b) Locate and clear the ZooKeeper Service property.
   c) Click Save Changes.
3. Go to Clusters and select the ZooKeeper service.
4. Stop the ZooKeeper service:
   a) Select Actions Stop .
   b) Click Stop to confirm the action.
   c) Click Close after the process is finished.
5. Delete the ZooKeeper service:
   a) Select Actions Delete .
   b) Click Delete to confirm the action.

Results

The ZooKeeper service is deleted from the cluster.

What to do next

- If you configured Attached Volume per Instances and the volume count is not identical for the **Broker** and **Core_brokers** host groups, continue with *Configuring data directories for clusters with custom disk configurations*.
- If you did not customize Attached Volume per Instances, or customized it in a way that the volume count remained identical for the **Brokers** and **Core_brokers** host groups, continue with *Give users access to your cluster*.

Related Information

Give users access to your cluster

Configuring data directories for clusters with custom disk configurations

Configuring data directories for clusters with custom disk configurations

Cloudera Data Hub clusters provisioned with the default Streams Messaging cluster definitions that have non-identical volume numbers configured for the Broker and Core_brokers host groups require additional configuration after cluster provisioning. If configuration is not done, the Kafka service deployed on these clusters does not utilize all available volumes. Additionally, scaling the cluster might not be possible.

About this task

Clusters provisioned with version 7.2.12 or later of the Streams Messaging cluster definitions have two host groups that contain Kafka brokers (broker type groups). These are as follows:

- **Core_brokers**: This host group is provisioned by default, it contains a core set of brokers, and is not scalable.
- **Broker**: This host group is not provisioned by default (instance count is 0), but is scalable.

By default both broker type groups have the same disk configuration. The disk configuration can be changed during cluster provisioning in Advanced ConfigurationHardware and Storage. If you customize the Attached Volume per Instance property and the volume numbers set for the **Broker** and **Core_broker** groups are not identical, manually configuring the data directories (log.dirs) that Kafka uses is required after the cluster is provisioned.

If configuration is not done, Kafka does not utilize all volumes that are available. For example, assume you set Attached Volume per Instances to 2 for the **Broker** host group and 3 for the **Core_brokers** host group. In a case like this, Kafka by default utilizes two out of the three volumes. That is, it will utilize the lower number of volumes out of the two host group configurations. Additionally, scaling the cluster might also not be possible.

Important: Configuration is required even if you provision the cluster with no **Broker** host group. This is needed to avoid issues in case the **Broker** host group is upscaled at a later point in time.

Procedure

1. Access the Cloudera Manager instance managing the cluster.
2. Select the Kafka service.
3. Configure the Data Directories property for the default Kafka Broker role group.
   a) Go to Configuration.
   b) Find and configure the Data Directories property.

      The Data Directories property for the default role group must reflect the disk configuration (number of volumes) of the **Broker** host group. The property is configured by adding a unique data directory for each of the volumes. Each unique directory must be added as a separate entry. New entries can be added by clicking the + (add) icon next to the property. If configured correctly, the number of entries in the property should be identical with the number of volumes configured for the **Broker** host group.

      The data directories you specify must be /hadoopfs/fs*[***NUMBER***]*/kafka. For example, in case of two volumes, the data directories you need to specify are as follows:

      ```
      /hadoopfs/fs1/kafka
      /hadoopfs/fs2/kafka
      ```

   c) Click Save Changes.
4. Go to Instances.

5. Configure role level overrides for the Data Directories property.

   The following substeps must be repeated for each Kafka Broker role instance that is part of the **Core_brokers** host group. Ensure that the override you configure is identical for all Kafka Broker role instances.

   a) Select a Broker role instance that is deployed in the **Core_brokers** host group.
   b) Go to Configuration.
   c) Find and configure the Data Directories property.

      In this step you are configuring an override for the property. The value you set here is only used by the role instance you have selected in substep 5.a on page 9.

      Configure the override so that it reflects the disk configuration (number of volumes) of the **Core_brokers** host group. The property override is configured by adding a unique data directory for each of the volumes. Each unique directory must be added as a separate entry. New entries can be added by clicking the + (add) icon next to the property. If configured correctly, the number of entries in the property should be identical with the number of volumes configured for the **Core_brokers** host group.

      The data directories you specify must be /hadoopfs/fs*[***NUMBER***]*/kafka. For example, in case of three volumes, the data directories you need to specify are as follows:

      ```
      /hadoopfs/fs1/kafka
      /hadoopfs/fs2/kafka
      /hadoopfs/fs3/kafka
      ```

   d) Click Save Changes.

6. Review and verify your configuration.

   After both the default role group configuration and the overrides are specified, navigate to the service level configuration page of Kafka and find the Data Directories property. The property is displayed with the full configuration including the overrides. For example, if **Core_brokers** had three

volumes and **Broker** had two volumes, the correct configuration would be similar to the following example:

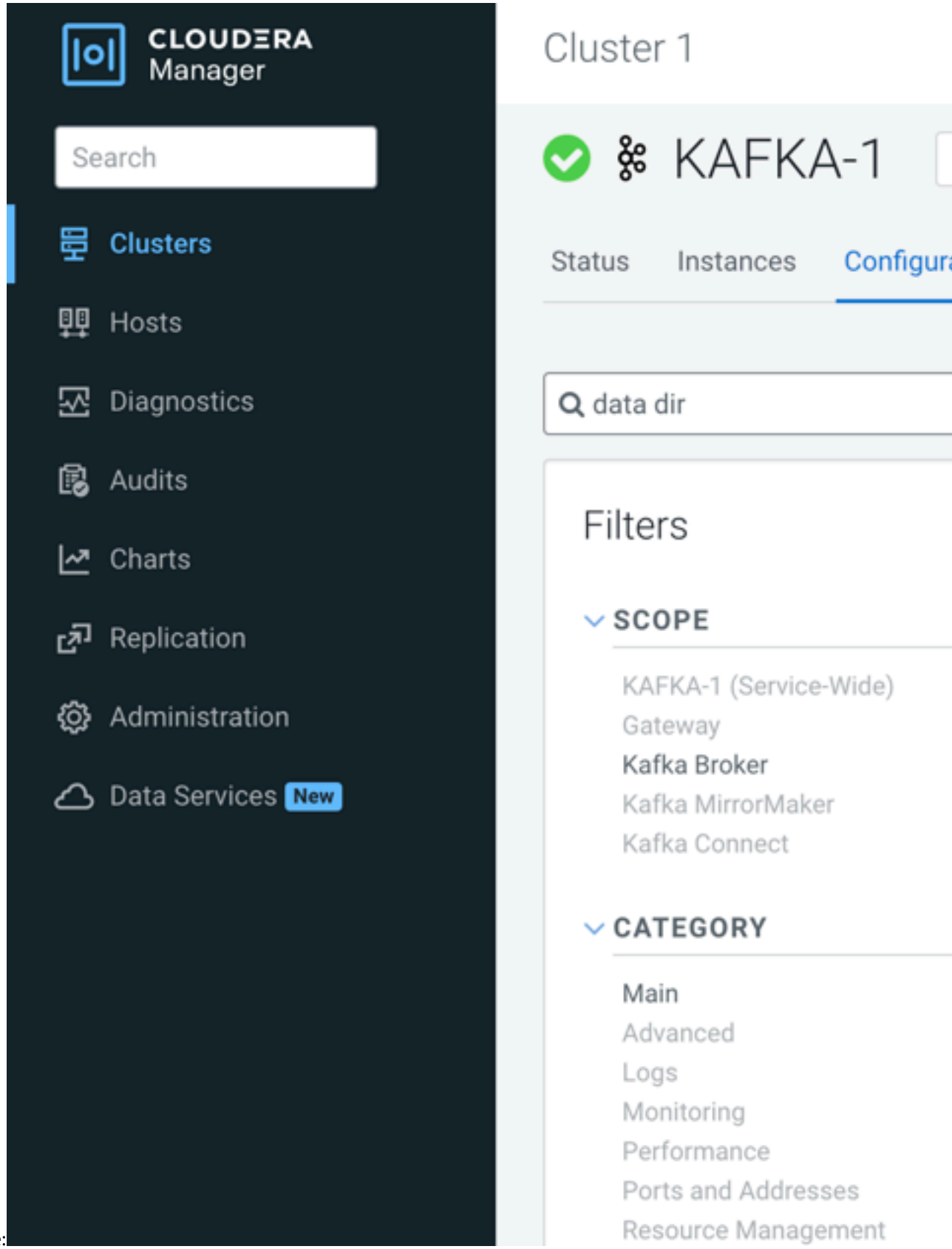

Results

Kafka data directories are configured. The Kafka service now uses all volumes available for the cluster.

What to do next

Continue with *Giving access to your cluster*.

Related Information

Give users access to your cluster

Giving access to your cluster

As an EnvironmentAdmin, you need provide the users access to your environment and to the new Streams Messaging cluster by assigning user roles and adding users to Ranger policies.

About this task

The cluster you have created using the Streams Messaging cluster definition is kerberized and secured with SSL. Users can access cluster UIs and endpoints through a secure gateway powered by Apache Knox. Before you can begin working with Kafka, Schema Registry, Streams Messaging Manager, and Streams Replication Manager, you must give users access to the Streams Messaging cluster components.

Procedure

1. Assign the EnvironmentUser role to the users to grant access to the Cloudera environment and the Streams Messaging cluster.
2. Sync the updated user permissions to the Data Lake (Ranger) using ActionsSynchronize Users to FreeIPA.

   This synchronizes the user to the FreeIPA identity management system to enable SSO.
3. Add the user to the appropriate pre-defined Ranger policies.

   These policies are specified within the Ranger instance that provides authorization to the Kafka service in your Streams Messaging cluster.

   If needed, you can also create a custom Ranger access policy and add the user to it.

What to do next

After you have created your Streams Messaging cluster, you are ready to start working with Kafka workloads in Cloudera Public Cloud.

Related Information

Enable authorization in Kafka with Ranger

Configure the resource-based Ranger service used for authorization

Cloudera DataFlow for Data Hub 7.3.1

Switching from Database Flow Persistence Provider to Git Flow Persistence Provider using NiFi CLI Export/Import all from NiFi Registry

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Exporting or importing data flows with NiFi Toolkit CLI..... 4
Flow persistance providers..... 4
Flow persistance providers..... 4
Switching flow persistance providers..... 5

Exporting or importing data flows with NiFi Toolkit CLI

NiFi CLI is a command line utility included in the NiFi Toolkit. You can use it to interact with NiFi and NiFi Registry instances to deploy versioned flows or manage process groups. You can also use the NiFi CLI to export and import flows in NiFi Registry when switching from one persistence provider to another, or for other use cases like syncing two different NiFi Registry instances.

In this use case, you can see how to switch from DatabaseFlowPersistenceProvider to GitFlowPersistenceProvider with the help of NiFi CLI. You can find details on how to:

- Export your flows from NiFi Registry using the CLI
- Clean up the databases
- Change the flow persistence provider
- Import all flows to your local Git repository
- Configure NiFi Registry so that versioned flows in NiFi are automatically saved to the Git repository.

Flow persistance providers

Persistence providers in NiFi Registry store and manage metadata and versioned flow information. Each type of versioned item, such as a versioned flow or extension bundle, has its own persistence provider. Each persistence provider has its own configuration parameters, which can be configured in an XML file specified in nifi-registry.properties.

Flow persistence providers store the content of the flows saved to the registry. In Cloudera DataFlow for Data Hub, the Flow Management Data Hub cluster is configured to use the database provider as the flow persistence provider by default. DatabaseFlowPersistenceProvider uses a relational database for storing and managing the versioned flow information and metadata. GitFlowPersistenceProvider stores flow contents in a Git repository, leveraging the capabilities of Git, such as version control, collaboration, and auditing.

The GitFlowPersistenceProvider has the following limitations:

- You cannot make changes to your flows in Git and commit these flows for NiFi Registry to pick up. All commits need to be done by NiFi Registry following actions through NiFi or NiFi Registry.
- You cannot have multiple NiFi Registry instances sharing the same Git repository because Registry does not read from Git but only pushes for persistence.
- Git repositories can grow in size over time, especially if they contain large files or a long history of commits. This can lead to increased storage requirements and slower operations, so it is important to manage repository size and consider strategies like using Git LFS (Large File Storage) for large binary files.

Flow persistance providers

Use this checklist to make sure that you meet all the requirements before you switch flow persistence providers.

- You have a Flow Management Data Hub cluster in your Cloudera Public Cloud environment.
- You have the necessary credentials and permissions to perform administrative tasks for NiFi Registry in Cloudera Manager.
- You have the host name of your NiFi Registry instance.
- NiFi CLI interactive shell is launched.
- You have configured Git.
    - You cloned a remote Git repository.
    - You prepared the necessary credentials.

Important:

To switch providers successfully, the flow content directory and the metadata database must be fully empty, including NiFi Registry buckets.

Switching flow persistance providers

This tutorial walks you through how to switch from DatabaseFlowPersistenceProvider to GitFlowPersistenceProvider with the help of NiFi CLI.

Procedure

1. Connect to your NiFi Registry instance through NiFi CLI using `ssh`.
2. Create the target directory where you will save the flows that you are exporting from NiFi Registry.

   For example: sudo mkdir /tmp/flowbkp

   Note: Make sure you have write access to the target directory.

3. Export the flows stored in NiFi Registry and save them in the directory you have created using the following command:

   ```
   sudo -u nifiregistry **[/PATH/TO/CLI.SH]** registry export-all-flows --out
   putDirectory "**[PATH/TO/DIRECTORY]**"
   ```

   For example:

   ```
   sudo -u nifiregistry /opt/cloudera/parcels/CFM/TOOLKIT/bin/cli.sh registry
    export-all-flows --outputDirectory "/tmp/flowbkp"
   ```

   Where:

   - sudo -u nifiregistry executes the subsequent command as the user nifiregistry.
   - /opt/cloudera/parcels/CFM/TOOLKIT/bin/cli.sh specifies the full path to the NiFi Toolkit CLI script.
   - registry export-all-flows instructs the CLI to export all flows and the associated resources from the NiFi Registry.
   - --outputDirectory "/tmp/flowbkp" specifies the output directory where the exported flows will be saved.
4. Stop NiFi Registry in Cloudera Manager.

   This ensures a clean transition and prevents conflicts or data inconsistencies during switching persistence providers.

   a) Click the Clusters tab in the left-hand navigation.
   b) Select the NiFi Registry service in the list of services to access the service dashboard page.
   c) Stop the service by opening the Actions drop-down menu next to the service name and clicking Stop.

   Note: It may take a few moments for the service to stop completely. You can verify that the NiFi Registry service is stopped by checking its status on the Cloudera Manager UI.

5. Back up the existing database.

   It is crucial to create a backup of your existing NiFi Registry database before making any changes. This ensures that you have a copy of the data in case any issues occur during switching persistence providers.

   a) Click the Configuration tab of the NiFi service dashboard page in Cloudera Manager.
   b) Locate the configuration property named nifi.registry.flow.storage.directory.

      This property specifies the directory where the Flow Storage files are stored for the FileSystemFlowPersistenceProvider. It is only required if Enable Flow File Provider is enabled. Otherwise, both flow content and metadata are stored in the database.
   c) Create a directory as a backup location, where you want to store the Flow Storage files.
   d) Update the flow storage directory to the backup location you created using the Providers: Default Flow Persistence File Provider Property - Flow Storage Directory property.

      You can search for ‘providers’ to find this property more easily. Make sure to specify the full absolute path to the new location.
   e) Use file system commands or tools to move the contents of the original Flow Storage directory to the new backup directory following your usual backup procedure.

      This involves copying all the files and directories from the original location to the new location while preserving the directory structure.
   f) Delete all data from the original database location.

      The NiFi Registry should be completely empty including the buckets.
6. Configure the NiFi Registry flow persistence provider properties in Cloudera Manager.

   a) Click the Configuration tab of the NiFi service dashboard page in Cloudera Manager.

      This tab displays the configuration settings for NiFi Registry.
   b) Disable the database provider by unclicking the Providers: Enable Database Flow Persistence Provider property.

      If Flow File Provider was enabled, it needs to be disabled as well using Enable Flow File Provider.
   c) Enable the Git-based provider by selecting the Providers: Enable the git-based flow persistence provider property.
   d) Configure the GitFlowPersistenceProvider provider.

      1. Set the path to your cloned Git repository using the Providers: File system path for a directory where flow contents files are persisted to property.

         This is the file system path for the directory where flow contents files are persisted to. The directory must exist when NiFi Registry starts, and it must be initialized as a Git directory.
      2. Define where you want to push the flow changes using the Providers: Name of the remote repository to push to property.

         When a new flow snapshot is created, files are updated in the specified Git directory. The GitFlowPersistenceProvider then creates a commit to the local repository. If `Remote to push` is defined, it also pushes to the specified remote repository.
      3. Provide a Git user name using the Providers: Git username for the remote repository property.

         This username should be used by the NiFi Registry to interact with the repository. It is used to make push requests to the remote repository when `Remote to push` is enabled, and the remote repository is accessed by HTTP protocol. If `ssh` is used, user authentication is done with SSH keys.
      4. Provide a Git password using the Providers: The password for the Remote Access User property.
   e) Save the configuration changes.
7. Start NiFi Registry to apply the new configuration.

   a) From Cloudera Manager, click the Clusters tab in the left-hand navigation.
   b) Select NiFi Registry.
   c) Start the service by opening the Actions drop-down menu next to the service name and clicking Start.
8. Ensure the user has full access to the Git repository and that the Git working tree is clean.

9. Import all flows from the directory where you saved your exported flows using the following command:

```
sudo -u nifiregistry **[/PATH/TO/CLI.SH]** registry import-all-flows --inp
ut "**[PATH/TO/DIRECTORY]**" --skipExisting
```

For example:

```
sudo -u nifiregistry /opt/cloudera/parcels/CFM/TOOLKIT/bin/cli.sh registry
 import-all-flows --input "/tmp/flowbkp" --skipExisting
```

Where:

- sudo -u nifiregistry executes the subsequent command as the user nifiregistry.
- /opt/cloudera/parcels/CFM/TOOLKIT/bin/cli.sh specifies the full path to the NiFi Toolkit CLI script.
- registry import-all-flows instructs the CLI to import all flows and the associated resources to the NiFi Registry.
- --input "/tmp/flowbkp" specifies the input directory from which the flows will be imported.
- --skipExisting indicates that any existing flows with the same UUIDs in the Registry should be skipped during the import process. It helps to avoid overwriting existing flows.

Note: Depending on your settings, `git push` may be required to finalize the change.

10. Verify and test:

Ensure that NiFi Registry is functioning as expected with the new flow persistence provider. Test the functionality, including creating and versioning flows, accessing metadata, and verifying data integrity.

11. Monitor and troubleshoot:

Monitor the NiFi Registry instance closely after the migration to ensure there are no unexpected issues. If any problems arise, check the documentation and troubleshooting resources provided by NiFi Registry and the chosen flow persistence provider.

Cloudera DataFlow for Data Hub 7.3.1

Exporting/importing data flows with NiFi Toolkit CLI

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Overview **4**

Connecting to NiFi Registry with NiFi Toolkit CLI **4**

Exporting a flow from NiFi Registry **6**

Importing a new flow into NiFi Registry **6**

Overview

NiFi CLI is a command line utility included in the NiFi Toolkit. You can use it to interact with NiFi and NiFi Registry instances to deploy versioned flows or manage process groups. You can also use it to export flows from or import flows to the NiFi Registry.

In this use case, you can see how to connect the NiFi CLI to a NiFi Registry running in a Flow Management Data Hub cluster using basic authentication, how to use the NiFi CLI to export a versioned flow from this Registry instance, and on how to import it into a different one.

Connecting to NiFi Registry with NiFi Toolkit CLI

In Cloudera Public Cloud, access from outside the cluster should go through the cluster's Knox gateway. Knox provides basic authentication to its API topologies, allowing you to authenticate with your Cloudera workload username and password.

About this task

Knox has two ports that can be accessed, 443 and 8443. This procedure shows you how to connect to the 8443 port of the Knox Gateway Server from outside the Flow Management Data Hub cluster (for example, from a local machine). This port is secured by Cloudera Manager AutoTLS. If you want to use the 443 port, you can use this procedure as a template and modify it for your production certs.

Before you begin

- You have a Flow Management Data Hub cluster in your Cloudera Public Cloud environment.
- You have the necessary permissions to create a NiFi Registry CLI properties file.

Procedure

1. Confirm or set your Cloudera workload username and password.

   To use non-SSO interfaces, each user and machine user must set a workload password. You can manage your workload password from your user profile page under User Management in the Cloudera Management Console. For instructions, see *Setting the workload password*.

   You will need to reference your workload username and password in the basic authentication parameters (basicAuthUsername, basicAuthPassword) in *Step 4*.
2. Identify the Knox Gateway Server in your Flow Management Data Hub cluster.

   You can find the hostname of the Knox Gateway Server in Cloudera Manager by checking the hostname in Clusters Knox Instances .

   Example hostname: flow-management-azure-management0.dataflow.xcu2-8y8x.dev.cldr.work

   Note: You can also find the Knox Gateway Server hostname on the Hardware tab of the Flow Management cluster overview page. It is the FQDN of the Management node.

3. Transfer the truststore JKS file (cm-auto-global_truststore.jks) from one of your Cloudera Data Hub nodes to your local machine.
   a) Connect to one of your NiFi nodes.
   b) Locate the truststore file.

      You can find it in the following directory:

      /var/run/cloudera-scm-agent/process/XYZ-nifi-NIFI_NODE/
   c) Get the truststore password with grep PASSWORD proc.json.
   d) Copy the cm-auto-global_truststore.jks file to your local machine.
4. Create a NiFi Registry CLI properties file referring to:
   - Knox Gateway Server (specifically pointing to port 8443)
   - Truststore JKS file
   - User and workload password using Basic Authentication

   You can use this properties file to store all your settings/default values for the Registry related CLI commands, so you do not need to specify the baseUrl and other arguments for every command.

   Example properties file datahub-registry-8443.properties:

```
baseUrl=https://flow-management-azure-management0.dataflow.xcu2-8y8x.de
v.cldr.work/flow-management-azure/cdp-proxy/nifi-registry-app
truststore=/Users/john.doe/Downloads/nifi-toolkit-1.13.2/bin/cm-auto-globa
l_truststore.jks
truststoreType=jks
truststorePasswd=password
basicAuthUsername=jdoe156
basicAuthPassword=password1
```

   Note: If you want to use the 443 port, you can use this procedure as a template and modify it for your production certs.

5. Check that the 8443 port is open.

   If the port is not open, configure the security policy in your given cloud provider to allow external traffic to reach port 8443.
6. Set your CLI session to refer to the NiFi Registry CLI properties file you created.

```
john.doe@john bin % ./cli.sh session set nifi.reg.props datahub-registry
-8443.properties
```

7. List the Registry buckets as a quick test for your connection.

   You get the list of available buckets in the NiFi Registry.

   In this example, the Registry has two buckets, TestProject and OtherProject:

```
john.doe@john bin % ./cli.sh registry list-buckets

# Name          Id                                      Description
- ---------     ------------------------------------    -----------
1 TestProject   2a5566b5-7380-46b8-9598-328e092e8899    (empty)
2 OtherProject  175fb557-43a2-4abb-871f-81a354f47bc2    (empty)
```

Related Information

Setting the workload password

Exporting a flow from NiFi Registry

This procedure shows you how to use the NiFi Toolkit CLI to export a versioned flow from a Registry instance.

Before you begin

- You have established a connection to the 8443 port of the Knox Gateway Server from outside the Flow Management Data Hub cluster.
- You have a flow in the Registry that you want to export.

Procedure

1. List the flows in the bucket that you want to explore.

   You get the list of the flows in the chosen bucket.

   In this example, the flows of the TestProject bucket are listed:

```
john.doe@john bin % ./cli.sh registry list-flows -bucketIdentifier 2a556
6b5-7380-46b8-9598-328e092e8899
# Name      Id                                      Description
- -------   ------------------------------------    -----------
1 BaseFlow  06acb207-d2f1-447f-85ed-9b8672fe6d30    First flow.
2 TestFlow  ce2fdaf9-bf81-43fa-846f-f38f6891be95    Second flow.
```

2. Export a flow from the Registry to a defined location using the flow ID.

   In this example, you export TestFlow as a JSON file called mytestflow.json to the tmp folder:

```
john.doe@john bin % ./cli.sh registry export-flow-version -f
          ce2fdaf9-bf81-43fa-846f-f38f6891be95 -o /tmp/mytestflow.json -
ot json
```

   where -f specifies the flow ID, -o specifies the name of the output file, and -ot specifies the output type.

   Note: The output types supported are json and simple. NiFi and NiFi Registry support flow definitions in JSON format.

Results

The flow is exported as a JSON file, so you have a NiFi flow definition on your local machine that you can import and deploy, for example in a different NiFi instance.

Importing a new flow into NiFi Registry

This procedure shows you how to import a brand new flow or a new version of a flow into a Registry instance.

Procedure

1. Create a new one or update the NiFi Registry CLI properties file that refers to the Registry of interest. Set your CLI session to refer to this NiFi Registry CLI properties file.

   For instructions, see Steps 4-6 in *Connecting to NiFi Registry with NiFi Toolkit CLI*.

2. Create a new flow.

```
john.doe@john bin % ./cli.sh registry create-flow -b 699ac53c-7be7-4082-
8711-f9fff823826c -fd "This is the flow description" -fn "This is the flow
 name"
```

```
b748827f-565f-4a69-8ce0-7218fcab35f4
```

where -b specifies the bucket ID in the Registry where you want to add the new flow, -fd specifies the description of the flow, and -fn specifies the name of the flow.

This generates a flow ID for an empty flow with an initial version of 0.

3. Import a flow definition to the Registry as version 1.

```
john.doe@john bin % ./cli.sh registry import-flow-version -f b748827f-56
5f-4a69-8ce0-7218fcab35f4 -i dh_export_test.json
1
```

where -f specifies the flow ID and -i specifies the name of the input file.

Note: If a flow already exists in the Registry, and you want to import a new version of the flow into Registry, you just need to reference the existing flow ID in the import command and the Registry will increment it as the next version.

For example:

```
john.doe@john bin % ./cli.sh registry import-flow-version -f 7ce2fdaf9-b
f81-43fa-846f-f38f6891be95 -i mytestflow.json
2
```

Related Information

Connecting to NiFi Registry with NiFi Toolkit CLI

Cloudera DataFlow for Data Hub 7.3.1

Cloudera DataFlow for Data Hub Release Notes

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What's New in Cloudera DataFlow for Data Hub 7.3.1 5

What's new in Flow Management with NiFi 1 5
What's new in Flow Management with NiFi 2 [Technical Preview] 5
What's new in Edge Management [Technical Preview] 6
What's New in Streams Messaging 6
What's New in Cloudera Streaming Analytics 8

Component support in Cloudera DataFlow for Data Hub 7.3.1 8

Supported NiFi extensions 9

Supported NiFi processors 9
Supported NiFi controller services 16
Supported NiFi reporting tasks 20
Supported NiFi parameter providers 21
Supported NiFi flow analysis rules [Technical Preview] 21
Supported NiFi Python components [Technical Preview] 22
Cloudera exclusive components [Technical Preview] 24
Components supported by partners 26

Unsupported Features in Cloudera DataFlow for Data Hub 7.3.1 27

Unsupported Flow Management features 27
Unsupported Edge Management features [Technical Preview] 27
Unsupported Streams Messaging features 27
Unsupported Cloudera Streaming Analytics features 28

Known Issues In Cloudera DataFlow for Data Hub 7.3.1 29

Known issues in Flow Management 29
Known issues in Edge Management [Technical Preview] 34
Known Issues in Streams Messaging 34
Known Issues in Cloudera Streaming Analytics 41

Deprecation notices In Cloudera DataFlow for Data Hub 7.3.1 43

Deprecation Notices for Cloudera Streaming Analytics 43

Fixed Issues in Cloudera DataFlow for Data Hub 7.3.1 44

Fixed issues in Flow Management 44
Fixed issues in Edge Management [Technical Preview] 55
Fixed Issues in Streams Messaging 55
Fixed Issues in Cloudera Streaming Analytics 56
UI fixes and improvements 57

Fixed CVEs in Cloudera DataFlow for Data Hub 7.3.1 57

CVE-2021-45105 & CVE-2021-44832 remediation for CDF for Data Hub....57
Fixed CVEs in Flow Management....58

Behavioral Changes in Cloudera DataFlow for Data Hub 7.3.1....62

Behavioral Changes in Flow Management....62
Behavioral Changes in Streams Messaging....67
Behavioral Changes in Cloudera Streaming Analytics....67

What's New in Cloudera DataFlow for Data Hub 7.3.1

Cloudera DataFlow for Data Hub 7.3.1 includes components for Flow Management, Edge Management, Streaming Analytics, and Streams Messaging. Learn about the new features and improvements in each of these components.

What's new in Flow Management with NiFi 1

Learn about the new features of Flow Management using NiFi 1.28 in Cloudera DataFlow for Data Hub 7.3.1.

Flow Management Data Hub clusters are compatible with both NiFi 1 and NiFi 2. The following sections provide details about Flow Management Data Hub based on Apache NiFi 1.28, with information on the most important new features, improvements, and fixes included in this release.

Rebase on NiFi 1.28

This upgrade offers access to the newest NiFi features and enhancements on the 1.x branch.

Important:

Direct upgrades from NiFi 1.28 to NiFi 2.0 are currently not supported. To transition, you can create new Flow Management clusters using NiFi 2-based templates and migrate your existing flows to the new clusters.

Note that NiFi 2.0 in Data Hub is currently in Technical Preview. NiFi 2-based clusters are not production-ready and should not be used for critical workloads.

New NiFi components

For a comprehensive list of supported NiFi components in Flow Management 7.3.1 Data Hub clusters, see Supported NiFi extensions.

For more information about the latest updates in Flow Management Data Hub clusters using NiFi 2, see What's new in Flow Management with NiFi 2.

What's new in Flow Management with NiFi 2 [Technical Preview]

Learn about the new features of Flow Management using NiFi 2.0 in Cloudera DataFlow for Data Hub 7.3.1.

Flow Management Data Hub clusters are compatible with both NiFi 1 and NiFi 2. The following sections provide details about Flow Management Data Hub based on Apache NiFi 2.0, with information on the most important new features, improvements, and fixes included in this release.

NiFi 2 introduces numerous changes from NiFi 1, including several breaking changes. See Behavioral changes for more information about the differences. Additionally, expect further breaking changes in future releases, particularly with components being removed in favor of more efficient alternatives.

Rebase on NiFi 2.0.0 M2

This upgrade offers access to the newest NiFi features and enhancements on the 2.x branch.

Important:

Direct upgrades from NiFi 1.28 to NiFi 2.0 are currently not supported. To transition, you can create new Flow Management clusters using NiFi 2-based templates and migrate your existing flows to the new clusters.

Note that NiFi 2.0 in Data Hub is currently in Technical Preview. NiFi 2-based clusters are not production-ready and should not be used for critical workloads.

Python processors

One of the key features introduced in Apache NiFi 2 is native support for Python processors. This capability allows you to create custom processors using Python, enabling seamless integration of Python scripts into your dataflows. With each milestone release of NiFi 2, Python integration continues to evolve, providing developers with enhanced functionality, greater flexibility, and more powerful tools for building robust dataflows.

The below list shows the Python processors that are available in Flow Management 7.3.1 clusters using NiFi 2.

- Bedrock
- ChunkData
- ChunkDocument
- EmbedData
- InsertToMilvus
- LexicalQueryMilvus
- ParseDocument
- PartitionCsv
- PartitionDocx
- PartitionHtml
- PartitionPdf
- PartitionText
- PromptChatGPT
- PutChroma
- PutOpenSearchVector
- PutPinecone
- PutQdrant
- QueryChroma
- QueryOpenSearchVector
- QueryPinecone
- QueryQdrant
- VectorQueryMilvus

For a comprehensive list of supported NiFi components in Flow Management 7.3.1 Data Hub clusters, see Supported NiFi extensions.

For more information about the latest updates in Flow Management Data Hub clusters using NiFi 1, see What's new in Flow Management with NiFi 1.

What's new in Edge Management [Technical Preview]

Learn about the Technical Preview for Light Duty Edge Flow Management cluster definitions introduced in Cloudera DataFlow for Data Hub 7.3.1 in Cloudera Public Cloud.

Edge Flow Management cluster definitions provide all Cloudera Edge Management 2.0.0 functionalities. This means an improved user experience with enhanced functionalities for seamless management and integration. Edge Flow Manager integrates with Cloudera User Management, making it easier to manage users and groups. For more information about the integration between Edge Flow Manager and Cloudera User Management, see After creating your cluster and Managing user groups using LDAP.

What's New in Streams Messaging

Learn about the new Streams Messaging features in Cloudera DataFlow for Data Hub 7.3.1.

Kafka

Kafka Rolling Restart check—all partitions fully replicated

A new broker rolling restart check option, all partitions fully replicated has been introduced. Selecting this option ensures that all partitions are in a fully synchronized state when a broker is stopped.

Schema Registry

Enable Streams Messaging Manager principal as trusted proxy user in Schema Registry

Streams Messaging Manager usually connects to Schema Registry on behalf of an end user. For requests coming from Streams Messaging Manager, Schema Registry can now extract and authorize the end user to authorize the request.

Streams Messaging Manager

Validation for duplicate property keys in Kafka Connect connector configuration

When validating Kafka Connect connector configurations, a warning is displayed if the configuration contains duplicate property keys. Duplicate property keys are highlighted with orange. The form can still be validated with the warnings present, but if there are duplicates, you are notified that only the value of the last occurrence is used.

Search supports regular expressions

The search component on the Topics, Brokers, Consumers, Producers page can now perform a regexp search.

Visual clue when restarting on Kafka Connect

When clicking restart on Kafka Connect tasks or connectors, a loading circle is displayed in case of synchronous calls. The loading circle disappears once a response is received. For asynchronous calls, a pop-up is displayed, stating that the task or connector is restarted.

UX improvements

- Fixed text overflow in the side panel column headers
- Listing page table headers are now sticky of the nested table headers
- Listing page table styling has been improved for readability
- Filter selector drop-downs are now styled consistently
- Sidebar menu pop-ups are no longer hidden under tables
- Class names on the Kafka Connect popup are now wrapped into the containing pop-ups
- The password field is no longer obfuscated when using a file provider as a password
- Fixed the alignment of values on the Connector metrics page
- Source and sink connectors are now separate tabs on the connector creation modal
- Fixed visual issues on the topic creation modal
- Increased consistency in element contrast and text style throughout the UI
- Active and Inactive statuses now have high contrast
- The expand icon is now consistent throughout the UI

Expand security-related headers set by SMM

The following security related headers were added to Streams Messaging Manager UI endpoints:

- Referrer-Policy
- Cross-Origin-Embedder-Policy
- Cross-Origin-Opener-Policy
- Cross-Origin-Resource-Policy

Streams Messaging Manager uses trusted proxy authentication when connecting to Schema Registry

You can only interact with schemas through SMM if the necessary Ranger policies are set up for Schema Registry. For SMM UI, you must have the correct permissions to check messages deserialized with Avro on Data Explorer.

Streams Replication Manager

The --to option in srm-control now creates the file if it does not exist

From now on, srm-control creates the file specified with the --to option if the file does not exist.

Cruise Control

Cruise Control is added to Streams Messaging Manager UI

A new page is added to Streams Messaging Manager to monitor the Kafka cluster state and rebalancing process with Cruise Control. The Cruise Control User Interface (UI) enables you to review and configure the rebalancing of Kafka clusters through dashboards and a rebalancing wizard. The available goals and anomaly detectors are based on the Cloudera Manager configurations of Cruise Control. You can access Cruise Control from Streams Messaging Manager using the on the navigation sidebar.

For more information about Cruise Control in Streams Messaging Manager, see Monitoring and managing Kafka cluster rebalancing.

What's New in Cloudera Streaming Analytics

Learn about the new Cloudera Streaming Analytics features in Cloudera DataFlow for Cloudera Data Hub 7.3.1.

The following new features are introduced in Cloudera Streaming Analytics 7.3.1:

Rebase to Apache Flink 1.19.1

Apache Flink 1.19.1 is supported in the Cloudera Streaming Analytics 7.3.1 cluster definition.

For more information on what is included in the Apache Flink 1.19.1 version, see the Apache Flink 1.19.1 Release Announcementand Release Notes.

Support for Python UDFs in Cloudera SQL Stream Builder

This feature allows customers to start using Python for creating User-Defined Functions (UDFs). Cloudera recommends that customers start using Python UDFs for all new developments, and start migrating their JavaScript UDFs to Python to prepare for future upgrades, as Javascript UDFs will be removed in the future due to the deprecation of the Nashorn engine used in JDK 8 and 11.

For more information on using Python UDFs, see Python UDFs. For more information on supported JDK versions, refer to the Support Matrix.

Global logging configuration for Cloudera SQL Stream Builder jobs

A new global settings view has been enabled, which currently includes log4j configuration of Flink jobs started on SSB. Users with SSB administrator rights can set a default logging configuration applied to all SSB jobs , which can be overridden at the job level.

For more information see Adjusting logging configuration in Advanced Settings.

Customizable default Kafka TrustStore configuration in Streaming SQL Console

Customizing default Kafka TrustStore configurations was added to Streaming SQL Console. Kafka TrustStore can be configured during adding Kafka as a Data Source on the UI.

Component support in Cloudera DataFlow for Data Hub 7.3.1

Cloudera DataFlow for Data Hub 7.3.1 includes the following components.

Flow Management clusters with NiFi 1

- Apache NiFi 1.28.1.2.2.9.0

- Apache NiFi Registry 1.28.1.2.2.9.0
- Schema Registry 0.10.0.7.3.1.0

Flow Management clusters with NiFi 2 [Technical Preview]

- Apache NiFi 2.0.0.4.2.1.0
- Apache NiFi Registry 2.0.0.4.2.1.0
- Schema Registry 0.10.0.7.3.1.0

Edge Management clusters [Technical Preview]

- Edge Flow Manager 2.2.99

Streams Messaging clusters

- Apache Kafka 3.4.0
- Schema Registry 0.10.0
- Streams Messaging Manager 2.3.0
- Streams Replication Manager 1.1.0
- Cruise Control 2.5.85

Streaming Analytics clusters

- Apache Flink 1.19.1

Supported NiFi extensions

Apache NiFi versions 1.28 and 2.0 offer a comprehensive set of extensions, with the majority fully supported by Cloudera. To ensure seamless operation and reliable support, it is recommended to avoid using unsupported extensions in production environments.

The following sections provide detailed information on the supported extensions available for both NiFi versions.

Supported NiFi processors

This release is based on Apache NiFi versions 1.28 and 2.0, with added support for Python processors in NiFi 2. Most processors included in these NiFi versions are supported by Cloudera.

To ensure optimal performance and reliable support, it is crucial to use only supported processors and avoid deploying unsupported ones in production environments.

Additional processors are developed and tested by the community but are not officially supported by Cloudera. Processors may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

By adhering to the above guidelines, you can maintain stable and reliable workflows in your production environments.

NiFi 1.28

AttributesToCSV	GetElasticsearch	PutDropbox
AttributesToJSON	GetFile	PutDynamoDB
Base64EncodeContent	GetFTP	PutDynamoDBRecord
CalculateParquetOffsets	GetGcpVisionAnnotateFilesOperationStatus	PutElasticsearchHttp1
CalculateParquetRowGroupOffsets	GetGcpVisionAnnotateImagesOperationStatus	PutElasticsearchHttpRecord1
CalculateRecordStats	GetHBase	PutElasticsearchJson

CaptureChangeDebeziumDB2 [Technical Preview]	GetHDFS	PutElasticsearchRecord1
CaptureChangeDebeziumMySQL [Technical Preview]	GetHDFSFileInfo	PutEmail
CaptureChangeDebeziumOracle [Technical Preview]	GetHDFSSequenceFile	PutFile
CaptureChangeDebeziumPostgreSQL [Technical Preview]	GetHTMLElement	PutFTP1
CaptureChangeDebeziumSQLServer [Technical Preview]	GetHTTP	PutGCSObject
CaptureChangeMySQL	GetHubSpot	PutGoogleDrive
CompressContent1, 2	GetIgniteCache	PutGridFS
ConnectWebSocket	GetJiraIssue	PutHBaseCell
ConsumeAMQP	GetJMSQueue	PutHBaseJSON
ConsumeAzureEventHub	GetJMSTopic	PutHBaseRecord1
ConsumeElasticsearch	GetMongoRecord	PutHDFS
ConsumeEWS	GetSFTP	PutHive3QL
ConsumeGCPubSub	GetShopify	PutHive3Streaming
ConsumeGCPubSubLite	GetSNMP	PutHiveQL
ConsumeJMS	GetSnowflakeIngestStatus	PutHiveStreaming
ConsumeKafka_1_0	GetSolr	PutHTMLElement
ConsumeKafka_2_0	GetSplunk	PutIceberg
ConsumeKafka_2_6	GetSQS	PutIcebergCDC [Technical Preview]
ConsumeKafka2CDP	GetTCP	PutInfluxDB
ConsumeKafka2RecordCDP	GetTwitter	PutJiraIssue
ConsumeKafkaRecord_1_0	GetWorkdayReport	PutJMS1
ConsumeKafkaRecord_2_0	GetZendesk	PutKinesisFirehose
ConsumeKafkaRecord_2_6	HandleHttpRequest	PutKinesisStream
ConsumeKinesisStream	HandleHttpResponse	PutKudu
ConsumeMQTT1	HashAttribute	PutLambda
ConsumeTwitter	HashContent	PutMongoRecord
ConsumeWindowsEventLog	IdentifyMimeType	PutORC1
ControlRate	InvokeAWSGatewayApi	PutParquet
ConvertAvroSchema	InvokeGRPC	PutRecord
ConvertAvroToJSON	InvokeGRPC	PutRedisHashRecord [Technical Preview]
ConvertAvroToORC	InvokeHTTP	PutRiemann
ConvertAvroToParquet	InvokeScriptedProcessor	PutS3Object
ConvertCharacterSet	JoinEnrichment	PutSalesforceObject
ConvertCSVToAvro	JoltTransformJSON	PutSFTP
ConvertJSONToAvro	JoltTransformRecord	PutSmbFile
ConvertJSONToSQL	JSLTTransformJSON	PutSnowflakeInternalStage
ConvertProtobuf	JsonQueryElasticsearch	PutSNS

ConvertRecord	ListAzureBlobStorage	PutSolrContentStream
CreateHadoopSequenceFile	ListAzureBlobStorage_v12	PutSolrRecord
CryptographicHashAttribute	ListAzureDataLakeStorage	PutSplunk
CryptographicHashContent	ListBoxFile	PutSplunkHTTP
DecryptContent	ListCDPObjectStore	PutSQL
DecryptContentAge	ListDatabaseTables	PutSQS1
DecryptContentCompatibility	ListDropbox	PutSyslog
DecryptContentPGP	ListenBeats	PutTCP
DeduplicateRecord	ListenFTP	PutUDP
DeleteAzureBlobStorage	ListenGRPC	PutWebSocket
DeleteAzureBlobStorage_v12	ListenGRPC	PutZendeskTicket
DeleteAzureDataLakeStorage	ListenHTTP	QueryAirtableTable
DeleteByQueryElasticsearch	ListenNetFlow	QueryCassandra
DeleteCDPObjectStore	ListenOTLP	QueryDatabaseTable1
DeleteDynamoDB	ListenRELP	QueryDatabaseTableRecord
DeleteGCSObject	ListenSyslog	QueryElasticsearchHttp
DeleteGridFS	ListenTCP	QueryRecord
DeleteHBaseCells	ListenTCPRecord	QuerySalesforceObject
DeleteHBaseRow	ListenTrapSNMP	QuerySolr
DeleteHDFS	ListenUDP	QuerySplunkIndexingStatus
DeleteS3Object	ListenUDPRecord	QueryWhois
DeleteSQS	ListenWebSocket	RemoveRecordField
DetectDuplicate	ListFile	ReplaceText
DistributeLoad	ListFTP	ReplaceTextWithMapping
DuplicateFlowFile	ListGCSBucket	ResizeImage1
EncodeContent	ListGoogleDrive	RetryFlowFile
EncryptContent2	ListHDFS	RouteHL7
EncryptContentAge	ListS3	RouteOnAttribute
EncryptContentPGP	ListSFTP	RouteOnContent
EnforceOrder	ListSmb	RouteText
EvaluateJsonPath	LogAttribute	SampleRecord
EvaluateXPath	LogMessage	ScanAccumulo
EvaluateXQuery	LookupAttribute	ScanAttribute1
ExecuteGroovyScript	LookupRecord	ScanContent
ExecuteInfluxDBQuery	MergeContent	ScanHBase
ExecuteProcess	MergeRecord1	ScriptedFilterRecord
ExecuteScript	ModifyCompression	ScriptedPartitionRecord
ExecuteSQL	ModifyHTMLElement	ScriptedTransformRecord
ExecuteSQLRecord	MonitorActivity	ScriptedValidateRecord
ExecuteStateless12	MoveAzureDataLakeStorage	ScrollElasticsearchHttp

ExecuteStreamCommand	MoveHDFS	SearchElasticsearch
ExtractAvroMetadata	Notify	SegmentContent
ExtractGrok	PackageFlowFile	SelectClouderaHiveQL
ExtractHL7Attributes	PaginatedJsonQueryElasticsearch	SelectHive3QL1
ExtractImageMetadata	ParseCEF1	SelectHiveQL
ExtractRecordSchema	ParseEvtx	SendTrapSNMP
ExtractText	ParseSyslog	SetSNMP
FetchAzureBlobStorage	PartitionRecord	SignContentPGP
FetchAzureBlobStorage_v12	PostHTTP	SplitAvro
FetchAzureDataLakeStorage	PublishAMQP	SplitContent
FetchBoxFile	PublishGCPubSub1	SplitJson1
FetchCDPObjectStore	PublishGCPubSubLite1	SplitRecord1
FetchDistributedMapCache	PublishJMS1	SplitText1
FetchDropbox	PublishKafka_1_0	SplitXml
FetchElasticsearchHttp	PublishKafka_2_0	StartAwsPollyJob
FetchFile	PublishKafka_2_6	StartAwsTextractJob
FetchFTP	PublishKafka2CDP	StartAwsTranscribeJob
FetchGCSObject	PublishKafka2RecordCDP	StartAwsTranslateJob
FetchGoogleDrive	PublishKafkaRecord_1_0	StartGcpVisionAnnotateFilesOperation
FetchGridFS	PublishKafkaRecord_2_0	StartGcpVisionAnnotateImagesOperation
FetchHBaseRow	PublishKafkaRecord_2_6	StartSnowflakeIngest
FetchHDFS	PublishMQTT	TagS3Object
FetchParquet	PublishSlack	TailFile
FetchS3Object	PutAccumuloRecord1	TransformXml
FetchSFTP	PutAzureBlobStorage	TriggerClouderaHiveMetaStoreEvent
FetchSmb	PutAzureBlobStorage_v12	TriggerHiveMetaStoreEvent
FilterAttribute	PutAzureCosmosDBRecord	UnpackContent
FlattenJson	PutAzureDataLakeStorage1	UpdateAttribute
ForkEnrichment	PutAzureEventHub	UpdateByQueryElasticsearch
ForkRecord	PutAzureQueueStorage1	UpdateClouderaHiveTable
GenerateFlowFile	PutAzureQueueStorage_v12	UpdateCounter
GenerateRecord	PutBigQuery	UpdateDatabaseTable
GenerateTableFetch	PutBigQueryBatch	UpdateDeltaLakeTable [Technical Preview]
GeoEnrichIP	PutBigQueryStreaming	UpdateHive3Table
GeoEnrichIPRecord	PutBoxFile	UpdateHiveTable
GeohashRecord	PutCassandraQL1	UpdateRecord
GetAsanaObject	PutCassandraRecord1	ValidateCsv
GetAwsPollyJobStatus	PutCDPObjectStore	ValidateJson
GetAwsTextractJobStatus	PutClouderaHiveQL	ValidateRecord
GetAwsTranscribeJobStatus	PutClouderaHiveStreaming	ValidateXml

GetAwsTranslateJobStatus	PutClouderaORC	VerifyContentMAC
GetAzureEventHub	PutCloudWatchMetric	VerifyContentPGP
GetAzureQueueStorage	PutCouchbaseKey	Wait
GetAzureQueueStorage_v12	PutDatabaseRecord1	YandexTranslate
GetCouchbaseKey1	PutDistributedMapCache	

Footnotes

- 1 – indicates a memory-intensive processor
- 2 – indicates a CPU-intensive processor

NiFi 2.0

AttributesToCSV	GetElasticsearch	PutFile
AttributesToJSON	GetFile	PutFTP1
CalculateParquetOffsets	GetFTP	PutGCSObject
CalculateParquetRowGroupOffsets	GetGcpVisionAnnotateFilesOperationStatus	PutGoogleDrive
CalculateRecordStats	GetGcpVisionAnnotateImagesOperationStatus	PutGridFS
CaptureChangeDebeziumDB2 [Technical Preview]	GetHBase	PutHBaseCell
CaptureChangeDebeziumMySQL [Technical Preview]	GetHDFS	PutHBaseJSON
CaptureChangeDebeziumOracle [Technical Preview]	GetHDFSFileInfo	PutHBaseRecord1
CaptureChangeDebeziumPostgreSQL [Technical Preview]	GetHDFSSequenceFile	PutHDFS
CaptureChangeDebeziumSQLServer [Technical Preview]	GetHubSpot	PutHive3QL
CaptureChangeMySQL	GetJiraIssue	PutHive3Streaming
ChunkDocument [Technical Preview]	GetMongoRecord	PutIceberg
CompressContent1, 2	GetSFTP	PutIcebergCDC [Technical Preview]
ConnectWebSocket	GetShopify	PutJiraIssue
ConsumeAMQP	GetSNMP	PutKinesisFirehose
ConsumeAzureEventHub	GetSnowflakeIngestStatus	PutKinesisStream
ConsumeElasticsearch	GetSolr	PutKudu
ConsumeGCPubSub	GetSplunk	PutLambda
ConsumeGCPubSubLite	GetSQS	PutMongoBulkOperations
ConsumeJMS	GetWorkdayReport	PutMongoRecord
ConsumeKafka_2_6	GetZendesk	PutORC1
ConsumeKafka2CDP	HandleHttpRequest	PutParquet
ConsumeKafka2RecordCDP	HandleHttpResponse	PutPinecone [Technical Preview]
ConsumeKafkaRecord_2_6	IdentifyMimeType	PutPLC [Technical Preview]
ConsumeKinesisStream	InvokeAWSGatewayApi	PutRecord
ConsumeMQTT1	InvokeGRPC	PutRedisHashRecord [Technical Preview]
ConsumePLC [Technical Preview]	InvokeHTTP	PutS3Object

ConsumeSlack	InvokeScriptedProcessor	PutSalesforceObject
ConsumeTwitter	JoinEnrichment	PutSFTP
ConsumeWindowsEventLog	JoltTransformJSON	PutSmbFile
ControlRate	JoltTransformRecord	PutSnowflakeInternalStage
ConvertAvroToJSON	JSLTTransformJSON	PutSNS
ConvertAvroToParquet	JsonQueryElasticsearch	PutSolrContentStream
ConvertCharacterSet	ListAzureBlobStorage_v12	PutSolrRecord
ConvertJSONToSQL	ListAzureDataLakeStorage	PutSplunk
ConvertProtobuf	ListBoxFile	PutSplunkHTTP
ConvertRecord	ListCDPObjectStore	PutSQL
CopyAzureBlobStorage_v12	ListDatabaseTables	PutSQS1
CreateHadoopSequenceFile	ListDropbox	PutSyslog
CryptographicHashContent	ListenBeats	PutTCP
DecryptContent	ListenFTP	PutUDP
DecryptContentAge	ListenGRPC	PutWebSocket
DecryptContentCompatibility	ListenHTTP	PutZendeskTicket
DecryptContentPGP	ListenNetFlow	QueryAirtableTable
DeduplicateRecord	ListenOTLP	QueryCassandra
DeleteAzureBlobStorage_v12	ListenRELP	QueryChroma [Technical Preview]
DeleteAzureDataLakeStorage	ListenSlack	QueryDatabaseTable1
DeleteByQueryElasticsearch	ListenSyslog	QueryDatabaseTableRecord
DeleteCDPObjectStore	ListenTCP	QueryPinecone [Technical Preview]
DeleteDynamoDB	ListenTCPRecord	QueryRecord
DeleteGCSObject	ListenTrapSNMP	QuerySalesforceObject
DeleteGridFS	ListenUDP	QuerySolr
DeleteHBaseCells	ListenUDPRecord	QuerySplunkIndexingStatus
DeleteHBaseRow	ListenWebSocket	QueryWhois
DeleteHDFS	ListFile	RemoveRecordField
DeleteS3Object	ListFTP	RenameRecordField
DeleteSQS	ListGCSBucket	ReplaceText
DetectDuplicate	ListGoogleDrive	ReplaceTextWithMapping
DistributeLoad	ListHDFS	ResizeImage1
DuplicateFlowFile	ListS3	RetryFlowFile
EncodeContent	ListSFTP	RouteHL7
EncryptContentAge	ListSmb	RouteOnAttribute
EncryptContentPGP	LogAttribute	RouteOnContent
EnforceOrder	LogMessage	RouteText
EvaluateJsonPath	LookupAttribute	SampleRecord
EvaluateXPath	LookupRecord	ScanAccumulo
EvaluateXQuery	MergeContent	ScanAttribute1

ExecuteGroovyScript	MergeRecord1	ScanContent
ExecuteProcess	ModifyCompression	ScanHBase
ExecuteScript	MonitorActivity	ScriptedFilterRecord
ExecuteSQL	MoveAzureDataLakeStorage	ScriptedPartitionRecord
ExecuteSQLRecord	MoveHDFS	ScriptedTransformRecord
ExecuteStateless1, 2	Notify	ScriptedValidateRecord
ExecuteStreamCommand	PackageFlowFile	SearchElasticsearch
ExtractAvroMetadata	PaginatedJsonQueryElasticsearch	SegmentContent
ExtractGrok	ParseCEF1	SelectClouderaHiveQL
ExtractHL7Attributes	ParseDocument [Technical Preview]	SelectHive3QL1
ExtractImageMetadata	ParseEvtx	SendTrapSNMP
ExtractRecordSchema	ParseSyslog	SetSNMP
ExtractText	PartitionRecord	SignContentPGP
FetchAzureBlobStorage_v12	PromptChatGPT [Technical Preview]	SplitAvro
FetchAzureDataLakeStorage	PublishAMQP	SplitContent
FetchBoxFile	PublishGCPubSub1	SplitJson1
FetchCDPObjectStore	PublishGCPubSubLite1	SplitRecord1
FetchDistributedMapCache	PublishJMS1	SplitText1
FetchDropbox	PublishKafka_2_6	SplitXml
FetchFile	PublishKafka2CDP	StartAwsPollyJob
FetchFTP	PublishKafka2RecordCDP	StartAwsTextractJob
FetchGCSObject	PublishKafkaRecord_2_6	StartAwsTranscribeJob
FetchGoogleDrive	PublishMQTT	StartAwsTranslateJob
FetchGridFS	PublishSlack	StartGcpVisionAnnotateFilesOperation
FetchHBaseRow	PutAccumuloRecord1	StartGcpVisionAnnotateImagesOperation
FetchHDFS	PutAzureBlobStorage_v12	StartSnowflakeIngest
FetchParquet	PutAzureCosmosDBRecord	TagS3Object
FetchPLC [Technical Preview]	PutAzureDataLakeStorage1	TailFile
FetchS3Object	PutAzureEventHub	TransformXml
FetchSFTP	PutAzureQueueStorage_v12	TriggerClouderaHiveMetaStoreEvent
FetchSmb	PutBigQuery	TriggerHiveMetaStoreEvent
FilterAttribute	PutBoxFile	UnpackContent
FlattenJson	PutCassandraQL1	UpdateAttribute
ForkEnrichment	PutCassandraRecord1	UpdateByQueryElasticsearch
ForkRecord	PutCDPObjectStore	UpdateClouderaHiveTable
GenerateFlowFile	PutChroma [Technical Preview]	UpdateCounter
GenerateRecord	PutClouderaHiveQL	UpdateDatabaseTable
GenerateTableFetch	PutClouderaHiveStreaming	UpdateDeltaLakeTable [Technical Preview]
GeoEnrichIP	PutClouderaORC	UpdateHive3Table
GeoEnrichIPRecord	PutCloudWatchMetric	UpdateRecord

GeohashRecord	PutCouchbaseKey	ValidateCsv
GetAsanaObject	PutDatabaseRecord1	ValidateJson
GetAwsPollyJobStatus	PutDistributedMapCache	ValidateRecord
GetAwsTextractJobStatus	PutDropbox	ValidateXml
GetAwsTranscribeJobStatus	PutDynamoDB	VerifyContentMAC
GetAwsTranslateJobStatus	PutDynamoDBRecord	VerifyContentPGP
GetAzureEventHub	PutElasticsearchJson	Wait
GetAzureQueueStorage_v12	PutElasticsearchRecord1	YandexTranslate
GetCouchbaseKey1	PutEmail	

Footnotes

- 1 – indicates a memory-intensive processor
- 2 – indicates a CPU-intensive processor

Supported NiFi controller services

This release is based on Apache NiFi versions 1.28 and 2.0, and includes a set of controller services. Most controller services included these NiFi versions are supported by Cloudera.

To ensure optimal performance and reliable support, it is crucial to use only supported controller services and avoid deploying unsupported ones in production environments.

Additional controller services are developed and tested by the community but are not officially supported by Cloudera. Controller services may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

By adhering to the above guidelines, you can maintain stable and reliable workflows in your production environments.

NiFi 1.28

AccumuloService	ImpalaConnectionPool
ActionHandlerLookup	IPFIXReader
ActiveMQJMSConnectionFactoryProvider	IPLookupService
ADLSCredentialsControllerService	JASN1Reader
ADLSCredentialsControllerServiceLookup	JiraRecordSink
ADLSIDBrokerCloudCredentialsProviderControllerService	JMSConnectionFactoryProvider
AlertHandler	JndiJmsConnectionFactoryProvider
AmazonGlueSchemaRegistry	JsonConfigBasedBoxClientService
AvroReader	JsonPathReader
AvroRecordSetWriter	JsonRecordSetWriter
AvroSchemaRegistry	JsonTreeReader
AWSCredentialsProviderControllerService	KafkaRecordSink_1_0
AWSIDBrokerCloudCredentialsProviderControllerService	KafkaRecordSink_2_0
AzureBlobIDBrokerCloudCredentialsProviderControllerService	KafkaRecordSink_2_6
AzureCosmosDBClientService	KerberosKeytabUserService
AzureEventHubRecordSink	KerberosPasswordUserService

AzureServiceBusJMSConnectionFactoryProvider	KerberosTicketCacheUserService
AzureStorageCredentialsControllerService	KeytabCredentialsService
AzureStorageCredentialsControllerService_v12	KuduLookupService
AzureStorageCredentialsControllerServiceLookup	LoggingRecordSink
AzureStorageCredentialsControllerServiceLookup_v12	LogHandler
CassandraDistributedMapCache	MongoDBControllerService
CassandraSessionProvider	MongoDBLookupService
CdpCredentialsProviderControllerService	ParquetReader
CdpOauth2AccessTokenProviderControllerService	ParquetRecordSetWriter
CEFReader	PostgreSQLConnectionPool
CiscoEmblemSyslogMessageReader	PrometheusRecordSink
ClouderaHiveConnectionPool	ProtobufReader
ClouderaSchemaRegistry	RabbitMQJMSConnectionFactoryProvider
CMLLookupService	ReaderLookup
ConfluentSchemaRegistry	RecordSetWriterLookup
CouchbaseClusterService	RecordSinkHandler
CouchbaseKeyValueLookupService	RecordSinkServiceLookup
CouchbaseMapCacheClient	RedisConnectionPoolService
CouchbaseRecordLookupService	RedisDistributedMapCacheClientService
CSVReader	RedshiftConnectionPool
CSVRecordLookupService	RestLookupService
CSVRecordSetWriter	ScriptedActionHandler
DatabaseRecordLookupService	ScriptedLookupService
DatabaseRecordSink	ScriptedReader
DatabaseTableSchemaRegistry	ScriptedRecordSetWriter
DBCPConnectionPool	ScriptedRecordSink
DBCPConnectionPoolLookup	ScriptedRulesEngine
DistributedMapCacheClientService	SimpleDatabaseLookupService
DistributedMapCacheLookupService	SimpleKeyValueLookupService
DistributedMapCacheServer	SimpleRedisDistributedMapCacheClientService
DistributedSetCacheClientService	SimpleScriptedLookupService
DistributedSetCacheServer	SiteToSiteReportingRecordSink
EasyRulesEngineProvider	SmbjClientProviderService
EasyRulesEngineService	SnowflakeComputingConnectionPool
EBCDICRecordReader [Technical Preview]	StandardAsanaClientProviderService
ElasticSearchClientServiceImpl	StandardAzureCredentialsControllerService
ElasticSearchLookupService	StandardDropboxCredentialService
ElasticSearchStringLookupService	StandardFileResourceService
EmailRecordSink	StandardHashiCorpVaultClientService
EmbeddedHazelcastCacheManager	StandardHttpContextMap

ExcelReader	StandardJsonSchemaRegistry [Technical Preview]
ExpressionHandler	StandardOauth2AccessTokenProvider
ExternalHazelcastCacheManager	StandardPGPPrivateKeyService
FreeFormTextRecordSetWriter	StandardPGPPublicKeyService
GCPCredentialsControllerService	StandardPrivateKeyService
GrokReader	StandardProxyConfigurationService
HadoopCatalogService	StandardRestrictedSSLContextService
HadoopDBCPConnectionPool	StandardS3EncryptionService
HazelcastMapCacheClient	StandardSnowflakeIngestManagerProviderService
HBase_1_1_2_ClientMapCacheService	StandardSSLContextService
HBase_1_1_2_ClientService	StandardWebClientServiceProvider
HBase_1_1_2_ListLookupService	Syslog5424Reader
HBase_1_1_2_RecordLookupService	SyslogReader
HBase_2_ClientMapCacheService	UDPEventRecordSink
HBase_2_ClientService	VolatileSchemaCache
HBase_2_RecordLookupService	WindowsEventLogReader
Hive3ConnectionPool	XMLReader
HiveCatalogService	XMLRecordSetWriter
HiveConnectionPool	YamlTreeReader
HortonworksSchemaRegistry	ZendeskRecordSink

NiFi 2.0

AccumuloService	IPLookupService
ActiveMQJMSConnectionFactoryProvider	JASN1Reader
ADLSCredentialsControllerService	JiraRecordSink
ADLSCredentialsControllerServiceLookup	JMSConnectionFactoryProvider
ADLSIDBrokerCloudCredentialsProviderControllerService	JndiJmsConnectionFactoryProvider
AmazonGlueSchemaRegistry	JsonConfigBasedBoxClientService
ApicurioSchemaRegistry	JsonPathReader
AvroReader	JsonRecordSetWriter
AvroRecordSetWriter	JsonTreeReader
AvroSchemaRegistry	KafkaRecordSink_2_6
AWSCredentialsProviderControllerService	KerberosKeytabUserService
AWSIDBrokerCloudCredentialsProviderControllerService	KerberosPasswordUserService
AzureBlobIDBrokerCloudCredentialsProviderControllerService	KerberosTicketCacheUserService
AzureCosmosDBClientService	KeytabCredentialsService
AzureEventHubRecordSink	KuduLookupService
AzureServiceBusJMSConnectionFactoryProvider	LoggingRecordSink
AzureStorageCredentialsControllerService_v12	MongoDBControllerService
AzureStorageCredentialsControllerServiceLookup_v12	MongoDBLookupService

CassandraDistributedMapCache	ParquetReader
CassandraSessionProvider	ParquetRecordSetWriter
CdpCredentialsProviderControllerService	PostgreSQLConnectionPool
CdpOauth2AccessTokenProviderControllerService	PrometheusRecordSink
CEFReader	ProxyPLC4XConnectionPool [Technical Preview]
CiscoEmblemSyslogMessageReader	RabbitMQJMSConnectionFactoryProvider
ClouderaHiveConnectionPool	ReaderLookup
ClouderaSchemaRegistry	RecordSetWriterLookup
CMLLookupService	RecordSinkServiceLookup
ConfluentEncodedSchemaReferenceReader	RedisConnectionPoolService
ConfluentEncodedSchemaReferenceWriter	RedisDistributedMapCacheClientService
ConfluentSchemaRegistry	RedshiftConnectionPool
CouchbaseClusterService	RestLookupService
CouchbaseKeyValueLookupService	ScriptedLookupService
CouchbaseMapCacheClient	ScriptedReader
CouchbaseRecordLookupService	ScriptedRecordSetWriter
CSVReader	ScriptedRecordSink
CSVRecordLookupService	SimpleDatabaseLookupService
CSVRecordSetWriter	SimpleKeyValueLookupService
DatabaseRecordLookupService	SimpleRedisDistributedMapCacheClientService
DatabaseRecordSink	SimpleScriptedLookupService
DatabaseTableSchemaRegistry	SiteToSiteReportingRecordSink
DBCPConnectionPool	SlackRecordSink
DBCPConnectionPoolLookup	SmbjClientProviderService
DistributedMapCacheClientService	SnowflakeComputingConnectionPool
DistributedMapCacheLookupService	StandardAsanaClientProviderService
DistributedMapCacheServer	StandardAzureCredentialsControllerService
DistributedSetCacheClientService	StandardDropboxCredentialService
DistributedSetCacheServer	StandardFileResourceService
EBCDICRecordReader [Technical Preview]	StandardHashiCorpVaultClientService
ElasticSearchClientServiceImpl	StandardHttpContextMap
ElasticSearchLookupService	StandardJsonSchemaRegistry [Technical Preview]
ElasticSearchStringLookupService	StandardOauth2AccessTokenProvider
EmailRecordSink	StandardPGPPrivateKeyService
EmbeddedHazelcastCacheManager	StandardPGPPublicKeyService
ExcelReader	StandardPLC4XConnectionPool [Technical Preview]
ExternalHazelcastCacheManager	StandardPrivateKeyService
FreeFormTextRecordSetWriter	StandardProxyConfigurationService
GCPCredentialsControllerService	StandardRestrictedSSLContextService
GCSFileResourceService	StandardS3EncryptionService

GenericPLC4XConnectionPool [Technical Preview]	StandardSnowflakeIngestManagerProviderService
GrokReader	StandardSSLContextService
HadoopCatalogService	StandardWebClientServiceProvider
HadoopDBCPConnectionPool	Syslog5424Reader
HazelcastMapCacheClient	SyslogReader
HBase_2_ClientMapCacheService	UDPEventRecordSink
HBase_2_ClientService	VolatileSchemaCache
HBase_2_RecordLookupService	WindowsEventLogReader
Hive3ConnectionPool	XMLReader
HiveCatalogService	XMLRecordSetWriter
ImpalaConnectionPool	YamlTreeReader
IPFIXReader	ZendeskRecordSink

Supported NiFi reporting tasks

This release is based on Apache NiFi versions 1.28 and 2.0 and includes a set of reporting tasks. Most reporting tasks included these NiFi versions are supported by Cloudera.

To ensure optimal performance and reliable support, it is crucial to use only supported reporting tasks and avoid deploying unsupported ones in production environments.

Additional reporting tasks are developed and tested by the community but are not officially supported by Cloudera. Reporting tasks may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

By adhering to the above guidelines, you can maintain stable and reliable workflows in your production environments.

NiFi 1.28

- AmbariReportingTask
- ControllerStatusReportingTask
- MetricsEventReportingTask
- MonitorDiskUsage
- MonitorMemory
- PrometheusReportingTask
- QueryNiFiReportingTask
- ReportLineageToAtlas
- ScriptedReportingTask
- SiteToSiteBulletinReportingTask
- SiteToSiteMetricsReportingTask
- SiteToSiteProvenanceReportingTask
- SiteToSiteStatusReportingTask

NiFi 2.0

- ControllerStatusReportingTask
- MonitorDiskUsage
- MonitorMemory
- PrometheusReportingTask
- QueryNiFiReportingTask

- ReportLineageToAtlas
- ScriptedReportingTask
- SiteToSiteBulletinReportingTask
- SiteToSiteMetricsReportingTask
- SiteToSiteProvenanceReportingTask
- SiteToSiteStatusReportingTask

Supported NiFi parameter providers

This release is based on Apache NiFi versions 1.28 and 2.0 and includes a set of parameter providers. Most parameter providers included these NiFi versions are supported by Cloudera.

To ensure optimal performance and reliable support, it is crucial to use only supported parameter providers and avoid deploying unsupported ones in production environments.

Additional parameter providers are developed and tested by the community but are not officially supported by Cloudera. Parameter providers may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

By adhering to the above guidelines, you can maintain stable and reliable workflows in your production environments.

NiFi 1.28

- AwsSecretsManagerParameterProvider
- AzureKeyVaultSecretsParameterProvider
- CyberArkConjurParameterProvider
- DatabaseParameterProvider
- EnvironmentVariableParameterProvider
- FileParameterProvider
- GcpSecretManagerParameterProvider
- HashiCorpVaultParameterProvider

NiFi 2.0

- AwsSecretsManagerParameterProvider
- AzureKeyVaultSecretsParameterProvider
- CyberArkConjurParameterProvider
- DatabaseParameterProvider
- EnvironmentVariableParameterProvider
- FileParameterProvider
- GcpSecretManagerParameterProvider
- HashiCorpVaultParameterProvider
- OnePasswordParameterProvider

Supported NiFi flow analysis rules [Technical Preview]

Apache NiFi 2.0 introduces flow analysis rules, a new feature designed to enhance flow validation and management by evaluating components or parts of a flow and may generate rule violations to help optimize or maintain flow design.

Important:

The Flow Analysis Rules feature is provided in Technical Preview. Work is still in progress and breaking changes may occur in the next releases. Do not use this feature in production!

Currently, only the DisallowComponentType flow analysis rule is available for use.

Supported NiFi Python components [Technical Preview]

Apache NiFi 2.0 introduces a set of NiFi components written in Python. Most of these Python components are supported by Cloudera.

Important:

The Python API feature is provided in Technical Preview. Work is still in progress and breaking changes may occur in the upcoming release. Do not use this feature in production!

Supported Python components:

Bedrock

Invokes different type of models with the given prompt via Bedrock.

ChunkData

Processes the output of Partition* processors, and creates chunks from the input document in a standardized format, based on the user defined settings. The processor processes only text content, any other data - like images - are silently ignored. The output is a JSON document.

ChunkDocument

Divides a large text document into smaller chunks. Input is expected in the form of a FlowFile containing a JSON Lines document, where each line includes a 'text' and a 'metadata' element.

EmbedData

Embeds incoming data using a locally present model. The processor either embeds the whole incoming data, or specific values of an incoming JSON input. Models can be downloaded for example from huggingface.co by cloning the model's repository.

InsertToMilvus

Inserts or updates a vector in a Milvus collection. The input data is expected to be a float vector in JSON format (the dimension of the input must match dimension of the collection). Usually used together with EmbedData processor providing the float vector as the input for InsertToMilvus.

LexicalQueryMilvus

Performs a lexical search on a Milvus collection. The processor can query Milvus either by a list of IDs or by a filter. The IDs can be specified in a comma separated list in a specified attribute or in the content of the FlowFile. If the IDs are extracted from the content, the FlowFile should be in JSON format having an array of Milvus element objects. The JSON format is either the format of the output of the VectorQueryMilvus processor (list of lists) or a simple JSON list of Milvus objects. Each Milvus object is expected to have at least the primary key field specified.

ParseDocument

Parses incoming unstructured text documents and performs optical character recognition (OCR) to extract text from PDF and image files. The output is formatted as 'json-lines' with two keys: 'text' and 'metadata'. The use of this processor may require significant storage space and RAM utilization due to third-party dependencies necessary for processing PDF and image files. Additionally, it is important to install Tesseract and Poppler on your system to enable the processing of PDFs or images.

PartitionCsv

Partitions a CSV file using the partition_csv function of unstructured.io. Properties are forwarded to partition_csv as parameters. The output is a JSON document in the format output by partition_csv.

PartitionDocx

Partitions DOCX data using the partition_docx function of unstructured.io. Properties are forwarded to partition_docx as parameters. The output is a JSON document in the format output by partition_docx.

PartitionHtml

Partitions HTML data using the partition_html function of unstructured.io. Properties are forwarded to partition_html as parameters. The output is a JSON document in the format output by partition_html.

PartitionPdf

Partitions a PDF file using the partition_pdf function of unstructured.io. Properties are forwarded to partition_pdf as parameters. The output is a JSON document in the format output by partition_pdf.

PartitionText

Partitions a text file using the partition_text function of unstructured.io. Properties are forwarded to partition_text as parameters. The output is a JSON document in the format output by partition_text.

PromptChatGPT

Submits a prompt to ChatGPT, writing the results either to a FlowFile attribute or to the contents of the FlowFile.

PutChroma

Publishes JSON data to a Chroma VectorDB. The incoming data must be in single JSON per Line format, containing two keys: 'text' and 'metadata'. The text must be a string, while metadata must be a map with string values. Any additional fields are ignored. If the collection name specified does not exist, the processor automatically creates the collection.

PutOpenSearchVector

Publishes JSON data to OpenSearch. The Incoming data must be in single JSON per Line format, each with two keys: 'text' and 'metadata'. The text must be a string, while metadata must be a map with strings for values. Any additional fields will be ignored.

PutPinecone

Creates vectors/embeddings that represent text content and sends the vectors to Pinecone. This use case assumes that the data has already been formatted in JSONL format with the text to be stored in Pinecone provided in the 'text' field.

PutQdrant

Publishes JSON data to Qdrant. The Incoming data must be in single JSON per Line format, each with two keys: 'text' and 'metadata'. The text must be a string, while metadata must be a map with strings for values. Any additional fields will be ignored.

QueryChroma

Queries a Chroma Vector Database to gather a specified number of documents that are most closely related to the given query.

QueryOpenSearchVector

Queries OpenSearch in order to gather a specified number of documents that are most closely related to the given query.

QueryPinecone

Queries Pinecone to gather a specified number of documents that are most closely related to the given query.

QueryQdrant

Queries Qdrant in order to gather a specified number of documents that are most closely related to the given query.

VectorQueryMilvus

Performs a vector search in a Milvus collection. The input data is expected to be a float vector in JSON format. (the dimension of the input must match dimension of the collection). Usually used together with EmbedData processor providing the float vector as the input for VectorQueryMilvus.

While additional Python components are developed and tested by the community, they are not officially supported by Cloudera. Python components may be excluded due to various reasons, such as insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices. Do not use these unsupported Python components in your production environments.

Cloudera exclusive components [Technical Preview]

Cloudera provides a set of NiFi components available only to its customers. These components provide additional functionality and are tailored to enhance the Cloudera NiFi experience. The list of these components is provided below.

In Flow Management clusters with NiFi 1.28

Processors

- CaptureChangeDebeziumDB2
- CaptureChangeDebeziumMySQL
- CaptureChangeDebeziumOracle
- CaptureChangeDebeziumPostgreSQL
- CaptureChangeDebeziumSQLServer
- ConsumeKafka2CDP
- ConsumeKafka2RecordCDP
- ConvertProtobuf
- DeleteCDPObjectStore
- FetchCDPObjectStore
- GetJiraIssue
- InvokeGRPC
- ListCDPObjectStore
- ListenGRPC
- ListenNetFlow
- PublishKafka2CDP
- PublishKafka2RecordCDP
- PutCDPObjectStore
- PutClouderaHiveQL
- PutClouderaHiveStreaming
- PutClouderaORC
- PutIcebergCDC
- PutJiraIssue
- SelectClouderaHiveQL
- TriggerClouderaHiveMetaStoreEvent
- UpdateClouderaHiveTable
- UpdateDeltaLakeTable

Controller Services

- ActiveMQJMSConnectionFactoryProvider

- ADLSIDBrokerCloudCredentialsProviderControllerService
- AWSIDBrokerCloudCredentialsProviderControllerService
- AzureBlobIDBrokerCloudCredentialsProviderControllerService
- AzureServiceBusJMSConnectionFactoryProvider
- CdpCredentialsProviderControllerService
- CdpOauth2AccessTokenProviderControllerService
- CiscoEmblemSyslogMessageReader
- ClouderaHiveConnectionPool
- ClouderaSchemaRegistry
- CMLLookupService
- EBCDICRecordReader
- ImpalaConnectionPool
- IPFIXReader
- JiraRecordSink
- PostgreSQLConnectionPool
- RabbitMQJMSConnectionFactoryProvider
- RedshiftConnectionPool

Parameter Providers

- CyberArkConjurParameterProvider

In Flow Management clusters with NiFi 2.0

Processors

- CaptureChangeDebeziumDB2
- CaptureChangeDebeziumMySQL
- CaptureChangeDebeziumOracle
- CaptureChangeDebeziumPostgreSQL
- CaptureChangeDebeziumSQLServer
- ConsumeKafka2CDP
- ConsumeKafka2RecordCDP
- ConvertProtobuf
- DeleteCDPObjectStore
- FetchCDPObjectStore
- GetJiraIssue
- InvokeGRPC
- ListCDPObjectStore
- ListenGRPC
- ListenNetFlow
- PublishKafka2CDP
- PublishKafka2RecordCDP
- PutCDPObjectStore
- PutClouderaHiveQL
- PutClouderaHiveStreaming
- PutClouderaORC
- PutIcebergCDC
- PutJiraIssue
- SelectClouderaHiveQL
- TriggerClouderaHiveMetaStoreEvent
- UpdateClouderaHiveTable

- UpdateDeltaLakeTable

Controller Services

- ActiveMQJMSConnectionFactoryProvider
- ADLSIDBrokerCloudCredentialsProviderControllerService
- AWSIDBrokerCloudCredentialsProviderControllerService
- AzureBlobIDBrokerCloudCredentialsProviderControllerService
- AzureServiceBusJMSConnectionFactoryProvider
- CdpCredentialsProviderControllerService
- CdpOauth2AccessTokenProviderControllerService
- CiscoEmblemSyslogMessageReader
- ClouderaHiveConnectionPool
- ClouderaSchemaRegistry
- CMLLookupService
- EBCDICRecordReader
- IPFIXReader
- JiraRecordSink
- PostgreSQLConnectionPool
- RabbitMQJMSConnectionFactoryProvider
- RedshiftConnectionPool

Parameter Providers

- CyberArkConjurParameterProvider

Components supported by partners

This release is based on Apache NiFi versions 1.28 and 2.0 and includes a set of components built, maintained and supported by Cloudera partners. You should reach out directly to these partners in case you need assistance.

Although Cloudera's Quality Engineering teams have added test coverage for these components, they are not officially supported by Cloudera.

NiFi 1.28 processors supported by partners

- ConsumePulsar (1.18.0)
- ConsumePulsarRecord (1.18.0)
- PublishPulsar (1.18.0)
- PublishPulsarRecord (1.18.0)

NiFi 1.28 controller services supported by partners

- PulsarClientAthenzAuthenticationService (1.18.0)
- PulsarClientJwtAuthenticationService (1.18.0)
- PulsarClientOauthAuthenticationService (1.18.0)
- PulsarClientTlsAuthenticationService (1.18.0)
- StandardPulsarClientService (1.18.0)

These components can be used to push data into Apache Pulsar as well as getting data out of it. In case you have issues or questions while using these components, Cloudera recommends you to reach out to your StreamNative representative team.

Note: In Flow Management Data Hub clusters using NiFi 2, the Pulsar components are not included. This is temporary and the components will be added in an upcoming release. Meanwhile, you can manually download the components from a Maven repository and add them into your cluster.

Unsupported Features in Cloudera DataFlow for Data Hub 7.3.1

Some features exist within Cloudera DataFlow for Data Hub 7.3.1 components, but are not supported by Cloudera.

Unsupported Flow Management features

Some Flow Management features exist in Cloudera DataFlow for Data Hub 7.3.1, but are not supported by Cloudera.

NiFi 1.28

There are no unsupported features in this release.

NiFi 2.0

The following technical preview features are available in Flow Management in Cloudera DataFlow for Data Hub 7.3.1 but are not ready for production deployment. Cloudera encourages you to explore these technical preview features in non-production environments and provide feedback on your experiences through the Cloudera Community Forums.

- The Flow Analysis Rules engine is in Technical Preview and all the provided rules are also in Technical Preview.
- The Python API feature is in Technical Preview and all the Python extensions using this feature are also in Technical Preview.
- The ability to run a Process Group using the Stateless Engine is in Technical Preview.

NiFi Registry

There are no unsupported features in this release.

Related Information

Cloudera Community Forum

Unsupported Edge Management features [Technical Preview]

See the unsupported features listed in the Cloudera Edge Management documentation.

Unsupported Streams Messaging features

Some Streams Messaging features exist in Cloudera DataFlow for Data Hub 7.3.1, but are not supported by Cloudera.

Kafka

The following Kafka features are not ready for production deployment. Cloudera encourages you to explore these features in non-production environments and provide feedback on your experiences through the *Cloudera Community Forums*.

- Only Java and .Net based clients are supported. Clients developed with C, C++, Python, and other languages are currently not supported.
- The Kafka default authorizer is not supported. This includes setting ACLs and all related APIs, broker functionality, and command-line tools.
- SASL/SCRAM is only supported for delegation token based authentication. It is not supported as a standalone authentication mechanism.

- Kafka KRaft in this release of Cloudera Runtime is in technical preview and does not support the following:
  - Deployments with multiple log directories. This includes deployments that use JBOD for storage.
  - Delegation token based authentication.
  - Migrating an already running Kafka service from ZooKeeper to KRaft.
  - Atlas Integration.

Schema Registry

There are no updates for this release.

Streams Messaging Manager

There are no updates for this release.

Streams Replication Manager

There are no updates for this release.

Cruise Control

There are no updates for this release.

Related Information

Cloudera Community Forum
Setting up your Streams Messaging cluster

Unsupported Cloudera Streaming Analytics features

Some Cloudera Streaming Analytics features exist in Cloudera DataFlow for Cloudera Data Hub 7.3.1, but are not supported by Cloudera.

The following features are not ready for production deployment. Cloudera encourages you to explore these features in non-production environments and provide feedback on your experiences through the *Cloudera Community Forums*.

Cloudera SQL Stream Builder

- Virtual environments for Python are not supported.

Flink

- Apache Flink batch (DataSet) API
- GPU Resource Plugin
- SQL Client
- RAZ-enabled GCP environment
- The following features are not supported in SQL and Table API:
  - HBase Table Connector
  - Old Planner
  - Non-windowed (unbounded) joins, distinct

Related Information

Cloudera Community Forum

Known Issues In Cloudera DataFlow for Data Hub 7.3.1

You must be aware of the known issues and limitations, the areas of impact, and workaround in Cloudera DataFlow for Data Hub 7.3.1.

Known issues in Flow Management

Learn about the known issues and limitations in Flow Management clusters, their impact on functionality, and any available workarounds to mitigate these issues.

NiFi 1.28 with Cloudera Flow Management 2.2.9

CFM-4331: HBase 1.1.2 components incompatible with JDK17

HBase 1.1.2 components are not compatible with JDK 17.

To ensure full functionality and compatibility:

1. Upgrade HBase 1.1.2 components to their corresponding versions in HBase 2.
2. Upgrade your HBase servers.

   If upgrading the servers is not feasible, the HBase 2 client can still interact with HBase 1 servers, but compatibility is limited. While basic functionalities work, new features introduced in the HBase 2 client are supported when interacting with an HBase 1 server.

Unused NiFi configuration values

The following NiFi configuration values are no longer in use. They are still visible in the UI, but they are obsolete and have no effect on functionality.

- nifi.nar.hotfix.provider.file.list.identifier
- nifi.nar.hotfix.provider.location.identifier
- nifi.nar.hotfix.provider.last.modification.identifier
- nifi.nar.hotfix.provider.directory.identifier
- nifi.nar.hotfix.provider.date.time.format
- nifi.nar.hotfix.provider.proxy.user
- nifi.nar.hotfix.provider.proxy.password
- nifi.nar.hotfix.provider.proxy.server
- nifi.nar.hotfix.provider.proxy.server.port
- nifi.nar.hotfix.provider.connect.timeout
- nifi.nar.hotfix.provider.read.timeout
- nifi.nar.hotfix.provider.nar.location
- nifi.nar.hotfix.provider.poll.interval
- nifi.nar.hotfix.provider.implementation
- nifi.nar.hotfix.provider.user.name
- nifi.nar.hotfix.provider.password
- nifi.nar.hotfix.provider.base.url
- nifi.nar.hotfix.provider.required.version
- nifi.nar.hotfix.provider.enabled

Unable to view user interface after upgrade due to change in NiFi group authorization

After upgrading to Cloudera Public Cloud 7.3.1 (or 7.2.18), you may encounter the error message "Unable to view the user interface." This issue occurs because, in versions prior to 7.2.18, NiFi group authorization relied on the host's SSSD configuration for group synchronization.

With the deprecation of the SHELL user group provider, Cloudera Public Cloud 7.3.1 (and 7.2.18) defaults to the LDAP user group provider in Flow Management Data Hub clusters to handle user group management. This change offers enhanced compatibility, security, and performance.

Important: Only newly deployed Cloudera Public Cloud 7.3.1 (or 7.2.18) Flow Management Data Hub clusters use the LDAP user group provider. Upgraded clusters from previous versions still use the SHELL user group provider, leading to potential authorization issues.

To resolve this issue in upgraded clusters, follow the steps below to manually configure your Flow Management Data Hub cluster to use the LDAP user group provider:

1. Identify the management node of the Flow Management cluster and copy the Fully Qualified Domain Name (FQDN).
2. SSH into the management node.
3. Copy the script provided below and save it to a file.
4. Set executable permissions on the script file: chmod 755 script_name.sh
5. Run the script using the following command: ./script_name.sh *FQDN_OF_MANAGEMENT_NODE*
6. Enter your Cloudera credentials when prompted for a username and password.

After completing these steps, NiFi will be configured to use the LDAP user group provider, resolving the "Unable to view the user interface" issue.

```
#!/bin/bash
              clear
              #init incoming variables
              GREEN="\033[1;32m"
              ORANGE="\033[38;2;255;165;0m"
              RESET="\033[0m"
              RED="\033[1;31m"
              CM_HOST=$1

              #Get my auth
              echo -ne "${ORANGE}User Name: " # Need to bracket
this var to avoid a space in front
              read -s USERNAME
              echo -ne "\nEnter Password: $RESET"
              read -s PASSWORD
              echo #to prevent weird need to hit enter twice
              AUTH="$USERNAME:$PASSWORD"

              # Get My CM_API and if this fails it could be bad
password or host so I will ERROR
              CM_API=$(curl -s -k -u "$AUTH" https://$CM_HOST:7
183/api/version)
              if [[ ${#CM_API} -gt 4 ]]; then # This means prob
ably bad user or password
              echo -ne "$RED Error! Most likely bad credentials
 below is response\n\n$CM_API $RESET"
              exit 1
              fi

              CM_HOST_API_URL="https://$CM_HOST:7183/api/$CM_API"
              CM_CLUSTER_NAME=$(curl -s -k -u "$AUTH" -X GET "
$CM_HOST_API_URL/clusters?clusterType=any&view=SUMMARY" |\
              jq -r '.items[].name')
              mapfile -t CM_ROLES < <(curl --header "Content-Typ
e: application/json" --silent --insecure  --request GET \
              "$CM_HOST_API_URL/clusters/$CM_CLUSTER_NAME/serv
ices/nifi-NIFI-BASE/roleConfigGroups" \
```

```
               -u $AUTH | jq -r '.items[].name' | grep -v "nifi-N
IFI-BASE-GATEWAY-BASE")

               #extract password and ldap info from cm.settings
               LDAP_URL=$(awk '/setsettings LDAP_URL/ {print $N
F}' /etc/cloudera-scm-server/cm.settings)
               LDAP_BIND_DN=$(awk '/setsettings LDAP_BIND_DN/ {
print $NF}' /etc/cloudera-scm-server/cm.settings)
               LDAP_BIND_PW=$(awk '/setsettings LDAP_BIND_PW/ {
print $NF}' /etc/cloudera-scm-server/cm.settings)
               LDAP_USER_SEARCH_BASE=$(awk '/setsettings LDAP_U
SER_SEARCH_BASE/ {print $NF}' /etc/cloudera-scm-server/cm.settin
gs)
               LDAP_GROUP_SEARCH_BASE=$(awk '/setsettings LDAP_GR
OUP_SEARCH_BASE/ {print $NF}' /etc/cloudera-scm-server/cm.settin
gs)

               echo -ne "$GREEN Building CM API payload to move aw
ay from shell-user-group and into ldap-user-group provider\n $RE
SET"

               cat > .cloudera-payload.json <<- EOF
               {"items":[
               {"name":"nifi.ldap.url","value":"$LDAP_URL"},
               {"name":"nifi.ldap.manager.dn","value":"$LDAP_BIND
_DN"},
               {"name":"nifi.ldap.manager.password","value":"$LDAP
_BIND_PW"},
               {"name":"nifi.ldap.user.search.base","value":"$LDA
P_USER_SEARCH_BASE"},
               {"name":"xml.authorizers.userGroupProvider.ldap-
user-group-provider.property.Group Search Base","value":"$LDAP_G
ROUP_SEARCH_BASE"},
               {"name":"nifi.ldap.enabled","value":"true"},
               {"name":"xml.authorizers.userGroupProvider.shell-us
er-group-provider.enabled","value":"false"},
               {"name":"nifi.ldap.authentication.strategy","valu
e":"LDAPS"},
               {"name":"nifi.ldap.tls.protocol","value":"TLS"},
               {"name":"nifi.ldap.tls.keystore.type","value":"jks
"},
               {"name":"nifi.ldap.tls.truststore.type","value":"j
ks"},
               {"name":"nifi.ldap.tls.keystore","value":"\${nif
i.security.keystore}"},
               {"name":"nifi.ldap.tls.truststore","value":"\${nif
i.security.truststore}"},
               {"name":"xml.authorizers.userGroupProvider.ldap-
user-group-provider.property.Group Object Class","value":"top"},
               {"name":"xml.authorizers.userGroupProvider.ldap-u
ser-group-provider.property.User Group Name Attribute","value":"
memberOf"},
               {"name":"xml.authorizers.userGroupProvider.ldap-us
er-group-provider.property.User Identity Attribute","value":"uid
"},
               {"name":"xml.authorizers.userGroupProvider.ldap-us
er-group-provider.property.Group Name Attribute","value":"cn"},
               {"name":"xml.authorizers.userGroupProvider.composi
te-user-group-provider.property.User Group Provider 2","value":"
ldap-user-group-provider"},
               {"name":"staging/login-identity-providers.xml_role
_safety_valve","value":"<property><name>xml.loginIdentityProvide
rs.provider.ldap-provider.property.TLS - Keystore Password</name
><value>\${nifi.security.keystorePasswd}</value></property><prop
```

```
erty><name>xml.loginIdentityProviders.provider.ldap-provider.pro
perty.TLS - Truststore Password</name><value>\${nifi.security.tr
uststorePasswd}</value></property>"},
              {"name":"staging/authorizers.xml_role_safety_val
ve","value":"<property><name>xml.authorizers.userGroupProvider.l
dap-user-group-provider.property.TLS - Keystore Password</name><
value>\${nifi.security.keystorePasswd}</value></property><proper
ty><name>xml.authorizers.userGroupProvider.ldap-user-group-provi
der.property.TLS - Truststore Password</name><value>\${nifi.secu
rity.truststorePasswd}</value></property>"}
              ]}
              EOF

              echo -ne "\n$GREEN Call CM api to change nifi co
nfig\n\n $RESET"
              for role in "${CM_ROLES[@]}"; do
              echo -ne "$ORANGE Updating role $role *** $RESET\n"
              curl -s --header "Content-Type: application/json"
 --insecure  --request PUT --data @.cloudera-payload.json \
              -u $AUTH "$CM_HOST_API_URL/clusters/$CM_CLUSTER_N
AME/services/nifi-NIFI-BASE/roleConfigGroups/$role/config" > /de
v/null
              done
              echo -ne "$GREEN\nRestarting NiFi $RESET\n"
              curl -s --header "Content-Type: application/json"
--insecure  --request POST \
              -u $AUTH "$CM_HOST_API_URL/clusters/$CM_CLUSTER_N
AME/services/nifi-NIFI-BASE/commands/restart" > /dev/null
              rm -f .cloudera-payload.json
```

PutIcebergCDC processor error: Unable to specify server’s Kerberos Principal name

When using the PutIcebergCDC processor, you may encounter an error if the Hadoop Configuration Resources property specified for the Catalog Service only includes the standard Hadoop configuration files from Cloudera environment (/etc/hadoop/conf/core-site.xml, /etc/hadoop/conf/ssl-client.xml, and /etc/hive/conf/hive-site.xml). The error message states:

```
Failed to specify server’s Kerberos principal name.
```

To resolve this issue, simply add the hdfs-site.xml file to the Hadoop Configuration Resources of the PutIcebergCDC processor’s Catalog Service.

Incomplete Ranger policy for NiFi metrics in Cloudera Manager

Cloudera Manager does not accurately reflect NiFi metrics for the NiFi service due to incomplete Flow NiFi access policies in Ranger. The required 'nifi' group is not included in the access policies, resulting in restricted access to the metrics data.

To ensure that Cloudera Manager accurately reflects the NiFi metrics for the NiFi service:

1. Log in to Ranger and navigate to the Flow NiFi access policies.
2. Add the 'nifi' group to the relevant access policies to ensure that Cloudera Manager can access the metrics data.
3. Confirm and save the updated policies.

InferAvroSchema may fail when inferring schema for JSON data

In Apache NiFi 1.17, the dependency on Apache Avro has been upgraded to 1.11.0. However, the InferAvroSchema processor depends on the hadoop-libraries NAR from which the Avro version comes from, causing a NoSuchMethodError exception.

Important: This processor is not supported by Cloudera and its use is highly discouraged as inferring a schema from the data is not recommended in production data flows.

Having well defined schemas ensures consistent behavior, allows for proper schema versioning, and prevents downstream systems from generating errors because of unexpected schema changes. Besides, schema inference may not always be 100% accurate and can be an expensive operation in terms of performances.

Use the ExtractRecordSchema processor with the proper Reader to infer the Avro schema for your data.

NiFi 2.0 with Cloudera Flow Management 4.2.1

Important: The current version of Apache NiFi 2 included in Flow Management clusters is not suitable for production when using the newly released features. Specifically, there are known issues regarding resource consumption when using the Python-based extensions.

Processors using OpenAI library may not work

When using Flow Management clusters, several processors relying on the OpenAI library are not functional due to compatibility issues caused by OpenAI API changes. The affected processors use an outdated OpenAI library version that is no longer supported. The impacted processors are:

- PutChroma
- QueryChroma
- PromptChatGPT
- PutOpenSearchVector
- QueryOpenSearchVector
- PutPinecone
- QueryPinecone
- PutQdrant
- QueryQdrant

These processors require an updated OpenAI library version (1.56.2 or later) to function correctly.

To restore functionality for the impacted processors, follow these steps:

1. Update the OpenAI library version in the associated Python code to version 1.56.2.
   a. Locate the processor's py file.

      For example: /opt/cloudera/parcels/CFM-4.0.0.0-382/NIFI-2/python/extensions/openai/PromptChatGPT.py
   b. Find "openai==1.9.0" and replace it with "openai==1.56.2".
2. Navigate to the NiFi work directory and delete the folder for the affected processors.

   The directory path for an affected processor typically follows this structure: /var/lib/nifi/python_artifacts/extensions/<ProcessorName>/<Version>

   For example, for the PromptChatGPT processor, the path would be: /var/lib/nifi/python_artifacts/extensions/*PROMPTCHATGPT/2.0.0.4.0.0.0-382*

   So in this case, delete the entire PromptChatGPT folder, including its version folder.
3. Restart the NiFi service to apply the changes.

Invalid Python version

Due to the invalid Python version defined for the NiFi service, the Python API based processors (such as PromptChatGPT, QueryPinecone, and so on) will remain invalid as the NiFi service will be unable to download the associated dependencies. The issue can be resolved by changing the version for the nifi.python.command property.

1. Go to your cluster in Cloudera Manager.
2. Select NiFi from the list of services.
3. Select Configuration.
4. Review the value defined for nifi.python.command property.
5. Change the value to python3.11 if the current value is python3.9.
6. Click Save changes.
7. Stop the NiFi service.
8. Delete the /hadoopfs/fs4/working-dir/python_artifacts directory from all NiFi nodes.
9. Restart the NiFi service.

PutIcebergCDC processor error: Unable to specify server's Kerberos Principal name

When using the PutIcebergCDC processor, you may encounter an error if the Hadoop Configuration Resources property specified for the Catalog Service only includes the standard Hadoop configuration files from Cloudera environment (/etc/hadoop/conf/core-site.xml, /etc/hadoop/conf/ssl-client.xml, and /etc/hive/conf/hive-site.xml). The error message states: Failed to specify server's Kerberos principal name.

To resolve this issue, simply add the hdfs-site.xml file to the Hadoop Configuration Resources of the PutIcebergCDC processor's Catalog Service.

Known issues in Edge Management [Technical Preview]

Learn about the known issues in Edge Management clusters, the impact or changes to the functionality, and any available workaround.

For Edge Management known issues, see the Cloudera Edge Management documentation.

Known Issues in Streams Messaging

Learn about the known issues in Streams Messaging clusters, the impact or changes to the functionality, and the workaround.

Kafka

Learn about the known issues and limitations in Kafka in this release:

Known Issues

OPSAPS-59553: Streams Messaging Manager's bootstrap server config should be updated based on Kafka's listeners

Streams Messaging Manager does not show any metrics for Kafka or Kafka Connect when multiple listeners are set in Kafka.

Workaround: Streams Messaging Manager cannot identify multiple listeners and still points to bootstrap server using the default broker port (9093 for SASL_SSL). You need to override the bootstrap server URL by performing the following steps:

1. In Cloudera Manager, go to Streams Messaging Manager Configuration Streams Messaging Manager Rest Admin Server Advanced Configuration Snippet (Safety Valve) for streams-messaging-manager.yaml
2. Override bootstrap server URL (hostname:port as set in the listeners for broker) for streams-messaging-manager.yaml.
3. Save your changes.
4. Restart Streams Messaging Manager.

The offsets.topic.replication.factor property must be less than or equal to the number of live brokers

The offsets.topic.replication.factor broker configuration is now enforced upon auto topic creation. Internal auto topic creation will fail with a GROUP_COORDINATOR_NOT_AVAILABLE error until the cluster size meets this replication factor requirement.

None

Requests fail when sending to a nonexistent topic with auto.create.topics.enable set to true

The first few produce requests fail when sending to a nonexistent topic with auto.create.topics.e nable set to true.

Increase the number of retries in the producer configuration setting retries.

KAFKA-2561: Performance degradation when SSL Is enabled

In some configuration scenarios, significant performance degradation can occur when SSL is enabled. The impact varies depending on your CPU, JVM version, Kafka configuration, and message size. Consumers are typically more affected than producers.

Configure brokers and clients with ssl.secure.random.implementation = SHA1PRNG. It often reduces this degradation drastically, but its effect is CPU and JVM dependent.

CDPD-45183: Kafka Connect active topics might be visible to unauthorised users

The Kafka Connect active topics endpoint (/connectors/*[***CONNECTOR NAME***]*/topics) and the Connect Cluster page on the Streams Messaging Manager UI disregard the user permissions configured for the Kafka service in Ranger. As a result, all active topics of connectors might become visible to users who do not have permissions to view them. Note that user permission configured for Kafka Connect in Ranger are not affected by this issue and are correctly applied.

None.

RANGER-3809: Idempotent Kafka producer fails to initialize due to an authorization failure

Kafka producers that have idempotence enabled require the Idempotent Write permission to be set on the cluster resource in Ranger. If permission is not given, the client fails to initialize and an error similar to the following is thrown:

```
org.apache.kafka.common.KafkaException: Cannot execute transacti
onal method because we are in an error state
              at org.apache.kafka.clients.producer.internals.Tr
ansactionManager.maybeFailWithError(TransactionManager.java:1125)
              at org.apache.kafka.clients.producer.internals.T
ransactionManager.maybeAddPartition(TransactionManager.java:442)
              at org.apache.kafka.clients.producer.KafkaProduce
r.doSend(KafkaProducer.java:1000)
              at org.apache.kafka.clients.producer.KafkaProduc
er.send(KafkaProducer.java:914)
              at org.apache.kafka.clients.producer.KafkaProducer
.send(KafkaProducer.java:800)
              .
              .
              .
              Caused by: org.apache.kafka.common.errors.Cluste
rAuthorizationException: Cluster authorization failed.
```

Idempotence is enabled by default for clients in Kafka 3.0.1, 3.1.1, and any version after 3.1.1. This means that any client updated to 3.0.1, 3.1.1, or any version after 3.1.1 is affected by this issue.

This issue has two workarounds, do either of the following:

- Explicitly disable idempotence for the producers. This can be done by setting enable.idempoten ce to false.
- Update your policies in Ranger and ensure that producers have Idempotent Write permission on the cluster resource.

CDPD-49304: AvroConverter does not support composite default values

AvroConverter cannot handle schemas containing a STRUCT type default value.

None.

DBZ-4990: The Debezium Db2 Source connector does not support schema evolution

The Debezium Db2 Source connector does not support the evolution (updates) of schemas. In addition, schema change events are not emitted to the schema change topic if there is a change in the schema of a table that is in capture mode. For more information, see DBZ-4990.

None.

CFM-3532: The Stateless NiFi Source, Stateless NiFi Sink, and HDFS Stateless Sink connectors cannot use Snappy compression

This issue only affects Stateless NiFi Source and Sink connectors if the connector is running a dataflow that uses a processor that uses Hadoop libraries and is configured to use Snappy compression. The HDFS Stateless Sink connector is only affected if the Compression Codec or Compression Codec for Parquet properties are set to SNAPPY.

If you are affected by this issue, errors similar to the following will be present in the logs.

```
Failed to write to HDFS due to java.lang.UnsatisfiedLinkError: o
rg.apache.hadoop.util.NativeCodeLoader.buildSupportsSnappy()
```

```
Failed to write to HDFS due to java.lang.RuntimeException: nativ
e snappy library not available: this version of libhadoop was bu
ilt without snappy support.
```

Download and deploy missing libraries.

Important: Ensure that you complete steps 1-11 on all Kafka Connect hosts. Additionally, ensure that the advanced configuration snippet in step 12 is configured for all Kafka Connect role instances.

1. Create the /opt/nativelibs directory.

   ```
   mkdir /opt/nativelibs
   ```

2. Change the owner to kafka.

   ```
   chown kafka:kafka /opt/nativelibs
   ```

3. Locate the directory containing the Hadoop native libraries and copy its contents to the directory you created.

   ```
   cp /opt/cloudera/parcels/CDH/lib/hadoop/lib/native/* /opt/na
   tivelibs
   ```

4. Verify that libsnappy.so was copied to the directory you created.
5. Remove the following from /opt/nativelibs.

   ```
   libhadoop.a
                     libhadoop.so
                     libhadoop.so.1.0.0
   ```

6. Run the following command.

   ```
   hadoop version
   ```

   The command returns the Hadoop version running in the cluster. Note down the first three digits in the version.

7. Go to https://archive.apache.org/dist/hadoop/common/ and download the Hadoop version that matches the first three digits of the version running in the cluster.

   For example, if your Hadoop version is 3.1.1.7.1.9.0-296, then you need to download Hadoop 3.1.1.
8. Extract the downloaded archive.
9. Copy the following libraries from the downloaded archive to /opt/nativelibs on the cluster host.

```
libhadoop.a
                    libhadoop.so.1.0.0
```

   The libraries are located in hadoop-*[***VERSION***]*/lib/native.
10. Create a symlink named libhadoop.so and point it to /opt/nativelibs/libhadoop.so.1.0.0.

```
ln -s /opt/nativelibs/libhadoop.so.1.0.0 /opt/nativelibs/lib
hadoop.so
```

11. Change the owner of every entry within /opt/nativelibs to kafka.

```
chown -h kafka:kafka /opt/nativelibs/*
```

12. In Cloudera Manager, go to Kafka service Configuration .
13. Add the following key-value pair to Kafka Connect Environment Advanced Configuration Snippet (Safety Valve).
    - Key: LD_LIBRARY_PATH
    - Value: /opt/nativelibs
14. Click Save Changes.
15. Restart the Kafka service.

OPSAPS-69317: Kafka Connect Rolling Restart Check fails if SSL Client authentication is required

The rolling restart action does not work in Kafka Connect when the ssl.client.auth option is set to required. The health check fails with a timeout which blocks restarting the subsequent Kafka Connect instances.

You can set ssl.client.auth to requested instead of required and initiate a rolling restart again. Alternatively, you can perform the rolling restart manually by restarting the Kafka Connect instances one-by-one and checking periodically whether the service endpoint is available before starting the next one.

Unsupported features

The following Kafka features are not supported in Cloudera:

- Only Java and .Net based clients are supported. Clients developed with C, C++, Python, and other languages are currently not supported.
- The Kafka default authorizer is not supported. This includes setting ACLs and all related APIs, broker functionality, and command-line tools.
- SASL/SCRAM is only supported for delegation token based authentication. It is not supported as a standalone authentication mechanism.
- Kafka KRaft in this release of Cloudera Runtime is in technical preview and does not support the following:
  - Deployments with multiple log directories. This includes deployments that use JBOD for storage.
  - Delegation token based authentication.
  - Migrating an already running Kafka service from ZooKeeper to KRaft.
  - Atlas Integration.

Limitations

Collection of Partition Level Metrics May Cause Cloudera Manager Performance to Degrade

If the Kafka service operates with a large number of partitions, collection of partition level metrics may cause Cloudera Manager performance to degrade.

If you are observing performance degradation and your cluster is operating with a high number of partitions, you can choose to disable the collection of partition level metrics.

Important: If you are using Streams Messaging Manager to monitor Kafka or Cruise Control for rebalancing Kafka partitions, be aware that both Streams Messaging Manager and Cruise Control rely on partition level metrics. If partition level metric collection is disabled, Streams Messaging Manager will not be able to display information about partitions. In addition, Cruise Control will not operate properly.

Complete the following steps to turn off the collection of partition level metrics:

1. Obtain the Kafka service name:
   a. In Cloudera Manager, Select the Kafka service.
   b. Select any available chart, and select Open in Chart Builder from the configuration icon drop-down.
   c. Find $SERVICENAME= near the top of the display.

      The Kafka service name is the value of $SERVICENAME.
2. Turn off the collection of partition level metrics:
   a. Go to Hosts Hosts Configuration .
   b. Find and configure the Cloudera Manager Agent Monitoring Advanced Configuration Snippet (Safety Valve) configuration property.

      Enter the following to turn off the collection of partition level metrics:

      ```
      [KAFKA_SERVICE_NAME]_feature_send_broker_topic_partition_ent
      ity_update_enabled=false
      ```

      Replace [KAFKA_SERVICE_NAME] with the service name of Kafka obtained in step 1. The service name should always be in lower case.
   c. Click Save Changes.

Schema Registry

Learn about the known issues and limitations in Schema Registry in this release:

CDPD-40380: Authorization checking issue when Kerberos is disabled

Due to an issue in Ranger, when Kerberos is disabled then it is not possible to check authorization.

1. Open Schema Registry configuration in Cloudera Manager.
2. Find the ranger.plugin.schema-registry.service.name field.
3. Replace GENERATED_RANGER_SERVICE_NAME with the actual name of the service.
4. Restart the Schema Registry service.

CDPD-49304: AvroConverter does not support composite default values

AvroConverter cannot handle schemas containing a STRUCT type default value.

None.

OPSAPS-70971: Schema Registry does not have permissions to use Atlas after an upgrade

Following an upgrade, Schema Registry might not have the required permissions in Ranger to access Atlas. As a result, Schema Registry's integration with Atlas might not function in secure clusters where Ranger authorization is enabled.

1. Access the Ranger Console (Ranger Admin web UI).

2. Click the cm_atlas resource-based service.
3. Add the schemaregistry user to the all - * policies.
4. Click Manage Service Edit Service .
5. Add the schemaregistry user to the default.policy.users property.

OPSAPS-69317: Kafka Connect Rolling Restart Check fails if SSL Client authentication is required

The rolling restart action does not work in Kafka Connect when the ssl.client.auth option is set to required. The health check fails with a timeout which blocks restarting the subsequent Kafka Connect instances.

You can set ssl.client.auth to requested instead of required and initiate a rolling restart again. Alternatively, you can perform the rolling restart manually by restarting the Kafka Connect instances one-by-one and checking periodically whether the service endpoint is available before starting the next one.

Streams Messaging Manager

Learn about the known issues in Streams Messaging Manager in this release.

OPSAPS-59597: Streams Messaging Manager UI logs are not supported by Cloudera Manager

Cloudera Manager does not display a Log Files menu for Streams Messaging Manager UI role (and Streams Messaging Manager UI logs cannot be displayed in the Cloudera Manager UI) because the logging type used by Streams Messaging Manager UI is not supported by Cloudera Manager.

View the Streams Messaging Manager UI logs on the host.

CDPD-39313: Some numbers are not rendered properly in Streams Messaging Manager UI

Very large numbers can be imprecisely represented on the UI. For example, bytes larger than 8 petabytes would lose precision.

None.

OPSAPS-59553: Streams Messaging Manager's bootstrap server config should be updated based on Kafka's listeners

Streams Messaging Manager does not show any metrics for Kafka or Kafka Connect when multiple listeners are set in Kafka.

Streams Messaging Manager cannot identify multiple listeners and still points to bootstrap server using the default broker port (9093 for SASL_SSL). You need to override bootstrap server URL (hostname:port as set in the listeners for broker). Add the bootstrap server details in Streams Messaging Manager safety valve in the following path:

1. In Cloudera Manager, go to Streams Messaging Manager Configuration Streams Messaging Manager Rest Admin Server Advanced Configuration Snippet (Safety Valve) for streams-messaging-manager.yaml
2. Add the following value for bootstrap servers.

```
streams.messaging.manager.kafka.bootstrap.servers=<comma-sep
arated list of brokers>
```

3. Save your changes.
4. Restart Streams Messaging Manager.

CDPD-45183: Kafka Connect active topics might be visible to unauthorised users

The Kafka Connect active topics endpoint (/connectors/*[***CONNECTOR NAME***]*/topics) and the Connect Cluster page on the Streams Messaging Manager UI disregard the user permissions configured for the Kafka service in Ranger. As a result, all active topics of connectors might become visible to users who do not have permissions to view them. Note that user permission configured for Kafka Connect in Ranger are not affected by this issue and are correctly applied.

None.

Limitations

CDPD-36422: 1MB flow.snapshot freezes Safari

While importing large connector configurations, flow.snapshots reduces the usability of the Streams Messaging Manager when using Safari browser.

Use a different browser (Chrome/Firefox/Edge).

Streams Replication Manager

Learn about the known issues and limitations in Streams Replication Manager in this release:

Known Issues

CDPD-22089: Streams Replication Manager does not sync re-created source topics until the offsets have caught up with target topic

Messages written to topics that were deleted and re-created are not replicated until the source topic reaches the same offset as the target topic. For example, if at the time of deletion and re-creation there are a 100 messages on the source and target clusters, new messages will only get replicated once the re-created source topic has 100 messages. This leads to messages being lost.

None

CDPD-11079: Blacklisted topics appear in the list of replicated topics

If a topic was originally replicated but was later disallowed (blacklisted), it will still appear as a replicated topic under the /remote-topics REST API endpoint. As a result, if a call is made to this endpoint, the disallowed topic will be included in the response. Additionally, the disallowed topic will also be visible in the Streams Messaging Manager UI. However, it's Partitions and Consumer Groups will be 0, its Throughput, Replication Latency and Checkpoint Latency will show N/A.

None

CDPD-30275: Streams Replication Manager may automatically re-create deleted topics on target clusters

If auto.create.topics.enable is enabled, deleted topics might get automatically re-created on target clusters. This is a timing issue. It only occurs if remote topics are deleted while the replication of the topic is still ongoing.

1. Remove the topic from the topic allowlist with srm-control. For example:

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGE
T_CLUSTER] --remove [TOPIC1]
```

2. Wait until Streams Replication Manager is no longer replicating the topic.
3. Delete the remote topic in the target cluster.

Limitations

Streams Replication Manager cannot replicate Ranger authorization policies to or from Kafka clusters

Due to a limitation in the Kafka-Ranger plugin, Streams Replication Manager cannot replicate Ranger policies to or from clusters that are configured to use Ranger for authorization. If you are using Streams Replication Manager to replicate data to or from a cluster that uses Ranger, disable authorization policy synchronization in Streams Replication Manager. This can be achieved by clearing the Sync Topic Acls Enabled (sync.topic.acls.enabled) checkbox.

Streams Replication Manager cannot ensure the exactly-once semantics of transactional source topics

Streams Replication Manager data replication uses at-least-once guarantees, and as a result cannot ensure the exactly-once semantics (EOS) of transactional topics in the backup/target cluster.

Note: Even though EOS is not guaranteed, you can still replicate the data of a transactional source, but you must set isolation.level to read_committed for Streams Replication Manager's internal consumers. This can be done by adding *[***SOURCE CLUSTER ALIAS***]->[***TARGET CLUSTER ALIAS***]*.consumer.isolation.level=read_committed to the Streams Replication Manager's Replication Configs Streams Replication Manager service property in Cloudera Manager. The isolation.level property can be set on a global connector or replication level. For example:

```
#Global connector level
                  connectors.consumer.isolation.level=read_committed
                  #Replication level
                  uswest->useast.consumer.isolation.level=read_committed
```

Streams Replication Manager checkpointing is not supported for transactional source topics

Streams Replication Manager does not correctly translate checkpoints (committed consumer group offsets) for transactional topics. Checkpointing assumes that the offset mapping function is always increasing, but with transactional source topics this is violated. Transactional topics have control messages in them, which take up an offset in the log, but they are never returned on the consumer API. This causes the mappings to decrease, causing issues in the checkpointing feature. As a result of this limitation, failover operations for transactional topics is not possible.

Cruise Control

Learn about the known issues and limitations in Cruise Control in this release:

CDPD-44676: Rebalancing with Cruise Control does not work if the metric reporter fails to report the CPU usage metric

If the CPU usage metric is not reported, the numValidWindows in Cruise Control will be 0 and proposal generation as well as partition rebalancing will not work. If this issue is present, the following message will be included in the Kafka logs:

```
WARN com.linkedin.kafka.cruisecontrol.metricsreporter.CruiseControlMetricsReporter:
       [CruiseControlMetricsReporterRunner]: Failed reporting CPU util.
```

```
java.io.IOException: Java Virtual Machine recent CPU usage is not available.
```

This issue is only known to affect Kafka broker hosts that have the following specifications:

- CPU: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
- OS: Linux 4.18.5-1.el7.elrepo.x86_64 #1 SMP Fri Aug 24 11:35:05 EDT 2018 x86_64
- Java version: 8-18

Move the broker to a different machine where the CPU is different. This can be done by moving the host to a different cluster. For more information, see Moving a Host Between Clusters

Note: Cluster nodes affected by this issue are not displayed as unhealthy.

Known Issues in Cloudera Streaming Analytics

Learn about the known issues in Cloudera Streaming Analytics clusters, the impact or changes to the functionality, and the workaround.

Cloudera SQL Stream Builder

CSA-4858 - Kerberos encryption type detection does not always work correctly for Cloudera SQL Stream Builder

Cloudera SQL Stream Builder detects no supported encryption types even though there is a list of allowed encryption types in the krb5.conf file. This causes an error when generating keytabs from the principal and password pair.

1. Run ktutil on your cluster.
2. Change the configuration with the following commands:

```
addent -password -p <username> -k 1 -e aes256-cts
wkt /tmp/new_keytab.keytab
```

3. Upload the new keytab on Streaming SQL Console.

Flink

In Cloudera Streaming Analytics, the following SQL API features are in preview:

- Match recognize
- Top-N
- Stream-Table join (without rowtime input)

CSA-5525 - Illegal join reordering in Flink optimizer

Flink optimizer's reordering might violate certain clauses (for example FOR SYSTEM_TIME AS OF) that are supported only on a specific side of a join operation, resulting in an error.

Example error message:

```
Caused by: org.apache.flink.table.api.TableException: Temporal t
able join only support apply FOR SYSTEM_TIME AS OF on the right
table
```

Set table.optimizer.join-reorder-enabled to false, until the issue is fixed in upstream Flink.

Third-party dependencies upgraded in Cloudera Public Cloud might cause Flink jobs to fail

After upgrading Cloudera Public Cloud, Flink jobs might fail due to upgraded 3rd-party dependencies. For example, this could happen with awssdk, which has been updated to version 2.23.10 in Cloudera Public Cloud version 7.2.18.

Verify your application's dependency versions against the Cloudera-supported versions before upgrading to a newer version of Cloudera Public Cloud. For more information see Updating Flink job dependencies.

DataStream conversion limitations

- Converting between Tables and POJO DataStreams is currently not supported in Cloudera Streaming Analytics.
- Object arrays are not supported for Tuple conversion.
- The java.time class conversions for Tuple DataStreams are only supported by using explicit TypeInformation: LegacyInstantTypeInfo, LocalTimeTypeInfo.getInfoFor(LocalDate/LocalDateTime/LocalTime.class).
- Only java.sql.Timestamp is supported for rowtime conversion, java.time.LocalDateTime is not supported.

Kudu catalog limitations

- CREATE TABLE
  - Primary keys can only be set by the kudu.primary-key-columns property. Using the PRIMARY KEY constraint is not yet possible.
  - Range partitioning is not supported.
- When getting a table through the catalog, NOT NULL and PRIMARY KEY constraints are ignored. All columns are described as being nullable, and not being primary keys.
- Kudu tables cannot be altered through the catalog other than simply renaming them.

Schema Registry catalog limitations

- Currently, the Schema Registry catalog / format only supports reading messages with the latest enabled schema for any given Kafka topic at the time when the SQL query was compiled.
- No time-column and watermark support for Registry tables.
- No CREATE TABLE support. Schemas have to be registered directly in the SchemaRegistry to be accessible through the catalog.
- The catalog is read-only. It does not support table deletions or modifications.
- By default, it is assumed that Kafka message values contain the schema id as a prefix, because this is the default behaviour for the SchemaRegistry Kafka producer format. To consume messages with schema written in the header, the following property must be set for the Registry client: store.schema.version.id.in.header: true.

Deprecation notices In Cloudera DataFlow for Data Hub 7.3.1

Certain features and functionalities have been removed or deprecated in Cloudera DataFlow for Data Hub 7.3.1. You must review these items to understand whether you must modify your existing configuration. You can also learn about the features that will be removed or deprecated in the future release to plan for the required changes.

Terminology

Items in this section are designated as follows:

Deprecated

Technology that Cloudera is removing in a future release. Marking an item as deprecated gives you time to plan for removal in a future release.

Moving

Technology that Cloudera is moving from a future release and is making available through an alternative Cloudera offering or subscription. Marking an item as moving gives you time to plan for removal in a future release and plan for the alternative Cloudera offering or subscription for the technology.

Removed

Technology that Cloudera has removed from the product and is no longer available or supported as of this release. Take note of technology marked as removed since it can potentially affect your upgrade plans.

Deprecation Notices for Cloudera Streaming Analytics

Certain features and functionality are deprecated or removed in Cloudera Streaming Analytics. You must review these changes along with the information about the features in Cloudera Streaming Analytics that will be removed or deprecated in a future release.

Deprecated

Support for JavaScript UDFs

Due to the deprecation of the Nashorn engine used in JDK 8 and 11, User-Defined Functions (UDFs) written in JavaScript are deprecated in Cloudera Streaming Analytics 1.13. Cloudera recommends that customers start using Python UDFs for all new developments, and start migrating their JavaScript UDFs to Python to prepare for future upgrades.

v1 API

The v1 REST API for Cloudera SQL Stream Builder has been deprecated and will be removed in a future version of Cloudera Streaming Analytics.

Customers are advised to migrate to the v2 API, available for Cloudera SQL Stream Builder.

For more information on the v2 API, see Cloudera SQL Stream Builder REST API reference.

Fixed Issues in Cloudera DataFlow for Data Hub 7.3.1

Fixed issues represent selected issues that were previously logged through Cloudera Support, but are addressed in the current release. These issues may have been reported in previous versions within the Known Issues section; meaning they were reported by customers or identified by Cloudera Quality Engineering team.

Review the list of issues that are resolved in Cloudera DataFlow for Data Hub 7.3.1.

Fixed issues in Flow Management

Review the list of Flow Management issues that are resolved in Cloudera DataFlow for Data Hub 7.3.1.

NiFi 1.28 with Cloudera Flow Management 2.2.9

Cloudera Flow Management 2.2.9 is based on Apache NiFi 1.28.0. It includes all fixed issues of this Apache NiFi release, as well as the following additional fixes:

- CFM-3673 Removed deprecation notice from downstream InvokeGRPC
- CFM-3673 Re-added InvokeGRPC processor to downstream
- CFM-3681 Be able to use downstream solr version
- CFM-3726 Added deprecation notice in GRPC processors
- CFM-3768 Exclude aws libs and put those into the common lib directory
- CFM-3775 Update scala-library version to 2.13.12 to mitigate CVE-2022-36944 and Reactor Netty client to 1.0.34 to mitigate CVE-2023-34062
- CFM-3775 Update org.json:json version to 20231013 to mitigate CVE-2023-5072
- CFM-3775 Updated snappy-java to 1.1.10.5 (CVE-2023-43642) excluded bcprov-ext references (CVE-2018-1000180, CVE-2018-1000613) updated snakeyaml to 2.2 (CVE-2017-18640)
- CFM-3775 Remove support for old Postgres versions, update Postgres to…
- CFM-3819 Change logging for debezium unit tests
- CFM-3846 Add missing Ozone dependency to PutIcebergCDC processor
- CFM-4125 CFM-4067 Bump postgresql driver version in debezium-connector-postgres lib to mitigate CVE-2024-1597
- CFM-4155 NIFI-13836 Dependency upgrades
- CFM-4201 Exclude log4j-slf4j-impl
- CFM-4289 NIFI-13720 Component is not reloaded when the isolation key depends on service property
- CFM-4289 NIFI-13722 Kerberos ticket renewal issue due static thread pool in Iceberg library
- CFM-4374 update azure-identity
- CFM-4411 Fix CVEs in bouncycastle

- CFM-4412 Fix CVE-2024-32888 in redshift
- CFM-4434 Use symlinks in nars for jar/war files (part 1+2)
- NIFI-1931 Add auto commit property to QueryDatabaseTable and QueryDatabaseTable processors to allow disabling auto commit so PostgreSQL Fetch Size will work
- NIFI-6379 Added SSL Context to PutSNS, DeleteSQS, GetSQS, and PutSQS
- NIFI-9677 Fixed issue that an empty JSON array causes flow file to be considered unmatched even though it should be considered as a match.
- NIFI-13993 Upgraded Netty to 4.1.115.Final
- NIFI-13993 Upgraded aws-java-sdk to 1.12.778 along with others
- NIFI-13993 Updated Docker version to 1.28.1
- NIFI-13993 Update Apache parent pom version to 33
- NIFI-13993 Removing pom versionsBackup files
- NIFI-13993 Set project version to 1.28.1-SNAPSHOT
- NIFI-13988 Adjusted Record number conversion to treat empty String as null
- NIFI-13963 Default to Drop Unknown Fields in JSON Record Reader
- NIFI-13991 Fix GetAwsTextractJobStatus so that a ProvisionedThroughputExceededException properly sends the flowfile to the "throttled" relationship.
- NIFI-13994 Upgraded Jackson to 2.18.1 along with others
- NIFI-13922 Fixed SplitExcel to use the evaluated formula value for cell
- NIFI-13970 Added DISCLAIMER regarding NiFi version 1 to assemblies
- NIFI-13930 PutAzureDataLakeStorage sets close flag on file write so that Azure can emit FlushWithClose event
- NIFI-13927 Use synchronized lists in PublishGCPubSub
- NIFI-13897-RC1 prepare release nifi-1.28.0-RC1
- NIFI-13971 Removed Parameter Context debug logging in Flow Synchronizer
- NIFI-13633 Set JsonRecordSetWriter.AllowScientificNotation default value to 'true' on 1.x line in order to be backward compatible
- NIFI-13755 Improved Controller Service Enabling Process
- NIFI-13860 Avoid Throwing Exceptions for Failures in IPLookupService
- NIFI-13744 Corrected Excel Reader Cell Type Inferencing
- NIFI-13798 Renamed Airtable's API Key property to Personal Access Token and updated docs due to API Keys deprecation
- NIFI-13404 recreate s3Object before returning input stream
- NIFI-13726 Set cell style copy policy to false in order to avoid exceeding the maximum number of cell styles (64000) in a .xlsx Workbook
- NIFI-13335 Added ability for the XMLRecordReader to handle where an array of data has different types.
- NIFI-13842 Fixed truststore/keystore setup in AWS v2 components
- NIFI-13840 Fixed Proxy URL Configuration in AWS v2 components
- NIFI-12532 Ensure that when CommunicateAction completes (exceptionally or otherwise) that it gets removed from the list of all CommunicationActions
- NIFI-13819 Set Row Number and Sheet Name for ExcelReader Exceptions
- NIFI-13776 Updated CopyS3Object to Handle Files over 5 GB
- NIFI-13829 Mitigate false positive reports of MonitorActivity, in case of infrequent Flow Files
- NIFI-13831 Adding inheritance to versioned component synchronizer parameter context synchronization when considering referencing components to restart
- NIFI-13831 Adding inheritance to versioned component synchronizer parameter context synchronization when considering referencing components to restart
- NIFI-13836 Dependency upgrades
- NIFI-13837 QueryRecord changes timezones of Timestamp field
- NIFI-12016 Allow use of compatible NAR bundles when loading flow from cluster connection; when determining what bundles are compatible, consider not just any bundle if it's the only one but also any bundle whose version matches the framework version so that when NiFi is upgraded, it is handled more gracefully.

- NIFI-13549 Resolve ability to go to provenance event from lineage ui when nifi is clustered
- NIFI-13722 Kerberos ticket renewal issue due static thread pool in Iceberg library
- NIFI-13720 Component is not reloaded when the isolation key depends on service property
- NIFI-13763 Fixed HashSet Filtering for DeduplicateRecord
- NIFI-13742 Normalize column names in SelectHiveQL processors
- NIFI-13727 Add DeleteSFTP processor
- NIFI-13543 Backport HttpRecordSink service to NiFi 1.x
- NIFI-13723 Add standalone RecordPath function recordOf
- NIFI-13709 Added more meaningful message when validation fails with non-compliant XML and no schema is provided
- NIFI-13715 Fixed StandardProvenanceEventRecord.hashCode() to sort Parent/Child FlowFiles as equals() does
- NIFI-13620 Resolved MaxWaitTime Issue in QueryCassandra
- NIFI-13686 Make TestListFile.testFilterAge() more resilient to time delays
- NIFI-13692 Catch All Exceptions in ResizeImage
- NIFI-13691 Added Kerberos User Service to KuduLookupService
- NIFI-13690 Upgraded AWS SDK to 2.27.14 and other dependencies
- NIFI-13605 Make AbstractHadoopProcessor.KERBEROS_USER_SERVICE public
- NIFI-13675 Fixed Tooltip for Parameter Description
- NIFI-13669 Adding alternative processor suggestion in InvokeAWSGatewayApi deprecation notice
- NIFI-13666 Applied the change to 1.X
- NIFI-13576 Upgrade Iceberg from 1.5.2 to 1.6.0
- NIFI-13430 Added CopyS3Object and GetS3ObjectMetadata
- NIFI-13655 Upgrade 1.x Shared Dependencies including JacksonXML and others
- NIFI-13640 Extract Ranger Solr version to property
- NIFI-13627 Bump azure-sdk-bom to 1.2.25 and msal4j to 1.16.1 for nifi-property-protection-azure
- NIFI-13623 Bump gcp.sdk.version to 26.40.0 for nifi-property-protection-gcp
- NIFI-13621 Upgraded JGit to 5.13.3.202401111512 for CVE-2023-4759
- NIFI-13574 Upgraded Azure SDK BOM from 1.2.23 to 1.2.25
- NIFI-13439 Add performance tracking to ProcessGroupStatus
- NIFI-13439 Add performance tracking to ProcessGroupStatus
- NIFI-12741 Remove write permission requirement for the referenced controller service when changing component property referencing a controller service through parameter
- NIFI-13593 PutIceberg issue with decimal scale
- NIFI-13573 Bump google.libraries.version from 26.37.0 to 26.40.0
- NIFI-13557 Fixed TestExcelSchemaInference for Single Digit Month
- NIFI-13557 Fixed TestSchemaInferenceUtil for Single Digit Month
- NIFI-13557 Corrected Date Time Matcher to support single digit months
- NIFI-13566 Catch Throwable during JettyServer start to ensure any issue during start will exit
- NIFI-13550 Added documentation about the ExcelReader Starting Row Strategy
- NIFI-13542 Added missing Max String Length property for JSON Readers
- NIFI-12491 Added Starting Row Schema Strategy to ExcelReader
- NIFI-13538 Do not include exception details in FlowFile attributes in DeleteFile
- NIFI-13418 Updated ExcelReader to handle spreadsheets with shared formulas
- NIFI-13461 Added DeleteFile Processor
- NIFI-13420 Maintain consistent maxNonHeapBytes for clustered diagnostics
- NIFI-13031 Changed PG StatusSnapshotDTO to use cloned copy for status
- NIFI-13496 Included Hadoop configuration file paths in the classloader isolation key of HDFS processors
- NIFI-13478 Protobuf Reader fails to coerce type of repeated fields
- NIFI-13464 Replaced nifi-deprecation-log with logger in Registry
- NIFI-13304 Added SplitExcel Processor

- NIFI-13422 Use unique instances for ScanRule implementatations for QueryNiFiReportingTask
- NIFI-12750 ExecuteStreamCommand incorrectly decodes std error stream
- NIFI-13429 Corrected EncryptContentPGP Packet Detection
- NIFI-13415 Deprecated Property Protection and Encrypt Config
- NIFI-10666 PrometheusReportingTask needs to use UTF-8 (not jvm default charset) for /metrics endpoint
- NIFI-13400 ensure container has time to startup before interacting with smb
- NIFI-12231 FetchSmb supports Move and Delete Completion Strategies
- NIFI-13356 Fixed ProtobufReader handling of repeated fields
- NIFI-13397 Updated PutDatabaseRecord to retry transient ProcessException causes
- NIFI-13411 Upgraded Spring to 5.3.37 along with other common dependencies
- NIFI-12704 Avoid NPE in escapeJson() for Root Path
- NIFI-13340 Fixed a bug in which an Output Port can leave a Process Group's DataValve open for output, but then the last FlowFile is terminated instead of going to an Output Port, ultimately resulting in the DataValve remaining open indefinitely. Now, this will be detected and the valve will be closed.
- NIFI-13379 Replaced use of deprecated com.nimbusds.oauth2.sdk.http.HTTPResponse method getContentAsJSONObject with API suggested replacement getBodyAsJSONObject.
- NIFI-13359 Tune ExecuteSQL/Record to create fewer transient flow files
- NIFI-13374 Fix tooltip for Parameter Context in new Process Group dialog
- NIFI-13030 Adding endpoint for comparing versions of registered flows (1.x support)
- NIFI-13323 Removed instantiation of Object arrays for log arguments
- NIFI-13315 Fixed ListAzureBlobStorage_v12 fails when Record Writer is used
- NIFI-13298 Removed unused instantiated java.util.HashSet from RouteAttribute
- NIFI-13294 Deprecated Apache Knox SSO Authentication
- NIFI-13296 Deprecated Kerberos SPNEGO Authentication
- NIFI-11658 Streamline using single Parameter Context for nested PGs
- NIFI-12896 Added Endpoint Override URL property for PutSNS Processor
- NIFI-12669 Fixed ByteArrayOutputStream.toString() for Java 8 in EvaluateXQuery
- NIFI-12669 Fix EvaluateXQuery processor which incorrectly encodes result attributes in certain case
- NIFI-13072 Fix MonitorActivity problems with cluster scope flow monitoring
- NIFI-13156 Replaced JsonParser deprecated getCurrentName() with currentName()
- NIFI-13208 Increased PropertyDescriptor visibility in HadoopDBCPConnectionPool
- NIFI-13172 Deprecated nifi-kafka-connect components
- NIFI-13201 Deprecated Accumulo Components for Removal
- NIFI-13191 Deprecated InvokeAWSGatewayApi
- NIFI-13181 Updated msal4j to 1.15.0
- NIFI-13151 Deprecated Couchbase Components
- NIFI-13152 Deprecated DataDogReportingTask
- NIFI-13133 RC1 prepare for next development iteration
- NIFI-13133 RC1 prepare release nifi-1.26.0-RC1
- NIFI-12231 FetchSmb supports Move and Delete Completion Strategies
- NIFI-13137 Switch to Zulu for MacOS Java 8 action
- NIFI-13008 CLI command to upgrade all instances of a versioned flow
- NIFI-13133 RC1 prepare for next development iteration
- NIFI-13133 RC1 prepare release nifi-1.26.0-RC1
- NIFI-13131 dependency updates
- NIFI-13121 Handle runtime exceptions in FetchHDFS
- NIFI-13103 Make AutoCommit default to no value set in PutDatabaseRecord
- NIFI-13006 Deprecated nifi-solr-bundle components
- NIFI-12960 Corrected default Protection Type in ExcelReader
- NIFI-12960 Support reading password-protected files in ExcelReader

- NIFI-13084 Backport Allow disabling scientific notation when writing JSON (NIFI-12697)
- NIFI-13090 Backport Improve handling of embedded JSON records (NIFI-12480)
- NIFI-12858 Corrected Order of Previous Property Values
- NIFI-13069 Deprecated ConvertAvroToJSON
- NIFI-12923 Added append avro mode to PutHDFS
- NIFI-13070 Upgraded Netty from 4.1.108 to 4.1.109
- NIFI-13066 Upgraded Bouncy Castle from 1.77 to 1.78.1
- NIFI-13064 Upgrade commons-configuration2 to 2.10.1 for Registry
- NIFI-12993 Add auto commit feature and add batch processing for the sql stmt type
- NIFI-13049 Upgraded nimbus-jose-jwt to 9.37.3 for Registry and Toolkit
- NIFI-13052 allow CRON driven components to be searchable
- NIFI-13046 Upgrade Solr dependencies to 8.11.3
- NIFI-13037 Upgraded Spring Framework from 5.3.31 to 5.3.34
- NIFI-13040 Upgraded Commons IO from 2.15.1 to 2.16.1
- NIFI-13025 Removed custom validation from NifiRegistryFlowRegistryClient
- NIFI-12890 Refactor HadoopDBCPConnectionPool to extend AbstractDBCPConnectionPool
- NIFI-12614 Create record reader service for Protobuf messages (1.x version)
- NIFI-12889 Retry Kerberos login on auth failure in HDFS processors
- NIFI-12837 Fix checkstyle issue following a manual cherrypick
- NIFI-13010 Fix UpdateDatabaseTable to work with DBCPConnectionPoolLookup
- NIFI-12837 Added DFS support in SMB processors
- NIFI-13012 Upgraded Apache Tika from 2.9.1 to 2.9.2
- NIFI-12984 Bump Snowflake Ingest SDK to 2.1.0
- NIFI-12918 Corrected Nested Versioned Flows for Stateless
- NIFI-13002 Restored zstd Compression Level in CompressContent
- NIFI-12996 Moved zstd-jni to standard-shared-nar
- NIFI-12969 Fixed a typo in the #isTempDestinationNecessary method, where we were comparing existingConnection.getProcessGroup() to newDestination.getProcessGroup(), instead of comparing existingConnection.getDestination().getProcessGroup() to newDestination.getProcessGroup(). I.e., we were comparing the Destination's PG to the Connection's PG instead of comparing Destination's PG to Destination's PG. This resulted in using a temporary funnel when we shouldn't. And as a result it attempted to change a Connection's Destination while the destination was running.
- NIFI-12987 allow controller service type to be searchable
- NIFI-12980 Deprecated Hive 3 Components for Removal
- NIFI-12979 Upgraded Kotlin from 1.9.10 to 1.9.23
- NIFI-12966 Upgraded Netty from 4.1.106 to 4.1.108
- NIFI-12900 Avoid unnecessary file listing in PutSFTP
- NIFI-12957 Upgraded Azure SDK BOM from 1.2.19 to 1.2.21
- NIFI-12939 Retry Kerberos login on authentication failure in Iceberg processors
- NIFI-12888 In AbstractEmailProcessor check for expired oauth2 token when determining whether mail receiver needs to be recreated.
- NIFI-12954 Upgraded AWS BOM from 2.23.3 to 2.25.16
- NIFI-12949 Upgraded Box SDK from 4.6.1 to 4.8.0
- NIFI-12925 Updated ListenHTTP to return 405 for TRACE and OPTIONS
- NIFI-12930 Catch FlowFileAccessException in FetchFile
- NIFI-12887 Added Binary String Format property to PutDatabaseRecord
- NIFI-12943 Upgraded Hadoop from 3.3.6 to 3.4.0
- NIFI-12947 Upgraded MIME4J to 0.8.11
- NIFI-12936 ListGCSBucket resets its tracking state after configuration change
- NIFI-12944 Add PeerAddress as Attribute into the flowfile
- NIFI-12928 Added Simple Write strategy in PutAzureDataLakeStorage

- NIFI-12929 Fix logout infinite redirect loop in case of knox
- NIFI-12895 Added Timeout property to GetSmbFile and PutSmbFile
- NIFI-12938 Upgrade Iceberg from 1.4.3 to 1.5.0
- NIFI-12931 Upgraded Commons Configuration from 2.9.0 to 2.10.1
- NIFI-12926 Upgraded Jackson from 2.16.2 to 2.17.0
- NIFI-12503 Render from-data in swagger.json and RestAPI docs correctly
- NIFI-12919 Deprecated Cassandra 3 Components for Removal
- NIFI-12877 Added Sensitive Dynamic Property Support to RestLookUpService
- NIFI-12911 Upgraded Jagged from 0.3.0 to 0.3.2
- NIFI-12909 Upgraded Nimbus JOSE+JWT from 9.33.0 to 9.37.3
- NIFI-12700 refactored PutKudu to optimize memory handling for AUTO_FLUSH_SYNC flush mode (unbatched flush)
- NIFI-12901 Removed time units in description of time period properties
- NIFI-12906 Upgraded ZooKeeper from 3.9.1 to 3.9.2
- NIFI-12886 Upgraded Jackson JSON from 2.16.1 to 2.16.2
- NIFI-12840 Expose REMOTE_POLL_BATCH_SIZE property for ListSFTP
- NIFI-12871 Upgraded Commons Compress from 1.25.0 to 1.26.1
- NIFI-12879 Upgraded Clojure from 1.11.1 to 1.11.2
- NIFI-12785 Corrected InvokeHTTP URL handling to avoid double encoding
- NIFI-12876 Upgraded Surefire Plugin from 3.1.2 to 3.2.5
- NIFI-12874 Upgraded Log4j from 2.20.0 to 2.23.0
- NIFI-12868 Upgraded Commons DBCP from 2.11.0 to 2.12.0
- NIFI-12860 Fixed NPE in ExtensionMetadata Constructor
- NIFI-12846 Fixed Region handling for AWS Assume Role Credentials with VPCE Endpoint URL
- NIFI-12645 Fix to correctly invoke onStopped method of scripted processor
- NIFI-12850 Prevent indexing of overly large filename attribute
- NIFI-12828 Added Mapping for BIT type to return INT and handled boolean case for postgres
- NIFI-12851 ConsumeKafka, remove limitation on count of subscribed topics
- NIFI-12847 Add Enum data type handling to Iceberg record converter
- NIFI-12843 Fix incorrect read of parquet data, when record.count is inherited
- NIFI-12843 Fix incorrect read of parquet data, when record.count is inherited
- NIFI-12839 Explicitly set nifiVersion for processor bundle archetype dependencies
- NIFI-11859 Ensure Hazelcast can not join a network when Cluster is NONE
- NIFI-12835 Upgraded node-ip from 1.1.8 to 1.1.9 for Registry
- NIFI-12826 Added timing lag in TestFTP method for improved stability
- NIFI-12784 Set Path Not Found as a Dependent Property in EvaluateJsonPath
- NIFI-12827 Upgraded PostgreSQL JDBC test driver from 42.6.0 to 42.7.2
- NIFI-12232 Corrected Group Component ID Handling for Clustered Flows
- NIFI-12818 Deprecated ReportLineageToAtlas for Removal
- NIFI-12808 Upgraded Commons Codec from 1.16.0 to 1.16.1
- NIFI-12810 Upgraded SLF4J from 2.0.11 to 2.0.12
- NIFI-12725 Upgraded json-schema-validator from 1.1.0 to 1.3.2
- NIFI-12789 fix broken link in couchbase additional details
- NIFI-12770 Deprecated Ranger Authorizers for Removal
- NIFI-12792 Deprecated nifi-spark-bundle components for removal
- NIFI-12777 Add support for UUID record field type in QueryRecord processor
- NIFI-12779 Upgraded Okio from 3.7.0 to 3.8.0
- NIFI-12769 Updated copyright year to 2024 in NOTICE file headers
- NIFI-12680 Fixed JAR for DefaultedDynamicClassPathModificationIT
- NIFI-12745 Fix AvroReader silently dropping malformed records

- NIFI-12749 Handled Forward Slash in Flow Name for nifi-toolkit-cli
- NIFI-12732 ListS3 resets its tracking state after configuration change
- NIFI-12728 Upgraded brotli4j from 1.13.0 to 1.16.0
- NIFI-12729 Upgraded unboundid-ldapsdk from 6.0.10 to 6.0.11
- NIFI-12730 Upgraded Spring Integration from 5.5.18 to 5.5.20
- NIFI-12441 Added No Tracking Strategy to ListS3
- NIFI-12726 Update commons-email to 1.6.0
- NIFI-12731 Ensure state is updated in GetHBase whenever the session is committed
- NIFI-12715 Updated Snowflake SDKs
- NIFI-12719 Upgraded metrics-core from 4.2.22 to 4.2.25
- NIFI-12718 Upgraded greenmail from 1.6.14 to 1.6.15
- NIFI-12717 Upgraded Gremlin from 3.7.0 to 3.7.1
- NIFI-12713 Upgraded mysql-binlog-connector from 0.28.3 to 0.29.0
- NIFI-12695 Enabled PKCE Support for OIDC Integration
- NIFI-12699 Set timeout to 10 seconds for TestStandardFlowFileQueue.testListFlowFilesResultsLimitedCollection
- NIFI-12691 Update okio to 3.7.0
- NIFI-12706 Update reactor-test to 3.5.14
- NIFI-12688 Upgrade mysql-connector-j to 8.3.0
- NIFI-12705 Update metrics-jvm to 4.2.25
- NIFI-12692 Update jline.version to 3.25.1
- NIFI-12690 Upgraded opentelemetry-proto from 1.0.0 to 1.1.0
- NIFI-12689 Upgraded Testcontainers from 1.19.3 to 1.19.4
- NIFI-12682 Fix MiNiFi agent manifest hash swaps
- NIFI-12677 Removed documentation of non-existent strategy for ExcelReader
- NIFI-12500 Add dynamic target for Get/Set/SendTrapSNMP

NiFi 2.0 with Cloudera Flow Management 4.2.1

Cloudera Flow Management 4.2.1 is based on Apache NiFi 2.0.0-M2. It includes all fixed issues of this Apache NiFi release, as well as the following additional fixes:

- CFM-2279 Added nifi-cdf-flow-analysis-rules-bundle and multiple rule implementations.
- CFM-2663 Add Kite module back - Revert "NIFI-9591 Removed nifi-kite-bundle" - Set NiFi version to 1.16.0
- CFM-2663 Add NARs shipped in CFM to nifi-assembly
- CFM-2900 Fix build error, use com.google.cloud.bigdataoss downstream versions There was a StackOverFlow error, because ozone-common downstream lib transitive dependency had a collision with com.google.cloud.bigdataoss dependencies. As turned out, com.google.cloud.bigdataoss has downstream versions as well, which are compatible with other downstream libs, so we have to use that one - CFM-2900: Fix atlas downstream errors Extend interface with new method + remove EmbeddedKafka as no longer support that way - CFM-2925: Be compatible with hive downstream library - CFM-2946: Ignoring failing Atlas tests, which work downstream - CFM-2946: Updating hive3 version to downstream version 3.1.3000.7.2.16.0-147 - CDPDFX-5826 - fix log4j exclusion - Removing log4j dependencies pulled in with hadoop version 3.1.1.7.2.16.0-171 - Adding more log4j exclusions for hadoop-cloud-storage - CDPDFX-6081 Resolve issues after rebase to 1.18.0 - CDPDFX-6081 Fix more versions after rebase
- CFM-2946 Updating version to 1.17.0, and excluding log4j from kite bundle
- CFM-3219 Downstream library derby lib scope is runtime which cause build failure, exclude extra org.slf4j:slf4j-reload4j dependencies come from downstream libs
- CFM-3219 exclude extra org.slf4j:slf4j-reload4j dependencies come from downstream libs
- CFM-3680 fix dependency collision issue
- CFM-3686 Fix snappy cve error + add some rules from apache nifi
- CFM-3719 Fix hive-jdbc version in nifi-hive-bundle
- CFM-3742 Exclude orc-core from hive3

- CFM-3761 In nifi-cdf-flow-analysis-rules adding RestrictThreadPoolSize in src/main/resources/META-INF/services/org.apache.nifi.flowanalysis.FlowAnalysisRule.
- CFM-3775 Remove support for old Postgres versions, update Postgres to…
- CFM-4140 NIFI-13675 Fixed Tooltip for Parameter Description
- CFM-4273 Fix concurrent issues - not use virtual thread for this release
- CFM-4360 Create configured python temp directory if it does not exists
- CFM-4360 Add python package install tempdir NiFi property
- CFM-4411 Fix CVEs in bouncycastle
- CFM-4412 Fix CVE-2024-32888 in redshift
- CFM-4374 update spring-core to latest 6.0.x
- CDPDFX-2216 Use the SAN from X509Certificate instead of CN
- CDPDFX-2216 Refactored methods incompatible with Java 8
- CDPDFX-2216 Corrected character set usage and dependency declaration
- CDPDFX-2216 Added HeaderX509CertificateExtractor
- CDPDFX-2216 Switch to using StringUtils.isNotBlank
- CDPDFX-2216 Add dev mode groups to anonymous user when dev mode is enabled
- CDPDFX-2470 Ensure anonymous user has idp groups populated - Remove RunOnceIT - CDPDFX-2597: Appending the context path to any entry in localStorage.
- CDPDFX-4049 Initial setup of FlowDesignerClient and Redis implementation
- CDPDFX-6081 More fixes after rebase to 1.18.0 - CPDFX-6081 Fix ignite NAR version to 1.18.0 - CDPDFX-6081 Fix versions in nifi-hashicorp-vault poms - CDPDFX-6081 Use regular gcs version in poms, RE scripts will do version replacement - CDPDFX-6081 Add addition log4j 2 exclusions to ranger poms
- CDPDFX-6130 Adding null check before resolving sensitive parameter
- CDPDFX-6285 Use flow designer id in flow control event handler
- CDPDFX-6285 Passing through flow designer id during component creation
- CDPDFX-6813 Add more bouncycastle exclusions, fix poms
- CDPDFX-6813 Fix use of flow comparator in standard flow change event handler
- CDPDFX-6813 Fix pom versions to 1.20.0 after rebase - CFM Fix problems, caused by downstream library differencies - CFM: Some reason releng build is unable to resolve guava from asana dep pom.xml [ERROR] Failed to execute goal on project nifi-asana-services-api: Could not resolve dependencies for project org.apache.nifi:nifi-asana-services-api:jar:1.18.0.2.1.6.0-33: Failed to collect dependencies at com.asana:asana:jar:1.0.0 -> com.google.guava:guava:jar:[31.1,): No versions available for com.google.guava:guava:jar:[31.1,) within specified range -> [Help 1] - CFM Fix asana guava dependency resolving - CFM - Fix guava resolving problem Some reason downstream library is not able to resolve guava version, when version is defined with [31.1,) format
- CDPDFX-6813 Remove duplicate NAR in assembly - CDPDFX-6831 Fix publishing of controller service bulletins
- CDPDFX-6848 Ensuring that all child components are included in each published BulletinGroup. This separation allows them to not be pruned when there are many other descendent components.
- CDPDFX-6854 Correcting process group ID mapping in flow bulletin task
- CDPDFX-7171 Correcting resolution of VCI for scheduled state changes
- CDPDFX-7192 Add lastModifier field to FlowChangeEvent, set lastModifier when publishing state change events
- CDPDFX-7379 Use identity state lookup when determining affected components for a PG sync, Ensure VCI is set back to original value if an exception is encountered during pg sync, Updated StandardScheduledStateListener to use VCI from the passed in FD PG
- CDPDFX-7395 Resolving bug in bulk actions as applied to nested groups
- CDPDFX-7445 Adding requestId to FlowChangeEvent
- CDPDFX-7517 Fixing NPE when publishing flow change events for bulk actions, publishing events on initially stopping components
- CDPDFX-7757 Fixes after rebasing to Apache 1.23.2
- CDPDFX-7815 Fix atlas dependencies for upgrading to CDP 7.2.17 dependencies
- CDPDFX-8320 Removing nifi-pulsar-nar, which was not compatible with 2.X

- CDPDFX-8348 Updating NiFiRecordSerDe for NIFI-9458 changes
- CDPDFX-8350 Adding python extensions autoloading
- CDPDFX-8623 Updating snakeyaml and postgresql versions
- CDPDFX-8623 Updating COS API version, Upgrading dependencies for CVEs, removing Kite, Upgrading dependencies for CVEs, Remediating Heimdall findings
- CDPDFX-8804 Adding Swagger 1 annotations to domain model
- CDPDFX-8878 Adding support for createConfigurationContext in StandardFlowChangeEventHandler - This allows Flow Designer to continue to work with recent changes from 2.0.0-M2
- CDPDFX-9164 Fixing FlowDesignerEventIT
- CDPDFX-9174 Correcting loadBalancerStrategy enum on VersionedConnection
- CDPDFX-9263 extend manifest generation to add dependencies for property descriptors in case of python processors
- NIFI-10976 Added Previous Cluster State Provider configuration
- NIFI-13722 Kerberos ticket renewal issue due static thread pool in Iceberg library
- NIFI-13593 PutIceberg issue with decimal scale
- NIFI-12939 Retry Kerberos login on authentication failure in Iceberg processors
- NIFI-12697 Allow disabling scientific notation when writing JSON
- NIFI-13872 extend manifest generation to add dependencies for property descriptors in case of python processors
- NIFI-13971 Removed Parameter Context debug logging in Flow Synchronizer
- NIFI-13831 Adding inheritance to versioned component synchronizer parameter context synchronization when considering referencing components to restart
- NIFI-13831 Adding inheritance to versioned component synchronizer parameter context synchronization when considering referencing components to restart
- NIFI-13184 mark invalid processors to start once they become valid
- NIFI-12755 Upgraded Jetty from 12.0.5 to 12.0.6
- NIFI-13365 Fix unit tests running in kubernetes pod
- NIFI-13604 Python Source processors can be triggered without creating new FlowFiles
- NIFI-13528 fixed Python processor's customValidate function
- NIFI-13427 Added FlowFileSource interface for Python Processors
- NIFI-13396 Added Python constant Allowable Values to Manifests
- NIFI-13394 Check candidate directory for python command
- NIFI-13324 Set FlowFile attributes for Python Processors on failure
- NIFI-13433 Fixed PutChroma handling of list values in properties
- NIFI-13118 Add LANGUAGE to property_descriptor list
- NIFI-13042 Support Python 3.12 for Python Processors
- NIFI-12970 Generate documentation for Python Processors
- NIFI-12740 Fixed Python to Java Object Binding
- NIFI-12959 Support loading Python processors from NARs
- NIFI-12913 Corrected NPE for Python Log Listener ID
- NIFI-12514 Added Windows support for Python venv
- NIFI-12884 Corrected documentation for python debugging
- NIFI-11443 Route Python Framework Logging to SLF4J
- NIFI-12791 Added pillow-heif to ParseDocument Processor
- NIFI-12232 Corrected Group Component ID Handling for Clustered Flows
- NIFI-12693 Moved notification of python process that a Processor was removed to a background (virtual) thread. Also noted in testing that in one instance a Python Processor never became valid because it had cached property descriptors before the processor was fully initialized, so updated code to ensure that we do not cache values before initialization is completed.
- NIFI-12659 Respawn Python Processes on Unexpected Termination
- NIFI-12675 Fixed custom Relationships with Python Processors

- NIFI-12757 Issue GC commands to Python for FlowFileTransformResults and RecordTransformResults when no longer needed on Java side
- NIFI-12739 Import ProcessPoolExecutor to fix bug in python 3.9+
- NIFI-12740 Fixed Threading Bug with Java to Python Bound Objects
- NIFI-12675 Fixed custom Relationships with Python Processors
- NIFI-12616 Added Processor Documentation Support for Python
- NIFI-12591 Upgraded from Swagger Annotations 1.6.12 to 2.2.20
- NIFI-12766 Fixed Region handling for AWS Assume Role Credentials
- NIFI-12636 Upgrade dependencies for Pinecone, ChromaDB and OpenAI processors
- NIFI-12676 Fixed Servlet Registration in HandleHttpRequest
- NIFI-12394 Fixed Service references for Migrated Configurations
- NIFI-11739 Add ability to ignore missing fields in PutIceberg
- NIFI-11177 Add defensive code for null values for Iceberg
- NIFI-11716 Backported nifi-schema-inference-utils from NIFI-11241
- NIFI-11907 Backport from NIFI-11241 for JSON Schema Inference
- NIFI-11916 Iceberg processor extensibility improvement
- NIFI-11334 Fixed PutIceberg processor instance interference due to same class loader usage
- NIFI-11817 Improve ListHDFS extensibility
- NIFI-11178 Improve ListHDFS performance, incremental loading refactor.
- NIFI-10976 Replaced toUnmodifiableList() with toList() for Java 8
- NIFI-10975 Add Kubernetes Leader Election and State Provider
- NIFI-12646 Set Python Processor version to 2.0.0-M2
- NIFI-12668 Fix conflict in Registry Git provider with gpg.format=ssh
- NIFI-12664 Removed deprecated DMC in GetHBase
- NIFI-12676 Fixed Servlet Registration in HandleHttpRequest
- NIFI-12611 Introduce the ability to view and clear state for extension types that support state.
- NIFI-12500 Add dynamic target for Get/Set/SendTrapSNMP
- NIFI-12387 Initialize Controller Service Comments with Empty String
- NIFI-12666 Corrected Registry Data Source Configuration
- NIFI-12596 PutIceberg is missing case-insensitive Record type handling in List and Map types
- NIFI-12660 Added missing Filter property to QueryPinecone
- NIFI-12657 Removed MiNiFi C2 Server modules
- NIFI-12656 Upgraded vite for new frontend from 4.5.1 to 4.5.2
- NIFI-12653 Upgraded Spring from 6.0.15 to 6.0.16
- NIFI-12654 Upgraded Netty from 4.1.105 to 4.1.106
- NIFI-12652 Upgraded SLF4J from 2.0.9 to 2.0.11
- NIFI-12650 Upgraded json-path from 2.8.0 to 2.9.0
- NIFI-12506 Added Threading for Status Analytics Retrieval
- NIFI-12501 Encrypt MiNiFi bootstrap properties
- NIFI-12554 Moved JoltTransformJSON and JoltTransformRecord to nifi-jolt-nar
- NIFI-12647 Added MultiProcessorUseCase for ListFile/FetchFile together
- NIFI-8606 Added Disable & Configure button to the Controller Services Details dialog
- NIFI-12629 adding metadata filtering to QueryPinecone
- NIFI-12402 Added Wait for Activity to MonitorActivity
- NIFI-12506 Added Threading for Status Analytics Retrieval
- NIFI-12640 Moved servlet-api and jetty-schemas to nifi-jetty-bundle
- NIFI-12634 Ignored Blank Prefix Values in Kubernetes Components
- NIFI-12616 Added Processor Documentation Support for Python
- NIFI-12638 Add Use Case on how to use QueryRecord as a router
- NIFI-12637 Handle migrating Proxy properties for InvokeHTTP

- NIFI-12635 Upgraded Slack client from 1.36.1 to 1.37.0
- NIFI-12631 Upgraded Apache MINA SSHD from 2.11.0 to 2.12.0
- NIFI-12628 Upgraded Netty from 4.1.104 to 4.1.105
- NIFI-12627 Extract nifi-file-transfer from nifi-standard-processors
- NIFI-12613 Renamed asDescribedValue() to asAllowableValue()
- NIFI-11294 Support Component State Checkpoints in ConsumeAzureEventHub
- NIFI-12625 Listed Supported Python Versions in Docs
- NIFI-12619 Fixed Python dependencies in ParseDocument
- NIFI-12590 Added Prefix Properties for Kubernetes Leases and ConfigMaps
- NIFI-12394 Fixed Service references for Migrated Configurations
- NIFI-12604 Empty Queue
- NIFI-12621 Upgraded AWS SDK from 2.20.148 to 2.23.3
- NIFI-12620 Upgraded JLine from 3.24.1 to 3.25.0
- NIFI-8278 Fixed Proxy Service property in Azure Storage processors
- NIFI-12623 Expose ability to fetch User Details in ListenSlack and receive App Mention events
- NIFI-11958 Added PutAzureDataExplorer and StandardKustoQueryService
- NIFI-12596 PutIceberg is missing case-insensitive Record type handling in List and Map types
- NIFI-9458 Replaced SimpleDateFormat with DateTimeFormatter
- NIFI-12615 fix ExpressionChanged error on Counters page. NIFI-12618 Upgraded Azure SDK BOM from 1.2.18 to 1.2.19
- NIFI-11389 Fixed controller services's link to referencing controller
- NIFI-12441 Added No Tracking Strategy to ListS3 NIFI-12593 Added Include all violations property to ValidateCsv
- NIFI-12597 Introducing a common navigation bar across all pages
- NIFI-12612 In asn1 bundle handle OBJECT IDENTIFIER type as string.
- NIFI-12573 Improved support for Enums in ProperyDescriptor.Builder
- NIFI-11288 Add AWS STS dependency for AssumeRolWithWebIdentity method
- NIFI-8278 Added Credentials Type to ADLSCredentalsControllerService
- NIFI-12608 Add nifi-standard-services-api-nar to Processor Archetype
- NIFI-12610 Corrected default_value example in Python Developer guide
- NIFI-12594 ListS3 - observe min/max object age when entity state tracking is used
- NIFI-12588 Flow Analysis Rules
- NIFI-12606 Upgrade parent Apache POM to version 31
- NIFI-12607 remove kernel 2.6 TIMED_WAIT documentation
- NIFI-12589 Queue Listing
- NIFI-12561 Fixed MergeContent DELIMITER_STRATEGY_NONE Handling
- NIFI-12599 Added READ_FILESYSTEM Permissions to Lookup Services
- NIFI-12548 Policy Management
- NIFI-12602 Upgraded follow-redirects from 1.15.2 to 1.15.4
- NIFI-12600 Upgraded Apache Maven from 3.9.5 to 3.9.6
- NIFI-12592 Upgraded Apache Curator from 5.5.0 to 5.6.0
- NIFI-12530 Support CREATE TABLE in Oracle database adapters
- NIFI-12591 Upgraded from Swagger Annotations 1.6.12 to 2.2.20
- NIFI-12572 Updated nifi-azure-bundle using current API methods
- NIFI-12434 Upgraded Registry to Spring Framework 6.1.1
- NIFI-12089 Fix typo in additionalDetails of CSVReader
- NIFI-11583 Removed nifi-ignite-nar module from assembly

Fixed issues in Edge Management [Technical Preview]

Learn about the fixed issues in Edge Management clusters, the impact or changes to the functionality, and any available workaround.

For Edge Management fixed issues, see the Cloudera Edge Management documentation.

Fixed Issues in Streams Messaging

Review the list of Streams Messaging issues that are resolved in Cloudera DataFlow for Data Hub 7.3.1.

Kafka

CDPD-65649: ReplicaAlterLogDirs stuck with Offset mismatch for the future replica

This is a backported fix, see KAFKA-9087 for more information.

CDPD-66986: Mirrormaker 2 auto.offset.reset=latest not working

This is a backported fix, see KAFKA-13988 for more information.

OPSAPS-71258: Kafka, Streams Replication Manager, and Streams Messaging Manager cannot process messages compressed with Zstd or Snappy is /tmp is mounted as noexec

The issue is fixed by using JVM flags that point to a different temporary folder for extracting the native library.

CDPD-71433: Connect logical type null values are not handled in AvroConnectTranslator

When the time.precision.mode property is set to connect for the Debezium connector, the connect logical types are used and null values are now handled.

OPSAPS-69481: Some Kafka Connect metrics missing from Cloudera Manager due to conflicting definitions

Cloudera Manager now registers the metrics kafka_connect_connector_task_metrics_batch_size_avg and kafka_connect_connector_task_metrics_batch_size_max correctly.

Schema Registry

OPSAPS-68708: Schema Registry might fail to start if a load balancer address is specified in Ranger

Schema Registry now always ensures that the address it uses to connect to Ranger ends with a trailing slash (/). As a result, Schema Registry no longer fails to start if Ranger has a load balancer address configured that does not end with a trailing slash.

Streams Messaging Manager

OPSAPS-71258: Kafka, Streams Replication Manager, and Streams Messaging Manager cannot process messages compressed with Zstd or Snappy is /tmp is mounted as noexec

The issue is fixed by using JVM flags that point to a different temporary folder for extracting the native library.

CDPD-72543: Security headers are not set for static files in Streams Messaging Manager

Streams Messaging Manager now applies the following security-related headers to static files:

- Content-Security-Policy
- X-XSS-PROTECTION
- X-Content-Type-Options
- X-Frame-Options
- Strict-Transport-Security

CDPD-73643: Unused CM_USER parameter is visible in /cm-configs internal endpoint

The unused CM_USER field has been removed from the /cm-configs internal endpoint

CDPD-70313: KNOX does not send Authentication header on FIPS configuration

KNOX now sends the Authentication header on FIPS clusters.

Streams Replication Manager

OPSAPS-71258: Kafka, Streams Replication Manager, and Streams Messaging Manager cannot process messages compressed with Zstd or Snappy is /tmp is mounted as noexec

The issue is fixed by using JVM flags that point to a different temporary folder for extracting the native library.

Cruise Control

OPSAPS-69978: Cruise Control capacity.py script fails on Python 3

The script querying the capacity information is now fully compatible with Python 3.

Fixed Issues in Cloudera Streaming Analytics

Review the list of Cloudera Streaming Analytics issues that are resolved in Cloudera DataFlow for Cloudera Data Hub 7.3.1.

CSA-5423 - Extend Cloudera SQL Stream Builder diag bundle data points

CSA-5440 - Permit Spring Flyway plugin execution on PvC

CSA-5364 - Add number of topics/tables to successful data source validation message on UI

CSA-5306 - Cloudera SQL Stream Builder API does not validate catalog type

CSA-5362 - Update "ssb-sse" ASCII text banner to not contain special characters

CSA-5359 - Improve error message when creating a JS UDF with a Java version that doesn't support it

CSA-5296 - Samples table fields are limited to 32 characters in mysql and oracle dbs

CSA-5324 - Cloudera SQL Stream Builder default admin does not have admin privileges

CSA-5428 - Sampling renders null as Invalid Number in some cases

CSA-5474 - Cloudera SQL Stream Builder can't execute any jobs due to permission issue in the artifacts directory

CSA-5475 - Local-kafka connector template not showing in Cloudera SQL Stream Builder

CSA-5479 - Using Temp View based on kudu lookup table leaks eventpolls

CSA-5499 - Bump Avro to 1.11.4 in parcel to mitigate CVE-2024-47561

Cannot submit SQL jobs with UDF JARs when checkpoints enabled

CSA-5048 - Generate correct default log configuration for Cloudera SQL Stream Builder jobs

CSA-5055 - Backport FLINK-20539 to Cloudera Streaming Analytics

CSA-5065 - Artifact Storage request thread does not timeout when storage is offline, hanging the UI

CSA-5120 - Connector dependencies are missing from Cloudera SQL Stream Builder

CSA-5122 - Cloudera SQL Stream Builder keeps reconciling after a failed job

CSA-5126 - Undeterministic classloader behavior with flink-metrics-kafka and kafka-connector resulting in job failure

CSA-5161 - analyzeQuery returning false validation errors in some cases

CSA-5166 - Cloudera SQL Stream Builder Local Kafka data source doesn't work with a user-specified TLS truststore

CSA-5199 - Cloudera registry catalog type registry is not compatible with UI

CSA-5221 - SsbCatalog uses user session in prod mode

CSA-5236 - Implement MetricReporterFactory for KafkaMetricsReporter

CSA-5251 - UiConfigController throws NullPointerException

CSA-5270 - If custom log is set, the job cannot be saved

CSA-5282 - Check Flink job status before submitting Cloudera SQL Stream Builder job

CSA-5283 - [Cloudera SQL Stream Builder] Make all overloaded methods transactional in JobService

CSA-5303 - Fix Kerberos/SPNEGO authentication for Flink Deployments

CSA-5306 - Cloudera SQL Stream Builder API does not validate catalog type

CSA-5315 - Load balancer role cannot start

FLINK-20539 - Type mismatch when using ROW() in computed column

UI fixes and improvements

CSA-4602 - Changing existing MV filter type can't be saved

CSA-5038 - Widget is empty when added to the dashboard before initialization completes

CSA-5140 - Fix No Rows To Show message when switching from sampler to MV in dashboard preview

CSA-5024 - Polling samples feedback is on even when polling is turned off

CSA-5025 - Cursor jumps to the end after the first keystroke when using templates

CSA-5026 - Oversize widget cannot be sized down

CSA-5294 - Add job save button to job settings component

Fixed CVEs in Cloudera DataFlow for Data Hub 7.3.1

Review the list of Common Vulnerabilities and Exposures (CVEs) that are resolved in Cloudera DataFlow for Data Hub 7.3.1.

CVE-2021-45105 & CVE-2021-44832 remediation for CDF for Data Hub

Learn more about the CVE-2021-45105 and CVE-2021-44832 remediation for the Flow Management, Streams Messaging and Streaming Analytics cluster templates in Cloudera DataFlow for Data Hub.

On February 1, 2022, Cloudera released a hotfix to Public Cloud Runtime version 7.2.12. It addresses the CVE and other vulnerability concerns as listed below:

- CVE-2021-45105 which affects Apache Log4j2 versions from 2.0-beta9 to 2.16.0, excluding 2.12.3
- CVE-2021-44832 which affects Apache Log4j2 versions from 2.0-alpha7 to 2.17.0, excluding 2.3.2 and 2.12.4

The following table summarizes which template is impacted by the vulnerabilities:

Template	Impacted versions
Flow Management	All versions

Template	Impacted versions
Streams Messaging	Not impacted
Streaming Analytics	All versions from 7.2.10

As the Cloudera DataFlow for Data Hub cluster templates are running in the Cloudera Public Cloud environment powered by Runtime, Cloudera encourages users to upgrade the Cloudera services running Runtime versions from 7.2.7 so that they include the latest hotfixes. You can update your existing data lakes and data hubs by doing a maintenance upgrade. For more information, see the Data Lake upgrade and Data Hub upgrade documentation.

Note: Maintenance upgrades are not supported for RAZ-enabled environments.

If you are running a version of Runtime lower than 7.2.7, contact Cloudera Support for details on how to upgrade Runtime.

For more information about the impacts of CVE-2021-45105, see the TSB 2021-547: Critical vulnerability in log4j2 CVE-2021-45105 Knowledge Base article.

Fixed CVEs in Flow Management

Review the list of Common Vulnerabilities and Exposures (CVEs) fixed in Cloudera Flow Management in Data Hub 7.3.1.

All known NiFi CVEs are addressed in both clusters based on NiFi 1.28 and clusters based on NiFi 2.0-M2. See Apache NiFi Security for more information about NiFi's CVEs.

In Flow Management clusters using NiFi 1.28, vulnerability scanners may detect certain CVEs in some legacy components. For these components, it is not possible to update the client library NiFi depends on. You can find the list of affected components below. Although NiFi does not expose ways to exploit those vulnerabilities, you may want to remove the associated NARs. Note that these NARs are deprecated and no longer available in NiFi clusters using NiFi 2.0.

- nifi-kite-nar (CVE-2022-42889, CVE-2023-39410)
- nifi-kafka-1-0-nar, nifi-kafka-2-0-nar (CVE-2018-17196)
- nifi-couchbase-nar (CVE-2020-9040)

The following CVEs have been fixed in Flow Management in Data Hub 7.3.1

CVE-2022-40149, CVE-2022-40150

Those using Jettison to parse untrusted XML or JSON data may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow. This effect may support a denial of service attack.

CVE-2022-45685

A stack overflow in Jettison before v1.5.2 allows attackers to cause a Denial of Service (DoS) via crafted JSON data.

CVE-2022-45693

Jettison before v1.5.2 was discovered to contain a stack overflow via the map parameter. This vulnerability allows attackers to cause a Denial of Service (DoS) via a crafted string.

CVE-2023-1436

An infinite recursion is triggered in Jettison when constructing a JSONArray from a Collection that contains a self-reference in one of its elements. This leads to a StackOverflowError exception being thrown.

CVE-2021-23358

The package underscore from 1.13.0-0 and before 1.13.0-2, from 1.3.2 and before 1.12.1 are vulnerable to Arbitrary Code Injection via the template function, particularly when a variable property is passed as an argument as it is not sanitized.

CVE-2024-1597

pgjdbc, the PostgreSQL JDBC Driver, allows attacker to inject SQL if using PreferQueryMode=SIMPLE. Note this is not the default. In the default mode there is no vulnerability. A placeholder for a numeric value must be immediately preceded by a minus. There must be a second placeholder for a string value after the first placeholder; both must be on the same line. By constructing a matching string payload, the attacker can inject SQL to alter the query,bypassing the protections that parameterized queries bring against SQL Injection attacks. Versions before 42.7.2, 42.6.1, 42.5.5, 42.4.4, 42.3.9, and 42.2.28 are affected.

CVE-2022-31197

PostgreSQL JDBC Driver (PgJDBC for short) allows Java programs to connect to a PostgreSQL database using standard, database independent Java code. The PGJDBC implementation of the `java.sql.ResultRow.refreshRow()` method is not performing escaping of column names so a malicious column name that contains a statement terminator, e.g. `;`, could lead to SQL injection. This could lead to executing additional SQL commands as the application's JDBC user. User applications that do not invoke the `ResultSet.refreshRow()` method are not impacted. User application that do invoke that method are impacted if the underlying database that they are querying via their JDBC application may be under the control of an attacker. The attack requires the attacker to trick the user into executing SQL against a table name who's column names would contain the malicious SQL and subsequently invoke the `refreshRow()` method on the ResultSet. Note that the application's JDBC user and the schema owner need not be the same. A JDBC application that executes as a privileged user querying database schemas owned by potentially malicious less-privileged users would be vulnerable. In that situation it may be possible for the malicious user to craft a schema that causes the application to execute commands as the privileged user. Patched versions will be released as `42.2.26` and `42.4.1`. Users are advised to upgrade. There are no known workarounds for this issue.

CVE-2022-41946

pgjdbc is an open source postgresql JDBC Driver. In affected versions a prepared statement using either `PreparedStatement.setText(int, InputStream)` or `PreparedStatemet.setBytea(int, InputStream)` will create a temporary file if the InputStream is larger than 2k. This will create a temporary file which is readable by other users on Unix like systems, but not MacOS. On Unix like systems, the system's temporary directory is shared between all users on that system. Because of this, when files and directories are written into this directory they are, by default, readable by other users on that same system. This vulnerability does not allow other users to overwrite the contents of these directories or files. This is purely an information disclosure vulnerability. Because certain JDK file system APIs were only added in JDK 1.7, this this fix is dependent upon the version of the JDK you are using. Java 1.7 and higher users: this vulnerability is fixed in 4.5.0. Java 1.6 and lower users: no patch is available. If you are unable to patch, or are stuck running on Java 1.6, specifying the java.io.tmpdir system environment variable to a directory that is exclusively owned by the executing user will mitigate this vulnerability.

CVE-2022-21724

pgjdbc is the offical PostgreSQL JDBC Driver. A security hole was found in the jdbc driver for postgresql database while doing security research. The system using the postgresql library will be attacked when attacker control the jdbc url or properties. pgjdbc instantiates plugin instances based on class names provided via `authenticationPluginClassName`, `sslhostnameverifier`, `socketFactory`, `sslfactory`, `sslpasswordcallback` connection properties. However, the driver did not verify if the class implements the expected interface before instantiating the class. This can lead to code execution loaded via arbitrary classes. Users using plugins are advised to upgrade. There are no known workarounds for this issue.

CVE-2018-10936

A weakness was found in postgresql-jdbc before version 42.2.5. It was possible to provide an SSL Factory and not check the host name if a host name verifier was not provided to the driver. This could lead to a condition where a man-in-the-middle attacker could masquerade as a trusted server by providing a certificate for the wrong host, as long as it was signed by a trusted CA.

CVE-2020-13692

PostgreSQL JDBC Driver (aka PgJDBC) before 42.2.13 allows XXE.

CVE-2022-36944

Scala 2.13.x before 2.13.9 has a Java deserialization chain in its JAR file. On its own, it cannot be exploited. There is only a risk in conjunction with Java object deserialization within an application. In such situations, it allows attackers to erase contents of arbitrary files, make network connections, or possibly run arbitrary code (specifically, Function0 functions) via a gadget chain.

CVE-2022-36944

Scala 2.13.x before 2.13.9 has a Java deserialization chain in its JAR file. On its own, it cannot be exploited. There is only a risk in conjunction with Java object deserialization within an application. In such situations, it allows attackers to erase contents of arbitrary files, make network connections, or possibly run arbitrary code (specifically, Function0 functions) via a gadget chain.

CVE-2023-22899

Zip4j through 2.11.2, as used in Threema and other products, does not always check the MAC when decrypting a ZIP archive.

CVE-2024-22233

In Spring Framework versions 6.0.15 and 6.1.2, it is possible for a user to provide specially crafted HTTP requests that may cause a denial-of-service (DoS) condition. Specifically, an application is vulnerable when all of the following are true: * the application uses Spring MVC * Spring Security 6.1.6+ or 6.2.1+ is on the classpath Typically, Spring Boot applications need the org.springframework.boot:spring-boot-starter-web and org.springframework.boot:spring-boot-starter-security dependencies to meet all conditions.

CVE-2024-35255

Azure Identity Libraries and Microsoft Authentication Library Elevation of Privilege Vulnerability

GHSA-xpw8-rcwv-8f8p

A client might overload the server by issue frequent RST frames. This can cause a massive amount of load on the remote system and so cause a DDOS attack.

CVE-2023-34462

Netty is an asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers & clients. The `SniHandler` can allocate up to 16MB of heap for each channel during the TLS handshake. When the handler or the channel does not have an idle timeout, it can be used to make a TCP server using the `SniHandler` to allocate 16MB of heap. The `SniHandler` class is a handler that waits for the TLS handshake to configure a `SslHandler` according to the indicated server name by the `ClientHello` record. For this matter it allocates a `ByteBuf` using the value defined in the `ClientHello` record. Normally the value of the packet should be smaller than the handshake packet but there are not checks done here and the way the code is written, it is possible to craft a packet that makes the `SslClientHelloHandler`. This vulnerability has been fixed in version 4.1.94.Final.

GHSA-58qw-p7qm-5rvh

There are no circumstances in a normally deployed Jetty server where potentially hostile XML is given to the XmlParser class without the attacker already having arbitrary access to the server. I.e. in order to exploit XmlParser the attacker would already have the ability to deploy and execute

hostile code. Specifically, Jetty has no protection against malicious web application and potentially hostile web applications should only be run on an isolated virtualisation.

CVE-2023-35887

Exposure of Sensitive Information to an Unauthorized Actor vulnerability in Apache Software Foundation Apache MINA. In SFTP servers implemented using Apache MINA SSHD that use a RootedFileSystem, logged users may be able to discover "exists/does not exist" information about items outside the rooted tree via paths including parent navigation ("..") beyond the root, or involving symlinks. This issue affects Apache MINA: from 1.0 before 2.10. Users are recommended to upgrade to 2.10

CVE-2023-34055

In Spring Boot versions 2.7.0 - 2.7.17, 3.0.0-3.0.12 and 3.1.0-3.1.5, it is possible for a user to provide specially crafted HTTP requests that may cause a denial-of-service (DoS) condition. Specifically, an application is vulnerable when all of the following are true: * the application uses Spring MVC or Spring WebFlux * org.springframework.boot:spring-boot-actuator is on the classpath

CVE-2022-39135

Apache Calcite 1.22.0 introduced the SQL operators EXISTS_NODE, EXTRACT_XML, XML_TRANSFORM and EXTRACT_VALUE do not restrict XML External Entity references in their configuration, making them vulnerable to a potential XML External Entity (XXE) attack. Therefore any client exposing these operators, typically by using Oracle dialect (the first three) or MySQL dialect (the last one), is affected by this vulnerability (the extent of it will depend on the user under which the application is running). From Apache Calcite 1.32.0 onwards, Document Type Declarations and XML External Entity resolution are disabled on the impacted operators.

GHSA-6g3j-p5g6-992f

A flaw was discovered in OpenSearch, affecting the _search API that allowed a specially crafted query string to cause a Stack Overflow and ultimately a Denial of Service. The issue was identified by Elastic Engineering and corresponds to security advisory ESA-2023-14 (CVE-2023-31419).

CVE-2023-50298

Exposure of Sensitive Information to an Unauthorized Actor vulnerability in Apache Solr.This issue affects Apache Solr: from 6.0.0 through 8.11.2, from 9.0.0 before 9.4.1. Solr Streaming Expressions allows users to extract data from other Solr Clouds, using a "zkHost" parameter. When original SolrCloud is setup to use ZooKeeper credentials and ACLs, they will be sent to whatever "zkHost" the user provides. An attacker could setup a server to mock ZooKeeper, that accepts ZooKeeper requests with credentials and ACLs and extracts the sensitive information, then send a streaming expression using the mock server's address in "zkHost". Streaming Expressions are exposed via the "/streaming" handler, with "read" permissions. Users are recommended to upgrade to version 8.11.3 or 9.4.1, which fix the issue. From these versions on, only zkHost values that have the same server address (regardless of chroot), will use the given ZooKeeper credentials and ACLs when connecting.

CVE-2021-41561

Improper Input Validation vulnerability in Parquet-MR of Apache Parquet allows an attacker to DoS by malicious Parquet files. This issue affects Apache Parquet-MR version 1.9.0 and later versions.

CVE-2023-52428

In Connect2id Nimbus JOSE+JWT before 9.37.2, an attacker can cause a denial of service (resource consumption) via a large JWE p2c header value (aka iteration count) for the PasswordBasedDecrypter (PBKDF2) component.

CVE-2020-13955

HttpUtils#getURLConnection method disables explicitly hostname verification for HTTPS connections making clients vulnerable to man-in-the-middle attacks. Calcite uses internally this method to connect with Druid and Splunk so information leakage may happen when using the respective Calcite adapters. The method itself is in a utility class so people may use it to create vulnerable HTTPS connections for other applications. From Apache Calcite 1.26 onwards, the hostname verification will be performed using the default JVM truststore.

CVE-2024-36114

Aircompressor is a library with ports of the Snappy, LZO, LZ4, and Zstandard compression algorithms to Java. All decompressor implementations of Aircompressor (LZ4, LZO, Snappy, Zstandard) can crash the JVM for certain input, and in some cases also leak the content of other memory of the Java process (which could contain sensitive information). When decompressing certain data, the decompressors try to access memory outside the bounds of the given byte arrays or byte buffers. Because Aircompressor uses the JDK class `sun.misc.Unsafe` to speed up memory access, no additional bounds checks are performed and this has similar security consequences as out-of-bounds access in C or C++, namely it can lead to non-deterministic behavior or crash the JVM. Users should update to Aircompressor 0.27 or newer where these issues have been fixed. When decompressing data from untrusted users, this can be exploited for a denial-of-service attack by crashing the JVM, or to leak other sensitive information from the Java process. There are no known workarounds for this issue.

CVE-2024-29857

An issue was discovered in ECCurve.java and ECCurve.cs in Bouncy Castle Java (BC Java) before 1.78, BC Java LTS before 2.73.6, BC-FJA before 1.0.2.5, and BC C# .Net before 2.3.1. Importing an EC certificate with crafted F2m parameters can lead to excessive CPU consumption during the evaluation of the curve parameters.

CVE-2024-32888

The Amazon JDBC Driver for Redshift is a Type 4 JDBC driver that provides database connectivity through the standard JDBC application program interfaces (APIs) available in the Java Platform, Enterprise Editions. Prior to version 2.1.0.28, SQL injection is possible when using the non-default connection property `preferQueryMode=simple` in combination with application code which has a vulnerable SQL that negates a parameter value. There is no vulnerability in the driver when using the default, extended query mode. Note that `preferQueryMode` is not a supported parameter in Redshift JDBC driver, and is inherited code from Postgres JDBC driver. Users who do not override default settings to utilize this unsupported query mode are not affected. This issue is patched in driver version 2.1.0.28. As a workaround, do not use the connection property `preferQueryMode=simple`. (NOTE: Those who do not explicitly specify a query mode use the default of extended query mode and are not affected by this issue.)

Behavioral Changes in Cloudera DataFlow for Data Hub 7.3.1

You can review the changes in certain features or functionalities of components that have resulted in a change in behavior from the previously released version to this version of Cloudera DataFlow for Data Hub 7.3.1.

Behavioral Changes in Flow Management

Review the list of Flow Management behavioral changes in Cloudera DataFlow for Data Hub 7.3.1.

Flow Management with NiFi 1

Secure communication between NiFi and ZooKeeper configured by default

If both ZooKeeper and NiFi services are secured, NiFi communication with ZooKeeper will be automatically configured as secured (TLS) using a new port, 2182. If you enforce TCP communication through a firewall and explicitly allow certain ports, you need to open them for port 2182.

If you do not want to use secure communication between ZooKeeper and NiFi, follow these steps to configure unsecured communication on port 2181:

1. Update the ZooKeeper connection string:
   a. In Cloudera Manager, navigate to NiFi Configuration .
   b. Set nifi.zookeeper.connect.string by replacing ${ZK_QUORUM} with the unsecure ZK QUORUM string, which has port 2181.

   To find your ZooKeeper quorum string from a NiFi node, run the following command as root:

   ```
   NIFI_PROC=$(ls -td /var/run/cloudera-scm-agent/process/NIFI/ | head -1);
    grep "Connect String" $NIFI_PROC/state-management.xml | cut -d\> -f2 |
   cut -d\< -f1; unset NIFI_PROC
   ```

   This command will provide your connect string. For example:

   ```
   host1:2181,host2:2181,host3:2181
   ```
2. Add a safety valve for staging/state-management.xml in Cloudera Manager with the following property:
   - Name: xml.state-management.cluster-provider.zk-provider.property.Connect String
   - Value: <YOUR ZOOKEEPER CONNECT STRING>
3. After upgrading to version 2.1.7, uncheck the nifi.zookeeper.client.secure option in Cloudera Manager.

ScriptedTransformRecord processor requires proper schema name attribute for record writer

NiFi-11523 introduced a fix that ensures the ScriptedTransformRecord processor uses the correct schema defined for the record writer. Previously, if the schema name attribute was set in the writer but not in the flow, it was ignored, defaulting to the reader schema. This behavior has been corrected, which may cause the processor to fail after upgrading if the schema name attribute is not set in the flow.

The failure is typically logged as:

```
org.apache.nifi.schema.access.SchemaNotFoundexception: ${schema.name} did no
t provide appropriate Schema Name
```

To prevent failures, ensure that the schema name attribute is properly configured in the flow or match it to the schema defined for the record reader for identical behavior.

Flow Management with NiFi 2

NiFi 2.0 introduces a lot of significant changes and enhancements, including some breaking changes for Flow Management clusters based on NiFi 2.X. It is important to familiarize yourself with the following points before migrating your existing flows.

Important: Currently, there is no upgrade path from NiFi 1-based Flow Management clusters to clusters with NiFi 2. You need to start new clusters using the NiFi 2 templates and migrate your existing flows to the new clusters. Note that these new NiFi 2-based clusters are available in Technical Preview. They are not production-ready and should not be used for critical workloads.

If you want to migrate a data flow, you need to export the process group as a JSON file from your NiFi 1.x cluster and import this JSON file into your NiFi 2.X cluster. Tooling to help with upgrades and automatically manage the breaking changes will be provided in an upcoming Flow Management release.

Java 21

Java 21 is the minimum Java version required with NiFi 2.0. This version is automatically installed and configured on new Data Hub clusters using NiFi 2.0.

Templates and XML flow definitions

The concept of templates in NiFi has been deprecated. Instead, versioning flows should be managed using the DataFlow Catalog and/or the NiFi Registry. It is highly recommended to handle any existing templates in your NiFi 1.x clusters by:

- Versioning the templates into the desired registry (DataFlow Catalog, NiFi Registry)
- Deleting the templates from NiFi process groups

Additionally, flow.xml.gz no longer exists, only flow.json.gz can be used in NiFi clusters for defining flows in the canvas.

Custom components / NARs

Although not certain, it is very likely that a custom NAR designed for NiFi 1 will not be successfully loaded into NiFi 2. If your NiFi setup includes custom components or NARs, it is a requirement to update your dependencies to align with NiFi 2. This entails making the necessary adjustments and rebuilding your NARs using Java 21.

Variables are removed in favor of parameters

Variables and the variable registry have been removed from NiFi. Only Parameter Contexts and parameters should be used going forward. In future releases, tools will be provided to help with the conversion of variables to parameters. In the meantime, this conversion should be done manually when migrating flows to NiFi 2. Any variables left will simply be ignored when loading the flow definition.

Event driven thread pool no longer exists

The event driven thread pool has been removed, leaving only the time driven thread pool available. Any components previously configured using the event driven scheduling strategy should be switched to the time driven scheduling strategy.

Removed languages in scripted components

In NiFi 2.0, support for certain languages in scripted components has been removed. The affected languages are: ECMAScript, Lua, Ruby, and Python. It is recommended to switch to Groovy or to leverage the new Python API feature for developing processors.

Removed components and replacement options

The following list contains the list of the components that have been removed between clusters based on NiFi 1.28 and clusters based on NiFi 2.0, along with the recommended alternatives where available.

- Processors
  - Base64EncodeContent => EncodeContent
  - CompareFuzzyHash => no replacement
  - ConsumeEWS => no replacement
  - ConsumeKafka_1_0 => ConsumeKafka_2_6
  - ConsumeKafka_2_0 => ConsumeKafka_2_6
  - ConsumeKafkaRecord_1_0 => ConsumeKafkaRecord_2_6
  - ConsumeKafkaRecord_2_0 => ConsumeKafkaRecord_2_6
  - ConvertAvroSchema => ConvertRecord
  - ConvertAvroToORC => no replacement
  - ConvertCSVToAvro => ConvertRecord
  - ConvertExcelToCSVProcessor => ConvertRecord with ExcelReader
  - ConvertJSONToAvro => ConvertRecord
  - CryptographicHashAttribute => UpdateAttribute
  - DeleteAzureBlobStorage => DeleteAzureBlobStorage_v12
  - DeleteRethinkDB => no replacement
  - EncryptContent => EncryptContentAge or EncryptContentPGP
  - ExecuteInfluxDBQuery => use Influx Data NARs for NiFi
  - ExtractCCDAAttributes => no replacement
  - FetchAzureBlobStorage => FetchAzureBlobStorage_v12
  - FetchElasticsearchHttp => GetElasticsearch
  - FuzzyHashContent => no replacement
  - GetAzureQueueStorage => GetAzureQueueStorage_v12
  - GetHTMLElement => no replacement
  - GetHTTP => InvokeHTTP
  - GetIgniteCache => no replacement
  - GetJMSQueue => ConsumeJMS
  - GetJMSTopic => ConsumeJMS
  - GetRethinkDB => no replacement
  - GetTCP => no replacement
  - GetTwitter => ConsumeTwitter
  - HashAttribute => CryptographicHashAttribute
  - HashContent => CryptographicHashContent
  - InferAvroSchema => ExtractRecordSchema
  - ListAzureBlobStorage => ListAzureBlobStorage_v12
  - ModifyHTMLElement => no replacement
  - PostHTTP => InvokeHTTP
  - PostSlack => PublishSlack
  - PublishKafka_1_0 => PublishKafka_2_6
  - PublishKafka_2_0 => PublishKafka_2_6
  - PublishKafkaRecord_1_0 => PublishKafkaRecord_2_6
  - PublishKafkaRecord_2_0 => PublishKafkaRecord_2_6
  - PutAzureBlobStorage => PutAzureBlobStorage_v12
  - PutAzureQueueStorage => PutAzureQueueStorage_v12
  - PutBigQueryBatch => PutBigQuery
  - PutBigQueryStreaming => PutBigQuery
  - PutElasticsearchHttp => PutElasticsearchJson
  - PutElasticsearchHttpRecord => PutElasticsearchRecord
  - PutHiveQL => PutClouderaHiveQL
  - PutHiveStreaming => PutClouderaHiveStreaming

    - PutHTMLElement => no replacement
    - PutIgniteCache => no replacement
    - PutInfluxDB => use Influx Data NARs for NiFi
    - PutJMS => PublishJMS
    - PutRethinkDB => no replacement
    - PutRiemann => no replacement
    - PutSlack => PublishSlack
    - QueryElasticsearchHttp => PaginatedJsonQueryElasticsearch
    - ScrollElasticsearchHttp => SearchElasticsearch
    - SelectHiveQL => SelectClouderaHiveQL
    - SpringContextProcessor => no replacement
    - StoreInKiteDataset => no replacement
    - UpdateHiveTable => UpdateClouderaHiveTable
- Controller services
    - ActionHandlerLookup => no replacement
    - AlertHandler => no replacement
    - AzureStorageCredentialsControllerService => AzureStorageCredentialsControllerService_v12
    - AzureStorageCredentialsControllerServiceLookup => AzureStorageCredentialsControllerServiceLookup_v12
    - AzureStorageEmulatorCredentialsControllerService => no replacement
    - EasyRulesEngineProvider => no replacement
    - EasyRulesEngineService => no replacement
    - ExpressionHandler => no replacement
    - GraphiteMetricReporterService => no replacement
    - GremlinClientService => no replacement
    - HBase_1_1_2_ClientMapCacheService => HBase_2_ClientMapCacheService
    - HBase_1_1_2_ClientService => HBase_2_ClientService
    - HBase_1_1_2_ListLookupService => no replacement
    - HBase_1_1_2_RecordLookupService => HBase_2_RecordLookupService
    - HiveConnectionPool => ClouderaHiveConnectionPool
    - HortonworksSchemaRegistry => ClouderaSchemaRegistry
    - KafkaRecordSink_1_0 => KafkaRecordSink_2_6
    - KafkaRecordSink_2_0 => KafkaRecordSink_2_6
    - LogHandler => no replacement
    - OAuth2TokenProviderImpl => StandardOauth2AccessTokenProvider
    - OpenCypherClientService => no replacement
    - RecordSinkHandler => no replacement
    - ScriptedActionHandler => no replacement
    - ScriptedRulesEngine => no replacement
- Reporting tasks
    - AmbariReportingTask => no replacement
    - MetricsEventReportingTask => no replacement
    - MetricsReportingTask => no replacement

- Components with new coordinates
  - InvokeGRPC => moved into nifi-cdf-grpc-nar
  - ListenGRPC => moved into nifi-cdf-grpc-nar
  - KerberosKeytabUserService => moved into nifi-kerberos-user-service-nar
  - KerberosPasswordUserService => moved into nifi-kerberos-user-service-nar
  - KerberosTicketCacheUserService => moved into nifi-kerberos-user-service-nar

  Tooling will be provided in upcoming releases to automatically handle these changes. Currently, two options are available:

  - Manually edit the flow.json.gz file to update the coordinates of the impacted components.
  - Make the changes after the flow is imported in NiFi 2.0 by replacing the ghost components with the new implementations for each instance of the components listed above.
- Pulsar components

  All Pulsar components have been temporarily removed. They will be reintroduced in an upcoming release. In the meantime, you can download the NARs from a public Maven repository and deploy them as custom NARs.

  - nifi-pulsar-nar
  - nifi-pulsar-client-service-nar

Behavioral Changes in Streams Messaging

Review the list of Streams Messaging behavioral changes in Cloudera DataFlow for Data Hub 7.3.1.

Kafka

There are no behavioral changes for Kafka in Cloudera DataFlow for Data Hub 7.3.1.

Schema Registry

There are no behavioral changes for Schema Registry in Cloudera DataFlow for Data Hub 7.3.1.

Streams Messaging Manager

There are no behavioral changes for Streams Messaging Manager in Cloudera DataFlow for Data Hub 7.3.1.

Streams Replication Manager

There are no behavioral changes for Streams Replication Manager in Cloudera DataFlow for Data Hub 7.3.1.

Cruise Control

There are no behavioral changes for Cruise Control in Cloudera DataFlow for Data Hub 7.3.1.

Behavioral Changes in Cloudera Streaming Analytics

Review the list of Cloudera Streaming Analytics behavioral changes in Cloudera DataFlow for Cloudera Data Hub 7.3.1.

Cloudera SQL Stream Builder

Summary:

CSA-5068 - Page refresh is not required to update result samples.

Previous behavior:

Sample ids could become invalid if another user restarted the job while polling in the UI.

New behavior:

Sample ids are updated before and during polling.

Summary:

CSA-5238 - Yarn is set as default token renewer automatically based on configuration.

Previous behavior:

Configuration check wasn't performed and yarn wasn't set as the default renewer.

New behavior:

Yarn is set as the default token renewer if the deployment target is yarn-session or yarn-per-job.

Summary:

CSA-5258 - Extended validation for dynamic MV parameter names.

Previous behavior:

Dynamic pattern validation was restrictive which could result in errors when querying MV.

New behavior:

Validation pattern is extended to [A-Za-z_\-0-9\.\~]+

Summary:

CSA-5214 - Add option to Cloudera SQL Stream Builder service to customize default truststore

Previous behavior:

Default Kafka TrustStore configurations could only be customized in Flink configurations.

New behavior:

Customizing default Kafka TrustStore configurations was added to Streaming SQL Console. Kafka TrustStore can be configured during adding Kafka as a Data Source on the UI.

Summary:

CSA-5199 - API change: the Cloudera registry catalog type registry is now named cloudera-registry

Previous behavior:

When creating a Schema Registry data source catalog type property had to be registry

New behavior:

Catalog type property now has to be cloudera-registry when creating a Schema Registry data source, but registry is backward compatible.

Summary:

CSA-5306 - Cloudera SQL Stream Builder API does not validate data sources before saving

Previous behavior:

There was no validation before saving a data source via API. If the user wanted to make sure to save a valid data source they had to use the validate endpoint before saving. When creating a data source on SSB UI, the user had the option to save an invalid data source.

New behavior:

Saving the data source is only allowed if the data source is valid. SSB API validates the data source before saving and on the SSB UI Create/Save button is only active if the validation is successful.

Cloudera DataFlow for Data Hub 7.3.1

Streaming Analytics

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Streaming Analytics in Cloudera.....4

What is Apache Flink?..... 5

Core features of Flink..... 6

What is SQL Stream Builder?.....9

Key features of SSB.....10

SQL Stream Builder architecture.....11

Streaming Analytics in Cloudera

Cloudera Streaming Analytics (CSA) offers real-time stream processing and streaming analytics powered by Apache Flink. Flink implemented on CDP provides a flexible streaming solution with low latency that can scale to large throughput and state. In addition to Flink, CSA includes SQL Stream Builder (SSB) to offer data analytical experience using SQL queries on your data streams.

Key features of Cloudera Streaming Analytics

Apache Flink

CSA is powered by Apache Flink that offers a framework for real-time stream processing and streaming analytics. CSA offers the features and functionalities of the upstream Apache Flink integrated on CDP Public Cloud.

SQL Stream Builder

SQL Stream Builder is a job management interface to compose and run Continuous Streaming SQL on streams using Apache Flink as an engine, as well as to create REST APIs for the results.

Cloudera Platform

Implementing Flink on the Cloudera Platform allows you to easily integrate with Runtime components, and have all the advantages of cluster and service management with Cloudera Manager.

Streaming Platform

For streaming analytics, CSA fits into a complete streaming platform augmented by Apache Kafka, Schema Registry, Streams Messaging Manager in the Cloudera Runtime stack.

Supported Connectors

CSA offers a set of connectors for Flink and SSB from which you can choose from based on your requirements. Kafka, HBase, HDFS, Kudu and Hive connectors are available for Flink. Kafka, HDFS/S3, JDBC and a set of the Debezium CDC connectors are available for SSB. Other than the connectors, SSB also supports Schema Registry, Hive and Kudu as catalogs.

Monitoring Solutions

In CDP Public Cloud, Streams Messaging components provide tools that support the operational needs of CSA. For example, when Kafka is used as a connector, you can use Kafka Metrics Reporter and Streams Messaging Manager (SMM) for Kafka management and alerting of Kafka actions. Beside SMM, you can use the Flink Dashboard to monitor your Flink and SSB jobs.

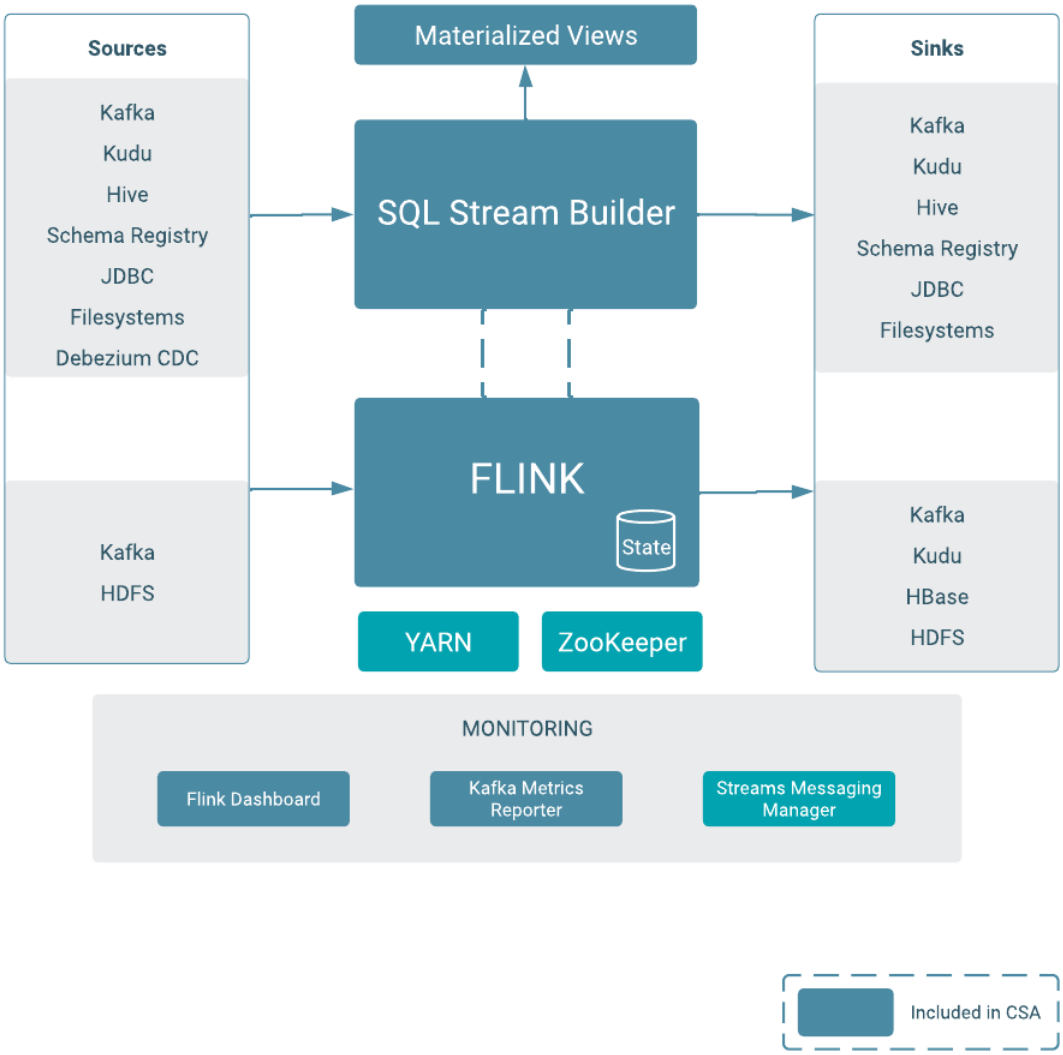

What is Apache Flink?

Flink is a distributed processing engine and a scalable data analytics framework. You can use Flink to process data streams at a large scale and to deliver real-time analytical insights about your processed data with your streaming application.

Flink is designed to run in all common cluster environments, perform computations at in-memory speed and at any scale. Furthermore, Flink provides communication, fault tolerance, and data distribution for distributed computations over data streams. A large variety of enterprises choose Flink as a stream processing platform due to its ability to handle scale, stateful stream processing, and event time.

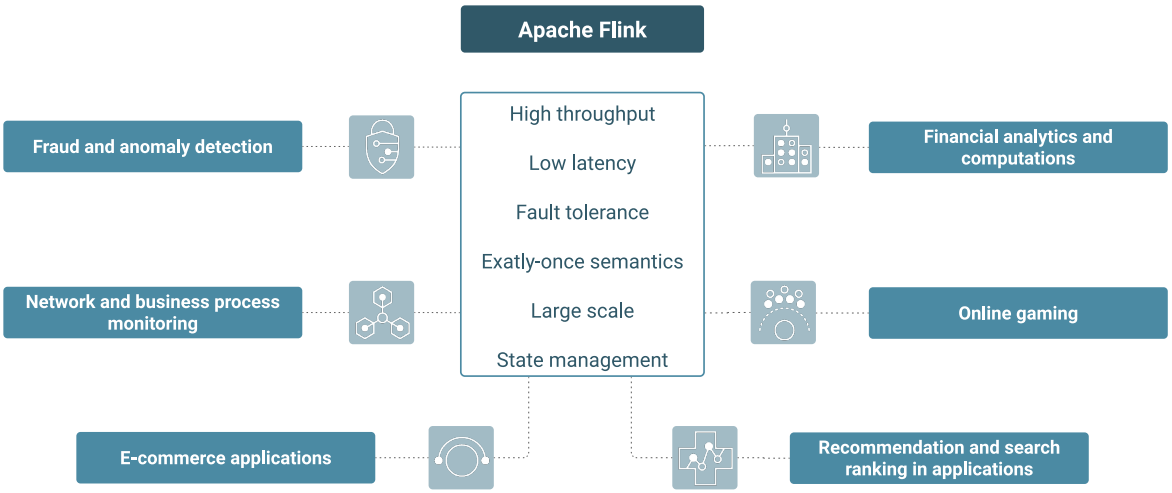

Core features of Flink

Learn more about the specific details of Flink architecture, the DataStream API, how Flink handles event time and watermarks, and how state management works in Flink.

Architecture

The two main components for the task execution process are the Job Manager and Task Manager. The Job Manager on a master node starts a worker node. On a worker node the Task Managers are responsible for running tasks and the Task Manager can also run more than one task at the same time. The resource management for the tasks are completed by the Job manager in Flink. In a Flink cluster, Flink jobs are executed as YARN applications. HDFS is used to store recovery and log data, while ZooKeeper is used for high availability coordination for jobs.

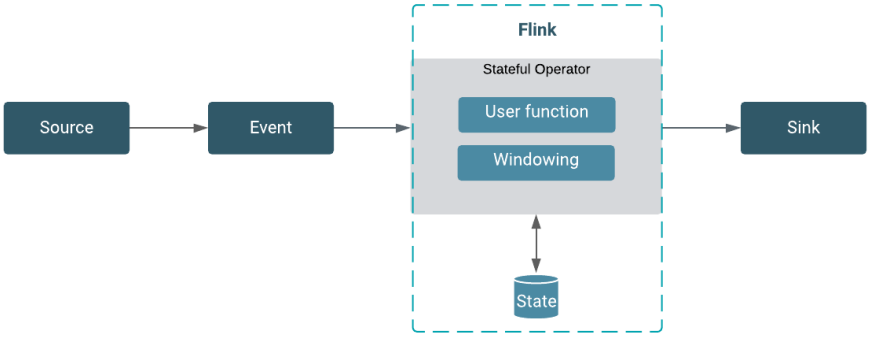

DataStream API

The DataStream API is used as the core API to develop Flink streaming applications using Java or Scala programming languages. The DataStream API provides the core building blocks of the Flink streaming application: the datastream and the transformation on it. In a Flink program, the incoming data streams from a source are transformed by a defined operation which results in one or more output streams to the sink.

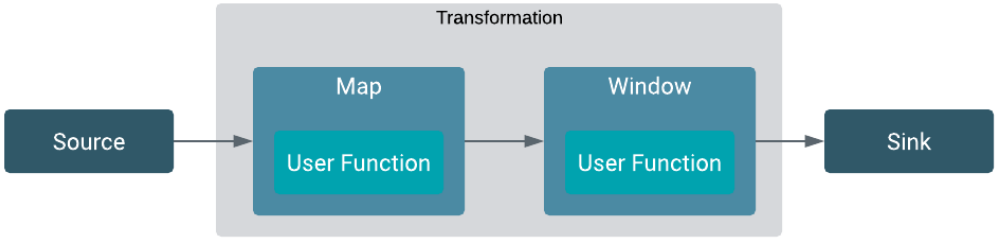

Operators

Operators transform one or more DataStreams into a new DataStream. Programs can combine multiple transformations into sophisticated data flow topologies. Other than the standard transformations like map, filter, aggregation, you can also create windows and join windows within the Flink operators. On a dataflow one or more operations can be defined which can be processed in parallel and independently to each other. With windowing functions, different computations can be applied to different streams in the defined time window to further maintain the processing of events. The following image illustrates the parallel structure of dataflows.

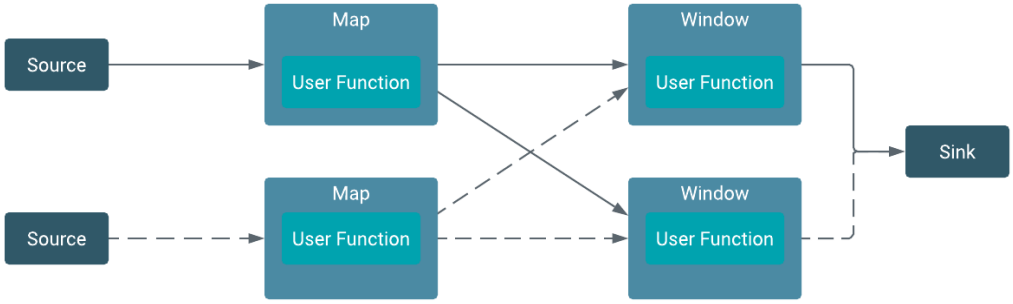

State and state backend

Stateful applications process dataflows with operations that store and access information across multiple events. You can use Flink to store the state of your application locally in state backends that guarantee lower latency when accessing your processed data. You can also create checkpoints and savepoints to have a fault-tolerant backup of your streaming application on a durable storage.

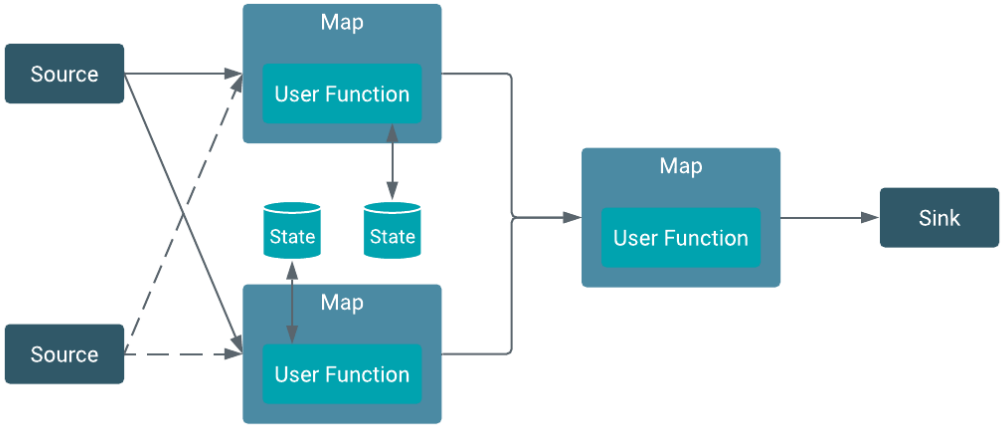

Event time and watermark

In time-sensitive cases where the application uses alerting or triggering functions, it is important to distinguish between event time and processing time. To make the designing of applications easier,

you can create your Flink application either based on the time when the event is created or when it is processed by the operator.

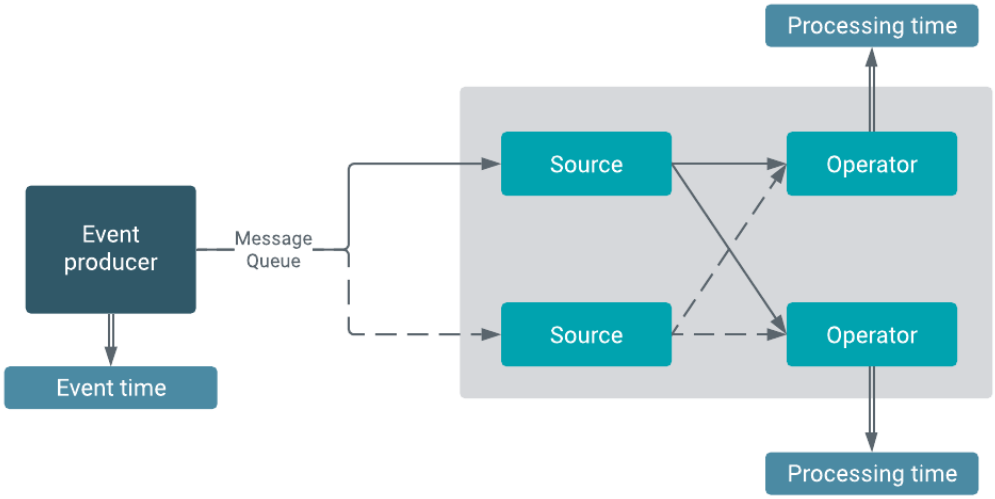

With only the event time, it is not clear when the events are processed in the application. To track the time for an event time based application, watermark can be used.

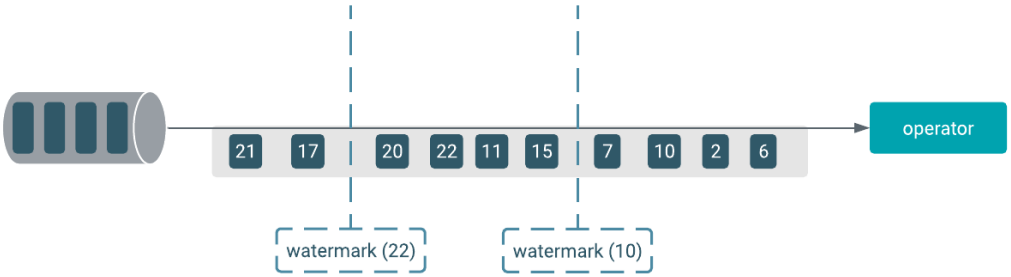

Checkpoints and savepoints

Checkpoints and savepoints can be created to make the Flink application fault tolerant throughout the whole pipeline. Flink contains a fault tolerance mechanism that creates snapshots of the data stream continuously. The snapshot includes not only the dataflow, but the state attached to it. In case of failure, the latest snapshot is chosen and the system recovers from that checkpoint. This guarantees that the result of the computation can always be consistently restored. While checkpoints are created and managed by Flink, savepoints are controlled by the user. A savepoint can be described as a backup from the executed process.

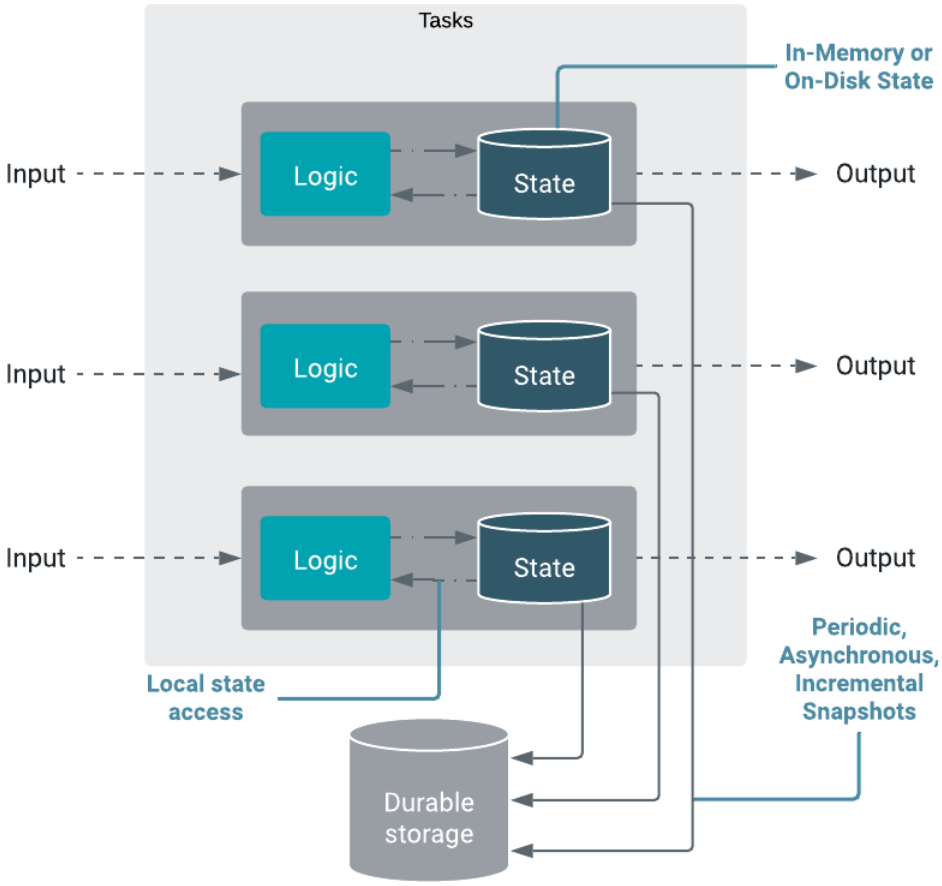

Related Information

Flink application structure
Configuring RocksDB state backend
Enabling checkpoints for Flink applications
Enabling savepoints for Flink applications

What is SQL Stream Builder?

Cloudera Streaming Analytics offers an easy to use and interactive SQL Stream Builder as a service to create queries on streams of data through SQL.

The SQL Stream Builder (SSB) is a comprehensive interactive user interface for creating stateful stream processing jobs using SQL. By using SQL, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. SSB is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

What is Continuous SQL?

SSB runs Structured Query Language (SQL) statements continuously, this is called Continuous SQL or Streaming SQL. Continuous SQL can run against both bounded and unbounded streams of data. The results are sent to a sink of some type, and can be connected to other applications through a Materialized View interface. Compared to traditional SQL, in Continuous SQL the data has a start, but no end. This means that queries continuously process results. When

you define your job in SQL, the SQL statement is interpreted and validated against a schema. After the statement is executed, the results that match the criteria are continuously returned.

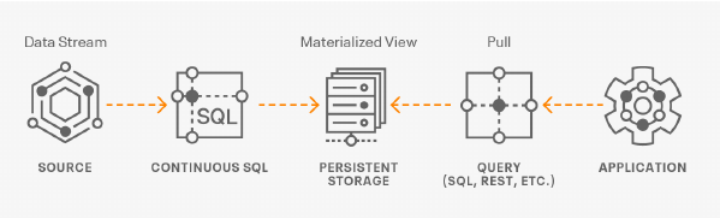

Integration with Flink

SSB runs in an interactive fashion where you can quickly see the results of your query and iterate on your SQL syntax. The executed SQL queries run as jobs on the Flink cluster, operating on boundless streams of data until canceled. This allows you to author, launch, and monitor stream processing jobs within SSB as every SQL query is a Flink job. You can use Flink and submit Flink jobs without using Java, as SSB automatically builds and runs the Flink job in the background.

As a result of Flink integration, you are able to use the basic functionalities offered by Flink. You can choose exactly once processing, process your data stream using event time, save your jobs with savepoints, and use Flink SQL to create tables and use connectors based on your requirements. As a result of the various connectors, you are able to enrich your streaming data with data from slowly changing connectors.

Key features of SSB

SQL Stream Builder (SSB) within Cloudera supports out-of-box integration with Flink. Using Flink SQL, you can create tables directly from the Streaming SQL Console window or built-in templates. For integration with Business Intelligence tools you can create Materialized Views.

Flink SQL

SQL Stream Builder allows you to use Data Definition Language (DDL), Data Manipulation Language (DML) and Query Language directly from the Streaming SQL Console.

For more information about the supported Flink SQL statements, see the Flink SQL statements section.

Change Data Capture

SQL Stream Builder supports PostgreSQL, Oracle, MySQL, Db2 and SQL Server as Debezium connectors using Flink SQL. With Change Data Capture (CDC), you can capture changes in your databases and update your applications with the newly added data.

For more information about Debezium, see the official documentation.

Session Cluster

When you start SSB, a session cluster is deployed to share and maintain resources. The submitted SQL jobs are executed as Flink jobs in the same session cluster that share a Job Manager. The properties for the session cluster can be viewed using the Streaming SQL Console, and can be modified with SET statements in the SQL window.

Built-in Templates

The Build-in Templates in SSB allows you to quickly and simply create tables for your SQL queries. You only need to provide the connection and job specific information to the template to use it in SSB.

Streaming SQL Console

SSB comes with an interactive user interface that allows you to easily create, and manage your SQL jobs in one place. It allows you to create and iterate on SQL statements with robust tooling and capabilities. Query parsing is logged to the console, and results are sampled back to the interface to help with iterating on the SQL statement as required.

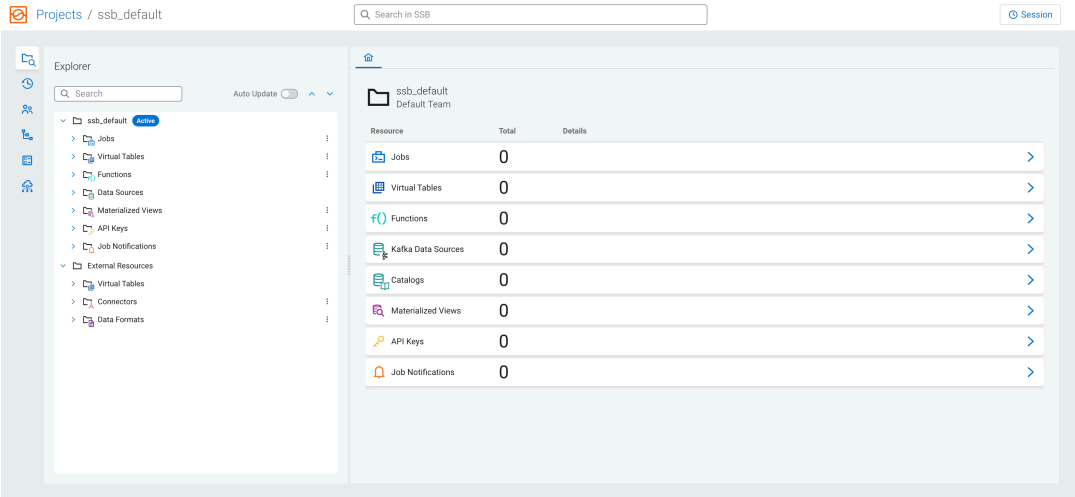

Materialized Views

SSB has the capability to materialize results from a Streaming SQL query to a persistent view of the data that can be read through REST and over the PG wire protocol. Applications can use this mechanism to query streams of data in a way of high performance without deploying additional database systems. Materialized Views are built into the SQL Stream Builder service, and require no configuration or maintenance. The Materialized Views act like a special kind of sink, and can even be used in place of a sink. They require no indexing, storage allocation, or specific management.

Input Transform

In case you are not aware of the incoming data structure or raw data is being collected from for example sensors, you can use the Input Transform to clean up and organize the incoming data before querying. Input transforms also allow access to Kafka header metadata directly in the query itself. Input transforms are written in Javascript and compiled to Java bytecode deployed with the Flink jar.

User-defined Functions

You can create customized and complex SQL queries by using User-defined Functions to enrich your data, apply computations or a business logic on it. User defined functions are written in Javascript or Java language.

Related Information

Integration with Flink

Flink SQL statements

SQL Stream Builder architecture

The SQL Stream Builder (SSB) service is integrated on Cloudera Data Platform (CDP) and connected to Flink and its services. The SSB architecture includes SQL Stream Engine and Materialized View Engine. These main components within SSB are responsible for executing jobs, populating topics, creating metadata and querying data that happens in the background.

SSB consists of the following main components:

- SQL Stream Engine
- Materialized View Engine

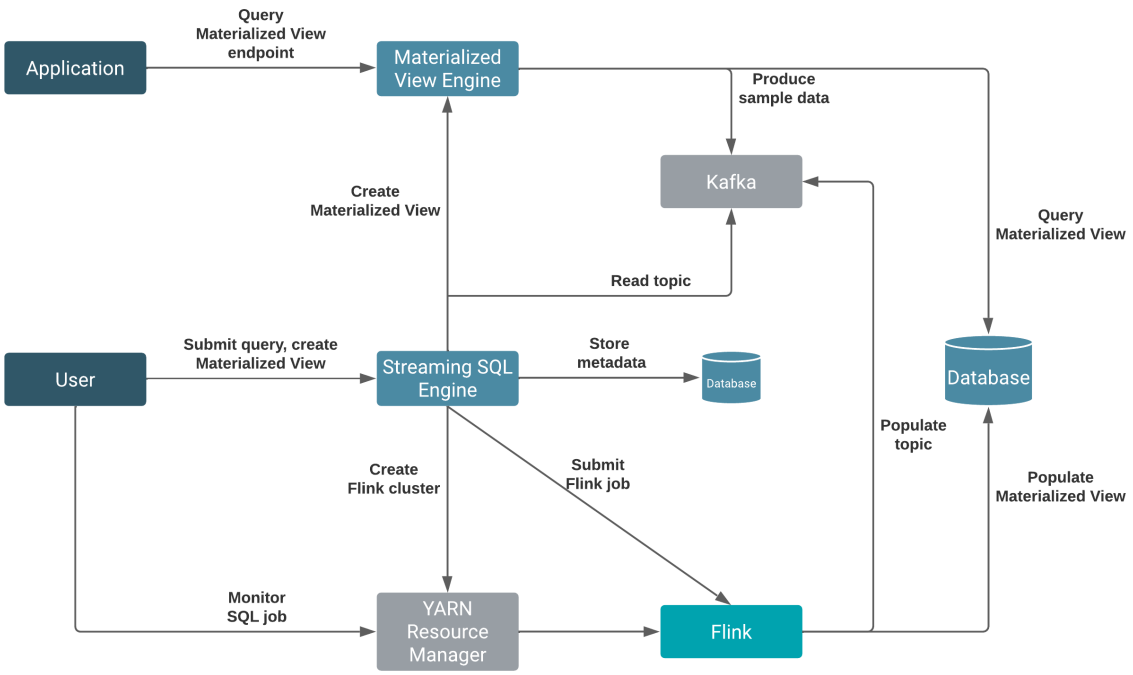

The primary point of user interaction for SQL Stream Builder is the Streaming SQL Console User Interface running on the Streaming SQL Engine. When a query is submitted on the Streaming SQL Console, the Streaming SQL Engine automatically creates a Flink job in the background on the cluster. SSB also requires a Kafka service on the same cluster. This mandatory Kafka service is used to automatically populate topics for the websocket output. The websocket output is needed for sampling data to the Console, and when no table is added to output the results of the SQL query.

When a Materialized View query is submitted, Flink generates the data to the Materialized View database from which the Materialized View Engine queries the required data.

Database management in SSB

SSB uses databases in the following cases:

- To store metadata of SQL jobs
- To store data for creating Materialized Views
- As a connector for Flink SQL

In CDP Public Cloud, PostgreSQL is supported as a default database for SQL Stream Builder and the Materialized View database.

Cloudera DataFlow for Data Hub 7.3.1

Moving data from Cloudera Private Cloud Base to Public Cloud with NiFi site-to-site

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Moving data from Private Cloud to Public Cloud with NiFi site-to-site..... 4
Understanding the use case..... 4
Preparing your clusters..... 4
Setting up your network configuration..... 5
Configuring your truststores..... 5
Defining your CDP Public Cloud data flow..... 7
Configuring Ranger policies for site-to-site communication..... 8
Defining your Cloudera Private Cloud Base data flow..... 10

Moving data from Private Cloud to Public Cloud with NiFi site-to-site

You can use Apache NiFi's site-to-site functionality to design a flow that moves data from your CDP Private Cloud Base environment to CDP Public Cloud to ensure scalability and load balancing.

Understanding the use case

You can use the Apache NiFi site-to-site functionality to move data between a Public Cloud and a Cloudera Private Cloud Base environment. To do this, set up a cluster in each environment, prepare your network and truststore configurations, and then define your Cloudera Private Cloud Base and CDP Public Cloud data flows and Apache Ranger configuration for site-to-site functionality.

Moving data between CDP Public Cloud and Cloudera Private Cloud Base clusters is a common use case when there is a need for a lot of temporary compute resources that can be quickly provisioned in the cloud.

Imagine you have a large dataset on-premises and you wish to perform heavy computations on the dataset. You can use the following workflow to design a data flow that:

- Moves the dataset from your Cloudera Private Cloud Base environment to your CDP Public Cloud environment
- Pushes the data to the appropriate destination
- Triggers the workload that processes the data while leveraging the auto-scaling capabilities that CDP Public Cloud provides
- Returns the results in your Cloudera Private Cloud Base environment

All of this is powered by Cloudera Private Cloud Base and CDP Public Cloud distributions, while ensuring consistent security policies at a fine-grained level with Apache Ranger, and data management and data lineage with Apache Atlas across the environments.

Preparing your clusters

The first step in preparing to move data from a Cloudera Private Cloud Base cluster to a CDP Public Cloud cluster is to ensure that you have each cluster set up correctly.

Requirements for your CDP Private Cloud Base cluster:

- CFM running on Cloudera Private Cloud Base
- Three-node NiFi compute cluster, secured with AutoTLS and configured with Apache Ranger

For details on deploying your CFM cluster on Cloudera Private Cloud Base, see *CFM Deployment to Cloudera Private Cloud Base*.

Requirements for your CDP Public Cloud Flow Management cluster:

- Flow Management clusters running on CDP Public Cloud
- Three-node NiFi compute cluster, secured with AutoTLS and configured with Apache Ranger

For details on deploying your CFM cluster on CDP Public Cloud, see *Setting up your Flow Management cluster*.

Related Information

CFM Deployment to CDP Private Cloud Base

Setting up your Flow Management cluster

Setting up your network configuration

You can use NiFi's site-to-site capabilities over a RAW TCP or over an HTTP network configuration. For this use case, you must configure site-to-site using HTTP over TLS. This has the advantage of using the NiFi port, which is also used to access the NiFi UI and APIs.

For the purpose of this use case, set up your site-to-site network configurations with the following assumptions:

- You are not using site-to-site through any proxy configuration
- You have a direct connection on port 8443 between NiFi nodes on your Cloudera Private Cloud Base and CDP Private Cloud clusters

Set up your network configuration according to your architecture and requirements. For more information, see your Cloud provider documentation.

In this use case, NiFi on Cloudera Private Cloud Base is responsible for initiating the site-to-site connection between the two environments to push and pull data to and from the NiFi cluster in Public Cloud. The NiFi nodes in CDP Public Cloud must be reachable on port 8443 from the NiFi nodes in Cloudera Private Cloud Base, but not necessarily the other way around.

Tip:

The site-to-site connection is bi-directional and depending on the cluster initiating the site-to-site connection, you will be in a push or pull model.

Configuring your truststores

You must configure your truststores so that each cluster is aware of and trusts the other cluster, to support the two-way TLS that is used to initiate the site-to-site communication between clusters. To do this, you need to download and merge the truststores for NiFi in Cloudera Private Cloud Base and CDP Public Cloud.

Before you begin

- You have set up a Cloudera Private Cloud Base and CDP Public Cloud cluster, and have the necessary network configurations established.
- You have the necessary administrative permissions to manipulate the truststore files. You require root access, and this is typically done by an Environment Administrator.
- You have the passwords for the Java Keystore (JKS) files available.

Procedure

1. Download the truststore from the clusters.
   a) Create a temporary directory in which you can edit the truststore files.

   ```
   $ mkdir s2s-temp && cd s2s-temp
   ```

   b) Download the JKS truststore file for CA used by NiFi in your Cloudera Private Cloud Base cluster.

   ```
   $ scp -i <key>
   root@<nifi_node>:/var/lib/cloudera-scm-agent/agent-cert/cm-auto-global
   _truststore.jks
   privatecloud_cm-auto-global_truststore.jks
   ```

   c) Download the JKS truststore file for CA used by NiFi in your CDP Public Cloud cluster.

   ```
   $ scp -i <key>
   ```

```
cloudbreak@<nifi_node_public_cloud>:/var/lib/cloudera-scm-agent/agent-
cert/cm-auto-global_truststore.jks
publiccloud_cm-auto-global_truststore.jks
```

2. Merge the truststores.

   a. Make a copy of the Cloudera Private Cloud Base JKS:

```
$ cp privatecloud_cm-auto-global_truststore.jks
privatecloud_cm-auto-global_truststore.jks.bak
```

   b. Merge the CDP Public Cloud JKS into the Cloudera Private Cloud Base JKS and rename the entries alias to prevent conflict:

```
$ keytool
-importkeystore
-srckeystore publiccloud_cm-auto-global_truststore.jks
-destkeystore privatecloud_cm-auto-global_truststore.jks
```

   The result will be similar to:

```
Importing keystore publiccloud_cm-auto-global_truststore.jks to private
cloud_cm-auto-global_truststore.jks...
Enter destination keystore password:
Enter source keystore password:
Entry for alias imported-ca-b379e6601f5ecfbbee2fefc4eb2efd4a successfull
y imported.
[...]
Entry for alias imported-ca-5945bad341623ae14991e09ffe851725 successfu
lly imported.
Entry for alias cmrootca-1 successfully imported.
Existing entry alias cmrootca-0 exists, overwrite? [no]:  no
Enter new alias name (RETURN to cancel import for this entry):  cmrootc
a-1-bis
Entry for alias cmrootca-0 successfully imported.
Entry for alias imported-ca-10c56ecc972802e53d1b7287ac2d1c6c successfu
lly imported.
[...]
Entry for alias imported-ca-840644351dd523125493ff4c28e694f7 successfull
y imported.
Import command completed:  140 entries successfully imported, 0 entries
failed or cancelled
```

   c. Use the copy you made to merge the Cloudera Private Cloud Base JKS into the CDP Public Cloud JKS and rename the entries alias to prevent conflict:

```
$ keytool
-importkeystore
-srckeystore privatecloud_cm-auto-global_truststore.jks.bak
-destkeystore publiccloud_cm-auto-global_truststore.jks
```

   The result will be similar to:

```
Importing keystore privatecloud_cm-auto-global_truststore.jks.bak to pu
bliccloud_cm-auto-global_truststore.jks...
Enter destination keystore password:
Enter source keystore password:
Existing entry alias cmrootca-0 exists, overwrite? [no]:  no
```

```
Enter new alias name (RETURN to cancel import for this entry):  cmrootc
a-0-bis
Entry for alias cmrootca-0 successfully imported.
Import command completed:  1 entries successfully imported, 0 entries
failed or cancelled
```

3. Deploy the truststores.

   Deploy the modified Cloudera Private Cloud Base and CDP Public Cloud JKS files on each NiFi node of the respective clusters.

   Note:

   Do not change the permissions and owners of the file (chmod/chown).

4. Restart your Cloudera Private Cloud Base and CDP Public Cloud clusters.

What to do next

After you have configured your truststores, proceed by defining your data flow in your CDP Public Cloud cluster.

Defining your CDP Public Cloud data flow

To move data between cloud environments using NiFi site-to-site communication, you require a data flow in CDP Public Cloud that can receive data from the Cloudera Private Cloud Base data flow. To create this data flow, configure a process group, and both an input and output port.

Before you begin

You have prepared you clusters, set up your network configurations, and configured your truststores.

Procedure

1. From your CDP Public Cloud NiFi cluster, create a Process Group to perform the operations you want to complete on the data received from and returned to the Cloudera Private Cloud Base cluster.
2. Drag an Input Port onto the NiFi canvas.

   You must use this port for receiving data from NiFi's Cloudera Private Cloud Base cluster.
3. Drag an Output Port onto the NiFi canvas.

   You must use this port to make data available for download to the Cloudera Private Cloud Base cluster.
4. Connect your CDP Public Cloud data flow components.

   Ensure that you have specified the public endpoints of your NiFi nodes in the CDP Public Cloud cluster.
5. Start your data flow and ensure that both the input and output ports are running.

Example

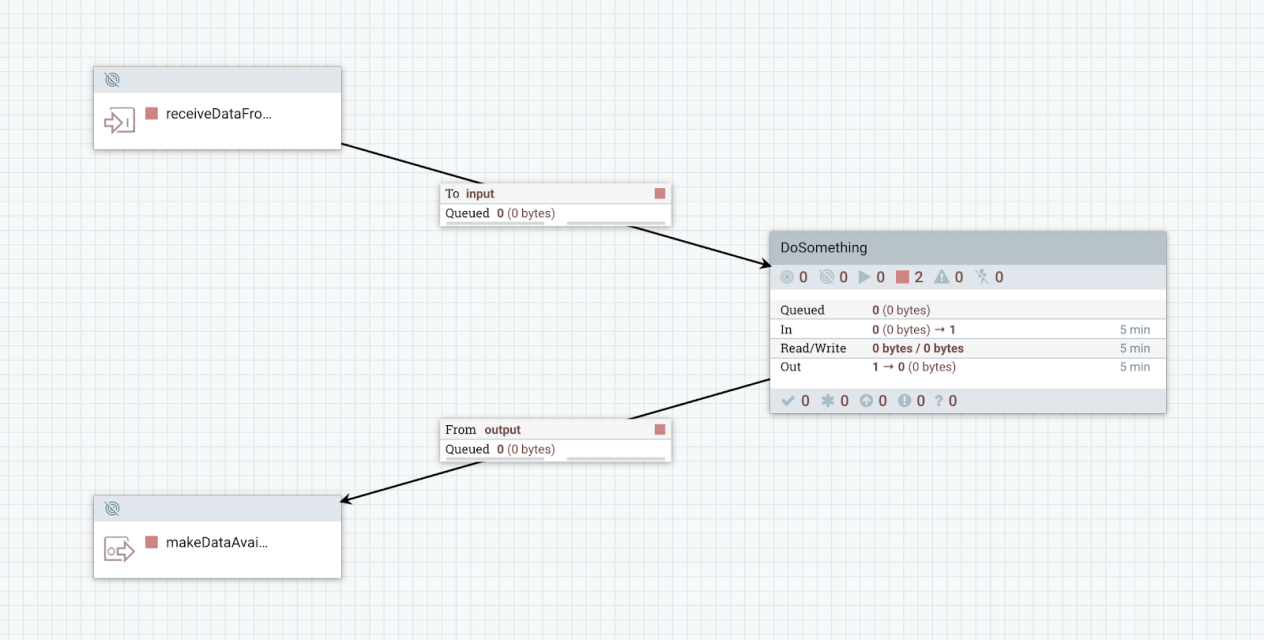

What to do next

When you have completed your CDP Public Cloud data flow, proceed by configuring Apache Ranger to allow NiFi's site-to-site transmission.

Configuring Ranger policies for site-to-site communication

To allow NiFi's site-to-site communication between CDP Public Cloud and Cloudera Private Cloud Base clusters, you need to configure Ranger authorization between the two clusters. To do this, create Ranger users in your CDP Public Cloud cluster that correspond to the Cloudera Private Cloud Base NiFi nodes. Then create a new Ranger policy with site-to-site resources configured, and assign your Cloudera Private Cloud Base NiFi node users to the policy.

Before you begin

- You have defined your CDP Public Cloud data flow.
- You have a list of your FQDN Cloudera Private Cloud Base host names. You need the host names to create the Ranger policies in CDP Public Cloud.

Procedure

1. In you CDP Public Cloud environment, launch the Ranger UI, click Settings Users/Groups/Roles User Create to add the users corresponding to the nodes of the Cloudera Private Cloud Base cluster.

2. Click User Create to create one user per NiFi node running your Cloudera Private Cloud Base environment.

   You create this user to make Ranger aware of the Cloudera Private Cloud Base nodes, so that you can create policies by including them. Because this user is not used to authenticate on the Ranger UI, the password can be random.

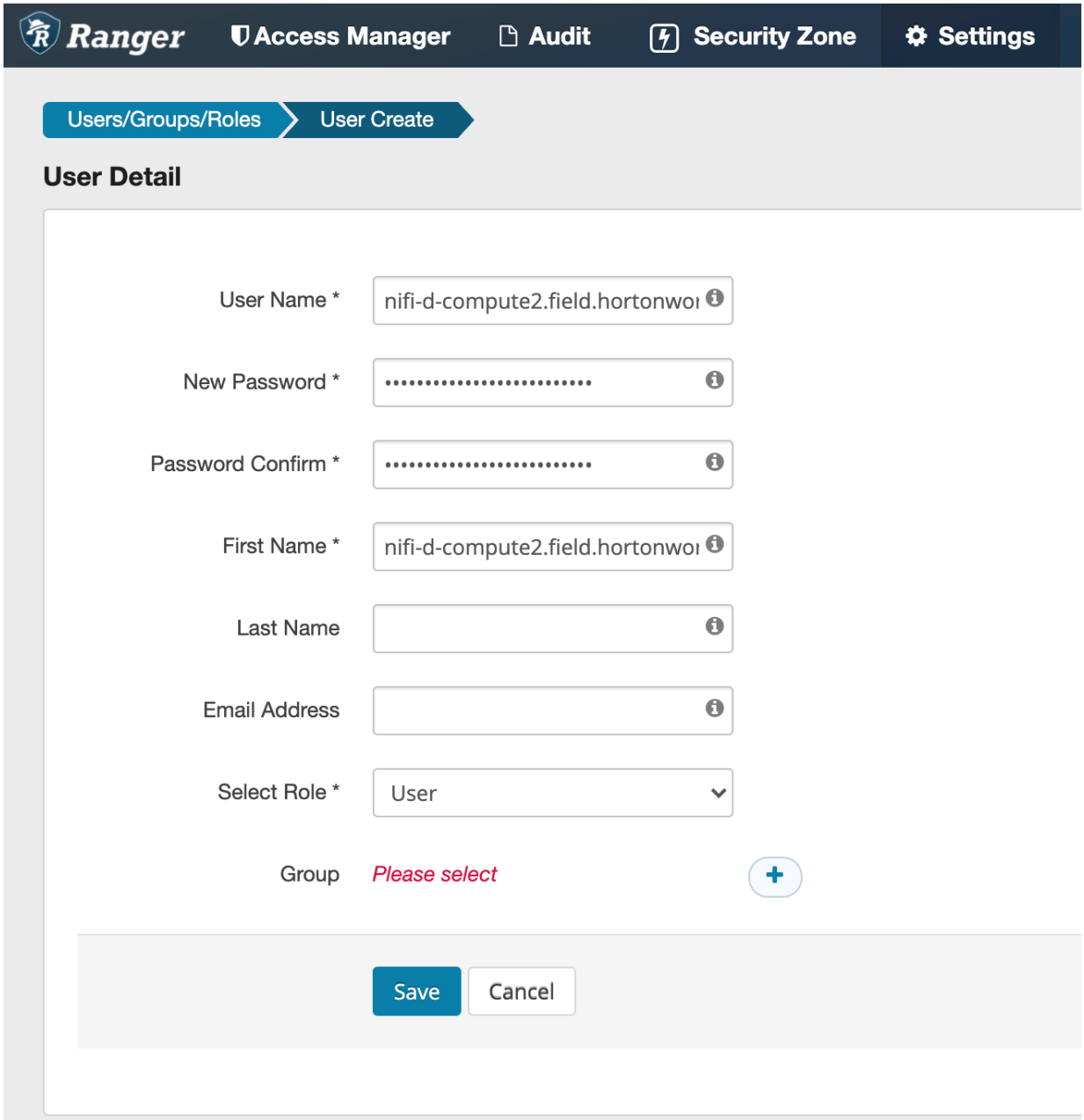

3. Create a new policy in the NiFi Service in Ranger.

   You need to enter the following NiFi Resources:

   - /site-to-site
   - /data-transfer/input-ports/<ID of the Input Port>
   - /data-transfer/output-ports/<ID of the Output Port>

   Note:

   You can retrieve the input and output port IDs by right-clicking the component and reviewing the configuration view.

4. Add the Cloudera Private Cloud Base users you created in Step 2, and assign Read and Write permissions:

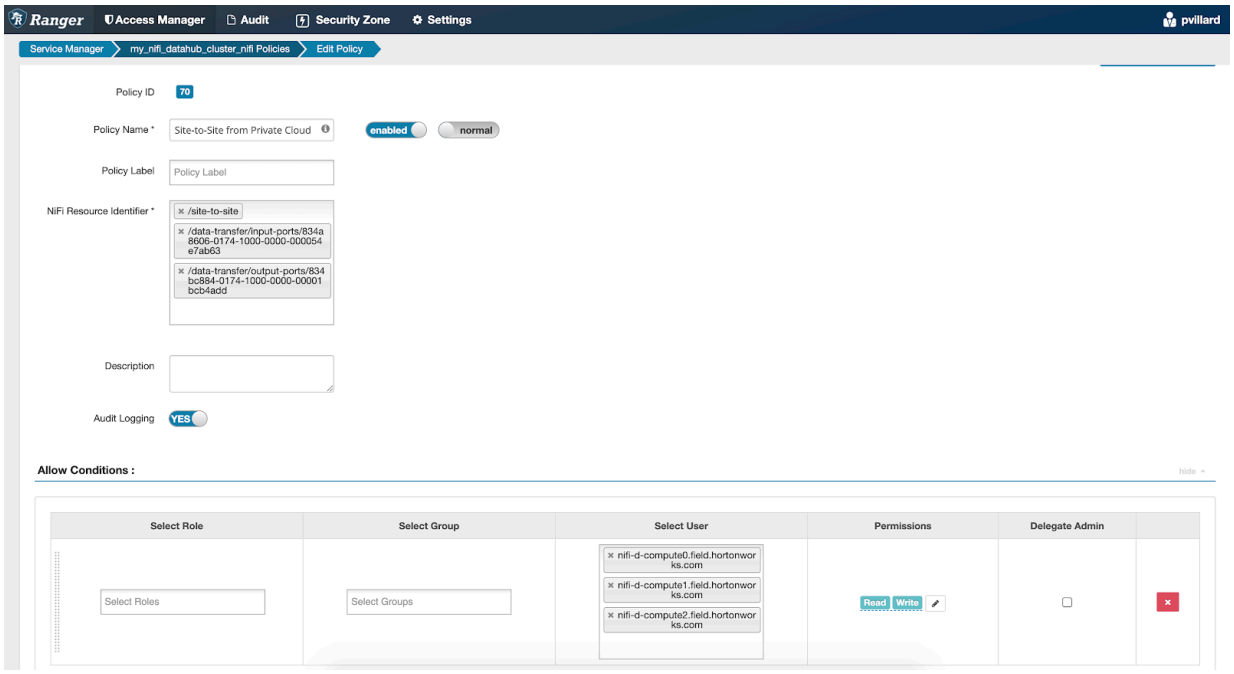

Results

Your policies are now listed.

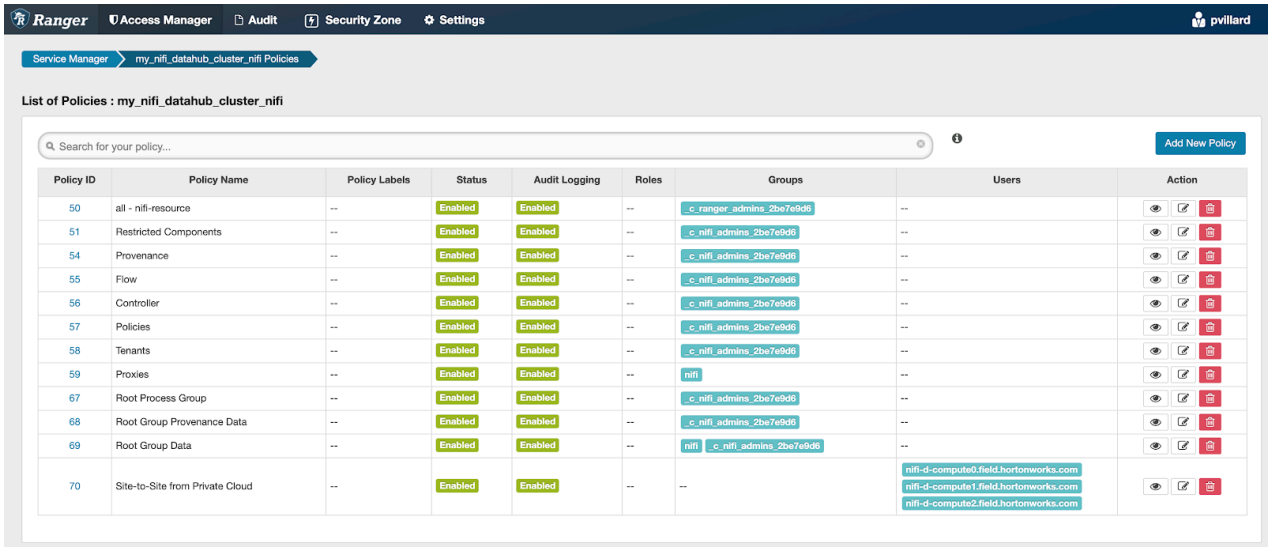

Defining your Cloudera Private Cloud Base data flow

To move data between cloud environments using NiFi site-to-site communication, you require a data flow in your Cloudera Private Cloud Base cluster that can send and receive data from the CDP Public Cloud cluster. To create this data flow, connect a processor to a Remote Process Group configured with HTTP and enable transmission.

Before you begin

- You have defined your CDP Public Cloud data flow and configured Ranger policies for site-to site communication.

- You have the public FQDNs for your CDP Public Cloud cluster nodes.

Procedure

1. In your Cloudera Private Cloud Base cluster, launch the NiFi UI and drag a `GenerateFlowFile` processor onto the canvas.

   For this use case, `GenerateFlowFile` creates 1MB files every 10 seconds.
2. Drag a Remote Process Group onto the NiFi canvas, configure HTTP protocol, and specify one or more of the NiFi nodes running on your CDP Public Cloud cluster.

   After the site-to-site connection is initiated, the source NiFi cluster is aware of the topology of the remote NiFi cluster and of any increase or decrease of the size of the remote cluster. However, it is recommended that you specify at least 2 nodes to ensure higher availability when the site-to-site connection is initiated.

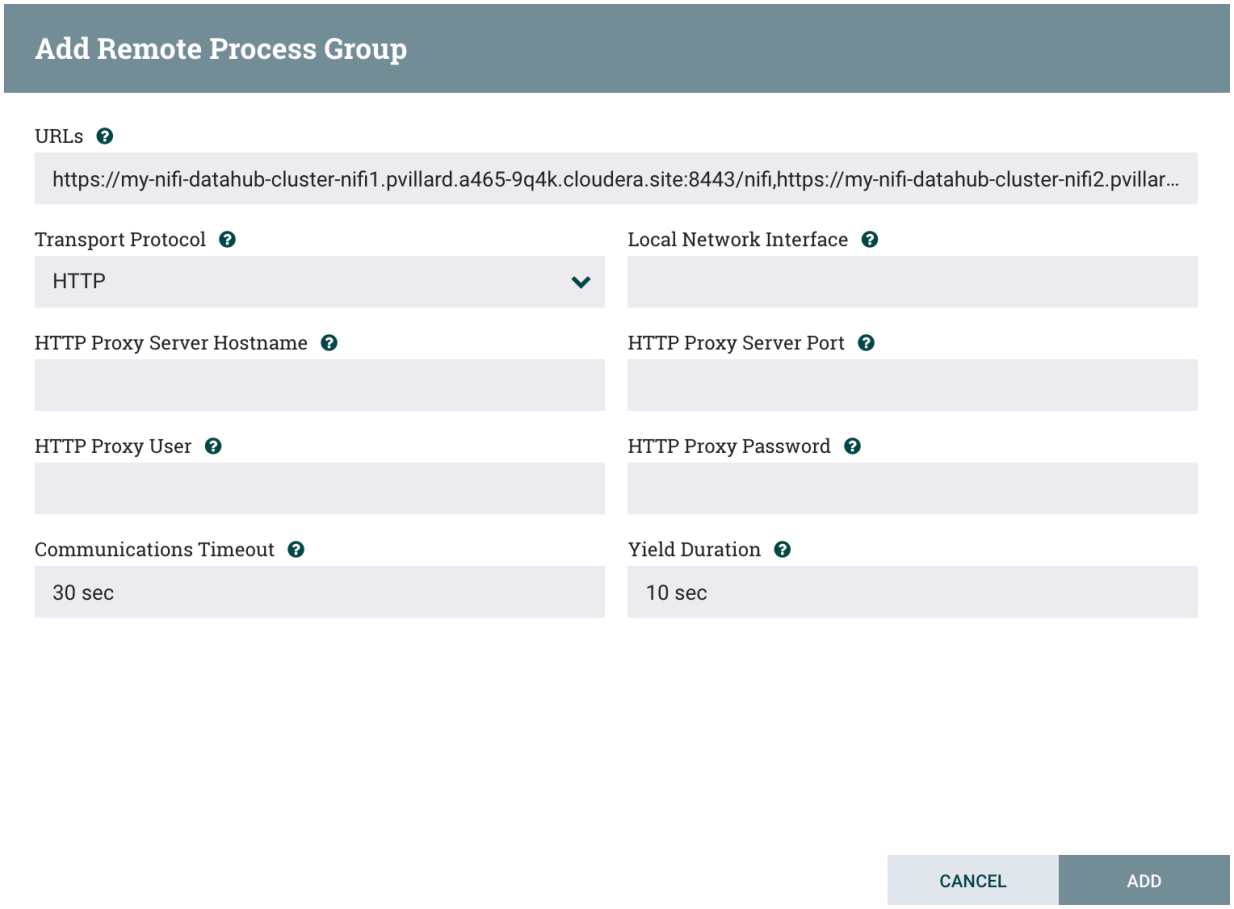

3. Right-click the Remote Process group and select Enable transmission.

4. Connect the `GenerateFlowFile` processor to the Remote Process Group and select the Input Port that you created and started on the remote cluster in CDP Public Cloud:

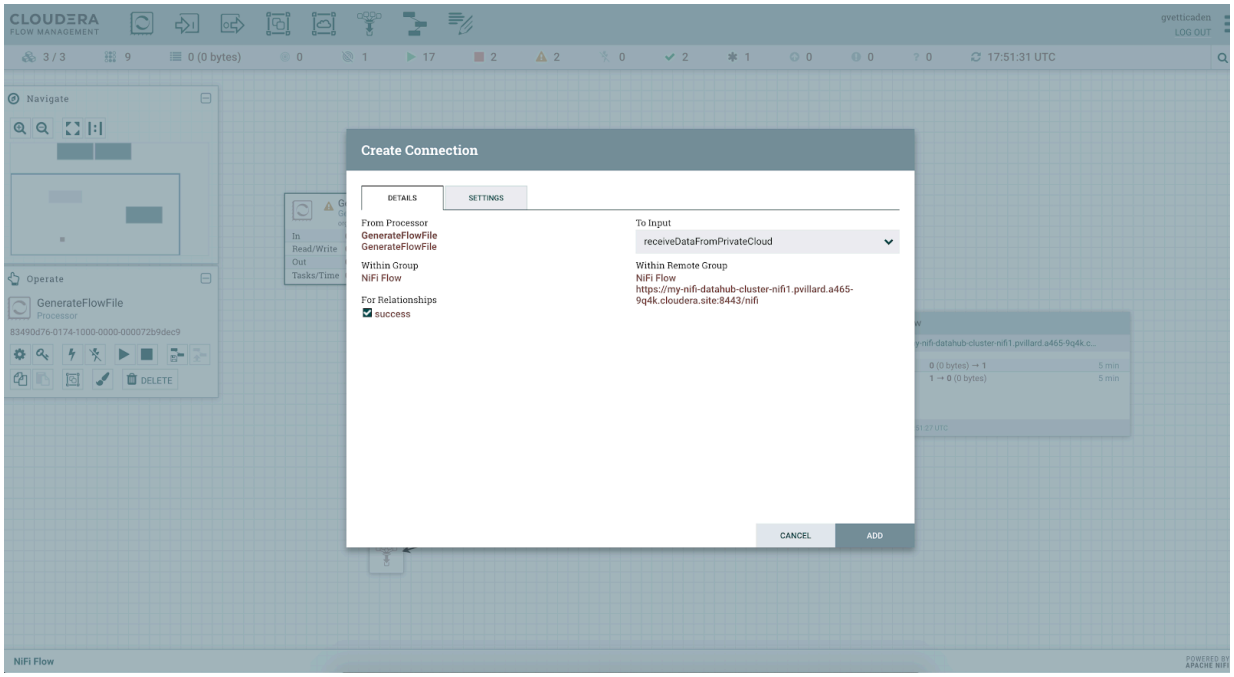

5. You can also define a connection from the Remote Process Group to another component to download data made available by the remote cluster running in the CDP Public Cloud environment. In the this example, the Remote Process Group is connected to a funnel.

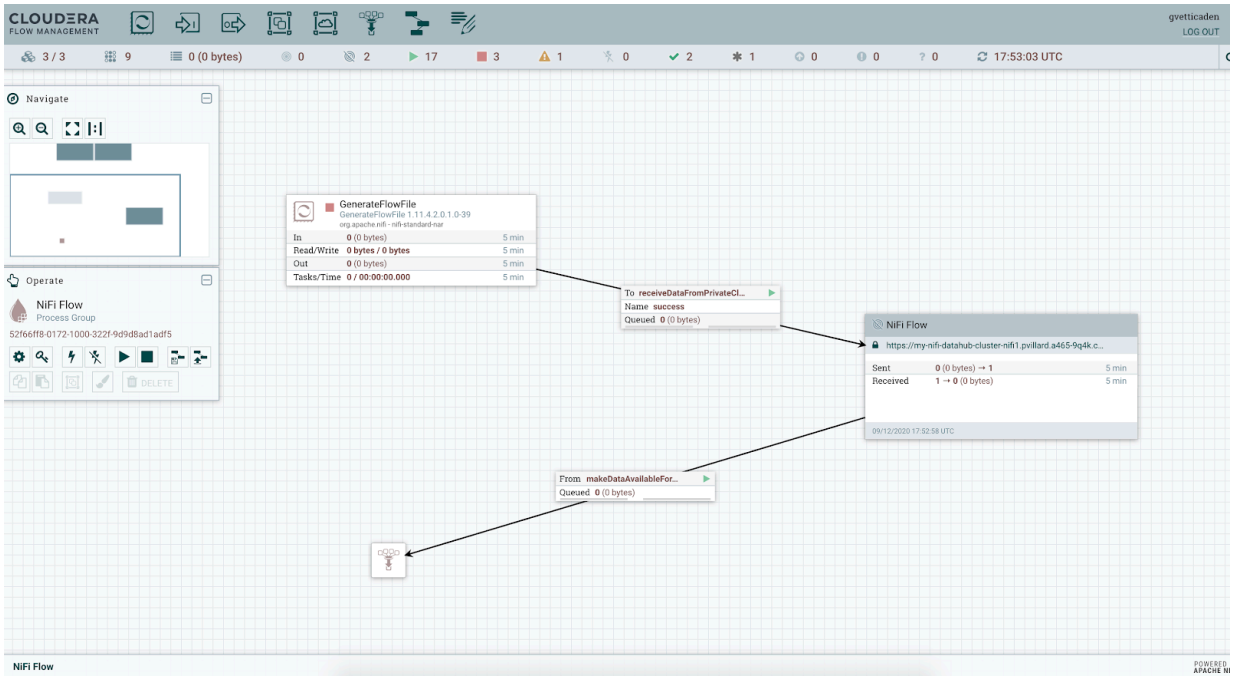

Results

After you have defined the data flow for your Cloudera Private Cloud Base cluster, start the Cloudera Private Cloud Base data flow and confirm that the data is moving back and forth between the environments:

In the Cloudera Private Cloud Base environment, your data flow looks similar to the following:

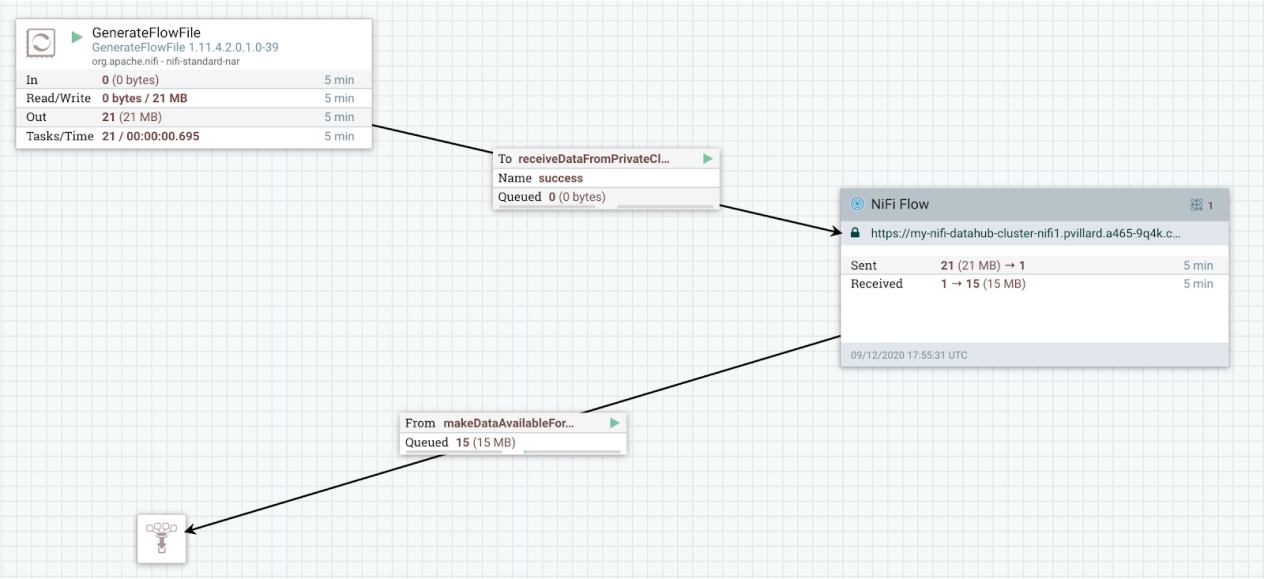

In the CDP Public Cloud environment, your data flow will look similar to the following:

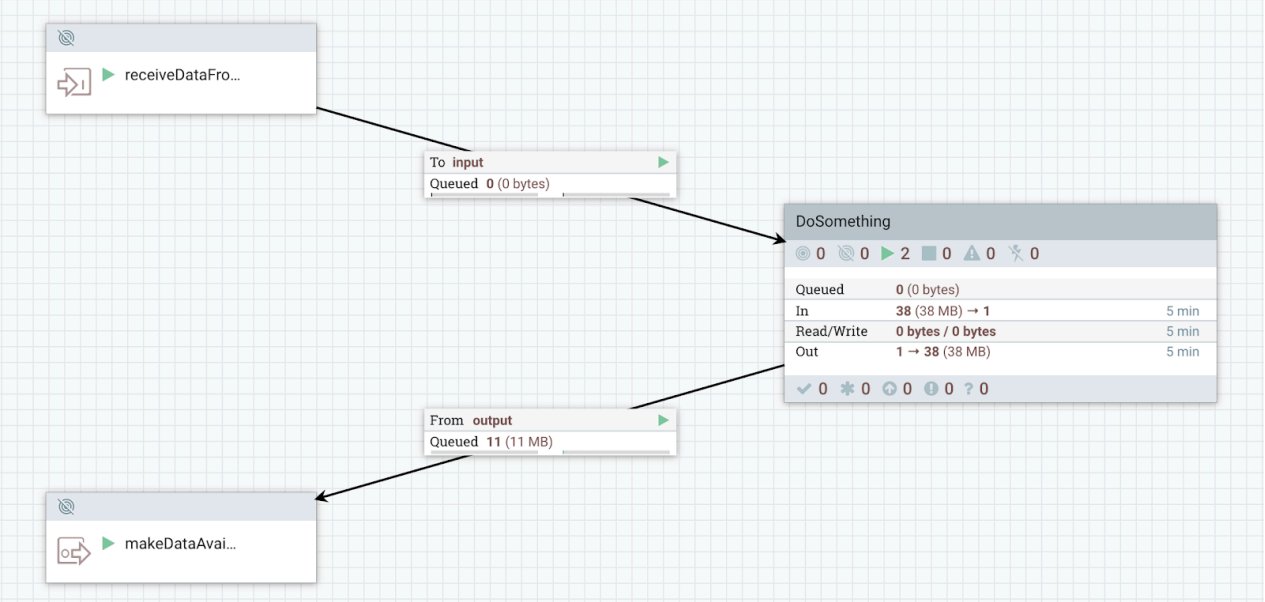

Cloudera DataFlow for Data Hub 7.3.1

Getting Started with Streams Messaging Cluster on CDP Public Cloud

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introducing streams messaging cluster on CDP Public Cloud..... 5

Meet the prerequisites to create streams messaging cluster.....5

Creating Machine User..... 6

Granting Machine User access to environment.....7

Creating Kafka topic.....8

Create Ranger policies for Machine User account.....9

Create topic policy.....10
Create consumer group policy..... 11

Produce data to Kafka topic..... 12

Setting workload password.....12
Connecting to Kafka host.....13
Configuring LDAP authentication.....14
Producing data to Kafka topic..... 14

Consuming data from Kafka topic..... 15

Use Kerberos authentication..... 16

Kerberos authentication using the ticket cache.....17
Kerberos authentication using a keytab..... 18

Monitoring Kafka activity in Streams Messaging Manager..... 19

Use Schema Registry..... 22

Gather configuration information.....23
Finding list of brokers.....23
Finding Schema Registry endpoint..... 24
Creating TLS truststore..... 24
Defining Schema Registry access policies.....25
Producing data in Avro format..... 27
Checking schema registration.....29
Checking producer activity.....30
Consuming data from Kafka topics using stored schemas.....32

Monitor end-to-end latency **33**
Setting up authorization policies 33
Enabling end-to-end latency monitoring 34

Evolve your schema **36**
Reconfiguring the Kafka consumer 36
Reconfiguring the Kafka producer 37

What to do next **38**

Introducing streams messaging cluster on CDP Public Cloud

You can create a streams messaging cluster in CDP Public Cloud and produce data to and consume data from an Apache Kafka topic by using that cluster. The concepts and usage of streams messaging cluster on CDP Public Cloud are explained using a simple workflow.

In this example you use a simple application (vmstat) to generate some raw text data and a kafka-console-producer command to send the data to a Kafka topic. You then use a kafka-console-consumer command to read data from that Kafka topic. After you have the producer and consumer running, you use Stream Messaging Manager (SMM), which is also available on CDP Public Cloud, to monitor the functionality of the workflow that you create.

Alternately, you can also structure the data in an object format and send it to Kafka in Avro format. To do this, you store a schema in CDP's Schema Registry and understand how to use a simple Java Kafka client to send and read data using that schema.

You can also update your schema to fit your requirements, and reconfigure the consumer and producer to use the latest schema.

The complete workflow consists of the following tasks:

1. Meeting the prerequisites
2. Creating a Machine User
3. Granting Machine User access to environment
4. Creating a Kafka topic
5. Creating Ranger policies
6. Producing data to Kafka topic
7. Consuming data from Kafka topic
8. Using Kerberos authentication
9. Monitoring Kafka activity in SMM
10. Using Schema Registry
11. Monitoring end-to-end latency in SMM
12. Evolving your schema

Meet the prerequisites to create streams messaging cluster

You need to ensure that your user is assigned the required role and privilege to create the workflow with Kafka on CDP Public Cloud.

Ensure that you have met the prerequisites outlined in the *Creating your First Streams Messaging Cluster in CDP Public Cloud*, in the *Meet the prerequisites* section.

Note: As stated in the prerequisites document, your user must have the EnvironmentAdmin role. This is required so that you have admin privileges on Kafka and Ranger to perform the tasks required in this tutorial. If you do not have this role, request your environment administrator for assistance.

After you meet the prerequisites, you need to create your streams messaging cluster using a default cluster definition.

To complete this tutorial you need a running Kafka cluster on CDP Public Cloud. For instructions, see *Create your cluster*.

Related Information

Meet the prerequisites

Create your cluster

Creating Machine User

You must create a dedicated service account called Machine User to use in the production environment. You create a Machine User in the Management Console with a unique name.

Before you begin

You must have created a Kafka cluster on CDP Public Cloud.

About this task

You can use your personal user account to connect to CDP for all the examples in this tutorial. However, in a production environment, you must create a dedicated service account to use ingestion processes or pipelines. So, instead of using your own account, this tutorial takes you through creating a service account.

Machine User is the CDP nomenclature for a service account. For more information on Machine User, see *CDP machine user*.

Procedure

1. Navigate to Management Console User Management .
2. Click Actions Create Machine user .
3. Enter a unique name for the user and click Create.
   After you create a machine user, you see a page like the following one with the user details.

   Note: The Workload User Name (srv_kafka-client, in the example below) is different from the actual user name (kafka-client). The Workload User Name is the identifier you use to configure the access for your workload or application later.

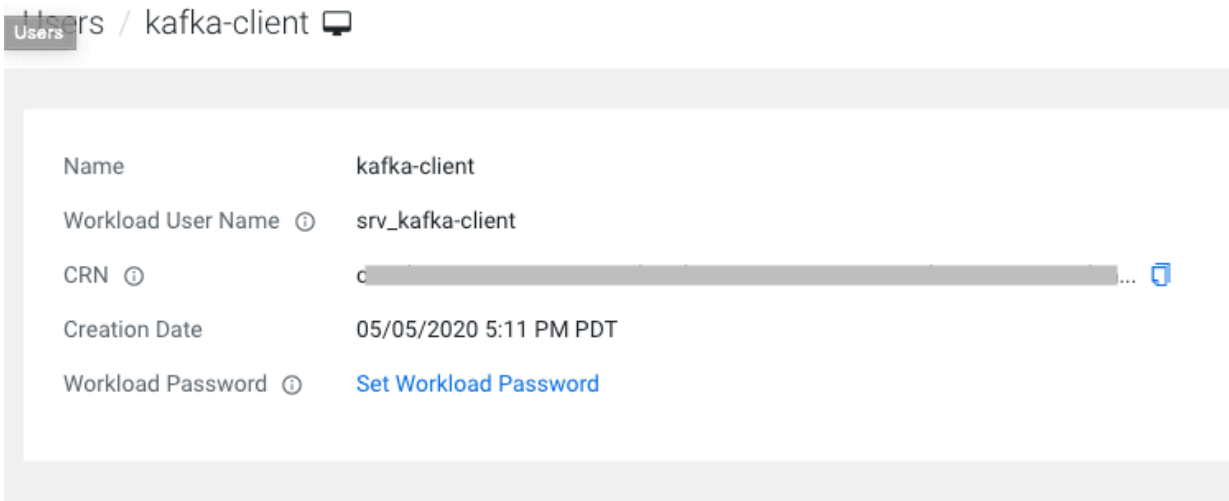

What to do next

Click Set Workload Password and follow the prompts to set a password for the Machine User. You can leave the Environment field empty.

Related Information

CDP machine user

Granting Machine User access to environment

Before you start working with the Machine User account, to access the Kafka cluster on CDP Public cloud, you must grant the user access to the environment that contains the cluster.

Before you begin

You must have created a Machine User account.

About this task

For more information on providing the user access to your cluster, see *Give users access to your cluster*.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. Click Actions Manage Access .
3. In the search field, type the name of the Machine User and select it from the search results.
4. Check the EnvironmentUser role and click Update Roles.

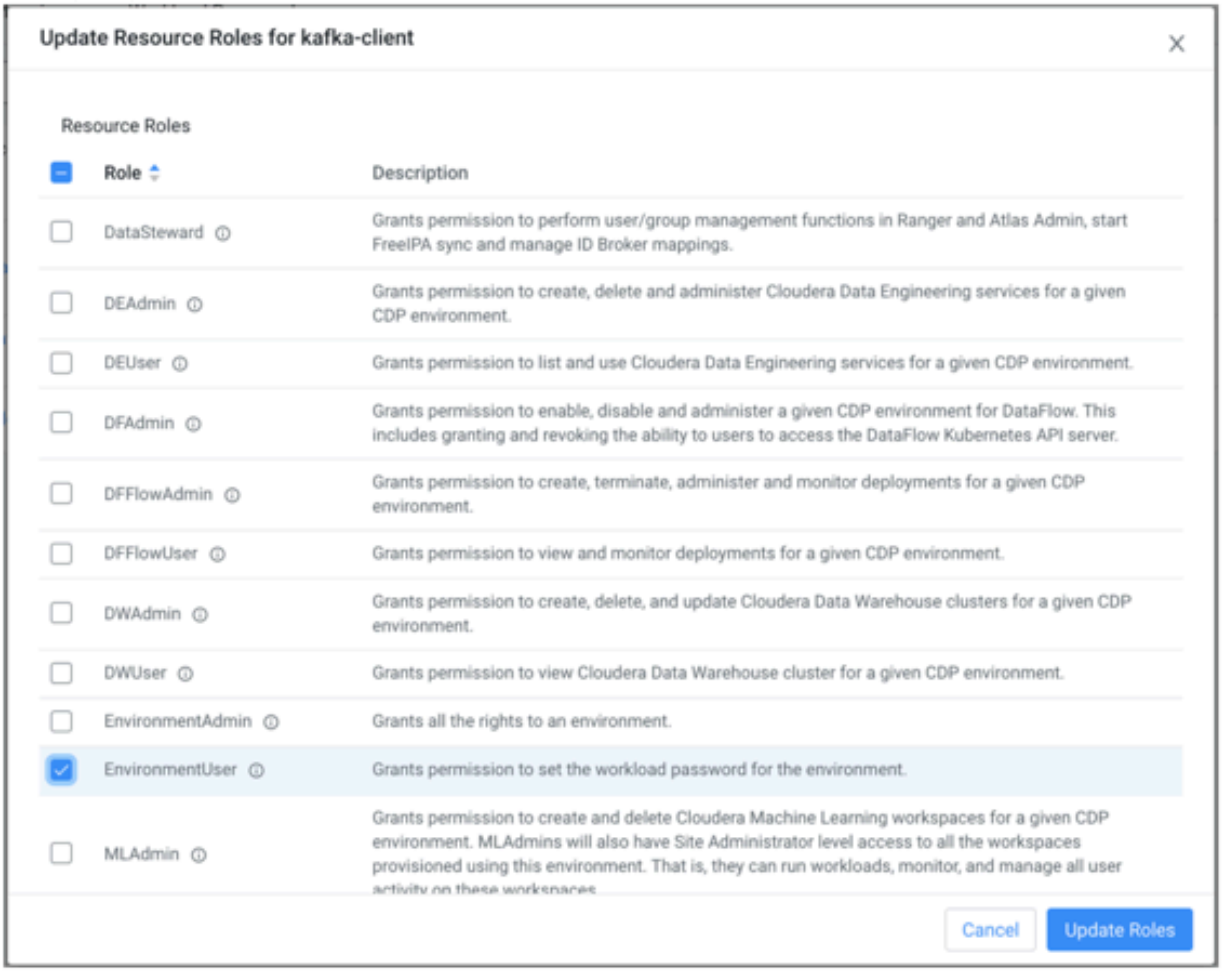

5. Go to the environment page and click Actions Synchronize Users to FreeIPA , to ensure that the role assignment is in effect for the environment.

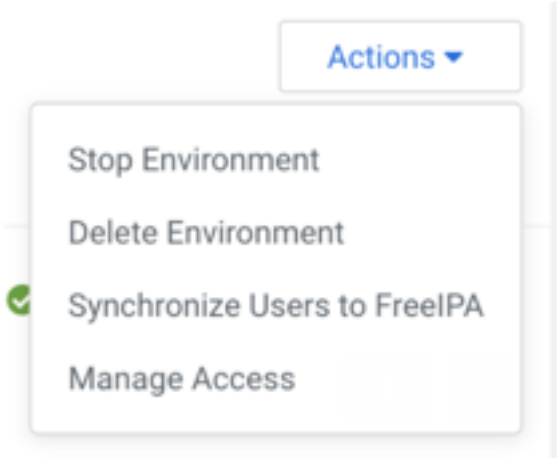

6. On the Synchronize Users to FreeIPA page, click Synchronize`Users.

What to do next

You need to create Kafka topics.

Related Information

Give users access to your cluster

Creating Kafka topic

After you create a Kafka cluster, you can create Kafka topics. You create Kafka topics in the environment where your Kafka cluster is running by using the Streams Messaging Manager (SMM) web UI.

Before you begin

You must have created a Kafka cluster and a Machine User account, and granted environment access to the Machine User.

About this task

You create two Kafka topics in the tutorial:

- machine-data

  Topic containing machine usage data in plain text format.
- machine-data-avro

  Topic containing machine usage data in Avro format.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click Streams Messaging Manager (SMM) on the Services pane to open the SMM web UI.

   SMM is CDP’s tool to monitor and manage Kafka. You use it to create the topics you need.

4. In the SMM web UI, click the topics icon ( ) on the left-hand bar.
5. Click Add New.
6. Enter the following details and click Save:
   - Topic name: machine-data
   - Partitions: 10
   - Availability: Maximum
   - Cleanup Policy: delete
7. Click Add New to add a second topic.
8. Enter the following details for the new topic and click Save:
   - Topic name: machine-data-avro
   - Partitions: 10
   - Availability: Maximum
   - Cleanup Policy: delete

Results

In the SMM Topics page, type machine in the search field to filter the topics. You must see both topics you created.

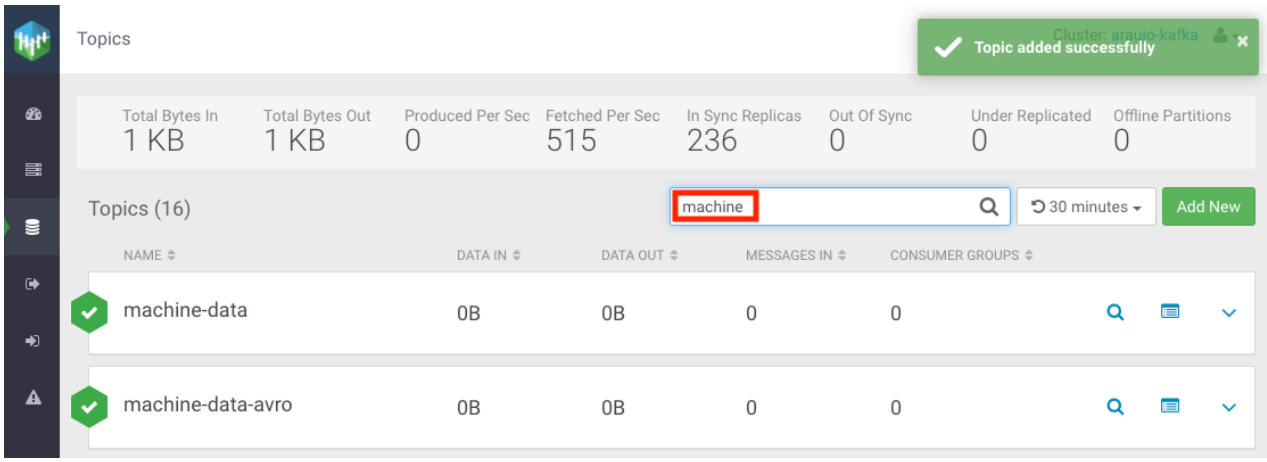

Create Ranger policies for Machine User account

To read data from or write data to Kafka topics, you must grant certain privileges to your Machine User. To do this, you create new access policies for the Kafka topics in Ranger. You create one Topic policy and one Consumer Group policy.

Because your personal user is an environment administrator, you already have privileges to access and modify Kafka topics. This is not the case, though, with the Machine User account you created.

Authorization policies in CDP are maintained and enforced by Apache Ranger. To learn more about Apache Ranger authorization policies, see *Using Ranger to Provide Authorization in CDP*.

To allow the Machine User account to read from and write data to the Kafka topics you created, it is necessary that you define new access policies for those topics in Ranger. You need to create the following policies:

- Topic policy to grant access to the two Kafka topics.
- Consumer Group policy to define which consumer groups the Machine User can use to consume data from the Kafka topics.

Related Information

Using Ranger to Provide Authorization in CDP

Create topic policy

Learn how to grant privileges to a Machine User for accessing and modifying Kafka topics using the Ranger web UI. You must create a new policy, select the Machine User, and add permissions for the Machine User.

Before you begin

You must have created a Machine User, granted environment access to the Machine User, and created two Kafka topics.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. Click Ranger on the top pane to open the Ranger web UI.
3. In the Ranger web UI, click Access Manager Resource Based Policies .
4. Under the KAFKA group, select the policy of your Kafka cluster.

   To select it, click on the policy name, and not on the icon. You see the list of pre-defined policies for your Kafka cluster.

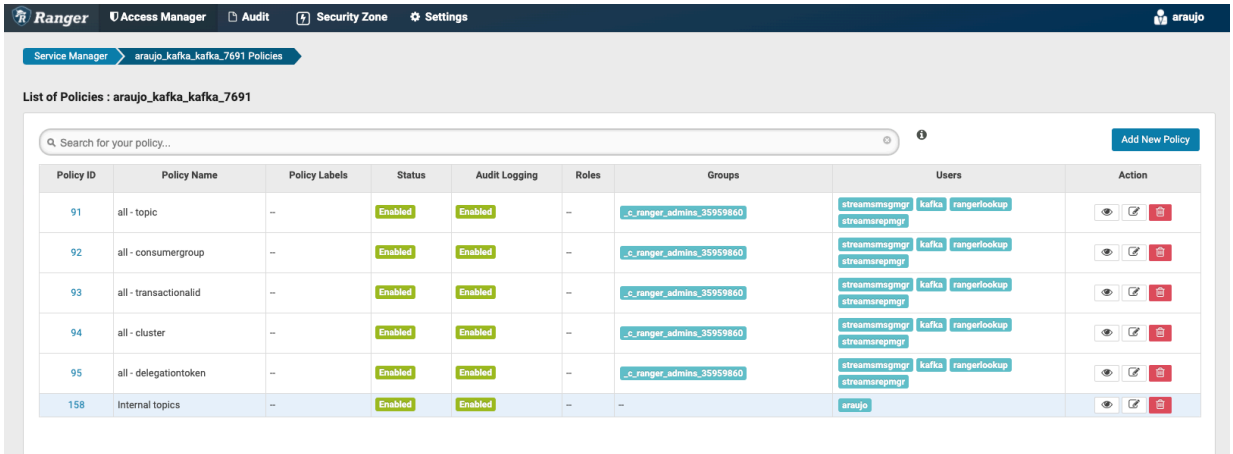

5. Click Add New Policy to create a new policy and enter the following details:
   - Policy Name: Machine Data topics
   - Topic: machine-data*
   - Description: All topics prefixed with machine-data
6. Below the Allow Conditions section, click the empty Select Users box and select the Machine User you created.

7. Click Add Permissions, select the Publish, Consume, and Describe permissions, and then click the tick mark.

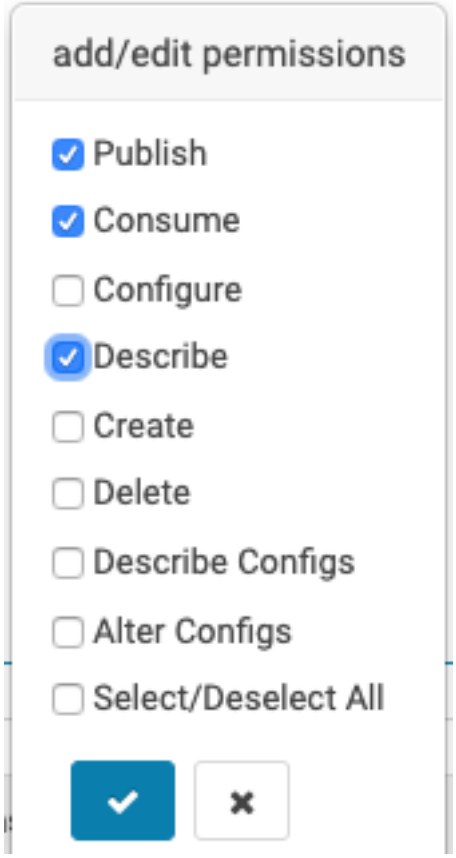

8. Scroll to the bottom of the page and click Add to save the policy.

Create consumer group policy

Learn how to grant privileges to a Machine User for using consumer groups to consume data from the Kafka topics by using the Ranger UI. You create a new policy, select the Machine User, and add permissions for the Machine User.

Before you begin

You must have created a Machine User, granted environment access to the Machine User, and created two Kafka topics.

Procedure

1. Click Add New Policy, to create a new one, and enter the following details:
   - Policy Name: Machine Data consumer groups
   - Consumer group: machine-data*
   - Description: All consumer groups prefixed with machine-data
2. Below the Allow Conditions section, click on the empty Select User box and select the Machine User you had created.

   You can type the name of the user to filter the list of users. Also note that the user name appears with the srv_ prefix, indicating that it is a Machine User.

3. Click Add Permissions, select the Consume and Describe permissions and then click the tick mark.

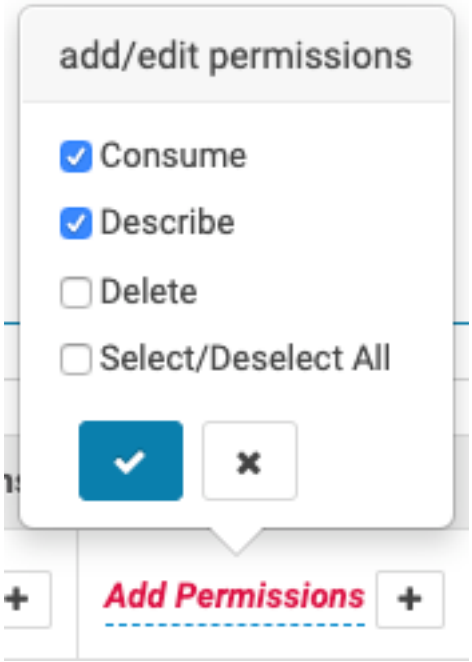

4. Scroll to the bottom of the page and click Add to save the policy.

Results

Now you should see your new policies listed in Ranger. Your Machine User is ready for use.

What to do next

You can start producing data to the Kafka topic.

Produce data to Kafka topic

Learn how to produce data to a Kafka topic. To do this, you need to set the workload password, connect to a Kafka host, provide LDAP authentication, and finally produce data to a Kafka topic.

The applications that produce data to or consume data from Kafka can run on any of the servers in your network, provided they have connectivity to the Kafka cluster in CDP. These applications should not be executed on the hosts of the Kafka cluster, because this could affect the Kafka operation.

To produce data to a Kafka topic, you use the kafka-console-producer command line application because the command line application is already installed on the Kafka hosts and helps you to understand the application configuration without having to build an application. In a later step you configure and run an application on a separate host to connect to the cluster.

Setting workload password

Learn how to set your personal workload password.

Before you begin

You must have created a Machine User account.

Procedure

1. Click Profile in your user menu on the CDP console.

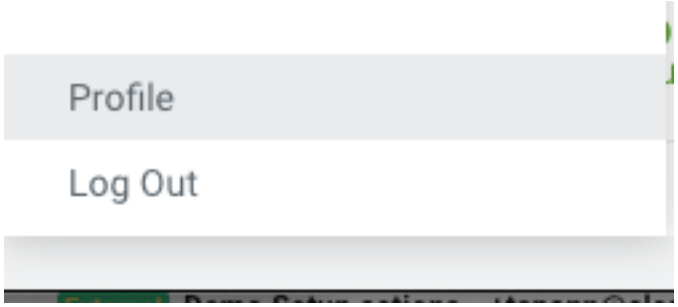

2. On your profile page, click Set Workload Password and follow the prompts to set a password for your user. You can leave the Environment field empty.

Connecting to Kafka host

Learn how to connect to a Kafka host from the Management Console. You need to select the Kafka cluster you created, select a Kafka broker, and then connect to the broker host.

Before you begin

You must have created a Machine User account, granted environment access to the Machine User, created Kafka topics, and created Ranger policies.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. On the Kafka cluster page, click Hardware and select one of the Kafka brokers to run your producer test.

   Select and copy the full hostname of the broker.

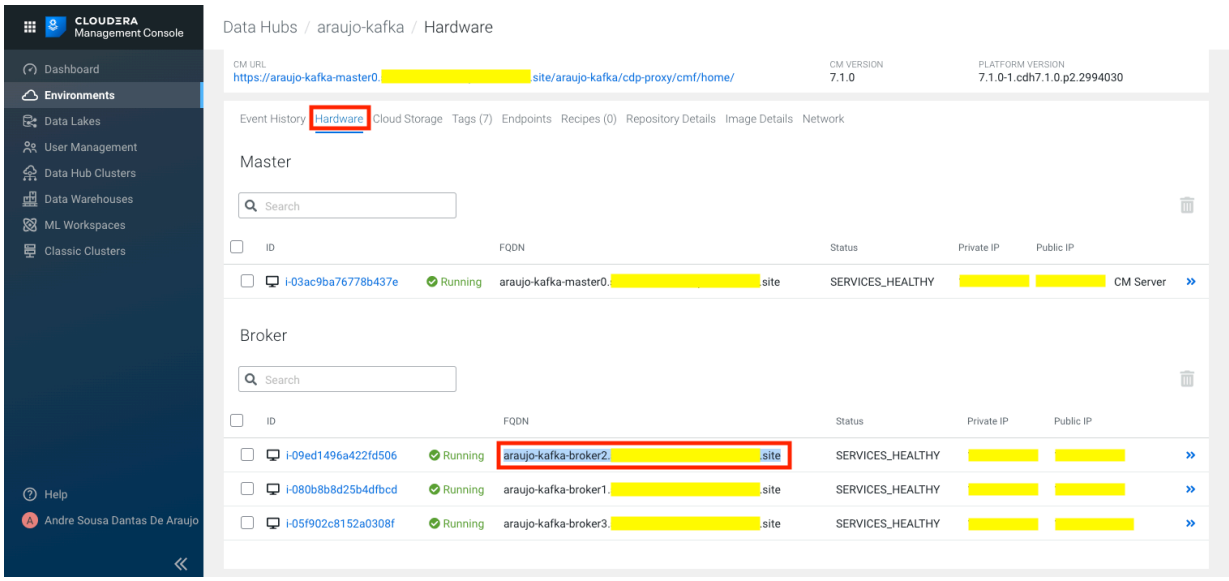

4. Using the SSH client of your preference, connect to the broker host using your CDP user and your workload password for authentication.

   For example, if connecting from a terminal, you can use:

```
ssh [***YOUR_USER_NAME***]@[***BROKER_HOST_NAME***]
```

Configuring LDAP authentication

All the Kafka clusters created in CDP are secure because TLS is enabled to encrypt communications between clients and the cluster, and strong authentication is enforced, requiring clients to authenticate either through Kerberos or LDAP.

About this task

In this example, you use LDAP authentication. LDAP authentication is not enabled by default.

Before you begin

You must have connected to the Kafka host.

Procedure

1. Provide the client with the security.protocol and sasl.mechanism information to indicate the security protocol and authentication mechanism to use.
2. Provide the client with the ssl.truststore.location information which is required for the client to trust the brokers' certificates.

   All the Kafka hosts deployed by CDP have a valid truststore deployed at /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks. You need to use this truststore. Later you look at how to create a truststore from scratch.
3. Provide the client with the sasl.jaas.config information which is the LDAP credentials to use for authentication.

   For a detailed explanation on how to configure TLS/SSL and LDAP authentication for Kafka clients, see *Configure Kafka clients*.

Related Information

Configure Kafka clients for LDAP authentication

Producing data to Kafka topic

To produce data to Kafka topics, you create a property file, ensure that its permissions are set restrictively, list all the topics that the Machine User has access to, and produce a text message to the Kafka topic.

About this task

Here, you produce data to a Kafka topic on your SSH session.

Before you begin

You must have created a Kafka cluster, created a Machine User, granted environment access to the Machine User, created Kafka topics, and created Ranger policies.

Procedure

1. Create a new client-ldap.properties file with the following content:

```
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
```

```
ssl.truststore.location=/var/lib/cloudera-scm-agent/agent-cert/cm-auto-g
lobal_truststore.jks
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
 required username="srv_kafka-client" password="SuperSecretPassword";
```

Note: Replace the values of the username and password to match with the machine user credentials that you created.

2. The client-ldap.properties file contains sensitive information. Ensure its permissions are set restrictively.

```
chmod 400 client-ldap.properties
```

With this configuration you can start running Kafka client applications.

3. Run the following command to list all the topics that the Machine User has access to.

```
BROKER_HOST=$(hostname -f)

kafka-topics \
  --bootstrap-server $BROKER_HOST:9093 \
  --command-config client-ldap.properties \
  --list
```

The output include a number of INFO lines, with the two topics you created at the end of the output:

```
machine-data
machine-data-avro
```

If you do not see them, go back and review the previous steps.

4. Run the following command to produce a simple text message to the Kafka topic by using the kafka-console-producer command.

```
echo 'Hello, Kafka!' | kafka-console-producer \
  --broker-list $BROKER_HOST:9093 \
  --producer.config client-ldap.properties \
  --topic machine-data
```

Results

If the output of the command above contains only INFO lines and no ERROR lines, it indicates that data was produced correctly to the Kafka topic.

What to do next

In the next section, you use the kafka-console-consumer command to verify the contents of the Kafka topic.

Consuming data from Kafka topic

After you produce data to a Kafka topic, you can consume data from the Kafka topic.

About this task

The configuration that you used previously to produce data to Kafka topics is the same that you use to consume data from Kafka topics. You need to verify that the data was written correctly to the Kafka topic and can also be read by the client.

Before you begin

You must have produced data to your Kafka topic.

Procedure

Run the following command to consume the data produced previously:

```
kafka-console-consumer \
  --bootstrap-server $BROKER_HOST:9093 \
  --consumer.config client-ldap.properties \
  --topic machine-data \
  --group machine-data-1 \
  --from-beginning
```

Results

You see the following output, which tells you that the client successfully read it from the topic:

```
Hello, Kafka!
```

Note:

If you rerun the same command, reusing the consumer group name (machine-data-1 in the previous example), the data is not shown because the client continues from where it left off in the previous execution. To read the data again, change the group name to something else (for example, machine-data-2).

Use Kerberos authentication

Learn how and when to configure Kerberos authentication.

LDAP authentication is generally easier to configure for remote clients, because it does not require Kerberos libraries or clients to be installed and the remote clients do not need direct connectivity to your Kerberos or LDAP server.

In some situations, though, Kerberos authentication may be required and/or preferred. In this section, you run the same kafka-console-consumer command you used for LDAP authentication, but this time using Kerberos authentication to demonstrate the required configuration.

When using Kerberos, you can use two different sources for the authentication credentials: a ticket cache or a keytab. The ticket cache is the place where the Kerberos ticket for a user is stored after the user authenticates successfully. You can use the ticket cache when the user has been already authenticated using a username and password. The keytab is a special and sensitive file that contains the user credentials. Keytabs do not require that the user authenticates previously.

In this example, you create a new configuration file called client-kerberos.properties, which is similar to the client-ldap.properties file that you used in the previous example but with Kerberos-specific parameters. You also need an additional configuration file, which you call either jaas-cache.conf or jaas-keytab.conf, depending on the credentials source you use. This JAAS configuration file is used to communicate the authentication credentials source (ticket cache or keytab) to the Kafka client.

For more details on how to enable Kerberos authentication for Kafka, see *Enable Kerberos authentication* for Kafka clients.

Related Information

Enable Kerberos authentication

Kerberos authentication using the ticket cache

You can configure Kerberos authentication using a ticket cache. To do so, you authenticate with Kerberos to acquire a valid Kerberos ticket. Then you connect to Kafka authenticating with the Kerberos ticket.

Before you begin

You must have produced data to a Kafka topic and consumed data from the Kafka topic.

Procedure

1. Create a file called client-kerberos.properties with the following content:

```
security.protocol=SASL_SSL
sasl.mechanism=GSSAPI
sasl.kerberos.service.name=kafka
ssl.truststore.location=/var/lib/cloudera-scm-agent/agent-cert/cm-auto-glo
bal_truststore.jks
sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required us
eTicketCache=true;
```

2. Before connecting to Kafka, you need to authenticate successfully with Kerberos to acquire a valid Kerberos ticket. You can use kinit for that. Run the following commands, replacing srv_kafka-client with your Machine User name. When prompted for the password, provide the Workload Password that you had set.

```
kdestroy

kinit srv_kafka-client
```

3. If the command is successful, you should have a valid ticket in your ticket cache, as shown by the following klist command:

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_1501600556
Default principal: srv_kafka-client@XYZ.SITE

Valid starting       Expires              Service principal
05/07/2020 03:10:58  05/08/2020 03:10:52  krbtgt/XYZ.SITE@XYZ.SITE
 renew until 05/14/2020 03:10:52
```

If the command is not successful, your ticket cache should be empty, as shown in the following example:

```
$ klist
klist: No credentials cache found (filename: /tmp/krb5cc_1501600556)
```

4. After you have a valid ticket in the ticket cache you can run the following commands to connect to Kafka, authenticating with the Kerberos ticket.

```
BROKER_HOST=$(hostname -f)

kafka-console-consumer \
  --bootstrap-server $BROKER_HOST:9093 \
  --consumer.config client-kerberos.properties \
  --topic machine-data \
  --group machine-data-$RANDOM \
  --from-beginning
```

You should see the contents of the machine-data topic as follows:

```
Hello, Kafka!
```

5. After testing, destroy the ticket in your ticket cache:

```
kdestroy
```

If you try to run the kafka-console-consumer command again, without the ticket, you see authentication errors like the following example:

```
Caused by: javax.security.auth.login.LoginException:
Could not login: the client is being asked for a password, but the Kafka c
lient code does not
currently support obtaining a password from the user. not available to
garner  authentication
information from the user
```

Kerberos authentication using a keytab

You can configure Kerberos authentication using a keytab. To do so, you download the keytab from the environment where your Kafka cluster is running, copy the keytab to your broker host, create a configuration file with required properties, and connect to Kafka.

About this task

As mentioned previously, instead of using a ticket from the ticket cache you can also authenticate using a keytab.

Before you begin

You must have produced data to a Kafka topic and consumed data from the Kafka topic.

Procedure

1. Navigate to Management Console User Management .
2. Select the Machine User.
3. Click Actions Get Keytab .
4. On the Get Keytab dialog box, select the environment where the Kafka cluster is running and click Download.
5. Copy the keytab that was downloaded on your computer to the broker host you were connected to.
6. The keytab file contains sensitive information. Ensure that its permissions are set restrictively:

```
chmod 400 ./kafka-client.keytab
```

7. Create the jaas-keytab.conf file with the following content:

```
KafkaClient {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab="./kafka-client.keytab"
  principal="srv_kafka-client@XYZ.SITE";
};
```

Where the value of the keyTab property is the path of the keytab copied to the host and the value of the principal property is the Kerberos principal name of the Machine User account. You can find the principal name by listing the content of the keytab, as shown in the following example:

```
$ klist -kt ./kafka-client.keytab
Keytab name: FILE:kafka-client.keytab
KVNO Timestamp           Principal
---- ------------------- ------------------------------------------------
------
   0 05/07/2020 03:32:01 srv_kafka-client@XYZ.SITE
```

```
    0 05/07/2020 03:32:01 srv_kafka-client@XYZ.SITE
```

8. Run the following commands to connect to Kafka, authenticating with the Kerberos keytab.

```
BROKER_HOST=$(hostname -f)

export KAFKA_OPTS="-Djava.security.auth.login.config=./jaas-keytab.conf"

kafka-console-consumer \
  --bootstrap-server $BROKER_HOST:9093 \
  --consumer.config client-kerberos.properties \
  --topic machine-data \
  --group machine-data-$RANDOM \
  --from-beginning
```

The KAFKA_OPTS environment variable is used to communicate the location of the the jaas-keytab.conf configuration file to kafka-console-consumer command.

Results

You see the content of the machine-data topic as follows:

```
Hello, Kafka!
```

Monitoring Kafka activity in Streams Messaging Manager

You can use Streams Messaging Manager (SMM) to monitor Kafka activity. To do this, you run two SSH sessions, one for producing data to a Kafka topic and the other to consume the data from the Kafka topic. Then you go to the SMM UI from the Management Console to track the producer and consumer activity.

About this task

Now that you know how to configure a Kafka client to connect to the Kafka cluster in CDP, generate some activity and monitor it using SMM. To do this, you collect machine usage data from the broker host using the vmstat command and stream that data into the machine-data topic.

For detailed and comprehensive information about SMM, see *Streams Messaging Manager*.

Procedure

1. Start by opening two SSH connections to the same broker host.
2. On the first session, run the following command.

```
BROKER_HOST=$(hostname -f)

vmstat 1 1000 | kafka-console-producer \
  --broker-list $BROKER_HOST:9093 \
  --producer.config client-ldap.properties \
  --topic machine-data
```

This command runs vmstat and sends each line from its output as one message every second to the machine-data topic.

This example assumes that you are using LDAP authentication but you can use the authentication method of your preference.

Leave the session running.

3. On the second SSH session, run the following command to consume the data:

```
BROKER_HOST=$(hostname -f)

kafka-console-consumer \
  --bootstrap-server $BROKER_HOST:9093 \
  --consumer.config client-ldap.properties \
  --topic machine-data \
  --group machine-data-$RANDOM \
  --from-beginning
```

   You see the data produced by the first session continuously appear on the screen as output.

   Leave the second session also running.

4. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
5. On the Data Hubs tab of your environment, select the Kafka cluster you created.
6. Click Streams Messaging Manager on the services pane to open the SMM web UI.

   It may take a few minutes for the consumer or producer activity to start showing in the SMM UI. You may have to refresh your page a few times until the data starts to appear.

   Once it does, you see one active producer on the left-hand side of the page and at least one active consumer listed on the right-hand side of the page.

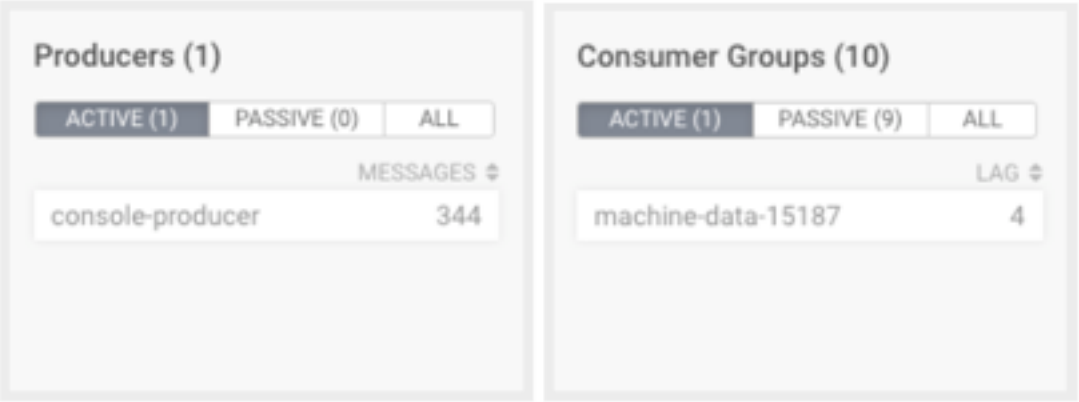

Note: SMM shows consumers and producers with some activity in the last few minutes, as active. Because of this, you may see those producers and consumers that finished recently, as active.

You also see the LAG metrics listed besides each consumer. This metric indicates the number of messages the consumer is behind the latest message produced to the Kafka topic. You can use this to quickly identify slow consumers that are lagging behind producers.

7. Click either on the producer or on the consumer, and SMM shows their activity, highlighting all the topics and partitions they are writing to or reading from, respectively.

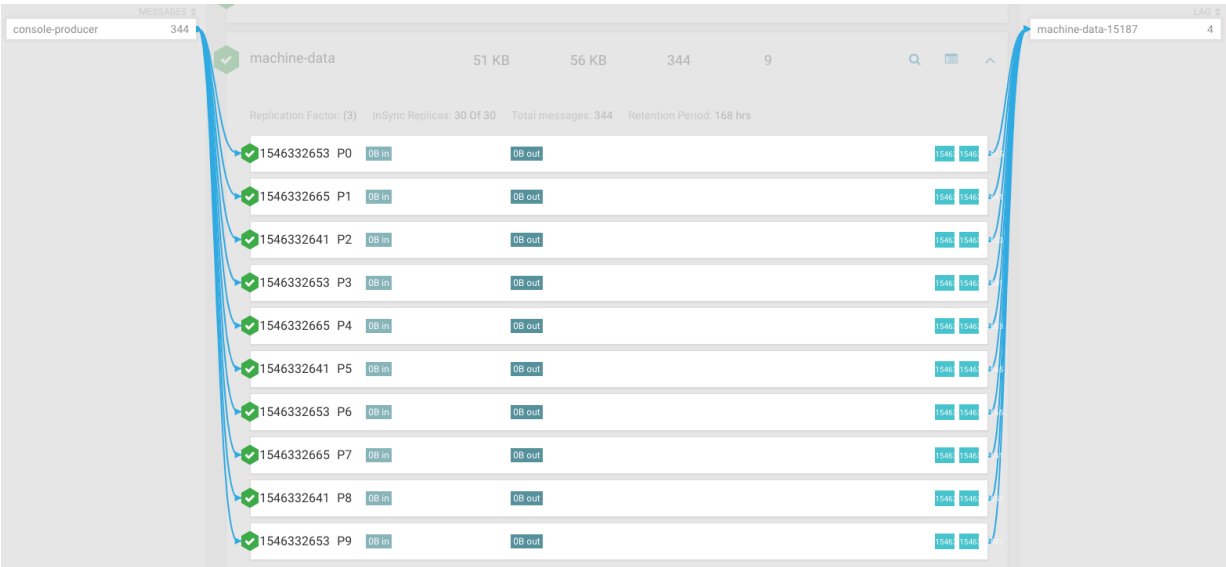

In this page you can identify metrics for the overall Kafka topic activity and partition-level. The In Sync Replica (ISR) set for each partition can be identified on the screen. The ID of the LEADER replica for each partition is shown on the left-hand side, while the FOLLOWER replicas are represented as teal-colored boxes on the right-hand side.

8. Click on an empty part of the page to clear the consumer or producer selection, and then click the profile icon for the machine-data topic.

This opens the Kafka topic details page where you can find all the topic-related metrics and utilization charts.

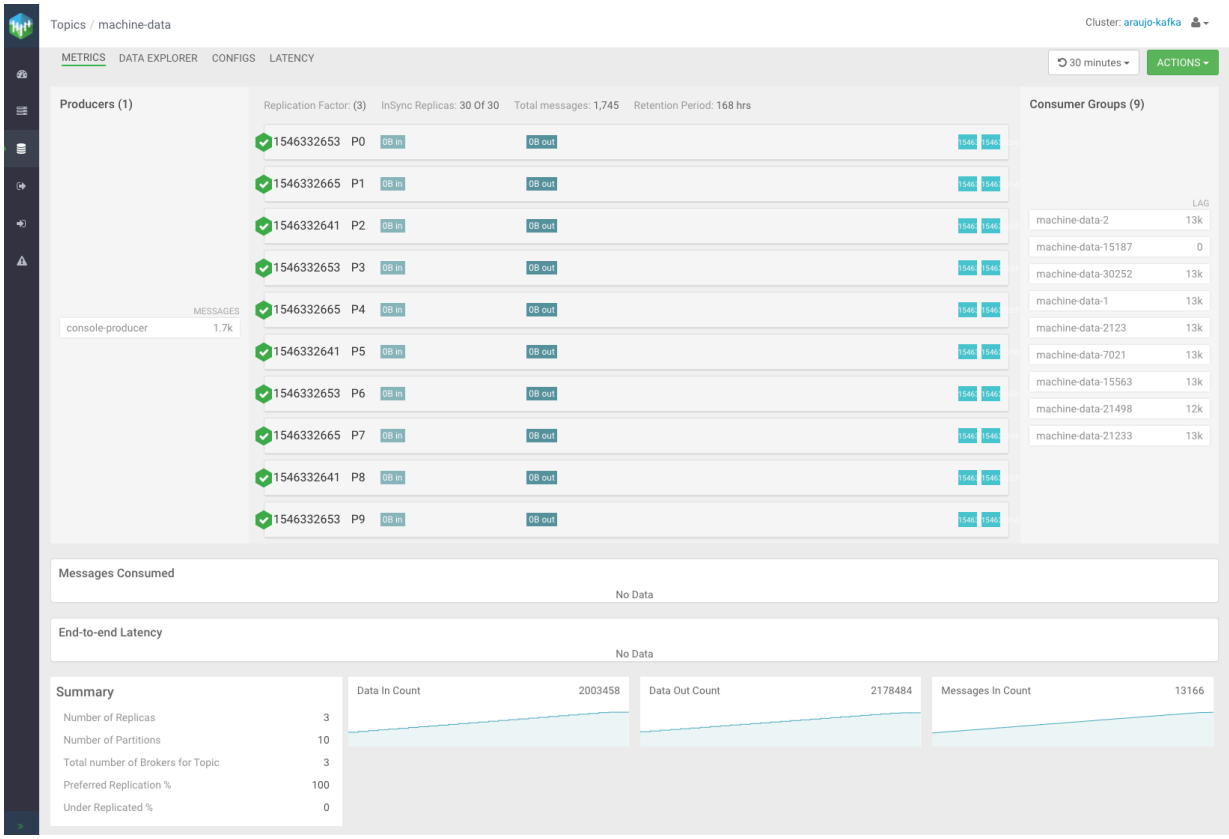

9. Click the DATA EXPLORER tab.

   In the Data Explorer page you can sample the data that is flowing through the Kafka topic. Select different partitions and adjust the offset sliders to become familiar with those controls.

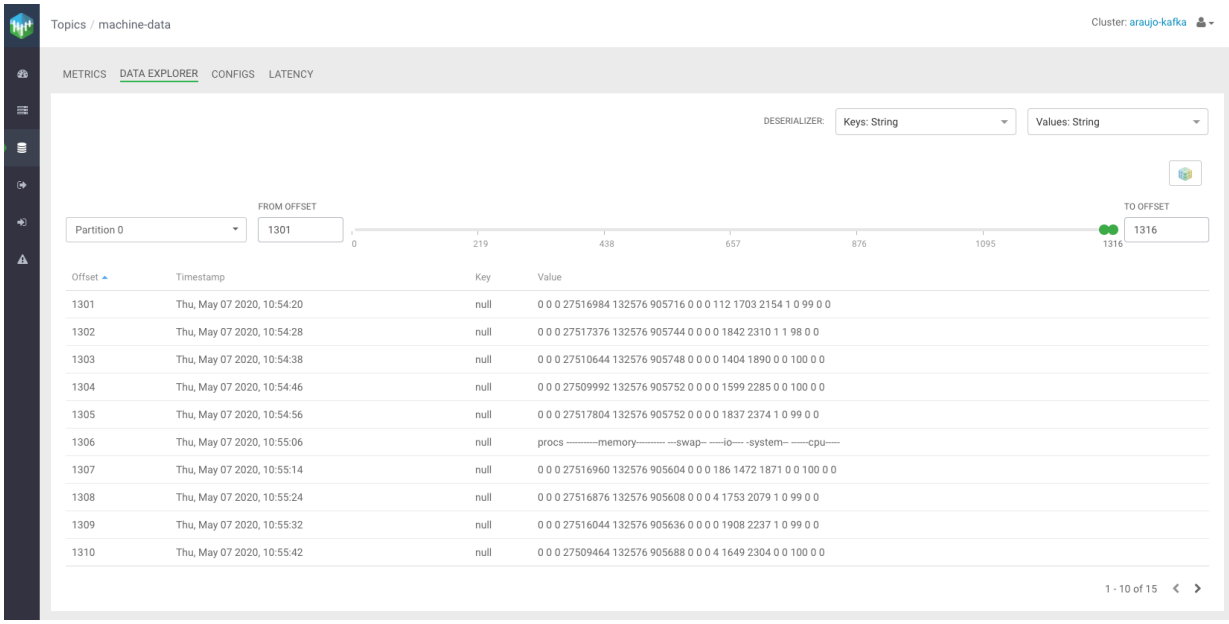

Related Information

Streams Messaging Manager

Use Schema Registry

You can use a sample Kafka client application written in Java, to produce data to and consume data from Kafka topics using schemas stored in the Schema Registry.

Schema Registry provides a shared repository of schemas that allows applications to flexibly interact with each other. Applications frequently need a way to share metadata across data format, schema, and semantics. Schema Registry addresses these challenges by evolving schemas such that a producer and consumer can understand different versions of the schemas but still read all information shared between both versions and safely ignore the rest.

For more information on the topics covered in this section, check the following Cloudera online documentation resources:

- *Connecting Kafka clients to Data Hub provisioned clusters*
- *Schema Registry*
- *Developing Apache Kafka applications*

Note:

In this section, you build a sample Java application. To do this you must have the following installed on your local computer:

- Java 8 SDK (or later)
- Maven
- Git

Related Information

Connecting Kafka clients to Data Hub provisioned clusters

Schema Registry

Developing Apache Kafka applications

Gather configuration information

To produce data to and consume data from Kafka topics using schemas stored in the Schema Registry, you need to collect some configuration information such as the list of brokers and Schema Registry endpoint. You must also create TLS truststore.

Besides the Machine User name and password that you used in previous examples, you also need the following additional information to configure the client application:

- Broker list

  Because the Kafka client application is a remote application running on your computer, you need to configure it with a list of brokers to which the application can connect.
- Schema Registry endpoint

  You need to provide the application with the Schema Registry endpoint so that the application can store and retrieve schemas from it.
- TLS truststore

  Because this is a remote application connecting to a secure CDP cluster, for which TLS encryption is enabled, you need to provide the client with a truststore that can be used to validate the cluster certificates to establish secure communication channels.

Finding list of brokers

Learn how to find the list of brokers in your Kafka cluster from the Brokers page in the Streams Messaging Manager (SMM) UI.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click Streams Messaging Manager on the services pane to open the SMM web UI.
4. Click the Brokers tab ( ).
5. Take note of the broker endpoints (host and port) listed in the Brokers page.

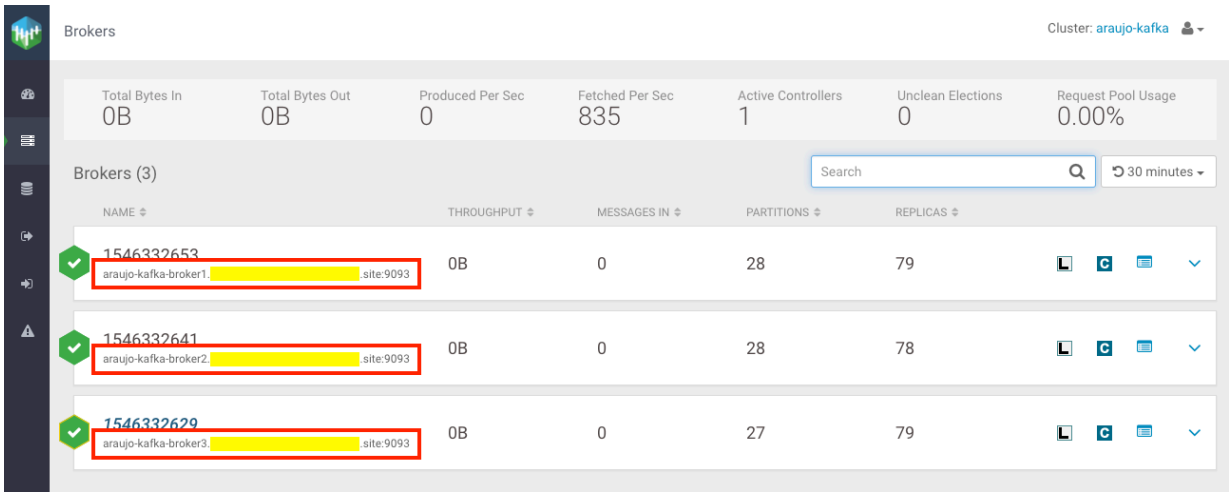

Note: If your cluster contains more than three brokers, you need to select a few to provide to the client. Pick only three of them.

Finding Schema Registry endpoint

You can find the Schema Registry endpoints by selecting the Kafka cluster, you created, from the Management Console.

Before you begin

You must have created a list of brokers in your Kafka cluster.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click the Endpoints tab and make a note of the Schema Registry endpoint.

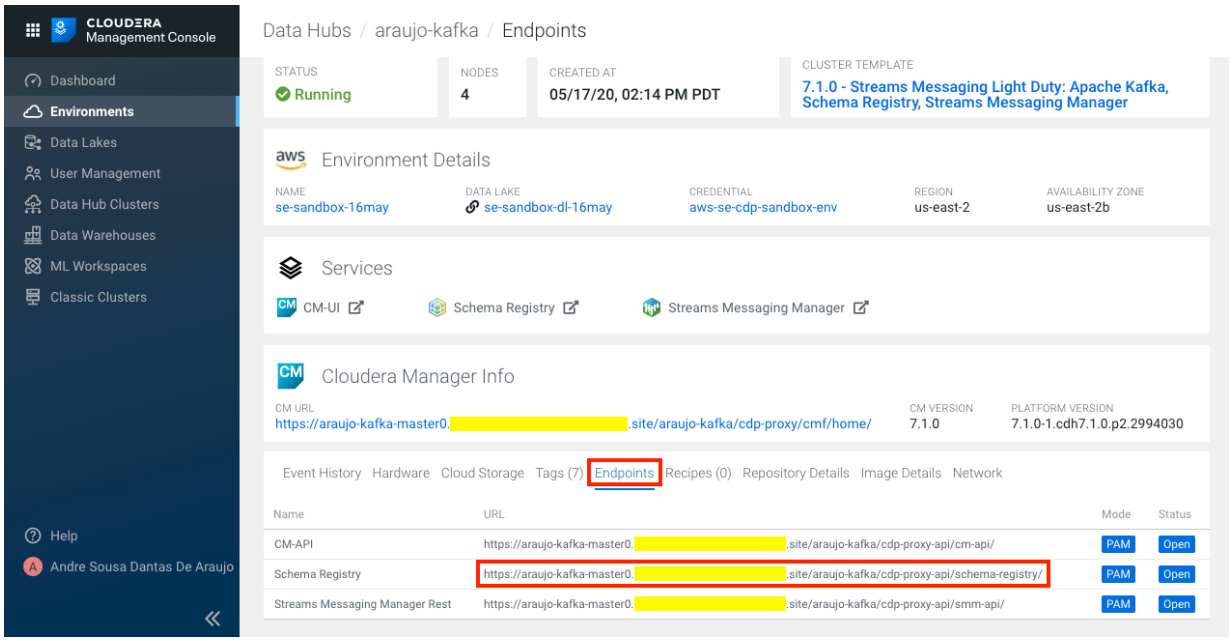

Creating TLS truststore

You can create a TLS truststore that the Kafka client needs to successfully connect to the secure Kafka cluster. You need to go to the environment where your Kafka cluster is running, download the FreeIPA certificate file, and create the TLS truststore by using the certificate file.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. Click the Summary tab, scroll to the FreeIPA section, and then click Actions Get FreeIPA Certificate .

   This downloads the FreeIPA certificate file (<environment_name>.crt) to your computer.
3. Run the following command to create the truststore:

```
keytool \
    -importcert \
    -noprompt \
    -storetype JKS \
    -keystore truststore.jks \
    -storepass changeit \
    -alias freeipa-ca \
```

```
    -file /path/to/ca.crt
```

Defining Schema Registry access policies

You must grant your application certain privileges to use the Schema Registry by creating the appropriate policies in Ranger. Learn how to define access policies and permissions for using Schema Registry.

About this task

In this example, you want to allow the application to create and manage schemas in Schema Registry. You can adjust the permissions granted to the applications in your environment to avoid granting privileges that are not necessary for each use case.

Before you begin

You must have gathered configuration information for using Schema Registry.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. Click the Ranger icon ( ) on the top pane, to open the Ranger web UI.
3. On the Ranger UI, click Access Manager Resource Based Policies .
4. Under the SCHEMA REGISTRY group, select the policy associated to your Schema Registry service.

   To select it, click on the policy name, not on the icons. You should see the list of predefined policies for your Schema Registry.

5. Create a policy to allow access to the schema metadata.
   a) Click Add New Policy, to create a new one, and enter the following details:
      - Policy Name: Machine Data Schema Metadata
      - schema-group: kafka
      - Schema Name: machine-data-avro
      - schema-branch: Click on the schema-branch option and select none
      - Description: Access to machine-data-avro schema metadata
   b) Below the Allow Conditions section, click on the empty Select Users box and select the Machine User you created previously.
   c) Click Add Permissions, select all the permissions and click on the tick mark button.

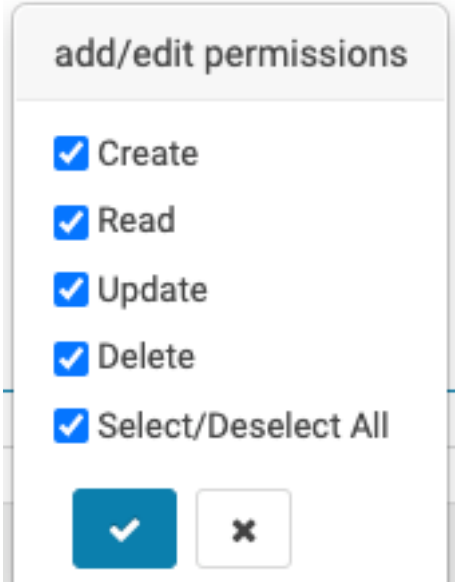

   d) Scroll to the bottom of the page and click Add to save the policy.

6. Create a policy to allow access to the schema versions:
   a) Click Add New Policy, to create a new one, and enter the following details:
      - Policy Name: Machine Data Schema Versions
      - schema-group: kafka
      - Schema Name: machine-data-avro
      - schema-branch: MASTER
      - schema-version: *
      - Description: Access to machine-data-avro schema versions
   b) Below the Allow Conditions section, click on the empty Select Users box and select the Machine User you created previously.
   c) Click Add Permissions, select all the permissions and click on the tick mark button.

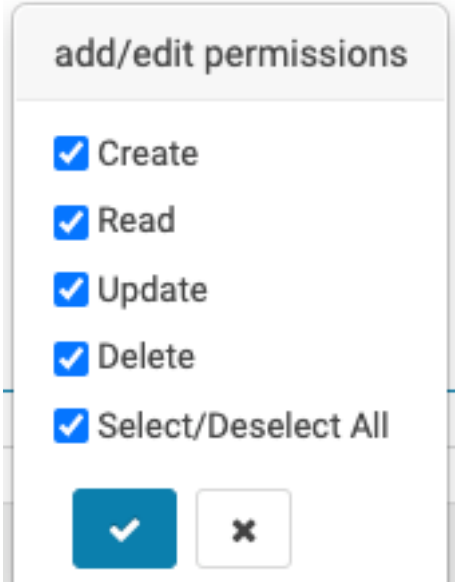

   d) Scroll to the bottom of the page and click Add to save the policy.

Producing data in Avro format

You can run the sample producer application to produce data to a Kafka topic in Avro format. You can send data to a Kafka topic in Avro format. Learn how to store a schema in CDP#s Schema Registry and use a simple Java Kafka client to send and read data using that schema.

Before you begin

You must have defined access policies for Schema Registry.

Procedure

1. On your computer, clone this repository using git and change to the kafka-client-avro directory.

```
git clone https://github.com/asdaraujo/cdp-examples
cd cdp-examples/kafka-client-avro
```

2. Build the client binaries.

```
mvn clean package
```

3. Create a copy of the producer properties template file.

```
cp src/main/resources/producer.properties.template producer.properties
```

4. Edit the producer.properties file and replace the following placeholders with the corresponding values:
   - [***MACHINE_USER_NAME***]

     The Machine User name (prefixed with srv_). Note that this needs to be replaced at two locations in the template.
   - [***MACHINE_USER_PASSWORD***]

     The Machine User's Workload Password. Note that this needs to be replaced at two locations in the template.
   - [***BROKER1***], [***BROKER2***], [***BROKER3***]

     The hostnames of three cluster brokers. Also, ensure that the broker port numbers match the numbers seen in the Streams Messaging Manager (SMM) Brokers page.
   - [***TRUSTSTORE_PATH***]

     The path of the truststore.jks file created previously.
   - [***SCHEMA_REGISTRY_ENDPOINT***]

     The Schema Registry endpoint found previously.
5. Run the producer application with the following command:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataProducer \
  ./producer.properties \
  ./src/main/avro/MachineData.v1.avsc
```

   The producer application (MachineDataProducer class) takes the following arguments:
   - The producer.properties file.
   - An Avro schema definition file that the producer uses to produce messages.

   If the producer is working correctly, you see several messages similar to the following:

```
... INFO  c.c.examples.MachineDataProducer - Successfully produced message
 to partition [machine-data-avro-8], offset [0]
```

6. Leave the SSH session open and the producer running for now.

   Note: If you browse the producer code, you see that there is no need for explicit serialization of the Avro objects in the code. The objects are sent directly to the producer:

```
for (GenericData.Record data :newMachineDataCollector(schema)) {
    ProducerRecord<String, GenericData.Record> producerRecord =newP
roducerRecord<>(topicName, data);
    producer.send(producerRecord, newProducerCallback());
}
```

   The serialization is handled automatically by the KafkaAvroSerializer class that you used to configure the Kafka client's value.serializer property in the properties file:

```
value.serializer=com.hortonworks.registries.schemaregistry.serdes
.avro.kafka.KafkaAvroSerializer
```

Checking schema registration

When the KafkaAvroSerializer is used to produce data to a Kafka topic, the serializer first checks with the configured Schema Registry service to ensure that the schema is compatible with the one registered for that Kafka topic. You can view the registered schema and its the versions in the Schema Registry web UI.

About this task

If the Kafka topic does not have the schema registered yet, the serializer registers the schema on behalf of the client, provided it has permissions to do so. If there is a schema registered for the Kafka topic, the serializer checks if the schema being used by the client is either the same or a compatible version of that schema.

If the schema is the same, the producer uses it for producing messages. If the schema is a new version of the currently registered schema that is compatible with the previous versions, the serializer registers it with a new version number and then uses it for producing messages. If the schema being used by the client is not compatible with the currently registered schema, the serializer issues an exception.

In this example, because this is the first time that you used that topic, there was previously no schema registered for it. If everything works as expected, the schema defined in the MachineData.v1.avsc file is successfully registered in Schema Registry.

Before you begin

You must have produced data in Avro format.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click Schema Registry on the Services pane to open the Schema Registry web UI.

   One schema registered with the name of the topic being used is displayed.
4. Click on the machine-data-avro schema to expand:

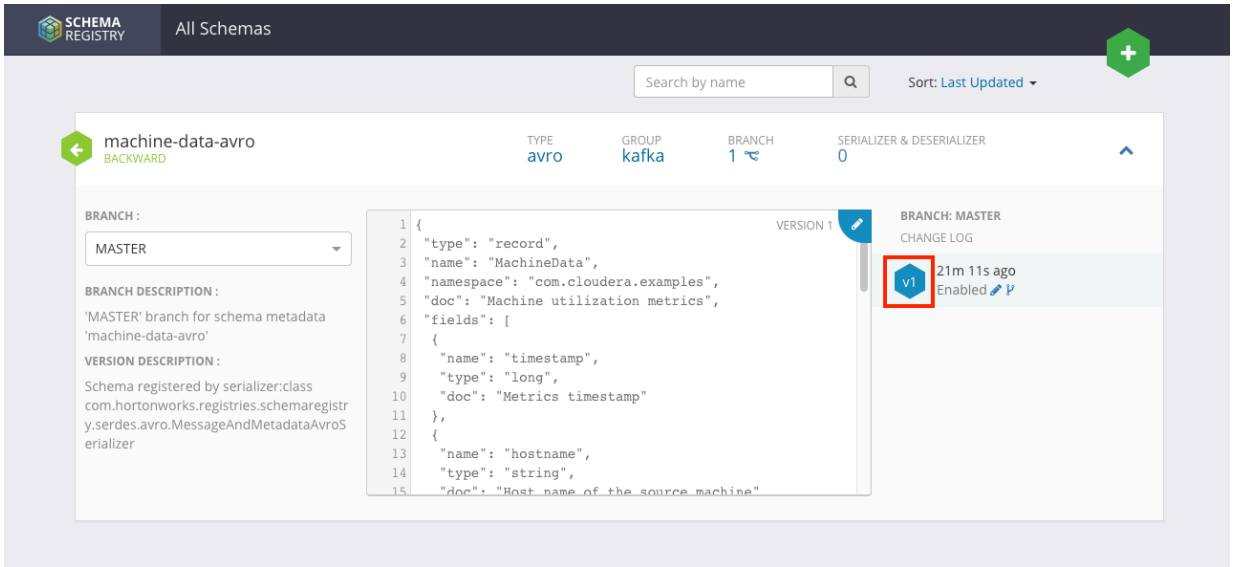

In this page, you can view all the versions of the schema. You only see version v1, which is the version just registered by the producer's KafkaAvroSerializer.

Checking producer activity

You can also use Streams Messaging Manager (SMM) to check the activity generated by the producer on the Kafka cluster. Learn how to check the producer activity in SMM after you send data to a Kafka topic in Avro format.

Before you begin

You must have sent data to a Kafka topic in Avro format and checked the schema registration.

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click Streams Messaging Manager on the Services pane to open the SMM web UI.
4. On the SMM UI, click the Overview tab ( ).
5. Click the Topics dynamic filter, type machine in the Search field, and select the checkbox next to the Name header to include all the filtered topics.

   You should be able to see the machine-data-avro topic and verify that it has some inbound activity (DATA IN > 0).
6. The producer.properties file sets the producer name with the property client.id=producer-java. Find this producer from the list and click on it.

   All the partitions that are receiving data from the producer is displayed.

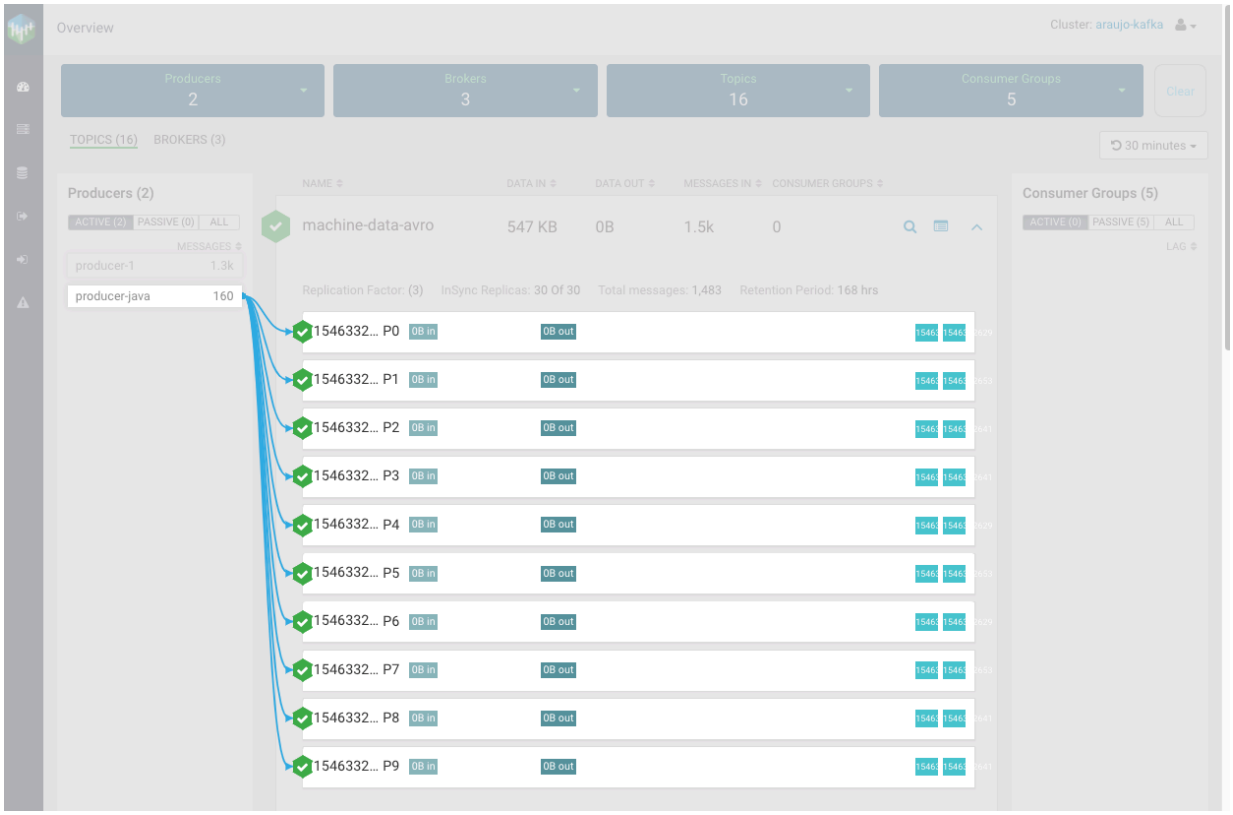

7. Click on an empty part of the page to clear the consumer or producer selection, and then click the profile icon ( ) for the machine-data-avro topic to open the topic page.

8. Click the DATA EXPLORER tab.

   Because the data you are ingesting now is binary data (Avro serialization), you see that the message contents shown by SMM contain a lot of garbage:

Offset	Timestamp	Key	Value
215	Sun, May 17 2020, 8:40:29	null	◆◆◆◆◆\8MacBookPro-AndreAraujo.local,free active◆◆◆ specul◆ inactive◆ throttle wired◆◆◆ prgable◆◆ faults◆jcopy 0fill◆i reactive... show more
216	Sun, May 17 2020, 8:40:39	null	◆◆◆◆◆\8MacBookPro-AndreAraujo.local,free◆◆ active◆ specul◆ inactive◆◆ throttle wired◆◆◆ prgable◆◆ faults◆◆ copy 0fill◆◆ reactive... show more
217	Sun, May 17 2020, 8:40:50	null	◆◆◆◆◆\8MacBookPro-AndreAraujo.local,free◆◆ active◆ specul◆ inactive◆◆ throttle wired◆◆◆ prgable◆◆ faults◆vcopy 0fill◆o reactive ... show more
218	Sun, May 17 2020, 8:41:00	null	◆◆◆◆◆\8MacBookPro-AndreAraujo.local,free◆◆ active◆ specul◆+ inactive◆ throttle wired◆◆◆ prgable◆◆ faults◆{copy 0fill◆n reactive ... show more

   SMM integrates with Schema Registry and is able to decode the binary messages using the schema registered previously by the client.

9. Click the Values drop-down for DESERIALIZER, and select the Avro format.

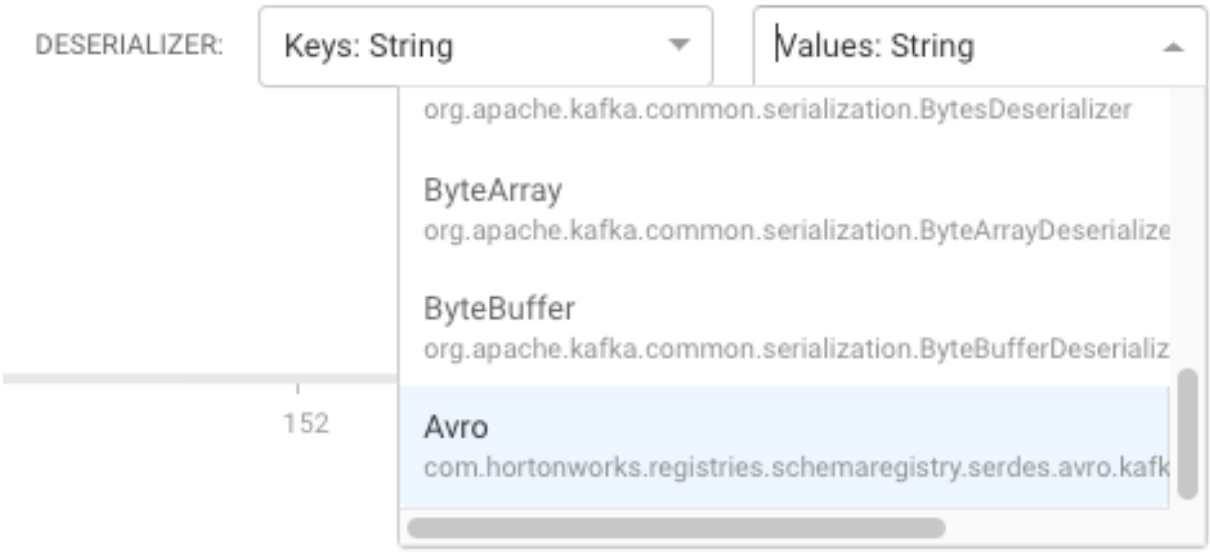

   SMM automatically retrieves the correct schema from the Schema Registry and deserializes the messages shown in SMM, showing them as JSON strings:

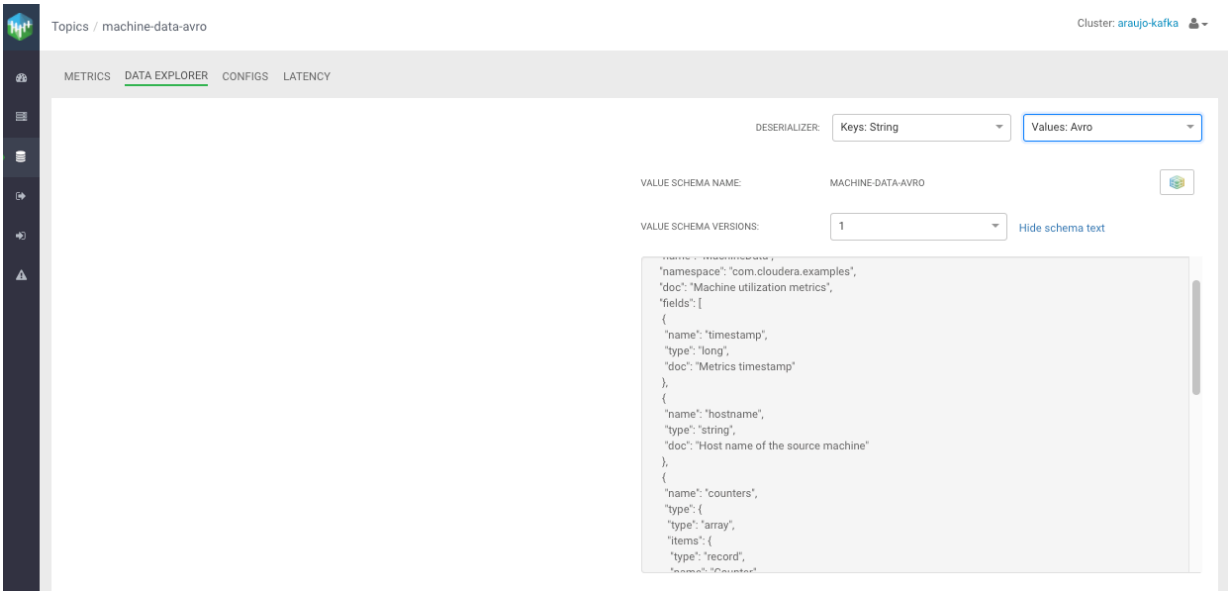

Consuming data from Kafka topics using stored schemas

You can consume data from the machine-data-avro topic where you have produced data in Avro format. You need to create a copy of the consumer properties template file, replace the values of some properties in that file, and run the consumer application.

About this task

While the producer is running on the original terminal, open a new terminal and start consuming data from the mach ine-data-avro topic.

Before you begin

You must have produced data to a Kafka topic in Avro format.

Procedure

1. On the second terminal, ensure that you are on the same directory you used before.

```
cd cdp-examples/kafka-client-avro
```

2. Create a copy of the consumer properties template file.

```
cp src/main/resources/consumer.properties.template consumer.properties
```

3. Edit the consumer.properties file and replace the following placeholders with the respective values:
   - [***MACHINE_USER_NAME***]

     The Machine User name (prefixed with srv_). Note that you need to replace the machine username at two locations in the template.
   - [***MACHINE_USER_PASSWORD***]

     The Machine User's Workload Password. Note that you need to replace the machine user password at two locations in the template.
   - [***BROKER1***], [***BROKER2***], [***BROKER3***]

     The hostnames of three cluster brokers. Also, ensure that the broker port numbers match the numbers seen on the Brokers page in SMM.
   - [***TRUSTSTORE_PATH***]

     The path of the truststore.jks file created in the previous section.
   - [***SCHEMA_REGISTRY_ENDPOINT***]

     The Schema Registry endpoint discovered in a previous section.

   The consumer application (MachineDataConsumer class) takes one argument which is the consumer.properties file.

   Unlike the producer application, the consumer does not require a schema to be provided. Because the schema is already registered in Schema Registry by the producer, the consumer can retrieve the correct schema from the registry.

   Each message send to Kafka by the producer has a few additional bytes that contain the reference to the correct schema version to be used from Schema Registry. These bytes are either prefixed to the message payload or, optionally, stored in the message header.
4. Run the consumer application with the following command:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataConsumer \
```

```
./consumer.properties
```

If the consumer is working correctly, you should see several messages similar to the following:

```
... INFO  c.c.examples.MachineDataConsumer - Consumed 1 records
... INFO  c.c.examples.MachineDataConsumer - Received message: (null, {"
timestamp": ..., }) at partition [machine-data-avro-4], offset [407], with
 headers: [RecordHeader(key = value.schema.version.id, value = [3, 0, 0, 0
, 1])]
```

Note:

- The consumer group being used by the consumer application, configured in the consumer.properties file as group.id=machine-data-1 must match the pattern configured in the Consumer Group policy in Ranger at the beginning of this tutorial. If you chose a consumer group that does not match the policy, the consumer application fails with an authorization error.
- The producer properties template configured the producer to add the schema reference in the message header (store.schema.version.id.in.header=true). You can see this reference in the consumer output with key value.schema.version.id and value [3, 0, 0, 0, 1]. The first byte (3) is the protocolId used by the KafkaAvroSerializer and the next 4 bytes are an integer representing the schema version ID (1). Note that this is not the actual version number, but an internal Schema Registry ID that identifies that schema version object.

5. Go to Streams Messaging Manager (SMM) and verify that now you can see a consumer reading from the topic, besides the producer you had seen before.
6. Stop the producer and consumer.

Monitor end-to-end latency

With Streams Messaging Manager (SMM) you can monitor end-to-end latency for your applications, which is the time taken by a consumer to consume a message that is produced in a Kafka topic.

To find more about end-to-end latency, see *End to end latency overview*.

Learn how to enable end-to-end latency monitoring for your sample Java application.

Related Information

End to end latency overview

Setting up authorization policies

For Streams Messaging Manager (SMM) to show end-to-end latency data for an application, configure the application to generate additional metrics and enable monitoring. This includes sending metadata to internal SMM Kafka topics.

About this task

To do this in a secure Kafka cluster you must grant permissions on those topics through Ranger. Perform the following steps to grant the permissions to the Machine User account you are using:

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. Click the Ranger icon ( ) on the top pane, to open the Ranger web UI.
3. On the Ranger UI, click Access Manager Resource Based Policies .

4. Under the KAFKA group, select the policy of your Kafka cluster.

   To select it, click on the policy name, and not on the icons.
5. Click Add New Policy, to create a new one, and enter the following details:
   - Policy Name: SMM Metrics topics
   - Topic: __smm_consumer_metrics and __smm_producer_metrics
   - Description: SMM topics for end-to-end latency monitoring
6. Under the Allow Conditions section, click on the empty Select Users box and select the Machine User you created previously.
7. Click Add Permissions, select the Publish and Describe permissions, and click the tick mark.

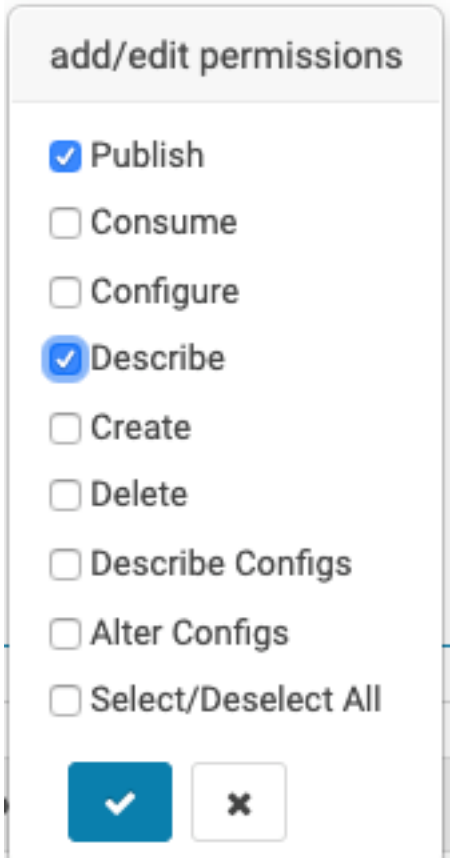

8. Scroll to the bottom of the page and click Add to save the policy.

Enabling end-to-end latency monitoring

Learn how to enable end-to-end latency monitoring of Kafka topics using Streams Messaging Manager (SMM). You need to edit the consumer.property and producer.property files, start the producer in one terminal and the consumer on another, and then open the SMM web UI for monitoring end-to-end latency.

About this task

To enable the application start logging end-to-end latency metrics you use an interceptor for the Kafka client, which can be set through the configuration file.

Perform the following steps to enable the monitoring for the sample Java application and verify the latency in SMM.

Before you begin

You must have set Ranger authorization policies.

Procedure

1. Edit the consumer.properties file and uncomment the following line:

```
interceptor.classes=com.hortonworks.smm.kafka.monitoring.interceptors
.MonitoringConsumerInterceptor
```

2. Edit the producer.properties file and uncomment the following line:

```
interceptor.classes=com.hortonworks.smm.kafka.monitoring.interceptors
.MonitoringProducerInterceptor
```

3. Start the producer on one terminal window with the following command:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataProducer \
  ./producer.properties \
  ./src/main/avro/MachineData.v1.avsc
```

4. Start the consumer on another terminal window with the following command:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataConsumer \
  ./consumer.properties
```

5. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
6. On the Data Hubs tab of your environment, select the Kafka cluster you created.
7. Click Streams Messaging Manager on the Services pane to open the SMM web UI.
8. On the SMM UI, click Overview ( ).
9. Click on the profile icon ( ) for the machine-data-avro topic to open the topic page.

   You see data in the charts Messages Consumed and End-to-end latency. You may have to wait a few seconds until the charts appear.

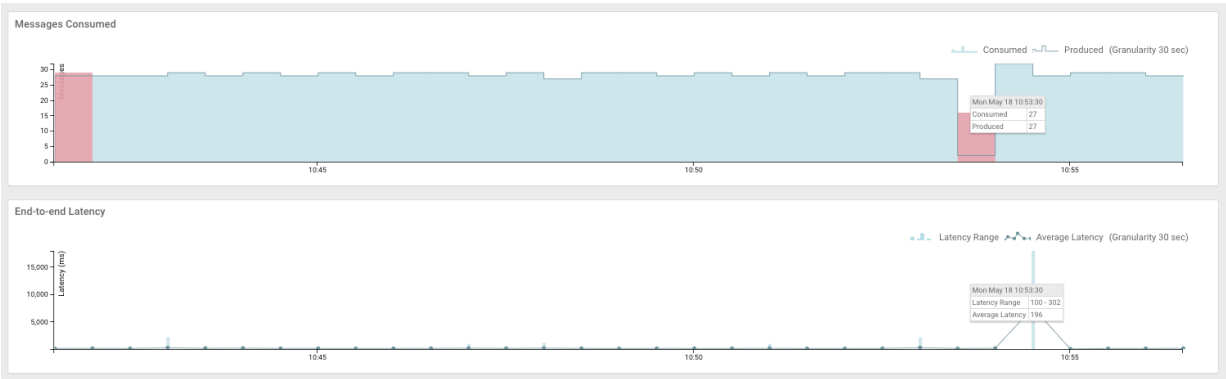

Results

Refresh the page after some time to view the updated data.

Evolve your schema

You can update the schema to keep track of the type of operating system (OS) from the host(s) you are collecting data. After you update the schema, you need to reconfigure the consumer and producer to enable them to use the updated schema.

You have the application producing and consuming data and being monitored successfully, you identified a new requirement for your application and now you want to keep track of the type of OS from the host(s) you are collecting data.

For this, you must add the following field to your schema:

```
{
  "name": "os_type",
  "doc": "OS type of the source machine",
  "type": "string",
  "default": "UNKNOWN"
}
```

Because this is a new field in the schema, you need to specify a default value, as shown above, so that you can maintain backward compatibility, which is being enforced by Schema Registry. If you do not do this, Schema Registry does not allow the new schema to be used. You can check the complete modified schema in the https://github.com/asdaraujo/cdp-examples/blob/master/src/main/avro/MachineData.v2.avsc file.

This section covers the modification of the schema and verification of what changed in Schema Registry and Streams Messaging Manager (SMM).

Reconfiguring the Kafka consumer

After you update a schema stored in the Schema Registry, you must reconfigure the consumer to use the new schema.

About this task

Because the latest schema version is backward compatible, it can be used to deserialize messages produced with the older versions of the schema. Hence, if you first reconfigure the consumer to use the new version of the schema, it can continue handling messages serialized with the old schema. Furthermore, it can also handle the new message format as soon as the producer starts sending them. By reconfiguring the consumers first, you avoid incompatibility between producers and consumers.

Perform the following steps to get the consumer to use the new schema:

Procedure

1. Navigate to Management Console Environments , and select the environment where your Kafka cluster is running.
2. On the Data Hubs tab of your environment, select the Kafka cluster you created.
3. Click Schema Registry on the Services pane to open the Schema Registry web UI.
4. Click on the machine-data-avro schema to expand it, and then click the pencil icon to edit the schema.
5. In the DESCRIPTION field enter a description for this modification: Add os_type field.
6. In the SCHEMA TEXT field, click CLEAR.

   This displays the BROWSE button.

7. Click BROWSE and select the schema definition file for the new version of the schema, located at cdp-examples/kafka-client-avro/src/main/avro/MachineData.v2.avsc, and then click SAVE.

   Schema Registry performs compatibility validations on the new schema and if the outcome is positive, it saves the schema and displays a success message: Schema validated successfully.

   You can view both versions of the schema in Schema Registry.

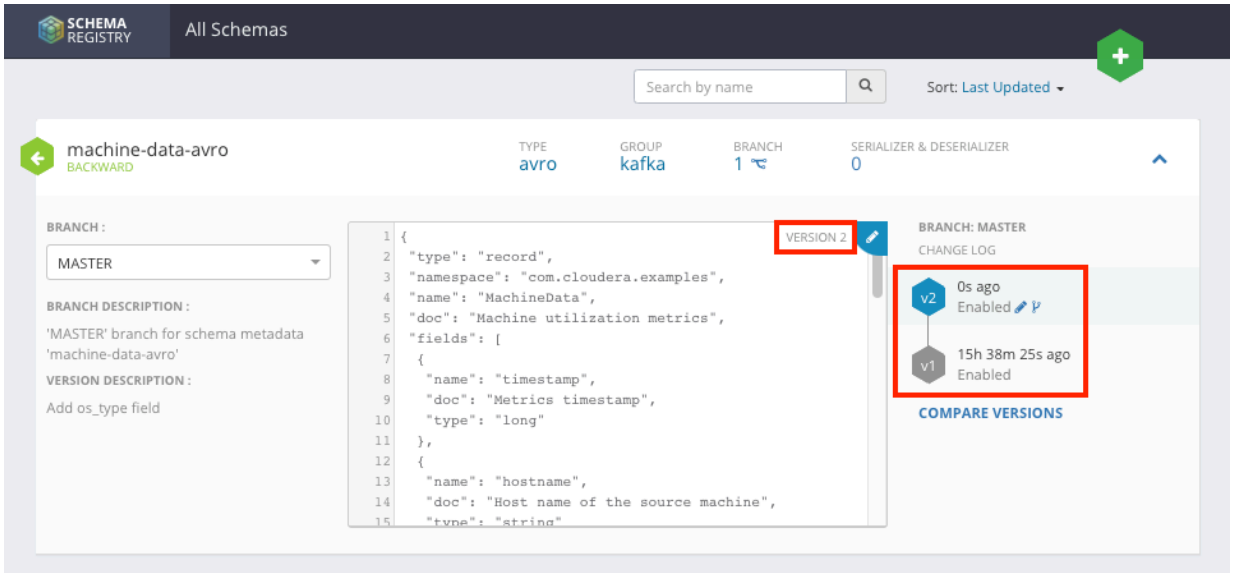

   Now that you have the new version of the schema stored in Schema Registry, you need to reconfigure the consumer to use it.

8. If the consumer application is still running, stop it.
9. Edit the consumer.properties file, uncomment the following line and ensure it refers to the correct version number for the schema, as shown in the previous screenshot.

```
schemaregistry.reader.schema.versions={"machine-data-avro": 2}
```

   This property is a JSON representation of a Map, associating a topic name to the version of the schema that the consumer should use.

10. Start the consumer on one terminal window with the same command used earlier:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataConsumer \
  ./consumer.properties
```

   The consumer displays messages which includes the new field, as shown in the following example:

```
... INFO  c.c.examples.MachineDataConsumer - Received message: (null, {.
.., "os_type": "UNKNOWN", ...}) at partition ...
```

   Note: The value shown for the field is UNKNOWN. This is because the producer is unaware of the new schema and is not producing messages with that field. When the consumer deserializes the message, it uses the default value defined in the schema.

Reconfiguring the Kafka producer

After you update a schema stored in the Schema Registry, you must reconfigure the producer to use the new schema.

About this task

Now that the consumer is already aware of the new schema, you can reconfigure the producer to send messages using the new schema.

Your producer has been created in a way that the schema file argument is optional. If that file is not provided, the producer tries to retrieve the latest version of the schema from Schema Registry with the topic name as the schema key.

Because you already uploaded the new schema to the Schema Registry, you only need to restart the producer without the schema file argument. Perform the following steps to get the producer to use the new schema:

Before you begin

You must have reconfigured the consumer to use the new schema.

Procedure

1. If the producer application is still running, stop it.
2. Start the producer with the following command using the new schema:

```
java \
  -cp ./target/kafka-client-avro-1.0-SNAPSHOT.jar \
  com.cloudera.examples.MachineDataProducer \
  ./producer.properties
```

The producer displays messages with the correct values assigned to the new os_type field, as shown in the following example:

```
... INFO  c.c.examples.MachineDataConsumer - Received message: (null, {.
.., "os_type": "mac", ...}) at partition ...
```

What to do next

After you create your first streams messaging cluster in CDP Public Cloud, you can explore more about how Apache Kafka, Schema Registry, and Streams Messaging Manager (SMM) work in CDP Public Cloud.

For more information on Apache Kafka in CDP Public Cloud, see *Apache Kafka*.

For more information on Schema Registry in CDP Public Cloud, see *Schema Registry*.

For more information on Streams Messaging Manager in CDP Public Cloud, see *Streams Messaging Manager*.

Related Information

Apache Kafka

Schema Registry

Streams Messaging Manager

Cloudera DataFlow for Data Hub 7.3.1

Scaling Streams Messaging Clusters

Date published: 2019-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Scaling Kafka brokers....4
Scaling up Kafka brokers....4
Scaling down Kafka brokers....5
Troubleshooting....6
The downscale operation fails with decommission failed....6

Scaling Kafka Connect....7
Scaling up Kafka Connect....8
Scaling down Kafka Connect....8

Scaling KRaft.... 10

Scaling Kafka brokers

The Kafka brokers provisioned with the Light and Heavy duty versions of the Streams Messaging cluster definitions can be scaled. This is done by adding or removing nodes from the host groups containing Kafka brokers. During a scaling operation Cruise Control automatically rebalances partitions on the cluster. Before scaling Kafka brokers, Cloudera recommends that you review these notes regarding scaling operations.

Version limitations

Scaling brokers is only supported on:

- Clusters newly provisioned with Cloudera Runtime 7.2.12 or higher.
- Clusters upgraded from Cloudera Runtime 7.2.12 or higher to a newer version.

Not all Kafka host groups are scalable

Streams Messaging clusters running Cloudera Runtime 7.2.12 or higher have two host groups of Kafka broker nodes. These are the **Core_broker** and **Broker** host groups. During an upscale or downscale operation, new broker nodes are added to or removed from the **Broker** host group. The **Core_broker** group contains a core set of brokers and is not scalable. For more information regarding the Streams Messaging cluster templates, see *Streams Messaging cluster layout*.

Downscale operations ensure that the cluster remains healthy

During a downscale operation there are a number of automated safety checks to ensure that the cluster stays healthy at all times and no data is lost when brokers are decommissioned. For example:

- Downscale operations are only carried out if sufficient resources are available. Otherwise they fail.
- The partitions of a decommissioned broker are automatically moved to other brokers.
- A broker is only decommissioned and removed from the broker's host group if there is no load on that broker.

Related Information

Streams Messaging cluster layout

Scaling up Kafka brokers

Learn how you can upscale Kafka brokers in a Streams Messaging cluster.

About this task

The Kafka brokers in a Streams Messaging cluster can be upscaled. During an upscale operation, broker nodes are added to the **Broker** host group. Newly commissioned brokers are automatically detected by Cruise Control, a rebalance is triggered once they become available, and data is moved to the new brokers.

Before you begin

- Ensure that you have reviewed Scaling Kafka broker nodes.
- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that the Kafka brokers are commissioned and running.
- Ensure that the Cruise Control service is running.

Procedure

1. Log in to the Cloudera web interface.
2. Go to Management Console Data Hub Clusters .

3. Find and select the cluster you want to scale.
4. Click Actions Resize.
5. Select the **Broker** host group.
6. Set the desired size of the host group.

   You can enter a number or use the plus and minus buttons to adjust the number of nodes.
7. Click Resize.
8. Wait until the new nodes become available.
9. Access the Cloudera Manager instance managing the cluster.
10. Restart all services that have stale configurations.

Results

The cluster is upscaled. The number of Kafka broker nodes available in the **Broker** host group is equal to the configured number of nodes. Data is automatically moved to the new brokers by Cruise Control.

Scaling down Kafka brokers

Learn how you can downscale Kafka brokers in a Streams Messaging cluster.

About this task

The Kafka brokers in a Streams Messaging cluster can be downscaled. During a downscale operation, broker nodes are decommissioned from the **Broker** host group. Data and partitions of the decommissioned nodes are automatically moved to other nodes.

There are two methods that can be used to downscale a cluster. You can either have Cloudera Data Hub choose which nodes to decommission or manually specify which nodes should be decommissioned. The following steps walk you through both methods.

Before you begin

- Ensure that you have reviewed Scaling Kafka broker nodes.
- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that the Kafka brokers are commissioned and running.
- Ensure that the Cruise Control service is running.
- Using the Cruise Control REST API, verify that Cruise Control is reporting that all partitions are healthy.

  Important: An access level (admin, user or viewer) must be set for the user calling the API endpoint in Cruise Control. For more information, see Cruise Control REST API endpoints.

  This can be done by calling the state API endpoint and ensuring the following in the response:

  - numValidPartitions is equal to numTotalPartitions
  - monitoringCoveragePct is 100.0

  API call example:

  ```
  https://[***MY-CLOUDERA-DATA-HUB-CLUSTER.COM***]/cdp-proxy-api/cruise-co
  ntrol/kafkacruisecontrol/state?json=true
  ```

  You can find the API base URL on the Endpoints tab of the **Cluster Details** page. Management Console Data Hub Clusters *[***YOUR CLUSTER***]* Endpoints .
- Do not restart Kafka or Cruise Control during a downscale operation.
- Do not create new partitions during a downscale operation.

Procedure

1. Log in to the Cloudera web interface.
2. Go to Management Console Data Hub Clusters .
3. Find and select the cluster you want to scale.
4. Decommission nodes.

   You can either let Cloudera Data Hub choose which nodes to decommission or you can select the nodes that you want to have decommissioned.

   For Let Data Hub choose which nodes to decommission

   a. Click ActionsResize.
   b. Select the Broker host group.
   c. Set the desired size of the host group.

      You can enter a number or use the plus and minus buttons to adjust the number of nodes.
   d. Click Resize.

      Note: Cloudera Data Hub chooses which nodes to decommission based on how suitable it is to remove a specific node. As a result, the nodes selected by Cloudera Data Hub might not be the nodes that have the highest node ID.

   For Manually specify which nodes to decommission

   a. Go to Hardware.
   b. Scroll down to the **Broker** host group.
   c. Select the nodes you want to remove.
   d. Click the Delete (trash bin) icon.
   e. Click Yes to confirm deletion.
5. Access the Cloudera Manager instance managing the cluster.
6. Restart all services that have stale configurations.

Results

The cluster is downscaled. The number of Kafka broker nodes available in the **Broker** host group is equal to the configured number of nodes. Partitions are automatically moved from the decommissioned brokers. Once no load is left on the broker, the broker is fully decommissioned and removed from the **Broker** host group.

Troubleshooting

Learn about the common issues that you might encounter when scaling Kafka broker nodes in Streams Messaging Clusters.

The downscale operation fails with decommission failed

Condition

The downscale operation fails with the Decommission failed error.

Cause

This issue can be caused by either of the following:

- The broker is not available.
- Cruise Control is reporting unhealthy partitions

- A manually initiated rebalance is running.
- An insufficient amount of brokers would remain after the downscale operation to host all replicas.

  This only affects deployments where there is a topic with at least four-way replication set up.

Solution

Procedure

1. Back up the data disks of the failed node.

   This can be done by creating a backup of /hadoopfs/fs*[***DATA VOL NUMBER***]*/kafka.
2. Access the Cloudera Manager instance managing the nodes that you want to decommission.
3. Select the Kafka service and go to Instances.
4. Select the Kafka broker you want to decommission.
5. Click Actions for SelectedRecommission and start.
6. Wait until the process is finished and the cluster is in a healthy (green) state.
7. In Management Console, navigate to the cluster containing the node.
8. Go to Hardware.
9. Select the node and click the Repair (wrench) icon next to the host group.

   Important: Do not select the Repair action from the upper right hand Actions dropdown. This action attempts to repair all unhealthy nodes in the selected host group. Under certain circumstances, the node that you want to repair might not be recognized as unhealthy. As a result, the repair operation might not be carried out on the affected node. Ensure that you initiate a repair on the affected nodes using the Repair (wrench) icon.

Scaling Kafka Connect

Kafka Connect provisioned with the Light and Heavy duty versions of the Streams Messaging cluster definitions can be scaled. However, there are differences between how scaling is performed in each of the templates. Cloudera recommends that you review the following notes and recommendations before you scale Kafka Connect.

Light Duty

In the Light duty cluster template, Kafka Connect roles are deployed in the **Broker** and **Core_broker** host groups. The **Core_boker** host group has a default instance count of 3, the **Broker** host group has a default instance count of 0. This means that by default all Light Duty clusters will a have a minimum of three Kafka Connect roles. To scale Kafka Connect in a Light Duty cluster, you resize the **Broker** host group. The **Core_broker** host group is not scalable. When scaling the **Broker** host group all other service roles are also scaled. Scaling Kafka Connect on its own in Light Duty clusters is not possible.

Heavy Duty

In the Heavy Duty cluster template, Kafka Connect roles are deployed in a dedicated host group named **Connect**. To scale Kafka Connect in a Heavy Duty cluster, you resize the **Connect** host group. The instance count of this host group is set to 0 by default. This means that unless specifically provisioned, no Kafka Connect roles are deployed in Heavy Duty clusters.

For more information regarding the Streams Messaging cluster templates, see *Streams Messaging cluster layout*.

Related Information

Streams Messaging cluster layout

Scaling up Kafka Connect

Learn how you can upscale Kafka Connect in a Streams Messaging cluster.

About this task

Kafka Connect in a Streams Messaging cluster can be upscaled. The upscale operation differs depending on which Streams Messaging cluster template you are using. In the Light Duty template Kafka Connect is upscaled by adding nodes to the **Broker** host group. In the Heavy Duty template, Kafka Connect is upscaled by adding nodes to the **Connect** host group.

Before you begin

- Ensure that you have reviewed Scaling Kafka Connect on page 7.
- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that the Kafka brokers are commissioned and running.
- If using the Light Duty cluster template, ensure that the Cruise Control service is running.

  This is required because in the Light Duty template, Kafka Connect is deployed on the same node as Kafka brokers. While Kafka Connect does not require Cruise Control for scaling, Kafka brokers do.

Procedure

1. Log in to the Cloudera web interface.
2. Go to Management Console Data Hub Clusters .
3. Find and select the cluster you want to scale.
4. Click Actions Resize.
5. Select the host group you want to scale.

   For Light Duty clusters, select the **Broker** host group. For Heavy Duty clusters, select the **Connect** host group.
6. Set the desired size of the host group.

   You can enter a number or use the plus and minus buttons to adjust the number of nodes.
7. Click Resize.
8. Wait until the new nodes become available.
9. Access the Cloudera Manager instance managing the cluster.
10. Restart all services that have stale configurations.

Results

The cluster is upscaled. In Light Duty clusters, the number of nodes available in the **Broker** host group is equal to the configured number of nodes. In Heavy Duty clusters, the number of nodes available in the **Connect** host group is equal to the configured number of nodes.

Scaling down Kafka Connect

Learn how you can downscale Kafka Connect in a Streams Messaging cluster.

About this task

Kafka Connect in a Streams Messaging cluster can be downscaled. The downscale operation differs depending on which Streams Messaging cluster template you are using. In the Light Duty template, Kafka Connect is downscaled by decommissioning nodes from the **Broker** host group. In the Heavy Duty template, Kafka Connect is downscaled by decommissioning nodes from the **Connect** host group.

There are two methods that can be used to downscale a cluster. You can either have Cloudera Data Hub choose which nodes to decommission or manually specify which nodes should be decommissioned. The following steps walk you through both methods.

Before you begin

- Ensure that you have reviewed Scaling Kafka Connect on page 7.
- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that the Kafka brokers are commissioned and running.
- Do not restart Kafka or Cruise Control during a downscale operation.
- Do not create new partitions during a downscale operation.
- If using the Light Duty cluster template ensure the following:
  - Ensure that the Cruise Control Service is running.
  - Using the Cruise Control REST API, verify that Cruise Control is reporting that all partitions are healthy.

    Important: An access level (admin, user or viewer) must be set for the user calling the API endpoint in Cruise Control. For more information, see Cruise Control REST API endpoints.

    This can be done by calling the state API endpoint and ensuring the following in the response:

    - numValidPartitions is equal to numTotalPartitions
    - monitoringCoveragePct is 100.0

    API call example:

    ```
    https://[***MY-CLOUDERA-DATA-HUB-CLUSTER.COM***]/cdp-proxy-api/cruise-co
    ntrol/kafkacruisecontrol/state?json=true
    ```

    You can find the API base URL on the Endpoints tab of the **Cluster Details** page. Management Console Data Hub Clusters *[***YOUR CLUSTER***]* Endpoints .

  This is required because in the Light Duty template, Kafka Connect is deployed on the same node as Kafka brokers. While Kafka Connect does not require Cruise Control for scaling, Kafka brokers do.

Procedure

1. Log in to the Cloudera web interface.
2. Go to Management Console Data Hub Clusters .
3. Find and select the cluster you want to scale.

4. Decommission nodes.

   You can either let Cloudera Data Hub choose which nodes to decommission or you can select the nodes that you want to have decommissioned.

   For Let Data Hub choose which nodes to decommission

   a. Click ActionsResize.
   b. Select the host group you want to scale.

      For Light Duty clusters, select the **Broker** host group. For Heavy Duty clusters, select the **Connect** host group.
   c. Set the desired size of the host group.

      You can enter a number or use the plus and minus buttons to adjust the number of nodes.
   d. Click Resize.

      Note: Cloudera Data Hub chooses which nodes to decommission based on how suitable it is to remove a specific node. As a result, the nodes selected by Cloudera Data Hub might not be the nodes that have the highest node ID.

   For Manually specify which nodes to decommission

   a. Go to Hardware.
   b. Scroll down to the appropriate host group.

      For Light Duty clusters, go to the **Broker** host group. For Heavy Duty clusters, go to the **Connect** host group.
   c. Select the nodes you want to remove.
   d. Click the Delete (trash bin) icon.
   e. Click Yes to confirm deletion.

5. Access the Cloudera Manager instance managing the cluster.
6. Restart all services that have stale configurations.

Results

The cluster is downscaled. In Light Duty clusters, the number of nodes available in the **Broker** host group is equal to the configured number of nodes. In Heavy Duty clusters, the number of nodes available in the **Connect** host group is equal to the configured number of nodes. Connectors and tasks assigned to the decommissioned nodes are redistributed among remaining connect workers

Scaling KRaft

Learn how to scale KRaft nodes (KRaft Controllers) in a Streams Messaging cluster.

About this task

Note: Kafka KRaft is available in this version of Cloudera but is not ready for production deployment. Cloudera encourages you to explore this technical preview feature in non-production environments and provide feedback on your experiences through the Cloudera Community Forums. For more information regarding KRaft limitations and unsupported features, see Unsupported Streams Messaging features.

Streams Messaging clusters that you provision with any of the default Streams Messaging cluster definitions may include optional KRaft nodes. KRaft nodes house the KRaft Controller roles. If your cluster includes KRaft nodes, the Kafka service on the cluster is running KRaft mode and uses KRaft for metadata management instead of ZooKeeper. The number of KRaft nodes in the cluster can be scaled. This is done by adding nodes to or removing nodes from the **KRaft Nodes** host group using the Resize action in Cloudera Data Hub.

Before you begin

- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that the Kafka brokers and KRaft Controllers are commissioned and running.

Important: The number of KRaft nodes in a cluster must be uneven. If you upscale or downscale the number of nodes to an even number, the scaling operation will fail.

Procedure

1. Log in to the Cloudera web interface.
2. Go to Management Console Data Hub Clusters .
3. Find and select the cluster you want to scale.
4. Click Actions Resize .
5. Select the kraft host group.
6. Set the desired size of the host group.

   You can enter a number or use the plus and minus buttons to adjust the number of nodes. Ensure that you set an odd number of nodes.
7. If you are upscaling the cluster, select an Adjustment type based on your requirements.
8. If you are downscaling the cluster, select Force downscale to forcibly remove hosts.
9. Click Resize.
10. Wait until the new nodes become available.
11. Access the Cloudera Manager instance managing the cluster.
12. Restart all services that have stale configurations.

Results

The cluster is resized.

Cloudera DataFlow for Data Hub 7.3.1

Changing Java version of Flow Management clusters

Date published: 2019-12-16
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Changing the Java version of Flow Management Data Hub clusters.....4

Changing the Java version of Flow Management Data Hub clusters

Java 8 and 11 are installed on Cloudera Data Hub nodes in Cloudera Public Cloud by default. When you create a new cluster, Java 8 is used as the default JDK. However, it is possible to switch to Java 11 after provisioning a new cluster or after upgrading your cluster to version 7.2.17.

About this task

Procedure

1. Open the Cloudera Manager UI and click the Clusters tab in the left-hand navigation.
2. Click NiFi in the list of services to display the NiFi service page.
3. Click Configuration and update the nifi.jdk.home property.

   Paths:

   - Java 8: /usr/lib/jvm/java
   - Java 11: /usr/lib/jvm/java-11

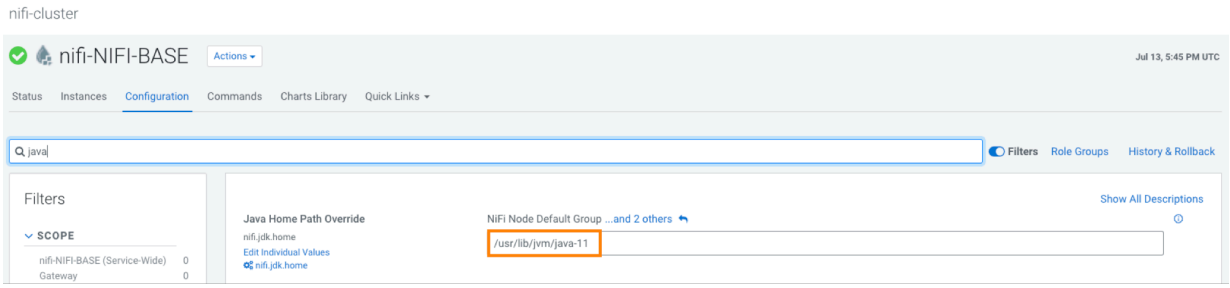

4. Repeat the same steps for the NiFi Registry service as well.

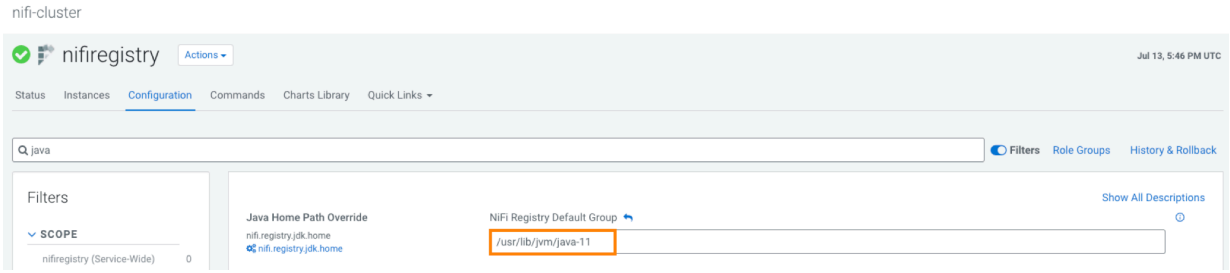

5. Restart NiFi and NiFi Registry.

6. When the services are restarted, you can confirm the Java version used by NiFi.

a) Open the NiFi UI and click the Global menu in the top right corner.

b) Click Cluster Versions

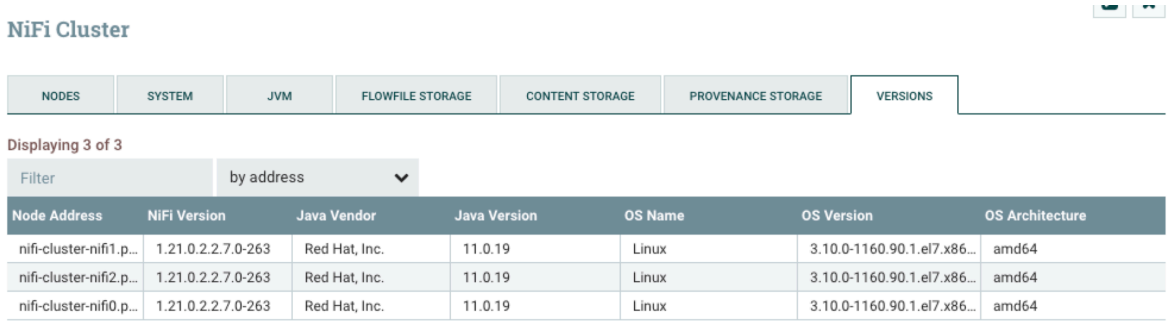

Node Address	NiFi Version	Java Vendor	Java Version	OS Name	OS Version	OS Architecture
nifi-cluster-nifi1.p...	1.21.0.2.2.7.0-263	Red Hat, Inc.	11.0.19	Linux	3.10.0-1160.90.1.el7.x86...	amd64
nifi-cluster-nifi2.p...	1.21.0.2.2.7.0-263	Red Hat, Inc.	11.0.19	Linux	3.10.0-1160.90.1.el7.x86...	amd64
nifi-cluster-nifi0.p...	1.21.0.2.2.7.0-263	Red Hat, Inc.	11.0.19	Linux	3.10.0-1160.90.1.el7.x86...	amd64

Cloudera Flow Management 2.1.6

Using Apache NiFi

Date published: 2019-06-26
Date modified: 2022-12-13

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Apache NiFi.....4

Using Apache NiFi

For more information, see the Apache NiFi User Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi documentation

Cloudera Flow Management 2.1.6

Using Apache NiFi Registry

Date published: 2019-06-26
Date modified: 2022-12-13

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Apache NiFi Registry..... 4

Using Apache NiFi Registry

For more information, see the Apache NiFi Registry User Guide.

Important:

The Apache NiFi Registry documentation is available both online and within the Registry NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi Registry version. If you are using an earlier NiFi Registryversion, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi Registry version directly from the NiFi Registry UI:

1. Launch the NiFi Registry UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi Registry documentation

Cloudera Flow Management 2.1.6

Using the Apache NiFi Toolkit

Date published: 2019-06-26
Date modified: 2022-12-13

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Apache NiFi Toolkit..... 4

Using Apache NiFi Toolkit

For more information, see the Apache NiFi Toolkit Guide.

Important:

The Apache NiFi documentation is available both online and within the NiFi User Interface (UI). The online documentation consistently reflects the latest NiFi version. If you are using an earlier NiFi version, content differences may exist between the latest documentation and the version you would need. You can access the documentation specific to your NiFi version directly from the NiFi UI:

1. Launch the NiFi UI.
2. From the Global Menu, select Help Documentation .

Related Information

Apache NiFi documentation

Cloudera Flow Management 7.3.1

Using DataFlow Catalog Registry Client

Date published: 2019-06-26
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a Machine User **4**

Adding a new Registry Client **6**

Checking out a ReadyFlow **8**

Checking out your flows **9**

Versioning a flow in the Catalog **10**

Creating a Machine User

Learn how to create a Machine User with the proper roles for the DataFlow Catalog and generate credentials.

Before you begin

You are a Cloudera customer for Cloudera Public Cloud.

Procedure

1. Log in to the Cloudera Public Cloud web interface.
2. Navigate to Cloudera Management Console User Management to create a Machine User that will be used by NiFi to interact with the DataFlow Catalog.
3. Select Create Machine User from the Actions menu on the Users tab.

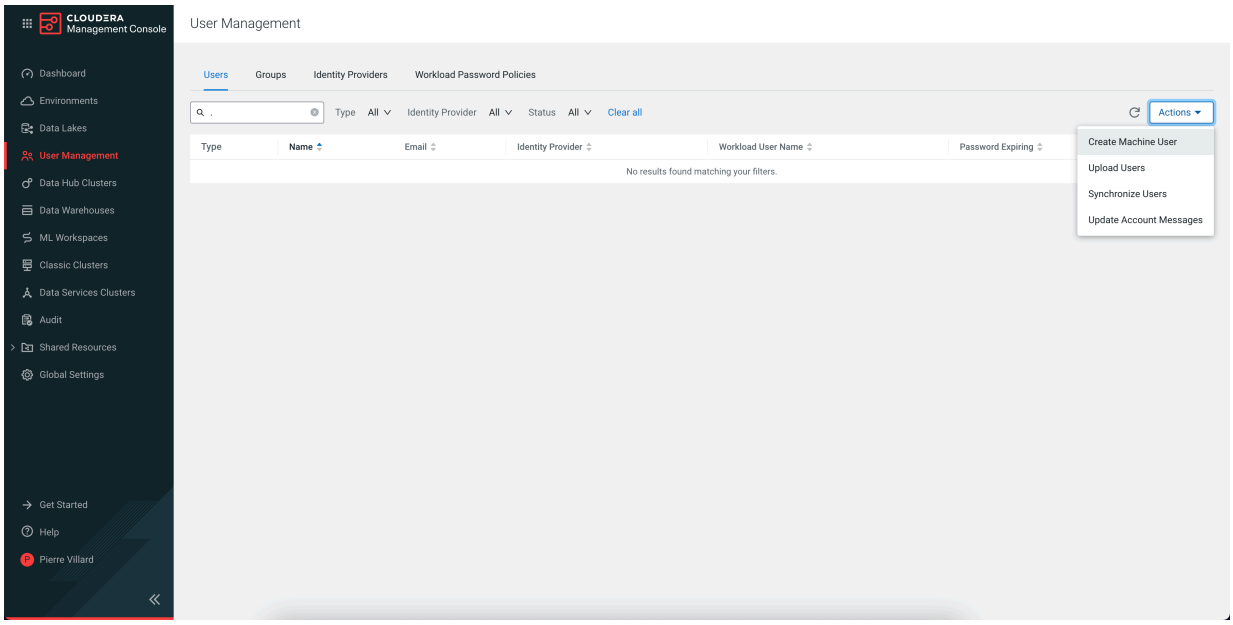

4. Give the user a meaningful name and give it the proper roles:

DFCatalogViewer

It gives read-only access to the DataFlow Catalog. This is enough if you just want to check out existing flows from the Catalog.

DFCatalogAdmin

It gives read-write access to the DataFlow Catalog. This would allow you to version and control flows from NiFi into the Catalog, and to check out existing flows from the Catalog.

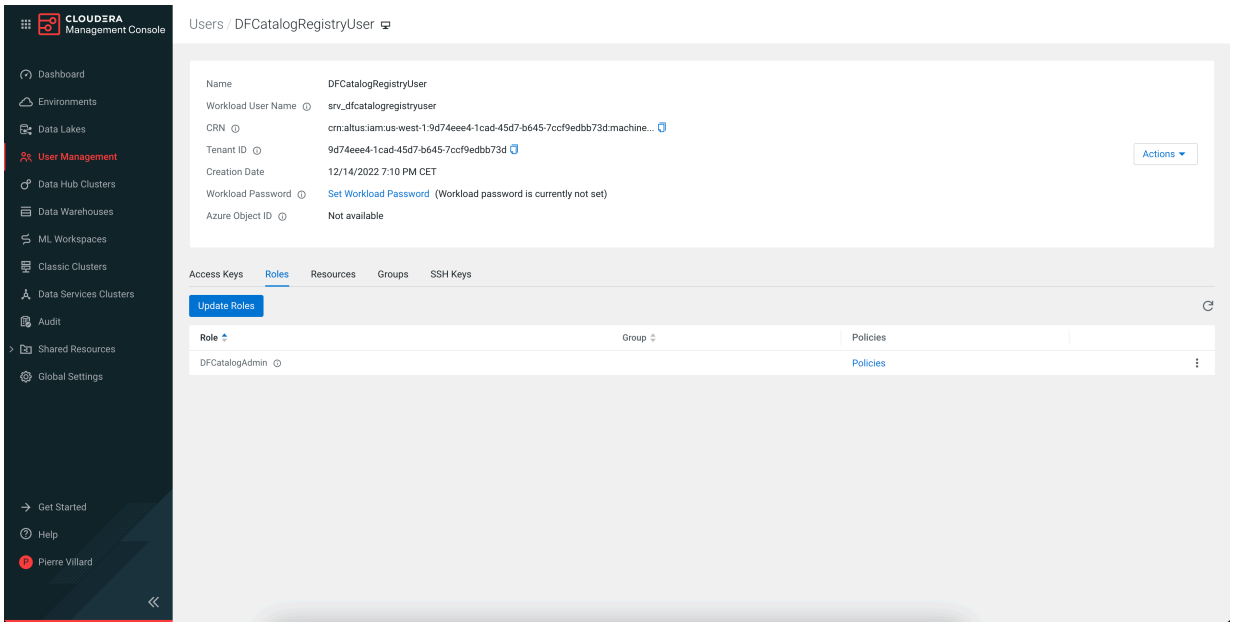

5. Go to the Access Keys tab and generate credentials for your user.

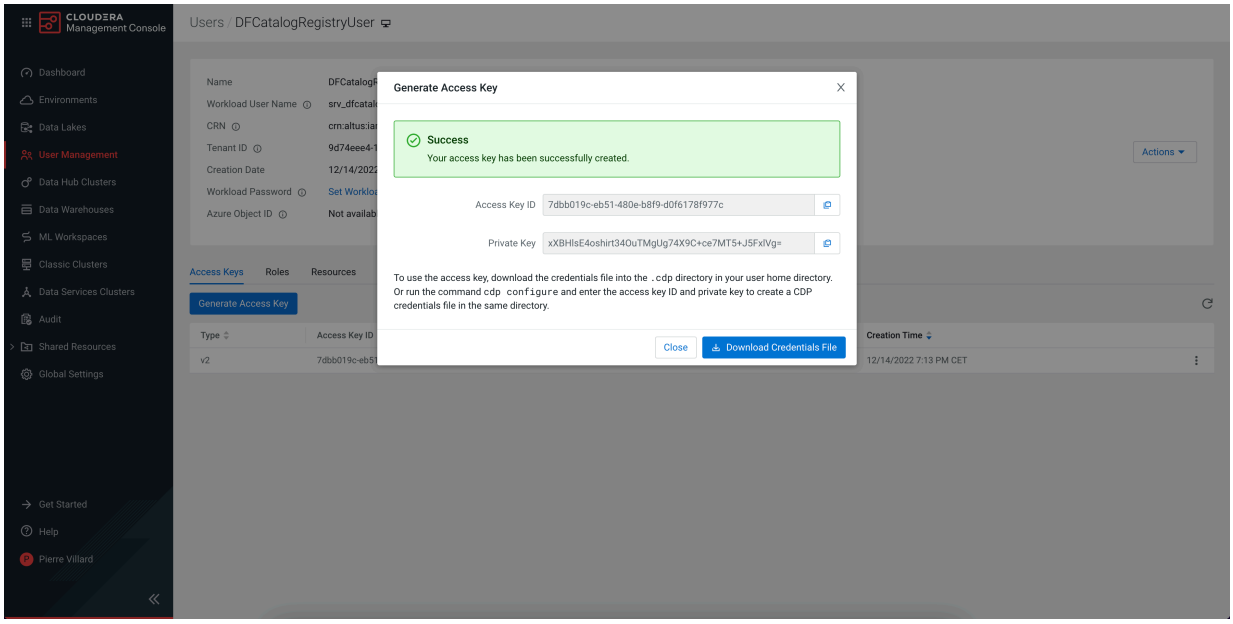

Note: Make sure to keep track of the Access Key ID and the Private Key as you will need these credentials to configure the Registry Client in NiFi.

What to do next

You can now create and configure the DF Catalog client with the available credentials.

Adding a new Registry Client

Learn how to add and configure the DF Catalog client as a new Registry Client.

Procedure

1. Open the NiFi UI.
2. Click the Global menu and select Controller Settings.

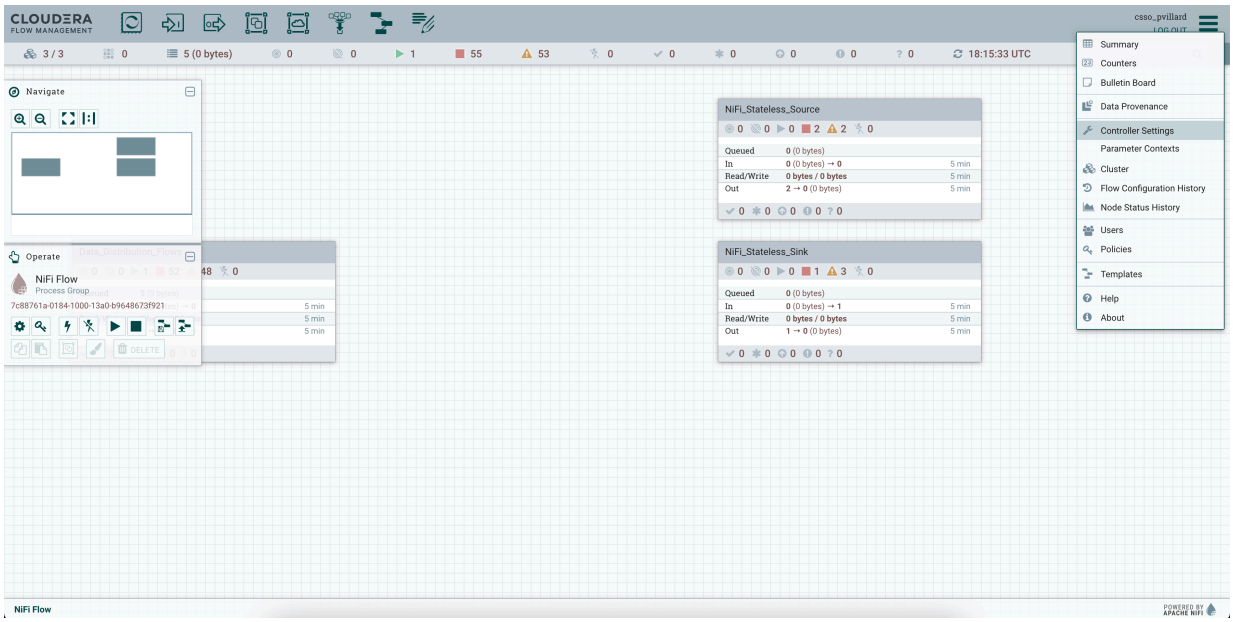

3. Switch to the Registry Clients tab.

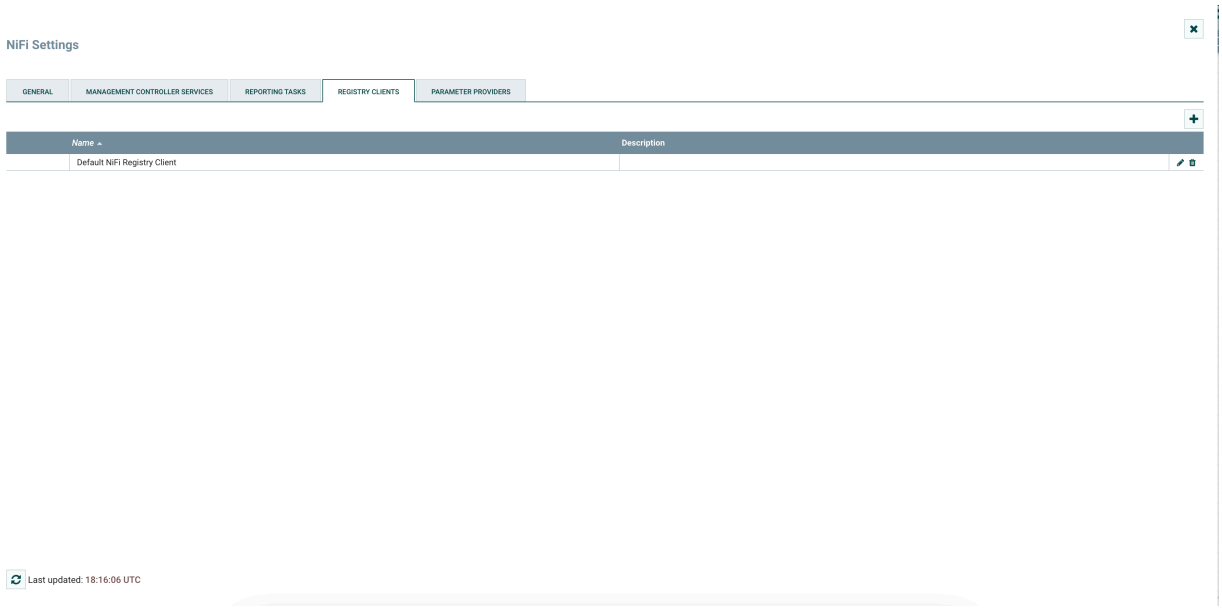

4. Add a new client.
   a) Select the type and give it a name.

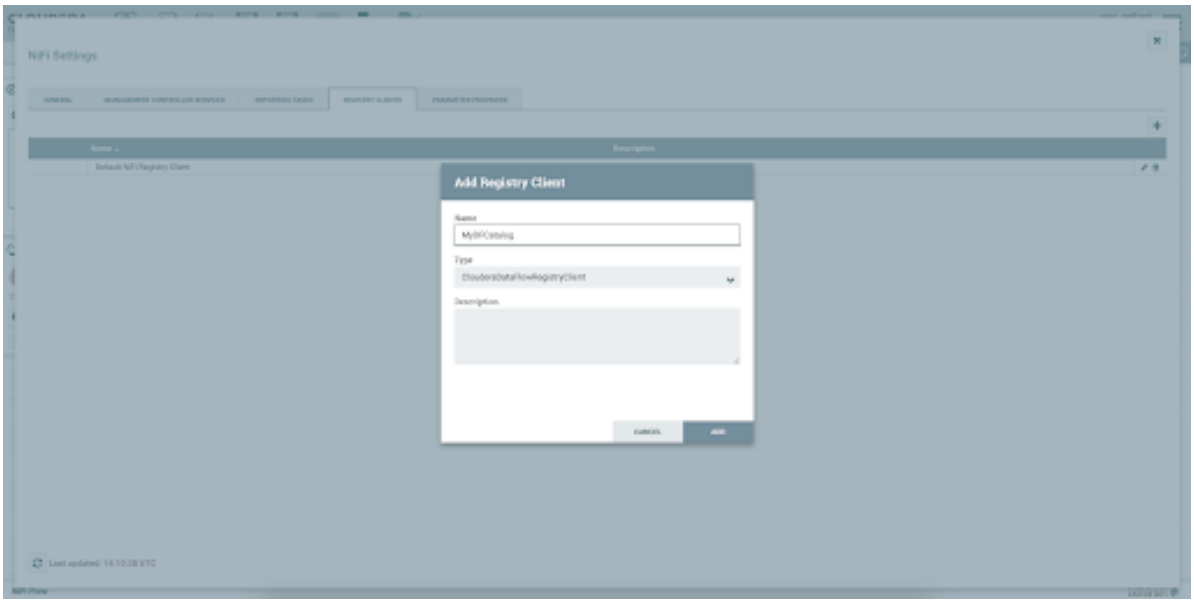

   b) Configure the client by providing the information related to your Machine User.

Note: You will have to create a StandardWebClientServiceProvider controller service and start it with its default configuration.

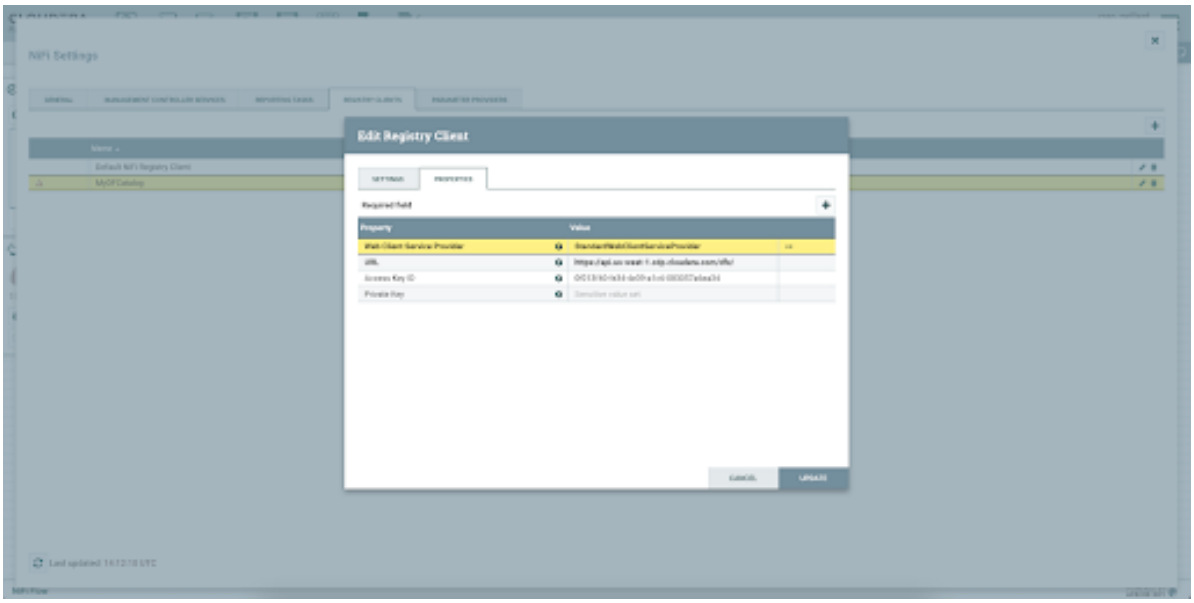

Results

When properly configured, you are ready to use the DataFlow Catalog to either version control your flows or check out flow definitions from the Catalog into NiFi.

What to do next

You can now start using the new client to version or check out flows from the DataFlow Catalog.

Checking out a ReadyFlow

Learn how to check out a ReadyFlow from the DataFlow Catalog.

About this task

ReadyFlows are predefined, out-of-the-box data flows, which can be deployed by providing a small set of required parameters. They are read-only by default so you would not be able to commit any change made to them. With the Registry Client in NiFi connecting to Cloudera DataFlow Catalog, you can check out a ReadyFlow from the Catalog and edit it as a custom flow.

Procedure

1. Drag and drop a process group onto the canvas and click Import from Registry.

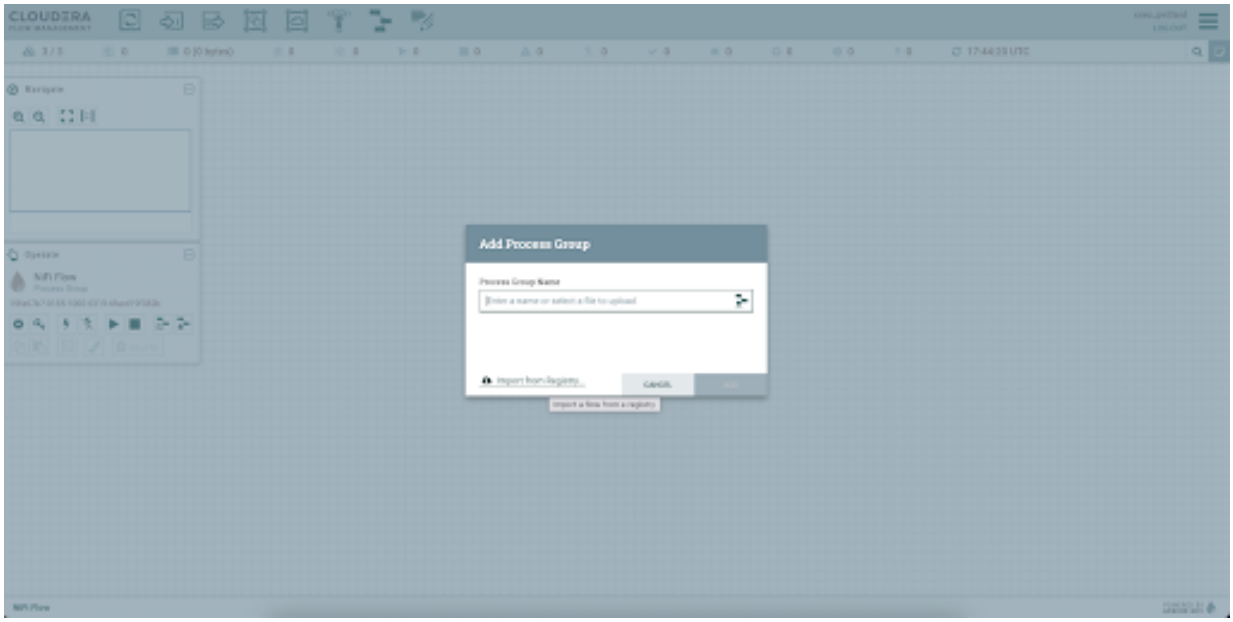

2. In the drop-down list, select the Registry Client configured to access your DataFlow Catalog.

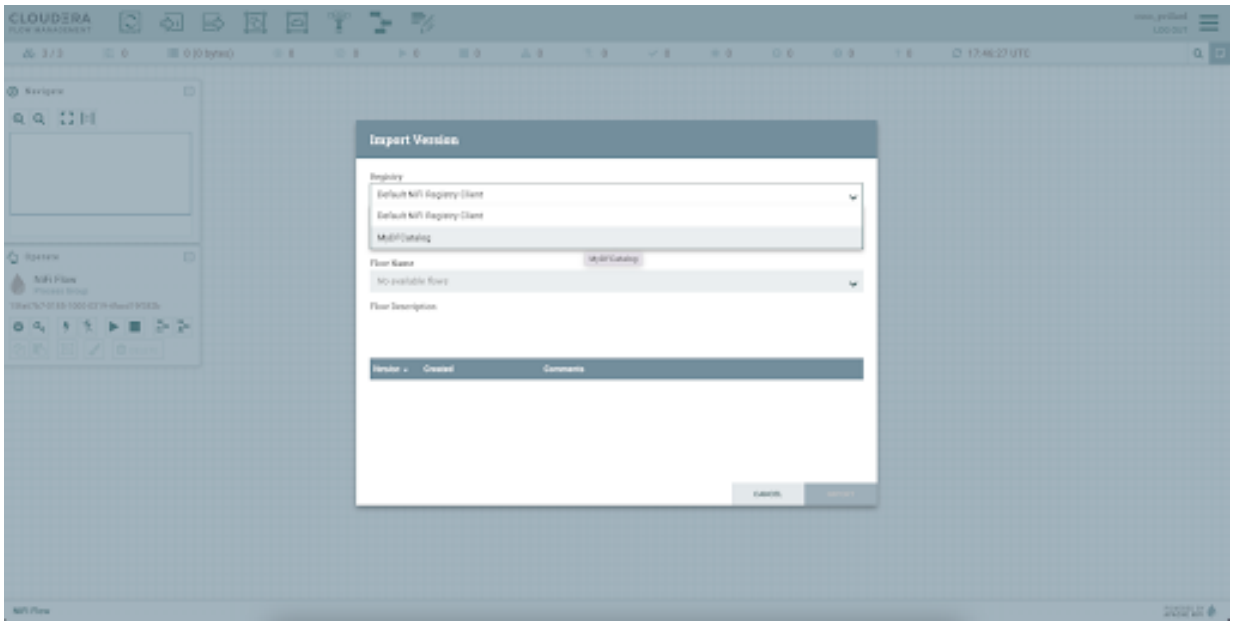

3. Select the ReadyFlows bucket.

   You have two buckets - one for the ReadyFlows (which is read-only) and one for your own flows that are stored in the Catalog.

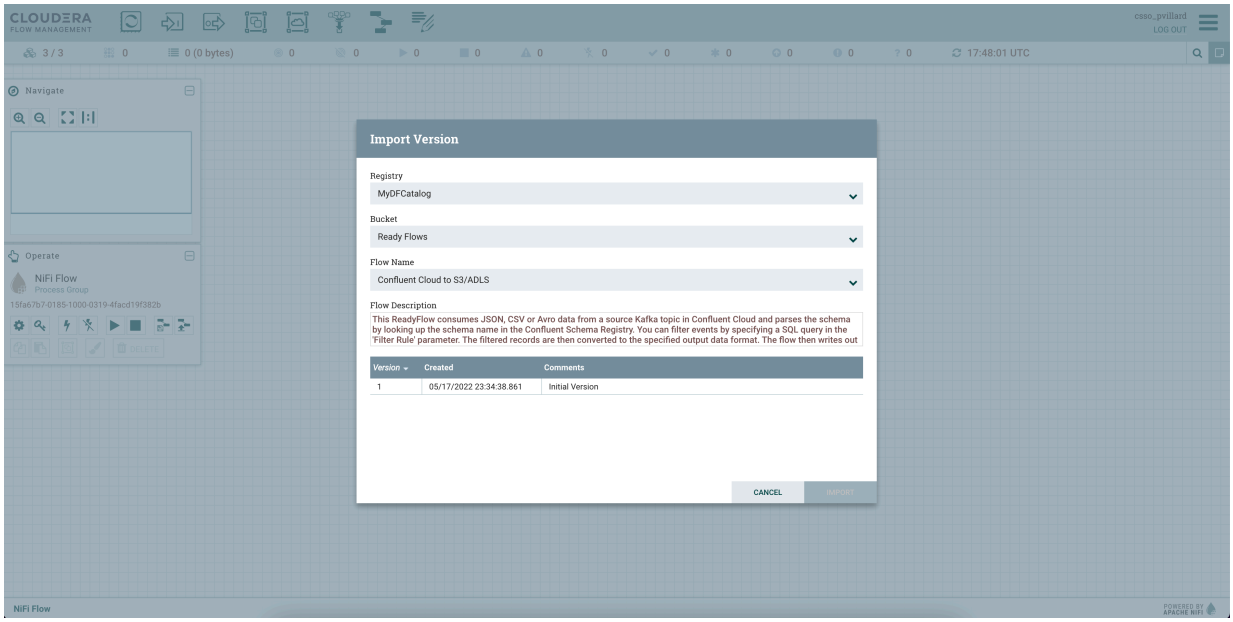

4. Select the ReadyFlow that you want to check out in NiFi by clicking the selected version and then clicking Import.

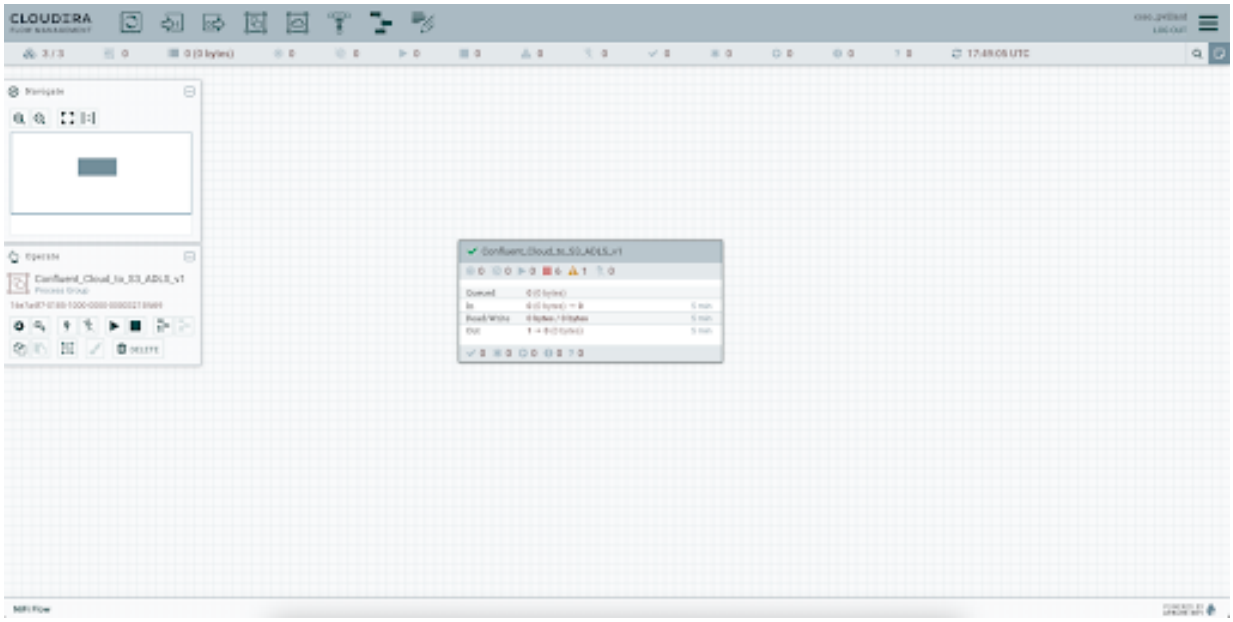

5. ReadyFlows are read-only so you would not be able to change flow. If you want to edit the flow:
   a) Right-click the process group, select Version and click Stop version control.
   b) Make the changes you want.
   c) Version the flow as a custom flow in your Catalog: right-click the process group, select Version and click Start version control.

Checking out your flows

Learn how to check out your own flows from the DataFlow Catalog.

About this task

If you want check out your own flow from the Cloudera DataFlow Catalog, you can follow the same approach as when checking out a ReadyFlow. You just need to select the Custom flows bucket. You can then make changes to your flow and commit the changes to create a new version of the flow in the Catalog.

Procedure

1. Drag and drop a process group onto the canvas and click Import from Registry.
2. In the drop-down list, select the Registry Client configured to access your DataFlow Catalog.
3. Select the Custom flows bucket.
4. Edit your flow and commit the changes to create a new flow version in the Catalog.

Versioning a flow in the Catalog

Learn how to use the DataFlow Catalog to version your flows.

About this task

Follow the below steps if you have a process group that you want to version as a completely new flow in the Cloudera DataFlow Catalog.

Procedure

1. Right-click the process group and select Version Start version control .

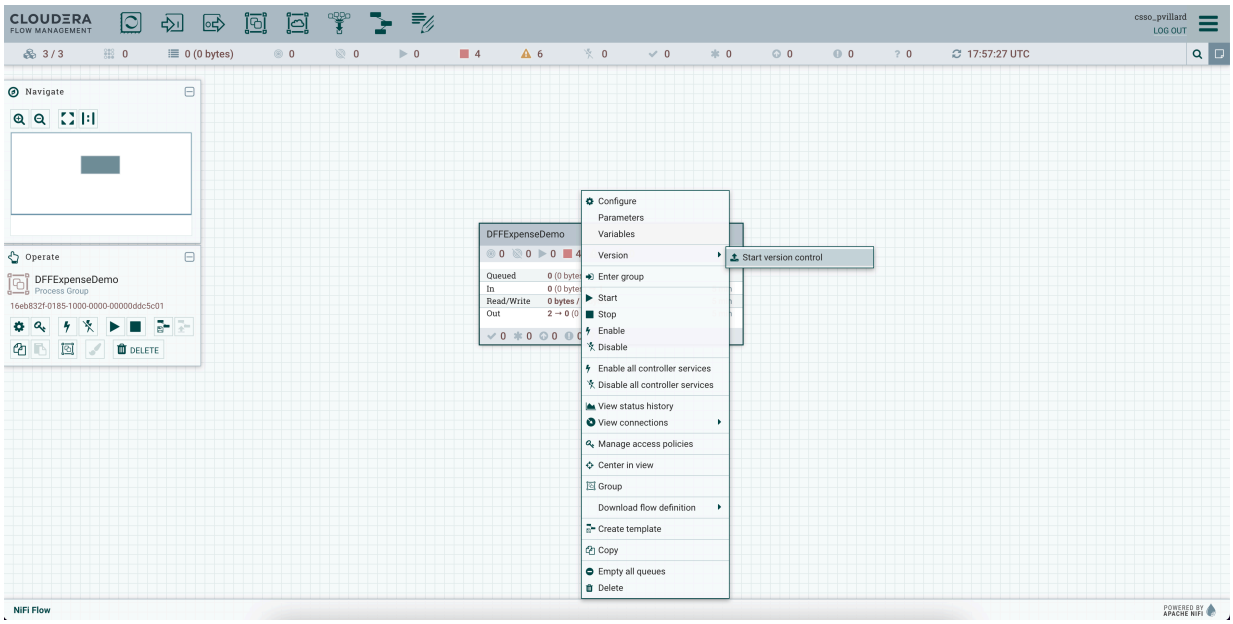

2. Select the Registry that you want to use.

3. Select Custom flows for the bucket and give your flow a name, a description, and a comment for the first version.

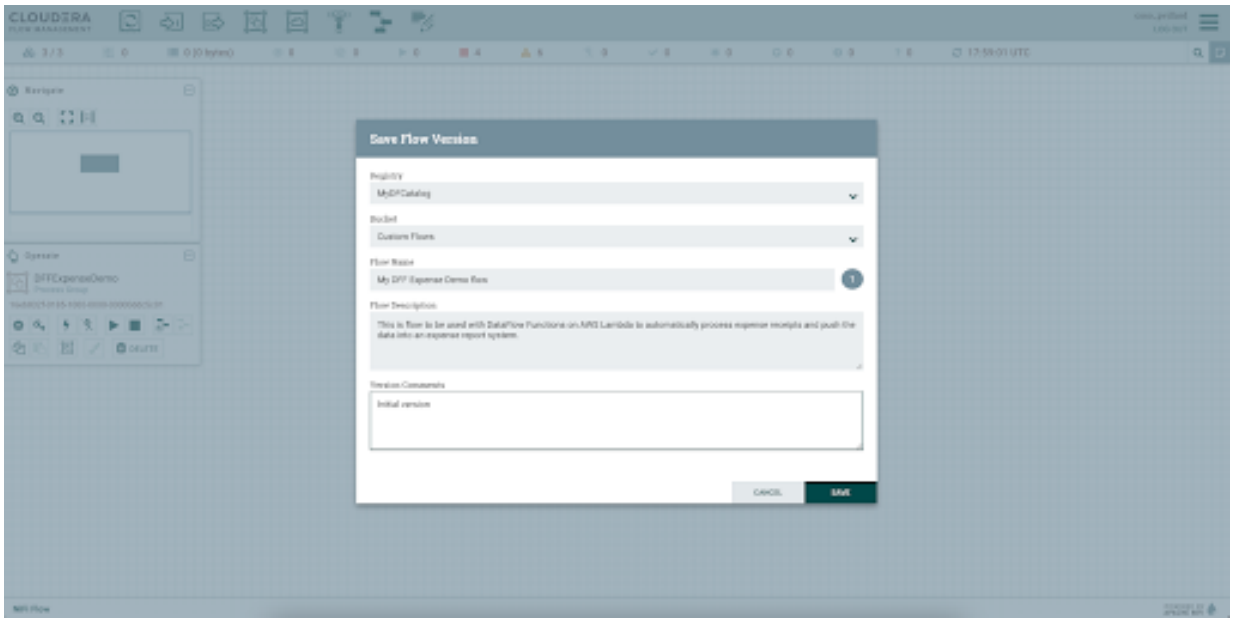

4. Click Save.

Results

The flow is now available in the DataFlow Catalog.

Note: All commits will be associated with the Machine User you configured when creating the Registry client.

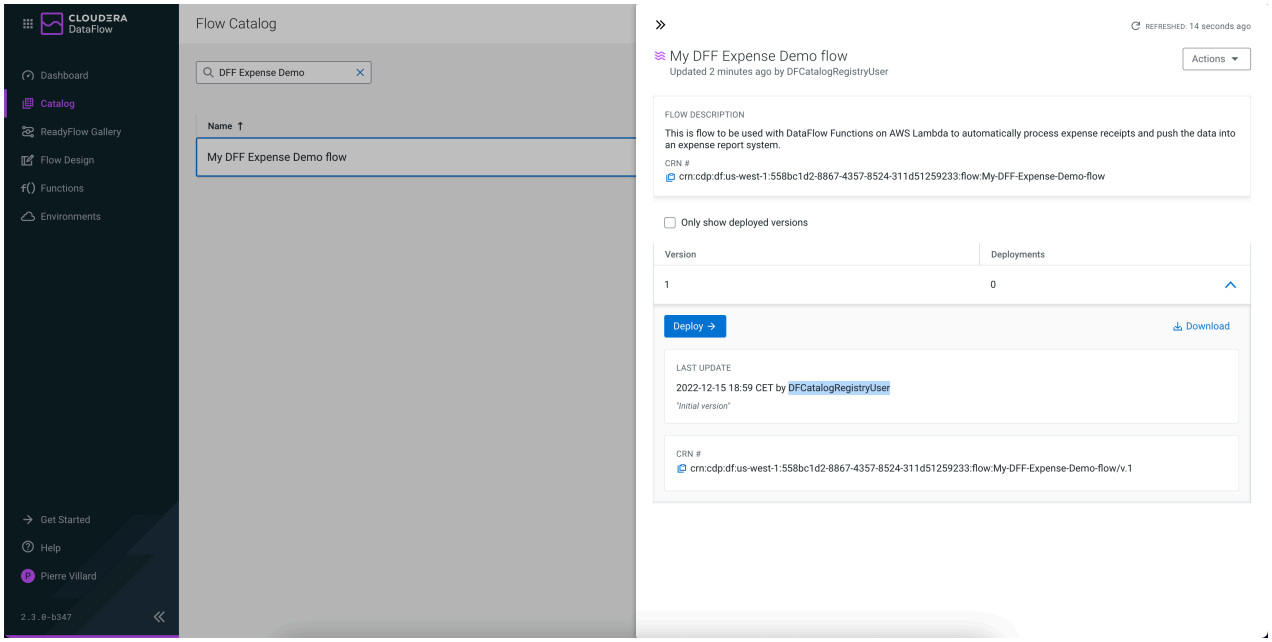

Cloudera Public Cloud

Cloudera API

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera API overview.....4
Getting started with the Cloudera API.....4
Request signing.....5
Cloudera API reference.....5

Cloudera API overview

Cloudera provides a REST API that enables you to access Cloudera functionality from a script, or to integrate Cloudera features with an application. In practice you can use the Cloudera API to script repetitive tasks, manage Cloudera resources, or even create custom applications.

The Cloudera API contains much of the functionality within the Cloudera Management Console and Cloudera Data Hub, as well as Cloudera AI, Cloudera Data Warehouse, and Cloudera Operational Database. For a complete Cloudera API reference, see the Cloudera Control Plane Public API Documentation.

Before you can use the Cloudera API, you must generate an API access key pair, which consists of an access key ID and a private key. You can generate the API access key through the Cloudera Management Console.

There are multiple ways to access the API:

- CDP CLI client
- Cloudera SDK for Java
- Directly, through a tool such as cdpcurl

Getting started with the Cloudera API

Getting started with the Cloudera API involves determining the best method for using the API, generating an API access key, and understanding permissions in the Cloudera API.

API access methods

Cloudera offers a CLI client, and all API commands have a CLI equivalent. See the CLI documentation for more details.

Java programmers might prefer to use the Cloudera SDK for Java. For more information, see the SDK documentation.

Alternatively, you can access API endpoints directly through your chosen HTTP client, such as curl. Cloudera offers cdpcurl, which is a Python command line tool that you can use to directly make Cloudera API calls, or embed in an application. cdpcurl returns JSON output for ease of use. It is based on awscurl and installation and usage are similar. Like the CDP CLI and SDK, cdpcurl has built-in Cloudera request signing to construct the headers required to make an API call. However, if you want to use a different HTTP client, such as curl, you can use a script within cdpcurl to generate the required headers.

You can also use the Cloudera API service definition files to create your own client-side code. The service definitions are in OpenAPI / Swagger 2.0 format and can be accessed here.

Generating an API access key

A Cloudera user account must have API access credentials (an API access key and private key) to access the Cloudera API through the CLI or other means. To use the CLI client, you configure the Cloudera client with the API access key and private key. To make API calls with cdpcurl, you pass the API access key and private key to cdpcurl.

Permissions

You are restricted to making API calls that are permissible based upon your user roles and resource roles. For more information on user roles, see the topic *Understanding Roles and Resource Roles* in the Cloudera Management Console documentation.

Related Information

Getting started with the Cloudera API

Request signing

cdpcurl, the Cloudera SDK, and CDP CLI offer built-in request signing, but if you use a different HTTP client to make an API call, you must generate the required headers.

A Cloudera API call requires a request signature to be passed in the "x-altus-auth" header, along with a corresponding timestamp in the "x-altus-date" header. cdpcurl constructs the headers automatically. However, if you would rather use a different HTTP client, such as ordinary curl, you can use the cdpv1sign script within cdpcurl to generate these required headers.

You can then parse the header values from the script output and feed them to your preferred client. Note that Cloudera API services will reject calls with timestamps too far in the past, so generate new headers for each call.

For example:

```
$ cdpv1sign -X POST https://api.us-west-1.cdp.cloudera.com/api/v1/environmen
ts2/listEnvironments
Content-Type: application/json
x-altus-date: Fri, 28 Aug 2020 20:38:38 GMT
x-altus-auth: (long string value)
```

The request signing specification is available here.

Cloudera API reference

The Cloudera API reference documentation is available here. Use this documentation to find the desired API endpoint, request parameters, and responses.

Cloudera Public Cloud

AWS Onboarding Quickstart

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

AWS quickstart (Deprecated)..... 4
Verify AWS prerequisites..... 4
Create a Cloudera credential..... 5
Register a Cloudera environment..... 7

AWS quickstart (Deprecated)

If you've reached the Cloudera landing page for the first time, you've come to the right place! In this quickstart, we'll show you step-by-step how to connect Cloudera to your AWS account, so that you can begin to provision clusters and workloads.

Warning: This quickstart has been deprecated and is no longer being maintained. For quickly setting up Cloudera on AWS, refer to Deploy Cloudera using Terraform.

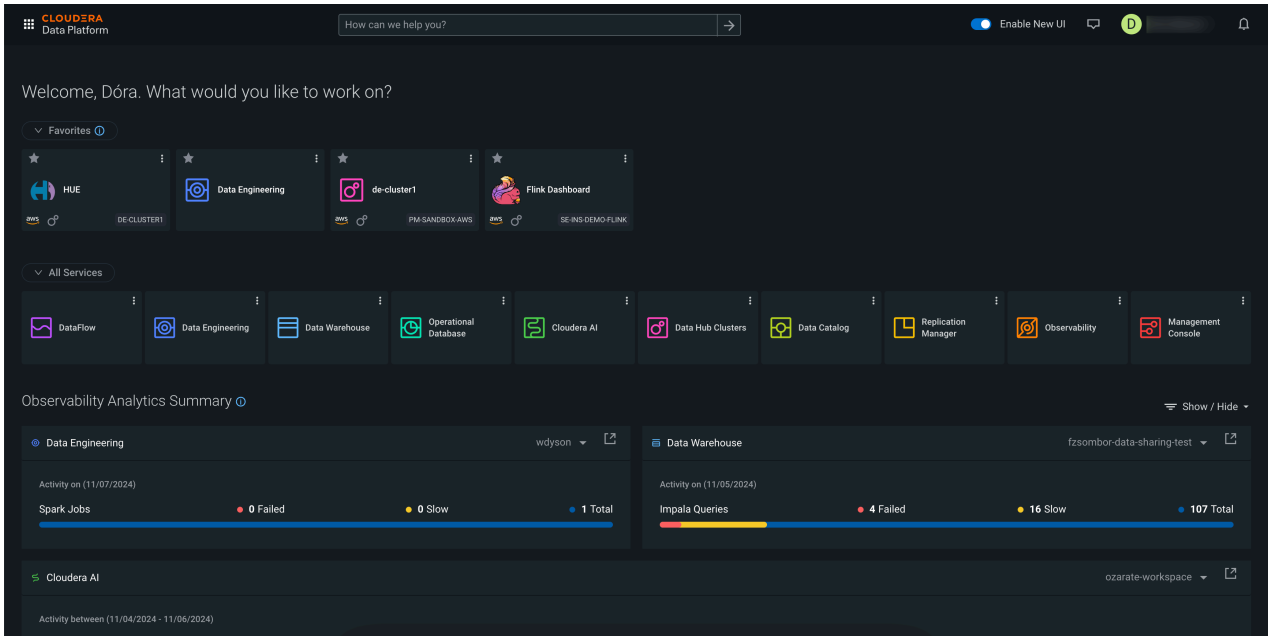

To complete this quickstart, you'll need access to two things:

- The Cloudera console pictured above
- The AWS console

Note: This AWS onboarding quickstart is intended for simple Cloudera evaluation deployments only. It may not work for scenarios where AWS resources such as VPC, security group, storage accounts, and so on, are pre-created or AWS accounts have restrictions in place.

The steps that we will perform are:

Step 0: Verify the AWS prerequisites

Step 1: Create a provisioning credential

Step 2: Register an AWS environment in Cloudera

Verify AWS cloud platform prerequisites

Before getting started with the AWS onboarding quickstart, review and acknowledge the following:

- This AWS onboarding quickstart is intended for simple Cloudera evaluation deployments only. It may not work for scenarios where AWS resources such as VPC, security group, storage accounts, and so on, are pre-created or AWS accounts have restrictions in place.

- Users running the AWS onboarding quickstart should have:
  - AWS Administrator permissions on the AWS account that you would like to use for Cloudera.
  - Rights to create AWS resources required by Cloudera. See list of AWS resources used by Cloudera.
  - Cloudera Admin role or Power User role in Cloudera subscription.
- This AWS onboarding quickstart uses a CloudFormation template that automatically creates the required resources such as buckets, IAM roles and policies, and so on.
- Cloudera Public Cloud relies on several AWS services that should be available and enabled in your region of choice. Verify if you have enough quota for each AWS service to set up Cloudera in your AWS account. See list of AWS resources used by Cloudera.

If you have more complex requirements than those listed here, contact Cloudera Sales Team to help you with the Cloudera onboarding.

Create a Cloudera credential

In the Cloudera console, the first step is to create a Cloudera credential. The Cloudera credential is the mechanism that allows Cloudera to create resources inside of your cloud account.

Procedure

1. Log in to the Cloudera web interface.
2. From the Cloudera home screen, click the Cloudera Management Console icon.
3. In the Cloudera Management Console, select Shared Resources > Credentials from the navigation pane.
4. Click Create Credential.
5. Click the Copy icon to the right of the **Create Cross-account Access Policy** text box.

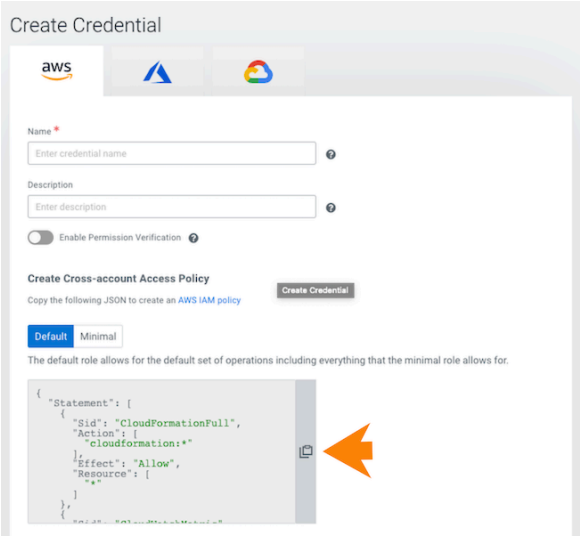

6. In a second browser tab, open the **AWS Console** and navigate to Identity and Access ManagementPolicies. Click Create Policy.

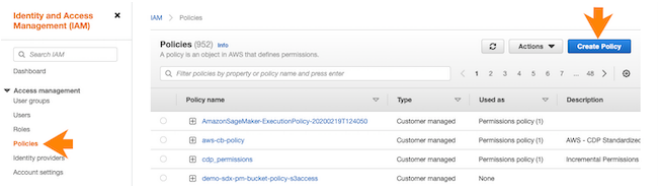

7. Click on the JSON tab and paste the access policy in the text box.

   You may get a warning related to using wildcards. You may ignore it and proceed to the next step.

8. Click Next: Tags.
9. Click Review Policy.
10. Give the policy a unique name and a description.
11. Click Create Policy.

    Next, you create the required cross-account role.
12. In the AWS console, navigate back to Identity and Access Management.
13. Click RolesCreate Role.
14. Under **Select type of trusted entity**, select AWS Account > Another AWS account.
15. Return to the Cloudera Management Console and copy the contents of the Service Manager Account ID field on the **Credentials** page.
16. In the AWS console, paste the Service Manager Account ID into the Account ID field.
17. Return to the Cloudera Management Console and copy the contents of the External ID field on the **Credentials** page.
18. In the AWS console, check the "Require external ID" options box, and then paste the External ID copied from Cloudera into the External ID field.
19. Click Next: Permissions.
20. Under Permissions, select the checkbox next to the name of the policy that you created in Step 8.
21. Click Next: Tags.
22. Click Next: Review.
23. Give the role a unique name and description, then click Create Role.
24. Still in the role page of the AWS console, search for the role you just created, and click on it.
25. Copy the Role ARN at the top of the **Summary** page.

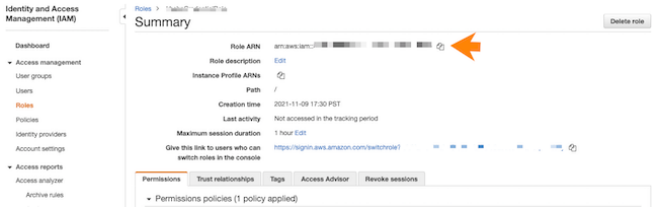

26. Return to the **Credentials** page in the Cloudera Management Console.
27. Give the Cloudera credential a name and description. The name can be any valid name.

28. Paste the Role ARN that you copied from the AWS console into the Cross-account Role ARN field, then click Create.

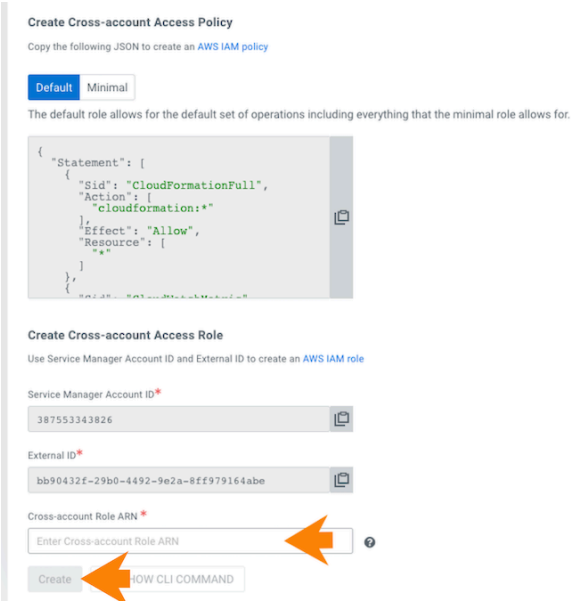

Now that you've created a cross-account role, proceed to creating a Cloudera environment.

Register a Cloudera environment

Before you register an environment, you'll want to create specific IAM roles and policies so that Cloudera can operate in a secure manner.

About this task

For background information, a description of what we're building and why can found here. For this quickstart, we'll use CloudFormation to set all of this up for you.

Procedure

1. Download the CloudFormation provided template here.

2. In the AWS console, deploy the CloudFormation template:
   a) In **AWS Services**, search for CloudFormation.
   b) Click Create Stack and select With new resources (standard).
   c) Select Template is ready and then Upload a template file.

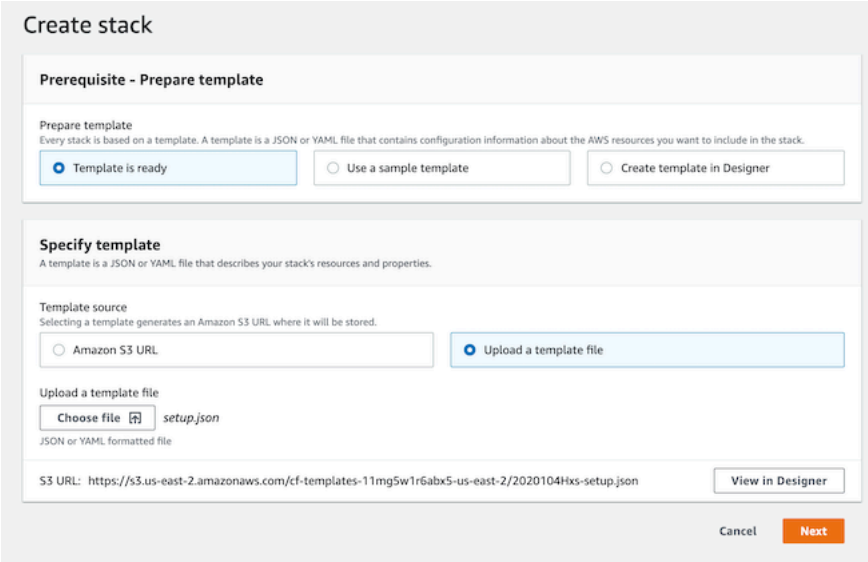

   d) Click Choose file and select the CloudFormation template that you downloaded.
   e) Click Next.
   f) Under Stack name, enter a stack name. The name can be any valid name.
   g) Under **Parameters**, complete the following fields:

      - BackupLocationBase: Choose an unused bucket name and path for the FreeIPA backups. Cloudera will be creating the bucket for you. The same bucket can be used for BackupLocationBase, LogsLocationBase, and StorageLocationBase. By default this is set to my-bucket/my-backups.
      - CrossAccountARN: Do not change the default value. This parameter is only required when encryption is enabled, but since in this quickstart we do not enable encryption, you should leave this value as is.
      - LogsLocationBase: Choose an unused bucket name and path for the logs. Cloudera will be creating the bucket for you. The same bucket can be used for BackupLocationBase, LogsLocationBase, and StorageLocationBase. By default this is set to my-bucket/my-logs.
      - StorageLocationBase: Choose an unused bucket name and path for the data. Cloudera will be creating the bucket for you. The same bucket can be used for BackupLocationBase, LogsLocationBase, and StorageLocationBase. By default this is set to my-bucket/my-data.
      - Prefix: A short prefix of your choosing, which will be added to the names of the IAM resources Cloudera will be creating. We chose "cloudera" as an example.
      - s3KmsEncryption: Encryption will be disabled for the created bucket. You don't need to change this value.

      For example:

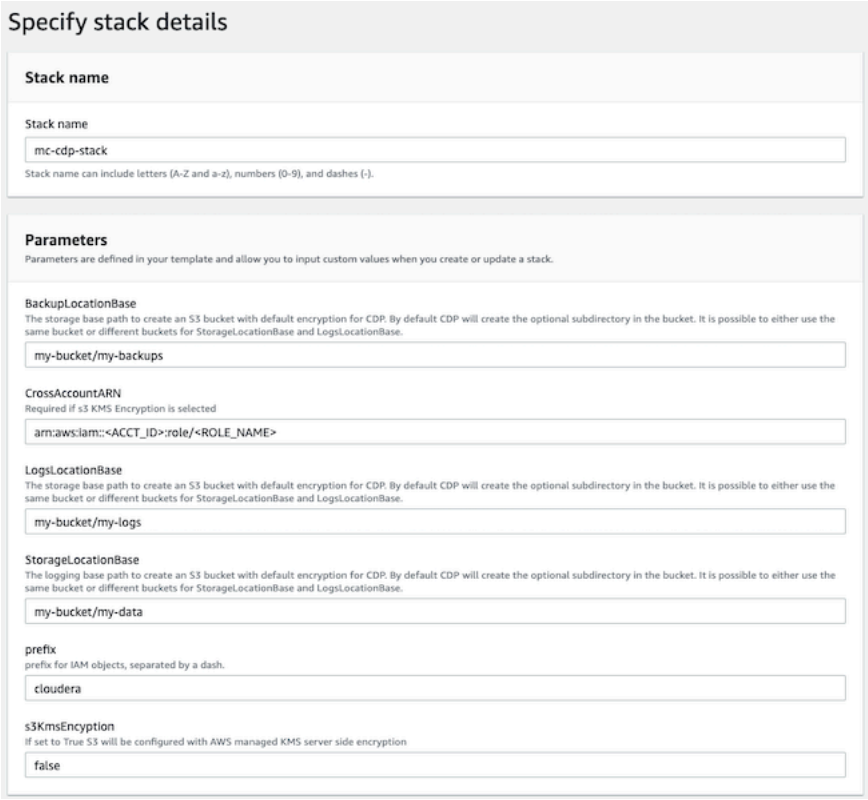

Make a note of the BackupLocationBase, LogsLocationBase, StorageLocationBase, and Prefix that you define. You will need them later.

h) Click Next.

i) At the **Configure Stack Options** page, click Next.

j) At the bottom of the **Review** page, under Capabilities, click the checkbox next to I acknowledge that AWS Cloudformation might create IAM resources with custom names, as that is exactly what we will be doing.

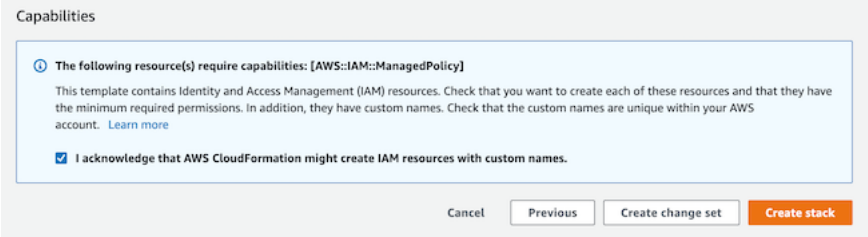

k) Click Submit.

3. Still in the AWS console, create an SSH key in the region of your choice. If there is already an SSH key in your preferred region that you'd like to use, you can skip these steps.

a) In **AWS Services**, search for EC2.

b) In the top right corner, verify that you are in your preferred region.

c) On the left hand navigation bar, choose Key Pairs.

d) On the top right of the screen, select Create Key Pair.

e) Provide a name. The name can be any valid name.

f) Choose RSA type, and then choose the pem format.

g) Click Create key pair.

4. Return to the Cloudera Management Console and navigate to Environments Register Environments .

5. Provide an environment name and description. The name can be any valid name.

6. Choose Amazon as the cloud provider.

7. Under **Amazon Web Services Credential**, choose the credential that you created earlier.
8. Click Next.
9. Under **Data Lake Settings**, give your new data lake a name. The name can be any valid name. Choose the latest data lake version.
10. Under **Data Access and Audit**:
    - Choose prefix-data-access-instance-profile>
    - For Storage Location Base, choose the StorageLocationBase from the cloud formation template.
    - For Data Access Role, choose prefix-datalake-admin-role.
    - For Ranger Audit Role, choose prefix-ranger-audit-role, where "prefix" is the prefix you defined in the **Parameters** section of the stack details in AWS.

    For example:

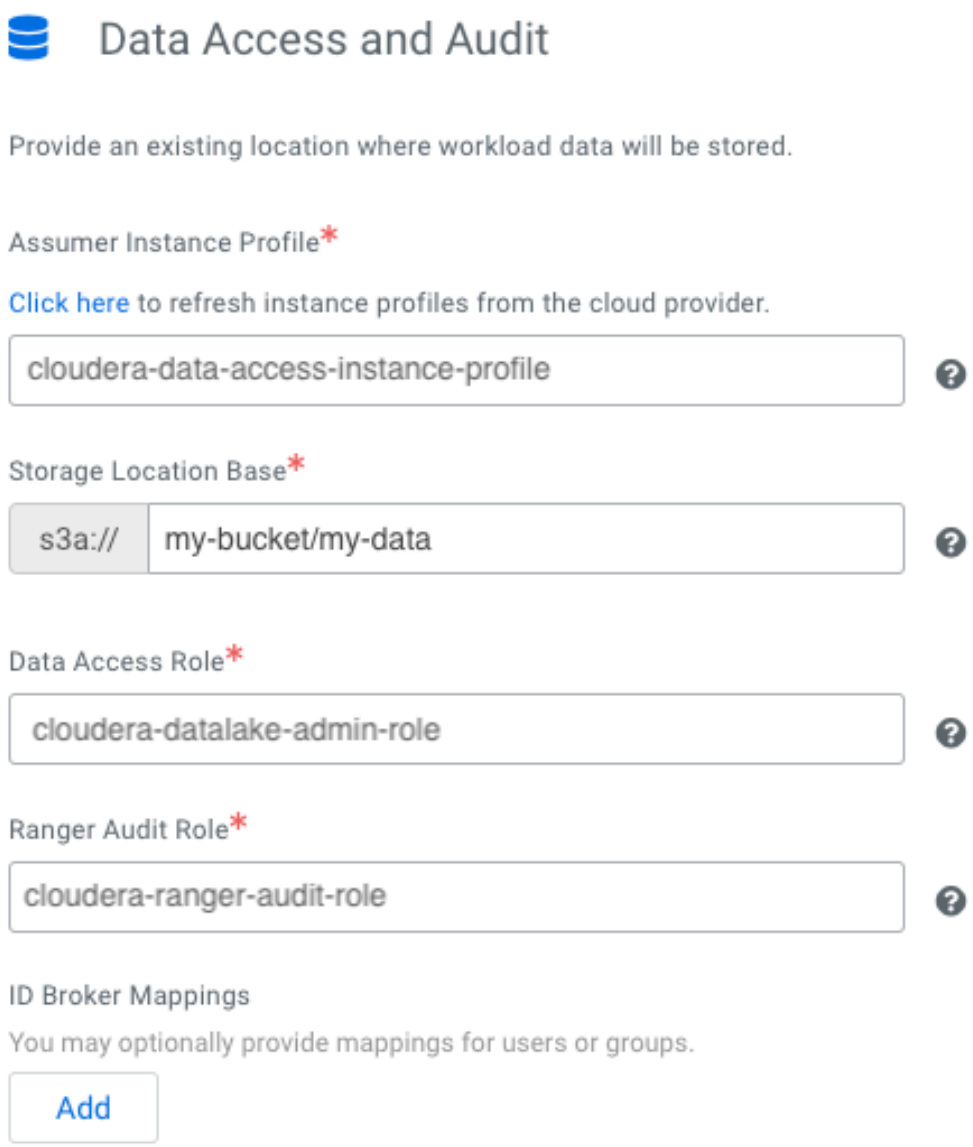

11. For Data Lake **Scale**, choose Light Duty.
12. Click Next.
13. Under Select Region, choose your desired region. This should be the same region you created an SSH key in previously.

14. Under **Select Network**, choose Create New Network.
15. Create private subnets should be enabled by default. If it isn't, enable it.

    Note:

    By enabling private subnets you will not have SSH access to cluster nodes unless you have access to the VPC.
16. Click the toggle button to enable Enable Public Endpoint Access Gateway.

    For example:

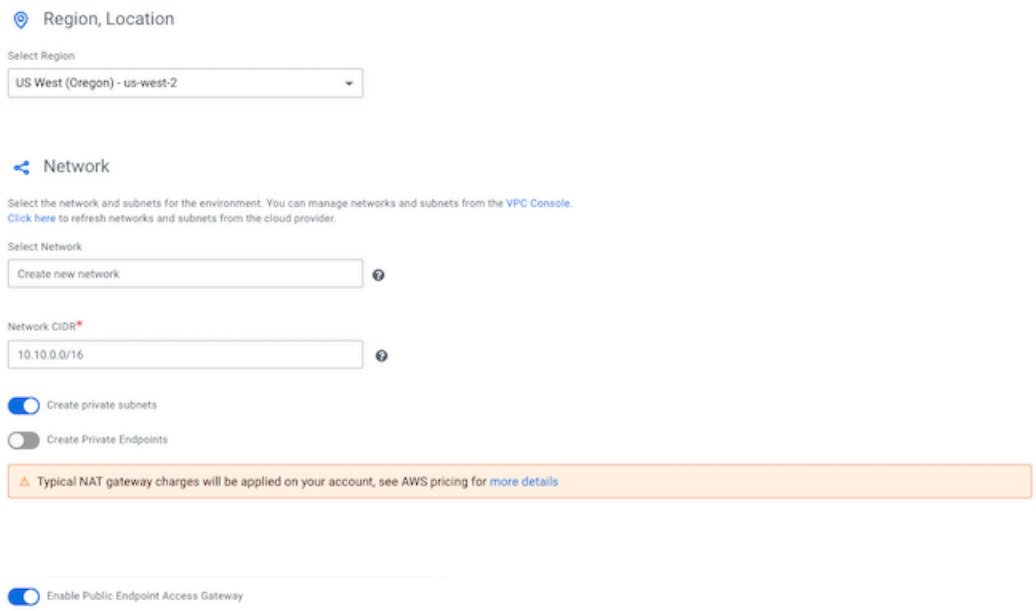

17. Under **Security Access Settings**, choose Create New Security Groups.

18. Under **SSH Settings**, choose the SSH key you created earlier.
    For example:

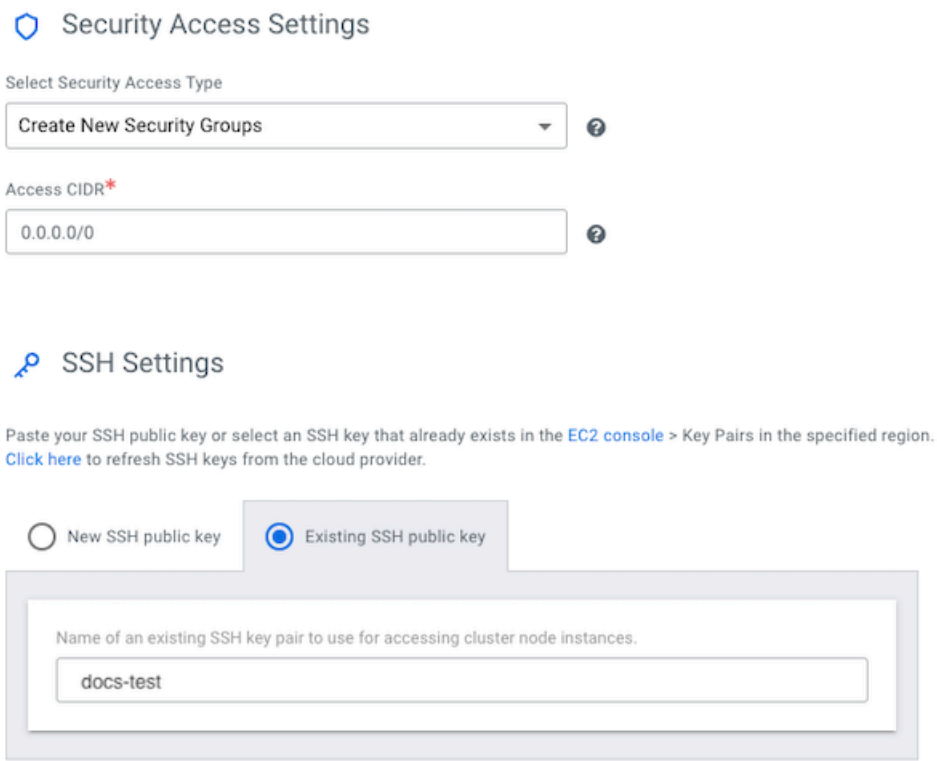

19. Optionally, under **Add Tags**, provide any tags that you'd like the resources to be tagged with in your AWS account.
20. Click Next.

21. Under **Logs**:

a) Choose the Instance Profile titled prefix-log-access-instance-profile, where "prefix" is the prefix you defined in the **Parameters** section of the stack details in AWS.

b) For Logs Location Base, choose the LogsLocationBase from the CloudFormation template.

c) For Backup Location Base, choose the BackupLocationBase from the CloudFormation template.

For example, using the parameters we defined earlier:

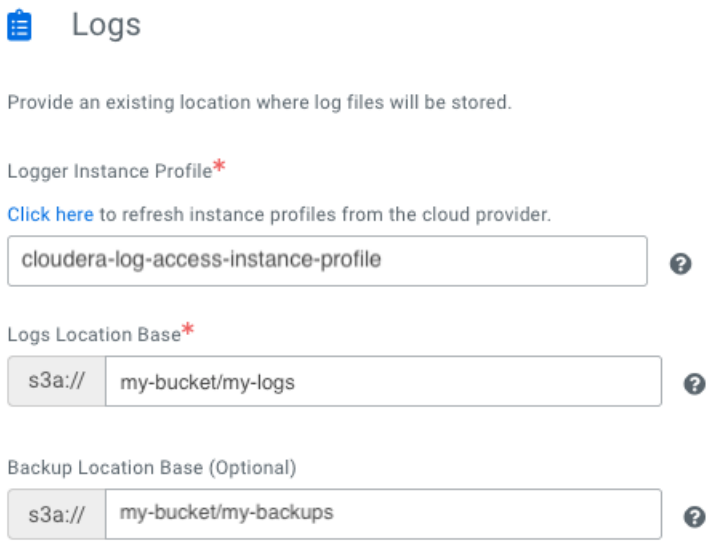

22. Click Register Environment.

Cloudera Public Cloud

AWS Reference Network Architecture

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Public Cloud reference network architecture for AWS.....4

Taxonomy of network architectures.....5
Cloudera Management Console to customer cloud network.....6
Customer on-prem network to cloud network.....8

Network architecture.....8
Architecture diagrams.....8
Component description.....11
VPC.....11
Subnets.....11
Gateways and route tables.....12
Security groups.....12
DNS.....14
DHCP option set.....16
Determining the CIDR range.....16
Option 1: Cloudera creates the VPCs and subnets.....16
Option 2: Existing VPC and subnets.....16
DNS.....19
Associating additional CIDRs to a VPC.....22
Cloudera Private Links Network for AWS.....23
Supported service components.....24
Setting up Cloudera Private Links Network for AWS environments.....25
Troubleshooting Cloudera Private Links Network.....31
References.....31

Cloudera Public Cloud reference network architecture for AWS

This topic includes a conceptual overview of the Cloudera Public Cloudnetwork architecture for AWS, its use cases, and personas who should be using it

Overview

Cloudera Public Cloud allows customers to set up cloud Data Lakes and compute workloads in their cloud accounts on AWS, Azure, and Google Cloud. It maps a cloud account to a concept called the environment into which all compute workload clusters (Cloudera Data Hub clusters) and data services (such as Cloudera Data Engineering, Cloudera Data Warehouse, Cloudera AI, Cloudera Operational Database, Cloudera DataFlow) are launched. For these Data Lakes, compute workload clusters, and data services to function correctly, several elements of the cloud architecture need to be configured appropriately: access permissions, networking setup, cloud storage and so on. Broadly, these elements can be configured in one of two ways:

- Cloudera can set up these elements for the customer

  Usually, this model helps to set up a working environment quickly and try out Cloudera. However, many enterprise customers prefer or even mandate specific configurations of a cloud environment for Infosec or compliance reasons. Setting up elements such as networking and cloud storage requires prior approvals and they would generally not prefer, or even actively prevent, a third party vendor like Cloudera to set up these elements automatically.
- Cloudera can work with pre-created elements provided by the customer

  In this model, the flow for creating the cloud Data Lakes accepts pre-created configurations of the cloud environment and launches workloads within those boundaries. This model is clearly more aligned with enterprise requirements. However, it brings with it the risk that the configuration might not necessarily play well with Cloudera requirements. As a result, customers might face issues launching Cloudera workloads and the turnaround time to get to a working environment might be much longer and involve many tedious interactions between Cloudera and the customer cloud teams.

From our experience in working with several enterprise customers, the most complicated element of the cloud environment setup is the cloud network configuration. The purpose of this document is to clearly articulate the networking requirements needed for setting up a functional Cloudera Public Cloud environment into which the Data Lakes and compute workloads of different types can be launched. It attempts to establish the different points of access to these workloads and establishes how the given architecture helps to accomplish this access.

Along with this document, you can use the cloudera-deploy tool to automatically set up a model of this reference architecture, which can then be reviewed for security and compliance purposes.

Use cases

Cloudera Public Cloud allows customers to process data in the cloud storage under a secure and governed Data Lake using different types of compute workloads that are provisioned via Data Hub or data services. Typically the lifecycle of these workloads is as follows:

- A Cloudera environment is set up by a Cloudera admin using their cloud account. This sets up a cloud Data Lake cluster and FreeIPA cluster with security and governance services and an identity provider for this environment. The Cloudera admin may need to work with a cloud administrator to create all the cloud provider resources (including networking resources) that are required by Cloudera.
- Then one or more compute workload clusters can be launched, linked to the Data Lake. Each of these workload clusters typically serves a specific purpose such as data ingestion, analytics, machine learning and so on.
- These compute workload clusters are accessed by data consumers like data engineers, analysts or scientists. This is the core purpose of using Cloudera Public Cloud.
- These compute workload clusters can be long-running or ephemeral, depending on the customer needs.

There are two types of users for Cloudera who interact with the product for different purposes:

- Cloudera admins - These persons are usually concerned with the launch and maintenance of the cloud environment, and the Data Lake, Cloudera Data Hub clusters, FreeIPA, and Cloudera data services running inside the environment. They use a Cloudera Management Console running in the Cloudera AWS account to perform these operations of managing the environment.
- Data consumers - These are the data scientists, data analysts, and data engineers who use the Data Hubs and data services to process data. They mostly interact directly with the compute workloads (Cloudera Data Hub clusters and data services) running in their cloud account. They could access these either from their corporate networks (typically through a VPN) or other cloud networks their corporate owns.

These two types of users and their interaction with Cloudera are represented in the following diagram:

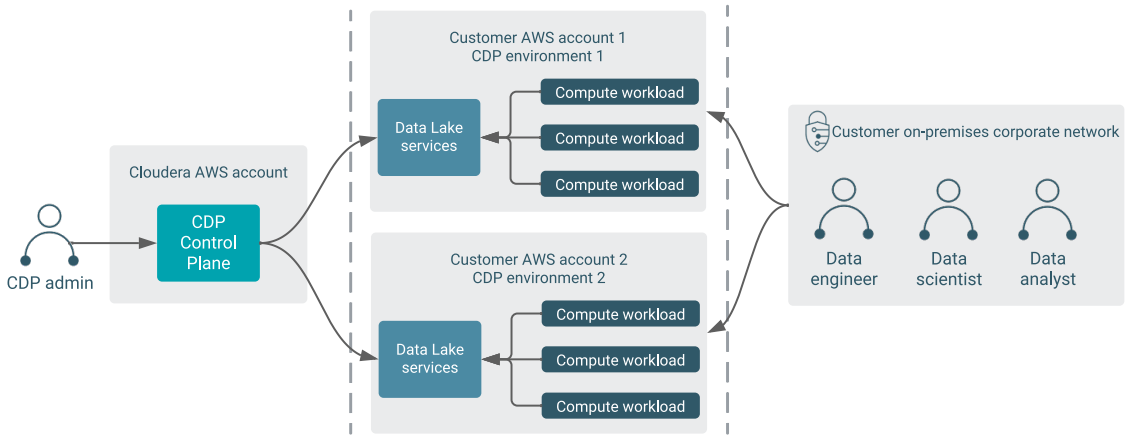

Taxonomy of network architectures

This topic provides a high-level overview of each type of network architecture that Cloudera supports.

At a high level, there are several types of network architectures Cloudera supports. As can be expected, each type brings a unique trade-off among various aspects, such as ease of setup, security provided, workloads supported, and so on. This section only provides a high level overview of each type. The characteristics of each type are explained under appropriate sections in the rest of the document. The users must review the advantages and disadvantages of each of these taxonomies in detail before making a choice suitable to their needs.

Name	Description	Trade-offs
Publicly accessible networks	Deploys customer workloads to hosts with public IP addresses. Security groups must be used to restrict access only to corporate networks as needed.	Easy to set up for POCs. Low security levels.
Semi-private networks	Deploys customer workloads to private subnets, but exposes services to which data consumers need access over a load balancer with a public IP address. Security groups or allow-lists (of IP addresses or ranges) on load balancers must be used to restrict access to these public services only to corporate networks as needed.	This option is fairly easy to set up too, but it may not solve all the use cases of access (in semi private networks). The surface of exposure is reduced, and it is reasonably secure.
Fully private networks	Deploys customer workloads to private subnets and even services to which data consumers need access are only on private IPs. Requires connectivity to corporate networks to be provided using solutions like VPN gateways, and so on.	Complex to set up depending on prior experience of establishing such connectivity, primarily due to the way the customer has to solve the corporate network peering problem. But it is very secure.

Name	Description	Trade-offs
Fully private outbound restricted networks	This is the same as fully private networks; Except, in addition, Cloudera provides a mechanism for users to configure an outbound proxy or firewall to monitor or restrict the communication outside their networks.	Most complex to set up, mainly considering the varied needs that data consumers would have to connect outside the VPC on an evolving basis. It is also the most secure for an enterprise.

Cloudera Management Console to customer cloud network

This topic explains the possible ways in which Cloudera Control Plane can communicate with the compute infrastructure in the customer network, in the context of the Cloudera Management Console.

As described previously, the Cloudera admin would typically use the Cloudera Management Console that runs in the Cloudera Control Plane to launch Cloudera environments with Data Lakes, FreeIPA, Cloudera Data Hub clusters, and data services into their cloud accounts. In order to accomplish this, the Cloudera Control Plane and the compute infrastructure in the customer network (such as EC2 instances, EKS clusters) should be able to communicate with each other. Depending on the chosen network architecture, this communication can occur in the ways described below.

Publicly accessible networks

In this model of publicly accessible networks, the compute infrastructure must be reachable over the public internet from the Cloudera Management Console. While this is fairly easy to set up, it is usually not preferred by enterprise customers, as it implies that the EC2 nodes or EKS nodes are assigned public IP addresses. While the access control rules for these nodes can still be restricted to the IP addresses of the Cloudera Management Console components, it is still considered insecure for each of the network architectures described earlier.

Semi-private networks

Publicly accessible networks are easy to set up for connectivity, both from the Cloudera Control Plane and the customer on-prem network, but have a large surface area of exposure as all compute infrastructure has public IP addresses. In contrast, fully private networks need special configuration to enable connectivity from the customer on-prem network, due to having no surface area of exposure to any of the compute infrastructure. While very secure, it is more complex to establish.

There is a third configuration supported by Cloudera, semi-private networks, that provides some trade-offs between these two options. In this configuration, the user deploys the worker nodes of the compute infrastructure on fully private networks as described above. However, the user chooses to expose UIs or APIs of the services fronting these worker nodes over a public network load balancer. By using this capability, the data consumers can access the UIs or APIs of the compute infrastructure through these load balancers. It is also possible to restrict the IP ranges from which such access is allowed using security groups.

While this option provides a trade-off between ease of setup and exposure levels, it may not satisfy all use cases related to communication between various endpoints. For example, some compute workloads involving Kafka or NiFi would not benefit from having a simple publicly exposed NLB. It is recommended that customers evaluate their use cases against the trade-off and choose an appropriately convenient and secure model of setup.

Fully private networks

In this model of fully private networks, the compute infrastructure is not assigned any public IP addresses. In this case, communication between the Cloudera Control Plane and compute infrastructure is established using a "tunnel" that originates from the customer network to the Cloudera Control Plane. All communication from the Cloudera Control Plane to the compute nodes is then passed through this tunnel. From experience, Cloudera has determined that this is the preferred model of communication for customers.

To elaborate on the tunneling approach, Cloudera uses a solution called Cluster Connectivity Manager. At a high level, the solution uses two components, an agent (Cluster Connectivity Manager agent) that runs on a VM provisioned in the customer network and a service (Cluster Connectivity Manager service) that runs on the Cloudera

Control Plane. The Cluster Connectivity Manager agent, at start-up time, establishes a connection with the Cluster Connectivity Manager service. This connection forms the tunnel. This tunnel is secured by asymmetric encryption. The private key is shared with the agent over cloud specific initialization mechanisms, such as a user-data script in AWS.

When any service on the Cloudera Control Plane wants to send a request to a service deployed on the customer environment (depicted in the below diagram as the “logical flow”), it physically sends a request to the Cluster Connectivity Manager service running in the Cloudera Control Plane. The Cluster Connectivity Manager agent and Cluster Connectivity Manager service collaborate over the established tunnel to accept the request, forward it to the appropriate service, and send a response over the tunnel to be handed over the calling service on the Cloudera Control Plane.

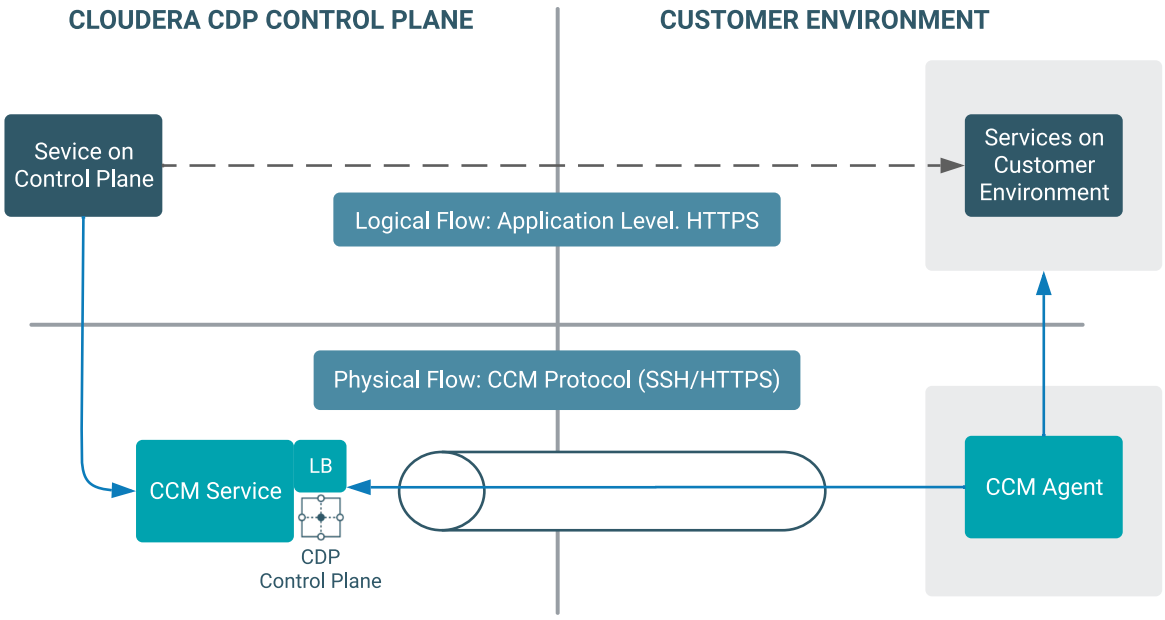

Currently, all EKS clusters provisioned by various Cloudera data services are enabled with public and private cluster endpoints even under fully private network setup (see Amazon EKS cluster endpoint access control). The EKS public endpoint is needed to facilitate the interactions between Cloudera Control Plane and the EKS cluster while worker nodes and Kubernetes control plane interact over private API endpoints. There are plans to support private EKS endpoints in the future. When this occurs, the documentation will be updated to reflect the same.

Fully private outbound restricted networks

Fully private outbound restricted networks is a variant of the fully private network where customers would like to pass outbound traffic originating from their cloud account through a proxy or firewall and explicitly allow-list URLs that are allowed to pass through. Cloudera Public Cloud supports such configuration. If suvh network architecture is chosen, the customer must ensure the following:

- Users configure a proxy for the environment via Cloudera, as documented in Use a non-transparent proxy with Cloudera Data Warehouse on AWS environments for Cloudera Data Warehouse and Using a non-transparent proxy for all other compute workloads and the Data Lake itself.
- Compute resources (such as VMs used by Cloudera Data Hub clusters and data services) can connect to the proxy or firewall via appropriate routing rules.
- The proxy or firewall is set up to allow connections to all hosts, IP ranges, ports, and protocol types that are documented in Outbound network access destinations for AWS.

Note:

Given that fully private networks is the recommended option of connectivity in most cases, this document describes the architecture assuming a fully private network setup.

Customer on-prem network to cloud network

After compute workload clusters are launched in the customer's cloud network, data consumers such as data engineers, data scientists, and data analysts access services running in these Cloudera data services. Sometimes, Cloudera admins who set up and operate these clusters might need this access to diagnose any issues the clusters face.

Examples of these include:

- Web UIs such as:
  - Hue: For running SQL queries in Hive tables
  - Cloudera AI Workbenches: For accessing Cloudera AI projects, models, notebooks, and so on
  - Cloudera Manager: For Cloudera Data Hub clusters and Data Lakes
  - Atlas and Ranger: For metadata, governance, and security in the Data Lake
- JDBC endpoints: Customers can connect tools such as Tableau using a JDBC URL pointing to the Hive server.
- SSH access: Data engineers might log in to nodes on the compute Cloudera data services to run data processing jobs using YARN, Spark, or other data pipeline tools.
- Kube API access: Cloudera data services that run on Amazon EKS (such as Cloudera Data Warehouse and Cloudera AI) also provide admin access to Kubernetes for purposes of diagnosing issues.
- API access: Customers can use APIs for accessing many of the services exposed via the web UIs for purposes of automation and integration with other tools, applications, or other workloads they have. For example, Cloudera AI exposes the "Cloudera AI API v2" to work with Cloudera AI projects and other entities. See Cloudera AI API v2.

These services are accessed by these consumers from within a corporate network inside a VPN. These services typically have endpoints that have a DNS name, the format of which is described more completely in the DNS section of this reference architecture documentation. These DNS names resolve to IP addresses assigned to the nodes, or load balancers fronting the ingest controllers of Kubernetes clusters. Note that these IP addresses are usually private IPs; Therefore, in order to be able to connect to these IPs from the on-premise network within a VPN, some special connectivity setup would be needed, typically accomplished using technologies like VPN peering, DirectConnect, transit gateways, and so on. While there are many options possible here, this document describes one concrete option of achieving this connectivity.

Network architecture

Cloudera recommends that customers configure their cloud networks as fully private networks, as described in this chapter. This will help on-boarding Data Lakes, Cloudera Data Hub clusters, and data services smoothly.

Note: This network architecture only covers the fully private networks and assumes unrestricted outbound access.

The cdpctl tool, which is released along with this document can be used to automatically set up a model of this reference architecture, which can then be reviewed for security and compliance purposes.

Architecture diagrams

This topic includes diagrams illustrating the various elements of the network architecture in the customer's cloud account into which Cloudera environments with Data Lakes, Cloudera Data Hub clusters, and data services will be launched.

Cloudera recommends that customers configure their cloud networks as described in this chapter and illustrated in the following diagrams. This will help onboarding Data Lakes, Cloudera Data Hub clusters, and data services smoothly. The following diagram illustrates the configuration for a fully private network that can be configured by the customer. This configuration can be provided by the Cloudera admins when they are setting up Cloudera environments and workloads which will get launched into this configuration.

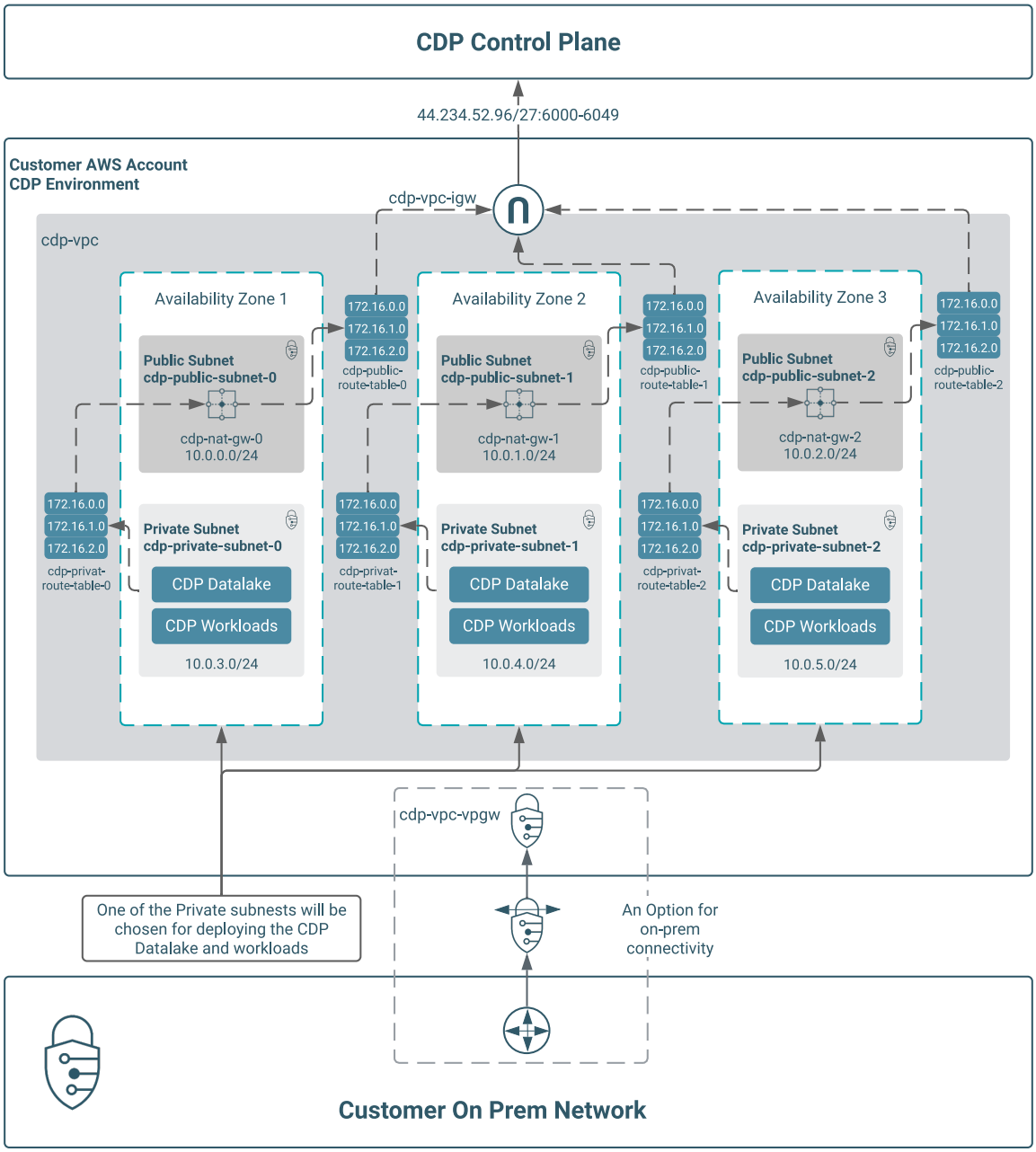

Note the following points about the architecture:

- The configuration is a fully private network configuration - that is, the workloads are launched on nodes that do not have public IP addresses into a private subnet.
- They connect outbound to the Cloudera Control Plane over a fixed IP and port range.
- For users to be able to connect from the customer on-prem network to the Cloudera workloads in the private subnet, some network connectivity setup is required. In this case, a customer's VPN server peered to an AWS virtual private gateway is shown.

Some of the Cloudera data services are based on AWS EKS clusters. Amazon EKS manages the Kubernetes Control Plane while the worker nodes that make up the cluster get provisioned in the customer’s VPC. The EKS Control Plane has an API endpoint for administrative purposes which is commonly referred to as "cluster endpoint". The Cloudera data service itself is accessible through a service endpoint ELB.

This is illustrated in the following diagram:

As can be seen in the above diagram, Cloudera workloads have dependencies on some AWS cloud services such as RDS, EFS and so on. A full list of these services, described in the context of each workload is specified in AWS outbound network access destinations.

In the chapters that follow, we detail the elements of this architecture, including specifying the configuration and options in each of the components.

Component description

This topic provides an overview of the VPC, subnets, gateways and route tables, and security groups required for Cloudera Public Cloud for AWS.

VPC

An Amazon Virtual Private Cloud (VPC) is needed for deploying Cloudera workloads into the customer's cloud account. Cloudera recommend that the VPC used for Cloudera is configured with properties as specified in this topic.

- The CIDR block for the VPC should be sufficiently large for supporting all the Cloudera Data Hub clusters and data services that you intend to run. Refer to Determining the CIDR range for understanding how to compute the CIDR block range.
- The VPC properties for DNS hostnames and DNS resolution must be ENABLED. DNS resolution lets Kubernetes pods resolve external host names and also to support DNS hostnames. The DNS hostnames option needs to be enabled as several Cloudera data services rely on EFS (see Mounting on Amazon EC2 with a DNS name). Enabling these properties is also a requirement (see Amazon EKS cluster endpoint access control) to enable private access of EKS cluster endpoints.
- VPCs are associated with a DHCP Option Set. The DHCP option set for the VPCs must be set up as per the section described in DHCP option set.

Subnets

A subnet is a partition of the virtual network in which Cloudera workloads are launched.

It is recommended that the subnets be configured with the following properties:

- It is recommended to have 3 private subnets and 3 public subnets, such that each private-public subnet pair is in a different availability zone (AZ). Even if a region has two AZs instead of three, it's recommended that three private subnets are created, two in the same AZ. This is required to prevent cross AZ routing of traffic and to maintain Quorum-based consistency required by some services.
  - Note that a subnet becomes 'private' or 'public' based on the routing devices it is associated with in the route tables. This is described in Gateways and route tables.
  - The private subnets will be where the compute workloads will be launched by Cloudera. This ensures that these nodes are working in an isolated and secure environment that does not have internet connectivity.
  - The public subnet is needed to host a NAT gateway as this will allow the compute nodes to reach out to the Cloudera Control Plane over the internet. More on this will be described in Gateways and route tables.
- The CIDR block for the subnets should be sufficiently large for supporting all the Cloudera data services you intend to run. Refer to Determining the CIDR range for understanding how to compute the CIDR block range.
- The CIDR block for the subnets should not overlap with known AWS EKS ranges for pods/services. Several EKS based Cloudera data services in Overlay networks
- In addition, you may want to ensure that the CIDR ranges assigned to the Subnets will not overlap with any of your on-premise network CIDR ranges, as this may be a requirement for setting up connectivity from your on-prem network to the subnets.
- Since Cloudera recommends fully private network configuration, the 'Auto-assign public IPs' option must be disabled for the private subnets.
- A subnet can be associated with a Network ACL (NACL). However, since Cloudera works with a fully private network configuration where communication is always initiated from EC2 nodes within the subnets, a NACL is generally not useful for this configuration.
- Tag private subnets with a tag 'kubernetes.io/role/internal-elb:1'. The key is the string and the value is '1'. Cloud Controller Manager and AWS Load Balancer Controller both require private subnets to have this tag for automatic creation of private ELBs. Private ELBs are created in these subnets by EKS. This is applicable when Cloudera is supporting EKS versions < 1.20 (which is currently the case). See How can I tag the Amazon VPC subnets in my Amazon EKS cluster.

Gateways and route tables

This topic covers recommended gateway and route table configurations for Cloudera Public Cloud for AWS.

Connectivity from Cloudera Control Plane to Cloudera workloads

- As described in Taxonomy of network architectures, nodes in the Cloudera workloads need to connect to the Cloudera Control Plane over the internet to establish a 'tunnel' over which the Cloudera Control Plane can send instructions to the workloads.
- In order to accomplish this, there are two gateways that need to be configured - a NAT Gateway in each of the public subnets and an Internet Gateway at the VPC level.
- The private subnet hosting the Cloudera workloads should be configured with a route table where the default route (0.0.0.0/0) points to a NAT Gateway in the public subnet of its AZ.
- The public subnet hosting the NAT Gateway should be configured with a route table where the default route (0.0.0.0/0) points to an Internet Gateway the VPC is configured with.
- Each NAT gateway requires an elastic IP address. The VPC should contain as many elastic IP addresses as NAT gateways across the AZs in the VPC.

Connectivity from customer on-prem to Cloudera workloads

- As described in Use cases, data consumers need to access data processing or consumption services in the Cloudera workloads. Given these are created with private IP addresses in private subnets, the customers will need to arrange for access to these addresses from their on-prem or corporate networks in specific ways.
- There are several possible solutions for achieving this, but one that is depicted in the Architecture diagram, uses a AWS VPN Gateway service.
- In this solution, the customer has to create a Virtual Private Gateway, and connect it to the VPN service on the on-prem network.

Security groups

During the specification of a VPC to Cloudera, the Cloudera admin specifies the security groups that will be associated with all the Cloudera workloads launched within that VPC. These security groups will be used in allowing the incoming traffic to the hosts.

Security groups for Data Lakes and Cloudera Data Hub clusters

During the specification of a VPC to Cloudera, the Cloudera admin can either let Cloudera create security groups, taking a list of IP address CIDRs as input; or create them in AWS and then provide them to Cloudera.

When getting started with Cloudera, the Cloudera admin can let Cloudera create security groups, taking a list of IP address CIDRs as input. These will be used in allowing the incoming traffic to the hosts. The list of CIDR ranges should correspond to the address ranges from which the Cloudera workloads will be accessed. In a VPN-peered VPC, this would also include address ranges from customer's on-prem network. This model is useful for initial testing given the ease of set up.

Alternatively, the Cloudera admin can create security groups on their own and select them during the setup of the VPC and other network configuration. This model is better for production workloads, as it allows for greater control in the hands of the Cloudera admin. However, note that the Cloudera admin must ensure that the rules meet the requirements described below.

For a fully private network, security groups should be configured according to the types of access requirements needed by the different services in the workloads:

- Services accessed only within the VPC must be configured with the following inbound rules:
  - All TCP / UDP / ICMP access is allowed for the CIDRs corresponding to the VPC.
  - Conversely, there is no need to provide any access to these services for any IP CIDRs outside the VPC.

- Endpoint services are the services that can be accessed outside the VPC through the gateway, chiefly by data consumers or Cloudera admins. For example, UIs like Hue, Atlas, Ranger, Cloudera Manager all need to be accessed by data consumers or other administrators. For enabling this, the following in-bound rules are set up:
  - All TCP / UDP / ICMP access is allowed for the CIDRs corresponding to the VPC.
  - All TCP ports that correspond to services like Kafka, HBase, and so on that need to be accessed outside the VPC are to be allowed for the list of CIDR ranges specified at the time of creating the environment/Cloudera data service, or in the security group created by the Cloudera admin. Alternatively, all TCP / UDP / ICMP access may be allowed for these CIDR ranges.
  - SSH access is allowed for the CIDR ranges specified at the time of creating the environment.
  - HTTPS access is allowed for the CIDR ranges specified at the time of creating the environment.
- Note that for a fully private network, even specifying an open access here (such as 0.0.0.0/0) is restrictive because these services are deployed in a private subnet without a public IP address and hence do not have a route to the Internet gateway. However, the list of CIDR ranges may be useful to restrict which private subnets of the customer's on-prem network can access the services. Rules for EKS based workloads are described in the following section.
- Rules for AKS based workloads are described separately in the following section.

Additional rules for EKS-based workloads

At the time of enabling a Cloudera data service, the Cloudera admin can specify a list of CIDR ranges that will be used in allowing the incoming traffic to the workload Elastic Load Balancer (ELB). This list of CIDR ranges should correspond to the address ranges from which the Cloudera data service workloads will be accessed. In a VPN peered VPC, this would also include address ranges from customer's on-prem network. In a fully private network setup, 0.0.0.0/0 implies access only within the VPC and the peered VPN network which is still restrictive.

Since a public endpoint is enabled by default for all EKS cluster Control Planes at the moment, it is highly recommended to provide a list of outbound public CIDR ranges at the time of provisioning a Cloudera data service to restrict access to the EKS clusters. By default, the public endpoint is always allowed to connect to the Cloudera public CIDR range. The following screenshot is an example configuration section for a Cloudera data service:

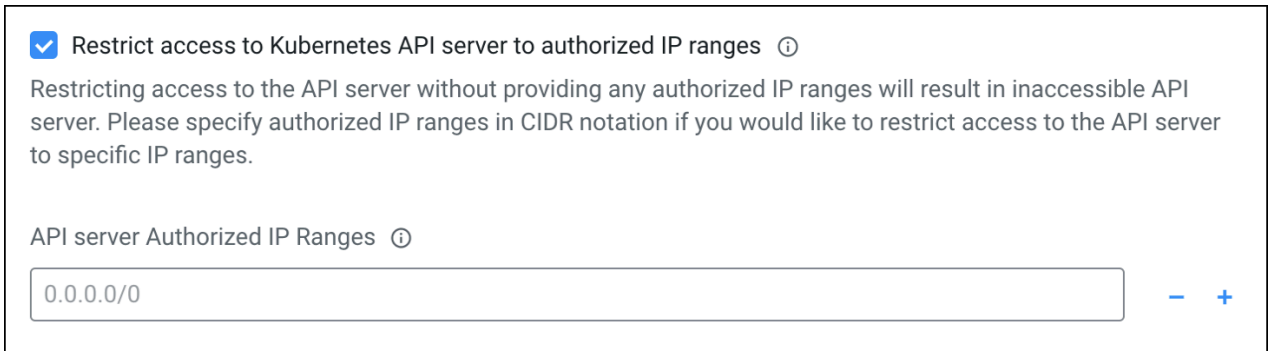

Specific guidelines for restricting access to Kubernetes API server and workloads are detailed in Restricting access for Cloudera services that create their own security groups on AWS by each Cloudera data service.

Within the EKS cluster, there are several security groups defined to facilitate EKS control plane-pod communication, inter-pod and inter-worker node communication as well as workload communication through ELBs. These groups are in accordance with AWS documentation (see Amazon EKS security group considerations.)

Outbound connectivity requirements

Outbound traffic from the worker nodes is unrestricted and is targeted at other AWS services and Cloudera services. The comprehensive list of services that get accessed from a Cloudera environment can be found in AWS documentation (see Amazon EKS security group considerations).

DNS

This topic covers recommended DNS configurations for Cloudera Public Cloud for AWS.

The previous sections dealt with how connectivity is established to the workload infrastructure. This section deals with ‘addressability’. The workloads launched by Cloudera contain a few services that need to be accessed by the Cloudera admins or data consumers. These include services like Cloudera Manager, metadata services like the Hive Metastore, Atlas or Ranger, data processing or consumption services such as Oozie server, Hue, and so on. Given the nature of the cloud infrastructure, the IP addresses for the nodes running these services may change (for example, if the infrastructure is restarted or repaired). However, these should have statically addressable DNS names so that users can access them with the same names.

In order to help with this, Cloudera assigns DNS names to these nodes. These naming schemes have the following properties:

- The DNS name is of the following format for each Data Lake node, Cloudera Data Hub node, and the Data Lake/Cloudera Data Hub cluster endpoint: <CLUSTER_NAME>-{<HOST_GROUP><i>}.<ENVIRONMENT_IDENTIFIER>.<CUSTOMER_IDENTIFIER>.cloudera.site

  An example could be my-dataeng-master0.my-envir.aaaa-1234.cloudera.site

  This name has the following components:

  - The base domain is cloudera.site. This is a publicly registered DNS suffix (see Public Suffix List). It is also a registered Route53 hosted zone in a Cloudera owned AWS account.
  - The <CUSTOMER_IDENTIFIER> is unique to a customer account on Cloudera made of alphanumeric characters and "-"
  - The <ENVIRONMENT_IDENTIFIER> is generated based on the environment name and is truncated to 8 characters.
  - The <CLUSTER_NAME> is the cluster name given to the Data Lake or Cloudera Data Hub. It is appended with a <HOST_GROUP> name such as "gateway", "master", "worker", and so on, depending on the role that the node plays in the cluster. If there are more than one of these nodes playing the same role, they are appended with a serial number, <i>.

- The DNS name of the endpoints of the Cloudera data services is of the following format:
  - For a Virtual Warehouse in Cloudera Data Warehouse, it is <VIRTUAL_WAREHOUSE_NAME>.<CDW_ENVIRONMENT_IDENTIFIER>.dw.<CUSTOMER_IDENTIFIER>.cloudera.site
    - The <VIRTUAL_WAREHOUSE_NAME> is the name of the Virtual Warehouse created. There could be multiple virtual warehouses for a given Cloudera environment.
    - The <CDW_ENVIRONMENT_IDENTIFIER> is the identifier for the Cloudera environment.
  - For a Session Terminal in a Cloudera AI workspace, it is <TTY_SESSION_ID>.<CML_WORKSPACE_ID>.<ENVIRONMENT_IDENTIFIER>.<CUSTOMER_IDENTIFIER>.cloudera.site
    - The <TTY_SESSION_ID> is the ID of the Cloudera AI Terminal Session ID.
    - The <CML_WORKSPACE_ID> is the ID of the Cloudera AI workspace created.
    - The <ENVIRONMENT_IDENTIFIER> is generated based on the environment name and is truncated to 8 characters. If the 8th character is a "-" (dash), then it is truncated to 7 characters instead.
  - For all the Cloudera data services listed above, the common portions of the DNS include.
    - The base domain is cloudera.site. This is a publicly registered DNS suffix (see Public Suffix List). It is also a registered Route53 hosted zone in a Cloudera owned AWS account.
    - The <CUSTOMER_IDENTIFIER> is unique to a customer account on Cloudera made of alphanumeric characters and a "-" (dash).
  - For a virtual cluster in Cloudera Data Engineering, it is <VIRTUAL_CLUSTER_ID>.<CDE_SERVICE_ID>.<ENVIRONMENT_IDENTIFIER>.<CUSTOMER_IDENTIFIER>.cloudera.site
    - The <VIRTUAL_CLUSTER_ID> is the 8-character ID of the Cloudera Data Engineering virtual cluster, for example, afg57p98.
    - The <CDE_SERVICE_ID> is the ID of the Cloudera Data Engineering service containing the virtual cluster, for example, cde-g6th4kjv.
    - The <ENVIRONMENT_IDENTIFIER> is generated based on the Cloudera environment name and is truncated to 8 characters. if the 8th character is a "-" (dash), then it is truncated to 7 characters instead.
  - For a DataFlow service in Cloudera DataFlow, it is

    dfx.<CDF_WORKLOAD_ENDPOINT_ID>.<CUSTOMER_IDENTIFIER>.cloudera.site
    - The <CDF_WORKLOAD_ENDPOINT_ID> is the 8-character ID of the Cloudera DataFlow Service Workload Endpoint, for example, 1bxt50kk.
  - For a database in Cloudera Operational Database, it is <COD_WORKLOAD_NAME>-{<HOST-GROUP><i>}.<ENVIRONMENT_IDENTIFIER>.<CUSTOMER_IDENTIFIER>.cloudera.site
    - <COD_Workload_Name> is the ID of the Cloudera Operational Database, for example, cod-1m6yz9uwqhrg2.
    - The user provides a database name and the environment where they want to create the database. These two entities are hashed together to create the internal <COD_WORKLOAD_NAME>, which is set as the Cloudera Data Hub cluster.
    - Except for the <COD_WORKLOAD_NAME>, the rest of the DNS name of the endpoint is implemented as per Cloudera Data Hub DNS format as mentioned above.
  - For all the Cloudera data services listed above, the common portions of the DNS include:
    - The base domain is cloudera.site. This is a publicly registered DNS suffix. It is also a registered Route53 hosted zone in a Cloudera owned AWS account.
    - The <CUSTOMER_IDENTIFIER> is unique to a customer account on Cloudera made of alphanumeric characters and a "-" (dash).
- The length of the DNS name is restricted to 64 characters due to some limitations with Hue workloads.
- These names are stored as A records in the Route53 hosted zone in the Cloudera managed Cloudera Control Plane AWS account. Hence, you can resolve these names from any location outside of the VPC. However, note that they would still resolve to private IP addresses and hence are constrained by the connectivity setup described in preceding sections.

- Within a Cloudera environment, the DNS resolution happens differently. Every Cloudera environment has a DNS server that is played by a component called FreeIPA. This server is seeded with the hostnames of the nodes of all workload clusters in the environment. Every node in a Data Lake, Cloudera Data Hub, and a Cloudera data service is configured to look up the FreeIPA DNS service for name resolution within the cluster.

DHCP option set

This topic covers recommended DHCP Option Set configurations for Cloudera Public Cloud for AWS.

- Ensure the DHCP Option Set has the DNS Server set to AmazonProvidedDNS. This is required for EFS (see Mounting on Amazon EC2 with a DNS name) and EKS (see Amazon EKS cluster endpoint access control).
- Customers may choose to use their own DNS servers from their corporate network in the DHCP option set. However, the DNS servers need to be configured to conditionally forward DNS queries for the Amazon subdomains to Amazon Provided DNS resolver in the VPC. This also helps ensure the Cloudera connectivity to AWS services. If the Cloudera data services do not allow activation with the custom DNS servers, please contact Cloudera support to get the relevant entitlements added.
- Ensure the DHCP Option Set associated with the VPC has only one domain and it is recommended that it remains default.

Determining the CIDR range

This topic covers options for determining the CIDR range for Cloudera Public Cloud for AWS.

When registering an AWS environment in Cloudera, you are asked to select a VPC and one or more subnets. You have two options:

1. Cloudera will create a new VPC and subnets.
2. Select an existing VPC.

In both cases, use this guide to calculate and verify the limits of the VPC and subnets available in your AWS subscription to ensure that you have enough networking resources to create clusters in Cloudera.

Option 1: Cloudera creates the VPCs and subnets

If you would like Cloudera to create a new VPC, you will need to specify a /16 CIDR in IPv4 range that will be used to define the range of private IPs for VM instances provisioned into these subnets. The default is 10.10.0.0/16.

By default Cloudera creates 6 subnets (3 private and 3 public) and divides the address space as follows:

- 3 x /19 private subnets for FreeIPA, Data Lake, Cloudera Data Hub, Cloudera Data Warehouse, Cloudera AI, Cloudera Data Engineering, Cloudera DataFlow, Cloudera Operational Database.
- 3 x /24 public subnets reserved for future use.
- Cloudera creates an internet gateway for your VPC and deploys a NAT gateway in each public subnet and configures it in the route table of each private subnet.
- All the resources created in the VPC are assigned with relevant tags (like ‘kubernetes.io/role/internal-elb:1’ for private subnets and ‘kubernetes.io/role/elb:1’ for public subnets).

Option 2: Existing VPC and subnets

If you would like to use an existing VPC, the subnet requirements vary based on the services used. This section is a guide for calculating network requirements per service.

Subnets for Data Lake and Cloudera Data Hub

Both Data Lake and Cloudera Data Hub share the same subnet, but due to Data Lake’s dependency on RDS, Cloudera admins need at least two subnets.

It is recommended to use subnets of /19 CIDR. If you would like to use a smaller subnets, use the following guidelines:

- One IP address is used for each VM.
- Two IPs for your RDS instances.
- One Light Duty Data Lake cluster uses 2 VMs and 1 additional VMs for FreeIPA.
- One Medium Duty Data Lake cluster uses 10 VMs and 3 additional VMs for FreeIPA.
- The exact number of VMs depends on the Cloudera Data Hub's cluster definition, but a typical Cloudera Data Hub cluster uses a minimum of four VMs as a starting point and this number can be dynamically scaled up or down.
- Make sure you allocate enough IPs to handle each cluster running at peak capacity

Subnets for Cloudera Data Warehouse

The Cloudera Data Warehouse service needs three subnets. You can choose the specific subnets that should be used when you activate Cloudera Data Warehouse for an environment and if the specified subnets are shared with other Cloudera data services then ensure that there are sufficient IPs left for Cloudera Data Warehouse.

An environment can be activated for Cloudera Data Warehouse with Overlay Network or without Overlay Network, the next section outlines the benefits and the pitfalls of one choice over the other.

Using Cloudera Data Warehouse with Overlay Network vs without Overlay Network

Enabling overlay networks creates two network spaces in your EKS:

- A node network space, which derives per-node IP addresses from the VPC.
- A Kubernetes pod network space, which derives per-pod IP addresses from the CNI plugin's own network space.

The overlay network is bridged into the node network. As a result, one IP address is required per node instead of one IP address needed per pod. It is recommended to enable overlay networks with Cloudera Data Warehouse if the available IPs in your subnets is less than 1024.

Even though using an overlay network for Cloudera Data Warehouse requires less number of IPs it adds more metadata to each network packet which even though are processed in the kernel can have a performance impact. Due to this extra metadata added to each network packet it may not be straightforward to debug incoming and outgoing traffic with some of the network debugging tools. To employ an overlay network for Cloudera Data Warehouse, Weave net CNI (Container Network Interface) plugin is used. Since weave-net is a full-mesh network where each node is connected to every other node in the mesh, the number of nodes the overlay network can span across is limited to 200.

Cloudera Data Warehouse with Overlay Networks

Use these guidelines to arrive at your desired subnet CIDR if you choose to use overlay networks. It is recommended to enable overlay networks with Cloudera Data Warehouse if the available IPs in your subnets is less than 1024.

VM type	No of VMs	Total no of IPs addresses required
Cloudera Data Warehouse Shared Services - (Shared among all VWs in an environment)	3	3
Per Database Catalog (One catalog is created by default, you can create additional catalogs)	2	2
Shared Services per Virtual Warehouse (HS2, Hue, DAS, coordinators, catalog, statestored, and so on.)	3	3
Per Virtual Warehouse (XS) *	2 to 20	2 to 20
Per Virtual Warehouse (S) *	10 to 100	10 to 100
Per Virtual Warehouse (M) *	20 to 200	20 to 200
Per Virtual Warehouse (L) *	40 to 400	40 to 400
Per Virtual Warehouse (Custom)*	x to 10x (where x is the initial node count)	x to 10x (where x is the initial node count)

* Each autoscaling activity can be treated as deploying a new Virtual Warehouse. For example, when a XS Virtual Warehouse is scaled once, it uses four VMs instead of two.

Cloudera Data Warehouse without Overlay Network

Use these guidelines to arrive at your desired subnet CIDR if you choose to not to use overlay networks while activating your DW environment.

VM type	No of VMs	Maximum no of pods per VM	Maximum number of IPs per VM (No of pods per VM +1 per VM)	Maximum no of IPs addresses required
Cloudera Data Warehouse Shared Services - (Shared among all VWs in an environment)	3	25	26	78
Per Database Catalog (One catalog is created by default, you can create additional catalogs)	2	25	26	52
Shared Services per Virtual Warehouse (HS2, Hue, DAS, coordinators, catalog, statestored, and so on.)	3	25	26	78
Per Virtual Warehouse (XS) *	2 to 20	10	11	22 to 220
Per Virtual Warehouse (S) *	10 to 100	10	11	110 to 1100
Per Virtual Warehouse (M) *	20 to 200	10	11	220 to 2200
Per Virtual Warehouse (L) *	40 to 400	10	11	440 to 4400
Per Virtual Warehouse (Custom)*	x to 10 x (where x is the initial node count)	10	11	11x to 110x

* Each autoscaling activity can be treated as deploying a new Virtual Warehouse. For example, when a XS Virtual Warehouse is scaled once, it uses four VMs instead of two.

Query Isolation

If the Query Isolation feature has been enabled for a Virtual Warehouse and a query scans more than the threshold set in the hive.query.isolation.scan.size.threshold parameter, the planner runs the query in isolation. This means that an isolated standalone executor group is spawned to run the data-intensive query. The number of executors spawned to run the query does not exceed the default setting for the hive.query.isolation.max.nodes.per.query parameter, which default to 2 times the virtual warehouse template size. The number of isolated parallel queries (can go up to 400) and number of nodes per isolated queries (can go up to 400) is configurable for a given virtual warehouse. Each of these nodes can consume up to 11 IPs when overlay networks are not enabled and 1 IP if the overlay network is enabled against your environment.

Subnets for Cloudera AI

Cloudera AI requires at least two subnets in two different availability zones and you can choose which subnets should be used by a workbench at the time of provisioning. If the specified subnets are shared with other Cloudera data services then ensure that there are sufficient IPs left for Cloudera AI. Cloudera AI uses Calico CNI (Container Network Interface) to run Cloudera AI pods in an overlay network. The formula to calculate IP Addresses per workbench is as follows:

- Each workbench can grow up to 100 CPU worker nodes and 100 GPU workers; each node consumes 1 IP address.
- In addition, you will need to allocate 9 IP addresses for infrastructure nodes (3 IPs for Cloudera AI infra nodes, 4 for Liftie infra nodes and 2 load balancers).

Subnets for Cloudera Data Engineering

Cloudera Data Engineering requires at least two subnets in two different availability zones. You choose at the time of workspace provisioning which subnets should be used. Ensure that there are adequate IPs left for Cloudera Data Engineering to provide for the maximum expected size of the cluster. Cloudera Data Engineering uses Calico CNI (Container Network Interface) to run pods in an overlay network. A /24 CIDR is recommended for Cloudera Data Engineering subnets, but for a custom range the formula to calculate IP addresses per Cloudera Data Engineering service is as follows:

- Each Cloudera Data Engineering service can scale up to 100 compute nodes, each node consumes one IP address.
- In addition, you need to allocate 5 IP addresses for the infrastructure nodes (1 IP for Cloudera Data Engineering infra node, 4 for Liftie infra nodes) and 2 IP addresses per virtual cluster for the virtual cluster service nodes.

Subnets for Cloudera DataFlow

Cloudera DataFlow requires at least two subnets in two different availability zones. Cloudera DataFlow by default configures EKS to run in private subnets, if they are available. Cloudera DataFlow uses Calico CNI (Container Network Interface) to run pods in an overlay network. The CIDR block for the subnets must be sized appropriately in each Cloudera DataFlow environment to accommodate the following:

- Each Cloudera DataFlow cluster can grow up to 50 autoscaling compute instances, each of which consumes 1 IP address.
- A fixed overhead of 48 IP addresses for three instances for core Cloudera DataFlow services.

Subnets for Cloudera Operational Database

Cloudera Operational Database currently leverages Cloudera Data Hub to deploy infrastructure in a private subnet. Clients associated with HBase REST server, Thrift Server, or the Phoenix Thin server can be proxied via the VPN gateway. If Apache HBase Java API or Apache Phoenix Thick JDBC client are used, an edge node must be configured to access the private computing resources. The formula to calculate IP addresses per Cloudera Operational Database database is as follows:

- Each Cloudera Operational Database database defaults to a minimum of 9 nodes (1 leader, 2 master, 1 gateway, and 5 worker nodes), requiring 9 IP addresses.
- Each Cloudera Operational Database database can autoscale the number of worker nodes given various factors/attributes. The range defaults to a minimum of 5 nodes and a maximum of 20 nodes. Users can reduce the minimum to 3 nodes, but there is currently no fixed upper limit. They must consider the behaviour of their database while providing the CIDR range to accommodate for the potential autoscaling growth, where each node takes 1 IP address.
- In addition, configuring an edge node, if required for client applications, takes 1 IP address for 1 accessory node.

DNS

This topic covers recommended DNS configurations for Cloudera Public Cloud for AWS.

The previous sections dealt with how connectivity is established to the workload infrastructure. This section deals with ‘addressability’. The workloads launched by Cloudera contain a few services that need to be accessed by the Cloudera admins or data consumers. These include services like Cloudera Manager, metadata services like the Hive Metastore, Atlas or Ranger, data processing or consumption services such as Oozie server, Hue, and so on. Given the nature of the cloud infrastructure, the IP addresses for the nodes running these services may change (for example, if the infrastructure is restarted or repaired). However, these should have statically addressable DNS names so that users can access them with the same names.

In order to help with this, Cloudera assigns DNS names to these nodes. These naming schemes have the following properties:

- The DNS name is of the following format for each Data Lake node, Cloudera Data Hub node, and the Data Lake/Cloudera Data Hub cluster endpoint: <CLUSTER_NAME>-{<HOST_GROUP><i>}.<ENVIRONMENT_IDENTIFIER>.<CUSTOMER_IDENTIFIER>.cloudera.site

  An example could be my-dataeng-master0.my-envir.aaaa-1234.cloudera.site

  This name has the following components:

  - The base domain is cloudera.site. This is a publicly registered DNS suffix (see Public Suffix List). It is also a registered Route53 hosted zone in a Cloudera owned AWS account.
  - The <CUSTOMER_IDENTIFIER> is unique to a customer account on Cloudera made of alphanumeric characters and "-"
  - The <ENVIRONMENT_IDENTIFIER> is generated based on the environment name and is truncated to 8 characters.
  - The <CLUSTER_NAME> is the cluster name given to the Data Lake or Cloudera Data Hub. It is appended with a <HOST_GROUP> name such as "gateway", "master", "worker", and so on, depending on the role that the node plays in the cluster. If there are more than one of these nodes playing the same role, they are appended with a serial number, <i>.

- The DNS name of the endpoints of the Cloudera data services is of the following format:
  - For a Virtual Warehouse in Cloudera Data Warehouse, it is <VIRTUAL_WAREHOUSE_NAME>.<CDW_ENVIRONMENT_IDENTIFIER>.dw.<CUSTOMER_IDENTIFIER>.cloudera.site
    - The <VIRTUAL_WAREHOUSE_NAME> is the name of the Virtual Warehouse created. There could be multiple virtual warehouses for a given Cloudera environment.
    - The <CDW_ENVIRONMENT_IDENTIFIER> is the identifier for the Cloudera environment.
  - For a Session Terminal in a Cloudera AI workspace, it is <TTY_SESSION_ID>.<CML_WORKSPACE_ID>.<ENVIRONMENT_IDENTIFIER>.<CUSTOMER_IDENTIFIER>.cloudera.site
    - The <TTY_SESSION_ID> is the ID of the Cloudera AI Terminal Session ID.
    - The <CML_WORKSPACE_ID> is the ID of the Cloudera AI workspace created.
    - The <ENVIRONMENT_IDENTIFIER> is generated based on the environment name and is truncated to 8 characters. If the 8th character is a "-" (dash), then it is truncated to 7 characters instead.
  - For all the Cloudera data services listed above, the common portions of the DNS include.
    - The base domain is cloudera.site. This is a publicly registered DNS suffix (see Public Suffix List). It is also a registered Route53 hosted zone in a Cloudera owned AWS account.
    - The <CUSTOMER_IDENTIFIER> is unique to a customer account on Cloudera made of alphanumeric characters and a "-" (dash).
  - For a virtual cluster in Cloudera Data Engineering, it is <VIRTUAL_CLUSTER_ID>.<CDE_SERVICE_ID>.<ENVIRONMENT_IDENTIFIER>.<CUSTOMER_IDENTIFIER>.cloudera.site
    - The <VIRTUAL_CLUSTER_ID> is the 8-character ID of the Cloudera Data Engineering virtual cluster, for example, afg57p98.
    - The <CDE_SERVICE_ID> is the ID of the Cloudera Data Engineering service containing the virtual cluster, for example, cde-g6th4kjv.
    - The <ENVIRONMENT_IDENTIFIER> is generated based on the Cloudera environment name and is truncated to 8 characters. if the 8th character is a "-" (dash), then it is truncated to 7 characters instead.
  - For a DataFlow service in Cloudera DataFlow, it is

    dfx.<CDF_WORKLOAD_ENDPOINT_ID>.<CUSTOMER_IDENTIFIER>.cloudera.site
    - The <CDF_WORKLOAD_ENDPOINT_ID> is the 8-character ID of the Cloudera DataFlow Service Workload Endpoint, for example, 1bxt50kk.
  - For a database in Cloudera Operational Database, it is <COD_WORKLOAD_NAME>-{<HOST-GROUP><i>}.<ENVIRONMENT_IDENTIFIER>.<CUSTOMER_IDENTIFIER>.cloudera.site
    - <COD_Workload_Name> is the ID of the Cloudera Operational Database, for example, cod-1m6yz9uwqhrg2.
    - The user provides a database name and the environment where they want to create the database. These two entities are hashed together to create the internal <COD_WORKLOAD_NAME>, which is set as the Cloudera Data Hub cluster.
    - Except for the <COD_WORKLOAD_NAME>, the rest of the DNS name of the endpoint is implemented as per Cloudera Data Hub DNS format as mentioned above.
  - For all the Cloudera data services listed above, the common portions of the DNS include:
    - The base domain is cloudera.site. This is a publicly registered DNS suffix. It is also a registered Route53 hosted zone in a Cloudera owned AWS account.
    - The <CUSTOMER_IDENTIFIER> is unique to a customer account on Cloudera made of alphanumeric characters and a "-" (dash).
- The length of the DNS name is restricted to 64 characters due to some limitations with Hue workloads.
- These names are stored as A records in the Route53 hosted zone in the Cloudera managed Cloudera Control Plane AWS account. Hence, you can resolve these names from any location outside of the VPC. However, note that they would still resolve to private IP addresses and hence are constrained by the connectivity setup described in preceding sections.

- Within a Cloudera environment, the DNS resolution happens differently. Every Cloudera environment has a DNS server that is played by a component called FreeIPA. This server is seeded with the hostnames of the nodes of all workload clusters in the environment. Every node in a Data Lake, Cloudera Data Hub, and a Cloudera data service is configured to look up the FreeIPA DNS service for name resolution within the cluster.

Associating additional CIDRs to a VPC

You can choose to add another CIDR to your VPC in case you are close to exhausting the available IPs in your VPC by following the necessary steps from Associate a secondary IPv4 CIDR block with your VPC. Once a new CIDR has been added to your VPC, a created environment not will pick these changes automatically. For the environment to pick up the new CIDR, follow these steps.

Procedure

1. Go to your environment page and under Summary tab click on the refresh icon in the Network card.

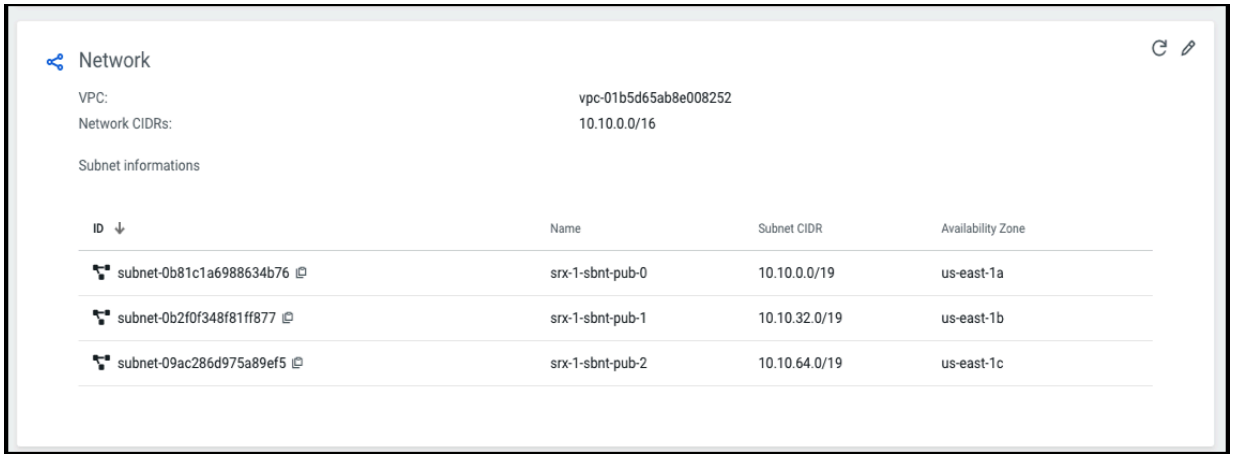

2. To add any new subnets created with the new CIDR, click on the pencil icon and choose the new subnet from the dropdown.

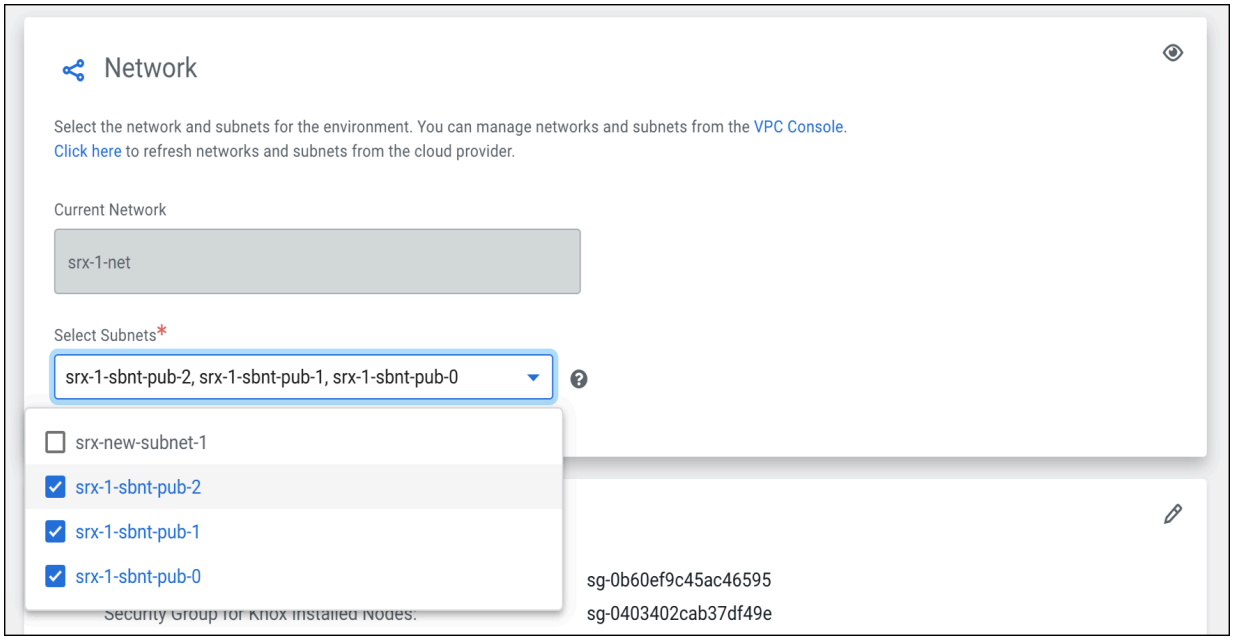

Results

The user can now use the new subnet for any subsequent workloads that will be created.

Cloudera Private Links Network for AWS

Cloudera Private Links Network enables you to connect privately and securely to the Cloudera Control Plane without traversing the internet. You can use Cloudera Private Links Network for end-to-end encryption of your workloads between Cloudera Control Plane and AWS VPC endpoints.

This documentation provides the following details and steps about Cloudera Private Links Network:

- High-level options of VPC endpoint placement
- Configuration options and other considerations for DNS overrides
- Cloudera Private Links Network deployment process
- Instructions of how to set up both Private Link options:
  - VPC: Setup of Cloudera Private Links Network for a workload VPC through CDP CLI
  - Authorization: Authorization with CDP CLI to enable the setup of Cloudera Private Links Network through your automation tools
- References for proxy profile configuration and considerations, and Cloudera Private Links Network commands

Note: Cloudera Private Links Network can be obtained through an entitlement, reach out to Cloudera Support for more information.

Comparison of connectivity setup without and with Cloudera Private Links Network

Without Cloudera Private Links Network, your workload environment communicates with the Cloudera Control Plane through the internet. This traffic may optionally flow through a managed egress proxy. The following two diagrams illustrate this:

Figure 1: Connectivity from workload environment to Cloudera Control Plane via internet

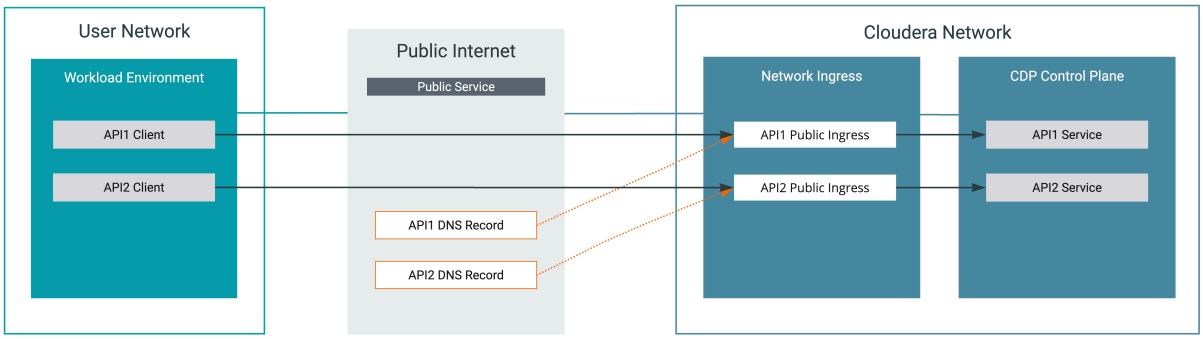

Figure 2: Connectivity from workload Environment to Cloudera Control Plane via internet and egress proxy

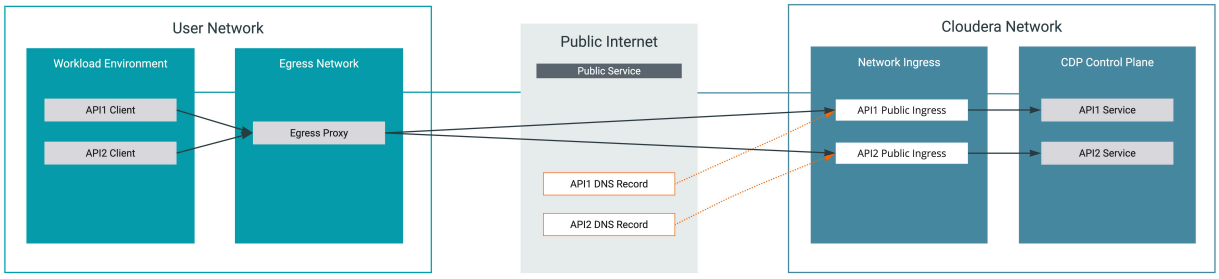

With Cloudera Private Links Network, the Cloudera Control Plane is accessed as if the Cloudera Control Plane would be on your network. This means that IP addresses are assigned to the Cloudera Control Plane services from your network, and DNS lookups will return your local IP addresses.

To ensure private connectivity through network ingress between the workload environment and Cloudera Control Plane, VPC endpoints can be added. The following illustration details the scenario where the VPC endpoints are in the same VPC as your workload environment. In this case, the VPC endpoints receive IPs from the workload environment VPC subnets:

Figure 3: VPC endpoint in workload environment VPC

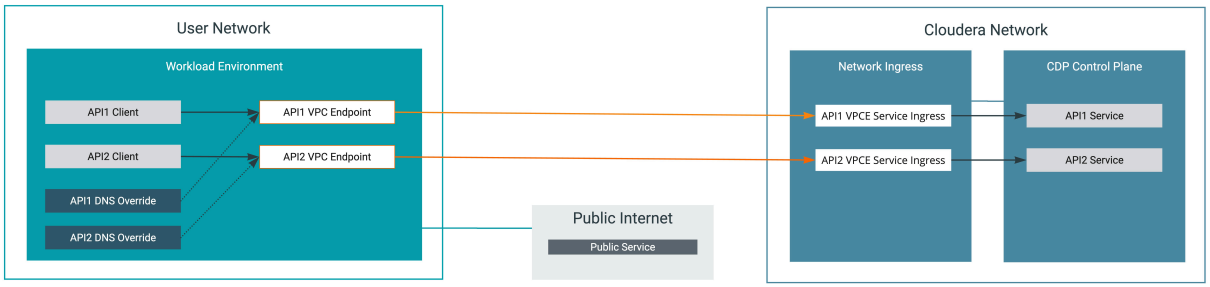

The following options are available for DNS overrides:

- DNS is a per-VPC view: Deploying a VPC endpoint with private DNS option enabled will automatically install the DNS overrides in the local VPC.
- DNS is a regional or global view: Installing overrides at a regional or global scope will impact DNS resolution for other VPCs, other VPCs will attempt to use the VPC endpoints of the local VPC.

This section does not include an exhaustive list of design options, but should cover most cases. For more information about more advanced use cases, see the Additional VPC scenarios section.

Cloudera Private Links Network creates an additional layer of security with Private Links and security groups with inbound rule. If security groups are already present in your network, they will be updated with an inbound rule for safe HTTPs traffic. As inbound rules are created or updated within setting up the Cloudera Private Link Network, outbound rules can be removed.

DNS overrides

The DNS hostnames for Cloudera services remain the same, whether the services are accessed through the internet or through VPC endpoints. DNS resolution in the customer networks perform DNS lookups against the public internet. Responses will contain the public internet IP addresses for the Cloudera services. DNS override records must be set up in your DNS infrastructure, and the hostnames need to be repointed to the VPC endpoint or IP addresses. For more information, see *Setting up DNS overrides*.

Supported service components

Review the list of supported service components.

Currently, Cloudera supports multiple service components for which Cloudera Private Links Network can be created:

- Cloudera API
- Databus API
- Cluster Connectivity Manager v2
- Console authentication
- Monitoring, which includes Cloudera Observability

The following table describes the available service components for each region:

Cloudera Control Plane Region	VPCE service region in AWS	Available VPCE service components
us-west-1 (USA)	eu-west-1	API, DBUSAPI, CCMV2, CONSOLEAUTH, MONITORING
us-west-1 (USA)	ap-southeast-1	API, DBUSAPI, CCMV2, CONSOLEAUTH, MONITORING
us-west-1 (USA)	eu-central-1	API, DBUSAPI, CCMV2, CONSOLEAUTH, MONITORING
us-west-1 (USA)	us-east-1	API, DBUSAPI, CCMV2, CONSOLEAUTH, MONITORING
us-west-1 (USA)	us-east-2	API, DBUSAPI, CCMV2, CONSOLEAUTH, MONITORING
us-west-1 (USA)	us-west-2	API, DBUSAPI, CCMV2, CONSOLEAUTH, MONITORING
eu-1 (Germany)	eu-west-1	API, CCMV2
eu-1 (Germany)	eu-central-1	API, CCMV2
ap-1 (Australia)	ap-southeast-1	API, CCMV2
ap-1 (Australia)	ap-southeast-2	API, CCMV2

Note: In case you need to use a region that is not mentioned in the supported region table, reach out to Cloudera Support.

Setting up Cloudera Private Links Network for AWS environments

Learn about how to set up Cloudera Private Links Network using CDP CLI for both the VPC and Authorization options.

You can configure private links from the workload environment to Cloudera Control Plane using Cloudera Private Links Network. You have the following options to choose from:

- VPC option: Using CDP CLI to set up private links in a Cloudera environment network. Cloudera automation assists you with authorizing and end-to-end setup. Requires cross-account permissions.
- Authorization option: Using your own network automation to create private links in a Cloudera environment. CDP CLI is used to authorize your cloud provider account to connect to Cloudera. Does not require cross-account permissions.

Prerequisites

Before you begin, review the following prerequisites and requirements.

The following requirements must be met before creating Cloudera Private Links Network:

- Each environment must have its own private links configured.
- The private links should be set up before creating an environment.
- Configuring private links in Cloudera can only be performed through CDP CLI. You need to have the Beta CDP CLI installed and configured. See Installing Beta CDP CLI.
- In order to configure private links in Cloudera through CDP CLI, you must have the EnvironmentCreator or PowerUser role in Cloudera.

AWS IAM requirements (VPC option only)

You must have AWS admin privileges to create Cloudera Private Links Network with VPC option. You need to have the ability to update the cross-account access IAM role to include the policy required for private links.

Note: This is only required if you create private links with the VPC option. If you use your own network automation to create private links, you do not need to add this additional AWS policy.

Cloudera documents two types of policies: the default policy and the minimum access policies. If you are using the former, you do not need to do anything. If you are using the latter, add the following additional policy that will allow the creation of private links:

```
{
    "Sid": "ElasticComputeCloudFull",
    "Action": [
        "ec2:*"
        ],
    "Effect": "Allow",
    "Resource": [
        "*"
    ]
},
{
    "Sid": "IdentityAccessManagementFull",
    "Action": [
        "iam:AttachRolePolicy",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam:DeleteInstanceProfile",
        "iam:DeleteRole",
        "iam:ListRolePolicies",
        "iam:GetInstanceProfile",
        "iam:GetRolePolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListInstanceProfiles",
        "iam:PutRolePolicy",
        "iam:PassRole",
        "iam:GetRole",
        "iam:AddRoleToInstanceProfile",
        "iam:RemoveRoleFromInstanceProfile",
        "iam:DetachRolePolicy",
        "iam:DeleteRolePolicy",
        "iam:SimulatePrincipalPolicy",
        "iam:ListRoles"
    ],
    "Effect": "Allow",
    "Resource": [
"*"
    ]
},
{
    "Sid": "IdentityAccessManagementLimited",
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Effect": "Allow",
    "Resource": [
        "arn:aws:iam::*:role/aws-service-role/*"
    ]
}
```

```
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:ModifyVpcEndpointServicePermissions",
                "ec2:ModifyVpcEndpointServiceConfiguration",
                "ec2:CreateVpcEndpointConnectionNotification",
```

```
                "ec2:DeleteVpcEndpoints",
                "ec2:CreateTags",
                "ec2:ModifySecurityGroupRules",
                "ec2:DescribeVpcEndpointServices",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:DeleteVpcEndpointServiceConfigurations",
                "ec2:ModifyVpcEndpointConnectionNotification",
                "ec2:DescribeSecurityGroups",
                "ec2:CreateVpcEndpointServiceConfiguration",
                "ec2:DescribeVpcEndpointServicePermissions",
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeVpcs",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteSecurityGroup",
                "route53:AssociateVPCWithHostedZone",
                "ec2:DescribeVpcEndpoints",
                "ec2:ModifyVpcEndpoint"
            ],
            "Resource": "*"
        }
    ]
}
```

This is only required for the Cloudera Private Links Network setup. Once Cloudera Private Links Network is created, you can revert back to the minimum access policy.

Setting up DNS overrides

If your private DNS is disabled, you need to set up the DNS manually, adding the specified DNS entries for the respective service component.

About this task

Note: If the workload VPC is the same as VPC where the endpoint is deployed, you need to update the existing endpoints with "enablePrivateDns" : "true" through CDP CLI and the manual DNS override steps mentioned below are not required.

In the example below, a private hosted zone named cloudera.com is created for the VPC, and an alias record with Route traffic to the Cluster Connectivity Manager v2 VPC endpoint is created.

Procedure

1. Create private hosted zones in Amazon Route 53 with the names below for the VPC endpoint region.

   The following example shows how the endpoints should look like if the Cloudera Control Plane region us-west-1.

Service component	Endpoint
API	us-west-1.cdp.cloudera.com
CCMV2	v2.us-west-1.ccm.cdp.cloudera.com
DBUSAPI	us-west-1.sigma.altus.cloudera.com
CONSOLEAUTH	us-west-1.cdp.cloudera.com
MONITORING	api.monitoring.us-west-1.cdp.cloudera.com

2. Create a DNS record in Amazon Route 53 for each of the private hosted zones created in the previous step. You also need to add an alias record and route traffic to the respective endpoint.
   a) For the API endpoint: *.us-west-1.cdp.cloudera.com
   b) For the CCMV2 endpoint: *.v2.us-west-1.ccm.cdp.cloudera.com
   c) For the DBUSAPI endpoint: dbusapi.us-west-1.sigma.altus.cloudera.com

   The following is an example record routing traffic to the endpoint:

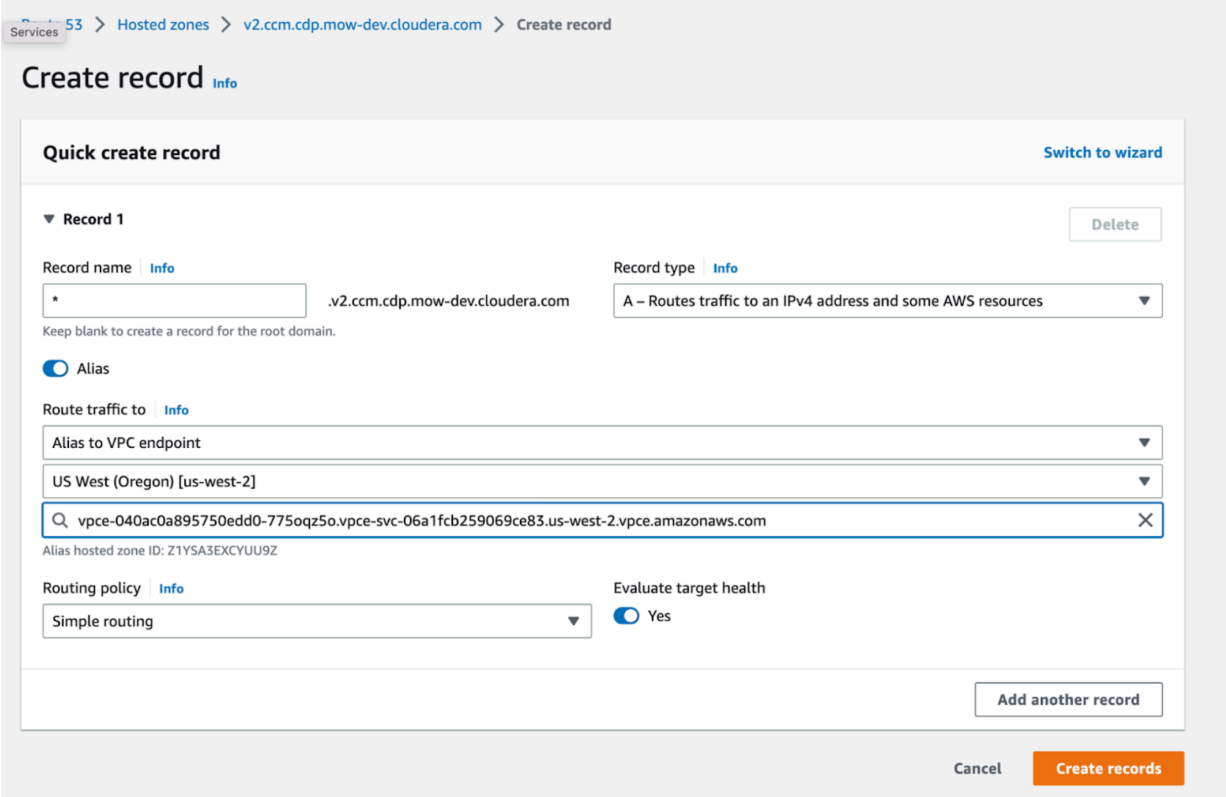

Note: The VPC endpoint name to be routed is available in CDP CLI response of the respective endpoint.

Creating Cloudera Private Links Network with VPC option

Learn about how to create Cloudera Private Links Network using the VPC option.

Required Role: EnvironmentCreator or PowerUser

You need to use the following CDP CLI command to create the Cloudera Private Links Network with VPC option:

```
cdp cloudprivatelinks create-private-link-endpoint
```

This command is used to create a private link endpoint in your workload VPC. This establishes the private link connectivity between the VPC endpoint created and the respective VPC endpoint service present in the Cloudera Control Plane. The private link will be created for all supported components.

The following parameters should be specified:

Parameter	Description
enablePrivateDns	Specifies whether private DNS needs to be enabled on the VPC endpoint. **Note:** As mentioned in Setting up DNS override, if you have a workload VPC that is different from the VPC where the endpoint is deployed, it is recommended that you set this value to false in the request and perform the DNS override.

Parameter	Description
awsAccountDetails	This should have the following fields • cloudAccountId - Your AWS account ID where the private endpoints are created. The account ID needs to be provided, because Cloudera needs to authorize the account for private link service access. • crossAccountRoleDetails or credentialCrn - The crossAccount RoleDetails can be provided with explicit crossAccountRole and externalId. You can also use credentialCrn that is configured with default policy or reduced access policy for provisioning the VPC endpoint. • region - You can provide the AWS region where the private endpoint needs to be created. • vpcId - The VPC ID in which private endpoint needs to be createdsubnetIds - IDs of the private subnets in which a private endpoint needs to be created • subnetIds - It is recommended to provide the private subnet from each of the available zones. Without a specific subnet, there is no way to associate the private endpoint with your VPC resources. This is also required for fault tolerance and high availability.

Note: Enabling private DNS provides DNS lookup of the hostname from within the VPC. You have the option to use endpoints outside of the VPC, and create your own DNS overrides. In this case, you need to disable private DNS.

The following is an example command for creating Cloudera Private Links Network:

```
cdp cloudprivatelinks create-private-link-endpoint --cloud-service-provider
aws
 --cli-input-json '{
  "enablePrivateDns": true,
  "awsAccountDetails": {
  "cloudAccountId": "112313717721"
    "crossAccountRoleDetails": {
      "crossAccountRole":
"arn:aws:iam::112313717721:role/test-privatelinkservice-xaccount-role",
      "externalId": "51962fcf-4205-495d-ab4a-c82c5613a99f"
    },
    "region": "us-west-2",
    "vpcId": "vpc-0d006493309f23c3",
    "subnetIds": [
      "subnet-0e5812d895de60c47"]
  }
}'
```

The executed command performs the following sequence of steps:

1. Identifying the appropriate VPC endpoint service for the request. Existing vpc-svcs are filtered for the requested Cloudera service component and region.
2. Authorizing access to the VPC endpoint service is performed.
3. Creating a security group for the VPC endpoint with the provided ports.
4. Creating a VPC endpoint with the security group created and the VPC endpoint service filtered.

The command returns a trackingId, which can be used to verify that the domains of the respective Cloudera service components are reachable and resolve to private IPs from your VPC using the following command with the returned tracking ID from creating private link endpoints:

```
cdp cloudprivatelinks list-private-link-endpoint-statuses
--tracking-id [***ID***]
```

You can also check this by accessing the **Monitoring** tab of the endpoints in the AWS Console.

Creating Cloudera Private Links Network with Authorization option

Learn about how to create Cloudera Private Links Network using the Authorization option.

Required Role: EnvironmentCreator or PowerUser

You need to use the following CLI command to create the Cloudera Private Links Network using your own network automation:

```
cdp cloudprivatelinks authorize-private-link-service-access
```

This command is used to authorize access to the Private Link services for your cloud account.

The following parameters should be specified:

Parameter	Description
cloudAccountId	Your AWS account ID where the private endpoints are created. The account ID needs to be provided, because Cloudera needs to authorize the account for private link service access.
region	Region of the Cloudera Control Plane.
serviceGroup	Name of the Cloudera service group.

The following is an example command for creating Cloudera Private Links Network:

```
cdp cloudprivatelinks authorize-private-link-service-access --cli-input-json
 '{
"cloudAccountId" : "112313717721",
"region": "us-west-2",
"serviceGroup": "cdp-control-plane",
}'
```

The executed command performs the following sequence of steps:

1. Identifying the appropriate VPC endpoint service for the request. Existing vpc-svcs are filtered for the requested Cloudera service component and region.
2. Authorizing access to the VPC endpoint service is performed.

The command returns the Private Link service name, Cloudera service component and the authorization status as shown in the following example:

```
{
    "authorizePrivateLinkServiceAccessResults": [
        {
            "privateLinkService": "com.amazonaws.vpce.us-west-2.vpce-svc-0
ce5a427a56d80603",
            "serviceComponent": "ccmv2",
            "authorizationStatus": "SUCCESS",
            "availabilityZoneList": "[usw2-az2, usw2-az1, usw2-az3]",
            "vpceClientTcpPortList": "[443]",
            "hostname": "[*.v2.ccm.cdp.mow-dev.cloudera.com]"
        },
    ]
}
```

After executing the command, you have to manually complete the following steps:

1. Create a security group for the VPC endpoint with the returned ports.

   For more information about how to create security groups, see the AWS documentation.
2. Create a VPC endpoint with the previously created security group and the VPC endpoint service filtered.

   For more information about how to create VPC endpoints, see the AWS documentation.

You can verify that the domains of the respective Cloudera service components are reachable and resolve to private IPs from your VPC using the following command with the returned tracking ID from creating private link endpoints:

```
cdp cloudprivatelinks list-private-link-endpoint-statuses
--tracking-id [***ID***]
```

You can also check this by accessing the **Monitoring** tab of the endpoints in the AWS Console.

Deleting Cloudera Private Links Network

You have the option to delete the created Cloudera Private Links Network using CDP CLI. The command first deletes the security groups and then removes the private link endpoints.

Required Role: EnvironmentCreator or PowerUser

Warning: In case the Cloudera Private Links Network is deleted, traffic will be routed through the internet.

If you want to delete the Cloudera Private Links Network, use the following command:

```
cdp cloudprivatelinks delete-private-link-endpoint
--cloud-service-provider aws
--aws-account-info [***ACCOUNT DETAILS***]
```

You can verify that the Cloudera Private Links Network is removed with the following command with the tracking ID returned from deleting the private link endpoints:

```
cdp cloudprivatelinks list-private-link-endpoint-statuses
--tracking-id [***ID***]
```

Troubleshooting Cloudera Private Links Network

Learn more about how to troubleshoot certain errors when creating Cloudera Private Links Network.

The following error message is displayed if the environment creation is failing with a connectivity issue:

```
Please check your connection and proxy settings and make sure the instance c
an reach *.v2.ccm.cdp.cloudera.com
```

In this case, you can login to the FreeIPA instance, and confirm if the respective domain resolves to a private IP. The following domains are available for the Cloudera service component:

Service component	Endpoint
API	api.*[***CLOUDERA CONTROL PLANE REGION***]*.cdp.cl
CCMv2	v2.ccm.*[***CLOUDERA CONTROL PLANE REGION***]*.c
DBUSAPI	dbusapi.*[***CLOUDERA CONTROL PLANE REGION***]*.s
CONSOLEAUTH	consoleauth.thunderhead-*[***ENVIRONMENT NAME***]*.cl
MONITORING	api.monitoring.*[***CLOUDERA CONTROL PLANE REGION*

The *[***CLOUDERA CONTROL PLANE REGION***]* should be one of the following:

- US control plane: us-west-1
- EU control plane: eu-1
- AP control plane: ap-1

References

Learn more about additional information related to Cloudera Private Links Network.

CLI commands for Cloudera Private Links Network

Learn more about the available CLI command for Cloudera Private Links Network.

Command	Description
create-private-link-endpoint	Creating Cloudera Private Links Network with VPC option.
authorize-private-link-service-access	Creating Cloudera Private Links Network with Authorization option. It returns the list of private link service names and their respective authorization status.
list-private-link-endpoint-statuses	This command can be used after creating and deleting Cloudera Private Links Network, or whenever there is a change to the existing setup. The tracking-id returned from the create or delete command needs to be passed to get the status of request.
list-private-link-services-for-region	Returns a list of the private link services that are supported for the AWS region and the service group. It returns the list of private link service names and the mapped service component. You can use this command to check which private link services are supported in your AWS region.
delete-private-link-endpoint	Deleting the previously created Cloudera Private Links Network. First the associated security group is deleted, and then deletes the specified endpoints are deleted.

For the full list of command references, see CDP CLI beta documentation.

Additional VPC scenarios

Learn more about the additional VPC scenarios that show how VPC can be configured between your workload environment and Cloudera Control Plane.

Scenario 1: VPC endpoint in your egress VPC with no HTTP proxy

This scenario has the VPC endpoints in your egress VPC, allowing the VPC endpoints to be shared by multiple VPCs in your network. In this setup, VPC endpoints receive IPs from egress VPC.

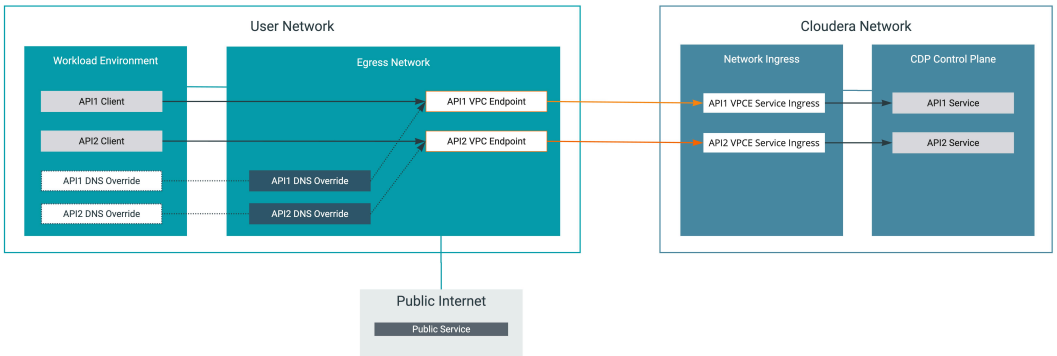

- DNS overrides where DNS is a per-VPC view:
  - Deploying a VPC endpoint with private DNS option enabled will automatically install the DNS overrides in the egress VPC, but this does not solve the overrides for the workload environment VPC.
  - DNS override zones and records can be deployed in each workload environment VPC, or a single set of DNS override zones and records are deployed and the zones are associated with each workload environment VPC.
- DNS overrides where DNS is a regional or global view:
  - The DNS override will have a regional or global impact to resolution of the Cloudera hostnames, clients in the region/globally will receive these VPC endpoints.

Scenario 2: VPC endpoint in your egress VPC, HTTP proxy

Similar to scenario 1, except you have egress traffic flowing through an HTTP proxy in the egress VPC. In this setup, VPC endpoints receive IPs from egress VPC.

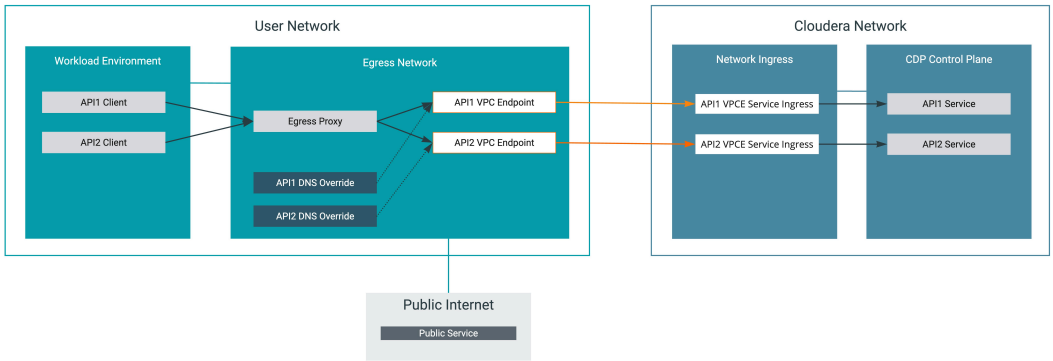

- DNS overrides:
  - Deploying a VPC endpoint with private DNS option enabled will automatically install the DNS overrides in the egress VPC. If the egress proxy is performing the DNS lookup for the destination service, this approach should be sufficient.
  - Transparent proxy or egress firewall policy configurations may require the original destination IP to match the DNS resolution. If this is the case, the override zones/records/ VPC associations can be deployed as described in scenario 1.

Scenario 3: VPC endpoint in your workload environment VPC, HTTP proxy

VPC endpoints deployed in your workload environment network. In this setup, VPC endpoints receive IPs from egress VPC.

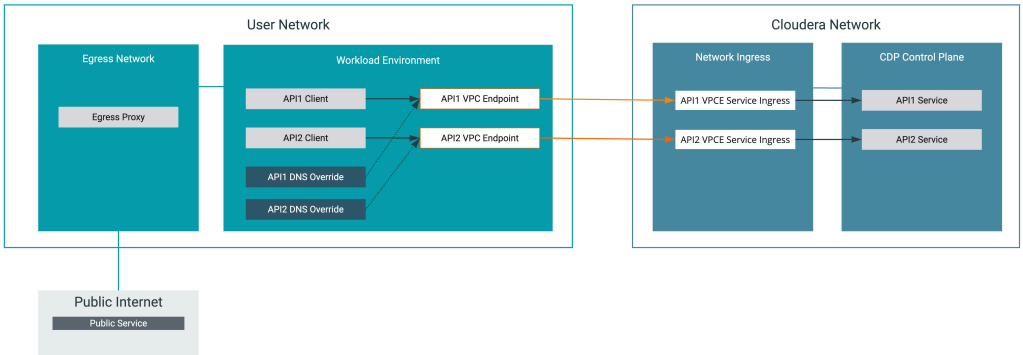

- DNS overrides where DNS is a per-VPC view:
  - Deploying a VPC endpoint with “private DNS” option enabled will automatically install the DNS overrides in the egress VPC. This is recommended.
- DNS overrides where DNS is a regional or global view:
  - The overrides will impact resolution for clients elsewhere in the region and globally.
  - Traffic to these hostnames from outside this VPC will attempt to use these VPC endpoints, which may not be a desired configuration
- HTTP forward proxy or non-transparent proxy
  - Workload environment will be configured to use an HTTP proxy profile.
  - The no_proxy configuration of the profile must include the hostnames of the APIs reachable through VPC endpoint. HTTP requests for destinations in the no_proxy list will not be forwarded to the proxy, local DNS and therefore the VPC endpoints will be used for that traffic

Cloudera Public Cloud

Azure Onboarding Quickstart

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Azure quickstart (Deprecated) **4**
Verify Azure prerequisites 5
Create an Azure AD app 6
Deploy the template 7
Assign roles 8
Create or locate an SSH Key 10
Create a Cloudera credential 10
Register a Cloudera environment 10

Azure quickstart (Deprecated)

If you've reached the Cloudera landing page for the first time, you've come to the right place! In this quickstart, we'll show you step-by-step how to connect Cloudera to your Azure subscription, so that you can begin to provision clusters and workloads.

Warning: This quickstart has been deprecated and is no longer being maintained. For quickly setting up Cloudera on Azure, refer to Deploy Cloudera using Terraform.

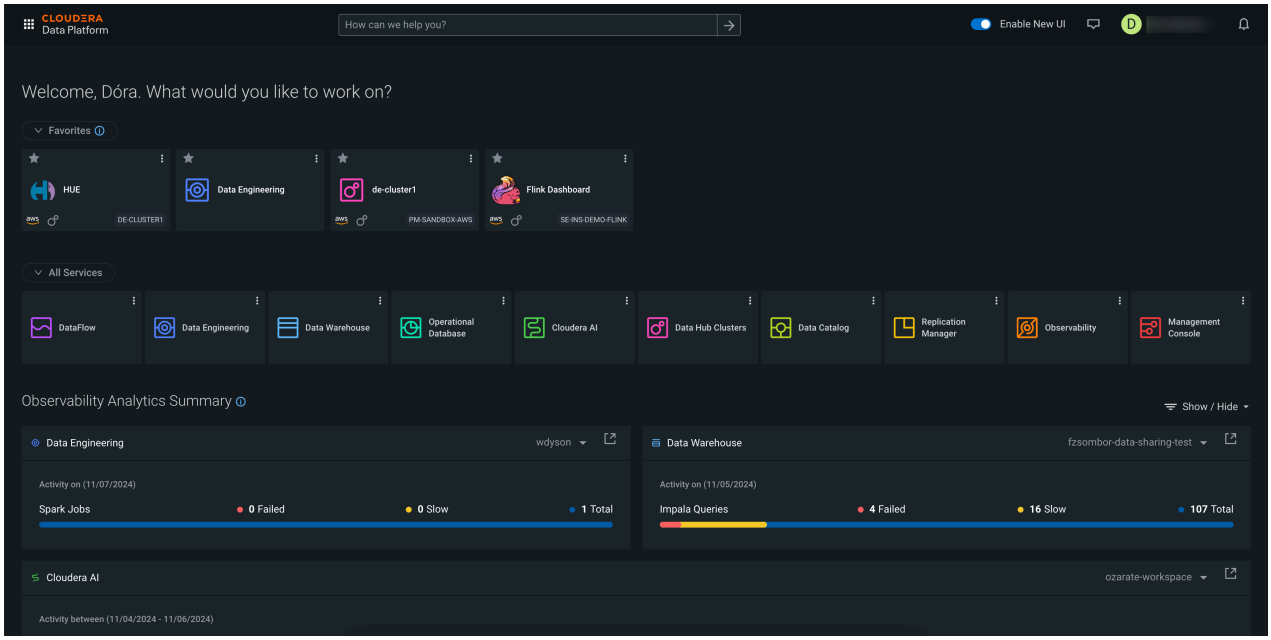

To complete this quickstart, you'll need access to three things:

- The Cloudera console pictured above
- The Azure console
- Azure Cloud shell

Note: This Azure onboarding quickstart is intended for simple Cloudera evaluation deployments only. It may not work for scenarios where Azure resources such as VNet, security group, storage accounts, and so on, are pre-created or Azure accounts have restrictions in place.

In addition to this documentation, you can refer to the following video:

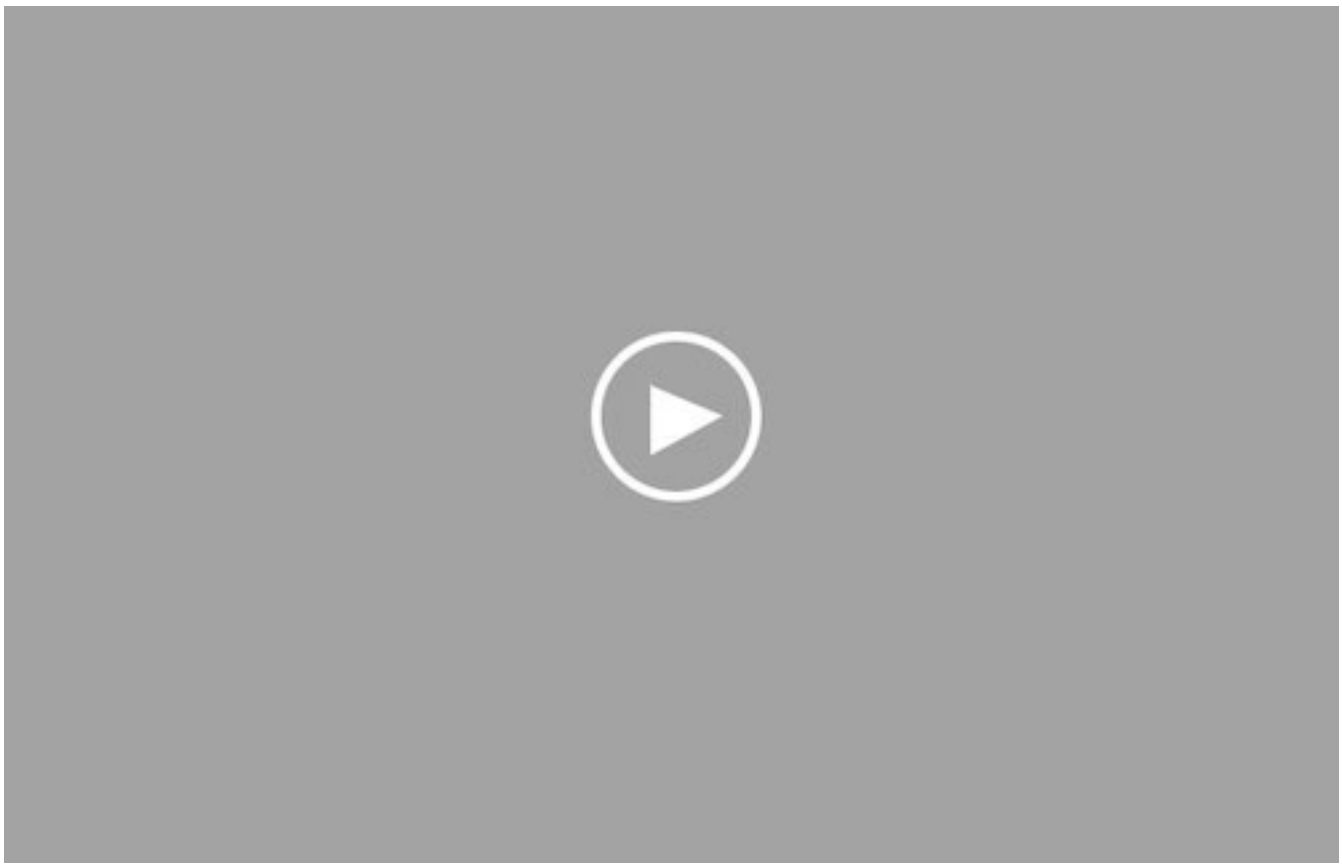

The steps that we will perform are:

Step 0: Verify the Azure prerequisites

Step 1: Create an Azure AD app

Step 2: Deploy the Azure quickstart template

Step 3: Assign roles

Step 4: Create or locate an SSH key

Step 5: Create a Cloudera credential

Step 6: Register a Cloudera environment

Verify Azure cloud platform prerequisites

Before getting started with the Azure onboarding quickstart, review and acknowledge the following:

- This Azure onboarding quickstart is intended for simple Cloudera evaluation deployments only. It may not work for scenarios where Azure resources such as VNet, security group, storage accounts, and so on, are pre-created or Azure accounts have restrictions in place.
- User running the Azure onboarding quickstart should have:
  - Owner permissions on the Azure subscription that you would like to use for Cloudera.
  - Rights to create Azure resources required by Cloudera. See list of Azure resources used by Cloudera.
  - Rights to create an Azure AD application (service principal) and assign Contributor role at subscription level.
  - Cloudera Admin role or Power User role in Cloudera subscription.
- This Azure onboarding quickstart uses an Azure ARM template that automatically creates the required resources such as storage accounts, containers, managed identities, resource groups, and so on.

- Cloudera Public Cloud relies on several Azure services that should be available and enabled in your region of choice. Verify if you have enough quota for each Azure service to set up Cloudera in your Azure account. See list of Azure resources used by CDP.

If you have more complex requirements than those listed here, contact Cloudera Sales Team to help you with Cloudera onboarding.

Create an Azure AD app

In the Azure portal, create an application in your Azure Active Directory tenant. This steps allows you to use the native Cloud Shell terminal and not have to set up Azure CLI.

Procedure

1. Log in to the Azure portal and launch Cloud Shell.

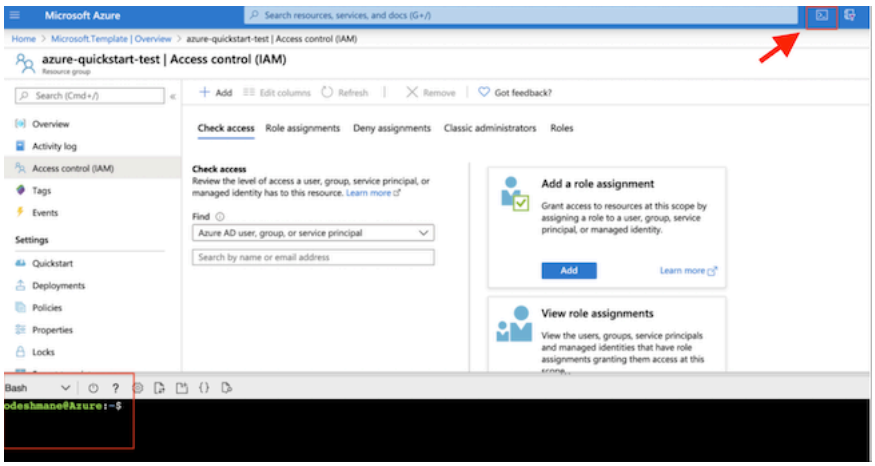

2. When prompted, select Bash to use Bash shell.
3. Run the following command to return the subscription ID and tenant ID:

```
az account list|jq '.[]|{"name": .name, "subscriptionId": .id, "tenantId
": .tenantId, "state": .state}'
```

The output of this command is shown below:

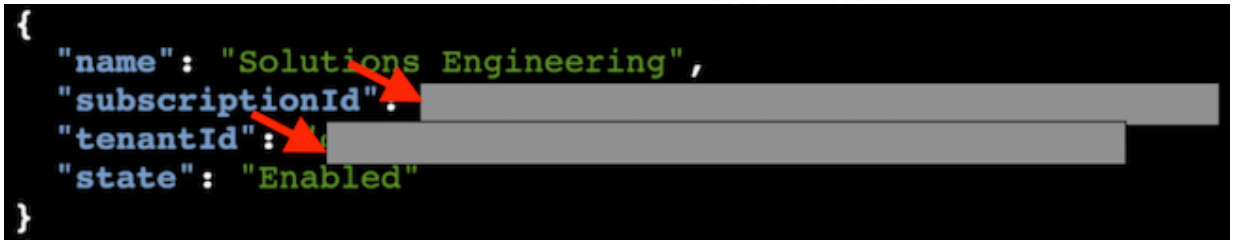

Make a note of the subscriptionId and tenantId values. You will need them later.

Note: In case you have more than one subscription, make sure to only make a note of the subscription that you would like to use for Cloudera.

4. Run the command below to create an app in Azure AD and assign the "Contributor" role at the subscription.

   Note: Replace {subscriptionId} in the command with the subscription ID value from the previous step.

```
az ad sp create-for-rbac --name http://cloudbreak-app --role Contributor
 --scopes /subscriptions/{subscriptionId}
```

   The output of this command is shown below:

Deploy the Azure quickstart template

The Azure quickstart template is a customized ARM template that deploys essential Azure resources for the Cloudera environment.

Procedure

1. Click Deploy to Azure to begin ARM template deployment of Cloudera prerequisites in your Azure subscription.
2. Log in to Azure to create the resources in your subscription that are required for Cloudera deployment. These resources include VNet, ADLS Gen2, and 4 user managed identities.
3. On the **Custom deployment** screen, click Create new under the Resource group field and then give the resource group a name (it should only contain letters, numbers, and hyphens).

4. Under **Settings**, provide an Environment Name in the corresponding field. The Environment Name should be short (between five and twelve characters) and should include only lowercase characters and hyphens (no underscores).

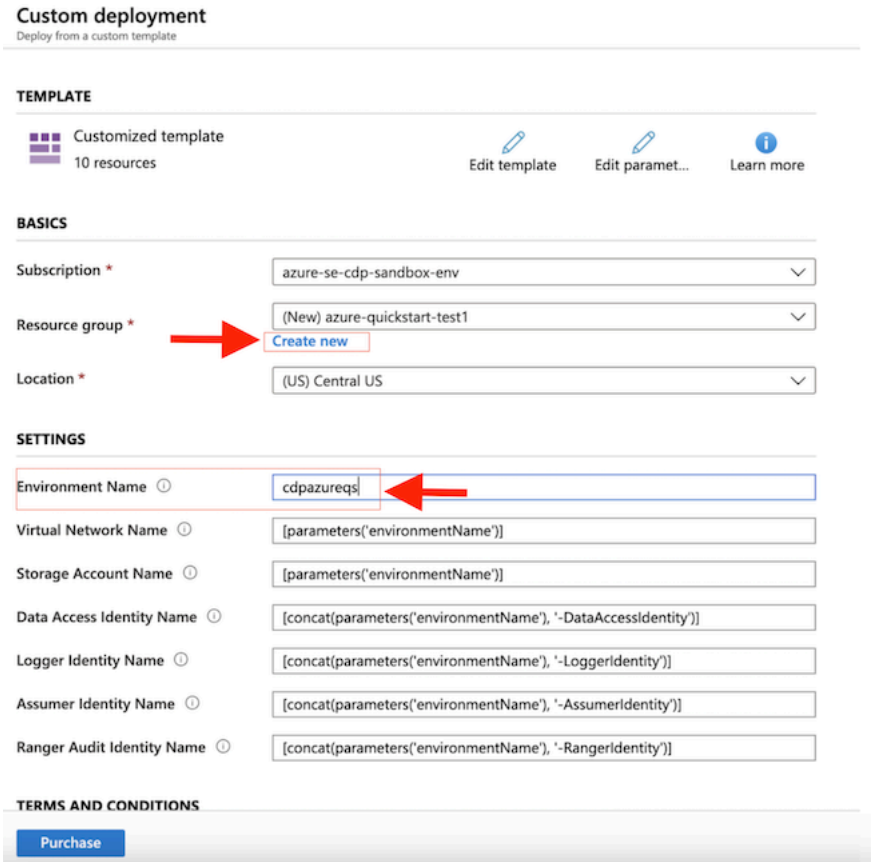

5. Accept the terms and conditions, and click Purchase.

   An ARM script begins to run in the background, creating the resources required for a Cloudera environment. This may take around 10 minutes.
6. When your resource group is up, navigate to the **Overview** page of the resource group.
7. Copy and paste the following values into a note, as you will need them in the next task:
   - Subscription ID: Your subscription ID is found at the top of the resource group **Overview** page.
   - Resource group: The name of the resource group that you created.

Assign roles

Azure Resource Manager templates do not support role assignments at a scope other than resource groups. Perform the following role assignments through UI or CLI.

Before you begin

Make sure that you have your note from the previous step, where you copied values for the Subscription ID and resource group name.

Procedure

1. Once you have values for the subscription ID, resource group name, storage account, environment name, and all four managed identities, click here to download a script.
2. Create a new file in Cloud Shell with the same name, and copy the content of the script there.

3. Replace the following values in the script with the values you have collected thus far:

```
#!/bin/sh

export SUBSCRIPTIONID="<REPLACE WITH YOUR AZURE SUBSCRIPTION ID>"
export RESOURCEGROUPNAME="<REPLACE WITH EXISTING RESOURCE GROUP NAME>"
export STORAGEACCOUNTNAME=$(az storage account list -g $RESOURCEGROUPNAME --subscription $SUBSCRIPTION
export ASSUMER_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subscription $SUBSCRIPTIONID|jq '.[
export DATAACCESS_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subscription $SUBSCRIPTIONID|jq
```

For example, your script should look similar to this:

```
#!/bin/sh

export SUBSCRIPTIONID="jfs85ls8-sik8-8329-fq0m-jqo7v06dk6sy"
export RESOURCEGROUPNAME="myCDPresourcegroup"
export STORAGEACCOUNTNAME=$(az storage account list -g $RESOURCEGROUPNAME
--subscription $SUBSCRIPTIONID|jq '.[]|.name'| tr -d '"')
export ASSUMER_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subscri
ption $SUBSCRIPTIONID|jq '.[]|{"name":.name,"principalId":.principalId}|
select(.name | test("AssumerIdentity"))|.principalId'| tr -d '"')
export DATAACCESS_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subs
cription $SUBSCRIPTIONID|jq '.[]|{"name":.name,"principalId":.principalI
d}|select(.name | test("DataAccessIdentity"))|.principalId'| tr -d '"')
export LOGGER_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subscr
iption $SUBSCRIPTIONID|jq '.[]|{"name":.name,"principalId":.principalId}|
select(.name | test("LoggerIdentity"))|.principalId'| tr -d '"')
export RANGER_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subscrip
tion $SUBSCRIPTIONID|jq '.[]|{"name":.name,"principalId":.principalId}|s
elect(.name | test("RangerIdentity"))|.principalId'| tr -d '"')
# Assign Managed Identity Operator role to the assumerIdentity principal
 at subscription scope
az role assignment create --assignee $ASSUMER_OBJECTID --role 'f1a07417-
d97a-45cb-824c-7a7467783830' --scope "/subscriptions/$SUBSCRIPTIONID"
# Assign Virtual Machine Contributor role to the assumerIdentity principal
 at subscription scope
az role assignment create --assignee $ASSUMER_OBJECTID --role '9980e02c-c
2be-4d73-94e8-173b1dc7cf3c' --scope "/subscriptions/$SUBSCRIPTIONID"
# Assign Storage Blob Data Contributor role to the assumerIdentity prin
cipal at logs filesystem scope
az role assignment create --assignee $ASSUMER_OBJECTID --role 'ba92f5b4-2
d11-453d-a403-e96b0029c9fe' --scope "/subscriptions/$SUBSCRIPTIONID/reso
urceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAccounts/
$STORAGEACCOUNTNAME/blobServices/default/containers/logs"
# Assign Storage Blob Data Contributor role to the loggerIdentity princ
ipal at logs/backup filesystem scope
az role assignment create --assignee $LOGGER_OBJECTID --role 'ba92f5b4-
2d11-453d-a403-e96b0029c9fe' --scope "/subscriptions/$SUBSCRIPTIONID/res
ourceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAccoun
ts/$STORAGEACCOUNTNAME/blobServices/default/containers/logs"
az role assignment create --assignee $LOGGER_OBJECTID --role 'ba92f5b4-
2d11-453d-a403-e96b0029c9fe' --scope "/subscriptions/$SUBSCRIPTIONID/res
ourceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAccoun
ts/$STORAGEACCOUNTNAME/blobServices/default/containers/backups"
# Assign Storage Blob Data Owner role to the dataAccessIdentity principal
 at logs/data/backup filesystem scope
az role assignment create --assignee $DATAACCESS_OBJECTID --role 'b7e6dc6
d-f1e8-4753-8033-0f276bb0955b' --scope "/subscriptions/$SUBSCRIPTIONID/r
esourceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAcco
unts/$STORAGEACCOUNTNAME/blobServices/default/containers/data"
az role assignment create --assignee $DATAACCESS_OBJECTID --role 'b7e6dc6
d-f1e8-4753-8033-0f276bb0955b' --scope "/subscriptions/$SUBSCRIPTIONID/r
esourceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAcco
unts/$STORAGEACCOUNTNAME/blobServices/default/containers/logs"
```

```
az role assignment create --assignee $DATAACCESS_OBJECTID --role 'b7e6dc6
d-f1e8-4753-8033-0f276bb0955b' --scope "/subscriptions/$SUBSCRIPTIONID/r
esourceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAcco
unts/$STORAGEACCOUNTNAME/blobServices/default/containers/backups"
# Assign Storage Blob Data Contributor role to the rangerIdentity principa
l at data/backup filesystem scope
az role assignment create --assignee $RANGER_OBJECTID --role 'ba92f5b4-2d1
1-453d-a403-e96b0029c9fe' --scope "/subscriptions/$SUBSCRIPTIONID/resour
ceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAccounts/
$STORAGEACCOUNTNAME/blobServices/default/containers/data"
az role assignment create --assignee $RANGER_OBJECTID --role 'ba92f5b4-2d1
1-453d-a403-e96b0029c9fe' --scope "/subscriptions/$SUBSCRIPTIONID/resour
ceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAccounts/
$STORAGEACCOUNTNAME/blobServices/default/containers/backups"
```

4. Run the Cloud Shell script: sh azure_msi_role_assign.sh

Create or locate an SSH Key

Cloudera requires that you provide a public SSH key for admin access to VM instances.

You can find more information on SSH key requirement in the topic SSH key. If you need to create one, you can do so by running ssh-keygen -t rsa.

When you complete this step, you have created all of the Azure resources required for this quickstart.

Create a Cloudera credential

In the Cloudera Console, the first step is to create a Cloudera credential. The Cloudera credential is the mechanism that allows Cloudera to create resources inside of your cloud account.

Procedure

1. Log in to the Cloudera web interface.
2. From the Cloudera home screen, click the Cloudera Management Console icon.
3. In the Cloudera Management Console, select Shared Resources > Credentials from the navigation pane.
4. Click in the Create Credential button to create a new credential.
5. Select the Azure tab, name your credential, under "Credential Type", select "App based", and enter the values you previously collected for subscription ID, app ID, and password.

Register a Cloudera environment

When you register an environment, you set properties related to data lake scaling, networking, security, and storage. You will need your Azure environment name, resource group name, storage account name, and virtual network name from your resource group.

Procedure

1. In the Cloudera Management Console, navigate to Environments and click Register Environment.
2. Provide an Environment Name and description. The name can be any valid name.
3. Choose Azure as the cloud provider.

4. Under Microsoft Azure Credential, choose the credential you created in the previous task.

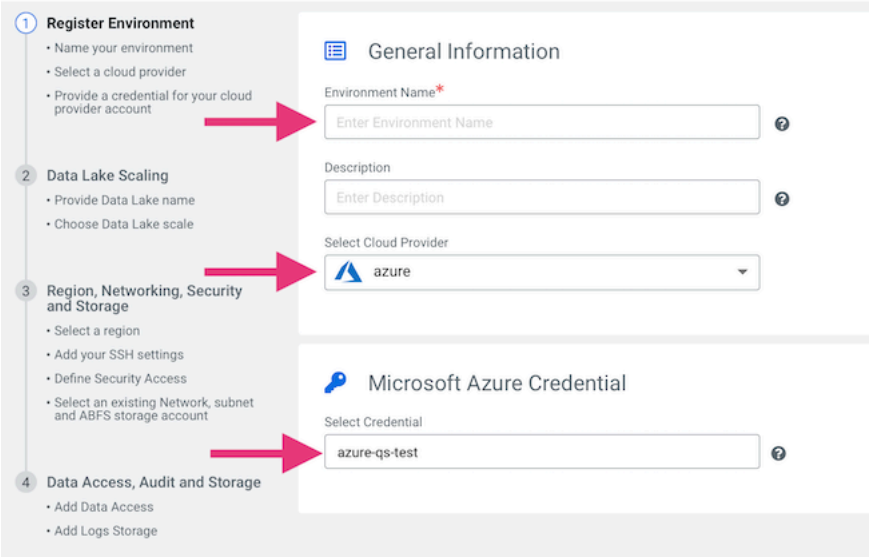

5. Click Next.
6. Under **Data Lake Settings**, give your new data lake a name. The name can be any valid name. Choose the latest data lake version.

7. Under **Data Access and Audit**, choose the following:
   - Assumer Identity: <resourcegroup-name>-<envName>-AssumerIdentity
   - Storage Location Base: data@<storageaccount-name>
   - Data Access Identity: <resourcegroup-name>-<envName>-DataAccessIdentity
   - Ranger Audit Role: <resourcegroup-name>-<envName>-RangerIdentity

   Warning: Ensure that you have entered the correct location base. If the name entered does not match the actual location base created by the quickstart script, environment registration will fail.

   For example:

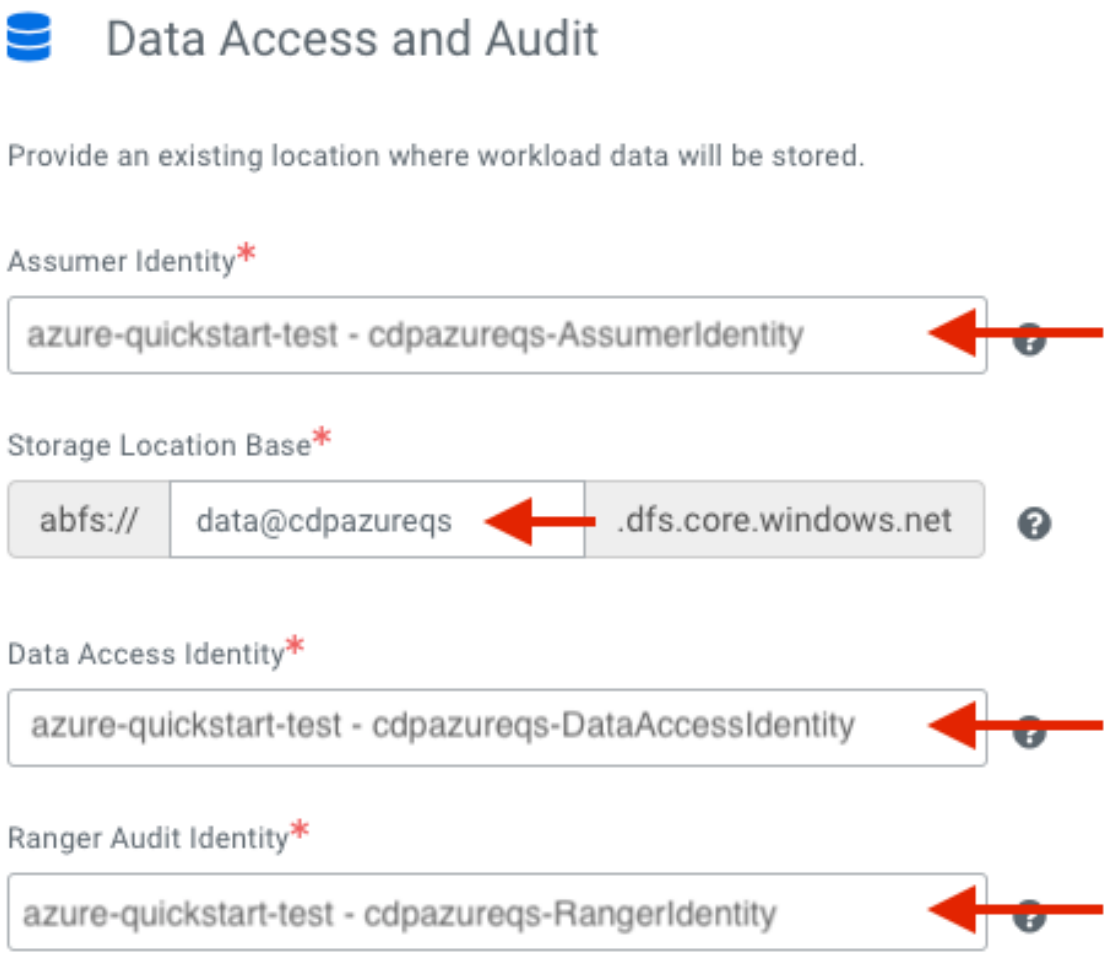

8. For Data Lake **Scale**, choose Light Duty.

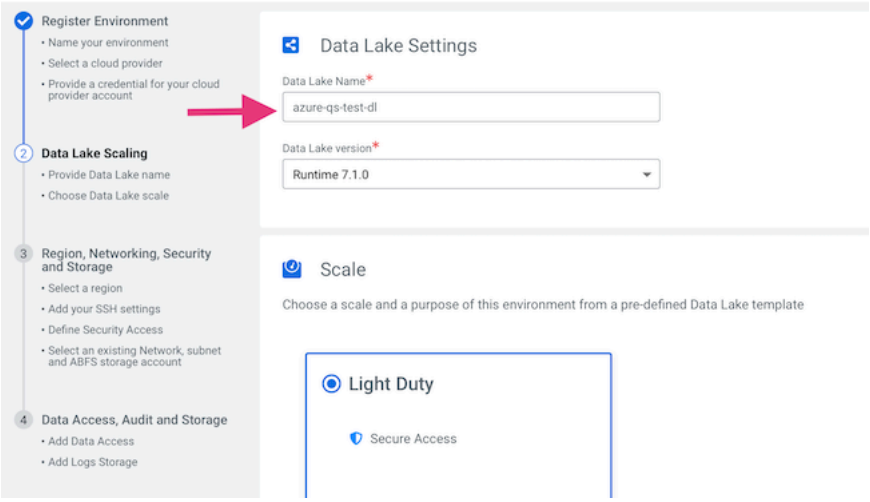

9. Click Next.
10. Under Select Region, choose your desired region. This should be the same region you created an SSH key in previously.
11. Under Select Resource Group, choose your resource group <resourcegroup-name>.
12. For the Select Network field, select the name of the "Virtual Network" resource that was created when you deployed the ARM template to create the resource group. The name of the Virtual Network should be the same as your environment name, but you can verify this in the Azure portal on the Overview page of your resource group. In the following Select Subnets field, ensure all three subnets are selected.
13. Slide the Enable Public Endpoint Access Gateway toggle to the Enabled position. Select any of your three subnets.
14. The Create Public IPs toggle can remain in the default Enabled position.
15. In the next drop-down input box, keep Flexible Server selected.
16. Under the Encryption section, leave both the Enable encryption at host and Enable Customer-Managed Keys toggles in their default deselected state.
17. Under the Proxies section, leave the Select Proxy Configuration field in its default "Do not use Proxy Configuration" state.

18. Under **Security Access Settings**, select Create New Security Groups for the Security Access Type.

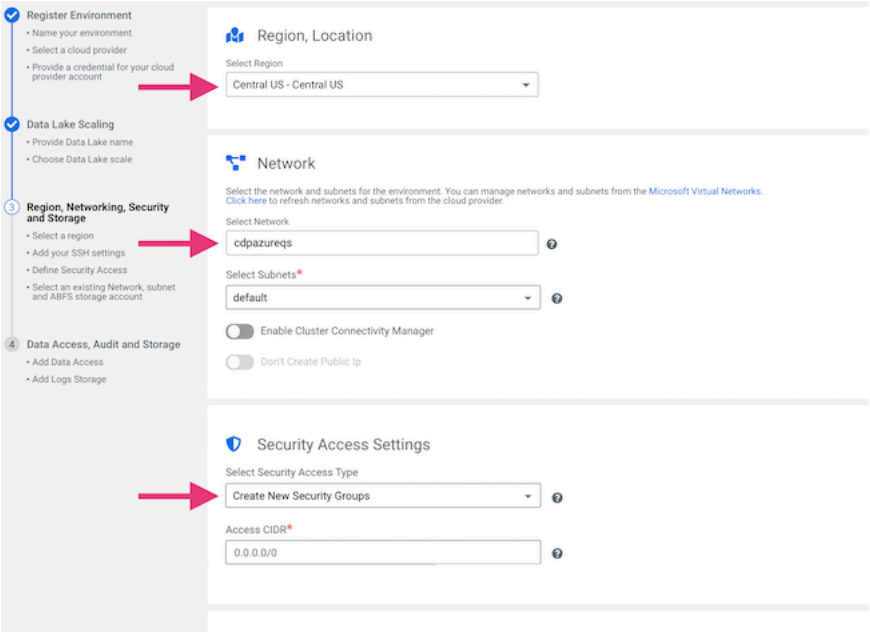

19. Under **SSH Settings**, paste the public SSH key that you created earlier.
20. Optionally, under **Add Tags**, provide any tags that you'd like the resources to be tagged with in your Azure account.
21. Click Next.

22. Under **Logs**, choose the following:

- Logger Identity: <resourcegroup-name>-<envName>-LoggerIdentity
- Logs Location Base: logs@<storageaccount-name>
- Backup Location Base: backups@<storageaccount-name>

Warning: Ensure that you have entered the correct location base. If the name entered does not match the actual location base created by the quickstart script, environment registration will fail.

For example:

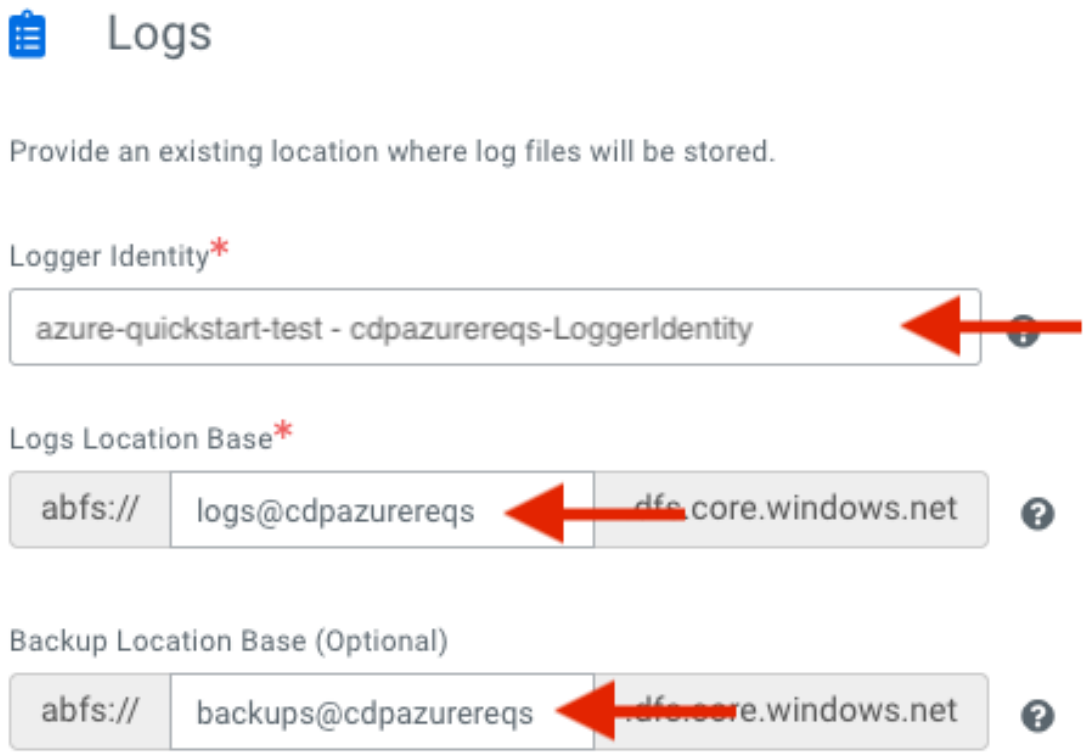

23. Under the Telemetry section, leave the inputs as default (Observability Enabled, Deployment Cluster Logs Collection Disabled).

24. Before finishing, click the Show CLI Command button and then Copy the full command for future reference. This can be useful for e.g. analyzing any errors in the deployment process.

25. Click Register Environment.

Cloudera Public Cloud

Azure Reference Network Architecture

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Public Cloud reference network architecture for Azure..... 4

Taxonomy of network architectures.....5

Cloudera Management Console to customer cloud network.....6
Customer on-prem network to cloud network.....8

Network architecture.....9

Architecture diagrams.....9
Component description.....12
VNet.....12
Subnets.....12
Gateways and route tables.....12
Private endpoints.....13
Security groups.....14
Security groups for Data Lakes.....14
Additional rules for AKS-based workloads.....14
Outbound connectivity requirements.....15
Domain names for the endpoints.....15
DNS.....17

Cloudera Public Cloud reference network architecture for Azure

This topic includes a conceptual overview of the Cloudera Public Cloud network architecture for Azure, its use cases, and personas who should be using it.

Overview

Cloudera Public Cloud allows customers to set up cloud Data Lakes and compute workloads in their cloud accounts on AWS, Azure, and Google Cloud. It maps a cloud account to a concept called the environment into which all CDP workloads, including Cloudera Data Hub clusters (compute workload clusters)and data services (such as Cloudera Data Engineering, Cloudera Data Warehouse, Cloudera AI, Cloudera Operational Database, Cloudera DataFlow) are launched. For these Data Lakes, Cloudera Data Hub clusters, and data services to function correctly, several elements of the cloud architecture need to be configured appropriately: access permissions, networking setup, cloud storage and so on. Broadly, these elements can be configured in one of two ways:

- Cloudera can set up these elements for the customer

  Usually, this model helps to set up a working environment quickly and try out Cloudera. However, many enterprise customers prefer or even mandate specific configurations of a cloud environment for Infosec or compliance reasons. Setting up elements such as networking and cloud storage requires prior approvals and they would generally not prefer, or even actively prevent, a third party vendor like Cloudera to set up these elements automatically.
- Cloudera can work with pre-created elements provided by the customer

  In this model, the flow for creating the cloud Data Lakes accepts pre-created configurations of the cloud environment and launches workloads within those boundaries. This model is clearly more aligned with enterprise requirements. However, it brings with it the risk that the configuration might not necessarily play well with Cloudera requirements. As a result, customers might face issues launching Cloudera workloads and the turnaround time to get to a working environment might be much longer and involve many tedious interactions between Cloudera and the customer cloud teams.

From our experience in working with several enterprise customers, the most complicated element of the cloud environment setup is the cloud network configuration. The purpose of this document is to clearly articulate the networking requirements needed for setting up a functional Cloudera Public Cloud environment into which the Data Lakes and compute workloads of different types can be launched. It attempts to establish the different points of access to these workloads and establishes how the given architecture helps to accomplish this access.

Along with this document, you can use the Terraform Cloudera deployment to automatically set up a model of this reference architecture, which can then be reviewed for security and compliance purposes. For more information, see *Deploy Cloudera using Terraform*.

Note:

Currently this document only covers network architecture required for registering a Cloudera environment (with a Data Lake and FreeIPA) and deploying Cloudera Data Warehouse, and Cloudera AI in the environment. It does not currently cover Cloudera Data Hub clusters and the remaining data services (Cloudera Data Engineering, Cloudera DataFlow, and Cloudera Operational Database).

Use cases

Cloudera Public Cloud allows customers to process data in the cloud storage under a secure and governed Data Lake using different types of compute workloads that are provisioned via Cloudera Data Hub or data services. Typically the lifecycle of these workloads is as follows:

- A Cloudera environment is set up by a Cloudera admin using their cloud account. This sets up a cloud Data Lake cluster and FreeIPA cluster with security and governance services and an identity provider for this environment.

The Cloudera admin may need to work with a cloud administrator to create all the cloud provider resources (including networking resources) that are required by Cloudera.

- Then one or more Cloudera Data Hub clusters and data services can be launched, linked to the Data Lake. Each of these Cloudera Data Hub clusters and data services typically serves a specific purpose such as data ingestion, analytics, machine learning and so on.
- These Cloudera Data Hub clusters and data services are accessed by data consumers such as data engineers, analysts or scientists. This is the core purpose of using Cloudera on the public cloud.
- These compute workload clusters and data services can be long-running or ephemeral, depending on the customer needs.

There are two types of Cloudera users who interact with the product for different purposes:

- Cloudera admins - These persons are usually concerned with the launch and maintenance of the cloud environment, and the Data Lake, FreeIPA, Cloudera Data Hub clusters, and data services running inside the environment. They use a Cloudera Management Console running in the Cloudera AWS account to perform these operations of managing the environment.
- Data consumers - These are the data scientists, data analysts, and data engineers who use the Cloudera Data Hub clusters and data services to process data. They mostly interact directly with the compute workloads (Cloudera Data Hub clusters and data services) running in their cloud account. They could access these either from their corporate networks (typically through a VPN) or other cloud networks their corporate owns.

These two types of users and their interaction with Cloudera are represented in the following diagram:

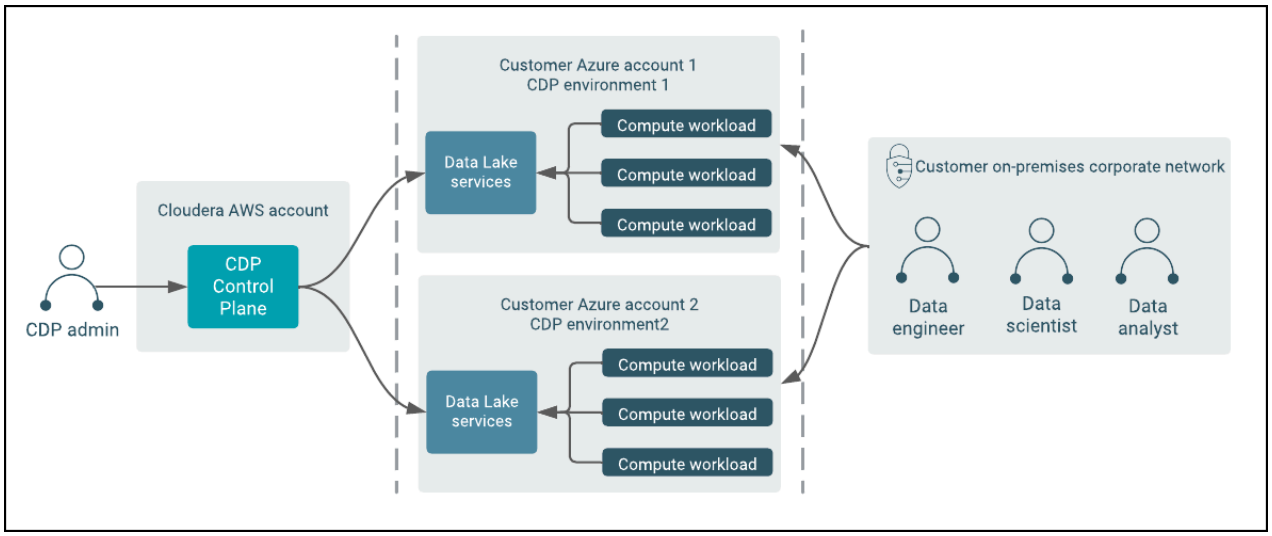

Related Information

Deploy Cloudera using Terraform

Taxonomy of network architectures

This topic provides a high-level overview of each type of network architecture that Cloudera supports.

At a high level, there are several types of network architectures that Cloudera supports. As can be expected, each type brings a unique trade-off among various aspects, such as ease of setup, security provided, workloads supported, and so on. This section only provides a high level overview of each type. The characteristics of each type are explained under appropriate sections in the rest of the document. The users must review the advantages and disadvantages of each of these taxonomies in detail before making a choice suitable to their needs.

Name	Description	Trade-offs
Publicly accessible networks	Deploys customer workloads to hosts with public IP addresses. Security groups must be used to restrict access only to corporate networks as needed.	Easy to set up for POCs. Low security levels.
Semi-private networks	Deploys customer workloads to private subnets, but exposes services to which data consumers need access over a load balancer with a public IP address. Security groups or allow-lists (of IP addresses or ranges) on load balancers must be used to restrict access to these public services only to corporate networks as needed.	This option is fairly easy to set up too, but it may not solve all the use cases of access (in semi private networks). The surface of exposure is reduced, and it is reasonably secure.
Fully private networks	Deploys customer workloads to private subnets, and the services to which data consumers need access are only on private IPs. Requires connectivity to corporate networks to be provided using solutions like VPN gateways, and so on.	Complex to set up depending on prior experience of establishing such connectivity, primarily due to the way the customer has to solve the corporate network peering problem. But it is very secure.
Fully private outbound restricted networks	This is the same as fully private networks; Except, in addition, Cloudera provides a mechanism for users to configure an outbound proxy or firewall to monitor or restrict the communication outside their networks.	Most complex to set up, mainly considering the varied needs that data consumers would have to connect outside the VNet on an evolving basis. It is also the most secure for an enterprise.

Cloudera Management Console to customer cloud network

This topic explains the possible ways in which the Cloudera Control Plane can communicate with the compute infrastructure in the customer network, in the context of the Cloudera Management Console.

As described previously, the Cloudera admin would typically use the Cloudera Management Console that runs in the Cloudera Control Plane to launch Cloudera environments with Data Lakes, FreeIPA, Cloudera Data Hub clusters, and data services into their cloud accounts. In order to accomplish this, the Cloudera Control Plane and the compute infrastructure in the customer network (such as VMs, AKS clusters) should be able to communicate with each other. Depending on the chosen network architecture, this communication can occur in the ways described below.

Publicly accessible networks

In this model of publicly accessible networks, the compute infrastructure must be reachable over the public internet from the Cloudera Management Console. While this is fairly easy to set up, it is usually not preferred by enterprise customers, as it implies that the VM nodes or AKS nodes are assigned public IP addresses. While the access control rules for these nodes can still be restricted to the IP addresses of the Cloudera Management Console components, it is still considered insecure for each of the network architectures described earlier.

Semi-private networks

Publicly accessible networks are easy to set up for connectivity, both from the Cloudera Control Plane and the customer on-prem network, but have a large surface area of exposure as all compute infrastructure has public IP addresses. In contrast, fully private networks need special configuration to enable connectivity from the customer on-prem network, due to having no surface area of exposure to any of the compute infrastructure. While very secure, it is more complex to establish.

There is a third configuration supported by Cloudera, semi-private networks, that provides some trade-offs between these two options. In this configuration, the user deploys the worker nodes of the compute infrastructure on fully private networks as described above. However, the user chooses to expose UIs or APIs of the services fronting these worker nodes over a public network load balancer. By using this capability, the data consumers can access the UIs or APIs of the compute infrastructure through these load balancers. It is also possible to restrict the IP ranges from which such access is allowed using security groups.

While this option provides a trade-off between ease of setup and exposure levels, it may not satisfy all use cases related to communication between various endpoints. For example, some compute workloads involving Kafka or NiFi would not benefit from having a simple publicly exposed load balancer. It is recommended that customers evaluate their use cases against the trade-off and choose an appropriately convenient and secure model of setup.

Fully private networks

In this model of fully private networks, the compute infrastructure is not assigned any public IP addresses. In this case, communication between the Cloudera Control Plane and compute infrastructure is established using a “tunnel” that originates from the customer network to the Cloudera Control Plane. All communication from the Cloudera Control Plane to the compute nodes is then passed through this tunnel. From experience, Cloudera has determined that this is the preferred model of communication for customers.

To elaborate on the tunneling approach, Cloudera uses a solution called “Cluster Connectivity Manager”. At a high level, the solution uses two components, an agent (Cluster Connectivity Manager agent) that runs on a VM provisioned in the customer network and a service (CCM service) that runs on the Cloudera Control Plane. The Cluster Connectivity Manager agent, at start-up time, establishes a connection with the Cluster Connectivity Manager service. This connection forms the tunnel. This tunnel is secured by asymmetric encryption. The private key is shared with the agent over cloud specific initialization mechanisms, such as a user-data script in Azure.

When any service on the Cloudera Control Plane wants to send a request to a service deployed on the customer environment (depicted in the below diagram as the “logical flow”), it physically sends a request to the Cluster Connectivity Manager service running in the Cloudera Control Plane. The Cluster Connectivity Manager agent and Cluster Connectivity Manager service collaborate over the established tunnel to accept the request, forward it to the appropriate service, and send a response over the tunnel to be handed over the calling service on the Cloudera Control Plane.

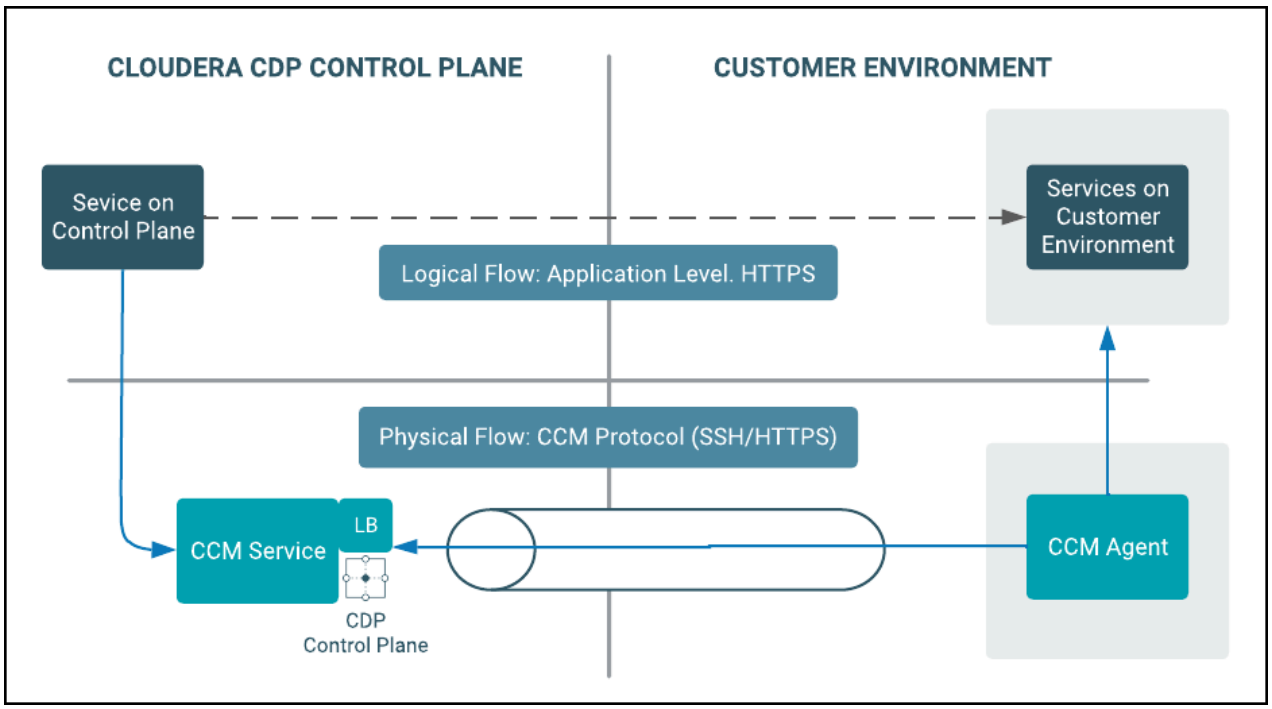

Currently, all AKS clusters provisioned by various Cloudera data services are enabled with public and private cluster endpoints. The AKS public endpoint is needed to facilitate the interactions between Cloudera Control Plane and the AKS cluster while worker nodes and Kubernetes control plane interact over private API endpoints. Cloudera Data Warehouse supports private AKS endpoints today (see “Enabling a private Cloudera Data Warehouse environment in Azure Kubernetes Service”). There are plans to support private AKS endpoints for other data services in the future. When this occurs, the documentation will be updated to reflect the same.

Fully private outbound restricted networks

Fully private outbound restricted networks is a variant of the fully private network where customers would like to pass outbound traffic originating from their cloud account through a proxy or firewall and explicitly allow-list URLs that are allowed to pass through. Cloudera Public Cloud supports such configuration. If such network architecture is chosen, the customer must ensure the following:

- Users configure a proxy for the environment via Cloudera, as documented in “Using a non-transparent proxy”.
- Compute resources (such as VMs used by Cloudera Data Hub clusters and data services) can connect to the proxy or firewall via appropriate routing rules.
- The proxy or firewall is set up to allow connections to all hosts, IP ranges, ports, and protocol types that are documented in “Azure outbound network access destinations”.

Note:

Given that fully private networks is the recommended option of connectivity in most cases, this document describes the architecture assuming a fully private network setup.

Related Information

Cluster Connectivity Manager

Enabling a private Cloudera Data Warehouse environment in Azure Kubernetes Service

Using a non-transparent proxy

Azure outbound network access destinations

Customer on-prem network to cloud network

After Cloudera Data Hub clusters and data services are launched in the customer’s cloud network, data consumers such as data engineers, data scientists, and data analysts access services running in these Cloudera data services. Sometimes, Cloudera admins who set up and operate these clusters might need this access to diagnose any issues the clusters face.

Examples of these include:

- Web UIs such as:
  - Hue: For running SQL queries in Hive tables
  - Cloudera AI Workbenches: For accessing Cloudera AI projects, models, notebooks, and so on
  - Cloudera Manager: For Cloudera Data Hub clusters and Data Lakes
  - Atlas and Ranger: For metadata, governance, and security in the Data Lake
- JDBC endpoints: Customers can connect tools such as Tableau using a JDBC URL pointing to the Hive server.
- SSH access: Data engineers might log in to nodes on the compute workload clusters and data services to run data processing jobs using YARN, Spark, or other data pipeline tools.
- Kube API access: Cloudera data services that run on AKS (such as Cloudera Data Warehouse and Cloudera AI) also provide admin access to Kubernetes for purposes of diagnosing issues.
- API access: Customers can use APIs for accessing many of the services exposed via the web UIs for purposes of automation and integration with other tools, applications, or other workloads they have. For example, Cloudera AI exposes the Cloudera AI API v2 to work with Cloudera AI projects and other entities. See Cloudera AI API v2.

These services are accessed by these consumers from within a corporate network inside a VPN. These services typically have endpoints that have a DNS name, the format of which is described more completely in the DNS section of this reference architecture documentation. These DNS names resolve to IP addresses assigned to the nodes, or load balancers fronting the ingest controllers of Kubernetes clusters. Note that these IP addresses are usually private IPs; Therefore, in order to be able to connect to these IPs from the on-premise network within a VPN, some special connectivity setup would be needed, typically accomplished using technologies like VPN peering, DirectConnect, transit gateways, and so on. While there are many options possible here, this document describes one concrete option of achieving this connectivity.

Related Information

Cloudera AI API v2

Network architecture

Cloudera recommends that customers configure their cloud networks as fully private networks, as described in this chapter. This will help on-boarding Data Lakes, Cloudera Data Hub cluster, and data services smoothly.

Note:

This network architecture only covers the fully private networks and assumes unrestricted outbound access.

The cloudera-deploy tool, which is released along with this document can be used to automatically set up a model of this reference architecture, which can then be reviewed for security and compliance purposes.

Architecture diagrams

This topic includes diagrams illustrating the various elements of the network architecture in the customer's cloud account into which Cloudera environments with Data Lakes, Cloudera Data Hub clusters, and data services will be launched.

Cloudera recommends that customers configure their cloud networks as described in this chapter and illustrated in the following diagrams. This will help onboarding Data Lakes, Cloudera Data Hub cluster, and data services smoothly. The following diagram illustrates the configuration for a fully private network that can be configured by the customer. This configuration can be provided by the Cloudera admins when they are setting up Cloudera environments and workloads which will get launched into this configuration.

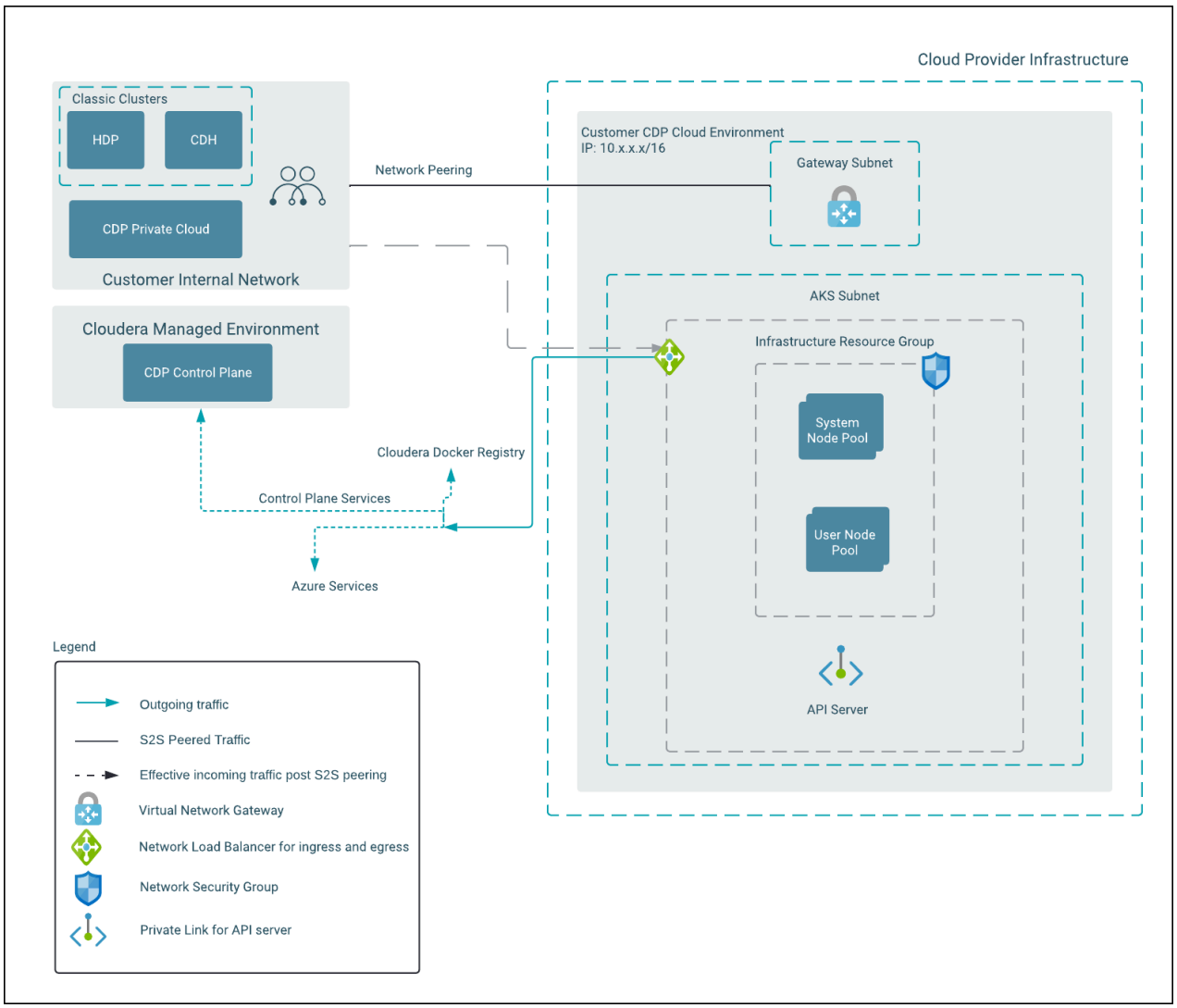

Note the following points about this architecture:

- The configuration is a fully private network configuration - that is, the workloads are launched on nodes that do not have public IP addresses.
- Workloads connect to the Cloudera Control Plane over a fixed IP and port range.
- For users to be able to connect from the customer on-prem network to the Cloudera workloads in the private subnet, some network connectivity setup is required. In this case, a customer's network peered to Azure VNet via Azure VPN gateway is shown.

Some of the Cloudera data services are based on Azure AKS clusters. Azure’s AKS manages the Kubernetes control plane nodes while the worker nodes that make up the Cloudera workload cluster get provisioned in the customer’s VPC. The AKS control plane has an API endpoint for administrative purposes which is commonly referred to as API server address. The data service itself is accessible through a service endpoint ELB.

This is illustrated in the following diagram:

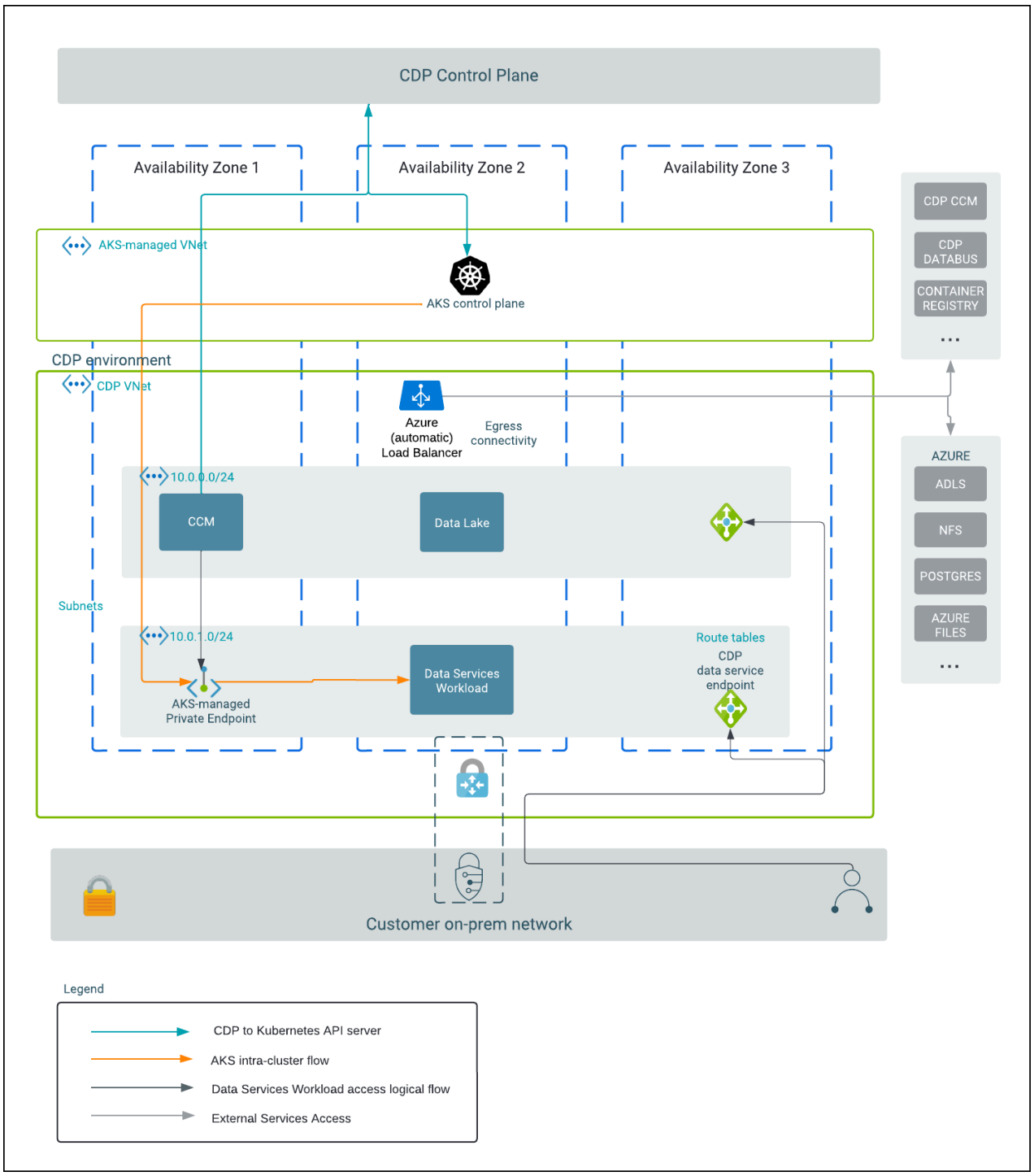

As can be seen in the above diagram, Cloudera workloads have dependencies on some Azure cloud services such as ADLS, Azure Postgres and so on. A full list of these services, described in the context of each workload is specified in “Azure outbound network access destinations”.

In this example of a fully private deployment with an existing network of your own configuration, you must configure your own egress connectivity through a NAT gateway setup or user-defined routing. For more information see VNet and subnets and Azure Load Balancers in Data Lakes and Cloudera Data Hub clusters.

In the chapters that follow, we detail the elements of this architecture, including specifying the configuration and options in each of the components.

Related Information

Azure outbound network access destinations

Component description

This section includes an overview of the VNet, subnets, gateways and route tables, and security groups required for Cloudera Public Cloud for Azure.

VNet

An Azure Virtual Network (VNet) is needed for deploying Cloudera workloads into the customer's cloud account. VNet is similar to a traditional network that you would operate in your own data center, but brings with it additional benefits of Azure's infrastructure such as scale, availability, and isolation.

Cloudera recommends that the VNet used for Cloudera is configured with the properties specified below:

- The CIDR block for the VNet should be sufficiently large for supporting all the Cloudera Data Hub clusters and data services that you intend to run. Refer to "VNet and subnet planning" to understand how to compute the CIDR block range.
- In addition, you may want to ensure that the CIDR ranges assigned to the VNet do not overlap with any of your on-premise network CIDR ranges, as this may be a requirement for setting up connectivity from your on-premise network to the subnets.

Related Information

VNet and subnet planning

Subnets

A subnet is a partition of the virtual network in which Cloudera workloads are launched.

It is recommended that the subnets be configured with the following properties:

- The CIDR block for the subnet should be sufficiently large for supporting the Cloudera workload targeted to run inside it. Refer to "VNet and subnet planning" to understand how to compute the CIDR block range.
- Several Cloudera data services run on Kubernetes and use Kubenet CNI plugin for networking. Azure's Kubenet CNI plugin requires that the subnet is not shared between multiple Kubernetes clusters as it adds custom routes (see "Bring your own subnet and route table with kubenet"). Therefore, as many subnets as the expected number of workloads need to be created.
- In addition, you may want to ensure that the CIDR ranges assigned to the subnets do not overlap with any of your on-premise network CIDR ranges, as this may be a requirement for setting up connectivity from your on-prem network to the subnets.
- A subnet can be associated with a Network Security Group (NSG). However, since Cloudera works with a fully private network configuration where communication is always initiated from VMs within the subnets, an NSG at subnet level is generally not useful for this configuration.

Related Information

VNet and subnet planning

Bring your own subnet and route table with kubenet

Gateways and route tables

This topic covers recommended gateway and route table configurations for Cloudera Public Cloud for Azure.

Connectivity from Cloudera Control Plane to Cloudera workloads

- As described in the "Subnets" section above, each Cloudera data service workload requires its own subnet and a non-shared route table associated with it (see "Bring your own subnet and route table with kubenet").

- As described in the “Taxonomy of network architectures”, nodes in the Cloudera workloads need to connect to the Cloudera Control Plane over the internet to establish a “tunnel” over which the Cloudera Control Plane can send instructions to the workloads.
- Private AKS cluster is a feature that lets Cloudera access a Kubernetes workload cluster over a private IP address (see “Enabling a private Cloudera Data Warehouse environment in Azure Kubernetes Service”). When it is enabled for a Cloudera Data Warehouse workload, Cloudera Data Warehouse requires the user to have already set up internet connectivity for the subnet (see “Outbound type of userDefinedRouting)”.
- Cloudera data services such as Datalake and Cloudera AI create a public load balancer for internet connectivity.
- If a firewall is configured, the destinations described in “Azure outbound network access destinations” need to be allowed for the Cloudera workloads to work.

Connectivity from customer on-prem to Cloudera workloads

- As described in the “Use cases” section (see “Cloudera Public Cloud reference network architecture for Azure”>“Use Cases”), data consumers need to access data processing or consumption services in the Cloudera workloads.
- Given that these are created with private IP addresses in private subnets, the customers need to arrange for access to these addresses from their on-prem or corporate networks in specific ways.
- There are several possible solutions for achieving this, but one that is depicted in the architectural diagram, uses “Azure VPN Gateway”.
- Each workload provides an option to connect to the workload endpoint over a public or a private IP. It's recommended that the public IPs are disabled and that users rely on the corporate network connectivity for accessing the workloads with private IPs.

Related Information

Subnets

Bring your own subnet and route table with kubenet

Taxonomy of network architectures

Enabling a private Cloudera Data Warehouse environment in Azure Kubernetes Service

Outbound type of userDefinedRouting

Azure outbound network access destinations

Cloudera Public Cloud reference network architecture for Azure

Azure VPN Gateway

Private endpoints

Cloudera workloads can be configured to access the Azure resources over a private IP (which is called a private endpoint) or over a public IP (which is called a service endpoint).

The private endpoint setup requires a private DNS zone which the VNet is linked to and resolves the Azure resources to private IP addresses. Cloudera supports a private endpoint configuration for Azure postgres only. Cloudera admin can choose to either create the private DNS zone and link it to the VNet as described in “Bringing your own private DNS”, or let the Cloudera create them when provided with necessary configuration described in “Using Cloudera-managed private DNS”.

Cloudera supports only service endpoint configuration for other Azure resources(such as Microsoft Storage). The subnets need to be enabled to support the service endpoints. See “Service endpoint for Azure Postgres” for detailed steps.

Related Information

Bringing your own private DNS

Using Cloudera-managed private DNS

Service endpoint for Azure Postgres

Security groups

During the specification of a VNet to Cloudera, the Cloudera admin specifies security groups that will be associated with all the Cloudera workloads launched within that VNet. These security groups will be used in allowing the incoming traffic to the hosts.

Security groups for Data Lakes

During the specification of a VNet to Cloudera, the Cloudera admin can either let Cloudera create the required security groups, taking a list of IP address CIDRs as input; or create them in Azure and then provide them to Cloudera.

When getting started with Cloudera, the Cloudera admin can let Cloudera create security groups, taking a list of IP address CIDRs as input. These will be used in allowing the incoming traffic to the hosts. The list of CIDR ranges should correspond to the address ranges from which the Cloudera workloads will be accessed. In a VPN-peered VNet, this would also include address ranges from customer's on-prem network. This model is useful for initial testing given the ease of set up.

Alternatively, the Cloudera admin can create security groups on their own and select them during the setup of the VNet and other network configuration. This model is better for production workloads, as it allows for greater control in the hands of the Cloudera admin. However, note that the Cloudera admin must ensure that the rules meet the requirements described below.

For a fully private network, network security groups should be configured according to the types of access requirements needed by the different services in the workloads. The "Network security groups" section includes all the details of the necessary rules that need to be configured.

Note that for a fully private network, even specifying an open access here (such as 0.0.0.0/0) is restrictive because these services are deployed in a private subnet without a public IP address and hence do not have an incoming route from the Internet. However, the list of CIDR ranges may be useful to restrict which private subnets of the customer's on-prem network can access the services.

Rules for AKS based workloads are described separately in the following section.

Related Information

Network security groups

Additional rules for AKS-based workloads

At the time of enabling a Cloudera data service, the Cloudera admin can specify a list of CIDR ranges that will be used in allowing the incoming traffic to the workload Load Balancer.

This list of CIDR ranges should correspond to the address ranges from which the Cloudera data service workloads will be accessed. In a VPN peered VNet, this would also include address ranges from customer's on-prem network. In a fully private network setup, 0.0.0.0/0 implies access only within the VNet and the peered VPN network which is still restrictive.

Cloudera Data Warehouse data service currently supports fully private AKS clusters, which means the Kubernetes API server will have a private IP address. Cloudera AI uses a public endpoint by default for all AKS cluster control planes. It is highly recommended to provide a list of outbound public CIDR ranges at the time of provisioning a data service to restrict access to the AKS clusters. In addition, the data service adds Cloudera public CIDR range to allow access between workloads and Cloudera Control Plane. An example configuration section for a data service looks like below:

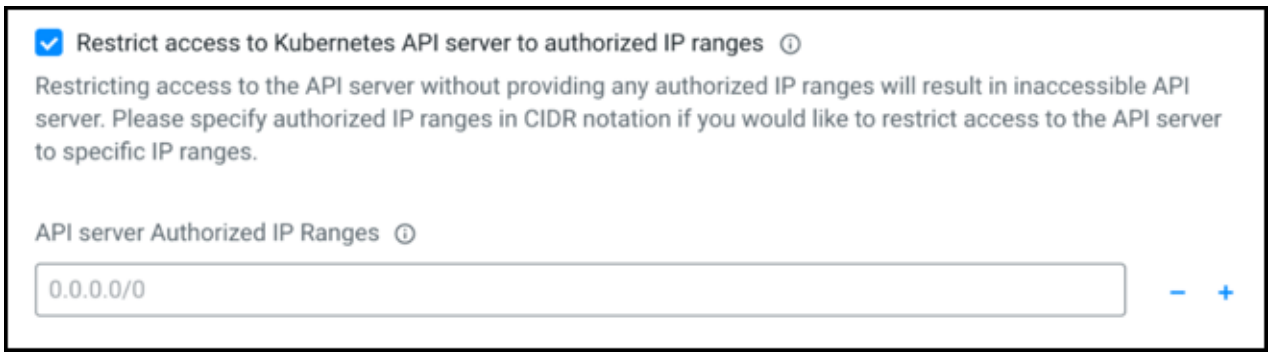

Specific guidelines for restricting access to Kubernetes API server and workloads are detailed in “Restricting access for Cloudera services that create their own security groups on Azure” by each data service.

Within the AKS cluster, security groups are defined to facilitate AKS control plane-pod communication, inter-pod and inter-worker node communication as well as workload communication through Load Balancers.

Related Information

Restricting access for Cloudera services that create their own security groups on Azure

Outbound connectivity requirements

Outbound traffic from the worker nodes is unrestricted and is targeted at other Azure services and Cloudera services. The comprehensive list of services that get accessed from a Cloudera environment can be found in “Azure outbound network access destinations”.

Related Information

Azure outbound network access destinations

Domain names for the endpoints

The previous sections dealt with how connectivity is established to the workload infrastructure. This section deals with “addressability”.

The workloads launched by Cloudera contain a few services that need to be accessed by the Cloudera admins and data consumers. These include services such as Cloudera Manager; metadata services such as the Hive Metastore, Atlas, or Ranger; and data processing or consumption services such as Oozie server, Hue, and so on. Given the nature of the cloud infrastructure, the IP addresses for the nodes running these services may change (for example if the infrastructure is restarted or repaired). However, these should have statically addressable DNS names so that users can access them with the same names.

In order to help with this, Cloudera assigns DNS names to these nodes. These naming schemes have the following properties:

- The DNS name is of the following format for each Data Lake node and the Data Lake cluster endpoint:

  ```
  <CLUSTER_NAME>-{<HOST_GROUP><i>}.<ENVIRONMENT-IDENTIFIER>.<CUSTOMER_IDEN
  TIFIER>.cloudera.site
  ```

  An example could be:

  ```
  my-dataeng-master0.my-envir.aaaa-1234.cloudera.site
  ```

  This name has the following components:

  - The base domain is cloudera.site. This is a publicly registered “DNS suffix”. It is also a registered Route53 hosted zone in a Cloudera owned AWS account.
  - The <CUSTOMER_IDENTIFIER> is unique to a customer account on Cloudera made of alphanumeric characters and a “-” (dash).
  - The <ENVIRONMENT_IDENTIFIER> is generated based on the environment name and is truncated to 8 characters.
  - The <CLUSTER_NAME> is the cluster name given to the Data Lake or Cloudera Data Hub. It is appended with a <HOST_GROUP> name such as “gateway”, “master”, “worker”, and so on, depending on the role that the node plays in the cluster. If there are more than one of these nodes playing the same role, they are appended with a serial number <i>.
- The DNS name of the endpoints of the Cloudera data services is of the following format:
  - For a Virtual Warehouse in Cloudera Data Warehouse, it is:

    ```
    <VIRTUAL_WAREHOUSE_NAME>.dw-<CDW_ENVIRONMENT_IDENTIFIER>.<CUSTOMER_IDENT
    IFIER>.cloudera.site
    ```

    - <VIRTUAL_WAREHOUSE_NAME> is the name of the Virtual Warehouse created. There could be multiple virtual warehouses for a given Cloudera environment.
    - <CDW_ENVIRONMENT_IDENTIFIER> is the identifier for the Cloudera environment.
  - For a Session Terminal in Cloudera AI workbench, it is:

    ```
    <TTY_SESSION_ID>.<CML_WORKSPACE_ID>.<ENVIRONMENT_IDENTIFIER>.<CUSTOMER_I
    DENTIFIER>.cloudera.site
    ```

    - <TTY_SESSION_ID> is the ID of the Cloudera AI Terminal Session ID.
    - <CML_WORKSPACE_ID> is the ID of the Cloudera AI workspace created.
    - The <ENVIRONMENT_IDENTIFIER> is generated based on the environment name and is truncated to 8 characters. If the 8th character is a “-” (dash), then it is truncated to 7 characters instead.
  - For all the Cloudera data services listed above, the common portions of the DNS include:
    - The base domain is cloudera.site. This is a publicly registered “DNS Suffix”. It is also a registered Route53 hosted zone in a Cloudera owned AWS account.
    - The <CUSTOMER_IDENTIFIER> is unique to a customer account on Cloudera made of alphanumeric characters and a “-” (dash).
  - The length of the DNS name is restricted to 64 characters due to some limitations with Hue workloads.
  - These names are stored as A records in the Route53 hosted zone in the Cloudera managed Cloudera Control Plane AWS account. Hence, you can resolve these names from any location outside of the VPC. However, note that they would still resolve to private IP addresses and hence are constrained by the connectivity setup described in preceding sections.
  - Within a Cloudera environment, the DNS resolution happens differently. Every Cloudera environment has a DNS server that is played by a component called FreeIPA. This server is seeded with the hostnames of the nodes of all workload clusters in the environment. Every node in a Data Lake, Cloudera Data Hub and data service is configured to look up the FreeIPA DNS service for name resolution within the cluster.
  - FreeIPA nodes are internal to the Cloudera environment both for resolving and use. They don't have public DNS records.

Related Information

DNS Suffix

DNS

Azure-provided DNS is recommended for the VNet. This is required to enable private AKS clusters and private access to Postgres databases among others, as described in “Private endpoints” above.

Cloudera Public Cloud

cdpcurl

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

cdpcurl.....4
Building cdpcurl.....4
Generating an API access key.....4
Passing the API access keys.....5
Making API calls with cdpcurl.....6
cdpcurl example.....6

cdpcurl

This documentation provides information about cdpcurl.

Building cdpcurl

You can access cdpcurl and review the license here.

Procedure

1. Download cdpcurl:

```
wget https://github.com/cloudera/cdpcurl/archive/master.zip
unzip master.zip
cd cdpcurl-master
```

2. If desired, create a virtual environment:

   For a typical virtual environment:

```
$ virtualenv cdpcurlenv
$ . cdpcurlenv/bin/activate
```

   For a python virtual environment:

```
$ pyenv virtualenv cdpcurlenv
$ pyenv activate cdpcurlenv
```

3. In this directory, run the following command:

   $ pip install .

Generating an API access key

A Cloudera user account (a user or a machine user) must have API access credentials to access Cloudera services through the CDP CLI or API.

When you use this method to generate an access key and then manually configure the access key in the ~/.cdp/credentials, the access credentials are permanent until they are removed from the ~/.cdp/credentials file. A login command is not required if access credentials are stored in the ~/.cdp/credentials file. If you prefer that the API access key is shorter-lived, refer to the topic *Logging into the CDP CLI/SDK*, which describes a method of logging into the CLI/SDK through any SAML-compliant identity provider.

Required roles: Users who have the IAMUser role can generate an API access key from their own account page. As a Cloudera administrator or PowerUser, you can generate an API access key for all user accounts.

Generate your own access key

Steps

1. Sign in to the Cloudera console.
2. Click on your user name in the bottom left corner and then select Profile.
3. On the user profile page that appears, click Generate Access Key.

4. Cloudera creates the key and displays the information on the screen.

   Copy the access key and private key to a text file and send it to the Cloudera user who requires it. The private key is a very long string of characters. Make sure that you copy the full string. You can optionally download the credentials file containing the access key information.
5. Click OK to exit the access key window.

   Note:

   The Cloudera console displays the API access key immediately after you create it. You must copy or download the access key ID and private key information when it is displayed. Do not exit the console without copying the private key. After you exit the console, there is no other way to view or copy the private key.

Once you've generated the access key, you can configure CDP CLI, SDK, or other utilities that require it.

Generate an access key for another user or machine user (admin only)

Steps

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. On the side navigation panel, click Users.
4. On the Users page, click the name of the user or machine user account for which you want to generate an access key.
5. On the user account page, go to the Access Keys section and click Generate Access Key.
6. Cloudera creates the key and displays the information on the screen.

   Copy the access key and private key to a text file and send it to the Cloudera user who requires it. The private key is a very long string of characters. Make sure that you copy the full string. You can optionally download the credentials file containing the access key information.
7. Click OK to exit the access key window.

   Note:

   The Cloudera console displays the API access key immediately after you create it. You must copy or download the access key ID and private key information when it is displayed. Do not exit the console without copying the private key. After you exit the console, there is no other way to view or copy the private key.

Once you've generated the access key, you can configure CDP CLI, SDK, or other utilities that require it.

Passing the API access keys to cdpcurl

After you generate the API access key and private key, you must pass the keys to cdpcurl.

You have two options for passing the API access key and private key to cdpcurl:

- Recommended method: Create a profile in $HOME/.cdp/credentials that contains the keys, and then use the --profile option in cdpcurl calls. For example:

  ```
  [myuserprofile]
  cdp_access_key_id = 6744f22e-c46a-406d-ad28-987584f45351
  cdp_private_key = abcdefgh.....................................=
  ```
- Alternatively, when you make the API call, pass the keys to cdpcurl using the --access-key and --private-key options.

Making API calls with cdpcurl

To use cdpcurl to make API calls, first form the correct URI using the API reference documentation, and then construct the request.

To form the required URI, start by determining the hostname based on the service being called:

- iam:
  - iamapi.us-west-1.altus.cloudera.com if your Cloudera account runs in Cloudera Control Plane region us-west-1
  - api.<control-plane-region>.cdp.cloudera.com if your Cloudera account runs in any other Cloudera Control Plane region
- All other services: api.<control-plane-region>.cdp.cloudera.com

Then form the URI. The correct URI is an HTTPS URL at the chosen host, with a path indicated for your desired endpoint in the API documentation. For example, the URI for a listEnvironments call would look something like this:

```
https://api.us-west-1.cdp.cloudera.com/api/v1/environments2/listEnvironments
```

Most Cloudera API calls are POST requests, so be sure to use the [-X REQUEST] and [-d DATA] arguments in the command. These are briefly documented in the awscurl documentation, but [-X REQUEST] specifies the type of request (most often POST in Cloudera) and [-d DATA] specifies the request content (in the POST example below, an empty JSON request). If the -d option value begins with "@", then the remainder of the value is the path to a file containing the content; otherwise, the value is the content itself.

You can then run the request. For example:

```
$ cdpcurl --profile sandbox -X POST -d '{}' https://api.us-west-1.cdp.cloude
ra.com/api/v1/environments2/listEnvironments
```

For a complete list of cdpcurl options, run cdpcurl --help.

Note: The CDP CLI has a --debug option that prints out details that might be helpful when using cdpcurl. For example, it returns the hostname that it calls to, which may help if you're unsure whether to use api.us-west-1.cdp.cloudera.com or something else. It also reports on the path (/api/v1/...) that it calls.

cdpcurl example

After you install cdpcurl and generate the API access key, you are ready to make an API call.

Before you begin

If you have not created a profile in $HOME/.cdp/credentials that contains the API access key ID and private key pair, make sure that you have the API access key ID and private key that you generated.

About this task

The example below walks you through the steps required to make an API call that lists all environments in the CDP Control Plane.

Procedure

1. In the CDP Control Plane Public API Documentation, find the endpoint for the list environments request and form the correct URI.

   The correct URI is an HTTPS URL that contains the hostname for the desired service and the path to the desired endpoint. In this "list environments" example, the correct URI for a CDP account running in the us-west-1 Control Plane region would be:

   ```
   https://api.us-west-1.cdp.cloudera.com/api/v1/environments2/listEnvironm
   ents
   ```

2. Form the cdpcurl command.

   The CDP API uses the HTTPS protocol to make API calls, and most CDP API calls are POST requests, so be sure to use the [-X REQUEST] and [-d DATA] arguments in the command. For more details, refer to the awscurl documentation and the topic *Making API calls with cdpcurl*. The --profile argument below points to the profile you likely created in $HOME/.cdp/credentials, that contains the API access key and private key pair. If you did not create this profile, you will have to pass the --access_key and --private_key arguments as well.

   In this example, the command would be:

   ```
   $ cdpcurl --profile sandbox -X POST -d '{}' https://api.us-west-1.cdp.cl
   oudera.com/api/v1/environments2/listEnvironments
   ```

3. Run this command in a terminal window to make the API call, and then receive the response. If the call is successful, the response is the EnvironmentSummary array.

Cloudera Public Cloud

Data services tools

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Iceberg in Cloudera.....4

Apache Iceberg in Cloudera

Apache Iceberg is a cloud-native, high-performance open table format for organizing petabyte-scale analytic datasets on a file system or object store. Combined with Cloudera, users can build an open data lakehouse architecture for multi-function analytics and to deploy large scale end-to-end pipelines.

Open data lakehouse on Cloudera simplifies advanced analytics on all data with a unified platform for structured and unstructured data and integrated data services to enable any analytics use case from ML, BI to stream analytics and real-time analytics. Apache Iceberg is the secret sauce of the open lakehouse.

The following table shows the support for Iceberg in Cloudera and below the table Iceberg versions v1 and v2 are defined:

Table 1: Iceberg Support Matrix

Release	Iceberg support level	SQL Engine				
		Impala	Hive	Spark	NiFi	Flink
Cloudera Data Hub						
Cloudera Data Hub 7.3.1	GA	v1, v2: create table, read, insert, update, and delete	v1, v2: create table, read, insert, update, delete	v1, v2: create table, read, insert, update, and delete	v1, v2: read and insert	v1, v2: create table, read, append, overwrite ***
Cloudera Data Hub 7.2.18	GA	v1, v2: create table, read, insert, update, and delete	v1, v2: create table, read, insert, update, delete	v1, v2: create table, read, insert, update, and delete	v1, v2: read and insert	v1, v2: create table, read, append, overwrite ***
Cloudera Data Hub 7.2.17	GA	v1, v2: create table, read, insert,	v1, v2: create table, read, insert, update, delete	v1, v2: create table, read, insert, update, and delete	v1, v2: read and insert	v1, v2: create table, read, append, overwrite ***
Cloudera Data Hub 7.2.16.2	GA	v1, v2: create table, read	v1: create table, read, insert, update, delete	v1, v2: create table, read, insert, update, and delete	v1, v2: read and insert	v1: create table, read, and insert
Cloudera Private Cloud Base						
Cloudera Private Cloud Base 7.3.1	GA	v1, v2: create table, read, insert, delete	v1, v2: create table, read, insert, delete	v1, v2: create table, read, insert, update, and delete	v1, v2: read and insert	v1, v2: create table, read, and insert
Cloudera Private Cloud Base 7.1.9	GA	v1, v2: create table, read, insert	No Iceberg support	v1, v2: create table, read, insert, update, and delete	v1, v2: read and insert	v1, v2: create table, read, and insert
Cloudera Private Cloud Base 7.1.7 SP2, 7.1.8	No Iceberg support					
Cloudera Public Cloud Data Services						
Cloudera Public Cloud Data Services	GA	v1, v2: create table, read, insert, update, and delete	v1, v2: create table, read, insert, update, and delete	v1, v2: create table, read, insert, update, and delete	v1, v2: read and insert	N/A
Cloudera Private Cloud Data Services						

Release	Iceberg support level	SQL Engine				
		Impala	Hive	Spark	NiFi	Flink
Cloudera Private Cloud Data Services 1.5.4	GA (Cloudera Private Cloud Base 7.1.9) Technical Preview (Cloudera Private Cloud Base 7.1.7, 7.1.8)	v1, v2: create table, read, insert, update, and delete	v1, v2: create table, read, insert, update, and delete	v1, v2: create table, read, insert, update, and delete	v1, v2: read and insert (7.1.9 Base)	v1, v2: create table, read, and insert (7.1.9 Base)
Cloudera Private Cloud Data Services 1.5.3	GA (Cloudera Private Cloud Base 7.1.9) Technical Preview (Cloudera Private Cloud Base 7.1.7, 7.1.8)	v1, v2: create table, read, insert, update, and delete	v1, v2: create table, read, insert, update, and delete	v1, v2: create table, read, insert, update, and delete	v1, v2: read and insert (7.1.9 Base)	v1, v2: create table, read, and insert (7.1.9 Base)
Cloudera Private Cloud Data Services 1.5.2	GA (7.1.9 Base) Technical Preview (Cloudera Private Cloud Base 7.1.7, 7.1.8)	v1, v2: create table, read, insert, and delete	v1, v2: create table, read, insert, update, and delete	v1, v2: create table, read, insert, update, and delete	v1, v2: read and insert (7.1.9 Base)	v1, v2: create table, read, and insert (7.1.9 Base)
Cloudera Private Cloud Data Services 1.5.1 2023.0.13.0-20	Technical Preview (Cloudera Private Cloud Base 7.1.7, 7.1.8)	v1, v2: create table, read	v1, v2: create table, read, insert, update, and delete	v1, v2: create table, read, insert, update, and delete	No Private Cloud support	No Private Cloud support

** The support for delete operations, except from Flink, shown in this table is limited to position deletes. Equality deletes are not supported in these releases except from Flink.

*** Iceberg v2 updates and deletes from Flink are a technical preview in Cloudera Public Cloud 7.2.17.

The Apache Iceberg format specification describes the following versions of tables:

- v1

  Defines large analytic data tables using open format files.
- v2

  Specifies ACID compliant tables including row-level deletes and updates.

Table 2: Iceberg Docs and Availability Matrix

Release	Docs	Iceberg Support Level
Cloudera Private Cloud Base 7.3.1	Iceberg support for Hive	GA
Open Data Lakehouse (Cloudera Private Cloud Base 7.1.9)	Iceberg in Open Data Lakehouse	GA
	Iceberg support for Atlas	GA
	SQL Stream Builder with Iceberg (CSA 1.11) and Flink with Iceberg (CSA 1.11) Iceberg replication policies	GA
Cloudera Data Engineering Public Cloud	Using Iceberg	GA
Cloudera Data Warehouse Public Cloud	Iceberg features	GA
Cloudera Data Engineering Private Cloud	Using Iceberg	Technical Preview
Cloudera Data Warehouse Private Cloud	Iceberg introduction Moving data into Iceberg tables on Cloudera Data Warehouse	GA (Cloudera Private Cloud Base 7.1.9), Technical Preview (Cloudera Private Cloud Base 7.1.7-7.1.8)
Cloudera Data Hub 7.2.16 and later	Iceberg features Iceberg features	Technical Preview

Release	Docs	Iceberg Support Level
Cloudera Data Hub 7.2.17 and later	Iceberg in Apache Atlas Iceberg in Apache Atlas	Technical Preview
	Streaming Analytics Iceberg support in Flink	GA
	Flink/Iceberg connector	GA
	Using NiFi to ingest data into Cloudera Data Warehouse Public Cloud in Iceberg table format	GA
Cloudera Data Hub 7.2.18	Iceberg in Apache Atlas Iceberg in Apache Atlas	GA
Cloudera Data Hub 7.3.1	Iceberg features No new features in this release	GA
Cloudera Flow Management for Cloudera Private Cloud	Using the PutIcebergCDC processor	Technical Preview
	Using NiFi to ingest data into Cloudera Data Warehouse Private Cloud in Iceberg table format	GA
Cloudera DataFlow	Using the Kafka to Apache Iceberg ReadyFlow	GA
Cloudera AI Public Cloud	Connection to Iceberg	GA

Migrating Data to CDP

Date published: 2023-08-14
Date modified: 2023-08-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Overview 5

Migrating data from CDH to Cloudera 5
Migrating HDFS and Hive data from CDH to Cloudera 5
Migration prerequisites 5
Introduction to Cloudera Replication Manager 7
HDFS data migration from CDH to Cloudera 13
Hive migration from CDH to Cloudera 17
Migrating Oozie workflows from CDH to Cloudera 21
Migrating HDFS and Hive data from CDH to Cloudera 24
Migrating HDFS native permissions to Cloudera 24
Extracting HDFS native permissions 24
Converting HDFS native permissions into Ranger HDFS policies 26
Transforming Ranger HDFS policies into Ranger S3 policies 26
Importing Ranger AWS S3 policies 28
Migrating workflows directly created in Oozie to Cloudera 30
Migrating Sentry policies from CDH to Cloudera 31
About Migrating Sentry policies 31

Migrating data from HDP to Cloudera 39
Migrating HDFS data from HDP to Cloudera 39
Migration prerequisites 39
About DistCp tool 40
Unbanning hdfs user in HDP cluster 40
Before migrating 41
HDFS data migration from HDP to Cloudera 41
Migrating HDFS native permissions to Cloudera 41
Extracting HDFS native permissions 42
Converting HDFS native permissions into Ranger HDFS policies 43
Transforming Ranger HDFS policies into Ranger S3 policies 44
Importing Ranger AWS S3 policies 45
Migrating Ranger policies from HDP to Cloudera 48
About Migrating Ranger policies 48
Performing Export and Transform operations 49
Performing Import operation 50
Supported Input parameters for Export operation 54
Supported Input parameters for Transform operation 56
Migrating Hive data from HDP 2.x or HDP 3.x to Cloudera 59
Migration prerequisites 62
HMS Mirror command summary 66
Migrating Hive metadata 67
HMS Mirror generated files 69
Verifying metadata migration 70
Migrating actual Hive data 70
Adjust AVRO table schema URLs 71
Verifying actual Hive data migration 71
Table locations 72

Fixing statistics..... 72
Changes to HDP Hive tables..... 72

Overview

You can migrate HDFS and Hive data from CDH and HDP to CDP. You learn about semantic changes in component versions that affect migration.

Migrating data from CDH to Cloudera

How to migrate data from CDH to Cloudera.

Migrating HDFS and Hive data from CDH to Cloudera

An overview of the migration process from CDH to Cloudera prepares you to migrate HDFS and Hive data to the AWS S3 endpoint.

Once fulfilling migration prerequisites, you set up security for communications between the CDH on-premise cluster and the destination cloud services. You can use the Cloudera Replication Manager service to migrate HDFS and Hive data from CDH or HDP to Cloudera

Migration prerequisites

You need to know the prerequisites for migrating HDFS and Hive data from CDH to Cloudera.

You must meet the following general prerequisites before starting the migration process:

- AWS is your Cloud provider. Migrating to Azure / GCP is currently not supported.
- Classic clusters (CDH/HDP/Cloudera Private Cloud Base) are on-premises clusters that host the data to be replicated to the Cloudera deployment. These clusters are on-premises versions of Cloudera and must be registered on the Management Console before they can be used for data migration purposes. For more information, see Support Matrix for Cloudera Replication Manager.
- Validate the CDH source cluster as per cluster requirements.
- CDH clusters must have been created using Cloudera Manager.
- Clusters that are not managed by Cloudera Manager cannot be registered to Cloudera.
- You must log in as the admin user to Cloudera Manager in the CDH cluster.
- Replication user needs to have hadoop admin privilege on source cluster.
- On the source cluster which is running CDH, make sure the user we specify on the replication manager source cluster side has hadoop admin privileges.
- Your source cluster supports compatible versions of Cloudera Manager and Cloudera Runtime shown in the following table:

Source cluster	Earliest supported Cloudera Manager	Earliest supported Cloudera Runtime
CDH 5	6.3.0	5.1.0
CDH 6	6.3.0	6.1.0
CDH 6	7.3.1	6.3.3

The only required migration tool is Cloudera Replication Manager, which is pre-installed in Cloudera Manager and licensed for use in Cloudera Data Hub.

Specifically note that:

- Your CDH cluster is registered on the CDP Management Console. For more information, see Adding a CDH cluster.
- Open communication ports between the source CDH cluster and Cloudera list.

- Set up line-of-sight from the CDH cluster to the AWS endpoint.
- Ensure adequate network connectivity between the CDH cluster and AWS endpoint.
- Create an external account on the CDH cluster that has a non-expiring access credentials and secret key pair.
- You must have the license to perform your tasks in Cloudera Replication Manager. To understand more about Cloudera license requirements, see Managing Licenses.
- Allocate storage on HDFS to create HDFS snapshots that allows administrators to modify the active file-system.
- You must be a part of the super-group on each host on the CDH cluster.

Ports for Cloudera Replication Manager on Cloudera Public Cloud

You must open certain ports on the CDH source host to allow communication between the source on-premises cluster and Cloudera.

Connectivity required	Default Port	Type	Description	Required for the following replication scenarios
Data transfer from CDH cluster hosts to AWS S3	80 or 443 (TLS)	Outbound	Outgoing port. All CDH nodes must be able to access S3 endpoints	HDFS to Cloud replication (S3 Target) Hive Replication
Cloudera Manager Admin Console HTTP	7180 or 7183 when TLS enabled	Inbound	Incoming port. Open on the source cluster to enable the target Cloudera Manager (in cloud) to communicate to the on-premises Cloudera Manager.	Hive Replication (Not applicable for HDFS migration)
Classic cluster setup (if using Cloudera Replication Manager App)	Varies based on CCMv1 (6000-6049) or CCMv2 (443)	Outbound	Connecting the source CDH cluster to the Cloudera Management Console via Cluster Connectivity Manager Outbound network access for Cluster Connectivity Manager documented here and Cluster Connectivity Manager overview is documented here	Required when using the Cloudera Replication Manager UI/App

Setting up an external account

As CDH Administrator, you need to create an External Account on the CDH cluster that has an access key and secret key pair for accessing the cloud storage used in the Cloudera environment.

About this task

Using the external account along with the cloud storage path, administrators can create one or more HDFS replication schedules to migrate the HDFS data from the CDH cluster to the cloud storage used in the CDP environment.You must have valid Amazon S3 credentials to access cloud storage for migrating data.

You must raise a support case with Cloudera to provide the key pair.

You must have valid Amazon S3 credentials to access cloud storage for migrating data. You add the access key / secret key pair to the account to access Cloudera cloud storage s3a://<data lake bucket path>. The access key / secret key pair ensures that the external account configured on CDH can successfully access the S3 buckets used by Data Lake.

Procedure

1. In Cloudera Manager#Administration, click Add Access Key Credentials.
2. In Edit Credential, name the credential and enter the AWS Access Key ID and AWS Secret Key.
3. Save changes,

Setting up SSL/TLS certificate exchange

In order to replicate data securely, you must configure an SSL/TLS certificate exchange between the two Cloudera Manager instances that manage the source and target clusters.

About this task

The following sample commands use the open-jdk-11 java version. Use the Java version that you use in Cloudera clusters in these commands.

Procedure

1. List the contents of the trustore file located in the source cluster Cloudera Manager.

```
/usr/lib/jvm/java-openjdk-11/bin/keytool -list -keystore /var/lib/
cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks -store
pass [***PASSWORD***]
```

   The global truststore password is available in the /etc/hadoop/conf/ssl-client.xml file.

2. Export the certificate contents to the cert.txt file.

```
/usr/java/default/bin/keytool -exportcert -keystore /var/lib/cloudera-scm-
agent/agent-cert/cm-auto-global_truststore.jks -alias cmrootca-0 -file ./c
ert.txt -storepass [***TRUSTSTORE_PASS***]
```

3. Copy the cert.txt file to all the hosts of the destination cluster Cloudera Manager securely.

```
$ mv cert.txt othercert.txt
$ scp
```

4. Import the certificate into the trustore file on all the hosts of the destination cluster Cloudera Manager.

```
/usr/java/default/bin/keytool -importcert -noprompt -v -trustcac
erts -keystore /var/lib/cloudera-scm-agent/agent-cert/cm-auto-gl
obal_truststore.jks -alias cmrootca-1 -file ./othercert.txt --storep
ass [***TRUSTSTORE_PASSWORD***]
```

5. Run steps 1 and 2 in the target cluster Cloudera Manager, and then copy the cert.txt file to all the hosts in the source cluster Cloudera Manager securely, and import the certificate into the trustore file on all the hosts of the source cluster Cloudera Manager (steps 3 and 4).

Cloudera license requirements for

You must have the required license to perform your tasks in .

Specifically, the source cluster must have a Enterprise license in order to use the Replication Manager.

To learn more about license requirements, see Managing Licenses.

Introduction to Cloudera Replication Manager

Cloudera Replication Manager is a service to copy and migrate data from CDH 5.13+ version and above clusters to Cloudera endpoint. The currently supported Public Cloud service is Amazon S3.

Accessing the Cloudera Replication Manager service

You can access the Cloudera Replication Manager service by logging into Cloudera.

When you log into Cloudera, the Cloudera Public Cloud web interface appears. Click Replication Manager. The Overview page of the Cloudera Replication Manager appears.

The Cloudera Replication Manager has the following pages:

- Overview
- Classic Clusters

- Cloud Credentials
- Replication Policies

The following image shows the Cloudera Public Cloud web interface:

You can also access the Cloudera Replication Manager service by logging into the Cloudera Management Console. In Cloudera Management Console, click the

icon and select Cloudera Replication Manager.

How replication policies work

In Cloudera Replication Manager, you create replication policies to establish the rules you want applied to your replication jobs.

The policy rules you set can include which cluster is the source and which is the destination, what data is replicated, what day and time the replication job occurs, the frequency of job runs, and bandwidth restrictions.

The first time you run a job (an instance of a policy) with data that has not been previously replicated, the Cloudera Replication Manager creates a new folder or database and bootstraps the data. During a bootstrap operation, all data is replicated from the source cluster to the destination. As a result, the initial execution of a job can take a significant amount of time, depending on how much data is being replicated, network bandwidth, and so on. So you should plan the bootstrap operation accordingly.

After the bootstrap operation succeeds, an incremental data replication is automatically performed. This job synchronizes, between the source and destination clusters, any events that occurred during the bootstrap process. After the data is synchronized, the replicated data is ready for use on the destination. Data is in a consistent state only after incremental replication has captured any new changes that occurred during bootstrap.

Subsequent replication jobs from the same source location to the same target on the destination are incremental, so only the changed data is copied.

When a bootstrap operation is interrupted, such as due to a network failure or an unrecoverable error, the Cloudera Replication Manager does not retry the job instead it runs the job at the next scheduled interval, if available. Therefore, if the bootstrap operation is interrupted, you must manually correct the issue and then run the policy.

When scheduling how often you want a replication job to run, you should consider the recovery point objective (RPO) of the data being replicated; that is, what is the acceptable lag time between the active site and the replicated data on the destination.

The following diagram shows the HDFS replication architecture:

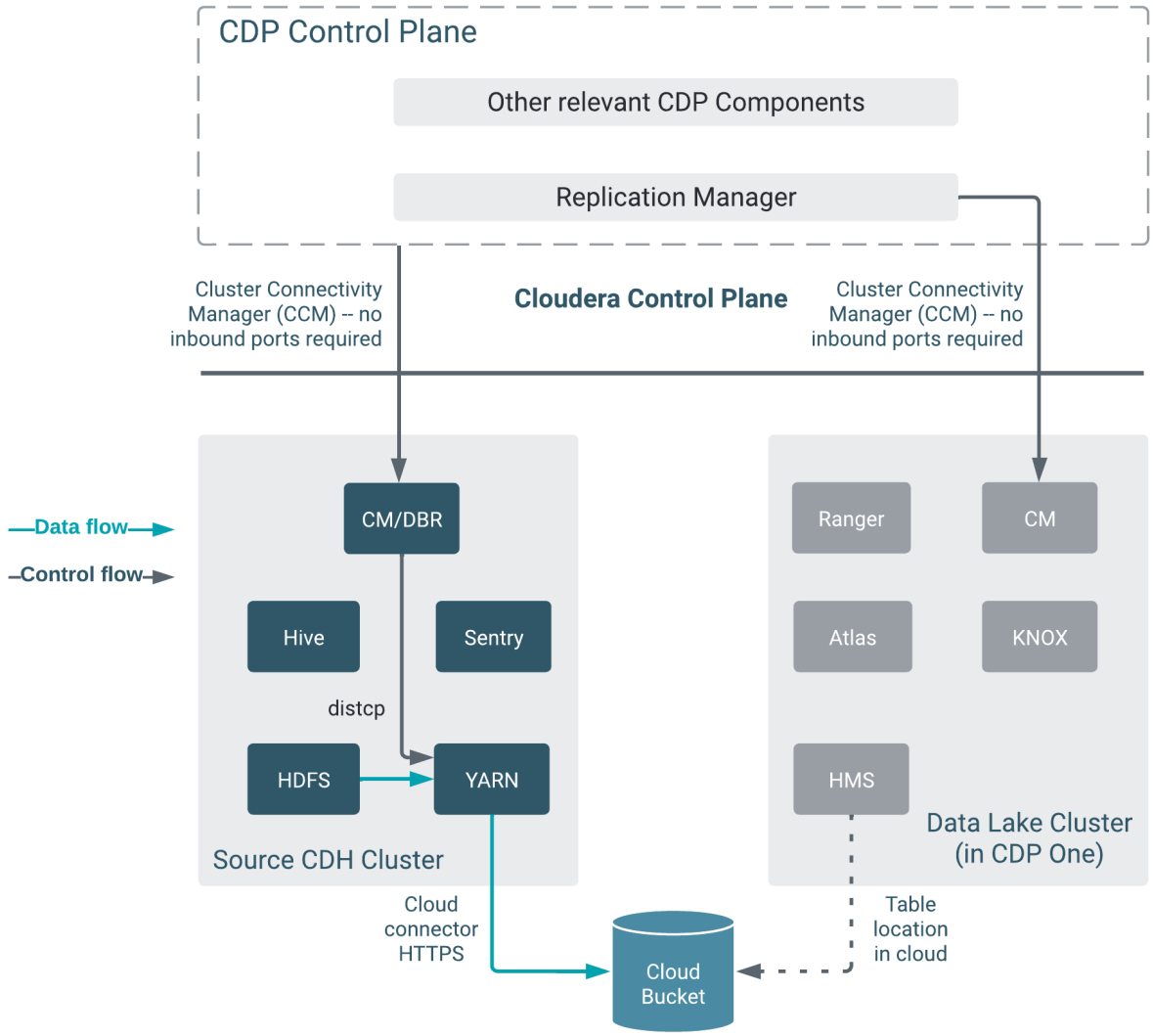

The following diagram shows the Hive replication architecture:

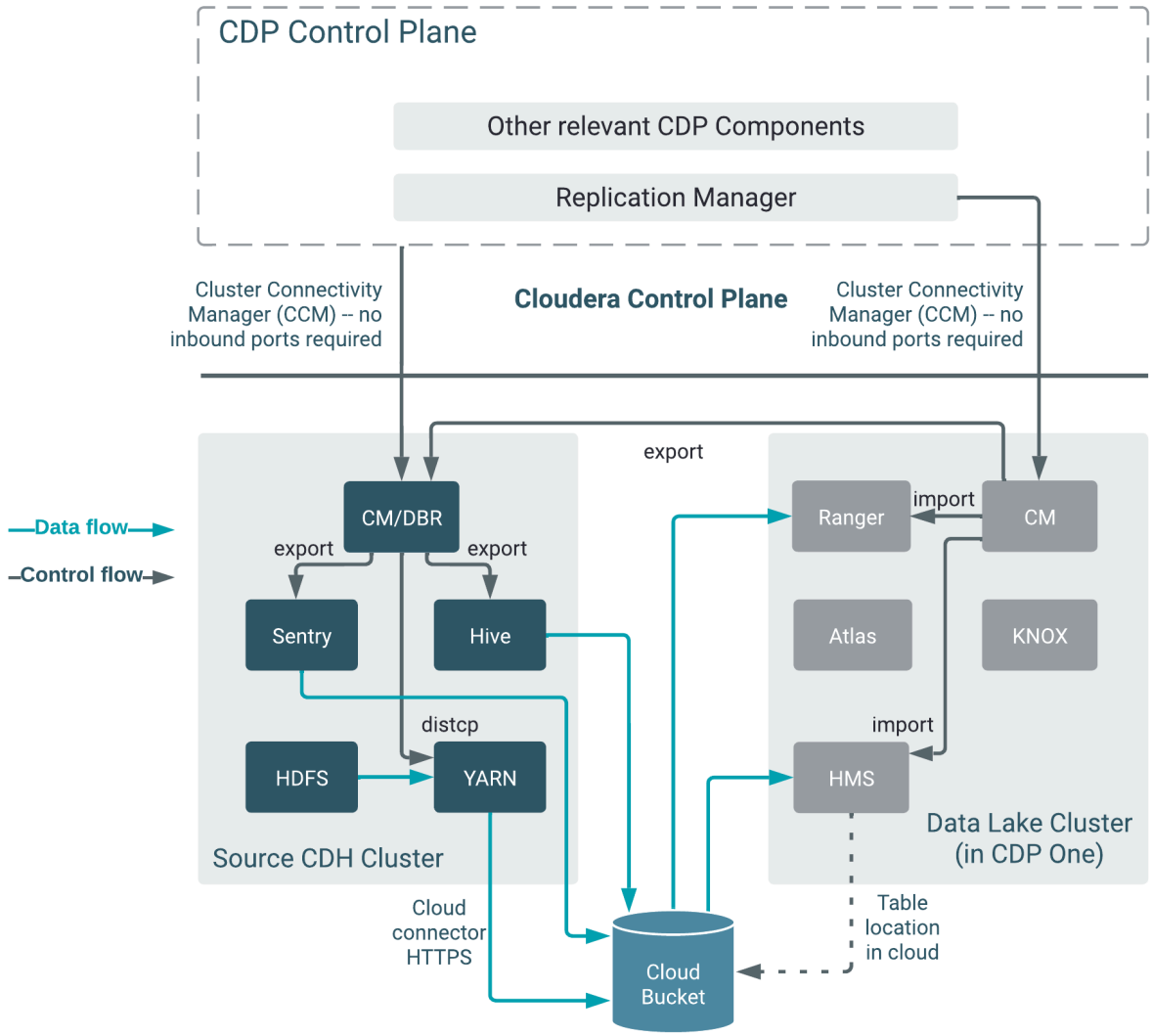

Replication policy considerations

You should take into consideration certain guidelines when creating or modifying a replication policy. It is important for you to understand the security restrictions and encryption policies within Cloudera Replication Manager.

The guidelines you need to consider before you create or modify a replication policy includes:

Data security

To use an S3 cluster for your policy, register your credentials on the Cloud Credentials page.

A user with access to the Replication Manager user interface has the ability to browse, within the Cloudera Replication Manager UI, the folder structure of any clusters enabled for Cloudera Replication Manager.

Therefore, users can view folders, files, and databases in the Replication Manager user interface, that they might not have access to in HDFS. Users cannot view from the Cloudera Replication Manager UI the content of files on the source or destination clusters. Nor do these administrators have the ability to modify or delete folders or files that are viewable from the Cloudera Replication Manager UI.

Replication Manager policy properties and settings

Consider the recovery point objective (RPO) of the data being replicated when you schedule a replication policy. The RPO is the acceptable lag time between the active site and the replicated data on the destination. Ensure that the frequency is set so that a job finishes before the next job starts.

Jobs based on the same policy cannot overlap. If a job is not completed before another job starts, the second job does not execute and is given the status Skipped. If a job is consistently skipped, you might need to modify the frequency of the job.

Specify bandwidth per map, in MBps. Each map is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value.

Cluster requirements

- Pair the clusters before you include them in a replication policy.
- With a single cluster, you can replicate data on-premises to the cloud.
- With a single cluster, you cannot replicate data on-premises to on-premises.
- If the clusters are Replication-Manager enabled, it appears in the Source Cluster or Destination in the Create Policy wizard. You must ensure that the clusters you select are healthy before you start a policy instance (job).

Working with cloud credentials

The Cloud Credentials page shows the registered cloud credentials for Cloudera Replication Manager. To replicate data to a storage cloud account, you must register the cloud credentials, so that the Cloudera Replication Manager can access your cloud account. The supported cloud storage accounts are Amazon S3.

On the Cloud Credentials page, you can add, update, or delete cloud credentials. Before you register an Amazon S3 cloud account, ensure the cloud bucket requirements are met.

Adding cloud credentials

You can perform the following tasks on the Cloud Credentials page to manage cloud credentials.

About this task

You can add cloud credentials to migrate data to the cloud.

Unregistered credentials can impact the replication process. Credentials associated with a cluster node that do not have updated credentials are called unregistered credentials. For example, if a node is down when the credentials are changed on a bucket or when the node is brought up that has the old credentials.

Procedure

1. Log into Public Cloud Cloudera Management Console > Replication Manager.
2. Click Cloud Credentials on the left pane, click Add.

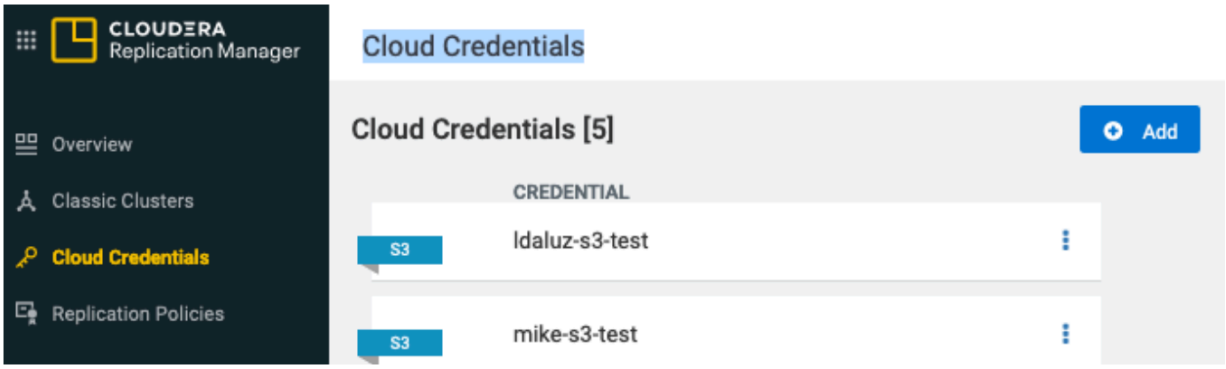

3. Fill out the information as seen in the image and click Validate.

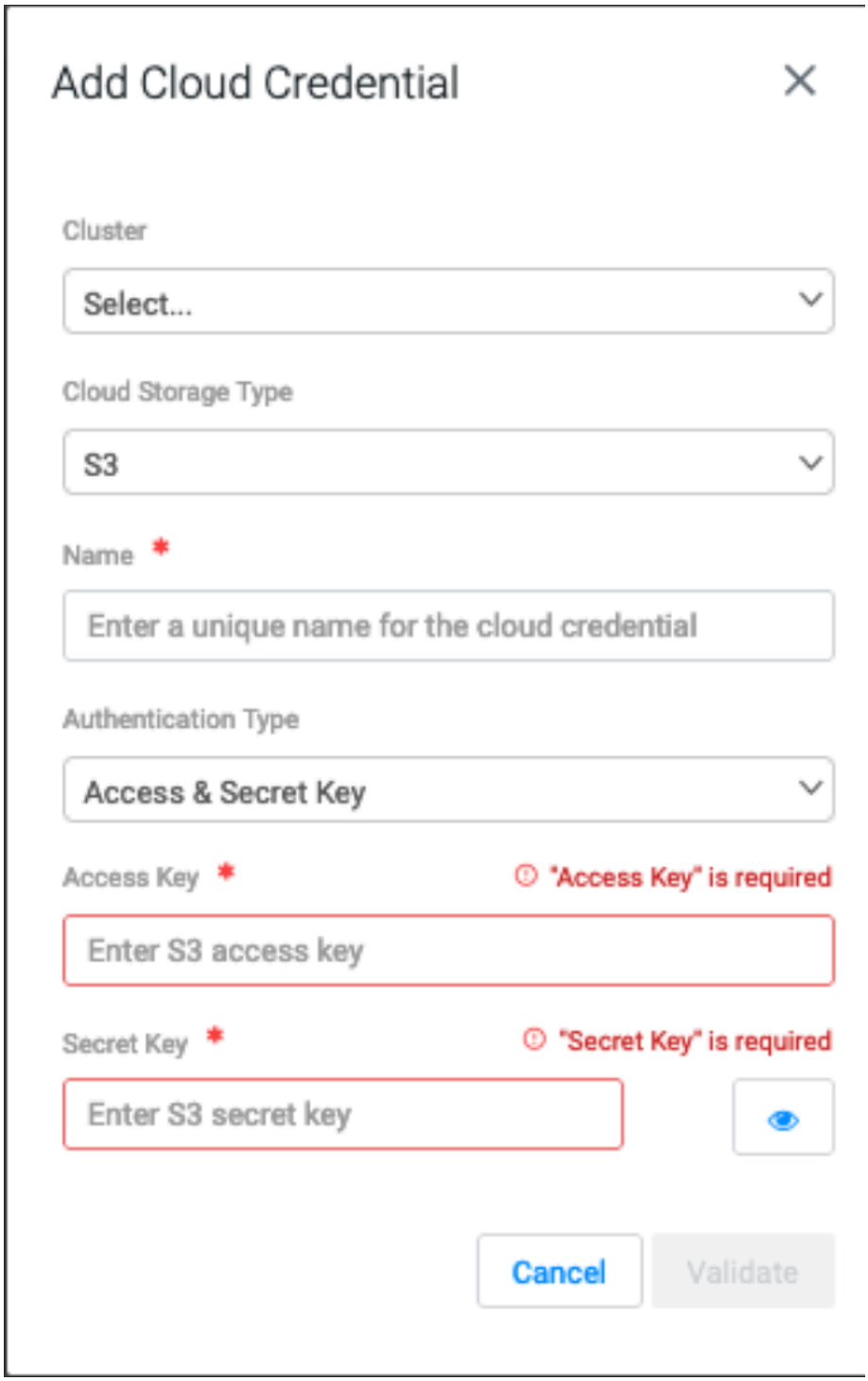

4. Once validated successfully, click Save.

Update cloud credentials

You can update the cloud credentials based on various factors.

When the bucket configuration such as secret or access keys, bucket name or endpoint, and encryption type is changed, it can affect the Replication Manager replication policy run and might require an update to the Cloudera Replication Manager cloud credentials.

Credential changes are picked up by the next run of the policy. When you change the credentials, the in-progress policy runs might fail but the succeeding runs pick up the changes.

To update a cloud credential, click Actions > Update option.

Delete cloud credentials

You can delete unwanted credentials from the Replication Manager.

When you delete cloud credentials, the replication policies that use the deleted cloud credentials might fail. To avoid failures, delete the Cloudera Replication Manager cloud policies associated with the deleted credentials and recreate the policies with the new credentials. You can view a list of policies associated with specific credentials on the Cloud Credentials page.

To delete a cloud credential, click Actions > Delete option.

HDFS data migration from CDH to Cloudera

To create a HDFS replication policy from on-premises to the cloud account, you must register your cloud account credentials with Replication Manager service, so that Replication Manager can access your cloud storage.

Before performing HDFS replication using CDH clusters, see Working with cloud credentials.

Replication Manager supports replication of HDFS data from cluster to cloud storage. The replication policy runs on the cluster and pushes the data from cloud storage. The cluster can be an on-premises or IaaS cluster with data on local HDFS. The cluster requires HDFS, YARN, Ranger, and Knox services to perform replication.

You can schedule taking HDFS snapshots for replication in the Replication Manager. HDFS snapshots are read-only point-in-time copies of the filesystem. You can enable snapshots on the entire filesystem, or on a subtree of the filesystem. In Replication Manager, you take snapshots at a dataset level. Understanding how snapshots work and some of the benefits and costs involved can help you to decide whether or not to enable snapshots.

For more information, see HDFS snapshots.

Creating a HDFS replication policy

You can use a HDFS replication policy to replicate data from on-premises to cloud. Before you create a new replication policy, you must register a cloud account with the Cloudera Replication Manager service.

Procedure

1. On the Cloudera Management Console > Replication Manager > Replication Policies page, click Add Policy.
2. In the Create Replication Policy wizard, select HDFS.

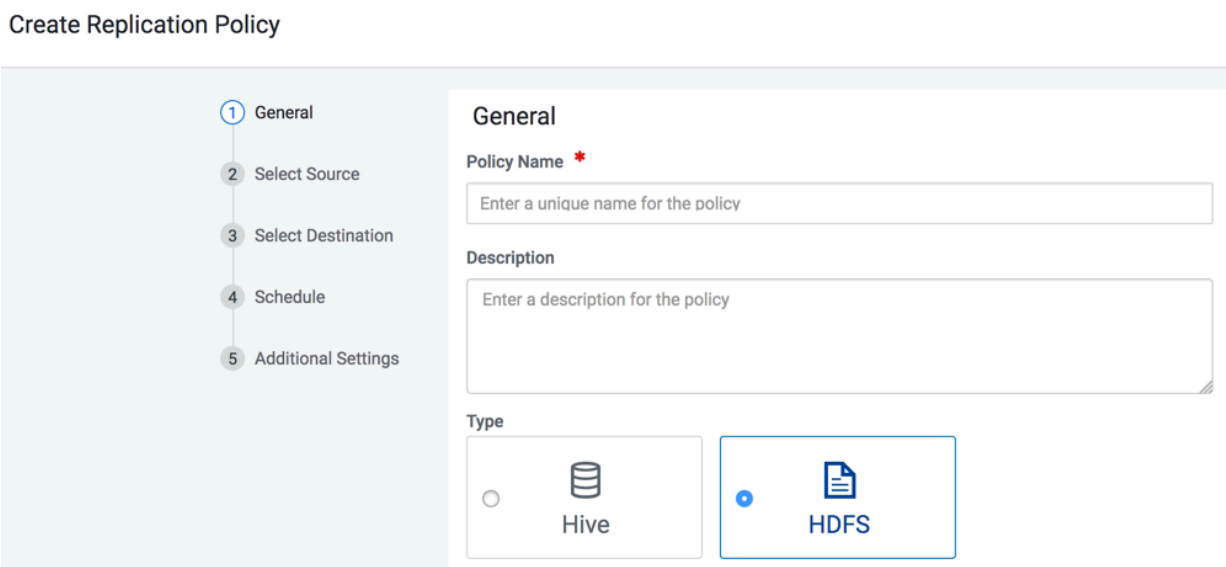

3. Enter the HDFS replication Policy Name and Description. Click Next.

4. Select Source Cluster.
5. Enter the Source Path where the source data resides.
6. In the Run As Username (on source) field, enter the source user.
7. Click Next.
8. Choose the destination Type as S3.
9. Select a Cloud Credential on Source. You can also add cloud credentials using the Add Cloud Credential link.
10. In the Path field, enter the values based on the Type you need and choose S3 type, provide a folder path in the bucket_name/path format.
11. Click Validate Policy.
    Cloudera Replication Manager verifies the data with a status Validate Policy Source and Destination information.
12. Click Next to schedule the replication policy.
13. On the Schedule page, choose one of the following options:
    - Run Now (Default) - The replication policy is immediately submitted and processed.
    - Schedule Run - The replication policy can be scheduled to run at specified time intervals.
14. (Optional) In the Repeat field, you can choose one of the following options:
    - Does Not Repeat

      Custom - In the Custom Recurrence dialog box, choose the time, date, and the frequency to run the policy.
15. Click Next.
16. On the Additional Settings page, enter or choose the values as necessary:

Option	Description
YARN Queue Name	Enter the name of the YARN queue for the cluster to which the replication job is submitted if you are using Capacity Scheduler queues to limit resource consumption. The default value for this field is default.
Maximum Maps Slots	Set the maximum number of map tasks (simultaneous copies) per replication job. The default value is 20.
Maximum Bandwidth	Adjust this setting so that each map task is throttled to consume only the specified bandwidth. Each map task ((simultaneous copy) is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value. You can adjust this setting so that each map task is throttled to consume only the specified bandwidth so that the net bandwidth used tends towards the specified value. The default value for the bandwidth is 100MB per second for each mapper.
Path Exclusion	Enter a regular expression-based path. When you add an exclusion, include the snapshotted relative path for the regex. For example, to exclude the /user/bdr directory, use the `.*/user/\.snapshot/.+/bdr.*` regular expression, which includes the snapshots for the `bdr` directory. To exclude top-level directories from replication in a globbed source path, you can specify the relative path for the regex without including .snapshot in the path. For example, to exclude the bdr directory from replication, use the `.*/user+/bdr.*` regular expression. You can add more than one regular expression to exclude.

Option	Description
Replication Strategy	Choose one of the following replication strategies to determine whether the file replication tasks should be distributed among the mappers statically or dynamically. • Static distributes file replication tasks among the mappers up front to achieve an uniform distribution based on the file sizes. • Dynamic distributes the file replication tasks in small sets to the mappers, and as each mapper completes its tasks, it dynamically acquires and processes the next unallocated set of tasks. The default replication strategy is Dynamic.
MapReduce Service	Choose the MapReduce or YARN service to use.
Log Path	Enter an alternate path for the logs, if required.
Error Handling	Select the following options as necessary: • Skip Checksum Checks - Determines whether to skip checksum checks on the copied files. If selected, checksums are not validated. Checksums are checked by default. **Note:** You must skip checksum checks to prevent replication failure due to non-matching checksums in the following cases: • Replications from an encrypted zone on the source cluster to an encrypted zone on a destination cluster. • Replications from an encryption zone on the source cluster to an unencrypted zone on the destination cluster. • Replications from an unencrypted zone on the source cluster to an encrypted zone on the destination cluster. Checksums are used for two purposes: • To skip replication of files that have already been copied. If Skip Checksum Checks is selected, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination. • To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data. • Skip Listing Checksum Checks - Whether to skip checksum check when comparing two files to determine whether they are same or not. If skipped, the file size and last modified time are used to determine if files are the same or not. Skipping the check improves performance during the mapper phase. Note that if you select the Skip Checksum Checks option, this check is also skipped. • Abort on Error - Whether to abort the job on an error. If selected, files copied up to that point remain on the destination, but no additional files are copied. Abort on Error is not selected by default. • Abort on Snapshot Diff Failures - If a snapshot diff fails during replication, the replication policy uses a complete copy to replicate data. If you select this option, the policy aborts the replication when it encounters an error instead.

Option	Description
Preserve	Choose the required options to preserve the block size, replication count, permissions (including ACLs), and extended attributes (XAttrs) as they exist on the source file system, or to use the settings as configured on the destination file system. By default source system settings are preserved. When Permission is selected, and both the source and destination clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. When Extended attributes is selected, and both the source and destination clusters support extended attributes, replication preserves them. (This option only displays when both source and destination clusters support extended attributes.) If you select one or more of the Preserve options and you are replicating to S3, the values all of these items are saved in metadata files on S3. When you replicate from S3 to HDFS, you can select which of these options you want to preserve. **Note:** To preserve permissions to HDFS, you must be running as a superuser on the destination cluster. Use the Run As Username option to set the username.
Delete Policy	Choose the required options to determine whether the files that were deleted on the source should also be deleted from the destination directory. This policy also determines the handling of files in the destination location that are unrelated to the source. Options include: • Keep Deleted Files - Retains the destination files even when they no longer exist at the source. This is the default option. • Delete to Trash - If the HDFS trash is enabled, files are moved to the trash folder. This is not supported when replicating to S3 • Delete Permanently - Uses the least amount of space; use with caution.
Alerts	Choose the required options to generate alerts for various state changes in the replication workflow. You can alert on failure, on start, on success, or when the replication workflow is aborted.

17. Click Create.

Once the replication policy runs successfully, you can view the replication job status on the Replication Policies page. Verify whether the job starts and runs as expected.

Verifying HDFS data migration

Post-migration process, you must check if the migration process is completed successfully by verifying the data on the AWS S3 endpoint.

There are two ways to verify the HDFS data replication:

1. Verification from Cloudera Replication Manager App.

   Once the replication policy is created and scheduled, go to Cloudera Public Cloud -> choose replication manager -> click Replication Policies on the left pane.

   You can view the number of replication policies that was created and the replication jobs status as seen from the following image:

   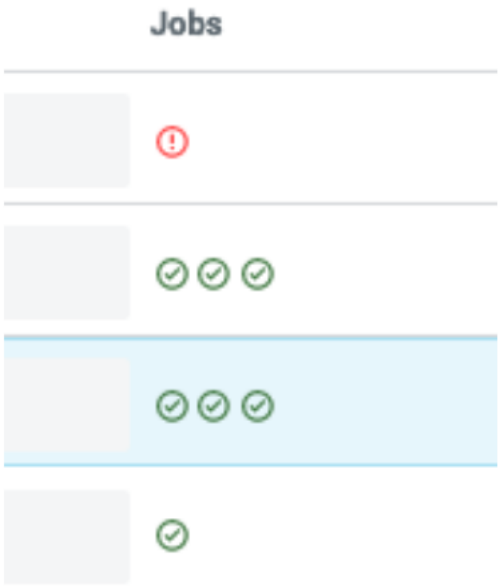

   The green check mark indicates successful replication jobs. The red escalation mark indicates the failed replication jobs.
2. Verifying from the AWS S3 Console.

   Log into AWS S3 > click Buckets on left pane -> search the bucket name in the search box of right pane.

   After accessing the target bucket, search the path name where the HDFS data is migrated. View if the HDFS data in source CDH cluster is replicated in the target S3 bucket path

Hive migration from CDH to Cloudera

To create a Hive replication policy from on-premises to the cloud account, you must register your cloud account credentials with the Replication Manager service, so that Cloudera Replication Manager can access your cloud storage.

Before performing Hive replication using CDH clusters, see Working with cloud credentials.

Creating a Hive replication policy

To replicate Hive metadata from on-premises to cloud, you must set the Ranger policy in Ranger, and then create the Hive replication policy in Replication Manager.

About this task

The Apache Ranger access policy model consists of the following components:

- Specification of the resources that you can apply to a replication policy which includes the HDFS files and directories; Hive databases, tables, and columns; and HBase tables, column-families, and columns.
- Specification of access conditions for specific users and groups.

Note: You must set the Ranger policy for the hdfs user on the target cluster to perform all operations on all databases and tables. The same user role is used to import the Hive Metastore. The hdfs user should have access to all Hive datasets, including all operations. Otherwise, the Hive import fails during the replication process.

On the target cluster, the hive user must have Ranger admin privileges. The same hive user performs the metadata import operation.

1. Log in to the Ranger Admin UI.
2. In the Hadoop_SQL section, provide the hdfs user permission to "all-database, table, column" in hdfs.

Note: You can replicate data from on-premises to cloud with a single cluster. The Metastore must be running on the cloud.

Procedure

1. On the Cloudera Management Console > Replication Manager > Replication Policies, click Add Policy.
2. In the Create Replication Policy wizard, select Hive.

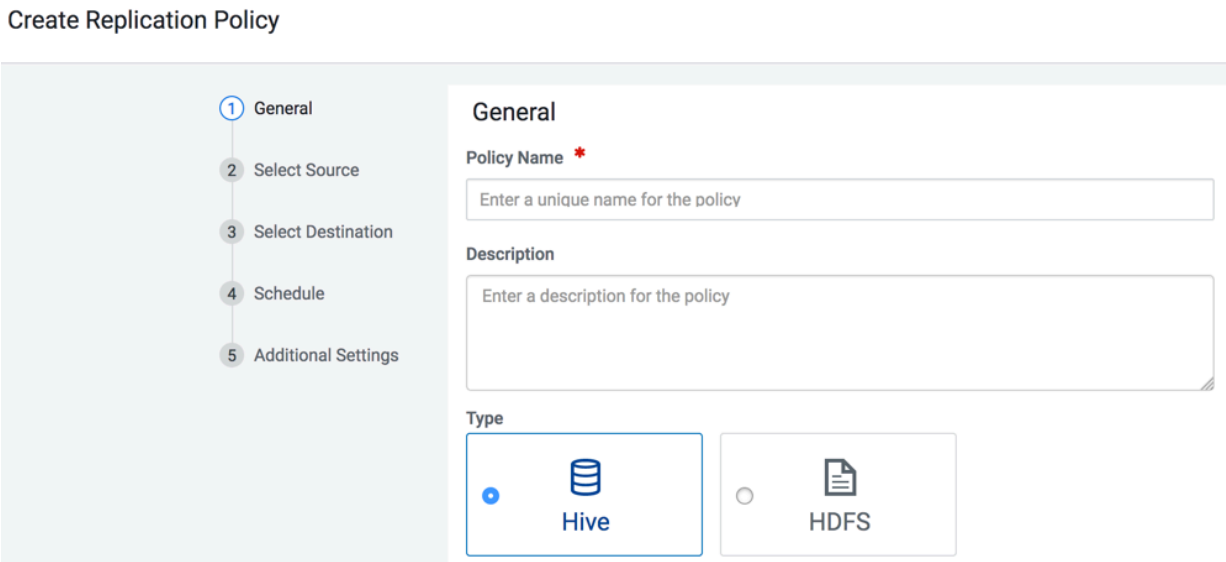

3. Enter the Hive replication Policy Name and Description. Click Next.
4. Select the Source Cluster.
5. Enter the database name and table name in Source Databases and Tables. Click the + icon to enter more databases and tables as necessary.
6. Enter the username in Source User. Ensure that the user has the necessary permissions to replicate data. The user should have Read access to the data.
7. Click Next.
8. Select the Destination Data Lake cluster. The Managed Warehouse Path and the Hive External Table Base Directory path for the Data Lake appears.
   a. Administrators can edit the Hive External Table Base Directory field to add another path to override the default storage location for replicated Hive external tables. Before you add another path to override the default storage location, ensure that the following steps are complete in the Ranger UI:
      1. Alter the ranger policy Default: Hive warehouse locations in cm_s3 service to allow the Hive service to access the updated locations of S3 bucket path.
      2. Manually update the Ranger and Sentry permissions.
9. Select a Cloud Credential on Source. You can also add cloud credentials using the Add Cloud Credentials link.
10. Enter the source cluster user name in Run as Username.
11. Click Validate Policy.

    Cloudera Replication Manager verifies the data with a status Validate Policy Source and Destination information.
12. Click Next to schedule the replication policy.

13. On the Schedule page, choose one of the following options:

- Run Now (Default) - The replication policy is immediately submitted and processed.
- Schedule Run - The replication policy can be scheduled to run at specified time interval.

14. In the Repeat field, you can choose one of the following options:

- Does Not Repeat
- Custom - In the Custom Recurrence dialog box, choose the time, date, and the frequency to run the policy.

15. Click Next.

16. On the Additional Settings page, enter the values as necessary:

Option	Description
YARN Queue Name	Enter the name of the YARN queue for the cluster to which the replication job is submitted if you are using Capacity Scheduler queues to limit resource consumption. The default value for this field is default.
Maximum Maps Slots	Set the maximum number of map tasks (simultaneous copies) per replication job. The default value is 20.
Maximum Bandwidth	Adjust this setting so that each map task is throttled to consume only the specified bandwidth. Each map task ((simultaneous copy) is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value. You can adjust this setting so that each map task is throttled to consume only the specified bandwidth so that the net bandwidth used tends towards the specified value. The default value for the bandwidth is 100MB per second for each mapper.
Path Exclusion	Enter a regular expression-based path. When you add an exclusion, include the snapshotted relative path for the regex. For example, to exclude the /user/bdr directory, use the `.*/user/\.snapshot/.+/bdr.*` regular expression, which includes the snapshots for the `bdr` directory. To exclude top-level directories from replication in a globbed source path, you can specify the relative path for the regex without including .snapshot in the path. For example, to exclude the bdr directory from replication, use the `.*/user+/bdr.*` regular expression. You can add more than one regular expression to exclude.
Replication Strategy	Choose one of the following replication strategies to determine whether the file replication tasks should be distributed among the mappers statically or dynamically. • Static distributes file replication tasks among the mappers up front to achieve an uniform distribution based on the file sizes. • Dynamic distributes the file replication tasks in small sets to the mappers, and as each mapper completes its tasks, it dynamically acquires and processes the next unallocated set of tasks. The default replication strategy is Dynamic.
MapReduce Service	Choose the MapReduce or YARN service to use.
Log Path	Enter an alternate path for the logs, if required.

Error Handling	Select the following options as necessary: • Skip Checksum Checks - Determines whether to skip checksum checks on the copied files. If selected, checksums are not validated. Checksums are checked by default. **Note:** You must skip checksum checks to prevent replication failure due to non-matching checksums in the following cases: • Replications from an encrypted zone on the source cluster to an encrypted zone on a destination cluster. • Replications from an encryption zone on the source cluster to an unencrypted zone on the destination cluster. • Replications from an unencrypted zone on the source cluster to an encrypted zone on the destination cluster. Checksums are used for two purposes: • To skip replication of files that have already been copied. If Skip Checksum Checks is selected, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination. • To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data. • Skip Listing Checksum Checks - Whether to skip checksum check when comparing two files to determine whether they are same or not. If skipped, the file size and last modified time are used to determine if files are the same or not. Skipping the check improves performance during the mapper phase. Note that if you select the Skip Checksum Checks option, this check is also skipped. • Abort on Error - Whether to abort the job on an error. If selected, files copied up to that point remain on the destination, but no additional files are copied. Abort on Error is not selected by default. • Abort on Snapshot Diff Failures - If a snapshot diff fails during replication, the replication policy uses a complete copy to replicate data. If you select this option, the policy aborts the replication when it encounters an error instead.
Preserve	Choose the required options to preserve the block size, replication count, permissions (including ACLs), and extended attributes (XAttrs) as they exist on the source file system, or to use the settings as configured on the destination file system. By default source system settings are preserved. When Permission is selected, and both the source and destination clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. When Extended attributes is selected, and both the source and destination clusters support extended attributes, replication preserves them. (This option only displays when both source and destination clusters support extended attributes.) If you select one or more of the Preserve options and you are replicating to S3 or ADLS, the values all of these items are saved in metadata files on S3 or ADLS. When you replicate from S3 or ADLS to HDFS, you can select which of these options you want to preserve. **Note:** To preserve permissions to HDFS, you must be running as a superuser on the destination cluster. Use the Run As Username option to set the username.

Delete Policy	Choose the required options to determine whether the files that were deleted on the source should also be deleted from the destination directory. This policy also determines the handling of files in the destination location that are unrelated to the source. Options include: • Keep Deleted Files - Retains the destination files even when they no longer exist at the source. This is the default option. • Delete to Trash - If the HDFS trash is enabled, files are moved to the trash folder. This is not supported when replicating to S3 or ADLS. • Delete Permanently - Uses the least amount of space; use with caution.
Alerts	Choose the required options to generate alerts for various state changes in the replication workflow. You can alert on failure, on start, on success, or when the replication workflow is aborted.
Sentry Permissions	Choose the following Sentry permissions as necessary: • Include Sentry Permissions with Metadata - Select this option to migrate Sentry permissions during the replication job. • Exclude Sentry Permissions from Metadata - Select this option if you do not want to migrate Sentry permissions during the replication job. • Skip URI Privileges - Select this option if you do not want to include URI privileges when you migrate Sentry permissions. During migration, the URI privileges are translated to point to an equivalent location in S3. If the resources have a different location in Amazon S3, do not migrate the URI privileges because the URI privileges might not be valid.
Replication Option	Specify metadata and data, or metadata only.
Directory for metadata file	The folder path in the destination cluster to save the metadata file. If the folder does not exist, Replication Manager creates a new folder.

17. Click Create.

Verifying Hive data migration

After the replication policy runs successfully, you can view the replication job status on the Replication Policies page.

An administrator can verify whether the job starts and runs as expected using the following steps:

- For HDFS data replication, check the cloud storage path (S3 bucket path) to verify whether the data was successfully copied in the specified bucket.
- For Hive metadata replication, verify whether the specified source database, along with tables, partitions, UDFs, and column stats are available in the Data Lake HMS instance. For this, the administrator can use a Cloudera Data Hub cluster and run the corresponding queries using Hue or beeline.
- For Ranger policies, query the Ranger policies to ensure that the Sentry policies are properly mapped as Ranger policies for the right users and groups.

Note: If the CDH source database contains functions, you must explicitly run the reload function command to view the migrated replication functions in the target location. You can perform and monitor various tasks after running a replication policy. You can view the job progress, notifications, and replication logs. You can use the advanced options to optimize a job run. You can suspend a job and also activate a suspended job. You can edit the replication policy as necessary.

Migrating Oozie workflows from CDH to Cloudera

Hue stoores the workflows within the Hue database which is created using Hue. The data residing in Hue is migrated to Cloudera.

About Migrating Oozie workloads

The Oozie database on the source CDH cluster is automatically configured to purge data that is older than 30 days. If you need to retain this data, Cloudera recommends that you back up the data and make it available for running simple SQL queries. See Back Up the Databases.

Oozie workflows, query history, and saved query are some of the workloads that get migrated to Cloudera.

Most Oozie workloads and configurations will need to be migrated manually to the destination cluster. Although you can copy the XML and property files from the source cluster, these files require extensive modifications before they can be used in the destination cluster. Cloudera Professional Services can assist you with these migrations.

Migration prerequisites

You need to know the prerequisites for migrating Hue databases from CDH to Cloudera.

You must meet the following general prerequisites before starting the migration process:

- AWS is your Cloud provider. Migrating to Azure / GCP is currently not supported.
- You can log in as the admin user to Cloudera Manager in the CDH cluster.
- CDH clusters must have been created using Cloudera Manager.

Setting up an external account

As CDH Administrator, you need to create an External Account on the CDH cluster that has an access key and secret key pair for accessing the cloud storage used in the Cloudera environment.

About this task

You must raise a support case with Cloudera to provide the key pair.

You must have valid Amazon S3 credentials to access cloud storage for migrating data. You add the access key / secret key pair to the account to access Cloudera cloud storage s3a://<data lake bucket path>. The access key / secret key pair ensures that the external account configured on CDH can successfully access the S3 buckets used by Data Lake.

Procedure

1. In Cloudera Manager#Administration, click Add Access Key Credentials.
2. In Edit Credential, name the credential and enter the AWS Access Key ID and AWS Secret Key.
3. Save changes,

Migrating Hue databases from CDH to Cloudera

You must migrate Hue data from the CDH cluster to Cloudera.

About this task

On your CDH environment enabled with Hue cluster migrate the Hue database:

Procedure

1. In the Hue Web UI, click the home icon to view the documents that you plan to migrate.
2. In Cloudera Manager, stop the Hue service. Go to Hue and select Actions > Stop.

   Note: Refresh the web page to confirm that the Hue service is stopped.

3. From Cloudera Manager-> Hue -> Select Actions > Dump Database and click Dump Database.

   This exports a JSON file that is database-agnostic. The file is written to

   /tmp/hue_database_dump.json

   on the host running the Hue service.
4. Log on to the host of the Hue server in a command-line terminal. You can find the hostname on the Dump Database window and at Hue > Hosts.

5. Remove the following lines from the hue_database_dump.json file.

   ALERT: This appears to be a CM Managed environment

   ALERT: HUE_CONF_DIR must be set when running hue commands in CM Managed environment

   ALERT: Please run 'hue <command> --cm-managed'

   On your Cloudera Data Platform environment:

   a. In Cloudera Manager for the Cloudera environment that is, Cloudera Data Hub Cloudera Manager (not Data Lake Cloudera Manager), set up the database for Hue.
   b. Synchronize: Select Actions > Synchronize Database and click Synchronize Database.
   c. Truncate all the records in the Hue tables available under the Hue database.

      Note: Must be performed directly on the Hue database; there are no Hue commands or options in the UI to perform this task.

   Additionally, on the Cloudera Data Platform environment: As no information is stored in the Hue database yet:

   a. Copy the hue_database_dump.json from the CDH cluster to the Cloudera cluster.
   b. Place the JSON file taken out from the legacy CDH Hue Cluster and place it in /tmp on the new Cloudera Hue server host.

      Note: The import process can only be run from one Hue server. Pick one to run the import process on and temporarily turn off the other Hue server instances.

      Attention: The file name should not be changed from hue_database_dump.json.You must load the Hue dump using the CLI method and must NOT use the UI option via Hue Server.

   c. Stop the Hue Service.
   d. Run the following commands:

      cd /opt/cloudera/parcels/CDH/lib/hue

      ./build/env/bin/hue loaddata --ignorenonexistent /tmp/<HUE_DATABASE_JSON_FILE> --cm-managed
   e. Start the Hue service.
   f. From Cloudera Manager -> Hue -> Select Actions -> Synchronize: Select Actions > Synchronize Database and click Synchronize Database.
   g. Start the Hue Service.

Performing post-migration tasks

You must complete the post-migration activities on your Cloudera cluster.

Note: The “userid” should NOT change while performing the post-migration task. If they are different, the documents do not populate for the user, as the documents are mapped with the specific userid.

Login to Hue using the same credentials used while using the source CDH Hue cluster.

The saved documents must be populated.

Attention: While carrying out the post-migration task, if the userid is different from the one that was used earlier, use the following method to resolve the differences.

Post Migration requirements

You must note the list of manual refactoring changes that should be applied to the Oozie jobs to get them working in the new environment.

- Name-node settings for each workflow
- Resource-manager settings for each workflow
- Paths, if the locations are being changed

- JDBC URLs, including Hive must point to the new database
- DFS paths may need to be replaced with S3
- New cluster-specific configurations for the Workflow (for example, Amazon S3 credentials.)
- Coordinator Jobs that have future sdf runs scheduled. Recreate these jobs to match what was included in any other cluster. For such Jobs, the Workflow and its configuration might be stored in Hue
- Copy any custom sharelib jars from the source to the destination cluster. The jars are stored here: /user/oozie/share/lib/lib_{TIMESTAMP}/{COMPONENT}
- If you have custom application jars used by any of your workflows, please make sure you recompile them with the 3rd-party libraries used by the latest Cloudera runtime.

For example on older versions of HDP, Oozie was using Guava 11, but in Cloudera 7 it is using Guava 28 or higher. If you compiled your application with Guava 11 you must also upgrade Guava.

- On the destination cluster, update the sharelib using the following command: oozie admin -oozie {URL} -share libupdate

Migrating HDFS and Hive data from CDH to Cloudera

An overview of the migration process from CDH to Cloudera prepares you to migrate HDFS and Hive data to the AWS S3 endpoint.

Once fulfilling migration prerequisites, you set up security for communications between the CDH on-premise cluster and the destination cloud services. You can use the Cloudera Replication Manager service to migrate HDFS and Hive data from CDH or HDP to Cloudera

Migrating HDFS native permissions to Cloudera

If you have HDFS native permissions in your CDH or HDP clusters, you learn how to convert the native permissions into Ranger policy format and import the policies to Cloudera. You can choose to ignore this migration process if you do not have any HDFS native native permissions.

Migrating HDFS native permissions from CDH or HDP to Cloudera involves a combination of Extract Convert Transform Import operations.

The HDFS native permissions are first extracted as a file in a specific format, which are then converted and transformed into Ranger AWS S3 policies using the Ranger policy migration utility. The transformed Ranger policies are finally imported into Cloudera using the authzmigrator tool.

Extracting HDFS native permissions

Learn how to use the HDFS Permissions Export utility to extract HDFS native POSIX permissions from a source cluster running CDH or HDP. The extracted HDFS permissions are then used to create Ranger S3 policies that can be used in Cloudera.

About this task

The HDFS Permissions Export utility exports the permissions to a .csv file in the following format with the permissions sorted in ascending order:

```
Syntax:
"/resource/path"|"username"|"groupname"|"permission"
```

where,

- "/resource/path" refers to resource entities
- "username" refers to user entries
- "groupname" refers to group entries
- "permission" refers to the HDFS permission entries

```
Example:
```

```
"/dir1"|"hdfs"|"supergroup, public"|"read, execute"
"/dir1/dir1"||"supergroup, public"|"read, execute"
"/dir1/dir1"|"hdfs"||"read, write, execute"
```

Before you begin

- The source and target migration clusters must be running the following supported versions:
  - Source cluster: HDP 2.6.5, HDP 3.1.5, CDH 5.16, CDH 6.3
  - Target cluster: CDP 7.2.15 or higher
- Secure Copy (SCP) the HDFS keytab file from the source cluster to your local system
- Run the klist command to view the list of Kerberos principals available in the keytab — ${RANGER_ADMIN_KEYTAB_PATH} and then run the kinit command to authenticate the user.

```
klist -kt ${RANGER_ADMIN_KEYTAB_PATH}

kinit -kt ${RANGER_ADMIN_KEYTAB_PATH} rangeradmin/_HOST@REALM
```

Procedure

1. Download the HDFSPermissionsExportTool-<version>.jar file from Cloudera Archive.
2. Run the following command to extract HDFS permissions from the source cluster.

```
hadoop jar ./<path>/HDFSPermissionsExportTool-<version>.jar hdfs_namenod
e_host:hdfs_port source_directories
```

   where,

   - <path> refers to the location of the downloaded .jar file.
   - hdfs_namenode_host:hdfs_port refers to the URI of the active NameNode and the HDFS NameNode port. For example, active-namenode.example.com:8020
   - source_directories refers to the root directory from where you want the permissions scan to begin. For example, "/" or "/dir1". You can also specify multiple directories — "/hbase/yarn"

```
hadoop jar ./target/HDFSPermissionsExportTool-<version>.jar active-namen
ode.example.com:8020 /user /ranger
        /hbase
```

   This creates a HDFS_Permissions_Export_<timestamp>.csv file that contains the required HDFS permissions.

   Important: Be sure to use the URI of the active NameNode. Using the passive NameNode URI or the NameNode HA URI results in a failure.

3. Use the scp command and copy HDFS_Permissions_Export_<timestamp>.csv from the source cluster to the migration node of the target cluster.

   Important: Since the .csv filename contains a colon (:), ensure that you provide the full path in the scp command to prevent the command from treating ":" as a separator.

What to do next

Convert the extracted HDFS native permissions into Ranger HDFS policies and then transform them into Ranger S3 policies.

Related Information

Converting HDFS native permissions into Ranger HDFS policies

Converting HDFS native permissions into Ranger HDFS policies

Learn how to convert the extracted HDFS native permissions into Ranger HDFS service policies, which can then be transformed into Ranger S3 policies.

Before you begin

- The source and target migration clusters must be running the following supported versions:
  - Source cluster: HDP 2.6.5, HDP 3.1.5, CDH 5.16, CDH 6.3
  - Target cluster: Cloudera Public Cloud 7.2.15 or higher
- You must have copied the extracted HDFS native permissions (.csv file) into the migration cluster node.
- You must have copied the Ranger policy migration utility, ranger-<version>-policymigration.tar.gz file to the migration cluster node.
  1. Log in to the migration cluster's node (or a Data Lake node of a cloud cluster) where the ranger-admin binaries are available. For example, /opt/cloudera/parcels/CDH/lib/ranger-admin
  2. Copy /opt/cloudera/parcels/CDH/lib/ranger-admin/policymigration/ranger-<version>-policymigration.tar.gz to some location of the migration cluster machine from where you want to run the Ranger policy migration utility.
- You must have read, write, and execute permissions on the directory where the Ranger policy migration utility is copied.
- You must have SUDO or ROOT access to perform the migration process.

Procedure

1. Log in to the migration cluster node where you have copied the ranger-<version>-policymigration.tar.gz file. Untar the file and navigate to the ranger-<version>-policymigration directory.
2. Update the “Transform” part of the env.sh file with the right set of values according to your environment.

   For more information, see Supported Input parameters for Transform operation.
3. Run the transform.sh script with the convert command to convert the extracted HDFS native permissions to Ranger HDFS policy format.

```
Syntax:
./transform.sh convert $native_hdfs_permissions_csv_file_path
```

```
Example:
./transform.sh convert /tmp/HDFS_Permissions_Export_<timestamp>.csv
```

Results

The command generates a HDFS_Permissions_Export_<timestamp>_convert.json file, which is then transformed into the required Ranger S3 policies. If you encounter any errors, see the logs/ranger-policymigration.log file.

What to do next

Transform the converted Ranger HDFS policies into Ranger S3 policies.

Related Information

Supported Input parameters for Transform operation

Extracting HDFS native permissions

Transforming Ranger HDFS policies into Ranger S3 policies

Transforming Ranger HDFS policies into Ranger S3 policies

Learn how to transform Ranger HDFS service policies into Ranger AWS S3 policies, which can then be imported into the Cloudera cluster Ranger.

Before you begin

- You must have converted the extracted Ranger HDFS native permissions (.csv file) into Ranger HDFS policies (.json file) using the transform.sh convert command.
- You must have copied the Ranger policy migration utility, ranger-<version>-policymigration.tar.gz file to the migration cluster node.
  1. Log in to the migration cluster's node (or a Data Lake node of a cloud cluster) where the ranger-admin binaries are available. For example, /opt/cloudera/parcels/CDH/lib/ranger-admin
  2. Copy /opt/cloudera/parcels/CDH/lib/ranger-admin/policymigration/ranger-<version>-policymigration.tar.gz to some location of the migration cluster machine from where you want to run the Ranger policy migration utility.
- You must have read, write, and execute permissions on the directory where the Ranger policy migration utility is copied.
- You must have SUDO or ROOT access to perform the migration process.

Procedure

1. Log in to the migration cluster node where you have copied the ranger-<version>-policymigration.tar.gz file. Untar the file and navigate to ranger-<version>-policymigration directory.
2. Update the "Transform" part of the env.sh file by updating the following parameters or as required for your environment.
   - S3_BUCKET_NAME
   - SERVICE_NAME_FOR_NATIVE_POLICIES

   Note: If the usernames in the target cluster are different from the usernames in the source cluster, provide a user mapping file as an input by updating the RANGER_POLICYMIGRATION_USERS_MAPPING _FILE parameter in the env.sh file.

   For more information, see Supported Input parameters for Transform operation.
3. Run the transform.sh script with the transform command to convert the Ranger HDFS policies to Ranger AWS S3 policy format.

```
Syntax:
./transform.sh transform $converted_ranger_policies_file_path
```

```
Example:
./transform.sh transform /tmp/HDFS_Permissions_Export_<timestamp>_conve
rt.json
```

Results

The command generates a HDFS_Permissions_Export_<timestamp>_convert_transform.json file, which can then be imported using the Authzmigrator utility into the AWS S3 service of the target Cloudera cluster. If you encounter any errors, see the logs/ranger-policymigration.log file.

What to do next

Import the transformed HDFS native permissions into the Ranger AWS S3 service of the target Cloudera cluster using the Authzmigrator tool.

Related Information

Supported Input parameters for Transform operation

Extracting HDFS native permissions

Converting HDFS native permissions into Ranger HDFS policies

Importing Ranger AWS S3 policies

Learn how to import the transformed HDFS native permissions into the Ranger AWS S3 service of the target Cloudera cluster using the Authzmigrator tool.

Before you begin

- You must have extracted, converted, and transformed the native HDFS permissions into Ranger AWS S3 policy format.
- Ensure that the RANGER_ADMIN role is up and running in the Cloudera Manager/Cloudera portal of the target environment.

Procedure

1. Log in to the RANGER_ADMIN portal with administrator user credentials or user privileges as ADMIN.
   a) From Settings User/Groups/Roles Search , search for the "ranger" user.
   b) Click the "ranger" user and change its role to "Admin", and save the changes.
2. Confirm authz-ingest.zip is available under '/opt/cloudera/cm/lib/dr/' path. If authz-ingest.zip is not available, contact Cloudera customer support to provide it. Once you have authz-ingest.zip, copy to the /opt directory of the migration cluster node (or a Data Lake node of the cloud cluster).
3. Extract the authz-ingest.zip file into the /opt directory.

   Note: /opt is referred to as an example directory.

   The authz-ingest file contains the following directories:

   - config-files
   - scripts
4. If Ranger is TLS/SSL enabled using Cloudera Manager AUTO-TLS, then copy the cm-auto-global-truststore.jks file from the Ranger admin process directory of the Ranger admin host to /opt/authz-ingest/config-files/ of the migration tool host. Perform these steps to find the location of the Ranger admin process directory:
   a) Run the following command to get to the Ranger admin process:

      ps -ef | grep proc_rangeradmin
   b) Search for the -cp option from the output of the above command and look for "/var/run/cloudera-scm-agent/process/<ID>-ranger-RANGER_ADMIN"

   If you have set up SSL manually or using a different method, then copy the file that you have created for trust stores to the /opt/authz-ingest/config-files directory.

   For example, cp /path/to/cm-auto-global_truststore.jks /opt/authz-ingest/config-files/
5. Export the following environment variables to the target cluster Ranger:

   - JAVA_HOME - Java home path used on the cluster
   - RANGER_ADMIN_URL - URL of the Ranger Admin portal
   - RANGER_ADMIN_SSL_ENABLED - Indicates if SSL is enabled or disabled
   - RANGER_ADMIN_TRUSTORE_PASSWORD - If Ranger is TLS/SSL enabled using Cloudera Manager AUTO-TLS, then the password can be found in the Ranger host file, /etc/hadoop/conf/ssl-client.xml, property — 'ssl.client.truststore.password'

```
export JAVA_HOME=/usr/java/jdk1.8.0_232-cloudera/
export RANGER_ADMIN_URL=<ranger admin url with port>
export RANGER_ADMIN_SSL_ENABLED=<true or false>
export RANGER_ADMIN_TRUSTORE_PASSWORD=<ssl.client.truststore.password>
```

6. Go to /opt/authz-ingest/config-files and open the authorization-migration-site.xml file.

   The authorization-migration-site.xml contains the following properties:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>authorization.migration.export.target_services</name>
    <value>HIVE,KAFKA</value>
  </property>
  <property>
    <name>authorization.migration.export.migration_objects</name>
    <value />
  </property>
  <property>
    <name>authorization.migration.export.output_file</name>
    <value>hdfs:///user/sentry/export-permissions/permissions.json</value>
  </property>
  <property>
    <name>authorization.migration.ingest.is_dry_run</name>
    <value>false</value>
  </property>
  <property>
    <name>authorization.migration.role.permissions</name>
    <value>true</value>
  </property>
  <property>
    <name>authorization.migration.translate.url.privileges</name>
    <value>false</value>
  </property>
  <property>
    <name>authorization.migration.ingest.merge.ifexists</name>
    <value>true</value>
  </property>
  <property>
    <name>authorization.migration.migrate.url.privileges</name>
    <value>true</value>
  </property>
</configuration>
```

7. Search for the authorization.migration.ranger.import.input_file property in the authorization-migration-site.xml file and provide the location of the transformed HDFS permissions file as the property value.

```
<property>
 <name>authorization.migration.ranger.import.input_file</name>
<value>file:///tmp/HDFS_Permissions_Export_<timestamp>_convert_transform.j
son</value>
</property>
```

   If you are unable to find this property, then add the property in the file.

8. For Kerberos-based authentication, perform the following steps:

   Important: If the Key Distribution Center (KDC) server is not installed in your environment or imported without the Kerberos authentication, you can skip this procedure and use the process provided for non-Kerberos authentication. Also, if PAM authentication is enabled in the target cluster, then Kerberos authentication is recommended to import ranger policies quickly.

   a) Find the Ranger admin keytab location and provide it in the following command:
      export RANGER_ADMIN_KEYTAB_PATH=/path/to/ranger.keytab

      In the CDEP cluster, ranger keytab is available in the ranger host location /cdep/keytabs/rangeradmin.keytab or under the ranger admin process directory — /var/run/cloudera-scm-agent/process/<ranger-process-ID>-ranger-RANGER_ADMIN
   b) If your ranger node and migration node are different, copy the ranger keytab to the migration node.
   c) Run the klist command on the migration node to view the list of Kerberos principals available in the keytab — ${RANGER_ADMIN_KEYTAB_PATH}
      klist -kt ${RANGER_ADMIN_KEYTAB_PATH}
   d) Run the following kinit command and update _HOST and REALM correctly.
      kinit -kt ${RANGER_ADMIN_KEYTAB_PATH} rangeradmin/_HOST@REALM

9. For non-Kerberos authentication, add the following properties in the /opt/authz-ingest/config-files/authorization-migration-site.xml file with the appropriate values, and then save and close the file.

```
<property>
    <name>ranger.admin.username</name>
    <value>admin</value>
</property>

<property>
    <name>ranger.admin.passwd</name>
    <value>admin123</value>
</property>
<property>
<name>authorization.migration.kerberos.authentication</name>
<value>false</value>
</property>
```

10. Go to the /opt/authz-ingest/scripts directory and run the import script.
    sh authz-import.sh

    If the import operation fails, look for details in the output of the command printed in the terminal window or see the target Ranger admin log files.

Results

After the import operation is complete, the authzmigrator tool displays the total policies count, created, skipped, and failed policies count. You can refer to the console output logs to verify if policies are imported or not.

```
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: ========================SUMMARY==================================
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: Ingesting ranger policies finished successfully
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: Total Policies=10000, Created=10000, Skipped=0, Failed=0, Timedout=0
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: ================================================================
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: Ingesting of Ranger Policies to ranger service is complete
```

Related Information

Extracting HDFS native permissions

Converting HDFS native permissions into Ranger HDFS policies

Transforming Ranger HDFS policies into Ranger S3 policies

Migrating workflows directly created in Oozie to Cloudera

The oozie workflows present on HDFS must be migrated to Cloudera.

Oozie workflows created manually or outside of the Hue Workflow editor require specific review and manual updates. No automated tool or a process is required for updating manually created workflows and property files.

The Oozie workflows are available in HDFS within CDH and HDP clusters. To migrate the Oozie workflows from these clusters to Cloudera SaaS involves a separate workflow process.

While using the HDP clusters, you can use the DistCp tool to migrate the Oozie workflows present in HDFS. While using the CDH clusters, you can employ the Replication Manager App to migrate the Oozie workflows present in HDFS

Specifically, some manual updates are required to process the Oozie migration created outside of the Hue Workflow. Before you proceed further, you must understand:

- During the migration process you must copy across all Oozie job files (workflow.xml, job.properties, and any supporting JARs).
- Which Oozie workflow files must be copied or migrated to your cluster. Identify the workflow.xml file and job.properties file for each Oozie workload that must be migrated. These files are stored in HDFS and must be copied to your cluster.
- The job.properties file must be updated with the appropriate cluster endpoints.
- Optionally, the workflow.xml file needs to be updated. For example, while currently using the legacy “hive action” requires an update to the newer “hive2 action”.

Depending on where (the location) you have stored your Oozie workflow data in HDFS, note the following information:

The workflow.xml and any job JAR files reside within HDFS in the source cluster. These will have to be copied across into the cluster you are migrating to.

The job.properties file for a job contains a reference to the location of where the workflow files are stored. The job .properties file will need to be updated during a migration with the new target environment settings / locations.

Note: The job.properties file must be run from "local" and not HDFS, but some customers may have these files also stored on HDFS in the source cluster.

Attention: Using the Oozie Hive Action with HiveServer2 is deprecated. If using the legacy HivAction in your Oozie workflow, you must modify your workflow.xml to use the Hive 2 Action instead.

- If required, the refactoring of the component level code. For example, Oozie job executing Spark 1.6 code requires an update to the newer Spark 2.4 version. For more information, see Migrating Spark workloads to Cloudera.

Related Information

Migrating HDFS and Hive data from CDH to Cloudera

Migrating HDFS data from HDP to Cloudera

Migrating Sentry policies from CDH to Cloudera

In the CDH environment consisting of Sentry policies, permissions need to be migrated from Sentry to Ranger. This migration process is supported by the Authzmigrator tool.

About Migrating Sentry policies

Migrating Sentry policies to Cloudera is a two step process.

Export the Sentry policies from the CDH cluster and copy the generated output file(s) to the target environment. Later, the import utility (Authzmigrator tool) performs the Transform and Import operations to complete the Sentry migration process. Export operation is performed using the CDH cluster.

The following policy migrations are included as part of the migration process:

- Kafka and Hive Sentry permissions

- URL transformation to the AWS S3 endpoint
- Default AWS S3 policy creation for Hive Warehouse location

Migration prerequisites

You need to know the prerequisites for migrating Sentry policies from CDH to Cloudera.

You must meet the following general prerequisites before starting the migration process:

- AWS is your Cloud provider. Migrating to Azure / GCP is currently not supported.
- The source and target migration clusters must be running the following supported versions:
  - Source cluster: CDH 5.16, CDH 6.3
  - Target cluster: Cloudera Public Cloud 7.2.15 or higher
- You can log in as the admin user to Cloudera Manager in the CDH cluster.
- You must have SUDO or ROOT access to perform the migration process.
- CDH clusters must have been created using Cloudera Manager. Clusters that are not managed by Cloudera Manager cannot be registered to CDP.
- Create an external account on the CDH cluster that has non-expiring access credentials and secret key pair.
- Your source cluster supports compatible versions of Cloudera Manager and Cloudera Runtime shown in the following table:

Source cluster	Earliest supported Cloudera Manager	Earliest supported Cloudera Runtime
CDH 5	6.3.0	5.1.0
CDH 6	6.3.0	6.1.0
CDH 6	7.3.1	6.3.3

Specifically note that:

- Your CDH cluster is registered on the Cloudera Management Console. For more information, see Adding a CDH cluster.
- Open communication ports between the source CDH cluster and Cloudera list.
- When you create databases, you must assign valid roles for tables and grant adequate permissions as well. And the same applies to Kafka.

Setting up an external account

As CDH Administrator, you need to create an External Account on the CDH cluster that has an access key and secret key pair for accessing the cloud storage used in the Cloudera environment.

About this task

Using the external account along with the cloud storage path, administrators can migrate the Sentry policies from the CDH cluster to the cloud storage used in the Cloudera environment.You must have valid Amazon S3 credentials to access cloud storage for migrating data.

You must raise a support case with Cloudera to provide the key pair.

You must have valid Amazon S3 credentials to access cloud storage for migrating data. You add the access key / secret key pair to the account to access Cloudera cloud storage s3a://<data lake bucket path>. The access key / secret key pair ensures that the external account configured on CDH can successfully access the S3 buckets used by Data Lake.

Procedure

1. In Cloudera Manager#Administration, click Add Access Key Credentials.
2. In Edit Credential, name the credential and enter the AWS Access Key ID and AWS Secret Key.
3. Save changes,

Exporting Sentry permissions

You must export the Sentry policies from CDH cluster to Cloudera cluster Ranger.

Procedure

1. SSH to the Sentry host of the source cluster.
2. Copy the file located here to the Sentry host and extract it to a suitable location.

   tar -xvf authz_export.tar.gz

   Later, a directory named authzmigrator is created which contains the following files:

   - Jars
   - Config
   - Authz_export.sh

```
[root@source-cluster-1 ~]# ll authzmigrator/
total 12
-rwxr--r-- 1 502 wheel  587 Jan  8  2021 authz_export.sh
drwxr-xr-x 2 502 wheel 4096 Mar 16 11:22 config
drwxr-xr-x 2 502 wheel 4096 Mar 16 09:00 jars
```

3. Verify the SENTRY_SERVER process path

   ps -ef | grep SENTRY_SERVER
4. Replace sentry-site.xml and core-site.xml in the extracted file with config files from Sentry-run directory. For example,

   - cp /var/run/cloudera-scm-agent/process/<process-id>-sentry-SENTRY_SERVER/sentry-site.xml authzmigrator/config/
   - cp /var/run/cloudera-scm-agent/process/<process-id>-sentry-SENTRY_SERVER/core-site.xml authzmigrator/config/
5. Update ‘Sentry.store.jdbc.user’ and sentry.store.jdbc.password in sentry-site.xml.

   The mandatory values are the Sentry "database user" and "database user password" in clear text.
6. Remove the property hadoop.security.credential.provider.path in sentry-site.xml.
7. Update the value for the property fs.defaultFS to file:/// in core-site.xml.
8. Make sure that the following configurations are updated in

   authorization-migration-site.xml present in the authzmigrator/config/

   - • authorization.migration.export.target_services= HIVE,KAFKA should have a list of services for which permissions are needed to export the Sentry permissions. Valid values: HIVE, KAFKA

- authorization.migration.export.output_file=<path> should be updated to the absolute location of the file where permissions should be exported

To set up role based permissions, you must add the following properties:

<property>

<name>authorization.migration.role.permissions</name>

<value>true</value>

</property>

An example property file:

```
~ — root@source-cluster-1:~ — ssh source-cluster-1.source-cluster.root.hwx.site
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>authorization.migration.export.target_services</name>
    <value>HIVE,KAFKA</value>
  </property>
  <property>
    <name>authorization.migration.export.migration_objects</name>
    <value />
  </property>
  <property>
    <name>authorization.migration.export.cluster_name</name>
    <value>cluster1</value>
  </property>
  <property>
    <name>authorization.migration.export.output_file</name>
    <value>/root/permissions.json</value>
  </property>
  <property>
    <name>authorization.migration.ingest.is_dry_run</name>
    <value>false</value>
  </property>
  <property>
    <name>authorization.migration.kerberos.authentication</name>
    <value>false</value>
  </property>
  <property>
    <name>authorization.migration.role.permissions</name>
    <value>true</value>
  </property>

</configuration>
```

9. Export JAVA_HOME variable

10. Run the script:

    cd authzmigrator/

    sh authz_export.sh

    Exporting the permissions is completed.

11. SCP the exported JSON to the target cluster. For more information about logging into Cloudera clusters, see Using SSH to access the cluster.

    scp <exported json> root@<target_clustert>:/root

Related Information

Importing Sentry permissions into Ranger

Importing Sentry permissions into Ranger

You must import the Sentry permissions to Cloudera cluster Ranger.

Procedure

1. Login to the RANGER_ADMIN portal with administrator user credentials or user privileges as ADMIN
   a. Under Settings > Users/Groups/Roles > Search for the ranger user.
   b. Click on the ranger user and change its role to Admin and save the changes.
2. Confirm authz-ingest.zip is available under '/opt/cloudera/cm/lib/dr/' path. If authz-ingest.zip is not available, contact Cloudera customer support to provide it. Once you have authz-ingest.zip, copy to the /opt directory of the migration cluster node (or a Data Lake node of the cloud cluster).
3. Unzip the authz-ingest.zip into the /opt directory, which contains the following files:

   Note: /opt is referred to as an example directory.

   - config-files
   - scripts

   The config-files contains the following:

   authorization-migration-site.xml

   ranger-hive-security.xml

   ranger-kafka-security.xml

   ranger-plugin-policymgr-ssl.xml

   Note: cm-auto-global_truststore.jks is directly not available. If you have an SSL/TLS enabled cluster, you must manually add cm-auto-global_truststore.jks.

4. Ranger is TLS/SSL enabled using Cloudera Manager AUTO-TLS. From the Ranger admin host, copy cm-auto-global_truststore.jks (from the Ranger-admin process directory) to the following location:

   /opt/authz-ingest/config-files/ of the migration tool host.

   Steps to find the ranger admin process directory:

   - • Execute the below command to get the RANGER_ADMIN process.

     ps -ef | grep proc_rangeradmin

     • Search for -cp option from the above command output and look for "/var/run/cloudera-scm-agent/process/<ID>-ranger-RANGER_ADMIN"

     If you have done SSL setup manually or in a different method, copy the filename which you created for truststores.

     For example: cp /path/to/cm-auto-global_truststore.jks /opt/authz-ingest/config-files/

5. Export the following variables:
   - JAVA_HOME - Java home path used on the cluster
   - RANGER_ADMIN_URL - Ranger Admin portal url
   - RANGER_ADMIN_SSL_ENABLED - either true or false
   - RANGER_ADMIN_TRUSTORE_PASSWORD - If Ranger is TLS/SSL enabled using Cloudera Manager AUTO-TLS, then the password can be found on the Ranger host /etc/hadoop/conf/ssl-client.xml property 'ssl.client.truststore.password'

```
<configuration>
  <property>
    <name>ssl.client.truststore.location</name>
    <value>/var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks</value>
  </property>
  <property>
    <name>ssl.client.truststore.password</name>
    <value>fk0GqCp0FSVfNeM9znEdEDMbcqeC6xUnSlM4rjIUif4</value>
  </property>
  <property>
    <name>ssl.client.truststore.type</name>
    <value>jks</value>
  </property>
  <property>
    <name>ssl.client.truststore.reload.interval</name>
    <value>10000</value>
  </property>
</configuration>
```

EXAMPLE:

export JAVA_HOME=/usr/java/jdk1.8.0_232-cloudera/

export RANGER_ADMIN_URL=<ranger admin url with port>

export RANGER_ADMIN_SSL_ENABLED=<true or false>

export RANGER_ADMIN_TRUSTORE_PASSWORD=<ssl.client.truststore.password>

6. For the Kerberos-based authentication, use the following process:
   a. Find the Ranger admin keytab location and provide it in the following command:

      export RANGER_ADMIN_KEYTAB_PATH=/path/to/rangeradmin.keytab

      In the CDEP cluster ranger keytab shall be available in the “/cdep/keytabs/rangeradmin.keytab” or under the ranger admin process directory “/var/run/cloudera-scm-agent/process/<ranger-process-number>-ranger-RANGER_ADMIN” location.

      Steps to find the ranger admin process directory:

      - Execute the below command to get the RANGER_ADMIN process.

      ps -ef | grep proc_rangeradmin

      - Search for -cp option from the above command output and look for "/var/run/cloudera-scm-agent/process/<ID>-ranger-RANGER_ADMIN"

      If your ranger node and migration node are different, copy the ranger keytab to the migration node.

      On the migration node run the following klist command to fetch a list of principals available in

      ${RANGER_ADMIN_KEYTAB_PATH}

      klist -kt ${RANGER_ADMIN_KEYTAB_PATH}

      Run the following kinitcommand using

      ${RANGER_ADMIN_KEYTAB_PATH} with rangeradmin/_HOST@REALM principle.

      Update _HOST and REALM using the following command:

      kinit -kt ${RANGER_ADMIN_KEYTAB_PATH} rangeradmin/_HOST@REALM
7. Go to location /opt/authz-ingest/config-files and open the file authorization-migration-site.xml
8. Add below property and provide the path of the sentry permissions.json file which needs to be imported.

```
<property>
    <name>authorization.migration.export.output_file</name>
<value>file:///root/permissions.json</value>
</property>
```

   Note: If you have permission which is role-based, you have to set this property

```
<property>
    <name>authorization.migration.role.permissions</name>
<value>true</value>
</property>
```

9. Set the following properties to create AWS S3 policy for Hive Warehouse location:

```
<property>
    <name>authorization.migration.create.s3.policy</name>
<value>true</value>
</property>
<property>
    <name>authorization.migration.target.s3.bucket.name</name>
<value>bucket_name</value>
</property>
```

10. Set the following properties which ensures that URI/URL policies are appropriately converted into AWS S3 URL Hive policy:

    <property>

    <name>authorization.migration.translate.url.privileges</name>

    <value>true</value>

    </property>

    <property>

    <name>authorization.migration.migrate.url.privileges</name>

    <value>true</value>

    </property>

    <property>

    <name>authorization.migration.destination.location.prefix</name>

    <value>s3a://mybucket</value>

    </property>

11. (Optional) Transforming old user name into new user name requires you to set the following properties:

    <property>

    <name>authorization.migration.translate.user</name>

    <value>true</value>

    </property>

    <property>

    <name>authorization.migration.usermap.file.path</name>

    <value>file:///root/UserMapping.json</value>

    </property>

    Use the following information to provide the valid user mapping:

    Format:

    => For the JSON file format we can have multiple users also in the JSON object formatted like :

    {“testuser1”:”testuser1@host.com”, “testuser2”:”testuser2@host.com”}

    => And with the CSV format, it's pipe-delimited with one entry per line.

    testuser1,testuser1@host.com

    testuser2,testuser2@host.com

    Note: If user mapping is provided in CSV format, set the following property

    <property>

    <name>authorization.migration.usermap.separator</name>

    <value>,</value>

    </property>

12. A flag-based new functionality is added in Authzmigrator, which skips {OWNER} policy while migrating from Sentry to Ranger. You must add the below property in authorization-migration-site.xml to enable this feature. By default, this feature is disabled by setting the below value to false.

```
<property>
```

```
    <name>authorization.migration.skip.owner.policy</name>
    <value>true</value>
    </property>
```

13. Save the file and exit
14. Run the script:

    cd authz-ingest/scripts

    sh authz-import.sh

    A confirmation message appears.

    "Ingesting Ranger policies finished successfully"

    "Ingesting of permissions to Ranger service is complete"

Migrating data from HDP to Cloudera

How to migrate data from HDP to Cloudera.

Migrating HDFS data from HDP to Cloudera

An overview of the migration process from HDP to Cloudera prepares you to migrate HDFS data to the AWS S3 endpoint.

The HDP to Cloudera migration migrates HDFS data using the Cloudera Replication Manager service. Once fulfilling migration prerequisites, you set up security for communications between the HDP cluster and the destination cloud services.

Migration prerequisites

You need to know the prerequisites for migrating HDFS data from HDP to Cloudera.

You must meet the following general prerequisites before starting the migration process:

- You can log in as the “hdfs” user to Ambari in the HDP cluster.
- Set up line-of-sight from the HDP cluster to the AWS endpoint.
- Ensure adequate network connectivity between the HDP cluster and AWS S3 endpoint.
- You can add cloud credentials and configure them in one of the two ways:
  - Add them into the core-site.xml configuration.
  - Provide as a runtime parameter when running the job (that job can be DistCp, Spark job and others).
- Make sure the HDFS service ports are configured.
- As the migration of HDFS data to Cloudera SaaS is directly to S3 via a push mechanism, the key network requirement is to ensure that the YARN / HDFS nodes have network access to the AWS S3 endpoints (port 80 or port 443 for TLS).
- Must allocate compute resources to execute YARN (DistCp) jobs to transfer the HDFS data to the S3 bucket.
- Any user running the DistCp job should not be a banned user and must be a part of a supergroup on each host in the cluster.
- Since there is no superuser on each host, the administrator needs to unban ‘hdfs’ user in the HDP cluster. For more information, see Unbanning hdfs user in HDP cluster on page 40.

Once fulfilling migration prerequisites, you set up security for communications between the HDP cluster and the destination cloud services.

Hadoop DistCp tool is used to migrate the HDFS data (and metadata) from an on-prem HDP cluster to the cloud storage (AWS S3) used in the Cloudera environment.

About DistCp tool

DistCp (distributed copy) is a tool used for large inter/intra-cluster copying.

DistCp uses MapReduce to enable its distribution, error handling and recovery, and reporting. It expands a list of files and directories into input to map tasks, each of which will copy a partition of the files specified in the source list.

Using the DistCp tool

Use DistCp to copy files between various clusters.

The most common use of DistCp is an inter-cluster copy:

hadoop distcp hdfs://nn1:8020/source hdfs://nn2:8020/destination

Where hdfs://nn1:8020/source is the data source, and hdfs://nn2:8020/ destination is the destination.

This will expand the name space under /source on NameNode "nn1" into a temporary file, partition its contents among a set of map tasks, and start copying from "nn1" to "nn2". Note that DistCp requires absolute paths.

You can also specify multiple source directories:

```
hadoop distcp hdfs://nn1:8020/source/a hdfs://nn1:8020/source/b hdfs://
          nn2:8020/destination
```

Or specify multiple source directories from a file with the -f option:

```
hadoop distcp -f hdfs://nn1:8020/srclist hdfs://nn2:8020/destination
```

Where srclist contains:

hdfs://nn1:8020/source/a

hdfs://nn1:8020/source/b

Note: The DistCp tool is also used to migrate data to the cloud storage (AWS S3, Microsoft Azure, and Google Cloud).

Unbanning hdfs user in HDP cluster

The hdfs user must be unbanned and allowed full access to the HDP cluster.

About this task

Procedure to unban the hdfs user.

Procedure

1. Remove the hdfs entry from the following file, from line banned.users on the ambari-server host.

   ```
   /var/lib/ambari-server/resources/common-services/YARN/2.1.0.2.0/package/
   templates/container-executor.cfg.j2
   ```

   The updated file contents are as follows

   yarn.nodemanager.local-dirs={{nm_local_dirs}}

   yarn.nodemanager.log-dirs={{nm_log_dirs}}

   yarn.nodemanager.linux-container-executor.group={{yarn_executor_container_group}}

   banned.users=yarn,mapred,bin

   min.user.id={{min_user_id}}

2. Check the YARN service configuration on Ambari UI for ‘container-executor configuration template’ configuration. Ensure that the hdfs user is not mentioned in the banned.users on the ambari-server host. If it is still present, remove it from the configuration, save the changes and restart stale services.
3. Restart ambari-server. On the ambari-server host, execute the following command:

   service ambari-server restart
4. Restart ambari-agent. On each ambari-agent host of the cluster, execute the following command:

   service ambari-agent restart
5. Add hdfs user in YARN ACLs configuration by editing

   yarn.admin.acl
6. Allow hdfs user to submit YARN jobs by editing the following configuration

   yarn.scheduler.capacity.root.acl_submit_applications
7. Restart the YARN service in the Ambari UI.

Before migrating

Before you commence to migrate HDFS data from HDP clusters, you must note about some of the underlying requirements.

To run the Hadoop DistCp tool to copy the HDFS data from the source HDP cluster to the Cloudera S3 bucket, the following additional details must be met:

- The access key, secret key pair of the specified S3 bucket must be available.
  - Before you run the DistCp tool on your on-premises cluster, you must have valid cloud credentials to access and use AWS. Create a Support JIRA that states the time period for the migration.
  - The AWS access secret key will be disabled when the migration time expires.
- SSM-KMS is enabled on the S3 bucket, the KMS key is also required.
- Perform the appropriate kinit to the Hadoop DistCp tool for authentication by the Kerberos system.
  - For hdfs user, use the kinit command with hdfs user’s keytab file. (Look for keytab in /etc/security/keytabs)

HDFS data migration from HDP to Cloudera

Use the DistCp tool to migrate the data.

An example DistCp command:

```
$ sudo -u hdfs hadoop distcp -Dfs.s3a.access.key=###########
          -Dfs.s3a.secret.key=######### -Dfs.s3a.server-side-encryption-alg
orithm=SSE-KMS
          -Dfs.s3a.server-side-encryption.key=arn:aws:kms:<region>:######
##### /data/datafiles/
          s3a://demo2-cdp-private-default-saas/cdp-saas/hdp-data/
```

Additional configuration:

The following AWS S3 specific configurations are recommended, apart from configuring the default settings like the numbers of mappers and bandwidth per mapper.

1. fs.s3a.fast.upload=true, this configuration allows to speed up the data transfer
2. Setting up s3 proxy server configurationsfs.s3a.proxy.* configs
3. fs.s3a.connection.ssl.enabled=true, this configuration enables SSL connection to the S3 endpoint.

Migrating HDFS native permissions to Cloudera

If you have HDFS native permissions in your CDH or HDP clusters, you learn how to convert the native permissions into Ranger policy format and import the policies to Cloudera. You can choose to ignore this migration process if you do not have any HDFS native native permissions.

Migrating HDFS native permissions from CDH or HDP to Cloudera involves a combination of Extract Convert Transform Import operations.

The HDFS native permissions are first extracted as a file in a specific format, which are then converted and transformed into Ranger AWS S3 policies using the Ranger policy migration utility. The transformed Ranger policies are finally imported into Cloudera using the authzmigrator tool.

Extracting HDFS native permissions

Learn how to use the HDFS Permissions Export utility to extract HDFS native POSIX permissions from a source cluster running CDH or HDP. The extracted HDFS permissions are then used to create Ranger S3 policies that can be used in Cloudera.

About this task

The HDFS Permissions Export utility exports the permissions to a .csv file in the following format with the permissions sorted in ascending order:

```
Syntax:
"/resource/path"|"username"|"groupname"|"permission"
```

where,

- "/resource/path" refers to resource entities
- "username" refers to user entries
- "groupname" refers to group entries
- "permission" refers to the HDFS permission entries

```
Example:
"/dir1"|"hdfs"|"supergroup, public"|"read, execute"
"/dir1/dir1"||"supergroup, public"|"read, execute"
"/dir1/dir1"|"hdfs"||"read, write, execute"
```

Before you begin

- The source and target migration clusters must be running the following supported versions:
  - Source cluster: HDP 2.6.5, HDP 3.1.5, CDH 5.16, CDH 6.3
  - Target cluster: CDP 7.2.15 or higher
- Secure Copy (SCP) the HDFS keytab file from the source cluster to your local system
- Run the klist command to view the list of Kerberos principals available in the keytab — ${RANGER_ADMIN_KEYTAB_PATH} and then run the kinit command to authenticate the user.

```
klist -kt ${RANGER_ADMIN_KEYTAB_PATH}

kinit -kt ${RANGER_ADMIN_KEYTAB_PATH} rangeradmin/_HOST@REALM
```

Procedure

1. Download the HDFSPermissionsExportTool-<version>.jar file from Cloudera Archive.

2. Run the following command to extract HDFS permissions from the source cluster.

```
hadoop jar ./<path>/HDFSPermissionsExportTool-<version>.jar hdfs_namenod
e_host:hdfs_port source_directories
```

where,

- <path> refers to the location of the downloaded .jar file.
- hdfs_namenode_host:hdfs_port refers to the URI of the active NameNode and the HDFS NameNode port. For example, active-namenode.example.com:8020
- source_directories refers to the root directory from where you want the permissions scan to begin. For example, "/" or "/dir1". You can also specify multiple directories — "/hbase/yarn"

```
hadoop jar ./target/HDFSPermissionsExportTool-<version>.jar active-namen
ode.example.com:8020 /user /ranger
        /hbase
```

This creates a HDFS_Permissions_Export_<timestamp>.csv file that contains the required HDFS permissions.

Important: Be sure to use the URI of the active NameNode. Using the passive NameNode URI or the NameNode HA URI results in a failure.

3. Use the scp command and copy HDFS_Permissions_Export_<timestamp>.csv from the source cluster to the migration node of the target cluster.

Important: Since the .csv filename contains a colon (:), ensure that you provide the full path in the scp command to prevent the command from treating ":" as a separator.

What to do next

Convert the extracted HDFS native permissions into Ranger HDFS policies and then transform them into Ranger S3 policies.

Related Information

Converting HDFS native permissions into Ranger HDFS policies

Converting HDFS native permissions into Ranger HDFS policies

Learn how to convert the extracted HDFS native permissions into Ranger HDFS service policies, which can then be transformed into Ranger S3 policies.

Before you begin

- The source and target migration clusters must be running the following supported versions:
  - Source cluster: HDP 2.6.5, HDP 3.1.5, CDH 5.16, CDH 6.3
  - Target cluster: Cloudera Public Cloud 7.2.15 or higher
- You must have copied the extracted HDFS native permissions (.csv file) into the migration cluster node.
- You must have copied the Ranger policy migration utility, ranger-<version>-policymigration.tar.gz file to the migration cluster node.
  1. Log in to the migration cluster's node (or a Data Lake node of a cloud cluster) where the ranger-admin binaries are available. For example, /opt/cloudera/parcels/CDH/lib/ranger-admin
  2. Copy /opt/cloudera/parcels/CDH/lib/ranger-admin/policymigration/ranger-<version>-policymigration.tar.gz to some location of the migration cluster machine from where you want to run the Ranger policy migration utility.
- You must have read, write, and execute permissions on the directory where the Ranger policy migration utility is copied.
- You must have SUDO or ROOT access to perform the migration process.

Procedure

1. Log in to the migration cluster node where you have copied the ranger-<version>-policymigration.tar.gz file. Untar the file and navigate to the ranger-<version>-policymigration directory.
2. Update the “Transform” part of the env.sh file with the right set of values according to your environment.

   For more information, see Supported Input parameters for Transform operation.
3. Run the transform.sh script with the convert command to convert the extracted HDFS native permissions to Ranger HDFS policy format.

```
Syntax:
./transform.sh convert $native_hdfs_permissions_csv_file_path
```

```
Example:
./transform.sh convert /tmp/HDFS_Permissions_Export_<timestamp>.csv
```

Results

The command generates a HDFS_Permissions_Export_<timestamp>_convert.json file, which is then transformed into the required Ranger S3 policies. If you encounter any errors, see the logs/ranger-policymigration.log file.

What to do next

Transform the converted Ranger HDFS policies into Ranger S3 policies.

Related Information

Supported Input parameters for Transform operation

Extracting HDFS native permissions

Transforming Ranger HDFS policies into Ranger S3 policies

Transforming Ranger HDFS policies into Ranger S3 policies

Learn how to transform Ranger HDFS service policies into Ranger AWS S3 policies, which can then be imported into the Cloudera cluster Ranger.

Before you begin

- You must have converted the extracted Ranger HDFS native permissions (.csv file) into Ranger HDFS policies (.json file) using the transform.sh convert command.
- You must have copied the Ranger policy migration utility, ranger-<version>-policymigration.tar.gz file to the migration cluster node.
  1. Log in to the migration cluster's node (or a Data Lake node of a cloud cluster) where the ranger-admin binaries are available. For example, /opt/cloudera/parcels/CDH/lib/ranger-admin
  2. Copy /opt/cloudera/parcels/CDH/lib/ranger-admin/policymigration/ranger-<version>-policymigration.tar.gz to some location of the migration cluster machine from where you want to run the Ranger policy migration utility.
- You must have read, write, and execute permissions on the directory where the Ranger policy migration utility is copied.
- You must have SUDO or ROOT access to perform the migration process.

Procedure

1. Log in to the migration cluster node where you have copied the ranger-<version>-policymigration.tar.gz file. Untar the file and navigate to ranger-<version>-policymigration directory.

2. Update the "Transform" part of the env.sh file by updating the following parameters or as required for your environment.
   - S3_BUCKET_NAME
   - SERVICE_NAME_FOR_NATIVE_POLICIES

   Note: If the usernames in the target cluster are different from the usernames in the source cluster, provide a user mapping file as an input by updating the RANGER_POLICYMIGRATION_USERS_MAPPING _FILE parameter in the env.sh file.

   For more information, see Supported Input parameters for Transform operation.
3. Run the transform.sh script with the transform command to convert the Ranger HDFS policies to Ranger AWS S3 policy format.

```
Syntax:
./transform.sh transform $converted_ranger_policies_file_path
```

```
Example:
./transform.sh transform /tmp/HDFS_Permissions_Export_<timestamp>_conve
rt.json
```

Results

The command generates a HDFS_Permissions_Export_<timestamp>_convert_transform.json file, which can then be imported using the Authzmigrator utility into the AWS S3 service of the target Cloudera cluster. If you encounter any errors, see the logs/ranger-policymigration.log file.

What to do next

Import the transformed HDFS native permissions into the Ranger AWS S3 service of the target Cloudera cluster using the Authzmigrator tool.

Related Information

Supported Input parameters for Transform operation

Extracting HDFS native permissions

Converting HDFS native permissions into Ranger HDFS policies

Importing Ranger AWS S3 policies

Learn how to import the transformed HDFS native permissions into the Ranger AWS S3 service of the target Cloudera cluster using the Authzmigrator tool.

Before you begin

- You must have extracted, converted, and transformed the native HDFS permissions into Ranger AWS S3 policy format.
- Ensure that the RANGER_ADMIN role is up and running in the Cloudera Manager/Cloudera portal of the target environment.

Procedure

1. Log in to the RANGER_ADMIN portal with administrator user credentials or user privileges as ADMIN.
   a) From Settings User/Groups/Roles Search , search for the "ranger" user.
   b) Click the "ranger" user and change its role to "Admin", and save the changes.
2. Confirm authz-ingest.zip is available under '/opt/cloudera/cm/lib/dr/' path. If authz-ingest.zip is not available, contact Cloudera customer support to provide it. Once you have authz-ingest.zip, copy to the /opt directory of the migration cluster node (or a Data Lake node of the cloud cluster).

3. Extract the authz-ingest.zip file into the /opt directory.

   Note: /opt is referred to as an example directory.

   The authz-ingest file contains the following directories:

   - config-files
   - scripts

4. If Ranger is TLS/SSL enabled using Cloudera Manager AUTO-TLS, then copy the cm-auto-global-truststore.jks file from the Ranger admin process directory of the Ranger admin host to /opt/authz-ingest/config-files/ of the migration tool host. Perform these steps to find the location of the Ranger admin process directory:

   a) Run the following command to get to the Ranger admin process:

      ps -ef | grep proc_rangeradmin

   b) Search for the -cp option from the output of the above command and look for "/var/run/cloudera-scm-agent/process/<ID>-ranger-RANGER_ADMIN"

   If you have set up SSL manually or using a different method, then copy the file that you have created for trust stores to the /opt/authz-ingest/config-files directory.

   For example, cp /path/to/cm-auto-global_truststore.jks /opt/authz-ingest/config-files/

5. Export the following environment variables to the target cluster Ranger:

   - JAVA_HOME - Java home path used on the cluster
   - RANGER_ADMIN_URL - URL of the Ranger Admin portal
   - RANGER_ADMIN_SSL_ENABLED - Indicates if SSL is enabled or disabled
   - RANGER_ADMIN_TRUSTORE_PASSWORD - If Ranger is TLS/SSL enabled using Cloudera Manager AUTO-TLS, then the password can be found in the Ranger host file, /etc/hadoop/conf/ssl-client.xml, property — 'ssl.client.truststore.password'

```
export JAVA_HOME=/usr/java/jdk1.8.0_232-cloudera/
export RANGER_ADMIN_URL=<ranger admin url with port>
export RANGER_ADMIN_SSL_ENABLED=<true or false>
export RANGER_ADMIN_TRUSTORE_PASSWORD=<ssl.client.truststore.password>
```

6. Go to /opt/authz-ingest/config-files and open the authorization-migration-site.xml file.

   The authorization-migration-site.xml contains the following properties:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>authorization.migration.export.target_services</name>
    <value>HIVE,KAFKA</value>
  </property>
  <property>
    <name>authorization.migration.export.migration_objects</name>
    <value />
  </property>
  <property>
    <name>authorization.migration.export.output_file</name>
    <value>hdfs:///user/sentry/export-permissions/permissions.json</value>
  </property>
  <property>
    <name>authorization.migration.ingest.is_dry_run</name>
    <value>false</value>
  </property>
  <property>
    <name>authorization.migration.role.permissions</name>
    <value>true</value>
  </property>
```

```
  <property>
    <name>authorization.migration.translate.url.privileges</name>
    <value>false</value>
  </property>
  <property>
    <name>authorization.migration.ingest.merge.ifexists</name>
    <value>true</value>
  </property>
  <property>
    <name>authorization.migration.migrate.url.privileges</name>
    <value>true</value>
  </property>
</configuration>
```

7. Search for the authorization.migration.ranger.import.input_file property in the authorization-migration-site.xml file and provide the location of the transformed HDFS permissions file as the property value.

```
<property>
 <name>authorization.migration.ranger.import.input_file</name>
<value>file:///tmp/HDFS_Permissions_Export_<timestamp>_convert_transform.j
son</value>
</property>
```

If you are unable to find this property, then add the property in the file.

8. For Kerberos-based authentication, perform the following steps:

Important: If the Key Distribution Center (KDC) server is not installed in your environment or imported without the Kerberos authentication, you can skip this procedure and use the process provided for non-Kerberos authentication. Also, if PAM authentication is enabled in the target cluster, then Kerberos authentication is recommended to import ranger policies quickly.

a) Find the Ranger admin keytab location and provide it in the following command:
export RANGER_ADMIN_KEYTAB_PATH=/path/to/ranger.keytab

In the CDEP cluster, ranger keytab is available in the ranger host location /cdep/keytabs/rangeradmin.keytab or under the ranger admin process directory — /var/run/cloudera-scm-agent/process/<ranger-process-ID>-ranger-RANGER_ADMIN

b) If your ranger node and migration node are different, copy the ranger keytab to the migration node.

c) Run the klist command on the migration node to view the list of Kerberos principals available in the keytab — ${RANGER_ADMIN_KEYTAB_PATH}
klist -kt ${RANGER_ADMIN_KEYTAB_PATH}

d) Run the following kinit command and update _HOST and REALM correctly.
kinit -kt ${RANGER_ADMIN_KEYTAB_PATH} rangeradmin/_HOST@REALM

9. For non-Kerberos authentication, add the following properties in the /opt/authz-ingest/config-files/authorization-migration-site.xml file with the appropriate values, and then save and close the file.

```
<property>
    <name>ranger.admin.username</name>
    <value>admin</value>
</property>

<property>
    <name>ranger.admin.passwd</name>
    <value>admin123</value>
</property>
<property>
<name>authorization.migration.kerberos.authentication</name>
<value>false</value>
</property>
```

10. Go to the /opt/authz-ingest/scripts directory and run the import script.
    sh authz-import.sh

    If the import operation fails, look for details in the output of the command printed in the terminal window or see the target Ranger admin log files.

Results

After the import operation is complete, the authzmigrator tool displays the total policies count, created, skipped, and failed policies count. You can refer to the console output logs to verify if policies are imported or not.

```
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: ========================SUMMARY==================================
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: Ingesting ranger policies finished successfully
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: Total Policies=10000, Created=10000, Skipped=0, Failed=0, Timedout=0
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: ==================================================================
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: Ingesting of Ranger Policies to ranger service is complete
```

Related Information

Extracting HDFS native permissions

Converting HDFS native permissions into Ranger HDFS policies

Transforming Ranger HDFS policies into Ranger S3 policies

Migrating Ranger policies from HDP to Cloudera

In the HDP environment consisting of Ranger policies, the migration is performed using the combination of Export -> Transform -> Import operations.

About Migrating Ranger policies

Migrating Ranger policies involves export and transform operations.

Firstly, the Ranger policies available in the source HDP clusters are exported and copied to the target cluster, followed by the transform and import operations. The export and transform operations are performed using the policy migration utility.

The Ranger Policy Migration utility files in the directory

ranger-<version>-policymigration consists of the following scripts:

- env.sh
- exportPolicy.sh
- transform.sh

The contents of the policy migration utility would be copied to the source HDP cluster from which the Ranger policies are to be migrated. The output file from the transform process would be transitioned to the import operations. The import process consists of the import utility (Authzmigrator tool).

Note: Currently, the migration process does not support migrating Group / Role name transformation.

Migration prerequisites

You need to know the prerequisites for migrating Ranger policies from HDP to Cloudera.

You must meet the following general prerequisites before starting the migration process:

- AWS is your Cloud provider. Migrating to Azure / GCP is currently not supported.
- The source and target migration clusters must be running the following supported versions:
  - Source cluster: HDP 2.6.5, HDP 3.1.5.
  - Target cluster: Cloudera Public Cloud 7.2.15 or higher.
  - The minimum JAVA version. Default is 1.8.

Copying Policy Migration utility to the source cluster

Policy migration utility must be available on the source cluster.

About this task

You must copy the contents of the policy migration tool to the source cluster.

Procedure

1. Sign-in to the Migration cluster's node (or a Data Lake node of the cloud cluster) where the ranger-admin binaries are available.
   For example: /opt/cloudera/parcels/CDH/lib/ranger-admin
2. Copy /opt/cloudera/parcels/CDH/lib/ranger-admin/policymigration/ranger-<version>-policymigration.tar.gz to the source HDP cluster machine from where you want to export the HDP Ranger policies.

   Additionally, copy the same file to some other location of the migration cluster machine (or a Data Lake node of the cloud cluster) from where you want to execute the Ranger policy migration utility to transform Ranger policies.
3. Login to the HDP source cluster where you have copied the ranger-<version>-policymigration.tar.gz file. Untar the file and navigate to ranger-<version>-policymigration directory.

Performing Export and Transform operations

Using the Policy Migration utility, you must perform export and transform operations.

About the export operation

Call exportPolicy.sh script to export Ranger policies.

The exportPolicy.sh script performs the following tasks:

- Reads the env.sh file and loads the input parameters values
- Makes a CURL request to ranger/export JSON REST API and sends the required parameters
- Downloads and saves the exported Ranger policies at the location provided in $EXPORT_OUTPUT_LOCATION. It also downloads and saves the exported Ranger services at the same location with the name exportedRangerServices.json

Note: Valid for HDP cluster only.

Running the export operation

You must export the Ranger policies.

Procedure

1. Update the "export" part of the env.sh file with the right set of values as per your environment.
2. Run the exportPolicy.sh file.

   Running sh.exportPolicy.sh could throw an error because env.sh file is not available. You can resolve this condition by running the script file with the following usage: ./exportPolicy.sh
3. Verify if the exported Ranger policies file was created or not.
4. Verify if the exported Ranger services file was created or not.

   If any error refers to the source ranger-admin logs.
5. Copy the exported Ranger policies JSON file and exported Ranger services JSON file to the migration cluster node (or a Data Lake node of cloud cluster).

Related Information

Supported Input parameters for Export operation

About the transform operation

Call transform.sh to transform Ranger HDFS policies to Ranger AWS S3 policies and replace policy field's value as per the target environment.

The transform.sh script performs the following tasks:

- Reads env.sh file and loads the input parameters values
- Checks Java versions
- Creates provided system user and group
- Provides ownership to the above created system user on install files and folders
- Update given parameters values to ranger-policymigration-site.xml
- Exports JAVA_OPTS and adds required options in the environment.
- Can call the Ranger PolicyMigration jar/code for convert/transform as requested

Running the transform operation

You must transform the Ranger policies.

Procedure

1. Login to the migration cluster node (or a Data Lake node of the cloud cluster) where you have copied the ranger-<version>-policymigration.tar.gz file. Untar the file and navigate to ranger-<version>-policymigration directory.
2. Update the "Transform" part of the env.sh file with the right set of values as per your environment.
3. Transform Ranger policies: Run the transform.sh file.
   a. Syntax: ./transform.sh transform $exported_ranger_policies_file_path $exported_ranger_services_file_path
   b. Example: ./transform.sh transform /tmp/policies.json /tmp/services.json

   Note: $exported_ranger_policies_file_path and $exported_ranger_services_file_path are the paths of exported Ranger policies and services file in JSON format received from Running the export operation on page 49.

   Attention: Output of the above command is in the JSON format and this JSON file is required for Performing Import operation on page 50. If any error is found, refer to logs/ranger-policymigration.log file.

Related Information

Supported Input parameters for Transform operation

Performing Import operation

You must import the Ranger policies to Cloudera cluster Ranger.

Note: If the users that are defined within the policy do not exist on the target cluster / storage, they are auto-created, import operation succeeds, and no further user action is required.

Procedure

1. Login to the RANGER_ADMIN portal with administrator user credentials or user privileges as ADMIN.
   a. Under Settings > Users/Groups/Roles > Search for the ranger user.
   b. Click on the ranger user and change its role to Admin and save the changes.
2. Confirm authz-ingest.zip is available under '/opt/cloudera/cm/lib/dr/' path. If authz-ingest.zip is not available, contact Cloudera customer support to provide it. Once you have authz-ingest.zip, copy to the /opt directory of the migration cluster node (or a Data Lake node of the cloud cluster).

3. Unzip the authz-ingest.zip file into the /opt directory which contains the following files:
   - config-files
   - scripts

   The config-files contains the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>authorization.migration.export.target_services</name>
    <value>HIVE,KAFKA</value>
  </property>
  <property>
    <name>authorization.migration.export.migration_objects</name>
    <value />
  </property>
  <property>
    <name>authorization.migration.export.output_file</name>
    <value>hdfs:///user/sentry/export-permissions/permissions.json</value>
  </property>
  <property>
    <name>authorization.migration.ingest.is_dry_run</name>
    <value>false</value>
  </property>
  <property>
    <name>authorization.migration.role.permissions</name>
    <value>true</value>
  </property>
  <property>
    <name>authorization.migration.translate.url.privileges</name>
    <value>false</value>
  </property>
  <property>
    <name>authorization.migration.ingest.merge.ifexists</name>
    <value>true</value>
  </property>
  <property>
    <name>authorization.migration.migrate.url.privileges</name>
    <value>true</value>
  </property>
</configuration>
```

Attention: While using permissions.json file, you might encounter an error or a warning message while using the default configuration. You can ignore this and continue with the process.

4. If Ranger is TLS/SSL enabled using Cloudera Manager AUTO-TLS, you must then from the Ranger admin host, copy cm-auto-global_truststore.jks (from the Ranger-admin process directory).

   Note: In Cloudera One or Cloudera Public Cloud environment TLS is always enabled by default.

   For example:

   /var/run/cloudera-scm-agent/process/<ranger-process-ID>-ranger-RANGER_ADMIN) to the location /opt/authz-ingest/config-files/ of the migration tool host.

   Follow these steps to find the Ranger admin process directory:

   - Execute the below command to get to the RANGER_ADMIN process:

   ps -ef | grep proc_rangeradmin

   - Search for -cp option from the above command output and look for

     "/var/run/cloudera-scm-agent/process/<ID>-ranger-RANGER_ADMIN"

   If you have done SSL setup manually or in a different method, copy the filename which you created for truststores.

   For example:

   cp /path/to/cm-auto-global_truststore.jks /opt/authz-ingest/config-files/

5. Export the following variables to the target Ranger:

   - **a.** JAVA_HOME - Java home path used on the cluster
     b. RANGER_ADMIN_URL - Ranger Admin portal url
     c. RANGER_ADMIN_SSL_ENABLED - either true or false
     d. RANGER_ADMIN_TRUSTORE_PASSWORD - If Ranger is TLS/SSL enabled using Cloudera Manager AUTO-TLS, then the password can be found on the Ranger host /etc/hadoop/conf/ssl-client.xml property 'ssl.client.truststore.password'

6. Go to the location /opt/authz-ingest/config-files and open the file authorization-migration-site.xml

   Locate the property “authorization.migration.ranger.import.input_file” and provide the location of the transformed output file generated from Running the transform operation on page 50.

   Note: If the property is not available then you must add this property in the file.

   Example: file:///tmp/exportedRangerPolicies_transform.json

   <property>

   <name>authorization.migration.ranger.import.input_file</name>

   <value>file:///tmp/exportedRangerPolicies_transform.json</value></property>

7. For the Kerberos-based authentication (or in a non Cloudera-SaaS environment) use the following process

   If the KDC server is not installed in your environment or imported without the Kerberos authentication, you can skip this procedure and use the process provided for non-Kerberos authentication. Also if PAM auth is enabled in the target cluster then Kerberos authentication is recommended to import Ranger policies quickly.

   - Find the Ranger admin keytab location and provide it in the following command:

   export RANGER_ADMIN_KEYTAB_PATH=/path/to/ranger.keytab

   In the CDEP cluster, Ranger keytab shall be available in the Ranger host location “/cdep/keytabs/rangeradmin.keytab” or under the Ranger admin process directory

   “/var/run/cloudera-scm-agent/process/<ranger-process-ID>-ranger-RANGER_ADMIN” location.

   Steps to find the Ranger admin process directory:

   - Execute the following command to get to the RANGER_ADMIN process.

   ps -ef | grep proc_rangeradmin

   - Search for -cp option from the above command output and look for "/var/run/cloudera-scm-agent/process/<ID>-ranger-RANGER_ADMIN"
   - If your Ranger node and migration node (target data lake Ranger node) are different, copy the Ranger keytab to the migration node (target data lake Ranger node).
   - Run the below klist command on the migration node (target data lake Ranger node) to get the list of principals available in ${RANGER_ADMIN_KEYTAB_PATH}

   klist -kt ${RANGER_ADMIN_KEYTAB_PATH}

   - Run the below kinit command using the ${RANGER_ADMIN_KEYTAB_PATH} with rangeradmin/_HOST@REALM principle.
   - Update the _HOST and the REALM correctly using the below command.

   kinit -kt ${RANGER_ADMIN_KEYTAB_PATH} rangeradmin/_HOST@REALM

8. For non-Kerberos authentication, add the following properties and provide relevant values. After providing the values to configurations, save, and close the file.

```
<property>
    <name>ranger.admin.username</name>
    <value>admin</value>
</property>
<property>
<name>authorization.migration.kerberos.authentication</name>
<value>false</value>
</property>
```

9. Run the migration script:

   cd authz-ingest/scripts

   sh authz-import.sh

   Note: If import operation fails, locate for details in the output of the command printed on the terminal window or refer to the target Ranger admin log files.

10. After execution, the authMigrator utility shall print total policies count, skipped policies count and the failed policy counts on the same console window. Users

should refer to the console output logs to ensure policies have been imported or

```
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: ========================SUMMARY======
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: Ingesting ranger policies finished s
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: Total Policies=10000, Created=10000,
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: ====================================
22/04/22 10:42:12 INFO authzmigrator.RangerIngestTask: Ingesting of Ranger Policies to rang
```

not.

Supported Input parameters for Export operation

When you perform the export operation, depending on your cluster environment, you must make sure to configure the env.sh file for the supported input values accordingly.

Input Parameters	Description
POLICYMGR_EXTERNAL_URL	This configuration is used for the Ranger administrator URL. Ranger admin URL should be given along with protocol and port. Multiple Ranger admin URLs are not supported. Example: http://testhost.root.hwx.site:6080 Default : http://localhost:6080 Mandatory: Yes
POLICYMGR_KERBEROS_ENABLED	For kerberos authentication, set this configuration value to true. Kinit with right principal/keytab for authentication before executing the expo rtPolicy.sh script. Default : false Mandatory: No
POLICYMGR_HTTPS_ENABLED	For SSL/HTTPS authentication, set this configuration value to true. If this property is set to true, provide a valid certificate file path in the configuration POLICYMGR_HTTPS_CERT_FILE Default : false Mandatory: No
POLICYMGR_HTTPS_CERT_FILE	For SSL/HTTPS authentication set config POLICYMGR_HTTPS_ENABLED value to true and provide a valid certificate file path for this config Example: /tmp/cacerts.pem Default : empty string Mandatory: No
POLICYMGR_ADMIN	This configuration is for Ranger admin user loginID. loginID provided in this config must exist in ranger-admin and must have an ‘admin’ role. For example: If POLICYMGR_KERBEROS_ENABLED is set to true, POLICYMG R_ADMIN does not need to be configured with a user id. It is inherited from the authenticated Kerberos principal." This is optional for secured/kerberos/knoxsso authentication. Default : empty string Mandatory: No

Input Parameters	Description
POLICYMGR_PWD	This configuration is for the login password of the loginID provided in config POLICYMGR_ADMIN. This is optional for secured/kerberos/knoxsso authentication. Default : empty string Mandatory: No
SOURCE_SERVICE_NAME	This configuration is for Ranger service names of which Ranger policies need to be exported. Multiple service names can be given separated by comma(,) characters. If an empty string is provided then the utility will try to export Ranger policies of all services. Example: hadoopdev,hivedev Default : empty string Mandatory: No
EXPORT_OUTPUT_LOCATION	This configuration is for the output file of exported Ranger policies. If a location is given then the user must have write permission on that file/directory. If an empty string is given then the location shall be the current location. Only local file system path is supported and HDFS file system path is not supported. If directory path is given then file name shall be “exportedRangerPolicies_$timestamp”. If path contains the file name also then it will create/overwrite the existing file with the exported Ranger policies JSON content. example : /tmp Default : empty string Mandatory: No
POLICYMGR_KNOXSSO_TOKEN	This config is to provide the KNOX SSO Token for knox sso based authentication to source Ranger-admin. Example : eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJhZG1pbjEiLCJpc3MiOiJLTk9YU1NPIiwiZXhwIjox gB-AVIbnfoKQoJobHTAAXgHiV-6dehFK2oWKWDX7ZXwj5Oxf-11a7ptBkRlRiPeJW7Mr Default : empty string Mandatory: No

Note: To retrieve the Knox SSO token make a CURL request to ranger-admin and fetch details.

Example : curl -iku ‘admin’:’admin’ "https://{KNOX-HOST}:8443/gateway/knoxsso/api/v1/websso?originalUrl=http://{RANGER-ADMIN-HOST}:6080/service/plugins/services/1"

Sample output:

HTTP/1.1 307 Temporary Redirect

Date: Wed, 09 Mar 2022 04:11:28 GMT

X-Frame-Options: DENY

Set-Cookie: KNOXSESSIONID=eabe18mw4qrs1f58xjg9d000q;Path=/gateway/knoxsso;Secure;HttpOnly

Expires: Thu, 01 Jan 1970 00:00:00 GMT

Set-Cookie: rememberMe=deleteMe; Path=/gateway/knoxsso; Max-Age=0; Expires=Tue, 08-Mar-2022 04:11:28 GMT

Set-Cookie: hadoop-jwt=eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJhZG1pbjEiLCJpc3MiOiJLTk9YU1NPIiwiZXhwIjoxNjQ2ODk5MDg4fQ.TZf18N0zHMU

gB-
AVIbnfoKQoJobHTAAXgHiV-6dehFK2oWKWDX7ZXwj5Oxf-11a7ptBkRlRiPeJW7MmHuZgdJIKmauJMmp8Z_67diImU__6wj

Location: http://localhost:6080/service/plugins/services/1

Content-Length: 0

Server: Jetty(9.2.15.v20160210)

Supported Input parameters for Transform operation

When you perform the transform operation, depending on your cluster environment, you must make sure to configure the env.sh file for the supported input values accordingly.

Input paramaters	Description
PYTHON_COMMAND_INVOKER	This configuration is for the python command name or path to the python command file. Python 2.x and 3.x both versions are supported. If python version 3.X command is not aliased with python then python3 must be entered here. If python 2 is installed then python or python2 must be provided as a value to this config. Default : python Mandatory: Yes
RANGER_POLICYMIGRATION_EXCLUDE_SERVICE_TYPES	This configuration is to exclude the Ranger policies of given Ranger service types during the transformation process. Ranger policies of given service-types will not be added in the transformed output file. **Note:** Currently “tag” migration is not supported, hence it is recommended that users must add "tag" to exclude service types. Example: storm,tag Default : storm,tag Mandatory: No
SOURCE_DESTINATION_RANGER_SERVICE_TYPE_MAPPING	This config is to tell the utility the mapping of source and target Ranger service type. Based on the given source service type, Ranger policies will be transformed to a given target service type. Source and target service type are separated by colon (:) At present, Only Ranger hdfs service policies can be converted to s3 only. Example: {hdfs:s3} Default : {hdfs:s3} Mandatory: No
SOURCE_DESTINATION_RANGER_SERVICE_NAME_MAPPING	This configuration is to provide the mapping of source and target Ranger service names to the utility. Based on the given service name mapping, during transformation service names of given Ranger policies will be replaced with the target service names. Source and target service names are separated by colon (:). Multiple entries are allowed and need to be separated by comma(,) character. If no mapping is provided then service names will remain the same even after the transformation process. Example: {cm_hdfs:cm_s3,c1_yarn:cm_yarn} Default : empty string Mandatory: No

Input paramaters	Description
SOURCE_DESTINATION_RANGER_ZONE_NAME_MAPPING	This configuration is to provide the mapping of source and target Ranger zone names to the utility. Based on the given zone name mapping, during transformation, zone names of given Ranger policies will be replaced with the target zone names. Source and target zone names are separated by colon (:). Multiple entries are allowed and need to be separated by comma(,) character. If no mapping is provided then zone names will remain the same even after the transformation process. Example: {oldzone1:newzone1} Default : empty string Mandatory: No

RANGER_POLICYMIGRATION_RESOURCE_MAPPING_FILE	This configuration is to provide the source and target service resource mappings to the utility. Based on the given resources mapping, during transformation resources of given Ranger policies will be replaced with the target resources. If no mapping is provided then the resource string will remain the same even after the transformation process.
	
	File format is CSV. Local filesystem path is supported only.
	
	Example value: /path to /resourceMappings.csv
	
	Default : empty string
	
	Mandatory: No
	
	Resource mapping for hdfs policy resource path=> For the JSON file format we can have multiple resource also in the JSON object formatted like :
	
	Sample resource mapping entries for a json file:
	
	`{"/srcdir1/srcfile1":"/tardir1/tarfile1",`
	`"/srcdir2/srcfile2":"/tardir2/tarfile2"}`
	
	=> And with the CSV format, it's pipe-delimited with one entry per line.
	
	Sample resource mapping entries for a csv file:
	
	/srcdir1/srcfile1|/tardir1/tarfile1
	
	/srcdir2/srcfile1|/tardir2/tarfile1

Input paramaters	Description
RANGER_POLICYMIGRATION_HIVE_URL_MAPPING_FILE	This config is to provide the source and target HIVE service URL policy resource mappings to the utility. Based on the given URL mapping, during transformation the URL resource of given Ranger Hive URL policies will be replaced with the target URL value. If no mapping is provided then the resource string will remain the same even after the transformation process. File format is CSV. Local filesystem path is supported only. Example value: /path to /resourceMappings.csv Default : empty string Mandatory: No Resource mapping for hive policy URL resource=> For the JSON file format we can have multiple resource also in the JSON object formatted like : Sample resource mapping entries for a json file: {"/user/hive/warehouse/customers/customers":"s3a://mybucket/user/hive/warehouse/customers/customers","hdfs://dysentry01-3.dysentry01.root.hwx.site:8020":"s3a://mybucket1"} => And with the CSV format, it's pipe-delimited with one entry per line. Sample resource mapping entries for a csv file: /user/hive/warehouse|s3a://mybucket/user/hive/warehouse hdfs://localhost:8020|s3a://mybucket1
S3_BUCKET_NAME	This configuration is for S3 bucket name which shall be assigned as a value to the resource type “bucket” key of each transformed policy of S3. If the target Ranger service type is S3 then value for this property must be provided. Default : myS3Bucket Mandatory: No
RANGER_POLICYMIGRATION_USERS_MAPPING_FILE	This configuration is to provide the mapping of usernames in the source and target Ranger policies. Based on the given user names mapping, during transformation user names in the given Ranger policies will be replaced with the target user names. If no mapping is provided then user names will remain the same even after the transformation process. File format is CSV. Local filesystem path is supported only. Example: /path/to/usersMappings.csv Default : empty string Mandatory: No Format: => For the JSON file format we can have multiple users also in the JSON object formatted like : {“testuser1”:”testuser1@host.com”, “testuser2”:”testuser2@host.com”} => And with the CSV format, it's pipe-delimited with one entry per line. testuser1|testuser1@host.com testuser2|testuser2@host.com

Input paramaters	Description
RANGER_POLICYMIGRATION_TRANSFORM_OUTPUT_FILE	This configuration is for the output file of transformed Ranger policies. If a location is given then the user must have write permission on that file/directory. If an empty string is given then the location shall be the same as the location of the input file. Local file system path should be used for this configuration. If a directory path is given then the suffix “transform” will be added in the input file name. If path contains file name also then utility shall create/overwrite the existing file with the transformed Ranger policies JSON content. Example : /tmp Default : empty string Mandatory: No
SERVICE_NAME_FOR_NATIVE_POLICIES	This configuration is to assign a service name to the HDFS Ranger policies. These Ranger policies are those Ranger policies which are transformed from the native HDFS permissions. Example : cm_hdfs Default : cm_hdfs Mandatory: Yes
RANGER_POLICYMIGRATION_LOG_DIR	Path to log file directory. Transform utility shall write the log files in this directory.
POLICYMIGRATION_INSTALL_DIR	Path of migration utility parents directory, default is present working directory. Example : ${PWD} Default : ${PWD} Mandatory: No
LOGFILE	Name of the log file. Transform utility shall write all the logs to this file.
LOGFILES	Name of the log file. Presently there is a single log file for all operations of the migration utility.
RANGER_POLICYMIGRATION_MAX_HEAP_SIZE	This allocates Max heap size for the transform utility execution. Default value is 4GB.
UNIX_USER	This configuration is for the user who will be the owner of Ranger policy migration utilities directory and files.
UNIX_USER_PWD	
UNIX_GROUP	This configuration is for the group name which shall have the same permissions as the owner of the Ranger policy migration utility directory and files.

Migrating Hive data from HDP 2.x or HDP 3.x to Cloudera

The recommended way to migrate Hive data from HDP to Cloudera depends on the types of tables you are migrating: external, legacy-managed, or ACID (managed) tables.

The default behavior in Cloudera has changed so that all newly created managed tables are transactional tables by default. To minimize impact to applications during upgrade and keep legacy behavior, all non-acid legacy tables (both managed and external) will be converted to external tables in Cloudera during migration. Legacy ACID tables will be migrated to managed ACID tables in CDP.

LEGACY TABLE TYPE	MIGRATION PATH FOR CDP
EXTERNAL TABLE	EXTERNAL TABLE
MANAGED TABLE (No ACID)	CONVERT TO EXTERNAL TABLE
MANAGED ACID TABLE (ACID enabled - HDP only)	MANAGED TABLE

Scenario 1: Migrating Non-ACID tables (SCHEMA_ONLY + distcp)

All managed and external Non-ACID tables will be migrated to external tables in Cloudera using the following components:

- HMS-Mirror: using the SCHEMA_ONLY mode to transfer metadata
- DistCP

In this scenario, HMS-Mirror and distcp are triggered from the source cluster (LEFT):

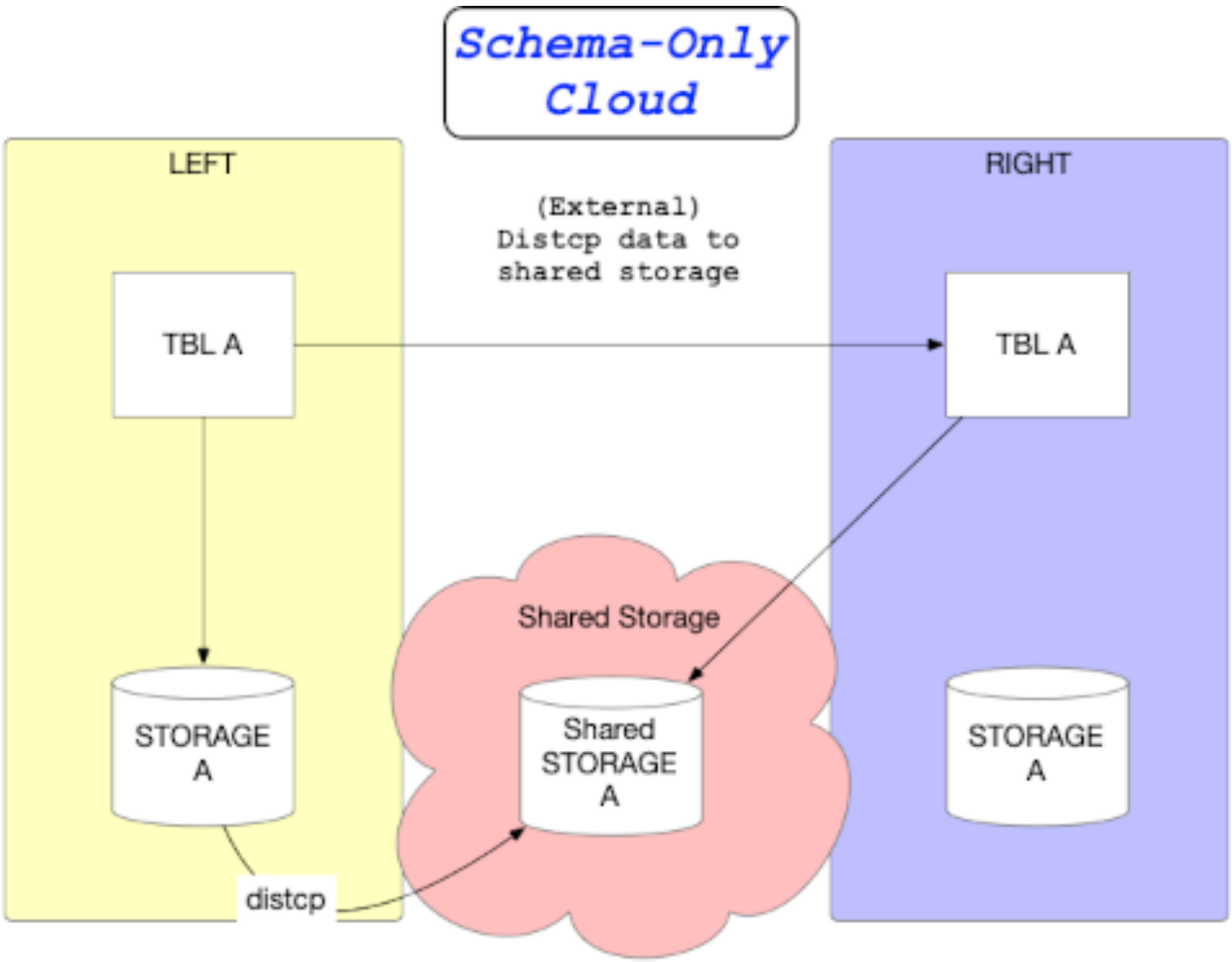

If direct Hive connectivity from on-prem to Cloudera is not available, you can dump of the schema to export it, and then preform a re-run manually in the target cluster.

Scenario 2: Migrating ACID tables (HYBRID + MIGRATE_ACID)

All managed ACID source tables (applicable to HDP) are migrated to ACID tables in Cloudera using HMS Mirror in the HYBRID migrate-acid mode with the intermediate-storage option. This approach migrates both data and metadata using Hive queries, and stages the data in an intermediate-storage location on S3. This approach avoids the need to link the target cloud environment with the on-prem source.

In this scenario, HMS-Mirror is triggered from the source cluster (LEFT), as shown in the following diagram:

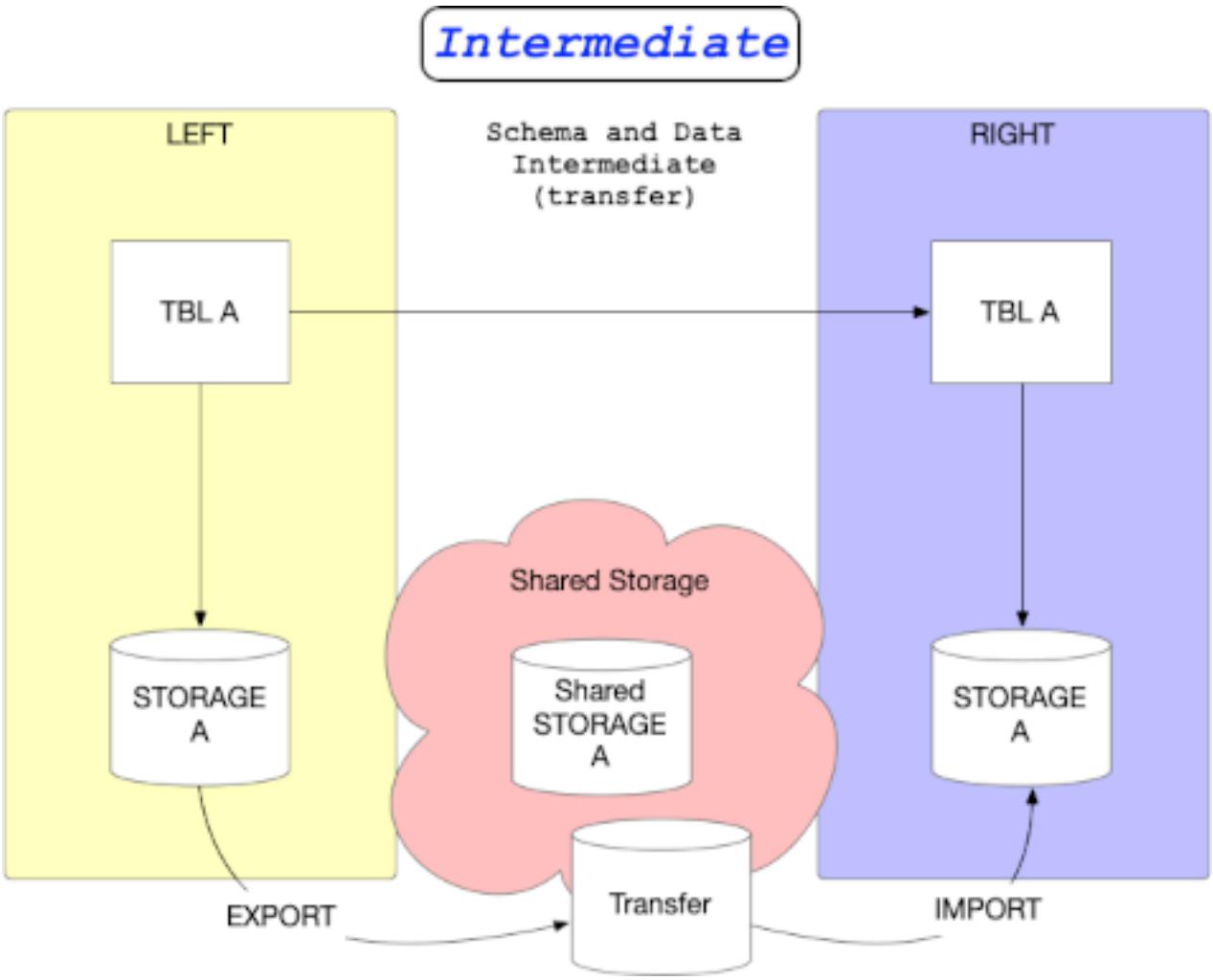

Migration prerequisites

Migrating Hive metadata to Cloudera requires setup tasks, some of which are related to security.

In addition to the step-by-step set up tasks listed below, put the SSL certificate on the HDP cluster edge node.You need to migrate Ranger policies from HDP to Cloudera, set up AVRO table migration if you have Hive tables in AVRO format. You need to setup Hive JDBC standalone JARS for connecting the HDP and Cloudera clusters.

After performing the following setup tasks, you simply run the HMS Mirror command to migrate the data.

Setting up Hive JDBC standalone JARS

You must have the Hive JDBC standalone jar compatible with your HDP version and the latest Hive JDBC standalone jar compatible with Cloudera on the edge node between the HDP and Cloudera cluster.

Procedure

1. On the HDP cluster, navigate to /usr/hdp/current/hive-client/lib check that the Apache Hive standalone JAR was installed by HDP on your cluster.
   For example, on HDP 2.6.5 you find a JAR having a name something like hive-jdbc-1.2.1000.2.6.5.5000-33-standalone.jar.

   On HDP 3.1.5, you find a JAR having a name something like hive-jdbc-3.1.0-315.133-2-standalone.jar.

   Important: Hive 3 jars do not work on an HDP 2.x cluster.

2. If the JAR is not on the cluster, using the version number of the JAR in /usr/hdp/current/hive-client/lib, search for and download the Apache Hive JDBC driver hive-jdbc from the driver archive.
3. On the Cloudera cluster, navigate to /opt/cloudera/parcels/<version>/jars/hive-jdbc-3<version>, and check that the Apache Hive standalone JAR is on the cluster.
4. If the JAR is not on the cluster, download the latest Hive JDBC driver from the Cloudera Downloads page.
5. Make a note of the path to the JAR to connect the HDP and Cloudera clusters later.

Save Hive Metastore by Dumping

You must dump the Hive metastore metadata immediately before upgrading. Dumping the metastore data to a file is critical to prevent data loss.

Procedure

1. On the node where the database you use for Hive Metastore resides, dump Hive Metastore metadata before upgrading to HDP.
   For example, in MySQL, dump each database as follows:

   ```
   mysqldump <hive_db_schema_name> > </path/to/dump_file>
   ```

   In Postgres, dump each database as follows:

   ```
   pg_dump --opt <metastore_db_name> > /path/to/dump_file
   ```

2. Proceed to upgrade HDP, assuming no Hive update, delete, or merge occurred after compaction; otherwise, repeat the compaction and Hive Metastore dump procedures, and then upgrade HDP.

Take a Mandatory Snapshot of Hive Tables

Taking a snapshot of Hive tables is mandatory before upgrading. You also need to keep track of how many tables you have before upgrading for comparison after upgrading.

Procedure

1. In Ambari, go to Services/Hive/Configs, and check the value of hive.metastore.warehouse.dir to determine the location of the Hive warehouse, /apps/hive/warehouse by default.
2. On any node in the cluster, as the HDFS superuser, enable snapshots.

   ```
   $ sudo su - hdfs
   $ hdfs dfsadmin -allowSnapshot /apps/hive/warehouse
   ```

   ```
   Allowing snapshot on /apps/hive/warehouse succeeded
   ```

3. Create a snapshot of the Hive warehouse.

   ```
   $ hdfs dfs -createSnapshot /apps/hive/warehouse
   ```

   Output includes the name and location of the snapshot.

   ```
   Created snapshot /apps/hive/warehouse/.snapshot/s20181204-164645.898
   ```

4. Start Hive as a user who has SELECT privileges on the tables.

   ```
   $ beeline beeline> !connect jdbc:hive2://
   Enter username for jdbc:hive2://: hive
   Enter password for jdbc:hive2://: *********
   ```

   ```
   Connected to: Apache Hive (version 1.2.1000.2.6.5.0-292)
   Driver: Hive JDBC (version 1.2.1000.2.6.5.0-292)
   ```

5. Identify all tables outside /apps/hive/warehouse/.

```
hive> USE my_database;
hive> SHOW TABLES;
```

6. Determine the location of each table using the DESCRIBE command. For example:

```
hive> DESCRIBE FORMATTED my_table partition (dt='20201130');
```

7. Create a snapshot of the directory shown in the location section of the output.
8. Repeat steps 5-7 for each database and its tables outside /apps/hive/warehouse/.

Setting up security

You need to turn off Kerberos and provide an authentication certificate on respective clusters before running HMS Mirror.

Procedure

1. Copy the cert.txt file to all the hosts in the destination Cloudera cluster.

```
$ mv cert.txt othercert.txt
$ scp othercert.txt <path on CDP SaaS>
```

2. Import the certificate into the keystore file on all the hosts of the destination cluster.

```
/usr/java/default/bin/keytool -importcert -noprompt -v -trustcacerts -ke
ystore /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.
jks -alias cmrootca-1 -file ./othercert.txt --storepass [***truststore_p
assword***]
```

Installing and configuring HMS Mirror

You can perform the migration when HMS Mirror is installed on the HDP cluster.

About this task

You download, install, and configure HMS Mirror software. Configuration consists of generating and customizing an HMS Mirror config file in YAML format. For example, you set the path of the hcfsNamespace, which is the HDFS file system path.

Before you begin

- Remove any previous installation of HMS Mirror. For example, as root run the following command. rm -f /usr/local/hms-mirror/lib/*.jar. For more information, see setup.sh generated by a dry run of HMS Mirror.>>
- On the HDP cluster, from the Ambari configuration in /etc/ambari-server conf, get the hcfsNamespace.
- On the Cloudera cluster, get the path to S3 object storage.
- On the Cloudera cluster, get the HiveServer (HS2) URI from the SaaS interface.
- On the Cloudera cluster, get the Java KeyStore (jks) file name and path to set the sslTrustStore.

Procedure

1. On the source HDP cluster, go to https://github.com/cloudera-labs/hms-mirror.
2. Download the HMS Mirror distribution, and extract the files.
3. Go to a directory other than the HMS Mirror installation directory.

   You should not run HMS Mirror from the directory where you installed the software.

4. Generate a config file .hms-mirror/cfg/default.yaml using the -setup option.

```
hms-mirror -setup
```

The config file is generated.

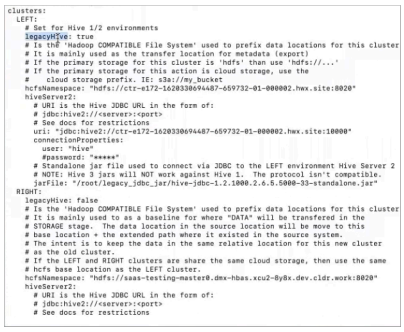

```
clusters:
  LEFT:
    # Set for Hive 1/2 environments
    legacyHive: true
    # Is the 'Hadoop COMPATIBLE File System' used to prefix data locations for this cluster
    # It is mainly used as the transfer location for metadata (export)
    # If the primary storage for this cluster is 'hdfs' than use 'hdfs://...'
    # If the primary storage for this action is cloud storage, use the
    #    cloud storage prefix. IE: s3a://my_bucket
    hcfsNamespace: "hdfs://ctr-e172-1620330694487-659732-01-000002.hwx.site:8020"
    hiveServer2:
      # URI is the Hive JDBC URL in the form of:
      # jdbc:hive2://<server>:<port>
      # See docs for restrictions
      uri: "jdbc:hive2://ctr-e172-1620330694487-659732-01-000002.hwx.site:10000"
      connectionProperties:
        user: "hive"
        #password: "*****"
      # Standalone jar file used to connect via JDBC to the LEFT environment Hive Server 2
      # NOTE: Hive 3 jars will NOT work against Hive 1.  The protocol isn't compatible.
      jarFile: "/root/legacy_jdbc_jar/hive-jdbc-1.2.1000.2.6.5.5000-33-standalone.jar"
  RIGHT:
    legacyHive: false
    # Is the 'Hadoop COMPATIBLE File System' used to prefix data locations for this cluster
    # It is mainly used to as a baseline for where "DATA" will be transfered in the
    # STORAGE stage.  The data location in the source location will be move to this
    # base location + the extended path where it existed in the source system.
    # The intent is to keep the data in the same relative location for this new cluster
    # as the old cluster.
    # If the LEFT and RIGHT clusters are share the same cloud storage, then use the same
    # hcfs base location as the LEFT cluster.
    hcfsNamespace: "hdfs://saas-testing-master0.dmx-hbas.xcu2-8y8x.dev.cldr.work:8020"
    hiveServer2:
      # URI is the Hive JDBC URL in the form of:
      # jdbc:hive2://<server>:<port>
      # See docs for restrictions
```

5. Scroll to the cluster information about both LEFT (HDP) and RIGHT (Cloudera) clusters at the bottom of the file.
6. Enter Boolean values for legacyHive to identify the source cluster.
   For example:
   - On the LEFT (HDP) cluster, set legacyHive: true
   - On the RIGHT (Cloudera) cluster, you set legacyHive: false
7. On the RIGHT cluster, enter the hostname and port for S3 object storage.
   For example: S3a://bucket_name/path
8. Enter the HiveServer (HS2) server name and port for the jdbc url of the LEFT source HDP cluster.
   For example, jdbc:hive2://<hs2 server name>: 10000
9. Enter the HiveServer URI of the RIGHT destination Cloudera cluster, including the sslTrustStore path and trustStorePassword.
   The URI looks something like this:

```
"jdbc:hive2://saas/testing-master0.dmx:443/;ssl=true;transportMode=http:
httpPath=saas-testing/cdp-proxy-token/hive;sslTrustStore=/root/cert/gate
way-client-trust.jks;trustStorePassword=changeit;"
```

10. Save changes to the YAML config.

Sample YAML configuration file

You customize a YAML file generated by the HMS Mirror dry run for your migration by looking at an example YAML.

```
clusters:
  LEFT:
    # Set for Hive 1/2 environments
    legacyHive: false
    # Is the 'Hadoop COMPATIBLE File System' used to prefix data locations f
or this cluster.
    # It is mainly used as the transfer location for metadata (export)
    # If the primary storage for this cluster is 'hdfs' than use 'hdfs://...
'
    # If the primary storage for this action is cloud storage, use the
    #    cloud storage prefix. IE: s3a://my_bucket
    hcfsNamespace: "hdfs://hdp_test:8020"
    hiveServer2:
      # URI is the Hive JDBC URL in the form of:
      # jdbc:hive2://<server>:<port>
      # See docs for restrictions
      uri: "jdbc:hive2://<Hive Server2 Host>:10000"
      connectionProperties:
        user: "hive"
```

```
        password: "hive"
      # Standalone jar file used to connect via JDBC to the LEFT environment
 Hive Server 2
      # NOTE: Hive 3 jars will NOT work against Hive 1.  The protocol isn't
compatible.
      jarFile: "/root/.hms-mirror/aux_libs/hive-jdbc-3.1.0.3.1.5.6091-7-stan
dalone.jar"
  RIGHT:
    legacyHive: false
    # Is the 'Hadoop COMPATIBLE File System' used to prefix data locations
 for this cluster.
    # It is mainly used to as a baseline for where "DATA" will be transfered
 in the
```

Testing the YAML and the cluster connection

It makes sense to test the YAML and cluster connection before starting the migration using HMS Mirror. Connection troubleshooting is easier before, rather than after, running HMS Mirror.

Procedure

1. Using a Hive client, such as DBeaver, beeline, or Tableau, connect the left cluster to the right cluster, and then vice versa.
2. If you use ZooKeeper Service Discovery on the HDP source cluster, use a plain URL in your connection string. For example:jdbc:hive2://<hive-server2-hostname>:10000/

HMS Mirror command summary

There are several critical options you need to use with the hms-mirror command to migrate HDP to Cloudera migration.

Run the following HMS Mirror help command, or go to the Cloudera HMS Mirror github site, to see a description of all command options:

```
hms-mirror -h
```

For the HDP to Cloudera migration, you generally use the subset of options shown in the following command:

```
hms-mirror -db <database name> -ma -mnn -e
```

-asm

Migrate avro based schema tables.

-d,--data-strategy <strategy>

Specify how the data will follow the schema. [SCHEMA_ONLY, LINKED, SQL, EXPORT_IMPORT, HYBRID, CONVERT_LINKED, STORAGE_MIGRATION, COMMON]

-da, -downgrade-acid

Downgrade ACID tables to EXTERNAL tables with purge.

-db [<db1,db2,..>]

Comma separated list of databases (up to 100) where dbn is the database name. The database name is optional, only required if the database value is null in the YAML configuration file.

-dc, --distcp

Generate the distcp commands.

-e, --execute

Executes HMS Mirror. Omit for a dry run.

--hadoop-classpath

Required for migrating from a Kerberos-enabled HDP cluster. For more information, see HMS Mirror documentation.

-ma | --migrate-acid

Manage managed tables, translating source metadata to the destination.

-mao

Migrate only acid tables.

-mnn

Manage non-native tables, such as HBase/Kafka/JDBC backed hive tables, translating source metadata to the destination.

-mnno

Migrate only non-native tables.

-o <output directory>

Output directory location.

-s, --sync

Drop metadata on the destination Cloudera cluster if it does not match the definition on the HDP source cluster. (Ideally, there should not be any data on the destination cluster before metadata migration.)

--setup

Creates a default.yaml file in the $HOME/.hms-mirror/cfg/ directory by prompting you for information about your migration. You customize the YAML manually for your migration.

-v | --views-only

Migrates only VIEWs between two environments. This option alone must be run to separately generate views.

You can run any combination of options, except -v, which only migrates views. Run HMS Mirror to migrate views separately. VIEW creation requires dependent tables to exist. Run hms-mirror to create all the target tables before running it with the -v option.

This flag is an OR for processing views OR tables. They are NOT processed together.

Requirements:

- The dependent tables must exist in the RIGHT cluster
- When using -dbp|--db-prefix option, VIEW definitions are not modified and will most likely cause VIEW creation to fail.

Migrating Hive metadata

You first perform a dry run of HMS Mirror, and check the output. You see how to execute HMS Mirror to migrate the Hive metadata in HMS on the HDP cluster to the Cloudera as a Service cluster.

About this task

The dry run of HMS Mirror generates scripts, action files, reports, and a blueprint of what migration will take place.

Tip: To bypass Ranger security checks, which speeds up the migration, in Cloudera Manager, set ranger.plugin.hive.urlauth.filesystem.schemes=file in Cloudera Manager HiveServer (HS2) for Hive on TEZ.

In this procedure, you run the HMS Mirror command as shown in the step 6. In summary, use the Migrate ACID flag option -ma to migrate Hive ACID tables. Alternatively, you can migrate the data by specifying the HYBRID data strategy. An HMS Mirror command that sets the intermediate storage flag stages data in S3, as required for transfers in the public cloud. The target Cloudera SaaS environment uses the S3 object store in the cloud instead of the on-prem HDFS because the target does not have direct access to the on-prem HDFS.

Before you begin

- If you are going to migrate non-native Hive tables, such as HBase, Kafka, or JDBC-backed tables, make these services available on the Cloudera cluster; otherwise metadata migration does not work.
- If you have many databases to migrate, separate them into batches of 100 databases, or fewer. HMS Mirror does not accept a larger batch.
- A batching strategy is recommended. Inspect the number of tables/partitions within each database and develop a general criterion for batching.

Procedure

1. Stop all workloads on the source HDP and destination Cloudera clusters.
2. Run HMS Mirror to perform a dry run by omitting the -e option.

```
hms-mirror -db migrate_db -mnn
```

The output looks something like this:

```
================================================================================
TBLS (Started/Completed/Errors)->(0/7/0)          Elapse Time: 3secs

Run Mode                  : DRY-RUN
Config file               : /root/.hms-mirror/cfg/default.yaml

Databases(<db>):
migrate_db

Report file               : /root/.hms-mirror/reports/2022-03-16_19-09-08/<db>_hms-mirror.md|html
LEFT Execute file         : /root/.hms-mirror/reports/2022-03-16_19-09-08/<db>_LEFT_execute.sql
RIGHT Execute file        : /root/.hms-mirror/reports/2022-03-16_19-09-08/<db>_RIGHT_execute.sql
LEFT Action file          : /root/.hms-mirror/reports/2022-03-16_19-09-08/<db>_LEFT_action.sql
RIGHT Action file         : /root/.hms-mirror/reports/2022-03-16_19-09-08/<db>_RIGHT_action.sql
[root@ctr-e172-1620308694487-659732-01-000013 ~]#
```

3. List the generated reports.
For example,

```
ls /root/.hms-mirror/reports/2022-03-16_19-09-08/
```

```
20308694487-659732-01-000013 ~]# ls /root/.hms-mirror/reports/2022-03-16_19-09-08/
  migrate_db_1_distcp_source.txt  migrate_db_hms-mirror.md    migrate_db_LEFT_execute.sql  migrate_db_RIGHT_execute.sql
#  migrate_db_hms-mirror.html      migrate_db_LEFT_action.sql  migrate_db_RIGHT_action.sql
20308694487-659732-01-000013 ~]#
```

4. Open the HMS Mirror report in the migrate_db_hms-mirror.html file, and scroll down to see the DB Create Statement.

DB Create Statement

```
CREATE DATABASE IF NOT EXISTS migrate_db
LOCATION "hdfs://saas-testing-master0.dmx-hbas.xcu2-8y8x.dev.cldr.work:8020/apps/hive/warehouse/migrate_db.db"
```

DB Issues

none

Reports follow the DB Create Statement.

5. Scroll down to Skipped Tables/Views.

managed_table_3	SCHEMA_ONLY	SUCCESS	.01	2	.00	init
					1.83	LEFT Fetched Schema
					.63	RIGHT No Schema
					.03	TRANSFER SCHEMA_ONLY
					.01	Definitions Built
					.00	SQL Built

Skipped Tables/Views

Table / View	Reason
managed_table_2	ACID table and ACID processing not selected (-ma
managed_view	This is a VIEW and VIEW processing wasn't selected.
storage_handler_based_table	This is a Non-Native hive table and non-native process wasn't selected.

Your managed tables will not be listed in Skipped Tables/Views, if you omit the -ma option, for example.

6. Run the HMS Mirror command to execute the migration (-e option) of the migrate_db database.
   For example:
   - Migrating standard

     hms-mirror -db migrate_db -e
   - Migrating ACID tables

     ```
     hms-mirror -db migrate_db -ma -is s3a://intermediate_storage_location -e
      -d HYBRID
     ```
   - Migrating non-native tables (HBase, Kafka, JDBC)

     ```
     hms-mirror -db migrate_db -mnn -e
     ```

   Note: The -d HYBRID migrates the data). The intermediate data is staged in S3 via hms-mirror, required in a public cloud transfer scenario because the target CDP SaaS env does not have direct access to the on-prem HDFS).
7. At the prompt to confirm that you backed up the Hive Metastore (HMS) in both clusters, enter TRUE.

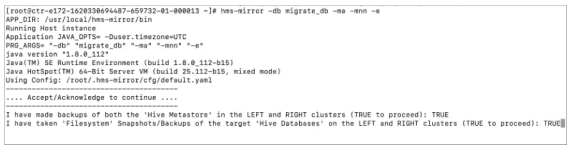

```
[root@ctr-e172-1620380694487-659732-01-000013 ~]# hms-mirror -db migrate_db -ma -mnn -e
APP_DIR: /usr/local/hms-mirror/bin
Running Host instance
Application JAVA_OPTS= -Duser.timezone=UTC
PRG_ARGS= "-db" "migrate_db" "-ma" "-mnn" "-e"
java version "1.8.0_112"
Java(TM) SE Runtime Environment (build 1.8.0_112-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.112-b15, mixed mode)
Using Config: /root/.hms-mirror/cfg/default.yaml
----------------------------------------
.... Accept/Acknowledge to continue ....
----------------------------------------
I have made backups of both the 'Hive Metastore' in the LEFT and RIGHT clusters (TRUE to proceed): TRUE
I have taken 'Filesystem' Snapshots/Backups of the target 'Hive Databases' on the LEFT and RIGHT clusters (TRUE to proceed): TRUE
```

8. At the prompt to confirm that you configured the file system TRASH, enter TRUE.

   HMS Mirror generates reports in the directory shown in the output.

   The metadata migration process completes.

HMS Mirror generated files

You use generated scripts, reports, and other files as you migrate the table metadata and data. Reports generated by HMS Mirror include information about HMS Mirror configurations. A list of tables on left HDP and translated tables on right Cloudera cluster are also generated.

The following files are generated by a dry run of HMS Mirror:

- <dbname>_1_distcp_source.txt

  A list of source cluster table file locations.
- <dbname>_LEFT_execute.sql

  A list of commands for creation of databases and tables under the database for the HDP cluster.
- <dbname>_LEFT_action.sql

  Actions that may be required on the HDP cluster to achieve the optimum state for migration.
- <dbname>_RIGHT_execute.sql

  A list of commands for creation of databases and tables under the database for the Cloudera SaaS cluster.
- <dbname>_RIGHT_action.sql

  Actions that may be required on the Cloudera SaaS cluster to achieve the optimum state for migration.
- <dbname>_hms-mirror.md Options you use for the hms-mirror command, config yaml, database and tables present on left cluster, translated tables on right cluster in an md file. You can use any markdown file viewer to view this file.
- <dbname>_hms-mirror.html — This is the html version of the above md file.
- distcp_script.sh

  The distcp commands needed to migrate actual table data from HDP to S3 location. Moving HDFS data from the HDP to the S3 object store for the Cloudera cluster is fully documented in Migrating data from HDP to Cloudera on page 39.

- distcp_workbook.md

  Shows the HDP source and Cloudera destination locations that we need to migrate to.

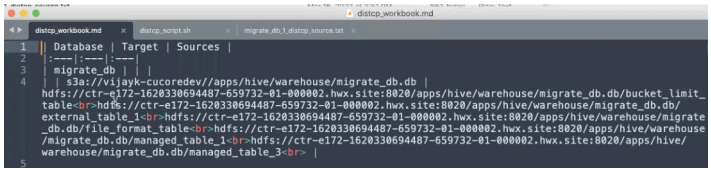

  This information can help you build a transfer job to specify multiple sources using distcp -f.

Verifying metadata migration

After running the HMS Mirror tool to migrate Hive metadata from HDP to Cloudera, you take a look at the migrated metadata on the Cloudera cluster.

About this task

Procedure

1. From a JDBC client, such as DBeaver, create a connection to the Cloudera cluster.
2. Generate a passcode token to connect to the Cloudera cluster.
3. Run SQL queries and check the output to verify that the databases and tables are migrated.
   For example:

```
SHOW DATABASES;
USE migrate_db;
SHOW TABLES;
```

Migrating actual Hive data

Although HMS Mirror migrates only Hive metastore (HMS) metadata, not the actual Hive data, the tool generates a script for running the Hadoop DistCp tool. Using DistCp, you migrate actual data from HDP to Cloudera.

About this task

DistCp is fully documented in HDP to Cloudera SaaS HDFS Migration. When you run the HMS Mirror tool, you generated a number of distcp files:

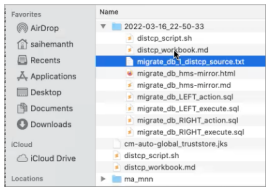

The distcp_script.sh contains a template and instructions for running the script to migrate the actual Hive data. The template looks something like this:

```
hadoop distcp ${DISTCP_OPTS} -f ${HCFS_BASE_DIR}/migrate_db_1_distcp_source.
txt s3a://myserver//apps/hive/warehouse/migrate_db.db
```

In this task, you customize the DistCp command to substitute values specific to your migration for the variables above.

Before you begin

Give the user who is migrating the data permissions to read/write/execute on both the HDP and CDP clusters.

Procedure

1. From the information in the distcp_workbook.md file, get the names of source files from the Sources column to migrate from the <dbname>_1_distcp_source.txt
2. On the source HDP cluster, export the environment variable HCFS_BASE_DIR that represents the path to the source files.

   ${HCFS_BASE_DIR} must be available to the user running DistCp.
3. Export the DISTCP_OPTS environment variable to customize job settings.
   For example, you might adjust memory settings for large jobs.
4. Using information from steps 1-3 plus your S3 access key and password, customize the DistCp template.
5. On the HDP source cluster edge node, run the customized DistCp commands.
   For example:

```
hadoop distcp -Dfs.s3a.access.key=AKIAR2YKDFBDE64CJ5XG -Dfs.s3a.secret.k
ey=<mykey> hdfs://ctr-e172-1620330694487-659732-01-000002.hwx.site:8020/
apps/hive/warehouse/migrate_db.db/managed_table_1  s3a://myserver//apps/
hive/warehouse/migrate_db.db
hadoop distcp -Dfs.s3a.access.key=AKIAR2YKDFBDE64CJ5XG -Dfs.s3a.secret.
key=<mykey> hdfs://ctr-e172-1620330694487-659732-01-000002.hwx.site:8020/
apps/hive/warehouse/migrate_db.db/file_format_table  s3a://myserver//apps/
hive/warehouse/migrate_db.db
hadoop distcp -Dfs.s3a.access.key=AKIAR2YKDFBDE64CJ5XG -Dfs.s3a.secret.k
ey=<mykey> hdfs://ctr-e172-1620330694487-659732-01-000002.hwx.site:8020/
apps/hive/warehouse/migrate_db.db/external_table_1  s3a:/myserver//apps/
hive/warehouse/migrate_db.db
```

6. Check the status of the migration in HTML reports generated by DistCP to determine success or failure.
7. If an error occurs, check $HOME/.hms-mirror/logs/hms-mirror.log.

Adjust AVRO table schema URLs

If any of your Hive tables are in AVRO format, before migration already created the AVRO schema on the Cloudera object store. After migration, you must change the URLs in table properties to point to the Cloudera location.

Procedure

On the destination Cloudera cluster, correct the path of AVRO schema URL table property.

```
ALTER TABLE my_table SET TBLPROPERTIES('avro.schema.url'='S3a:///mybucket/my
path/my_table.avsc');
```

Verifying actual Hive data migration

After running the DistCp to migrate Hive data from HDP to Cloudera, you take a look at the migrated data on the Cloudera cluster.

Procedure

1. From a JDBC client, such as DBeaver, create a connection to the Cloudera cluster.
2. Generate a passcode token to connect to the Cloudera cluster.

3. Run SQL SELECT queries and check the output to verify that the external table data is migrated.
   For example:

```
SELECT * FROM migrate_db.external_table_1;
```

```
1 row selected (0.442 seconds)
0: jdbc:hive2://ctr-e172-1620338694487-659732> select * from  migrate_db.external_table_1;
+-----------------------+---+
| external_table_1.i  |
+-----------------------+---+
| 5                     |
| 1                     |
| 2                     |
| 3                     |
| 4                     |
+-----------------------+---+
5 rows selected (0.736 seconds)
```

4. Run SELECT queries and check the output to verify that managed data is migrated.

```
SELECT * FROM migrate_db.managed_table_3;
```

Table locations

After migration from HDP 2.x, the relative paths for databases, tables, and partitions on the destination Cloudera cluster are different from HDP 2.x.

- HDP 2.x managed table location: /apps/hive/warehouse
- HDP 2.x external table location: /user/hive/warehouse”

After migration from HDP 3.x, the relative paths for databases, tables, partitions on the destination Cloudera cluster are the same as the HDP 3.x:

- HDP 3.x and Cloudera managed table location: /warehouse/tablespace/managed/hive
- HDP 3.x and Cloudera external table location: /warehouse/tablespace/external/hive

Fixing statistics

Upgrading or migrating from Hive 1 or Hive 2 to Hive 3 might result in missing statistics. In Hive 3, these missing statistics, when detected by the cost-based optimizer (CBO), could cause datasets to be disregarded. As Data Engineer, you need to fix these statistics after upgrading.

Procedure

1. Run DESCRIBE FORMATTED <table>, and check the value of numrows.
   If the value is 0, you must fix statistics.
2. Run ANALYZE on the tables and columns to fix the statistics.

```
ANALYZE TABLE credit_card_01.cc_acct COMPUTE STATISTICS[FOR COLUMNS];
```

Changes to HDP Hive tables

As a Data Scientist, Architect, Analyst, or other Hive user you need to locate and use your Apache Hive 3 tables after an upgrade. You also need to understand the changes that occur during the upgrade process.

Managed, ACID tables that are not owned by the hive user remain managed tables after the upgrade, but hive becomes the owner.

After the upgrade, the format of a Hive table is the same as before the upgrade. For example, native or non-native tables remain native or non-native, respectively.

After the upgrade, the location of managed tables or partitions do not change under any one of the following conditions:

- The old table or partition directory was not in its default location /apps/hive/warehouse before the upgrade.
- The old table or partition is in a different file system than the new warehouse directory.
- The old table or partition directory is in a different encryption zone than the new warehouse directory.

Otherwise, the upgrade process from HDP to Cloudera moves managed files to the Hive warehouse /warehouse/table space/managed/hive. The upgrade process carries the external files over to Cloudera with no change in location. By default, Hive places any new external tables you create in /warehouse/tablespace/external/hive. The upgrade process sets the hive.metastore.warehouse.dir property to this location, designating it the Hive warehouse location.

Changes to table references using dot notation

Upgrading to Cloudera includes the Hive-16907 bug fix, which rejects `db.table` in SQL queries. The dot (.) is not allowed in table names. To reference the database and table in a table name, both must be enclosed in backticks as follows: `db`.`table`.

Changes to ACID properties

Hive 3.x in Cloudera Private Cloud Base supports transactional and non-transactional tables. Transactional tables have atomic, consistent, isolation, and durable (ACID) properties. In Hive 2.x, the initial version of ACID transaction processing was ACID v1. In Hive 3.x, the mature version of ACID is ACID v2, which is the default table type in Cloudera Private Cloud Base.

Native and non-native storage formats

Storage formats are a factor in upgrade changes to table types. Hive 2.x and 3.x support the following native and non-native storage formats:

- Native: Tables with built-in support in Hive, such as those in the following file formats:
  - Text
  - Sequence File
  - RC File
  - AVRO File
  - ORC File
  - Parquet File
- Non-native: Tables that use a storage handler, such as the DruidStorageHandler or HBaseStorageHandler

Cloudera upgrade changes to HDP table types

The following table compares Hive table types and ACID operations before an upgrade from HDP 2.x and after an upgrade to Cloudera. The ownership of the Hive table file is a factor in determining table types and ACID operations after the upgrade.

Table 1: HDP 2.x and Cloudera Table Type Comparison

HDP 2.x				CDP	
Table Type	**ACID v1**	**Format**	**Owner (user) of Hive Table File**	**Table Type**	**ACID v2**
External	No	Native or non-native	hive or non-hive	External	No
Managed	Yes	ORC	hive or non-hive	Managed, updatable	Yes
Managed	No	ORC	hive	Managed, updatable	Yes
			non-hive	External, with data delete	No
Managed	No	Native (but non-ORC)	hive	Managed, insert only	Yes
			non-hive	External, with data delete	No
Managed	No	Non-native	hive or non-hive	External, with data delete	No

Migrating Hive and Impala Workloads to CDP

Date published: 2023-08-14
Date modified: 2023-08-18

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Migrating Hive and Impala workloads to Cloudera.....4
Handling prerequisites.....4
Hive 1 and 2 to Hive 3 changes.....7
Reserved keywords.....7
Spark-client JAR requires prefix.....7
Hive warehouse directory.....8
Replace Hive CLI with Beeline.....8
PARTIALSCAN.....8
Concatenation of an external table.....9
INSERT OVERWRITE.....9
Managed to external table.....10
Property changes affecting ordered or sorted subqueries and views.....10
Runtime configuration changes.....11
Prepare Hive tables for migration.....12
Impala changes from CDH to Cloudera.....15
Impala configuration differences in CDH and Cloudera.....16
Additional documentation.....16

Migrating Hive and Impala workloads to Cloudera

You learn how to accelerate the migration process, to refactor Hive applications, and to handle semantic changes from Hive 1/2 to Hive 3. You get pointers to Impala documentation about workload migration and application refactoring.

Handling prerequisites

You must perform a number of tasks before refactoring workloads.

You need to handle the following prerequisites:

- Identify all the Hive 1/2 workloads in the cluster (CDH 5.x, CDH 6.x, HDP 2.x, HDP 3.x). (This document assumes all the identified workloads are in working condition).
- Prepare tables for migration.

  To prepare the tables for migration, use the Hive SRE Tool which is a Cloudera Lab tool that scans your Hive Metastore and HDFS to identify common upgrade problems that can cause the Cloudera upgrade to fail. The tool provides guidance for fixing those problems before migrating the tables to Cloudera. This guidance is provided through reports that the administrator must take action on. The tool does not take corrective action itself.

  Cloudera strongly recommends running the Hive SRE Tool to aid in this pre-upgrade HMS healthcheck. If you do not run it, you must manually investigate your HMS for the following types of problems.

  Refer to Hive SRE Tooling for tool setup, tool execution & interpretation of output using Hive SRE Tool on Hive Metadata.

  The general transition from Hive 1 and 2 to Hive 3 includes the following types of HMS operations. The Hive SRE Tool performs the equivalent types of checks in an automated fashion. Please review these from the Cloudera documentation site (CDH 5, CDH 6, HDP 2, HDP 3).

The following table lists Hive table conditions, the impact of the conditions, and distributions affected. In this table, XXX stands for some value/count. This value is the number of affected paths you get from the Hive-SRE tool output reports. For each type of condition listed in the table, there is some number of paths/tables affected.

Condition	Impact	Distributions Affected
Bad Filenames	Tables that would be converted from a Managed Non-Acid table to an ACID transactional table require the files to match a certain pattern. This process will scan the potential directories of these tables for bad filename patterns. When located, it will indicate which tables/partitions have file naming conventions that would prevent a successful conversion to ACID. The best and easiest way to correct these file names is to use HiveSQL to rewrite the contents of the table/partition with a simple 'INSERT OVERWRITE TABLE xxx SELECT * FROM xxx'. This type of statement will replace the current bad filenames with valid file names by rewriting the contents in HiveSQL. There are approximately XXX paths that may need remediation. The list of paths can be found in the output of the Hive assessment report in the u3/bad_filenames_for_orc_conversion.md.	HDP 2

Condition	Impact	Distributions Affected
Missing Directories	Missing Directories cause the upgrade conversion process to fail. This inconsistency is an indicator that data was removed from the file system, but the Hive MetaStore was not updated to reflect that operation. An example of this is deleting a partition in HDFS, without dropping the partition in Hive. There are XXX affected paths that need remediation. The list of directories can be found in the output of the Hive assessment report, in the file u3/loc_scan_missing_dirs.md.	CDH5, CDH6, HDP2, HDP3
Managed Shadow Tables	In Hive 3, Managed tables are 'ACID' tables. Sharing a location between two 'ACID' tables will cause compaction issues and data issues. These need to be resolved before the upgrade. There are XXX affected tables that need remediation. The list of paths can be found in the output of the Hive assessment report, in the file u3/hms_checks.md	CDH5, CDH6, HDP2, HDP3
Managed Table Migrations	This process will list tables that will and 'could' be migrated to "Managed ACID" tables during the upgrade process. Tables used directly by Spark or if data is managed by a separate process that interacts with the FileSystem, you may experience issues post-upgrade. Recommended: Consider converting these tables to external tables. There are XXX affected tables that may need remediation. The list of tables can be found in the output of the Hive assessment report in the file managed_upgrade_2_acid.sql.	CDH5, CDH6, HDP2
Compaction Check	In the upgrade to Hive 3, ACID tables must be compacted prior to initiating the upgrade. XXX tables were noted as requiring compaction. Because CDH does not support Hive ACID tables, this may be a leftover from the previous HDP to CDH migration that the customer implemented. This should be investigated further. The affected table is ers_stage_tls.test_delete	CDH5, CDH6, HDP2, HDP3
Unknown SerDe Jars	A list tables using SerDe’s that are not standard to the platform appears. Review the list of SerDes and verify they are still necessary and available for Cloudera. There are approximately XXX tables configured with 3rd party SerDes.	CDH5, CDH6, HDP2

Condition	Impact	Distributions Affected
Remove transactional=false from Table Properties	In CDH 5.x it is possible to create tables with the property transactional=false set. While this is a no-op setting, if any of your Hive tables explicitly set this, the upgrade process fails. You must remove 'transactional'='false' from any tables you want to upgrade from CDH 5.x to Cloudera. Alter the table as follows: ALTER TABLE my_table UNSET TBLPROPERTIES ('transactional');	CDH5, CDH6
Make Tables SparkSQL Compatible	Non-Acid, managed tables in ORC or in a Hive Native (but non-ORC) format that are owned by the POSIX user hive will not be SparkSQL compatible after the upgrade unless you perform manual conversions. If your table is a managed, non-ACID table, you can convert it to an external table using this procedure (recommended). After the upgrade, you can easily convert the external table to an ACID table, and then use the Hive Warehouse Connector to access the ACID table from Spark. Take one of the following actions. • Convert the tables to external Hive tables before the upgrade. ALTER TABLE ... SET TBLPROPERTIES('EXTERNAL'='TRUE','external.table.purge'='true') • Change the POSIX ownership to an owner other than hive. You will need to convert managed, ACID v1 tables to external tables after the upgrade.	CDH5, CDH6, HDP2
Legacy Kudu Serde Report	Early versions of Hive/Impala tables using Kudu were built before Kudu became an Apache Project. After Kudu became an Apache Project, the base Kudu Storage Handler classname changed. This report locates and reports on tables using the legacy storage handler class.	CDH5, CDH6, HDP2, HDP3
Legacy Decimal Scale and Precision Check	When the DECIMAL data type was first introduced in Hive 1, it did NOT include a Scale or Precision element. This causes issues in later integration with Hive and Spark. You need to identify and suggest corrective action for tables where this condition exists.	CDH5, CDH6, HDP2, HDP3
Database / Table and Partition Counts	Use this to understand the scope of what is in the metastore.	CDH5, CDH6, HDP2, HDP3
Small Files, Table Volumes, Empty Datasets	Identify and fix these details to clean up unwanted datasets in the cluster which would speed up the Hive upgrade process.	CDH5, CDH6, HDP2, HDP3

Condition	Impact	Distributions Affected
Merge Independent Hive and Spark Catalogs	In HDP 3.0 - 3.1.4, Spark and Hive use independent catalogs for accessing tables created using SparkSQL or Hive tables. A table created from Spark resides in the Spark catalog. A table created from Hive resides in the Hive catalog. Databases fall under the catalog namespace, similar to how tables belong to a database namespace. In HPD 3.1.5, Spark and Hive share a catalog in Hive metastore (HMS) instead of using separate catalogs. The Apache Hive schematool in HDP 3.1.5 and Cloudera releases supports the mergeCatalog task.	HDP3

Hive 1 and 2 to Hive 3 changes

A description of the change, the type of change, and the required refactoring provide the information you need for migrating from Hive 1 or 2 to Hive 3.

In addition to the topics in this section that describe Hive changes, see the following documentation about changes applicable to CDH and HDP to prepare the workloads.

- Hive Configuration Changes Requiring Consent
- Hive unsupported interfaces and features

Reserved keywords

Reserved words (also called keywords) have a predefined meaning and syntax in Hive. These keywords have to be used to develop programming instructions. Reserved words cannot be used as identifiers for other programming elements, such as the name of variable or function.

Hive 1 and 2

TIME, NUMERIC, SYNC are not reserved keywords.

Hive 3

TIME, NUMERIC, SYNC are reserved keywords.

Action Required

Reserved keywords are permitted as identifiers if you quote them. As an example, if in scripts there are identifiers like TIME, NUMERIC, SYNC use backtick `` to quote it like, `TIME`, `NUMERIC`, `SYNC`.

Distribution Affected

CDH5, CDH6, HDP2

Spark-client JAR requires prefix

The Spark-client JAR should be prefixed with hive- to make sure the jar name is consistent across all Hive jars.

Hive 1 and 2

You reference spark-client jar as spark-client.

Hive 3

You reference the spark-client jar as hive-spark-client.

Action Required

Change references as described above.

Hive warehouse directory

Hive stores table files by default in the Hive Warehouse directory.

Hive 1 and 2

The warehouse directory path is /user/hive/warehouse (CDH) or /apps/hive/warehouse (HDP)

Hive 3

hte warehouse directory path for a managed table is /warehouse/tablespace/managed/hive and the default location for external table is /warehouse/tablespace/external/hive.

Action Required

As ACID tables reside by default in /warehouse/tablespace/managed/hive and only the Hive service can own and interact with files in this directory, applications accessing the managed tables directly need to change their behavior in Hive3.

As an example, consider the changes required for Spark application. As shown in the table below, if during upgrade Hive1/2 managed tables gets converted to an external table in Hive3, no refactoring is required for Spark application. If the Hive1/2 managed table gets converted to the managed table in Hive3, Spark application needs to refactor to use Hive Warehouse Connector (HWC) or the HWC Spark Direct Reader.

Hive Table before upgrade in Hive1/2	Hive Table after upgrade in Hive3	Spark Refactoring
Managed	External	None
Managed	Managed	Use HWC or HWC Spark Direct Reader
External	External	None

Distribution Affected

CDH5, CDH6, HDP2

Replace Hive CLI with Beeline

Beeline is the new default command line SQL interface for Hive3.

Hive 1 and 2

Hive CLI is deprecated.

Hive 3

The original hive CLI is removed and becomes a wrapper around beeline.

Action Required

Remove, replace and test all hive CLI calls with the beeline command line statements.

For more information, see Converting Hive CLI scripts to Beeline.

Distribution Affected

CDH5, CDH6, HDP2, HDP3

PARTIALSCAN

You need to identify and apply configuration-level changes, including removing the PARTIALSCAN option

Hive 1 and 2

ANALYZE TABLE ... COMPUTE STATISTICS supports the PARTIALSCAN option.

Hive 3

ANALYZE TABLE ... COMPUTE STATISTICS does not support PARTIALSCAN, which is retired and throws error.

For example:

```
ANALYZE TABLE test_groupby COMPUTE STATISTICS PARTIALSCAN;
Error: Error while compiling statement: FAILED: ParseException line 1:46
extraneous input 'PARTIALSCAN' expecting EOF near '<EOF>' (state=42000,code=
40000)
```

Action Required

Remove statements containing ALTER TABLE ... COMPUTE STATISTICS PARTIALSCAN. For example:

```
ANALYZE TABLE test_groupby COMPUTE STATISTICS PARTIALSCAN;
Error: Error while compiling statement: FAILED: ParseException line 1:46
extraneous input 'PARTIALSCAN' expecting EOF near '<EOF>' (state=42000,code=
40000)
```

Distribution Affected

CDH5, CDH6, HDP2.

Concatenation of an external table

If the table or partition contains many small RCFiles or ORC files, then ALTER TABLE table_name [PARTITION (partition_key = 'partition_value' [, ...])] CONCATENATE will merge them into larger files. In the case of RCFile the merge happens at block level whereas for ORC files the merge happens at stripe level thereby avoiding the overhead of decompressing and decoding the data

Hive 1 and 2

You can concatenate an external table. For example:

```
ALTER TABLE table_name [PARTITION (partition_key = 'partition_value' [, ...]
)] CONCATENATE
```

Hive 3

Concatenation of an external table using CONCATENATE is not supported. For example, you get the following error:

```
alter table t6 concatenate;
Error: Error while compiling statement: FAILED: SemanticException [Error 300
34]: Concatenate/Merge can only be performed on managed tables (state=42000,
code=30034)
```

Action Required

Remove the CONCATENATE operation on external tables.

Distribution Affected

CDH5, CDH6

INSERT OVERWRITE

INSERT OVERWRITE from a source with UNION ALL is not allowed for full ACID tables.

Hive 1 and 2

For managed tables, INSERT OVERWRITE from a source with UNION ALL is allowed.

Hive 3

INSERT OVERWRITE from a source with UNION ALL on full CRUD ACID tables is not allowed.

For example, test_groupby_3 is a managed table and insert overwrite with union all will throw an error.

```
insert overwrite table test_groupby_3
. . . . . . . . . . . . . . . . . . . . . . .> select col1,count from test
_groupby
. . . . . . . . . . . . . . . . . . . . . . .> union all
. . . . . . . . . . . . . . . . . . . . . . .> select col1,count from test
_groupby_1;

ERROR : FAILED: Error in acquiring locks: QueryId=hive_20220321114459_851b
1546-ca9f-4643-b888-7df23adaec66 is not supported due to OVERWRITE and UNION
 ALL.  Please use truncate + insert
```

Action Required

Instead of overwrite, use truncate and insert.

Distribution Affected

CDH5, CDH6, HDP2

Managed to external table

In Hive1/2, a managed table can be converted to an external table by setting the external property of the table as true. In Hive3 this is restricted.

Hive 1 and 2

ALTER TABLE table_name SET TBLPROPERTIES('EXTERNAL'='true') can be performed on managed tables to convert external tables.

For example:

```
ALTER TABLE test_groupby_3 SET TBLPROPERTIES('EXTERNAL'='true')
ERROR : Failed
org.apache.hadoop.hive.ql.metadata.HiveException: Unable to alter table.
default.test_groupby_3 cannot be converted to external table as it is transa
ctional table.
```

Hive 3

ALTER TABLE table_name SET TBLPROPERTIES('EXTERNAL'='true') cannot be performed on managed tables to convert external tables.

Action Required

Remove a table property that sets EXTERNAL=true for a managed table.

Distribution Affected

CDH5, CDH6, HDP2

Property changes affecting ordered or sorted subqueries and views

Order by/sort by without limit in subqueries is not supported in Hive 3.

Hive 1 and 2

You can use order by/sort by functionality in subqueries.

For example, the optimizer orders a subquery of the following table:

```
+--------------------+---------------------+
```

```
| test_groupby.col1  | test_groupby.col2   |
+--------------------+---------------------+
| test               | 2                   |
| test               | 3                   |
| test1              | 5                   |
+--------------------+---------------------+

select * from (select * from test_groupby order by col2 desc) t2;

+----------+-----------+
| t2.col1  | t2.col1   |
+----------+-----------+
| test1    | 5         |
| test     | 3         |
| test     | 2         |
+----------+-----------+
```

Hive 3

The optimizer removes order by/sort by without limit in subqueries and views. For example:

```
select * from (select * from test_groupby order by col2 desc) t2;

+----------+-----------+
| t2.col1  | t2.col1   |
+----------+-----------+
| test     | 2         |
| test     | 3         |
| test1    | 5         |
+----------+-----------+
```

Action Required

If the outer query has to perform some functional logic based on order of a subquery, the following query in Hive 2 returns a different result from Hive 3:

```
select col1 from (select * from test_groupby order by col2t desc ) t2 limit
1;
```

You must rewrite the query or set the following property to restore previous behavior:

```
set hive.remove.orderby.in.subquery=False
```

Distribution Affected

CDH5, CDH6

Runtime configuration changes

There are a number of runtime configurations that Hive 3 does not support.

Remove unsupported configurations if explicitly set in scripts.

The following configurations supported in Hive 1 or 2 have been removed from Hive 3 and are not supported:

- hive.limit.query.max.table.partition
- hive.warehouse.subdir.inherit.perms
- hive.stats.fetch.partition.stats

Development of an HBase metastore for Hive started in release 2.0.0, but the work has stopped and the code was removed from Hive 3.0.0. The following Hive Metastore HBase configurations have been removed and are not supported:

- hive.metastore.hbase.cache.ttl
- hive.metastore.hbase.catalog.cache.size
- hive.metastore.hbase.aggregate.stats.cache.size
- hive.metastore.hbase.aggregate.stats.max.partitions
- hive.metastore.hbase.aggregate.stats.false.positive.probability
- hive.metastore.hbase.aggregate.stats.max.variance
- hive.metastore.hbase.cache.ttl
- hive.metastore.hbase.cache.max.writer.wait
- hive.metastore.hbase.cache.max.reader.wait
- hive.metastore.hbase.cache.max.full
- hive.metastore.hbase.catalog.cache.size
- hive.metastore.hbase.aggregate.stats.cache.size
- hive.metastore.hbase.aggregate.stats.max.partitions
- hive.metastore.hbase.aggregate.stats.false.positive.probability
- hive.metastore.hbase.aggregate.stats.max.variance
- hive.metastore.hbase.cache.ttl
- hive.metastore.hbase.cache.max.full
- hive.metastore.hbase.cache.clean.until
- hive.metastore.hbase.connection.class
- hive.metastore.hbase.aggr.stats.cache.entries
- hive.metastore.hbase.aggr.stats.memory.ttl
- hive.metastore.hbase.aggr.stats.invalidator.frequency
- hive.metastore.hbase.aggr.stats.hbase.ttl

Prepare Hive tables for migration

To prepare the tables for migration, use Hive SRE Tool which is a Cloudera Lab tool that scans your Hive Metastore and HDFS to identify common upgrade problems that can cause the Cloudera upgrade to fail. The tool provides guidance for fixing those problems before migrating the tables to Cloudera. This guidance is provided through reports that the administrator must take action on. The tool does not take corrective action itself.

Cloudera strongly recommends running the Hive SRE Tool to aid in this pre-upgrade HMS healthcheck. If you do not run it, you must manually investigate your HMS for the following types of problems.

Please refer to Hive SRE Tooling for tool setup, tool execution & interpretation of output using Hive SRE Tool on Hive Metadata.

The general transition from Hive 1 and 2 to Hive 3 includes the following types of HMS operations. The Hive SRE Tool performs the equivalent types of checks in an automated fashion. Please review these from the Cloudera documentation site (CDH 5, CDH 6, HDP 2, HDP 3).

Note for section below:XXX stands for some value/count, this is a value of the number of affected paths we will get from the Hive-SRE tool output reports. For each type of condition listed in the table , there will be some number of paths/tables affected.

Condition	Impact	Distribution Affected
Bad Filenames	Tables that would be converted from a Managed Non-Acid table to an ACID transactional table require the files to match a certain pattern. This process will scan the potential directories of these tables for bad filename patterns. When located, it will indicate which tables/partitions have file naming conventions that would prevent a successful conversion to ACID. The best and easiest way to correct these file names is to use HiveSQL to rewrite the contents of the table/partition with a simple 'INSERT OVERWRITE TABLE xxx SELECT * FROM xxx'. This type of statement will replace the current bad filenames with valid file names by rewriting the contents in HiveSQL. There are approximately XXX paths that may need remediation. The list of paths can be found in the output of the Hive assessment report in the u3/bad_filenames_for_orc_conversion.md.	
Missing Directories	Missing Directories cause the upgrade conversion process to fail. This inconsistency is an indicator that data was removed from the file system, but the Hive MetaStore was not updated to reflect that operation. An example of this is deleting a partition in HDFS, without dropping the partition in Hive. There are XXX affected paths that need remediation. The list of directories can be found in the output of the Hive assessment report, in the file u3/loc_scan_missing_dirs.md.	
Managed Shadow Tables	In Hive 3, Managed tables are 'ACID' tables. Sharing a location between two 'ACID' tables will cause compaction issues and data issues. These need to be resolved before the upgrade. There are XXX affected tables that need remediation. The list of paths can be found in the output of the Hive assessment report, in the file u3/hms_checks.md	
Managed Table Migrations	This process will list tables that will and 'could' be migrated to "Managed ACID" tables during the upgrade process. Tables used directly by Spark or if data is managed by a separate process that interacts with the FileSystem, you may experience issues post-upgrade. Recommended: Consider converting these tables to external tables. There are XXX affected tables that may need remediation. The list of tables can be found in the output of the Hive assessment report in the file managed_upgrade_2_acid.sql.	

Condition	Impact	Distribution Affected
Compaction Check	In the upgrade to Hive 3, ACID tables must be compacted prior to initiating the upgrade. XXX tables were noted as requiring compaction. Because CDH does not support Hive ACID tables, this may be a leftover from the previous HDP to CDH migration that the customer implemented. This should be investigated further. The affected table is ers_stage_tls.test_delete	
Unknown SerDe Jars	Will list tables using SerDe's that are not standard to the platform. Review list of SerDes and verify they are still necessary and available for Cloudera. There are approximately XXX tables configured with 3rd party SerDes.	
Remove transactional=false from Table Properties	In CDH 5.x it is possible to create tables with the property transactional=false set. While this is a no-op setting, if any of your Hive tables explicitly set this, the upgrade process fails. You must remove 'transactional'='false' from any tables you want to upgrade from CDH 5.x to Cloudera. Alter the table as follows: ALTER TABLE my_table UNSET TBLPR OPERTIES ('transactional');	
Make Tables SparkSQL Compatible	Non-Acid, managed tables in ORC or in a Hive Native (but non-ORC) format that are owned by the POSIX user hive will not be SparkSQL compatible after the upgrade unless you perform manual conversions. If your table is a managed, non-ACID table, you can convert it to an external table using this procedure (recommended). After the upgrade, you can easily convert the external table to an ACID table, and then use the Hive Warehouse Connector to access the ACID table from Spark. Take one of the following actions. • Convert the tables to external Hive tables before the upgrade. `ALTER TABLE ... SET TBLPROPERTIES('EXTERNAL'='TRUE','external.table.purge'='true')` • Change the POSIX ownership to an owner other than hive. You will need to convert managed, ACID v1 tables to external tables after the upgrade.	

Condition	Impact	Distribution Affected
Legacy Kudu Serde Report	Early versions of Hive/Impala tables using Kudu were built before Kudu became an Apache Project. Once it became an Apache Project, the base Kudu Storage Handler classname changed. This report locates and reports on tables using the legacy storage handler class.	
Legacy Decimal Scale and Precision Check	When the DECIMAL data type was first introduced in Hive 1, it did NOT include a Scale or Precision element. This causes issues in later integration with Hive and Spark. We'll identify and suggest corrective action for tables where this condition exists.	
Database / Table and Partition Counts	Use this to understand the scope of what is in the metastore.	
Small Files, Table Volumes, Empty Datasets	Identify and fix these details to clean up unwanted datasets in the cluster which would speed up the Hive upgrade process.	
Merge Independent Hive and Spark Catalogs	In HDP 3.0 - 3.1.4, Spark and Hive use independent catalogs for accessing tables created using SparkSQL or Hive tables. A table created from Spark resides in the Spark catalog. A table created from Hive resides in the Hive catalog. Databases fall under the catalog namespace, similar to how tables belong to a database namespace. In HPD 3.1.5, Spark and Hive share a catalog in Hive metastore (HMS) instead of using separate catalogs. The Apache Hive schematool in HDP 3.1.5 and Cloudera releases supports the mergeCatalog task.	

Impala changes from CDH to Cloudera

Cloudera documentation provides details about changes in Impala when migrating from CDH 5.13-5.16 or CDH 6.1 or later to Cloudera.

- Impala changes in Cloudera:This document lists down the syntax, service, property and configuration changes that affect Impala after upgrade from CDH 5.13-5.16 or CDH 6.1 or later to Cloudera.
- Change location of Datafiles: This document explains the impact on Hive warehouse directory location for Impala managed and external tables.
- Set Storage Engine ACLs : This document describes the steps to set ACLs for Impala to allow Impala to write to the Hive Warehouse Directory.
- Automatic Invalidation/Refresh of Metadata: In Cloudera, HMS and file system metadata is automatically refreshed. This document provides the details of change in behavior for invalidation and refresh of metadata between CDH and Cloudera.
- Metadata Improvements: In Cloudera, all catalog metadata improvements are enabled by default. This document lists down the metadata properties in Cloudera that can be customized for better performance and scalability.
- Default Managed Tables: In Cloudera, managed tables are transactional tables with the insert_only property by default. This document describes modifying file systems on a managed table in Cloudera and the methods to switch to the old CDH behavior.
- Automatic Refresh of Tables on Impala Clusters: In Cloudera tables or partitions are refreshed automatically on other impala clusters, this document describes the property that enables this behavior.
- Interoperability between Hive and Impala:This document describes the changes made in Cloudera for the optimal interoperability between Hive and Impala.

- ORC Support Disabled for Full-Transactional Tables: In Cloudera 7.1.0 and earlier versions, ORC table support is disabled for Impala queries. However, you have an option to switch to the CDH behavior by using the command line argument ENABLE_ORC_SCANNER.
- Metadata Improvements: In Cloudera, all catalog metadata improvements are enabled by default. This document lists down the metadata properties in Cloudera that can be customized for better performance and scalability.
- Default Managed Tables: In Cloudera, managed tables are transactional tables with the insert_only property by default. This document describes modifying file systems on a managed table in Cloudera and the methods to switch to the old CDH behavior.
- Automatic Refresh of Tables on Impala Clusters: In Cloudera tables or partitions are refreshed automatically on other impala clusters, this document describes the property that enables this behavior.
- Interoperability between Hive and Impala:This document describes the changes made in Cloudera for the optimal interoperability between Hive and Impala.
- ORC Support Disabled for Full-Transactional Tables: In Cloudera 7.1.0 and earlier versions, ORC table support is disabled for Impala queries. However, you have an option to switch to the CDH behavior by using the command line argument ENABLE_ORC_SCANNER.
- Authorization Provider for Impala:In Cloudera, Ranger is the authorization provider instead of Sentry. This document describes changes about Ranger enforcing a policy which may be different from using Sentry.
- Data Governance Support by Atlas:As part of upgrading a CDH cluster to Cloudera, Atlas replaces Cloudera Navigator Data Management for your cluster. This document describes the difference between two environments.

Impala configuration differences in CDH and Cloudera

There are some configuration differences related to Impala in CDH and Cloudera. These differences are due to the changes made in Cloudera for the optimal interoperability between Hive and Impala and an improved user experience.

Review the changes before you migrate your Impala workload from CDH to CDP.

- Config changes:This document describes the default value changes and new configurations between CDH and Cloudera.
- Default File Formats: This document describes the change in default file format of the table in Cloudera and the steps to switch back to CDH behavior.
- Reconnect to HS2 Session: This document describes the behavior change in Cloudera for client connection to Impala and HS2.
- Automatic Row Count Estimation: This document describes the new default behavior in Cloudera for calculating statistics of the table and the property to switch back to CDH behavior.
- Using Reserved Words in SQL Queries:Cloudera follows ANSI SQL compliance, in which reserved words in SQL queries are rejected. This document provides details about reserved keywords and the method to use them.
- Other Miscellaneous Changes in Impala:This document describes other miscellaneous changes related to Impala Syntax and services that affect Impala during migration.

Additional documentation

Cloudera documentation can help you migrate your Hive workloads.

For more information about migrating Hive workload, see the following documentation:

- Migrating tables to Cloudera
- Migrating Hive workloads from CDH
- Migrating Hive Workloads from HDP 2.6.5 after an in-place upgrade
- Replicating Hive data from HDP 3 to Cloudera
- Migrating Hive workloads to ACID

- Apache Hive Changes in Cloudera capture additional changes that need to be considered while upgrading from Hive 1/2 to Hive 3. The following documents summarize these changes:
  - Hive Configuration Property Changes
  - LOCATION and MANAGEDLOCATION clauses
  - Handling table reference syntax
  - Identifying semantic changes and workarounds
  - Unsupported Interfaces and Features
  - Changes to CDH Hive Tables
  - Changes to HDP Hive tables
- Cloudera Upgrade and Migrations Paths
- Migrating Hive 1-2 to Hive 3
- Migrating Hive Workloads from CDH
- SRE Tool
- Apache Hive Language Manual
- Release Notes

..

Migrating Spark to CDP Private Cloud

Date published: 2022-07-29
Date modified: 2022-07-29

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Migrating Spark workloads to Cloudera.....4
Spark 1.6 to Spark 2.4 Refactoring.....4
Handling prerequisites.....4
Spark 1.6 to Spark 2.4 changes.....5
Configuring storage locations.....11
Querying Hive managed tables from Spark.....12
Compiling and running Spark workloads.....12
Post-migration tasks.....17
Spark 2.3 to Spark 2.4 Refactoring.....17
Handling prerequisites.....18
Spark 2.3 to Spark 2.4 changes.....18
Configuring storage locations.....22
Querying Hive managed tables from Spark.....23
Compiling and running Spark workloads.....23
Post-migration tasks.....23

Migrating Spark workloads to Cloudera

Migrating Spark workloads from CDH or HDP to Cloudera involves learning the Spark semantic changes in your source cluster and the Cloudera target cluster. You get details about how to handle these changes.

Spark 1.6 to Spark 2.4 Refactoring

Because Spark 1.6 is not supported on Cloudera, you need to refactor Spark workloads from Spark 1.6 on CDH or HDP to Spark 2.4 on Cloudera.

This document helps in accelerating the migration process, provides guidance to refactor Spark workloads and lists migration. Use this document when the platform is migrated from CDH or HDP to Cloudera.

Handling prerequisites

You must perform a number of tasks before refactoring workloads.

About this task

Assuming all workloads are in working condition, you perform this task to meet refactoring prerequistes.

Procedure

1. Identify all the workloads in the cluster (CDH/HDP) which are running on Spark 1.6 - 2.3.
2. Classify the workloads.

   Classification of workloads will help in clean-up of the unwanted workloads, plan resources and efforts for workload migration and post upgrade testing.

   Example workload classifications:

   - Spark Core (scala)
   - Java-based Spark jobs
   - SQL, Datasets, and DataFrame
   - Structured Streaming
   - MLlib (Cloudera AI)
   - PySpark (Python on Spark)
   - Batch Jobs
   - Scheduled Jobs
   - Ad-Hoc Jobs
   - Critical/Priority Jobs
   - Huge data Processing Jobs
   - Time taking jobs
   - Resource Consuming Jobs etc.
   - Failed Jobs

Identify configuration changes

3. Check the current Spark jobs configuration.

   - Spark 1.6 - 2.3 workload configurations which have dependencies on job properties like scheduler, old python packages, classpath jars and might not be compatible post migration.
   - In Cloudera, Capacity Scheduler is the default and recommended scheduler. Follow Fair Scheduler to Capacity Scheduler transition guide to have all the required queues configured in the Cloudera cluster post upgrade. If any configuration changes are required, modify the code as per the new capacity scheduler configurations.
   - For workload configurations, see the Spark History server UI http://spark_history_server:18088/history/<application_number>/environment/.

4. Identify and capture workloads having data storage locations (local and HDFS) to refactor the workloads post migration.
5. Refer to unsupported Apache Spark features, and plan refactoring accordingly.

Spark 1.6 to Spark 2.4 changes

A description of the change, the type of change, and the required refactoring provide the information you need for migrating from Spark 1.6 to Spark 2.4.

New Spark entry point SparkSession

There is a new Spark API entry point: SparkSession.

Type of change

Syntactic/Spark core

Spark 1.6

Hive Context and SQLContext, such as import SparkContext, HiveContext are supported.

Spark 2.4

SparkSession is now the entry point.

Action Required

Replace the old SQLContext and HiveContext with SparkSession. For example:

```
import org.apache.spark.sql.SparkSession
    val spark = SparkSession
    .builder()
    .appName("Spark SQL basic example")
    .config("spark.some.config.option", "some-value")
    .getOrCreate()
```

.

Dataframe API registerTempTable deprecated

The Dataframe API registerTempTable has been deprecated in Spark 2.4.

Type of change:

Syntactic/Spark core change

Spark 1.6

registerTempTable is used to create a temporary table on a Spark dataframe. For example, df.registerTempTable('tmpTable").

Spark 2.4

registerTempTable is deprecated.

Action Required

Replace registerTempTable using createOrReplaceTempView. df.createOrReplaceTempView('tmpTable').

union replaces unionAll

The dataset and DataFrame API unionAll has been deprecated and replaced by union.

Type of change: Syntactic/Spark core change

Spark 1.6

unionAll is supported.

Spark 2.4

unionAll is deprecated and replaced by union.

Action Required

Replace unionAll with union. For example:val df3 = df.unionAll(df2) with val df3 = df.union(df2)

Empty schema not supported

Writing a dataframe with an empty or nested empty schema using any file format, such as parquet, orc, json, text, or csv is not allowed.

Type of change: Syntactic/Spark core

Spark 1.6 - 2.3

Writing a dataframe with an empty or nested empty schema using any file format is allowed and will not throw an exception.

Spark 2.4

An exception is thrown when you attempt to write dataframes with empty schema. For example, if there are statements such as df.write.format("parquet").mode("overwrite").save(somePath), the following error occurs: org.apache.spark.sql.AnalysisException: Parquet data source does not support null data type.

Action Required

Make sure that DataFrame is not empty. Check whether DataFrame is empty or not as follows:

```
if (!df.isEmpty) df.write.format("parquet").mode("overwrite").save("somePath
")
```

Referencing a corrupt JSON/CSV record

In Spark 2.4, queries from raw JSON/CSV files are disallowed when the referenced columns only include the internal corrupt record column.

Type of change: Syntactic/Spark core

Spark 1.6

A query can reference a _corrupt_record column in raw JSON/CSV files.

Spark 2.4

An exception is thrown if the query is referencing _corrupt_record column in these files. For example, the following query is not allowed: spark.read.schema(schema).json(file).filter($"_corrupt_record".isNotNull).count()

Action Required

Cache or save the parsed results, and then resend the query.

```
val df = spark.read.schema(schema).json(file).cache()
    df.filter($"_corrupt_record".isNotNull).count()
```

Dataset and DataFrame API explode deprecated

Dataset and DataFrame API explode has been deprecated.

Type of change: Syntactic/Spark SQL change

Spark 1.6

Dataset and DataFrame API explode are supported.

Spark 2.4

Dataset and DataFrame API explode have been deprecated. If explode is used, for example dataframe.explode(), the following warning is thrown:

```
warning: method explode in class Dataset is deprecated: use flatMap() or sel
ect() with functions.explode() instead
```

Action Required

Use functions.explode() or flatMap (import org.apache.spark.sql.functions.explode).

CSV header and schema match

Column names of csv headers must match the schema.

Type of change: Configuration/Spark core changes

Spark 1.6 - 2.3

Column names of headers in CSV files are not checked against the against the schema of CSV data.

Spark 2.4

If columns in the CSV header and the schema have different ordering, the following exception is thrown:java.lang.Il legalArgumentException: CSV file header does not contain the expected fields.

Action Required

Make the schema and header order match or set enforceSchema to false to prevent getting an exception. For example, read a file or directory of files in CSV format into Spark DataFrame as follows: df3 = spark.read.option("delimiter", ";").option("header", True).option("enforeSchema", False).csv(path)

The default "header" option is true and enforceSchema is False.

If enforceSchema is set to true, the specified or inferred schema will be forcibly applied to datasource files, and headers in CSV files are ignored. If enforceSchema is set to false, the schema is validated against all headers in CSV files when the header option is set to true. Field names in the schema and column names in CSV headers are checked by their positions taking into account spark.sql.caseSensitive. Although the default value is true,you should disable the enforceSchema option to prevent incorrect results.

Table properties support

Table properties are taken into consideration while creating the table.

Type of change: Configuration/Spark Core Changes

Spark 1.6 - 2.3

Parquet and ORC Hive tables are converted to Parquet or ORC by default, but table properties are ignored. For example, the compression table property is ignored:

```
CREATE TABLE t(id int) STORED AS PARQUET TBLPROPERTIES (parquet.compression
'NONE')
```

This command generates Snappy Parquet files.

Spark 2.4

Table properties are supported. For example, if no compression is required, set the TBLPROPERTIES as follows: (parquet.compression 'NONE').

This command generates uncompressed Parquet files.

Action Required

Check and set the desired TBLPROPERTIES.

CREATE OR REPLACE VIEW and ALTER VIEW not supported

ALTER VIEW and CREATE OR REPLACE VIEW AS commands are no longer supported.

Type of change: Configuration/Spark Core Changes

Spark 1.6

You can create views as follows:

```
CREATE OR REPLACE [ [ GLOBAL ] TEMPORARY ] VIEW [ IF NOT EXISTS ] view_name
  [ column_list ]
  [ COMMENT view_comment ]
  [ properties ]
  AS query

  ALTER VIEW view_name
  { rename |
  set_properties |
  unset_properties |
  alter_body }
```

Spark 2.4

ALTER VIEW and CREATE OR REPLACE commands above are not supported.

Action Required

Recreate views using the following syntax:

```
CREATE [ [ GLOBAL ] TEMPORARY ] VIEW [ IF NOT EXISTS ] view_name
    [ column_list ]
    [ COMMENT view_comment ]
    [ properties ]
    AS query
```

Managed table location

Creating a managed table with nonempty location is not allowed.

Type of change: Property/Spark core changes

Spark 1.6 - 2.3

You can create a managed table having a nonempty location.

Spark 2.4

Creating a managed table with nonempty location is not allowed. In Spark 2.4, an error occurs when there is a write operation, such as df.write.mode(SaveMode.Overwrite).saveAsTable("testdb.testtable"). The error side-effects are the cluster is terminated while the write is in progress, a temporary network issue occurs, or the job is interrupted.

Action Required

Set spark.sql.legacy.allowCreatingManagedTableUsingNonemptyLocation to true at runtime as follows:

```
spark.conf.set("spark.sql.legacy.allowCreatingManagedTableUsingNonemptyLocat
ion","true")
```

Write to Hive bucketed tables

Type of change: Property/Spark SQL changes

Spark 1.6

By default, you can write to Hive bucketed tables.

Spark 2.4

By default, you cannot write to Hive bucketed tables.

For example, the following code snippet writes the data into a bucketed Hive table:

```
newPartitionsDF.write.mode(SaveMode.Append).format("hive").insertInto(hive_t
est_db.test_bucketing)
```

The code above will throw the following error:

```
org.apache.spark.sql.AnalysisException: Output Hive table `hive_test_db`.`te
st_bucketing` is bucketed but Spark currently does NOT populate bucketed out
put which is compatible with Hive.
```

Action Required

To write to a Hive bucketed table, you must use hive.enforce.bucketing=false and hive.enforce.sorting=false to forego bucketing guarantees.

Rounding in arithmetic operations

Arithmetic operations between decimals return a rounded value, instead of NULL, if an exact representation is not possible.

Type of change: Property/Spark SQL changes

Spark 1.6

Arithmetic operations between decimals return a NULL value if an exact representation is not possible.

Spark 2.4

The following changes have been made:

- Updated rules determine the result precision and scale according to the SQL ANSI 2011.
- Rounding of the results occur when the result cannot be exactly represented with the specified precision and scale instead of returning NULL.
- A new config spark.sql.decimalOperations.allowPrecisionLoss which default to true (the new behavior) to allow users to switch back to the old behavior. For example, if your code includes import statements that resemble those below, plus arithmetic operations, such as multiplication and addition, operations are performed using dataframes.

```
from pyspark.sql.types import DecimalType
from decimal import Decimal
```

Action Required

If precision and scale are important, and your code can accept a NULL value (if exact representation is not possible due to overflow), then set the following property to false. spark.sql.decimalOperations.allowPrecisionLoss = false

Precedence of set operations

Set operations are executed by priority instead having equal precedence.

Type of change: Property/Spark SQL changes

Spark 1.6 - 2.3

If the order is not specified by parentheses, equal precedence is given to all set operations.

Spark 2.4

If the order is not specified by parentheses, set operations are performed from left to right with the exception that all INTERSECT operations are performed before any UNION, EXCEPT or MINUS operations.

For example, if your code includes set operations, such as INTERSECT , UNION, EXCEPT or MINUS, consider refactoring.

Action Required

Change the logic according to following rule:

If the order of set operations is not specified by parentheses, set operations are performed from left to right with the exception that all INTERSECT operations are performed before any UNION, EXCEPT or MINUS operations.

If you want the previous behavior of equal precedence then, set spark.sql.legacy.setopsPrecedence.enabled=true.

HAVING without GROUP BY

HAVING without GROUP BY is treated as a global aggregate.

Type of change: Property/Spark SQL changes

Spark 1.6 - 2,3

HAVING without GROUP BY is treated as WHERE. For example, SELECT 1 FROM range(10) HAVING true is executed as SELECT 1 FROM range(10) WHERE true, and and returns 10 rows.

Spark 2.4

HAVING without GROUP BY is treated as a global aggregate. For example, SELECT 1 FROM range(10) HAVING true returns one row, instead of 10, as in the previous version.

Action Required

Check the logic where having and group by is used. To restore previous behavior, set spark.sql.legacy.parser.havingWithoutGroupByAsWhere=true.

CSV bad record handling

How Spark treats malformations in CSV files has changed.

Type of change: Property/Spark SQL changes

Spark 1.6 - 2.3

CSV rows are considered malformed if at least one column value in the row is malformed. The CSV parser drops malformed rows in the DROPMALFORMED mode or outputs an error in the FAILFAST mode.

Spark 2.4

A CSV row is considered malformed only when it contains malformed column values requested from CSV datasource, other values are ignored.

Action Required

To restore the Spark 1.6 behavior, set spark.sql.csv.parser.columnPruning.enabled to false.

Spark 1.4 - 2.3 CSV example

A Spark 1.4 - 2.3 CSV example illustrates the CSV-handling change in Spark 2.6.

In the following CSV file, the first two records describe the file. These records are not considered during processing and need to be removed from the file. The actual data to be considered for processing has three columns (jersey, name, position).

```
These are extra line1
These are extra line2
10,Messi,CF
7,Ronaldo,LW
9,Benzema,CF
```

The following schema definition for the DataFrame reader uses the option DROPMALFORMED. You see only the required data; all the description and error records are removed.

```
schema=Structtype([Structfield(“jersy”,StringType()),Structfield(“name”,Stri
ngType()),Structfi
eld(“position”,StringType())])
df1=spark.read\
.option(“mode”,”DROPMALFORMED”)\
.option(“delimiter”,”,”)\
```

```
.schema(schema)\
.csv("inputfile")
df1.select("*").show()
```

Output is:

jersy	name	position
10	Messi	CF
7	Ronaldo	LW
9	Benzema	CF

Select two columns from the dataframe and invoke show():

```
df1.select("jersy","name").show(truncate=False)
```

jersy	name
These are extra line1	null
These are extra line2	null
10	Messi
7	Ronaldo
9	Benzema

Malformed records are not dropped and pushed to the first column and the remaining columns will be replaced with null.This is due to the CSV parser column pruning which is set to true by default in Spark 2.4.

Set the following conf, and run the same code, selecting two fields.

```
spark.conf.set("spark.sql.csv.parser.columnPruning.enabled",False)
```

```
df2=spark.read\
   .option("mode","DROPMALFORMED")\
   .option("delimiter",",")\
   .schema(schema)\
   .csv("inputfile")
   df2.select("jersy","name").show(truncate=False)
```

jersy	name
10	Messi
7	Ronaldo
9	Benzema

Conclusion: If working on selective columns, to handle bad records in CSV files, set spark.sql.csv.parser.columnP runing.enabled to false; otherwise, the error record is pushed to the first column, and all the remaining columns are treated as nulls.

Configuring storage locations

To execute the workloads in Cloudera, you must modify the references to storage locations. In Cloudera, references must be changed from HDFS to a cloud object store such as S3.

About this task

The following sample query shows a Spark 2.4 HDFS data location.

```
scala> spark.sql("CREATE TABLE IF NOT EXISTS default.sales_spark_2(Region st
ring, Country string,Item_Type string,Sales_Channel string,Order_Priority st
ring,Order_Date date,Order_ID int,Ship_Date date,Units_sold string,Unit_Pric
e string,Unit_cost string,Total_revenue string,Total_Cost string,Total_Profi
t string) row format delimited fields terminated by ','")
scala> spark.sql("load data local inpath '/tmp/sales.csv' into table defau
lt.sales_spark_2")
scala> spark.sql("select count(*) from default.sales_spark_2").show()
```

The following sample query shows a Spark 2.4 S3 data location.

```
scala> spark.sql("CREATE TABLE IF NOT EXISTS default.sales_spark_2(Region st
ring, Country string,Item_Type string,Sales_Channel string,Order_Priority st
ring,Order_Date date,Order_ID int,Ship_Date date,Units_sold string,Unit_Pric
e string,Unit_cost string,Total_revenue string,Total_Cost string,Total_Profi
t string) row format delimited fields terminated by ','")
scala> spark.sql("load data inpath 's3://<bucket>/sales.csv' into table de
fault.sales_spark_2")
scala> spark.sql("select count(*) from default.sales_spark_2").show()
```

Querying Hive managed tables from Spark

Hive-on-Spark is not supported on Cloudera. You need to use the Hive Warehouse Connector (HWC) to query Apache Hive managed tables from Apache Spark.

To read Hive external tables from Spark, you do not need HWC. Spark uses native Spark to read external tables. For more information, see the Hive Warehouse Connector documentation.

The following example shows how to query a Hive table from Spark using HWC:

```
spark-shell --jars /opt/cloudera/parcels/CDH/jars/hive-warehouse-connector-a
ssembly-1.0.0.7.1.4.0-203.jar --conf spark.sql.hive.hiveserver2.jdbc.url=jdb
c:hive2://cdhhdp02.uddeepta-bandyopadhyay-s-account.cloud:10000/default --co
nf spark.sql.hive.hiveserver2.jdbc.url.principal=hive/cdhhdp02.uddeepta-band
yopadhyay-s-account.cloud@Uddeepta-bandyopadhyay-s-Account.CLOUD
scala> val hive = com.hortonworks.hwc.HiveWarehouseSession.session(spark).b
uild()
scala> hive.executeUpdate("UPDATE hive_acid_demo set value=25 where key=4")
scala> val result=hive.execute("select * from default.hive_acid_demo")
scala> result.show()
```

Compiling and running Spark workloads

After modifying the workloads, compile and run (or dry run) the refactored workloads on Spark 2.4.

You can write Spark applications using Java, Scala, Python, SparkR, and others. You build jars from these scripts using one of the following compilers.

- Java (with Maven/Java IDE),
- Scala (with sbt),
- Python (pip).
- SparkR (RStudio)

Compiling and running a Java-based job

You see by example how to compile a Java-based Spark job using Maven.

About this task

In this task, you see how to compile the following example Spark program written in Java:

```
/* SimpleApp.java */
import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.Dataset;

public class SimpleApp {
  public static void main(String[] args) {
    String logFile = "YOUR_SPARK_HOME/README.md"; // Should be some file on
 your system
    SparkSession spark = SparkSession.builder().appName("Simple Applicatio
n").getOrCreate();
    Dataset<String> logData = spark.read().textFile(logFile).cache();

    long numAs = logData.filter(s -> s.contains("a")).count();
    long numBs = logData.filter(s -> s.contains("b")).count();

   System.out.println("Lines with a: " + numAs + ", lines with b: " + num
Bs);

    spark.stop();
  }
}
```

You also need to create a Maven Project Object Model (POM) file, as shown in the following example:

```
<project>
  <groupId>edu.berkeley</groupId>
  <artifactId>simple-project</artifactId>
  <modelVersion>4.0.0</modelVersion>
  <name>Simple Project</name>
  <packaging>jar</packaging>
  <version>1.0</version>
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
<dependencies>
    <dependency> <!-- Spark dependency -->
   <groupId>org.apache.spark</groupId>
   <artifactId>spark-sql_2.12</artifactId>
   <version>2.4.0</version>
   <scope>provided</scope>
    </dependency>
</dependencies>
</project>
```

Before you begin

- Install Apache Spark 2.4.x, JDK 8.x, and maven
- Write a Java Spark program .java file.
- Write a pom.xml file. This is where your Scala code resides.
- If the cluster is Kerberized, ensure the required security token is authorized to compile and execute the workload.

Procedure

1. Lay out these files according to the canonical Maven directory structure.
   For example:

```
$ find .
./pom.xml
./src
./src/main
./src/main/java
./src/main/java/SimpleApp.java
```

2. Package the application using maven package command.
   For example:

```
# Package a JAR containing your application
$ mvn package
...
[INFO] Building jar: {..}/{..}/target/simple-project-1.0.jar
```

   After compilation, several new files are created under new directories named project and target. Among these new files, is the jar file under the target directory to run the code. For example, the file is named simple-project-1.0.jar.

3. Execute and test the workload jar using the spark submit command.
   For example:

```
# Use spark-submit to run your application
spark-submit \
--class "SimpleApp" \
--master yarn \
target/simple-project-1.0.jar
```

Compiling and running a Scala-based job

You see by example how to use sbt software to compile a Scala-based Spark job.

About this task

In this task, you see how to use the following .sbt file that specifies the build configuration:

```
cat build.sbt
name := "Simple Project"
version := "1.0"
scalaVersion := "2.12.15"
libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.4.0"
```

You also need to create a compile the following example Spark program written in Scala:

```
/* SimpleApp.scala */
import org.apache.spark.sql.SparkSession

object SimpleApp {
  def main(args: Array[String]) {
 val logFile = "YOUR_SPARK_HOME/README.md" // Should be some file on your
system
 val spark = SparkSession.builder.appName("Simple Application").getOrCrea
te()
 val logData = spark.read.textFile(logFile).cache()
 val numAs = logData.filter(line => line.contains("a")).count()
 val numBs = logData.filter(line => line.contains("b")).count()
    println(s"Lines with a: $numAs, Lines with b: $numBs")
 spark.stop()
```

```
  }
```

Before you begin

- Install Apache Spark 2.4.x.
- Install JDK 8.x.
- Install Scala 2.12.
- Install Sbt 0.13.17.
- Wrtie an .sbt file for configuration specifications, similar to a C include file.
- Write a Scala-based Spark program (a .scala file).
- If the cluster is Kerberized, ensure the required security token is authorized to compile and execute the workload.

Procedure

1. Compile the code using sbt package command from the directory where the build.sbt file exists.
   For example:

   ```
   # Your directory layout should look like this
   $ find .
   .
   ./build.sbt
   ./src
   ./src/main
   ./src/main/scala
   ./src/main/scala/SimpleApp.scala

   # Package a jar containing your application
   $ sbt package
   ...
   [info] Packaging {..}/{..}/target/scala-2.12/simple-project_2.12-1.0.jar
   ```

   Several new files are created under new directories named project and target, including the jar file named simple-project_2.12-1.0.jar after the project name, Scala version, and code version.
2. Execute and test the workload jar using spark submit.
   For example:

   ```
   # Use spark-submit to run your application
   spark-submit \
     --class "SimpleApp" \
     --master yarn \
     target/scala-2.12/simple-project_2.12-1.0.jar
   ```

Running a Python-based job

You can run a Python script to execute a spark-submit or pyspark command.

About this task

In this task, you execute the following Python script that creates a table and runs a few queries:

```
/* spark-demo.py */
from pyspark import SparkContext
sc = SparkContext("local", "first app")
from pyspark.sql import HiveContext
hive_context = HiveContext(sc)
hive_context.sql("drop table default.sales_spark_2_copy")
hive_context.sql("CREATE TABLE IF NOT EXISTS default.sales_spark_2_copy as
 select * from default.sales_spark_2")
hive_context.sql("show tables").show()
hive_context.sql("select * from default.sales_spark_2_copy limit 10").show()
```

```
hive_context.sql("select count(*) from default.sales_spark_2_copy").show()
```

Before you begin

Install Python 2.7 or Python 3.5 or higher.

Procedure

1. Log into a Spark gateway node.
2. Ensure the required security token is authorized to compile and execute the workload (if your cluster is Kerberized).
3. Execute the script using the spark-submit command.

   ```
   spark-submit spark-demo.py --num-executors 3 --driver-memory 512m --exec
   utor-memory 512m --executor-cores 1
   ```

4. Go to the Spark History server web UI at http://<spark_history_server>:18088, and check the status and performance of the workload.

Using pyspark

About this task

Run your application with the pyspark or the Python interpreter.

Before you begin

Install PySpark using pip.

Procedure

1. Log into a Spark gateway node.
2. Ensure the required security token is authorized to compile and execute the workload (if your cluster is Kerberized).
3. Ensure the user has access to the workload script (python or shell script).
4. Execute the script using pyspark.

   ```
   pyspark spark-demo.py --num-executors 3 --driver-memory 512m --executor-
   memory 512m --executor-cores 1
   ```

5. Execute the script using the Python interpreter.

   ```
   python spark-demo.py
   ```

6. Go to the Spark History server web UI at http://<spark_history_server>:18088, and check the status and performance of the workload.

Running a job interactively

About this task

Procedure

1. Log into a Spark gateway node.
2. Ensure the required security token is authorized to compile and execute the workload (if your cluster is Kerberized).

3. Launch the “spark-shell”.
   For example:

```
 spark-shell --jars target/mylibrary-1.0-SNAPSHOT-jar-with-dependencies.
jar
```

4. Create a Spark context and run workload scripts.

```
cala> import org.apache.spark.sql.hive.HiveContext
scala> val sqlContext = new HiveContext(sc)
scala> sqlContext.sql("CREATE TABLE IF NOT EXISTS default.sales_spark_1(R
egion string, Country string,Item_Type string,Sales_Channel string,Order
_Priority string,Order_Date date,Order_ID int,Ship_Date date,Units_sold
string,Unit_Price string,Unit_cost string,Total_revenue string,Total_Cos
t string,Total_Profit string) row format delimited fields terminated by
','")
scala> sqlContext.sql("load data local inpath '/tmp/sales.csv' into table
default.sales_spark_1")
scala> sqlContext.sql("show tables")
scala> sqlContext.sql("select * from default.sales_spark_1 limit 10").s
how()
scala> sqlContext.sql ("select count(*) from default.sales_spark_1").show(
)
```

5. Go to the Spark History server web UI at http://<spark_history_server>:18088, and check the status and performance of the workload.

Post-migration tasks

After the workloads are executed on Spark 2.4, validate the output, and compare the performance of the jobs with CDH/HDP cluster executions.

After the workloads are executed on Spark 2.4, validate the output, and compare the performance of the jobs with CDH/HDP cluster executions. After you perform the post migration configurations, do benchmark testing on Spark 2.4.

Troubleshoot the failed/slow performing workloads by analyzing the application event logs/driver logs and fine tune the workloads for better performance.

For more information, see the following documents:

- https://spark.apache.org/docs/latest/sql-migration-guide.html
- https://spark.apache.org/releases/spark-release-2-4-0.html
- https://spark.apache.org/releases/spark-release-2-2-0.html
- https://spark.apache.org/releases/spark-release-2-3-0.html
- https://spark.apache.org/releases/spark-release-2-1-0.html
- https://spark.apache.org/releases/spark-release-2-0-0.html
- For additional information about known issues please also refer to:

  Known Issues in Cloudera Manager 7.4.4 | Cloudera Private Cloud

Spark 2.3 to Spark 2.4 Refactoring

Because Spark 2.3 is not supported on Cloudera, you need to refactor Spark workloads from Spark 2.3 on CDH or HDP to Spark 2.4 on Cloudera.

This document helps in accelerating the migration process, provides guidance to refactor Spark workloads and lists migration. Use this document when the platform is migrated from CDH or HDP to Cloudera.

Handling prerequisites

You must perform a number of tasks before refactoring workloads.

About this task

Assuming all workloads are in working condition, you perform this task to meet refactoring prerequistes.

Procedure

1. Identify all the workloads in the cluster (CDH/HDP) which are running on Spark 1.6 - 2.3.
2. Classify the workloads.

   Classification of workloads will help in clean-up of the unwanted workloads, plan resources and efforts for workload migration and post upgrade testing.

   Example workload classifications:

   - Spark Core (scala)
   - Java-based Spark jobs
   - SQL, Datasets, and DataFrame
   - Structured Streaming
   - MLlib (Cloudera AI)
   - PySpark (Python on Spark)
   - Batch Jobs
   - Scheduled Jobs
   - Ad-Hoc Jobs
   - Critical/Priority Jobs
   - Huge data Processing Jobs
   - Time taking jobs
   - Resource Consuming Jobs etc.
   - Failed Jobs

Identify configuration changes

3. Check the current Spark jobs configuration.

   - Spark 1.6 - 2.3 workload configurations which have dependencies on job properties like scheduler, old python packages, classpath jars and might not be compatible post migration.
   - In Cloudera, Capacity Scheduler is the default and recommended scheduler. Follow Fair Scheduler to Capacity Scheduler transition guide to have all the required queues configured in the Cloudera cluster post upgrade. If any configuration changes are required, modify the code as per the new capacity scheduler configurations.
   - For workload configurations, see the Spark History server UI http://spark_history_server:18088/history/<application_number>/environment/.
4. Identify and capture workloads having data storage locations (local and HDFS) to refactor the workloads post migration.
5. Refer to unsupported Apache Spark features, and plan refactoring accordingly.

Spark 2.3 to Spark 2.4 changes

A description of the change, the type of change, and the required refactoring provide the information you need for migrating from Spark 1.6 to Spark 2.4.

Empty schema not supported

Writing a dataframe with an empty or nested empty schema using any file format, such as parquet, orc, json, text, or csv is not allowed.

Type of change: Syntactic/Spark core

Spark 1.6 - 2.3

Writing a dataframe with an empty or nested empty schema using any file format is allowed and will not throw an exception.

Spark 2.4

An exception is thrown when you attempt to write dataframes with empty schema. For example, if there are statements such as df.write.format("parquet").mode("overwrite").save(somePath), the following error occurs: org.apache.spark.sql.AnalysisException: Parquet data source does not support null data type.

Action Required

Make sure that DataFrame is not empty. Check whether DataFrame is empty or not as follows:

```
if (!df.isEmpty) df.write.format("parquet").mode("overwrite").save("somePath
")
```

CSV header and schema match

Column names of csv headers must match the schema.

Type of change: Configuration/Spark core changes

Spark 1.6 - 2.3

Column names of headers in CSV files are not checked against the against the schema of CSV data.

Spark 2.4

If columns in the CSV header and the schema have different ordering, the following exception is thrown:java.lang.IllegalArgumentException: CSV file header does not contain the expected fields.

Action Required

Make the schema and header order match or set enforceSchema to false to prevent getting an exception. For example, read a file or directory of files in CSV format into Spark DataFrame as follows: df3 = spark.read.option("delimiter", ";").option("header", True).option("enforeSchema", False).csv(path)

The default "header" option is true and enforceSchema is False.

If enforceSchema is set to true, the specified or inferred schema will be forcibly applied to datasource files, and headers in CSV files are ignored. If enforceSchema is set to false, the schema is validated against all headers in CSV files when the header option is set to true. Field names in the schema and column names in CSV headers are checked by their positions taking into account spark.sql.caseSensitive. Although the default value is true,you should disable the enforceSchema option to prevent incorrect results.

Table properties support

Table properties are taken into consideration while creating the table.

Type of change: Configuration/Spark Core Changes

Spark 1.6 - 2.3

Parquet and ORC Hive tables are converted to Parquet or ORC by default, but table properties are ignored. For example, the compression table property is ignored:

```
CREATE TABLE t(id int) STORED AS PARQUET TBLPROPERTIES (parquet.compression
'NONE')
```

This command generates Snappy Parquet files.

Spark 2.4

Table properties are supported. For example, if no compression is required, set the TBLPROPERTIES as follows: (parquet.compression 'NONE').

This command generates uncompressed Parquet files.

Action Required

Check and set the desired TBLPROPERTIES.

Managed table location

Creating a managed table with nonempty location is not allowed.

Type of change: Property/Spark core changes

Spark 1.6 - 2.3

You can create a managed table having a nonempty location.

Spark 2.4

Creating a managed table with nonempty location is not allowed. In Spark 2.4, an error occurs when there is a write operation, such as df.write.mode(SaveMode.Overwrite).saveAsTable("testdb.testtable"). The error side-effects are the cluster is terminated while the write is in progress, a temporary network issue occurs, or the job is interrupted.

Action Required

Set spark.sql.legacy.allowCreatingManagedTableUsingNonemptyLocation to true at runtime as follows:

```
spark.conf.set("spark.sql.legacy.allowCreatingManagedTableUsingNonemptyLocat
ion","true")
```

Precedence of set operations

Set operations are executed by priority instead having equal precedence.

Type of change: Property/Spark SQL changes

Spark 1.6 - 2.3

If the order is not specified by parentheses, equal precedence is given to all set operations.

Spark 2.4

If the order is not specified by parentheses, set operations are performed from left to right with the exception that all INTERSECT operations are performed before any UNION, EXCEPT or MINUS operations.

For example, if your code includes set operations, such as INTERSECT , UNION, EXCEPT or MINUS, consider refactoring.

Action Required

Change the logic according to following rule:

If the order of set operations is not specified by parentheses, set operations are performed from left to right with the exception that all INTERSECT operations are performed before any UNION, EXCEPT or MINUS operations.

If you want the previous behavior of equal precedence then, set spark.sql.legacy.setopsPrecedence.enabled=true.

HAVING without GROUP BY

HAVING without GROUP BY is treated as a global aggregate.

Type of change: Property/Spark SQL changes

Spark 1.6 - 2,3

HAVING without GROUP BY is treated as WHERE. For example, SELECT 1 FROM range(10) HAVING true is executed as SELECT 1 FROM range(10) WHERE true, and and returns 10 rows.

Spark 2.4

HAVING without GROUP BY is treated as a global aggregate. For example, SELECT 1 FROM range(10) HAVING true returns one row, instead of 10, as in the previous version.

Action Required

Check the logic where having and group by is used. To restore previous behavior, set spark.sql.legacy.parser.havingWithoutGroupByAsWhere=true.

CSV bad record handling

How Spark treats malformations in CSV files has changed.

Type of change: Property/Spark SQL changes

Spark 1.6 - 2.3

CSV rows are considered malformed if at least one column value in the row is malformed. The CSV parser drops malformed rows in the DROPMALFORMED mode or outputs an error in the FAILFAST mode.

Spark 2.4

A CSV row is considered malformed only when it contains malformed column values requested from CSV datasource, other values are ignored.

Action Required

To restore the Spark 1.6 behavior, set spark.sql.csv.parser.columnPruning.enabled to false.

Spark 1.4 - 2.3 CSV example

A Spark 1.4 - 2.3 CSV example illustrates the CSV-handling change in Spark 2.6.

In the following CSV file, the first two records describe the file. These records are not considered during processing and need to be removed from the file. The actual data to be considered for processing has three columns (jersey, name, position).

```
These are extra line1
These are extra line2
10,Messi,CF
7,Ronaldo,LW
9,Benzema,CF
```

The following schema definition for the DataFrame reader uses the option DROPMALFORMED. You see only the required data; all the description and error records are removed.

```
schema=Structtype([Structfield(“jersy”,StringType()),Structfield(“name”,Stri
ngType()),Structfi
eld(“position”,StringType())])
df1=spark.read\
.option(“mode”,”DROPMALFORMED”)\
.option(“delimiter”,”,”)\
.schema(schema)\
.csv(“inputfile”)
df1.select(“*”).show()
```

Output is:

jersy	name	position
10	Messi	CF
7	Ronaldo	LW
9	Benzema	CF

Select two columns from the dataframe and invoke show():

```
df1.select(“jersy”,”name”).show(truncate=False)
```

jersy	name
These are extra line1	null
These are extra line2	null
10	Messi
7	Ronaldo
9	Benzema

Malformed records are not dropped and pushed to the first column and the remaining columns will be replaced with null.This is due to the CSV parser column pruning which is set to true by default in Spark 2.4.

Set the following conf, and run the same code, selecting two fields.

```
spark.conf.set(“spark.sql.csv.parser.columnPruning.enabled”,False)
```

```
df2=spark.read\
   .option(“mode”,”DROPMALFORMED”)\
   .option(“delimiter”,”,”)\
   .schema(schema)\
   .csv(“inputfile”)
   df2.select(“jersy”,”name”).show(truncate=False)
```

jersy	name
10	Messi
7	Ronaldo
9	Benzema

Conclusion: If working on selective columns, to handle bad records in CSV files, set spark.sql.csv.parser.columnPruning.enabled to false; otherwise, the error record is pushed to the first column, and all the remaining columns are treated as nulls.

Configuring storage locations

To execute the workloads in Cloudera, you must modify the references to storage locations. In Cloudera, references must be changed from HDFS to a cloud object store such as S3.

About this task

The following sample query shows a Spark 2.4 HDFS data location.

```
scala> spark.sql("CREATE TABLE IF NOT EXISTS default.sales_spark_2(Region st
ring, Country string,Item_Type string,Sales_Channel string,Order_Priority st
ring,Order_Date date,Order_ID int,Ship_Date date,Units_sold string,Unit_Pric
e string,Unit_cost string,Total_revenue string,Total_Cost string,Total_Profi
t string) row format delimited fields terminated by ','")
scala> spark.sql("load data local inpath '/tmp/sales.csv' into table defau
lt.sales_spark_2")
scala> spark.sql("select count(*) from default.sales_spark_2").show()
```

The following sample query shows a Spark 2.4 S3 data location.

```
scala> spark.sql("CREATE TABLE IF NOT EXISTS default.sales_spark_2(Region st
ring, Country string,Item_Type string,Sales_Channel string,Order_Priority st
ring,Order_Date date,Order_ID int,Ship_Date date,Units_sold string,Unit_Pric
e string,Unit_cost string,Total_revenue string,Total_Cost string,Total_Profi
t string) row format delimited fields terminated by ','")
```

```
scala> spark.sql("load data inpath 's3://<bucket>/sales.csv' into table de
fault.sales_spark_2")
scala> spark.sql("select count(*) from default.sales_spark_2").show()
```

Querying Hive managed tables from Spark

Hive-on-Spark is not supported on Cloudera. You need to use the Hive Warehouse Connector (HWC) to query Apache Hive managed tables from Apache Spark.

To read Hive external tables from Spark, you do not need HWC. Spark uses native Spark to read external tables. For more information, see the Hive Warehouse Connector documentation.

The following example shows how to query a Hive table from Spark using HWC:

```
spark-shell --jars /opt/cloudera/parcels/CDH/jars/hive-warehouse-connector-a
ssembly-1.0.0.7.1.4.0-203.jar --conf spark.sql.hive.hiveserver2.jdbc.url=jdb
c:hive2://cdhhdp02.uddeepta-bandyopadhyay-s-account.cloud:10000/default --co
nf spark.sql.hive.hiveserver2.jdbc.url.principal=hive/cdhhdp02.uddeepta-band
yopadhyay-s-account.cloud@Uddeepta-bandyopadhyay-s-Account.CLOUD
scala> val hive = com.hortonworks.hwc.HiveWarehouseSession.session(spark).b
uild()
scala> hive.executeUpdate("UPDATE hive_acid_demo set value=25 where key=4")
scala> val result=hive.execute("select * from default.hive_acid_demo")
scala> result.show()
```

Compiling and running Spark workloads

After modifying the workloads, compile and run (or dry run) the refactored workloads on Spark 2.4.

You can write Spark applications using Java, Scala, Python, SparkR, and others. You build jars from these scripts using one of the following compilers.

- Java (with Maven/Java IDE),
- Scala (with sbt),
- Python (pip).
- SparkR (RStudio)

Post-migration tasks

After the workloads are executed on Spark 2.4, validate the output, and compare the performance of the jobs with CDH/HDP cluster executions.

After the workloads are executed on Spark 2.4, validate the output, and compare the performance of the jobs with CDH/HDP cluster executions. After you perform the post migration configurations, do benchmark testing on Spark 2.4.

Troubleshoot the failed/slow performing workloads by analyzing the application event logs/driver logs and fine tune the workloads for better performance.

For more information, see the following documents:

- https://spark.apache.org/docs/latest/sql-migration-guide.html
- https://spark.apache.org/releases/spark-release-2-4-0.html
- https://spark.apache.org/releases/spark-release-2-2-0.html
- https://spark.apache.org/releases/spark-release-2-3-0.html
- https://spark.apache.org/releases/spark-release-2-1-0.html
- https://spark.apache.org/releases/spark-release-2-0-0.html
- For additional information about known issues please also refer to:

  Known Issues in Cloudera Manager 7.4.4 | Cloudera Private Cloud

Cloudera Public Cloud

CDP Public Cloud Upgrade Advisor

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Public Cloud upgrade advisor....4

Cloudera Public Cloud upgrades FAQ....4

Preparing for an upgrade....8

Upgrading to Cloudera Runtime 7.2.18....12
Identify cluster version details....13
Identify your upgrade path....16
Review the prerequisites....18
High-level upgrade steps....18
Step 1: Upgrade Data Lake and all Cloudera Data Hub clusters to the latest service pack and OS....18
Step 2: Upgrade FreeIPA to RHEL 8....20
Step 3: Upgrade Data Lake and Cloudera Data Hub clusters to the latest Cloudera Runtime 7.2.17 and the latest RHEL 8 OS image....20
Step 4: Resize Medium Duty Data Lake to Enterprise Data Lake....22
Step 5: Upgrade Data Lake and Cloudera Data Hub clusters from 7.2.17 to 7.2.18....23

Upgrading to Cloudera Runtime 7.3.1....24
Identify cluster version details....24
Identify your upgrade path....28
Review the prerequisites....29
High-level upgrade steps....29
Upgrade to a supported version....30
Upgrade database to PostgreSQL 14....30
Resize to Enterprise Data Lake....31
Upgrade Data Lake Cloudera Runtime version to 7.3.1. and OS to RHEL 8.10....31
Upgrade Cloudera Runtime version of Cloudera Data Hub clusters to 7.3.1....32
Upgrade OS version of Cloudera Data Hub clusters to RHEL 8.10....33

Upgrading from CentOS to RHEL....33
Prerequisites for upgrading from CentOS to RHEL....34
Upgrade clusters to RHEL....34

Upgrading from Medium Duty to Enterprise Data Lake....36

Upgrading from Spark 2 to Spark 3....36

Rolling upgrades....37

Cloudera Public Cloud upgrade advisor

Compute resources deployed via cloud services are generally considered transient in nature. With the separation of compute and storage, compute clusters in Cloudera can be stopped or decommissioned, while the data used or created by workloads running on these clusters generally remains accessible on persistent cloud storage.

There are some exceptions to the above, most importantly SDX metadata and cloud service-specific metadata. SDX metadata is stored in the Data Lake, while metadata specific to a cloud service may be stored in databases, local configurations, or even locally attached disks. This local storage can also be persistent (block storage volumes) or transient (ephemeral disks).

In cloud services where compute is elastic and state is transient, we need safeguards to protect all data that is not persistent, especially when changes are performed on the services themselves. In Cloudera Public Cloud, there can be several changes, most notably Runtime service pack upgrades, Runtime minor/major software version upgrades, and OS upgrades.

In general, there are two main approaches to upgrading cloud services:

1. Backup, re-create and restore
2. In-place upgrade

These two approaches have similarities: a prior backup should be performed, service endpoints should remain stable after the operation, and they should result in the same outcome. Cloudera Public Cloud supports both approaches. While the first approach may be convenient for simple data workloads, complex data applications and their custom data pipelines spanning multiple clusters may require an in-place upgrade path.

In this guide we will describe the high-level steps of performing in-place upgrade of Data Lake and Cloudera Data Hub clusters. For steps required for the backup and restore approach, refer to the respective documentation on backing up Data Lakes and Cloudera Data Hub and performing metadata restore (automated for Data Lake clusters only).

Cloudera Public Cloud upgrades FAQ

During the preparation for an upgrade, it is recommended to carefully review the questions and answers below.

General questions related to upgrading Cloudera

What is the length of the available maintenance window?

Currently, Data Lake backup and restore requires a maintenance window, where no metadata changes occur. Furthermore, Cloudera Data Hub clusters need to be stopped during an upgrade.

The Cloudera Public Cloud environment does not need to be upgraded in one go: you may opt to upgrade the Data Lake and all attached Data Hubs together, or start with the Data Lake upgrade only and perform individual Data Hub upgrades consecutively. Whether or not a Cloudera Data Hub upgrade is required depends on the Cloudera Runtime version of the Cloudera Data Hub cluster:

- If your Cloudera Data Hub clusters are on Cloudera Runtime version 7.2.15 or earlier, they must run the same major/minor version of Cloudera Runtime as the Data Lake. In this scenario, after a Data Lake upgrade you are required to upgrade any Cloudera Data Hub clusters that are 7.2.15 or earlier to the same version as the Data Lake.
- If your Cloudera Data Hub clusters are on Cloudera Runtime version 7.2.16 or later, they are compatible with newer versions of the Data Lake (7.2.17 or later). You can independently upgrade your Cloudera Data Hub clusters at a later time if you choose to, though it is not required.

What type of upgrade is required?

Currently, there are three types of upgrades available to Data Lake and Cloudera Data Hub clusters: service pack upgrades; minor/major version upgrades; and OS upgrades. Service pack and minor/major version upgrades install a newer version of Cloudera Manager and/or Cloudera Runtime. OS upgrades for Data Lakes and Cloudera Data Hub clusters are complementary and will bring the image of the cluster hosts to a newer version. If you plan to also perform an OS upgrade, plan the maintenance window accordingly.

Are ephemeral disks used for user or workload-related persistent data?

Major/minor version upgrades as well as service pack upgrades will bring Cloudera Manager and Cloudera Runtime to the selected version without impacting the underlying VM. However, OS upgrades will recreate the underlying VM with a fresh image, which results in the loss of any data stored on ephemeral disks.

If you are currently storing user or workload-related data on volumes using ephemeral disks, please reach out to Cloudera support while planning for the upgrade.

What Data Hub cluster templates are in use? Are you using custom templates?

Check whether in-place upgrade is supported for your built-in or custom Cloudera Data Hub template. Depending on the type and version of the Cloudera Data Hub cluster, additional backup steps, manual configuration changes or post-upgrade steps may be required. Check specific steps for upgrading the OS if you use Cloudera Flow Management. Cloudera Operational Database clusters have a different upgrade process.

What is the size of the SDX / Data Lake metadata?

SDX metadata includes the Hive Metastore database, Ranger audit log index, as well as Atlas metadata. If you are planning to perform a Data Lake backup before an upgrade (which is recommended), prepare your maintenance window accordingly. Cloudera supports skipping the backup of certain metadata to reduce the time required for backup and restore operations.

Are you using Data Services?

If you have deployed Cloudera Data Engineering, Cloudera Data Warehouse, Cloudera DataFlow, or Cloudera AI in your environment, be sure to check the *Preparing for an upgrade* topic to verify compatibility between the data service version and the Data Lake version or desired features/Cloudera Runtime services.

Questions related to upgrading to Cloudera Runtime 7.2.18

Will upgrading to Cloudera Runtime 7.2.18 and changing from CentOS to RHEL 8 cost Cloudera customers money?

Upgrading to Cloudera Public Cloud Runtime version 7.2.18 and changing the operating system from CentOS 7 to RHEL 8 will not incur additional costs, either to your organization or to the Cloud Service Provider. This update is designed for a frictionless transition and continued support, without financial impact.

What are the key features of Cloudera Runtime 7.2.18?

- RHEL 8 is set as the default operating system, in anticipation of the CentOS 7 sunset on June 30, 2024, to ensure a modern and fully supported infrastructure.
- Rolling upgrades allow for seamless updates of services without operational interruptions.
- Iceberg support is now fully integrated with Atlas, enriching data management capabilities with comprehensive data lineage support.
- The transition from Medium Duty to Enterprise Data Lakes enhances performance and scalability, aligning with advanced workload requirements.
- Amazon S3 Express One Zone support provides a fast and cost-effective data storage option.

What are Rolling Upgrades, and how do they affect my operations?

Rolling Upgrades allow you to upgrade without causing any interruption to ongoing operations. This means customers can continue using their services while the upgrade process is underway. While Rolling Updates are not available for all services, key services including Data Lakes, Cloudera Operational Database, and Cloudera Streams Messaging Data Hub clusters now support this feature, significantly enhancing operational efficiency and minimizing disruption during upgrades.

Why is the upgrade to Cloudera Runtime 7.2.18 recommended?

Upgrading to Cloudera Runtime 7.2.18 is recommended for several reasons:

- Cloudera Runtime 7.2.18 transitions to RHEL for modern, fully supported Linux infrastructure.
- Cloudera Runtime 7.2.18 introduces rolling upgrades for key services.
- Cloudera Runtime 7.2.18 integrates Iceberg with Atlas for enhanced data management and data lineage.
- Cloudera Runtime 7.2.18 supports Amazon S3 Express One Zone for cost-effective, high-speed storage.

How does the transition to RHEL 8 benefit Cloudera customers?

The transition to RHEL 8 benefits Cloudera customers by ensuring a modern, fully supported Linux infrastructure that meets the latest industry standards. It enhances security, streamlines InfoSec approval processes, and offers greater automation capabilities. This transition supports advanced functionality such as Generative AI and real-time streaming, providing faster time-to-value with no additional costs.

Can you explain the Iceberg support with Atlas integration?

The integration of Iceberg with Atlas in the 7.2.18 update completes Cloudera's Iceberg integration story by providing comprehensive data lineage support. With Atlas, users gain visibility into the lineage of their Iceberg data. This enhancement enriches the platform's ability to manage and understand data across its lifecycle, facilitating better data governance and compliance.

What happens if a customer does NOT upgrade to RHEL 8 by June 30, 2024?

If you do not upgrade to RHEL 8 by the time it is deprecated, Cloudera will still accept support cases. We do ask customer account teams to file for an extension with CentOS. However, Cloudera will not publish OS patches or CVE fixes for CentOS-based images after June 2024.

How do I prepare my environment for upgrading to Cloudera Runtime 7.2.18?

To prepare your environment for upgrading to 7.2.18, check out Upgrading to Cloudera Runtime 7.2.18 for more information. Here you will find instructions on how to identify cluster versions, identify your upgrade path, and more.

How do I prepare my environment for upgrading to RHEL 8?

To prepare your environment for upgrading to RHEL 8, follow the guidance provided in Upgrading from CentOS to RHEL.

Is there a recommended upgrade path for users on various Cloudera Runtime versions?

Yes, and Cloudera offers documentation that guides you through the upgrade process. Check out Upgrading to Cloudera Runtime 7.2.18 for more information.

Can I roll back to a previous Cloudera Runtime version after upgrading to 7.2.18?

No, rolling back to a previous version after upgrading to 7.2.18 is not supported due to compatibility risks. Should you run into any errors or issues, Cloudera support is here to help.

Questions related to upgrading to Cloudera Runtime 7.3.1

What are the key features of Cloudera Runtime 7.3.1?

- Unified Runtime for Cloudera Public Cloud and Cloudera Private Cloud.
- RHEL 8.10 is set as the default operating system, while RHEL 8.8 continues to be supported.
- Python 3.9 is set as the new default version.

- OpenJDK 17 becomes the new default Java runtime version, new clusters will be launched with this version.

  OpenJDK 11 is no longer supported, and OpenJDK 11 clusters cannot be upgraded to Cloudera Runtime 7.3.1.

  After upgrading clusters from Runtime 7.2.x to 7.3.1, the JDK version remains the same, that is, JDK8.

What features will no longer be available after the upgrade to Cloudera Runtime 7.3.1?

- Customers need to rebase their applications to Spark 3 as Spark 2 is no longer supported and clusters with Spark 2 will not be allowed to upgrade toCloudera Runtime 7.3.1.
- Zeppelin service is turned off for Cloudera Runtime 7.3.1 clusters and gets removed automatically during the upgrade.
- Cloudera Data Engineering 1.23 and earlier versions are not supported on Cloudera Runtime 7.3.1

Why is the upgrade to Cloudera Runtime 7.3.1 recommended?

Upgrading to Cloudera Runtime 7.3.1 is recommended because the Cloudera Runtime 7.3.1 unified Runtime provides seamless portability of workloads across cloud and on-prem environments without rewrite.

How do I prepare my environment for upgrading to Cloudera Runtime 7.3.1?

To prepare your environment for upgrading to version 7.3.1, check out Upgrading to Cloudera Runtime 7.3.1 for more information. Here you will find instructions on how to identify cluster versions, identify your upgrade path, and more.

Is there a recommended upgrade path for users on various Cloudera Runtime versions?

Yes, and Cloudera offers documentation that guides you through the upgrade process. Check out Upgrading to Cloudera Runtime 7.3.1 for more information.

Can I roll back to a previous Cloudera Runtime version after upgrading to 7.3.1?

No, rolling back to a previous version after upgrading to 7.3.1 is not supported due to compatibility risks. Should you run into any errors or issues, Cloudera support is here to help.

How do I prepare my Cloudera Data Hub cluster for Spark 2 to Spark 3 migration?

If you have a Cloudera Data Engineering Data Hub cluster, during the Cloudera Runtime upgrade process to version 7.3.1, you need to remove Spark 2 while creating a new cluster with Spark 3 and migrate your data. Depending on your environment and Spark applications, besides the cluster upgrade tasks, you need to perform Spark application migration tasks and sidecar migration tasks for the Cloudera Data Hub cluster in multiple steps. For details, see Upgrading Apache Spark.

What happens to the Zeppelin service?

Zeppelin service is turned off for Cloudera Runtime 7.3.1 clusters and gets removed automatically during the upgrade. Therefore, you need to back up your data from the Zeppelin service before starting the Cloudera Data Hub upgrade to Runtime 7.3.1.

How do I back up data from the Zeppelin service?

You can back up Zeppelin data, to ensure all necessary components are preserved.

1. **Interpreter Configuration**: Copy the interpreter.json file located in the zeppelin_home/conf folder. This file contains the interpreter settings crucial for Zeppelin operations.
2. **Notebooks**: Back up the entire notebooks folder from the configured repository. This folder includes all Zeppelin notebooks.
3. **Cloudera Manager Configuration**: Access the Cloudera Manager UI, locate the Zeppelin configurations and copy any custom configurations set for the site.xml and env.sh files. Ensure any safety valve variables added in Cloudera Manager are also documented as part of the backup.

Related Information

Preparing for an upgrade

Preparing for an upgrade

Upgrading Cloudera Public Cloud consists of two major steps: upgrading the Data Lake within an environment and then potentially upgrading the attached Data Hubs. If applicable, you can upgrade any data services after the Data Lake upgrade.

FreeIPA upgrades

You should periodically upgrade the environment (FreeIPA cluster) to ensure that you are running the latest OS-level security patches, but this is not required at the same time as upgrading the Data Lake and Cloudera Data Hub clusters. While FreeIPA upgrade can be performed without a downtime, Cloudera recommends to perform this separately from other operations, preferably before upgrading the Data Lake and Cloudera Data Hub clusters. See upgrade the FreeIPA cluster.

Data Lake upgrade workflow

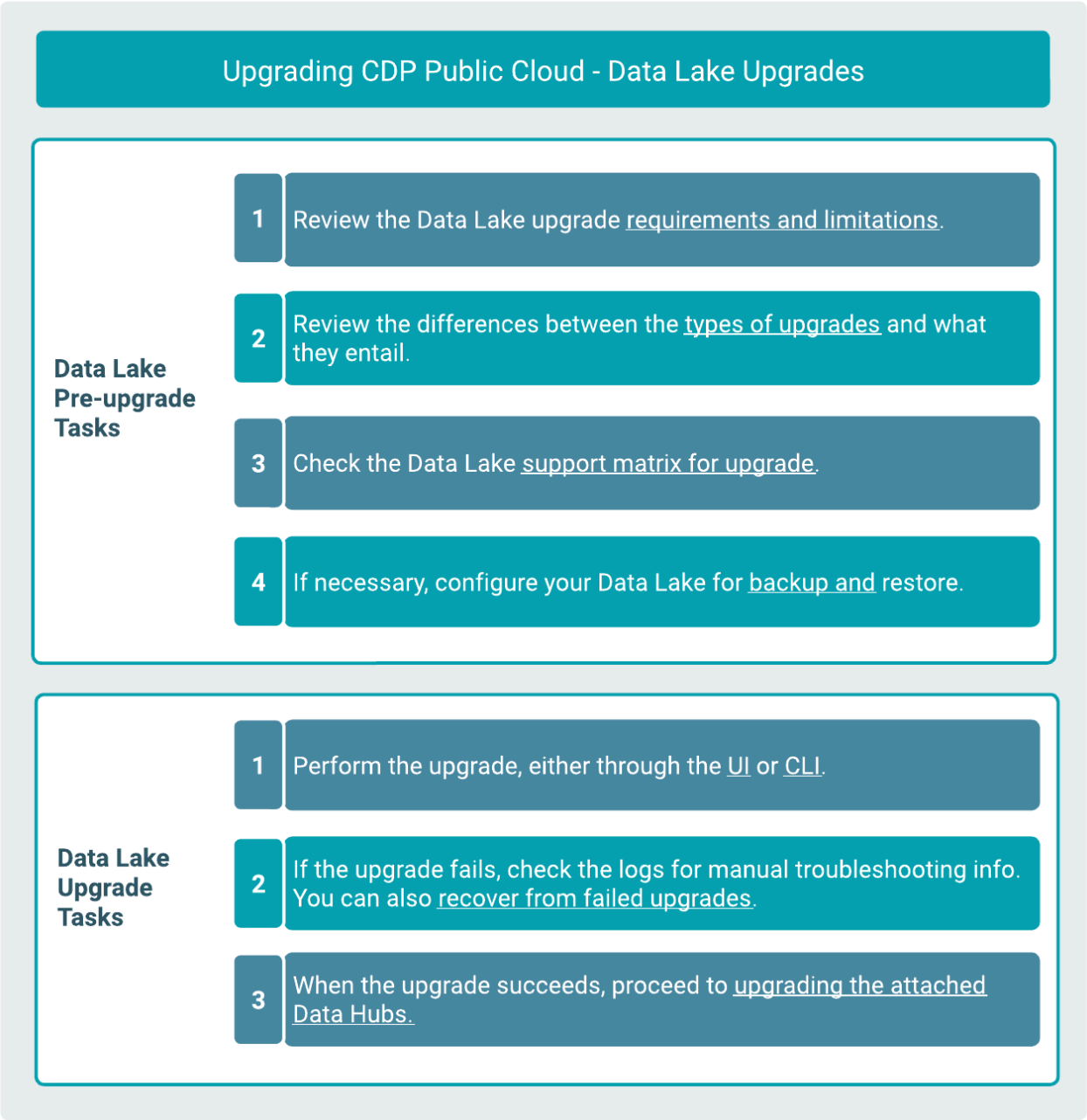

Pre-upgrade tasks

1. Review the Data Lake upgrade requirements and limitations.
2. Carefully review the differences between the types of upgrades and what they entail.
3. Check the Data Lake support matrix for upgrade to verify which Runtime versions you can upgrade to and from.
4. If you have not configured your Data Lake for backup and restore, you will need to do so. The backup and restore process is integrated into the upgrade flow automatically, but successful upgrade requires that the correct IAM policies exist on the DATALAKE_ADMIN_ROLE and RANGER_AUDIT_ROLE (for backup), and the DATALAKE_ADMIN_ROLE, RANGER_AUDIT_ROLE, and LOG_ROLE (for restore).

   If your roles are not configured correctly, Cloudera will not be able to write the backup to the BACKUP_LOCATION_BASE path of your cloud storage.

5. If you are performing the backup manually (as opposed to the integrated backup available during the upgrade process), you can launch the Data Lake backup from the UI or CLI. When using the CLI, you can specify to skip certain backup actions (skip HMS, Atlas metadata or Ranger audit log index backup). You can monitor the backup process using the CLI.
6. From the Data Lake UI, run the Validate and Prepare option to check for any configuration issues and begin the Cloudera Runtime parcel download and distribution. Using the validate and prepare option does not require downtime and makes the maintenance window for an upgrade shorter. Validate and prepare also does not make any changes to your cluster and can be run independently of the upgrade itself. Although you can begin the upgrade without first running the validate and prepare option, using it will make the process smoother and the downtime shorter. (The parcels that are downloaded and distributed by the validate and prepare option are specific to the Runtime version that you have selected, so if you use validate and prepare and then decide to upgrade to a different Runtime version instead, you will need to re-run validate and prepare. Be aware that if you use validate and prepare for multiple major/minor Runtime versions, the parcels for different versions are not cleaned up and may saturate the disk. These parcels are cleaned up only once the upgrade is complete.)

Data Lake upgrade tasks

1. Perform the upgrade, either through the UI or CLI. The type of upgrade that you perform will depend on whether a major/minor or service pack version of Runtime is available. A new OS image may also be available to upgrade to.
2. If the upgrade fails, check the logs for manual troubleshooting info. You can also recover from failed upgrades.
3. When the upgrade succeeds, proceed to upgrading the attached Cloudera Data Hub clusters if required.

Data Services upgrades

Any data services in use should be upgraded after the Data Lake upgrade. Some data services have their own compatibility matrix with different versions of the Data Lake. Other data services contain features that may not be compatible with every Data Lake (Cloudera Runtime) version. Refer to the data services documentation for information on Data Lake compatibility and upgrading these services.

- Cloudera AI and Data Lake Compatibility & Cloudera AI service with Data Lake upgrades
- Cloudera Data Engineering and Data Lake Compatibility
- Cloudera DataFlow feature support matrix
- Cloudera Data Warehouse version mapping

Cloudera Data Hub upgrade workflow

Upgrading CDP Public Cloud - Data Hub Upgrades

Data Hub Pre-upgrade Tasks

1. Carefully review the differences between the types of upgrades and what they entail
2. Check that the cluster that you want to upgrade is supported.

Data Hub Major/Minor Upgrade Tasks

1. Backup the cluster data and CM configurations.
2. Perform the upgrade, either through the UI or CLI. (Operational database clusters have a different process.)
3. Complete any post-upgrade tasks for the type of cluster that you upgraded.
4. For DE clusters, use the CM UI to add any configs that were not added during the upgrade.
5. If the upgrade fails, check the Event log and the troubleshooting section.

Data Hub Maintenance Upgrade Tasks

1. Backup the cluster data and CM configurations.
2. Perform the upgrade, either through the UI or CLI.
3. If the upgrade fails, check the Event log and the troubleshooting section.

Pre-upgrade tasks

1. Carefully review the differences between the types of upgrades and what they entail.
2. Check the Cloudera Data Hub upgrade support matrix to verify that the Cloudera Runtime versions you want to upgrade to and from are supported for your clusters.
3. From the Cloudera Data Hub UI, run the Validate and Prepare option to check for any configuration issues and begin the Cloudera Runtime parcel download and distribution. Using the validate and prepare option does not require downtime and makes the maintenance window for an upgrade shorter. Validate and prepare also does not make any changes to your cluster and can be run independently of the upgrade itself. Although you can begin the upgrade without first running the validate and prepare option, using it will make the process smoother and the downtime shorter. (The parcels that are downloaded and distributed by the validate and prepare option are specific to the Runtime version that you have selected, so if you use validate and prepare and then decide to upgrade to a different Runtime version instead, you will need to re-run validate and prepare. Be aware that if you use validate and prepare for multiple major/minor Runtime versions, the parcels for different versions are not cleaned up and may saturate the disk. These parcels are cleaned up only once the upgrade is complete.)

Major/minor Runtime version upgrade tasks

1. Backup the cluster data and Cloudera Manager configurations.
2. Perform the upgrade, either through the UI or CLI. (Cloudera Operational Database clusters have a different process.)
3. Complete any post-upgrade tasks for the type of cluster that you upgraded.
4. For Cloudera Data Engineering clusters, use the Cloudera Manager UI to add any configs that were not added during the upgrade.
5. If the upgrade fails, check the Event log and the troubleshooting section.
6. Complete any post-upgrade tasks for the type of cluster that you upgraded.
7. For Cloudera Data Engineering clusters, use the Cloudera Manager UI to add any configs that were not added during the upgrade.
8. If the upgrade fails, check the Event log and the troubleshooting section.

Service pack upgrade tasks

1. Backup the cluster data and Cloudera Manager configurations.
2. Perform the upgrade, either through the UI or CLI.
3. If the upgrade fails, check the Event log and the troubleshooting section.

OS upgrade tasks

1. Review the Before you begin section to verify that there is no data belonging to NiFi or NiFi Registry on the root disk of the VM. Note that during an OS upgrade, any data on the root volume (parcels, service logs, custom software) will be lost.
2. Unlike the Data Lake upgrade, OS upgrades are not integrated in the larger upgrade flow and must be performed separately, either through the UI or CLI.

Upgrading to Cloudera Runtime 7.2.18

Cloudera Runtime 7.2.18 introduces significant new features and improvements. As a result, if you are planning to upgrade the Cloudera Runtime version in your existing Data Lake or Cloudera Data Hub clusters to 7.2.18, you might be required to perform this in multiple steps.

Some of the important Cloudera Runtime 7.2.18 features include:

- RHEL 8 images as CentOS reaches End of Life (EOL)

  As of June 30, 2024, CentOS reaches End of Life (EOL), and consequently, Cloudera Runtime 7.2.18 supports RHEL 8-based images only. New deployments of Data Lakes and Cloudera Data Hub clusters with Cloudera Runtime 7.2.18 and upgrades to 7.2.18 are only possible with RHEL 8. Data Lake and Cloudera Data Hub clusters

running Cloudera Runtime 7.2.17 support both CentOS 7 and RHEL 8. Earlier Cloudera Runtime versions support CentOS 7 only. Cloudera will not publish any updates or fixes for CentOS-based images after June 2024.

For Azure environments, Cloudera only provides RHEL 8 images via Azure Marketplace.

- Discontinuation of Medium Duty Data Lake

  Starting with Cloudera Runtime 7.2.18, Medium Duty Data Lake is discontinued as it does not support rolling upgrades and horizontal scaling, which have been introduced with the Enterprise Data Lake . While earlier Cloudera Runtime versions support Medium Duty Data Lake (7.2.17 supports Enterprise, Medium Duty, Light Duty, and older versions support Medium Duty and Light Duty), 7.2.18 only supports Enterprise and Light Duty Data Lakes.

Use the following documentation to identify your Data Lake cluster's version details and then based on that determine your upgrade path:

Identify cluster version details

Identify your Data Lake and Cloudera Data Hub cluster's Cloudera Runtime version, Data Lake, Cloudera Data Hub, and FreeIPA image version and operating system (OS) type, and Data Lake shape.

Your upgrade path to versions 7.2.18 or 7.3.1 may depend on the current Cloudera Runtime version, image version, and OS of your cluster. Cloudera recommends that you identify your version information as follows:

1. Identify cluster's Cloudera Runtime version on page 13
2. Identify cluster's image version, image creation date, OS type on page 14
3. Identify Data Lake shape on page 16

Identify cluster's Cloudera Runtime version

Follow these instructions to identify your Data Lake and Cloudera Data Hub's Cloudera Runtime version. You should follow these steps separately for each cluster.

Steps

For CDP UI

1. In the Cloudera Management Console, navigate to the details page of the Data Lake or Cloudera Data Hub cluster that you would like to upgrade.
2. In the Cloudera Manager Info section find the field called "Runtime version":

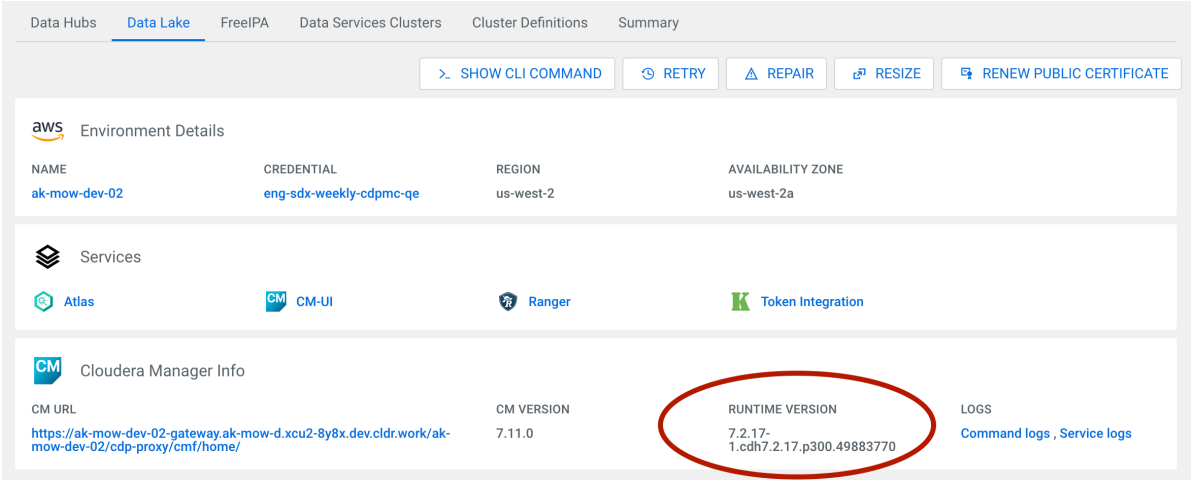

For CDP CLI

Get the Cloudera Runtime version of your Data Lake from the CDP CLI by using the following command:

```
cdp datalake describe-datalake --datalake-name <DL-NAME>
```

Get the Cloudera Runtime version of your Cloudera Data Hub from the CDP CLI by using the following command:

```
cdp datahub describe-cluster --cluster-name <DH-NAME>
```

The version information is displayed in the following format:

7.2.17-1.cdh7.2.17.p200.46967063

The highlighted digits show the current Cloudera Runtime version and the number after p indicates the current Service Pack version. In the above example, the cluster is using Cloudera Runtime version 7.2.17.200.

Identify cluster's image version, image creation date, OS type

Follow these instructions to identify Data Lake and Cloudera Data Hub image version and OS type. You should follow these steps separately for each cluster.

Steps

For CDP UI

1. In the Cloudera Management Console, navigate to the details page of the Data Lake or Cloudera Data Hub cluster that you would like to upgrade.
2. Scroll down to access the left navigation pane.
3. From the left navigation pane, select Image Details:

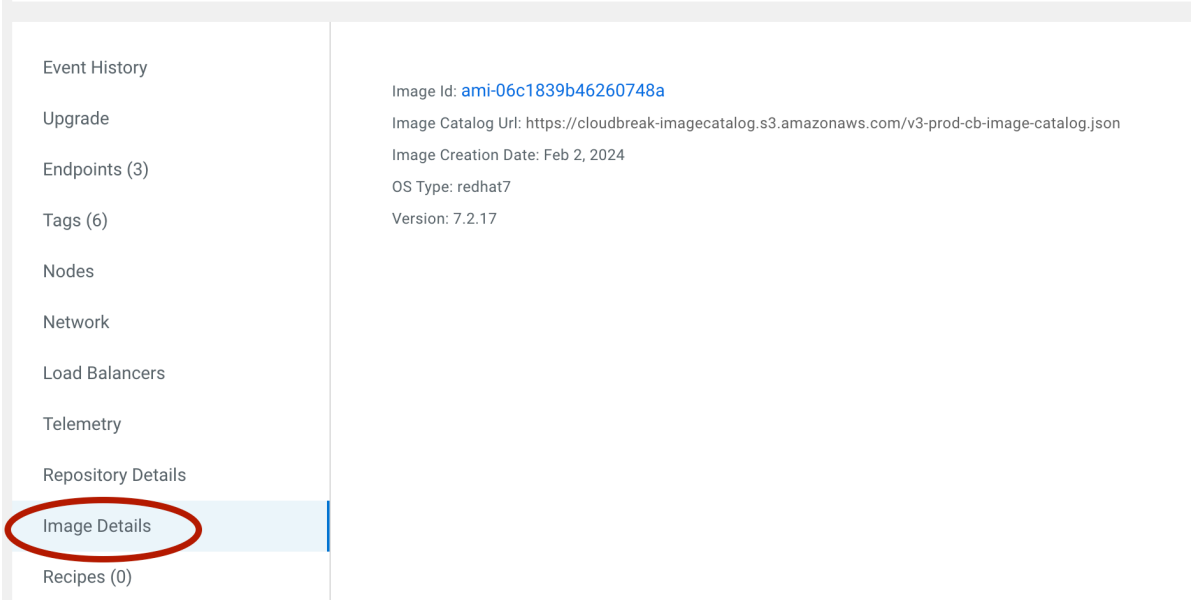

4. You will see information similar to the following:

```
Image Id: ami-0e95ecddb7fc75ea8
Image Catalog Url: https://cloudbreak-imagecatalog.s3.amazonaws.com/v3-
prod-cb-image-catalog.json
Image Creation Date: Jan 9, 2024
OS Type:redhat7
```

```
Version: 7.2.17
```

Under OS Type:

- If you see “redhat7”, your cluster is using a CentOS 7 OS image. This is a known issue.
- If you see “redhat8”, your cluster is using a RHEL 8 OS image.

Clicking on image ID will redirect you to the image catalog and provide you with further information, including the actual OS Type:

Details of 7ee9d8bc-a133-456a-a6ca-80b3cb93c9f7 image

UUID	7ee9d8bc-a133-456a-a6ca-80b3cb93c9f7	**Description**	7.2.17.300 Service Pack Release
OS Type	CentOS7	**Stack Name**	Cloudera Runtime
Cluster Manager Version	7.11.0	**Stack Version**	7.2.17
Created On	Feb 2, 2024	**Published On**	Feb 13, 2024

For CDP CLI

Get Data Lake’s image version from the Cloudera CLI by using the following command:

```
cdp datalake upgrade-datalake --show-latest-available-image-per-runtime --
datalake-name <DL-NAME>
```

Get Cloudera Data Hub cluster’s image version from the CDP CLI by using the following command:

```
cdp datahub upgrade-cluster --show-latest-available-image-per-runtime --
cluster-name <DH-NAME>
```

The output of this command looks like this, providing you with the OS information:

```
{
 "targetImage": {
 "catalog": "https://cloudbreak-imagecatalog.s3.amazonaws.com/v3-prod-fr
eeipa-image-catalog.json",
 "id": "a5b2580e-7487-4ee6-a80f-d991fa00cd13",
 "os": "redhat8",
 "imageName": "ami-0935d849de6a25e45",
 "date": "2024-01-09"
 },
 "originalImage": {
 "catalog": "https://cloudbreak-imagecatalog.s3.amazonaws.com/v3-prod-f
reeipa-image-catalog.json",
 "id": "e1f92688-3773-47dd-89ff-561933540108",
 "os": "centos7",
 "imageName": "ami-0db1b7000c8a27fd4",
 "date": "2023-11-22"
 },
 "operationId": "3a50f30f-33f3-4f81-b9bb-30c119023ff0"
}
```

Follow these instructions to identify FreeIPA OS type.

Steps

1. In the Cloudera Management Console, navigate to the environment details page,.
2. Click on the FreeIPA tab.
3. In the Nodes tab (which is open by default), expand one section corresponding to an instance ID.

4. Under Image Details, you can find the OS Type:

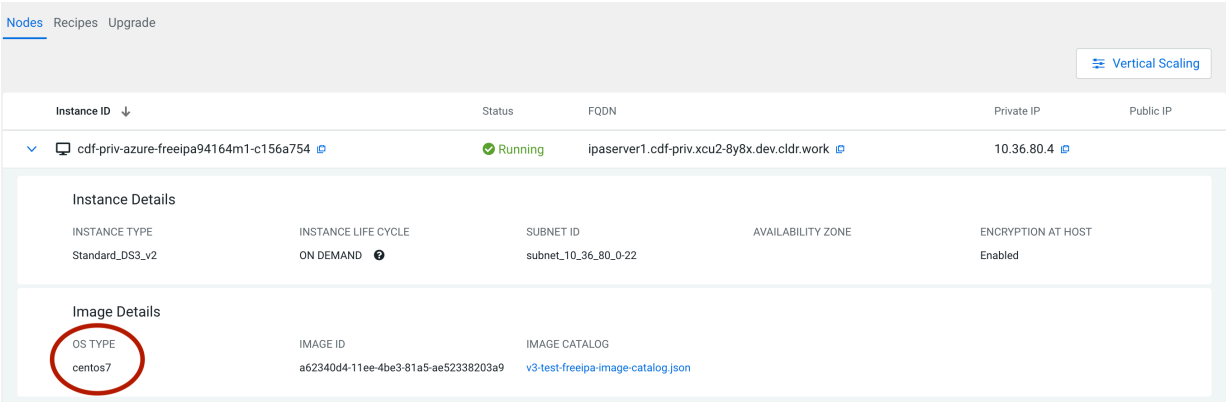

Identify Data Lake shape

Follow these instructions to identify Data Lake's shape.

For CDP UI

1. In the Cloudera Management Console, navigate to the details page of the Data Lake that you would like to upgrade:

2. Find the Scale field. There are three possible values: "Light Duty", "Medium Duty HA" and "Enterprise Data Lake".

For CDP CLI

Get the Data Lake shape from the CDP CLI by using the following command:

```
cdp datalake describe-datalake –datalake-name <DL-NAME>
```

Find the "scale" field in the output of this command. There are three possible values: "Light Duty", "Medium Duty HA" and "Enterprise Data Lake".

Identify your upgrade path

Refer to the following diagram for a high-level overview of the upgrade steps. Your starting point will vary, depending on the Runtime version and image version that your environment, Data Lake, and Cloudera Data Hub clusters are using.

The high-level upgrade steps are presented horizontally, from step 1 to step 5 and under each step you can see:

- Operations: A list of the actual operations that are performed as part of this step
- Result: The resulting state after these operations have been performed

For example, if your clusters are using Cloudera Runtime 7.2.16 and if Step 1 is applicable to your clusters, you would be performing a Data Lake upgrade and for each of your Cloudera Data Hub clusters a service pack upgrade and an OS upgrade in Step 1. If you started with Cloudera Runtime 7.2.16, after performing the operations in Step 1 you would have your Data Lake and all Cloudera Data Hub clusters upgraded to the latest service pack of Cloudera Runtime 7.2.16 and your cluster OS would be patched with up-to-date Python 3 binaries.

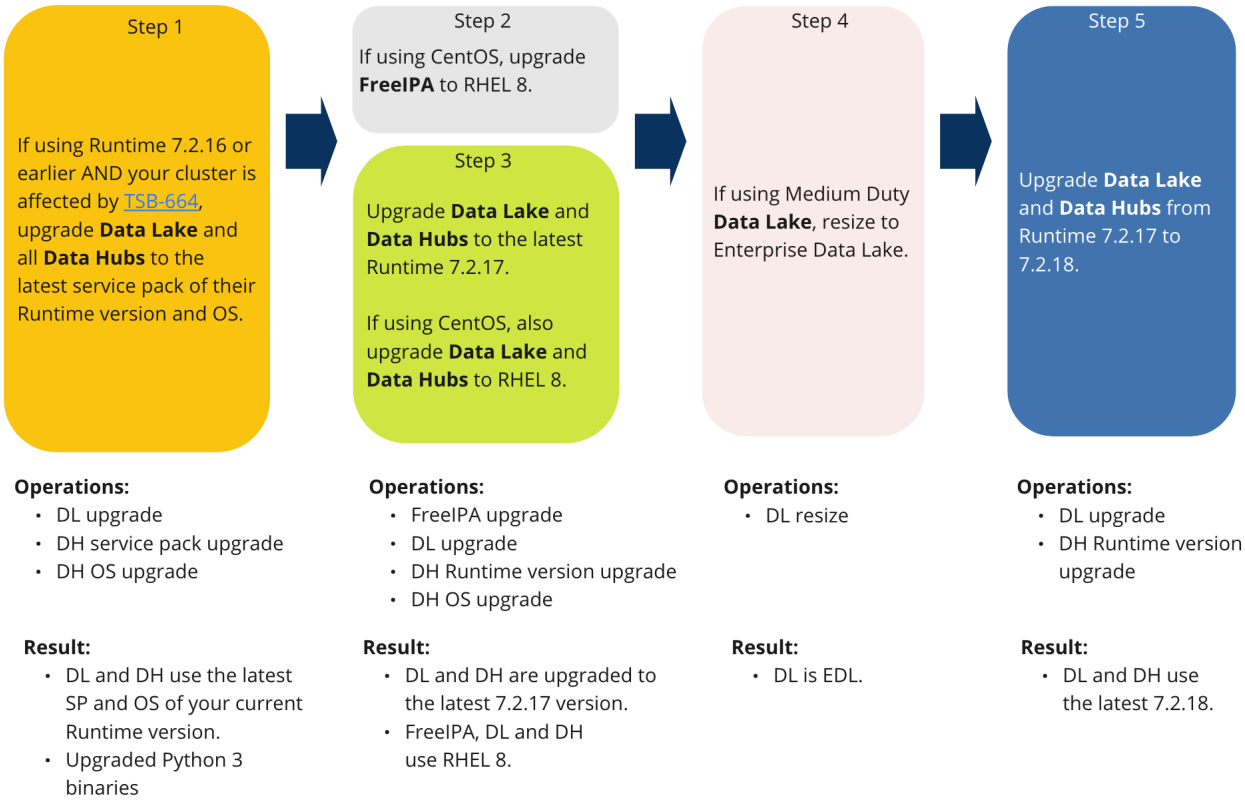

As presented in the diagram, you may be required to perform the following operations:

- Runtime service pack upgrades for your Data Lake and each Cloudera Data Hub cluster
- Major/Minor Runtime and OS upgrades for your Data Lake and each Cloudera Data Hub cluster
- CentOS to RHEL 8 upgrade for your FreeIPA, Data Lake, and each Cloudera Data Hub cluster
- Data Lake resize

For example, if your clusters are using Cloudera Runtime 7.2.16 or earlier and are impacted by Python 3.8 dependency described in TSB-664, you need to start at Step 1, but if this does not apply to you, you may start at a later step. Furthermore, depending on the operating system that your clusters are using, the CentOS to RHEL 8 upgrade steps (which are part of in Step 2) may not apply to you. and if you are not using Medium Duty Data Lake, you skip Enterprise Data Lake resize step (Step 4).

Important:

- Each of these operation has its own prerequisites and postrequisites so in addition to the actual upgrade steps, the prerequisites and postrequisites need to be performed as well.
- You will need to upgrade all Data Lakes and Cloudera Data Hub clusters separately, first the Data Lakes and then each Cloudera Data Hub cluster.
- You can perform each upgrade operation either from the Upgrade UI of the respective cluster or using the CDP CLI.
- The upgrade user interface in the Cloudera Management Console shows you the next Data Lake and Cloudera Data Hub upgrade step available, so you can refer to it as a guideline.

USEFUL LINKS:

- Data Lake upgrade

- Cloudera Data Hub upgrade
- Rolling upgrades
- Data Lake resizing
- Upgrading from CentOS to RHEL

Review the prerequisites

Before you start upgrading your clusters, ensure that you meet the prerequisites for each upgrade that is required.

Refer to the following documentation for detailed prerequisites:

- Review Data Lake Upgrade: Before you begin and Cloudera Data Hub Upgrade: Limitations and Prerequisites.
- If you are upgrading a Data Lake, use the Backup and Restore functionality. This will allow you to restore the SDX metadata if required.
- Certain Data Hubs might require certain pre-upgrade and post-upgrade steps. Review the related documentation before upgrading Data Hubs.
- If you are upgrading from CentOS to RHEL, review the Prerequisites for upgrading from CentOS to RHEL.
- If your upgrade path involves a Data Lake resize, refer to Review Data Lake resizing: Prerequisites.

High-level upgrade steps

Refer to this section for high-level upgrade steps and links to more detailed upgrade documentation. This document is not meant to provide you with complete upgrade steps, but rather it outlines the general upgrade path and provides you with the links to existing upgrade documentation.

Important:

- Each of these operations has its own prerequisites and postrequisites so in addition to the actual upgrade steps, the prerequisites and postrequisites need to be performed as well.
- You will need to upgrade all Data Lakes and Cloudera Data Hub clusters separately, first the Data Lakes and then each Data Hub.
- You can perform each upgrade operation either from the Upgrade UI of the respective cluster or using the CDP CLI.
- The upgrade user interface in the Cloudera Management Console shows you the next Data Lake and Cloudera Data Hub upgrade step available, so you can refer to it as a guideline.

1. Upgrade Data Lake and all Cloudera Data Hub clusters to the latest service pack and OS
2. Upgrade FreeIPA to RHEL 8
3. Upgrade Data Lake and Cloudera Data Hub clusters to the latest Cloudera Runtime 7.2.17 and the latest RHEL 8 OS image
4. Resize Medium Duty Data Lake to Enterprise Data Lake
5. Upgrade Data Lake and Cloudera Data Hub clusters from 7.2.17 to 7.2.18

Step 1: Upgrade Data Lake and all Cloudera Data Hub clusters to the latest service pack and OS

Upgrade Data Lake and Cloudera Data Hub to the latest service pack of their Cloudera Runtime version.

Important:

This step applies only to Data Lake and Cloudera Data Hub clusters running Cloudera Runtime 7.2.16 or earlier AND using an OS image that has been published before March 2023 (that is, clusters impacted by Python 3.8 dependency described in TSB-664).

Another way to identify whether you are required to perform this step is opening the Upgrade UI and verifying that 7.2.17 is available from the drop-down as an upgrade target. If it is, you can skip to Step 2.

Follow these steps first for your Data Lake and then for each attached Cloudera Data Hub separately. The examples and screenshots below assume that your cluster is using Cloudera Runtime 7.2.16. For older Cloudera Runtime, just replace 7.2.16 with your actual Cloudera Runtime version.

1. Upgrade Data Lake and Cloudera Data Hub to the latest service pack of their Runtime version.

   To upgrade your Data Lake and all Cloudera Data Hub clusters to the latest service pack of their Runtime version, perform the following for your Data Lake and each Cloudera Data Hub from the Upgrade UI each cluster:

   a. Select the upgrade target that matches your current Cloudera Runtime version. If you are upgrading a Data Lake, you will see 7.2.16 (Cloudera Runtime upgrade, OS; centos7); for Cloudera Data Hub clusters you will see Select 7.2.16 (Cloudera Runtime upgrade). Select this line and then run Validate and Prepare.
   b. Once Validate and Prepare completes, return to this page, select the same line again and run Upgrade:

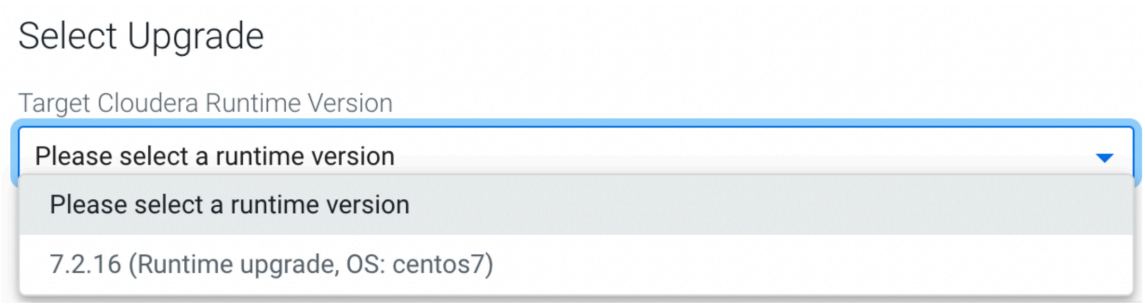

   For Data Lakes, this will perform the OS image upgrade, so step 2 below is required for Cloudera Data Hub clusters only.

2. Upgrade the OS image on all Cloudera Data Hub clusters.

   Additionally, you are required to upgrade the OS image on all Cloudera Data Hub clusters:

   a. Open the Cloudera Data Hub Upgrade UI again and select the upgrade target that matches your current Cloudera Runtime version and run Validate and Prepare. Select 7.2.16 (OS upgrade, OS; centos7).
   b. Once Validate and Prepare completes, return to this page, select the same lineagain and run Upgrade.

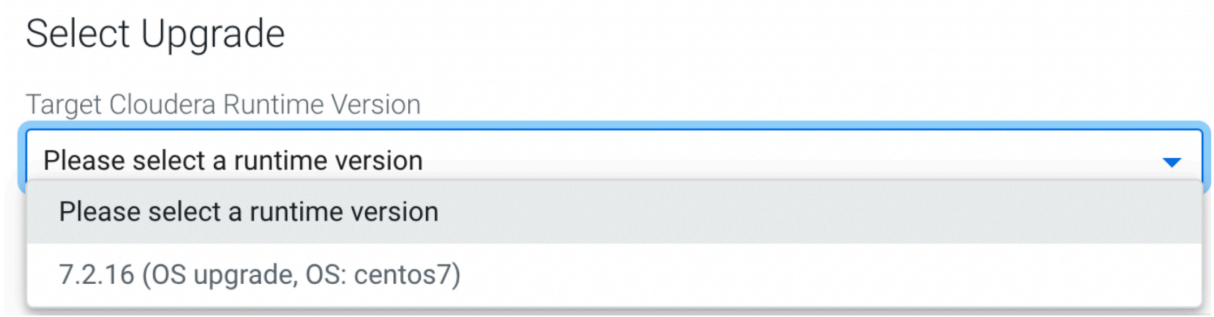

Tip:

When upgrading a Data Lake, we recommend allocating a 60 minutes maintenance window for this upgrade step. All Cloudera Data Hub clusters in your environment should be stopped and Data Services should not be running workloads for the duration of the operation.

When upgrading a Cloudera Data Hub, allocate 90-120 minutes (as you will be performing this upgrade in two steps).

USEFUL LINKS:

- Upgrading a Data Lake (UI)
- Upgrading a Data Lake (CLI)
- Performing a Cloudera Data Hub service pack upgrade (UI)
- Performing a Cloudera Data Hub OS upgrade (UI)

- Upgrading a Cloudera Data Hub with CLI

Step 2: Upgrade FreeIPA to RHEL 8

Before your Data Lake or Cloudera Data Hub clusters can be upgraded to RHEL 8, you first need to upgrade your environment (FreeIPA) cluster.

Important:

This step applies only to FreeIPA clusters using a CentOS 7 image.

Open the FreeIPA tab on your environment's details page and select Upgrade. From the drop-down menu select FreeIPA (Latest, OS: redhat8) and run the upgrade:

Note:

If in the drop-down selector described below you do not see the option FreeIPA (Latest, OS: centos7), then your cluster is already using RHEL 8 and you can skip step completely.

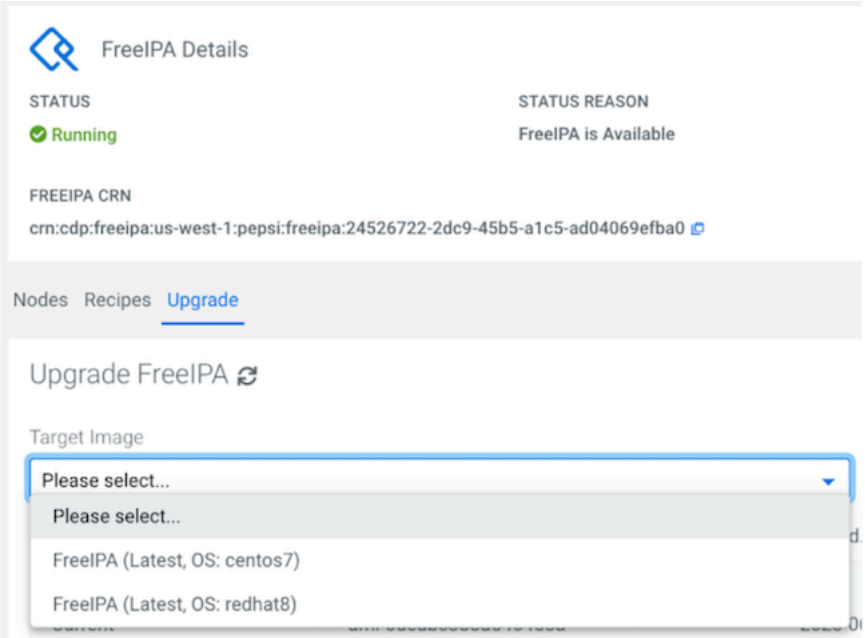

Tip:

We recommend allocating a 120-minute maintenance window for this upgrade step for a 3-node FreeIPA cluster (1-node and 2-node clusters will take less time).

USEFUL LINKS:

- Upgrading FreeIPA (UI and CLI)

Step 3: Upgrade Data Lake and Cloudera Data Hub clusters to the latest Cloudera Runtime 7.2.17 and the latest RHEL 8 OS image

Upgrade your clusters to the latest 7.2.17 and to RHEL 8.

Important:

This step applies only to Data Lake and Cloudera Data Hub clusters using a CentOS 7 image. If your clusters are already using Cloudera Runtime 7.2.17 and a RHEL 8 image, then you are not required to perform this step.

Perform the following steps for the Data Lake and then each Cloudera Data Hub:

1. Understand RHEL 8 requirements.

   Before performing the OS version change from CentOS 7 to RHEL 8 make sure you have understood the requirements mentioned in Upgrading from CentOS to RHEL.
2. Upgrade to the latest 7.2.17 service pack if needed.

   To determine whether you need to perform a Cloudera Runtime and OS upgrade, or only an OS upgrade, check whether you see the target version in the drop-down labeled 7.2.17 (OS upgrade, OS: redhat8). If you do, you only need an OS upgrade. and you can skip to step 3 below. If you do not see it, perform the following steps for a Cloudera Runtime upgrade:

   a. If you are upgrading a Data Lake, select 7.2.17 (Cloudera Runtime upgrade, OS: redhat8). For upgrading a Cloudera Data Hub, select 7.2.17 (Cloudera Runtime upgrade). Next, run Validate and Prepare.
   b. Once Validate and Prepare completes, return to this page, select the same line again, and run Upgrade.

      Data Lake UI

      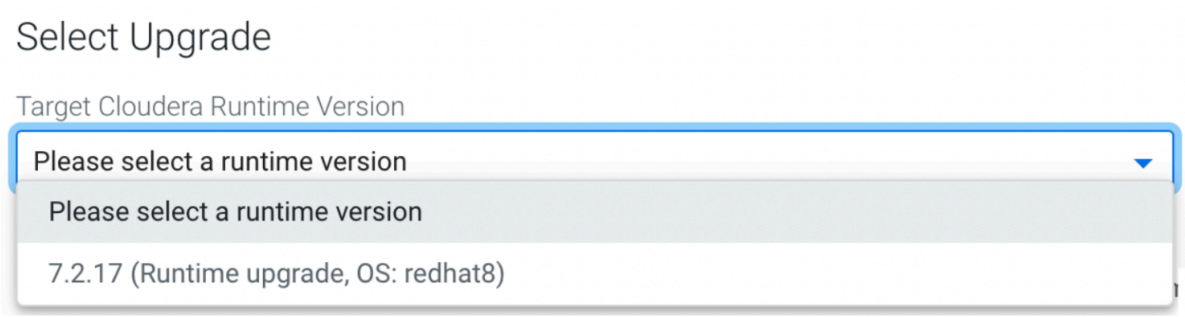

      Cloudera Data Hub UI

      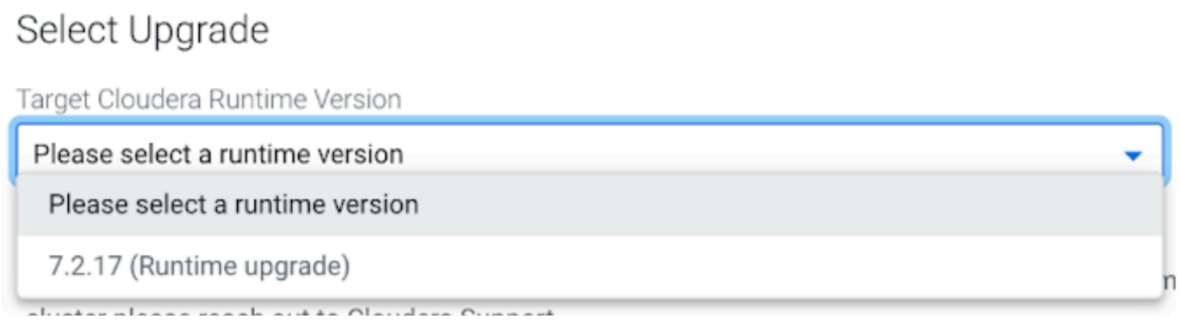

      For Data Lakes this will perform the RHEL 8 upgrade as well.

3. Upgrade all Cloudera Data Hub clusterss to RHEL.

   Open the Upgrade UI again and the 7.2.17 (OS upgrade, OS: redhat8) option will now be available in the drop-down menu.

   a. Select Select 7.2.17 (OS upgrade, OS: redhat8) and run Validate and Prepare.
   b. Once Validate and Prepare completes, return to this page, select Select 7.2.17 (OS upgrade, OS: redhat8) again and run Upgrade.

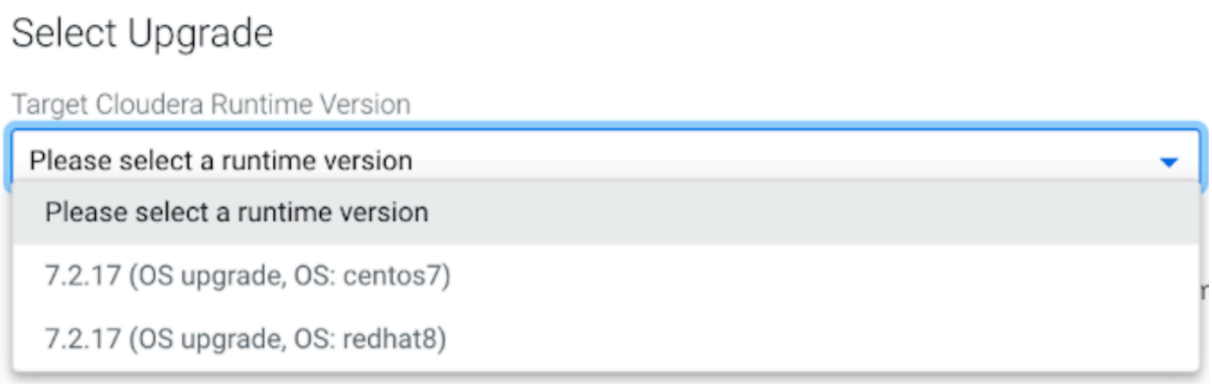

Tip: When upgrading a Data Lake, all Cloudera Data Hub clusters in your environment should be stopped and Cloudera data services should not be running workloads for the duration of the operation. We recommend allocating a 90-120 minute maintenance window for this upgrade step.

USEFUL LINKS:

- Upgrading a Data Lake (UI)
- Upgrading a Data Lake (CLI)
- Performing a Cloudera Data Hub OS upgrade
- Upgrading a Cloudera Data Hub with CLI

Step 4: Resize Medium Duty Data Lake to Enterprise Data Lake

If using a Medium Duty Data Lake, you should resize it to Enterprise Data Lake.

Important: This step applies only to Data Lake clusters that have been upgraded to 7.2.17 and have a Medium Duty shape. If your cluster is already using RHEL 8 and has the Enterprise Data Lake shape then you are not required to perform this step. You also do not need to perform this if you are using a Light Duty Data Lake.

Before you begin, note the following:

- Prior to attempting the resize, ensure that all Cloudera Data Hub clusters in your environment have been upgraded to Cloudera Runtime 7.2.17 and your data services have been upgraded to the latest version available.
- During the resize operation your cluster's Cloudera Manager and Cloudera Runtime configuration will be updated to the most recent recommendations (performance tuning, etc). You will be required to re-apply these custom configurations after you have performed the resize to Enterprise Data lake.
- There might be a requirement to vertically scale the Data Lake nodes to increase the available resources in case high resource utilization is observed after Data Lake resize to Enterprise and Operating System upgrade to RHEL from CentOS.
- The resize operation will perform an SDX backup and restore in the background. This involves writing and reading from cloud storage. If you have not yet used data lake backup, please make sure your environment's storage permissions are configured correctly for these operations.
- Ensure that all Data Hubs in your environment have been upgraded to Runtime 7.2.17 and your data services have been upgraded to the latest version available.
- The resizing operation requires a downtime and should be performed during a maintenance window. No metadata changes may occur during the resizing, as these changes will no longer be present once the resizing operation

completes (the previously backed up metadata is being restored). Suspend any operations that may result in any SDX metadata change during the resizing operation.
- Cloudera Data Hub clusters should be stopped before the resizing operation begins. For any cluster that cannot be stopped, stop all of the services on the Cloudera Data Hub through the Cloudera Manager UI.
- With Cloudera DataFlow 2.0 or lower, some flows must be re-created after a resizing operation.
- Review Data Lake resizing: Prerequisites.

Follow the steps for Data Lake resizing. During this operation, the metadata maintained in your current Data Lake is automatically backed up, a new Enterprise Data Lake is created within the environment, and the metadata is automatically restored to this new cluster. As mentioned above, any custom cluster configuration that you previously made will need to be reapplied after the resize completes.

The maintenance window required for this operation depends on the size of your SDX metadata. When you open the Resize cluster UI, it will show you the estimated duration of the operation.

USEFUL LINKS:

- Data Lake resizing prerequisites and limitations
- Resizing the Data Lake (UI)
- Resizing the Data Lake (CLI)

Step 5: Upgrade Data Lake and Cloudera Data Hub clusters from 7.2.17 to 7.2.18

Perform the following steps for your Data Lake and each Cloudera Data Hub:

If you are upgrading a Data Lake, all Cloudera Data Hub clusters in your environment should be stopped and Data Services should not be running workloads for the duration of the operation.

1. Once Cloudera Runtime 7.2.18 (Cloudera Runtime upgrade, OS: redhat8) is available to select from the Upgrade UI, select it and run Validate and Prepare.
2. Once Validate and Prepare completes, return to this page, select 7.2.18 (Cloudera Runtime upgrade, OS: redhat8) again and run Upgrade.

Attention:

Enterprise Data Lakes created with Cloudera Runtime 7.2.17.300 or newer and a RHEL 8 image can be directly upgraded in a rolling manner to 7.2.18. This allows the upgrade operation to be performed without stopping attached Cloudera Data Hub clusters and workloads on any data services that are running in the environment. To determine if your cluster is eligible, refer to Rolling upgrades.

Cloudera recommends that you perform the rolling upgrade during a maintenance window, when no critical workloads are running on your attached Cloudera Data Hub clusters or data services.

To perform a rolling upgrade of the Data Lake, select “Rolling Upgrade”:

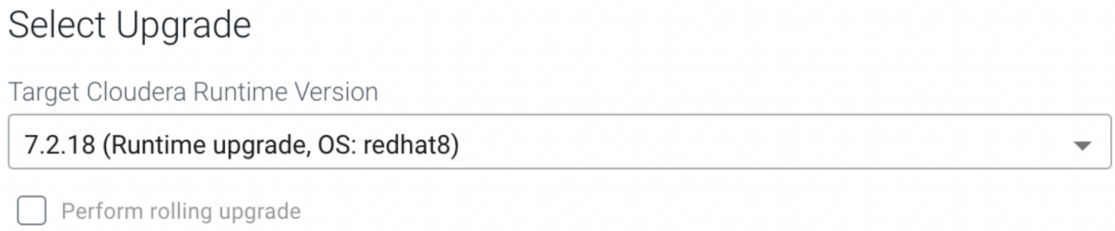

Tip:

We recommend allocating a 60 minutes maintenance window for this upgrade step

USEFUL LINKS:

- Upgrading a Data Lake (UI)
- Upgrading a Data Lake (CLI)
- Performing a Cloudera Data Hub major/minor version upgrade (UI)
- Upgrading a Cloudera Data Hub with CLI

- Rolling upgrades

Upgrading to Cloudera Runtime 7.3.1

Cloudera Runtime 7.3.1 introduces significant new features and improvements. As a result, if you are planning to upgrade the Cloudera Runtime version in your existing Data Lake or Cloudera Data Hub clusters to 7.3.1, you might be required to perform this in multiple steps.

Some of the important Cloudera Runtime 7.3.1 features include:

- Unified Runtime for Cloudera Public Cloud and Cloudera Private Cloud.
- RHEL 8.10 is set as the default operating system.
- Python 3.9 is set as the new default version.
- OpenJDK 17 becomes the new default Java runtime version, new clusters will be launched with this version.

  OpenJDK 11 is no longer supported, and OpenJDK 11 clusters cannot be upgraded to Cloudera Runtime 7.3.1.

  After upgrading clusters from Cloudera Runtime 7.2.x to 7.3.1, the JDK version remains the same, that is, JDK8.
- Spark 2 applications need to be rebased to Spark 3 as Spark 2 is no longer supported and clusters with Spark 2 will not be allowed to upgrade to Cloudera Runtime 7.3.1.
- Zeppelin service is turned off for Cloudera Runtime 7.3.1 clusters and gets removed automatically during the upgrade.
- Cloudera Data Engineering 1.23 and earlier versions are not supported on Cloudera Runtime 7.3.1

Direct upgrade to Cloudera Runtime 7.3.1 is supported from the following service packs:

- 7.2.17.200
- 7.2.17.300
- 7.2.17.400
- 7.2.17.500
- 7.2.18.0
- 7.2.18.100
- 7.2.18.200

If your Cloudera Runtime version is not in the list of supported versions for a direct upgrade, it is possible that you can still upgrade to one of the above versions first, and then upgrade to Cloudera Runtime 7.3.1. For more information on these upgrade paths, see Identify your upgrade path.

Currently, there is no supported upgrade path to Cloudera Runtime 7.3.1 in the following cases:

- from Cloudera Runtime 7.2.18.300 or later version
- if the cluster uses JDK 11

Identify cluster version details

Identify your Data Lake and Cloudera Data Hub cluster's Cloudera Runtime version, Data Lake, Cloudera Data Hub, and FreeIPA image version and operating system (OS) type, and Data Lake shape.

Your upgrade path to versions 7.2.18 or 7.3.1 may depend on the current Cloudera Runtime version, image version, and OS of your cluster. Cloudera recommends that you identify your version information as follows:

1. Identify cluster's Cloudera Runtime version on page 13
2. Identify cluster's image version, image creation date, OS type on page 14
3. Identify Data Lake shape on page 16

Identify cluster's Cloudera Runtime version

Follow these instructions to identify your Data Lake and Cloudera Data Hub's Cloudera Runtime version. You should follow these steps separately for each cluster.

Steps

For CDP UI

1. In the Cloudera Management Console, navigate to the details page of the Data Lake or Cloudera Data Hub cluster that you would like to upgrade.
2. In the Cloudera Manager Info section find the field called "Runtime version":

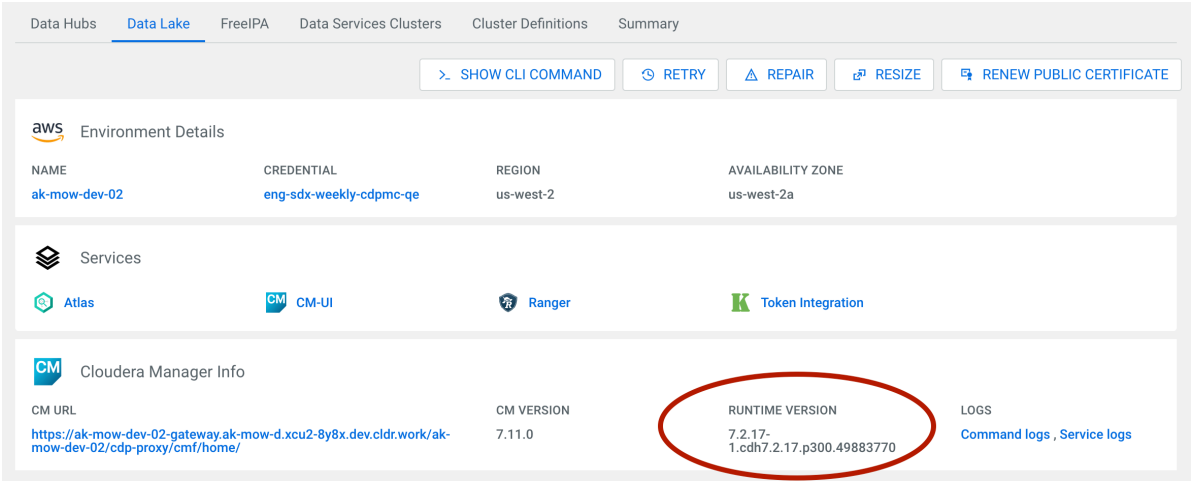

For CDP CLI

Get the Cloudera Runtime version of your Data Lake from the CDP CLI by using the following command:

```
cdp datalake describe-datalake --datalake-name <DL-NAME>
```

Get the Cloudera Runtime version of your Cloudera Data Hub from the CDP CLI by using the following command:

```
cdp datahub describe-cluster --cluster-name <DH-NAME>
```

The version information is displayed in the following format:

7.2.17-1.cdh7.2.17.p200.46967063

The highlighted digits show the current Cloudera Runtime version and the number after p indicates the current Service Pack version. In the above example, the cluster is using Cloudera Runtime version 7.2.17.200.

Identify cluster's image version, image creation date, OS type

Follow these instructions to identify Data Lake and Cloudera Data Hub image version and OS type. You should follow these steps separately for each cluster.

Steps

For CDP UI

1. In the Cloudera Management Console, navigate to the details page of the Data Lake or Cloudera Data Hub cluster that you would like to upgrade.
2. Scroll down to access the left navigation pane.

3. From the left navigation pane, select Image Details:

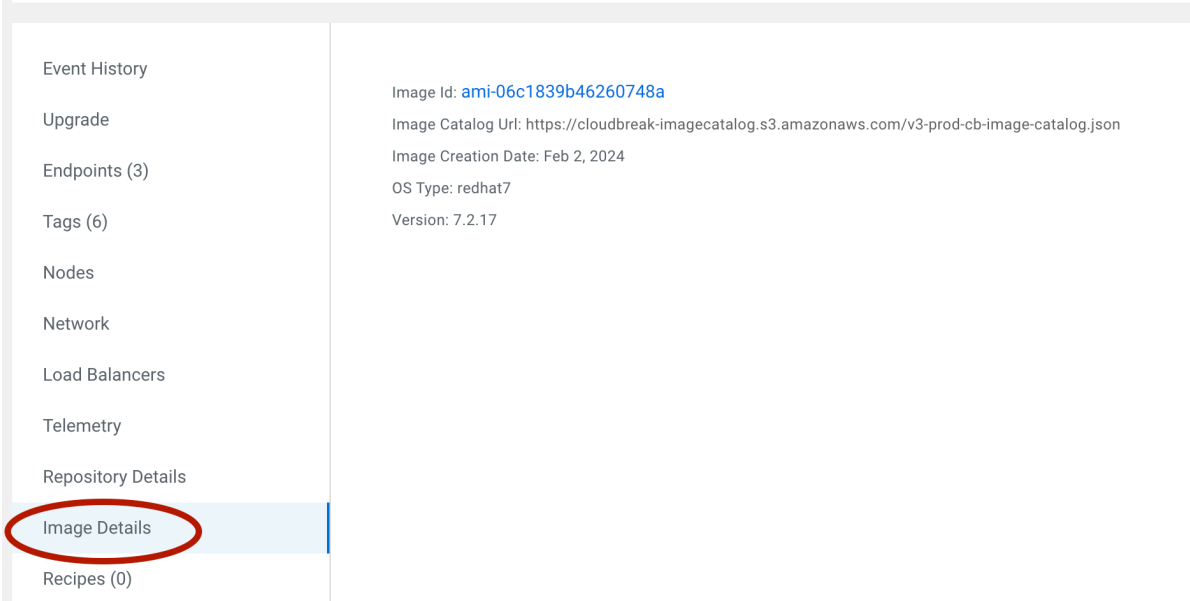

4. You will see information similar to the following:

```
Image Id: ami-0e95ecddb7fc75ea8
Image Catalog Url: https://cloudbreak-imagecatalog.s3.amazonaws.com/v3-
prod-cb-image-catalog.json
Image Creation Date: Jan 9, 2024
OS Type:redhat7
Version: 7.2.17
```

Under OS Type:

- If you see “redhat7”, your cluster is using a CentOS 7 OS image. This is a known issue.
- If you see “redhat8”, your cluster is using a RHEL 8 OS image.

Clicking on image ID will redirect you to the image catalog and provide you with further information, including the actual OS Type:

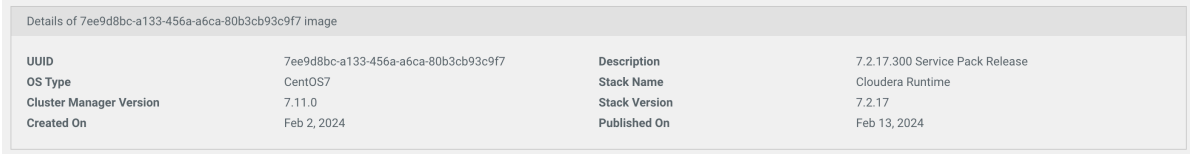

For CDP CLI

Get Data Lake’s image version from the Cloudera CLI by using the following command:

```
cdp datalake upgrade-datalake --show-latest-available-image-per-runtime --
datalake-name <DL-NAME>
```

Get Cloudera Data Hub cluster’s image version from the CDP CLI by using the following command:

```
cdp datahub upgrade-cluster --show-latest-available-image-per-runtime --
cluster-name <DH-NAME>
```

The output of this command looks like this, providing you with the OS information:

```
{
 "targetImage": {
```

```
 "catalog": "https://cloudbreak-imagecatalog.s3.amazonaws.com/v3-prod-fr
eeipa-image-catalog.json",
 "id": "a5b2580e-7487-4ee6-a80f-d991fa00cd13",
 "os": "redhat8",
 "imageName": "ami-0935d849de6a25e45",
 "date": "2024-01-09"
 },
 "originalImage": {
 "catalog": "https://cloudbreak-imagecatalog.s3.amazonaws.com/v3-prod-f
reeipa-image-catalog.json",
 "id": "e1f92688-3773-47dd-89ff-561933540108",
 "os": "centos7",
 "imageName": "ami-0db1b7000c8a27fd4",
 "date": "2023-11-22"
 },
 "operationId": "3a50f30f-33f3-4f81-b9bb-30c119023ff0"
}
```


Follow these instructions to identify FreeIPA OS type.

Steps

1. In the Cloudera Management Console, navigate to the environment details page,.
2. Click on the FreeIPA tab.
3. In the Nodes tab (which is open by default), expand one section corresponding to an instance ID.
4. Under Image Details, you can find the OS Type:

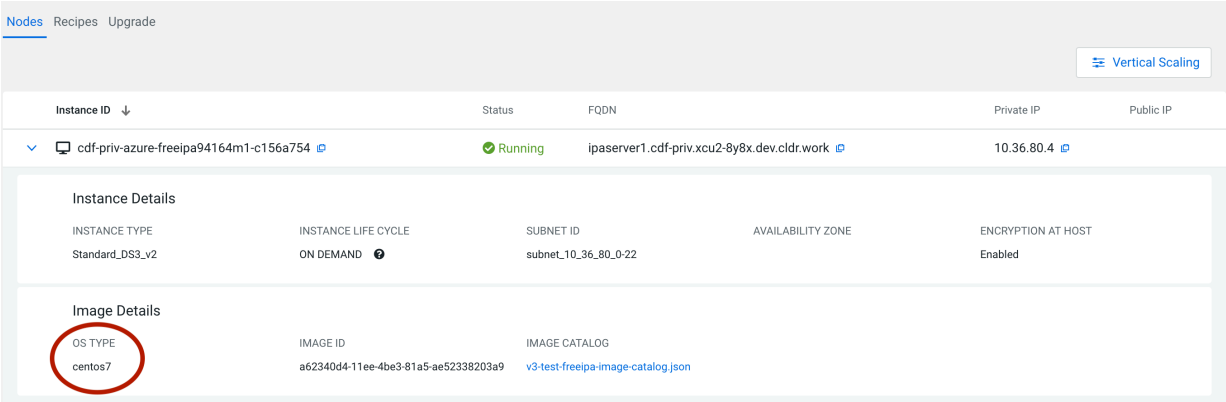

Identify Data Lake shape

Follow these instructions to identify Data Lake's shape.

For CDP UI

1. In the Cloudera Management Console, navigate to the details page of the Data Lake that you would like to upgrade:

2. Find the Scale field. There are three possible values: “Light Duty”, “Medium Duty HA” and “Enterprise Data Lake”.

For CDP CLI

Get the Data Lake shape from the CDP CLI by using the following command:

```
cdp datalake describe-datalake –datalake-name <DL-NAME>
```

Find the “scale” field in the output of this command. There are three possible values: “Light Duty”, “Medium Duty HA” and “Enterprise Data Lake”.

Identify your upgrade path

Refer to the following diagram for a high-level overview of the upgrade steps. Your starting point will vary, depending on the Cloudera Runtime version and image version that your environment, Data Lake, and Cloudera Data Hub clusters are using.

The high-level upgrade steps are presented horizontally with the different step-in points identified in medium gray boxes:

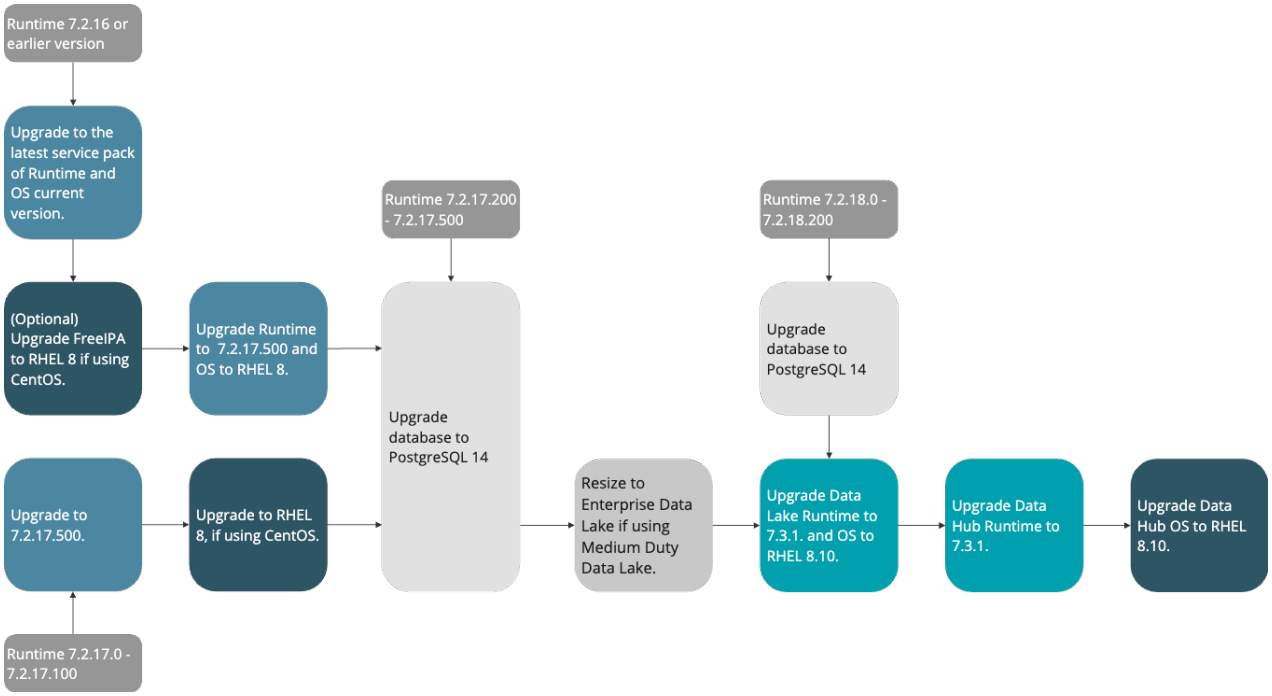

As presented in the diagram, you may be required to perform the following operations:

- Runtime service pack upgrades for your Data Lake and each Cloudera Data Hub cluster
- Major/Minor Runtime and OS upgrades for your Data Lake and each Cloudera Data Hub cluster
- CentOS to RHEL 8 upgrade for your FreeIPA, Data Lake, and each Cloudera Data Hub cluster
- Database PostreSQL version upgrade
- Data Lake resize

For example, if you are starting with Cloudera Runtime 7.2.16 and your Data Lake is Medium Duty, you need to have your Data Lake and all Cloudera Data Hub clusters upgraded to the latest service pack of Cloudera Runtime 7.2.16 first, then you need to upgrade FreeIPA to RHEL 8, upgrade Data Lake and Cloudera Data Hub clusters to 7.2.17.500, upgrade the database to PostgreSQL 14, resize your Data Lake to Enterprise Data Lake, then upgrade

Data Lake to Cloudera Runtime 7.3.1 and RHEL 8.10, then Upgrade Cloudera Data Hub clusters to Runtime 7.3.1, and finally, upgrade Cloudera Data Hub clusters to RHEL 8.10.

Important:

- Each of these operation has its own prerequisites and postrequisites so in addition to the actual upgrade steps, the prerequisites and postrequisites need to be performed as well.
- You will need to upgrade all Data Lakes and Cloudera Data Hub clusters separately, first the Data Lakes and then each Cloudera Data Hub cluster.
- You can perform each upgrade operation either from the Upgrade UI of the respective cluster or using the Cloudera CLI.
- The upgrade user interface in the Cloudera Management Console shows you the next Data Lake and Cloudera Data Hub upgrade step available, so you can refer to it as a guideline.

Review the prerequisites

Before you start upgrading your clusters, ensure that you meet the prerequisites for each upgrade that is required.

Refer to the following documentation for detailed prerequisites:

- Review Data Lake Upgrade: Before you begin and Cloudera Data Hub Upgrade: Limitations and Prerequisites.
- If you are upgrading a Data Lake, use the Backup and Restore functionality. This will allow you to restore the SDX metadata if required.
- Certain Data Hubs might require certain pre-upgrade and post-upgrade steps. Review the related documentation before upgrading Data Hubs.
- If you are upgrading from CentOS to RHEL, review the Prerequisites for upgrading from CentOS to RHEL.
- If your upgrade path involves a Data Lake resize, refer to Review Data Lake resizing: Prerequisites.

High-level upgrade steps

Refer to this section for high-level upgrade steps and links to more detailed upgrade documentation. This document is not meant to provide you with complete upgrade steps, but rather it outlines the general upgrade path and provides you with the links to existing upgrade documentation.

Important:

- Each of these operation has its own prerequisites and postrequisites so in addition to the actual upgrade steps, the prerequisites and postrequisites need to be performed as well.
- You will need to upgrade all Data Lakes and Cloudera Data Hub clusters separately, first the Data Lakes and then each Cloudera Data Hub cluster.
- For Data Lakes, the Runtime and OS upgrade is performed in one step. For Cloudera Data Hub clusters, the Cloudera Runtime needs to be updated first, and then the OS in a separate step.
- You can perform each upgrade operation either from the Upgrade UI of the respective cluster or using the Cloudera CLI.
- The upgrade user interface in the Cloudera Management Console shows you the next Data Lake and Cloudera Data Hub upgrade step available, so you can refer to it as a guideline.

Depending on the upgrade paths you have identified in Resizing to Enterprise Data Lake if using Medium Duty Data Lake, you need to start the upgrade procedure at different points as follows:

- If the Cloudera Runtime version of your Data Lake and Cloudera Data Hub clusters is 7.2.17.100 or earlier, you need to start at Upgrading to a supported version.
- If the Cloudera Runtime version of your Data Lake and Cloudera Data Hub clusters is 7.2.17.200-7.2.17.500, you can proceed to Resizing to Enterprise Data Lake if using Medium Duty Data Lake.
- If the Cloudera Runtime version of your Data Lake and Cloudera Data Hub clusters is 7.2.18.100-7.2.18.200, you can proceed to Upgrading Data Lake Cloudera Runtime to 7.3.1. and OS to RHEL 8.10.

Important:

Currently, there is no supported upgrade path to Cloudera Runtime 7.3.1 in the following cases:

- from Cloudera Runtime 7.2.18.300 or later version
- from Cloudera Runtime 7.2.17.600 to 7.2.17.900
- if the cluster uses JDK 11

Upgrade Data Lakes and all Cloudera Data Hub clusters to a version from which it is supported to upgrade to Cloudera Runtime 7.3.1

Before being able to upgrade to Cloudera Runtime 7.3.1, you need to upgrade Data Lakes and all Cloudera Data Hub clusters to a Cloudera Runtime 7.2.17 or 7.2.18 version from which it is supported to upgrade to Cloudera Runtime 7.3.1.

Important: This step is only required if your current Cloudera Runtime version is 7.2.17.100 or earlier.

As part of this upgrade step, you need to upgrade Data Lakes and Cloudera Data Hub clusters to one of the following versions from which upgrade to Cloudera Runtime 7.3.1 is supported:

- 7.2.17.200-7.2.17.500

Upgrading Cloudera Runtime 7.2.16 or earlier version to Cloudera Runtime 7.2.17.500

If the Cloudera Runtime version of your Data Lakes and Cloudera Data Hub clusters is 7.2.16 or earlier, perform steps 1 through 4 in Identify your upgrade path of the Upgrading to Cloudera Runtime 7.2.18 documentation. However, do not upgrade to the latest 7.2.17 service pack, only upgrade to Cloudera Runtime 7.2.17.500. It is not required to upgrade to Cloudera Runtime 7.2.18, there is a direct upgrade path from Cloudera Runtime 7.2.17.200-7.2.17.500 to Cloudera Runtime 7.3.1.

To be able to upgrade to Cloudera Runtime 7.3.1, you must perform the following steps:

1. Upgrade Data Lake and Cloudera Data Hub clusters to the latest service pack of Cloudera Runtime and OS current version.
2. Upgrade FreeIPA to RHEL 8 if using CentOS.
3. Upgrade Data Lake and Cloudera Data Hub clusters Runtime to 7.2.17.500.

   Note: This is only possible through the CDP CLI as you cannot select this specific service pack during an upgrade through the UI.

4. Upgrade Data Lake and Cloudera Data Hub clusters OS to RHEL 8.

Upgrading Cloudera Runtime 7.2.17.0 or 7.2.17.100 to Cloudera Runtime 7.2.17.500

If the Cloudera Runtime version of your Data Lakes and Cloudera Data Hub clusters is 7.2.17.0 or 7.2.17.100, a direct upgrade to Cloudera Runtime 7.3.1 is not supported. To upgrade to a Cloudera Runtime 7.2.17 service pack from which the direct upgrade to Cloudera Runtime 7.3.1 is supported, perform the following steps:

1. Upgrade the Cloudera Runtime of your Data Lakes and Cloudera Data Hub clusters to the Cloudera Runtime service packs 7.2.17.200 through 7.2.17.500.

   For more information, see Upgrading a Data Lake manually via CLI and Performing a Cloudera Data Hub service pack upgrade.
2. If you are using CentOS, upgrade to RHEL 8.

   For more information, see Upgrading from CentOS to RHEL.

Upgrade database to PostgreSQL 14

Before being able to upgrade to Cloudera Runtime 7.3.1, you need to upgrade Data Lakes and all Cloudera Data Hub databases to a to PostgreSQL 14.

For information on upgrading Data Lake and Cloudera Data Hub databases to PostgreSQL 14, see Upgrading Data Lake/Cloudera Data Hub database.

Note:

If you are using an Azure environment, upgrading the PostgreSQL database used by the Data Lake and Cloudera Data Hub clusters also upgrades Azure Single Server to Azure Flexible Server as Single Server is not supported with PostgreSQL 14.

For more information, see Upgrading Azure Single Server to Flexible Server. Make sure to review both the database upgrade and the Azure Single to Flexible Server upgrade documentation.

Resize to Enterprise Data Lake if using Medium Duty Data Lake

If you are upgrading from Cloudera Runtime 7.2.17.200-7.2.17.500 to Cloudera Runtime 7.3.1, you must resize your Data Lake from Medium Duty to Enterprise Data Lake as Medium Duty Data Lake is not supported starting with Cloudera Runtime 7.2.18.

Important: This step applies only to Data Lake clusters that have been upgraded to 7.2.17 and have a Medium Duty shape. If your cluster is already using RHEL 8 and has the Enterprise Data Lake shape then you are not required to perform this step. You also do not need to perform this if you are using a Light Duty Data Lake.

For information on why the Data Lake resize is needed, see Upgrading from Medium Duty to Enterprise Data Lake.

For information on how to perform the resize, see Data Lake resizing.

Prerequisites

Prior to resizing the Data Lake, ensure that the following are in place:

- Review the notes, limitations, and prerequisites listed in Data Lake resizing.

During this operation, the metadata maintained in your current Data Lake is automatically backed up, a new Enterprise Data Lake is created within the environment, and the metadata is automatically restored to this new cluster. As mentioned above, any custom cluster configuration that you previously made will need to be reapplied after the resize completes.

The maintenance window required for this operation depends on the size of your SDX metadata. When you open the Resize cluster UI, it will show you the estimated duration of the operation.

Upgrade Data Lake Cloudera Runtime version to 7.3.1. and OS to RHEL 8.10

From the supported Cloudera Runtime 7.2.17 or 7.2.18 versions, you need to upgrade Data Lake Cloudera Runtime version to 7.3.1. and OS to RHEL 8.10 in one step prior to upgrading Cloudera Data Hub clusters.

All Cloudera Data Hub clusters in your environment should be stopped and Data Services should not be running workloads for the duration of the operation.

- 1. Once Runtime 7.3.1 (Runtime upgrade, OS: redhat8) is available to select from the Upgrade UI, select it and run Validate and Prepare.

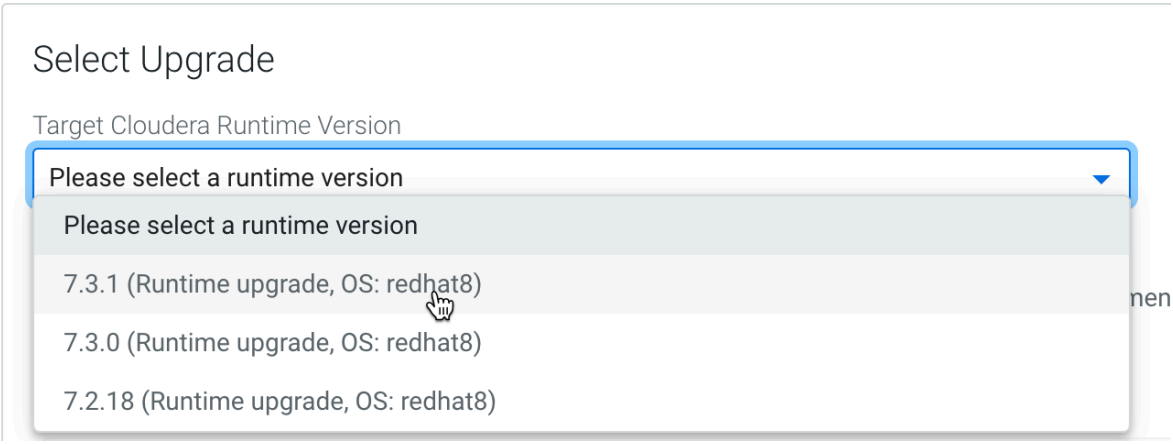

2. Once Validate and Prepare completes, return to this page, select 7.3.1 (Runtime upgrade) again and run Upgrade.

Tip: When upgrading a Data Lake, Cloudera recommends allocating a 60-minute maintenance window for this upgrade step.

For more information, see Upgrading a Data Lake.

Upgrade Cloudera Runtime version of Cloudera Data Hub clusters to 7.3.1

Once the Data Lake Cloudera Runtime version has been upgraded to 7.3.1 and simultaneously, the OS has been upgraded to RHEL 8.10, the Cloudera Runtime version of each Cloudera Data Hub cluster needs to be upgraded to 7.3.1.

Note:

If you have a Cloudera Data Engineering Data Hub cluster, during the Cloudera Runtime upgrade process to version 7.3.1, you need to remove Spark 2 while creating a new cluster with Spark 3 and migrate your data. Depending on your environment and Spark applications, besides the cluster upgrade tasks, you need to perform Spark application migration tasks and sidecar migration tasks for the Cloudera Data Hub cluster in multiple steps. For more information, see Upgrading Apache Spark.

- 1. Once Runtime 7.3.1 (Runtime upgrade) is available to select from the Upgrade UI, select it and run Validate and Prepare.

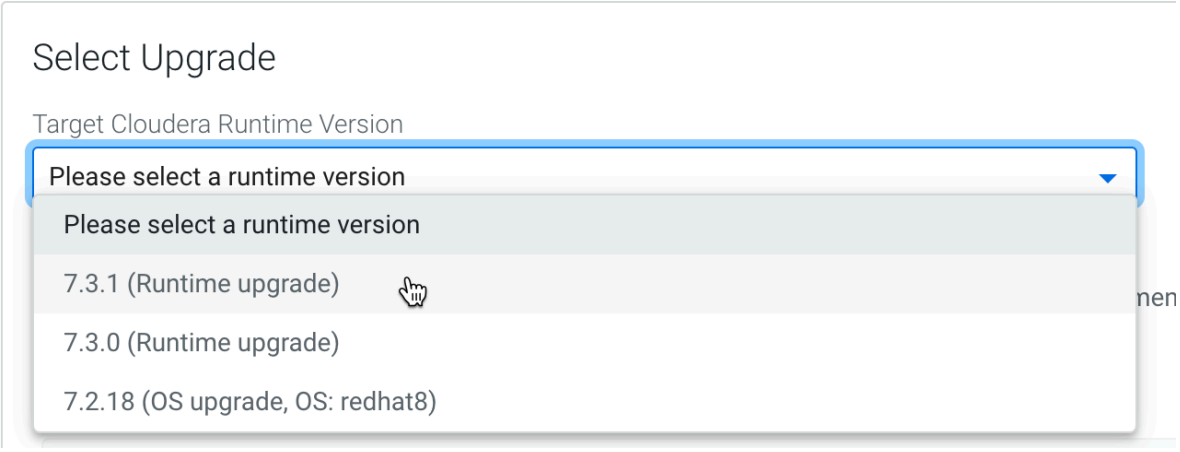

2. Once Validate and Prepare completes, return to this page, select 7.3.1 (Runtime upgrade) again and run Upgrade.

Tip: When upgrading a Cloudera Data Hub cluster,allocate 90-120 minutes (as you will be performing this upgrade in two steps).

For more information, see Upgrading Cloudera Data Hub clusters.

Upgrade OS version of Cloudera Data Hub clusters to RHEL 8.10

During a Data Lake upgrade, the Data Lake OS is automatically upgraded to RHEL 8.10. In case of Cloudera Data Hub clusters, you need to upgrade the OS to RHEL 8.10 manually.

After performing Cloudera Runtime version upgrade of the Cloudera Data Hub cluster, open the Upgrade UI again and the 7.3.1 (OS upgrade, OS: redhat8)option will now be available in the drop-down menu.

1. Select 7.3.1 (OS upgrade, OS: redhat8) and run Validate and Prepare.
2. Once Validate and Prepare completes, return to this page, select 7.3.1 (OS upgrade, OS: redhat8) again and run Upgrade.

Upgrading from CentOS to RHEL

As part of FreeIPA, Data Lake, and Cloudera Data Hub upgrade, you have the option to upgrade the operating system (OS) on the virtual machines (VMs) from CentOS 7 to Red Hat Enterprise Linux 8 (RHEL 8).

As of June 30, 2024, CentOS reaches End of Life (EOL), and consequently, Cloudera Public Cloud 7.2.18 only supports RHEL 8-based images. New deployments of Data Lakes and Cloudera Data Hub clusters with Cloudera Runtime 7.2.18 and upgrades to 7.2.18 are only possible with RHEL 8. Data Lake and Cloudera Data Hub clusters running Cloudera Runtime 7.2.17 support both CentOS 7 and RHEL 8. Earlier Cloudera Runtime versions support CentOS 7 only. Cloudera will not publish any updates or fixes for CentOS-based images after June 2024.

You do not need to purchase a RHEL license for the upgrade. The license is covered by Cloudera.

Cloudera recommends that you perform the CentOS to RHEL in the following sequence:

1. Upgrade FreeIPA via the FreeIPA upgrade web UI or CLI.
2. Upgrade Data Lake via the Data Lake upgrade web UI or CLI.
3. Upgrade Cloudera Data Hub clusterss via the Cloudera Data Hub OS upgrade web UI or CLI.

Attention: Your Data Lake or Cloudera Data Hub must be using Cloudera Runtime 7.2.17 in order to access the CentOS to RHEL upgrade option.

Prerequisites for upgrading from CentOS to RHEL

Prior to attempting the CentOS to RHEL upgrade, ensure that you have taken care of the following prerequisites:

- Rewrite and test recipes if needed
- Check compatibility of third party software you might have installed on the nodes
- Opt in to use Azure Marketplace images (Azure only)

Rewrite and test recipes

If you are using FreeIPA, Data Lake, or Cloudera Data Hub recipes, you must ensure that they are compatible with RHEL. As some CentOS-specific commands may not work on RHEL, you may need to rewrite them. The steps are:

1. Rewrite your existing CentOS recipes so that they are compatible with RHEL.
2. Test the updated recipes on a newly created FreeIPA, Data Lake, or Cloudera Data Hub VM that uses RHEL operating system.
3. Detach your existing CentOS recipes and attach your updated RHEL recipes. You can find the steps here:
   a. Update FreeIPA recipes
   b. Update Data Lake recipes
   c. Update Cloudera Data Hub recipes

If after rewriting and testing your recipes are not working, contact Cloudera support for help.

Check compatibility of third party software

If you normally install additional software on your Data Lake, FreeIPA, or Cloudera Data Hub VMs (using recipes or manually), you may need to contact Cloudera support to make additional OS packages available. You should file a support ticket and provide details of the additional packages that you are installing (software name, version, and software URL).

Opt in to use Azure Marketplace images

Important:

To be able to upgrade Azure clusters to RHEL 8, you must first accept Azure Marketplace terms and conditions via Azure CLI or Cloudera web UI. See Cloudera images hosted in Azure Marketplace.

RHEL 8 images are only available through Azure Marketplace. In order for Cloudera to be able to load Cloudera-published virtual machine images in your subscription from the Azure Marketplace, you must first accept Azure Marketplace terms and conditions either via Cloudera web UI or Azure CLI . If you do not accept the terms and conditions, Cloudera will not be able to access the images hosted in Azure Marketplace and consequently environment creation will fail. For more information, see Cloudera images hosted in Azure Marketplace.

Upgrade clusters to RHEL

To upgrade your environment, Data Lake, and Cloudera Data Hub clusters to RHEL, follow the usual operating system (OS) upgrade documentation for FreeIPA, Data Lake, and Cloudera Data Hub.

Upgrade FreeIPA cluster to RHEL

When upgrading via the web interface, you navigate to the FreeIPA tab on your environment's details page and select Upgrade. Next, from the drop-down menu you select FreeIPA (Latest, OS: redhat8) and run the upgrade:

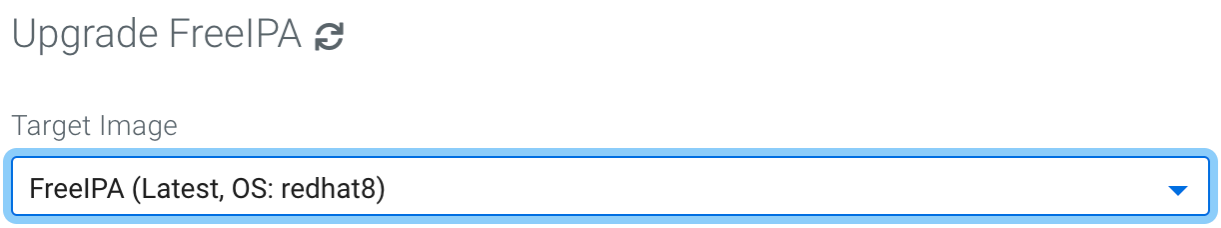

When upgrading via CDP CLI, you use the typical upgrade command and specify a RHEL image available for your cloud platform.

For detailed Cloudera UI and CLI instructions, see Upgrade FreeIPA.

Upgrade Data Lake to RHEL

When upgrading via the web interface, you navigate to your Data Lake's Upgrade page and perform the following steps:

1. Select 7.2.17 (OS upgrade, OS: redhat8) from the dropdown and run Validate and Prepare:

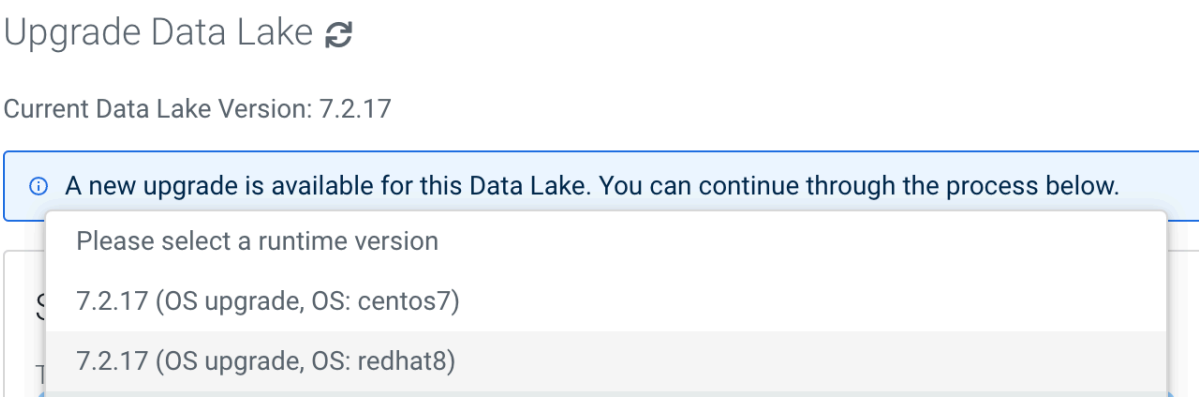

2. Once Validate and Prepare completes, you return to this page, select 7.2.17 (OS upgrade, OS: redhat8) again, and run Upgrade.

When upgrading via CDP CLI, you use the typical upgrade command and specify a RHEL image available for your cloud platform.

For full documentation including prerequisites and UI/CLI instructions, refer to Data Lake upgrade.

Upgrade Cloudera Data Hubs to RHEL

When upgrading via the web interface, you navigate to your Cloudera Data Hub's Upgrade page and perform the following steps:

1. Select 7.2.17 (OS upgrade, OS: redhat8) from the dropdown and run Validate and Prepare.

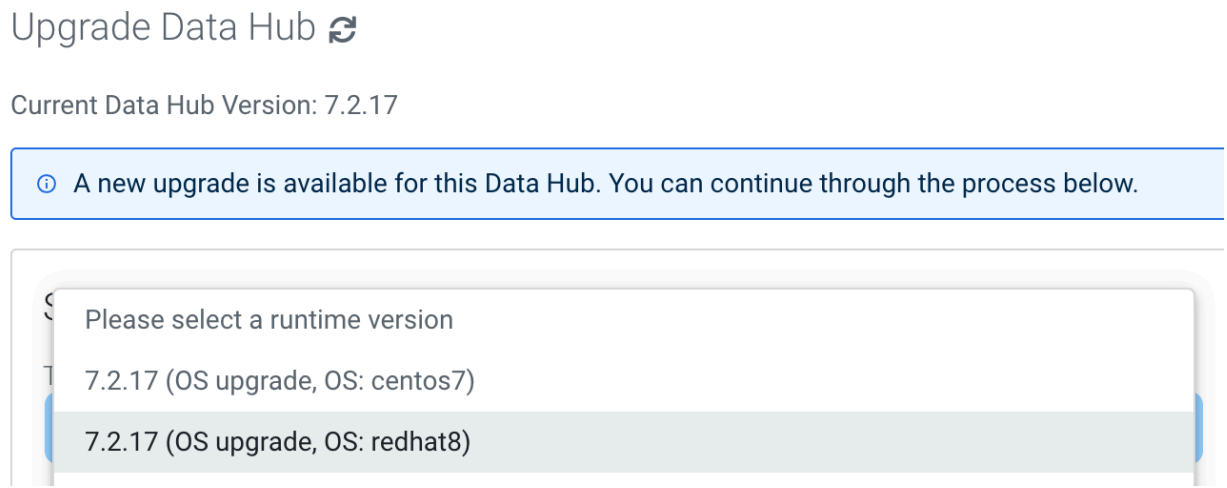

2. Once Validate and Prepare completes, return to this page, select 7.2.17 (OS upgrade, OS: redhat8) again, and run Upgrade.

When upgrading via CDP CLI, you use the typical upgrade command and specify a RHEL image available for your cloud platform.

For detailed instructions, see Performing a Cloudera Data Hub OS upgrade. For CLI instructions, see Upgrading a Cloudera Data Hub with the CDP CLI.

Upgrading from Medium Duty to Enterprise Data Lake

Refer to this content for an overview of the Data Lake resizing from Medium Duty to Enterprise Data Lake. The resizing of Medium Duty Data Lakes to Enterprise Data Lake should be performed on Cloudera Runtime 7.2.17 and is a prerequisite for upgrading to Cloudera Runtime 7.2.18 or 7.3.1.

Starting with Cloudera Runtime 7.2.18, Medium Duty Data Lake is discontinued and is replaced by Enterprise Data Lake. While earlier Cloudera Runtime versions support Medium Duty Data Lake (7.2.17 supports Enterprise, Medium Duty, Light Duty, and older versions support Medium Duty and Light Duty), 7.2.18 and later versions only support Enterprise and Light Duty Data Lakes.

Enterprise Data Lake enables Horizontal Scaling and Zero-Downtime Upgrades without increasing the infrastructure cost. For more information about Enterprise Data Lake, see Data Lake scale.

In order for existing Data Lakes to be upgraded to Cloudera Runtime 7.2.18 or later version, they must be using Enterprise or Light Duty Data Lake. If you are using Medium Duty Data Lake and would like to upgrade to Cloudera Runtime 7.2.18 or later version, you will need to upgrade to 7.2.17 first and then resize your Data Lake to Enterprise Data Lake using the existing Data Lake resize functionality. Afterwards, you will be able to proceed with the upgrade to Cloudera Runtime 7.2.18 or later version.

Note: After upgrading to EDL, Data Lakes with high resource utilization might require vertical scaling, which can increase cost. For more information, see Data Lake resizing

Upgrading from Spark 2 to Spark 3

If you have a Cloudera Data Engineering Data Hub cluster, during the Cloudera Runtime upgrade process to version 7.3.1, you need to remove Spark 2 while creating a new cluster with Spark 3 and migrate your data. Depending on your environment and Spark applications, besides the cluster upgrade tasks, you need to perform Spark

application migration tasks and sidecar migration tasks for the Cloudera Data Hub cluster in multiple steps. For more information, see Upgrading Apache Spark.

Rolling upgrades

With the release of Cloudera Runtime 7.2.18, rolling upgrades for certain Data Lakes and Cloudera Data Hub clusters are now available.

Rolling upgrades allow you to upgrade the Data Lake Cloudera Runtime and OS, or the Cloudera Data Hub Runtime and OS, without stopping the cluster and its services. This means that you can upgrade a Data Lake without stopping the attached Cloudera Data Hub clusterss and Data Services, or upgrade a Cloudera Data Hub without workload downtime.

Rolling upgrades for the Data Lake are limited to certain Data Lake Cloudera Runtime versions and shapes. Rolling upgrades for Cloudera Data Hub clusters are limited to certain Cloudera Runtime versions and cluster templates.

Related Information

Data Lake rolling upgrade | Cloudera Management Console

Data Hub rolling upgrade | Cloudera Data Hub

Cloudera Public Cloud

CDP CLI

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

CDP CLI 4

CLI client setup 4

Installing Cloudera client 4
- Install Cloudera client on Linux 4
- Install Cloudera client on macOS 5
  - Install Python on macOS 5
  - Install Cloudera client after Python on macOS 5
- Install Cloudera client on Windows 6
  - Install Python and pip on Windows 6
  - Install Cloudera client after Python on Windows 6

Logging into the CDP CLI/SDK 7

Generating an API access key 8

Configuring Cloudera client 9

Configuring CLI autocomplete 10

CLI reference 10

CDP CLI parameter types 10

Accessing CLI help 11

Installing Beta CDP CLI 11

CDP CLI

CDP CLI allows you to manage Cloudera environments, clusters, and other entities.

Related Information

CLI client setup
Generating an API access key
Configuring Cloudera client
Logging into the CDP CLI/SDK
Configuring CLI autocomplete
CLI reference
Accessing CLI help
Installing Beta CDP CLI

CLI client setup

To set up the Cloudera client, complete the following tasks:

1. Installing Cloudera client
2. Generate an API access key
3. Configuring Cloudera client with the API access key

Instead of steps 2 and 3, you can use the interactive login described in Logging into the CDP CLI/SDK.

Installing Cloudera client

You can install the Cloudera client through pip on Linux, macOS, or Windows. The Cloudera client works with Python version 3.6 or later.

For more information, refer to the following instructions. Note that similar instructions are available from the Cloudera web interface from the Help menu in the bottom left corner > Download CLI.

For current information on changes to the CLI, see the API changelog.

Install Cloudera client on Linux

Install Cloudera client on Linux and verify that the CDP CLI client is working.

Procedure

1. To avoid conflicts with older versions of Python or other packages, Cloudera recommends that you install the Cloudera client in a virtual environment. Use the virtualenv tool to create a virtual environment and install the client.

2. Use pip to install the client on a virtual environment:

```
mkdir ~/cdpclienv
```

```
virtualenv ~/cdpclienv
```

```
source ~/cdpclienv/bin/activate
```

```
~/cdpclienv/bin/pip install cdpcli
```

3. To upgrade the Cloudera client to the latest version, run the following command:

```
~/cdpclienv/bin/pip install --upgrade cdpcli
```

4. After the client installation process is complete, run the following command to confirm that the Cloudera client is working:
   - If virtualenv is activated: cdp --version
   - If virtualenv is not activated: ~/cdpclienv/bin/cdp --version

Install Cloudera client on macOS

The version of Python and pip installed with macOS is older than the one required for the Cloudera client. You must install the latest version of Python before you install the Cloudera client.

Install Python on macOS

Install Python and verify if it is working.

Procedure

1. Download and install the latest version of Python for Mac OS X from the Python downloads page.
2. Use the following commands to install pip:

```
$ curl -O https://bootstrap.pypa.io/get-pip.py
```

```
$ python3 get-pip.py --user
```

3. Use the following commands to verify that Python and pip are correctly installed:

```
python3 --version
```

```
pip3 --version
```

Install Cloudera client after Python on macOS

Install the CDP CLI client and verify if it is working.

Procedure

1. Use the following command to install the Cloudera client:

```
$ pip3 install cdpcli --user
```

2. Check the location of your Python directory that contains the client executable files. For example, for Python 3.6.x the executable files are in the following directory:

```
~/Library/Python/3.6/bin
```

3. Add the Python directory that contains the client executable files to the PATH environment variable. For example:

```
echo "~/Library/Python/3.6/bin" | sudo tee -a /etc/paths
```

4. Use the following command to verify that the Cloudera client is correctly installed:

```
cdp --version
```

To upgrade the Cloudera client to the latest version, run the following command:

```
$ pip3 install cdpcli --upgrade --user
```

Install Cloudera client on Windows

You must install Python and pip before you install the Cloudera client.

Install Python and pip on Windows

To install Python and pip:

Procedure

1. Download and install the latest version of Python for Windows from the Python downloads page.
2. Add the Python home and scripts directories to the Windows PATH environment variable.
3. The Python directory can vary for different versions of Python and different versions of Windows. For example, on the Windows 2016 server, the Python 3.6.1 home directory and the scripts directory are in the following locations:

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python36-32\;
```

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python36-32\Scripts\;
```

Install Cloudera client after Python on Windows

To install the Cloudera client:

Procedure

1. Open a Command Prompt window.
2. Use the following commands to verify that Python and pip are correctly installed:

```
python --version
```

```
pip --version
```

3. Use pip to install the Cloudera client using the following command:

```
pip install cdpcli
```

4. Use the following command to verify that the Cloudera client is correctly installed:

```
cdp --version
```

To upgrade the client to the latest version, run the following command:

```
pip install --upgrade cdpcli
```

Logging into the CDP CLI/SDK

You can log into the CDP CLI using two different methods. The interactive method described in this topic integrates with any SAML-compliant external identity provider.

About this task

There are two ways to log into the CDP CLI/SDK. The traditional method requires generating access credentials (an API access key and private key), then configuring the ~/.cdp/credentials file with the key pair. In this method the access credentials are permanent until they are removed from the ~/.cdp/credentials file. If you prefer to use this method of access, you can skip to the following topics: *Generating an API access key* and *Configuring Cloudera client with the API access key*. It is possible that this traditional login method may be disabled by an administrator, and therefore unavailable to some users.

If you would prefer that user access to the CLI/SDK is shorter-lived, you can use the "interactive" method of logging into the CDP CLI/SDK. By default, this login method grants a 12-hour access key to the CLI/SDK. The access key will time out after one hour of inactivity. The interactive method integrates with any SAML-compliant external identity provider. To set up the interactive method of logging into the CLI/SDK, follow the steps below.

Before you begin

1. Determine which identity provider you want to use for login. If you do not specify an IdP in the login command, the login command will use your default IdP, which is typically the oldest identity provider configured under User ManagementIdentity Providers.
2. Verify that the identity provider you are using has been configured for SAML-based sign on.
3. Take note of your Cloudera tenant/account ID. To find it, click your user name in the bottom left hand navigation menu, select Profile, and then copy the Tenant ID field.

Procedure

1. Use the following command to login to the CDP CLI:

```
cdp login --account-id <tenant-account-id> --identity-provider <idp-name>
 --use-device-code
```

- --account-id is required unless it is configured in the ~/.cdp/config file (account_id).
- --identity-provider is optional, but if not specified, the command uses the default IdP. Give the name or the CRN of the desired IdP as listed in the details page for the IdP, which you can access from User Management Identity Providers .
- --use-device-code allows for the CDP CLI login on a device that does not have a browser. If you use this parameter in the login command, you are instructed to open a URL from any other device with a browser present, input a code displayed on the CLI screen, then complete the authentication flow in the browser.

  Note: The --use-device-code option is not enabled by default. To have this feature enabled, contact your account team.

Running the command sends a request to the specified IdP, which opens the login page in a browser window.

2. Login to the IdP using your company credentials.

   The browser window will inform you when it is safe to close the window. The access key that is generated by the command is saved to the ~/.cdp/credentials file. You can run a test command in the CLI to verify that you are successfully logged in. If you receive an authentication error, login in again.

Related Information

Configuring Cloudera client

Generating an API access key

Generating an API access key

A Cloudera user account (a user or a machine user) must have API access credentials to access Cloudera services through the CDP CLI or API.

When you use this method to generate an access key and then manually configure the access key in the ~/.cdp/credentials, the access credentials are permanent until they are removed from the ~/.cdp/credentials file. A login command is not required if access credentials are stored in the ~/.cdp/credentials file. If you prefer that the API access key is shorter-lived, refer to the topic *Logging into the CDP CLI/SDK*, which describes a method of logging into the CLI/SDK through any SAML-compliant identity provider.

Required roles: Users who have the IAMUser role can generate an API access key from their own account page. As a Cloudera administrator or PowerUser, you can generate an API access key for all user accounts.

Generate your own access key

Steps

1. Sign in to the Cloudera console.
2. Click on your user name in the bottom left corner and then select Profile.
3. On the user profile page that appears, click Generate Access Key.
4. Cloudera creates the key and displays the information on the screen.

   Copy the access key and private key to a text file and send it to the Cloudera user who requires it. The private key is a very long string of characters. Make sure that you copy the full string. You can optionally download the credentials file containing the access key information.
5. Click OK to exit the access key window.

   Note:

   The Cloudera console displays the API access key immediately after you create it. You must copy or download the access key ID and private key information when it is displayed. Do not exit the console without copying the private key. After you exit the console, there is no other way to view or copy the private key.

Once you've generated the access key, you can configure CDP CLI, SDK, or other utilities that require it.

Generate an access key for another user or machine user (admin only)

Steps

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. On the side navigation panel, click Users.
4. On the Users page, click the name of the user or machine user account for which you want to generate an access key.
5. On the user account page, go to the Access Keys section and click Generate Access Key.

6. Cloudera creates the key and displays the information on the screen.

   Copy the access key and private key to a text file and send it to the Cloudera user who requires it. The private key is a very long string of characters. Make sure that you copy the full string. You can optionally download the credentials file containing the access key information.
7. Click OK to exit the access key window.

   Note:

   The Cloudera console displays the API access key immediately after you create it. You must copy or download the access key ID and private key information when it is displayed. Do not exit the console without copying the private key. After you exit the console, there is no other way to view or copy the private key.

Once you've generated the access key, you can configure CDP CLI, SDK, or other utilities that require it.

Related Information

Logging into the CDP CLI/SDK

Configuring Cloudera client

After you generate the access key, go to the command line and run the following commands to configure the Cloudera client.

Procedure

1. On the command line, run the following command to configure the client with the access key:

   cdp configure
2. Enter the following information at the prompt:

   a. Cloudera Data Platform Access key. Copy and paste the access key ID that you generated in the Cloudera Management Console.
   b. Cloudera Data Platform Private key. Copy and paste the private key that you generated in the Cloudera Management Console.

   The configuration utility creates the following file to store your user credentials:

   ~/.cdp/credentials
3. To verify that the credentials were created correctly, run the following command:

   cdp iam get-user

   The command displays your Cloudera client credentials.
4. Optionally, configure the Cloudera client to use the correct Cloudera Control Plane region. In the ~/.cdp/config file, set the Control Plane region as follows:

   cdp_region = <CONTROL_PLANE_REGION>

   For example: cdp_region = eu-1

   The CDP CLI chooses the endpoint URL in the following order:

   a. The endpoint-url from the command line or config file
   b. The --cdp-region argument from the command line (for example: cdp iam get-user --cdp-region eu-1
   c. The cdp_region from the ~/.cdp/config file
   d. Default to us-west-1

   For more information about Cloudera Control Plane regions, see *Cloudera Control Plane regions*.
5. After you configure the client, go back to the Cloudera Management Console and click OK to exit the access key window. Note that a login command is not required if access credentials are stored in the ~/.cdp/credentials file.

Related Information

Cloudera Control Plane regions

Configuring CLI autocomplete

To configure CLI autocompletion, you must have two pieces of information: the name of the shell you're using and the location of the cdp_completer script.

About this task

The CDP CLI includes a command-completion feature that enables you to use the Tab key to complete a partially entered command. This feature isn't automatically installed, so you need to configure it manually. When you partially enter a command, parameter, or option, the command-completion feature either automatically completes your command or displays a suggested list of commands. To prompt command completion, you partially enter a command and press Tab.

Procedure

1. Locate the CLI completer.

   which cdp_completer
2. Add the completer's folder to your path:
   a) Find your shell's profile script in your user folder.

      For example:

      ~/.bash_profile
   b) Add the following command to your bash_profiler, replacing /usr/local/bin/cdp_completer with the actual location discovered in the previous sub-step:

      complete -C /usr/local/bin/cdp_completer cdp
   c) Reload the profile into the current session to put the changes into effect. Replace .bash_profile with the name of the shell script you discovered in the first sub-step:

      ```
      $ source ~/.bash_profile
      ```
3. Enter cdp <TAB> to display the available options for autocomplete.

CLI reference

The CLI reference documentation is available on GitHub:

- CDP CLI Reference
- Beta CDP CLI Reference

CDP CLI parameter types

This section describes the format of common CLI parameter types.

Timestamp

Format the timestamp parameter according to the ISO 8601 standard.

For example:

- YYYY-MM-DDThh:mm:ss.sssTZD (UTC)
- YYYY-MM-DDThh:mm:ss.sssTZD (with offset)
- YYYY-MM-DD
- Unix time in seconds

Accessing CLI help

CDP CLI includes help that can be accessed by using the cdp help command.

To get more information about a certain CDP CLI, use:

```
cdp <module-name> <command-name> help
```

For example:

```
cdp iam add-ssh-public-key help
```

```
cdp iam help|grep group
```

You can also find all of the CDP CLI commands in the CDP CLI Reference documentation.

Installing Beta CDP CLI

Refer to this documentation if you need to install Beta CDP CLI.

About this task

You may need to install Beta CDP CLI if you would like to access one or more preview features included in Cloudera. In most other cases, you should use the standard CDP CLI. Do not install both the standard and beta CLIs in the same Python environment, as they use the same entry points and will therefore conflict. Either use a separate virtual environment or uninstall the cdpcli first before installing cdpcli-beta.

Procedure

1. Install Python if needed, as described in CDP CLI documentation for your platform. Do not install CDP CLI. Instead, proceed to step 2.
2. Run the following to install Beta CDP CLI:

   ```
   pip3 install cdpcli-beta
   ```

3. Configure access keys as described in Generating an API access key.
4. If you have previously installed Beta CDP CLI, you can update it to the latest version using:

   ```
   pip3 install cdpcli-beta --upgrade --user
   ```

Related Information

Beta CDP CLI Reference

Cloudera Migration Assistant 3.4.0

Migrating from CDH to CDP Public Cloud

Date published: 2023-10-11
Date modified: 2024-09-30

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Migration Assistant Overview....4
Supported Cloudera Public Cloud migration paths.... 4
Release Notes....4

Cloudera Migration Assistant server deployment....7
Deploying Cloudera Migration Assistant locally or with Docker.... 8
Deploying Cloudera Migration Assistant with parcel....13
Enabling TLS/SSL for Cloudera Migration Assistant.... 15
Storing secrets in Vault.... 17

Migrating to Cloudera Public Cloud with Cloudera Migration Assistant..... 18
Reviewing prerequisites before migration.... 18
Registering source clusters.... 19
Scanning the source cluster....21
Creating collections for migration....26
Registering destination clusters.... 27
Migrating from source cluster to destination cluster.... 30

Migrating Spark applications....35

Migrating Oozie workflows.... 38

Migrating SQL queries.... 40

Migrating HBase tables....42

Cloudera Migration Assistant Overview

Cloudera Migration Assistant is a user interface based extensible tool to assist Hadoop (CDH) users to easily migrate data, metadata and certain workloads to the various form factors of Cloudera.

Supported Cloudera Public Cloud migration paths

Cloudera Migration Assistant (previously AM2CM) can be used to migrate from the legacy CDH and Cloudera Private Cloud Base platforms to Cloudera Public Cloud. The supported migration paths vary based on the Cloudera Migration Assistant version.

Cloudera Migration Assistant version	Migration paths	Cloud provider	Workloads	Data
Cloudera Migration Assistant 3.5	Cloudera Private Cloud Base 7.1.7, 7.1.8, 7.1.9 # Cloudera Public Cloud 7.2.18 migration	AWS, Azure	SQL Oozie Spark	HDFS files HMS tables HBase tables
Cloudera Migration Assistant 3.4	Cloudera Private Cloud Base 7.1.7, 7.1.8, 7.1.9 # Cloudera Public Cloud 7.2.18 migration	AWS, Azure	SQL Oozie Spark	HDFS files HMS tables HBase tables
Cloudera Migration Assistant 3.3	CDH 5.1.x # Cloudera Public Cloud 7.2.x CDH 6.3.x # Cloudera Public Cloud 7.2.x Cloudera Private Cloud Base 7.1.7# Cloudera Public Cloud 7.2.x	AWS, Azure	SQL Oozie	HDFS files HMS tables HBase tables
Cloudera Migration Assistant 3.2	CDH 5.1.x # Cloudera Public Cloud 7.2.x CDH 6.3.x # Cloudera Public Cloud 7.2.x	AWS	SQL Oozie (Hive action)	HDFS files HMS tables HBase tables
Cloudera Migration Assistant 3.0	CDH 5.1.x # Cloudera Public Cloud 7.2.x CDH 6.3.x # Cloudera Public Cloud 7.2.x	AWS	SQL	HDFS files HMS tables
Cloudera Migration Assistant 2.8	CDH 6.3.x # Cloudera Public Cloud 7.2.17 (Technical Preview)	AWS	SQL	HDFS files HMS tables

Release Notes

Learn about the known issues, and fixed issues and behavioral changes in Cloudera Migration Assistant.

3.5.0

What's new

- AMCM-2422 Enhanced **Migration Dashboard** with Component Versions
- AMCM-793 Managing Secrets with **Vault**
- AMCM-2455 **Mask** supplied **passwords** in configuration and log files
- AMCM-2308 **Spark3** to Spark3 Support

- AMCM-2400 Execution Instructions for **Livy**
- AMCM-2594 Parcel compatibility check to CSD
- Various CVE fixes

Behavioral changes

- Support for upgrading from HDP to Cloudera platform is discontinued from this release.

3.4.1

What's new

- AMCM-2489 Configurable JVM heap size for CDH discovery tool

Fixed issues

- AMCM-2692 Stop All Services In The Cluster - fails with non-default password
- AMCM-2637 Spark application parameters with single - won't work
- AMCM-2570 Remove Label assignment removes HDFS locations
- AMCM-2650 SQL Workload Migration doesn't replicate its SQL file from HDFS
- AMCM-2625 Multiple HiveSQL migration ignores the HiveSQL files
- AMCM-2626 NPE when trying to get source server clusters after db upgrade
- AMCM-2624 /root/.cdp/credentials Missing for hive migration
- AMCM-2636 Use master node in execution Instruction

Known issues

- AMCM-2457 Python 3.12.x dependency collision on Cloudera Migration Assistant server node

3.4.0

What's new

- AMCM-1168 Spark Application Migration
  - Exploring Past Spark Workloads via Spark History Server
  - Spark Workload's dependency discovery based on spark-submit and application parameters
  - Limitations
    - CDH6 and Cloudera Private Cloud Base as Source with Spark 2
    - No spark code change available
    - No automated jar and Python package dependency discovery
- AMCM-2426 Store SSH Settings as Credentials
- AMCM-2300 Upgradability - Tech Preview
- Several UX improvements and fixes

Known issues

- AMCM-2457 Python 3.12.x dependency collision on Cloudera Migration Assistant server node

3.3.1

What's new

- Parcel deployment - available as Technical Preview

  The parcel deployment enables you to install Cloudera Migration Assistant as a service in Cloudera Manager.

Fixed issues

- AMCM-2273 Cloudera Migration Assistant doesn't handle long clusternames
- AMCM-2233 Create a field to manage the number of Ansible forks
- AMCM-2232 Fix user settings validation on target registration when changing clusters

- AMCM-2218 Replace Browser icon (Favico) to Cloudera Migration Assistant icon small version
- AMCM-2067 Obtain Hive HMS DB Password from User Directly
- Pre-checks on Cloudera Migration Assistant Server Start:
  - AMCM-2188 Checking if Java11 and JAVA_HOME exists
  - AMCM-2188 Checking Java before starting the server
  - AMCM-2186 Checking Python version

3.3.0

What's new

- AMCM-1764 Azure Support
- AMCM-1418 Cloudera Private Cloud Base to Public Cloud Migration (without security metadata migration)
- AMCM-1361 Oozie Migration with Map-Reduce action and genric support for all other action types
- AMCM-1765 Cloudera Migration Assistant Landing Page
- AMCM-1878 Side-bar Redesign
- AMCM-2135 One-time HBase Migration without Cloudera Replication Manager

Limitations

You need to manually migrate Ranger policies from Cloudera Private Cloud Base to Cloudera Public Cloud as described in Migrating from source cluster to destination cluster.

3.2.1

Fixed Issues

- AMCM-1879 Missing airgapped for localhost (#1696)
- AMCM-1908 Hive SQL Export button does not work (#1693)
- AMCM-1906 Fix Broken background color (#1691)
- AMCM-1888 Show Refresh option of scanned data tables as in UI design - rebase fix (#1686)
- AMCM-1880 ps is missing from the rhel8 docker image (#1659) (#1685)
- AMCM-1902 Make navigation panel not scrollable (#1679) (#1684)
- AMCM-1872 Filtered scan results are reset after switch to another page (#1655) (#1678)
- AMCM-1830 discovery_bundle_builder fails if no mysql-connector-java.jar is found (#1682)
- AMCM-1892 L&S Data migration Cloudera Replication Manager policy should have default value for frequencyInSec (#1681) (#1683)
- AMCM-1888 Show Refresh option of scanned data tables as in UI design (#1666) (#1670)
- AMCM-1887 Fix Mapping discrepancies (#1662) (#1671)
- AMCM-1891 Change API info (#1668) (#1675)
- AMCM-1867 Set step status before perform (#1667) (#1674)
- AMCM-1820 Toast message appears unexpectedly (#1663) (#1669)
- AMCM-1871 Fix wrong position of collection component (#1643) (#1672)

Known Issues

AMCM-1860 Airgapped download

The following files needs to be downloaded and copied manually before any --airgapped install.

- <CMA_ROOT_DIR>
  - atlas-migration-exporter-0.8.0.2.6.6.0-332.tar.gz
  - jdbc-drivers/
    - mysql-connector-java-5.1.46.jar
    - ojdbc8.jar
    - postgresql-42.3.4.jar

Only add the JDBC driver your cluster uses.

Behavioral Changes

- /info endpoint API response changed to correctly return the product name (AMCM-1891)

Fixed Common Vulnerabilities and Exposures

Learn more about the Common vulnerabilities and Exposures (CVEs) that were fixed in this release.

- CVE-2024-1597: org.postgresql:postgresql vulnerable to SQL Injection via line comment generation - AMCM-1910
- /info endpoint API response changed to correctly return the product name (AMCM-1891)

3.2.0

Known Issues

HBase migration from CDH needs extra parcels Installed

In order to use Cloudera Migration Assistant for you need obtain the following parcels from Cloudera Support :

- CDH5 : CLOUDERA_OPDB_REPLICATION-1.0-1.CLOUDERA_OPDB_REPLICATION5.14.4.p0.31473501-el7.parcel
- CDH6 : CLOUDERA_OPDB_REPLICATION-1.0-1.CLOUDERA_OPDB_REPLICATION6.3.3.p0.8959316-el7.parcel

These along with their hash files need to be copied to parcels directory in Cloudera Migration Assistant root directory

Cloudera Migration Assistant server deployment

You can deploy the Cloudera Migration Assistant server (cma-server) to any of your source cluster nodes, or in the case of local or docker mode, to an external node that has visibility to the cluster. It requires 1.5 GB of extra memory, and unless you are in parcel deployment mode, you can install Cloudera Migration Assistant with or without internet connection. You can choose between deploying the Cloudera Migration Assistant server locally, using Docker or with parcel.

Dependencies

The following components must be installed on the cma-server host:

- Python 3.8.12 or higher version
- Docker deployment requires docker 20+ or higher versions
- Local and parcel feployment requires JDK11 (with JAVA_HOME set)

In case you do not have JDK11 installed on you cluster, you can download it using the following commands:

```
wget https://download.java.net/java/GA/jdk11/9/GPL/openjdk-11.0.2_linux-x64_
bin.tar.gz -C /usr/java/
ln -s /usr/java/jdk-11 /usr/java/jdk-11.0.2/
```

You can skip specifying Java Home if it is located on any of the following default paths:

- /usr/lib/jvm/java-11
- /usr/java/jdk-11
- /usr/lib/jvm/jdk-11
- /usr/lib64/jvm/jdk-11
- /usr/lib/jvm/zulu-11
- /usr/lib/jvm/zulu11
- /usr/lib/jvm/java-11-zulu-openjdk
- /usr/lib/jvm/java-11-oracle

When setting up Cloudera Migration Assistant without internet connection, the installation script ensures to install the required Python dependencies without using internet connection, and creates the Python Package Index (pypi) repository locally.

You can view the list of components installed with Cloudera Migration Assistant under the following directory:

```
cma-[***VERSION NUMBER***]/am2cm-ansible/python_requirements/
```

The Python requirements file details the Python packages that are needed to set up the virtual environment to run Cloudera Migration Assistant. No internet connection is used to download these components when setting up Cloudera Migration Assistant in an air-gapped network.

Note: Even though the installation of Cloudera Migration Assistant can be completed without internet connection, you need to ensure that you have internet connection when downloading the JDBC drivers and Atlas artifacts. The following files must be manually downloaded and copied before installing Cloudera Migration Assistant in an air-gapped environment:

- *[***CMA ROOT DIRECTORY***]*
  - atlas-migration-exporter-0.8.0.2.6.6.0-332.tar.gz
  - jdbc-drivers/
    - mysql-connector-java-5.1.46.jar
    - ojdbc8.jar
    - postgresql-42.3.4.jar

    Only add the JDBC driver your cluster uses.

Deploying Cloudera Migration Assistant locally or with Docker

Learn more about how to deploy Cloudera Migration Assistant locally and in docker with or without internet connection.

Procedure

1. Download the binaries of the Cloudera Migration Assistant tool from https://archive.cloudera.com/cma/3.5/tars/ with one of the following commands:

   For With internet connection

   - ```
     wget https://archive.cloudera.com/cma/3.5/tars/cma-3.5.0.0-8-bin.tar.gz
     ```
   - ```
     curl https://archive.cloudera.com/cma/3.5/tars/cma-3.5.0.0-8-bin.tar.gz
     ```

   For Without internet connection

   - ```
     mkdir cma-3.5
     cd cma-3.5
     wget https://archive.cloudera.com/cma/3.5/tars/cma-3.5.0.0-8-bin.tar.gz
     wget https://archive.cloudera.com/cma/3.5/tars/cma-extras-gpl-3.5.0.0-8-bin.tar.gz-bin.tar.gz
     ```
   - ```
     mkdir cma-3.5
     cd cma-3.5
     curl https://archive.cloudera.com/cma/3.5/tars/cma-3.5.0.0-8-bin.tar.gz
     curl https://archive.cloudera.com/cma/3.5/tars/cma-extras-gpl-3.5.0.0-8-bin.tar.gz-bin.tar.gz
     ```

   When the required binaries are downloaded successfully, the directory structure should look like the following example:

   ```
   drwxr-xr-x 14 testuser testuser 4096 febr 27 13:21 cma-3.5.0.0-8/
                                                -rw-rw-r-- 1 testuser
   testuser 518140466 febr 27 13:28 cma-3.5.0.0-8-bin.tar.gz
                                                -rw-rw-r-- 1 testuser
   testuser 85089637 febr 27 13:28 cma-extras-gpl-3.5.0.0-8.tar.gz
   ```

   The supported version of Cloudera Migration Assistant is 2.8.0 and higher.

2. Extract the downloaded file using the following command:

   ```
   tar xzf cma-3.5.0.0-8-bin.tar.gz
   ```

3. Start the Cloudera Migration Assistant server locally or in a Docker container. The preferred method is the Docker mode.

   Ensure that Python 3.8.12 or a higher version is installed on the host. In case you do not define the python executable when running the script, you will be prompted to enter the python executable path.

   - Docker

     For With internet connection

     Run the cma-docker.sh script in the untarred top-level folder to launch the Cloudera Migration Assistant server in a Docker container. cma-*[***VERSION NUMBER***]*/bin/cma-docker.sh --start

     Note: The script creates the docker image if necessary. Additionally, the script provides the following operations to manage the Cloudera Migration Assistant Docker container: start, stop, restart, or rebuild. If you want to explore other available options, run the following command: cma-*[***VERSION NUMBER***]*/bin/cma-docker.sh --help.

     For Without internet connection

     ```
     cd cma-[***VERSION NUMBER***]/
     bin/cma-docker.sh --start --airgapped --python-executable=python3
     ```

     Note: If the GPL file is not located in the same directory as the Cloudera Migration Assistant file, you can use the following command, where you define the path of the GPL file:

     ```
     cma-[***VERSION NUMBER***]/bin/cma-docker.sh --start --
     airgapped
             --cma-extras-gpl-tar-location=[***ABSOLUTE PATH
      TO EXTRAS GPL***]
     ```

     Check that the local pypi repository is installed correctly.

     ```
     netstat -atnp | grep 9003
     (Not all processes could be identified, non-owned process info
      will not be shown, you would have to be root to see it all.)
     tcp        0      0 0.0.0.0:9003            0.0.0.0:*
      LISTEN      201503/python3
     ```

   - Locally

     For With internet connection

     Run the cma-local.sh script in the untarred top-level folder and follow its instructions to launch the Cloudera Migration Assistant server locally. cma-*[***VERSION NUMBER***]*/bin/cma-local.sh --start

     Note: The script creates a Python virtual environment in the top-level folder where the dependencies are installed. Additionally, the script provides the following operations to manage the Cloudera Migration Assistant locally: start, stop, restart, or rebuild. To explore other available options, run the following command: cma-*[***VERSION NUMBER***]*/bin/cma-local.sh --help.

     For Without internet connection

     ```
     cd cma-[***VERSION NUMBER***]/
     ```

```
bin/cma-local.sh --start --airgapped --python-executable=python3
```

Note: If the GPL file is not located in the same directory as the Cloudera Migration Assistant file, you can use the following command, where you define the path of the GPL file:

```
cma-[***VERSION NUMBER***]/bin/cma-local.sh --start --
airgapped
        --cma-extras-gpl-tar-location=[***ABSOLUTE PATH
 TO EXTRAS GPL***]
```

Check that the local pypi repository is installed correctly.

```
netstat -atnp | grep 9003
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:9003            0.0.0.0:*
 LISTEN      201503/python3
```

4. Access http://localhost:8090 in a browser, such as Chrome or Firefox, to open the Cloudera Migration Assistant tool.

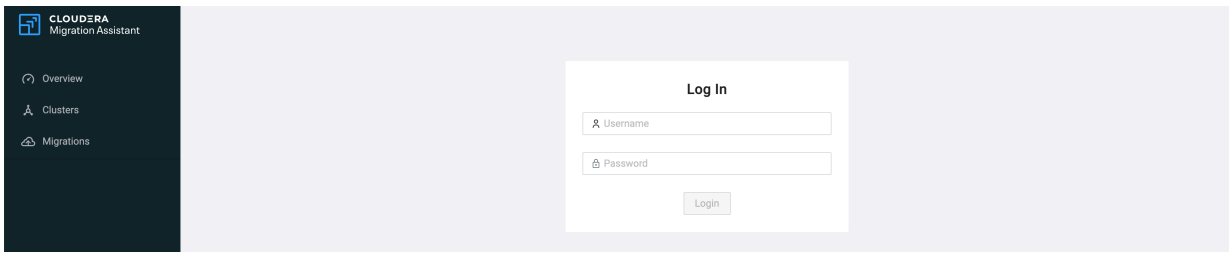

The default user password pair is admin-admin, but you can change your password at the user profile page.

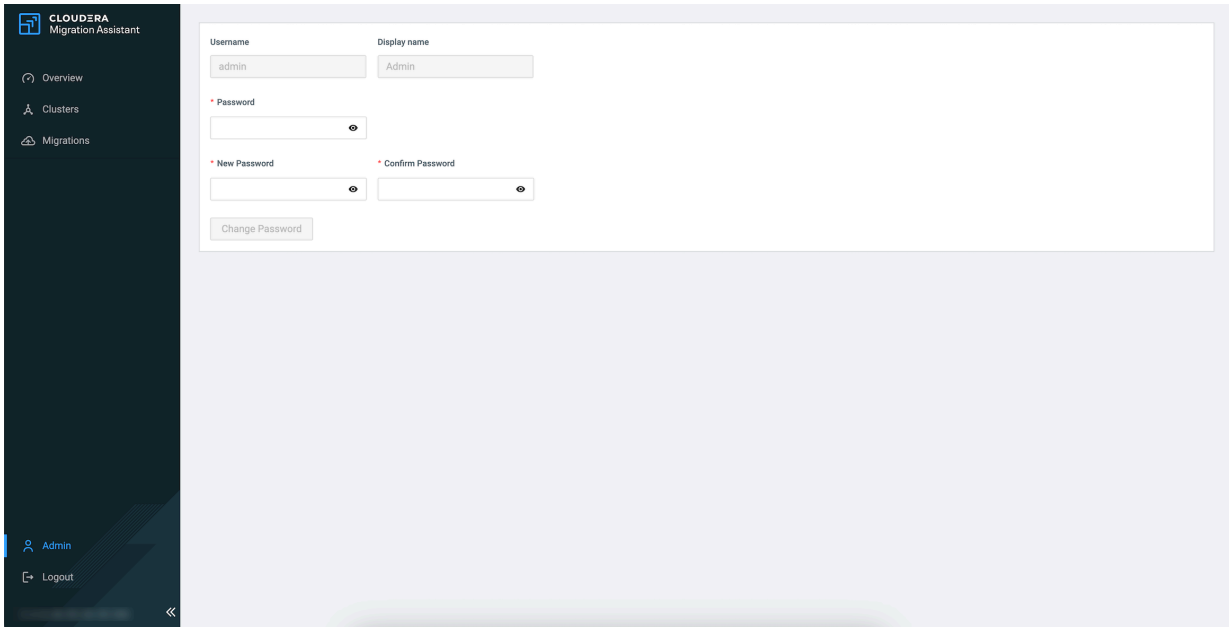

After changing the password, you can start using the Cloudera Migration Assistant.

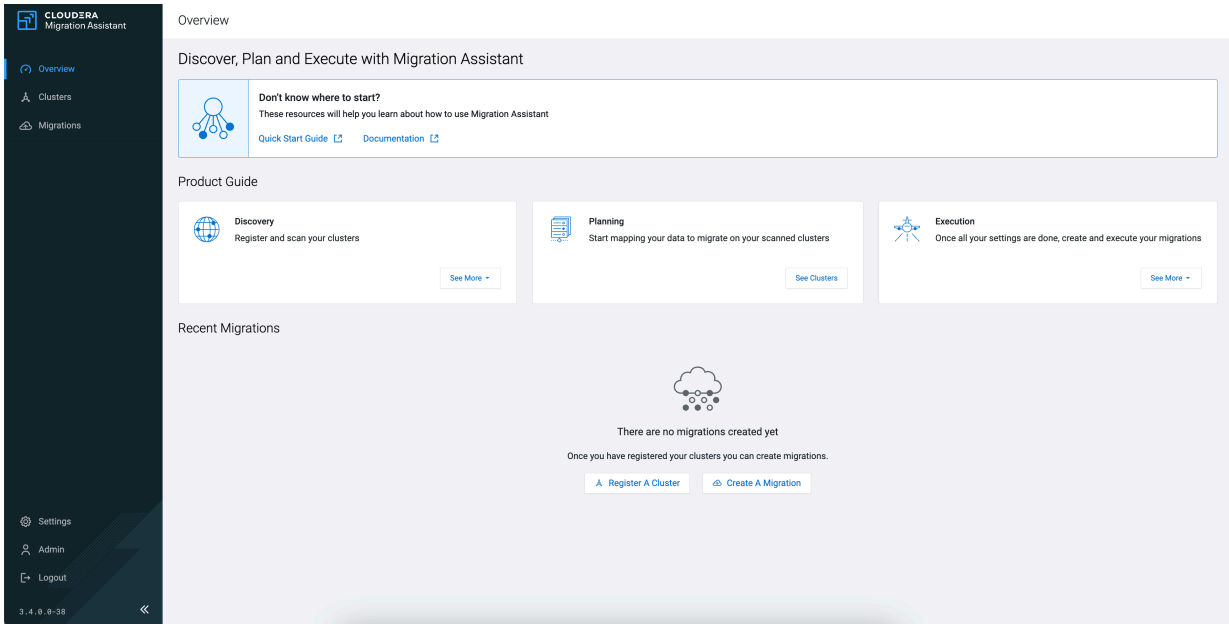

Deploying Cloudera Migration Assistant with parcel

Learn more about how to deploy Cloudera Migration Assistant with a parcel in Cloudera Manager.

About this task

Cloudera Migration Assistant can be an add-on service in Cloudera Manager. To deploy Cloudera Migration Assistant with a parcel, you need to upload the Cloudera Migration Assistant Custom Service Descriptor (CSD) files to the default CSD directory, and add the Cloudera Migration Assistant parcel to your cluster using Cloudera Manager.

Note:

The parcel deployment of Cloudera Migration Assistant is in Technical Preview and not ready for production deployment. Cloudera encourages you to explore these features in non-production environments and provide feedback on your experiences through the *Cloudera Community Forums*.

Procedure

1. Copy the Cloudera Migration Assistant CSD files to /opt/cloudera/csd/ directory on the Cloudera Manager node:

   ```
   wget -P /opt/cloudera/csd/ https://archive.cloudera.com/cma/3.5/csd/CMA-3.5.0.0-8.jar
   ```

   Cloudera Manager automatically detects the CSD files.
2. Change the ownership of the CSD files.

   ```
   chown cloudera-scm:cloudera-scm /opt/cloudera/csd/CMA-3.5.0.0-8.jar
   ```
3. Restart Cloudera Manager and CMS services for the changes to take effect.

   ```
   systemctl restart cloudera-scm-server
   ```
4. Log into Cloudera Manager.
5. Select Hosts Parcels in the left navigation bar.

6. Search for Cloudera Migration Assistant, and click Download to download the parcel to the local repository.

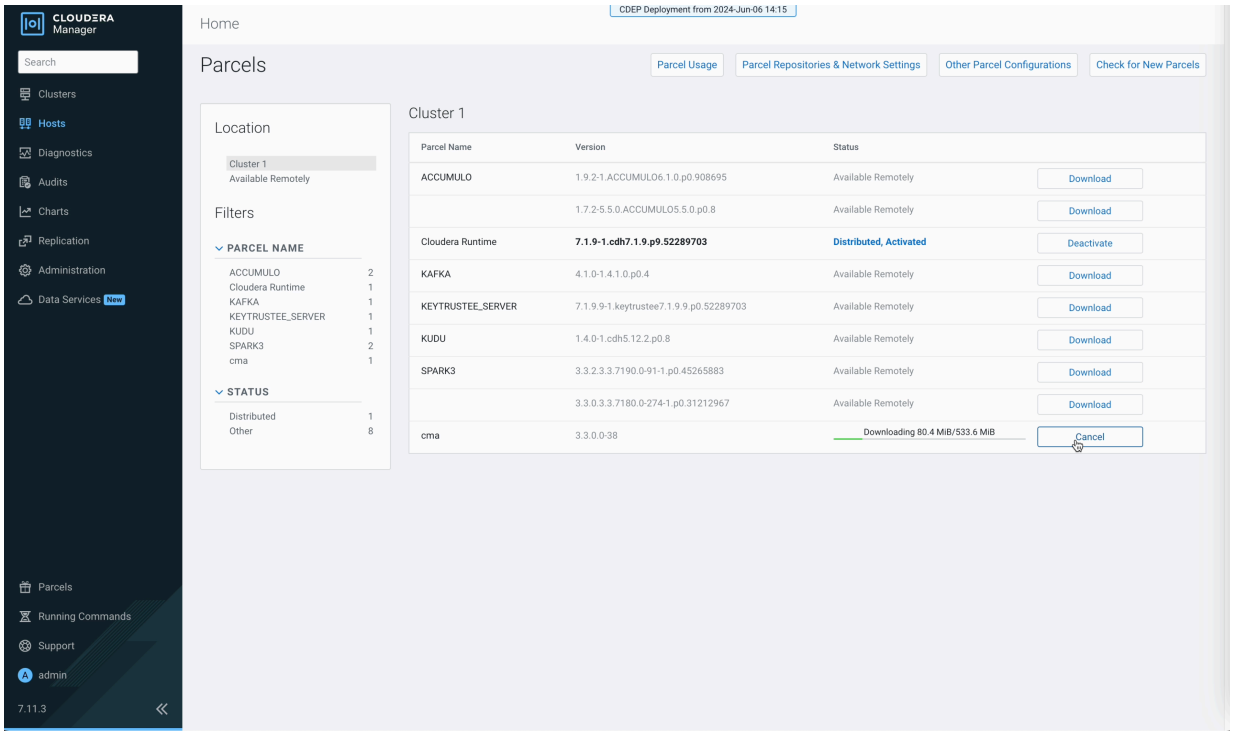

7. After the download is completed, click Distribute to distribute the parcel to all clusters.
8. After the parcel is distributed, click Activate to activate the parcel.
9. Click OK when confirmation is required.
10. Click Clusters on the left navigation pane.
11. Select the drop-down menu to the right of your cluster.
12. Select Add Service.
13. From the list, select CMA as the type of service, then click Continue.
    The **Add CMA Service** wizard opens.
14. Assign the Cloudera Migration Assistant server role to the hosts where you Python 3.8.12 and JDK11 installed, and click Continue.
15. Click Continue after reviewing the configurations of the Cloudera Migration Assistant service.
    The first run of the Cloudera Migration Assistant service starts. When the command finishes, the Cloudera Migration Assistant service is added to the cluster.

16. Access the Cloudera Migration Assistant User Interface (UI).

After going back to the homepage of your cluster, open the Cloudera Migration Assistant service page, and click on the CMA Server UI tab that opens the Cloudera Migration Assistant UI.

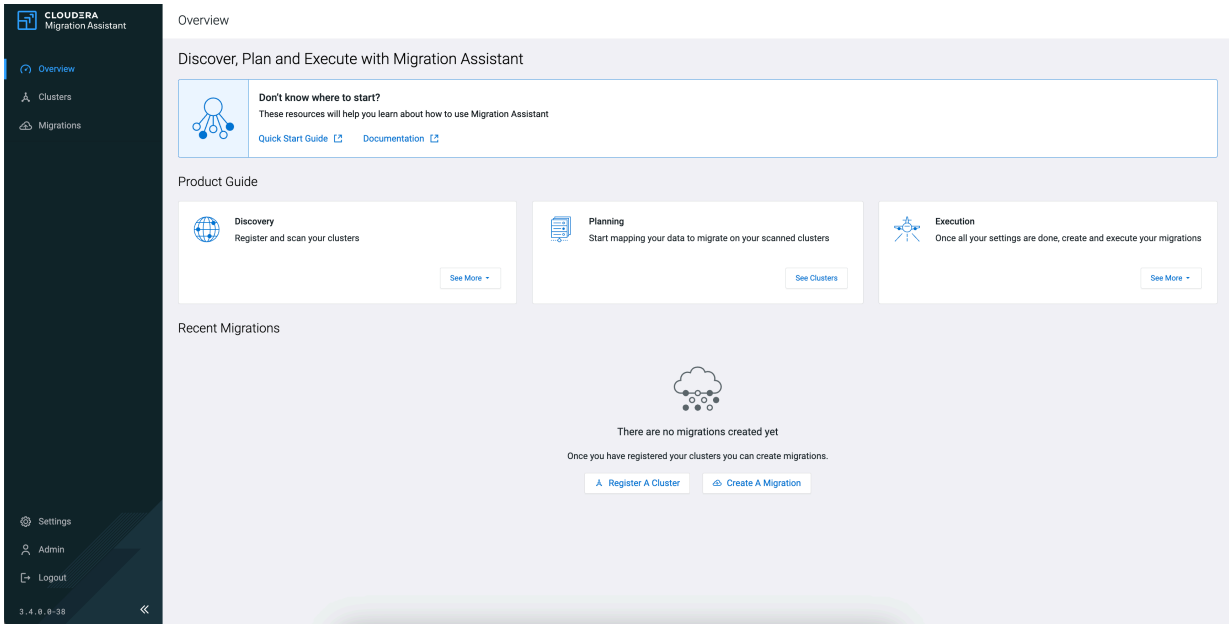

Related Information

Add-on Services

Registering source clusters

Enabling TLS/SSL for Cloudera Migration Assistant

Learn more about how to enable and configure TLS/SSL for Cloudera Migration Assistant when deploying the Cloudera Migration Assistant server locally or with Docker.

Before you begin

- Ensure that you have a PKCS12 keystore created with the cma key alias. JKS keystore is also supported.

Procedure

1. Click Settings on the homepage of Cloudera Migration Assistant.

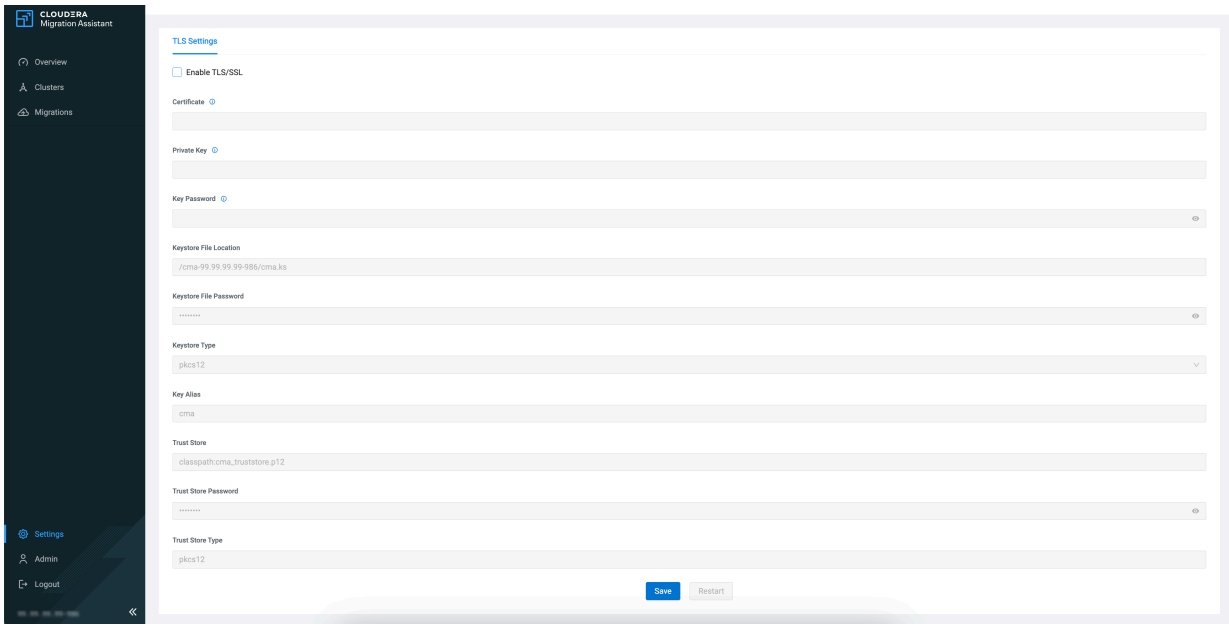

2. Enable TLS/SSL using the checkbox on the **TLS Settings** page.

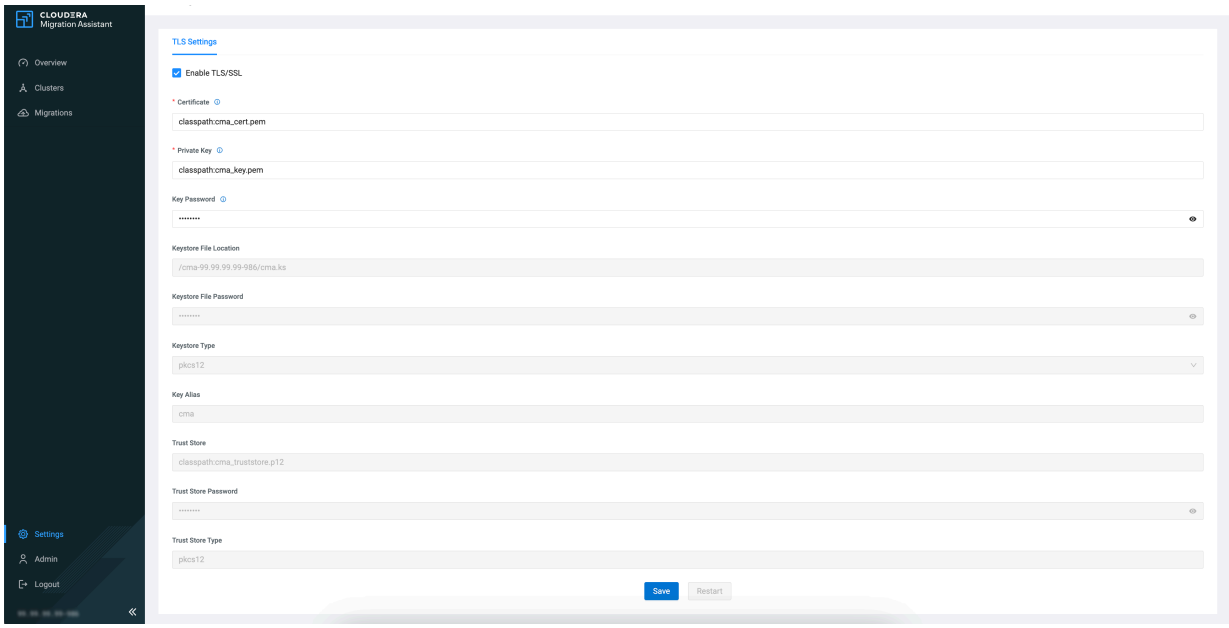

3. Provide the TLS/SSL Certificate in PEM format.
4. Provide the Private Key in PEM format.
5. Provide the Private Key Password if the Private Key is encrypted.
6. Provide the Keystore File Location.
7. Provide the Keystore File Password.
8. Select the Keystore Type from the drop-down menu.
9. Provide a Key Alias.
10. Provide the Trust Store.
11. Provide the Trust Store Password.
12. Provide the Trust Store Type.

13. Click Save.
14. Click Restart after saving the TLS/SSL configurtations.

Results

After restarting, the Cloudera Migration Assistant server listens on the default port 8090 (HTTP) and 8093 (HTTPS), and all HTTP requests are redirected to the HTTPS port.

Storing secrets in Vault

You can store your secrets in a local Vault instance managed by Cloudera Migration Assistant. If the Vault is configured, no credentials are stored in the database, and the files created during the migration process contain only Vault paths. Additionally, credentials stored in the Vault are also masked on the UI in this case.

Note: Storing secrects in Vault is in technical preview.

Local or Docker Deployment

Starting CMA with Vault for the first time

Starting Cloudera Migration Assistant with the --vault option will start and configure a local Vault server, and configure the Cloudera Migration Assistant Server to connect to it. Once the connection is set up, the Cloudera Migration Assistant will store the credentials on the Vault Server.

1. When Cloudera Migration Assistant Server is not running, start Cloudera Migration Assistant with the following command:

```
cma-[***VERSION NUMBER***]/bin/cma start --vault
```

2. When Cloudera Migration Assistant Server is already running, use the following command to restart Cloudera Migration Assistant with Vault:

```
cma-[***VERSION NUMBER***]/bin/cma restart --vault
```

To restart the Vault server, you can use the following command:

```
cma-[***VERSION NUMBER***]/bin/cma vault restart
```

After the initial configuration is complete, Cloudera Migration Assistant Server expects the Vault Server to be running. If you want to stop the Vault, it is recommend that you stop CMA as well using the following commands:

```
cma-[***VERSION NUMBER***]/bin/cma vault stop
cma-[***VERSION NUMBER***]/bin/cma stop
```

Note:

In the docker deployment, the Vault runs in the Cloudera Migration Assistant container so it cannot be started, stopped or restarted without the Cloudera Migration Assistant.

Parcel Deployment

When you deploy Cloudera Migration Assistant in parcel mode, the Vault Server role is installed on the Cloudera Migration Assistant Server node, and configured by default.

Note:

After the initial configuration is complete, Cloudera Migration Assistant Server and Vault Server roles can be started, stopped or restarted separately in Cloudera Manager.

Migrating to Cloudera Public Cloud with Cloudera Migration Assistant

The following steps will guide you through how to migrate your data, metadata and workload from a CDH or Cloudera Private Cloud Base cluster to a Cloudera Public Cloud cluster.

Reviewing prerequisites before migration

Before migrating from CDH 5, CDH 6 or Cloudera Private Cloud Base to Cloudera Public Cloud, review the list of prerequisites that are required for the migration process.

- Ensure that the Cloudera Migration Assistant server is deployed as described in *Setting up Cloudera Migration Assistant server*.
- The CDH 5 source cluster minimum version requirement is CDH 5.16.1 and CDH 5.16.2 in case of HBase migration.
- CDH 6 source cluster minimum version requirement is CDH 6.3.3.
- The Cloudera Private Cloud Base source cluster minimum version requirement is 7.1.7.
- For HBase migration, you need either of the following parcels procured from Cloudera Professional Services:
  - CLOUDERA_OPDB_REPLICATION-1.0-1.CLOUDERA_OPDB_REPLICATION5.14.4.p0.31473501-el7.parcel
  - CLOUDERA_OPDB_REPLICATION-1.0-1.CLOUDERA_OPDB_REPLICATION6.3.3.p0.8959316-el7.parcel
- For data and metadata migration, you need a Data Lake cluster already created in a Cloudera Public Cloud environment. To create a Data Lake cluster, you can follow the process described in *Registering an AWS environment* and *Registering an Azure environment* based on your cloud provider.
- For a Hive workload migration, you need a Cloudera Data Engineering Data Hub already created in a Cloudera Public Cloud environment. To create a Cloudera Data Engineering Data Hub cluster, you can follow the process described in *Creating a cluster on AWS* and *Creating a cluster on Azure* based on your cloud provider.
- You must use the Cluster Connectivity Manager to manually register the source CDH cluster as a classic cluster in the Cloudera Control Plane, following the process described in *Adding a CDH cluster (CCMv2)*.
- Information to gather before you begin the migration:
  - For the source CDH cluster: The Cloudera Manager URL, Admin username and password, SSH user, port, and private key of source nodes
  - For the destination Cloudera cluster/environment: Cloudera Control Plane URL, Admin username and password, SSH user, port, and private key
  - In S3: S3 bucket access key and S3 bucket secret key, S3 credential name. Potentially, you might also need the S3 bucket base path for HDFS files, S3 bucket path for Hive external tables (these paths should auto-fill from the selected destination cluster, but can be changed if needed)
- The Cloudera Manager node of the source CDH cluster must have Python 3.8.12 or higher installed.
- Redaction needs to be off in Cloudera Manager. To disable redaction in Cloudera Manager, you can follow the process described in *Disabling Redaction of sensitive information*.

Related Information

Setting up CMA server

Registering an AWS environment

Creating a cluster on AWS

Cluster Connectivity Manager
Adding a CDH cluster (CCMv2)
Disabling Redaction of sensitive information when using the Cloudera Manager API

Registering source clusters

To migrate from CDH to Cloudera Public Cloud, you need to register the CDH or Cloudera Private Cloud Base cluster as a source from which the data, metadata and workload will be migrated.

Before you begin

Make a note of the following information about the CDH cluster to complete the source registration:

- Cloudera Manager URL
- Admin username and password for Cloudera Manager
- SSH user, port, and private key

Procedure

1. Click See More Register A New Source on the homepage of Cloudera Migration Assistant to register a cluster that will be used as a source for the migration.

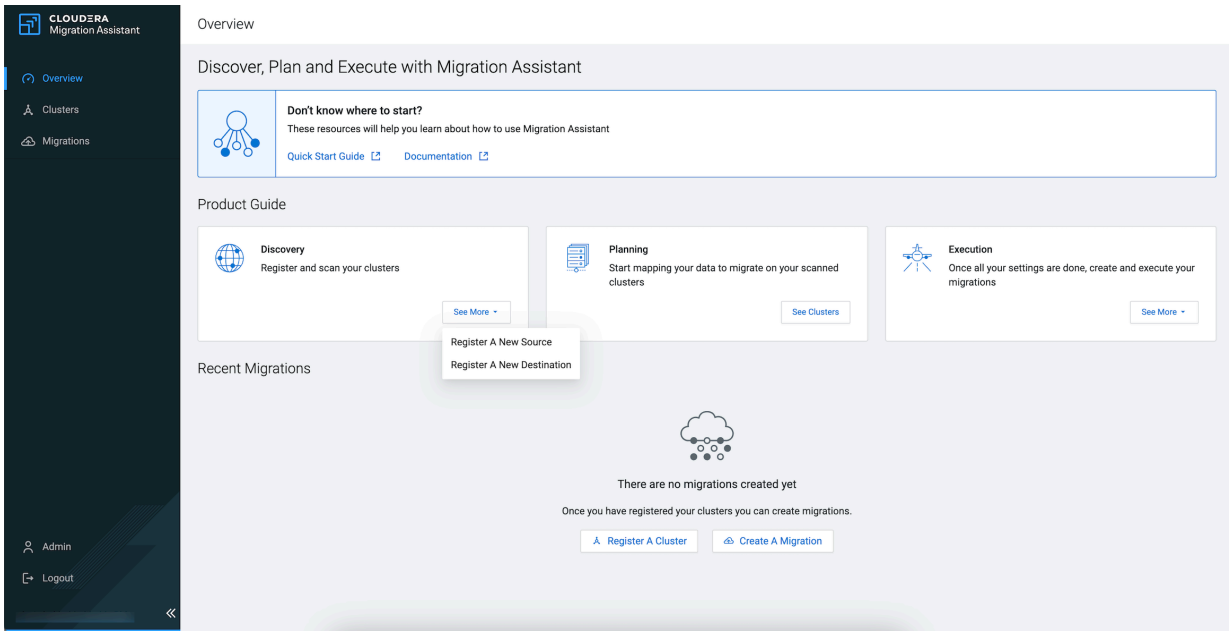

   Alternatively, you can open the New Cluster wizard by selecting Clusters on the left navigation pane, and clicking New Source .

2. Select Cloudera Distributed Hadoop 5, Cloudera Distributed Hadoop 6 or Cloudera Private Cloud Base as **Source Type**.
3. Provide the URL of Cloudera Manager that belongs to the CDH 5, CDH 6 or Cloudera Private Cloud Base cluster.

   The URL needs to be in the following format:

```
https://[***CLOUDERA MANAGER HOSTNAME***]:[***CLOUDERA MANAGER PORT***]/
```

4. Provide the Admin user and Admin password of Cloudera Manager.
5. Click Next.
6. Choose the cluster based on the Cluster Name that you want to use for the migration.

   The drop-down list contains all of the clusters that are managed by Cloudera Manager on the provided host.

7. Click Next.
8. Select the Configuration Preference based on which authentication method you prefer.
   - Choose Use existing if you want to use the SSH configuration and keys of the user running Cloudera Migration Assistant server to access the hosts.
   - Choose New if you want to use a newly provided SSH key to configure Ansible automatically.
     a. Provide the SSH User and SSH Port.
     b. Copy the SSH Key to the SSH Key box or upload a .pem file containing the key.
     c. If you want to use Password, provide the password and make sure the sshpass utility is installed.

     Note: The user specified during SSH configuration must have **passwordless sudo rights**.
9. Click Create.

Results

The registered CDH or Cloudera Private Cloud Base cluster is listed on the **Clusters** page.

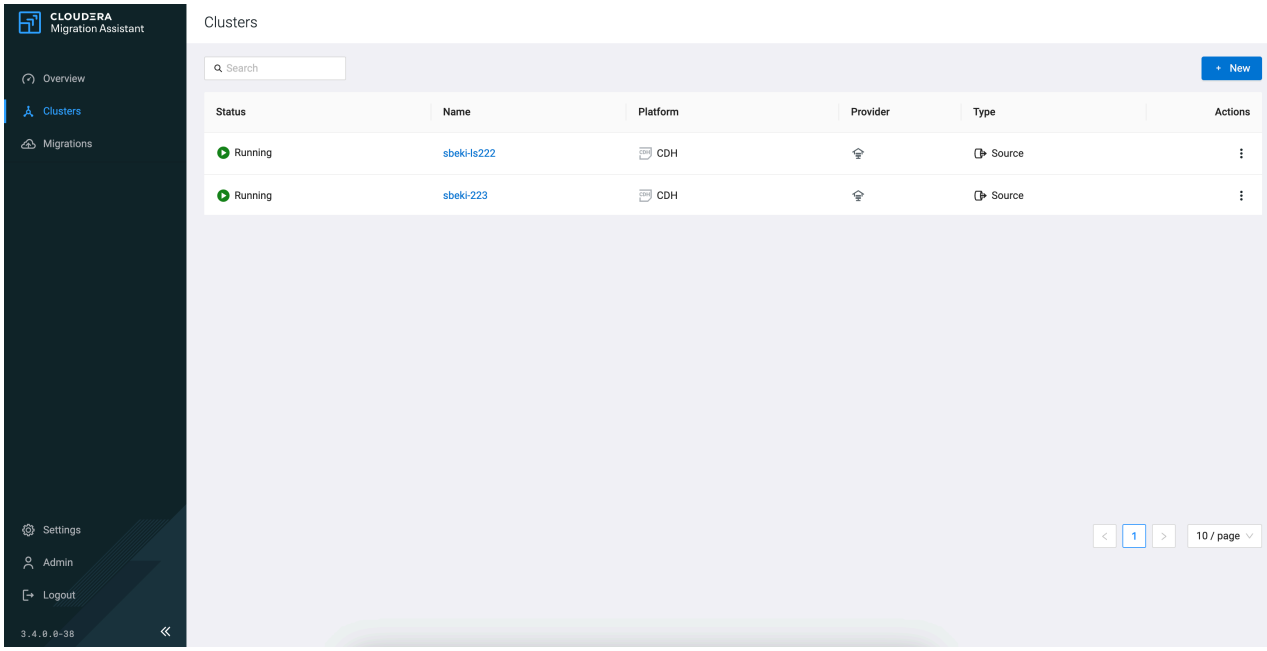

You can review the details and services of the cluster by clicking on the Name of the cluster.

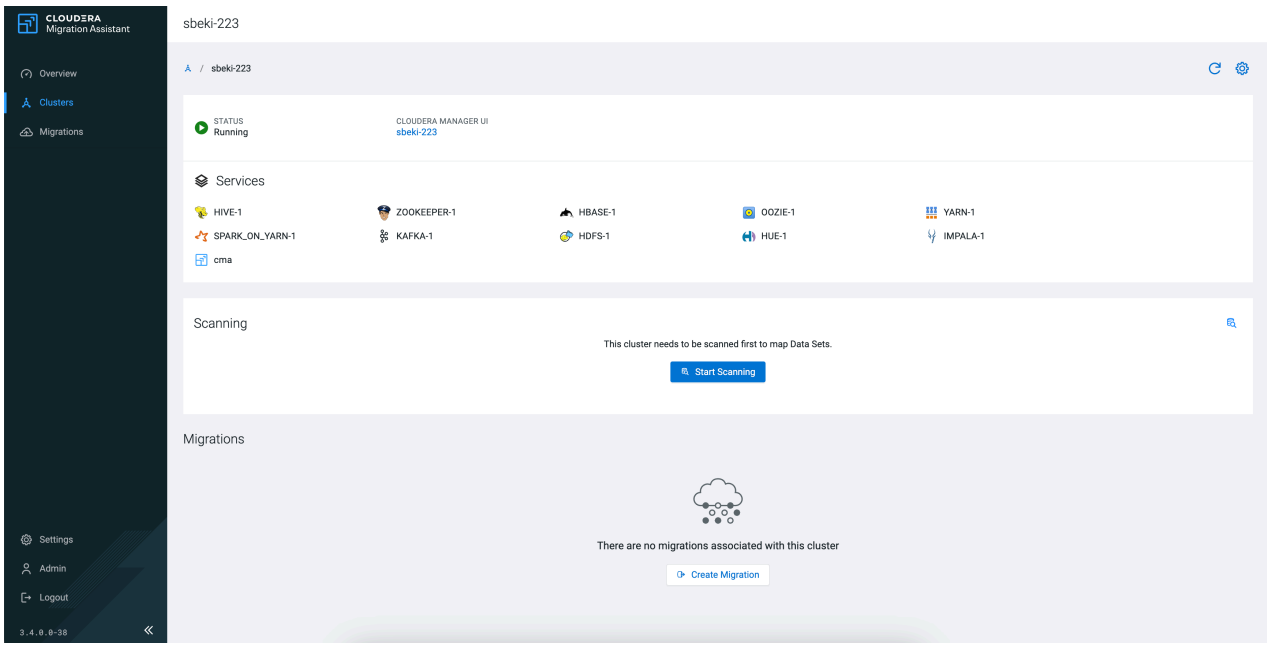

What to do next

Scan the data and workloads on the registered source cluster and add labels to specify the objects, which should be included in the migration.

Scanning the source cluster

You need to scan the CDH or Cloudera Private Cloud Base source cluster to identify the available datasets and workloads that can be migrated. Scanning also enables you to review and resolve syntax errors that can occur after the migration.

Procedure

1. Click on the CDH or Cloudera Private Cloud Base cluster you want to use for the migration on the **Clusters** page.

2. Click Start Scanning to open the **Scan Settings** where you can select the data and workloads for scanning.

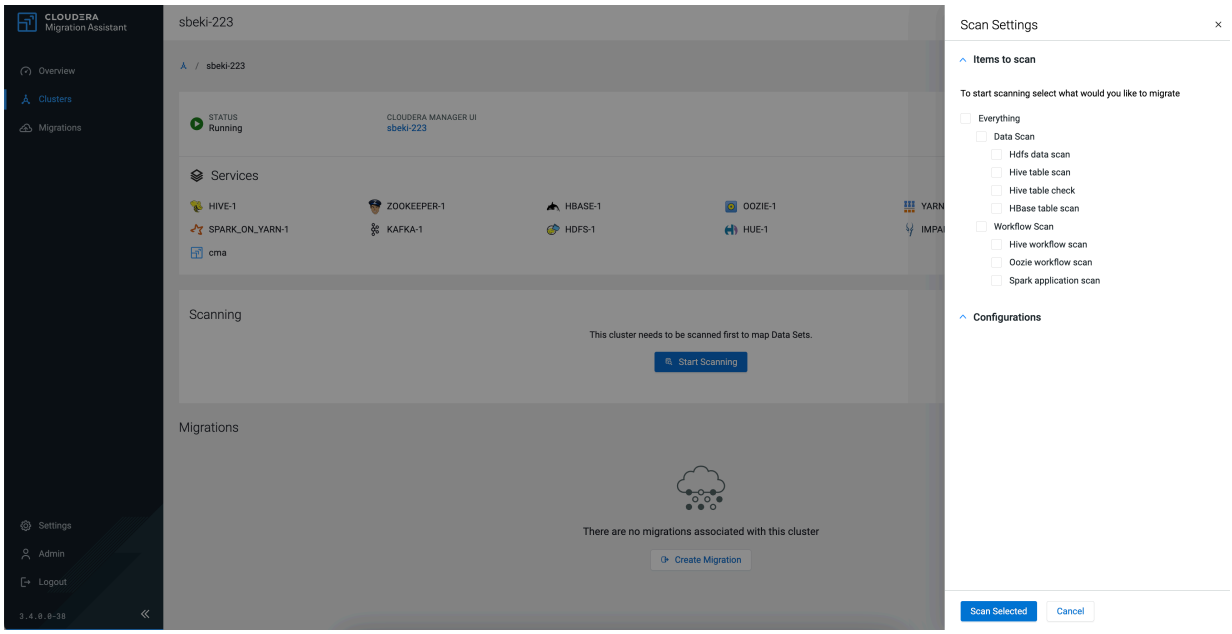

3. Select Everything or choose from the different scanning options.

   The following items are available for scanning:

   HDFS data scan

   The HDFS data scan uses _hdfs_report_ module from the *CDH Discovery Tool* to scan HDFS on the source cluster.

   Hive table scan

   The Hive table scan uses _hive_metastore_ module from the *CDH Discovery Tool* to scan Hive on the source cluster.

   Hive table check

   Scanning Hive tables on the source cluster. _Hive Table Check_ embeds sre and u3 sub-programs of the *Hive SRE Tooling*. The result will be visible at the SRE column of the Hive datasets.

   HBase table scan

   Scanning HBase tables on the source cluster.

   Hive workflow scan

   Scanning Hive SQL queries on the source cluster. You can pre-scan Hive2 SQL queries against Hive3 with the Hive Workflow scan option. When selecting this Hive Workflow option, you need to provide the location of your queries as shown in the following example:

   - HDFS paths
     - With default namespace: hdfs:///dir/, hdfs:///dir/file
     - With specified namespace: hdfs://namespace1/dir, hdfs://namespace1/dir/file
     - With namenode address: hdfs://nameNodeHost:port:/dir, hdfs://nameNodeHost:port:/dir/file
   - Native file paths
     - your/local/dir
     - nodeFQDN:/your/local/dir/sqlFile

   Oozie workflow scan

   Scanning Oozie workloads on the source cluster. If you selected Oozie workflow scan, you need to provide the Number of latest days to scan.

   Spark application scan

   - Scanning Spark applications on the source cluster. If you selected Spark application scan, you need to provide the Number of latest days to scan.
   - Spark History Server needs to be configured to be accessible by the spark user, to do that you can add spark to the list in spark.history.ui.admin.acls at the SPARK_ON_YARN service in Cloudera Manager.
   - Spark jobs also need to be visible by the the user spark.

4. Click Scan selected.

   You will be redirected to the scanning progress where you can monitor if the selected items are successfully scanned or encountered an error.

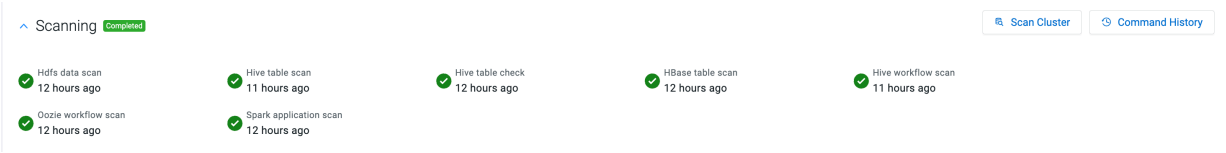

   a) Click Scan Cluster to open the **Scan Settings** again to add more items to the scan or trigger a rescan of the already scanned items.

   b) Click Command Historz to open the **Source command history** to have more insight about the scanning progress, stop an in progress scan and review the log.

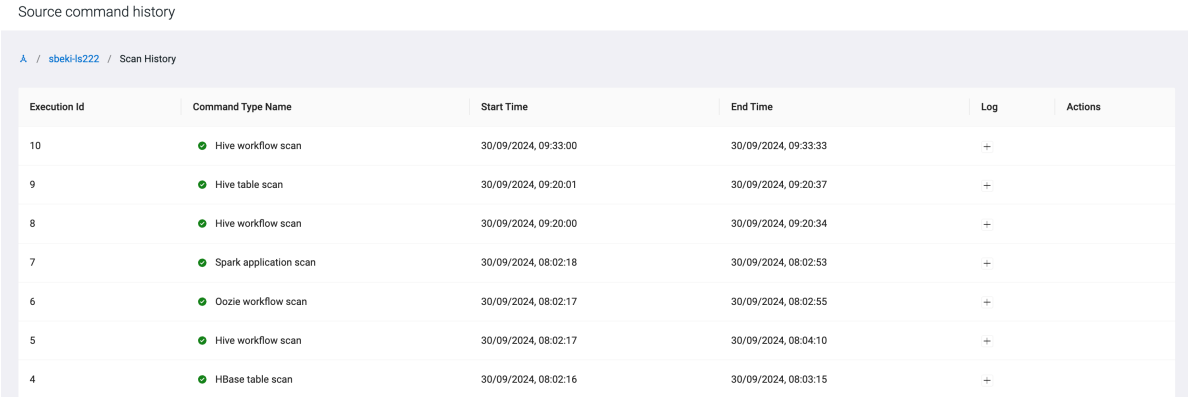

Execution Id	Command Type Name	Start Time	End Time	Log	Actions
10	Hive workflow scan	30/09/2024, 09:33:00	30/09/2024, 09:33:33		
9	Hive table scan	30/09/2024, 09:20:01	30/09/2024, 09:20:37		
8	Hive workflow scan	30/09/2024, 09:20:00	30/09/2024, 09:20:34		
7	Spark application scan	30/09/2024, 08:02:18	30/09/2024, 08:02:53		
6	Oozie workflow scan	30/09/2024, 08:02:17	30/09/2024, 08:02:55		
5	Hive workflow scan	30/09/2024, 08:02:17	30/09/2024, 08:04:10		
4	HBase table scan	30/09/2024, 08:02:16	30/09/2024, 08:03:15		

   Note: The scan results are also available at the data directory of the source cluster, which is located in the /<CMA_ROOT_DIR>/data/sources/ClusterName/Source_ID/cluster-scan/ folder on the Cloudera Migration Assistant server node.

5. Click Start Mapping to review the data, workflows and applications on the source cluster and map their configuration to the destination cluster.

   For example, when reviewing **Hive SQL**, you can check and edit any SQL query related errors before migrating the workflows to Cloudera Public Cloud. The migration will be successful regardless of fixing the statement

errors. However, you will not be able to execute the SQL queries on the new cluster due to the compatibility issues between Hive2 and Hive3. You can review the list of errors using , and open the editor using .

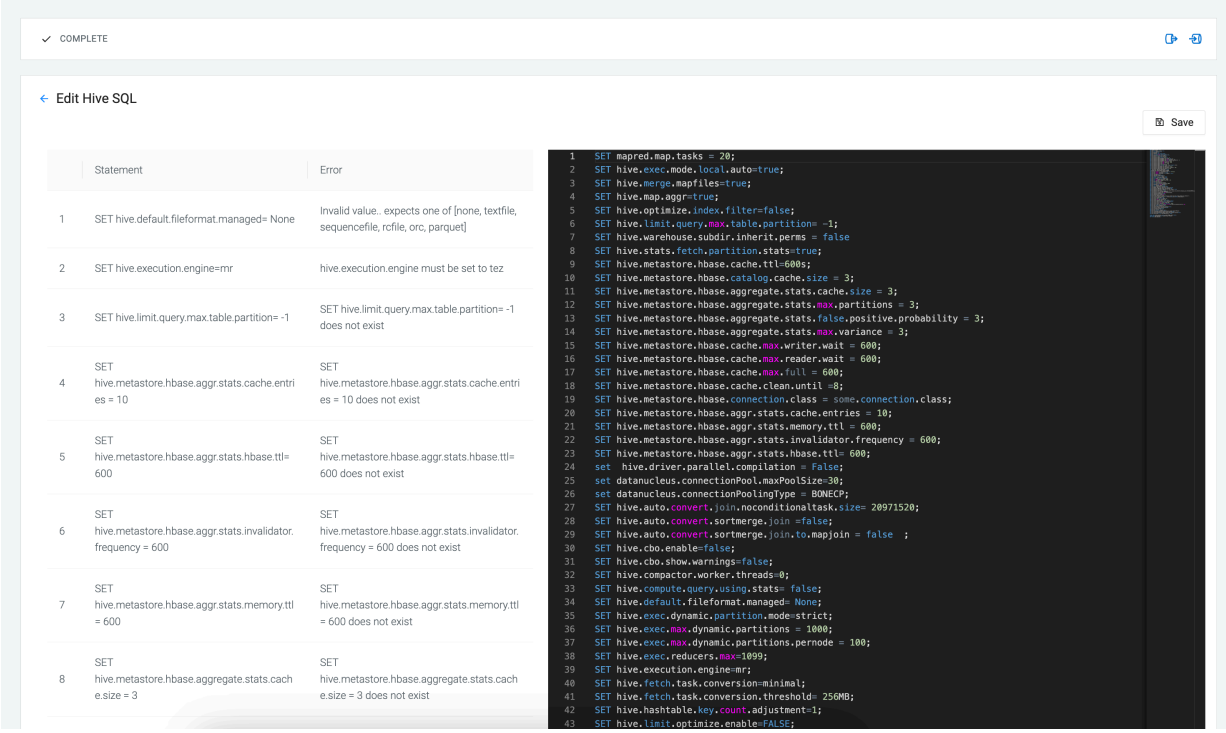

After fixing the statement errors in the SQL editor window, Save the changes. The edited queries are replicated and saved in the S3 bucket of the destination cluster. The original files are not overwritten.

After the scanning is completed, you can add the tables and workflows from the selected services to collections. **Collections** serve as an organizational method to sort out the data and workflows resulted from the scan for migration.

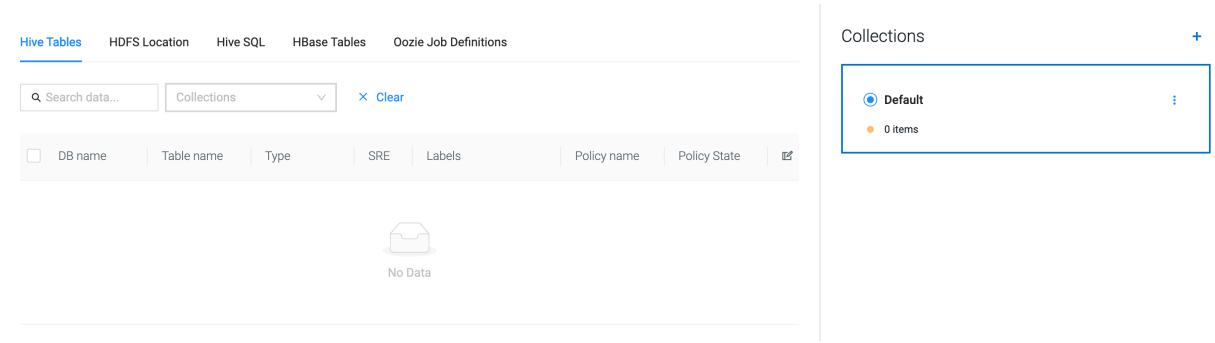

Results

The datasets and workflow on the CDH or Cloudera Private Cloud Base source cluster is scanned for Hive, HDFS, HBase, Oozie and Spark.

What to do next

Sort the scanned data and workflows into collections to have more control over what is migrated from the source cluster to the target cluster.

Related Information

CDH Discovery Tool

Hive SRE Tooling

Creating collections for migration

After scanning the source cluster, you can use collections to sort the datasets that need to be migrated to the destination cluster. Collections can also be useful to organize different types of data, workflows and applications before migration.

Procedure

1. Click Start Mapping or Collections.
2. Click Hive Tables, HDFS Location, Hive SQL, HBase Tables, Oozie Job Definitions or Spark Applications based on which items you would like to add to a collections.

   The different windows show the results of the scanning. For example, the Hive Tables display all the existing tables in Hive on the source cluster.

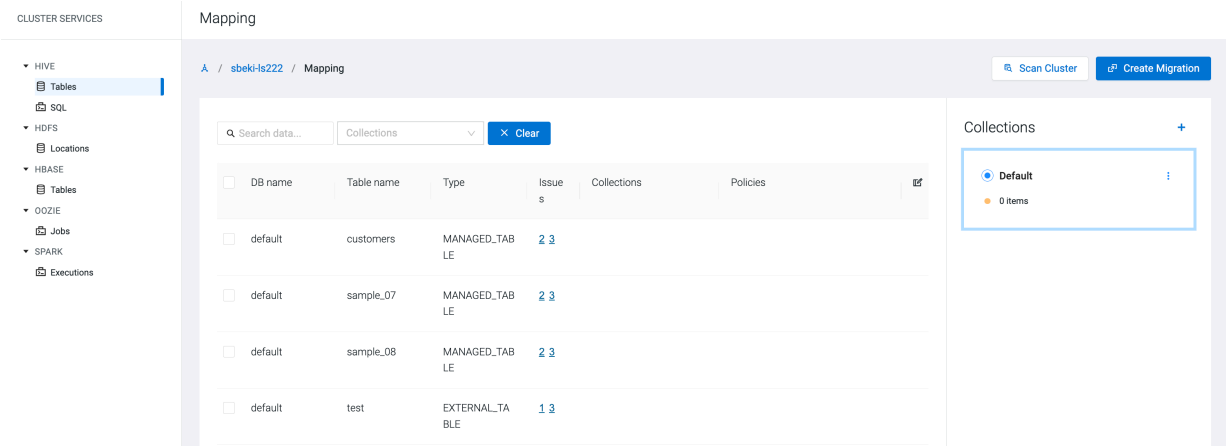

3. Select the items that you want to add to a collection, and click Add to collection.

   The selected items are added to the **Default** collection, and the **Default** label is assigned to the selected items.

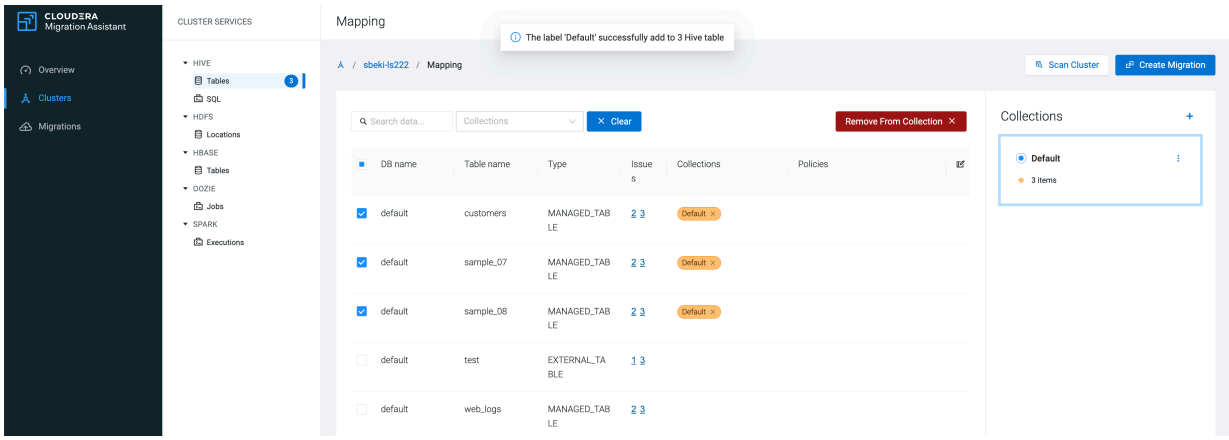

You have the option to create more collections beside the **Default** collection.

4. Click + next to **Collections**, and customize the collection by providing a Label name and selecting a Color. Click Create.

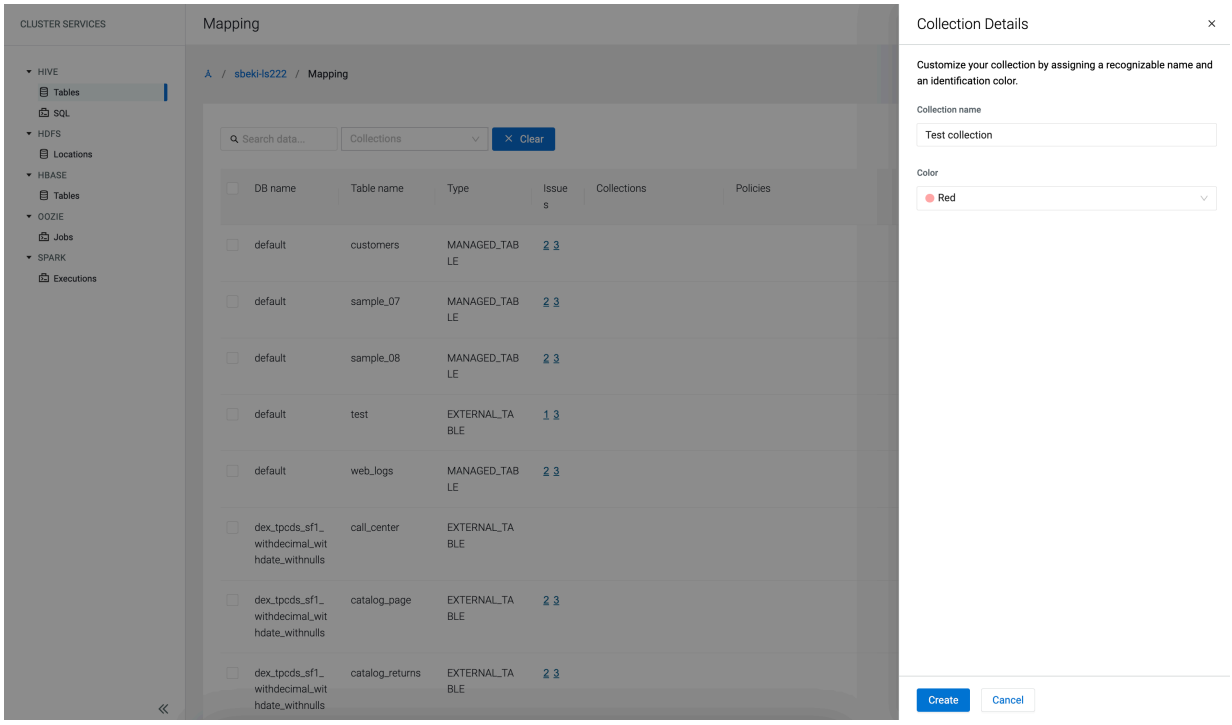

Select more items from the scanning results, and add it to the newly created collection.

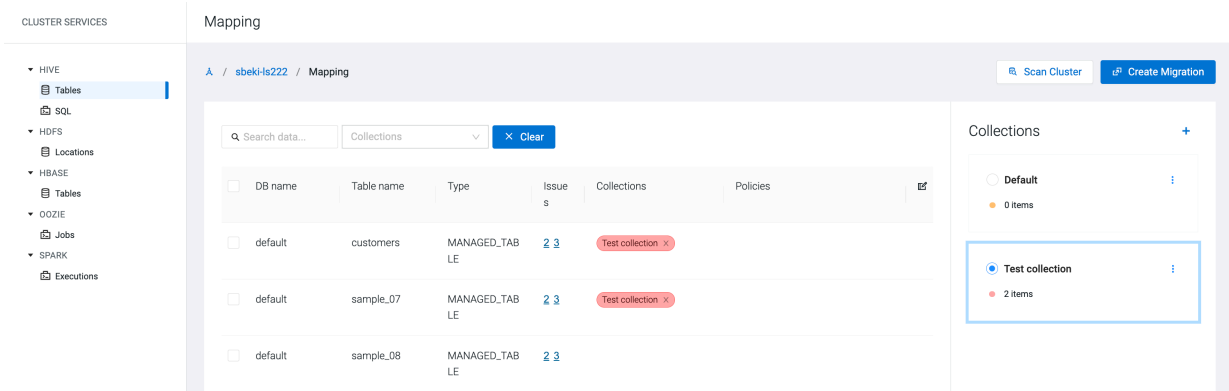

You can manage the created collections by clicking on ⋮ to open the collections menu. You can review the labeled results in a collections using View Items, and modify or delete the created collection using Edit and Remove. Removing a collection does not affect the items on the source cluster, only the labelling is deleted.

Results

The datasets are labeled for migration.

What to do next

Register the destination cluster to which the datasets and workflows are going to be migrated.

Registering destination clusters

As you are migrating from CDH or Cloudera Private Cloud Base to Cloudera Public Cloud, you need to register the Cloudera Public Cloud cluster as a destination to which the data, metadata, and workload will be migrated.

Before you begin

Note down the following information about the Cloudera Public Cloud cluster to complete the registration:

- Access key and private key

  For more information about how to generate access and private key, see the Generating an API access key documentation.
- SSH user, port, and private key

Procedure

1. Click See More Register A New Destination on the homepage of Cloudera Migration Assistant to register a cluster that will be used as a source for the migration.

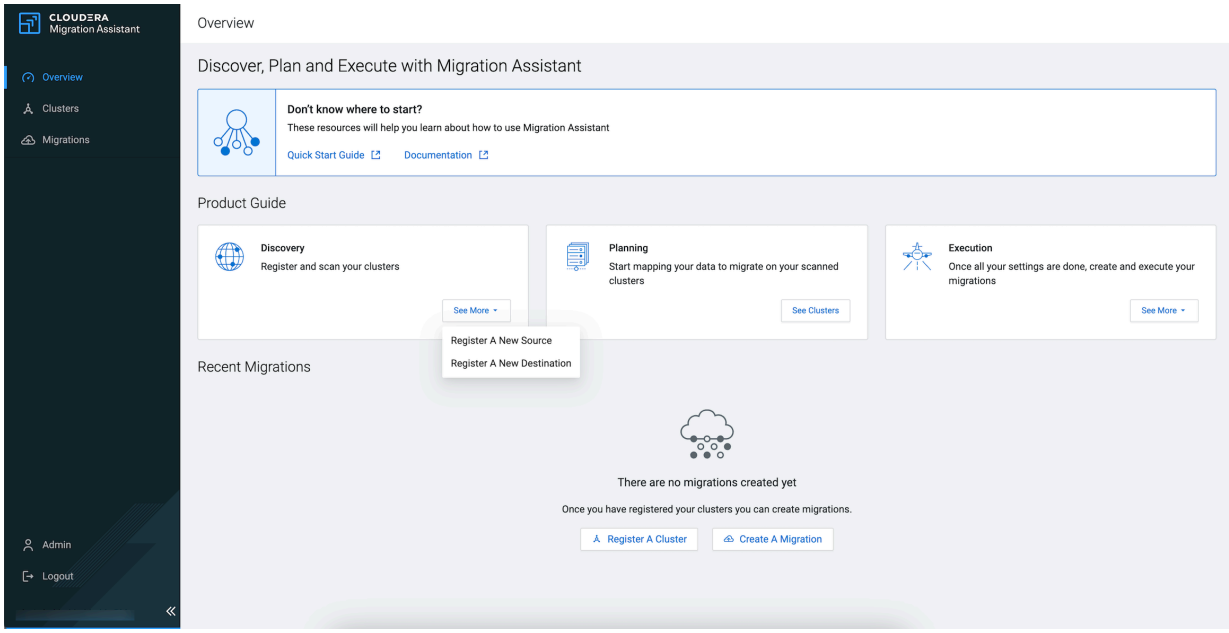

   Alternatively, you can open the New Cluster wizard by selecting Clusters on the left navigation pane, and clicking New Destination .
2. Select Cloudera Public Cloud as **Target Type**.
3. Select the Control Plane URL where the cluster is located.
4. Provide the Access Key and Private Key of your Cloudera user account.
5. Click Next.
6. Choose the cluster based on the Cluster Name that you want to use for the migration.

   The drop-down list contains all of the existing clusters in Cloudera Public Cloud that you have access to.
7. Click Next.
8. Select the Configuration Preference based on which authentication method you prefer.
   - Choose Use existing if you want to use the SSH configuration and keys of the user running Cloudera Migration Assistant server to access the hosts.
   - Choose New if you want to use a newly provided SSH key to configure Ansible automatically.

     a. Provide the SSH User and SSH Port.

     b. Copy the SSH Key to the SSH Key box or upload a .pem file containing the key.

     c. If you want to use Password, provide the password and make sure the sshpass utility is installed.

     Note: The user specified during SSH configuration must have **passwordless sudo rights**.

9. Provide the following information based on the cloud provider of your cluster:
   - S3: S3 Bucket Access Key and S3 Bucket Secret Key
   - ABFS: Client Id, Client Secret Key and Tenant Id
10. Click Create.

Results

The registered Cloudera Public Cloud cluster is listed on the **Clusters** page.

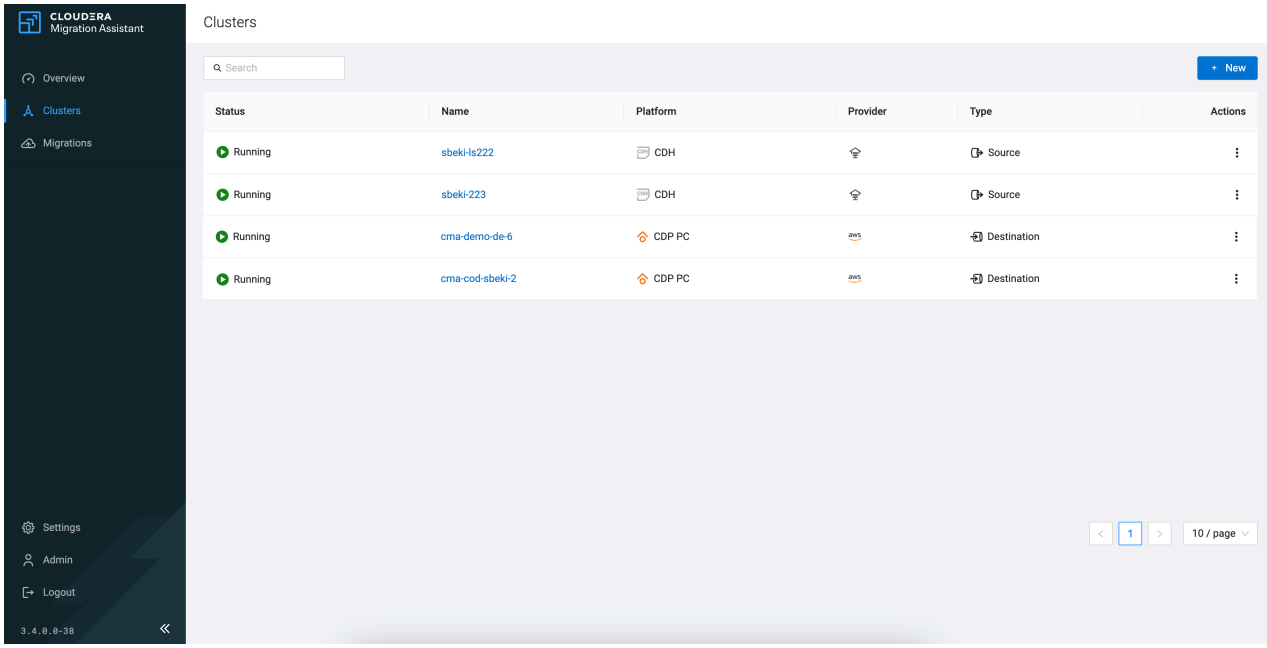

You can review the details and services of the cluster by clicking on the Name of the cluster.

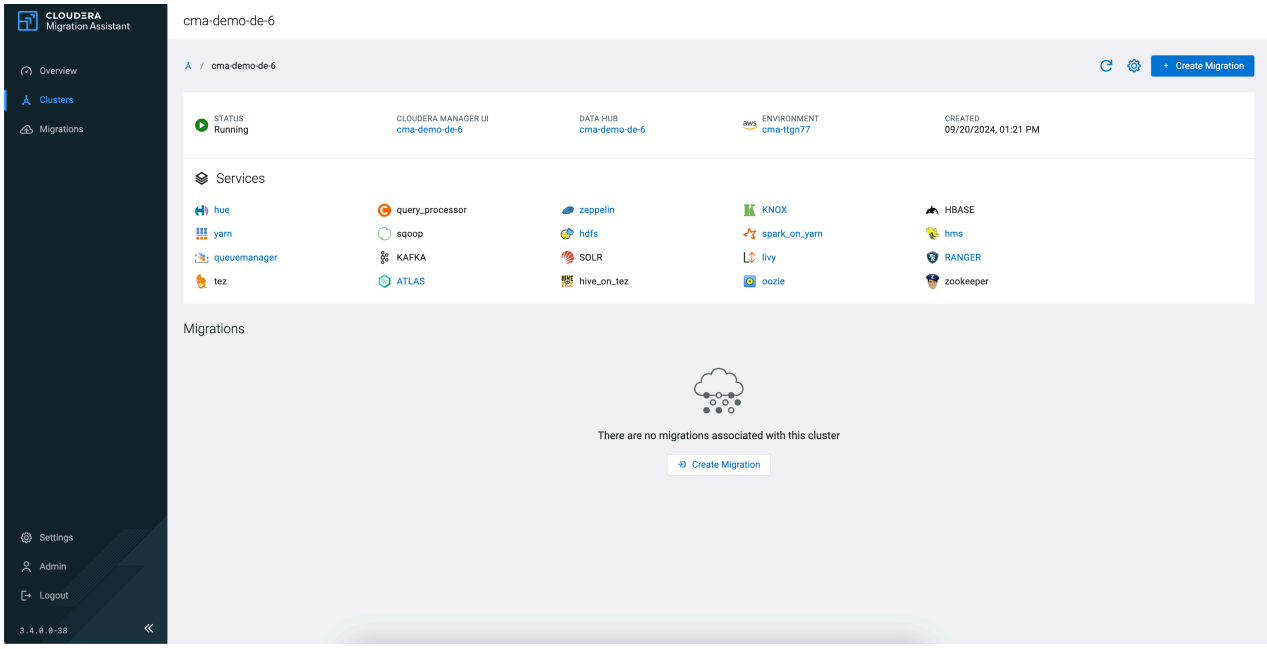

What to do next

Start the migration from CDH or Cloudera Private Cloud Base to Cloudera Public Cloud.

Migrating from source cluster to destination cluster

After registering the source and destination cluster, and labeling the scanned datasets, workloads and applications on the source cluster, you can start the migration process.

About this task

Because migrating data to S3 can take a long time, you can perform multiple migrations between a source and destination cluster to move the data in stages. You can also choose to migrate only part of your data as opposed to all of it. A single Cloudera Migration Assistant server is designed to handle multiple migrations.

Before you begin

Procedure

1. Click Migrations on the left navigation pane.
2. Click Start Your First Migration.
3. Select Cloudera Distributed Hadoop 5, Cloudera Distributed Hadoop 6 or Cloudera Private Cloud Base as **Source Type**.

   The registered source cluster is selected by default. You can select any other cluster using the drop-down menu . In case you have not registered a source cluster at this point, click New Source and complete the steps in Registering the source cluster.
4. Click Next.

   Cloudera Public Cloud and the registered destination cluster are selected by default. You can select any other cluster using the drop-down menu. In case you have not registered a source cluster at this point, click New Target and complete the steps in Registering the destination cluster.
5. Click Next.
6. Click Next to confirm the migration path.
7. Select one or more labels for migration migrate to the destination cluster.

   You can select if the migration should Run Now or be completed in a Scheduled Run. Run Now means that all of the datasets and workloads that were selected with the labels are going to be migrated as soon as the process starts. When choosing the Scheduled Run, you can select the start date of the migration, and set a frequency in which the migration process should proceed.

8. Configure the Migration by providing the required information.

   The Cloudera workload user and password will be used to access Cloudera Manager of the Cloudera Data Hub cluster in Cloudera Public Cloud. You also must set the S3 Bucket Base Path for HDFS or Cloud Storage Path when migrating HDFS data. Some migration require SSH login credentials with sudo-less password.

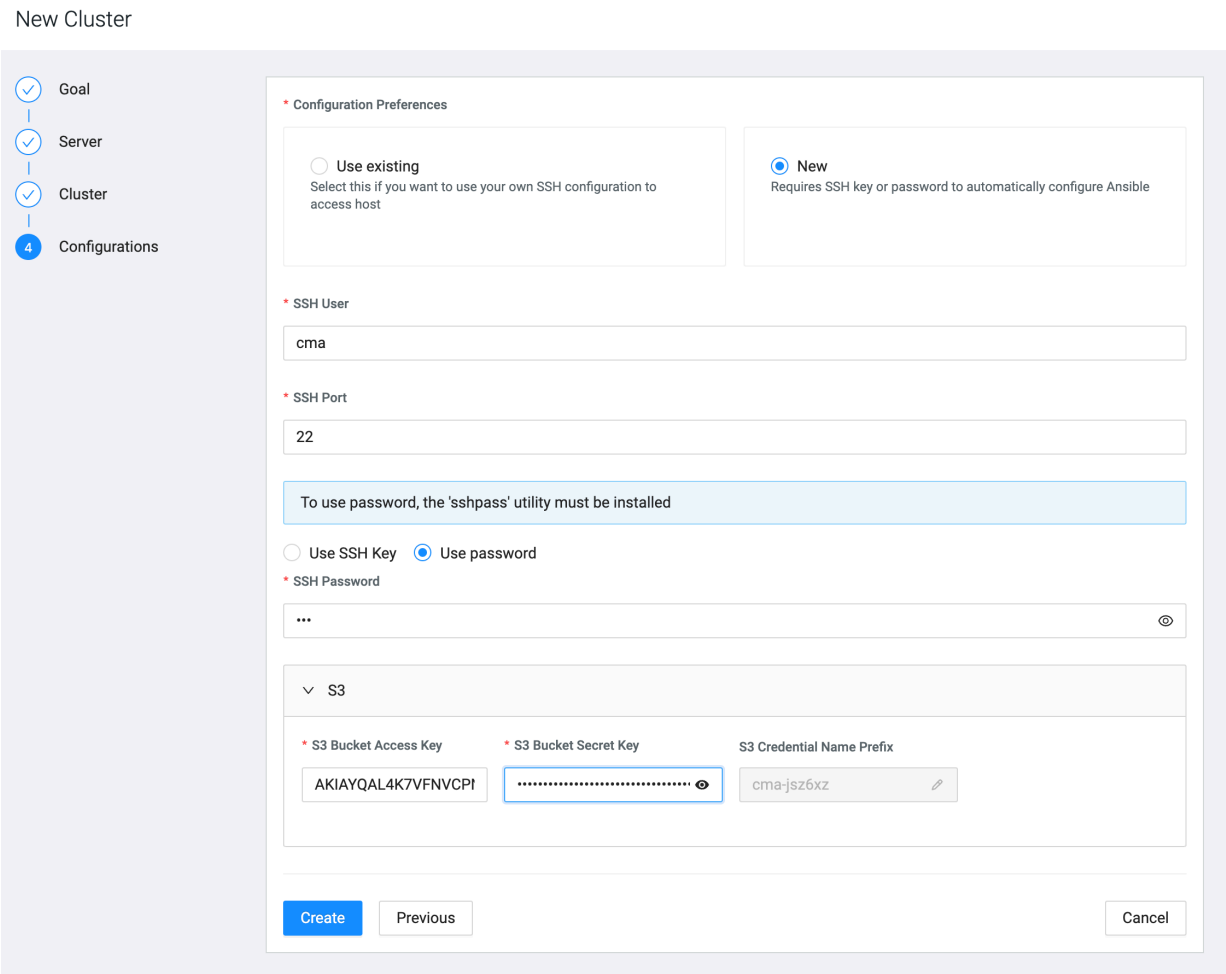

   The remaining service specific settings on the **Configurations** page are automatically filled out, but can be changed based on your requirements.

9. Click Next.

10. Review the information on the Overview page and ensure that the information is correct.

    At this point, you can go back and change any configuration if the information is not correct.

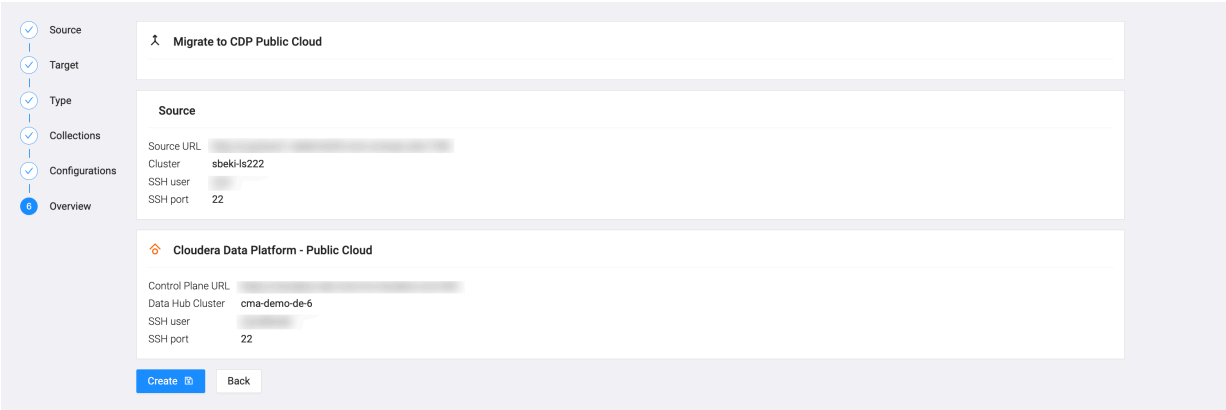

11. Click Create to save the migration plan.. You can follow the progress of creating the migration plan.
12. Click Go to Migrations, and select the created CDH to Cloudera Public Cloud or Cloudera Private Cloud Base to Cloudera Public Cloud migration.

13. Click Execute Migration to start the migration on the **Migration Dashboard**.

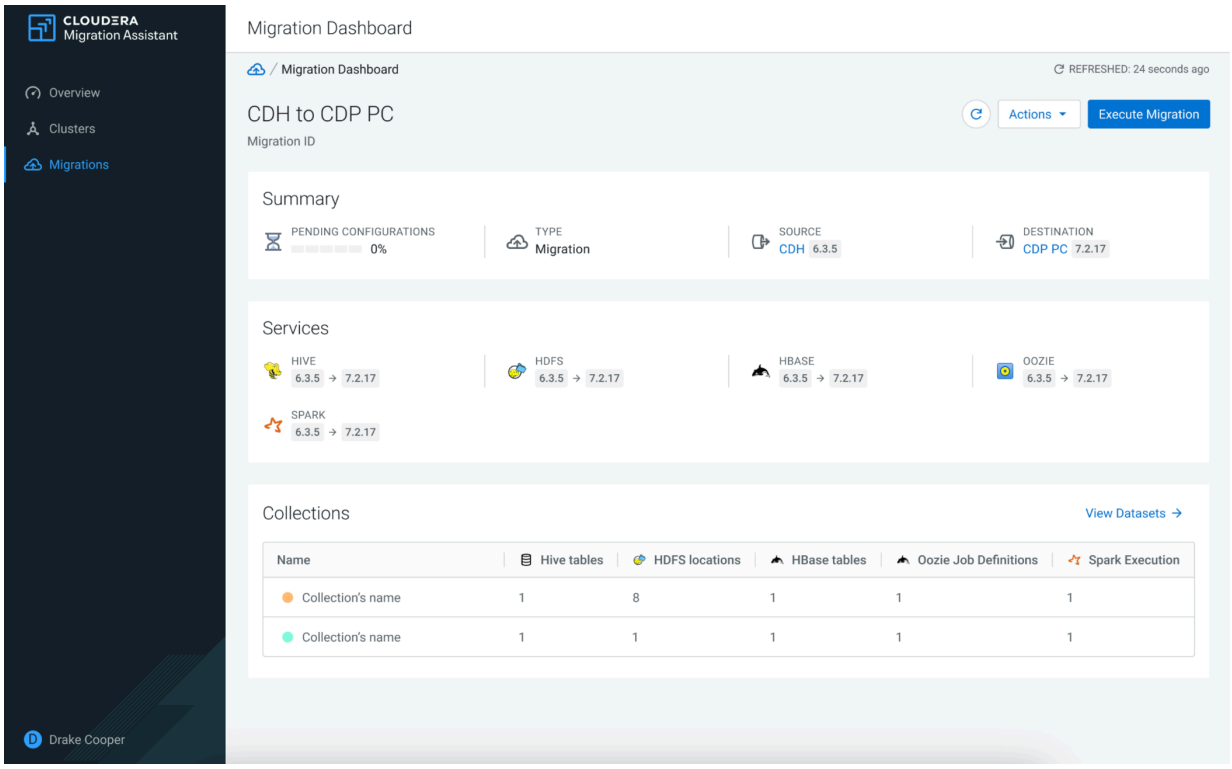

Before you can start the migration, you can update the Cluster, S3, HDFS and YARN specific settings on the **Configuration** tab.

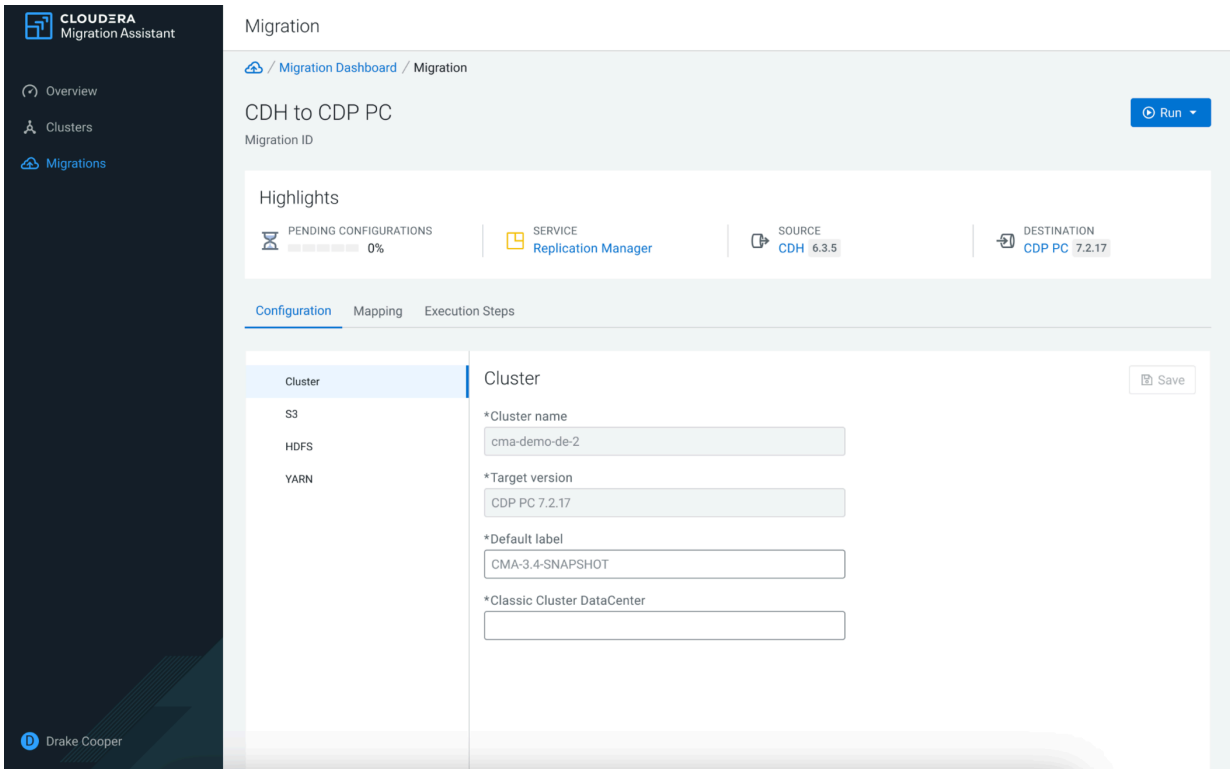

Under Mapping, you can ensure that the datasets and applications from the source cluster are in a compatible format on the target cluster.

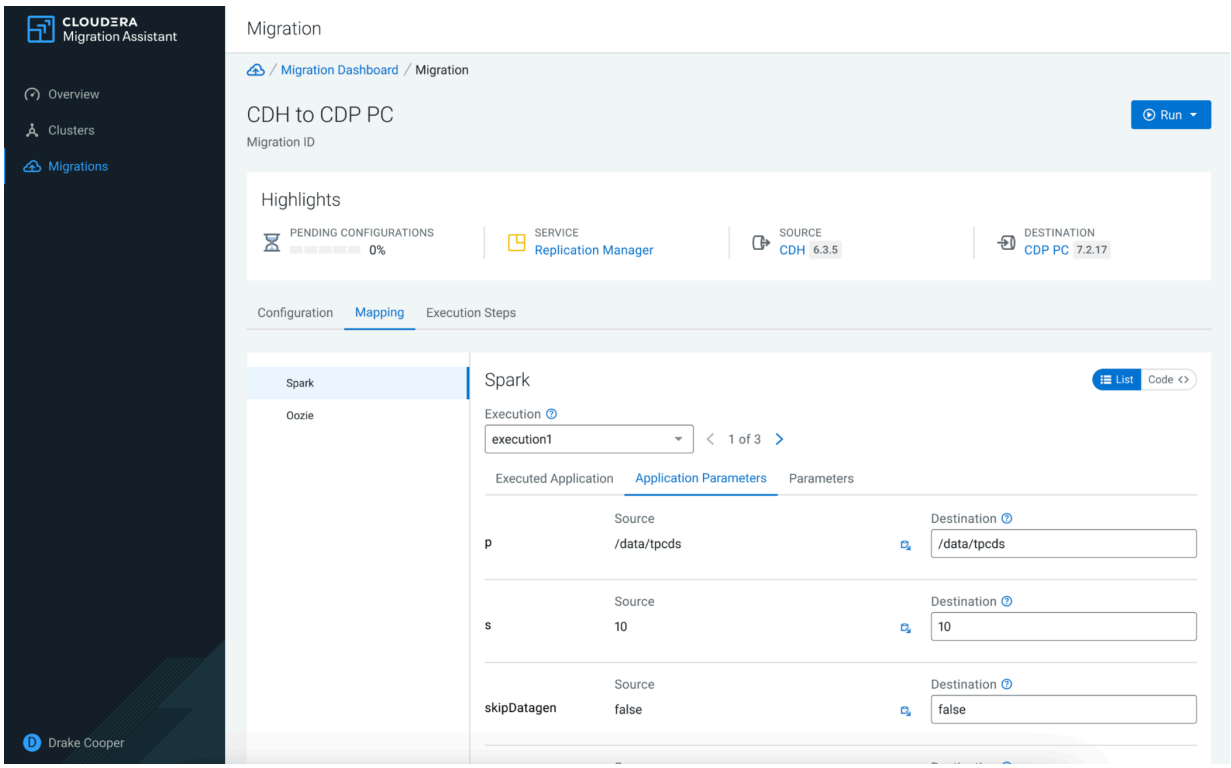

After finishing the configuration and mapping, click Run to start either the whole migration or specific phases of the migration. You can track the process of each migration step using the **Execution Steps** tab.

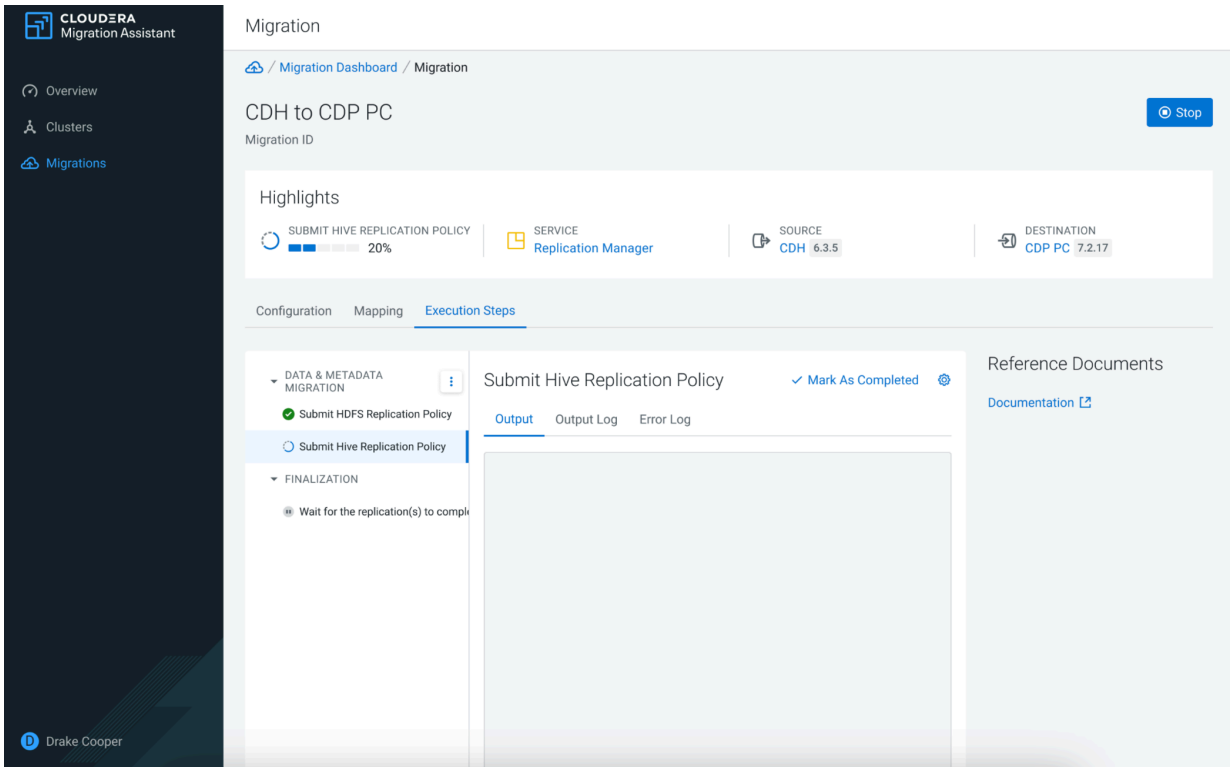

The **Data & Metadata Migration** executes the data migration of the labeled datasets with Replication Manager.

You can also view the migration process of the data and workloads based on the selected services. For example, the **Hive SQL Migration** replicates the Hive SQL queries that were fixed to be Hive complied during the Hive Workload migraton steps.

The **Finalization** waits until all the Cloudera Replication Manager policies complete their jobs. If the label is created as a frequently scheduled migration, the Cloudera Replication Manager waits only for the first jobs.

When migrating from Cloudera Private Cloud Base to Cloudera Public Cloud, you need to manually export and import the Ranger policies from the source cluster to the destination cluster using the following curl commands:

- Exporting policies
  - To export all policies:

```
curl -X GET --header "text/json" -H "Content-Type: text/
json" -o file.json -u [***USERNAME***]:[***PASSWORD***] "ht
tp://[***HOSTNAME***]:[***RANGER PORT***]/service/plugins/policies/ex
portJson"
```

  - To export for specific HDFS resource:

```
curl -X GET --header "text/json" -H "Content-Type: text/
json" -o file.json -u [***USERNAME***]:[***PASSWORD***] "ht
tp://[***HOSTNAME***]:[***RANGER PORT***]/service/plugins/policies/ex
portJson?resource%3Apath=[***PATH NAME***]"
```

  - To export for policies for specific resource such as Hive database and Hive column:

```
curl -X GET --header "text/json" -H "Content-Type: text/
json" -o file.json -u [***USERNAME***]:[***PASSWORD***] "ht
tp://[***HOSTNAME***]:[***RANGER PORT***]/service/plugins/policies/ex
portJson??resource%3Adatabase=[***DATABASE NAME***]&resource%3Acolu
mn=[***COLUMN NAME***]"
```

- Importing policies
  - To Import policies from JSON file without servicesMap:

```
curl -i -X POST -H "Content-Type: multipart/form-data" -F 'f
ile=@/path/file.json' -u [***USERNAME***]:[***PASSWORD***] htt
p://[***HOSTNAME***]:[***RANGER PORT***]/service/plugins/policies/im
portPoliciesFromFile?isOverride=true
```

  - To Import policies from JSON file with servicesMap:

```
curl -i -X POST -H "Content-Type: multipart/form-data" -
F 'file=@/path/file.json' -F ‘servicesMapJson=@/path/ser
vicesMapping.json’ -u [***USERNAME***]:[***PASSWORD***] htt
p://[***HOSTNAME***]:[***RANGER PORT***]/service/plugins/policies/im
portPoliciesFromFile?isOverride=true
```

Results

The datasets and workloads selected are migrated from CDH or Cloudera Private Cloud Base to Cloudera Public Cloud.

Migrating Spark applications

During the Spark workflow migration, the job JAR file, job properties and other Spark job related data are migrated from a CDH or Cloudera Private Cloud Base cluster to a Cloudera Data Hub cluster.

About this task

Before the migration, the source cluster is scanned to collect the Spark application JAR files. During the migration process, the Spark applications are not affected on the source cluster and can remain in running state. The source code of the Spark applications also remain the same, only the JAR files are copied from the Source cluster to the Destination cluster. To migrate the files that have dependency to the Spark application, you need to ensure to select them during the migration creation. When the migration is finished, the job definitions are stored in the S3 bucket and the application properties are stored in the local filesystem.

Note: The following limitation apply to the Spark migration:

- **No** refactoring of the Spark application's source code for newer Spark versions
- JAR and Python package dependencies are not migrated automatically
- No R is supported
- Dependency discovery relies on spark-submit parameters **hdfs locations** only

Before you begin

- Ensure that Cloudera Migration Assistant is set up correctly using the steps in Setting up Cloudera Migration Assistant server.
- Ensure that you have met the requirements detailed in Reviewing prerequisites before migration.
- Ensure that you have a CDH 5, CDH 6 or Cloudera Private Cloud Base cluster registered as a source from which you want to migrate your Spark applications. If you do not have a source cluster yet, complete the steps in Registering source clusters.
- Ensure that you have a Cloudera Data Hub cluster registered as a destination cluster to which you want to migrate your Spark applications. If you do not have a destination cluster yet, complete the steps in Registering destination clusters.

Procedure

1. Click on the CDH or Cloudera Private Cloud Base cluster you want to use for the migration on the **Clusters** page.
2. Click Start Scanning to open the **Scan Settings**.
3. Select Spark application scan.

   Important: In case you want migrate the Spark application dependencies, such as HDFS files, you need to make sure to select the dependent services during the Source cluster scanning, and add the dependent files and configurations to the same Collection as the Spark applications.

   a) Provide the Number of latest days to scan to define the period from which the Spark applications are collected.
   b) Click Scan selected.

      You will be redirected to the scanning progress, where you can monitor if the scanning process was successful or encountered any error.
4. Click on Start Mapping to view the collected job definitions when the scan is finished.
5. Add Spark workloads to Collections.

   Collections serve as an organization method to sort and bundle the job definitions into groups for the migration. You can create more collections beside the Default collection based on your requirements.

   After you are finished with sorting the Spark workloads and their dependencies to collections, you can start the migration process by creating the migration plan.
6. Click Create Migration or select Migrations Start Your First Migration .
   a) Select the source cluster, and click Next.
   b) Select the destination cluster, and click Next.
   c) Select the type of migration, and click Next.
   d) Select the collections that you want to migrate, and click Next.

      You can select if the migration should Run Now or be completed in a Scheduled Run. Run Now means that the Oozie job definitions in the selected collections are going to be migrated as soon as the process starts. When

choosing the Scheduled Run, you can select the start date of the migration, and set a frequency in which the migration process should proceed.

e) Provide he Cloudera workload user and password will be used to access Cloudera Manager of the Cloudera Data Hub cluster in Cloudera Public Cloud.

f) Ensure that the path of the Folder for the Spark scripts on the target Data Lake is correct.

g) Click Next.

An overview of the migration plan is displayed. At this point, you can go back and change any configuration if the information is not correct. If the information is correct, click Create.

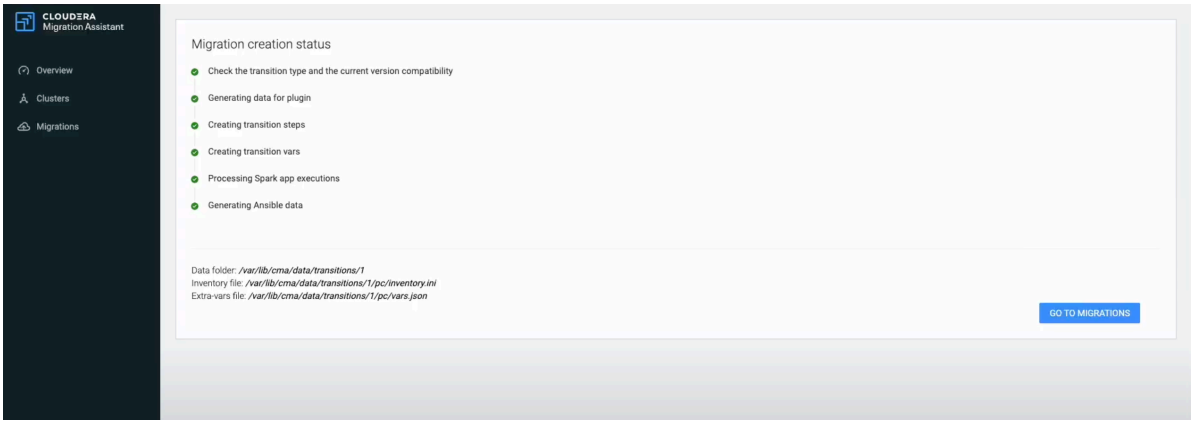

7. Click Execute Migrations when the migration plan is successfully created.

You can use the **Mapping** tab to update the Spark workloads, and you can view the steps that will be completed during the migration on the **Execution Step** tab.

Under **Configuration**, you can view how the configuration are being mapped from the Source cluster to the Destination cluster. The configurable parameters on the destination cluster are filled out automatically based on how the Destination cluster was configured for the migration, but these parameters can be changed based on your requirements before executing the migration.

8. Click Run to start migration.

You also have the option to click ⋮ and select Run All. In this case the migration steps are executed manually. Choosing Run All in Current Phase enables you to manually start the next phase of the migration.

During the Spark migration, the HDFS replication policies are created if there are any HDFS file dependencies for the Spark application. In the next step, the Spark shell scripts are created on the selected cloud storage. During the finalization, there is a manual step to review if the Spark application migration was successful to the Destination cluster. You can choose how to ensure that the Spark applications are on the Destination cluster, but you can use the CLI commands provided on the screen.

What to do next

When all of the steps are successfully completed, the migration of Spark applications from CDH or Cloudera Private Cloud Base to Cloudera Public Cloud is finished. You can restart the Spark applications on the destination Cloudera Data Hub cluster using Command Line Interface (CLI) or via Livy as instructed in the last step.

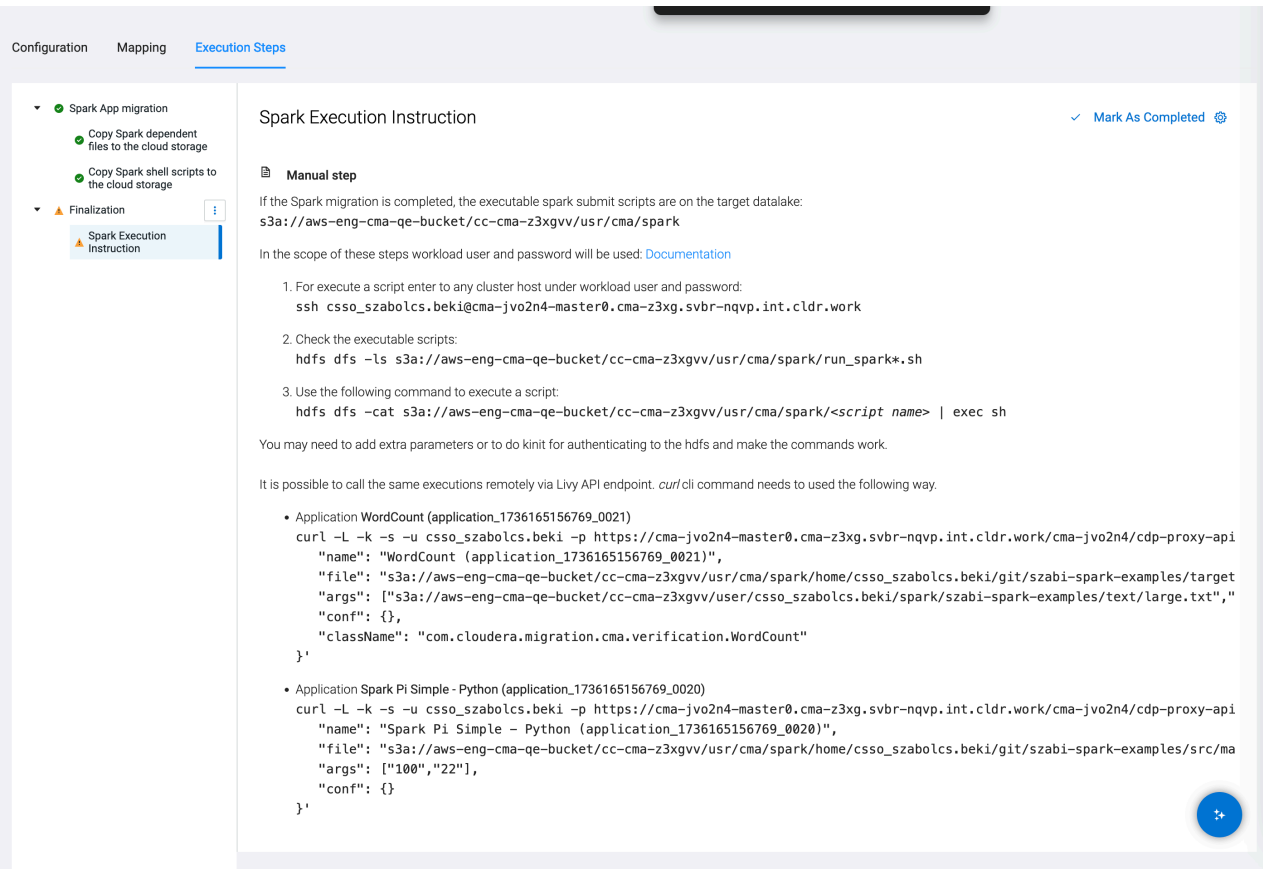

Migrating Oozie workflows

During the Oozie workflow migration, the job definitions, job properties and other Oozie job related data are migrated from a CDH or Cloudera Private Cloud Base cluster to a Cloudera Data Hub cluster.

About this task

Before the migration, the source cluster is scanned to collect the workflows, coordinators, bundles and discover the relations between them. You also have the option to parse the Hive SQL files to obtain the related databases and tables names. During the migration process, the Oozie jobs are not affected on the source cluster and can remain in running state. When the migration is finished, the job definitions are stored in the S3 bucket and the job properties are stored in the local filesystem.

Before you begin

- Ensure that Cloudera Migration Assistant is set up correctly using the steps in Setting up Cloudera Migration Assistant server.
- Ensure that you have met the requirements detailed in Reviewing prerequisites before migration.
- Ensure that you have a CDH 5, CDH 6 or Cloudera Private Cloud Base cluster registered as a source from which you want to migrate your Oozie workflows. If you do not have a source cluster yet, complete the steps in Registering source clusters.
- Ensure that you have a Cloudera Data Hub cluster registered as a destination cluster to which you want to migrate your Oozie workflows. If you do not have a destination cluster yet, complete the steps in Registering destination clusters.

Procedure

1. Click on the CDH or Cloudera Private Cloud Base cluster you want to use for the migration on the **Clusters** page.
2. Click Start Scanning to open the **Scan Settings**.
3. Select Oozie workflow scan.
   a) Provide the Number of latest days to scan to define the period from which the Oozie jobs are collected.
   b) Click Scan selected.

      You will be redirected to the scanning progress, where you can monitor if the scanning process was successful or encountered any error.
4. Click on Oozie Job Definitions to view the collected job definitions when the scan is finished.

   You have the option to analyze the Hive scripts when you migrate Oozie jobs that depend on Hive SQL files. In this case, Cloudera Migration Assistant scans and identifies the SQL file location stored either in HDFS or other custom directories, and adds the SQL files to the migration plan.

   a. Enable Run Hive3Parser.
   b. Select the Oozie jobs to analyze.
   c. Click .

   After the scan is completed, the Hive scripts related to the selected Oozie jobs are listed under Hive SQL tab.
5. Add the Oozie job definitions to Collections.

   Collections serve as an organization method to sort and bundle the job definitions into groups for the migration. You can create more collections beside the Default collection based on your requirements. The Hive scripts that belong to the Oozie job definitions are automatically added to the same collection.

   After you are finished with sorting the job definitions to collections, you can start the migration process by creating the migration plan.
6. Click Create Migration or select Migrations Start Your First Migration .
   a) Select the source cluster, and click Next.
   b) Select the destination cluster, and click Next.
   c) Select the type of migration, and click Next.
   d) Select the collections that you want to migrate, and click Next.

      You can select if the migration should Run Now or be completed in a Scheduled Run. Run Now means that the Oozie job definitions in the selected collections are going to be migrated as soon as the process starts. When choosing the Scheduled Run, you can select the start date of the migration, and set a frequency in which the migration process should proceed.
   e) Provide he Cloudera workload user and password will be used to access Cloudera Manager of the Cloudera Data Hub cluster in Cloudera Public Cloud.
   f) Enable Oozie service configuring to prepare Oozie service on destination cluster for running jobs to include a service preparation step during the migration process.

      You can set the paths used by Oozie services. These paths are used when configuring the Oozie service for migration.
   g) Click Next.

      An overview of the migration plan is displayed. At this point, you can go back and change any configuration if the information is not correct. If the information is correct, click Create.
7. Click Execute Migrations when the migration plan is successfully created.
8. Review and configure the Oozie job definitions under **Mapping** before starting the migration process.
   a) Select a job definition to list the corresponding **Job properties** and **Workflow**.

      The original and proposed values are filled out based on the source and destination cluster information.
   b) Modify the values of the job definition based on the warnings highlighted in the **Workflow** diff view. You can save the job definition changes using the Save button.

      Cloudera Migration Assistant typically looks for configuration values that are related to service endpoints, Kerberos principals, and so on. These configuration values are used to update the file locations and other

configurations accordingly. While the automatic changes work without any reservation, ensure to review the propositions and update the configurations based on the destination cluster requirements. The following properties and values should be reviewed before the migration:

- HDFS file paths changed to S3 or ABFS
- Hostnames
- Service settings
- Paths to user-related directories

c) Click Save property changes to update the configurations.

You have the option to save the changes for only the edited jobs or apply the changes to all of the jobs.

9. Click Run to start migration.

You also have the option to click ⋮ and select Run All. In this case the migration steps are executed manually. Choosing Run All in Current Phase enables you to manually start the next phase of the migration.

During the Hive SQL migration, the Hive scripts are copied to the Hive S3 bucket on the destination cluster.

When the Hive SQL Migration is finished, click ⊙ to start preparing the Oozie service on the destination cluster for running the jobs that are stored in S3. When the service preparation is finished, click ⊙ to start uploading the job definitions and configurations to the local file system and S3 bucket.

What to do next

When all of the steps are successfully completed, the migration of Oozie job definitions from CDH or Cloudera Private Cloud Base to Cloudera Public Cloud is finished. You can restart the Oozie jobs on the destination Cloudera Data Hub cluster using Command Line Interface (CLI) or Hue.

Migrating SQL queries

During the Hive migration beside the SQL query, the query related tables and data are also migrated from a CDH or Cloudera Private Cloud Base cluster to a Cloudera Data Hub cluster.

About this task

Before the migration, the source cluster is scanned to collect the SQL queries, tables and data from Hive or Impala. This migration can be used in cases when there is a heavy SQL query load and you want to unload the less time sensitive queries to another cluster. Using the scheduling feature of the underlying Cloudera Replication Manager, you can keep the queries in sync between the source and destination cluster. During the migration process, the SQL queries are not affected on the source cluster and can remain in running state.

Before you begin

- Ensure that Cloudera Migration Assistant is set up correctly using the steps in Setting up Cloudera Migration Assistant server.
- Ensure that you have met the requirements detailed in Reviewing prerequisites before migration.
- Ensure that you have a CDH 5, CDH 6 or Cloudera Private Cloud Base cluster registered as a source from which you want to migrate your Hive queries. If you do not have a source cluster yet, complete the steps in Registering source clusters.
- Ensure that you have a Cloudera Data Engineering Data Hub cluster registered as a destination cluster to which you want to migrate your Hive queries. If you do not have a destination cluster yet, complete the steps in Registering destination clusters.

Procedure

1. Click on the CDH or Cloudera Private Cloud Base cluster you want to use for the migration on the **Clusters** page.

2. Click Start Scanning to open the **Scan Settings**.

3. Select Hive table scan, Hive table check and Hive workflow scan.
   a) Provide the Hive query parser input.

      You can pre-scan Hive2 SQL queries against Hive3 with the Hive Workflow scan option. When selecting this Hive Workflow option, you need to provide the location of your queries as shown in the following example:

      - HDFS paths
        - With default namespace: hdfs:///dir/, hdfs:///dir/file
        - With specified namespace: hdfs://namespace1/dir, hdfs://namespace1/dir/file
        - With namenode address: hdfs://nameNodeHost:port:/dir, hdfs://nameNodeHost:port:/dir/file
      - Native file paths
        - your/local/dir
        - nodeFQDN:/your/local/dir/sqlFile
   b) Click Scan selected.

      You will be redirected to the scanning progress, where you can monitor if the scanning process was successful or encountered any error.
4. Click on Hive SQL to view the collected queries when the scan is finished.

   You can also find the tables that are related to the queries under Hive tables.
5. Add the Hive queries to Collections.

   Collections serve as an organization method to sort and bundle the queries into groups for the migration. You can create more collections beside the Default collection based on your requirements. The Hive tables that belong to the Hive queries are automatically added to the same collection.

   After you are finished with sorting the queries to collections, you can start the migration process by creating the migration plan.
6. Click Create Migration or select Migrations Start Your First Migration .
   a) Select the source cluster, and click Next.
   b) Select the destination cluster, and click Next.
   c) Select the type of migration, and click Next.
   d) Select the collections that you want to migrate, and click Next.

      You can select if the migration should Run Now or be completed in a Scheduled Run. Run Now means that the Hive queries in the selected collections are going to be migrated as soon as the process starts. When choosing the Scheduled Run, you can select the start date of the migration, and set a frequency in which the migration process should proceed. In case your goal is to keep the queries in sync between the source and destination cluster, select the Scheduled Run with a frequent time period for migration.
   e) Review the default configurations that are filled out automatically.
   f) Click Next.

      An overview of the migration plan is displayed. At this point, you can go back and change any configuration if the information is not correct. If the information is correct, click Create.
7. Click Execute Migrations when the migration plan is successfully created.

   You can view the steps that will be completed during the migration on the **Execution Step** tab.
8. Click Run to start migration.

   You also have the option to click ⋮ and select Run All. In this case the migration steps are executed manually. Choosing Run All in Current Phase enables you to manually start the next phase of the migration.

   During the Hive SQL migration, a replication policy is created using the Cloudera Replication Manager. When the policy is created, click ⊙ to start uploading the SQL migration. At this step, the Hive scripts from the source cluster are copied to the Hive S3 bucket on the destination. When the Hive SQL Migration is finished, click ⊙ to finalize the replication policies.

What to do next

When all of the steps are successfully completed, the migration of Hive queries from CDH or Cloudera Private Cloud Base to Cloudera Public Cloud is finished. You can restart the queries on the destination Cloudera Data Engineering Data Hub cluster using Command Line Interface (CLI) or Hue.

Migrating HBase tables

During the HBase migration, the HBase tables with their related metadata are migrated from a CDH or Cloudera Private Cloud Base cluster to an Cloudera Operational Database Data Hub cluster.

About this task

Before the migration, the source cluster is scanned to collect the HBase tables. During the migration process, the tables are not affected on the source cluster and can remain part of running jobs.

Before you begin

- Ensure that Cloudera Migration Assistant is set up correctly using the steps in Setting up Cloudera Migration Assistant server.
- Ensure that you have met the requirements detailed in Reviewing prerequisites before migration.
- Ensure that you have a CDH 5, CDH 6 or Cloudera Private Cloud Base cluster registered as a source from which you want to migrate your HBase tables. If you do not have a source cluster yet, complete the steps in Registering source clusters.
- Ensure that you have an Cloudera Operational Database Data Hub cluster registered as a destination cluster to which you want to migrate your HBase tables. If you do not have a destination cluster yet, complete the steps in Registering destination clusters.
- Ensure that you have one of the following parcels (together with its corresponding .sha files) procured from Cloudera Support based on which CDH version you use:
  - - CLOUDERA_OPDB_REPLICATION-1.0-1.CLOUDERA_OPDB_REPLICATION5.14.4.p0.31473501-el7.parcel
    - CLOUDERA_OPDB_REPLICATION-1.0-1.CLOUDERA_OPDB_REPLICATION6.3.3.p0.8959316-el7.parcel
    - Copy the OpDB Replication Manager parcels to the following directories based on the CDH version:
      - *[***CMA ROOT FOLDER***]*/parcels/hbase/cdh5/
      - *[***CMA ROOT FOLDER***]*/parcels/hbase/cdh6/

Procedure

1. Click on the CDH or Cloudera Private Cloud Base cluster you want to use for the migration on the **Clusters** page.
2. Click Start Scanning to open the **Scan Settings**.
3. Select HBase table scan.
4. Click Scan selected.

   You will be redirected to the scanning progress, where you can monitor if the scanning process was successful or encountered any error.
5. Click on HBase table to view the collected tables when the scan is finished.
6. Add the needed HBase tables to Collections.

   Collections serve as an organization method to sort and bundle the tables into groups for the migration. You can create more collections beside the Default collection based on your requirements.

   After you are finished with sorting the tables to collections, you can start the migration process by creating the migration plan.

7. Click Create Migration or select Migrations Start Your First Migration .
   a) Select the source cluster, and click Next.
   b) Select the destination cluster, and click Next.
   c) Select the type of migration, and click Next.
   d) Select the collections that you want to migrate, and click Next.
   e) Provide he Cloudera workload user and password will be used to access Cloudera Manager of the Cloudera Data Hub cluster in Cloudera Public Cloud.
   f) Click Next.

      An overview of the migration plan is displayed. At this point, you can go back and change any configuration if the information is not correct. If the information is correct, click Create.
8. Click Execute Migrations when the migration plan is successfully created.

   You can view the steps that will be completed during the migration on the **Execution Step** tab.
9. Click Run to start migration.

   You also have the option to click ⋮ and select Run All. In this case the migration steps are executed manually. Choosing Run All in Current Phase enables you to manually start the next phase of the migration.

   During the HBase table migration, the source cluster is prepared with the HBase Replication Manager parcel.

   When the replication policy is submitted, click ⊙ to finalize the replication policies.

What to do next

When all of the steps are successfully completed, the migration of HBase tables from CDH or Cloudera Private Cloud Base to Cloudera Public Cloud is finished. You can start using the HBase tables on the destination Cloudera Operational Database Data Hub cluster with the available services.

Cloudera Public Cloud

Control Plane Regions

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Control Plane regions.....4

Cloudera Control Plane regions

Each Cloudera Control Plane and associated account runs within a geographic region. If your organization requires your Cloudera account and its data to remain within a specific geographical region, review the regions below to determine the most appropriate region to create your Cloudera account and Cloudera Control Plane in.

Every Cloudera account runs within a shared infrastructure called the Cloudera Control Plane. Historically, the Cloudera Control Plane and its services have been hosted in a single geographical region in the United States referred to as us-west-1.

Certain countries may have regulatory requirements that limit or forbid specific types of data from being sent or stored outside of the country or region where an organization operates. For this reason Cloudera has introduced new regions in which the Cloudera Control Plane can run. By choosing a region other than us-west-1, you can ensure that sensitive metadata (such as information about Cloudera users) does not leave its region or country of origin.

During the Cloudera account creation process, your administrator will work with the account creation team to choose the appropriate Cloudera Control Plane region to create your Cloudera account in. Your Cloudera account together with users, groups, and Cloudera resources like environments, Data Lakes, and Cloudera Data Hub clusters are tied to the Cloudera Control Plane region that they are created in. After account creation, there is no option to change the region in which the Cloudera Control Plane and Cloudera account will run.

Regions	us-west-1	eu-1	ap-1
Location	United States	Germany	Australia
Year Opened	2019	2021	2021

As new Cloudera Control Plane regions are added, not every Cloudera feature or data service will be available in every region, though these features will eventually be supported for every region. Refer to the following support matrix to determine which Cloudera data services and major Cloudera features are available within the region where your Cloudera Control Plane resides:

CDP service	us-west-1 (USA)	eu-1 (Germany)	ap-1 (Australia)
Data Engineering	GA	GA	GA
Data Hub	GA	GA	GA
Data Warehouse	GA	GA	GA
DataFlow	GA	GA	GA
Machine Learning	GA	GA	GA
Operational Database	GA	GA	GA
Replication Manager	GA	GA	GA
Workload Manager	GA	GA	GA
Data Catalog	GA	GA	GA

CDP feature	us-west-1 (USA)	eu-1 (Germany)	ap-1 (Australia)
Auditing	GA	GA	GA
CCMv1	GA	N/A	N/A

CDP feature	us-west-1 (USA)	eu-1 (Germany)	ap-1 (Australia)
CCMv2	GA	GA	GA
Classic Clusters	GA	GA	GA
Unified Diagnostics	GA	N/A	N/A
Metering	GA	GA	GA

The following terminology is used in the above table:

- GA - The service is general availability in the specific region
- Preview - The service is preview in the specific region
- N/A - The service is not available in the specific region

Although there are few visible differences in Cloudera depending on which Control Plane region you choose, it is important that you set up outbound network access (AWS, Azure, GCP) correctly for the Cloudera Control Plane region that you use. You should also take note that for different Cloudera Control Plane regions, the endpoints will be different for the Cloudera Control Plane API and UIs.

You can configure your Cloudera client to use a certain Cloudera Control Plane region by default. For more information on this, see Configuring Cloudera Client.

Important: Note that the name of the region in which a Cloudera Control Plane runs is used within CRNs (the ID that Cloudera uses to define a resource, such as an environment, user, or cluster); within API endpoints; and more, though in Cloudera documentation, examples of these strings often use “us-west-1.”

Cloudera Public Cloud

Diagnostic Bundle Collection

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Send diagnostic bundle to Cloudera.....4

CDP CLI commands..... 8

Diagnostic bundle content.....11

Send a diagnostic bundle to Cloudera Support

You can trigger diagnostic bundle collection for your Cloudera environment in order to send Data Lake, FreeIPA, and Cloudera Data Hub logs to Cloudera Support for support case resolution.

Note: Diagnostic bundles can only be generated for Data Lake, FreeIPA, Cloudera Data Hub, Cloudera DataFlow, and Cloudera Data Warehouse. Additionally, there are separate interfaces for generating bundles for Cloudera Data Engineering and Cloudera AI. Other Cloudera services are not supported. A separate bundle is created for each cluster. To send a Cloudera AI diagnostic bundle, follow the instructions in Download diagnostic bundles for an Cloudera AI workspace instead of following the instructions below. To send a Cloudera Data Engineering diagnostic bundle, follow the instructions in Downloading a diagnostic bundle for Cloudera Data Engineering.

Required role: You need the EnvironmentAdmin or Owner role for the environment for which you would like to create a bundle. If you would like to include Cloudera Data Warehouse diagnostics in the bundle, you also need the DWAdmin role.

Prerequisites

- FreeIPA deployment time must be September 2020 or later.
- The VMs from which diagnostic data is to be collected must be running.
- Salt minions must be started properly on the Cloudera Manager nodes.

Steps

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Click Help in the bottom left corner and select Collect Diagnostic Data.

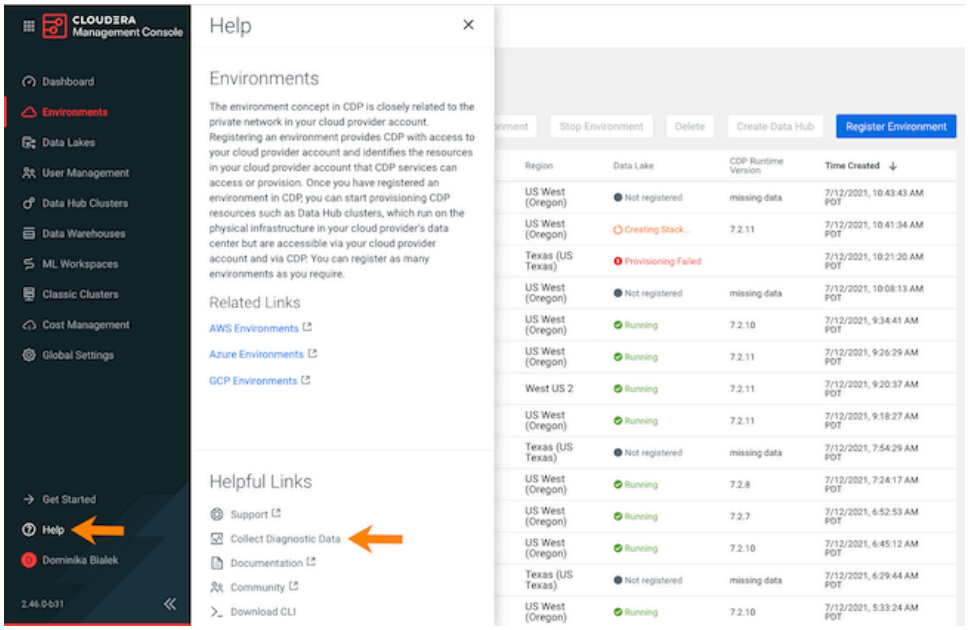

4. Click the Collect New Diagnostic Data button.

5. On the General Information page, provide the following:
   - Select the environment for which you would like to generate a diagnostic bundle. This is required.
   - Enter the related support case number. This is required. You can obtain the case number from the MyCloudera portal or from your email by checking support case notification emails.
   - Under Time Range, you can optionally specify a specific point in time (date and time) from which to start the collection. The cutoff date and time is automatically set to the time when you initiate the collection and it cannot be modified.

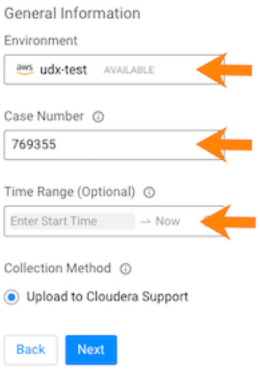

6. Click Next.

7. On the Selected Services page:
   - Cloudera automatically collects logs from all nodes of the Data Lake and FreeIPA clusters. By default diagnostics is collected from all nodes, but you can optionally limit it to one or more specific nodes.
   - Under Cloudera Data Hub clusters, you can optionally select to generate diagnostic bundles for Cloudera Data Hub clusters. After adding a Cloudera Data Hub, by default diagnostics is collected from all nodes, but you can optionally limit it to one or more specific nodes.
   - Under Cloudera Data Warehouse clusters, you can optionally select to generate diagnostic bundles for Hive or Impala virtual warehouses created via Cloudera Data Warehouse.
   - Under Cloudera DataFlow, you can optionally select to generate diagnostic bundles for Cloudera DataFlow environment components and DataFlow deployments.

     Environment Components

     Select a list of DataFlow Environment Components from which you want to collect logs. This list operates as an allowed list when present, only collecting logs for specified components.

     If you leave this list empty, it implies you want to collect logs from all components.

     Select Components

     Specify a list of DataFlow Deployment CRNs for which you want to collect logs. This list operates as an allowed list when present, only collecting logs for specified deployments.

     If you leave this list empty, it implies log collection from all Flow Deployments in the environment.

     Collection Scope

     Specify whether you want to collect environment logs, deployment logs, or both.

     Regardless of the selection you make here, logs for environment components and deployments you specified in their respective allowed lists will be collected.

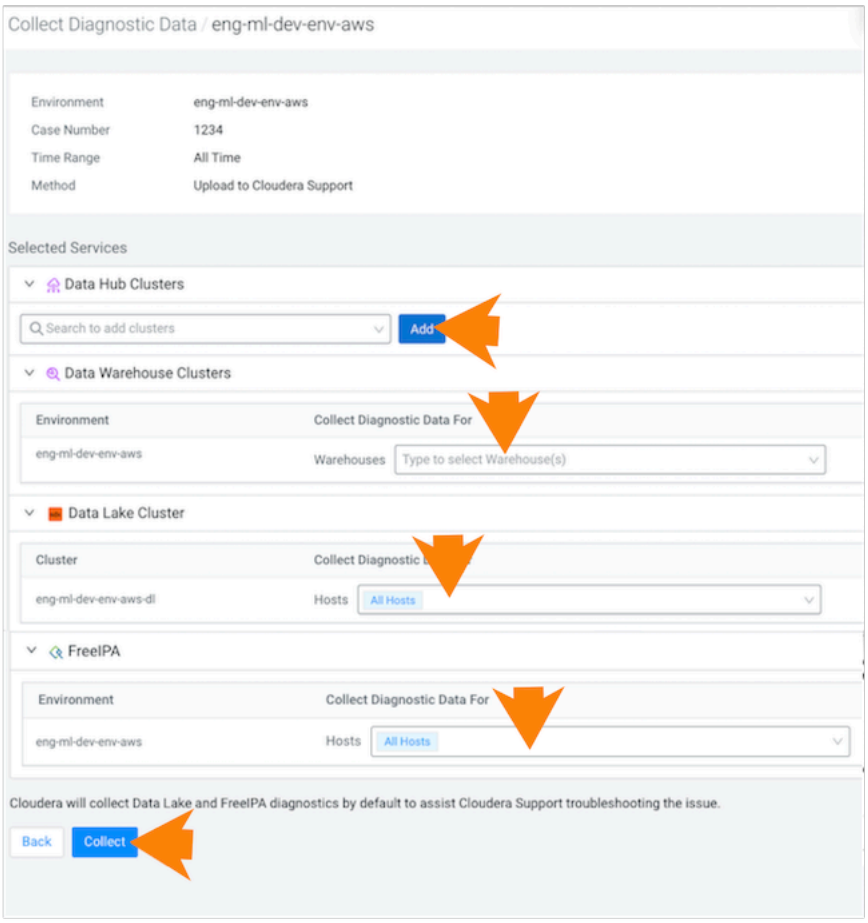

8. Click Collect.
9. New entries appear on the page, allowing you to track the status of the log collection. Since a separate bundle is created for each service, there is one entry for each of the following: Data Lake, FreeIPA, one entry for each Cloudera Data Hub, one entry for each Cloudera Data Warehouse warehouse, and one entry for each Cloudera DataFlow service. While the collection is in progress, the collection status is In progress:

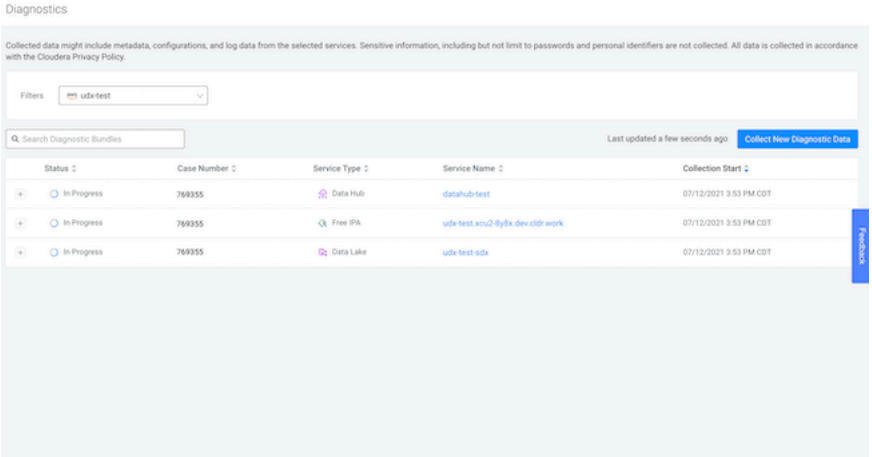

10. Once the collection is complete, the status changes to Completed. This means that Cloudera Support can now access the diagnostic bundles:

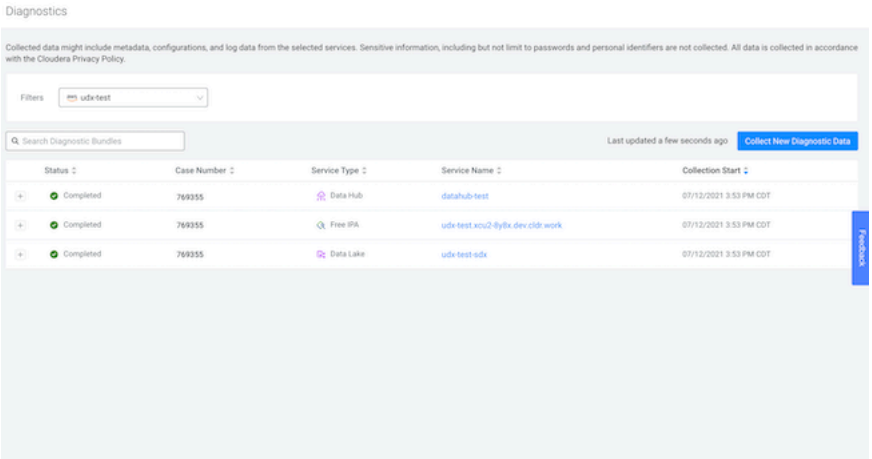

Related Information

Diagnostic bundle content

CDP CLI commands for generating a diagnostic bundle

CDP CLI commands for generating a diagnostic bundle

Cloudera includes a set of CLI commands that allow you to generate an on-demand diagnostic bundle that gathers logs and command output from your Data Lake, FreeIPA, or Cloudera Data Hub cluster VMs. The bundle can also be used for support case troubleshooting if the logs captured via the equivalent UI option are not sufficient.

For each resource type (freeipa, datalake, or datahub), there are two commands that you can use:

- get-<resource-type>-log-descriptors - Allows you to obtain a list of logs that get collected for diagnostics.
- collect-<resource-type>-diagnostics - Triggers diagnostics collection. You need to provide a CRN of a resource (Data Lake, FreeIPA, or Cloudera Data Hub cluster), a description, and a destination for the bundle.
- list-<resource-type>-diagnostics - Lists the running or finished diagnostic operations.
- cancel-<resource-type>-diagnostics - Stops the ongoing diagnostics operations.

Required role: You need the EnvironmentAdmin or Owner role for the environment for which you would like to create a bundle.

Prerequisites

- FreeIPA deployment time must be September 2020 or later.
- The VMs from which diagnostic data is to be collected must be running.
- Salt minions must be started properly on the Cloudera Manager nodes.

get-<resource-type>-log-descriptors

The get-<resource-type>-log-descriptors commands allow you to obtain a list of logs that get collected. These commands can also be used to obtain available log label/path pairs that can be provided in the collect-<resource-type>-diagnostics command via the --labels parameter in order to limit the command output.

The output is printed in the Event History tab on the UI in Data Lake or Cloudera Data Hub details. Since there is no FreeIPA Event History on the UI, in case of FreeIPA the output is saved to /var/log/salt/minion, which can be accessed from your logs location in cloud storage.

Commands

To gather available log label/path pairs, use one of the following commands:

cdp environments get-freeipa-log-descriptors

cdp datalake get-datalake-log-descriptors

cdp datahub get-datahub-log-descriptors

collect-<resource-type>-diagnostics

The collect-<resource-type>-diagnostics commands trigger log collection. You need to provide a CRN of a resource (Data Lake, FreeIPA, or Cloudera Data Hub cluster), a description, and a destination for the bundle. The generated bundle can be saved to cloud storage or to a local file. The command progress (including the location where the bundle is generated) is printed in the Event History tab on the UI.

Note: This feature only works out of the box with environments created or upgraded in October 2020 or later. If your environment is older and your nodes have internet access, you can use the --update-package option to download the newest library that is needed for the collection.

Prerequisites

- If you would like to use cloud storage as a destination, ensure that the Logger and IDBroker instance profiles (on AWS) or managed identities (on Azure) are able to write to the specified cloud storage location.
- If you would like to use Cloudera support as a destination, your network must be able to reach the following destinations: https://dbusapi.us-west-1.altus.cloudera.com if your CDP account runs in Control Plane region us-west-1, or https://api.<control-plane-region>.cdp.cloudera.com if your CDP account runs in any other Control Plane region (only required if your cloud provider is AWS) and https://cloudera-dbus-prod.s3.amazonaws.com (required even if your cloud provider is not AWS).

Example commands

To run collection with salt script included and update cdp-telemetry package on the nodes, use one of the following commands, Replace <DESCRIPTION> with an actual description and the <NAME-OR-CRN>, <DATALAKE-CRN>, and <DATAHUBCRN> with an actual CRN.

To send the diagnostics bundle to cloud storage, use:

```
cdp environments collect-freeipa-diagnostics --environment-name <NAME-OR-CRN
> \
--description <DESCRIPTION> \
--destination CLOUD_STORAGE \
--include-salt-logs
```

```
cdp datalake collect-datalake-diagnostics --crn <DATALAKE-CRN> \
--description <DESCRIPTION> \
--destination CLOUD_STORAGE \
--include-salt-logs
```

```
cdp datahub collect-datahub-diagnostics --crn <DATAHUB-CRN> \
--description <DESCRIPTION> \
--destination CLOUD_STORAGE \
--include-salt-logs
```

To send the diagnostic bundle directly to Cloudera support, use:

```
cdp datahub collect-datahub-diagnostics --crn <DATAHUB-CRN> \
--description <DESCRIPTION> \
--case-number <SUPPORT_CASE_NUMBER>
--destination SUPPORT \
--update-package \
--include-salt-logs
```

To collect only specific custom logs (such as /var/log/audit/audit.log used in this example), use:

```
cdp environments collect-freeipa-diagnostics --environment-name <name> \
--description <DESCRIPTION>
--destination CLOUD_STORAGE \
--additional-logs path=/var/log/audit/audit.log,label=audit \
--labels audit
```

If you run this command:

- Logs specified under --additional-logs will be collected
- Due to the specified label, all other logs will be filtered out and no other logs will be included in the bundle.

Required parameters

--environment-name (for Data Lake or Cloudera Data Hub) or --crn (for Cloudera Data Hub)Provide a CRN of your Data Lake, FreeIPA, or Cloudera Data Hub cluster. You can obtain it by running a list or describe command on your resource.

--description Provide a description for the diagnostics bundle.

--destination Specify a destination of the diagnostics collection: SUPORT, CLOUD_STORAGE, or LOCAL:

- SUPPORT: Sends diagnostic bundles directly to Cloudera support.
- CLOUD_STORAGE: Saves collected diagnostics into a tar.gzfile file and uploads it to your logs cloud storage location specified during environment registration, to a PATH that looks like this:

  ```
  cluster-logs/<CLUSTER-TYPE>/<CLUSTER-NAME>_<UUID PART-OF-CLUSTER-CRN>/d
  iagnostics.
  ```

  The “cluster logs” directory is only created if your cloud storage location is a bucket. If your cloud storage location includes a directory within a bucket, then the “cluster logs” directory is not created.

  Note: The value of <CLUSTER_NAME> for FreeIPA is <ENVIRONMENT-NAME>-freeipa. For example, if your environment is called "test", the <CLUSTER_NAME> for FreeIPA is "test-freeipa".

- LOCAL: Diagnostics will be collected to a compressed file, and saved to /var/lib/filecollector on every cluster node.

Optional parameters

--labels You can use this to pass a list of labels that can be used to filter collected logs. This is useful if instead of including all logs, you would only include certain specific logs in the bundle. This list can be obtained from the get-<cluster-type>-log-descriptors response.

--additional-logs One or more additional VM log objects that will be included in the diagnostics bundle. A VM log object is a path/label pair. Cloudera support may ask you to provide specific values here.

--hosts Run diagnostics only the specified hosts (both IP addresses or fqdns are valid).

--host-groups Run diagnostics only on the those hosts that are included in the specified host groups.

--exclude-hosts Same as the --hosts filter but for exclusion.

--skip-unresponsive-hosts Skip unhealthy hosts from the diagnostics collection (where "unhealthy" means that salt minion is unresponsive). If this is not used, the diagnostics will fail if there is an unresponsive host during diagnostics.

--include-salt-logs If this is set, salt minion/master logs will be included in the diagnostic bundle.

--case-number Allows you to provide a support case number.

--update-package If this is set, during diagnostics collection, the required third party applications will be downloaded and updated on the VM nodes. This requires internet access.

--storage-validation You can use this if using CLOUD_STORAGE as a destination. If this is set, cloud storage write validation check will happen during an initial phase of the diagnostics collection.

For more information about the command and its options, use the --help CLI option.

list-<resource-type>-diagnostics

Lists the running or finished diagnostic operations.

Examples:

```
cdp environments list-freeipa-diagnostics --environment-name <NAME-OR-CRN>
cdp datalake list-datalake-diagnostics --crn <DATALAKE-CRN>
cdp datahub list-datahub-diagnostics --crn <DATAHUB-CRN>
```

cancel-<resource-type>-diagnostics

Stops the ongoing diagnostics operations.

Examples:

```
cdp environments cancel-freeipa-diagnostics --environment-name <NAME-OR-CRN>
cdp datalake cancel-datalake-diagnostics --crn <DATALAKE-CRN>
cdp datahub cancel-datahub-diagnostics --crn <DATAHUB-CRN>
```

Related Information

Diagnostic bundle content
Send a diagnostic bundle to Cloudera Support
Downloading Impala diagnostic bundles from AWS
Downloading Diagnostic Bundles for a Workspace

Diagnostic bundle content

This topic summarizes the content of a diagnostic bundle.

The following information is included in a diagnostic bundle:

- Cluster details required to identify the bundle (account ID, cluster CRN, cluster type, and so on)
- Redacted service and system logs
- Web server configuration from the salt master node (nginx)
- An HTML report including:
  - The result of running a number of system-level commands on each host
  - Network endpoint accessibility for outbound network traffic requirements
  - Health details about the required infra services
- When triggering diagnostic bundle collection via Cloudera CLI, you may specify additional logs that are not part of the default log collection to be collected in the diagnostic bundle.

The information is collected from all running nodes of your cluster (Data Lake, FreeIPA, or Cloudera Data Hub). It is not collected from nodes that are not running, such as nodes that are currently being provisioned or stopped or deleted nodes.

Note: The logs collected by default in the on-demand diagnostic bundle are the same as when you enable cluster log collection. Whether or not you enabled the log collection feature on tenant or environment level, the on-demand diagnostic bundle can be generated independently when needed.

Note: The Cloudera Manager service logs can also be gathered via Cloudera Manager, as described in Collecting Data Lake diagnostics via Cloudera Manager. However, if Cloudera Manager is down or cannot be started, generating a diagnostic bundle may help troubleshoot Cloudera Manager issues.

Cloudera Public Cloud

GCP Onboarding Quickstart

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

GCP quickstart 4
Verify GCP prerequisites 4
Create a provisioning credential 5
Create GCP prerequisites 7
Register a Cloudera environment 9

GCP quickstart

In this quickstart, we will show you step by step how to connect Cloudera to your Google Cloud Platform (GCP) account, so that you can begin to provision clusters and workloads.

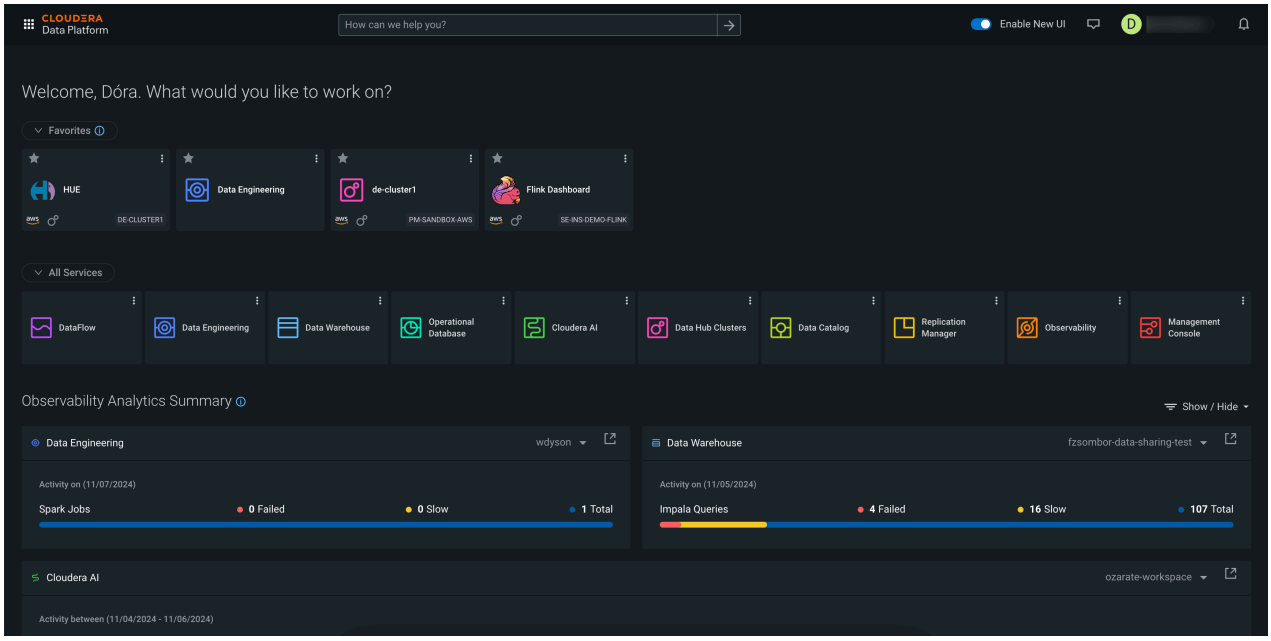

To complete this quickstart, you will need access:

- The Cloudera console pictured above
- The GCP console

Note: This GCP onboarding quickstart is intended for simple Cloudera evaluation deployments only. It may not work for scenarios where GCP resources such as VPC network, firewall rules, storage accounts, and so on, are pre-created or GCP accounts have restrictions in place.

The steps that we will perform are:

Step 0: Verify the GCP prerequisites

Step 1: Create a provisioning credential

Step 2: Create GCP prerequisites

Step 3: Register a GCP environment in Cloudera

Verify GCP cloud platform prerequisites

Before getting started with the Google Cloud Platform (GCP) onboarding quickstart, review and acknowledge the following:

- This GCP onboarding quickstart is intended for simple Cloudera evaluation deployments only. It may not work for scenarios where GCP resources such as VPC network, firewall rules, storage accounts, and so on, are pre-created or GCP accounts have restrictions in place.
- Users running the GCP onboarding quickstart should have Cloudera Admin role or Power User role in Cloudera subscription.
- The following APIs should be enabled in the project that you would like to use to run the quickstart: GCP APIs.

- In order to run the quickstart, the Google APIs service agent user must be set to Owner. For instructions on how to check or update the Google APIs service agent permissions, see Grant permissions to the Google APIs service account.

  Note: The Grant permissions to the Google APIs service account instructions show how to assign the Role Administrator role, but you should assign the Owner role instead.

  If your organization's policies don't allow you to assign the Owner role, and you are required to use granular roles or permissions instead, you should make sure to assign, among other roles, the Role Administrator role (or equivalent granular permissions allowing access to the Deployment Manager).
- This GCP onboarding quickstart uses a Deployment Manager template that automatically creates the required resources such as the VPC network, firewall rules, storage buckets, service accounts, and so on.
- Cloudera Public Cloud relies on several GCP services that should be available and enabled in your project of choice. Verify if you have enough quota for each GCP service to set up Cloudera. See list of GCP resources used by Cloudera.

If you have more complex requirements than those listed here, contact Cloudera Sales Team to help you with the onboarding process.

Create a provisioning credential

The first step is to create a provisioning credential. The Cloudera credential is the mechanism that allows Cloudera to create resources inside of your GCP account.

Steps

1. Log in to the Cloudera web interface.
2. From the Cloudera home screen, click the Cloudera Management Console icon.
3. In the Cloudera Management Console, select Shared Resources > Credentials from the navigation pane.
4. Click on the "Create Credential" button.
5. Select the (Google Cloud Platform) tab.
6. Give your credential a name and description.

7. Copy the script provided under “Create Service Account”:

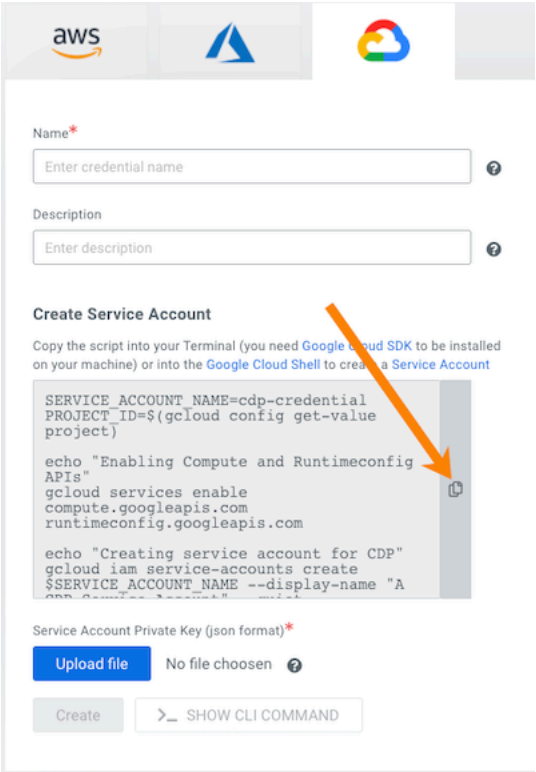

This script can create the service account that is a prerequisite for the Cloudera credential.

8. Navigate to the GCP console.
9. Verify that you are in the project that you would like to use for Cloudera. Switch projects if needed:

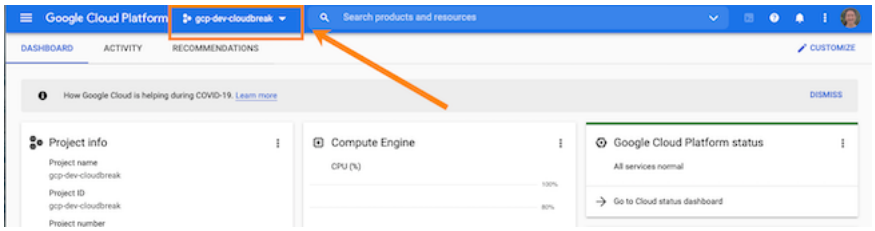

10. Open the Cloud Shell (available from upper right corner):

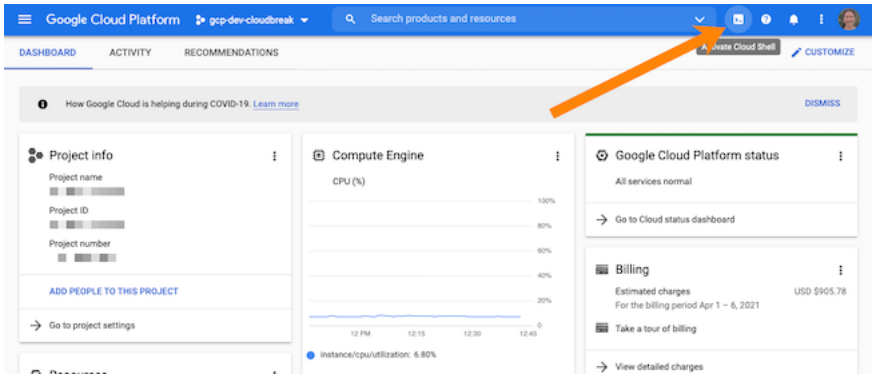

The Cloud Shell window opens in the bottom of the browser window.

11. Paste the script directly into the Cloud Shell terminal.
12. When prompted, click Authorize.

13. The script will run and then end by prompting you to download the credential file to your local machine. Click Download to download the file:

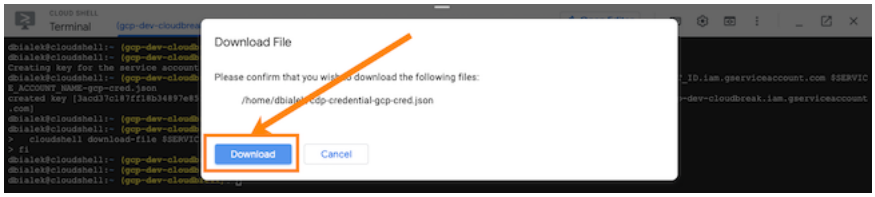

14. Head back to the Cloudera console and upload the JSON credential file you just downloaded from the GCP console:

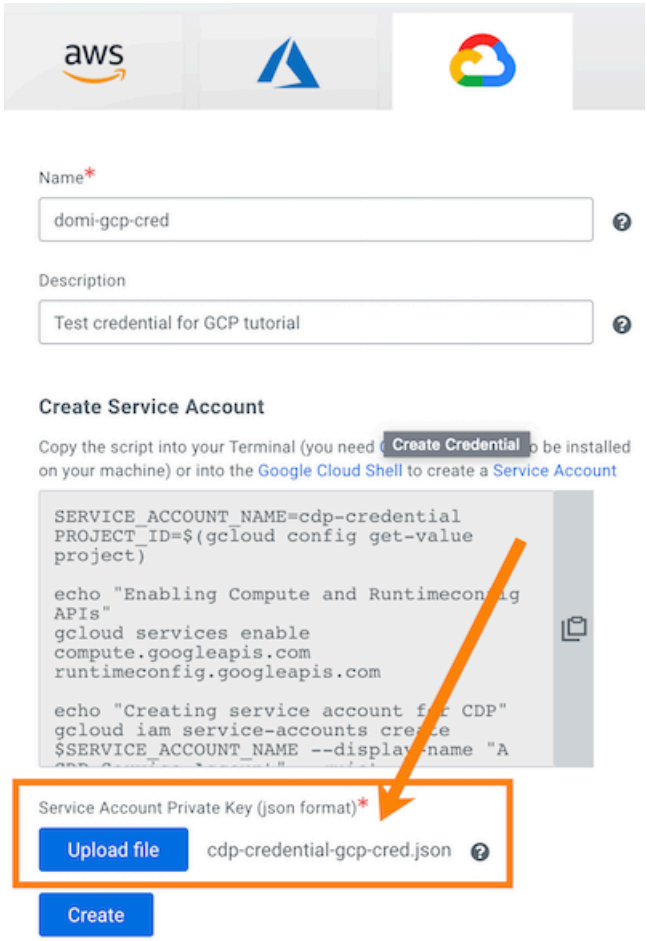

15. Click the "Create" button and you're done!

Create GCP prerequisites

The second step is to create the Cloudera prerequisites in your GCP project. To get this done quickly, we will use a script that creates a VPC network, a subnet, firewall rules, service accounts, storage buckets, and so on.

Steps

1. Navigate to the browser tab with the GCP console.

2. Click on the project navigation widget:

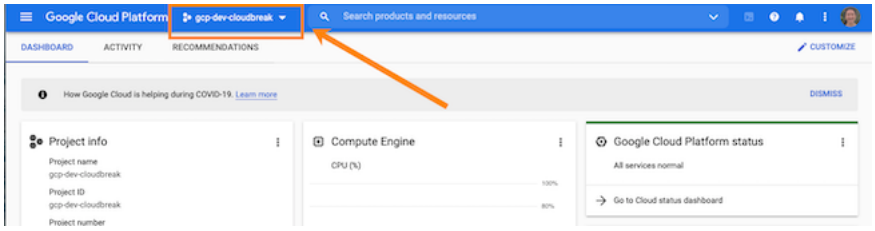

3. A window appears, allowing you to select a project. Copy the ID of the project where you created your credential earlier:

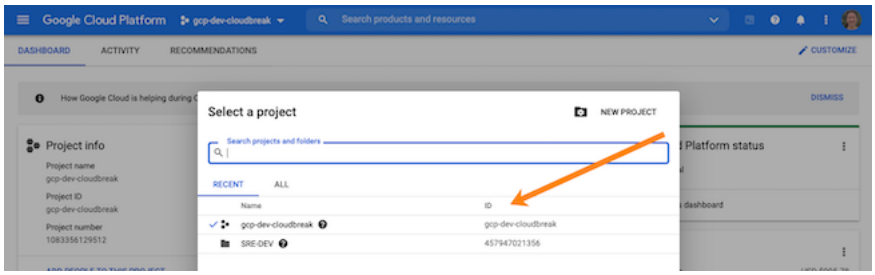

You will need it in a later step

4. In the GCP console, download the following files gcp-script.sh and gcp_param.yml:

```
wget https://docs.cloudera.com/cdp-public-cloud/cloud/gcp-script.sh
wget https://docs.cloudera.com/cdp-public-cloud/cloud/gcp_param.yml
```

This is a script that creates all of the prerequisite resources required for Cloudera.

5. Run the provided bash script using the following command:

```
bash gcp-script.sh <prefix> <region> <project-id> <IP-CIDR-to-whitelist>
```

Replace the following with actual values:

- Prefix - A prefix to prepend to names of all resources that the script creates for you. For example, if your chosen prefix is "cloudera", Cloudera prepends "cloudera-" to the names of all created resources.
- Region - A GCP region where you would like to deploy your environment. For a list of supported regions, see Supported GCP regions.
- Project ID - The project ID that you obtained in an earlier step. This is the project where you will deploy the resources required for a Cloudera environment.
- CIDR-to-whitelist - The IPv4 CIDR range for SSH and UI access.

For example:

```
bash gcp-script.sh test us-east4 gcp-dev 73.221.71.0/24
```

6. The script creates a new deployment called <prefix>-cdp-deployment in the Deployment Manager and creates resources in your GCP account:

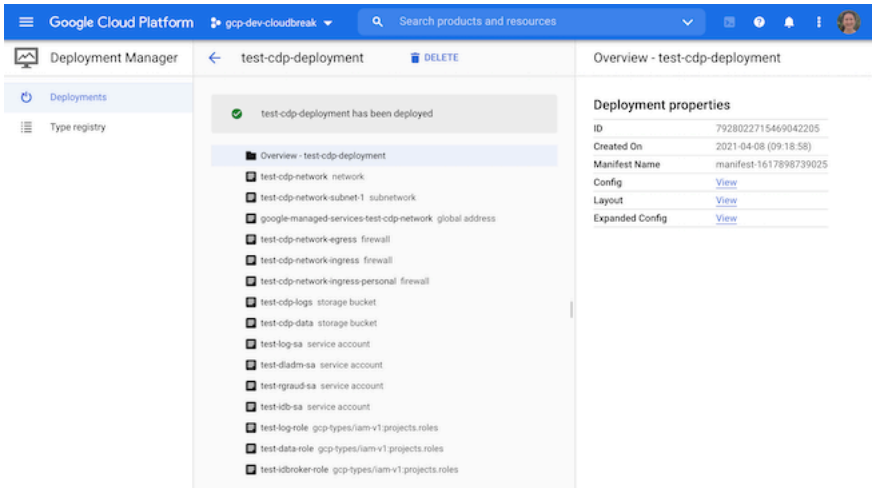

The script does the following:

1. Verify that the correct number of arguments were supplied.
2. Replace the Deployment Manager config parameters with those arguments that you supplied.
3. Check for the existence of a custom IAM role with the provided prefix (i.e. check if you have run this script before with the same prefix).
4. Run the Deployment Manager (which creates resources such as a VPC network, a subnet, firewall rules, service accounts, storage buckets, and so on).
5. Add policy bindings to the created service accounts
6. Change GCS bucket permissions for created service accounts
7. Create a VPC peering to servicenetworking.googleapis.com (for purposes of the CloudSQL DB that Cloudera creates).

Once the deployment is ready, you will see a message “<prefix>-cdp-deployment has been deployed”. At that point, you can proceed to the next step.

Note: If you need to delete all of the prerequisites created by the script, you can delete the deployment using the Deployment Manager UI (using the "Delete" button visible in the above screenshot).

Register a Cloudera environment

The third (and last) step is to register your GCP environment in Cloudera. You will:

- Use the credential created in Step1.
- Point Cloudera to the resources created in Step 2.

You have two options for performing the environment registration step:

- Option 1: Cloudera web interface
- Option 2: CDP CLI

Prerequisites

You need an RSA key pair. You will be asked to provide a public key and you will use the matching private key for admin access to Cloudera instances.

Register a GCP environment using Cloudera web interface

1. Switch back to the browser window with the Cloudera console.

2. Navigate to the Cloudera Management Console > Environments.
3. Click the Register Environment button.
4. In the General Information section provide the following:
   - Environment name - Provide a name for the environment.
   - Select Cloud Provider - Select Google Cloud.
5. Under Select Credential, select the credential that you created in Step 1.

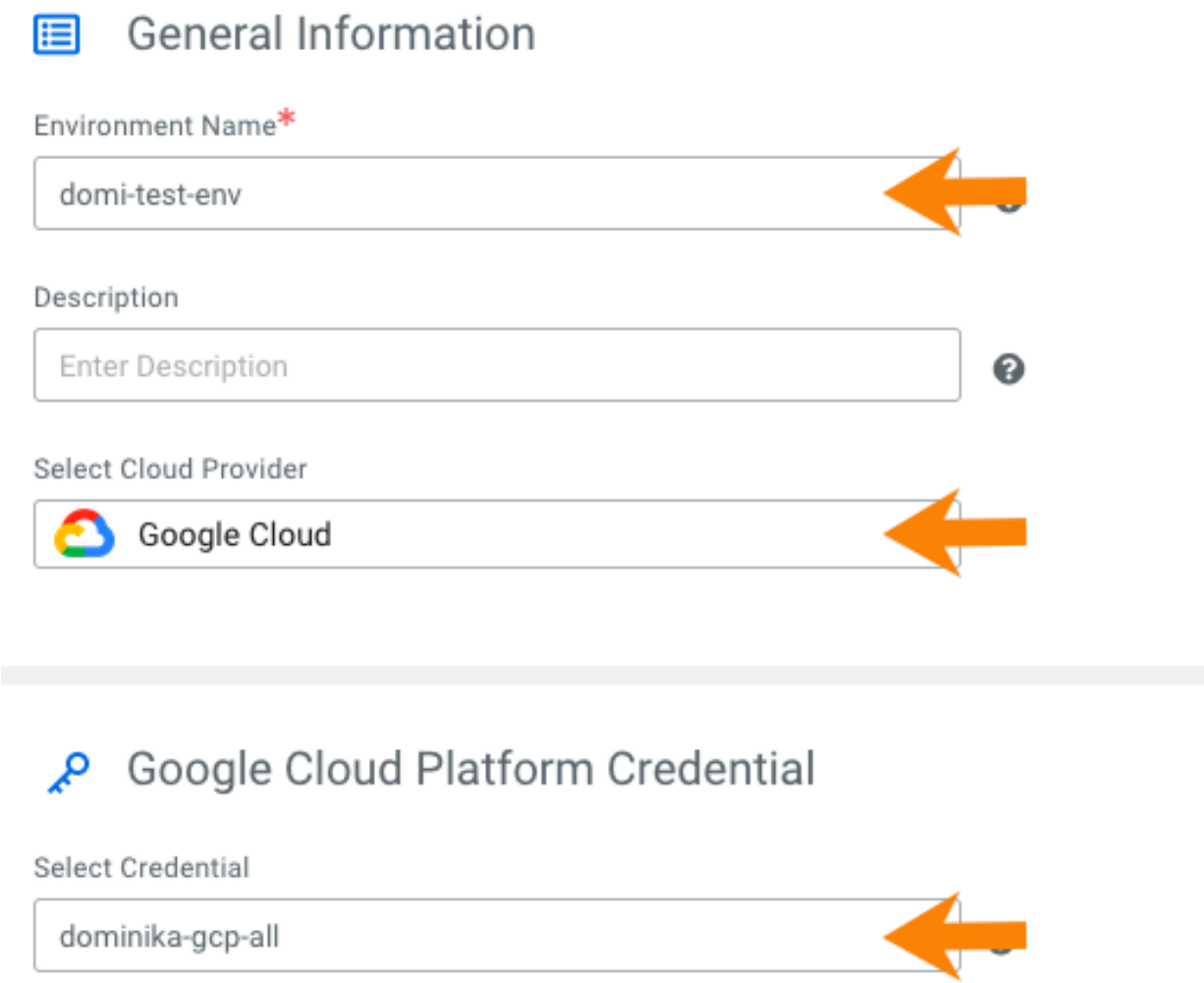

6. Click Next.
7. In the Data Lake Settings section, provide the following:
   - Data Lake Name - Enter a name for the Data Lake that Cloudera creates for your environment.
   - Data Lake Version - The latest version should be pre-selected by default.

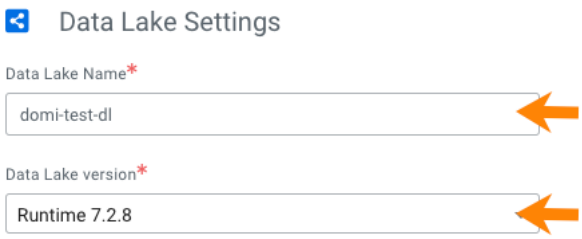

8. In the Data Access and Audit section, provide the following:

- Assumer Service Account - Select <prefix>-idb-sa. The prefix is what you provided in Step 2.
- Storage Location Base - Enter <prefix>-cdp-data
- Data Access Service Account - Select <prefix>-dladm-sa
- Ranger Audit Service Account - Select <prefix>-rgraud-sa
- Under IDBroker Mappings:

  a. Click Add
  b. Under User or Group, select your user
  c. Under Service Account, enter the following service account name <prefix>-dladm-sa

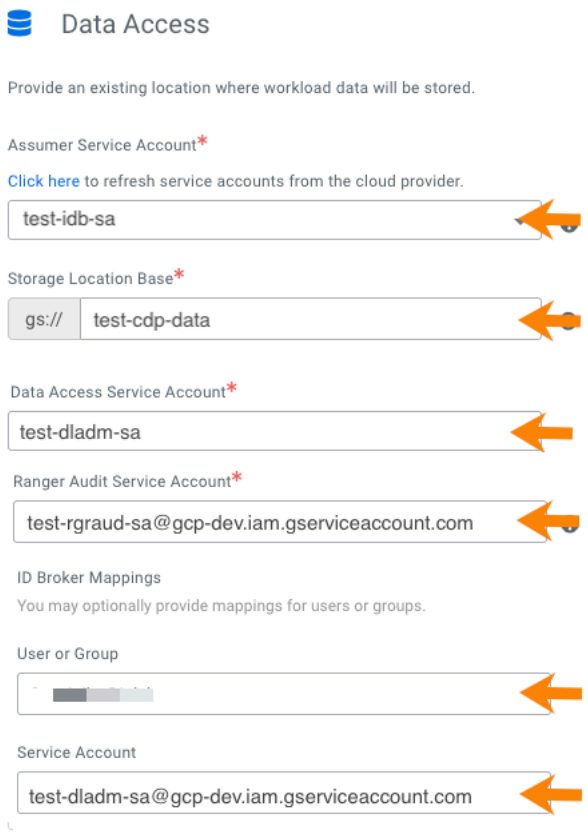

9. Click Next.

10. Under Region, Location, select the same region that you provided in Step 2.

11. In the Network section, provide the following:

- Select Network - Select the network called <prefix>-cdp-network
- Select Subnets - Select <prefix>-cdp-network-subnet-1
- Ensure that CCM is enabled.
- Enable Create Public IPs.

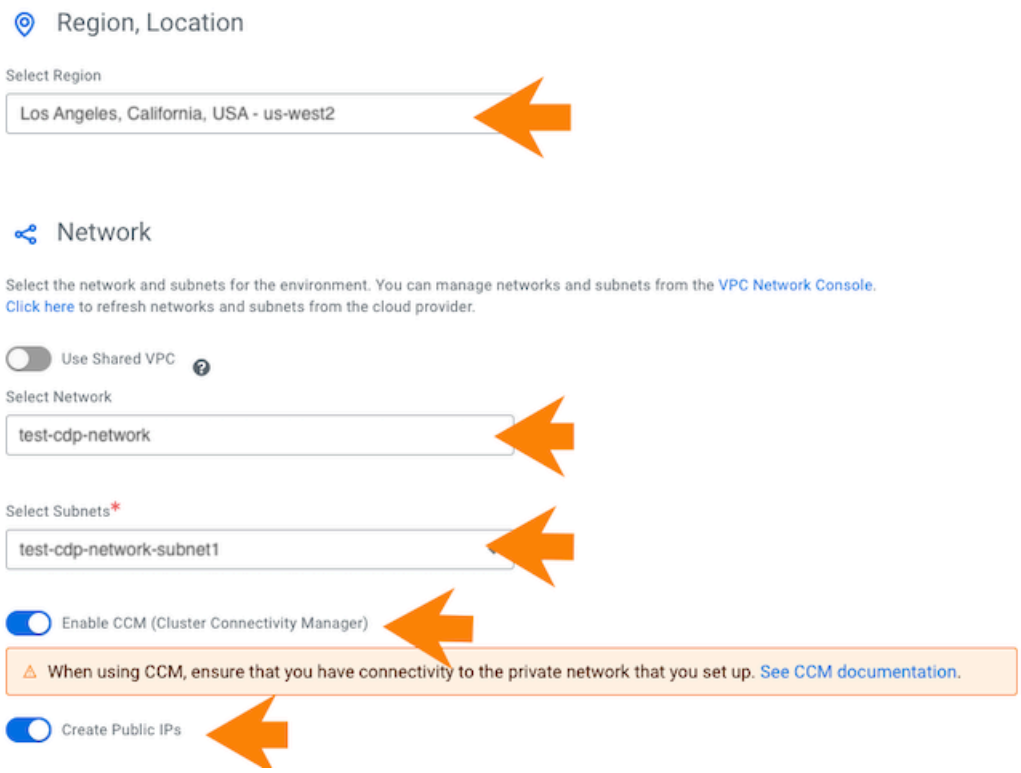

12. Under Security Access Settings, under Select Security Access Type, select Do not create firewall rule.

13. Under SSH Settings, paste your RSA public key.

14. Under Add Tags add tags if necessary.

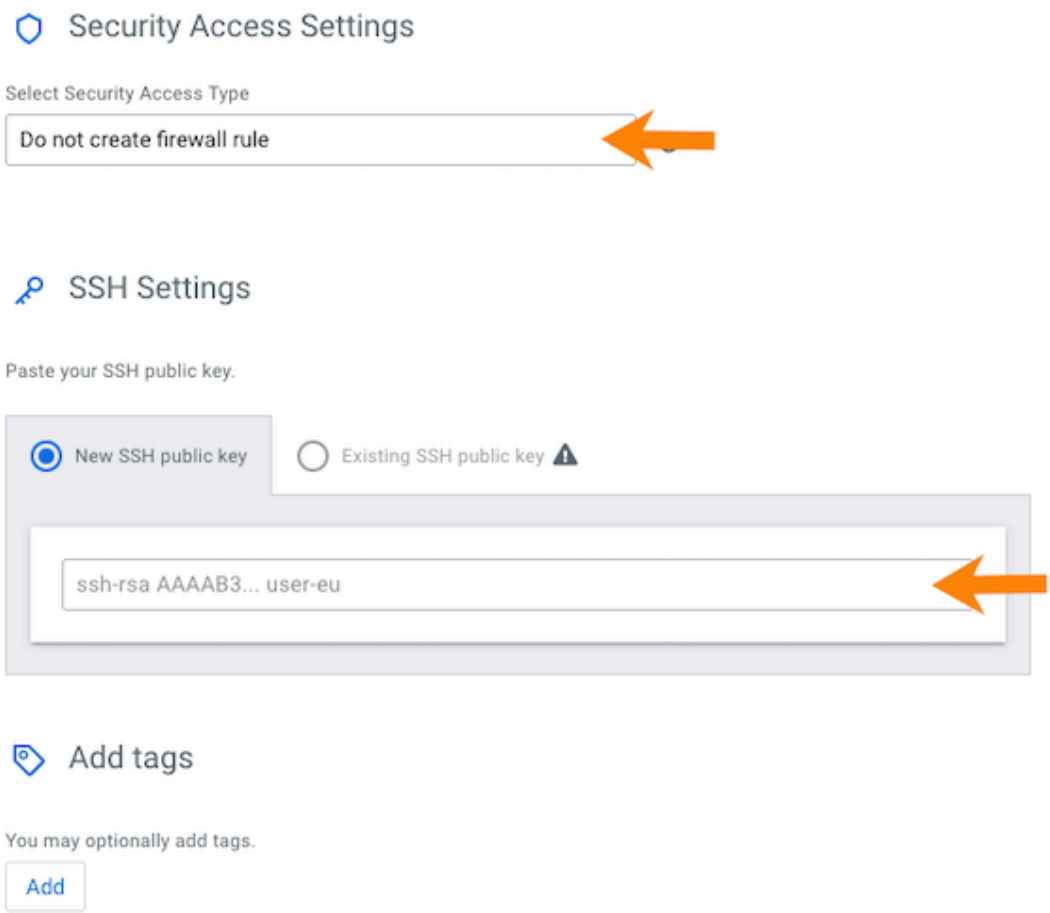

15. Click Next.

16. Under Logs, provide the following:
    - Logger Service Account - Select <prefix>-log-sa
    - Logs Location Base - Enter <prefix>-cdp-logs
    - Backups Location Base - Enter <prefix>-cdp-backup

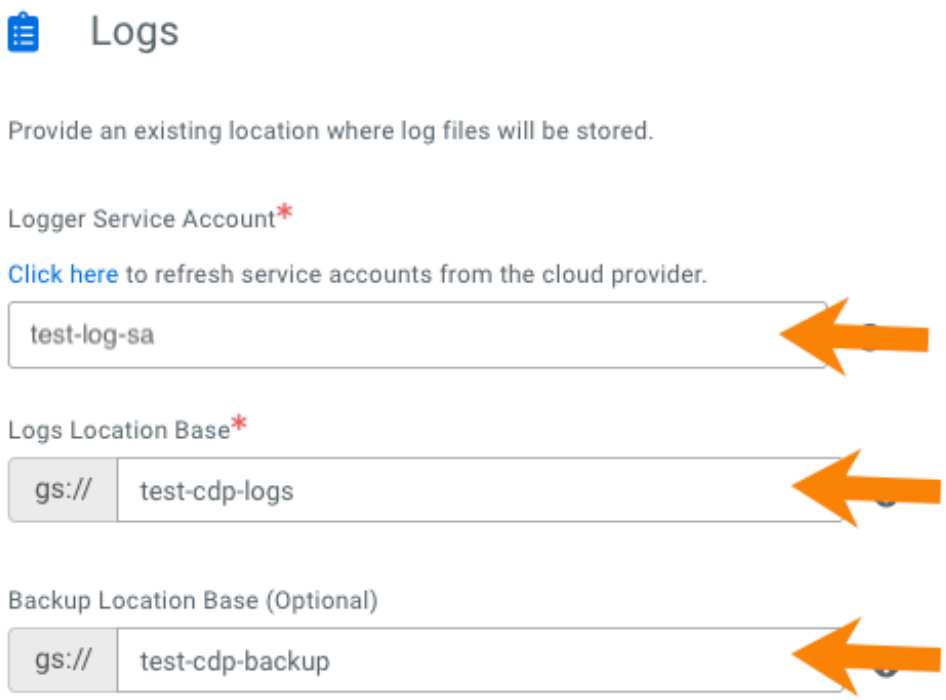

17. Click Register Environment.
18. Once your environment is created, its status will change to Available and the Data Lake status will change to Running.

Once your environment is running, you can start creating Data Hub clusters.

Register a GCP environment using CDP CLI

1. Install and configure CDP CLI. If you haven't, refer to CLI client setup.
2. Open the terminal app on your computer.
3. Create your environment using the following command. Replace the following with actual values:
    - <NAME_OF_YOUR_CLOUDERA_CREDENTIAL> - Replace this with the actual name that you provided on the Cloudera web UI in step 1.
    - <REGION> - Replace this with the ID of the region selected in step 2.
    - <RSA_PUBLIC_KEY> - Replace this with your RSA public key. You will use the matching private key for admin access to Cloudera instances.
    - <PREFIX> - Replace this with the prefix specified in step 2.
    - <PROJECT_ID> - Replace this with the ID of the GCP project specified in step 2.

```
cdp environments create-gcp-environment  --environment-name "<PREFIX>-cdp-
env" \
    --credential-name "<NAME_OF_YOUR_CLOUDERA_CREDENTIAL>" \
    --region "<REGION>" \
    --public-key "<RSA_PUBLIC_KEY>" \
    --log-storage storageLocationBase="gs://<PREFIX>-cdp-logs/",serviceAcc
ountEmail="<PREFIX>-log-sa@<PROJECT_ID>.iam.gserviceaccount.com", backup
StorageLocationBase="gs://<PREFIX>-cdp-backup" \
```

```
    --existing-network-params networkName="<PREFIX>-cdp-network",subnet
Names="<PREFIX>-cdp-network-subnet-1",sharedProjectId="<PROJECT_ID>"  \
    --enable-tunnel \
    --use-public-ip
```

4. Find your user CRN using the following command:

```
user_crn=$(cdp iam get-user | jq -r .user.crn)
```

5. Set the IDBroker mappings between users and service accounts using the following command. Replace the following with actual values:
   - <PREFIX> - Same as used earlier
   - <PROJECT_ID> - Same as used earlier
   - <USER_CRN> - Replace with your user CRN.

```
cdp environments set-id-broker-mappings \
    --environment-name "<PREFIX>-cdp-env" \
    --baseline-role "<PREFIX>-rgraud-sa@<PROJECT_ID>.iam.gserviceaccount.c
om" \
    --data-access-role "<PREFIX>-dladm-sa@<PROJECT_ID>.iam.gserviceaccount
.com" \
    --mappings accessorCrn="<USER_CRN>",role="<PREFIX>-dladm-sa@<PROJECT
_ID>.iam.gserviceaccount.com"
```

6. Create the Data Lake using the following command. Replace the following with actual values:
   - <PREFIX> - Same as used earlier
   - <PROJECT_ID> - Same as used earlier

```
cdp datalake create-gcp-datalake --datalake-name "<PREFIX>-cdp-dl" \
    --environment-name "<PREFIX>-cdp-env" \
    --cloud-provider-configuration "serviceAccountEmail=<PREFIX>-idb-sa
@<PROJECT_ID>.iam.gserviceaccount.com",storageLocation="gs://<PREFIX>-cdp-
data"
```

7. Once your environment is created, its status will change to Available and the Data Lake status will change to Running.

Once your environment is running, you can start creating Data Hub clusters.

Cloudera Public Cloud

Getting Started in CDP Public Cloud

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Getting started as an admin 4

Getting started as a user 5

Creating and managing Cloudera deployments 7
- What is a Cloudera deployment 7
- Cloudera deployment patterns 8
- Cloudera deployment pattern definitions 9

Deploy Cloudera using Terraform 12
- Prerequisites 12
- Deploy Cloudera 13
- Cloud provider requirements 16
- Prerequisites for deploying Cloudera 17

Terraform module for deploying Cloudera 18

Getting started as an admin

Refer to this section if you are a Cloudera admin who is trying to get started in Cloudera.

Accessing Cloudera for the first time

Access the Cloudera web interface at https://console.cdp.cloudera.com (if your Cloudera account was created in the Cloudera Control Plane region us-west-1) or https://console.<control-plane-region>.cdp.cloudera.com (if your Cloudera account was created in any other Cloudera Control Plane region). When logging in for the first time, log in by using your MyCloudera credentials.

Trying out a Cloudera quick start

If you would like to quickly set up Cloudera for evaluation purposes, you can use our AWS Quick Start, Azure Quick Start, or Google Cloud Quick Start.

Reviewing cloud provider requirements

You should review the cloud provider requirements for setting up a Cloudera environment:

- AWS: AWS Requirements, AWS Reference Network Architecture, and AWS environment validation tool.
- Azure: Azure Requirements
- GCP: Google Cloud Requirements

Installing CDP CLI

You can install and configure CDP CLI. See CLI client setup.

Setting up Identity provider

In order to add users from your organization to Cloudera, set up your identity provider. For instructions, refer to Onboarding users.

Registering an environment

Register an environment for your organization. An environment determines the specific cloud provider region and virtual network in which resources can be provisioned, and includes the credential that should be used to access the cloud provider account. For instructions, refer to AWS environments, Azure environments, or Google Cloud environments documentation.

Assigning users or groups to your environment

Once your environment is up and running, you should assign users or groups to the environment and then perform user sync. For instructions, refer to Enabling admin and user access to environments.

Onboarding users and groups for cloud storage

The minimal setup for cloud storage defined in environment prerequisites spins up a Cloudera environment and Data Lake with no end user access to cloud storage. Adding users and groups to a Cloudera cluster involves ensuring they are properly mapped to IAM roles to access cloud storage. For instructions, refer to:

- Onboarding Cloudera users and groups for AWS cloud storage
- Onboarding Cloudera users and groups for Azure cloud storage
- Onboarding Cloudera users and groups for GCP cloud storage

Setting up Ranger authorization for your Data Lake

Once your environment is up and running, you should log in to Ranger and create policies for access to specific tables and databases. You can either log in to Hive first and create resources and then create policies for them in Ranger, or you can create Ranger policies in advance.

For instructions on how to access your Data Lake cluster, refer to Accessing Data Lake services. For instructions on how to set up authorization in Ranger, refer to Using Ranger to provide authorization documentation.

Provisioning compute resources

After performing these steps, you are set to start provisioning compute resources (Cloudera Data Hub clusters, Cloudera Data Warehouse, and so on). For more information, refer to the following documentation:

- Cloudera Data Engineering
- Cloudera DataFlow
- Cloudera Data Hub
- Cloudera Data Warehouse
- Cloudera AI
- Cloudera Operational Database

Registering your existing clusters

You can optionally register your existing CDH and HDP clusters in Cloudera if you would like to generate a workload, data movement, and compute capacity plan and replicate your data. For instructions, refer to Managing classic clusters.

Getting started as a user

Refer to this section if you are a non-admin Cloudera user who is trying to get started in Cloudera.

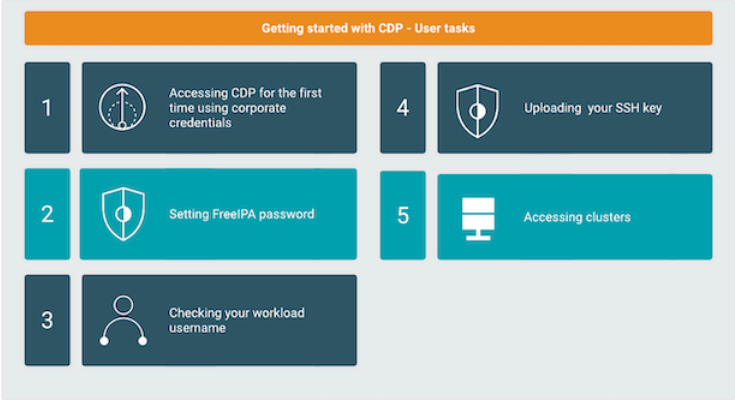

Accessing Cloudera for the first time

Access the Cloudera web interface at https://console.cdp.cloudera.com and log in by using your corporate credentials or other credentials that you received from your Cloudera administrator.

Setting workload password

If you are planning to access certain resources such as:

- Access clusters via SSH
- Connect to clusters via JDBC or ODBC
- Access Data Analytics Studio (DAS)
- Access Cloudera AI workspaces

you must access these by using your workload password. Initially, you must set your workload password, and then you need to reset it each time a new environment is shared with you. For more information about when and how to set and reset your workload password, refer to Accessing non-SSO interfaces using IPA credentials.

Checking your workload user name

If you are planning to access certain resources such as:

- Access clusters via SSH
- Connect to clusters via JDBC or ODBC
- Access Data Analytics Studio (DAS)
- Access Cloudera AI workbenches

you must access these by using your workload user name. To check your workload user name, navigate to the Cloudera Management Console > User Management > Users, find your user and check your Workload User Name.

Uploading SSH key

As an alternative for using workload password for SSHing to workload clusters, you can also upload your SSH public key to Cloudera and use the matching SSH private key for access. For more information, refer to Managing SSH keys.

Accessing resources

Your Cloudera administrator decided which Cloudera resources are available to you. You can access these resources from the Cloudera web interface. For more information, refer to the following documentation:

- Data Hub
- Cloudera Data Engineering

- Cloudera DataFlow
- Cloudera Data Hub
- Cloudera Data Warehouse
- Cloudera AI
- Cloudera Operational Database

Creating and managing Cloudera deployments

In this topic, we provide an overview of best practices for deploying Cloudera and demonstrate how to create and manage Cloudera deployments through a simple yet powerful Terraform framework.

If you are looking for a high-level overview of best practices for setting up Cloudera by using our standardized Terraform-based Cloudera deployment patterns, continue reading this article.

Note:

Creating new Cloudera deployments, adding data services, and managing the platform is also possible via the Cloudera web interface and CDP CLI. These options enable you to create customized deployments with a high degree of flexibility.

Note:

This guide currently covers deploying Cloudera on AWS and Azure only. For instructions on how to quickly deploy Cloudera on GCP, refer to Cloudera quickstarts.

What is a Cloudera deployment

A Cloudera deployment is a set of Cloudera management services and data services including related cloud provider resources that exist in your AWS, Azure, or GCP account. It is a combination of the cloud infrastructure that may span multiple cloud providers and regions, and the Cloudera platform that abstracts this underlying cloud provider infrastructure into an integrated, unified, logical data platform layer.

Each Cloudera deployment consists of Cloudera services and the underlying cloud provider resources.

In order for Cloudera to be deployed, a set of cloud provider prerequisites needs to be provided first, including a virtual network and subnets, storage accounts, and access roles/identities and policies. These cloud provider prerequisites are typically customer-managed and exist in the cloud provider account independently of Cloudera services. As such, they may be shared with other, non-Cloudera cloud services.

Once the cloud provider prerequisites are present, a Cloudera environment can be deployed in the virtual network. Once your Cloudera environment is up and running, your core Cloudera and cloud provider infrastructure is in place and you can start creating Cloudera Data Hub clusters and data services in order to run workloads. When these services are created, additional cloud provider resources such as VM instances, security groups, and load balancers are deployed in your cloud account. For each service, you can select which subnets of the underlying virtual network and what storage locations within your specified storage accounts they should use.

These three high-level deployment steps are described in the following diagram:

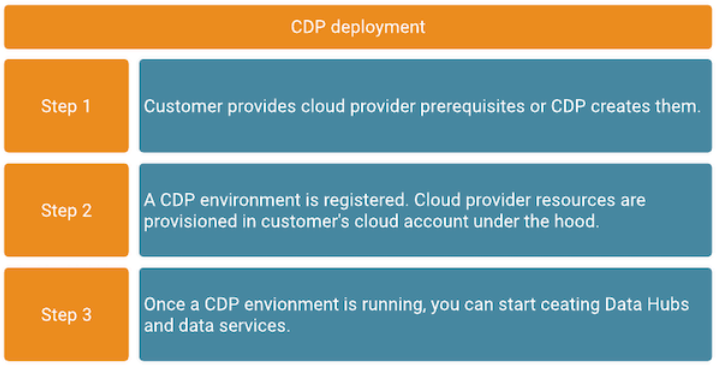

Cloudera deployment can be performed by using either Cloudera web interface or Cloudera CLI, or Terraform-based Cloudera deployment patterns. Continue reading to learn about deploying Cloudera using Terraform.

Note: As a best practice, cloud provider prerequisites (such as a VPC/VNet and subnets) should be created and managed outside of Cloudera. The Terraform quickstart module provided creates all these cloud provider prerequisites, but in case you would like to use an existing AWS VPC or Azure VNet and subnets, you can achieve this by providing a few additional optional parameters.

Note: If you would like to understand the details of the automation tooling provided here or are looking for more flexibility for your automated Cloudera deployments, refer to Terraform module for deploying Cloudera.

Related Information

Cloudera's Shared Responsibility Model

Cloudera deployment patterns

To simplify the task of defining and creating Cloudera deployments, we provide and describe a set of predefined target architectures recommended by Cloudera. These target architectures are called deployment patterns.

In Cloudera's Terraform framework, each pattern is represented by a deployment template that allows you to quickly instantiate one of the reference deployments. The templates can be used as a starting point and modified according to your needs. You can learn more about the recommended configurations of Cloudera Public Cloud from the documentation of our end-to-end deployment patterns as well as our network reference architectures for AWS and Azure.

Currently, we provide templates that represent the following deployment patterns, each matching a different use case:

Private	Production-like setup fully deployed on private subnets without public IPs or direct outbound internet access. Demonstrates a possible production deployment with typical network security features enabled. **Note:** Since private subnets have no internet connectivity by default, such a setup of Cloudera would not function out of the box, unless additional network components such as Internet Gateways or NAT Gateways are present. For convenience, we deploy these additional components by default. To turn off this behavior (for example when deploying to an existing private network), you can set the optional parameter private_network_extensions=false.
Semi-private	Production-like setup with access over the public internet to the user interfaces and API endpoints only. It serves as a reference for production deployments without the need for configuring VPNs, jump hosts and user-defined routing for outbound (egress) traffic

Public	Simple setup with access over public internet to all endpoints and with a minimal footprint. It can be used for quick testing, tutorial, demonstration, or simply to understand the internal workings of Cloudera Public Cloud. This setup is not secure enough for production, but can be used for proof of concept.

Note:

A real-life production Cloudera deployment often differs from the patterns described here. The examples that we provide intend to simplify the initial creation of a Cloudera deployment and serve as a reference for customized, customer-owned templates. While based on observed patterns and best practices, these templates are provided as-is and maintained by the Cloudera field community. If you plan to set up Cloudera for production, we assume that you customize the provided examples to match your IT networking and security guidelines.

Cloudera deployment pattern definitions

Deployment patterns are predefined architectures recommended by Cloudera that simplify the task of defining and creating Cloudera deployments. There are many options available for deploying Cloudera, but as a best practice, Cloudera recommends that you use one of the following three deployment patterns: private, semi-private, or public.

These patterns are based on the identically named network reference architectures and extend them, by incorporating Cloudera's recommended configuration for deploying Cloudera in multiple availability zones, selecting the Data Lake scale, configuring storage access policies and setting up fine-grained access control.

As can be expected, each of these deployment patterns brings a unique trade-off among various aspects, such as ease of setup, security provided, workloads supported, and so on. Read the following content to understand what specific networking, IAM, and storage cloud provider configurations, and Cloudera configurations are applied as part of the supported deployment patterns.

Cloud provider prerequisites

This section summarizes the networking, IAM, and storage cloud provider configurations that are made when Cloudera is deployed based on one of the deployment patterns.

Networking

For AWS

	Private	Semi-private	Public
VPC	A new VPC is provisioned in your cloud provider account.	A new VPC is provisioned in your cloud provider account.	A new VPC is provisioned in your cloud provider account.
Subnets	1x /18 public subnet for network access (when using private_network_extension=true) 3x /18 private subnets (for cluster nodes)	3x /19 public subnets (for load balancers) 3x /19 subnets (for cluster nodes)	3x /18 public subnets (for load balancers and cluster nodes)
Public IPs	1 Elastic IP is allocated (when using private_network_extension=true)	3 Elastic IPs are allocated (for load balancers)	All deployed nodes have public IPs
Egress traffic	One AWS Internet Gateway and one AWS Public and Private NAT Gateway are created (when using private_network_extension=true)	One AWS Internet Gateway, three AWS Public and Private NAT Gateways (1 per public subnet)	One AWS Internet Gateway
Ingress traffic	Public Load Balancer (when using private_network_extension=true)	Public Load Balancer	Public Load Balancer

Security groups	2 Security Groups (Rules set up based on user input/ configuration)	2 Security Groups (Rules set up based on user input/ configuration)	2 Security Groups (Rules set up based on user input/ configuration)

For Azure

	Private	Semi-private	Public
Resource Group	A single new resource group is created.	A single new resource group is created.	A single new resource group is created.
VNet	A new VNet is provisioned in your cloud provider account.	A new VNet is provisioned in your cloud provider account.	A new VNet is provisioned in your cloud provider account.
Subnets	3x /18 subnets (for load balancers and cluster nodes)	1x /18 subnets (for load balancers)) 3x /18 subnets (for cluster nodes)	3x /18 subnets
Public IPs	No public IPs are assigned.	Load balancers have public hostnames. No public IPs are assigned to cluster nodes	All nodes deployed have public IPs.
Egress traffic	Managed by Azure	Managed by Azure	Managed by Azure
Ingress traffic	Managed by Azure	Managed by Azure	Managed by Azure
Security groups	2 Network Security Groups (Rules set up based on user input/configuration)	2 Network Security Groups (Rules set up based on user input/configuration)	2 Network Security Groups (Rules set up based on user input/configuration)

Identity and access management

For AWS

	Private	Semi-private	Public
Federated access	Cross-account policy	Cross-account policy	Cross-account policy
Storage access	IAM roles, policies, and instance profiles	IAM roles, policies, and instance profiles	IAM roles, policies, and instance profiles

For Azure

	Private	Semi-private	Public
Federated access	Azure service principal	Azure service principal	Azure service principal
Storage access	User-assigned managed identities and role assignments using built-in roles	User-assigned managed identities and role assignments using built-in roles	User-assigned managed identities and role assignments using built-in roles

Storage

For AWS

	Private	Semi-private	Public
S3 buckets	3 base locations	3 base locations	3 base locations

For Azure

	Private	Semi-private	Public

Azure storage accounts	One storage account for data, logs and backup (3 containers) One additional storage account for locally caching VM images	One storage account for data, logs and backup (3 containers) One additional storage account for locally caching VM images	One storage account for data, logs and backup (3 containers) One additional storage account for locally caching VM images

Environment and Data Lake

This section summarizes Cloudera networking, security, and other configurations that are made when Cloudera is deployed based on one of the deployment patterns.

Cloudera networking setup

For AWS

	Private	Semi-private	Public
Communication with Cloudera Control Plane	Reverse HTTPS tunnel (Cluster Connectivity Manager), no private link	Reverse HTTPS tunnel (Cluster Connectivity Manager), no private link	Reverse HTTPS tunnel (Cluster Connectivity Manager), no private link
Load balancer and node placement	2 load balancers are placed in private subnets. All cluster nodes are placed in private subnets.	2 load balancers are placed in the external subnets and all cluster nodes are placed in the internal subnets.	2 load balancers and all cluster nodes are placed in public subnets.
Multiple availability zones	Environment and Data Lake clusters are spread across three availability zones.	Environment and Data Lake clusters are spread across three availability zones.	Environment cluster is spread across three availability zones. Basic Data Lake cluster is deployed in one availability zone.
Ports open (in the external and internal network)	Ports 22 and 443 are open by default.	Ports 22 and 443 are open by default.	Ports 22 and 443 are open by default.

For Azure

	Private	Semi-private	Public
Communication with Cloudera Control Plane	Reverse HTTPS tunnel (Cluster Connectivity Manager), no private link	Reverse HTTPS tunnel (Cluster Connectivity Manager), no private link	Reverse HTTPS tunnel (Cluster Connectivity Manager), no private link
Load balancer and node placement	Azure Standard Network Load Balancers are created by the data lake	Azure Standard Network Load Balancers are created by the data lake	Azure Standard Network Load Balancers are created by the data lake
Availability zones	Zone placement is managed by Azure	Zone placement is managed by Azure	Zone placement is managed by Azure
Ports open (in the external and internal network)	Ports 22 and 443 are open by default.	Ports 22 and 443 are open by default.	Ports 22 and 443 are open by default.

Cloudera security setup

For AWS

	Private	Semi-private	Public
Fine-grained storage access control (RAZ)	Enabled	Enabled	Enabled
SSH access to cluster hosts	Root access is possible with a customer-provided keypair.	Root access is possible with a customer-provided keypair.	Root access is possible with a customer-provided keypair.

For Azure

	Private	Semi-private	Public
Fine-grained storage access control (RAZ)	Enabled	Enabled	Enabled
SSH access to cluster hosts	Root access is possible with a customer-provided keypair.	Root access is possible with a customer-provided keypair.	Root access is possible with a customer-provided keypair.

Cloudera versions and details

For AWS

	Private	Semi-private	Public
Data Lake Runtime version	Latest	Latest	Latest
Data Lake shape	Medium Duty	Medium Duty	Light Duty

For Azure

	Private	Semi-private	Public
Data Lake Runtime version	Latest	Latest	Latest
Data Lake shape	Medium Duty	Medium Duty	Light Duty

Related Information

Overview of AWS resources used by Cloudera

Overview of Azure resources used by Cloudera

Deploy Cloudera using Terraform

This guide demonstrates how to deploy Cloudera on AWS or Azure by using one of the Cloudera deployment templates.

The templates use Terraform, an open source Infrastructure as Code (IaC) software tool for defining and managing cloud or data center infrastructure. You interface the templates via a simple configuration file residing in a GitHub repository.

For an overview of best practices for deploying Cloudera, refer to Creating and managing Cloudera deployments.

Note: As a best practice, cloud provider prerequisites (such as a VPC/VNet and subnets) should be created and managed outside of Cloudera. The Terraform quickstart module provided creates all these cloud provider prerequisites, but in case you would like to use an existing AWS VPC or Azure VNet and subnets, you can achieve this by providing a few additional optional parameters.

Prerequisites

Prior to deploying Cloudera, you should make sure that your cloud account meets the basic requirements and that you've installed a few prerequisites.

To meet these requirements and install the prerequisites, refer to the following documentation:

- Cloud provider requirements
- Prerequisites for deploying Cloudera

You should also familiarize yourself with the background information about CDP deployment patterns and deployment pattern definitions described in Creating and managing Cloudera deployments.

Next, you can follow the instructions below for deploying CDP.

Deploy Cloudera

Setting up a Cloudera deployment involves cloning a GitHub repository, editing the configuration, and running Terraform commands.

Step 1: Clone the repository

The cdp-tf-quickstarts repository contains Terraform resource files to quickly deploy Cloudera Public Cloud and associated pre-requisite cloud resources. It uses the Cloudera Terraform Modules provided by Cloudera to do this.

Clone this repository and navigate to the directory with the cloned repository:

```
git clone https://github.com/cloudera-labs/cdp-tf-quickstarts.git
cd cdp-tf-quickstarts
```

Step 2: Edit the configuration file for the required cloud provider

In the cloned repository, change to the required cloud provider directory. Currently AWS and Azure are available.

Next, edit the input variables in the configuration file as required:

For AWS

```
cd aws
mv terraform.tfvars.template terraform.tfvars
vi terraform.tfvars
```

For Azure

```
cd azure
mv terraform.tfvars.template terraform.tfvars
vi terraform.tfvars
```

Sample content of this file, with indicators of values to change are shown below. The variables are explained below the sample. You should review and update all the variables.

For AWS

```
# ------- Global settings -------
env_prefix = "<ENTER_VALUE>" # Required name prefix for cloud and Cloudera
 resources, e.g. cldr1

# ------- Cloud Settings -------
aws_region = "<ENTER_VALUE>" # Change this to specify Cloud Provider re
gion, e.g. eu-west-1

# ------- Cloudera Environment Deployment -------
deployment_template = "<ENTER_VALUE>"  # Specify the deployment pattern
below. Options are public, semi-private or private
```

For Azure

```
# ------- Global settings -------
```

```
env_prefix = "<ENTER_VALUE>" # Required name prefix for cloud and Cloudera
 resources, e.g. cldr1

# ------- Cloud Settings -------
azure_region = "<ENTER_VALUE>" # Change this to specify Cloud Provider
region, e.g. eastus
# ------- Cloudera Environment Deployment -------
deployment_template = "<ENTER_VALUE>"  # Specify the deployment pattern
below. Options are public, semi-private or private
```

As an outcome of this step, your configuration file should look similar to the following:

For AWS

```
# ------- Global settings -------
env_prefix = "TEST-ENV" # Required name prefix for cloud and CDP resources
, e.g. cldr1

# ------- Cloud Settings -------
aws_region = "EU-WEST-1" # Change this to specify Cloud Provider region,
 e.g. eu-west-1

# ------- Cloudera Environment Deployment -------
deployment_template = "PUBLIC"  # Specify the deployment pattern below. O
ptions are public, semi-private or private
```

For Azure

```
# ------- Global settings -------
env_prefix = "TEST-ENV" # Required name prefix for cloud and CDP resources
, e.g. cldr1

# ------- Cloud Settings -------
azure_region = "WESTEUROPE" # Change this to specify Cloud Provider region
, e.g. eastus

# ------- Cloudera Environment Deployment -------
deployment_template = "PUBLIC"  # Specify the deployment pattern below. O
ptions are public, semi-private or private
```

The following tables explain the mandatory inputs that need to be provided in the configuration file.

Table 1: Mandatory inputs

For AWS

Input	Description	Default value
env_prefix	A string prefix that will be used to name the cloud provider and Cloudera resources created.	Not set
aws_region	The AWS region in which the cloud prerequisites and Cloudera will be deployed. For example, eu-west-1. For a list of supported AWS regions, see Supported AWS regions.	Not set
deployment_template	The selected deployment pattern. Values allowed: private, semi-private and public.	public

For Azure

Input	Description	Default value
azure_region	The Azure region in which the cloud prerequisites and Cloudera will be deployed. For example, eastus. For a list of supported Azure regions, see Supported Azure regions.	Not set
env_prefix	A string prefix that will be used to name the cloud provider and Cloudera resources created.	Not set
deployment_template	The selected deployment pattern. Values allowed: private, semi-private and public.	public

The following tables explain the optional inputs. The optional inputs can optionally be added to the configuration file. While the mandatory inputs are present in the configuration file and only their values need to be provided, the optional inputs should be added manually.

Table 2: Optional inputs

For AWS

Input	Description	Default value
aws_key_pair	The name of an AWS keypair that exists in your account in the selected region.	Not set
ingress_extra_cidrs_and_ports	Inbound access to the UI and API endpoints of your deployment will be allowed from the CIDRs (IP ranges) and ports specified here. Enter your machine's public IP here, with ports 443 and 22. If unsure, you can check your public IP address here.	CIDRs are not set. Ports are set to 443, 22 by default.
create_vpc	Flag to specify if the VPC should be created	true
cdp_vpc_id	VPC ID for Cloudera environment. Required if create_vpc is false	Empty string
cdp_public_subnet_ids	List of public subnet ids. Required if crea te_vpc is false	Empty list
cdp_private_subnet_ids	List of private subnet ids. Required if crea te_vpc is false	Empty list
private_network_extensions	Enable creation of resources for connectivity to Cloudera Control Plane (public subnet and NAT Gateway) for Private Deployment. Only relevant for private deployment template	true

For Azure

Input	Description	Default value
public_key_text	An SSH public key string to be used for the nodes of the Cloudera environment.	Not set

ingress_extra_cidrs_and_ports	Inbound access to the UI and API endpoints of your deployment will be allowed from the CIDRs (IP ranges) and ports specified here. Enter your machine's public IP here, with ports 443 and 22. If unsure, you can check your public IP address here.	CIDRs are not set. Ports are set to 443, 22 by default.
create_vnet	Flag to specify if the VNet should be created	true
cdp_resourcegroup_name	Preexisting Azure resource group for Cloudera environment. Required if create_vnet is false	Empty string
cdp_vnet_name	VNet name for Cloudera environment. Required if create_vnet is false	Empty string
cdp_subnet_names	List of subnet names for Cloudera resources. Required if create_vnet is false	Empty list
cdp_gw_subnet_ids	List of subnet names for Cloudera Gateway. Required if create_vnet is false	Empty list

Step 3: Launch the deployment

Run the Terraform commands to validate the configuration and launch the deployment with the following commands:

```
terraform init
terraform apply
```

Terraform will show a plan with the list of cloud provider and Cloudera resources that will be created.

When you are prompted, type yes to tell Terraform to perform the deployment. Typically, this will take about 60 minutes. Once the deployment is complete, Cloudera will print output similar to the following:

```
Apply complete! Resources: 46 added, 0 changed, 0 destroyed.
```

You can navigate to the Cloudera web interface at https://cdp.cloudera.com/ and see your deployment progressing. Once the deployment completes, you can create Cloudera Data Hub clusters and data services.

Clean up the Cloudera environment and infrastructure

If you no longer need the infrastructure that's provisioned by Terraform, run the following command to remove the deployment infrastructure and terminate all resources:

```
terraform destroy
```

Cloud provider requirements

Review the requirements related to the AWS account that you would like to use with Cloudera.

AWS account

To follow this guide, you need to have access to an AWS account. In this guide, we assume that you have a newly created account or a sub-account with default settings and no network restrictions (custom routes to the Internet) or policy restrictions (AWS Organizations policies or Service Control Policies (SCPs)) in place. SCPs configured on the parent AWS Organization of your AWS account may impact certain steps described in this guide and may require that you follow a custom deployment path.

You also need the following account-level AWS settings:

- An AWS role that has permissions to create IAM objects (cross-account role and policy, IAM roles and policies, S3 buckets). You will also need to create credentials for your IAM user role. You will need these in the next section for configuring the Terraform Provider for AWS on your machine. See AWS security credentials.
- Select a supported AWS region for your deployment. See Supported AWS regions.
- A vCPU quota of at least 200 cores. You may need a higher limit for larger deployments. You can check your current vCPU quota under the name Running On-Demand Standard (A, C, D, H, I, M, R, T, Z) instances. Make sure that the quota value is 200 or larger. See the AWS documentation for requesting an EC2 vCPU limit increase.
- An elastic IP quota of at least 5 elastic IPs (for the public and semi-private patterns). The recommended quota is 10 elastic IPs.

Azure account

To follow this guide, you need to have access to an Azure account. In this guide, we assume that you have a newly created account or a sub-account with default settings and no network restrictions (custom routes to the Internet) or policy restrictions (Azure Organizations policies or Service Control Policies (SCPs)) in place. SCPs configured on the parent Azure Organization of your Azure account may impact certain steps described in this guide and may require that you follow a custom deployment path.

You also need the following tenant and subscription-level Azure permissions and settings:

- You need to have write permissions for Azure AD in order to create the Azure service principal (App registration).
- Your user needs to have Contributor privileges at least at the scope of the Azure resource group in which you will deploy Cloudera; That is, your user needs to have permissions to create managed identities, grant role assignments at the scope of the resource group, and create VNet/subnets and storage accounts.
- Select a supported Azure region for your deployment. See Supported Azure regions.
- A Total Regional vCPU quota of at least 200 cores. You may need a higher limit for larger deployments. For requesting a compute quota increase, see the Azure documentation. Make sure that the Standard DSv3 Family vCPUs quota is also 200 cores or larger.
- A Public IP Addresses quota of at least 5 public IP addresses (for the public and semi-private patterns). The recommended quota is 10 IP addresses.
- Make sure that all services required by Cloudera are available in your selected Azure region. You may need to request that Azure Support whitelists a particular service (such as Azure Database for PostgreSQL) for your subscription in your selected region. See Overview of Azure resources used by Cloudera.

Related Information

Overview of AWS resources used by Cloudera

Prerequisites for deploying Cloudera

To set up Cloudera via deployment automation using this guide, the following prerequisites must be installed and configured in your local environment:

- Terraform version 1.3 or newer
- Terraform Provider for AWS or Azure
- Terraform Provider for Cloudera

Install Terraform

Install Terraform version 1.3 or newer. See installation instructions in the Terraform installation guide.

Configure Terraform Provider for AWS or Azure

For AWS examples see Build Infrastructure | Terraform | HashiCorp Developer.

For Azure examples see Build Infrastructure - Terraform Azure Example | Terraform | HashiCorp Developer.

Configure Terraform Provider for Cloudera

Configure Terraform Provider for Cloudera by downloading or creating a Cloudera configuration file. You can find the required steps for Generating an API access key and Configuring Cloudera client in our documentation.

Terraform module for deploying Cloudera

The Terraform Modules for Cloudera Prerequisites on AWS and Azure contain Terraform resource files and example variable definition files for creating the prerequisite cloud provider resources required for deploying Cloudera. These modules use the official Terraform Providers for AWS or Azure, both maintained by Hashicorp. They include a VPC/VNet configured with public and private subnets according to the network deployment pattern specified, data and log buckets/containers for the Cloudera environment, and a number of AWS IAM roles and policies or Azure managed identities to enable fine-grained permissions for access to the Cloudera Control Plane and AWS/Azure services.

Furthermore, the Terraform Module for Cloudera Deployment is used to create a Cloudera credential and deploy a Cloudera environment and a Data Lake.

The aforementioned modules support the network deployment patterns described in Cloudera deployment pattern definitions below and are coupled with the Cloudera Quickstart Terraform Module that we provide for simplifying end-to-end setup including both the cloud prerequisites and the Cloudera services.

The following diagram illustrates the hierarchy of modules and providers used by the onboarding automation tooling (AWS is used as an example):

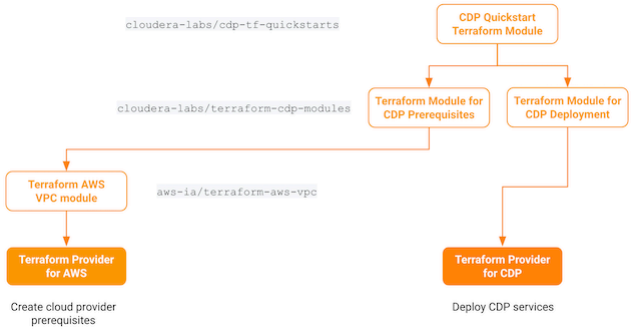

In our Deploy Cloudera using Terraform onboarding guides, we use these modules to quickly deploy Cloudera.

..

Migrating Hive workloads to Iceberg

Date published: 2023-01-16
Date modified: 2023-01-17

CLOUDERA

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Migrating Hive tables to Iceberg tables.....4
Use cases for migrating to Iceberg.....4
In-place migration from Hive.....4
Prerequisites.....5
Migrating a Hive table to Iceberg.....6
In-place migration from Spark.....7
Prerequisites and limitations for using Iceberg.....7
Importing and migrating Iceberg table in Spark 3.....8
Importing and migrating Iceberg table format v2.....9
Best practices for Iceberg in Cloudera.....10

Migrating Hive tables to Iceberg tables

Using Iceberg tables facilitates multi-cloud open lakehouse implementations. You can move Iceberg-based workloads in Cloudera across deployment environments on AWS and Azure. You can migrate existing external Hive tables from Hive to Iceberg in Cloudera Data Warehouse or from Spark to Iceberg in Cloudera Data Engineering.

Cloudera has chosen Apache Iceberg as the foundation for an open lakehouse in Cloudera. Any compute engine can insert, update, and delete data in Iceberg tables. Any compute engine can read Iceberg tables.

The following Cloudera data services support Iceberg for performing multi-function analytics for the open lakehouse with Cloudera SDX shared security and governance:

- Cloudera Data Warehouse: Batch ETL, SQL and BI analytic workloads, row-level database operations like updates, deletes, and merge
- Cloudera Data Engineering: Batch ETL, row-level database operations, table maintenance
- Cloudera AI: Data Science through Python, R, and other languages, ML model training and inferencing, table maintenance
- Cloudera DataFlow: Nifi streaming ingestion
- Cloudera Streaming: Unified streaming ingestion with SQL

Use cases for migrating to Iceberg

A key use case for migrating Hive tables to Iceberg is the elimination of data segmentation, or data silos, and duplication. Removing these silos facilitates the tedious preparation, curation, cleansing, and moving of data before you can get any meaningful insights from your data. This use case and others are summarized.

The following list includes other use cases for migrating Hive tables to Iceberg:

- End-to-end, multi-function analytics

  Cloudera provides an entire lifecycle for lower cost of ownership and architectural simplicity.
- Regulatory compliance

  Improved visibility provided by quick insights into your data to lower risk and improve infosec management.
- Data protection and recovery

  Data managed by Cloudera services are protected by Cloudera Shared Data Experience (SDX), an integrated set of security and governance technologies.
- Compliance with General Data Protection Regulator/California Consumer Privacy Act (GDPR/CCPA)

  Iceberg supports ACID-compliance with row-level updates and deletes.

In-place migration from Hive

If you are looking for an efficient way to Hive migrate Hive tables to Iceberg, you came to the right place. An overview of using Cloudera Data Warehouse prepares you to convert Apache Hive external tables to Apache Iceberg with no downtime. You learn the advantages of moving Hive tables to Iceberg for implementing an open lakehouse.

You can accelerate data ingestion, curation, and consumption at petabyte scale by migrating from Hive to Iceberg.

Hive-to-Iceberg table migration is fast because the migration does not regenerate data files. Just metadata is rewritten. Cloudera Data Warehouse provides a simple API for migrating Hive tables to Iceberg to simplify adoption of Iceberg. An ALTER TABLE command sets the storage handler to convert the data without regeneration or remigration.

Using CDW, you can take advantage of schema evolution and partition evolution features. For more information about using Iceberg in CDW, see "Using Apache Iceberg". Newly generated metadata is built on top of existing Hive external tables and points to source data files as shown in the following diagram:

Migration using ALTER TABLE (recommended)

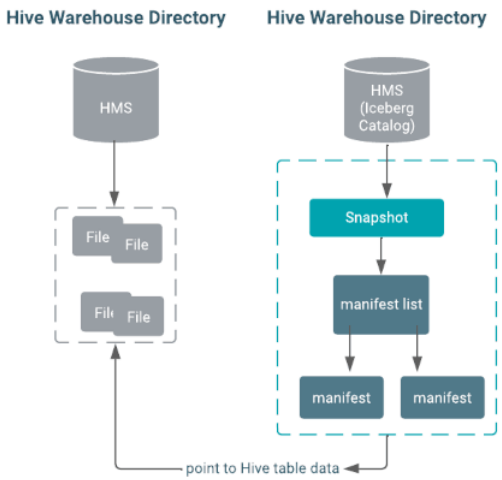

You can also run CREATE TABLE AS SELECT to migrate a Hive table to Iceberg; however, this approach reads data from the source and regenerates data in Iceberg:

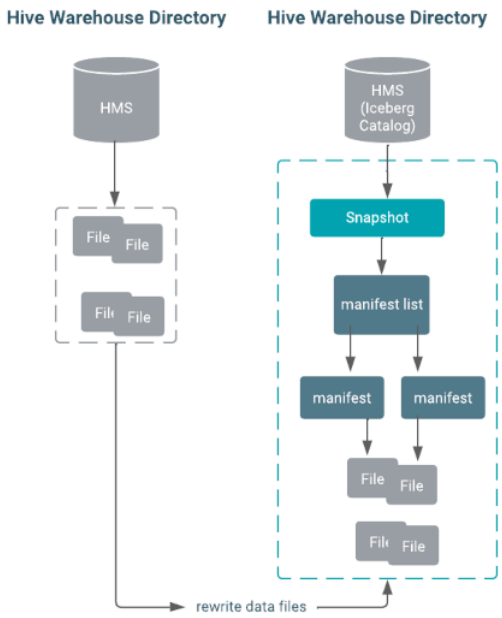

You can read and write files in following types of files from Hive:

- Avro
- Optimized Row Columnar (ORC)
- Parquet

Prerequisites

You need to set up your environment and meet Data Lake prerequisites for querying Iceberg tables in Cloudera. You learn which query editors are supported and which roles are required.

Note: Do not drop, move, or change the old table during a migration operation. Doing so will delete the data files of the old and new tables.

The following list covers prerequisites for using Iceberg.

- You are using the Cloudera Data Warehouse.
- You must have access to an activated environment.

-
- One of the following query editors are required to query Iceberg tables:
  - Hue (recommended)
  - A JDBC client
  - The Impala shell for remote users
- You must have the required role: DWUser.
- You must obtain permission to run SQL queries from the Env Admin, who must add you to a Hadoop SQL policy.
- You must use the HadoopFileIO. S3FileIO is not supported. URLs in metadata starting with s3://... cause query failure.

Migrating a Hive table to Iceberg

You see how to use a simple ALTER TABLE statement to migrate an external Hive table to an Iceberg table. You see how to configure table input and output by setting table properties.

About this task

When you migrate an external Hive table to Iceberg, Hive makes the following changes:

- Converts the storage_handler, serde, inputformat and outputformat properties of the table in HMS to use the Iceberg specific classes.
- Reads the footers of the existing data files and generates the necessary Iceberg metadata files based on the footers.
- Commits all the data files to the Iceberg table in a single commit.

Note: To prevent loss of new and old table data during migration of a table to Iceberg, do not drop or move the old table during migration. Exception: If you set the table property 'external.table.purge'='FALSE', no data loss occurs when you drop the table.

Before you begin

- You must meet the prerequisites for using Iceberg mentioned earlier.
- .

  Restriction: Migrating an Impala table to Iceberg is not supported in this release.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, in the Overview page, locate your Virtual Warehouse, and click Hue.

   Instead of using Hue, you can connect over JDBC to the Virtual Warehouse, and run the query.
3. Enter a query to use a database.
   For example:

   ```
   USE mydb;
   ```

4. Enter a Hive query to migrate an existing external Hive table to an Iceberg v2 table.
   For example:

   ```
   ALTER TABLE tbl
   SET TBLPROPERTIES ('storage_handler'='org.apache.iceberg.mr.hive.HiveIc
   ebergStorageHandler',
    'format-version' = '2');
   ```

   Do not drop the table as explained above unless you set the 'external.table.purge' table property to false.
5. Click ▶ to run the queries.
   An Iceberg V2 table is created. The Hive table remains intact.

In-place migration from Spark

In Cloudera Data Engineering, you can use Spark SQL to migrate Hive tables to Iceberg. You can convert Apache Hive external tables to Apache Iceberg with no downtime. Cloudera recommends moving Hive tables to Iceberg for implementing an open lakehouse.

You use one of the following, similar procedures to import and migrate Hive tables to Iceberg:

- Importing and migrating Iceberg table in Spark 3

  A backup table is created, but does not incur the overhead of moving the physical location of the table on the object store.
- Importing and migrating Iceberg table format v2

  The Iceberg merge on read operation is used.

By default, when you migrate a Hive external table to an Iceberg v2 table, the file is not rewritten. A write occurs to a new file. A read merges changes into the original file. The default merge on read tends to speed up the write and slow down the read. You can configure several other types of migration behavior if the default merge on read does not suit your use case.

For more information about using Iceberg in Cloudera Data Engineering, see Using Apache Iceberg in Cloudera Data Engineering.

Prerequisites and limitations for using Iceberg

Learn about the supported versions for Cloudera Data Engineering, Spark, and Data Lake to use with Apache Iceberg.

To use Apache Iceberg in Cloudera Data Engineering, you'll need the following prerequisites:

- Spark 3.2 or higher
- A compatible version of Data Lake as listed in Cloudera Data Engineering and Data Lake compatibility linked below
- Cloudera Data Engineering 1.16 or higher
- **Note:** Only AWS is supported in Cloudera Data Engineering 1.16 (which supports Iceberg 0.13)
- AWS or Azure is supported starting in Cloudera Data Engineering 1.17-h1 (which supports Iceberg 0.14)

Iceberg table format version 2

Iceberg table format version 2 (v2) is available starting in Iceberg 0.14. Iceberg table format v2 uses row-level UPDATE and DELETE operations that add deleted files to encoded rows that were deleted from existing data files. The DELETE, UPDATE, and MERGE operations function by writing delete files instead of rewriting the affected data files. Additionally, upon reading the data, the encoded deletes are applied to the affected rows that are read. This functionality is called merge-on-read.

To use Iceberg table format v2, you'll need the following prerequisites:

- Cloudera Data Engineering 1.17-h1 or higher
- Iceberg 0.14
- Spark 3.2 or higher

With Iceberg table format version 1 (v1), the above-mentioned operations are only supported with copy-on-write where data files are rewritten in their entirety when rows in the files are deleted. Merge-on-read is more efficient for writes, while copy-on-write is more efficient for reads.

Note: Unless otherwise indicated, the operations in the subsequent documentation apply to both v1 and v2 formats.

Importing and migrating Iceberg table in Spark 3

Importing or migrating tables are supported only on existing external Hive tables. When you import a table to Iceberg, the source and destination remain intact and independent. When you migrate a table, the existing Hive table is converted into an Iceberg table. You can use Spark SQL to import or migrate a Hive table to Iceberg.

Importing

Call the snapshot procedure to import a Hive table into Iceberg using a Spark 3 application.

```
spark.sql("CALL <catalog>.system.snapshot('<src>', '<dest>')")
```

Definitions:

- <src> is the qualified name of the Hive table
- <dest> is the qualified name of the Iceberg table to be created
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.

For example:

```
spark.sql("CALL spark_catalog.system.snapshot('hive_db.hive_tbl',
          'iceberg_db.iceberg_tbl')")
```

For information on compiling Spark 3 application with Iceberg libraries, see Iceberg library dependencies for Spark applications linked below.

Migrating

When you migrate a Hive table to Iceberg, a backup of the table, named <table_name>backup, is created.

Ensure that the TRANSLATED_TO_EXTERNAL property, that is located in TBLPROPERTIES, is set to false before migrating the table. This ensures that a table backup is created by renaming the table in Hive metastore (HMS) instead of moving the physical location of the table. Moving the physical location of the table would entail copying files in Amazon s3.

We recommend that you refrain from dropping the backup table, as doing so will invalidate the newly migrated table.

If you want to delete the backup table, set the following:

```
'external.table.purge'='FALSE'
```

Note: For Cloudera Data Engineering 1.19 and above, the property will be set automatically.

Deleting the backup table in the manner above will prevent underlying data from being deleted, therefore, only the table will be deleted from the metastore.

To undo the migration, drop the migrated table and restore the Hive table from the backup table by renaming it.

Call the migrate procedure to migrate a Hive table to Iceberg.

```
spark.sql("CALL <catalog>.system.migrate('<src>')")
```

Definitions:

- <src> is the qualified name of the Hive table
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.

For example:

```
spark.sql("CALL
      spark_catalog.system.migrate('hive_db.hive_tbl')")
```

Importing and migrating Iceberg table format v2

Importing or migrating Hive tables Iceberg table formats v2 are supported only on existing external Hive tables. When you import a table to Iceberg, the source and destination remain intact and independent. When you migrate a table, the existing Hive table is converted into an Iceberg table. You can use Spark SQL to import or migrate a Hive table to Iceberg.

Importing

Call the snapshot procedure to import a Hive table into Iceberg table format v2 using a Spark 3 application.

```
spark.sql("CALL <catalog>.system.snapshot(source_table => '<src>', table =>
'<dest>', properties => map('format-version', '2', 'write.delete.mode', '<de
lete-mode>', 'write.update.mode', '<update-mode>', 'write.merge.mode', '<mer
ge-mode>'))")
```

Definitions:

- <src> is the qualified name of the Hive table
- <dest> is the qualified name of the Iceberg table to be created
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.
- <delete-mode> <update-mode> and <merge-mode> are the modes that shall be used to perform the respective operation. If unspecified, they default to 'merge-on-read'

For example:

```
spark.sql("CALL spark_catalog.system.snapshot('hive_db.hive_tbl',
          'iceberg_db.iceberg_tbl')")
```

For information on compiling Spark 3 application with Iceberg libraries, see Iceberg library dependencies for Spark applications linked below.

Migrating

Call the migrate procedure to migrate a Hive table to Iceberg.

```
spark.sql("CALL <catalog>.system.migrate('<src>', map('format-version', '2',
 'write.delete.mode', '<delete-mode>', 'write.update.mode', '<update-mode>',
 'write.merge.mode', '<merge-mode>'))")
```

Definitions:

- <src> is the qualified name of the Hive table
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.
- <delete-mode> <update-mode> and <merge-mode> are the modes that shall be used to perform the respective operation. If unspecified, they default to 'merge-on-read'

For example:

```
spark.sql("CALL spark_catalog.system.migrate('hive_db.hive_tbl', map('format
-version', '2', 'write.delete.mode', 'merge-on-read', 'write.update.mode', '
merge-on-read', 'write.merge.mode', 'merge-on-read'))")
```

Upgrading Iceberg table format v1 to v2

To upgrade an Iceberg table format from v1 to v2, run an `ALTER TABLE` command as follows:

```
spark.sql("ALTER TABLE <table_name> SET TBLPROPERTIES('merge-on-read', '2')"
)
```

<delete-mode>,<update-mode>, and <merge-mode> can be specified as the modes that shall be used to perform the respective operation. If unspecified, they default to ‘merge-on-read'

Best practices for Iceberg in Cloudera

Based on large scale TPC-DS benchmark testing, performance testing and real-world experiences, Cloudera recommends several best practices when using Iceberg.

Follow the best practices listed below when using Iceberg:

- Use Iceberg as intended for analytics.

  The table format is designed to manage a large, slow-changing collection of files. For more information, see the Iceberg spec.
- Increase parallelism to handle large manifest list files in Spark.

  By default, the number of processors determines the preset value of the iceberg.worker.num-threads system property. Try increasing parallelism by setting the iceberg.worker.num-threads system property to a higher value to speed up query compilation.
- Reduce read amplification

  Monitor the growth of positional delta files, and perform timely compactions.
- Speed up drop table performance, preventing deletion of data files by using the following table properties:

  ```
  Set external.table.purge=false and gc.enabled=false
  ```
- Tune the following table properties to improve concurrency on writes and reduce commit failures: commit.retry .num-retries (default is 4), commit.retry.min-wait-ms (default is 100)
- Maintain a relatively small number of data files under the iceberg table/partition directory for efficient reads. To alleviate poor performance caused by too many small files, run the following queries:

  ```
  TRUNCATE TABLE target;
                 INSERT OVERWRITE TABLE target select * from target FOR SYST
  EM_VERSION AS OF <preTruncateSnapshotId>;
  ```
- To minimize the number of delete files and file handles and improve performance, ensure that the Spark write.distribution.mode table property value is “hash” (the default setting for Spark Iceberg 1.2.0 onwards).

..

Migration paths from HDP 3 to CDP for LLAP users

Date published: 2022-02-04
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Migration paths from HDP 3 to Cloudera for LLAP users..... 4
Migration paths for Hive users..... 4
Migration to Cloudera Private Cloud Base or Cloudera Public Cloud.....5
Migration to Cloudera Data Warehouse..... 5
Apache Tez processing of Hive jobs..... 6
Migration paths for Spark users.....6
Migration to Cloudera Private Cloud Base.....7
HWC changes from HDP to Cloudera.....8

Migration paths from HDP 3 to Cloudera for LLAP users

If you are running your Hive HDP 3.x workloads using LLAP (low-latency analytical processing), you need to decide on the best migration path to Cloudera without compromising on the performance offered by LLAP. Migration recommendations depend on a number of factors, such as your workload type and whether you have Hive or Spark users.

Migration paths for Hive users

If you are on HDP and executing Hive workloads in Hive LLAP mode and want to upgrade to Cloudera, you can follow the migration path that matches your use case.

Using LLAP is recommended for running complex ETL jobs that are unpredictable. LLAP is offered as part of Cloudera Data Warehouse and is not available in Cloudera Public Cloud and Cloudera Private Cloud Base. Using Hive on Tez (no LLAP) is recommended for running jobs that are scheduled and predictable. Hive on Tez is supported on Cloudera Public Cloud and Cloudera Private Cloud Base.

The following migration paths are recommended based on how predictable your jobs are and your workload type:

- Migrate to Cloudera Public Cloud (Cloudera Data Hub) or Cloudera Private Cloud Base — If your jobs are scheduled and predictable.
- Migrate to Cloudera Data Warehouse — If your jobs are unpredictable and if they could increase demand on compute resources.

The following diagram shows these recommended paths:

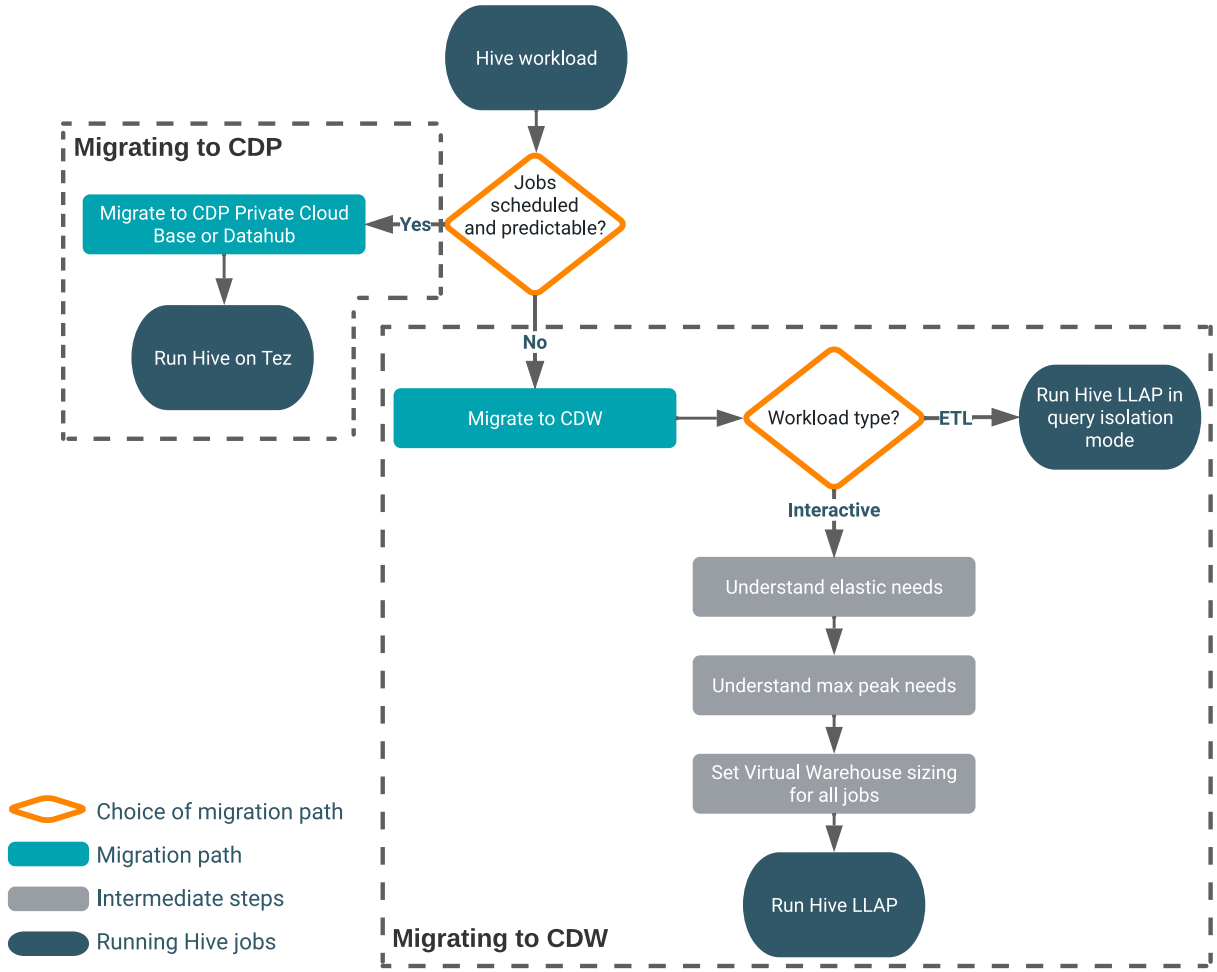

Migration to Cloudera Private Cloud Base or Cloudera Public Cloud

If your Hive jobs are scheduled, or otherwise considered predictable, consider migrating to Cloudera Private Cloud Base or Cloudera Public Cloud and running your Hive workloads on Hive on Tez.

You learn why you should consider the predictability of your jobs when choosing a migration path from HDP 3 to Cloudera. If you were using LLAP on HDP, a path to Cloudera can offer equivalent performance and might make sense cost-wise.

LLAP is not supported in Cloudera Private Cloud Base and Cloudera Public Cloud, but Hive-on-Tez performs well for jobs that are not subjected to unexpected big spikes in demand that would require elasticity, such as rapid service scaling. Predictable jobs do not require the immediate deployment of extra resources to respond to unexpected traffic to your cluster. You do not need to shut down resources when traffic drops suddenly.

Related Information

Apache Tez execution engine for Apache Hive

Migration to Cloudera Data Warehouse

If your Hive jobs are unpredictable, consider migrating to Cloudera Data Warehouse and running your Hive workloads on LLAP mode. LLAP offers ETL performance and scalability for complex data warehousing jobs.

Such a move can meet or exceed your customer satisfaction requirements, or cut the expense of running your jobs, or both. You set up automatic scaling up of compute resources when needed, or shutting down when not needed.

LLAP along with the query isolation feature is best suited for data-intensive queries, such as ETL queries, and require auto-scaling based on the total scan size of the query.

If your Hive jobs are unpredictable and if you are running complex interactive queries, migrate to Cloudera Data Warehouse. You perform the following configuration when taking this path:

- Configure Cloudera Data Warehouse according to your plan for scaling and concurrency
- Tune the Hive Virtual Warehouse to handle peak workloads

When you create a Cloudera Data Warehouse Virtual Warehouse, you configure the size and concurrency of queries to set up LLAP. The configuration is simple compared to HDP configurations of LLAP. In Hive Virtual Warehouses, each size setting indicates the number of concurrent queries that can be run. For example, an X-Small Hive on LLAP Virtual Warehouse can run 2 TB of data in its cache.

Related Information

Apache Tez execution engine for Apache Hive
Query isolation for data-intensive queries
Auto scaling of Virtual Warehouses in CDW
Virtual Warehouse sizing requirements
Tuning Virtual Warehouses

Apache Tez processing of Hive jobs

After migrating to Cloudera Private Cloud Base, Cloudera Public Cloud, or Cloudera Data Warehouse, you must understand how the Apache Tez execution engine is used to run your Hive workloads.

Apache Tez provides the framework to run a job that creates a graph with vertices and tasks. The entire execution plan is created under this framework. Apache Tez provides the following execution modes:

- Container mode — Every time you run a Hive query, Tez requests a container from YARN.
- LLAP mode — Every time you run a Hive query, Tez asks the LLAP daemon for a free thread, and starts running a fragment.

SQL syntax in Hive is the same irrespective of execution mode used in Hive. In Cloudera Private Cloud Base and Cloudera Public Cloud, Tez always runs in container mode and is commonly called Hive on Tez. In Cloudera Data Warehouse, Tez always runs in LLAP mode. You can use the query isolation feature if you are running complex ETL workloads.

Migration paths for Spark users

If you are on HDP and executing Spark workloads in Hive LLAP mode, and you want to upgrade to Cloudera, you can follow the migration path that matches your security needs. It is recommended that you upgrade to Cloudera Private Cloud Base and choose either Hive Warehouse Connector (HWC) or native Spark readers to query Hive from Spark.

As a replacement for the HWC LLAP execution mode in HDP, you can use the HWC Secure Access Mode in Cloudera Private Cloud Base that offers fine-grained access control (FGAC) column masking and row filtering to secure managed (ACID), or even external, Hive table data that you read from Spark.

The following migration paths are recommended based on certain factors:

Migrate to Cloudera Private Cloud Base

- Use HWC JDBC Cluster mode — If the user does not have access to data in the file system and if the database query returns are less than 1 GB of data.
- Use HWC Secure access mode — If the user does not have access to data in the file system and if the database query returns are more than 1 GB of data.
- Use HWC Direct reader mode — If the user has access to data in the file system and if you are querying Hive managed tables.

- Use native Spark reader — If the user has access to data in the file system and if you are querying Hive external tables.

The following diagram shows these recommended paths:

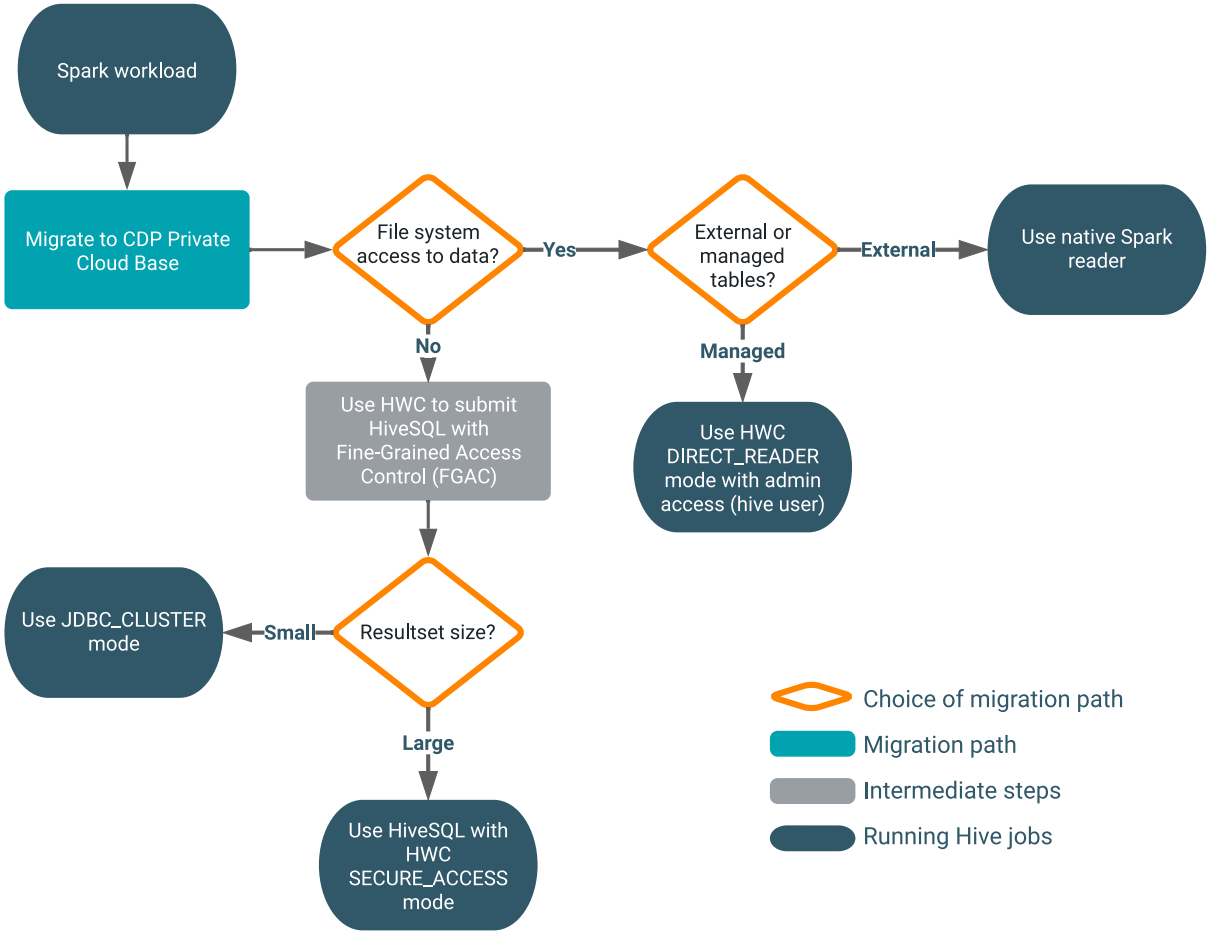

Migration to Cloudera Private Cloud Base

If you are a Spark user migrating from HDP to Cloudera and accessing Hive workloads through the Hive Warehouse Connector (HWC), consider migrating to Cloudera Private Cloud Base and based on your use cases, use the various HWC read modes to access Hive managed tables from Spark.

Use HWC JDBC Cluster mode

If the user does not have access to the file systems (restricted access), you can use HWC to submit HiveSQL from Spark with benefits of fine-grained access control (FGAC), row filtering and column masking, to securely access Hive tables from Spark.

However, if the size of your database query returns are less than 1 GB of data, it is recommended that you use HWC JDBC Cluster mode in which Spark executors connect to Hive through JDBC, and execute the query. Larger workloads are not recommended for JDBC reads in production due to slow performance.

Use HWC Secure access mode

If the user does not have access to the file systems (restricted access) and if the size of database query returns are greater than 1 GB of data, it is recommended to use HWC Secure access mode that offers fine-grained access control (FGAC), row filtering and column masking to access Hive table data from Spark.

Secure access mode enables you to set up an HDFS staging location to temporarily store Hive files that users need to read from Spark and secure the data using Ranger FGAC.

Use HWC Direct reader mode

If the user has access to the file systems (ETL jobs do not require authorization and run as super user) and if you are accessing Hive managed tables, you can use the HWC Direct reader mode to allow Spark to read directly from the managed table location.

Important: This workload must be run with 'hive' user permissions.

If you are querying Hive external tables, use Spark native readers to read the external tables from Spark.

Related Information

Row-level filtering and column masking in Hive

Introduction to HWC JDBC read mode

Introduction to HWC Secure access mode

Introduction to HWC Direct reader mode

HWC changes from HDP to Cloudera

You need to understand the Hive Warehouse Connector (HWC) changes from HDP to Cloudera. Extensive HWC documentation can prepare you to update your HWC code to run on Cloudera. In Cloudera, methods and the configuration of the HWC connections differ from HDP.

Deprecated methods

The following methods have been deprecated in Cloudera Runtime 7.1.7:

- hive.execute()
- hive.executeQuery()

The HWC interface is simplified in Cloudera resulting in the convergence of the execute/ executeQuery methods to the sql method. The execute/ executeQuery methods are deprecated and will be removed from Cloudera in a future release. Historical calls to execute/ executeQuery used the JDBC connection and were limited to 1000 records. The 1000 record limitation does not apply to the sqlmethod, although using JDBC cluster mode is recommended only for production for workloads having a data size of 1GB or less. Larger workloads are not recommended for JDBC reads in production due to slow performance.

Although the old methods are still supported in Cloudera for backward compatibility, refactoring your code to use the sql method for all configurations (JDBC client, Direct Reader V1 or V2, and Secure Access modes) is highly recommended.

Recommended method refactoring

The following table shows the recommended method refactoring:

API	From HDP	To Cloudera	HDP Example	Cloudera Example
HWC sql API	execute and executeQuery methods	sql method	hive.execute("select * from default.hwctest").show(1, false)	hive.sql("select * from default.hwctest").show(1, false)
Spark sql API	sql and spark.read.table methods	No change	sql("select * from managedTable").show scala> spark.read.table("managedTable").show	sql("select * from managedTable").show scala> spark.read.table("managedTable").show

API	From HDP	To Cloudera	HDP Example	Cloudera Example
DataFrames API	spark.read.format method	No change	val df = spark.read.format("HiveAcid").options(Map("table" -> "default.acidtbl")).load()	val df = spark.read.format("HiveAcid").options(Map("table" -> "default.acidtbl")).load()

Deprecated and changed configurations

HWC read configuration is simplified in Cloudera. You use a common configuration for Spark Direct Reader, JDBC Cluster, or Secure Access mode.

The following HWC configurations that you might have used in HDP 3.1.5 cannot be used in Cloudera:

- --conf spark.hadoop.hive.llap.daemon.service.hosts
- --conf spark.hadoop.hive.zookeeper.quorum

Recommended configuration refactoring

Refactor configuration code to remove unsupported configurations. Use the following common configuration property: spark.datasource.hive.warehouse.read.mode.

You can transparently read data from Spark with HWC in different modes using just spark.sql("<query>").

Secured cluster configurations

In secured cluster configurations, you must set configurations as described in the HWC documentation for Cloudera:

- --conf "spark.security.credentials.hiveserver2.enabled=true"
- --conf "spark.sql.hive.hiveserver2.jdbc.url.principal=hive/_HOST@ROOT.HWX.SITE"

  The jdbc url must not contain the jdbc url principal and must be passed as shown here.

Deprecated features

The following features have been deprecated and will be removed from Cloudera in a future release:

- Catalog browsing
- JDBC client mode configuration

Related Information

Introduction to Hive Warehouse Connector

..

Migrating Operational Database to Cloudera Public Cloud

Date published: 2021-10-25
Date modified: 2023-03-10

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

HBase Migration through Cloudera Replication Manager.....4
Cloudera Replication Plugin.....5
HBase migration prerequisites.....6
Migrating HBase data from CDH/HDP to Cloudera Operational Database Public Cloud..... 12
Cloudera Replication Manager.....17

Phoenix Replication to Cloudera Operational Database.....18
Replicating Phoenix Data Tables.....18
Replicating Phoenix tables for versions lower than 4.14..... 18
Replicating Phoenix 4.14 and newer versions including 5.x.....19
Replicating Phoenix Index Tables.....20

HBase Migration through Cloudera Replication Manager

Apache HBase is a scalable, distributed, column-oriented datastore that provides real-time read/write random access to very large datasets hosted on HDFS. In Cloudera Operational Database you use Apache HBase as a datastore with HDFS and/or S3/ABFS providing the storage infrastructure.

You can use one of the following methods to replicate HBase data based on your requirements:

Table 1: Replication methods

Methods	Description	When to use
Cloudera Replication Plugin for cluster versions that Cloudera Replication Manager does not support.	You can prepare your data for migration, then set up the replication plugin and use a snapshot to migrate your data.	The following list consolidates all the minimum supported versions of source and target cluster combinations for which you can use the replication plugin to replicate HBase data. The replication plugin is compatible with all the ClouderaPublic Cloud releases. • From CDH 5.10 using Cloudera Manager 6.3.0 to ClouderaPublic Cloud on AWS. See Replicate data from CDH 5 source clusters. • From CDH 5.10 using Cloudera Manager 6.3.4 to ClouderaPublic Cloudon Azure. See Replicate data from CDH 5 source clusters. • From CDH 6.1 using Cloudera Manager 6.3.0 to ClouderaPublic Cloud on AWS. See Replicate data from CDH 6 source clusters. • From CDH 6.1 using Cloudera Manager 7.1.1/6.3.4 to ClouderaPublic Cloud on Azure. See Replicate data from CDH 6 source clusters. • Cloudera Private Cloud Base 7.1.1 using Cloudera Manager 7.1.1 to ClouderaPublic Cloud on AWS and Azure. See Replicate data from Cloudera Private Cloud Base source clusters. • HDP 2.6.5 and HDP 3.1.1 to ClouderaPublic Cloud on AWS and Azure. See Replicate data from HDP 2 and HDP 3 source clusters. For information about use cases that are not supported by Cloudera Replication Manager, see support matrix.

Methods	Description	When to use
Cloudera Replication Manager	You can use Cloudera Replication Manager to migrate HBase data that uses HBase replication through HBase replication policy.	When the source cluster and target cluster meet the requirements of supported use cases. See caveats. The following list consolidates all the minimum supported versions of source and target cluster combinations for which you can use HBase replication policies to replicate HBase data. • From Cloudera Private Cloud Base 7.1.6 using Cloudera Manager 7.3.1 to Cloudera Public Cloud 7.2.14 using Cloudera Manager 7.6.0. See Replicate data from Cloudera Private Cloud Base source clusters. • From CDH 6.3.3 using Cloudera Manager 7.3.1 to Cloudera Public Cloud 7.2.14 using Cloudera Manager 7.6.0. See Replicate data from CDH 6 source clusters. • From CDH 5.16.2 using Cloudera Manager 7.4.4 (patch-5017) to Cloudera Operational Database 7.2.14. See Replicate data from CDH 5 source clusters. • From Cloudera Operational Database 7.2.14 to Cloudera Operational Database 7.2.14. See support matrix for more information.

Related Information

HBase Replication Policy

Creating HBase Replication Policy

Cloudera Replication Plugin

You can migrate your HBase data from CDH or HDP to Cloudera Operational Database Public Cloud. First you have to prepare your data for migration, then set up the replication plugin and use a snapshot to migrate your data.

Before starting the migration you have to understand and perform all the necessary data migration preparation tasks. Then, in the absence of Cloudera Replication Manager support, you must perform certain tasks on both your source cluster (Cloudera or HDP) and your destination Cloudera cluster to ensure a successful migration.

The following is an overview of the HBase migration steps:

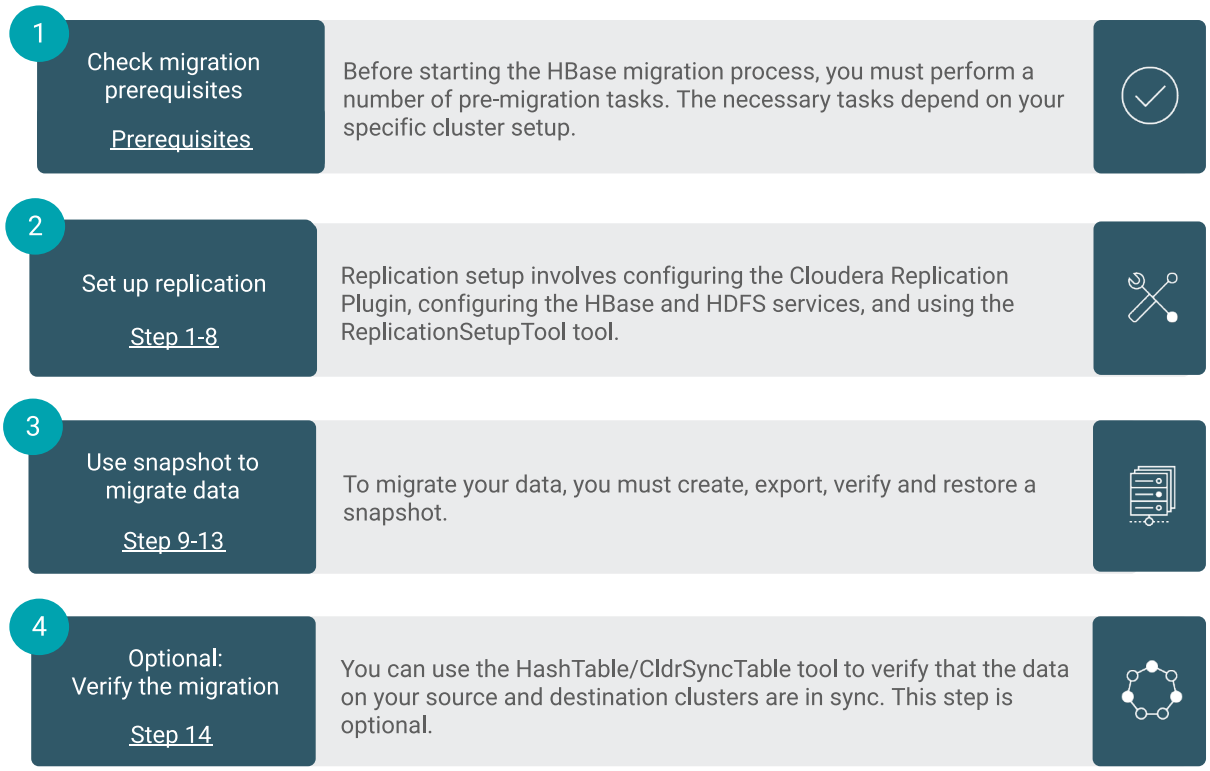

rect 45, 45, 219, 121 HBase migration prerequisites
rect 41, 185, 221, 254 Step 1-8
rect 39, 305, 217, 382 Step 9-13
rect 47, 433, 215, 509 Step 14

HBase migration prerequisites

Before migrating HBase data from Cloudera/HDP to Cloudera Operational Database Public Cloud, you have to understand and perform all the necessary migration preparation tasks.

- If you are migrating from CDH, configure Ranger ACLs in Cloudera corresponding to the HBase ACLs in your existing CDH cluster.

  For more information, see Configure a resource-based service:HBase.
- Migrate all applications to use the new HBase-Spark connector because the Spark-HBase connector that you were using in CDH or HDP is no longer supported in Cloudera.

  For more information, see Using the HBase-Spark connector.
- Ensure that all data has been migrated to a supported encoding type before the migration. For more information, see Removing PREFIX_TREE Data Block Encoding on page 7 . This step is applicable for HBase 1.x when you are migrating data from HDP 2 or CDH 5. This step is not applicable for HBase 2.x.
- Ensure that your HFiles are compatible with the version Apache HBase in Cloudera. For more information, see Validating HFiles on page 9.
- Ensure that you upgrade any external or third-party co-processors manually because they are not automatically upgraded during migration. Ensure that your co-processor classes are compatible with Cloudera. For more information, see Checking co-processor classes on page 8.
- Review the deprecated APIs and incompatibilities when upgrading from HDP 2.x or CDH 5.x to Cloudera. For more information, see HBase in Deprecation notices in Cloudera Runtime.

- Install Cloudera Replication Plugin on the source cluster.

  Important: Contact your Cloudera account team to get access to the Cloudera Replication Plugin version compatible with your source cluster CDH or HDP version.

- On the target Cloudera Operational Database cluster, make sure ZooKeeper (2181) and RegionServer (16020) ports are opened for the IP addresses of all RegionServers in the source cluster.

  For AWS, this involves editing the Security Groups which are in use by the Cloudera Operational Database cluster.
- Ensure that all the RegionServers' hosts in the Cloudera Operational Database cluster are resolvable by both the IP and name addresses by all the RegionServers' hosts in the source cluster.

Removing PREFIX_TREE Data Block Encoding

Before migrating to Cloudera Operational Database Public Cloud, ensure that you have transitioned all the data to a supported encoding type.

About this task

Important:

Ensure that you complete all the pre-migration steps if you have Apache HBase installed in your existing CDH cluster.

The PREFIX_TREE data block encoding code is removed in Cloudera Operational Database Public Cloud, meaning that HBase clusters with PREFIX_TREE enabled will fail. Therefore, before migrating to Cloudera Operational Database Public Cloud you must ensure that all data has been transitioned to a supported encoding type.

The following pre-upgrade command is used for validation: hbase pre-upgrade validate-dbe

If your cluster is Kerberized, ensure that you run the kinit command as a hbase user before running the pre-upgrade commands. Ensure you have valid Kerberos credentials. You can list the Kerberos credentials using the klist command, and you can obtain credentials using the kinit command.

Before you begin

1. Download and distribute parcels for the target version of Cloudera.

   Important: Do not activate the parcel yet.

   If the downloaded parcel version is higher than the current Cloudera Manager version, the following error message displayed:

   Error for parcel CDH-7.x.parcel : Parcel version 7.X is not supported by this version of Cloudera Manager. Upgrade Cloudera Manager to at least 7.X before using this version of the parcel.

   You can safely ignore this error message.
2. Find the installed parcel at /opt/cloudera/parcels.

   For example: /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase

Use the Cloudera parcel to run the pre-upgrade commands. Cloudera recommends that you run them on an HMaster host.

Procedure

1. Run the hbase pre-upgrade validate-dbe command using the new installation.

   For example, if you have installed the CDH-7.1.1-1 parcel, you must run the following command:

   ```
   /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase pre-upgr
   ade validate-dbe
   ```

   The commands check whether your table or snapshots use the PREFIX_TREE Data Block Encoding.

   This command does not take much time to run because it validates only the table-level descriptors.

   If PREFIX_TREE Data Block Encoding is not used, the following message is displayed:

   ```
   The used Data Block Encodings are compatible with HBase 2.0.
   ```

   If you see this message, your data block encodings are compatible, and you do not have to perform any more steps.

   If PREFIX_TREE Data Block Encoding is used, a similar error message is displayed:

   ```
   2018-07-13 09:58:32,028 WARN  [main] tool.DataBlockEncodingValidator: In
   compatible DataBlockEncoding for table: t, cf: f, encoding: PREFIX_TREE
   ```

   If you got an error message, continue with Step 2 and fix all your PREFIX_TREE encoded tables.
2. Fix your PREFIX_TREE encoded tables using the old installation.

   You can change the Data Block Encoding type to PREFIX, DIFF, or FAST_DIFF in your source cluster.

   Our example validation output reported column family f of table t is invalid. Its Data Block Encoding type is changed to FAST_DIFF in this example:

   ```
   hbase> alter 't', { NAME => 'f', DATA_BLOCK_ENCODING => 'FAST_DIFF' }
   ```

Checking co-processor classes

External co-processors are not automatically upgraded, you must upgrade them manually. Before migrating, ensure that your co-processors are compatible with the migration.

About this task

Important:

Ensure that you complete all the pre-migration steps if you have Apache HBase installed in your existing CDH cluster.

There are two ways to handle co-processor upgrade:

- Upgrade your co-processor jars manually before continuing the migration.
- Temporarily unset the co-processors and continue the migration.

  Once they are manually upgraded, they can be reset.

Attempting to migrate without upgrading the co-processor jars can result in unpredictable behaviour such as HBase role start failure, HBase role crashing, or even data corruption.

If your cluster is Kerberized, ensure that you run the kinit command as a hbase user before running the pre-upgrade commands.

Procedure

1. Download and distribute parcels for target version of Cloudera Operational Database Public Cloud.

   Important: Do not activate the parcel yet.

   If the downloaded parcel version is higher than the current Cloudera Manager version, the following error message displayed:

   Error for parcel CDH-7.X.parcel : Parcel version 7.X is not supported by this version of Cloudera Manager. Upgrade Cloudera Manager to at least 7.X before using this version of the parcel.

   You can safely ignore this error message.

2. Run the hbase pre-upgrade validate-cp commands to check if your co-processors are compatible with the migration.

   Use the Cloudera parcel to run the pre-upgrade commands. Cloudera recommends that you run them on an HMaster host.

   For example, you can check for co-processor compatibility on master:

   ```
   $ /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase pre-up
   grade validate-cp -jar /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.322
   4867/jars/ -config
   ```

   Or, you can validate every table level co-processors where the table name matches to the .* regular expression:

   ```
   $ /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase pre-up
   grade validate-cp -table '.*'
   ```

   Optionally, you can run the following command for a more detailed output:

   ```
   HBASE_ROOT_LOGGER=DEBUG,console hbase pre-upgrade validate-cp -table '.*'
   ```

   This way you can verify that all of the required tables were checked. The detailed output should contain lines like the following where test_table is a table on the server:

   ```
   21/05/10 11:07:58 DEBUG coprocessor.CoprocessorValidator: Validating table
    test_table
   ```

3. Check the output to determine if your co-processors are compatible with the upgrade.

   The output looks similar to the following:

   ```
   $ hbase pre-upgrade validate-cp -config
                           ... some output ...
                           $ echo $?
                           0
   ```

   If echo $? prints 0, the check was successful and your co-processors are compatible. A non-zero value means unsuccessful, your co-processors are not compatible.

Validating HFiles

Before migrating to Cloudera Operational Database Public Cloud ensure that your HFiles are compatible with the version Apache HBase in Cloudera.

About this task

Important:

Ensure that you complete all the pre-migration steps if you have Apache HBase installed in your existing CDH cluster.

After converting all the PREFIX_TREE encoded HFiles to a supported format, there may be HFiles that are not compatible with HBase in Cloudera.

Use the following pre-upgrade commands to validate HFiles:

- ```
  hbase pre-upgrade validate-hfile
  ```
- ```
  hbase hbck
  ```

The validate-hfile tool lists all the corrupt HFiles with details. The hbck command identifes HFiles that are in a bad state.

If your cluster is Kerberized, ensure that you run the kinit command as a hbase user before running the pre-upgrade commands.

Before you begin

1. Download and distribute parcels for the target version of Cloudera.

   Important: Do not activate the parcel yet.

   If the downloaded parcel version is higher than the current Cloudera Manager version, the following error message displayed:

   Error for parcel CDH-7.x.parcel : Parcel version 7.X is not supported by this version of Cloudera Manager. Upgrade Cloudera Manager to at least 7.X before using this version of the parcel.

   You can safely ignore this error message.
2. Find the installed parcel at /opt/cloudera/parcels.

   For example: /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase

Use the Cloudera parcel to run the pre-upgrade commands. Cloudera recommends that you run them on an HMaster host.

Procedure

1. Run the hbase pre-upgrade validate-hfile command using the new Cloudera installation.

   This command checks every HFile (including snapshots) to ensure that the HFiles are not corrupted and have the compatible block encoding. It opens every HFile, and this operation can take a long time based on the size and number of HFiles in your cluster.

   For example, if you have installed the CDH-7.1.1-1 parcel, you must run the following command:

   ```
   /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase pre-upgr
   ade validate-hfile
   ```

   If there are no incompatible HFiles, the following message is displayed:

   ```
   Checked 3 HFiles, none of them are corrupted.
   ```

```
                            There are no incompatible HFiles.
```

If you have an incompatible HFile, a similar error message to this is displayed:

```
2018-06-05 16:20:47,322 INFO  [main] tool.HFileContentValidator: Corrupted
 file: hdfs://example.com:8020/hbase/data/default/t/72ea7f7d625ee30f959897
d1a3e2c350/prefix/7e6b3d73263c4851bf2b8590a9b3791e
                            2018-06-05 16:20:47,383 INFO  [main] tool.HFileCo
ntentValidator: Corrupted file: hdfs://example.com:8020/hbase/archive/da
ta/default/t/56be41796340b757eb7fff1eb5e2a905/f/29c641ae91c34fc3bee881f4
5436b6d1
```

If you get an error message, continue with Step 2 otherwise skip to Step 5.

2. Identify what kind of HFiles were reported in the error message.

The report can contain two different kinds of HFiles, they differ in their path:

- If an HFile is in /hbase/data then it belongs to a table.
- If an HFile is located under /hbase/archive/data then it belongs to a snapshot.

3. Fix the HFiles that belong to a table.

a) Get the table name from the output.

Our example output showed the /hbase/data/default/t/… path, which means that the HFile belongs to the t table which is in the default namespace.

b) Rewrite the HFiles by issuing a major compaction. Use the old installation.

shell> major_compact 't'

When compaction is finished, the invalid HFile disappears.

4. Fix the HFiles that belong to a snapshot that is needed. Use the old installation.

a) Find the snapshot which refers the invalid HFile. You can do this using a shell script and the following example code. In our example output, the HFile is 29c641ae91c34fc3bee881f45436b6d1:

```
#!/bin/bash
                                # Find the snapshot which referes to the
 invalid HFile
                                for snapshot in $(hbase snapshotinfo -
list-snapshots 2> /dev/null | cut -f 1 -d \|);
                                do
                                echo "checking snapshot named '${snapsho
t}'"
                                hbase snapshotinfo -snapshot "${snapsho
t}" -files 2> /dev/null | grep 29c641ae91c34fc3bee881f45436b6d1
                                done
```

The following output means that the invalid file belongs to the t_snap snapshot:

```
checking snapshot names 't__snap'
                                     1.0 K t/56be41796340b757eb7fff1eb5
e2a905/f/29c641ae91c34fc3bee881f45436b6d1 (archive)
```

b) Convert snapshot to another HFile format if data encoding is the issue using the following command:

```
# creating a new namespace for the cleanup process
                                hbase> create_namespace 'pre_upgrade_
cleanup'
                                # creating a new snapshot
                                hbase> clone_snapshot 't_snap', 'pre_
upgrade_cleanup:t'
```

```
                                hbase> alter 'pre_upgrade_cleanup:t',
 { NAME => 'f', DATA_BLOCK_ENCODING => 'FAST_DIFF' }
                                hbase> major_compact 'pre_upgrade_clean
up:t'
```

Important: Confirm if the major compaction is complete in the RegionServer Web user interface or the RegionServer logs before you run the following commands to remove invalid snapshots. If the major compaction process is not completed when you delete the snapshot, your files may get corrupted.

```
# removing the invalid snapshot
                                hbase> delete_snapshot 't_snap'
                                # creating a new snapshot
                                hbase> snapshot 'pre_upgrade_cleanup:t',
 't_snap'
                                # removing temporary table
                                hbase> disable 'pre_upgrade_cleanup:t'
                                hbase> drop 'pre_upgrade_cleanup:t'
                                hbase> drop_namespace 'pre_upgrade_cle
anup'
```

5. Run the hbck command on to identify HFiles in a bad state and remedy those HFiles.

Migrating HBase data from CDH/HDP to Cloudera Operational Database Public Cloud

When you are migrating from CDH or HDP to Cloudera Operational Database Public Cloud, you have to set up the Cloudera Replication Plugin and then use a snapshot to migrate your data to Cloudera Operational Database Public Cloud. Once your data is migrated, optionally you can use the HashTable/CldrSyncTable tool to verify the migration.

About this task

Important: In this topic, we are discussing the manual steps to migrate the HBase data into the Cloudera Operational Database Public Cloud. Alternatively, you can use the Replication Manager to migrate your HBase data. For more information, see *Using HBase replication policies*.

Before you begin

Ensure you have understood and performed all the necessary data migration preparation tasks listed in *HBase migration prerequisites*.

Procedure

1. Provide the Cloudera Replication Plugin by configuring HBase Client Environment Advanced Configuration Snippet (Safety Valve) for hbase-env.sh for Gateway and HBase Service Environment Advanced Configuration Snippet (Safety Valve) for RegionServers and Masters on the source cluster.

For CDH

a. In Cloudera Manager, navigate to HBase Configuration .

b. Find the HBase Client Environment Advanced Configuration Snippet (Safety Valve) for hbase-env-sh property and add the following configuration to provide the Cloudera Replication Plugin path for the Gateway:

```
HBASE_CLASSPATH=$HBASE_CLASSPATH:/opt/cloudera/parcels/cloudera-opdb
-replication-[***REPLICATION PLUGIN VERSION***]-cdh[***CDH VERSION**
*]-/lib/*
```

For example:

```
HBASE_CLASSPATH=$HBASE_CLASSPATH:/opt/cloudera/parcels/cloudera-opdb-
replication-1.0-cdh5.14.4-/lib/*
```

c. Find the HBase Service Environment Advanced Configuration Snippet (Safety Valve) property and add the same configuration to provide the Cloudera Replication Plugin path for RegionServers and Masters:

```
HBASE_CLASSPATH=$HBASE_CLASSPATH:/opt/cloudera/parcels/cloudera-opdb
-replication-[***REPLICATION PLUGIN VERSION***]-cdh[***CDH VERSION**
*]-/lib/*
```

For example:

```
HBASE_CLASSPATH=$HBASE_CLASSPATH:/opt/cloudera/parcels/cloudera-opdb-
replication-1.0-cdh5.14.4-/lib/*
```

For more information on CDH parcels, see *Migrating HBase tables*.

For HDP

a. In Ambari, navigate to CONFIGS ADVANCED Advanced hbase-env hbase-env template.

b. Find the following line:

```
export HBASE_CLASSPATH=${HBASE_CLASSPATH}
```

c. Modify the line to include the following configuration:

```
export HBASE_CLASSPATH=${HBASE_CLASSPATH}:/usr/hdp/cloudera-opdb-rep
lication-[***REPLICATION PLUGIN VERSION***]-hdp[***version***]-SNAPS
HOT/lib/*
```

For example:

```
export HBASE_CLASSPATH=${HBASE_CLASSPATH}:/usr/hdp/cloudera-opdb-rep
lication-1.0-hdp-2.6.5-SNAPSHOT/lib/*
```

2. Generate and distribute credentials.
   a) Create a machine user with the name hbase-replication (or any one of your choice) in User Management Service (UMS) on the Cloudera Management Console.
   b) Set your workload password.
   c) Add this machine user as an Environment User using Manage Access in the destination Cloudera environment.
   d) Perform FreeIPA synchronization for the destination environments.
   e) Verify credentials using the kinit srv_$YOUR_REP_USER (for example, if the specified user was *HBASE-REPLICATION*, this would be srv_hbase-replication) command on any node.
   f) In the destination cluster, run the following command to generate the keystore:

```
hbase
com.cloudera.hbase.security.token.CldrReplicationSecurityTool -shared
key cloudera -password [***PASSWORD***] -keystore localjceks://file/tmp/
credentials.jceks
```

   g) Obtain hdfs user kerberos credentials on the destination cluster.

      Ensure that you have root access to execute the following command.

```
kinit -kt /var/run/cloudera-scm-agent/process/`ls -1 /var/run/cloudera-s
cm-agent/process/ | grep -i "namenode\|datanode" | sort -n | tail -1`/hd
fs.keytab hdfs/$(hostname -f)
```

   h) Add the generated jceks file to related hdfs folder in the destination cluster:

```
hdfs dfs -mkdir /hbase-replication
hdfs dfs -put /tmp/credentials.jceks /hbase-replication
hdfs dfs -chown -R hbase:hbase /hbase-replication
```

   i) Copy the generated jceks file to any host of the source cluster.

      Warning: You must copy the same jceks file to the source cluster instead of regenerating it.

   j) Add the copied jceks file to related hdfs folder in the source cluster (if source cluster is secured, you need hdfs kerberos credentials for destination cluster):

```
hdfs dfs -mkdir /hbase-replication
hdfs dfs -put /tmp/credentials.jceks /hbase-replication
hdfs dfs -chown -R hbase:hbase /hbase-replication
```

3. Set the hbase.security.replication.credential.provider.path property on the destination cluster using the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property.

   For example, if the path to the jceks file is hdfs://ns1/hbase-replication/credentials.jceks, set the following configuration:

```
<property>
<name>hbase.security.replication.credential.provider.path</name>
<value>cdprepjceks://hdfs@[***NAMESERVICE***]/hbase-replication/credenti
als.jceks</value>
</property>
```

   Warning: You have to customize the value of the hbase.security.replication.credential.provider.path property. You might need to replace this sample "ns1" value with the actual name service value in case of High Availability, or with the active NameNode address otherwise.

4. In the source cluster, enable replication for each table that you want to replicate:

   This can be done b setting the REPLICATION_SCOPE on the desired column families for that HBase table in the hbase shell.

```
$ hbase shell
alter 'my-table', {NAME=>'my-family', REPLICATION_SCOPE => '1'}
```

   If you do not know what column families exist in your HBase table, use the describe command:

```
$ hbase shell
describe 'MY-TABLE'
```

   Note: The enable_table_replication command is not supported in this configuration.

5. Set up the cloud services provider credentials for the source cluster using the core-site.xml HDFS configuration file.

   a) Find the core-site.xml HDFS configuration file:

      - Cloudera Manager: Navigate to HDFS Configuration and find the Cluster-wide Advanced Configuration Snipper (Safety Valve) for core-site.xml property.
      - Ambari: Navigate to HDFS CONFIGS ADVANCED and find the custom core-site template.

   b) Add the following configuration:

For AWS

Configure the following properties on the source cluster with the AWS keys and write permission to the bucket:

- fs.s3a.access.key
- fs.s3a.secret.key
- fs.s3a.endpoint

For example, fs.s3a.endpoint="s3.eu-central-1.amazonaws.com"

For ABFS

Configure your storage account access key by setting the fs.azure.account.key.[***ACCOUNT NAME***].blob.core.windows.net property to the access key:

```
<property>  <name>fs.azure.account.key.[***ACCOUNT NAME***].blob
.core.windows.net</name>
<value>[***ACCOUNT NAME***]-ACCESS-KEY</value>
</property>
```

Access key can be obtained from the Access keys in your storage account settings.

6. Restart the RegionServers and add the client configurations on the source cluster:

   - Cloudera Manager: Click Restart stale services.
   - Ambari: Click Restart all required.

7. Use the ReplicationSetupTool tool on the source cluster to define the replication peer.

   Run this command as the hbase user. Additionally, use the same keystore path that you provided in Step 3.

```
sudo -u hbase hbase org.apache.hadoop.hbase.client.replication.Replicati
onSetupTool -clusterKey "zk-host-1,zk-host-2,zk-host-3:2181:/hbase" -end
pointImpl "org.apache.hadoop.hbase.replication.regionserver.CldrHBaseInt
erClusterReplicationEndpoint" -peerId 1 -credentialPath "cdprepjceks://h
```

```
dfs@[***NAMESERVICE***]/hbase-replication/credentials.jceks" -replicatio
nUser srv_hbase-replication
```

This example uses sudo. However, you can use kinit if it is appropriate for your source cluster setup.

The clusterKey parameter reflects the destination cluster's zookeeper quorum address. Here, *ZK-HOST-1*, *ZK-HOST-2*, *ZK-HOST-3* represent the destination cluster Zookeeper hostnames.

NAMESERVICE is the source cluster nameservice for HA enabled cluster. It can also be the hostname of an active NameNode for a non-HA cluster.

8. Disable the replication peer in the hbase shell.

Disabling the peer before taking the snapshot ensures that incremental data written to your table is not lost while copying the HBase snapshot to Cloudera Operational Database.

```
$./bin/hbase shell
hbase> disable_peer '1'
```

9. Take a snapshot on the source cluster.

For Cloudera Manager

a. In Cloudera Manager on the source cluster, select the HBase service.
b. Click Table Browser.
c. Select a table.
d. Click Take Snapshot.
e. Specify the name of the snapshot and click Take Snapshot.

For hbase shell

a. You can take the snapshot through the hbase shell:

```
$ hbase shell
hbase> snapshot 'myTable', 'myTableSnapshot-122112'
```

10. Export the snapshot to the destination cluster using the ExportSnapshot tool.

You must run the ExportSnapshot command as the hbase user or as the user that owns the files. The ExportSnapshot tool executes a MapReduce job similar to distcp to copy files to the other cluster. The tool works at the file-system level, so the HBase cluster can be offline, but ensure that the HDFS cluster is online.

a) Navigate to Cloudera Management Console Cloudera Operational Database your Database or Environment .
b) Copy the Cloud Storage Location.
   - In the case of AWS, this is a s3a path: s3a://bucket/path/hbase
   - In the case of Azure blob storage (ABFS), this is an abfs path: abfs://<storagename>/datalake/<datahub_name>/hbase
c) Run the following command in the hbase shell on the source cluster to export a snapshot from the source cluster to the destination cluster:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snaps
hot [***SNAPSHOT NAME***] -copy-to [***DESTINATION***] -mappers 16
```

Replace [***DESTINATION***] with the s3a or abfs path obtained in *step b*.

Important: Snapshots must be enabled on the source and destination clusters. When you export a snapshot, the table's HFiles, logs, and the snapshot metadata are copied from the source cluster to the destination cluster.

It may be necessary to increase the value of the hbase.master.hfilecleaner.ttl property in the source cluster to work around known issues in HBase file cleaning due to known issues in these legacy products.

11. Verify that the snapshot exist in the Cloudera Operational Database database using the following command:

```
$ hbase shell
hbase> list_snapshots
```

12. Restore the snapshot into an HBase table using the following command:

```
$./bin/hbase shell
hbase> restore_snapshot [***SNAPSHOT_NAME***]
```

13. Enable the peer on the source cluster to begin replicating data:

```
$./bin/hbase shell
hbase> enable_peer '1'
```

14. Use the HashTable/CldrSyncTable tool to ensure that data is synchronized between your source and destination clusters.
    a) Run HashTable on the source cluster:

    ```
    hbase org.apache.hadoop.hbase.mapreduce.HashTable [***TABLE
     NAME***] [***HASH OUTPUT PATH***]
    hdfs dfs -ls -R [***HASH OUTPUT PATH***]
    ```

    Hashed indexes are generated on the source cluster.

    b) Run CldrSyncTable with the -cldr.cross.domain option on the source cluster:

    ```
    hbase org.apache.hadoop.hbase.mapreduce.CldrSyncTable --cldr.cross.domai
    n --targetzkcluster=[***TARGET ZOOKEEPER QUORUM***]:[***TARGET ZOOKEEPER
     PORT***]:[***TARGET ZOOKEEPER ROOT FOR HBASE***]
    [***HASH OUTPUT PATH***] [***SOURCE TABLE NAME***] [***TARGET TABLE NAME
     ON THE TARGET CLUSTER***]
    ```

    Hash indexes are generated on the destination cluster and compared to the hash indexes generated on the source cluster. If the hash indexes are not identical, synchronization is run for that section of rows.

Related Information

Using HBase replication policies

Migrating HBase tables

Cloudera Replication Manager

Cloudera Replication Manager is a key Cloudera service to migrate data between Cloudera environments, on-prem clusters, and clusters on cloud infrastructure services. Cloudera Replication Manager provides a simple, easy-to-use, and feature-rich data movement capability to deliver data and metadata where it is needed and provides secure data backup with disaster recovery functionality.

Cloudera Replication Manager provides HBase replication policies to replicate HBase data from source classic clusters to Cloudera Operational Database.

The following table lists the minimum versions that HBase replication policies support to replicate HBase data.

Table 2: Minimum supported versions for HBase replication policies

Lowest Supported Source Cloudera Manager Version	Lowest Supported Source CDH/Cloudera Version	Source Cluster Type	Lowest Supported Target Cloudera Manager Version	Lowest Supported Target Cloudera Version	Target Cluster Type
7.3.1	7.1.6	Cloudera Private Cloud Base	7.6.0	7.2.14	Cloudera Operational Database

Lowest Supported Source Cloudera Manager Version	Lowest Supported Source CDH/Cloudera Version	Source Cluster Type	Lowest Supported Target Cloudera Manager Version	Lowest Supported Target Cloudera Version	Target Cluster Type
7.3.1	6.3.3	CDH*	7.6.0	7.2.14	Cloudera Operational Database
7.4.4 (patch-5017)	5.16.2	CDH	-	7.2.14	Cloudera Operational Database
-	7.2.14	Cloudera Operational Database	-	7.2.14	Cloudera Operational Database
*Cloudera Replication Manager supports HBase replication from non-kerberized source clusters only for CDH 6.3.x versions.					

Related Information

HBase Replication Policy

Creating HBase Replication Policy

Phoenix Replication to Cloudera Operational Database

Cloudera Operational Database provides both HBase and Phoenix. You might request a backported SchemaTool utility for HDP 2.6, HDP3, and Phoenix parcel for CDH6 from Cloudera support to reconstruct the original DDL.

Cloudera also provides the Cloudera Operational Database Replication plugin, which enables HBase replication from a number of products which also include HBase and Phoenix to Cloudera Operational Database, such as CDH 5, CDH 6, HDP 2.6, and HDP 3.1. You can replicate Phoenix tables to Cloudera Operational Database using the Replication plugin.

Currently, Cloudera Operational Database includes Apache Phoenix 5.1.1 while other products include a range of versions of Phoenix from 4.7.0 to 5.0.0.

Replicating Phoenix Data Tables

When you want to configure replication of Phoenix tables from all versions, you need to know the schema of the data to be replicated. However, all on-prem products which include Phoenix today do not have the ability to print the equivalent `CREATE TABLE` command for a table in Phoenix.

Cloudera has built some functionality to recreate the `CREATE TABLE` command for previously released versions of Phoenix as a standalone tool. Please contact your Cloudera representative to evaluate the ability to use this utility against your version of Phoenix.

Replicating Phoenix tables for versions lower than 4.14

For Phoenix versions lower than 4.14, you must treat both the Phoenix data tables and Phoenix index tables similar to any other HBase table.

Procedure

1. Copy the contents of your Phoenix data table in the source cluster to Cloudera Operational Database in the same manner that you would do for an HBase table (either using Cloudera Replication Manager or through the HBase ExportSnapshot utility). To copy the contents of the Phoenix data table, see HBase migration steps 1-11.

   Note: Ensure the name of the HBase table that you provide in snapshot operations is the same as your Phoenix table name. When creating the Phoenix table, ensure it appears within the quotation marks. Otherwise, it appears in capital letters.

   ```
   hbase> disable 'myTable'
   ```

```
hbase> restore_snapshot 'myTableSnapshot-122112'
```

2. Validate that the data is readable by scanning the restored table in HBase through hbase shell before proceeding further.
3. Verify if the corresponding HBase table for your Phoenix data table exists in Cloudera Operational Database.
4. Obtain the DDL statement (`CREATE TABLE`) that you have used to create the table in your legacy system.
5. Run the `CREATE TABLE` command in phoenix-sqlline in Cloudera Operational Database with the option COLUMN_ENCODED_BYTES=0 appended to it.

```
For example, if the original create table statement is:
jdbc:phoenix> CREATE TABLE MY_DATA(rk VARCHAR NOT NULL PRIMARY KEY, col1
 INTEGER) SALT_BUCKETS = 4;
The corresponding statement to run in Cloudera Operational Database will
 be:

jdbc:phoenix> CREATE TABLE MY_DATA(rk VARCHAR NOT NULL PRIMARY KEY, col1
INTEGER) SALT_BUCKETS = 4,
COLUMN_ENCODED_BYTES=0;
```

   This option disables the column encoding feature, which is enabled by default since Phoenix versions 4.14 till 5.0. This feature uses binary representations in the HBase column qualifier rather than the column name provided to Phoenix. If the data in the Phoenix table does not match this configuration, Phoenix does not display any data on query but the data appears if queried through the HBase APIs. Running a create table command when an HBase table already exists creates the corresponding internal Phoenix metadata while leaving all other data in place.
6. Validate that query using phoenix-sqlline. This must return the expected data from your Phoenix table.

Replicating Phoenix 4.14 and newer versions including 5.x

Phoenix 4.14 and 5.x have options which may or may not use column encoding, whereas the older versions did not have this feature. You must determine if your existing Phoenix table uses column encoding. Following are the various ways to check this.

About this task

You can also create an HBase replication policy using Cloudera Replication Manager to replicate Phoenix 4.14 and 5.x versions into Cloudera Operational Database. For more information, refer to the second row of the *Replication methods* table in *HBase Migration through Replication Manager*.

Procedure

1. Follow the same steps as described for Phoenix older than 4.14, steps 1 - 4, and modify step 5 according to the following instructions.
2. Scan the raw data in HBase, and inspect if there are human-readable column names in the column qualifier. If you see human-readable names, you must set COLUMN_ENCODED_BYTES = 0. If you encounter a binary data ('\x00'), assume that COLUMN_ENCODED_BYTES must be set.
3. Inspect the create-table command from the legacy product. If you explicitly set COLUMN_ENCODED_BYTES in an earlier version, you must use the same value when re-creating the Phoenix table in Cloudera Operational Database.

   Note: Consult the SchemaTool output if you are on a sufficiently new version of Phoenix.

4. Validate the query using phoenix-sqlline. This must return the expected data from your Phoenix table.

   You might have installed python2 or python3, depending on the operating system version installed on your system; however, for the phoenix-sqlline script to work properly, you must ensure that the Python alternatives point to a specific version. Consider the following examples for Python alternatives,

```
$ alternatives --list|grep python
```

```
python                                          auto    /usr/bin/python3.6
```

Or

```
$ alternatives --list|grep python
python                                          auto    /usr/bin/python2.9
```

Related Information

HBase Replication Policy

Creating HBase Replication Policy

HBase Migration through Cloudera Replication Manager

Replicating Phoenix Index Tables

The secondary indexing implementation in Phoenix changed drastically between versions lower than Phoenix 4.14 and higher. The manner in which HBase replication sends Phoenix data is fundamentally incompatible with the new style of Phoenix secondary index maintenance. Architecturally, it is most desirable that we replicate only Phoenix data tables, and let the “local” HBase cluster maintain any secondary indexes for you. However, currently no “Phoenix-aware” replication strategy exists which is capable of automatically maintaining the Phoenix secondary indexes for replicated Phoenix data tables.

About this task

Follow the recommended steps in *Replicating Phoenix Data Tables* to replicate all Phoenix data tables before proceeding. For each index table that you want to replicate, perform the following steps:

1. Create the corresponding index in Cloudera Operational Database. Be sure to specify the same exact create index command that was used in the legacy product (both indexes columns and “include” columns). If you have a significant amount of data in the corresponding data table, include the async keyword in the create index command. The async keyword will create the table but not populate any data into this table.

   ```
   jdbc:phoenix> CREATE INDEX SMALL_INDEX on SMALL_TABLE(col1) INCLUDE (col
   2);
   jdbc:phoenix> CREATE INDEX BIG_INDEX ON BIG_TABLE(col1) INCLUDE (col2) A
   SYNC;
   ```

2. Enable replication on the Phoenix index table in the source cluster by using the “alter” command in the HBase shell like enabling replication for HBase and Phoenix Data tables.

   ```
   hbase> alter “BIG_INDEX”, {NAME=>”0”, REPLICATION_SCOPE=>1}
   ```

3. If you used the ASYNC keyword in the create index in Cloudera Operational Database, now you have to use IndexTool to build the index. This will launch a MapReduce job to build in the index in parallel, rather than synchronously from a single client as is normally done. An example IndexTool invocation would be done on the command line for the Cloudera Operational Database cluster (using the HBase client tarball) like the following:

   ```
   $ hbase org.apache.phoenix.mapreduce.index.IndexTool --data-table BIG_TA
   BLE --index-table
   BIG_INDEX --output-path /tmp/BIG_INDEX-files
   ```

4. Ensure the MapReduce job launched by this command completes before proceeding.

5. Validate the contents of the index table, as the index table is used for queries. For example, you can read a few rows from the index table directly:

```
jdbc:phoenix> SELECT * FROM BIG_INDEX LIMIT 10;
```

Alternatively, you could look at the explain plan for a query that uses the index (noting that BIG_INDEX is used instead of BIG_TABLE in this example)

```
jdbc:phoenix> EXPLAIN SELECT col1, col2 FROM BIG_TABLE;
```

Related Information

Replicating Phoenix Data Tables

Cloudera Public Cloud Overview

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Public Cloud.....4
Cloudera Public Cloud use cases.....5
Cloudera Public Cloud services.....5
Cloudera Public Cloud interfaces.....8
Services and favorites on Cloudera homepage.....9

Cloudera Public Cloud glossary.....14

Cloudera Public Cloud

Cloudera is a hybrid data platform designed for unmatched freedom to choose—any cloud, any analytics, any data. Cloudera consists of Public Cloud and Private Cloud.

What is Cloudera

Cloudera delivers faster and easier data management and data analytics for data anywhere, with optimal performance, scalability, and security. With Cloudera you get the value of Private Cloud and Public Cloud for faster time to value and increased IT control.

Cloudera provides the freedom to securely move applications, data, and users bi-directionally between the data center and multiple public clouds, regardless of where your data lives. All thanks to modern data architectures:

- A unified data fabric centrally orchestrates disparate data sources intelligently and securely across multiple clouds and on premises.
- An open data lakehouse enables multi-function analytics on both streaming and stored data in a cloud-native object store across hybrid multi-cloud.
- A scalable data mesh helps eliminate data silos by distributing ownership to cross-functional teams while maintaining a common data infrastructure.

With Cloudera Shared Data Experience (SDX), Cloudera offers enterprise-grade security and governance. Cloudera SDX combines enterprise-grade centralized security, governance, and management capabilities with shared metadata and a data catalog, eliminating costly data silos, preventing lock-in to proprietary formats, and eradicating resource contention. Now all users and administrators can enjoy the advantages of a shared data experience.

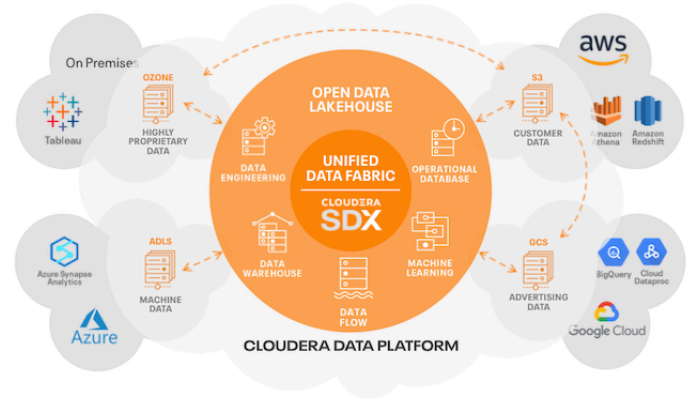

Cloudera Public Cloud

Create and manage secure data lakes, self-service analytics, and machine learning services without installing and managing the data platform software. Cloudera Public Cloud services are managed by Cloudera, but unlike other public cloud services, your data will always remain under your control in your workloads and your data will always remain under your control in your cloud account. Cloudera runs on AWS, Azure and Google Cloud.

Cloudera Public Cloud lets you:

- Control cloud costs by automatically spinning up workloads when needed, scaling them as the load changes over time and suspending their operation when complete.
- Isolate and control workloads based on user type, workload type, and workload priority.
- Combat proliferating silos and centrally control customer and operational data across multi-cloud and hybrid environments.

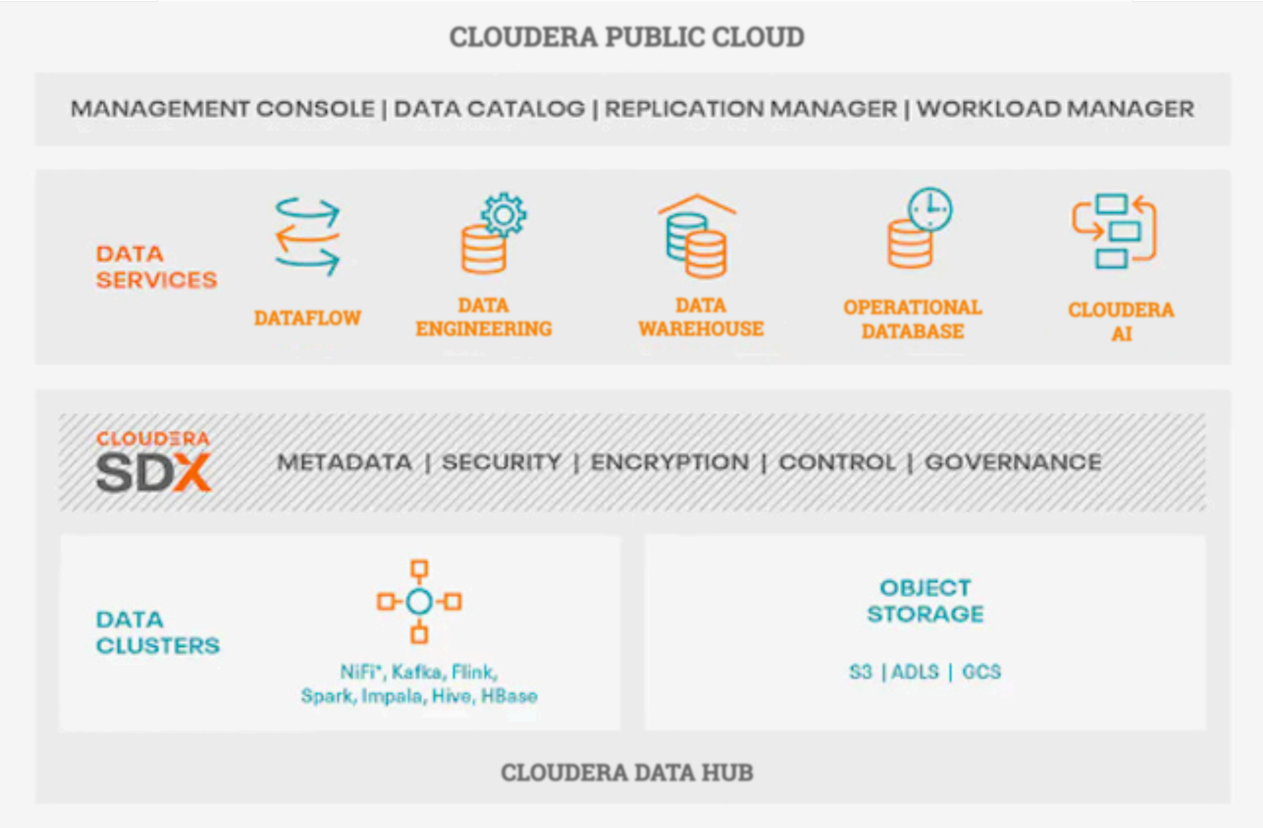

Cloudera Public Cloud use cases

Cloudera Public Cloud not only offers all the analytics experiences available on-premises, but also includes tools that enable hybrid and multi-cloud use cases. Customers benefit from leveraging a single interface whether deploying on a single cloud provider or using multiple cloud providers.

Some of the common use cases are:

- Ingest large volumes of data in real time with streaming solutions that can further enrich the data sets and curate production data that can be made available for practitioners downstream. Ensure real time data flow control to help manage transfer of data between various sources and destinations.
- Leverage object stores as centralized storage to bring together various datasets, enrich and analyze using analytical engines in Cloudera and generate a comprehensive understanding of the various entities in your value chain, thereby operationalizing a Customer 360 use case.
- Collect metrics from a variety of systems in both process or discrete manufacturing to ensure deviations are captured, modeled in real time and alerts are sent out for course-correction before it's too late.
- For companies looking to optimize cloud costs, run your workloads in the right place at the right time to reduce compute and storage costs and avoid lock-in with cloud providers.

Cloudera Public Cloud services

Cloudera Public Cloud consists of a number of cloud services designed to address specific enterprise data cloud use cases.

This includes Cloudera Data Hub powered by Cloudera Runtime, data services (Cloudera Data Warehouse, Cloudera AI, Cloudera Data Engineering, and Cloudera DataFlow), the administrative layer (Cloudera Management Console), and SDX services (Data Lake, Cloudera Data Catalog, Cloudera Replication Manager, and Cloudera Observability).

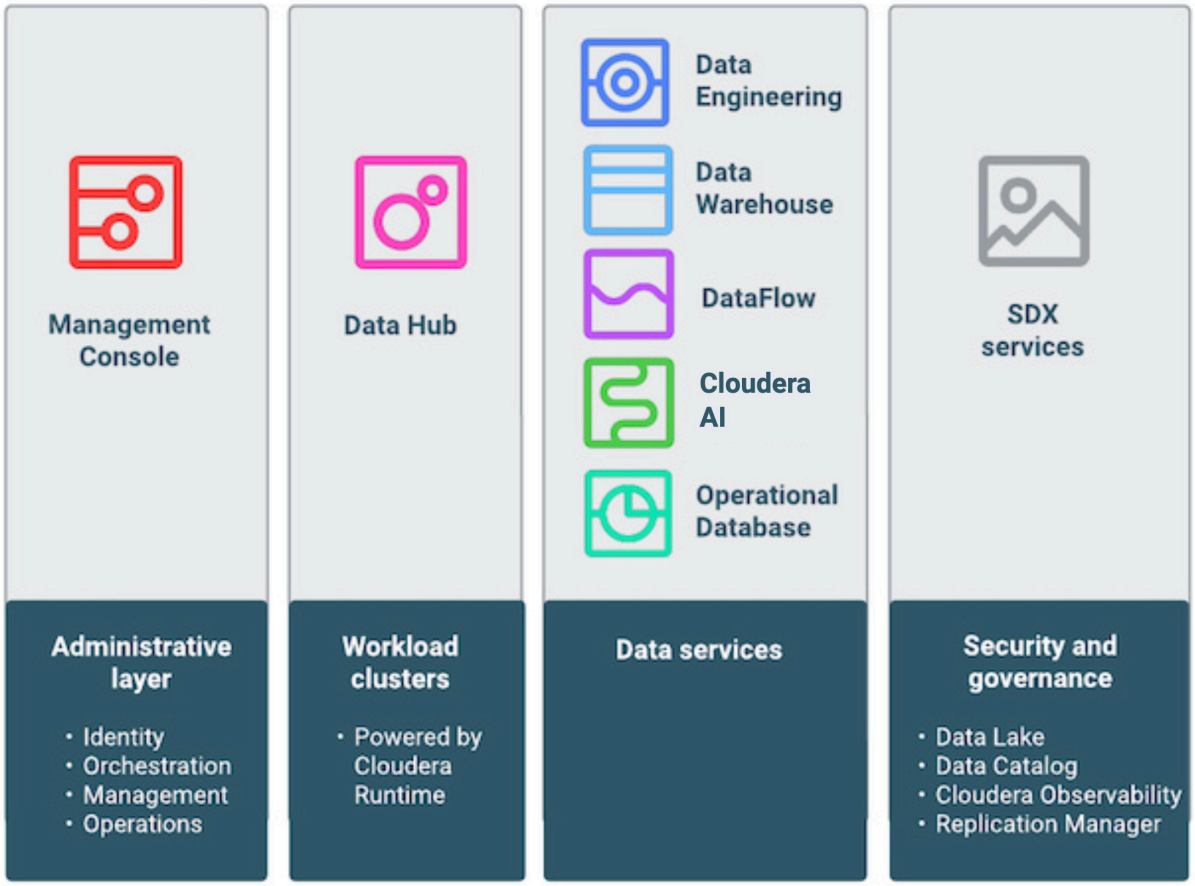

Administrative layer

Cloudera Management Console is a general service used by Cloudera administrators to manage, monitor, and orchestrate all of the Cloudera services from a single pane of glass across all environments. If you have deployments in your data center as well as in multiple public clouds, you can manage them all in one place - creating, monitoring, provisioning, and destroying services.

Workload clusters

Cloudera Data Hub is a service for launching and managing workload clusters powered by Cloudera Runtime (Cloudera's new unified open source distribution including the best of CDH and HDP). This includes a set of cloud optimized built-in templates for common workload types as well as a set of options allowing for extensive customization based on your enterprise's needs.

Cloudera Data Hub provides complete workload isolation and full elasticity so that every workload, every application, or every department can have their own cluster with a different version of the software, different configuration, and running on different infrastructure. This enables a more agile development process.

Since Cloudera Data Hub clusters are easy to launch and their lifecycle can be automated, you can create them on demand and when you don't need them, you can return the resources to the cloud.

Data services

Cloudera Data Engineering is an all-inclusive data engineering toolset building on Apache Spark that enables orchestration automation with Apache Airflow, advanced pipeline monitoring, visual troubleshooting, and comprehensive management tools to streamline ETL processes across enterprise analytics teams.

Cloudera DataFlow is a cloud-native universal data distribution service powered by Apache NiFi that lets developers connect to any data source anywhere with any structure, process it, and deliver to any destination. It offers a flow-based low-code development paradigm that aligns best with how developers design, develop, and test data distribution pipelines.

Cloudera Data Warehouse is a service for creating and managing self-service data warehouses for teams of data analysts. This service makes it easy for an enterprise to provision a new data warehouse and share a subset of the data with a specific team or department. The service is ephemeral, allowing you to quickly create data warehouses and terminate them once the task at hand is done.

Cloudera AI is a service for creating and managing self-service Cloudera AI Workbenches. This enables teams of data scientists to develop, test, train, and ultimately deploy machine learning models for building predictive applications all on the data under management within the enterprise data cloud.

Cloudera Operational Database is a service for self-service creation of an operational database. Cloudera Operational Database is a scale-out, autonomous database powered by Apache HBase and Apache Phoenix. You can use it for your low-latency and high-throughput use cases with the same storage and access layers that you are familiar with using in CDH and HDP.

Security and governance

Shared Data Experience (SDX) is a suite of technologies that make it possible for enterprises to pull all their data into one place to be able to share it with many different teams and services in a secure and governed manner. There are four discrete services within SDX technologies: Data Lake, Cloudera Data Catalog, Cloudera Replication Manager, and Cloudera Observability.

Data Lake is a set of functionality for creating safe, secure, and governed data lakes which provides a protective ring around the data wherever that’s stored, be that in cloud object storage or HDFS. Data Lake functionality is subsumed by the Cloudera Management Console service and related Cloudera Runtime functionality (Ranger, Atlas, Hive MetaStore).

Cloudera Data Catalog is a service for searching, organizing, securing, and governing data within the enterprise data cloud. Cloudera Data Catalog is used by data stewards to browse, search, and tag the content of a data lake, create and manage authorization policies (by file, table, column, row, and so on), identify what data a user has accessed, and access the lineage of a particular data set.

Cloudera Replication Manager is a service for copying, migrating, snapshotting, and restoring data between environments within the enterprise data cloud. This service is used by administrators and data stewards to move, copy, backup, replicate, and restore data in or between data lakes. This can be done for backup, disaster recovery, or migration purposes, or to facilitate dev/test in another virtual environment.

Cloudera Observability is a service for analyzing and optimizing workloads within the enterprise data cloud. This service is used by database and workload administrators to troubleshoot, analyze, and optimize workloads in order to improve performance and/or cost.

The following table illustrates which services are available on supported cloud providers:

	Cloudera service	AWS	Azure	GCP
Data services	Cloudera Data Engineering	GA	GA	Not available
	Cloudera DataFlow	GA	GA	Only DataFlow Functions are available on GCP. Other DataFlow features are not available.
	Cloudera Data Hub	GA	GA	GA
	Cloudera Data Warehouse	GA	GA	Not available
	Cloudera AI	GA	GA	Not available
	Cloudera Operational Database	GA	GA	GA
Control Plane services	Cloudera Data Catalog	GA	GA	GA

	Cloudera service	AWS	Azure	GCP
	Cloudera Management Console	GA	GA	GA
	Cloudera Replication Manager	GA	GA	GA
	Cloudera Observability	GA	GA	GA

Related Information

Management Console
Data Hub
Data Engineering
DataFlow
Data Warehouse
Machine Learning
Data Catalog
Replication Manager
Cloudera Observability

Cloudera Public Cloud interfaces

There are three basic ways to access and use Cloudera Public Cloud: web interface, CLI client, and SDK.

Web interface

The Cloudera Public Cloud web interface provides a web-based, graphical user interface. As an admin user, you can use the web interface to register environments, manage users, and provision Cloudera service resources for data practitioners. As a data practitioner, you can use the web interface to access Cloudera services.

The interactive homepage of Cloudera allows you to bookmark your favorite resources for easy access, quickly check the analytics summary of your applications, search Cloudera documentation, browse the latest blog posts and explore various solutions.

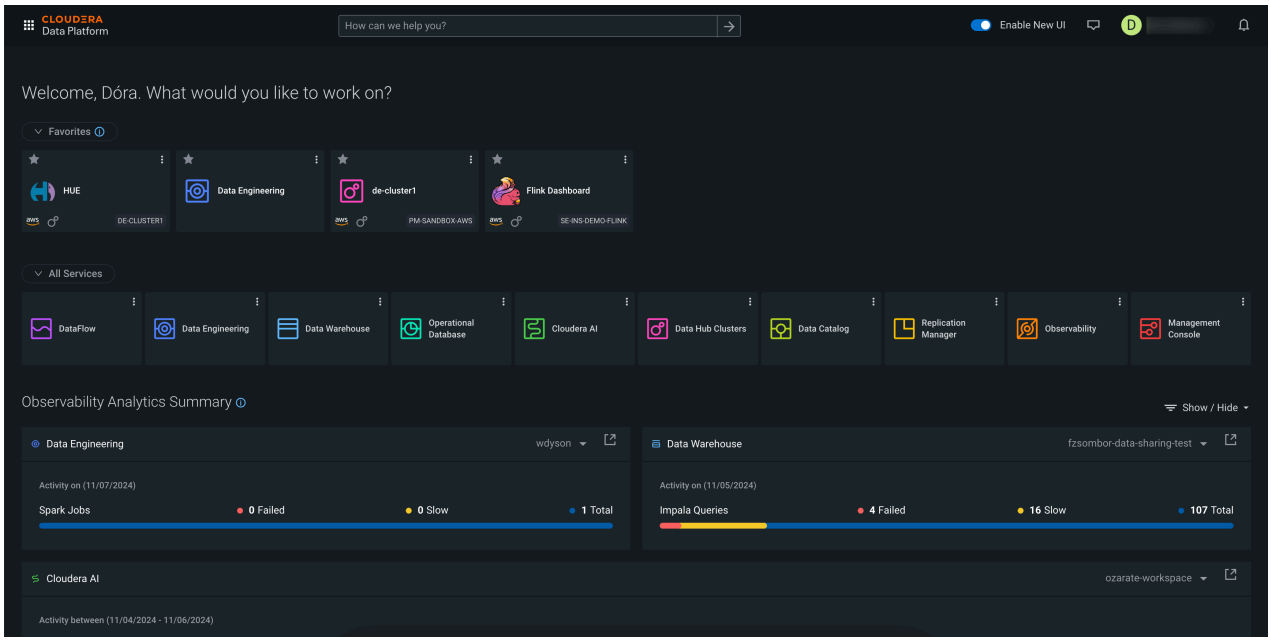

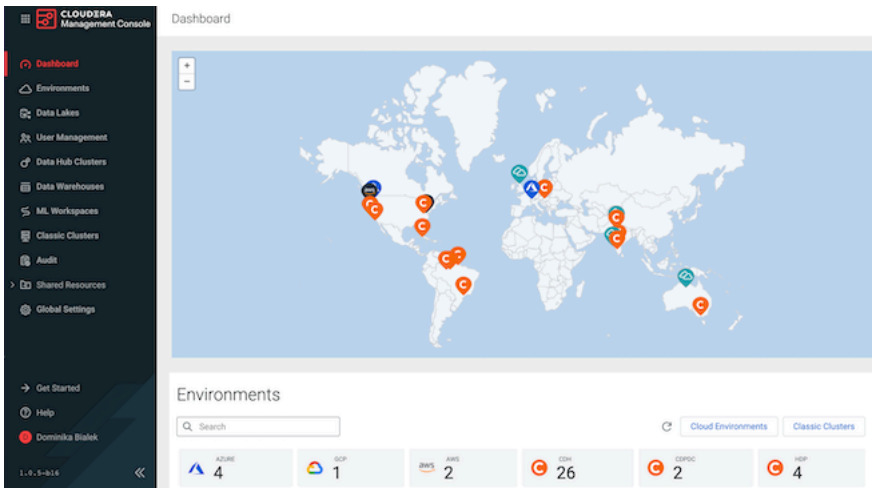

CLI

If you prefer to work in a terminal window, you can download and configure the Cloudera client that gives you access to the CDP CLI tool. The CDP CLI allows you to perform the same actions as can be performed from the web console. Furthermore, it allows you to automate routine tasks such as cluster creation.

SDK

You can use the Cloudera SDK for Java to integrate Cloudera services with your applications. Use the Cloudera SDK to connect to Cloudera services, create and manage clusters, and run jobs from your Java application or other data integration tools that you may use in your organization.

Related Information

Supported Browsers Policy

Services and favorites on Cloudera homepage

When accessing the Cloudera homepage, you can access all services, customize and bookmark your most used workspaces, Cloudera Data Hub clusters and Cloudera Data Hub services for easy access.

After enabling the new User Interface (UI) on the Cloudera homepage, you can bookmark your most used workspaces, Cloudera Data Hub clusters and Cloudera Data Hub services. When clicking Cloudera AI under **All Services**, you can use the ★ icon to bookmark the available workbenches that will appear under **Favorites**.

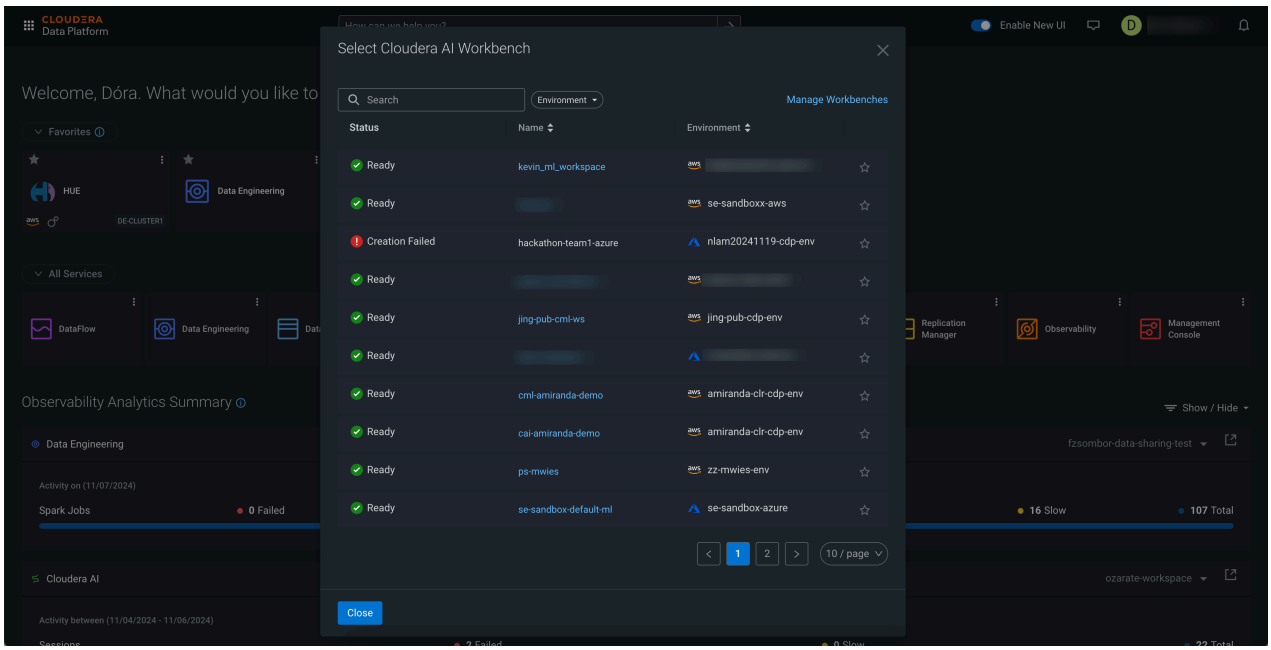

Bookmarking Cloudera Data Hub works the same way: after clicking on Data Hub Clusters under **All Services**, you can add a specific cluster or a service running in the Cloudera Data Hub cluster to your favorites.

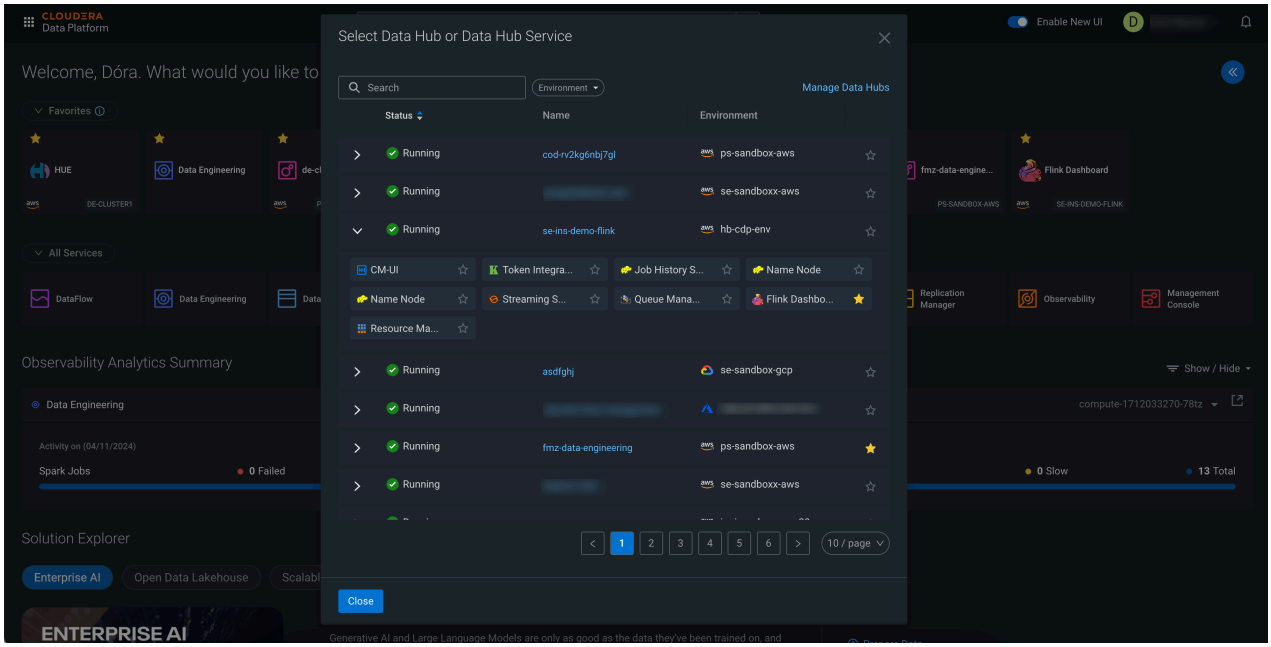

You can rearrange the bookmarked resources in your favorites by dragging them to any order you would like them to appear on Cloudera Homepage:

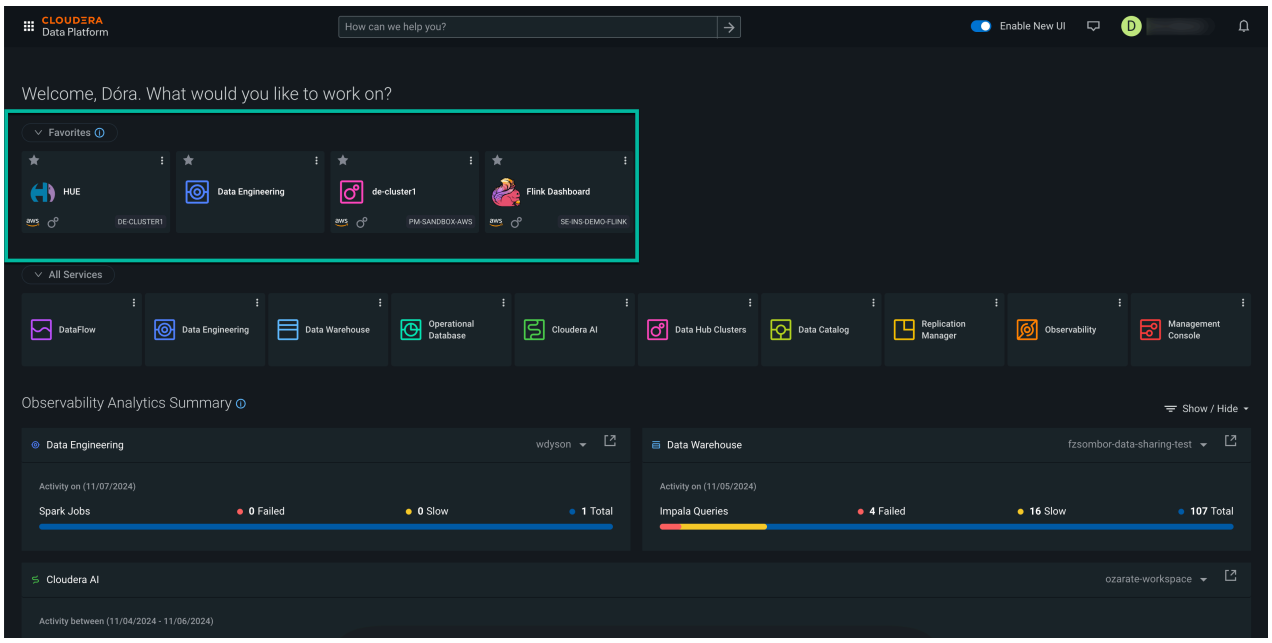

Cloudera Observability from Cloudera Homepage

You can access the Cloudera Observability Dashboard of the supported services to review the cluster and virtual warehouse information under **Observability Analytics Summary**. You can choose to display all of the available services or choose the ones you would like to appear on the Cloudera Homepage:

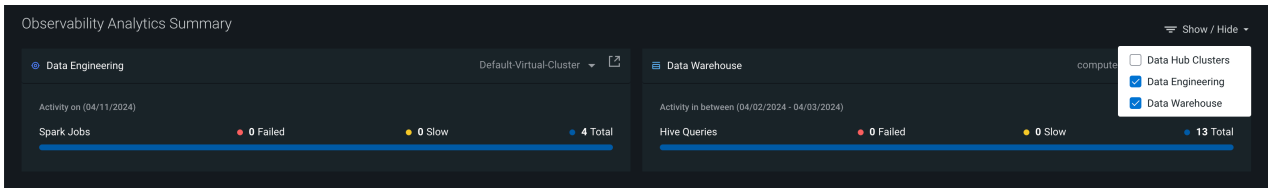

Note: You need to enable telemetry for the environment and services to display the analytical information on the Cloudera Homepage. For more information, see the Enabling environment telemetry documentation.

Notifications

The list of recent notifications and announcements can be viewed using . You can also access all of the notifications with See All Notifications, and manage the subscription settings with Subscriptions.

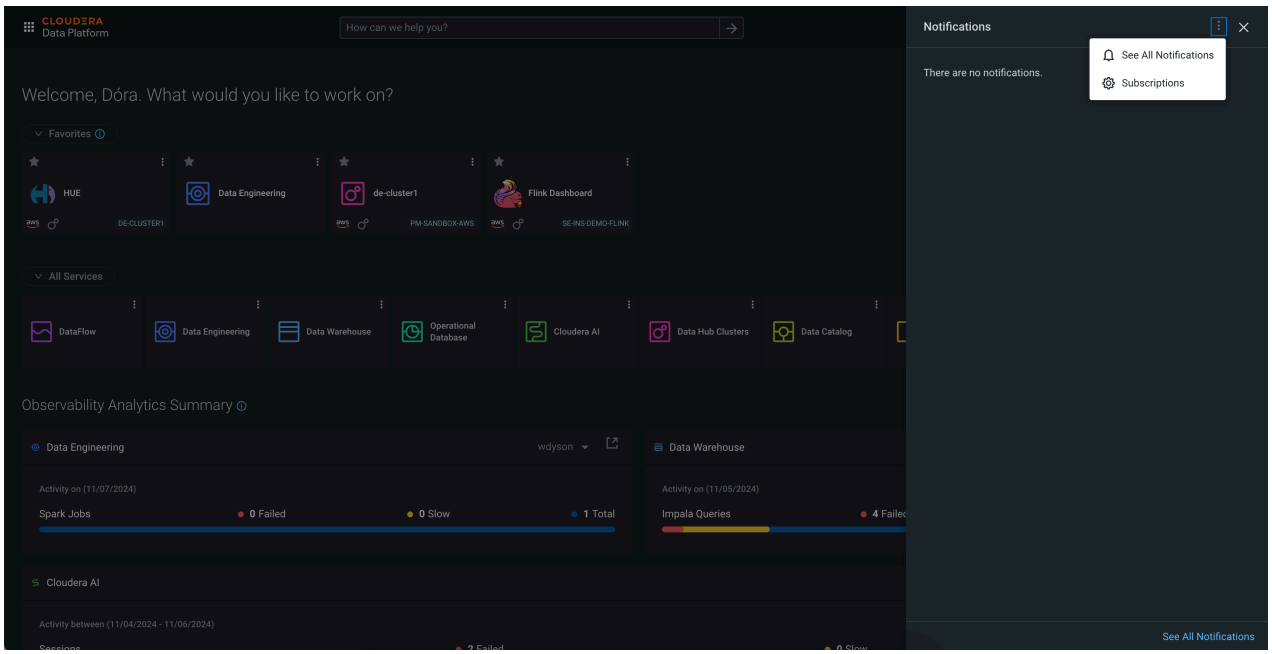

Resources and What's New

Cloudera Homepage also has multiple resources to learn more about Cloudera and its services. The **Solution Explorer** guides you through the latest Cloudera innovations with links to the documentation, resource library and related blog posts.

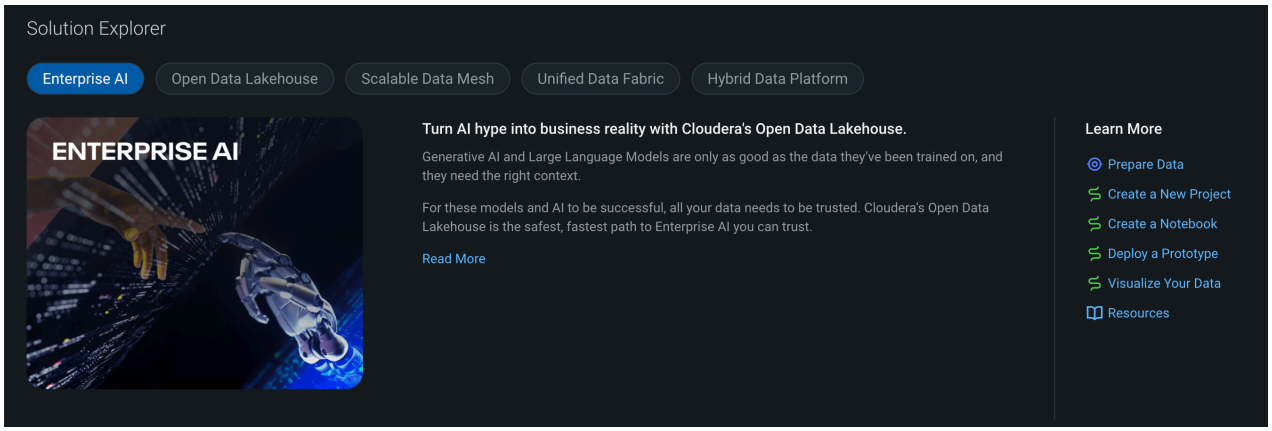

The expandable side panel has the latest blog posts from Cloudera under **What's new**. The articles under **What's new** are updated using RSS feed to ensure that the newest posts can be accessed right away.

You can also view the documentation for the top tasks under **Quick Start**, where the procedures are collected based on the Data Services.

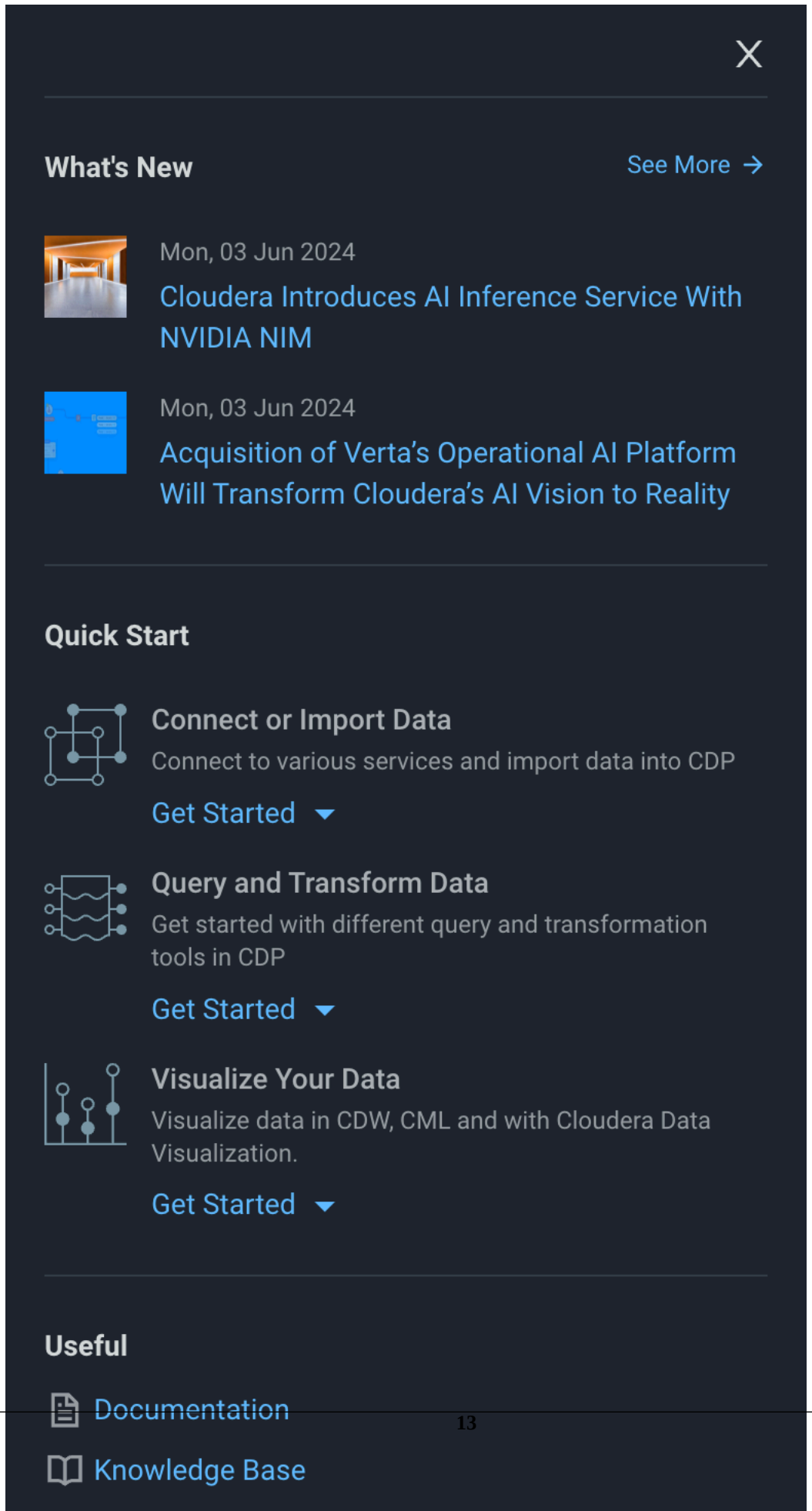
What's New
See More
Mon, 03 Jun 2024
Cloudera Introduces AI Inference Service With NVIDIA NIM
Mon, 03 Jun 2024
Acquisition of Verta's Operational AI Platform Will Transform Cloudera's AI Vision to Reality
Quick Start
Connect or Import Data
Connect to various services and import data into CDP
Get Started
Query and Transform Data
Get started with different query and transformation tools in CDP
Get Started
Visualize Your Data
Visualize data in CDW, CML and with Cloudera Data Visualization.
Get Started
Useful
Documentation
Knowledge Base

Cloudera Public Cloud glossary

Cloudera Public Cloud documentation uses terminology related to enterprise data cloud and cloud computing.

Cloudera Public Cloud - Cloudera Public Cloud is a cloud service platform that consists of a number of services. It enables administrators to deploy Cloudera service resources and allows end users to process and analyze data by using these resources.

CDP CLI - Provides a command-line interface to access and manage Cloudera services and resources.

Cloudera web console - The web interface for accessing and manage Cloudera services and resources.

Cloudera Observability (data service) - A Cloudera data service used by database and workload administrators to troubleshoot, analyze and optimize workloads in order to improve performance and/or cost.

Cloudera Runtime - The open source software distribution within Cloudera that is maintained, supported, versioned, and packaged by Cloudera. Cloudera Runtime combines the best of Cloudera and HDP. Cloudera Runtime 7.0.0 is the first version.

Cluster - Also known as compute cluster, workload cluster, or Cloudera Data Hub cluster. The cluster created by using the Cloudera Data Hub service for running workloads. A cluster makes it possible to run one or more Cloudera Runtime components on some number of VMs and is associated with exactly one data lake.

Cluster definition - A reusable cluster template in JSON format that can be used for creating multiple Cloudera Data Hub clusters with identical cloud provider settings. Cloudera Data Hub includes a few built-in cluster definitions and allows you to save your own cluster definitions. A cluster definition is not synonymous with a blueprint, which primarily defines Cloudera Runtime services.

Cluster Repair - A feature in Cloudera Management Console that enables you to select specific nodes within a node group for a repair operation. This feature reduces the downtime incurred when only a subset of the nodes are unhealthy.

Cluster template - A reusable cluster template in JSON format that can be used for creating multiple Cloudera Data Hub clusters with identical Cloudera Runtime settings. It primarily defines the list of Cloudera Runtime services included and how their components are distributed on different host groups. Cloudera Data Hub includes a few built-in blueprints and allows you to save your own blueprints. A blueprint is not synonymous with a cluster definition, which primarily defines cloud provider settings.

Control Plane - A Cloudera operated cloud service that includes services like Cloudera Management Console, Cloudera Observability, Cloudera Replication Manager and Cloudera Data Catalog. These services interact with your account in Amazon Web Services (AWS), Microsoft Azure, and Google Cloud to provision and manage compute infrastructure that you can use to manage the lifecycle of data stored in your cloud account. In addition, the Control Plane can interface with your on-premises and Private Cloud infrastructure to support hybrid cloud deployments.

Credential - Allows an administrator to configure access from Cloudera to a cloud provider account so that Cloudera can communicate with that account and provision resources within it. There is one credential per environment.

Cloudera Data Catalog (data service) - A Cloudera data service used by data stewards to browse, search, and tag the content of a data lake, create and manage authorization policies, identify what data a user has accessed, and access the lineage of a particular data set.

Cloudera DataFlow (data service) - A Cloudera data service that enables you to import and deploy your data flow definitions efficiently, securely, and at scale.

Data Lake - A single logical store of data that provides a mechanism for storing, accessing, organizing, securing, and managing that data.

Data Lake cluster - A special cluster type that implements the Cloudera Runtime services (such as HMS, Ranger, Atlas, and so on) necessary to implement a data lake that further provides connectivity to a particular cloud storage service such as S3 or ADLS.

Cloudera Data Hub (service) - A Cloudera service that administrators use to create and manage clusters powered by Cloudera Runtime.

Cloudera Data Warehouse (data service) - A Cloudera data service for creating and managing self-service data warehouses for teams of data analysts.

Data warehouse - The output of the Cloudera Data Warehouse service. Users access data warehouses via standard business intelligence tools such as JDBC or Tableau

Environment - A logical environment defined with a specific virtual network and region in a customer’s cloud provider account. One can refer to a "Cloudera environment" once a cloud provider virtual network, cloud storage, and other cloud provider artifacts present in a customer's AWS, Azure, or GCP account have been registered in Cloudera. Cloudera service components such as Cloudera Data Hub clusters, Cloudera Data Warehouse clusters, and so on, run in an environment.

Image catalog - Defines a set of images that can be used for provisioning Cloudera Data Hub cluster. Cloudera Data Hub includes a built-in image catalog with a set of built-in base and prewarmed images and allows you to register your own image catalog.

Cloudera AI (data service) - A Cloudera data service that administrators use to create and manage Cloudera AI Workbench and that allows data scientists to do their machine learning.

Cloudera AI Workbench- The output of the Cloudera AI Inference service. Each workbench corresponds to a single cluster that can be accessed by end users.

Cloudera Management Console (data service) - A Cloudera data service that allows an administrator to manage environments, users, and services; and download and configure the CLI.

Cloudera Operational Database (data service) - A Cloudera data service that administrators use to create and manage scale-out, autonomous database powered by Apache HBase and Apache Phoenix.

Recipe - A reusable script that can be used to perform a specific task on a specific resource.

Cloudera Replication Manager (data service) - A Cloudera data service used by administrators and data stewards to move, copy, backup, replicate, and restore data in or between data lakes.

Service - A defined subset of Cloudera functionality that enables a Cloudera user to solve a specific problem related to their data lake (process, analyze, predict, and so on). Example services: Cloudera Data Hub, Cloudera Data Warehouse, Cloudera AI.

Shared resources - A set of resources such as cloud credentials, recipes (custom scripts), and other that can be reused across multiple environments.

Cloudera Public Cloud

AWS Requirements

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

AWS account requirements....4

AWS permissions....4

AWS resources and services....4

AWS region....4
Supported AWS regions....4
VPC and subnets....5
Security groups....7
Default AWS security groups....10
SSH key pair....14
EC2 instances....14
Custom images....15
Cross-account access IAM role....15
IAM policy definitions....15
Create a cross-account IAM role....16
Reduced access policy definition....20
AWS cloud storage prerequisites....23
Minimal setup for AWS cloud storage....23
Onboarding Cloudera users and groups (RAZ)....33
Onboarding Cloudera users and groups (No RAZ)....33
Policy definitions - minimal....39
Policy definitions - onboarding....41
Using S3 encryption....41
Using S3 Express One Zone for data storage....42
Supported AWS block storage....44
Customer managed encryption keys....44
AWS limits....46
List of AWS resources....47

AWS outbound network destinations....52

Access to workload UIs....58

Supported browsers....58

Other resources....58

Cloudera CIDR....59

AWS account requirements

Prior to registering your AWS environment in Cloudera, use this document to verify that your AWS account has all the resources required by Cloudera and that your Cloudera administrator has adequate permissions to configure the resources and services in AWS.

Related Information

AWS permissions

AWS resources and services

Overview of AWS resources used by Cloudera

AWS outbound network access destinations

Access to workload UIs

Supported browsers

Other resources

AWS permissions

As a Cloudera administrator, you must be able to perform the following tasks in your AWS account:

- Create policies and roles in IAM.
- Create and administer VPCs, subnets, security groups, and S3 buckets.
- Perform administrative tasks in various AWS services such as EC2 and CloudFormation.

AWS Administrator privileges provides all the permissions you need to create the resources for Cloudera. For the list of AWS resources that Cloudera services use, refer to AWS resources used by Cloudera.

AWS resources and services

Cloudera uses the following resources in your AWS account.

Use the following guidelines to ensure that Cloudera has access to the resources in your AWS account and that your AWS account has all the necessary resources required by Cloudera:

AWS region

Prior to registering an environment, you should decide which AWS region you would like to use.

A single AWS environment registered in Cloudera corresponds to a single VPC located in a specific region, and all the resources deployed by Cloudera on AWS are deployed into that VPC.

Typically, you may want to deploy clusters into the same region that contains the S3 buckets that you want to access for input and output data. Therefore, when selecting the region to use, you should consider where your data is located. Furthermore, Cloudera requires that the S3 storage location provided during environment registration is in the same region as the region selected for the environment.

If you need to use multiple regions, you need to register multiple environments, one per region.

Supported AWS regions

Cloudera supports the following AWS regions.

Region Name	Region ID	Environmen	Cloudera Data Hub	Cloudera Data Warehouse	Cloudera AI	Cloudera Data Engineering	Cloudera DataFlow	Cloudera Operational Database
US East (Ohio)	us-east-2	##	##	##	##	##	##	##
US East (N. Virginia)	us-east-1 **Note:** The us-east-1e availability zone is not supported.	##	##	##	##	##	##	##
US West (N. California)	us-west-1	##	##			##	##	##
US West (Oregon)	us-west-2	##	##	##	##	##	##	##
Africa (Cape Town)	af-south-1	##	##		##		##	##
Asia Pacific (Mumbai)	ap-south-1	##	##	##	##	##	##	##
Asia Pacific (Hyderabad)	ap-south-2	##	##					
Asia Pacific (Seoul)	ap-northeast-2	##	##	##	##	##	##	##
Asia Pacific (Singapore)	ap-southeast-1	##	##	##	##	##	##	##
Asia Pacific (Sydney)	ap-southeast-2	##	##	##	##	##	##	##
Asia Pacific (Jakarta)	ap-southeast-3	##	##	#	##	##	##	#
Asia Pacific (Hong Kong)	ap-east-1	##	##		##	##	##	#
Asia Pacific (Tokyo)	ap-northeast-1	##	##	##	##	##	##	##
Canada (Central)	ca-central-1	##	##	##	##	##	##	##
Canada West (Calgary)	ca-west-1	##	##					
EU (Frankfurt)	eu-central-1	##	##	##	##	##	##	##
EU (Ireland)	eu-west-1	##	##	##	##	##	##	##
EU (London)	eu-west-2	##	##	##	##	##	##	##
EU (Milan)	eu-south-1	##	##	##	##	##	##	##
EU (Paris)	eu-west-3	##	##	##	##	##	##	##
EU (Spain)	eu-south-2	##	##					##
EU (Stockholm)	eu-north-1	##	##	##	##	##	##	##
Israel (Tel Aviv)	il-central-1	##	##					
Middle East (Bahrain)	me-south-1	##	##				##	##
Middle East (UAE)	me-central-1	##	##				##	##
South America (São Paulo)	sa-east-1	##	##	##	##	##	##	##

Related Information

Unable to view subnet created in "us-east-1e" in Cloudera console

VPC and subnets

When registering an AWS environment in Cloudera, you will be asked to select a VPC and two or more subnets.

You have two options:

- Use your existing VPC and subnets for provisioning Cloudera resources.
- Have Cloudera create a new VPC and subnets. All Cloudera resources will be provisioned into this new VPC and subnets.

Existing VPC and subnets

Verify the limits of the VPC and subnets available in your AWS account to ensure that you have enough resources to create clusters in Cloudera.

If you would like to use your own AWS VPC, it must meet the following requirements:

- The VPC has at least two subnets, each in a different availability zone.
- If you plan to expose your cluster via a public load balancer, you need to have at least one public and one private subnet in the same availability zone and your cluster must be deployed to one of those subnets.
- The VPC subnets must be connected to an Internet Gateway OR a NAT Gateway. The VPC should be able to make an outbound connection with the internet or set of CIDRs and ports provided by Cloudera.
- Cloudera supports public subnets and private subnets. For private subnets, you must enable Cluster Connectivity Manager.

  Important:

  For production deployments, Cloudera recommends that you use private subnets. Work with your internal IT teams to ensure that users can access the browser interfaces for cluster services.

  Warning:

  Be careful if deploying into an existing subnet that has a non-default Network Access Control List (NACL) applied to it. In such case, a security group at EC2 level and a NACL at subnet level are both present at the same time. This can cause a problem because the two settings are evaluated independently (For incoming traffic, the NACL set at the subnet level is evaluated first, then the security group set at the EC2 level. For outgoing traffic the evaluation is the converse), and if an "allow" rule does not exist at both levels, the traffic will not be admitted. Therefore, if you need to use such a setup, you should ensure that the "allow" rule exists on both levels.
- Ensure the CIDR block for the subnets is sized appropriately. In general there is no way to increase the subnet size without recreating the environment and VPC, although Cloudera Data Warehouse service allows you to use overlay networks.
- When using Cluster Connectivity Manager on AWS, you have an option to use Public Endpoint Access Gateway. If you would like to use it, you should have at least 2 public subnets in the VPC. The exact number of public subnets should be equal to the number of private subnets. Furthermore, availability zones of the public and private subnets must match.

Depending on the Cloudera services that you are planning to use, you may also need the following:

- If you are planning to use Cloudera DataFlow, Cloudera Data Engineering, Cloudera Data Warehouse, or Cloudera AI you must enable Amazon DNS with the VPC. Corporate DNS is not supported. For guidelines on how verify your DNS settings, refer to sections 1-3 in AWS environment requirements checklist for the Cloudera Data Warehouse service.
- If you are planning to use Cloudera DataFlow, Cloudera Data Engineering or Cloudera AI, you must tag the VPC and the subnets as shared so that Kubernetes can find them. For load balancers to be able to choose the subnets correctly, you are also required to tag private subnets with the kubernetes.io/role/internal-elb:1 tag, and public subnets with the kubernetes.io/role/elb:1 tag.
- If you want to use the private network feature with Cloudera Data Warehouse service, you must have three public subnets and three private subnets in your VPC.
- If you are planning to use the Cloudera Data Warehouse service, you must verify that the VPC has the settings listed in the AWS environment requirements checklist for the Cloudera Data Warehouse service.
- If you plan to use the private networking feature in the Cloudera Data Warehouse service, refer to Supported deployment modes for private networking in AWS and Prerequisites for private networking in AWS environments.

- If you are planning to use Cloudera DataFlow, Cloudera Data Engineering, Cloudera Data Warehouse, or Cloudera AI, you may also want to review the following AWS documentation: Amazon EKS - Cluster VPC Considerations and Creating a VPC for your Amazon EKS Cluster.

New VPC and subnets

If you choose to allow Cloudera to create a new VPC, six subnets will be created automatically. One subnet is created for each availability zone assuming three AZs per region; If a region has two AZs instead of three, then still three subnets are created, two in the same AZ.

You will need to specify a valid CIDR in IPv4 range that will be used to define the range of private IPs for EC2 instances provisioned into these subnets. Default is 10.10.0.0/16. Consider changing the IP range to correspond to corporate policies for standardized IP address ranges. The CIDR must match the <network mask>/16 pattern.

By default Cloudera creates 6 subnets (3 private and 3 public) and divides the address space as follows:

- 3 x /19 private subnets for FreeIPA, Data Lake, Cloudera Data Hub, Cloudera Data Warehouse, Cloudera AI
- 3 x /24 public subnets

You can disable creating private subnets, in which case only 3 public subnets will be created.

By default, when creating a new network, Cloudera uses public endpoints. But during environment registration you can optionally select the “Create Private Endpoints” option to use private endpoints instead of public endpoints.

If you choose to use private endpoints, make sure to review Outbound network access destinations.

VPCs can be created and managed from the VPC console on AWS. For instructions on how to create a new VPC on AWS, refer to Create and configure your VPC in AWS documentation.

Security groups

Security groups determine the inbound and outbound traffic to and from your Cloudera environment. That is, you should use security group settings to allow users from your organization access to Cloudera resources.

You have two options:

- Use your existing security groups (recommended for production)
- Have Cloudera create new security groups

You should verify the security group limits in your AWS account to ensure that you can create security groups for Cloudera.

Existing security groups

If you would like to create your own security groups, two security groups must be created: the first security group will be used for all gateway nodes and the second security group will be used for all other nodes. The gateway nodes accept incoming requests for the cluster services and so require an additional port. These security groups will be applied when creating a data lake and FreeIPA during environment creation and when you create Cloudera Data Hub clusters.

Review the following guidelines prior to adding security groups rules. This describes all the inbound ports that need to be open and provides guidelines for what to enter as a source range:

Note: The communication via TCP/UDP 0-65535 and ICMP is essential for healthy operation of CDP environments, Data Hubs, and data services running within the VPC, so ensure that you open these ports as described below. While some services only need well-known fixed ports, a majority of them depend on ephemeral (i.e. dynamically or randomly allocated) ports; This is why the wildcard 0-65535 TCP/UDP port range is used in the absence of a detailed breakdown of individual ports. Since overall access to the VPC is typically secured by other means, the use of the wildcard rules does not pose a higher risk against external attacks.

Gateway security group

This is used for gateway nodes:

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts. You should open it to your organization's CIDR.
TCP	443	Your CIDR and CDP CIDR	This port is used to access the Data Lake and Data Hub cluster UIs via Knox gateway. You must open this port to your organization's CIDR in order to access cluster UIs. When CCM is enabled, you only need to set this to your CIDR.
TCP	9443	CDP CIDR	This port is used by CDP to maintain management control of clusters and data lakes. By default, when CDP creates the security groups automatically, it opens this port to the correct IP. This port is not needed when CCM is enabled.
TCP, UDP	0-65535	Your internal VPC CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VPC.
ICMP	N/A	Your internal VPC CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VPC.

Example rules provided in the VPC console on AWS:

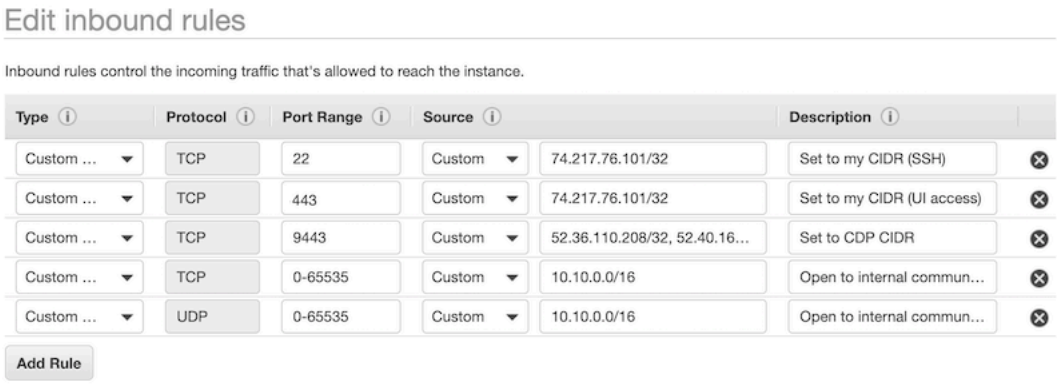

Default security group

This is used for all nodes except Knox gateway nodes:

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to the hosts. You should open it to your organization's CIDR.

Protocol	Port Range	Source	Description
TCP	443	Your CIDR	This port is only required if you are planning to spin up Machine Learning workspaces since HTTPS access to ML workspaces is available over port 443. If you are not planning to use the Machine Learning service, you do not need to open this port.
TCP	9443	CDP CIDR	This port is used by CDP to maintain management control of clusters and data lakes. By default, when CDP creates the security groups automatically, it opens this port to the correct IP. This port is not needed when CCM is enabled.
TCP, UDP	0-65535	Your VPC CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VPC. TCP port 5432 is used by the Data Lake for communication with its attached database.
ICMP	N/A	Your internal VPC CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VPC.

Example rules provided in the VPC console on AWS:

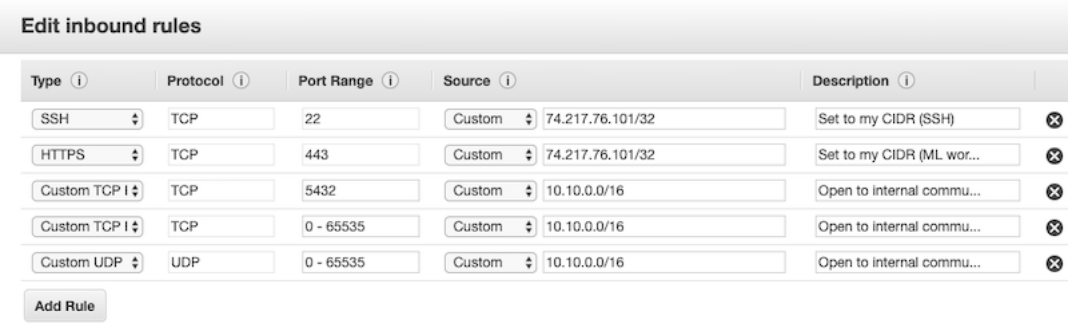

On AWS, you can create security groups and edit their rules from the VPC console > Security Groups.

To create a security group, click on Create security group and provide the following:

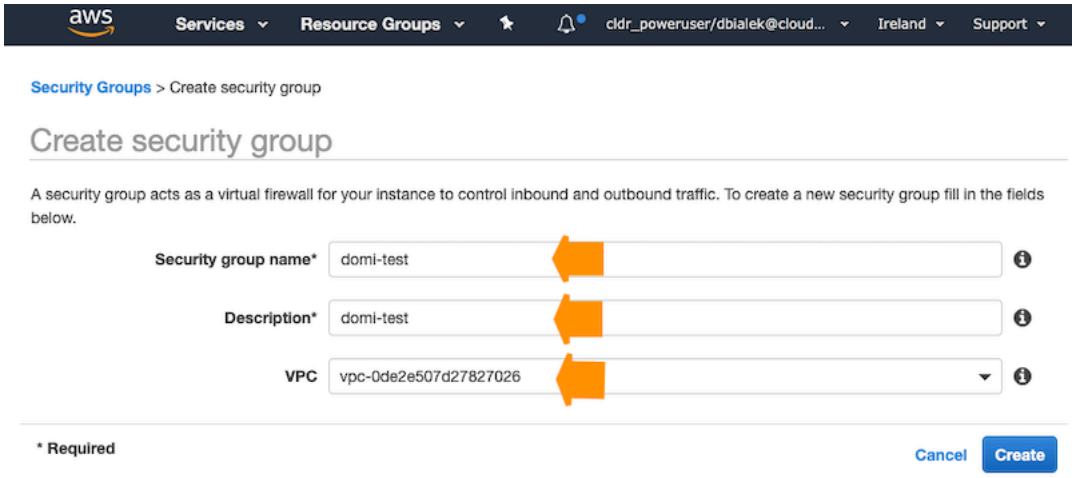

You need to create two security groups: Knox and Default (You will see this terminology in the Cloudera Management Console UI and CLI, so if you decide to choose different names, make sure that you are able to distinguish between the two security groups).

Use the guidelines and examples provided above when editing rules.

To edit security group rules, select the security group and click on Inbound Rules > Edit rules:

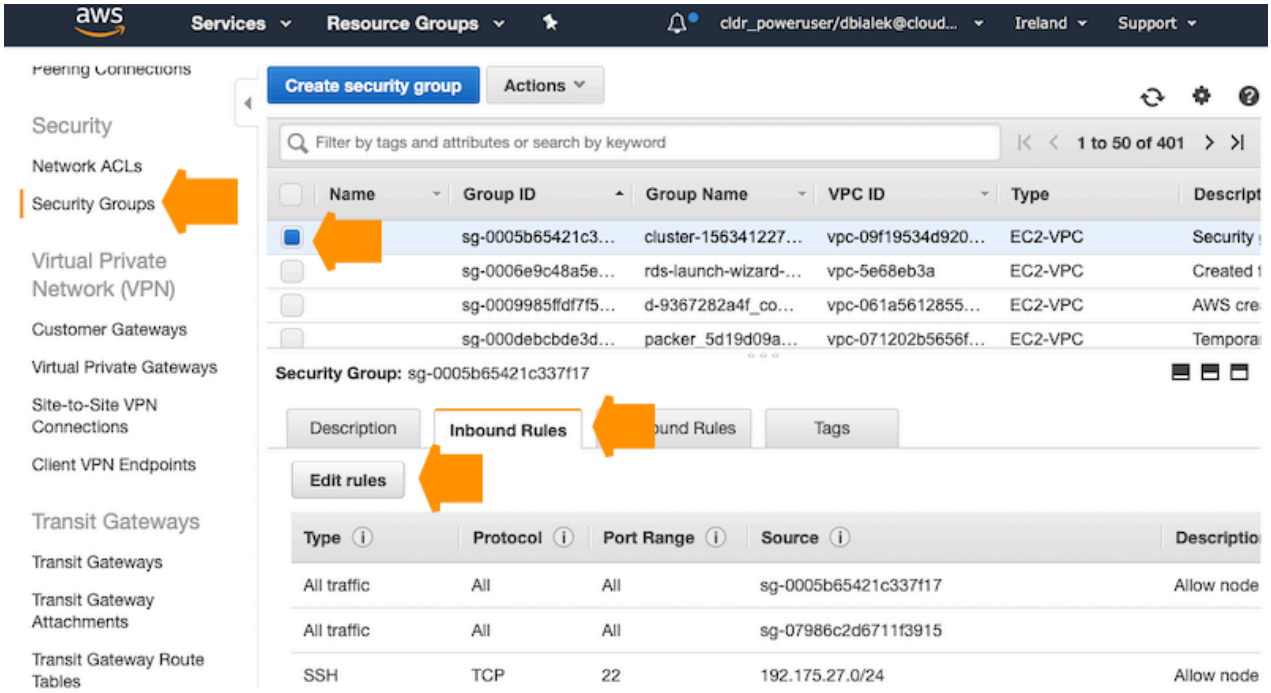

Note:

There is a known issue where even if you create and specify your own security groups, the Cloudera Data Warehouse and Cloudera AI services create their own security groups. For instructions on how to restrict access on the security groups created by the Cloudera Data Warehouse service, refer to Restricting access to endpoints in AWS environments.

New security groups

If you would like Cloudera to create the security groups for you, you need to provide a CIDR range for inbound traffic to EC2 instances from your organization. Cloudera creates multiple security groups: one for each Data Lake host group, one for each FreeIPA host group, and one per host group when Cloudera DataFlow, Cloudera Data Hub, Cloudera Data Warehouse, and Cloudera AI clusters are created. On these security groups, Cloudera opens ports as described in Default security group settings documentation.

Note:

There is a known issue where even if you create and specify your own security groups, the Cloudera Data Warehouse and Cloudera AI services create their own security groups. For instructions on how to restrict access on the security groups created by the Cloudera Data Warehouse service, refer to Restricting access to endpoints in AWS environments.

Related Information

Restricting access for Cloudera services that create their own security groups on AWS

Default security group settings on AWS

Depending on what you chose during environment creation, Cloudera can create security groups for your environment automatically or you can provide your own security groups.

Note:

Even if you create and specify your own security groups, the Cloudera Data Engineering, Cloudera Data Warehouse, and Cloudera AI services create their own security groups. Refer to Restricting access for Cloudera services that create their own security groups on AWS for instructions on how to restrict access.

Note: The communication via TCP/UDP 0-65535 and ICMP is essential for healthy operation of CDP environments, Data Hubs, and data services running within the VPC, so ensure that you open these ports as described below. While some services only need well-known fixed ports, a majority of them depend on ephemeral (i.e. dynamically or randomly allocated) ports; This is why the wildcard 0-65535 TCP/UDP port range is used in the absence of a detailed breakdown of individual ports. Since overall access to the VPC is typically secured by other means, the use of the wildcard rules does not pose a higher risk against external attacks.

Environment security groups

Depending on what you chose during environment creation, Cloudera can create security groups for your environment automatically or you can provide your own security groups.

- If you choose to use your own security groups, you are asked to create Knox and Default security groups as described in the Security groups documentation.
- If you choose for Cloudera to create all security groups required for an environment, the following security groups are created:

Data Lake: master

AWS naming convention: ${environment-name}-${random-id}-ClusterNodeSecurityGroupmaster-${random-id}

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts. You should open it to your organization's CIDR.
TCP	443	Your CIDR and Cloudera CIDR	This port is used to access the Data Lake and Cloudera Data Hub cluster UIs via Knox gateway. You should open it to your organization's CIDR in order to access cluster UIs. This port is also required if you are planning to spin up Cloudera AI workbenches since HTTPS access to Cloudera AI workbenches is available over port 443. If you are not planning to use the Cloudera AI service, you do not need to open this port. When Cluster Connectivity Manager is enabled, you only need to set this to your CIDR.
TCP	9443	Cloudera CIDR	This port is used by Cloudera to maintain management control of clusters and data lakes. This port is not used when Cluster Connectivity Manager is enabled.
TCP, UDP	0-65535	Your VPC's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VPC.
ICMP	N/A	Your internal VPC CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VPC.

Data Lake: IDBroker

AWS naming convention: ${environment-name}-${random-id}-ClusterNodeSecurityGroupidbroker-${random-id}

Azure naming convention: idbroker${dl-name}${numeric-id}sg

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts.
TCP, UDP	0-65535	Your VPC's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VPC.
ICMP	N/A	Your internal VPC CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VPC.

FreeIPA

AWS naming convention: ${environment-name}-freeipa-${random-id}-ClusterNodeSecurityGroupmaster-${random-id}

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts. You should open it to your organization's CIDR.
TCP	9443	Cloudera CIDR	This port is used by Cloudera to maintain management control of clusters and data lakes. This port is not used when Cluster Connectivity Manager is enabled.
TCP, UDP	0-65535	Your VPC's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VPC.
ICMP	N/A	Your internal VPC CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VPC.

Database

AWS naming convention: dsecg-dbsvr-${random-id}

Protocol	Port Range	Source	Description
TCP	5432	Your VPC's CIDR (for example 10.10.0.0/16)	This port is used for communication between the Data Lake and its attached database.

Cloudera Data Hub security groups

Depending on what you chose during environment creation, Cloudera can create security groups for your Data Hub clusters automatically or it can use your pre-created security groups:

- If during environment creation, you provided your own security groups, Cloudera uses these security groups when deploying clusters.
- If during environment creation you chose for Cloudera to create new security groups, new security groups are created for each Cloudera Data Hub cluster as follows:

Cloudera Data Hub: master

AWS naming convention: ${cluster-name}-${random-i}-ClusterNodeSecurityGroupmaster-${random-id}

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts. You should open it to your organization's CIDR.
TCP	443	Your CIDR and Cloudera CIDR	This port is used to access the Data Lake and Cloudera Data Hub cluster UIs via Knox gateway. You should open it to your organization's CIDR in order to access cluster UIs. When Cluster Connectivity Manager is enabled, you only need to set this to your CIDR.
TCP	9443	Cloudera CIDR	This port is used by Cloudera to maintain management control of clusters and data lakes. This port is not used when Cluster Connectivity Manager is enabled.
TCP, UDP	0-65535	Your VPC's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VPC.
ICMP	N/A	Your internal VPC CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VPC.

Cloudera Data Hub: worker

AWS naming convention: ${cluster-name}-${random-id}-ClusterNodeSecurityGroupworker-${random-id}

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts.
TCP, UDP	0-65535	Your VPC's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VPC.
ICMP	N/A	Your internal VPC CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VPC.

Cloudera Data Hub: compute

AWS naming convention: ${cluster-name}-${random-id}-ClusterNodeSecurityGroupcompute-${random-id}

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts.
TCP, UDP	0-65535	Your VPC's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VPC.
ICMP	N/A	Your internal VPC CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VPC.

Cloudera Data Warehouse security groups

Cloudera always creates new security groups when Cloudera Data Warehouse are deployed.

Cloudera AI security groups

Cloudera always creates new security groups when Cloudera AI workbenches are deployed.

Cloudera Data Engineering security groups

Cloudera always creates new security groups when Cloudera Data Engineering clusters are deployed.

Cloudera DataFlow security groups

Cloudera always creates new security groups when Cloudera DataFlow environments are enabled.

SSH key pair

When registering an environment, you will be asked to provide an SSH public key for which you have a matching private key. The minimum SSH key size is 2048 bits.

You have two options:

- Select a public SSH key that is already in EC2 in the region that you would like to use.
- Upload a public SSH key from your computer.

The SSH public key will be used for root-level access to Data Lake and Cloudera Data Hub instances. Only those users who have the corresponding private key would be able to login as an admin user.

On AWS, you can create or import SSH keys in the EC2 console > Key Pairs in the specified region by using the Create Key Pair or Import Key Pair options:

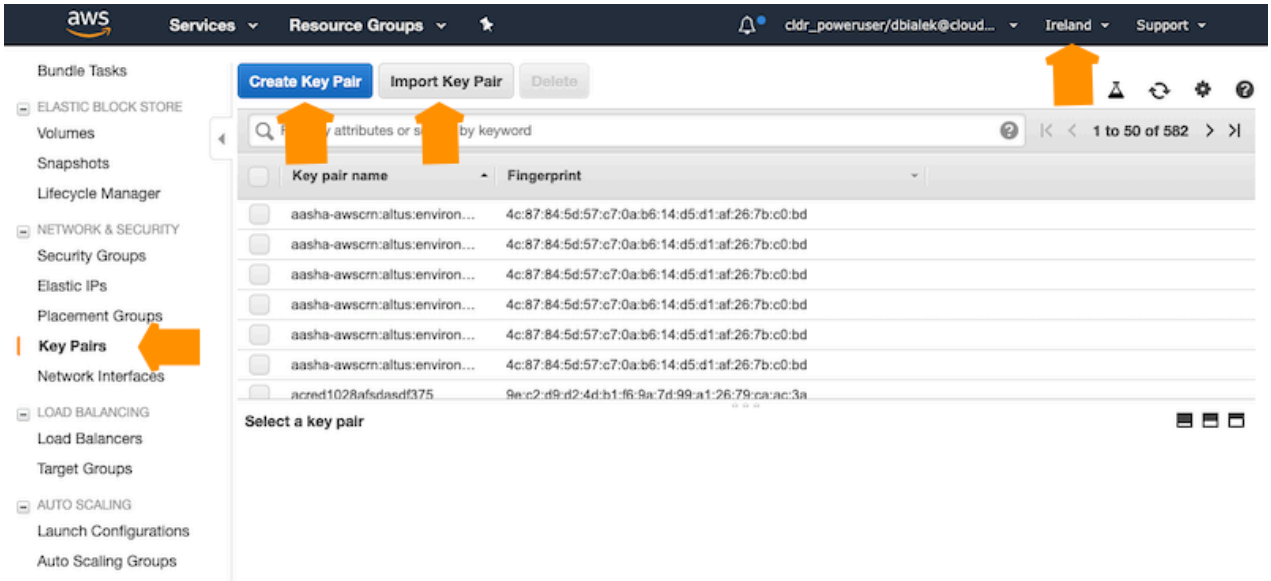

For instructions on how to create or import your SSH key on AWS, refer to Creating a key pair using Amazon EC2 and Importing your own public key to Amazon EC2 in AWS documentation.

EC2 instances

Cloudera provisions EC2 instances as part of environment creation process (for Data Lake and FreeIPA) and for compute clusters.

Therefore, you should verify the limits on the number and type of EC2 instances in your AWS account to ensure that you are able to provision an environment and create clusters in Cloudera DataFlow.

Cloudera supports Amazon EC2 reserved instances; That is, If you have purchased reserved instances, Cloudera uses them automatically according to AWS policy.

For a list of supported EC2 instance types, refer to Cloudera Public Cloud service rates.

EC2 instances for Cloudera Data Warehouse clusters

On AWS, you can access EC2 instance metadata for your Cloudera Data Warehouse cluster from a running instance using only Instance Metadata Service V2 (IMDSv2). Cloudera provisions instances for a Cloudera Data Warehouse cluster with IMDSv2. Cloudera no longer supports IMDSv1 for Cloudera Data Warehouse. Cloudera has removed the capability to access IMDSv1 for a Cloudera Data Warehouse cluster in Cloudera.

ECS instances for Cloudera Data Hub and other clusters

On AWS, you can access EC2 instance metadata for Cloudera Data Hub and clusters other than Cloudera Data Warehouse from a running instance using IMDSv1 or IMDSv2. Cloudera supports IMDSv1 but does not support IMDSv2 for these clusters, so you should not enable IMDSv2 alone for any EC2 instances used by Cloudera. For information about configuring IMDS, refer to Configuring the instance metadata service.

Custom images

By default Cloudera provides a set of default images that are used for all provisioned VMs, but you can optionally use custom images for Data Lake, FreeIPA, and Cloudera Data Hub.

You might require a custom image for compliance or security reasons (a “hardened” image), or to have your own packages pre-installed on the image, for example monitoring tools or software.

If you would like to use custom images instead of the default images, refer to Custom images and image catalogs.

Cross-account access IAM role

To allow Cloudera to create resources in your AWS account, you create a cross-account access IAM role in your AWS account and grant Cloudera access to the role as a trusted principal by specifying a specific AWS account and an external ID.

The policy for the cross-account access IAM role must have the permissions enumerated in the documentation linked below. In addition, the IAM role must reference the specific AWS account ID and external ID provided in the Cloudera Management Console.

IAM policy definitions

When creating an IAM policy for the provisioning credential, use the following IAM policy definitions.

You have two options:

Option	Description	Policies
Default policy	This policy is appropriate for registering a Cloudera environment and provisioning all Cloudera services. This is the same IAM policy as the Default policy provided in the Cloudera Management Console web interface.	• Default policy

Option	Description	Policies
Reduced access policies	Instead of the extensive default policy, you can use minimal reduced access policies. Note that the first policy is required for provisioning a Cloudera environment. The remaining policies are optional and are only needed if you would like to use the specific Cloudera services.	• Minimal policy for Cloudera environment, Cloudera Data Hub, and Cloudera Operational Database • Minimal policy for Cloudera Data Engineering • Minimal policy for Cloudera Data Warehouse • Minimal policy for Cloudera DataFlow • AWS IAM restricted roles and policies for compute and Cloudera AI • Minimal policy for using customer managed encryption keys • Compute cluster restricted permissions

Create a cross-account IAM role

In order to use role-based authentication, you must create an IAM role on AWS.

Before you begin

Prior to creating a cross-account IAM role on AWS, log in to the Cloudera web interface and obtain the parameters that need to provide for the IAM role:

1. In the Cloudera Management Console, navigate to Shared Resources Credentials Create Credential :

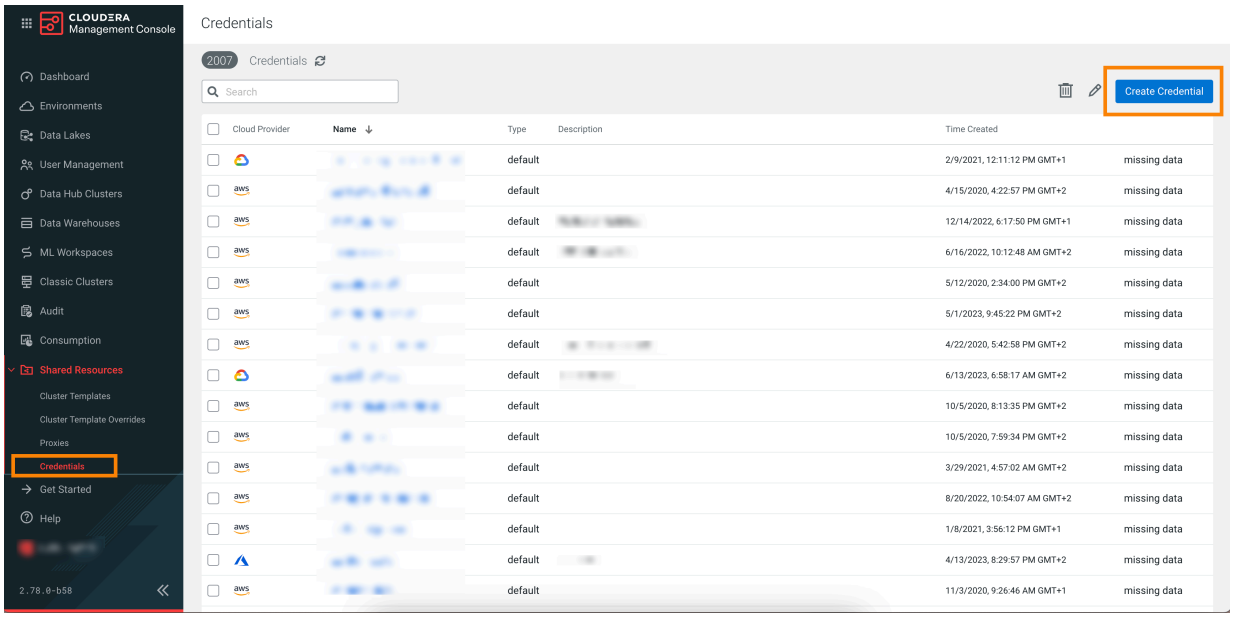

2. Note or copy the Cross-account Access Policy, Account ID, and External ID listed in the UI:

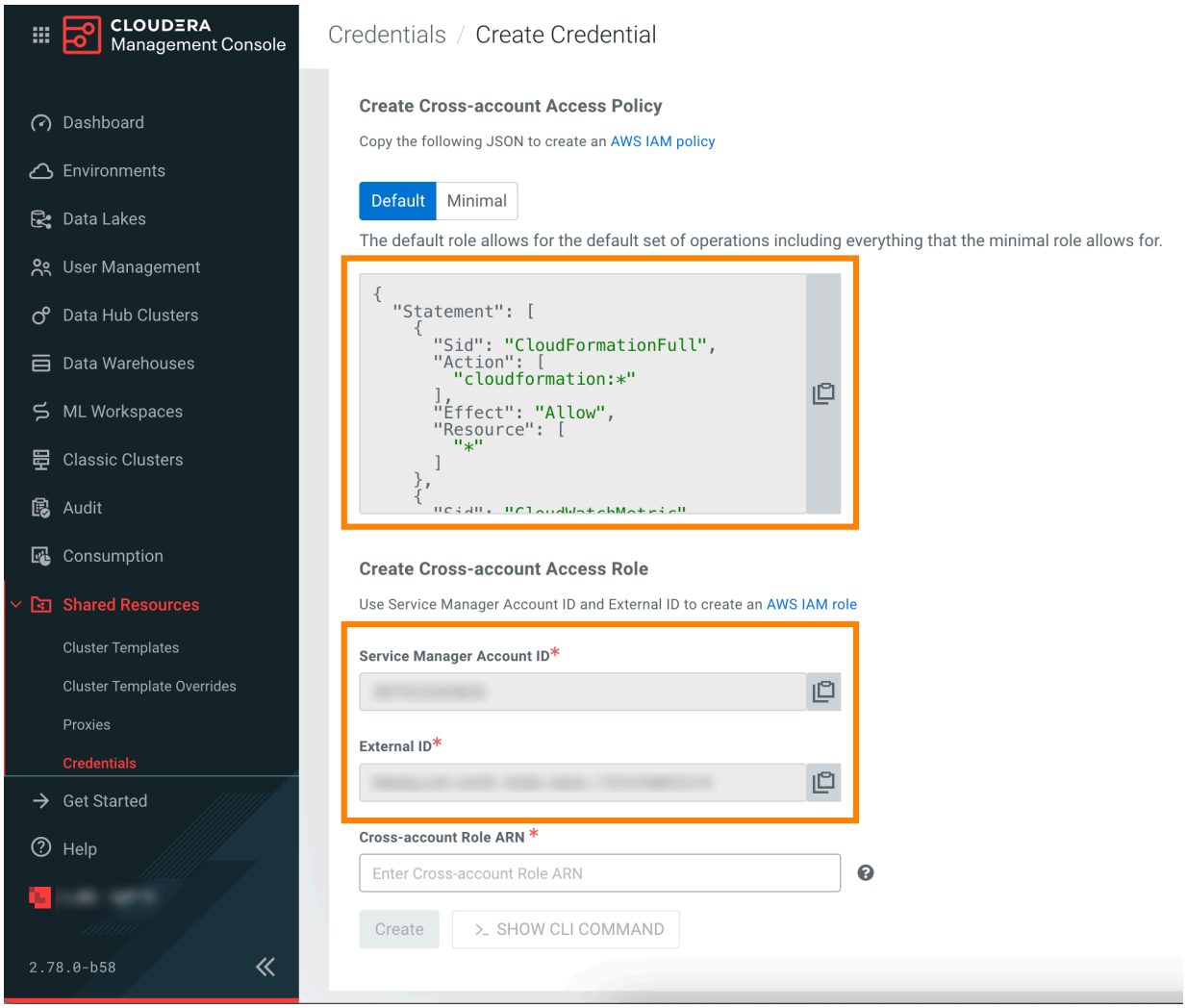

Note:

The external ID is tied to the Cloudera user who obtained it. Therefore, only the user who obtained the external ID is able to complete the credential creation flow in Cloudera with a given external ID. If a Cloudera user tries using an IAM role with an external ID obtained by another Cloudera user, Cloudera will return an error message.

You will need them to complete the following steps.

Procedure

1. Log in to the AWS Management Console.

2. Navigate to the IAM console Policies and click Create policy:

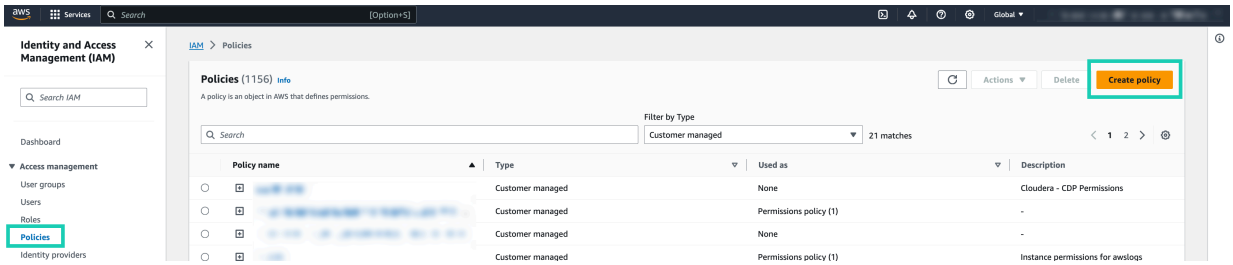

a) Select the JSON tab and paste the Cross-account Access Policy to the **Policy editor**.

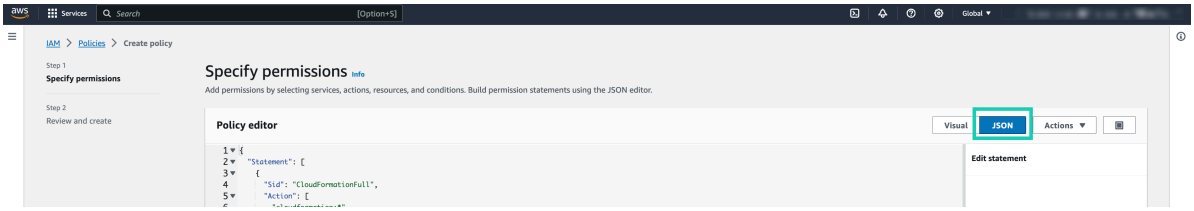

b) Click Next.

c) Enter a name for your policy in the Name field of the **Review policy** page:

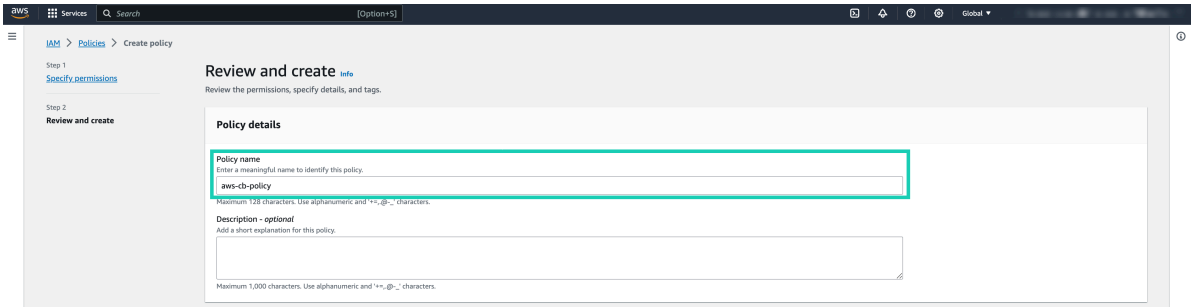

d) Click Create policy when finished.

3. Navigate to the IAM console Roles and click Create role:

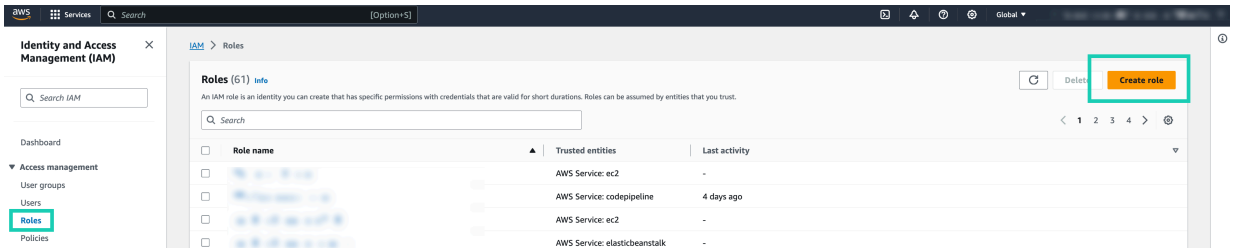

4. Select AWS account as **Trusted Entity type**.

5. Check Another AWS account and provide the following information:
   - Copy and paste your Account ID provided in Cloudera to the Accound ID field.
   - Check Require external ID under **Options**.
   - Copy and paste the External ID from Cloudera to the External ID field.

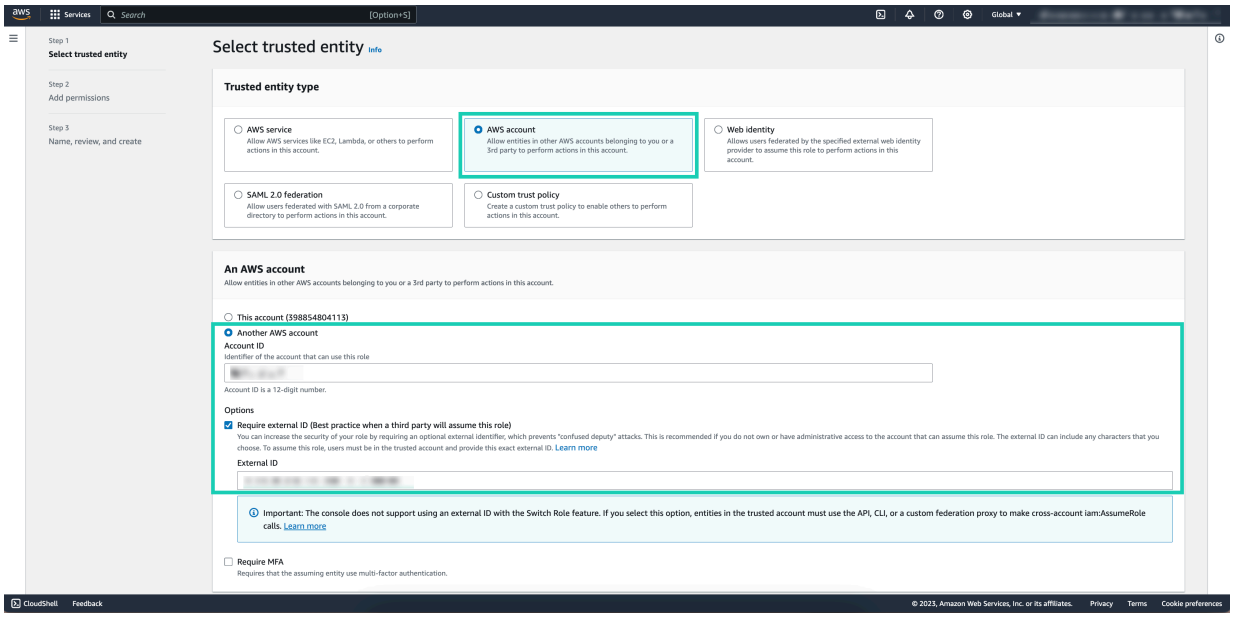

6. Click Next.
7. Search for the previously created policy, and add the permission to the role by checking the box.

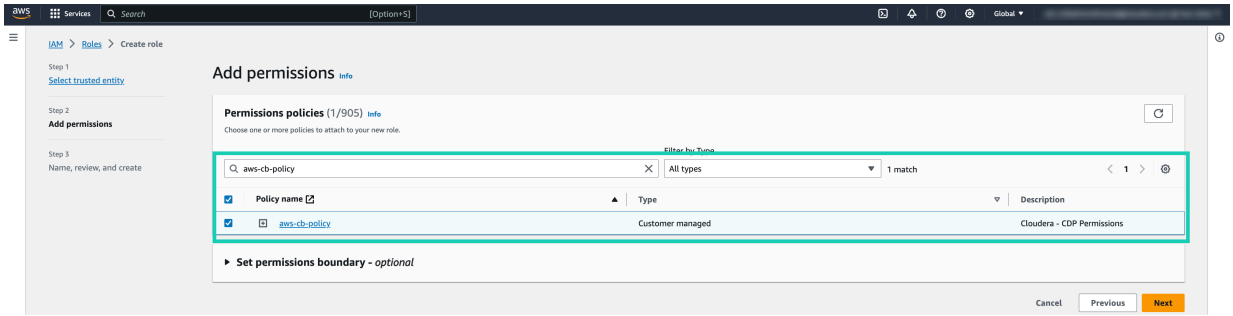

8. Click Next.
9. Enter a role name to the Role name field:

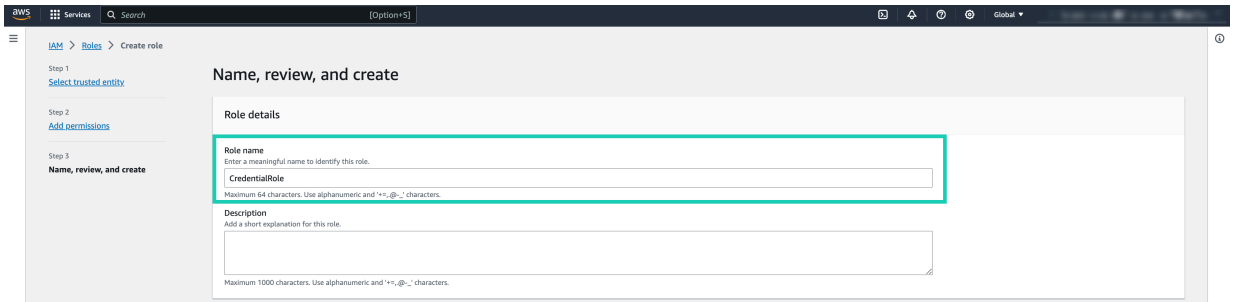

10. Click Create role to finish the role creation process.

11. Search for the created role, and obtain the IAM Role ARN. You will need it to create a role-based credential:

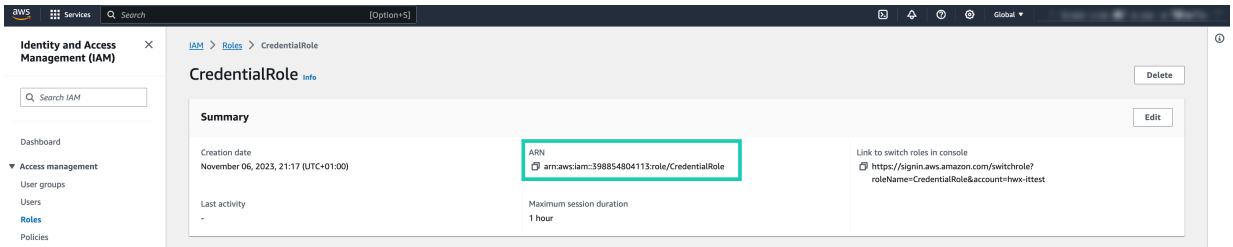

What to do next

Once you are done creating the IAM role on AWS, create a role-based credential in Cloudera.

Reduced access policy definition

The following reduced policy definition can optionally be used to register a Cloudera environment and create Cloudera Data Hub clusters and Cloudera Operational Database databases in it.

Note: This policy is only sufficient for registering a Cloudera environment and creating Cloudera Data Hub clusters and Cloudera Operational Database databases in it. In order to use other Cloudera services, you must add additional service-specific permissions to your cross-account role.

The IAM role used for the provisioning credential can use the reduced access policy instead of the default policy.

You should:

1. Copy this policy.
2. Update it as described below.
3. Once done, you can create the cross-account role using this policy.

Customize the scope of iam:PassRole

Cloudera utilizes the iam:PassRole as part of the cloud identity federation setup process via IDBroker on AWS. The default policy uses “*” as the resource scope for this action:

```
{
     "Effect": "Allow
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "*"
}
```

However the scope can be reduced to just a few roles. To do this, you need to edit the following block and replace the “*” with the ARNs of the following roles used for the environment:

- IDBROKER_ROLE
- LOG_ROLE
- RANGER_AUDIT_ROLE
- DATALAKE_ADMIN_ROLE

You choose the name for these roles when you set up Cloudera as part of the Minimal Setup for Cloud Storage.

After updating according to the minimal setup for cloud storage, the iam:PassRole policy excerpt should like similar to the following:

```
{
      "Effect": "Allow
      "Action": [
        "iam:PassRole"
```

```
        ],
        "Resource": [
                "arn:aws:iam::$ACCOUNTID:role/IDBROKER_ROLE",
                "arn:aws:iam::$ACCOUNTID:role/LOG_ROLE",
                “arn:aws:iam::$ACCOUNTID:role/RANGER_AUDIT_ROLE",
                "arn:aws:iam::$ACCOUNTID:role/DATALAKE_ADMIN_ROLE"
            ]
}
```

If you have already created these roles and you don’t know the name of these roles, you can obtain them using the following steps:

IDBROKER_ROLE

1. Log in to Cloudera web interface and navigate to the Cloudera Management Console service > Data Lakes.
2. Scroll down to where you see Event History and select the Hardware tab.
3. Locate the EC2 instance for the idbroker node (if you are running Medium Duty Data Lake, you will see two IDBroker nodes, you can use either):

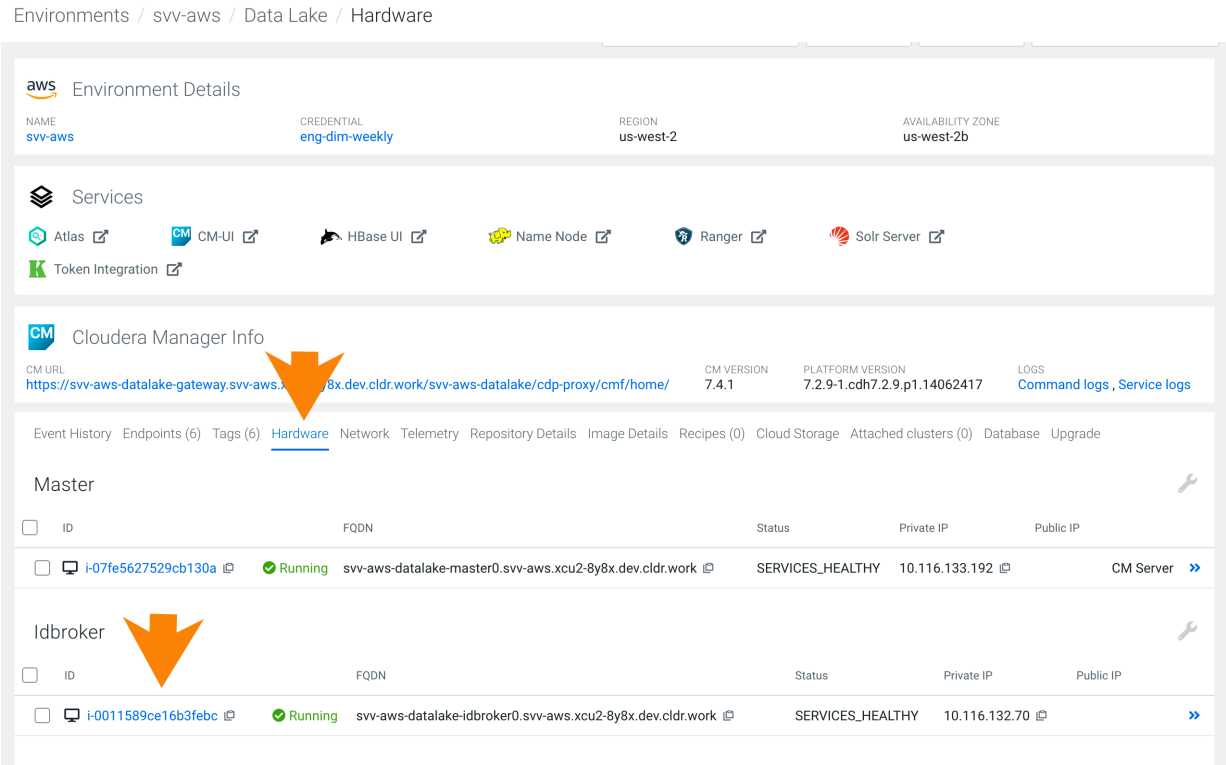

4. Click on the instance id. You will be redirected to the EC2 console.
5. In the Instance Details section for this node, you can see the name of the IAM role being used by this instance. Click on this role to get to the details.
6. Copy the ARN of the instance profile. It follows the following naming convention: arn:aws:iam::<12-digit-AWS-account-id>:instance-profile/<name-of-idbroker-role>

LOG_ROLE

1. Navigate to Environments .
2. Click on the existing environment
3. Click on the Summary tab.
4. Scroll to Logs Storage and Audits and you will see the Instance Profile entry.

5. Copy the ARN of the Instance Profile:

RANGER_AUDIT_ROLE and DATALAKE_ADMIN_ROLE

1. Navigate to Environments.
2. Click on the existing environment.
3. From the Actions menu, select Manage Access.
4. Click on the IDBroker Mappings tab.
5. Under Current Mappings you will see the mappings for the Data Access Role and Ranger Audit Role.
6. Copy the ARNs of the Data Access Role and Ranger Audit Role:

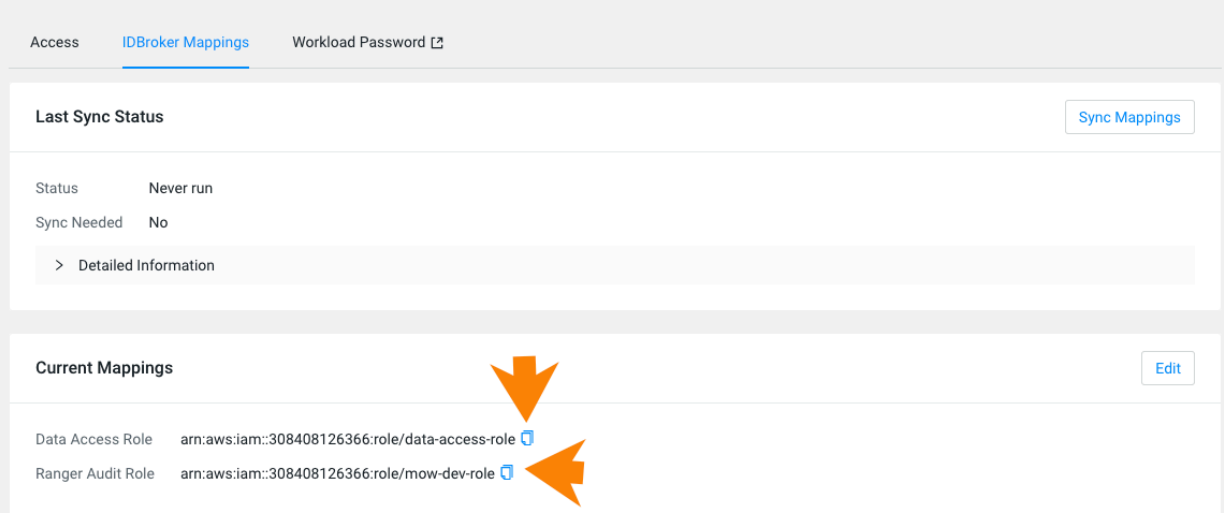

Further permissions reduction (Optional)

You can reduce these policies even further by pre-creating some resources:

- If you set up your own VPC and subnets, then the following actions can be removed from the policy:

```
ec2:CreateVpc
ec2:CreateNatGateway
ec2:CreateRouteTable
ec2:CreateSubnet
ec2:CreateVpcEndpoint
ec2:CreateInternetGateway
ec2:DeleteSubnet
ec2:DeleteInternetGateway
ec2:AttachInternetGateway
ec2:DetachInternetGateway
ec2:DescribePrefixLists
ec2:AllocateAddress
ec2:AssociateRouteTable
ec2:CreateRoute
ec2:DeleteRouteTable
ec2:DeleteVpcEndpoints
ec2:DisassociateRouteTable
ec2:ReleaseAddress
```

```
ec2:DeleteRoute
ec2:DeleteNatGateway
ec2:DeleteVpc
```

- If you use your own security groups, then the following actions can be removed from the policy:

```
ec2:CreateSecurityGroup
ec2:AuthorizeSecurityGroupIngress
ec2:RevokeSecurityGroupEgress
ec2:AuthorizeSecurityGroupEgress
```

- If you use private IPs or set up Cluster Connectivity Manager to communicate with the Cloudera Control Plane, then the following actions can be removed from the policy:

```
ec2:CreateSecurityGroup
ec2:AuthorizeSecurityGroupIngress
ec2:RevokeSecurityGroupEgress
ec2:AuthorizeSecurityGroupEgress
```

- If you use private IPs or set up Cluster Connectivity Manager to communicate with the Cloudera Control Plane, then the following actions can be removed from the policy:

```
ec2:AllocateAddress
ec2:ReleaseAddress
```

S3 bucket, and IAM roles and policies for logs, backups, and data storage

Cloudera requires that you create and provide at least one S3 bucket for storing workload data and logs. You also need to create and provide multiple IAM roles and policies that allow access to the S3 bucket.

Note:

The Cloudera Data Warehouse service creates its own S3 buckets and IAM roles and policies. It does not use the environment's S3 buckets or IAM roles and policies.

The S3 bucket is used for:

- Storage location base - Workload data storage and Ranger audits
- Logs location base - Service logs, FreeIPA logs
- Backup location base - FreeIPA and Data Lake backups

The S3 bucket must be in the same region as the environment.

For detailed information about the required setup, review the following documentation:

Minimal setup for AWS cloud storage

This minimal secure setup uses one S3 bucket for each Data Lake, and multiple IAM roles and policies.

The example setup includes:

- One S3 bucket with a sub-directory named after your data lake such as s3a://my-bucket/my-dl.
- Four IAM roles and two instance profiles:
  - IDBROKER_ROLE role and instance profile
  - LOG_ROLE role and instance profile
  - RANGER_AUDIT_ROLE
  - DATALAKE_ADMIN_ROLE

- Nine IAM policies:
  - One AssumeRole ipolicy (idbroker-assume-role) that can be used by the IDBroker component of the data lake cluster to assume each of the following roles: RANGER_AUDIT_ROLE and DATALAKE_ADMIN_ROLE.
  - A shared policy for accessing S3:
    - aws-cdp-bucket-access-policy
  - Five (or six) policies for specific bucket directory access:
    - Log storage (aws-cdp-log-policy)
    - Ranger audit (aws-cdp-ranger-audit-s3-policy)
    - Data Lake admin and RAZ (aws-cdp-datalake-admin-s3-policy)
    - Data Lake backups (aws-datalake-backup-policy)
    - Data Lake restore (of backups) (aws-datalake-restore-policy)
    - (Optional) FreeIPA backup storage - if using a separate bucket (aws-cdp-backup-policy)
  - Two trust policies:
    - aws-cdp-ec2-role-trust-policy
    - aws-cdp-idbroker-role-trust-policy

Note:

S3 Express One Zone buckets should not be used for the S3 cloud storage setup. S3 Express buckets are only supported for use with Cloudera Data Hub workloads as additional temporary data buckets.

Required cloud storage

One S3 bucket with a sub-directory named after your data lake such as s3a://my-bucket/my-dl is required for this setup.

Note:

You can use separate locations for your data, logs, and backups. The Storage Location Base, Logs Location Base, and Backup Location Base, happen to be in the same bucket in this example, but this is not required. If using multiple buckets, the aws-cdp-bucket-access-policy needs to have the additional buckets specified.

Important:

Ranger audits are stored under Storage Location Base and not under Logs Location Base. The Logs Location Base is used for Data Lake, Cloudera Data Hub and FreeIPA logs, and (if no separate location is provided) FreeIPA and Data Lake backups.

During Data Lake creation, Cloudera automatically creates a location for Ranger audits, Data Lake, Cloudera Data Hub and FreeIPA logs, and FreeIPA and Data Lake backups. The location for each of these depends on the supplied storage location base and logs location base. The directory structure is created automatically by Cloudera within these base directories:

Storage Location Base examples

	s3a://my-bucket/	s3a://my-bucket/my-dl
Ranger Audit Logs	s3a://my-bucket/ranger/audit	s3a://my-bucket/my-dl/ranger/audit

Logs Location Base examples

	s3a://my-bucket/	s3a://my-bucket/my-dl
FreeIPA Logs	s3a://my-bucket/cluster-logs/freeipa	s3a://my-bucket/my-dl/cluster-logs/freeipa If your environment was created prior to February 2021, this is s3a://my-bucket/my-dl/freeipa

Backup Location Base examples

	s3a://my-bucket/	s3a://my-bucket/my-dl
FreeIPA Backup	s3a://my-bucket/cluster-backups/freeipa	s3a://my-bucket/my-dl/cluster-backups/freeipa

Required IAM resources

The following table lists the IAM roles and IAM policies that need to be created on AWS, and describes which policies should be assigned to which roles (as presented in the diagram, in some cases policies should be assigned to multiple roles). The policy definitions are provided in IAM policy definitions:

Role	Permissions policy	Trust policy	Description
IDBROKER_ROLE	aws-cdp-idbroker-assume-role-policy aws-cdp-log-policy	aws-cdp-ec2-role-trust-policy	The assume role permissions policy must, at a minimum, allow the IDBROKER_ROLE to assume the RANGER_AUDIT_ROLE and the DATALAKE_ADMIN_ROLE. This policy must also allow the IDBROKER_ROLE to assume any other role for which a user or group mapping exists in the IDBroker. Furthermore, the IDBROKER_ROLE needs the same permissions policy as the LOG_ROLE so that it can access the Logs Location Base. The trust policy allows the role to be assumed by the IDBroker EC2 instance.
LOG_ROLE	aws-cdp-log-policy aws-datalake-restore-policy aws-cdp-backup-policy (Optional)	aws-cdp-ec2-role-trust-policy	This role uses the aws-cdp-log-policy permissions policy to provide Cloudera with access to the specific location called Logs Location Base for logs. If your Backup Location Base is in a separate bucket or folder, you also need to provide the aws-cdp-backup-policy. The trust policy allows the role to be assumed by EC2 instances in the cluster. When using lower versions than Cloudera Public Cloud 7.2.17, the aws-datalake-restore-policy is required for both upgrading a Data Lake and restoring a Data Lake backup. When using Cloudera Public Cloud 7.2.17 or higher versions, the aws-datalake-restore-policy is not needed for the LOG_ROLE.

Role	Permissions policy	Trust policy	Description
RANGER_AUDIT_ROLE	aws-cdp-ranger-audit-s3-policy aws-cdp-bucket-access-policy aws-datalake-backup-policy aws-datalake-restore-policy	aws-cdp-idbroker-role-trust-policy	This role uses the four permissions policies to provide write access to the Ranger audit sub-directory that Cloudera creates within the Storage Location Base. The trust policy allows the role to be assumed by IDBroker. The aws-datalake-backup-policy is required for both upgrading a Data Lake and backing up a Data Lake. The aws-datalake-restore-policy is required for both upgrading a Data Lake and restoring a Data Lake backup.
DATALAKE_ADMIN_ROLE	aws-cdp-datalake-admin-s3-policy aws-cdp-bucket-access-policy aws-datalake-backup-policy aws-datalake-restore-policy	aws-cdp-idbroker-role-trust-policy	This role uses the four permissions policies to provide the Data Lake admin with full access to the whole Storage Location Base. Additionally, if you choose to configure Fine-grained access control, RAZ uses this role to orchestrate the access control. The trust policy allows the role to be assumed by IDBroker. The aws-datalake-backup-policy is required for both upgrading a Data Lake and backing up a Data Lake. The aws-datalake-restore-policy is required for both upgrading a Data Lake and restoring a Data Lake backup.

The following diagram summarizes the roles, policies, and S3 bucket directories in this example setup:

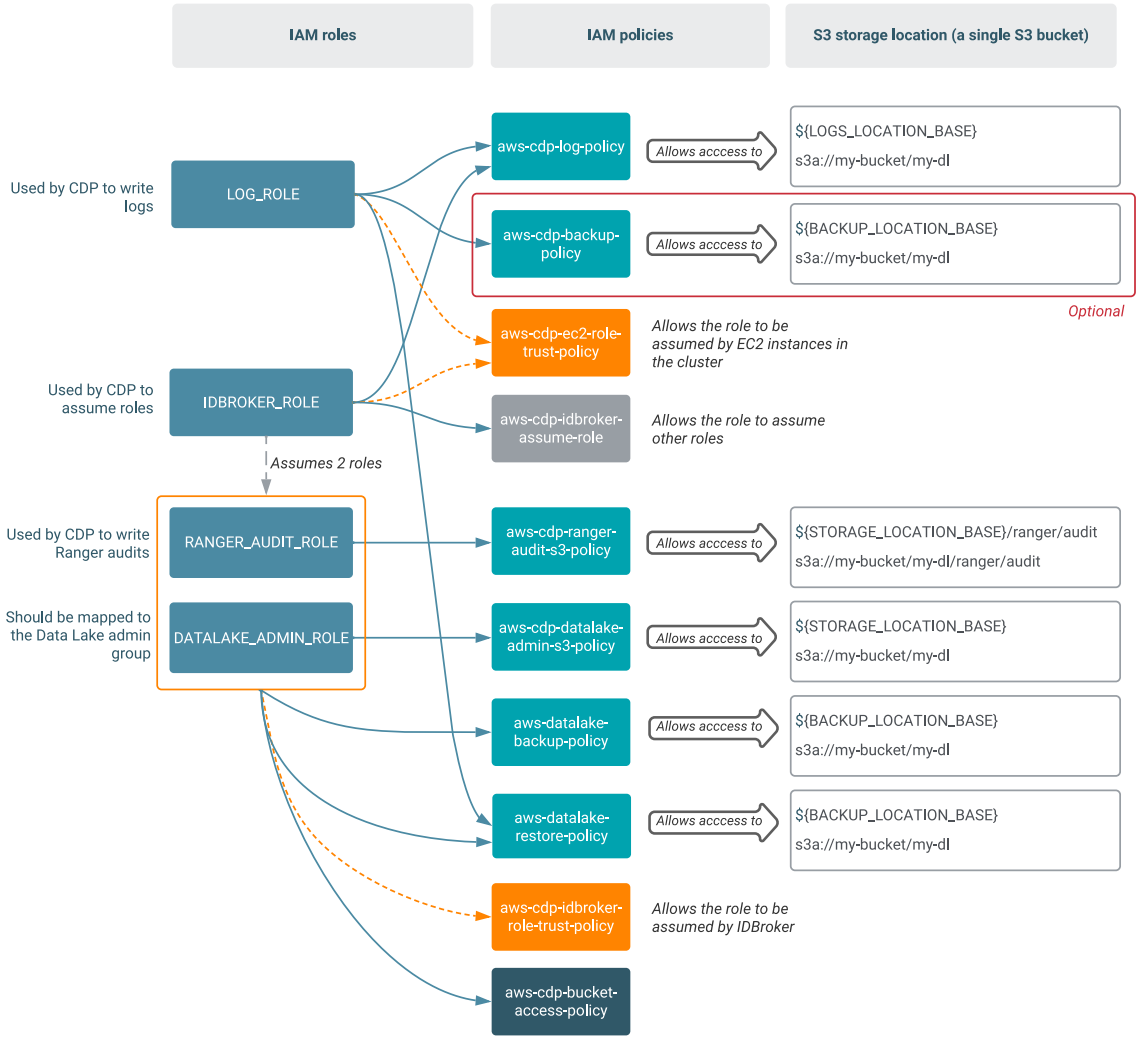

Note: You may choose a different setup, for example one using multiple buckets. Similarly, the IAM role and policy setup and names are just examples and you may choose a different setup. It is possible to have a setup with fewer roles and policies with broader access rights; however, such setup may not be secure for a production environment.

Use the following documentation in order to create the required S3 bucket and IAM resources, and provide them in Cloudera. The steps involve:

1. Creating an S3 bucket if you don't have one already
2. Creating permissions policies for the minimal setup
3. Creating the required IAM roles and their associated trust policies
4. Providing the parameters in the Cloudera UI

Create an S3 bucket

You need to provide an S3 location that Cloudera can use for logs and data storage. If you don't already have an S3 bucket, you can create one from the AWS S3 console or AWS CLI.

Steps

For AWS console

1. In the AWS web interface, navigate to the S3 console.
2. Click on Create bucket.

3. Enter bucket name.
4. Select the region where the bucket should be created.
5. Edit additional settings if needed. For example:
   - Enable encryption
   - Add tags if required by your organization.
6. Click on Create bucket.

For AWS CLI

Use the following command to create an S3 bucket:

```
aws s3api create-bucket --bucket <BUCKET-NAME>
```

See AWS documentation linked below for more information.

Related Information

Creating a bucket

AWS CLI Reference: create-bucket

Create permissions policies for the minimal setup

You need to create permissions policies that can be assigned to specific roles required by Cloudera. You can create these IAM permissions policies from the AWS IAM console or AWS CLI.

Before you begin

Obtain the relevant IAM policy definitions from IAM policy definitions. You need to replace all the placeholders in the policy with actual values.

Note: You only need to create permissions policies. Do not create trust policies in this manner. Trust policies are not created and attached to roles in the same manner as regular IAM policies. Establishing a trust relationship via trust policies is covered as part of a later step described below.

Steps

For AWS console

1. In the AWS web interface, navigate to the AIM console.
2. From the navigation pane, select Policies and then click on Create Policy
3. Choose the JSON tab.
4. Paste the appropriate JSON policy document.

   Note: Make sure to replace all the placeholders in the policy with actual values.

5. Click Next.
6. Add tags if required by your organization.
7. Click Next.
8. Type a Name and an optional Description.
9. Click Create policy.

For AWS CLI

Use the following command to create permissions policies:

```
aws iam create-policy --policy-name <NAME> --policy-document <JSON-DOCUM
ENT>
```

Repeat these steps for all permissions policies that need to be created. For more detailed instructions on how to create IAM policies, refer to the AWS documentation linked below.

Related Information

Creating IAM policies (console)

AWS CLI Reference: create-policy

Create the IAM roles and establish trust relationships: IDBROKER_ROLE and LOG_ROLE

You can create the IDBROKER_ROLE and LOG_ROLE IAM roles from the IAM console or AWS CLI.

Before you begin

Review the minimal setup for cloud storage outlined above and make sure that you understand which policies you need to attach to which roles.

Steps

For AWS console

1. In the AWS web interface, navigate to the IAM console.
2. From the navigation pane, select Roles.
3. Click on Create Role.
4. Under "Select type of trusted entity" select AWS service.
5. Under "Choose a use case" select EC2.

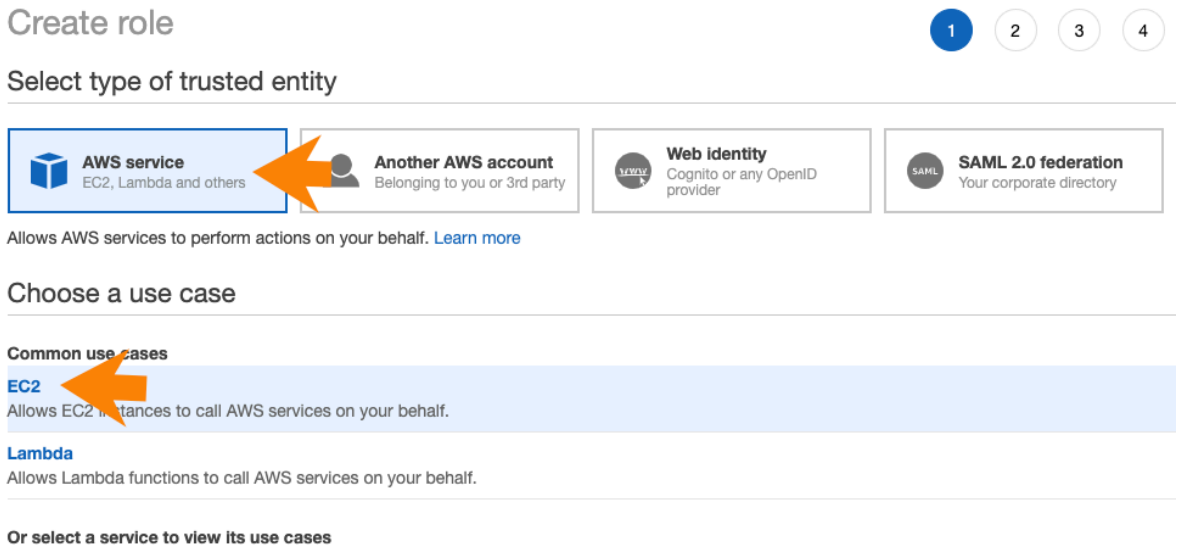

6. Click Next.
7. Attach the permissions policies required for the role that you are creating. Make sure to attach all required policies.
8. Click Next.
9. Add tags if required by your organization.
10. Click Next.
11. Provide a name and an optional description for your role, and verify that you have attached all required permissions policies.
12. Click on Create role. This creates an IAM role and instance profile.
13. Click on the role to see its details.

14. Verify that the "Instance Profile ARN" is present:

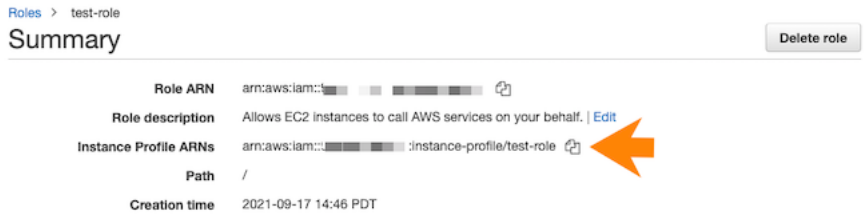

15. Click on the Trust relationships tab.

16. Click on Edit trust relationship.

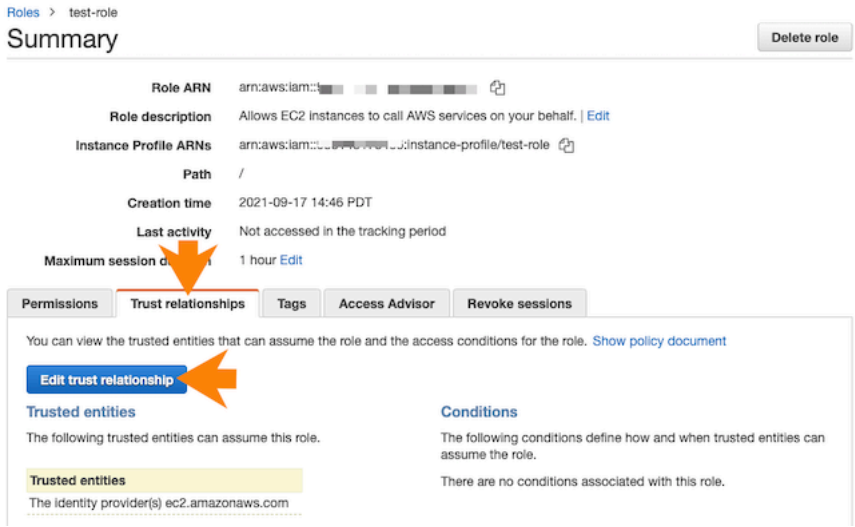

17. Verify that the JSON declaring a trust policy is the same as described in the minimal setup. If it isn't, replace it with the correct trust policy definition.

18. Click on Update trust policy.

Repeat these steps to create both roles.

For AWS CLI

Use these commands to create the required IAM roles and instance profiles via AWS CLI:

```
aws iam create-role --role name <ROLE-NAME> --assume-role-policy-document
 <TRUST-POLICY>
     aws iam attach-role-policy --role name <ROLE-NAME> --policy-arn <P
ERMISSIONS-POLICY-ARN>
     aws iam create-instance-profile --instance-profile-name <IP-NAME>
     aws iam add-role-to-instance-profile --instance-profile-name <IP-NAM
E> --role-name <ROLE-NAME>
```

You should assign the appropriate policies described in the minimal setup. Note that the aws iam attach-role-poli cy command needs to be executed separately for each policy that needs to be attached.

Related Information

AWS CLI Reference: iam

Create the IAM roles and establish trust relationships: RANGER_AUDIT_ROLE and DATALAKE_ADMIN_ROLE

Use the following steps to create the RANGER_AUDIT_ROLE and DATALAKE_ADMIN_ROLE.

Before you begin

Review the minimal setup for cloud storage outlined above and make sure that you understand which policies you need to attach to which roles.

Steps

For AWS console

1. In the AWS web interface, navigate to the IAM console.
2. From the navigation pane, select Roles.
3. Click on Create Role.
4. Under "Select type of trusted entity" select Another AWS account.
5. Under "Account ID", enter your AWS account ID. You can obtain it by clicking on your user name in the top bar.

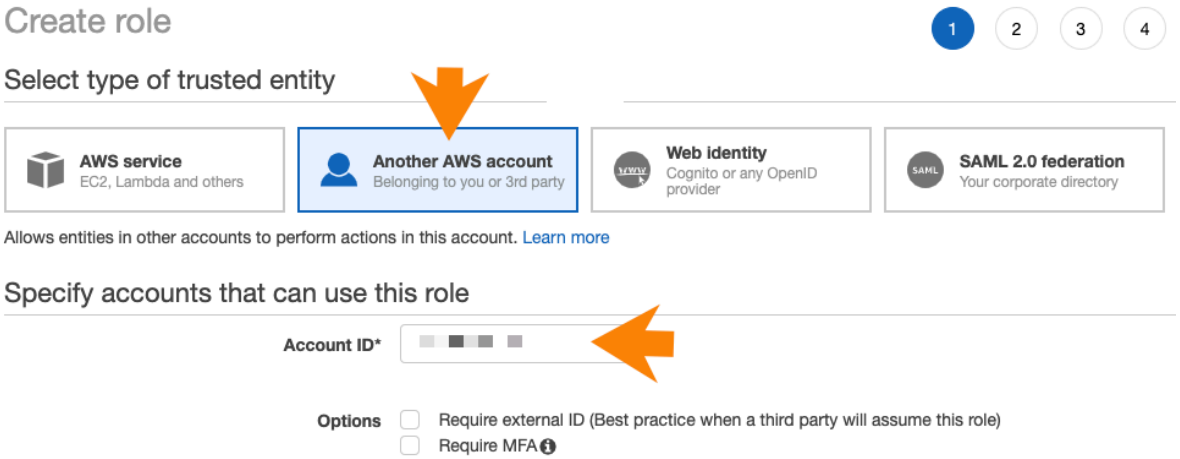

6. Click Next.
7. Attach the permissions policies required for the role that you are creating. Make sure to attach all required policies described in the minimal setup
8. Click Next.
9. Add tags if required by your organization.
10. Click Next.
11. Provide a name and an optional description for your role, and verify that you have attached all required permissions policies.
12. Click on Create role. This creates an IAM role. No instance profile is created in this case.
13. Click on the role to see its details.
14. Click on the Trust relationships tab.

15. Click on Edit trust relationship.

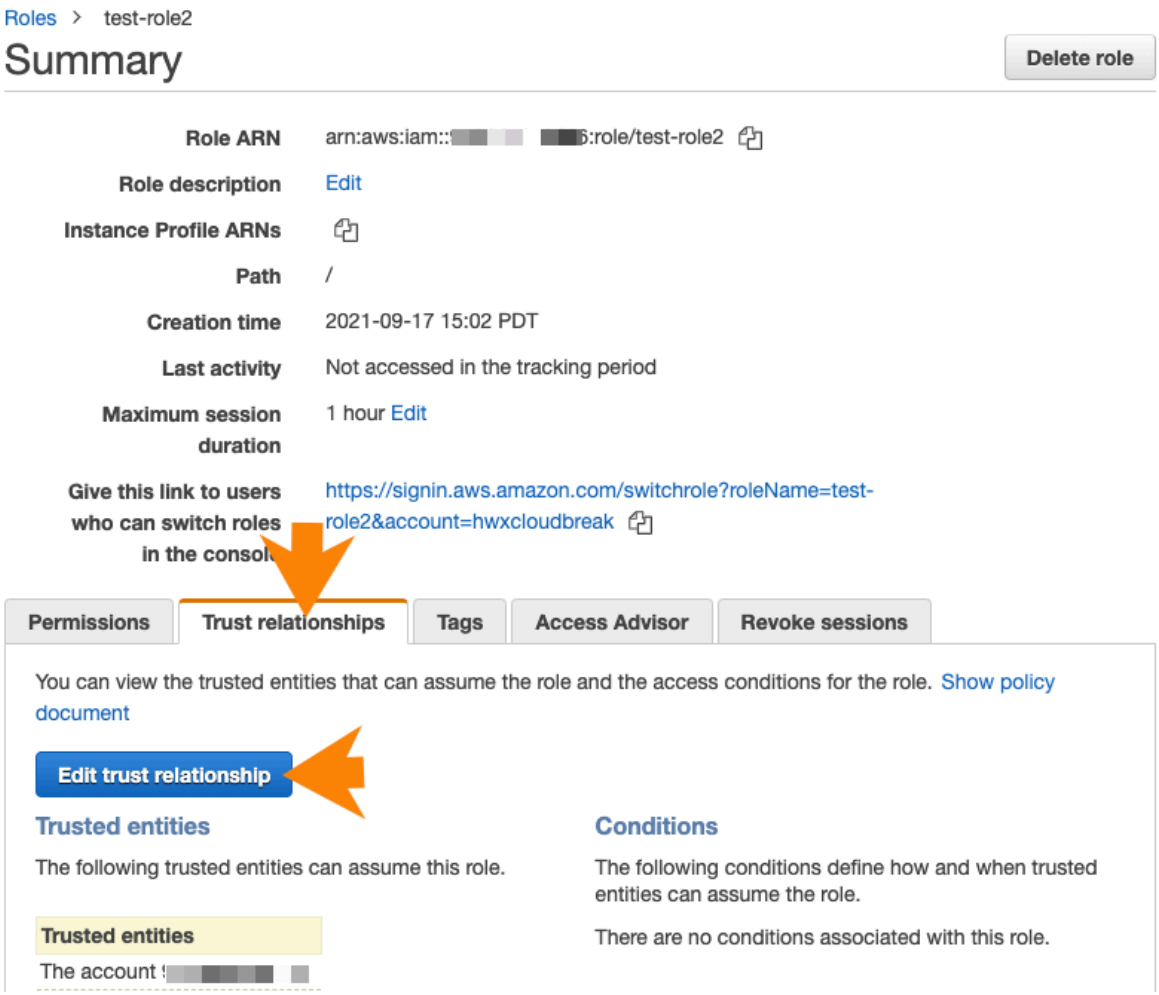

16. Paste the JSON declaring a trust policy in place of the default policy. Use the appropriate policy described in the minimal setup.

17. Click on Update trust policy.

Repeat these steps for both roles.

For AWS CLI

Use these commands to create the required IAM roles via AWS CLI:

```
aws iam create-role --role name <ROLE-NAME> --assume-role-policy-document
 <TRUST-POLICY>
      aws iam attach-role-policy --role name <ROLE-NAME> --policy-arn <
PERMISSIONS-POLICY-ARN>
```

Note that the aws iam attach-role-policy command needs to be repeated if there are multiple policies to attach. You should assign the appropriate policies described in the minimal setup.

Related Information

AWS CLI Reference: iam

Providing the parameters in the Cloudera UI

Once you’ve created the S3 bucket and the required IAM policies, roles, and instance profiles, provide the information related to these resources in the Register Environment wizard.

Parameter	Description	Example
Data Access and Audit		
Assumer Instance Profile	Select the IDBROKER_ROLE instance profile created earlier during IDBROKER_ROLE role creation.	IDBROKER_ROLE
Storage Location Base	Enter the Storage Location Base S3 bucket location created earlier.	my-bucket/my-dl
Data Access Role	Select the DATALAKE_ADMIN_ROLE created earlier.	DATALAKE_ADMIN_ROLE
Ranger Audit Role	Select the RANGER_AUDIT_ROLE created earlier.	RANGER_AUDIT_ROLE
Fine-grained access control on ADLS Gen2		
Enable Ranger authorization for ADLS Gen2	If you would like to use Fine-grained access control, click on "Enable Ranger authorization for ADLS Gen2". Next, select the DATALAKE_ADMIN_ROLE created earlier.	DATALAKE_ADMIN_ROLE
Select Azure managed identity for Ranger authorization		
Logs		
Logger Instance Profile	Select the LOG_ROLE instance profile created earlier during LOG_ROLE role creation.	LOG_ROLE
Logs Location Base	Enter the Logs Location Base S3 bucket location created earlier.	my-bucket-/my-dl
Backup Location Base (Optional)	If you created it, enter the Backup Location Base S3 bucket location. This is optional. If you don't provide this, FreeIPA and Data Lake backups will be stored in the Logs Location Base.	my-bucket-/my-dl

Onboarding Cloudera users and groups for AWS cloud storage (RAZ environments)

If your AWS environment has Fine-grained access control enabled, you should onboard your users using Ranger instead of using IDBroker.

For more information, refer to Ranger policy options for RAZ-enabled AWS environment and Using Ranger to Provide Authorization in ClouderaUsing Ranger to Provide Authorization in Cloudera.

Onboarding Cloudera users and groups for AWS cloud storage (no RAZ)

The minimal setup defined earlier spins up a Cloudera environment and Data Lake with no end user access to cloud storage. Adding users and groups to a Cloudera environment involves ensuring they are properly mapped to IAM roles to access cloud storage.

Note: If you are using Fine-grained access control, you should onboard your users using Ranger instead of using IDBroker mappings. Adding IDBroker mappings is disabled for RAZ-enabled environments.

In general, to onboard a new user or group to be onboarded you should have the following IAM roles and policies pre-created in AWS:

- One IAM role for the user/group
- One IAM policy for the user/group role to access the required S3 bucket(s) and path(s)

In the example below, we are adding a data engineering group and a data science group to the Cloudera environment. The final goal is to have the following that builds on the minimal setup:

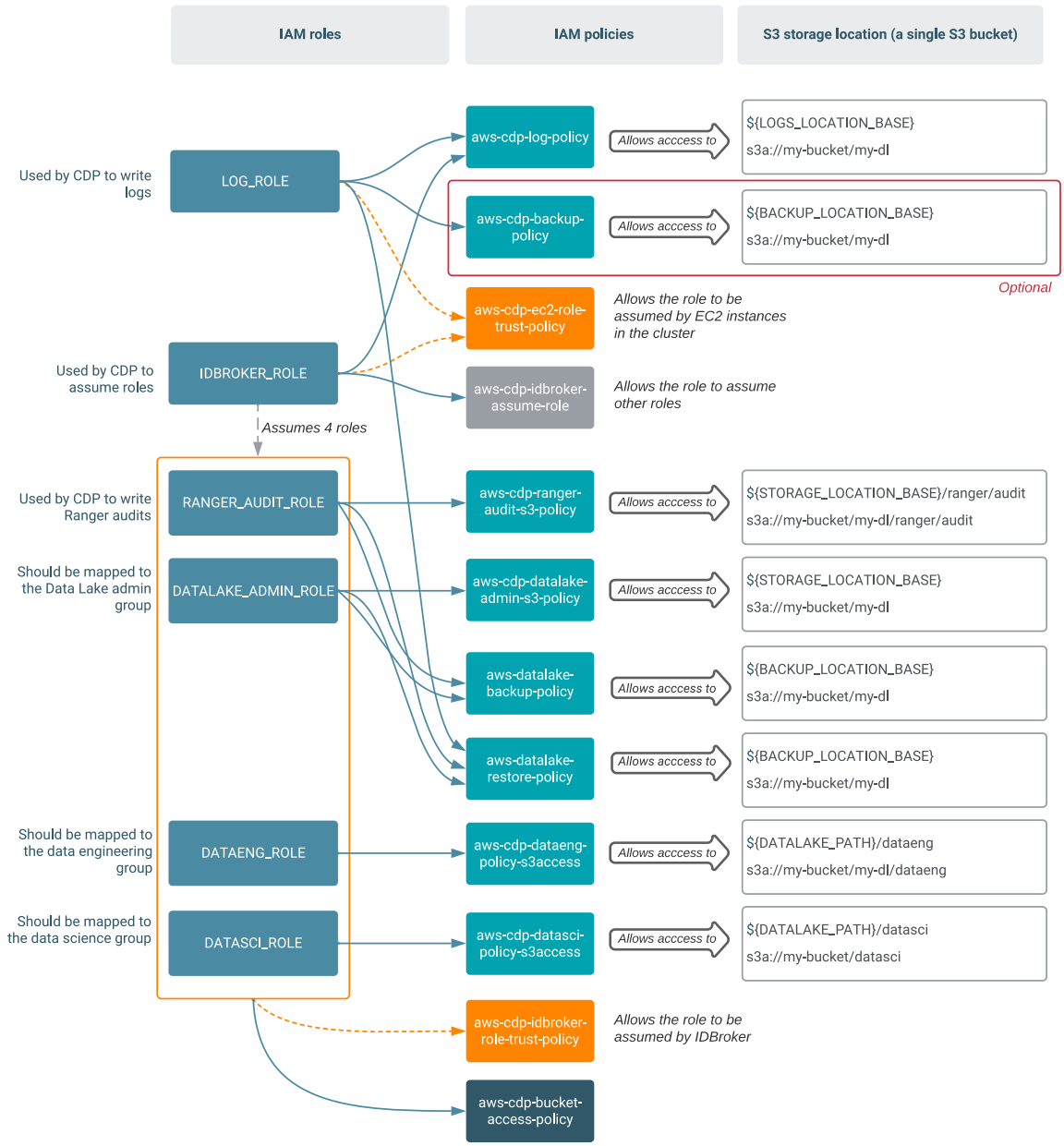

In the table below, bolded names are new compared to the minimal setup. The definitions for the policies mentioned in the table can be found in IAM policy definitions.

Role	Permissions policy	Trust policy	Description
DATAENG_ROLE	aws-cdp-dataeng-policy-s3access aws-cdp-bucket-access-policy	aws-cdp-idbroker-role-trust-policy	This role uses the three permissions policies to provide data engineers with access to a specific S3 location (s3://my-bucket/my-dl/dataeng). The trust policy allows the role to be assumed by IDBroker.

Role	Permissions policy	Trust policy	Description
DATASCI_ROLE	aws-cdp-datasci-policy-s3access aws-cdp-bucket-access-policy	aws-cdp-idbroker-role-trust-policy	This role uses the three permissions policies to provide data scientists with access to a specific S3 location (s3://my-bucket/my-dl/datasci). The trust policy allows the role to be assumed by IDBroker.

Use the following documentation in order to create the required S3 bucket and IAM resources, and provide them in Cloudera. The steps involve:

1. Creating the additional permissions policies for the minimal setup
2. Creating the required IAM roles and their associated trust policies
3. Creating the mappings in Cloudera

Create permissions policies for user access

You can create the IAM permissions policies required for user access to the data storage S3 location from the IAM console or AWS CLI.

Before you begin

For IAM policy definitions, refer to IAM policy definitions. Make sure to replace all the placeholders in the policy with actual values.

Note: You only need to create permissions policies. Do not create trust policies in this manner. Trust policies are not created and attached to roles in the same manner as regular IAM policies. Establishing a trust relationship via trust policies is covered as part of a later step described below.

Steps

For AWS console

1. In the AWS web interface, navigate to the IAM console.
2. From the navigation pane, select Policies and then click on Create Policy
3. Choose the JSON tab.
4. Paste the appropriate JSON policy document.

   Note: Make sure to replace all the placeholders in the policy with actual values.

5. Click Next.
6. Add tags if required by your organization.
7. Click Next.
8. Type a Name and an optional Description.
9. Click Create policy.

For AWS CLI

Use the following command to create permissions policies:

```
aws iam create-policy --policy-name <NAME> --policy-document <JSON-DOCUM
ENT>
```

Repeat these steps for all permissions policies that need to be created. For more detailed instructions on how to create IAM policies, refer to the AWS documentation linked below.

Related Information

Creating IAM policies (console)

AWS CLI Reference: create-policy

Create the IAM roles and establish trust relationships: DATAENG_ROLE and DATASCI_ROLE

Use the following steps to create the DATAENG_ROLE and DATASCI_ROLE. These roles allow end users to access specific subsets of S3 cloud storage that you configure in Cloudera.

Before you begin

Review the setup described above and make sure that you understand which policies you need to attach to which roles.

Steps

For AWS console

1. In the AWS web interface, navigate to the IAM console.
2. From the navigation pane, select Roles.
3. Click on Create Role.
4. Under "Select type of trusted entity" select Another AWS account.
5. Under "Account ID", enter your AWS account ID. You can obtain it by clicking on your user name in the top bar.

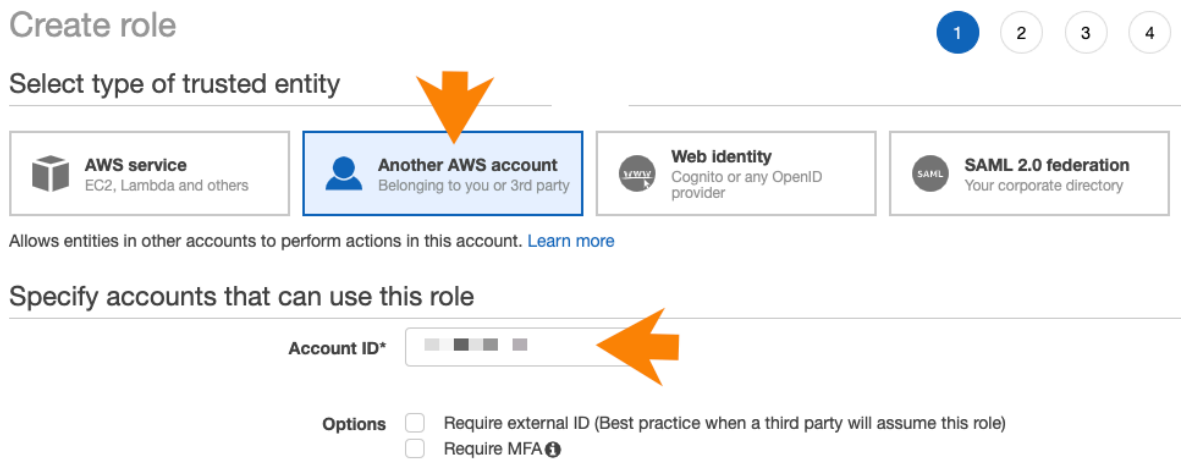

6. Click Next.
7. Attach the permissions policies required for the role that you are creating. Make sure to attach all required policies.
8. Click Next.
9. Add tags if required by your organization.
10. Click Next.
11. Provide a name and an optional description for your role, and verify that you have attached all required permissions policies.
12. Click on Create role. This creates an IAM role. No instance profile is created in this case.
13. Click on the role to see its details.
14. Click on the Trust relationships tab.

15. Click on Edit trust relationship.

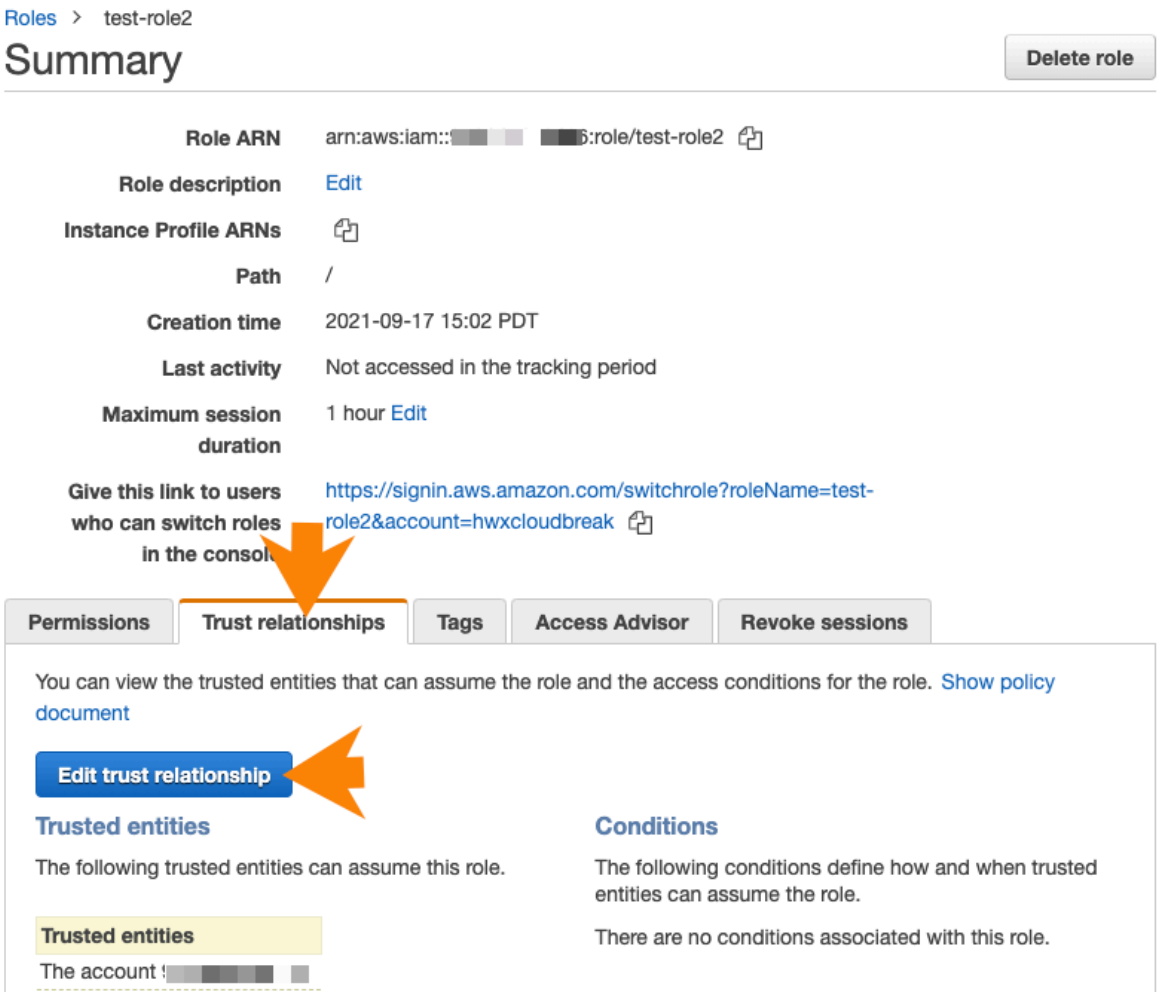

16. Paste the JSON declaring a trust policy in place of the default policy. Use the appropriate policy described above.

17. Click on Update trust policy.

Repeat these steps for both roles.

For AWS CLI

Use these commands to create the required IAM roles via AWS CLI:

```
aws iam create-role --role name <ROLE-NAME> --assume-role-policy-document
 <TRUST-POLICY>
aws iam attach-role-policy --role name <ROLE-NAME> --policy-arn <PERMIS
SIONS-POLICY-ARN>
```

Note that the aws iam attach-role-policy command needs to be repeated if there are multiple policies to attach. You should assign the appropriate policies described above.

Once these resources have been created, you can provide them be editing an existing environment in Cloudera.

Related Information

AWS CLI Reference: iam

Adding Cloudera user/group to IAM role mappings

After creating the two additional IAM roles, one for data engineers and one data scientists, map them to specific user/group in Cloudera.

Note: If you are using Fine-grained access control, you should onboard your users using Ranger instead of using IDBroker mappings. Adding IDBroker mappings is disabled for RAZ-enabled environments.

Note:

If a user is mapped to multiple roles via group membership, the specific role to be used needs to be provided at runtime. If the user is mapped directly to a role, the direct mapping takes precedence over mapping via group membership. For information on how to specify the role, refer to Specifying a group when user belongs to multiple groups.

Required role: DataSteward, EnvironmentAdmin, or Owner

Steps

For CDP UI

1. The option to add/modify these mappings is available from the Cloudera Management Console under Environments > click on an environment > Actions > Manage Access > IDBroker Mappings > Edit.
2. Under Current Mappings, click Edit.
3. Click + to display a new field for adding a mapping.
4. Provide the following:
   a. The User or Group dropdown is pre-populated with Cloudera users and groups. Select the user or group that you would like to map.
   b. Under Role, specify the role ARN (copied from the IAM role page on AWS). You should select your DATAENG_ROLE here.
5. Repeat the previous two steps to add additional mapping for the DATASCI_ROLE.
6. For example, in the example setup we created the following roles:
   - DATAENG_ROLE - We created this role while onboarding users and we assume that there is a DataEngineers group that was created in Cloudera.
   - DATASCI_ROLE - We created this role while onboarding users and we assume that there is a DataScientists group that was created in Cloudera.

   Based on the roles and groups created in this example, the mappings that need to be created are:

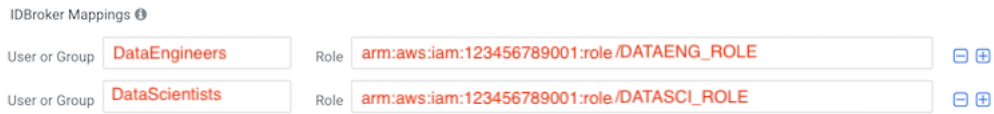

7. Click Save and Sync.

For CDP CLI

If you would like to create the mappings via CDP CLI, you can:

1. Use the cdp environments get-id-broker-mappings command to obtain your current mappings.
2. Use the cdp environments set-id-broker-mappings command to set additional mappings. The only way to use this command is to:
   - Pass all the current mappings
   - Add the new mappings
3. Next, sync IDBroker mappings. For example:

```
cdp environments sync-id-broker-mappings --environment-name demo3
```

4. Finally, check the sync status. For example:

```
cdp environments get-id-broker-mappings-sync-status --environment-name d
emo3
```

IAM policy definitions for the minimal cloud storage setup

Use the following IAM policy definitions when defining IAM policies for the minimal cloud storage setup described in the parent topic.

Note that:

- The policy definitions refer to roles by using the convention presented in the table in the parent topic. If the IAM roles that you created use different names, you should update these names in the policy definitions below.
- The policy definitions refer to the example S3 subdirectories presented in the parent topic. If the S3 bucket sub-directories that you created use different names, you should update these names in the policy definitions below.

While creating these IAM policies, make sure to replace the following with actual values:

- ${ARN_PARTITION} - Replace this with "aws". Or, if you are using one of China regions, replace it with "aws-cn". See Amazon Resource Names (ARNs).
- ${AWS_ACCOUNT_ID} - Replace this with your our AWS account ID.
- ${DATALAKE_BUCKET} - Replace this with the name of your S3 bucket. For example my-bucket.
- ${STORAGE_LOCATION_BASE} - Replace this with the path to your Data Lake directory in the S3 bucket specified as ${DATALAKE_BUCKET}/SOME_PATH. For example my-bucket/my-dl.
- ${LOGS_BUCKET} - Replace this with the name of your S3 bucket for logs. For example my-bucket.
- ${LOGS_LOCATION_BASE} - Replace this with the path to your S3 location for logs. For example my-bucket/my-dl.
- ${BACKUP_BUCKET} - Replace this with the name of your S3 bucket for backups. For example my-bucket.
- ${BACKUP_LOCATION_BASE} - Replace this with the path to your S3 location for backups. This location is used for both FreeIPA and Data Lake backups. For example my-bucket/my-dl.

aws-cdp-idbroker-assume-role-policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Note:

The * in the aws-cdp-idbroker-assume-role-policy means that the IDBroker role will be able to assume any role that also has a trust with the IDBroker role. If preferred, the resource can be a list of role ARNs, but this policy would need to be updated each time a new role is mapped. Both a trust from the target role to the IDBroker role and an AssumeRole policy are necessary for roles to be assumed.

aws-cdp-log-policy

Refer to aws-cdp-log-policy.json.

aws-cdp-backup-policy

Refer to aws-cdp-backup-policy.json.

Note: The aws-cdp-backup-policy policy is only required if your Backup Location Base is in a separate bucket.

aws-cdp-ranger-audit-s3-policy

Refer to aws-cdp-ranger-audit-s3-policy.json.

aws-cdp-datalake-admin-s3-policy

Refer to aws-cdp-datalake-admin-s3-policy.json.

aws-cdp-bucket-access-policy

Refer to aws-cdp-bucket-access-policy.json.

Note:

If using multiple buckets for Storage Location Base, Logs Location Base, and Backup Location Base, the aws-cdp-bucket-access-policy needs to have all the buckets specified.

Note:

The action "s3:ListAllMyBuckets" is optional and allows Hue to display a list of buckets in the S3 browser. Without "s3:ListAllMyBuckets", a user must explicitly enter which S3 bucket they want to browse in Hue.

aws-datalake-backup-policy

Refer to aws-datalake-backup-policy.json.

aws-datalake-restore-policy

Refer to aws-datalake-restore-policy.json.

aws-cdp-ec2-role-trust-policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

aws-cdp-idbroker-role-trust-policy

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
      {
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::${AWS_ACCOUNT_ID}:role/${IDBROKER_ROLE}"
          },
          "Action": "sts:AssumeRole"
          }
          ]
          }
```

IAM policy definitions for cloud storage location users

Use the following IAM policy definitions when defining IAM policies for cloud storage location users, such as data scientists and data engineers.

Note that:

- The policy definitions refer to roles by using the convention presented in the table in the parent topic. If the IAM roles that you created use different names, you should update these names in the policy definitions below.
- The policy definitions refer to the example S3 subdirectories presented in the parent topic. If the S3 bucket sub-directories that you created use different names, you should update these names in the policy definitions below.

While creating these IAM policies, make sure to replace the following with actual values:

- ${AWS_ACCOUNT_ID} - Your AWS account ID.
- ${DATALAKE_PATH} - Path to your Data Lake directory under the Storage Location Base. For example my-bucket/my-dl. This does not have to be under the Storage Location Base, but for simplicity this example assumes it is a subdirectory of the Storage Location Base.

aws-cdp-dataeng-policy-s3access

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::${DATALAKE_PATH}/dataeng/*"
        }
    ]
}
```

aws-cdp-datasci-policy-s3access

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::${DATALAKE_PATH}/datasci/*"
        }
    ]
}
```

Using S3 encryption

AWS S3 encryption requires additional permissions added to the IAM roles depending if read only or read/write access is needed.

All IAM roles which need to read data encrypted with SSE-KMS must have the permissions to decrypt using the specific key the data was encrypted with. All IAM roles which need to both read and write data need the encrypt and decrypt permissions (encrypt-only permission is not supported).

IAM role permissions for working with SSE-KMS

Depending on the type of access required, one of the two policies needs to be attached to the roles that access S3. From the examples above, these would be the following roles:

- LOG_ROLE
- RANGER_AUDIT_ROLE
- DATALAKE_ADMIN_ROLE
- DATAENG_ROLE
- DATASCI_ROLE

Use the json below to create the necessary policies and attach them to the correct IAM roles for your environment.

aws-cdp-sse-kms-read-only-policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": [
                "${KEY_ARN}"
            ]
        }
    ]
}
```

aws-cdp-sse-kms-read-write-policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "${KEY_ARN}"
            ]
        }
    ]
}
```

Related Information

Encrypting data on S3

Using S3 Express One Zone for data storage

You can use S3 Express One Zone (S3 Express) with Cloudera.

If you have additional data buckets that you would like to use with Cloudera and you do not need zone redundancy, you may use S3 Express buckets, for example for faster processing of temporary data.

The following limitations apply when using S3 Express buckets:

- You can only use S3 Express buckets with Cloudera Data Hub running Cloudera Runtime 7.2.18 or newer. Data services do not currently support it.
- S3 Express buckets may not be used for logs and backups.

If you would like to use an S3 Express bucket for running Cloudera Data Hub workloads, you should:

1. Add the required permissions in AWS
2. Set bucket region in core-site.xml

Permissions

In order to use an S3Express bucket, you should add the following permissions to the DATALAKE_ADMIN_ROLE role described in the Minimal setup for AWS cloud storage:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "VisualEditor2",
   "Effect": "Allow",
   "Action": "s3express:CreateSession",
   "Resource": "arn:aws:s3express:region:account-id:bucket/base-bucket-name
--azid--x-s3"
  }
 ]
}
```

Set bucket region in core-site.xml

Set the AWS region of the S3Epress bucket in the core-site.xml as follows:

```
<property>
  <name>fs.s3a.bucket.<bucket-name>.endpoint.region</name>
  <value>us-west-2</value>
</property>
```

The following screenshot illustrates how to perform this configuration in Cloudera Manager:

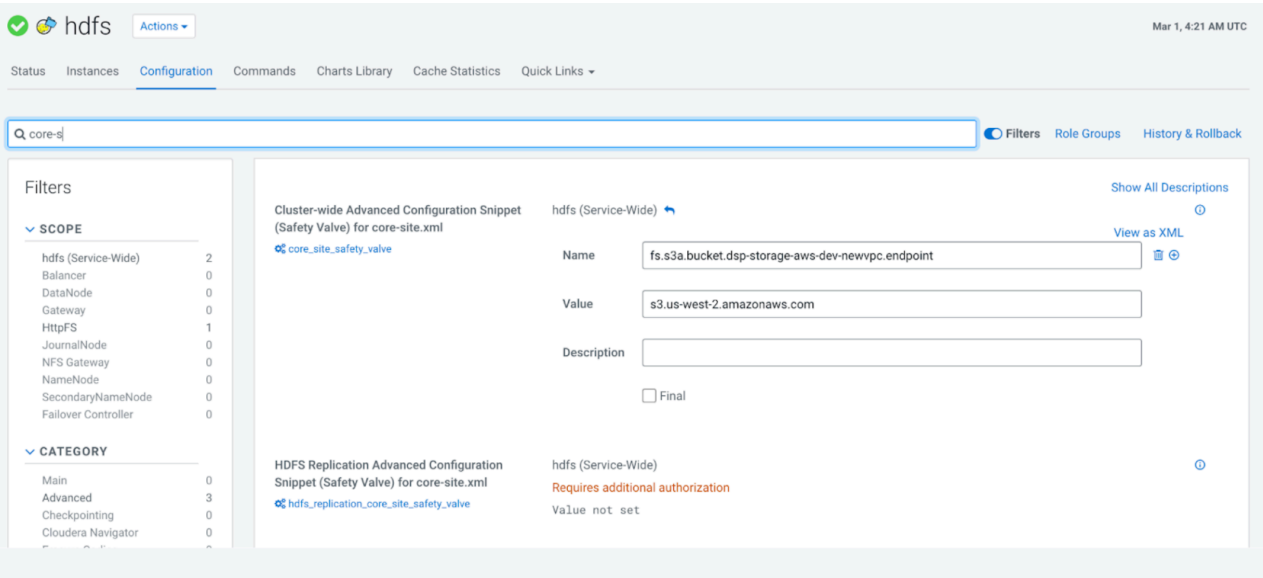

Supported AWS block storage

Review the list of supported AWS disk types for Cloudera.

Cloudera supports the following AWS disk types:

Cloudera name	AWS name
standard	Magnetic
ephemeral	Ephemeral
gp2	General Purpose (GP2 SSD)
gp3	General Purpose (GP3 SSD)
st1	Throughput Optimized HDD

Related Information

Amazon EBS volume types
Amazon EC2 instance store

Customer managed encryption keys

By default, Data Lake and FreeIPA's Amazon Elastic Block Store (EBS) volumes and Relational Database Service (RDS) are encrypted using a default key from Amazon's KMS, but you can optionally configure encryption using Customer Managed Keys (CMK). Data Hubs inherit environment's encryption key by default but you have an option to specify a different CMK during Cloudera Data Hub creation.

Amazon offers the option to encrypt EBS volumes and RDS instances using a default key from Amazon's Key Management System (KMS) or using an external customer-managed KMS. By default, Data Lake and FreeIPA are encrypted using the default key from Amazon's KMS present in the region where the environment is running, but you can provide a customer-managed KMS key instead of the default key. Encryption is configured for block devices and root devices. When encryption is configured for a given cluster, it is automatically applied to all the disk devices of any new VM instances added as a result of cluster scaling or repair.

This document includes the following content:

1. Environment and Cloudera Data Hub encryption options on page 44
2. Permissions for using encryption on page 45
3. Ensuring that an existing encryption key can be used on page 46
4. Creating a new encryption key on AWS on page 46

Environment and Cloudera Data Hub encryption options

By default, Cloudera Data Hub clusters use the same default key from Amazon's KMS or CMK as the parent environment but you have an option to pass a different CMK during Cloudera Data Hub creation.

All possible scenarios are summarized in the following table:

Encryption key during environment registration	Encryption key during Cloudera Data Hub creation	Result
Absent	Absent	• EBS and RDS encryption for Data Lake, FreeIPA, and Cloudera Data Hub clusters is with the default regional encryption key.
Present	Absent	• EBS and RDS encryption for Data Lake, FreeIPA, and all Cloudera Data Hub clusters is with the CMK provided during environment registration.

Encryption key during environment registration	Encryption key during Cloudera Data Hub creation	Result
Present	Present	• EBS and RDS encryption for Data Lake and FreeIPA is with the CMK provided during environment registration. • If a CMK is provided for a Cloudera Data Hub, then EBS encryption for the Cloudera Data Hub is with the CMK provided per host group during the Cloudera Data Hub creation.
Absent	Present	• No EBS and RDS encryption is configured for Data Lake and FreeIPA. • EBS encryption for the specific Cloudera Data Hub is with the default or CMK provided per host group during Cloudera Data Hub creation.

Permissions for using encryption

If you are planning to use encryption, ensure that the cross-account IAM role used for the provisioning credential includes the following permissions:

EC2 permissions

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "ec2:CopyImage",
      "ec2:CreateSnapshot",
      "ec2:DeleteSnapshot",
      "ec2:DescribeSnapshots",
      "ec2:CreateVolume",
      "ec2:DeleteVolume",
      "ec2:DescribeVolumes",
      "ec2:DeregisterImage",
    ],
    "Resource": "*"
  }
}
```

KMS permissions

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:ListKeys",
      "kms:ListAliases"
    ],
    "Resource": "*"
  }
}
```

If planning to use encryption, ensure that your existing encryption key can be used or create a new encryption key.

Ensuring that an existing encryption key can be used

If you already have an existing encryption key, make sure that the key fulfills the following requirements.

If you have an existing encryption key that you would like to use with Cloudera Data Hub, make sure that:

- The following are attached as key user:
  - The AWSServiceRoleForAutoScaling built-in role.
  - Your IAM role or IAM user used for the cloud credential.
- To check that these are attached, in the AWS Management Console, navigate to the KMS console > Customer managed keys, select your encryption key, and scroll to Key Users.
- The encryption key is located in the same region where you would like to create clusters with encrypted volumes.

Creating a new encryption key on AWS

If you don't have an existing encryption key, use the following instructions to create one.

1. In the AWS Management Console, navigate to KMS console.
2. Select Customer managed keys.
3. From the Region dropdown, select the region in which you would like to create and use the encryption key.
4. Click Create key.
5. In Step 1: Configure Key:
   a. Under Key type, choose Symmetric.
   b. Expand Advanced Options and under Key Material Origin, select “KMS” or “External”.
6. In Step 2: Create Alias and Description:
   a. Enter an alias for your key.
   b. Defining tags is optional.
7. In Step 3: Define Key Administrative Permissions, select the following:
   a. Choose your own IAM user/role used for logging into the AWS Management Console. Do not set AWSServiceRoleForAutoScaling or the cross-account IAM role as the key admin.
8. In Step 4: Define Key Usage Permissions:
   a. Select the AWSServiceRoleForAutoScaling built-in role.
   b. Select the cross-account IAM role.
9. In Step 5: Review and edit key policy, you may optionally tweak the key policy as desired, or simply leave it as generated by AWS.
10. Navigate to the last page of the wizard and then click Finish to create an encryption key.

Related Information

Adding a customer managed encryption key to a Cloudera environment running on AWS

Configuring encryption for Cloudera Data Hub's EBS volumes on AWS

AWS limits

When you create your AWS account, AWS sets limits to the resources available to you. The limits can vary by region.

For example:

- Depending on your AWS account limits, you might only be allowed to provision a certain number of CPU instances, or you might not have default access to GPU instances at all.
- If you are planning to use Endpoint Access Gateway, Cloudera will create two AWS network load balancers (AWS NLB) per cluster (that is for each Data Lake and Cloudera Data Hub). Make sure that your AWS NLB limits allow for the load balancer creation.
- If you are planning to use the Cloudera AI service, review Limitations on AWS in Cloudera AI docs.

Make sure to review your AWS service limits before your proceed.

To view the limits set by Amazon for your account, log in to AWS and go to EC2 > Limits. The EC2 Service Limits page lists the limits for the resources available to you in your EC2 instance, including limits to the number of instances and hosts. The Networking Limits section on the page lists the VPC, subnet, and security group limits for your AWS account. For example, there is typically a limit of 5 VPCs per region.

To review what resources Cloudera creates on your AWS account during environment registration and later when you deploy specific Cloudera services, refer to AWS resources used by Cloudera.

If you require more resources than the limit set by Amazon, you can request Amazon to raise the limit of a resource. On the EC2 Service Limits page, click Request limit increase for the resource that you want to increase and create an AWS support case for a Service Limit Increase.

For more information about AWS limits, refer to AWS Service Limits in AWS documentation.

Overview of AWS resources used by Cloudera

The following AWS resources are used by Cloudera and Cloudera services.

AWS resources created for a Cloudera environment

When a Cloudera environment is created, a FreeIPA cluster and a Data Lake cluster are created.

The following AWS resources are created for FreeIPA (one per environment):

Resource	Description
Virtual Private Cloud (VPC)	If during environment creation you select to have a new VPC and subnets created, then the new VPC and subnets are created on your AWS account. Alternatively, you can provide your own existing VPC and subnets. In both cases, all the resources that Cloudera provisions for the environment are provisioned into this specific VPC. For example, the EC2 instances provisioned for Cloudera Data Hub or Cloudera Data Warehouse are provisioned into that VPC.
Identity and Access Management (IAM)	The cross-account IAM policy that you provided as your credential allows Cloudera to obtain an access and secret key from AWS, allowing Cloudera to create resources for your environment and for Cloudera services such as Cloudera Data Hub, Cloudera Data Warehouse, and Cloudera AI on your AWS account.
CloudFormation	During environment creation, CloudFormation stack is provisioned for FreeIPA to create required resources. This generates an AWS stack which links and describes the resources of your FreeIPA server. Multi-AZ deployments do not use a CloudFormation template for VM creation. Neither autoscaling groups or launch templates are created. The cluster resources are managed individually using AWS native components (for example, EC2 instances). **Caution:** Do not attempt to change the configuration of the Auto Scaling groups or the launch template. Any manual modifications (such as modifying suspended processes or subnets) that you make to the Auto Scaling groups or the launch template are not supported and may cause future errors or failures, particularly during repair processes.
Auto Scaling	FreeIPA uses the Auto Scaling service for upscaling FreeIPA, except in a multi-AZ deployment. **Caution:** After FreeIPA instances have been created, do not attempt to change their configuration of the Auto Scaling groups or the launch template. Any manual modifications (such as modifying suspended processes or subnets) that you make to the Auto Scaling groups or the launch template are not supported and may cause future errors or failures, particularly during repair processes.

Resource	Description
Elastic Compute Cloud (EC2)	During environment creation, two or three m5.large EC2 instances are provisioned for the FreeIPA HA server by default. The number of instances depends on the selected Data Lake type. Furthermore, security groups with the rules specified during environment creation are provisioned to define inbound and outbound access to the instances.

In addition, the following resources are created for each Data Lake (one per environment):

Resource	Description
CloudFormation	A CloudFormation stack is provisioned for your Data Lake to create instances, disks, and RDS required. This generates an AWS stack which links and describes the resources of your Data Lake cluster. Multi-AZ deployments do not use a CloudFormation template for VM creation. Neither autoscaling groups nor launch templates are created. In a multi-AZ setup the cluster resources are managed individually using AWS native components (for example, EC2 instances). **Caution:** Do not attempt to change the configuration of the Auto Scaling groups or the launch template. Any manual modifications (such as modifying suspended processes or subnets) that you make to the Auto Scaling groups or the launch template are not supported and may cause future errors or failures, particularly during repair processes.
Auto Scaling	The Data Lake uses the Auto Scaling service for upscaling clusters, except in a multi-AZ deployment. **Caution:** After a Data Lake has been created, do not attempt to change the configuration of the Auto Scaling groups or the launch template. Any manual modifications (such as modifying suspended processes or subnets) that you make to the Auto Scaling groups or the launch template are not supported and may cause future errors or failures, particularly during repair processes.
Elastic Compute Cloud (EC2)	EC2 instances with attached storage are provisioned for the Data Lake nodes: • Light duty: Two instances are provisioned: One t3.medium instance (IDBroker) and one m5.2xlarge instance (Data Lake Master node). • Medium duty: Ten instances are provisioned: Two t3.medium instances (IDBroker), three m5.xlarge instances (two Data Lake Master nodes and one Auxiliary node), and five m5.2xlarge instances (three Data Lake Core nodes and two Gateway nodes). Furthermore, security groups with the rules specified during environment creation are provisioned to define inbound and outbound access to the instances.
Relational Database Service (RDS)	An RDS instance (db.m5.large) is provisioned for the Data Lake. This RDS instance is used for Cloudera Manager, Ranger, and Hive MetaStore.
Simple Storage Service (S3)	The existing S3 that you provide during environment creation for the Data Lake is used for Data Lake log storage and workload data storage.

AWS resources created for Cloudera Data Hub

The following AWS resources are created for the Cloudera Data Hub service:

Resource	Description
CloudFormation	A CloudFormation stack is created for each Cloudera Data Hub cluster to create instances and disks. This generates an AWS stack which links and describes the resources of your Cloudera Data Hub cluster. Multi-AZ deployments do not use a CloudFormation template for VM creation. Neither autoscaling groups or launch templates are created. In a multi-AZ setup the cluster resources are managed individually using AWS native components (for example, EC2 instances). **Caution:** Do not attempt to change the configuration of the Auto Scaling groups or the launch template. Any manual modifications (such as modifying suspended processes or subnets) that you make to the Auto Scaling groups or the launch template are not supported and may cause future errors or failures, particularly during repair processes.
Elastic Compute Cloud (EC2)	An EC2 instance is created for each cluster node. The instance type varies depending on what you selected during Cloudera Data Hub cluster creation. For each instance, attached storage is provisioned. The storage size and type varies depending on what you selected during cluster creation. Furthermore, security groups with the rules specified during environment creation are provisioned to define inbound and outbound access to the instances. For a list of supported EC2 instance types, refer to Cloudera Public Cloud service rates.
Relational Database Service (RDS)	Cloudera Data Hub connects to the Hive MetaStore database on the RDS instance provisioned for the Data Lake.
Simple Storage Service (S3)	The existing S3 bucket that you provided for the Data Lake to use for workload data storage can be accessed from Cloudera Data Hub clusters via the S3A connector.
Auto Scaling	Cloudera Data Hub uses the Auto Scaling service for upscaling clusters, except in a multi-AZ deployment. **Caution:** After a Cloudera Data Hub has been created, do not attempt to change the configuration of the Auto Scaling groups or the launch template. Any manual modifications (such as modifying suspended processes or subnets) that you make to the Auto Scaling groups or the launch template are not supported and may cause future errors or failures, particularly during repair processes.
Key Management Service (KMS)	Cloudera Data Hub uses KMS for encrypting your disks if during Cloudera Data Hub cluster creation you select to use disk encryption.

AWS resources created for Cloudera Data Engineering

The following AWS resources are created for the Cloudera Data Engineering service:

Resource	Description
CloudFormation	The initial deployment of services such as the EKS cluster is orchestrated through CloudFormation. This generates an AWS stack which links and describes the resources of your Cloudera Data Engineering cluster.
Elastic Compute Cloud (EC2)	Cloudera Data Engineering uses EC2 instances as cluster nodes. For a list of supported EC2 instance types, refer to Cloudera Public Cloud service rates.
Auto Scaling	Cloudera Data Engineering uses AWS AutoScaler to add or remove EC2 instances to the kubernetes cluster. Whenever the kubernetes cluster is running low on resources, new EC2 instances are provisioned and jointed into the EKS cluster. Whenever the AutoScaler detects an over-provisioning of resources, it removes and suspends EC2 instances.
Elastic Kubernetes Service (EKS)	EKS is the AWS implementation of the kubernetes stack. All PODs are running within an EKS cluster (one per environment).

Resource	Description
ELB Classic Load Balancer	Cloudera Data Engineering uses Classic Load Balancers for redirecting traffic to EC2 instances.
Key Management Service (KMS)	Cloudera Data Engineering uses KMS for encrypting your disks if you select to use disk encryption.
Elastic Block Store (EBS)	Cloudera Data Engineering uses EBS for persistent instance storage.
Elastic File System (EFS)	Cloudera Data Engineering uses EFS for persistent service and virtual cluster storage.
Relational Database Service (RDS)	Cloudera Data Engineering uses RDS for provisioning relational databases.

AWS resources created for Cloudera DataFlow

The following AWS resources are created for the Cloudera DataFlow service:

Resource	Description
CloudFormation	The initial deployment of services such as the EKS cluster is orchestrated through CloudFormation. This generates an AWS stack which links and describes the resources of your Cloudera DataFlow cluster.
Elastic Compute Cloud (EC2)	Cloudera DataFlow uses EC2 instances as cluster nodes. For a list of supported EC2 instance types, refer to Cloudera Public Cloud service rates.
Auto Scaling	Cloudera DataFlow uses AWS AutoScaler to add or remove EC2 instances to the Kubernetes cluster. Whenever the Kubernetes cluster is running low on resources, new EC2 instances are provisioned and jointed into the EKS cluster. Whenever the AutoScaler detects an over-provisioning of resources, it removes and suspends EC2 instances.
Elastic Kubernetes Service (EKS)	EKS is the AWS implementation of the kubernetes stack. All PODs are running within an EKS cluster (one per environment).
ELB Classic Load Balancer	Cloudera DataFlow uses Classic Load Balancers for redirecting traffic to EC2 instances.
Elastic Block Store (EBS)	Cloudera DataFlow uses EBS for persistent instance storage.
Relational Database Service (RDS)	Cloudera DataFlow uses RDS for provisioning relational databases.

AWS resources created for Cloudera Data Warehouse

The following AWS resources are created for the Cloudera Data Warehouse service:

Resource	Description
Identity and Access Management (IAM)	During Cloudera Data Warehouse cluster provisioning, the Cloudera Data Warehouse service creates an IAM role that defines access to S3 and other provisioned resources. Such role is then attached to the EC2 instance profile to grant PODs within the kubernetes environment access to these resources.
Certificate Manager	Cloudera Data Warehouse creates, stores, and maintains a certificate in the AWS certificate manager. This certificate is used to allow HTTPS connections to the external facing endpoints (i.e. for JDBC or the DAS UI). The certificate is signed by a trusted certificate authority, therefore external consumers and browser can securely connect to Cloudera Data Warehouse services without having to deal with untrusted CA or self-signed certificates.

Resource	Description
CloudFormation	The initial deployment of services such as the EKS cluster, the Cloudera Data Warehouse-specific RDS database, and S3 buckets is orchestrated through CloudFormation. This generates an AWS stack which links and describes the resources of your Cloudera Data Warehouse cluster.
Elastic Compute Cloud (EC2)	Cloudera Data Warehouse uses EC2 instances as cluster nodes. Two different EC2 instance types (through two different auto scaler groups) are used to support shared services and compute requirements within the cluster: m5.2xlarge for always on components, and r5d.4xlarge for compute nodes (Hive and Impala executors). Furthermore, security groups with the rules specified during environment creation are provisioned to define inbound and outbound access to the instances. For a list of supported EC2 instance types, refer to Cloudera Public Cloud service rates.
Simple Storage Service (S3)	Cloudera Data Warehouse creates its own S3 buckets (separate from the environment's S3 bucket(s)) for storing data and logs.
Auto Scaling	Cloudera Data Warehouse uses AWS AutoScaler to add or remove EC2 instances to the kubernetes cluster. Whenever the kubernetes cluster is running low on resources, new EC2 instances are provisioned and jointed into the EKS cluster. Whenever the AutoScaler detects an over-provisioning of resources, it removes and suspends EC2 instances.
Elastic File System (EFS)	EFS is used as shared filesystem across PODs to persist data (i.e. result cache).
Elastic Load Balancing (ELB)	All inbound traffic is routed through ELB towards the ingress controller of the kubernetes cluster. The ELB is provisioned as a result of the kubernetes ingress controller, which is the single point of entry for services, running in the kubernetes cluster.
Managed Kubernetes Service (EKS)	EKS is the AWS implementation of the kubernetes stack. All DW-deployed PODs are running within an EKS cluster (one per environment).
Key Management Service (KMS)	Cloudera Data Warehouse encrypts data at rest in S3. This requires an encryption key to be generated and stored in KMS. The key is completely under the control of AWS and cannot be exported or otherwise extracted. The S3 buckets are directly referencing the key within KMS, using it to encrypt the stored data.
Relational Database Service (RDS)	During cluster provisioning, Cloudera Data Warehouse provisions an RDS instance to be used as backend database system for metadata, managed and stored by the HMS instances, represented by "DB Catalogs". Each DB Catalog is implemented as separate database within this single RDS instance.
Security Token Service (STS)	STS is used to generate access tokens (based on roles) to access the resources within the environment's VPC.

AWS resources created for Cloudera AI

The following AWS resources are created for the Cloudera AI service:

Resource	Description
Identity and Access Management (IAM)	Cloudera AI creates additional IAM roles and policies for each cluster. Such roles are then attached to the EC2 instance profile.
Elastic Block Store (EBS)	Cloudera AI uses EBS as block storage.
Elastic Load Balancer (ELB)	Cloudera AI uses Classic Load Balancers for redirecting traffic to EC2 instances.
Key Management Service (KMS)	Cloudera AI uses KMS for encrypting your disks if you select to use disk encryption.
Elastic File System (EFS)	EFS is used for project file storage.

Resource	Description
Elastic Compute Cloud (EC2)	Cloudera AI uses EC2 instances as cluster nodes. Three different EC2 instance types (through three different auto scaler groups) are used to support Cloudera AI infra and compute requirements within the kubernetes cluster. Furthermore, security groups with the rules specified during environment creation are provisioned to define inbound and outbound access to the instances. For a list of supported EC2 instance types, refer to Cloudera Public Cloud service rates.
Auto Scaling	Cloudera AI uses AWS AutoScaler to add or remove EC2 instances to the kubernetes cluster. Whenever the kubernetes cluster is running low on resources, new EC2 instances are provisioned and jointed into the cluster. Whenever the AutoScaler detects an over-provisioning of resources, it removes and suspends EC2 instances.
Simple Storage Service (S3)	Cloudera AI uses S3 as the primary store for data and logs.
Security Token Service (STS)	STS is used to generate access tokens (based on roles) to access the resources within the environment's VPC.
Managed Kubernetes Service (EKS)	EKS is the AWS implementation of the kubernetes stack. All PODs are running within an EKS cluster (one per environment).

AWS resources created for Cloudera Operational Database

The following AWS resources are created for the Cloudera Operational Database service:

Resource	Description
CloudFormation	A CloudFormation stack is created for each Cloudera Operational Database database to create instances and disks. This generates an AWS stack which links and describes the resources of your Cloudera Operational Database database.
Elastic Compute Cloud (EC2)	An EC2 instance is created for each node. The instance type, storage size, and storage type is determined automatically by Cloudera Operational Database. Furthermore, security groups with the rules specified during environment creation are provisioned to define inbound and outbound access to the instances.
Simple Storage Service (S3)	The existing S3 bucket that you provided for the Data Lake to use for workload data storage can be accessed from Cloudera Operational Database database via the S3A connector.
Relational Database Service (RDS)	An RDS instance is provisioned for the Cloudera Operational Database. This RDS instance is used by Cloudera Manager.

AWS outbound network access destinations

If you have limited outbound internet access (for example due to using a firewall or proxy), review this content to learn which specific outbound destinations must be available in order to register a Cloudera environment.

We recommend hostname-based policies, as some of the destination services do not have static IP addresses. IP address details in CIDR notation have been provided where static IPs are in-use.

Note:

If the cloud provider network that you would like to use for registering a Cloudera environment uses a custom DNS server that does not allow name resolution for public domain, you should add all the domains listed in the below tables to the DNS forwarder for name resolution.

The following list includes general destinations as well as AWS-specific destinations.

General endpoints

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
Control Plane API	All services	US-based Control Plane: api.us-west-1.cdp.cloudera.com EU-based Control Plane: api.eu-1.cdp.cloudera.com AP-based Control Plane: api.ap-1.cdp.cloudera.com	HTTPS with Cloudera-generated access key	TCP/443	Cloudera Control Plane REST API.
Cloudera CCMv1 Persistent Control Plane connection	All services	*.ccm.cdp.cloudera.com 44.234.52.96/27	SSH public/ private key authentication	TCP/6000-6049	One connection per cluster configured; persistent
Cloudera CCMv2 Persistent Control Plane connection	All services	US-based Control Plane: *.v2.us-west-1.ccm.cdp.cloudera.com 35.80.24.128/27 EU-based Control Plane: *.v2.ccm.eu-1.cdp.cloudera.com 3.65.246.128/27 AP-based Control Plane: *.v2.ccm.ap-1.cdp.cloudera.com 3.26.127.64/27	HTTPS with mutual authentication	TCP/443	Multiple long-lived/persistent connections
Cloudera Databus Telemetry, billing and metering data	All services	US-based Control Plane: dbusapi.us-west-1.sigma.altus.cloudera.com api.us-west-1.cdp.cloudera.com https://cloudera-dbus-prod.s3.amazonaws.com EU-based Control Plane: api.eu-1.cdp.cloudera.com https://mow-prod-eu-central-1-sigmadbus-dbus.s3.eu-central-1.amazonaws.com https://mow-prod-eu-central-1-sigmadbus-dbus.s3.amazonaws.com AP-based Control Plane: api.ap-1.cdp.cloudera.com https://mow-prod-ap-southeast-2-sigmadbus-dbus.s3.ap-southeast-2.amazonaws.com https://mow-prod-ap-southeast-2-sigmadbus-dbus.s3.amazonaws.com	HTTPS with Cloudera-generated access key for dbus HTTPS for S3	TCP/443	Regular interval for telemetry, billing, metering services, and used for Cloudera Observability if enabled. Larger payloads are sent to a Cloudera managed S3 bucket.

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
Cloudera Observability Metrics System metrics collection	All services	US-based Control Plane: *.api.monitoring.us-west-1.cdp.cloudera.com EU-based Control Plane: *.api.monitoring.eu-1.cdp.cloudera.com AP-based Control Plane: *.api.monitoring.ap-1.cdp.cloudera.com	HTTPS	TCP/443	New as of March 2024
Cloudera Manager parcels Software distribution	All services	archive.cloudera.com	HTTPS	TCP/443	Cloudera's public software repository. CDN backed service; IP range not predictable.
RPMs Cloudera RPMs for workload agents	All services	cloudera-service-delivery-cache.s3.amazonaws.com	HTTPS	TPC/443	RPM packages for some workload components
Container Images Software Distribution	Cloudera Data Engineering Cloudera DataFlow Cloudera Data Warehouse Cloudera AI	container.repo.cloudera.com container.repository.cloudera.com container.repo.cloudera.com prod-us-west-2-starport-layer-bucket.s3.us-west-2.amazonaws.com prod-us-west-2-starport-layer-bucket.s3.amazonaws.com s3-r-w.us-west-2.amazonaws.com *.execute-api.us-west-2.amazonaws.com prod-eu-west-1-starport-layer-bucket.s3.eu-west-1.amazonaws.com prod-eu-west-1-starport-layer-bucket.s3.amazonaws.com s3-r-w.eu-west-1.amazonaws.com *.execute-api.eu-west-1.amazonaws.com prod-ap-southeast-1-starport-layer-bucket.s3.ap-southeast-1.amazonaws.com prod-ap-southeast-1-starport-layer-bucket.s3.amazonaws.com s3-r-w.ap-southeast-1.amazonaws.com *.execute-api.ap-southeast-1.amazonaws.com	HTTPS	TCP/443	CDN-backed and AWS ECR-backed services; IP range not predictable. container.repo.cloudera.com uses ECR backend which requires S3 URLs. IP geolocation attempts to select closest API and ECR backend; clients may be directed to any of the destinations.

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
Flow Definitions Cloudera AWS bucket with flow definitions	Cloudera DataFlow	US-based Control Plane: s3.us-west-2.amazonaws.com/dfx-flow-artifacts.mow-prod.mow-prod.cloudera.com EU-based Control Plane: cldr-mow-prod-eu-central-1-dfx-flow-artifacts.s3.eu-central-1.amazonaws.com AP-based Control Plane: cldr-mow-prod-ap-southeast-2-dfx-flow-artifacts.s3.ap-southeast-2.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	Outbound internet access to S3 hosts is necessary on all cloud providers when using Cloudera DataFlow as the workload needs to query outbound to an S3 location to retrieve the flow definition when creating a deployment.
Public Signing Key Retrieval	Cloudera Data Engineering Cloudera DataFlow	US-based Control Plane: consoleauth.altus.cloudera.com console.us-west-1.cdp.cloudera.com EU-based Control Plane: console.eu-1.cdp.cloudera.com AP-based Control Plane: console.ap-1.cdp.cloudera.com	HTTPS	TCP/443	Required to allow authentication to Cloudera Data Engineering virtual Cluster using a Cloudera Access Key.
Control Plane IAM API	Cloudera AI	US-based Control Plane: iamapi.us-west-1.altus.cloudera.com console.us-west-1.cdp.cloudera.com EU-based Control Plane: console.eu-1.cdp.cloudera.com AP-based Control Plane: console.ap-1.cdp.cloudera.com	HTTPS	TCP/443	For connecting to the IAMAPI for fetching the entitlement details.
AMPs Cloudera Accelerators for Machine Learning Projects	Cloudera AI	https://raw.githubusercontent.com https://github.com	HTTPS	TCP/443	Files for AMPs are hosted on GitHub.
Learning Hub	Cloudera AI	https://github.com/cloudera/learning-hub-content	HTTPS	TCP/443	Access Learning Hub in air-gapped environments

AWS-specific endpoints

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
AWS STS	All services	sts.amazonaws.com sts.*.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	Cloudera 7.1.1+ required before can be made internal with VPC endpoints.

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
AWS S3	All services	*.s3.amazonaws.com *.s3.<AWS_REGION>.amazonaws.com s3.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	The <AWS_REGION> should be replaced with the AWS region used for your workloads. *.s3.<AWS_REGION>.amazonaws.c is VPC internal. *.s3.amazonaws.com and s3.amazonaws.com can be made internal with VPC endpoints.
AWS RDS	All services	*.*.rds.amazonaws.com	JDBC / Postgres binary protocol / MySQL / RDS CA certs	TCP 5432 / 3306 / 443	VPC Internal. Only Cloudera Data Engineering uses MySQL and requires port 3306 to be open.
AWS EC	Cloudera DataFlow Cloudera Data Warehouse Cloudera AI	api.ecr.<AWS_REGION>.amazonaws.com *.dkr.ecr.<AWS_REGION>.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	VPC Internal. The <AWS_REGION> should be replaced with the AWS region used for your workloads.
AWS EC2	Cloudera DataFlow Cloudera Data Warehouse Cloudera AI Cloudera Operational Database	ec2.<AWS_REGION>.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	VPC Internal. The <AWS_REGION> should be replaced with the AWS region used for your workloads.
AWS EKS	Cloudera Data Engineering Cloudera DataFlow Cloudera Data Warehouse Cloudera AI	eks.*.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	AWS does not support EKS VPC endpoints at this time.
AWS Cloudformation	Cloudera DataFlow Cloudera Data Warehouse Cloudera AI	cloudformation.*.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	Can be made internal with VPC endpoints.

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
AWS Autoscaling	Cloudera Data Engineering Cloudera DataFlow Cloudera Data Warehouse Cloudera AI	autoscaling.*.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	Can be made internal with VPC endpoints.
AWS EFS	Cloudera Data Engineering Cloudera Data Warehouse Cloudera AI	elasticfilesystem.*.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	Can be made internal with VPC endpoints.
AWS ELB	Cloudera Data Engineering Cloudera Data Warehouse	elasticloadbalancing.*.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	Can be made internal with VPC endpoints.
AWS EKS k8s cluster API	Cloudera Data Warehouse	<UNIQUEID>.*.eks.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	Optional for new clusters. The <UNIQUEID> should be replaced with the unique hostname that is assigned when an EKS k8s cluster is deployed.
AWS RDS API	Cloudera Data Warehouse	rds.*.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	AWS does not support RDS API VPC endpoints at this time. This requirement is under further evaluation. Cloudera Data Warehouse uses Amazon RDS for PostgreSQL.
AWS Service Quotas	Cloudera Data Warehouse	servicequotas.*.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	AWS does not support Service Quota via VPC endpoints. Used to check limits and warn prior to hitting the limits.
AWS Price List Service	Cloudera DataFlow Cloudera Data Warehouse	pricing.*.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	AWS Price List Service uses us-east-1 or ap-south-1 as the region.

Access to workload UIs

If you have restricted DNS or networking setup, make sure that *.cloudera.site is resolvable from your network so that members of your organization can access workload UIs.

Cloudera workloads (including Data Lake) use subdomains under cloudera.site to host various UI endpoints (Cloudera Manager, Ranger, Knox, Hue and so on). Cloudera automatically provisions these endpoints whenever a Data Lake, Cloudera Data Hub or another type of workload (for example, Virtual Warehouse in Cloudera Data Warehouse) is created, and routing is set up so that you can access these endpoints from your network.

The subdomains are assigned under cloudera.site using the following convention:

```
<endpoint-name>.<env-truncated-name>.<customer-workload-subdomain>.<regional
-subdomain>.cloudera.site
```

Supported browsers

Cloudera validates and tests against the latest version and supports recent versions of the following browsers:

- Google Chrome
- Mozilla Firefox

  Note:

  - Mozilla Firefox is not supported by Cloudera Data Engineering.
  - Certain accessibility features in Cloudera DataFlow do not work in Mozilla Firefox.
- Safari
- Microsoft Edge

Other resources

While this document attempts to provide a complete overview of cloud provider requirements, there is additional documentation that you should review if planning to deploy Cloudera data services.

The following table includes links to documentation that you should review if planning to deploy the respective Cloudera data service:

Cloudera data service	Documentation links
Cloudera Data Engineering	Cloudera Data Engineering AWS prerequisites Cloudera Data Engineering cost management Cloudera Data Engineering performance management
Cloudera Data Warehouse	Virtual Warehouse sizing requirements for public cloud environments Virtual Warehouse IP address and cloud resource requirements for Public Cloud environments Managing costs in the public cloud environments for Cloudera Data Warehouse AWS environment requirements checklist

Cloudera data service	Documentation links
Cloudera DataFlow	AWS requirements for Cloudera DataFlow Cloudera DataFlow AWS networking Cloudera DataFlow AWS limits
Cloudera AI	Cloudera AI limitations on AWS Cloudera AI restricted policies for AWS Use a non-transparent proxy with Cloudera AI on AWS environments
Cloudera Operational Database	Cloudera Operational Database cloud checklist AWS requirements for Cloudera Operational Database deployment

Cloudera CIDR

List of Cloudera Control Plane source IP addresses

In cases where the Cloudera Control Plane is communicating directly with resources in the Cloudera environment, the Cloudera Control Plane source IP address will be from the ranges below.

Security groups deployed/managed by Cloudera Control Plane will automatically include the required rules. These source IP ranges will be included if the Cloudera Control Plane requires direct inbound access to the resource.

Cloudera Control Plane Region	IP Ranges
us-west-1	35.80.24.128/27, 35.166.86.177/32, 52.36.110.208/32, 52.40.165.49/32
eu-1	3.65.246.128/27
ap-1	3.26.127.64/27

If the security groups (or other network access policies) in the network path are not managed by the Cloudera Control Plane, ensure that inbound access from the appropriate range(s) are permitted.

Cloudera Public Cloud

Azure Requirements

Date published: 2019-08-22
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Azure subscription requirements 5

Cloudera images hosted in Azure Marketplace 5
Private Azure Marketplace prerequisites 9

Azure resources and services 11
Azure credential prerequisites 11
Azure permissions 11
Obtain subscription and tenant ID 16
Create an app registration and assign a role to it 17
Azure region 20
Supported Azure regions 20
Resource groups 22
VNet and subnets 22
VNet and subnet planning 24
Private setup for Azure Flexible Server 27
Using Cloudera-managed private DNS 28
Bringing your own private DNS 29
Resources created under the hood 31
Private setup for Azure Single Server 31
Service endpoint for Azure Postgres 31
Private endpoint for Azure Postgres 32
Network security groups 37
Default Azure security groups 41
SSH key pair 44
Virtual machines 44
Custom images 45
Azure cloud storage prerequisites 45
Minimal setup for Azure cloud storage 45
Onboarding Cloudera users and groups (RAZ) 53
Onboarding Cloudera users and groups (No RAZ) 53
Using ADLS Gen2 encryption 57
Storage account for OS images 57
Creating a storage account 58
Copying an image manually 59
Creating an image resource 60
Supported Azure block storage 61
Azure Database for PostgreSQL 61
Encrypting VM disks with customer managed keys 62
Add additional permissions to your Azure policy 63
Create a vault and add a vault key 64
Managed identity for encrypting Azure Database for PostgreSQL Flexible Server 68
Encrypting a storage account with a key vault that has role-based access control 69
Azure Files storage account and file share for Cloudera AI 70
Azure Files NFS for Cloudera AI 70
Azure quota limits 70
List of Azure resources 71

Azure outbound network destinations..... 76

Access to workload UIs.....80

Supported browsers.....80

Other resources.....81

Cloudera CIDR.....81

Azure subscription requirements

Before registering your Azure environment in Cloudera, use this document to verify that your Azure account has all the resources required by Cloudera and that your Cloudera administrator has adequate permissions to configure the resources and services in Azure.

As an administrator, you must be able to create and manage the resources in the Azure subscription where Cloudera users create clusters and run jobs. You must be able to perform all administrative tasks and have administrative rights to all resources. Cloudera recommends that the administrator has the role of Owner in the Azure subscription and the Application Developer role or higher in the Azure Active Directory.

Related Information

Cloudera images hosted in Azure Marketplace
Azure resources and services
Overview of Azure resources used by Cloudera
Azure outbound network access destinations
Access to workload UIs
Supported browsers
Other resources

Cloudera images hosted in Azure Marketplace

Cloudera publishes virtual hard disks (VHD) images on Azure Marketplace for each minor Runtime release. Cloudera uses these images by default during environment and Cloudera Data Hub creation on Azure cloud.

By default, Cloudera uses VHD images for deploying a Cloudera environment and Cloudera Data Hub clusters. The VHD images go through Microsoft's certification process and are then published on Azure Marketplace for each minor version of Cloudera Runtime (for example, 7.2.17). Red Hat Enterprise Linux 8 (RHEL 8) images (for Cloudera Runtime 7.2.17 and newer only) and CentOS 7 images (for Cloudera Runtime 7.2.17 and earlier only) are available.

In order for Cloudera to be able to load Cloudera-published virtual machine images in your subscription from the Azure Marketplace, you must first accept Azure Marketplace terms and conditions either via Azure CLI or Cloudera web UI. If you do not accept the terms and conditions, Cloudera cannot access the images hosted in Azure Marketplace and so it downloads them from Cloudera storage account instead.

Prior to introducing this feature, Cloudera published VHD images for Azure in regional repositories and copied these images to customer's storage account before environment creation. The images would then remain in the storage account to speed up the provisioning of future environments and clusters. If you use CentOS and you do not fulfill the requirements for Azure Marketplace images, Cloudera falls back to this storage account method.

Note: RHEL 8 images are not available using the storage account method and are only available through Azure Marketplace.

The Cloudera images hosted in Azure Marketplace allow you to:

- Deploy your Cloudera environment, Data Lake and Cloudera Data Hub clusters faster than when using an image hosted in Cloudera's storage account as there is no need to copy the VHD image anymore.
- Use RHEL images instead of CentOS images for your Cloudera environment, Data Lake and Cloudera Data Hub clusters. When using Cloudera on Azure, RHEL 8 images are only available in the Marketplace for Azure.

Cloudera recommends that for best security, performance, and cost-effectiveness you use the Azure Marketplace images.

There are two ways to accept Azure Marketplace terms and conditions:

- You can do this by enabling auto-acceptance via the Cloudera web interface. In this case, you need to do it one time only.
- You can do this via Azure CLI. In this case, you need to do it individually for each image set corresponding to a single Cloudera Runtime version.

Cloudera recommends enabling auto-acceptance via the Cloudera web interface.

Accepting the terms and conditions via Azure CLI

You can accept Azure Marketplace terms and conditions via Azure CLI. In this case, you need to do it individually for each image offering corresponding to a single Runtime version.

Steps

1. Run the following command to obtain image URN for a specific minor Cloudera Runtime version (this is, a three-digit version such as 7.2.17):

```
az vm image list -p <NAME-OF-THE-PUBLISHER> --all -f <IMAGE-OFFER>
```

For example:

```
az vm image list -p Cloudera --all -f cdp-7_2_17
  {
    "offer": "cdp-7_2_17",
    "publisher": "cloudera",
    "sku": "runtime-7_2_17",
    "urn": "cloudera:cdp-7_2_17:runtime-7_2_17:100.44441663.1694778077",
    "version": "100.44441663.1694778077"
  },
```

2. Run the following command to print the terms and conditions details for the specific image, replacing the <SUBSCRIPTION-NAME-OR-ID> with an actual Azure subscription ID:

```
az vm image terms show \
  --urn <IMAGE-URN> \
  --subscription "<SUBSCRIPTION-NAME-OR-ID>"
```

For example:

```
az vm image terms show \
  --urn cloudera:cdp-7_2_17:runtime-7_2_17:100.44441663.1694778077 \
  --subscription "azure-eng-cloud-daily"
```

Among other information, the following details will be printed:

```
  "accepted": false,
  "id": "/subscriptions/<subscription-id>/providers/Microsoft.Marketpla
ceOrdering/offerTypes/VirtualMachine/publishers/cloudera/offers/cdp-7_2/
plans/runtime-7_2_17/agreements/current",
  "licenseTextLink": "https://mpcprodsa.blob.core.windows.net/legalterms/3
E5ED_legalterms_CLOUDERA%253a24CDP%253a2D7%253a5F2%253a24RUNTIME%253a2D7
%253a5F2%253a5F6%253a24LLJOPLSWAKNIAHKFQAUSJDXQEUIJ2TS5XNFDHNDYUPRK3LU2T
2DJPN2S32U3RVTR4DT2SVCUE5BYHA5UYFXGLSHFKCNQH7YIUS4JNXI.txt",
  "marketplaceTermsLink": "https://mpcprodsa.blob.core.windows.net/mark
etplaceterms/3EDEF_marketplaceterms_VIRTUALMACHINE%253a24AAK2OAIZEAWW5H4
MSP5KSTVB6NDKKRTUBAU23BRFTWN4YC2MQLJUB5ZEYUOUJBVF3YK34CIVPZL2HWYASPGDUY5
O2FWEGRBYOXWZE5Y.txt"
```

3. Accept the image terms by issuing the following command. The command needs to be issued individually for each image and subscription:

```
az vm image terms accept \
```

```
  --urn <image-urn> \
  --subscription "<subscription-name-or-id>"
```

For example:

```
az vm image terms accept \
  --urn cloudera:cdp-7_2_17:runtime-7_2_17:100.44441663.1694778077 \
  --subscription "azure-eng-cloud-daily"
```

Accepting the terms and conditions via Cloudera UI

You can enable auto-acceptance of Azure Marketplace terms and conditions via Cloudera web interface. This allows you to accept automatically for all Runtime versions.

Prerequisites

- You need to grant the service principal the following Azure permissions on the scope of your Azure subscription:

```
"Microsoft.MarketplaceOrdering/offertypes/publishers/offers/plans/agreemen
ts/write",
"Microsoft.MarketplaceOrdering/offerTypes/publishers/offers/plans/agreeme
nts/read”
```

Note: Granting this permissions on the scope of an Azure resource group only is not sufficient. You must grant them on the scope of your Azure subscription.

- You need to grant the service principal the following Azure permission on the scope of the Cloudera Azure resource group:

```
“Microsoft.Resources/deployments/whatIf/action”
```

The Contributor role in Azure includes these permissions.

The Prerequisites for the provisioning credential: Azure permissions have been updated to include the aforementioned permissions.

Required roles

You need to be EnvironmentCreator or PowerUser in Cloudera.

Steps

Navigate to the Cloudera Management Console > Global Settings > Azure tab and check the box next to “Enable Terms & Conditions automatic acceptance”. This allows Cloudera to automatically accept Cloudera image terms and conditions for Azure Marketplace:

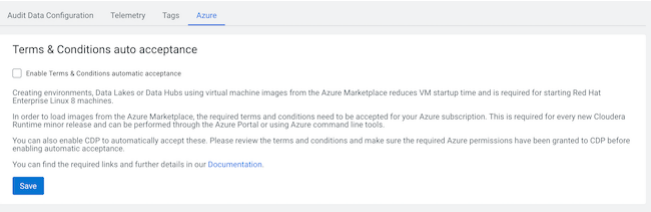

Accepting the terms and conditions via Azure CLI

You can accept Azure Marketplace terms and conditions via Azure CLI. In this case, you need to do it individually for each image offering corresponding to a single Runtime version.

Steps

1. Run the following command to obtain image URN for a specific minor Runtime version (this is, a three-digit version such as 7.2.17):

```
az vm image list -p <NAME-OF-THE-PUBLISHER> --all -f <IMAGE-OFFER>
```

For example:

```
az vm image list -p Cloudera --all -f cdp-7_2_17
  {
    "offer": "cdp-7_2_17",
    "publisher": "cloudera",
    "sku": "runtime-7_2_17",
    "urn": "cloudera:cdp-7_2_17:runtime-7_2_17:100.44441663.1694778077",
    "version": "100.44441663.1694778077"
  },
```

2. Run the following command to print the terms and conditions details for the specific image, replacing the <SUB SCRIPTION-NAME-OR-ID> with an actual Azure subscription ID:

```
az vm image terms show \
  --urn <IMAGE-URN> \
  --subscription "<SUBSCRIPTION-NAME-OR-ID>"
```

For example:

```
az vm image terms show \
  --urn cloudera:cdp-7_2_17:runtime-7_2_17:100.44441663.1694778077 \
  --subscription "azure-eng-cloud-daily"
```

Among other information, the following details will be printed:

```
  "accepted": false,
  "id": "/subscriptions/<subscription-id>/providers/Microsoft.Marketpla
ceOrdering/offerTypes/VirtualMachine/publishers/cloudera/offers/cdp-7_2/
plans/runtime-7_2_17/agreements/current",
  "licenseTextLink": "https://mpcprodsa.blob.core.windows.net/legalterms/3
E5ED_legalterms_CLOUDERA%253a24CDP%253a2D7%253a5F2%253a24RUNTIME%253a2D7
%253a5F2%253a5F6%253a24LLJOPLSWAKNIAHKFQAUSJDXQEUIJ2TS5XNFDHNDYUPRK3LU2T
2DJPN2S32U3RVTR4DT2SVCUE5BYHA5UYFXGLSHFKCNQH7YIUS4JNXI.txt",
  "marketplaceTermsLink": "https://mpcprodsa.blob.core.windows.net/mark
etplaceterms/3EDEF_marketplaceterms_VIRTUALMACHINE%253a24AAK2OAIZEAWW5H4
MSP5KSTVB6NDKKRTUBAU23BRFTWN4YC2MQLJUB5ZEYUOUJBVF3YK34CIVPZL2HWYASPGDUY5
O2FWEGRBYOXWZE5Y.txt"
```

3. Accept the image terms by issuing the following command. The command needs to be issued individually for each image and subscription:

```
az vm image terms accept \
  --urn <image-urn> \
  --subscription "<subscription-name-or-id>"
```

For example:

```
az vm image terms accept \
  --urn cloudera:cdp-7_2_17:runtime-7_2_17:100.44441663.1694778077 \
  --subscription "azure-eng-cloud-daily"
```

Private Azure Marketplace prerequisites

If you would like to use Private Azure Marketplace with Cloudera, there are some manual configuration steps you need to do as a prerequisite.

About this task

If you would like to use Private Azure Marketplace with Cloudera, perform the following steps outside the context of Cloudera.

Procedure

1. Assign Marketplace Admin role in the scope of Private Azure Marketplace for your admin user through the Microsoft Azure portal. For more information, see *Assign the Marketplace admin role with access control (IAM)* in the Microsoft Azure documentation linked below in the *Related information*.
2. With the same user, create ("Get") Private Azure Marketplace. For more information, see *Create private Azure Marketplace*.
3. (Optional) Create a new collection.
4. Add at least the following items to the collection:
   - Cloudera Data Platform (CDP) Freeipa
   - Cloudera Data Platform (CDP) 7.x.xx
5. Make sure the collection is enabled.
6. Make sure the collection scope contains the deployment's subscription.
7. Set Private Marketplace status to Private Marketplace enabled.
8. Launch environments, Data Lakes, or Cloudera Data Hub clusters as desired.

Results

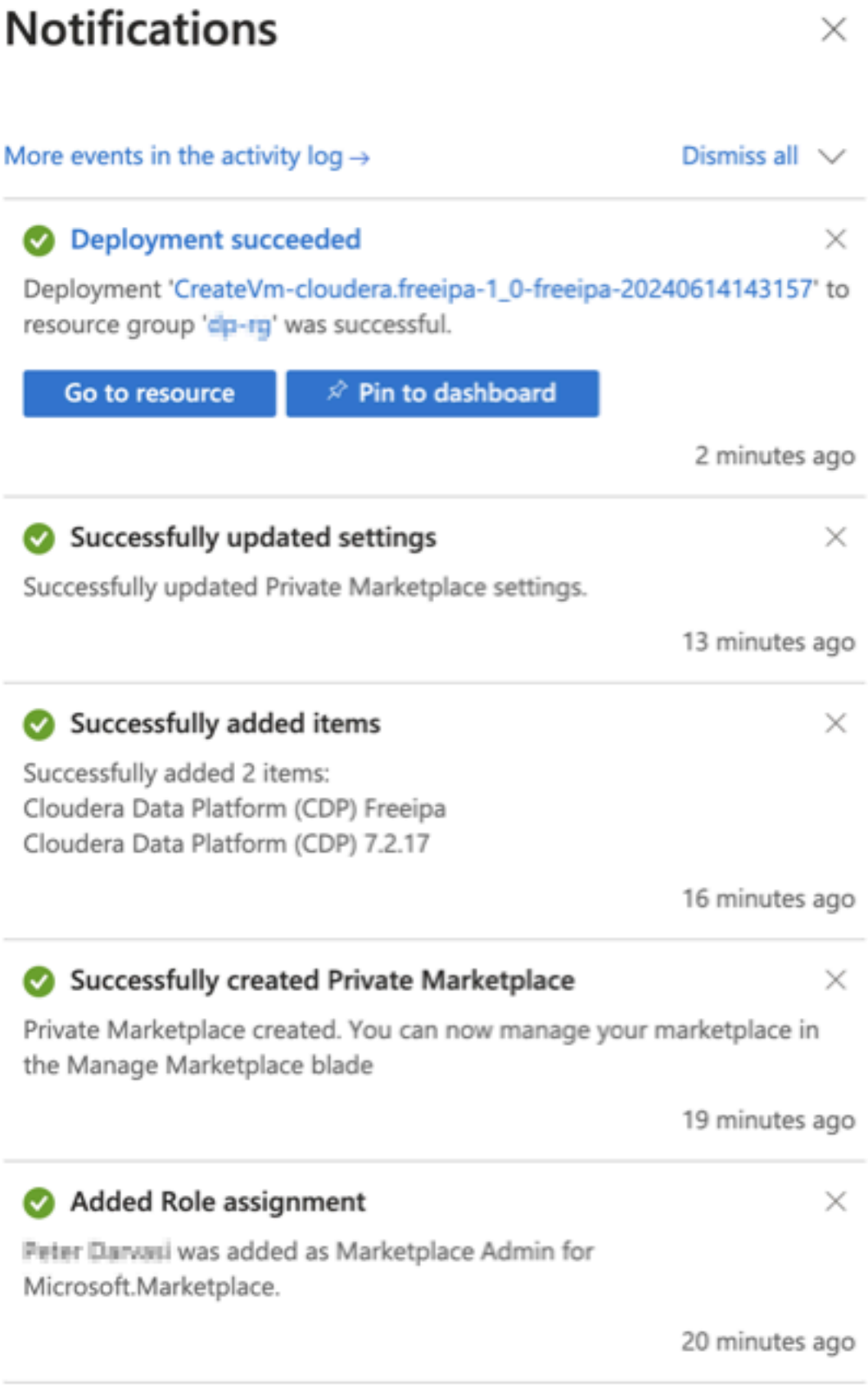
Notifications
More events in the activity log →
Dismiss all
Deployment succeeded
Deployment 'CreateVm-cloudera.freeipa-1_0-freeipa-20240614143157' to resource group '[illegible]' was successful.
Go to resource
Pin to dashboard
2 minutes ago
Successfully updated settings
Successfully updated Private Marketplace settings.
13 minutes ago
Successfully added items
Successfully added 2 items:
Cloudera Data Platform (CDP) Freeipa
Cloudera Data Platform (CDP) 7.2.17
16 minutes ago
Successfully created Private Marketplace
Private Marketplace created. You can now manage your marketplace in the Manage Marketplace blade
19 minutes ago
Added Role assignment
[illegible] was added as Marketplace Admin for Microsoft.Marketplace.
20 minutes ago

For more information, see *Govern and control using private Azure Marketplace* in the Microsoft Azure documentation.

Related Information

Assign the Marketplace admin role with access control (IAM)

Create private Azure Marketplace

Govern and control using private Azure Marketplace

Azure resources and services

Cloudera uses various resources in your Azure account.

Use the following guidelines to ensure that Cloudera has access to the resources in your Azure account and that your Azure account has all the necessary resources required by Cloudera:

Prerequisites for the provisioning credential

To allow Cloudera to create resources on your Azure account, you must create the app-based credential. The credential allows Cloudera to access and provision a set of resources in your Azure account.

Cloudera uses an app-based credential to authenticate your Azure account and obtain authorization to create resources on your behalf. The app-based credential requires that you to manually configure the service principal created within your Azure Active Directory. The app-based method requires Owner role to be able to create a service principal, which must be given Contributor role or its equivalent.

To meet Azure prerequisites for Cloudera:

1. Review the provided policies.
2. Obtain the subscription and tenant ID.
3. Create an app registration on Azure

Azure permissions

Your Azure administrator must create custom roles in the Azure subscription.

1. The administrator must create a custom roles containing the following sets of permissions sufficient for registering an environment and creating Cloudera Data Hub clusters and Cloudera Operational Database:
   a. The administrator must create a custom role containing one of the following sets of permissions on the scope of the resource group used for Cloudera:
      1. Option 1: Use the Role definition 1: Allows Cloudera to access and use only a single existing resource group and create service endpoints on page 12 if you would like to use service endpoints.
      2. Option 2: Use the Role definition 2: Allows Cloudera to access and use only a single existing resource group and create private endpoints on page 13 if you would like to use private endpoints.
   b. In order to use the Azure Marketplace images that Cloudera publishes, the administrator also needs to grant the service principal the additional Azure permissions on the scope of your Azure subscription. The Role definition for Azure Marketplace images on page 16 policy includes these permissions.
2. Additionally, if you would like to provision other Cloudera services (Cloudera Data Engineering, Cloudera DataFlow, Cloudera Data Warehouse, or Cloudera AI), you should assign the built-in Contributor Azure role either at the resource group level (if you are providing your own resource group) or at the Azure subscription level (if Cloudera is creating resource groups).
   - If you need Cloudera Data Warehouse only and Cloudera is creating resource groups, you can use the minimal policy documented in Cloudera Data Warehouse documentation.
   - If you need Cloudera DataFlow only and Cloudera is creating resource groups, you can use the minimal policy documented in Cloudera DataFlow documentation.

- If you need Cloudera AI only and Cloudera is creating resource groups, you can use the minimal policy documented in Cloudera AI documentation.

Role definition 1: Allows Cloudera to access and use only a single existing resource group and create service endpoints

The following role definition allows Cloudera to create resources only within the specified resource group:

```
{
    "Name": "Cloudera Management Console Azure Operator For Single Resource
 Group",
    "IsCustom": true,
    "Description": "Can use Cloudera Management Console managed clusters and
 resources updated for single resource group.",
    "Actions": [
      "Microsoft.Storage/storageAccounts/read",
      "Microsoft.Storage/storageAccounts/write",
      "Microsoft.Storage/storageAccounts/blobServices/write",
      "Microsoft.Storage/storageAccounts/blobServices/containers/delete",
      "Microsoft.Storage/storageAccounts/blobServices/containers/read",
      "Microsoft.Storage/storageAccounts/blobServices/containers/write",
      "Microsoft.Storage/storageAccounts/fileServices/write",
      "Microsoft.Storage/storageAccounts/listkeys/action",
      "Microsoft.Storage/storageAccounts/regeneratekey/action",
      "Microsoft.Storage/storageAccounts/delete",
      "Microsoft.Storage/locations/deleteVirtualNetworkOrSubnets/action",
      "Microsoft.Network/virtualNetworks/read",
      "Microsoft.Network/virtualNetworks/write",
      "Microsoft.Network/virtualNetworks/delete",
      "Microsoft.Network/virtualNetworks/subnets/read",
      "Microsoft.Network/virtualNetworks/subnets/write",
      "Microsoft.Network/virtualNetworks/subnets/delete",
      "Microsoft.Network/virtualNetworks/subnets/join/action",
      "Microsoft.Network/publicIPAddresses/read",
      "Microsoft.Network/publicIPAddresses/write",
      "Microsoft.Network/publicIPAddresses/delete",
      "Microsoft.Network/publicIPAddresses/join/action",
      "Microsoft.Network/networkInterfaces/read",
      "Microsoft.Network/networkInterfaces/write",
      "Microsoft.Network/networkInterfaces/delete",
      "Microsoft.Network/networkInterfaces/join/action",
      "Microsoft.Network/networkInterfaces/ipconfigurations/read",
      "Microsoft.Network/networkSecurityGroups/read",
      "Microsoft.Network/networkSecurityGroups/write",
      "Microsoft.Network/networkSecurityGroups/delete",
      "Microsoft.Network/networkSecurityGroups/join/action",
      "Microsoft.Network/virtualNetworks/subnets/joinViaServiceEndpoint/ac
tion",
      "Microsoft.Network/loadBalancers/delete",
      "Microsoft.Network/loadBalancers/read",
      "Microsoft.Network/loadBalancers/write",
      "Microsoft.Network/loadBalancers/backendAddressPools/join/action",
      "Microsoft.Compute/availabilitySets/read",
      "Microsoft.Compute/availabilitySets/write",
      "Microsoft.Compute/availabilitySets/delete",
      "Microsoft.Compute/disks/read",
      "Microsoft.Compute/disks/write",
      "Microsoft.Compute/disks/delete",
      "Microsoft.Compute/images/read",
      "Microsoft.Compute/images/write",
      "Microsoft.Compute/images/delete",
      "Microsoft.Compute/virtualMachines/read",
      "Microsoft.Compute/virtualMachines/write",
```

```
      "Microsoft.Compute/virtualMachines/delete",
      "Microsoft.Compute/virtualMachines/start/action",
      "Microsoft.Compute/virtualMachines/restart/action",
      "Microsoft.Compute/virtualMachines/deallocate/action",
      "Microsoft.Compute/virtualMachines/powerOff/action",
      "Microsoft.Compute/virtualMachines/vmSizes/read",
      "Microsoft.Authorization/roleAssignments/read",
      "Microsoft.Resources/subscriptions/resourceGroups/read",
      "Microsoft.Resources/deployments/read",
      "Microsoft.Resources/deployments/write",
      "Microsoft.Resources/deployments/delete",
      "Microsoft.Resources/deployments/operations/read",
      "Microsoft.Resources/deployments/operationstatuses/read",
      "Microsoft.Resources/deployments/exportTemplate/action",
      "Microsoft.Resources/subscriptions/read",
      "Microsoft.ManagedIdentity/userAssignedIdentities/read",
      "Microsoft.ManagedIdentity/userAssignedIdentities/assign/action",
      "Microsoft.DBforPostgreSQL/servers/read",
      "Microsoft.DBforPostgreSQL/servers/write",
      "Microsoft.DBforPostgreSQL/servers/delete",
      "Microsoft.DBforPostgreSQL/servers/virtualNetworkRules/write",
      "Microsoft.DBforPostgreSQL/flexibleServers/read",
      "Microsoft.DBforPostgreSQL/flexibleServers/write",
      "Microsoft.DBforPostgreSQL/flexibleServers/delete",
      "Microsoft.DBforPostgreSQL/flexibleServers/start/action",
      "Microsoft.DBforPostgreSQL/flexibleServers/stop/action",
      "Microsoft.DBforPostgreSQL/flexibleServers/firewallRules/write",
      "Microsoft.DBforPostgreSQL/flexibleServers/configurations/write",
      "Microsoft.Resources/deployments/cancel/action",
      "Microsoft.Resources/deployments/whatIf/action"
    ],
    "NotActions": [],
    "DataActions": [
      "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/re
ad",
      "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/writ
e",
      "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/delet
e",
      "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/add/a
ction"
    ],
    "NotDataActions": [],
    "AssignableScopes": [
      "/subscriptions/{SUBSCRIPTION-ID}/resourceGroups/{RESOURCE-GROUP-
NAME}"
    ]
  }
```

When creating the role definition, make sure to:

- Replace the {SUBSCRIPTION-ID} with your actual subscription ID.
- Replace the {RESOURCE-GROUP-NAME} with the ID of your existing resource group

Role definition 2: Allows Cloudera to access and use only a single existing resource group and create private endpoints

The following role definition allows Cloudera to create resources only within the specified resource group:

```
{
    "Name": "Cloudera Management Console Azure Operator for Single Resource
 Group",
    "IsCustom": true,
```

```
    "Description": "Can use Cloudera Management Console managed clusters and
 resources, updated for use with single resource group for all resources.",
    "Actions": [
      "Microsoft.Storage/storageAccounts/read",
      "Microsoft.Storage/storageAccounts/write",
      "Microsoft.Storage/storageAccounts/blobServices/write",
      "Microsoft.Storage/storageAccounts/blobServices/containers/delete",
      "Microsoft.Storage/storageAccounts/blobServices/containers/read",
      "Microsoft.Storage/storageAccounts/blobServices/containers/write",
      "Microsoft.Storage/storageAccounts/fileServices/write",
      "Microsoft.Storage/storageAccounts/listkeys/action",
      "Microsoft.Storage/storageAccounts/regeneratekey/action",
      "Microsoft.Storage/storageAccounts/delete",
      "Microsoft.Storage/locations/deleteVirtualNetworkOrSubnets/action",
      "Microsoft.Network/virtualNetworks/read",
      "Microsoft.Network/virtualNetworks/write",
      "Microsoft.Network/virtualNetworks/delete",
      "Microsoft.Network/virtualNetworks/subnets/read",
      "Microsoft.Network/virtualNetworks/subnets/write",
      "Microsoft.Network/virtualNetworks/subnets/delete",
      "Microsoft.Network/virtualNetworks/subnets/join/action",
      "Microsoft.Network/publicIPAddresses/read",
      "Microsoft.Network/publicIPAddresses/write",
      "Microsoft.Network/publicIPAddresses/delete",
      "Microsoft.Network/publicIPAddresses/join/action",
      "Microsoft.Network/networkInterfaces/read",
      "Microsoft.Network/networkInterfaces/write",
      "Microsoft.Network/networkInterfaces/delete",
      "Microsoft.Network/networkInterfaces/join/action",
      "Microsoft.Network/networkInterfaces/ipconfigurations/read",
      "Microsoft.Network/networkSecurityGroups/read",
      "Microsoft.Network/networkSecurityGroups/write",
      "Microsoft.Network/networkSecurityGroups/delete",
      "Microsoft.Network/networkSecurityGroups/join/action",
      "Microsoft.Compute/availabilitySets/read",
      "Microsoft.Compute/availabilitySets/write",
      "Microsoft.Compute/availabilitySets/delete",
      "Microsoft.Compute/disks/read",
      "Microsoft.Compute/disks/write",
      "Microsoft.Compute/disks/delete",
      "Microsoft.Compute/images/read",
      "Microsoft.Compute/images/write",
      "Microsoft.Compute/images/delete",
      "Microsoft.Compute/virtualMachines/read",
      "Microsoft.Compute/virtualMachines/write",
      "Microsoft.Compute/virtualMachines/delete",
      "Microsoft.Compute/virtualMachines/powerOff/action",
      "Microsoft.Compute/virtualMachines/start/action",
      "Microsoft.Compute/virtualMachines/restart/action",
      "Microsoft.Compute/virtualMachines/deallocate/action",
      "Microsoft.Compute/virtualMachines/vmSizes/read",
      "Microsoft.Authorization/roleAssignments/read",
      "Microsoft.Resources/subscriptions/resourceGroups/read",
      "Microsoft.Resources/deployments/read",
      "Microsoft.Resources/deployments/write",
      "Microsoft.Resources/deployments/delete",
      "Microsoft.Resources/deployments/operations/read",
      "Microsoft.Resources/deployments/operationstatuses/read",
      "Microsoft.Resources/deployments/exportTemplate/action",
      "Microsoft.Resources/subscriptions/read",
      "Microsoft.ManagedIdentity/userAssignedIdentities/read",
      "Microsoft.ManagedIdentity/userAssignedIdentities/assign/action",
      "Microsoft.DBforPostgreSQL/servers/read",
      "Microsoft.DBforPostgreSQL/servers/write",
```

```
      "Microsoft.DBforPostgreSQL/servers/delete",
      "Microsoft.DBforPostgreSQL/flexibleServers/read",
      "Microsoft.DBforPostgreSQL/flexibleServers/write",
      "Microsoft.DBforPostgreSQL/flexibleServers/delete",
      "Microsoft.DBforPostgreSQL/flexibleServers/start/action",
      "Microsoft.DBforPostgreSQL/flexibleServers/stop/action",
      "Microsoft.DBforPostgreSQL/flexibleServers/firewallRules/write",
      "Microsoft.DBforPostgreSQL/flexibleServers/configurations/write",
      "Microsoft.DBforPostgreSQL/flexibleServers/privateEndpointConnectio
nsApproval/action",
      "Microsoft.Network/privateDnsZones/read",
      "Microsoft.Network/privateEndpoints/read",
      "Microsoft.Network/privateEndpoints/write",
      "Microsoft.Network/privateEndpoints/delete",
      "Microsoft.Network/privateEndpoints/privateDnsZoneGroups/read",
      "Microsoft.Network/privateEndpoints/privateDnsZoneGroups/write",
      "Microsoft.DBforPostgreSQL/servers/privateEndpointConnectionsApproval/
action",
      "Microsoft.Network/privateDnsZones/A/read",
      "Microsoft.Network/privateDnsZones/A/write",
      "Microsoft.Network/privateDnsZones/A/delete",
      "Microsoft.Network/privateDnsZones/join/action",
      "Microsoft.Network/privateDnsZones/write",
      "Microsoft.Network/privateDnsZones/delete",
      "Microsoft.Network/privateDnsZones/virtualNetworkLinks/read",
      "Microsoft.Network/privateDnsZones/virtualNetworkLinks/write",
      "Microsoft.Network/privateDnsZones/virtualNetworkLinks/delete",

      "Microsoft.Network/virtualNetworks/join/action",
      "Microsoft.Network/loadBalancers/delete",
      "Microsoft.Network/loadBalancers/read",
      "Microsoft.Network/loadBalancers/write",
      "Microsoft.Network/loadBalancers/backendAddressPools/join/action",
      "Microsoft.Resources/deployments/cancel/action",
      "Microsoft.Resources/deployments/whatIf/action"
    ],
    "NotActions": [],
    "DataActions": [
      "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read"
,
      "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/wri
te",
      "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/dele
te",
      "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/add/
action"
    ],
    "NotDataActions": [],
    "AssignableScopes": [
      "/subscriptions/{SUBSCRIPTION-ID}/resourceGroups/{RESOURCE-GROUP-
NAME}"
    ]
  }
```

When creating the role definition, make sure to:

- Replace the {SUBSCRIPTION-ID} with your actual subscription ID.
- Replace the {RESOURCE-GROUP-NAME} with the name of your existing resource group

Role definition for Azure Marketplace images

In order to use the Azure Marketplace images, you need to grant the service principal additional Azure permissions on the scope of your Azure subscription. The following policy includes these permissions:

```
{
    "properties": {
        "roleName": "Cloudera Management Console Azure Operator for Azure
Marketplace",
        "description": "Can use Azure Marketplace images read and accept i
mage terms if the corresponding setting is enabled in Management Console ->
Global Settings -> Azure Settings-> Terms & Conditions auto acceptance. Scop
e must be subscription level.",
        "assignableScopes": [
            "/subscriptions/{SUBSCRIPTION-ID}"
        ],
        "permissions": [
            {
                "actions": [
                    "Microsoft.MarketplaceOrdering/offertypes/publishers/off
ers/plans/agreements/read",
                    "Microsoft.MarketplaceOrdering/offertypes/publishers/off
ers/plans/agreements/write"
                ],
                "notActions": [],
                "dataActions": [],
                "notDataActions": []
            }
        ]
    }
}
```

When creating the role definition, make sure to:

- Replace the {SUBSCRIPTION-ID} with your actual subscription ID.

Obtain subscription and tenant ID

Obtain subscription and tenant ID. You need them in order to create a provisioning credential for Azure.

These steps should be performed by someone who has the Owner built-in Azure role and the Application Developer role in Azure Active Directory.

- You can obtain both the Subscription ID and Tenant ID from Azure CLI by using the following Azure CLI command:

  ```
  az account list|jq '.[]|{"subscriptionId": .id, "tenantId": .tenantId, "
  state": .state}'
  ```

- You can obtain your Azure Subscription ID from your Azure Portal > Subscriptions:

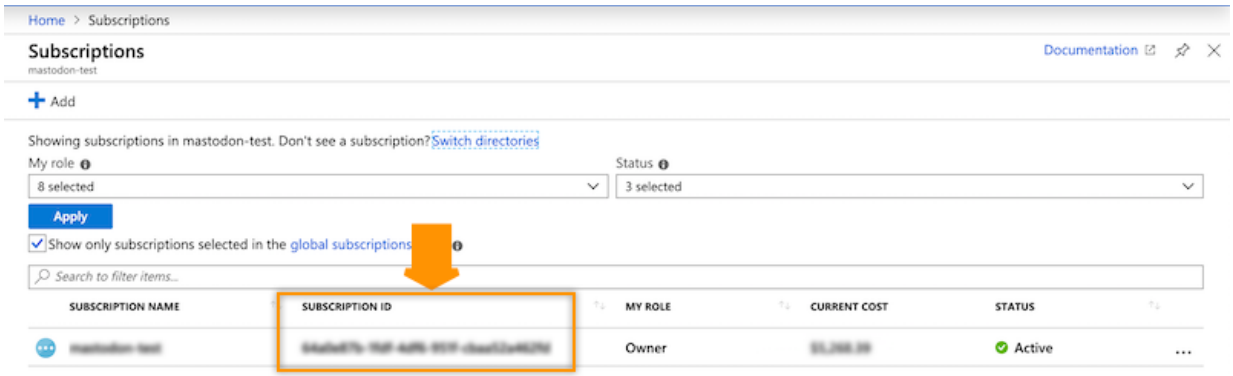

- You can obtain your Azure Tenant ID from your Azure Portal > Azure Active Directory > Properties:

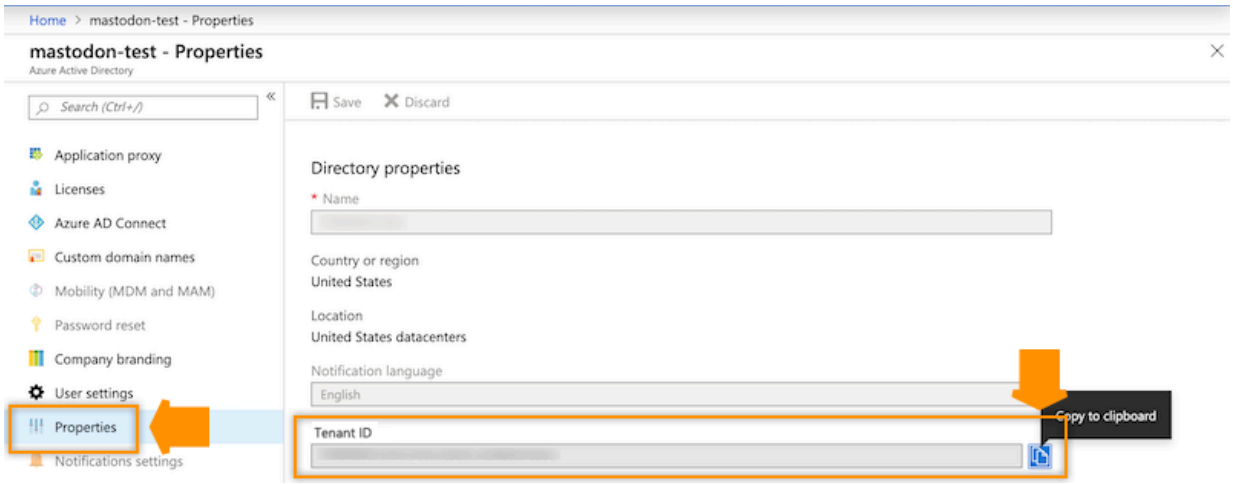

Create an app registration and assign a role to it

Create an app registration and assign a role to it. You need it in order to create a provisioning credential for Azure.

1. On Azure Portal, navigate to the Azure Active Directory > App Registrations and click on + New Registration:

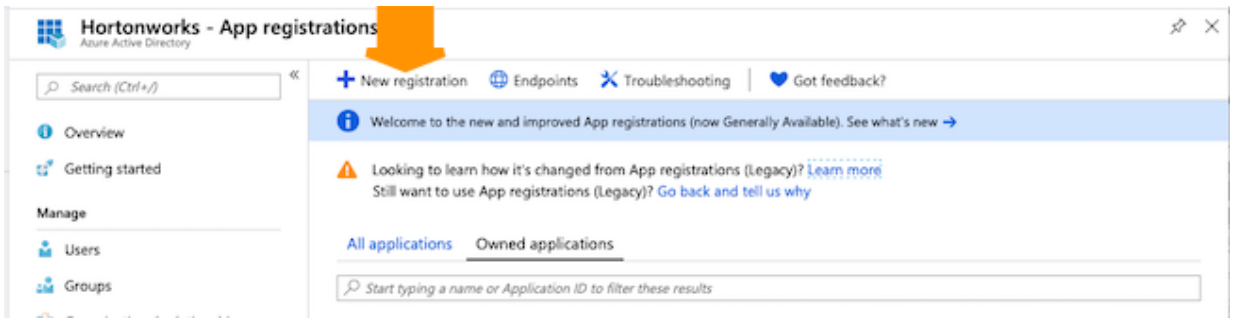

2. Register a new application as follows and then click Register:

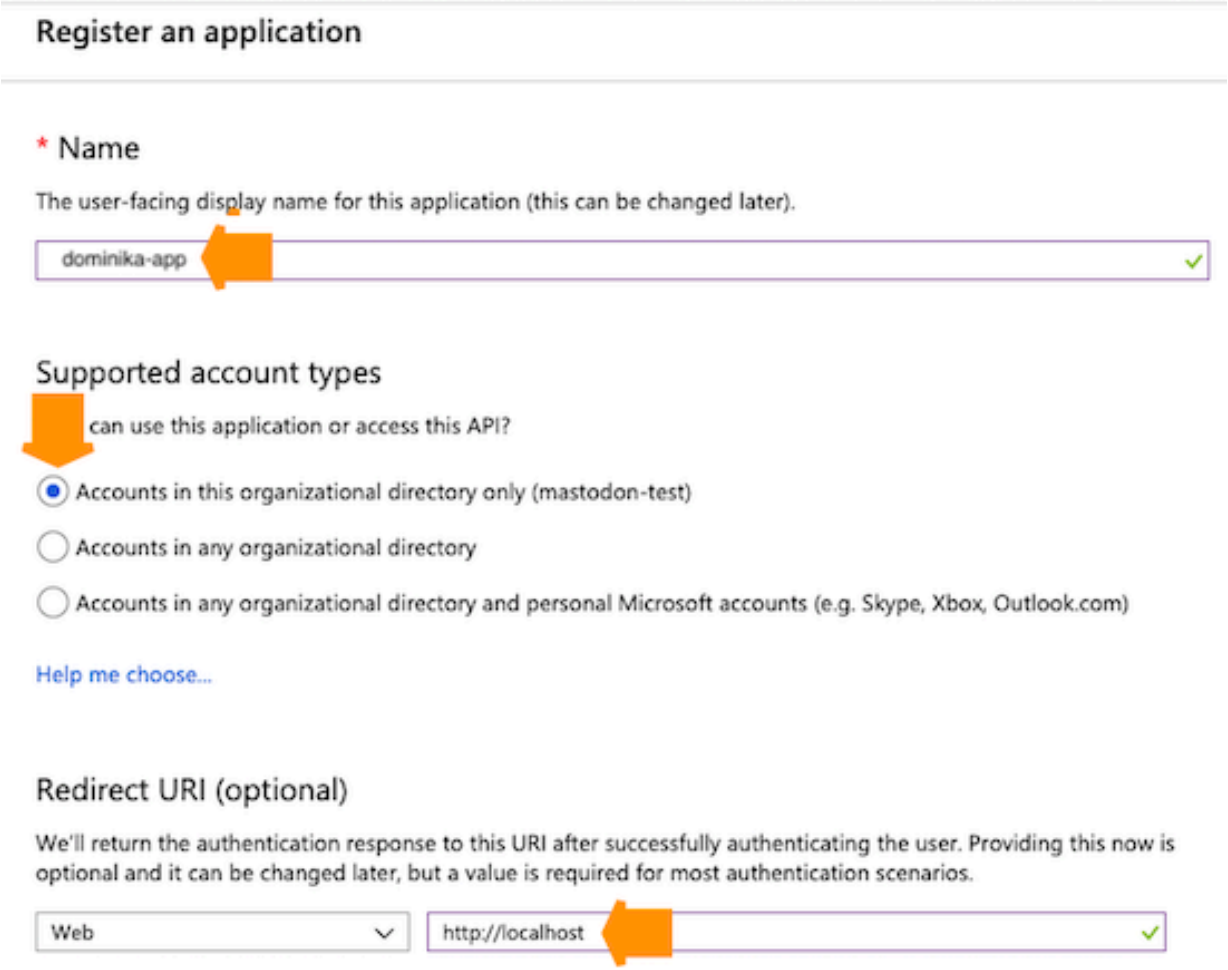

3. Once your app registration is created, you will be redirected to the app registration's overview page. Copy and save the Application ID before closing this page. You will need to provide it to Cloudera later:

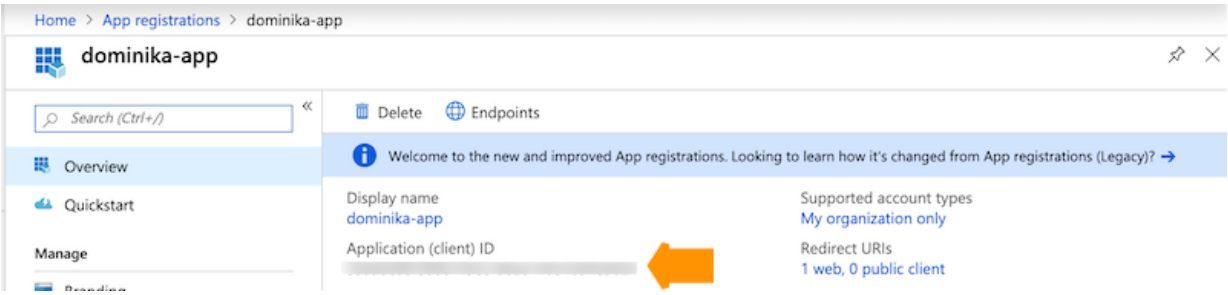

4. Next, navigate to Certificates & secrets and generate a new secret by clicking + New client secret, providing a description and expiration time, and clicking Add:

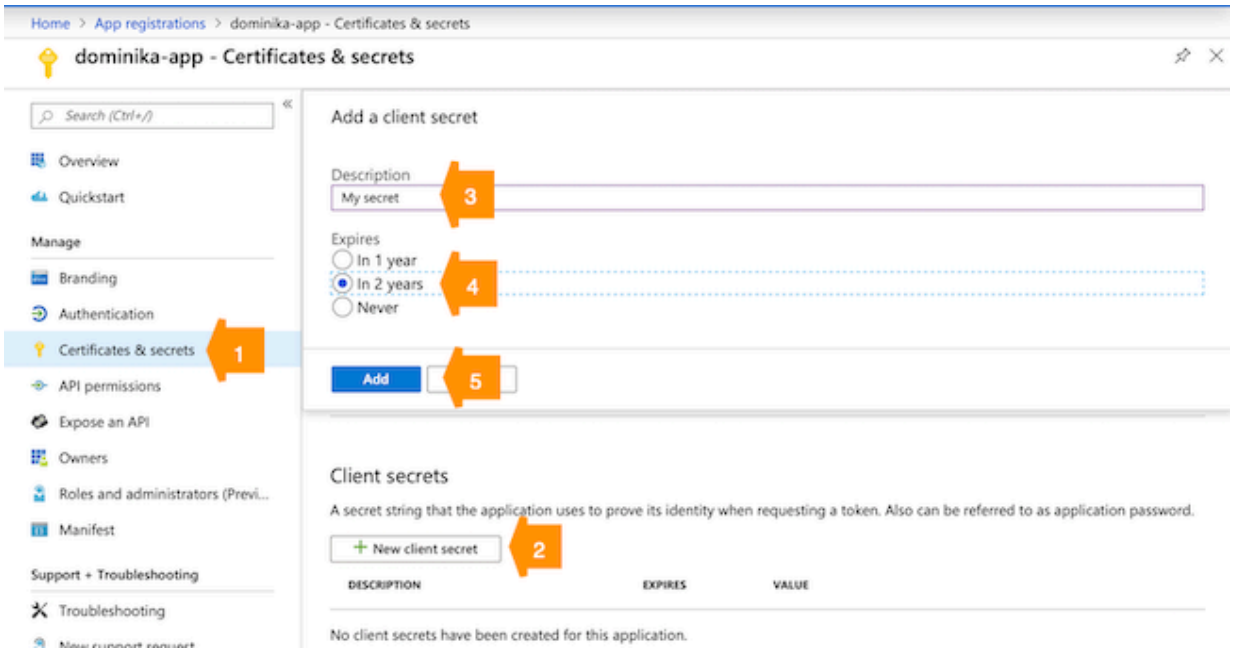

Note: The configured expiration time for the client secret is the default expiration time of the Cloudera credentials. As environments cannot be restarted with an expired secret, ensure that the client secret is renewed before the end of the expiration time. This can be accomplished by creating a new client secret using the Azure portal or Azure CLI. After creating the new client secret, make sure to update your Cloudera credential and set the new secret.

5. Copy and save the Client secret value. You will need to provide it to Cloudera later.
6. Next, you need to assign a role to your application. To do that, browse to Subscriptions, click on your subscription, and choose Access control (IAM).
7. Click Add > Add role assignment and then assign the Contributor role or the custom role to your newly created application by:
   - Under Role, selecting Contributor or the custom role.
   - Typing your app name under Select and then selecting it:

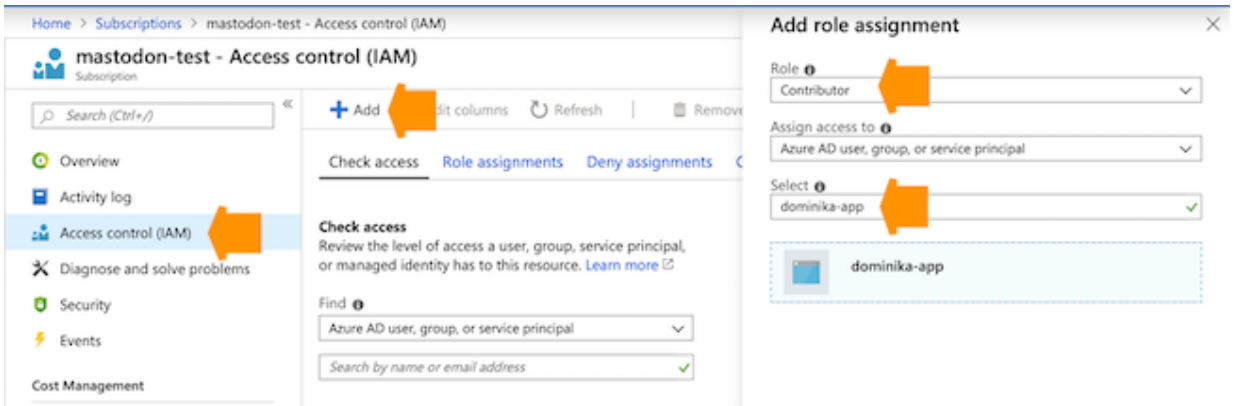

8. Once done, click Save.

What to do next

Once you have this setup ready, you can Create a provisioning credential for Azure in Cloudera.

Azure region

Prior to registering an environment, you should decide which Azure region you would like to use.

A single Azure environment registered in Cloudera corresponds to a single VNet located in a specific region, and all the resources deployed by Cloudera on Azure are deployed into that VNet.

Typically, to speed up data access, you may want to deploy clusters into the region containing the ADLS Gen2 containers that you want to access for input and output data. Therefore, when selecting the region to use, you should consider where your data is located.

Cloudera requires that the ADLS Gen2 storage location provided during environment registration is in the same region as the region selected for the environment.

If you need to use multiple regions, you need to register multiple environments, one per region.

Supported Azure regions

Cloudera supports the following Azure regions. The regions that are not mentioned below are not supported.

Note: Ensure that your selected region and HA option has compute availability for Flexible Server. See Flexible Server Azure Regions.

Note: Some Azure regions (such as Switzerland West, France South, Norway West, and so on) are disaster recovery regions and therefore cannot be supported by Cloudera. For more information, see Support matrix for Azure VM disaster recovery between Azure regions.

Region Name	Environment	Cloudera Data Hub	Cloudera Data Warehouse	Cloudera AI	Cloudera Data Engineering	Cloudera DataFlow	Cloudera Operational Database
Australia Central	##	##	##	##	##	##	##
Australia East	##	##	##	## (Partial GPU)	##	##	##
Australia Southeast	##	##	##	## (No ANF) (No GPU)	##	##	##
Brazil South	##	##	##	## (No ANF)	##	##	##
Canada Central	##	##	##	## (Partial GPU)	##	##	##
Canada East	##	##	##	## (No ANF) (No GPU)	##	##	##
Central India	##	##	##	## (No ANF) (Partial GPU)	##	##	##
Central US	##	##	## (AZ)	##	##	##	##
East Asia	##	##	##	## (No ANF)	##	##	##
East US	##	##	## (AZ)	##	##	##	##
East US 2	##	##	## (AZ)	##	##	##	##
Finland Central							
France Central	##	##	## (AZ)	## (No ANF)	##	##	##
Germany West Central (Public)	##	##	##	## (No ANF)	##	##	##
Japan East	##	##	## (AZ)	## (Partial GPU)	##	##	##
Japan West	##	##	##	## (No ANF) (No GPU)	##	##	##

Region Name	Environment	Cloudera Data Hub	Cloudera Data Warehouse	Cloudera AI	Cloudera Data Engineering	Cloudera DataFlow	Cloudera Operational Database
Korea Central	##	##	##	## (No ANF) (Partial GPU)	##	##	##
Korea South	##	##	##	## (No ANF) (No GPU)	##	##	##
North Central US	##	##	##	## (No ANF)	##	##	##
North Europe	##	##	## (AZ)	##	##	##	##
Norway East	##	##	##	## (No ANF)	##	##	##
Qatar Central	##	##		##			
South Africa North	##	##	##	## (No ANF)	##	##	##
South Central US	##	##	##	##	##	##	##
South India	##	##	##	## (No ANF) (No GPU)			##
Southeast Asia	##	##	## (AZ)	##	##	##	##
Sweden Central	##	##					
Switzerland North	##	##	##	## (No ANF)	##	##	##
UAE North	##	##	##	##	##	##	##
UK South	##	##	## (AZ)	## (Partial GPU)	##	##	##
UK West	##	##	##	## (No GPU)	##	##	##
US Gov Virginia							
West Central US	##	##			##	##	##
West Europe	##	##	## (AZ)	##	##	##	##
West India	##	##					##
West US	##	##	##	## (No ANF)	##	##	##
West US 2	##	##	## (AZ)	##	##	##	##

The regions marked with a check mark ("#")# are supported.

Note the following when reviewing regions supported for Data Warehouse:

- AZ means that a region supports availability zones.

Note the following when reviewing regions supported for Cloudera AI:

- Regions marked with a "#" and without any other annotation fulfill both ANF and GPU requirements.
- No ANF means that a region does not support Azure NetApp Files. Azure NetApp Files is required for the Cloudera AI service. If you select a region that does not include support for Azure NetApp Files, you must set up your own NFS service.
- Partial GPU means that a region supports VM types other than NCsv2. If you would like to utilize GPUs for faster computation, additional configuration will be necessary to use them.
- No GPU means that a region does not support VMs with GPUs.

Related Information

Azure geographies

Resource groups

Cloudera can provision all the environment and cluster resources into your existing resource group or you can also have Cloudera create multiple new resource groups.

Cloudera supports two resource group related scenarios:

Option	Description	Requirements	Permissions	Termination
Provide a single existing resource group	Provide a single existing resource group during environment creation and all Cloudera resources will be provisioned into that single resource group. No other resource groups will be created by Cloudera. You should select this option if you are planning to use Fine-grained access control.	If planning to use Cloudera Data Warehouse, do not use an underscore (_) when naming the resource group and use a short resource group name. The name must be fewer than 64 characters.	The scope of permissions in the role definition provided for Cloudera can be reduced to only the existing resource group where you would like Cloudera to create resources.	The resource group will not be deleted when your environment is terminated. The VHDs copied into your resource group during environment creation will not be deleted during environment termination. Cloudera preserves them to speed up subsequent environment deployments.
Cloudera creates multiple resource groups	Cloudera can create multiple resource groups. For the list of all the resource groups created, refer to Azure resources used by Cloudera. Do not use this option if you are planning to use fine-grained access control.	N/A	The scope of permissions in the role definition provided for Cloudera should be the whole subscription.	The resource groups will be deleted when your environment is terminated, except for the cloudbreak-images resource group (which stores VHDs for VM deployment).

If you would like to provide your own resource group, you can create it using the following instructions:

- Manage Azure Resource Manager resource groups by using the Azure portal
- Manage Azure Resource Manager resource groups by using the Azure CLI
- Manage Azure Resource Manager resource groups by using the Azure PowerShell

Related Information

Azure permissions

VNet and subnets

When registering an Azure environment in Cloudera, you will be asked to select a VNet and one or more subnets.

You have two options:

- Use your existing VNet and subnets for provisioning Cloudera resources.
- Have Cloudera create a new VNet and subnets. All Cloudera resources will be provisioned into this new VNet and subnets.

Note:

If you want to create an environment with cloud storage where access is allowed only from selected networks, then you must provide your own existing network. Furthermore, ensure that the network service endpoint for storage is placed on all the subnets and that the endpoint is connected to the storage account that you want to use.

Existing VNet and subnets

If you would like to use your own VNet, it needs to fulfill the following requirements:

- The VNet has at least one subnet. If you are planning to run more than Cloudera Data Hub, you need additional subnets as specified in VNet and subnet planning.
- VNet should be able to make an outbound connection with the internet or set of CIDRs and ports provided by Cloudera.
- If you would like to use private Flexible Server with delegated subnet as your Azure database server type, then you should delegate a subnet to it, as described in Private setup for Azure Flexible Server. When using Private Link, a delegated subnet is not required.
- If you would like to deploy Cloudera Data Warehouse in your environment, make sure Azure VNet subnets are large enough to support the Cloudera Data Warehouse load. When an Azure environment is activated for Cloudera Data Warehouse service, an Azure Kubernetes Service (AKS) cluster is provisioned in your subscription. The AKS cluster uses the Azure Container Networking Interface (CNI) plug-in for Kubernetes. This plug-in assigns IP addresses for every pod running inside the Kubernetes cluster. By default, the maximum number of pods per node is 30. This means that you need approximately 3,200 IP addresses for a 99-node cluster. if you activate an environment for Cloudera Data Warehouse service, make sure that the subnets are large enough on the Azure VNet for the Cloudera Data Warehouse load. Cloudera recommends using a CIDR/20 subnet or larger.
- Configure service endpoints as described in Set up service endpoint for network. If you would like to use private endpoints instead of service endpoints for Data Lake and Cloudera Data Hub, meet the requirements described in Private endpoint for PostgreSQL; However, if you would like to use the Cloudera Data Warehouse service, you still need service endpoints regardless of what you choose for Data Lake and Cloudera Data Hub.
- If you would like to deploy Cloudera AI, note that each Cloudera AI workbench requires its own subnet.
- If you would like to deploy Cloudera Data Engineering, note that each Cloudera Data Engineering Service requires its own subnet.
- If you would like to deploy Cloudera DataFlow, note that the Cloudera DataFlow service requires its own subnet. You can have only one Cloudera DataFlow service per environment.

Warning:

Be careful if deploying into an existing subnet that has an existing Network Security Group (NSG) applied to it. In such case, there is an NSG present at VM (NIC) level and subnet level at the same time. This can cause a problem because the two settings are evaluated independently (For incoming traffic, the NSG set at the subnet level is evaluated first, then the NSG set at the VM (NIC) level. For outgoing traffic the evaluation is the reverse), and if an "allow" rule does not exist at both levels, the traffic will not be admitted. Therefore, if you need to use such a setup, you should ensure that the "allow" rule exists on both levels.

Verify the limits of the VNet and subnets available in your Azure subscription to ensure that you have enough resources to create clusters in CDP.

VNets can be created and managed from the Azure Portal > Virtual Networks. For detailed instructions on how to create a new VNet on Azure, refer to Create a virtual network using the Azure portal in Azure documentation.

Egress connectivity for existing VNets and subnets

When you deploy an environment with an existing network of your own configuration, it is your responsibility to create egress connectivity for the required subnets in your VNet. Egress connectivity can be accomplished through a NAT gateway setup or user-defined routing. Alternatively you can create a secondary load balancer for public egress. See Azure Load Balancers in Data Lake and Data Hub for more information.

New VNet and subnets

If you would like Cloudera to create a new VNet, you will need to specify a valid CIDR in IPv4 range that will be used to define the range of private IPs for VM instances provisioned into these subnets. Default is 10.10.0.0/16. Consider changing the IP range to correspond to corporate policies for standardized IP address ranges. The CIDR must match the <network mask>/16 pattern.

By default Cloudera creates more than 30 subnets and divides the address space as follows:

- 3 x /24 public subnets for Data Lake and Cloudera Data Hub
- 3 x /19 private subnets for Cloudera Data Warehouse

- 32 x /24 private subnets for Cloudera AI, Cloudera Data Engineering, and Cloudera DataFlow
- 3 x /19 private subnets reserved for future use

You can disable creating private subnets, in which case only 3 public subnets will be created.

If you would like to use private Flexible Server with delegated subnet as your Azure database server type in private service mode, then you should delegate a subnet to it, as described in Private setup for Azure Flexible Server. Cloudera does not create the delegated subnet for you. When using Private Link, a delegated subnet is not required.

For more information about VNet and subnets, refer to the following VNet and subnet planning documentation.

Egress connectivity for new VNets and subnets

If you are creating a new network during environment registration, Cloudera ensures that egress connectivity is available. If the "Create Public IPs" option and Public Endpoint Access Gateway are disabled in your network, a separate load balancer is created for egress, though this load balancer requires certain public IP permissions that are granted as part of the required Azure permissions. If either "Create Public IPs" or Public Endpoint Access Gateway is enabled, then a public load balancer is created to handle both public ingress to port 443 and public egress.

Azure Load Balancers in Data Lake and Cloudera Data Hub for more information.

VNet and subnet planning

Whether you decide to use your own VNet for Cloudera or have Cloudera create one for you, you should carefully plan your network, calculating and verifying the limits of the VNet and subnets available in your Azure subscription to ensure that you have enough networking resources to create clusters in Cloudera.

When registering an Azure environment in Cloudera, you are asked to select a VNet and one or more subnets. You have two options:

- Cloudera will create a new VNet and subnets
- Select a VNet and subnets that you previously created

In both cases, use this guide to calculate and verify the limits of the VNet and subnets available in your Azure subscription to ensure that you have enough networking resources to create clusters in Cloudera.

Option 1: Cloudera creates the VNet and subnets

If you would like Cloudera to create a new VNet, you will need to specify a valid CIDR in IPv4 range that will be used to define the range of private IPs for VM instances provisioned into these subnets. This must be a /16 CIDR, but you can customize the IP Range. The default is 10.10.0.0/16.

You cannot use the following reserved CIDR blocks for your VNet:

- 10.0.0.0/16
- 10.244.0.0/16
- 172.17.0.1/16
- 10.20.0.0/16
- 10.244.0.0/16

Cloudera will divide this address range as follows:

- 32 x /24 subnets - Recommended for Cloudera AI workbenches, Cloudera Data Engineering Services, and Cloudera DataFlow Services
- 3 x /19 subnets - Recommended for Cloudera Data Warehouse service
- 3 x /19 subnets - Recommended for Data Lake and Cloudera Data Hub
- 3 x /24 subnets - Reserved for future use

Note:

Not all subnets are used upon creation. Many of the above start getting used once you enable certain features and create more workloads in Cloudera.

If you would like to have a minimal virtual network instead, you can use the guide outlined in the next option.

Option 2: Existing VNet and subnets

If you would like to use an existing VNet, the subnet requirements vary based on the services used. Below is a guide for calculating network requirements per service.

In addition, make sure you follow the following guidelines:

- You cannot use the following reserved CIDR blocks for your VNet:
  - 10.0.0.0/16
  - 10.244.0.0/16
  - 172.17.0.1/16
  - 10.20.0.0/16
  - 10.244.0.0/16
- The Microsoft.Storage and Microsoft.SQL Service endpoints should be registered for all subnets that will be used by Cloudera.

Subnets for Data Lake and Cloudera Data Hub

Both Data Lake and Cloudera Data Hub share the same subnet, so only one subnet is required.

Cloudera recommends a minimum of a /24 CIDR. If you would like to use a smaller subnet, use the following guidelines:

- One IP address is used for each VM
- One Light Duty Data Lake cluster uses three VMs
- A typical Cloudera Data Hub cluster uses a minimum of four VMs as a starting point, but this number can be dynamically scaled up or down
- Make sure you allocate enough IPs to handle each cluster running at peak capacity

Subnets for Cloudera Data Warehouse

The Cloudera Data Warehouse service needs one subnet. You can choose the specific subnet used by Cloudera Data Warehouse when you activate Cloudera Data Warehouse for an environment. This subnet should not be shared with any of the other Cloudera applications.

Cloudera recommends a /20 or larger subnet as it can be difficult to accurately predict the size of each VW due to autoscaling.

If you would like to size the subnets to a smaller CIDR, the following guidelines assume that you are activating your Cloudera Data Warehouse environment with the default settings (no overlay networks):

VM Purpose	# VMs	# pods per VM	IPs per VM (1 for the instance + 1 per pod)	Total IP addresses required
Cloudera Data Warehouse Shared Services - (Shared among all VWs in an environment)	3	30	31	93
Per Database Catalog (One catalog is created by default, you can create additional catalogs)	2	30	31	62
Per Virtual Warehouse (XS) - without autoscaling*	2	10	11	22
Per Virtual Warehouse (S) - without autoscaling*	10	10	11	110
Per Virtual Warehouse (M) - without autoscaling*	20	10	11	220

Per Virtual Warehouse (L) - without autoscaling*	40	10	11	440

If you would like to use a smaller subnet, you can enable the Overlay Networks feature for Cloudera Data Warehouse, which will provision your AKS cluster with the kubenet network plugin instead of Azure CNI. When you use kubenet, you may need to manage the Azure Route Table (UDR) manually in accordance with AKS documentation. In this case, use the following table as your guideline:

VM Purpose	# VMs	Total IP addresses required
Cloudera Data Warehouse Shared Services (shared among all VWs in an environment)	3	3
Per Database Catalog (One catalog is created by default, you can create additional catalogs)	2	2
Per Virtual Warehouse (XS) - without autoscaling*	2	2
Per Virtual Warehouse (S) - without autoscaling*	10	10
Per Virtual Warehouse (M) - without autoscaling*	20	20
Virtual Warehouse (L) - without autoscaling*	40	40

* Each autoscaling activity can be treated as deploying a new Virtual Warehouse. For example, when a XS Virtual Warehouse is scaled once, it uses four VMs instead of two.

Subnets for Cloudera AI

Azure Files NFS v4.1 is a managed, POSIX compliant NFS service on Azure. The file share is used to store files for the Cloudera AI infrastructure and Cloudera AI workbenches. This is the recommended NFS service for use with Cloudera AI. You need one separate subnet delegated to the Azure Files NFS service (all workbenches in a region will share this service). Cloudera recommends a /28 subnet for this purpose.

You also need one subnet per each workbench that you plan to run. Cloudera AI uses the Kubenet plugin to AKS and the subnet used for a workbench cannot be shared with another workbench or another service. You can choose which subnet is used by a workbench at the time of provisioning. Cloudera recommends a /25 CIDR for these subnet, but if you would like to provide a custom range, the formula to calculate IP Addresses per workbench is as follows:

- Each workbench can grow up to 30 CPU worker nodes and 30 GPU workers; each node consumes one IP address.
- In addition, you need to allocate up to 11 IP address (6 infrastructure nodes and 5 for auxiliary networking usage).

For more information, see Network Planning for Cloudera AI on Azure.

Subnets for Cloudera Data Engineering

Cloudera Data Engineering runs in the VNet registered in Cloudera as part of your Azure environment.

Each Cloudera Data Engineering service requires its own subnet. CDE on AKS uses the Kubenet CNI plugin provided by Azure. In order to use Kubenet CNI, we need to create multiple smaller subnets when creating an Azure environment. It is recommended to partition the vnet with subnets that is just the right size to fit the expected max nodes in the cluster.

Cloudera recommends a /24 CIDR for these subnets, but if you would like to provide a custom range, the formula to calculate IP Addresses per Cloudera Data Engineering service is as follows:

- Each Cloudera Data Engineering service can scale up to 100 compute nodes; each node consumes one IP address.
- In addition, you need to allocate 3 IPs for the base infra nodes and 2 IP addresses per virtual cluster for the virtual cluster service nodes.

Subnets for Cloudera DataFlow

Cloudera DataFlow runs in the VNet registered in Cloudera as part of your Azure environment.

The Cloudera DataFlow service requires its own subnet. Cloudera DataFlow on AKS uses the Kubenet CNI plugin provided by Azure. In order to use Kubenet CNI, create multiple smaller subnets when creating an Azure environment.

Cloudera recommends a /24 CIDR for these subnets, but if you would like to provide a custom range, the formula to calculate the IP Addresses is as follows:

- Each Cloudera DataFlow service can scale up to 50 compute nodes; each node consumes one IP address.
- In addition, allocate two IPs for the base infra nodes.

Private setup for Azure Flexible Server

When Cloudera creates an Azure Database for PostgreSQL - Flexible Server instance, you must choose one of the following networking options: Private access (VNet integration or Private Link) or Public access (allowed IP addresses). Public access is used by default.

For more general information, see Networking overview for Azure Database for PostgreSQL - Flexible Server with private access (VNET Integration) and Networking overview for Azure Database for PostgreSQL - Flexible Server with public access (allowed IP addresses).

Delegated subnet

You have two options to deploy Flexible Server instances in private service mode (without public endpoints): using Private Link or without Private Link. Flexible Server with Private Link is the recommended option. If Private Link is not used, Flexible Server instances need to be deployed in a “delegated subnet” within the VNet. The Flexible Server with delegated subnet option, although still available on the UI, is deprecated and it is recommended to use Flexible Server with Private Link instead.

Delegated subnets are no longer a requirement as you can use Azure Private Link instead. However, when private endpoints are not used, it is a prerequisite to delegate a subnet to Microsoft.DBforPostgreSQL/flexibleServers as mentioned in Azure documentation. This delegation means that only Azure Database for PostgreSQL Flexible Servers can use that subnet. No other Azure resource types can be in the delegated subnet.

You need to create such a delegated subnet and provide it to Cloudera during environment registration. This delegated subnet will be used by Azure Database for PostgreSQL instances. The delegated subnet provided during environment registration will be used by default for all Azure Database for PostgreSQL instances used in Cloudera.

Note: Although you can currently select multiple subnets, the larger subnet is always used by Cloudera. That is, if there are two subnets provided, one with 128 available IPs and another with 256 available IPs, the second one will be picked. Even though you can select multiple delegated subnets, we recommend that you provide only one.

Creating a delegated subnet

For a step-by-step official guide on how to perform the delegation, see Delegate a subnet to an Azure service. For considerations on the delegated subnet’s sizing, see Virtual network concepts.

Here is a screenshot from Azure Portal showing the desired setting:

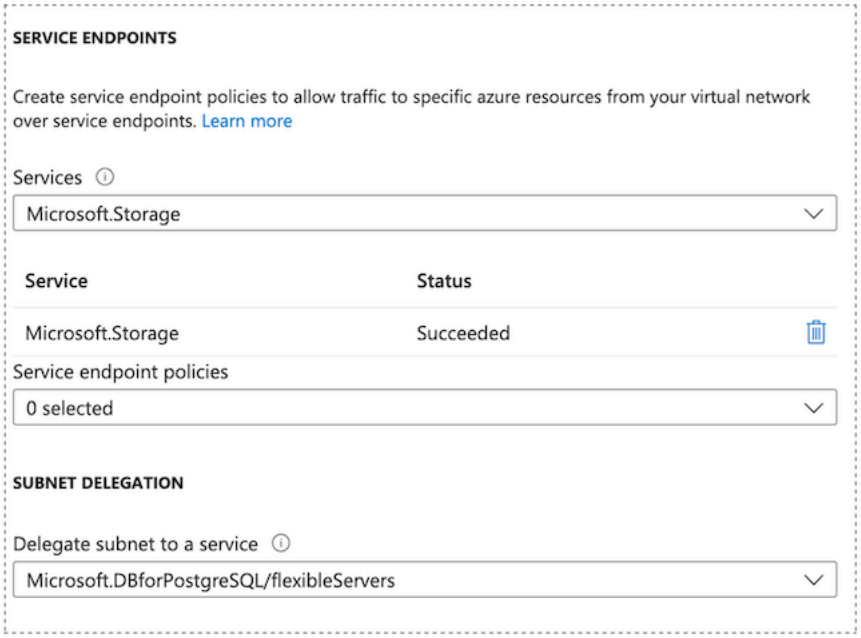

The Microsoft.Storage service endpoint is set automatically during deployment by Azure.

After the subnet has successfully been delegated, don't forget to record the full subnet ID or the name of the subnet for later use as an input (referred to as <delegated-subnet-id>). For example: /subscriptions/3ddda1c7-d1f5-4e7b-ac81-abcdefgh/resourceGroups/rg/providers/Microsoft.Network/virtualNetworks/vnet/subnets/subnet

For background information, see Using Azure Database for PostgreSQL Flexible Server.

Private DNS options

Cloudera supports using VNet integration based private setup with an existing Private DNS zone that can be either pre-created and provided by you, or created by Cloudera.

When using a private setup for Azure Postgres, an Azure private DNS zone is used for the DNS service resolving the FQDN to the private IP. Cloudera offers two options. The DNS zone can be:

- Created and managed entirely by Cloudera (You select "Create new private DNS zone" during environment registration), or
- Created by you before registering an environment (You pre-create the DNS zone and select it during environment registration). You need to provide access for discovery, validation, and adding and removing DNS A records.

Private setup (with either of the two DNS options) can only be used with a single customer-provided resource group. They cannot be used with Cloudera-created multiple resource groups.

Depending on whether you prefer to bring your own DNS or have Cloudera create and manage it, refer to the following documentation:

Using Cloudera-managed private DNS

Review this documentation if you are planning to use a private setup for Azure Postgres with a Cloudera-managed DNS.

Requirements and limitations

The following limitations apply when using a Cloudera-managed private DNS:

- Only Azure's Private DNS Zone is supported. Using an on-premise DNS is not supported.
- The Private DNS Zone will be residing in the single existing resource group, even if the VNet is located elsewhere.

- Only one resource group can have private setup with a given VNet. This is because:
  - Only one DNS zone with a given name can be linked to a VNet.
  - That DNS zone is deployed in the single resource group where all the resources are located.
- The private DNS zone and virtual network links are shared within the single resource group. The first environment ever created in that resource group will create them. They will never be deleted by Cloudera.

Prerequisites

In order to use a Cloudera-managed private DNS, you should meet the following prerequisites:

1. Review DNS zones existing in your resource group on page 29
2. Ensure that Cloudera has adequate permissions on page 29

Review DNS zones existing in your resource group

If you would like Cloudera to create and manage the Private DNS Zone, review the DNS zones that exist in the resource group that you are planning to use for Cloudera and make sure that one of the following is true:

- No Private DNS Zone named privatelink.postgres.database.azure.com or flexible.postgres.database.azure.com (for Flexible Server with Private Link or Flexible Server with delegated subnet respectively) is connected to the VNnet.
- If there is a Private DNS Zone named privatelink.postgres.database.azure.com or flexible.postgres.database.azure .com (for Flexible Server with Private Link or Flexible Server with delegated subnet respectively) connected to the VNet, verify that the zone is located in the existing resource group that you are planning to use for Cloudera. If the Private DNS Zone is already used for one environment, Cloudera can reuse it for another environment.

Ensure that Cloudera has adequate permissions

Ensure that the role that you are using for the Azure credential has the permissions mentioned in Role definition 2: Allows Cloudera to use only a single existing resource group create private endpoints.

Bringing your own private DNS

Review this documentation if you are planning to use a private setup for Azure Postgres with your own private DNS.

Requirements and limitations

The following limitations apply when using your own private DNS:

- Only Azure's private DNS zone is supported. Using an on-prem DNS is not supported.
- The private DNS zone provided to Cloudera must have the name privatelink.postgres.database.azure.com in case of Flexible Server with Private Link and the name of the private DNS zone must ending with postgres.database.az ure.com in case of Flexible Server with Delegated Subnet.
- The private DNS zone must be in a subscription that is accessible to the service principal used by the Cloudera app-based credential; that is, the subscription where the private DNS zone is created must be in the same tenant where the service principal is located.

Prerequisites

When bringing your own private DNS, you should meet the following prerequisites:

Create a private DNS zone and link it to your VNet

If you choose to provide your own Azure private DNS zone then you should:

1. Create a private DNS zone with a name ending with the name privatelink.postgres.database.azure.com in case of Flexible Server with Private Link or with a name ending with postgres.database.azure.com in case of Flexible Server with delegated subnet in any subscription accessible to the service principal used by the Cloudera app-

based credential; that is, the subscription where the private DNS zone is created must be in the same tenant where the service principal is located.

- In case of Flexible Server with Private Link, use the following Azure CLI command to create a private DNS zone:

```
az network private-dns zone create \
 --name privatelink.postgres.database.azure.com \
 --resource-group <YOUR_DNS_RESOURCE_GROUP> \
 --subscription <YOUR_SUBSCRIPTION_FOR_DNS>
```

- In case of Flexible Server with delegated subnet, use the following Azure CLI command to create a private DNS zone:

```
az network private-dns zone create \
 --name flexible.postgres.database.azure.com \
 --resource-group <YOUR_DNS_RESOURCE_GROUP> \
 --subscription <YOUR_SUBSCRIPTION_FOR_DNS>
```

2. Link it to the VNet that you are planning to use for the Cloudera environment. You can do this using the following Azure CLI command:

```
az network private-dns link vnet create \
 --name <DESIRED_LINK_NAME> \
 --resource-group <YOUR_VNET_RESOURCE_GROUP> \
 --zone-name <PRIVATE_DNS_ZONE_NAME> \
 --virtual-network <YOUR_VNET_RESOURCE_ID> \
 --subscription <YOUR_VNET_SUBSCRIPTION>
```

Ensure that Cloudera has adequate permissions

Ensure that the role that you are using for the Azure credential has the permissions mentioned in Role definition 2: Allows Cloudera to use only a single existing resource group and create private endpoints.

Additionally, if your DNS zone is outside of the single resource group that you are planning to provide to Cloudera, the following additional permissions are required for the service principal to be able to discover, validate, and use the DNS zone (add and remove A records). These additional permissions need to be included in a separate role definition and assigned separately:

```
{
 "Name": "Cloudera Management Console Azure Operator for Using Private DNS
 Zones",
 "IsCustom": true,
 "Description": "Can list, validate and use private DNS zones",
 "Actions": [
 "Microsoft.Network/privateDnsZones/join/action",
 "Microsoft.Network/privateDnsZones/read",
 "Microsoft.Network/privateDnsZones/virtualNetworkLinks/read"
 ],
 "NotActions": [],
 "DataActions": [
 ],
 "NotDataActions": [],
 "AssignableScopes": [
 "/subscriptions/{SUBSCRIPTION-ID}/resourcegroups/{RESOURCE-GROUP-NAME}"
 ]
 }
```

The placeholders in bold must be replaced with actual values:

- {SUBSCRIPTION-ID} - The ID of the subscription where the DNS zone is located.
- {RESOURCE-GROUP-NAME} - The name of the resource group where the DNS zone is located.

Related Information

Quickstart: Create an Azure private DNS zone using the Azure portal

Resources created under the hood

Creating a Cloudera environment on Azure with a private Flexible Server setup created by Cloudera involves creating several Azure resources, all of which are necessary to have a private setup that works out-of-the-box.

This private setup ensures that communication between Cloudera and the Azure Postgres server happens via a private IP address. However, cluster services do need to contact the Postgres server via FQDN, so the address needs to be resolvable from the VNet.

Resources created when using a Cloudera-managed Private DNS Zone

In the scenario where Cloudera creates the private DNS, several resources are created:

- A Private DNS Zone: It is a DNS zone, part of an Azure-hosted DNS server. It has a fixed name, privatelink.postgres.database.azure.com for Flexible Server with Private Link and flexible.postgres.database.azure.com for Flexible Server with delegated subnet. However, any DNS Zone name ending with postgres.database.azure.com is usable for Flexible Server with delegated subnet in the “Bring your own private DNS” setup.
- A virtual network link between the zone and a VNet where the domain resolution should happen.
- An A record within the zone: FQDN to IP address resolution. There is no reverse lookup.

Note: If there is a private DNS zone named privatelink.postgres.database.azure.com or flexible.postgres.database.azure.com (for Flexible Server with Private Link or Flexible Server with delegated subnet respectively) containing the required virtual network link in the deployment’s resource group, Cloudera reuses it. The A record and the network link are created only if they do not already exist.

When you delete the environment, the DNS zone and the network link that you provided will not be deleted.

Resources created when using your own private DNS zone

In this case the Private DNS Zone and the network link are provided by you and Cloudera creates the following:

- Cloudera creates an A record within the zone: FQDN to IP address resolution. There is no reverse lookup.

When you delete the environment, the DNS zone and the network link that you provided will not be deleted.

Private setup for Azure Single Server

By default, Cloudera creates and uses Azure Database for PostgreSQL - Flexible Server instance and so it requires the private setup described in Private setup for Azure Flexible Server. You should only review the documentation linked below if you specifically would like to use Single Server instead of Flexible Server; Otherwise, you can ignore this content.

When Cloudera creates an Azure Database for PostgreSQL - Single Server instance, you must choose one of the following networking options: service endpoint or private endpoint.

Service endpoint for Azure Postgres

On Azure, the external PostgreSQL database can reach the network via a service endpoint or a private endpoint. If you would like to use a service endpoint, before you can set up an external database for a given network, you must enable service endpoints for all subnets where the database should be reachable.

Note: Service endpoints are firewall rules that allow traffic only from those subnets where you explicitly granted permission. Therefore, when using service endpoints for Azure Postgres, the external database still needs to have public access enabled.

Set up a service endpoint via Azure Portal

1. From the Azure portal, go to the VNet for which you want to add service endpoints.

2. From the menu in the left pane, select Service endpoints.
3. In the Service endpoints window, click the + Add button.
4. In the pop up window, select:
   - Service:
     - Microsoft.Sql
     - Microsoft Storage (optional: only needed if you are using Cloudera Data Warehouse)
   - Subnets: Select all subnets for which you want to apply the service endpoint
5. Click Add.

Set up a service endpoint via ARM template

If you are using ARM templates to create your infrastructure, you can add the serviceEndpoints section to your template:

For example:

```
"subnets": [
        {
            "name": "<YOUR-SUBNET-NAME>",
            "properties": {
                "addressPrefix": "<YOUR-SUBNET-PREFIX>",
                "serviceEndpoints": [
                    {
                        "service": "Microsoft.Sql"
                    },
                    {
                        "service": "Microsoft.Storage"
                    }
                ]
            }
        }
    ]
```

Related Information

Private endpoint for Azure Postgres

Private endpoint for Azure Postgres

By default Cloudera uses service endpoints, but you can select to use private endpoints instead. During environment registration you can optionally select the “Create Private Endpoint” option to use private endpoints instead of using a service endpoint. Currently, only one service or private endpoint is used, for Azure Postgres.

Azure Postgres service can be reached via the following two methods, both of which are designed to allow customers to restrict who connects to the Azure Postgres service:

- Service endpoints (Created by default)
  - When a service endpoint is used, a database server must have a public IP address. This means the traffic is leaving their virtual network.
  - A service endpoint helps with the possibility of creating a firewall filter to allow connections only from the subnet explicitly linked to a given Postgres instance.
- Private endpoints (Created instead of a service endpoint only if explicitly enabled)
  - A private endpoint makes it possible to connect to an Azure Postgres server instance over a private IP address from the VNet, always ensuring traffic stays within your VNet hence increasing security.
  - A private endpoint consists of a network interface using a private IP in the VNet, and a DNS service that will resolve the FQDN of a server to the private IP address.

For more general information, see Use Virtual Network service endpoints and rules for Azure Database for PostgreSQL - Single Server and Private Link for Azure Database for PostgreSQL-Single server.

Private DNS options

Cloudera supports using private endpoints with an existing private DNS zone that can be either pre-created and provided by you, or created by Cloudera.

When using a private endpoint for Azure Postgres, an Azure private DNS zone is used for the DNS service resolving the FQDN to the private IP. Cloudera offers two options. The DNS zone can be:

- Created and managed entirely by Cloudera (you select “Create new private DNS zone” during environment registration), or
- Created by you before registering an environment (you pre-create the DNS zone and select it during environment registration). You need to provide access for discovery, validation, and adding and removing DNS A records.

Private endpoints (with either of the two DNS options) can only be used with a single customer-provided resource group. They cannot be used with Cloudera-created multiple resource groups.

You can only use private endpoints with your own private DNS zone if you specify an existing VNet for the environment. If you choose that Cloudera creates a new VNet and enables private endpoints, Cloudera is going to manage the private DNS zone for you.

Depending on whether you prefer to bring your own DNS or have Cloudera create and manage it, refer to the following documentation:

Using Cloudera-managed private DNS

Review this documentation if you are planning to use a private endpoint for Azure Postgres with a Cloudera-managed DNS.

Requirements and limitations

The following limitations apply when using a Cloudera-managed private DNS:

- Only Azure’s private DNS zone is supported. Using an on-premise DNS is not supported.
- Private endpoints can only be used with a single customer-provided resource group. They cannot be used with Cloudera-created multiple resource groups. When creating an environment from the UI, the “Create Private Endpoints” option is disabled if you select the option for Cloudera to create multiple resource groups. If using CDP CLI, a validation error appears if this option is used with Cloudera creating multiple resource groups.
- The private DNS zone must be in the single existing resource group, even if the VNet is located elsewhere. This is because:
  - Microsoft requires that the DNS zone has a fixed name.
  - A VNet cannot have links to two different DNS zones with the same name.
  - It is not possible to check if a VNet already has a DNS zone with a given name attached to it.
- Only one resource group can have private endpoints with a given VNet. This is because:
  - Only one DNS zone with a given name can be linked to a VNet.
  - That DNS zone must be in the single resource group where all the resources are located.
- The private DNS zone and virtual network links are shared within the single resource group. The first environment ever created in that resource group will create them. They will never be deleted by Cloudera.
- The private DNS zone creation cannot be turned off. There is no option to create private endpoints but not create a private DNS zone. You can, however, choose to specify your own existing private DNS zone. See Bringing your own private DNS.

Prerequisites

In order to use a Cloudera-managed private DNS, you should meet the following prerequisites:

1. Disable private endpoint network policies on page 33
2. Review DNS zones existing in your resource group on page 34
3. Ensure that Cloudera has adequate permissions on page 34

Disable private endpoint network policies

Only subnets that have private endpoint network policies turned off are eligible for private endpoint creation, because network security groups (NSG) are not supported for private endpoints.

You can use the following Azure CLI command to disable private endpoint network policies for a certain subnet:

```
az network vnet subnet update \
  --name <subnet-name> \
  --resource-group <resource-group-name> \
  --vnet-name <vnet-name> \
  --disable-private-endpoint-network-policies true
```

For example:

```
az network vnet subnet update \
  --name default-2 \
  --resource-group my-cdp-rg \
  --vnet-name my-azure-vnet \
  --disable-private-endpoint-network-policies true
```

Review DNS zones existing in your resource group

If you want Cloudera to create and manage the private DNS zone, review the DNS zones that exist in the resource group that you are planning to use for Cloudera and make sure that one of the following is true:

- No private DNS zone named “privatelink.postgres.database.azure.com“ is connected to the VNnet.
- If there is a private DNS zone named “privatelink.postgres.database.azure.com“ connected to the VNet, verify that the zone is located in the existing resource group that you are planning to use for Cloudera. If the private DNS zone is already used for one environment, Cloudera can reuse it for another environment.

Ensure that Cloudera has adequate permissions

Ensure that the role that you are using for the Azure credential has the permissions mentioned in Role definition 2: Allows Cloudera to use only a single existing resource group and create private endpoints.

Related Information

Manage network policies for private endpoints

Bringing your own private DNS

Review this documentation if you are planning to use a private endpoint for Azure Postgres with your own private DNS.

Requirements and limitations

The following limitations apply when using your own private DNS:

- Only Azure’s private DNS zone is supported. Using an on-prem DNS is not supported.
- Private endpoints can only be used with a single customer-provided resource group. They cannot be used with Cloudera-created multiple resource groups. When creating an environment from the UI, the “Create Private Endpoints” option is disabled if you select the option for Cloudera to create multiple resource groups. If using CDP CLI, a validation error appears if this option is used with Cloudera creating multiple resource groups.
- You can only use private endpoints with your own private DNS zone if you specify an existing VNet for the environment. If you chose Cloudera to create your network and enable private endpoints, then Cloudera is going to manage the private DNS zone for you.
- The private DNS zone provided to Cloudera must have the name “privatelink.postgres.database.azure.com“. Furthermore, it must be in a subscription that is accessible to the service principal used by the Cloudera app-based credential; That is, the subscription where the private DNS zone is created must be in the same tenant where the service principal is located.

Prerequisites

When bringing your own private DNS, you should meet the following prerequisites:

Disable private endpoint network policies

Only subnets that have private endpoint network policies turned off are eligible for private endpoint creation, because network security groups (NSG) are not supported for private endpoints.

You can use the following Azure CLI command to disable private endpoint network policies for a certain subnet:

```
az network vnet subnet update \
  --name <subnet-name> \
  --resource-group <resource-group-name> \
  --vnet-name <vnet-name> \
  --disable-private-endpoint-network-policies true
```

For example:

```
az network vnet subnet update \
  --name default-2 \
  --resource-group my-cdp-rg \
  --vnet-name my-azure-vnet \
  --disable-private-endpoint-network-policies true
```

Create a private DNS zone and link it to your VNet

If you choose to provide your own Azure private DNS zone then you should:

1. Create a private DNS zone with the name “privatelink.postgres.database.azure.com“ in any subscription accessible to the service principal used by the Cloudera app-based credential; That is, the subscription where the private DNS zone is created must be in the same tenant where the service principal is located.

   To create a private DNS zone, you can use the following Azure CLI command:

   ```
   az network private-dns zone create \
       --name privatelink.postgres.database.azure.com \
       --resource-group <YOUR_DNS_RESOURCE_GROUP> \
       --subscription <YOUR_SUBSCRIPTION_FOR_DNS>
   ```

2. Link it to the VNet that you are planning to use for the Cloudera environment.

   You can do this using the following Azure CLI command:

   ```
   az network private-dns link vnet create \
       --name <DESIRED_LINK_NAME> \
       --resource-group <YOUR_VNET_RESOURCE_GROUP> \
       --zone-name privatelink.postgres.database.azure.com \
       --virtual-network <YOUR_VNET_RESOURCE_ID> \
       --subscription <YOUR_VNET_SUBSCRIPTION>
   ```

Ensure that Cloudera has adequate permissions

Ensure that the role that you are using for the Azure credential has the permissions mentioned in Role definition 2: Allows Cloudera to access and use only a single existing resource group and create private endpoints.

Additionally, if your DNS zone is outside of the single resource group that you are planning to provide to Cloudera, the following additional permissions are required for the service principal to be able to discover, validate, and use the DNS zone (add and remove A records). These additional permissions need to be included in a separate role definition and assigned separately.

The placeholders in bold must be replaced with actual values:

- {SUBSCRIPTION-ID} - The ID of the subscription where the DNS zone is located.

- {RESOURCE-GROUP-NAME} - The name of the resource group where the DNS zone is located.

```
{
    "Name": "Cloudera Management Console Azure Operator for Using Private
DNS Zones",
    "IsCustom": true,
    "Description": "Can list, validate and use private DNS zones",
    "Actions": [
      "Microsoft.Network/privateEndpoints/privateDnsZoneGroups/read",
      "Microsoft.Network/privateEndpoints/privateDnsZoneGroups/write",
      "Microsoft.Network/privateDnsZones/join/action",
      "Microsoft.Network/privateDnsZones/read",
      "Microsoft.Network/privateDnsZones/virtualNetworkLinks/read"
    ],
    "NotActions": [],
    "DataActions": [
    ],
    "NotDataActions": [],
    "AssignableScopes": [
      "/subscriptions/{SUBSCRIPTION-ID}/resourcegroups/{RESOURCE-GROUP-
NAME}"
    ]
  }
```

Related Information

Manage network policies for private endpoints

Quickstart: Create an Azure private DNS zone using the Azure portal

Resources created under the hood

Creating a Cloudera environment on Azure with private endpoints in Cloudera involves creating several Azure resources, all of which are necessary to have a private endpoint setup that works out-of-the-box.

These create a setup where:

- A private endpoint to the Postgres server can be reached.
- A network interface with only a private IP address exists in the subnet where the private endpoint has been added.

This ensures that communication between Cloudera and the Azure Postgres server happens via a private IP address. However, cluster services do need to contact the Postgres server via FQDN, so the address needs to be resolvable from the VNet.

Resources created when using a Cloudera-managed private DNS zone

In the scenario where Cloudera creates the private DNS, several resources are created:

- A private DNS zone: It is a DNS zone, part of an Azure-hosted DNS server. It has a fixed name (“privatelink.postgres.database.azure.com”).
- A virtual network link between the zone and a VNet where the domain resolution should happen.
- An “A“ record within the zone: FQDN to IP address resolution. There is no reverse lookup.
- Since communication can flow over a private IP address, Cloudera sets “Deny public access” on the Azure Postgres server.

Note:

If there is a private DNS zone named “privatelink.postgres.database.azure.com” containing the required virtual network link in the deployment’s resource group, Cloudera reuses it. The “A“ record and the network link are created only if they do not already exist.

When you delete the environment, the DNS zone and the network link that you provided will not be deleted.

Resources created when using your own private DNS zone

In this case the private DNS zone and the network link are provided by you and Cloudera creates the following:

- Cloudera creates an “A“ record within the zone: FQDN to IP address resolution. There is no reverse lookup.
- Since communication can flow over a private IP address, Cloudera sets “Deny public access” on the Azure Postgres server.

When you delete the environment, the DNS zone and the network link that you provided will not be deleted.

Related Information

Virtual network workloads without custom DNS server

Azure services DNS zone configuration

Network security groups

Network security groups (NSGs) determine the inbound and outbound traffic to and from your Cloudera environment. That is, you should use security group settings to allow users from your organization access to Cloudera resources.

You have two options:

- Use your existing security groups (recommended for production)
- Have Cloudera create new security groups

You should verify the security group limits in your Azure account to ensure that you can create security groups for Cloudera.

Existing security groups

If you would like to create your own security groups, two security groups need to be created: the first security group will be used for all gateway nodes and the second security groups will be used for all other nodes. The gateway nodes communicate with the Cloudera Management Console and therefore require additional ports. These security groups will be applied when creating a data lake and FreeIPA during environment creation and when you create Cloudera Data Hub clusters.

Review the following guidelines prior to adding security groups rules. This describes all the inbound ports that need to be open and provides guidelines for what to enter as a source range:

Note: The communication via TCP/UDP 0-65535 and ICMP is essential for healthy operation of CDP environments, Data Hubs, and data services running within the VNet, so ensure that you open these ports as described below. While some services only need well-known fixed ports, a majority of them depend on ephemeral (i.e. dynamically or randomly allocated) ports; This is why the wildcard 0-65535 TCP/UDP port range is used in the absence of a detailed breakdown of individual ports. Since overall access to the VNet is typically secured by other means, the use of the wildcard rules does not pose a higher risk against external attacks.

“Knox” security group

This is used for gateway nodes:

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts. You should open it to your organization’s CIDR.

Protocol	Port Range	Source	Description
TCP	443	Your CIDR and CDP CIDR	This port is used to access the Data Lake and Data Hub cluster UIs via Knox gateway. You must open this port to your organization's CIDR in order to access cluster UIs. When CCM is enabled, you only need to set this to your CIDR.
TCP	9443	CDP CIDR	This port is used by CDP to maintain management control of clusters and data lakes. By default, when CDP creates the security groups automatically, it opens this port to the correct IP. This port is not needed when CCM is enabled.
TCP, UDP	0-65535	Your internal VNet CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VNet.
ICMP	N/A	Your internal VNet CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VNet.

"Default" security group

This is used for all nodes except Knox gateway nodes:

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to the hosts. You should open it to your organization's CIDR.
TCP	443	Your CIDR	This port is only required if you are planning to spin up Machine Learning workspaces since HTTPS access to ML workspaces is available over port 443. If you are not planning to use the Machine Learning service, you do not need to open this port.
TCP	9443	CDP CIDR	This port is used by CDP to maintain management control of clusters and data lakes. By default, when CDP creates the security groups automatically, it opens this port to the correct IP. This port is not needed when CCM is enabled.
TCP, UDP	0-65535	Your VNet CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VNet. TCP port 5432 is used by the Data Lake for communication with its attached database.
ICMP	N/A	Your internal VNet CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VNet.

Security groups can be created and managed from the Azure Portal > Network Security Groups. For detailed instructions on how to create new security groups on Azure, refer to Filter network traffic with a network security group using the Azure portal.

On the Network Interface page > Settings, click Network security group.

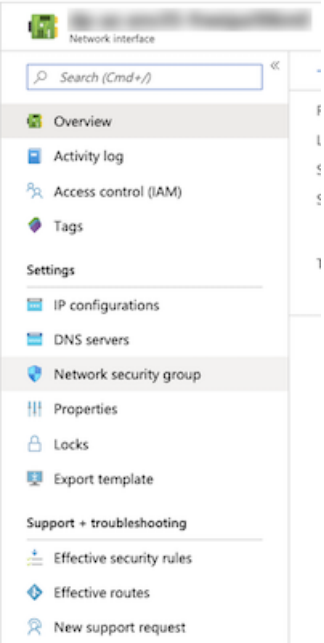

On the Network Security Groups page, click Inbound Security Rules to view the list of rules.

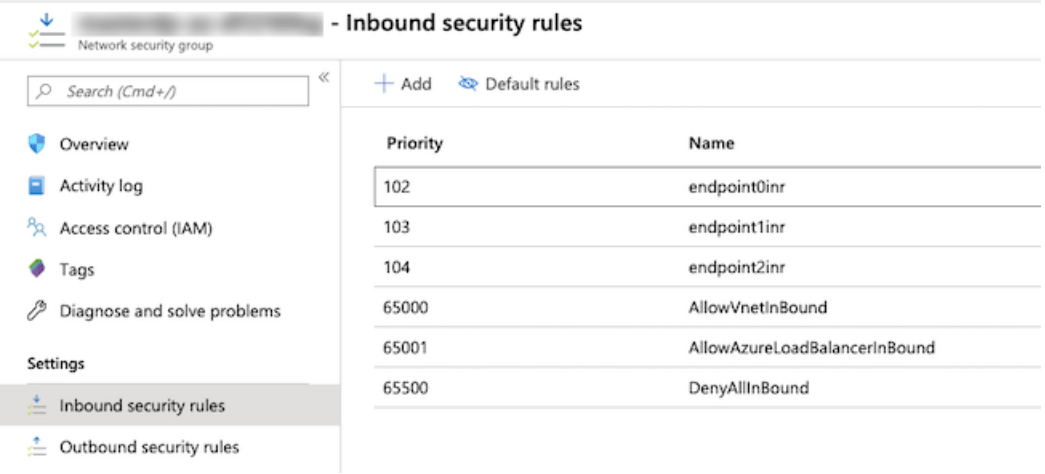

In the Inbound security rules tab, click Add.

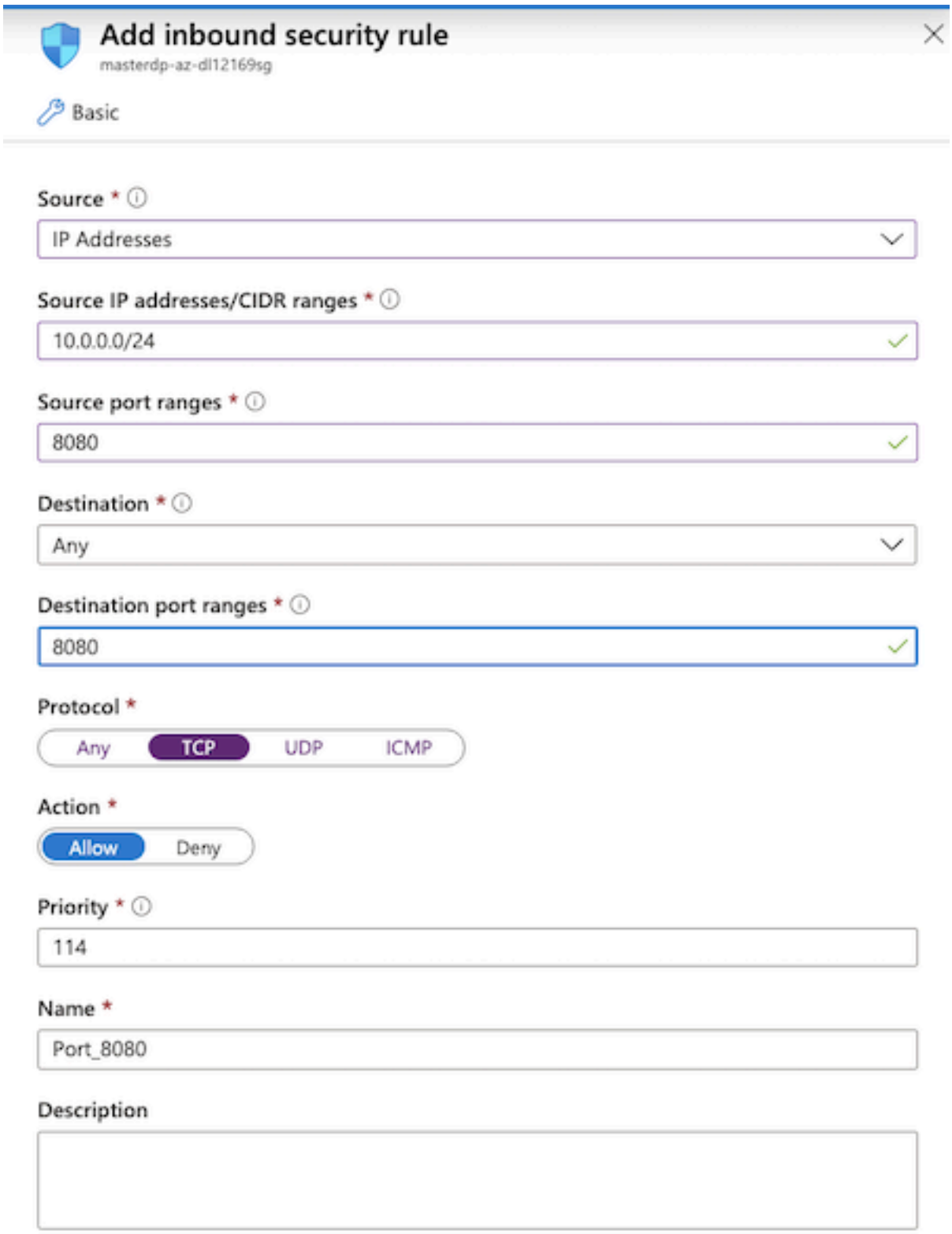

You need to create two security groups: Knox and Default (You will see this terminology in the Cloudera Management Console UI and CLI, so if you decide to choose different names, make sure that you are able to distinguish between the two security groups).

Note:

There is a known issue where even if you create and specify your own security groups the Cloudera Data Warehouse and Cloudera AI services create their own security groups. For instructions on how to restrict access on the security groups created by the Cloudera Data Warehouse service, refer to Restricting access to endpoints in AWS environments.

New security groups

If you would like Cloudera to create the security groups for you, you need to provide a CIDR range for inbound traffic to Azure instances from your organization. Cloudera creates multiple security groups: one for each Data Lake host group, one for each FreeIPA host group, and one per host group when Cloudera Data Hub, Cloudera Data Warehouse, and Cloudera AI clusters are created. On these security groups, Cloudera opens ports as described in Default security group settings documentation.

Note:

There is a known issue where even if you create and specify your own security groups, the Cloudera Data Warehouse and Cloudera AI services create their own security groups. For instructions on how to restrict access on the security groups created by the Cloudera Data Warehouse service, refer to Restricting access to endpoints in AWS environments.

Related Information

Restricting access for Cloudera services that create their own security groups on Azure

Default security group settings on Azure

Depending on what you chose during environment creation, Cloudera can create security groups for your environment automatically or you can provide your own security groups.

Note:

Even if you create and specify your own security groups, the Cloudera Data Warehouse and Cloudera AI services create their own security groups. Refer to Restricting access for Cloudera services that create their own security groups on Azure for instructions on how to restrict access.

Note: The communication via TCP/UDP 0-65535 and ICMP is essential for healthy operation of CDP environments, Data Hubs, and data services running within the VNet, so ensure that you open these ports as described below. While some services only need well-known fixed ports, a majority of them depend on ephemeral (i.e. dynamically or randomly allocated) ports; This is why the wildcard 0-65535 TCP/UDP port range is used in the absence of a detailed breakdown of individual ports. Since overall access to the VNet is typically secured by other means, the use of the wildcard rules does not pose a higher risk against external attacks.

Environment security groups

Depending on what you chose during environment creation, Cloudera can create security groups for your environment automatically or you can provide your own security groups.

- If you choose to use your own security groups, you are asked to create Knox and Default security groups as described in the Security groups documentation.
- If you choose for Cloudera to create all security groups required for an environment, the following security groups are created:

Data Lake: master

Azure naming convention: master${dl-name}${numeric-id}sg

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts. You should open it to your organization's CIDR.
TCP	443	Your CIDR and Cloudera CIDR	This port is used to access the Data Lake and Cloudera Data Hub cluster UIs via Knox gateway. You should open it to your organization's CIDR in order to access cluster UIs. This port is also required if you are planning to spin up Cloudera AI workbenches since HTTPS access to Cloudera AI workbenches is available over port 443. If you are not planning to use the Cloudera AI service, you do not need to open this port. When Cluster Connectivity Manager is enabled, you only need to set this to your CIDR.
TCP	9443	Cloudera CIDR	This port is used by Cloudera to maintain management control of clusters and data lakes. This port is not used when Cluster Connectivity Manager is enabled.
TCP, UDP	0-65535	Your VNet's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VNet.
ICMP	N/A	Your internal VNet CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VNet.

Data Lake: IDBroker

AWS naming convention: ${environment-name}-${random-id}-ClusterNodeSecurityGroupidbroker-${random-id}

Azure naming convention: idbroker${dl-name}${numeric-id}sg

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts.
TCP, UDP	0-65535	Your VNet's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VNet.
ICMP	N/A	Your internal VNet CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VNet.

FreeIPA

Azure naming convention: master0-${env-name}freeipa${numeric-id}sg

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts. You should open it to your organization's CIDR.

Protocol	Port Range	Source	Description
TCP	9443	Cloudera CIDR	This port is used by Cloudera to maintain management control of clusters and data lakes. This port is not used when Cluster Connectivity Manager is enabled.
TCP, UDP	0-65535	Your VNet's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VNet.
ICMP	N/A	Your internal VNet CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VNet.

Cloudera Data Hub security groups

Depending on what you chose during environment creation, Cloudera can create security groups for your Data Hub clusters automatically or it can use your pre-created security groups:

- If during environment creation, you provided your own security groups, Cloudera uses these security groups when deploying clusters.
- If during environment creation you chose for Cloudera to create new security groups, new security groups are created for each Cloudera Data Hub cluster as follows:

Cloudera Data Hub: master

Azure naming convention: ${hostgroup-name}${dh-name}${numeric-id}sg

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts. You should open it to your organization's CIDR.
TCP	443	Your CIDR and Cloudera CIDR	This port is used to access the Data Lake and Cloudera Data Hub cluster UIs via Knox gateway. You should open it to your organization's CIDR in order to access cluster UIs. When Cluster Connectivity Manager is enabled, you only need to set this to your CIDR.
TCP	9443	Cloudera CIDR	This port is used by Cloudera to maintain management control of clusters and data lakes. This port is not used when Cluster Connectivity Manager is enabled.
TCP, UDP	0-65535	Your VNet's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VNet.
ICMP	N/A	Your internal VNet CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VNet.

Cloudera Data Hub: worker

Azure naming convention: ${hostgroup-name}${dh-name}${numeric-id}sg

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts.
TCP, UDP	0-65535	Your VNet's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VNet.
ICMP	N/A	Your internal VNet CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VNet.

Cloudera Data Hub: compute

Azure naming convention: ${hostgroup-name}${dh-name}${numeric-id}sg

Protocol	Port Range	Source	Description
TCP	22	Your CIDR	This is an optional port for end user SSH access to cluster hosts.
TCP, UDP	0-65535	Your VNet's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).	This is required for internal communication within the VNet.
ICMP	N/A	Your internal VNet CIDR (for example 10.10.0.0/16).	This is required for internal communication within the VNet.

Cloudera Data Warehouse security groups

Cloudera always creates new security groups when Cloudera Data Warehouse are deployed.

Cloudera AI security groups

Cloudera always creates new security groups when Cloudera AI workbenches are deployed.

Cloudera Data Engineering security groups

Cloudera always creates new security groups when Cloudera Data Engineering clusters are deployed.

Cloudera DataFlow security groups

Cloudera always creates new security groups when Cloudera DataFlow environments are enabled.

SSH key pair

When registering an environment, you will be asked to upload an SSH key pair. The minimum SSH key size is 2048 bits.

The SSH key will be used for root-level access to Data Lake and Cloudera Data Hub instances.

If you need to generate an SSH key, you can use typical SSH key generation steps, such as those described in Quick steps: Create and use an SSH public-private key pair for Linux VMs in Azure .

Virtual machines

Cloudera provisions Virtual machines (VMs) as part of the environment creation process (for Data Lake and FreeIPA) and for compute clusters.

Therefore, you should verify the limits on the number and type of VM instances in your Azure account to ensure that you are able to provision an environment and create clusters in Cloudera.

Cloudera supports Azure reserved VM instances; That is, If you have purchased reserved instances, Cloudera uses them automatically according to Azure policy.

Cloudera provides default Virtual Hard Drives (VHDs) and provisions managed images based on these VHDs. These managed images are used for Data Lake, FreeIPA, and compute cluster instances.

For a list of supported VM types, refer to Cloudera Public Cloud service rates.

Custom images

By default Cloudera provides a set of default images that are used for all provisioned VMs, but you can optionally use custom images for Data Lake, FreeIPA, and Cloudera Data Hub.

You might require a custom image for compliance or security reasons (a “hardened” image), or to have your own packages pre-installed on the image, for example monitoring tools or software.

If you would like to use custom images instead of the default images, refer to Custom images and image catalogs.

ADLS Gen2 and managed identities for logs, backups, and data storage

Cloudera requires that you create and provide an Azure Data Lake Store Gen 2 (ADLS Gen2) storage account and create a container within it for storing workload data and logs. Furthermore, in order for Cloudera to be able to access this container, you must create and assign managed identities.

You must provide the following:

- Create an ADLS Gen2 storage account with hierarchical namespace enabled. The storage account must be in the same region as the environment.
- You must create and provide multiple user-assigned managed identities that allow access to the ADLS Gen2 location.

Cloudera supports both standard and premium ADLS Gen2 storage accounts.

For more information about the ADLS Gen2 container and managed identities setup, refer to the following documentation:

Minimal setup for Azure cloud storage

This minimal secure setup uses one ADLS Gen2 storage account with multiple containers in it, and multiple managed identities where each managed identity has at least one role assigned.

ADLS Gen2 storage account

You should create one ADLS Gen2 storage account with two containers within it (one for Storage Location Base and another for Logs Location Base). Additionally, you can specify a container for Backup Location Base to store FreeIPA and Data Lake backup data separately from logs:

- One ADLS Gen2 container is required to use as Storage Location Base such as abfs://storagefs@mydatalake.dfs.core.windows.net where mydatalake is your storage account name and storagefs is your container name. The Storage Location Base is used for storing workload data and Ranger audits.
- One ADLS Gen2 container is required to use as Logs Location Base such as abfs://logsfs@mydatalake.dfs.core.windows.net where mydatalake is your storage account name and logsfs is your container name. The Logs Location Base is used for Data Lake, FreeIPA and Data Hub logs, and FreeIPA and Data Lake backups.

  Note: Ranger audits are stored under Storage Location Base and not under Logs Location Base. The Logs Location Base is used for Data Lake, Data Hub and FreeIPA logs and, if no separate container is provided, FreeIPA backups.
- (Optional) One optional ADLS Gen2 container to use as Backup Location Base such as abfs://backupfs@mydatalake.dfs.core.windows.net where mydatalake is your storage account name and backupfs is your container name. The Backup Location Base is used for FreeIPA and Data Lake backups. If a separate container is not provided, the backups are stored in the Logs Location Base.

Storage Location Base examples

	abfs://storagefs@mydatalake.dfs.core.windows.net
Ranger Audit Logs	abfs://storagefs@mydatalake.dfs.core.windows.net/ranger/audit

Logs Location Base examples

	abfs://logsfs@mydatalake.dfs.core.windows.net
FreeIPA Logs	abfs://logsfs@mydatalake.dfs.core.windows.net/cluster-logs/freeipa If your environment was created prior to February 2021, this is abfs://logsfs@mydatalake.dfs.core.windows.net/freeipa

Backup Location Base examples

If you specify a separate container for FreeIPA backups, the backups are written to that container:

	abfs://backupfs@mydatalake.dfs.core.windows.net
FreeIPA Backup	abfs://backupfs@mydatalake.dfs.core.windows.net/cluster-backups/freeipa

If the separate container is not provided, the FreeIPA backups are written to the Logs Location Base. In both cases, the same cluster-backups/freeipa directory structure is created within the container.

Managed identities

You should create four managed identities.

The IDBroker component of Cloudera uses user-assigned managed identities for controlling access to ADLS Gen2 and stores and manages the mappings between the services/users and the corresponding managed identities. The following managed identities must be created:

Managed identity	Description	Steps
Assumer (Required)	During Data Lake cluster creation, Cloudera attaches this identity to the IDBroker VM. IDBroker then uses it to attach the other managed identities to the IDBroker VM. Once these identities are attached to the VM, IDBroker can acquire an access token for them (to eliminate the need to store credentials in the application).	**1.** Create a managed identity. **2.** Assign the Virtual Machine Contributor and Managed Identity Operator roles to this managed identity on the scope of either the subscription or resource group. **3.** Assign the Storage Blob Data Contributor role to this managed identity on the scope of the Logs Location Base and Backup Location Base (if created) containers created for Cloudera.
Data Lake Admin (Required)	This managed identity is used for Cloudera services to access data.	**1.** Create a managed identity. **2.** Assign the Storage Blob Data Owner role to this managed identity on the scope of the three containers (Storage Location Base, Logs Location Base, and Backup Location Base if it exists) created for Cloudera.

Managed identity	Description	Steps
Ranger Audit Logger (Required)	This managed identity is used by Ranger to write audits.	**1.** Create a managed identity. **2.** Assign the Storage Blob Data Contributor role to this managed identity on the scope of the Storage Location Base container created for Cloudera. **3.** Additionally, if you created a separate container for Backup Location Base, assign the Storage Blob Data Contributor role to this managed identity on the scope of the Backup Location Base. Otherwise, assign it on the Logs Location Base container.
Logger (Required)	This managed identity is used by Cloudera to write telemetry logs.	**1.** Create a managed identity. **2.** Assign Storage Blob Data Contributor role to this managed identity on the scope of the Logs Location Base and Backup Location Base (if created) created for Cloudera.
Ranger RAZ (Optional)	This managed identity is only required if you are planning to use Fine-grained access control. It is used by Cloudera to orchestrate fine-grained access control to your data storage location.	**1.** Create a managed identity. **2.** Assign the Storage Blob Data Owner role and the Storage Blob Delegator to this managed identity on the scope of the ADLS Gen2 storage account created for Cloudera. If you prefer not to assign the Storage Blob Data Owner role, you can Create a custom role to use in RAZ-enabled Azure environment and assign that role and the Storage Blob Delegator instead.

The following diagram illustrates the minimal required setup where Backup Location Base is in the same location as the Logs Location Base:

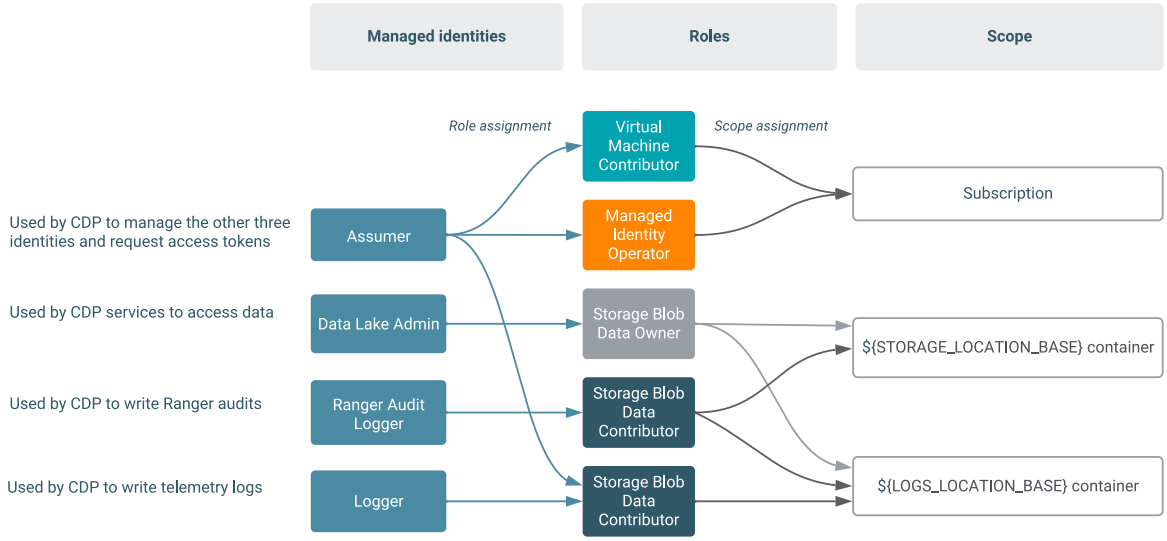

The following diagram illustrates the minimal required setup where Backup Location Base and Logs Location Base are separate:

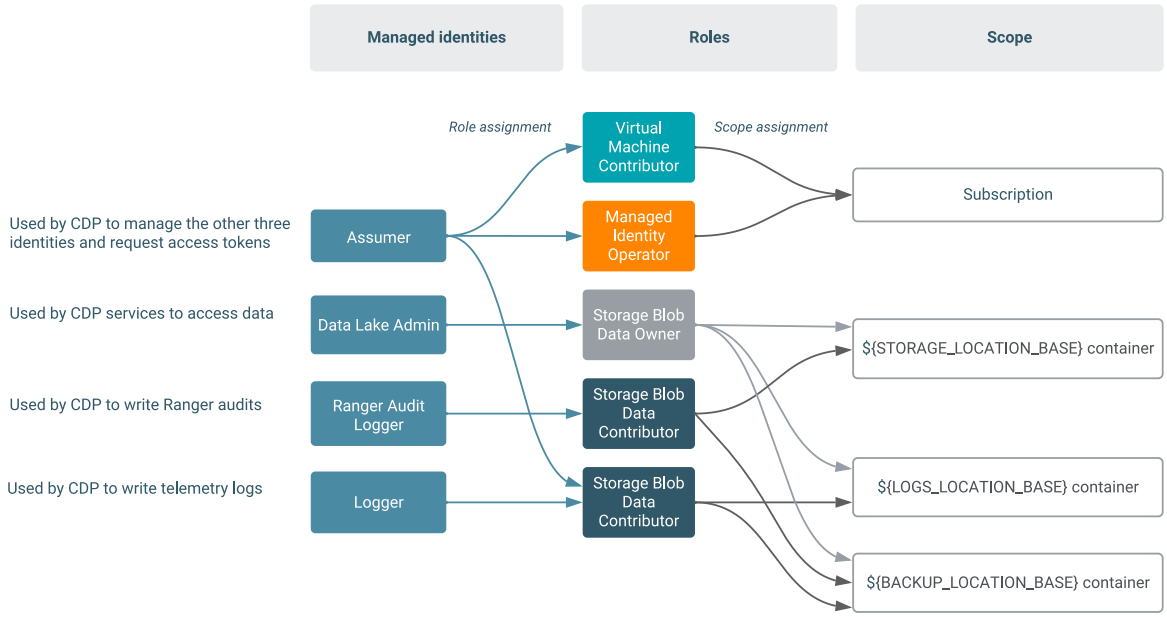

If you are planning to use fine-grained access control, one additional managed identity is required, as illustrated in the following diagrams.

The first diagram illustrates the scenario where Backup Location Base is in the same location as the Logs Location Base:

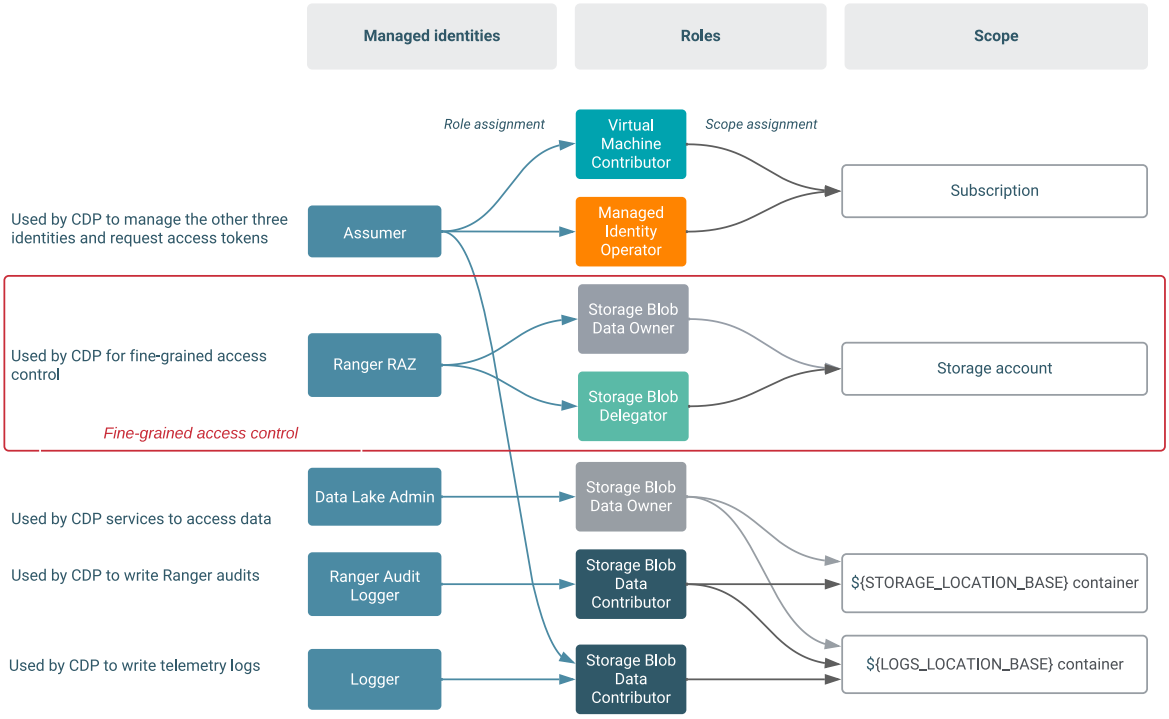

The second diagram illustrates the scenario where Backup Location Base and Logs Location Base are separate:

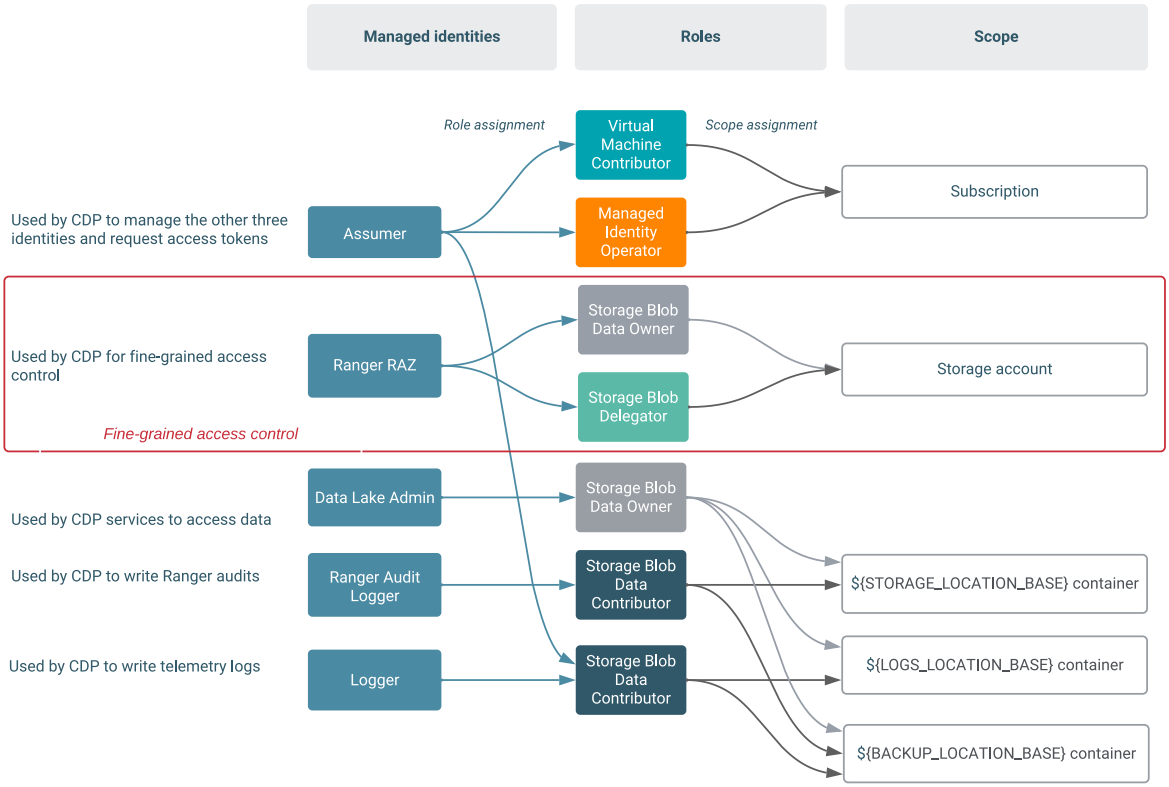

The following documentation provides detailed steps on how to create this setup. The steps involve:

1. Creating an ADLS Gen2 storage account and containers
2. Creating managed identities for the minimal setup
3. Providing the parameters in the Cloudera UI

Creating ADLS Gen2 storage account and containers

Create a resource group and then create a storage account and two containers within it. The storage account must have the hierarchical namespace enabled.

1. First, create a resource group that can act as a logical grouping of storage accounts. To create the resource group, follow the steps described in Create resource groups in Azure docs. Make sure to create it in the specific region that you would like to use for your environment.

2. Next, create an ADLS Gen2 storage account. In our example setup, the storage account name is my-datalake. To create the storage account, follow the Create an account using the Azure portal in Azure docs. In the Advanced > Data Lake Storage Gen2 section, make sure to Enable hierarchical namespace:

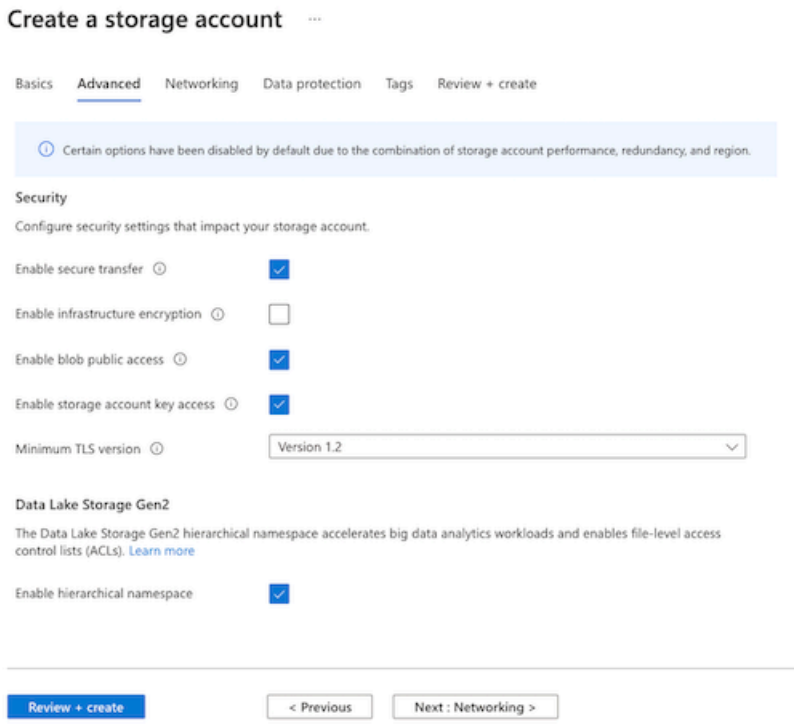

Note: You can reuse the same resource group that you created for the storage account or you can optionally create a new resource group that can act as a logical grouping of managed identities.

3. After creating an ADLS Gen 2 storage account, create two containers within it (one for Storage Location Base and another for Logs Location Base). You can also optionally create a third container for Backup Location Base.

   To create a container, follow the usual steps:

   a. On Azure Portal, navigate to Storage Accounts > your newly created storage account > Containers > +Container.
   b. Provide a name for your container and click OK.

   Repeat these steps to create all required containers. In our example setup, the containers are called storagefs, logs fs, and backupfs.

Once you have created the storage account and container, note the created resources in the following format:

abfs://$[container-name]@$[storage-account-name].dfs.core.windows.net

For example, in our example setup, we created the following two containers:

abfs://storagefs@mydatalake.dfs.core.windows.net

abfs://logsfs@mydatalake.dfs.core.windows.net

abfs://backupfs@mydatalake.dfs.core.windows.net

Creating Azure managed identities

Once you've created the storage account and file system within it, create the managed identities and then assign roles with specific scopes to these identities.

You can reuse the same resource group that you created for the storage account or you can optionally create a new resource group that can act as a logical grouping of managed identities.

You need to create four or five managed identities (Assumer, Data Lake Admin, Ranger Audit Logger, Logger, and optionally the Ranger RAZ). Use the following steps to create these managed identities:

1. On Azure Portal, navigate to Managed Identities.
2. Click +Add.
3. Specify managed identity name and select the resource group that you created earlier.

Repeat these steps to create each of the four managed identities. Once you've created these managed identities, assign roles with specific scopes (subscription or storage account) to these identities as follows.

Create the Assumer identity

Assign the Virtual Machine Contributor role and the Managed Identity Operator role to the Assumer managed identity on either the subscription or resource group level, and then assign the Storage Blob Data Contributor on the scope of the container created earlier for Logs Location Base.

Steps

1. Navigate to Subscriptions > your subscription > Access Control (IAM).
2. Click +Add.
3. Under Add role assignment:
   a. Under Role, select Virtual Machine Contributor.
   b. Under Assign access to, select User assigned managed identity.
   c. Under Select, select the Assumer managed identity created earlier.
   d. Click Save.
4. Repeat the role assignment steps 2-3, but this time assign the Managed Identity Operator role to the Assumer Identity.

Next, assign the Storage Blob Data Contributor role to the Assumer managed identity on the scope of the containers created earlier for Logs Location Base and Backup Location Base (if separate):

1. Navigate to Storage accounts > your storage account > Containers > your container > Access Control (IAM).
2. Click +Add > Add role assignment.
3. Under Add role assignment:
   a. Under Role, select Storage Blob Data Contributor.
   b. Under Assign access to, select User assigned managed identity.
   c. Under Select, select the Assumer managed identity created earlier.
   d. Click Save.

Create the Data Lake Admin identity

Assign the Storage Blob Data Owner role to the Data Lake Admin managed identity on the scope of the two containers created earlier for Storage Location Base and Logs Location Base, and to the Backup Location Base container if it exists. You need to do this separately for each of the containers.

Steps

1. Navigate to Storage accounts > your storage account > Containers > your container > Access Control (IAM).
2. Click +Add > Add role assignment.
3. Under Add role assignment:
   a. Under Role, select Storage Blob Data Owner.
   b. Under Assign access to, select User assigned managed identity.
   c. Under Select, select the Data Lake Admin managed identity created earlier.
   d. Click Save.

Repeat the steps for the second container and for the third container (if you created it).

Create the Ranger Audit Logger identity

Assign the Storage Blob Data Contributor role to the Ranger Audit Logger managed identity on the scope of the container created earlier for Storage Location Base. If you created a separate container for Backup Location Base, you should also repeat these steps on the container created earlier for Backup Location Base. Otherwise, they should be repeated on the Logs Location Base container.

Steps

1. Navigate to Storage accounts > your storage account > Containers > your container > Access Control (IAM).
2. Click +Add > Add role assignment.
3. Under Add role assignment:
   a. Under Role, select Storage Blob Data Contributor.
   b. Under Assign access to, select User assigned managed identity.
   c. Under Select, select the Ranger Audit Logger managed identity created earlier.
   d. Click Save.

Repeat the steps for the second container and for the third container (if you created it).

Create the Logger identity

Assign the Storage Blob Data Contributor role to the Logger managed identity on the scope of the container created earlier for Logs Location Base. If you created a separate container for Backup Location Base, you should also assign the same role to the same managed identity on the scope of the container created earlier for Backup Location Base.

Steps

1. Navigate to Storage accounts > your storage account > Containers > your container > Access Control (IAM).
2. Click +Add > Add role assignment.
3. Under Add role assignment:
   a. Under Role, select Storage Blob Data Contributor.
   b. Under Assign access to, select User assigned managed identity.
   c. Under Select, select the Logger managed identity created earlier.
   d. Click Save.
4. If you created a separate container for Backup Location Base, you should perform steps 1-3 to also assign the Storage Blob Data Contributor role to the Logger managed identity on the scope of the container created earlier for Backup Location Base.

Create the Ranger RAZ identity (Optional)

If you would like to use Fine-grained access control, you should create the Ranger RAZ managed identity. Assign the Storage Blob Data Owner role and the Storage Blob Delegator role to the Ranger RAZ managed identity on the scope of the storage account that you created earlier (which contains the Storage Location Base and Logs Location Base).

Steps

1. Navigate to Storage accounts > your storage account > Access Control (IAM).
2. Click +Add > Add role assignment.
3. Under Add role assignment:
   a. Under Role, select Storage Blob Data Owner.
   b. Under Assign access to, select User assigned managed identity.
   c. Under Select, select the Ranger RAZ managed identity created earlier.
   d. Click Save.
4. Repeat the role assignment steps 2-3, but this time assign the Storage Blob Delegator role to the Ranger RAZ managed identity.

After performing these steps, you should have the required managed identities created and their roles assigned on the correct scope.

Providing the parameters in the Cloudera UI

Once you've created the ADLS Gen2 location and the required managed identities, provide the information related to these resources in the Register Environment wizard.

Parameter	Description
Data Access and Audit	
Assumer Identity	Select the Assumer identity created earlier.
Storage Location Base	Enter the Storage Location Base created earlier. In our example this was abfs://storagefs@mydatalake.dfs.core.windows.net, so you should enter storagefs@mydatalake.
Data Access Identity	Select the Data Lake Admin identity created earlier.
Ranger Audit Identity	Select the Ranger Audit identity created earlier.
Fine-grained access control on ADLS Gen2	
Enable Ranger authorization for ADLS Gen2	If you would like to use Fine-grained access control, click on "Enable Ranger authorization for ADLS Gen2". Next, select the Ranger RAZ identity created earlier.
Select Azure managed identity for Ranger authorization	
Logs	
Logger Identity	Select the Logger identity created earlier.
Logs Location Base	Enter the Logs Location Base created earlier. In our example this was abfs://logsfs@mydatalake.dfs.core.windows.net, so you should enter logsfs@mydatalake.
Backup Location Base (Optional)	If you created it, enter the Backup Location Base. In our example this was abfs://backupfs@mydatalake.dfs.core.windows.net, so you should enter backupfs@mydatalake. This is optional. If you don't provide this, FreeIPA and Data Lake backups will be stored in the Logs Location Base.

Onboarding Cloudera users and groups for Azure cloud storage (RAZ environments)

If your Azure environment has Fine-grained access control enabled, you should onboard your users using Ranger instead of using IDBroker.

For more information, refer to Using Ranger to Provide Authorization in ClouderaUsing Ranger to Provide Authorization in Cloudera.

Onboarding Cloudera users and groups for Azure cloud storage (No RAZ)

The minimal setup defined earlier spins up a Cloudera environment and Data Lake with no end user access to cloud storage. Adding users and groups to a Cloudera environment involves ensuring that they are properly mapped to managed identities to access cloud storage.

Note: If you are using Fine-grained access control, you should onboard your users using Ranger instead of using IDBroker mappings. Adding IDBroker mappings is disabled for RAZ-enabled environments.

In general, to have new users or groups onboarded, you need to have the following pre-created in Azure:

1. First, you need to create two more containers within the storage account (mydatalake) created earlier, one for data engineers (for example, dataeng) and one for data scientists (for example, datascience).
2. Next, you need to create two more managed identities, one for data engineers (for example, data-eng-mi) and one for data scientists (for example, data-science-mi) and assign the Storage Blob Data Owner role on the scope of one these two nearly created containers. The data-eng-mi managed identity will need the Storage Blob Data Owner role on the scope of the dataeng container and the data-science-mi managed identity will need the Storage Blob Data Owner role on the scope of the datascience container.

3. Finally, you also need to grant the Data Lake Admin managed identity created earlier the Storage Blob Data Owner role on the scope of these two newly created containers.

The final goal is to have the following that builds on the minimal setup presented earlier.

The first diagram illustrates the scenario where Backup Location Base is in the same location as the Logs Location Base:

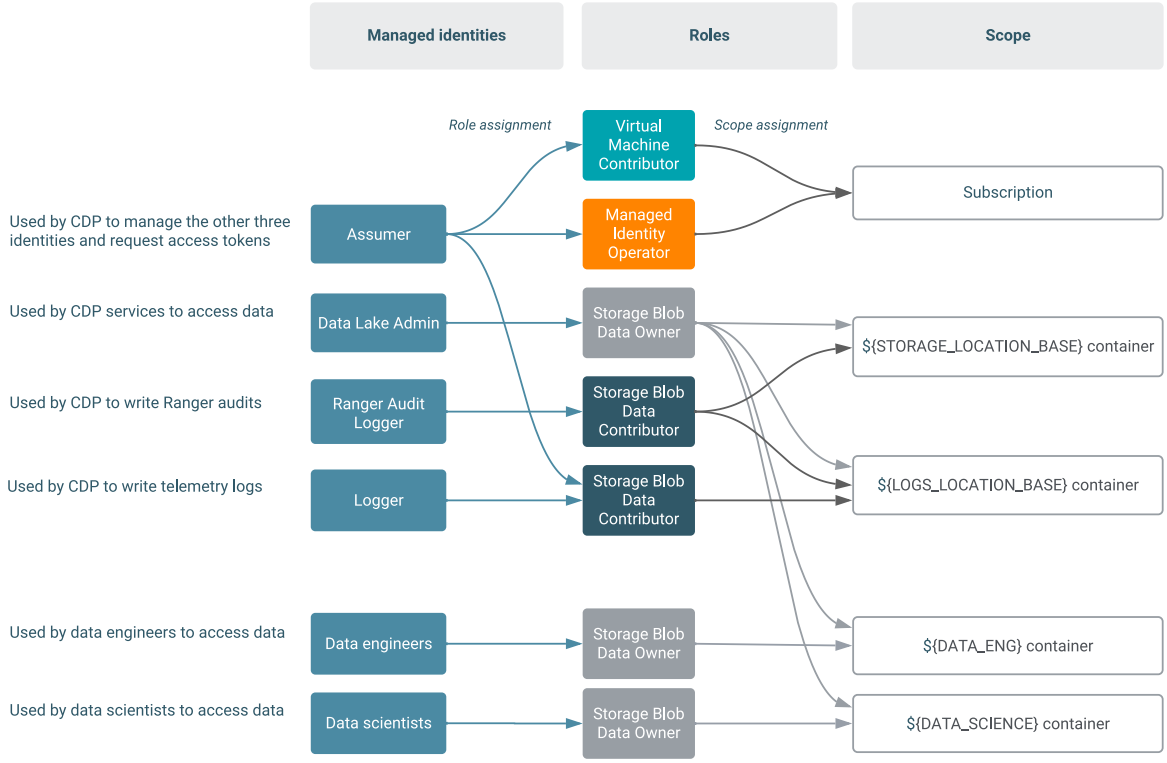

The second diagram illustrates the scenario where Backup Location Base and Logs Location Base are separate:

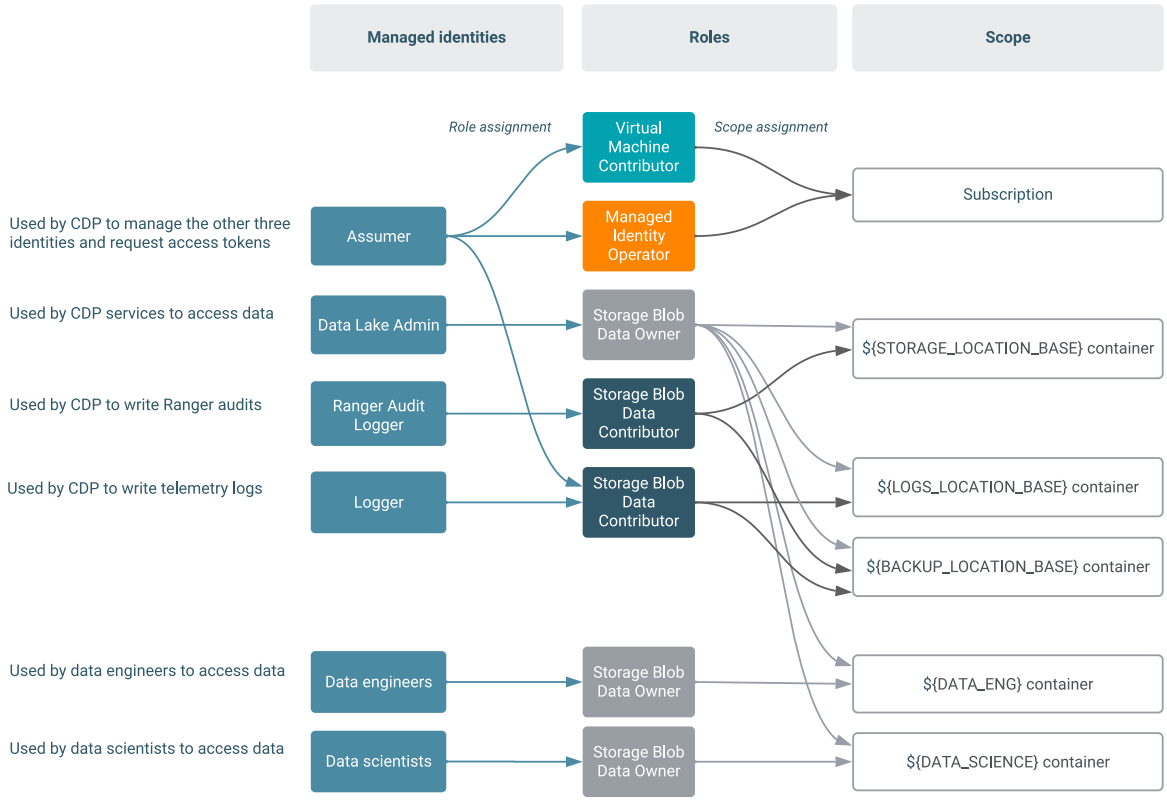

The following documentation provides detailed steps for how to create this setup. The steps involve:

1. Creating additional containers
2. Crating additional managed identities
3. Creating mappings in Cloudera
4. Updating the Data Lake Admin managed identity

Creating additional containers

Within the ADLS Gen 2 storage account (mydatalake) created earlier, create two more containers (dataeng and data science).

To create a file system, perform the following steps:

1. On Azure Portal, navigate to Storage Accounts > your newly created storage account > Containers > +Container.
2. Provide a name for your container and click OK.

Repeat these steps to create both containers.

Creating additional managed identities

After creating the two additional container, create two additional managed identities, one for data engineers (data-eng -mi) and one data scientists (data-science-mi).

To create the two new managed identities, perform the following steps:

1. On Azure Portal, navigate to Managed Identities.
2. Click +Add.
3. Specify managed identity name and select the resource group that you created earlier.

Repeat these steps to create each of the two managed identities. Once you've created these managed identities, assign roles with specific scopes (limited to one of the two containers, dataeng or datascience respectively) to these identities as follows:

1. Navigate to Storage accounts > your storage account > Containers > your container > Access Control (IAM).

2. Click +Add > Add role assignment.
3. Under Add role assignment:
   a. Under Role, select Storage Blob Data Owner.
   b. Under Assign access to, select User assigned managed identity.
   c. Under Select, select the managed identity.
   d. Click Save.

After performing these steps for each of the two managed identities, you should have the required managed identities created and their roles assigned on the correct scope.

Adding Cloudera user/group to managed identity mappings

After creating the two additional managed identities, one for data engineers (data-eng-mi) and one data scientists (data-science-mi), map them to specific user/group in Cloudera.

Note: If you are using Fine-grained access control, you should onboard your users using Ranger instead of using IDBroker mappings. Adding IDBroker mappings is disabled for RAZ-enabled environments.

Note:

If a user is mapped to multiple roles via group membership, the specific role to be used needs to be provided at runtime. If the user is mapped directly to a role, the direct mapping takes precedence over mapping via group membership. For information on how to specify the role, refer to Specifying a group when user belongs to multiple groups.

Required role: DataSteward, EnvironmentAdmin, or Owner

Steps

For CDP UI

1. The option to add/modify these mappings is available from the Management Console under Environments > click on an environment > Actions > Manage Access > IDBroker Mappings > Edit.
2. Under Current Mappings, click Edit.
3. Click + to display a new field for adding a mapping.
4. Provide the following:
   a. The User or Group dropdown is pre-populated with Cloudera users and groups. Select the user or group that you would like to map.
   b. Under Role, specify the resource ID of a managed identity (copied from Azure Portal). You should select your data-eng-mi here.
5. Repeat the previous two steps to add additional mapping for the data-science-mi.
6. Click Save and Sync.

For CDP CLI

If you would like to create the mappings via CDP CLI, you can:

1. Use the cdp environments get-id-broker-mappings command to obtain your current mappings.
2. Use the cdp environments set-id-broker-mappings command to set additional mappings. The only way to use this command is to:
   - Pass all the current mappings
   - Add the new mappings
3. Next, sync IDBroker mappings. For example:

```
cdp environments sync-id-broker-mappings --environment-name demo3
```

4. Finally, check the sync status. For example:

```
cdp environments get-id-broker-mappings-sync-status --environment-name d
emo3
```

Updating access for the Data Lake Admin managed identity

Grant the Data Lake Admin identity created earlier the Storage Blob Data Owner role on the scope of the two newly created containers.

Perform the following steps for both dataeng or datascience containers to grant the Data Lake Admin managed identity access to them:

1. Navigate to Storage accounts > your storage account > Containers > your container > Access Control (IAM).
2. Click +Add > Add role assignment.
3. Under Add role assignment:
   a. Under Role, select Storage Blob Data Owner.
   b. Under Assign access to, select User assigned managed identity.
   c. Under Select, select the Data Lake Admin Identity created earlier.
   d. Click Save.

Repeat these steps to provide Data Lake admin with access to both containers.

Using ADLS Gen2 encryption

By default ADLS Gen2 uses TLS. If you use the fs.azure.always.use.https property to turn off this behavior, you must specify abfss as the prefix in the URI to use TLS. Otherwise, you can use abfs.

ADLS Gen2 account for storing operating system images

Cloudera uses an ADLS Gen2 storage account for storing operating system images used for VMs. By default, Cloudera creates this account during environment registration, but you can optionally pre-create it. If needed, you can also copy the VHD files and create image resources manually.

Cloudera uses an ADLS Gen2 for storing operating system images used for VMs. These images are used for:

- Data Lake and Cloudera Data Hub clusters (A single image is used for both)
- FreeIPA (A separate image is used)

For each of these, there is one image for each Runtime version and for each Azure region.

Preparing images for VMs on Azure is a three-step process, involving the following steps:

1. Creating a new storage account in your subscription. The storage account is reachable from all networks so that Cloudera can access it (although the storage container is private).
2. Copying one or more virtual hard disks (VHDs) from Cloudera's regional storage account to your storage account created earlier.
3. Creating an image resource from each copied VHD.

By default, these steps are performed automatically by Cloudera during the "Setting up Cloudera image" and the "Creating infrastructure" steps of launching a stack. However, if due to your organization's security policies, you do not want Cloudera to create your storage account or you do not want the storage account to be publicly available, you have two alternatives:

Scenario	Solution	Steps
You do not want Cloudera to create a storage account in your Azure subscription or Cloudera is not granted the permission to create new storage accounts in your Azure subscription, but you are willing to provide Cloudera access to the storage account once it exists.	Manually pre-create a storage account that is available from all networks, and once it exists, Cloudera can copy VHD files and create the image resources.	Refer to Creating the storage account for images manually.
Image copying by Cloudera is not possible because the storage account should not be publicly available.	You should perform all three steps manually.	See all three steps linked in this section.

Note:

The storage account and the VHDs themselves are not deleted upon deleting the environment to save time when creating future environments.

If you would like to perform these manual steps, refer to the following instructions.

Creating a storage account for operating system images manually

In some cases where your organization's security policies require it, you may want to manually pre-create the storage account that Cloudera uses for storing operating system images.

Steps

1. Create a resource group on Azure with the name "cloudbreak-images".
2. Create a storage account on Azure.
   a. The name is the concatenation of the following elements. It is different depending on whether or not you would like to use your existing resource group for Cloudera:

Table 1:

	Naming convention
Single existing resource group	Enter a name that is a simple concatenation of the following four elements: • "cbimg" • Region identifier: The starting letters of the region where the SA is. For example, "eu" for East US, or "eu2" for East US 2 • The adler32 checksum without leading zeroes of the Subscription ID, without hyphens ('-') and all lowercase. For example a9d4456e-349f-46f5-bc73-54a8d523e504 => a9d4456e349f46f5bc7354a8d523e504 => 8e63086f • The adler32 checksum without leading zeroes of the resource group name. For example rg-my-cool-single-rg => 4e23077c When you put these together, if the storage account is in East US 2 region in a resource group called "rg-my-cool-single-rg" and with a Subscription ID of a9d4456e-349f-46f5-bc73-54a8d523e504, the storage account name would be cbimgeu28e63086f4e23077c For the adler32 calculation, you can use the Online ADLER32 Hash Calculator. **Note:** Make sure to remove all leading zeroes from the calculator's output. For example, if your resource group name is "cdppoc", convert it as follows: cdppoc => 089d027a => 89d027a

	Naming convention
Multiple resource groups created by Cloudera	Enter a name that is the concatenation of the following three elements: • "cbimg" • Region identifier: The starting letters of the region where the SA is. For example, "eu" for East US, or "eu2" for East US 2 • Subscription ID, without hyphens ('-') and all lowercase. For example, if your subscription ID is a9d4456e-349f-46f5-bc73-54a8d523e504, you should convert it to a9d4456e349f46f5bc7354a8d523e504 When you put these together, if the storage account is in East US 2 region in subscription a9d4456e-349f-46f5-bc73-54a8d523e504, the storage account name would be cbimgeu2a9d4456e349f46f5bc7354a8d523e504

b. Disable hierarchical namespace. Make sure not to omit this setting as it cannot be changed after the storage account is created.

Copying an image manually

In some cases where your organization's security policies require it, you must manually copy images to the storage account designated for image storage.

Caution:

Cloudera creates new images for Cloudera frequently for all new Runtime versions and occasional OS bug fixes. Whenever a new image is available, it is not possible to launch new clusters until the image is copied. Furthermore, many features may not work correctly unless new images are copied to your storage account. For example, OS upgrade requires new images being present in your storage account.

The presence of new images can be monitored in the Cloudera image catalog at https://cloudbreak-imagecatalog.s3.amazonaws.com/v3-prod-cb-image-catalog.json or in Cloudera web interface > Cloudera Management Console> Shared Resources > Image Catalogs > cdp-default.

When copying the images, follow these high-level steps:

1. Create a container named "images" and copy all images to that container. Make sure to create a page blob.
   - Note the URL of the created container. You will need it to perform the copying process.
2. Identify the FreeIPA image that needs to be copied. You should use the latest image from the FreIPA image catalog available at https://cloudbreak-imagecatalog.s3.amazonaws.com/v3-prod-freeipa-image-catalog.json.
3. Identify the Data Lake and Cloudera Data Hub image(s) that need to be copied and note the VHD URLs. Both Data Lake and Cloudera Data Hub use the same image, so there is one image used for both for each Runtime version. There are two ways to find these images in the Cloudera image catalog:
   - In Cloudera web interface > Cloudera Management Console > Shared Resources > Image Catalogs > cdp-default > search for your Cloudera Runtime version > Click on UUID of your selected entry (For each Cloudera Runtime version, there should be only one entry for Azure cloud platform) > You will see VHD URLs, one per region.
   - At https://cloudbreak-imagecatalog.s3.amazonaws.com/v3-prod-cb-image-catalog.json.
4. Use azcopy tool to copy the images. When copying the images via the azcopy tool, you have two options for authentication and authorization:
   - Azcopy login: Perform azcopy login with a service principal that has the permission to copy into images container.
   - SAS token: Create a SAS token for the images container.

Option 1: Azcopy login

Issue the following command and it will perform an interactive login:

```
azcopy login
```

If you want to set up an automatic script then you can use:

```
azcopy login --service-principal \
 --application-id <YOUR_APPLICATION_ID> \
 --tenant-id <YOUR_TENANT_ID>
```

Note:

On some distributions (such as Ubuntu Linux) login executes fine, but logout fails (and so would any further authorization). For troubleshooting steps, refer to https://github.com/Azure/azure-storage-azcopy/issues/452.

To run azcopy, use:

```
azcopy copy \
 'https://<SOURCE_BLOB_URL>' \
 'https://<DESTINATION_BLUB_URL>
```

As mentioned earlier, the https://<SOURCE_BLOB_URL> can be found in the image catalog.

Option 2: SAS-token

Go to the Azure portal and generate a SAS-token. Then, you can issue azcopy:

```
azcopy copy \
 'https://<SOURCE_BLOB_URL>' \
 'https://<DESTINATION_BLUB_URL>?<YOUR_SAS_TOKEN>'
```

As mentioned earlier, the https://<SOURCE_BLOB_URL> can be found in the image catalog.

For more information about Azure copy syntax and examples, refer to Get started with AzCopy in Azure docs.

Creating an image resource

After the copying process is complete, create an image resource from each copied VHD.

In Azure Portal you can do that as follows:

1. Navigate to Images and click on Add.

2. Provide the following information:

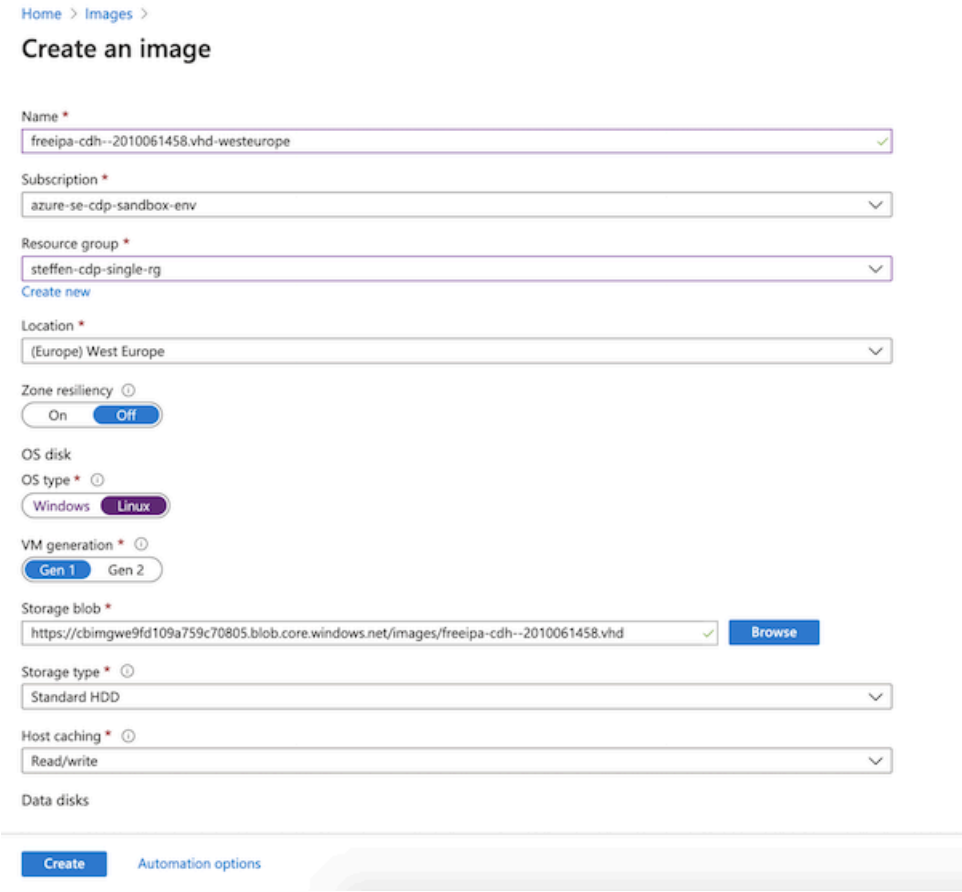

3. Once done, click Create.

Supported Azure block storage

Review the list of supported Azure disk types for Cloudera.

Cloudera supports the following Azure disk types:

Cloudera name	Azure name
Standard_LRS	Standard HDD
StandardSSD_LRS	Standard SSD
Premium_LRS	Premium SSD

Related Information

Azure managed disk types

Azure Database for PostgreSQL

Cloudera provisions a PostgreSQL database as part of the environment creation process (for Data Lake). Prior to creating an Azure environment in Cloudera, the Azure Database for PostgreSQL must be enabled in the regions that you want to use with Cloudera.

Furthermore, Cloudera Data Warehouse service provisions PostgreSQL databases for its data warehouses.

Enabling Azure Database for PostgreSQL

The Azure Database for PostgreSQL service is often not enabled by default in every region, and your Azure administrator may need to contact Azure Support to have it enabled. For information on submitting this request, refer to Azure region access request process documentation. When making this request, do the following:

- Specify "Azure Database for PostgreSQL" in the Subscription field in Step 1.
- Specify "Azure Database for PostgreSQL" in the Region to Enable field in Step 4.
- Finally, in Step 4, specify 100 in the Planned Compute Usage in Cores field. This limits the pool of VCores per region to 100 VCores, limiting the total number of PostgreSQL servers that Cloudera can provision.

Cluster deployments with Azure Policies for PostgreSQL

Cluster deployments with Azure Policies for PostgreSQL are not supported. Ensure that none of the Azure Policy built-in definitions for Azure Database for PostgreSQL are turned on or enforced. This is necessary because some policies (such as log_duration) could cause such performance degradation, making your clusters practically unusable.

Related Information

Private endpoint for Azure Postgres

Encrypting VM disks with customer managed keys

By default, local Data Lake, FreeIPA, and Cloudera Data Hub disks attached to Azure VMs and the PostgreSQL server instance used by the Data Lake and Cloudera Data Hub clusters are encrypted with server-side encryption (SSE) using Platform Managed Keys (PMK), but you can optionally configure SSE with Customer Managed Keys (CMK).

The CMK can be specified during environment registration and, if present, is used for encrypting Data Lake, FreeIPA, and Cloudera Data Hub disks and PostgreSQL server instances.

The disks that are attached to the VMs of the Data Lake, FreeIPA, and Cloudera Data Hub clusters will be associated with a Disk Encryption Set (DES) that is created with the key URL as the underlying encryption key version. The DES dedicated to the Cloudera environment will be created in the resource group of the environment before the FreeIPA launch at the beginning of the environment creation process.

When meeting Azure requirements for Cloudera, you should do the following:

- Add additional permissions for Cloudera provisioning credential
- Create a key vault and a vault key
- If you are using Azure Database for PostgreSQL Flexible Server with Cloudera, you can optionally use the CMK used for encrypting VM disks for encrypting the Azure Flexible Server database instance used by Cloudera. In this case, you should create a managed identity.

Note:

The CMK used for VM encryption can also be used for encrypting Azure Database for PostgreSQL Flexible Server used by Cloudera; this can be done through a user-supplied managed identity, as described in Configuring a CMK for data encryption in Azure Database for PostgreSQL Flexible Server and Managed identity for encrypting Azure Database for PostgreSQL Flexible Server. If the Flexible Server encryption with a CMEK is set up, the permissions for accessing the CMEK in Key Vault are conveyed by the user-supplied managed identity. In this case, the Key Vault resource may be configured with the "access policy" model or "RBAC" access control model. Cloudera will not make any changes to the Key Vault access policy; all permissions for accessing the Key Vault are encapsulated by the user-supplied managed identity. When using the "access policy" model, granting prior access for the user-supplied managed identity in the Key Vault access policy is your responsibility.

In other words, when you provide the managed identity for CMEK, VM disk encryption (via the DES resource) works identical to Flexible Server encryption.

If the user-supplied managed identity is not provided, the permissions for accessing the CMEK in Key Vault are conveyed using a system-assigned managed identity created by Cloudera. In this case, the Key Vault resource must be configured with the "access policy" model. Cloudera will automatically grant and revoke permissions for the DES system-assigned managed identity principal in the Key Vault resource access policy.

For more information about Azure Key Vault's authorization systems, see Azure role-based access control (Azure RBAC) vs. access policies (legacy).

Add additional permissions to your Azure policy

Make sure that the following additional permissions are set up for the Azure credential used in Cloudera environment creation, in addition to what is documented in Azure permissions.

All of them are actions and shall be granted at the scope of the resource group hosting the Cloudera environment:

```
"Microsoft.KeyVault/vaults/read",
"Microsoft.KeyVault/vaults/write",
"Microsoft.KeyVault/vaults/deploy/action",
"Microsoft.Compute/diskEncryptionSets/read",
"Microsoft.Compute/diskEncryptionSets/write",
"Microsoft.Compute/diskEncryptionSets/delete",
"Microsoft.DBforPostgreSQL/servers/read",
"Microsoft.DBforPostgreSQL/servers/keys/write",
"Microsoft.KeyVault/vaults/accessPolicies/write"
```

Note:

When a user-supplied managed identity is created for Flexible Server encryption, the following permissions are not needed:

```
"Microsoft.KeyVault/vaults/accessPolicies/write"
"Microsoft.KeyVault/vaults/write"
```

See Managed identity for encrypting Azure Database for PostgreSQL Flexible Server.

The following table explains why Cloudera needs these permissions:

Permission	Description
Microsoft.KeyVault/vaults/read and Microsoft.KeyVault/vaults/write	Microsoft.KeyVault/vaults/read is required to read the vaults. Without this, vaults in your subscription cannot be detected by Cloudera. Specifically, Cloudera must update the Key Vault access policy to add an entry for the DES SP. For this, Cloudera invokes an "update key vault" operation, which Azure implements as the following series of steps: **1.** Read Key Vault details, including its original access policy (Requires Microsoft.KeyVault/vaults/read). **2.** Add entry for DES SP. **3.** Write Key Vault details, including the updated access policy. In this step, Microsoft.KeyVault/vaults/write is required for adding an entry for the DES SP for the operations get, unwrap key and wrap key.
Microsoft.KeyVault/vaults/write	This is required to update the access policies for the DES created in the vault.
Microsoft.KeyVault/vaults/deploy/action	This is required to create DES resources. Specifically, it's required for DES SP creation (performed automatically alongside the DES creation) that will ultimately be used to access the Key Vault from the DES.
Microsoft.Compute/diskEncryptionSets/read	This is required to check for the existence of and fetch the properties and status of DES resources created by Cloudera.
Microsoft.Compute/diskEncryptionSets/write	This is required to create DES resources.
Microsoft.Compute/diskEncryptionSets/delete	This is required to delete the DES during environment termination.
Microsoft.DBforPostgreSQL/servers/keys/write and Microsoft.KeyVault/vaults/accessPolicies/write	This is required for setting up access policies and keys via an Azure Resource Manager template.

Create a vault and add a vault key

You can use your existing vault and vault key or create a new vault and vault key.

Regardless of which of the two options you choose, the vault and the vault key must fulfill the following requirements:

- The key vault must have purge protection enabled and be located in the same subscription and region as the target Cloudera environment.
- The CMK must be an RSA key with a size of 2048 bits.

  Note: The CMK must be an RSA key with a size of 2048 bits as Cloudera uses same key for all the resources and sizes other than 2048 bits are not supported by Azure for PostgreSQL.

- The number of Disk Encryption Set (DES) resources is limited to 1000 per region per subscription. In the present implementation, a single DES is created for each Cloudera environment, so this permits at most 1000 environments created in that region/subscription. The actual practical limit may be lower due to the limits set for other resource types.

You should also review the Azure-imposed restrictions for CMKs used for disk encryption, described in Server-side encryption of Azure Disk Storage: Restrictions in Azure documentation.

Using an existing vault and vault key

If you have an existing key vault in Key Vaults in your Azure Portal:

1. Navigate to the key vault's Overview page and verify that the following parameters are set to the correct values:
   - The Region matches the target region of the Cloudera environment. Storage accounts can be in different resource groups than the key vault, provided the location/region is the same.
   - The Purge protection is enabled.

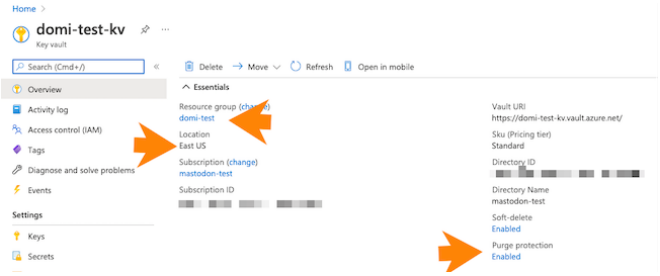

2. Next, navigate to the vault key and:
   - Make sure that it is an RSA key with a size of 2048 bits.
   - Copy the key identifier (which is a HTTPS URL) for the key that is created. You will need to provide it during Cloudera environment registration later.

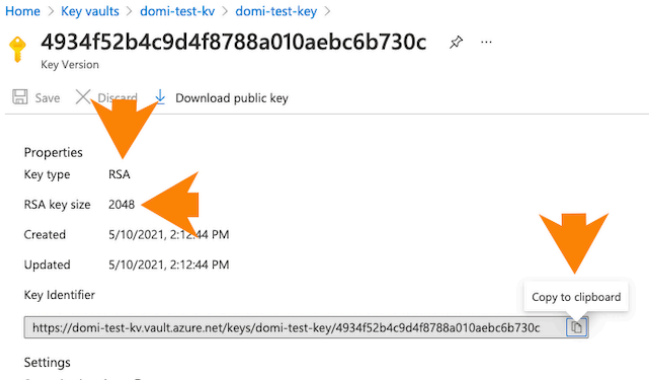

Creating a new vault and vault key

To create a vault and vault key, perform the following steps on your Azure portal:

1. Create a key vault in the same region and resource group as the one that you would like to use for registering the Cloudera environment.

   To create a key vault, navigate to Key vaults in Azure Portal and click on +New or on Create key vault. When providing key vault parameters, make sure that:

   - The Region matches the target region of the Cloudera environment. Storage accounts can be in different resource groups than the key vault, provided the location/region is the same.
   - The Purge protection is enabled.
   - Provide other parameters based on your organization's requirements. For instructions, see Create a vault in Azure documentation. Once done, click Create.

   The following screenshot points out these three important parameters when creating a new key vault:

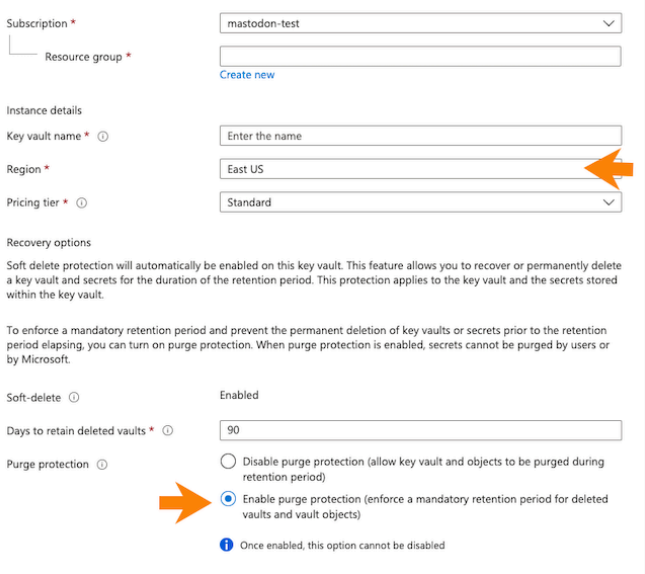

2. Generate or import a key in the previously created key vault. Make sure that it is an RSA key with a size of 2048, 3072 or 4096 bits.

   To generate a key, on your key vault's properties pages, select Keys and then click on Generate/Import:

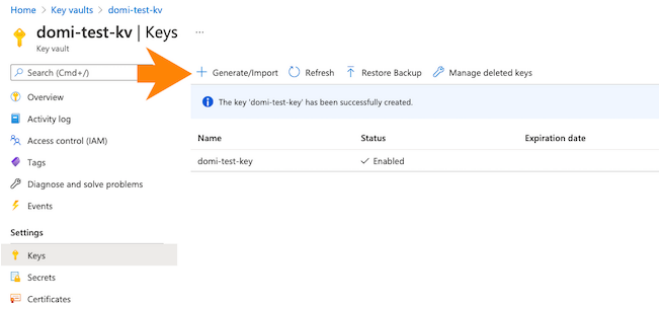

   Next provide key name and key type (make sure to select RSA). For detailed instructions, see Add a key to Key Vault in Azure documentation.

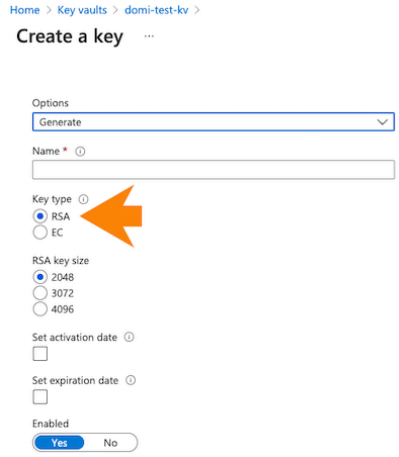

3. Once the key has been created, navigate to the vault key details and copy the key identifier (which is a HTTPS URL) for the key that is created. You will need to provide it during Cloudera environment registration later.

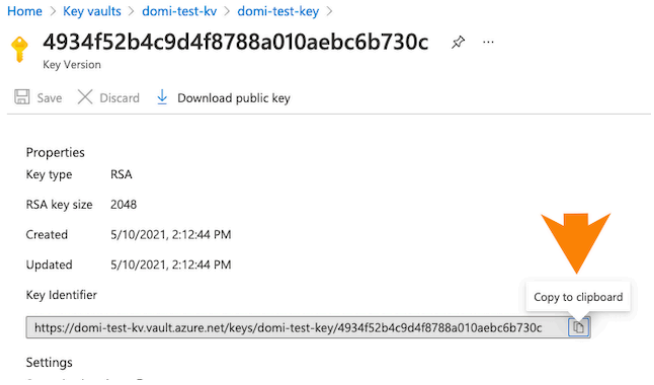

4. If you need encryption on the Azure storage account, you can set it up using Azure portal with the same or different encryption key. You can do this from the Encryption section of your storage account settings.

   Note: The key used for Azure storage encryption can be the same or different from the key vault used for disk encryption. Note that the storage account can be in a different resource group, but should be in the same location/region and subscription as the key vault.

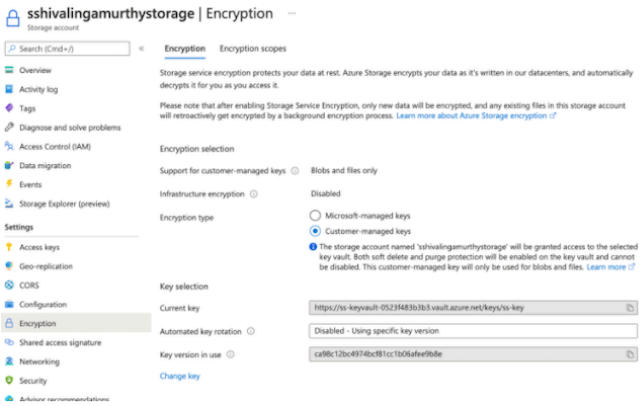

Managed identity for encrypting Azure Database for PostgreSQL Flexible Server

If you are using Azure Database for PostgreSQL Flexible Server with Cloudera, you can optionally use the CMK used for encrypting VM disks for encrypting the Azure Flexible Server database instance used by Cloudera.

In addition to the CMEK-related permissions described earlier, you should create a managed identity in Azure and assign to it an appropriate role with permissions that allow accessing the target key vault of the CMEK for cryptographic operations needed for storage encryption using that CMEK. In particular, the following dataActions shall be granted to the role:

```
"dataActions": [
"Microsoft.KeyVault/vaults/keys/read",
"Microsoft.KeyVault/vaults/keys/wrap/action",
"Microsoft.KeyVault/vaults/keys/unwrap/action"
],
```

There are two options:

- Using a built-in role provided by Azure:

  Key Vault Crypto Service Encryption User contains the bare minimum set of cryptographic permissions, although the Microsoft.EventGrid/* permissions are actually not needed. Alternatively, Key Vault Crypto User is also satisfactory, although it grants too many extra permissions on top of the ones that are strictly required.

- Using a custom role:

  We suggest cloning the Key Vault Crypto Service Encryption User and possibly removing the Microsoft.EventGrid/* permissions, which are not needed. Note that all the DataActions for Microsoft.KeyVault/* specified in this role are needed.

In either of the two cases, the role assignment of the managed identity should be scoped at the target Key Vault of the CMEK.

Note: If the managed identity for CMEK is absent during environment creation but the CMEK key URL is provided, the Disk Encryption Set (DES) resource is created using a system-assigned managed identity.

For information about providing the managed identity in Cloudera, see Configuring a CMK for data encryption in Azure Database for PostgreSQL Flexible Server.

Encrypting a storage account with a key vault that has role-based access control

To encrypt an ADLS Gen2 storage account that you would like to use with Cloudera with a key vault which has role-based access control set up, you need to perform the following steps on Azure Portal.

There are two scenarios described below:

- The first scenario assumes that you have not yet created the ADLS Gen2 storage account required for Cloudera on Azure.
- The second scenario assumes that you have already created the ADLS Gen2 storage account storage account required for Cloudera on Azure.

New storage account

The following steps should be performed in addition to the usual steps while creating the ADLS Gen2 account that you are planning to use with Cloudera.

Steps

1. Create a managed identity. Let's call it "key-vault-rbac".
2. Create a key vault.
   a. The key vault should have "purge protection" and "soft-delete" enabled.
   b. The key vault should be located in the same subscription and region as the target Cloudera environment.
   c. Set up the key's access policy to "Azure role-based access control".
3. Navigate to the access control for this key vault and:
   a. Click on Add role assignment.
   b. Assign the "Key Vault Administrator" role to all of the following users:
      - The user who created this key vault
      - The user(s) who register a Cloudera environment using this key vault
      - All the users/managed Identities who will be accessing this key vault

      Note: Without this step the key vault will not be accessible.
4. Once the key vault is created, create an RSA key with the size of 2048 bits.
5. Navigate to the access control for this key vault and:
   a. Click on Add role assignment.
   b. Assign the "Key Vault Crypto Service Encryption User" role to the "key-vault-rbac" managed identity created earlier. This will enable access to this key vault from the storage account.
6. Create the storage account and the managed Identities as mentioned in Minimal setup for cloud storage. During storage account creation, choose the following options to enable the customer managed key encryption for the storage account:
   a. Set Enable support for customer-managed keys to "All service types".

      Note: This is a mandatory step as this setting cannot be added or changed later.
   b. Set Identity type to "User-assigned".
   c. Set User-assigned identity to the "key-vault-rbac" managed identity created earlier.

Existing storage account

In case your ADLA Gen2 storage account already exists, perform the following steps instead of the ones above. The requirement is that the storage account must have been created with "Enable support for customer-managed keys" set

to "All service types". This cannot be set once the storage account exists, so if the storage account does not have this set, you cannot use it for this use case.

Steps

1. Create a managed identity. Let's call it "key-vault-rbac".
2. Navigate to the access control for the key vault that you are using for Cloudera and:
   a. Click on Add role assignment.
   b. Assign the "Key Vault Crypto Service Encryption User" role to the "key-vault-rbac" managed identity created earlier. This will enable access to this key vault from the storage account.
3. To enable the customer managed key encryption for the storage account used in Cloudera, the following options must be chosen during storage account creation:
   a. Verify that Enable support for customer-managed keys is set to "All service types".

      Note: This option can only be set during storage account creation and cannot be changed later.

   b. Set Identity type to "User-assigned".
   c. Set User-assigned identity to the "key-vault-rbac" managed identity created earlier.

Azure Files storage account and file share for Cloudera AI

If you would like to use the Cloudera AI data service, create Azure Files storage account and file share. Azure Files NFS v4.1 is a managed, POSIX compliant NFS service on Azure. The file share is used to store files for the Cloudera AI infrastructure and Cloudera AI workbenches.

See Create Azure Files Storage Account and File Share.

Azure Files NFS for Cloudera AI

If you would like to use the Cloudera AI data service, Azure Files NFS is the recommended NFS service.

If you would like to use Cloudera AI, you should create an Azure Files storage account and file share to store files for the Cloudera AI infrastructure and Cloudera AI workbenches, as described in Create Azure Files Storage Account and File Share.

Note: As an alternative to Azure Files NFS, you can configure an NFS server that is external to the Cloudera AI cluster. This is not the recommended approach. See Other NFS Options.

Azure quota limits

When you create your Azure Portal subscription, Azure Portal sets limits to the resources available to you. The limits can vary by region.

To register an Azure environment in Cloudera, you may need to increase some of these limits for the region(s) that you are planning to use. Cloudera creates resources such as VMs in your Azure subscription. For example:

- If you are planning to use the Cloudera AI service, review Limitations on Azure in Cloudera AI docs.
- Depending on the number of clusters that Cloudera creates in your Azure subscription, you might need to raise the limits for certain resources such as VMs and vCPUs in your Azure subscription. For a complete list of Azure resources used by Cloudera refer to Azure resources used by Cloudera.

If you require more resources than the limit set by Azure, you can create a support request on your Azure Portal. For information on Azure quotas, refer to Azure subscription and service limits, quotas, and constraints.

Overview of Azure resources used by Cloudera

The following Azure resources are used by Cloudera and Cloudera services.

Azure resources created for a Cloudera environment

When a Cloudera environment is created, a FreeIPA cluster and a Data Lake cluster are created.

The following Azure resources are created for FreeIPA (one per environment):

Resource	Description	Naming convention
Virtual Private Network (VNet)	If during environment creation you select to have a new VNet and subnets created, then a new VNet and subnets are created on your Azure account. Alternatively, you can provide your own existing VNet and subnets. In both cases (new and existing VPC), all compute resources that Cloudera provisions for the environment and Cloudera services are provisioned into the VNet specified during environment creation.	Specified by customer
Resource group for FreeIPA resources	If you chose for Cloudera to create multiple resource groups, a resource group is created to group all the resources created for FreeIPA. This resource group is not created if you chose to use a single existing resource group.	<env-name>-freeipa-<numeric-id>
Virtual Machines (VMs)	During environment creation, two or three Standard_DS3_v2 VMs are provisioned for the FreeIPA HA server by default. The number of VMs depends on the selected Data Lake type.	<env-name>-freeipa-<numeric-id>m0
OS disk	An OS disk is provisioned for the FreeIPA VM.	<env-name>-freeipa-<numeric-id>m0
Network interface	One network interface card (NIC) is provisioned for the FreeIPA VM.	<env-name>-freeipa-<numeric-id>m0
Public IP address	If you choose to use public IPs, your VM is assigned a public IP address.	<env-name>-freeipa-<numeric-id>m0
Network security group	Network security groups define inbound and outbound access to the instances.If during environment creation you choose to have new security groups created, then they are created on your Azure account. Alternatively, you can provide your own existing security groups.	master0-<env-name>freeipa<numeric-id>sg
ADLS Gen2 storage account for storing operating system images	By default, Cloudera creates an ADLS Gen2 storage account that is used solely for storing operating system images. If required, you can optionally pre-create this account and copy the required images.	The name is the concatenation of the following three elements: • “cbimg” • Region identifier: The starting letters of the region where the SA is. For example, “eu” for East US, or “eu2” for East US 2. • Subscription ID, without hyphens ('-') and all lowercase. For example, if your subscription ID is a9d4456e-349f-46f5-bc73-54a8d523e504,you should convert it to a9d4456e349f46f5bc7354a8d523e504. For example, the name of a storage account in East US 2 region in subscription a9d4456e-349f-46f5-bc73-54a8d523e504 would be cbimgeu2a9d4456e349f46f5bc7354a8d523e504.

Resource	Description	Naming convention
Resource group for the ADLS account used for storing operating system images	If you chose for Cloudera to create multiple resource groups, a separate resource group is created for the ADLS Gen2 account mentioned above. **Note:** To speed up future environment deployment, this resource group are its content are not deleted during environment deletion. This resource group is not created if you chose to use a single existing resource group.	cloudbreak-images

In addition, the following resources are created for each Data Lake (one per environment):

Resource	Description	Naming convention
Resource group	If you chose for Cloudera to create multiple resource groups, two resource groups are created: One new resource group is created to group all the resources created for the Data Lake and another resource group is created for the database used by the Data Lake. These resource groups are not created if you coose to use a single existing resource group.	<dl-name><numeric-id>
Virtual Machines (VMs)	VMs are provisioned for the Data Lake nodes: • Light duty: Two instances are provisioned: One Standard_D8s_v3 instance (Data Lake Master node) and one Standard_D2s_v3 instance (IDBroker). • Medium duty: Ten instances are provisioned: One Standard_D2s_v3 instance (IDBroker), three Standard_D4s _v3 instances (two Data Lake Master nodes and one Auxiliary node), and five Standard_D8s_v3 instances (three DataLake Core and two Gateway nodes).	<dl-name><numeric-id>[m0|i1]
OS disks	An OS disk is provisioned for each VM.	<env-name>-freeipa-<numeric-id>m0
Attached disks	An attached disk (StandardSSD_LRS) is provisioned for each VM.	<dl-name><numeric-id>-[m0|i1]<index>-<timestamp>
Network interface	One network interface card (NIC) is provisioned for each VM.	<dl-name><numeric-id>[m0|i1]
Public IP address	If you choose to use public IPs, each of the VMs is assigned a public IP address.	<dl-name><numeric-id>[m0|i1]
Network security groups	Network security groups define inbound and outbound access to the instances.If during environment creation you choose to have new security groups created, then they are created on your Azure account.	[master|idbroker]<dl-name><numeric-id>sg
Availability set	One availability set is created for the master host group only.	<dl-name>-master-as
Resource group for external DB	If you chose for Cloudera to create multiple resource groups, a resource group is created for the external database. This resource group is not created if you chooe to use a single existing resource group.	<env-name>-dbstck-<numeric-id>

Resource	Description	Naming convention
Azure Database for PostgreSQL server	An RDS instance is provisioned for Cloudera Manager, Ranger, and Hive MetaStore. When creating Flexible Server, Cloudera automatically chooses the latest generation of Standard_E4s instance family that is supported in the given region, for example, Standard_E4ds_v5, Standard_E4ds_v4 or Standard_E4s_v3 with 128 GB of storage. For more information, see Azure regions. If you choose to use Single Server, a database instance MO_Gen5_4 with 100 GB of storage) is provisioned.	dbsrv-<numeric-id>
ADLS Gen2 storage	Prior to registering your environment in Cloudera, you should create ADLS Gen2 storage containers as instructed in Cloudera documentation.	Specified by customer
Managed identities	Prior to registering your environment in Cloudera, you should create managed identities as instructed in Cloudera documentation.	Specified by customer

Azure resources created for Cloudera Data Hub

The following Azure resources are created for the Cloudera Data Hub service:

Resource	Description	Naming convention
Resource group	If you chose for Cloudera to create multiple resource groups, for each Cloudera Data Hub cluster, a new resource group is created to group all the resources created for the cluster. This resource group is not created if you chose to use a single existing resource group.	<dh-name><numeric-id>
Virtual Machines (VMs)	A VM is created for each cluster node. The VM type varies depending on what you selected during Cloudera Data Hub cluster creation. For a list of supported VM types, refer to Cloudera Public Cloud service rates.	<dh-name><numeric-id><hostgroup abbr.><node index>
OS disk	An OS disk is provisioned for each VM.	<dh-name><numeric-id>-osDisk<hostgroup abbr.><node index>
Attached Disks	An attached disk is provisioned for each VM, as specified during Cloudera Data Hub cluster creation. The disk size is selected during cluster creation.	<dh-name><numeric-id>-<hostgroup abbr.><node index>-<disk counter>-<timestamp>
Network interface	One network interface card (NIC) is provisioned for each VM.	<dh-name><numeric-id><hostgroup abbr.><node index>
Public IP address	If you choose to use public IPs, each of the VMs is assigned a public IP address.	<dh-name><numeric-id><hostgroup abbr.><node index>
Network security group	Network security groups define inbound and outbound access to the instances. If during environment creation you choose to have new security groups created, then they are created on your Azure account.	<hostgroup nm><dh-name><numeric-id>sg
Availability set	If the "Hardware and Storage" Advanced Options were used, one availability set is created for each host group. Otherwise, one availability set is created only for the host groups that contain KNOX and/or OOZIE service.	<dh-name>-<hostgroup>-as

Azure resources created for Cloudera Data Warehouse

The following Azure resources are created for the Cloudera Data Warehouse service:

Resource	Description
Resource Group	If you chose for Cloudera to create multiple resource groups, one resource group is created with the naming convention “<environment-id>-dwx-rg”. This resource group is not created if you chose to use a single existing resource group.
Azure Kubernetes Service (AKS)	Cloudera creates an AKS cluster for each activated Cloudera Data Warehouse environment to host Kubernetes-based resources. The underlying compute, network resources are managed by Azure, including: • Resource group • Virtual machine scale sets • Load balancer(s) • Public IP address(es) • Network security group • Disk(s) For a list of supported VM types, refer to Cloudera Public Cloud service rates.
Azure Database for PostgreSQL server	PostgreSQL database (General Purpose, Gen5, 4 vCore) is created for Cloudera Data Warehouse to store configuration data.

Azure resources created for Cloudera AI

The following Azure resources are created for the Cloudera AI service:

Resource	Description
Resource groups	If you chose for Cloudera to create multiple resource groups, one resource group is created with the naming convention “liftie-<unique string>" (which has an AKS cluster of the same name). This resource group is not created if you chose to use a single existing resource group.
Azure Kubernetes Service (AKS)	Cloudera creates an AKS cluster for each Cloudera AI workbench to host Kubernetes-based resources. The underlying compute, network resources are managed by Azure, including: • Resource group • Virtual machine scale sets • Load balancer(s) • Public IP address(es) • Route table • Network security group • Azure disk(s) (Premium_LRS) For a list of supported VM types, refer to Cloudera Public Cloud service rates.
Log analytics workspace	A logs analytics workspace is created for storing log data.
Azure Files Storage account	If you choose Azure Files NFS, you will need an existing Azure Files Storage account.

Azure resources created for Cloudera DataFlow

The following Azure resources are created for the Cloudera DataFlow service:

Resource	Description
Resource groups	If you chose for Cloudera to create multiple resource groups, one resource group is created with the naming convention “liftie-<unique string>" (which has an AKS cluster of the same name). This resource group is not created if you chose to use a single existing resource group.
Azure Kubernetes Service (AKS)	Cloudera creates an AKS cluster for the Cloudera DataFlow service. The underlying compute, network resources are managed by Azure, including: • Load balancer: Azure Load Balancer • Network security group • Public IP address(es) • Route table • Virtual machine scale set For a list of supported VM types, refer to Cloudera Public Cloud service rates.
Log analytics workspace	A logs analytics workspace is created for storing log data.
Azure Database for PostgreSQL	Azure Database for PostgreSQL is used for storing job-related metadata and histories.

Azure resources created for Cloudera Data Engineering

The following Azure resources are created for the Cloudera Data Engineering service:

Resource	Description
Resource groups	If you chose for Cloudera to create multiple resource groups, one resource group is created with the naming convention “liftie-<unique string>" (which has an AKS cluster of the same name). This resource group is not created if you chose to use a single existing resource group.
Azure Kubernetes Service (AKS)	Cloudera creates an AKS cluster for each Cloudera Data Engineering Service. The underlying compute, network resources are managed by Azure, including: • Load balancer: Azure Load Balancer • Network security group • Public IP address(es) • Route table • Virtual machine scale set For a list of supported VM types, refer to Cloudera Public Cloud service rates.
Log analytics workspace	A logs analytics workspace is created for storing log data.
Azure Files	Azure Files | Microsoft Azure contains job resources, application code, Apache Airflow DAG files and any other uploaded files.
Azure Database for MySQL Server	Azure Database for MySQL is used for storing job related metadata, histories.

Azure resources created for Cloudera Operational Database

The following Azure resources are created for the Cloudera Operational Database service:

Resource	Description
Resource Group	If you chose for Cloudera to create multiple resource groups, a resource group is created which contains all of the nodes that comprise the Cloudera Operational Database database. This resource group is not created if you chose to use a single existing resource group.

Resource	Description
Virtual Machines (VMs)	A compute VM is created for each node in a Cloudera Operational Database database. The instance type and managed storage are automatically determined by Cloudera Operational Database. Azure network security groups are automatically configured as a part of environment creation to define inbound and outbound network access to the created instances.
ADLS Gen2 storage	This existing blob storage account that you provided for the Data Lake to use for workload data storage is automatically used by the Cloudera Operational Database database for storage of data.

Azure outbound network access destinations

If you have limited outbound internet access (for example due to using a firewall or proxy), review this content to learn which specific outbound destinations must be available in order to register a Cloudera environment.

We recommend hostname-based policies, as some of the destination services do not have static IP addresses. IP address details in CIDR notation have been provided where static IPs are in-use.

Note:

If the cloud provider network that you would like to use for registering a Cloudera environment uses a custom DNS server that does not allow name resolution for public domain, you should add all the domains listed in the below tables to the DNS forwarder for name resolution.

Note:

On Azure, it is possible to define network security groups (NSGs) based on IP addresses, but not based on hostnames. At the same time, the egress filtering requirements documented here contain not only static IP addresses but also hostnames, where the IP address may change over time. This means that it is not possible to perform egress filtering using NSGs. Instead, if you would like to add outbound network access to your allow list based on hostnames, you should use Azure firewall with FQDN filtering in network rules. For that you can use either Azure hosted DNS or a custom DNS, as explained in Use FQDN filtering in network rules. Note that interfaces configured via Azure Private Link don't need to go through a proxy.

The following list includes general destinations as well as Azure-specific destinations.

General endpoints

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
Control Plane API	All services	US-based Control Plane: api.us-west-1.cdp.cloudera.com EU-based Control Plane: api.eu-1.cdp.cloudera.com AP-based Control Plane: api.ap-1.cdp.cloudera.com	HTTPS with Cloudera-generated access key	TCP/443	Cloudera Control Plane REST API.
Cloudera CCMv1 Persistent Control Plane connection	All services	*.ccm.cdp.cloudera.com 44.234.52.96/27	SSH public/ private key authentication	TCP/6000-6049	One connection per cluster configured; persistent

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
Cloudera CCMv2 Persistent Control Plane connection	All services	US-based Control Plane: *.v2.us-west-1.ccm.cdp.cloudera.com 35.80.24.128/27 EU-based Control Plane: *.v2.ccm.eu-1.cdp.cloudera.com 3.65.246.128/27 AP-based Control Plane: *.v2.ccm.ap-1.cdp.cloudera.com 3.26.127.64/27	HTTPS with mutual authentication	TCP/443	Multiple long-lived/persistent connections
Cloudera Databus Telemetry, billing and metering data	All services	US-based Control Plane: dbusapi.us-west-1.sigma.altus.cloudera.com api.us-west-1.cdp.cloudera.com https://cloudera-dbus-prod.s3.amazonaws.com EU-based Control Plane: api.eu-1.cdp.cloudera.com https://mow-prod-eu-central-1-sigmadbus-dbus.s3.eu-central-1.amazonaws.com https://mow-prod-eu-central-1-sigmadbus-dbus.s3.amazonaws.com AP-based Control Plane: api.ap-1.cdp.cloudera.com https://mow-prod-ap-southeast-2-sigmadbus-dbus.s3.ap-southeast-2.amazonaws.com https://mow-prod-ap-southeast-2-sigmadbus-dbus.s3.amazonaws.com	HTTPS with Cloudera-generated access key for dbus HTTPS for S3	TCP/443	Regular interval for telemetry, billing, metering services, and used for Cloudera Observability if enabled. Larger payloads are sent to a Cloudera managed S3 bucket.
Cloudera Observability Metrics System metrics collection	All services	US-based Control Plane: *.api.monitoring.us-west-1.cdp.cloudera.com EU-based Control Plane: *.api.monitoring.eu-1.cdp.cloudera.com AP-based Control Plane: *.api.monitoring.ap-1.cdp.cloudera.com	HTTPS	TCP/443	New as of March 2024
Cloudera Manager parcels Software distribution	All services	archive.cloudera.com	HTTPS	TCP/443	Cloudera's public software repository. CDN backed service; IP range not predictable.
RPMs Cloudera RPMs for workload agents	All services	cloudera-service-delivery-cache.s3.amazonaws.com	HTTPS	TPC/443	RPM packages for some workload components

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
Container Images Software Distribution	Cloudera Data Engineering Cloudera Data Warehouse Cloudera AI	container.repo.cloudera.com container.repository.cloudera.com container.repo.cloudera.com prod-us-west-2-starport-layer-bucket.s3.us-west-2.amazonaws.com prod-us-west-2-starport-layer-bucket.s3.amazonaws.com s3-r-w.us-west-2.amazonaws.com *.execute-api.us-west-2.amazonaws.com prod-eu-west-1-starport-layer-bucket.s3.eu-west-1.amazonaws.com prod-eu-west-1-starport-layer-bucket.s3.amazonaws.com s3-r-w.eu-west-1.amazonaws.com *.execute-api.eu-west-1.amazonaws.com prod-ap-southeast-1-starport-layer-bucket.s3.ap-southeast-1.amazonaws.com prod-ap-southeast-1-starport-layer-bucket.s3.amazonaws.com s3-r-w.ap-southeast-1.amazonaws.com *.execute-api.ap-southeast-1.amazonaws.com	HTTPS	TCP/443	CDN-backed and AWS ECR-backed services; IP range not predictable. container.repo.cloudera.com uses ECR backend which requires S3 URLs. IP geolocation attempts to select closest API and ECR backend; clients may be directed to any of the destinations.
Flow Definitions Cloudera AWS bucket with flow definitions	Cloudera DataFlow	US-based Control Plane: s3.us-west-2.amazonaws.com/dfx-flow-artifacts.mow-prod.mow-prod.cloudera.com EU-based Control Plane: cldr-mow-prod-eu-central-1-dfx-flow-artifacts.s3.eu-central-1.amazonaws.com AP-based Control Plane: cldr-mow-prod-ap-southeast-2-dfx-flow-artifacts.s3.ap-southeast-2.amazonaws.com	HTTPS (one way) IAM authentication	TCP/443	Outbound internet access to S3 hosts is necessary on all cloud providers when using Cloudera DataFlow as the workload needs to query outbound to an S3 location to retrieve the flow definition when creating a deployment.
Public Signing Key Retrieval	Cloudera Data Engineering Cloudera DataFlow	US-based Control Plane: consoleauth.altus.cloudera.com console.us-west-1.cdp.cloudera.com EU-based Control Plane: console.eu-1.cdp.cloudera.com AP-based Control Plane: console.ap-1.cdp.cloudera.com	HTTPS	TCP/443	Required to allow authentication to Cloudera Data Engineering virtual Cluster using a Cloudera Access Key.

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
Control Plane IAM API	Cloudera AI	US-based Control Plane: iamapi.us-west-1.altus.cloudera.com console.us-west-1.cdp.cloudera.com EU-based Control Plane: console.eu-1.cdp.cloudera.com AP-based Control Plane: console.ap-1.cdp.cloudera.com	HTTPS	TCP/443	For connecting to the IAMAPI for fetching the entitlement details.
AMPs Cloudera Accelerators for Machine Learning Projects	Cloudera AI	https://raw.githubusercontent.com https://github.com	HTTPS	TCP/443	Files for AMPs are hosted on GitHub.
Learning Hub	Cloudera AI	https://github.com/cloudera/learning-hub-content	HTTPS	TCP/443	Access Learning Hub in air-gapped environments

Azure-specific endpoints

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
General Azure guidelines	All services	See Safelist the Azure portal URLs on your firewall or proxy server for Azure egress best practices.			
Azure Data Lake Storage Gen 2	All services	<STORAGE_ACCOUNT_NAME>.dfs.core.windows.net	HTTPS Azure authentication	TCP/443	Azure Storage VPC endpoint is required (Microsoft.Storage).Replace the <STORAGE_ACCOUNT_NAME> with an actual storage account name.
Azure Database for Postgres	All services	*.postgres.database.azure.com	JDBC / Postgres binary protocol	TCP/5432	Azure SQL VPC endpoint is required (Microsoft.Sql).
ARM to manage User Assigned Managed Identities	All services	management.azure.com	HTTPS Azure authentication	TCP/443	This can be allowed by using the AzureResourceManager Azure service tag. Additionally IP addresses to whitelist are available to download.
Microsoft Log Analytics	All services	*.agentsvc.azure-automation.net *.ods.opinsights.azure.com *.oms.opinsights.azure.com *.blob.core.windows.net	HTTPS Azure authentication	TCP/443	Optional, but may cause issues with Azure approved images if blocked.

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
Azure Kubernetes Services (AKS)	Cloudera Data Engineering Cloudera DataFlow Cloudera Data Warehouse Cloudera AI	See Outbound network and FQDN rules for Azure Kubernetes Service (AKS) clusters. **Note:** Cloudera uses AKS and has the same requirements. Therefore, your setup must fulfill the Azure EKS requirements mentioned in the linked documentation (such as port 9000). If you skip this step, your deployment will fail.			
Azure Database for MySQL	Cloudera Data Engineering	*.mysql.database.azure.com	JDBC / Postgres binary protocol	TCP/3306	Azure Database for MySQL
Azure files	Cloudera Data Engineering	*.file.core.windows.net	SMB	TCP/445	What is Azure Files?
Azure Files NFS	Cloudera AI	*.file.core.windows.net	NFS	TCP/2049	Create an NFS Azure file share
Digicert CA Certificate	Cloudera Data Engineering Cloudera DataFlow	www.digicert.com cacerts.digicert.com	HTTPS Azure authentication	TCP/443	Fetching TLS CA for Azure MySQL DB secure connection

Access to workload UIs

If you have restricted DNS or networking setup, make sure that *.cloudera.site is resolvable from your network so that members of your organization can access workload UIs.

Cloudera workloads (including Data Lake) use subdomains under cloudera.site to host various UI endpoints (Cloudera Manager, Ranger, Knox, Hue and so on). Cloudera automatically provisions these endpoints whenever a Data Lake, Cloudera Data Hub or another type of workload (for example, Virtual Warehouse in Cloudera Data Warehouse) is created, and routing is set up so that you can access these endpoints from your network.

The subdomains are assigned under cloudera.site using the following convention:

```
<endpoint-name>.<env-truncated-name>.<customer-workload-subdomain>.<regional
-subdomain>.cloudera.site
```

Supported browsers

Cloudera validates and tests against the latest version and supports recent versions of the following browsers:

- Google Chrome
- Mozilla Firefox

  Note:
  - Mozilla Firefox is not supported by Cloudera Data Engineering.
  - Certain accessibility features in Cloudera DataFlow do not work in Mozilla Firefox.
- Safari
- Microsoft Edge

Other resources

While this document attempts to provide a complete overview of cloud provider requirements, there is additional documentation that you should review if planning to deploy Cloudera data services.

The following table includes links to documentation that you should review if planning to deploy the respective Cloudera data service:

Cloudera data service	Documentation links
Cloudera Data Engineering	Cloudera Data Engineering Azure prerequisites Cloudera Data Engineering cost management Cloudera Data Engineering performance management
Cloudera Data Warehouse	Virtual Warehouse sizing requirements for public cloud environments Virtual Warehouse IP address and cloud resource requirements for Public Cloud environments Managing costs in the public cloud environments for Cloudera Data Warehouse Azure environments requirements checklist
Cloudera DataFlow	Azure requirements for Cloudera DataFlow Azure load balancers in Cloudera DataFlow Cloudera DataFlow limitations on Azure
Cloudera AI	Azure requirements for Cloudera AI workbenches Cloudera AI limitations on Azure Network Planning for Cloudera AI on Azure
Cloudera Operational Database	Cloudera Operational Database cloud checklist Azure requirements for Cloudera Operational Database deployment

Cloudera CIDR

List of Cloudera Control Plane source IP addresses

In cases where the Cloudera Control Plane is communicating directly with resources in the Cloudera environment, the Cloudera Control Plane source IP address will be from the ranges below.

Security groups deployed/managed by Cloudera Control Plane will automatically include the required rules. These source IP ranges will be included if the Cloudera Control Plane requires direct inbound access to the resource.

Cloudera Control Plane Region	IP Ranges
us-west-1	35.80.24.128/27, 35.166.86.177/32, 52.36.110.208/32, 52.40.165.49/32
eu-1	3.65.246.128/27
ap-1	3.26.127.64/27

If the security groups (or other network access policies) in the network path are not managed by the Cloudera Control Plane, ensure that inbound access from the appropriate range(s) are permitted.

Cloudera Public Cloud

GCP Requirements

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

GCP requirements 4

GCP permissions 4

GCP resources and services 4

GCP project 4
GCP APIs 4
GCP region 5
Supported GCP regions 5
VPC network and subnet 6
Internet connectivity 6
Firewall rules 7
Managed service network connection 8
VM instances 9
Custom images 9
Service account for credential 9
Required permissions 9
Create service account 13
GCP cloud storage prerequisites 14
Minimum setup for GCP cloud storage 14
Onboarding Cloudera users and groups 23
Storage bucket for OS images 24
Supported GCP block storage 24
SSH key pair 24
Customer managed encryption keys 24
CMEK requirements 25
Create key ring and encryption key 25
Assign the required permissions to the encryption key 27
GCP limits 28
List of GCP resources 28

GCP outbound network destinations 29

Access to workload UIs 32

Supported browsers 32

Cloudera CIDR 32

GCP requirements

Use the following guidelines to ensure that your Google Cloud account has all the necessary resources required by Cloudera and that Cloudera can access these resources:

Related Information

GCP permissions

GCP resources and services

Overview of GCP resources used by Cloudera

GCP outbound network access destinations

Access to workload UIs

Supported browsers

GCP permissions

As an administrator, you must be able to create and manage the resources in the Google Cloud project where Cloudera environments and clusters run. You must be able to perform all administrative tasks and have administrative rights to all resources.

Cloudera recommends that the administrator has the role of Owner in the project used for Cloudera in your GCP account.

GCP resources and services

Cloudera uses the following resources in your Google Cloud account.

Use the following guidelines to ensure that your GCP account has all the necessary resources required by Cloudera and that Cloudera can access these resources:

GCP project

In order to deploy Cloudera resources, you must create a project on your Google Cloud account.

If you would like to use a shared VPC, the VPC from the host project needs to be shared with the newly created project.

If you need to create a project, refer to Creating a project in Google documentation.

GCP APIs

Review the following list of GCP APIs and make sure that they are enabled for the project used for Cloudera.

The following Google APIs must be enabled for the project:

- iamcredentials.googleapis.com
- iam.googleapis.com
- compute.googleapis.com
- storage.googleapis.com
- servicenetworking.googleapis.com
- sqladmin.googleapis.com

GCP region

When registering a GCP environment in Cloudera, you must specify the same GCP region as the one where your VPC and subnets are located. Cloudera recommends that the buckets used for storage and logs are also located in the same region as the VPC.

When registering a GCP environment in Cloudera, you should also select a specific availability zone within the selected region.

Supported GCP regions

Cloudera supports all Google Cloud regions.

Region name	Region ID	Environment	Cloudera Data Hub	Cloudera Operational Database
AMERICAS				
OREGON	us-west1	##	##	##
LOS ANGELES	us-west2	##	##	##
SALT LAKE CITY	us-west3	##	##	##
LAS VEGAS	us-west4	##	##	##
IOWA	us-central1	##	##	##
SOUTH CAROLINA	us-east1	##	##	##
N. VIRGINIA	us-east4	##	##	##
MONTRÉAL	northamerica-northeast1	##	##	##
SÃO PAULO	southamerica-east1	##	##	##
DALLAS	us-south-1	##	##	##
COLUMBUS	us-east5	##	##	##
EUROPE				
LONDON	europe-west2	##	##	##
BELGIUM	europe-west1	##	##	##
NETHERLANDS	europe-west4	##	##	##
ZURICH	europe-west6	##	##	##
FRANKFURT	europe-west3	##	##	##
FINLAND	europe-north1	##	##	##
MILAN	europe-west8	##	##	##
PARIS	europe-west9	##	##	##
MADRID	europe-southwest1	##	##	##
ASIA PACIFIC				
MUMBAI	asia-south1	##	##	##
SINGAPORE	asia-southeast1	##	##	##
JAKARTA	asia-southeast2	##	##	##
HONG KONG	asia-east2	##	##	##
TAIWAN	asia-east1	##	##	##
TOKYO	asia-northeast1	##	##	##

Region name	Region ID	Environment	Cloudera Data Hub	Cloudera Operational Database
OSAKA	asia-northeast2	##	##	##
SYDNEY	australia-southeast1	##	##	##
SEOUL	asia-northeast3	##	##	##
MELBOURNE	australia-southeast2	##	##	##
MIDDLE EAST				
Qatar Doha	me-central1	##	##	##
KSA	me-central2	##	##	##
TEL AVIV	me-west-1	##	##	##

When making a decision regarding what region to use, you should review Cloud locations in Google documentation and verify that the region that you would like to use has the compute instance types and storage options that meet your requirements.

VPC network and subnet

When registering a GCP environment in Cloudera, you must provide an existing VPC network. Cloudera will not create a VPC network for you.

The VPC must fulfill the following requirements:

- The VPC network must have one or more subnets in a single geographic region. All the subnets used for environment creation must be in the same geographic region.
- Your VPC should reside in the same region as the buckets used for storage and logs. This allows you to avoid latency and additional data transfer costs.
- A shared VPC can be used.
- If you would like to use a shared VPC, the VPC from the host project needs to be attached to the newly created project.
- When creating a VPC and subnets, there is an option to create subnets in a custom or automatic mode. Cloudera recommends using the custom mode.
- If you would like to use Public Endpoint Access Gateway, make sure that "Private Google Access" is disabled on at least one subnet in the VPC.

For instructions on how to create and manage VPC networks and subnets in GCP, refer to Using VPC networks in Google documentation.

Related Information

Provisioning Shared VPC

Internet connectivity

Depending on your network deployment scenario, Cloudera on GCP requires outbound internet connectivity and may additionally require inbound connectivity with public IP assignment.

The following is required, depending on whether or not you choose to use Cluster Connectivity Manager:

- With Cluster Connectivity Manager: Works with private IP address assignment and requires a NAT gateway to be provisioned. Note that NAT gateway creation will require a cloud router creation, as shown in the following screenshot:

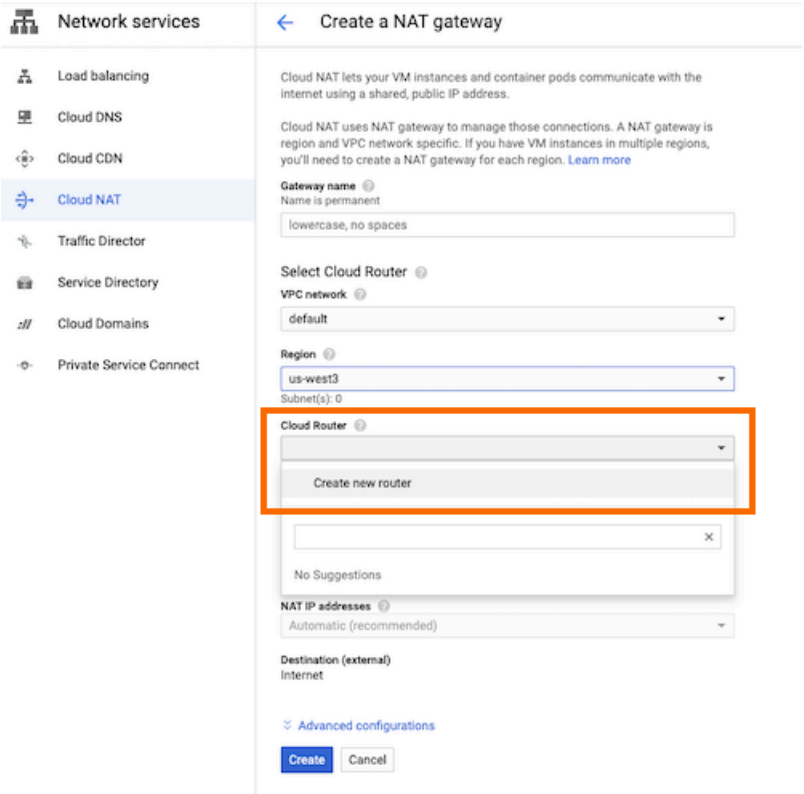

- Without Cluster Connectivity Manager: Works with public IP address assignment and requires specific firewall rule configuration (as described in Firewall rules).

If you are planning to use Cluster Connectivity Manager, review the following GCP documentation for your reference:

- Cloud NAT Overview
- Using Cloud NAT

Related Information

Cluster Connectivity Manager

Firewall rules

Cloudera requires that you pre-create a set of firewall rules allowing your organization SSH and UI access to Cloudera and allowing internal communication between Cloudera components. Cloudera does not offer an option to create these firewall rules for you.

You have two options:

Option	VPC type supported for this option	What to do during environment registration
• You create all required firewall rules at the VPC level.	Per project VPC Shared VPC	In this case, you do not provide them to Cloudera during environment registration (That is, during environment registration you select "Do not create firewall rule").

Option	VPC type supported for this option	What to do during environment registration
• You create the intravpc firewall rule at the VPC level. • Then, you create firewall rules for SSH and UI access via the security access mechanism in the Google Cloud UI. • If you need to create additional firewall rules (for example if you are not planning to use Cluster Connectivity Manager and you need to open ports 9443 and 443 for Cloudera), you should create these at the VPC level.	Per project VPC	In this case, you should select the firewall rules created for SSH and UI access during environment registration

Firewall rule requirements

You can add firewall rules in Google Cloud directly at the VPC level or via the security access control mechanism from VPC network > Firewall > Create firewall rule. For instructions on how to create and manage firewall rules in GCP, refer to Using firewall rules in Google documentation.

The firewall rules that you add should:

- Allow the instances in the VPC to connect with each other using TCP and UDP protocols on any port. To achieve this, add a TCP/UDP rule that is set to the subnet IP range. This is required for internal communication within the VPC. As an example, see the intravpcconnection firewall rule, which is set to the subnet IP range (10.0.0.0/16) in the following screenshot:

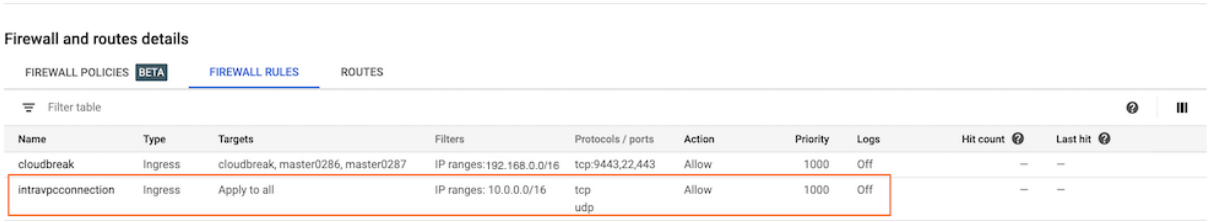

- Open TCP ports 22 and 443 to allow access from your organization's CIDR.
- If not using Cluster Connectivity Manager, also open TCP port 9443 to allow access from Cloudera CIDR.
- If not using Cluster Connectivity Manager, also open TCP port 443 to allow access from Cloudera CIDR. This is required for the gateway nodes.
- Open TCP/UDP ports 0-65535 to your VPC's CIDR (for example 10.10.0.0/16) and your subnet's CIDR (for example 10.0.2.0/24).
- Allow ICMP traffic from your internal VPC CIDR (for example 10.10.0.0/16).

Note: The communication via TCP/UDP 0-65535 and ICMP is essential for healthy operation of CDP environments, Data Hubs, and data services running within the , so ensure that you open these ports as described below. While some services only need well-known fixed ports, a majority of them depend on ephemeral (i.e. dynamically or randomly allocated) ports; This is why the wildcard 0-65535 TCP/UDP port range is used in the absence of a detailed breakdown of individual ports. Since overall access to the is typically secured by other means, the use of the wildcard rules does not pose a higher risk against external attacks.

Related Information

Cluster Connectivity Manager

Managed service network connection for CloudSQL

A CloudSQL database is created for the Data Lake cluster for external storage.

In order to use CloudSQL database with a private IP, your VPC needs to have private services access for CloudSQL. Private services access is implemented as a VPC peering connection between your VPC network and the underlying Google services VPC network where your CloudSQL instance resides. For more information about the setup and how to steps refer to Configuring private services access in Google documentation.

VM instances

Cloudera provisions VM instances as part of environment creation process (for Data Lake and FreeIPA) and for compute clusters.

Therefore, you should verify the limits on the number and type of VM instances in your GCP account to ensure that you are able to provision an environment and create clusters in Cloudera.

Custom images

By default Cloudera provides a set of default images that are used for all provisioned VMs, but you can optionally use custom images for Data Lake, FreeIPA, and Cloudera Data Hub.

You might require a custom image for compliance or security reasons (a “hardened” image), or to have your own packages pre-installed on the image, for example monitoring tools or software.

If you would like to use custom images instead of the default images, refer to Custom images and image catalogs.

Service account for the provisioning credential

The provisioning credential for Google Cloud relies on a service account that can be assumed by Cloudera.

The following flow describes how the Google Cloud provisioning credential works:

1. Your GCP account administrator creates a service account and assigns the minimum permissions allowing Cloudera to create and manage resources in your Google Cloud account. Next, the administrator generates a service account access key pair for the service account.
2. The service account is registered as a credential in Cloudera and its access key is uploaded to Cloudera.
3. The credential is then used for registering your Google Cloud environment in Cloudera.
4. Once this is done, Cloudera uses the credential for provisioning environment-related resources, workload clusters, and resources for other Cloudera services that you run in Cloudera.

Review the following to learn about the permissions required for the credential and how to create the service account.

Permissions for the provisioning credential's service account

To allow Cloudera to access and provision resources in your Google Cloud project, you should create a service account in your Google Cloud project, assign the following roles or granular permissions. Next, you generate a JSON access key that can later be provided to Cloudera. Cloudera will assume this service account via the service account access key provided during credential creation for provisioning resources for your environment.

The service account must fulfill one of the following requirements (choose one of the options):

- Option 1: Assign the following IAM roles at the project level. This is a simpler option.
- Option 2: Alternatively, you can create custom IAM roles with the following granular IAM permissions assigned and then assign the role to the service account at the project level. This allows you to minimize the number of permissions granted to Cloudera.

Option 1: IAM roles

IAM role	Scope	Description
iam.serviceAccounts.list IAM permission	Project	This is required in order for Cloudera to be able to list service account names that you created in your GCP project. You need to create a custom role in order to assign this permission.
Compute Instance Admin (v1) roles/compute.instanceAdmin.v1	Project	This is required for provisioning of Compute Engine instances, disks, and images in your VPC.
Storage Admin roles/storage.admin	Project	This is required for the creation of a storage bucket to store the Cloudbreak image objects. Delete permissions are not required.
Compute Network Viewer roles/compute.networkViewer	Project	This is required for read-only access to all networking resources.
Compute Load Balancer Admin roles/compute.loadBalancerAdmin	Project	This role is required for load balancing between HA components of the Data Lake.
Cloud SQL Admin roles/cloudsql.admin	Project	This is required in order for Cloudera to have the permission for creating and deleting a Data Lake and and heavy duty flow management Cloudera Data Hub clusters cleanly.
Compute Network User roles/compute.networkUser	Project	Required for shared VPC only If you would like to use a shared VPC, you need this additional role in the scope of the host project of the VPC.
Compute Public IP Admin roles/compute.publicIpAdmin	Project	Required only when not using Cluster Connectivity Manager This additional role is only required if you are planning to disable Cluster Connectivity Manager for your environment.

Note: Additionally, once you create the Logger and IDBroker service accounts discussed in the minimum setup for cloud storage, you need to update each of these two service accounts to grant the provisioning service account the Service Account User (iam.serviceAccountUser) role. See instructions provided as part of Minimum setup for cloud storage.

Option 2: Granular permissions

You should create a custom IAM role to assign these permissions.

Granular IAM permissions	Scope	Description
iam.serviceAccounts.list	Project	This is required in order for Cloudera to be able to access service accounts that you created.

Granular IAM permissions	Scope	Description
iam.serviceAccounts.list cloudsql.instances.create cloudsql.instances.delete cloudsql.instances.get Cloudsql.instances.list cloudsql.databases.update cloudsql.instances.startReplica cloudsql.instances.stopReplica cloudsql.instances.update cloudsql.instances.restart cloudsql.users.create	Project	Required for creating, stopping, starting, and deleting an external database for the Data Lake and Data Hub clusters.
compute.addresses.get compute.addresses.use compute.disks.create compute.disks.delete compute.disks.setLabels compute.disks.use compute.firewalls.list compute.globalOperations.get compute.images.create compute.images.get compute.images.list compute.images.useReadOnly compute.instances.create compute.instances.delete compute.instances.get compute.instances.list compute.instances.setLabels compute.instances.setMetadata compute.instances.setServiceAccount compute.instances.setTags compute.instances.start compute.instances.stop compute.machineTypes.list compute.networks.get compute.networks.list compute.regionHealthChecks.useReadOnly compute.regionOperations.get compute.regions.get compute.regions.list compute.subnetworks.get compute.subnetworks.list compute.subnetworks.use compute.subnetworks.useExternalIp compute.zoneOperations.get	Project	Required for creating VMs from images in your VPC.

Granular IAM permissions	Scope	Description
compute.addresses.create compute.addresses.delete compute.addresses.get compute.addresses.use compute.instanceGroups.create compute.instanceGroups.delete compute.instanceGroups.get compute.instanceGroups.list compute.instanceGroups.update compute.instanceGroups.use compute.forwardingRules.create compute.forwardingRules.delete compute.forwardingRules.get compute.forwardingRules.list compute.forwardingRules.setLabels compute.forwardingRules.update compute.forwardingRules.use compute.regionBackendServices.create compute.regionBackendServices.delete compute.regionBackendServices.get compute.regionBackendServices.list compute.regionBackendServices.update compute.regionBackendServices.use compute.regionHealthChecks.create compute.regionHealthChecks.delete compute.regionHealthChecks.get compute.regionHealthChecks.list compute.regionHealthChecks.update compute.regionHealthChecks.use	Project	Required for load balancing between HA components of the Data Lake.
compute.addresses.create compute.addresses.delete compute.addresses.get compute.addresses.use	Project	(Optional) Only required if public IPs are used. You do not need these permissions if you would like to use private IPs only.
storage.buckets.create storage.buckets.get storage.buckets.getIamPolicy storage.objects.create storage.objects.delete storage.objects.get storage.objects.getIamPolicy	Project	(Optional) This is not required if you are planning to pre-create the GCS bucket for storing OS images for VMs. By default, Cloudera creates this bucket, but you can optionally pre-create it. See Storage bucket for OS images.

Note: Additionally, once you create the Logger and IDBroker service accounts discussed in the minimum setup for cloud storage you need to update each of these two service accounts to grant the provisioning service account the Service Account User (iam.serviceAccountUser) role. See instructions provided as part of Minimum setup for cloud storage.

For instructions on how to create the service account, refer to the following documentation:

Create provisioning credential's service account and generate access key

Create a service account and generate a JSON access key.

Before you begin

Review the above permissions to learn what IAM permissions and IAM roles you need to assign to the service account that you will create.

Steps

1. Log in to your Google Cloud account.
2. Navigate to the project used for Cloudera.
3. Navigate to the IAM & Admin.
4. To create a custom role:
   a. Navigate to the Roles page.
   b. Click +Create Role.
   c. Specify a Title.
   d. Specify an ID.
   e. Click +Add Permissions.
   f. Add the required granular permission(s).
   g. Use the same steps to add all the required permissions.

      Note: If you are using the Option 1: IAM roles, you only need to assign the iam.serviceAccounts.list permission. If you are using the Option 2: Granular permissions, you need to assign all the permissions listed in the table listing the permissions.
   h. Click Create.
5. To create a service account:
   a. Navigate to the Service accounts page.
   b. Click Create service account.
   c. Enter a service account name.
   d. Click Create.
   e. Under Grant this service account access to project, choose the IAM roles to grant to the service account on the project. You need to assign all of the roles listed in the table.
   f. When you are done adding all the required roles, click Done to finish creating the service account.
6. To generate an access key:
   a. Once your account has been created, find the row of the service account that you want to create a key for. In that row, click the ⋮ (context menu) button, and then click Create key.
   b. Under Key type, select JSON and click Create.
   c. Clicking Create downloads the service account key file. You will use the JSON access key to register the service account as a credential in Cloudera.

      Warning: After you download the key file, you cannot download it again.
7. Additionally, once you create the Logger and IDBroker service accounts, you need to update each of these two service accounts to grant the provisioning service account the Service Account User (iam.serviceAccountUser) role. The instructions are provided as part of Minimum setup for cloud storage.

What to do next

Once you have this setup ready, you can Register a GCP credential in Cloudera.

Storage buckets and service accounts for logs, backups, and data storage

Cloudera requires that you pre-create and provide buckets for logs and data storage and create service accounts controlling access to them.

You should create two Google storage buckets:

- One for data storage
- One for logs
- Optionally, you can also create a third bucket for storing FreeIPA and Data Lake backups

Note: It is possible to use a single bucket. If you choose to do so, you must adjust service account permissions accordingly. This scenario is not covered in this documentation.

The buckets should fulfill the following requirements:

- For best performance, create the buckets in the same region as the VPC.
- If you would like to use encryption, use a Google-managed key.

In addition to the two Google storage buckets, you should create multiple service accounts and assign roles as described in the following documentation:

Minimum setup for GCP cloud storage

The minimal setup recommended for production includes two GCS buckets (one for storing workload data and another for storing logs) and four service accounts. Additionally, you can create a third bucket for storing FreeIPA and Data Lake backup data separately. If the third bucket is not provided, FreeIPA and Data Lake backup data is stored in the Logs bucket.

Note: You may choose a different setup. For example, for getting started with a test environment you may want to use a single GCS bucket. Just note that such setup is not covered in this documentation.

You need to create service accounts mentioned in the table and while creating them:

- The Service account name column lists all the service accounts that need to be created. You may choose different service account names. The ones provided here follow the same terminology as Cloudera web interface and CDP CLI making it easier to understand where to provide them to Cloudera.
- The Required IAM roles column explains what IAM role each service account needs over the item listed in the Scope column. For example, the Logger service account requires that you create a custom role with storage.buckets.get and storage.objects.create permissions. Next, you navigate to the Logs bucket permissions and add the Logger service account as a member with the custom role that you created earlier.

Service account name	Description	Required IAM roles	Scope
Logger	This service account will be assigned to all the workload instances in Cloudera. It will be used by Cloudera to: • Write telemetry logs to the Logs bucket. • Write FreeIPA backups to the Backups bucket or, if there is no designated bucket provided for backups, write to the Logs bucket.	A custom role with the following permissions: • storage.buckets.get • storage.objects.create • If you would like to use a bucket path (gs://<bucket>/<path>) instead of a bucket (gs://<bucket>) for the Logs or Backups bucket, you should also assign the storage.objects.list permission. When using lower versions than Cloudera Public Cloud 7.2.17, you need to create a custom role with the following permissions: • storage.buckets.get • storage.objects.create • storage.objects.get • storage.objects.list When using Cloudera Public Cloud 7.2.17 or higher versions, you need to create a custom role with the following permissions: • storage.buckets.get • storage.objects.create	Logs bucket and Backups bucket (if created)
Data Lake Admin	This service account will be used by Cloudera services to access workload data. It provides full access to the data storage location.	Storage Admin (roles/storage.admin) IAM role Alternatively, you can create a custom role and assign the following permissions: • storage.buckets.get • storage.objects.create • storage.objects.delete • storage.objects.get • storage.objects.list	Data storage bucket For Data Lake backup and restore: Backups bucket, if different from the main data storage bucket

Service account name	Description	Required IAM roles	Scope
Ranger Audit	This service account will be used by Cloudera to write Ranger audits to the storage bucket.	Storage Object Admin (roles/storage.objectAdmin) IAM role Alternatively, you can create a custom role and assign the following permissions: • storage.buckets.get • storage.objects.create • storage.objects.delete • storage.objects.get • storage.objects.list For Data Lake backup and restore, create a custom role and assign the following permissions: • storage.buckets.get • storage.objects.create • storage.objects.delete • storage.objects.get • storage.objects.list • storage.objects.getIamPolicy • storage.objects.setIamPolicy • storage.objects.update • resourcemanager.projects.get	Data storage bucket For Data Lake backup and restore: Backups bucket, if different from the main data storage bucket
Other service account for data access by users	Depending on your requirements, you may want to create a set of service accounts for data access by different user groups. For example, you may want to have one service account to assign to data science users and another service account for data engineering users.	Depending on your requirements, you should assign a custom role or a predefined role from Cloud Storage roles > Predefined roles on the bucket used for data storage.	Data storage bucket
IDBroker	This service account will be used by Cloudera to assume the other service accounts.	Workload Identity User (roles/iam.workloadIdentityUser) IAM role Alternatively, you can create a custom role and assign the following permissions: • iam.serviceAccounts.getAccessToken • iam.serviceAccounts.actAs	Service accounts (All of the above service accounts except Logger)
		Additionally, assign the same permissions as those assigned to the Logger service account.	Logs bucket

The following diagram illustrates the relationships between service accounts and buckets and between the IDBroker service account and other service accounts. The dotted arrows signify which entity needs access to what. For example, the Data Lake admin role must be able to access the Logs bucket:

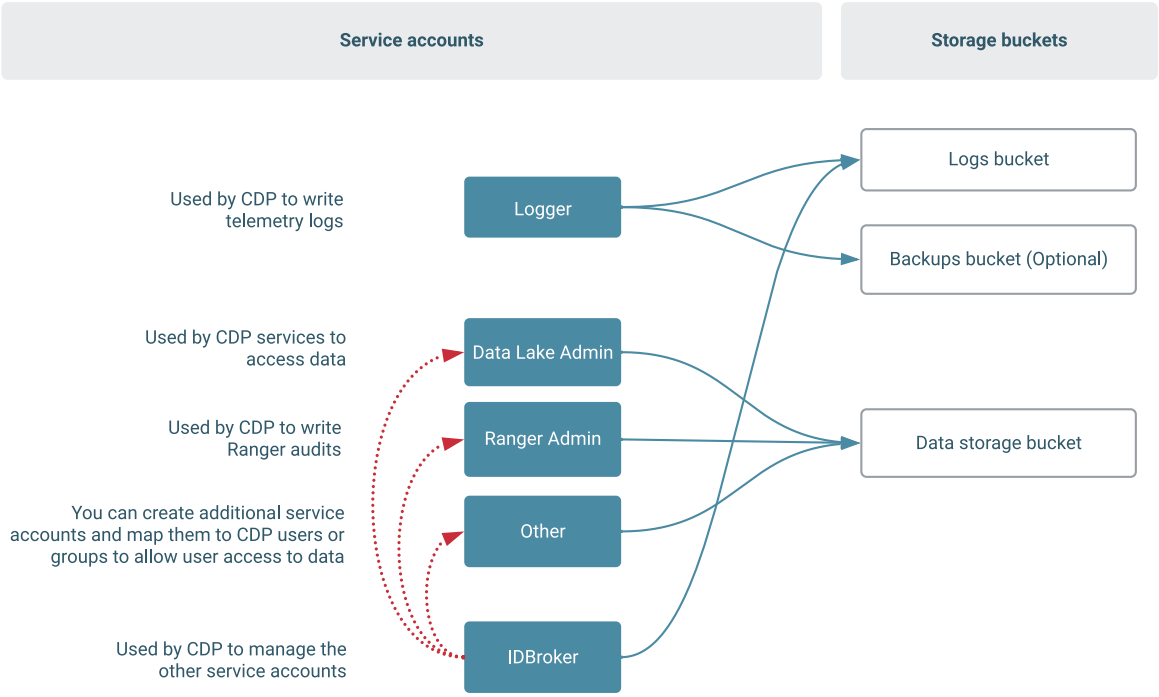

In addition, Cloudera provisioning credential's service account (that you create as part of Create provisioning service account and generate access key) needs to have the Service Account User role to access to the Logger and IDBroker service accounts:

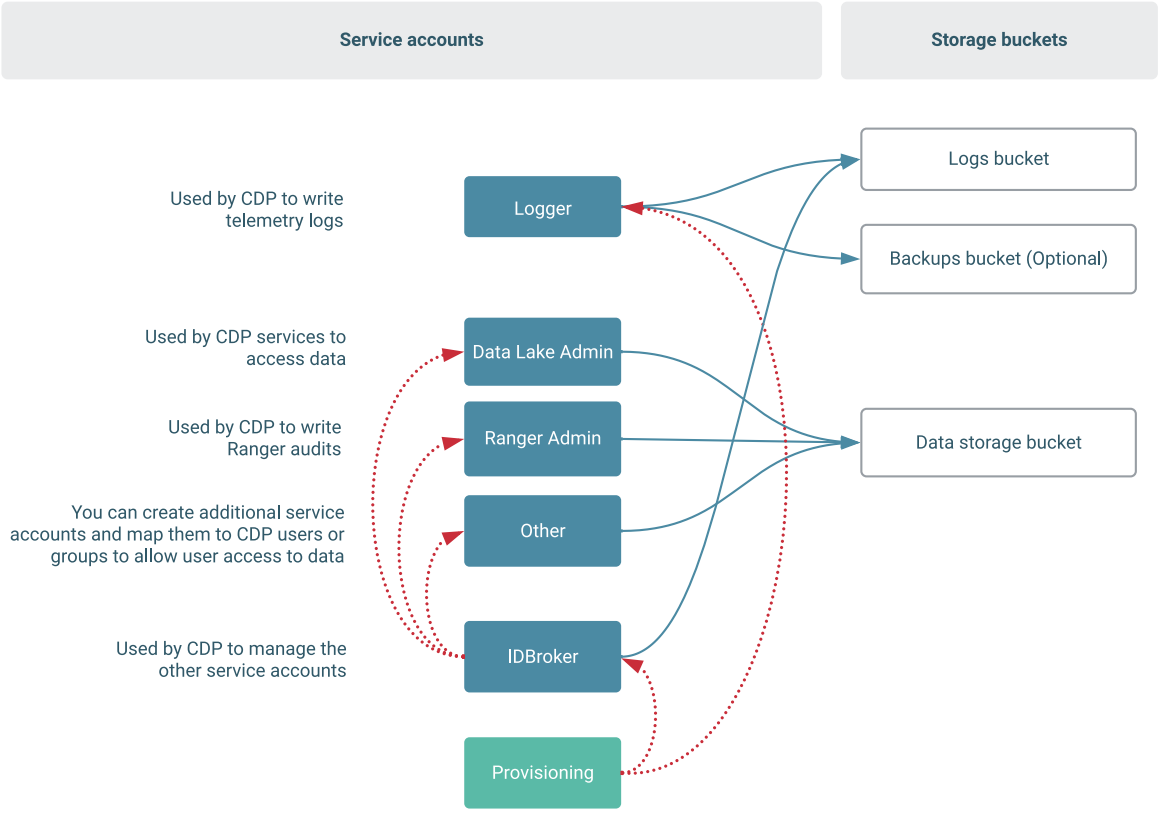

You need to perform the following high-level steps in order to create the required resources:

1. You should create the required Logs and Data storage GCS buckets. You can also create a separate Backups bucket.
2. Create the required service accounts.
3. Create the required custom roles.
4. Add service accounts as members to the Logs and Data Storage buckets.
5. Add the IDBroker service account as a member to other service accounts.
6. Add the provisioning service account as a service account user to the Logger and IDBroker service accounts.
7. Once you have met all of the GCS prerequisites, you can register a GCP environment in Cloudera.

The instructions for performing these steps are mentioned below.

Create the GCS buckets

Use these steps to create the two required GCS buckets.

Steps

1. In Google Cloud console, navigate to Cloud Storage > Browser.
2. Click on +Create bucket.
3. Name your bucket.
4. Click Create.

Repeat these steps for both buckets. Note the bucket names. You will need to provide them to Cloudera later.

For more information, see GCP docs linked below.

Related Information

Creating storage buckets

Create the service accounts

Use these steps to create the required service account.

Steps

1. In Google Cloud console, navigate to the project used for Cloudera.
2. Navigate to IAM > Service accounts.
3. Click on +Create service account.
4. Provide a name.
5. Click Create.

Repeat these steps for all service accounts. Copy and save the email addresses identifying the created service accounts. You will need to provide them to Cloudera. Service account naming convention is <service-account-name>@<project-id>.iam.gserviceaccount.com.

For more information, see GCP docs linked below.

Related Information

Creating a service account

Create the custom role for the Logger

Use these steps to create the custom role for the Logger service account.

Steps

1. In Google Cloud console, in the same project, navigate to IAM > Roles
2. Click on +Create Role
3. Enter a name
4. Click +Add permissions and add all the permissions mentioned in the respective entry in the above table.
5. When done adding permissions, click Create.

For more information, see GCP docs linked below.

Related Information

Creating a custom role

(Optional) Create other custom roles

If you would like to create custom roles for other service accounts, follow the same instructions as above. You only need to do this if you don't want to use the predefined roles listed in the table.

Add service accounts as members to buckets

Use the following steps to a add service account as a member to a bucket.

Steps

1. In Google Cloud console, in the same project, navigate to Cloud Storage > Browser.

2. Perform membership and role assignment for the Logs bucket:
   a. Find your Logs bucket.
   b. Click on ⋮ > Edit bucket permissions or double-click on the bucket and then click on the Permissions tab.
   c. Next to Permissions, click +Add:

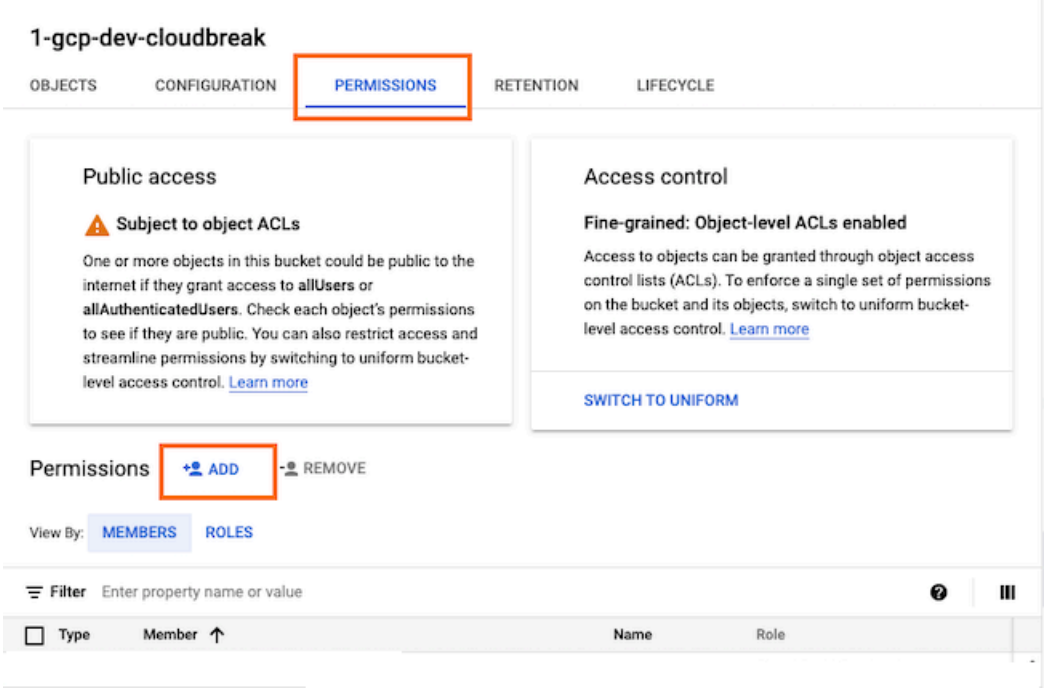

   d. Under New members, select the Logger service account and the IDBroker service account. Under Role, select the custom role created earlier (in the screenshot the role is called Logger):

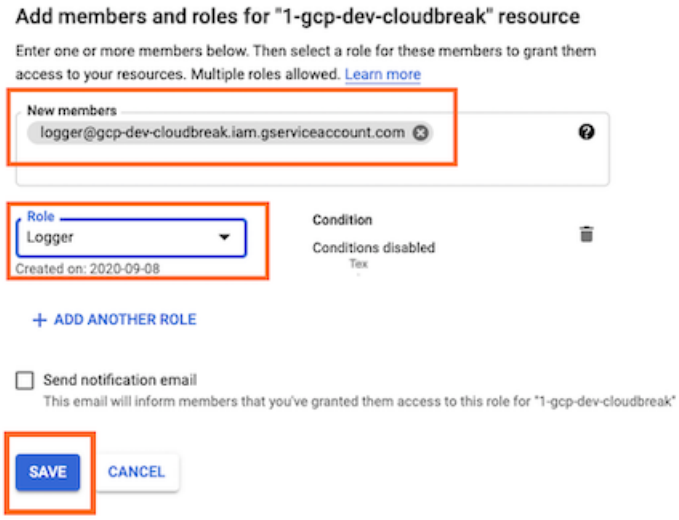

   e. Click Save.
3. If you created the separate Backups bucket for FreeIPA backups, repeat the above steps for the Backups bucket. Use the same Logger service account, IDBroker service account and the custom Logger role created earlier.
4. Repeat the above steps multiple times for the Data storage bucket. You need to add the Data Lake Admin, Ranger Audit, and other service accounts (if created) as members of the Data storage bucket and assign the respective roles mentioned in the above table. Each of these role assignments requires a separate set of steps, so you need to repeat the steps as many times as you have service accounts.

For more information, see GCP docs linked below.

Related Information

Adding a member to a bucket-level policy

Add IDBroker as a member to other service accounts

Similarly as with the buckets, you need to navigate to service account permissions and add the IDBroker service account as a member with the specified role.

Steps

You need to do this for all the service accounts except Logger (and except IDBroker itself):

1. In Google Cloud console, in the same project, navigate to IAM > Service accounts.
2. Double-click on the service account entry.
3. Navigate to the Permissions tab.
4. Click Grant Access and then add the IDBroker service account as a member with the role specified in the above table:

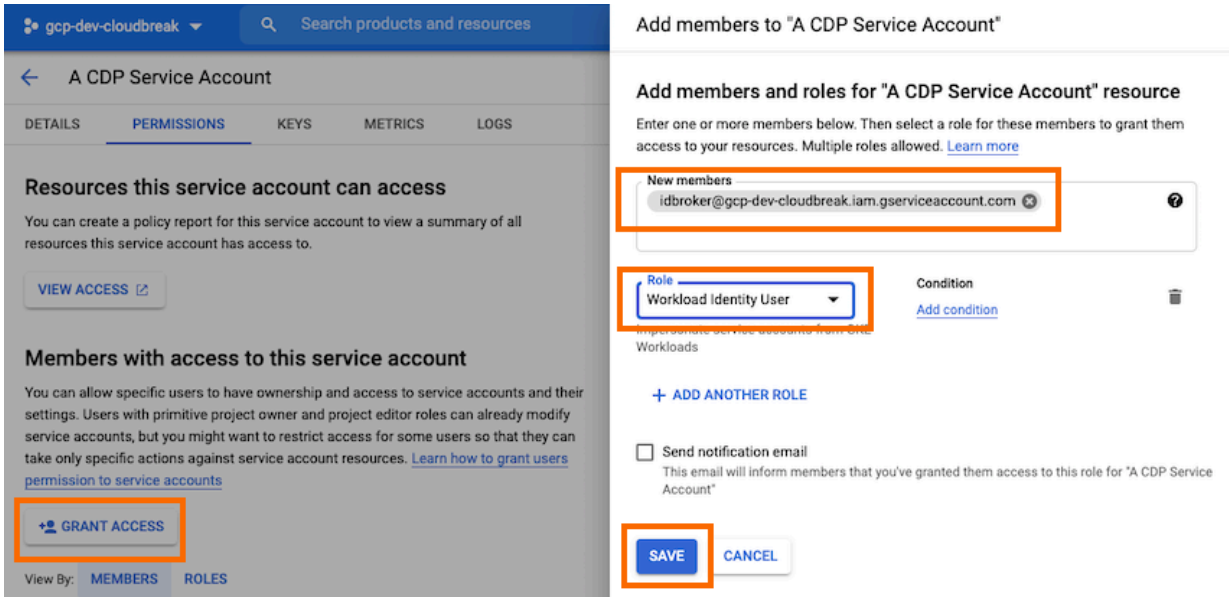

5. Click Save.

Repeat these steps for all service accounts except Logger.

For more information, see GCP docs linked below.

Related Information

Allowing a member to impersonate a single service account

Add provisioning service account as a service account user

To complete the setup, you need to update the permissions of the Logger and IDBroker service accounts, granting the provisioning service account the Service Account User role.

Steps

1. In GCP IAM console, navigate to Service Accounts.
2. Find your Logger service account.
3. Click Manage Permissions to access the Permissions tab.

4. Click Grant Access and then add the provisioning service account as a member with role Service Account User:

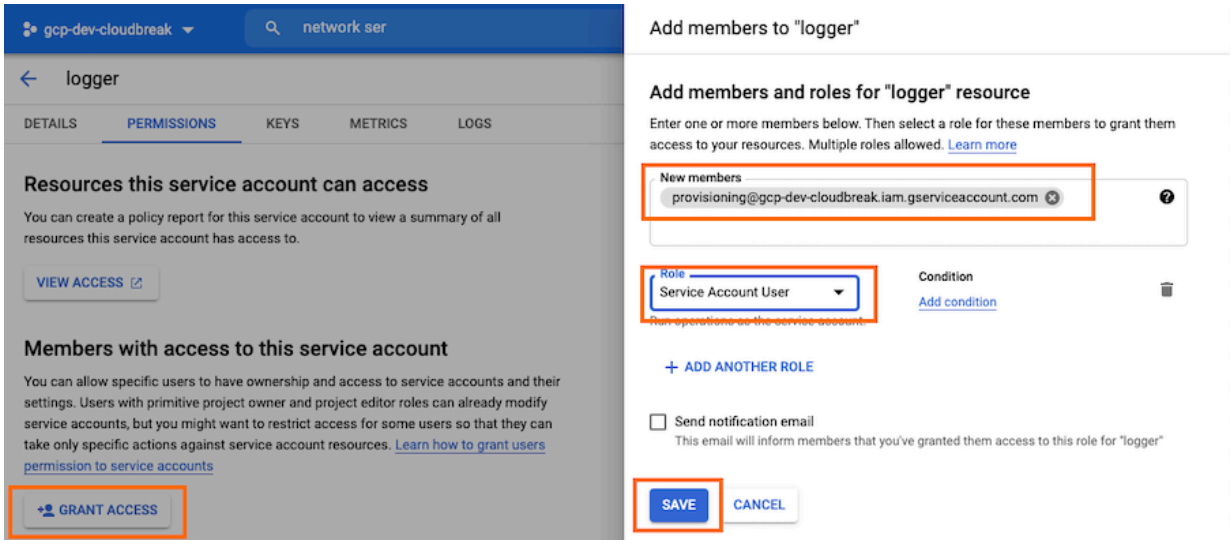

5. Click Save.
6. Repeat the steps for the IDBroker service account.

Providing the parameters in Cloudera

Once you've created the bucket and instance profiles, provide the information related to these resources in the Register Environment wizard as follows:

Data Access and Data Lake Scaling > Data Access:

UI parameter	What to provide
Assumer Service Account	Select the IDBroker service account created earlier.
Storage Location Base	Enter the name of your Data storage bucket created earlier for the storage location base.
Data Access Service Account	Select the Data Lake Admin service account created earlier.
Ranger Audit Service Account	Enter the email address for the Ranger Audit service account created earlier. The service account email address uses the following format: <service-account-name>@<project-id>.iam.gserviceaccount.com.
Backup Location Base (Optional)	If you created it, enter the name of your Backups bucket for storing FreeIPA and Data Lake backups. This is optional. If you don't provide this, FreeIPA and Data Lake backups will be stored in the Logger bucket.

Storage and Audit page > Logs

UI parameter	What to provide
Logger Service Account - Logger	Select the Logger service account created earlier.
Logs Location Base	Enter the name of your Logger bucket created earlier for logs location base.

Data Access and Data Lake Scaling > Mappings

You can use this section to set service account to Cloudera user/group mappings for the additional service accounts created for user access to data. Or you can do this once your environment is running, as part of Onboarding Cloudera users and groups for cloud storage.

Onboarding Cloudera users and groups for GCP cloud storage

The minimal setup defined earlier spins up a Cloudera environment and Data Lake with no end user access to cloud storage. Adding users and groups to a Cloudera environment involves ensuring they are properly mapped to service accounts to access cloud storage.

In general, to have new users or groups onboarded, you need to do the following:

1. Create a new service account and assign appropriate IAMroles or granular permissions on the scope of the Storage Location Base or its specific sub-directory. You might have already performed this step earlier during setting up the Minimum setup for cloud storage.
2. In order to use these storage accounts in Cloudera, create a user/group to service account mapping in Cloudera.

This needs to be done for each user type. For example, you can create two service accounts in GCP, one for Data Scientists and another for Data Engineers, and then you map each of them to a group of users in Cloudera.

The onboarding of users can either happen as part of environment registration, or you can do it once an environment is running. The steps below show you how to onboard users once an environment is running.

Creating Cloudera user/group to service account mappings

After creating the additional service accounts, map each of them to a specific user or group.

Before you begin

The steps below show how to add the mappings to an existing environment. Alternatively, you can add them during environment registration, as mentioned in the Minimum setup for cloud storage.

Note: If a user is mapped to multiple roles via group membership, the specific role to be used needs to be provided at runtime. If the user is mapped directly to a role, the direct mapping takes precedence over mapping via group membership. For information on how to specify the role, refer to Specifying a group when user belongs to multiple groups.

Required role: DataSteward, EnvironmentAdmin, or Owner

Steps

For CDP UI

1. The option to add/modify the service account to user/group mappings is available from the Cloudera Management Console under Environments > click on an environment > Actions > Manage Access > IDBroker Mappings.
2. Under Current Mappings, click Edit.
3. Click + to display a new field for adding a mapping.
4. Provide the following:
   a. The User or Group dropdown is pre-populated with Cloudera users and groups. Select the user or group that you would like to map.
   b. Under Role, specify the resource ID of a service account (copied from Google Cloud IAM). For example “datascientists@gcp-cdpdev.iam.gserviceaccount.com”.
5. Repeat the previous two steps if you would like to add additional mappings.
6. Click Save and Sync.

For example, in the example setup we created the following roles:

- DATAENG_ROLE - We created this role while onboarding users and we assume that there is a DataEngineers group that was created in Cloudera.
- DATASCI_ROLE - We created this role while onboarding users and we assume that there is a DataScientists group that was created in Cloudera.

For CDP CLI

If you would like to create the mappings via CDP CLI, you can:

1. Use the cdp environments get-id-broker-mappings command to obtain your current mappings.

2. Use the cdp environments set-id-broker-mappings command to set additional mappings. The only way to use this command is to:
   a. Pass all the current mappings
   b. Add the new mappings.
3. Next, sync IDBroker mappings to the environment:

```
cdp environments sync-id-broker-mappings --environment-name demo3
```

4. Finally, check the sync status:

```
cdp environments get-id-broker-mappings-sync-status --environment-name d
emo3
```

Storage bucket for OS images

By default, Cloudera creates a GCS bucket for storing OS images used for Data Lake and Cloudera Data Hub VMs, but you can optionally pre-create it if your organization requires it.

- It must have a name using the following convention: <ClouderatenantID>-<ProjectID>

You can your Cloudera tenant ID as described in Obtain Cloudera tenant ID.

Supported GCP block storage

Review the list of supported GCP disk types for Cloudera.

Cloudera supports the following GCP disk types:

Cloudera name	GCP name
pd-ssd	SSD Persistent Disk
pd-balanced	Balanced Persistent Disk
pd-extreme	Extreme Persistent Disk
pd-standard	Standard Persistent Disk
local-ssd	Local SSDs

Related Information

Google Cloud storage options

SSH key pair

When registering an environment, you will be asked to provide an SSH key pair for admin access to Cloudera. The minimum SSH key size is 2048 bits.

You will need to paste the public SSH key in Cloudera during environment registration. If you need help generating an SSH key, refer to https://www.ssh.com/ssh/keygen/.

Customer managed encryption keys

By default, a Google-managed encryption key is used to encrypt disks and Cloud SQL instances in Data Lake, FreeIPA, and Cloudera Data Hub clusters, but you can optionally configure Cloudera to use a customer-managed encryption key (CMEK) instead.

When a CMEK is provided during environment registration, all the Data Lake, FreeIPA, and Cloudera Data Hub disks and the Cloud SQL instances are encrypted using that key.

To set up a CMEK, perform the following tasks:

1. Review the CMEK requirements.
2. Create a key ring and an encryption key.
3. Assign the required permissions to the encryption key.

This document guides you through all the required steps performed using the GCP console and Google Cloud Shell. Once you've met the prerequisites, pass the encryption key when creating a Cloudera environment via Cloudera web interface or Cloudera CLI.

For general information about customer managed encryption keys, see Customer-managed encryption keys (CMEK).

Cloud HSM and Hosted Private HSM encryption

Cloudera supports encryption using Cloud HSM encryption keys. The overall requirements and steps are the same as usual, just when you create the encryption key, you need to select the protection level to be "HSM". The instructions provided here consider the scenario of a Cloud HSM encryption key.

Cloudera also supports encryption using encryption keys from Hosted Private HSM. If you wish to use Hosted Private HSM, the GCP support helps you configure the setup and then provides you with the encryption key ARN. Cloudera can then use the key ARN.

For more information, see Cloud HSM and Hosted Private HSM.

CMEK requirements

Ensure that the CMEK that you are planning to use for encrypting your Cloudera environment meets the requirements.

The CMEK must meet the following requirements:

- CMEK needs to be in the same region as the environment.
- The key should have the following permissions for the compute and cloud sql service agents:

  Cloud KMS CryptoKey Encrypter/Decrypter

The instructions below show you how to create a CMEK that meets these requirements.

Create key ring and encryption key

Use the following instructions to create a key ring and an encryption key in Google Cloud.

Note:

Key rotation and storage are Google-managed.

Steps

1. In the GCP console, navigate to Security > Key Management.

2. Create a key ring or use any existing one. Ensure that the key ring location and the location of the resources you create for the Cloudera environment are the same.

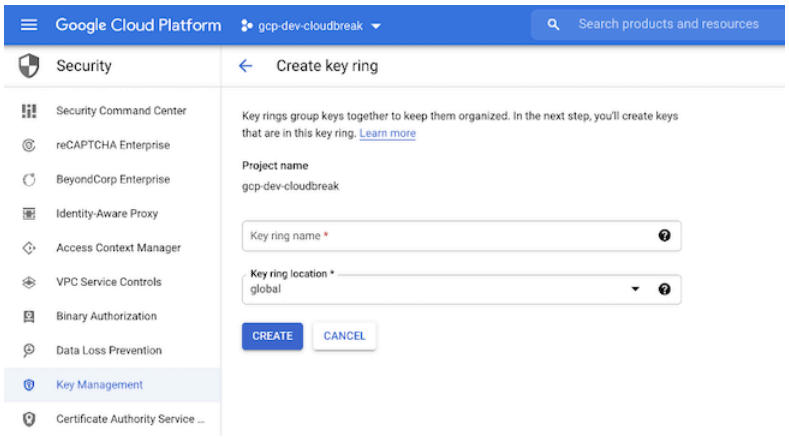

3. Navigate to the key ring that you have previously created.
4. Click on +Create Key to create a new key inside the key ring.

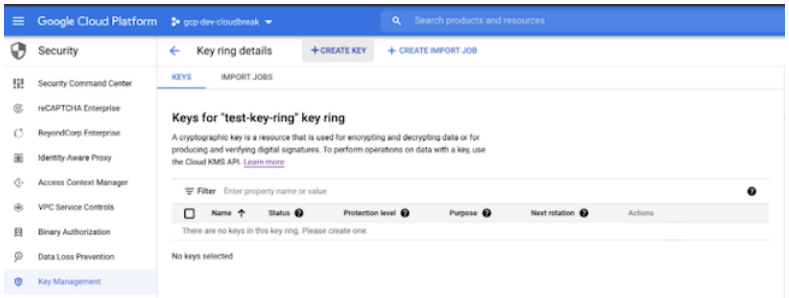

5. Under What type of key do you want to create?, select Generated key.
6. Under Key name, enter the name for your key.
7. From the Protection level dropdown:
   a. If you are using a standard CMEK, select Software.
   b. If you are using a Cloud HSM key, select HSM.
8. From the Purpose dropdown, select Symmetric encrypt/decrypt.
9. Use the default values for Rotation period and Starting on.
10. Click Create.

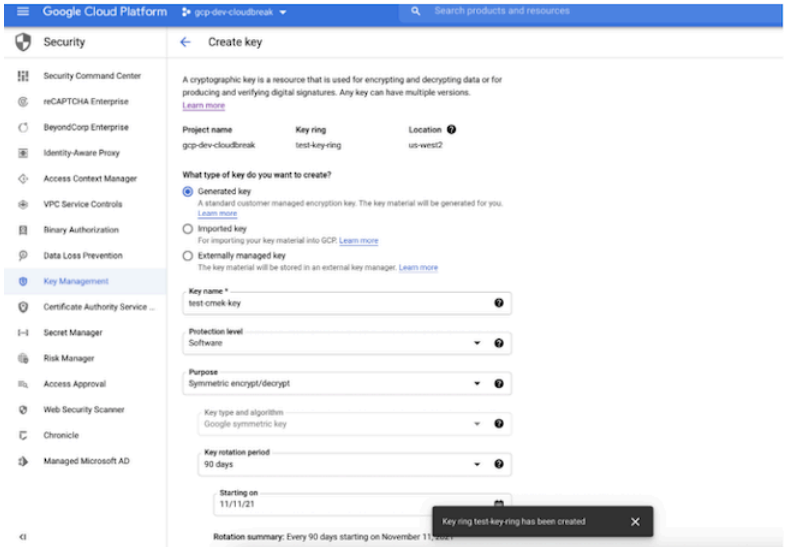

Assign the required permissions to the encryption key

Once the key has been created, you need to assign the required permissions to it. The following commands can be used to set it up using Google Cloud Shell.

Prerequisites

Make sure that you have the following available:

- The project number of the Google project number where the compute and SQL resources are created (PROJECT_NUMBER).
- Copy the key ring resource name (KEYRING_RESOURCE_NAME) and the key resource name (KEY_RESOURCE_NAME) from the Google cloud console. You can copy the key ring resource name from the dropdown after clicking three vertical dots next to the key ring. You can copy the key resource name in a similar manner.

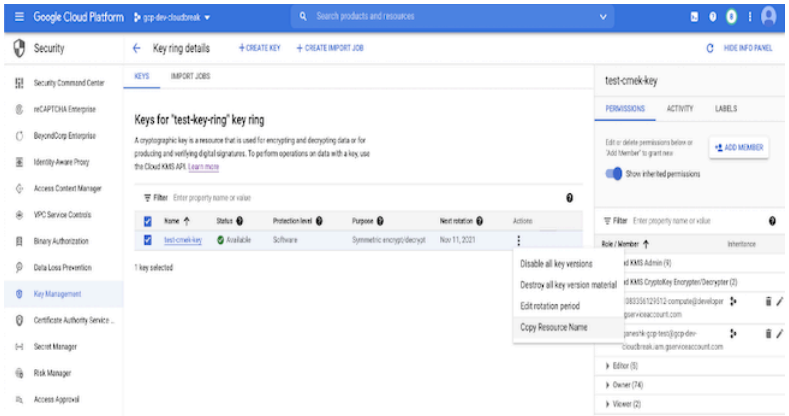

Steps

1. If the cloud sql service agent does not exist in the project, create it using:

```
gcloud beta services identity create --service=sqladmin.googleapis.com --
project=<project_name>
```

   This command creates a service identity in the following format:

   service-108335612.5..@gcp-sa-cloud-sql.iam.gserviceaccount.com

   where "108335612.5.." is the PROJECT_NUMBER to be used in step 2 and 3.

2. Assign the IAM policy to encrypt and decrypt KMS keys. Replace the variables in caps with the values obtained earlier:

```
gcloud kms keys add-iam-policy-binding KEY_RESOURCE_NAME \
 --location=GCP_REGION \
 --keyring=KEYRING_RESOURCE_NAME \
 --member=serviceAccount:service-PROJECT_NUMBER@gcp-sa-cloud-sql.iam.gser
viceaccount.com \
 --role=roles/cloudkms.cryptoKeyEncrypterDecrypter
```

3. Assign the IAM policy to the compute service agent. Replace the variables in caps with the values obtained earlier:

```
gcloud kms keys add-iam-policy-binding KEY_RESOURCE_NAME \
 --location=GCP_REGION \
 --keyring=KEYRING_RESOURCE_NAME \
 --member=serviceAccount:service-PROJECT_NUMBER@compute-system.iam.gservi
ceaccount.com \
 --role=roles/cloudkms.cryptoKeyEncrypterDecrypter
```

GCP limits

When you create your Google Cloud project, Google sets limits to the resources available to you. In some cases, the limits are insufficient for Cloudera and you need to request to have them increased.

For example, in a newly created project, you may need to increase the limits for:

- Public IPs
- Virtual CPU (vCPU)
- Disk storage

For a full list of resources that Cloudera provisions, refer to GCP resources used by Cloudera.

For information on checking and increasing your resources quotas, refer to Resource quotas.

Overview of GCP resources used by Cloudera

The following Google Cloud resources are used by Cloudera and Cloudera services.

GCP resources created for a Cloudera environment

When a Cloudera environment is created, a FreeIPA cluster and a Data Lake cluster are created.

The following Google Cloud resources are created for FreeIPA (one per environment):

Resource	Description
Service account for credential	To allow Cloudera to access and provision resources in your Google Cloud project, you must create a service account in your Google Cloud project, assign required roles, and generate a JSON access key that can later be provided to Cloudera.
VPC network and subnets	During environment creation you provide your own existing VPC network and subnets. All compute resources that Cloudera provisions for the environment and Cloudera services are provisioned into the VPC network specified during environment creation.
Firewall rules	Firewall rules define inbound and outbound access to the instances. If during environment creation you choose to have new firewall rules created, then they are created on your GCP account. Alternatively, you can provide your own existing firewall rules.
VM instances	During environment creation, two or three e2-standard-2 VM instances are provisioned for the FreeIPA HA server. The number of VMs depends on the selected Data Lake type.
OS disk	An OS disk is provisioned for the FreeIPA VM.
Attached disk	An attached disk (pd-standard) is provisioned for each VM.
Public IP address (if required)	If you choose to use public IPs, your VM is assigned a public IP address.
GCS bucket for storing operating system images	By default, Cloudera creates a storage bucket that is used solely for storing operating system images. If required, you can optionally pre-create this account and copy the required images.

In addition, the following resources are created for each Data Lake (one per environment):

Resource	Description
VM instances	VM instances are provisioned for the Data Lake nodes. • Light duty: Two VM instances are provisioned: One e2-standard-2 VM instance (for IDBroker) and one e2-standard-8 VM instance (for master) are created. • Medium duty: Ten VM instances are provisioned: Two e2-standard-2 (IDBroker), three e2-standard-4 (two Data Lake Master nodes and one Auxiliary node), and five e2-standard-8 (three DataLake Core nodes and two Gateway nodes).
Attached disk	An attached disk (pd-standard) is provisioned for each VM.
OS disk	An OS disk is provisioned for each VM.
PostgreSQL database	A custom PostgreSQL database instance (100GB SSD, 2vCPU, 13 GB RAM) is provisioned for the Data Lake. This databse instance is used for Cloudera Manager, Ranger, and Hive MetaStore.
Firewall rules	Firewall rules define inbound and outbound access to VM instances. If during environment creation you choose to have new firewall rules created, then they are created on your GCP project.
Google storage buckets	The existing Google Storage bucket that you provide during environment creation for the Data Lake is used for Data Lake log storage and workload data storage.
Service accounts	Prior to registering your environment in Cloudera, during Google storage setup, you should create service accounts and assign roles to them as instructed in Cloudera documentation.
Public IP address (if required)	If you choose to use public IPs, your VM is assigned a public IP address.

GCP resources created for Cloudera Data Hub

The following Google Cloud resources are created for each Cloudera Data Hub cluster:

Resource	Description
VM instances and attached storage	A VM is created for each cluster node. The VM type varies depending on what you selected during Cloudera Data Hub cluster creation. For a list of supported VM types, refer to Cloudera Public Cloud service rates.
Firewall rules	Firewall rules define inbound and outbound access to VM instances. If during environment creation you choose to have new firewall rules created, then they are created on your GCP project.
OS disk	An OS disk is provisioned for each VM.
Attached disk	An attached disk (pd-standard) is provisioned for each VM, as specified during Cloudera Data Hub cluster creation. The disk size is selected during cluster creation.
Public IP address (if required)	If you choose to use public IPs, each of the VMs is assigned a public IP address.

GCP outbound network access destinations

If you have limited outbound internet access (for example due to using a firewall or proxy), review this content to learn which specific outbound destinations must be available in order to register a Cloudera environment.

We recommend hostname-based policies, as some of the destination services do not have static IP addresses. IP address details in CIDR notation have been provided where static IPs are in-use.

Note:

If the cloud provider network that you would like to use for registering a Cloudera environment uses a custom DNS server that does not allow name resolution for public domain, you should add all the domains listed in the below tables to the DNS forwarder for name resolution.

The following list includes general destinations as well as GCP-specific destinations.

General endpoints

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
Control Plane API	All services	US-based Control Plane: api.us-west-1.cdp.cloudera.com EU-based Control Plane: api.eu-1.cdp.cloudera.com AP-based Control Plane: api.ap-1.cdp.cloudera.com	HTTPS with Cloudera-generated access key	TCP/443	Cloudera Control Plane REST API.
Cloudera CCMv1 Persistent Control Plane connection	All services	*.ccm.cdp.cloudera.com 44.234.52.96/27	SSH public/ private key authentication	TCP/6000-6049	One connection per cluster configured; persistent
Cloudera CCMv2 Persistent Control Plane connection	All services	US-based Control Plane: *.v2.us-west-1.ccm.cdp.cloudera.com 35.80.24.128/27 EU-based Control Plane: *.v2.ccm.eu-1.cdp.cloudera.com 3.65.246.128/27 AP-based Control Plane: *.v2.ccm.ap-1.cdp.cloudera.com 3.26.127.64/27	HTTPS with mutual authentication	TCP/443	Multiple long-lived/persistent connections
Cloudera Databus Telemetry, billing and metering data	All services	US-based Control Plane: dbusapi.us-west-1.sigma.altus.cloudera.com api.us-west-1.cdp.cloudera.com https://cloudera-dbus-prod.s3.amazonaws.com EU-based Control Plane: api.eu-1.cdp.cloudera.com https://mow-prod-eu-central-1-sigmadbus-dbus.s3.eu-central-1.amazonaws.com https://mow-prod-eu-central-1-sigmadbus-dbus.s3.amazonaws.com AP-based Control Plane: api.ap-1.cdp.cloudera.com https://mow-prod-ap-southeast-2-sigmadbus-dbus.s3.ap-southeast-2.amazonaws.com https://mow-prod-ap-southeast-2-sigmadbus-dbus.s3.amazonaws.com	HTTPS with Cloudera-generated access key for dbus HTTPS for S3	TCP/443	Regular interval for telemetry, billing, metering services, and used for Cloudera Observability if enabled. Larger payloads are sent to a Cloudera managed S3 bucket.

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
Cloudera Observability Metrics System metrics collection	All services	US-based Control Plane: *.api.monitoring.us-west-1.cdp.cloudera.com EU-based Control Plane: *.api.monitoring.eu-1.cdp.cloudera.com AP-based Control Plane: *.api.monitoring.ap-1.cdp.cloudera.com	HTTPS	TCP/443	New as of March 2024
Cloudera Manager parcels Software distribution	All services	archive.cloudera.com	HTTPS	TCP/443	Cloudera's public software repository. CDN backed service; IP range not predictable.
RPMs Cloudera RPMs for workload agents	All services	cloudera-service-delivery-cache.s3.amazonaws.com	HTTPS	TPC/443	RPM packages for some workload components

GCP-specific endpoints

Description/ Usage	Cloudera service	Destination	Protocol and Authentication	IP Protocol/Port	Comments
APIs	All services	storage.googleapis.com iamcredentials.googleapis.com	HTTPS	TCP/443	In addition to adding the listed destinations, you need to configure Private Service Connect. Private Service Connect lets you send traffic to Google APIs using a Private Service Connect endpoint that is private to your VPC network. To configure Private Service Connect, refer to Configuring Private Service Connect. **Note:** This is not optional. If you don't configure this, environment registration will fail.

Access to workload UIs

If you have restricted DNS or networking setup, make sure that *.cloudera.site is resolvable from your network so that members of your organization can access workload UIs.

Cloudera workloads (including Data Lake) use subdomains under cloudera.site to host various UI endpoints (Cloudera Manager, Ranger, Knox, Hue and so on). Cloudera automatically provisions these endpoints whenever a Data Lake, Cloudera Data Hub or another type of workload (for example, Virtual Warehouse in Cloudera Data Warehouse) is created, and routing is set up so that you can access these endpoints from your network.

The subdomains are assigned under cloudera.site using the following convention:

```
<endpoint-name>.<env-truncated-name>.<customer-workload-subdomain>.<regional
-subdomain>.cloudera.site
```

Supported browsers

Cloudera validates and tests against the latest version and supports recent versions of the following browsers:

- Google Chrome
- Mozilla Firefox

  Note:

  - Mozilla Firefox is not supported by Cloudera Data Engineering.
  - Certain accessibility features in Cloudera DataFlow do not work in Mozilla Firefox.
- Safari
- Microsoft Edge

Cloudera CIDR

List of Cloudera Control Plane source IP addresses

In cases where the Cloudera Control Plane is communicating directly with resources in the Cloudera environment, the Cloudera Control Plane source IP address will be from the ranges below.

Security groups deployed/managed by Cloudera Control Plane will automatically include the required rules. These source IP ranges will be included if the Cloudera Control Plane requires direct inbound access to the resource.

Cloudera Control Plane Region	IP Ranges
us-west-1	35.80.24.128/27, 35.166.86.177/32, 52.36.110.208/32, 52.40.165.49/32
eu-1	3.65.246.128/27
ap-1	3.26.127.64/27

If the security groups (or other network access policies) in the network path are not managed by the Cloudera Control Plane, ensure that inbound access from the appropriate range(s) are permitted.

Cloudera Public Cloud

Cloudera SDK

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

CDP SDK overview **4**
CDP SDK prerequisites 4
Using the Cloudera SDK for Java 4

Overview of the Cloudera SDK for Java

You can use the Cloudera SDK for Java to integrate Cloudera services with your applications. Use the Cloudera SDK to connect to Cloudera services and create and manage clusters and run jobs from your Java application or other data integration tools that you use in your organization.

Comprehensive Java documentation is available on Cloudera's Github. The Cloudera SDK repository is also available.

To use the Cloudera SDK for Java, you must have a Cloudera account and a user account or machine user account with the appropriate role and environment to enable you to perform your required tasks in Cloudera.

Prerequisites for using the Cloudera SDK for Java

Before you use the Cloudera SDK to perform the tasks from an application, we recommend that you first verify that you can perform the tasks on the Cloudera console or through the CLI.

When you are able to successfully create clusters and run jobs on the Cloudera Management Console or by using the CLI, use the Cloudera SDK to create clusters and run jobs from your application.

Using the Cloudera SDK for Java

You must download the SDK Jar file, set up your access key, create your application, and run the Cloudera SDK sample applications.

About this task

To use the Cloudera SDK for Java, complete the following steps:

Procedure

1. Download the latest version of the Cloudera SDK jar files from the Cloudera SDK repository.
2. Set up your API access credentials. Cloudera uses an access key to manage access to Cloudera services through the Cloudera API. You must set up the API credentials for your application.
   - Use the Cloudera Management Console to generate an access key. You must be a Cloudera administrator or have the IAM User role to generate an API access key.
     - If you are a Cloudera administrator, you can generate an access key for a user or machine user account. For more information, see *Generating an API access key* (for a long-lived API access key) or *Logging into the CDP CLI/SDK* (for a shorter-lived access key.
     - If you are a user with the IAM User role, you can generate an access key for your own user account.
   - You can set up the API access key for your application in one of the following ways:
     - Environment variable. For an example of how to set up the API access key in an environment variable, see the source code in the CDPEnvironmentVariableCredentialsProvider.java file.
     - Java system variable. For an example of how to set up the API access key in a Java system variable, see the source code in the CDPSystemPropertiesCredentialsProvider.java file.
     - Cloudera credentials file. For an example of how to set up the API access key in a Cloudera profile, see the source code in the CDPProfileCredentialsProvider.java file.
3. Create your application.
4. View or run the Cloudera SDK sample applications to understand how to use the Cloudera API to access Cloudera services.

Cloudera Public Cloud

Cloudera Security Overview

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera security FAQs....4

Cloudera identity management....5

Cloudera user management system....5
FreeIPA identity management....6
Cloud identity federation....7
Authentication with Apache Knox....9

Access to customer resources....10

Handling of sensitive data in Cloudera....11

User access to clusters....11

Secure inbound communication....11

Cloudera Private Links Network Overview....15

Cloudera Runtime Security and Governance....16

Cloudera security FAQs

This topic covers frequently asked questions related to the communication between the Cloudera Control Plane and workload subnets.

Data security within workload subnets and virtual networks

Cloudera provides comprehensive security features to ensure that minimal network configuration is needed on workload clusters and that only metrics, logs, and control signals go in and out of the customer's network. No data from the workload clusters is ever accessed by the Cloudera Control Plane.

To operate at the highest level of security, Cloudera recommends running Cloudera workload clusters in private subnets. This means that nodes in the workload clusters do not have public IP addresses and all outbound traffic to the internet goes through a gateway or a firewall. This allows your security operations team to ensure that hosts that are allowed to communicate with the clusters are legitimate and pose no security threat to the assets. A common best practice is to avoid inbound connections of any sort directly into the workload subnet.

Cloudera recommends that no inbound connections be allowed into the private network and that you use your sec-ops approved method(s) to provide outbound access to the list of hosts specified in the AWS Azure GCP outbound network access destinations.

Communication between Cloudera and workload subnets

Customers use Cloudera Management Console to operate and manage workload clusters running in their own VPCs. For every operation performed through the Cloudera Management Console on the workload clusters, a control signal is sent to the hosts in the private network. This is achieved through a feature called Cluster Connectivity Manager.

Cluster Connectivity Manager eliminates the need to configure inbound connections into your secure private network. Cluster Connectivity Manager, which is set up during cluster creation, configures a reverse tunnel from the customer’s private network to the Cloudera Control Plane. All control signals to create/delete clusters, stop/start environments and other management actions related to workload clusters go through the Cluster Connectivity Manager tunnel. Cluster Connectivity Manager allows customers to avoid configuring any inbound connections/routes to their private workload subnets, thus providing a better security posture for their public cloud assets.

More information on Cluster Connectivity Manager is available in the Cluster Connectivity Manager documentation.

Cloudera Private Links Network for additional privacy and security

Users who are concerned with privacy can utilize the Cloudera Private Links Network to establish private and secure connections from their workloads to the Cloudera Control Plane without using the public internet.

Cloudera Private Links Network is designed to provide seamless, private connectivity between your cloud workloads and the Cloudera Control Plane. For further information, see Cloudera Private Links Network Overview.

Egress network setup for better control and visibility on outbound traffic

Large security-conscious enterprises typically inspect all traffic going in and out of their private networks. This is generally achieved in the following way:

1. Identify a single egress virtual network that is used to provide internet access to all other subnets and virtual networks. Route outbound (Internet) traffic from all other subnets to this egress network.
2. Purpose-built technologies, such as web proxies, next-generation firewalls and cloud access security brokers are deployed in the egress VPC to monitor for anomalous outbound behavior. Network analyzers and forensic recorders can also be used.

Cloudera recommends a similar topology to configure outbound traffic rules that are needed for normal operation of the clusters in a private network. The alternative is to set up egress rules within the same VPC that hosts the private subnet.

Information exchanged between Cloudera and workload subnets

Irrespective of the security posture used by the customer to secure their assets in the workload subnet, a minimum set of communications is needed for normal operations of a Cloudera environment. The following set of points summarize the data exchanged between Cloudera Control Plane and workload subnets:

1. All user actions on the Cloudera Control Plane that are related to interacting with the workload clusters in the workload subnets happen via Cluster Connectivity Manager.
2. Metering and billing information is sent at regular intervals from the customer's workload subnet to the Cloudera Databus as specified in the AWS Azure GCP outbound network access destinations.
3. If used, an optional component called Cloudera Observability also leverages the Cloudera Databus to communicate with Cloudera Control Plane.
4. Diagnostic bundles for troubleshooting are generally sent by customers to engage in support cases. The diagnostic bundles can either be sent on demand or on a scheduled basis to Cloudera Customer Support. This feature also uses the Cloudera Databus channel. Refer to Sending Usage and Diagnostic Data to Cloudera page to read more about diagnostic bundles.
5. Customers generally share logs with Customer Support in a self-service fashion through Cloudera Control Plane. No customer data is ever shared in the diagnostics bundle. To ensure sensitive data does not show up inadvertently in the logs, we recommend that customers follow the directions specified in Redaction of Sensitive Information from Diagnostic Bundles.

Cloudera identity management

Cloudera Identity Management includes the Cloudera user management system and Knox authentication.

Cloudera user management system

Cloudera Management Console includes a user management system that allows you to integrate your identity provider and manage user access to Cloudera resources.

During the initial setup of a Cloudera subscription, Cloudera designates a user account as a Cloudera account administrator. A Cloudera account administrator has all privileges and can perform any task in Cloudera. Administrators can create other Cloudera administrators by assigning the PowerUser role to users. Cloudera administrators can also register environments and create Data Lake clusters.

Cloudera administrators can create users and groups and then assign roles and resource roles to users or groups. The Cloudera Management Console also enables Cloudera administrators to federate access to Cloudera by configuring an external identity provider. Cloudera users can include users corresponding to an actual living person within the organization or machine users.

Note: After setting up the externally managed identity provider integration (Single Sign-On), Cloudera strongly recommends disabling the Cloudera Single Sign-On (CSSO) account credentials using the steps described in Disabling the Cloudera SSO Login.

In addition to the SSO credentials mentioned above, Cloudera uses another set of credentials that must be used for accessing some Cloudera components (for example accessing Cloudera Data Hub clusters via SSH).

To access to the CDP CLI or SDK, each user must have an API access key and private key. Each user must generate this key pair using the Cloudera Management Console, and Cloudera creates a credentials file based on the API access key. When you use the CDP CLI or SDK, Cloudera uses the credentials file to get the cluster connection information and verify your authorization.

Related Information

User Management

FreeIPA identity management

Federating identity management with users/groups maintained in FreeIPA and passwords authenticated via SSO to an SAML-compliant identity provider (IDP) provides the necessary backbone infrastructure needed for Cloudera services, without requiring you to expose your on-prem identity management system over the network.

What is FreeIPA?

FreeIPA is an open-source product that combines four identity management capabilities:

- LDAP directory: a common user directory so that all services in both the SDX and workload clusters can consistently resolve users.
- Kerberos KDC: a single common Kerberos realm so that services can authenticate each other, within and between clusters. Kerberos is also used as a user authentication mechanism by some services.
- DNS server: a relatively simple way to discover and reach shared services in an SDX cluster from various workloads.
- Certificate Authority (CA): some services secure communication channels with TLS, which means they need certificates. A shared CA allows Cloudera to establish a common trusted root for all connected workloads.

Identity management with FreeIPA

IPA is an identity management framework used to assert who a user is. A subset of users and groups are replicated into IPA (and propagated to the nodes via SSSD). Making the users and groups available on the nodes with consistent user names enables security policies to be migrated from on-prem to the cloud. Users and groups are imported from on-prem and principally managed from the Cloudera Control Plane UMS (User Management System,) with IPA providing the backend propagation.

FreeIPA prerequisites

For IPA to work, you must have:

- An AD on-prem or a central LDAP where relationships between users and groups are maintained.
- A SAML identity provider (e.g., Okta or KeyCloak) that can be leveraged to authenticate users and import their groups.

How FreeIPA works

The following diagram illustrates how FreeIPA works:

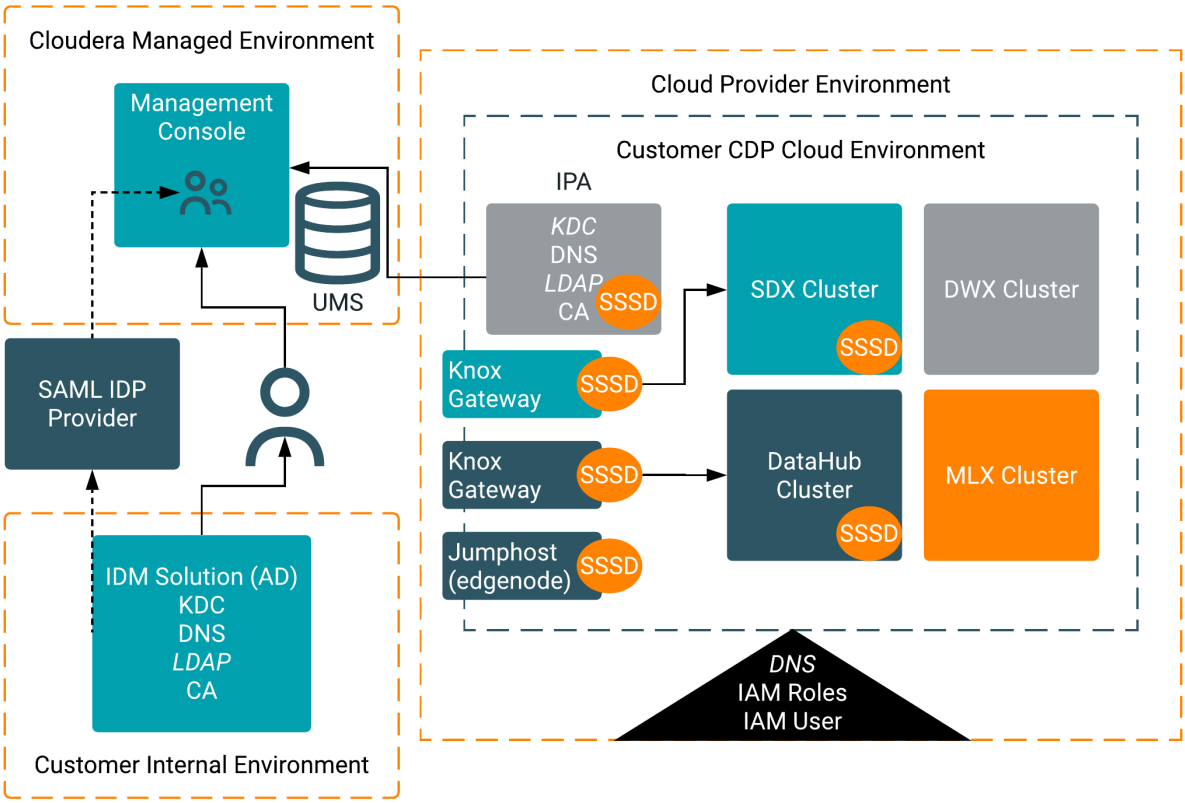

Cloud identity federation

When accessing cloud storage in Cloudera, credentials are provided by Knox IDBroker, an identity federation solution that exchanges cluster authentication for temporary cloud credentials.

What is IDBroker?

IDBroker is a REST API built as part of Apache Knox's authentication services. It allows an authenticated and authorized user to exchange a set of credentials or a token for cloud vendor access tokens.

IDBroker is automatically configured by Cloudera Manager in Cloudera deployments, where Knox is installed. Cloud roles can be mapped to users or groups that exist in UMS to help control authorization.

How IDBroker works

Object store data access permissions are managed within the native cloud provider's authorisation systems, not within Cloudera. So, when Knox IDBroker considers a particular user, it doesn't automatically have credentials for them to access the requested data. IDBroker addresses this problem by giving us a way to map UMS users and groups to cloud provider identities (roles) with associated permissions. Then, when data access operations happen via a Hadoop storage connector (eg: s3a), the connector will connect to IDBroker and ask for short lived access credentials which can be passed to the cloud provider when making data access calls. IDBroker will use the mappings to decide which credentials to obtain and return.

Knox IDBroker can help bridge the identity gap by requesting short term tokens from the cloud provider on behalf of a user that has successfully authenticated with Knox, and who is specified as being a part of a group mapped to a role that can access the bucket.

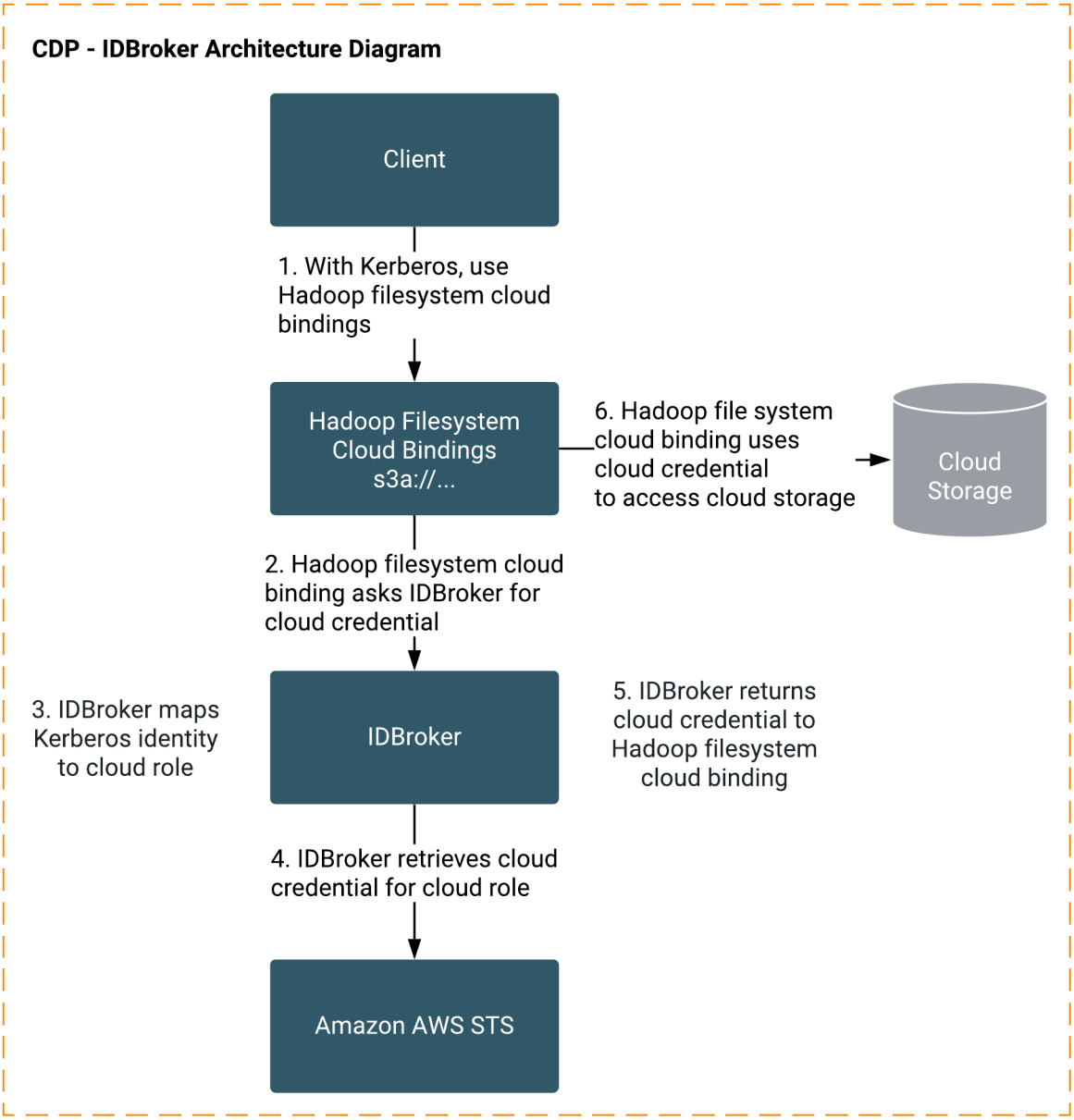

User and group mapping

IDBroker creates mappings between Cloudera users and groups (imported from corporate LDAP/AD, stored in UMS) and native cloud platform roles (e.g. in AWS: the IAM roles associated with policies). The mappings are specified in the Cloudera Control Plane and synchronised to deployed clusters.

IDBroker authentication delegation tokens lifetime

By default, IDBroker authentication delegation tokens, used to request cloud credentials, have a lifetime of 7 days. This lifetime can be adjusted by modifying the idbroker_knox_token_ttl_ms configuration property.

Authentication with Apache Knox

Apache Knox handles proxy for web UIs and APIs, and Trusted Proxy propagates the authenticated end user to the backend service.

Knox Gateway

Apache Knox Gateway is a reverse proxy that authenticates and provides a single access point for REST and HTTP interactions with the Cloudera Data Hub clusters.

Trusted proxy

Knox Trusted Proxy is useful in cloud deployments when you need the seamless and uniform authentication benefits of both proxy and SSO. Trusted Proxy is automatically configured by Cloudera Manager in Cloudera deployments.

Knox Trusted Proxy propagates the authenticated end user to the backend service. The request is "trusted" in that the given backend/service is able to validate that the request came from a certain place and was allowed to make the request. A backend in this case is any service that Knox is acting as a proxy for (e.g., Cloudera Manager, Hive JDBC, Ranger UI, etc). Each of these services have a mechanism to ensure that the 1) request IP address and 2) request user matches what it expects. If the request matches those two things, then the service will not have to authenticate again and can trust that Knox sent the request.

When making requests to the cluster, Knox first authenticates the end user, and then adds that user as a query parameter to the request (?doAs=USERNAME) to the backend. The backend then checks that the request is trusted (request IP and request user) and extracts the end user (USERNAME) from the query parameter. The backend service then does whatever is necessary as that backend user. Knox and the proxied services authenticate to each other via Kerberos.

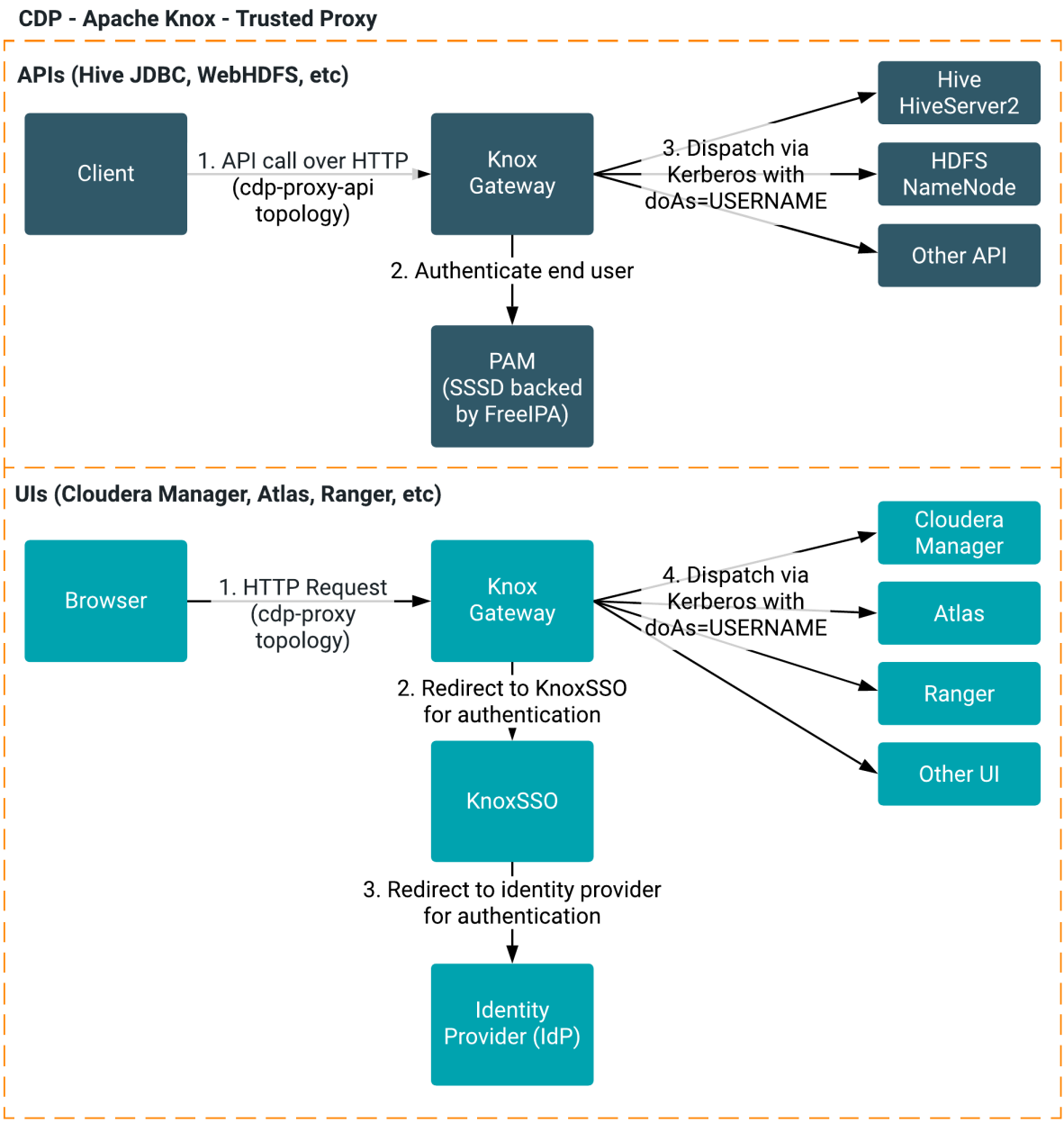

SSO via Knox

Knox SSO is configured automatically.

Related Information

Knox Supported Services Matrix

Access from Cloudera to customer resources

Cloudera creates clusters and runs jobs in your cloud provider account on your behalf.

Cloudera requires your permission to be able to use the resources required by the clusters and jobs in your cloud provider account. To allow Cloudera to create clusters or run jobs in your cloud provider account:

- Your AWS administrator must create a cross-account access IAM role and grant Cloudera access to the role as a trusted principal. The policy defined for the cross-account access IAM role must include permissions to allow Cloudera to create and manage resources and to perform the tasks and access the resources required for the Cloudera clusters and jobs.
- Your Azure account administrator must create an app registration and assign a role to it with permissions allowing Cloudera to create and manage resources and to perform the tasks and access the resources required for the Cloudera clusters and jobs.
- Your GCP account administrator must create a service account and assign permissions allowing Cloudera to create and manage resources and to perform the tasks and access the resources required for the Cloudera clusters and jobs.

For more information about credentials and security groups, refer to the following documentation:

Related Information

Role-based credential on AWS

App-based credential on Azure

Provisioning credential for Google Cloud

Default security group settings

Handling of sensitive data in Cloudera

Cloudera encrypts sensitive data (such as tokens, passwords, certificates, and encryption keys) by using Vault.

Classic cluster credentials

During HDP, CDH, or Cloudera Private Cloud "classic cluster" registration in the Cloudera Management Console, Cloudera asks you to enter cluster credentials. This is required for authenticating requests on the cluster side. Cloudera uses these credentials to access cluster APIs during and after cluster registration. During registration Cloudera stores these credentials securely in the vault and later whenever Cloudera makes an API call to the cluster, Cloudera reads these credentials from the vault and inject them inside the request.

Related Information

Enabling Cluster Connectivity Manager in Cloudera Management Console

Upgrading a classic cluster from Cluster Connectivity Manager v1 to Cluster Connectivity Manager v2

User access to clusters

Access to Cloudera Manager and other cluster UIs and endpoints (such as JDBC) is always via the secure Knox gateway on port 443. Users are automatically logged in with their Cloudera as a Service credentials.

Secure inbound communication

The Cloudera Control Plane communicates with workload environments for various command and control purposes. These connections currently go over the Internet to the workload environment hosts. Consequently, Cloudera deploys workloads into public (Internet routable) subnets.

To operate in public subnets, Cloudera secures in-bound communication to the listening ports using TLS encryption. Connections are authenticated using environment- or cluster-specific credentials that Cloudera manages internally. As appropriate, administrators can further secure listening ports by authenticating IP addresses and/or ports:

- Identifying authentic connections: Use security group rules to ensure inbound connections originate from the set of stable IP addresses that belong to your organization.
- Identifying authentic ports: Inbound connections take place on an identifiable set of ports, which can be used to additionally narrow the allowlist.

In addition to the Cloudera management traffic, your organization's use cases may specify connecting to the workload hosts over the Internet. These communications may involve different endpoints, protocols, ports, and credentials than Cloudera management traffic.

Endpoints fall into four categories today, associated with our shared Environment services on the Data Lake and on Cloudera Data Hub, Cloudera Data Warehouse, and Cloudera AI workloads. The following sections enumerate the communications for each of those categories.

Data Lake communication endpoints

In-bound communication for shared services support Cloudera management of FreeIPA and Data Lake services.

Each Environment contains a deployment of FreeIPA and a Data Lake cluster that support the following security and governance activities:

FreeIPA identity and security services

- LDAP: User Directory
- Kerberos KDC: Manages a Kerberos Realm for the Environment
- DNS: Provides internal resolution of workload hostnames
- Certificate Authority: Issues TLS certificates for internal use, and for the in-bound Cloudera control connections

Data Lake cluster services

- Hive Metastore: Tabular metadata storage
- Ranger: Security policies and audit trail
- Atlas: Data lineage, tagging, analytics
- ID Broker: Mapping of Cloudera identities to cloud provider identities
- Knox: A proxy gateway for access to cluster services
- Cloudera Manager: Local management for the data lake services

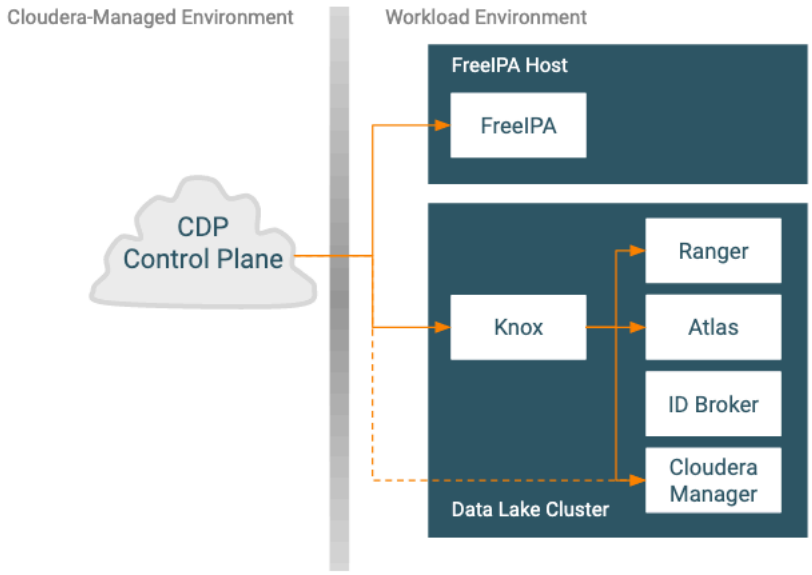

Communication to the Cloudera Control Plane includes the following:

FreeIPA identity and security services

- User and Group synchronization
- Service Principal management
- Retrieving the CA root certificate

Data Lake cluster services

- Core lifecycle management operations (sent directly between Cloudera and Cloudera Manager to include the Knox proxy as one of the managed entities; shown as a dashed line in the picture)
- Communication (via Knox proxy)
  - General Cloudera Manager operations
  - Ranger operations to manage repositories for workloads
  - ID Broker mappings (updated via Cloudera Manager)
  - Cloudera Data Catalog communicates with Atlas and Ranger to surface information

Cloudera Data Hub communication endpoints

Cloudera Data Hub clusters are built on the same underlying technology as the Data Lake cluster and so present a similar connectivity profile.

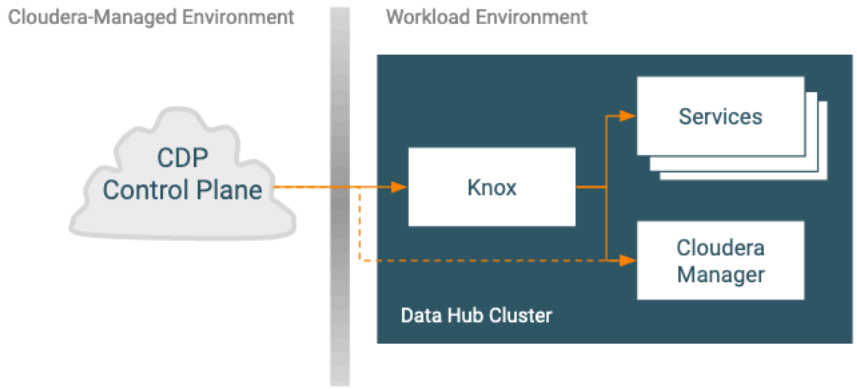

Communication to the Cloudera Control Plane includes the following:

FreeIPA identity and security services

- User and Group synchronization
- Service Principal management
- Retrieving the CA root certificate

Data Lake cluster services

- Core lifecycle management operations (sent directly between Cloudera and Cloudera Manager because the Knox proxy is one of the managed entities; shown as a dashed line in the previous picture)
- Communication (via Knox proxy)
  - General Cloudera Manager operations
  - Service-specific communication depending on the specific Cloudera Data Hub cluster in question

Cloudera Data Warehouse communication endpoints

The Cloudera Data Warehouse service operates significantly differently from Cloudera Data Hub, as it runs on top of a Kubernetes cluster and does not include a Cloudera Manager instance.

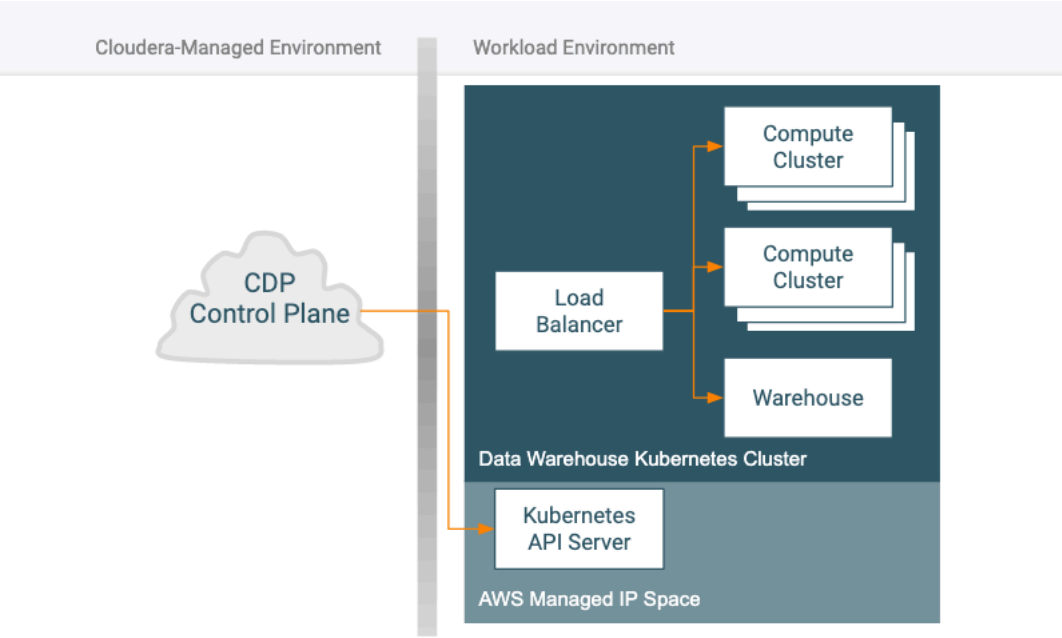

Primary command and control communication goes to the Kubernetes API server. This endpoint is specific to a particular Kubernetes cluster, but it is provisioned by the cloud provider outside of the customer VPC. Whether or not it is Internet-facing is independent of the VPC configuration. The Cloudera Data Warehouse service doesn't make connections to endpoints in the cluster.

Cloudera AI communication endpoints

In terms of communication, a Cloudera AI Workbench looks very similar to a Cloudera Data Warehouse workspace in that it is also a Kubernetes cluster, although the contents differ.

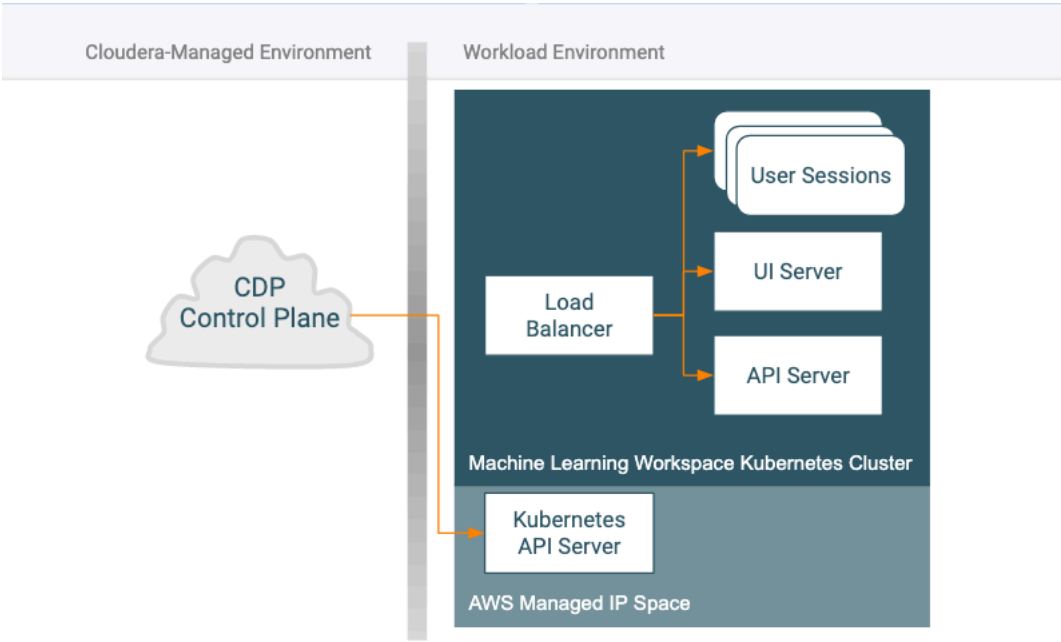

Primary command and control communication goes to the Kubernetes API server. This endpoint is specific to a particular Kubernetes cluster, but it is provisioned by the cloud provider outside of the customer VPC. Whether or not it is Internet-facing is independent of the VPC configuration. The Cloudera AI service doesn't make connections to endpoints in the cluster.

Cloudera Private Links Network Overview

Users who are concerned with privacy can utilize the Cloudera Private Links Network to establish private and secure connections from their workloads to the Cloudera Control Plane without using the public internet.

Cloudera Private Links Network is designed to provide seamless, private connectivity between your cloud workloads and the Cloudera Control Plane.

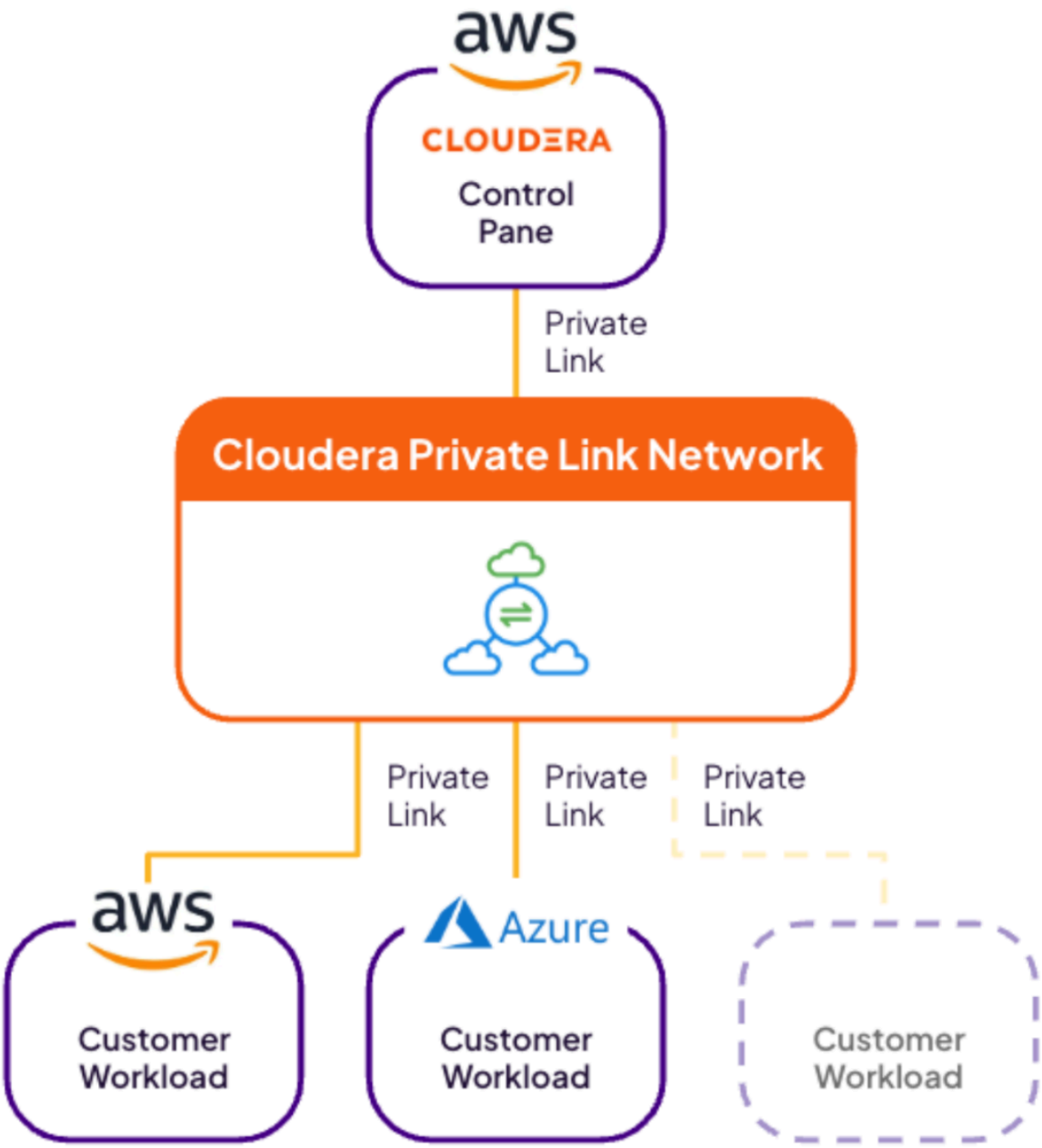

For more information, see Cloudera Private Links Network for AWS.

Cloudera Runtime Security and Governance

Cloudera Runtime security and governance is managed by Apache Ranger, Apache Knox, and Apache Atlas.

Data Lake cluster services

A Data Lake cluster is managed by Cloudera Manager, and includes the following services:

- Hive MetaStore (HMS) -- table metadata
- Apache Ranger -- fine-grained authorization policies, auditing
- Apache Atlas -- metadata management and governance: lineage, analytics, attributes

- Apache Knox:
  - Authenticating Proxy for Web UIs and HTTP APIs -- SSO
  - IDBroker -- identity federation; cloud credentials

Currently there is one Data Lake cluster for each CDP environment. Security in all DataHub clusters created in a Data Lake is managed by these shared security and governance services.

Links to the Atlas and Ranger web UIs are provided on each DataLake home page. A link to the Data Lake cluster Cloudera Manager instance provides access to Data Lake cluster settings.

Apache Ranger

Apache Ranger manages access control through a user interface that ensures consistent policy administration in Cloudera clusters.

Security administrators can define security policies at the database, table, column, and file levels, and can administer permissions for groups or individual users. Rules based on dynamic conditions such as time or geolocation can also be added to an existing policy rule. Ranger security zones enable you to organize service resources into multiple security zones.

Ranger also provides a centralized framework for collecting access audit history and reporting data, including filtering on various parameters.

Note: Authorization through Apache Ranger is just one element of a secure production cluster: Cloudera supports Ranger only when it runs on a cluster where Kerberos is enabled to authenticate users.

Apache Knox

The Apache Knox Gateway (“Knox”) is a system to extend the reach of Apache™ Hadoop® services to users outside of a Hadoop cluster without reducing Hadoop Security. Knox also simplifies Hadoop security for users who access the cluster data and run jobs. The Knox Gateway is designed as a reverse proxy.

Establishing user identity with strong authentication is the basis for secure access in Hadoop. Users need to reliably identify themselves and then have that identity propagated throughout the Hadoop cluster to access cluster resources.

Knox SSO provides web UI SSO (Single Sign-on) capabilities to Data Lakes and associated environments. Knox SSO enables users to log in once and gain access to Data Lake and DataHub cluster resources.

Knox IDBroker is an identity federation solution that provides temporary cloud credentials in exchange for various tokens or authentication.

Apache Atlas

Apache Atlas provides a set of metadata management and governance services that enable you to manage Cloudera cluster assets.

- Search and Proscriptive Lineage – facilitates pre-defined and ad hoc exploration of data and metadata, while maintaining a history of data sources and how specific data was generated.
- Ranger plugin for metadata-driven data access control.
- Flexible modeling of both business and operational data.
- Data Classification – helps you understand the nature of the data within Hadoop and classify it based on external and internal sources.

Apache Solr

Apache Solr is used as infrastructure by Ranger and Atlas to store and serve audit and governance information. Documentation refers to the Solr service Ranger and Atlas depends on as Infra Solr. You are recommended to separate other workloads from Infra Solr usage into a different Solr service. You find documentation on other supported workloads integrated with Solr under Cloudera Search.

Related Information

Data Lakes

Cloudera Public Cloud

Data Migration Tools and Methods

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Data Migration Tools and Methods Overview..... 5

Accelerate Your Migration to Cloudera with Workload Manager or Workload XM..... 5

Step 1 Identify Current and Potential Issues..... 6
Identifying Workload Problems and Health Issues..... 6
Identifying Resource Contention..... 8
Identifying Rogue Users from a Workload View..... 10
Identifying Resource-Hungry Workloads..... 12
Step 2 Create an Optimization Plan..... 13
Identifying and Correcting Inefficient SQL Code..... 14
Step 3 Capture Your Existing Baselines..... 15
Identifying Performance Trends..... 17

Use Cloudera Replication Manager to migrate to Cloudera Public Cloud..... 18

About Replication Manager..... 18
Fine-grained permission to access Cloudera Replication Manager..... 20
Providing role-based access control (RBAC) to Replication Manager users..... 20
Accessing Replication Manager UI..... 20
Access Replication Manager in Cloudera Public Cloud..... 21
Classic Clusters page..... 22
Cloud Credentials page..... 22
Replication Policies page..... 23
How replication policies work..... 25
HDFS replication policy..... 26
Hive replication policy..... 28
HBase replication policy..... 34
Using HDFS replication policies..... 39
Preparing to create an HDFS replication policy..... 40
Creating HDFS replication policy..... 43
Manage and monitor HDFS replication policies..... 48
Using Hive replication policies..... 51
Preparing to create a Hive replication policy..... 51
Creating Hive replication policy..... 55
Manage and monitor Hive replication policies..... 62
Using HBase replication policies..... 65
Preparing to create an HBase replication policy..... 65
Creating HBase replication policy..... 69
Manage and monitor HBase replication policies..... 77
Troubleshooting replication policies in Cloudera Replication Manager..... 82
Different methods to identify errors related to failed replication policy..... 82
Replication Policies page does not display all the replication policies..... 84
HDFS replication policy fails due to export HTTPS_PROXY environment variable..... 86
Cannot find destination clusters for HBase replication policies..... 86
HBase replication policy fails when Perform Initial Snapshot is chosen..... 87
Optimize HBase replication policy performance when replicating HBase tables with several TB data..... 87

Partition metadata replication takes a long time to complete....88
Replicating Hive nested tables....88
Target HBase folder is deleted when HBase replication policy fails....88
Replicate HBase data in existing and future tables....91
Appendix....91
Support matrix for Cloudera Replication Manager....92
Cloud credentials to use in Cloudera Replication Manager....97
Ports for Cloudera Replication Manager....112

Data Migration Tools and Methods Overview

Describes the Cloudera data migration services that help you to understand and optimize your existing workloads, clusters, and resources, plan your migration, and migrate your workload data.

Related Information

Accelerate Your Migration to Cloudera with Workload Manager or Workload XM

Use Cloudera Replication Manager to migrate to Cloudera Public Cloud

Accelerate Your Migration to Cloudera with Workload Manager or Workload XM

Overview of how Workload Manager and Workload XM can accelerate your migration from legacy CDH and HDP to Cloudera by reducing migration risks.

As part of your migration best practices, it is important to perform an in-depth analysis of your workloads before migrating to Cloudera. Workload Manager or Workload XM enable you to interactively understand your existing workloads, clusters, and resources. The wide range of metrics (collected by Telemetry Publisher or Databus Producer) and health tests help you identify and troubleshoot issues and the prescriptive guidance and recommendations help you quickly address and optimize those problems before or after migration. Finally, the performance baseline feature and historical analysis helps you identify and address performance problems and compare a workload's performance before and after your migration.

The following steps describe the tasks that you perform to plan and accelerate your migration to Cloudera. These steps provide you with an in-depth knowledge to make migration decisions by enabling you to identify which workloads are good candidates for cloud migration, plan which workloads, queries, and jobs require optimization and whether optimization is performed before or after migration, and identify and address performance problems by establishing baselines from health issues that will also enable a performance comparison of your workloads from before and after your migration:

- Identify current and potential issues and the priority in which your workloads are migrated from your legacy platform to a Cloudera form factor. This includes:
  - Monitoring and identifying issues in your existing clusters and recording their historical analysis.
  - Analyzing your existing resource consumption and resource contentions.
  - Identifying rogue users and resource hungry workloads.
- Plan which workloads, jobs, and queries require optimization and whether optimization is performed before or after migration.
- Enable a before and after migration workload performance comparison by capturing performance baselines.

Step 1 Identify Current and Potential Issues

Describes the tasks that help you discover which workloads to migrate and which workloads and users are currently containing issues and impacting costs, such as those resources that are consuming large amounts of CPU and memory.

Performing an in-depth analysis of your workloads helps you understand the level of effort required and helps you schedule and prioritize the order in which your workloads are migrated. Workload Manager and Workload XM identify existing and potential problems and their prescriptive guidance and recommendations enable you to quickly address and optimize those problems before or after migration.

The following tasks help you identify and analyze the state of your current workloads.

Identifying Workload Problems and Health Issues

This task identifies what workloads are running on your CDH and HDP clusters and if there are any health issues that exist or were encountered and not addressed.

About this task

Describes how to use Workload Manager or Workload XM to quickly identify resource and workload health issues.

Note: The health test metrics provided by Workload Manager and Workload XM are dependent on the workload's processing engine.

The following chart widgets quickly enable you to locate what engines that are running on your CDH and HDP clusters and what jobs and queries are failing the health tests:

- The cluster's Usage Analysis chart widget enables you to visually see what engines are running on the cluster, how many jobs or queries are processed by the engine, and how many jobs or queries have failed and missed their SLA.
- The Suboptimal chart widget enables you to visually see at a glance what issues are currently impacting your jobs or queries and how they are executing on your cluster.

  This chart uses the Workload Manager and Workload XM health tests and shows the distribution of jobs and queries that failed. The health tests are performed when a job or query has finished and provide insight into the performance of the job or query, such as how much data was processed and how long it took to process.

  The Health Check list, on the engine’s Jobs or Queries page, categorizes the health tests. For example, for an Impala engine, the Aggregation Spilled Partitions, HashJoin Spilled Partitions, and Slow Client health checks, test for resource health issues. For Hive, MapReduce, Oozie, and Spark engines, the Insufficient Partitioning and Many Materialized Columns health checks, test for query and schema issues, such as, is the code using SELECT * on millions of columns.

  These categories further enable you to understand the type of problem:

  - Metadata/Statistics
  - Optimal Configuration
  - Performance
  - Query/Schema Design
  - Skew
  - Baseline

For more information about each health check, see *RELATED INFORMATION*.

Procedure

1. In a supported browser, log in to the Workload Manager or the Workload XM web UI.
2. In the Clusters page, select the cluster required for analysis.
3. From the Usage Analysis chart widget, record the engines running workloads and note any issues.
4. From the Navigation panel or the Usage Analysis chart widget, click an engine that requires more analysis.

5. Scroll down to the engine's Suboptimal widget chart, as shown in the following example image. Record the health tests that contain a bar, which denote health tests that were violated.

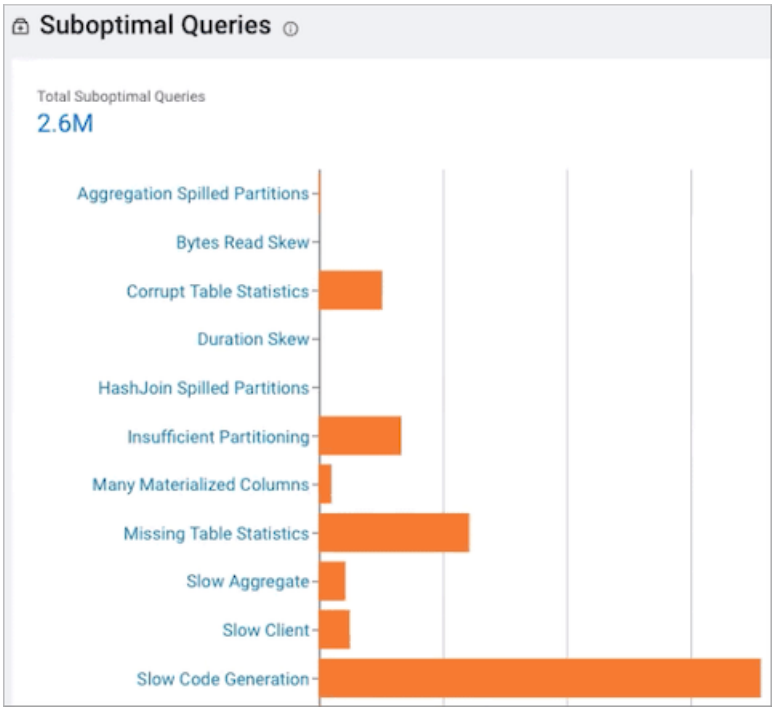

6. For more insights, click on any bar that displays a failed health check.

   The Jobs or Queries page opens.

7. Click a job or query with a Health Issue and then select the Health Checks tab.

   The Health Checks page is divided into three sections that display information about the job or query. Where:

   - The left section lists the health checks that were performed.
   - The middle section displays the stages of the job or query where a health check was performed. By default, all healthy stages are hidden and are revealed when a stage is selected.
   - The right section lists detailed information about a stage. Failed stages list the failed health check, the diagnosis, and recommendations.

8. Either record the diagnosis and prescriptive recommendation or use the recommendation to fix the issue.

Related Information

Hive, MapReduce, Oozie, and Spark Health Checks
Impala Health Checks

Identifying Resource Contention

This task identifies the resource usage and consumption of your workloads.

About this task

Describes how to use Workload Manager or Workload XM to identify workload resource consumption and contention problems by observing the types of resources and the amount of memory that is consumed.

Workload Manager and Workload XM provide the following chart widgets that help you analyze and identify resource consumption and contention problems:

- The Resource Consumption By Services chart widget, which is displayed in the cluster Summary page. It shows the CPU and memory consumption for each service across the time range you selected. Hover your mouse over the time line, to display the amount of CPU or memory, as a percentage, that is consumed by each of the cluster's services.

- The Resource Consumption By Nodes chart widget, which is displayed in the cluster Summary page. It shows the CPU and memory consumption for each node in the cluster. Hover your mouse over the time line, to display the amount of CPU or memory, as a percentage, that is consumed by each node and its services.
- The Memory Utilization chart widget, which is displayed in either the Impala engine or Workloads (Workload View) page. It shows the aggregated maximum amount of memory that is used by the queries on any node performing the processing. It helps you identify inefficient queries that are consuming the most amount of memory during processing and if you need to allocate more memory to continue running your queries.
- The Resource Consumption chart widget, which is displayed in either the Impala engine or Workloads (Workload View) page. It shows the concurrent use of CPU and memory consumption for a workload across the time line you selected.

Important: This task assumes that you have created workload views. For more information on how to create a Workload View, see *RELATED INFORMATION*.

Procedure

1. In a supported browser, log in to the Workload Manager or the Workload XM web UI.
2. In the Clusters page, select the cluster required for analysis.
3. From the Navigation panel, select Workloads.
4. In the Workloads page, select the required Workload.
5. Locate the Resource Consumption chart widget and identify any resource contentions by comparing the CPU Core Hours with Memory Usage.
6. Record any above average CPU and memory usage.

   For example, the following Resource Consumption chart widget image shows spikes when CPU and memory usage is high. This could be due to several job events occurring at the same time. The memory usage drops significantly around May 10th, which will need further investigation.

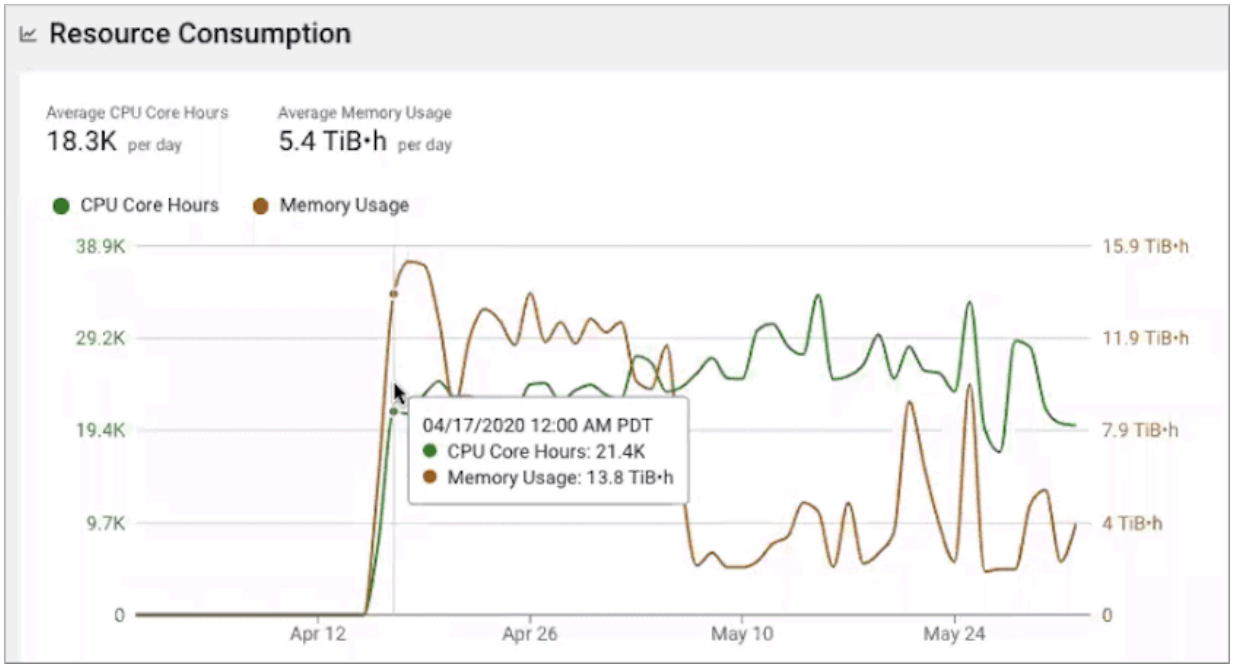

   To rule out any serious problems, Cloudera recommends monitoring your memory and CPU usage from one of the aforementioned resource chart widgets. As part of your migration plan, Cloudera recommends eliminating any resource contention before you migrate. Otherwise, these problems will resurface in CDP.

Related Information

Classifying Workloads for Analysis with Workload Views

Identifying Rogue Users from a Workload View

This task identifies rogue users, which are users that consume excessive amounts of resources that can impact your costs.

About this task

Describes how to use Workload Manager or Workload XM to identify rogue users.

Important: This task assumes that you have created workload views. For more information on how to create a Workload View, see *RELATED INFORMATION*.

Procedure

1. In a supported browser, log in to the Workload Manager or the Workload XM web UI.
2. In the Clusters page, select the cluster required for analysis.
3. From the Usage Analysis chart widget, select the required engine, such as Impala.
4. In the engine page, locate the engine Usage Analysis chart widget.
5. From the Top Users by list do the following:
   a. Select the # Queries option and identify any rogue users who are running more than the average share of jobs or queries.
   b. Select the Memory option and identify any rogue users whose jobs or queries are consuming the largest amount of memory.
   c. Select the CPU option and identify any rogue users whose jobs or queries are over exceeding the cluster's CPU.

6. Record any problems.

   For example, when comparing memory consumption with the number of queries assigned to all users in the following Usage Analysis chart widget you will notice that the 3xksle8z user is running only 3% of the queries assigned, yet their queries are consuming more memory (5.9 PiB) than any other user.

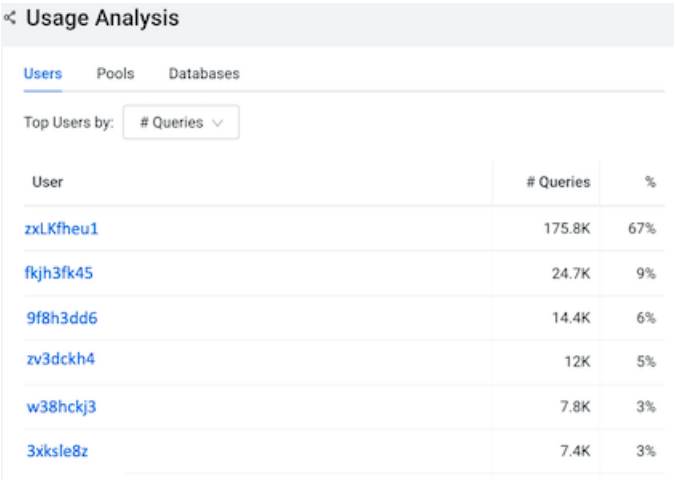

User	# Queries	%
zxLKfheu1	175.8K	67%
fkjh3fk45	24.7K	9%
9f8h3dd6	14.4K	6%
zv3dckh4	12K	5%
w38hckj3	7.8K	3%
3xksle8z	7.4K	3%

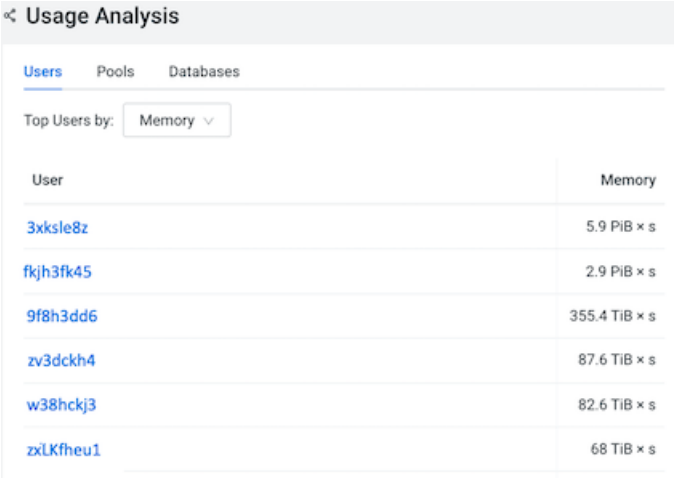

User	Memory
3xksle8z	5.9 PiB × s
fkjh3fk45	2.9 PiB × s
9f8h3dd6	355.4 TiB × s
zv3dckh4	87.6 TiB × s
w38hckj3	82.6 TiB × s
zxLKfheu1	68 TiB × s

To discover which queries executed by user 3xksle8z are consuming the largest amount of memory, a new workload view was created to track this user's resource consumption.

The following image shows the metrics of a SQL Statement that was identified as using an excessive amount of CPU from the new workload view, where there was over 200hrs CPU usage during a 34 hour period. Further

metrics are displayed that show performance issues, identified as aggregation issues, and that show there are 14 tables missing table statistics.

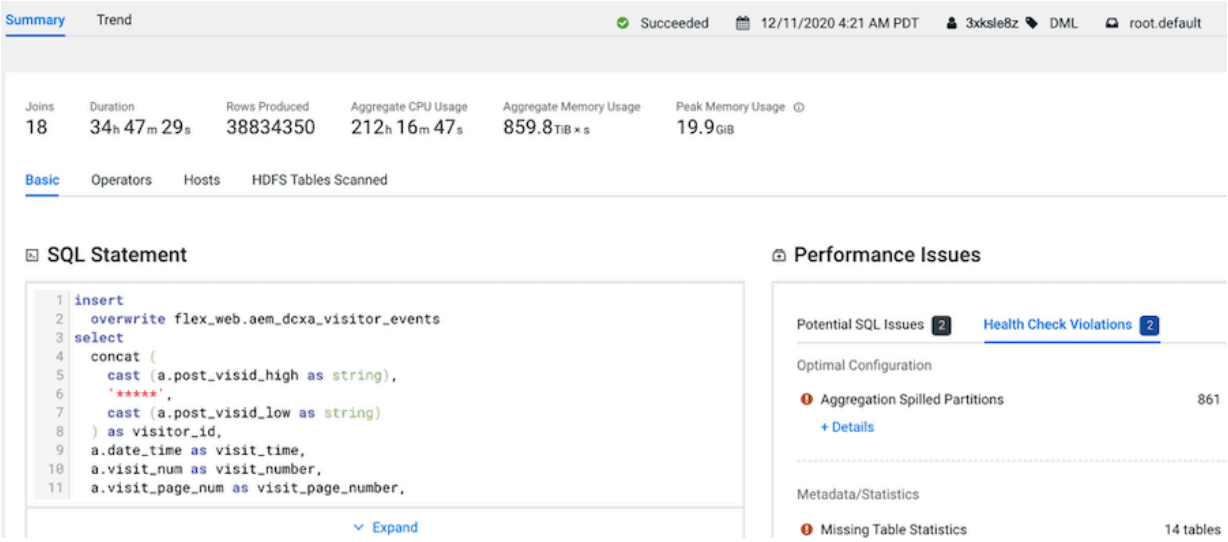

Related Information

Classifying Workloads for Analysis with Workload Views

Identifying Resource-Hungry Workloads

This task compares a workload with all other workloads in your cluster for identifying workloads that are using an excessive amount of resources.

Workload Manager and Workload XM enables you to compare the resource consumption of a particular workload or user against all other workloads or users in the cluster.

Important: This task assumes that you have created workload views. For more information on how to create a Workload View, see *RELATED INFORMATION*.

For example, you can compare the resource consumption of a specific workload against all the workloads in the cluster with the Resource Consumption chart widget, as shown in the following image:

When you compare the two chart metrics, the chart on the left shows that of a single workload that is consuming half the memory; 1.9 TiB verses 3.7 TiB.

You can delve deeper by creating a Workload View whose user criteria value is the user whose jobs are consuming an inordinate amount of memory and then create a Workload View whose user criteria value comprises all the users except the user that is of interest.

When you compare the two chart metrics over the same time frame, the chart on the left shows the memory consumption as 84 GiB and the CPU core hours as 6.1K compared to 3.7 TiB and 20K.

You could migrate this user to Cloudera and relieve some resources from your on-premises system.

Related Information

Classifying Workloads for Analysis with Workload Views

Step 2 Create an Optimization Plan

Describes how an Optimization Plan can accelerate your migration by defining which workloads, jobs, and queries to optimize before and which workloads to optimize after migration.

Your optimization plan should include a list of workloads or set of workloads and their existing status, including any current performance issues and health check violations, and the suggested improvement recommendations by Workload Manager or Workload XM. From your list of insights, decide which workloads, jobs, and queries require optimization before migration and which workloads, jobs, and queries require optimization after migration.

Your optimization plan should also consider:

- The type and severity of the issue
- The Cloudera form factor
- Your data infrastructure
- The number of clusters in your Cloudera environment
- Memory sizing considerations
- The before and after features, engines, and services

For example, during the processing of a SQL statement, the performance health tests discovered aggregate issues and missing table statistics.

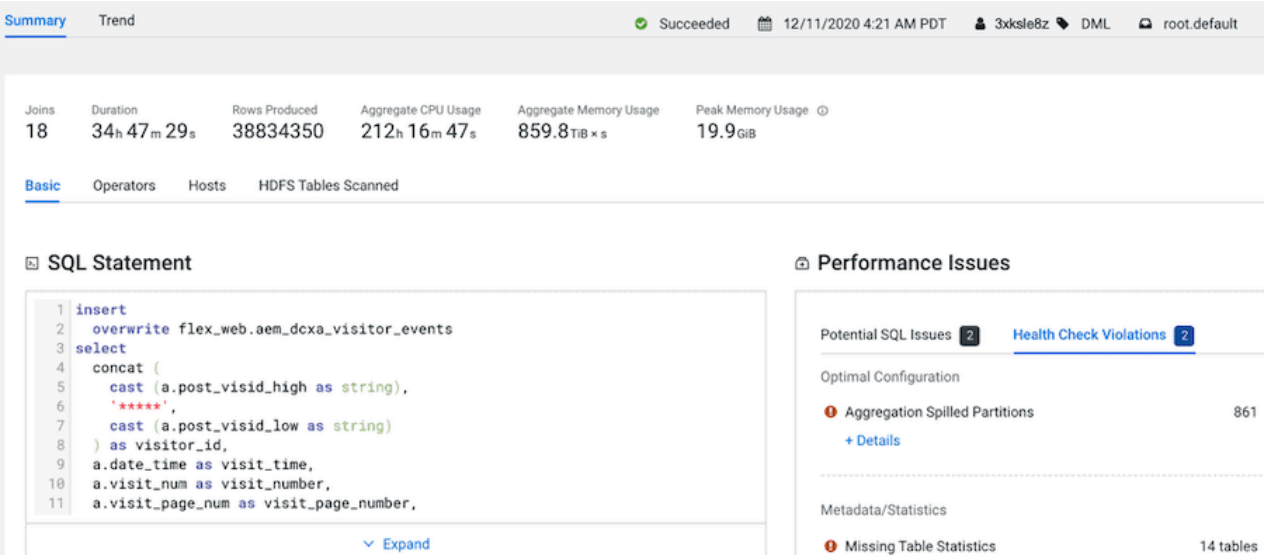

When a performance issue lists Aggregate Spilled Partitions as the cause it denotes memory issues. To ensure that all your SQL queries have sufficient memory when processing you should consider increasing your cluster's memory size. For this issue, the optimization would be solved as part of your Cloudera infrastructure and sizing considerations. Where you would use this information to research the costs of more memory for your Cloudera clusters as this would ensure enough memory for other such queries.

For the missing table statistics issue, it would also be more productive to collect the table statistic metadata after migration. Therefore, for this issue the optimization should be performed after migration.

Other considerations include:

- Understanding what features and engines are available in Cloudera.

  For example, users will have the ability to use Hive on TEZ and Hive LLAP in Cloudera, which are not available in CDH or HDP. Therefore, consider optimizing your Hive engine issues after migration.
- Your legacy applications and workloads that are at the end of their life cycle or those that would require development in order to work in Cloudera.
- Any issue or condition that would result in migrating garbage data. For example, you can immediately improve the processing performance of an Impala query by rewriting any poor SQL code, as described in the next topic *IDENTIFYING AND CORRECTING INEFFICIENT SQL CODE*.

Identifying and Correcting Inefficient SQL Code

Cloudera Observability dissects inefficient Impala SQL code and provides prescriptive tuning recommendations.

You can immediately improve the processing performance of an inefficient Impala query by rewriting any poor SQL code. For example, to prevent skewness, SQL operations that use JOINS clauses may require changes to the SQL code by selecting columns with the most evenly distributed values. Or, as in the following example, having too many joins and inline views are characteristic of inefficiently written SQL code.

This example uses the Queries chart widget, which provides the Duration, Complexity, CPU, and Memory options that help you identify SQL problems.

In this Impala engine case, several queries displayed a larger than average duration time in the Duration option of the Queries chart widget. When the Complexity option was selected it displayed two queries that contain a large number of joins, as shown in the following image:

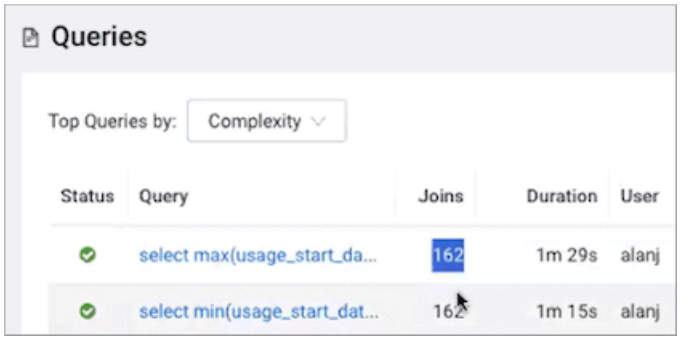

Even though Impala can process hundreds of joins in a minute, reducing an inordinate number of joins in an SQL statement will improve performance.

When we drill down further in Cloudera Observability by selecting one of the queries of interest, the Performance Issues, which are related to the SQL statement, display the Cloudera Observability prescriptive tuning recommendations. In this case, the recommendation is to denormalize the tables and materialize the inline views:

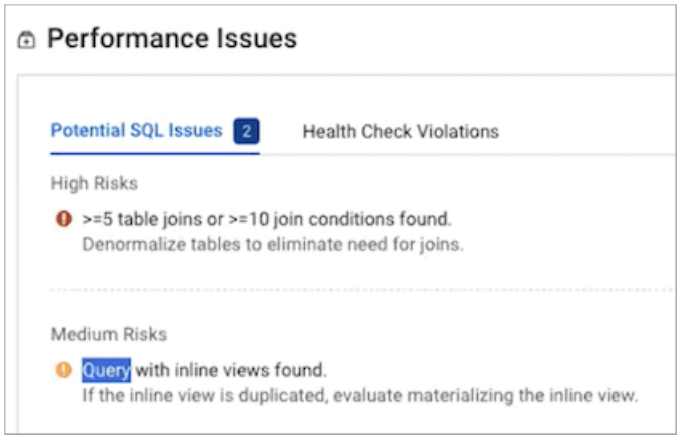

Step 3 Capture Your Existing Baselines

Describes how to create baselines that enable you to address performance problems and compare the performance of your workloads, jobs, and queries before and after migration.

Workload Manager and Workload XM baseline metrics measure the current performance of a job against the average performance of previous runs. They use performance data from 30 of the most recent runs of a job and require a minimum of three runs. The baseline comparisons start with the fourth run of a job.

Note: When a baseline is first created there will be comparison differences until more data is established.

Cloudera highly recommends that before you migrate, you establish and capture your workloads, jobs, and query baselines with Workload Manager or Workload XM. Baselines enable you to identify and address performance problems, as well as, enable you to compare the performances of your workloads, jobs, and queries before and after migration.

For more information on how to establish and display baselines and create job comparisons, see *RELATED INFORMATION*.

The following images show some of the baseline metrics that Workload Manager and Workload XM provide:

- This image shows a few of the performance metrics listed in the Baseline tab:

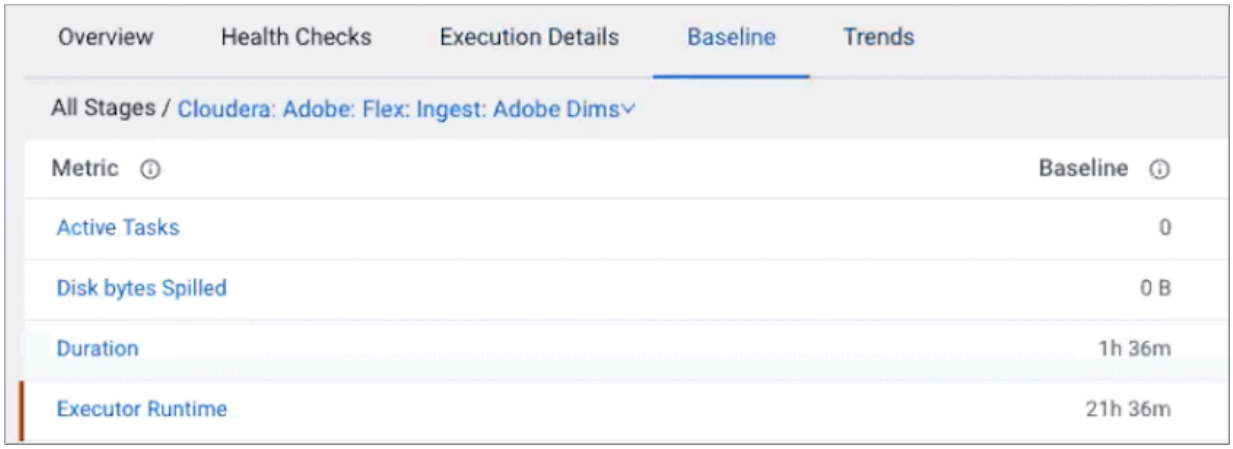

- This image shows the comparison between the baseline performance metrics and the current job run:

Metric	Baseline	Current Job	
Number of Task attempts	134	1K	+870
Number of Tasks	134	995	+861
Output bytes	728.5 MiB	0 B	-728.5 MiB
Output records	396.6K	0	-396.6K
Shuffle Read bytes	265.5 B	38.5 MiB	+38.5 MiB
Shuffle Read records	4.5	796K	+796K
Shuffle Write bytes	265.5 B	52.6 MiB	+52.6 MiB
Shuffle Write records	4.5	1.2M	+1.2M
Succeeded Task attempts	134	992	+858
Succeeded Tasks	134	992	+858
Total Task duration	24m 34s	2h 38m	+2h 13m

To capture a job's baseline performance:

1. In a supported browser, log in to the Workload Manager or the Workload XM web UI.
2. In the Clusters page, select the cluster required for analysis.
3. From the Usage Analysis chart widget, select the required engine, such as Spark.
4. From the Slow Jobs chart widget, click on a job and record its baseline and any prescriptive improvements that can be made.
5. If a baseline is not displaying, click Compare with Previous Run.
6. Capture the Job Comparison page and record any health issues.

You can also compare two different runs of the same job with the Job comparison feature.

To compare two different runs of the same job:

1. In a supported browser, log in to the Workload Manager or the Workload XM web UI.
2. In the Clusters page, select the cluster required for analysis.
3. In the Trend chart widget, select the tab of an engine whose jobs you want to analyze and then click its Total Jobs value.

   The engine's Jobs page opens.
4. List and display details of all the runs of a specific job by selecting one of the job runs and then, in the Jobs details page, click the Trends tab.

5. To compare two job runs, select the check boxes adjacent to the job runs you require and then click Compare.

   The Job Comparison page opens displaying more details about each job.

Before moving to Cloudera, it is highly recommended capturing those jobs that have a maximum impact on your CDH or HDP clusters and establishing their baselines. After migrating to Cloudera, capture those job baselines again with Workload Manager or Workload XM. If you see significant deviation between the CDH or HDP and Cloudera baselines, drill down further in Workload Manager or Workload XM to understand the effects of the migration.

You can also identify trends as well as baselines by analyzing your engine's or cluster's performance trends from the Trends chart widget and the Trend tab. As described in the next topic.

Related Information

Troubleshooting with the Job Comparison Feature

Identifying Performance Trends

This task identifies performance trends over a selected time range.

About this task

Describes how to use Cloudera Observability to identify performance trends.

You can identify trends as well as baselines by analyzing your engine's or cluster's performance trends from the Trends chart widget and the Trend tab. Where:

- The Trends time-series chart widget displays more detailed metrics about the processed jobs and queries and enables you to view historical trends for analysis when you select a predefined or custom time period from the Range filter list.
- The Trends tab displays the job or query's instances executed during the selected time range. Depending on the engine, the Trends page displays a job's historical trend from Duration, Data Input, and Data Output histogram charts or lists the runs of the query to show how its performance changes overtime.

Procedure

1. In a supported browser, log in to the Cloudera Observability web UI.
2. In the Clusters page, select the cluster required for analysis.
3. From the Trend chart widget, select the tab of each engine that has processed jobs or queries. Notice if there are any failed or slow jobs or queries.

4. For any Failed Jobs or Queries and for any slow jobs or queries, labeled Job Duration or Query Active Time, select the number under the aforementioned labels.

   The following image shows the Total Queries, Failed Queries, and Query Active Time labels from a Workload View's Trend chart widget:

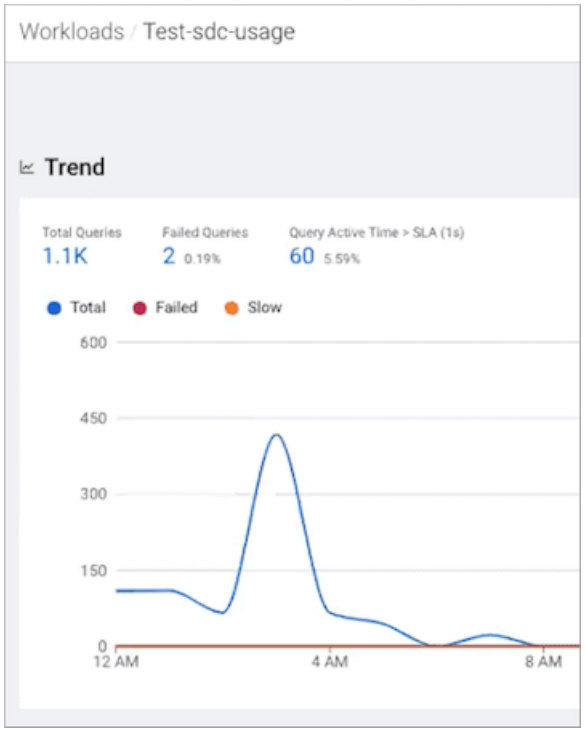

5. From either the Job or Queries page that opens, record those jobs or queries that are failing and their health issue.
6. For more insights, click a job or query and then select the Trend tab.
7. Record your findings.

Use Cloudera Replication Manager to migrate to Cloudera Public Cloud

Cloudera Replication Manager is a service to copy and migrate data from CDH 5.13+ and above clusters (HDFS, Hive, and HBase data) and Cloudera Private Cloud Base 7.1.4 and above clusters (HDFS , Hive external tables, and HBase data) to Cloudera Public Cloud clusters. The supported Public Cloud services include Amazon S3 or Microsoft Azure ADLS Gen2 (ABFS). Replication Manager from HDP clusters to Cloudera Public Cloud Azure is a beta feature and is not available for general use.

About Replication Manager

Replication Manager is a service in Cloudera Public Cloud. You can create replication policies in Replication Manager to copy and migrate data from CDH (version 5.13 and higher) clusters (HDFS, Hive, and HBase data) and Cloudera Private Cloud Base (version 7.1.1 and higher) clusters (HDFS, Hive external tables, and HBase data) to Cloudera Public Cloud clusters. You can also replicate HDFS data from cloud storage to classic clusters (CDH or Cloudera Private Cloud Base clusters), and Hive external tables to Data Hubs. The supported Public Cloud services include Amazon S3 and Microsoft Azure ADLS Gen2 (ABFS). Replicating Hive managed tables using Replication Manager from HDP clusters to Cloudera Public Cloud is a beta feature and is not available for general use.

Replication Manager can replicate between services where at least the source service or the destination service is a local service. The term *local service* refers to a service of a cluster managed by Cloudera Manager. In the on-

premises to on-premises replication scenarios, you create the replication policy in the target Cloudera Manager. It is not possible to create replication policies between two non-local services, services on separate peer clusters, or between cloud and a peer cluster's service. In these unsupported scenarios, you can create replication policies on the Cloudera Manager that has the service that you are replicating to/from.

Before you create replication policies, you must ensure that the source cluster is supported by Replication Manager. For more information, see Support matrix for Cloudera Replication Manager on page 92.

You can access the Replication Manager service on the Cloudera Public Cloud web interface. To replicate data between clusters, add the source on-premises clusters as classic clusters on the Management Console Clusters page, add/create one or more Cloudera Public Cloud SDX Data Lakes and/or Data Hubs, and then create the replication policies in Replication Manager. The **Replication Policies** page shows the progress and status of replication policy jobs. You can also use CDP CLI to create HDFS and Hive replication policies.

Replication Manager provides the following functionalities that you can use to accomplish your data replication goals:

HDFS replication policies

These policies replicate HDFS data and metadata from on-premises clusters (CDH, Cloudera Private Cloud Base, and HDP) to Public Cloud storage buckets such as S3 and ABFS, and from cloud storage to classic clusters (CDH or Cloudera Private Cloud Base clusters). You can choose the frequency of replicating data.

Some use cases where you can use HDFS replication policies include:

- Moving legacy data (from CDH clusters) to cloud deployments (AWS or Azure on Cloudera Public Cloud).
- Archiving cold data.
- Replicating the required data to another cluster to run analytics on it.

Hive replication policies

These policies support table-level replication and can replicate Hive external tables from on-premises clusters (CDH and Cloudera Private Cloud Base) to cloud storage such as S3 and ABFS and to Data Hubs. They also can:

- replicate data stored in Hive tables, Hive metadata, data in Hive metastore, and Impala metadata (catalog server metadata) associated with Impala tables registered in the Hive metastore, and

  Note: Hive2 managed tables are converted to external tables after replication.

- migrate Sentry permissions to Ranger.

  Note: To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.

Some use cases where you can use Hive replication policies include:

- Backing up data periodically.
- Performing a recovery operation when necessary.
- Creating a development and test system for engineers to run quality checks.

HBase replication policies

You can create these policies to replicate HBase data from a source classic cluster (CDH or Cloudera Private Cloud Base cluster), COD, or Data Hub to a target Data Hub or COD cluster. You can also copy or replicate HBase data between different environments within a Virtual Private Cloud (VPC) using these policies. Any future data change in the source cluster is pushed to the target cluster automatically without user intervention.

Some use cases where you can use HBase replication policies include:

- Performing an active-active disaster recovery with conflict resolution (enabling other disaster recovery use cases which provides an efficient utilization of resources).

- Copying required data to the cloud clusters for heavy-duty analytics workloads which helps to optimize on-premises cluster performance.
- Utilizing the continuous data synchronize feature to implement a hybrid cloud that in turn helps you to use it in various other use cases.

CDP CLI for HDFS and Hive replication policies

You can also use CDP CLI commands to create HDFS and Hive replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see *CDP CLI for Replication Manager*.

CDP CLI is a unified tool to manage all the CDP services. This gives you the flexibility to use it across services from a single pane, view required information in a single scroll (for example, you can view all available clusters' service status in a single page), and collaborate to troubleshoot issues.

Fine-grained permission to access Cloudera Replication Manager

You can restrict access to specific users to view and use Cloudera Replication Manager in a Cloudera Public Cloud environment so that you can govern the access to critical replication functionalities.

Currently, any user in a Cloudera Public Cloud environment can view and use Cloudera Replication Manager to create, run, and manage replication policies. However, in some deployments, it is essential that only a few authorized users have access to Replication Manager. This requirement arises when you want to provide an added layer of control which aligns with the best practices for data management and security, and also to enhance security and control over replication management which includes monitoring the replication jobs, and troubleshooting issues efficiently.

Providing role-based access control (RBAC) to Replication Manager users

You can provide fine-grained permissions to specific users to view and use Cloudera Replication Manager.

Procedure

1. Enable the RBAC entitlement. Contact your Cloudera account team to accomplish this task.
2. Identify the users that require access to Cloudera Replication Manager.
3. Go to the Cloudera Management Console User Management page. You can manage the role assignments on this page.

   Important: Administrators with the PowerUser role can add, modify, or delete users and groups in the Cloudera Public Cloud environment.

4. Assign the ReplicationAdmin role to one or more users, and Save the changes.

   Note: You must have the ReplicationAdmin or PowerUser role to use Replication Manager if the entitlement is enabled.

5. Optionally, create a group to manage the Replication Manager users and their roles. For example, replicationusers.

Accessing Replication Manager UI

You can access the Replication Manager user interface by logging into Cloudera Data Platform > Select Replication Manager.

When you log into Cloudera Data Platform, the Cloudera Public Cloud web interface appears. Click Replication Manager. The Overview page of the Replication Manager appears.

The Replication Manager has the following pages:

- Overview

- Classic Clusters
- Cloud Credentials
- Replication Policies

The following image shows the Cloudera Public Cloud web interface:

You can also access the Replication Manager service by logging into Management Console. In the Management Console, click and select Replication Manager.

Access Replication Manager in Cloudera Public Cloud

You can access the Replication Manager service by logging into Cloudera.

When you log into Cloudera Data Platform, the Cloudera Public Cloud web interface appears. Click Replication Manager to view the **Overview** page of the Replication Manager.

Replication Manager has the following pages:

- Overview
- Classic Clusters
- Cloud Credentials
- Replication Policies
- Snapshots

The following image shows the Cloudera Public Cloud web interface:

You can also access the Replication Manager service by logging into Cloudera Management Console. In Cloudera Management Console, click and select Replication Manager.

Classic Clusters page

The Classic Clusters page specifies the total number of clusters enabled for Replication Manager, the number of clusters that are in an error state, the number of clusters that are active, and the number of clusters for which a warning is issued.

You must register your existing on-premises Cloudera Distribution of Hadoop (CDH) and Cloudera Private Cloud Base clusters on the Cloudera Management Console, after which you can copy or move your data to the cloud. In Cloudera Public Cloud, these clusters are called *classic clusters*.

The Classic Clusters panel shows the following cluster status:

- **Active** clusters that are currently available to run the replication jobs.
- Clusters having less than 10% disk capacity appears as **Warning**. Click the number to open a table to track the cluster name and the exact disk capacity remaining.
- **Total** number of clusters that are in use.
- Number of clusters that are currently not running as expected or in **Error** state.

To investigate the issues associated with clusters that have an error or warning status, use Cloudera Manager for on-premises clusters.

Cloud Credentials page

The Cloud Credentials page shows the registered cloud credentials for Replication Manager. To replicate data to or from a storage cloud account, you must register the cloud credentials, so that the Replication Manager can access your cloud account. The supported cloud storage accounts are Amazon S3 and Azure Blob Filesystem (ABFS). On the Cloud Credentials page, you can add cloud credentials. You can also update or delete the credentials when necessary.

When you add cloud credentials for your Amazon S3 account, you can choose one of the following authentication methods:

- Access secret key. To use this authentication type, you require an AWS Access Key and an AWS Secret key that you obtain from Amazon. Cloudera Manager stores these values securely and does not store them in world-readable locations. The credentials are masked and encrypted in the configurations passed to processes managed by Cloudera Manager, and redacted from the logs.
- IAM role. Amazon Identity and Access Management (IAM) can be used to create users, groups, and roles for use with Amazon Web Services, such as EC2 and Amazon S3. IAM role-based access provides the same level of access to all clients that use the role.

Note: You can choose the IAM role authentication type only when the following conditions are met:

- The source cluster is hosted on an AWS EC2 infrastructure.
- The source cluster Cloudera Manager and all the nodes in the cluster are running on an EC2 instance.
- The source cluster Cloudera Manager has the same IAM role.

For information about configuring AWS credentials, see Introduction to role based provisioning credential in AWS.

You can perform the following tasks on the Cloud Credentials page to manage cloud credentials:

Add cloud credentials

You can add cloud credentials for your S3 or ABFS account. For information about adding cloud credentials, see Working with Cloud Credentials.

Note: Unregistered credentials can impact the replication process. Credentials associated with a cluster node that do not have updated credentials are called unregistered credentials. For example, if a node is down when the credentials are changed on a bucket or when the node is brought up that has the old credentials.

Update cloud credentials

You can update the cloud credentials based on various factors. When the bucket configuration such as secret or access keys, bucket name or endpoint, and encryption type is changed, it can affect the Replication Manager replication policy run and might require an update to the Replication Manager cloud credentials.

Credential changes are picked up by the next run of the policy. When you change the credentials, the in-progress policy runs might fail but the succeeding runs pick up the changes.

To update a cloud credential, click Actions Update .

Delete cloud credentials

You can delete unwanted credentials from the Replication Manager. When you delete cloud credentials, the replication policies that use the deleted cloud credentials might fail. To avoid failures, delete the Replication Manager cloud policies associated with the deleted credentials and recreate the policies with the new credentials. You can view a list of policies associated with specific credentials on the **Cloud Credentials** page.

To delete a cloud credential, click Actions Delete .

Replication Policies page

The "Replication Policies" page shows the number of replication policies that are active, the number of policies that have been suspended, the number of policies that are in error state, and the total number of replication policies available in Replication Manager. The page also provides a detailed view about the replication policies.

The Replication Policies page shows a replication policy status dashboard which shows the following panels and the number of policies with the status:

- **Error** shows the number of replication policies associated with a cluster designated as Error on the Classic Clusters map. Click the number to understand the policy names, the names of the source and destination clusters, and which services are stopped on the source or destination cluster.
- Active replication policies that are in Submitted or Running state. This item is not actionable.
- **Suspended** replication policies that have been suspended by the administrator. This item is not actionable.

- **Total** number of running policies.

Click a replication policy to view more details about the policy. Click Actions ( ) to perform more actions on a replication policy.

Additionally, you can perform the following tasks on the **Replication Policies** page:

- Change the timezone, if required, using .
- View the list of unreachable clusters using .

Replication policy details

You can also view the following policy details on the Replication Policies page:

- Current policy Status .
- Policy Type shows HDFS, Hive, or HBase.
- Replication policy Name .
- Source cluster name.
- Destination cluster name.
- Jobs that were run for the replication policy and its current status.
- Duration or time taken to run the policy.
- Last Success timestamp of the last successful run.
- Next Run timestamp of the next scheduled run.

Optimize Replication Policies page performance

By default, the replication policies are loaded only partially on the **Replication Policies** page, therefore the page might display incomplete statistics about a job status. This is because the job history is necessary to decide whether a policy failed or succeeded. The replication policies with failed jobs might take a longer time to load.

You can change the page load behavior depending on your requirements using . Choose one of the following options to load the **Replication Policies** page faster by delaying to load the job history:

- Delay loading job history when it takes too long attempts to load the job history, but omits the load operation above a certain threshold. By default, Replication Manager uses this option.
- Never load job history minimizes the load on Cloudera Manager and maximizes Replication Manager performance.
- Always load job history ensures that the job history is always loaded for all the displayed replication policies.

Use Case

Sometimes, Replication Manager fails to reach a healthy Cloudera Manager when there is a temporary networking blip or when there is a load spike on Cloudera Manager. When a cluster becomes unreachable for Replication Manager, the cluster is placed in the list of unreachable clusters (the list appears when you click). Replication Manager retries to reach the cluster again after 20 minutes. After you confirm that the Cloudera Manager is healthy and expect it to

be reachable by Replication Manager, you can force reload the **Replication Policies** page using to reconnect every cluster.

For more information, see Replication Policies page does not display all the replication policies.

How replication policies work

In Cloudera Replication Manager, you create replication policies to establish the rules you want applied to your replication jobs. The policy rules you set can include which cluster is the source and which is the destination, what data is replicated, what day and time the replication job occurs, the frequency of job runs, and bandwidth restrictions.

The first time you run a job (an instance of a policy) with data that has not been previously replicated, Replication Manager creates a new folder or database and bootstraps the data. During a bootstrap operation, all data is replicated from the source cluster to the destination. As a result, the initial execution of a job can take a significant amount of time, depending on how much data is being replicated, network bandwidth, and so on. So you should plan the bootstrap operation accordingly.

After the bootstrap operation succeeds, an incremental replication is automatically performed for data replication. This job synchronizes, between the source and destination clusters, any events that occurred during the bootstrap process. After the data is synchronized, the replicated data is ready for use on the destination. Data is in a consistent state only after incremental replication has captured any new changes that occurred during bootstrap.

Subsequent replication jobs from the same source location to the same target on the destination are incremental data replication, so only the changed data is copied.

When a bootstrap operation is interrupted, such as due to a network failure or an unrecoverable error, the Replication Manager does not retry the job instead it runs the job at the next scheduled interval, if available. Therefore, if the bootstrap operation is interrupted, you must manually correct the issue and then run the policy.

When scheduling how often you want a replication job to run, you should consider the recovery point objective (RPO) of the data being replicated; that is, what is the acceptable lag time between the active site and the replicated data on the destination.

Replication policy considerations

You should take into consideration certain guidelines when creating or modifying a replication policy. It is important for you to understand the security restrictions and encryption policies within Replication Manager.

The guidelines you need to consider before you create or modify a replication policy includes:

Data security

To use an S3 or ABFS cluster for your policy, register your credentials on the Cloud Credentials page. A user with access to the Replication Manager user interface has the ability to browse, within the Replication Manager UI, the folder structure of any clusters enabled for Replication Manager.

Therefore, users can view folders, files, and databases in the Replication Manager user interface that they might not have access to in HDFS. Users cannot view from the Replication Manager UI the content of files on the source or destination clusters. Nor do these administrators have the ability to modify or delete folders or files that are viewable from the Replication Manager UI.

Policy properties and settings

Consider the recovery point objective (RPO) of the data being replicated when you schedule a replication policy. The RPO is the acceptable lag time between the active site and the replicated data on the destination. Ensure that the frequency is set so that a job finishes before the next job starts.

Jobs based on the same policy cannot overlap. If a job is not completed before another job starts, the second job does not execute and is given the status Skipped. If a job is consistently skipped, you might need to modify the frequency of the job.

Specify bandwidth per map, in MBps. Each map is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value.

Cluster requirements

- Pair the clusters before you include them in a replication policy.
- With a single cluster, you can replicate data on-premises to cloud.
- With a single cluster, you cannot replicate data on-premises to on-premises.
- If the clusters are Replication Manager-enabled, it appears in the Source Cluster or Destination or Data Lake Cluster fields in the Create Policy wizard. You must ensure that the clusters you select are healthy before you start a policy instance (job).

Hive restrictions

- When creating a schedule for a Hive replication policy, you should set the frequency so that changes are replicated often enough to avoid overly large copies.
- ACID tables, managed tables, storage handler-based tables such as Apache HBase, and column statistics are not replicated. ACID tables and managed tables are converted to external tables after replication.

HDFS replication policy

You can use the HDFS replication policies in Cloudera Replication Manager to replicate HDFS data. The HDFS replication policies can replicate HDFS data and metadata from classic clusters (CDH, Cloudera Private Cloud Base, and HDP) to Cloudera Public Cloud storage buckets such as S3 and ABFS, and from cloud storage to classic clusters (CDH or Cloudera Private Cloud Base clusters). To use an on-premises cluster (CDH or Cloudera Private Cloud Base cluster) in the replication policy, you must register it as a classic cluster in the Management Console. To use the cloud storage for data replication, you must register the cloud credentials in Replication Manager so that the Replication Manager service can access the cloud storage. You must also verify cluster access and configure minimum ports for replication before you create HDFS replication policies.

Before you create replication policies, see Support matrix for Replication Manager on Cloudera Public Cloud to verify whether your clusters are supported by Replication Manager.

You can also use CDP CLI commands to create HDFS replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see CDP CLI for Replication Manager.

HDFS snapshots

You can schedule taking HDFS snapshots for replication in the Replication Manager. HDFS snapshots are read-only point-in-time copies of the filesystem. You can enable snapshots on the entire filesystem, or on a subtree of the filesystem. In Replication Manager, you take snapshots at a dataset level. Understanding how snapshots work and some of the benefits and costs involved can help you to decide whether or not to enable snapshots.

To improve the performance and consistency of HDFS replications, enable the HDFS replication source directories for snapshots, and for Hive replications, enable the Hive warehouse directory for snapshots. For more information, see HDFS snapshots.

Enabling snapshots on a folder requires HDFS admin permissions because it impacts the NameNode. When you enable snapshots, all the subdirectories are automatically enabled for snapshots as well. So when you create a snapshot copy of a directory, all content in that directory including the subdirectories is included as part of the copy. If a directory contains snapshots but the directory is no longer snapshot-enabled, you must delete the snapshots before you enable the snapshot capability on the directory.

Take snapshots on the highest-level parent directory that is snapshot-enabled. Snapshot operations are not allowed on a directory if one of its parent directories is already snapshot-enabled (snapshottable) or if descendants already contain snapshots.

For example, in the following directory tree image, if directory-1 is snapshot-enabled but you want to replicate subdirectory-2, you cannot select only subdirectory-2 for replication. You must select directory-1 for your replication policy.

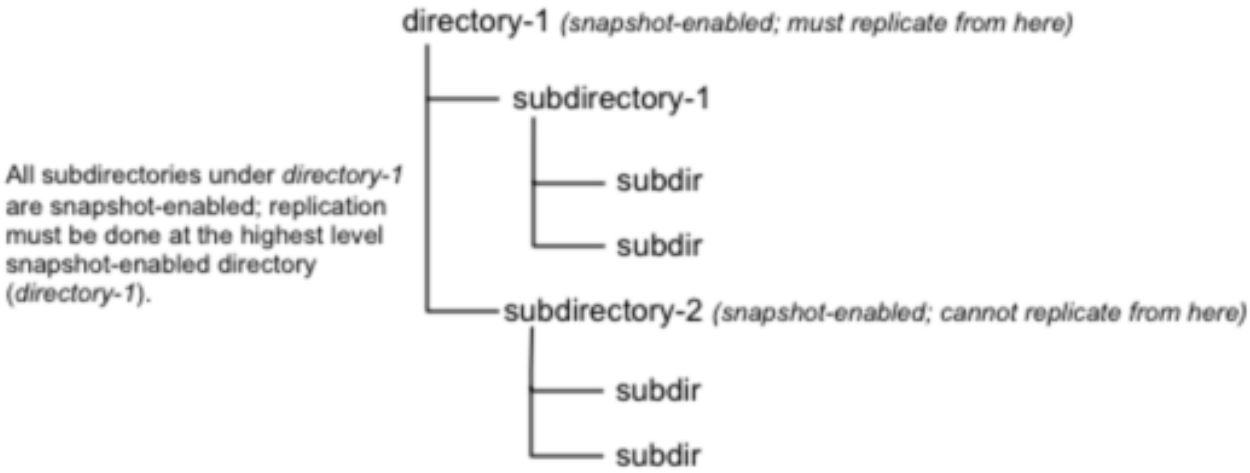

There is no limit to the number of snapshot-enabled directories you can have. A snapshot-enabled directory can accommodate 65,536 simultaneous snapshots. Blocks in datanodes are not copied during snapshot replication. The snapshot files record the block list and the file size. There is no data copying.

When snapshots are initially created, a directory named .snapshot is created on the source and destination clusters under the directory being copied. All snapshots are retained within the .snapshot directories. By default, the last snapshot of a file or directory is retained. Snapshots older than this are automatically deleted. You can configure the number of snapshots to retain when you create or edit an HDFS Snapshot Policy in the target Cloudera Manager using Cloudera Private Cloud Base Replication Manager 7.1.1 or higher or in Backup and Disaster Recovery 5.10 or higher depending on your on-premises target cluster.

Requirements and benefits of HDFS snapshots

You might want to consider the benefits and memory cost of using snapshots. Verify the requirements before you enable snapshots.

Requirements

You must have HDFS superuser privilege to enable or disable snapshot operations. Replication using snapshots requires that the target filesystem data being replicated is identical to the source data for a given snapshot. There must be no modification to the data on the target. Otherwise, the integrity of the snapshot cannot be guaranteed on the target and replication can fail in various ways.

Benefits

Snapshot-based replication helps you to avoid unnecessary copying of renamed files and directories. If a large directory is renamed on the source side, a regular DistCp update operation sees the renamed directory as a new one and copies the entire directory.

Generating copy lists during incremental synchronization is more efficient with snapshots than using a regular DistCp update, which can take a long time to scan the whole directory and detect identical files. And because snapshots are read-only point-in-time copies between the source and destination, modification of source files during replication is not an issue, as it can be using other replication methods.

A snapshot cannot be modified. This protects the data against accidental or intentional modification, which is helpful in governance.

Memory cost

There is a memory cost to enable and maintain snapshots. Tracking the modifications that are made relative to a snapshot increases the memory footprint on the NameNode and can therefore stress NameNode memory. Because of the additional memory requirements, snapshot replication is

recommended for situations where you expect to do a lot of directory renaming, if the directory tree is very large, or if you expect changes to be made to source files while replication jobs run.

Enabling and taking snapshots in Cloudera Manager

Before you take snapshots (in Cloudera Manager) or create snapshot policies (in Replication Manager) for HDFS directories, you must enable snapshots for the directories in Cloudera Manager.

Procedure

1. To enable snapshots for HDFS directories, navigate to the directory on the Cloudera Manager Clusters HDFS service File Browser tab, and click Enable Snapshots.

   Note: If you enable snapshots for a directory, you cannot enable snapshots for its parent directory. Snapshots can be taken only on directories that have snapshots enabled.

   Tip: To disable snapshots for a directory that has snapshots enabled, click Disable Snapshots. Ensure that the snapshots of the directory are deleted before you disable snapshots for the directory.

2. To take a snapshot of a directory or table, perform the following steps:
   a) Navigate to the directory or folder.
   b) Click Take Snapshot in the drop-down menu next to the full file path.
   c) Specify a unique name for the snapshot.

   The snapshot name with the timestamp of its creation appears in the snapshot list.

3. Click Actions Delete to delete a snapshot.

   Note: After you delete a snapshot, you can restore it, if required.

Results

Create a snapshot policy in the Replication Manager to schedule taking a snapshot of the directory at regular intervals.

Hive replication policy

You can create a Hive replication policy in Cloudera Replication Manager after you configure the required Ranger policy in Ranger, register the on-premises cluster (CDH or Cloudera Private Cloud Base) as a classic cluster in Management Console, register cloud account credentials in the Replication Manager service, verify cluster access, and configure minimum ports for replication. The replication load happens on the source on-premises cluster. You can replicate data on-premises to the cloud with a single cluster if the Metastore is running on the cloud.

Replication Manager supports replication of the Hive database from a cluster with underlying HDFS to another cluster with cloud storage. It uses push-based replication, with the replication job running on the cluster with HDFS. For information about Hive cloud replication, see Hive cloud replication

These policies support table-level replication and can replicate Hive external tables from on-premises clusters (CDH and Cloudera Private Cloud Base) to cloud storage such as S3 and ABFS and to Data Hubs. They also can:

- replicate data stored in Hive tables, Hive metadata, data in Hive metastore, and Impala metadata (catalog server metadata) associated with Impala tables registered in the Hive metastore, and

  Note: Hive2 managed tables are converted to external tables after replication.

- migrate Sentry permissions to Ranger.

  Note: To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.

Hive metadata replication involves multiple entities. Replication Manager supports replication of external tables in Hive. Hive supports replication of external tables to the target cluster and it retains all the properties of external

tables. The data files permission and ownership are preserved so that the relevant external processes can continue to write in it even after failover.

Important: Before you create replication policies, see Support matrix for Cloudera Replication Manager on page 92 to verify whether your clusters are supported by Replication Manager.

You can also use CDP CLI commands to create Hive replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see *CDP CLI for Replication Manager*.

The Apache Ranger access policy model consists of the following components:

- Specification of the resources that you can apply to a replication policy which includes the HDFS files and directories; Hive databases, tables, and columns; and HBase tables, column-families, and columns.
- Specification of access conditions for specific users and groups.

You must set the Ranger policy for the hdfs user on the target cluster to perform all operations on all databases and tables. The same user role is used to import Hive Metastore. The hdfs user should have access to all Hive datasets, including all operations. Otherwise, Hive import fails during the replication process.

On the target cluster, the hive user must have Ranger admin privileges. The same hive user performs the metadata import operation.

For more information about Hive replication policies to replicate data from CDH clusters to Cloudera Public Cloud, see Migrate Hive data from CDH to Cloudera Public Cloud blog.

Hive replication

Replication Manager allows you to replicate Hive databases from a source cluster to a target location on a destination cluster. The first time you run a job with data that has not been previously replicated, the Replication Manager creates a new folder or database and bootstraps the data. To replicate Hive metadata, Replication Manager performs a full replication. To replicate the data stored in Hive tables, Replication Manager uses snapshot diff-based replication to perform incremental replication.

During bootstrap operation, all of the data from the source location is copied to the destination. This bootstrapping of data can take hours to days, depending on factors such as the amount of data being copied and available network bandwidth.

After the bootstrap operation succeeds, an incremental replication is automatically performed for data replication using snapshot diff-based replication. The job synchronizes, between the source and destination clusters, any events that occurred during the bootstrap process. After the data is synchronized, the replicated data is ready for use on the destination. Data is in a consistent state only after incremental replication has captured any new changes that occurred during bootstrap.

Subsequent replication jobs from the same source location to the same target on the destination are incremental, so only the changed data is copied.

If a bootstrap operation is interrupted, such as due to a network failure or an unrecoverable error, the Replication Manager automatically retries the job. If a retry succeeds, the replication job continues from the point at which it was interrupted. If the automatic retries are not successful, you must manually correct the issue before running the policy again. When you activate the policy again, the replication job resumes from the point at which it was suspended.

Functions such as User Defined Functions (UDF) in Hive are replicated. To enable this, UDFs have to be created using a syntax. An example of UDF creation syntax:

```
CREATE FUNCTION [db_name.]function_name AS class_name  USING JAR|FILE|ARCHIV
E 'file_uri' [, JAR|FILE|ARCHIVE 'file_uri'] ;
```

- ACID tables, external tables, storage handler-based tables (such as HBase), and column statistics are currently not replicated.
- When creating a schedule for a Hive replication policy, you should set the frequency so that changes are replicated often enough to avoid overly large copies.

Snapshot diff-based replication

By default, Replication Manager uses snapshot differences ("diff") to improve performance by comparing HDFS snapshots and only replicating the files that are changed in the source directory.

While Hive metadata requires a full replication, the data stored in Hive tables takes advantage of snapshot diff-based replication. To replicate a database using a Hive replication policy, ensure that all the HDFS paths for the tables in that database are either snapshottable or under a snapshottable root. For example, if the database that is being replicated has external tables, all the external table HDFS data locations should be snapshottable too. Failing to do so can cause the Replication Manager to fail to generate a diff report. Without a diff report, Replication Manager will not use snapshot diff.

An HDFS directory is referred to as snapshottable if an administrator - having superuser privilege or having owner access to the directory - has enabled snapshots for the directory in Cloudera Manager.

You must ensure that the following guidelines are met for efficient incremental replication:

- HDFS snapshots are immutable.

  Tip: In the source Cloudera Manager, go to Clusters *HDFS SERVICE* Configuration section, and search for Enable Immutable Snapshots.

- Snapshot root directory is set to as low in the hierarchy as possible.
- Replication Manager user is a super user or the owner of the snapshottable root. This is because the user specified in the Run-as-username field in the replication policy must have the permission to list the snapshots.
- Paths from both source and destination clusters in the replication policy are under a snapshottable root or are snapshottable for the replication policy to run using snapshot diff.

Replication Manager performs a complete replication when one or more of the following change: Delete Policy, Preserve Policy, Target Path, or Exclusion Path.

Note: Ensure the source data does not contain an encrypted subdirectory. This is because snapshot diff-based replication might fail if an encrypted subdirectory exists in the source data.

Hive tables

Managed tables are Hive owned tables where the entire lifecycle of the tables' data are managed and controlled by Hive. External tables are tables where Hive has loose coupling with the data. Replication Manager replicates external tables successfully to a target cluster, and the Hive2 managed tables are converted to external tables after replication.

Hive supports replication of external tables with data to target cluster and it retains all the properties of external tables. The data files' permission and ownership are preserved so that the relevant external processes can continue to write in it even after failover.

The writes on external tables are performed using the Hive SQL commands and the data files can also be accessed and managed by processes outside of Hive. If an external table or partition is dropped, only the metadata associated with the table or partition is deleted but the underlying data files stay intact. A typical example for an external table is to run analytical queries on HBase or Druid owned data using Hive, where the data files are written by HBase or Druid and Hive reads them for analytics.

Important: Hive Materialized Views replication is not supported. However, Replication Manager does not skip it during replication and the replicated data might not work as expected in the target cluster.

When you create a schedule for a Hive replication policy, set the frequency so that changes are replicated often enough to avoid overly large copies.

You might come across the following use cases during Hive replication:

Replication Manager upgrade use case

In a normal scenario, if you have external tables that are replicated as managed tables, after the upgrade process, you must drop those tables from the target cluster and set the base directory. In the next instance, they get replicated as external tables.

Conflicts in external tables' data location for multiple source clusters replication to the same target cluster

To handle the conflicts in external tables' data location for multiple source clusters replication to the same target cluster, the Replication Manager assigns an unique base directory for each source cluster under which the external tables' data from the corresponding source cluster is copied.

For example, if the external table location in a source cluster is /ext/hbase_data, then the location in the target cluster after replication is <base_dir>/ext/hbase_data. You can use the DESCRIBE TABLE command to track the new location of external tables.

Replication conflicts between HDFS and Hive external table location

When you run the Hive replication policy on an external table, the data is stored on the target directory at a specific location. Next, when you run the HDFS replication policy which tries to copy data at the same external table location, Replication Manager ensures that the Hive data is not overridden by HDFS.

For example, when you run a Hive replication policy on an external table, the policy creates a target directory /tmp/db1/ext1. When you run an HDFS replication policy, the policy should not override the data by replicating on the /tmp/db1/ext1 directory.

Conflicts during external tables replication process

Conflicts appear when two Hive replication policies on DB1 and DB2 (either from the same source cluster or different source clusters) have external tables that point to the same data location (for example, /abc) and are replicated to the same target cluster. To avoid such conflicts, you must set different paths for the external table base directory configuration, for both the policies.

For example, set /db1 for DB1 and /db2 for DB2. This ensures that the target external table data location is different for both databases. For example, /db1/abcd and /db2/abcd.

Note: Replication conflicts are not supported for on-premises to cloud scenario.

Hive cloud replication

Replication Manager supports replication of the Hive database from a cluster with underlying HDFS to another cluster with cloud storage. It uses push-based replication, with the replication job running on the cluster with HDFS.

Hive stores its metadata in Hive Metastore, but the underlying data is stored in HDFS or cloud storage. In a Hadoop cluster with Hive service, the Hive warehouse directory can be configured with either HDFS or cloud storage.

You can perform the following tasks with Hive replication:

- You can rename the dataset in the policy that is replicated.
- You can create a pull-based policy on the source cluster to move data from the target back into the source cluster Hive database.

Hive replication from an HDFS-based cluster to a cloud storage-based cluster requires the following components:

- Source cluster - The cluster with a Hive warehouse directory on local HDFS. This can be an on-premises cluster or an IaaS cluster with data on local HDFS. The required services are HDFS, YARN, Hive, Ranger, Knox, and DLM Engine.
- Destination cluster - The cluster with data on cloud storage. The cluster minimally requires Hive Metastore, Ranger, Knox, and DLM Engine.

Replication Manager does not manage Ranger policies and any Personally Identifiable Information (PII) or any other secure data that gets replicated from on-premises to Amazon S3. You must manage these items outside of Replication Manager.

Table-level replication

To enable table-level replication, you must specify the list of tables to be replicated in a given replication policy. Table-level replication enables you to replicate just the critical tables. It also helps you to speed-up the replication process and also reduces network bandwidth utilization.

You can define table-level replication using regular expressions. You can include or exclude tables in a database for Hive replication during the Hive replication policy creation process.

The following examples illustrate how you can include or exclude Hive tables in the Hive replication policy:

- To include only table1, table2, and table3 in a database for replication, enter the database name in the Database field, and then enter table1|table2|table3 in the Tables field.
- To exclude table5, table7, and table9 and include the rest of the tables in the database, enter the database name in the Database field, and then enter (?!table5|table7|table9).+ in the Tables field.

Limitations using table-level replication

- If a table is dynamically added for replication due to changes in regular expression or added to the include list, the tables' data may not be point-in-time consistent with other tables which are already replicated incrementally. However, this inconsistency is seen for a very small duration until the completion of the next incremental replication after tables are added in the bootstrapped manner.
- Hive does not support overlapping replication policies such as db., db.[t1], and *. to the same target database but works as expected if the target databases are different.

Migrate Sentry authorization policies into Ranger

During Hive replication, Replication Manager migrates Sentry authorization policies into Ranger as part of the replication policy.

The Sentry service serves authorization metadata from the database-backed storage but does not handle actual privilege validation. The Hive and Impala services are clients of this service and it enforces Sentry privileges when the services are configured to use Sentry. Replication Manager allows administrators to migrate the existing Sentry permissions from the source CDH cluster to the Ranger policies in Cloudera Public Cloud.

When you create a replication policy, you can choose to migrate the Sentry policies for the resources that you want to migrate. During replication policy job run, the resources and its Sentry policies are migrated to the destination cluster. To migrate the Sentry policies for the resources, select the Include Sentry Permissions with Metadata option in the Additional Settings page of the Create Replication Policy wizard.

Note: To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.

The Sentry Permissions section of the Create Replication Policy wizard has the following options:

- Include Sentry Permissions with Metadata migrates the Sentry permissions during the replication job run.
- Exclude Sentry Permissions with Metadata ensures that the Sentry permissions are not migrated during the replication job.
- Choose Skip URI Privileges if you do not want to include URI privileges when you migrate Sentry permissions. During migration, the URI privileges are translated to point to an equivalent location in S3. If the resources have a different location in Amazon S3, do not migrate the URI privileges because the URI privileges might not be valid.

The following image shows the Sentry Permissions section in the Create Replication Policy wizard:

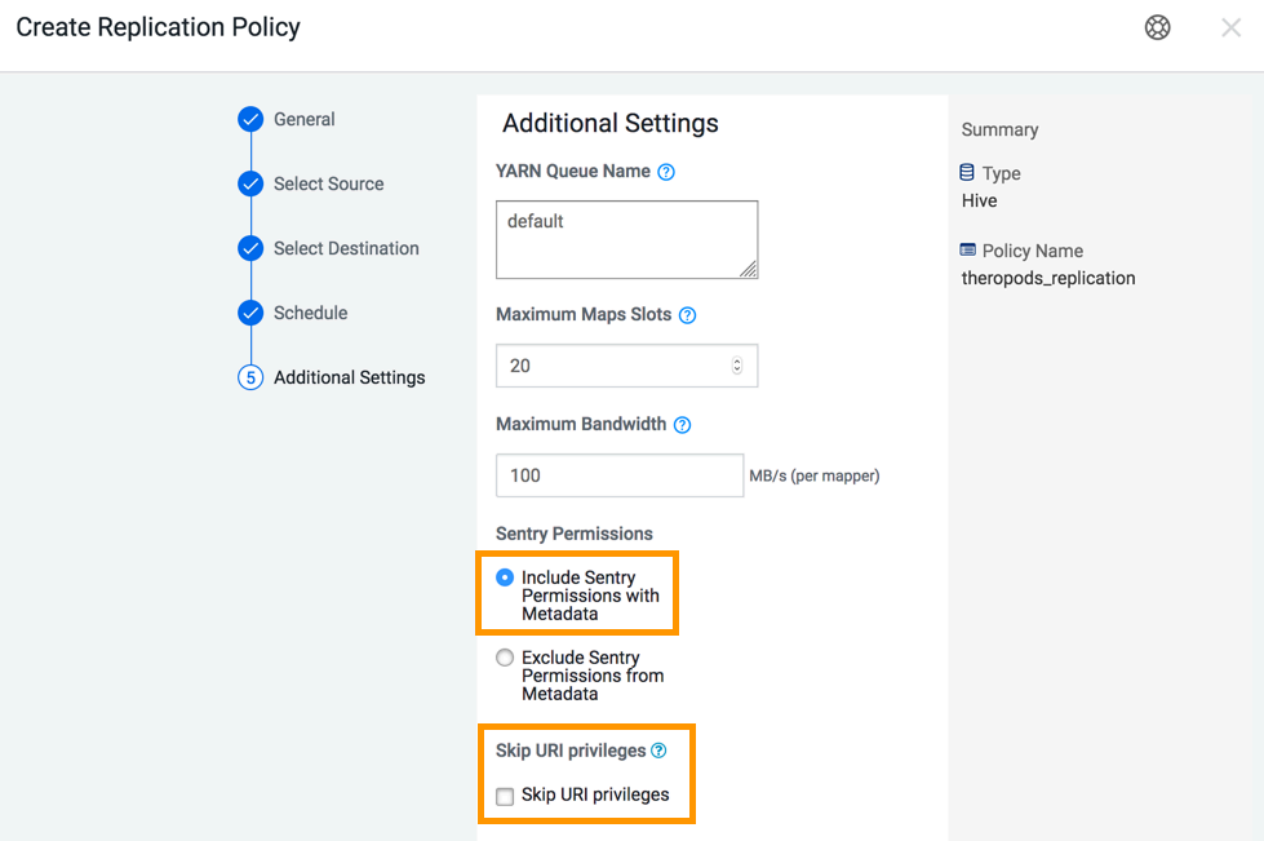

The following steps are completed during the migration of Sentry policies into Ranger:

- The Export operation runs in the source cluster. During this operation, the Sentry permissions are fetched and exported to a JSON file. This file might be in a local file system or HDFS or S3, based on the configuration.
- The Translate and Ingest operations take place on the target cluster. In the translate operation, Sentry permissions are translated into a format that can be read by Ranger. The permissions are then imported into Ranger. When the permissions are imported, they are tagged with the source cluster name and the time that the ingest took place. After the import, the file containing the permissions is deleted.

> **Note:** During Hive replication from an on-premises CDH cluster to a cloud cluster, the Replication Manager migrates Sentry authorization policies into Ranger as part of the replication policy. However, no import operation is initiated if the end service in the cloud cluster (AWS) is Sentry.

A Ranger policy is created for each resource, such as a database, table, or column. The policy name is derived from the resource name. For example, if the resource is Database:dinosaurs, table= theropods, then the derived policy name is database=dinosarus->table=theropods.

The priority for migrated policies is set to normal in Ranger. The normal priority allows you to create another policy for the same resource that overrides the policy that is imported from Sentry.

Sentry to Ranger permissions

There are no one-to-one mapping between Sentry privileges and Ranger service policies, therefore the Sentry privileges are translated to their equivalents within the Ranger service policies.

The following list illustrates how the Sentry privileges appear in Ranger after the migration:

- Sentry permissions that are granted to roles are granted to groups in Ranger.
- Sentry permissions that are granted to a parent object are granted to the child object as well. The migration process preserves the permissions that are applied to child objects. For example, a permission that is applied at the database level also applies to the tables within that database.
- Sentry OWNER privileges are translated to the Ranger OWNDER privilege.

- Sentry OWNER WITH GRANT OPTION privileges are translated to Ranger OWNER with Delegated Admin checked.
- Sentry does not differentiate between tables and views. When view permissions are migrated, they are treated as table names.
- Sentry privileges on URIs uses the object store location as the base location.
- If your cluster contains the Kafka service and the Kafka sentry policy had "action": "ALL" permission, the migrated Ranger policy for "cluster" resource will be missing the "alter" permission. This is only applicable for "cluster" resource. You will need to add the policy manually after the upgrade. This missing permission will not have any functional impact. Adding the "alter" permission post upgrade is needed only for completeness because the 'configure' permission will allow alter operations.

The table below shows how actions in Sentry are applied to the corresponding action in Ranger:

Table 1: Sentry Actions to Ranger Actions

Sentry Action	Ranger Action
SELECT	SELECT
INSERT	UPDATE
CREATE	CREATE
REFRESH	REFRESH
ALL	ALL
SELECT with Grant	INSERT
INSERT with Grant	INSERT
CREATE with Grant	CREATE
ALL with Grant	ALL with Delegated Admin Checked

HBase replication policy

You can replicate HBase and Phoenix tables using HBase replication policies in Cloudera Replication Manager. An HBase replication policy replicates the data at table-level granularity. After you create an HBase replication policy, you can delete one or more tables from the policy. The replication policy replicates the data in the specified tables and continues to replicate the generated data (that is, future changes in data) unless you suspend the policy or delete the tables.

The replication policy replicates the data in the specified tables and continues to replicate the generated data (that is, future changes in data) unless you suspend the policy or delete the tables. You can replicate only existing HBase data, generated HBase data, or both depending on your requirements. You also can choose to replicate all the HBase tables or only the required tables in a database.

Before you create an HBase replication policy, you must:

- verify whether your clusters are supported by Replication Manager.
- understand how first-time setup configuration works.
- understand how cluster pairing works.
- understand the available methods to replicate HBase data.

You can also replicate HBase data simultaneously between multiple clusters.

Supported clusters for HBase replication policies

Before you create HBase replication policies, you must verify whether your clusters are supported by Replication Manager.

The following image shows a high-level view of the support matrix for HBase replication policies, you must consult the Support matrix for Replication Manager on Cloudera Public Cloud for the complete list of supporting clusters and scenarios:

Figure 1: High-level replication scenarios supported by HBase replication policies

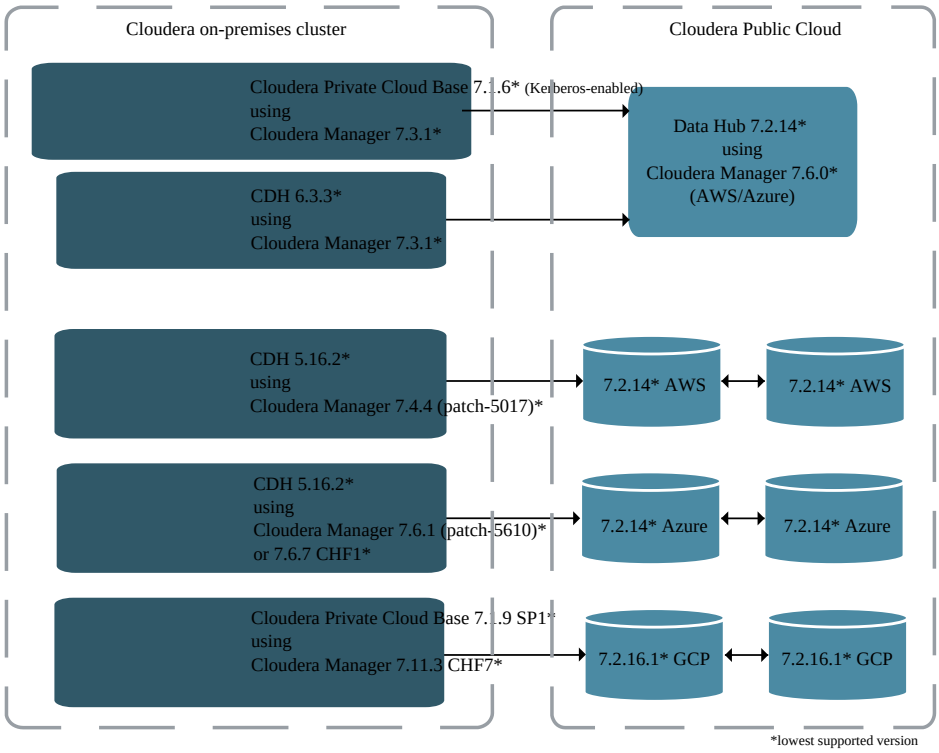

How HBase replication policies work

After you create the first HBase replication policy between a source cluster and target cluster, the Replication Manager service starts several background processes. After the processes complete, the service initiates the HBase data replication. One of the main background process is the first-time setup configuration.

- Understanding first-time setup configuration process
- What is a cluster pair

Steps in first-time setup configuration process

When you create the first HBase replication policy to replicate HBase data from a source cluster to a target cluster, the Replication Manager completes the first-time setup configuration steps and then replicates the data. The first-time setup configuration between a cluster pair is a one-time process, therefore subsequent HBase replication policies for the same cluster pair (ClusterA and ClusterB) do not require a first-time setup.

The first-time setup configuration completes the following steps:

1. Creates peers between the source (ClusterA) and target (ClusterB), that is, creates a cluster pair between ClusterA and ClusterB.
2. Initiates the required configuration changes. One of the steps ensures that both the clusters use the same credentials.jceks file.
3. Restarts the HBase service on both the clusters.

Important: If you are replicating HBase data from on-premises cluster to Cloudera Operational Database or Cloudera Data Hub, you must manually restart HBase service on the source cluster.

When HBase replication policies are created, modified, or deleted simultaneously, Replication Manager processes each operation independent of each other.

Note: When the first-time setup between two clusters is in progress, you can create HBase replication policies between them. However, you cannot use one of these clusters with another cluster to create an HBase replication policy (in multi-cluster replication scenario) until the first-time setup is complete.

What is a cluster pair

The first-time setup configuration creates peers between the source (ClusterA) and target (ClusterB), that is, creates a cluster pair between ClusterA and ClusterB. If a cluster pair (ClusterA and ClusterB) has one or more active/suspended HBase replication policies between them, you cannot pair either of the clusters with another cluster.

You can use ClusterA or ClusterB with another cluster in an HBase replication policy only if the following conditions are true:

- ClusterA and ClusterB do not have existing HBase replication policies.
- All existing active/suspended HBase replication policies are deleted.

A warning appears on the **Select Destination** page during HBase replication policy creation if you choose ClusterA or ClusterB as source or target (when no HBase replication policies exist between them) with another cluster or if the other cluster in the cluster pair is unreachable. To continue HBase replication policy creation, you must acknowledge the forced re-pairing of the clusters.

Methods to replicate HBase data

You can replicate only the existing HBase data, generated HBase data, or both depending on your requirements. You also can choose to replicate all the HBase tables or only the required tables in a database.

When you create an HBase replication policy, you can choose one or more of the following methods to replicate HBase data depending on your requirements:

- Replicate only the generated data from chosen tables.
- Replicate existing data and generated data from chosen tables.
- Replicate existing tables and future tables in the database.
- Replicate existing data and generated data from chosen tables and future tables

Replicate only the generated data from chosen tables

In this method, you choose one or more tables during the replication policy creation process. Replication Manager replicates only the data that is generated after policy creation.

Replicate existing data and generated data from chosen tables

In this method, you choose one or more tables, and also choose the Select Source Perform Initial Snapshot option during the HBase replication policy creation process. Replication Manager replicates the existing data and the data that is generated after policy creation.

For example, you have two tables named 'Orders' and 'Customers' in the source cluster and you want to copy the data from these tables from March 1, 2021 onwards. To accomplish this task, you create an HBase replication policy without choosing the Perform Initial Snapshot option in the **Create Replication Policy** wizard on March 1, 2021. The data that you create, update, or delete in the source cluster after you created the HBase replication policy is automatically replicated to the target cluster.

Replicate existing tables and future tables in the database

In this method, you choose the Select Source Replicate Database option during the HBase replication policy creation process. Replication Manager replicates the generated data from the existing tables, and it replicates data from the future tables that are created after the HBase replication policy creation process is complete.

To replicate data from the future tables successfully, you must create similar empty tables on the target cluster. You can perform this action when you create or add a table to the database on the source cluster.

You can choose the Replicate Database option only if the following conditions are true:

- Target Cloudera Manager version is 7.11.0 or higher.
- Source cluster's CDH version is 6.x or higher.

  CDH 5.16.2 and higher versions also support the Replicate Database option after you upgrade the source cluster Cloudera Manager.
- No existing HBase replication policies exist between the source and target clusters.

  Tip: If you want to replicate the new tables that are created after the replication policy creation is complete, you must configure the replication scope to "1" for those tables on the source cluster.

  To configure the replication scope for a table on the master cluster, run the alter *[***TABLE NAME***]*, {NAME => *[***COLUMN FAMILY***]*, REPLICATION_SCOPE => 1} command for each column family that must be replicated. *REPLICATION_SCOPE* is a column-family level attribute, where the value '0' means replication is disabled, and '1' means replication is enabled.

After you select the Select Source Replicate Database option in the HBase replication policy wizard, you can choose one of the following options to determine the tables in the database to replicate:

- Replicate all user tables - Replicates all the HBase tables in the database after the replication scope of the tables are set to 1.
- Replicate only tables where replication is already enabled - Replicates only those tables for which the replication scope is already set to 1.

  This option is supported only if the target cluster Cloudera version is 7.2.17.300 using Cloudera Manager 7.11.0-h3 or higher versions or Cloudera version 7.2.16.500 using Cloudera Manager 7.9.0-h7 or higher versions, or Cloudera version 7.12.0.0.

Replicate existing data and generated data from chosen tables and future tables

In this method, you choose the Perform Initial Snapshot and Replicate Database options on the **Select Source** page during the HBase replication policy creation process. You can also choose to replicate all the tables in the database or only those tables for which the replication scope is already set to 1. Replication Manager replicates the existing and generated data from the existing tables in addition to the data in future tables.

Replicate HBase data simultaneously between multiple clusters

Starting from Cloudera 7.2.16.500, 7.2.17.200, and 7.2.18, you can create multiple HBase replication policies between multiple clusters to replicate HBase data. You must consider the limitations before you create a multi-cluster replication scenario. You can use the multi-cluster replication scenario for various use cases.

- How multi-cluster HBase replication works
- Limitations
- Sample use cases

How multi-cluster HBase replication works

The first-time setup configuration consists of several steps of which one step is to ensure that the source and target cluster use the same credentials.jceks file. Therefore, if multiple supported clusters share the same credentials.jceks file, you can replicate HBase data between them seamlessly using HBase replication policies.

The following image shows a sample multi-cluster HBase data replication scenario and a few possible directions of replication:

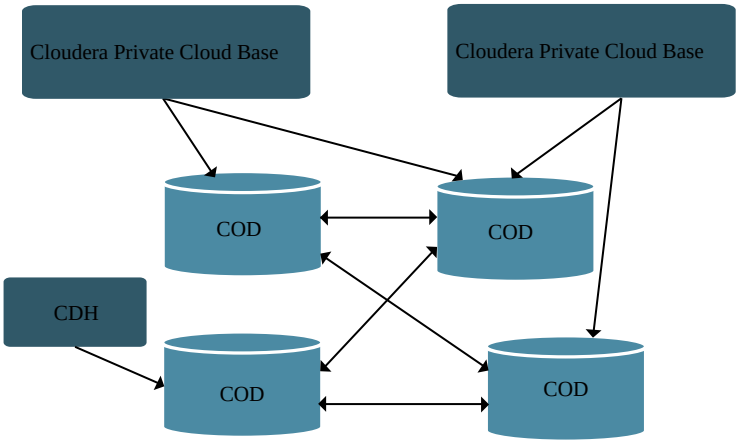

It is recommended that you do not replace the credentials.jceks file manually to create a multi-cluster HBase replication scenario. This is because when you create the first HBase replication policy between a pair of clusters, Replication Manager triggers the first-time setup process during which the credentials.jceks file in both the clusters get synchronized as required for HBase data replication.

Limitations

Consider the following limitations before you replicate the HBase data between multiple clusters using HBase replication policies:

- An HBase replication policy in a multi-cluster HBase replication setup fails when you use clusters that are part of another independent replication setup. This is because the clusters use a different credentials.jceks file. To use these clusters, you must break the cluster pairing and then create the required HBase replication policies.

  Tip: To break the cluster pairing, use the `POST /dmx/api/clusters/<target cluster crn>/hbase/resetFirstTimeSetup` API with the `{"sourceCluster": "<source cluster data center>$<source cluster name>"}` payload.

  Monitor the growing multi-cluster replication network so that it does not get disconnected. This ensures that the credentials.jceks file is the same on all clusters, the replication setup is always consistent, and no existing replication scenarios have to be reset.
- The Replication Manager UI does not allow the HBase replication policy creation to proceed if you choose a cluster (as source or target) that is in another first-time setup process. In this instance, you can wait for a few minutes to allow the first-time setup to complete and then create the HBase replication policy.

  When you create the first HBase replication policy between two clusters, the first-time setup configuration is initiated. After the configuration completes, the HBase data replication is initiated.

  Important: In a multi-cluster replication scenario, when you use a source on-premises cluster, you must manually restart the HBase service on the cluster after the first-time setup configuration completes. However, the next time you choose the same source cluster with another target cluster, the manual restart is not required. The same is true for the automatic HBase service restart in Data Hubs where the restart is performed only when the first replication is created with a particular cluster.
- The following conditions must be met to use the IDBroker credentials to create multiple HBase replication policies between multiple clusters when the target COD clusters are in separate AWS accounts or when a single AWS Role does not have access to all the required S3 buckets for all HBase target clusters:
  - You are using Cloudera Public Cloud 7.2.18.200 or higher versions.
  - You choose the Perform Initial Snapshot option, and then specify the custom username in the Export snapshot user field in the **Select Source** page during the HBase replication policy creation process.

Use cases

Some use cases where you can use the multi-cluster HBase replication scenarios are illustrated below:

- Multiple source clusters and a single target cluster. You might have a disaster-recovery use case where you want to use a single COD to back up all the HBase data. The following image illustrates this scenario:

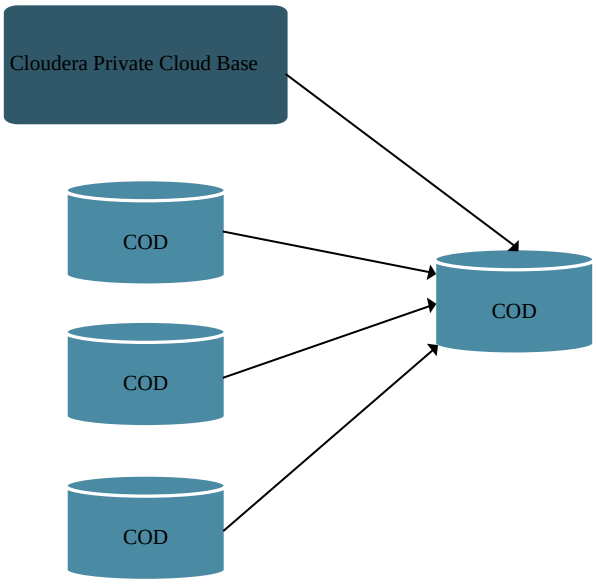

- Single source cluster and multiple target clusters. You might have a use case where all the HBase data is located in a cluster and you want to replicate only specific HBase tables to different environments to fulfill specific requirements. For example, QE environments and/or experimentation use case. The following image illustrates this scenario:

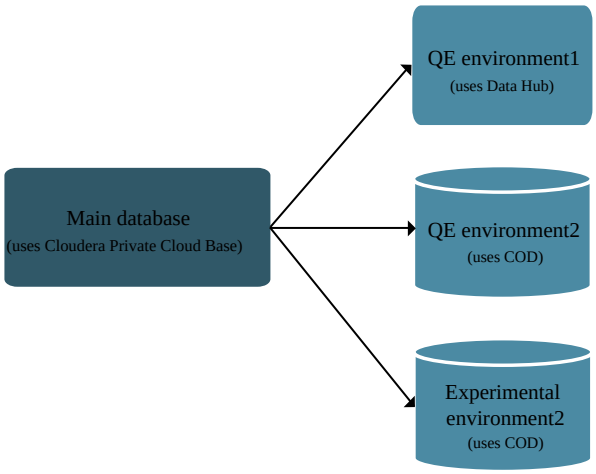

Using HDFS replication policies

You can use the HDFS replication policies in Cloudera Replication Manager to replicate HDFS data. The HDFS replication policies can replicate HDFS data and metadata from classic clusters (CDH, Cloudera Private Cloud Base, and HDP) to Cloudera Public Cloud storage buckets such as S3 and ABFS, and from cloud storage to classic clusters (CDH or Cloudera Private Cloud Base clusters). To use an on-premises cluster (CDH or Cloudera Private Cloud

Base) in the replication policy, you must register it as a classic cluster in the Cloudera Management Console. To use the cloud storage for data replication, you must register the cloud credentials in Replication Manager so that the Replication Manager service can access the cloud storage. You must also verify cluster access and configure minimum ports for replication before you create HDFS replication policies.

Important: Before you create replication policies, see *Support matrix for Replication Manager on Cloudera Public Cloud* to verify whether your clusters are supported by Replication Manager.

You can also use CDP CLI commands to create HDFS replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see *CDP CLI for Replication Manager*.

Preparing to create an HDFS replication policy

Before you create the HDFS replication policies in Cloudera Replication Manager to replicate HDFS data, register the on-premises cluster (CDH or Cloudera Private Cloud Base) as a classic cluster in Management Console, register cloud account credentials in the Replication Manager service, verify cluster access, and configure minimum ports for replication.

Procedure

- Do the source cluster and target cluster meet the requirements to create an HDFS replication policy?

  The following image shows a high-level view of the support matrix for HDFS replication policies, you must consult the Support matrix for Replication Manager on Cloudera Public Cloud for the complete list of supporting clusters and scenarios:

  Figure 2: High-level replication scenarios supported by HDFS replication policies

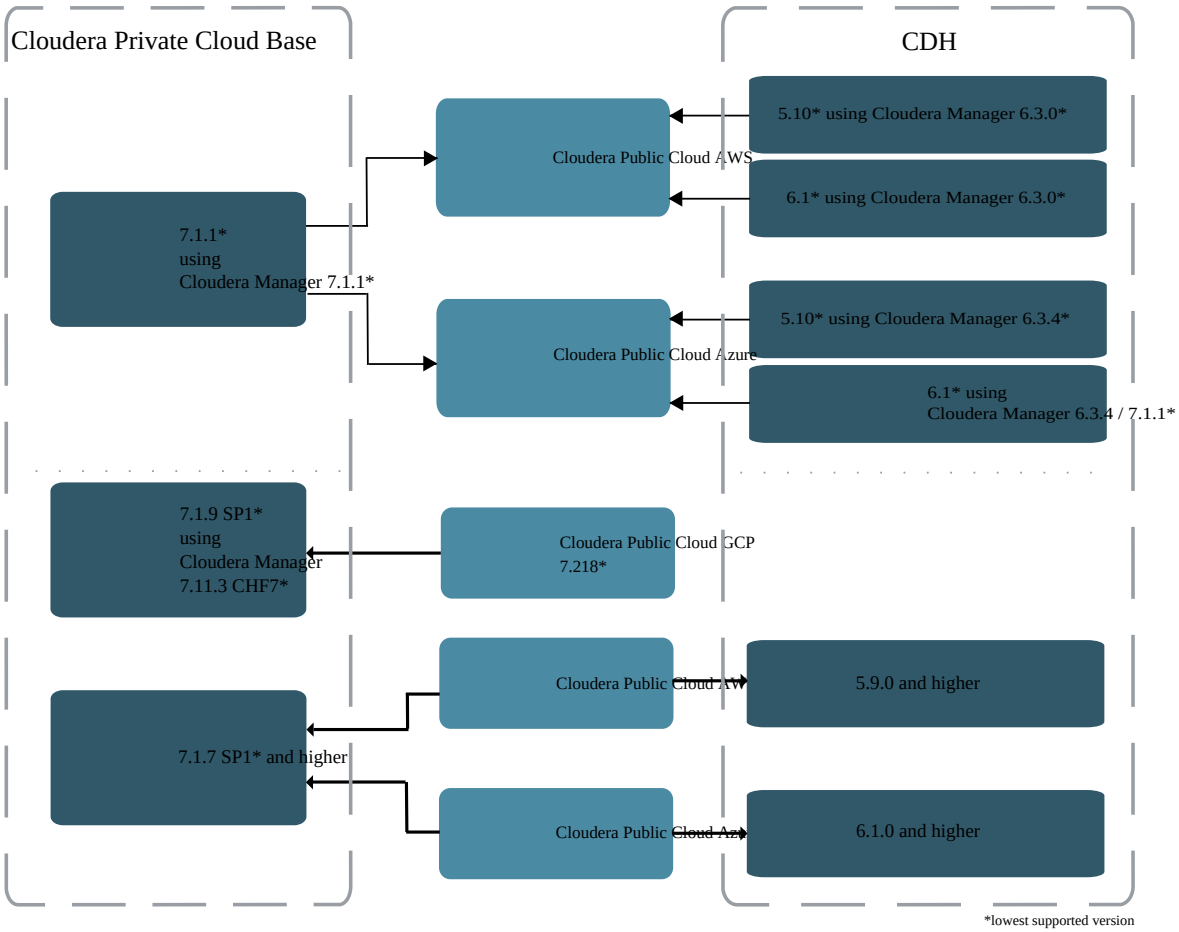

- Is the required on-premises cluster (CDH cluster or Cloudera Private Cloud Base cluster) registered as a classic cluster on the Management Console?

  CDH clusters and Cloudera Private Cloud Base clusters are managed by Cloudera Manager. To enable these on-premises clusters for Replication Manager, you must register them as Classic Clusters on the Management Console. After registration, you can use them for data migration purposes.

  Important: When you register a Cloudera Private Cloud Base cluster as a classic cluster, ensure that you use only the Cloudera Manager IP Address and Cloudera Manager Port options and not the Register KNOX endpoint (Optional) option.

  For information about registering an on-premises cluster as a classic cluster, see Adding a CDH cluster and Adding a Cloudera Private Cloud Base cluster.

- Is an external account available in the Cloudera Manager instance that has access to the bucket or container that you are using in the HDFS replication policy?

  For more information, see Role-based credential on AWS, App-based credential on Azure, and Cloudera Manager documentation.

- Do you have the required cluster access to create replication policies?

  Power users, the user who onboarded the source and target clusters, and users with ClassicClusterAdmin or ClassicClusterUser resource roles can create replication policies on clusters for which they have access. For more information, see Understanding account roles and resource roles.

  Tip: Ensure that you have *Replication Administrator* or Full Administrator role on the source on-premises cluster.

- Do you have the required cluster access to view the replication policies?

  Existing HDFS replication policies are visible to users who have access to the source cluster in the replication policy. A warning appears if you do not have access to the source cluster.

  If you can view the policies, you can perform other actions on the policy including policy update and policy delete operations.

  Note: A Replication Manager user can browse, within Replication Manager, the folder structure of any cluster enabled for Replication Manager. Therefore, permitted users can view folders, files, and databases in Replication Manager that they might not have access to in HDFS. The users cannot view the content of files on the source or destination clusters, nor can they modify or delete folders or files that are viewable from Replication Manager.

- Is the required cloud credential that you want to use in the replication policy registered with the Replication Manager service?

  For more information, see Working with cloud credentials.

- Are the following ports open and available for Replication Manager?

Table 2: Minimum ports required for HDFS replication policies

Connectivity required	Default Port	Type	Description
Data transfer from classic cluster hosts to cloud storage	80 or 443 (TLS)	Outbound	Outgoing port. All classic cluster nodes must be able to access S3/ ADLS Gen2 endpoint.
Classic cluster	6000-6049 for CCMv1 443 for CCMv2	Outbound	Connecting source classic cluster to the Cloudera Management Console through Cluster Connectivity Manager (CCM). For more information, see Outbound network access for CCM, and CCM overview.

Consider the following best practices while using Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS):

- Ensure that the on-premises cluster (port 443) can access the *https://login.microsoftonline.com* endpoint. This is because the Hadoop client in the on-premises cluster (CDH/) connects to the endpoint to acquire the access

tokens before it connects to Azure ADLS storage. For more information, see the *General Azure guidelines* row in the Azure-specific endpoints table.

- Ensure that the steps mentioned in the *General Azure guidelines* and *Azure Data Lake Storage Gen 2* rows in the Azure-specific endpoints table are complete so that the endpoint connects to the target path successfully.

The following system architecture diagram shows the interaction between components during HDFS replication using HDFS replication policies:

Figure 3: System architecture diagram for HDFS replication in Cloudera Replication Manager

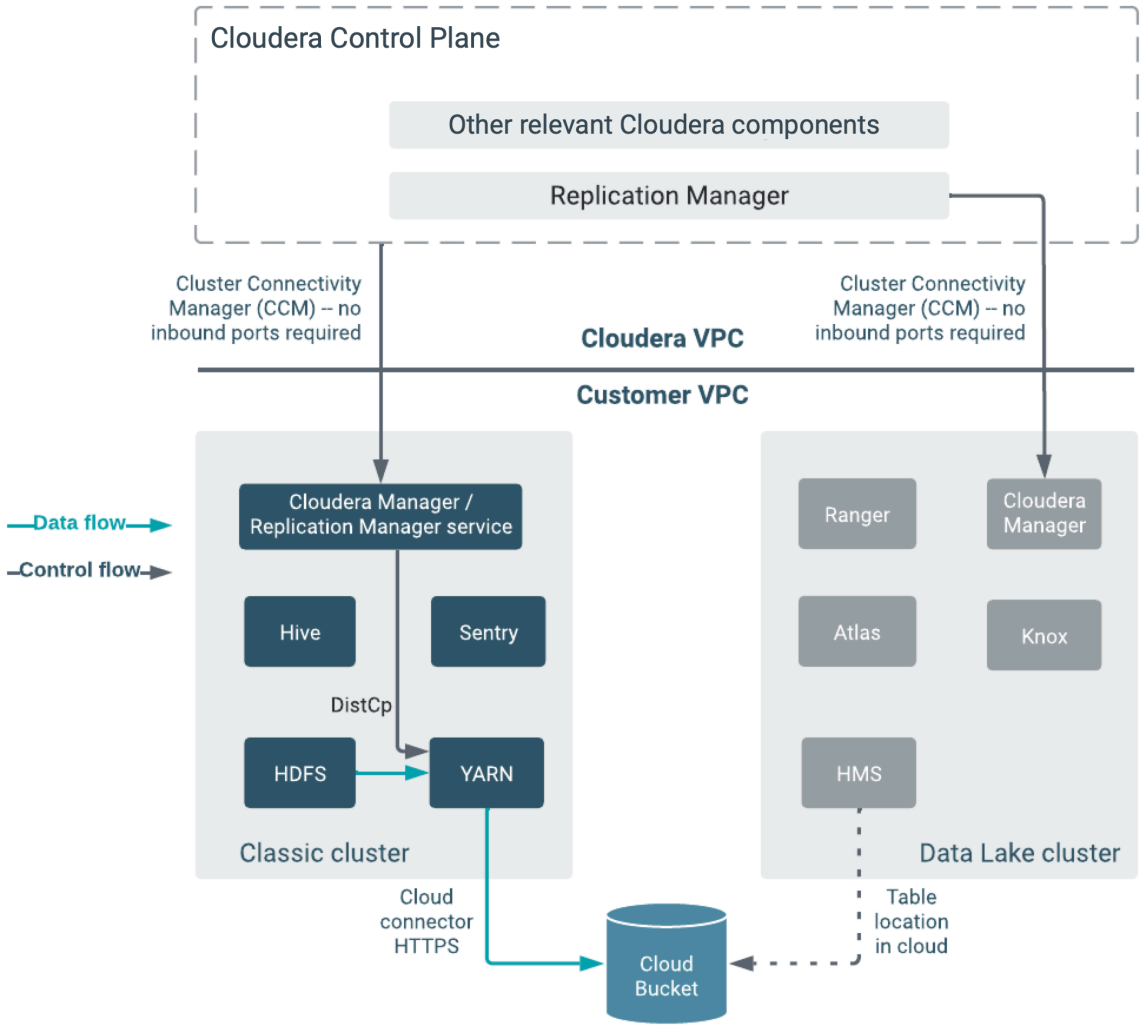

What to do next

After the clusters and cloud storage requirements are met, you can create an HDFS replication policy.

Creating HDFS replication policy

After you register the classic clusters in Management Console and register the cloud credentials in Cloudera Replication Manager, you can create an HDFS replication policy to replicate HDFS data. An HDFS replication policy can replicate HDFS data from a classic cluster (CDH, Cloudera Private Cloud Base, and HDP clusters) to cloud storage, and from cloud storage to classic clusters (CDH or Cloudera Private Cloud Base clusters). You can replicate data on-premises to cloud storage account with a single cluster.

About this task

Alternatively, you can also use CDP CLI commands to create HDFS replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see CDP CLI for Replication Manager.

Procedure

1. On the Cloudera Management Console Replication Manager Replication Policies page, click Add Policy.

   The **Create Replication Policy** wizard appears.
2. On the General page, choose or enter the following information:

Option	Description
HDFS	Creates an HDFS replication policy.
Policy Name	Enter a unique name for the replication policy.
Description	Optional. Enter a brief description about the replication policy.

   The following image shows a sample General page in the **Create Replication Policy** wizard:

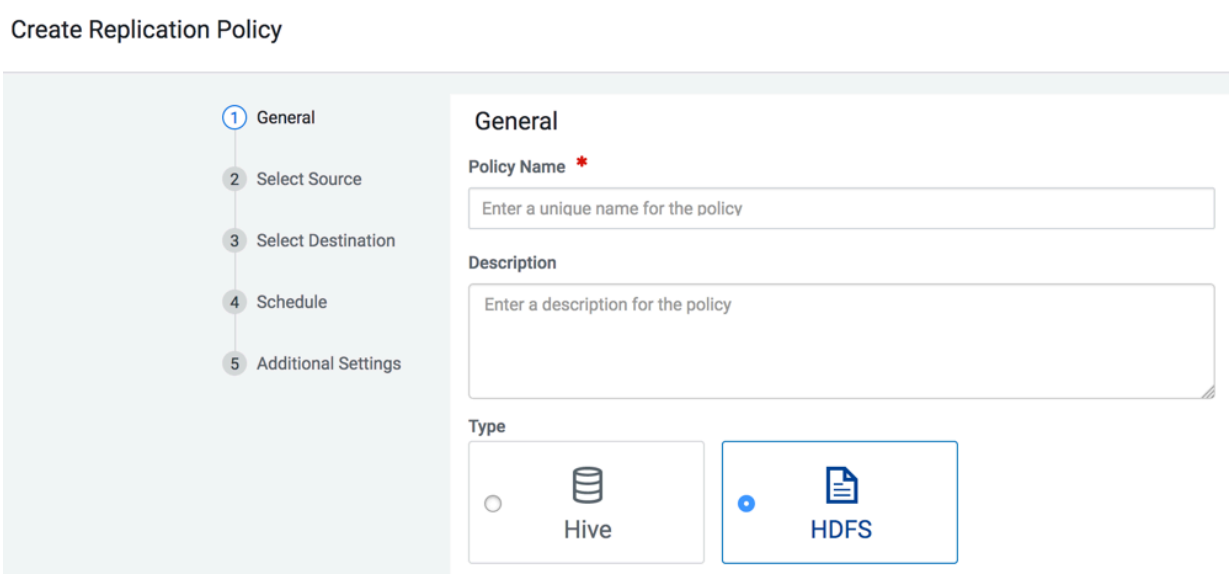

3. Click Next.
4. On the Select Source page, the options change depending on whether you choose a classic cluster or cloud storage as the source for data to be replicated.

   a) If you are replicating from a classic cluster , choose or enter the following information:

Option	Description
Type	Choose CLUSTER to select an on-premises cluster as the source cluster.
Source Cluster	Select a classic cluster.
Source path	Choose one of the following methods to determine the directory where source data resides on the source cluster: • Enter the complete directory path. • Click File Browser to view and navigate the existing directory list on the selected cluster. Select the required directory that you want to replicate.

Option	Description
Run As Username (on source)	Optional. The replication policy uses the Default username to replicate HDFS data. If you are using a kerberized cluster, enter the required username. The replication policy uses this username to replicate the data in the kerberized cluster.

b) If the source of your replication is cloud storage, choose or enter the following information:

Option	Description
Type	Choose S3 or ABFS to select your cloud provider.
Cloud Credential on Source	Choose the required cloud credential. The cloud credentials that you register for Replication Manager on the Cloud Credentials page appear in this field.
Path	Based on the cloud provider, enter the path to the source data in the required format. • For an S3 bucket, provide the path of the directory in the *[***BUCKET NAME***]/[***PATH***]* format. • For an ABFS container, provide the path of the directory in the *[***FILESYSTEM***]@[***STORAGE ACCOUNT***]/[***LOCATION***]* format.

5. Click Next.
6. On the Select Destination page, the options change depending on whether you choose a classic cluster or cloud storage as the destination for the replicated data:

a) If you are replicating from a classic cluster to cloud storage, choose or enter the following information:

Option	Description
Type	Choose S3 or ABFS to select your cloud provider.
Cloud Credential on Source	Choose the required cloud credential. The cloud credentials that you register for the selected source cluster on the **Cloud Credentials** page appear in this field. You can also add cloud credentials using the Add Cloud Credential link.
Path	Based on the cloud provider, enter the target path, where data is replicated to, in the required format. • For an S3 bucket, provide the path of the directory in the *[***BUCKET NAME***]/[***PATH***]* format. • For an ABFS container, provide the path of the directory in the *[***FILESYSTEM***]@[***STORAGE ACCOUNT***]/[***LOCATION***]* format.

b) If you are replicating from cloud storage to a classic cluster, choose or enter the following information:

Option	Description
Type	Choose Classic Cluster.
Destination Cluster	Choose a classic cluster as the target cluster.
Destination Path	Enter the directory on the target cluster to which the replication policy replicates HDFS data.
Run As Username	Optional. The replication policy uses the Default username to replicate HDFS data. Enter another username if you want the replication policy to use it to replicate data.

7. Click Validate Policy.

   Replication Manager verifies whether the details provided are correct.
8. Click Next.

9. On the **Schedule** page, choose or enter the following information:

Option	Description
Run Now	Starts to replicate the existing HDFS data after the replication policy creation is complete. Choose the frequency to replicate data periodically.
Schedule Run	Runs the replication policy to replicate data at a later time. Choose the date and time for the first run, and then choose the frequency to replicate data periodically. **Tip:** On the **Replication Policies** page, click to change the timezone.
Frequency	Choose one of the following options: • Does Not Repeat • Custom - In the Custom Recurrence dialog box, choose the time, date, and the frequency to run the policy. Replication Manager ensures that the same number of seconds elapse between the runs. For example, if you choose the Start Time as January 19, 2022 11.06 AM and Interval as 1 day, Replication Manager runs the replication policy for the first time at the specified time in the timezone the replication policy was created in, and then runs it exactly after 1 day that is, after 24 hours or 86400 seconds. **Note:** Ensure that the frequency in a schedule enables a job to finish before the next job starts. Also, ensure that the jobs based on the same policy do not overlap. If a job is not completed before another job starts, the second job does not run and the job status appears as Skipped. If a job is consistently skipped, you might need to modify the frequency of the job.

10. Click Next.
11. On the Additional Settings page, enter or choose the values as necessary:

Option	Description
YARN Queue Name	Enter the name of the YARN queue for the cluster to which the replication job is submitted if you are using Capacity Scheduler queues to limit resource consumption. The default value for this field is default.
Maximum Maps Slots	Set the maximum number of map tasks (simultaneous copies) per replication job. The default value is 20.
Maximum Bandwidth	Adjust this setting so that each map task is throttled to consume only the specified bandwidth. Each map task ((simultaneous copy) is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value. You can adjust this setting so that each map task is throttled to consume only the specified bandwidth so that the net bandwidth used tends towards the specified value. The default value for the bandwidth is 100MB per second for each mapper.
Path Exclusion	Enter one or more regular expressions separated by comma. Replication Manager does not copy the subdirectories or files from the source that matches one of the specified regular expressions to the target cluster.

Option	Description
Replication Strategy	Choose one of the following replication strategies to determine whether the file replication tasks should be distributed among the mappers statically or dynamically. • Static distributes file replication tasks among the mappers up front to achieve an uniform distribution based on the file sizes. • Dynamic distributes the file replication tasks in small sets to the mappers, and as each mapper completes its tasks, it dynamically acquires and processes the next unallocated set of tasks. The default replication strategy is Dynamic.
MapReduce Service	Choose the MapReduce or YARN service to use.
Log Path	Enter an alternate path for the logs, if required.
Error Handling	Select the following options as necessary: • Skip Checksum Checks - Determines whether to skip checksum checks on the copied files. If selected, checksums are not validated. Checksums are checked by default. **Note:** You must skip checksum checks to prevent replication failure due to non-matching checksums in the following cases: • Replications from an encrypted zone on the source cluster to an encrypted zone on a destination cluster. • Replications from an encryption zone on the source cluster to an unencrypted zone on the destination cluster. • Replications from an unencrypted zone on the source cluster to an encrypted zone on the destination cluster. Checksums are used for two purposes: • To skip replication of files that have already been copied. If Skip Checksum Checks is selected, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination. • To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data. • Skip Listing Checksum Checks - Whether to skip checksum check when comparing two files to determine whether they are same or not. If skipped, the file size and last modified time are used to determine if files are the same or not. Skipping the check improves performance during the mapper phase. Note that if you select the Skip Checksum Checks option, this check is also skipped. • Abort on Error - Whether to abort the job on an error. If selected, files copied up to that point remain on the destination, but no additional files are copied. Abort on Error is not selected by default. • Abort on Snapshot Diff Failures - If a snapshot diff fails during replication, the replication policy uses a complete copy to replicate data. If you select this option, the policy aborts the replication when it encounters an error instead.

Option	Description
Preserve	Choose the required options to preserve the block size, replication count, permissions (including ACLs), and extended attributes (XAttrs) as they exist on the source file system, or to use the settings as configured on the destination file system. By default source system settings are preserved. When Permission is selected, and both the source and destination clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. When Extended attributes is selected, and both the source and destination clusters support extended attributes, replication preserves them. (This option only displays when both source and destination clusters support extended attributes.) If you select one or more of the Preserve options and you are replicating to S3 or ADLS, the values all of these items are saved in metadata files on S3 or ADLS. When you replicate from S3 or ADLS to HDFS, you can select which of these options you want to preserve. **Note:** To preserve permissions to HDFS, you must be running as a superuser on the destination cluster. Use the Run As Username option to set the username.
Delete Policy	Choose the required options to determine whether the files that were deleted on the source should also be deleted from the destination directory. This policy also determines the handling of files in the destination location that are unrelated to the source. Options include: • Keep Deleted Files - Retains the destination files even when they no longer exist at the source. This is the default option. • Delete to Trash - If the HDFS trash is enabled, files are moved to the trash folder. This is not supported when replicating to S3 or ADLS. • Delete Permanently - Uses the least amount of space; use with caution.
Alerts	Choose to generate alerts for various state changes in the replication workflow. You can choose to generate an alert On Failure, On Start, On Success, or On Abort of the replication job. You can configure alerts to be delivered by email or sent as SNMP traps. If alerts are enabled for events, you can search for and view the alerts on the **Events** tab, even if you do not have email notification configured. For example, if you choose Command Result that contains the Failed filter on the Diagnostics Events page, the alerts related to the On Failure alert for all the replication policies for which you have set the alert appear. For more information, see Managing Alerts and Configuring Alert Delivery.

12. Click Create.

What to do next

You can track the replication policy job status on the **Replication Policies** page.

Manage and monitor HDFS replication policies

After you create an HDFS replication policy in Cloudera Replication Manager, you can perform and monitor various tasks related to the replication policy. You can view the job progress and replication logs. You can edit the advanced options to optimize a job run. You can suspend a job and also activate a suspended job. You can edit the replication policy as necessary.

About this task

On the **Replication Policies** page, you can perform the following actions and tasks on a replication policy and its jobs:

Procedure

- When you click Actions for an HDFS replication policy, the following actions appear:

Action	Description
Edit*	Change the replication policy options as required for non-expired policies that are in active or suspended state. Based on the schedule you choose, the replication policy replicates data. You can edit the replication policies to better align with changing requirements. For example, you might want to change the frequency of a policy depending on the data size and importance of the data being replicated. **Note:** A replication policy is associated with a cluster or a cluster pair, therefore you cannot change the clusters in the policy. Optionally, expand a replication policy on the **Replication Policies** page to edit the replication policy options which include frequency (start time cannot be modified if the policy has already started), queue name, maximum bandwidth, and maximum map slots. **Tip:** To optimize the replication policy performance, you can configure the queue name, maximum bandwidth, and maximum map slots as necessary.
Delete	Deletes the replication policy permanently.
Suspend	Suspends a running replication policy. Activate the replication policy, if required.
View Log	Download, copy, or open the log. The log shows a brief output of the stdout and stderr logs of a single step of the latest replication policy job run. You can also view the current job status in the Replication Manager Overview Issues & Updates Job Status column. If the job failed, click **Failed** to view the log details about the job.
Collect diagnostic bundle	Generates a diagnostic bundle for the replication policy. You can download the bundle as a ZIP file to your machine. Ensure that you are logged into the Cloudera Manager instances for both the source and target clusters before you download the bundle in Replication Manager.

*

To view and use the replication policies with an empty name in Replication Manager, you must understand the following implementation:

- If the API version is lower than 51, an existing replication policy with an empty name can be used and updated. However, if you edit the replication policy and provide a name for the replication policy in versions higher or equal to 51, you must ensure that the name conforms to the validation rules.
- If the API version is higher or equal to 51, it is mandatory that you provide a unique name to the replication policy to continue using it. This is because API version 51 and higher enforces the validation rules on all the replication policies.

To pass the replication policy name validation, you must ensure that the replication policy name is unique. The name can contain letters, numbers, and the _ / - characters. You must also ensure that it does not contain the characters % . ; \ nor any character that is not ASCII printable, which includes the ASCII characters less than 32 and the ASCII characters that are greater than or equal to 127.

- When you expand the policy details, the **Job History** panel appears.

  You can view the following details on the panel:

  a) Previous jobs, current job, and one future scheduled job if any.

  b) Job details which include:

Job details	Description
Started	Timestamp when the job started.
Ended	Timestamp when the job ended.
Duration	Time taken to complete the job.

Job details	Description
Progress	Current status of a running job.
Expected	Remaining number of files and bytes expected to be copied for a running job.
Copied	Number of files and bytes copied for a running job and completed job.
Failed	Number of files and bytes that failed to be copied for a completed job.
Deleted	Number of files deleted for a completed job.
Skipped	Remaining number of files and bytes skipped from copying for a running job and complete job.

c) Click Actions to:

- Abort the job.
- Re-run an aborted or failed job.
- View Log for the job. You can download, copy, or open it to track the job and to troubleshoot any issues for the job.

- When you click a job on the **Job History** panel, the following tabs appear:

Tab	Description
General	Shows the following job details: • **Started at** timestamp • **Duration** to complete the job • **HDFS Replication Report** to download the job statistics in CSV format • Job status **Message**
Command Details	Shows the steps that Replication Manager ran for the job along with the timestamp.

- You can download the following CSV reports from the General HDFS Replication Report field to track the replication jobs and to troubleshoot issues:

Report	Description
Listing	Lists all the files and directories copied during the replication job.
Status	Shows the complete status report of each file as: • an **Error** occurred and the file was not copied. • a **Deleted** file. • an up-to-date file for which the replication was **Skipped**.
Error Status	Status report of all the copied files with errors. Each file shows the status, path, and message for the copied files with errors.
Skipped Status	Status report of all skipped files. Each file lists the status, path, and message for the databases and tables that were skipped.
Deleted Status	Status report of all deleted files. Each file lists the status, path, and message for the databases and tables that were deleted.
Performance	Summary report about the performance of the running replication job which includes the last performance sample for each mapper that is working on the replication job.
Full Performance	Performance report of the job which includes the samples taken for all mappers during the replication job.

Note: The reports are generated based on the source Cloudera Manager response. If the Cloudera Manager response is interrupted or is not handled as expected, corresponding error messages appear in HTML format in the reports.

Using Hive replication policies

To create a Hive replication policy in Cloudera Replication Manager, you must configure the required Ranger policy in Ranger, register the on-premises cluster (CDH or Cloudera Private Cloud Base) as a classic cluster in Cloudera Management Console, register cloud account credentials in the Replication Manager service, verify cluster access, and configure minimum ports for replication. The replication load happens on the source on-premises cluster. You can replicate data on-premises to the cloud with a single cluster if the Metastore is running on the cloud.

These policies support table-level replication and can replicate Hive external tables from on-premises clusters (CDH and Cloudera Private Cloud Base) to cloud storage such as S3 and ABFS and to Cloudera Data Hub clusterss. They also can:

- replicate data stored in Hive tables, Hive metadata, data in Hive metastore, and Impala metadata (catalog server metadata) associated with Impala tables registered in the Hive metastore, and

  Note: Hive2 managed tables are converted to external tables after replication.

- migrate Sentry permissions to Ranger.

  Note: To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.

Hive metadata replication involves multiple entities. Replication Manager supports replication of external tables in Hive. Hive supports replication of external tables to the target cluster and it retains all the properties of external tables. The data files permission and ownership are preserved so that the relevant external processes can continue to write in it even after failover.

You can also use Cloudera CLI commands to create Hive replication policies. The Cloudera CLI commands for Replication Manager are under the replicationmanager Cloudera CLI option. For more information, see *Cloudera CLI for Replication Manager*.

Important: Before you create Hive replication policies, you must ensure that the required Ranger policy is set in Ranger and see Support matrix for Cloudera Replication Manager on page 92 to verify whether your clusters are supported by Replication Manager.

The Apache Ranger access policy model consists of the following components:

- Specification of the resources that you can apply to a replication policy which includes the HDFS files and directories; Hive databases, tables, and columns; and HBase tables, column-families, and columns.
- Specification of access conditions for specific users and groups.

You must set the Ranger policy for the hdfs user on the target cluster to perform all operations on all databases and tables. The same user role is used to import Hive Metastore. The hdfs user should have access to all Hive datasets, including all operations. Otherwise, Hive import fails during the replication process.

On the target cluster, the hive user must have Ranger admin privileges. The same hive user performs the metadata import operation.

For more information about Hive replication policies to replicate data from CDH clusters to Cloudera Public Cloud, see Migrate Hive data from CDH to Cloudera Public Cloud blog.

Preparing to create a Hive replication policy

Before you create the Hive replication policies in Cloudera Replication Manager, you must prepare the clusters and verify cluster access and cloud credentials.

Procedure

- Do the source cluster and target cluster meet the requirements to create an Hive replication policy?

  The following image shows a high-level view of the support matrix for Hive replication policies, you must consult the Support matrix for Replication Manager on Cloudera Public Cloud for the complete list of supporting clusters and scenarios:

  Figure 4: High-level replication scenarios supported by Hive replication policies

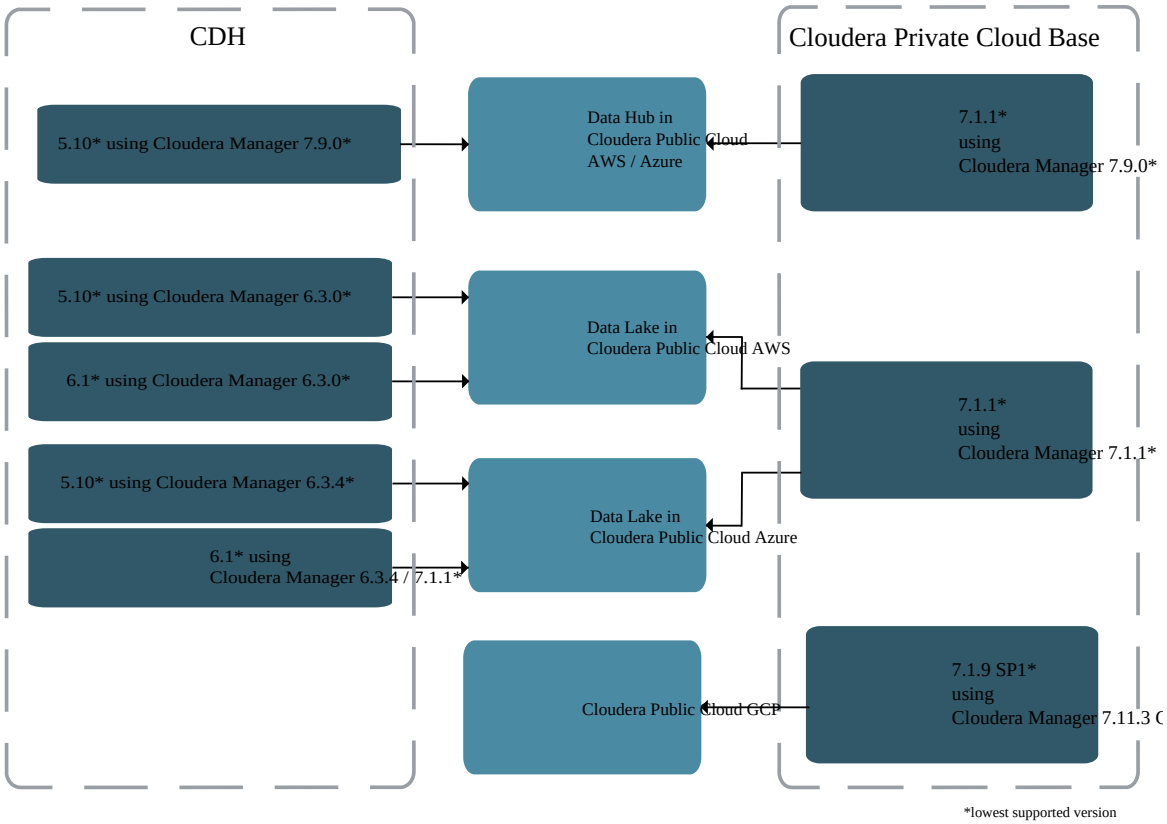

- Is the source CDH cluster or source Cloudera Private Cloud Base cluster registered as a classic cluster on the Management Console?

  CDH clusters and Cloudera Private Cloud Base clusters are managed by Cloudera Manager. To enable these on-premises clusters for Replication Manager, you must register them as Classic Clusters on the Management Console. After registration, you can use them for data migration purposes.

  Important: When you register a Cloudera Private Cloud Base cluster as a classic cluster, ensure that you use only the Cloudera Manager IP Address and Cloudera Manager Port options and not the Register KNOX endpoint (Optional) option.

  For information about registering an on-premises cluster as a classic cluster, see Adding a CDH cluster and Adding a Cloudera Private Cloud Base cluster.
- Does the target Data Hub use Cloudera Manager 7.9.0 or higher? If not, upgrade Cloudera Manager to version 7.9.0 or higher.

- Have you configured the **all-database, table, column** Ranger policy for the hdfs user on the source and target cluster to perform all the operations on all databases and tables?

  The hdfs user role is used to import Hive Metastore and must have access to all Hive datasets, including all operations. Otherwise, Hive import fails during the replication process. On the target cluster, the hive user must have Ranger admin privileges. The same hive user performs the metadata import operation.

  To provide access, navigate to the Ranger Admin UI Service Manager Hadoop_SQL Policies Access section, and provide hdfs user permission to the **all-database, table, column** policy name.

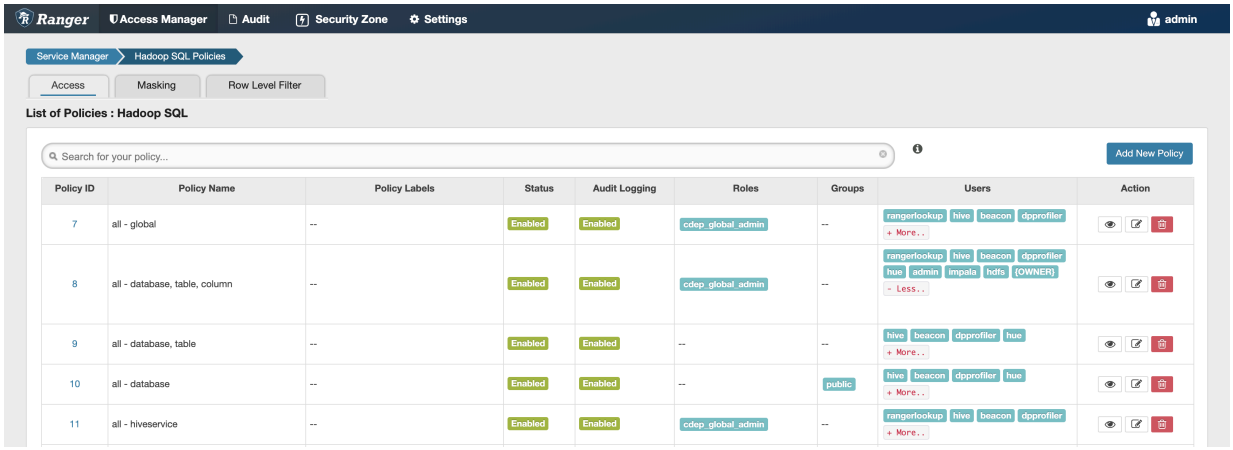

- Is an external account configured on the source CDH cluster's Cloudera Manager which allows the CDH cluster to access Cloudera cloud storage?

  Tip: The external account can be configured on the Cloudera Manager Administration page and the account has the access key / secret key pair that you can use to access Cloudera cloud storage.

- Do you have the required cluster access to create replication policies?

  Power users, the user who onboarded the source and target clusters, and users with ClassicClusterAdmin or ClassicClusterUser resource roles can create replication policies on clusters for which they have access. For more information, see Understanding account roles and resource roles.

  Tip: Ensure that you have *Replication Administrator* or Full Administrator role on the source on-premises cluster.

- Do you have the required cluster access to view the replication policies?

  Existing Hive replication policies are visible to users who have access to the source cluster in the replication policy. A warning appears if you do not have access to the source cluster.

  If you can view the policies, you can perform other actions on the policy including policy update and policy delete operations.

  Note: A Replication Manager user can browse, within Replication Manager, the folder structure of any cluster enabled for Replication Manager. Therefore, permitted users can view folders, files, and databases in Replication Manager that they might not have access to in HDFS. The users cannot view the content of files on the source or destination clusters, nor can they modify or delete folders or files that are viewable from Replication Manager.

- Is the required cloud credential that you want to use in the replication policy registered with the Replication Manager service?

  For more information, see Working with cloud credentials.

- Are the following ports open and available for Replication Manager?

Table 3: Minimum ports required for Hive replication policies

Connectivity required for	Default Port	Type	Description
Data transfer from classic cluster hosts to cloud storage	80 or 443 (TLS)	Outbound	Outgoing port. All classic cluster nodes must be able to access S3/ ADLS Gen2 endpoint.
Cloudera Manager Admin Console HTTP	7180 or 7183 (when TLS enabled)	Inbound	Incoming port. Open on the source cluster to enable the target Cloudera Manager in cloud to communicate to the on-premises Cloudera Manager.
Classic cluster	6000-6049 for CCMv1 443 for CCMv2	Outbound	Connecting the source classic cluster to the Cloudera Management Console through Cluster Connectivity Manager (CCM) For more information, see Outbound network access for CCM,and CCM overview.

Consider the following best practices while using Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS):

- Ensure that the on-premises cluster (port 443) can access the *https://login.microsoftonline.com* endpoint. This is because the Hadoop client in the on-premises cluster (CDH/) connects to the endpoint to acquire the access tokens before it connects to Azure ADLS storage. For more information, see the *General Azure guidelines* row in the Azure-specific endpoints table.

- Ensure that the steps mentioned in the *General Azure guidelines* and *Azure Data Lake Storage Gen 2* rows in the Azure-specific endpoints table are complete so that the endpoint connects to the target path successfully.

The following system architecture diagram shows the interaction between components during Hive replication using Hive replication policies:

Figure 5: System architecture diagram for Hive replication in Cloudera Replication Manager

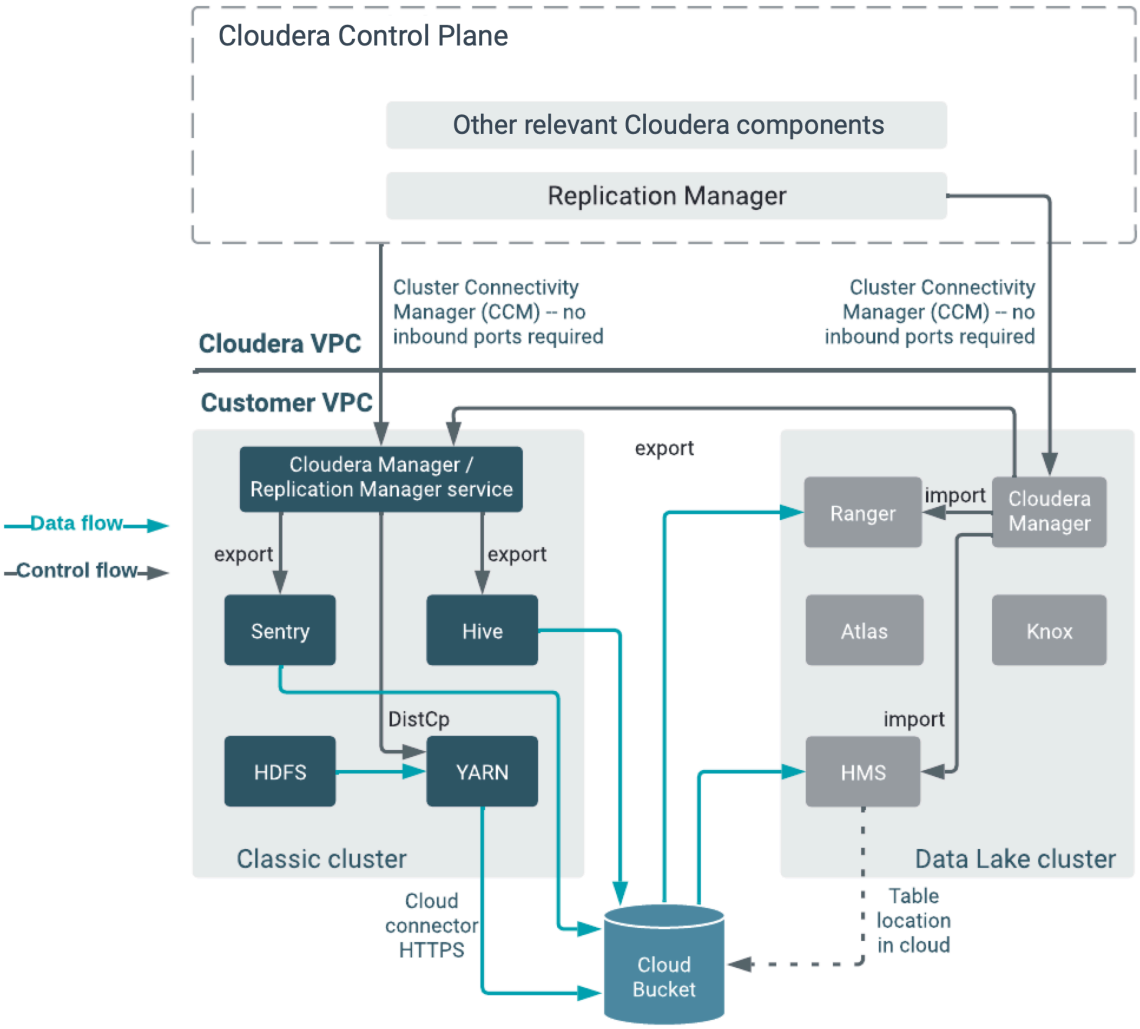

What to do next

After the clusters and cloud storage requirements are met, you can create a Hive replication policy.

Creating Hive replication policy

Before you create a Hive replication policy in Cloudera Replication Manager, you must ensure that the prerequisites are complete and the required Ranger policy is set in Ranger.

About this task

Alternatively, you can also use CDP CLI commands to create Hive replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see CDP CLI for Replication Manager.

Procedure

1. On the Cloudera Management Console Replication Manager Replication Policies page, click Add Policy.
   The **Create Replication Policy** wizard appears.
2. On the General page, choose or enter the following information:

Option	Description
Hive	Creates a Hive replication policy.
Policy Name	Enter a unique name for the replication policy.
Description	Optional. Enter a brief description about the replication policy.

The following image shows a sample General page in the **Create Replication Policy** wizard:

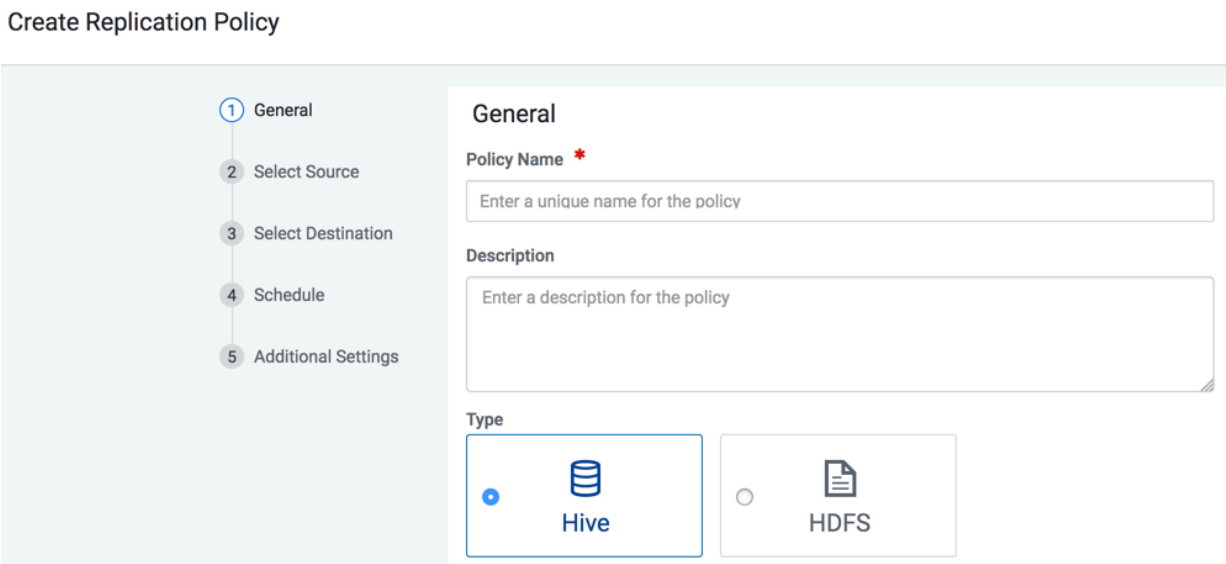

3. Click Next.
4. On the Select Source page, enter or choose the options as required:

Option	Description
Source cluster	Choose a classic cluster as a source cluster.
Source Databases and Tables	Enter the database name and table name. Click Add to enter more databases and tables as necessary. **Note:** Ensure that you do not add the sys and information_schema databases for replication if you want to choose the Additional Settings Invalidate Impala Metadata on Destination option for the Hive replication policy.
Run As Username (on source)	Optional. The replication policy uses the Default username to replicate data. Enter another username if you want the replication policy to use it to replicate data. Ensure that the user has the necessary permissions to replicate data. **Note:** You must provide a username for Kerberized clusters.

5. Click Next.

6. On the Select Destination page, enter or choose the options as required:

Option	Description
Destination Data Lake or Cloudera Data Hub	Choose a Data Lake or Cloudera Data Hub as the destination cluster. The Managed Warehouse Path and the Hive External Table Base Directory Path for the Data Lake appears. **Note:** Administrators can edit the Hive External Table Base Directory field to add another path to override the default storage location for replicated Hive external tables. Before you add another path to override the default storage location, ensure that the following steps are complete in the Ranger UI: • Alter the ranger policy Default: Hive warehouse locations in cm_s3 service to allow the Hive service to access the updated locations of S3 bucket path. • Manually update the Ranger and Sentry permissions. **Note:** The Data Hubs, using Cloudera Manager version 7.9.0 and higher, for which you have access permissions appear automatically. If you want to use a Cloudera Data Hub that appears disabled, you must upgrade the Cloudera Manager version to 7.9.0 or higher for the Data Hub.
Cloud Credential on Source	Choose the required cloud credential. The cloud credentials that you register for the selected source cluster on the Cloud Credentials page appear in this field. You can also add cloud credentials using the Add Cloud Credential link.
Run as Username	Optional. The replication policy uses the Default username to replicate data. Enter another username if you want the replication policy to use it to replicate data.

7. Click Validate Policy.

   Replication Manager verifies whether the details provided are correct.

8. Click Next.
9. On the **Schedule** page, choose or enter the following information:

Option	Description
Run Now	Starts to replicate the existing HDFS data after the replication policy creation is complete. Choose the frequency to replicate data periodically.
Schedule Run	Runs the replication policy to replicate data at a later time. Choose the date and time for the first run, and then choose the frequency to replicate data periodically. **Tip:** On the **Replication Policies** page, click to change the timezone.

Option	Description
Frequency	Choose one of the following options: • Does Not Repeat • Custom - In the Custom Recurrence dialog box, choose the time, date, and the frequency to run the policy. Replication Manager ensures that the same number of seconds elapse between the runs. For example, if you choose the Start Time as January 19, 2022 11.06 AM and Interval as 1 day, Replication Manager runs the replication policy for the first time at the specified time in the timezone the replication policy was created in, and then runs it exactly after 1 day that is, after 24 hours or 86400 seconds. **Note:** Ensure that the frequency in a schedule enables a job to finish before the next job starts. Also, ensure that the jobs based on the same policy do not overlap. If a job is not completed before another job starts, the second job does not run and the job status appears as Skipped. If a job is consistently skipped, you might need to modify the frequency of the job.

10. Click Next.

11. On the Additional Settings page, enter or choose the values as necessary:

Option	Description
YARN Queue Name	Enter the name of the YARN queue for the cluster to which the replication job is submitted if you are using Capacity Scheduler queues to limit resource consumption. The default value for this field is default.
Maximum Maps Slots	Set the maximum number of map tasks (simultaneous copies) per replication job. The default value is 20.
Maximum Bandwidth	Adjust this setting so that each map task is throttled to consume only the specified bandwidth. Each map task ((simultaneous copy) is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value. You can adjust this setting so that each map task is throttled to consume only the specified bandwidth so that the net bandwidth used tends towards the specified value. The default value for the bandwidth is 100MB per second for each mapper.
Number of concurrent HMS connections	Enter the number of concurrent Hive Metastore connections. The connections are used to concurrently import and export metadata from Hive. Increase the number of threads to improve Replication Manager performance. By default, a new replication policy uses 4 connections. • If you set the value to 1 or more, Replication Manager uses multi-threading with the number of connections specified. • If you set the value to 0 or fewer, Replication Manager uses single threading and a single connection.
Sentry Permissions	Choose Include Sentry Permissions with Metadata to migrate Sentry permissions during the replication job. Otherwise, choose Exclude Sentry Permissions from Metadata to not migrate Sentry permissions during the replication job.
Skip URI Privileges	Select this option if you do not want to include URI privileges when you migrate Sentry permissions. During migration, the URI privileges are translated to point to an equivalent location in S3. If the resources have a different location in Amazon S3, do not migrate the URI privileges because the URI privileges might not be valid.

Option	Description
Replication Option	Choose Metadata and Data to replicate metadata and data in files and directories. Otherwise, choose Metadata only to replicate the metadata of files and directories
Directory for metadata file	Enter / or a valid folder path in the target cluster to save the metadata file. If the field is empty or if the specified folder does not exist, Replication Manager creates a new folder.
Force Overwrite	Select to overwrite data in the destination metastore if incompatible changes are detected. For example, if the destination metastore was modified, and a new partition was added to a table, this option forces deletion of that partition, overwriting the table with the version found on the source. If you do not choose the option and the Hive replication process detects incompatible changes on the source cluster, Hive replication fails. This sometimes occurs with recurring replications, where the metadata associated with an existing database or table on the source cluster changes over time.
Invalidate Impala Metadata on Destination	Choose the option to run the Impala INVALIDATE METADATA statement per table on the destination cluster after completing the replication. The statement purges the metadata of the replicated tables and views within the destination cluster's Impala upon completion of replication, allowing other Impala clients at the destination to query these tables successfully with accurate results. You must run the INVALIDATE METADATA statement manually for the following scenarios: • If the destination Cloudera Manager does not have an Impala service. • If the source contains User Defined Functions (UDF). **Warning:** However, this operation is potentially unsafe if DDL operations are being performed on any of the replicated tables or views while the replication is running. In general, directly modifying replicated data/metadata on the destination is not recommended. Ignoring this can lead to unexpected or incorrect behavior of applications and queries using these tables or views.
Replication Strategy	Choose one of the following replication strategies to determine whether the file replication tasks should be distributed among the mappers statically or dynamically. • Static distributes file replication tasks among the mappers up front to achieve an uniform distribution based on the file sizes. • Dynamic distributes the file replication tasks in small sets to the mappers, and as each mapper completes its tasks, it dynamically acquires and processes the next unallocated set of tasks. The default replication strategy is Dynamic.
MapReduce Service	Choose the MapReduce or YARN service to use.
Log Path	Enter an alternate path for the logs, if required.

Option	Description
Error Handling	Select the following options as necessary: • Skip Checksum Checks - Determines whether to skip checksum checks on the copied files. If selected, checksums are not validated. Checksums are checked by default. **Note:** You must skip checksum checks to prevent replication failure due to non-matching checksums in the following cases: • Replications from an encrypted zone on the source cluster to an encrypted zone on a destination cluster. • Replications from an encryption zone on the source cluster to an unencrypted zone on the destination cluster. • Replications from an unencrypted zone on the source cluster to an encrypted zone on the destination cluster. Checksums are used for two purposes: • To skip replication of files that have already been copied. If Skip Checksum Checks is selected, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination. • To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data. • Skip Listing Checksum Checks - Whether to skip checksum check when comparing two files to determine whether they are same or not. If skipped, the file size and last modified time are used to determine if files are the same or not. Skipping the check improves performance during the mapper phase. Note that if you select the Skip Checksum Checks option, this check is also skipped. • Abort on Error - Whether to abort the job on an error. If selected, files copied up to that point remain on the destination, but no additional files are copied. Abort on Error is not selected by default. • Abort on Snapshot Diff Failures - If a snapshot diff fails during replication, the replication policy uses a complete copy to replicate data. If you select this option, the policy aborts the replication when it encounters an error instead.

Option	Description
Preserve	Choose the required options to preserve the block size, replication count, permissions (including ACLs), and extended attributes (XAttrs) as they exist on the source file system, or to use the settings as configured on the destination file system. By default source system settings are preserved. When Permission is selected, and both the source and destination clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. When Extended attributes is selected, and both the source and destination clusters support extended attributes, replication preserves them. (This option only displays when both source and destination clusters support extended attributes.) If you select one or more of the Preserve options and you are replicating to S3 or ADLS, the values all of these items are saved in metadata files on S3 or ADLS. When you replicate from S3 or ADLS to HDFS, you can select which of these options you want to preserve. **Note:** To preserve permissions to HDFS, you must be running as a superuser on the destination cluster. Use the Run As Username option to set the username.
Delete Policy	Choose the required options to determine whether the files that were deleted on the source should also be deleted from the destination directory. This policy also determines the handling of files in the destination location that are unrelated to the source. Options include: • Keep Deleted Files - Retains the destination files even when they no longer exist at the source. This is the default option. • Delete to Trash - If the HDFS trash is enabled, files are moved to the trash folder. This is not supported when replicating to S3 or ADLS. • Delete Permanently - Uses the least amount of space; use with caution. **Note:** If the source path is globbed, the replication policy ignores the Delete to Trash and Delete Permanently options and performs the Keep Deleted Files operation on the target files. The target files are *not* moved to trash or deleted regardless of the option you choose for the replication policy.
Alerts	Choose to generate alerts for various state changes in the replication workflow. You can choose to generate an alert On Failure, On Start, On Success, or On Abort of the replication job. You can configure alerts to be delivered by email or sent as SNMP traps. If alerts are enabled for events, you can search for and view the alerts on the **Events** tab, even if you do not have email notification configured. For example, if you choose Command Result that contains the Failed filter on the Diagnostics Events page, the alerts related to the On Failure alert for all the replication policies for which you have set the alert appear. For more information, see Managing Alerts and Configuring Alert Delivery.

12. Click Create.

What to do next

After the replication policy runs successfully, you can view the replication job status on the **Replication Policies** page. Verify whether the job starts and runs as expected.

The administrator can verify whether the job starts and runs as expected using the following steps:

- For HDFS data replication, check the cloud storage path (for example, S3 bucket path) to verify whether the data was successfully copied in the specified bucket.
- For Hive metadata replication, verify whether the specified source database, along with tables, partitions, UDFs and column stats are available in the Data Lake HMS instance. For this, the administrator can use a Cloudera Data Hub cluster and run the corresponding queries using Hue or beeline.

- For Ranger policies, query the Ranger policies to ensure that the Sentry policies are properly mapped as Ranger policies for the right users and groups.

Note: If the CDH source database contains functions, you must explicitly run the `reload function` command to view the migrated replication functions in the target location.

Manage and monitor Hive replication policies

After you create a Hive replication policy in Cloudera Replication Manager, you can perform and monitor various tasks related to the replication policy. You can view the job progress and replication logs. You can edit the advanced options to optimize a job run. You can suspend a job and also activate a suspended job. You can edit the replication policy as necessary.

About this task

On the **Replication Policies** page, you can perform the following actions and tasks on a replication policy and its jobs.

Procedure

- When you click Actions for a Hive replication policy, the following actions appear:

Action	Description
Edit*	Change the replication policy options as required for non-expired policies that are in active or suspended state. Based on the schedule you choose, the replication policy replicates data. You can edit the replication policies to better align with changing requirements. For example, you might want to change the frequency of a policy depending on the data size and importance of the data being replicated. **Note:** A replication policy is associated with a cluster or a cluster pair, therefore you cannot change the clusters in the policy. Optionally, expand a replication policy on the **Replication Policies** page to edit the replication policy options which include frequency (start time cannot be modified if the policy has already started), queue name, maximum bandwidth, and maximum map slots. **Tip:** To optimize the replication policy performance, you can configure the queue name, maximum bandwidth, and maximum map slots as necessary.
Delete	Deletes the replication policy permanently.
Suspend	Suspends a running replication policy. Activate the replication policy, if required.
View Log	Download, copy, or open the log. The log shows a brief output of the stdout and stderr logs of a single step of the latest replication policy job run. You can also view the current job status in the Replication Manager Overview Issues & Updates Job Status column. If the job failed, click **Failed** to view the log details about the job.
View Command Details	Opens the latest Hive replication policy job page. The steps and substeps appear in a tree view. The failed steps are expanded by default, showing the last 15 lines of the log. You can also view the command details for a Hive replication policy on the Overview Issues & Updates panel. **Tip:** To view the complete log for all the jobs, go to the target cluster Cloudera Manager Running Commands page.

Action	Description
Collect diagnostic bundle	Generates a diagnostic bundle for the replication policy. You can download the bundle as a ZIP file to your machine. Ensure that you are logged into the Cloudera Manager instances for both the source and target clusters before you download the bundle in Replication Manager.

*

To view and use the replication policies with an empty name in Replication Manager, you must understand the following implementation:

- If the API version is lower than 51, an existing replication policy with an empty name can be used and updated. However, if you edit the replication policy and provide a name for the replication policy in versions higher or equal to 51, you must ensure that the name conforms to the validation rules.
- If the API version is higher or equal to 51, it is mandatory that you provide a unique name to the replication policy to continue using it. This is because API version 51 and higher enforces the validation rules on all the replication policies.

To pass the replication policy name validation, you must ensure that the replication policy name is unique. The name can contain letters, numbers, and the _ / - characters. You must also ensure that it does not contain the characters % . ; \ nor any character that is not ASCII printable, which includes the ASCII characters less than 32 and the ASCII characters that are greater than or equal to 127.

- When you expand the policy details, the **Job History** panel appears.

  You can view the following details on the panel:

  a) Previous jobs, current job, and one future scheduled job if any.

  b) Job details which include:

Job details	Description
Started	Timestamp when the job started.
Ended	Timestamp when the job ended.
Duration	Time taken to complete the job.
Tables	Number of imported or exported tables.
Progress	Current status of a running job.
Expected	Remaining number of files and bytes expected to be copied for a running job.
Copied	Number of files and bytes copied for a running job and completed job.
Failed	Number of files and bytes that failed to be copied for a completed job.
Deleted	Number of files deleted for a completed job.
Skipped	Remaining number of files and bytes skipped from copying for a running job and complete job.

  c) Click Actions to:

  - Abort the job.
  - Re-run an aborted or failed job.
  - View Log for the job. You can download, copy, or open it to track the job and to troubleshoot any issues for the job.

- When you click a job on the **Job History** panel, the following tabs appear:

Tab	Description
General	Shows the following job details: • **Started at** timestamp • **Duration** taken to complete the job • **HDFS Replication Report** to download the job statistics in CSV format • **Hive Replication Report** to download the job statistics in CSV format • **Hive Export/Import** is the number of external tables exported or imported using Hive replication. • Number of **Errors** encountered during the replication job. • **Impala UDFs** is the number of tables exported or imported using Impala. • Job status **Message**.
Command Details	Shows the details about the commands that ran on the source Cloudera Manager for the job, along with the timestamp.
Setup Error	Shows the stack trace for the commands that ran on the source Cloudera Manager for the failed job.

- You can download the following CSV reports from the General HDFS Replication Report field to track the replication jobs and to troubleshoot issues:

Report	Description
Listing	Lists all the files and directories copied during the replication job.
Status	Shows the complete status report of each file as: • an **Error** occurred and the file was not copied. • a **Deleted** file. • an up-to-date file for which the replication was **Skipped**.
Error Status	Status report of all the copied files with errors. Each file shows the status, path, and message for the copied files with errors.
Skipped Status	Status report of all skipped files. Each file lists the status, path, and message for the databases and tables that were skipped.
Deleted Status	Status report of all deleted files. Each file lists the status, path, and message for the databases and tables that were deleted.
Performance	Summary report about the performance of the running replication job which includes the last performance sample for each mapper that is working on the replication job.
Full Performance	Performance report of the job which includes the samples taken for all mappers during the replication job.

Note: The reports are generated based on the source Cloudera Manager response. If the Cloudera Manager response is interrupted or is not handled as expected, corresponding error messages appear in HTML format in the reports.

- You can download the following CSV reports from the General Hive Replication Report field to track the replication jobs and to troubleshoot issues:

Report	Description
Hive Result	List of replicated tables.
Hive Performance	Performance report for Hive replication.

Using HBase replication policies

To create an HBase replication policy in Cloudera Replication Manager, you must register the on-premises cluster (CDH or Cloudera Private Cloud Base) as a classic cluster in Cloudera Management Console, register cloud account credentials in the Replication Manager service, verify cluster access, and configure minimum ports for replication.

Preparing to create an HBase replication policy

Before you create HBase replication policies in Cloudera Replication Manager, you must prepare the clusters, register cloud storage in Replication Manager, and verify cluster access.

Procedure

- Do the source cluster and target cluster meet the requirements to create an HBase replication policy?

  The following image shows a high-level view of the support matrix for HBase replication policies, you must consult the Support matrix for Replication Manager on Cloudera Public Cloud for the complete list of supporting clusters and scenarios:

  Figure 6: High-level replication scenarios supported by HBase replication policies

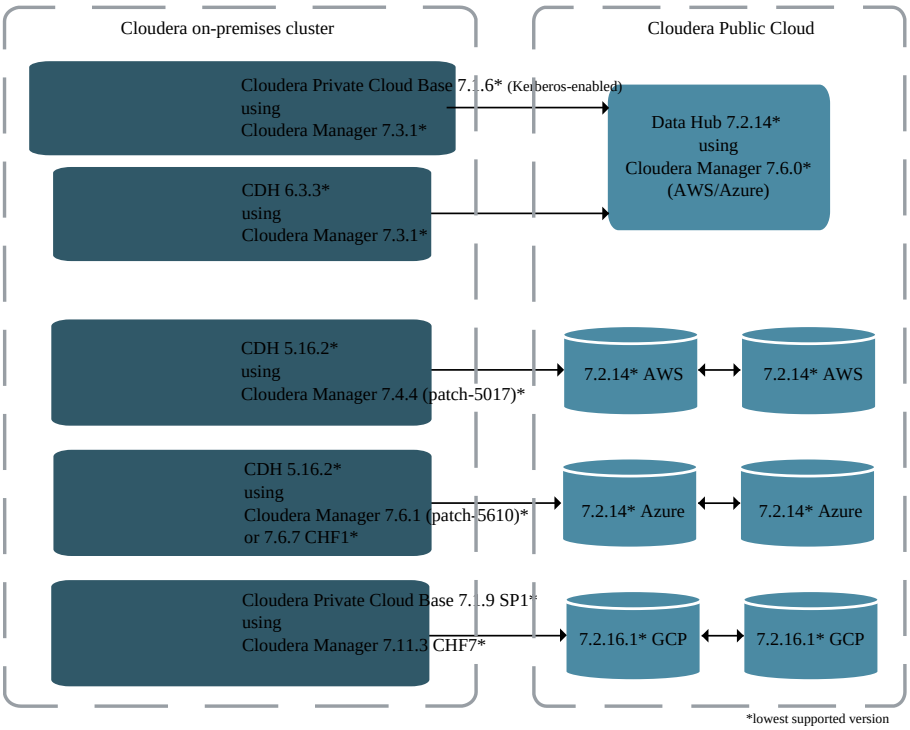

- Is the source CDH cluster or source Cloudera Private Cloud Base cluster registered as a classic cluster on the Management Console?

  CDH clusters and Cloudera Private Cloud Base clusters are managed by Cloudera Manager. To enable these on-premises clusters for Replication Manager, you must register them as Classic Clusters on the Management Console. After registration, you can use them for data migration purposes.

  Important: When you register a Cloudera Private Cloud Base cluster as a classic cluster, ensure that you use only the Cloudera Manager IP Address and Cloudera Manager Port options and not the Register KNOX endpoint (Optional) option.

  For information about registering an on-premises cluster as a classic cluster, see Adding a CDH cluster and Adding a Cloudera Private Cloud Base cluster.

  Note:

  - Cloudera Private Cloud Base 7.1.6 and higher clusters must be Kerberos enabled to use them as source classic clusters.
  - You must have *Full Administrator* role on the source on-premises cluster.
- Are the following steps complete on the source Cloudera Private Cloud Base cluster or source CDH cluster (these steps are not required for COD sources)?

  1. Complete Step 1 of Migrating HBase data from CDH/HDP to COD to install the HBase replication plugin parcel in the CDH source clusters.

     This step is applicable for CDH versions 7.2.x that are lower than 7.2.2, versions 7.1.x that are lower than 7.1.5, and for versions lower than 7.x.
  2. Create the /user/hbase folder for the hbase user in HDFS in the source cluster using the following commands:

     ```
     sudo -u hdfs hdfs dfs -mkdir /user/hbase
     sudo -u hdfs hdfs dfs -chown hbase:hbase /user/hbase
     ```

     These commands allow the HBase replication policy to replicate the existing data in the source cluster.

     This step is applicable for Cloudera Manager versions 7.4.3 or lower; Cloudera Manager 7.4.4 only if the API version is lower than v45. The endpoint http://*[***CM_HOST***]*:*[***CM_PORT***]*/api/version shows the API version of the Cloudera Manager.
- Is the required target cluster (Data Hub or COD) available and healthy?

  Note: HBase replication policies do not support source or destination Cloudera Public Cloud clusters created using the Micro Duty template.
- Do you have the necessary permission to run the HBase replication jobs on YARN?

  To verify whether you have the necessary permission, perform the following steps:

  1. Go to the source Cloudera Manager *YARN SERVICE* Configuration tab.
  2. Ensure one of the following conditions is met:

     - The Allowed System Users property must have *hbase*. Otherwise, add hbase to the existing property value.

       This property lists the users permitted (or allowed) to run containers. Note that the users with IDs lower than the Minimum User ID property might be permitted (or allowed) to run containers.
     - The Minimum User ID property is set to a value that is lower than the HBase user's ID.

       To check the HBase user's ID, SSH into a cluster node and run the id hbase command.

       The following sample snippet shows the HBase user's ID when you run the id hbase command:

       ```
       # id hbase
       uid=39993(hbase) gid=39993(hbase) groups=39993(hbase)
       ```

- Is the required cloud credential that you want to use in the replication policy registered with the Replication Manager service?

  For more information, see Working with cloud credentials.

  You can also add the following advanced configuration settings to use Google Cloud, Amazon S3, and ADLS accounts in Replication Manager:

  1. Go to the source Cloudera Manager Clusters *HDFS SERVICE* Configuration tab.
  2. Locate the HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml property.
  3. Add the following key-value pairs to register a Google account to use in Replication Manager:
     - fs.gs.impl=com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystem
     - fs.gs.project.id=*[***ENTER THE PROJECT ID***]*
     - fs.gs.system.bucket=*[***ENTER THE BUCKET NAME***]*
     - fs.gs.working.dir=/
     - fs.gs.auth.service.account.enable=true
     - fs.gs.auth.service.account.email=*[***ENTER THE SERVICE PRINCIPAL EMAIL ID***]*
     - fs.gs.auth.service.account.keyfile=/*[***ENTER THE LOCAL PATH OF THE P12 FILE***]*
  4. Add the following key-value pairs to register an S3 account to use in Replication Manager:
     - fs.s3a.access.key=*[***ENTER THE SESSION ACCESS KEY***]*
     - fs.s3a.secret.key=*[***ENTER THE SESSION SECRET KEY***]*
  5. Add the following key-value pairs to register an ADLS account to use in Replication Manager:
     - fs.azure.account.oauth2.client.id=*[***ENTER THE ABFS STORAGE CLIENT ID***]*
     - fs.azure.account.oauth2.client.secret=*[***ENTER THE ABFS STORAGE CLIENT SECRET KEY***]*
  6. Save and restart the HDFS service for the changes to take effect.
- Do you have the required cluster access to create or view replication policies?
- Have you assigned the managed identity of source roles, Storage Blob Data Owner or Storage Blob Data Contributor, to the destination storage data container and vice versa for bidirectional replication when you are using COD on Microsoft Azure?

  The roles allow writing a snapshot in the destination cluster container.
- Does DNS resolution work as expected between the source and destination clusters?

  Tip: If the destination cluster is not reachable from the source RegionServer hosts, add the hostname and IP address of the destination hosts to the /etc/hosts file on the RegionServers of the source cluster.
- Is the outgoing SSH port open on the Cloudera Manager host?
- Are the following ports open and available for Replication Manager?

Table 4: Minimum ports required for HBase replication policies

Ports	Service	Description
2181 and 16020	Destination hosts of the AWS cluster or ADLS cluster (target cluster), and the Cloudera Manager server port on the source cluster	Verify whether the ports 16020 for worker security group and 2181 for worker, master, and leader groups are open for connection from the source cluster to the destination cluster on AWS or Azure. This ensures that the source HBase service can communicate with Zookeeper and HBase services on the destination hosts uninterruptedly. For more information, see Ports for HBase replication.

Ports	Service	Description
16000	HMaster	Open the port on the Master Nodes (HBase Master Node and any back-up HBase Master node). Before you select the Validate Replication option during the first HBase replication policy creation between two specific clusters, you must ensure that the port is open on the target cluster. **Note:** Irrespective of whether this port is open or not on the Master nodes, Replication Manager displays a warning message to inform you that this port should be open on the target cluster (to communicate with the source cluster) when you choose Validate Replication on the **Select Destination** page during the HBase replication policy creation process.
7180 or 7183	Cloudera Manager Admin Console HTTP	Open on the source cluster to enable Data lake Cloudera Manager to communicate to the on-premises Cloudera Manager. Connects to destination SDX Data Lake Cloudera Manager.
9000	Cloudera Manager Agent	Open on the source and target cluster to retrieve diagnostic and log information.
6000-6049	Cluster Connectivity Manager (CCM)	Required for SSL connections to the Control Plane via CCM to communicate with Replication Manager.
80 or 443	Data transfer from secondary node for AWS / ADLS Gen2	Outgoing port. Open on all the HDFS nodes for AWS and ADLS Gen2.
8443	Data Lake cluster	Outgoing port. Configure the port on the Data Lake cluster as the outgoing port for Cloudera Management Console to communicate with Cloudera Manager and Knox.
8032	YARN Resource Manager	Open on the source and target cluster to access the YARN ResourceManager.

Consider the following best practices while using Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS):

- Ensure that the on-premises cluster (port 443) can access the *https://login.microsoftonline.com* endpoint. This is because the Hadoop client in the on-premises cluster (CDH/) connects to the endpoint to acquire the access tokens before it connects to Azure ADLS storage. For more information, see the *General Azure guidelines* row in the Azure-specific endpoints table.
- Ensure that the steps mentioned in the *General Azure guidelines* and *Azure Data Lake Storage Gen 2* rows in the Azure-specific endpoints table are complete so that the endpoint connects to the target path successfully.

- Are the following steps complete to use the Export snapshot user field during the HBase replication policy creation process?

  Replication Manager uses the specified username on the source cluster to export the initial snapshots to the target. The *hbase* user impersonates the specified *Export snapshot user* credentials and permissions to submit the HBase YARN application to export the HBase snapshot.

  To ensure the YARN application succeeds, the *hbase* user must be allowed to impersonate the *exportSnapshotUser* in HDFS by configuring the required properties and values in the HDFS core_site_safety_valve of the source cluster.

  For example, if you want to allow the impersonation from any host and if the *exportSnapshotUser* is in the *repl* user group, you can set the following key-value pairs in source Cloudera Manager Clusters *CORE SETTINGS* Configuration Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml :

  - hadoop.proxyuser.hbase.groups = repl
  - hadoop.proxyuser.hbase.hosts = *

  The Export snapshot user field appears in the **Create HBase replication policy** wizard when the following conditions are true:

  - The target cluster version is 7.2.18.200 and higher.
  - The source cluster version is 7.1.9 SP1 using Cloudera Manager 7.11.3 CHF7 and higher.
  - You choose the Select Source Perform Initial Snapshot option in the **Create HBase replication policy** wizard.

  When you use IDBroker credentials to replicate from Cloudera Private Cloud Base clusters, you must map the Kerberos username to AWS in the *[***USERNAME***]=[***ARN***]* format in the source Cloudera Manager Clusters *KNOX SERVICE* Instances Configuration Knox IDBroker AWS User Mapping property, and allow the *hbase* user to impersonate this user in IDBroker.
- (Optional) Complete the steps mentioned in the Optimize HBase replication policy performance when replicating HBase tables with several TB data FAQ if you choose Perform Initial Snapshot during HBase replication policy creation to replicate HBase tables with several TB data.

What to do next

After the clusters and cloud storage requirements are met, you can create an HBase replication policy.

Creating HBase replication policy

You can replicate HBase data from a source classic cluster (CDH or Cloudera Private Cloud Base cluster), Cloudera Operational Database, or Cloudera Data Hub to a target Cloudera Data Hub or Cloudera Operational Database cluster in Cloudera Replication Manager.

Before you begin

Ensure that the HBase replication policy prerequisites are complete.

Procedure

1. On the Cloudera Management Console Replication Manager Replication Policies page, click Add Policy.

   The **Create Replication Policy** wizard appears.
2. On the General page, choose or enter the following information:

Option	Description
HBase	Creates an HBase replication policy.
Policy Name	Enter a unique name for the replication policy.

Option	Description
Description	Optional. Enter a brief description about the replication policy.

The following image shows a sample General page in the **Create Replication Policy** wizard:

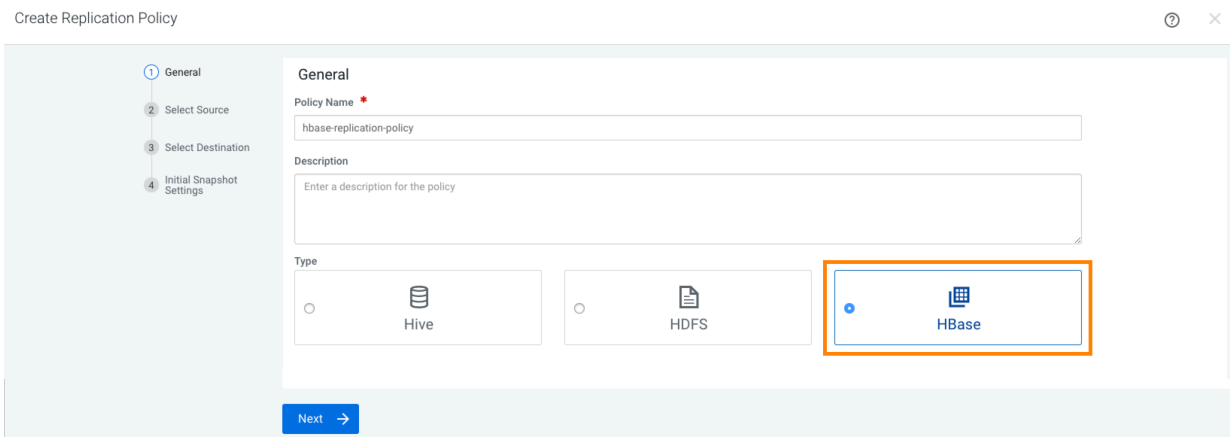

3. Click Next.
4. On the Select Source page, enter or choose the options as required:

Option	Action
Source Cluster or Database	Choose a source cluster. **Note:** HBase replication policies do not support source Cloudera Public Cloud clusters created using the Micro Duty template.
Source Tables	Enter a table name that you want to replicate. Click the Add icon to add more table names.
Perform Initial Snapshot	Select the option to replicate existing data.
Credentials are available in source cluster HDFS service configuration setting	You can choose this option when you want to use a Google Cloud account. You can use this option for S3 and ADLS accounts as well. Before you use this option, ensure that the advanced configuration settings in Preparing to create an HBase replication policy are configured.
Credentials from External Account	Choose this option for S3 and ADLS storage options. This option appears when you choose a Cloudera Private Cloud Base cluster or CDH cluster as the source cluster and you choose Perform Initial Snapshot. Click Add Cloud Credential. In the Add Cloud Credential dialog box, enter a unique name for the cloud credential. Click Save after you choose one of the following cloud storage types and enter the required options: • S3 - Choose an authentication type, enter an access key and secret key. • ADLS - Enter the client ID, tenant ID, and secret key.

Option	Action
Replicate Database	Replication Manager replicates: • current and future data from the existing tables if you choose the Perform Initial Snapshot option. Otherwise, only the future data is replicated. • data from the future tables that are created after policy creation. To replicate data from the future tables successfully, you must create similar empty tables on the target cluster. You can perform this action when you create or add a table to the database on the source cluster. You can choose the Replicate Database option only if the following conditions are true: • Target Cloudera Manager version is 7.11.0 or higher. • Source cluster version is CDH 6.x or higher. CDH 5.16.2 and higher versions also support the Replicate Database option after you upgrade the source cluster Cloudera Manager. • No existing HBase replication policies exist between the source and target clusters. **Tip:** If you want to replicate the new tables that are created after the replication policy creation is complete, you must configure the replication scope to "1" for those tables on the source cluster. To configure the replication scope for a table on the master cluster, run the alter *[***TABLE NAME***]*, {NAME => *[***COLUMN FAMILY***]*, REPLICATION_SCOPE => 1} command for each column family that must be replicated. REPLICATION_SCOPE is a column-family level attribute, where the value '0' means replication is disabled, and '1' means replication is enabled.
Replicate all user tables	Appears after you choose the Replicate Database option. Choose this option to replicate all the HBase tables in the database. This action sets the replication scope to 1 for all the tables and then replicates the tables to the target cluster.
Replicate only tables where replication is already enabled	Appears after you choose the Replicate Database option. Choose this option to replicate only those tables for which the replication scope is already set to 1. This provides you a choice to replicate only the required tables in a database. This option is supported only if the target cluster Cloudera version is 7.2.17.300 using Cloudera Manager 7.11.0-h3 or higher versions, Cloudera 7.2.16.500 using Cloudera Manager 7.9.0-h7 or higher versions, or Cloudera 7.12.0.0. To enable this feature, contact your Cloudera Account team.

Option	Action
Export snapshot user	Enter the custom username. Replication Manager uses the specified username to export the initial snapshots to the target. The option appears after you choose the Select Source Perform Initial Snapshot option. **Note:** Before you specify the username, complete the prerequisites for this field as shown in HBase replication policy prerequisites. To enable this feature, contact your Cloudera Account team.

The following sample image shows the Select Source page in the Create Replication Policy wizard when you choose a COD as source cluster:

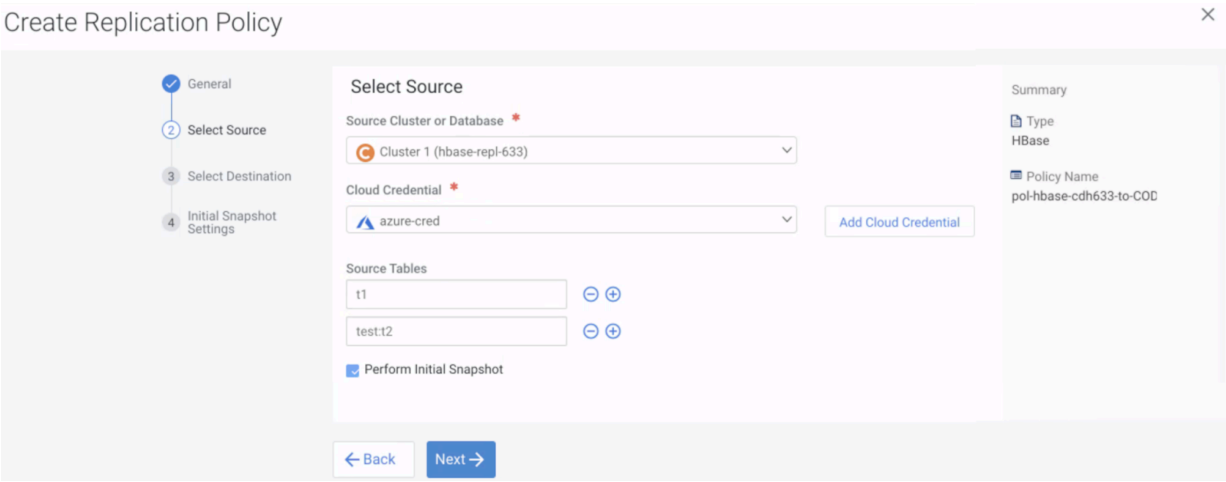

5. Click Next.
6. On the Select Destination page, enter or choose the options as required:

Option	Description
Destination Cloudera Data Hub or Cloudera Operational Database	Choose a Cloudera Data Hub cluster or Cloudera Operational Database. **Note:** HBase replication policies do not support destination Cloudera Public Cloud clusters created using the Micro Duty template.
Set HBase Replication Machine User	Optional. Choose the option and then enter the username and password. Ensure that you enter the correct password for an existing user because if the password is incorrect, the data is not replicated even though the policy is created successfully. Based on the username and password that you enter, one of the following possible scenarios is implemented by Replication Manager: • If Set HBase Replication Machine User is not selected, an HBase replication machine user is created automatically with an auto-generated username. • If Set HBase Replication Machine User is selected and Create User If Does Not Exist is not selected, ensure that the username you enter exists in the Cloudera User Management System (UMS), otherwise an error message appears. • If Set HBase Replication Machine User is selected and Create User If Does Not Exist is selected and the username does not exist in UMS, the username is created.

Option	Description
Sync Replication User	Optional. Replication Manager validates the existing username with the UMS and synchronizes the new username and password to the destination cluster's environment (and to the source's as well if the source is Cloudera Operational Database). **Note:** Error appears after you click Sync Replication User if you entered a non-existent username and did not choose Create User If Does Not Exist.
Replicate via a Network Load Balancer	Enable if the source on-premises cluster uses a network load balancer (NLB) to communicate with ZooKeeper and RegionServers of the destination Cloudera Manager of Cloudera Operational Database clusters. You can use this option when the Cloudera Operational Database clusters are isolated and the on-premises clusters can only use NLB to communicate with them. The option is disabled by default if the chosen source and target clusters already have HBase replication policies between them. To enable this feature, contact your Cloudera Account team.
Endpoint	Enter the endpoint details in the *[***NLB ADDRESS***]:[***ZOOKEEPER LISTENER PORT (2181)***]:[***TARGET HBASE ZNODE PATH***]* format. For example, my.nlb.us-west-2.amazonaws.com:2181:/hbase.

The following options appear if the source Cloudera Manager is 7.6.0 or higher:

Option	Description
Rolling HBase Service Restart on Source	[Appears if you select Cloudera Operational Database or Cloudera Data Hub as the source cluster] Select to enable automatic rolling restart* of HBase service on the source cluster after the HBase replication policy first-time setup steps are complete. Otherwise, Cloudera Manager performs an automatic full restart* of the service.
Rolling HBase Service Restart on Destination	Select to enable automatic rolling restart* of HBase service on the target cluster as a rolling restart* after the HBase replication policy first-time setup steps are complete. Otherwise, Cloudera Manager performs an automatic full restart of the service.
I want to force the setup of this HBase replication policy	Choose to run the first-time setup configuration between the selected source and destination clusters. This option appears when the selected source or target cluster is part of an existing cluster pair, and one of the following is true about the cluster pair: • No HBase replication policies exist between them. • The other cluster in the pair is currently unreachable. When you select the option, you acknowledge that the existing pairing for the selected source or target cluster will be cleared and the first-time setup will be initiated with the chosen new source or destination cluster.

Option	Description
Validate Replication	Select the option to notify Replication Manager to verify the provided details so that the replication is initiated after the policy creation is complete. **Note:** This option is available for target clusters using Cloudera Manager version 7.6.2 and higher. Before you select the Validate Replication option during the first HBase replication policy creation between two specific clusters, you must ensure that the 16000 port is open on the target cluster. **Note:** Irrespective of whether this port is open or not on the Master nodes, Replication Manager displays a warning message to inform you that this port should be open on the target cluster (to communicate with the source cluster) when you choose Validate Replication on the **Select Destination** page during the HBase replication policy creation process.
*During rolling restart, one node is restarted at a time and this continues until all the nodes in the cluster are restarted. This type of restart ensures that there is no disruption of service. During full restart, all the nodes are shut down at once and restarted simultaneously.	

The following sample image shows the Select Destination page in the Create Replication Policy wizard:

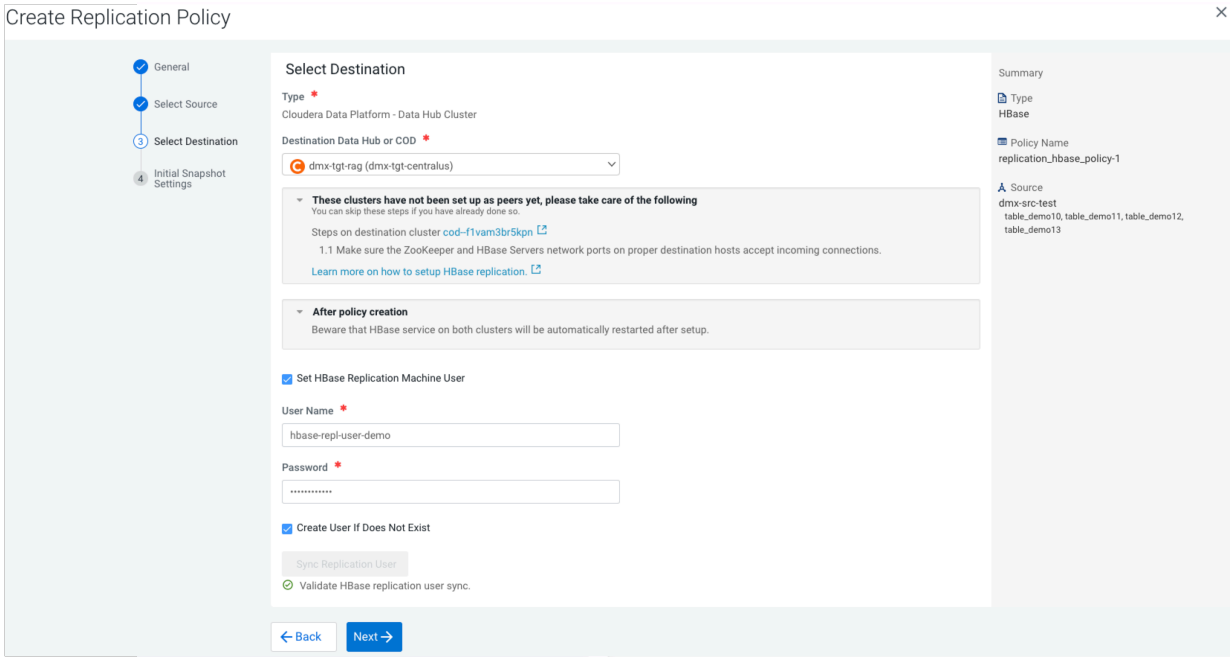

7. Click Next.
8. On the Initial Snapshot Settings page, configure the following options for the source cluster:

Option	Description
YARN Queue Name	Enter the name of the YARN queue for the cluster to which the replication job is submitted only if you are using Capacity Scheduler queues to limit resource consumption. The default value for this field is default.
Maximum Maps Slots	Configure the maximum number of map tasks (simultaneous copies) per replication job. The default value is 20.

Option	Description
Maximum Bandwidth	Adjust this setting so that each map task is throttled to consume only the specified bandwidth. Each map task ((simultaneous copy) is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value. You can adjust this setting so that each map task is throttled to consume only the specified bandwidth so that the net bandwidth used tends towards the specified value. The default value for the bandwidth is 100MB per second for each mapper. You can adjust the setting only if the source and destination Cloudera Manager instances support this option. To enable this feature, contact your Cloudera Account team.
Maximum parallel snapshots	Specify the maximum number of tables to process in parallel during the initial snapshot export and import step for the HBase replication policy. If you do not enter any value, Replication Manager chooses an appropriate value, depending on the resources in the source and target cluster, to optimize the performance. To enable this feature, contact your Cloudera Account team.

The following sample image shows the Initial Snapshot Settings page in the Create Replication Policy wizard:

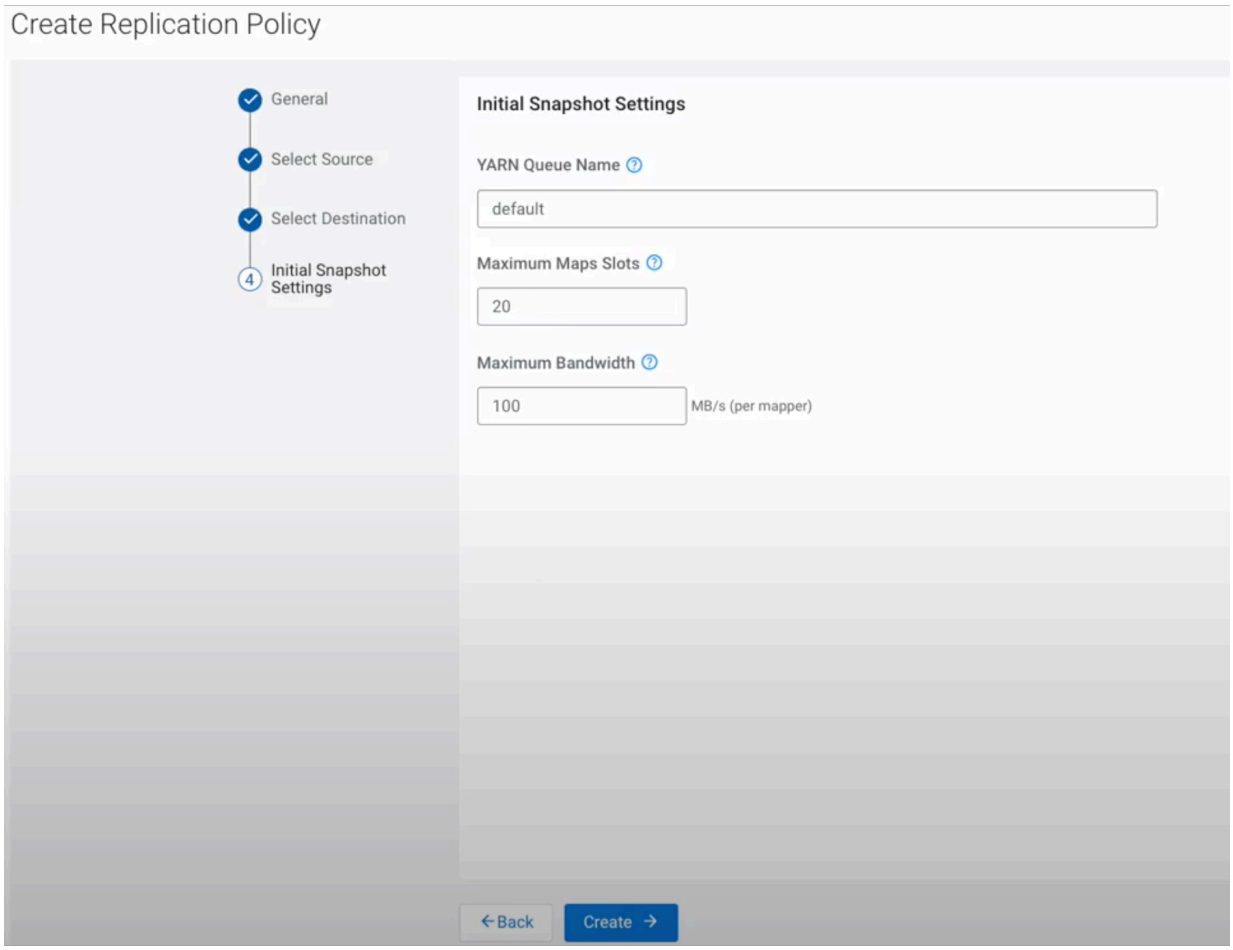

9. Click Create.

10. Restart the HBase service on the on-premises source cluster when the policy status on the **Replication Policies** page shows Manual restart (src) / restarting (dest) or Manual HBase restart needed on source. After the service restart is complete, the setup continues automatically for the replication policy. You do not need to restart the HBase service if the source is Cloudera Operational Database or Cloudera Data Hub.

    Important: If the source cluster Cloudera Manager version is 7.6.0 or lower and you are using an on-premises source cluster, you must perform the following steps to complete the HBase replication policy setup:

    a. Restart the HBase service on the on-premises source cluster when the policy status on the **Replication Policies** page shows **Waiting for 'Continue Setup' action call**.
    b. Click Continue setup for the policy on the **Replication Policies** page after the service restart is complete. This action informs Replication Manager to continue the replication policy setup.

    The following image shows the Continue setup option for the HBase replication policy on the Replication Policies page:

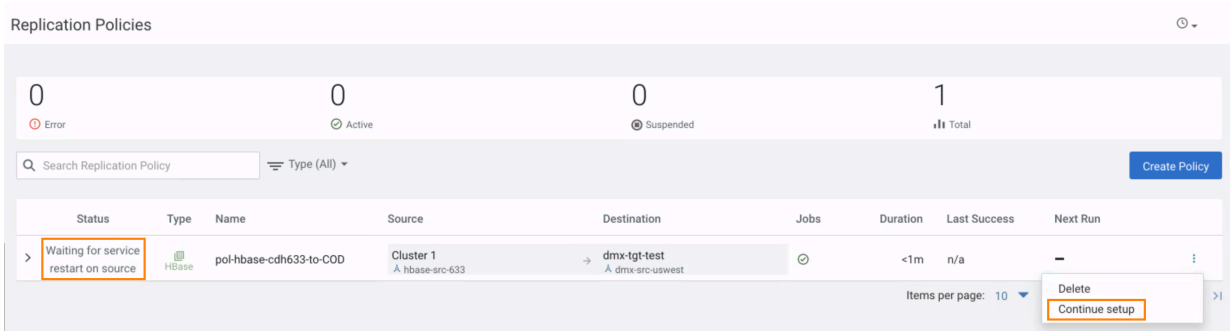

Results

After you create the first replication policy between a source cluster and target cluster (policy that is in setup/service restart state), Replication Manager creates and runs two schedules or jobs. The first schedule shows the service configuration and service restart progress, and the second schedule shows the policy creation progress. Subsequent replication policies between the same source cluster and target cluster creates only one job. Replication Manager restarts the HBase services on both the clusters if they are Cloudera Operational Database clusters.

Note: The first HBase replication policy between a source cluster and target cluster takes more time to complete because of the first-time setup configuration process that runs in the background. You can continue creating more HBase replication policies while the setup process of the in-progress policies is running in the background. The subsequent replication policies for the same source and target cluster do not run the first-time setup configuration.

What to do next

To verify whether a replication policy is running, you can either click the replication policy on the Replication Policies page, or click Running Commands in Cloudera Manager.

The following sample image shows the Job History page of a HBase replication policy between Cloudera Operational Database clusters:

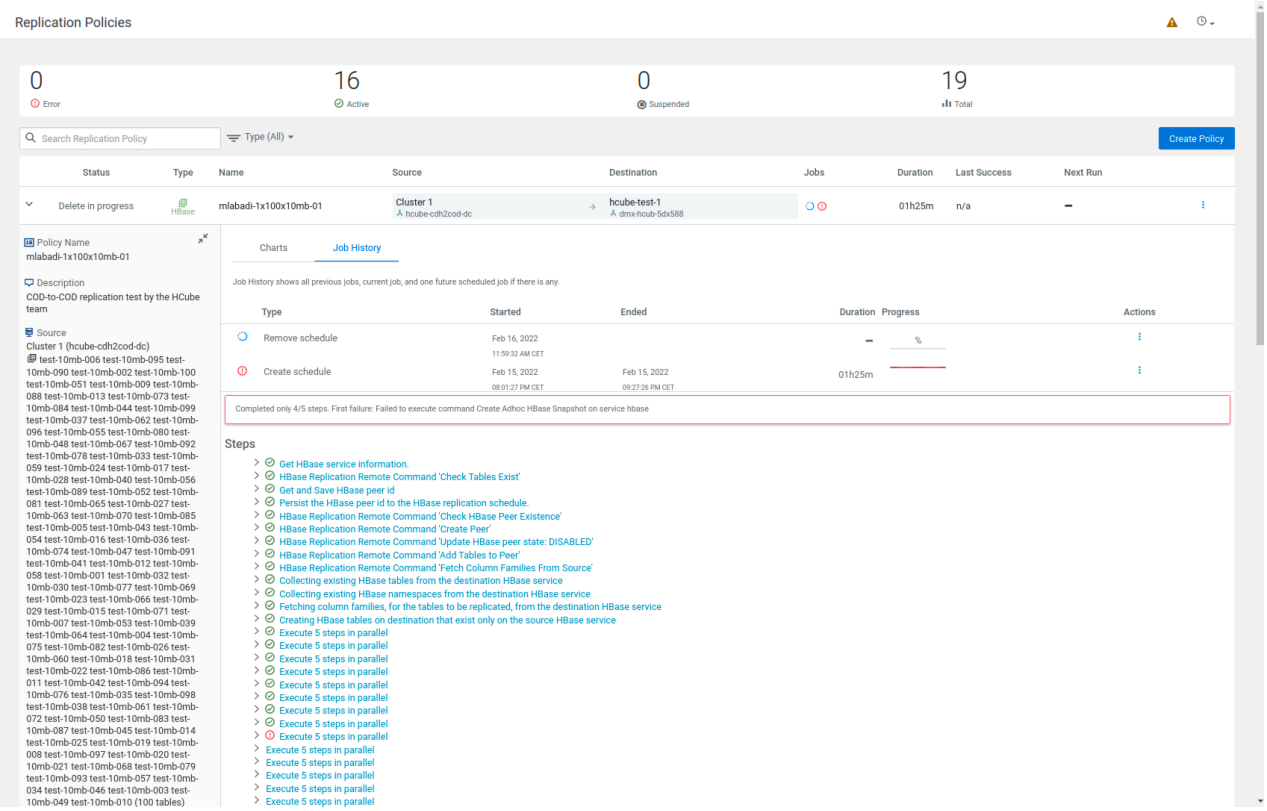

Manage and monitor HBase replication policies

After you create an HBase replication policy in Cloudera Replication Manager, you can perform and monitor various tasks related to the replication policy. You can also view the job progress and replication logs.

About this task

You can perform the following actions and tasks on the replication policy and its jobs on the **Replication Policies** page:

Procedure

- You can edit the policy name and description; view the source cluster name and tables selected in it to be replicated; and view the destination Data Hub or COD and the tables that have been replicated. You cannot remove the tables if you choose Select Source Replicate Database option during the HBase replication policy creation process.
- On the **Job History** tab, you can view the previous replication jobs, current running job, and one future scheduled job for the replication policy. You can perform **Actions** on the HBase replication policy.
- On the **Charts** tab, you can view the HBase RegionServer metrics for a specific replication peer.

Monitor HBase replication policy job details

When you click a replication policy on the "Replication Policies" page, the "Job History" tab appears. On this tab, you can view the previous replication jobs, current running job, and one future scheduled job for the replication policy.

Replication policy job details

The following columns appear for each replication policy:

- **Type** of job. Click the job to view the completed and in-progress steps that Replication Manager takes to complete the job. Here, you can verify whether the first-time setup configuration between the source and destination clusters was successful.

  Tip: You can also view the progress and results of the first-time setup on the Cloudera Manager Running Commands page for the source and destination clusters.

- Timestamp when the job **Started**.
- Timestamp when the job **Ended**.
- **Duration** taken to complete the job.
- **Progress** of the job run.

Actions menu

On the **Actions** menu, you can perform the following tasks on the HBase replication policy:

Option	Description
Edit*	Enables you to edit the replication policy name and description, or delete one or more tables for the HBase replication policy. You can also edit **Source** to delete the tables in the HBase replication policy.
Delete	Removes the HBase replication policy permanently. You can Force Delete an HBase replication policy after Replication Manager fails to delete the replication policy.
Suspend	Pauses an active HBase replication policy. This option appears for target clusters using Cloudera Manager 7.9.0 (Cloudera Manager API** version 51) and higher. **Warning:** When you suspend an HBase replication policy, Replication Manager suspends all the HBase replication policies between the same source and destination cluster. You can suspend only the specified replication policy for target clusters using Cloudera Manager with API version lower than 45. However, the Suspend option is not available for clusters using Cloudera Manager with API versions 45 through 50.
Activate	Resumes a suspended HBase replication policy. **Warning:** When you activate a suspended HBase replication policy, Replication Manager activates all the suspended HBase replication policies between the same source and destination cluster. You can activate only the specified suspended policy for target clusters using Cloudera Manager with API** version lower than 51.
View command details	Opens the latest HBase replication policy job page. The steps and substeps appear in a tree view. The failed steps are expanded by default, showing the last 15 lines of the log. You can also view the command details for a Hive replication policy on the Overview Issues & Updates panel. To view the complete log for all the jobs, go to the target cluster Cloudera Manager Running Commands page.

Option	Description
Retry First Time Setup	Runs the first-time setup configuration between the source and destination clusters if the first-time setup has failed. This option is available only if the first-time setup configuration fails. After you click Retry Create, you can choose: • to set an Set HBase Replication Machine User and Sync Replication User to run the first-time setup. • to opt for Rolling HBase Service Restart on Source and Rolling HBase Service Restart On Destination after the first-time setup is complete. For more information about these options, see *Creating HBase replication policy*.
Retry Failed Snapshots	Reruns the failed initial snapshots (and only the failed ones) in the replication policy if the Replication Manager failed to replicate the existing data of some tables. This option appears if you selected Select Source Perform Initial Snapshot during policy creation and Replication Manager failed to replicate the existing data. In this scenario, the policy status shows **Active** with a *Snapshot Failure* warning message on the **Replication Policies** page for the HBase replication policy.
Collect diagnostic bundle	Generates a diagnostic bundle for the replication policy. You can download the bundle as a ZIP file to your machine. Ensure that you are logged into the Cloudera Manager instances for both the source and target clusters before you download the bundle in Replication Manager.

*

To view and use the replication policies with an empty name in Replication Manager, you must understand the following implementation:

- If the API version is lower than 51, an existing replication policy with an empty name can be used and updated. However, if you edit the replication policy and provide a name for the replication policy in versions higher or equal to 51, you must ensure that the name conforms to the validation rules.
- If the API version is higher or equal to 51, it is mandatory that you provide a unique name to the replication policy to continue using it. This is because API version 51 and higher enforces the validation rules on all the replication policies.

To pass the replication policy name validation, you must ensure that the replication policy name is unique. The name can contain letters, numbers, and the _ / - characters. You must also ensure that it does not contain the characters % . ; \ nor any character that is not ASCII printable, which includes the ASCII characters less than 32 and the ASCII characters that are greater than or equal to 127.

**

The endpoint http://*[***CM_HOST***]*:*[***CM_PORT***]*/api/version shows the API version of the .

Creating triggers and monitoring replication-related metrics in Cloudera Manager

After you create an HBase replication policy between two Cloudera Operational Database (COD) clusters in Cloudera Replication Manager, you can set up the triggers in Cloudera Manager to monitor specific replication-related metrics for the HBase replication policy. When the condition specified in the trigger is met, Cloudera Manager triggers an alert, and you can view the metrics for the replication policy in Cloudera Replication Manager and in Cloudera Manager.

Procedure

1. Go to the source Cloudera Manager Hosts All Hosts page.
2. Click a link in the **Name** column to open the host status page.
3. Click Create Trigger in the **Health Tests** section.

   For information about creating a trigger, see Cloudera Manager Trigger Use Cases. For information about the list of supported HBase RegionServer replication peer metrics, see HBase RegionServer Replication Peer Metrics.

4. Enter the required values and the following expression to create a trigger for a metric with a certain threshold value on the **Create New Trigger** page:

   IF (SELECT *[***ENTER THE METRIC NAME***]* WHERE serviceName=$SERVICENAME AND max(*[***ENTER THE METRIC NAME***]*) > *[***ENTER THE THRESHOLD VALUE***]* AND roleType=RegionServer) DO health:concerning

   The following examples illustrate how you can use triggers to monitor the replication metrics in Cloudera Manager:

   a) The `IF (SELECT log_queue_size WHERE serviceName=$SERVICENAME AND max(log_queue_size) > 10 AND roleType=RegionServer) DO health:concerning` trigger statement triggers the health alert when the value of the log_queue_size is greater than 2.

      This trigger condition is met when the processing of the queue becomes slow which results in the logs being queued or the replication process being halted.

   b) The following sample image shows the chart that appears on the source Cloudera Manager Charts page when the health alert is triggered for the following trigger expression: `IF (SELECT shipped_batches_rate WHERE serviceName=$SERVICENAME AND max(shipped_batches_rate) > 0.01 AND roleType=RegionServer) DO health:concerning`

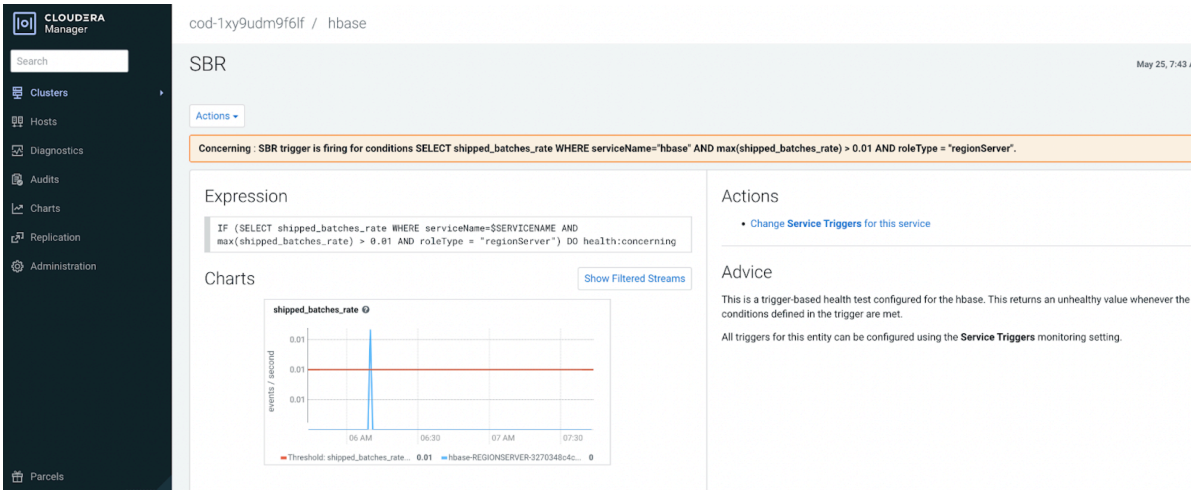

   c) The following image shows the **Chart Builder** where you can configure the options as required and save it for troubleshooting purposes or for future references.

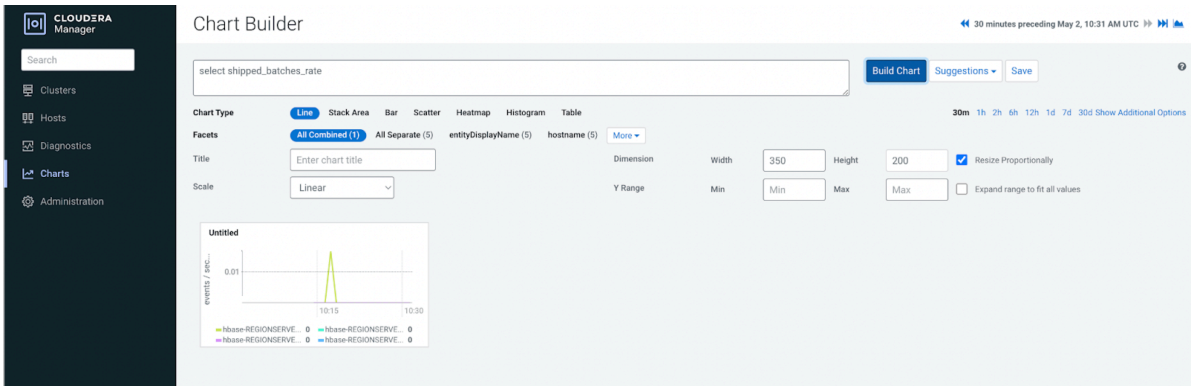

   For information about monitoring the metrics related to replication and HBase health on your COD environment, see Monitor COD metrics.

What to do next

You can view and monitor the metrics in Cloudera Manager and on the **Charts** page in Cloudera Replication Manager.

Monitor HBase RegionServer replication peer metrics in Replication Manager

When you click a replication policy on the "Replication Policies" page, the "Charts" tab appears. On this tab, when you choose the HBase replication peer metric, aggregate rollup level, start time, and end time, a graphical representation of HBase RegionServer metrics specific to the replication peer appears.

Cloudera Manager monitors several metrics which includes performance metrics for the HBase RegionServer. Replication Manager leverages this capability to display the HBase RegionServer metrics specific to a replication peer as a graph on the **Charts** tab for a replication policy, and stores it in the source cluster of the replication policy. You can use these metrics to monitor a HBase replication job and to find and diagnose issues related to the HBase replication peer.

You can view these metrics for a replication peer only if the following conditions are true:

- The source Cloudera Manager API version is 42 or higher and the target Cloudera Manager API version is 53 or higher.

  Tip: The endpoint http://*[***CM HOST***]*:*[***CM PORT***]*/api/version shows the API version of the Cloudera Manager.

- The HBase replication policy has been successfully created and is not in an erroneous state.

Note: If the target Cloudera Manager version is 45 or higher, the graphs generated for HBase replication policies are the same if they share the same source and target clusters. If the target Cloudera Manager version is lower than 45, the graph differs from one HBase replication policy to another because one replication peer is created for each HBase replication policy in these API versions.

For more information about the available metrics, see HBase RegionServer Replication Peer Metrics.

Viewing HBase RegionServer replication peer metrics

You can view HBase RegionServer metrics specific to a replication peer as a graph on the "Charts" tab for a replication policy on the "Replication Policies" page.

Procedure

1. Go to the **Replication Policies** page in Cloudera Replication Manager.
2. Click a successful HBase replication policy that is not in an erroneous state.
3. Go to the **Charts** tab.

4. Choose the following options depending on your requirement:
   a) Choose one of the following HBase replication peer metric:
      - Age of last shipped operation
      - Age of last shipped operation rate
      - Log edits filtered rate
      - Log edits read rate
      - Log queue size
      - Log read in bytes rate
      - Shipped batches rate
      - Shipped ops rate
      - Shipped size in kb rate
      - Shipped hfiles rate
      - Size of hfile refs queue
   b) Choose an aggregate rollup level for the metrics:
      - Raw
      - 10 minutes
      - 1 hour
      - 6 hours
      - Daily
      - Weekly
   c) Choose a Start Time.
   d) Choose an End Time.
5. Click Load Metrics to view the graphs.

Troubleshooting replication policies in Cloudera Replication Manager

The troubleshooting scenarios in this topic help you to troubleshoot issues in the Cloudera Replication Manager.

Different methods to identify errors related to failed replication policy

What are the different methods to identify errors while troubleshooting a failed replication policy?

Procedure

You can choose one of the following methods to identify the errors to troubleshoot a job failure:

- On the **Replication Policies** page, click the failed job in the **Job History** pane. The errors for the failed job appear.

  The following sample image shows the **Job History** pane for a replication policy job:

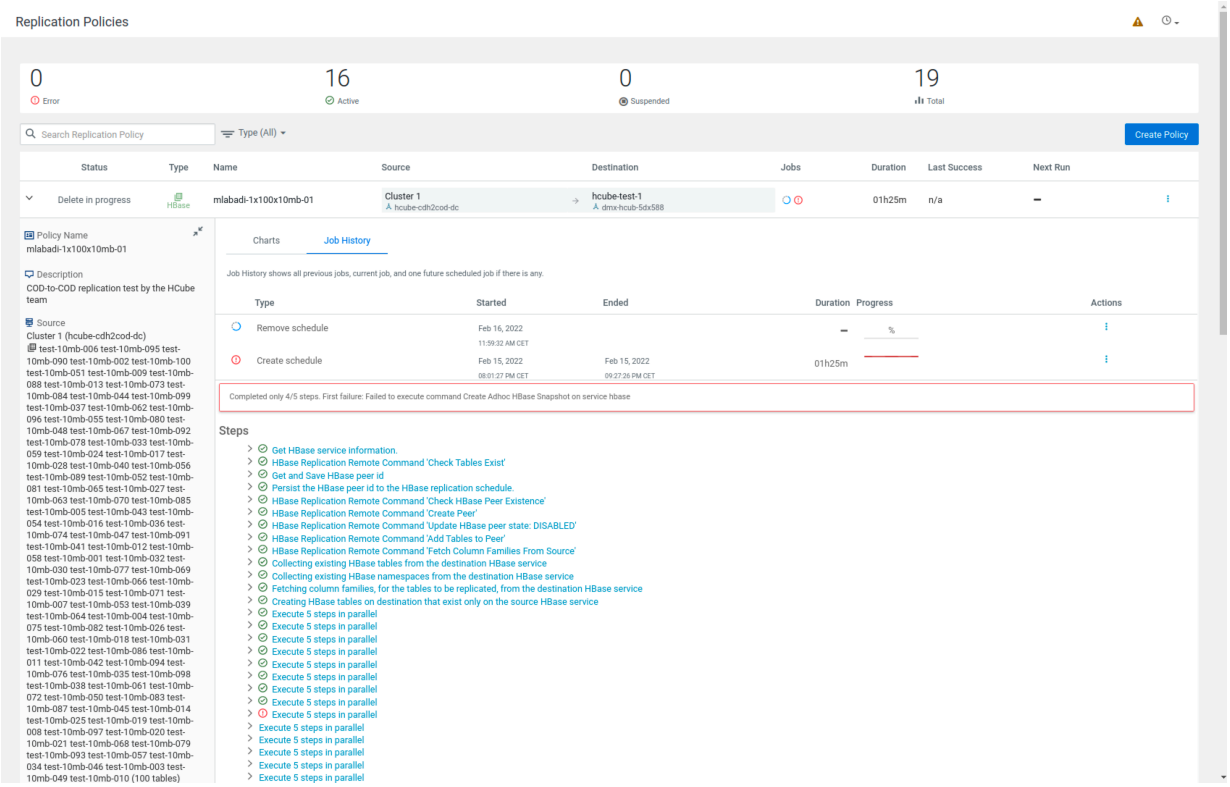

- In the source and target Cloudera Manager, click **Running Commands** on the left navigation bar. The recent command history shows the failed commands.

  The following sample image shows the **Running Commands** page for an HBase replication policy:

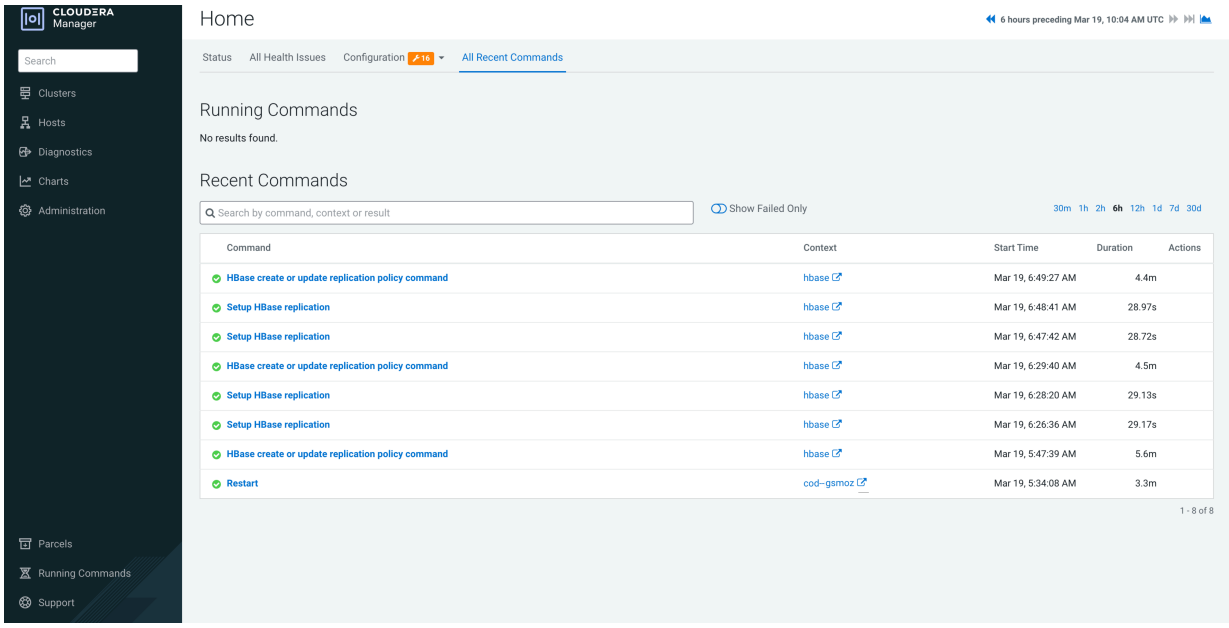

- On the source cluster and target cluster, open the service logs to track the errors (For example, HBase service logs).

  You can also search on the Cloudera Manager Diagnostics Logs page to view the logs.

Replication Policies page does not display all the replication policies

The "Replication Policies" page might not display all the replication policies depending on various factors. In such scenarios, you can choose to reload the page, choose a load page option, or use CDP CLIs to view and monitor the replication policies and its statistics.

Problem

When a Cloudera Manager instance is slow, that is while handling more than 650 replication policies or when it is generally under heavy load, it might slow down the ‘policy list request’ operation. In such scenarios, the replication policies take more time than expected to appear, or might not get displayed on the **Replication Policies** page.

Solution

Procedure

- Force reload page using the option on the **Replication Policies** page.

  You can use this option if the ‘policy list request’ operation has timed out on the Cloudera Manager. You can identify this scenario when the cluster that stores the replication policies shows up in the Error list. Click

  to see the error list and the list of unreachable clusters.

  Tip: Sometimes, Replication Manager fails to reach a healthy Cloudera Manager when there is a temporary networking blip or when there is a load spike on Cloudera Manager. When a cluster becomes unreachable for Replication Manager, the cluster is placed in the list of unreachable clusters. Replication Manager retries to reach the cluster again after 20 minutes. After you confirm that the Cloudera Manager is healthy and expect it to be reachable by Replication Manager, you can force reload the **Replication Policies** page to reconnect every cluster.

- Omit the job history of the policies to speed up the ‘policy list query’ operation using the Never load history option.

  Tip: Depending on your requirements, you can choose one of the following options in

  to load the **Replication Policies** page faster by delaying to load the job history:

  - Delay loading job history when it takes too long attempts to load the job history, but omits the load operation above a certain threshold. By default, Replication Manager uses this option.
  - Never load job history minimizes the load on Cloudera Manager and maximizes Replication Manager performance.
  - Always load job history ensures that the job history is always loaded for all the displayed replication policies.

  The following sample image shows the options for

  :

  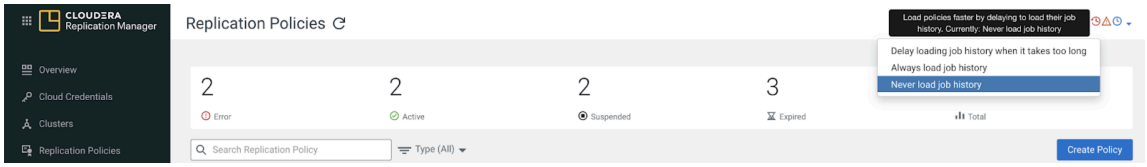

- If the Replication Manager UI still performs slowly and you are not able to view the job history and monitor the dashboards, Cloudera recommends that you use CDP CLIs for Replication Manager. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option.

  For more information, see CDP CLI for Replication Manager.

  Important: Policy statistics and job details are incorrect if the job history for a replication policy does not load completely. This is because the job history is necessary to decide whether a policy failed or succeeded.

  You can identify this scenario when warnings appear on the **Overview** or **Replication Policies** pages. The following sample images shows the warnings that appear on the **Overview** and **Replication Policies** pages respectively:

  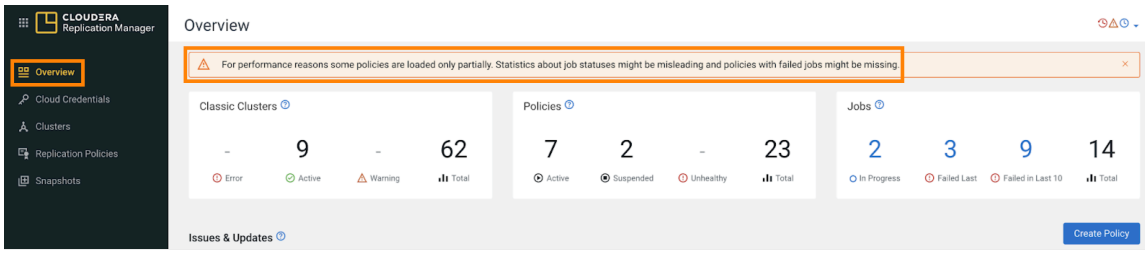

  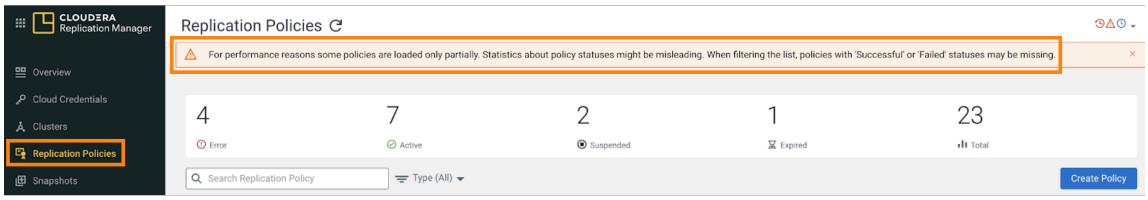

HDFS replication policy fails due to export HTTPS_PROXY environment variable

HDFS replication policies fail when the export HTTPS_PROXY environment variable is set to access AWS through proxy servers. How to resolve this issue?

Remedy

Procedure

To resolve this issue, perform the following steps:

1. Open the core-site.xml file on the source cluster.
2. Enter the following properties in the file:

```
<property>
  <name>fs.s3a.proxy.host</name>
  <description>Hostname of the (optional) proxy server for S3 connection
s.</description>
</property>

<property>
  <name>fs.s3a.proxy.port</name>
  <description>Proxy server port. If this property is not set
    but fs.s3a.proxy.host is, port 80 or 443 is assumed (consistent with
    the value of fs.s3a.connection.ssl.enabled).</description>
</property>
```

3. Save and close the file.
4. Restart the source Cloudera Manager.
5. Run the failed HDFS replication policies in Replication Manager.

   Replication Manager completes the replication successfully.

Cannot find destination clusters for HBase replication policies

When you ping destination clusters using their host names, the source cluster hosts for HBase replication policies do not find the destination clusters. How to resolve this issue?

Cause

This might occur for on-premises clusters such as Cloudera Private Cloud Base clusters or CDH clusters because the source clusters are not on the same network as the destination Data Hub. Therefore, hostnames cannot be resolved by the DNS service on the source cluster.

Remedy

Procedure

Add the destination Region Server and Zookeeper IP to host name mappings in the /etc/hosts files of all the Region Servers on the source cluster.

The following snippet shows the contents in a sample /etc/hosts file:

```
10.115.74.181 dx-7548-worker2.dx-hbas.x2-8y.dev.dr.work
10.115.72.28 dx-7548-worker1.dx-hbas.x2-8y.dev.dr.work
10.115.73.231 dx-7548-worker0.dx-hbas.x2-8y.dev.dr.work
10.115.72.20 dx-7548-master1.dx-hbas.x2-8y.dev.dr.work
10.115.74.156 dx-7548-master0.dx-hbas.x2-8y.dev.dr.work
10.115.72.70 dx-7548-leader0.dx-hbas.x2-8y.dev.dr.work
```

HBase replication policy fails when Perform Initial Snapshot is chosen

An HBase replication policy fails for COD on Microsoft Azure when the "Perform Initial Snapshot" option is chosen but data replication is successful when the option is not chosen. How to resolve this issue?

Cause

This issue appears when the required managed identity of source roles are not assigned.

Remedy

Procedure

Assign the managed identity of source roles, Storage Blob Data Owner or Storage Blob Data Contributor, to the destination storage data container and vice versa for bidirectional replication.

The roles allow writing a snapshot in the destination cluster container.

Optimize HBase replication policy performance when replicating HBase tables with several TB data

Can HBase replication policy performance be optimized when replicating HBase tables with several TB of data if the "Perform Initial Snapshot" option is chosen during HBase replication policy creation?

Complete the following manual steps to optimize HBase replication policy performance when replicating several TB of HBase data if you choose the Perform Initial Snapshot option during the HBase replication policy creation process.

Remedy

Procedure

1. Before you create the HBase replication policy, perform the following steps:
   a) Navigate to the source Cloudera Manager YARN service Configuration tab.
   b) Search for the mapreduce.task.timeout parameter.
   c) Increase the value or set it to 0 to switch off the timeout.
   d) Restart the YARN service.
   e) Navigate to the source Cloudera Manager HBase service Configuration tab.
   f) Search and configure the following key-value pairs:
      - hbase.snapshot.master.timeout.millis = 840000
      - hbase.client.sync.wait.timeout.msec = 180000
      - hbase.client.operation.timeout = 2400000
      - hbase.client.procedure.future.get.timeout.msec = 3000000
      - hbase.hfilearchiver.thread.pool.max=100
      - hbase.snapshot.thread.pool.max=24
   g) Restart the HBase service.
   h) Perform steps e through g on the target Cloudera Manager.
2. When you create the HBase replication policy for the first time using the above configured source cluster, you must increase the Maximum Map Slots value to a higher number on the Advanced Settings page.

   Tip: Calculate this number by multiplying the number of nodes on the source cluster and the number of cores in the node.

3. If Store File Tracking (SFT) is enabled in the target COD, perform the steps mentioned in the COD migration topic after the replication policy creation is complete.

   Note: SFT is enabled by default on Cloudera clusters with COD version 7.2.14.2 and higher using Cloudera Manager versions 7.2.16 and higher.

Partition metadata replication takes a long time to complete

How can partition metadata replication be improved when the Hive tables use several Hive partitions?

Hive metadata replication process takes a long time to complete when the Hive tables use several Hive partitions. This is because the Hive partition parameters are compared during the import stage of the partition metadata replication process and if the exported and existing partition parameters do not match, the partition is dropped and recreated. You can configure a key-value pair to support partition metadata replication.

Procedure

1. Go to the Cloudera Manager Clusters *HIVE SERVICE* Configuration tab.
2. Search for the Hive Replication Environment Advanced Configuration Snippet (Safety Valve) property.
3. Enter the HIVE_IGNORED_PARTITION_PARAMETERS=*[***COMMA SEPARATED LIST OF HIVE PARTITION PARAMETERS***]* key-value pair.

   For example,

   ```
   HIVE_IGNORED_PARTITION_PARAMETERS=transient_lastDdlTime,totalSize,numRow
   s,COLUMN_STATS_ACCURATE,numFiles
   ```

   The partition parameter names you provide are not compared during the import stage of the partition metadata replication process. Therefore, even if the partition parameters do not match between the exported and existing partitions, the partition is not dropped or recreated. After you configure this key-value pair, the import stage of the partition metadata replication process completes faster.
4. Save the changes, and restart the Hive service.

Replicating Hive nested tables

Cloudera Replication Manager does not support Hive nested tables. What do I do if there are Hive nested tables in the source cluster?

Cloudera Replication Manager does not support Hive nested tables for replication. Therefore, it is recommended that you move the nested tables to a different location in HDFS and then replicate Hive external tables. However, if this is not possible, you can perform the following steps in the given order as a workaround.

Solution

Procedure

1. Create a Hive replication policy on the target cluster. Ensure that the Additional Settings Replication Option Metadata only option is selected to replicate the metadata of required files and directories.

   Note: If you are using Cloudera Public Cloud 7.2.15 or lower and if you are using Amazon S3 as the disaster recovery cluster, ensure that you (the administrator) run the following command after the Hive metadata replication is complete:

   ALTER TABLE table SET LOCATION "*[***S3A://S3_BUCKET/WAREHOUSE_PATH***]*";
2. Create a HDFS replication policy on the source cluster to replicate the table data.

Target HBase folder is deleted when HBase replication policy fails

When the snapshot export fails during the HBase replication policy job run, the target HBase folder in the destination Data Hub or COD gets deleted.

Note: This scenario appears if you are using Cloudera Manager versions that are lower than 7.6.7 CHF8, 7.11.0, or 7.9.0-h6 on the source cluster.

You can either revoke the delete permission for the user, or ensure that you use an access key/role that does not have delete permissions to the required storage component.

The following steps show how to create an access key in AWS and an Azure service principal, which do not have delete permission for the storage component.

Solution in AWS

Procedure

If the Data Hub or COD is on Amazon S3, you can perform the following steps to create an access key that does not have delete permission for the storage component:

1. Login to AWS.
2. Click Create policy on the IAM Policies Create policy page.

   The following sample image shows the **Policies** page in the AWS IAM section to create a policy.

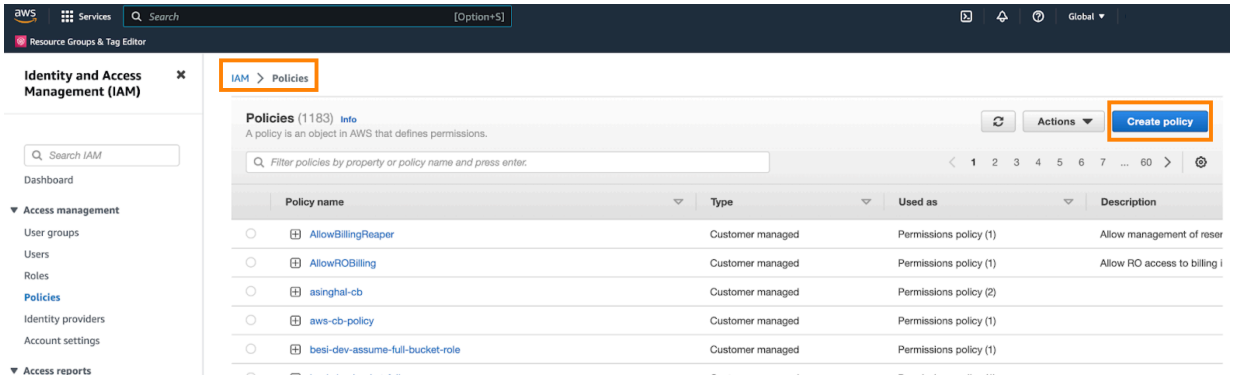

3. Choose the S3 Service, and then choose the following permissions for generic access (assign resources as required):
   - *List/ListBucket*
   - *Read/GetBucketLocation*
   - *Read/GetObject*
   - *Write/AbortMultipartUpload*
   - *Write/PutObject*
   - *Permissions management/PutBucketPublicAccessBlock*
4. Add *Delete/DeleteObject* permission to the target COD cluster's snapshot temporary folder.
   For example, the target COD cluster's snapshot temporary folder might be located in *[***TARGET COD S3 PATH***]*/hbase/.hbase-snapshot/.tmp/*.
5. Enter a Name for the policy, add tags, and click Create policy.
6. Click Add Users to create a user on the IAM Users page.
7. Enter a Name, and click Next.
8. Choose the Attach policies directly option on the **Set permission** page, and then assign the previously created policy to the user.
9. Optionally, add tags and create the user.
10. Click Create access key to create an access key for the user on the IAM Users *[***NEW USER***]* page.
11. On the Security Credentials Access keys page, choose Application running outside AWS. Click Next.

12. Optionally, attach the tags, create and save the access key. This access key is used as an external account for replication.

    How do I verify whether the target HBase folder in the destination Data Hub or COD does not get deleted if the snapshot export fails during the HBase replication policy job run?

    Perform the following steps to verify if the delete operation is allowed for the access key that you previously created:

    a. Run the aws configure --profile delete-test command to setup the credentials in AWS CLI.
    b. Delete an arbitrarily created temporary file from the account using the aws s3 --profile delete-test rm --recursive s3:*//[***ACCOUNT NAME***]*/delete-testing/ command.

    The delete operation is not allowed.

Solution in Microsoft Azure

Procedure

If the Data Hub or COD is on ABFS, you can perform the following steps to create an Azure service principal that does not have delete permission for the storage component:

1. Login to Microsoft Azure.
2. Click Add Add a custom role on the Subscriptions Select the subscription Access Control (IAM) page in Microsoft Azure, and complete the following steps:
   a) On the **Basics** tab, provide a name for the role, select Clone a role for the **Baseline permissions** field, and choose Storage Blob Data Contributor for the **Role to clone** field. Click Next.
   b) On the **Permissions** tab, remove the *Microsoft.Storage/storageAccounts/blobServices/containers/delete* and *Microsoft.Storage/storageAccounts/blobServices/containers/blobs/delete* permissions.
   c) Click Review + create.

   The following sample image shows the Access control (IAM) page in Microsoft Azure:

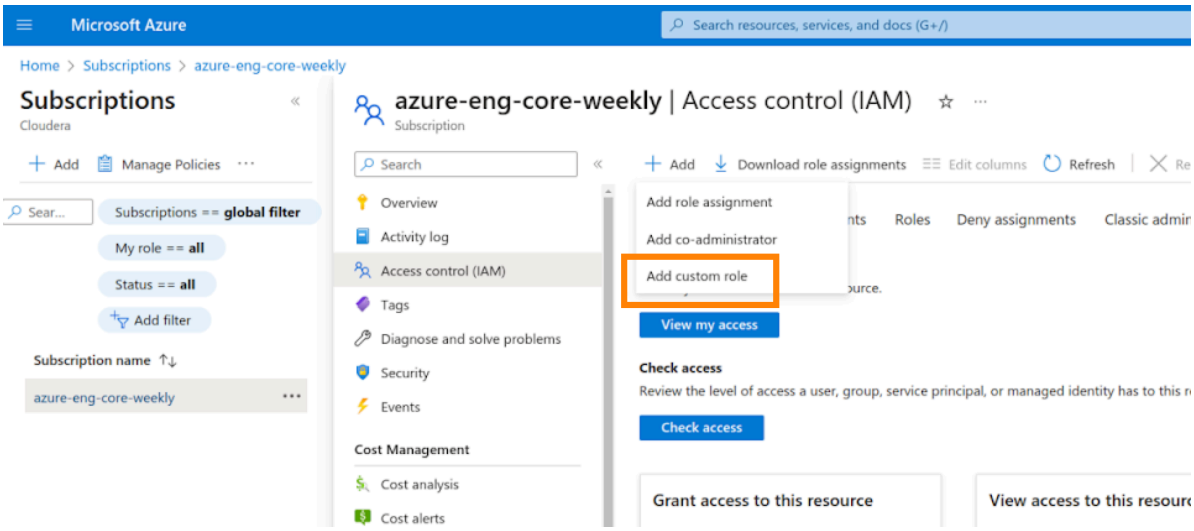

3. Click Add Add role assignment on the Storage accounts Select the account where snapshots will be written Access Control (IAM) page, and complete the following steps:
   a) On the **Role** tab, select the custom role previously created. Click Next.
   b) On the **Members** tab, select User, group, or service principal for **Assign access to** field, and select the required service principal.
   c) Click Review + assign.
   d) Click Review + assign on the **Conditions (optional)** tab.

4. Click Add principal on the Storage accounts *[***ACCOUNT NAME***] [***CONTAINER WHERE SNAPSHOTS ARE TO BE WRITTEN***]* Manage ACL page.
   a) Select the required service principal.
   b) Choose the Execute permission for the required container, and click Save.

   Important: Ensure that no other role containing delete access to the given storage account is assigned to the new service principal on the Access Control (IAM) page.

   How do I verify whether the target HBase folder in the destination Data Hub or COD does not get deleted if the snapshot export fails during the HBase replication policy job run?

   To verify if the delete operation is allowed on the service principal that you previously created, perform the following steps:

   a. Open the Azure Cloud Shell terminal.
   b. Login using the service principal that you created previously using the az login --service-principal -u *[***CLIENT ID***]* -p *[***CLIENT SECRET***]* --tenant *[***TENANT ID***]* command.
   c. Delete an arbitrarily created temporary file from the account using the az storage fs file delete --path *[***TEMPORARY FILE***]* -f data --account-name *[***ACCOUNT NAME***]* --auth-mode login command.

   The delete operation is not allowed.

Replicate HBase data in existing and future tables

Errors might appear when you try to replicate HBase data from existing tables and future tables in a database using the “Replicate Database” option during the HBase replication policy creation process. These errors appear when there are compatibility issues.

The following list shows a few errors that might appear and how to mitigate these issues:

Procedure

- The error *Destination cluster doesn't support replication of all HBase tables. Please change the destination cluster or specify the tables that should be replicated.* appears during HBase replication policy creation process. How to mitigate this issue?

  This error appears if you choose the Select Source Replicate Database option during the HBase replication policy creation process for unsupported cluster versions.

  To mitigate this issue, ensure that the target Cloudera Manager version is 7.11.0 or higher and the source cluster version is CDH 6.x or higher before you choose the Select Source Replicate Database option during the HBase replication policy creation process.
- The error *Exception in thread "main" java.lang.IllegalArgumentException: Cannot add a peer with id=_repl__f1907207cdbc4a528777bb4a316_ba23f09f7328494bbcbf81f40b because that id already exists* appears after creating subsequent HBase replication policies.

  This error appears if you created an HBase peer manually using the *hbase shell* to replicate all column families on a source cluster using CDH 5.x.

  You can replicate the HBase data (existing tables and future tables) in a database only if the target Cloudera Manager version is 7.11.0 or higher and the source cluster version is CDH 6.x or higher. Therefore, even if you create a peer using *hbase shell* in the source cluster version CDH 5.16.2 or any other unsupported version, errors might appear.

Appendix

Before you create replication policies, you must register the Amazon S3 or Azure cloud credentials to use as cloud storage in Cloudera Replication Manager, and register the on-premises clusters (CDH or Cloudera Private Cloud Base) as classic clusters in the Management Console.

Support matrix for Cloudera Replication Manager

You can use Replication Manager or other alternate replication methods to replicate HDFS, Hive external tables, and HBase data between on-premises clusters (CDH clusters, Cloudera Private Cloud Base clusters, HDP clusters) and Cloudera Public Cloud (Amazon S3 (AWS), Microsoft Azure ADLS Gen2 (ABFS), and Google Cloud Platform (GCP)) clusters. Replication Manager from HDP clusters to Cloudera Public Cloud Azure is a beta feature and is not available for general use.

Note: Before you create replication policies, you must verify whether the on-premises cluster versions are supported by Replication Manager, register the on-premises clusters as classic clusters in the Cloudera Management Console, register the cloud account credentials in the Replication Manager service, verify cluster access, and configure minimum ports for replication.

See the other sections in this topic for the supported cluster and runtime versions.

- Replication policies and features.
- Cloudera Private Cloud Base and Cloudera Public Cloud source clusters that Replication Manager supports.
- CDH and HDP source clusters that Replication Manager supports.

List of features supported by Cloudera Replication Manager

Replication Manager provides replication policies that you can create, edit, and manage to accomplish your data replication goals. You can use other alternate replication methods for scenarios that Replication Manager does not support. Certain features in Cloudera Replication Manager are available only if the source and target clusters' Cloudera Manager versions support the feature. Verify whether your source and target cluster's Cloudera Manager version support the required feature.

Supported replication policies

You can use the following replication policies in Cloudera Replication Manager:

- HDFS replication policies
- Hive replication policies
- HBase replication policies

HDFS replication policies

Replicate HDFS data and metadata from:

- on-premises clusters (CDH, Cloudera Private Cloud Base, and HDP) to cloud storage.
- cloud storage to classic clusters (CDH or Cloudera Private Cloud Base clusters).

You can choose the frequency during policy creation to replicate the data.

Hive replication policies

Support table-level replication, and can replicate Hive external tables from on-premises clusters (CDH and Cloudera Private Cloud Base) to cloud storage and to Data Hubs. The replication policies can also:

- replicate data stored in Hive tables, Hive metadata, data in Hive metastore, and Impala metadata (catalog server metadata) associated with Impala tables registered in the Hive metastore, and

  Note: Hive2 managed tables are converted to external tables after replication.

- migrate Sentry permissions to Ranger.

  Note: To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.

You can choose the frequency during policy creation to replicate the data.

HBase replication policies

Replicate HBase data from a source classic cluster (CDH or Cloudera Private Cloud Base cluster), COD, or Data Hub to a target Data Hub or COD cluster. You can also copy or replicate HBase data between different environments within a Virtual Private Cloud (VPC) using these policies.

Table 5: Supported cluster and runtime versions for HBase replication policies

Source Cluster Type	Lowest Supported Source CDH/ Cloudera Version	Lowest Supported Source Cloudera Manager Version	Target Cluster Type	Lowest Supported Target Cloudera Version	Lowest Supported Target Cloudera Manager Version
Cloudera Private Cloud Base	7.1.6*	7.3.1	Data Hub in Cloudera Public Cloud AWS/ Azure	7.2.14	7.6.0
CDH	6.3.3	7.3.1	Data Hub in Cloudera Public Cloud AWS/ Azure	7.2.14	7.6.0
CDH	5.16.2	7.4.4 (patch-5017)	COD (AWS)	7.2.14	-
CDH	5.16.2	• 7.6.1 (patch-5610) • 7.6.7 CHF1 and higher	COD (Azure)	7.2.14	-
COD (AWS/ Azure)	7.2.14	-	COD (AWS/ Azure)	7.2.14	-
COD (GCP)	7.2.16.1**	-	COD (GCP)	7.2.16.1**	-
COD (GCP)	7.2.16.500 7.2.17.300 7.2.18.0	-	COD (GCP)	7.2.16.500 7.2.17.300 7.2.18.0	-
Cloudera Private Cloud Base	7.1.9 SP1	7.11.3 CHF7	GCP	7.2.16.1	-

*Cloudera Private Cloud Base 7.1.6 and higher clusters must be Kerberos enabled to use them as source classic clusters in an HBase replication policy.

**You must add key-value pairs to register a Google account to use in Replication Manager. For more information about the key-value pairs, see Preparing to create an HBase replication policy.

Important:

- You can replicate HBase data in SFT-enabled clusters for target clusters with version Cloudera Public Cloud 7.2.16 or higher using HBase replication policies.
- You can replicate Phoenix tables from CDH 5.16.2 and higher using Cloudera Manager 7.4.4 and higher to COD using HBase replication policies.
- You cannot create HBase replication policies if the target Cloudera version is 7.2.16, 7.2.16.1, 7.2.16.2, 7.2.16.3, or 7.2.16.200, the source Cloudera Manager version is 7.7.3 or lower, and the source Cloudera Manager API version is v50 or higher.

  Tip: The endpoint http://*[***CM HOST***]*:*[***CM PORT***]*/api/version shows the API version of the Cloudera Manager.

- HBase replication policies do not support source or destination Cloudera Public Cloud clusters created using the Micro Duty template.

HBase replication policies replicate all the data from the specified tables and then continue to replicate the changed data automatically without user intervention.

Note: Alternate replication methods:

- Use the replication plugin for HBase data. For more information, see COD replication in a Nutshell, Cloudera replication plugin, and HBase data replication.
- Contact Cloudera Support for Hive external tables.

Supported features

The following table lists the features and the Cloudera Manager instances that are required for source clusters and target clusters to use the features:

Feature	Lowest supported source Cloudera Manager version	Lowest supported target Cloudera Manager version
Register the GCP credentials to use in Replication Manager on the **Cloud Credentials** page.	• 7.9.0-h7 and higher • 7.11.0-h3 and higher • 7.12.0.0 and higher	Supports all Cloudera Public Cloud Cloudera Manager versions.
Replicate HBase data simultaneously between multiple clusters*.	• 7.9.0-h7 and higher • 7.11.0-h2 and higher • 7.12.0.0 and higher	• 7.9.0-h7 and higher • 7.11.0-h2 and higher • 7.12.0.0 and higher
Replicate only those HBase tables where the replication scope is already enabled using the Select Source Replicate only tables where replication is already enabled * option during the HBase replication policy creation process.	Supports all Cloudera Public Cloud Cloudera Manager versions.	• 7.9.0-h7 and higher • 7.11.0-h3 and higher • 7.12.0.0 and higher
Specify the network load balancer (NLB) Endpoint after you enable the Select Destination Replicate via a Network Load Balancer* option during the HBase replication policy creation process if the on-premises cluster uses NLB to communicate with the COD clusters.	CDH 5.16.2	7.12.0.100
Specify the YARN queue bandwidth using the Initial Snapshot Settings Maximum Bandwidth * option during the HBase replication policy creation process to export the HBase initial snapshot.	• 7.9.0-h7 and higher • 7.11.0-h3 and higher • 7.12.0.0 and higher	• 7.9.0-h7 and higher • 7.11.0-h3 and higher • 7.12.0.0 and higher
Enter Initial Snapshot Settings Maximum parallel snapshots* to specify the maximum number of tables to process in parallel during the initial snapshot export and import step for an HBase replication policy. If you do not enter any value, Replication Manager chooses an appropriate value, depending on the resources in the source and target cluster, to optimize the performance.	Supports all Cloudera Public Cloud Cloudera Manager versions.	• 7.9.0-h7 and higher • 7.11.0-h3 and higher • 7.12.0.0 and higher
Add IDBroker credentials* to use in Replication Manager on the **Cloud Credentials** page.	7.11.3 CHF7	7.11.3 CHF7
Enter the Select Source Export snapshot user * field during the HBase replication policy creation process to specify the username to export the initial snapshot to the target.	7.11.3 CHF7	7.11.3 CHF7
*To enable this feature, contact your Cloudera Account team.		

Replicate data from Cloudera Private Cloud Base and Cloudera Public Cloud source clusters

Replication Manager replicates HDFS (Cloudera Private Cloud Base source clusters and Cloudera Public Cloud storage on AWS and Azure), Hive external tables (Cloudera Private Cloud Base source clusters), and HBase (Cloudera Private Cloud Base source clusters) data to Cloudera Public Cloud (Amazon S3 and Microsoft Azure

ADLS Gen2 (ABFS)) clusters. You can use the replication plugin as an alternate replication method to replicate HBase data for scenarios that are not supported by Replication Manager.

The following tables list the minimum source and destination cluster versions, minimum Cloudera Manager versions, supported cloud providers, and supported scenarios:

Replicate data from Cloudera Private Cloud Base source clusters

Source cluster	Lowest supported source Cloudera Manager version	Lowest supported source Cloudera Runtime version	Cloud provider	Supported services on Replication Manager	Services that require alternate replication methods
Cloudera Private Cloud Base	7.1.1	7.1.1	Cloudera Public Cloud AWS/Azure	HDFS	HBase To replicate HBase data, see COD replication in a Nutshell and HBase data replication.
Cloudera Private Cloud Base	7.1.1	7.1.1	Data Lake in Cloudera Public Cloud AWS/Azure	Hive external tables	
Cloudera Private Cloud Base	7.9.0	7.1.1	Data Hub in Cloudera Public Cloud AWS/ Azure	Hive external tables	None
Cloudera Private Cloud Base	7.3.1	7.1.6	Data Hub in Cloudera Public Cloud AWS/ Azure	HBase	None
Cloudera Private Cloud Base	7.11.3 CHF7	7.1.9 SP1	Cloudera Public Cloud GCP	HDFS, Hive external tables, HBase	None

Note: Replication Manager converts Hive2 managed tables to external tables after replication.

Replicate data from Cloudera Public Cloud source clusters

Consider the following limitations while using Cloudera Public Cloud source and Cloudera Public Cloud target clusters:

- Replication across cross-cloud providers, that is from AWS to Azure and vice-versa is not supported.
- The source and target clusters must use the same account.

Source cluster	Destination cluster	Supported services on Replication Manager	Services that require alternate replication methods
Cloudera Public Cloud AWS* / Azure	CDH 5.x CDH 6.x HDP 2.x HDP 3.x	Not applicable	HBase To replicate HBase data, see COD replication in a Nutshell and HBase data replication.
Cloudera Public Cloud AWS*	CDH 5.9.0 and higher Cloudera Private Cloud Base 7.1.7 SP1 and higher	HDFS	None
Cloudera Public Cloud Azure	CDH 6.1.0 and higher Cloudera Private Cloud Base 7.1.7 SP1 and higher	HDFS	None
Cloudera Public Cloud GCP 7.2.18 and higher	Cloudera Private Cloud Base 7.1.9 SP1 and higher	HDFS	None
COD version 7.2.14 and higher - Cloudera Public Cloud AWS	AWS	HBase	None
COD version 7.2.14 and higher - Cloudera Public Cloud Azure	Azure	HBase	None

Source cluster	Destination cluster	Supported services on Replication Manager	Services that require alternate replication methods
COD version 7.2.16.1 and higher - Cloudera Public Cloud GCP	GCP	HBase	None
*Replication Manager does not support S3 as a source or destination when S3 is configured to use SSE-KMS.			

Replicate data from CDH and HDP source clusters

Replication Manager replicates HDFS data (CDH source clusters and HDP source clusters), Hive external tables (CDH source clusters), and HBase data (CDH 6 source clusters) to Cloudera Public Cloud (Amazon S3 and Microsoft Azure ADLS Gen2 (ABFS)) clusters. Replication Manager from HDP clusters to Cloudera Public Cloud Azure is a beta feature and is not available for general use. You can use alternate methods to replicate Hive external tables and HBase data for scenarios that are not supported by Replication Manager.

The following tables list the minimum CDH and HDP source cluster versions, minimum Cloudera Manager versions, supported cloud providers, and supported scenarios:

Table 6: Replicate data from CDH 5 source clusters

Source cluster	Lowest supported source Cloudera Runtime version	Lowest supported source Cloudera Manager version	Cloud provider	Supported services on Replication Manager	Services that require alternate replication methods
CDH 5	5.10	6.3.0	Cloud storage in Cloudera Public Cloud AWS	HDFS	HBase To replicate HBase data, see COD replication in a Nutshell, Migrating HBase data, and HBase data replication.
CDH 5	5.10	6.3.0	Data Lake in Cloudera Public Cloud AWS	• Sentry to Ranger* • Hive external tables	
CDH 5	5.10	6.3.4	Cloud storage in Cloudera Public Cloud Azure	HDFS	
CDH 5	5.10	6.3.4	Data Lake in Cloudera Public Cloud Azure	• Sentry to Ranger* • Hive external tables	
CDH 5	5.10	7.9.0	Data Hub in Cloudera Public Cloud AWS/ Azure	Hive external tables	None
*To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.					

Table 7: Replicate data from CDH 6 source clusters

Source cluster	Lowest supported source Cloudera Runtime version	Lowest supported source Cloudera Manager version	Cloud provider	Supported services on Replication Manager	Services that require alternate replication methods
CDH 6	6.1	6.3.0	Cloud storage in Cloudera Public Cloud AWS	HDFS	HBase To replicate HBase data, see COD replication in a Nutshell, Migrating HBase data, and HBase data replication.
CDH 6	6.1	6.3.0	Data Lake in Cloudera Public Cloud AWS	• Sentry to Ranger* • Hive external tables	
CDH 6	6.1	7.1.1 / 6.3.4	Cloud storage in Cloudera Public Cloud Azure	HDFS	

Source cluster	Lowest supported source Cloudera Runtime version	Lowest supported source Cloudera Manager version	Cloud provider	Supported services on Replication Manager	Services that require alternate replication methods
CDH 6	6.1	7.1.1 / 6.3.4	Data Lake in Cloudera Public Cloud Azure	• Sentry to Ranger* • Hive external tables	
CDH 6	6.1	7.9.0	Data Hub in Cloudera Public Cloud AWS/ Azure	• Sentry to Ranger* • Hive external tables	
CDH 6	6.3.3	7.3.1	Data Hub in Cloudera Public Cloud AWS/ Azure	HBase	None
*To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.					

Note: Replication from HDP clusters to Cloudera Public Cloud clusters is a technical preview feature and is not available for general use. Before you replicate data from HDP, contact your Cloudera account team.

Table 8: Replicate data from HDP 2 and HDP 3 source clusters

Lowest supported source HDP version	Cloud provider	Supported services on Replication Manager	Services that require alternate replication methods
HDP 2.6.5*	AWS	HDFS	• HBase To replicate HBase data, see COD replication in a Nutshell and HBase data replication. • Hive external tables For more information, contact Cloudera Support.
HDP 2.6.5*	Azure	HDFS	HBase To replicate HBase data, see COD replication in a Nutshell and HBase data replication.
HDP 3.1.1*	AWS Azure	HDFS	• HBase To replicate HBase data, see COD replication in a Nutshell and HBase data replication. • Hive external tables For more information, contact Cloudera Support.
*No alternate replication methods are available for HDFS, Ranger, and Atlas replication.			

Cloud credentials to use in Cloudera Replication Manager

The Cloud Credentials page shows the registered cloud credentials for Replication Manager. To replicate data to a storage cloud account, you must register the cloud credentials, so that the Replication Manager can access your cloud account. The supported cloud storage accounts are Amazon S3 and Azure Blob Filesystem (ABFS). On the Cloud Credentials page, you can add, update, or delete cloud credentials. Before you register an Amazon S3 cloud account, ensure the cloud bucket requirements are met. Before you add Azure Cloud Credentials in Cloudera Manager, ensure the Blob container requirements are met.

You can perform the following tasks on the Cloud Credentials page to manage cloud credentials:

Add cloud credentials

You can add cloud credentials for your S3 or ABFS account. For information about adding cloud credentials, see Working with Cloud Credentials.

Note: Unregistered credentials can impact the replication process. Credentials associated with a cluster node that do not have updated credentials are called unregistered credentials. For example, if a node is down when the credentials are changed on a bucket or when the node is brought up that has the old credentials.

Update cloud credentials

You can update the cloud credentials based on various factors. When the bucket configuration such as secret or access keys, bucket name or endpoint, and encryption type is changed, it can affect the Replication Manager replication policy run and might require an update to the Replication Manager cloud credentials.

Credential changes are picked up by the next run of the policy. When you change the credentials, the in-progress policy runs might fail but the succeeding runs pick up the changes.

To update a cloud credential, click Actions Update option.

Delete cloud credentials

You can delete unwanted credentials from the Replication Manager. When you delete cloud credentials, the replication policies that use the deleted cloud credentials might fail. To avoid failures, delete the Replication Manager cloud policies associated with the deleted credentials and recreate the policies with the new credentials. You can view a list of policies associated with specific credentials on the **Cloud Credentials** page.

To delete a cloud credential, click Actions Delete option.

Registering Amazon S3 cloud account in Replication Manager

You must have valid Amazon S3 credentials to register the cloud account with Replication Manager.

Before you begin

Consider the following requirements before you register an Amazon S3 cloud account in Replication Manager:

- You need a cloud bucket with user credentials that you can enter in Replication Manager, so Replication Manager can access the bucket.
- The bucket has to have enough space for the replicated data, and write permissions to copy the data.
- The bucket needs to support cloud storage encryption types supported by Replication Manager (SSE-S3 & SSE-KMS).

About this task

When you add cloud credentials for your Amazon S3 account, you can choose one of the following authentication methods:

- Access secret key. To use this authentication type, you require an AWS Access Key and an AWS Secret key that you obtain from Amazon. Cloudera Manager stores these values securely and does not store them in world-readable locations. The credentials are masked and encrypted in the configurations passed to processes managed by Cloudera Manager, and redacted from the logs.

- IAM role. Amazon Identity and Access Management (IAM) can be used to create users, groups, and roles for use with Amazon Web Services, such as EC2 and Amazon S3. IAM role-based access provides the same level of access to all clients that use the role.

  Important: IAM role conditions

  You can choose the IAM role authentication type only when the following conditions are met:

  - The source cluster is hosted on an AWS EC2 infrastructure.
  - The source cluster Cloudera Manager and all the nodes in the cluster are running on an EC2 instance.
  - The source cluster Cloudera Manager has the same IAM role.

  For information about configuring AWS credentials, see Introduction to role based provisioning credential in AWS.

Procedure

1. Go to Replication Manager Cloud Credentials page, and click Add.
2. In the Add Cloud Credential window, perform the following steps:
   a) Select the Cluster.
   b) Select S3 as the Cloud Storage Type.
   c) Name - Provide a unique cloud credential name.
   d) Authentication Type - Select one of the following authentication type:
      - Select the authentication type as Access Secret Key from the drop-down.
        - Access Key - Enter the valid access key.
        - Secret Key - Enter the valid secret key.
      - Select IAM Role if the conditions mentioned in the IAM Role conditions section are met, and click Save.

3. Click Validate.

Note: Using the validation feature is recommended to ensure that the Amazon S3 bucket keys are valid. If the keys are not valid, the Replication Manager policy cannot execute a copy of data to the target Amazon S3 bucket.

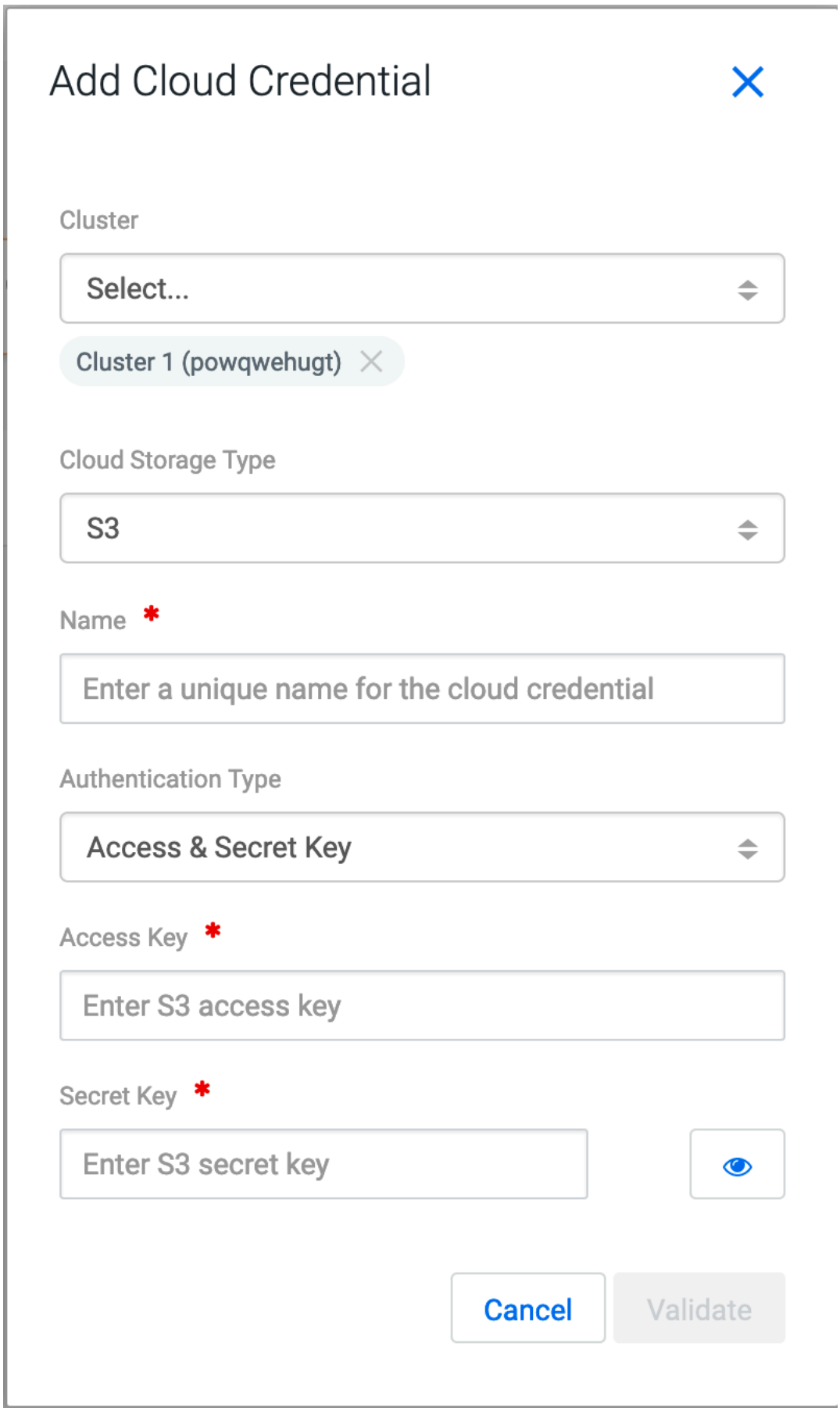
Add Cloud Credential
Cluster
Select...
Cluster 1 (powqwehugt)
Cloud Storage Type
S3
Name *
Enter a unique name for the cloud credential
Authentication Type
Access & Secret Key
Access Key *
Enter S3 access key
Secret Key *
Enter S3 secret key
Cancel
Validate

What to do next

Verify whether the credentials are listed on the Cloud Credentials page.

Register Azure cloud credentials in Replication Manager

You require an ADLS Gen2 storage account that has a cloud Blob container with user credentials to use in the Replication Manager service. The container must have enough space for the replicated data, and write permissions to copy the data.

Currently, registering Azure cloud credentials using the Replication Manager UI does not automatically create the same authorisation rules on the source Cloudera Manager cluster. When you plan to submit the replication policies with Azure as your cloud storage, Cloudera recommends that you update the cloud credentials in the source Cloudera Manager.

Registering ABFS cloud account in Replication Manager

You must have valid ABFS credentials in Cloudera Replication Manager to register the cloud account.

Before you begin

Consider the following requirements before you register an ABFS cloud account in Replication Manager:

- You require an ADLS Gen2 storage account that has a cloud Blob container with user credentials to use in the Replication Manager service.
- The container must have enough space for the replicated data, and write permissions to copy the data.
- The container must support cloud storage encryption types supported by Replication Manager (SSE-S3 & SSE-KMS).

Procedure

1. Click Cloudera Replication Manager Cloud Credentials Add .
2. Perform the following steps on the Add Cloud Credential modal window:
   a) Select the Cluster.
   b) Select ABFS as the Cloud Storage Type.
   c) Enter the cloud credential Name.
   d) Enter your ABFS Storage Client Id.
   e) Enter your ABFS Storage Client Secret Key.
   f) Enter your ABFS Storage Tenant Id.
3. Click Validate.

After you add the ABFS cloud credentials and you create a replication policy with ABFS as your selected cloud storage for your target cluster, the following error message might appear:

To resolve this issue, update the ABFS cloud credential values in the source Cloudera Manager instance. For more information, see *Updating Azure Cloud Credentials in Cloudera Manager*.

What to do next

Verify whether the credentials appear on the **Cloud Credentials** page.

Updating Azure Cloud Credentials in Cloudera Manager

Before you register an ABFS cloud account in Replication Manager, Cloudera recommends that you update the cloud credentials in the source cluster Cloudera Manager. This is because registering Azure cloud credentials using the Replication Manager UI does not automatically create the same authorisation rules on the source Cloudera Manager cluster.

Procedure

1. Click Add AD Service Principal on the Cloudera Manager Administration Azure Credentials page for the source cluster instance.

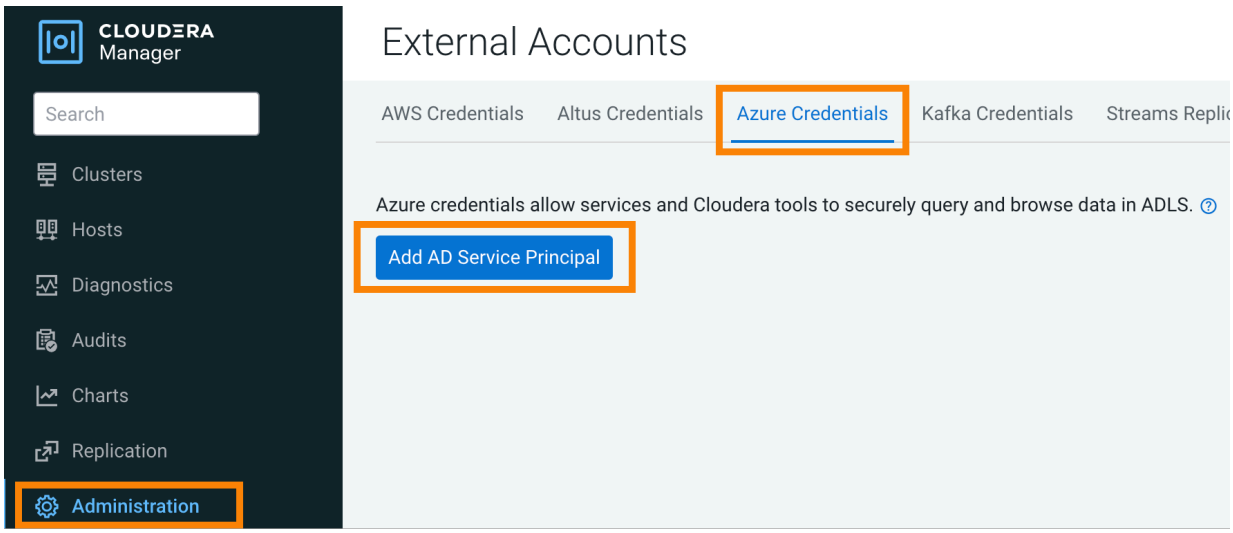

2. Enter the following details on the **Add AD Service Principal** modal window:
   a) Name of the credential.
   b) Client ID for the credential.
   c) Client Secret Key for the credential.
   d) Tenant ID for the credential.

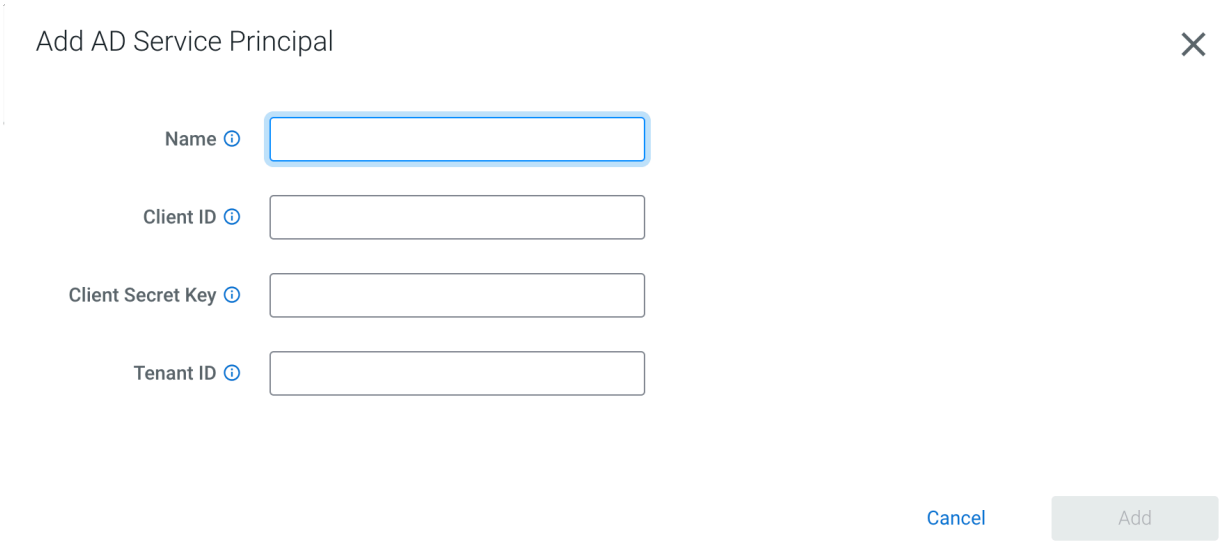

3. Click Add.
   The Azure Cloud credentials are successfully registered with the Cloudera Manager instance.

What to do next

Register the ABFS cloud account in Cloudera Replication Manager.

Registering GCP credentials to use in Replication Manager

You can register Google Cloud Storage (GCS) credentials to use in Replication Manager after you provide access to GCP for the source cluster in the source Cloudera Manager.

Before you begin

Ensure that the following details are available before you register the GCS credentials to use in Replication Manager.

- A valid Google Cloud Platform service account. For more information, see Configuring access to Google Cloud Storage.
- Enough space for the replicated data and the required write permissions to copy the data in the storage of the destination cluster.

Procedure

1. Click Cloud Credentials Add inCloudera Replication Manager.
2. Select the required source Cluster.
3. Choose GCS as the Cloud Storage Type.
4. Enter a unique Name for the cloud credential.

5. Select one of the following Input Type, and then complete the required steps:
   a) Select Manual if you have the following details about the GCP service account:
      - Enter your GCS service account Client Email address.
      - Enter your GCS service account Private Key.
      - Enter your GCS service account Private Key ID.

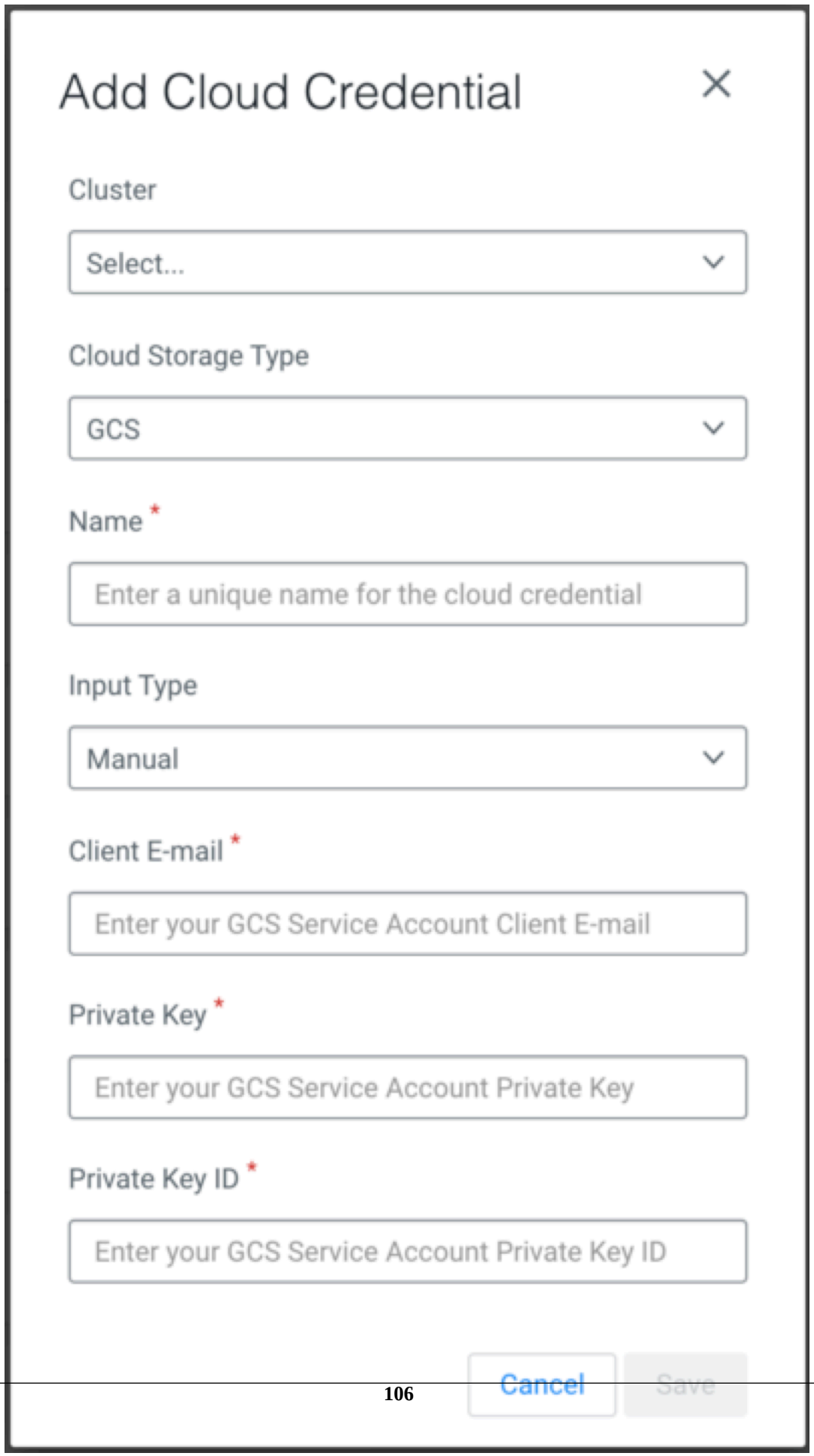
Add Cloud Credential
Cluster
Select...
Cloud Storage Type
GCS
Name *
Enter a unique name for the cloud credential
Input Type
Manual
Client E-mail *
Enter your GCS Service Account Client E-mail
Private Key *
Enter your GCS Service Account Private Key
Private Key ID *
Enter your GCS Service Account Private Key ID
Cancel
Save

b) Select File if you chose to save the service account private key in a file in JSON format. Ensure that you have the required permissions to access and use the file.

Upload File for Replication Manager to use the credentials in it to access data.

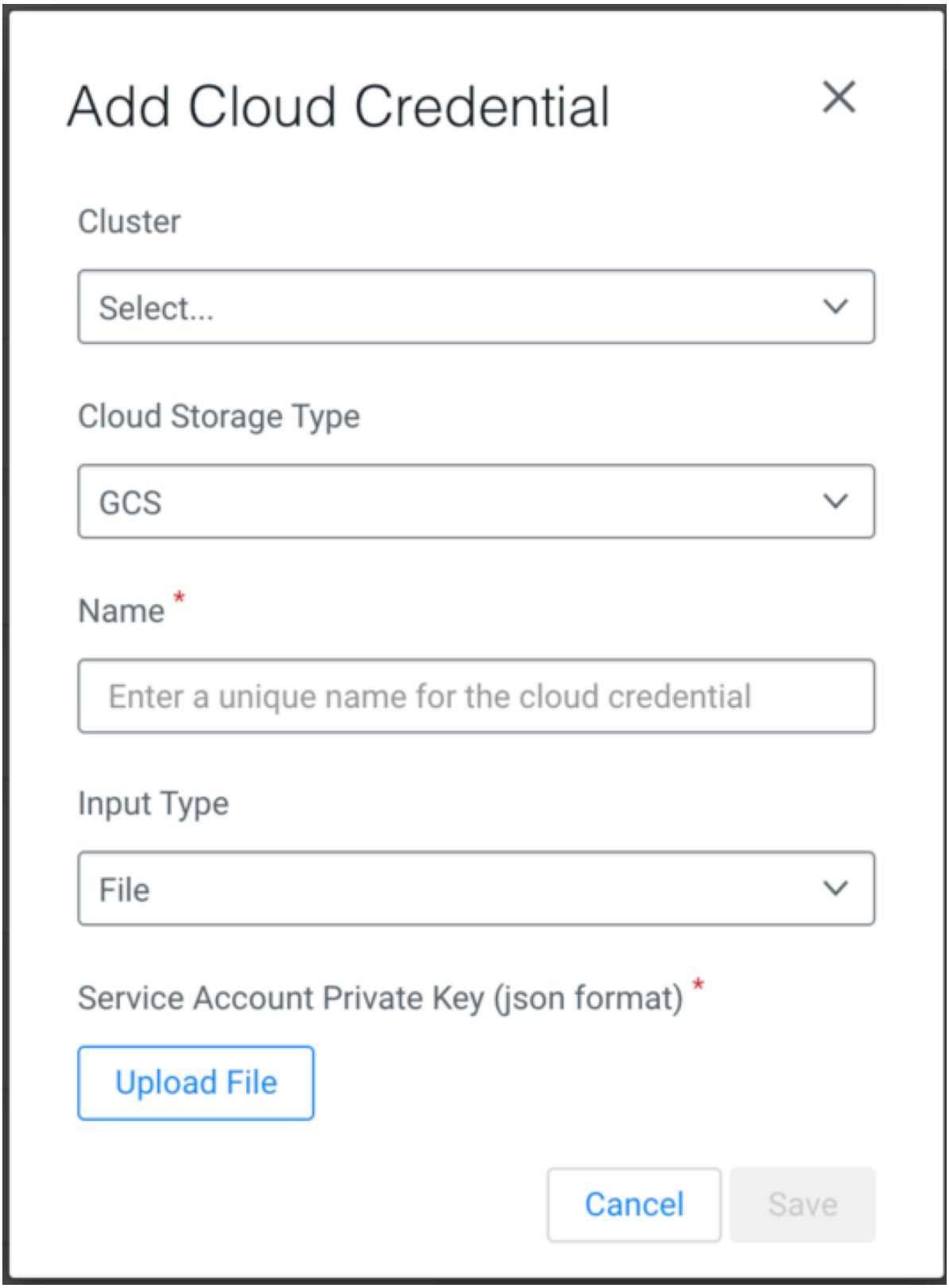

6. Click Save.

Results

The GCP cloud credentials appear on the **Cloud Credentials** page. You can use these credentials when you create the replication policies. The credentials allow Replication Manager to access the cloud data in the source cluster.

Add IDBroker to use temporary AWS session credentials

You can use temporary AWS session credentials through IDBroker to provide just-in-time, minimum required access to replicate data using replication policies. You must complete a few prerequisites before you configure IDBroker to use in replication policies. You can then add the credentials in Cloudera Replication Manager. Alternatively, you can add an external account for the IDBroker topology in Cloudera Manager.

Before you use temporary AWS session credentials in a replication policy, you must:

1. have an AWS account with an IAM role that has the required permissions to access the target S3 bucket and has the necessary trust relationships set up.
2. install a role instance for IDBroker
3. configure non-HA IDBroker on the Cloudera Private Cloud Base cluster.
4. add the cloud credential in Cloudera Replication Manager.

   Alternatively, you can add an external account for the IDBroker topology in Cloudera Manager.

How temporary AWS credentials for replication policies works

Some deployments require temporary AWS session credentials to provide just-in-time, minimum required access to replicate data using replication policies. You can achieve this task using IDBroker. You can use temporary AWS credentials, through the IDBroker service, to replicate HDFS data, Hive external tables, and HBase data from Kerberized Cloudera Private Cloud Base 7.1.9 SP1 clusters or higher using Cloudera Manager 7.11.3 CHF7 or higher versions to S3 buckets using Cloudera Replication Manager.

You can also use the temporary AWS credentials to replicate the HDFS data from S3 buckets to Kerberized Cloudera Private Cloud Base 7.1.9 SP1 clusters or higher using Cloudera Manager 7.11.3 CHF7 or higher versions.

IDBroker is a REST API built as an extension of Apache Knox's authentication services. It allows an authenticated and authorized user to exchange a set of credentials or a token for short-lived cloud vendor access tokens.

To acquire the temporary AWS credentials, you create an IDBroker topology and then map the Kerberos users (or groups) to an AWS IAM Role. During the replication policy run, Replication Manager invokes IDBroker, and the IDBroker then uses the mapping between the on-premises Kerberized user and the IAM Role to request an AWS session token for that role.

Use case

An organization uses the same on-premises cluster across all their departments, and each department has its own AWS account so that it can replicate its required data from the on-premises cluster to its own AWS account when necessary. Each department depending on their requirements might either want to leverage the cloud storage capabilities to store data, or use the cloud processing capabilities to run workloads, analyze the data, or any other purposes.

Authentication methods to use AWS credentials in replication policies

You can choose long-term AWS cloud credentials or temporary AWS session credentials when you want to replicate HDFS data, Hive external tables, and HBase data from Cloudera Private Cloud Base clusters to S3 buckets on Cloudera Public Cloud.

Long-term cloud credentials

You can use long-term credentials to replicate data to the cloud using replication policies. To use long-term cloud credentials in a replication policy, you must:

- have an AWS account, and access key and secret key for it.
- register an external account in Cloudera Manager using AWS access key and AWS secret key.

  You can add an external account on the Cloudera Manager Administration External Accounts page. The external account serves as an authentication method during data replication, using replication policies, from Cloudera Private Cloud Base clusters to cloud.

- add the cloud credential in Cloudera Replication Manager.

The following use cases illustrate scenarios where you can use long-term AWS credentials:

- Environments where you have multiple users and multi-tenancy – In this instance, you can add an **Add Access Key Credentials** external account in Cloudera Manager for Cloudera Private Cloud Base cluster, add the cloud credentials in the Cloudera Replication Manager, and then create a replication policy.
- Single user cluster, or where all the users of the cluster have the same privileges to the data in Amazon S3 – In this instance, you can add **IAM role-based authentication** in Cloudera Manager for Cloudera Private Cloud Base cluster, add the cloud credentials in the Cloudera Replication Manager, and then create a replication policy.

Temporary AWS session credentials

You can use temporary AWS session credentials to provide just-in-time, minimum required access to replicate data using replication policies. Before you use temporary AWS session credentials in a replication policy, you must:

1. have an AWS account with an IAM role that has the required permissions to access the target S3 bucket and has the necessary trust relationships set up.
2. install and configure IDBroker on the Cloudera Private Cloud Base cluster.
3. add the cloud credential in Cloudera Replication Manager.

   Alternatively, you can add an external account for the IDBroker topology in Cloudera Manager.

Adding a role instance to IDBroker in Cloudera Manager

To use IDBroker to access the cloud credentials, you must add a role instance to IDBroker, and then you configure the required properties for it in Cloudera Manager.

Before you begin

You must have an AWS user account that has:

- an IAM policy which allows you to access the S3 bucket to which you want to replicate the data.
- an IAM role, which you can assume, that has the above policy attached to it.

If you have upgraded or installed the Cloudera Private Cloud Base cluster, ensure that IDBroker is available. IDBroker is automatically configured by Cloudera Manager in Cloudera deployments, where Knox is installed. For more information about IDBroker, see Configuring access to S3.

Procedure

1. Go to the source Cloudera Manager Clusters *KNOX SERVICE* Instances page.

   Important: If you are replicating HDFS data from cloud to Cloudera Private Cloud Base cluster, add the role instance on the target Cloudera Manager.

2. Click Actions Add Role Instances .
3. Select the required **Knox IDBroker** host on the **Add Role Instance to Knox** modal window to install it to the required host, and then click Continue.
4. Specify an IDBroker Master Secret, and click Finish.
   The **Instances** tab shows the added role instance.

What to do next

Configure the required properties for IDBroker to use in replication policies.

Configuring IDBroker to use in replication policies

After you add a role instance to install IDBroker, you configure the required properties for it in Cloudera Manager for the Cloudera Private Cloud Base cluster.

Procedure

1. Go to the source Cloudera Manager Clusters *KNOX SERVICE* Instances page.

   Important: If you are replicating HDFS data from cloud to Cloudera Private Cloud Base cluster, add the role instance on the target Cloudera Manager.

2. Click the role instance on the **Instances** tab to open the Knox IDBroker service.
3. Click the **Configuration** tab.
4. Click **Continue Editing Role Instance** if an alert appears.
5. Add the Kerberos username and ARN in the *[***USERNAME***]=[***ARN***]* format in the Knox IDBroker AWS User Mapping property. This maps the Kerberos user to the AWS role.

   If you have multiple usernames, you can map it as shown in the following sample Knox IDBroker AWS User Mapping value. The sample shows Kerberos users *repl*, *hdfs*, and *hbase* mapped to an AWS IAM role:

   repl=arn:aws:iam::134232123254:role/cldr-demo-role;hdfs=arn:aws:iam::134232123254:role/cldr-demo-role;hbase=arn:aws:iam::134232123254:role/cldr-demo-role

   Note: You can map multiple Kerberos users and groups to the same AWS IAM role, but one Kerberos user cannot have multiple AWS IAM roles mapped to it.

6. Add the following in the Kerberos Proxy Block property for HDFS replication policies. This allows the hdfs user to impersonate the Kerberos user during the replication policy run:
   - “hadoop.proxyuser.hdfs.groups”: “*[***KERBEROS USER***]*
   - “hadoop.proxyuser.hdfs.hosts”: “*”
7. Perform the following steps for HBase replication policies:
   a) Add the following details in the Kerberos Proxy Block property. This allows the hbase users to impersonate the Kerberos users during the replication policy run.
      - “hadoop.proxyuser.hbase.groups”: “*[***KERBEROS USER***]*
      - “hadoop.proxyuser.hbase.hosts”: “*”

   b) Add the following details in the Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml in HDFS configuration. This allows the hbase users to impersonate the Kerberos users when exporting the initial snapshot from HDFS.
      - “hadoop.proxyuser.hbase.groups”: “*[***KERBEROS USER***]*
      - “hadoop.proxyuser.hbase.hosts”: “*”
8. Configure the IDBroker Knox Token TTL property to ensure that the configured Knox session token time is greater than the time required to complete a replication policy run, and Save Changes.
9. Use the default aws-cab topology, or create a custom topology, if required, using the Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property. You can also create multiple topologies depending on your use case requirements.

   The following sample code shows a custom topology added to the Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property:

```
<property>
    <name>[***TOPOLOGY1***]</name>
    <value>
         providerConfigRef=cab-providers#IDBROKER:cloud.policy.config.prov
ider=default#IDBROKER:cloud.client.provider=AWS
    </value>
</property>
```

10. Complete the following steps to create the aws.credentials.key and aws.credentials.secret aliases in the topology.
    a) Search for the Save Alias Command Input property:
    b) Enter *[***TOPOLOGY1***]*.aws.credentials.secret=*[***SECRET***]*, and click Save Changes.
    c) Click Actions Save Alias - IDBroker .
    d) Enter *[***TOPOLOGY1***]*.aws.credentials.key=*[***ACCESS KEY***]*, and click Save Changes.
    e) Click Actions Save Alias - IDBroker .
11. Add the following credential details to use the default AWS topology in IDBroker if all the required IAM roles are assumed by a single set of long-term AWS keys. IDBroker uses these credentials to authenticate and to request session tokens from AWS Session Token Service (AWS STS). These credentials are used by IDBroker only to request session tokens and are not used during replication.
    - Knox IDBroker AWS Credentials Key
    - Knox IDBroker AWS Credentials Secret
12. Save the changes.
13. Restart Stale Services, if any.

What to do next

You can add the cloud credential in Cloudera Replication Manager. Alternatively, you can add an external account for the IDBroker topology in Cloudera Manager to use in replication policies.

Adding IDBroker credentials in Cloudera Replication Manager

After you install and configure non-HA IDBroker in Cloudera Manager, you add the cloud credentials in Cloudera Replication Manager. Alternatively, you can create an IDBroker-based external account in Cloudera Manager to use AWS temporary credentials for data replication using replication policies.

Before you begin

Ensure that you have completed the steps to install and configure IDBroker in Adding a role instance to IDBroker in Cloudera Manager on page 109.

Procedure

1. Go to the Cloudera Replication Manager Cloud Credentials page.
2. Click Add.
3. Enter the following details on the **Add Cloud Credential** modal window:

Option	Description
Cluster	Choose the source Cloudera Private Cloud Base cluster where you have configured the IDBroker mapping. **Important:** If you are replicating HDFS data from cloud to Cloudera Private Cloud Base cluster, the IDBroker must be configured on the target Cloudera Manager.
Cloud Storage Type	Choose S3.
Name	Provide a name for the IDBroker topology.
Authentication Type	Choose IDBroker Topology.
IDBroker Address	Enter the IDBroker host and port details in the https://*[***IDBROKER HOST***]*:*[***IDBROKER PORT***]*/gateway format. To identify the required IDBroker address to add in this field, go to the source Cloudera Manager Clusters *KNOX SERVICE* Instances page. The Hostname for the Knox IDBroker Role Type is the IDBroker address. For example, `https://cldrcld-1.cld.root.site:8444/gateway`.

Option	Description
Cloud Topology	Enter the topology name. Enter the topology name you added in the Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property on the source Cloudera Manager Clusters Knox service Configuration tab.

Results
The credentials appear on the **Cloud Credentials** page.

What to do next
You can now use the temporary AWS session credentials through the IDBroker credentials when you create HDFS, Hive, or HBase replication policies in Cloudera Replication Manager.

Adding and managing an IDBroker-based external account in Cloudera Manager
You can create an IDBroker-based external account in Cloudera Manager to use AWS temporary credentials for data replication using replication policies. Ensure that you have configured the required properties for it in Cloudera Manager for the Cloudera Private Cloud Base cluster. Alternatively, you can add the IDBroker credentials in Cloudera Replication Manager.

Procedure

1. Go to the source Cloudera Manager Administration Settings AWS Credentials page.

   Important: If you are replicating HDFS data from cloud to Cloudera Private Cloud Base cluster, add the role instance on the target Cloudera Manager.

2. Complete the following steps to add an IDBroker-based external account in Cloudera Manager:
   a) Click Add IDBroker Topology for Authentication.

      The **Add IDBroker Topology for Authentication** modal window appears.
   b) Provide a Name for the IDBroker topology.
   c) Enter one or more comma-separated list of IDBroker addresses. The IDBroker address includes the IDBroker host and port details in the https://*[***IDBROKER HOST***]*:*[***IDBROKER PORT***]*/gateway format.

      To identify the required IDBroker address to add in this field, go to the source Cloudera Manager Clusters *KNOX SERVICE* Instances page. The Hostname for the Knox IDBroker Role Type is the IDBroker address.

      For example, https://cldrcld-1.cld.root.site:8444/gateway.
   d) Enter the Cloud topology or IDBroker topology to use for cloud connections.

      Tip: Enter the topology name that you added in the source Cloudera Manager Clusters Knox service Configuration Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property.
3. Perform one or more of the following steps to manage an IDBroker-based external account:
   a) Go to the source Cloudera Manager Administration Settings AWS Credentials page.

      Important: If you are replicating HDFS data from cloud to Cloudera Private Cloud Base cluster, add the role instance on the target Cloudera Manager.
   b) Click Actions Edit Credential to edit the credentials for the required **IDBroker Topology**.
   c) Click Actions Remove the credentials to remove the credentials for the required **IDBroker Topology**.
   d) Click Actions Edit Connectivity to open the **Connect to Amazon Web Services** modal window to view more details.

Ports for Cloudera Replication Manager
Before you create replication policies, you must ensure that the required ports are open and available for data replication. You can verify the mandatory ports using the Replication Manager network security diagram.

HDFS replication polices

The following ports must be open and available for Replication Manager for HDFS replication policies:

Table 9: Minimum ports required for HDFS replication policies

Connectivity required	Default Port	Type	Description
Data transfer from classic cluster hosts to cloud storage	80 or 443 (TLS)	Outbound	Outgoing port. All classic cluster nodes must be able to access S3/ADLS Gen2 endpoint. **Note:** Before you create replication policies, ensure that the best practices for Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS) are complete.
Classic cluster	6000-6049 for CCMv1 443 for CCMv2	Outbound	Connecting source classic cluster to the Cloudera Management Console through Cluster Connectivity Manager. For more information, see Outbound network access for Cluster Connectivity Manager, and Cluster Connectivity Manager overview.

The following system architecture diagram shows the interaction between components during HDFS replication using HDFS replication policies:

Figure 7: System architecture diagram for HDFS replication in Cloudera Replication Manager

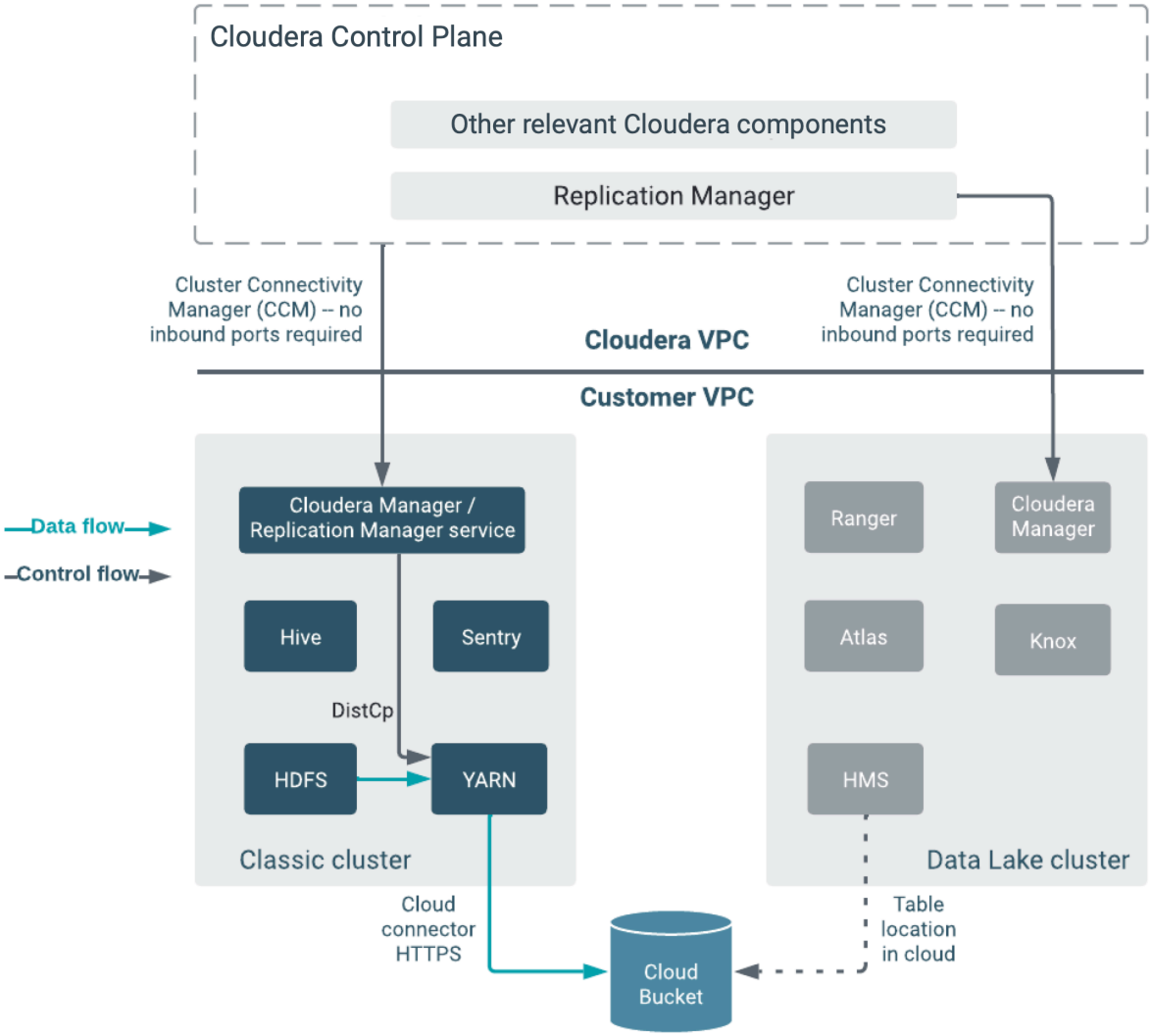

Hive replication polices

The following ports must be open and available for Replication Manager for Hive replication policies:

Table 10: Minimum ports required for Hive replication policies

Connectivity required for	Default Port	Port opened toward	Description
Data transfer from classic cluster hosts to cloud storage	80 or 443 (TLS)	Outbound towards cloud provider public hosts (For example, AWS)	Outgoing port. All classic cluster nodes must be able to access S3/ADLS Gen2 endpoint. **Note:** Before you create replication policies, ensure that the best practices for Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS) are complete.

Connectivity required for	Default Port	Port opened toward	Description
Cloudera Manager Admin Console HTTP	7180 or 7183 (when TLS enabled)	Inbound towards destination Cloudera Public Cloud Cloudera Manager host	Incoming port. Open on the source cluster to enable the target Cloudera Manager in cloud to communicate to the on-premises Cloudera Manager.
Classic cluster	6000-6049 for CCMv1 443 for CCMv2	Outbound towards Cloudera Control Plane (Cluster Connectivity Manager and cluster proxy)	Connecting the source classic cluster to the Cloudera Management Console through Cluster Connectivity Manager For more information, see Outbound network access for Cluster Connectivity Manager, and Cluster Connectivity Manager overview.

The following system architecture diagram shows the interaction between components during Hive replication using Hive replication policies:

Figure 8: System architecture diagram for Hive replication in Cloudera Replication Manager

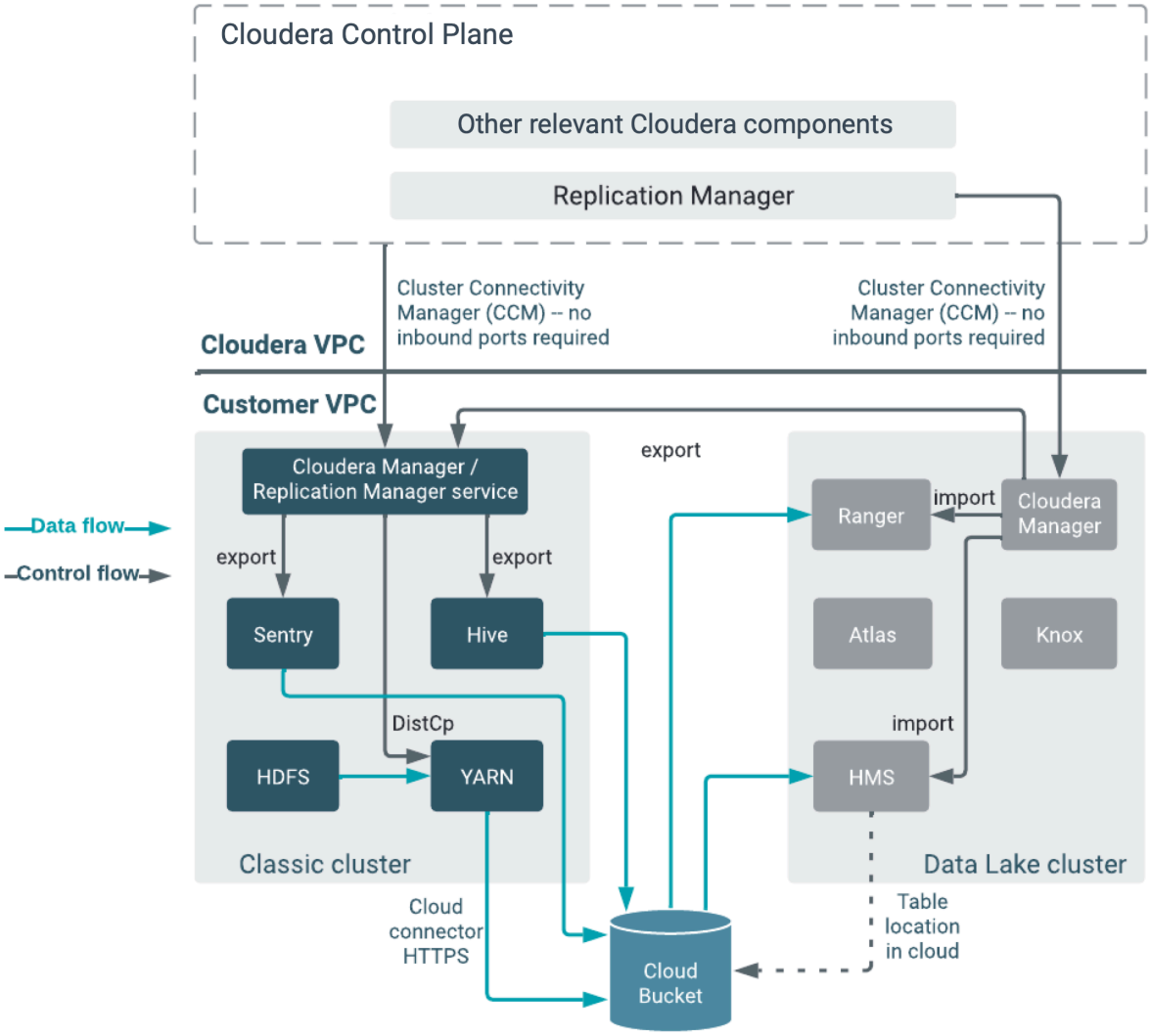

HBase replication policies

The following ports must be open and available for Replication Manager for HBase replication policies:

Table 11: Minimum ports required for HBase replication policies

Service	Ports	Ports opened toward	Description
Destination hosts of the AWS cluster or ADLS cluster (target cluster), and the Cloudera Manager server port on the source cluster	2181 and 16020	Outbound in destination Zookeeper and HBase RegionServer host(s)	Verify whether the ports 16020 for worker security group and 2181 for worker, master, and leader groups are open for connection from the source cluster to the destination cluster on AWS or Azure. This ensures that the source HBase service can communicate with Zookeeper and HBase services on the destination hosts uninterruptedly. For more information, see Ports for HBase replication.
HMaster	16000	Outbound in destination HBase Master host(s)	Open the port on the Master Nodes (HBase Master Node and any back-up HBase Master node). Before you select the Validate Replication option during the first HBase replication policy creation between two specific clusters, you must ensure that the port is open on the target cluster.
Cloudera Manager Admin Console HTTP	7180 or 7183	Inbound towards destination Cloudera Public Cloud Cloudera Manager host	Open on the source cluster to enable Data lake Cloudera Manager to communicate to the on-premises Cloudera Manager. Connects to destination SDX Data Lake Cloudera Manager.
Cluster Connectivity Manager for Cluster Connectivity Manager v1	6000-6049	For Cloudera Control Plane (Cluster Connectivity Manager and Cluster Proxy)	Required for SSL connections to the Cloudera Control Plane via Cluster Connectivity Manager to communicate with Replication Manager.
Data transfer from secondary node for AWS / ADLS Gen2 for Cluster Connectivity Manager v2	80 or 443	Outbound towards data transfer from secondary node for AWS / ADLS Gen2	Outgoing port. Open on all the HDFS nodes for AWS and ADLS Gen2. **Note:** Before you create replication policies, ensure that the best practices for Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS) are complete.
Data Lake cluster	8443	On destination Cloudera Public Cloud Cloudera Manager/Knox host. (applicable when Knox is available on the on-premises source cluster)	Outgoing port. Configure the port on the Data Lake cluster as the outgoing port for Cloudera Management Console to communicate with Cloudera Manager and Knox.

Figure 9: System architecture diagram for HBase replication in Cloudera Replication Manager

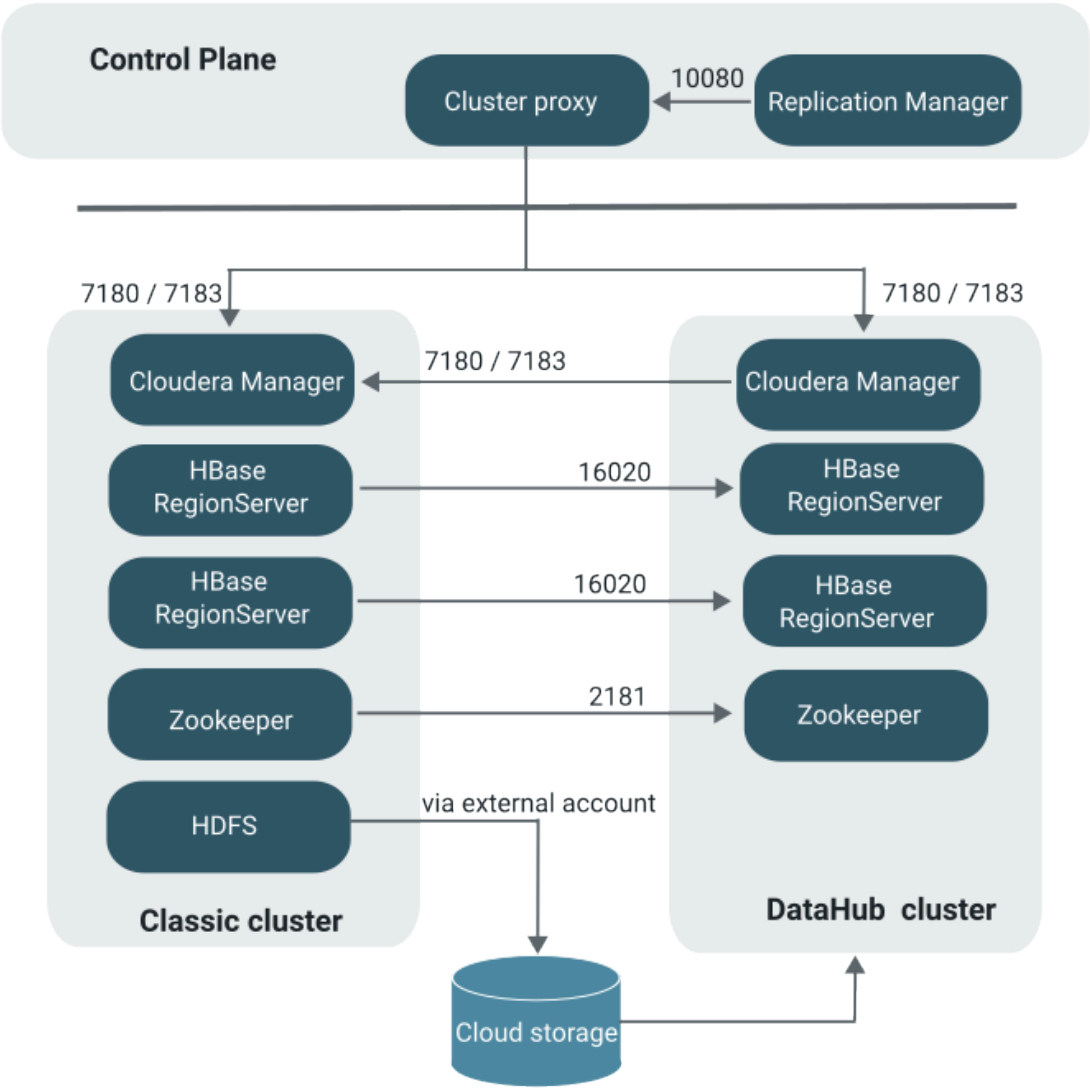

Best practices

Consider the following best practices while using Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS):

- Ensure that the on-premises cluster (port 443) can access the *https://login.microsoftonline.com* endpoint. This is because the Hadoop client in the on-premises cluster (CDH/Cloudera Private Cloud Base) connects to the endpoint to acquire the access tokens before it connects to Azure ADLS storage. For more information, see the *General Azure guidelines* row in the Azure-specific endpoints table.
- Ensure that the steps mentioned in the *General Azure guidelines* and *Azure Data Lake Storage Gen 2* rows in the Azure-specific endpoints table are complete so that the endpoint connects to the target path successfully.

Ports required for HBase replication policies

Open the ports 2181 and 16020 on the source and destination secondary nodes to ensure that the source HBase service can reach Zookeeper and HBase services on the destination hosts.

Use one of the following methods to open the required ports for HBase replication:

- Choose a security group for your environment and open the ports manually. In this method, you choose the security groups that are automatically created for the environment. By default, the security groups do not have any rules for Zookeeper and HBase ports, therefore, you must open the required ports manually after you create a Data Hub.

  After you open the ports, the required security groups are assigned to the nodes when the nodes are autoscaled. This is a one-time process that you must perform when you create a Cloudera Data Hub cluster.
- Define a security group with the required ports open, and assign it to the new Cloudera Data Hub environment. In this method, you define a security group for a VPC that contains inbound rules to open the required ports which include Zookeeper and HBase ports. When you create an environment, you assign this security group to it. If required, you can assign different security groups to the gateway node and other nodes.

  This method allows you to reuse the security groups in other new Cloudera Data Hub clusters. Security issues do not appear because the nodes in the same security group do not access each other by default. However, if required, you can add a separate rule to impose this restriction. Sharing the same security group for inbound and outbound network access rules remains as strict as having separate security groups for each environment, but the extra rules for Zookeeper and HBase ports do not need to be added at each environment creation.

The following use cases illustrate the situations where a requirement for autoscaling nodes during HBase replication might appear:

- You replicate HBase data to another Cloudera account or region in the same cloud provider. In this use case, ensure that the VPC/VNET peering is complete before you open the ports to establish connection over private networks.
- You replicate HBase data to Cloudera Operational Database or Cloudera Data Hub using a direct connection. In this use case, you ensure that public IPs and Zookeeper ports are not open to the internet.

..

Self-service Exploratory Analytics

Date published: 2021-07-15
Date modified:

CLOUDERA

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Self-service exploratory analytics.....4
Understanding the use case.....5
Self-service exploratory analytics use case.....6
Self-service exploratory analytics steps.....8
Meeting the prerequisites.....8
Downloading sample data.....9
Exploring and querying data in Hue.....11
Generating and sharing dashboards.....17
Troubleshooting.....19
Unable to see the S3 File Browser on the Hue web interface.....19
External File option is unavailable.....19
Error while accessing a directory in S3.....20
Incorrect region error while uploading a file to S3.....20

Self-service exploratory analytics

The self-service exploratory analytics pattern provides exact steps that you can follow to quickly explore, query, and analyze data present in CDP Data Lakes, public cloud storage, or on your computers in a file, and generate visual dashboards—all using Cloudera Data Warehouse.

Figure 1: Self-service exploratory analytics in a nutshell

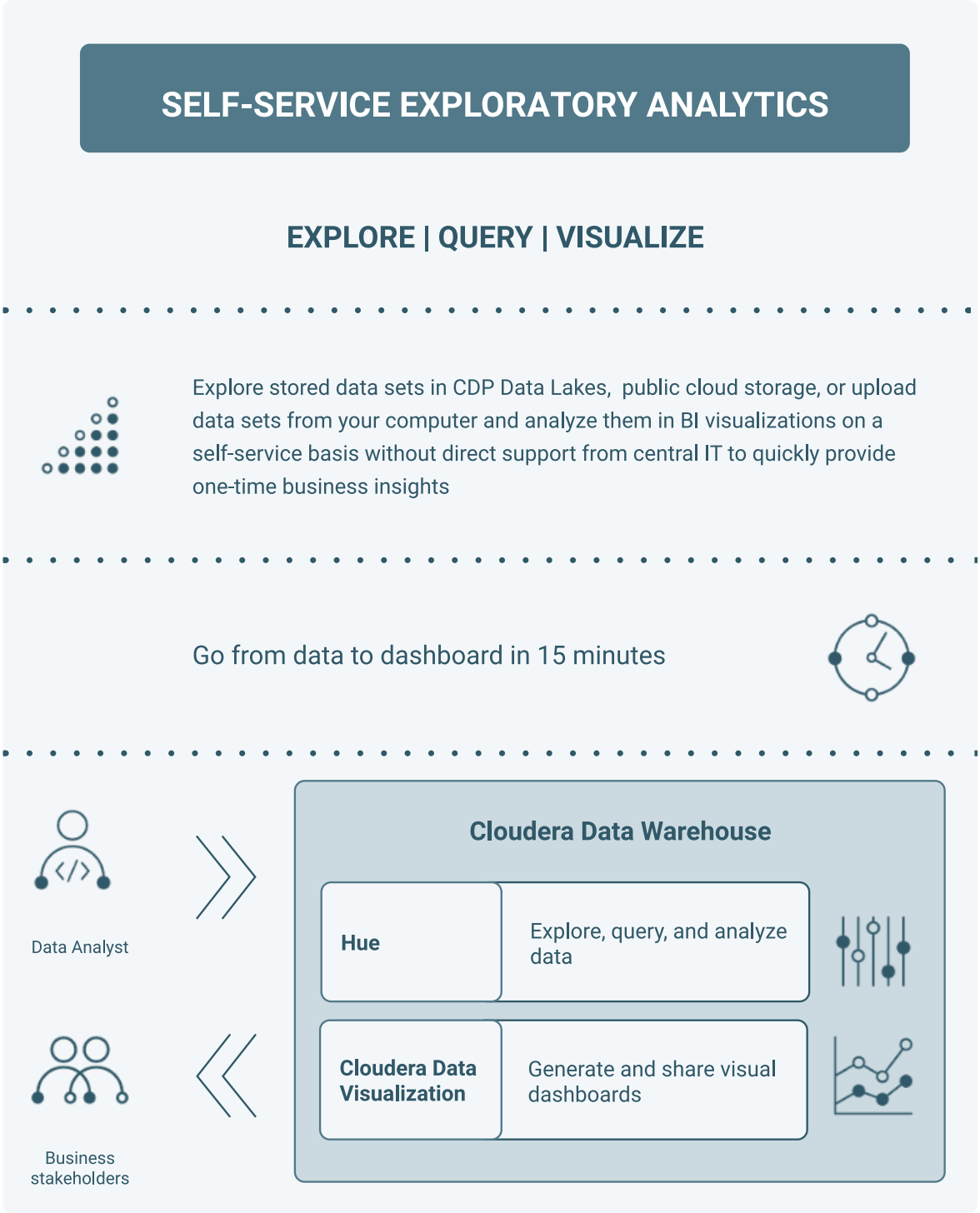

Understanding the use case

Learn about the self-service exploratory analytics pattern.

In modern data-driven businesses, Line of Business (LoB) Data Analysts are often tasked with exploring and analyzing new data sets and providing business insights on the fly, without waiting for the curated and prepared data or depending on the central IT.

Today, it is hard for LoB Data Analysts, and the Data Engineers that support them, to respond in a timely manner to LoB's needs to update existing dashboards and reports for new insights or to create new reports to respond to an emergent need. Often, these requests need integration of various data and application silos that require high levels of support from central IT services to make resources available, to set up ingest and curation processes to bring usable data to the report or dashboard consumer or to safely import and process the data available outside the current infrastructure (flat files on a user computer). This process is time-consuming, expensive, and often quite rigid. As a result, many LoB decisions are delayed, costing businesses opportunities, revenue, and customers.

With Cloudera Data Warehouse (CDW), LoB Data Analysts and Business Analysts can explore data sets stored in Cloudera Data Platform (CDP) Data Lakes, cloud object stores such as Amazon S3, or uploaded data sets from their computers and analyze them in new or existing BI visualizations on a self-service basis without direct support from central IT resources, as shown in the following diagram:

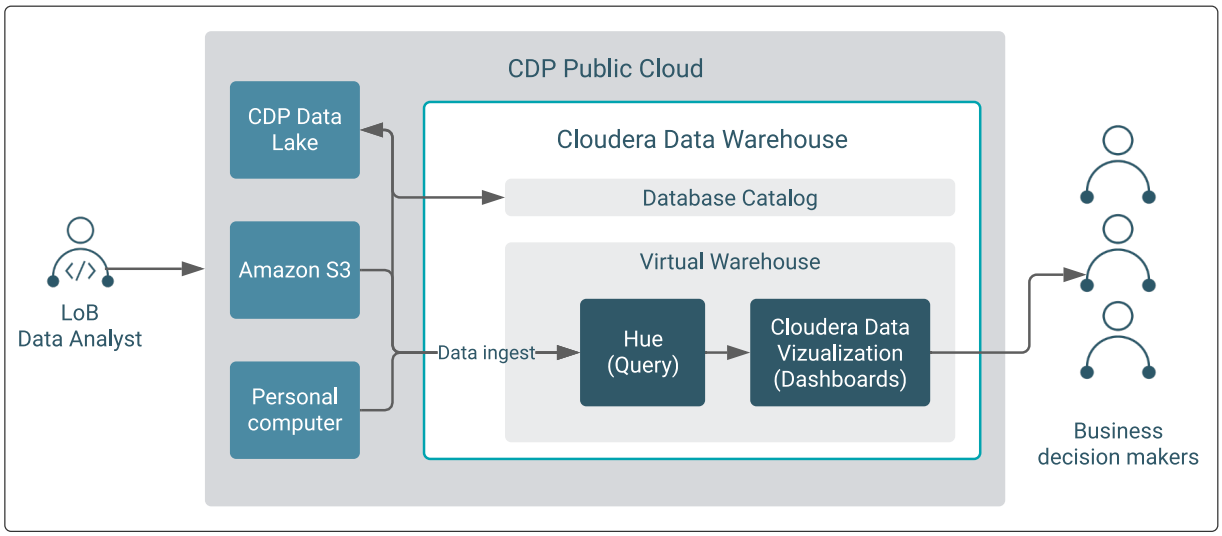

CDW enables you to go from data to dashboard in 15 minutes.

Self-service exploratory analytics use case

A real-life business narrative enables you to follow and understand the Self-service exploratory analytics CDP pattern. The narrative introduces the key personas at play, establishes burning business questions, and sets parameters for the success criteria of the pattern.

This is a sample use case. The steps in this pattern can be applied to similar use cases across other industries.

The use case: Sales promotion at duty-free stores for airline customers with longer layovers or flight delays

You are a marketing data aggregator (Special Marketing Group (SMG)) company, helping a group of duty-free shops in airports to answer some questions around a new marketing campaign they plan to run. The duty-free shop owners want to partner with specific airlines to offer discounts through the airlines to their passengers who spend more time in an international terminal than warranted for their flight. In this partnership, the airlines can improve customer satisfaction during extended wait times, and duty-free shops can increase sales.

A representative of a number of duty-free shops has requested help to select airlines to partner with. The shops will offer discounts to passengers of those airlines experiencing delays.

To decide which airlines to partner with, the marketing managers from the duty-free shops need to know which airlines have the most customers who experience delays and base the size of the discount on a range of dissatisfaction levels.

This short-lived sales promotion does not warrant a permanent dashboard or report, and you do not expect more than 100 people to access the data resulting from this pattern.

Key personas of this use case

The primary personas have access to the data and perform analytics on it. The secondary personas draft sales promotion policies from business insights.

Primary personas:

- Data Analyst: The Data Analyst from the SMG company is responsible for providing the duty-free shop owners with a visual tool to choose airline partners for offering duty-free discounts.
- LoB Marketing Representative: The LoB Marketing Representative for the duty-free shops is globally responsible for supporting a number of duty-free shop owners in sales and marketing activities, and requests decision support services from the SMG company.

Secondary personas:

- Marketing Data Analysts: The Marketing Data Analysts from each duty-free organization would want to access the dashboards for insight. The assumption is that there are no more than 10 duty-free shop management firms with no more than 10 analysts interested in evaluating data that is published based on this request.
- LoB Managers: Managers from each duty-free organization use the insight and draft the promotional policies.

Criteria and conditions for the sales promotion

The LoB Marketing Representative has identified ideal airline partners as ones that have the most customers that experience the following inconveniences:

- Long layovers before an international flight
- Elongated layovers caused by delayed international flights (connection)
- Missed international connections due to a delay on the pre-connection flights

The LoB Marketing Representative wants to offer a discount program in partnership with their ideal airlines, which partially subsidize the discount percentage based on passenger inconveniences as follows:

- 10% discount at the duty-free shop for passengers with long layovers not caused by delays. A long layover is any layover greater than 90 minutes.
- 20% discount for passengers that have an elongated layover due to a delay in their next leg. An elongated delay is a delay of the departing flight greater than 60 minutes.
- 30% discount for passengers displaced by missing their connecting flight due to a delay in their previous leg. A missed connection is a delay of 45 minutes, or more, between arrival and the departure of the connecting international flight. The minimum time between flights to safely make a connection is 45 minutes.

Burning business questions

The marketing data coordinator from the SMG company has data about the airlines, flights, airports, and passengers in a number of places. The SMG Data Analyst needs to address the following business questions using data they have aggregated from airlines and airports tables:

- Which airlines have the most passengers who have long layovers built into their tickets?
- Which airlines have the most passengers who have delayed international legs in their itinerary causing an extended layover in their flight itinerary?
- Which airlines have the most passengers who are displaced by a missed international connection, caused by a delay in their previous leg?

Success criteria

Going from data to dashboard in 15 minutes.

The SMG Data Analyst is able to explore and analyze data using Hue and can share a dashboard in CDP Data Visualization with the Marketing Data Analysts and LoB Managers. The Marketing Data Analysts and LoB Managers should be able to use this dashboard and the insights to frame the promotional policies for their duty-free outlets.

Self-service exploratory analytics steps

For this CDP pattern, you must be onboard with CDP Public Cloud and your IT team must have set up the Cloudera Data Warehouse service with other necessary prerequisites listed in the *Meeting the prerequisites* section. As a Data Analyst or a Line of Business user, you can then explore and query data, import new or missing data, create data sets for the business questions that you are trying to answer, and generate visual dashboards to share with your stakeholders.

Meeting the prerequisites

Before the Data Analysts can explore and query data, your central or departmental IT must have onboarded to CDP Public Cloud and must meet the requirements listed in this section.

To do in the CDP Management Console

1. Make sure that the Cloudera Data Lake is created and running. For more information, see Creating an AWS environment with a medium duty data lake using the CLI.
2. Grant EnvironmentUser role to the Data Analyst user and synchronize the user to FreeIPA. For more information, see Assigning account roles to users and Performing user sync.

To do in Cloudera Data Warehouse (CDW)

1. Make sure that the CDW environment is activated. For more information, see Activating AWS environments.
2. Add an S3 bucket as an external bucket in the CDW environment with read-only access.

   Important: You must add your own S3 bucket to the CDW environment.

   a. Go to CDW service Environments and click its edit icon.
   b. On the Environment Details page, type the name of the AWS bucket you want to configure access to in the Add External S3 Bucket text box and select read-only access mode.
   c. Click Add Bucket and then click Apply to update the CDW environment.
3. Create a non-default Database Catalog with the Load Demo Data option enabled.

   a. Go to CDW service Database Catalogs and click Add New.
   b. Specify a name, select an environment, select SDX from the Datalake drop-down list and enable the Load Demo Data option, and click CREATE.
4. Create an Impala-based Virtual Warehouse with the Enable Data Visualization option and make sure it is in the running state.

   a. Go to CDW service Virtual Warehouses and click Add New.
   b. Select the type as IMPALA, select the Database Catalog associated with the Virtual Warehouse, select the size, click Enable Data Visualization, and then click CREATE.

5. Enable the S3 File Browser for Hue.
   a. Go to CDW service Virtual Warehouses , select your Virtual Warehouse and click edit.
   b. On the **Virtual Warehouses** page, go to CONFIGURATIONS Hue and select hue-safety-valve from the drop-down list.
   c. Add the following configuration for Hive or Impala Virtual Warehouse in the space provided and click APPLY:

```
[desktop]
# Remove the file browser from the blocked list of apps.
# Tweak the app_blacklist property to suit your app configuration.
app_blacklist=spark,zookeeper,hive,hbase,search,oozie,jobsub,pig,sqoop
,security

[aws]
[[aws_accounts]]
[[[default]]]
access_key_id=[***AWS-ACCESS-KEY***]
secret_access_key=[***SECRET-ACCESS-KEY***]
region=[***AWS-REGION***]
```

6. Enable a link to the Data VIZ application (Cloudera Data Visualization) from the Hue web interface or provide the Data VIZ application URL to the Data Analyst.
   a. Go to CDW service Virtual Warehouses , select your Virtual Warehouse and click edit.
   b. On the Virtual Warehouses details page, go to CONFIGURATIONS Hue and select hue-safety-valve from the drop-down list.
   c. Add the following lines in the safety valve and click APPLY:

```
[desktop]
  custom_dashboard_url=[***DATA-VIZ-URL***]
```

To do in Ranger

Grant the required DDL and DML Hadoop SQL policies to the Data Analyst user in Ranger.

1. Go to CDW service Database Catalogs , click the more option, and click Open Ranger.
2. On the **Ranger Service Manager** page, click Hadoop SQL.
3. Select the all - url policy.

   The **Edit Policy** page is displayed.
4. Under the Add Conditions section, add the users under the Select User column and add permissions such as Create, Alter, Drop, Select, and so on from the Permissions column.
5. Scroll to the bottom of the page and click Save.

Other prerequisites

- Data in a file to be ingested is in a structured format, such as CSV, with headers.
- The data file is available to the Data Analyst on their computer or accessible on a shared drive with permissions already granted.
- Hue application URL is shared with the Data Analyst, if a link to the Data Visualization application is not enabled in Hue.

Downloading sample data

The airline's data is already available in CDW (Hue), but the passenger data is not. Download the sample data for this exercise provided with this pattern.

Airlines data (available in CDP)

The airline data consists of the following four tables:

- airlines
- airports
- flights
- planes

The airline data looks as follows:

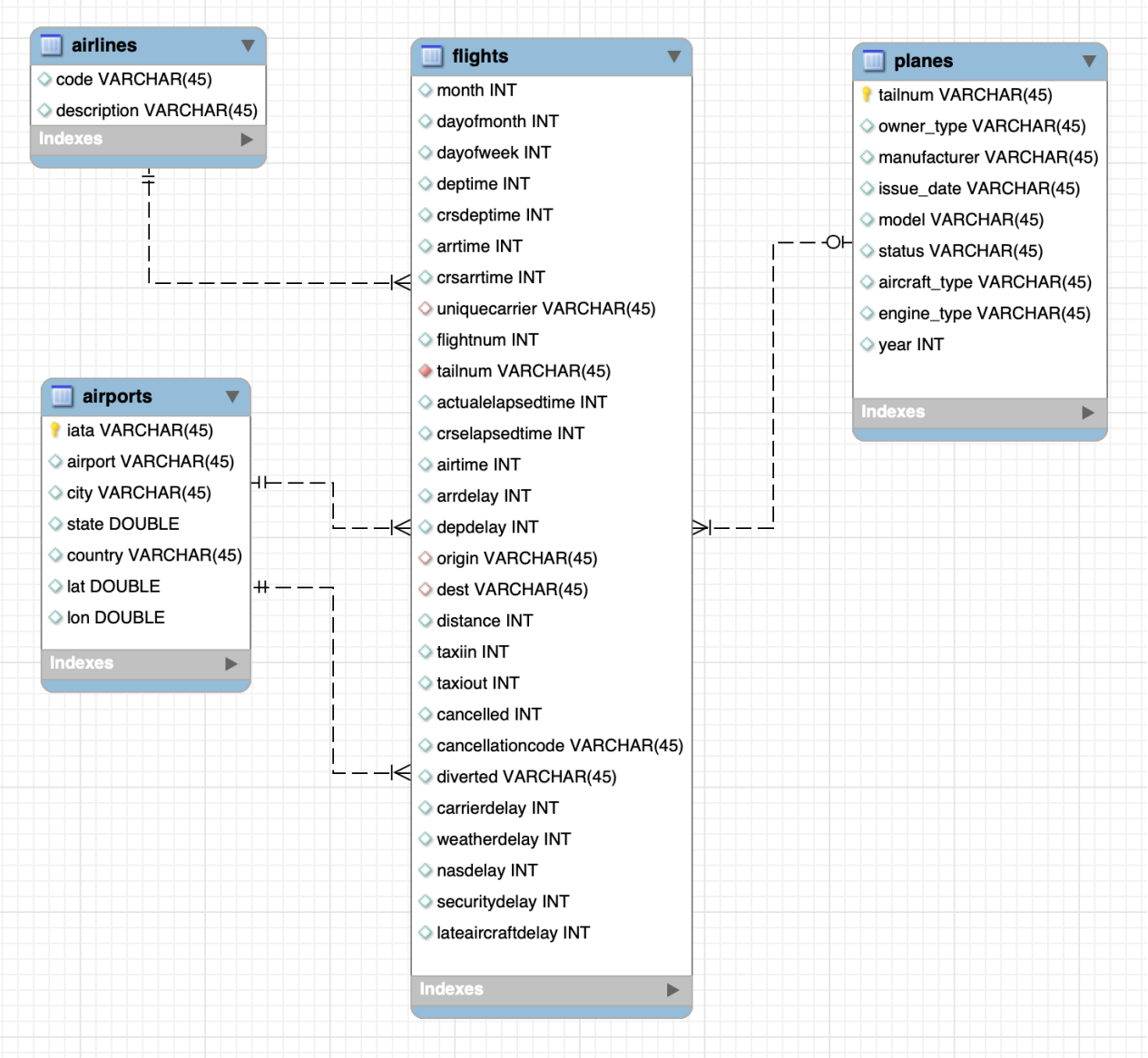

Passenger data (to be downloaded)

The passenger data is present in the following CSV file: passengertickets.csv. You will create a “unique_tickets” table from the CSV file by importing it in Hue.

SQL queries for this exercise (to be downloaded)

The following file contains the SQL queries that you will use for this exercise: sample-queries.txt

Exploring and querying data in Hue

As an SMG Data Analyst, you follow logical steps to explore and query the airline data to create data sets and then use those data sets to draw insight and create dashboards.

Exploring the databases and airline tables

Use Hue to look up existing databases, tables, and data that is available in your data warehouse.

Procedure

1. Log in to Hue with your credentials.
2. Explore the databases and airline tables from the left-assist panel.
3. Run the following query to get a list of international flights:

```
-- Query to find all international flights: flights where destination ai
rport country is not the same as origin airport country
SELECT DISTINCT
   flightnum,
   uniquecarrier,
   origin,
   dest,
   month,
   dayofmonth,
   `dayofweek`
FROM
   `airline_ontime_orc`.flights f,
   `airline_ontime_orc`.airports oa,
   `airline_ontime_orc`.airports da
WHERE
   f.origin = oa.iata
   and f.dest = da.iata
   And oa.country <> da.country
ORDER BY
   month ASC,
   dayofmonth ASC
;
```

Importing the passenger data

Use Hue to import data which is not present in your data warehouse, but is available on your computer or cloud storage such as Amazon S3.

About this task

In this exercise, the passenger data is not already available in Hue or in CDP but instead is available on your computer in a CSV file. Import the passenger data into Hue from your computer as follows:

Procedure

1. Go to Importer and click .. at the end of the Path field.

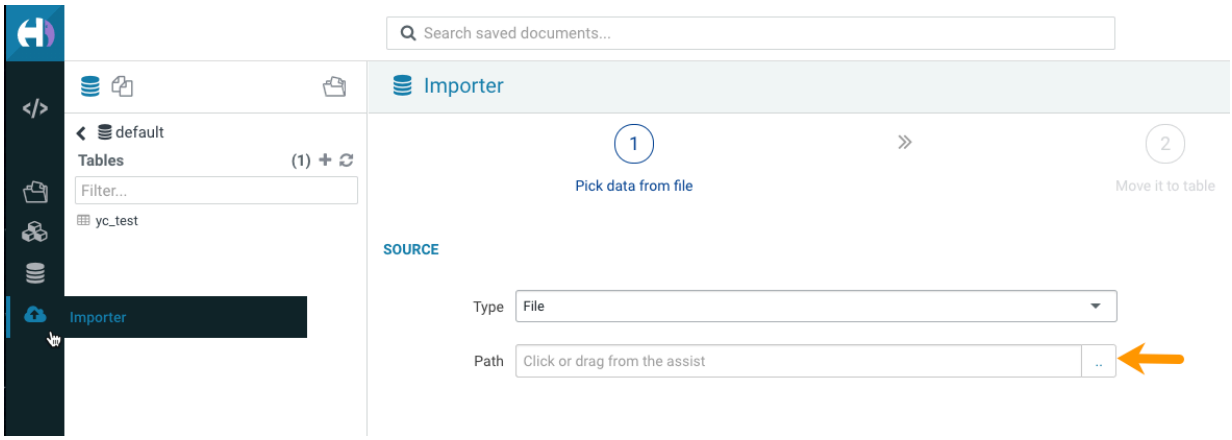

A pop-up window is dispalyed where you can choose a file.

2. Type s3a:// in the address text box and press enter.

   The S3 buckets associated with the CDW environment are displayed. You can narrow down the list of results using the search option.

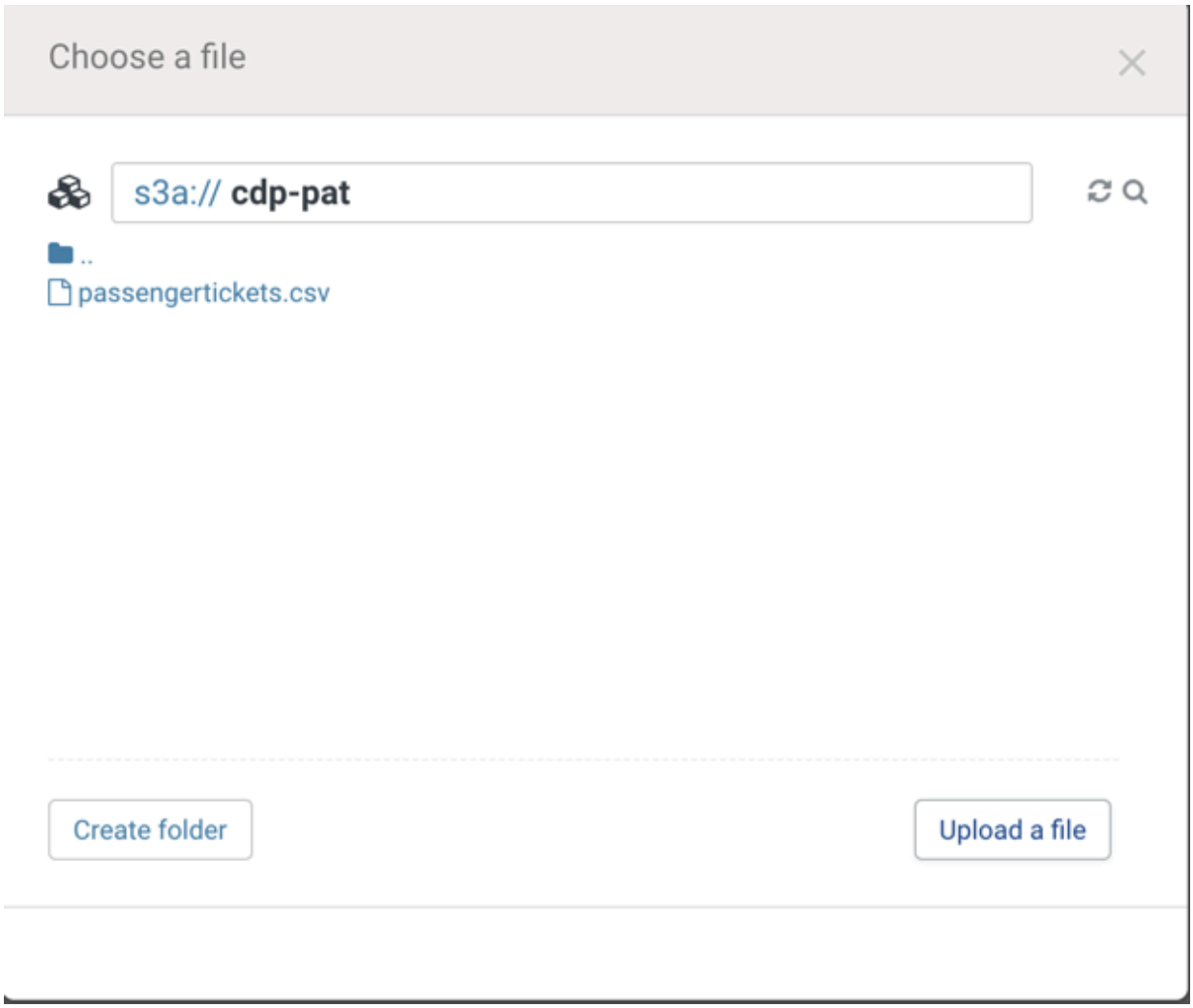

3. Navigate to the bucket in which you want to upload the file and click Upload a file.

4. Select the CSV file that you want to import into Hue.

   Hue displays the preview of the table along with the format.

   Hue automatically detects the field separator, record separator, and the quote character from the CSV file. If you want to override a specific setting, select a different value from the drop-down list.

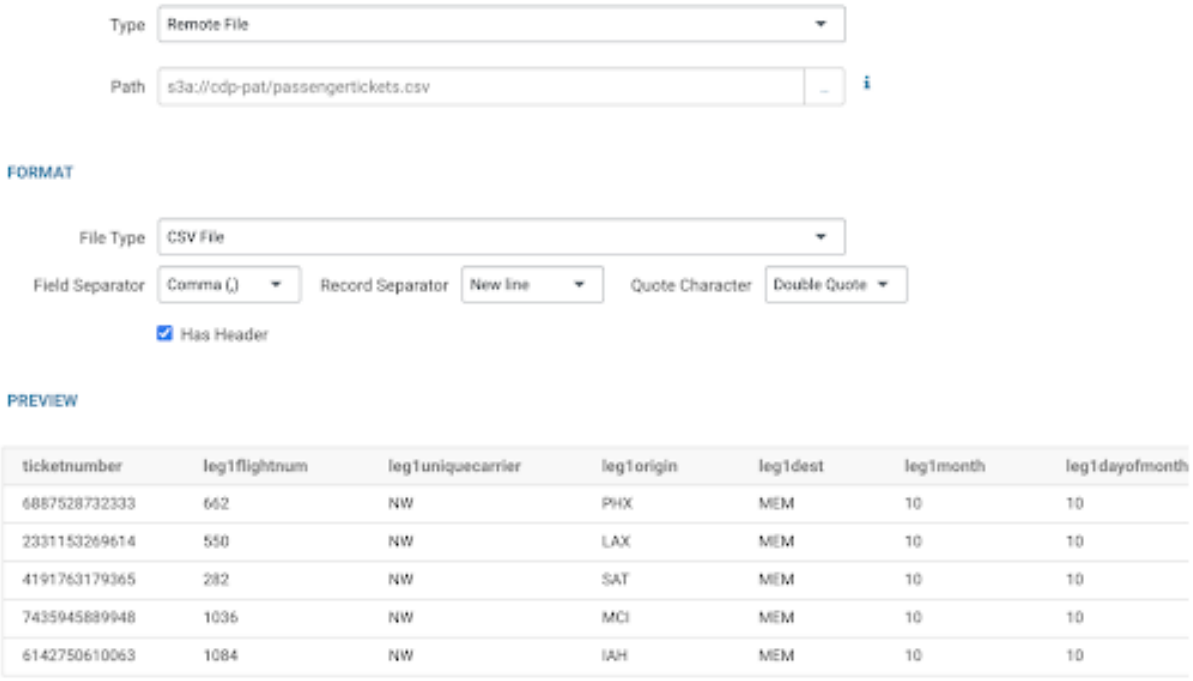

ticketnumber	leg1flightnum	leg1uniquecarrier	leg1origin	leg1dest	leg1month	leg1dayofmonth
6887528732333	662	NW	PHX	MEM	10	10
2331153269614	550	NW	LAX	MEM	10	10
4191763179365	282	NW	SAT	MEM	10	10
7435945889948	1036	NW	MCI	MEM	10	10
6142750610063	1084	NW	IAH	MEM	10	10

5. Click Next.

   On this page under DESTINATION, enter airline_ontime_orc.unique_tickets in the Name field.

   The unique_tickets table is created in the airline_ontime_orc database.

6. Select Format as Text.

7. Expand Extras and select the Store in Default location option.

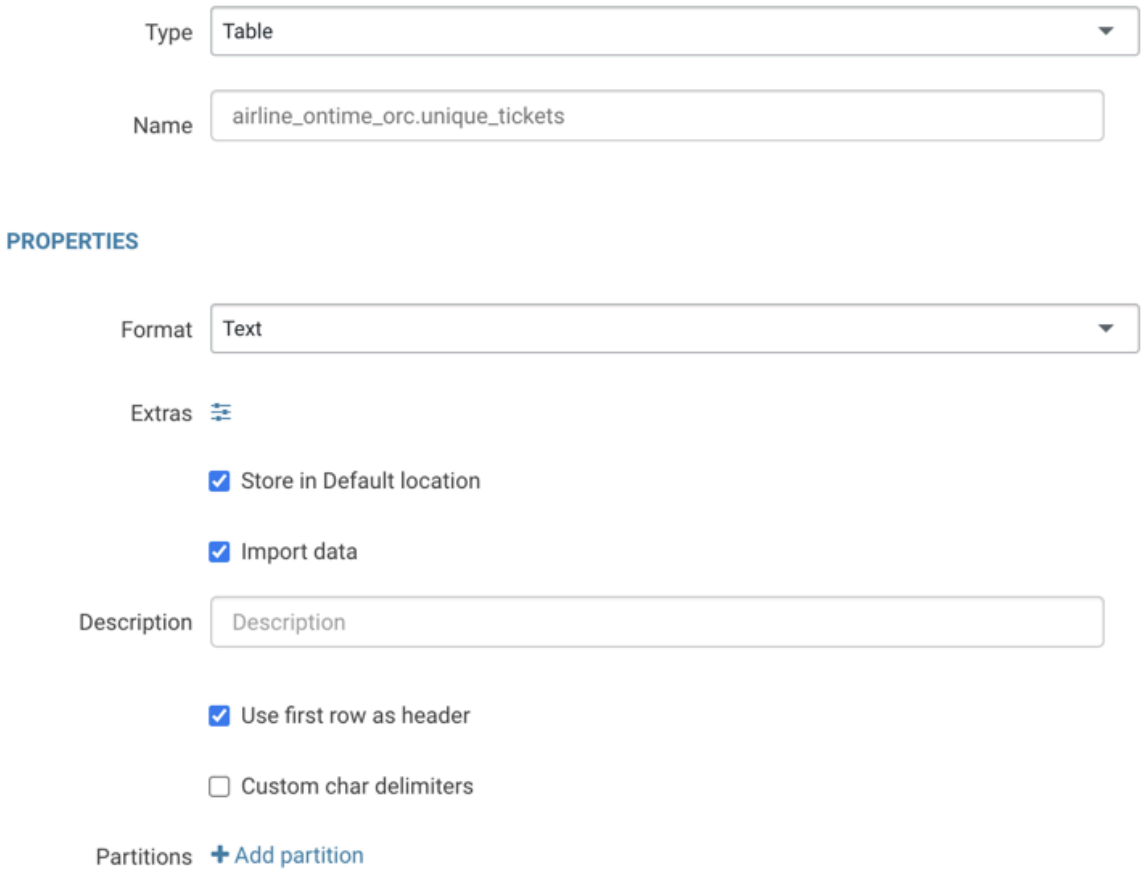

8. Change all the “bigint” column types to “int” under FIELDS and click Submit.
   The CREATE TABLE query is triggered.

   Hue displays the logs and opens the **Table Browser** from which you can view the newly created table when the operation completes successfully.

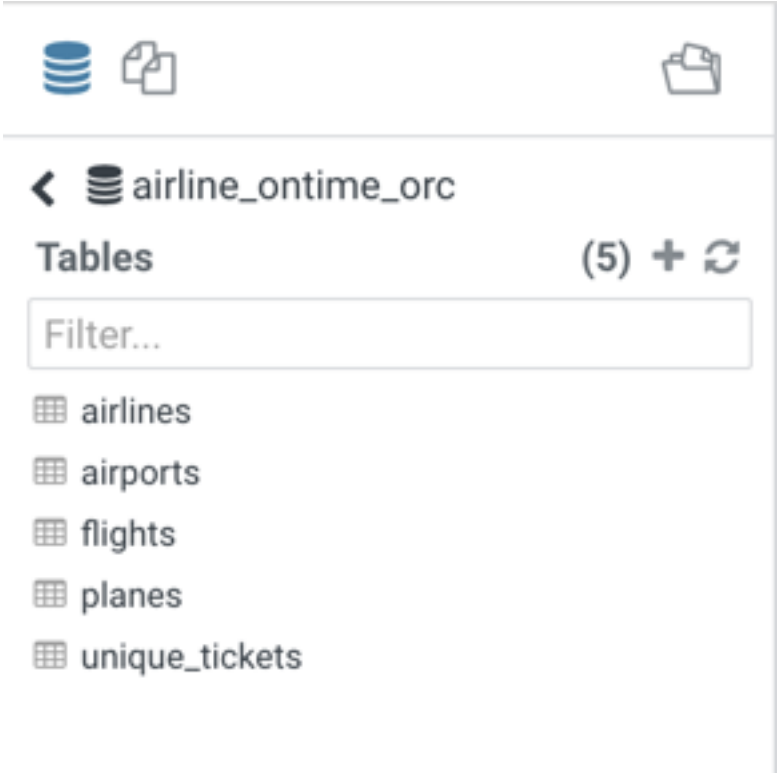

9. Run the following query to find passengers with international connecting flights:

```
-- Query to explore passenger manifest data:  do we have international c
onnecting flights?
SELECT
*
FROM
   `airline_ontime_orc`.unique_tickets a,
   `airline_ontime_orc`.flights o,
   `airline_ontime_orc`.flights d,
   `airline_ontime_orc`.airports oa,
   `airline_ontime_orc`.airports da
WHERE
   a.leg1flightnum = o.flightnum
   AND a.leg1uniquecarrier = o.uniquecarrier
   AND a.leg1origin = o.origin
   AND a.leg1dest = o.dest
   AND a.leg1month = o.month
```

```
   AND a.leg1dayofmonth = o.dayofmonth
   AND a.leg1dayofweek = o.`dayofweek`
   AND a.leg2flightnum = d.flightnum
   AND a.leg2uniquecarrier = d.uniquecarrier
   AND a.leg2origin = d.origin
   AND a.leg2dest = d.dest
   AND a.leg2month = d.month
   AND a.leg2dayofmonth = d.dayofmonth
   AND a.leg2dayofweek = d.`dayofweek`
   AND d.origin = oa.iata
   AND d.dest = da.iata
   AND oa.country <> da.country
;
```

Creating data sets as per the business criteria

Run queries to create data sets that help answer burning business questions.

Procedure

1. Run the following query to get the number of passengers on an airline with long layovers:

```
-- Number of passengers on the airline that have long, planned layovers
for an international flight
SELECT
   a.leg1uniquecarrier as carrier,
   count(a.leg1uniquecarrier) as passengers
FROM
   `airline_ontime_orc`.unique_tickets a
where
   a.leg2deptime - a.leg1arrtime>90
group by
   a.leg1uniquecarrier
;
```

2. Run the following query to get the number of passengers on an airline with delayed international leg:

```
-- Number of passengers on airlines that have elongated layovers for an
international flight caused by delayed connection
SELECT
   a.leg1uniquecarrier as carrier,
   count(a.leg1uniquecarrier) as passengers
FROM
   `airline_ontime_orc`.unique_tickets a,
   `airline_ontime_orc`.flights o,
   `airline_ontime_orc`.flights d
where
       a.leg1flightnum = o.flightnum
   AND a.leg1uniquecarrier = o.uniquecarrier
   AND a.leg1origin = o.origin
   AND a.leg1dest = o.dest
   AND a.leg1month = o.month
   AND a.leg1dayofmonth = o.dayofmonth
   AND a.leg1dayofweek = o.`dayofweek`
   AND a.leg2flightnum = d.flightnum
   AND a.leg2uniquecarrier = d.uniquecarrier
   AND a.leg2origin = d.origin
   AND a.leg2dest = d.dest
   AND a.leg2month = d.month
   AND a.leg2dayofmonth = d.dayofmonth
   AND a.leg2dayofweek = d.`dayofweek`
   AND o.depdelay > 60
group by
```

```
   a.leg1uniquecarrier
;
```

3. Run the following query to get the number of passengers on an airline with missed international connections:

```
-- Number of passengers on airlines that have elongated layovers for an
international flight caused by missed connection
SELECT
   a.leg1uniquecarrier as carrier,
   count(a.leg1uniquecarrier) as passengers
--   o.arrdelay as delay
FROM
   `airline_ontime_orc`.unique_tickets a,
   `airline_ontime_orc`.flights o,
   `airline_ontime_orc`.flights d
where
       a.leg1flightnum = o.flightnum
   AND a.leg1uniquecarrier = o.uniquecarrier
   AND a.leg1origin = o.origin
   AND a.leg1dest = o.dest
   AND a.leg1month = o.month
   AND a.leg1dayofmonth = o.dayofmonth
   AND a.leg1dayofweek = o.`dayofweek`
   AND a.leg2flightnum = d.flightnum
   AND a.leg2uniquecarrier = d.uniquecarrier
   AND a.leg2origin = d.origin
   AND a.leg2dest = d.dest
   AND a.leg2month = d.month
   AND a.leg2dayofmonth = d.dayofmonth
   AND a.leg2dayofweek = d.`dayofweek`
   AND d.deptime-o.arrtime < o.arrdelay-45
group by
   a.leg1uniquecarrier
;
```

Generating and sharing dashboards

Use Cloudera Data Visualization to create dashboards from queries and share them with your stakeholders.

Procedure

1. Log in to the CDW service.
2. Go to Virtual Warehouses, click the options menu and select Open Viz.
3. Go to the DATA interface and click NEW DATASET.
4. On the **New Dataset** pop-up, specify a Dataset title.

   Select From SQL from the Dataset Source drop-down list.

   Copy and paste the following query in the Enter SQL below field and click CREATE:

```
-- Number of passengers on the airline that have long, planned layovers
for an international flight
SELECT
   a.leg1uniquecarrier as carrier,
   count(a.leg1uniquecarrier) as passengers
FROM
   `airline_ontime_orc`.unique_tickets a
where
   a.leg2deptime - a.leg1arrtime>90
group by
   a.leg1uniquecarrier
```

```
;
```

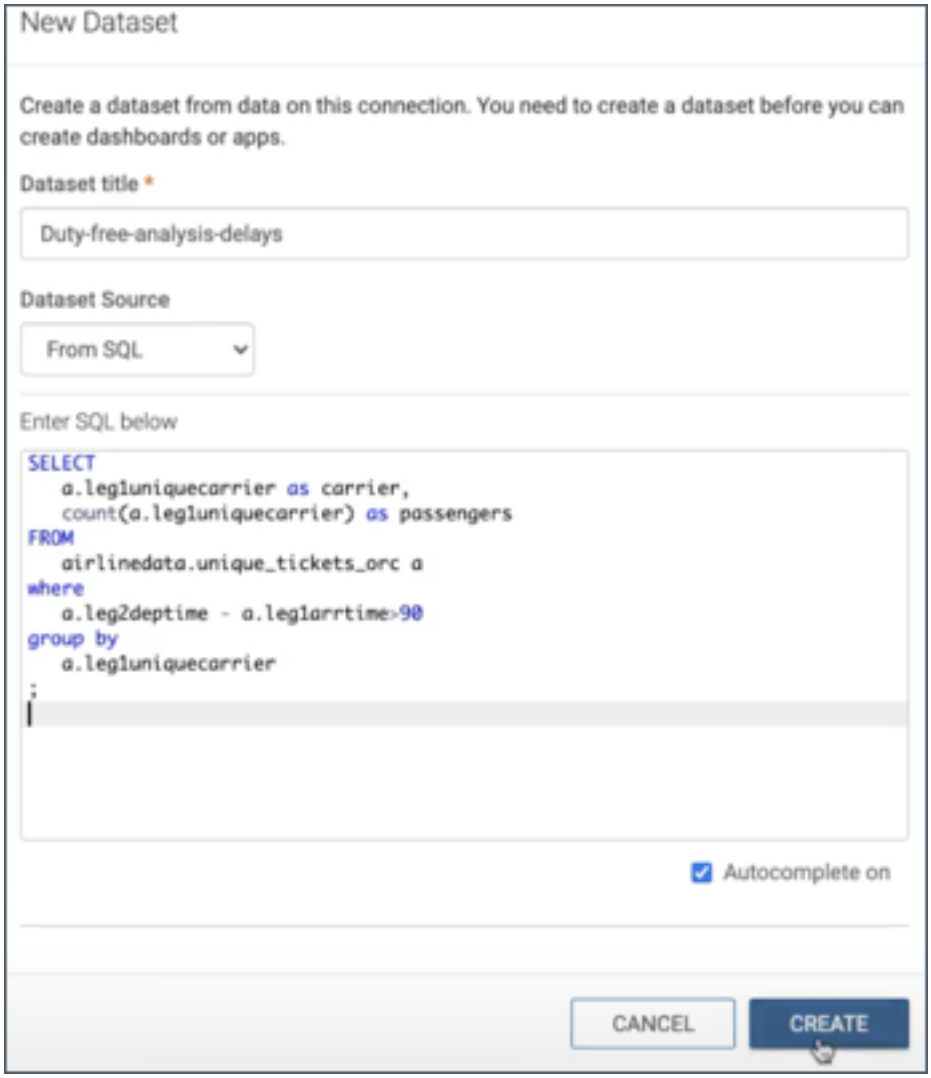

A new data set is created based on which you can create the visualization.

5. Go to the HOME interface and click NEW DASHBOARD.

   The **Dashboard Designer** opens on an untitled dashboard.

6. Enter a name for your dashboard, select the dataset that you just created from the ADD VISUALS menu, and click NEW VISUAL.

   The data is rendered in a tabular format from the SQL query.

7. Select a visual type from the VISUALS menu, such as a pie chart. Select the appropriate Dimensions and Measures from the DATA menu, and click REFRESH VISUAL.

   The pie chart shows the airlines having the most passengers with long layovers on international flights.

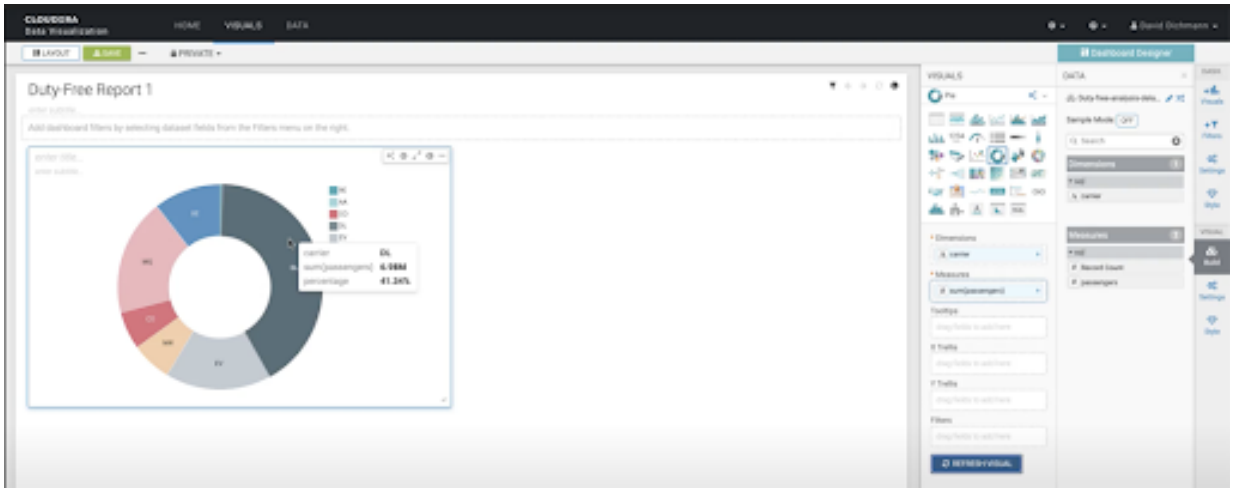

8. Click SAVE.
9. To share the dashboard:
   a) Open your dashboard from the VISUAL interface.
   b) Click the more option and select Get URL.

      A pop-up containing the URL is displayed.
   c) Copy the URL by right-clicking it and share it with your stakeholders.

Troubleshooting

This section helps you to narrow down a cause for an issue and provides a possible direction to resolve it.

Unable to see the S3 File Browser on the Hue web interface

Condition

Unable to see the S3 File Browser on the Hue web interface.

Cause

You may not have configured your Cloudera Data Warehouse (CDW) environment to have an S3 bucket that you can use to save data as part of the import process.

Solution

Add an S3 bucket in the CDW environment that you have access to. For more information, see the prerequisites section.

External File option is unavailable

Condition

External File option is unavailable on the Hue Importer dropdown list.

Cause

You may not have configured Hue to access the S3 bucket that you added in the environment.

Solution

Add the AWS access key and secret access key for your S3 bucket in the hue-safety-valve property. For more information, see the prerequisites section.

Error while accessing a directory in S3

Condition

Error stating "cannot access directory" while importing a file in Hue.

Cause

You may not have specified the right credentials in the hue-safety-valve property correctly.

Solution

Ensure that the account that owns the S3 bucket that you have added to your CDW environment has been added to the hue-safety-valve correctly.

Incorrect region error while uploading a file to S3

Condition

"Incorrect region" error while importing and uploading a file to S3 from your computer using Hue.

Cause

You may not have specified the AWS region where your S3 bucket is stored in the hue-safety-valve.

Solution

Ensure that the AWS region that you have specified in the hue-safety-valve is the same as the AWS region where your S3 bucket that you have added to your environment is managed.

CDP Patterns Public Cloud..

Business Intelligence at Scale

Date published: 2022-02-04
Date modified: 2022-02-04

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to Business Intelligence at Scale 4

Understanding the use case 6
Business Intelligence at Scale use case 7

Setting up your CDP pattern infrastructure 8

Giving access to users and groups 9
Giving access to Streams Messaging users 11
Giving access to DataFlow users 11
Giving access to Data Engineering users 12
Giving access to Data Warehouse users 13
Creating and assiging Ranger policies 13
Setting up the Streams Messaging cluster 15
Creating a Streams Messaging cluster 15
Setting up the Cloudera DataFlow service 17
Enabling DataFlow for an environment 17
Setting up the Cloudera Data Engineering service 18
Creating Data Hub cluster with Data Engineering cluster definition 18
Enabling Cloudera Data Engineering 18
Creating a virtual cluster 19
Setting up Cloudera Data Warehouse 20
Activating features under entitlement 20
Activating an AWS Environment for CDW 20
Adding an Impala Virtual Warehouse 20

Business Intelligence at Scale steps 21

Downloading the use case artifacts 21
Meeting the prerequisites 22
Ingesting data using Streams Messaging 22
Setting up the streaming data source 22
Adding schemas to the Schema Registry 23
Creating Kafka Topics 24
Building data pipelines using Cloudera DataFlow 24
Adding a ReadyFlow to the Catalog 24
Importing the custom NiFi flow definition 25
Collecting configuration information for ReadyFlow 26
Deploying the NiFi flows 27
Transforming and organizing data using Cloudera Data Engineering 31
Analyzing data and visualizing results using Cloudera Data Warehouse 33
About adding data to Data Warehouse 33
Creating tables and inserting data 33
Creating a materialized view 35
Creating a Data Visualization dashboard by importing visual artifacts 36

Introduction to Business Intelligence at Scale

The Business Intelligence (BI) at Scale pattern helps you to build and implement a sustainable BI solution using the Cloudera Data Platform (CDP) and various data services using which you can ingest, stream, explore, optimize, and analyze real-time and enterprise data, and then visualize the results to drive business insights.

Note: The Business Intelligence at Scale pattern is in Technical Preview and is not recommended for production use. Cloudera recommends that you try this pattern in your development or test environments.

The Business Intelligence at Scale pattern provides prescriptive steps to help you use the Cloudera data pipeline to stream data from a Streams Messaging cluster in CDP Public Cloud to AWS buckets. You can use Cloudera Data Engineering to organize and convert your data, and make the data available to Cloudera Data Warehouse for queries and analysis. Finally, you can use Cloudera Data Visualization tools to create consumable visualizations and dashboards.

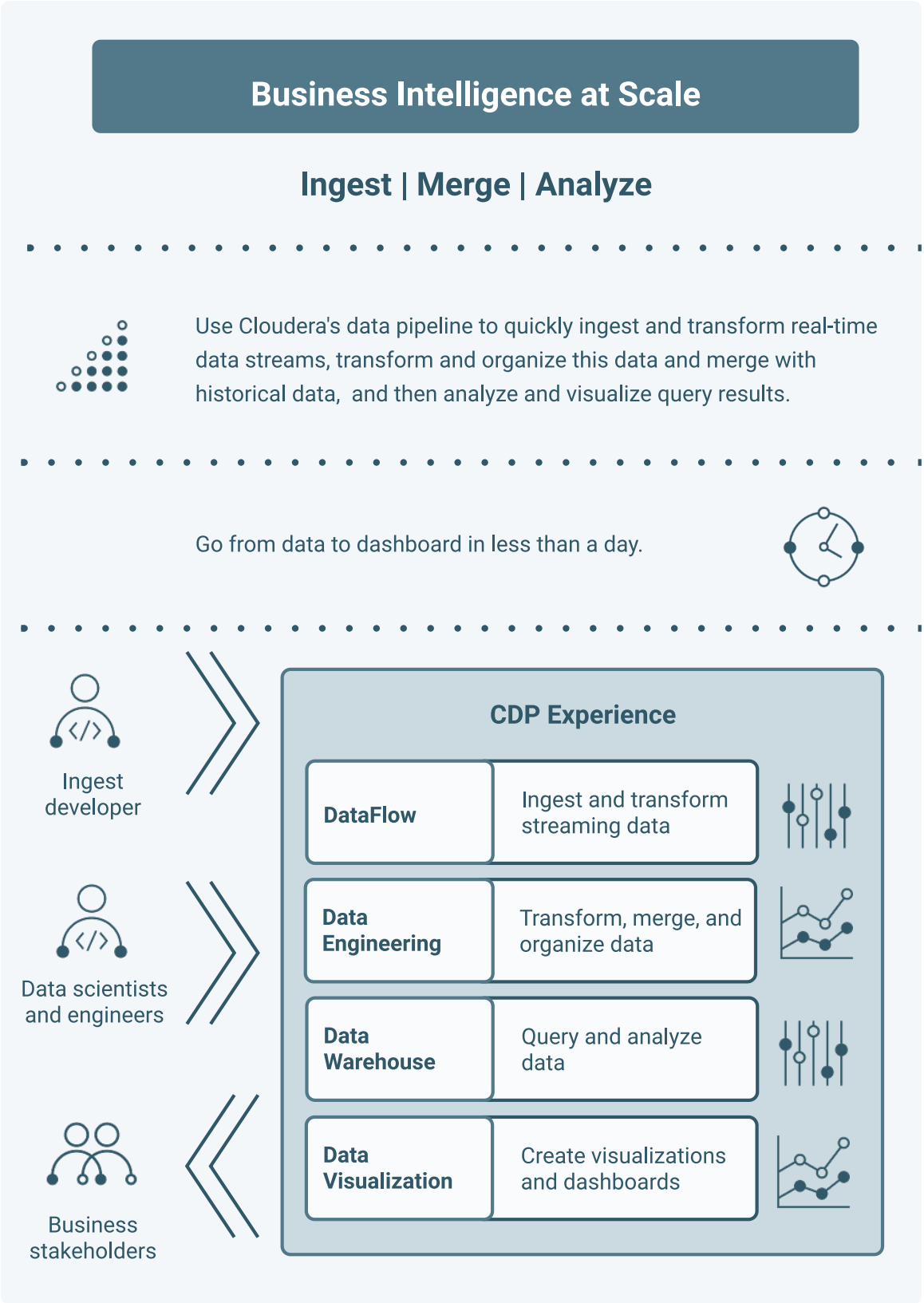
Business Intelligence at Scale
Ingest | Merge | Analyze
Use Cloudera's data pipeline to quickly ingest and transform real-time data streams, transform and organize this data and merge with historical data, and then analyze and visualize query results.
Go from data to dashboard in less than a day.
Ingest developer
Data scientists and engineers
Business stakeholders
CDP Experience
DataFlow
Ingest and transform streaming data
Data Engineering
Transform, merge, and organize data
Data Warehouse
Query and analyze data
Data Visualization
Create visualizations and dashboards

Understanding the use case

Learn how the Business Intelligence at Scale pattern empowers Ingest developers, Data engineers, Data Analysts, and Business stakeholders by using various Cloudera data services such as Streams Messaging Manager, DataFlow, Data Engineering, Data Warehouse, and Data Visualisation.

Business Intelligence at Scale is a common design pattern that you can implement in situations where business groups need insight from both enterprise and streaming data to make timely decisions. A traditional analytics solution would struggle with ingesting streaming data due to the technical challenges or experience needed in setting up a streaming ingest. This pattern guides Ingest Engineers to easily ingest streaming data, Data Engineers to model and curate data, and Business Analysts or Data Scientists to analyze, visualize, and share insights through an integrated self-service experience.

The objective is to allow business stake holders to quickly analyze all the data by using the integrated data lifecycle tools built into CDP. This service offers a smooth user experience for line of business decision makers to get answers to common questions through ready to use reports and dashboards, and an enhanced exploratory experience to line of business data analysts and power users to scrutinize data by building pipelines and querying the data sets.

Key problem this pattern solves

Customers can be more agile and better informed when making business decisions or creating ad-hoc reports based on new data as it arrives and becomes available.

Cloudera data lifecycle components

Business Intelligence at Scale pattern uses:

- Cloudera's Streams Messaging Manager (SMM) to ingest external (non-enterprise) data by creating Kafka topics
- Cloudera DataFlow Experience to convert and upload data to Amazon S3 (or any Cloud object store) in Avro format
- Cloudera Data Engineering service to transform data into ORC or Parquet format using Spark jobs
- Cloudera Data Warehouse service to create tables and materialized views from the datasets, run queries, and ad-hoc exploratory analysis
- Cloudera Data Visualization to create reports and dashboards

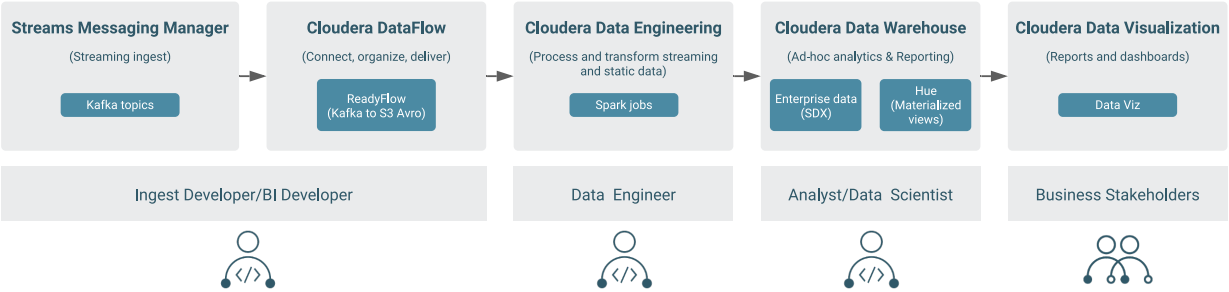

Using these Cloudera services:

- Ingest developers can quickly develop data flows for real-time data and merge incoming data with existing data sets
- Data engineers can transform, organize, and enrich data
- Data Analysts and Data Scientists can query the data sets using Hue
- Business stakeholders can derive insights using visualizations and dashboards

Business Intelligence at Scale use case

A real-life business narrative enables you to follow and understand the Business Intelligence at Scale CDP pattern. The narrative introduces the key personas at play, establishes burning business questions, and sets parameters for the success criteria of the pattern.

This is a sample use case. The steps in this pattern can be applied to similar use cases across other industries.

The use case: Analyzing the impact of dynamic weather conditions on static inventory management for perishable goods

Broger Foods is a mega grocery retail chain, with multiple outlets across the country, selling 100+ perishable goods. Each year, Broger Foods suffers losses ranging between 30 and 45 million dollars due to inventory shrinkage. Damages and spoilage of perishable items in warehouses has been identified as a key contributor to the shrinkage. The challenge Broger faces is the relatively static inventory management system for perishable goods. Stocks of fresh produce and food rot or expire in the warehouses due to intermittent demand and over stocking. Understocking is not a viable option due to the competitive nature of the industry, leading to customers turning to Hole Foods, a rival retail chain gaining popularity.

The CFO of Broger Foods, Roger Walters has tasked their VP of Analytics, Richard, to look into this issue and help solve Broger's supply chain management problem.

Key business questions

- How can Broger Foods optimize its inventory management system to reduce revenue loss and prevent food wastage while not loosing customers to its competitors?
- How can the BI team at Broger Foods deliver insights at scale?

Solving the problem

Richard formulates two hypotheses to analyze the sales trends. His first hypothesis is based on population density of a location and the second one is based on the weather conditions. Upon delving deeper using ad-hoc analytics, Richard figured out that there is a strong correlation between demand for perishable items in each location and weather conditions in that location. To continuously improve Broger’s inventory management process, Richard plans to use the location-wise sales information present in Broger's CDP Public Cloud environment along with real-time weather information from a weather data provider.

To deliver the insights at scale, Richard calls for a meeting with Syd (Streaming App Developer), David (Data Engineer), and Nick (Data Analyst) to explain his model and what he needs to implement this new inventory forecasting model.

To build a streaming pipeline, Richard asks Syd to capture live weather data from OpenWeather and ingest it into CDP. Syd sets up a data flow that ingests data into CDP in real time.

He then asks David to build an unified data pipeline that aggregates data from each stream into two separate tables:

- Current weather data. This table captures temperature, humidity, precipitation, and pressure.
- Historical data for 5 previous days.

David’s pipeline will be triggered to process and aggregate the freshly ingested data every hour and update the Optimized Row Columnar (ORC) or Parquet tables in CDW every hour.

Now, Nick is tasked to write queries that averages out the weather data from both the weather tables and combine it with the location-based sales data for additional analysis. He also has to build a dashboard that showcases the variance between item sales and temperature at a given store location on a daily basis. The dashboard also shows a 7 day forecast of sales. The sales forecast is done on Monday and Friday every week.

Key personas in this use case

- Ingest Developer – Develops and explores the data ingest streams

- Data Engineer – Prepares the data pipelines, optimizes and aggregrates data for use by Analysts
- Data Analyst – Analyzes data using queries and by building reports and dashboards
- Business stakeholders – Consume analyzed data and visualizations

Success criteria

Rapidly develop and build data streams to deliver business insights at scale by enabling self-service DataFlow development, Data Engineering, Data Warehousing, and Business Intelligence reporting and dashboarding capabilities.

Setting up your CDP pattern infrastructure

Before starting, ensure that your central or departmental IT has onboarded to CDP Public Cloud and registered an environment with Cloudera. Business Intelligence at Scale pattern leverages Streams Messaging Manager (SMM), Cloudera DataFlow (CDF), Cloudera Data Engineering (CDE), and Cloudera Data Warehouse (CDW) services. You must set up these services as part of your infrastructure.

Most steps required to set up your CDP pattern infrastructure should be performed by a CDP administrator and require the EnvironmentAdmin CDP role. Since the goal of this pattern is to enable self-service DataFlow development, Data Engineering, and Data Warehousing, make sure that the IT team has created the necessary users and groups, and have granted the required permissions to these users and groups.

When your IT team creates and registers an environment with CDP, it is assumed that they create a medium-duty Data Lake to import and manage streaming data in the Streams Messaging Data Hub cluster, as well as when running other CDP services.

Before you begin self-service development and infrastructure setup:

- Ensure that you have an available CDP environment
- Ensure that you have CDP login credentials
- Ensure that you have a running Data Lake
- Ensure that your CDP user is synchronized to the CDP Public Cloud environment
- Review the AWS environments requirements checklist

For test and development environments, you can set up small to medium sized clusters, and scale up for production use.

After you register your environment with CDP, you must also activate entitlements for using Unified Analytics and Data Visualization in CDW by contacting your Cloudera Account Representative.

The following diagram shows all the Data Services used to implement the Business Intelligence at Scale pattern:

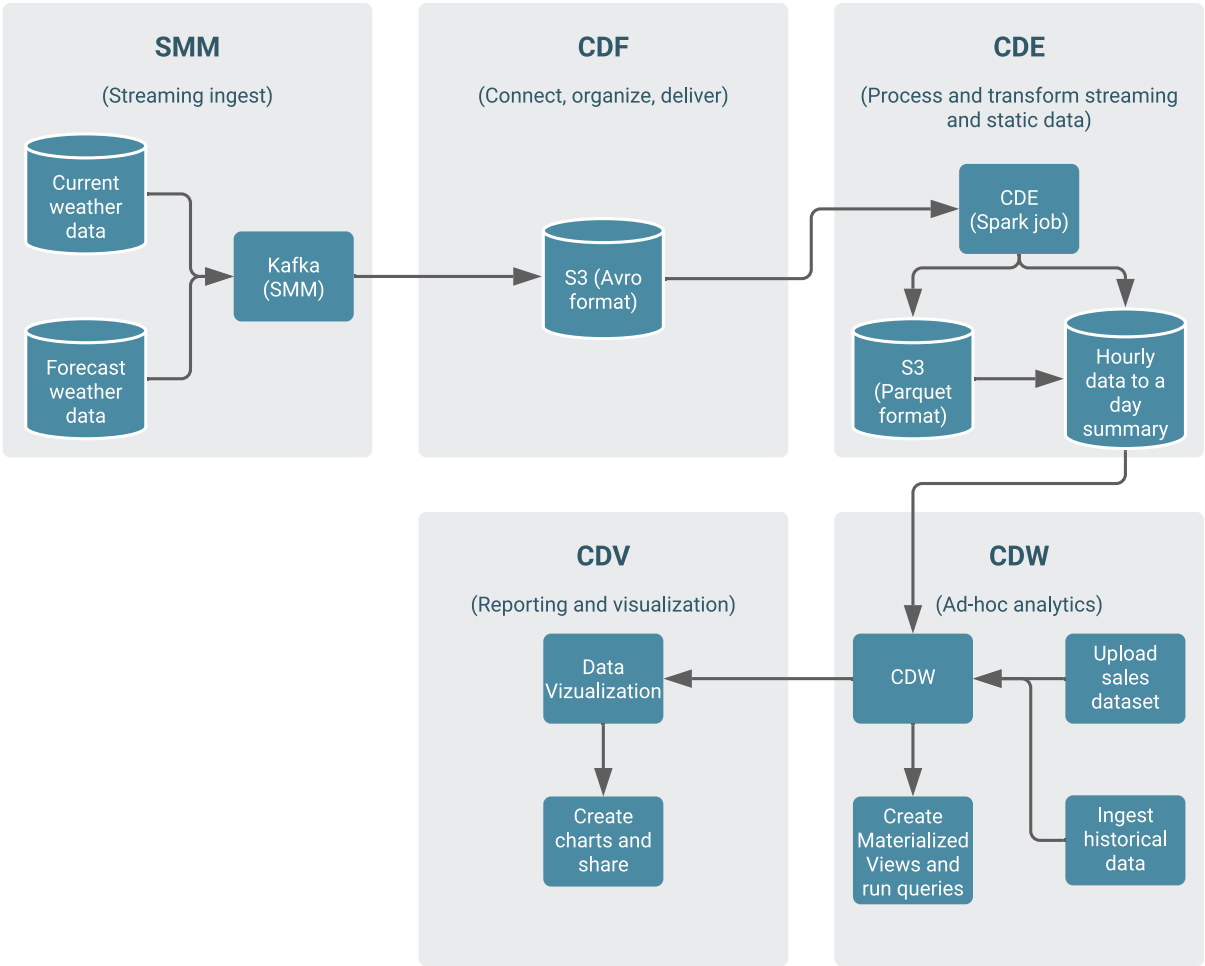

Giving access to users and groups

To access and use the CDP Data Services, you must have the required roles and permissions. Learn about the various roles that users need to perform specific tasks within the Business Intelligence at Scale pattern.

The following graphic shows the roles required to set up and access various CDP services with this pattern and who must grant those roles:

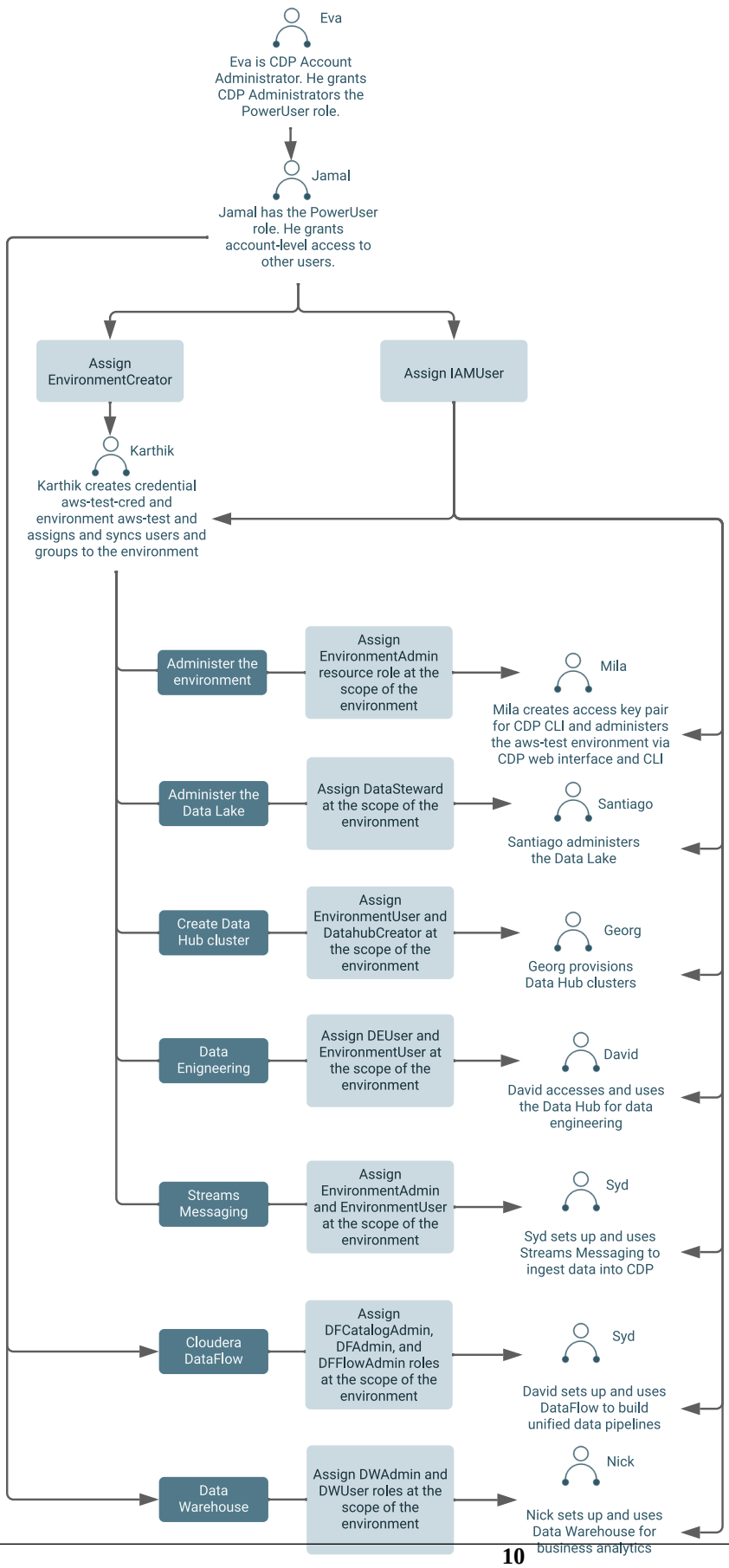

Eva is the CDP Account Administrator. Eva is typically the head of IT and purchases the CDP license for her organization.

Eva registers their cloud environment (such as AWS) with CDP and attaches an identity federation to add users to the CDP environment.

Jamal is a Power User. He has full admin access to the CDP tenant. He can assign account roles to other users who need access to CDP.

He assigns IAMUser to all onboarded users so that they can generate access keys, upload SSH keys, and assign resouces to other users.

Jamal assigns EnvironmentCreator to Karthik. This allows Karthik to create a provisioning credential aws-test-cred in CDP and register a cloud provider environment aws-test-env.

Note that since Karthik created the credential and the environment, he automatically gets the EnvironmentAdmin and Owner role over both resources.

Karthik assigns the EnvironmentAdmin resource role to Mila over his aws-test environment. This allows Mila to administer this specific environment.

Karthik assigns the DataSteward resource role to Santiago over his aws-test environment. This allows Santiago to administer the Data Lake running in this specific environment.

Karthik assigns the EnvironmentUser and DatahubCreator resource role to Georg over the aws-test environment.

Georg provisions a Data Hub and automatically gets Owner role at the scope of that Data Hub.

Karthik assigns the EnvironmentUser and DEUser resource roles to David over the aws-test environment so that David can access Data Hubs provisioned by Georg within the environment and import and run Spark jobs.

In addition, Jamal assigns the IAMUser account role to David so that David can access the S3 buckets.

Syd provisions a Streams Messaging cluster, creates Kafka topics, and adds schemas to the Schema Regisrty.

Syd also sets up the CDF service, enables the CDF environment, and builds and deploys flows.

In addition, Jamal assigns IAMUser account to Syd so that Syd can access the S3 buckets.

Nick activates the aws-test environment in CDW , creates Virtual Warehouses, runs queries, and generate reports using Data Visualization.

In addition, Jamal assigns IAMUser account to Nick so that Nick can access the S3 buckets.

Tip: If more than one user needs a certain role or permission to work with a CDP service, Cloudera recommends that you add those users into a group and grant a role or permission to that group.

Related Information

Resource roles

Giving access to Streams Messaging users

An "EnvironmentAdmin" or a "DataSteward" must grant the "EnvironmentUser" role to the user who is tasked to set up the Streams Messaging cluster.

Before you begin

Important: You must have an EnvironmentAdmin or a DataSteward role to perform this task.

About this task

The cluster you have created using the Streams Messaging cluster definition is kerberized and secured with SSL. Users can access cluster UIs and endpoints through a secure gateway powered by Apache Knox. Before you can begin working with Kafka, Schema Registry, Streams Messaging Manager, and Streams Replication Manager, you must give users access to the Streams Messaging cluster components.

Procedure

1. Log in to the CDP web interface.
2. Go to Management Console Environments .
3. Go to Actions Manage Access to display the **Environment Access** page.
4. Find the user to whom you want to grant the EnvironmentUser role, and click Update Roles.
5. Assign the EnvironmentUser role to the users to grant access to the CDP environment and the Streams Messaging cluster.
6. Go to the **Environments** page.
7. Click Actions Synchronize Users to FreeIPA .

   Depending on how many users have access to the environment, this synchronization process can take a few seconds to a few minutes.

Giving access to DataFlow users

Cloudera DataFlow (CDF) restricts who can import flow definitions and deploy them. A "PowerUser" must grant "DFAdmin", "DFFlowAdmin", and "DFCatalogAdmin" roles to users so that they can enable CDF for an environment, import, add, and deploy flow definitions in CDF.

About this task

- The DFAdmin role is required to enable CDF for an environment.
- The DFFlowAdmin role is required deploy flow definitions in CDF.
- The DFCatalogAdmin role is required to import flow definitions to the CDF Catalog.

Before you begin

Important: You must have a PowerUser role to perform this task.

Cloudera recommends that you create a dedicated machine user as a CDP Workload User for CDF. Make sure to set the Workload Password. You need the workload username and password for deploying NiFi flows.

Note: When you create a machine user, then a prefix, "srv", is automatically added to the machine user name. For example, if you create a machine user as "machine-user-1", then the Workload User Name reads "srv_machine-user-1".

You must also assign the EnvironmentUser role to this machine user within your environment.

Procedure

1. Log in to the CDP web interface.
2. To give a user the permission to enable CDF for an environment:
   a) Go to Management Console Environments .

      The **Environment List** page is displayed.
   b) Select the environment in which you want a user to enable CDF.
   c) Go to Actions Manage Access to display the Environment Access page.
   d) Find the user to whom you want to grant the DFAdmin role, and click Update Roles.
   e) Select DFAdmin and click Update Roles.
3. To give a user or group the permission to deploy flow definitions.
   a) Go to Management Console Environments .

      The **Environment List** page is displayed.
   b) Select the environment to which you want a user or group to deploy flow definitions.
   c) Go to Actions Manage Access .

      The **Environment Access** page is displayed.
   d) Find the user or group and click Update Roles.
   e) Select DFFlowAdmin and click Update Roles.
4. To give a user the permission to import flow definitions:
   a) Go to Management Console User Management .
   b) Enter the name of the user or group you wish to authorize in the Search field.
   c) Select the user or group from the list that displays.
   d) Click Roles, then Update Roles.
   e) From Update Roles, select DFCatalogAdmin and click Update.
5. Go to the **Environments** page.
6. Click Actions Synchronize Users to FreeIPA .

   Depending on how many users have access to the environment, this synchronization process can take a few seconds to a few minutes.

Giving access to Data Engineering users

Cloudera Data Engineering (CDE) restricts access to importing and running Spark jobs. An "EnvironmentAdmin" must grant both "DEUser" and "EnvironmentUser" roles to users so that they can run Spark jobs in CDE and access the Data Hub clusters for data engineering.

Before you begin

Important: You must have an EnvironmentAdmin role to perform this task.

Procedure

1. Log in to the CDP web interface.
2. Go to Management Console Environments .
3. Select the environment in which you want a user or group to import and run Spark jobs.

4. Go to Actions Manage Access .

   The **Environment Access** page is displayed.
5. Find the user or group using Search and click Update Roles.
6. Select the DEUser and EnvironmentUser options and click Update Roles.
7. Go to the **Environments** page.
8. Click Actions Synchronize Users to FreeIPA .

   Depending on how many users have access to the environment, this synchronization process can take a few seconds or a few minutes.

Giving access to Data Warehouse users

A "PowerUser" must grant "DWAdmin" and "DWUser" roles to users and groups so that they can activate the Cloudera Data Warehouse (CDW) service, create Database Catalogs and Virtual Warehouses, and manage Hue and other applications within CDW.

Before you begin

Important: You must have a PowerUser role to perform this task.

Procedure

1. Log in to the CDP web interface.
2. Go to Management Console Environments and select your CDP environment.
3. Go to Actions Manage Access .

   The **Environment Access** page is displayed.
4. Find the user or group using Search and click Update Roles.
5. Select the DWAdmin and DWUser options and click Update Roles.

   Note: When you grant access to groups, you must grant the DWAdmin resource role to the group instead of the DWUser role.
6. Go to the **Environments** page.
7. Click Actions Synchronize Users to FreeIPA .

   Depending on how many users have access to the environment, this synchronization process can take a few seconds or a few minutes.

Creating and assiging Ranger policies

Apart from assigning resource roles to users, you need to set up Ranger policies to authorize users (service, machine, or workload users) to perform specific operations and rights to certain resources. Learn which Ranger policies you need for the Business Intelligence at Scale pattern and how to create them.

Before you begin

Important: You must have the EnvironmentUser role to perform this task.

Ensure that you have a list of machine users and workload users you want to add to the specific Ranger policies.

About this task

The following table lists the CDP components for which you need to set up resource-based Ranger policies:

CDP component	Resource-based Ranger policy	Purpose
Kafka (Streams Messaging)	Kafka	Allow the machine user to publish, configure, and consume Kafka topics.
	Kafka	Allow the machine user to consume the Consumer Group IDs.
Schema Registry (Streams Messaging)	Schema-Registry	Allow the machine user to read schema groups from the Schema Registry.
Hue (Cloudera Data Warehouse)	Hadoop SQL	Allow the workload user to perform all database, table, and column operations such as select, update, alter, create, drop, insert, read, and write.

Note: A set of default policies are automatically created for your deployment. You must create a new policy within your main deployment policy.

Procedure

1. Log in to the CDP web interface.
2. Go to Management Console Environments and click on Ranger.
   The Ranger **Service Manager** page is displayed.
3. Create a Kafka policy to allow the machine user to publish, configure, and consume Kafka topics.
   a) Click on your main deployment (SMM) policy under **KAFKA**.
   b) Click Add New Policy on the **List of Policies** page.
   c) Specify a name for your policy in the Policy Name field.
   For this pattern, you can specify bias-smm-kafka-ingest.
   d) Select topic from the drop-down menu.
   e) Enter the names of the Kafka topics to which you want to authorize user access.
   For this pattern, specify weather, weather_forecast or weather*.
   f) Go to the **Allow Conditions** section and enter the machine username under the Select User column.
   g) Click + under the Permissions column and select Publish, Consume, and Configure options and click ✓.
   h) Click Add at the bottom of the page.
4. Create a Kafka policy to allow the machine user to consume, describe, and delete the Consumer Group IDs.
   a) Click on your main deployment (SMM) policy under **KAFKA**.
   b) Click Add New Policy on the **List of Policies** page.
   c) Specify a name for your policy in the Policy Name field.
   For this pattern, you can specify bias-smm-kafka-consumer.
   d) Select consumergroup from the drop-down menu.
   e) Enter the names of the Kafka Consumer Group ID to which you want to authorize user access.
   For this pattern, specify WeatherConsumer, ForecastConsumer.
   f) Go to the **Allow Conditions** section and enter the machine username under the Select User column.
   g) Click + under the Permissions column and select the Consume option and click ✓.
   h) Click Add at the bottom of the page.

5. Create a Schema-Registry policy to allow the machine user to read the schema groups from the Schema Registry.
   a) Click on your main deployment (SMM) policy under **SCHEMA-REGISTRY**.
   b) Click Add New Policy on the **List of Policies** page.
   c) Specify a name for your policy in the Policy Name field.
      For this pattern, you can specify bias-smm-sr-schema-group.
   d) Select schema-group from the drop-down menu and enter weather next to it.
   e) Enter the names of the Schema Registry schemas to which you want to authorize user access in the Schema Name field.
      For this pattern, specify WeatherCurrent, WeatherForecast.
   f) Select schema-branch from the drop-down menu and enter *.
   g) Select schema-version from the drop-down menu and enter *.
   h) Go to the **Allow Conditions** section and enter the machine username under the Select User column.
   i) Click + under the Permissions column and select the Read option and click ✓.
   j) Click Add at the bottom of the page.
6. Create a Hadoop SQL policy to allow the workload users to perform operations on databases, tables, and columns.
   a) Click Hadoop SQL under **HADOOP SQL**.
   b) Click Add New Policy on the **List of Policies** page.
   c) Specify a name for your policy in the Policy Name field.
      For this pattern, you can specify bias-all-database-table-column.
   d) Select database from the drop-down menu and enter *.
   e) Select table from the drop-down menu and enter *.
   f) Select column from the drop-down menu and enter *.
   g) Go to the **Allow Conditions** section and enter the workload username under the Select User column.

      Tip: To provide access to all users, you can specify {USER}.

   h) Click + under the Permissions column and select the select, update, Create, Drop, Alter, Read, and Write option and click ✓.
   i) Click Add at the bottom of the page.

Setting up the Streams Messaging cluster

Streams Messaging provides advanced messaging and real-time processing on streaming data using Apache Kafka, centralized schema management using Schema Registry, as well as management and monitoring capabilities powered by Streams Messaging Manager. Learn how to set up a Streams Messaging cluster in CDP Public Cloud for use with your Business Intelligence at Scale pattern.

For more information about the Streams Messaging cluster, see *Streams Messaging clusters* and *How To: Streams Messaging* in the latest CDF for Data Hub library.

Related Information

Streams Messaging clusters
CDF for Data Hub

Creating a Streams Messaging cluster

Learn how to create a Streams Messaging cluster for the Business Intelligence at Scale pattern. A Data Hub cluster with the Streams Messaging cluster definition hosts the Kafka topics into which you capture real-time streaming data. For this example, you are importing the weather data.

Before you begin

Important: You must have an EnvironmentAdmin role to perform this task.

About this task

For the Business Intelligence at Scale pattern proof of concept, use the Streams Messaging Light Duty template for AWS. When you are ready to move your data pipeline to production, you can use either a Light or Heavy duty template, depending on your use case. For this pattern, use the default cluster definition.

Procedure

1. Log into the CDP web interface.
2. Navigate to Management Console Environments , and select the environment where you would like to create a cluster.
3. Click Create Data Hub.

   The following page is displayed:

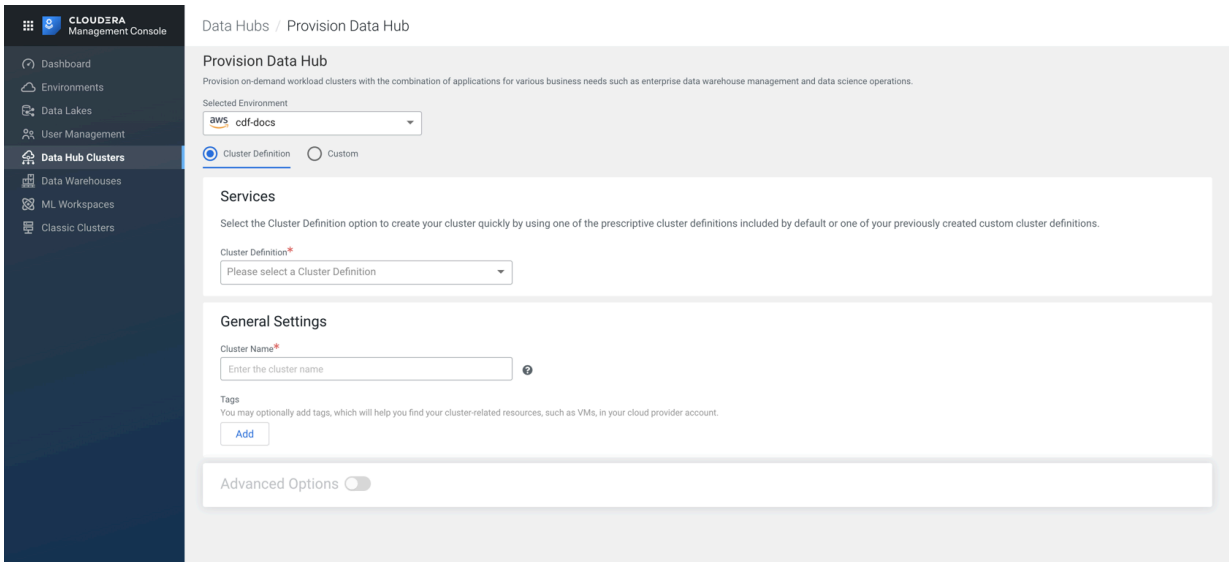

4. Select the Cluster Definition option.
5. Select the appropriate Streams Messaging cluster definition from the Cluster Definition drop-down menu under the Services section depending on your operational objectives.

   There are two definition options available for each cloud provider:

   - Streams Messaging Heavy Duty for AWS
   - Streams Messaging Light Duty for AWS

   The list of services is automatically shown below the selected cluster definition name as shown in the following image:

6. Specify a name for your cluster and add any tags you might need.
7. Click Provision Cluster.

Results

You are redirected to the Data Hub cluster dashboard, and a new tile representing your cluster appears at the top of the page.

Setting up the Cloudera DataFlow service

In the Business Intelligence at Scale pattern, Cloudera DataFlow (CDF) provides basic flow development and streaming data flow organization and management capabilities.

To set up a CDF cluster, a user with the PowerUser role must first grant you the DFAdmin role. You can then enable the CDF service in your CDP environment.

Attention: Enabling CDF can take upto one hour.

For the Business Intelligence at Scale pattern, you will be using the "Kafka to Avro S3" ReadyFlow flow definition.

For more information about the CDF service, see the *Cloudera DataFlow documentation*.

Related Information

Cloudera DataFlow documentation

Enabling DataFlow for an environment

To deploy flow definitions, you must enable Cloudera DataFlow (CDF) for a CDP Public Cloud environment. Enabling CDF for an environment means that you are preparing an active and healthy CDP environment for use with CDF.

Before you begin

Important: You must have the DFAdmin role to perform this task.

Procedure

1. Log in to the CDP web interface and click on the DataFlow service.
2. Go to Environments and click Enable against the environment you want to enable.
3. On the **Enable Environment** page:
   - Specify the minimum and maximum number of the Kubernetes nodes you want to deploy in the **DataFlow Capacity** section.
   - Under the **Networking** section, select the subnet from the Available Subnets that CDF can use for worker node and load balancer placement.

     If you want to access CDF components from the internet, select the Enable public endpoint option.

     Important: To specify a public endpoint, you must have at least one public subnet in your environment.
   - A list of source IP address ranges which are allowed to connect to the Kubernetes API server.
4. Click Enable.

   Enabling CDF can take up to one hour.

Setting up the Cloudera Data Engineering service

Set up Cloudera Data Engineering in CDP Public Cloud for use with your Business Intelligence at Scale pattern. In the Business Intelligence at Scale pattern, Data Engineering provides the streaming and static data processing and transformation capabilities.

Attention: Enabling CDE can take upto two hours.

For more information about Cloudera Data Engineering, see the *Data Engineering Overview*.

Related Information

Data Engineering Overview

Creating Data Hub cluster with Data Engineering cluster definition

You must create a Data Hub cluster with the "Data Engineering for AWS" cluster definition to obtain the Hive JDBC URL. You need the Hive JDBC URL while configuring the Spark job.

Before you begin

Important: You must have the EnvironmentUser role to perform this task.

Procedure

1. Log into the CDP web interface.
2. Navigate to Management Console Environments , and select the environment where you would like to create a cluster.
3. Click Create Data Hub.
4. Select the Cluster Definition option.
5. Select Data Engineering for AWS cluster definition from the Cluster Definition drop-down menu under Services.

   The list of services is automatically shown below the selected cluster definition name.
6. Specify a name for your cluster and add any tags you might need.
7. Click Provision Cluster.

Enabling Cloudera Data Engineering

Before you can use the Cloudera Data Engineering (CDE) service, you must enable it on each environment that you want to use CDE on.

Before you begin

Important: You must have the DEAdmin role to perform this task.

Procedure

1. Log in to the CDP web interface and click on the Data Engineering service.
2. In the CDE Services column, click ⊕ or Enable new CDE Service to enable CDE for an environment.
3. On the **Enable CDE Service** page, specify a name for the CDE service.
4. Select your environment from the Environment drop-down menu.

5. Select the Workload Type.

   The workload type corresponds to the instance size that will be deployed to run your submitted Spark jobs. When you select a type, the corresponding cloud provider instance size is displayed in the **Summary** section to the right.

   For proof of concept of this pattern, you can select General - Medium.
6. Accept the default settings for the Autoscale Range.
7. Select the Enable Public Endpoint option under the **Networks** section.
8. Click Enable.

Results

The CDE Overview page displays the status of the environment initialization. You can view logs for the environment by clicking ⋮ on the environment, and then clicking View Logs.

Creating a virtual cluster

In Cloudera Data Engineering (CDE), a virtual cluster is an individual auto-scaling cluster with defined CPU and memory ranges. Jobs are associated with virtual clusters, and virtual clusters are associated with an environment. You can create as many virtual clusters as you need.

Before you begin

Important: You must have the DEAdmin role to perform this task.

Procedure

1. From the CDE Overview page, select the CDE service that you created earlier.
2. In the Virtual Clusters column, click ⊕ to create a new virtual cluster.

   If the environment has no virtual clusters associated with it, the page displays a Create DE Cluster option that launches the same wizard.
3. On the **Create a Cluster** page, enter a Cluster Name.

   Cluster names must:

   - begin with a letter
   - be between 3 and 30 characters (inclusive)
   - contain only alphanumeric characters and hyphens
4. Select the CDE Service that you created earlier from the CDE Service drop-down menu.

   The CDE service you selected before launching the wizard is selected by default, but you can use the wizard to create a virtual cluster in a different CDE service.
5. Accept the default settings for the Autoscale Max Capacity.
6. Select Spark 2.4.7 as the Spark Version to use in the virtual cluster.

   You cannot use Spark 2 and Spark 3 in the same virtual cluster, but you can have separate Spark 2 and Spark 3 virtual clusters within the same CDE service.
7. Click Create.

Results

On the CDE Overview page, select the environment to view the virtual cluster initialization status. You can also click ⋮ for the virtual cluster to view the logs.

Setting up Cloudera Data Warehouse

In the Business Intelligence at Scale pattern, Data Warehouse provides ad-hoc data analytics capabilities and the Data Visualization integration provides visual reporting and dashboarding capabilities. To set up the CDW service, you must activate an environment and create an Impala Virtual Warehouse with the Unified Analytics and Data Visualization options enabled. You can use the default Database Catalog that is created when you activate an environment.

For more information about the Cloudera Data Warehouse service, see the *Data Warehouse Overview*.

Related Information

Cloudera Data Warehouse Data Service overview

Activating features under entitlement

Unified Analytics and Data Visualization features in Cloudera Data Warehouse (CDW) are under entitlement. Contact your Cloudera Account Representative to activate these features for your environment.

Procedure

Activate the following entitlements after you register your cloud environment with CDP:

- CDW_STANDALONE_DATA_VISUALIZATION: To enable Cloudera Data Visualization
- CDW_FENG_IMPALA: To use Unified Analytics in Hue

Activating an AWS Environment for CDW

Activating an environment with Cloudera Data Warehouse service sets up the Kubernetes cluster, which provides the computing resources for the Database Catalog. In addition, activating an environment enables the CDW service to use the existing Data Lake that was set up for the environment, including all data, metadata, and security.

Before you begin

Important: You must have the DWAdmin role to perform this task.

Procedure

1. Log in to the CDP web interface and click Data Warehouse.
2. Expand the Environments column by clicking More… and locate the environment that you want to activate.
3. When you locate the environment, click the activation icon to launch the **Activation Settings** dialog.
4. In the **Activation Settings**, select the Private Load Balancer, Private Worker Nodes option from the Deployment Mode drop-down menu.
5. Click ACTIVATE.

Adding an Impala Virtual Warehouse

A Virtual Warehouse is an instance of compute resources that is equivalent to a cluster. You must create an Impala Virtual Warehouse, with the Unified Analytics option selected, to work with materialized views. You must also select the Enable Data Visualization option, so that you can use tables and queries to generate dashboards in Cloudera Data Visualization.

Before you begin

Important: You must have the DWAdmin role to perform this task.

Procedure

1. Log in to the CDP web interface and click the Data Warehouse service.
2. Go to Virtual Warehouses and click ADD NEW.
3. On the **New Virtual Warehouse** dialog:
   a) Specify a Name.

      Important: The fully qualified domain name of your Virtual Warehouse, which includes the Virtual Warehouse name plus the environment name, must not exceed 64 characters; otherwise, Hue cannot load.
   b) Select IMPALA as the Virtual Warehouse type.
   c) Select the Database Catalog associated with your environment.
   d) Specify the User Groups that can access the endpoints.
   e) Select a size from the Size drop-down menu.
   f) Accept the default configuration for the auto-scaling thresholds.
   g) Click Enable Unified Analytics.
   h) Click Enable Data Visualization .
4. Click CREATE.

Business Intelligence at Scale steps

Learn how to deploy the Broger Foods' inventory management use case in your environment. You can follow the same steps as a guide to deploy your use case leveraging the Business Intelligence at Scale pattern.

Downloading the use case artifacts

To get you started with the Broger Foods' Business Intelligence at Scale use case, Cloudera has made the required use case artifacts available to you. You can use these artifacts as a starting point to build your own use case.

Schema Registry schemas for ingesting data into your Streams Messaging cluster

- weather-current-schema
- weather-forecast-schema

Custom Cloudera DataFlow flow definition for ingesting data into Kafka in your Streams Messaging cluster

- weather-data-ingest-into-kafka.json

Historical weather data and sales data ingest scripts to run in Impala Virtual Warehouse

- historical-weather-data-ingest.txt
- sale-data-ingest.txt
- bi-at-scale-historic-weather-data.zip
- bi-at-scale-historic-sales-data.zip

Spark jar file for running the Spark job in Cloudera Data Engineering

- bi-workflow_2.11-0.1.jar

Data Visualization dashboard to visualize the analytic findings

- weather-impact-on-sales.json

Meeting the prerequisites

The Business Intelligence at Scale pattern is designed to run on CDP and leverages various Cloudera Data Services. You must meet the individual prerequisites for Cloudera DataFlow (CDF), Cloudera Data Engineering (CDE), and Cloudera Data Warehouse (CDW) Data Services. All your users must have the necessary roles and permissions to perform their respective tasks.

Prerequisites for using the Business Intelligence at Scale pattern

- You have downloaded the pattern artifacts for use throughout the Business Intelligence at Scale pattern steps.
- You have signed up with openweathermap.org and have obtained their API key for real time weather data. This is so that you can stream weather data into Kafka.
- You have set up the necessary Ranger policies across your data pipeline. You have granted a user or group access to:
  - Kafka
  - Schema Registry
  - Cloudera DataFlow
  - Cloudera Data Engineering
  - Cloudera Data Warehouse

Prerequisites for using CDF

- You have the CDF service enabled and running.
- You have set up a Streams Messaging cluster in CDP Public Cloud with Kafka installed.
- You have created the required Kafka topics.
- You have the required schemas managed by Schema Registry.
- You have set up IDBroker mapping for CDF and CDE to access S3 buckets.

Prerequisites for using CDE

- You have created a Data Hub cluster with the Data Engineering cluster definition.
- You have the CDE experience enabled.
- You have one Virtual Cluster.
- You have access control set up for the team.

Prerequisites for using CDW

- You have the CDW experience enabled.
- You have activated the Unified Analytics and Data Visualization entitlement features with help of your Cloudera Account Representative.
- You have created an Impala Virtual Warehouse with Enable Unified Analytics and Enable Data Visualization options.
- You have the Hive Warehouse Connector installed.

Ingesting data using Streams Messaging

You must configure access to the streaming data from a third-party data provider, create Kafka topics, and add schemas to the Schema Registry in the Streams Messaging cluster.

Setting up the streaming data source

For the Business Intelligence at Scale pattern, you need to stream weather data from an external data provider into CDP. To obtain data from OpenWeather, you must sign up for an account with them and generate API keys.

Procedure

1. Go to https://openweathermap.org/api and create your account.
2. Go to your profile and click My API keys.
3. Specify an API key name and click Generate.

   A new API key is created. You will need this while deploying flows in CDF.

Adding schemas to the Schema Registry

Learn how to add new schemas to Schema Registry so that you ensure data consistency across your data pipeline.

About this task

Schema Registry provides a shared repository for your schemas to ensure that your data is accessible across the data pipeline. For the Business Intelligence at Scale Pattern, you must create the following two schemas using the Schema Regisrty files that you downloaded as part of the pattern artifacts:

- WeatherCurrent
- WeatherForecast

Before you begin

Important: You must have the EnvironmentAdmin role to perform this task.

Attention: Ensure that you understand compatibility policies. Once selected, you cannot change the compatibility policy for a schema.

Procedure

1. Log in to the CDP web interface.
2. Go to Management Console Data Hub Clusters and select the CMM cluster that you created earlier.
3. Click Schema Registry under the **Services** section.

   The **All Schemas** page is displayed.
4. Click **+** .

   The **Add New Schema** page is displayed.
5. Add the schema metadata as follows:

Field name	Description and values
NAME	Used as a key to look up schemas. Specify WeatherCurrent and WeatherForecast if you are trying the Business Intelligence at Scale pattern.
DESCRIPTION	A short description of the schema. For example, "Schema for getting the current weather data".
TYPE	The schema format. Select Avro schema provider if you are trying the Business Intelligence at Scale pattern.
SCHEMA GROUP	To group schemas in any logical order. Specify Kafka if you are trying the Business Intelligence at Scale pattern.
COMPATIBILITY	To set the compatibility policy for the schema. Once set, this cannot be changed. Select BACKWARD

Field name	Description and values
	if you are trying the Business Intelligence at Scale pattern.

6. To allow schema to evolve over time by creating multiple versions, select the Evolve option.

   Note: If you deselect the Evolve option, then you can only have one version of the schema.

7. Upload the current and forecast weather schemas that you downloaded as part of the pattern artifacts by clicking Browse in the SCHEMA TEXT area.
8. Click SAVE.

Creating Kafka Topics

For Business Intelligence at Scale, you need to create the following two Kafka Topics: "weather" and "weather_forecast". The "weather" Topic is used for streaming current weather information and the "weather_forecast" Topic is used for the weather forecast information.

Before you begin

Important: You must have an EnvironmentUser role to perform this task.

Procedure

1. Log in to the CDP web interface.
2. Go to Management Console Data Hub Clusters and select the CMM cluster that you created earlier.
3. Click Streams Messaging Manager under the **Services** section.
4. Go to Topics and click Add New.
5. Provide the following required information if you are trying the Business Intelligence at Scale pattern:

Field name	Description and values
TOPIC NAME	A name for your Topic. Specify weather and weather_forecast.
PARTITIONS	Specify 2.
Availability	The level of availability. Select MAXIMUM.
CLEANUP.POLICY	The operation to cleanup the data. Select delete. .

6. Click Save.

Building data pipelines using Cloudera DataFlow

You must deploy a ReadyFlow in the Cloudera DataFlow cluster, develop your flow definition, collect flow information, and deploy Nifi flows.

Adding a ReadyFlow to the Catalog

ReadyFlows are out-of-the-box flow definitions designed to help you get started with Cloudera DataFlow (CDF) quickly and easyily. For the Business Intelligence at Scale pattern, select the "Kafka to S3 Avro" ReadyFlow definition.

About this task

To get started with the data ingest portion of your Business Intelligence at Scale pattern, start by adding the necessary data flows to your CDF Catalog. You must add a custom NiFi data flow (in JSON format) which is available in your pattern artifacts downloads package, and the Kafka to S3 Avro ReadyFlow to your CDF Catalog. The custom NiFi data flow moves data from your streaming data source into the Kafka topics that you have created. The Kafka to S3 Avro ReadyFlow moves data from Kafka into an AWS S3 bucket so that it can be consumed by Cloudera Data Engineering.

Before you begin

Important: You must have the DFCatalogAdmin role to perform this task.

Procedure

1. From the ReadyFlow Gallery page, identify the ReadyFlow you want use.
2. Review the ReadyFlow details by clicking View Details.
3. Click Add To Catalog to add the ReadyFlow into the CDF Catalog and make it ready for deployment.
4. Click Add to confirm that you want to add the ReadyFlow to the Catalog.

Results

The ReadyFlow is added to the CDF Catalog and is ready for deployment.

What to do next

Import the custom NiFi data flow.

Importing the custom NiFi flow definition

Before you can deploy a flow, you must import a flow definition that contains the data flow logic you want to deploy to a Cloudera DataFlow (CDF) environment.

About this task

Import the custom NiFi flow definition available in your pattern artifacts downloads package. This flow definition moves streaming weather data into Kafka.

Before you begin

Important: You must have the DFCatalogAdmin role to perform this task.

Procedure

1. From the CDF left navigation pane, click Catalog.
2. Click Import Flow Definition and provide the following information:

Field name	Description
Flow Name	Specify an unique name for the flow definition you are importing. For example, Weather-data-to-kafka.
Flow Description	(Optional) Provide a description of your flow definition. Use this value to help you differentiate between the other flows imported to your catalog.
NiFi Flow Configuration	Upload the flow definition JSON file.

3. Click Import.

What to do next

Now that you have imported your custom NiFi flow definition, you can move on to collecting the flow configuration that you need to deploy the ReadyFlow and this custom NiFi flow definition for your Business Intelligence at Scale pattern.

Collecting configuration information for ReadyFlow

Learn how to collect the information you need to configure the "Kafka to S3 Avro" ReadyFlow.

The following sections help you to gather configuration information such as Schema Registry host name, Kafka brokers, and other details before you deploy NiFi flows:

For your ReadyFlow data ingest source

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have the Kafka broker end points.
  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in DataFlow.

For your ReadyFlow data ingest target

- You have your S3 path and bucket into which you want to ingest data.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your CDP credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow. For example: kafka-to-s3-avro-ingest

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.

       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your CDP Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Deploying the NiFi flows

Learn about the steps to deploy flows (custom flow definitions or ReadyFlows) so that you can start running your NiFi flows with Cloudera DataFlow.

Before you begin

Important: You must have the DFFlowAdmin role to perform this task.

About this task

Now that you have imported the custom NiFi flow definition and added the ReadyFlow to the catalog you may proceed by deploying your Business Intelligence at Scale data flows. You must deploy the following three flows:

- weather_forecast_to_kafka: This flow gets both current and forecast weather data and sends it to the corresponding Kafka topics.
- Current_Weather_Data_To_S3: This deployment moves current weather data from the Kafka topic into an S3 bucket.
- Forecast_Weather_Data_To_S3: This deployment moves forecast weather data from the Kafka topic into an S3 bucket.

Procedure

1. Log in to the CDP web interface.

2. To create the custom flow deployment "weather_forecast_to_kafka", perform the following steps:
   a) Go to DataFlow Catalog and click the "Kafka to S3 Avro" ReadyFlow definition that you added earlier.
      The version information about the flow is displayed.
   b) Click Deploy #.
   c) Select your CDP environment from the Target Environment drop-down menu on the **New Deployment** dialog and click Continue.
   d) Provide a unique name to your deployment on the **Overview** page.
      For the Business Intelligence at Scale pattern, specify weather_forecast_to_kafka.
   e) Accept the default settings on the **NiFi Configuration** page.
   f) Specify the following on the **Parameters** page:

Field name	**Description and value**
CDP Workload User	Specify your CDP workload username.
CDP Workload User Password	Specify your CDP workload password.
Current Weather Topic Name	Specify weather.
Kafka Brokers	Provide a comma separated list of Kafka FQDNs that you collected from the Data Hub cluster earlier as follows: [***broker1.fqdn***]:9093,[***broker2.fqdn***]:9093,[***broker3.fqdn***]:9093
Schema Name	Specify WeatherForecast. This should match with the schema name that you specified in Schema Registry.
Schema Registry URL	Specify [***master-node-fqdn***]:7790/api/v1 that you collected earlier from the Data Hub cluster.
Topic Name	Specify weather_forecastThis should match with the Kafka topics that you created in Streams Messaging Manager (SMM).
Weather Forecast Topic Name	Specify weather_forecast.
api.key	Specify the API key that you generated on the OpenWeather portal.
api.key.current	Specify the API key that you generated on the OpenWeather portal.

   g) In KPIs, identify key performance indicators, the metrics to track those KPIs, and when and how to receive alerts about the KPI metrics tracking.

      Tip: You can reorder the KPIs by dragging them up or down the list. The order you configure here is reflected in the detailed monitoring view of the resulting deployment.

      You can skip this step for this pattern.
   h) Accept the default settings on the **Sizing & Scaling** page for the Business Intelligence at Scale pattern.
      For production use cases, you can specify your initial flow deployment size, whether to enable auto-scaling, and up to how many nodes you want in your cluster.

      Note: Flow deployments scale horizontally by adding or removing NiFi nodes to a deployment. Deployments do not scale vertically, and the node size cannot be changed.

   i) Review the configuration summary and click Deploy.
      You will be redirected to the Alerts tab in the detail view for the deployment where you can track its progress.

3. To create the "Current_Weather_Data_To_S3" deployment:
   a) Go to DataFlow Catalog and click the "Kafka to S3 Avro" ReadyFlow definition that you added earlier.
      The version information about the flow is displayed.
   b) Click Deploy #.
   c) Select your CDP environment from the Target Environment drop-down menu on the **New Deployment** dialog and click Continue.
   d) Provide a unique name to your deployment on the **Overview** page.
      For the Business Intelligence at Scale pattern, specify Current_Weather_Data_To_S3.
   e) Accept the default settings on the **NiFi Configuration** page.
   f) Specify the following on the **Parameters** page:

Field name	Description and value
CDP Workload User	Specify your CDP workload username.
CDP Workload User Password	Specify your CDP workload password.
CSV Delimiter	Specify , (a comma).
Data Input Format	Specify JSON. Ensure to input this is in All-Caps.
Kafka Broker Endpoint	Provide a comma separated list of Kafka FQDNs that you collected from the Data Hub cluster earlier as follows: [***broker1.fqdn***]:9093,[***broker2.fqdn***]:9093,[***broker3.fqdn***]:9093
Kafka Consumer Group ID	Specify WeatherConsumer.
Kafka Source Topic	Specify weather.
S3 Path	Specify the path to your S3 bucket upto the root.
S3 Bucket	Specify the name of your S3 bucket.
Schema Name	Specify WeatherCurrent. This should match with the schema name that you specified in Schema Registry.
Schema Registry Hostname	Specify [***master-node-fqdn***] that you collected earlier from the Data Hub cluster.

   g) In KPIs, identify key performance indicators, the metrics to track those KPIs, and when and how to receive alerts about the KPI metrics tracking.

      Tip: You can reorder the KPIs by dragging them up or down the list. The order you configure here is reflected in the detailed monitoring view of the resulting deployment.

      You can skip this step for this pattern.
   h) Accept the default settings on the **Sizing & Scaling** page for the Business Intelligence at Scale pattern.
      For production use cases, you can specify your initial flow deployment size, whether to enable auto-scaling, and up to how many nodes you want in your cluster.

      Note: Flow deployments scale horizontally by adding or removing NiFi nodes to a deployment. Deployments do not scale vertically, and the node size cannot be changed.

   i) Review the configuration summary and click Deploy.
      You will be redirected to the Alerts tab in the detail view for the deployment where you can track its progress.

4. To create the "Forecast_Weather_Data_To_S3" deployment:

a) Go to DataFlow Catalog and click the "Kafka to S3 Avro" ReadyFlow definition that you added earlier.

The version information about the flow is displayed.

b) Click Deploy #.

c) Select your CDP environment from the Target Environment drop-down menu on the **New Deployment** dialog and click Continue.

d) Provide a unique name to your deployment on the **Overview** page.

For the Business Intelligence at Scale pattern, specify Forecast_Weather_Data_To_S3.

e) Accept the default settings on the **NiFi Configuration** page.

f) Specify the following on the **Parameters** page:

Field name	**Description and value**
CDP Workload User	Specify your CDP workload username.
CDP Workload User Password	Specify your CDP workload password.
CSV Delimiter	Specify , (a comma).
Data Input Format	Specify JSON. Ensure to input this is in All-Caps.
Kafka Broker Endpoint	Provide a comma separated list of Kafka FQDNs that you collected from the Data Hub cluster earlier as follows: [***broker1.fqdn***]:9093,[***broker2.fqdn***]:9093,[***broker3.fqdn***]:9093
Kafka Consumer Group ID	Specify ForecastConsumer.
Kafka Source Topic	Specify weather_forecast.
S3 Path	Specify the path to your S3 bucket upto the root.
S3 Bucket	Specify the name of your S3 bucket.
Schema Name	Specify WeatherForecast. This should match with the schema name that you specified in Schema Registry.
Schema Registry Hostname	Specify [***master-node-fqdn***] that you collected earlier from the Data Hub cluster.

g) In KPIs, identify key performance indicators, the metrics to track those KPIs, and when and how to receive alerts about the KPI metrics tracking.

Tip: You can reorder the KPIs by dragging them up or down the list. The order you configure here is reflected in the detailed monitoring view of the resulting deployment.

You can skip this step for this pattern.

h) Accept the default settings on the **Sizing & Scaling** page for the Business Intelligence at Scale pattern.

For production use cases, you can specify your initial flow deployment size, whether to enable auto-scaling, and up to how many nodes you want in your cluster.

Note: Flow deployments scale horizontally by adding or removing NiFi nodes to a deployment. Deployments do not scale vertically, and the node size cannot be changed.

i) Review the configuration summary and click Deploy.

You will be redirected to the Alerts tab in the detail view for the deployment where you can track its progress.

Results

When you have deployed all three flows associated with the Business Intelligence at Scale pattern you will have 3 flow deployments, such as:

- Current_Weather_Data_To_S3
- Forecast_Weather_Data_To_S3
- weather_forecast_to_kafka

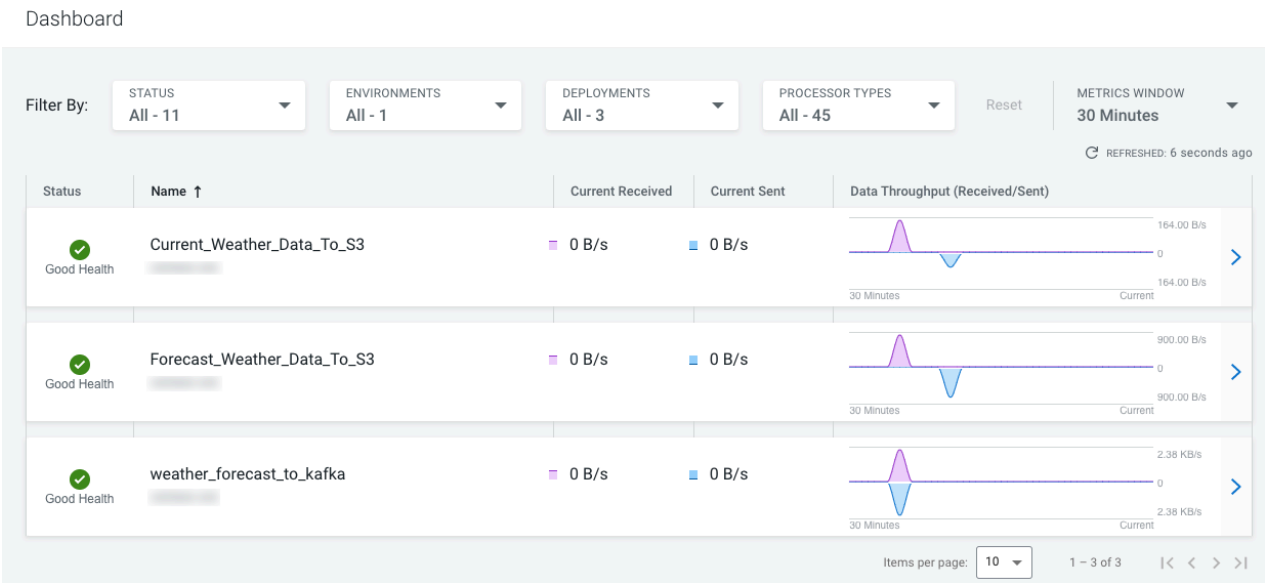

Transforming and organizing data using Cloudera Data Engineering

You must define the Spark job configurations, resources, and schedule in the Cloudera Data Engineering (CDE) cluster.

About this task

In CDE, you perform the organization and transformation piece of the Business Intelligence at Scale pattern. A Spark jar file (bi-workflow_2.11-0.1.jar) has been created for you and is available as part of your pattern artifacts download package to create a Spark job. The Spark job converts the weather data that you put into the AWS S3 bucket in Avro format to Parquet, and makes the Parquet data available to Cloudera Data Warehouse.

Before you begin

Important: You must have the DEUser role to perform this task.

Attention: You must load data into CDW and then run the Spark job. Perform this task after completing About adding data to Data Warehouse and Creating tables and inserting data.

Procedure

1. Log in to your AWS console and create a tmp sub-folder in your S3 bucket within the path that you are using for this pattern.
2. Log in to the CDP web interface.
3. Go to Data Engineering.
4. In the Environments column, select the environment containing the virtual cluster where you want to create the job.

5. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster where you want to create the application.
6. Go to Jobs Create Job .
7. Provide the job details:
   a) Select Spark 2.4.7 as the Job Type.
   b) Specify the Name.
   c) Select File and upload the JAR file that you downloaded earlier.
   d) Specify the following in the Main Class field:

```
com.cloudera.cde.workflow.BiWorkflow
```

   e) Specify the following in the Arguments field:

Argument
--weatherTable
weather
--weatherForecastTable
weather_forecast
--weatherDataDirRoot
s3a://[****FULL-PATH-TO-YOUR-S3-BUCKET****]

   Click ⊕ (Add Argument) to add multiple arguments as necessary.
   weather and weather_forecast sub-folders are automatically created in the S3 bucket by the Spark job.
   f) Enter the following in the Configurations field:

Configuration	**Value**
spark.datasource.hive.warehouse.load.staging.dir	[***PATH-TO-S3-BUCKET***]/tmp
spark.driver.extraJavaOptions	-Duser.timezone=UTC
spark.executor.extraJavaOptions	-Duser.timezone=UTC
spark.sql.hive.hiveserver2.jdbc.url	jdbc:hive2://[****HIVE-SERVER-2-FQDN****]-fsio.int.cldr.work:2181/default;httpPath=cliservice;serviceDiscoveryMode=zooKeeper;ssl=tr Obtain the [****HIVE-SERVER-2-FQDN****] from going to CDP Home Management Console Data Hub Clusters [***DATA-HUB-WITH-DATA-ENGINEERING-CLUSTER-DEFINITION***] Cloudera Manager Clusters Hive service Instances and copying the FQDN corresponding to HiveServer2.
spark.sql.session.timeZone	UTC

   Click ⊕ (Add Configuration) to add multiple configuration parameters as necessary.
   g) Click Schedule to display scheduling options.

8. Schedule the application to run periodically using the Basic controls as follows:
   Every hour at 30 minute(s) past the hour.
   Select a Start Date and an End Date.
9. Click Schedule to create the job.

Analyzing data and visualizing results using Cloudera Data Warehouse

To run queries and draw insights, you must add data to the Data Warehouse, create materialized view, create tables to generate reports and dashboards in Cloudera Data Visualization, and render and share dashboards from Cloudera Data Visualization with your stakeholders.

About adding data to Data Warehouse

To run queries and understand the correlation between the historical sales and historical weather data, you must make this data available in Cloudera Data Warehouse tables.

In a real-world scenario, the historical sales data and weather data could be available in your CDP Data Lake. For the Business Intelligence at Scale pattern, you can download the sample historical sales and weather data set provided by Cloudera from your Business Intelligence at Scale download package on your computer and then upload it into your S3 bucket in individual folders. These two packages contain historical data in Parquet format.

You will fetch this data in Hue directly by specifying the complete path of the S3 directory in the LOCATION clause of a query.

Important: You must have access to the S3 bucket to perform this task.

Creating tables and inserting data

After making the sales and weather data available in S3, you first create external tables, insert data into them, and then create managed tables using the external tables.

Before you begin

Important: You must have the DWUser role to perform this task.

Procedure

1. Log in to the CDP web interface.
2. Go to Data Warehouse.
3. Locate your Impala Virtual Warehouse and click on Hue.
4. Open the Unified Analytics editor from the left assist panel.
5. Run the following query to create an external table "sales":

```
CREATE EXTERNAL TABLE `sales_external`(
  `store_nbr` bigint,
  `item_nbr` bigint,
  `local_date` date,
  `city_name` string,
  `item_name` string,
  `units` double)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT
```

```
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3a://[***COMPLETE-PATH-OF-S3-BUCKET***]/bi-at-scale-historic-sales-d
ata/';
```

bi-at-scale-historic-sales-data is the name of the directory containing Parquet files that you uploaded to S3.

6. Verify that the historical sales data is available by running the following query:

```
SELECT count(*) FROM sales_external;
```

7. Run the following query to create a managed table that can receive sales data from the external table:

```
CREATE TABLE sales (
    store_nbr bigint,
    item_nbr bigint,
    local_date date,
    item_name string,
    units double
                    )
PARTITIONED BY (city_name string);
```

8. Run the following query to insert data into the managed "sales" table:

```
INSERT INTO sales PARTITION (city_name)
SELECT  store_nbr,
        item_nbr,
        local_date,
        item_name,
        units,
        city_name
FROM sales_external;
```

9. Verify that the historical sales data is available by running the following query:

```
SELECT count(*) FROM sales;
```

10. Run the following query to create an external table "weather_hist_external":

```
CREATE external TABLE weather_hist_external(
  city string,
  local_date date,
  avg_temperature DOUBLE,
  min_temperature DOUBLE,
  max_temperature DOUBLE,
  avg_feels_like_temperature DOUBLE,
  min_feels_like_temperature DOUBLE,
  max_feels_like_temperature DOUBLE,
  avg_humidity DOUBLE,
  min_humidity DOUBLE,
  max_humidity DOUBLE,
  avg_pressure DOUBLE,
  min_pressure DOUBLE,
  max_pressure DOUBLE,
  avg_wind_speed DOUBLE,
  min_wind_speed DOUBLE,
  max_wind_speed DOUBLE,
  avg_snow_1h DOUBLE,
  min_snow_1h DOUBLE,
  max_snow_1h DOUBLE,
  avg_rain_1h DOUBLE,
  min_rain_1h DOUBLE,
  max_rain_1h DOUBLE,
```

```
  creation_timestamp timestamp
  )
  location 's3a://[***COMPLETE-PATH-OF-S3-BUCKET***]/bi-at-scale-historic-
weather-data/';
```

bi-at-scale-historic-weather-data is the name of the directory containing Parquet files that you uploaded to S3.

11. Verify that the historical weather data is available by running the following query:

```
SELECT count(*) FROM weather_hist_external;
```

12. Run the following query to insert data into the "weather_forecast" table:

```
INSERT INTO weather_forecast --PARTITION (city) -- CHECK FOR PARTITION E
RROR ----
SELECT  local_date,
        avg_temperature as avg_temp_k,
        min_temperature as min_temp_k,
        max_temperature as max_temp_k,
        (avg_temperature - 273.15) * 9/5 + 32 as avg_temp_f,
        (min_temperature - 273.15) * 9/5 + 32 as min_temp_f,
        (max_temperature - 273.15) * 9/5 + 32 as max_temp_f,
        avg_feels_like_temperature as avg_feels_like_temp_k,
        min_feels_like_temperature as min_feels_like_temp_k,
        max_feels_like_temperature as max_feels_like_temp_k,
        (avg_feels_like_temperature - 273.15) * 9/5 + 32 as avg_feels_li
ke_temp_f,
        (min_feels_like_temperature - 273.15) * 9/5 + 32 as min_feels_li
ke_temp_f,
        (max_feels_like_temperature - 273.15) * 9/5 + 32 as max_feels_li
ke_temp_f,
        avg_humidity,
        min_humidity,
        max_humidity,
        avg_pressure,
        min_pressure,
        max_pressure,
        avg_wind_speed,
        min_wind_speed,
        max_wind_speed,
        avg_snow_1h,
        min_snow_1h,
        max_snow_1h,
        avg_rain_1h,
        min_rain_1h,
        max_rain_1h,
        creation_timestamp,
        city
FROM    weather_hist_external;
```

Note: "weather" and "weather_forecast" tables are created by the Spark job.

13. Verify that the forecast weather data is available by running the following query:

```
SELECT count(*) FROM weather_forecast;
```

Creating a materialized view

After creating managed tables and inserting the data, you can join the tables using a materialized view. You run the query, and the optimizer takes advantage of the precomputation performed by the materialized view for a faster response time.

Before you begin

Important: You must have the DWUser role to perform this task.

Procedure

1. Log in to the CDP web interface.
2. Go to Data Warehouse.
3. Locate your Impala Virtual Warehouse and click on Hue.
4. Open the Unified Analytics editor from the left assist panel.
5. Run the following query to create materialized view:

```
CREATE MATERIALIZED VIEW sales_weather_join
STORED AS PARQUET AS
SELECT  w.local_date,
        w.min_temp_f,
        w.min_temp_k,
        w.max_temp_f,
        w.max_temp_k,
        w.avg_temp_k,
        w.avg_temp_f,
        w.min_rain_1h,
        w.max_rain_1h,
        w.avg_rain_1h,
        w.min_snow_1h,
        w.max_snow_1h,
        w.avg_snow_1h,
        w.city,
        s.item_name,
        s.units,
        s.city_name AS city_name
FROM    weather_forecast w INNER JOIN sales s
ON      w.city = s.city_name
AND     w.local_date = s.local_date;

ANALYZE TABLE weather COMPUTE STATISTICS;

ANALYZE TABLE weather_forecast COMPUTE STATISTICS;

ANALYZE TABLE sales COMPUTE STATISTICS;

ALTER MATERIALIZED VIEW sales_weather_join REBUILD;
```

Creating a Data Visualization dashboard by importing visual artifacts

To visualize the correlation and share the insights with your stakeholders, you can manually create a dashboard in Cloudera Data Visualization (CDV) or use a visual artifact in the form of a JSON file and import it in CDV.

About this task

For the Business Intelligence at Scale pattern, use the "cdv-dashboard-Effect-of-Weather-on-Sales.json" script provided as part of the download package to create the tables required for generating dashboard.

Before you begin

Important: You must have the DWUser role to perform this task.

Procedure

1. Log in to the CDP web interface.
2. Go to Data Warehouse.
3. Locate your Impala Virtual Warehouse and click on Data VIZ.
4. Click DATA on the main navigation bar.

   All Connections and the available Datasets are displayed.
5. Click on Default Impala VW from the left navigation pane.
6. Click ••• at the top of the page and click Import Visual Artifacts.
7. On the **Import Visual Artifacts** pop-up, click Choose File and select the file you want to import.
8. Specify the destination workspace in the Import to Workspace drop-down menu.

   By default, Data Visualization imports the visual artifacts into your “Private” workspace. However, you can choose the “Public” workspace, or another workspace.
9. Select Import from the file from the Thumbnail Action drop-down menu.
10. Click IMPORT and confirm by clicking ACCEPT AND IMPORT.

Results

After the import completes, a success message appears on the Import interface. All the artifacts have an assigned ID, which are generated by the system, sequentially. Visuals/Dashboards and datasets have separate ID queues.

Example

Completing this task for your Business Intelligence at Scale pattern results in a dashboard that looks similar to the following example:

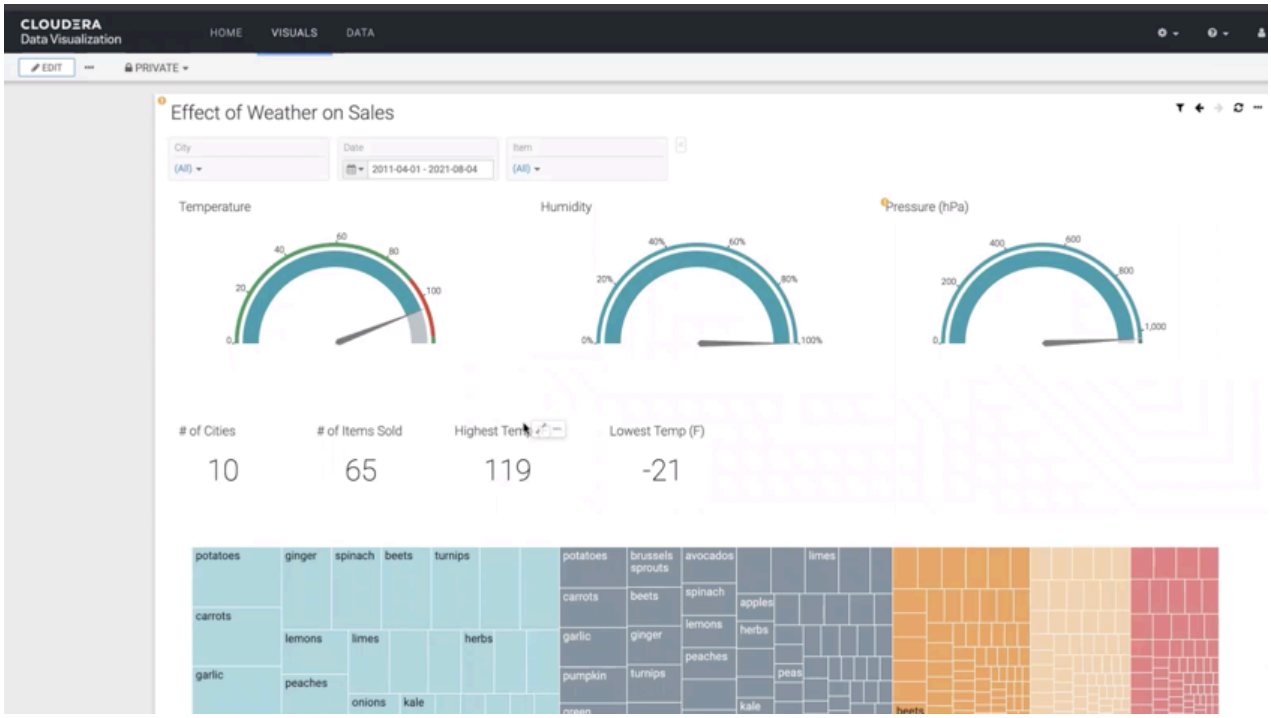

What to do next

Now that you have created your Business Intelligence at Scale pattern dashboard, share it with other stakeholders.

1. Open your dashboard from the VISUAL interface.
2. Click the more option and select Get URL.
3. Copy the URL by right-clicking it and share it with your stakeholders.

Exploratory Data Science

Date published: 2022-06-10
Date modified: 2022-06-15

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Exploratory Data Science **4**

Data Discovery and Exploration Steps **6**

Meeting the prerequisites 6
About sample data 7
Starting Exploratory Data Science and Visualization 8
Creating a Flight Details dashboard 9

Troubleshooting **11**

Troubleshooting: 401 Unauthorized 11
Troubleshooting: 401 Unauthorized when accessing Hive 11
Troubleshooting: Existing connection name 12
Troubleshooting: Empty data page 12
Troubleshooting: Some connections not shown 12

Exploratory Data Science

The Exploratory Data Science pattern shows you how a data scientist or other data practitioners can kickstart a data science project. In this pattern, you discover the data available in your CDP environment, explore that data using SQL queries, and then set up some data visualizations to better understand the data — all of this from within the Cloudera Machine Learning (CML) interface.

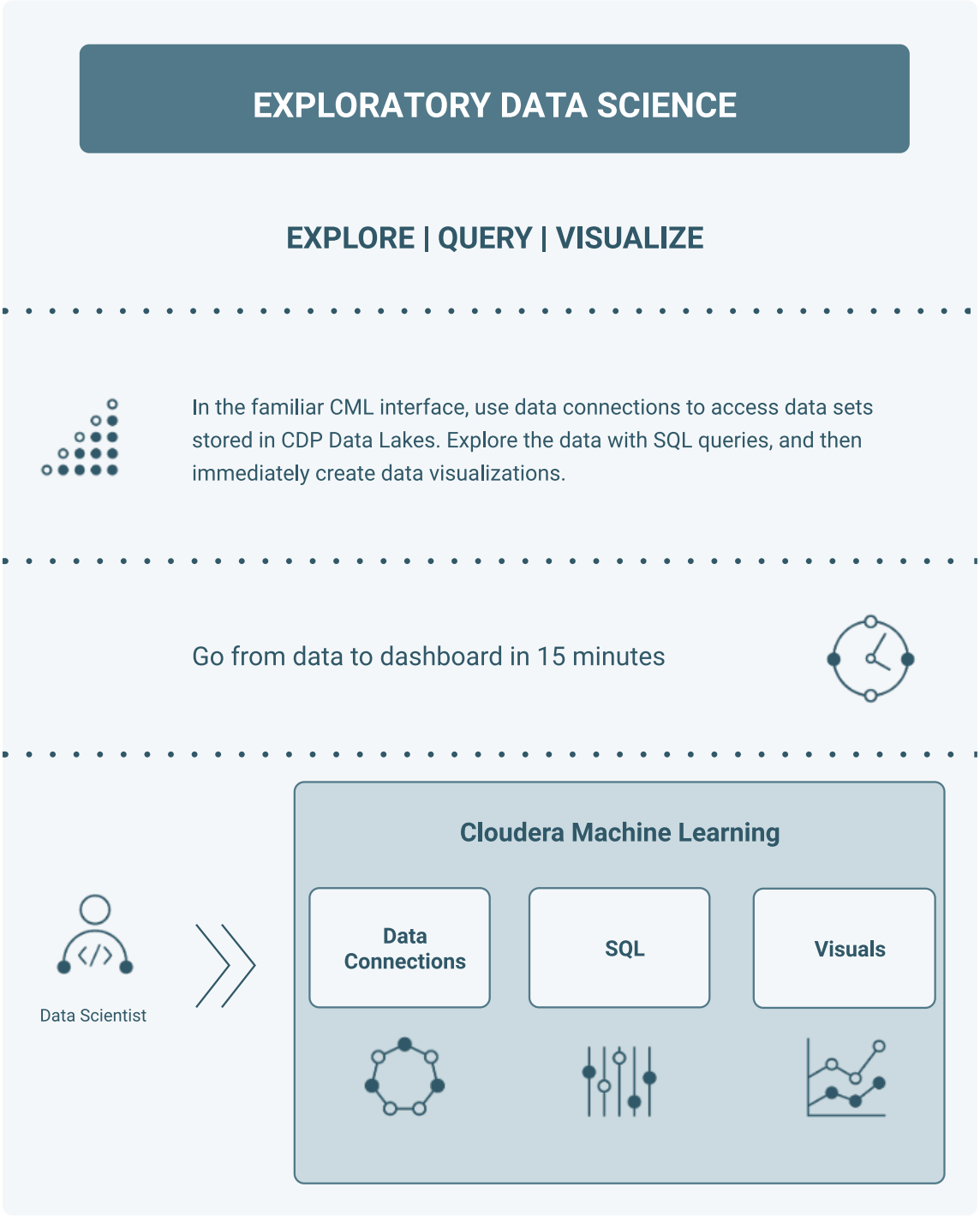

We will show you how easy it is to find and explore data in CML.

Data Discovery and Exploration Steps

To follow this CDP pattern, you must have access to CDP Public Cloud and your IT team must have set up the Cloudera Data Warehouse and Cloudera Machine Learning services with other necessary prerequisites listed in the *Meeting the prerequisites* section. You will use sample data that is provided in the Virtual Warehouse.

Related Information

Meeting the prerequisites

Meeting the prerequisites

Before the Data Analysts can explore and query data, your central or departmental IT must have onboarded to CDP Public Cloud and must meet the requirements listed in this section.

To do in the CDP Management Console

1. Make sure that the Cloudera Data Lake is created and running. For more information, see Creating an AWS environment with a medium duty data lake using the CLI.
2. Grant EnvironmentUser role to the Data Analyst user and synchronize the user to FreeIPA. For more information, see Assigning account roles to users and Performing user sync.

To do in Cloudera Data Warehouse (CDW)

1. Make sure that the CDW environment is activated. For more information, see Activating AWS environments.
2. Add an S3 bucket as an external bucket in the CDW environment with read-only access.

   Important: You must add your own S3 bucket to the CDW environment.

   a. Go to CDW service Environments and click its edit icon.
   b. On the Environment Details page, type the name of the AWS bucket you want to configure access to in the Add External S3 Bucket text box and select read-only access mode.
   c. Click Add Bucket and then click Apply to update the CDW environment.
3. Create a non-default Database Catalog with the Load Demo Data option enabled.
   a. Go to CDW service Database Catalogs and click Add New.
   b. Specify a name, select an environment, select SDX from the Datalake drop-down list and enable the Load Demo Data option, and click CREATE.
4. Create an Impala-based Virtual Warehouse with the Enable Data Visualization option and make sure it is in the running state.
   a. Go to CDW service Virtual Warehouses and click Add New.
   b. Select the type as IMPALA, select the Database Catalog associated with the Virtual Warehouse, select the size, click Enable Data Visualization, and then click CREATE.
5. Enable the S3 File Browser for Hue.
   a. Go to CDW service Virtual Warehouses , select your Virtual Warehouse and click edit.
   b. On the **Virtual Warehouses** page, go to CONFIGURATIONS Hue and select hue-safety-valve from the drop-down list.
   c. Add the following configuration for Hive or Impala Virtual Warehouse in the space provided and click APPLY:

```
[desktop]
# Remove the file browser from the blocked list of apps.
```

```
# Tweak the app_blacklist property to suit your app configuration.
app_blacklist=spark,zookeeper,hive,hbase,search,oozie,jobsub,pig,sqoop
,security

[aws]
[[aws_accounts]]
[[[default]]]
access_key_id=[***AWS-ACCESS-KEY***]
secret_access_key=[***SECRET-ACCESS-KEY***]
region=[***AWS-REGION***]
```

6. Enable a link to the Data VIZ application (Cloudera Data Visualization) from the Hue web interface or provide the Data VIZ application URL to the Data Analyst.
   a. Go to CDW service Virtual Warehouses , select your Virtual Warehouse and click edit.
   b. On the Virtual Warehouses details page, go to CONFIGURATIONS Hue and select hue-safety-valve from the drop-down list.
   c. Add the following lines in the safety valve and click APPLY:

```
[desktop]
  custom_dashboard_url=[***DATA-VIZ-URL***]
```

To do in Ranger

Grant the required DDL and DML Hadoop SQL policies to the Data Analyst user in Ranger.

1. Go to CDW service Database Catalogs , click the more option, and click Open Ranger.
2. On the **Ranger Service Manager** page, click Hadoop SQL.
3. Select the all - url policy.

   The **Edit Policy** page is displayed.
4. Under the Add Conditions section, add the users under the Select User column and add permissions such as Create, Alter, Drop, Select, and so on from the Permissions column.
5. Scroll to the bottom of the page and click Save.

Other prerequisites

- Data in a file to be ingested is in a structured format, such as CSV, with headers.
- The data file is available to the Data Analyst on their computer or accessible on a shared drive with permissions already granted.
- Hue application URL is shared with the Data Analyst, if a link to the Data Visualization application is not enabled in Hue.

Related Information

Data Discovery and Exploration Steps

About sample data

For this pattern, we use a dataset about airlines and flights. This dataset is available in the Virtual Warehouse when you create a non-default Database Catalog with the Load Demo Data option enabled.

Airlines data (available in CDP)

The airline data consists of the following four tables:

- airlines
- airports
- flights

- planes

The airline data looks as follows:

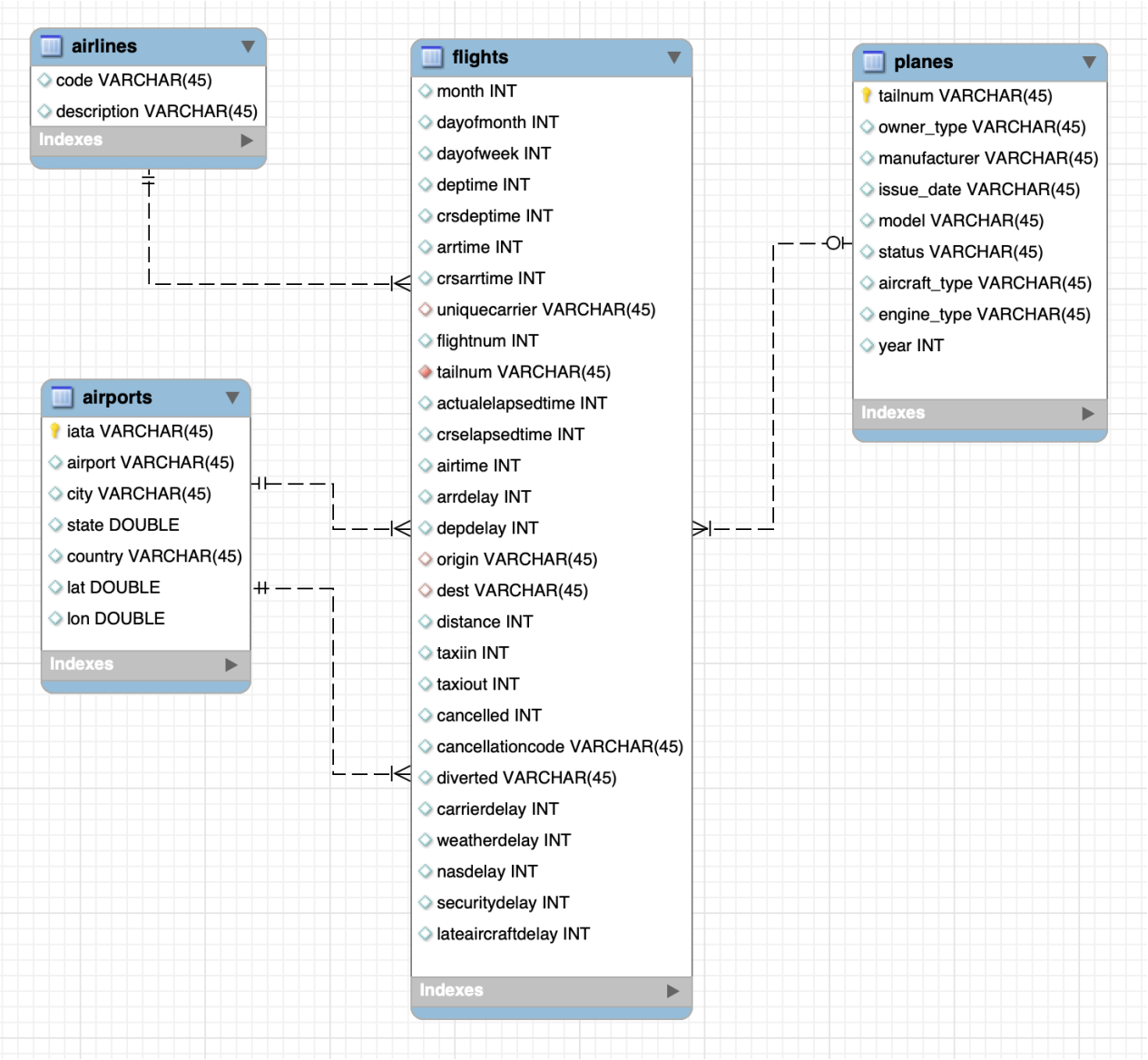

Starting Exploratory Data Science and Visualization

You can start the Exploratory Data Science and Visualization application within Cloudera Machine Learning (CML).

1. Create or open a project.
2. In the project, click Data in the left navigation pane.
3. If the application was previously started, it appears in the UI.
4. If not, click Launch Data Application to start the application. It takes a few minutes to start the first time.

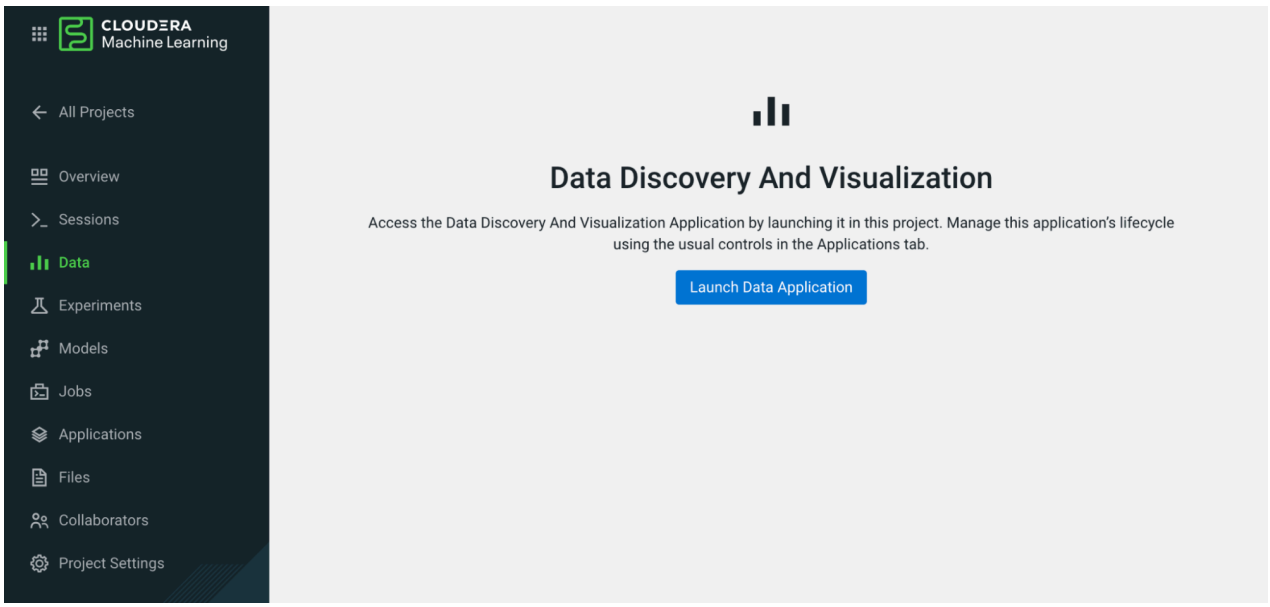

When the application starts, you can see four tabs at the top of the UI. From here, you can follow the steps in Get Started, or try some of the following tasks.

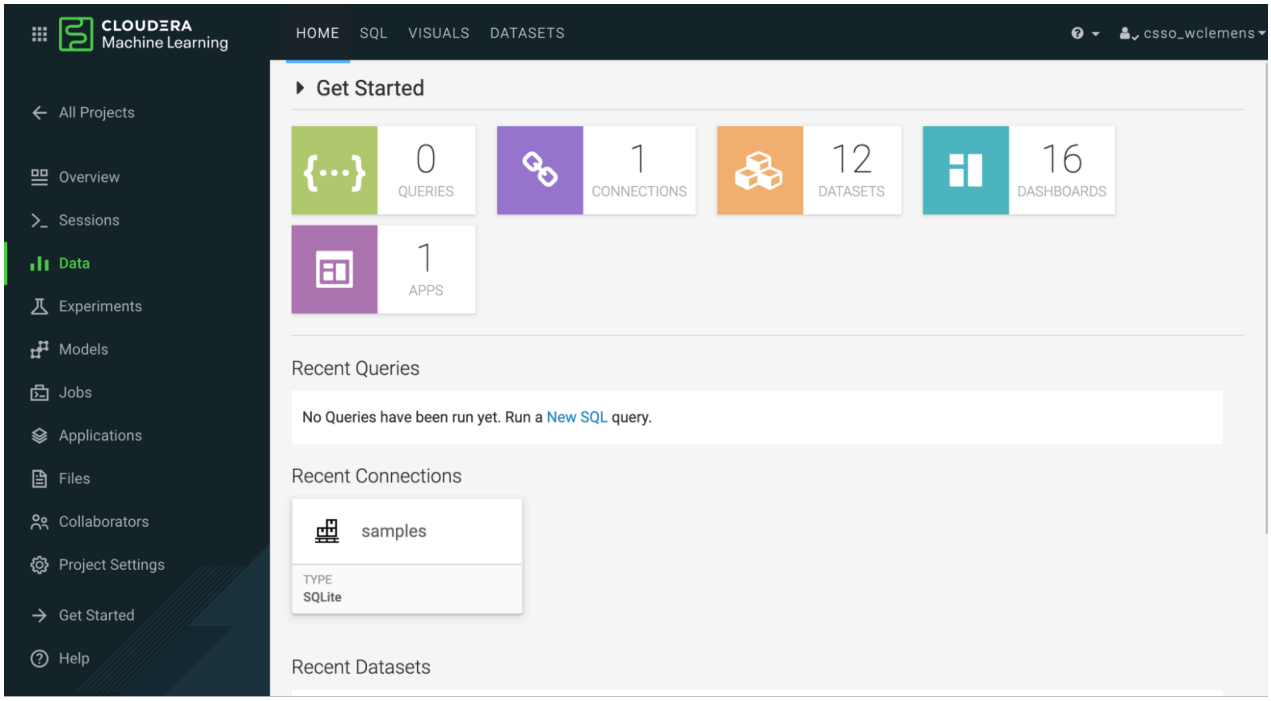

Creating a Flight Details dashboard

In this exercise, you will create a dashboard to show information about airlines and flight cancellations. You can create this dashboard using the Data Discovery and Visualization application integrated into Cloudera Machine Learning.

Procedure

1. In Data, select the SQL tab, then select the airline data connection.
2. In Database, select airline_ontime_parquet.

3. In Enter SQL below, enter this SQL statement:

```
select * from airline_ontime_parquet.flights
```

   Enter the table name by clicking flights in the Tables list. The database name is automatically prepended in the SQL statement.
4. Click Run.

   The SQL statement executes, and the results appear in the Results tab.
5. Click New Dashboard.

   The Visuals tab opens.
6. In the Visuals menu, select the Table graphic type.
7. In the Dimensions menu, select uniquecarrier and drag it to the Dimensions shelf.
8. In the Measures menu, select Record Count and drag it to the Measures shelf.
9. In the Dimensions menu, select cancelled and drag it to the Measures shelf.

   It appears as sum(cancelled).
10. On this sum(cancelled) item, in the Field Properties, click Alias, and rename the item to Cancel Count.
11. Drag the cancelled item again from the Dimensions menu to the Measures shelf.
12. For the second item, click the right arrow next to the label, and select Enter/Edit Expression.

    The expression displays in an edit pane.
13. Change the expression to the following:

```
sum([cancelled])/count(1)*100 as "Cancel Percent"
```

14. Click Save.

    Note that the name of the item is changed to "Cancel Percent".
15. Click on Record Count, then Alias, and change the name to Flight Count.
16. Click on uniquecarrier, then Alias, and change it to Carrier.
17. Click Refresh Visual.

    The column names in the visual area are updated.
18. Click ⋮ and select Clone.

    The table is copied to a new area in the visual designer.
19. In the new table, click Configure ⚙.
20. In the Visuals menu, select Bars to create a bar chart. The table transforms into a bar chart.
21. Move items around on the shelves so that they are arranged as shown:
    - X Axis: Carrier
    - Y Axis: Cancel Percent
    - Tooltips: Cancel Count, Flight Count
22. Click Refresh Visual.

    When you move the pointer onto a bar, tooltips showing data for that bar appear.
23. Click Cancel Percent, and in Field Properties, click Order and Top K.
24. Select Descending.
25. Click Refresh Visual.

    The bars are reordered from largest to smallest.
26. In the Action menu on the graphic pane, select Edit Dataset SQL. The SQL editor page opens.
27. Click on the Dataset name field, and rename it to Flight Details. Click Apply, then click Apply again in the modal window that appears.

Results

This completes the Flight Details dashboard.

Troubleshooting

You may encounter the following issues.

Troubleshooting: 401 Unauthorized

Problem: Session returns a 401: Unauthorized error.

When you are in a session and try to run the code for a Hive or Impala connection, the session returns a 401: Unauthorized HTTP error.

```
Modify next 2 lines to update your credentials

> USERNAME = os.getenv('HADOOP_USER_NAME')
> PASSWORD = os.getenv('WORKLOAD_PASSWORD')
> conn = cmldata.getConnection({
          'CONNECTION_NAME': CONNECTION_NAME,
          'USERNAME': USERNAME,
          'PASSWORD': PASSWORD
        })
HttpError: HTTP code 401: Unauthorized
HttpError                                 Traceback (most recent call last)
<ipython-input-1-a69c5e2af83b> in <module>
      2          'CONNECTION_NAME': CONNECTION_NAME,
      3          'USERNAME': USERNAME,
----> 4          'PASSWORD': PASSWORD
      5        })

~/.local/lib/python3.7/site-packages/cmldata.py in getConnection(properties)
    105     fmt_codesnippet = _getConnectionSnippet(properties)
    106     scope = {}
--> 107     exec(fmt_codesnippet, scope)
    108     return scope["conn"]

<string> in <module>

<string> in getCursor(self)

/usr/local/lib/python3.7/site-packages/impala/hiveserver2.py in cursor(self, user, configuration, convert_type
    127         log.debug('_cursor(): getting new session handle')
>
```

Solution: You need to set your workload password. Ensure you are using the correct credentials.

Troubleshooting: 401 Unauthorized when accessing Hive

Problem: Session returns a 401: Unauthorized error when you are accessing a Hive database.

Check that the Hive data warehouse is working.

Solution: Follow these steps to check that Hive data warehouse is running, and restart if needed.

1. In the control plane, go to Data Warehouses.
2. Click Virtual Warehouses, and select the data warehouse for the connection.
3. Check the status of that data warehouse, and make sure that it is running or in a good state.

Troubleshooting: Existing connection name

Problem: When the user attempts to sync data connections, an error message displays, stating the crn or name is a duplicate.

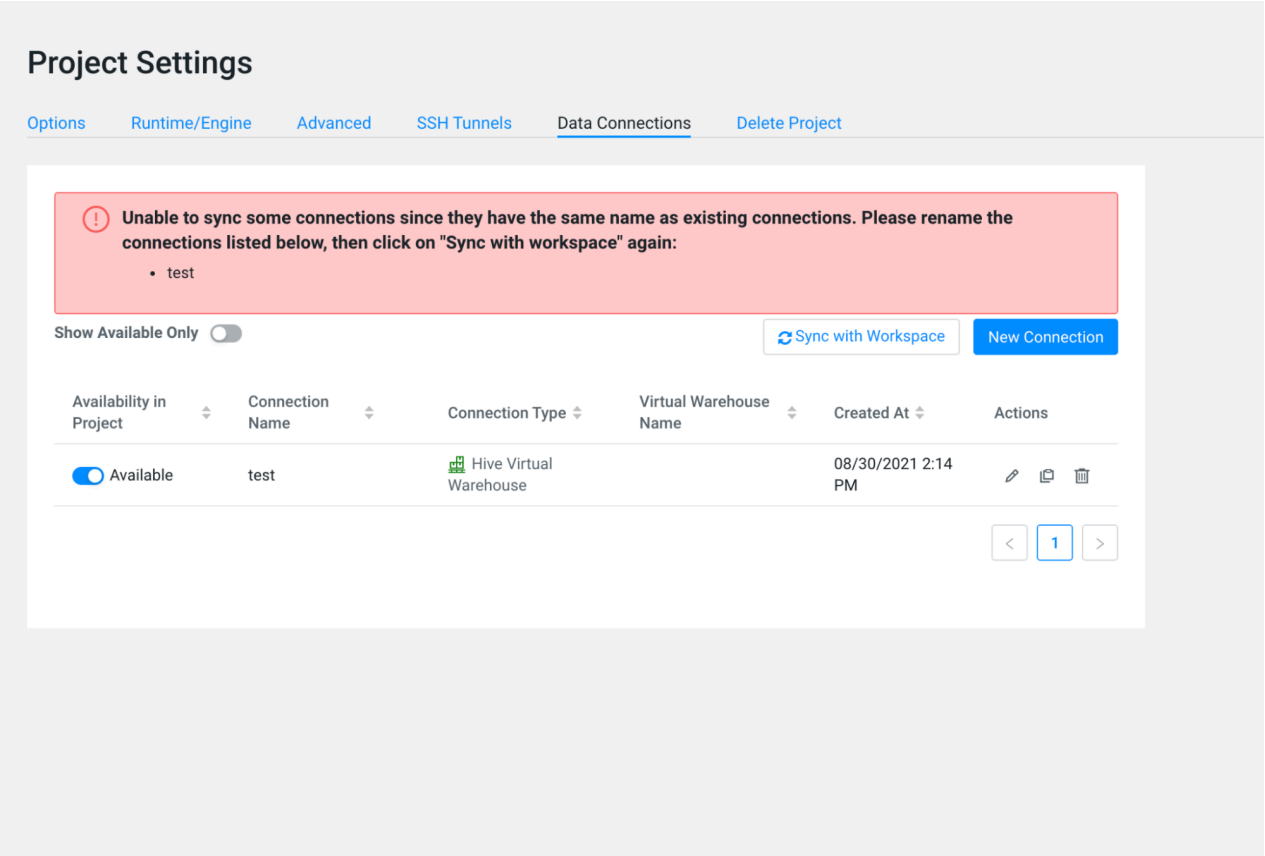

Solution: This indicates a project connection (one that is not copied from the workspace) has the same name or crn as a workspace connection. To resolve this, you need to change the name or crn of the data connection at the project level.

Troubleshooting: Empty data page

You launch the application and see the spinner, but then the empty data page returns.

On the applications tab, click Data Discovery and Visualization application, and check the logs. Check if another user stopped or restarted the application.

On the applications tab, click on Settings Resource Profile , and check if the available resources are sufficient. You can increase the resources if necessary, then restart the application.

Troubleshooting: Some connections not shown

The data application doesn't show all data connections.

See the error message to fix the workload password. To set the workload password, follow the procedure in *Setting the workload password*. You will need to restart the data application after setting the workload password.

Alternatively, you may need to click Sync under the Data Connections to get all of the data connections available at the project level.

AWS Graviton instances in Cloudera Operational Database (Preview)

Date published: 2024-12-11
Date modified: 2025-01-18

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2
Contents 3
Overview 4
Prerequisites for using AWS Graviton in COD 4
Considerations for using AWS Graviton in your cluster 4
Creating AWS Graviton-based COD clusters 5

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Overview

AWS Graviton is a general-purpose, ARM-based processor family that currently delivers the best price-to-performance ratio for cloud workloads running in AWS Elastic Compute Cloud (EC2). With AWS Graviton, you can optimize costs.

Prerequisites for using AWS Graviton in COD

- To use this feature, you must have the following entitlements:
    - COD_USE_GRAVITON
    - CDP_AWS_ARM_DATAHUB
    - COD_USE_I8G_INSTANCE_TYPE
    - COD_USE_CUSTOM_INSTANCE_TYPES
- You must be logged into the COD as an **ODAdmin**.
- Verify if the selected instance types are available in your preferred regions

Considerations for using AWS Graviton in your cluster

Important:

This feature is under technical preview. Technical previews are considered under development. Do not use these features in production environments.

- You can create the Graviton-based COD clusters only on the AWS environments.
- The supported storage type is S3 with ephemeral cache enabled.
- The supported scale type is HEAVY.
- AWS is still rolling out Graviton, this feature is only available in some regions.
- Deploying COD on Graviton might take longer based on the cluster shape and size.
- To use COD on Graviton, you need to install beta CDP CLI. For more information, see the Installing beta CDP CLI documentation.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Creating AWS Graviton-based COD clusters

You can create AWS Graviton-based COD clusters using the Cloudera Operational Database UI or CLI command.

Using the Cloudera Operational Database UI:

1. In the COD web interface, click **Create Database**.
2. Specify the location of the database where you want to store it.
    a. Provide a name for the database in the **Database Name** field.
    b. Select the CDP environment from the list in which you want to associate the database. The environment does not have to be Graviton-based; it has to use a CDP version that supports COD on Graviton.
    c. Click **Next**.
    If an environment does not exist, you can create one by clicking **Create New Environment**.
    For more information, see *Register your first environment*.
3. Select ARM64 as the **Architecture** from the drop-down list.
4. Commission your database by defining a scale for your database using a predefined Data Lake template.
    Select **Heavy Duty**.
    The template helps you to structure your database automatically thereby saving your time and cost. COD creates the predefined number of LITE or HEAVY gateway and master nodes, a set of worker nodes, and also adds additional functionalists into the new database. In case you need to modify the default number of nodes defined in the template, you can do so after the database creation.
    The available templates are Micro Duty, Light Duty, and Heavy Duty. By default, Light Duty is selected.
5. Configure your database by selecting the storage type as **Cloud Storage with Caching**.

    The storage type Cloud Storage with Caching is equivalent to using `--storage-type CLOUD_WITH_EPHEMERAL` option on CDP CLI while creating an operational database.
6. Check or update the settings for your database.
    a. Check all the default settings for your database under the **Default** tab.
    b. Go to the **Advanced** tab to modify any of the default values.

        - The HDFS Volume Type option appears under the Advanced tab only if you select HDFS as the storage type in the Configuration step.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

    - If you disable the Autoscaling option using the Advanced tab, the Worker Nodes and Compute Nodes options are hidden. Instead, a Node Count option appears.
  c. The minimum and maximum number of worker nodes vary for different storage types.

    - Micro duty: Minimum node count: 1. Maximum node count: 5.
    - Light duty: Minimum node count: 3. Maximum node count: 100.
    - Heavy duty: Minimum node count: 3. Maximum node count: 800.

7. Review the details before creating the database.
   Click **Show CLI Command** to get the complete command details corresponding to your settings. You can use it to create the database using CDP CLI.
   Alternatively, you can use the following sample command to create the database using CDP CLI.

```
cdp opdb create-database --environment-name cod-731
--database-name test --architecture ARM64 --scale-type HEAVY
--storage-type CLOUD_WITH_EPHEMERAL --auto-scaling-parameters
'{"minWorkersForDatabase":5, "maxWorkersForDatabase":100}'
--num-edge-nodes 0
```

8. Click **Create Database**.

Result

An information page shows the database's status. Once it is available, your new database is ready to use.

Using the beta CLI command:

1. Launch the beta CDP CLI tool.
2. Run the following command to create Graviton-based COD clusters.

```
cdp opdb create-database --environment-name <env_name>
--database-name <database_name> --architecture <string>
--storage-type CLOUD_WITH_EPHEMERAL --scale-type HEAVY
```

   For example,

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
cdp opdb create-database --environment-name aws_test
--database-name cod_7214 --architecture ARM64 --storage-type
CLOUD_WITH_EPHEMERAL --scale-type HEAVY
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Azure VM Encryption at Host (Preview)

Date published: 2022-06-06
Date modified: 2024-12-04

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2
Contents 3
Introduction 4
Limitations 4
Prerequisites 4
Enable encryption at host for an environment 5
Enable encryption at host for a Cloudera Data Hub cluster 8

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Introduction

You can optionally enable encryption at host for Data Lake, FreeIPA, and Cloudera Data Hub clusters. Currently, you need to enable it individually for each Virtual Machine (VM) on Azure Portal.

As described in Azure documentation, when you enable encryption at host, the encryption starts on the VM host, where the data for your temporary disk, and OS and data disk caches are stored. After enabling encryption at host, all this data is encrypted at rest and flows encrypted to the Storage service, where it is persisted. Thus, encryption at host essentially encrypts your data from end to end.

Temporary disks and ephemeral OS disks are encrypted at rest with platform-managed keys when you enable end-to-end encryption. The OS and data disk caches are encrypted at rest with either customer-managed keys (CMK) or platform-managed keys, depending on the selected disk encryption type. For example, if a disk is encrypted with customer-managed keys, then the cache for the disk is encrypted with customer-managed keys, and if a disk is encrypted with platform-managed keys then the cache for the disk is encrypted with platform-managed keys.

Encryption at host does not use your VM's CPU and doesn't impact your VM's performance.

Related links
Encryption at host - End-to-end encryption for your VM data

Limitations

When using Azure VM encryption at host with Cloudera, the following limitations apply:

- Even if you wish to use the feature for a single subscription only, you need to enable encryption at host for all subscriptions within your Azure tenant.
- This feature can currently be configured individually for each Data Lake and FreeIPA node. Additionally, encryption at host for Data Hub nodes can be configured individually for each Cloudera Data Hub node. The configuration must be performed on the Azure Portal.

Prerequisites

Prior to enabling encryption at host in Cloudera, meet the following Azure prerequisites:

1. Enable encryption at host, as described in Use the Azure CLI to enable end-to-end encryption using encryption at host: Prerequisites in Azure documentation.
   Note: You need to enable this for each subscription within your Azure tenant.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

2. If you would like to use Azure disk encryption with a customer-managed key (CMK) along with encryption at host, meet the prerequisites mentioned in Customer managed encryption keys.

Enable encryption at host for an environment

Use these steps to enable encryption at host for an existing Cloudera environment running on Azure. The steps involve manually enabling encryption at host for each Data Lake and FreeIPA VM via the Azure Portal.

Steps

1. In the Cloudera Management Console, select **Environments** and then click on the specific environment.
2. Make sure that your Cloudera environment is running.
3. Click on the **Data Lake** tab and then navigate to the **Hardware** tab. Here you can find the list of all Data Lake VMs organized into host groups:

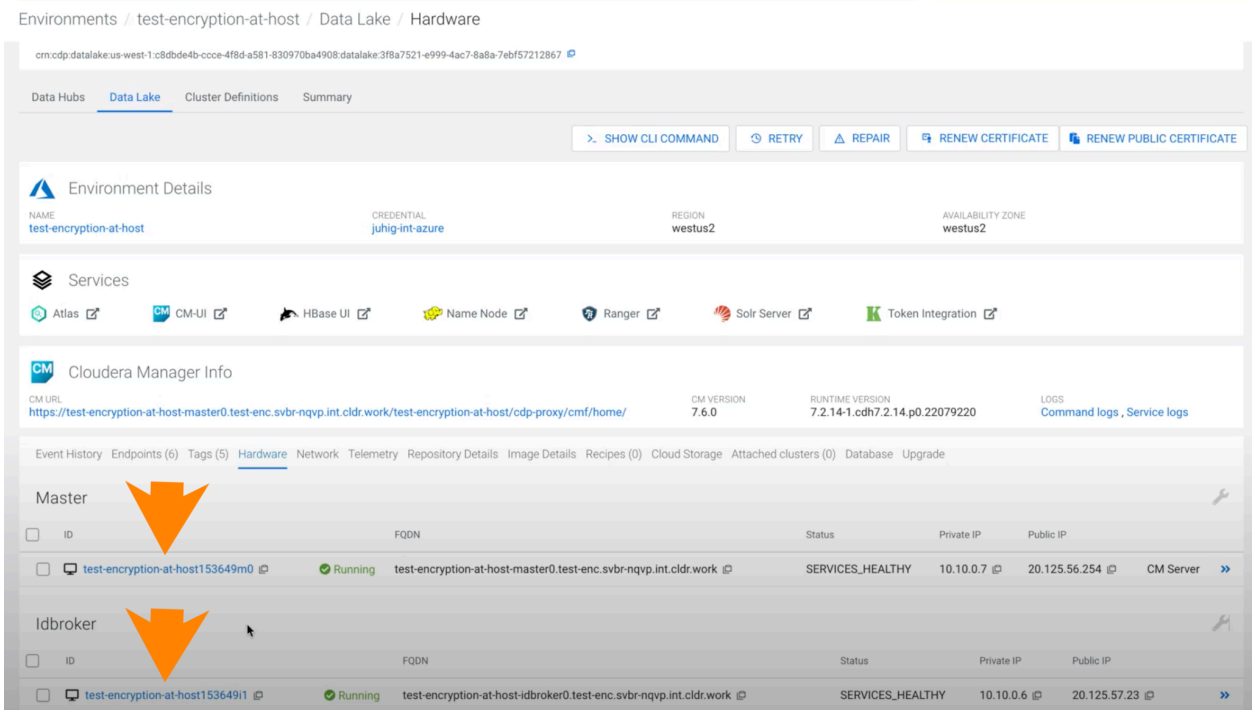

4. Click on each of the VM links and a new browser tab will open for each, redirecting you to the Azure Portal. You need to do this individually for each VM.
5. For each of the VMs, navigate to the **Disks** section in Azure Portal. It will show the “Encryption at Host” as “disabled”:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

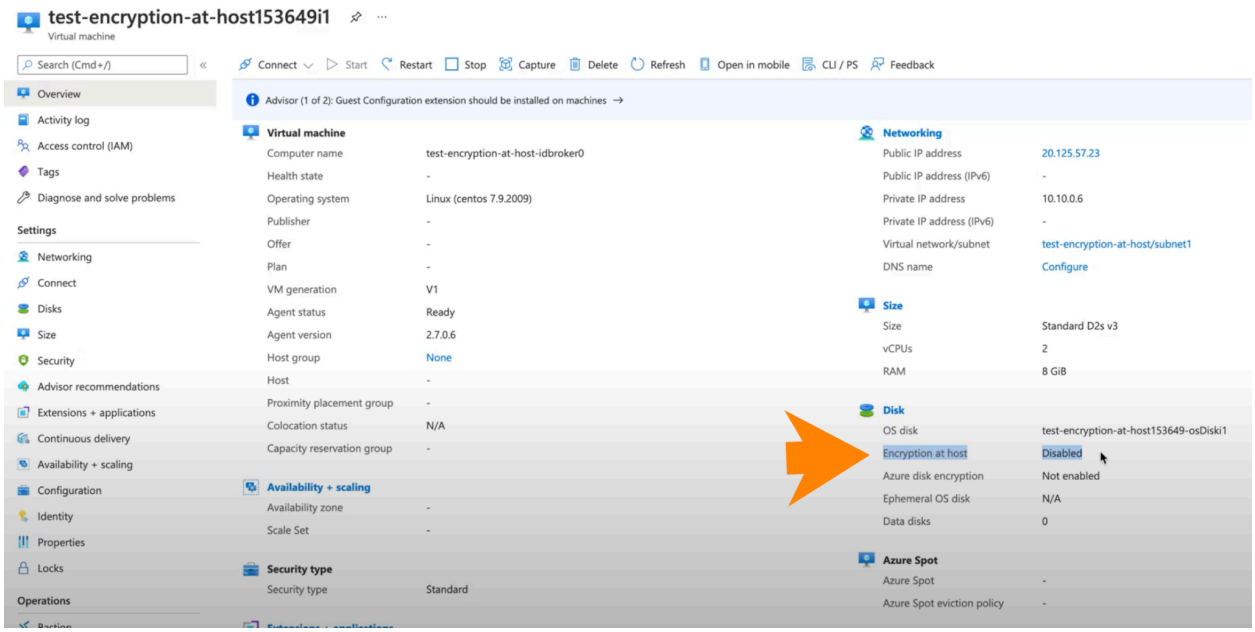

Leave the Azure Portal browser tabs open. You will need to get back to them shortly.

6. Navigate back to the Cloudera Management Console to repeat the same steps for FreeIPA VMs. To access FreeIPA VMs, navigate to environment details and click on the **Summary** tab. Next, scroll down to the **FreeIPA** tile. Here you can find the list of all FreeIPA VMs organized into host groups:

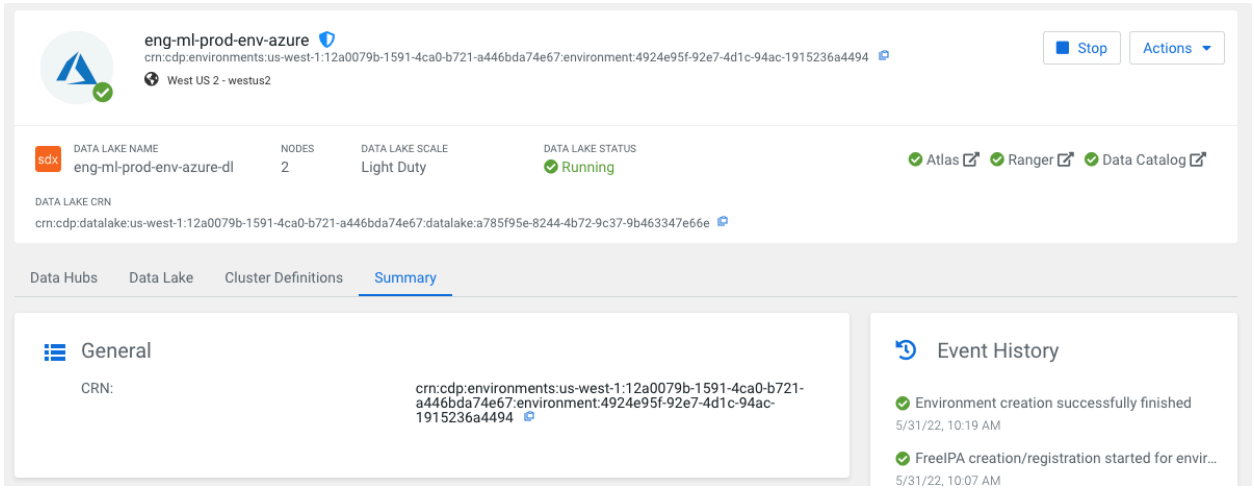

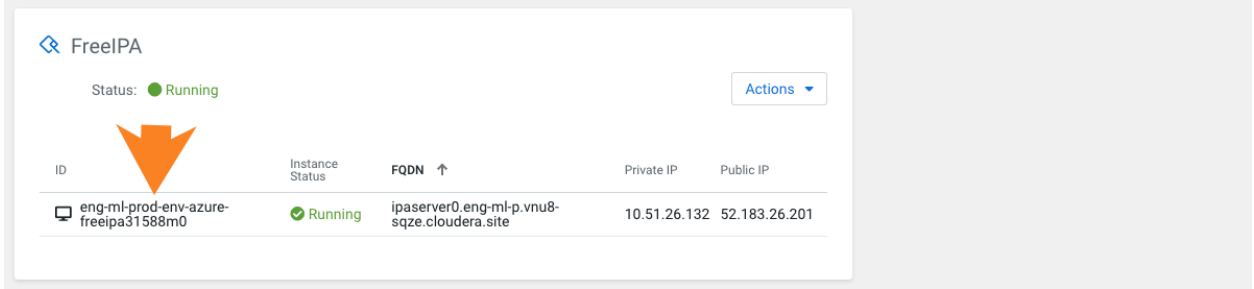

There is no link to Azure Portal, but you can copy the IDs of the VMs and search for them on Azure Portal. Just as you did for each Data Lake node, for each FreeIPA node,

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

navigate to the **Disks** section in Azure Portal. It will show the “Encryption at Host” as “disabled”. Again, leave the Azure Portal browser tabs open. You will need to get back to them shortly.

7. Navigate back to the Cloudera Management Console and stop the environment by clicking the **Stop** button in the top right corner in environment details. If you need detailed instructions, see Stop and restart an environment.
8. Once the environment has been successfully stopped, navigate back to the Azure Portal browser tabs opened earlier.
9. In Azure Portal, perform this for each VM of the Data Lake and for each VM of FreeIPA:
    a. Navigate to the **Disks** section.
    b. Within the Disks tab, navigate to the **Additional settings** section.
    c. Select “Yes” for the **“Encryption at Host”** setting:

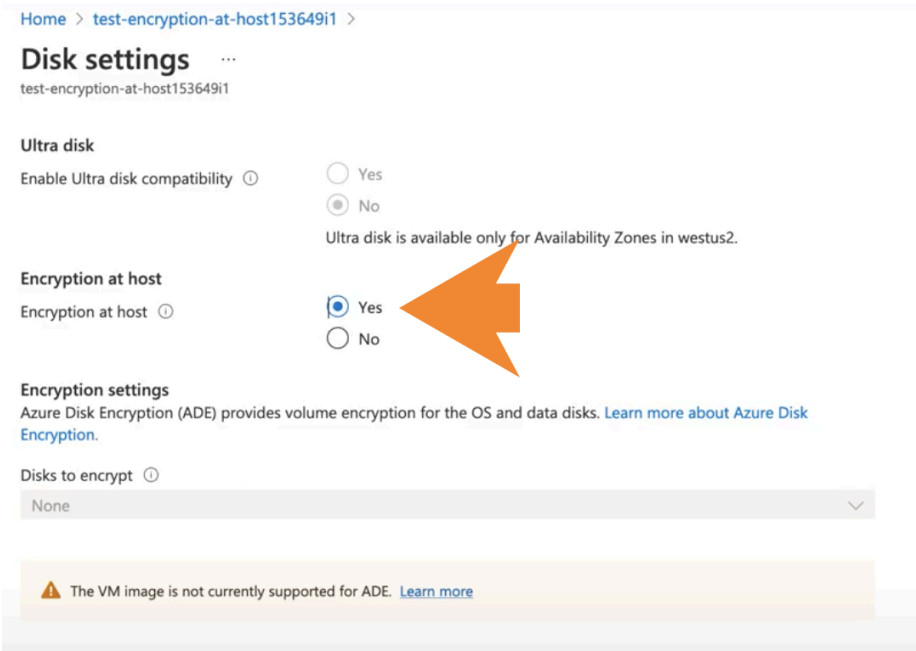

    d. Click on **Save**.
    e. Once the update is complete, you will see a message stating “Updated virtual machine”.
10. Before proceeding, ensure that you have performed the above steps for all Data Lake VMs and for all FreeIPA VMs.
11. Navigate back to the browser tab with the Cloudera Management Console and restart the environment by clicking the **Start** button in the top right corner in environment details. If you need detailed instructions, see Stop and restart an environment.
12. Once the environment has been successfully restarted, find the **Hardware** section in the **Data Lake** tab, just like you did earlier, and click on each of the Data Lake VM links. A new browser tab will open for each, redirecting you to the Azure Portal. For each of these VMs, navigate to the **Disks** section in Azure Portal. It will show the “Encryption at Host” as “enabled”:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

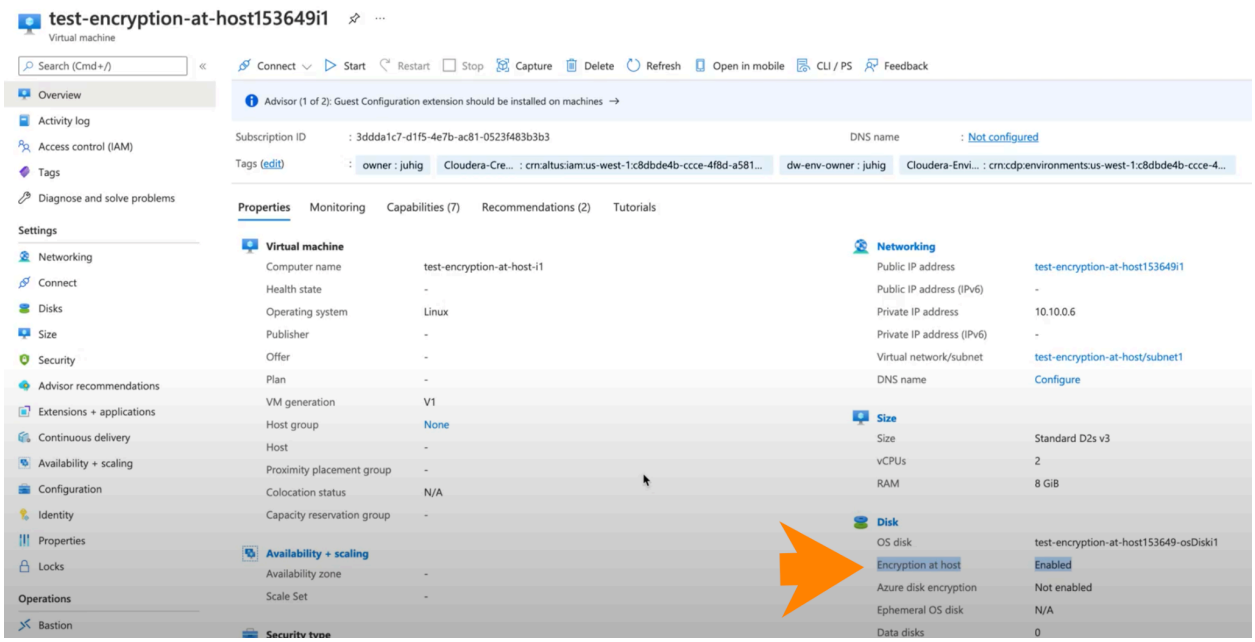

Next, confirm the same for all FreeIPA VMs in the **Summary** tab > **FreeIPA** tile.

Enable encryption at host for a Cloudera Data Hub cluster

Use these steps to enable encryption at host for an existing Cloudera Data Hub running on Azure. The steps involve manually enabling encryption at host for each Cloudera Data Hub VM via the Azure Portal.

Before you begin

Not all Azure VM types support encryption at host. In order to use encryption at host, when creating your Data Hub, select VM types that support encryption at host for each Data Hub host group. VM types can be selected per host group during Cloudera Data Hub creation in **Advanced Options > Hardware and Storage**. To find which VM types support encryption at host, follow the steps in Finding supported VM sizes**.**

Steps

1. In the Cloudera Management Console, select **Data Hubs**, and click on the specific Data Hub.
2. Make sure that your Cloudera Data Hub cluster is running.
3. In **Data Hub details**, navigate to the **Hardware** tab. Here you can find the list of all Cloudera Data Hub VMs organized into host groups:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

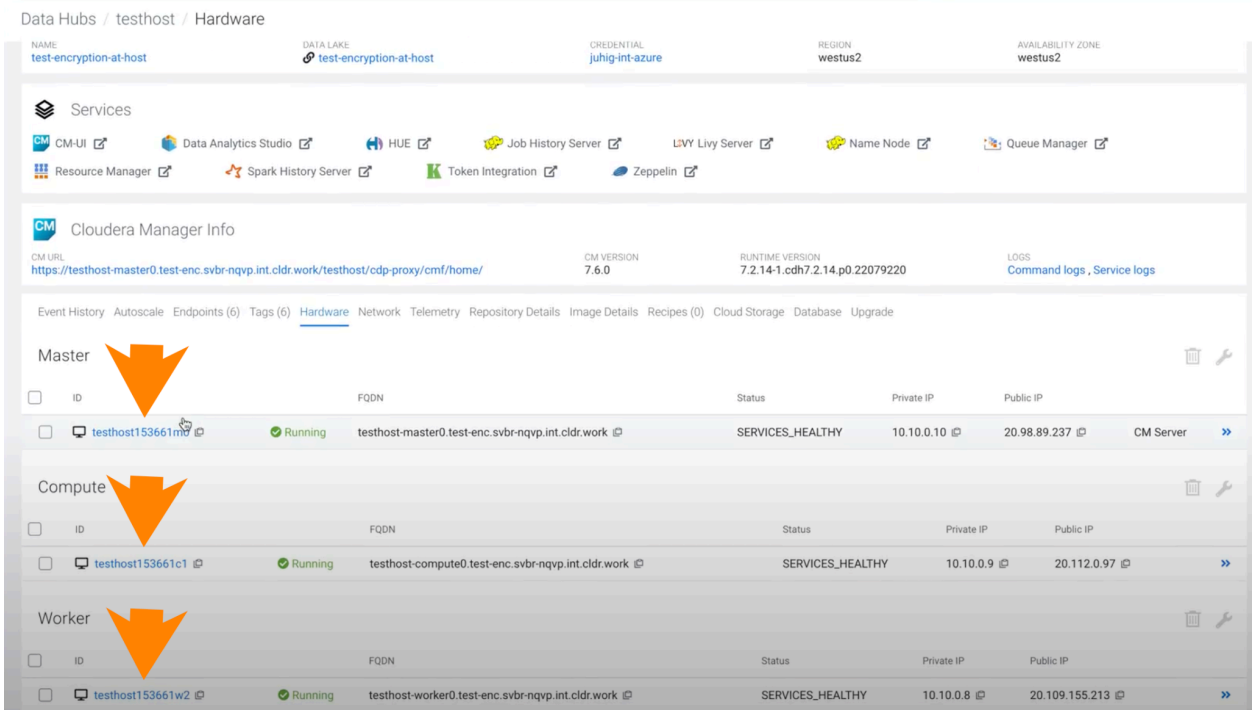

4. Click on each VM link and a new browser tab will open for each, redirecting you to the Azure Portal. You need to do this individually for each VM.
5. For each of the VMs, navigate to the **Disks** section in Azure Portal. It will show the "Encryption at Host" as "disabled":

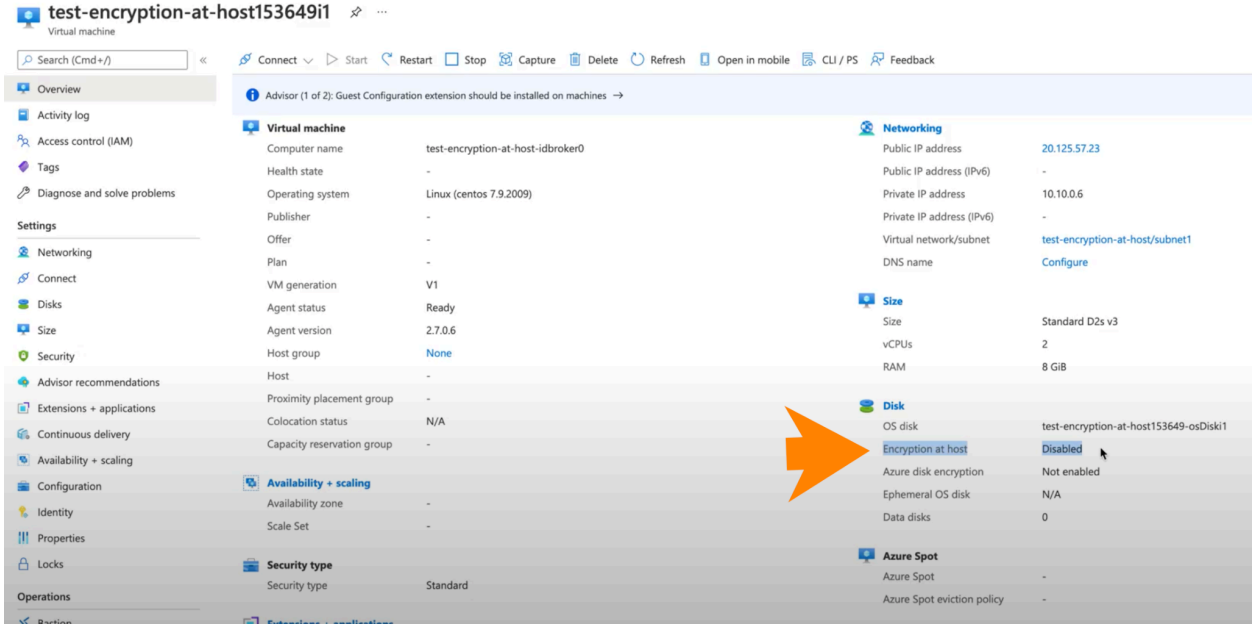

6. Leave the Azure Portal browser tabs open. You will need to get back to them shortly.
7. Navigate back to the browser tab with the Cloudera Management Console and restart the Cloudera Data Hub cluster by clicking the **Stop** button in the top right corner in environment details. If you need detailed instructions, see Stop a cluster.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

8. Once the Cloudera Data Hub cluster has been successfully stopped, navigate back to the Azure Portal browser tabs opened earlier.
9. In Azure Portal, perform the following for each Cloudera Data Hub VM:
    a. Navigate to the **Disks** section.
    b. Within the Disks tab, navigate to the **Additional settings** section.
    c. Select "Yes" for the **"Encryption at Host"** setting:

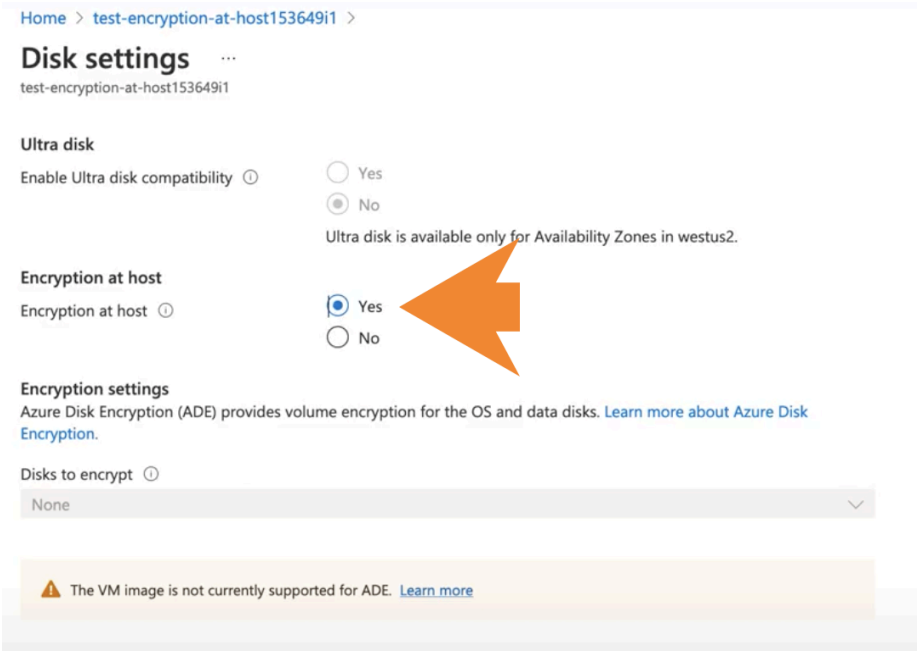

    d. Click on **Save**.
    e. Once the update is complete, you will see a message stating "Updated virtual machine".
10. Before proceeding, ensure that you have performed the above steps for all Data Hub VMs.
11. Navigate back to the browser tab with the Cloudera Management Console and restart the Cloudera Data Hub cluster by clicking the **Start** button in the top right corner. If you need detailed instructions, see Restart a cluster.
12. Once the environment has been successfully restarted, find the **Hardware** section in the **Data Hub details**, just like you did earlier, and click on each of the Cloudera Data Hub VM links.. A new browser tab will open for each, redirecting you to the Azure Portal. For each of these VMs, navigate to the **Disks** section in Azure Portal. It will show the "Encryption at Host" as "enabled":

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

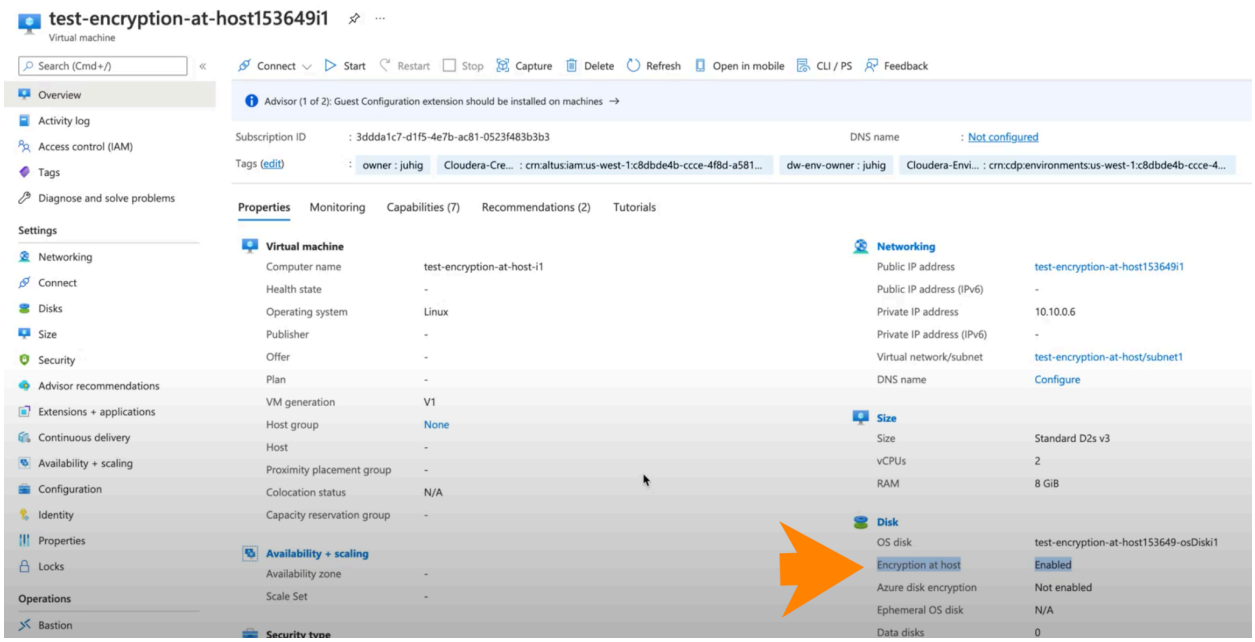

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Using Java virtual machine (JVM) debugger with Apache Spark jobs in Cloudera Data Engineering (Preview)

Date published: 2022-11-23
Date modified: 2022-11-23

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Java virtual machine (JVM) debugger with Apache Spark jobs in Cloudera Data Engineering (Preview) **1**

Legal Notice **2**

Contents **3**

Before you begin 4

Creating Cloudera CDP v2 credentials 4

Note the Role ARN 4

Setting up AWS accounts to run kubectl commands 4

Setting up Azure Virtual Clusters 5

Debugging with JVM 5

Flags to use with CDE CLI Job Run / CDE Spark Submit to Perform Debugging 5

Run a job in CDE 5

(Option 1) CDE CLI - Job Run 5

(Option 2) CDE CLI - Spark Submit 6

Starting the debugging session 6

Required flags for AWS jobs 6

Optional Flags: 6

Example Command For AWS 6

Example Command for Azure 7

Learn how to connect a Java virtual machine (JVM) debugger remotely to Spark pods (driver/executor) in Cloudera Data Engineering (CDE). You can use this JVM debugger to identify bugs or performance issues for Java/Scala Spark jobs where a JAR file is uploaded.

Important: This feature is not yet supported for Azure Private AKS and AWS Private EKS users.

Before you begin

The user who performs the debugging must be a CDE Admin or must be given credential from a CDE Admin. There are two tasks that you need to complete before using the debugger:

1. Creating Cloudera CDP v2 credentials in the CDE CLI
2. Create an ARN role.
3. Setting up AWS accounts to run kubectl commands in the AWS console

Azure setup is not required.

Creating Cloudera CDP v2 credentials

As a CDE Admin, set up the CDE CLI using API access keys before connecting the JVM debugger. See Cloudera Data Engineering CLI authentication for instructions on how to do this.

Note the Role ARN

You'll need the Role ARN to set up the AWS accounts to run kubectl commands.

1. Go to the Cloudera **Management Console**.
2. Click **Environment**
3. Click the environment where your CDE Service is deployed.
4. Click **Summary**.
5. Note the **Role ARN** under **Credentials**.

Setting up AWS accounts to run kubectl commands

If your environment has AWS EKS Virtual Clusters, as a system administrator, you must set up AWS accounts to run a Kubernetes port forward session before you use the JVM debugger. A system administrator must set up AWS accounts to run kubectl commands. These steps are to be performed once. After the setup is complete, IAM user credentials can be shared with those who want to use the JVM debugger. The credentials file must be stored at **$HOME/.aws/credentials**.

1. In the AWS console, create an IAM user (for example, kubectl-user) with Programmatic access (you need not grant any permissions).
2. Write down the User ARN to use in step 5, and copy the Access key ID and Secret access key and set up an AWS profile as follows:

```
[kubectl-user]

aws_access_key_id = <Access Key ID>
```

```
aws_secret_access_key = <Secret access key>
```

3. Navigate to IAM Roles and edit the cross-account IAM role (note the Role ARN) that was created as part of the CDP prerequisites. You can note the Role ARN in the previous set of steps under Note the Role ARN.
4. Navigate to **Trust relationships > Edit trust relationships**.
5. Add the following to the policy document, then click **Update trust policy**.

```
"Effect": "Allow",
 "Principal": {
  "AWS": "User ARN from step 2"
 },
 "Action": "sts:AssumeRole"

 },
```

6. Note the IAM User (that is, the kubectl-user) and Amazon Resource Names (ARNs) for all accounts because the user will need to provide the user names during debugging.
7. Install the **aws-iam-authenticator** CLI tool. See amazon docs: https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html

Setting up Azure Virtual Clusters

No setup is needed.

Debugging with JVM

Flags to use with CDE CLI Job Run / CDE Spark Submit to Perform Debugging

Note the flags below to use in your debugging process:
`--debug-driver` - To debug driver
`--debug-executor` - To debug executor

Run a job in CDE

There are two ways to run a job:

- CDE CLI - Job Run
- CDE CLI- Spark Submit

(Option 1) CDE CLI - Job Run

Run the job with debug flags from the CDE CLI:

```
$> cde job run --name java-spark --debug-driver
```

(Option 2) CDE CLI - Spark Submit

Run the job using the spark submit command from the CDE CLI:

```
$> cde spark submit
learning-spark-with-java/target/learning-spark-with-java-1.0-SNAPSHOT
.jar debug-driver --class dataframe.LongSleep --conf
spark.executor.memory=2g --debug-driver=true
```

The job run process starts. If a debugger is not attached, then progress on the job run will not be made.

Starting the debugging session

Run the following using the run id that resulted from running a job in option 1 or 2 above:

```
$> cde run debug –id run_id_of_job
```

Required flags for AWS jobs

Use these required flags for jobs using AWS EKS Virtual Clusters:

```
--aws-profile (string eg: kubectl-user)
     * If not provided, the "default" profile will be chosen.

--aws-role-arn (string eg: cross_account_role_from_aws_prerequisites)
     * It is mandatory that this is provided for aws running jobs.
     An assume-role operation will be performed on this role using
     the IAM User credentials in $HOME/.aws/credentials (Check
     Special Setup in case of AWS VCs).
```

Optional Flags:

If ports are not provided, a default port is assigned for both driver and executor. Run the following to assign a different port for each:

- `--driver-port (int)`
- `--executor-port (int)`

Example Command For AWS

```
$> cde  run debug --aws-role-arn
arn:aws:iam::302225162231:role/dex-priv-default-aws-cross-account-rol
e --aws-profile kubectl-user --id 20
```

Example Command for Azure

```
$> cde  run debug -id 20
```

The CLI command waits for Spark pods to display, and then a debug session to those pods opens in another window. Once a session is established, it will print localhost:local_port for the debugger to connect to.

Using Custom Spark Runtime Docker Images via API/CLI (Preview)

Date published: 2022-09-06
Date modified: 2023-07-31

Legal Notice

© Cloudera Inc. 2022-2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Custom Spark Runtime Docker Images via API/CLI (Preview) 1

Legal Notice 2

Contents 3

This is a detailed user guide that demonstrates how to run Spark jobs using custom Spark runtime Docker images via API/CLI. Custom Spark runtime Docker images are used when custom packages and libraries need to be installed and used when executing Spark jobs. These custom packages and libraries can be proprietary software packages like RPMs that need to be compiled to generate the required binaries. Docker images allow you to pre-bake these dependencies into a self-contained Docker file that can be used across multiple Spark jobs.

Important: If you're using Cloudera Data Engineering (CDE) 1.18 and above, you can't use Custom Spark Runtime Docker Images with CDE 1.17 and below.

Prerequisites

Entitlement for Customer Docker images

In order to use custom Spark runtime Docker images, your CDP tenant must have the **DE_CUSTOM_RUNTIME** entitlement enabled. If not yet in place, this should be requested via your Cloudera Account Team (primarily your Solution Engineer). They will fulfill the request internally and confirm to you when the entitlement has been applied. Please allow 24 hours for fulfillment.

Note that the DE_CUSTOM_RUNTIME entitlement must be enabled **before** a Virtual Cluster is created in order to be able to create custom runtime resources within that cluster.

Docker repository credentials

In order to pull the base Docker image, you must have credentials and authenticate your docker client to `docker.repository.cloudera.com`. To get credentials:

1. Raise an "Admin" type case with Cloudera Support and request a **License Key**.
2. Once the License Key is received, navigate to https://www.cloudera.com/downloads.html and **log in** with your MyCloudera credentials.
3. Use the Credentials Generator tool - this is about half-way down the page (you will see an orange "Sign In" button if not already logged in).

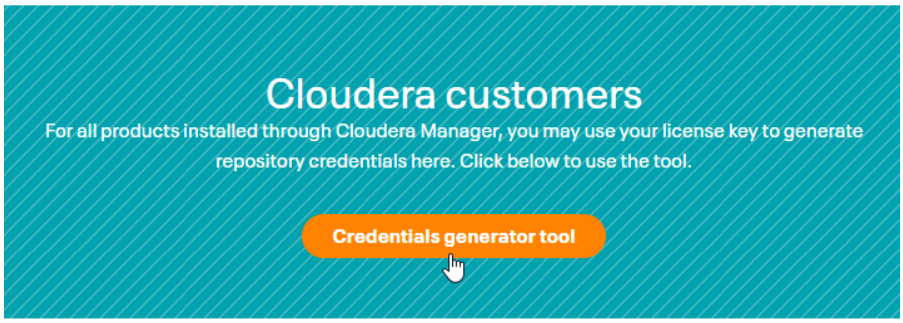

As directed on the page, copy and paste the entire contents of your license file into the text box and press the Get Credentials button to generate your username and password.

Steps

1. Create a custom Docker image
   Build the “custom-spark-dex-runtime” image based on the “dex-spark-runtime” image of the Cloudera Data Engineering (CDE) version.
   The image should be based on the dex-spark-runtime of the current dex version.
   The relevant dex-spark-runtime image is
   <registry-host>/cloudera/dex/dex-spark-runtime-<spark version>-<cdh version>:<dex version>

   Example: DockerFile for DEX `1.15.0-b117`, Spark 2.4.8 and CDP Runtime version `7.2.14.0`

```
Unset
FROM
docker.repository.cloudera.com/cloudera/dex/dex-spark-runtime-2.4
.8-7.2.14.0-7.2.14.0:1.15.0-b117
USER root
RUN yum install -y git && yum clean all && rm -rf /var/cache/yum
RUN pip2 install virtualenv-api
RUN pip3 install virtualenv-api
USER ${DEX_UID}
```

2. Build the docker image by tagging it with the custom registry and push it to the custom registry.

   Example:

```
Unset
mac@local:$ docker build --network=host -t
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8
-7.2.14.0:1.15.0-b117-custom . -f Dockerfile
```

```
mac@local:$ docker push
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8
-7.2.14.0:1.15.0-b117-custom
```

Here, the custom registry is “docker.my-company.registry.com” and the registry namespace is “custom-dex”.

3. Create a custom runtime image resource.
   Register “custom-spark-dex-runtime” docker image as a resource of type “custom-runtime-image”.

   a. Create a resource for the registries which do not require any authentication. If using a public docker registry or If the docker registry is in the same environment, for example, the same AWS account or Azure subscription where the CDE service is running, then you do not need to create credentials.

   CLI

```
Unset
mac@local:$ cde resource create --name custom-image-resource
--image
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8
-7.2.14.0:1.15.0-b117-custom --image-engine spark2 --type
custom-runtime-image
```

Note: To obtain $CDE_TOKEN to execute the REST API examples, follow the [Getting a Cloudera Data Engineering API access token] document.

REST API

```
Unset
curl -X POST -k 'https://<dex-vc-host>/dex/api/v1/resources \
  -H "Authorization: Bearer ${CDE_TOKEN}"  \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
```

```
  --data  '{
  "customRuntimeImage": {
    "engine": "spark2",
    "image":
"docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.
8-7.2.14.0:1.15.0-b117
-custom"
  },
  "name": "custom-image-resource",
   "type": "custom-runtime-image"
}'
```

Once done, skip to [step 4](#) to submit the job.

b. Create a resource which requires the credentials to access the registry. Use the command below or the API request to create the credentials. These credentials are stored as a secret.

CLI

```
Unset
mac@local:$ ./cde credential create --name docker-creds --type
docker-basic --docker-server docker-sandbox.infra.cloudera.com
--docker-username my-username
```

REST API

```
Unset
curl -X POST -k 'https://<dex-vc-host>/dex/api/v1/credentials' \
  -H "Authorization: Bearer ${CDE_TOKEN}"  \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  --data  '{
  "dockerBasic": {
    "password": "password123",
    "server": "docker-sandbox.infra.cloudera.com",
```

```
    "username": "my-username"
  },
  "name": "docker-creds",
  "type": "docker-basic"
}'
```

C. Register “custom-spark-dex-runtime” docker image as a resource of type “custom-runtime-image” by specifying the name of the credential created above.

CLI

```
Unset
mac@local:$ ./cde resource create --name custom-image-resource
--image
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8
-7.2.14.0:1.15.0-b117
-custom --image-engine spark2 --type custom-runtime-image
--image-credential docker-creds
```

REST API

```
Unset
curl -X POST -k 'https://<dex-vc-host>/dex/api/v1/resources \
  -H "Authorization: Bearer ${CDE_TOKEN}"  \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  --data  '{
  "customRuntimeImage": {
    "credential": "docker-creds",
    "engine": "spark2",
    "image":
"docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.
8-7.2.14.0:1.15.0-b117
```

```
-custom"
  },
  "name": "custom-image-resource",
   "type": "custom-runtime-image"
}'
```

4. Submit a job by setting the “custom-spark-dex-runtime” image as a resource using the CDE CLI.

   SPARK COMMAND

```
Unset
mac@local:$ ./cde --user cdpuser1 spark submit
/Users/my-username/spark-examples_2.11-2.4.4.jar
--class org.apache.spark.examples.SparkPi 1000
--runtime-image-resource-name=custom-image-resource
```

   JOB COMMAND

```
Unset
mac@local:$ ./cde --user cdpuser1 resource create --name
spark-jar
mac@local:$ ./cde --user cdpuser1 resource upload --name
spark-jar --local-path spark-examples_2.11-2.4.4.jar
mac@local:$ ./cde --user cdpuser1 job create --name
spark-pi-job-cli --type spark --mount-1-resource spark-jar
--application-file spark-examples_2.11-2.4.4.jar --class
org.apache.spark.examples.SparkPi --user cdpuser1 --arg 22
--runtime-image-resource-name custom-image-resource
```

5. The spark driver/executor pods should use this image and you can confirm it by opening a shell into those pods and verifying if the external installed libraries or files exist.

Accessing ECR repository from different AWS environments

Use Case:

There are separate AWS environments for different business units and there is an ECR repository which contains the common docker images that needs to be accessed from these environments (having different AWS accounts).

By default, The ECR will be accessible to the AWS services running in the same AWS environment. The same ECR can be accessed from different AWS environments by updating the ECR repository permissions.

For Example :

- ECR repository is in env1 (AWS-ACCOUNT-1)
- Sales Team services are running in env2 (AWS-ACCOUNT-2)
- Finance Team services are running in env3 (AWS-ACCOUNT-3)

Below steps can be followed to update the ECR permissions to allow the access for both Sales and Finance teams' services.

Steps:

1. Login into AWS console and open the desired ECR repository.
2. Click on the Permissions tab.

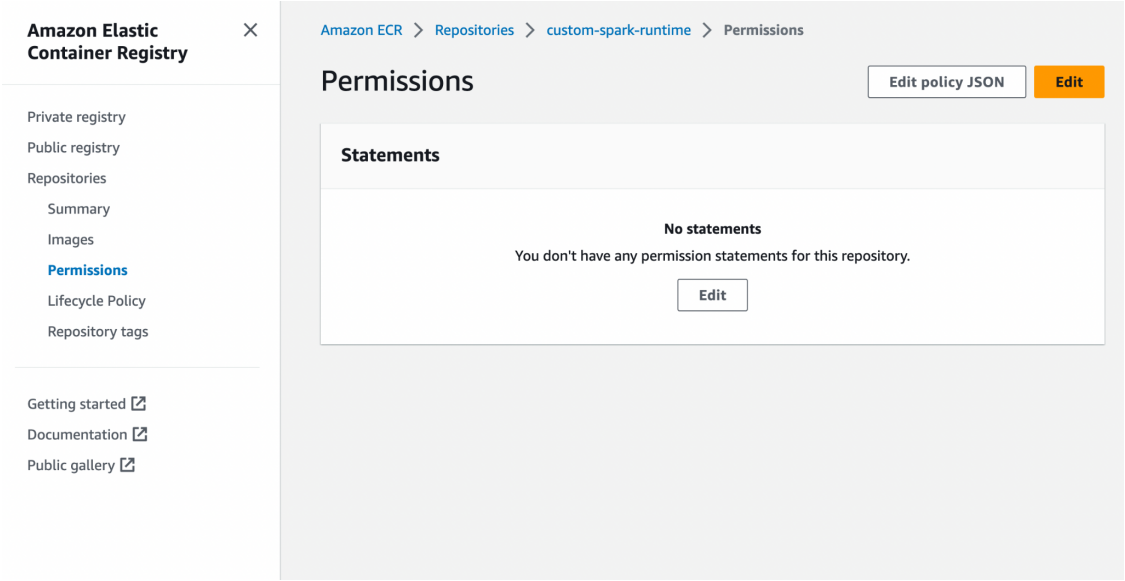

3. Click on Edit Policy Json. This will show you the current JSON policy document for the repository.

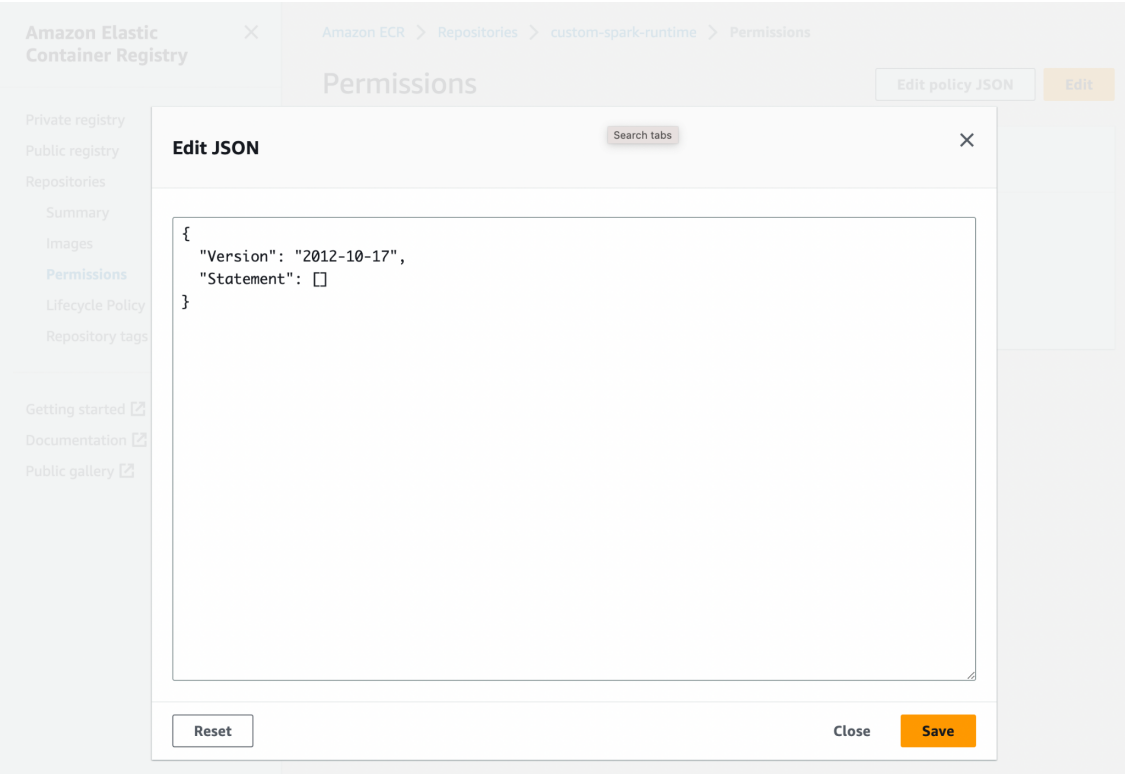

4. Modify the current JSON policy document by adding a new statement. If the repository has no permissions set yet then you can simply copy and paste the below JSON policy document (Update the required AWS account ids in this document)

```
Unset
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "crossAccountAllowRead",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::<AWS-ACCOUNT-2>:root",
          "arn:aws:iam::<AWS-ACCOUNT-3>:root"
        ]
      },
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer",
        "ecr:ListImages"
      ]
    }
  ]
}
```

5. Click on Save button.
6. Verify If the policy has been set successfully by clicking on the Permissions tab again. The new permissions and the AWS accounts IDs should be there.

Note : Ensure the AWS account IDs are correct otherwise you may see errors while saving the policy.

Once the permissions are set successfully on ECR for different AWS accounts then the services running in these AWS environments can access the ECR without any changes in the services or environment itself. There is no need to provide any credentials to access the ECR from these AWS accounts.

Troubleshooting

Error: Custom image resource with missing or wrong credentials

Creating a custom image resource with missing or wrong credentials should result in the below error which can be seen in the logs or in Kubernetes pod events.

Example

```
Unset
Failed to pull image
"docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.
8-7.2.14.0:1.15.0-b117-custom":
rpc error: code = Unknown desc = Error reading manifest
1.15.0-b117-custom in
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8
-7.2.14.0:errors: denied: requested access to the resource is
denied unauthorized: authentication required
```

CDF Service Upgrade (Preview)

Date published: 2022-06-28
Date modified: 2023-04-27

Legal Notice

© Cloudera Inc. 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
Pre-upgrade information **4**
Update cdp-liftie-instance-profile **5**
Upgrade a DataFlow service **13**
Troubleshooting DataFlow upgrade errors **15**
Setting up kubectl to connect to the DataFlow Kubernetes cluster 15
Upgrade failure scenarios 16
Upgrade fails on initiation 17
Upgrade fails and rollback is not possible 17
Upgrade fails and rollback is possible 18
Retrying an upgrade 19

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Pre-upgrade information

Learn about tasks you need to perform and information you need to consider before starting a CDF service upgrade.

important

If you are upgrading on AWS to CDF-PC 2.3.0 or higher from release 2.2.0 or lower and you use a pre-created IAM policy, you need to update the cdp-liftie-instance-profile before the upgrade. For more information, see *Update cdp-liftie-instance-profile*.

Note:

In CDF-PC versions 2.4.0 and lower, ZooKeeper may lose component state information under certain conditions due to its previous configuration to use ephemeral storage. If a ZooKeeper pod fails, it retains component state information as long as it is rescheduled to the same Kubernetes (K8s) node, as it can reattach to storage already present there. If the ZooKeeper pod gets rescheduled to a different K8s node, existing storage is not moved with it, therefore state information is lost. If your use case requires that component state information is retained even during an upgrade, contact Cloudera Support to provide you with a patch that you need to apply to every deployment before starting the upgrade.

- You trigger an upgrade for a selected CDF environment. Upgrades move from an older version to the latest version available.
- Once you trigger the upgrade, the service enters **UPGRADING** state and is upgraded to the latest supported DataFlow and Kubernetes versions. Kubernetes upgrade can take up to an hour to complete. During the upgrade service actions are restricted, that is, you are not able to create or manage deployments in the given DataFlow service.
- As part of a DataFlow Service (Environment) upgrade, all existing deployments are also upgraded.
- If the upgrade fails, all components are rolled back to their original state.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- To keep existing DataFlow deployments working, you may choose not to upgrade the NiFi version of existing deployments during the upgrade. This selection applies to all existing deployments.
  Note:
  The option to skip NiFi version upgrade is not always available: NiFi version upgrade is mandatory if the upgrade wizard determines that the new workload version is not compatible with the current NiFi version running in any deployment. The same automatism prevents you from accidentally downgrading NiFi below the minimum supported version later.
  You can manually upgrade NiFi for individual deployments later.
- If upgrading a deployment fails, all components are rolled back to their original state and it causes the overall DataFlow service upgrade to be rolled back as well.

Update cdp-liftie-instance-profile

When upgrading to CDF-PC 2.3.0 or higher from release 2.2.0-h4 or lower on AWS and you use a pre-created IAM policy, you need to update the cdp-liftie-instance-profile before the upgrade. These updates are necessary because CDF-PC 2.3.0 has migrated to use EKS k8s 1.23 and CSI EBS provisioner. This storage provisioner requires updated policies to allow the creation and deletion of volumes from within the EKS cluster.

Without these policy changes, volumes provisioned by a pre-2.3.0 DataFlow version will be left behind if the corresponding deployment is terminated or the data flow is disabled after the upgrade.

About this task: Update the `cdp-liftie-instance-profile` using either the AWS UI or CLI.

1. Add the ebs-csi policy to the `cdp-liftie-instance-profile` IAM profile.

```
{
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
"Version": "2012-10-17",

"Statement": [

{

"Action": [

"ec2:CreateSnapshot",

"ec2:AttachVolume",

"ec2:DetachVolume",

"ec2:ModifyVolume",

"ec2:DescribeAvailabilityZones",

"ec2:DescribeInstances",

"ec2:DescribeSnapshots",

"ec2:DescribeTags",

"ec2:DescribeVolumes",

"ec2:DescribeVolumesModifications"

],

"Resource": "*",

"Effect": "Allow"

},

{

"Condition": {

"StringEquals": {

"ec2:CreateAction": [

"CreateVolume",

"CreateSnapshot"

]
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
}

},

"Action": [

"ec2:CreateTags"

],

"Resource": [

"arn:aws:ec2:*:*:volume/*",

"arn:aws:ec2:*:*:snapshot/*"

],

"Effect": "Allow"

},

{

"Action": [

"ec2:DeleteTags"

],

"Resource": [

"arn:aws:ec2:*:*:volume/*",

"arn:aws:ec2:*:*:snapshot/*"

],

"Effect": "Allow"

},

{

"Condition": {

"StringLike": {

"aws:RequestTag/ebs.csi.aws.com/cluster": "true"
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
}

},

"Action": [

"ec2:CreateVolume"

],

"Resource": "*",

"Effect": "Allow"

},

{

"Condition": {

"StringLike": {

"aws:RequestTag/CSIVolumeName": "*"

}

},

"Action": [

"ec2:CreateVolume"

],

"Resource": "*",

"Effect": "Allow"

},

{

"Condition": {

"StringLike": {

"aws:RequestTag/kubernetes.io/cluster/*": "owned"}

},
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
"Action": [
"ec2:CreateVolume"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Condition": {
"StringLike": {
"ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
}
},
"Action": [
"ec2:DeleteVolume"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Condition": {
"StringLike": {
"ec2:ResourceTag/CSIVolumeName": "*"
}
},
"Action": [
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
"ec2:DeleteVolume"

],

"Resource": "*",

"Effect": "Allow"

},

{

"Condition": {

"StringLike": {

"ec2:ResourceTag/kubernetes.io/created-for/pvc/name": "*"

}

},

"Action": [

"ec2:DeleteVolume"

],

"Resource": "*",

"Effect": "Allow"

},

{

"Condition": {

"StringLike": {

"ec2:ResourceTag/CSIVolumeSnapshotName": "*"

}

},

"Action": [

"ec2:DeleteSnapshot"
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
],
"Resource": "*",
"Effect": "Allow"
},
{
"Condition": {
"StringLike": {
"ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
}
},
"Action": [
"ec2:DeleteSnapshot"
],
"Resource": "*",
"Effect": "Allow"
}
]
}
```

2. Add the efs-csi policy to the `cdp-liftie-instance-profile` IAM profile.

```
{
"Version": "2012-10-17",
"Statement": [
{
"Action": [
"elasticfilesystem:DescribeAccessPoints",
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
"elasticfilesystem:DescribeFileSystems",

"elasticfilesystem:DescribeMountTargets"

],

"Resource": "*",

"Effect": "Allow"

},

{

"Condition": {

"StringLike": {

"aws:RequestTag/efs.csi.aws.com/cluster": "true"

}

},

"Action": [

"elasticfilesystem:CreateAccessPoint"

],

"Resource": "*",

"Effect": "Allow"

},

{

"Condition": {

"StringEquals": {

"aws:ResourceTag/efs.csi.aws.com/cluster": "true"

}

},

"Action": [
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
"elasticfilesystem:DeleteAccessPoint"

],

"Resource": "*",

"Effect": "Allow"

}

]

}
```

Upgrade a DataFlow service

You can upgrade your Cloudera DataFlow (CDF) service together with all DataFlow deployments in that environment with the push of a button.

Steps

1. From the CDF left navigation pane select **Environments**.
2. Select the CDF service you want to upgrade and click **Manage DataFlow**.
3. Click **Actions→Upgrade DataFlow**.
   Note: If the upgrade wizard determines that all NiFi versions of your DataFlow deployments will be compatible with the upgraded CDF service version, you have the option to skip upgrading NiFi in current deployments. In this case you can upgrade NiFi

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

manually later using the Change NiFi Version feature.

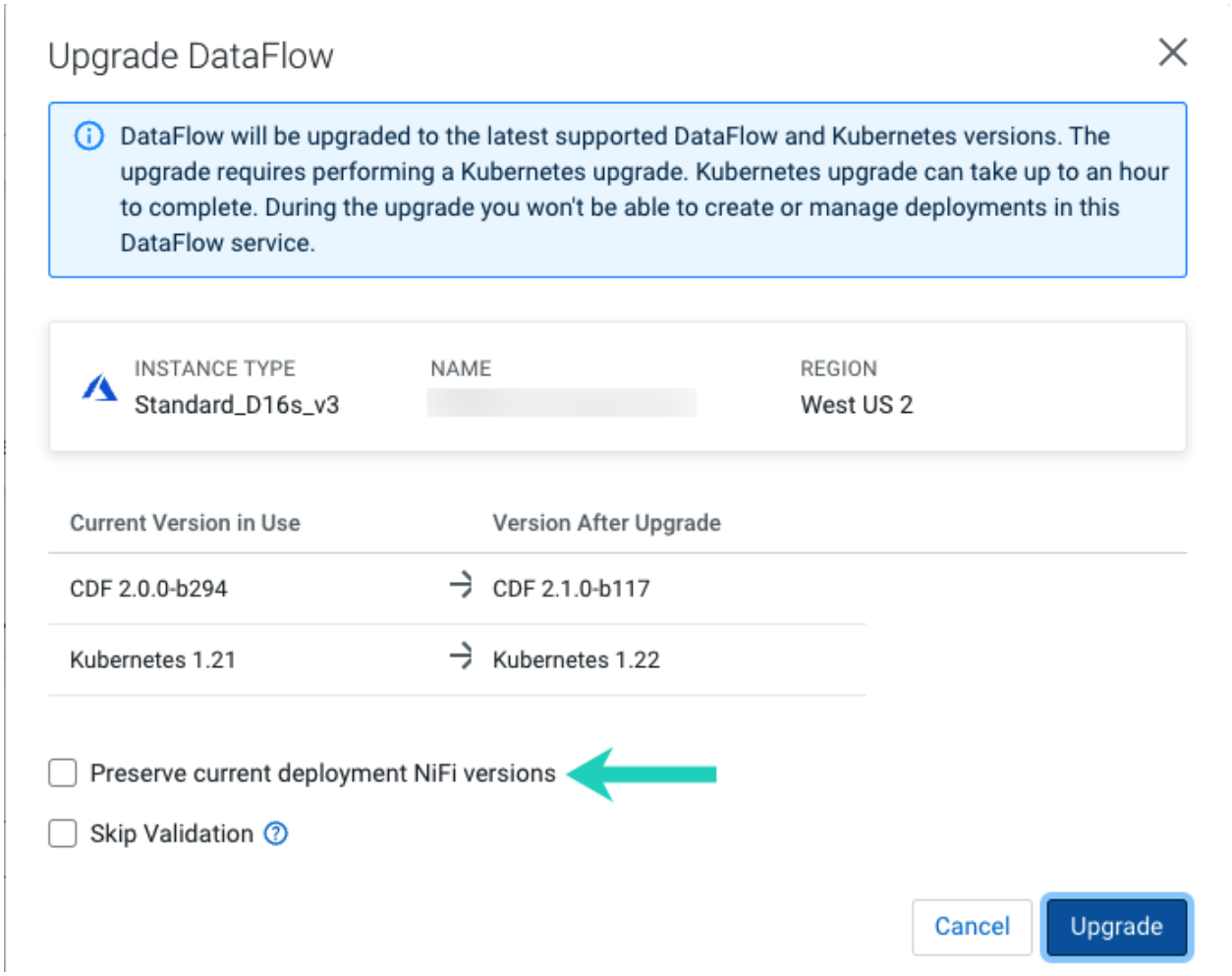

4. Click **Upgrade**.

   Result: The environment status changes to **Upgrading** and you see a similar message under **Events**:

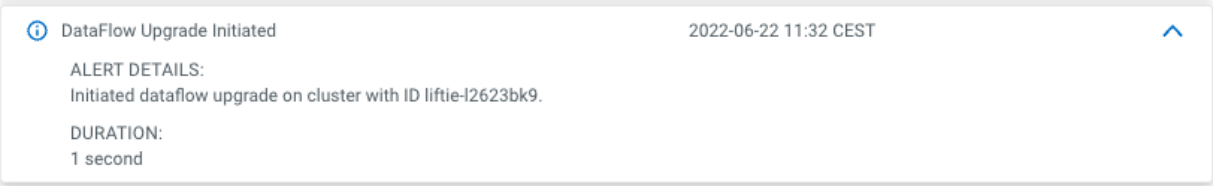

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

5. You can monitor the upgrade procedure in **Event History**. Click **Manage DataFlow** then Select **Alerts**.

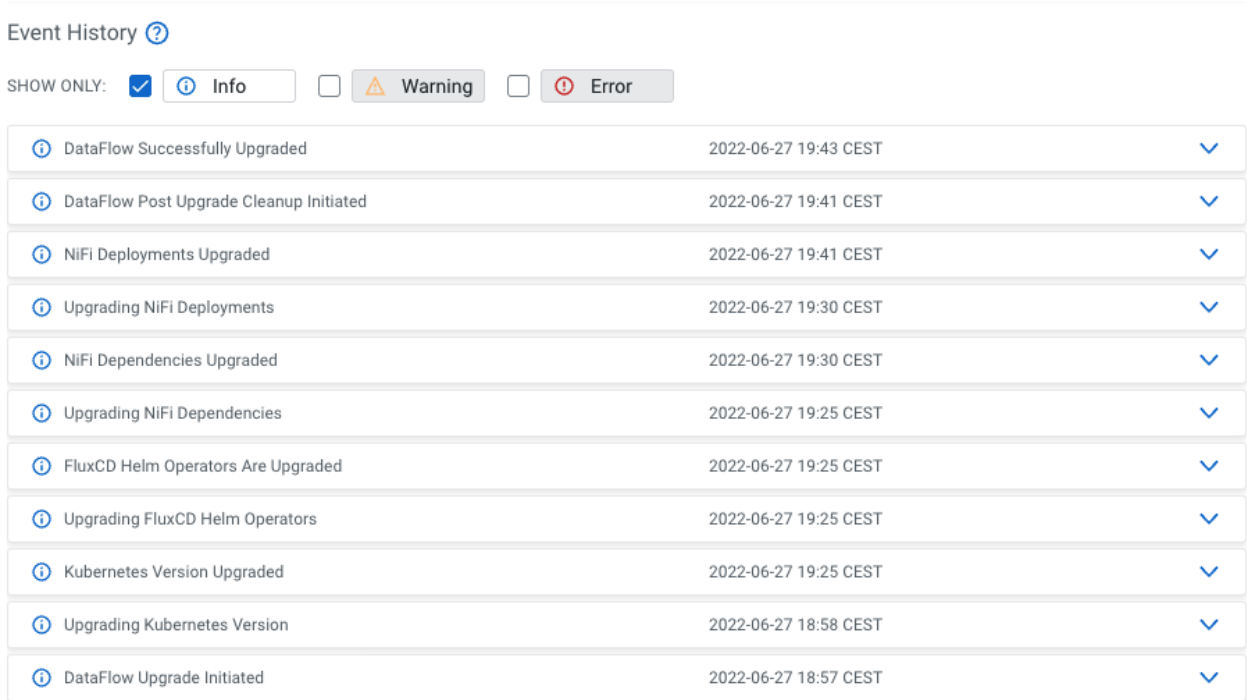

Results: The message **DataFlow Successfully Upgraded** in **Event History** and the **Status** of the service changing from **Upgrading** to **Good Health** on the **Environments** page signal that the upgrade has ended. Depending on the number of running nodes in the cluster, the upgrade can take more than an hour to complete.

Troubleshooting DataFlow upgrade errors

Short Description: Learn how to recognize and address common errors with your Cloudera DataFlow (CDF) service upgrades.

Setting up kubectl to connect to the DataFlow Kubernetes cluster

It is helpful to have access to the DataFlow Kubernetes cluster using command line tools such as kubectl when you are troubleshooting deployment or upgrade failures. To set up kubectl access, follow these steps:

1. In DataFlow, from the **Environments** page, select the DataFlow service for which you want to add or remove user access.
2. Click **Manage DataFlow**.
3. From the **Actions** menu, click **Manage Kubernetes API Server User Access**.
4. Add the AWS IAM role that you will authenticate as to the list of authorized users for DataFlow by entering the ARN in the **Add User** dialog.
5. Use the *Download Kubeconfig* action to retrieve the kubeconfig file for connecting to your cluster.
6. Set up your kubectl to use the downloaded kubeconfig file:
   export KUBECONFIG=*[***PATH/TO/DOWNLOADED/KUBECONFIG/FILE***]*
7. Run kubectl get ns and validate that your output looks similar to:

```
NAME                        STATUS   AGE
cadence                     Active   37h
cert-manager                Active   37h
cfm-operator-system         Active   37h
default                     Active   38h
dfx-dev-environment-ns      Active   36h
dfx-idbrokers3-ns           Active   36h
dfx-kafkatos3-ns            Active   22h
dfx-local                   Active   37h
dfx-ops                     Active   37h
kube-node-lease             Active   38h
kube-public                 Active   38h
kube-system                 Active   38h
liftie                      Active   37h
logging                     Active   37h
monitoring                  Active   37h
nfs-provisioner-system      Active   37h
nginx-ingress               Active   37h
prometheus-operator-system  Active   37h
```

With kubectl being set up correctly, you are able to access NiFi and other DataFlow logs directly through the CLI.

Upgrade failure scenarios

An upgrade may fail at various points, for different reasons. This document covers the following scenarios:

- The upgrade fails upon initiation.
- The upgrade starts but fails and rollback is not possible.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- The upgrade starts but fails, and rollback is possible.

Upgrade fails on initiation

Symptom: When the upgrade fails to start, the status of the environment does not change to **Upgrading** and you receive an error message stating the cause of the failure.

Possible causes:

- Any of the deployments is in one of the following states:
    - FAILED_TO_UPDATE
    - FAILED_TO_UPDATE_NIFI_VERSION

How to fix it: Address the deployment(s) in failure state by re-attempting the previously failed update. A deployment in FAILED_TO_UPDATE has changes that were not successfully applied. A deployment in FAILED_TO_UPDATE_NIFI_VERSION was unable to successfully complete a Change NiFi Version request. If unable to resolve the deployment issues, you may terminate them instead. Once the issues are addressed, retry the upgrade.

Possible causes:

- Any of the deployments is in one of the following states:
    - Any transitional / in progress state

How to fix it: Wait for the deployment(s) that are actively performing another requested action to complete. Retry the upgrade.

Upgrade fails and rollback is not possible

Symptom: The status of the service, and shortly after, the deployments, changes to **Failed to Upgrade**. The status icon on the service changes to **Bad Health**.

How to fix it:

1. Check the **Event History** if there is anything obvious in there. For example, did the upgrade fail during or after the K8s upgrade?
2. If K8s upgrade failed, check on the cloud provider side whether there is anything obvious as a reason for the failure. For example, check activity logs and node pools on Azure, or EKS cluster on AWS.
3. If the K8s upgrade succeeded, but the upgrade failed during FluxCD Helm Operators upgrade, check pods and Flux deployments in `flux-system` namespace.
4. If the K8s and FluxCD Helm Operators upgrades succeeded, but the upgrade failed during NiFi Dependencies upgrade, check pods and Helm releases (the label

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

`com.cloudera.dfx.workload/component=environment` identifies environment-related Helm releases).

5. If the K8s upgrade, FluxCD Helm Operators, and NiFi Dependencies upgrades were successful but NiFi Deployments upgrade failed, check deployment pods and Helm releases with label `com.cloudera.dfx.workload/component=deployment`
6. If you managed to pinpoint and fix the possible cause of the failure, retry the upgrade.
7. If you did not find the cause of the failure, or the upgrade still fails upon retry, contact Cloudera Support.

Upgrade fails and rollback is possible

Symptom: Upgrade fails and rollback process starts. If the rollback is successful, the Status of the service returns to **Good Health**, CDF and NiFi versions return to their pre-upgrade values. If the rollback fails, the Status of both the CDF service and running deployments becomes **Failed to Upgrade**. The order in which this transition happens depends on where the rollback actually failed.

How to fix it: In the event of an upgrade failure, DataFlow determines if rollback is possible and automatically initiates it. No user intervention is required.

If the rollback is successful:

1. After the rollback ends, from the **Environments** page, select the DataFlow service you were trying to upgrade and click the **Alerts** tab. Check **Event History** to see if there is anything obvious in there or something that narrows down the possible cause of the failure.
2. From the **Dashboard page** select running deployments that failed to upgrade and check **Alerts→Event History** to see if there is anything obvious in there or something that narrows down the possible cause of the failure.
3. Check the K8s cluster using CLI and try to fix errors based on the information you obtained from **Event History** in the previous steps.
4. Retry the upgrade.
5. If the upgrade still fails contact Cloudera Support.

If the rollback fails:
The status of both the service and the running deployments changes to **Failed to Upgrade**. The order in which they transition depends on where the rollback actually failed.

1. Retry the upgrade. During the retry, it is useful to monitor the status of the upgrade on the K8s cluster to better pinpoint the cause of a possible failure.
2. If the upgrade still fails upon retry, Contact Cloudera Support.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Note: A DataFlow upgrade can only be rolled back if the upgrade did not require a new Kubernetes (K8s) version.

Retrying an upgrade

1. On the DataFlow **Dashboard** select **Environments** from the left navigation pane.
2. Select the service where you want to retry the upgrade and click **Manage DataFlow**.
3. Click **Actions→Retry Upgrade DataFlow**.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Cluster Orchestrator Component Password Rotation (Preview)

Date published: 2023-03-02
Date modified: 2024-12-04

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Cloudera uses a cluster orchestrator to install and configure Cloudera services in Data Lake/Cloudera Data Hub clusters as well as FreeIPA. The cluster orchestrator operates with passwords (for each component) that expire every 180 days. Although the password is rotated automatically every 180 days, you can rotate it manually using the CDP CLI if desired. This may be useful if your cluster was not running when the cluster orchestrator attempted to rotate the password.

Use the following CDP CLI commands to manually rotate the cluster orchestrator component passwords:

FreeIPA

```
cdp environments rotate-salt-password --environment <value>
```

Data Lake

```
cdp datalake rotate-salt-password --datalake <value>
```

Cloudera Data Hub

```
cdp datahub rotate-salt-password --cluster <value>
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

CMK Encryption on AWS (Preview)

Date published: 2021-08-05
Date modified: 2022-08-10

Legal Notice

© Cloudera Inc. 2021. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**

What is a Customer Managed Key (CMK) **4**

Prerequisites **4**

Limitations **4**

Implementing CMK on AWS **4**

1. Create a key [[Creation of key will be done by Env services]] 4

2. Update policies **5**

Overview of the setup steps **6**

Permissions for using encryption **6**

EC2 permissions 6

KMS permissions 7

Encryption key requirements **7**

Ensuring that an existing encryption key can be used 7

Create a new encryption key on AWS 8

Create an environment with a CMK **9**

Set a CMK for an existing environment **10**

Steps - CDP UI 10

Steps - CDP CLI 10

References **11**

How to choose your CMK configuration **11**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

What is a Customer Managed Key (CMK)

In AWS, the Key Management System (KMS) provides keys for encrypting data and file systems. Typically, KMS is used to generate and manage keys, but as an alternative, you are able to generate a key yourself and use it to encrypt data and file systems in AWS. In this case, you are responsible for generating, maintaining, and rotating keys. This is the Customer Managed Key (CMK). You have full control over CMKs, including establishing and maintaining the key policies, IAM policies, grants, enabling and disabling them, rotating their cryptographic material, adding tags, creating aliases that refer to the CMK, and scheduling the CMKs for deletion.

CMKs are used to encrypt the following:

- EFS
- EBS volumes attached to nodes
- K8s etcd secrets
- S3 log buckets (that are used with CDE)

KMS keys are not used by CML to encrypt S3 buckets, which CDE uses and Liftie supports.

The key is automatically performed by Environment Services. For more information, see: CMK Encryption on AWS.

Limitations

- Only automatic key rotation is supported.
- CML only supports symmetric keys for CMK encryption

Implementing CMK on AWS

Follow the steps below to begin using CMK.

1. Update policies

To use CMKs in CML, ensure that the following three permission blocks are added to the policy section, in addition to the default policies:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
{
   "Sid":"Allow Autoscaling service-linked role for attachment of
persistent resources",
   "Effect":"Allow",
   "Principal":{

"AWS":"arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/autoscali
ng.amazonaws.com/AWSServiceRoleForAutoScaling"
   },
   "Action":"kms:CreateGrant",
   "Resource":"*",
   "Condition":{
      "Bool":{
         "kms:GrantIsForAWSResource":"true"
      }
   }
},
{
   "Sid":"Allow Autoscaling service-linked role use of the CMK",
   "Effect":"Allow",
   "Principal":{

"AWS":"arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/autoscali
ng.amazonaws.com/AWSServiceRoleForAutoScaling"
   },
   "Action":[
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
      "kms:GenerateDataKey*",
      "kms:DescribeKey"
   ],
   "Resource":"*"
}{
   "Sid":"Allow EKS access to EBS.",
   "Effect":"Allow",
   "Principal":{
      "AWS":"*"
   },
   "Action":[
      "kms:CreateGrant",
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource":"*",
    "Condition":{
        "StringEquals":{
            "kms:CallerAccount":"[YOUR-ACCOUNT-ID]",
            "kms:viaService":"ec2.[YOUR-ACCOUNT-REGION].amazonaws.com"
        }
    }
}
```

2. Update the cross account role attached to the corresponding environment.

Overview of the setup steps

Configuring your environment to use a CMK involves the following steps:

1. Ensure that your provisioning credential has the minimum access permissions.
2. Ensure that your existing encryption key fulfills the required criteria or create a new encryption key according to the instructions provided in this document.
3. When creating an environment, specify the encryption key that should be used for encrypting the environment, including the Data Hubs running in it.

Note: Once your environment is running, if you would like to use a different key for encrypting a specific Data Hub, you can configure it as described in Encryption for Data Hub's EBS volumes on AWS.

Permissions for using encryption

If you are planning to use encryption, ensure that the cross-account IAM role used for the provisioning credential includes the following permissions:

EC2 permissions

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
      "ec2:CopyImage",
      "ec2:CreateSnapshot",
      "ec2:DeleteSnapshot",
      "ec2:DescribeSnapshots",
      "ec2:CreateVolume",
      "ec2:DeleteVolume",
      "ec2:DescribeVolumes",
      "ec2:DeregisterImage",
    ],
    "Resource": "*"
  }

}
```

KMS permissions

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:ListKeys",
      "kms:ListAliases"
    ],
    "Resource": "*"
  }

}
```

Encryption key requirements

If planning to use encryption, ensure that your encryption key can be used or create a new encryption key.

Ensuring that an existing encryption key can be used

If you already have an existing encryption key, make sure that the key fulfills the following requirements.

If you have an existing encryption key that you would like to use with Data Hub, make sure that:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- The following are attached as key user:
    - The *AWSServiceRoleForAutoScaling* built-in role.
    - Your IAM role or IAM user used for the cloud credential.
- To check that these are attached, in the AWS Management Console, navigate to the KMS console > Customer managed keys, select your encryption key, and scroll to Key Users.
- The encryption key is located in the same region where you would like to create clusters with encrypted volumes.

Create a new encryption key on AWS

If you don't have an existing encryption key, use the following instructions to create one.

1. In the AWS Management Console, navigate to KMS console.
2. Select Customer managed keys.
3. From the Region dropdown, select the region in which you would like to create and use the encryption key.
4. Click Create key.
5. In Step 1: Configure Key:
    1. Under Key type, choose Symmetric.
    2. Expand Advanced Options and under Key Material Origin, select “KMS” or “External”.
6. In Step 2: Create Alias and Description:
    1. Enter an Alias for your key.
    2. Defining Tags is optional.
7. In Step 3: Define Key Administrative Permissions, select the following:
    1. Choose your own IAM user / role used for logging into the AWS Management Console. Do not set *AWSServiceRoleForAutoScaling* or the cross-account IAM role as the key admin.
8. In Step 4: Define Key Usage Permissions:
    1. Select the *AWSServiceRoleForAutoScaling* built-in role.
    2. Select the cross-account IAM role.
9. In Step 5: Review and edit key policy, you may optionally tweak the key policy as desired, or simply leave it as generated by AWS.
10. Navigate to the last page of the wizard and then click Finish to create an encryption key.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Create an environment with a CMK

You can register your environment as described in [Register an AWS environment from CDP UI](), just make sure to specify the CMK that should be used to encrypt data, as described in the below steps.

Steps - CDP UI

1. Log in to the CDP web interface.
2. Navigate to the **Management Console > Environments**, and click **Register environment.**
3. Provide an **Environment Name.**
4. Select a **provisioning credential.**
5. Click **Next.**
6. Provide a **Data Lake Name.**
7. In the **Data Access and Audit** section, provide your data storage location and IAM resources created for minimal setup for cloud storage.
8. Click **Next.**
9. Select your region.
10. Under **Customer-Managed Keys**, click **Enable Customer-Managed Keys.**
11. In the same section, select the CMK:

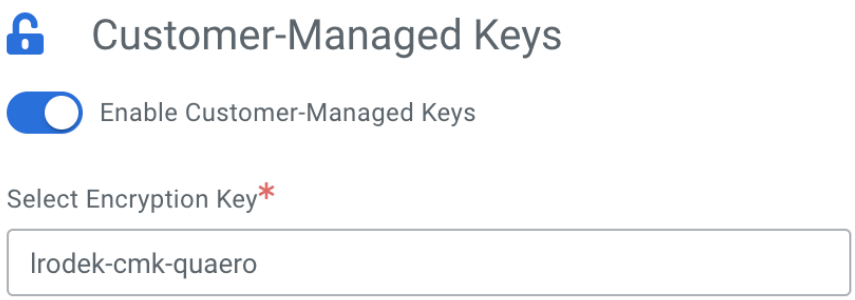

12. Select network, security groups, and provide an SSH key. If required, add tags.
13. Click **Next.**
14. In the **Logs** section, provide your logs storage location and managed identities created for minimal setup for cloud storage.
15. Click **Register Environment.**

Steps - CDP CLI

You can use your usual CDP CLI command to create an environment with a CMK, just add the **`--encryption-key-arn`** parameter and provide the encryption key created earlier. The easiest way to obtain the correct CLI template for creating an environment is by obtaining it from CDP wen UI as described in [Obtain CLI commands from the register environment wizard]().

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

For example:

```
cdp environments create-aws-environment \
--environment-name <ENVIRONMENT-NAME> \
--credential-name <EXISTING_CREDENTIAL> \
--region "<REGION>" \
--security-access cidr=<CIDR> \
--authentication publicKeyId="<SSH_KEY>" \
--log-storage storageLocationBase=<BUCKET_URL>,instanceProfile=<IDBROKER_IP>
\
--vpc-id <VPC_ID> \
--subnet-ids <SUBNETS \
--encryption-key-arn <ENCRYPTION_KEY_ARN>
```

The ARN of the encryption key created earlier should be passed in the parameter **`--encryption-key-arn`**

If the customer-managed encryption key ARN is not passed, then the AWS region-specific default encryption key is used for encrypting EBS volumes and RDS instances.

Set a CMK for an existing environment

You can set a CMK for an existing environment. The CMK is only used for encrypting disks of Data Hubs created after the CMK was added.

Note: The CMK added to an existing environment is only used for encrypting disks of Data Hubs created after the CMK was added. FreeIPA disks are not encrypted with the CMK. Data Lake and the external RDS are not encrypted except if they are created after the CMK was added.

Steps - CDP CLI

You can add an encryption key for an existing environment that does not yet have encryption enabled:

```
cdp environments update-aws-disk-encryption-parameters \
  --environment <ENVIRONMENT_NAME> \
  --encryption-key-arn <CMK_ARN> \
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

References

Instructions for using AWS IAM restricted Role and Policy for Compute(Liftie) & CML	https://docs.google.com/document/d/1zGIcYVF4Y8b8jsA2kvss5Qhb8WkS62peFpClBAovgM8
Customer master key - Concepts	https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#master_keys
Rotating customer master keys	https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html
Creating CMK	https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html
How to choose your CMK configuration	https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-choose.html

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Configuring a Proxy for a CDP Environment Using CCMv2 (Preview)

Date published: 2023-04-17
Date modified: 2023-04-17

Legal Notice

© Cloudera Inc. 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Configuring a proxy for a CDP environment using CCMv2

While creating a CDP environment, you can set up an HTTP CONNECT proxy such as Squid or a comparable product.

For a majority of simple use cases, this is enough to direct the traffic through a proxy. Since the proxy is not aware of the private keys used to secure the communication, in this setup the proxy is unable to perform a deep packet inspection.

Prerequisites

To allow TLS interception, you need to set your proxy to terminate TLS and add it back while relaying it to the agent. To do this, the account must have the CDP_CCM_V2_USE_ONE_WAY_TLS entitlement granted.

Steps

1. Create a new CDP environment.
2. After the FreeIPA nodes are running, SSH into the FreeIPA nodes and perform the following set of steps.
    1. Get the CA certificate from `/etc/jumpgate/config.toml` and grab the pinned CA certificate from the `agent.relayServerCertificate` parameter.
    2. Configure your proxy server to trust this certificate for the CCM traffic.
    3. Copy your proxy server's CA certificate and replace the contents of `agent.relayServerCertificate` in `/etc/jumpgate/config.toml`.
    4. Configure your proxy to start MITM-ing the underlying TLS connection.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Custom Table Coprocessors in Cloudera Operational Database (Preview)

Date published: 2022-02-10
Date modified: 2023-08-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
Custom table coprocessors in Cloudera Operational Database **4**
Adding custom coprocessors into HBase tables 4
Removing custom coprocessors from HBase tables 5
Obtaining a list of coprocessors from a COD environment 5
Verifying table coprocessors in HBase tables 6

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Custom table coprocessors in Cloudera Operational Database

You can add table coprocessors so that HBase can run custom code on the server side against the stored data and filter local minimum or maximum values during ingestion without scanning the entire table.

You can use built-in table coprocessors from the upstream HBase releases. COD supports custom table coprocessors, which you can implement and extend from HBase coprocessors' interfaces.

You can add custom table coprocessors into the HBase tables or remove them from HBase tables, also obtain a list of coprocessors with its status on the COD environment.

You must ensure that:

- You have the CDP CLI setup.
- You have uploaded the JAR file (custom coprocessor) to a remote location that the COD instance can access.

Important:
This feature is in Technical Preview and is not ready for production deployment. Cloudera encourages you to explore these technical preview features in non-production environments and provide feedback on your experiences.

Related information

- Coprocessor Introduction

Adding custom coprocessors into HBase tables

Learn how to add custom table coprocessors to the HBase table.

Steps

1. Launch the CDP CLI tool.
2. Upload your custom coprocessor to a remote location. For example, S3.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

3. Run the following **add-coprocessor** command to add a specific coprocessor using its canonical class name.

```
# add table-level coprocessor
cdp opdb add-coprocessor \
--environment ENVIRONMENT \
--database DATABASE  \
--table-name TABLE_NAME \
--coprocessor-canonical-name COPROCESSOR_CANONICAL_NAME \
[--coprocessor-location-url COPROCESSOR_LOCATION_URL] \
[--coprocessor-args COPROCESSOR_ARGS]
```

Removing custom coprocessors from HBase tables

Learn how to remove the custom coprocessors from the HBase tables.

Steps

1. Launch the CDP CLI tool.
2. Run the following **remove-coprocessor** command to remove a specific coprocessor using its canonical class name.

```
# remove table-level coprocessor
cdp opdb remove-coprocessor \
--environment ENVIRONMENT \
--database DATABASE \
--table-name TABLE_NAME \
--coprocessor-canonical-name COPROCESSOR_CANONICAL_NAME \
[--force] \
[--no-force]
```

Obtaining a list of coprocessors from a COD environment

Learn how to obtain a list of coprocessors from a COD environment.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Steps

1. Launch the CDP CLI tool.
2. Run the following **list-coprocessors** command to obtain a list of all the added coprocessors.

```
# Get a list of coprocessors with its status on the COD
environment
cdp opdb list-coprocessors \
--environment ENVIRONMENT \
--database DATABASE \
[--table-name TABLE_NAME] \
[--command-id COMMAND_ID]
```

Verifying table coprocessors in HBase tables

After you add or remove the table coprocessors through CDP CLI, you can verify whether they are added successfully.

Steps

1. Open the HBase UI from the Cloudera Manager.
2. Click **User Tables** under **Tables**.
3. Check the coprocessors' details in the **Description** column for the respective HBase tables.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Deploying Cloudera In Multiple GCP Availability Zones (Preview)

Date published: 2025-01-22
Date modified: 2025-01-22

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
Use cases 6
Limitations 7
Register a multi-AZ environment **7**
Cloudera UI 7
CDP CLI 8
Modify an environment to add AZs **9**
Cloudera UI 9
CDP CLI 9
Create a multi-AZ Cloudera Data Hub cluster **10**
Prerequisites 10
Cloudera UI 10
CDP CLI 10

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

You can optionally choose to deploy Data Lake, FreeIPA, and Cloudera Data Hub clusters across multiple availability zones (multi-AZ). With multi-AZ support, newly created GCP environments, enterprise Data Lakes and Cloudera Data Hub clusters using HA templates can be deployed across multiple availability zones of the selected GCP region. This provides fault tolerance during the extreme event of an availability zone outage.

Note: Only Enterprise Data Lakes can use multi-AZ; other Data Lake shapes do not support it. Enterprise Data Lake is only available in Cloudera Runtime 7.2.17 and newer; therefore, multi-AZ for Cloudera on GCP is only available in Cloudera Runtime 7.2.17 and newer. You need to contact Cloudera Support to have this feature enabled.

Each GCP region has multiple availability zones, which act as failure domains, preventing small outages from affecting entire regions. If you choose to deploy your Cloudera environment (FreeIPA and Data Lake) and Cloudera Data Hub clusters across multiple availability zones, each of these components is spread across the configured availability zones, providing high availability and fault tolerance. This is illustrated in the following diagram:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

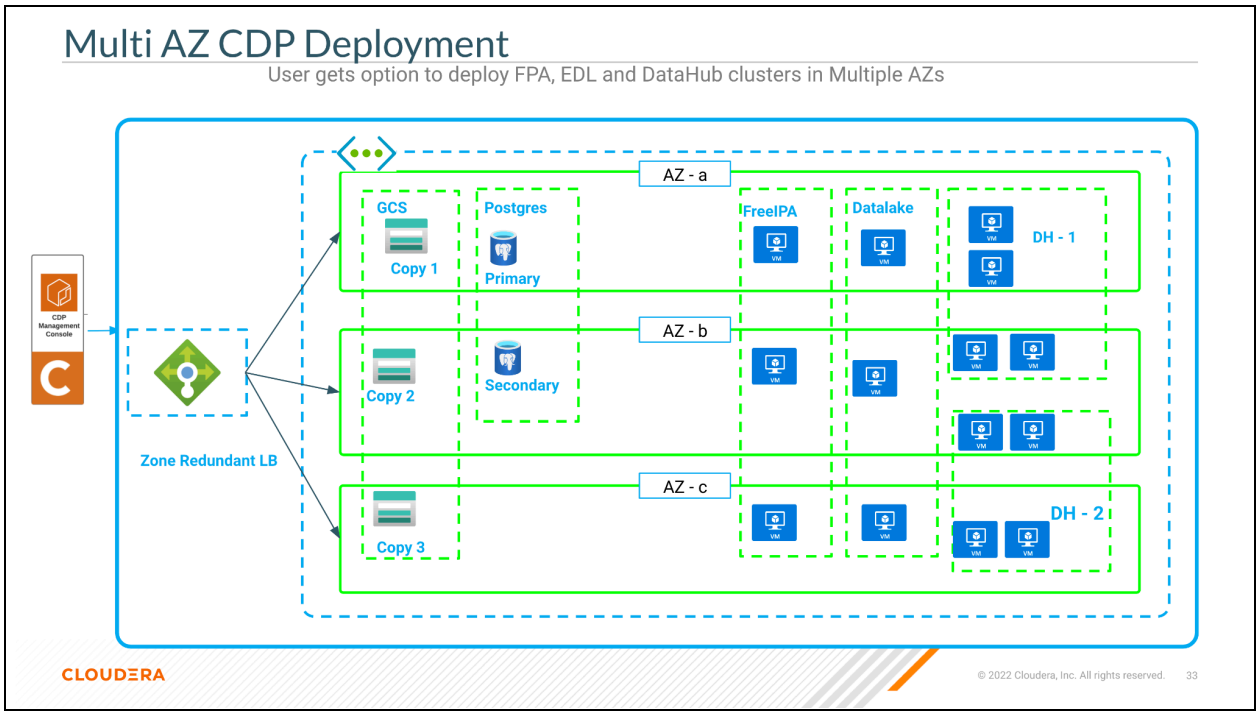

With the multi-AZ option enabled, your services are deployed in the following way:

- GCP environments are always created with the configured number of FreeIPA servers, deployed on virtual machines spread across the selected availability zones.
- In a GCP Enterprise Data Lake each host group is configured so that virtual machines of all critical services are spread across the selected availability zones.
- In Cloudera Data Hub clusters, virtual machines of each host group are evenly spread across the selected availability zones, following a round-robin logic.

When a zone failure happens and a cluster needs to be repaired, the replacement VMs are always provisioned in the same subnet and availability zone as the old ones since the detached disks can only be reattached to a VM in the same availability zone. This means that if there is an availability zone outage, cluster repair is not possible.

By default, If you do not enable multi-AZ, All GCP resources are placed in a fixed Availability Zone and the zone is either provided by the customer or selected randomly.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

When creating Cloudera Data Hub clusters via CDP CLI, you have the option to specify the AZ, which, in addition to allowing you to select the AZs that should be used, allows you to set up AZ targeting, where all nodes of the cluster are placed on the same AZ. This enables creating disaster recovery scenarios, where a primary and secondary cluster are running in different AZs. If an AZ outage occurs and the primary cluster is lost, it is guaranteed that the secondary cluster is not impacted.

Use cases

A multi-AZ Data Lake and FreeIPA constitute a resilient environment that provides a solid basis for multi-AZ Cloudera Data Hub clusters and Cloudera data services. Cloudera Data Hub clusters and Cloudera data services depend on the FreeIPA instance in the Data Lake to provide DNS resolution. Deploying FreeIPA across multiple availability zones ensures that critical DNS resolution is available in the event of an availability zone outage. Furthermore, a medium duty or enterprise Data Lake provides high availability, and additional compute and memory resources for key SDX services and is recommended for production workloads.

Deploying your Cloudera Data Hub clusters across multiple availability zones is key if your mission-critical applications depend on HBase and Kafka. Multiple availability zone deployment for operational workloads is considered best practice by the cloud vendors. It ensures that your applications can continue to run in the event of an availability zone outage.

When an entire availability zone fails, HBase automatically rebalances regions among the remaining instances in the cluster to maintain availability. The write-ahead log (WAL), which is replicated across the configured availability zones is automatically replayed by the newly assigned region servers in other availability zones to ensure writes to the database are not lost.

When using the multi availability zone feature, Cloudera ensures that Kafka replicates partitions across brokers in different availability zones. During an availability zone failure this ensures that no data is lost and applications can continue to access the data they need. Cruise Control, which is deployed alongside every Kafka cluster in Cloudera Public Cloud, detects that topics need to be rebalanced to the remaining brokers. Once the availability zone is back online, you can repair your Kafka cluster, restoring the initial broker distribution across availability zones. Afterwards Cruise Control kicks in and ensures that all topic partitions are balanced across the cluster.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Limitations

The following limitations apply when deploying a multi-AZ Cloudera:

- When an AZ is down, you cannot create a new Cloudera Data Hub cluster, and create or activate Cloudera data services within the environment. Existing workloads will continue to work.
- When an AZ is down, you cannot resize, stop, or restart Cloudera Data Hub clusters.
- Non-AZ environments or clusters cannot be converted to multi-AZ.

Register a multi-AZ environment

You can register a multi-AZ GCP environment via Cloudera UI or CDP CLI. You may choose to enable multi-AZ for Data Lake only or for FreeIPA only. There is no requirement to enable both.

Cloudera UI

Register your environment as usual, just make sure to do the following:

1. On the Data Access and Data Lake Scaling page:
    a. Select to use the Enterprise Data Lake.
    b. On the same page, scroll down and in the bottom of the page enable the **Advanced Options**.
    c. In the **Network and Availability** section enable the **Enable Multiple Availability Zones for Data Lake** toggle button in order to enable multi-AZ for Data Lake. The option is disabled by default. The option only appears when the Enterprise Data Lake is selected.
2. On the **Region, Networking, and Security** page:
    a. Scroll down and in the bottom of the page enable the **Advanced Options**.
    b. In the **Network and Availability** section enable the **Enable Multiple Availability Zones for Data Lake** toggle button in order to enable multi-AZ for FreeIPA. The option is disabled by default.
3. Finish registering your environment as usual.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

CDP CLI

Use the following CDP CLI commands to register an environment with a multi-AZ Data Lake and FreeIPA:

1. Register an GCP environment using the `cdp environments create-gcp-environment` command and include `multiAz=true` in the `--free-ipa` parameter as shown in this example:

```
Unset
cdp environments create-gcp-environment \
          --environment-name test-env \
          ...
          --free-ipa instanceCountByGroup=3,multiAz=true \
```

If you do not include the `multiAz=true`, the fixed AZ will be used.

You can also optionally include the `--availability-zones` parameter to select the specific availability zones that should be used. Valid value for availability zone is `<region>-<zone>`. `<zone>` is a letter like `a`, `b`, `c`, etc. Each region has a different number of availability zones. For example, the us-west2 region has zones `us-west2-a`, `us-west2-b`, and `us-west2-c`. If this parameter is not provided, all AZs are used. For example:

```
Unset
     cdp environments create-gcp-environment \
          --environment-name test-env \
          ...
          --free-ipa instanceCountByGroup=3,multiAz=true \
          --availability-zones us-west2-a,us-west2-b
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

2. Set IDBroker mappings as usual using the `cdp environments set-id-broker-mappings` command.
3. Create a Data Lake using the `cdp datalake create-gcp-datalake` command and adding the `--multi-az` parameter. For example:

```
Unset
     cdp datalake create-gcp-datalake \
          --datalake-name test-dl \
          --environment-name test-env \
          ...
          --scale ENTERPRISE \
          --runtime 7.2.17 \
          --multi-az
```

Modify an environment to add AZs

You can modify an environment to add more AZs to it. In this case, added AZs will only be used for new Cloudera Data Hub cluster clusters. Existing clusters will continue to use the original AZ.

Cloudera UI

1. In the Cloudera Management Console, navigate to environment details > **Summary**.
2. Scroll down to the **Region** section.
3. Under **Select GCP availability zones**, select the availability zones:
4. Click **Save.**

CDP CLI

Use the CLI command for updating AZs of an existing environment:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
Unset
cdp environments update-gcp-availability-zones \
--environment-name <ENV-NAME> \
--availability-zones <LIST-OF-AZs>

cdp environments update-gcp-availability-zones \
--environment-name test-env \
--availability-zones us-west2-a us-west2-b
```

Create a multi-AZ Cloudera Data Hub cluster

You can create multi-AZ Cloudera Data Hub clusters within any existing environment. Detailed steps are provided below.

Prerequisites

You can create a multi-AZ Cloudera Data Hub cluster in a multi-AZ environment only. If you are trying to create a multi-AZ Cloudera Data Hub cluster in an environment that uses the default AZ distribution, you need to first edit that environment and add AZs to it.

Cloudera UI

To enable multi-AZ when creating a Cloudera Data Hub cluster on GCP, navigate to the **Advanced Options > Network And Availability** and in the “GCP Availability Zones” section click the toggle button next to **Enable using multiple availability zones.**
.

CDP CLI

You can create a multi-AZ Cloudera Data Hub cluster by adding the `--multi-az` option to the Cloudera Data Hub cluster creation command.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

In the `--instance-groups` parameter, you can optionally include the `availabilityZones` to select the specific availability zones that should be used. If this parameter is not provided, all three AZs are used. For example:

Unset

```
cdp datahub create-gcp-cluster \
  --cluster-name test-cluster1 \
  --environment-name test-env \
  --cluster-template-name "7.2.17 - Data Engineering: Apache
Spark, Apache Hive, Apache Oozie" \
  --multi-az \
  --instance-groups
nodeCount=1,instanceGroupName=compute,instanceGroupType=CORE,inst
anceType=Standard_D5_v2,rootVolumeSize=100,attachedVolumeConfigur
ation=\[\{volumeSize=100,volumeCount=0,volumeType=StandardSSD_LRS
\}\],recoveryMode=MANUAL,availabilityZones=\[us-west2-a,us-west2-
b\]
nodeCount=0,instanceGroupName=gateway,instanceGroupType=CORE,inst
anceType=Standard_D8_v3,rootVolumeSize=100,attachedVolumeConfigur
ation=\[\{volumeSize=100,volumeCount=1,volumeType=StandardSSD_LRS
\}\],recoveryMode=MANUAL,availabilityZones=\[us-west2-b,us-west2-
c\]
nodeCount=1,instanceGroupName=master,instanceGroupType=GATEWAY,in
stanceType=Standard_D16_v3,rootVolumeSize=100,attachedVolumeConfi
guration=\[\{volumeSize=100,volumeCount=1,volumeType=StandardSSD_
LRS\}\],recoveryMode=MANUAL,availabilityZones=\[us-west2-a,us-wes
t2-b,us-west2-c\]
nodeCount=3,instanceGroupName=worker,instanceGroupType=CORE,insta
nceType=Standard_D5_v2,rootVolumeSize=100,attachedVolumeConfigura
tion=\[\{volumeSize=100,volumeCount=1,volumeType=StandardSSD_LRS\
}\],recoveryMode=MANUAL,availabilityZones=\[us-west2-a,us-west2-c
\]
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Disabling S3Guard in an Existing Cloudera Environment (Preview)

Date published: 2022-10-05
Date modified: 2024-12-04

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**

Contents **3**

Pre-requisites 4

How-To 4

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Pre-requisites

- The environment version should be Cloudera Runtime 7.2.2 or older.

How-To

On an existing Cloudera environment with S3Guard, disabling S3Guard requires the following manual steps. Note that the workloads may give inconsistent results while these steps are in progress. Stop all the workloads during this operation.

1. Download the latest Beta CDP CLI. For more information see *Installing Beta CDP CLI*.

   ```
   pip install cpdcli-beta
   ```

2. Write down the DynamoDB table associated with the environment, then go to the environment in the Cloudera UI. Click the **Summary** tab --> **Advanced**.

3. In the Beta CLI, run the command to disable S3Guard for the environment:

   ```
   cdp environments disable-s3-guard --environment-name
   <environment name>
   ```

   This command returns the entire environment response as output, and may contain the last environment status like “Error message: \"Cannot update the status of the stack to STOPPED when stack is in UPDATE_IN_PROGRESS state.\"". Note that this status is just the last environment status, and doesn’t imply the results of the `disable s3guard` operation. Verify that the returned environment response does not contain ‘s3guard’ under **awsdetails:{}.** This will be fixed soon in an upcoming release.

4. Data Lake: Go to **Cloudera Manager** –> **Clusters** --> **hdfs** –> **Configuration** –> **Cluster-wide Advanced Configuration Snippet (Safety Valve)** for `core-site.xml`. Update the following config:

   ```
   fs.s3a.metadatastore.impl=org.apache.hadoop.fs.s3a.s3guard.Null
   MetadataStore
   ```

   Restart all the services in the Data Lake.

5. For each of the Datahubs: Go to **Cloudera Manager** –> **Clusters** -> **hdfs** --> **Configuration** –> **Cluster-wide Advanced Configuration Snippet (Safety Valve)** for

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

`core-site.xml`. Update the following config:

```
fs.s3a.metadatastore.impl=org.apache.hadoop.fs.s3a.s3guard.Null
MetadataStore
```

Restart all the services in the Cloudera Data Hub cluster.

6. Verify that S3guard is disabled:

```
cdp environments describe-environment --environment-name
<environment name>
```

The output shouldn't contain a section with "S3GUARD" under **awsdetails:{}**

7. *Cloudera Data Warehouse*: For both 'SDX Datalake' and 'Cloudera Data Warehouse Datalake', you can disable S3Guard by editing the configs for `core-site.xml` in Data Catalog and Virtual Warehouse. Go to **Edit** –> **Configurations** →, click on each component, and add the value:

```
fs.s3a.metadatastore.impl=org.apache.hadoop.fs.s3a.s3guard.Null
MetadataStore
```

`to` **both** `hadoop-core-site` and `hadoop-core-site-default-warehouse`. This should restart the Virtual Warehouse and Database Catalog.

8. *Cloudera Data Engineering*: The S3Guard config will be updated automatically every hour, synchronized with config-manager pod start.

9. *Cloudera AI*: With Cloudera AI 2.0.19, S3Guard configs will be updated automatically.

10. Verify that your workloads run successfully.

11. Drop the DynamoDB table from the AWS console. You should have the table name from Step 1 above.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Disk Vertical Scaling in Azure (Preview)

Date published: 2023-12-12
Date modified: 2024-12-04

Legal Notice

© Cloudera Inc. 2024, 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
About this feature **4**
Introduction 4
Limitations 4
Azure permissions 4
Download Beta CDP CLI **5**
Modify Disks **5**
Modifying disks using CDP CLI 5
Verifying that modification is complete 6
Verifying disk size/type in Cloudera UI 6
Add Disks **7**
Adding disks using CDP CLI 7
Deleting Disks **8**
Deleting disks using CDP CLI 8

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

About this feature

Introduction

There are many clusters that are deployed with standard magnetic storage. With the growing lineage data, these disks are running out of space on core nodes. These need to be moved to General Purpose SSDs and or resized to a bigger disk.

The disks attached to the Data Lake and Cloudera Data Hub clusters can be changed or resized Azure without downtime.

Limitations

When using this preview feature, be aware of the following limitations:

- This feature is only available for AZURE.
- The disks can only be resized up, meaning you cannot reduce the size of an attached block storage. If there are multiple disks of different sizes, the size of all the disks attached to the instances in the group that are smaller or lesser than the requested size will be increased to the requested size.
- This feature will only resize additional block storages in an instance and not the root volume.
- Clusters and cluster services must be in running state before disk vertical scaling is performed.
- Current implementation does not support this feature through Cloudera UI; It is available only through Beta CDP CLI.
- Disk type change is not supported for AZURE.
- Not supported for Ultra disks or Premium SSD v2 disks.
- Not supported for shared disks.
- Not available on some classic VMs.
- If a disk is 4 TB or less, you should deallocate your VM and detach the disk before expanding it beyond 4 TB. If a disk is already greater than 4 TB, you can expand it without deallocating the VM and detaching the disk, as mentioned in Expand virtual hard disks on a Linux VM.
- If the above points aren't satisfied, resizing will require downtime; That is, Cloudera will have to deallocate (stop VM), and then resize and start VM.

Azure permissions

This feature requires the following permissions to be added to the minimal set of permissions described in Prerequisites for the provisioning credential:

- Microsoft.Compute/disks/write

This permission allows users to update disk sizes on a resource.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Download Beta CDP CLI

You have to use Beta CDP CLI to modify disks attached to an instance group. To install Beta CDP, refer to Installing Beta CDP CLI.

Modify Disks

Modifying disks using CDP CLI

Use the following Beta CDP CLI command to modify the volumes attached to the instances in a host group. Replace the placeholders with actual values. For example <DATALAKE-NAME> should be replaced with an actual name. As part of this update, the instance-template parameter in the vertical scaling command has been made optional. But one of the instance-template or disk-options have to be provided.

Additional parameters for vertical scaling have been added to both the datalake and datahub commands. The modification request is sent as part of the --disk-options parameter.

The <VOLUME-TYPE> placeholder is for the type of volume the disks are being modified to. It is an optional field and should be added if the volume type has to be modified, in which case the type will not be modified.

The <SIZE> placeholder is optional as well and is for the size the disks are being increased to in GB. Please specify only the integer value.

```
//DATA LAKE
cdp datalake start-datalake-vertical-scaling
          --datalake <DATALAKE-NAME>
          --group <INSTANCE-GROUP-NAME>
          --disk-options
modifyDisks="{volumeType=\"<VOLUME-TYPE>\",size=<SIZE>}"



//DATA HUB

cdp datahub start-cluster-vertical-scaling
          --datahub <DATAHUB-NAME>
          --group <INSTANCE-GROUP-NAME>
          --disk-options
modifyDisks="{volumeType=\"<VOLUME-TYPE>\",size=<SIZE>}"
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Verifying that modification is complete

The change to the disk can be verified through Cloudera UI, or by logging into the instances directly, after the flow is completed in the Event History.

Here is a screenshot from Event History showing the completion of the disk update:

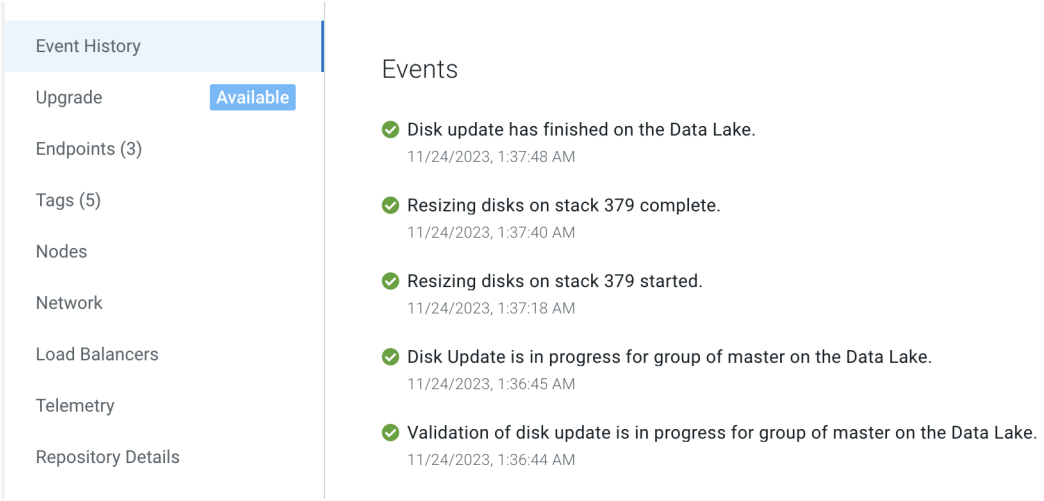

Verifying disk size/type in Cloudera UI

1. Navigate to the **Management Console > Data Lake** or **Management Console > Data Hub.**
2. Click into the Data Lake / Cloudera Data Hub cluster that was modified.
3. Click on the **Nodes** in the left hand tree.
4. Open accordion of the instance group that was modified.
5. Open **Storage Settings** accordion in any of the instances in the group.
6. The Storage Type and Volume Size should be updated based on the request.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

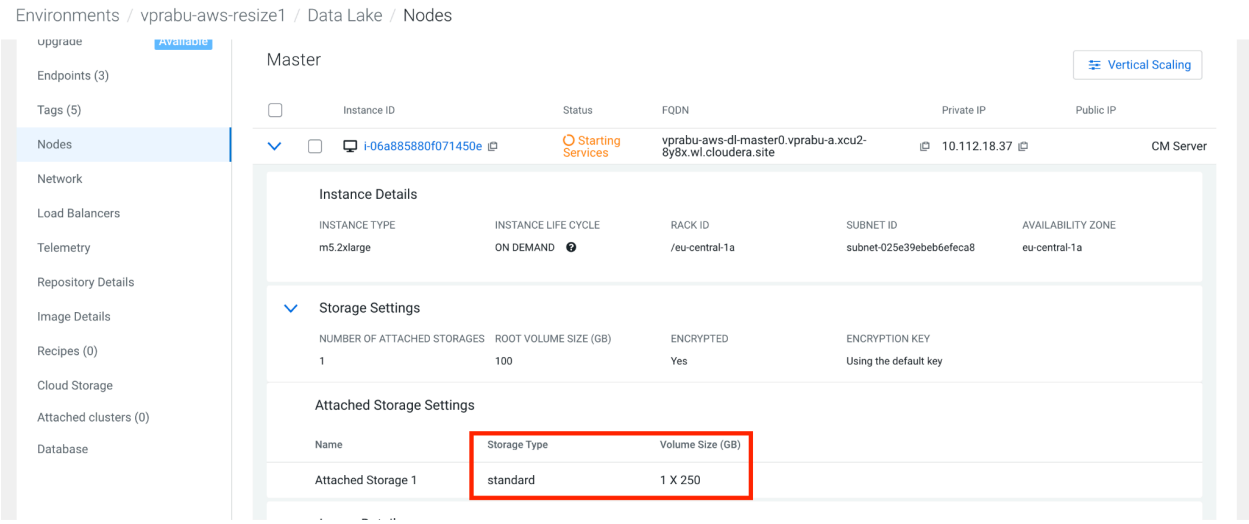

Add Disks

In some cases when customers run out of memory, it is cheaper to add an additional volume instead of resizing to a bigger volume. This feature allows for adding additional block storages to an instance group with minimum interruption. Only the services running on the instance group are stopped, as the additional volume has to be mounted and configured for use on the instances.

Based on the request, ‘N’ number of disks of the same type and size can be added to the instance group.

Adding disks using CDP CLI

Use the following CDP CLI Beta command to add additional volumes attached to the instances in a host group. Replace the placeholders with actual values. For example <DATALAKE-NAME> should be replaced with an actual name. As part of this update, the instance-template parameter in the vertical scaling command has been made optional. But one of the instance-template or disk-options have to be provided.

The add disks request is sent as part of the `--disk-options` parameter. All parameters for the add disk input are required.
The <NUMBER_OF_DISKS> placeholder is for the number of volumes being added.
The <VOLUME-TYPE> placeholder is for the type of volume being added.
The <SIZE> placeholder is for the size the disks being added in GB. Please specify only the integer value.
The cloudVolumeUsageType is the purpose for which the disk is being added - it is an enum field with either “GENERAL” or “DATABASE” value.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
//DATALAKE

cdp datalake start-datalake-vertical-scaling --datalake=<DATALAKE_NAME>
--group=<DATALAKE_INSTANCE_GROUP_NAME> --disk-options
addDisks="{numberOfDisks=<NUMBER_OF_DISKS_TO_ADD>,
volumeType=\"<TYPE_OF_COLUME_TO_ADD>\",size=<SIZE_OF_VOLUME>,cloudVolume
UsageType=\"GENERAL\"|\"DATABASE\"}"


//DATAHUB

cdp datahub start-cluster-vertical-scaling --datahub=<DATAHUB_NAME>
--group=<DATAHUB_INSTANCE_GROUP_NAME> --disk-options
addDisks="{numberOfDisks=<NUMBER_OF_DISKS_TO_ADD>,
volumeType=\"<TYPE_OF_COLUME_TO_ADD>\",size=<SIZE_OF_VOLUME>,cloudV
olumeUsageType=\"GENERAL\"|\"DATABASE\"}"
```

Deleting Disks

In cases where only compute services are being run on an instance group, any block storage attached to the instance is going to cost the customer even if the cluster is stopped. As compute services do not store any persistent data on attached volumes, having additional volumes to store temporary data is unnecessary. In such cases, customers can use this command to delete all attached volumes on instances in an instance group.

This can be done only for compute instance groups on only datahubs, as datalakes need persistent volumes. This command deletes all additional volumes for instances in an instance group.

Deleting disks using CDP CLI

Use the following CDP CLI Beta command to deleting all attached additional volumes to the instances in a host group. Replace the placeholders with actual values. For example <DATALAKE-NAME> should be replaced with an actual name. As part of this update, the instance-template parameter in the vertical scaling command has been made optional. But one of the instance-template or disk-options have to be provided.

The delete disks request is sent as part of the --disk-options parameter. All parameters for the add disk input are required.
The <`deleteDisks`> parameter accepts a boolean value with true/ false.

```
//DATAHUB
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
cdp datahub start-cluster-vertical-scaling --datahub=<DATAHUB_NAME>
--group=<DATAHUB_INSTANCE_GROUP_NAME> --disk-options
deleteDisks=true|false
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Deploying Hue at Environment Level in Data Warehouse Public Cloud (Preview)

Date published: 2023-11-20
Date modified: 2023-11-28

Legal Notice

© Cloudera Inc. 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2
Contents 3
About deploying Hue at the environment level 3
Advantages of deploying Hue at the environment level 4
Limitations 4
Understanding access control 4
FAQs 5

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

About deploying Hue at the environment level

Cloudera Data Warehouse (CDW) allows you to deploy Hue at an environment level. Learn about the advantages and upgrade limitations of deploying Hue at the environment level and some FAQs that can help you understand more about the feature.

Note: This feature is in technical preview and not recommended for production deployments. Cloudera recommends that you try this feature in test and development environments.

Advantages of deploying Hue at the environment level

By deploying Hue at the Virtual Warehouse level, you can manage costs by keeping only those Virtual Warehouses running which your users need at that time. Data Analysts only need to know or bookmark one Hue instance URL and can run queries on any Virtual Warehouses available to them.

If you want to isolate the Hue instances, and thereby, the query history and saved queries for different teams within your organization, then by deploying Hue at the environment level, you can create multiple Hue instances, each with its own database.

Moreover, the Hue instance deployed at the environment level remains active as long as the environment is active. Hue instances, along with the query history and saved queries are deleted when you delete or shutdown a Virtual Warehouse. You can spin up temporary Virtual Warehouses without causing disruption to the Data Analysts.

Limitations

- In CDW version 1.8.1, you cannot upgrade the Hue instances that are deployed at the environment level.
- Currently, you cannot move or migrate saved queries and query history from a Virtual Warehouse-level Hue to an environment-level Hue.
- When you use the Importer to create tables from files in Hue, by default, Hue creates a Hive table if Hive is available and uses the first Hive Virtual Warehouse that was created. You cannot select a Virtual Warehouse using which you want to create a table by importing a file.
  To create an Impala table using the Importer, you must first select the editor type as Impala, and then click **+** on the **Table Browser**.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Understanding access control

You can specify user groups that you have created in the CDP Management Console, similar to how you specify while creating the Virtual Warehouses.

When you specify user groups while creating the Hue instance (at the environment level) or Virtual Warehouses, the subset of users who have access to the Hue instance as well as the Virtual Warehouse can submit queries through that Virtual Warehouse instance.

If you do not specify user groups while creating a Hue instance (at the environment level), then all users within your organization can access the Hue UI. If you do not specify user groups for a Virtual Warehouse, all users within your organization can submit queries through that Virtual Warehouse.

As a best practice, specify user groups while creating Hue and Virtual Warehouse instances so that specific users have access to specific compute resources.

FAQs

Can I still use Hue deployed at the Virtual Warehouse level?

Yes, you can continue to access and use Hue from a particular Virtual Warehouse even after deploying Hue at the environment level. However, you cannot view the query history or saved queries in Hue which is deployed at the environment level.

Can I create more than one Hue instance at the environment level?

Yes, you can create any number of Hue instances. However, Cloudera recommends that you deploy a single Hue unless isolating saved queries is a requirement.
When you create more than one Hue instance, each instance has its own database. The Hue instances deployed at the environment level do not share query history or saved queries.

Can I view queries submitted from other BI tools?

Hue superusers and administrators can view all queries that are submitted from all Virtual Warehouses linked to a particular Database Catalog. Other logged-in users can view only their queries on the **Impala Queries** and **Hive Queries** tabs.

Where can I specify advanced Hue configurations (safety valve)?

On a Hue deployed at the environment level, click the more options icon > **Edit** corresponding to the Hue instance that you want to configure, go to the **CONFIGURATIONS** tab, and select **hue-safety-valve** from the **Configuration files** drop-down menu.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Enabling Hue to run at the environment level

By deploying Hue at the environment level, you can retain the Hue instance with its query history and saved queries even if the Virtual Warehouses are deleted. You can enable this option by editing the environment configuration.

Steps

1. Log in to the Data Warehouse service as a DWAdmin.
2. Go to the **Environment(s)** tab, locate the environment in which you want to enable this feature, and click the More Options icon > **Edit**.
3. Go to the **CONFIGURATIONS** tab, select the **Enable Query Editor** option, and click **Apply Changes**.
   A new **Query Editor** option is added on the left navigation pane on the CDW UI.

Creating a Hue instance at the environment level in CDW

After you enable the option to deploy Hue at the environment level, you can create any number of Hue instances. Each Hue instance has its own database. The Hue instances deployed at the environment level do not share query history or saved queries.

Steps

1. Log in to the Data Warehouse service as a DWAdmin.
2. Click the **Query Editor** option from the left navigation pane of the CDW UI.
3. Create a query editor instance by clicking **Create** on the **Query Editor** page.
   The **New Query Editor** modal is displayed.
4. Specify a name for your query editor instance and select an environment from the drop-down menu.
   You can optionally specify user groups, the same as what you specify while creating a Virtual Warehouse for fine-grained access control.
5. Select the size of the Hue instance.
   The default value is **1 x Hue Node**. This indicates the number of Hue backend pods you want to create.
6. Click **Create**.
   A new Hue instance is created at the environment level. You can select the Virtual Warehouse you want to use from the **Virtual Warehouse** drop-down menu on the Hue web interface.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Deploying shared Hue service in Data Warehouse Public Cloud (Preview)

Date published: 2023-11-20
Date modified: 2024-07-26

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2
Contents 3
About deploying the shared Hue service 4
Advantages of deploying a shared Hue service 4
Limitations 4
Access control for the shared Hue service 4
Key differences in database management approach for Virtual Warehouse-level Hue and shared Hue service 5
Enabling the shared Hue service 5
Creating a shared Hue instance 6
Rebuilding a shared Hue service 7
Upgrading a shared Hue instance 7
FAQs 7
- Can I still use Hue which is deployed at the Virtual Warehouse level? 7
- Can I create more than one shared Hue service instance? 8
- Can I view queries submitted from other BI tools? 8
- Where can I specify advanced Hue configurations (safety valve) for the shared Hue instance? 8

About deploying the shared Hue service

Cloudera Data Warehouse (CDW) allows you to deploy a shared Hue service at an environment level. Learn about the advantages and limitations of deploying a shared Hue service and some FAQs that can help you understand more about the feature.

Note: This feature is in technical preview and not recommended for production deployments. Cloudera recommends that you try this feature in test and development environments.

Advantages of deploying a shared Hue service

By deploying a shared Hue service, you can manage costs by keeping only those Virtual Warehouses running which your users need at that time. Data Analysts only need to know or bookmark one Hue instance URL and can run queries on any Virtual Warehouses available to them.

Each shared Hue service instance has its own database where queries and query history is saved. You can create isolation for different teams within your organization by deploying multiple shared Hue instances.

Moreover, the shared Hue service remains active as long as the environment is active.

Limitations

- When you use the Importer to create tables from files in Hue, by default, Hue creates a Hive table if Hive is available and uses the first Hive Virtual Warehouse that was created. You cannot select a Virtual Warehouse using which you want to create a table by importing a file.
  To create an Impala table using the Importer, you must first select the editor type as Impala, and then click **+** on the **Table Browser**.

Access control for the shared Hue service

You can specify user groups you created in the CDP Management Console, similar to how you specify while creating the Virtual Warehouses.

When you specify user groups while creating the shared Hue instance or Virtual Warehouses, the subset of users who have access to the Hue instance as well as the Virtual Warehouse can submit queries through that Virtual Warehouse instance.

If you do not specify user groups while creating a shared Hue instance, then all users within your organization can access the Hue UI. If you do not specify user groups for a Virtual Warehouse, all users within your organization can submit queries through that Virtual Warehouse.

As a best practice, specify user groups while creating Hue and Virtual Warehouse instances so that specific users have access to specific compute resources.

Key differences in database management approach for Virtual Warehouse-level Hue and shared Hue service

All Hue instances linked to a Database Catalog through Virtual Warehouses within a CDW environment share a single database. The Hue database is not deleted unless you deactivate the environment. If you delete a Virtual Warehouse and create a new one, the Hue instance linked to that Virtual Warehouse continues to display old query history and saved queries.

Each shared Hue service instance has its Hue database. CDW does not delete the Hue database when you delete the shared Hue service instance. The Hue database exists in the backend until a database administrator manually deletes it. Each Hue database is named after the shared Hue service name. In case you have deleted a shared Hue service instance, you can reuse the Hue database by specifying the name of the Hue instance you deleted. This brings back the query history and saved queries.

CDW provides you a one-time option to copy the Hue database content from the Hue database linked to a Database Catalog to the shared Hue service database while creating a new shared Hue service instance. The data between the two databases is not synchronized after the initial copy event.

Enabling the shared Hue service

By deploying a shared Hue service, you can retain the Hue instance with its query history and saved queries even if the Virtual Warehouses are deleted. You can enable this option by editing the environment configuration.

Steps

1. Log in to the Data Warehouse service as a DWAdmin.
2. Go to the **Environment(s)** tab, locate the environment in which you want to enable this feature, and click the More Options icon > **Edit**.

3. Go to the **CONFIGURATIONS** tab, select the **Enable Shared Hue Service** option, and click **Apply Changes**.
   A new **Shared Hue Service** option is added on the left navigation pane on the CDW UI.

Creating a shared Hue instance

After you enable the option to deploy Hue at the environment level, you can create any number of Hue instances. Each Hue instance has its own database. The Hue instances deployed at the environment level do not share query history or saved queries.

Steps

1. Log in to the Data Warehouse service as a DWAdmin.
2. Click the **Shared Hue Service** option from the left navigation pane of the CDW UI.
3. Create a shared Hue instance by clicking **ADD NEW** on the **Shared Hue Service** page.
   The **Create Shared Hue Service** modal is displayed.
4. Specify a name for your Hue instance and select an environment from the drop-down menu.
   a. Select a size for the shared Hue instance from the **Size** drop-down menu.
      The default value is **1 x Hue Node**. This indicates the number of Hue backend pods you want to create.
   b. Select one of the following options from the **Select the Hue database initialization strategy** drop-down menu to initialize a database for this Hue instance.
      i. **Reuse if Hue data is present**: Select this option to use an existing Hue database for your new shared Hue instance.
      ii. **Copy Virtual Warehouse database**: Select this option to copy the contents of the Hue database within a Database Catalog into the shared Hue service database. This helps you to copy the shared queries and query history.
      Note: This is a one-time copy operation. After the data is copied from the Database Catalog to the shared Hue database, the data between the two databases is not synchronized.
      iii. **Flush the current database content**: Select this option to create a new database for the shared Hue instance.
      Select this option to recreate a shared Hue service instance with the same name, but a fresh database. The existing database linked to the Hue instance is dropped.

   You can optionally specify user groups and tags in the same way you specify while creating a Virtual Warehouse for fine-grained access control.
5. Click **Create**.
   A new shared Hue service instance is created.

After you finish

Click **Editor** to open Hue.

You can select the Virtual Warehouse you want to use from the **Virtual Warehouse** drop-down menu on the Hue web interface.

Rebuilding a shared Hue service

The rebuild operation deletes and recreates the Hue pods while preserving Hue's image version and configurations. By rebuilding the Hue service, you fix pods that are in a bad state, thereby improving performance.

Steps

1. Log in to the Data Warehouse service as DWAdmin.
2. Click **Shared Hue Service** from the left navigation pane.
3. Locate the shared Hue service instance you want to rebuild and click the More icon > **Rebuild**.
4. Review the message on the **Review Shared Hue Service** modal and click **Review Shared Hue Service**.
   The "Rebuild in progress" message is displayed.

Upgrading a shared Hue instance

If you are on an older version of the shared Hue service, you can upgrade the Hue image version by upgrading Hue from the Shared Hue Service page in Cloudera Data Warehouse.

Steps

1. Log in to the Data Warehouse service as DWAdmin.
2. Click **Shared Hue Service** from the left menu.
3. Locate the shared Hue service instance you want to upgrade and click **Upgrade**.
   The **Upgrade Shared Hue Service** modal is displayed.
4. Click **Upgrade**.
   The shared Hue service is upgraded to the latest available image version.

FAQs

Can I still use Hue which is deployed at the Virtual Warehouse level?

Yes, you can continue to access and use Hue from a particular Virtual Warehouse even after deploying the shared Hue service at the environment level.

Can I create more than one shared Hue service instance?

Yes, you can create any number of shared Hue service instances. However, Cloudera recommends deploying a single Hue unless isolating saved queries is a requirement. When you create multiple shared Hue instances, each instance has its own database. The shared Hue service instances do not share query history or saved queries.

Can I view queries submitted from other BI tools?

Hue superusers and administrators can view all queries submitted from all Virtual Warehouses linked to a Database Catalog. Other logged-in users can view only their queries on the **Impala Queries** and **Hive Queries** tabs.

Where can I specify advanced Hue configurations (safety valve) for the shared Hue instance?

On the **Shared Hue Service** page, click the more options icon > **Edit** corresponding to the Hue instance that you want to configure, go to the **CONFIGURATIONS** tab, and select **hue-safety-valve** from the **Configuration files** drop-down menu.

Using Hive Data Connectors to support External Data Sources in Data Warehouse Public Cloud (Preview)

Date published: 2023-11-20
Date modified: 2023-11-20

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **3**
Contents **4**
Using Hive data connectors to support external data sources **5**
What is a data connector 5
Benefits of using data connectors 5
Creating a remote database using a data connector **6**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Using Hive data connectors to support external data sources

Cloudera Data Warehouse allows you to use Hive data connectors to map databases present in external data sources to a local Hive Metastore (HMS). The external data sources can be of different types, such as MySQL, PostgreSQL, Oracle, Redshift, or Derby. You can create external tables to represent the data, and then query the tables.

***Note**: This feature is in technical preview and not recommended for use in production deployments. Cloudera recommends that you try this feature in test and development environments.*

What is a data connector

Data connectors in Hive are objects that are defined using a set of properties, such as hostname, user credentials, and type that are required to connect Hive to the external data source. You define a connector and then use the same connector to map multiple databases from the remote data source to the local HMS.

Benefits of using data connectors

Currently, you can use a JDBCStorageHandler to connect Hive to external data sources. Users define a table in the HMS for which data resides in a remote JDBC data store. The Hive table's metadata is persisted locally in the HMS. When HiveServer (HS2) runs a query against this table, data is retrieved from the remote JDBC table. However, there are certain limitations which make it difficult to map databases having a large number of tables.

- Each remote table in the remote data source has to be individually mapped to a local Hive table. It is cumbersome when you have to map an entire database having a lot of tables.
- New tables created in the remote data source are not automatically visible and should be manually mapped in Hive.
- The metadata for the mapped table is static. It does not track changes to the remote table. If the remote table is modified (added columns or dropped columns), the mapped Hive table has to be dropped and recreated. This is not feasible for organizations where tables are constantly changing.

With data connectors, you can map a database or schema in the remote data source to a Hive database. Such databases are referred to as 'Remote' databases in Hive. The benefits of creating remote databases using data connectors are as follows:

- Tables within a mapped database are automatically visible from Hive. The metadata for these tables are not persisted in the HMS. They are retrieved at runtime through an active connector.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- New tables created are automatically visible in Hive.
- The columns and their data types are mapped to compatible Hive data types during runtime. Therefore, metadata changes to tables in the remote datasource are immediately visible in hive.
- The same connector can be used to map another database to a Hive database. All the connection information is shared between these remote databases.

To create a remote database, you must first define a data connector with the required properties and then use this connector to create and map a remote database in an external data source to Hive.

Hive currently supports the following data connector types - 'mysql', 'postgres', 'oracle', 'derby', 'mssql', and 'hivejdbc'. Use the appropriate data connector type to map databases present in the corresponding external data sources.

The 'hivejdbc' connector helps you achieve SQL query federation between two distinct Hive clusters or between Hive and Hive-like compute engines, for example, Amazon EMR. For more information, see [Data Connector for Hive and Hive-like engines](#).

Creating a remote database using a data connector

Learn how to create a Hive data connector, which you can then use to create and map a remote database to Hive. The remote database can reside in an external data source, such as MySQL, PostGreSQL, Oracle, Redshift, or Derby.

Prerequisites:
You must ensure that the `hive.security.temporary.authorization.for.data.connectors` property is set to "true".
In the Hive virtual warehouse details page, go to **Configurations** > **Hiveserver2** > **hive-site** configuration file and click + to add this property as a custom configuration.

Steps:

1. Create a data connector using the following syntax:

   Syntax:

   ```
   CREATE CONNECTOR [IF NOT EXISTS] connector_name
         [TYPE datasource_type]
         [URL datasource_url]
         [COMMENT connector_comment]
         [WITH DC PROPERTIES (property_name=property_value, ...)];
   ```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Example:

If you are using a cleartext password:

```
CREATE CONNECTOR postgres_local TYPE 'postgres' URL
'jdbc:postgresql://localhost:5432' WITH DC PROPERTIES
("hive.sql.dbcp.username"="postgres",
"hive.sql.dbcp.password"="postgres");
```

If you are using a Java keystore instead of a cleartext password:

```
CREATE CONNECTOR postgres_local_ks TYPE 'postgres' URL
'jdbc:postgresql://localhost:5432' WITH DC PROPERTIES
("hive.sql.dbcp.username"="postgres",
"hive.sql.dbcp.password.keystore"="jceks://app/local/hive/secrets.jc
eks",
"hive.sql.dbcp.password.key"="postgres.credential");
```

Note: The example provided above shows a data connector created for a Postgres data source type. Hive currently supports the following data connector types - 'mysql', 'postgres', 'oracle', 'derby', 'mssql', and 'hivejdbc'. Use the appropriate data connector type to map databases present in the corresponding external data sources.

2. Create a REMOTE database in Hive using the data connector created in the previous step.

   Syntax:

```
CREATE [REMOTE] (DATABASE) [IF NOT EXISTS] database_name
      [COMMENT database_comment]
      [USING connector_name]
      [WITH DBPROPERTIES (property_name=property_value, ...)];
```

   Example:

```
CREATE REMOTE DATABASE postgres_hive_test USING postgres_local WITH
DBPROPERTIES ("connector.remoteDbName"="remote_db_test");
```

   This statement maps a remote database named "remote_db_test" to a Hive database named "postgres_hive_test" in Hive.

3. You can now use the tables that are available in the remote database.

   Example:

```
USE remote_db_test;
SHOW TABLES;
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
DESCRIBE [formatted] <tablename>;
SELECT <col1> from <tablename> where <filter1> and <filter2>;
```

Important:

- The metadata for these tables are not persisted in HMS.
- CREATE, ALTER, and DROP table DDLs are currently not supported in remote databases.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

SQL AI Assistant in Data Warehouse Public Cloud (Preview)

Date published: 2023-11-20
Date modified: 2023-11-28

Legal Notice

© Cloudera Inc. 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
About the Hue SQL AI Assistant **4**
Which AI models and services does Hue use? **4**
What data is shared with the LLM models? **5**
Prerequisites for enabling the SQL AI Assistant **5**
Configuring the SQL AI Assistant on the Microsoft Azure OpenAI Service **5**
Configuring the SQL AI Assistant on the OpenAI Service **7**
Service and model-related configurations for setting up the SQL AI Assistant in CDW **9**
How to use the SQL AI Assistant? **10**
Generating SQL from NQL 10
Editing the query in natural language 11
Getting an explanation of a SQL query in natural language 11
Optimizing a query 12
Fixing a SQL query 13
Limitations **14**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

About the Hue SQL AI Assistant

A SQL AI Assistant has been integrated into Hue with the capability to leverage the power of Large Language Models (LLMs) for various SQL tasks. It helps you to create, edit, optimize, fix, and succinctly summarize queries using natural language and makes SQL development faster, easier, and less error-prone. SQL AI assistant is available with the Hue image version 2023.0.16.0 and higher on the public cloud. Both Hive and Impala dialects are supported.

Note: *The Hue SQL AI assistant is in technical preview and not recommended for use in production deployments. Cloudera recommends that you try this feature in test and development environments.*

Which AI models and services does Hue use?

The AI Assistant supports various LLMs and hosting services. The model can run in the infrastructure of a service provider, and the AI Assistant can be configured to use them remotely. Cloudera has tested with GPT running in Azure OpenAI, and the following service-model combinations are supported

Service Provider	Model	Model versions
OpenAI	OpenAI GPT	gpt-3.5-turbo gpt-3.5-turbo-16k
Microsoft Azure	OpenAI GPT	gpt-3.5-turbo gpt-3.5-turbo-16k
Amazon Bedrock	Anthropic Claude	anthropic.claude-v1 anthropic.claude-v2
Amazon Bedrock	Amazon Titan	amazon.titan-text-express-v1

Note: *Cloudera recommends using the Hue AI assistant with the Azure OpenAI service.*

For augmenting the results, the SQL AI Assistant uses a Retrieval Augmented Generation (RAG)-based architecture. It uses the sentence-transformer library for semantic search, and Hue can be configured with any of these pre-trained models for better multi-lingual support. By default, **all-MiniLM-L6-v2** models are used.

Embedding Model	Language Support
all-MiniLM-L6-v2	English only

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

distiluse-base-multilingual-cased-v1	Arabic, Chinese, Dutch, English, French, German, Italian, Korean, Polish, Portuguese, Russian, Spanish, and Turkish.

What data is shared with the LLM models?

The following details are shared with the Large Language Models (LLMs):

- Everything that a user inputs
- Dialect in use
- Table details such as table name, column names, column data types and related keys, partitions, and constraints that the logged-in user has access to.
- Three sample rows from the tables (as per the best practices specified in *Rajkumar et al, 2022*)

Prerequisites for enabling the SQL AI Assistant

As an administrator, you must obtain clearance from your organization's infosec team to make sure it is safe to use the SQL AI Assistant because some of the table metadata and data, as mentioned in the previous section, is shared with the LLM.

Configuring the SQL AI Assistant on the Microsoft Azure OpenAI Service

Microsoft Azure allows dedicated deployments of OpenAI GPT models. Using Azure's OpenAI service is much more secure than the publicly hosted OpenAI APIs because the data can be processed in your Virtual Private Cloud (VPC). Due to security considerations, Cloudera recommends that you use GPT models in Hue SQL AI Assistant with Azure's OpenAI service.

Steps

1. Obtain a Microsoft Azure subscription by working with your organization's IT team. Subscriptions vary based on your team and purpose.
2. Register to access the Azure OpenAI service.
   Azure OpenAI requires registration and is currently only available to approved enterprise customers and partners. Customers who wish to use Azure OpenAI are required to submit a registration form.
3. Create an Azure OpenAI resource in the Azure portal.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

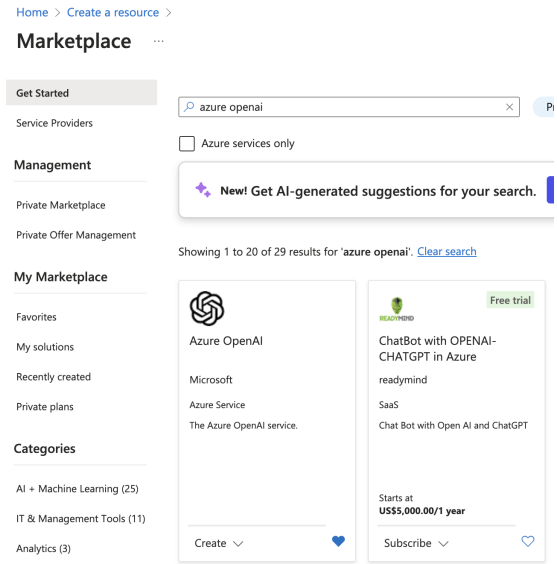

4. Obtain the resource URL and resource keys from the Develop tab under the resource details page.
   You can use any one of the two keys as shown in the following image:

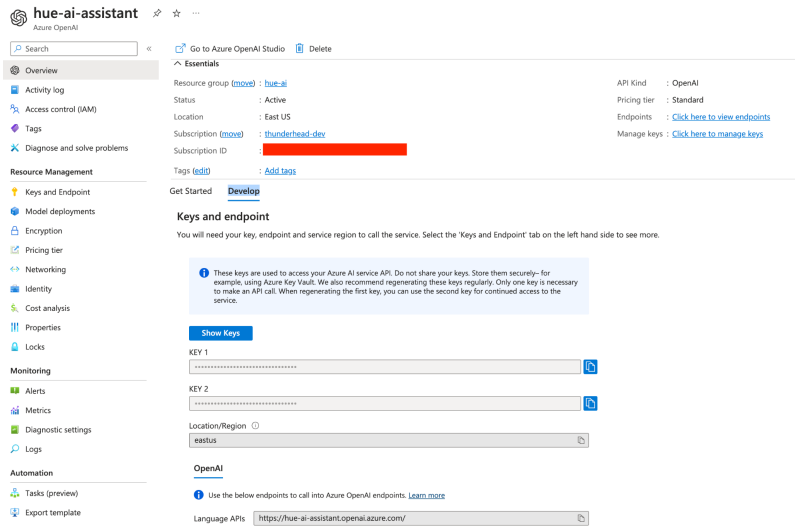

5. Go to Azure OpenAI Studio at https://oai.azure.com/portal and create your deployment under Management > Deployments. Select **gpt-35-turbo-16k** or higher.
6. Enable the Hue SQL AI Assistant in CDW as follows:
   a. Log in to the Data Warehouse service as DWAdmin.
   b. Go to the Virtual Warehouse tab, locate the Virtual Warehouse on which you want to enable this feature, and click **Edit**.
   c. Go to **CONFIGURATIONS** > **Hue**, select **hue-safety-valve** from the **Configuration files** drop-down menu, and add the following lines under the [desktop] section:
      ```
      [desktop]
      ```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
[[ai_interface]]
    service='azure'
    model_name='[***DEPLOYMENT-NAME***]'
base_url="https://[***RESOURCE***].openai.azure.com/"
    token="[***RESOURCE-KEY***]”
```

d. Click **Apply Changes**.

Configuring the SQL AI Assistant on the OpenAI Service

Before you begin

You must have created an account with the OpenAI platform.

Steps

1. Log in to the OpenAi portal.
2. Obtain the API key by navigating to the API keys from the left navigation pane. The key is required to integrate Hue SQL AI Assistant with the OpenAI service.

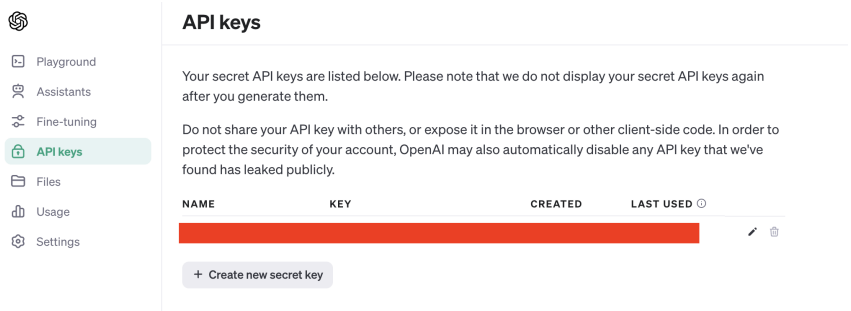

3. Enable the Hue SQL AI Assistant in CDW as follows:
   a. Log in to the Data Warehouse service as DWAdmin.
   b. Go to the Virtual Warehouse tab, locate the Virtual Warehouse on which you want to enable this feature, and click **Edit**.
   c. Go to **CONFIGURATIONS** > **Hue**, select **hue-safety-valve** from the **Configuration files** drop-down, and add the following lines under the [desktop] section:

```
[desktop]
      [[ai_interface]]
            service='openai'
            token='[***API-KEY***]'
```

   d. Click **Apply Changes**.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Configuring the SQL AI Assistant on the Amazon Bedrock Service

Amazon Bedrock is a fully managed service that makes foundation models from leading AI startups and Amazon available through an API.

Before you begin
You must have an AWS account with Bedrock access.

Steps

1. Log in to the Amazon Bedrock service.
2. Obtain your access key and secret as follows:
    a. Go to the IAM console: https://console.aws.amazon.com/iam
    b. Click on **Users** from the left menu and select the user you want to access.
    c. Click on **Security credentials**.
    d. Go to the **Access keys** section and note the access keys.
3. Establish Anthropic Claude access.
   Claude from Anthropic is one of the best models available in Bedrock for SQL-related tasks. By default, Claude is not available on Bedrock. You need to place a special request for Claude. Once you have access, you can try Claude in the text playground under the Amazon Bedrock service. If you are in the us-east-1 region, this must take you to https://us-east-1.console.aws.amazon.com/bedrock/home?region=us-east-1#/text-playground.
4. Enable the Hue SQL AI Assistant in CDW as follows:
    a. Log in to the Data Warehouse service as DWAdmin.
    b. Go to the Virtual Warehouse tab, locate the Virtual Warehouse on which you want to enable this feature, and click **Edit**.
    c. Go to **CONFIGURATIONS** > **Hue**, select **hue-safety-valve** from the **Configuration files** drop-down menu, and add the following lines:

```
[aws]
      [[bedrock_account]]
            access_key_id='[***ACCESS-KEY***]'
            secret_access_key='[***SECRET-KEY***]'
            region='us-east-1'
[desktop]
      [[ai_interface]]
            service='bedrock'
            model='claude'
```

5. Click **Apply Changes**.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Service and model-related configurations for setting up the SQL AI Assistant in CDW

You can configure the AI services and models you want to use by going to **CONFIGURATIONS** > **Hue** > **hue-safety-valve** and adding the following lines:

```
[desktop]
      [[ai_interface]]
            [***CONFIG-KEY1***]='[***VALUE***]'
            [***CONFIG-KEY2***]='[***VALUE***]'
      [[semantic_search]]
            [***CONFIG-KEY1***]='[***VALUE***]'
            [***CONFIG-KEY2***]='[***VALUE***]'
```

Specify the service and model-related configurations under the [[ai_interface]] section as listed in the following table:

AI interface config key	Description
service	API service to be used for AI tasks. AI is disabled when a service is not configured. For example, `azure`, `openai`, `bedrock`, and `ai_assistant`.
trusted_service	Indicates whether the LLM is trusted or not. Turn on to disable the warning. The default value is "`True`".
model	The AI model you want to use for AI tasks. For example, `gpt`, `llama`.
model_name	The fully qualified name of the model to be used. For example, `gpt-3.5-turbo-16k`.
base_url	Service API base URL.
add_table_data	When enabled, sample rows from the table are added to the prompt. The default value is "`True`".
table_data_cache_size	Size of the LRU cache used for storing table sample data.
auto_fetch_table_meta_limit	Number of tables to load from a database, initially.
token	Service API secret token.
token_script	Provides a secure way to get the service API secret token.

Specify the semantic search-related configurations used for RAG under the [[semantic_search]] section, as listed in the following table:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Semantic search config key	Description
relevancy	The technology you want to use for semantic search. Acceptable values are `vector_search` or `vector_db`.
embedding_model	The model you want to use for data-embedding. This must be compatible with SentenceTransformer.
cache_size	Size of the LRU cache used for storing embedding.

How to use the SQL AI Assistant?

To launch the Hue AI assistant, click the blue dot on the Hue web interface interface as shown in the following image:

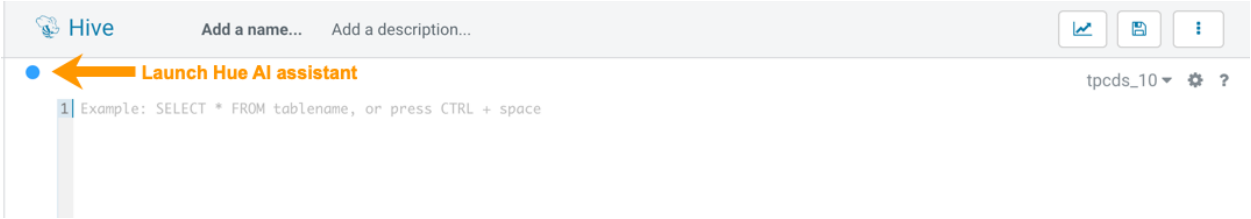

Generating SQL from NQL

Click **GENERATE** and type the query in natural language as shown in the following image:

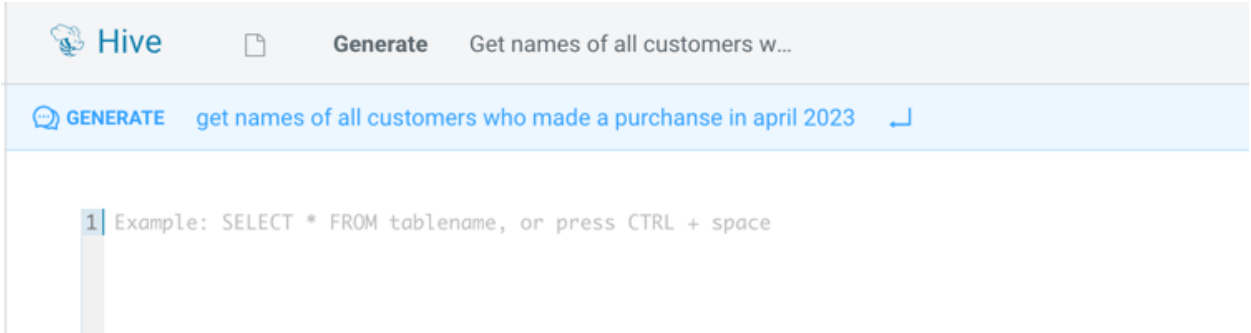

The SQL query is generated as shown in the following image. You can insert it into the editor by clicking **Insert** or you can copy the query into the editor.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Editing the query in natural language

Click **Edit** as shown in the following image:

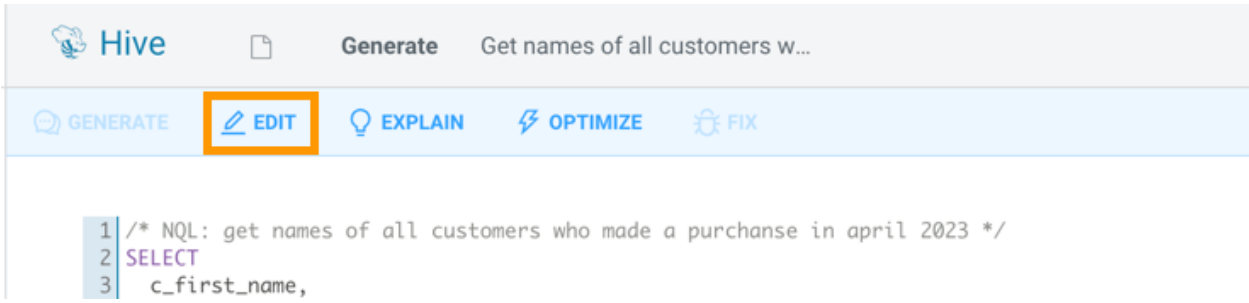

Make the required changes and press enter:

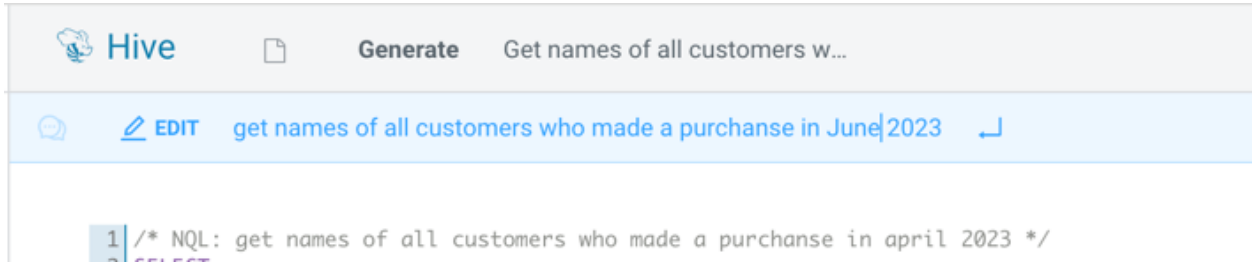

Getting an explanation of a SQL query in natural language

To understand complex queries in natural language, click **EXPLAIN** as shown in the following image:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

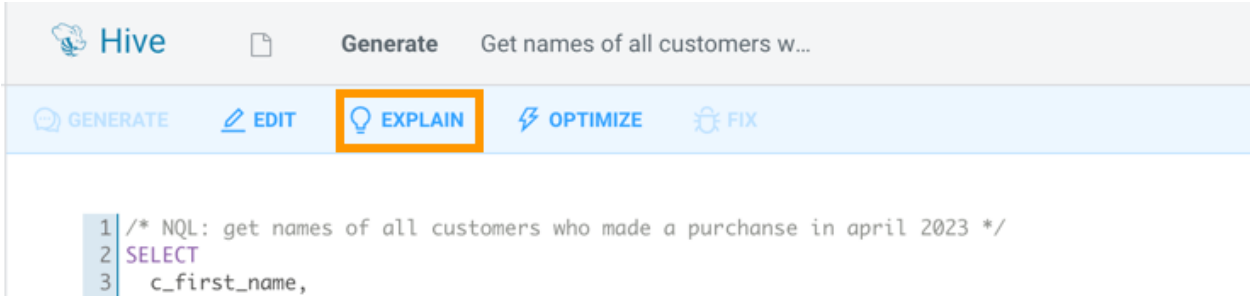

The LLM generates the explanation of the SQL query as shown in the following image:

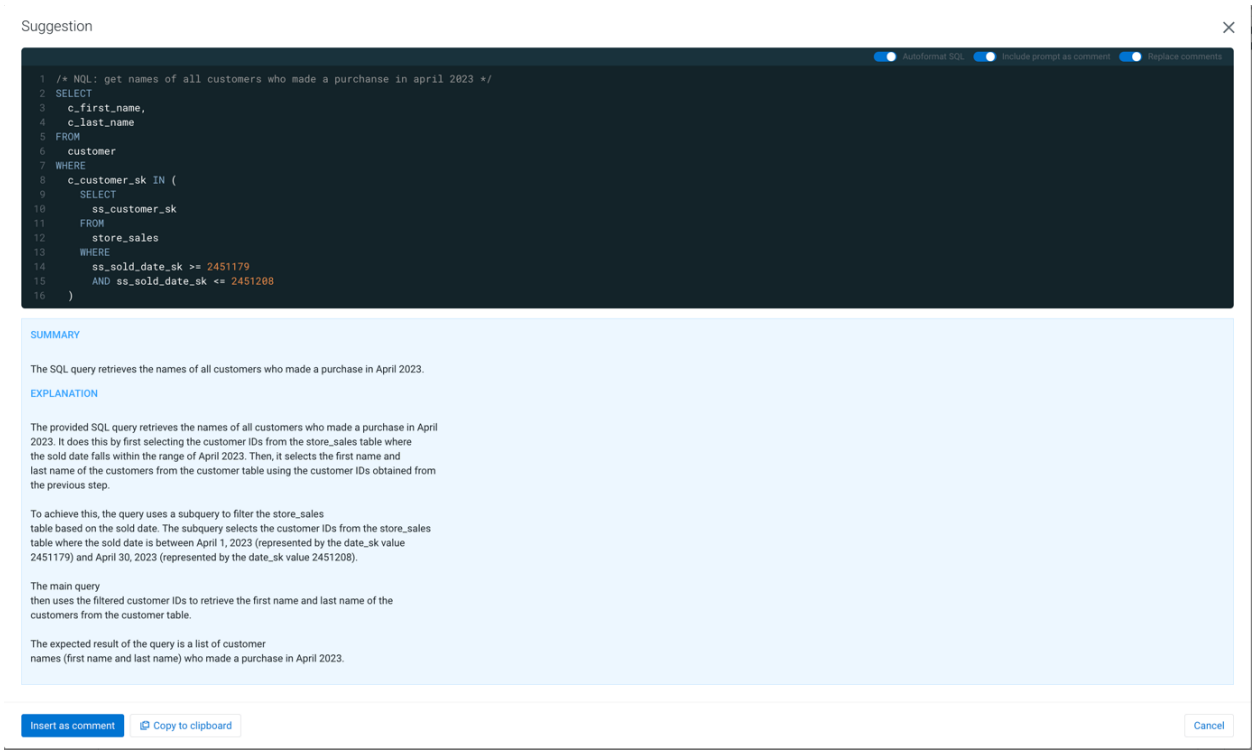

Optimizing a query

Click **OPTIMIZE** to improve an existing query as shown in the following image:

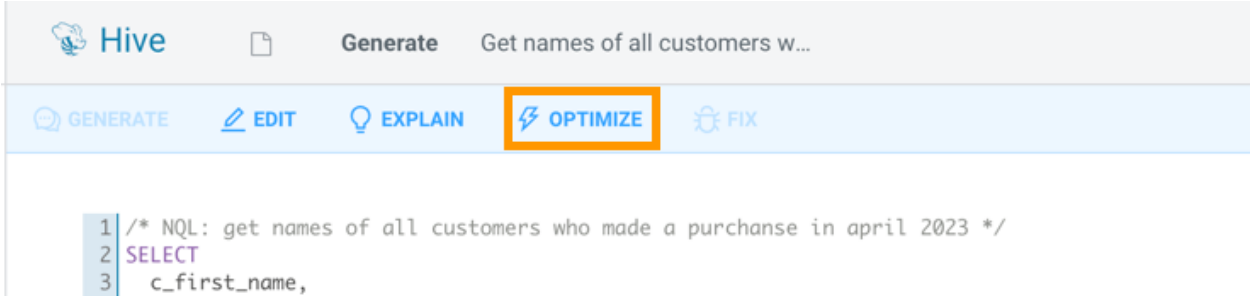

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

The following example shows how Hue identifies issues, optimizes a query, and provides an explanation:

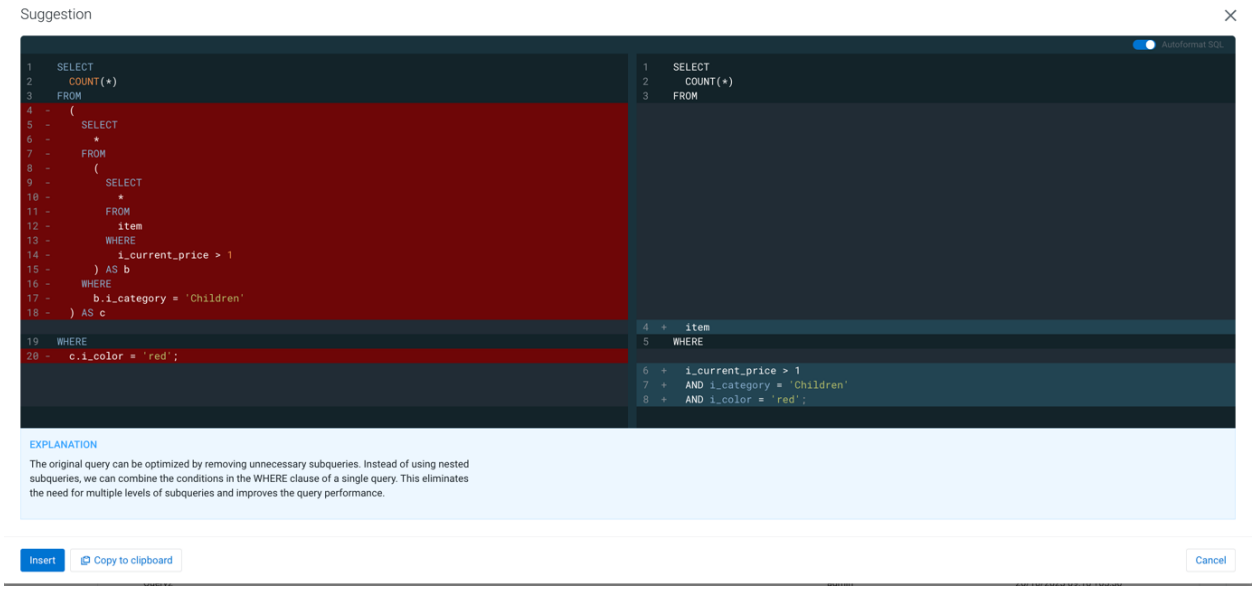

Fixing a SQL query

Click **FIX** to correct any syntax errors present in your SQL statement causing the query to fail as shown in the following image:

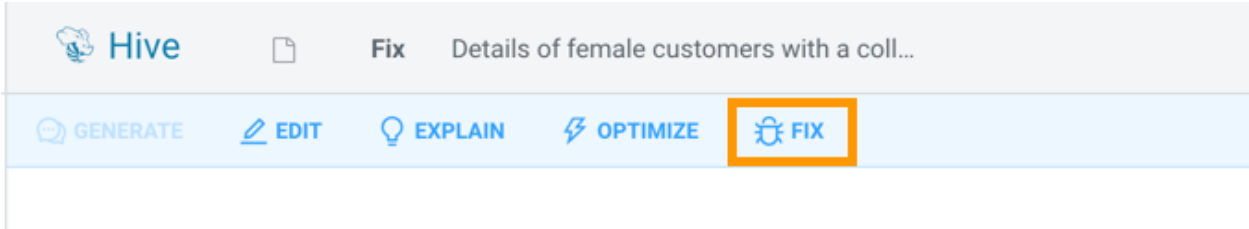

For example:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

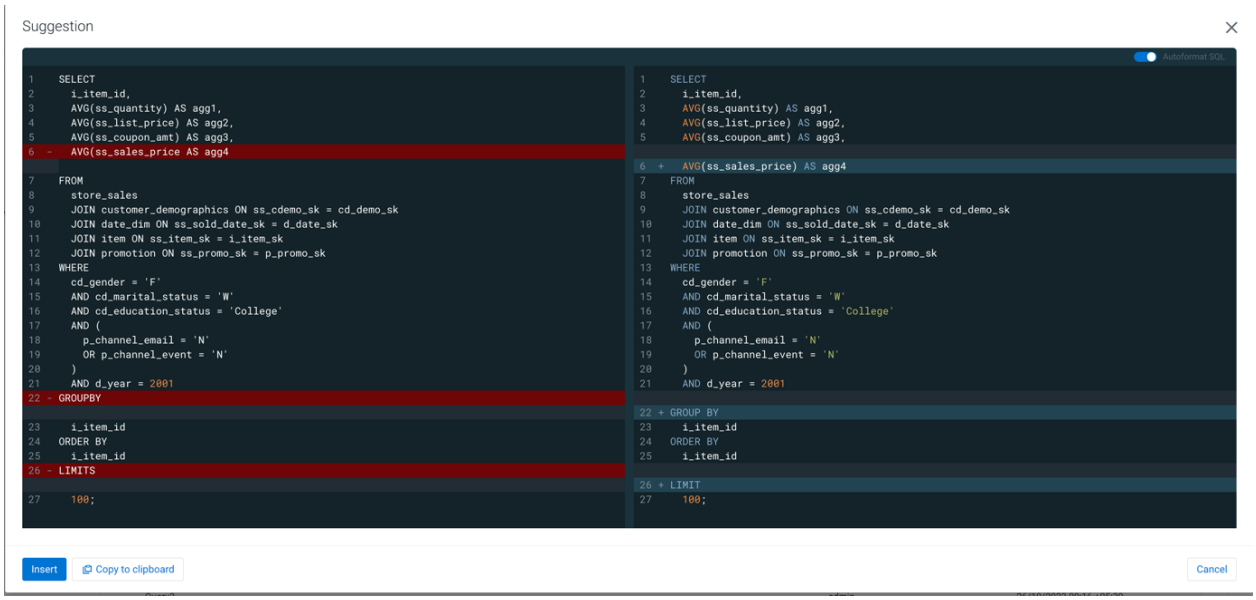

Limitations

- **Non-deterministic**: LLMs are non-deterministic, which means you cannot guarantee the same output for the same input every time, and it can lead to different responses to **similar** queries.
- **Ambiguity**: LLMs may struggle to handle ambiguous queries or contexts. SQL queries often rely on specific and unambiguous language, but LLMs can misinterpret or generate ambiguous SQL queries, leading to incorrect results.
- **Hallucination:** In the context of LLMs, hallucination refers to a phenomenon where these models generate text or responses that are incorrect, nonsensical, or fabricated. Occasionally you might see incorrect identifiers or literals in the response.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Using Impala AI function in Data Warehouse Public Cloud (Preview)

Date published: 2024-07-26
Date modified: 2024-07-26

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
Advantages and use cases of Impala AI functions **3**
Use LLMs directly in SQL with Impala's ai_generate_text function 4
Advantages of using AI functions 4
List of possible use cases 5
Overview of AI function syntax and arguments **5**
Syntax for AI built-in function arguments 5
Key parameters for using the AI model 6
Examples of AI functions and UDFs **6**
Examples of using the built-in AI function 6
Examples of creating and using custom UDFs along with the built-in AI function 7
Configuring the JCEKS keystore for Impala AI integration **8**

Advantages and use cases of Impala AI functions

Cloudera Data Warehouse allows you to use Impala's ai_generate_text function to access Large Language Models (LLMs) in SQL queries. This function enables you to input a prompt, retrieve the LLM response, and include it in results. You can create custom UDFs for complex tasks like sentiment analysis and translation.

Note: *This feature is in technical preview and not recommended for production deployments. Cloudera recommends that you try this feature in test and development environments.*

Use LLMs directly in SQL with Impala's ai_generate_text function

Impala introduces a built-in AI function called `ai_generate_text` that enables direct access to and utilization of Large Language Models (LLMs) in SQL queries. With this function, you can input a prompt, which may include data. The function communicates with a supported LLM endpoint, sends the prompt, retrieves the response, and includes it in the query result.

Alternatively, seamlessly integrate LLM intelligence into your Impala workflow by creating custom User Defined Functions (UDFs) on top of `ai_generate_text`. This allows you to use concise SQL statements for sending prompts to an LLM and receiving responses. You can define UDFs for complex tasks like sentiment analysis, language translation, and generative contextual analysis.

For examples of AI built-in function arguments and how to create and use custom UDFs with the built-in AI function, see *Overview of AI function syntax and arguments.*

Advantages of using AI functions

- Simplified Workflow: Eliminates the necessity for setting up intricate data pipelines.
- No ML Expertise Required: No specialized machine learning skills are needed.

- Swift Decision-Making: Enables faster insights on the data, facilitating critical business decisions by using in-database function calls.
- Integrated Functionality: Requires no external applications, as it is a built-in feature in Data Warehouse.

List of possible use cases

Here are some practical applications of using AI models with the function:

- Sentiment Analysis: Use the AI model to examine customer reviews for a product and identify their sentiment as positive, negative, or neutral.
- Language Translation: Translate product reviews written in different languages to understand customer feedback from various regions.
- Generative Contextual Analysis: Generate detailed reports and insights on various topics based on provided data.

Overview of AI function syntax and arguments

Learn how you can use the ai_generate_text functions to connect to OpenAI or Azure OpenAI endpoints and integrate AI models into your Impala SQL queries.

Syntax for AI built-in function arguments

The following example of a built-in AI function demonstrates the use of the OpenAI API as a large language model. Currently, OpenAI's public endpoint and Azure OpenAI endpoints are supported.

AI_GENERATE_TEXT_DEFAULT

Syntax

```
ai_generate_text_default(prompt)
```

AI_GENERATE_TEXT

Syntax

```
ai_generate_text(ai_endpoint, prompt, ai_model,
ai_api_key_jceks_secret, additional_params)
```

The `ai_generate_text` function uses the values you provide as an argument in the function for `ai_ endpoint`, `ai_model`, and `ai_api_key_jceks_secret`. If any of them are left empty or NULL, the function uses the default values set at the instance level instead.

When using the `ai_generate_text_default` function, make sure to set all parameters (`ai_ endpoint`, `ai_model`, and `ai_api_key_jceks_secret`) in the coordinator/executor flagfiles with appropriate values.

Key parameters for using the AI model

- ai_endpoint: The endpoint for the model API that is being interfaced with, supports services like OpenAI and Azure OpenAI Service, for example, https://api.openai.com/v1/chat/completions.
- prompt: The text you submit to the AI model to generate a response.
- ai_model: The specific model name you want to use within the desired API, for example, gpt-3.5-turbo.
- ai_api_key_jceks_secret: The key name for the JCEKS secret that contains your API key for the AI API you are using. You need a JCEKS keystore containing the specified JCEKS secret referenced in ai_api_key_jceks_secret. To configure the keystore file location, see *Configuring the JCEKS keystore for Impala AI integration*.
- additional_params: Additional parameters that the AI API offers that is provided to the built-in function as a JSON object.

Examples of AI functions and UDFs

Learn to use the ai_generate_text_default function for tasks like analyzing Amazon book reviews. Also, see how to create custom UDFs to make AI text analysis easier and more efficient, enhancing your data workflows and insights.

Examples of using the built-in AI function

The following example lists the steps needed to turn a prompt into a custom SQL function using just the built-in function `ai_generate_text_default`.

```
> select ai_generate_text_default('hello');

Response:

Hello! How can I assist you today?
```

In the below example, a query is sent to the Amazon book reviews database for the book titled Artificial Superintelligence. The large language model (LLM) is prompted to classify the sentiment as positive, neutral, or negative.

```
> select customer_id, star_rating,
ai_generate_text_default(CONCAT('Classify the following review as
positive, neutral, or negative', and only include the uncapitalized
category in the response: ', review_body)) AS review_analysis,
review_body from amazon_book_reviews where product_title='Artificial
Superintelligence' order by customer_id LIMIT 1;
```

```
Response:

+--+------------+------------+----------------+------------------+

|  |customer_id |star_rating |review_analysis |review_body       |

+--+------------+------------+----------------+------------------+

|1 |4343565     | 5          |positive        |What is this book |

|  |            |            |                |all about ……      |

+--+------------+------------+----------------+------------------+
```

Examples of creating and using custom UDFs along with the built-in AI function

Instead of writing the prompts in a SQL query, you can build an UDF with your intended prompt. Once you build your custom UDF with the prompt, you can pass that prompt into the ai_generate_text_default built-in Impala function.

Example: Classify input customer reviews

The following UDF uses the Amazon book reviews database as the input and requests the LLM to classify the sentiment.

Classify input customer reviews:

```
IMPALA_UDF_EXPORT
```

```
StringVal ClassifyReviewsDefault(FunctionContext* context, const
StringVal& input) {

 std::string request =

     std::string("Classify the following review as positive, neutral,
or negative")

     + std::string(" and only include the uncapitalized category in
the response: ")

     + std::string(reinterpret_cast<const char*>(input.ptr),
input.len);

 StringVal prompt(request.c_str());

 return context->Functions()->ai_generate_text_default(context,
prompt);

}
```

Now you can define these prompt building UDF and build them in Impala. Once you have them running, you can query your data sets using them.

Creating `analyze_reviews` function:

```
> CREATE FUNCTION analyze_reviews(STRING)

RETURNS STRING

LOCATION 's3a://dw-...............'

SYMBOL='ClassifyReviews'
```

Using SELECT query for Sentiment analysis to classify Amazon book reviews

```
> SELECT customer_id, star_rating, analyze_reviews(review_body) AS
review_analysis, review_body from amazon_book_reviews where
product_title='Artificial Superintelligence' order by customer_id;
```

Query History Saved Queries Results (6)

	customer_id	star_rating	review_analysis	review_body
1	44254093	5	positive	What is this book all about? It is all about a mind-blowing universal law of nature. Mind-blowing, because it is unbelievable that at the end of the 20th c
2	50050072	5	positive	The two tightly-connected ideas strike you as amazed. In the first place, what has never before been clearly and stricltly formulated, the fundamental tr
3	50050072	5	positive	The two tightly-connected ideas strike you as amazed. In the first place, what has never before been clearly and stricltly formulated, the fundamental tr
4	52932308	1	negative	This book is seriously flawed. I could not work out if the author was a mathemetician dabbling (very poorly) with philosophy, or a muddled philosopher
5	52971961	1	negative	Abdoullaev's exploration of AI issues appears to be very technological and straightforward for such a complex matter. Author did not discuss any phil
6	53008416	4	positive	As Co Founder of ArtilectWorld:ultra intelligent machines, I recommend reading this book!

Configuring the JCEKS keystore for Impala AI integration

Learn to configure the JCEKS keystore for AI functions in Impala, including setting the keystore file location and updating Cloudera Data Warehouse settings. This ensures secure storage and management of API keys, facilitating a secure and efficient AI integration within data workflows.

1. Log in to the Cloudera Data Warehouse service as an administrator.
2. Go to **Impala Virtual Warehouse→→ Edit→Configurations→Impala coordinator** and select `hadoop-core-site-default-warehouse` from the **Configuration files** drop-down list.
3. Set the value of the `hadoop.security.credential.provider.path` property to **`[***FILE PATH***]`**.localjceks.
4. Go to the **Impala executor** tab and select `hadoop-core-site-default-warehouse` from the **Configuration files** drop-down list.
5. Set the value of the `hadoop.security.credential.provider.path` property to **`[***FILE PATH***]`**.localjceks.
6. Restart the Impala Virtual Warehouse.

Using Impala to query external JDBC data sources (Preview)

Date published: 2024-07-26
Date modified: 2024-07-26

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
Impala support for external JDBC tables **4**
Prerequisites 4
Syntax 4
Table Properties 6
Supported Data Types 6
Limitations 7
Securing the JDBC password 7
Support for case-sensitive table and column names 8
Modifying the external JDBC table **8**
Querying external JDBC tables **8**
Query options for external JDBC tables **9**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Impala support for external JDBC tables

Apache Impala now supports reading from external JDBC data sources. An external JDBC table represents a table or a view in a remote RDBMS database or another Impala cluster. Using external JDBC tables, you can connect Impala to a database, such as MySQL, PostgreSQL, or another Impala cluster and read the data in the remote tables.

***Note**: This feature is in technical preview and not recommended for use in production deployments. Cloudera recommends that you try this feature in test and development environments.*

Prerequisites

The Impala package does not include JDBC drivers of the remote databases. Ensure that the MySQL, Postgres, and Impala JDBC drivers are uploaded to a location in HDFS, Ozone, or Amazon S3 that can be read by Impala.

Syntax

To connect to a remote database, you create an external JDBC table with the appropriate table properties, such as the database type, JDBC URL, driver class, driver file location, JDBC username and password, and name of the remote table to be mapped to the Impala external table.

Syntax:

```
CREATE EXTERNAL TABLE [IF NOT EXISTS] [db_name.]table_name
  (col_name data_type,
   ....)
  STORED BY JDBC
  TBLPROPERTIES (
    "database.type"="[***VALUE***]",
    "jdbc.url"="[***VALUE***]",
    "jdbc.driver"="[***VALUE***]",
    "driver.url"="[***VALUE***]",
    "dbcp.username"="[***VALUE***]",
    "dbcp.password"="[***VALUE***]",

    "table"="[***VALUE***]");
```

Examples:

- Creating an external JDBC table to map a table in a remote PostgreSQL database:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
CREATE EXTERNAL TABLE student_jdbc (
      id INT,
      bool_col BOOLEAN,
      tinyint_col TINYINT,
      smallint_col SMALLINT,
      int_col INT,
      bigint_col BIGINT,
      float_col FLOAT,
      double_col DOUBLE,
      date_col DATE,
      string_col STRING,
      timestamp_col TIMESTAMP)
STORED BY JDBC
TBLPROPERTIES (
      "database.type"="POSTGRES",
      "jdbc.url"="jdbc:postgresql://[***IP_ADDRESS***]:[***PORT***]
   /[***DATABASE NAME***]",
      "jdbc.driver"="org.postgresql.Driver",
      "driver.url"="/test-warehouse/data-sources/jdbc-drivers/postg
   resql-jdbc.jar",
"dbcp.username"="[***USERNAME***]",
"dbcp.password"="[***PASSWORD***]",
"table"="student");
```

- Creating an external JDBC table to map a table in another Impala cluster:

```
CREATE EXTERNAL TABLE student_jdbc (
  id INT,
  bool_col BOOLEAN,
  tinyint_col TINYINT,
  smallint_col SMALLINT,
  int_col INT,
  bigint_col BIGINT,
  float_col FLOAT,
  double_col DOUBLE,
  date_col DATE,
  string_col STRING,
  timestamp_col TIMESTAMP)
STORED BY JDBC
TBLPROPERTIES (
  "database.type"="IMPALA",
  "jdbc.url"="jdbc:impala://[***IP
ADDRESS***]:[***PORT***]/[***DATABASE NAME***]",
  "jdbc.auth"="AuthMech=3",
  "jdbc.properties"="MEM_LIMIT=1000000000, MAX_ERRORS = 10000",
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
  "jdbc.driver"="com.cloudera.impala.jdbc.Driver",
"driver.url"="hdfs://test-warehouse/data-sources/jdbc-drivers/I
mpalaJDBC42.jar",
  "dbcp.username"="[***USERNAME***]",
"dbcp.password.keystore"="jceks://hdfs/test-warehouse/data-sour
ces/test.jceks",
  "dbcp.password.key"="[***KEY***]",
  "table"="student");
```

Table Properties

While creating an external JDBC table, you are required to specify the following table properties:

- `database.type`: POSTGRES, MYSQL, IMPALA
- `jdbc.url`: JDBC connection string with the required parameters — database type, hostname, port number, and database name.

  Example: “jdbc:impala://10.96.132.138:21050/sample_db”.

- `jdbc.driver`: Class name of the JDBC driver
- `driver.url`: URL to download the JAR file package that is used to access the external database
- table: Name of the table in the remote database that you want to map in Impala

Besides the above required properties, you can also specify optional parameters that allow you to use different authentication methods, allow case sensitive column names in remote tables, or to specify additional database properties:

- `jdbc.auth`: Authentication mechanism of the JDBC driver
- `dbcp.username`: JDBC username
- `dbcp.password`: JDBC password in clear text.

  Note: Storing JDBC passwords in clear text is not recommended in production environments. The recommended way is to store the password in a Java keystore file.

- `dbcp.password.key`: Key of the Java keystore
- `dbcp.password.keystore`: Location of the keystore file
- `jdbc.properties`: Additional properties applied to database engines, like Impala Query options. The properties are specified as comma-separated "key-value" pairs.
- `jdbc.fetch.size`: Number of rows to fetch in a batch
- `column.mapping`: Mapping of column names between external table and Impala JDBC table.

Supported Data Types

The following column data types are supported for an external JDBC table:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- Numeric data type: boolean, tinyint, smallint, int, bigint, float, double
- Decimal with scale and precision
- String type: string
- Date
- Timestamp

Note: The following data types are currently not supported — char, varchar, and binary. Complex data type: struct, map, array and nested type.

Limitations

- Unsupported column data types: char, varchar, binary. Complex data type - struct, map, array, and nested type
- JDBC tables have to be defined one table at a time
- Writing to a JDBC table is not supported
- Only support binary predicates with operators =, !=, <=, >=, <, > to be pushed to RDBMS

Securing the JDBC password

The `dbcp.password` table property stores the JDBC password in clear text. To avoid the risk of a password leak, the `SHOW CREATE TABLE <table-name>` and `DESCRIBE FORMATTED | EXTENDED <table-name>` statements mask the value of the `dbcp.password` table property in their outputs.

In production environments, it is recommended that you do not store the JDBC password in clear text using the `dbcp.password` table property. Instead, you can store the password in a Java Keystore file on HDFS or on cloud storage like Amazon S3 using the following command:

- Creating a Java keystore file on HDFS with the key as "host1.password" and password as "passwd1":

  ```
  hadoop credential create host1.password -provider
  jceks://hdfs/user/foo/test.jceks -v passwd1
  ```

- Creating a Java keystore file on Amazon S3 with the key as "impala" and password as "passwd2":

  ```
  hadoop credential create impala -provider
  jceks://s3a@dw-impala-test/jceks/demo.jceks -v passwd2
  ```

For more information, see the [Apache Hadoop CredentialProvider API] guide.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Support for case-sensitive table and column names

The column names of tables in the remote database can be different from the external JDBC table schema. For example, Postgres allows case-sensitive column names, however, Impala saves column names in lowercase. In such situations, you can set the `column.mapping` table property to map column names between Impala external JDBC tables and the remote tables.

Example:

```
"column.mapping"="id=id, bool_col=Bool_col, tinyint_col=Tinyint_col,
smallint_col=Smallint_col, int_col=Int_col, bigint_col=Bigint_col,
float_col=Float_col, double_col=Double_col, date_col=date_col,
string_col=String_col, timestamp=Timestamp");
```

Modifying the external JDBC table

You can use the `ALTER TABLE` statement to add, drop, or modify columns, or modify the table properties of existing external JDBC tables. The syntax is the same as the other Impala tables.

Example:

- Using ALTER TABLE statement to add, drop, or modify columns:

  ```
  ALTER TABLE student_jdbc ADD COLUMN IF NOT EXISTS date_col DATE;
  ALTER TABLE student_jdbc DROP COLUMN int_col;
  ALTER TABLE student_jdbc CHANGE COLUMN date_col timestamp_col
  TIMESTAMP;
  ```

- Using ALTER TABLE statement to modify table properties:

  ```
  ALTER TABLE student_jdbc
  SET TBLPROPERTIES ("dbcp.username"="impala",
  "dbcp.password"="password");
  ```

Querying external JDBC tables

Querying or reading external JDBC tables is the same as querying regular tables in Impala. You can use SELECT statements to query data and can also join the external table with other tables across databases. However, do note that the metadata for the external tables is not persisted in Hive Metastore (HMS).

Example:

```
SELECT * from student_jdbc;
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Query options for external JDBC tables

A new query option, **CLEAN_DBCP_DS_CACHE** is added to save the DBCP SQL DataSource objects in the cache for a longer period of time. This allows the DBCP connection pools to be reused across multiple queries. When the value is set to false, the DBCP SQL DataSource object is not closed when its reference count is 0. The SQL DataSource object is kept in cache until the object is idle for more than 5 minutes.

Type: BOOLEAN
Default: True (1)

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Impala workload management in Data Warehouse Public Cloud (Preview)

Date published: 2024-07-26
Date modified: 2024-07-30

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
About Impala workload management **4**
Table format **4**
Use cases **9**
How Impala workload management works **10**
Impala coordinator startup flags 10
Known Issue **12**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

About Impala workload management

Cloudera Data Warehouse (CDW) Public Cloud provides you the option to enable logging Impala queries on an existing Virtual Warehouse or while creating a new Impala Virtual Warehouse. By logging the Impala queries in CDW, you gain increased observability of the workloads running on Impala, which you can use to improve the performance of your Impala Virtual Warehouses.

Note: *This feature is in technical preview and not recommended for use in production deployments. Cloudera recommends that you try this feature in test and development environments.*

This feature significantly enhances the query profiling capabilities. You can have Impala archive crucial data from each query's profile into dedicated database tables, namely the query history table and live query table. These tables are part of the `sys` database and are designed to store valuable information that can later be queried using any Impala client, providing a consolidated view of reports from previously executed queries.

The query history table, `sys.impala_query_log`, proves particularly useful when dissecting workloads for in-depth analysis of query performance. Unlike the limitations associated with query profiles, which are only available to the client that initiated the query, the query history table offers a comprehensive solution for querying completed queries without the need to parse the text of each query profile. Additionally, the query history table provides a comprehensive view across all Impala coordinators.

The Impala query information is stored indefinitely in the `sys.impala_query_log` table whereas the `sys.impala_query_live` table reflects the in-memory state of all Impala coordinators. Actively running and recently completed queries are stored in the `sys.impala_query_live` table. Data is removed from this table once it is persisted in the `sys.impala_query_log` table or if the coordinator is restarted. Therefore, it is possible that some of the records are momentarily duplicated in both these tables.

Because the `sys.impala_query_live` table is stored only in memory, recently completed queries that still need to be persisted to the `sys.impala_query_log` table are lost if the coordinator crashes. However, if the coordinator shuts down gracefully, then the recently completed queries are stored in the `sys.impala_query_log` table and are not lost.

Table format

The Impala query history and live query system tables contain the following columns:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Column Name	Description	Data Type	Sample Value
cluster_id	String specified through the Impala startup flag to uniquely identify an instance	string	cluster-123
query_id	Impala assigned query identifier	string	214d08bef0831e7a:3c65392400000000
session_id	Impala assigned session identifier	string	ea4f661af43993d8:587839553a41adb8
session_type	Client session type	string	HIVESERVER2
hiveserver2_protocol_version	Version of the HiveServer (HS2) protocol that was used by the client when connecting	string	HIVE_CLI_SERVICE_PROTOCOL_V6
db_user	Effective user on the cluster	string	csso_name
db_user_connection	Username from an authenticated client	string	csso_name
db_name	Name of the database being queried	string	default
impala_coordinator	Name of the coordinator for the query	string	coord-22899:27000
query_status	Status of the query when it completes	string	OK
query_state	Final state of the query	string	FINISHED
impala_query_end_state	Final Impala state of the query	string	FINISHED
query_type	Type of the query	string	QUERY
network_address	Client IP address and port	string	127.0.0.1:40120

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

start_time_utc	Time when the query started. Time zone is in UTC	timestamp	2024-07-17 17:13:46.414316000
total_time_ms	Difference between the query end time and start time, in milliseconds	decimal(18,3)	136.121
query_opts_config	List of query options stored as a single string containing comma-separated values of key-value pairs	string	TIMEZONE=America/Los_Angeles,CLIENT_IDENTIFIER=Impala Shell v4.4.0a1 (04bdb4d) built on Mon Nov 20 10:49:35 PST 2023
resource_pool	Name of the resource pool for the query	string	default-pool
per_host_mem_estimate	Size, in bytes of the per-host memory estimate	bigint	5
dedicated_coord_mem_estimate	Size, in bytes of the dedicated coordinator memory estimate.	bigint	4
per_host_fragment_instances	Comma-separated string listing each host and its fragment instances	string	myhost-1:27000=1,myhost-2:27001=2
backends_count	Count of the number of backends used by this query	integer	2
admission_result	Result of the admission (not applicable to DDLs)	string	Admitted immediately
cluster_memory_admitted	Cluster memory, in bytes that was admitted	integer	4
executor_group	Name of the executor group	string	executor_group

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

executor_groups	List of all executor groups including the groups that were considered and rejected as part of Workload Aware Auto Scaling	string	executor_group1, executor_group2…
exec_summary	Full text of the executor summary	string	
num_rows_fetched	Number of rows fetched by the query	bigint	6001215
row_materialization_rows_per_sec	Count of the number of rows materialized per second	bigint	3780
row_materialization_time_ms	Time spent materializing rows converted to milliseconds	decimal(18,3)	1.58
compressed_bytes_spilled	Count of bytes that were written (or spilled) to scratch disk space	bigint	241515
event_planning_finished	Event from the timeline. The value represents the number of milliseconds since the query was received.	decimal(18,3)	27.253
event_submit_for_admission	Event from the timeline. The value represents the number of milliseconds since the query was received.	decimal(18,3)	30.204
event_completed_admission	Event from the timeline. The value represents the number of	decimal(18,3)	30.986

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

	milliseconds since the query was received.		
event_all_backends_started	Event from the timeline. The value represents the number of milliseconds since the query was received.	decimal(18,3)	31.969
event_rows_available	Event from the timeline. The value represents the number of milliseconds since the query was received.	decimal(18,3)	31.969
event_first_row_fetched	Event from the timeline. The value represents the number of milliseconds since the query was received.	decimal(18,3)	135.175
event_last_row_fetched	Event from the timeline. The value represents the number of milliseconds since the query was received.	decimal(18,3)	135.181
event_unregister_query	Event from the timeline. The value represents the number of milliseconds since the query was received.	decimal(18,3)	141.435
read_io_wait_total_ms	Total read I/O wait time converted to milliseconds	bigint	15.091
read_io_wait_mean_	Average read I/O wait	bigint	35.515

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

ms	time across executors converted to milliseconds		
bytes_read_cache_total	Total bytes read from the data cache	bigint	45823
bytes_read_total	Total bytes read	bigint	745227
pernode_peak_mem_min	Minimum value of all the per-node peak memory usages	bigint	5552846
pernode_peak_mem_max	Maximum value of all the per-node peak memory usages	bigint	5552846
pernode_peak_mem_mean	Mean value of all the per-node peak memory usages	bigint	5552846
sql	SQL statement as provided by the user	string	SELECT db_user, total_time_ms from impala_query_log where query_state = 'EXCEPTION';
plan	Full text of the query plan	string	
tables_queried	Comma-separated string containing all the tables queried in the SQL statement	string	

Use cases

A consolidated view of reports from previously executed queries can be useful in the following use cases:

- For collecting the history of all queries run and reported by a user or by date and time
  Example: `SELECT db_user, start_time, end_time, total_time_ms, sql from impala_query_log order by db_user;`
- For collecting the frequently-run queries
  Example: `SELECT db_user, query_status, query_state, sql from impala_query_log where query_state = 'EXCEPTION';`
- For reporting queries that are running over 10 minutes

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Example: `SELECT db_user, total_time_ms, sql from impala_query_log where total_time_ms > 600000;`

How Impala workload management works

To use this feature, enable Impala query logging while creating a new Virtual Warehouse or by editing an existing one by selecting the **Log Impala Queries (Preview)** option. By default, the **Log Impala Queries (Preview)** option is off.

You can then configure the Impala coordinator using specific startup flags to store query history. Impala manages the table serving as a centralized repository for all query histories across databases. Completed queries are periodically inserted into this table based on a preconfigured interval.

This feature streamlines the process of query history management, providing a more accessible and comprehensive way to analyze and retrieve information about completed queries

Note: This feature is available only on CDW running Impala image versions 2024.0.18.0 and higher.

Important: The following query types are not written into the query logging tables:

- SET
- SHOW
- USE
- DESCRIBE

Impala coordinator startup flags

Each Impala coordinator runs an SQL statement on startup to create the query logging tables. The following table lists the Impala startup coordinator flags that you can configure:

Name	Data Type	Default	Description
cluster_id	string		Specifies an identifier string that uniquely represents this cluster. This identifier is included in both tables and is used as a table partition for the `sys.impala_query_log` table.
query_log_shutdown_deadline_s	number (seconds)	30	Hidden flag. Number of seconds to wait for the queue of completed queries to be carried into the query history table before timing out and continuing the shutdown

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

			process. The query history table drain process runs after the shutdown process completes, therefore the max shutdown time is extended by the value specified in this flag.
workload_mgmt_user	string	impala	Specifies the user that will be used to insert records into the query history table.
query_log_write_interval_s	number (seconds)	300	Number of seconds to wait before inserting completed queries into the query history table. Allows for batching inserts to help avoid small files.
query_log_max_queued	number	5000	Maximum number of completed queries that can be queued before they are written to the query history table. This flag operates independently of the 'query_log_write_interval_m' flag. If the number of queued records reaches this value, the records will be written to the query log table no matter how much time has passed since the last write. A value of 0 indicates no maximum number of queued records.
query_log_max_sql_length	number	16777216	Maximum length of an SQL statement that will be recorded in the completed queries table. If an SQL statement with a length longer than this specified value is executed, the SQL inserted into the completed queries table will be trimmed to this length. Any characters that require escaping will have their backslash character counted towards this limit.
query_log_max_plan_length	number	16777216	Maximum length of the SQL query plan that will be recorded in the completed queries table. If a query plan has a length longer than this value, the plan inserted into the completed queries table will be trimmed to this length.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

			Any characters that require escaping will have their backslash character counted towards this limit.
query_log_request_pool	string		Specifies a pool or queue used by the queries that insert into the query log table. Empty value causes no pool to be set.

Known Issue

DWX-18554: You cannot access the `sys.impala_query_live` table when Impala cannot schedule the query.

Workaround: None.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Enabling Secure Boot Option for GCP (Preview)

Date published: 2025-01-30
Date modified: 2025-01-30

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

GCP's Secure Boot feature helps ensure that the system only runs authentic software by verifying the digital signature of all boot components, and halting the boot process if signature verification fails. For more information, read the dedicated GCP documentation.

Note: You need to contact Cloudera Support to have this feature enabled. The CDP_GCP_CB_SECURE_BOOT entitlement needs to be granted before the VMs can be started with the Secure Boot option.

Once the entitlement has been granted, all new VMs will automatically be provisioned with the Secure Boot feature enabled.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Fast autoscaling in Cloudera operational database (Preview)

Date published: 2023-12-18
Date modified: 2024-10-01

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
Fast autoscaling in Cloudera operational database **4**
Introduction to Compute Instance Group 4
Metrics categorization 5
Fast autoscaling mechanism 6
Enabling fast autoscaling in COD **6**
Disabling fast autoscaling in COD **8**
Runtime upgrade of the cluster **8**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Fast autoscaling in Cloudera operational database

Cloudera Operational Database (COD) supports fast autoscaling for higher computing requirements. COD enables fast autoscaling whenever high CPU utilization or higher RPC latency is observed in the system.

The fast autoscaling is achieved using the following mechanisms.

Introduction to Compute Instance Group

A Compute node instance group in COD consists of nodes that host all the services, similar to those on Worker nodes, except the HDFS data services. The region servers on these nodes communicate with HDFS (data nodes) running on the Worker nodes.

The Compute nodes add to the system's higher compute capacity. During scale-up operations due to higher compute requirements, the Compute nodes are scaled up using the same storage as the existing Worker nodes. Hence, during the scale-down of these Compute nodes, the data remains in the Worker nodes and does not require data redistribution. The Worker nodes continue to host the data services and are scaled (up or down) according to the data requirements (such as Data storage and Regions per server).

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

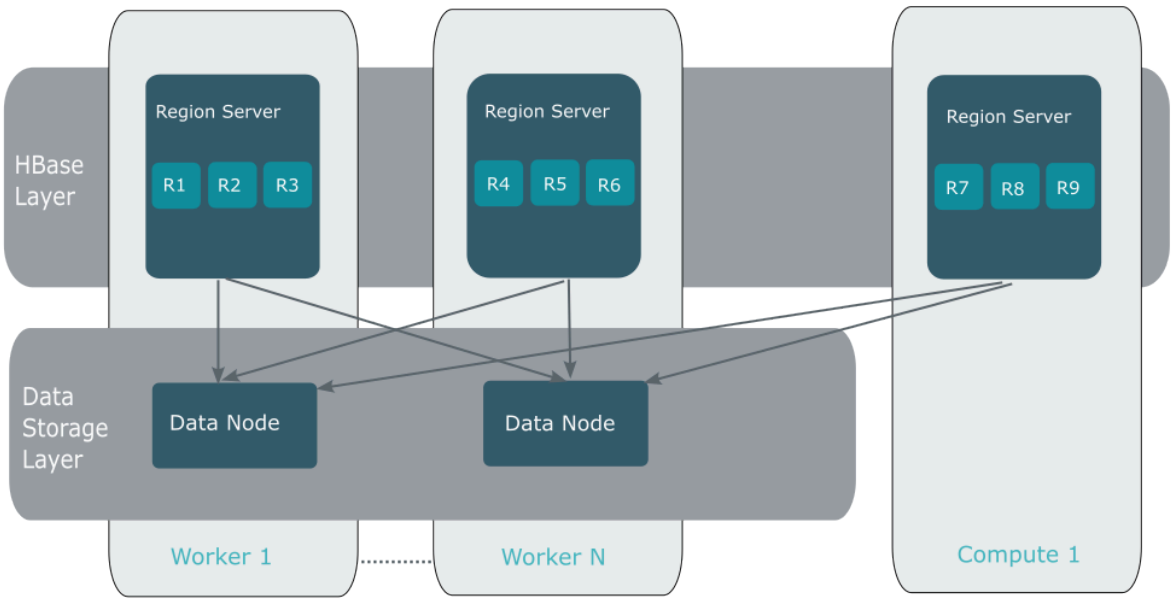

Compute

	Instance ID	Status	FQDN
>	i-0c8a437905093ec9d	Stopped	cod-zfo9yuhetvix-compute2.jrh19-co.xcu2-8y8x.dev.cldr.work
>	i-0c7d468b3b29aa1e7	Stopped	cod-zfo9yuhetvix-compute0.jrh19-co.xcu2-8y8x.dev.cldr.work
>	i-093b3c2b99fb1f45d	Running	cod-zfo9yuhetvix-compute1.jrh19-co.xcu2-8y8x.dev.cldr.work

Metrics categorization

The metrics used for autoscaling are categorized into the following groups: The data metrics scale the Worker instance group, while the compute metrics scale the Compute instance group.

- Worker Group (Data-related metrics)
    a. HDFS usage
    b. Region density (Regions per server)
    c. Total Store File size across region servers
- Compute Group (Compute-related metrics)
    a. CPU utilization
    b. RPC latency

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Fast autoscaling mechanism

COD relies on suspending and resuming (start and stop) autoscaling, where the maximum allowed number of nodes in the system are instantiated but kept in a Stopped state. This mechanism is used for the scaling of the Compute instance group.

When a requirement for higher compute power arises in the cluster, COD starts the required number of Compute nodes. The scale-up time is drastically reduced because the Compute nodes are already initialized. Similarly, when the requirement for computing goes down, the Compute nodes are stopped but not dropped. They remain in the cluster in the Stopped state. The scale-up and scale-down time is reduced since the instance initialization is skipped during every scale-up or scale-down operation.

COD supports the suspending and resuming of fast autoscaling for the Compute nodes only. The Worker nodes continue to use the existing autoscaling mechanism.

For more information, see the following topics:

- Autoscaling clusters.
- Faster Auto-scaling for Higher Computing Requirements

Enabling fast autoscaling in COD

Learn how to enable the fast autoscaling mechanism in COD.

Before you begin

- The cluster must have the COD setup in the environment.
- The node on which the steps are executed must have the CDP CLI installed.
- You must have the `COD_USE_COMPUTE_ONLY_NODES` entitlement.
- You must use COD 1.37.0 or a higher version.

Steps

1. Launch the CDP CLI tool.
2. Use the CDP CLI `create-database` command to create the compute instance group. Use the `--auto-scaling-parameters` option to specify the maximum and minimum number of compute nodes.

   ```
   cdp opdb create-database —environment-name <env_name>
   -database-name <db_name> --auto-scaling-parameters
   ```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
'{"minComputeNodesForDatabase":<min_compute_nodes>,
"maxComputeNodesForDatabase": <max_compute_nodes>}'
```

Where,

- *<max_compute_nodes>* specifies the maximum number of Compute nodes required in the system.
- *<min_compute_nodes>* specifies the minimum number of Compute nodes in the system.

  When the command is successfully executed, the newly created database contains the instantiated *<max_compute_nodes>* nodes. However, in the subsequent, scaling-executor cycle, the Compute nodes are stopped until the required number of nodes or the count *<min_compute_nodes>* is reached.

  Alternatively, if the database is already created on 1.37.0, we can use the update database CDP CLI command to update the autoscaling parameters mentioned above.

3. Configure the autoscaling parameter to have an appropriate maxCpuUtilization which represents the average CPU utilization percent across Worker nodes and Compute nodes.

   ```
   cdp opdb update-database --environment-name <env_name>
   --database-name <db_name> \
   --auto-scaling-parameters '{"maxCpuUtilization": 80. . . }'
   ```

   The subsequent scaling cycles start or stop the Compute nodes as per the compute requirements.

4. Execute the update-database CDP CLI command to start all the Compute nodes in the system, if the cluster needs to be stopped or upgraded.

   ```
   cdp opdb update-database --environment-name <env_name>
   --database-name <db_name> \
   --auto-scaling-parameters
   '{"minComputeNodesForDatabase":<max_compute_nodes>}'
   ```

   All the Compute nodes are started. The nodes are running after the cluster upgrade or restart.

5. Revert to the original <min_compute_node> after the restart or upgrade.

   ```
   cdp opdb update-database --environment-name <env_name>
   --database-name <db_name> \
   --auto-scaling-parameters
   '{"minComputeNodesForDatabase":<min_compute_nodes>}'
   ```

   The subsequent scaling cycles stop the additional Compute nodes in the cluster.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

For more information, see CDP CLI beta documentation.

Disabling fast autoscaling in COD

You can disable the fast autoscaling in your COD cluster. However, the existing autoscaling mechanism operates as usual in the cluster.

Before you begin

- The cluster must have the COD setup in the environment.
- The node on which the steps are executed must have the CDP CLI installed.
- Fast autoscaling must be enabled in your cluster.

Steps

1. Launch the CDP CLI tool.
2. Set the minimum and maximum Compute node counts to zero. This command deletes all the Compute nodes in the cluster.

   ```
   cdp opdb create-database —environment-name <env_name>
   -database-name <db_name> --auto-scaling-parameters
   '{"minComputeNodesForDatabase":0, "maxComputeNodesForDatabase":
   0}'
   ```

   The subsequent autoscaling executions only impact the Worker instance group.

   For more information, see CDP CLI beta documentation.

Runtime upgrade of the cluster

Learn how to resume the stopped nodes in the cluster before performing a runtime upgrade.

The cluster upgrade might fail if the cluster contains nodes that are in the Stopped state. You must restart these nodes before executing the upgrade or drop the stopped nodes before the upgrade. They can be added later after the upgrade.

Ensure that you execute the following command to increase the minimum number of nodes in the cluster to the maximum number of allowed Compute nodes in the cluster. This triggers the restart of the stopped nodes. After that, the cluster can be upgraded.

```
./clients/cdpcli/cdp.sh opdb update-database --environment-name
jrh16-cod-7216 --database-name jrh13-7216 \
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
        --auto-scaling-parameters
'{"minComputeNodesForDatabase":<MAX_COMPUTE_NODES>}'
```

Revert the `minComputeNodesForDatabase` to the original value after the upgrade.

Alternatively, drop all the stopped Compute nodes in the cluster using the following CDP CLI command.

```
cdp opdb create-database —environment-name <env_name> -database-name
<db_name> --auto-scaling-parameters '{"maxComputeNodesForDatabase":
<number_of running_compute_nodes in the system>}'
```

For more information, see [CDP CLI beta documentation](#).

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

GCS Fine-Grained Access Control (Preview)

Date published: 2023-06-29
Date modified: 2024-12-04

Legal Notice

© Cloudera Inc. 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
Introduction to RAZ in GCP environments **4**
Supported use cases for RAZ in GCP environments 5
Limitations for using RAZ in GCP environments 5
GCP requirements for RAZ-enabled GCP environments **6**
Registering a RAZ-enabled GCP environment **7**
Using Cloudera UI to register RAZ-enabled GCP environment 7
Using Beta CDP CLI to register RAZ-enabled GCP environment 8
Ranger policy options for RAZ-enabled GCP environment **9**
Troubleshooting **11**
Creation of Kudu table is failing 11
RAZ fails to authorize request 11

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Introduction to RAZ in GCP environments

Cloudera Public Cloud defaults to using cloud storage which might be challenging while managing data access across teams and individual users. The Ranger Authorization Service (RAZ) resolves this challenge by enabling Google Cloud Storage (GCS) users to use fine-grained access policies and audit capabilities available in Apache Ranger similar to those used with HDFS files in an on-premises or IaaS deployment.

Many of the use cases that RAZ for GCS enables are cases where access control on files or directories is needed. Some examples include:

- Per-user home directories.
- Data engineering (Spark) efforts that require access to cloud storage objects and directories.
- Data warehouse queries (Hive/Impala) that use external tables.
- Access to Ranger's rich access control policies such as date-based access revocation, user/group/role-based controls, along with corresponding audit.
- Tag-based access control using the classification propagation feature that originates from directories.

Prior to the introduction of RAZ, controlling access to GCS could be enforced at coarse-grained group level using IDBroker mappings. This required rearchitecting the implementation of important file-centric activities as well as admin-level access to both the Google Cloud account and the Cloudera account.

In HDP and CDH deployments, files and directories are protected with a combination of HDFS Access Control Lists (ACLs) (in CDH, HDP) and Ranger HDFS policies (in HDP). Similarly, in an GCP CDP Public Cloud environment with RAZ for GCS enabled, Ranger's rich access control policies can be applied to Cloudera's access to GCS buckets, directories, and files and can be controlled with admin-level access to Cloudera alone.

Important
It is recommended that you do not set up IDBroker mappings for workload users in a RAZ-enabled GCP environment.

The following diagram illustrates RAZ authorization process for controlling user access to GCS:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

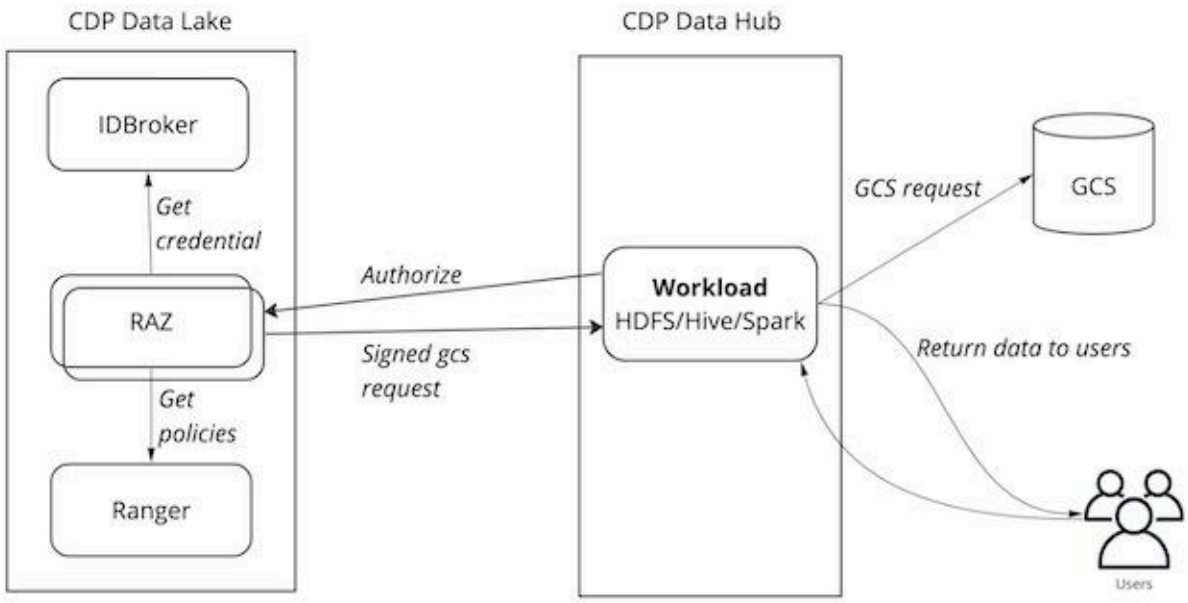

Supported use cases for RAZ in GCP environments

The core RAZ for GCP for Data Lakes and several Cloudera Data Hub templates are available. The following Cloudera Data Hub cluster types are supported with Cloudera Runtime 7.2.17 or newer:

- Data Engineering
- Data Engineering HA
- Data Engineering Spark3
- Data Mart
- Flow Management Light Duty
- Operational Database with SQL
- Streams Messaging HA

Limitations for using RAZ in GCP environments

The following limitations and known issues have been identified:

- RAZ on GCP uses HMAC keys for authentication. When multiple Cloudera environments are created on GCP with RAZ enabled, the account limit of 10 HMAC keys per service account may be reached. To avoid reaching the limit, you should create a new Ranger Cloud Access Authorizer service account for each Cloudera environment on GCP.
- Due to the fact that Google HMAC key generation cannot be limited in scope, the Ranger Cloud Access Authorizer service account has elevated credentials to create HMAC keys for any service account in the same GCP project. Therefore, for security reasons, it is

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

recommended that you create all Cloudera service accounts and storage buckets in a separate dedicated project.
- There is currently no automated way to enable RAZ in an existing CDP environment which does not have RAZ enabled. If you have a non-RAZ environment, you should create a new environment with RAZ, and then migrate your data using the steps described in Data Lake backup and restore documentation.
- Hue file browser is not supported.
- Flink is not supported.
- S3A access and GCS access may not work from the same cluster without configuration changes, because there are overlapping configuration parameter names for GCS and S3.
- Enabling RAZ for GCP is supported only with Cloudera Runtime 7.2.17 or newer.

GCP requirements for RAZ-enabled GCP environments

The minimal GCS cloud storage setup recommended for production is described in the Minimum setup for cloud storage for GCP. In addition to this, you should create the following service account for RAZ:

Service account name	Description	Required IAM roles or permissions	Scope
Ranger Cloud Access Authorizer	This service account will be used by Cloudera services to access workload data via RAZ. It provides full access to the data storage location.	Choose one of the following two approaches: - Option A: Use both built-in and custom roles - Option B: Use only a custom role **Option A: Use both built-in and custom roles:** ● Assign the **storage.objectAdmin** built-in GCP role ● Assign the **storage.hmacKeyAdmin** built-in GCP role ● Create and assign a custom role that grants the **storage.buckets.get** permission. **Option B: Use only a custom role:**	*Data storage* bucket For Data Lake backup and restore: *Backups* bucket, if different from the main data storage bucket

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

		Create a custom role with the below permissions and assign it: ● storage.buckets.get ● storage.hmacKeys.create ● storage.hmacKeys.delete ● storage.hmacKeys.get ● storage.hmacKeys.list ● storage.hmacKeys.update ● storage.multipartUploads.abort ● storage.multipartUploads.create ● storage.multipartUploads.list ● storage.multipartUploads.listParts ● storage.objects.create ● storage.objects.delete ● storage.objects.get ● storage.objects.getIamPolicy ● storage.objects.list ● storage.objects.setIamPolicy ● storage.objects.update	

Use the usual instructions to create GCS buckets, create the service accounts, create custom roles, and add service accounts as members and users to other service accounts. Refer to the instructions in the Minimum setup for cloud storage for GCP.

Note that this additional **Ranger Cloud Access Authorizer** service account mentioned above also needs to be added as a member to the *Data storage* bucket by using the steps described in Add service accounts as members to buckets.

Registering a RAZ-enabled GCP environment

You can use the Cloudera web interface or Beta CDP CLI to register a RAZ-enabled GCP environment.

You can enable RAZ on the latest available version of Cloudera Runtime. The minimum version supporting RAZ for GCP environments is Cloudera Runtime 7.2.17.

Using Cloudera UI to register RAZ-enabled GCP environment

Follow the usual instructions to register a GCP environment in Cloudera. Make sure to do the following:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

1. Select GCP as the cloud provider and then provide the necessary details.
2. Make sure to select the Cloudera Runtime version 7.2.17 or newer.
3. To enable Ranger RAZ authorization, click on **Enable Authorization for Google Cloud Storage** and then select the **Ranger Cloud Access Authorizer** service account (that you created earlier) from the dropdown:

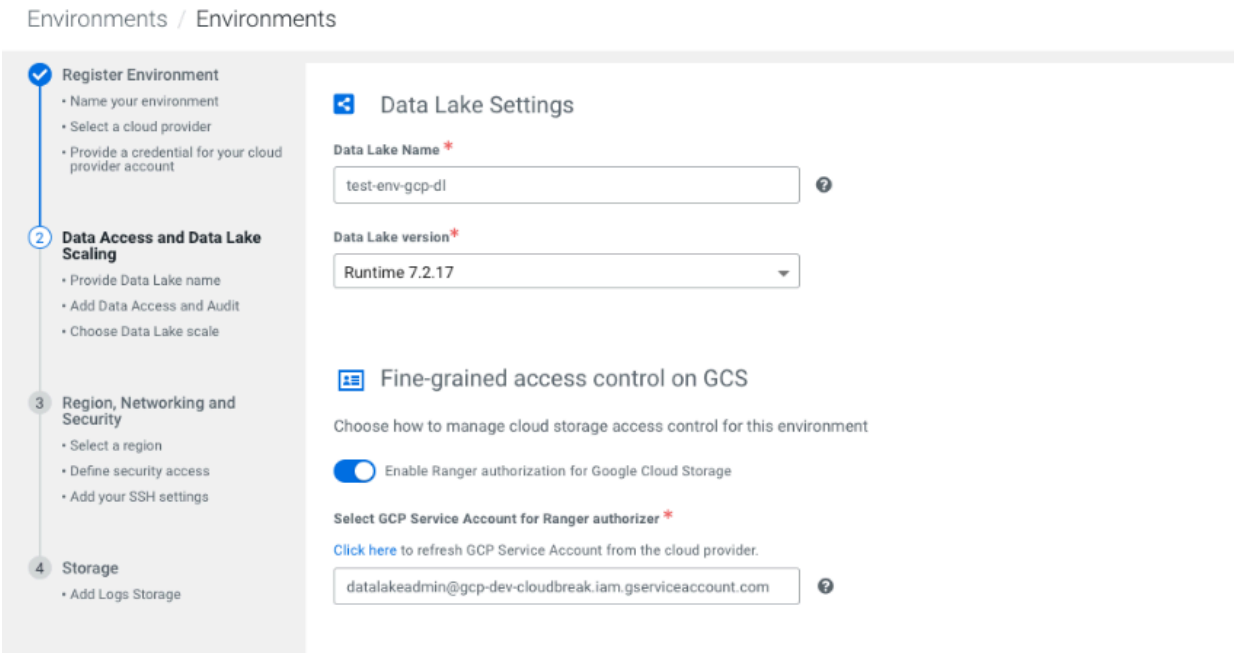

4. Continue registering the environment as usual.

Using Beta CDP CLI to register RAZ-enabled GCP environment

RAZ for GCP is supported via Beta CLI. Below are the steps to configure RAZ via Beta CDP CLI:

1. Install Beta CLI. See Install Beta CDP CLI.
2. Create a GCP environment with CDP CLI by using the `cdp environments create-gcp-environment` command. For more information, see Register a GCP environment from CDP CLI.
3. Use the following command to set empty IDBroker mappings for the environment. Since RAZ is to be configured, `ranger-cloud-access-authorizer-role` also needs to be set. Below is a CLI command example:

```
Unset
cdp environments set-id-broker-mappings \
--environment-name myraz-gcp-env \
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
--data-access-role
datalakeadmin@gcp-dev-cloudbreak.iam.gserviceaccount.com \
--set-empty-mappings \
--ranger-audit-role
rangeraudit@gcp-dev-cloudbreak.iam.gserviceaccount.com \
--ranger-cloud-access-authorizer-role
datalakeadmin@gcp-dev-cloudbreak.iam.gserviceaccount.com
```

4. Create RAZ enabled Data Lake using Beta CLI. Note that the `--enable-ranger-raz` option is provided to enable RAZ. Runtime version should be greater than or equal to 7.2.17. Below is a CLI command example:

```
Unset
./cdp.sh datalake create-gcp-datalake \
--environment-name myraz-gcp-env \
--datalake-name myraz-gcp-dl \
--runtime 7.2.17  \
--cloud-provider-configuration
"serviceAccountEmail=idbroker@gcp-dev-cloudbreak.iam.gserviceacco
unt.com,storageLocation=gs://perf-team-west2-bucket" \
--enable-ranger-raz --scale LIGHT_DUTY
```

5. Once Data Lake is created successfully, log in to Cloudera Manager web UI and verify that ranger-RANGER_RAZ server has been configured.

Ranger policy options for RAZ-enabled GCP environment

The default Ranger policies for GCS are located under the **GS Tab (ServiceDef)** in the **cm_gs repo:**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

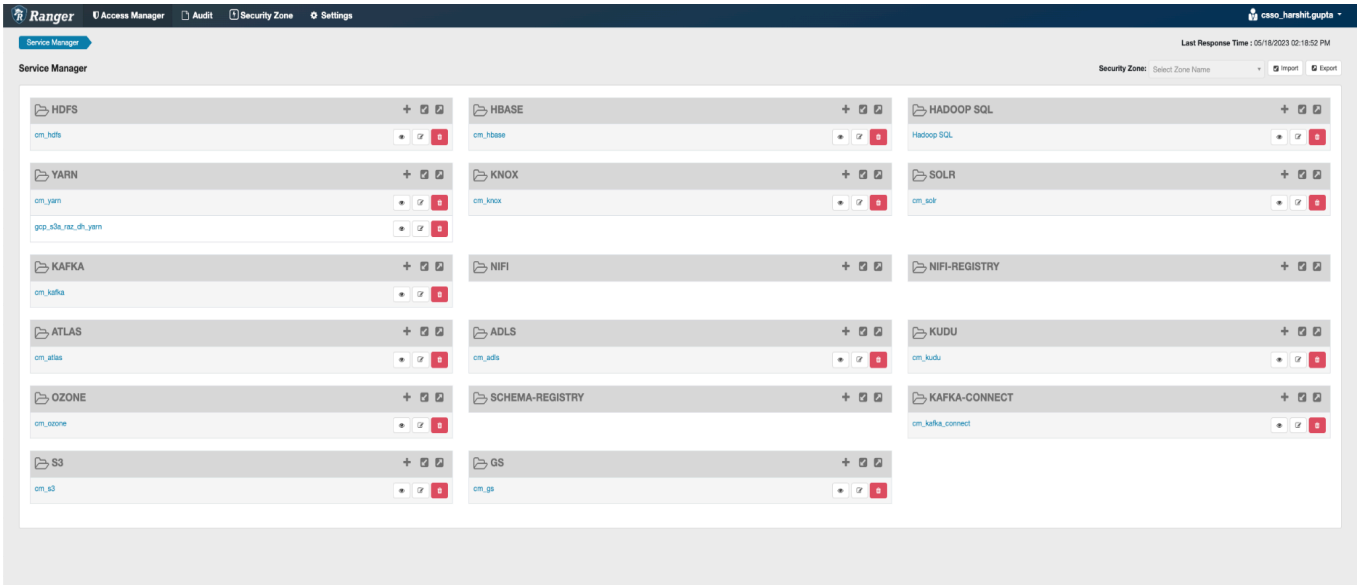

This is the default view of the **cm_gs** repo with the default Ranger policies specified:

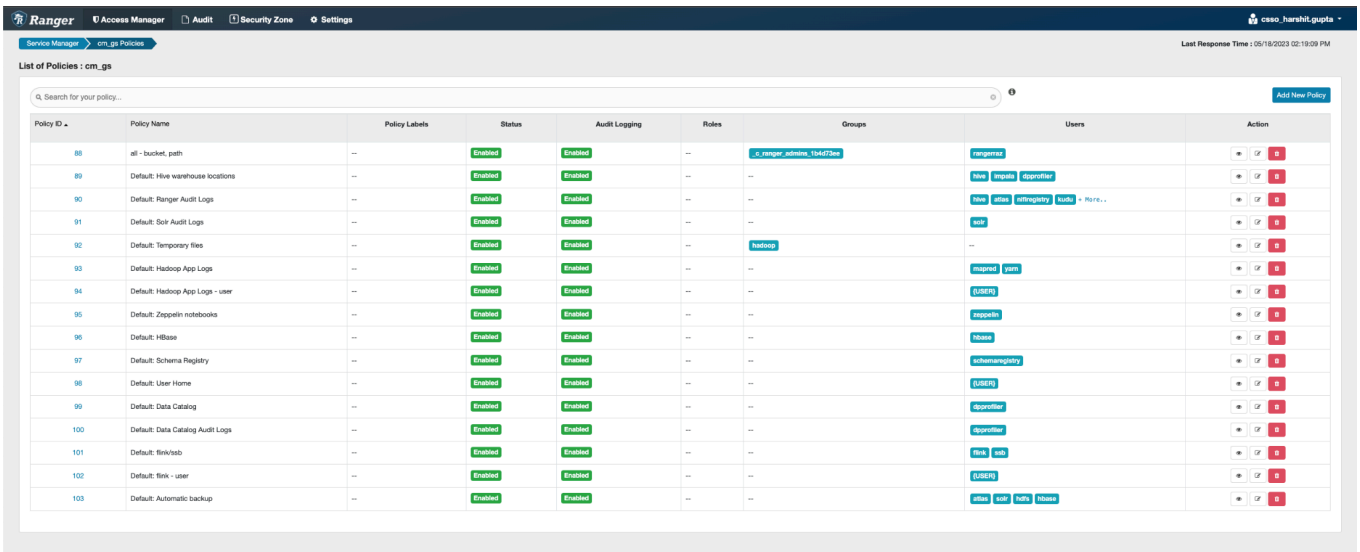

The Ranger policy creation works similar to that of RAZ for S3. You just specify the GCS bucket name, the path, and the policy parameters:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

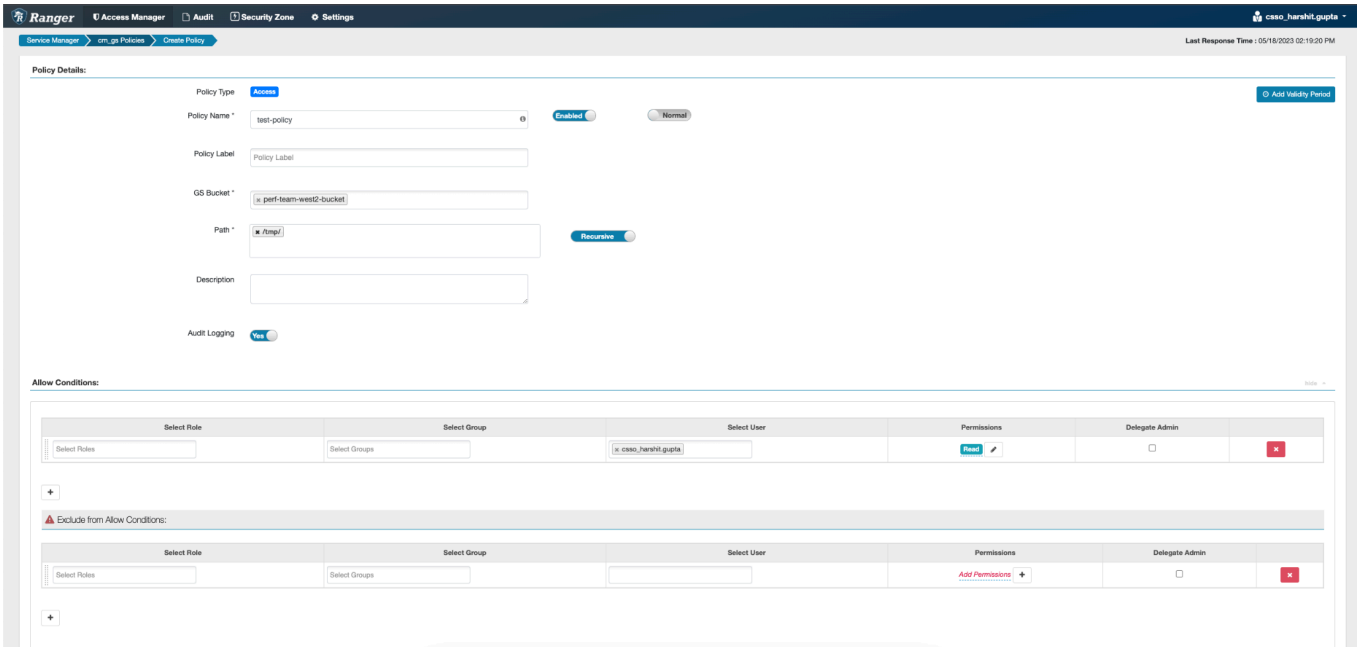

Troubleshooting

Refer to this section to learn about errors that you may encounter and steps to resolve them.

Creation of Kudu table is failing

Problem: The Kudu service user is not authorized to access Hive warehouse locations on cloud object stores, which under certain conditions can prevent Kudu tables from being created. This results in the following error:

```
ImpalaRuntimeException: Error creating Kudu table 'default.truckspeedevents'
CAUSED BY: NonRecoverableException: failed to create HMS catalog entry for table
[id=b764bceb167746b7bb3dc1e8722e66e6]: failed to create Hive MetaStore table:
TException - service has thrown: MetaException(message=Got exception:
java.nio.file.AccessDeniedException
```

Workaround: Add "kudu" service user to the allow list for "Default: Hive warehouse locations" in the `cm_gs` Ranger repository.

RAZ fails to authorize request

Problem: RAZ fails to authorize a request with the following error.

StackTrace on Client side:

```
hdfs dfs -ls gs://perf-team-west2-bucket/

...

CLOUD_PROVIDER_ERROR:
org.apache.ranger.raz.s3.lib.exceptions.RCSUECloudProviderException: Could not
determine region for bucket: [perf-team-west2-bucket]

org.apache.ranger.raz.s3.lib.signer.util.ExceptionUtils.createUnableToFindRegionExce
ption(ExceptionUtils.java:29)

org.apache.ranger.raz.processor.s3.signer.S3RemoteSigner.getRegion(S3RemoteSign
er.java:217)

org.apache.ranger.raz.processor.s3.signer.S3RemoteSigner.getAuthorizationRequest(
S3RemoteSigner.java:379)

org.apache.ranger.raz.processor.s3.S3RazProcessor.getRangerAccessRequests(S3Ra
zProcessor.java:183)

org.apache.ranger.raz.processor.s3.S3RazProcessor.preProcess(S3RazProcessor.jav
a:120)

org.apache.ranger.raz.RangerRemoteAuthorizer.authorize(RangerRemoteAuthorizer.ja
va:151)

org.apache.ranger.raz.rest.AuthzREST.authorizeAccess(AuthzREST.java:98)
ls: gs://perf-team-west2-bucket/:
org.apache.ranger.raz.hook.s3.RazS3ClientCredentialsException:

....

23/05/16 08:10:50 INFO impl.MetricsSystemImpl: Stopping s3a-file-system metrics
system...

23/05/16 08:10:50 INFO impl.MetricsSystemImpl: s3a-file-system metrics system
stopped.
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
23/05/16 08:10:50 INFO impl.MetricsSystemImpl: s3a-file-system metrics system
shutdown complete.
```

StackTrace on RAZ server side:

```
com.amazonaws.SdkClientException: Unable to load AWS credentials from any
provider in the chain:
[org.apache.ranger.raz.processor.gs.RazGSCredentialProvider@4eeb9a91:
RazGSCredentialsProvider.java: Exception while creating HMac Keys in GCP. If service
account HMAC key limit reached, manually remove the key from GCP console and
restart the service.

Root cause: { com.google.cloud.storage.StorageException: Exception from GCP:
Service account HMAC key limit reached }]

com.amazonaws.auth.AWSCredentialsProviderChain.getCredentials(AWSCredentials
ProviderChain.java:136)

org.apache.ranger.raz.s3.lib.utils.S3RegionUtils.lookupRegion(S3RegionUtils.java:118)

org.apache.ranger.raz.processor.s3.signer.S3RemoteSigner.getRegion(S3RemoteSign
er.java:212)

org.apache.ranger.raz.processor.s3.signer.S3RemoteSigner.getAuthorizationRequest(
S3RemoteSigner.java:379)

org.apache.ranger.raz.processor.s3.S3RazProcessor.getRangerAccessRequests(S3Ra
zProcessor.java:183)

org.apache.ranger.raz.processor.s3.S3RazProcessor.preProcess(S3RazProcessor.jav
a:120)

org.apache.ranger.raz.RangerRemoteAuthorizer.authorize(RangerRemoteAuthorizer.ja
va:151)

org.apache.ranger.raz.rest.AuthzREST.authorizeAccess(AuthzREST.java:98)
```

Workaround: As mentioned in the stack trace of server logs above, manually remove the key from the GCP console and restart the RAZ service.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

HBase Time-based Data Tiering using Persistent BucketCache (Preview)

Date published: 2024-10-01
Date modified: 2024-10-09

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2
Contents 3
HBase Time-based Data Tiering using Persistent BucketCache 4
Configuration details 6
Enabling the time-based data tiering 7

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

HBase Time-based Data Tiering using Persistent BucketCache

You can configure time-based priority caching for HBase, define a specific time range for data to remain in the cache, and choose to evict older data. Based on the configuration, more recent data is given a higher priority.

Introduction

Using this functionality, you can optimize storage efficiency and access performance by segregating data based on its recency. By keeping recent data in the cache and evicting older data, the system aims to provide more flexible control over the cache allocation and eviction logic through configuration, allowing to define time priorities for cached data. The need for a more extensive cache allocation mechanism becomes even more critical on HBase deployments where cache access reflects on significant performance gains, such as when using cloud storage as the underlying file system.

The following figure depicts how data is fetched between the bucket cache (hot tier or SSD storage) and file system (cold tier or cloud storage).

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

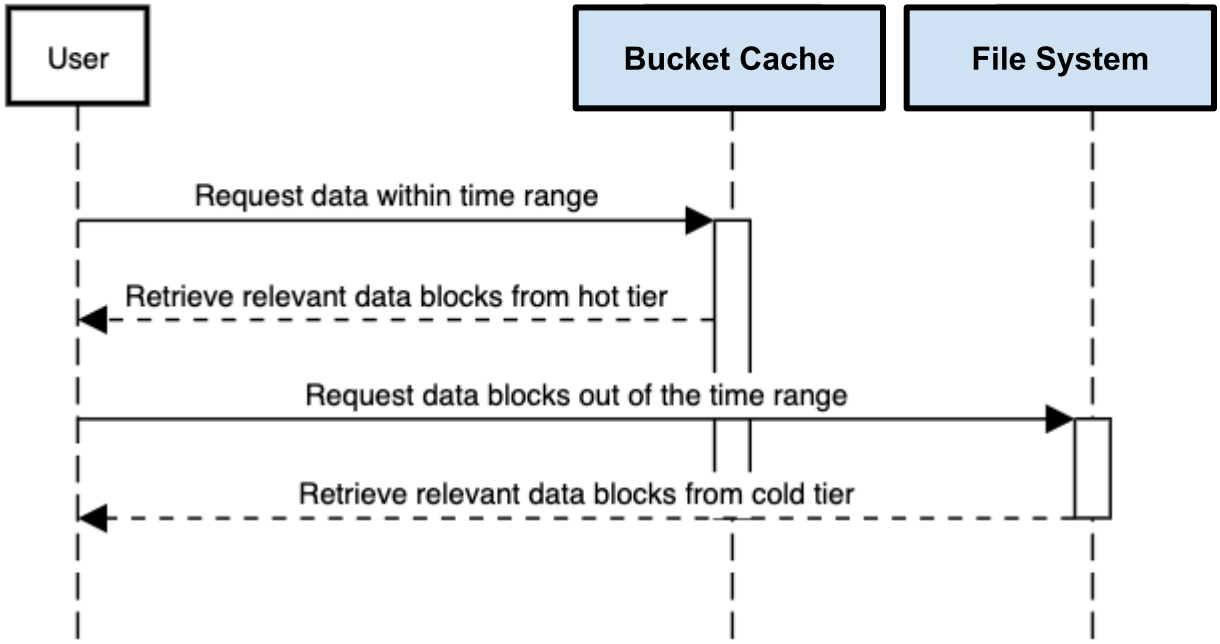

Components

The following are the key components involved in the time-based data tiering functionality.

- Hot tier (SSD storage): This tier consists of high-speed local storage, typically SSD storage, where recent data is cached for fast access. This is based on the BucketCache implementation in HBase.
- Cold tier (Cloud storage): This tier comprises cost-effective cloud storage for storing older data.
- Compaction: Organizes data into files based on the provided time range and compaction strategy. Date-tiered compaction is a date-aware store file compaction strategy that categorizes data into files based on time, making it a prerequisite for time-based data tiering.
- Eviction policy: Determines which data blocks should be evicted from the cache based on the provided time range.
- Prefetch logic: Controls the prefetching of data files, ensuring that only relevant files within the user-specified time range are processed.

Workflow

The following workflow shows how data is processed with the time-based data tiering functionality.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- Data ingestion:
  - Data is automatically flushed from the memstore to the cache based on a predefined configuration (cacheOnWrite). Hence the recent data always stays in the cache.
- Compaction:
  - When the date-tiered compaction is enabled, the recent data within the specified time range is compacted into smaller files, while older data is grouped into larger files. Additionally, if cacheCompactedDataOnWrite is enabled, the data is sent for caching, and a time-based data tiering is performed again to cache only "hot" data while declining to cache the "cold" data.
- Eviction:
  - The eviction policy determines which data blocks in the cache do not belong to the specified time range.
  - When the cache reaches its capacity and evictions must be triggered, the blocks containing data outside the user-provided time range are prioritized for eviction but not necessarily evicted every time. If only a small portion of the data within the specified time range is sufficient to make space for the incoming blocks, the eviction process stops, and the remaining data outside the time range stays in the cache.
- Prefetching:
  - When accessing data within the specified time range, the prefetch logic ensures that only relevant files are fetched from the cloud storage for caching.
  - Files outside the specified time range are not prefetched, reducing unnecessary data transfer and cache usage.

Configuration details

Know the configuration parameters to enable the time-based data tiering functionality and set the data age priorities.

1. Log in to the HBase shell.
2. Use the `ALTER` command to modify the following configuration parameters at the table or column family level.

Configuration parameter	Description	Supported values
hbase.hstore.datatiering.type	Set this parameter to enable the time-based data tiering at the table or column family	TIME_RANGE Default value: NONE

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Configuration parameter	Description	Supported values
	levels.	
hbase.hstore.datatiering.hot.age.millis	Specifies the time duration for which the data is considered hot and must have the highest priority to be kept in the cache. Set this parameter at the table level to implement the time-based data-tiering for all the table's column families, or at the column-family level to set the functionality for the column family.	Default value: 7 Days or 604800000 milliseconds
hbase.hstore.engine.class	Set this parameter to enable the date-tiered compaction at the table or column family level.	org.apache.hadoop.hbase.regionserver.DateTieredStoreEngine

Enabling the time-based data tiering

Know how to enable the time-based data tiering in HBase.

About this task

To use the data tiering functionality at the table or column-family level, you must set the global data tiering configuration, **hbase.regionserver.datatiering.enable** to true. This configuration activates the data tiering across your HBase instances.

This task demonstrates how to enable the global data tiering across multiple HBase instances.

Note:
This functionality is only beneficial when the write access pattern primarily consists of sequential writes based on the data arrival time to the backend, and the read access pattern mainly involves accessing the recent data.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Prerequisites

- Ensure that the bucket cache is enabled for the cluster. This is mandatory for the data tiering to function. For more information, see HBase persistent BucketCache.
- Before enabling the time-based data tiering at the table or column-family level, you must enable the data-tiered compaction using the **hbase.hstore.engine.class** configuration. For more information on Apache data-tiered compaction, see Date Tiered Compaction.

Steps

1. Log in to the Cloudera Manager as an Administrator.
2. Select the **HBase** service.
3. Go to **Configuration > Advanced > HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml**.
4. Set the value of **hbase.regionserver.datatiering.enable** as `true` to enable the global data tiering.
5. Click **Save Changes**.
6. Restart the HBase service.

Post-requisites

- After enabling the global data tiering configuration, you can start using the data tiering feature at the table or column-family level. This feature is effective only when used with the date-tiered compaction strategy. Cloudera recommends enabling the date-tiered compaction with the additional configurations to optimize it for the tables or column families and data-tiering.
  Refer to the following example.

```
$ hbase > alter '<table_name>', CONFIGURATION =>
{'hbase.hstore.datatiering.type' => 'TIME_RANGE',
'hbase.hstore.datatiering.hot.age.millis' => '<hot_age>'}
```

- Run the following command in the HBase shell to check whether the data tiering functionality is enabled.

```
$ hbase shell
$ hbase > describe <table_name>
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Expected output:

```
hbase:009:0> describe 'usertable'
Table usertable is ENABLED
usertable, {TABLE_ATTRIBUTES => {METADATA => {
  'hbase.hstore.datatiering.hot.age.millis' => '3600000',
  'hbase.hstore.datatiering.type' => 'TIME_RANGE',
  'hbase.hstore.engine.class' =>
'org.apache.hadoop.hbase.regionserver.DateTieredStoreEngine'
}}}
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS
=> '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING =>
'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS =>
'0', BLOCKCACHE => 'true', BLOCKSIZE => '1024', REPLICATION_SCOPE =>
'0'}

1 row(s)
Quota is disabled
Took 0.0392 seconds
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Horizontal scaling for the Data Lake (Preview)

Date published: 2024-03-22
Date modified: 2024-12-04

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2
Contents 3
Overview 3
Prerequisites for Data Services 4
Horizontally scaling Data Lake services through the CDP CLI 6
Horizontal scaling of Solr instances 7
Post-requisites for Data Services 7

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Overview

Certain Data Lake services may require additional instances as the Data Lake grows, or performance for these services can suffer. With an enterprise Data Lake (EDL), several of these services can be scaled horizontally, meaning that you can add additional instances to dedicated host groups for some services.

Horizontal scaling for these services prevents you from having to scale a base host group like the gateway or core host groups. Vertically scaling the gateway or core host groups is not cost-efficient, as they may include more services than what you want to scale.

The enterprise Data Lake (EDL) contains specific host groups for horizontal scaling of the following services, which can all be scaled up individually:

- Solr
- Kafka
- Ranger Authorization Service (RAZ)
- Hive metastore (HMS)
- Storage

Note: Upscaling the Solr host group requires manual steps after the scaling action to rebalance the collection replicas. Contact Cloudera Customer Support for details.

The following hostgroups can also be scaled down:

- Solr
- Ranger Authorization Service (RAZ)
- Hive metastore (HMS)

The storage and Kafka host groups cannot be scaled down once they have been scaled up, as the action might result in data loss.

Horizontal scaling of Data Lake services is a technical preview feature under entitlement and is currently available only through the beta CDP CLI.

Limitations:

- Currently, Atlas is not supported for horizontal scaling despite having a dedicated host group in the EDL. Support for Atlas will be added in the future.
- The Kafka and Storage host groups cannot be scaled down through this feature.
- As the EDL can have a total of 40 nodes and 10 are used by default, the horizontal scaling host groups combined are limited to 30 nodes.
- Upscaling the Solr host group requires manual steps after the scaling action to rebalance the collection replicas.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Prerequisites for Data Services

Prior to horizontally scaling the Data Lake, ensure that the following are in place:

1. If you are using Cloudera Data Warehouse, you must upgrade to version 1.4.1 or higher before you can resize the Data Lake. Determine the Cloudera Data Warehouse version you are on by clicking edit on the environment:

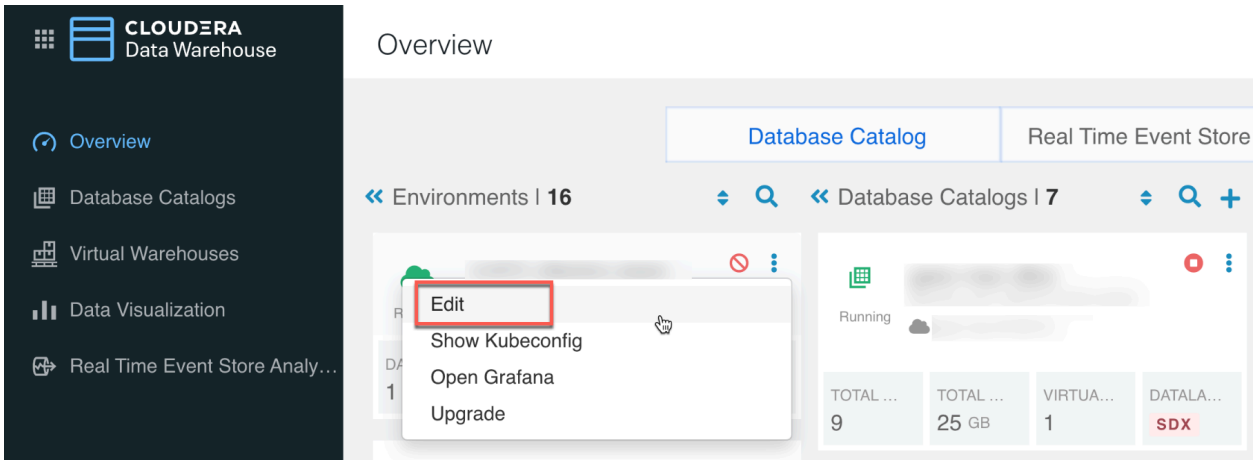

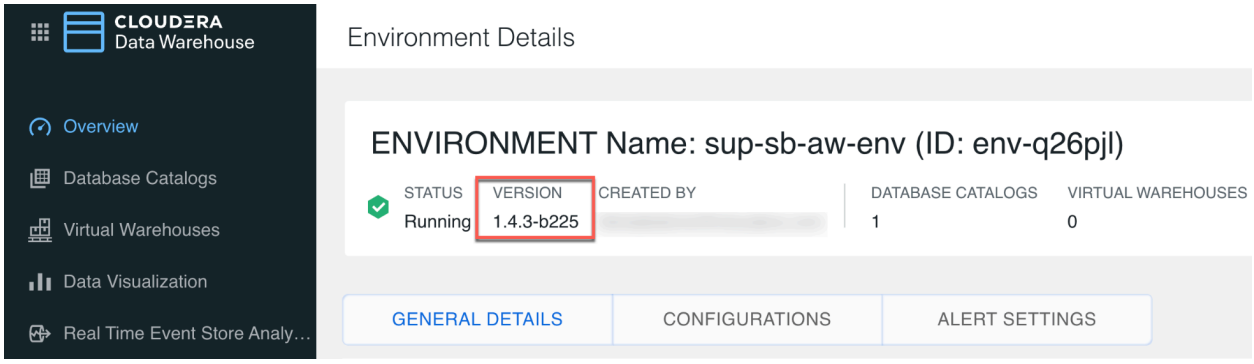

2. If you are using Cloudera Data Warehouse, stop the virtual warehouses and data catalogs associated with the environment.
3. If you are using Cloudera Data Engineering, do one of the following:
    1. Upgrade to Cloudera Data Engineering 1.15, or
    2. Create a new service.
        1. Take a backup of your jobs following Backing up Cloudera Data Engineering jobs.
        2. Create a new Cloudera Data Engineering service and virtual cluster.
        3. Restore the jobs following the instructions in Restoring Cloudera Data Engineering jobs from backup.
4. If you are using Cloudera AI:
    1. Backup Cloudera AI workbenches (AWS only). If backup is not supported, then proceed to the next step.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

2. Suspend Cloudera AI workspaces. If the suspend capability is not available, follow the steps in Refreshing Cloudera AI governance pods after resizing the Data Lake.

Horizontally scaling Data Lake services through the CDP CLI

If you have not already done so, download the beta CDP CLI. You do not need to stop the Data Lake, and both Cloudera Data Hubs and Data Services can continue to run during this operation.

To horizontally scale a service, run the following command in the CLI:

```
cdp datalake scale-horizontally --datalake-name <DATALAKE_NAME>
--instance-group-name <INSTANCE_GROUP_NAME>
--instance-group-desired-count <DESIRED_COUNT>
```

Option	Description
`-datalake-name`	Name or CRN of the Data Lake to be horizontally scaled.
`-instance-group-name`	Name of the service host group that you want to scale up or down. Valid inputs are: ● Solr_scale_out ● Storage_scale_out ● Kafka_scale_out ● Raz_scale_out ● Hms_scale_out Note that the Storage and Kafka groups are not available for downscale.
`-instance-group-desired-count`	Total number of instances that the service host group should contain after the scaling action finishes. For instance, if the host group already contains one instance, and you provide “3” for `-instance-group-desired-count`, the scaling action will add two instances to the host group. If you provide a number that is less than the current number of instances for the host

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

	group, the action will downscale the host group by removing nodes to the desired count.

You can monitor the progress of the scaling operation from the Data Lake *Event History* tab.

Important: If the horizontal scaling fails, due to issues on the cloud provider side or other issues, use the cdp datalake retry-datalake CLI command once you have fixed the issue that is causing the failure. The **`retry-datalake`** command will resume the horizontal scaling flow.

Important: To use the new instances added after an upscale, you must refresh the Cloudera Data Hub clusters that run these services after the horizontal scaling action completes. You can either stop and start the Cloudera Data Hub, or you can use the `cdp datalake refresh-datahub` CLI command:

```
cdp datalake refresh-datahub -datalake <datalake_name> -datahub
<datahub_name>
```

Option	Description
`-datalake`	Name or CRN of the Data Lake.
`-datahub`	Name or CRN of the Cloudera Data Hub to refresh. If none is provided, all Cloudera Data Hub clusters associated with the Data Lake are refreshed.

After the refresh, the Cloudera Data Hub clusters will use the new instances that were added to the Data Lake during the upscale.

Horizontal scaling of Solr instances

After you scale the Solr host group horizontally, you may need to perform some manual steps so that the new instances are available for Solr operations. These steps are required to rebalance the collection replicas. For assistance with scaling Solr, contact Cloudera Customer Support.

Post-requisites for Data Services

After horizontally scaling the Data Lake, complete the following tasks if you use the noted Data Services.

Cloudera AI:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

If you did not suspend your Cloudera AI workspaces, refresh the Cloudera AI governance pods after the horizontal scaling is complete.

Cloudera Data Warehouse: Start the Data Catalogs and Virtual Warehouses. For each virtual warehouse, Cloudera recommends that you start, stop, and start again. This will completely refresh the Data Lake details for the virtual warehouse.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

SQL AI Assistant in DataHub (Preview)

Date published: 2024-02-29
Date modified: 2024-02-29

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
About the Hue SQL AI Assistant **4**
Which AI models and services does Hue use? **4**
What data is shared with the LLM models? **5**
Prerequisites for enabling the SQL AI Assistant **5**
Configuring the SQL AI Assistant on the Microsoft Azure OpenAI Service **5**
Configuring the SQL AI Assistant on the OpenAI Service **7**
Service and model-related configurations for setting up the SQL AI Assistant in DataHub **9**
How to use the SQL AI Assistant? **10**
Generating SQL from NQL 10
Editing the query in natural language 11
Getting an explanation of a SQL query in natural language 11
Optimizing a query 12
Fixing a SQL query 13
Limitations **14**

About the Hue SQL AI Assistant

A SQL AI Assistant has been integrated into Hue with the capability to leverage the power of Large Language Models (LLMs) for various SQL tasks. It helps you to create, edit, optimize, fix, and succinctly summarize queries using natural language and makes SQL development faster, easier, and less error-prone. The Hue AI assistant is available with Cloudera Runtime 7.2.18 and higher.

Note: *The Hue SQL AI assistant is in technical preview and not recommended for use in production deployments. Cloudera recommends that you try this feature in test and development environments.*

Which AI models and services does Hue use?

The AI Assistant supports various LLMs and hosting services. The model can run in the infrastructure of a service provider, and the AI Assistant can be configured to use them remotely. Cloudera has tested with GPT running in Azure OpenAI, and the following service-model combinations are supported

Service Provider	Model	Model versions
OpenAI	OpenAI GPT	gpt-3.5-turbo gpt-3.5-turbo-16k
Microsoft Azure	OpenAI GPT	gpt-3.5-turbo gpt-3.5-turbo-16k
Amazon Bedrock	Anthropic Claude	anthropic.claude-v1 anthropic.claude-v2
Amazon Bedrock	Amazon Titan	amazon.titan-text-express-v1

Note: *Cloudera recommends using the Hue AI assistant with the Azure OpenAI service.*

For augmenting the results, the SQL AI Assistant uses a Retrieval Augmented Generation (RAG)-based architecture. It uses the sentence-transformer library for semantic search, and Hue can be configured with any of these pre-trained models for better multi-lingual support. By default, **all-MiniLM-L6-v2** models are used.

Embedding Model	Language Support
all-MiniLM-L6-v2	English only

distiluse-base-multilingual-cased-v1	Arabic, Chinese, Dutch, English, French, German, Italian, Korean, Polish, Portuguese, Russian, Spanish, and Turkish.

What data is shared with the LLM models?

The following details are shared with the Large Language Models (LLMs):

- Everything that a user inputs
- Dialect in use
- Table details such as table name, column names, column data types and related keys, partitions, and constraints that the logged-in user has access to.
- Three sample rows from the tables (as per the best practices specified in *Rajkumar et al, 2022*)

Prerequisites for enabling the SQL AI Assistant

As an administrator, you must obtain clearance from your organization's infosec team to make sure it is safe to use the SQL AI Assistant because some of the table metadata and data, as mentioned in the previous section, is shared with the LLM.

Configuring the SQL AI Assistant on the Microsoft Azure OpenAI Service

Microsoft Azure allows dedicated deployments of OpenAI GPT models. Using Azure's OpenAI service is much more secure than the publicly hosted OpenAI APIs because the data can be processed in your Virtual Private Cloud (VPC). Due to security considerations, Cloudera recommends that you use GPT models in Hue SQL AI Assistant with Azure's OpenAI service.

Steps

1. Obtain a Microsoft Azure subscription by working with your organization's IT team. Subscriptions vary based on your team and purpose.
2. Register to access the Azure OpenAI service.
   Azure OpenAI requires registration and is currently only available to approved enterprise customers and partners. Customers who wish to use Azure OpenAI are required to submit a registration form.
3. Create an Azure OpenAI resource in the Azure portal.

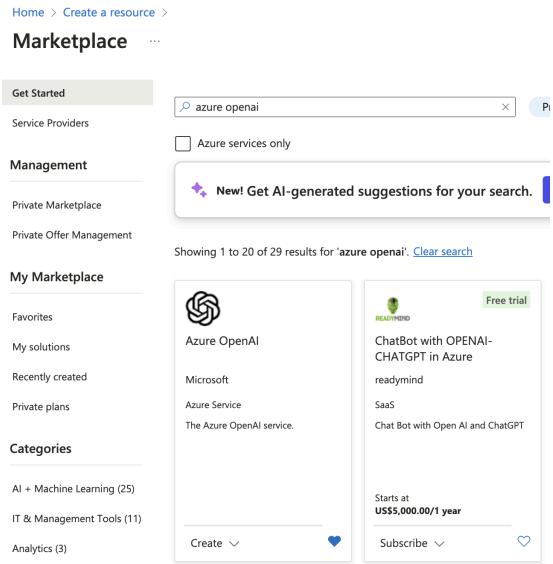

4. Obtain the resource URL and resource keys from the **Develop** tab under the resource details page.
   You can use any one of the two keys as shown in the following image:

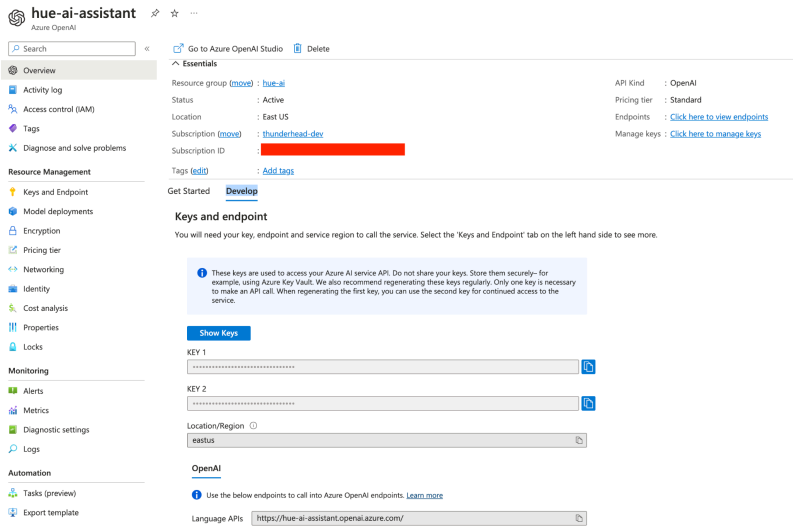

5. Go to Azure OpenAI Studio at https://oai.azure.com/portal and create your deployment under **Management** > **Deployments**. Select **gpt-35-turbo-16k** or higher.
6. Enable the Hue SQL AI Assistant as follows:
   a. Log in to the CDP Management Console as an Administrator.
   b. Go to **Environments** and select your environment.
   c. Go to the **Data Lake** tab and click on the CM URL to open Cloudera Manager.
   d. Go to **Clusters** > **Hue service** > **Configuration** and select add the following lines in the **Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini** field:
   ```
   [desktop]
   ```

```
[[ai_interface]]
    service='azure'
    model_name='[***DEPLOYMENT-NAME***]'
    base_url="https://[***RESOURCE***].openai.azure.com/"
    token="[***RESOURCE-KEY***]”
```

e. Click **Save Changes**.

Configuring the SQL AI Assistant on the OpenAI Service

Before you begin

You must have created an account with the OpenAI platform.

Steps

1. Log in to the OpenAI portal.
2. Obtain the API key by navigating to the API keys from the left navigation pane. The key is required to integrate Hue SQL AI Assistant with the OpenAI service.

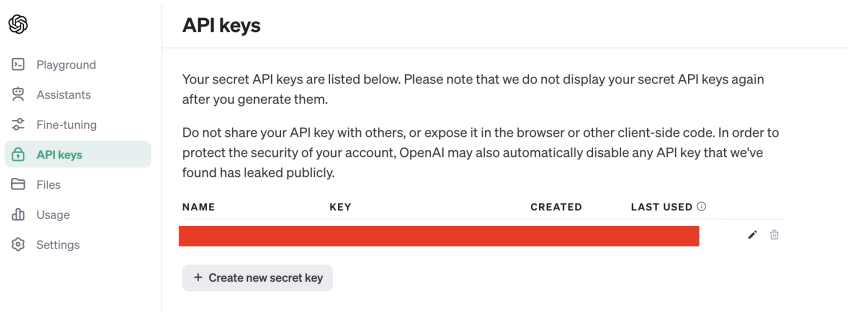

3. Enable the Hue SQL AI Assistant as follows:
    a. Log in to the CDP Management Console as an Administrator.
    b. Go to **Environments** and select your environment.
    c. Go to the **Data Lake** tab and click on the CM URL to open Cloudera Manager.
    d. Go to **Clusters** > **Hue service** > **Configuration** and select add the following lines in the **Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini** field:

```
[desktop]
     [[ai_interface]]
          service='openai'
          token='[***API-KEY***]'
```

    e. Click **Save Changes**.

Configuring the SQL AI Assistant on the Amazon Bedrock Service

Amazon Bedrock is a fully managed service that makes foundation models from leading AI startups and Amazon available through an API.

Before you begin
You must have an AWS account with Bedrock access.

Steps

1. Log in to the Amazon Bedrock service.
2. Obtain your access key and secret as follows:
    a. Go to the IAM console: https://console.aws.amazon.com/iam
    b. Click on **Users** from the left menu and select the user you want to access.
    c. Click on **Security credentials**.
    d. Go to the **Access keys** section and note the access keys.
3. Establish Anthropic Claude access.
   Claude from Anthropic is one of the best models available in Bedrock for SQL-related tasks. By default, Claude is not available on Bedrock. You need to place a special request for Claude. Once you have access, you can try Claude in the text playground under the Amazon Bedrock service. If you are in the us-east-1 region, this must take you to https://us-east-1.console.aws.amazon.com/bedrock/home?region=us-east-1#/text-playground.
4. Enable the Hue SQL AI Assistant as follows:
    a. Log in to the CDP Management Console as an Administrator.
    b. Go to **Environments** and select your environment.
    c. Go to the **Data Lake** tab and click on the CM URL to open Cloudera Manager.
    d. Go to **Clusters** > **Hue service** > **Configuration** and select add the following lines in the **Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini** field:

```
[aws]
      [[bedrock_account]]
            access_key_id='[***ACCESS-KEY***]'
            secret_access_key='[***SECRET-KEY***]'
            region='us-east-1'
[desktop]
      [[ai_interface]]
            service='bedrock'
            model='claude'
```

5. Click **Save Changes**.

Service and model-related configurations for setting up the SQL AI Assistant in DataHub

You can configure the AI services and models you want to use by going to **Cloudera Manager** > **Clusters** > **Hue service** > **Configurations** > **Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini** and adding the following lines:

```
[desktop]
      [[ai_interface]]
            [***CONFIG-KEY1***]='[***VALUE***]'
            [***CONFIG-KEY2***]='[***VALUE***]'
      [[semantic_search]]
            [***CONFIG-KEY1***]='[***VALUE***]'
            [***CONFIG-KEY2***]='[***VALUE***]'
```

Specify the service and model-related configurations under the [[ai_interface]] section as listed in the following table:

AI interface config key	Description
service	API service to be used for AI tasks. AI is disabled when a service is not configured. For example, `azure`, `openai`, `bedrock`, and `ai_assistant`.
trusted_service	Indicates whether the LLM is trusted or not. Turn on to disable the warning. The default value is "`True`".
model	The AI model you want to use for AI tasks. For example, `gpt`, `llama`.
model_name	The fully qualified name of the model to be used. For example, `gpt-3.5-turbo-16k`.
base_url	Service API base URL.
add_table_data	When enabled, sample rows from the table are added to the prompt. The default value is "`True`".
table_data_cache_size	Size of the LRU cache used for storing table sample data.
auto_fetch_table_meta_limit	Number of tables to load from a database, initially.
token	Service API secret token.
token_script	Provides a secure way to get the service API secret token.

Specify the semantic search-related configurations used for RAG under the [[semantic_search]] section, as listed in the following table:

Semantic search config key	Description
relevancy	The technology you want to use for semantic search. Acceptable values are `vector_search` or `vector_db`.
embedding_model	The model you want to use for data-embedding. This must be compatible with SentenceTransformer.
cache_size	Size of the LRU cache used for storing embedding.

How to use the SQL AI Assistant?

To launch the Hue AI assistant, click the blue dot on the Hue web interface interface as shown in the following image:

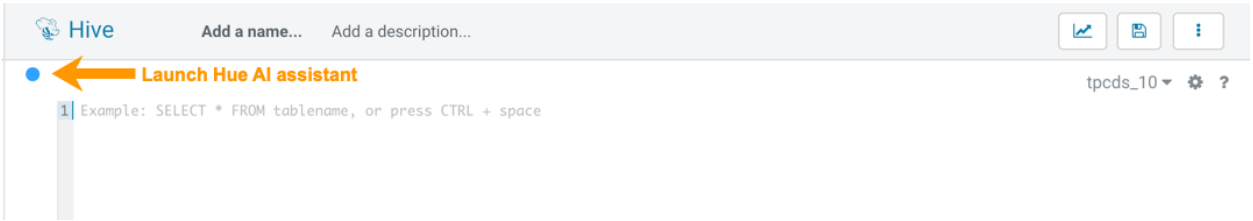

Generating SQL from NQL

Click **GENERATE** and type the query in natural language as shown in the following image:

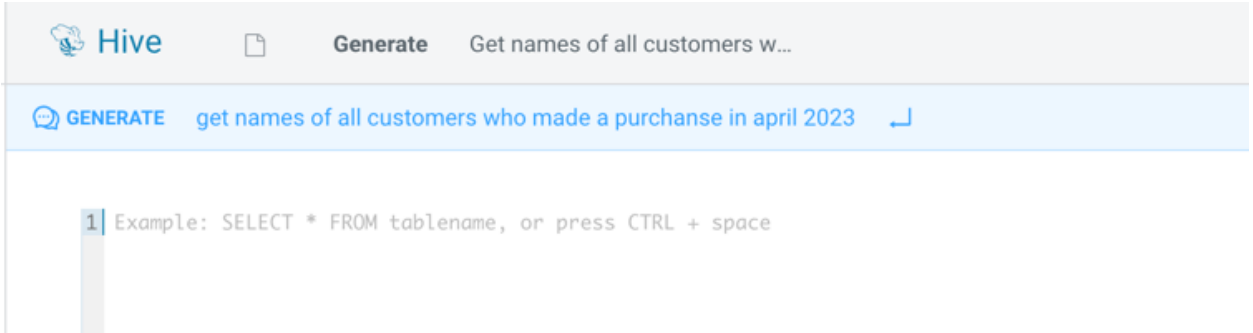

The SQL query is generated as shown in the following image. You can insert it into the editor by clicking **Insert** or you can copy the query into the editor.

Editing the query in natural language

Click **Edit** as shown in the following image:

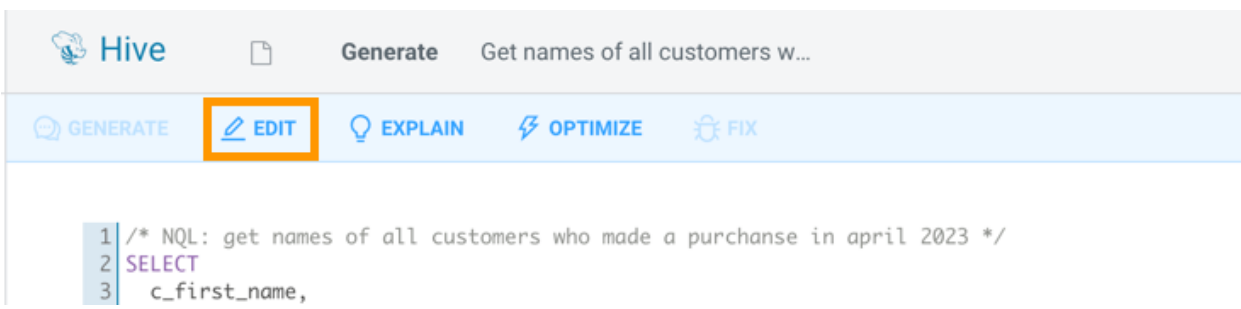

Make the required changes and press enter:

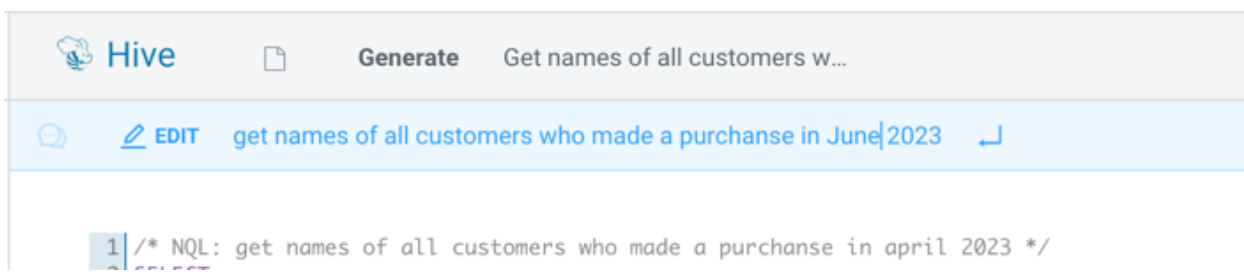

Getting an explanation of a SQL query in natural language

To understand complex queries in natural language, click **EXPLAIN** as shown in the following image:

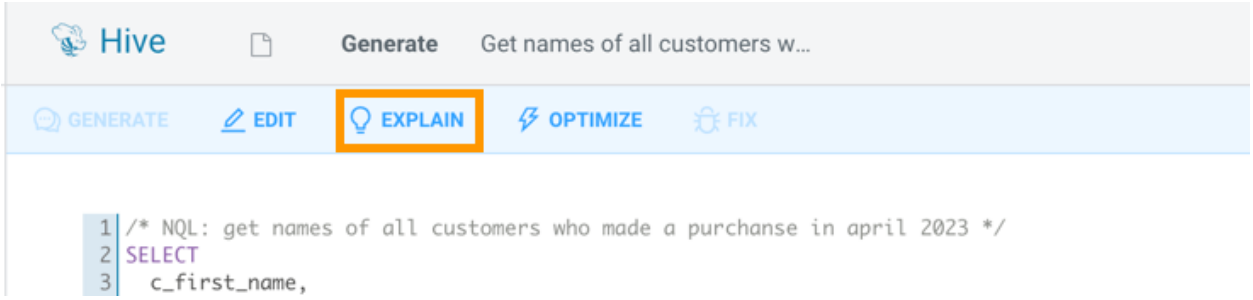

The LLM generates the explanation of the SQL query as shown in the following image:

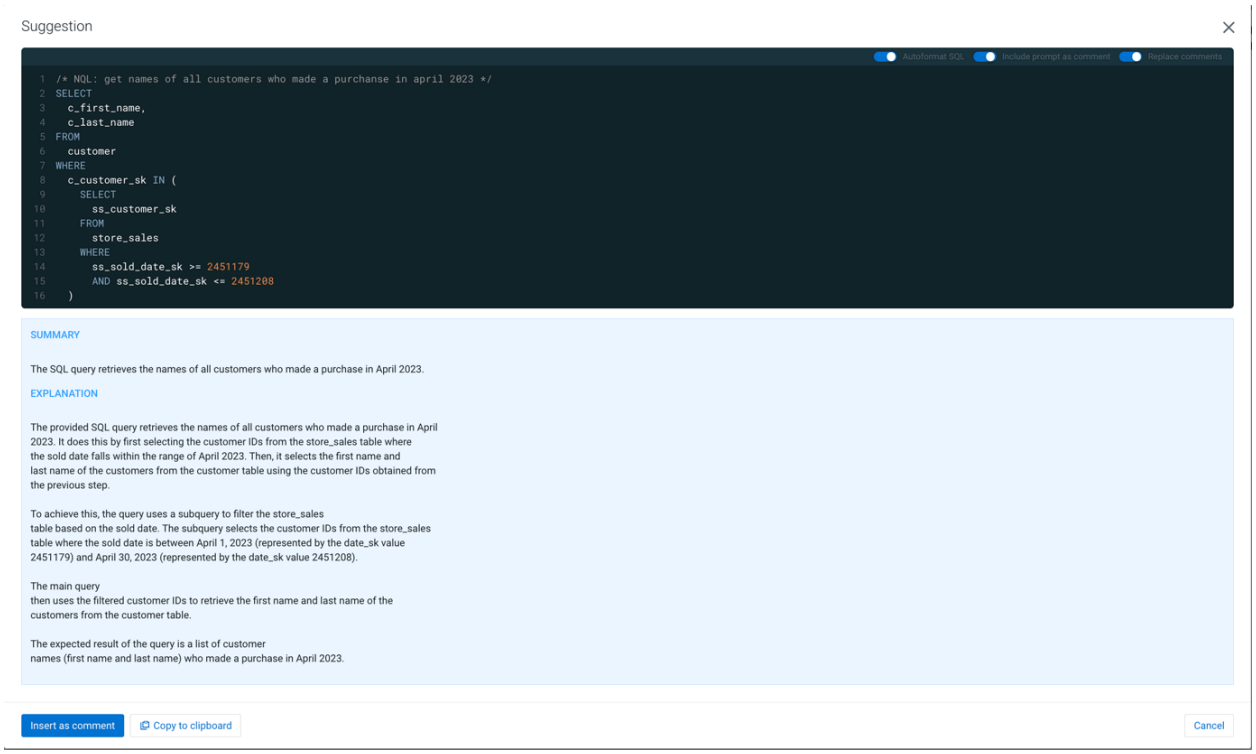

Optimizing a query

Click **OPTIMIZE** to improve an existing query as shown in the following image:

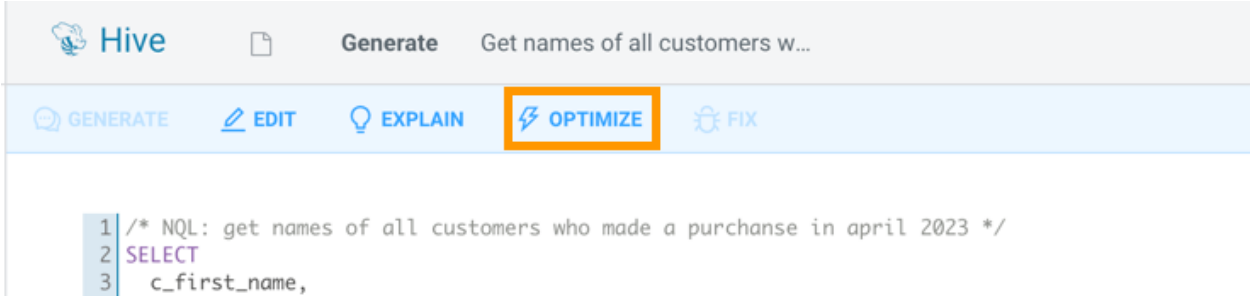

The following example shows how Hue identifies issues, optimizes a query, and provides an explanation:

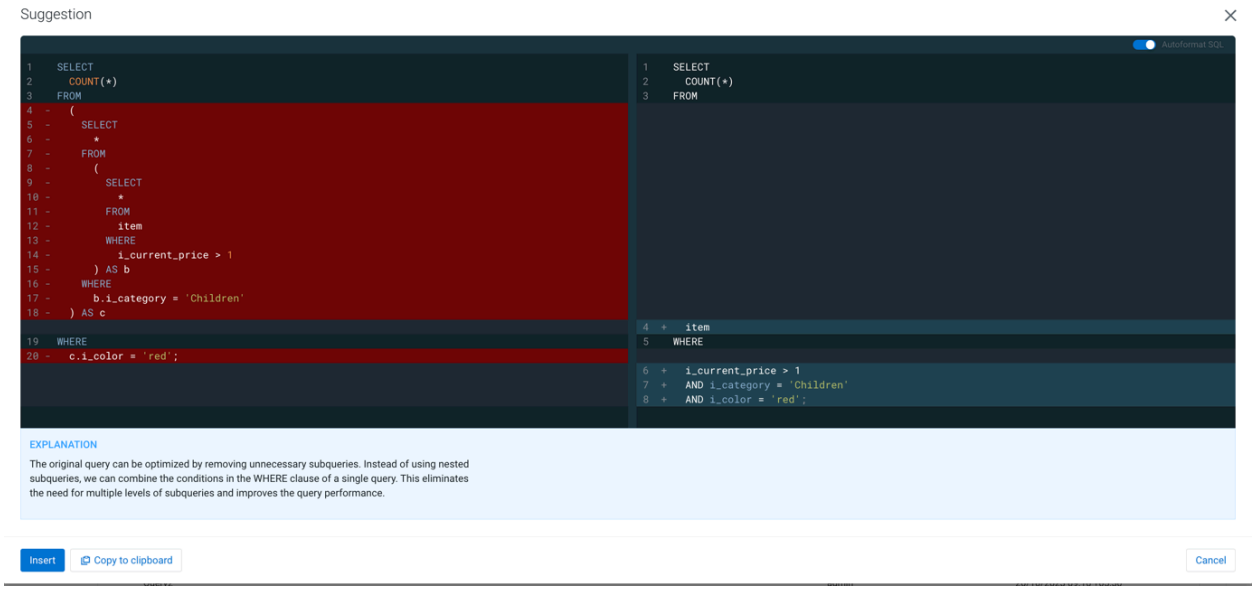

Fixing a SQL query

Click **FIX** to correct any syntax errors present in your SQL statement causing the query to fail as shown in the following image:

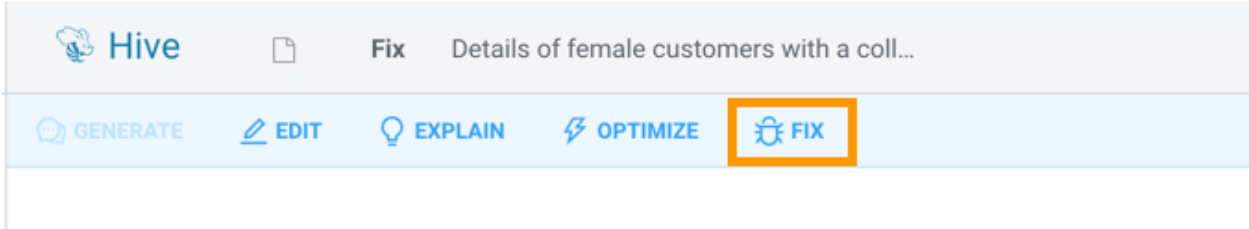

For example:

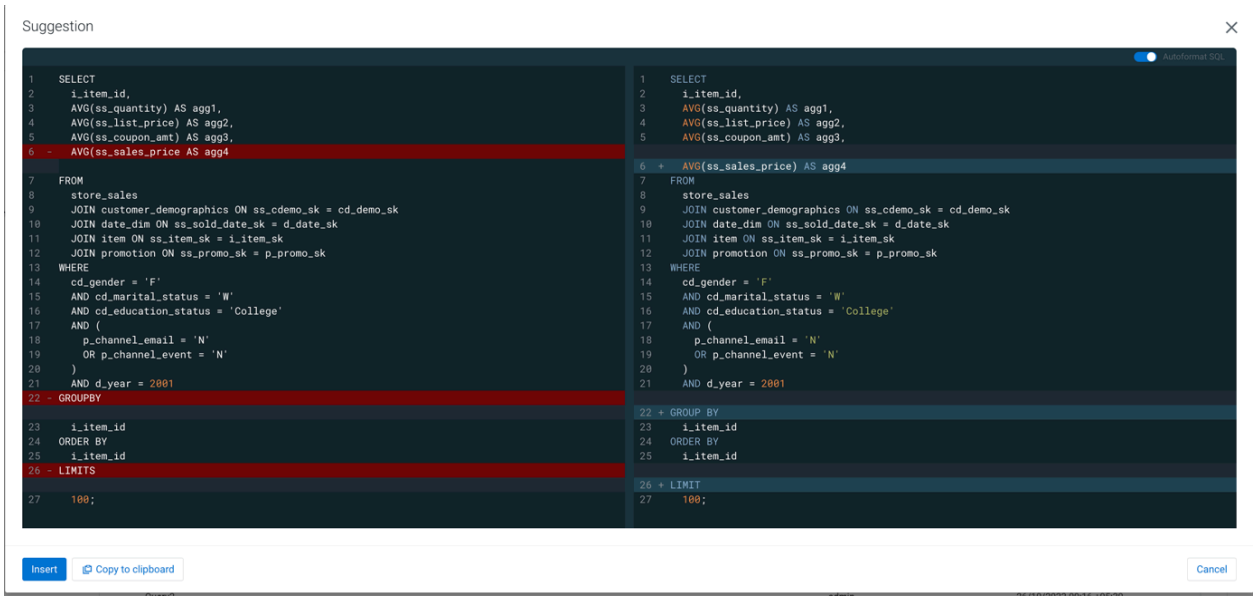

Limitations

- **Non-deterministic**: LLMs are non-deterministic, which means you cannot guarantee the same output for the same input every time, and it can lead to different responses to **similar** queries.
- **Ambiguity**: LLMs may struggle to handle ambiguous queries or contexts. SQL queries often rely on specific and unambiguous language, but LLMs can misinterpret or generate ambiguous SQL queries, leading to incorrect results.
- **Hallucination:** In the context of LLMs, hallucination refers to a phenomenon where these models generate text or responses that are incorrect, nonsensical, or fabricated. Occasionally you might see incorrect identifiers or literals in the response.

Integrating CDP Data Catalog with AWS Glue Data Catalog (Preview)

Date published: 2021-08-09
Date modified: 2021-12-08

Legal Notice

© Cloudera Inc. 2021. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2

Contents 3

Integrating CDP Data Catalog with AWS Glue Data Catalog 4

How integration works 4

Setting up AWS Glue Catalog with CDP Data Catalog 4

Working with AWS Glue in CDP Data Catalog 9

Accessing AWS Glue in Data Catalog 9

Listing the Glue assets 10

Unsupported features 13

Known Issues 13

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Integrating CDP Data Catalog with AWS Glue Data Catalog

Integrating CDP Data Catalog with AWS Glue Catalog enables the users to browse and discover data as well as register data into SDX (via metadata translation or copy), so that it can be used with Data Hubs and other relevant experiences. While using AWS Glue in Data Catalog, you will be able to experience a complete snapshot metadata view, along with other visible attributes that can power your data governance capabilities.

How integration works

Assuming that the SDX is running in the users' AWS account (that contains the same AWS account which has GlueDataCatalog and the data that has to be discovered), the credentials with the ExternalDataDiscoveryService (which is hosted in SDX) must be shared, so that these two entities can interact with each other. These credentials are used to launch SDX and other workload clusters on the users' AWS account.

Prerequisites:

- You must have full access to AWS Glue Catalog and also have access to the EMR cluster's Hive Metastore instance.
- You must set up the CDP. For more information, see Getting Started with CDP.
- You must have access to your AWS IT Admin and CDP Admin user credentials, which is required to enable CDP to access AWS/EMR managed data in CDP.

Note: AWS policies are managed by AWS IAM and the AWS roles are added to the CDP Management Console. Refer to Using Instance Profile in AWS and Using credentials in Management Console. For more information about the AWS access, see AWS Environments.

Setting up AWS Glue Catalog with CDP Data Catalog

Follow these set up tasks to map your CDP Data Catalog instance with AWS Glue Catalog:

1. Enable the entitlement for your Data Catalog instance by running the following command on your CDP environment:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

For example: *$ cdp coreadmin grant-entitlement --entitlement-name DATA_CATALOG_ENABLE_AWS_GLUE --account-id {account_id}*

2. You must add relevant permissions in the corresponding AWS account:

    a. Include permission to access Glue Catalog service by editing the policy accordingly.

Make a note of the **Assumer Instance Profile** role that you intend to use and include full access authorization for AWS Glue.

Assumer Instance profile role is the one which you select in the available public cloud documentation listed under **step 10**. For more information, see [Register a CDP environment].

Use the following screenshots as a reference to complete the set up.

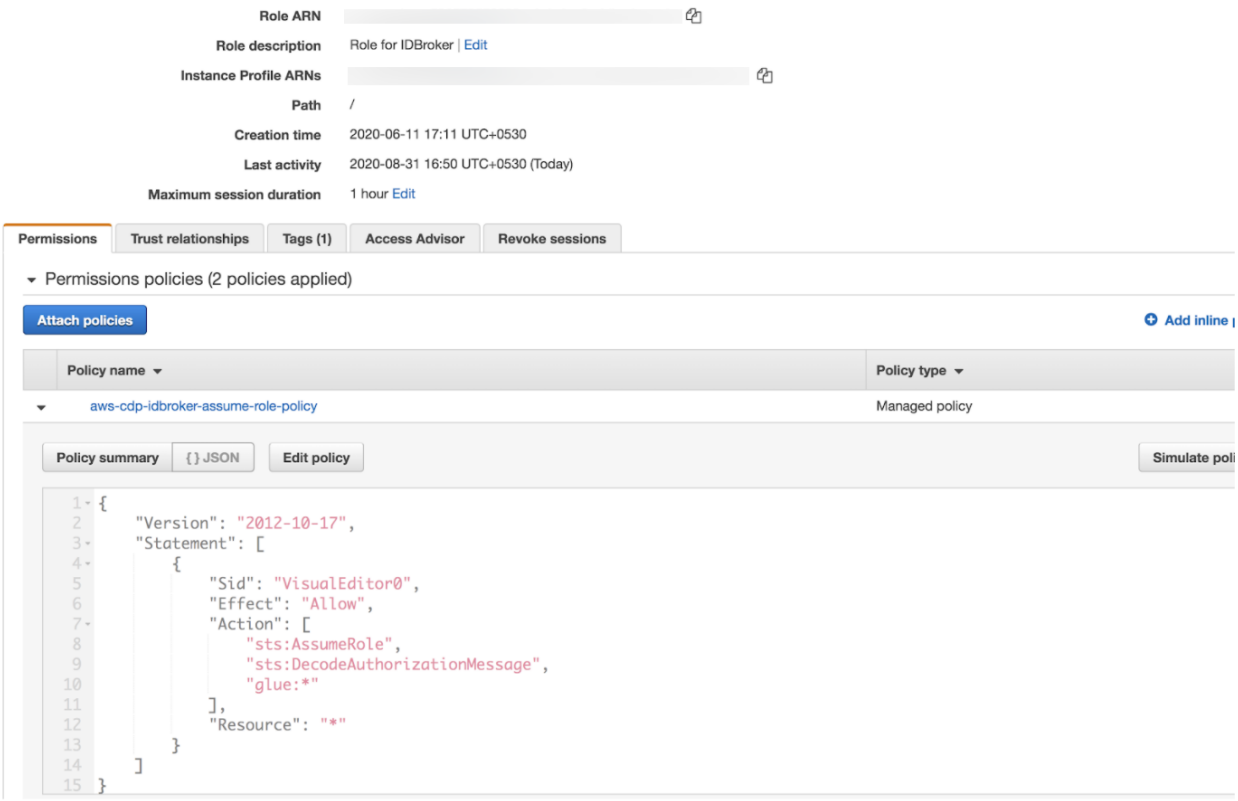

Note: For **Role ARN** and **Instance Profile ARNs**, you must include the appropriate account number and role respectively.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

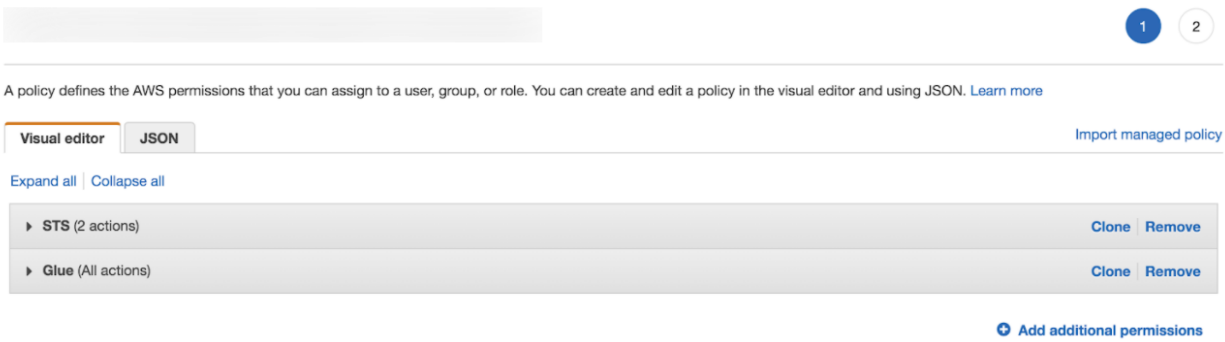

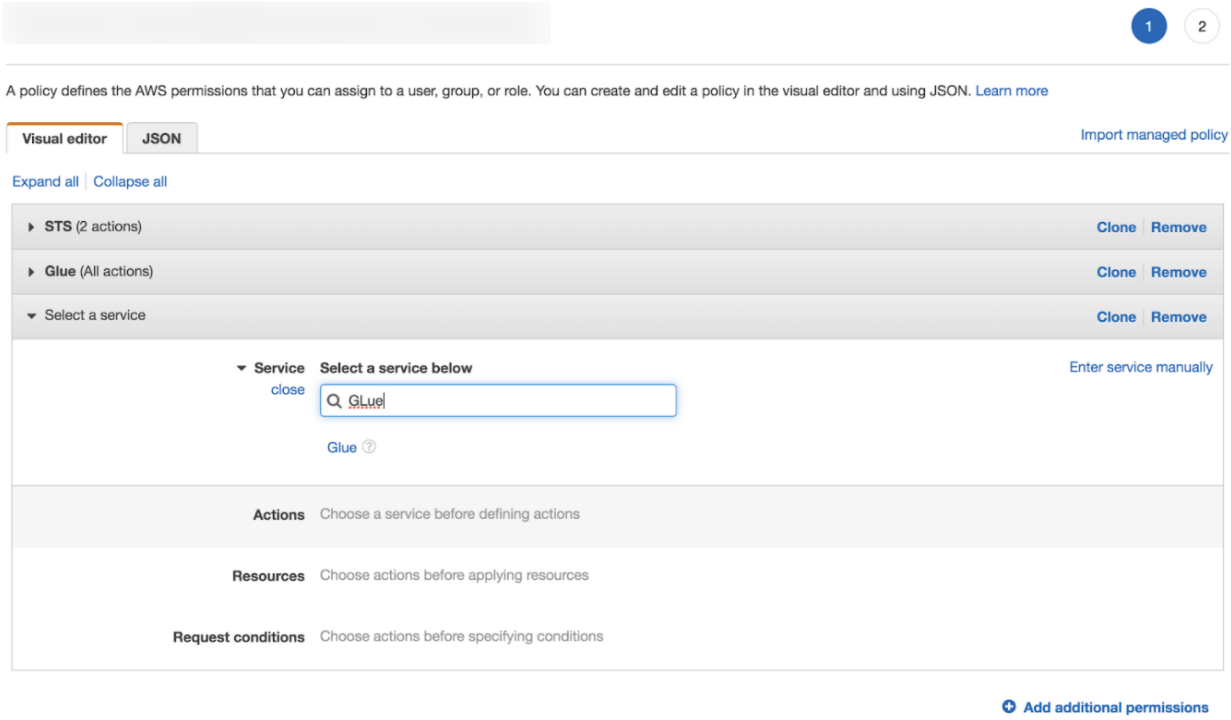

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

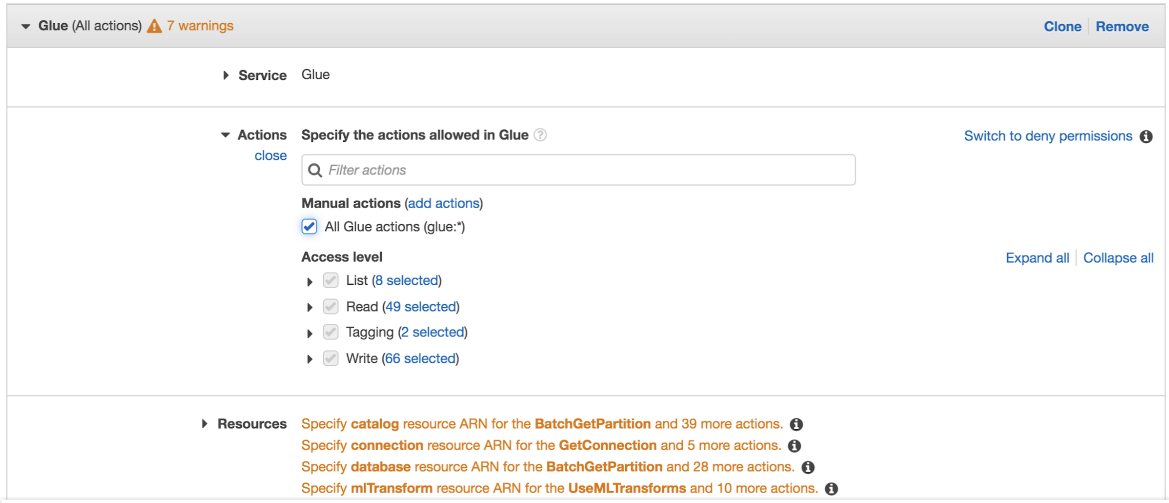

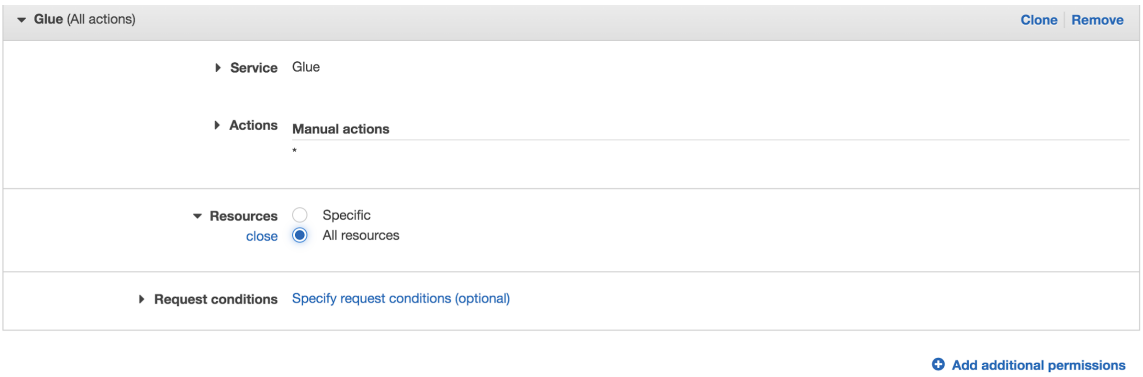

b. Search for the role attached to the **Instance Profile** of the CDP environment.

Use the **Instance Profile** that you have configured above with Glue related policy in your AWS Environment creation command.

Use the following examples to setup AWS environment and AWS data lake as part of the Glue setup:

```
cdp environments create-aws-environment --profile default --cli-input-json '
{"environmentName":"ab-ds-cli-7321",
 "credentialName":"cd2d-1234",
 "Region":"us-region-2",
            "securityAccess":{-insert the value--"},
 "Authentication":{---insert the value---"},

"logStorage":{"storageLocationBase":"s3a://demo-e2e-test-state-bucket/ab-ds-cli-7321/logs","instanceProfile":"arn:aws:iam::<xxxxxxxxxxx>:instance-profile/<role-name>"},
 "vpcId":"vpc-0123456",
 "subnetIds":["subnet-04fe923b902aa5cf2","subnet-099c7a631f0ebed3c"],
 "s3GuardTableName":"dc-pro-cli-7210",
 "Description":"ab-ds-cli-7321",
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
 "enableTunnel":false,
 "workloadAnalytics":false,
 "freeIpa":{"instanceCountByGroup":1},
 }'

cdp environments set-id-broker-mappings \
--environment-name "ab-ds-cli-7321" \
--profile default \
--set-empty-mappings \
--data-access-role arn:aws:iam::<xxxxxxxxxxxx>:role/add-role \
--ranger-audit-role arn:aws:iam::<xxxxxxxxxxxx>:role/add-role
```

Similarly, while setting up the data lake use the **Instance Profile** that you configured above with Glue related policy in your data lake creation command:

```
cdp datalake create-aws-datalake --profile default --runtime 7.2.12 --cli-input-json '
{"datalakeName":"ab-ds-cli-7321-sdx",
 "environmentName":"ab-ds-cli-7321",

"cloudProviderConfiguration":{"instanceProfile":"arn:aws:iam::<xxxxxxxxxxx>:instance-profile/<role-name>","storageBucketLocation":"s3a://demo-e2e-test-state-bucket/ab-ds-cli-7321"},
 "scale":"LIGHT_DUTY",
 }'
```

For more information, see [Creating an AWS environment with a medium duty data lake using the CLI](#).

c. Navigate to the attached policy for the role.
d. When you manually create tables in AWS Glue Data Catalog, you must set the **fully qualified path** for the table location.

For example:
“`s3://manowar-e2e-test-state-bucket.s3-us-west-2.amazonaws.com/dc-pro-721-storage/glue/`”

3. You must set up the AWS Glue Data Catalog. For more information, see [Populating the Glue Data Catalog](#). You must select only the CSV format which is currently supported for CDP Data Catalog and the delimiter which is used in the data.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

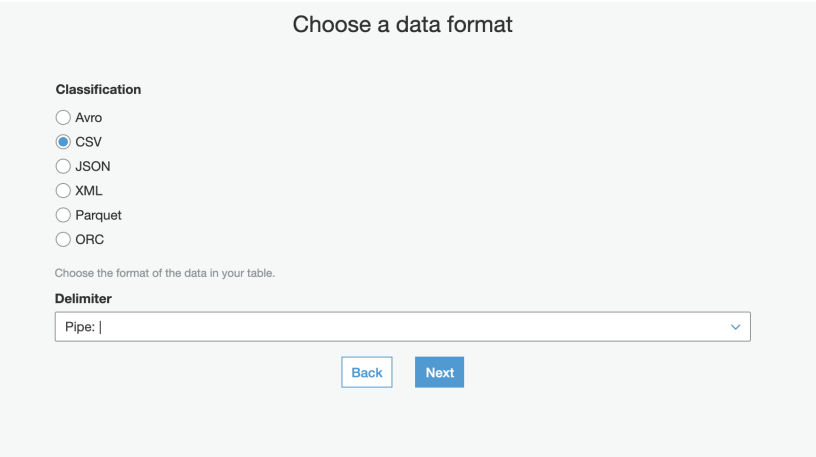

4. While creating tables in AWS Glue Data Catalog manually, set the fully qualified path for location. For example: ("s3://manowar-e2e-test-state-bucket.s3-us-west-2.amazonaws.com/dc-pro-721-storage/glue/")

Working with AWS Glue in CDP Data Catalog

The AWS Glue metadata must be registered with the CDP Data Catalog. The Glue contains the metadata that is synchronized with Data Catalog. The Glue metadata is accessed using the Data lake option in the Data Catalog service. After setting up AWS Glue with CDP Data Catalog and once the Glue synchronization with CDP is complete, the AWS Glue data lake appears in the Data Catalog instance.

Note: For each AWS environment, there would be a separate listing under the data lake option. The Glue data lake name / identifier in Data Catalog follows the format: <**glue**: **Data Lake Name**>. The Glue assets are of the type: **GLUE EXTERNAL TABLE**.

Accessing AWS Glue in Data Catalog

To access the Glue metadata in Data Catalog, you must note the following in your Data Catalog instance.

- List the Glue Metadata by selecting the Glue data lake.
- Select one or more Glue assets and register the same with CDP
- Verify if the registered Glue assets are listed in the Data Catalog owned data lake
- Select the registered Glue asset and click to open the **Asset Details** page

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Listing the Glue assets

In Data Catalog, when you select the AWS Glue data lake, you can view the list of Glue assets. These metadata assets are directly sourced from Glue.

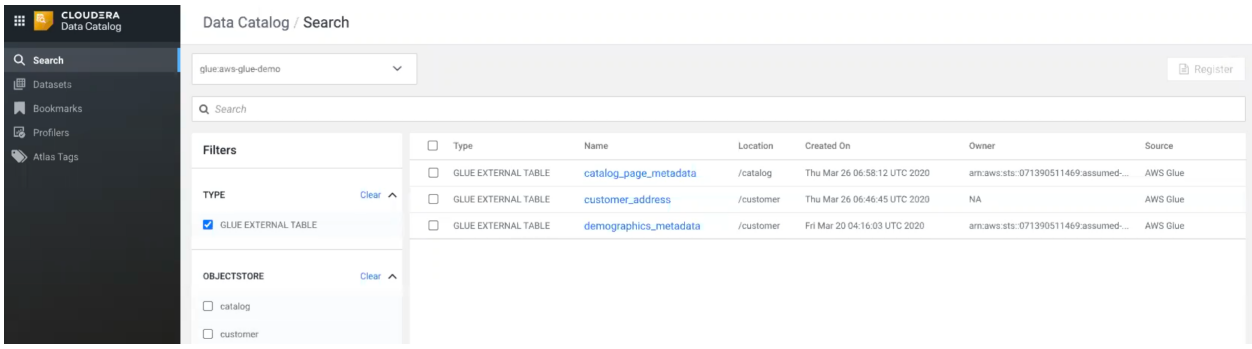

When you click on one of the assets, the Asset Details page is displayed.

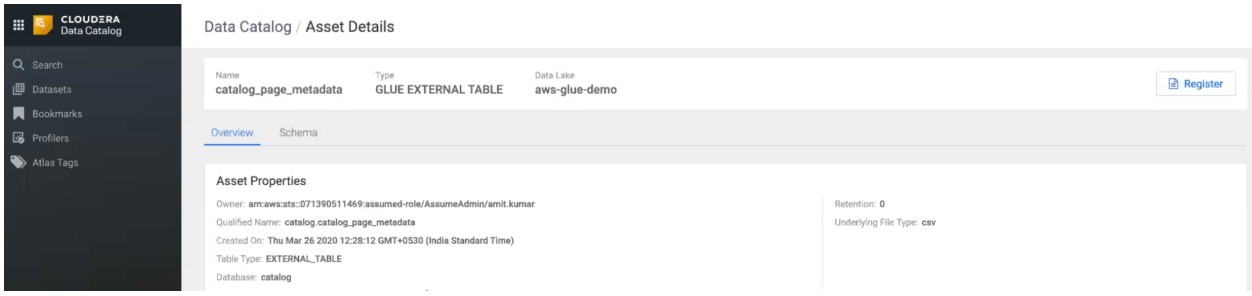

Next, on the main Data Catalog page, you must select the Glue data lake and select one of the Glue assets and register the asset to CDP. Click **Register**.

Optionally, you directly click on the Glue asset and register the asset on the **Asset Details** page.

Note: You can select one or more Glue assets on the asset listing page to register them in CDP.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

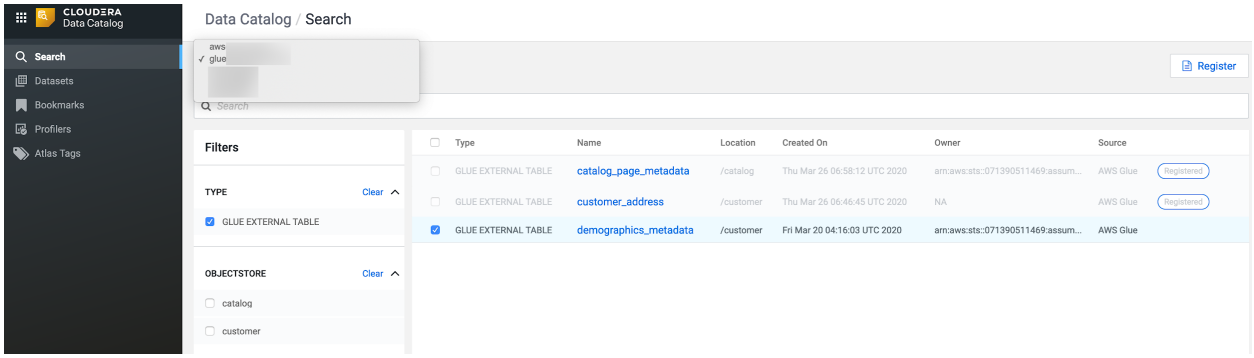

Once the Glue asset is registered, the asset is imported into CDP.

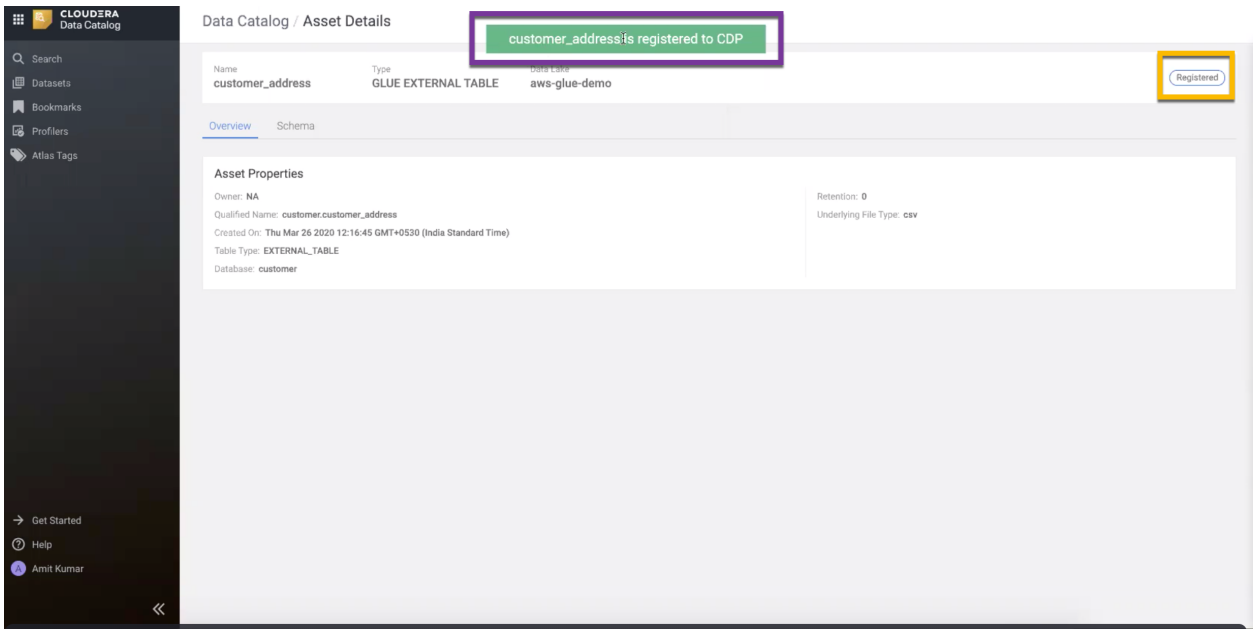

Next, navigate back to the Data Catalog main page and select the Data Catalog owned data lake and select the type as **Hive Table**. The search results lists all the Hive table assets and you can view the Glue registered asset(s) as well. The registered Glue asset can be identified using the 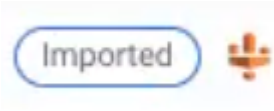

icon.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

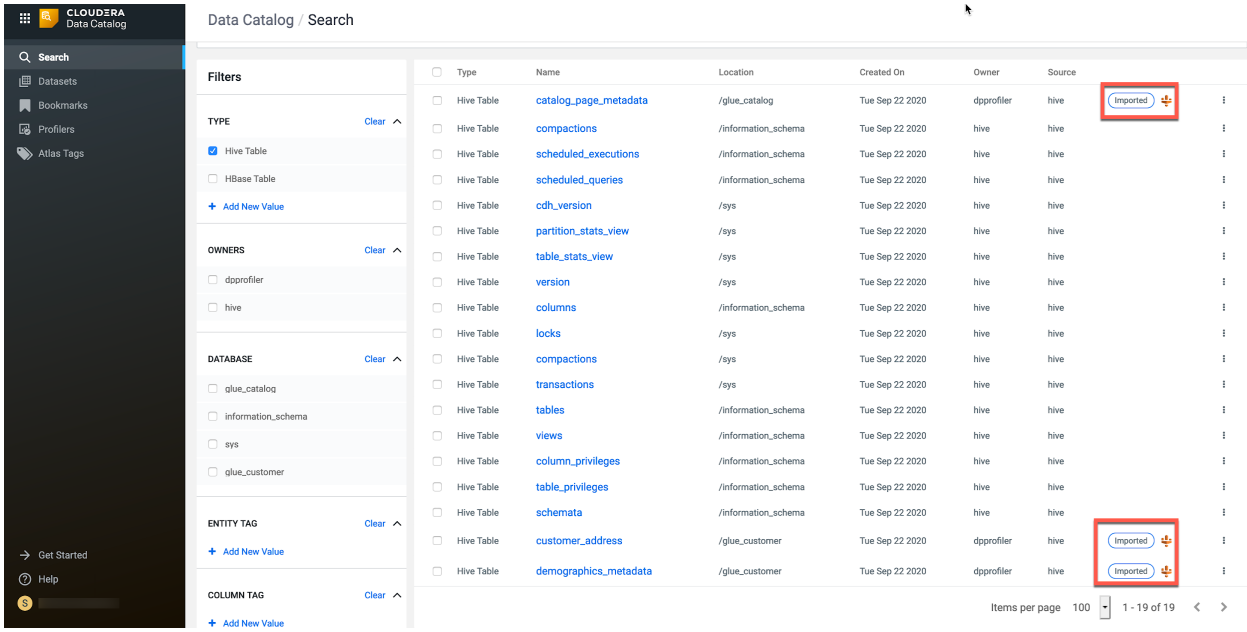

Note: The entries in the Data Catalog for Glue assets are created in the Hive Metastore.

Click on the Glue registered asset and you can view the **Asset Details** page for the selected Glue asset.

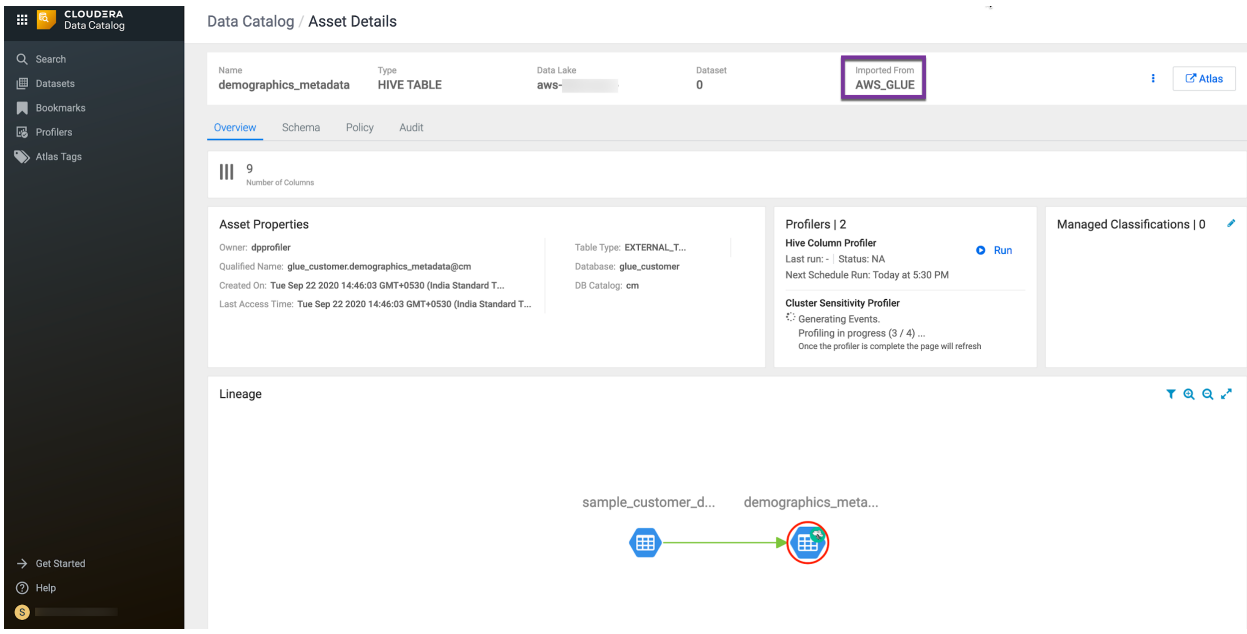

The Asset Details page for the Glue asset is populated by Atlas. While registering the Glue data, the data was written to the Hive Metastore and later Atlas synchronised the metadata.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Go back to the main **Data Catalog** page and select the Glue data lake. Note that the registered Glue asset(s) are greyed out or cannot be selected again.

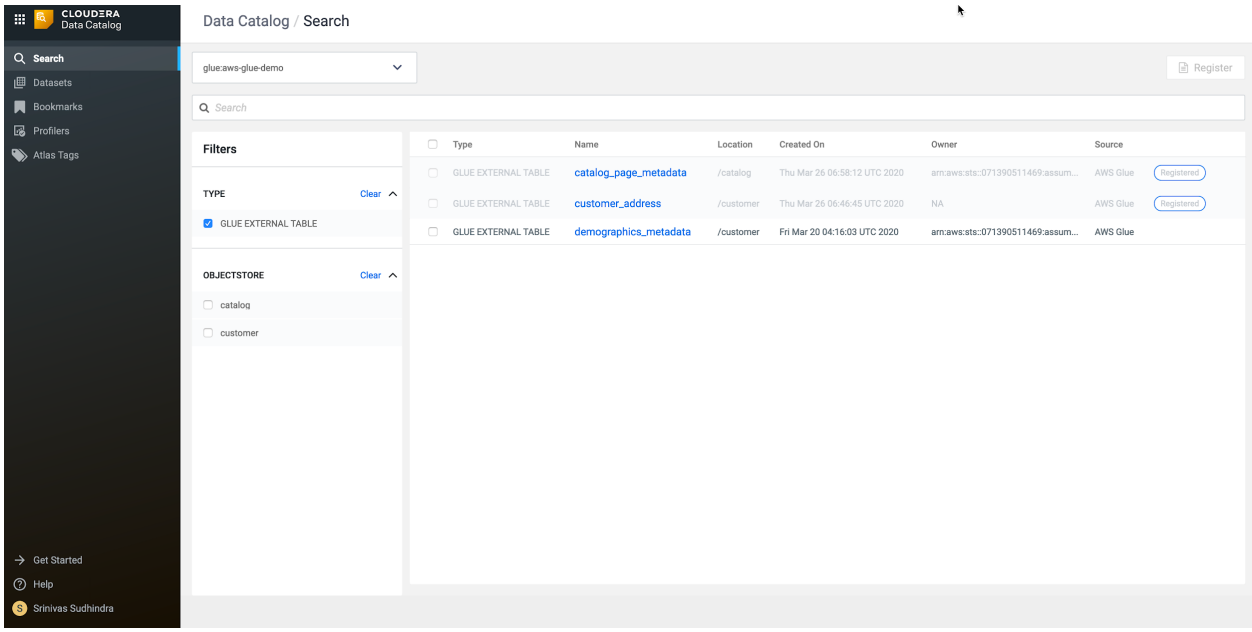

You can still view the registered Glue assets (powered by Atlas) by clicking on the same and it navigates to the **Asset Details** page as seen above in the image.

Working with Ranger Authorization Service (RAZ) enabled AWS environment

For RAZ enabled AWS environment, you must employ the following permission settings to work with Data Catalog - Glue Integration.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

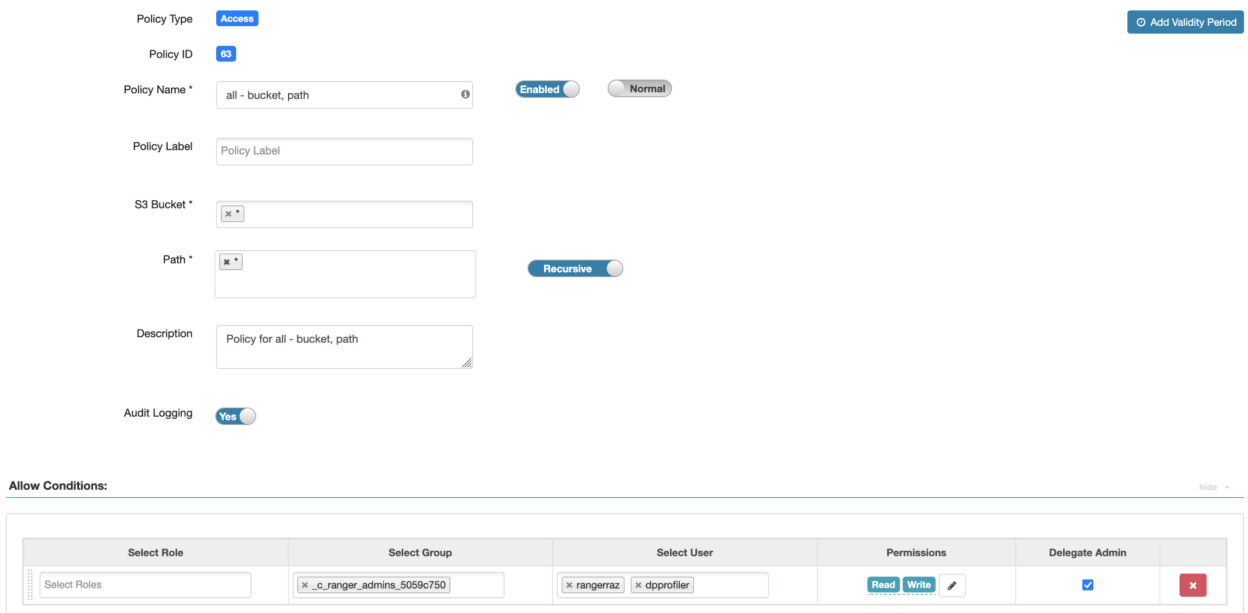

Note: By default "**dpprofiler**" user is not included in the allowed users list. You must manually add the "**dpprofiler**" user in the allowed users list.

Unsupported features

For a Glue data lake, while searching for a specific asset, multiple calls fail with "**400 Bad Request**" which is reflected on the Data Catalog UI in search suggestions. Search for Glue assets in Data Catalog is currently not supported.

Known Issues

CDPDSS-1179: Any Glue registered asset does not fully display the respective Asset Details page when accessed from the Glue Data Lake.

Problem: When any asset in the Data Catalog is Glue registered, it does not display any information in the Asset Details page, when viewed from the Glue Data Lake.

Workaround: Select the same asset in Data Catalog, which is not Glue Data Lake and click on the same to view the Asset Details information.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Private Cluster Support (Preview)

Date published: 2022-01-06
Date modified: 2023-08-02

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Private Cluster Support 4

Requirements 4

Enable the Private Cluster 4

Notes 5

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Private Cluster Support

Each type of network architecture supported by CDP has a unique set of tradeoffs among ease of setup, security, workloads (Experiences) supported, and so on.

Private Clusters, which are available on both AWS (GA) and Azure (Preview), provide a simple way to create a secure cluster, where the API server and the workloads themselves only rely on private IP addresses that are not accessible from the internet. Connectivity to the cluster from the CDP control place is provided by the Cluster Connectivity Manager v2 (CCM v2). CCMv2 uses an agent running in the cluster, and an inverting proxy running on CDP, which creates a HTTPS tunnel between the workload and the control plane.

Requirements

Ensure the following entitlements are needed to enable provisioning of private Kubernetes clusters. Customers should file SRE Jira tickets to request these entitlements, if necessary.

- CDP_CCM_V2_JUMPGATE
- CCMV2 JUMPGATE
- ML_ENABLE_PRIVATE_CLUSTER

Enable the Private Cluster

To enable a private cluster, select the option when provisioning the workspace.

1. In **ML Workspaces**, select **Provision Workspace.**
2. Enter a **Workspace Name**, and select **Environment**.
3. Select the **Advanced Options** toggle.
4. In **Network Settings**, select **Enable Fully Private Cluster**.
5. Make any other settings needed, and select **Provision Workspace**.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

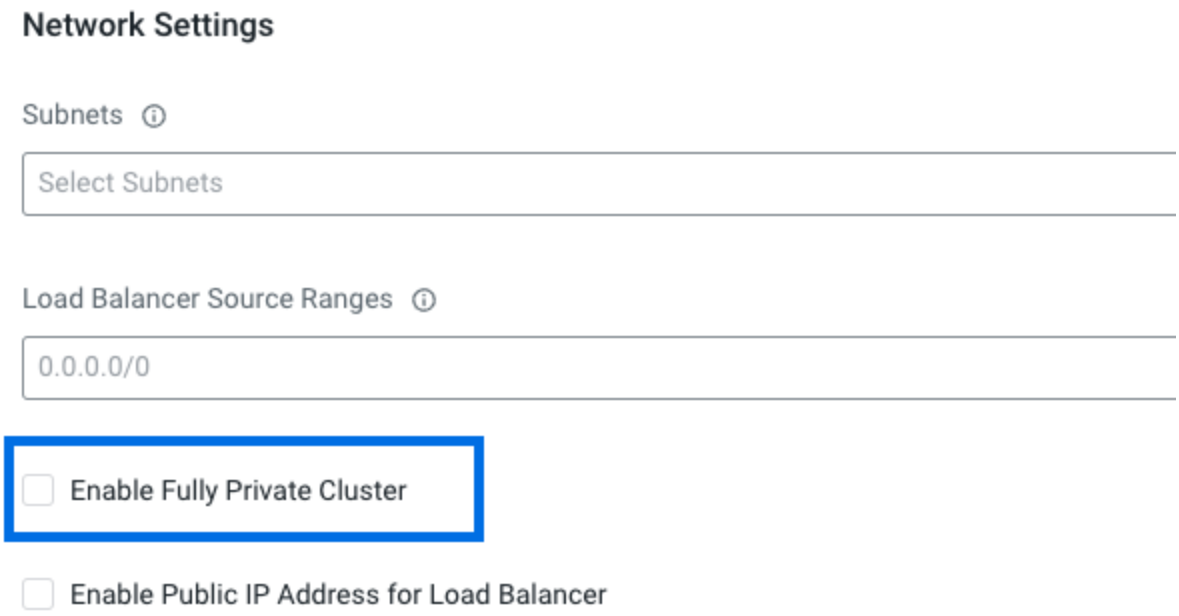

The workspace is provisioned using a fully private cluster.

User Defined Routing (Azure only)

With the Fully Private Cluster configuration, Azure still creates some public IP resources to support load balancer egress. If necessary, you can avoid creating public IP addresses in the CML cluster by using a User Defined Routing (UDR) table. A UDR table can be configured in the cluster subnet to route packets to a customer-configured firewall, for example to limit internet access or analyze traffic. For more information on setting up UDR, see the Microsoft articles Virtual appliance scenario or Virtual network traffic routing.

To utilize a UDR and firewall in the Azure CML private cluster, select the following when setting up the cluster.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

1. Select a subnet with a default route configuration to forward the traffic to the network appliance or firewall.

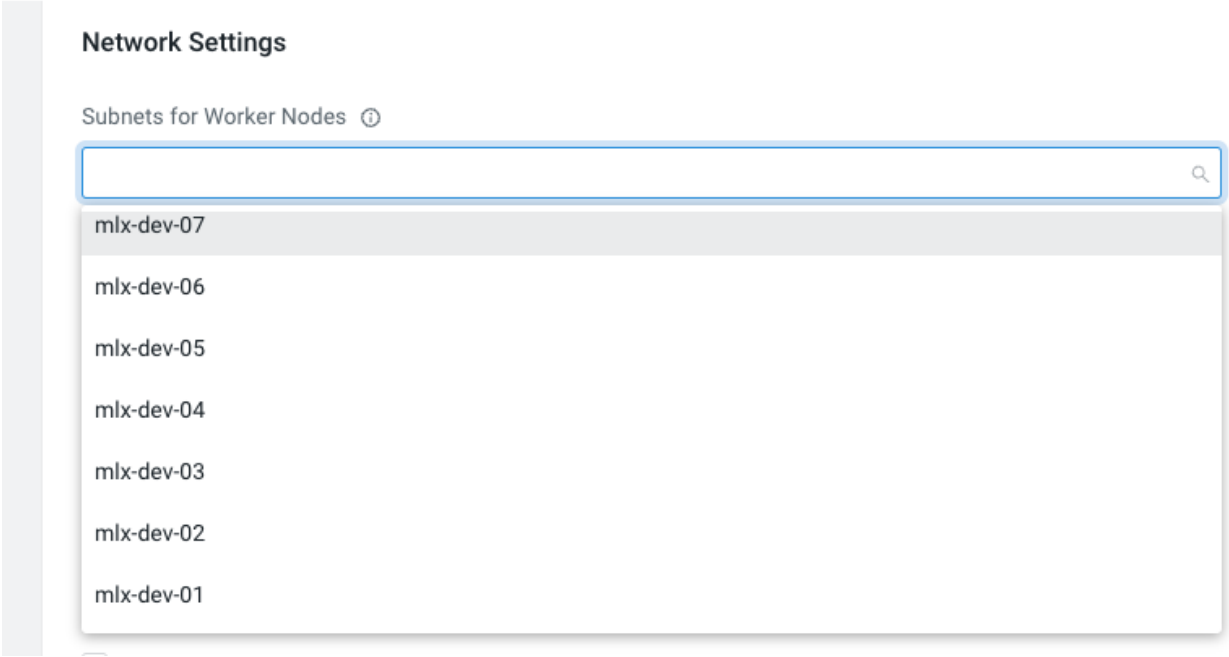

2. Create load balancers with private IP addresses. This is the default choice when creating clusters in CML.

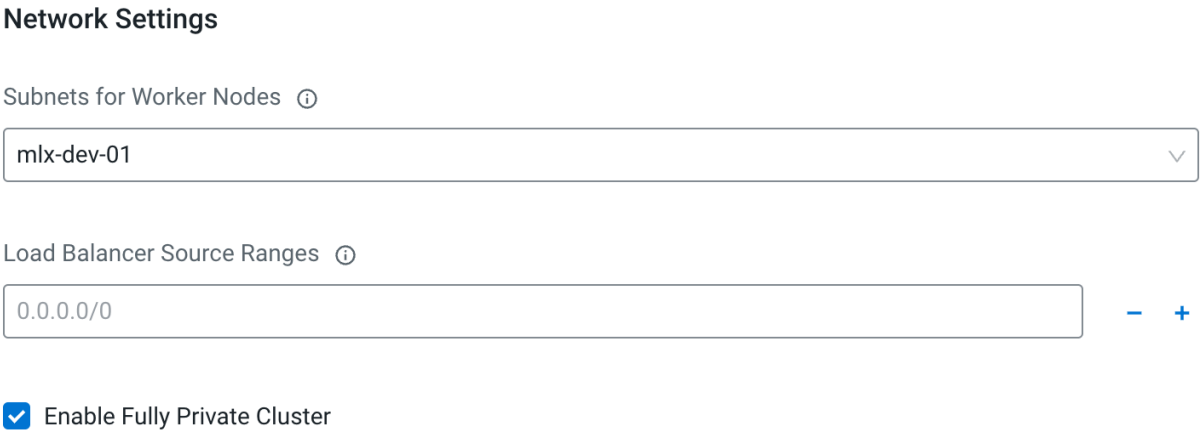

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

3. Select **Enable User Defined Routing**.

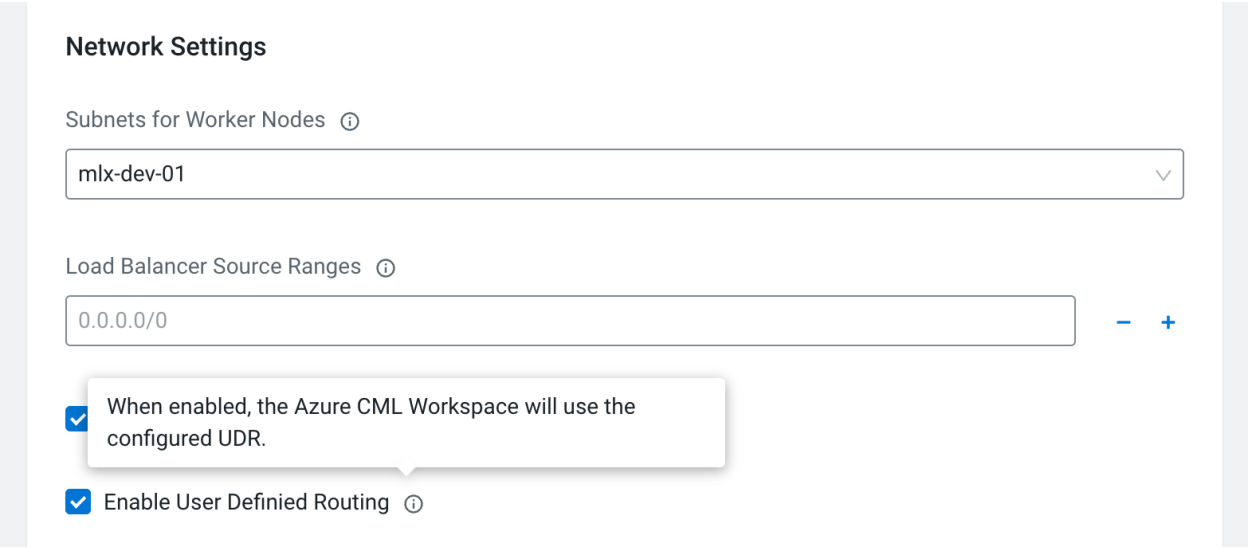

Limitations of private cluster configuration in Azure

There are several limitations to note:

- **While using a Private cluster**, you must disable the network acceleration in the freeIPA VM. In the Azure portal, go to **Virtual machines**, and identify the freeIPA VM (usually, <environment name>-freeipa<random string>). In **Settings** > **Networking**, open the network interface, select **Edit accelerated networking**, and disable it.
- There is a limitation in Azure (tracking# 2212090040005952) where more than 15 tags fail private-DNS-Zone provisioning. Because of this limitation, CML won't add the following tags to the cluster during cluster provisioning. If necessary, you can add them manually to the resources in the managed cluster.
    - WorkspaceCrn
    - TenantID
    - Any custom tags that are configured during the CML cluster provision.
- If a customer script uses **tenantID**, it needs to be derived from one of the CRN tags.
- Instead of workspace CRN, the workspace name tag needs to be used.
- No tenant tags can be configured in the environment service.
- If the cluster creation fails due to TCP issues, customers should delete the cluster, clean up the cloud resources and retry the cluster creation.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Creating a Model Registry on an Azure UDR Private Cluster

Use the following template CDP CLI command to create a UDR private cluster on Azure with a Model Registry. You must replace the following template items with your own information.

- <environment CRN>
- <environment name> (in two places)
- <existing NFS name>
- <subnet>

For more information on Model Registry, see the Preview Feature document.

If you have not yet downloaded the CDP CLI tool, see the documentation.

The required CDP CLI version is version 0.9.93 or higher.

CDP CLI command

This CDP CLI command has three key sections:

1. Enables support for private clusters in Azure ("privateCluster": true,)
2. Enables UDR for the private cluster ("outboundTypes": ["OUTBOUND_TYPE_UDR"],)
3. Specifies the subnet for the UDR-enabled private cluster ("subnets")

```
cdp ml create-model-registry --cli-input-json '{
  "environmentCrn": "<environment CRN>",
  "environmentName": "<environment name>",
  "createWorkspacePayload": {
     "environmentName": "<environment name>",
     "workspaceName": "modelregistry",
     "privateCluster": true, # This setting enables the support for private cluster in azure.
     "outboundTypes": ["OUTBOUND_TYPE_UDR"], # Required for enabling UDR.
     "skipValidation": true,
     "disableTLS": false,
     "disableSSO": false,
     "existingNFS": "<existing NFS name>",
     "nfsVersion": "3",
     "xEntitlements": [
        "ML_MODEL_REGISTRY",
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
        "ML_ENABLE_PRIVATE_CLUSTER"
      ],
    "provisionK8sRequest": {
      "environmentName": "<environment name>",
      "network": {
        "topology": {
          "subnets": [
            "<subnet>" # subnet with a default route configuration to forward the traffic to the
network
appliance or firewall. This is required to enable UDR.
            ]
        }
      },
      "instanceGroups": [{
        "autoscaling": {
            "minInstances": 1,
            "maxInstances": 5
          },
          "instanceType": "Standard_DS5_v2",
          "rootVolume": {
            "size": 256
          }
        }
        ]
    }
  }
}' --profile eu-stage
```

Notes

- In a private cluster configuration, the Kubernetes API server has a private IP address, and it is not routable from the internet, thus reducing the attack surface for the cluster. If a customer wants to have access to the cluster, they need to peer their on-premises network to the CDP environment through VPN peering, for example.
- In a private cluster setup, Azure may still create public IP resources for egress. To avoid having any public IP addresses, enable User Defined Routing (UDR).
- An alternative configuration that is supported is to provide a non-transparent proxy server for the private cluster (for AWS only). For more information, see https://docs.cloudera.com/machine-learning/cloud/requirements-aws/topics/ml-non-transparent-proxy-aws.html

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Retry Install Workspace (Technical Preview)

Date published: 2023-04-26
Date modified: 2023-07-25

Legal Notice

© Cloudera Inc. 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Retry Install Workspace **4**
Description **4**
Prerequisites **4**
CDP Entitlement 4
AWS CDP Cross-Account Role Permissions 5
Retrying CML Create Workspace **5**
Using CDPCLI 5
Using UI 5
Footnotes **6**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Retry Install Workspace

Tech Preview Documentation

Description

Creating a workspace in CML involves a series of complex steps that can take up to 30-40 minutes to complete. These steps include provisioning the cluster, setting up worker nodes, and installing the required helm charts for running CML applications. During this process, there are multiple underlying activities that must be performed, and each of these activities can take several minutes to complete.

Workspace creation is not always a smooth process. Users may encounter various issues such as IAM permission problems or intermittent networking issues, which can cause the workspace creation to fail at any stage. In such cases, users are forced to delete the failed workspace and create a new one, wasting precious time and resources.

However, with the introduction of this new feature, users can now retry the installation of a failed workspace creation process from the exact point where it failed. This means that users no longer need to delete the cluster and try to provision a new workspace again. Instead, they can retry the installation, and the system will pick up from where it left off, saving valuable time and resources.

This new feature can significantly improve the user experience when creating workspaces in CML. Users can now create workspaces more efficiently, with fewer interruptions and with a greater level of confidence that the process will be successful.

We can initiate a retry of the Create Workspace workflow by simply making a call to the same API endpoint used for the original CreateWorkspace request, with the identical payload as before, including the same workspace name and environment name. This will allow the system to resume the installation process from the exact point where it was interrupted, without any need to modify or adjust the original request parameters.

Prerequisites

CDP Entitlement

The feature is gated behind the CREATE_WORKSPACE_V2 entitlement. This entitlement must be granted to your CDP account before trying to suspend your workspace.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

AWS CDP Cross-Account Role Permissions

None

Retrying CML Create Workspace

Using CDPCLI

If your workspace is in the 'Creation Failed', 'Validation Failed', or 'Installation Failed' state, you can retry installing the CML Workspace by using the same call that was used during your initial attempt.

Here is a sample CDPCLI `create-workspace` call:

```
$ cdp ml create-workspace --cli-input-json '{ "environmentName":
"environment_name","workspaceName": "workspace_name","provisionK8sRequest":
{"instanceGroups": [{"instanceType": "m5.4xlarge","rootVolume":{"size":
96},"autoscaling": {"minInstances": 1,"maxInstances": 5}},{"instanceType":
"p2.8xlarge","rootVolume": {"size": 96},"autoscaling": {"minInstances":
1,"maxInstances": 5}}],"environmentName": "environment_name","network": {"topology":
{}}},"enableMonitoring": true,"existingDatabaseConfig": {}, "kubernetesVersion":
"1.24","mlVersion": "2.0.36-b123"}' --profile cml_user
```

Note: When initiating a CDPCLI create workspace call again, the subsequent retry calls must contain the identical payload as that which was specified in the preceding calls.

Using UI

In case your workspace is in a state of 'Creation Failed', 'Validation Failed', or 'Installation Failed', you can attempt to reinstall the CML Workspace by following these steps:

1. Click on the 'Action Menu'.
2. Select the 'Retry Install Workspace' action menu item, which is illustrated in Figure 1.
3. The 'Retry Install Workspace' modal, as shown in Figure 2, should then open.
4. Click the 'OK' button to confirm, and the retry process of the 'Create Workspace Workflow' should begin.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

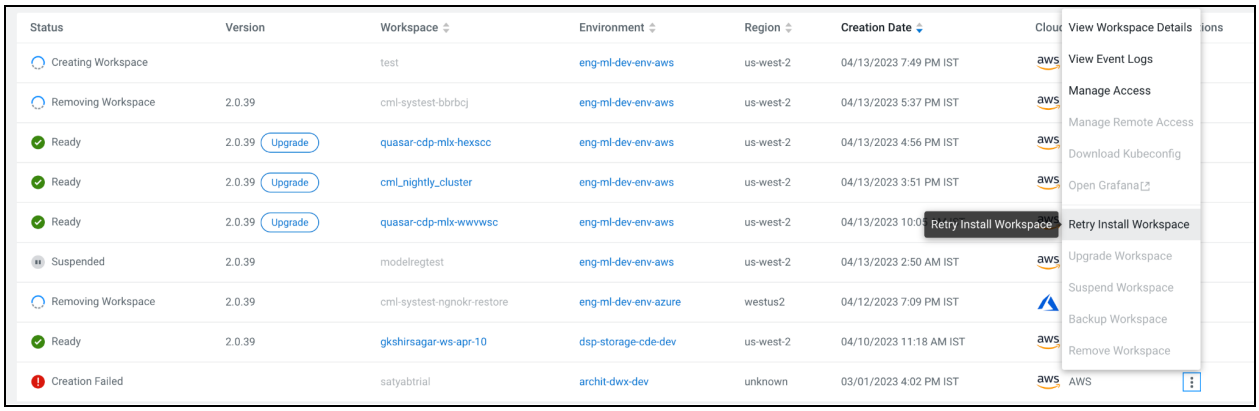

Figure 1: Retry Install Workspace Action Menu Item

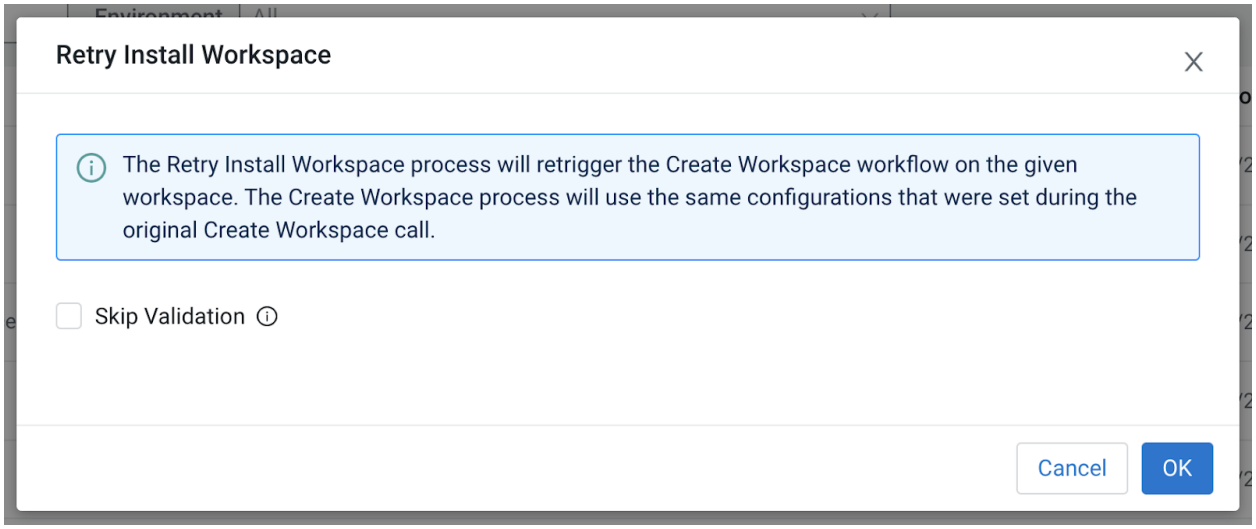

Figure 2: Retry Install Workspace Modal

Footnotes

1. While it's possible to attempt a failed installation or provisioning of a workspace multiple times, it's important to keep in mind that if you choose not to retry, or if the retry attempts fail, you should delete the cluster.
2. Sometimes, workspace creation may fail due to incorrect workspace configuration or persistent infrastructure issues. If this happens, retrying the installation process will not succeed, no matter how many times you try. In such cases, you have two options: delete the workspace and perform a fresh installation, or contact the Cloudera support team to investigate the issue.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Monitoring metrics in the Cloudera Operational Database with Grafana (Preview)

Date published: 2023-08-10
Date modified: 2024-11-12

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 3
Contents 4
Background 5
Enabling Grafana dashboard in COD 6
Enabling Grafana dashboard for an existing COD database 10
Importing a Grafana dashboard 17

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Background

Cloudera Operational Database (COD) provides a pre-defined solution to visualize the COD metrics comprehensively. COD uses Grafana to store and visualize the metrics. You can seamlessly access all the COD metrics using this Grafana solution.

Grafana deployment for COD aims to provide the following benefits.

- *Enables access restriction to Cloudera Manager*: You can obtain necessary information and perform the required operations through the COD API, CLI, or UI while maintaining appropriate access restrictions to Cloudera Manager for other operations.
- *Utilizes ready-made dashboards with advanced widgets*: COD utilizes the pre-built dashboards that Grafana offers, to visualize and monitor the performance of various components.
- *Integrates external metrics sources*: COD allows you to incorporate metrics from external sources such as S3 storage. This integration provides a comprehensive view of COD, which helps to understand COD performance problems more effectively.

Consider the following aspects while enabling Grafana for COD.

- *Grafana server upgrade*: COD focuses on utilizing the existing version to meet the requirements and the upgrade of the Grafana server is not within the scope of this solution.
- *Manual dashboard updates*: You must explicitly download the updated dashboards from the Cloudera Public repository. This allows flexibility and ensures that you have the latest dashboard versions whenever necessary.
- *Operating System support*: The existing solution works on Red Hat, CentOS, RHEL, and Fedora OS because of an RPM-based installation. That is why COD does not allow you to install Grafana with custom images having different OS.
- *Data Lake metrics support*: COD does not include the Data Lake metrics in this solution. Only individual COD (Data Hub) metrics appear in the Grafana dashboard.
- *HA Knox support*: Currently, COD only supports a single Gateway host with a single Knox Gateway.
- *Foursquare plugin support*: The Cloudera Manager foursquare datasource plugin does not support sending alerts. That is why alerts cannot be created on HBase, HDFS, and ZooKeeper dashboard panels.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Enabling Grafana dashboard in COD

Learn how to enable the Grafana URL to visualize the Cloudera Operational Database (COD) metrics.

Before you begin

- You must whitelist the Cloudera archive URL so that the necessary RPM packages for Grafana can be installed in the instances.
- You must whitelist the Cloudera repository so that the dashboards are created automatically.
- You must whitelist the Grafana RPM packages URL so that Grafana can be installed in the instances.
- You must whitelist the Google API storage URL so that the Cloudera Manager fousquare plugin can be installed.
- You must attach the following policy for CloudWatch plugin under cdp-infra2-logs-role (or the role with which the ec2 instances are created) to enable the Amazon S3 metrics inthe Grafana dashboard. To attach a policy under the **cdp-infra2-logs-rolee**, see Create a cross-account IAM role.

```
{

    "Version": "2012-10-17",


    "Statement": [


      {

        "Sid": "AllowReadingMetricsFromCloudWatch",

         "Effect": "Allow",

          "Action": [

              "cloudwatch:DescribeAlarmsForMetric",

              "cloudwatch:DescribeAlarmHistory",

                "cloudwatch:DescribeAlarms",

                "cloudwatch:ListMetrics",
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
            "cloudwatch:GetMetricStatistics",
            "cloudwatch:GetMetricData",
            "cloudwatch:GetInsightRuleReport"
        ],
      "Resource": "*"
},

{
      "Sid": "AllowReadingLogsFromCloudWatch",
       "Effect": "Allow",
        "Action": [
        "logs:DescribeLogGroups",
        "logs:GetLogGroupFields",
        "logs:StartQuery",
        "logs:StopQuery",
        "logs:GetQueryResults",
        "logs:GetLogEvents"
         ],
         "Resource": "*"
    },

{
   "Sid": "AllowReadingTagsInstancesRegionsFromEC2",
     "Effect": "Allow",
     "Action": [
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
            "ec2:DescribeTags",
            "ec2:DescribeInstances",
            "ec2:DescribeRegions"
             ],
              "Resource": "*"
        },


        {
           "Sid": "AllowReadingResourcesForTags",
           "Effect": "Allow",
           "Action": "tag:GetResources",
           "Resource": "*"
        },
        {
          "Sid": "AllowReadingAcrossAccounts",
          "Effect": "Allow",
          "Action": [
            "oam:ListSinks",
            "oam:ListAttachedLinks"
            ],
             "Resource": "*"
        }
      ]


    }
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- You must also create a CloudWatch metrics configuration to enable the Amazon S3 metrics in the Grafana dashboard. See the steps mentioned under Using the S3 console in Creating a CloudWatch metrics configuration for all the objects in your bucket.

Steps

1. Log in to the CDP CLI command tool.
2. Run the following command.
   cdp opdb create-database --environment <environment_name> --database <database_name> --enable-grafana
   For example,

```
cdp opdb create-database --environment-name cod-7216-micro10
--database-name odx2408 --enable-grafana
```

Result

On successfully executing the command, the Grafana URL is added under the GRAFANA DASHBOARD option inside the COD database as shown in the following figure.

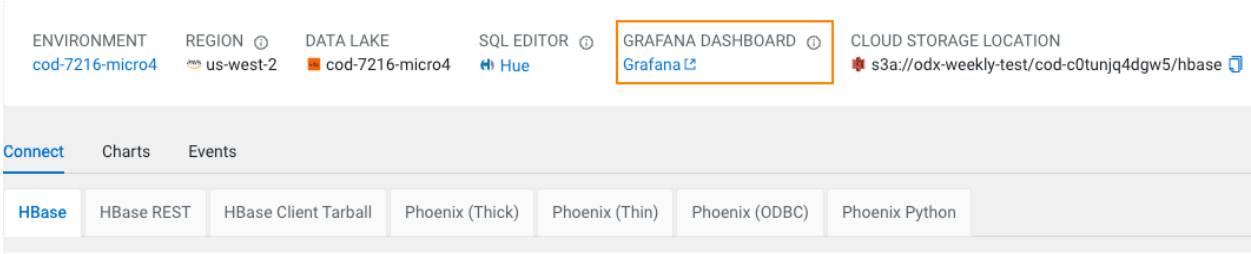

When you click on the Grafana URL, it takes you to the Grafana dashboard.

Here is an example of the HBase dashboard using Grafana.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

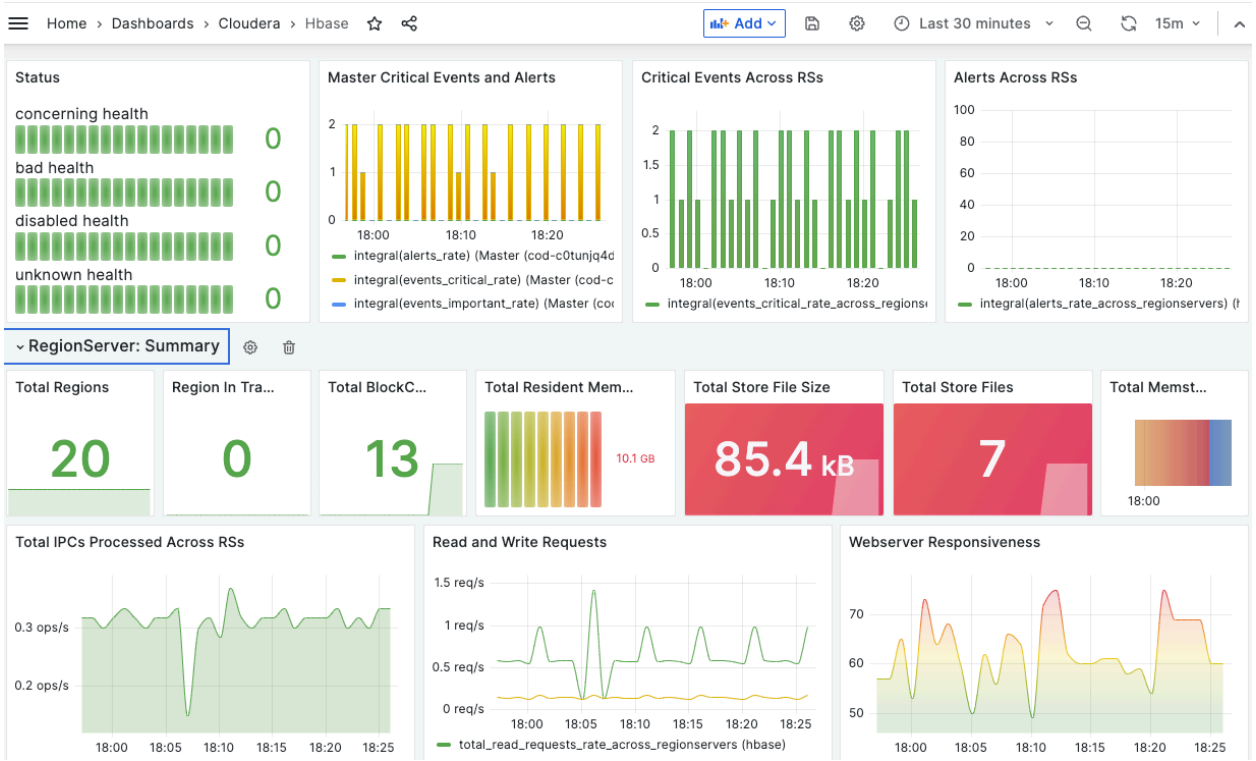

Related information

- CDP CLI Beta

Enabling Grafana dashboard for an existing COD database

Learn how to enable the Grafana URL to visualize the Cloudera Operational Database (COD) metrics for an existing COD database.

Before you begin

- You must whitelist the Cloudera archive URL so that the necessary RPM packages for Grafana can be installed in the instances.
- You must whitelist the Cloudera repository so that the dashboards are created automatically.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- You must whitelist the Grafana RPM packages URL so that Grafana can be installed in the instances.
- You must whitelist the Google API storage URL so that the Cloudera Manager fousquare plugin can be installed.
- You must attach the following policy for CloudWatch plugin under cdp-infra2-logs-role (or the role with which the ec2 instances are created) to enable the Amazon S3 metrics in Grafana dashboard. To attach a policy under the cdp-infra2-logs-rolee, see Create a cross-account IAM role.

```
{

    "Version": "2012-10-17",


    "Statement": [


      {

        "Sid": "AllowReadingMetricsFromCloudWatch",

         "Effect": "Allow",

          "Action": [

              "cloudwatch:DescribeAlarmsForMetric",

              "cloudwatch:DescribeAlarmHistory",

                "cloudwatch:DescribeAlarms",

                "cloudwatch:ListMetrics",

                "cloudwatch:GetMetricStatistics",

                "cloudwatch:GetMetricData",

                "cloudwatch:GetInsightRuleReport"

            ],

         "Resource": "*"

      },
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
{
     "Sid": "AllowReadingLogsFromCloudWatch",
      "Effect": "Allow",
       "Action": [
       "logs:DescribeLogGroups",
       "logs:GetLogGroupFields",
       "logs:StartQuery",
       "logs:StopQuery",
       "logs:GetQueryResults",
       "logs:GetLogEvents"
        ],
        "Resource": "*"
   },


{
  "Sid": "AllowReadingTagsInstancesRegionsFromEC2",
    "Effect": "Allow",
    "Action": [
    "ec2:DescribeTags",
    "ec2:DescribeInstances",
    "ec2:DescribeRegions"
     ],
      "Resource": "*"
},
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
        {
           "Sid": "AllowReadingResourcesForTags",
           "Effect": "Allow",
           "Action": "tag:GetResources",
           "Resource": "*"
        },
        {
          "Sid": "AllowReadingAcrossAccounts",
          "Effect": "Allow",
          "Action": [
            "oam:ListSinks",
            "oam:ListAttachedLinks"
            ],
             "Resource": "*"
        }
      ]


}
```

- You must also create a CloudWatch metrics configuration to enable the Amazon S3 metrics in Grafana dashboard. See the steps mentioned under Using the S3 console in Creating a CloudWatch metrics configuration for all the objects in your bucket.

Steps

1. Go to the Cloudera repository and copy these files (*grafana-install-configure-v2.sh* and *configure-knox-for-grafana.sh*) to the gateway node of the COD cluster.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

If you are using CDH version 7.2.15, download *configure-knox-for-grafana7215.sh* file. Following is an example command.

```
scp -i ~/.ssh/odx-developers.pem
./grafana-install-configure-v2.sh cloudbreak@10.1.1.1:
```

```
scp -i ~/.ssh/odx-developers.pem ./configure-knox-for-grafana.sh
cloudbreak@10.1.1.1:
```

2. Connect to the gateway node.
   Following is an example command.

```
ssh -i ~/.ssh/odx-developers.pem cloudbreak@10.8.2.5
```

3. Get the root permission for the folder where you copied the script files.
   Following is an example command.

```
sudo -i cd /home/cloudbreak/
```

4. Set the owner permission for the script files so that it can run successfully.

```
chown root:root *
```

5. Run the Grafana installation script.
   Following is an example command.

```
./grafana-install-configure-v2.sh
```

Result of step

```
extracted string: cod--186yjxqvwcwoh

Failed to set locale, defaulting to C

Loaded plugins: fastestmirror, versionlock
```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
Determining fastest mirrors

epel/x86_64/metalink
|  27 kB  00:00:00

* base: download.cf.centos.org

* centos-sclo-rh: download.cf.centos.org

* centos-sclo-sclo: download.cf.centos.org

* epel: ftp-osl.osuosl.org

* extras: download.cf.centos.org

* updates: download.cf.centos.org

  base
| 3.6 kB  00:00:00

  cdp-infra-tools
| 2.9 kB  00:00:00

  centos-sclo-rh
| 3.0 kB  00:00:00



…



Created symlink from
/etc/systemd/system/multi-user.target.wants/grafana-server.servic
e to /usr/lib/systemd/system/grafana-server.service.
```

If the command is successful, you must see the symlink created message.

6. Check the Grafana service status using the systemctl command.

```
systemctl status grafana-server.service
```

Result of step

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
grafana-server.service - Grafana instance

Loaded: loaded (/usr/lib/systemd/system/grafana-server.service;
enabled; vendor preset: disabled)

Active: active (running) since Tue 2023-08-22 10:51:08 UTC; 16s
ago

Docs: http://docs.grafana.org

Main PID: 25273 (grafana)

CGroup: /system.slice/grafana-server.service

└─25273 /usr/share/grafana/bin/grafana server
--config=/etc/grafana/grafana.ini
--pidfile=/var/run/grafana/grafana-server.pid --packaging=rpm
cfg:default.paths.logs=/var/log/gr…
```

7. Run the Knox configuration script.
   Following is an example command.

   ```
   ./configure-knox-for-grafana.sh
   ```

   If you are using CDH version 7.2.15, use the following command `./configure-knox-for-grafana7215.sh.`

   Result of step

   ```
   Installing knox service configs for grafana

   Adding GRAFANA service in the cdp-proxy topology

   Adding GRAFANA service in the cdp-proxy topology in the active
   directory
   ```

8. Restart the Knox service using Cloudera Manager.

Result

The following is a sample Grafana dashboard URL.

`https://<GATEWAY-FQDN>/grafanacod/dashboards`

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

You can obtain the value of <GATEWAY-FQDN> from COD DATAHUB > Nodes > Gateway. In the listed table, you can find the FQDN column. For example, `https://cod--186yjxqvwcwoh-gateway0.cod-7216.xcu2-8y8x.dev.cldr.work/grafanacod/dashboards`.

Related information

- CDP CLI Beta

Importing a Grafana dashboard

Know how to import an existing Grafana dashboard into your COD environment.

Before you begin

- You must whitelist the Cloudera archive URL so that the necessary RPM packages for Grafana can be installed in the instances.
- You must whitelist the Cloudera repository so that the dashboards are created automatically.

Steps

1. Download the dashboard JSON files (for example, *S3.json*) in your local computer from the Cloudera repository.
2. Open your Grafana portal and go to the Dashboard page.
3. Choose your folder where you want to install the dashboards (for example, *Cloudera*).
4. Click **New > Import** from the drop-down menu.
5. Upload the dashboard JSON file which you had downloaded earlier.

   Result of step

   After uploading the file you can see that the dashboard is created in your chosen folder.

6. Select the dashboard to see the graphs.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Navigating to tables and databases in Hue using Data Catalog (Preview)

Date published: 2021-08-07
Date modified: 2021-08-07

Legal Notice

© Cloudera Inc. 2021. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2

Contents 3

Navigating to tables and databases in Hue using Data Catalog 4

Prerequisites for accessing Hue tables and databases from Data Catalog UI 4

Accessing Hue tables and databases from the Data Catalog UI 5

Accessing Hue tables and databases from the Data Catalog Search page 5

Accessing Hue tables and databases from the Asset Details page 7

Accessing Hue assets 9

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Navigating to tables and databases in Hue using Data Catalog

Data Catalog helps you to explore assets within your Data Lake. Hue allows you to run analytics on Hive assets. The integration between Data Catalog and Cloudera Data Warehouse (CDW) service provides a direct web link to the Hue instance from the Data Catalog web UI, making it easy to navigate across services.

Note: This feature is under entitlement. Contact your Cloudera account representative to activate this feature in your CDP environment.

Prerequisites for accessing Hue tables and databases from Data Catalog UI

Hue is a part of the CDW service. Before you can access Hue from Data Catalog, you must set up the CDW service.

Note: You must have the **DWAdmin** role to perform these tasks.

1. **An AWS or Azure environment in CDW that has been activated and registered in the CDP Management Console**.

   Activating an AWS or Azure environment in CDW enables the CDW service to use the existing Data Lakes that are present in the CDP environment. This also creates a default Database Catalog that is used by a Virtual Warehouse.

   For more information, see Activating AWS environments or Activating Azure environments.

   Currently, CDW only supports AWS and Azure Cloud environments.

2. **A Hive or an Impala Virtual Warehouse created using CDW.**
   Depending on your data set type, create a Hive or an Impala Virtual Warehouse using CDW.

   For more information, see Adding a new Virtual Warehouse.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

3. **Select the asset type in the Data Catalog.**
   The asset type must be either **hive_table** or **hive_db**.

After these conditions are met, Hue links for the selected assets are displayed in the Data Catalog UI.

Accessing Hue tables and databases from the Data Catalog UI

When you log into Data Catalog, you can access web links to the Hue instances from the Data Catalog Search page or the Asset Details page.

You can access the Hue tables and databases by using one of the following options in Data Catalog:

- When you navigate to the entity search page , you can click on the icon present at the end of the row which opens the **Link to Experiences** dialog.
- Directly on the selected asset's Asset Details page, you can click **Link to Experiences**.

Accessing Hue tables and databases from the Data Catalog Search page

1. Log into Cloudera Data Catalog.
2. Select the Data Lake associated with your environment.
3. Select the asset type under **Filters**, for example Hive Table.
   Hive tables present within the Data Lake are displayed.
4. Click the icon at the end of the row for a table that you want to display in Hue:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

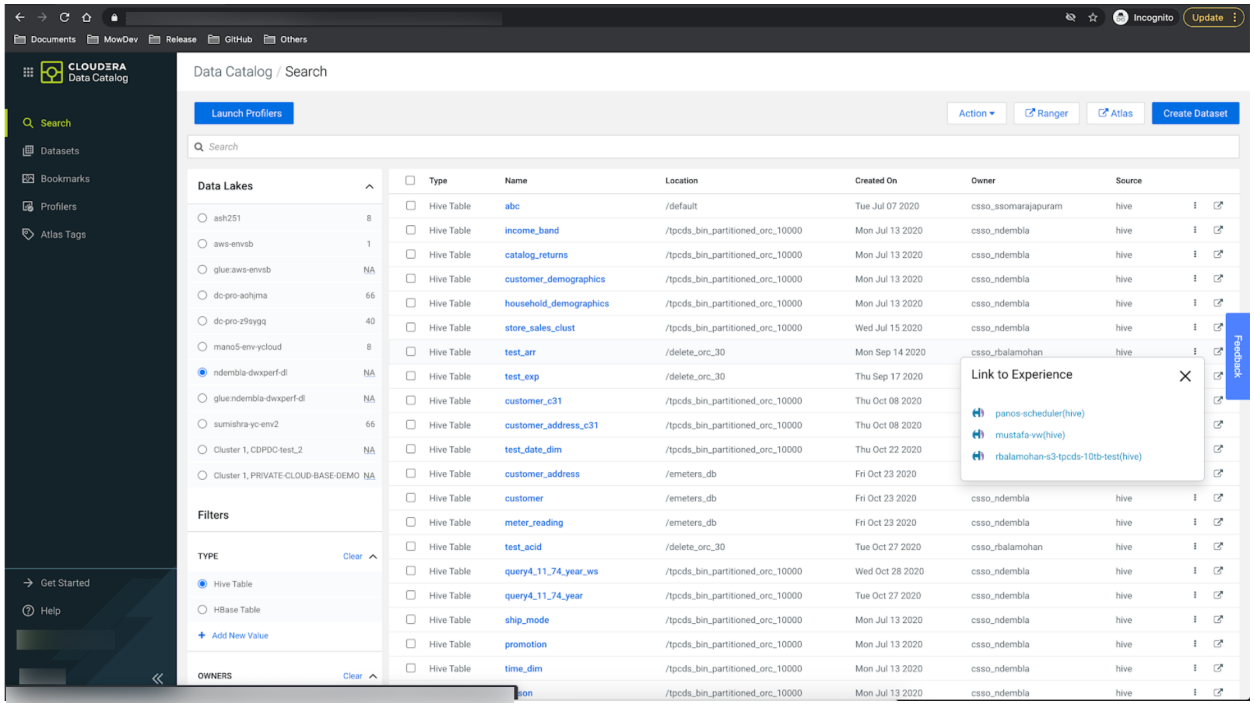

A **Link to Experiences** pop-up appears. It contains a list of Hue instances within the Virtual Warehouses that are associated with this particular table.

5. Click the link to open Hue web interface.
   The Hive table is displayed on the Hue **Table Browser**.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

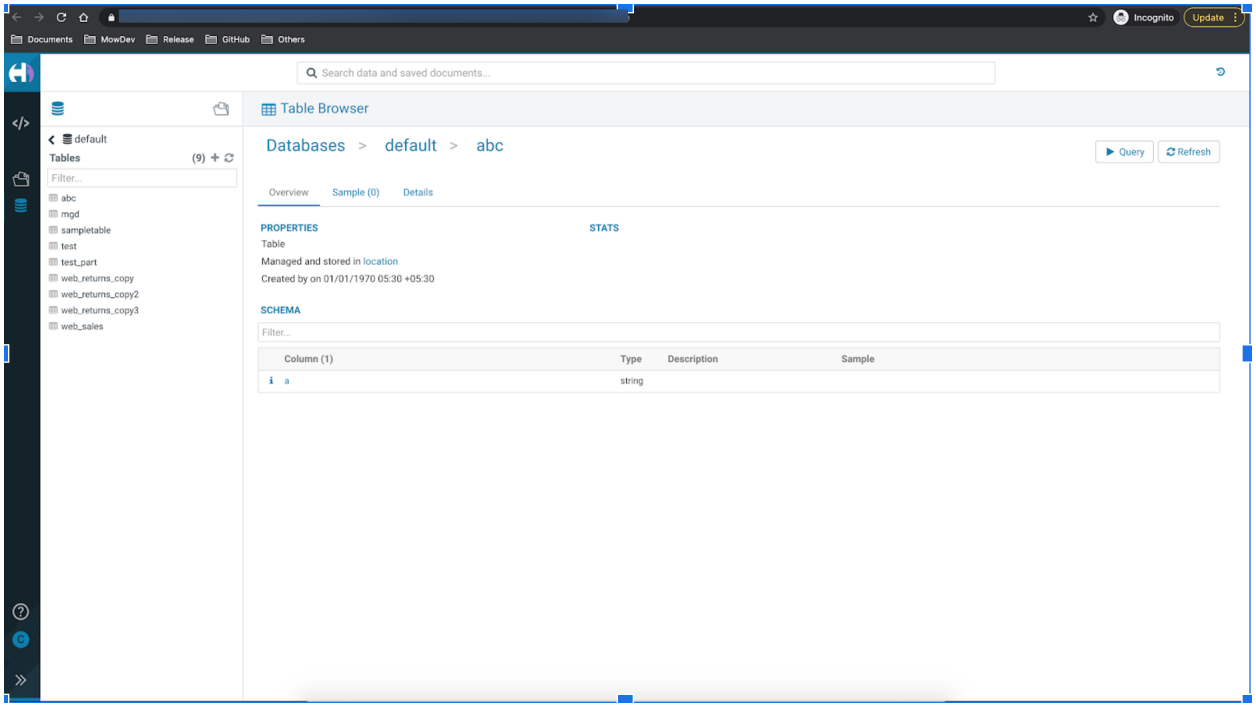

-

Accessing Hue tables and databases from the Asset Details page

1. Log into Cloudera Data Catalog.
2. Select the Data Lake associated with your environment.
3. Select the asset type under **Filters**, for example **Hive Table**.
   All the Hive tables present within the Data Lake are displayed.
4. Click on the asset name to open the **Asset Details** page.
   The **Asset Details** page contains the **Link to Experiences** button as shown in the following image:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

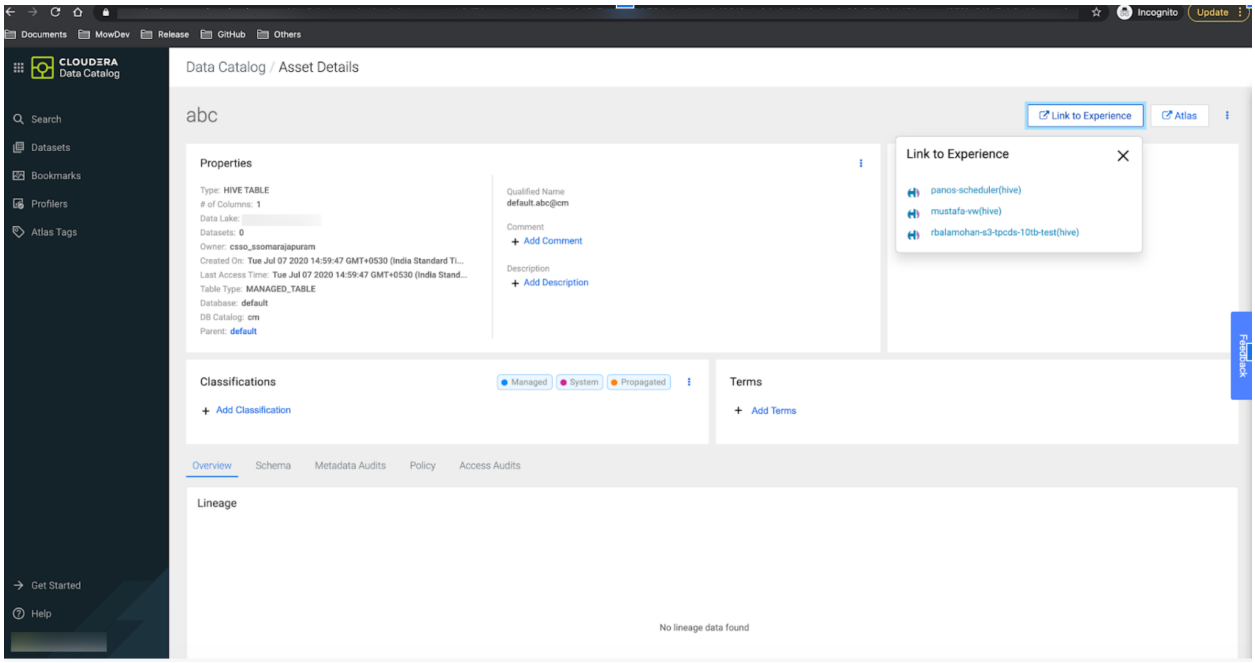

5. Click **Link to Experiences**.
   The pop-up contains a list of Hue instances within the Virtual Warehouses that are associated with this particular table.
6. Click the link to open Hue web interface.
   The Hive table is displayed on the Hue **Table Browser**.

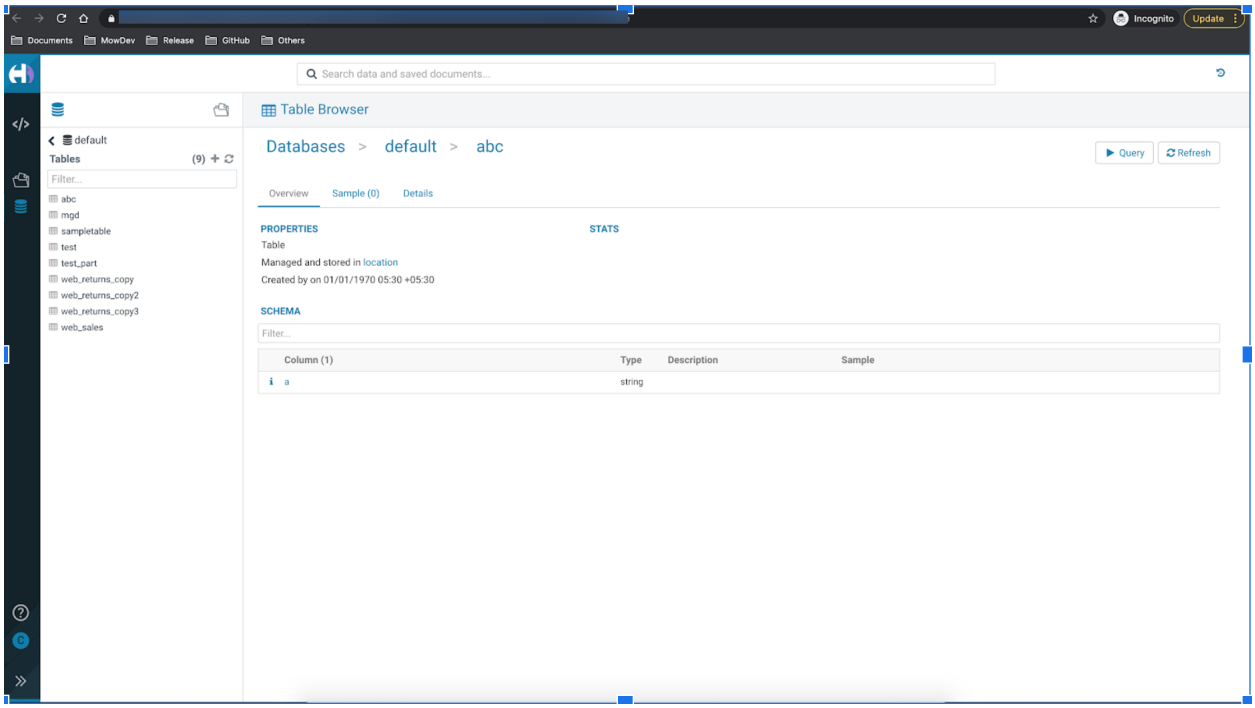

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Accessing Hue assets

Currently, you cannot access Hue assets directly from Data Catalog as this feature is under entitlement.

Note: When you are on the Data Catalog search page or the **Asset Details** page, an external link button appears if all the three conditions for viewing links to Hue are satisfied as detailed under [Prerequisites for accessing Hue tables and databases from Data Catalog UI](#).

1. You must enable the feature for the required tenant, for example, dev tenant:

   ```
   CDP_PROFILE=admin_dev——— cdpenv/bin/cdp coreadmin grant-entitlement --entitlement-name DATA_CATALOG_ENABLE_NAVIGATION_TO_EXPERIENCE --account-id 9d74eee4-1cad-45d7-b645-7ccf9edbb73d
   ```

   This will grant the entitlement for the dev tenant.

2. Enable access to AWS or Azure environments for Hue.
3. Activate CDW for AWS.

   To activate CDW on your selected environment, follow these steps:

   - Under **Management Console** > select **Data Warehouse**.
   - On the **Overview** page, click on the left pane property displaying **Environments**. AWS/Azure environments are listed.
   - Locate the enabled AWS / Azure environment and activate CDW by clicking on the bolt icon. This action also creates a default **Database Catalogs**.

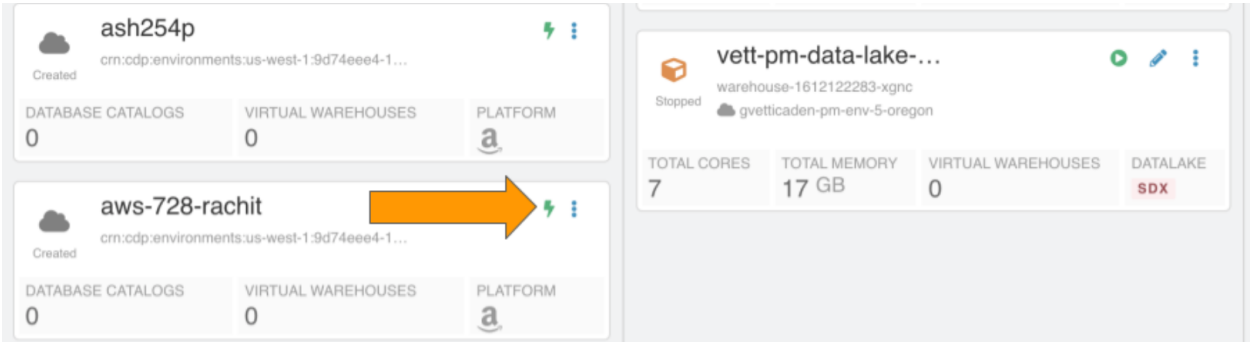

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

3. Once the environment is activated and a DB Catalog is created and running, you can create a Virtual Warehouse as per the requirement.

   For more information about AWS and Azure Environment, see [AWS environments overview] and [Azure environments overview].

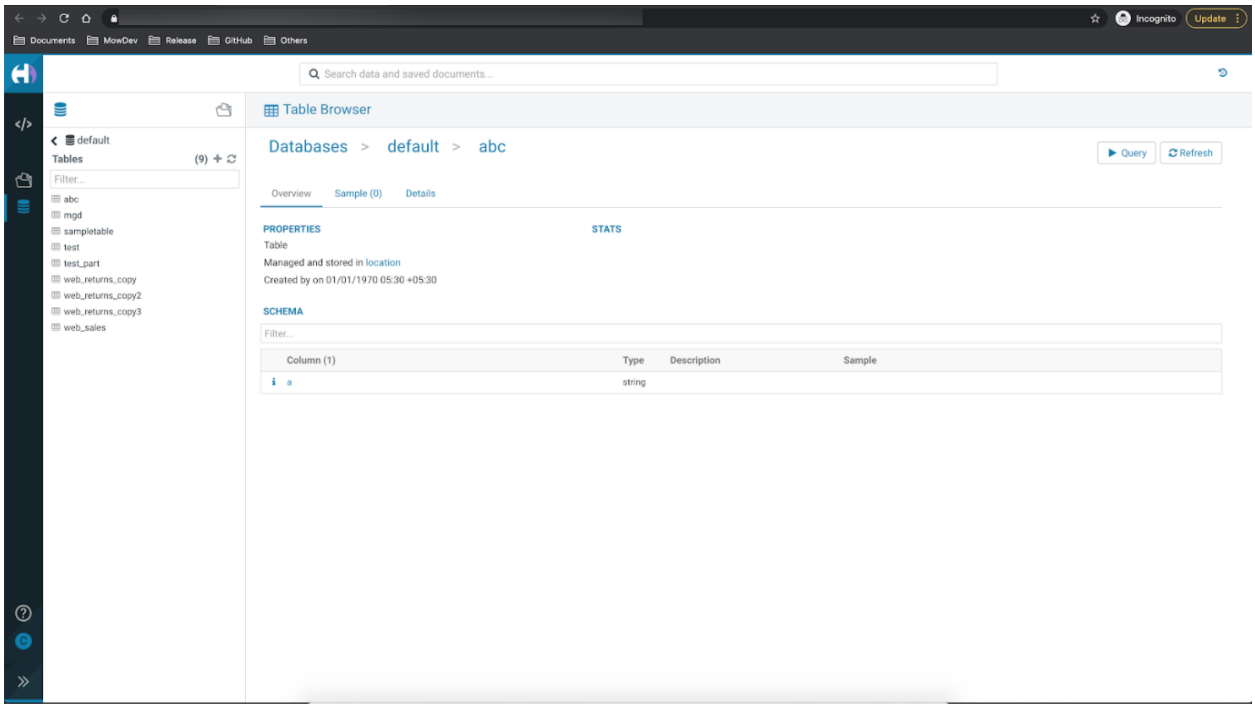

You will now be able to navigate to the Hue assets on these virtual warehouses using the links provided by Data Catalog.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Reserving nodes for auto-scaling (Preview)

Date published: 2022-07-26
Date modified: 2022-07-26

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**

Contents **3**

Reserving nodes for auto-scaling **4**

Getting the Availability Zone used by the cluster overprovisioner 5

Changing the Availability Zone used by the overprovisioner 5

Creating the Virtual Warehouse 6

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Reserving nodes for auto-scaling

To speed up Virtual Warehouse startup and autoscaling, you need to keep some number of compute instances on standby. You need to know how to configure extra buffer nodes to stand by, ready to join a new compute or autoscaled cluster.

Important information about this feature is:

- This feature supports AWS and Azure.
- This feature is behind an entitlement: CDW_CLUSTER_OVERPROVISIONER.
- Adding more reserved nodes increases your cloud costs.
- This feature is a technical preview.

Before you begin:

- You are logged into one of the following CDP environments in Cloudera Data Warehouse (CDW) Kubernetes Public Cloud service.
    - AWS
    - Azure
- You have the CDW_CLUSTER_OVERPROVISIONER entitlement to use this feature.

Required role: DWAdmin

About this task: You increase reserved nodes by configuring the **Overprovision compute nodes or Overprovision shared service nodes** when you activate the CDW environment. Use the slider to set the Node Count to the additional number of nodes you want to reserve. Compute nodes are used by executors and coordinators, while the shared service nodes are used by other services in the cluster. Compute nodes are relevant to autoscaling events as executors or coordinators are scaled up/down during autoscaling.

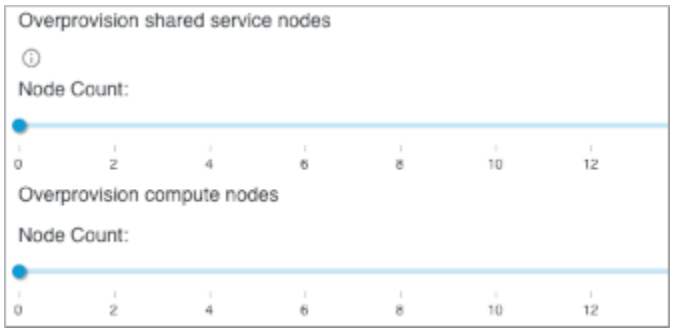

Adding more reserved nodes increases your cloud costs.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Getting the Availability Zone used by the cluster overprovisioner

When using this feature for compute nodes in AWS, make sure the Availability Zone selected during Virtual Warehouse creation is the same as the one being used by the cluster overprovisioner.

You can determine Availability Zone used by the cluster overprovisioner by following the steps:

1. Run the command to get the compute-reserved-node deployment spec.

   ```
   kubectl get deployment compute-reserved-node -n cluster -o yaml
   ```

2. Open the YAML and locate the spec > affinity > nodeAffinity.

   ```
   spec:
         affinity:
           nodeAffinity:
             requiredDuringSchedulingIgnoredDuringExecution:
               nodeSelectorTerms:
               - matchExpressions:
                 - key: failure-domain.beta.kubernetes.io/zone
                   operator: In
                   values:
                   - us-west-2a
                 - key: compute
                   operator: In
                   values:
                   - "true"
   ```

   In this example, you see us-west-2a is the availability zone being used.

Changing the Availability Zone used by the overprovisioner

Generally, you do not change the Availability Zone used by the overprovisioner, but it is possible to do so. For example, if the cloud provider runs out of nodes in an Availability Zone, or for some other reason, you do not want to configure the Virtual Warehouse in the default Availability Zone of the overprovisioner, change the Availability Zone used by the overprovisioner.

In this task, you edit the compute-reserved-node deployment script to change the value for the key failure-domain.beta.kubernetes.io/zone to the Availability Zone you want, and save the changes.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

1. Edit the deployment script.

```
kubectl edit deployment compute-reserved-node -n cluster
```

2. Update the value for failure-domain.beta.kubernetes.io/zone with the AZ you want, such as *us-west-2b* or *us-west-2c*, associated with your CDW env.
3. Save the changes to the deployment spec.

Creating the Virtual Warehouse

1. Create a Virtual Warehouse as described in Adding a new Virtual Warehouse, selecting a Hive or Impala SQL engine type.
2. Configure the Virtual Warehouse properties: Name, Type, Database Catalog.
3. Select the Availability Zone that the cluster overprovisioner uses, as described above. Alternatively, change the Availability Zone of the cluster overprovisioner, as described above, and select that Availability Zone for the Virtual Warehouse.

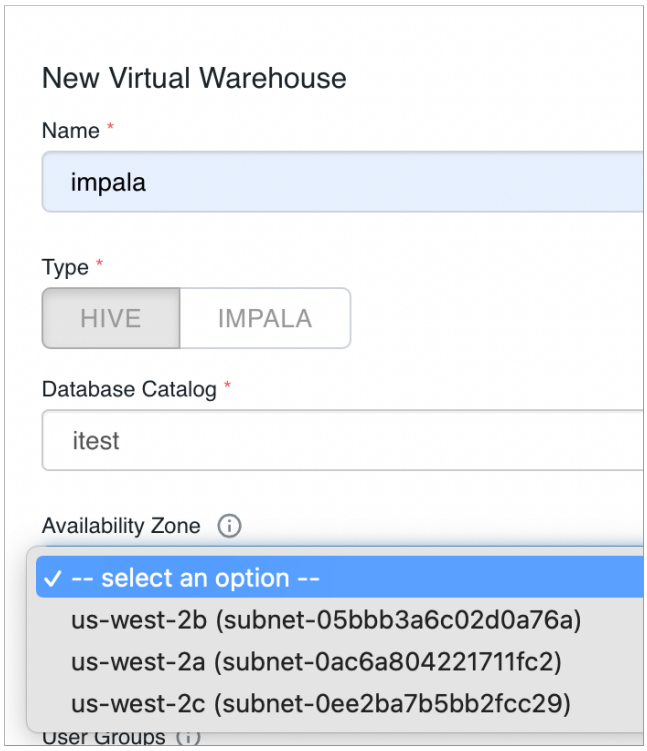

4. Click Create.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Rolling upgrade in Cloudera Operational Database (Preview)

Date published: 2023-09-28
Date modified: 2024-10-01

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
Performing a Cloudera Runtime upgrade **4**
Performing a Cloudera operating system upgrade **7**
Rolling upgrade limitations **9**
Operating system upgrade limitations 9

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Performing a Cloudera Runtime upgrade

Learn how to perform a rolling and a non-rolling Cloudera Runtime upgrade for your existing Cloudera Operational Database in the CDP environment.

About this task

- You must use the CDP beta CLI and run the `upgrade-database` command to upgrade your database.
- The Cloudera Runtime rolling upgrade is currently supported for the operational database clusters whose storage is selected as HDFS or cloud without ephemeral storage.
- Zero downtime upgrade or rolling upgrade is not supported on the Micro operational database clusters.
- It is recommended not to perform backups or run YARN jobs during the operational database upgrade operations.

Note:
Cloudera Runtime upgrade is supported from the CDP Runtime version 7.2.16 onwards.

Before you begin

- You must have the ODAdmin rights to make changes to the operational database database.
- In the Cloudera Manager, increase the **Region Mover Threads** property for the HBase service to 30 for a faster rolling upgrade.
- In the Cloudera Manager, increase the **Omid Max Heapsize** property for the Omid service to at least 3GB.
- You must download and install the latest CDP CLI beta version.

Steps

1. Launch the CDP CLI tool.
2. Run the following command to check the available Cloudera Runtime upgrades.

   Important:
   Ensure that a Cloudera Runtime upgrade is available from the specified target.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

```
cdp datahub upgrade-cluster --cluster-name <cod-internal-name>
--show-latest-available-image-per-runtime
```

Following is a sample output where a runtime upgrade is possible from 7.2.17 to 7.2.17.x (hotfix upgrade).

Validate the output that `upgradeCandidates` contains, which highlights a different upgrade target compared to the current one.


```
  {
"current": {
    "imageName": "ami-0b9adbe77c66a2277",
    "imageId": "0df7887b-a841-4569-aab5-269041a015eb",
    "imageCatalogName": "cdp-default",
    "created": 1694041314,
    "componentVersions": {
        "cm": "7.11.0",
        "cmGBN": "42373020",
        "cdp": "7.2.17",
        "cdpGBN": "42350016",
        "os": "centos7",
        "osPatchLevel": "2023-09-06"
    }
},
"upgradeCandidates": [
    {
        "imageName": "ami-0ac26feaf1ae4f9e8",
        "imageId": "42e762f0-9731-4056-83b0-b9e7bbe7c5fb",
        "imageCatalogName": "cdp-default",
        "created": 1694777926,
        "componentVersions": {
            "cm": "7.11.0",
            "cmGBN": "44461729",
            "cdp": "7.2.17",
            "cdpGBN": "44441663",
            "os": "centos7",
            "osPatchLevel": "2023-09-15"
        }
    }
],
"reason": ""

}
```


This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

3. Run the following command to perform a rolling Cloudera Runtime upgrade for the database.

   The following command upgrades the Cloudera Runtime for the operational database using the rolling restart mode that ensures continuous service availability.

```
cdp opdb upgrade-database --environment <environment-name>
--database <database-name> [--runtime <runtime-version> |
--image <imageId>] --rolling-upgrade
```

Option	Description
--environment (string)	The name or CRN of the environment.
--database <value>	The name or CRN of the database.
--runtime <value>	The 3-digit runtime version to upgrade to. Alternatively, specify the `--image` option.
--image <value>	The image ID to upgrade to. Alternatively, specify the `--runtime` option.
--rolling-upgrade	Controls whether to perform a rolling upgrade for the operational database.

Run the following command to perform a non-rolling Cloudera Runtime upgrade for the database.

```
cdp opdb upgrade-database --environment <environment-name>
--database <database-name> [--runtime <runtime-version> | --image
<imageId>]
```

Result

The Cloudera Operational Database is upgraded to the provided runtime version in a rolling mode.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Related information

- Upgrading Cloudera Operational Database clusters
- COD User Management
- CDP Environments
- COD CLI command reference GitHub repository
- CDP CLI BETA command reference GitHub repository

Performing a Cloudera operating system upgrade

Learn how to perform a rolling and non-rolling Operating System (OS) upgrade for your existing Cloudera Operational Database in the CDP environment.

About this task

- You must use the CDP Beta CLI and run the `upgrade-database` command to upgrade your database.
- The operating system rolling upgrade is currently supported for the operational database clusters whose storage is selected as HDFS or cloud without ephemeral storage.
- Zero downtime upgrade or rolling upgrade is not supported on the Micro operational database clusters.
- It is recommended not to perform backups or run YARN jobs during the operational database upgrade operations.

Note:
Cloudera operating system upgrade is supported from the CDP Runtime version 7.2.16 onwards.

Before you begin

- You must have the ODAdmin rights to make changes to the operational database database.
- In the Cloudera Manager, increase the Region Mover Threads property for the HBase service to 30 for a faster rolling upgrade.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- In the Cloudera Manager, increase the Omid Max Heapsize property for the Omid service to at least 3GB.
- You must download and install the latest CDP CLI beta version.

Steps

1. Launch the CDP CLI tool.
2. Run the following command to perform a rolling OS upgrade for the database.
   This command upgrades the OS for the operational database using the rolling restart mode that ensures continuous service availability.
   ```
   cdp opdb upgrade-database --environment <environment-name>
   --database <database-name> [--runtime <runtime-version> |
   --image <imageId>] --os-upgrade-only --rolling-upgrade
   ```

Option	Description
--environment (string)	The name or CRN of the environment.
--database <value>	The name or CRN of the database.
--os-upgrade-only	Requests OS upgrade.
--runtime <value>	The 3-digit runtime version to upgrade to. Alternatively, specify the --image option.
--image <value>	The image ID to upgrade to. Alternatively, specify the --runtime option.
--rolling-upgrade	Controls whether to perform a rolling upgrade for the operational database.
--edge-upgrade-strategy <type>	Controls the upgrade strategy for the edge nodes in the cluster. Following are the supported values: • ALL: Upgrades all the edge nodes together. • ONE_BY_ONE: Upgrades the edge nodes, one by one.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Run the following command to perform a non-rolling OS upgrade for the database.

```
cdp opdb upgrade-database --environment <environment-name> --database
<database-name> [--runtime <runtime-version> | --image <imageId>]
--os-upgrade-only
```

Result

The Cloudera Operational Database is upgraded to the provided operating system version in a rolling mode.

Related information

- Upgrading Cloudera Operational Database clusters
- COD User Management
- CDP Environments
- COD CLI command reference GitHub repository
- CDP CLI BETA command reference GitHub repository

Rolling upgrade limitations

Consider the following points while performing a rolling runtime and an operating system upgrade of an operational database cluster.

Operating system upgrade limitations

- An unwanted rolling OS upgrade can be triggered if a runtime upgrade is not available, resulting in operational database downtime. Ensure that the runtime upgrade is available for the selected runtime using the cdp opdb describe-upgrade-database command before using the following command.
  ```
  cdp opdb upgrade-database --environment <environment-name>
  --database <database-name> --runtime <runtime-version>
  --rolling-upgrade
  ```
- The operational database rolling upgrade is focused on HBase and Phoenix thick clients excluding Phoenix Query Servers (PQS) because PQS connectivity depends on Knox which is not deployed in HA mode. When Knox is unavailable during the upgrade, PQS, HBase REST and other endpoints connected through Knox can be interrupted.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- The Cloudera Runtime and operating system rolling upgrades are currently supported for the operational database clusters whose storage is selected as HDFS or cloud without ephemeral storage.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Scaling Cloudera Operational Database instances vertically (Preview)

Date published: 2023-05-10
Date modified: 2024-10-01

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2
Contents 3
Scaling Cloudera Operational Database instances vertically 4

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Scaling Cloudera Operational Database instances vertically

You can scale up a Cloudera Operational Database (COD) cluster vertically, which means upgrading a COD cluster from a LIGHT duty to a HEAVY duty instance type. You can only upgrade COD clusters belonging to the Gateway and Master nodes.

About this task

Selecting a larger instance type adds more vCPU and/or RAM to your instances. You can scale up instances from LIGHT to HEAVY using the CDP CLI command tool.

Before you begin

- You must stop the COD cluster before you vertically scale any of the instances.
- You must grant **CDP_CB_AWS_VERTICAL_SCALE** entitlement for the specific tenant.
- You must stop EBS-backed instances before scaling. Vertical scaling is supported on AWS only. See Change the instance type in AWS documentation for more information.
- You must ensure that the COD clusters belong only to Gateway nodes or Master nodes.
- You must download and install the latest CDP CLI beta version. For more information, see, Installing Beta CDP CLI.

Steps

1. Log in to the CDP CLI command tool.
2. Run the following command

   ```
   cdp opdb update-database --environment <environment_name>
   --database <database_name> --vertical-scale
   <Group_type_for_the_database_nodes>
   ```

   The following are the possible values for the group type for the database nodes:

   - **GATEWAY**: Value of the group variable as GATEWAY.
   - **MASTER**: Value of the group variable as MASTER.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

For example,

```
cdp opdb update-database --environment-name cod-7216-micro10
--database-name odx2408 --vertical-scale MASTER
```

After you finish

Start the cluster after you have vertically scaled it. You can also configure the services on the cluster to use the additional or reduced resources/memory.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Schedule-based Autoscaling for Cloudera Data Hub Clusters Using Impala (Preview)

Date published: 2023-11-09
Date modified: 2024-12-04

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Legal Notice

© Cloudera Inc. 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice **2**
Contents **3**
Schedule-based autoscaling for Data Hub clusters using Impala **4**
Autoscaling behavior 5
Configuring schedule-based autoscaling for Data Hub clusters using Impala 5
Configuring schedule-based autoscaling with the CDP CLI for Data Hub clusters using Impala 7

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Schedule-based autoscaling for Data Hub clusters using Impala

Autoscaling is a feature that adjusts the capacity of cluster nodes running YARN and/or Impala by automatically increasing or decreasing, or suspending and resuming, the nodes in a host group.

Generally speaking, you can enable autoscaling based either on a schedule that you define (schedule-based autoscaling), or the real-time demands of your workloads (load-based autoscaling). However, for cluster nodes that run only Impala, such as the Data Mart clusters, you are limited to only **schedule-based autoscaling policies**.

Important! This document focuses on schedule-based autoscaling for clusters running Impala, as both load and schedule-based autoscaling for YARN is covered in Autoscaling clusters.

Schedule-based autoscaling for Impala scales the number of nodes in an executor host group up or down based upon a schedule that you define. Schedule-based autoscaling is useful if workload demands tend to be high or low on a fairly regular, consistent basis.

When you configure a schedule-based autoscaling policy, you define a target node count, which is the number of nodes that you want to scale up or down to at a particular time. You finalize the schedule by entering a CRON expression and selecting the desired timezone. When the particular time/date that you define in the CRON expression occurs, the cluster is upscaled or downscaled to your target node count. The time taken to add nodes to a cluster varies by cloud provider and the configuration of the nodes (for example, recipes can have an impact on the time it takes to add a node).

Typically, schedules (especially those configured to scale-up) should be defined in a manner to factor in the time it takes to add new nodes. For example, workloads starting at 9pm will typically set a scale-up schedule for 8:45 pm, assuming it takes around 15 minutes to add a node.

Note: Schedule-based autoscaling for Impala is available for clusters provisioned in AWS and Azure with Cloudera Runtime versions 7.2.15 or newer. This is a technical preview feature under entitlement. This feature is available by default in the Data Mart cluster definitions. Schedule-based autoscaling can also be configured for clusters with custom templates that include Impala.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Autoscaling behavior

Before you define an autoscaling policy, note the following autoscaling behaviors:

- Schedule-based autoscaling for Impala is available for executor host groups only on clusters running Impala.
- During scale-down, nodes are decommissioned randomly. Note that if YARN is also running in addition to Impala, YARN takes priority and nodes are decommissioned based on the recommendations of YARN.
- Clusters can perform one upscale or downscale operation at a time.
- A cluster will continue to accept jobs while it is running, regardless of any in-progress upscale or downscale operations.
- You can create just one or multiple schedule-based policies for a host group.
- If there are not enough nodes available to match the requested scale operation, the operation will proceed on however many nodes are available (for example, during a request for a 10 node scale-up, if the cluster loses 1 node, the operation will proceed with scaling-up 9 nodes instead of 10).
- Autoscaling will be disabled if the Cloudera Manager cluster node fails.

Configuring schedule-based autoscaling for Cloudera Data Hub clusters using Impala

To configure autoscaling for clusters using Impala, add a schedule-based policy to a cluster and define the policy parameters.

1. From the Cloudera Management Console, click **Data Hub Clusters** and then select the cluster that you want to add an autoscaling policy to. You can add autoscaling policies to clusters after they have been created, not during the cluster creation process.
2. From the cluster details page, select the **Autoscale** tab and then click the slider button to enable autoscaling. Autoscaling is disabled by default.
3. Click **Add Autoscale Policy** and select **Schedule-Based**.
4. Define the policy parameters:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

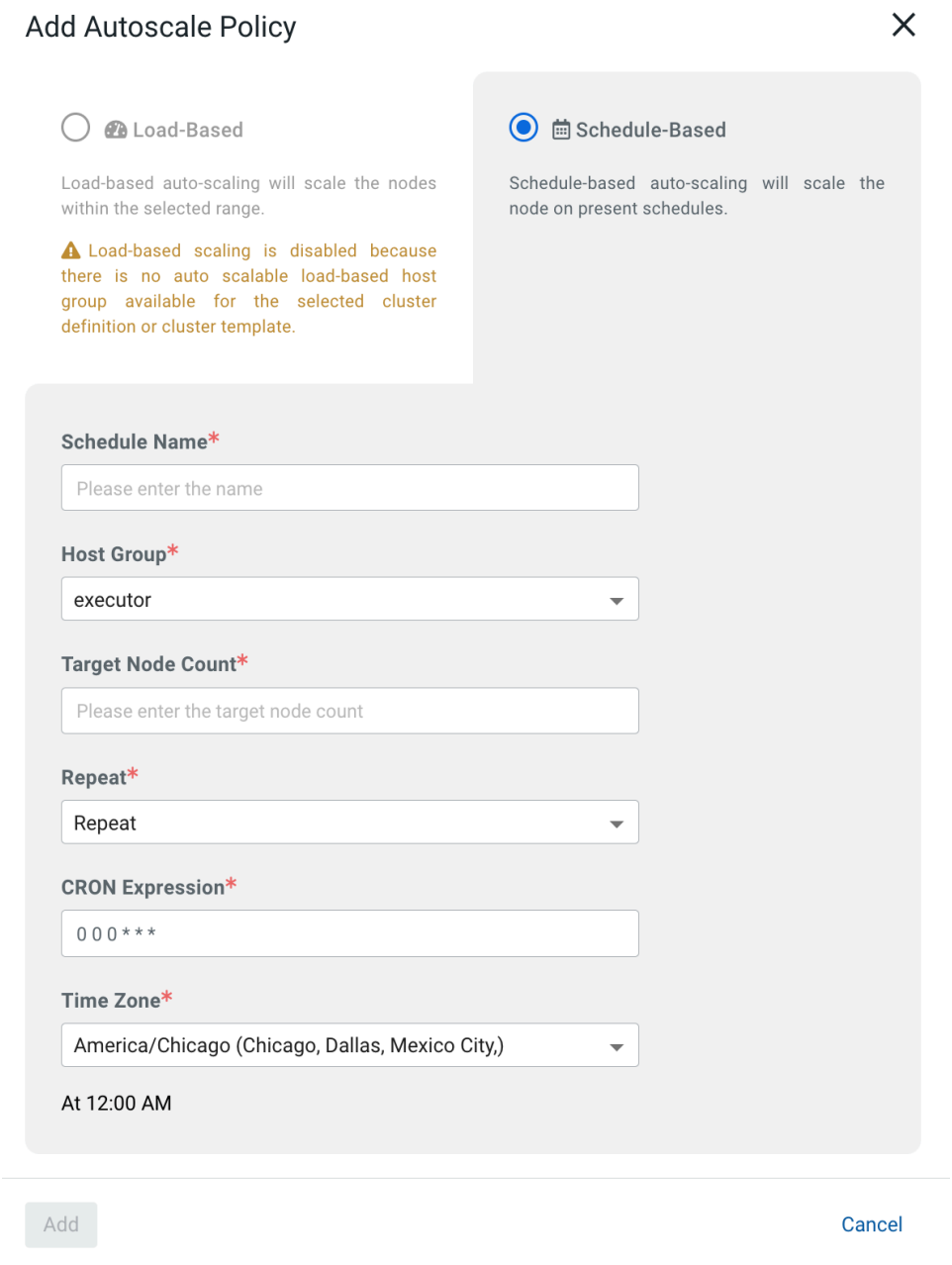

Parameter	Description
Schedule Name	Enter a unique name for the schedule.
Hostgroup	Select the host group that you want to scale. The list of available host groups is determined by which host groups include services that can be scaled.
Target Node Count	Enter the number of nodes that you want to upscale or downscale the host group to.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Repeat	Currently, only repeating schedules are supported.
CRON Expression	Enter a CRON expression string to define the details of the schedule that you want to create.
Time Zone	Select the appropriate timezone on which to base the CRON expression.

5. Click **Add**. The policy appears under Auto Scaling on the **Provision Data Hub** page.

Configuring schedule-based autoscaling with the CDP CLI for Cloudera Data Hub clusters using Impala

Use the CDP CLI to add a schedule-based autoscaling policy for Impala clusters and define the policy parameters.

To create an autoscaling policy, use the CLI command `create-auto-scale-rules:`

```
create-auto-scale-rules
--cluster-name <value>
--auto-scale-policies <value>
[--enabled | --no-enabled]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

Option	Description
`--cluster-name (string)`	Name or CRN of cluster on which to create the autoscaling policy.
`--auto-scale-policies (array)`	JSON array containing the actual policy values.
`[--enabled | --no-enabled] (boolean)`	Enables or disables the autoscaling policy on the cluster.
`[--cli-input-json] (string)`	Performs service operation based on the JSON string provided. The JSON string

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

	follows the format provided by `--generate-cli-skeleton`. If other arguments are provided on the command line, the CLI values will override the JSON provided values.
`[--generate-cli-skeleton] (boolean)`	Prints a sample input JSON to standard output. Note the specified operation is not run if this argument is specified. The sample input can be used as an argument for `--cli-input-json`.

Example JSON syntax for a schedule-based policy:

```
cdp datahub create-auto-scale-rules --cli-input-json \
"{
    \"clusterName\": \"cluster_name\",
    \"autoScalePolicies\": [
            {
                \"hostGroups\": \"host_group\",
                \"scheduleBasedPolicy\": {
                    \"schedules\": [
                        {
                            \"configuration\": {
                                \"trigger\": {
                                    \"cronExpression\": \"cron_expression\",
                                    \"timeZone\": \"time_zone\"
                                },
                                \"action\": {
                                    \"resourceAdjustmentType\": \"ABSOLUTE_COUNT\",
                                    \"resourceAdjustmentValue\": resource_count
                                }
                            },
                            \"identifier\": \"policy_name\",
                            \"description\": \"description\"
                        }
                    ]
                }
             }
        ],
    \"enabled\": true,
    \"useStopStartMechanism\": false
}"
```

Other autoscaling CLI commands include:

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

- list-scaling-activities
- describe-scaling-activity
- delete-auto-scale-rules
- describe-auto-scale-rules
- list-auto-scale-history
- update-auto-scale-rules

See the Data Hub CDP CLI documentation and Managing autoscaling for details.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Snapshot Policies in Replication Manager (Preview)

Date published: 2022-02-25
Date modified: 2022-02-25

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2

Contents 3

What's New 4

Snapshots page in Replication Manager 4

Snapshots 5

HDFS snapshots 6

Requirements and benefits of HDFS snapshots 7

HBase snapshots 8

Snapshot policies 9

Enabling and taking snapshots in Cloudera Manager 9

Creating snapshot policies in Replication Manager 10

Snapshot policy operations 11

View snapshot history 12

Restoring snapshots in Replication Manager 13

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

What's New

A snapshot is a set of metadata information and a point-in-time backup of HDFS files and HBase tables. You can create snapshot policies for HDFS directories and HBase tables in registered classic clusters and SDX Data Lake clusters to take snapshots at regular intervals. Before you create an HDFS snapshot policy for an HDFS directory, you must enable snapshots for the directory in Cloudera Manager.

After a snapshot policy takes a snapshot of the HDFS directory or HBase table, you can perform the following tasks:

- Restore the snapshot to the original directory or table using the **Restore Snapshot** option.
- Restore a directory or table to a different directory or table using the **Restore Snapshot as** option.

Snapshots page in Replication Manager

The **Snapshots** page in Replication Manager shows the list of snapshot policies with its current status, the classic cluster and the directories and tables that reside on it for which you are taking snapshots using the policy, the configured schedule to take snapshots, and the last policy run. You can edit, suspend, or delete the snapshot policy, and run the snapshot policy on demand on the **Snapshots** page.

When you click a snapshot on the **Snapshots** page, the **Snapshot History** tab appears. Along with the tab, the page also shows the classic cluster name, service name as HDFS or HBase, the directories, paths, or tables chosen for the snapshot policy, number of times the policy ran, and the snapshot location. You can also perform actions such editing the policy, suspend or resume, and remove the snapshot policy.

The **Snapshot History** tab shows the list of snapshots, current status, timestamp of the previous snapshot policy run, the configured schedule for the snapshot, and the time the snapshot expires. After the snapshot exceeds the maximum number specified in **Create Snapshot Policy > Keep** field, the least recent snapshot is deleted by Cloudera Manager and the snapshot is marked as **Expired**. It appears in the **Expired Snapshot** list on the **Snapshots History** tab. Use the **Search Snapshot History** field to search for snapshots. You can view the logs for a snapshot, restore a snapshot, or delete a snapshot.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Snapshots

A snapshot is a set of metadata information and a point-in-time backup of HDFS files and HBase tables. Snapshots save the state of all or part of the HDFS filesystem or a set of HBase tables at a specified point in time. You can create snapshot policies to take snapshots of HDFS directories and HBase tables in the registered classic clusters at regular intervals.

Snapshots are efficient because they copy only the changed data. You can schedule the snapshot policies to take snapshots at regular intervals or on demand. You can restore the data in these snapshots to a version of the data from any previous snapshot in Replication Manager.

Cloudera Manager and Replication Manager support the following snapshots:

- **HDFS snapshots.** These are point-in-time backup of HDFS directories or the entire filesystem without data cloning. These snapshots appear on the filesystem as read-only directories that can be accessed just like other ordinary directories. Snapshots improve data replication performance and prevent errors caused by changes to a source directory. Before you create a HDFS snapshot policy for an HDFS directory in Replication Manager, you must enable snapshots for the directory in the cluster's Cloudera Manager. Enable snapshots for the HDFS replication source directories to improve HDFS replication performance. Enable snapshot for Hive warehouse directories to improve Hive replications. For more information about snapshots and how to enable HDFS snapshots, see HDFS snapshots.
- **HBase snapshots.** These are point-in-time backups of tables without making data copies and with minimal impact on RegionServers. Snapshots are enabled by default. For more information, see HBase snapshots.

You can use snapshots to restore a directory or table (to a previous version) to the original location or to a different location. For example, you enable snapshots for an HDFS directory **/hive/tmp** in Cloudera Manager on Monday. You then create a snapshot policy **hive-tmp-snapshot** for the directory in Replication Manager where you set a daily schedule for taking snapshots. After a few days, you notice that a user deleted some files by mistake on Thursday. To restore the deleted files, you can restore the snapshot taken on Wednesday that has the deleted files. To restore the files, go to the **Replication Manager > Snapshots Policies** page, click the snapshot policy to view the snapshot history. Click **Restore Snapshot** action for the snapshot taken on Wednesday. This action triggers an HDFS replication policy job to restore the data.

Note: When a file or directory is deleted, the snapshot ensures that the file blocks corresponding to the deleted files or directories are not removed from the file system. Only the metadata is modified to reflect the deletion.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

HDFS snapshots

You can schedule taking HDFS snapshots for replication in the Replication Manager. HDFS snapshots are read-only point-in-time copies of the filesystem. You can enable snapshots on the entire filesystem, or on a subtree of the filesystem. In Replication Manager, you take snapshots at a dataset level. Understanding how snapshots work and some of the benefits and costs involved can help you to decide whether or not to enable snapshots.

To improve the performance and consistency of HDFS replications, enable the HDFS replication source directories for snapshots, and for Hive replications, enable the Hive warehouse directory for snapshots. For more information, see HDFS snapshots.

Enabling snapshots on a folder requires HDFS admin permissions because it impacts the NameNode. When you enable snapshots, all the subdirectories are automatically enabled for snapshots as well. So when you create a snapshot copy of a directory, all content in that directory including the subdirectories is included as part of the copy. If a directory contains snapshots but the directory is no longer snapshot-enabled, you must delete the snapshots before you enable the snapshot capability on the directory.

Take snapshots on the highest-level parent directory that is snapshot-enabled. Snapshot operations are not allowed on a directory if one of its parent directories is already snapshot-enabled (snapshottable) or if descendants already contain snapshots.

For example, in the following directory tree image, if directory-1 is snapshot-enabled but you want to replicate subdirectory-2, you cannot select only subdirectory-2 for replication. You must select directory-1 for your replication policy.

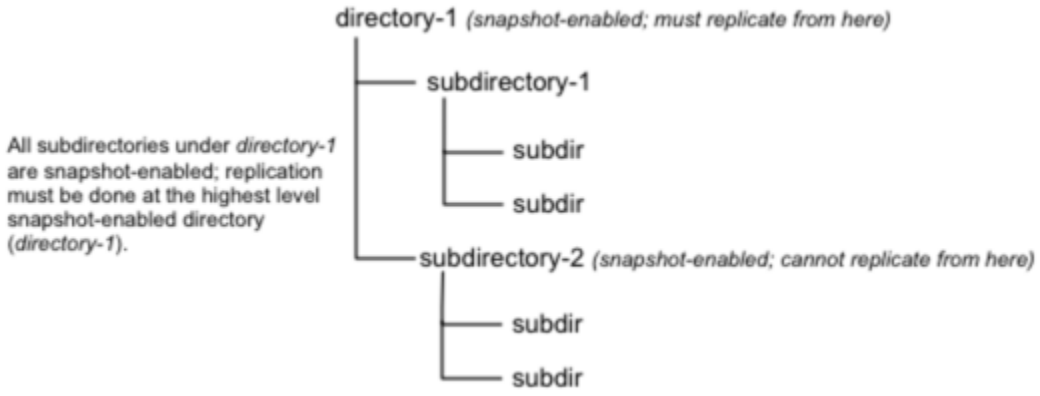

There is no limit to the number of snapshot-enabled directories you can have. A snapshot-enabled directory can accommodate 65,536 simultaneous snapshots. Blocks in datanodes are not copied during snapshot replication. The snapshot files record the block list and the file size. There is no data copying.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

When snapshots are initially created, a directory named ***.snapshot*** is created on the source and destination clusters under the directory being copied. All snapshots are retained within .snapshot directories. By default, the last three snapshots of a file or directory are retained. Snapshots older than the last three are automatically deleted.

Requirements and benefits of HDFS snapshots

You might want to consider the benefits and memory cost of using snapshots. Verify the requirements before you enable snapshots.

Requirements
You must have HDFS superuser privilege to enable or disable snapshot operations. Replication using snapshots requires that the target filesystem data being replicated is identical to the source data for a given snapshot. *There must be no modification to the data on the target*. Otherwise, the integrity of the snapshot cannot be guaranteed on the target and replication can fail in various ways.

Benefits
Snapshot-based replication helps you to avoid unnecessary copying of renamed files and directories. If a large directory is renamed on the source side, a regular DistCp update operation sees the renamed directory as a new one and copies the entire directory.

Generating copy lists during incremental synchronization is more efficient with snapshots than using a regular DistCp update, which can take a long time to scan the whole directory and detect identical files. And because snapshots are read-only point-in-time copies between the source and destination, modification of source files during replication is not an issue, because it can be using other replication methods.

A snapshot cannot be modified. This protects the data against accidental or intentional modification, which is helpful in governance.

Memory cost
There is a memory cost to enable and maintain snapshots. Tracking the modifications that are made relative to a snapshot increases the memory footprint on the NameNode and can therefore stress NameNode memory.

Because of the additional memory requirements, snapshot replication is recommended for situations where you expect to do a lot of directory renaming, if the directory tree is very large, or if you expect changes to be made to source files while replication jobs run.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

HBase snapshots

HBase snapshots are point-in-time backups of the table without making data copies, and with minimal impact on RegionServers. Snapshots are enabled by default. You can create HBase snapshot policies in Replication Manager to take snapshots of a table at regular intervals.

HBase snapshots support enables you to take a snapshot of a table without much impact on RegionServers, because snapshot, clone, and restore operations do not involve data copying. For more information, see HBase snapshots.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Snapshot policies

You can create an HDFS snapshot policy for a snapshottable HDFS directory A directory is *snapshottable* after it has been enabled for snapshots, or because a parent directory is enabled for snapshots in Cloudera Manager. Subdirectories of a snapshottable directory are included in the snapshot. You can also create HBase snapshot policies for HBase tables. Snapshots for HBase tables are enabled by default. Snapshot policies take snapshots of the directory or table at regular intervals.

Enabling and taking snapshots in Cloudera Manager

Before you take snapshots (in Cloudera Manager) or create snapshot policies (in Replication Manager) for HDFS directories, you must enable snapshots for the directories in Cloudera Manager.

1. To enable snapshots for HDFS directories, navigate to the directory on the **Cloudera Manager > Clusters > HDFS service > File Browser** tab, and click **Enable Snapshots**.

   Note: If you enable snapshots for a directory, you cannot enable snapshots for its parent directory. Snapshots can be taken only on directories that have snapshots enabled.

   Tip: To disable snapshots for a directory that has snapshots enabled, click **Disable Snapshots**. Ensure that the snapshots of the directory are deleted before you disable snapshots for the directory.

2. To take a snapshot of a directory or table, perform the following steps:
   a. Navigate to the directory or folder.
   b. Click **Take Snapshot** in the drop-down menu next to the full file path.
   c. Specify a unique name for the snapshot.

   The snapshot name with the timestamp of its creation appears in the snapshot list.

3. Click **Actions > Delete** to delete a snapshot. After you delete a snapshot, you can restore it, if required.

After you finish: You can create a snapshot policy to schedule taking snapshots of the directory at regular intervals in Replication Manager.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Creating snapshot policies in Replication Manager

After you enable snapshots for an HDFS directory in Cloudera Manager, you can create an HDFS snapshot policy to schedule taking snapshots of the directory at regular intervals in Replication Manager. You can also create an HBase snapshot policy to take snapshots of HBase tables at regular intervals.

1. In the Replication Manager, go to the **Snapshots** page.
2. Click **Create Snapshot.**
3. In the **Create Snapshot Policy** wizard, enter or choose the following options.
4. **Source Cluster** - Choose the classic cluster where the directory or table resides for which you want to take a snapshot.
5. **Service** - Choose **HDFS** to choose one or more HDFS directories or **HBase** to choose the required HBase tables.
6. **Snapshot name** - Enter a name for the snapshot which is unique in the selected cluster.

   Important: Ensure that the snapshot policy name neither contains the characters **% . ; / ** nor any character that is not ASCII printable, which includes the ASCII characters less than 32 and the ASCII characters that are greater than or equal to 127.
7. If you choose **HDFS**, enter the following options as required:
   - **HDFS Directory Path** - Choose one or more directory paths. The directory path appears here after you enable snapshots for the directory in Cloudera Manager.
   - **Schedule** - Choose a schedule to take snapshots of the directory. In the **Keep** field, enter the maximum number of snapshots to be retained as currently available snapshots by the Replication Manager. When a snapshot exceeds the maximum number, the least recent snapshot is deleted by Cloudera Manager and the snapshot is marked as an **Expired**. It appears in the **Expired Snapshot** list on the **Snapshots History** tab.
   - **Alerts** - Choose one or more options to generate alerts during various state changes in the replication workflow. You can choose to generate alerts during **On Failure**, **On Start**, **On Success**, and **On Abort** of the replication job.
8. If you choose **HBase**, enter the following options as required:
   - **HBase Table** - Choose one or more tables.
   - **Schedule** - Choose a schedule to take snapshots of the directory. In the **Keep** field, enter the maximum number of snapshots to be retained as currently

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

available snapshots by the Replication Manager. When a snapshot exceeds the maximum number, the least recent snapshot is deleted by Cloudera Manager and the snapshot is marked as an **Expired**. It appears in the **Expired Snapshot** list on the **Snapshots History** tab.

- **Alerts** - Choose one or more options to generate alerts during various state changes in the replication workflow. You can choose to generate alerts during **On Failure**, **On Start**, **On Success**, and **On Abort** of the replication job.

9. Click **Create**.
   The snapshot policy appears on the **Snapshots** page.

Tip: The snapshot policy appears on the **Cloudera Manager > Replication > Snapshot** page as well.

Snapshot policy operations

On the **Snapshots** page, you can edit, suspend, or delete the snapshot policy. You can run the snapshot policy on demand. When you click a snapshot policy, you can view the snapshot history.

Use the **Actions** menu to perform the following policy operations on a snapshot policy:

1. Click **Edit** to make changes to the snapshot policy in the **Edit Snapshot Policy** wizard. The next policy run uses the changed configuration to take snapshots.
2. Click **Suspend** to pause taking snapshots. You can resume taking snapshots when required.
3. Click **Remove** to remove the snapshot policy. Replication Manager removes the snapshot policy and all the snapshots taken by the policy.
4. Click the snapshot name to view the snapshots taken by a snapshot policy on the **Snapshot History** tab.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

View snapshot history

When you click a snapshot on the **Snapshots** page, the **Snapshot History** tab appears. Along with the tab, the page also shows the classic cluster name, service name as HDFS or HBase, the directories, paths, or tables chosen for the snapshot policy, number of times the policy ran, and the snapshot location. You can also perform actions on the snapshot policy or on a specific snapshot on this page.

The following options are available in the **Actions** menu for the snapshot policy:

- **Edit** - Opens the **Edit Snapshot Policy** wizard where you can edit the snapshot as required.
- **Suspend** - Pauses the snapshot policy. You can resume the snapshot policy run when required.
- **Remove** - Removes the snapshot policy and all the snapshots taken by the snapshot policy.

The **Snapshot History** tab shows the list of snapshots, current status, timestamp of the previous snapshot policy run, the configured schedule for the snapshot, and the time the snapshot expires. After the snapshot exceeds the maximum number specified in **Create Snapshot Policy > Keep** field, the least recent snapshot is deleted by Cloudera Manager and the snapshot is marked as **Expired**. It appears in the **Expired Snapshot** list on the **Snapshots History** tab. Use the **Search Snapshot History** field to search for snapshots.

The **Snapshot History** tab lists the currently available snapshots (**Finished** status), next scheduled snapshot policy run (**Scheduled Next** status), suspended snapshots (**Suspended** status), and expired snapshots (appears under the **Expired Snapshots** list).
You can perform the following actions on the currently available snapshots:

- **View Log** - Shows the log details in a window. You can download the log file to your machine as a text file, copy the contents, or open the log window in a new tab.
- **Restore Snapshot** - Opens the **Restore Snapshot Policy** wizard to restore the snapshot to its original location. For more information, see Restoring snapshots.
- **Restore Snapshot As** - Opens the **Restore Snapshot Policy** wizard to restore the snapshot to another location. For more information, see Restoring snapshots.
- **Delete Snapshot** - Deletes the snapshot.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Restoring snapshots in Replication Manager

In Replication Manager, you can choose to restore the data to a version from a previous snapshot. You can also choose to restore it to its original location or to another location.

Before you begin: Ensure you have adequate disk space for restored snapshots in the source cluster.

1. Click the snapshot policy on the **Snapshots** page.
2. On the **Snapshot History** tab, click the **Actions** menu for the required snapshot and choose one of the following options:
    - **Restore Snapshot -** Choose to restore the snapshot to the same directory or table. This action overwrites existing data.
    - **Restore Snapshot As** - Choose to restore the snapshot to another specified directory or table. If the specified directory or table exists or has similar data, this action overwrites it.
3. In the **Restore Snapshot** wizard, enter the following details:
4. **Source Cluster** - Shows the source cluster of the snapshot.
5. **Service** - Shows the service for the snapshot.
6. **Recover Data from Snapshot** - Shows the timestamp of the scheduled snapshot policy.
7. The following options appear for HDFS snapshots:
    - **Directory path to restore** - Shows the HDFS directory of the snapshot that is to be restored.
    - **The new directory path** - Enter the new location to save the snapshot. This option appears when you choose **Restore Snapshot As.**

    Caution: If the directory or table entered already exists, that directory is overwritten.
8. The following options appear for HBase snapshots:
    - **Table to restore** - Shows the table to restore.
    - **Restore as** - Enter a name of the table to which the snapshot is to be restored. This option appears when you choose **Restore Snapshot As.**

    Caution: If the directory or table entered already exists, that directory is overwritten.
9. **Use the HDFS 'copy' command** - Choose this option to copy the contents of the snapshot as a subdirectory or as files in the target directory. This restore option takes time to complete and does not require credentials in a secure cluster.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

10. **Use DistCp / MapReduce -** Choose this option to merge the target directory with the contents of the source snapshot. This restore operation is faster and requires credentials (**Run As Username**) in secure clusters.

> **Important:** Before you restore HDFS snapshots using DistCp, ensure that the user name specified in the **Run as Username** in the HDFS snapshot policy is part of the superuser user-group.
>
> To add a user to the supergroup user-group, run the following commands:
>
> ```
> sudo adduser [***username***]
> sudo usermod -a -G hdfs [***username***]
> sudo usermod -a -G hadoop [***username***]
> id -Gn [***username***]
> kinit -kt /cdep/keytabs/hdfs.keytab hdfs
> hdfs dfs -mkdir /user/[***username***]
> hdfs dfs -chown
> [***username***]:[***username***]/user/[***username***]
> sudo groupadd supergroup
> sudo usermod -a -G supergroup [***username***]
> hdfs groups [***username***]
> ```

Choose the following options as required:

Option	Description
MapReduce Service	Choose a MapReduce or YARN service in the cluster. DistributedCopy (distcp) restores the directories and it increases the speed of restoration.
Scheduler Pool	Enter the name of a resource pool in the field. The value you enter is used by the MapReduce Service you specified when Cloudera Manager runs the MapReduce job for the replication. The job specifies the value using If a MapReduce or YARN service is not available, a normal copy is performed.one of these properties: ● MapReduce – Fair scheduler: **mapred.fairscheduler.pool** ● MapReduce – Capacity scheduler: **queue.name** ● YARN – **mapreduce.job.queuename**

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Run as Username	Uses the default username. Enter the username for kerberized clusters.
Log Path	Enter an alternate path for the log files.
Maximum Map Slots	The maximum number of map tasks (simultaneous copies) per replication job. The default value is 20.
Maximum Bandwidth	The maximum bandwidth in MB for a mapper. Each map task is throttled to consume only the specified bandwidth such that the net used bandwidth tends towards the specified value. The default is 100 MB.
Replication Strategy	Choose **Static** to distribute the file replication tasks among the mappers statically. Static replication distributes the file replication tasks among the mappers up front to achieve a uniform distribution based on the file sizes. **Dynamic** replication distributes the file replication tasks in small sets to the mappers, and as each mapper completes its tasks, it dynamically acquires and processes the next unallocated set of tasks. The default replication strategy is **Dynamic**.
Delete Policy	You can choose how to handle the replicated data in the destination cluster if the files are deleted on the source and also determine the handling of files in the destination location that are unrelated to the source.
Preserve	You can choose to preserve the block size, replication count, permissions (including ACLs), and extended attributes (XAttrs) as they exist on the source file system, or use the settings as configured on the destination file system.
Abort on Error	Aborts the restore job if an error appears.
Skip Checksum Checks	Does not validate the checksum checks on the copied files. By default, checksums are checked for the copied files.

11. Click **Restore**.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Support for CDP Private Cloud Base Clusters in Data Catalog (Preview)

Date published: 2022-02-24
Date modified: 2022-04-06

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 1

Contents 3

Support for CDP Private Cloud Base Clusters in Data Catalog 4

Pre-requisites 4

Exploring CDP Private Cloud Base cluster in Data Catalog 5

Support for CDP Private Cloud Base Clusters in Data Catalog

Data Catalog now supports discovering assets that reside in CDP Private Cloud Base clusters. With this functionality, you can now discover assets not just in your Public Cloud data lakes, but also your Private Cloud Base clusters from a single location inside the Data Catalog service. This enables hybrid cloud capabilities in the CDP platform. For more information, see CDP Private Cloud Base clusters.

Atlas and Ranger services must be available on the registered CDP Private Cloud Base clusters. Data Catalog queries Atlas and Ranger running on these clusters. The **Asset Details** page provides information about the profiled data. To use this feature, make sure that your account(s) have been granted the required entitlements.

Pre-requisites

Make sure to note the following:

- Make sure you use **7.1.4 or higher** versions of CDP Private Cloud Base clusters. For more information, see Release Summaries.

- Register your CDP Private Cloud Base clusters on the **Management Console** using the Classic Clusters registration procedure. For more information, see Adding CDP Private Cloud Base cluster for use in Replication Manager and Data Catalog.

- Make sure that your account has been granted the entitlements -> **CLASSIC_CLUSTERS_CDPDC_SSO** and **DATA_CATALOG_ENABLE_CDP_PRIVATE_CLOUD_BASE.** Once the entitlements are enabled, you must be able to explore all the registered CDP Private Cloud Base clusters as a logged in user. Contact your Cloudera account representative to activate this feature in your CDP environment.

- In the CDP Private Cloud Base cluster configuration, note that attribute used to sync users in Ranger and the attribute that is mapped to **idpUserId** while configuring SAML with an identity provider must be the same.

Note: Since the profiler support is not yet available on Classic Clusters, data can be browsed without running any profilers.

Exploring CDP Private Cloud Base cluster in Data Catalog

Data Catalog is a service within Cloudera Data Platform that enables you to understand, manage, secure, and govern data assets across the enterprise. Data Catalog helps you understand data across multiple clusters and across multiple CDP environments. You can search to locate relevant data of interest based on various parameters. Using Data Catalog, you can understand how data is interpreted for use, how it is created and modified, and how data access is secured and protected.

Important: There is no automatic login access to Atlas and Ranger services from Data Catalog for Private Cloud Base cluster. Clicking on the Atlas or Ranger link from Data Catalog will prompt you to enter login credentials. Use LDAP user credentials to proceed further.

By selecting the CDP Private Cloud Base cluster as a data lake, you can:

- Discover data assets in the data lake. For more information, see Search for Assets.
- Asset Detail page functionality
- Tag management functionality

Supporting High Availability for Profiler services (Preview)

Date published: 2021-08-07
Date modified: 2021-08-07

Legal Notice

© Cloudera Inc. 2021. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2

Contents 3

Supporting High Availability for Profiler services 4

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Supporting High Availability for Profiler services

The Profiler Services is now supported by enabling the High Availability (HA) feature.

Based on the entitlement, the HA functionality is supported on the Profiler cluster.

Note: This feature is under entitlement. Contact your Cloudera account representative to activate this feature in your CDP environment.

There are two types of Profiler Services:

- **Profiler Manager**
- **Profiler Scheduler**

The **Profiler Manager** service consists of Profiler administrators, metrics, and data discovery services. These three entities support HA. The HA feature supports **Active-Active** mode.

Attention: The **Profiler Scheduler** service does not support the HA functionality.

As of this update, there is a new user interface which is introduced to launch profilers in Data Catalog.

On the Data Catalog search page, select the data lake from which you want to launch the profiler cluster. On the right-hand side of the window, the application displays the page to set up the profiler for the selected data lake. Click the **Get Started** link to proceed.

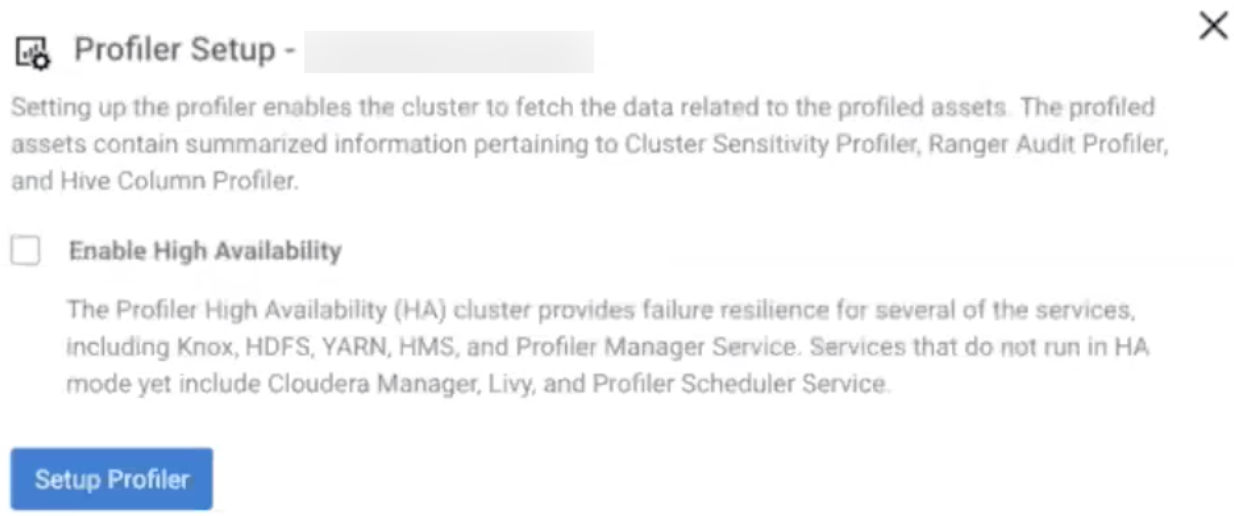

For setting up the profiler, you have the option to enable or disable the HA. Note that the HA functionality is being supported only from **Cloudera Runtime 7.2.10** release onwards. If you are

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

using the Cloudera **Runtime** version below 7.2.10, you shall not be able to use the HA feature for launching the profiler services.

Once you enable HA and click **Setup Profiler**, Data Catalog processes the request and the profiler creation is in progress.

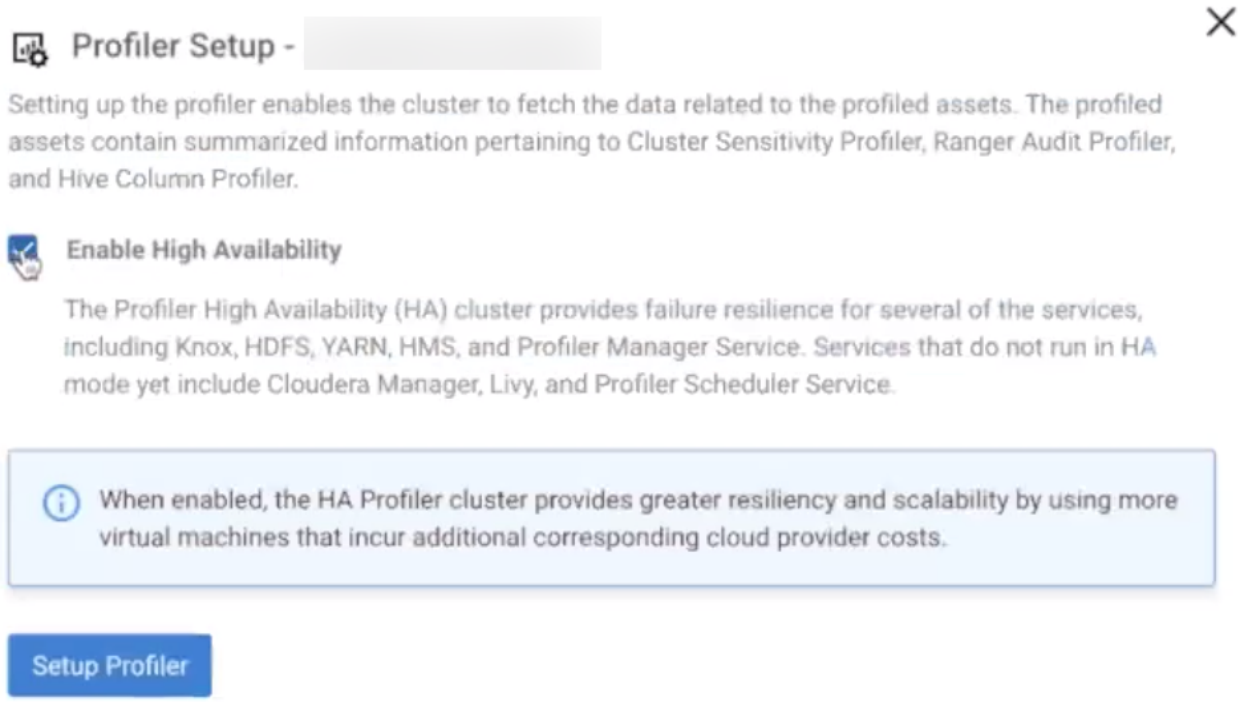

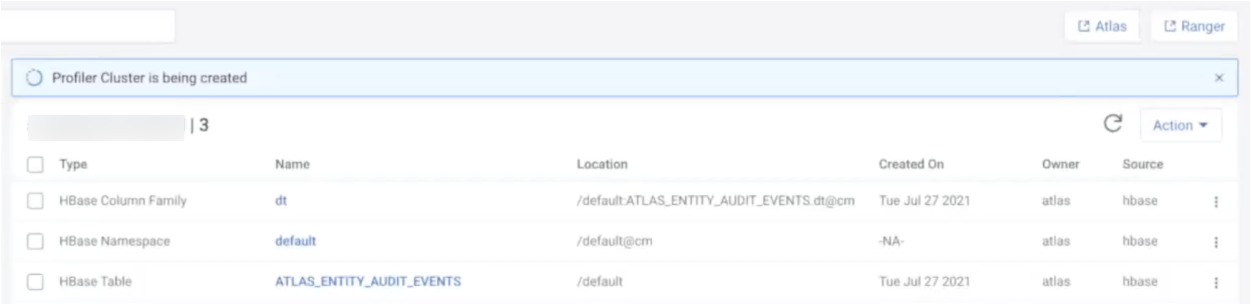

Later, a confirmation message appears that the profiler cluster is created.

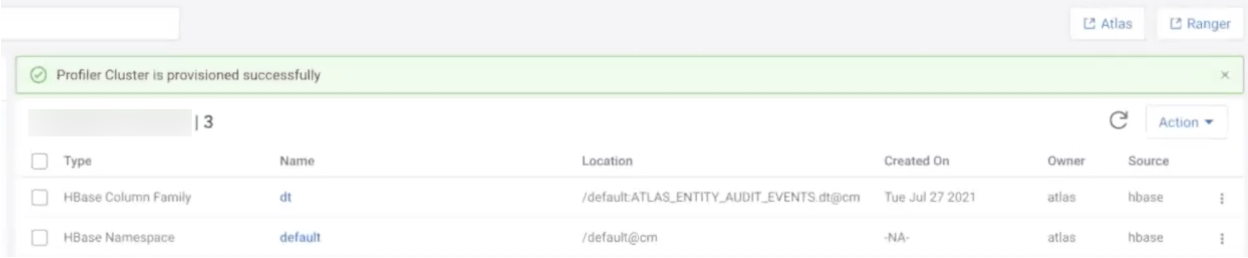

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Next, you can verify the profiler cluster creation under **CDP Management Console** > **Environments** > **DataHubs** pane.

Note that the newly created profiler cluster has some unique representations under the following categories:

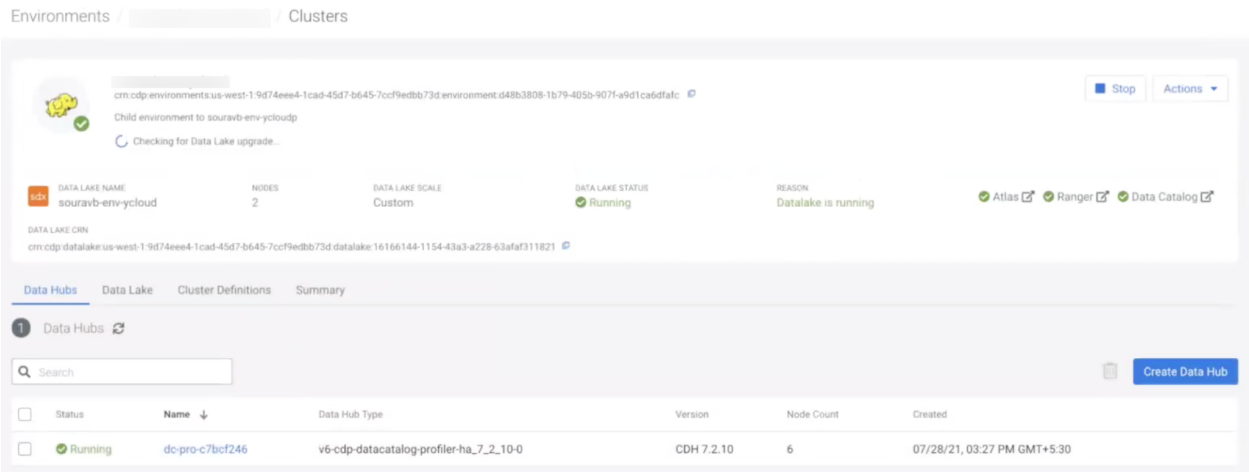

- **Data Hub Type** - The term "**ha**" is appended to the type of cluster that is newly created.
- **Version** - 7.2.10
- **Node Count** - (Which is **6**)

Your Profiler cluster with HA is set up successfully.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Transitioning Profiler Manager Service into SDX (Preview)

Date published: 2022-02-24

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Legal Notice 2

Contents 3

Transitioning Profiler Manager Service into SDX Infrastructure 4

Known limitations 4

Transitioning Profiler Manager Service into SDX Infrastructure

The current Data Catalog architecture includes profilers is a separate DataHub cluster that comprises of two types of components:

1. Profiler Manager Service (Profiler Admin/Profiler Metrics/Data Discovery Service) [PMS]
2. Profiler Scheduler
    - Profiler Services (Admin, Metrics, and Scheduler)
    - Profiler jobs - The Profiler services (Admin and Metrics) are required for serving user interactions from the Data Catalog app. This implies the profiler cluster cannot be shutdown without loss of service to users.

    Considering the above scenarios, the PMS is moved to the SDX infrastructure which results in retaining the scheduler and profiler jobs in a separate DataHub cluster.

These changes have the following benefits:

- Allows the user to completely shutdown the profiler cluster if needed to save time and cost.
- Allows the user to handle other platform capabilities like upgrade, High Availability and other pertinent features in a consistent manner for services needed by the Data Catalog App.

Known limitations

Some of the known limitations seen during the transition:

- Profiler Manager service on SDX is only supported in medium duty and it is behind an entitlement "**CDP_DATA_LAKE_MEDIUM_DUTY_WITH_PROFILER**".

- If the entitlement is enabled and the user starts using the MEDIUM duty SDX cluster, Data Catalog creates a profiler cluster without Profiler Manager Service. Else, as usual, Profiler Manager Service would be created on the DataHub cluster.

Upgrading from CCMv1 to CCMv2 (Preview)

Date published: 2023-04-17
Date modified: 2023-04-18

Legal Notice

© Cloudera Inc. 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms.

Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Upgrading from CCMv1 to CCMv2

Upgrading from CCMv1 to CCMv2 is available for customers who created their environments with CCMv1. The upgrade requires no downtime.

Both CCMv1 and CCMv2 provide the same functionality. However, as with all software, version 2 is an improvement over version 1: CCMv2 is more efficient in communicating with the CDP Control Plane and has a smaller footprint on the customer network than CCMv1.

The differences between CCMv2 and CCMv1 are related to improving reliability and efficiency, These differences are as follows:

- CCMv2 uses fewer agents (its agents are on FreeIPA nodes only), whereas CCMv1 has a larger footprint, with its agents running across not only on FreeIPA nodes but also on Data Lake and Data Hub nodes.
- CCMv2 uses the HTTPS protocol. In contrast, CCMv1 is based on the SSH protocol.
- Since CCMv2 uses HTTPS, it is more proxy-friendly. Customers can set up a proxy server such as Squid and inspect the underlying traffic to be more confident about the data that is being sent to the CDP Control Plane.
- CCMv2 supports token authentication.
- Newer Cloudera services, such as Machine Learning, Data Engineering, DataFlow, Data Warehouse, and others, only support CCMv2.

Prerequisites

1. To perform the upgrade, your account must have the following entitlements granted:

- CDP_CCM_V1_TO_V2_JUMPGATE_UPGRADE
- CDP_CCM_V2_TO_V2_JUMPGATE_UPGRADE

   Once the entitlements have been granted, you can upgrade to the latest FreeIPA image if needed and then trigger a CCM upgrade from CDP web interface.

2. Install Beta CDP CLI. See Installing Beta CDP CLI.

Steps

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

1. Check the version of CCM that you are currently on, you can run the following command using Beta CDP CLI after replacing `<ENVIRONMENT-NAME>` with an actual value:
   ```
   cdp environments describe-environment
   --environment-name=<ENVIRONMENT-NAME> --output=json | yq -r
   '.environment.tunnelType'
   ```

   CDP will return one of the following values:

   - CCM (if CCMv1 is installed)
   - CCMV2 (if CCMv2 is installed)

   If your CCM version is CCM, you should upgrade to CCMv2.

2. Prior to upgrading CCM, you should ensure that you are using the latest FreeIPA image. To upgrade your FreeIPA, follow the steps described in Upgrade FreeIPA.
3. Upgrade CCM either via CDP web interface or CLI:
   a. UI: Navigate to the **Management Console > Environments** and click on the tile corresponding to a specific environment. Trigger the upgrade by clicking on the **Upgrade** button:

   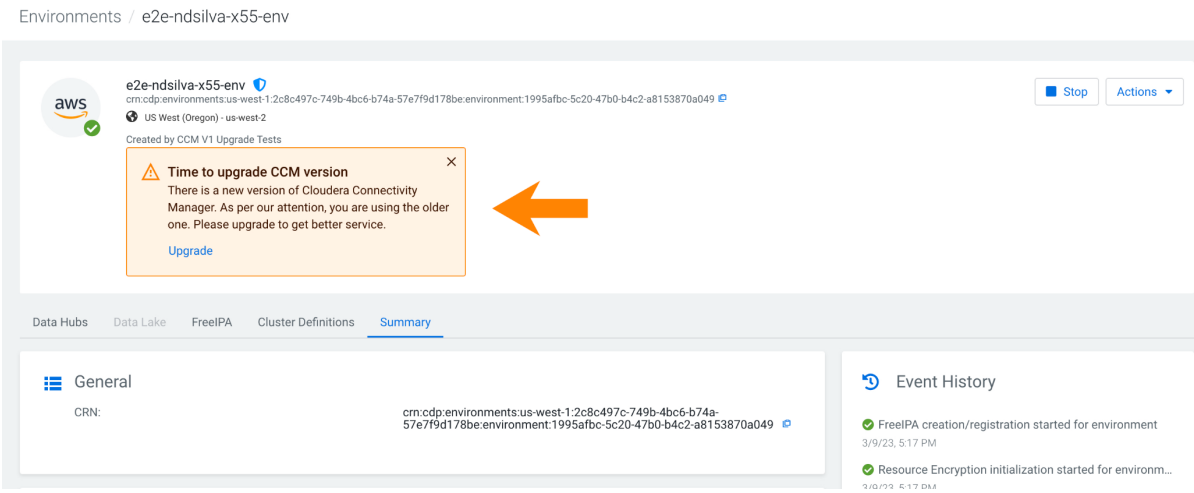

   b. Beta CDP CLI: Use the following command:

   ```
   cdp environments upgrade-ccm --environment
   <ENV_NAME_OR_CRN>
   ```

4. The status of your environment changes while the upgrade is pending. All environment components get upgraded (FreeIPA, Data Lake, and then Data Hubs). Check the event history to confirm that the upgrade has been completed. Once the upgrade has been completed, the status of the environment changes

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

back to “Running”. If an error occurs during the process, CDP rolls back the upgrade and reverts the environment to CCMv1.

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Cloudera Runtime 1.0.0

Administering Hue

Date published: 2021-12-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Hue configurations in Cloudera Data Warehouse 4

Hue supported browsers 6

Customizing the Hue web interface 6
- Adding a custom banner in Hue 6
- Changing the page logo in Hue 6
- Adding a splash screen in Hue 8
- Setting the cache timeout 9

Disabling the share option in Hue 9

Configuring idle session timeout for Hue 10

Adding Query Processor admin users and groups 10

Ways to clean up old queries from the Query Processor tables 11

Downloading debug bundles 12

Enabling Spark 3 engine in Hue 13

About the Hue task server in Cloudera Data Warehouse 15
- Scheduling document cleanup 15
- Scheduling tmp cleanup 16
- Advanced Configuration Snippet for Hue task server 16

Hue configurations in Cloudera Data Warehouse

You can customize the settings for Hue at a Vitrual Warehouse level by specifying the configurations in the hue-safety-valve field in the Cloudera Data Warehouse (CDW) UI.

The following table lists the safety valve parameters supported by Hue in Cloudera Data Warehouse:

Parameter	Description

`[notebook]`
`  [[interpreters]]`
`  [[[hive]]]`
`  name=Hive`
`  interface=hiveserver2`
`  [[[hplsql]]]`
`  name=Hplsql`
`  interface=hiveserver2`
	Used to activate and enable switching between Hive and HPL/SQL interpreters for your queries.
	Note: Hue enables you to switch between Hive and HPL/SQL interpreters. By default, the Hive interpreter is enabled when you create a Hive Virtual Warehouse. To enable the HPL/SQL interpreter, you must update the configuration in the hue-safety-valve field in your Hive Virtual Warehouse. However, updating hue-safety-valve overrides the default configuration. Therefore, to use both Hive and HPL/SQL interpreters, you must enable both by updating the configuration in the hue-safety-valve field.

`[[desktop]]` `app_blacklist`	Used to add or remove applications, such as the File Browser, Impala, Hive, Oozie, and so on from the blocked list of applications.

`[desktop]`
`  [[session]]`
`  ttl=[***NUMBER-OF-SECONDS***]`
	Used to configure the duration of a user session. The ttl property determines the length of time that the cookie with the user's session ID lives before expiring. After the ttl setting is reached, the user's session expires whether it is active or not.
	The default setting for ttl is 1,209,600 seconds, which equals two weeks.

`[jobbrowser]`
`[[query_store]]`
`server_url=[***QUERY-PROCESSOR-URL***]`
	Used to display the **Queries** tab for Hive and Impala on the **Job Browser** page. This configuration is enabled by default and is not exposed in the Hue safety valve.
	However, to hide the **Queries** tab, you can override the configuration by adding the following lines in Virtual Warehosue ⋮ Edit CONFIGURATIONS Hue hue-safety-valve :
	`[jobbrowser]`
	`[[query_store]]`
	`is_enabled=false`
	To enable the **Queries** tab, set is_enabled to true.

`[aws]`
`[[aws_accounts]]`
`[[[default]]]`
`access_key_id=[***AWS-ACCESS-KEY***]`
`secret_access_key=[***SECRET-ACCESS-KEY***]`
`region=[***AWS-REGION***]`
	(Non-RAZ) Used to enable the S3 File Browser for Hue in Cloudera Data Warehouse to access an S3 bucket from Hue.

`[azure]`
`  [[azure_accounts]]`
`  [[[default]]]`
`  client_id=[***AZURE-ACCOUNT-CLIENT-ID***]`
`  client_secret=[***AZURE-ACCOUNT-CLIENT-SECRET***]`
`  tenant_id=[***AZURE-ACCOUNT-TENANT-ID***]`
`  [[abfs_clusters]]`
`  [[[default]]]`
`  fs_defaultfs=abfs://[***CONTAINER-NAME***]@[***AZURE-STORAGE-ACCOUNT-NAME***]>.dfs.core.windows.net`
`  webhdfs_url=https://[***AZURE-STORAGE-ACCOUNT-NAME***].dfs.core.windows.net/`
	(Non-RAZ) Used to enable the ABFS File Browser for Hue in Cloudera Data Warehouse to access ADLS Gen2 containers.

Hue supported browsers

Hue works with the two most recent LTS (long term support) or ESR (extended support release) browsers. Cookies and JavaScript must be enabled.

The lists the minimum tested versions of the most common browsers:

- Chrome: (Version history)
- Firefox: (Version history)
- Safari (Mac only): Version history
- Microsoft Edge: (Version history)

Hue can display in other browsers and in older versions of the common browsers, but you might not have access to all features.

Customizing the Hue web interface

You can customize the page logo and set the cache timeout value by configuring the parameters in the Virtual Warehouse which is running Hue.

Adding a custom banner in Hue

You can add a custom banner to the Hue web interface by adding your custom HTML to the hue-safety-valve configuration for your Virtual Warehouse.

Procedure

1. Log in to the Data Warehouse service as DWAdmin.
2. Go to your Virtual Warehouse tile and click ⋮ Edit .
3. Go to CONFIGURATIONS Hue , select hue-safety-valve from the Configuration files drop-down menu and add the following lines:

```
[desktop]
[[custom]]
banner_top_html=<H1>Your company's custom Hue Web UI banner</H1>
```

4. Click Apply Changes.

   The Virtual Warehouse goes into an "Updating" state. Wait for the update to complete, and then open or refresh Hue.

Changing the page logo in Hue

You can replace the Hue web interface logo with a custom log that is created with an SVG code. Add any type of logo you want, but your custom logo should be designed to fit into a 160 x 40 pixel space.

About this task

For example, here is the Hue logo shown in the following image:

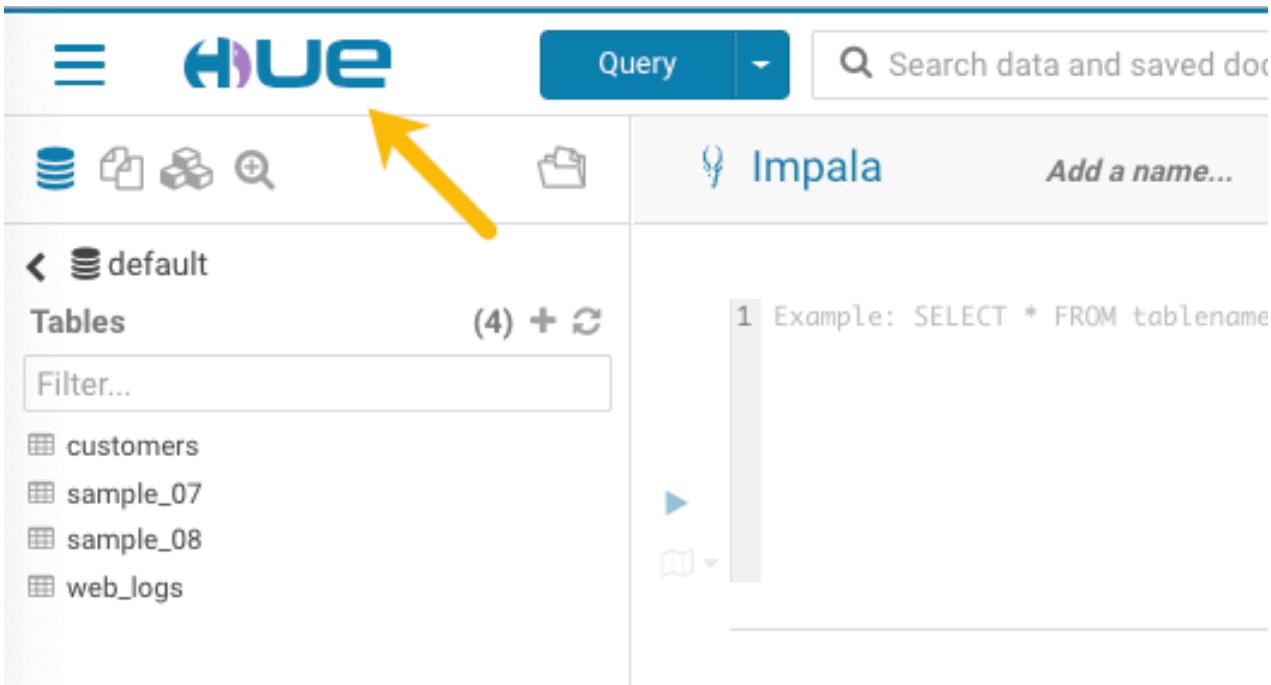

You can change this Hue logo by adding the appropriate SVG code to the logo_svg property under the [desktop] [[custom]] section in the hue_safety_valve configuration parameter in Cloudera Data Warehouse.

Procedure

1. Log in to the Cloudera Data Warehouse service as an administrator.
2. Go to the Virtual Warehouses Edit CONFIGURATIONS Hue and select hue-safety-valve from the Configuration files drop-down list.
3. Add the custom logo SVG code in the [desktop] [[custom]] section as shown in the following example:

```
[desktop]
[[custom]]
logo_svg='[***SVG-CODE-FOR-CUSTOM-LOGO***]'
```

For example, the following SVG code replaces the Hue logo with a red heart:

```
[desktop]
[[custom]]
logo_svg='<g><path stroke="null" id="svg_1" d="m44.41215,11.43463c-4.05
017,-10.71473
-17.19753,-5.90773 -18.41353,-0.5567c-1.672,-5.70253 -14.497,-9.95663
-18.411,0.5643c-4.35797,11.71793 16.891,22.23443 18.41163,23.95773c1.518
1,-1.36927 22.7696,-12.43803
18.4129,-23.96533z" fill="#ffffff"/> <path stroke="null" id="svg_2"
d="m98.41246,10.43463c-4.05016,-10.71473 -17.19753,-5.90773 -18.41353,-
0.5567c-1.672,-5.70253
-14.497,-9.95663 -18.411,0.5643c-4.35796,11.71793 16.891,22.23443 18.4116
4,23.95773c1.5181,-1.36927
22.76959,-12.43803 18.41289,-23.96533z" fill="#FF5A79"/> <path stroke="nu
ll" id="svg_3"
d="m154.41215,11.43463c-4.05016,-10.71473 -17.19753,-5.90773 -18.41353,-0
.5567c-1.672,-5.70253
-14.497,-9.95663 -18.411,0.5643c-4.35796,11.71793 16.891,22.23443 18.41164
,23.95773c1.5181,-1.36927 22.76959,-12.43803 18.41289,-23.96533z" fill="
#ffffff"/> </g>'
```

4. Click APPLY.
5. Restart the Virtual Warehouse.
6. Verify your changes by opening Hue.

   If you added the sample SVG code that defines a red heart as the logo, then your Hue web interface looks as shown in the following image:

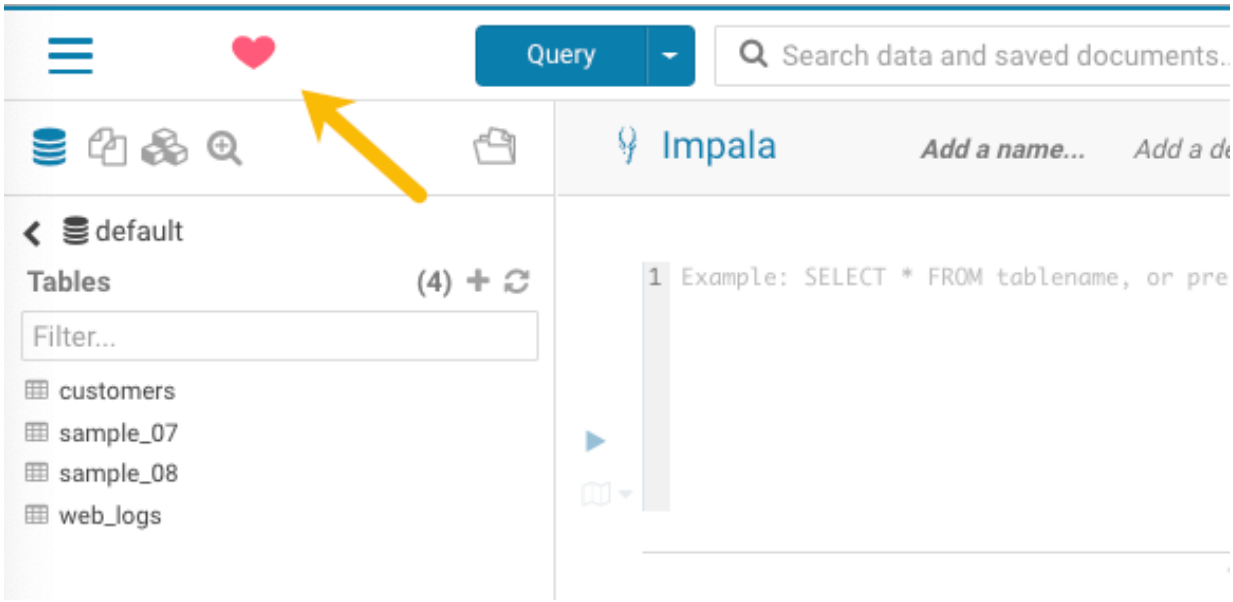

Adding a splash screen in Hue

You can add a custom splash screen to the Hue web interface by adding your custom HTML to the hue-safety-valve configuration for your Virtual Warehouse.

Procedure

1. Log in to the Cloudera Data Warehouse as DWAdmin.
2. Go to your Virtual Warehouse tile and click ⋮ Edit .
3. Go to CONFIGURATIONS Hue , select hue-safety-valve from the Configuration files drop-down menu and add the following lines:

```
[desktop]
[[custom]]
login_splash_html=[***CUSTOM-HTML***]
```

```
[desktop]
[[custom]]
login_splash_html=<h1>Hue, the next-gen SQL Assistant</h1>
```

4. Click Apply Changes.

   The Virtual Warehouse goes into an "Updating" state. Wait for the update to complete, and then open or refresh Hue.

Setting the cache timeout

Enable Hue UI caching by setting a timeout value in milliseconds. The default is 10 days or 864000000 milliseconds. Set the timeout to 0 to disable caching.

About this task

When you browse tables using the left assist panel or run queries, Hue caches this information for fetching information faster and query autocompletion. You can configure the time for which you want to Hue to cache this information by setting the value of the cacheable_ttl property under the [desktop][[custom]] section in the hue_safety_valve configuration property in Cloudera Data Warehouse.

Procedure

1.
2. Log in to the Cloudera Data Warehouse service as an administrator.
3. Go to the Virtual Warehouses ⋮ Edit CONFIGURATIONS Hue and select hue-safety-valve from the Configuration files dropdown menu.
4. Add the following parameters with the cache timeout value to the hue_safety_valve configuration parameter:

```
[desktop]
[[custom]]
cacheable_ttl=[***VALUE-IN-MILLISECONDS***]
```

   For example, the following configuration sets the cache timeout to the default value of 86400000 milliseconds:

```
[desktop]
[[custom]]
cacheable_ttl=86400000
```

5. Click APPLY.
6. Restart the Virtual Warehouse.

Disabling the share option in Hue

Hue allows you to share documents, queries, and workflows with other users, either for viewing only or viewing and modifying in any Hue instances across all Virtual Warehouses within a Database Catalog. Sharing is enabled by default in the Hue UI. For added privacy and control, you can disable sharing by setting the enable_sharing property to false in the hue-safety-valve.

About this task

The sharing option is always available to the admin users. To disable the share option:

Procedure

1. Log in to the Cloudera Data Warehouse service as an administrator.
2. Go to the Virtual Warehouses Edit CONFIGURATIONS Hue and select hue-safety-valve from the Configuration files drop-down list.
3. Add the following lines in the hue-safety-valve:

```
[desktop]
enable_sharing=false
```

4. Click APPLY.

5. Restart the Virtual Warehouse.

Configuring idle session timeout for Hue

By configuring idle session timeout, you can prevent unauthorized access to your data by automatically logging out users when they are not actively using the Hue web interface. Hue warns the users and then logs them out after a set period of inactivity that you specify in the Hue configuration.

About this task

By default, the value of the idle_session_timeout property is set to “-1”, which means that the users are not logged out until they manually log out of Hue.

Procedure

1. Log in to the Cloudera Data Warehouse service as an administrator.
2. Go to the Virtual Warehouse in which Hue is running and click Edit.
3. Go to CONFIGURATIONS Hue and select hue-safety-valve from the Configuration files drop-down list and add the following lines:

```
[desktop]
[[auth]]
idle_session_timeout=[***TIME-IN-SECONDS***]
```

4. Click APPLY.

   The Virtual Warehouse restarts to update the configuration.

Adding Query Processor Administrator users and groups in Cloudera Data Warehouse

The Query Processor Administrators have special privileges that enable them to view and monitor queries from all users, including the ones that were submitted from query interfaces, such as Beeline, Hive Warehouse Connector (HWC), Tableau, Impala-shell, Impyla, and other JDBC/ODBC clients.

Before you begin

Make sure that the Virtual Warehouse to which you want to add the Hue Query Processor Administrators users is in the stopped state.

Procedure

1. Log in to the Cloudera Data Warehouse web interface as a DWAdmin.
2. Click ⋮ Edit on the Database Catalog for which you want to add Hue Query Processor Administrators users.

3. On the **Database Catalog** details page, click CONFIGURATIONS Hue query processor and select hue-query-processor.json from the Configuration files drop-down menu, as shown in the following image:

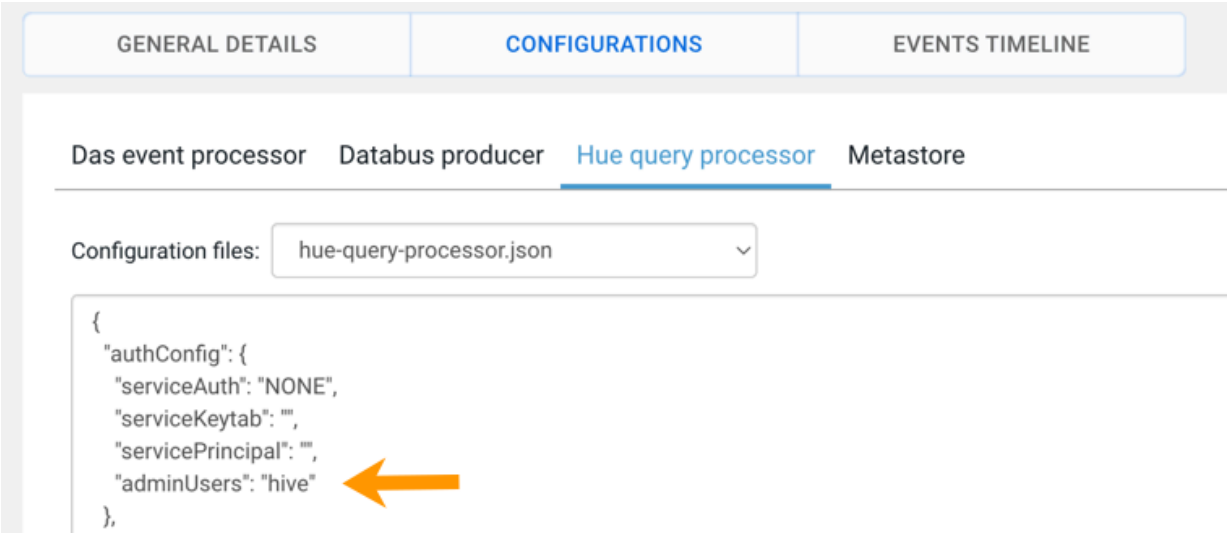

4. In the “authConfig” section, add the list of users to the “adminUsers” key.

   For example: "adminUsers": "hive, [***USER-1***], [***USER-2***]"

   You can also add a list of admin groups as follows:

```
"adminGroups": "admin-group, [***GROUP-1***], [***GROUP-2***]"
```

5. Click Apply.

   The Hue service will be unavailable for approximately 5 minutes to make the update.

Ways to clean up old queries from the Query Processor tables

Learn how to schedule a query clean-up and how to use the API to manually clean up queries from the following Query Processor tables: vertex_info, dag_details, dag_info, query_details, hive_query, tez_app_info.

Scheduling query clean-up

Both Hive and Impala queries are retained in the backend database for 30 days by default, after which they are cleaned up. You can change the clean-up interval from the Database Catalog configurations. Go to Database Catalogs ⋮ Edit CONFIGURATIONS Hive query processor and select the hue-query-processor.json from the Configuration files drop-down menu.

Add the following line under the dasConf section and specify the time interval in seconds:

```
 "dasConf": {
    "hue.query-processor.event-pipeline.cleanup-interval-secs": "[***TIME-
INTERVAL-IN-SECONDS***]",
    “hue.query-processor.event-pipeline.cleanup.cron.expression” : “[***CR
ON-EXPRESSION***]”
  },
```

For example:

```
“hue.query-processor.event-pipeline.cleanup.cron.expression” : “0 0 2 * * ?”
,
```

```
“hue.query-processor.event-pipeline.cleanup-interval-secs” : “2592000”
```

Manually cleaning up queries using an API

The ability to clean up queries manually in addition to the scheduled clean-up routines is useful when you have a high load of queries in a particular week that are hogging resources that you must free up. The API also runs a VACUUM command on the Query Processor table to reclaim storage that is occupied by dead tuples.

You can send an API request using tools such as cURL or Postman.

API format: `[***QUERY-PROCESSOR-ADDRESS***]/api/admin/cleanup/[***EPOCH-TIME***]`

Where,

- [***QUERY-PROCESSOR-ADDRESS***] is the query processor host address
- [***EPOCH-TIME***] is the Unix epoch time in seconds

Queries that were run before the specified epoch time are purged.

For example:

```
curl "http://machine1.company.com:30700/api/admin/cleanup/1670006742"
```

Downloading debug bundles

The debug bundle is a ZIP file that contains the query details in JSON format and an error-reports.json file, which is created only if an error occurs while the query is run.

About this task

Note: This feature is available only for Hive queries.

If Tez is used to run a query, then the debug bundle also contains DAG and Tez JSON files, as shown in the following image:

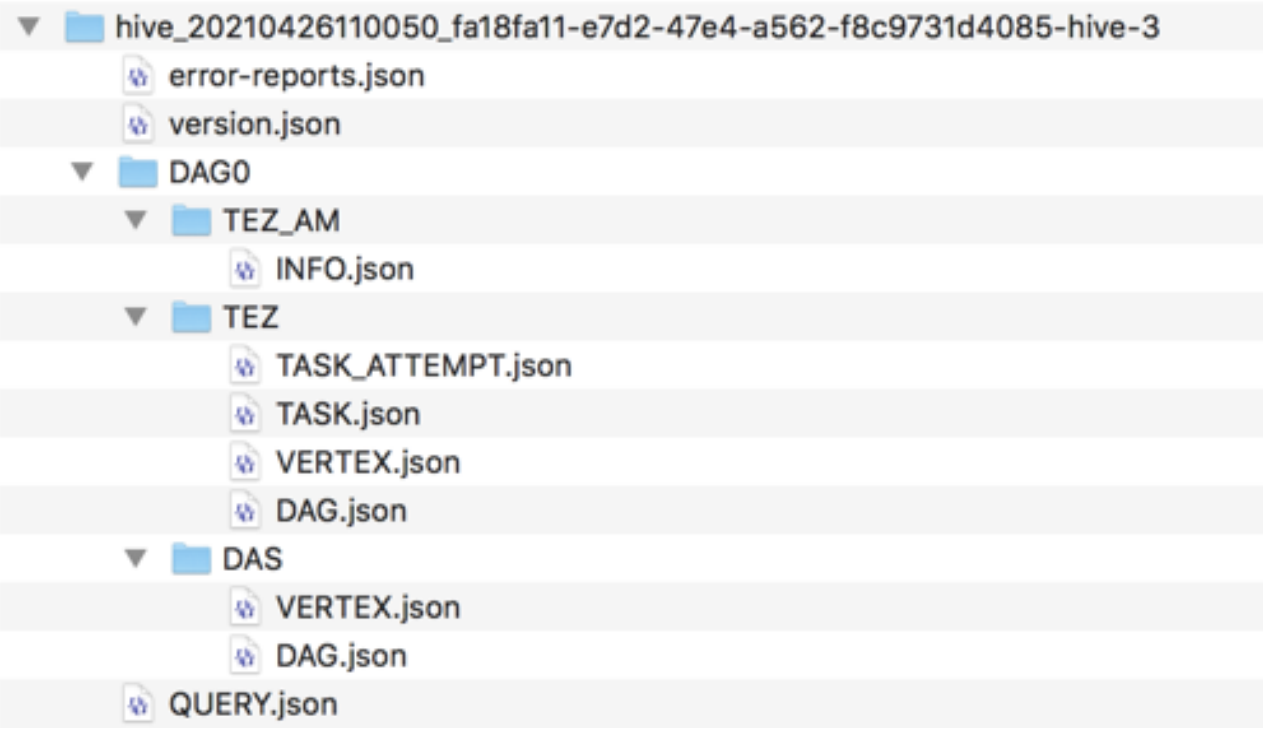

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.
   The **Job Browser** page is displayed.
3. Click Queries.
   The Hive queries that were run are displayed.
4. Select a query for which you want to download the debug bundle.
5. Click Download and save the ZIP file on your computer.
   The filename is in the following format:

```
hive_[***HIVE-QUERY-ID***]_[***USER-ID***]_[***UNIQUE-INDEX***]
```

Enabling Spark 3 engine in Hue

Hue leverages Apache Livy 3 to support Spark SQL queries in Hue on the Apache Spark 3 engine. To enable the Spark 3 engine, specify the Livy server URL in the Hue Advanced Configuration Snippet using Cloudera Manager, and enable the Spark SQL notebook. Livy for Spark 3 and Spark 3 services are installed when you create a Cloudera Data Hub cluster with the Cloudera Data Engineering cluster template.

Before you begin

Note: Livy v0.6 supports Python versions upto Python 3.7. If you install Python 3.8 or higher, then you may see the following error: TypeError: required field "type_ignores" missing from Module.

Procedure

1. Log in to Cloudera Manager as an Administrator.

2. Go to Clusters HDFS Configuration and add the following lines in the Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml field:

```
<property>
    <name>hadoop.proxyuser.hue.groups</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.hue.hosts</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.spark.groups</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.spark.hosts</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.livy.groups</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.livy.hosts</name>
    <value>*</value>
</property>
```

3. Click Save Changes.
4. Go to Clusters Livy for Spark 3 service Configuration and add the following configurations:
   a) Add the hue user in the Admin Users (livy.superusers) field.
   b) Go to the HMS Service field and select Hive.
   c) Click Save Changes.
5. Go to Clusters SPARK_ON_YARN Configuration Admin Users , add hue to the list of admin users (spark.history.ui.admin.acls) and click Save Changes.
6. Go to Clusters SPARK Configuration Admin Users , add hue to the list of admin users (spark.history.ui.admin.acls) and click Save Changes.
7. Go to Clusters SPARK 3 Configuration Admin Users , add hue to the list of admin users (spark.history.ui.admin.acls) and click Save Changes.
8. Go to Clusters Hue Configuration and enter the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field and click Save Changes:

```
[desktop]
app_blacklist=zookeeper, pig #custom list of blocked apps
[spark]
#This is not a thrift server port
#If this TLS/SSL is enabled then check to see whether the livy url is on h
ttps or http and modify the url accordingly.
livy_server_url=http(s)://[***LIVY-FOR-SPARK3-SERVER-HOST***]:[***LIVY-
FOR-SPARK3-SERVER-PORT***]
ssl_cert_ca_verify=false
security_enabled=true
[notebook]
[[interpreters]]
[[[sparksql]]]
name=Spark SQL
```

```
interface=livy
```

Attention:

- Ensure that the Spark application is not on the blocked list.
- Set ssl_cert_ca_verify=false if an SSL certificate is not present in Hue's truststore. ssl_cert_ca_verify=true if your environment is a secured environment.
- Set security_enabled=true for Kerberized clusters.

9. Restart the affected services.

Results

You can now select the “Spark SQL” dialect on the Hue editor and run Spark queries from Hue.

Note: Starting a Livy session can take 30-45 seconds. If you cancel a running Spark SQL query and rerun it, the Livy session expires, and a new session is created.

About the Hue task server in Cloudera Data Warehouse

The Hue task server helps you schedule tasks to clean up Hue documents and the /tmp directory, improving cluster maintenance experience and performance. The task server also improves the file-upload experience, allowing you to upload larger files up to 5 GB.

The task server is enabled by default. Hue superusers or administrators can schedule cleanup jobs from the **Task Server** page in Hue. They can enable or disable the task server from the Advanced Configuration Snippet in Cloudera Data Warehouse. They can also modify the default behavior, such as the frequency of the cleanup jobs, threshold of the disk space, and so on.

All users having access to Hue can leverage the improved file-upload capabilities from the Hue task server.

Note: The task server is available only for Hue that you access from the Virtual Warehouses. It is not available for the shared Hue service in Cloudera Data Warehouse.

Scheduling tasks to clean up Hue documents

If the task server is enabled, then you can clean up data from the backend Hue tables from the Hue web interface. You no longer need to run the document cleanup shell commands or queries for this purpose.

About this task

Note: Only Hue Superusers or Administrators can access the **Task Server** page to cleanup documents and /tmp directory.

Procedure

1. Log in to the Hue web interface as a superuser.
2. Click Administer Server from the left assist panel and then click the Task Server tab.
3. Click Schedule Task.
4. On the **Schedule Task** modal, select document cleanup from the dropdown menu.

   In the keep-days field, specify the number of days for which you want to retain the data in the Hue tables. For example, if you specify 30, the tables are cleaned up every 30 days.
5. Click Submit.

What to do next

You can monitor the logs by clicking the job ID. By default, only INFO-level logs are displayed. You can change this to DEBUG-level from the Hue Advanced Configuration Snippet for troubleshooting purposes.

Scheduling tasks to clean up Hue's tmp directory

Hue writes temporary files to a temporary block space, also used by other services such as Hive and Impala. If the tmp space is full, the Hue pod gets evicted disrupting operational continuity. Set up a cleanup threshold in the Hue task server to prevent pod eviction.

About this task

Files older than 60 minutes are cleaned up from the /tmp directory. This is a configurable parameter. The default cleanup threshold is 90%.

Note: Only Hue Superusers or Administrators can access the **Task Server** page to cleanup documents and /tmp directory.

Procedure

1. Log in to the Hue web interface as a superuser.
2. Click Administer Server from the left assist panel and then click the Task Server tab.
3. Click Schedule Task.
4. On the **Schedule Task** modal, select tmp clean up from the dropdown menu.

   In the threshold for clean up field, specify the threshold value in percentage.

   The cleanup job is triggered when the disk space reaches the specified threshold value. For example, if you specify 70, the temporary disk space is cleaned up when it is 70% full.
5. Click Submit.

What to do next

You can monitor the logs by clicking the job ID. By default, only INFO-level logs are displayed. You can change this to DEBUG-level from the Hue Advanced Configuration Snippet for troubleshooting purposes.

Advanced configuration snippet parameters for Hue task server in Cloudera Data Warehouse

The Hue task server is enabled by default in Cloudera Data Warehouse with preset cleanup job schedules. The preset values can work in most scenarios. If needed, you can override the values from the Hue Advanced Configuration Snippet.

Configuration parameter	Description	Default value
enabled	A setting to enable or disable the Task Server in Hue. To turn off using the task, set the value of this property to False.	True
check_disk_usage_and_clean_task_periodic_interval	The time interval in seconds at which the cleanup job is run.	1000
disk_usage_cleanup_threshold	A setting to specify the threshold value of disk space in percentage. For example, the cleanup job is triggered when the disk is 90% full.	90

Configuration parameter	Description	Default value
disk_usage_and_clean_task_time_delta	A setting to specify the time (in minutes) after which the files should be cleaned up. For example, if this is set to 60, then only the files with a timestamp older than 60 minutes will be cleaned up from the /tmp directory. This setting helps avoid race condition issues.	60
cleanup_stale_uploads_task_time_delta	The time interval in seconds to run the peroidic cleaner which cleans up failed upload tasks. Only successfully completed upload task keys get cleaned automatically. The failed upload task's keys are retained.	900
cleanup_stale_uploads_task_time_delta	A setting to clean up keys of uploads tasks older than the time delta. The unit of time delta is minutes.	60

How to override the default presets for the Hue task server

Go to Virtual Warehouses ⋮ Edit Configurations , select hue-safety-valve from the Configuration files drop-down menu, and add the following lines:

```
[desktop]
[[task_server_v2]]
```

Cloudera Runtime 7.2.11

Configuring Apache Hive

Date published: 2019-08-21
Date modified: 2022-08-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring Hive in Cloudera Data Warehouse..... 4

Understanding CREATE TABLE behavior..... 6

Generating Hive statistics in Cloudera Data Warehouse..... 7

Generating and viewing Apache Hive statistics in Cloudera Data Warehouse..... 8

Statistics generation and viewing commands in Cloudera Data Warehouse..... 8

Configuring Hive in Cloudera Data Warehouse

Configuring Hive performance in the Cloud is rarely, if ever, required relative to the extensive performance tuning typical in a bare metal environment. You might occasionally want to configure Hive properties unrelated to performance, such as HiveServer (HS2) administration or logging level properties.

About this task

Changing one of the vast array of configuration parameters from Cloudera Data Warehouse is recommended only when following Cloudera instructions. For example, you follow instructions in a compaction alert or the product documentation that describes the property and value to change, and how to do it.

In Cloudera Data Warehouse, if you have required permissions, you can configure Hive, Hive metastore, or HiveServer properties from a Database Catalog or Virtual Warehouse. In this task, you see how to change a property and add a property from the Database Catalog, and then from a Virtual Warehouse.

Before you begin

- You have the appropriate Admin role to make the configuration change.

Procedure

1. Click Cloudera Data Warehouse Overview Database Catalogs Options ⋮ Edit Configurations Metastore .
2. From the drop-down, select a configuration category, for example hive-site.

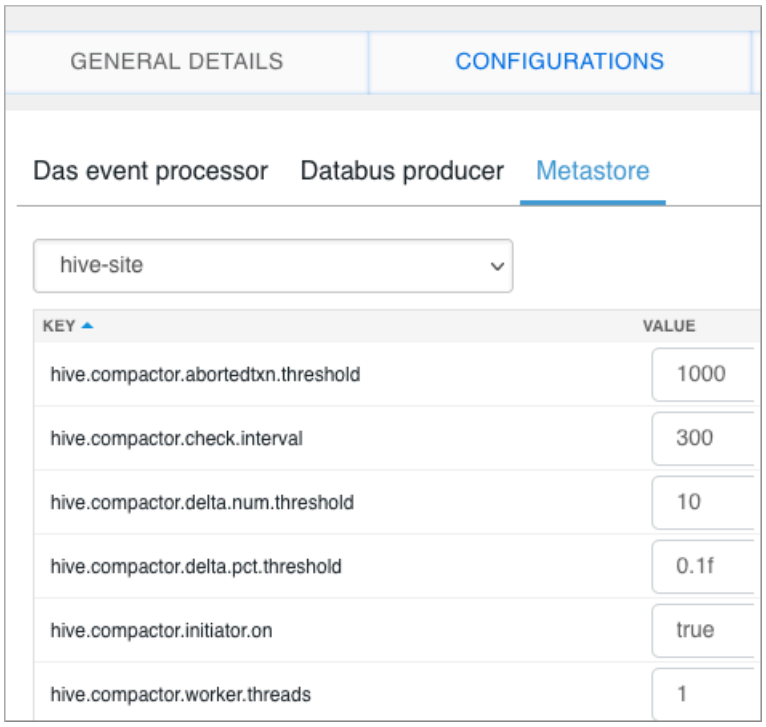

3. Change, or set, the property value.

4. Click + to add a property to hive-site, for example.

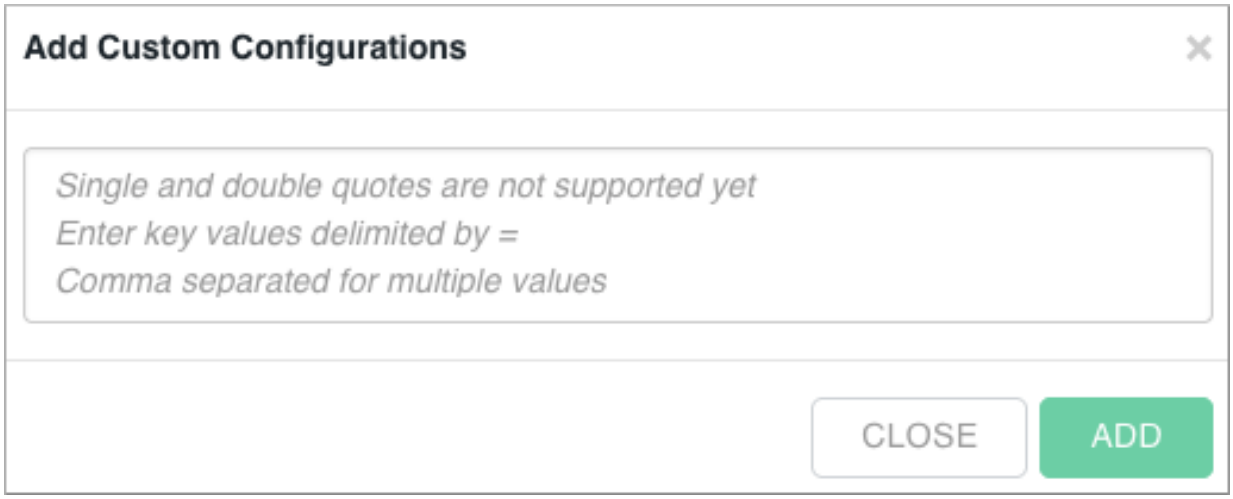

5. Enter the name of the property you want to add, the equals symbol, and one or more comma-separated values. Do not use single or double quotation marks to enclose the property or value.
6. Click Add.
7. Save, and restart the Database Catalog.
8. Click Virtual Warehouses, and select a Hive Virtual Warehouse.
9. Click Actions Edit Configurations HiveServer2 .

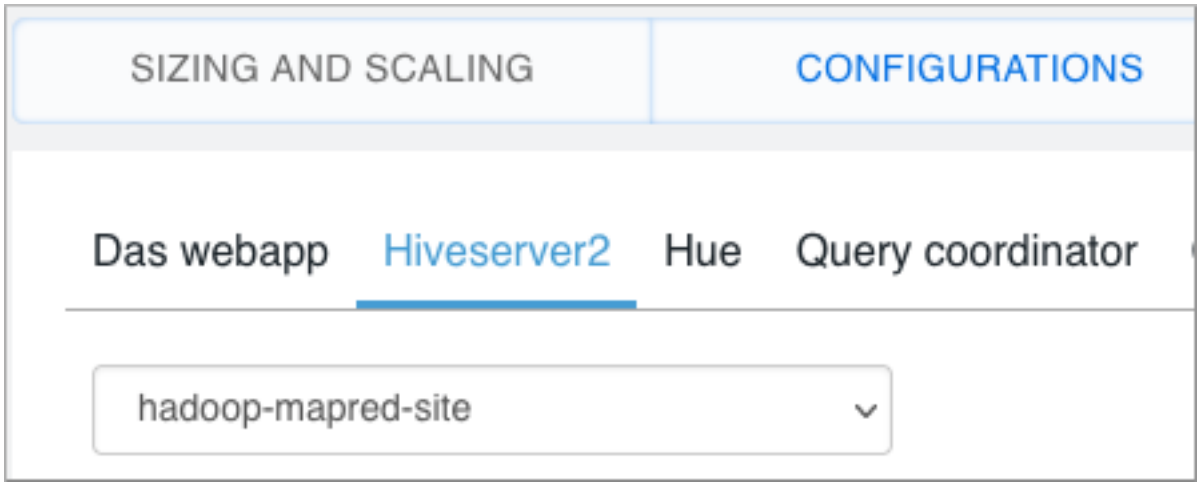

10. Search for a specific property, or view categories of properties by selecting a category in the dropdown.

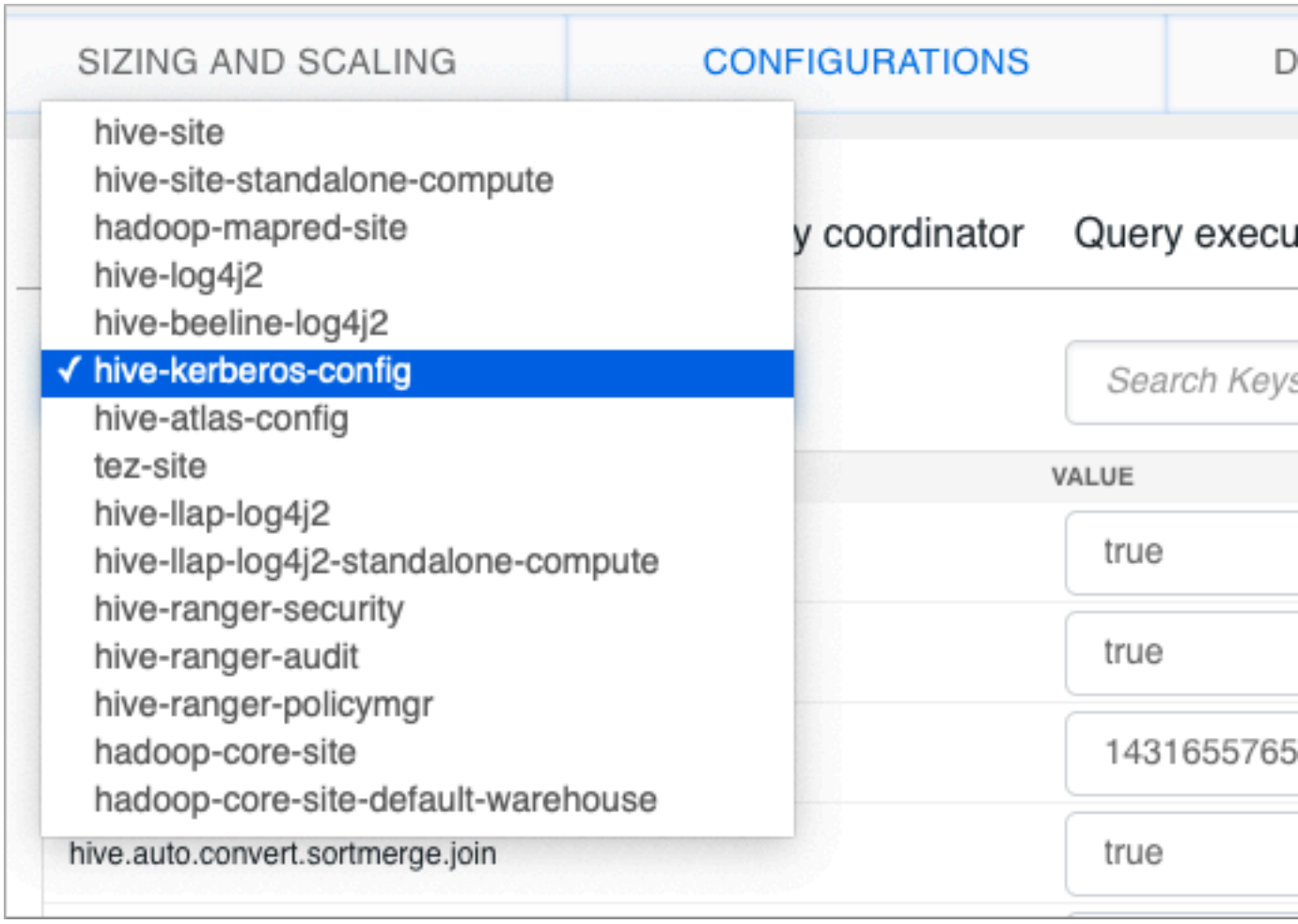

Understanding CREATE TABLE behavior

Hive table creation has changed significantly since Hive 3 to improve useability and functionality.

Hive has changed table creation in the following ways:

- Creates ACID-compliant table, which is the default in
- Supports simple writes and inserts
- Writes to multiple partitions
- Inserts multiple data updates in a single SELECT statement
- Eliminates the need for bucketing.

If you have an ETL pipeline that creates tables in Hive, the tables will be created as ACID. Hive now tightly controls access and performs compaction periodically on the tables. Using ACID-compliant, transactional tables causes no performance or operational overload. The way you access managed Hive tables from Spark and other clients changes. In , access to external tables requires you to set up security access permissions.

Modify the default CREATE TABLE behavior

Override default behavior when creating the table

Irrespective of the database, session, or site-level settings, you can override the default table behavior by using the MANAGED or EXTERNAL keyword in the CREATE TABLE statement.

```
CREATE [MANAGED][EXTERNAL] TABLE foo (id INT);
```

Set the default table type at a database level

You can use the database property, defaultTableType=EXTERNAL or ACID to specify the default table type to be created using the CREATE TABLE statement. You can specify this property when creating the database or at a later point using the ALTER DATABASE statement. For example:

```
CREATE DATABASE test_db WITH DBPROPERTIES ('defaultTableType'='E
XTERNAL');
```

In this example, tables created under the test_db database using the CREATE TABLE statement creates external tables with the purge fucntionality enabled (external.table.purge = 'true').

You can also choose to configure a database to allow only external tables to be created and prevent creation of ACID tables. While creating a database, you can set the database property, EXTER NAL_TABLES_ONLY=true to ensure that only external tables are created in the database. For example:

```
CREATE DATABASE test_db WITH DBPROPERTIES ('EXTERNAL_TABLES_ONLY
'='true');
```

Set the default table type at a session level

You can configure the CREATE TABLE behavior within an existing beeline session by setting hive.create.as.external.legacy to true or false. Setting the value to true results in configuring the CREATE TABLE statement to create external tables by default.

When the session ends, the default CREATE TABLE behavior also ends.

Set the default table type at a site level

You can configure the CREATE TABLE behavior at the site level by configuring the hive.create. as.insert.only and hive.create.as.acid properties in under Hive configuration. When configured at the site level, the behavior persists from session to session.

If you are a Spark user, switching to legacy behavior is unnecessary. Calling ‘create table’ from SparkSQL, for example, creates an external table after upgrading to as it did before the upgrade. You can connect to Hive using the Hive Warehouse Connector (HWC) to read Hive ACID tables from Spark. To write ACID tables to Hive from Spark, you use the HWC and HWC API. Spark creates an external table with the purge property when you do not use the HWC API. For more information, see Hive Warehouse Connector for accessing Spark data.

Generating Hive statistics in Cloudera Data Warehouse

A cost-based optimizer (CBO) generates efficient query plans. Hive does not use the CBO until you generate column statistics for tables. By default, Hive gathers only table statistics. You need to configure Hive to enable gathering of column statistics.

The CBO, powered by Apache Calcite, is a core component in the Hive query processing engine. The CBO optimizes plans for executing a query, calculates the cost, and selects the least expensive plan to use. In addition to increasing the efficiency of execution plans, the CBO conserves resources.

How the CBO works

After parsing a query, a process converts the query to a logical tree (Abstract Syntax Tree) that represents the operations to perform, such as reading a table or performing a JOIN. Calcite applies optimizations, such as query rewrite, JOIN re-ordering, JOIN elimination, and deriving implied predicates to the query to produce logically

equivalent plans. Bushy plans provide maximum parallelism. Each logical plan is assigned a cost that is based on distinct, value-based heuristics.

The Calcite plan pruner selects the lowest-cost logical plan. Hive converts the chosen logical plan to a physical operator tree, optimizes the tree, and converts the tree to a Tez job for execution on the Hadoop cluster.

Explain plans

You can generate explain plans by running the EXPLAIN query command. An explain plan shows you the execution plan of a query by revealing the operations that occur when you run the query. Having a better understanding of the plan, you might rewrite the query or change Tez configuration parameters.

Generating and viewing Apache Hive statistics in Cloudera Data Warehouse

You can use statistics to optimize queries for improved performance. The cost-based optimizer (CBO) also uses statistics to compare query plans and choose the best one. By viewing statistics instead of running a query, you can often get answers to your data questions faster.

About this task

This task shows how to generate different types of statistics about a table.

Procedure

1. Launch a Hive shell or editor.
2. Gather table statistics for the non-partitioned table mytable:

```
ANALYZE TABLE mytable COMPUTE STATISTICS;
```

3. View table statistics you generated:

```
DESCRIBE EXTENDED mytable;
```

4. Gather column-level statistics for the table:

```
ANALYZE TABLE mytable COMPUTE STATISTICS FOR COLUMNS;
```

5. View column statistics for the col_name column in my_table in the my_db database:

```
DESCRIBE FORMATTED my_db.my_table col_name;
```

Related Information

Apache Hive Wiki language reference
Apache Hive Wiki - Statistics in Hive

Statistics generation and viewing commands in Cloudera Data Warehouse

You can manually generate table and column statistics, and then view statistics using Hive queries. By default, Hive generates table statistics, but not column statistics, which you must generate manually to make cost-based optimization (CBO) functional.

Commands for generating statistics

The following ANALYZE TABLE command generates statistics for tables and columns:

ANALYZE TABLE [table_name] COMPUTE STATISTICS;

Gathers table statistics for non-partitioned tables.

ANALYZE TABLE [table_name] PARTITION(partition_column) COMPUTE STATISTICS;

Gathers table statistics for partitioned tables.

ANALYZE TABLE [table_name] COMPUTE STATISTICS for COLUMNS [comma_separated_column_list];

Gathers column statistics for the entire table.

ANALYZE TABLE partition2 (col1="x") COMPUTE STATISTICS for COLUMNS;

Gathers statistics for the partition2 column on a table partitioned on col1 with key x.

Commands for viewing statistics

You can use the following commands to view table and column statistics:

DESCRIBE [EXTENDED] table_name;

View table statistics. The EXTENDED keyword can be used only if the hive.stats.autogather property is enabled in the hive-site.xml configuration file. Use the Cloudera Manager Safety Valve feature (see link below).

DESCRIBE FORMATTED [db_name.]table_name [column_name] [PARTITION (partition_spec)];

View column statistics.

Related Information

Custom Configuration (about Cloudera Manager Safety Valve)

1.0.0

Cloudera Data Warehouse Runtime Release Notes

Date published: 2024-07-26
Date modified: 2024-08-15

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Data Warehouse Runtime changelog..... 4
Hive changelog for Cloudera Data Warehouse Public Cloud..... 4
Impala changelog for Cloudera Data Warehouse Public Cloud..... 6
Iceberg changelog for Cloudera Data Warehouse Public Cloud..... 10
Hue changelog for Cloudera Data Warehouse Public Cloud..... 12

CDW Public Cloud Runtime Changelog

Review the changes introduced in this release of Cloudera Data Warehouse (CDW) Runtime on Public Cloud. This includes new features, improvements, and fixes for Hive, Impala, Iceberg, and Hue.

Hive changelog for Cloudera Data Warehouse Public Cloud

Review the changes introduced in Hive for Cloudera Data Warehouse Public Cloud.

2024.0.18.4 (December 5, 2024) GA

- DWX-19668 HIVE-28616: OrcReader is not closed in OrcEncodedDataReader
- CDPD-75048 HIVE-13153: SessionID is appended to thread name twice
- CDPD-75556: Call createDmlFragment() when inserting into a partitioned table
- DWX-19504: HIVE-22948: QueryCache: Treat query cache locations as temporary storage
- CDPD-74640: HIVE-25907: IOW Directory queries fails to write data to final path when query result cache is enabled
- CDPD-74640: HIVE-21386: Extend the fetch task enhancement done in HIVE-21279 to make it work with query result cache
- CDPD-74640: HIVE-28530: Set files in thread safe manner in HiveSequenceFileInputFormat
- CDPD-72985: HIVE-27984: Support backward compatibility of hms thrift struct about column stats
- CDPD-71484 HIVE-27884: LLAP: Reuse FileSystem objects from cache across different tasks in the same LLAP daemon

2024.0.18.2-4 (October 4, 2024)

No new features or fixes.

2024.0.18.1-1 (August 15, 2024)

No new features or fixes.

2024.0.18.0-206 (July 26, 2024)

- DWX-18331: HIVE-28121: Use direct SQL for transactional altering table parameter
- CDPD-64754: HIVE-28202: Incorrect projected column size after ORC upgrade to v1.6.7
- DWX-17033: Fixed NPE for UA non BinaryPredicate join condition.
- CDPD-69885: HIVE-28253: Unable to set the value for hplsql.onerror in hplsql mode.
- CDPD-69724: HIVE-28247: Execute immediate `select count(*) from tbl` throwing ClassCastException in hplsql mode.
- CDPD-69648: HIVE-28215: Signalling CONDITION HANDLER is not working in HPLSQL.
- CDPD-69647: HIVE-28214: HPLSQL not using the hive variables passed through beeline using --hivevar option
- CDPD-69170:HIVE-28188:Upgrade Postgres to 42.7.3 to target CVE-2024-1597
- CDPD-7721: HIVE-22489: Reduce Sink operator should order nulls by parameter
- CDPD-68719: upgrade stack version to 7.2.18.0
- DWX-17905: HIVE-28165: HiveSplitGenerator: send splits through filesystem instead of RPC in case of big payload
- DWX-17905: HIVE-24415: HiveSplitGenerator blocks Tez dispatcher
- DWX-17957: HIVE-28167: Full table deletion fails when converting to truncate for Iceberg and ACID tables
- HIVE-27751 Log Query Compilation summary in an accumulated way
- DWX-17928: HIVE-27585: Upgrade kryo serialization lib to latest version (#4570).

- CDPD-67940: HIVE-28111: FSCountersSummary is only printed when LLAP IO is enabled
- CDPD-67793: HIVE-26955: Select query fails when decimal column data type is changed to string/char/varchar in Parquet
- CDPD-65291 :Drop database query failing with Invalid ACL Exception
- CDPD-67350: HIVE-27775: DirectSQL and JDO results are different when fetching partitions by timestamp in DST shift
- CDPD-66862: Upgrade mysql-connector-java in Hive packaging Docker File
- CDPD-65448: HIVE-28006: Materialized view with aggregate function incorrectly shows it allows incremental rebuild
- DWX-17487: HIVE-28051: LLAP: cleanup local folders on startup and periodically
- CDPD-67328: HIVE-28076: Selecting data from a bucketed table with decimal column type throwing NPE.
- CDPD-66902: HIVE-27406: Addendum: Query runtime optimization
- CDPD-61128: HIVE-27490: HPL/SQL says it support default value for parameters but not considering them when no value is passed
- CDPD-62437: HIVE-27692: Explore removing the maintenance task from the embedded Metastore
- CDPD-66930: Recompute scan ranges and stats in ImpalaHdfsScanNode
- CDPD-67205: HIVE-28083: Enable HMS client/query cache for EXPLAIN queries
- CDPD-64442: HIVE-27924: Incremental rebuild goes wrong when inserts and deletes overlap between the source tables
- CDPD-66644: HIVE-28050: Disable Incremental non aggregated materialized view rebuild in presence of delete operations
- CDPD-57121: HIVE-24411: Make ThreadPoolExecutorWithOomHook more awareness of OutOfMemoryError
- CDPD-67063: HIVE-28078: TestTxnDbUtil should generate csv files when we query the metastore database
- DWX-17615: HIVE-28093: Re-execute DAG in case of NoCurrentDAGException
- CDPD-61611: HIVE-27778: Alter table command gives error after computer stats is run with Impala
- CDPD-66267: HIVE-26882: Allow transactional check of Table parameter before altering the Table
- DWX-17274: TestInputFormatReaderDeletes fix
- CDPD-66632: CDPD-50060: HIVE-27114: Include all properties by default
- DWX-16861: CVE fixes for Apache hive JDBC driver
- DWX-17435: Facing dependency issues while using hive JDBC jar with JDBC client other than beeline
- CDPD-66644: HIVE-28050: Disable Incremental non aggregated materialized view rebuild in presence of delete operations
- CDPD-61250: Remove jwtprovider-knox
- CDPD-8806: HIVE-22961: Drop function in Hive should not send request for drop database to Ranger plugin
- CDPD-66527: HIVE-28054: SemanticException for join condition in subquery
- CDPD-63420: HIVE-25183: Inner Join condition parsing error in subquery
- CDPD-66089: HIVE-28000: Hive QL : "not in" clause gives incorrect results when type coercion cannot take place.
- CDPD-66090: HIVE-27161: MetaException when executing CTAS query in Druid storage handler
- DWX-17186 : HIVE-27938: Iceberg: Fix java.lang.ClassCastException during vectorized reads on partition columns
- CDPD-66084: HIVE-27948: Wrong results when using materialized views with non-deterministic/dynamic functions
- CDPD-63299:HIVE-27889: Bump org.apache.santuario:xmlsec from 2.3.0 to 2.3.4
- CDPD-65291 : HIVE-27911 : Drop database query failing with Invalid ACL Exception
- CDPD-65019: HIVE-28009: Shared work optimizer ignores schema merge setting in case of virtual column difference)
- CDPD-61127: HIVE-27489: HPL/SQL does not support table aliases on column names in loops
- CDPD-65805 HIVE-27993 : Netty4 ShuffleHandler: should use one boss thread

Impala changelog for Cloudera Data Warehouse Public Cloud

Review the changes introduced in Impala for Cloudera Data Warehouse Public Cloud.

2024.0.18.4 (December 5, 2024) GA

- CDPD-75836: Selectively backport CDPD-75588 to CDWH-2024.0.18.4
- CDPD-45946: IMPALA-13386: Fix test_metastore_service.py after HIVE-27984
- CDPD-45946: Fix test_metastore_service.py after HIVE-27984
- CDPD-69857: FENG: Disable test of insert partition with special characters in partition name
- CDPD-72961: IMPALA-13280: Speed up test_under_statement_expression_limit
- CDPD-75562: IMPALA-13469: Deflake test_query_cpu_count_on_insert
- IMPALA-13445: Ignore num partition for unpartitioned writes
- IMPALA-13312: Use client address from X-Forwarded-For Header in Ranger Audit Logs
- IMPALA-13310 Add the value of the http 'X-Forwarded-For' header to the runtime profile
- CDPD-73442: IMPALA-13313: Fix ExpireQueries deadlock

2024.0.18.2-4 (October 4, 2024)

No new features or fixes.

2024.0.18.1-1 (August 15, 2024)

- IMPALA-13270: Addressing IllegalStateException in Complex Views post upgrade
- IMPALA-13272: Stability Improvement for analytic functions on collections

2024.0.18.0-206 (July 26, 2024)

- IMPALA-13137: Add additional client fetch metrics columns to the queries page
- IMPALA-12800: Implement hashCode everywhere
- IMPALA-13159: Fix query cancellation caused by statestore failover
- IMPALA-13152: Avoid NaN, infinite, and negative ProcessingCost
- IMPALA-13138: Never smallify existing StringValue objects, only new ones during DeepCopy
- IMPALA-13131: Azure OpenAI API expects 'api-key' instead of 'Authorization' in the request header
- IMPALA-12800: Add cache for isTrueWithNullSlots() evaluation
- IMPALA-12800: Use HashMap for ExprSubstitutionMap lookups
- IMPALA-13119: Fix cost_ initialization at CostingSegment.java
- IMPALA-12616: Fix test_restart_services.py::TestRestart tests for S3
- IMPALA-13129: Move runtime filter skipping at registerRuntimeFilter
- IMPALA-8042: Assign BETWEEN selectivity for discrete-unique column
- IMPALA-12800: Skip O(n^2) ExprSubstitutionMap::verify() for release builds
- IMPALA-13040: Inject larger delay for sanitized build
- IMPALA-13134: DDL hang with SYNC_DDL enabled when Catalogd is changed to standby status
- IMPALA-11499: Refactor UrlEncode function to handle special characters
- IMPALA-13040: Add waiting mechanism in UpdateFilterFromRemote
- IMPALA-13018: Block push down of conjuncts with implicit casting on base columns for jdbc tables
- IMPALA-13058: Init first_arrival_time_ and completion_time_ with -1
- IMPALA-13054: Avoid revisiting children in QueryStateExpanded
- IMPALA-13061: Create query live as external table
- IMPALA-12910: Support running TPCH/TPCDS queries for JDBC tables
- IMPALA-13049: Add dependency management for log4j2 to use 2.18.0
- IMPALA-13045: Wait for impala_query_live to exist

- IMPALA-12997: Use graceful shutdown for query log tests
- IMPALA-13012: Lower default query_log_max_queued
- IMPALA-13005: Create Query Live table in HMS
- IMPALA-13024: Ignore slots if using default pool and empty group
- IMPALA-12657: Improve ProcessingCost of ScanNode and NonGroupingAggregator
- IMPALA-12988: Calculate an unbounded version of CpuAsk
- IMPALA-12980: Translate CpuAsk into admission control slots
- IMPALA-12965: Add debug query option RUNTIME_FILTER_IDS_TO_SKIP
- IMPALA-13004: Fix heap-use-after-free error in ExprTest AiFunctionsTest
- IMPALA-12874: Identify active and standby catalog and statestore in the web debug endpoint
- IMPALA-12564: Prevent Hive loading libfesupport.so in the minicluster during TSAN runs
- IMPALA-12998: Add SHOW_METADATA_TABLES to ignored DDL
- IMPALA-12963: Return parent PID when children spawned
- IMPALA-12920: Support ai_generate_text built-in function for OpenAI's chat completion API
- IMPALA-12969: Release JNI array if DeserializeThriftMsg failed
- IMPALA-12881: Use getFkPkJoinCardinality to reduce scan cardinality
- IMPALA-11430: Support custom hash schema for Kudu range tables
- IMPALA-12960: Fix Incorrect RowsPassedThrough Metric in Streaming Aggregation
- IMPALA-12925: Fix decimal data type for external JDBC table
- IMPALA-12899: Temporary workaround for BINARY in complex types
- IMPALA-12602: Unregister queries on idle timeout
- IMPALA-12782: Show info of the event processing in /events webUI
- IMPALA-12852: Make Kudu service start and stop independent
- IMPALA-12426: Switches the duration fields to be stored in decimal seconds.
- IMPALA-12944: Fixes Workload Management Test Flakiness
- IMPALA-12426: Skip Inserting HS2 Operation Queries into the Completed Queries Table
- IMPALA-12626: Add Tables Queried to profile/history
- IMPALA-12264: Add limit on number of HS2 sessions per user.
- IMPALA-12540: Add EventSequence arg to load
- IMPALA-12540: Query Live Table
- IMPALA-12824: Removes the prettyprint_duration Built-in Function
- IMPALA-12942: deflake test_virtual_column_file_position_generic
- IMPALA-12487: Skip reloading file metadata for ALTER_TABLE events with trivial changes in StorageDescriptor
- IMPALA-12928: Mask JDBC table property dbcp.password for DESC FORMATTED and SHOW CREATE TABLE
- IMPALA-12913: Refactor Workload Management Custom Cluster Tests
- IMPALA-12699: Set timeout for catalog RPCs
- IMPALA-12807: Add support for mold linker
- IMPALA-12915: Use libgtest.so when built with shared libs
- IMPALA-12903: Querying virtual column FILE__POSITION for TEXT and JSON tables crashes Impala
- IMPALA-12896: JDBC table must be created as external table
- IMPALA-4545: Simplify test dimension in test_decimal_casting.py
- IMPALA-12898: Tidy up test dimensions of test_scanner.py
- IMPALA-12929: Skip loading HDFS permissions in local-catalog mode
- IMPALA-12930: Fix TestExtDataSources.test_jdbc_data_source failure
- IMPALA-12923: Fix header alignment during horizontal scrolling in query timeline
- IMPALA-12926: Refactor BINARY type handling in the backend
- IMPALA-12819: Sync add-opens targets
- IMPALA-12443: Add catalog timeline for all DDL profiles

- IMPALA-11938: Raised error if NUM_NODES is set to invalid value.
- IMPALA-12802: Support ALTER TABLE for JDBC tables
- IMPALA-12924: Upgrade xmlsec to 2.2.6
- IMPALA-12426: Query History Table
- IMPALA-12883: Support updating the charge on an entry in the cache
- IMPALA-12904: test_type_conversions_hive3 silently passes because of wrongly defined test dimensions
- IMPALA-12914: TestBannedLogMessages no longer requires running cluster
- IMPALA-12818: Intermediate Result Caching plan node framework
- IMPALA-12835: Fix event processing without hms_event_incremental_refresh_transactional_table
- IMPALA-12917: Skip TestEventProcessingError tests requiring tez execution on hive
- IMPALA-12916: Fix test_event_processor_error_global_invalidate test random failure
- IMPALA-12896: Avoid JDBC table to be set as transactional table
- IMPALA-12831: Fix HdfsTable.toMinimalTCatalogObject() failed by concurrent modification
- IMPALA-12832: Implicit invalidate metadata on event failures
- IMPALA-12798: Tolerate floationg point precision for regr_intercept() TIMESTAMP test
- IMPALA-12730: Don't use -Weverything for clang-tidy
- IMPALA-12849: Standby catalogd should reject requests from coordinators
- IMPALA-12855: Fix NPE in firing RELOAD events when the partition doesn't exist
- IMPALA-12678: Deflake test_skipping_batching_events
- IMPALA-12878: Report invalid query if finalized
- IMPALA-12860: Invoke validateDataFilesExist for RowDelta operations
- IMPALA-12864: Deflake test_query_log_size_in_bytes.
- IMPALA-12851: Fix AllocWriteIdEvent process issue to add txnId-tableWriteIds mapping
- IMPALA-12815: Support timestamp for scan predicates for external data source table.
- IMPALA-12833: Enabled 'catalogd_ha_reset_metadata_on_failover' by default
- IMPALA-12821: Fix test_create_table_profile_events depend on event-processor
- IMPALA-12825: Install thrift into the impala-python virtualenv
- IMPALA-12834: Add number of concurrent queries to profile
- IMPALA-12859: Ensure query log estimated memory is initialized
- IMPALA-12838: Adds exec_options parameter to single_node_perf_run.py
- IMPALA-12717: Fix error message for missing table descriptor
- IMPALA-12426: QueryStateRecord Refactor
- IMPALA-12827: Fix failures in processing AbortTxnEvent due to aborted write id is cleaned up
- IMPALA-12426: Workload Management Supporting Changes
- IMPALA-12840: Exclude THdfsFileDesc in getJsonCatalogObject
- IMPALA-12793: Create JDBC table without data source
- IMPALA-12823: Fix repeated query not found messages in impalad.INFO logs
- IMPALA-12801: Increase query_log_ default size and bound its memory.
- IMPALA-12426: Remove Usage of Unnecessary this->
- IMPALA-12433: Share buffers among channels in KrpcDataStreamSender
- IMPALA-12824: Adds built-in functions prettyprint_duration and prettyprint_bytes.
- IMPALA-12842: Make node.js installation aware of CPU architecture
- IMPALA-12830: test_webserver_hide_logs_link() could fail in the exhaustive build
- IMPALA-12847: Expose computeScanRangeLocations and computeStats
- IMPALA-11123: Reimplement ORC optimized count star
- IMPALA-9086: Show Hive configurations in /hadoop-varz page
- IMPALA-12796: Add is_footer_only in TFileSplitGeneratorSpec
- IMPALA-12631: Improve count star performance for parquet scans
- IMPALA-12811: Exception during re-analyze can be lost
- IMPALA-12814: Allow use of row_regex for VERIFY_IS_NOT_IN

- IMPALA-11568: Fix SamplingTimeSeriesCounter not correctly cleared
- IMPALA-12816: Remove exclusion of Netty to allow SSL connections to HBase
- IMPALA-12790: Fix overestimation in ScanNode.getInputCardinality
- IMPALA-12803: Fix missing exchange lines in query timeline
- IMPALA-12795: Deflake test_catalog_operation_fields
- IMPALA-12688: Support JSON profile imports in webUI
- IMPALA-12744: Support for regr_count() aggregate/analytic function
- IMPALA-12788: Fix HBaseTable still get loaded even if HBase is down
- IMPALA-12378: Add commons-dbcp2.version replacement
- IMPALA-12578: Pass owner user of database and table to Ranger in GRANT/REVOKE
- IMPALA-12378: Auto Ship JDBC Data Source
- IMPALA-12780: Only show non-default options in the catalog operations page
- IMPALA-12784: Fix bug in ScanNode.getFilteredInputCardinality
- IMPALA-12503: Support date data type for predicates for external data source table
- IMPALA-12783: Nested struct with varlen data crashes
- IMPALA-12726: Simulate large-scale query in TpcdsCpuCostPlannerTest
- IMPALA-12655: Stabilize compute-table-stats.sh
- IMPALA-12770: Fix infinite loop for nested Case expressions
- IMPALA-12781: ARRAY<STRUCT<s: STRING> crashes in top-n
- IMPALA-12718: Provides UTF-8 support for the trim functions
- IMPALA-12778: Fix time strings in catalog operations page
- IMPALA-12642: Fixed unit-test to verify query options for JDBC external table
- IMPALA-12695: Crash with UNION with complex types
- IMPALA-12448: Avoid getting stuck when refreshing a non-existent partition
- IMPALA-12471: skip mysql ext jdbc tests if setup environment fails.
- IMPALA-12763: Union with string struct crashes in ASAN
- IMPALA-12762: Fix cmake error in package building
- IMPALA-12767: Upgrade Guava to 32.0.1 due to CVE-2023-2976
- IMPALA-12463: Batch non-consecutive table events in the event processor
- IMPALA-12745: Skip parallel symbol dumping with RPM/DEB packages
- IMPALA-12125: Support for dumping symbols from RPMs without separate symbols
- IMPALA-12746: Bump jackson.databind to 2.15.3
- IMPALA-12727: Reduce IO threads for non-TARGET_FILESYSTEM filesystems
- IMPALA-12740: Fix TestHdfsJsonScanNodeErrors fails in exhaustive mode
- IMPALA-12711: Fix DDL errors are not shown in impalad logs
- IMPALA-12590: Fix dmesg call during precommit for Ubuntu 20.04
- IMPALA-12584: Enable strict data file access by default
- IMPALA-12038: Switch report_benchmark_results.py to python 3
- IMPALA-12716: Fix timeout thresholds in test_catalog_operations_with_rpc_retry
- IMPALA-12431: Support reading compressed JSON file
- IMPALA-12690: Add TpcdsCpuCostPlannerTest
- IMPALA-12642: Support query options for Impala external JDBC table
- IMPALA-12054: Lazily check Kudu flags in tests
- IMPALA-12582: Fix crash when enabling MIN_MAX RuntimeFilter in Nested Loop Join
- IMPALA-12665: Adjust complete_micro_batch_ length to new scratch_batch_->capacity after ScratchTupleBatch::Reset
- IMPALA-12687: Fix key conflicts in tracking in-flight catalog operations
- IMPALA-12356: Fix first ALTER_PARTITION event from Hive could be treated as self event
- IMPALA-12747: Atomic update of execution state
- IMPALA-12719: Reload filemetadata for AlterTable event of type truncate

- IMPALA-12566: Fix RpcMgrKerberizedTest on RedHat 8
- IMPALA-12743: Fix incremental stats are filtered out by HMS due to HIVE-27114

Iceberg changelog for Cloudera Data Warehouse Public Cloud

Review the changes introduced in Iceberg for Cloudera Data Warehouse Public Cloud.

2024.0.18.4 (December 5, 2024) GA

- CDPD-68442: IMPALA-12970: Fix ConcurrentModificationException for Iceberg table scans
- IMPALA-13432: test_migrated_table_field_id_resolution_complex can fail with different Iceberg version
- CDPD-74017: IMPALA-13364: Schema resolution doesn't work for migrated partitioned Iceberg tables that have complex types
- CDPD-75084: HIVE-28570: Iceberg: Aggregate queries on translated iceberg table fails post rename.

2024.0.18.2-4 (October 4, 2024)

Hive - Iceberg changes

- DWX-18477: HIVE-28282: Addendum: Iceberg: Merge query failures with reserved keywords as column names

2024.0.18.1-1 (August 15, 2024)

No new features or fixes.

2024.0.18.0-206 (July 26, 2024)

Hive - Iceberg changes

- DWX-13719: Iceberg positional deletes cache support
- DWX-18477: HIVE-28282: Merging into iceberg table fails with copy on write when values clause has a function call
- CDPD-69704: HIVE-28266: Iceberg: select count(*) from data_files metadata tables gives wrong result
- CDPD-71800: Iceberg: Use 'merge-on-read' as default write mode for v2 tables
- CDPD-70188: HIVE-28278: Iceberg: Stats: IllegalStateException Invalid file: file length 0
- DWX-18213: NFQE SMG Iceberg: Hive select after deletes on table returning inconsistent data
- CDPD-68088: HIVE-28127,HIVE-28203: Exception when rebuilding materialized view with calculated columns on iceberg sources
- CDPD-68087: HIVE-28126: Use added record count in cost model when rebuilding materialized view stored by iceberg
- CDPD-68654: HIVE-28172: Iceberg: Truncate should work for V1 tables with partition evolution.
- CDPD-68285: HIVE-28166: Iceberg: Truncate on branch operates on the main table.
- DWX-17402: HIVE-28069: Iceberg: Implement Merge task functionality for Iceberg tables - Add serde properties for CTAS and CMV
- CDPD-67867: HIVE-28130: Iceberg: Add support for Rename Branch.
- DWX-17957: HIVE-28167: Full table deletion fails when converting to truncate for Iceberg and ACID tables
- DWX-17926: HIVE-28020: Iceberg: Upgrade iceberg version to 1.4.3
- DWX-17926: HIVE-27819: Iceberg: Upgrade iceberg version to 1.4.2
- CDPD-66779: HIVE-28087: Iceberg: Timestamp partition columns with transforms are not correctly sorted during insert
- DWX-17402: HIVE-28069: Iceberg: Implement Merge task functionality for Iceberg tables

- CDPD-66865: HIVE-28119: Iceberg: Allow insert clause with a column list in Merge query not_matched condition
- DWX-16464: HIVE-27653: Iceberg: Add conflictDetectionFilter to validate concurrently added data and delete files
- CDPD-67008: HIVE-28021: Iceberg: Attempting to create a table with a percent symbol fails
- CDPD-67521: HIVE-28015: Iceberg: Add identifier-field-ids support in Hive
- CDPD-67286: HIVE-28102: Iceberg: Invoke validateDataFilesExist for RowDelta operations.
- CDPD-67120: HIVE-28084: Iceberg: COW fix for Merge operation
- DWX-15939: HIVE-27850: Iceberg: Addendum 2: Set runAs user in CompactionInfo
- CDPD-66414: HIVE-28057: Iceberg: Branches with non-lowercase characters can't be accessed.
- DWX-17513: HIVE-27980: Iceberg: Compaction: Add support for OPTIMIZE TABLE syntax
- DWX-17274: Cache Iceberg delete files in LLAP cache
- DWX-15939: HIVE-27850: Iceberg: Addendum: Use fully qualified table name in compaction query
- CDPD-65978: HIVE-27409: Iceberg: table with EXTERNAL type can not use statistics to optimize the query
- CDPD-65863: HIVE-27593: Iceberg: Keep Iceberg properties in sync with HMS
- CDPD-65867: HIVE-27360: Iceberg: Don't create the redundant MANAGED location when creating table without EXTERNAL keyword
- DWX-17270: HIVE-27991: Utilise FanoutWriters when inserting records in an Iceberg table when the records are unsorted
- CDPD-65654: Move the iceberg version dependency to parent pom
- CDPD-65755: HIVE-28016: Iceberg: NULL column values handling in COW mode
- DWX-13719: Iceberg positional deletes cache support

Impala - Iceberg changes

- IMPALA-13046: Update Iceberg mixed format deletes test
- IMPALA-13002: Iceberg V2 tables with Avro delete files aren't read properly
- IMPALA-12991: Eliminate unnecessary SORT for Iceberg DELETEs
- IMPALA-13003: Handle Iceberg AlreadyExistsException
- IMPALA-12612: SELECT * queries expand complex type columns from Iceberg metadata tables
- IMPALA-12893: Update tests for Iceberg 1.4.3
- IMPALA-12611: Add support to MAP type Iceberg Metadata table columns
- IMPALA-12600: Schema evolution with equality delete files
- IMPALA-12609: Implement SHOW METADATA TABLES IN statement to list Iceberg Metadata tables
- IMPALA-12809: Iceberg metadata table scanner should always be scheduled to the coordinator
- IMPALA-12894: Fix optimized count(*) for Iceberg tables with dangling delete files
- IMPALA-12729: Allow creating primary keys for Iceberg tables
- IMPALA-12879: Conjunct not referring to table field causes ERROR for Iceberg table
- IMPALA-12893: Specify 'format-version' explicitly in Iceberg tests
- IMPALA-12894: Turn off the count(*) optimisation for V2 Iceberg tables
- IMPALA-12845: Crash with DESCRIBE on a complex type from an Iceberg table
- IMPALA-12610: Support reading ARRAY columns for Iceberg Metadata tables
- IMPALA-12598: Allow multiple equality field id lists for Iceberg tables
- IMPALA-12412: Support partition evolution in OPTIMIZE statement
- IMPALA-12808: test_iceberg_deletes_and_updates failed by timeout error
- IMPALA-12072: Include snapshot id of Iceberg tables in query plan / profile
- IMPALA-12787: Concurrent DELETE and UPDATE operations on Iceberg tables can be problematic

- IMPALA-12764: Fix Iceberg metadata table scan's LIMIT clause
- IMPALA-12742: Add interop tests for DATE-partitioned Iceberg tables
- IMPALA-12765: Balance consecutive partitions better for Iceberg tables
- IMPALA-12742: DELETE/UPDATE Iceberg table partitioned by DATE fails with error
- IMPALA-12708: An UPDATE creates 2 new snapshots in Iceberg tables
- IMPALA-11710: Table properties are not updated in Iceberg metadata files
- IMPALA-12704: Fix NPE when quering empty iceberg table's metadata
- IMPALA-12706: Fix nested struct querying for Iceberg metadata tables
- IMPALA-12495: Describe statement for Iceberg metadata tables

For more information about the core Apache Iceberg 1.4.3 changes, see the Apache Iceberg Release Notes.

Hue changelog for Cloudera Data Warehouse Public Cloud

Review the changes introduced in Hue for Cloudera Data Warehouse Public Cloud.

2024.0.18.4 (December 5, 2024) GA

- CDPD-58397: Integrate with model deployment
- CDPD-72754: vLLM integration with LLama

2024.0.18.2-4 (October 4, 2024)

No new features or fixes.

2024.0.18.1-1 (August 15, 2024)

No new features or fixes.

2024.0.18.0-206 (July 26, 2024)

New features

- General availability (GA) of the SQL AI Assistant
- Introduction of task server in Hue and significant improvement in the file upload functionality
- Improvements to the shared Hue service (Preview)

Improvements and bug fixes

- CDPD-68692: [beeswax] fixed the incorrect set_nulls in hive results
- CDPD-68893: [jb] Fix issues with Impala query detailed view in the Job Browser
- CDPD-68891: [jb] Fix issues where Hive was showing up in the job browser even though it was disabled
- CDPD-68683: [tb] Fix an issue where column comments were fetched for individual columns in the Table Browser
- CDPD-67359: [raz][s3] Fix encoding for special chars in directory name listing 1000+ keys
- CDPD-67711: CDPD-65521 DWX-17620 [raz][abfs] Improve url params encoding for RAZ ABFS client
- CDPD-68882: [editor] Replace the editor_autocomplete_timeout setting with disable_source_autocomplete
- CDPD-68875: [editor] Update the corrdinator URL pattern in Impala mini job browser
- CDPD-64979: [api] Provide configs to control CORS
- CDPD-67186: [frontend] Advance the 'AND' rule in the definition of 'between' syntax for Hive and Presto
- CDPD-39975: [api] Add new /rename public api
- CDPD-66185: [api] Add new /touch public api

- CDPD-67167: [api] Add new /mkdir public API
- CDPD-65034: Check csrf to see if it has value before setting it up
- CDPD-67174: [jdbc] Add a new Kyuubi jdbc connector
- CDPD-64884: [abfs] Fix none check for ABFSStat attributes
- CDPD-67175: [gs] Add missing gcs-oauth2-boto-plugin package for non-raz scenarios
- CDPD-62397: [core] Default to correct editor choice per user with default_app value
- CDPD-62397: [frontend] Make it possible to configure the default editor to use for all users in Hue
- CDPD-67341: [idbroker] Refactor IDBroker with improved HA support and giving more preference to RAZ when both are configured in Hue

Cloudera Runtime 1.0.0

Configuring Apache Hive Statistics

Date published: 2021-12-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Generating Hive statistics in Cloudera Data Warehouse..... 4
Generating and viewing Apache Hive statistics in Cloudera Data Warehouse..... 4
Statistics generation and viewing commands in Cloudera Data Warehouse..... 5

Generating Hive statistics in Cloudera Data Warehouse

A cost-based optimizer (CBO) generates efficient query plans. Hive does not use the CBO until you generate column statistics for tables. By default, Hive gathers only table statistics. You need to configure Hive to enable gathering of column statistics.

The CBO, powered by Apache Calcite, is a core component in the Hive query processing engine. The CBO optimizes plans for executing a query, calculates the cost, and selects the least expensive plan to use. In addition to increasing the efficiency of execution plans, the CBO conserves resources.

How the CBO works

After parsing a query, a process converts the query to a logical tree (Abstract Syntax Tree) that represents the operations to perform, such as reading a table or performing a JOIN. Calcite applies optimizations, such as query rewrite, JOIN re-ordering, JOIN elimination, and deriving implied predicates to the query to produce logically equivalent plans. Bushy plans provide maximum parallelism. Each logical plan is assigned a cost that is based on distinct, value-based heuristics.

The Calcite plan pruner selects the lowest-cost logical plan. Hive converts the chosen logical plan to a physical operator tree, optimizes the tree, and converts the tree to a Tez job for execution on the Hadoop cluster.

Explain plans

You can generate explain plans by running the EXPLAIN query command. An explain plan shows you the execution plan of a query by revealing the operations that occur when you run the query. Having a better understanding of the plan, you might rewrite the query or change Tez configuration parameters.

Generating and viewing Apache Hive statistics in Cloudera Data Warehouse

You can use statistics to optimize queries for improved performance. The cost-based optimizer (CBO) also uses statistics to compare query plans and choose the best one. By viewing statistics instead of running a query, you can often get answers to your data questions faster.

About this task

This task shows how to generate different types of statistics about a table.

Procedure

1. Launch a Hive shell or editor.
2. Gather table statistics for the non-partitioned table mytable:

```
ANALYZE TABLE mytable COMPUTE STATISTICS;
```

3. View table statistics you generated:

```
DESCRIBE EXTENDED mytable;
```

4. Gather column-level statistics for the table:

```
ANALYZE TABLE mytable COMPUTE STATISTICS FOR COLUMNS;
```

5. View column statistics for the col_name column in my_table in the my_db database:

```
DESCRIBE FORMATTED my_db.my_table col_name;
```

Related Information

Apache Hive Wiki language reference

Apache Hive Wiki - Statistics in Hive

Statistics generation and viewing commands in Cloudera Data Warehouse

You can manually generate table and column statistics, and then view statistics using Hive queries. By default, Hive generates table statistics, but not column statistics, which you must generate manually to make cost-based optimization (CBO) functional.

Commands for generating statistics

The following ANALYZE TABLE command generates statistics for tables and columns:

ANALYZE TABLE [table_name] COMPUTE STATISTICS;

Gathers table statistics for non-partitioned tables.

ANALYZE TABLE [table_name] PARTITION(partition_column) COMPUTE STATISTICS;

Gathers table statistics for partitioned tables.

ANALYZE TABLE [table_name] COMPUTE STATISTICS for COLUMNS [comma_separated_column_list];

Gathers column statistics for the entire table.

ANALYZE TABLE partition2 (col1="x") COMPUTE STATISTICS for COLUMNS;

Gathers statistics for the partition2 column on a table partitioned on col1 with key x.

Commands for viewing statistics

You can use the following commands to view table and column statistics:

DESCRIBE [EXTENDED] table_name;

View table statistics. The EXTENDED keyword can be used only if the hive.stats.autogather property is enabled in the hive-site.xml configuration file. Use the Cloudera Manager Safety Valve feature (see link below).

DESCRIBE FORMATTED [db_name.]table_name [column_name] [PARTITION (partition_spec)];

View column statistics.

Related Information

Custom Configuration (about Cloudera Manager Safety Valve)

Cloudera Runtime 1.0.0

Apache Hive metastore overview

Date published: 2021-12-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Hive storage in Public Cloud..... 4

Apache Hive storage in Public Cloud

Knowing the storage locations of Apache Hive data and metadata helps you troubleshoot problems and replicate data.

On Public Cloud, the Hive warehouse that stores Hive data is located in an object store, such as S3, by default. In the cloud, Hive uses HDFS merely for storing temporary files. The Hive Metastore (HMS) stores the schema for Hive tables. HMS uses a pre-installed MySQL database. You perform little, or no, configuration of HMS in the cloud.

Cloudera Runtime 1.0.0

Apache Hive Overview

Date published: 2021-12-01
Date modified: 2024-12-05

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Hive features in 4

Apache Tez and Hive LLAP..... 5

Apache Hive 3 in architectural overview..... 5

Apache Hive content roadmap.....6

Apache Hive features in

Major changes to Apache Hive 2.x improve Apache Hive 3.x transactions and security. Knowing the major differences between these versions is critical for SQL users, including those who use Apache Spark and Apache Impala.

Hive is a data warehouse system for summarizing, querying, and analyzing huge, disparate data sets.

ACID transaction processing

Hive 3 tables are ACID (Atomicity, Consistency, Isolation, and Durability)-compliant. Hive 3 write and read operations improve the performance of transactional tables. Atomic operations include simple writes and inserts, writes to multiple partitions, and multiple inserts in a single SELECT statement. A read operation is not affected by changes that occur during the operation. You can insert or delete data, and it remains consistent throughout software and hardware crashes. Creation and maintenance of Hive tables is simplified because there is no longer any need to bucket tables.

Materialized views

Because multiple queries frequently need the same intermediate roll up or joined table, you can avoid costly, repetitious query portion sharing, by precomputing and caching intermediate tables into views.

Query results cache

Hive filters and caches similar or identical queries. Hive does not recompute the data that has not changed. Caching repetitive queries can reduce the load substantially when hundreds or thousands of users of BI tools and web services query Hive.

Scheduled Queries

Using SQL statements, you can schedule Hive queries to run on a recurring basis, monitor query progress, temporarily ignore a query schedule, and limit the number running in parallel. You can use scheduled queries to start compaction and periodically rebuild materialized views, for example.

Security improvements

Apache Ranger secures Hive data by default. To meet demands for concurrency improvements, ACID support, render security, and other features, Hive tightly controls the location of the warehouse on a file system, or object store, and memory resources.

With Apache Ranger and Apache Hive ACID support, your organization will be ready to support and implement GDPR (General Data Protection Regulation).

Workload management at the query level

You can configure who uses query resources, how much can be used, and how fast Hive responds to resource requests. Workload management can improve parallel query execution, cluster sharing for queries, and query performance. Although the names are similar, Hive workload management queries are unrelated to Cloudera Workload XM for reporting and viewing millions of queries and hundreds of databases.

Connection Pooling

Hive supports HakariCP JDBC connection pooling.

Unsupported features

does not support the following features that were available in HDP and CDH platforms:

- Multiple insert overwrite queries that read data from a source table.
- CREATE TABLE that specifies a managed table location

  Do not use the LOCATION clause to create a managed table. Hive assigns a default location in the warehouse to managed tables.
- CREATE INDEX

  Hive builds and stores indexes in ORC or Parquet within the main table, instead of a different table, automatically. Set hive.optimize.index.filter to enable use (not recommended--use materialized views instead). Existing indexes are preserved and migrated in Parquet or ORC to during upgrade.

Apache Tez and Hive LLAP

Cloudera Data Warehouse supports low-latency analytical processing (LLAP) of Hive queries.

Hive uses Apache Tez to execute queries internally. Apache Tez provides the following execution modes:

- Container mode

  Every time you run a Hive query, Tez requests a container from YARN.
- LLAP mode

  Every time you run a Hive query, Tez asks the LLAP daemon for a free thread, and starts running a fragment.

Apache Tez provides the framework to run a job that creates a graph with vertexes and tasks. SQL semantics for deciding the query physical plan, which identifies how to execute the query in a distributed fashion, is based on Apache Tez. The entire execution plan is created under this framework. SQL syntax in Hive is the same irrespective of execution engine (mode) used in Hive.

Apache Tez does not have to start from the ground up, requesting a container and waiting for a JVM, to run a fragment in LLAP mode. LLAP mode provides dedicated capacity. Caching is automated. The primary difference between LLAP mode and container mode, is that in LLAP mode the LLAP executors are used to run query fragments.

Cloudera Data Warehouse, the Hive execution mode is LLAP. In Cloudera Data Hub on Cloudera Public Cloud and Cloudera Private Cloud Base, the Hive execution mode is container, and LLAP mode is not supported. When Apache Tez runs Hive in container mode, it has traditionally been called Hive on Tez.

Apache Hive 3 in architectural overview

Understanding Apache Hive 3 major design features, such as default ACID transaction processing, can help you use Hive to address the growing needs of enterprise data warehouse systems.

Data storage and access control

One of the major architectural changes to support Hive 3 design gives Hive much more control over metadata memory resources and the file system, or object store. The following architectural changes from Hive 2 to Hive 3 provide improved security:

- Tightly controlled file system and computer memory resources, replacing flexible boundaries: Definitive boundaries increase predictability. Greater file system control improves security.
- Optimized workloads in shared files and containers

Hive 3 is optimized for object stores in the following ways:

- Hive uses ACID to determine which files to read rather than relying on the storage system.
- In Hive 3, file movement is reduced from that in Hive 2.
- Hive caches metadata and data agressively to reduce file system operations

The major authorization model for Hive is Ranger. Hive enforces access controls specified in Ranger. This model offers stronger security than other security schemes and more flexibility in managing policies.

This model permits only Hive to access the Hive warehouse.

Transaction processing

You can deploy new Hive application types by taking advantage of the following transaction processing characteristics:

- Mature versions of ACID transaction processing:

  ACID tables are the default table type.

  ACID enabled by default causes no performance or operational overload.
- Simplified application development, operations with strong transactional guarantees, and simple semantics for SQL commands

  You do not need to bucket ACID tables.
- Materialized view rewrites
- Automatic query cache
- Advanced optimizations

Hive client changes

You can use the thin client Beeline for querying Hive from a client. You can run Hive administrative commands from the client. Beeline uses a JDBC connection to Hive to run commands. Hive parses, compiles, and runs operations. Beeline supports many of the command-line options that Hive CLI supported. Beeline does not support hive -e set key=value to configure the Hive Metastore.

You enter supported Hive CLI commands by invoking Beeline using the hive keyword, command option, and command. For example, hive -e set. Using Beeline instead of the thick client Hive CLI, which is no longer supported, has several advantages, including low overhead. Beeline does not use the entire Hive code base. A small number of daemons required to run queries simplifies monitoring and debugging.

Hive enforces allowlist and denylist settings that you can change using SET commands. Using the denylist, you can restrict memory configuration changes to prevent instability. Different Hive instances with different allowlists and denylists to establish different levels of stability.

Apache Hive Metastore sharing

Hive, Impala, and other components can share a remote Hive metastore.

Query execution of batch and interactive workloads

You can connect to Hive using a JDBC command-line tool, such as Beeline, or using an JDBC/ODBC driver with a BI tool, such as Tableau. You configure the settings file for each instance to perform either batch or interactive processing.

Related Information

Hive Warehouse Connector for accessing Apache Spark data

Blog: Enabling high-speed Spark direct reader for Apache Hive ACID tables

Apache Hive content roadmap

The content roadmap provides links to the available content resources for Apache Hive.

Table 1: Apache Hive Content roadmap

Task	Resources	Source	Description
Understanding	Presentations and Papers about Hive	Apache wiki	Contains meeting notes, presentations, and whitepapers from the Apache community.
Getting Started	Hive Tutorial	Apache wiki	Provides a basic overview of Apache Hive and contains some examples on working with tables, loading data, and querying and inserting data.
Developing	Materialized Views	Apache wiki	Covers accelerating query processing in Cloudera Data Warehouse by pre-computing summaries using materialized views.
	JdbcStorageHandler	Apache wiki	Describes how to read from a JDBC data source in Hive.
	Hive transactions	Apache wiki	Describes ACID operations in Hive.
	Hive Streaming API	Apache wiki	Explains how to use an API for pumping data continuously into Hive using clients such as NiFi and Flume.
	Hive Operators and Functions	Apache wiki	Describes the Language Manual UDF.
	Beeline: HiveServer2 Client	Apache wiki	Describes how to use the Beeline client.
Reference	SQL Language Manual	Apache wiki	Language reference documentation available in the Apache wiki.
Contributing	Hive Developer FAQ	Apache wiki	Resources available if you want to contribute to the Apache community.
	How to Contribute	Apache wiki	
	Hive Developer Guide	Apache wiki	
	Plug-in Developer Kit	Apache wiki	
	Unit Test Parallel Execution	Apache wiki	
	Hive Architecture Overview	Apache wiki	
	Hive Design Docs	Apache wiki	
	Project Bylaws	Apache wiki	

Cloudera Runtime 7.2.11

Working with Apache Hive metastore

Date published: 2019-08-21
Date modified: 2021-09-08

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

HMS table storage.....4

HMS table storage

You need to understand how the Hive metastore (HMS) stores Hive tables when you run a CREATE TABLE statement or migrate a table to Cloudera. The success or failure of the statement, the resulting table type, and the table location depends on a number of factors.

HMS table transformations

The HMS includes the following Hive metadata about tables that you create:

- A table definition
- Column names
- Data types
- Comments in a central schema repository

When you use the EXTERNAL keyword in the CREATE TABLE statement, HMS stores the table as an external table. When you omit the EXTERNAL keyword and create a managed table, or ingest a managed table, HMS might translate the table into an external table or the table creation can fail, depending on the table properties. An important table property that affects table transformations is the ACID or Non-ACID table type:

Non-ACID

Table properties do not contain any ACID related properties set to true. For example, the table does not contain such properties transactional=true or insert_only=true.

ACID

Table properties do contain one or more ACID properties set to true.

Full ACID

Table properties contain transactional=true but not insert_only=true

Insert-only ACID

Table properties contain insert_only=true.

The following matrix shows the table type and whether or not the location property is supported.

ACID	Managed	Location Property	Comments	Action
Non-ACID	Yes	Yes	Migrated to Cloudera, for example from an HDP or CDH cluster.	Table stored as external
Non-ACID	Yes	No	Table location is null	Table stored in subdirectory of external warehouse*

* metastore.warehouse.external.dir

HMS detects type of client for interacting with HMS, for example Hive or Spark, and compares the capabilities of the client with the table requirement. HMS performs the following actions, depending on the results of the comparison:

Table requirement	Client meets requirements	Managed Table	ACID table type	Action
Client can write to any type of ACID table	No	Yes	Yes	CREATE TABLE fails
Client can write to full ACID table	No	Yes	insert_only=true	CREATE TABLE fails
Client can write to insert-only ACID table	No	Yes	insert_only=true	CREATE TABLE fails

If, for example, a Spark client does not have the capabilities required, the following type of error message appears:

```
Spark has no access to table `mytable`. Clients can access this table only i
f
they have the following capabilities: CONNECTORREAD,HIVEFULLACIDREAD, HIVE
FULLACIDWRITE,
HIVEMANAGESTATS, HIVECACHEINVALIDATE, . . .
```

Cloudera Runtime 7.2.11

Apache Hive Performance Tuning

Date published: 2019-08-21
Date modified: 2021-09-08

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Query results cache 4

ORC file format 4

Performance improvement using partitions 6

Apache Tez and Hive LLAP 6

Bucketed tables in Hive 7

Query results cache

Hive filters and caches similar or identical queries in the query results cache. Caching repetitive queries can reduce the load substantially when hundreds or thousands of users of BI tools and web services query Hive.

Some operations support vast user populations, who connect to Hive using BI systems such as Tableau. In these situations, repetitious queries are inevitable. The query results cache, which is on by default, filters and stores common queries in a cache. When you issue the query again, Hive retrieves the query result from a cache instead of recomputing the result, which takes a load off the backend system.

Every query that runs in Hive 3 stores its result in a cache. Hive evicts invalid data from the cache if the input table changes. For example, if you preform aggregation and the base table changes, queries you run most frequently stay in cache, but stale queries are evicted. The query results cache works with managed tables only because Hive cannot track changes to an external table. If you join external and managed tables, Hive falls back to executing the full query. The query results cache works with ACID tables. If you update an ACID table, Hive reruns the query automatically.

Configuring the query results cache

You can enable and disable the query results cache from the command line. You might want to do so to debug a query. You disable the query results cache in HiveServer by setting the following parameter to false: SET hive.query.results.cache.enabled=false;

By default, Hive allocates 2GB for the query results cache. You can change this setting by configuring the following parameter in bytes: hive.query.results.cache.max.size

ORC file format

You can conserve storage in a number of ways, but using the Optimized Row Columnar (ORC) file format for storing Apache Hive data is most effective. ORC is the default storage for Hive data.

The ORC file format for Hive data storage is recommended for the following reasons:

- Efficient compression: Stored as columns and compressed, which leads to smaller disk reads. The columnar format is also ideal for vectorization optimizations.
- Fast reads: ORC has a built-in index, min/max values, and other aggregates that cause entire stripes to be skipped during reads. In addition, predicate pushdown pushes filters into reads so that minimal rows are read. And Bloom filters further reduce the number of rows that are returned.

- Proven in large-scale deployments: Facebook uses the ORC file format for a 300+ PB deployment.

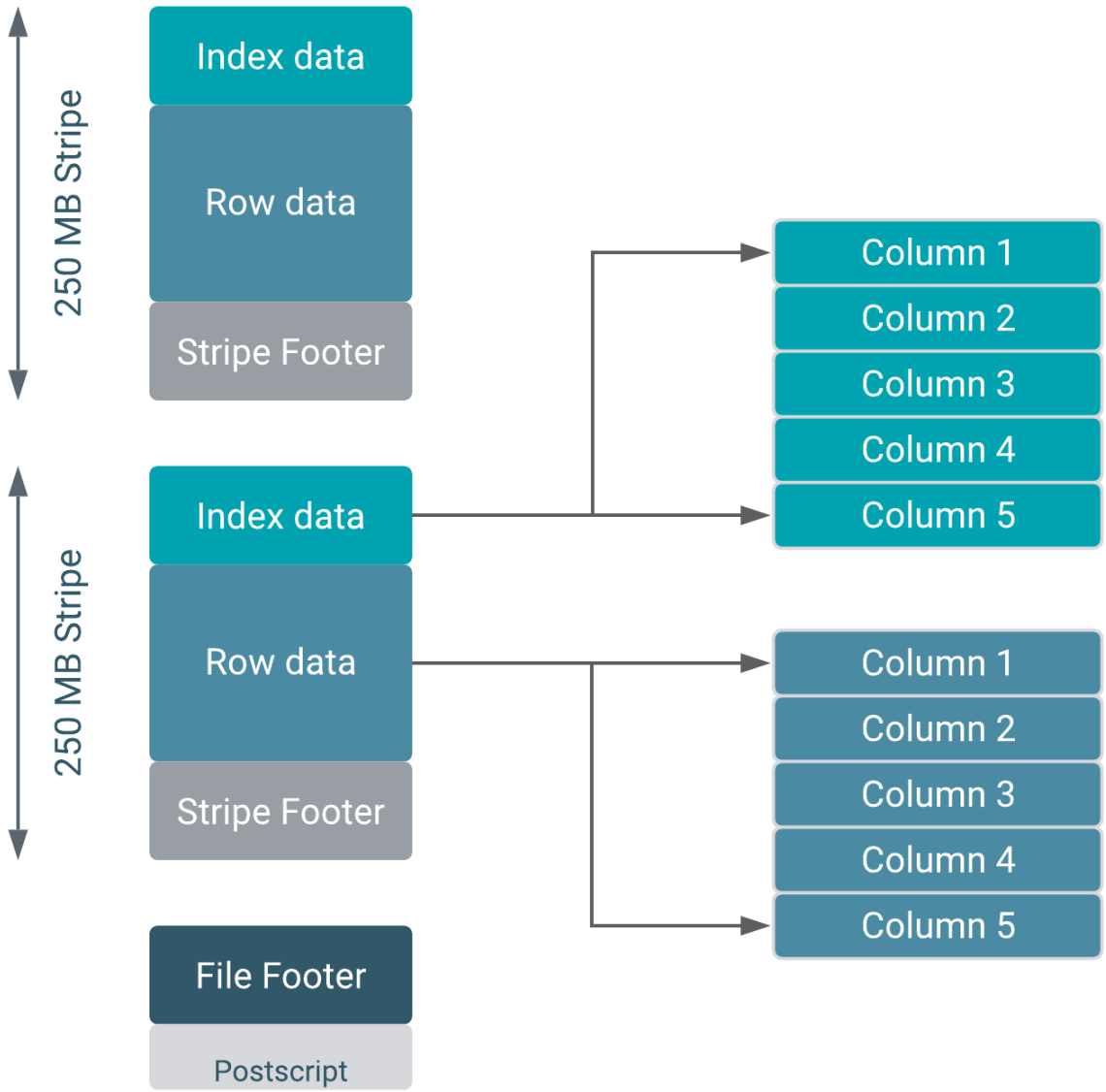

ORC provides the best Hive performance overall. In addition, to specifying the storage format, you can also specify a compression algorithm for the table, as shown in the following example:

```
CREATE TABLE addresses (
 name string,
 street string,
 city string,
 state string,
 zip int
 ) STORED AS orc TBLPROPERTIES ("orc.compress"="Zlib");
```

Setting the compression algorithm is usually not required because your Hive settings include a default algorithm. Using ORC advanced properties, you can create bloom filters for columns frequently used in point lookups.

Hive supports Parquet and other formats for insert-only ACID tables and external tables. You can also write your own SerDes (Serializers, Deserializers) interface to support custom file formats.

Performance improvement using partitions

You must be aware of partition pruning benefits, how to enable dynamic partitioning, and the configuration required for bulk-loading of data to ensure significant improvements in performance. You can use partitions to significantly improve performance.

You can design Hive table and materialized views partitions to map to physical directories on the file system/object store. For example, a table partitioned by date-time can organize data loaded into Hive each day. Large deployments can have tens of thousands of partitions. Partition pruning occurs indirectly when Hive discovers the partition key during query processing. For example, after joining with a dimension table, the partition key might come from the dimension table. A query filters columns by partition, limiting scanning that occurs to one or a few matching partitions. Partition pruning occurs directly when a partition key is present in the WHERE clause. Partitioned columns are virtual, not written into the main table because these columns are the same for the entire partition.

You do not need to specify dynamic partition columns. Hive generates a partition specification if you enable dynamic partitions.

Configuration for loading 1 to 9 partitions:

```
SET hive.exec.dynamic.partition.mode=nonstrict;
SET hive.exec.dynamic.partition=true;
```

For bulk-loading data into partitioned ORC tables, you use the following property, which optimizes the performance of data loading into 10 or more partitions.

Configuration for loading 10 or more partitions:

```
hive.optimize.sort.dynamic.partition=true
```

Apache Tez and Hive LLAP

Cloudera Data Warehouse supports low-latency analytical processing (LLAP) of Hive queries.

Hive uses Apache Tez to execute queries internally. Apache Tez provides the following execution modes:

- Container mode

  Every time you run a Hive query, Tez requests a container from YARN.
- LLAP mode

  Every time you run a Hive query, Tez asks the LLAP daemon for a free thread, and starts running a fragment.

Apache Tez provides the framework to run a job that creates a graph with vertexes and tasks. SQL semantics for deciding the query physical plan, which identifies how to execute the query in a distributed fashion, is based on Apache Tez. The entire execution plan is created under this framework. SQL syntax in Hive is the same irrespective of execution engine (mode) used in Hive.

Apache Tez does not have to start from the ground up, requesting a container and waiting for a JVM, to run a fragment in LLAP mode. LLAP mode provides dedicated capacity. Caching is automated. The primary difference between LLAP mode and container mode, is that in LLAP mode the LLAP executors are used to run query fragments.

Cloudera Data Warehouse, the Hive execution mode is LLAP. In Cloudera Data Hub on Cloudera Public Cloud and Cloudera Private Cloud Base, the Hive execution mode is container, and LLAP mode is not supported. When Apache Tez runs Hive in container mode, it has traditionally been called Hive on Tez.

Bucketed tables in Hive

If you migrated data from earlier Apache Hive versions to Hive 3, you might need to handle bucketed tables that impact performance. Review how Cloudera simplifies handling buckets. You learn about best practices for handling dynamic capabilities.

You can divide tables or partitions into buckets, which are stored in the following ways:

- As files in the directory for the table.
- As directories of partitions if the table is partitioned.

Specifying buckets in Hive 3 tables is not necessary. In Cloudera, Hive 3 buckets data implicitly, and does not require a user key or user-provided bucket number as earlier versions (ACID V1) did. For example:

V1:

```
CREATE TABLE hello_acid (load_date date, key int, value int)
CLUSTERED BY(key) INTO 3 BUCKETS
STORED AS ORC TBLPROPERTIES ('transactional'='true');
```

V2:

```
CREATE TABLE hello_acid_v2 (load_date date, key int, value int);
```

Performance of ACID V2 tables is on a par with non-ACID tables using buckets. ACID V2 tables are compatible with native cloud storage.

A common challenge related to using buckets in tables migrated from earlier versions is maintaining query performance while the workload or data scales up or down. For example, you could have an environment that operates smoothly using 16 buckets to support 1000 users, but a spike in the number of users to 100,000 for a day or two creates problems if you do not promptly tune the buckets and partitions. Tuning the buckets is complicated by the fact that after you have constructed a table with buckets, the entire table containing the bucketed data must be reloaded to reduce, add, or eliminate buckets.

In Cloudera, you only need to deal with the buckets of the biggest table. If workload demands change rapidly, the buckets of the smaller tables dynamically change to complete table JOINs.

Bucket configurations

You can enable buckets as follows:

```
SET hive.tez.bucket.pruning=true
```

When you load data into tables that are both partitioned and bucketed, set the hive.optimize.sort.dynamic.partition property to optimize the process:

```
 SET hive.optimize.sort.dynamic.partition=true
```

If you have 20 buckets on user_id data, the following query returns only the data associated with user_id = 1:

```
SELECT * FROM tab WHERE user_id = 1;
```

To best leverage the dynamic capability of table buckets, adopt the following practices:

- Use a single key for the buckets of the largest table.
- Usually, you need to bucket the main table by the biggest dimension table. For example, the sales table might be bucketed by customer and not by merchandise item or store. However, in this scenario, the sales table is sorted by item and store.

- Normally, do not bucket and sort on the same column.
- A table that has more bucket files than the number of rows is an indication that you should reconsider how the table is bucketed.

Cloudera Runtime 1.0.0

Hue Overview

Date published: 2021-12-01
Date modified: 2024-12-05

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Hue overview **4**

About Hue Query Processor **6**

Hue overview

Hue is a web-based interactive query editor that enables you to interact with databases and data warehouses. Data architects, SQL developers, and data engineers use Hue to create data models, clean data to prepare it for analysis, and to build and test SQL scripts for applications. Hue is integrated with Apache Hive and Apache Impala. You can access Hue from the Cloudera Data Warehouse Virtual Warehouses.

Cloudera Data Warehouse 1.1.2-b1520 offers the combined abilities of Data Analytics Studio (DAS) such as intelligent query recommendation, query optimization, and query debugging framework, and rich query editor experience of Hue, making Hue the next generation SQL assistant for Hive in Cloudera Data Warehouse.

Hue offers powerful execution, debugging, and self-service capabilities to the following key Big Data personas:

- Business Analysts
- Data Engineers
- Data Scientists
- Power SQL users
- Database Administrators
- SQL Developers

Business Analysts (BA) are tasked with exploring and cleaning the data to make it more consumable by other stakeholders, such as the data scientists. With Hue, they can import data from various sources and in multiple formats, explore the data using File Browser and Table Browser, query the data using the smart query editor, and create dashboards. They can save the queries, view old queries, schedule long-running queries, and share them with other stakeholders in the organization. They can also use Cloudera Data Visualization to get data insights, generate dashboards, and help make business decisions.

Data Engineers design data sets in the form of tables for wider consumption and for exploring data, as well as scheduling regular workloads. They can use Hue to test various Data Engineering (DE) pipeline steps and help develop DE pipelines.

Data scientists predominantly create models and algorithms to identify trends and patterns. They then analyze and interpret the data to discover solutions and predict opportunities. Hue provides quick access to structured data sets and a seamless interface to compose queries, search databases, tables, and columns, and execute query faster by leveraging Tez and LLAP. They can run ad hoc queries and start the analysis of data as pre-work for designing various machine learning models.

Power SQL users are advanced SQL experts tasked with analyzing and fine-tuning queries to improve query throughput and performance. They often strive to meet the TPC decision support (TPC-DS) benchmark. Hue enables them to run complex queries and provides intelligent recommendations to optimize the query performance. They can further fine-tune the query parameters by comparing two queries, viewing the explain plan, analyzing the Directed Acyclic Graph (DAG) details, and using the query configuration details. They can also create and analyze materialized views.

The Database Administrators (DBA) provide support to the data scientists and the power SQL users by helping them to debug long-running queries. They can download the query logs and diagnostic bundles, export and process log events to a database, or download it in JSON file format for further analysis. They monitor all the queries running in the system and terminate long-running queries to free up resources. Additionally, they may also provision databases, manage users, monitor and maintain the database and schemas for the Hue service, and provide solutions to scale up the capacity or migrate to other databases, such as PostgreSQL, Oracle, MySQL, and so on.

All Hue users can download logs and share them with their DBA or Cloudera Support for debugging and troubleshooting purposes.

SQL Developers can use Hue to create data sets to generate reports and dashboards that are often consumed by other Business Intelligence (BI) tools, such as Cloudera Data Visualization.

Hue can sparsely be used as a Search Dashboard tool, usually to prototype a custom search application for production environments.

For example, the following image shows a graphical representation of Impala SQL query results that you can generate with Hue:

Figure 1: Impala SQL query results generated with Hue

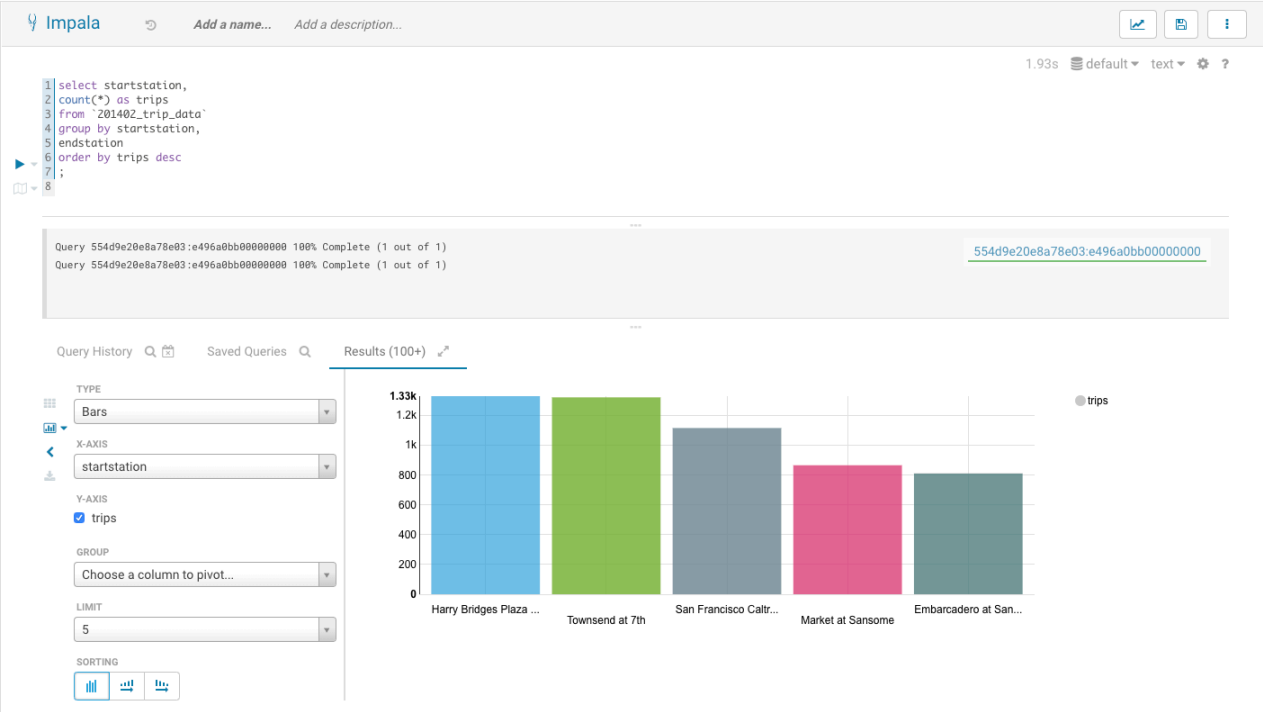

You can use Hue to:

- Explore, browse, and import your data through guided navigation in the left panel of the page.

  From the left panel, you can:

  - Browse your databases
  - Drill down to specific tables
  - Browse cloud storage
  - Discover indexes and HBase or Kudu tables
  - Find documents

  Objects can be tagged for quick retrieval, project association, or to assign a more "human-readable" name, if desired.
- Query your data, create a custom dashboard, or schedule repetitive jobs in the central panel of the page.

  The central panel of the page provides a rich toolset, including:

  - Versatile editors that enable you to create a wide variety of scripts.
  - Dashboards that you can create "on-the-fly" by dragging and dropping elements into the central panel of the Hue interface. No programming is required. Then you can use your custom dashboard to explore your data.
  - Schedulers that you can create by dragging and dropping, just like the dashboards feature. This feature enables you to create custom workflows and to schedule them to run automatically on a regular basis. A monitoring interface shows the progress, logs, and makes it possible to stop or pause jobs.
- Get expert advice on how to complete a task using the assistance panel on the right.

  The assistant panel on the right provides expert advice and hints for whatever application is currently being used in the central panel. For example, in the above image, Impala SQL hints are provided to help construct queries in the central panel.

- (Hive-only) View query details such as query information, visual explain, query timeline, query configuration, Directed Acyclic Graph (DAG) information, DAG flow, DAG swimlane, DAG counters, and DAG configurations.
- (Impala-only) View query details such as query type, duration, default database, request pool, CPU time, rows produced, peak memory, HDFS bytes read, coordinator details, execution plan, execution summary, and counters and metrics.
- (Hive-only) Terminate Hive queries.
- (Hive-only) Download debug bundles.
- Compare two queries.
- View query history.

About Hue Query Processor

The Hue Query Processor service indexes Hive and Impala query history and provides APIs to access them. Enabling the processor allows you to view Hive query history and query details on the Queries tab on the Hue Job Browser page.

The Hue Query Processor reads query event data from a Hadoop Compatible File System (HCFS), such as S3, and writes them into a relational database, such as PostgreSQL. The Query Processor pod is started when you create a Database Catalog AND IT runs at the Database Catalog level. Therefore, you can view query history from all Virtual Warehoues linked to a Database Catalog based on the dialect or the Virtual Warehouse type (Hive or Impala).

Related Information

Adding Query Processor Administrator users and groups in Cloudera Data Warehouse

1.0.0

Using Apache Iceberg

Date published: 2022-03-15
Date modified: 2024-12-05

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Iceberg features **5**
Alter table feature 5
Create table feature 6
Create table as select feature 8
Create partitioned table as select feature 8
Create table … like feature 9
Data compaction 9
Delete data feature 11
Describe table metadata feature 12
Drop partition feature 13
Drop table feature 14
Expire snapshots feature 14
Insert table data feature 15
Load data inpath feature 16
Load or replace partition data feature 17
Materialized view feature 17
Materialized view rebuild feature 18
Merge feature 21
Migrate Hive table to Iceberg feature 21
Changing the table metadata location 22
Migrate Hive table to Iceberg using Impala 22
Flexible partitioning 23
Partition evolution feature 23
Partition transform feature 24
Insert into/overwrite partition feature 26
Truncate partition feature 26
Query metadata tables feature 26
Rollback table feature 29
Row-level operations 29
Select Iceberg data feature 30
Schema evolution feature 31
Schema inference feature 32
Snapshot management 33
Branching feature 33
Tagging feature 34
Time travel feature 35
Truncate table feature 36
Update data feature 36
Best practices for Iceberg in Cloudera **37**
Migrating to V2 tables before row-level changes 38
Performance tuning **38**
Caching manifest files 38
Unsupported features and limitations **39**
Prerequisites **40**
Accessing Iceberg tables on S3 ..... **41**
Opening Ranger in Cloudera Data Warehouse 43
Editing a storage handler policy to access Iceberg files on the file system 43
Creating a policy to query an Iceberg table 45
Creating an Iceberg table **47**
Creating an Iceberg partitioned table **47**

Expiring snapshots 48
Inserting data into a table 49
Table migration overview 49
Migrating a Hive table to Iceberg 50
Selecting an Iceberg table 51
Running time travel queries 51
Updating an Iceberg partition 52
Test driving Iceberg from Impala 53
Hive demo data 54
Test driving Iceberg from Hive 57
Iceberg data types 59
Iceberg table properties 60

Apache Iceberg features

You can quickly build on your past experience with SQL to analyze Iceberg tables.

From Hive or Impala, you run SQL queries to create and query Iceberg tables. Impala queries are table-format agnostic. For example, Impala options are supported in queries of Iceberg tables. You can run nested, correlated, or analytic queries on all supported table types. Most Hive queries are also table-format agnostic.

This documentation does not attempt to show every possible query supported from Impala. Many examples of how to run queries on Iceberg tables from Impala are covered.

Supported ACID transaction properties

Iceberg supports atomic and isolated database transaction properties. Writers work in isolation, not affecting the live table, and perform a metadata swap only when the write is complete, making the changes in one atomic commit.

Iceberg uses snapshots to guarantee isolated reads and writes. You see a consistent version of table data without locking the table. Readers always see a consistent version of the data without the need to lock the table. Writers work in isolation, not affecting the live table, and perform a metadata swap only when the write is complete, making the changes in one atomic commit.

Iceberg partitioning

The Iceberg partitioning technique has performance advantages over conventional partitioning, such as Apache Hive partitioning. Iceberg hidden partitioning is easier to use. Iceberg supports in-place partition evolution; to change a partition, you do not rewrite the entire table to add a new partition column, and queries do not need to be rewritten for the updated table. Iceberg continuously gathers data statistics, which supports additional optimizations, such as partition pruning.

Iceberg uses multiple layers of metadata files to find and prune data. Hive and Impala keep track of data at the folder level and not at the file level, performing file list operations when working with data in a table. Performance problems occur during the execution of multiple list operations. Iceberg keeps track of a complete list of files within a table using a persistent tree structure. Changes to an Iceberg table use an atomic object/file level commit to update the path to a new snapshot file. The snapshot points to the individual data files through manifest files.

The manifest files track several data files across many partitions. These files store partition information and column metrics for each data file. A manifest list is an additional index for pruning entire manifests. File pruning increases efficiency.

Iceberg relieves Hive metastore (HMS) pressure by storing partition information in metadata files on the file system/object store instead of within the HMS. This architecture supports rapid scaling without performance hits.

Alter table feature

In Hive or Impala, you can use ALTER TABLE to set table properties. From Impala, you can use ALTER TABLE to rename a table, to change the table owner, or to change the role of the table owner. From Hive, you can alter the metadata location of the table if the new metadata does not belong to another table; otherwise, an exception occurs.

You can convert an Iceberg v1 table to v2 by setting a table property as follows:'format-version' = '2'.

Hive or Impala syntax

```
ALTER TABLE table_name SET TBLPROPERTIES table_properties;
```

- table_properties

  A list of properties and values using the following syntax:

  ```
  ('key' = 'value', 'key' = 'value', ... )
  ```

Impala syntax

```
ALTER TABLE table_name RENAME TO new_table_name;

ALTER TABLE table_name SET OWNER USER user_name;

ALTER TABLE table_name SET OWNER ROLE role_name;
```

Hive example

```
ALTER TABLE test_table SET TBLPROPERTIES('metadata_location'='s3a://bucketNa
me/ice_table/metadata/v1.metadata.json');
ALTER TABLE test_table2 SET TBLPROPERTIES('format-version' = '2');
```

Impala examples

```
ALTER TABLE t1 RENAME TO t2;

ALTER TABLE ice_table1 set OWNER USER john_doe;

ALTER TABLE ice_table2 set OWNER ROLE some_role;
ALTER TABLE ice_8 SET TBLPROPERTIES ('read.split.target-size'='268435456');

ALTER TABLE ice_table3 SET TBLPROPERTIES('format-version' = '2');
```

Create table feature

You use CREATE TABLE from Impala or CREATE EXTERNAL TABLE from Hive to create an external table in Iceberg. You learn the subtle differences in these features for creating Iceberg tables from Hive and Impala. You also learn about partitioning.

Hive and Impala handle external table creation a little differently, and that extends to creating tables in Iceberg. By default, Iceberg tables you create are v1. To create an Iceberg v2 table from Hive or Impala, you need to set a table property as follows:'format-version' = '2'.

Iceberg table creation from Hive

From Hive, CREATE EXTERNAL TABLE is recommended to create an Iceberg table in Cloudera.

When you use the EXTERNAL keyword to create the Iceberg table, by default only the schema is dropped when you drop the table. The actual data is not purged. Conversely, if you do not use EXTERNAL, by default the schema and actual data is purged. You can override the default behavior. For more information, see the Drop table feature.

From Hive, you can create a table that reuses existing metadata by setting the metadata_location table property to the object store path of the metadata. The operation skips generation of new metadata and re-registers the existing metadata. Use the following syntax:

```
CREATE EXTERNAL TABLE ice_fm_hive (i int) STORED BY ICEBERG TBLPROPERTIES ('
metadata_location'='<object store or file system path>')
```

See examples below.

Iceberg table creation from Impala

From Impala, CREATE TABLE is recommended to create an Iceberg table in Cloudera. Impala creates the Iceberg table metadata in the metastore and also initializes the actual Iceberg table data in the object store.

The difference between Hive and Impala with regard to creating an Iceberg table is related to Impala compatibility with Kudu, HBase, and other tables. For more information, see the Apache documentation, "Using Impala with Iceberg Tables".

Metadata storage of Iceberg tables

When you create an Iceberg table using CREATE EXTERNAL TABLE in Hive or using CREATE TABLE in Impala, HiveCatalog creates an HMS table and also stores some metadata about the table on your object store, such as S3. Creating an Iceberg table generates a metadata.json file, but not a snapshot. In the metadata.json, the snapshot-id of a new table is -1. Inserting, deleting, or updating table data generates a snapshot. The Iceberg metadata files and data files are stored in the table directory under the warehouse folder. Any optional partition data is converted into Iceberg partitions instead of creating partitions in the Hive Metastore, thereby removing the bottleneck.

To create an Iceberg table from Hive or from Impala, you associate the Iceberg storage handler with the table using one of the following clauses, respectively:

- Hive: STORED BY ICEBERG
- Impala: STORED AS ICEBERG or STORED BY ICEBERG

Supported file formats

You can write Iceberg tables in the following formats:

- From Hive: Parquet (default), Avro, ORC
- From Impala: Parquet

Impala supports writing Iceberg tables in only Parquet format. Impala does not support defining both file format and storage engine. For example, CREATE TABLE tbl ... STORED AS PARQUET STORED BY ICEBERG works from Hive, but not from Impala.

You can read Iceberg tables in the following formats:

- From Hive: Parquet, Avro, ORC
- From Impala: Parquet, Avro, ORC

Note: Reading Iceberg tables in Avro format from Impala is available as a technical preview. Cloudera recommends that you use this feature in test and development environments. It is not recommended for production deployments.

Hive syntax

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type, ... )]
  [PARTITIONED BY [SPEC]([col_name][, spec(value)][, spec(value)]...)]]
  [STORED AS file_format]
   STORED BY ICEBERG
  [TBLPROPERTIES ('key'='value', 'key'='value', ...)]
```

Impala syntax

```
CREATE TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type, ... )]
  [PARTITIONED BY [SPEC]([col_name][, spec(value)][, spec(value)]...)]]
   STORED {AS | BY} ICEBERG
  [TBLPROPERTIES (property_name=property_value, ...)]
```

Hive examples

```
CREATE EXTERNAL TABLE ice_1 (i INT, t TIMESTAMP, j BIGINT) STORED BY ICEBERG
;
CREATE EXTERNAL TABLE ice_2 (i INT, t TIMESTAMP) PARTITIONED BY (j BIGINT)
 STORED BY ICEBERG;
CREATE EXTERNAL TABLE ice_4 (i int) STORED AS ORC STORED BY ICEBERG;
CREATE EXTERNAL TABLE ice_5 (i int) STORED BY ICEBERG TBLPROPERTIES ('metad
ata_location'='s3a://bucketName/ice_table/metadata/v1.metadata.json')
CREATE EXTERNAL TABLE ice_6 (i int) STORED AS ORC STORED BY ICEBERG TBLPRO
PERTIES ('format-version' = '2');
```

Impala examples

```
CREATE TABLE ice_7 (i INT, t TIMESTAMP, j BIGINT) STORED BY ICEBERG; //creat
es only the schema
CREATE TABLE ice_8 (i INT, t TIMESTAMP) PARTITIONED BY (j BIGINT) STORED BY
 ICEBERG; //creates schema and initializes data
CREATE TABLE ice_v2 (i INT, t TIMESTAMP) PARTITIONED BY (j BIGINT) STORED BY
 ICEBERG TBLPROPERTIES ('format-version' = '2'); //creates a v2 table
```

Related Information

Drop table feature

Partition transform feature

Create table as select feature

You can create an Iceberg table based on an existing Hive or Impala table.

The create table as select (CTAS) query can optionally include a partitioning spec for the table being created.

Hive examples

```
CREATE EXTERNAL TABLE ctas STORED BY ICEBERG AS SELECT i, t, j FROM ice_1;
```

Impala examples

```
CREATE TABLE ctas STORED BY ICEBERG AS SELECT i, b FROM ice_11;
```

Create partitioned table as select feature

You can create a partitioned Iceberg table by selecting another table. You see an example of how to use PARTITIONED BY and TBLPROPERTIES to declare the partition spec and table properties for the new table.

You see an example of using a partition transform with the PARTITIONED BY SPEC clause.

The newly created table does not inherit the partition spec and table properties from the source table in SELECT. The Iceberg table and the corresponding Hive table is created at the beginning of the query execution. The data is inserted / committed when the query finishes. So for a transient period the table exists but contains no data.

Hive syntax

```
CREATE [EXTERNAL] TABLE prod.db.sample
    USING iceberg
```

```
    PARTITIONED BY (part)
    TBLPROPERTIES ('key'='value')
    AS SELECT ...
```

Hive examples

```
CREATE EXTERNAL TABLE ctas STORED BY ICEBERG AS SELECT i, t, j FROM ice_1;

CREATE EXTERNAL TABLE ctas_part PARTITIONED BY(z) STORED BY ICEBERG TBLPROPE
RTIES ('format-version'='2')
AS SELECT x, ts, z FROM t;
CREATE EXTERNAL TABLE ctas_part_spec PARTITIONED BY SPEC (month(d)) STORED
BY ICEBERG TBLPROPERTIES ('format-version'='2')
AS SELECT x, ts, d FROM source_t;
```

Impala examples

```
CREATE TABLE ctas STORED BY ICEBERG AS SELECT i, b FROM ice_11;

CREATE TABLE ctas_part PARTITIONED BY(b) STORED BY ICEBERG AS SELECT i, s, b
 FROM ice_11;
CREATE TABLE ctas_part_spec PARTITIONED BY SPEC (month(d)) STORED BY ICEBERG
 TBLPROPERTIES ('format-version'='2')
AS SELECT x, ts, d FROM source_t;
```

Create table … like feature

You learn by example how to create an empty table based on another table.

From Hive or Impala, you can create an Iceberg table schema based on another table. The table contains no data. The table properties of the original table are carried over to the new table definition. The following examples show how to use this feature:

Hive example

```
CREATE EXTERNAL TABLE target LIKE source STORED BY ICEBERG;
```

Impala example

```
CREATE TABLE target LIKE source STORED BY ICEBERG;
```

Data compaction

From Hive and Impala, you can compact Iceberg tables and optimize them for read operations. Compaction is an essential table maintenance activity that creates a new snapshot, which contains the table content in a compact form.

Frequent updates and row-level modifications on Iceberg tables can result in many small data files and delete files, which have to be merged-on-read. This degrades the query performance over time. You can use the following Hive and Impala SQL statements to compact Iceberg tables and optimize the table for reading.

Impala Syntax:

```
OPTIMIZE TABLE [db_name.]table_name;
```

Impala Example:

```
OPTIMIZE TABLE ice_table;
```

Hive Syntax:

```
ALTER TABLE [database_name.]table_name COMPACT 'compaction_type' [AND WAIT];
```

```
OPTIMIZE TABLE [database_name.]table_name REWRITE DATA;
```

Hive Example:

```
ALTER TABLE ice_table COMPACT 'MAJOR';
```

```
OPTIMIZE TABLE ice_table REWRITE DATA;
```

Important: When Cloudera Data Warehouse and Cloudera Data Hub are deployed in the same environment and use the same Hive Metastore (HMS) instance, the Cloudera Data Hub compaction workers can inadvertently pick up Iceberg compaction tasks. Since Iceberg compaction is not yet supported in the latest Cloudera Data Hub version, the compaction tasks will fail when they are processed by the Cloudera Data Hub compaction workers.

In such a scenario where both Cloudera Data Warehouse and Cloudera Data Hub share the same HMS instance and there is requirement to run both Hive ACID and Iceberg compaction jobs, it is recommended that you use the Cloudera Data Warehouse environment for these jobs. If you want to run only Hive ACID compaction tasks, you can choose to use either the Cloudera Data Warehouse or Cloudera Data Hub environments.

If you want to run the compaction jobs without changing the environment, it is recommended that you use Cloudera Data Warehouse. To avoid interference from Cloudera Data Hub, change the value of the hive .compactor.worker.threads Hive Server (HS2) property in Cloudera Data Hub to '0'. This ensures that the compaction jobs are not processed by Cloudera Data Hub. You can find this property in Cloudera Manager Hive Configuration .

To perform table optimization, ensure that the following prerequisites are met:

- The user performing compaction must have the 'ALL' permissions on the table, which can be set through Ranger.
- Impala can only write Parquet files, therefore the write.format.default table property must be set to parquet. Hive can write both Parquet and ORC file formats.
- Impala cannot compact tables with complex data types.
- Impala cannot compact views or empty tables.

The OPTIMIZE TABLE statement rewrites the entire table, performing the following tasks:

- Compact small files into larger files
- Merge delta files created due to previously run DELETE and UPDATE operations
- Rewrite all files, converting them to the latest table schema
- Rewrite all partitions according to the latest partition specification

When an Iceberg table is optimized, a new snapshot is created where all the old files of the table are replaced with newly written files. The old table state and old files can still be queried using time travel, because the rewritten data and delete files are not removed physically. This can lead to the accumulation of unused files that belong to old snapshots. Use the *Expire Snapshots* feature to permanently remove the old files from the file system.

Note: Since the compaction process rewrites the entire table, the operation can take a long time to complete depending on the table size.

Related Information

Expire snapshots feature

Delete data feature

Cloudera supports row-level deletes in Iceberg V2 tables.

From Hive and Impala, you can use the following formatting types defined by the Iceberg Spec:

- position deletes
- equality deletes

Note: The equality deletes feature is in technical preview and not recommended for use in production deployments. Cloudera recommends that you try this feature in test and development environments.

From Hive and Impala, you can read/write position deletes, but you can only read equality deletes. You cannot write equality deletes.

If you use an Iceberg table in the following ways, equality deletes are streamed or ingested, respectively:

- As a sink for streaming data from Flink to Cloudera Data Warehouse Public Cloud
- As an Apache NiFi data flow to ingest data in Iceberg table format into Cloudera Data Warehouse

If you cannot read equality deletes, the output of the read is incorrect with regard to any deletions.

For equality deletes, you must be aware of the following considerations:

- If you are using Apache Flink or Apache NiFi to write equality deletes, then ensure that you provide a PRIMARY KEY for the table. This is required for engines to know which columns to write into the equality delete files.
- If the table is partitioned then the partition columns have to be part of the PRIMARY KEY
- For Apache Flink, the table should be in 'upsert-mode' to write equality deletes
- Partition evolution is not allowed for Iceberg tables that have PRIMARY KEYs

To create an Iceberg table for these use cases, see the following documentation:

- Creating Iceberg tables in SQL Stream Builder
- Ingesting data into Cloudera Data Warehouse using Iceberg

Position delete files contain the following information:

- file_path, which is a full URI
- pos, the file position of the row

Delete files are sorted by file_path and pos. The following table shows an example of delete files in a partitioned table:

Table 1:

Partition	Partition information	Delete file
YEAR(ts)=2022	ts_year=2022/data-abcd.parquet ts_year=2022/data-bcde.parquet	ts_year=2022/delete-wxyz.parquet
YEAR(ts)=2023	ts_year=2023/data-efgh.parquet	ts_year=2023/delete-hxkw.parquet
MONTH(ts)=2023-06	ts_month=2023-06/data-ijkl.parquet	ts_month=2023-06/delete-uzwd.parquet
MONTH(ts)=2023-07	ts_month=2023-07/data-mnop.parquet	ts_month=2023-07/delete-udqx.parquet

Inserting, deleting, or updating table data generates a snapshot.

You use a WHERE clause in your DELETE statement. For example:

```
delete from tbl_ice where a <= 2,1;
```

Hive and Impala evaluate rows from one table against a WHERE clause, and delete all the rows that match WHERE conditions. If you want delete all rows, use the Truncate feature. The WHERE expression is similar to the WHERE expression used in SELECT. The conditions in the WHERE clause can refer to any columns.

Concurrent operations that include DELETE do not introduce inconsistent table states. Iceberg runs validation checks to check for concurrent modifications, such as DELETE+INSERT. Only one will succeed. On the other hand, DELETE+DELETE, and INSERT+INSERT can both succeed, but in the case of a concurrent DELETE+UPDATE, UPDATE+UPDATE, DELETE+INSERT, UPDATE+INSERT from Hive, only the first operation will succeed.

From joined tables, you can delete all matching rows from one of the tables. You can join tables of any kind, but the table from which the rows are deleted must be an Iceberg table. The FROM keyword is required in this case, to separate the name of the table whose rows are being deleted from the table names of the join clauses.

Hive or Impala syntax

```
delete from tablename [where expression]
```

```
delete joined_tablename from [joined_tablename, joined_tablename2, ...] [ wh
ere expression ]
```

Hive or Impala examples

```
create external table tbl_ice(a int, b string, c int) stored by iceberg tblp
roperties ('format-version'='2');

insert into tbl_ice values (1, 'one', 50), (2, 'two', 51), (3, 'three', 5
2), (4, 'four', 53), (5, 'five', 54), (111, 'one', 55), (333, 'two', 56);

delete from tbl_ice where a <= 2,1;
```

The following example deletes 0, 1, or more rows of the table. If col1 is a primary key, 0 or 1 rows are deleted:

```
delete from ice_table where col1 = 100;
```

Describe table metadata feature

You can use certain Hive and Impala show and describe commands to get information about table metadata. You can also query metadata tables.

The following table lists SHOW and DESCRIBE commands supported by Hive and Impala.

Command Syntax	Description	SQL Engine Support
SHOW CREATE TABLE table_name	Reveals the schema that created the table.	Hive and Impala
SHOW FILES IN table_name	Lists the files related to the table.	Impala
SHOW PARTITIONS table_name	Returns the Iceberg partition spec, just the column information, not actual partitions or files.	Impala
DESCRIBE [EXTENDED] table_name	The optional EXTENDED shows all the metadata for the table in Thrift serialized form, which is useful for debugging.	Hive and Impala
DESCRIBE [FORMATTED] table_name	The optional FORMATTED shows the metadata in tabular format.	Hive
DESCRIBE HISTORY table_name [BETWEEN timestamp1 AND timestamp2]	Optionally limits the output history to a period of time.	Impala

Hive example

```
DESCRIBE t;
```

Hive output includes the following information:

col_name	data_type	comment
x	int	
y	int	
	NULL	NULL
# Partition Transform Information	NULL	NULL
# col_name	transform_type	NULL
y	IDENTITY	NULL

The output of DESCRIBE HISTORY includes the following columns about the snapshot. The first three are self-explanatory. The is_current_ancestor column value is TRUE if the snapshot is the ancestor of the table:

- creation_time
- snapshot_id
- parent_id
- is_current_ancestor

Impala examples

```
DESCRIBE HISTORY ice_t FROM '2022-01-04 10:00:00';
DESCRIBE HISTORY ice_t FROM now() - interval 5 days;
DESCRIBE HISTORY ice_t BETWEEN '2022-01-04 10:00:00' AND '2022-01-05 10:00:0
0';
```

Drop partition feature

You can easily remove a partition from an Iceberg partition using an alter table statement from Impala.

Removing a partition does not affect the table schema. The column is not removed from the schema.

Prerequisites

- The filter expression in the drop partition syntax below is a combination of, or at least one of, the following predicates:
  - Minimum of one binary predicate
  - IN predicate
  - IS NULL predicate
- The argument of the predicate in the filter expression must be a partition transform, such as ‘YEARS(col-name)’ or ‘column’.
- Any non-identity transform must be included in the select statement. For example, if you partition a column by days, the filter must be days.

Impala syntax

```
alter table table-name drop partition (<filter expression>)
```

The following operators are supported in the predicate: =, !=,<, >, <=, >=

Impala example

```
CREATE TABLE ice_table PARTITIONED BY SPEC (day(d)) STORED BY ICEBERG TBLPRO
PERTIES ('format-version'='2') AS SELECT x, ts, d FROM source_t;

INSERT INTO ice_table(d) VALUES ('2024-04-30');

ALTER TABLE ice_table DROP PARTITION (day(d) = '2024-04-30');
```

Drop table feature

The syntax you use to create the table determines the default behavior when you drop the Iceberg table from Hive or Impala.

If you use CREATE TABLE, the external.table.purge flag is set to true. When the table is dropped, the contents of the table directory (actual data) are removed. If you use CREATE EXTERNAL TABLE from Hive, the external.table.purge flag is set to false. Dropping a table purges the schema only. The actual data is not removed. You can explicitly set the external.table.purge property to true to drop the data as well as the schema.

To prevent data loss during migration of a table to Iceberg, do not drop or move the table during migration. Exception: If you set the table property 'external.table.purge'='FALSE', no data loss occurs if you drop the table.

Hive or Impala syntax

```
DROP TABLE [IF EXISTS] table_name
```

Hive or Impala example

```
ALTER TABLE t SET TBLPROPERTIES('external.table.purge'='true');
DROP TABLE t;
```

Related Information

Create table feature

Expire snapshots feature

You can expire snapshots that Iceberg generates when you create or modify a table. During the lifetime of a table the number of snapshots of the table accumulate. You learn how to remove snapshots you no longer need.

You should periodically expire snapshots to delete data files that are no longer needed, and to reduce the size of table metadata. Each write to an Iceberg table from Hive creates a new snapshot, or version, of a table. Snapshots accumulate until expired.

You can expire snapshots based on the following conditions:

- All snapshots older than a timestamp or timestamp expression
- A snapshot having a given ID
- Snapshots having IDs matching a given list of IDs
- Snapshots within the range of two timestamps

You can keep snapshots you are likely to need, for example recent snapshots, and expire old snapshots. For example, you can keep daily snapshots for the last 30 days, then weekly snapshots for the past year, then monthly snapshots for the last 10 years. You can remove specific snapshots to meet GDPR “right to be forgotten” requirements.

Hive or Impala syntax

```
ALTER TABLE <table Name> EXECUTE EXPIRE_SNAPSHOTS(<timestamp expression>)

ALTER TABLE <table Name> EXECUTE EXPIRE_SNAPSHOTS('<Snapshot Id>')

ALTER TABLE <table Name> EXECUTE EXPIRE_SNAPSHOTS('<Snapshot Id1>,<Snapshot
 Id2>... ')

ALTER TABLE <table Name> EXECUTE EXPIRE_SNAPSHOTS BETWEEN (<timestamp exp
ression>) AND (<timestamp expression>)
```

Hive or Impala example

The first example removes snapshots having a timestamp older than August 15, 2022 1:50 pm. The second example removes snapshots from 10 days ago and before.

```
ALTER TABLE ice_11 EXECUTE EXPIRE_SNAPSHOTS('2022-15-08 13:50:00');
ALTER TABLE ice_t EXECUTE EXPIRE_SNAPSHOTS(now() - interval 10 days);
```

Preventing snapshot expiration

You can prevent expiration of recent snapshots by configuring the history.expire.min-snapshots-to-keep table property. You can use the alter table feature to set a property. The history.expire.min-snapshots-to-keep property refers to a number of snapshots, not a time delta. For example, assume you always want to keep all snapshots of your table for the last 24 hours. You configure history.expire.min-snapshots-to-keep as a safety mechanism to enforce this. If your table receives only one modification (insert / update / merge) per hour, then setting history.expire.min-snapshots-to-keep = 24 is sufficient to meet your requirement. However, if your table was consistently receiving updates every minute, then the last 24 hour period would entail 1440 snapshots, and the history.expire.min-snapshots-to-keep setting would need to be configured appropriately.

Table data and orphan maintenance

The contents of the table directory (actual data) might, or might not, be removed when you drop the table. An orphan data file can remain when you drop an Iceberg table, depending on the external.table.purge flag table property. An orphaned data file is one that has contents in the table directory, but no snapshot.

Expiring a snapshot does not remove old metadata files by default. You must clean up metadata files using writ e.metadata.delete-after-commit.enabled=true and write.metadata.previous-versions-max table properties. For more information, see "Iceberg table properties" below. Setting this property controls automatic metadata file removal after metadata operations, such as expiring snapshots or inserting data.

Related Information

Iceberg table properties

Insert table data feature

From Hive and Impala, you can insert data into Iceberg tables using the standard INSERT INTO a single table. INSERT statements work for V1 and V2 tables.

You can replace data in the table with the result of a query. To replace data, Hive and Impala dynamically overwrite partitions that have rows returned by the SELECT query. Partitions that do not have rows returned by the SELECT query, are not replaced. Using INSERT OVERWRITE on tables that use the BUCKET partition transform is not recommended. Results are unpredictable because dynamic overwrite behavior would be too random in this case.

From Hive, Cloudera also supports inserting into multiple tables as a technical preview; however, this operation is not atomic, so data consistency of Iceberg tables is equivalent to that of Hive external tables. Changes within a single table will remain atomic.

Inserting, deleting, or updating table data generates a snapshot. A new snapshot corresponds to a new manifest list. Manifest lists are named snap-*.avro.

Iceberg specification defines sort orders. At this point, Hive doesn't support defining sort orders. But if there are sort orders defined by using other engines Hive can utilize them on write operations. For more information about sorting, see sort orders specification.

Hive or Impala syntax

```
INSERT INTO TABLE tablename VALUES values_row [, values_row ...]

INSERT INTO TABLE tablename1 select_statement1 FROM tablename2

INSERT OVERWRITE TABLE tablename1 select_statement1 FROM tablename2
```

Hive or Impala examples

```
CREATE TABLE ice_10 (i INT, s STRING, b BOOLEAN) STORED BY ICEBERG;
INSERT INTO ice_10 VALUES (1, 'asf', true);
CREATE TABLE ice_11 (i INT, s STRING, b BOOLEAN) STORED BY ICEBERG;
INSERT INTO ice_11 VALUES (2, 'apache', false);
INSERT INTO ice_11 SELECT * FROM ice_10;
SELECT * FROM ice_11;
INSERT OVERWRITE ice_11 SELECT * FROM ice_10;
```

Hive example

```
FROM customers
    INSERT INTO target1 SELECT customer_id, first_name;
    INSERT INTO target2 SELECT last_name, customer_id;
```

Load data inpath feature

From Impala, you can load Parquet or ORC data from a file in a directory on your file system or object store into an Iceberg table. You might need to set the mem_limit or pool configuration (max-query-mem-limit, min-query-mem-limit) to accommodate the load.

Impala syntax

```
LOAD DATA INPATH '<path to file>' INTO table t;
```

Impala example

In this example, you create a table using the LIKE clause to point to a table stored as Parquet. This is required for Iceberg to infer the schema. You also load data stored as ORC.

```
CREATE TABLE test_iceberg LIKE my_parquet_table STORED AS ICEBERG;
SET MEM_LIMIT=1MB;
LOAD DATA INPATH 's3a://testbucket/files/my_data.parq' INTO TABLE test_ice
berg;

LOAD DATA INPATH 's3a://testbucket/files/my_data.orc' INTO TABLE test2_ic
eberg;
```

Load or replace partition data feature

There is no difference in the way you insert data into a partitioned or unpartitioned Iceberg table.

Working with partitions is easy because you write the query in the same way for the following operations:

- Insert into, or replace, an unpartitioned table
- Insert into, or replace, an identity partitioned table
- Insert into, or replace, a transform-partitioned table

Do not use INSERT OVERWRITE on tables that went through partition evolution. Truncate such tables first, and then INSERT the tables.

Impala example

```
CREATE TABLE ice_12 (i int, s string, t timestamp, t2 timestamp) STORED BY I
CEBERG;
```

Hive or Impala examples

```
INSERT INTO ice_12 VALUES (42, 'impala', now(), to_date(now()));
INSERT OVERWRITE ice_t VALUES (42, 'impala', now(), to_date(now()));
```

Materialized view feature

Using a materialized view can accelerate query execution. Creating a materialized view on top of Iceberg tables in Cloudera can further accelerate the performance. You can create a materialized view of an Iceberg V1 or V2 table based on an existing Hive or Iceberg table.

The materialized view is stored in Hive ACID or Iceberg format. Materialized view source tables either must be native ACID tables or must support table snapshots. Automatic rewriting of a materialized view occurs under the following conditions:

- The view definition contains the following operators only:
  - TableScan
  - Project
  - Filter
  - Join(inner)
  - Aggregate
- Source tables are native ACID or Iceberg v1 or v2
- The view is not based on time travel queries because those views do not have up-to-date data by definition.

Hive example

The following example creates a materialized view of an Iceberg table from Hive.

```
drop table if exists tbl_ice;
create external table tbl_ice(a int, b string, c int) stored by iceberg st
ored as orc tblproperties ('format-version'='2');

create materialized view mat1 as
select b, c from tbl_ice for system_version as of 5422037307753150798;
```

The following example creates a materialized view of two Iceberg tables. Joined tables must be in the same table format, either Iceberg or Hive ACID.

```
drop table if exists tbl_ice2;
create table tbl_ice2(a int, b string, c int) stored as orc TBLPROPERTIES ('
transactional'='true');
INSERT INTO tbl_ice2 VALUES (2, 'apache', 3);

drop table if exists tbl_ice;
create external table tbl_ice(a int, b string, c int) stored by iceberg sto
red as orc tblproperties ('format-version'='2');
INSERT INTO tbl_acid VALUES (4, 'iceberg', 5);
create materialized view mat1 as
select tbl_ice2.b, tbl_ice2.c from tbl_ice join tbl_ice2 on tbl_ice.a = tb
l_ice2.a;
```

The following example uses explain to examine a materialized view and then creates a materialized view of an Iceberg V1 table from Hive.

```
drop materialized view if exists mat1;
drop table if exists tbl_ice;

create table tbl_ice(a int, b string, c int) stored by iceberg stored as orc
 tblproperties ('format-version'='1');
insert into tbl_ice values (1, 'one', 50), (2, 'two', 51), (3, 'three', 52),
 (4, 'four', 53), (5, 'five', 54);
explain create materialized view mat1 stored by iceberg stored as orc tblpr
operties ('format-version'='1') as
select tbl_ice.b, tbl_ice.c from tbl_ice where tbl_ice.c > 52;

create materialized view mat1 stored by iceberg stored as orc tblproperties
 ('format-version'='1') as
select tbl_ice.b, tbl_ice.c from tbl_ice where tbl_ice.c > 52;
```

Related Information

Materialized view rebuild feature

Materialized view rebuild feature

Updates to materialized view contents when new data is added to the underlying table are critical; otherwise, queries can return stale data.

An update can occur under the following conditions:

- As a row-level incremental rebuild of the view after inserting data into a table

  Source tables can be Iceberg V2 or Hive full ACID.
- As a full rebuild of the view

A full rebuild can be expensive. An incremental rebuild updates only the affected parts of the materialized view, decreasing rebuild step execution time.

An incremental rebuild occurs automatically when you insert (append) data into a source table; otherwise, after you make some other type of change, for example a delete, you must manually start a full rebuild.

You use the ALTER command to manually start a full rebuild of the materialized view from Hive as follows:

Hive syntax

```
ALTER MATERIALIZED VIEW <name of view> REBUILD;
```

Full rebuild example

In this example, first you set required properties. Next, you create Iceberg tables, a V1 table and a V2 table, from Hive. You insert data into the tables and create a materialized view of the joined tables. You insert some new values into one of the source tables, rendering the materialized view stale. Finally, you rebuild the materialized view using explain cbo to show the rebuild plan The rebuild plan indicates a full rebuild will occur, which means the definition query will be executed.

```
drop table if exists tbl_ice;
drop table if exists tbl_ice_v2;
create external table tbl_ice(a int, b string, c int) stored by iceberg sto
red as orc tblproperties ('format-version'='1');
create external table tbl_ice_v2(d int, e string, f int) stored by iceberg
stored as orc tblproperties ('format-version'='2');
insert into tbl_ice_v2 values (1, 'one v2', 50), (4, 'four v2', 53), (5, 'f
ive v2', 54);

create materialized view mat1 as
select tbl_ice.b, tbl_ice.c, tbl_ice_v2.e from tbl_ice
join tbl_ice_v2 on tbl_ice.a=tbl_ice_v2.d where tbl_ice.c > 52;
group by tbl_ice.b tbl_ice.c;

-- view should be empty
select * from mat1;

-- view is up-to-date, use it
explain cbo
select tbl_ice.b, tbl_ice.c, tbl_ice_v2.e from tbl_ice join tbl_ice_v2 on
tbl_ice.a=tbl_ice_v2.d where tbl_ice.c > 52;

-- insert some new values to one of the source tables
insert into tbl_ice values (1, 'one', 50), (2, 'two', 51), (3, 'three', 52),
 (4, 'four', 53), (5, 'five', 54);

-- view is outdated, cannot be used
explain cbo
select tbl_ice.b, tbl_ice.c, tbl_ice_v2.e from tbl_ice join tbl_ice_v2 on tb
l_ice.a=tbl_ice_v2.d where tbl_ice.c > 52;

explain cbo
alter materialized view mat1 rebuild;

-- view should contain data
select * from mat1;

-- view is up-to-date again, use it
explain cbo
select tbl_ice.b, tbl_ice.c, tbl_ice_v2.e from tbl_ice join tbl_ice_v2 on tb
l_ice.a=tbl_ice_v2.d where tbl_ice.c > 52;
group by tbl_ice.b tbl_ice.c;
```

Automatic query rewrite example

In this example, you create an Iceberg table, insert some values, and create the materialized view. The view is partitioned using a partition specification and stored in the Iceberg ORC format. The v1 format version is specified

in this example (the v2 format is also supported). You then look at a description of the view and see that the query rewrite option is enabled by default. An automatic incremental rebuild is possible when this option is enabled.

```
   drop table if exists tbl_ice;
   create external table tbl_ice(a int, b string, c int) stored by iceberg
 stored as orc tblproperties ('format-version'='1');

   -- insert some new values into one of the source tables

   insert into tbl_ice values (1, 'one', 50), (2, 'two', 51), (3, 'three', 5
2), (4, 'four', 53), (5, 'five', 54);

   explain
   create materialized view mat1 partitioned on spec (bucket(16, b), trunc
ate(3, c)) stored by Iceberg stored as orc tblproperties(‘format-version’=’1
’)as
   select tbl_ice.b, tbl)ice.c from tbl_ice where tbl_ice.c > 52;

   -- the output query plan query indicates a rewrite is enabled

   POSTHOOK: query: explain
   create materialized view mat1 …
   …
   STAGE PLANS
  …
-- In stage one, the materialized view is created by calling the Iceberg
API to create the table object.
rewrite enabled
…
```

You use the DESCRIBE command to see the output query plan, which shows details about the view, including if it can be used in automatic query rewrites.

```
-- check the materialized view details
    describe formatted mat1;
    …
    #col_name     data_type      comment
    b             string
    c             int

    #Partition Transform Information
    #col_name     transform_type
    b             bucket(16)
    c             TRUNCATE[3]


    #detailed table information
    …
    Table_type:    MATERIALIZED_VIEW
    Table Parameters:
    …
    current-snapshot-id   563939E424367334713
    …
    metadata_location     …
    …
    Table_type            Iceberg
    …
    #Materialized View Information
    Original Query:  …
    Expanded Query:  …
    Rewrite Enabled: Yes
```

With the query rewrite option enabled, you insert data into the source table, and incremental rebuild occurs automatically. You do not need to rebuild the view manually before running queries.

Related Information

Materialized view feature

Merge feature

You can perform actions on an Iceberg table based on the results of a join with a v2 Iceberg table.

Hive syntax

```
MERGE INTO <target table> AS T USING <source expression/table> AS S
ON <boolean expression1>
WHEN MATCHED [AND <boolean expression2>] THEN UPDATE SET <set clause list>
WHEN MATCHED [AND <boolean expression3>] THEN DELETE
WHEN NOT MATCHED [AND <boolean expression4>] THEN INSERT VALUES <value list>
```

Hive example

Use the MERGE INTO statement to update an Iceberg table based on a staging table:

```
MERGE INTO customer USING (SELECT * FROM new_customer_stage) sub ON sub.id =
 customer.id
    WHEN MATCHED THEN UPDATE SET name = sub.name, state = sub.new_state
    WHEN NOT MATCHED THEN INSERT VALUES (sub.id, sub.name, sub.state);
```

Create an Iceberg table and merge it with a non-Iceberg table.

```
create external table target_ice(a int, b string, c int) partitioned by spec
 (bucket(16, a), truncate(3, b)) stored by iceberg stored as orc tblproperti
es ('format-version'='2');
create table source(a int, b string, c int);
...

merge into target_ice as t using source src ON t.a = src.a
when matched and t.a > 100 THEN DELETE
when matched then update set b = 'Merged', c = t.c + 10
when not matched then insert values (src.a, src.b, src.c);
```

Migrate Hive table to Iceberg feature

Cloudera supports Hive table migration from Hive and Impala to Iceberg tables using ALTER TABLE to set the table properties.

Note: Do not drop or move the old table during a migration operation. Doing so will delete the data files of the old and new tables. Exception: If you set the table property 'external.table.purge'='FALSE', no data loss occurs when you drop the table.

In-place table migration process

In-place table migration saves time generating Iceberg tables. There is no need to regenerate data files. Only metadata, which points to source data files, is regenerated.

To convert a Hive table to an Iceberg V1 table, use the following syntax:

```
ALTER TABLE table_name CONVERT TO ICEBERG;
```

To convert a Hive table to an Iceberg V2 table, you must run two queries. Use the following syntax:

```
ALTER TABLE table_name CONVERT TO ICEBERG

ALTER TABLE table_name SET TBLPROPERTIES ('format-version' = '2'
    ...)
```

In-place table migration saves time generating Iceberg tables. There is no need to regenerate data files. Only metadata, which points to source data files, is regenerated.

Changing the table metadata location

From Hive and Impala, you can change the table metadata location, also known as the snapshot location.

Configuring security of Iceberg file metadata

In this release, you can prevent overrides of the Iceberg file metadata location by unauthorized users. If you do not prevent overrides, an unauthorized party who knows the underlying schema and file location outside the table location can rewrite the manifest files within one table location to point to the data files in another table location to read your data. You accept the default (true), or set a property in the Virtual Warehouse to true to prevent overrides:

- Hive property: hive.iceberg.allow.datafiles.in.table.location.only

  ```
  SET hive.iceberg.allow.datafiles.in.table.location.only = true;
  ```

- Impala property: iceberg_restrict_data_file_location

  ```
  SET iceberg_restrict_data_file_location = true;
  ```

Prerequisites for changing the metadata location

- The new location must contain exactly the same metadata json file as the old location.
- Before changing the metadata location, you must migrate the table to Iceberg.

After migrating a table to Iceberg, you can change the metadata location using ALTER TABLE as shown below:

Hive and Impala example

```
ALTER TABLE test_table SET TBLPROPERTIES('metadata_location'='s3a://bucketN
ame/ice_table/metadata/v1.metadata.json');
```

Migrate Hive table to Iceberg using Impala

Cloudera supports Hive table migration from Impala to Iceberg tables using ALTER TABLE.

Requirements

- The original table is an EXTERNAL table from the Impala perspective. (‘EXTERNAL’ table property value is true)
- The original table is a non-ACID table.
- You have “ALL” privileges on the database containing the table.

Note: Do not drop, move, or change the old table during a migration operation. Doing so will delete the data files of the old and new tables.

Limitations

- Due to a bug in Apache Iceberg it is not possible to migrate tables that are partitioned by string columns having a partition value that contains the forward slash '/' character. For more information see https://github.com/apache/iceberg/issues/7612
- Some column types supported by Hive tables are not supported by Iceberg tables, such as tinyint or smallint. Tables with such columns are not migrated to Iceberg.
- The original Hive table must be in AVRO, ORC, or Parquet format.

In-place table migration process

In-place table migration saves time generating Iceberg tables. There is no need to regenerate data files. Only metadata, which points to source data files, is regenerated. As a result of an in-place table migration a new Iceberg table is created using the name and the location of the old Hive table. The old Hive table is dropped during the process.

Impala syntax example

To convert a Hive table to an Iceberg V1 table from Impala, use the following syntax:

```
ALTER TABLE table_name CONVERT TO ICEBERG;
```

To convert a Hive table to an Iceberg V2 table from Impala you must run two queries. Use the following syntax:

```
ALTER TABLE table_name CONVERT TO ICEBERG

ALTER TABLE table_name SET TBLPROPERTIES ('format-version' = '2'
...)
```

Flexible partitioning

Iceberg partition evolution, which is a unique Iceberg feature, and the partition transform feature, greatly simplify partitioning tables and changing partitions.

Partitions based on transforms are stored in the Iceberg metadata layer, not in the directory structure. You can change the partitioning completely, or just refine existing partitioning, and write new data based on the new partition layout--no need to rewrite existing data files. For example, change a partition by month to a partition by day.

Use partition transforms, such as IDENTITY, TRUNCATE, BUCKET, YEAR, MONTH, DAY, HOUR. Iceberg solves scalability problems caused by having too many partitions. Partitioning can also include columns with a large number of distinct values. Partitioning is hidden in the Iceberg metadata layer, eliminating the need to explicitly write partition columns (YEAR, MONTH for example) or to add extra predicates to queries for partition pruning.

```
SELECT * FROM tbl WHERE ts = '2023-04-21 20:56:08'
AND YEAR = 2023 AND MONTH = 4 AND DAY = 21
```

Year, month, and day can be automatically extracted from '2023-04-21 20:56:08' if the table is partitioned by DAY(ts)

Partition evolution feature

Partition evolution means you can change the partition layout of the table without rewriting existing data files. Old data files can remain partitioned by the old partition layout, while newly added data files are partitioned based on the new layout.

You can use the ALTER TABLE SET PARTITION SPEC statement to change the partition layout of an Iceberg table. A change to the partition spec results in a new metadata.json and a commit, but does not create a new snapshot.

Hive or Impala syntax

```
ALTER TABLE table_name SET PARTITION SPEC ([col_name][, spec(value)][, spec(
value)]...)]
```

- spec

  The specification for a transform listed in the next topic, "Partition transform feature".

Hive or Impala examples

```
ALTER TABLE t
SET PARTITION SPEC ( TRUNCATE(5, level), HOUR(event_time),
BUCKET(15, message), price);
ALTER TABLE ice_p
SET PARTITION SPEC (VOID(i), VOID(d), TRUNCATE(3, s), HOUR(t), i);
```

Related Information

Partition transform feature

Partition transform feature

From Hive or Impala, you can use one or more partition transforms to partition your data. Each transform is applied to a single column. Identity-transform means no transformation; the column values are used for partitioning. The other transforms apply a function to the column values and the data is partitioned by the transformed values.

Using CREATE TABLE ... PARTITIONED BY you create identity-partitioned Iceberg tables. Identity-partitioned Iceberg tables are similar to the Hive or Impala partitioned tables, which are stored in the same directory structure as the data files. Iceberg stores the partitioning columns of identity-partitioned Iceberg tables in a different directory structure from the data files if the tables are migrated to Iceberg from Hive external tables. Iceberg handles the tables and files regardless of the location.

Hive and Impala support Iceberg advanced partitioning through the PARTITION BY SPEC clause. Using this clause, you can define the Iceberg partition fields and partition transforms.

The following table lists the available transformations of partitions and corresponding transform spec.

Transformation	Spec	Supported by SQL Engine
Partition by year	years(time_stamp) | year(time_stamp)	Hive and Impala
Partition by month	months(time_stamp) | month(time_stamp)	Hive and Impala
Partition by a date value stored as int (dateint)	days(time_stamp) | date(time_stamp)	Hive
Partition by hours	hours(time_stamp)	Hive
Partition by a dateint in hours	date_hour(time_stamp)	Hive
Partition by hashed value mod N buckets	bucket(N, col)	Hive and Impala
Partition by value truncated to L, which is a number of characters	truncate(L, col)	Hive and Impala

Strings are truncated to length L. Integers and longs are truncated to bins. For example, truncate(10, i) yields partitions 0, 10, 20, 30 …

The idea behind transformation partition by hashed value mod N buckets is the same as hash bucketing for Hive tables. A hashing algorithm calculates the bucketed column value (modulus). For example, for 10 buckets, data is stored in column value % 10, ranging from 0-9 (0 to n-1) buckets.

You use the PARTITIONED BY SPEC clause to partition a table by an identity transform.

Hive syntax

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type][, time_stamp TIMESTAMP] )]
  [PARTITIONED BY SPEC([col_name][, spec(value)][, spec(value)]...)]
  [STORED AS file_format]
   STORED BY ICEBERG
  [TBLPROPERTIES (property_name=property_value, ...)]
```

Where spec(value)represents one or more of the following transforms:

- YEARS(col_name)
- MONTHS(col_name)
- DAYS(col_name)
- BUCKET(bucket_num,col_name)
- TRUNCATE(length, col_name)

Impala syntax

```
CREATE TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type, ... )]
  [PARTITIONED BY SPEC([col_name][, spec(value)][, spec(value)]...)]
   STORED (AS | BY) ICEBERG
  [TBLPROPERTIES (property_name=property_value, ...)]
```

Where spec(value) represents one or more of the following transforms:

- YEARS(col_name)
- MONTHS(col_name)
- DAYS(col_name)
- BUCKET(bucket_num,col_name)
- TRUNCATE(length, col_name)

Hive example

The following example creates a top level partition based on column i, a second level partition based on the hour part of the timestamp, and a third level partition based on the first 1000 characters in column j.

```
CREATE EXTERNAL TABLE ice_3 (i INT, t TIMESTAMP, j BIGINT) PARTITIONED BY SP
EC (i, HOUR(t), TRUNCATE(1000, j)) STORED BY ICEBERG;
```

Impala examples

```
CREATE TABLE ice_13 (i INT, t TIMESTAMP, j BIGINT) PARTITIONED BY SPEC (i, H
OUR(t), TRUNCATE(1000, j)) STORED BY ICEBERG;
```

The following examples show how to use the PARTITION BY SPEC clause in a CREATE TABLE query from Impala.The same transforms are available in a CREATE EXTERNAL TABLE query from Hive.

```
CREATE TABLE ice_t(id INT, name STRING, dept STRING)
PARTITIONED BY SPEC (bucket(19, id), dept)
STORED BY ICEBERG
TBLPROPERTIES ('format-version'='2');
```

```
CREATE TABLE ice_ctas
PARTITIONED BY SPEC (truncate(1000, id))
STORED BY ICEBERG
```

```
TBLPROPERTIES ('format-version'='2')
AS SELECT id, int_col, string_col FROM source_table;
```

Related Information

Creating an Iceberg partitioned table

Create table feature

Partition evolution feature

Insert into/overwrite partition feature

From Hive you can insert into, or overwrite data in, Iceberg tables that are statically or dynamically partitioned.

You can also insert data into a table using static and dynamic partitions. INSERT OVERWRITE queries do not work on tables that have undergone partition evolution.

Static partition syntax

```
INSERT INTO|OVERWRITE TABLE name PARTITION(col = val) VALUES (...)
INSERT INTO|OVERWRITE TABLE name PARTITION(col = val) SELECT <expression>
```

Dynamic partition syntax

```
INSERT INTO|OVERWRITE TABLE name PARTITION(col) VALUES (...)
INSERT INTO|OVERWRITE TABLE name PARTITION(col) SELECT <expression>
```

Static and dynamic partition syntax

Insertion into the static partition must precede the insertion into the dynamic partition.

```
INSERT INTO|OVERWRITE TABLE name PARTITION(col1 = val, col2) VALUES (...)
INSERT INTO|OVERWRITE TABLE name PARTITION(col1 = val, col2) SELECT <expre
ssion>
```

Truncate partition feature

You can truncate Iceberg partitioning from Hive.

You specify columns in the TRUNCATE TABLE ... PARTITION command as shown in the following syntax. This command does not support partitioning on columns that have partition transforms other than identity-transform.

Hive syntax

```
TRUNCATE TABLE t PARTITION (parCol1 = Val1, parCol2 = Val2 ... );
```

Query metadata tables feature

Apache Iceberg stores extensive metadata for its tables. From Hive and Impala, you can query the metadata tables as you would query a regular table. For example, you can use projections, joins, filters, and so on.

Iceberg metadata tables include information that is useful for efficient table maintenance (about snapshots, manifests, data, delete files, etc.) as well as statistics that help query engines plan and execute queries more efficiently (value count, min-max values, number of NULLs, etc.).

Note: Iceberg metadata tables are read-only. You cannot add, remove, or modify records in the tables. Also, you cannot drop or create new metadata tables.

For more information about the Apache Iceberg Iceberg metadata table types, see the *Apache Iceberg MetadataTableType enumeration*.

For more information about querying Iceberg metadata, see the *Apache Iceberg Spark documentation*.

The following sections describe how you can interact with and query Iceberg metadata tables:

List metadata tables

The SHOW METADATA TABLES statement lists all metadata tables belonging to an Iceberg table. You can also filter the tables according to a specific pattern.

Note: The SHOW METADATA TABLES statement is only available in Impala and is not supported in Hive.

Impala Syntax:

```
SHOW METADATA TABLES IN [database_name.]table_name [[LIKE] "pattern"];
```

Impala Example:

```
SHOW METADATA TABLES IN default.ice_table;

Output:
+----------------------+
| name                 |
+----------------------+
| all_data_files       |
| all_delete_files     |
| all_entries          |
| all_files            |
| all_manifests        |
| data_files           |
| delete_files         |
| entries              |
| files                |
| history              |
| manifests            |
| metadata_log_entries |
| partitions           |
| position_deletes     |
| refs                 |
| snapshots            |
+----------------------+
```

Query metadata tables

You can use the regular SELECT statement from Hive or Impala to query Iceberg metadata tables. To reference a metadata table, use the fully qualified path as shown in the syntax.

Hive or Impala Syntax:

```
SELECT … FROM database_name.table_name.metadata_table_name;
```

Hive or Impala Example:

```
SELECT * FROM default.ice_table.files;
```

You can select any subset of the columns or all of them using '*'. In comparison to regular tables, running a SELECT * from Impala on metadata tables always includes complex-typed columns in the result. The Impala query option,

EXPAND_COMPLEX_TYPES only applies to regular tables. However, Hive always includes complex columns irresepctive of whether SELECT queries are run on regular tables or metadata tables.

For Impala queries that have a mix of regular tables and metadata tables, a SELECT * expression where the sources are metadata tables always includes complex types, whereas for SELECT * expressions where the sources are regular tables, complex types are included only if the EXPAND_COMPLEX_TYPES query option is set to 'true'.

In the case of Hive, columns with complex types are always included.

You can also filter the result set using a WHERE clause, use aggregate functions such as MAX or SUM, JOIN metadata tables with other metadata tables or regular tables, and so on.

Example:

```
SELECT
    s.operation,
    h.is_current_ancestor,
    s.summary
FROM default.ice_table.history h
JOIN default.ice_table.snapshots s
  ON h.snapshot_id = s.snapshot_id
WHERE s.operation = 'append'
ORDER BY made_current_at;
```

Limitations

- Impala does not support the DATE and BINARY data types. NULL is returned instead of their actual values.
- Impala does not support unnesting collections from metadata tables.

DESCRIBE metadata tables

Like regular tables, Iceberg metadata tables have schemas that can be explored using the DESCRIBE statement. The DESCRIBE statement displays metadata about a table, such as the column names and their data types.

To reference the metadata table, use the fully qualified path as shown in the syntax.

Note: DESCRIBE FORMATTED|EXTENDED is not available for metadata tables. In Impala, using this statement results in an error whereas Hive displays the same output as the regular DESCRIBE statement.

Hive or Impala Syntax:

```
DESCRIBE database_name.table_name.metadata_table_name;
```

Hive or Impala Example:

```
DESCRIBE default.ice_table.history;
Output:
+---------------------+-----------+---------+----------+
| name                | type      | comment | nullable |
+---------------------+-----------+---------+----------+
| made_current_at     | timestamp |         | true     |
| snapshot_id         | bigint    |         | true     |
| parent_id           | bigint    |         | true     |
| is_current_ancestor | boolean   |         | true     |
+---------------------+-----------+---------+----------+
```

Related Information

Apache Iceberg MetadataTableType

Apache Spark documentation

Rollback table feature

In the event of a problem with your table, you can reset a table to a good state as long as the snapshot of the good table is available. You can roll back the table data based on a snapshot id or a timestamp.

When you modify an Iceberg table, a new snapshot of the earlier version of the table is created. When you roll back a table to a snapshot, a new snapshot is created. The creation date of the new snapshot is based on the Timezone of your session. The snapshot id does not change.

Hive and Impala syntax

```
ALTER TABLE test_table EXECUTE rollback(snapshotID);
ALTER TABLE test_table EXECUTE rollback('timestamp');
```

Hive and Impala examples

The following example rolls back to an earlier table, creating a new snapshot having a new creation date timestamp, but keeping the same snapshot id 3088747670581784990.

```
ALTER TABLE ice_t EXECUTE ROLLBACK(3088747670581784990);
```

The following example rolls the table back to the latest snapshot having a creation timestamp earlier than '2022-08-08 00:00:00'.

```
ALTER TABLE ice_7 EXECUTE ROLLBACK('2022-08-08 00:00:00')
```

Row-level operations

Hive supports the copy-on-write (COW) as well as merge-on-read (MOR) modes for handling Iceberg row-level updates and deletes. Impala supports only the MOR mode and will fail if configured for copy-on-write. Impala does support reading copy-on-write tables.

Depending on the COW or MOR setting, Hive performs updates and deletes to Iceberg tables as follows:

- COW: Hive creates a new version of files for each update/delete. Use COW when updating/deleting a large number of rows, or when reading data frequently.
- MOR (default): Updates/deletes are logged to delta files, which tends to be faster than creating new versions of the files. Later a compaction can eliminate the delete files and rewrite the affected data files.

Impala uses only the MOR method. Impala does not support copy-on-write and will fail if configured for copy-on-write. Impala does support reading copy on write tables.

Configuring Hive tables to support COW or MOR

You use the CREATE TABLE or ALTER commands to set either COW or MOR mode in table properties as shown in the following examples:

```
ALTER TABLE ice01 SET TBLPROPERTIES ('write.delete.mode'='copy-on-write');
ALTER TABLE ice01 SET TBLPROPERTIES ('write.update.mode'='copy-on-write');
```

When to use COW or MOR

Set either COW or MOR based on your use case and rate of data change. Consider the following advantages and disadvantages of the modes:

- MOR
  - Writes are efficient.
  - Reads are inefficient due to read amplification, but regularly scheduled compaction can reduce inefficiency.
  - A good choice when streaming.
  - A good choice when frequently writing or updating, such as running hourly batch jobs.
  - A good choice when the percentage of data change is low.

  COW
  - Reads are efficient.
  - A good choice for bulk updates and deletes, such as running a daily batch job.
  - Writes are inefficient due to write amplification, but the need for compaction is reduced.
  - A good choice when the percentage of data change is high.

Position or equality delete files

By default, Hive and Impala delete Iceberg V2 table data using position delete files. Hive and Impala can read equality deletes, which you might encounter in a table created and written to by Flink or another engine that supports writing equality deletes. Hive and Impala cannot write equality deletes.

Choose the MOR or COW mode based on your use case. The MOR mode keeps track of deleted rows, so it's expensive when you're making many deletions.

If you use an Iceberg table in the following ways, consider using the COW mode:

- As a sink for streaming data from Flink to Cloudera Data Warehouse Public Cloud
- As an Apache NiFi data flow to ingest data in Iceberg table format into Cloudera Data Warehouse

Equality deletes are streamed or ingested, respectively, in these cases. If you cannot read equality deletes, the output of the read is incorrect with regard to any deletions.

To create an Iceberg table for these use cases, see the following documentation:

- Creating Iceberg tables in SQL Stream Builder
- Ingesting data into Cloudera Data Warehouse using Iceberg

The position and equality deletes are summarized below:

Position delete: DELETE operations find the data records that need to be deleted, then write out their file path and position into delete files. Postponing the rewrite of the files speeds writes of updates/deletes. Read operations need to merge the data and delete files to retrieve the active rows.

Equality delete: DELETE operations write out column values (for example, primary keys) that identify the records to be deleted. If you already know the column values of the records you want to delete, then there is no need to read the whole table, which means equality deletes can be written faster than position deletes. Read operations need to merge the data and delete files to retrieve the active rows. Processing equality deletes is typically slower than processing position deletes.

Select Iceberg data feature

You can read Iceberg tables from Impala as you would any table. Joins, aggregations, and analytical queries, for example, are supported.

Impala supports reading V2 tables with position deletes.

Hive or Impala example

```
SELECT * FROM ice_t;

SELECT count(*) FROM ice_t i LEFT OUTER JOIN other_t b
```

```
ON (i.id = other_t.fid)
WHERE i.col = 42;
```

Schema evolution feature

You learn that the Hive or Impala schema changes when the associated Iceberg table changes. You see examples of changing the schema.

Although you can change the schema of your table over time, you can still read old data files because Iceberg uniquely identifies schema elements. A schema change results in a new metadata.json and a commit, but does not create a new snapshot.

The Iceberg table schema is synchronized with the Hive/Impala table schema. A change to the schema of the Iceberg table by an outside entity, such as Spark, changes the corresponding Hive/Impala table. You can change the Iceberg table using ALTER TABLE to make the following changes:

From Hive:

- Add a column
- Replace a column
- Change a column type or its position in the table

From Impala:

- Add a column
- Rename a column
- Drop a column
- Change a column type

An unsafe change to a column type, which would require updating each row of the table for example, is not allowed. The following type changes are safe:

- int to long
- float to double
- decimal(P, S) to decimal(P', S) if precision is increased

You can drop a column by changing the old column to the new column.

Hive syntax

```
ALTER TABLE table_name ADD COLUMNS (col_name type[, …])
ALTER TABLE table_name CHANGE COLUMN col_old_name col_new_name type

ALTER TABLE table_name CHANGE COLUMN col_old_name col_new_name type [FIRST|
AFTER col_name] [existing_col_name
]
ALTER TABLE table_name REPLACE COLUMNS (col_name type)
```

Impala syntax

```
ALTER TABLE table_name ADD COLUMNS(col_name type[, …])

ALTER TABLE table_name CHANGE COLUMN col_old_name col_new_name type
ALTER TABLE table_name DROP COLUMN col_name
```

Hive examples

```
ALTER TABLE t ADD COLUMNS(message STRING, price DECIMAL(8,1));
```

```
ALTER TABLE t REPLACE COLUMNS (i int comment '...', a string, ...);
ALTER TABLE t CHANGE COLUMN col_x col_x DECIMAL (22, 3) AFTER col_y;
```

Impala examples

```
ALTER TABLE ice_12 ADD COLUMNS(message STRING, price DECIMAL(8,1));
ALTER TABLE ice_12 DROP COLUMN i;

ALTER TABLE ice_12 CHANGE COLUMN s str STRING;
```

Schema inference feature

From Hive or Impala, you can base a new Iceberg table on a schema in a Parquet file. You see a difference in the Hive and Impala syntax and examples.

From Hive, you must use FILE in the CREATE TABLE LIKE ... statement. From Impala, you must omit FILE in the CREATE TABLE LIKE … statement. The column definitions in the Iceberg table are inferred from the Parquet data file when you create a table like Parquet from Hive or Impala. Set the following table property for creating the table:

```
hive.parquet.infer.binary.as = <value>
```

Where <value> is binary (the default) or string.

This property determines the interpretation of the unannotated Parquet binary type. Some systems expect binary to be interpreted as string.

Hive syntax

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name LIKE FILE PARQU
ET 'object_storage_path_of_parquet_file'
[PARTITIONED BY [SPEC]([col_name][, spec(value)][, spec(value)]...)]]
[STORED AS file_format]
STORED BY ICEBERG
[TBLPROPERTIES (property_name=property_value, ...)]
```

Impala syntax

```
CREATE TABLE [IF NOT EXISTS] [db_name.]table_name LIKE PARQUET 'object_stora
ge_path_of_parquet_file'
[PARTITIONED BY [SPEC]([col_name][, spec(value)][, spec(value)]...)]]
STORED (AS | BY) ICEBERG
[TBLPROPERTIES (property_name=property_value, ...)]
```

Hive example

```
CREATE TABLE ctlf_table LIKE FILE PARQUET 's3a://testbucket/files/schema.par
q'
STORED BY ICEBERG;
```

Impala example

```
CREATE TABLE ctlf_table LIKE PARQUET 's3a://testbucket/files/schema.parq'
STORED BY ICEBERG;
```

Snapshot management

From Hive, you can manage the lifecycle of snapshots using branches and tags. Branches are references to snapshots that have a lifecycle of their own. Tags identify snapshots you need for auditing and conforming to GDPR.

You can expire snapshots, list snapshots of a table, use a snapshot to rollback to a version of a table, and perform other snapshot management operations described in this documentation.

Managing snapshots using branches and tags

Iceberg branches and tags are available in Hive to reference snapshots for managing snapshot lifecycles. Branches track the ancestral relations of snapshots and point to the snapshot root. Cloudera recommends that you use this feature in test and development environments. It is not recommended for production deployments. Iceberg branches and tags are not supported in Impala or Spark.

Set the current snapshot from Hive

The following Hive syntax for sets the current snapshot using an ID or timestamp:

```
ALTER TABLE <table name> EXECUTE SET_CURRENT_SNAPSHOT (<snapshot ID>)
```

For example:

```
ALTER TABLE t EXECUTE SET_CURRENT_SNAPSHOT (7521248990126549311)
```

Related Information

Apache Hive-4.x with Iceberg Branches & Tags

Apache Iceberg documentation

Branching feature

Branches are references to snapshots that have a lifecycle of their own. You can create a branch by basing the branch on a snapshot ID, a timestamp, or the state of your table. Using the SNAPSHOT RETENTION clause, you can create a branch that limits the number of snapshots of a table.

Iceberg branching is available in Hive only. Iceberg branches are not supported in Impala or Spark.

List snapshots and branches of a table

The following syntax lists timestamps and IDs of snapshots of an Iceberg table. You can use the list of snapshots to create branches and tags.

```
SELECT * FROM <database>.<table name>.HISTORY
```

The following syntax lists the branches and tags of a table.

```
SELECT * from <database>.<table name>.REFS
```

Create a branch

Use either system version or system time in the following syntax from Hive to create a branch. Tables must be Iceberg V2 tables.

```
ALTER TABLE <table name> CREATE BRANCH <branch name> FOR SYSTEM_VERSION AS O
F <SNAPSHOT_ID>
ALTER TABLE <table name> CREATE BRANCH <branch name> FOR SYSTEM_TIME AS OF
'time_stamp' [expression]
```

If you do not have the ID or timestamp of a snapshot, you can also create a branch using the table name only and omitting the FOR clause

```
ALTER TABLE <table name> CREATE BRANCH <branch name>
```

This syntax creates a branch having the same state as the table.

Limit retained snapshots

When you create a branch, you can limit the number of snapshots a branch retains.

```
ALTER TABLE <table name> CREATE BRANCH <branch name> FOR SYSTEM_VERSION AS O
F <timestamp> WITH SNAPSHOT RETENTION <integer limit> SNAPSHOTS;
```

Query a branch

From Hive, you can ingest data into an Iceberg branch using dot notation as you would a SQL table. The branch name prefix branch_ must be lowercase.

```
INSERT INTO TABLE <database name>.<table name>.branch_<branch name> VALUES (
<column name>[, <column name> ...]
```

You can use SQL syntax to read, update, and delete data in a branch.

```
SELECT <column name 1>, <column name 2>, ... FROM <database name>.<table nam
e>.branch_<branch name>
UPDATE TABLE <database name>.<table name>.branch_<branch name> SET <column
name>=<new value>, <column name>=<new value> ... WHERE <condition>

DELETE FROM <database name>.<table name>.branch_<branch name> WHERE <conditi
on>
```

Fast forward a branch

Fast forwarding a branch updates the state of one branch to another branch within its hierarchy. For example, you can fast-forward branch x to branch z as shown in the following example:

```
ALTER TABLE <table name> EXECUTE FAST-FORWARD 'x' 'z';
```

Branch x must be an ancestor of branch z. If you omit the second branch name, the named branch is fast-forwarded to the current branch.

Delete a branch

You can delete a branch related to a particular table, using the following syntax:

```
ALTER TABLE <table name> DROP BRANCH [IF EXISTS] <branch name>
```

Related Information

Apache Hive-4.x with Iceberg Branches & Tags

Apache Iceberg documentation

Tagging feature

Tags identify snapshots you need for auditing and conforming to GDPR. You can tag a snapshot to help you track retention for a certain period of time.

Iceberg tagging is available in Hive only. Iceberg tagging is not available in Impala or Spark.

Create a tag

You can create a tag based on SYSTEM_VERSION, SYSTEM_TIME, or the current branch. Tables must be Iceberg V2 tables.

```
ALTER TABLE <table name> CREATE TAG <tag name> FOR SYSTEM_VERSION AS OF <sna
pshot ID>

ALTER TABLE <table name> CREATE TAG <tag name> FOR SYSTEM_TIME AS OF '<ti
mestamp>'

ALTER TABLE <table name> CREATE TAG <tag name>
```

Query a tag

You can run SQL read queries on tags using the name of the tag as follows:

```
<database name>.<table name>.tag_<tag name>
```

For example:

```
SELECT * from mydb.mytable.tag_mytag;
```

Delete a tag

Use the following syntax to delete a tag.

```
ALTER TABLE <table name> DROP tag [IF EXISTS] <tag name>
```

Related Information

Apache Hive-4.x with Iceberg Branches & Tags

Apache Iceberg documentation

Time travel feature

From Hive or Impala, you can run point in time queries for auditing and regulatory workflows on Iceberg tables. Time travel queries can be time-based or based on a snapshot ID.

Iceberg generates a snapshot when you create, or modify, a table. A snapshot stores the state of a table. You can specify which snapshot you want to read, and then view the data at that timestamp. In Hive, you can use projections, joins, and filters in time travel queries. You can add expressions to the timestamps, as shown in the examples. You can expire snapshots.

Snapshot storage is incremental and dependent on the frequency and scale of updates. By default, Hive and Impala use the latest snapshot. You can query an earlier snapshot of Iceberg tables to get historical information. Hive and Impala use the latest schema to query an earlier table snapshot even if it has a different schema.

Hive or Impala syntax

```
SELECT * FROM table_name FOR SYSTEM_TIME AS OF 'time_stamp' [expression]

SELECT * FROM table_name FOR SYSTEM_VERSION AS OF snapshot_id [expression]
```

- time_stamp

  The state of the Iceberg table at the time specified by the UTC timestamp.

- snapshot_id

  The ID of the Iceberg table snapshot from the history output.

Hive or Impala examples

```
SELECT * FROM t FOR SYSTEM_TIME AS OF '2021-08-09 10:35:57' LIMIT 100;

SELECT * FROM t FOR SYSTEM_VERSION AS OF 3088747670581784990 limit 100;
SELECT * from ice_11 FOR SYSTEM_TIME AS OF now() - interval 30 minutes;
```

Truncate table feature

Truncating an Iceberg table removes all rows from the table. A new snapshot is created. Truncation works for partitioned and unpartitioned tables.

Although the table data and the table and column stats are cleared, the old snapshots and their data files continue to exist to support time travel in the future.

Hive syntax

```
TRUNCATE table_name
```

Impala syntax

```
TRUNCATE [TABLE] table_name
```

Hive or Impala example

```
TRUNCATE t;
```

Update data feature

From Hive or Impala, you can update data from a V2 Iceberg table.

The Iceberg V2 format allows row-level modifictions using delete files. Row-level updates are supported. Hive and Impala write position to denote deleted or updated records. For information about position delete files, see the Delete data feature and Row-level operations topics.

Updating table data generates a new snapshot for the table.

You can use a WHERE clause in your UPDATE statement. For example:

```
update tbl_ice set b='Changed' where b in (select b from tbl_ice where a < 4
);
```

Hive or Impala evaluates rows from the target table against a WHERE clause, and updates all the rows that match the WHERE condition. The WHERE condition is similar to the WHERE condition used in SELECT.

Impala prerequisites

The target table should be an Iceberg V2 table. Therefore set the following table property during table creation, or use ALTER TABLE SET TBLPROPERTIES on existing Iceberg V1 tables: 'format-version' = '2'

You must write data and delete files in Parquet format. Set the following table properties to meet this prerequisite:

- 'write.format.default' = 'PARQUET'
- 'write.delete.format.default' = 'PARQUET'

Hive or Impala syntax

```
update tablename set column = value [, column = value ...] [where expres
sion]
```

Hive examples

```
create external table tbl_ice(a int, b string, c int) stored by iceberg tblp
roperties ('format-version'='2');
insert into tbl_ice values (1, 'one', 50), (2, 'two', 51), (3, 'three', 52),
 (4, 'four', 53), (5, 'five', 54), (111, 'one', 55), (333, 'two', 56);

update tbl_ice set b='Changed' where b in (select b from tbl_ice where a <
 4);
```

Updating a table from a non-Iceberg source

From Impala, you can use UPDATE FROM to update anIceberg table based on a source table or view that is another Iceberg or non-Iceberg table. For example:

```
UPDATE tbl_ice SET tbl_ice.one = t.one, tbl_ice.two = t.two, FROM tbl_ice, t
bl_hive one where tbl_ice.pk = one.pk;
```

Best practices for Iceberg in Cloudera

Based on large scale TPC-DS benchmark testing, performance testing and real-world experiences, Cloudera recommends several best practices when using Iceberg.

Follow these key best practices listed below when using Iceberg:

- Use Iceberg as intended for analytics.

  The table format is designed to manage a large, slow-changing collection of files. For more information, see the Iceberg spec.
- Reduce read amplification

  Monitor the growth of positional delta files, and perform timely compactions.
- Speed up drop table performance, preventing deletion of data files by using the following table properties:

  ```
  Set external.table.purge=false and gc.enabled=false
  ```
- Tune the following table properties to improve concurrency on writes and reduce commit failures: commit.retry .num-retries (default is 4), commit.retry.min-wait-ms (default is 100)
- Maintain a relatively small number of data files under the iceberg table/partition directory for efficient reads. To alleviate poor performance caused by too many small files, run the following queries:

  ```
  TRUNCATE TABLE target;
                 INSERT OVERWRITE TABLE target select * from target FOR SYST
  EM_VERSION AS OF <preTruncateSnapshotId>;
  ```
- To minimize the number of delete files and file handles and improve performance, ensure that the Spark write.distribution.mode table property value is “hash” (the default setting for Spark Iceberg 1.2.0 onwards).

Migrating to V2 tables before row-level changes

Learn the types of workloads best suited for Iceberg and when using V2 tables might improve query response.

Iceberg atomic DELETE and UPDATE operations resemble traditional RDBMS systems, but are not suitable for OLTP workloads. Iceberg is not designed to handle high frequency transactions. To handle very large datasets and frequent updates, use Apache Kudu.

Use Iceberg for managing large, infrequently changing datasets. You can update and delete Iceberg V2 tables at the row-level and not incur the overhead of rewriting the data files of V1 tables. Iceberg stores information about the deleted records in position delete files. These files store the file paths and positions of the deleted records, eliminating the need to rewrite the files. Iceberg performs a DELETE plus an INSERT operation in a single transaction. This technique speeds up queries. Query engines scan the data files and delete files associated with a snapshot and merge them, removing the deleted rows. For example, to remove all data belonging to a single customer:

DELETE FROM ice_tbl WHERE user_id = 1234;

To update a column value in a specific record:

UPDATE ice_tbl SET col_v = col_v + 1 WHERE id = 4321;

Performance tuning

Impala uses its own C++ implementation to deal with Iceberg tables. This implementation provides significant performance advantages over other engines.

To tune performance, try the following actions:

- Speed up drop table performance, preventing deletion of data files by using the following table properties:

  ```
  Set external.table.purge=false and gc.enabled=false
  ```

- Tune the following table properties to improve concurrency on writes and reduce commit failures: commit.retry .num-retries (default is 4), commit.retry.min-wait-ms (default is 100)
- Read Iceberg V2 tables from Hive using vectorization when heavy table scanning occurs as in SELECT COUNT(*) FROM TBL_ICEBERG_PART.
  - set hive.llap.io.memory.mode=cache;
  - set hive.llap.io.enabled=true;
  - set hive.vectorized.execution.enabled=true;
- Use Iceberg from Impala for querying Iceberg tables when latency is a concern.

  The massively parallel SQL query engine, backend executors written in C++, and frontend (analyzer, planner) written in Java delivers query results fast.
- Cache manifest files as described in the next topic.

Caching manifest files

Apache Iceberg provides a mechanism to cache the contents of Iceberg manifest files in memory. The manifest caching feature helps to reduce repeated reads of small Iceberg manifest files from remote storage by Impala Coordinators and Catalogd.

Impala caches table metadata in CatalogD and the local Coordinator's catalog, making table metadata analysis fast if the targeted table metadata and files were previously accessed. Impala might analyze the same table multiple times across concurrent query planning and also within single query planning, so caching is very important.

Having a frontend written in Java, Impala can directly analyze many aspects of the Iceberg table metadata through the Java Library provided by Apache Iceberg. Metadata analysis such as listing the table data file, selecting the table snapshot, partition filtering, and predicate filtering is delegated through Iceberg Java API.

To use the Iceberg Java API while still maintaining fast query planning, Iceberg implements caching strategies in the Iceberg Java Library similar to those used by Apache Impala. The Iceberg manifest caching feature constitutes these caching strategies. For more information about manifest caching, see the Iceberg Manifest Caching Blog.

You can configure manifest caching in Cloudera Data Warehouse.

Unsupported features and limitations

Cloudera does not support all features in Apache Iceberg. The list of unsupported features for Cloudera differs from release to release. Also, Apache Iceberg in Cloudera has some limitations you need to understand.

Unsupported features

The following table presents feature limitations or unsupported features:

means not yet tested

N/A means will never be tested, not a GA candidate

Iceberg Feature	Hive	Impala	Spark
Branching/Tagging	#	#	#
Read equality deletes for Flink upserts	#	#	#
Read equality deletes for NiFi	#	#	#
Write equality deletes	N/A	N/A	N/A
Read outside files	N/A	N/A	N/A
Bucketing	#	#	#

The table above shows that the following features are not supported in this release of Cloudera:

- Tagging and branching

  A technical preview is supported from Hive (not Impala or Spark) in Cloudera Data Warehouse Public Cloud.
- Writing equality deletes

  Hive and Impala only read equality deletes.
- Reading files outside the table directory

  Reading outside files is not supported due to the security risk. An unauthorized party who knows the underlying schema and file location outside the table location can rewrite the manifest files within one table location to point to the data files in another table location to read your data.
- Buckets defined from Hive do not create like buckets in Iceberg.

  For more information, see "Bucketing workaround" below.
- Using Iceberg tables as Spark Structured Streaming sources or sinks
- PyIceberg
- Migration of Delta Lake tables to Iceberg

Limitations

The following features have limitations or are not supported in this release:

- Multiple insert overwrite queries that read data from a source table.

- When the underlying table is changed, you need to rebuild the materialized view manually, or use the Hive query scheduling to rebuild the materialized view.
- You must be aware of the following considerations when using equality deletes:
  - Equality updates and deletes are not supported.
  - If you are using Apache Flink or Apache NiFi to write equality deletes, then ensure that you provide a PRIMARY KEY for the table. This is required for engines to know which columns to write into the equality delete files.
  - If the table is partitioned then the partition columns have to be part of the PRIMARY KEY
  - For Apache Flink, the table should be in 'upsert-mode' to write equality deletes
  - Partition evolution is not allowed for Iceberg tables that have PRIMARY KEYs
  - An equality delete file in the table is the likely cause of a problem with updates or deletes in the following situations:
    - In Change Data Capture (CDC) applications
    - In upserts from Apache Flink
    - From a third-party engine
- You must be aware of the following:
  - An Iceberg table that points to another Iceberg table in the HiveCatalog is not supported.

    For example:

    ```
    CREATE EXTERNAL TABLE ice_t
    STORED BY ICEBERG
    TBLPROPERTIES ('iceberg.table_identifier'='db.tb');
    ```
  - See also Iceberg data types.

Bucketing workaround

A query from Hive to define buckets/folders in Iceberg do not create the same number of buckets/folders as the same query creates in Hive. In Hive bucketing by multiple columns using the following clause creates 64 buckets maximum inside each partition.

```
| CLUSTERED BY (                                          |
|   id,                                                   |
|   partition_id)                                         |
| INTO 64 BUCKETS
```

Defining bucketing from Hive on multiple columns of an Iceberg table using this query creates 64*64 buckets/folders; consequently, bucketing by group does not occur as expected. The operation will create many small files at scale, a drag on performance.

Add multiple bucket transforms (partitions) to more than one column in the current version of Iceberg as follows:

```
bucket(p, col1, col2) =[ bucket(m, col1) , bucket(n, col2) ] where p = m * n
```

Prerequisites

You need to set up your environment and meet Data Lake prerequisites for querying Iceberg tables in Cloudera. You learn which query editors are supported and which roles are required.

Note: Do not drop, move, or change the old table during a migration operation. Doing so will delete the data files of the old and new tables.

The following list covers prerequisites for using Iceberg.

- You are using the Cloudera Data Warehouse.
- You must have access to an activated environment.
-
- One of the following query editors are required to query Iceberg tables:
  - Hue (recommended)
  - A JDBC client
  - The Impala shell for remote users
- You must have the role: DWUser.
- You must obtain permission to run SQL queries from the Env Admin, who must add you to the Hadoop SQL Storage Handler and the Hadoop SQL policies.
- You must use the HadoopFileIO. S3FileIO is not supported. URLs in metadata starting with s3://... cause query failure.

Migrating from Impala Prerequisites

- The original table is an EXTERNAL table from the Impala perspective: The EXTERNAL table property value is true.
- The original table is a non-ACID table.
- You have “ALL” privileges on the database containing the table.

Related Information

Activating AWS environments

Activating Azure environments

Grant permission to run SQL queries

Test driving Iceberg from Hive

Accessing Iceberg tables on S3

Cloudera uses Apache Ranger to provide centralized security administration and management. The Ranger Admin UI is the central interface for security administration. You can use Ranger to create two policies that allow users to query Iceberg tables.

How you open the Ranger Admin UI differs from one Cloudera service to another. In Cloudera Management Console, you can select your environment, and then click Environment Details Quick Links Ranger .

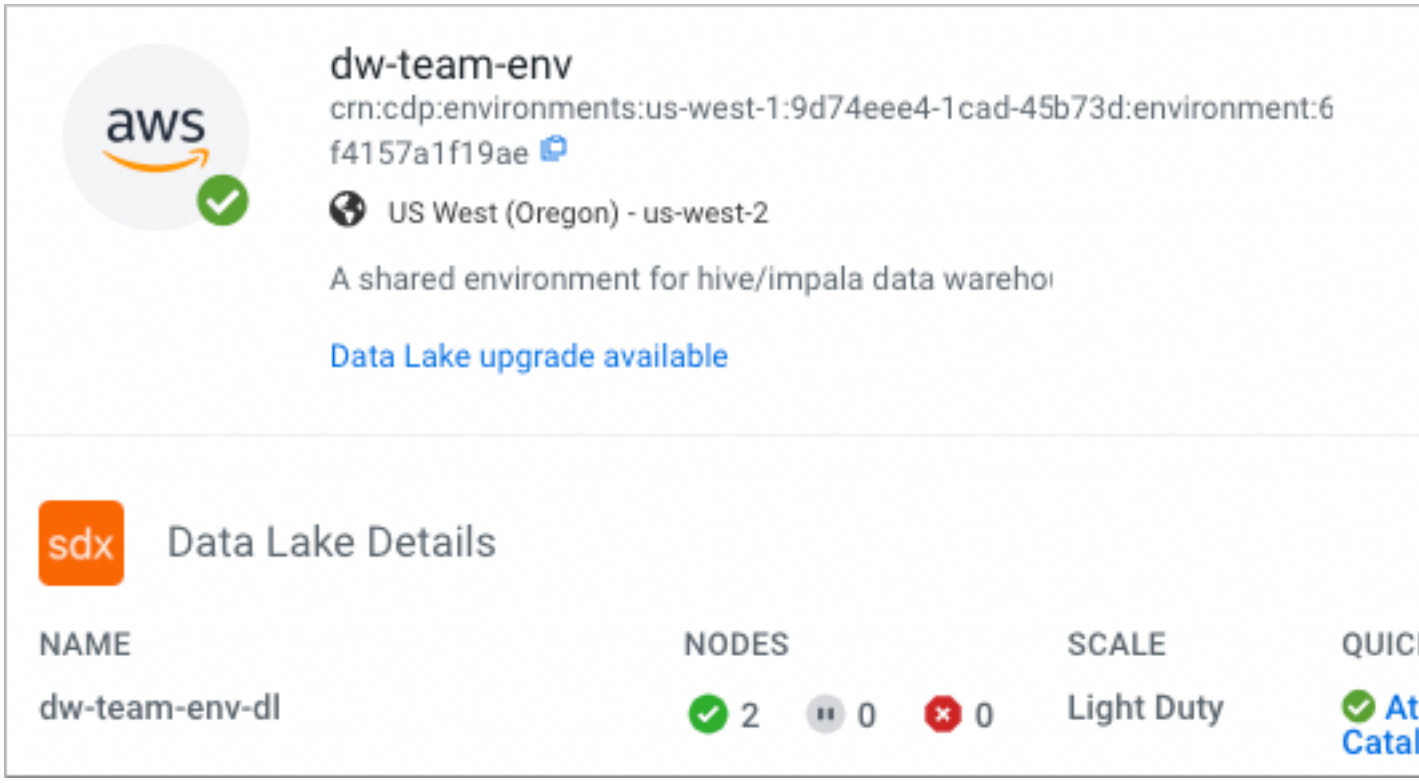

You log into the Ranger Admin UI, and the Ranger Service Manager appears.

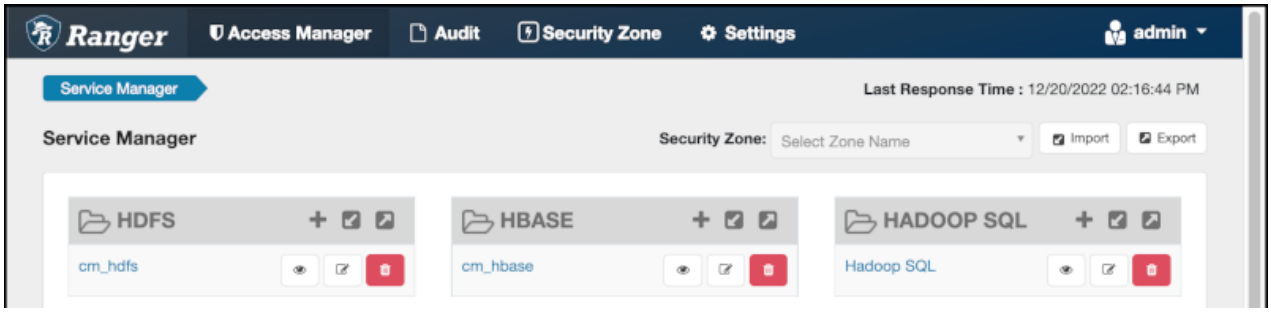

Policies for accessing tables on S3

The default policies that appear differ from service to service. You need to set up two Hadoop SQL policies to query Iceberg tables:

- One to authorize users to access the Iceberg files

  Follow steps in "Editing a policy to access Iceberg files" below.
- One to authorize users to query Iceberg tables

  Follow steps in "Creating a policy to query an Iceberg table" below.

Prerequisites

- Obtain the RangerAdmin role.
- Get the user name and password your Administrator set up for logging into the Ranger Admin.

  The default credentials for logging into the Ranger Admin Web UI are admin/admin123.

Opening Ranger in Cloudera Data Warehouse

You need to navigate to Ranger Admin UI to create a policy for users to access Iceberg tables. How you navigate to the Ranger Admin UI differs from one Cloudera service to another, and typically there is more than one path. You learn one way to navigate to the Ranger Admin UI fromCloudera Data Warehouse.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In the **Overview** page, click Database Catalogs and then click against the required database catalog.
3. Click Open Ranger to launch the Ranger Admin UI and manage access control for users.

Editing a storage handler policy to access Iceberg files on the file system

You learn how to edit the existing default Hadoop SQL Storage Handler policy to access files. This policy is one of the two Ranger policies required to use Iceberg.

About this task

The Hadoop SQL Storage Handler policy allows references to Iceberg table storage location, which is required for creating or altering a table. You use a storage handler when you create a file stored as Iceberg on the file system or object store.

In this task, you specify Iceberg as the storage-type and allow the broadest access by setting the URL to *.

The Hadoop SQL Storage Handler policy supports only the RW Storage permission. A user having the required RW Storage permission on a resource, such as Iceberg, that you specify in the storage-type properties, is allowed only to reference the table location (for create/alter operations) in Iceberg. The RW Storage permission does not provide access to any table data. You need to create the Hadoop SQL policy described in the next topic in addition to this Hadoop SQL Storage Handler policy to access data in tables.

For more information about these policy settings, see Ranger Storage Handler documentation.

Procedure

1. Log into Ranger Admin Web UI.
   The Ranger Service Manager appears:

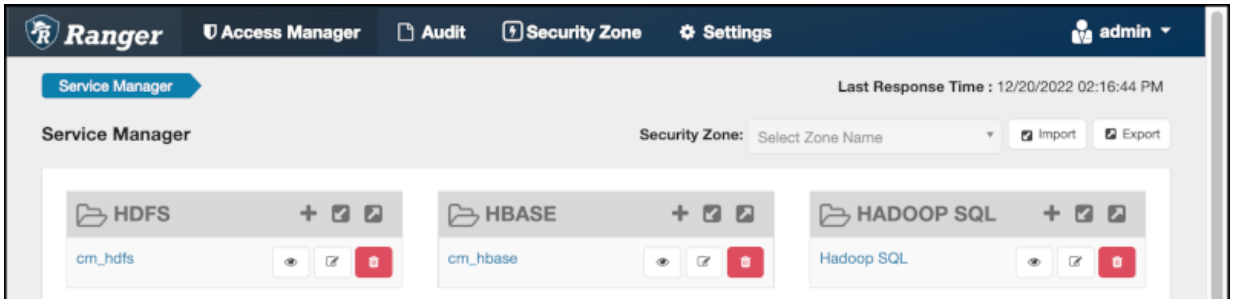

2. In Policy Name, enable the all - storage-type, storage-url policy.

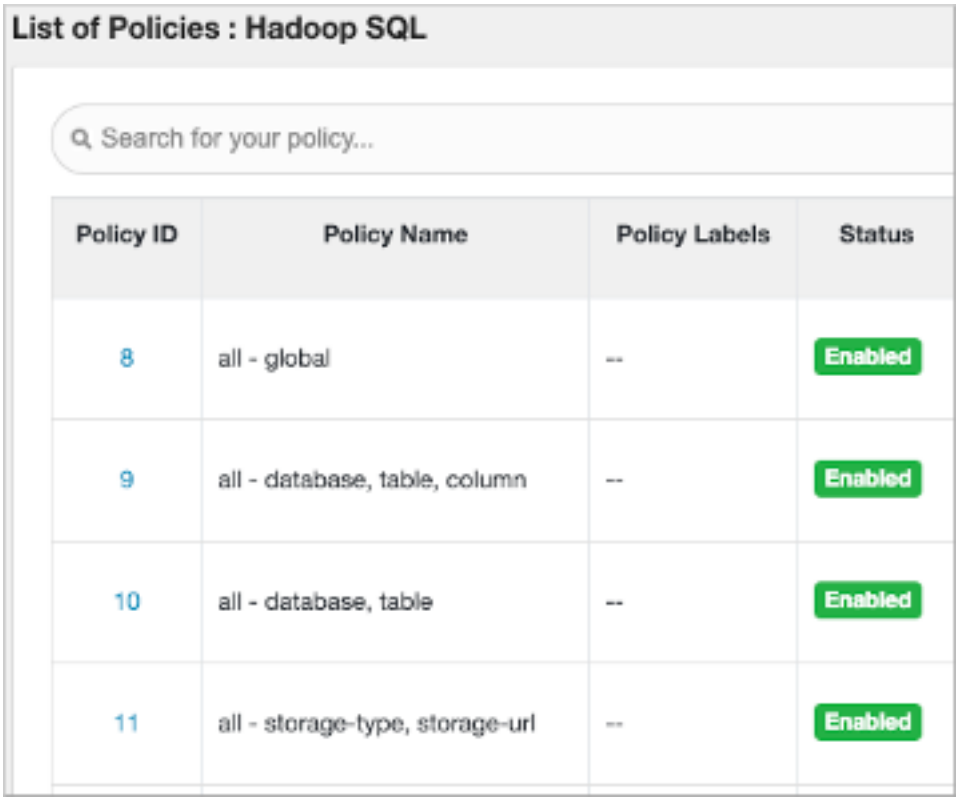

3. In **Service Manager**, in Hadoop SQL, select Edit and edit the all storage-type, storage-url policy.
4. Below Policy Label, select storage-type, and enter iceberg..
5. In Storage URL, enter the value *, enable Include.

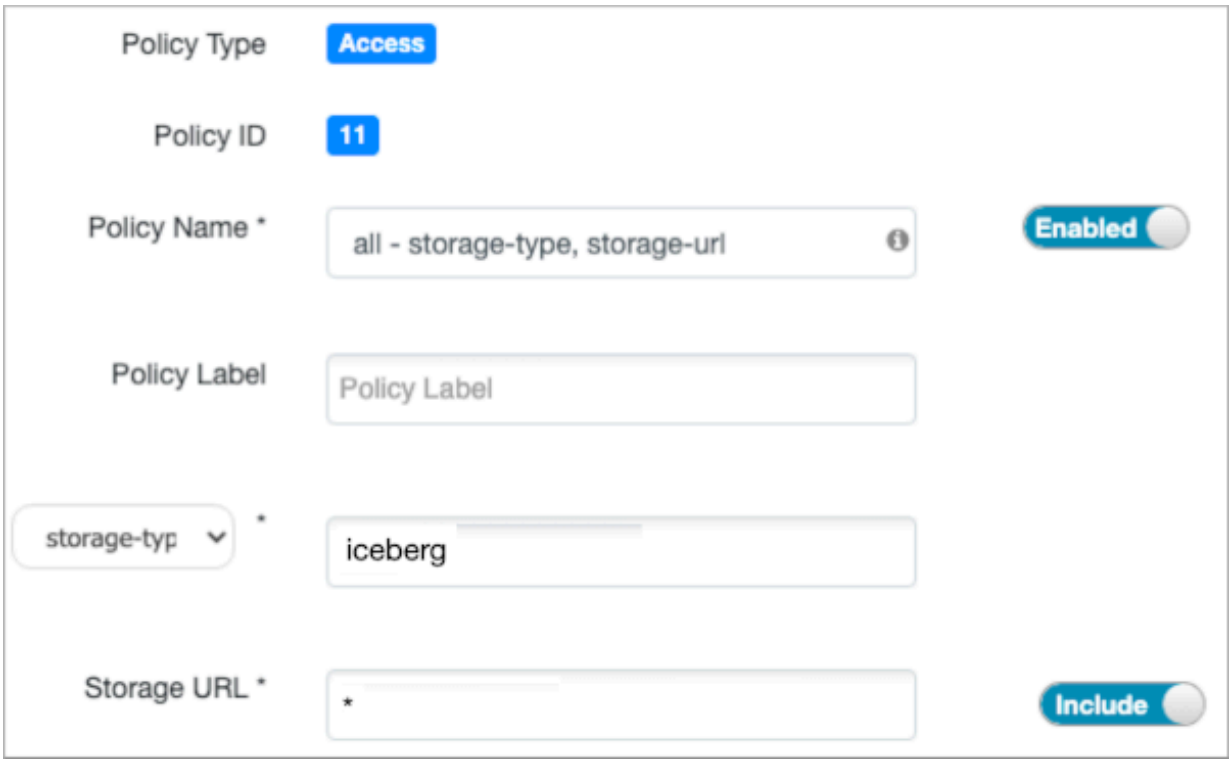

For more information about these policy settings, see Ranger storage handler documentation.

6. In Allow Conditions, specify roles, users, or groups to whom you want to grant RW storage permissions.

   You can specify PUBLIC to grant access to Iceberg tables permissions to all users. Alternatively, you can grant access to one user. For example, add the systest user to the list of users who can access Iceberg:

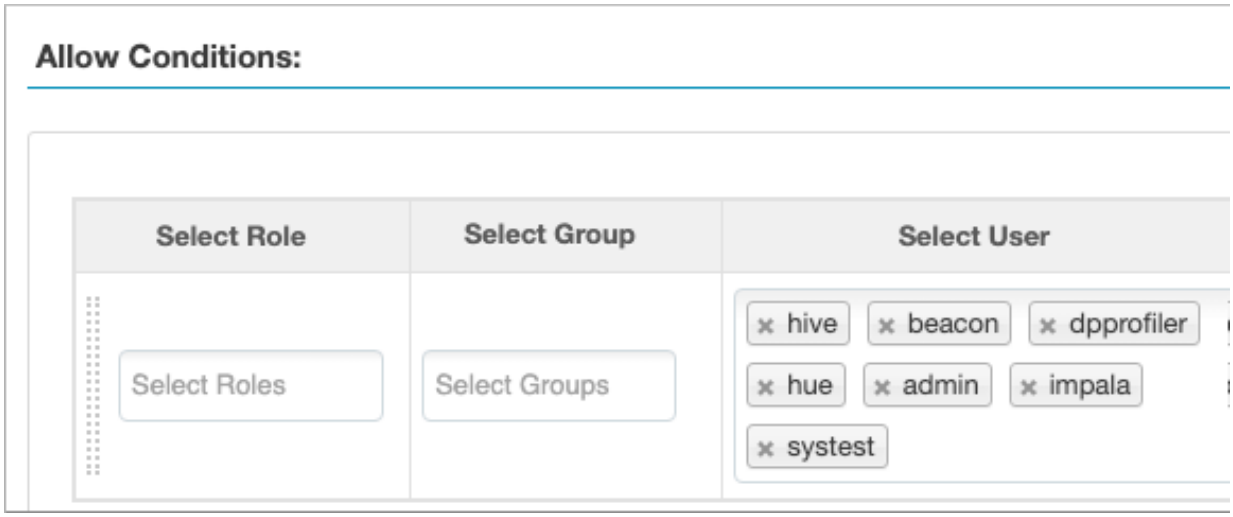

   For more information about granting permissions, see Configure a resource-based policy: Hadoop-SQL.

7. Add the RW Storage permission to the policy.
8. Save your changes.

Creating a policy to query an Iceberg table

You learn how to set up the second required policy for using Iceberg. This policy manages SQL query access to Iceberg tables.

About this task

In this task, you create a Hadoop SQL policy to allow a user to query an Iceberg table in a database. You set the conditions that allow (or deny) the selected user the following query permissions on the table: Select, Update, Create, Drop, Alter, and All. For more information about creating a policy, see Ranger documentation.

Procedure

1. Log into Ranger Admin Web UI.
   The Ranger Service Manager appears:
2. On the Service Manager page, select an existing HadoopSQL service.

   The List of Policies page appears.

   Note: The Service_name is cm_hive. The display name is HadoopSQL.

3. Click Add New Policy.

4. Fill in required fields.
   For example, enter the following required settings:
   - In Policy Name, set the name of the policy, for example IcebergPolicy1.
   - In database, set the name of the database controlled by this policy, for example icedb.
   - In table, set the name of the table controlled by this policy, for example icetable.
   - In columns, set the name of the column controlled by this policy, for example enter the wildcard asterisk (*) to allow access to all columns of icetable.
   - Accept defaults for other settings.

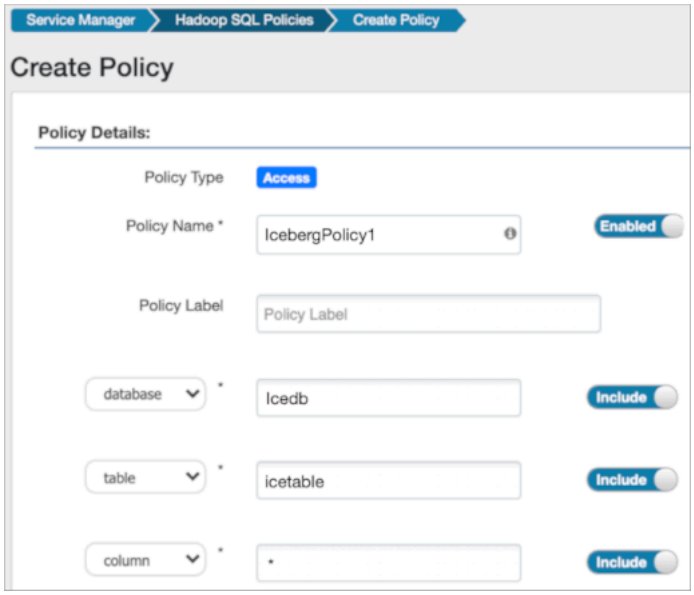

5. Scroll down to Allow Conditions, and select the user you want to access the table.
   You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.
6. Select permissions to give the user.
   For example, select Create, Select, and Alter. Alternatively, to provide the broadest permissions, select All.

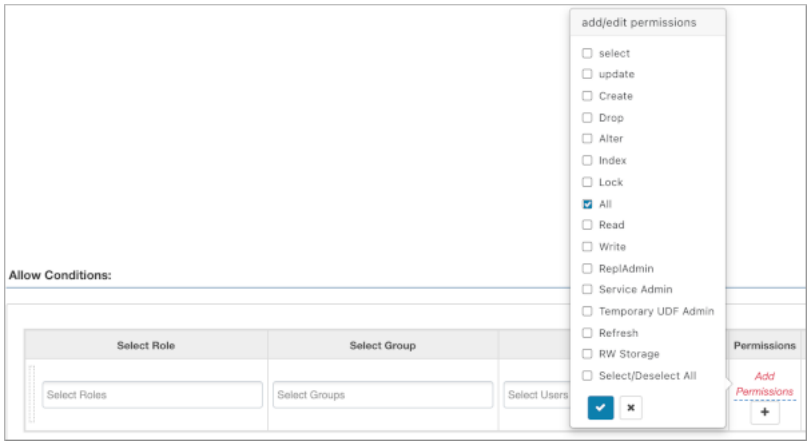

   Ignore RW Storage and other permissions not named after SQL queries. These are for future implementations.
7. Click Add.

Creating an Iceberg table

A step-by-step procedure describes how to create an Apache Iceberg table from a Hive or Impala Virtual Warehouse. You see how to access and use the recommended query editor Hue to create an Iceberg table.

About this task

In this task, you create an Iceberg table in Cloudera Data Warehouse. In a Virtual Warehouse, you open Hue, and use Hive or Impala to create a table.

Before you begin

- You must meet the prerequisites to query Iceberg tables from a Virtual Warehouse mentioned earlier, including obtaining Ranger access permissions.

Procedure

1. Create a new Virtual Warehouse, or select an existing one.
2. In Cloudera Data Warehouse Overview, select a Virtual Warehouse, and click Hue.

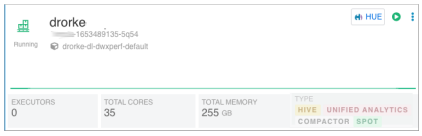

3. Select a database.
4. Enter a query to create a simple Iceberg table in the default Parquet format.
   Hive example:

```
CREATE EXTERNAL TABLE ice_t1 (i int, s string, ts timestamp, d date)
STORED BY ICEBERG;
```

   Impala example:

```
CREATE TABLE ice_t2 (i int, s string, ts timestamp, d date)
STORED BY ICEBERG;
```

   In CDP, CREATE EXTERNAL TABLE, and just CREATE TABLE, are valid from Hive. You use the EXTERNAL keyword from Hive to create the Iceberg table to purge the data when you drop the table. In CDP, from Impala, you must use CREATE TABLE to initialize the Iceberg table.
5. Click ▶ to run the query.

Related Information

Adding a new Virtual Warehouse
Submitting queries with Hue

Creating an Iceberg partitioned table

The ease of use of the Iceberg partitioning is clear from an example of how to partition a table using the backward compatible, identity-partition syntax. Alternatively, you can partition an Iceberg table by column values from Hive or Impala.

About this task

You can specify partitioning that is backward compatible with Iceberg V1 using the PARTITION BY clause. This type of table is called an identity-partitioned table. For more information about partitioning, see the Apache Iceberg documentation.

Before you begin

You must meet the prerequisites to query Iceberg tables from a Virtual Warehouse mentioned earlier.

Procedure

1. In Hue, select a database.
2. Create an identity-partitioned table.
   Hive:

   ```
   CREATE EXTERNAL TABLE ice_ext1 (i int, s string, ts timestamp, d date) P
   ARTITIONED BY (state string)
   STORED BY ICEBERG
   STORED AS ORC;
   ```

   Impala:

   ```
   CREATE TABLE ice_ext2 (i int, s string, ts timestamp, d date) PARTITIONED
    BY (state string)
   STORED BY ICEBERG;
   ```

3. Click ▶ to run the query.
4. Create a table and specify an identity transform, such as bucket, truncate, or date, using the Iceberg V2 PARTITION BY SPEC clause.
   Hive:

   ```
   CREATE TABLE ice_t_transforms_1 (i int, s string, ts timestamp, d date)
   PARTITIONED BY SPEC (TRUNCATE(10, i), BUCKET(11, s), YEAR(ts))
   STORED by ICEBERG;
   ```

   Impala:

   ```
   CREATE TABLE ice_t_transforms (i int, s string, ts timestamp, d date)PAR
   TITIONED BY SPEC (TRUNCATE(10, i), BUCKET(11, s), YEAR(ts))STORED AS ICE
   BERG;
   ```

5. Click ▶ to run the query.

Related Information

Partition transform feature

Adding a new Virtual Warehouse

Submitting queries with Hue

Expiring snapshots

You can expire snapshots of an Iceberg table using an ALTER TABLE query. You should periodically expire snapshots to delete data files that are no longer needed, and reduce the size of table metadata.

About this task

Each write to an Iceberg table creates a new snapshot, or version, of a table. You can use snapshots for time-travel queries, or to roll back a table to a valid snapshot. Snapshots accumulate until they are expired by the expire_snapshots operation.

Procedure

1. Enter a query to expire snapshots older than the following timestamp: '2021-12-09 05:39:18.689000000'

```
ALTER TABLE test_table EXECUTE EXPIRE_SNAPSHOTS('2021-12-09 05:39:18.689
000000');
```

2. Enter a query to expire snapshots having between December 10, 2022 and November 8, 2023.

```
ALTER TABLE test_table EXECUTE EXPIRE_SNAPSHOTS BETWEEN ('2022-12-10 00:
00:00.000000000') AND ('2023-11-08 00:00:00.000000000');
```

   You can also expire snapshots using a single snapshot ID or a list of IDs. For more information, see the "Expiring Snapshots Feature" topic.

Inserting data into a table

You can append data to an Iceberg table by inserting values or by selecting the data from another table. You can update data, replacing the old data.

You use the INSERT command in one of the following ways to populate an Iceberg table from Hive:

- INSERT INTO t VALUES (1, ‘asf’, true);
- INSERT INTO t SELECT * FROM s;
- INSERT OVERWRITE t SELECT * FROM s;

Examples

```
INSERT INTO t VALUES (1, ‘asf’, true);
INSERT INTO t SELECT * FROM s;
INSERT OVERWRITE t SELECT * FROM s;
```

Table migration overview

You learn what happens under the covers during Hive table to Iceberg table migration.

Migration from Hive

When you migrate an external Hive table to Iceberg, Hive makes the following changes:

1. Converts the storage_handler, serde, inputformat and outputformat properties of the table in HMS to use the Iceberg specific classes.
2. Reads the footers of the existing data files and generates the necessary Iceberg metadata files based on the footers.
3. Commits all the data files to the Iceberg table in a single commit.

Migration from Impala

Impala’s table migration process consists of multiple steps under the hood:

1. Sets table properties on the Hive table, such as 'external.table.purge'=false

2. Renames the original table name to a temporary name in the format “<original_name>_tmp_<random_ID>”
3. Refreshes the renamed Hive table.
4. Creates an Iceberg table using the original table name and the same location as Hive table.
5. Using the metadata for the data files of the Hive table, populates the Iceberg table with the data files. Reads the footer of the data files and saves statistics to Iceberg metadata.
6. Sets the Iceberg table ‘external.table.purge’ property to true.
7. Drops the renamed Hive table. The ‘external.table.purge’ property set to false prevents dropping the data files just deletes the metadata of the Hive table.

If for any reason the table migration fails, Impala cleans up the Iceberg metadata files created during step 6. Perhaps the original Hive table had already been renamed when the table migration failed. In this case, you see an error message about how to rename the Hive table to the original name. For example:

```
ALTER TABLE <tbl_name>_tmp_<random_ID> RENAME TO <tbl_name>;
```

Migrating a Hive table to Iceberg

You see how to use a simple ALTER TABLE statement from Hive or Impala to migrate an external Hive table to an Iceberg table. You see how to configure table input and output by setting table properties.

About this task

Note: To prevent loss of new and old table data during migration of a table to Iceberg, do not drop, move, or change the old table during migration.

Before you begin

In Impala, you can configure the NUM_THREADS_FOR_TABLE_MIGRATION query option to tweak the performance of the table migration. It sets the maximum number of threads to be used for the migration process but could also be limited by the number of CPUs. If set to zero then the number of available CPUs on the coordinator node is used as the maximum number of threads. Parallelism occurs on the basis of data files within a partition, which means one partition is processed at a time with multiple threads processing the files inside the partition. In case there is only one file in each partition, sequential execution occurs.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, in the Overview page, locate your Virtual Warehouse, and click Hue.

   Instead of using Hue, you can connect over JDBC to the Virtual Warehouse, and run the query.
3. Enter a query to use a database.
   For example:

   ```
   USE mydb;
   ```
4. Enter a query to migrate an existing external Hive table to an Iceberg v2 table.
   Hive example:

   ```
   ALTER TABLE tbl
   SET TBLPROPERTIES ('storage_handler'='org.apache.iceberg.mr.hive.HiveIce
   bergStorageHandler',
    'format-version' = '2');
   ```

   Impala example, which requires two queries:

   ```
   ALTER TABLE table_name CONVERT TO ICEBERG;
   ```

```
ALTER TABLE table_name SET TBLPROPERTIES ('format-version'='2');
```

The first ALTER command converts the Hive table to an Iceberg V1 table.

5. Click ▶ to run the queries.
   An Iceberg V2 table is created, replacing the Hive table.

Selecting an Iceberg table

You see an example of how to read an Apache Iceberg table, and understand the advantages of Iceberg.

About this task

Working with timestamps in Iceberg, you do not need to know whether the table is actually partitioned by month, day or hour, based on the timestamp value. You can simply supply a predicate for the timestamp value and Iceberg converts the timestamp to month/day/hour transparently. Hive/Impala must maintain actual partition values in a separate column (for example, ts_month or ts_day). Forgetting to reference the derived partition column in your query can lead to inadvertent full table scans.

By default iceberg.table_identifier is not set in Cloudera, so you can use the familiar <db_name.<table_name> in queries.

Procedure

1. Use a database.
   For example:

```
USE mydatabase;
```

2. Query an Iceberg table partitioned by city.
   For example:

```
SELECT * FROM ice_t2 WHERE city="Bangalore";
```

Running time travel queries

You query historical snapshots of data using the FOR SYSTEM_TIME AS OF '<timestamp>' FOR SYSTEM_VERSION AS OF <snapshot_id> clauses in a select statement. You see how to use AS OF to specify a snapshot of your Iceberg data at a certain time.

About this task

You can inspect the history of an Iceberg table to see the snapshots. You can query the metadata of the Iceberg table using a SELECT ... AS OF statement to run time travel queries. You use history information from a query of the database to identify and validate snapshots, and then query a specific snapshot AS OF a certain Timestamp value.

Before you begin

- You must be aware of the table history.

  However, this can include commits that have been rolled back.
- You must have access to valid snapshots.
- You must meet the prerequisites to query Iceberg tables mentioned earlier.

Procedure

1. View the table history.

```
SELECT * FROM db.table.history;
```

2. Check the valid snapshots of the table.

```
SELECT * FROM db.table.snapshots;
```

3. Query a specific snapshot by providing the timestamp and snapshot_id.

```
SELECT * FROM T
FOR SYSTEM_TIME AS OF <TIMESTAMP>;
SELECT * FROM t
FOR SYSTEM_VERSION AS OF <SNAPSHOT_ID>;
```

Updating an Iceberg partition

You see how to update Iceberg table partitioning in an existing table and then how to change the partitioning to be more granular.

About this task

Partition information is stored logically, and only in table metadata. When you update a partition spec, the old data written with an earlier spec remains unchanged. New data is written using the new spec in a new layout. Metadata for each of the partition versions is separate.

Before you begin

- You must meet the prerequisites to query Iceberg tables mentioned earlier.

Procedure

1. Create a table partitioned by year.
   Hive

```
CREATE EXTERNAL TABLE ice_t (i int, j int, ts timestamp)
PARTITIONED BY SPEC (truncate(5, j), year(ts))
STORED BY ICEBERG;
```

   Impala:

```
CREATE TABLE ice_t (i int, j int, ts timestmap)
PARTITIONED BY SPEC (truncate(5, j), year(ts))
STORED BY ICEBERG;
```

2. Split the data into manageable files using buckets.

```
ALTER TABLE ice_t SET PARTITION SPEC (bucket(13, i));
```

3. Partition the table by month.

```
ALTER TABLE ice_t SET PARTITION SPEC  (truncate(5, j), month(ts));
```

Test driving Iceberg from Impala

You complete a task that creates Iceberg tables from Impala with mock data that you can test drive using your own queries. You learn how to work with partitioned tables.

Before you begin

- You must meet the prerequisites to query Iceberg tables mentioned earlier, including obtaining Ranger access permissions.

Procedure

1. In Impala, use a database.
2. Create an Impala table to hold mock data for this task.

```
create external table mock_rows stored as parquet as
select x from (
with v as (values (1 as x), (1), (1), (1), (1))
select v.x from v, v v2, v v3, v v4, v v5, v v6
) a;
```

3. Create another Impala table based on mock_rows.

```
create external table customer_demo stored as parquet as
select
FROM_TIMESTAMP(DAYS_SUB(now() , cast ( TRUNC(RAND(7)*365*1) as bigint)), '
yyyy-MM') as year_month,
DAYS_SUB(now() , cast ( TRUNC(RAND(7)*365*1) as bigint)) as ts,
CONCAT(
  cast ( TRUNC(RAND(1) * 250 + 2) as string), '.' ,
  cast ( TRUNC(RAND(2) * 250 + 2) as string), '.',
  cast ( TRUNC(RAND(3) * 250 + 2) as string), '.',
  cast ( TRUNC(RAND(4) * 250 + 2) as string)
) as ip,
CONCAT("USER_", cast ( TRUNC(RAND(4) * 1000) as string),'@somedomain.com')
 as email,
CONCAT("USER_", cast ( TRUNC(RAND(5) * 1000) as string)) as username,
CONCAT("USER_", cast ( TRUNC(RAND(6) * 100) as string)) as country,
cast( RAND(8)*10000 as double) as metric_1,
cast( RAND(9)*10000 as double) as metric_2,
cast( RAND(10)*10000 as double) as metric_3,
cast( RAND(11)*10000 as double) as metric_4,
cast( RAND(12)*10000 as double) as metric_5
from mock_rows
;
```

4. Create another Impala table based on mock_rows.

```
create external table customer_demo2 stored as parquet as
select
FROM_TIMESTAMP(DAYS_SUB(now() , cast ( TRUNC(RAND(7)*365*1) as bigint)),
'yyyy-MM') as year_month,
DAYS_SUB(now() , cast ( TRUNC(RAND(7)*365*1) as bigint)) as ts,
CONCAT(
  cast ( TRUNC(RAND(1) * 250 + 2) as string), '.' ,
  cast ( TRUNC(RAND(2) * 250 + 2) as string), '.',
  cast ( TRUNC(RAND(3) * 250 + 2) as string), '.',
  cast ( TRUNC(RAND(4) * 250 + 2) as string)
) as ip,
```

```
CONCAT("USER_", cast ( TRUNC(RAND(4) * 1000) as string),'@somedomain.com')
 as email,
CONCAT("USER_", cast ( TRUNC(RAND(5) * 1000) as string)) as username,
CONCAT("USER_", cast ( TRUNC(RAND(6) * 100) as string)) as country,
cast( RAND(8)*10000 as double) as metric_1,
cast( RAND(9)*10000 as double) as metric_2,
cast( RAND(10)*10000 as double) as metric_3,
cast( RAND(11)*10000 as double) as metric_4,
cast( RAND(12)*10000 as double) as metric_5
from mock_rows
;
```

5. Create an Iceberg table from the customer_demo table.

```
CREATE TABLE customer_demo_iceberg STORED BY ICEBERG AS SELECT * FROM cu
stomer_demo;
```

6. Insert into the customer_demo_iceberg table the results of selecting all data from the customer_demo2 table.

```
INSERT INTO customer_demo_iceberg select * from customer_demo2;
INSERT INTO customer_demo_iceberg select * from customer_demo2;
INSERT INTO customer_demo_iceberg select * from customer_demo2;
```

7. Create an Iceberg table partitioned by the year_month column and based on the customer_demo_iceberg table.

```
CREATE TABLE customer_demo_iceberg_part PARTITIONED BY(year_month) STORED
 BY ICEBERG
AS SELECT ts, ip , email, username , country, metric_1 , metric_2 , metric
_3 , metric_4 , metric_5, year_month
FROM customer_demo_iceberg;
```

8. Split the partitioned data into manageable files.

```
ALTER TABLE customer_demo_iceberg_part SET PARTITION SPEC (year_month,BU
CKET(15, country));
```

9. Insert the results of reading the customer_demo_iceberg table into the partitioned table.

```
INSERT INTO customer_demo_iceberg_part (year_month, ts, ip, email, usern
ame, country, metric_1, metric_2, metric_3, metric_4, metric_5)
SELECT year_month, ts, ip, email, username, country, metric_1, metric_2,
 metric_3, metric_4, metric_5
FROM customer_demo_iceberg;
```

10. Run time travel queries on the Iceberg tables, using the history output to get the snapshot id, and substitute the id in the second SELECT query.

```
SELECT * FROM customer_demo_iceberg FOR SYSTEM_TIME AS OF '2021-12-09 05
:39:18.689000000' LIMIT 100;
DESCRIBE HISTORY customer_demo_iceberg;
SELECT * FROM customer_demo_iceberg FOR SYSTEM_VERSION AS OF <snapshot
id> LIMIT 100;
```

Hive demo data

To test drive Iceberg from Hive, you use demo data in the airline_online_iceberg database. If you enabled Load Demo Data when creating the Database Catalog, Iceberg users can work with the sample airline tables. Examples include creating a database, creating external tables using Hive, loading data into the newly created tables, and

converting the newly created Hive external tables to Iceberg tables. To become familiar with the demo data, take a look at the queries that created demo data.

Iceberg Database creation and setup

The Airlines demo data for Iceberg is stored in the airline_online_iceberg database. The following queries created and set up this database.

```
create database if not exists airline_ontime_iceberg;
use airline_ontime_iceberg;
set hive.vectorized.execution.enabled=false;
set hive.stats.column.autogather=false;
```

Hive external table creation

The following Hive external tables were created in the airline_online_iceberg database:

- airports
- airlines
- planes
- flights

```
create external table if not exists airports (
    iata string,
    airport string,
    city string,
    state double,
    country string,
    lat double,
    lon double
)
stored as orc;

create external table if not exists airlines (
    code string,
    description string
)
stored as orc;
create external table if not exists planes (
    tailnum string,
    owner_type string,
    manufacturer string,
    issue_date string,
    model string,
    status string,
    aircraft_type string,
    engine_type string,
    year int
)
stored as orc;

create external table if not exists flights (
    month int,
    dayofmonth int,
    dayofweek int,
    deptime  int,
    crsdeptime int,
    arrtime int,
    crsarrtime int,
    uniquecarrier string,
    flightnum int,
```

```
    tailnum string,
    actualelapsedtime int,
    crselapsedtime int,
    airtime int,
    arrdelay int,
    depdelay int,
    origin string,
    dest string,
    distance int,
    taxiin int,
    taxiout int,
    cancelled int,
    cancellationcode string,
    diverted string,
    carrierdelay int,
    weatherdelay int,
    nasdelay int,
    securitydelay int,
    lateaircraftdelay int
)
partitioned by (year int)
stored as orc;
```

Load data into the newly created tables

```
load data inpath '${datapath}/airline_ontime_iceberg.db/airports' into table
 airports;

load data inpath '${datapath}/airline_ontime_iceberg.db/airlines' into table
 airlines;

load data inpath '${datapath}/airline_ontime_iceberg.db/planes' into table p
lanes;

load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=1995' i
nto table flights partition (year=1995);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=1996'
into table flights partition (year=1996);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=1997'
 into table flights partition (year=1997);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=1998'
 into table flights partition (year=1998);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=1999' i
nto table flights partition (year=1999);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2000'
into table flights partition (year=2000);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2001'
 into table flights partition (year=2001);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2002'
 into table flights partition (year=2002);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2003' i
nto table flights partition (year=2003);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2004'
into table flights partition (year=2004);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2005'
 into table flights partition (year=2005);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2006'
 into table flights partition (year=2006);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2007' i
nto table flights partition (year=2007);
```

```
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2008'
into table flights partition (year=2008);
```

Convert these existing Hive external tables to Iceberg tables

```
ALTER TABLE planes ADD CONSTRAINT planes_pk PRIMARY KEY (tailnum) DISABLE NO
VALIDATE;
ALTER TABLE flights ADD CONSTRAINT planes_fk FOREIGN KEY (tailnum) REFEREN
CES planes(tailnum) DISABLE NOVALIDATE RELY;

ALTER TABLE airlines ADD CONSTRAINT airlines_pk PRIMARY KEY (code) DISABLE
 NOVALIDATE;
ALTER TABLE flights ADD CONSTRAINT airlines_fk FOREIGN KEY (uniquecarrier)
REFERENCES airlines(code) DISABLE NOVALIDATE RELY;

ALTER TABLE airports ADD CONSTRAINT airports_pk PRIMARY KEY (iata) DISABLE N
OVALIDATE;
ALTER TABLE flights ADD CONSTRAINT airports_orig_fk FOREIGN KEY (origin)
REFERENCES airports(iata) DISABLE NOVALIDATE RELY;
ALTER TABLE flights ADD CONSTRAINT airports_dest_fk FOREIGN KEY (dest) RE
FERENCES airports(iata) DISABLE NOVALIDATE RELY;

ALTER TABLE airports SET TBLPROPERTIES ('storage_handler'='org.apache.iceb
erg.mr.hive.HiveIcebergStorageHandler');
ALTER TABLE airlines SET TBLPROPERTIES ('storage_handler'='org.apache.icebe
rg.mr.hive.HiveIcebergStorageHandler');
ALTER TABLE planes SET TBLPROPERTIES ('storage_handler'='org.apache.iceberg.
mr.hive.HiveIcebergStorageHandler');
ALTER TABLE flights SET TBLPROPERTIES ('storage_handler'='org.apache.iceber
g.mr.hive.HiveIcebergStorageHandler');
```

Test driving Iceberg from Hive

You learn how to access the Hive demo data, which you can use to get hands-on experience running Iceberg queries.

About this task

A DWAdmin can optionally load demo data in Hue when you create a new Database Catalog.

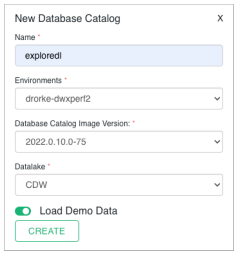

The Admin enables Load Demo Data when creating the Database Catalog. Users can then query sample airline demo data in Hue.

Before you begin

- You must meet the prerequisites to query Iceberg tables from a Virtual Warehouse mentioned earlier.
- You have access to a Hive Virtual Warehouse, having a Database Catalog in which demo data has been loaded.
- You obtained the required role for querying the Virtual Warehouse: DWUser

Procedure

1. In Cloudera Data Warehouse Overview, select a Hive Virtual Warehouse, and click Hue.

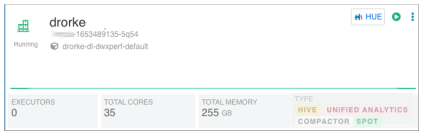

2. In Hue, expand the default database and verify that the airline demo data is available in your Virtual Warehouse. You see the following list of demo databases:

3. Select airline_ontime_iceberg to use the airline_ontime_iceberg database.
4. Take a look at the tables in the airline_ontime_iceberg database.

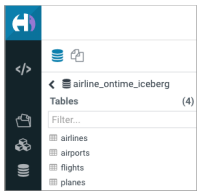

Flights is the fact table. It has 100M rows and three dirmensions, ariline, airports, and planes. This records flights for more than 10 years in the US, and includes the following details:

- origin
- destination
- delay
- air time

5. Become familiar with the Iceberg airline queries to set up this database. See the next topic.
6. Query the demo data from Hive.
For example, find the flights that departed each year, by IATA code, airport, city, state, and country. Find the average departure delay.

```
SELECT f.month, a.iata, a.airport, a.city, a.state, a.country
FROM flights f,
airports a
WHERE f.origin = a.iata
GROUP BY
f.month,
a.iata,
a.airport,
a.city,
a.state,
a.country
HAVING COUNT(*) > 10000
ORDER BY AVG(f.DepDelay) DESC
LIMIT 10;
```

Output appears as follows:

```
+----------+---------+-----------------------------------+---------------
+----------+------------+
| f.month  | a.iata  |             a.airport             |    a.city
 | a.state  | a.country  |
+----------+---------+-----------------------------------+---------------
+----------+------------+
| 12       | ORD     | Chicago O'Hare International      | Chicago
 | NULL     | USA        |
```

```
| 6         | EWR     | Newark Intl                          | Newark
 | NULL      | USA        |
| 7         | JFK     | John F Kennedy Intl                  | New York
 | NULL      | USA        |
| 6         | IAD     | Washington Dulles International      | Chantilly
 | NULL      | USA        |
| 7         | EWR     | Newark Intl                          | Newark
 | NULL      | USA        |
| 6         | PHL     | Philadelphia Intl                    | Philadelphia
 | NULL      | USA        |
| 1         | ORD     | Chicago O'Hare International         | Chicago
 | NULL      | USA        |
| 6         | ORD     | Chicago O'Hare International         | Chicago
 | NULL      | USA        |
| 7         | ATL     | William B Hartsfield-Atlanta Intl    | Atlanta
 | NULL      | USA        |
| 12        | MDW     | Chicago Midway                       | Chicago
 | NULL      | USA        |
+----------+---------+-------------------------------------+---------------
+----------+------------+
10 rows selected (103.812 seconds)
```

7. Split the partitioned data into manageable files.

```
ALTER TABLE airports SET PARTITION SPEC (iata,BUCKET(15, country));
```

Related Information

Prerequisites

Iceberg data types

References include Iceberg data types and a table of equivalent SQL data types by Hive/Impala SQL engine types.

Iceberg supported data types

Table 2:

Iceberg data type	SQL data type	Hive	Impala
binary		BINARY	BINARY
boolean	BOOLEAN	BOOLEAN	BOOLEAN
date	DATE	DATE	DATE
decimal(P, S)	DECIMAL(P, S)	DECIMAL (P, S)	DECIMAL (P, S)
double	DOUBLE	DOUBLE	DOUBLE
fixed(L)		BINARY	Not supported
float	FLOAT	FLOAT	FLOAT
int	TINYINT, SMALLINT, INT	INTEGER	INTEGER
list	ARRAY	ARRAY	Read only
long	BIGINT	BIGINT	BIGINT
map	MAP	MAP	Read only
string	VARCHAR, CHAR	STRING	STRING
struct	STRUCT	STRUCT	Read only

Iceberg data type	SQL data type	Hive	Impala
time		STRING	Not supported
timestamp	TIMESTAMP	TIMESTAMP	TIMESTAMP
timestamptz	TIMESTAMP WITH LOCAL TIME ZONE	Use TIMESTAMP WITH LOCAL TIMEZONE for handling these in queries	Read timestamptz into TIMESTAMP values Writing not supported
uuid	none	STRING Writing to Parquet is not supported	Not supported

Data type limitations

An implicit conversion to an Iceberg type occurs only if there is an exact match; otherwise, a cast is needed. For example, to insert a VARCHAR(N) column into an Iceberg table you need a cast to the VARCHAR type as Iceberg does not support the VARCHAR(N) type. To insert a SMALLINT or TINYINT into an Iceberg table, you need a cast to the INT type as Iceberg does not support these types.

Iceberg supports two timestamp types:

- timestamp (without timezone)
- timestamptz (with timezone)

In Spark 3.3 and earlier, Spark SQL supports a single TIMESTAMP type, which maps to the Iceberg timestamptz type. However, Impala is unable to write to Iceberg tables with timestamptz columns. To create Iceberg tables from Spark with timestamp rather than timestamptz columns, set the following configurations to true:

- spark.sql.iceberg.handle-timestamp-without-timezone
- spark.sql.iceberg.use-timestamp-without-timezone-in-new-tables

Configure these properties only on Spark 3.3 and earlier.

Spark still handles the timestamp column as a timestamp with local timezone. Inconsistent results occur unless Spark is running in UTC.

Unsupported data types

Impala does not support the following Iceberg data types:

- TIMESTAMPTZ (only read support)
- FIXED
- UUID

Iceberg table properties

The Cloudera environment for querying tables from Hive overrides some Iceberg table properties. You learn which table properties are supported for querying tables from Impala.

Iceberg documentation describes all the properties for configuring tables. This documentation focuses on key properties for working with Iceberg tables in Cloudera.

Iceberg supports concurrent writes by default. You can tune Iceberg v2 table properties for concurrent writes. You set the following properties if you plan to have concurrent writers on Iceberg v2 tables:

- commit.retry.min-wait-ms
- commit.retry.num-retries

Cloudera supports adding the Parquet compression type using table properties. For more information, see Iceberg documentation about Compression Types.

You can use the Alter Table feature to set a property. From Hive, the following Iceberg table property overrides are in effect:

- iceberg.mr.split.size overrides read.split.target-size.
- read.split.open-file-cost is overridden.

You can tune Iceberg v2 table properties for concurrent writes. From Impala, the following subset of Iceberg table properties are supported:

- history.expire.min-snapshots-to-keep

  Valid values: integers. Default = 1
- write.format.default

  Valid value: Parquet
- write.metadata.delete-after-commit.enabled

  Valid values: true or false.
- write.metadata.previous-versions-max

  Valid values: integers. Default = 100.
- write.parquet.compression-codec

  Valid values: GZIP, LZ4, NONE, SNAPPY (default value), ZSTD
- write.parquet.compression-level

  Valid values: 1 - 22. Default = 3
- write.parquet.row-group-size-bytes

  Valid values: 8388608 (or 8 MB) - 2146435072 (or 2047MB). Overiden by PARQUET_FILE_SIZE.
- write.parquet.page-size-bytes

  Valid values: 65536 (or 64KB) - 1073741824 (or 1GB).
- write.parquet.dict-size-bytes

  Valid values: 65536 (or 64KB) - 1073741824 (or 1GB)

1.0.0

Apache Iceberg Overview

Date published: 2022-03-15
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Iceberg overview..... 4

Iceberg overview

Apache Iceberg is a table format for huge analytics datasets that defines how metadata is stored and data files are organized. Iceberg is also a library that compute engines can use to read/write a table.

You can efficiently query large Iceberg tables on object stores. Iceberg supports concurrent reads and writes on all storage media. You can use Iceberg when a single table contains tens of petabytes of data, and you can read these tables without compromising performance. By default, Hive and Impala use the Iceberg HiveCatalog for Metastore management of Iceberg tables. The catalog stores Iceberg metadata, including the location of the table. Iceberg catalog use of Hive metastore (HMS) for catalog operations is lightweight. Iceberg relieves HMS pressure by storing partition information in metadata files on the file system/object store instead of within the HMS. This architecture supports rapid scaling without performance hits.

Cloudera supports Iceberg in Cloudera Data Warehouse in AWS and Azure environments.

Key features overview

From Hive and Impala, you can read and write V1 or V2 Iceberg tables. The following features are included:

- Reads of Iceberg V2 tables that have had row-level deletes or updates makes Apache Iceberg ACID-compliant with serializable isolation and an optimistic concurrency model
- Materialized views of Iceberg tables
- Enhanced maintenance features, such as expiring and removing old snapshots and compaction of small files
- Performance and scalability enhancements

Iceberg table security and visualization

Apache Iceberg integrates Apache Ranger for security. You can use Ranger integration with Hive and Impala to apply fine-grained access control to sensitive data in Iceberg tables. Iceberg is also integrated with Cloudera Data Visualization for creating dashboards and other graphics of your Iceberg data.

Supported ACID transaction properties

Iceberg supports atomic and isolated database transaction properties. Writers work in isolation, not affecting the live table, and perform an atomic swap for metadata pointers only when the transactions are committed, making the changes in one atomic commit.

Iceberg uses snapshots to guarantee isolated reads and writes. You see a consistent version of table data without locking the table. Readers always see a consistent version of the data without the need to lock the table. Writers work in isolation, not affecting the live table.

Iceberg partitioning

The Iceberg partitioning technique has performance advantages over conventional partitioning, such as Apache Hive partitioning. Iceberg hidden partitioning is easier to use. Iceberg supports in-place partition evolution; to change a partition, you do not rewrite the entire table to add a new partition column, and queries do not need to be rewritten for the updated table. Iceberg continuously gathers data statistics, which supports additional optimizations, such as partition pruning.

Iceberg uses multiple layers of metadata files to find and prune data. Impala, Spark, and Flink keep track of data at the folder level and not at the file level, performing file list operations when working with data in a table. Performance problems occur during the execution of multiple directory listings. Iceberg keeps track of a complete list of files within a table using a persistent tree structure. Changes to an Iceberg table use an atomic object/file level commit to update the path to a new snapshot file. The snapshot points to the individual data files through manifest files.

The manifest files track data files across partitions, storing partition information and column metrics for each data file. A manifest list is an additional index for pruning entire manifests. File pruning increases efficiency.

Iceberg relieves Hive metastore (HMS) pressure by storing partition information in metadata files on the file system/object store instead of within the HMS. This architecture supports rapid scaling without performance hits.

Related Information

Using Apache Iceberg

Iceberg support from Spark in Cloudera Data Engineering

1.0.0

Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-05

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Impala Overview **4**
Components of Impala **4**
Catalog Service High Availability **7**

Apache Impala Overview

The Apache Impala provides high-performance, low-latency SQL queries on data stored in popular Apache Hadoop file formats.

The Impala solution is composed of the following components.

Impala

The Impala service coordinates and executes queries received from clients. Queries are distributed among Impala nodes, and these nodes then act as workers, executing parallel query fragments.

Hive Metastore

Stores information about the data available to Impala. For example, the metastore lets Impala know what databases are available and what the structure of those databases is. As you create, drop, and alter schema objects, load data into tables, and so on through Impala SQL statements, the relevant metadata changes are automatically broadcast to all Impala nodes by the dedicated catalog service.

Clients

Entities including Hue, ODBC clients, JDBC clients, Business Intelligence applications, and the Impala Shell can all interact with Impala. These interfaces are typically used to issue queries or complete administrative tasks such as connecting to Impala.

Storage for data to be queried

Queries executed using Impala are handled as follows:

1. User applications send SQL queries to Impala through ODBC or JDBC, which provide standardized querying interfaces. The user application may connect to any impalad in the cluster. This impalad becomes the coordinator for the query.
2. Impala parses the query and analyzes it to determine what tasks need to be performed by impalad instances across the cluster. Execution is planned for optimal efficiency.
3. Storage services are accessed by local impalad instances to provide data.
4. Each impalad returns data to the coordinating impalad, which sends these results to the client.

Components of Impala

The Impala service is a distributed, massively parallel processing (MPP) database engine. It consists of different daemon processes called as components. This topic describes the different roles these components play for a selected Virtual Warehouse.

When you create an Impala Virtual Warehouse, it is automatically optimized for your workload by the Cloudera Data Warehouse (CDW) service. Due to the containerized and compute-isolated architecture of CDW, as well as intelligence to assign different default configurations, you do not have to customize your environment to optimize performance or to avoid resource usage spikes and out-of-memory conditions. However, if you must adjust some settings to suit your needs after creating an Impala Virtual Warehouse, you can add particular component configuration details in one of the free-form fields on the SIZING AND SCALING tab or under one of the components available under the CONFIGURATIONS tab for a selected Virtual Warehouse.

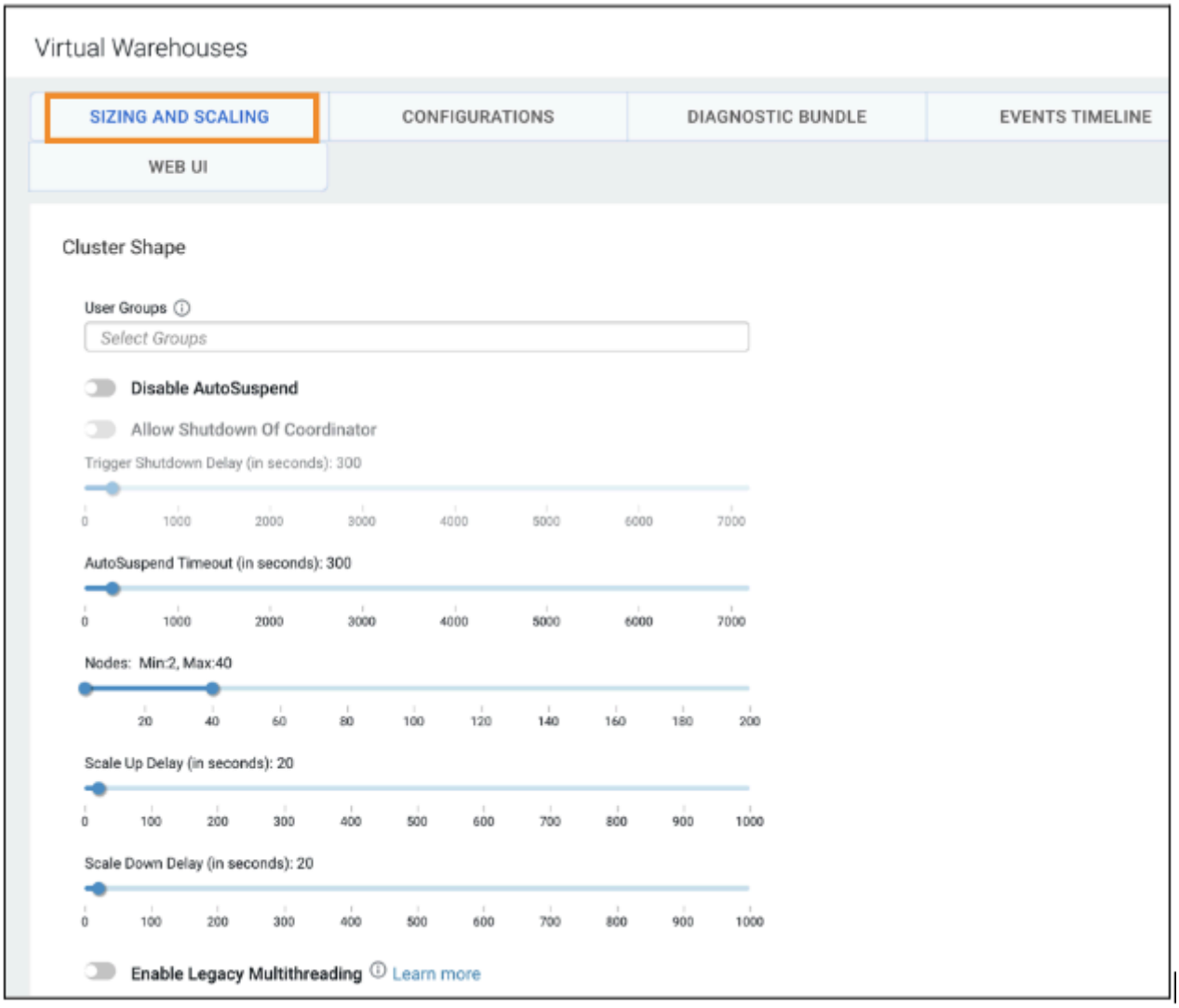

Before making any configuration changes, you can review the default configuration details in the components available under the CONFIGURATIONS tab in a Virtual Warehouse.

Impala service in CDW consists of the following different daemon processes.

- Impala catalogd
- Impala coordinator
- Impala executor
- Impala statestored
- Impala autoscaler
- Impala proxy

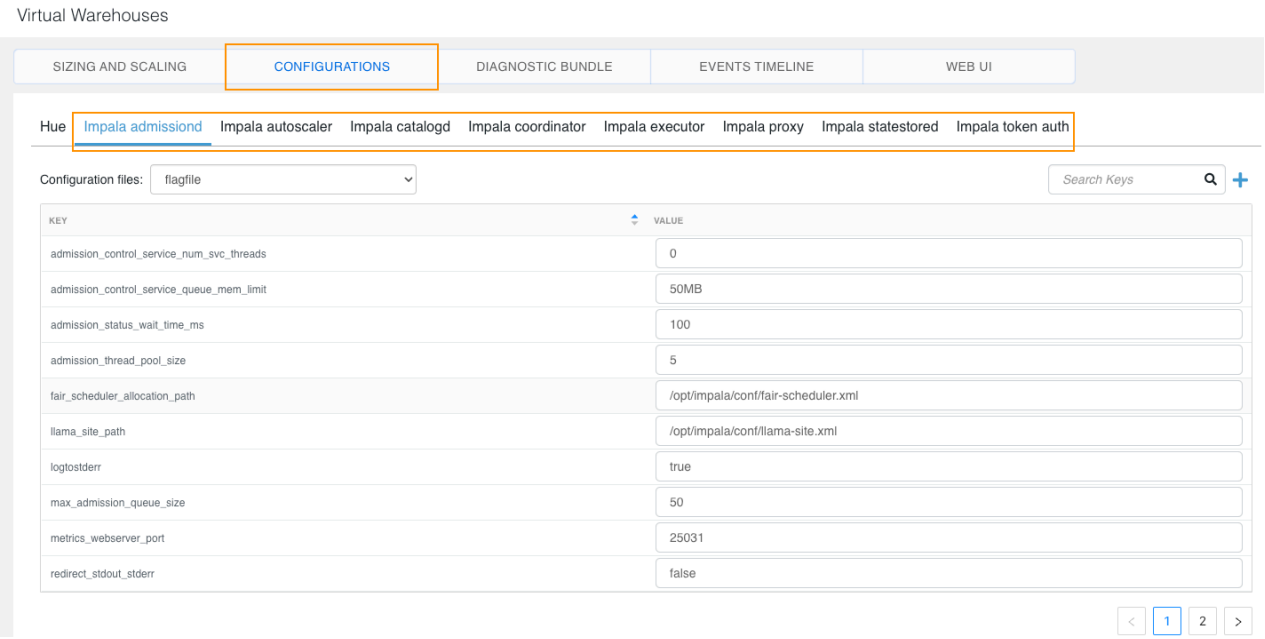

Components available for setting the configuration

This section provides reference information on supported configuration properties under the listed components. For the full list of configuration properties, see Impala Properties in Cloudera Runtime. Based on the information provided under the components, choose the appropriate component/role to tune a configuration if you must.

Note: Some of the configuration properties under these components can be tuned to your needs and some of the properties are not editable. In addition, the available properties may differ by CDW Impala Runtime version.

Impala catalogd

The Catalog Server relays the metadata changes from Impala SQL statements to all the Impala daemons in a cluster. The catalog service avoids the need to issue REFRESH and INVALIDATE METADATA statements when the metadata changes are performed by statements issued through Impala.

Impala coordinator

A few of the key functions that an Impala coordinator performs are:

- Reads and writes to data files.
- Accepts queries transmitted from the impala-shell command, Hue, JDBC, or ODBC.
- Parallelizes the queries and distributes work across the cluster.

It is in constant communication with StateStore, to confirm which executors are healthy and can accept new work. Based on the health information it receives it assigns tasks to the executors. It also receives broadcast messages from the Catalog Server daemon whenever a cluster creates, alters, or drops any type of object, or when an INSERT or LOAD DATA statement is processed through Impala. It also communicates to Catalog Server daemon to load table metadata.

Impala executor

A few of the key functions that an Impala executor performs are:

- Executes the queries and transmits query results back to the central coordinator.
- Also transmits intermediate query results.

Depending on the size and the complexity of your queries you can select the number of executor nodes that are needed to run a typical query in your workloads. For more information on the

executor groups and recommendations on sizing your virtual warehouse to handle queries that must be run in your workloads concurrently, see Impala auto-scaling on public clouds.

Impala statestored

The Impala StateStore checks on the health of all Impala daemons in a cluster, and continuously relays its findings to each of those daemons. If an Impala daemon goes offline due to hardware failure, network error, software issue, or other reason, the StateStore informs all the other Impala daemons so that future queries can avoid making requests to the unreachable Impala daemon.

Components available for diagnostic purpose only

The following components are read-only, and available only to view for diagnostic purposes. The properties listed under them are not tunable.

Impala autoscaler

One of the core Impala Virtual Warehouse components is Impala autoscaler. Impala autoscaler is in constant communication with coordinators and executors to determine when more or fewer compute resources are needed, given the incoming query load. When the autoscaler detects an imbalance in resources, it sends a request to the Kubernetes framework to increase or decrease the number of executor groups in the Virtual Warehouse, thereby right-sizing the amount of resources available for queries. This ensures that workload demand is met without wasting cloud resources.

Impala proxy

Impala Proxy is a small footprint reverse proxy that will forward every http client request to the Coordinator endpoint.

When you create an Impala warehouse with the 'allow coordinator shutdown' option a proxy 'impala proxy' is created to act as a connection endpoint to the impala clients. This option when enabled allows Impala coordinators to automatically shut down during idle periods. If the coordinator has shut down because of idle period, and if there is a request from a client when the coordinator is not running, impala proxy triggers the coordinator to start. So the proxy acts as a layer between the clients and the coordinator so that the clients connected to the VW do not time out when the coordinator starts up.

Catalog Service High Availability

By default, the Impala Virtual Warehouse runs with Catalog service in Active-Passive HA mode which runs two Impala catalog instances. If you enable HA mode for Impala Virtual Warehouse, the statestore allows one of the catalog instances to become active, and the other paired catalog instance to become a standby.

With any new query requests, the Impala coordinator sends metadata requests to the catalog service and sends metadata updates to the catalog which in turn propagates metadata updates to HMS. With a pair of primary/standby Catalog instances, the standby instance will be promoted as the primary instance to continue executing queries when the primary instance goes down. This High Availability (HA) mode of catalog service in CDW reduces the outage duration of the Impala cluster when the primary catalog service fails.

Catalog Failure Detection

The Statestore instance continuously sends a heartbeat to its registered clients, including the primary and standby Catalog instances, to determine if they are healthy. If the Statestore finds the primary Catalog instance is not healthy, but the standby Catalog instance is healthy, the Statestore promotes the standby Catalog instance as the primary instance and notifies all coordinators and catalogs of this change. Coordinators will switch over to the new primary Catalog instance.

1.0.0

Planning for Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-05

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Guidelines for Schema Design.....4

Guidelines for Schema Design

Use the guidelines in this topic to construct an optimized and scalable schema that integrates well with your existing data management processes.

Prefer binary file formats over text-based formats

To save space and improve memory usage and query performance, use binary file formats for any large or intensively queried tables. Parquet file format is the most efficient for data warehouse-style analytic queries. Avro is the other binary file format that Impala supports, that you might already have as part of a Hadoop ETL pipeline.

Although Impala can create and query tables with the RCFile and SequenceFile file formats, such tables are relatively bulky due to the text-based nature of those formats, and are not optimized for data warehouse-style queries due to their row-oriented layout. Impala does not support INSERT operations for tables with these file formats.

Guidelines:

- For an efficient and scalable format for large, performance-critical tables, use the Parquet file format.
- To deliver intermediate data during the ETL process, in a format that can also be used by other Hadoop components, Avro is a reasonable choice.
- For convenient import of raw data, use a text table instead of RCFile or SequenceFile, and convert to Parquet in a later stage of the ETL process.

Use Snappy compression where practical

Snappy compression involves low CPU overhead to decompress, while still providing substantial space savings. In cases where you have a choice of compression codecs, such as with the Parquet and Avro file formats, use Snappy compression unless you find a compelling reason to use a different codec.

Prefer numeric types over strings

If you have numeric values that you could treat as either strings or numbers (such as YEAR, MONTH, and DAY for partition key columns), define them as the smallest applicable integer types. For example, YEAR can be SMALLINT, MONTH and DAY can be TINYINT. Although you might not see any difference in the way partitioned tables or text files are laid out on disk, using numeric types will save space in binary formats such as Parquet, and in memory when doing queries, particularly resource-intensive queries such as joins.

Partition, but do not over-partition

Partitioning is an important aspect of performance tuning for Impala. Set up partitioning for your biggest, most intensively queried tables.

If you are moving to Impala from a traditional database system, or just getting started in the Big Data field, you might not have enough data volume to take advantage of Impala parallel queries with your existing partitioning scheme. For example, if you have only a few tens of megabytes of data per day, partitioning by YEAR, MONTH, and DAY columns might be too granular. Most of your cluster might be sitting idle during queries that target a single day, or each node might have very little work to do. Consider reducing the number of partition key columns so that each partition directory contains several gigabytes worth of data.

You can reduce the Parquet block size to as low as 128 MB or 64 MB to increase the number of files per partition and improve parallelism. But also consider reducing the level of partitioning so that analytic queries have enough data to work with.

Run COMPUTE STATS after loading data

Impala makes extensive use of statistics about data in the overall table and in each column, to help plan resource-intensive operations such as join queries and inserting into partitioned Parquet tables. Because this information is only

available after data is loaded, run the COMPUTE STATS statement on a table after loading or replacing data in a table or partition.

Having accurate statistics can make the difference between a successful operation, or one that fails due to an out-of-memory error or a timeout. When you encounter performance or capacity issues, always use the SHOW STATS statement to check if the statistics are present and up-to-date for all tables in the query.

When doing a join query, Impala consults the statistics for each joined table to determine their relative sizes and to estimate the number of rows produced in each join stage. When doing an INSERT into a Parquet table, Impala consults the statistics for the source table to determine how to distribute the work of constructing the data files for each partition.

Verify sensible execution plans with EXPLAIN and SUMMARY

Before executing a resource-intensive query, use the EXPLAIN statement to get an overview of how Impala intends to parallelize the query and distribute the work. If you see that the query plan is inefficient, you can take tuning steps such as changing file formats, using partitioned tables, running the COMPUTE STATS statement, or adding query hints.

After you run a query, you can see performance-related information about how it actually ran by issuing the SUMM ARY command in `impala-shell`. Prior to Impala 1.4, you would use the PROFILE command, but its highly technical output was only useful for the most experienced users. SUMMARY, new in Impala 1.4, summarizes the most useful information for all stages of execution, for all nodes rather than splitting out figures for each node.

1.0.0

Apache Impala Reference

Date published: 2020-11-30
Date modified: 2024-12-05

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Performance Considerations **4**
Performance Best Practices 4
Query Join Performance 6
Table and Column Statistics 7
Generating Table and Column Statistics 19
Runtime Filtering 22
Distribute Runtime Filter Aggregation 25
Skip Scheduling Bloom Filter 25
Min/Max Filtering 26
Bloom Filtering 27
Late Materialization of Columns 28
Partitioning 28
Partition Pruning for Queries 31
Understanding Performance using EXPLAIN Plan 34
Understanding Performance using SUMMARY Report 35
Understanding Performance using Query Profile 36
Planner changes for CPU usage 37
Planner changes to improve cardinality estimation 38
Caching Codegen Functions 42
Scalability Considerations **43**
Scaling Limits and Guidelines 47
Hadoop File Formats Support **48**
Using Text Data Files 49
Using Parquet Data Files 52
Using ORC Data Files 60
Using Avro Data Files 61
Using RCFile Data Files 64
Using SequenceFile Data Files 65
Storage Systems Supports **66**
Impala with Azure Data Lake Store (ADLS) 66
Impala with Amazon S3 69
Specifying Impala Credentials to Access S3 70
Configure Impala Daemon to spill to S3 70
Ports Used by Impala **71**
SQL transactions in Impala **71**

Performance Considerations

The following sections explain the factors affecting the performance of Impala features, and procedures for tuning, monitoring, and benchmarking Impala queries and other SQL operations.

Performance Best Practices

Use the performance guidelines and best practices during planning, experimentation, and performance tuning for an Impala-enabled cluster.

Choose the appropriate file format for the data

Typically, for large volumes of data (multiple gigabytes per table or partition), the Parquet file format performs best because of its combination of columnar storage layout, large I/O request size, and compression and encoding.

Note: For smaller volumes of data, a few gigabytes or less for each table or partition, you might not see significant performance differences between file formats. At small data volumes, reduced I/O from an efficient compressed file format can be counterbalanced by reduced opportunity for parallel execution. When planning for a production deployment or conducting benchmarks, always use realistic data volumes to get a true picture of performance and scalability.

Avoid data ingestion processes that produce many small files

When producing data files outside of Impala, prefer either text format or Avro, where you can build up the files row by row. Once the data is in Impala, you can convert it to the more efficient Parquet format and split into multiple data files using a single INSERT ... SELECT statement. Or, if you have the infrastructure to produce multi-megabyte Parquet files as part of your data preparation process, do that and skip the conversion step inside Impala.

Always use INSERT ... SELECT to copy significant volumes of data from table to table within Impala. Avoid INSERT ... VALUES for any substantial volume of data or performance-critical tables, because each such statement produces a separate tiny data file.

For example, if you have thousands of partitions in a Parquet table, each with less than 256 MB of data, consider partitioning in a less granular way, such as by year / month rather than year / month / day. If an inefficient data ingestion process produces thousands of data files in the same table or partition, consider compacting the data by performing an INSERT ... SELECT to copy all the data to a different table; the data will be reorganized into a smaller number of larger files by this process.

Choose partitioning granularity based on actual data volume

Partitioning is a technique that physically divides the data based on values of one or more columns, such as by year, month, day, region, city, section of a web site, and so on. When you issue queries that request a specific value or range of values for the partition key columns, Impala can avoid reading the irrelevant data, potentially yielding a huge savings in disk I/O.

When deciding which column(s) to use for partitioning, choose the right level of granularity. For example, should you partition by year, month, and day, or only by year and month? Choose a partitioning strategy that puts at least 256 MB of data in each partition, to take advantage of HDFS bulk I/O and Impala distributed queries.

Over-partitioning can also cause query planning to take longer than necessary, as Impala prunes the unnecessary partitions. Ideally, keep the number of partitions in the table under 30 thousand.

When preparing data files to go in a partition directory, create several large files rather than many small ones. If you receive data in the form of many small files and have no control over the input format, consider using the INSERT ... SELECT syntax to copy data from one table or partition to another, which compacts the files into a relatively small number (based on the number of nodes in the cluster).

If you need to reduce the overall number of partitions and increase the amount of data in each partition, first look for partition key columns that are rarely referenced or are referenced in non-critical queries (not subject to an SLA). For example, your web site log data might be partitioned by year, month, day, and hour, but if most queries roll up the results by day, perhaps you only need to partition by year, month, and day.

If you need to reduce the granularity even more, consider creating “buckets”, computed values corresponding to different sets of partition key values. For example, you can use the TRUNC() function with a TIMESTAMP column to group date and time values based on intervals such as week or quarter.

Use smallest appropriate integer types for partition key columns

Although it is tempting to use strings for partition key columns, since those values are turned into HDFS directory names anyway, you can minimize memory usage by using numeric values for common partition key fields such as YEAR, MONTH, and DAY. Use the smallest integer type that holds the appropriate range of values, typically TINY INT for MONTH and DAY, and SMALLINT for YEAR. Use the EXTRACT() function to pull out individual date and time fields from a TIMESTAMP value, and CAST() the return value to the appropriate integer type.

Choose an appropriate Parquet block size

By default, the Impala INSERT ... SELECT statement creates Parquet files with a 256 MB block size. (This default was changed in Impala 2.0. Formerly, the limit was 1 GB, but Impala made conservative estimates about compression, resulting in files that were smaller than 1 GB.)

Each Parquet file written by Impala is a single block, allowing the whole file to be processed as a unit by a single host. As you copy Parquet files into HDFS or between HDFS filesystems, use hdfs dfs -pb to preserve the original block size.

If there is only one or a few data block in your Parquet table, or in a partition that is the only one accessed by a query, then you might experience a slowdown for a different reason: not enough data to take advantage of Impala's parallel distributed queries. Each data block is processed by a single core on one of the DataNodes. In a 100-node cluster of 16-core machines, you could potentially process thousands of data files simultaneously. You want to find a sweet spot between “many tiny files” and “single giant file” that balances bulk I/O and parallel processing. You can set the PARQUET_FILE_SIZE query option before doing an INSERT ... SELECT statement to reduce the size of each generated Parquet file. (Specify the file size as an absolute number of bytes, or in Impala 2.0 and later, in units ending with m for megabytes or g for gigabytes.) Run benchmarks with different file sizes to find the right balance point for your particular data volume.

Gather statistics for all tables used in performance-critical or high-volume join queries

Gather the statistics with the COMPUTE STATS statement.

Minimize the overhead of transmitting results back to the client

Use techniques such as:

- Aggregation. If you need to know how many rows match a condition, the total values of matching values from some column, the lowest or highest matching value, and so on, call aggregate functions such as COUNT(), SUM(), and MAX() in the query rather than sending the result set to an application and doing those computations there. Remember that the size of an unaggregated result set could be huge, requiring substantial time to transmit across the network.
- Filtering. Use all applicable tests in the WHERE clause of a query to eliminate rows that are not relevant, rather than producing a big result set and filtering it using application logic.
- LIMIT clause. If you only need to see a few sample values from a result set, or the top or bottom values from a query using ORDER BY, include the LIMIT clause to reduce the size of the result set rather than asking for the full result set and then throwing most of the rows away.
- Avoid overhead from pretty-printing the result set and displaying it on the screen. When you retrieve the results through `impala-shell`, use `impala-shell` options such as -B and --output_delimiter to produce results without special formatting, and redirect output to a file rather than printing to the screen. Consider using INSE RT ... SELECT to write the results directly to new files in HDFS.

Verify that your queries are planned in an efficient logical manner

Examine the EXPLAIN plan for a query before actually running it.

Verify performance characteristics of queries

Verify that the low-level aspects of I/O, memory usage, network bandwidth, CPU utilization, and so on are within expected ranges by examining the query profile for a query after running it.

Hotspot analysis

In the context of Impala, a hotspot is defined as “an Impala daemon that for a single query or a workload is spending a far greater amount of time processing data relative to its neighbours”.

Before discussing the options to tackle this issue, some background is first required to understand how this problem can occur.

By default, the scheduling of scan based plan fragments is deterministic. This means that for multiple queries needing to read the same block of data, the same node will be picked to host the scan. The default scheduling logic does not take into account node workload from prior queries. The complexity of materializing a tuple depends on a few factors, namely: decoding and decompression. If the tuples are densely packed into data pages due to good encoding/compression ratios, there will be more work required when reconstructing the data. Each compression codec offers different performance tradeoffs and should be considered before writing the data. Due to the deterministic nature of the scheduler, single nodes can become bottlenecks for highly concurrent queries that use the same tables.

If, for example, a Parquet based dataset is tiny, e.g. a small dimension table, such that it fits into a single HDFS block (Impala by default will create 256 MB blocks when Parquet is used, each containing a single row group) then there are a number of options that can be considered to resolve the potential scheduling hotspots when querying this data:

- Te scheduler’s deterministic behaviour can be changed using the following query options: REPLICA_PREFERENCE and RANDOM_REPLICA.
- HDFS caching can be used to cache block replicas. This will cause the Impala scheduler to randomly pick a node that is hosting a cached block replica for the scan. Note, although HDFS caching has benefits, it serves only to help with the reading of raw block data and not cached tuple data, but with the right number of cached replicas (by default, HDFS only caches one replica), even load distribution can be achieved for smaller datasets.
- Do not compress the table data. The uncompressed table data spans more nodes and eliminates skew caused by compression.
- Reduce the Parquet file size via the PARQUET_FILE_SIZE query option when writing the table data. Using this approach the data will span more nodes. However it’s not recommended to drop the size below 32 MB.

Query Join Performance

Joins are the main class of queries that you can tune at the SQL level.

Queries involving join operations often require more tuning than queries that refer to only one table. The maximum size of the result set from a join query is the product of the number of rows in all the joined tables. When joining several tables with millions or billions of rows, any missed opportunity to filter the result set, or other inefficiency in the query, could lead to an operation that does not finish in a practical time and has to be cancelled.

The simplest technique for tuning an Impala join query is to collect statistics on each table involved in the join using the COMPUTE STATS statement, and then to let Impala automatically optimize the query based on the size of each table, number of distinct values of each column, and so on. For accurate statistics about each table, issue the COMPUTE STATS statement after loading the data into that table, and again if the amount of data changes substantially due to operations, such as INSERT, LOAD DATA, or adding a partition.

If statistics are not available for all the tables in the join query, or if Impala chooses a join order that is not the most efficient, you can override the automatic join order optimization by specifying the STRAIGHT_JOIN keyword immediately after the SELECT and any DISTINCT or ALL keywords. In this case, Impala uses the order the tables appear in the query to guide how the joins are processed.

When you use the STRAIGHT_JOIN technique, you must order the tables in the join query manually instead of relying on the Impala optimizer. The optimizer uses sophisticated techniques to estimate the size of the result set at each stage of the join. For manual ordering, use this heuristic approach to start with, and then experiment to fine-tune the order:

- Specify the largest table first. This table is read from disk by each Impala node and so its size is not significant in terms of memory usage during the query.
- Next, specify the smallest table. The contents of the second, third, and so on tables are all transmitted across the network. You want to minimize the size of the result set from each subsequent stage of the join query. The most likely approach involves joining a small table first, so that the result set remains small even as subsequent larger tables are processed.
- Join the next smallest table, then the next smallest, and so on.

For example, if you had tables BIG, MEDIUM, SMALL, and TINY, the logical join order to try would be BIG, TINY, SMALL, MEDIUM.

The terms “largest” and “smallest” refers to the size of the intermediate result set based on the number of rows and columns from each table that are part of the result set. For example, if you join one table sales with another table cust omers, a query might find results from 100 different customers who made a total of 5000 purchases. In that case, you would specify SELECT ... FROM sales JOIN customers ..., putting customers on the right side because it is smaller in the context of this query.

The Impala query planner chooses between different techniques for performing join queries, depending on the absolute and relative sizes of the tables. Broadcast joins are the default, where the right-hand table is considered to be smaller than the left-hand table, and its contents are sent to all the other nodes involved in the query. The alternative technique is known as a partitioned join (not related to a partitioned table), which is more suitable for large tables of roughly equal size. With this technique, portions of each table are sent to appropriate other nodes where those subsets of rows can be processed in parallel. The choice of broadcast or partitioned join also depends on statistics being available for all tables in the join, gathered by the COMPUTE STATS statement.

To see which join strategy is used for a particular query, issue an EXPLAIN statement for the query. If you find that a query uses a broadcast join when you know through benchmarking that a partitioned join would be more efficient, or vice versa, add a hint to the query to specify the precise join mechanism to use.

How Joins Are Processed when Statistics Are Unavailable

If table or column statistics are not available for some tables in a join, Impala still reorders the tables using the information that is available. Tables with statistics are placed on the left side of the join order, in descending order of cost based on overall size and cardinality. Tables without statistics are treated as zero-size, that is, they are always placed on the right side of the join order.

Overriding Join Reordering with STRAIGHT_JOIN

If an Impala join query is inefficient because of outdated statistics or unexpected data distribution, you can keep Impala from reordering the joined tables by using the STRAIGHT_JOIN keyword immediately after the SELECT and any DISTINCT or ALL keywords. The STRAIGHT_JOIN keyword turns off the reordering of join clauses that Impala does internally, and produces a plan that relies on the join clauses being ordered optimally in the query text.

Note: The STRAIGHT_JOIN hint affects the join order of table references in the query block containing the hint. It does not affect the join order of nested queries, such as views, inline views, or WHERE-clause subqueries. To use this hint for performance tuning of complex queries, apply the hint to all query blocks that need a fixed join order.

Table and Column Statistics

Impala can do better optimization for complex or multi-table queries when it has access to statistics about the volume of data and how the values are distributed. Impala uses this information to help parallelize and distribute the work for a query. For example, optimizing join queries requires a way of determining if one table is “bigger” than another,

which is a function of the number of rows and the average row size for each table. This topic described the categories of statistics Impala can work with, and how to produce them and keep them up to date.

Overview of table statistics

The Impala query planner can make use of statistics about entire tables and partitions. This information includes physical characteristics such as the number of rows, number of data files, the total size of the data files, and the file format. For partitioned tables, the numbers are calculated per partition, and as totals for the whole table. This metadata is stored in the Metastore database, and can be updated by either Impala or Hive.

If a number is not available, the value -1 is used as a placeholder. Some numbers, such as number and total sizes of data files, are always kept up to date because they can be calculated cheaply, as part of gathering HDFS block metadata.

The following example shows table stats for an unpartitioned Parquet table. The values for the number and sizes of files are always available. Initially, the number of rows is not known, because it requires a potentially expensive scan through the entire table, and so that value is displayed as -1. The COMPUTE STATS statement fills in any unknown table stats values.

```
show table stats parquet_snappy;
+-------+--------+---------+--------------+-------------------+---------+-
------------------+...
| #Rows | #Files | Size    | Bytes Cached | Cache Replication | Format  |
 Incremental stats |...
+-------+--------+---------+--------------+-------------------+---------+---
----------------+...
| -1    | 96     | 23.35GB | NOT CACHED   | NOT CACHED        | PARQUET | f
alse             |...
+-------+--------+---------+--------------+-------------------+---------+-
------------------+...

compute stats parquet_snappy;
+------------------------------------------+
| summary                                  |
+------------------------------------------+
| Updated 1 partition(s) and 6 column(s). |
+------------------------------------------+


show table stats parquet_snappy;
+------------+--------+---------+--------------+-------------------+-------
--+-------------------+...
| #Rows      | #Files | Size    | Bytes Cached | Cache Replication | Format
  | Incremental stats |...
+------------+--------+---------+--------------+-------------------+-------
--+-------------------+...
| 1000000000 | 96     | 23.35GB | NOT CACHED   | NOT CACHED        | PARQ
UET | false            |...
+------------+--------+---------+--------------+-------------------+-------
--+-------------------+...
```

To check that table statistics are available for a table, and see the details of those statistics, use the statement SHOW TABLE STATS *TABLE_NAME*.

Overview of Column Statistics

The Impala query planner can make use of statistics about individual columns when that metadata is available in the metastore database. This technique is most valuable for columns compared across tables in join queries, to help estimate how many rows the query will retrieve from each table. These statistics are also important for correlated subqueries using the EXISTS or IN operators, which are processed internally the same way as join queries.

The following example shows column stats for an unpartitioned Parquet table. The values for the maximum and average sizes of some types are always available, because those figures are constant for numeric and other fixed-size types. Initially, the number of distinct values is not known, because it requires a potentially expensive scan through the entire table, and so that value is displayed as -1. The same applies to maximum and average sizes of variable-sized types, such as STRING. The COMPUTE STATS statement fills in most unknown column stats values. (It does not record the number of NULL values, because currently Impala does not use that figure for query optimization.)

```
show column stats parquet_snappy;
+-------------+----------+------------------+--------+----------+----------+
| Column      | Type     | #Distinct Values | #Nulls | Max Size | Avg Size |
+-------------+----------+------------------+--------+----------+----------+
| id          | BIGINT   | -1               | -1     | 8        | 8        |
| val         | INT      | -1               | -1     | 4        | 4        |
| zerofill    | STRING   | -1               | -1     | -1       | -1       |
| name        | STRING   | -1               | -1     | -1       | -1       |
| assertion   | BOOLEAN  | -1               | -1     | 1        | 1        |
| location_id | SMALLINT | -1               | -1     | 2        | 2        |
+-------------+----------+------------------+--------+----------+----------+

compute stats parquet_snappy;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 6 column(s). |
+-----------------------------------------+
show column stats parquet_snappy;
+-------------+----------+------------------+--------+----------+--------
-----------+
| Column      | Type     | #Distinct Values | #Nulls | Max Size | Avg Size
        |
+-------------+----------+------------------+--------+----------+--------
-----------+
| id          | BIGINT   | 183861280        | -1     | 8        | 8
        |
| val         | INT      | 139017           | -1     | 4        | 4
         |
| zerofill    | STRING   | 101761           | -1     | 6        | 6
        |
| name        | STRING   | 145636240        | -1     | 22       | 13.0002
0027160645 |
| assertion   | BOOLEAN  | 2                | -1     | 1        | 1
        |
| location_id | SMALLINT | 339              | -1     | 2        | 2
         |
+-------------+----------+------------------+--------+----------+----------
---------+
```

Note:

For column statistics to be effective in Impala, you also need to have table statistics for the applicable tables. When you use the Impala COMPUTE STATS statement, both table and column statistics are automatically gathered at the same time, for all columns in the table.

Note: Because Impala does not currently use the NULL count during query planning, Impala speeds up the COMPUTE STATS statement by skipping this NULL counting.

To check whether column statistics are available for a particular set of columns, use the SHOW COLUMN STATS *TABLE_NAME* statement, or check the extended EXPLAIN output for a query against that table that refers to those columns.

How Table and Column Statistics Work for Partitioned Tables

When you use Impala for “big data”, you are highly likely to use partitioning for your biggest tables, the ones representing data that can be logically divided based on dates, geographic regions, or similar criteria. The table and column statistics are especially useful for optimizing queries on such tables. For example, a query involving one year might involve substantially more or less data than a query involving a different year, or a range of several years. Each query might be optimized differently as a result.

The following examples show how table and column stats work with a partitioned table. The table for this example is partitioned by year, month, and day. For simplicity, the sample data consists of 5 partitions, all from the same year and month. Table stats are collected independently for each partition. (In fact, the SHOW PARTITIONS statement displays exactly the same information as SHOW TABLE STATS for a partitioned table.) Column stats apply to the entire table, not to individual partitions. Because the partition key column values are represented as HDFS directories, their characteristics are typically known in advance, even when the values for non-key columns are shown as -1.

```
show partitions year_month_day;
+-------+-------+-----+-------+--------+---------+--------------+----------
---------+---------+...
| year  | month | day | #Rows | #Files | Size    | Bytes Cached | Cache Repl
ication | Format  |...
+-------+-------+-----+-------+--------+---------+--------------+--------
-----------+---------+...
| 2013  | 12    | 1   | -1    | 1      | 2.51MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 2   | -1    | 1      | 2.53MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 3   | -1    | 1      | 2.52MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 4   | -1    | 1      | 2.51MB  | NOT CACHED   | NOT CAC
HED        | PARQUET |...
| 2013  | 12    | 5   | -1    | 1      | 2.52MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| Total |       |     | -1    | 5      | 12.58MB | 0B           |
        |         |...
+-------+-------+-----+-------+--------+---------+--------------+-----------
--------+---------+...

show table stats year_month_day;
+-------+-------+-----+-------+--------+---------+--------------+----------
---------+---------+...
| year  | month | day | #Rows | #Files | Size    | Bytes Cached | Cache Repl
ication | Format  |...
+-------+-------+-----+-------+--------+---------+--------------+--------
-----------+---------+...
| 2013  | 12    | 1   | -1    | 1      | 2.51MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 2   | -1    | 1      | 2.53MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 3   | -1    | 1      | 2.52MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 4   | -1    | 1      | 2.51MB  | NOT CACHED   | NOT CAC
HED        | PARQUET |...
| 2013  | 12    | 5   | -1    | 1      | 2.52MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| Total |       |     | -1    | 5      | 12.58MB | 0B           |
        |         |...
+-------+-------+-----+-------+--------+---------+--------------+-----------
--------+---------+...

show column stats year_month_day;
+-----------+---------+-----------------+--------+----------+----------+
| Column    | Type    | #Distinct Values | #Nulls | Max Size | Avg Size |
```

```
+-----------+---------+------------------+--------+----------+----------+
| id        | INT     | -1               | -1     | 4        | 4        |
| val       | INT     | -1               | -1     | 4        | 4        |
| zfill     | STRING  | -1               | -1     | -1       | -1       |
| name      | STRING  | -1               | -1     | -1       | -1       |
| assertion | BOOLEAN | -1               | -1     | 1        | 1        |
| year      | INT     | 1                | 0      | 4        | 4        |
| month     | INT     | 1                | 0      | 4        | 4        |
| day       | INT     | 5                | 0      | 4        | 4        |
+-----------+---------+------------------+--------+----------+----------+

compute stats year_month_day;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 5 partition(s) and 5 column(s). |
+-----------------------------------------+

show table stats year_month_day;
+-------+-------+-----+--------+--------+---------+--------------+-------
------------+---------+...
| year  | month | day | #Rows  | #Files | Size    | Bytes Cached | Cache
Replication | Format  |...
+-------+-------+-----+--------+--------+---------+--------------+-------
------------+---------+...
| 2013  | 12    | 1   | 93606  | 1      | 2.51MB  | NOT CACHED   | NOT CA
CHED        | PARQUET |...
| 2013  | 12    | 2   | 94158  | 1      | 2.53MB  | NOT CACHED   | NOT CA
CHED        | PARQUET |...
| 2013  | 12    | 3   | 94122  | 1      | 2.52MB  | NOT CACHED   | NOT CA
CHED        | PARQUET |...
| 2013  | 12    | 4   | 93559  | 1      | 2.51MB  | NOT CACHED   | NOT CA
CHED        | PARQUET |...
| 2013  | 12    | 5   | 93845  | 1      | 2.52MB  | NOT CACHED   | NOT CA
CHED        | PARQUET |...
| Total |       |     | 469290 | 5      | 12.58MB | 0B           |
          |         |...
+-------+-------+-----+--------+--------+---------+--------------+-------
------------+---------+...

show column stats year_month_day;
+-----------+---------+------------------+--------+----------+------------
-------+
| Column    | Type    | #Distinct Values | #Nulls | Max Size | Avg Size
     |
+-----------+---------+------------------+--------+----------+--------------
-----+
| id        | INT     | 511129           | -1     | 4        | 4
      |
| val       | INT     | 364853           | -1     | 4        | 4
     |
| zfill     | STRING  | 311430           | -1     | 6        | 6
     |
| name      | STRING  | 471975           | -1     | 22       | 13.0016002655
0293 |
| assertion | BOOLEAN | 2                | -1     | 1        | 1
      |
| year      | INT     | 1                | 0      | 4        | 4
     |
| month     | INT     | 1                | 0      | 4        | 4
     |
| day       | INT     | 5                | 0      | 4        | 4
     |
```

```
+-----------+---------+------------------+--------+----------+-----------
--------+
```

If you run the Hive statement ANALYZE TABLE COMPUTE STATISTICS FOR COLUMNS, Impala can only use the resulting column statistics if the table is unpartitioned. Impala cannot use Hive-generated column statistics for a partitioned table.

Detecting Missing Statistics

You can check whether a specific table has statistics using the SHOW TABLE STATS statement (for any table) or the SHOW PARTITIONS statement (for a partitioned table). Both statements display the same information. If a table or a partition does not have any statistics, the #Rows field contains -1. Once you compute statistics for the table or partition, the #Rows field changes to an accurate value.

The following example shows a table that initially does not have any statistics. The SHOW TABLE STATS statement displays different values for #Rows before and after the COMPUTE STATS operation.

```
[localhost:21000] > create table no_stats (x int);
[localhost:21000] > show table stats no_stats;
+-------+--------+------+--------------+--------+-------------------+
| #Rows | #Files | Size | Bytes Cached | Format | Incremental stats |
+-------+--------+------+--------------+--------+-------------------+
| -1    | 0      | 0B   | NOT CACHED   | TEXT   | false             |
+-------+--------+------+--------------+--------+-------------------+
[localhost:21000] > compute stats no_stats;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 1 column(s). |
+-----------------------------------------+
[localhost:21000] > show table stats no_stats;
+-------+--------+------+--------------+--------+-------------------+
| #Rows | #Files | Size | Bytes Cached | Format | Incremental stats |
+-------+--------+------+--------------+--------+-------------------+
| 0     | 0      | 0B   | NOT CACHED   | TEXT   | false             |
+-------+--------+------+--------------+--------+-------------------+
```

The following example shows a similar progression with a partitioned table. Initially, #Rows is -1. After a COMP UTE STATS operation, #Rows changes to an accurate value. Any newly added partition starts with no statistics, meaning that you must collect statistics after adding a new partition.

```
[localhost:21000] > create table no_stats_partitioned (x int) partitioned by
 (year smallint);
[localhost:21000] > show table stats no_stats_partitioned;
+-------+-------+--------+------+--------------+--------+-------------------
+
| year  | #Rows | #Files | Size | Bytes Cached | Format | Incremental stats
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
| Total | -1    | 0      | 0B   | 0B           |        |
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
[localhost:21000] > show partitions no_stats_partitioned;
+-------+-------+--------+------+--------------+--------+-------------------
+
| year  | #Rows | #Files | Size | Bytes Cached | Format | Incremental stats
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
```

```
| Total | -1    | 0      | 0B   | 0B           |        |
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
[localhost:21000] > alter table no_stats_partitioned add partition (year=
2013);
[localhost:21000] > compute stats no_stats_partitioned;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 1 column(s). |
+-----------------------------------------+
[localhost:21000] > alter table no_stats_partitioned add partition (year=
2014);
[localhost:21000] > show partitions no_stats_partitioned;
+-------+-------+--------+------+--------------+--------+-------------------
+
| year  | #Rows | #Files | Size | Bytes Cached | Format | Incremental stats
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
| 2013  | 0     | 0      | 0B   | NOT CACHED   | TEXT   | false
 |
| 2014  | -1    | 0      | 0B   | NOT CACHED   | TEXT   | false
 |
| Total | 0     | 0      | 0B   | 0B           |        |
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
```

Note: Because the default COMPUTE STATS statement creates and updates statistics for all partitions in a table, if you expect to frequently add new partitions, use the COMPUTE INCREMENTAL STATS syntax instead, which lets you compute stats for a single specified partition, or only for those partitions that do not already have incremental stats.

If checking each individual table is impractical, due to a large number of tables or views that hide the underlying base tables, you can also check for missing statistics for a particular query. Use the EXPLAIN statement to preview query efficiency before actually running the query. Use the query profile output available through the PROFILE command in `impala-shell` or the web UI to verify query execution and timing after running the query. Both the EXPLAIN plan and the PROFILE output display a warning if any tables or partitions involved in the query do not have statistics.

```
[localhost:21000] > create table no_stats (x int);
[localhost:21000] > explain select count(*) from no_stats;
+-----------------------------------------------------------------------
-----------+
| Explain String
         |
+-----------------------------------------------------------------------
---------+
| Estimated Per-Host Requirements: Memory=10.00MB VCores=1
          |
| WARNING: The following tables are missing relevant table and/or column s
tatistics. |
| incremental_stats.no_stats
         |
|
         |
| 03:AGGREGATE [FINALIZE]
          |
| |  output: count:merge(*)
         |
| |
         |
```

```
| 02:EXCHANGE [UNPARTITIONED]
          |
| |
           |
| 01:AGGREGATE
          |
| |  output: count(*)
          |
| |
          |
| 00:SCAN HDFS [incremental_stats.no_stats]
           |
|    partitions=1/1 files=0 size=0B
          |
+-----------------------------------------------------------------------
----------+
```

Because Impala uses the *partition pruning* technique when possible to only evaluate certain partitions, if you have a partitioned table with statistics for some partitions and not others, whether or not the EXPLAIN statement shows the warning depends on the actual partitions used by the query. For example, you might see warnings or not for different queries against the same table:

```
-- No warning because all the partitions for the year 2012 have stats.
EXPLAIN SELECT ... FROM t1 WHERE year = 2012;

-- Missing stats warning because one or more partitions in this range
-- do not have stats.
EXPLAIN SELECT ... FROM t1 WHERE year BETWEEN 2006 AND 2009;
```

To confirm if any partitions at all in the table are missing statistics, you might explain a query that scans the entire table, such as SELECT COUNT(*) FROM *TABLE_NAME*.

Manually Setting Table Statistics with ALTER TABLE

The most crucial piece of data in all the statistics is the number of rows in the table (for an unpartitioned or partitioned table) and for each partition (for a partitioned table). The COMPUTE STATS statement always gathers statistics about all columns, as well as overall table statistics. If it is not practical to do a full COMPUTE STATS or COMPUTE INCREMENTAL STATS operation after adding a partition or inserting data, or if you can see that Impala would produce a more efficient plan if the number of rows was different, you can manually set the number of rows through an ALTER TABLE statement:

```
-- Set total number of rows. Applies to both unpartitioned and partitioned
tables.
alter table TABLE_NAME set tblproperties('numRows'='NEW_VALUE', 'STATS_GE
NERATED_VIA_STATS_TASK'='true');

-- Set total number of rows for a specific partition. Applies to partitioned
 tables only.
-- You must specify all the partition key columns in the PARTITION clause.
alter table TABLE_NAME partition (KEYCOL1=VAL1,KEYCOL2=VAL2...) set tblprop
erties('numRows'='NEW_VALUE', 'STATS_GENERATED_VIA_STATS_TASK'='true');
```

This statement avoids re-scanning any data files. For example:

```
alter table analysis_data set tblproperties('numRows'='1001000000', 'STATS_
GENERATED_VIA_STATS_TASK'='true');
```

For a partitioned table, update both the per-partition number of rows and the number of rows for the whole table. For example:

```
alter table partitioned_data partition(year=2009, month=4) set tblproperties
 ('numRows'='30000', 'STATS_GENERATED_VIA_STATS_TASK'='true');
alter table partitioned_data set tblproperties ('numRows'='1030000', 'STA
TS_GENERATED_VIA_STATS_TASK'='true');
```

In practice, the COMPUTE STATS statement, or COMPUTE INCREMENTAL STATS for a partitioned table, should be fast and convenient enough that this technique is only useful for the very largest partitioned tables. Because the column statistics might be left in a stale state, do not use this technique as a replacement for COMPUTE STATS. Only use this technique if all other means of collecting statistics are impractical, or as a low-overhead operation that you run in between periodic COMPUTE STATS or COMPUTE INCREMENTAL STATS operations.

Manually Setting Column Statistics with ALTER TABLE

You can use the SET COLUMN STATS clause of ALTER TABLE to manually set or change column statistics. Only use this technique in cases where it is impractical to run COMPUTE STATS or COMPUTE INCREMENTAL ST ATS frequently enough to keep up with data changes for a huge table.

You specify a case-insensitive symbolic name for the kind of statistics: numDVs, numNulls, avgSize, maxSize. The key names and values are both quoted. This operation applies to an entire table, not a specific partition.

For example:

```
alter table t1 set column stats x ('numDVs'='2','numNulls'='0');
```

Examples of Using Table and Column Statistics with Impala

The following examples walk through a sequence of SHOW TABLE STATS, SHOW COLUMN STATS, ALTE R TABLE, and SELECT and INSERT statements to illustrate various aspects of how Impala uses statistics to help optimize queries.

This example shows table and column statistics for the STORE column used in the TPC-DS benchmarks for decision support systems. It is a tiny table holding data for 12 stores. Initially, before any statistics are gathered by a COMP UTE STATS statement, most of the numeric fields show placeholder values of -1, indicating that the figures are unknown. The figures that are filled in are values that are easily countable or deducible at the physical level, such as the number of files, total data size of the files, and the maximum and average sizes for data types that have a constant size such as INT, FLOAT, and TIMESTAMP.

```
[localhost:21000] > show table stats store;
+-------+--------+--------+--------+
| #Rows | #Files | Size   | Format |
+-------+--------+--------+--------+
| -1    | 1      | 3.08KB | TEXT   |
+-------+--------+--------+--------+
Returned 1 row(s) in 0.03s
[localhost:21000] > show column stats store;
+--------------------+-----------+------------------+--------+----------+--
--------+
| Column             | Type      | #Distinct Values | #Nulls | Max Size |
Avg Size |
+--------------------+-----------+------------------+--------+----------+
----------+
| s_store_sk         | INT       | -1               | -1     | 4        | 4
       |
| s_store_id         | STRING    | -1               | -1     | -1       | -1
       |
| s_rec_start_date   | TIMESTAMP | -1               | -1     | 16       | 16
       |
```

```
| s_rec_end_date     | TIMESTAMP | -1                | -1     | 16       |
 16       |
| s_closed_date_sk   | INT       | -1                | -1     | 4        | 4
       |
| s_store_name       | STRING    | -1                | -1     | -1       | -1
       |
| s_number_employees | INT       | -1                | -1     | 4        | 4
       |
| s_floor_space      | INT       | -1                | -1     | 4        | 4
       |
| s_hours            | STRING    | -1                | -1     | -1       | -1
       |
| s_manager          | STRING    | -1                | -1     | -1       | -1
       |
| s_market_id        | INT       | -1                | -1     | 4        | 4
       |
| s_geography_class  | STRING    | -1                | -1     | -1       |
 -1       |
| s_market_desc      | STRING    | -1                | -1     | -1       | -1
       |
| s_market_manager   | STRING    | -1                | -1     | -1       | -1
       |
| s_division_id      | INT       | -1                | -1     | 4        | 4
       |
| s_division_name    | STRING    | -1                | -1     | -1       |
 -1       |
| s_company_id       | INT       | -1                | -1     | 4        | 4
       |
| s_company_name     | STRING    | -1                | -1     | -1       | -1
       |
| s_street_number    | STRING    | -1                | -1     | -1       | -1
       |
| s_street_name      | STRING    | -1                | -1     | -1       |
 -1       |
| s_street_type      | STRING    | -1                | -1     | -1       | -1
       |
| s_suite_number     | STRING    | -1                | -1     | -1       | -1
       |
| s_city             | STRING    | -1                | -1     | -1       | -1
       |
| s_county           | STRING    | -1                | -1     | -1       |
 -1       |
| s_state            | STRING    | -1                | -1     | -1       | -1
       |
| s_zip              | STRING    | -1                | -1     | -1       | -1
       |
| s_country          | STRING    | -1                | -1     | -1       | -1
       |
| s_gmt_offset       | FLOAT     | -1                | -1     | 4        | 4
       |
| s_tax_percentage   | FLOAT     | -1                | -1     | 4        | 4
       |
+--------------------+-----------+------------------+--------+----------+--
--------+
Returned 29 row(s) in 0.04s
```

With the Hive ANALYZE TABLE statement for column statistics, you had to specify each column for which to gather statistics. The Impala COMPUTE STATS statement automatically gathers statistics for all columns, because it reads through the entire table relatively quickly and can efficiently compute the values for all the columns. This example shows how after running the COMPUTE STATS statement, statistics are filled in for both the table and all its columns:

```
[localhost:21000] > compute stats store;
```

```
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 29 column(s). |
+-----------------------------------------+
Returned 1 row(s) in 1.88s
[localhost:21000] > show table stats store;
+-------+--------+--------+--------+
| #Rows | #Files | Size   | Format |
+-------+--------+--------+--------+
| 12    | 1      | 3.08KB | TEXT   |
+-------+--------+--------+--------+
Returned 1 row(s) in 0.02s
[localhost:21000] > show column stats store;
+--------------------+-----------+------------------+--------+----------+-
------------------+
| Column             | Type      | #Distinct Values | #Nulls | Max Size | Av
g Size          |
+--------------------+-----------+------------------+--------+----------+-
------------------+
| s_store_sk         | INT       | 12               | -1     | 4        | 4
               |
| s_store_id         | STRING    | 6                | -1     | 16       | 16
               |
| s_rec_start_date   | TIMESTAMP | 4                | -1     | 16       | 16
               |
| s_rec_end_date     | TIMESTAMP | 3                | -1     | 16       | 16
               |
| s_closed_date_sk   | INT       | 3                | -1     | 4        | 4
               |
| s_store_name       | STRING    | 8                | -1     | 5        |
4.25              |
| s_number_employees | INT       | 9                | -1     | 4        | 4
               |
| s_floor_space      | INT       | 10               | -1     | 4        | 4
               |
| s_hours            | STRING    | 2                | -1     | 8        | 7.
083300113677979 |
| s_manager          | STRING    | 7                | -1     | 15       | 12
               |
| s_market_id        | INT       | 7                | -1     | 4        | 4
               |
| s_geography_class  | STRING    | 1                | -1     | 7        | 7
               |
| s_market_desc      | STRING    | 10               | -1     | 94       | 55
.5              |
| s_market_manager   | STRING    | 7                | -1     | 16       | 14
               |
| s_division_id      | INT       | 1                | -1     | 4        | 4
               |
| s_division_name    | STRING    | 1                | -1     | 7        | 7
               |
| s_company_id       | INT       | 1                | -1     | 4        | 4
               |
| s_company_name     | STRING    | 1                | -1     | 7        | 7
               |
| s_street_number    | STRING    | 9                | -1     | 3        | 2.
833300113677979 |
| s_street_name      | STRING    | 12               | -1     | 11       |
6.583300113677979 |
| s_street_type      | STRING    | 8                | -1     | 9        | 4.
833300113677979 |
| s_suite_number     | STRING    | 11               | -1     | 9        |
8.25              |
```

```
| s_city             | STRING    | 2                 | -1     | 8        | 6.
5              |
| s_county           | STRING    | 1                 | -1     | 17       | 17
               |
| s_state            | STRING    | 1                 | -1     | 2        | 2
               |
| s_zip              | STRING    | 2                 | -1     | 5        | 5
               |
| s_country          | STRING    | 1                 | -1     | 13       | 13
               |
| s_gmt_offset       | FLOAT     | 1                 | -1     | 4        | 4
               |
| s_tax_percentage   | FLOAT     | 5                 | -1     | 4        | 4
               |
+--------------------+-----------+------------------+--------+----------+-
------------------+
Returned 29 row(s) in 0.04s
```

The following example shows how statistics are represented for a partitioned table. In this case, we have set up a table to hold the world's most trivial census data, a single STRING field, partitioned by a YEAR column. The table statistics include a separate entry for each partition, plus final totals for the numeric fields. The column statistics include some easily deducible facts for the partitioning column, such as the number of distinct values (the number of partition subdirectories).

```
localhost:21000] > describe census;
+------+----------+---------+
| name | type     | comment |
+------+----------+---------+
| name | string   |         |
| year | smallint |         |
+------+----------+---------+
Returned 2 row(s) in 0.02s
[localhost:21000] > show table stats census;
+-------+-------+--------+------+---------+
| year  | #Rows | #Files | Size | Format  |
+-------+-------+--------+------+---------+
| 2000  | -1    | 0      | 0B   | TEXT    |
| 2004  | -1    | 0      | 0B   | TEXT    |
| 2008  | -1    | 0      | 0B   | TEXT    |
| 2010  | -1    | 0      | 0B   | TEXT    |
| 2011  | 0     | 1      | 22B  | TEXT    |
| 2012  | -1    | 1      | 22B  | TEXT    |
| 2013  | -1    | 1      | 231B | PARQUET |
| Total | 0     | 3      | 275B |         |
+-------+-------+--------+------+---------+
Returned 8 row(s) in 0.02s
[localhost:21000] > show column stats census;
+--------+----------+------------------+--------+----------+----------+
| Column | Type     | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+----------+------------------+--------+----------+----------+
| name   | STRING   | -1               | -1     | -1       | -1       |
| year   | SMALLINT | 7                | -1     | 2        | 2        |
+--------+----------+------------------+--------+----------+----------+
Returned 2 row(s) in 0.02s
```

The following example shows how the statistics are filled in by a COMPUTE STATS statement in Impala.

```
[localhost:21000] > compute stats census;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 3 partition(s) and 1 column(s). |
```

```
+----------------------------------------+
Returned 1 row(s) in 2.16s
[localhost:21000] > show table stats census;
+-------+-------+--------+------+---------+
| year  | #Rows | #Files | Size | Format  |
+-------+-------+--------+------+---------+
| 2000  | -1    | 0      | 0B   | TEXT    |
| 2004  | -1    | 0      | 0B   | TEXT    |
| 2008  | -1    | 0      | 0B   | TEXT    |
| 2010  | -1    | 0      | 0B   | TEXT    |
| 2011  | 4     | 1      | 22B  | TEXT    |
| 2012  | 4     | 1      | 22B  | TEXT    |
| 2013  | 1     | 1      | 231B | PARQUET |
| Total | 9     | 3      | 275B |         |
+-------+-------+--------+------+---------+
Returned 8 row(s) in 0.02s
[localhost:21000] > show column stats census;
+--------+----------+------------------+--------+----------+----------+
| Column | Type     | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+----------+------------------+--------+----------+----------+
| name   | STRING   | 4                | -1     | 5        | 4.5      |
| year   | SMALLINT | 7                | -1     | 2        | 2        |
+--------+----------+------------------+--------+----------+----------+
Returned 2 row(s) in 0.02s
```

You can see how Impala executes a query differently in each case by observing the EXPLAIN output before and after collecting statistics. Measure the before and after query times, and examine the throughput numbers in before and after SUMMARY or PROFILE output, to verify how much the improved plan speeds up performance.

Generating Table and Column Statistics

Use the COMPUTE STATS statement to collect table and column statistics. The COMPUTE STATS variants offer different tradeoffs between computation cost, staleness, and maintenance workflows.

Important:

For a particular table, use either COMPUTE STATS or COMPUTE INCREMENTAL STATS, but never combine the two or alternate between them. If you switch from COMPUTE STATS to COMPUTE INCREMENTAL STATS during the lifetime of a table, or vice versa, drop all statistics by running DROP STATS before making the switch.

COMPUTE STATS

The COMPUTE STATS command collects and sets the table-level and partition-level row counts as well as all column statistics for a given table. The collection process is CPU-intensive and can take a long time to complete for very large tables.

To speed up COMPUTE STATS consider the following options which can be combined.

- Limit the number of columns for which statistics are collected to increase the efficiency of COMPUTE STATS. Queries benefit from statistics for those columns involved in filters, join conditions, group by or partition by clauses. Other columns are good candidates to exclude from COMPUTE STATS. This feature is available since Impala 2.12.
- Set the MT_DOP query option to use more threads within each participating impalad to compute the statistics faster - but not more efficiently. Note that computing stats on a large table with a high MT_DOP value can negatively affect other queries running at the same time if the COMPUTE STATS claims most CPU cycles.
- Consider the experimental extrapolation and sampling features (see below) to further increase the efficiency of computing stats.

COMPUTE STATS is intended to be run periodically, e.g. weekly, or on-demand when the contents of a table have changed significantly. Due to the high resource utilization and long response time of COMPUTE STATS, it is most practical to run it in a scheduled maintenance window where the Impala cluster is idle enough to accommodate

the expensive operation. The degree of change that qualifies as “significant” depends on the query workload, but typically, if 30% of the rows have changed then it is recommended to recompute statistics.

If you reload a complete new set of data for a table, but the number of rows and number of distinct values for each column is relatively unchanged from before, you do not need to recompute stats for the table.

Extrapolation and Sampling

Impala supports extrapolation and sampling to alleviate the following common issues for computing and maintaining statistics on very large tables:

- Newly added partitions do not have row count statistics. Table scans that only access those new partitions are treated as not having stats. Similarly, table scans that access both new and old partitions estimate the scan cardinality based on those old partitions that have stats, and the new partitions without stats are treated as having 0 rows.
- The row counts of existing partitions become stale when data is added or dropped.
- Computing stats for tables with a 100,000 or more partitions might fail or be very slow due to the high cost of updating the partition metadata in the Hive Metastore.
- With transient compute resources it is important to minimize the time from starting a new cluster to successfully running queries. Since the cluster might be relatively short-lived, users might prefer to quickly collect stats that are "good enough" as opposed to spending a lot of time and resources on computing full-fidelity stats.

For very large tables, it is often wasteful or impractical to run a full COMPUTE STATS to address the scenarios above on a frequent basis.

The sampling feature makes COMPUTE STATS more efficient by processing a fraction of the table data, and the extrapolation feature aims to reduce the frequency at which COMPUTE STATS needs to be re-run by estimating the row count of new and modified partitions.

The sampling and extrapolation features are disabled by default. They can be enabled globally or for specific tables, as follows.

- Set the impalad start-up configuration --enable_stats_extrapolation to enable the features globally.
- To enable them only for a specific table, set the impala.enable.stats.extrapolation table property to true for the table. The table-level property overrides the global setting, so it is also possible to enable sampling and extrapolation globally, but disable it for specific tables by setting the table property to false. For example:

```
ALTER TABLE mytable test_table SET TBLPROPERTIES("impala.enable.stats.ex
trapolation"="true");
```

Note: Why are these features experimental? Due to their probabilistic nature it is possible that these features perform pathologically poorly on tables with extreme data/file/size distributions. Since it is not feasible for us to test all possible scenarios we only cautiously advertise these new capabilities. That said, the features have been thoroughly tested and are considered functionally stable. If you decide to give these features a try, please tell us about your experience at user@impala.apache.org! We rely on user feedback to guide future improvements in statistics collection.

Stats Extrapolation

The main idea of stats extrapolation is to estimate the row count of new and modified partitions based on the result of the last COMPUTE STATS. Enabling stats extrapolation changes the behavior of COMPUTE STATS, as well as the cardinality estimation of table scans. COMPUTE STATS no longer computes and stores per-partition row counts, and instead, only computes a table-level row count together with the total number of file bytes in the table at that time. No partition metadata is modified. The input cardinality of a table scan is estimated by converting the data volume of relevant partitions to a row count, based on the table-level row count and file bytes statistics. It is assumed that within the same table, different sets of files with the same data volume correspond to the similar number of rows on average. With extrapolation enabled, the scan cardinality estimation ignores per-partition row counts. It only relies on the table-level statistics and the scanned data volume.

The SHOW TABLE STATS and EXPLAIN commands distinguish between row counts stored in the Hive Metastore, and the row counts extrapolated based on the above process.

Sampling

A TABLESAMPLE clause may be added to COMPUTE STATS to limit the percentage of data to be processed. The final statistics are obtained by extrapolating the statistics from the data sample over the entire table. The extrapolated statistics are stored in the Hive Metastore, just as if no sampling was used. The following example runs COMPUTE STATS over a 10 percent data sample.

```
COMPUTE STATS test_table TABLESAMPLE SYSTEM(10) ;
```

We have found that a 10 percent sampling rate typically offers a good tradeoff between statistics accuracy and execution cost. A sampling rate well below 10 percent has shown poor results and is not recommended.

Important: Sampling-based techniques sacrifice result accuracy for execution efficiency, so your mileage may vary for different tables and columns depending on their data distribution. The extrapolation procedure Impala uses for estimating the number of distinct values per column is inherently non-detetministic, so your results may even vary between runs of COMPUTE STATS TABLESAMPLE, even if no data has changed.

COMPUTE INCREMENTAL STATS

In Impala 2.1.0 and higher, you can use the COMPUTE INCREMENTAL STATS and DROP INCREMENTAL STATS commands. The INCREMENTAL clauses work with incremental statistics, a specialized feature for partitioned tables.

When you compute incremental statistics for a partitioned table, by default Impala only processes those partitions that do not yet have incremental statistics. By processing only newly added partitions, you can keep statistics up to date without incurring the overhead of reprocessing the entire table each time.

You can also compute or drop statistics for a specified subset of partitions by including a PARTITION clause in the COMPUTE INCREMENTAL STATS or DROP INCREMENTAL STATS statement.

Important:

When you run COMPUTE INCREMENTAL STATS on a table for the first time, the statistics are computed again from scratch regardless of whether the table already has statistics. Therefore, expect a one-time resource-intensive operation for scanning the entire table when running COMPUTE INCREMENTAL STATS for the first time on a given table.

The metadata for incremental statistics is handled differently from the original style of statistics:

- Issuing a COMPUTE INCREMENTAL STATS without a partition clause causes Impala to compute incremental stats for all partitions that do not already have incremental stats. This might be the entire table when running the command for the first time, but subsequent runs should only update new partitions. You can force updating a partition that already has incremental stats by issuing a DROP INCREMENTAL STATS before running COMPUTE INCREMENTAL STATS.
- The SHOW TABLE STATS and SHOW PARTITIONS statements now include an additional column showing whether incremental statistics are available for each column. A partition could already be covered by the original type of statistics based on a prior COMPUTE STATS statement, as indicated by a value other than -1 under the #Rows column. Impala query planning uses either kind of statistics when available.
- COMPUTE INCREMENTAL STATS takes more time than COMPUTE STATS for the same volume of data. Therefore it is most suitable for tables with large data volume where new partitions are added frequently, making it impractical to run a full COMPUTE STATS operation for each new partition. For unpartitioned tables, or partitioned tables that are loaded once and not updated with new partitions, use the original COMPUTE STATS syntax.
- COMPUTE INCREMENTAL STATS uses some memory in the `catalogd` process, proportional to the number of partitions and number of columns in the applicable table. The memory overhead is approximately 400 bytes for

each column in each partition. This memory is reserved in the `catalogd` daemon, the `statestored` daemon, and in each instance of the `impalad` daemon.

- In cases where new files are added to an existing partition, issue a REFRESH statement for the table, followed by a DROP INCREMENTAL STATS and COMPUTE INCREMENTAL STATS sequence for the changed partition.
- The DROP INCREMENTAL STATS statement operates only on a single partition at a time. To remove statistics (whether incremental or not) from all partitions of a table, issue a DROP STATS statement with no INCREMENTAL or PARTITION clauses.

The following considerations apply to incremental statistics when the structure of an existing table is changed (known as *schema evolution*):

- If you use an ALTER TABLE statement to drop a column, the existing statistics remain valid and COMPUTE INCREMENTAL STATS does not rescan any partitions.
- If you use an ALTER TABLE statement to add a column, Impala rescans all partitions and fills in the appropriate column-level values the next time you run COMPUTE INCREMENTAL STATS.
- If you use an ALTER TABLE statement to change the data type of a column, Impala rescans all partitions and fills in the appropriate column-level values the next time you run COMPUTE INCREMENTAL STATS.
- If you use an ALTER TABLE statement to change the file format of a table, the existing statistics remain valid and a subsequent COMPUTE INCREMENTAL STATS does not rescan any partitions.

Runtime Filtering

Runtime filtering is a wide-ranging optimization feature available in Impala. When only a fraction of the data in a table is needed for a query against a partitioned table or to evaluate a join condition, Impala determines the appropriate conditions while the query is running, and broadcasts that information to all the `impalad` nodes that are reading the table so that they can avoid unnecessary I/O to read partition data, and avoid unnecessary network transmission by sending only the subset of rows that match the join keys across the network.

Runtime filtering is primarily used:

- To optimize queries against large partitioned tables (under the name *dynamic partition pruning*)
- To optimize joins of large tables

The following terms are used in this topic to describe runtime filtering.

plan fragment

Impala decomposes each query into smaller units of work that are distributed across the cluster. Wherever possible, a data block is read, filtered, and aggregated by plan fragments executing on the same host. For some operations, such as joins and combining intermediate results into a final result set, data is transmitted across the network from one Impala daemon to another.

SCAN and HASH JOIN plan nodes

In the Impala query plan, a *scan node* performs the I/O to read from the underlying data files. Although this is an expensive operation from the traditional database perspective, Hadoop clusters and Impala are optimized to do this kind of I/O in a highly parallel fashion. The major potential cost savings come from using the columnar Parquet format (where Impala can avoid reading data for unneeded columns) and partitioned tables (where Impala can avoid reading data for unneeded partitions).

Most Impala joins use the *hash join* mechanism. (It is only fairly recently that Impala started using the nested-loop join technique, for certain kinds of non-equijoin queries.) In a hash join, when evaluating join conditions from two tables, Impala constructs a hash table in memory with all the different column values from the table on one side of the join. Then, for each row from the table on the other side of the join, Impala tests whether the relevant column values are in this hash table or not.

hash join

In a hash join, when evaluating join conditions from two tables, Impala constructs a hash table in memory with all the different column values from the table on one side of the join. Then, for each row from the table on the other side of the join, Impala tests whether the relevant column values are in this hash table or not.

- A *hash join node* constructs such an in-memory hash table, then performs the comparisons to identify which rows match the relevant join conditions and should be included in the result set (or at least sent on to the subsequent intermediate stage of query processing). Because some of the input for a hash join might be transmitted across the network from another host, it is especially important from a performance perspective to prune out ahead of time any data that is known to be irrelevant.

  The more distinct values are in the columns used as join keys, the larger the in-memory hash table and thus the more memory required to process the query.

broadcast join vs shuffle join

In a broadcast join, the table from one side of the join (typically the smaller table) is sent in its entirety to all the hosts involved in the query. Then each host can compare its portion of the data from the other (larger) table against the full set of possible join keys. In a shuffle join, there is no obvious “smaller” table, and so the contents of both tables are divided up, and corresponding portions of the data are transmitted to each host involved in the query.

A shuffle join is sometimes referred to in Impala as a *partitioned join*.

build and probe phases

When Impala processes a join query, the *build phase* is where the rows containing the join key columns, typically for the smaller table, are transmitted across the network and built into an in-memory hash table data structure on one or more destination nodes. The *probe phase* is where data is read locally (typically from the larger table) and the join key columns are compared to the values in the in-memory hash table. The corresponding input sources (tables, subqueries, and so on) for these phases are referred to as the *build side* and the *probe side*.

Runtime Filters

The *filter* that is transmitted between plan fragments is essentially a list of values for join key columns. When this list of values is transmitted in time to a scan node, Impala can filter out non-matching values immediately after reading them, rather than transmitting the raw data to another host to compare against the in-memory hash table on that host.

Impala supports the following types of filters based on the payload:

- *Bloom filter*: For HDFS-based tables, the Bloom filter uses a probability-based algorithm to determine all possible matching values. The probability-based aspects means that the filter might include some non-matching values, but if so, that does not cause any inaccuracy in the final results.
- *Min-max filter*: The filter is a data structure representing a minimum and maximum value. These filters are passed to Kudu to reduce the number of rows returned to Impala when scanning the probe side of the join. This filter currently only applies to Kudu tables.

Based on how filters from all join instances are aggregated, each of the above filters can be categorized as one of the following:

- *Broadcast filter*: A broadcast filter reflects the complete list of relevant values and can be immediately evaluated by a scan node.

  Broadcast filters are also classified as local or global. With a local broadcast filter, the information in the filter is used by a subsequent query fragment that is running on the same host that produced the filter. A non-local broadcast filter must be transmitted across the network to a query fragment that is running on a different host. Impala designates 3 hosts to each produce non-local broadcast filters, to guard against the possibility of a single slow host taking too long. Depending on the setting of the RUNTIME_FILTER_MODE query option (LOCAL or GLOBAL), Impala either uses a conservative optimization strategy where filters are only consumed on the same host that produced them, or a more aggressive strategy where filters are eligible to be transmitted across the network. The default for runtime filtering is the GLOBAL setting.

- *Partitioned filter*: A partitioned filter reflects only the values processed by one host in the cluster; all the partitioned filters must be combined into one (by the coordinator node) before the scan nodes can use the results to accurately filter the data as it is read from storage.

File Format Considerations for Runtime Filtering

Parquet tables get the most benefit from the runtime filtering optimizations. Runtime filtering can speed up join queries against partitioned or unpartitioned Parquet tables, and single-table queries against partitioned Parquet tables.

For other file formats (text, Avro, RCFile, and SequenceFile), runtime filtering speeds up queries against partitioned tables only. Because partitioned tables can use a mixture of formats, Impala produces the filters in all cases, even if they are not ultimately used to optimize the query.

Wait Intervals for Runtime Filters

Because it takes time to produce runtime filters, especially for partitioned filters that must be combined by the coordinator node, there is a time interval above which it is more efficient for the scan nodes to go ahead and construct their intermediate result sets, even if that intermediate data is larger than optimal. If it only takes a few seconds to produce the filters, it is worth the extra time if pruning the unnecessary data can save minutes in the overall query time. You can specify the maximum wait time in milliseconds using the RUNTIME_FILTER_WAIT_TIME_MS query option.

By default, each scan node waits for up to 1 second (1000 milliseconds) for filters to arrive. If all filters have not arrived within the specified interval, the scan node proceeds, using whatever filters did arrive to help avoid reading unnecessary data. If a filter arrives after the scan node begins reading data, the scan node applies that filter to the data that is read after the filter arrives, but not to the data that was already read.

If the cluster is relatively busy and your workload contains many resource-intensive or long-running queries, consider increasing the wait time so that complicated queries do not miss opportunities for optimization. If the cluster is lightly loaded and your workload contains many small queries taking only a few seconds, consider decreasing the wait time to avoid the 1 second delay for each query.

Query Options for Runtime Filtering

The following query options control runtime filtering.

- RUNTIME_FILTER_MODE

  This query option controls how extensively the filters are transmitted between hosts. By default, it is set to the highest level (GLOBAL).
- The other query options are tuning knobs that you typically only adjust after doing performance testing, and that you might want to change only for the duration of a single expensive query.
  - MAX_NUM_RUNTIME_FILTERS
  - DISABLE_ROW_RUNTIME_FILTERING
  - RUNTIME_FILTER_MAX_SIZE
  - RUNTIME_FILTER_MIN_SIZE
  - RUNTIME_BLOOM_FILTER_SIZE

Runtime Filtering and Query Plans

In the same way the query plan displayed by the EXPLAIN statement includes information about predicates used by each plan fragment, it also includes annotations showing whether a plan fragment produces or consumes a runtime filter.

- A plan fragment that produces a filter includes an annotation such as runtime filters: *FILTER_ID* <- *TABLE.COLUMN*
- A plan fragment that consumes a filter includes an annotation such as runtime filters: *FILTER_ID* -> *TABLE.COLUMN*

Setting the query option EXPLAIN_LEVEL=2 adds additional annotations showing the type of the filter:

- *FILTER_ID*[bloom] (for HDFS-based tables)
- *FILTER_ID*[min_max] (for Kudu tables)

The query profile (displayed by the PROFILE command in `impala-shell`) contains both the EXPLAIN plan and more detailed information about the internal workings of the query. The profile output includes the Filter routing table section with information about each filter based on its ID.

Tuning and Troubleshooting Queries that Use Runtime Filtering

These tuning and troubleshooting procedures apply to queries that are resource-intensive enough, long-running enough, and frequent enough that you can devote special attention to optimizing them individually.

Use the EXPLAIN statement and examine the runtime filters: lines to determine whether runtime filters are being applied to the WHERE predicates and join clauses that you expect. For example, runtime filtering does not apply to queries that use the nested loop join mechanism due to non-equijoin operators.

Make sure statistics are up-to-date for all tables involved in the queries. Use the COMPUTE STATS statement after loading data into non-partitioned tables, and COMPUTE INCREMENTAL STATS after adding new partitions to partitioned tables.

If join queries involving large tables use unique columns as the join keys, for example joining a primary key column with a foreign key column, the overhead of producing and transmitting the filter might outweigh the performance benefit because not much data could be pruned during the early stages of the query. For such queries, consider setting the query option RUNTIME_FILTER_MODE=OFF.

Distribute Runtime Filter Aggregation

The process of aggregating runtime filters at runtime can exert a substantial memory load on the coordinator. In response to this challenge, we initially introduced local aggregation of runtime filters within a single executor node, aiming to reduce the strain on the coordinator by transmitting filter updates only after local aggregation.

As Impala clusters scale up, the limitations of local filter aggregation become apparent, particularly in scenarios involving numerous nodes. This situation places significant memory stress on the coordinator node.

To mitigate this challenge, we have implemented a solution that distributes the runtime filter aggregation across specific Impala backends. The final aggregation takes place after this distributed aggregation, and the results are then published to the coordinator. This strategic distribution effectively alleviates the memory pressure on the coordinator, resulting in a more streamlined and accelerated runtime filter aggregation process.

The control mechanism for this enhancement lies in the query option MAX_NUM_FILTERS_AGGREGATED_PER_HOST. This option determines the number of executor nodes denoted as M, which are randomly selected as intermediate aggregators for the runtime filter. This approach optimizes the runtime filter aggregation process and contributes to overall system performance.

```
M = ceil(N / MAX_NUM_FILTERS_AGGREGATED_PER_HOST)
```

Where N is the number of all backend executors running the query, excluding the coordinator.

For instance, in a cluster of 400 nodes, each coordinator and intermediate aggregator would be configured to handle 20 filter updates. By aligning the MAX_NUM_FILTERS_AGGREGATED_PER_HOST with the square root of the cluster size, this approach aims to strike a balance and improve the runtime filter aggregation process, particularly in larger-scale deployments.

Skip Scheduling Bloom Filter

As part of performance optimization, you can skip scheduling bloom filter from join node with certain characteristics.

PK-FK joins between a dimension table and a fact table are common occurrences in a query. Such joins often do not involve any predicate filters in the dimension table. As a result, a bloom filter generated from this kind of dimension table scan (PK) will most likely contain all values from the fact table column (FK). It becomes ineffective to generate this filter because it is unlikely to reject any rows, especially if the bloom filter size is large and has a high false positive probability (FPP) estimate.

As part of this optimization, this release skips scheduling bloom filter from join node that has the following characteristics:

- Build side is full table scan and has hard estimates.
- The build scan does not have any predicate filter nor consume any runtime filter.
- The join node is assumed to have PK-FK relationship.
- The planned bloom filter has a result with an estimate higher than the default set through this flagmax_filter_error _rate_from_full_scan (default to 0.9).

The following flag is added to control the generation of bloom filters.

```
max_filter_error_rate_from_full_scan = 0.9,
```

This flag allows skip generation of the bloom runtime filter that is generated from a full build scan and has resulting error rate estimation that is higher than the value set in this flag after the filter size limit is applied. This config may get ignored if target error rate is set with higher value through RUNTIME_FILTER_ERROR_RATE query option or max_filter_error_rate backend flag. Setting a value less than 0 will disable this runtime filter reduction feature. Similarly, setting max_filter_error_rate_from_full_scan to a value less than 0 will disable this runtime filter reduction feature.

Min/Max Filtering

This feature is available for users of Cloudera Data Warehouse release August 27, 2021 or later. In addition to filtering at the partition level, the Parquet file format supports filtering at three levels: row group, page and row levels based on the minimum and maximum values of row group or page, and the value of the row in the file. These minimum/maximum column values are stored in the footer of the file. If the range between the minimum and maximum value in the file does not overlap with the range of data specified by the query, then the system skips the row group, page or row during scans.

CDW August 27, 2021 release enables min/max filters, by default, for equi-joins on lexical (leading column) or Z-Order (all columns) sorted columns in a Parquet table created by Impala. When such a sorted column is also the equi-join column, the min/max values in the filter are computed from the hash table build side and applied to the sorted column. In this way, the user can take advantage of Impala sorting the min/max values in column index in each data file for the table by quickly eliminating non-matching row groups, pages or rows. The query option minmax_filter_so rted_columns which defaults to true can be used to control the feature. When minmax_filter_sorted_columns is true, the patch will generate min/max filters only for the sort columns. The normal control knobs minmax_filter_threshold (for threshold) and minmax_filtering_level (for filtering level) will also continue to work. When the threshold is 0, the patch automatically assigns a reasonable value for the threshold, and selects PAGE to be the filtering level.

In the backend, the skipped pages are quickly found by taking a fast code path to identify the corresponding lower and the upper bounds in the sorted min and max value arrays, given a range in the filter. The skipped pages are expressed as page ranges which are translated into row ranges later on.

A new query option minmax_filter_fast_code_path is added to control the work of the fast code path. This option can be assigned one of these three values ON (default), OFF, or VERIFICATION. The last value helps verify that the results from both the fast and the regular code path are the same.

The following example shows how min/max filtering on leading sort-by columns improves the performance of scan operators greatly.

```
select straight_join a.l_orderkey from
    simpflified_lineitem a join [SHUFFLE] tpch_parquet.lineitem b
```

```
    where a.l_orderkey = b.l_orderkey and b.l_receiptdate = "1998-12-31"
```

When this query is run on pages containing no more than 24000 rows, the results with different filtering options are as shown here:

- 84.62ms (page level filtering)
- 115.27ms (row group level filtering)
- 137.14ms (no filtering)

Support reading and writing Parquet bloom filters

Bloom filter is a performance optimization feature now available in Impala. This filter tells you, rapidly and memory-efficiently, whether the data you are looking for is present in a file.

Impala determines the appropriate conditions while the query is running. Impala can now read and write Parquet bloom filters. However, bloom filters can also provide false positives. If a bloom filter evaluates to:

- true: data might be present in the data file or might not be present.
- false: data is not present.

Currently, bloom filters are per column chunk entries which implies that you can skip entire row groups based on the filter. Writing a bloom filter is not useful for dictionary encoded columns, as all distinct values are included in the dictionary and the dictionary can give exact results in filtering with the predicate. If no value passes the predicate the whole row group can be skipped.

For example, if there is a predicate 'WHERE col = some_value' and some_value is not in the bloom filter the row group will be discarded.

Bloom filters support reading and writing columns with the following data types: integers, float, double, and Impala strings. Reading does not need any intervention from Impala, however, writing can be controlled by a new query option parquet_bloom_filter_write and the table property parquet.bloom.filter.columns.

The following table contains the list of the supported data types.

Parquet type	Impala type
INT32	TINYINT, SMALLINT, INT
INT64	BIGINT
FLOAT	FLOAT
DOUBLE	DOUBLE
BYTE_ARRAY	STRING

Query option

The query option for writing Parquet bloom filters (parquet_bloom_filter_write) accepts any of the following values:

- NEVER - never write Parquet bloom filters.
- IF_NO_DICT - write Parquet bloom filters if specified in the table properties AND if the row group is not fully dictionary encoded (the number of distinct values exceeds the maximum dictionary size); the row group may still be partially dictionary encoded, in which case the bloom filter contains all values from the whole row group, including those that are present in the dictionary.
- ALWAYS - always write Parquet bloom filters if specified in the table properties, even if the row group is fully dictionary encoded.

Table setting

The parquet.bloom.filter.columns table property is a comma separated list of 'col_name:bytes' pairs.

Where:

col_name is the name of the column for which a bloom filter should be written;

bytes represents the size (in bytes) of the bitset of the bloom filter, and is optional. If you do not provide the size, it will default to the maximal bloom filter size (ParquetBloomFilter::MAX_BYTES).

Example: “col1:1024,col2,col4:100"

Limitations

The following table contains the data types that are not supported currently. Support for these data types may be added in a future release.

Impala type	Reason for not supporting
VARCHAR(N)	truncation can change hash
CHAR(N)	padding / truncation can change hash
DECIMAL	multiple encodings supported
TIMESTAMP	multiple encodings supported, timezone conversion
DATE	not considered yet

Late Materialization of Columns

This is an optimization feature available in Impala and is implemented to optimize queries that reference a large number of columns and filter out a significant portion of the rows. This currently works only for Parquet tables.

When a query is processed, any fields referenced in the select list are decoded from file storage and materialized into row values in memory to be processed for joins, predicates, and so on. Prior to this optimization, the materialization is applied to all the row values, regardless of whether they are eliminated by predicates. However with late materialization, as the name implies, only fields that are referenced by predicates and rows that are not filtered out by predicates are fully materialized.

When a select * query is run over a 4 billion row table that returns a single row, it may take ~30 seconds to execute, however it may take less than 3 seconds if the query is replaced by select field1, field2 .... This is because the select * query is materializing all the fields for rows, regardless of whether they match or not. And in the case of the select f ield1, field2 query only the columns that are required for filtering the data need to be materialized.

Use the configuration parameter (query option) parquet_late_materialization_threshold to provide the minimum number of consecutive rows that are filtered out to avoid materialization. If set to less than 0, it disables the late materialization. In most cases the default setting of 20 will provide optimal performance.

Partitioning

Use partitioning to speed up queries that retrieve data based on values from one or more columns.

Partitioning is a technique for physically dividing the data during loading based on values from one or more columns. For example, with a school_records table partitioned on a year column, there is a separate data directory for each different year value, and all the data for that year is stored in a data file in that directory. A query that includes a WHERE condition such as YEAR=1966, YEAR IN (1989,1999), or YEAR BETWEEN 1984 AND 1989 can examine only the data files from the appropriate directory or directories, greatly reducing the amount of data to read and test.

Parquet is a popular format for partitioned Impala tables because it is well suited to handle huge data volumes.

You can add, drop, set the expected file format, or set the HDFS location of the data files for individual partitions within an Impala table.

When to Use Partitioned Tables

Partitioning is typically appropriate for:

- Tables that are very large, where reading the entire data set takes an impractical amount of time.
- Tables that are always or almost always queried with conditions on the partitioning columns. In our example of a table partitioned by year, SELECT COUNT(*) FROM school_records WHERE year = 1985 is efficient, only examining a small fraction of the data; but SELECT COUNT(*) FROM school_records has to process a separate data file for each year, resulting in more overall work than in an unpartitioned table. You would probably not partition this way if you frequently queried the table based on last name, student ID, and so on without testing the year.
- Columns that have reasonable cardinality (number of different values). If a column only has a small number of values, for example Male or Female, you do not gain much efficiency by eliminating only about 50% of the data to read for each query. If a column has only a few rows matching each value, the number of directories to process can become a limiting factor, and the data file in each directory could be too small to take advantage of the Hadoop mechanism for transmitting data in multi-megabyte blocks. For example, you might partition census data by year, store sales data by year and month, and web traffic data by year, month, and day. (Some users with high volumes of incoming data might even partition down to the individual hour and minute.)
- Data that already passes through an extract, transform, and load (ETL) pipeline. The values of the partitioning columns are stripped from the original data files and represented by directory names, so loading data into a partitioned table involves some sort of transformation or preprocessing.

SQL Statement for Partitioned Tables

In terms of Impala SQL syntax, partitioning affects these statements:

- CREATE TABLE: You specify a PARTITIONED BY clause when creating the table to identify names and data types of the partitioning columns. These columns are not included in the main list of columns for the table.
- CREATE TABLE AS SELECT: Use the PARTITIONED BY clause to create a partitioned table, copy data into it, and create new partitions based on the values in the inserted data.
- ALTER TABLE: Add or drop partitions, to work with different portions of a huge data set. You can designate the HDFS directory that holds the data files for a specific partition. With data partitioned by date values, you might “age out” data that is no longer relevant.

  If you are creating a partition for the first time and specifying its location, for maximum efficiency, use a single ALTER TABLE statement including both the ADD PARTITION and LOCATIONclauses, rather than separate statements with ADD PARTITION and SET LOCATION clauses.
- INSERT: When you insert data into a partitioned table, you identify the partitioning columns. One or more values from each inserted row are not stored in data files, but instead determine the directory where that row value is stored. You can also specify which partition to load a set of data into, with INSERT OVERWRITE statements; you can replace the contents of a specific partition but you cannot append data to a specific partition.

  By default, if an INSERT statement creates any new subdirectories underneath a partitioned table, those subdirectories are assigned default HDFS permissions for the impala user. To make each subdirectory have the same permissions as its parent directory in HDFS, specify the ##insert_inherit_permissions startup option for the Impala Daemon.
- Although the syntax of the SELECT statement is the same whether or not the table is partitioned, the way queries interact with partitioned tables can have a dramatic impact on performance and scalability. The mechanism that lets queries skip certain partitions during a query is known as partition pruning.
- SHOW PARTITIONS: Displays information about each partition in a table.

Static and Dynamic Partitioning Clauses

Specifying all the partition columns in a SQL statement is called *static partitioning*, because the statement affects a single predictable partition. For example, you use static partitioning with an ALTER TABLE statement that affects only one partition, or with an INSERT statement that inserts all values into the same partition:

```
INSERT INTO t1 PARTITION(x=10, y='a') SELECT c1 FROM some_other_table;
```

When you specify some partition key columns in an INSERT statement, but leave out the values, Impala determines which partition to insert. This technique is called *dynamic partitioning*.

```
INTSERT INTO t1 PARTITION(x, y='b') SELECT c1, c2 FROM some_other_table;
-- Create new partition if necessary based on variable year, month, and day;
 insert a single value.
INSERT INTO weather PARTITION (year, month, day) SELECT 'cloudy',2014,4,21;
-- Create new partition if necessary for specified year and month but variab
le day; insert a single value.
INSERT INTO weather PARTITION (year=2014, month=04, day) SELECT 'sunny',22;
```

The more key columns you specify in the PARTITION clause, the fewer columns you need in the SELECT list. The trailing columns in the SELECT list are substituted in order for the partition key columns with no specified value.

Refreshing a Single Partition

The REFRESH statement is typically used with partitioned tables when new data files are loaded into a partition by some non-Impala mechanism, such as a Hive or Spark job. The REFRESH statement makes Impala aware of the new data files so that they can be used in Impala queries. Because partitioned tables typically contain a high volume of data, the REFRESH operation for a full partitioned table can take significant time.

You can include a PARTITION (*PARTITION_SPEC*) clause in the REFRESH statement so that only a single partition is refreshed. For example, REFRESH big_table PARTITION (year=2017, month=9, day=30). The partition spec must include all the partition key columns.

Partition Key Columns

The columns you choose as the partition keys should be ones that are frequently used to filter query results in important, large-scale queries. Popular examples are some combination of year, month, and day when the data has associated time values, and geographic region when the data is associated with some place.

- For time-based data, split out the separate parts into their own columns, because Impala cannot partition based on a TIMESTAMP column.
- The data type of the partition columns does not have a significant effect on the storage required, because the values from those columns are not stored in the data files, rather they are represented as strings inside HDFS directory names.
- You can enable the OPTIMIZE_PARTITION_KEY_SCANS query option to speed up queries that only refer to partition key columns, such as SELECT MAX(year). This setting is not enabled by default because the query behavior is slightly different if the table contains partition directories without actual data inside.
- All the partition key columns must be scalar types.
- When Impala queries data stored in HDFS, it is most efficient to use multi-megabyte files to take advantage of the HDFS block size. For Parquet tables, the block size (and ideal size of the data files) is 256 MB.. Therefore, avoid specifying too many partition key columns, which could result in individual partitions containing only small amounts of data. For example, if you receive 1 GB of data per day, you might partition by year, month, and day; while if you receive 5 GB of data per minute, you might partition by year, month, day, hour, and minute. If you have data with a geographic component, you might partition based on postal code if you have many megabytes of data for each postal code, but if not, you might partition by some larger region such as city, state, or country. state

If you frequently run aggregate functions such as MIN(), MAX(), and COUNT(DISTINCT) on partition key columns, consider enabling the OPTIMIZE_PARTITION_KEY_SCANS query option, which optimizes such queries.

Setting Different File Formats for Partitions in a Table

Partitioned tables have the flexibility to use different file formats for different partitions. For example, if you originally received data in text format, then received new data in RCFile format, and eventually began receiving data in Parquet format, all that data could reside in the same table for queries. You just need to ensure that the table is structured so that the data files that use different file formats reside in separate partitions.

For example, here is how you might switch from text to Parquet data as you receive data for different years:

```
[localhost:21000] > CREATE TABLE census (name STRING) PARTITIONED BY (year S
MALLINT);
[localhost:21000] > ALTER TABLE census ADD PARTITION (year=2012); -- Text
format;
[localhost:21000] > ALTER TABLE census ADD PARTITION (year=2013); -- Text f
ormat switches to Parquet before data loaded;
[localhost:21000] > ALTER TABLE census PARTITION (year=2013) SET FILEFORMAT
 PARQUET;

[localhost:21000] > INSERT INTO census PARTITION (year=2012) VALUES ('Smith'
),('Jones'),('Lee'),('Singh');
[localhost:21000] > INSERT INTO census PARTITION (year=2013) VALUES ('Flo
res'),('Bogomolov'),('Cooper'),('Appiah');
```

At this point, the HDFS directory for year=2012 contains a text-format data file, while the HDFS directory for year =2013 contains a Parquet data file. As always, when loading non-trivial data, you would use INSERT ... SELECT or LOAD DATA to import data in large batches, rather than INSERT ... VALUES which produces small files that are inefficient for real-world queries.

For other file types that Impala cannot create natively, you can switch into Hive and issue the ALTER TABLE ... SET FILEFORMAT statements and INSERT or LOAD DATA statements there. After switching back to Impala, issue a REFRESH *TABLE_NAME* statement so that Impala recognizes any partitions or new data added through Hive.

Dropping Partitions

What happens to the data files when a partition is dropped depends on whether the partitioned table is designated as internal or external. For an internal (managed) table, the data files are deleted. For example, if data in the partitioned table is a copy of raw data files stored elsewhere, you might save disk space by dropping older partitions that are no longer required for reporting, knowing that the original data is still available if needed later. For an external table, the data files are left alone. For example, dropping a partition without deleting the associated files lets Impala consider a smaller set of partitions, improving query efficiency and reducing overhead for DDL operations on the table; if the data is needed again later, you can add the partition again.

Using Partitioning with Kudu Tables

Kudu tables use a more fine-grained partitioning scheme than tables containing HDFS data files. You specify a PART ITION BY clause with the CREATE TABLE statement to identify how to divide the values from the partition key columns.

Keeping Statistics Up to Date for Partitioned Tables

Because the COMPUTE STATS statement can be resource-intensive to run on a partitioned table as new partitions are added, Impala includes a variation of this statement that allows computing statistics on a per-partition basis such that stats can be incrementally updated when new partitions are added.

The COMPUTE INCREMENTAL STATS variation computes statistics only for partitions that were added or changed since the last COMPUTE INCREMENTAL STATS statement, rather than the entire table. It is typically used for tables where a full COMPUTE STATS operation takes too long to be practical each time a partition is added or dropped.

Partition Pruning for Queries

Partition pruning refers to the mechanism where a query can skip reading the data files corresponding to one or more partitions.

If you can arrange for queries to prune large numbers of unnecessary partitions from the query execution plan, the queries use fewer resources and are thus proportionally faster and more scalable.

For example, if a table is partitioned by columns YEAR, MONTH, and DAY, then WHERE clauses such as WHERE year = 2013, WHERE year < 2010, or WHERE year BETWEEN 1995 AND 1998 allow Impala to skip the data files in all partitions outside the specified range. Likewise, WHERE year = 2013 AND month BETWEEN 1 AND 3 could prune even more partitions, reading the data files for only a portion of one year.

If a view applies to a partitioned table, any partition pruning considers the clauses on both the original query and any additional WHERE predicates in the query that refers to the view.

In queries involving both analytic functions and partitioned tables, partition pruning only occurs for columns named in the PARTITION BY clause of the analytic function call. For example, if an analytic function query has a clause such as WHERE year=2016, the way to make the query prune all other YEAR partitions is to include PARTITION BY year in the analytic function call; for example, OVER (PARTITION BY year,*OTHER_COLUMNS OTHER_ANALYTIC_CLAUSES*).

Checking if Partition Pruning Happens for a Query

To check the effectiveness of partition pruning for a query, check the EXPLAIN output for the query before running it. For example, this example shows a table with 3 partitions, where the query only reads 1 of them. The notation #partitions=1/3 in the EXPLAIN plan confirms that Impala can do the appropriate partition pruning.

```
[localhost:21000] > INSERT INTO census PARTITION (year=2010) VALUES ('Smith'
),('Jones');
[localhost:21000] > INSERT INTO census PARTITION (year=2011) VALUES ('Smith
'),('Jones'),('Doe');
[localhost:21000] > INSERT INTO census PARTITION (year=2012) VALUES ('Smit
h'),('Doe');
[localhost:21000] > EXPLAIN select name from census where year=2010;
+----------------------------------------------------------------+
| Explain String                                                 |
+----------------------------------------------------------------+
| PLAN FRAGMENT 1                                                |
|   PARTITION: RANDOM                                            |
|                                                                |
|   STREAM DATA SINK                                             |
|     EXCHANGE ID: 1                                             |
|     UNPARTITIONED                                              |
|                                                                |
|   0:SCAN HDFS                                                  |
|      table=predicate_propagation.census #partitions=1/3 size=12B |
+----------------------------------------------------------------+
```

For a report of the volume of data that was actually read and processed at each stage of the query, check the output of the SUMMARY command immediately after running the query. For a more detailed analysis, look at the output of the PROFILE command; it includes this same summary report near the start of the profile output.

Predicate Propagation

Impala can do partition pruning in cases where the partition key column is not directly compared to a constant. Using the predicate propagation technique, Impala applies the transitive property to other parts of the WHERE clause.

In this example, the census table includes another column indicating when the data was collected, which happens in 10-year intervals. Even though the query does not compare the partition key column (YEAR) to a constant value, Impala can deduce that only the partition YEAR=2010 is required, and again only reads 1 out of 3 partitions.

```
[localhost:21000] > CREATE TABLE census (name STRING, census_year INT) PARTI
TIONED BY (year INT);
[localhost:21000] > INSERT INTO census PARTITION (year=2010) VALUES  ('Smi
th',2010),('Jones',2010);
[localhost:21000] > INSERT INTO census PARTITION (year=2011) VALUES  ('Smi
th',2020),('Jones',2020),('Doe',2020);
```

```
[localhost:21000] > INSERT INTO census PARTITION (year=2012) VALUES  ('Sm
ith',2020),('Doe',2020);
[localhost:21000] >
[localhost:21000] > EXPLAIN select name from census where year = census_year
 and census_year=2010;
+-----------------------------------------------------------------+
| Explain String                                                  |
+-----------------------------------------------------------------+
| PLAN FRAGMENT 1                                                 |
|   PARTITION: RANDOM                                             |
|                                                                 |
|   STREAM DATA SINK                                              |
|     EXCHANGE ID: 1                                              |
|     UNPARTITIONED                                               |
|                                                                 |
|   0:SCAN HDFS                                                   |
|      table=predicate_propagation.census #partitions=1/3 size=22B |
|      predicates: census_year = 2010, year = census_year         |
+-----------------------------------------------------------------+
```

Dynamic Partition Pruning

Impala supports two types of partition pruning.

- *Static partition pruning*: The conditions in the WHERE clause are analyzed to determine in advance which partitions can be safely skipped.
- *Dynamic partition pruning*: Information about the partitions is collected during the query execution, and Impala prunes unnecessary partitions. The information is not available in advance before runtime.

For example, if partition key columns are compared to literal values in a WHERE clause, Impala can perform static partition pruning during the planning phase to only read the relevant partitions:

```
-- The query only needs to read 3 partitions whose key values are known ahe
ad of time.
-- That's static partition pruning.
SELECT COUNT(*) FROM sales_table WHERE year IN (2005, 2010, 2015);
```

Dynamic partition pruning involves using information only available at run time, such as the result of a subquery. The following example shows a simple dynamic partition pruning.

```
CREATE TABLE yy (s STRING) PARTITIONED BY (year INT);
INSERT INTO yy PARTITION (year) VALUES ('1999', 1999), ('2000', 2000),
  ('2001', 2001), ('2010', 2010), ('2018', 2018);
COMPUTE STATS yy;

CREATE TABLE yy2 (s STRING, year INT);
INSERT INTO yy2 VALUES ('1999', 1999), ('2000', 2000), ('2001', 2001);
COMPUTE STATS yy2;
-- The following query reads an unknown number of partitions, whose key val
ues
-- are only known at run time. The runtime filters line shows the
-- information used in query fragment 02 to decide which partitions to skip.

EXPLAIN SELECT s FROM yy WHERE year IN (SELECT year FROM yy2);
+-----------------------------------------------------------------------+
| PLAN-ROOT SINK                                                        |
| |                                                                     |
| 04:EXCHANGE [UNPARTITIONED]                                           |
| |                                                                     |
| 02:HASH JOIN [LEFT SEMI JOIN, BROADCAST]                              |
```

```
| |  hash predicates: year = year                                          |
| |  runtime filters: RF000 <- year                                     |
| |                                                                        |
| |--03:EXCHANGE [BROADCAST]                                               |
| |  |                                                                     |
| |  01:SCAN HDFS [default.yy2]                                            |
| |     partitions=1/1 files=1 size=620B                                   |
| |                                                                        |
| 00:SCAN HDFS [default.yy]                                                |
|    partitions=5/5 files=5 size=1.71KB                                  |
|    runtime filters: RF000 -> year                                      |
+------------------------------------------------------------------------+

SELECT s FROM yy WHERE year IN (SELECT year FROM yy2); -- Returns 3 rows fro
m yy
PROFILE;
```

In the above example, Impala evaluates the subquery, sends the subquery results to all Impala nodes participating in the query, and then each `impalad` daemon uses the dynamic partition pruning optimization to read only the partitions with the relevant key values.

The output query plan from the EXPLAIN statement shows that runtime filters are enabled. The plan also shows that it expects to read all 5 partitions of the yy table, indicating that static partition pruning will not happen.

The Filter summary in the PROFILE output shows that the scan node filtered out based on a runtime filter of dynamic partition pruning.

```
Filter 0 (1.00 MB):
 - Files processed: 3
 - Files rejected: 1 (1)
 - Files total: 3 (3)
```

Dynamic partition pruning is especially effective for queries involving joins of several large partitioned tables. Evaluating the ON clauses of the join predicates might normally require reading data from all partitions of certain tables. If the WHERE clauses of the query refer to the partition key columns, Impala can now often skip reading many of the partitions while evaluating the ON clauses. The dynamic partition pruning optimization reduces the amount of I/O and the amount of intermediate data stored and transmitted across the network during the query.

Dynamic partition pruning is part of the runtime filtering feature, which applies to other kinds of queries in addition to queries against partitioned tables.

Understanding Performance using EXPLAIN Plan

To understand the high-level performance considerations for Impala queries, read the output of the EXPLAIN statement for the query. You can get the EXPLAIN plan without actually running the query itself.

The EXPLAIN statement gives you an outline of the logical steps that a query will perform, such as how the work will be distributed among the nodes and how intermediate results will be combined to produce the final result set. You can see these details before actually running the query. You can use this information to check that the query will not operate in some very unexpected or inefficient way.

Read the EXPLAIN plan from bottom to top:

- The last part of the plan shows the low-level details such as the expected amount of data that will be read, where you can judge the effectiveness of your partitioning strategy and estimate how long it will take to scan a table based on total data size and the size of the cluster.
- As you work your way up, next you see the operations that will be parallelized and performed on each Impala node.
- At the higher levels, you see how data flows when intermediate result sets are combined and transmitted from one node to another.

- The EXPLAIN_LEVEL query option lets you customize how much detail to show in the EXPLAIN plan depending on whether you are doing high-level or low-level tuning, dealing with logical or physical aspects of the query.

Read the EXPLAIN plan from bottom to top:

- The last part of the plan shows the low-level details such as the expected amount of data that will be read, where you can judge the effectiveness of your partitioning strategy and estimate how long it will take to scan a table based on total data size and the size of the cluster.
- As you work your way up, next you see the operations that will be parallelized and performed on each Impala node.
- At the higher levels, you see how data flows when intermediate result sets are combined and transmitted from one node to another.
- The EXPLAIN_LEVEL query option lets you customize how much detail to show in the EXPLAIN plan depending on whether you are doing high-level or low-level tuning, dealing with logical or physical aspects of the query.

The example below shows how the standard EXPLAIN output moves from the lowest (physical) level to the higher (logical) levels. The query begins by scanning a certain amount of data; each node performs an aggregation operation (evaluating COUNT(*)) on some subset of data that is local to that node; the intermediate results are transmitted back to the coordinator node (labelled here as the EXCHANGE node); lastly, the intermediate results are summed to display the final result.

```
[impalad-host:21000] > EXPLAIN select count(*) from customer_address;
+----------------------------------------------------------+
| Explain String                                           |
+----------------------------------------------------------+
| Estimated Per-Host Requirements: Memory=42.00MB VCores=1 |
|                                                          |
| 03:AGGREGATE [MERGE FINALIZE]                            |
| |  output: sum(count(*))                                 |
| |                                                        |
| 02:EXCHANGE [PARTITION=UNPARTITIONED]                    |
| |                                                        |
| 01:AGGREGATE                                             |
| |  output: count(*)                                      |
| |                                                        |
| 00:SCAN HDFS [default.customer_address]                  |
|    partitions=1/1 size=5.25MB                            |
+----------------------------------------------------------+
```

The EXPLAIN plan is also printed at the beginning of the query PROFILE report to provide convenience in examining both the logical and physical aspects of the query side-by-side.

The amount of detail displayed in the EXPLAIN output is controlled by the EXPLAIN_LEVEL query option. You customize how much detail to show in the EXPLAIN plan depending on whether you are doing high-level or low-level tuning, dealing with logical or physical aspects of the query. You typically increase this setting from standard to extended when double-checking the presence of table and column statistics during performance tuning, or when estimating query resource usage in conjunction with the resource management features.

Understanding Performance using SUMMARY Report

For an overview of the physical performance characteristics for a query, issue the SUMMARY command in impala-s hell immediately after executing a query. This condensed information shows which phases of execution took the most time, and how the estimates for memory usage and number of rows at each phase compare to the actual values.

Like the EXPLAIN plan, it is easy to see potential performance bottlenecks in the SUMMARY report. Like the PROF ILE output, the SUMMARY report is available after the query is run, and it displays actual timing statistics.

The SUMMARY report is also printed at the beginning of the query profile report described in the PROFILE output, for convenience in examining high-level and low-level aspects of the query side-by-side.

When the MT_DOP query option is set to a value larger than 0, the #Inst column in the output shows the number of fragment instances. Impala decomposes each query into smaller units of work that are distributed across the cluster, and these units are referred as fragments.

When the MT_DOP query option is set to 0, the #Inst column in the output shows the same value as the #Hosts column, since there is exactly one fragment for each host.

For example, here is a query involving an aggregate function on a single-node. The different stages of the query and their timings are shown (rolled up for all nodes), along with estimated and actual values used in planning the query. In this case, the AVG() function is computed for a subset of data on each node (stage 01) and then the aggregated results from all nodes are combined at the end (stage 03). You can see which stages took the most time, and whether any estimates were substantially different than the actual data distribution. (When examining the time values, be sure to consider the suffixes such as us for microseconds and ms for milliseconds, rather than just looking for the largest numbers.)

```
> SELECT AVG(ss_sales_price) FROM store_sales WHERE ss_coupon_amt = 0;
> SUMMARY;
+--------------+--------+--------+----------+----------+-------+------------
+----------+---------------+-----------------+
| Operator     | #Hosts | #Inst  | Avg Time | Max Time | #Rows | Est. #Rows
 | Peak Mem | Est. Peak Mem | Detail          |
+--------------+--------+--------+----------+----------+-------+------------
+----------+---------------+-----------------+
| 03:AGGREGATE | 1      | 1      | 1.03ms   | 1.03ms   | 1     | 1
 | 48.00 KB | -1 B          | MERGE FINALIZE  |
| 02:EXCHANGE  | 1      | 1      | 0ns      | 0ns      | 1     | 1
 | 0 B      | -1 B          | UNPARTITIONED   |
| 01:AGGREGATE | 1      | 1      |30.79ms   | 30.79ms  | 1     | 1
 | 80.00 KB | 10.00 MB      |                 |
| 00:SCAN HDFS | 1      | 1      | 5.45s    | 5.45s    | 2.21M | -1
| 64.05 MB | 432.00 MB     | tpc.store_sales |
+--------------+--------+--------+----------+----------+-------+------------
+----------+---------------+-----------------+
```

Notice how the longest initial phase of the query is measured in seconds (s), while later phases working on smaller intermediate results are measured in milliseconds (ms) or even nanoseconds (ns).

Understanding Performance using Query Profile

To understand the detailed performance characteristics for a query, issue the PROFILE command in `impala-shell` immediately after executing a query. This low-level information includes physical details about memory, CPU, I/O, and network usage, and thus is only available after the query is actually run.

The PROFILE command, available in `impala-shell`, produces a detailed low-level report showing how a query was executed. Unlike the EXPLAIN plan, this information is only available after the query has finished. It shows physical details such as the number of bytes read, maximum memory usage, and so on for each node. You can use this information to determine if the query is I/O-bound or CPU-bound, whether some network condition is imposing a bottleneck, whether a slowdown is affecting some nodes but not others, and to check that recommended configuration settings such as short-circuit local reads are in effect.

By default, time values in the profile output reflect the wall-clock time taken by an operation. For values denoting system time or user time, the measurement unit is reflected in the metric name, such as ScannerThreadsSysTime or ScannerThreadsUserTime. For example, a multi-threaded I/O operation might show a small figure for wall-clock time, while the corresponding system time is larger, representing the sum of the CPU time taken by each thread. Or a wall-clock time figure might be larger because it counts time spent waiting, while the corresponding system and user time figures only measure the time while the operation is actively using CPU cycles.

The EXPLAIN plan is also printed at the beginning of the query profile report, for convenience in examining both the logical and physical aspects of the query side-by-side. The EXPLAIN_LEVEL query option also controls the verbosity of the EXPLAIN output printed by the PROFILE command.

The Per Node Profiles section in the profile output includes the following metrics that can be controlled by the RESOURCE_TRACE_RATIO query option.

- CpuIoWaitPercentage
- CpuSysPercentage
- CpuUserPercentage
- HostDiskReadThroughput: All data read by the host as part of the execution of this query (spilling), by the HDFS data node, and by other processes running on the same system.
- HostDiskWriteThroughput: All data written by the host as part of the execution of this query (spilling), by the HDFS data node, and by other processes running on the same system.
- HostNetworkRx: All data received by the host as part of the execution of this query, other queries, and other processes running on the same system.
- HostNetworkTx: All data transmitted by the host as part of the execution of this query, other queries, and other processes running on the same system.

Until this release, the impala-shell 'profile' command only returns the profile for the most recent profile attempt. There was no option to get the original query profile (the profile of the first query attempt that failed) from the impala-shell. This release added support for returning both the original and retried profiles for a retried query.

impala-shell has been modified and the 'profile' command has a new set of options. The syntax is now:

```
PROFILE [ALL | LATEST | ORIGINAL]
```

If you specify 'ALL', both the latest and original profiles are printed. When you specify 'LATEST', only the latest profile is printed. If 'ORIGINAL' is specified, only the original profile is printed. The default behavior is equivalent to specifying 'LATEST'.

Note: Support for this has only been added to HS2 given that Beeswax is being deprecated soon. The new 'profile' options have no affect when the Beeswax protocol is used.

Planner changes for CPU usage

This release brings some changes to the query planner to improve parallel sizing and resource estimation. This is done for workload-aware autoscaling and will be available as query options. These additional query options are added for tuning purposes. This new functionality will allow additional customers to enable multi-threaded queries globally for improved performance.

Before you begin

- You must obtain the entitlement to use this feature.

Overview

Until this release, estimating the peak memory was the scaling factor in selecting an executor group to run large queries. This estimation was calculated during the query compilation time. When large queries were identified, they were enabled to run on larger executor groups mapped to different resource groups.

This release adds a new CPU costing algorithm as the scaling factor by establishing an infrastructure to allow the weighted total amount of data to process to be used as a new factor in the selection of an executor group. The new costing algorithm returns a number representing an ideal CPU core count required for a query to run efficiently. This number will be compared against the total CPU count of an Impala executor group to determine if it fits to run in that executor group or not. If the CPU requirement is higher than the max-query-cpu-core-per-node-limit, even in the largest exec group, the query may be rejected.

The following query options are added to control this CPU costing algorithm.

- COMPUTE_PROCESSING_COST

  Option to enable this CPU costing algorithm. Set MT_DOP > 0 to enable this query option.

  Note: If entitlement is enabled, COMPUTE_PROCESSING_COST will be set to true by default. If you use the REQUEST_POOL query option to direct a query to a specific pool or queue and you want to maintain the original MT_DOP behavior then you must explicitly set the flag COMPUTE_PROCESSING_COST to False.
- PROCESSING_COST_MIN_THREADS

  Option to control the minimum number of fragment instances (threads) that the costing algorithm is allowed to adjust. The costing algorithm is in charge of increasing the fragment's instance count beyond this minimum number through producer-consumer rate comparison. The maximum number of fragments is max between PROCESSING_COST_MIN_THREADS, MT_DOP, and number of cores per executor.

You can also use the following backend flags to tune the algorithm.

- query_cpu_count_divisor

  Use this flag to divide the CPU requirement of a query to fit the total available CPU in the executor group. For example, setting the value to 2 will fit the query with CPU requirement 2X to an executor group with total available CPU X. Note that setting with a fractional value less than 1 effectively multiplies the query CPU requirement. A valid value is > 0.0. The default value is 1.
- processing_cost_use_equal_expr_weight

  The default value of this flag is True. When the default value is retained, all expression evaluations are weighted equally to 1 during the plan node's processing cost calculation.

  Note: This flag is experimental and the default value should not be changed.
- min_processing_per_thread

  This flag defines the minimum processing load (in processing cost unit) that a fragment instance needs to work on before the planner considers increasing the instance count based on the processing cost rather than the MT_DOP setting. The decision is per fragment. If you set this flag to a higher number, it will reduce parallelism of a fragment (more workload per fragment), and when set to a lower number, it will increase parallelism (less workload per fragment). Actual parallelism might still be constrained by the total number of cores in the selected executor groups, MT_DOP, or the PROCESSING_COST_MIN_THREADS query option. The value of this flag must be a positive integer and it defaults to 10M.
- skip_resource_checking_on_last_executor_group_set

  This flag defines if the resource checking will be skipped on the last executor group set. If this backend flag is set to true, memory and cpu resource checking will be skipped when a query is being planned against the last (largest) executor group set. Setting true will ensure that query will always get admitted into the last executor group set if it does not fit in any other group set.

Planner changes to improve cardinality estimation

Changes have been implemented in the query planner to improve cardinality estimation, constituting a pivotal element of workload-aware autoscaling.

In earlier releases, Impala would generate a plan initially and then search for runtime filters based on the entire plan. In this release, selective runtime filters have been integrated. These filters serve to diminish the cardinality estimates of scan nodes and specific join nodes situated above them. This adjustment occurs after the generation of runtime filters and before the computation of resource requirements.

The cardinality reduction is implemented across all execution modes, but its effectiveness is currently contingent on the COMPUTE_PROCESSING_COST option being set to True. This condition exists because the optimal benefits of diminished scan cardinality are realized in scenarios where multiple executor group setups are employed. Reduced scan cardinality estimation has the potential to decrease Processing Costs, diminish scan fragment parallelism, and

enhance the likelihood of assigning queries to smaller executor group sets. It's worth noting that other execution modes, such as MT_DOP>0 or the legacy scanner parallelism mode, will not experience any changes in their execution plans.

Before you begin

- You must obtain the entitlement to use this feature.

Example

The following example demonstrates the execution of TPC-DS Query 3 at a scale factor of 1, utilizing 3 executor nodes, with the COMPUTE_PROCESSING_COST parameter set to True.

```
 Query:
   use tpcds_partitioned_parquet_snap;
   set COMPUTE_PROCESSING_COST=true;
   set PROCESSING_COST_MIN_THREADS=2;
   set MAX_FRAGMENT_INSTANCES_PER_NODE=16;

   -- TPCDS-Q3
   select dt.d_year,
   item.i_brand_id brand_id,
   item.i_brand brand,
   sum(ss_ext_sales_price) sum_agg
   from
   date_dim dt,
   store_sales,
   item
   where
   dt.d_date_sk = store_sales.ss_sold_date_sk
   and store_sales.ss_item_sk = item.i_item_sk
   and item.i_manufact_id = 436
   and dt.d_moy = 12
   group by
   dt.d_year,
   item.i_brand,
   item.i_brand_id
   order by
   dt.d_year,
   sum_agg desc,
   brand_id
   limit 100

Plan:
Max Per-Host Resource Reservation: Memory=33.25MB Threads=14
Per-Host Resource Estimates: Memory=169MB
Analyzed query: SELECT dt.d_year, item.i_brand_id brand_id, item.i_brand
brand,
sum(ss_ext_sales_price) sum_agg FROM tpcds_partitioned_parquet_snap.date_
dim dt,
tpcds_partitioned_parquet_snap.store_sales, tpcds_partitioned_parquet_snap.i
tem
WHERE dt.d_date_sk = store_sales.ss_sold_date_sk AND store_sales.ss_item_sk
=
item.i_item_sk AND item.i_manufact_id = CAST(436 AS INT) AND dt.d_moy = CA
ST(12
AS INT) GROUP BY dt.d_year, item.i_brand, item.i_brand_id ORDER BY dt.d_year
ASC, sum_agg DESC, brand_id ASC LIMIT CAST(100 AS TINYINT)

F05:PLAN FRAGMENT [UNPARTITIONED] hosts=1 instances=1
|  Per-Instance Resources: mem-estimate=4.03MB mem-reservation=4.00MB threa
d-reservation=1
|  max-parallelism=1 segment-costs=[406] cpu-comparison-result=18 [max(1 (se
lf) vs 18 (sum children))]
```

```
PLAN-ROOT SINK
|  output exprs: dt.d_year, item.i_brand_id, item.i_brand, sum(ss_ext_sale
s_price)
|  mem-estimate=4.00MB mem-reservation=4.00MB spill-buffer=2.00MB thread-re
servation=0 cost=400
|
12:MERGING-EXCHANGE [UNPARTITIONED]
|  order by: dt.d_year ASC, sum(ss_ext_sales_price) DESC, item.i_brand_id
 ASC
|  limit: 100
|  mem-estimate=34.63KB mem-reservation=0B thread-reservation=0
|  tuple-ids=4 row-size=52B cardinality=100 cost=6
|  in pipelines: 06(GETNEXT)
|
F04:PLAN FRAGMENT [HASH(dt.d_year,item.i_brand,item.i_brand_id)] hosts=3 in
stances=6 (adjusted from 48)
Per-Instance Resources: mem-estimate=10.38MB mem-reservation=1.94MB thread-
reservation=1
max-parallelism=6 segment-costs=[12331, 300, 6] cpu-comparison-result=18 [
max(6 (self) vs 18 (sum children))]
06:TOP-N [LIMIT=100]
|  order by: dt.d_year ASC, sum(ss_ext_sales_price) DESC, item.i_brand_id A
SC
|  mem-estimate=5.10KB mem-reservation=0B thread-reservation=0
|  tuple-ids=4 row-size=52B cardinality=100 cost=300
|  in pipelines: 06(GETNEXT), 11(OPEN)
|
11:AGGREGATE [FINALIZE]
|  output: sum:merge(ss_ext_sales_price)
|  group by: dt.d_year, item.i_brand, item.i_brand_id
|  mem-estimate=10.00MB mem-reservation=1.94MB spill-buffer=64.00KB thread-
reservation=0
|  tuple-ids=3 row-size=52B cardinality=3.04K cost=12164
|  in pipelines: 11(GETNEXT), 01(OPEN)
|
10:EXCHANGE [HASH(dt.d_year,item.i_brand,item.i_brand_id)]
|  mem-estimate=388.85KB mem-reservation=0B thread-reservation=0
|  tuple-ids=3 row-size=52B cardinality=3.04K cost=167
|  in pipelines: 01(GETNEXT)
|
F03:PLAN FRAGMENT [HASH(store_sales.ss_sold_date_sk)] hosts=3 instances=6 (a
djusted from 48)
Per-Instance Resources: mem-estimate=12.17MB mem-reservation=2.00MB thread-
reservation=1
max-parallelism=6 segment-costs=[15408, 167] cpu-comparison-result=18 [max
(18 (self) vs 11 (sum children))]
05:AGGREGATE [STREAMING]
|  output: sum(ss_ext_sales_price)
|  group by: dt.d_year, item.i_brand, item.i_brand_id
|  mem-estimate=10.00MB mem-reservation=2.00MB spill-buffer=64.00KB thread-r
eservation=0
|  tuple-ids=3 row-size=52B cardinality=3.04K cost=12164
|  in pipelines: 01(GETNEXT)
|
04:HASH JOIN [INNER JOIN, PARTITIONED]
|  hash-table-id=00
|  hash predicates: store_sales.ss_sold_date_sk = dt.d_date_sk
|  fk/pk conjuncts: store_sales.ss_sold_date_sk = dt.d_date_sk
|  mem-estimate=0B mem-reservation=0B spill-buffer=64.00KB thread-reservat
ion=0
|  tuple-ids=1,2,0 row-size=72B cardinality=3.04K cost=3041
|  in pipelines: 01(GETNEXT), 00(OPEN)
|
```

```
|--F06:PLAN FRAGMENT [HASH(store_sales.ss_sold_date_sk)] hosts=3 instance
s=6 (adjusted from 48)
|  |  Per-Instance Resources: mem-estimate=3.02MB mem-reservation=2.94MB
thread-reservation=1 runtime-filters-memory=1.00MB
|  |  max-parallelism=6 segment-costs=[6183]
|  JOIN BUILD
|  |  join-table-id=00 plan-id=01 cohort-id=01
|  |  build expressions: dt.d_date_sk
|  |  runtime filters: RF000[bloom] <- dt.d_date_sk, RF001[min_max] <- dt
.d_date_sk
|  |  mem-estimate=1.94MB mem-reservation=1.94MB spill-buffer=64.00KB thr
ead-reservation=0 cost=6087
|  |
|  09:EXCHANGE [HASH(dt.d_date_sk)]
|  |  mem-estimate=87.33KB mem-reservation=0B thread-reservation=0
|  |  tuple-ids=0 row-size=12B cardinality=6.09K cost=96
|  |  in pipelines: 00(GETNEXT)
|  |
|  F02:PLAN FRAGMENT [RANDOM] hosts=1 instances=1
|  Per-Instance Resources: mem-estimate=16.38MB mem-reservation=512.00KB thr
ead-reservation=1
|  max-parallelism=1 segment-costs=[124002]
|  00:SCAN HDFS [tpcds_partitioned_parquet_snap.date_dim dt, RANDOM]
|     HDFS partitions=1/1 files=1 size=2.15MB
|     predicates: dt.d_moy = CAST(12 AS INT)
|     stored statistics:
|       table: rows=73.05K size=2.15MB
|       columns: all
|     extrapolated-rows=disabled max-scan-range-rows=73.05K
|     parquet statistics predicates: dt.d_moy = CAST(12 AS INT)
|     parquet dictionary predicates: dt.d_moy = CAST(12 AS INT)
|     mem-estimate=16.00MB mem-reservation=512.00KB thread-reservation=0
|     tuple-ids=0 row-size=12B cardinality=6.09K cost=123906
|     in pipelines: 00(GETNEXT)
|
08:EXCHANGE [HASH(store_sales.ss_sold_date_sk)]
|  mem-estimate=877.96KB mem-reservation=0B thread-reservation=0
|  tuple-ids=1,2 row-size=60B cardinality=3.04K cost=203
|  in pipelines: 01(GETNEXT)
|
F00:PLAN FRAGMENT [RANDOM] hosts=3 instances=12 (adjusted from 48)
Per-Host Shared Resources: mem-estimate=2.00MB mem-reservation=2.00MB thr
ead-reservation=0 runtime-filters-memory=2.00MB
Per-Instance Resources: mem-estimate=17.60MB mem-reservation=1.00MB thread-r
eservation=1
max-parallelism=12 segment-costs=[91203296]
03:HASH JOIN [INNER JOIN, BROADCAST]
|  hash-table-id=01
|  hash predicates: store_sales.ss_item_sk = item.i_item_sk
|  fk/pk conjuncts: store_sales.ss_item_sk = item.i_item_sk
|  mem-estimate=0B mem-reservation=0B spill-buffer=64.00KB thread-reservat
ion=0
|  tuple-ids=1,2 row-size=60B cardinality=3.04K cost=3045
|  in pipelines: 01(GETNEXT), 02(OPEN)
|
|--F07:PLAN FRAGMENT [RANDOM] hosts=3 instances=3
|  |  Per-Instance Resources: mem-estimate=8.77MB mem-reservation=8.75MB
thread-reservation=1 runtime-filters-memory=1.00MB
|  |  max-parallelism=3 segment-costs=[22]
|  JOIN BUILD
|  |  join-table-id=01 plan-id=02 cohort-id=01
|  |  build expressions: item.i_item_sk
|  |  runtime filters: RF002[bloom] <- item.i_item_sk
```

```
|   |  mem-estimate=7.75MB mem-reservation=7.75MB spill-buffer=64.00KB threa
d-reservation=0 cost=19
|   |
|   07:EXCHANGE [BROADCAST]
|   |  mem-estimate=16.00KB mem-reservation=0B thread-reservation=0
|   |  tuple-ids=2 row-size=44B cardinality=19 cost=3
|   |  in pipelines: 02(GETNEXT)
|   |
|   F01:PLAN FRAGMENT [RANDOM] hosts=1 instances=1
|   Per-Instance Resources: mem-estimate=16.19MB mem-reservation=256.00KB
thread-reservation=1
|   max-parallelism=1 segment-costs=[68778]
|   02:SCAN HDFS [tpcds_partitioned_parquet_snap.item, RANDOM]
|      HDFS partitions=1/1 files=1 size=1.73MB
|      predicates: item.i_manufact_id = CAST(436 AS INT)
|      stored statistics:
|        table: rows=18.00K size=1.73MB
|        columns: all
|      extrapolated-rows=disabled max-scan-range-rows=18.00K
|      parquet statistics predicates: item.i_manufact_id = CAST(436 AS INT)
|      parquet dictionary predicates: item.i_manufact_id = CAST(436 AS INT)
|      mem-estimate=16.00MB mem-reservation=256.00KB thread-reservation=0
|      tuple-ids=2 row-size=44B cardinality=19 cost=68777
|      in pipelines: 02(GETNEXT)
|
01:SCAN HDFS [tpcds_partitioned_parquet_snap.store_sales, RANDOM]
   HDFS partitions=1824/1824 files=1824 size=199.36MB
   runtime filters: RF001[min_max] -> store_sales.ss_sold_date_sk, RF000[
bloom] -> store_sales.ss_sold_date_sk, RF002[bloom] -> store_sales.ss_item_s
k
   stored statistics:
     table: rows=2.88M size=199.36MB
     partitions: 1824/1824 rows=2.88M
     columns: all
   extrapolated-rows=disabled max-scan-range-rows=130.09K
   mem-estimate=16.00MB mem-reservation=1.00MB thread-reservation=0
   tuple-ids=1 row-size=16B cardinality=3.04K(filtered from 2.88M) cost=9
1200048
   in pipelines: 01(GETNEXT)
```

Caching Codegen Functions

In Impala, "codegen" refers to code generation, utilizing query-specific information to generate specialized machine code for each query.

When executing a standard query, the query optimizer generates an optimized query plan, passing it to the executor for processing. The codegen capability converts query-specific information into machine code for faster execution. However, for sub-second queries, codegen may consume excessive time; for instance, the compilation time for codegen could take 300ms for a sub-second query. To mitigate this, Cloudera introduces support for memory caches for codegen functions, enhancing the performance of sub-second queries. Users can specify the total cache size for all codegen functions. The configuration parameter 'codegen_cache_capacity' in Impala daemons controls the total codegen cache size. For example, when set to "1GB," it restricts the codegen cache size for the particular Impala daemon to one gigabyte.

The cache is a singleton instance for each daemon, housing multiple cache entries. Each cache entry operates at the fragment level and retains all codegen functions associated with a fragment. Upon reaching maximum capacity, the cache employs an LRU (Least Recently Used) eviction policy. Storing codegen functions in the cache enables their reuse when appropriate, reducing the need for repetitive codegen compilation, which may otherwise consume hundreds of milliseconds.

The LLVM module bitcode serves as the cache key, generated prior to LLVM module optimization and final compilation. When codegen_cache_mode is set to NORMAL, the complete bitcode string is stored as the key by default. Conversely, if codegen_cache_mode is set to OPTIMAL, only a key containing the hash code derived from the NORMAL mode key is stored, reducing memory consumption.

The memory allocated for the codegen cache operates independently of the admission memory reserved for query execution. While the codegen cache memory is initially limited to a certain percentage of the total process memory, adjustments can be made during startup if it exceeds that percentage.

You can deactivate the codegen capability by setting the query option disable_codegen_cache to true or by setting the flag file option 'codegen_cache_capacity' to 0. It is important to note that even if disable_codegen_cache is set to 'false', caching will still be disabled if you set codegen_cache_capacity to 0.

Scalability Considerations

The size of your cluster and the volume of data influences query performance. Typically, adding more cluster capacity reduces problems due to memory limits or disk throughput. On the other hand, larger clusters are more likely to have other kinds of scalability issues, such as a single slow node that causes performance problems for queries.

Impact of Many Tables or Partitions on Impala Catalog Performance and Memory Usage

Because Hadoop I/O is optimized for reading and writing large files, Impala is optimized for tables containing relatively few, large data files. Schemas containing thousands of tables, or tables containing thousands of partitions, can encounter performance issues during startup or during DDL operations such as ALTER TABLE statements.

Scalability Considerations for the Impala StateStore

If it takes a very long time for a cluster to start up with the message, This Impala daemon is not ready to accept user requests, the StateStore might be taking too long to send the entire catalog topic to the cluster. In this case, consider setting the Load Catalog in Background field to false in your Catalog Service configuration. This setting stops the StateStore from loading the entire catalog into memory at cluster startup. Instead, metadata for each table is loaded when the table is accessed for the first time.

Effect of Buffer Pool on Memory Usage

Most of the memory needed is reserved at the beginning of the query, avoiding cases where a query might run for a long time before failing with an out-of-memory error. The actual memory estimates and memory buffers are typically smaller than before, so that more queries can run concurrently or process larger volumes of data than previously.

Increase the MAX_ROW_SIZE query option setting when querying tables with columns containing long strings, many columns, or other combinations of factors that produce very large rows. If Impala encounters rows that are too large to process with the default query option settings, the query fails with an error message suggesting to increase the MAX_ROW_SIZE setting.

SQL Operations that Spill to Disk

Certain memory-intensive operations write temporary data to disk (known as *spilling* to disk) when Impala is close to exceeding its memory limit on a particular host.

What kinds of queries might spill to disk:

Several SQL clauses and constructs require memory allocations that could trigger spilling to disk:

- when a query uses a GROUP BY clause for columns with millions or billions of distinct values, Impala keeps a similar number of temporary results in memory, to accumulate the aggregate results for each value in the group.
- When large tables are joined together, Impala keeps the values of the join columns from one table in memory, to compare them to incoming values from the other table.
- When a large result set is sorted by the ORDER BY clause, each node sorts its portion of the result set in memory.

- The DISTINCT and UNION operators build in-memory data structures to represent all values found so far, to eliminate duplicates as the query progresses.

When the spill-to-disk feature is activated for a join node within a query, Impala does not produce any runtime filters for that join operation on that host. Other join nodes within the query are not affected.

The amount data depends on the portion of the data being handled by that host, and thus the operator may end up consuming different amounts of memory on different hosts.

Memory usage for SQL operators:

The memory required to spill to disk is reserved up front, and you can examine it in the EXPLAIN plan when the EXPLAIN_LEVEL query option is set to 2 or higher.

If an operator accumulates more data than can fit in the reserved memory, it can either reserve more memory to continue processing data in memory or start spilling data to temporary scratch files on disk. Thus, operators with spill-to-disk support can adapt to different memory constraints by using however much memory is available to speed up execution, yet tolerate low memory conditions by spilling data to disk.

The amount of data depends on the portion of the data being handled by that host, and thus the operator may end up consuming different amounts of memory on different hosts.

Avoiding queries that spill to disk:

Because the extra I/O can impose significant performance overhead on these types of queries, try to avoid this situation by using the following steps:

1. Detect how often queries spill to disk, and how much temporary data is written. Refer to the following sources:

- The output of the PROFILE command in the `impala-shell` interpreter. This data shows the memory usage for each host and in total across the cluster. The WriteIoBytes counter reports how much data was written to disk for each operator during the query.
- In the Queries tab in the Impala debug web user interface, select the query to examine and click the corresponding Profile link. This data breaks down the memory usage for a single host within the cluster, the host whose web interface you are connected to.

2. Use one or more techniques to reduce the possibility of the queries spilling to disk:

- Increase the Impala memory limit if practical. For example, using the SET MEM_LIMIT SQL statement, increase the available memory by more than the amount of temporary data written to disk on a particular node.
- Increase the number of nodes in the cluster, to increase the aggregate memory available to Impala and reduce the amount of memory required on each node.
- If the memory pressure is due to running many concurrent queries rather than a few memory-intensive ones, consider using the Impala admission control feature to lower the limit on the number of concurrent queries. By spacing out the most resource-intensive queries, you can avoid spikes in memory usage and improve overall response times.
- Tune the queries with the highest memory requirements, using one or more of the following techniques:
  - Run the COMPUTE STATS statement for all tables involved in large-scale joins and aggregation queries.
  - Minimize your use of STRING columns in join columns. Prefer numeric values instead.
  - Examine the EXPLAIN plan to understand the execution strategy being used for the most resource-intensive queries.
  - If Impala still chooses a suboptimal execution strategy even with statistics available, or if it is impractical to keep the statistics up to date for huge or rapidly changing tables, add hints to the most resource-intensive queries to select the right execution strategy.
- If your queries experience substantial performance overhead due to spilling, enable the DISABLE_UNSAFE_SPILLS query option. This option prevents queries whose memory usage is likely to be exorbitant from spilling to disk. As you tune problematic queries using the preceding steps, fewer and fewer will be cancelled by this option setting.

When to use DISABLE_UNSAFE_SPILLS:

The DISABLE_UNSAFE_SPILLS query option is suitable for an environment with ad hoc queries whose performance characteristics and memory usage are not known in advance. It prevents “worst-case scenario” queries that use large amounts of memory unnecessarily. Thus, you might turn this option on within a session while developing new SQL code, even though it is turned off for existing applications.

Organizations where table and column statistics are generally up-to-date might leave this option turned on all the time, again to avoid worst-case scenarios for untested queries or if a problem in the ETL pipeline results in a table with no statistics. Turning on DISABLE_UNSAFE_SPILLS lets you “fail fast” in this case and immediately gather statistics or tune the problematic queries.

Some organizations might leave this option turned off. For example, you might have tables large enough that the COMPUTE STATS takes substantial time to run, making it impractical to re-run after loading new data. If you have examined the EXPLAIN plans of your queries and know that they are operating efficiently, you might leave DISABLE_UNSAFE_SPILLS turned off. In that case, you know that any queries that spill will not go overboard with their memory consumption.

Limits on Query Size and Complexity

There are hard-coded limits on the maximum size and complexity of queries. Currently, the maximum number of expressions in a query is 2000. You might exceed the limits with large or deeply nested queries produced by business intelligence tools or other query generators.

If you have the ability to customize such queries or the query generation logic that produces them, replace sequences of repetitive expressions with single operators such as IN or BETWEEN that can represent multiple values or ranges. For example, instead of a large number of OR clauses:

```
WHERE val = 1 OR val = 2 OR val = 6 OR val = 100 ...
```

use a single IN clause:

```
WHERE val IN (1,2,6,100,...)
```

Scalability Considerations for Table Layout

Due to the overhead of retrieving and updating table metadata in the Metastore database, try to limit the number of columns in a table to a maximum of approximately 2000. Although Impala can handle wider tables than this, the Metastore overhead can become significant, leading to query performance that is slower than expected based on the actual data volume.

To minimize overhead related to the Metastore database and Impala query planning, try to limit the number of partitions for any partitioned table to a few tens of thousands.

If the volume of data within a table makes it impractical to run exploratory queries, consider using the TABLESAMPLE clause to limit query processing to only a percentage of data within the table. This technique reduces the overhead for query startup, I/O to read the data, and the amount of network, CPU, and memory needed to process intermediate results during the query.

Kerberos-Related Network Overhead for Large Clusters

When Impala starts up, or after each kinit refresh, Impala sends a number of simultaneous requests to the KDC. For a cluster with 100 hosts, the KDC might be able to process all the requests within roughly 5 seconds. For a cluster with 1000 hosts, the time to process the requests would be roughly 500 seconds. Impala also makes a number of DNS requests at the same time as these Kerberos-related requests.

While these authentication requests are being processed, any submitted Impala queries will fail. During this period, the KDC and DNS may be slow to respond to requests from components other than Impala, so other secure services might be affected temporarily.

Scalability Considerations for File Handle Caching

One scalability aspect that affects heavily loaded clusters is the load on the metadata layer from looking up the details as each file is opened. On HDFS, that can lead to increased load on the NameNode, and on object stores such as S3 or ABFS, this can lead to an excessive number of metadata requests. For example, a query that does a full table scan on a partitioned table may need to read thousands of partitions, each partition containing multiple data files. Accessing each column of a Parquet file also involves a separate “open” call, further increasing the load on the NameNode. High NameNode overhead can add startup time (that is, increase latency) to Impala queries, and reduce overall throughput for non-Impala workloads that also require accessing HDFS files.

You can reduce the number of calls made to your file system's metadata layer by enabling the file handle caching feature. Data files that are accessed by different queries, or even multiple times within the same query, can be accessed without a new “open” call and without fetching the file details multiple times.

Impala supports file handle caching for the following file systems:

- HDFS

  The cache_remote_file_handles flag controls local and remote file handle caching for an impalad. It is recommended that you use the default value of true as this caching prevents your NameNode from overloading when your cluster has many remote HDFS reads.
- S3

  The cache_s3_file_handles impalad flag controls the S3 file handle caching. The feature is enabled by default with the flag set to true.
- ABFS

  The cache_abfs_file_handles impalad flag controls the ABFS file handle caching. The feature is enabled by default with the flag set to true.

The feature is enabled by default with 20,000 file handles to be cached. To change the value, set the configuration option Maximum Cached File Handles (max_cached_file_handles) to a non-zero value for each Impala daemon (impalad). From the initial default value of 20000, adjust upward if NameNode request load is still significant, or downward if it is more important to reduce the extra memory usage on each host. Each cache entry consumes 6 KB, meaning that caching 20,000 file handles requires up to 120 MB on each Impala executor. The exact memory usage varies depending on how many file handles have actually been cached; memory is freed as file handles are evicted from the cache.

If a manual operation moves a file to the trashcan while the file handle is cached, Impala still accesses the contents of that file. This is a change from prior behavior. Previously, accessing a file that was in the trashcan would cause an error. This behavior only applies to non-Impala methods of removing files, not the Impala mechanisms such as TRUN CATE TABLE or DROP TABLE.

If files are removed, replaced, or appended by operations outside of Impala, the way to bring the file information up to date is to run the REFRESH statement on the table.

File handle cache entries are evicted as the cache fills up, or based on a timeout period when they have not been accessed for some time.

To evaluate the effectiveness of file handle caching for a particular workload, issue the PROFILE statement in `impala-shell` or examine query profiles in the Impala Web UI. Look for the ratio of CachedFileHandlesHitCount (ideally, should be high) to CachedFileHandlesMissCount (ideally, should be low). Before starting any evaluation, run several representative queries to “warm up” the cache because the first time each data file is accessed is always recorded as a cache miss.

To see metrics about file handle caching for each `impalad` instance, examine the following fields on the /metrics page in the Impala Web UI:

- impala-server.io.mgr.cached-file-handles-miss-count
- impala-server.io.mgr.num-cached-file-handles

Scaling Limits and Guidelines

Consider and respect the scalability limitations provided in this topic in order to achieve optimal scalability and performance in Impala. For example, while you might be able to create a table with 2000 columns, you will experience performance problems while querying the table. This topic does not cover functional limitations in Impala.

Unless noted otherwise, the limits listed in this topic were tested and certified.

The limits noted as "generally safe" are not certified, but recommended as generally safe. A safe range is not a hard limit as unforeseen errors or troubles in your particular environment can affect the range.

Deployment Limits

- Number of Impalad Executors: 150 nodes
- Number of Impalad Coordinators: 1 coordinator for at most every 50 executors

Data Storage Limits

There are no hard limits for the following, but you will experience gradual performance degradation as you increase these numbers.

- Number of databases
- Number of tables - total, per database
- Number of partitions - total, per table
- Number of files - total, per table, per table per partition
- Number of views - total, per database
- Number of user-defined functions - total, per database
- Parquet
  - Number of columns per row group
  - Number of row groups per block
  - Number of HDFS blocks per file

Schema Design Limits

- Number of columns
  - 300 for Kudu tables
  - 1000 for other types of tables

Security Limits

- Number of roles: 10,000

Query Limits - Compile Time

- Maximum number of columns in a query, included in a SELECT list, INSERT, and in an expression: no limit
- Number of tables referenced: no limit
- Number of plan nodes: no limit
- Number of plan fragments: no limit
- Depth of expression tree: 1000 hard limit
- Width of expression tree: 10,000 hard limit

- Codegen: Very deeply nested expressions within queries can exceed internal Impala limits, leading to excessive memory usage. Setting the query option disable_codegen=true may reduce the impact, at a cost of longer query runtime.

Hadoop File Formats Support

Impala supports a number of file formats used in Apache Hadoop.

Impala can load and query data files produced by other Hadoop components such as Spark, and data files produced by Impala can be used by other components also. The following sections discuss the procedures, limitations, and performance considerations for using each file format with Impala.

The file format used for an Impala table has significant performance consequences. Some file formats include compression support that affects the size of data on the disk and, consequently, the amount of I/O and CPU resources required to deserialize data. The amounts of I/O and CPU resources required can be a limiting factor in query performance since querying often begins with moving and decompressing data. To reduce the potential impact of this part of the process, data is often compressed. By compressing data, a smaller total number of bytes are transferred from disk to memory. This reduces the amount of time taken to transfer the data, but a tradeoff occurs when the CPU decompresses the content.

For the file formats that Impala cannot write to, create the table from within Impala whenever possible and insert data using another component such as Hive or Spark. See the table below for specific file formats.

The following table lists the file formats that Impala supports.

File Type	Format	Compression Codecs	Impala Can CREATE?	Impala Can INSERT?
Parquet	Structured	Snappy, gzip, zstd, LZ4; currently Snappy by default	Yes.	Yes. CREATE TABLE, INSERT, LOAD DATA, and query.
ORC	Structured	gzip, Snappy, LZ4; currently gzip by default	Yes. By default, ORC reads are enabled in Impala 3.4.0 and higher. To disable, set--enable_orc_scanner to false when starting the cluster.	No. Import data by using LOAD DATA on data files already in the right format, or use INSERT in Hive followed by REFRESH *TABLE_NAME* in Impala.
Text	Unstructured	bzip2, deflate, gzip, Snappy,zstd	Yes. CREATE TABLE with no STORED AS clause. The default file format is uncompressed text, with values separated by ASCII 0x01 characters (typically represented as Ctrl-A).	Yes if uncompressed. CREATE TABLE, INSERT, LOAD DATA, and query. No if compressed. If other kinds of compression are used, you must load data through LOAD DATA, Hive, or manually in HDFS.
Avro	Structured	Snappy, gzip, deflate	Yes.	No. Import data by using LOAD DATA on data files already in the right format, or use INSERT in Hive followed by REFRESH *TABLE_NAME* in Impala.
RCFile	Structured	Snappy, gzip, deflate, bzip2	Yes.	No. Import data by using LOAD DATA on data files already in the right format, or use INSERT in Hive followed by REFRESH *TABLE_NAME* in Impala.
SequenceFile	Structured	Snappy, gzip, deflate, bzip2	Yes.	No. Import data by using LOAD DATA on data files already in the right format, or use INSERT in Hive followed by REFRESH *TABLE_NAME* in Impala.

Impala supports the following compression codecs:

Snappy

Recommended for its effective balance between compression ratio and decompression speed. Snappy compression is very fast, but gzip provides greater space savings. Supported for text, RC, Sequence, and Avro files in Impala 2.0 and higher.

Gzip

Recommended when achieving the highest level of compression (and therefore greatest disk-space savings) is desired. Supported for text, RC, Sequence and Avro files in Impala 2.0 and higher.

Deflate

Supported for AVRO, RC, Sequence, and text files.

Bzip2

Supported for text, RC, and Sequence files in Impala 2.0 and higher.

Zstd

For Parquet and text files only.

Lz4

For Parquet files only.

Choosing the File Format for a Table

Different file formats and compression codecs work better for different data sets. Choosing the proper format for your data can yield performance improvements. Use the following considerations to decide which combination of file format and compression to use for a particular table.

- If you are working with existing files that are already in a supported file format, use the same format for the Impala table if performance is acceptable. If the original format does not yield acceptable query performance or resource usage, consider creating a new Impala table with different file format or compression characteristics, and doing a one-time conversion by rewriting the data to the new table.
- Text files are convenient to produce through many different tools and are human-readable for ease of verification and debugging. Those characteristics are why text is the default format for an Impala CREATE TABLE statement. However, when performance and resource usage are the primary considerations, use one of the structured file formats that include metadata and built-in compression.

  A typical workflow might involve bringing data into an Impala table by copying CSV or TSV files into the appropriate data directory, and then using the INSERT ... SELECT syntax to rewrite the data into a table using a different, more compact file format.

Using Text Data Files

Impala supports using text files as the storage format for input and output. Text files are a convenient format to use for interchange with other applications or scripts that produce or read delimited text files, such as CSV or TSV with commas or tabs for delimiters.

Text files are flexible in their column definitions. For example, a text file could have more fields than the Impala table, and those extra fields are ignored during queries. Or it could have fewer fields than the Impala table, and those missing fields are treated as NULL values in queries.

You could have fields that were treated as numbers or timestamps in a table, then use ALTER TABLE ... REPLACE COLUMNS to switch them to strings, or the reverse.

Creating Text Tables

You can create tables with specific separator characters to import text files in familiar formats such as CSV, TSV, or pipe-separated with the FIELDS TERMINATED BY clause preceded by the ROW FORMAT DELIMITED clause. For example:

```
CREATE TABLE tsv(id INT, s STRING, n INT, t TIMESTAMP, b BOOLEAN)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;
```

You can specify a delimiter character '\0' to use the ASCII 0 (nul) character for text tables.

You can also use these tables to produce output data files, by copying data into them through the INSERT ... SELECT syntax and then extracting the data files from the Impala data directory.

The data files created by any INSERT statements uses the Ctrl-A character (hex 01) as a separator between each column value.

Issue a DESCRIBE FORMATTED *TABLE_NAME* statement to see the details of how each table is represented internally in Impala.

Complex type considerations: Although you can create tables in this file format using the complex types (ARRAY, STRUCT, and MAP), currently, Impala cannot query these types in text tables.

Data Files for Text Tables

When Impala queries a table with data in text format, it consults all the data files in the data directory for that table, with some exceptions:

- Impala ignores any hidden files, that is, files whose names start with a dot or an underscore.
- Impala queries ignore files with extensions commonly used for temporary work files by Hadoop tools. Any files with extensions .tmp or .copying are not considered part of the Impala table. The suffix matching is case-insensitive, so for example Impala ignores both .copying and .COPYING suffixes.
- Impala uses suffixes to recognize when text data files are compressed text. For Impala to recognize the compressed text files, they must have the appropriate file extension corresponding to the compression codec, either .bz2, .gz, .snappy, or .zst, .deflate. The extensions can be in uppercase or lowercase.
- Otherwise, the file names are not significant. When you put files into an HDFS directory through ETL jobs, or point Impala to an existing HDFS directory with the CREATE EXTERNAL TABLE statement, or move data files under external control with the LOAD DATA statement, Impala preserves the original filenames.

An INSERT ... SELECT statement produces one data file from each node that processes the SELECT part of the statement. An INSERT ... VALUES statement produces a separate data file for each statement; because Impala is more efficient querying a small number of huge files than a large number of tiny files, the INSERT ... VALUES syntax is not recommended for loading a substantial volume of data. If you find yourself with a table that is inefficient due to too many small data files, reorganize the data into a few large files by doing INSERT ... SELECT to transfer the data to a new table.

Do not surround string values with quotation marks in text data files that you construct. If you need to include the separator character inside a field value, for example to put a string value with a comma inside a CSV-format data file, specify an escape character on the CREATE TABLE statement with the ESCAPED BY clause, and insert that character immediately before any separator characters that need escaping.

Special values within text data files:

- Impala recognizes the literal strings inf for infinity and nan for “Not a Number”, for FLOAT and DOUBLE columns.

- Impala recognizes the literal string \N to represent NULL. When using Sqoop, specify the options --null-non-s tring and --null-string to ensure all NULL values are represented correctly in the Sqoop output files. \N needs to be escaped as in the below example:

```
--null-string '\\N' --null-non-string '\\N'
```

- By default, Sqoop writes NULL values using the string null, which causes a conversion error when such rows are evaluated by Impala. (A workaround for existing tables and data files is to change the table properties through ALTER TABLE *NAME* SET TBLPROPERTIES("serialization.null.format"="null").)
- Impala can optionally skip an arbitrary number of header lines from text input files on HDFS based on the skip .header.line.count value in the TBLPROPERTIES field of the table metadata.

Loading Data into Text Tables

To load an existing text file into an Impala text table, use the LOAD DATA statement and specify the path of the file in HDFS. That file is moved into the appropriate Impala data directory.

To load multiple existing text files into an Impala text table, use the LOAD DATA statement and specify the HDFS path of the directory containing the files. All non-hidden files are moved into the appropriate Impala data directory.

Use the DESCRIBE FORMATTED statement to see the HDFS directory where the data files are stored, then use Linux commands such as hdfs dfs -ls *HDFS_DIRECTORY* and hdfs dfs -cat *HDFS_FILE* to display the contents of an Impala-created text file.

When you create a text file for use with an Impala text table, specify \N to represent a NULL value.

If a text file has fewer fields than the columns in the corresponding Impala table, all the corresponding columns are set to NULL when the data in that file is read by an Impala query.

If a text file has more fields than the columns in the corresponding Impala table, the extra fields are ignored when the data in that file is read by an Impala query.

You can also use manual HDFS operations such as hdfs dfs -put or hdfs dfs -cp to put data files in the data directory for an Impala table. When you copy or move new data files into the HDFS directory for the Impala table, issue a REFRESH *TABLE_NAME* statement in `impala-shell` before issuing the next query against that table, to make Impala recognize the newly added files.

Query Performance for Text Tables

Data stored in text format is relatively bulky, and not as efficient to query as binary formats such as Parquet. For the tables used in your most performance-critical queries, look into using more efficient alternate file formats.

For frequently queried data, you might load the original text data files into one Impala table, then use an INSERT statement to transfer the data to another table that uses the Parquet file format. The data is converted automatically as it is stored in the destination table.

For more compact data. consider using text data compressed in the gzip, bzip2, or Snappy formats. However, these compressed formats are not “splittable” so there is less opportunity for Impala to parallelize queries on them. You also have the choice to convert the data to Parquet using an INSERT ... SELECT statement to copy the original data into a Parquet table.

Using bzip2, deflate, gzip, Snappy-Compressed, or zstd Text Files

Impala supports using text data files that employ bzip2, deflate, gzip, Snappy, or zstd compression. These compression types are primarily for convenience within an existing ETL pipeline rather than maximum performance. Although it requires less I/O to read compressed text than the equivalent uncompressed text, files compressed by these codecs are not “splittable” and therefore cannot take full advantage of the Impala parallel query capability. Impala can read compressed text files written by Hive.

As each Snappy-compressed file is processed, the node doing the work reads the entire file into memory and then decompresses it. Therefore, the node must have enough memory to hold both the compressed and uncompressed data

from the text file. The memory required to hold the uncompressed data is difficult to estimate in advance, potentially causing problems on systems with low memory limits or with resource management enabled. This memory overhead is reduced for bzip2-, deflate-, gzip-, and zstd-compressed text files. The compressed data is decompressed as it is read, rather than all at once.

To create a table to hold compressed text, create a text table with no special compression options. Specify the delimiter and escape character if required, using the ROW FORMAT clause.

Because Impala can query compressed text files but currently cannot write them, produce the compressed text files outside Impala and use the LOAD DATA statement, manual HDFS commands to move them to the appropriate Impala data directory. (Or, you can use CREATE EXTERNAL TABLE and point the LOCATION attribute at a directory containing existing compressed text files.)

Using Parquet Data Files

Impala allows you to create, manage, and query Parquet tables. Parquet is a column-oriented binary file format intended to be highly efficient for the types of large-scale queries.

Parquet is suitable for queries scanning particular columns within a table, for example, to query “wide” tables with many columns, or to perform aggregation operations such as SUM() and AVG() that need to process most or all of the values from a column.

Each Parquet data file written by Impala contains the values for a set of rows (referred to as the “row group”). Within a data file, the values from each column are organized so that they are all adjacent, enabling good compression for the values from that column. Queries against a Parquet table can retrieve and analyze these values from any column quickly and with minimal I/O.

Creating Parquet Tables

To create a table in the Parquet format, use the STORED AS PARQUET clause in the CREATE TABLE statement. For example:

```
CREATE TABLE PARQUET_TABLE_NAME (x INT, y STRING) STORED AS PARQUET;
```

Or, to clone the column names and data types of an existing table, use the LIKE with the STORED AS PARQUET clause. For example:

```
CREATE TABLE PARQUET_TABLE_NAME LIKE OTHER_TABLE_NAME STORED AS PARQUET;
```

You can derive column definitions from a raw Parquet data file, even without an existing Impala table. For example, you can create an external table pointing to an HDFS directory, and base the column definitions on one of the files in that directory:

```
CREATE EXTERNAL TABLE ingest_existing_files LIKE PARQUET '/user/etl/destinat
ion/datafile1.dat'
  STORED AS PARQUET
  LOCATION '/user/etl/destination';
```

Or, you can refer to an existing data file and create a new empty table with suitable column definitions. Then you can use INSERT to create new data files or LOAD DATA to transfer existing data files into the new table.

```
CREATE TABLE columns_from_data_file LIKE PARQUET '/user/etl/destination/data
file1.dat'
  STORED AS PARQUET;
```

In this example, the new table is partitioned by year, month, and day. These partition key columns are not part of the data file, so you specify them in the CREATE TABLE statement:

```
CREATE TABLE columns_from_data_file LIKE PARQUET '/user/etl/destination/data
file1.dat'
  PARTITION (year INT, month TINYINT, day TINYINT)
  STORED AS PARQUET;
```

If the Parquet table has a different number of columns or different column names than the other table, specify the names of columns from the other table rather than * in the SELECT statement.

Data Type Considerations for Parquet Tables

The Parquet format defines a set of data types whose names differ from the names of the corresponding Impala data types. If you are preparing Parquet files using other Hadoop components such as Pig or MapReduce, you might need to work with the type names defined by Parquet. The following tables list the Parquet-defined types and the equivalent types in Impala.

Primitive types

Parquet type	**Impala type**
BINARY	STRING
BOOLEAN	BOOLEAN
DOUBLE	DOUBLE
FLOAT	FLOAT
INT32	INT
INT64	BIGINT
INT96	TIMESTAMP

Logical types

Parquet uses type annotations to extend the types that it can store, by specifying how the primitive types should be interpreted.

Parquet primitive type and annotation	**Impala type**
BINARY annotated with the UTF8 OriginalType	STRING
BINARY annotated with the STRING LogicalType	STRING
BINARY annotated with the ENUM OriginalType	STRING
BINARY annotated with the DECIMAL OriginalType	DECIMAL
INT64 annotated with the TIMESTAMP_MILLIS OriginalType	TIMESTAMP or BIGINT (for backward compatibility)
INT64 annotated with the TIMESTAMP_MICROS OriginalType	TIMESTAMP or BIGINT (for backward compatibility)
INT64 annotated with the TIMESTAMP LogicalType	TIMESTAMP or

Parquet primitive type and annotation	Impala type
	BIGINT (for backward compatibility)

Complex types:

Impala only supports queries against the complex types (ARRAY, MAP, and STRUCT) in Parquet tables.

Loading Data into Parquet Tables

Choose from the following process to load data into Parquet tables based on whether the original data is already in an Impala table, or exists as raw data files outside Impala.

If you already have data in an Impala or Hive table, perhaps in a different file format or partitioning scheme:

- Transfer the data to a Parquet table using the Impala INSERT...SELECT statement.

  For example:

  ```
  INSERT OVERWRITE TABLE parquet_table_name SELECT * FROM other_table_name;
  ```

  You can convert, filter, repartition, and do other things to the data as part of this same INSERT statement.

  When inserting into partitioned tables, especially using the Parquet file format, you can include a hint in the INSERT statement to fine-tune the overall performance of the operation and its resource usage.

Any INSERT statement for a Parquet table requires enough free space in the HDFS filesystem to write one block. Because Parquet data files use a block size of 1 GB by default, an INSERT might fail (even for a very small amount of data) if your HDFS is running low on space.

Avoid the INSERT...VALUES syntax for Parquet tables, because INSERT...VALUES produces a separate tiny data file for each INSERT...VALUES statement, and the strength of Parquet is in its handling of data (compressing, parallelizing, and so on) in large chunks.

If you have one or more Parquet data files produced outside of Impala, you can quickly make the data query-able through Impala by one of the following methods:

- The LOAD DATA statement moves a single data file or a directory full of data files into the data directory for an Impala table. It does no validation or conversion of the data.

  The original data files must be somewhere in HDFS, not the local filesystem.
- The CREATE TABLE statement with the LOCATION clause creates a table where the data continues to reside outside the Impala data directory.

  The original data files must be somewhere in HDFS, not the local filesystem.

  For extra safety, if the data is intended to be long-lived and reused by other applications, you can use the CREATE EXTERNAL TABLE syntax so that the data files are not deleted by an Impala DROP TABLE statement.
- If the Parquet table already exists, you can copy Parquet data files directly into it using the hadoop distcp -pb command, then use the REFRESH statement to make Impala recognize the newly added data.

  You must preserve the block size of the Parquet data files by using the hadoop distcp -pb command rather than a -put or -cp operation on the Parquet files.

Note:

Currently, Impala always decodes the column data in Parquet files based on the ordinal position of the columns, not by looking up the position of each column based on its name. Parquet files produced outside of Impala must write column data in the same order as the columns are declared in the Impala table. Any optional columns that are omitted from the data files must be the rightmost columns in the Impala table definition.

If you created compressed Parquet files through some tool other than Impala, make sure that any compression codecs are supported in Parquet by Impala. For example, Impala does not currently support LZO compression in Parquet files. Also doublecheck that you used any recommended compatibility settings in the other tool, such as spark.sql.parquet.binaryAsString when writing Parquet files through Spark.

Recent versions of Sqoop can produce Parquet output files using the --as-parquetfile option.

If the data exists outside Impala and is in some other format, combine both of the preceding techniques. First, use a LOAD DATA or CREATE EXTERNAL TABLE ... LOCATION statement to bring the data into an Impala table that uses the appropriate file format. Then, use an INSERT...SELECT statement to copy the data to the Parquet table, converting to Parquet format as part of the process.

Loading data into Parquet tables is a memory-intensive operation, because the incoming data is buffered until it reaches one data block in size, then that chunk of data is organized and compressed in memory before being written out. The memory consumption can be larger when inserting data into partitioned Parquet tables, because a separate data file is written for each combination of partition key column values, potentially requiring several large chunks to be manipulated in memory at once.

When inserting into a partitioned Parquet table, Impala redistributes the data among the nodes to reduce memory consumption. You might still need to temporarily increase the memory dedicated to Impala during the insert operation, or break up the load operation into several INSERT statements, or both.

Query Performance for Parquet Tables

Query performance for Parquet tables depends on the number of columns needed to process the SELECT list and WHERE clauses of the query, the way data is divided into large data files with block size equal to file size, the reduction in I/O by reading the data for each column in compressed format, which data files can be skipped (for partitioned tables), and the CPU overhead of decompressing the data for each column.

For example, the following is an efficient query for a Parquet table:

```
SELECT AVG(income) FROM census_data WHERE state = 'CA';
```

The query processes only 2 columns out of a large number of total columns. If the table is partitioned by the STATE column, it is even more efficient because the query only has to read and decode 1 column from each data file, and it can read only the data files in the partition directory for the state 'CA', skipping the data files for all the other states, which will be physically located in other directories.

The following is a relatively inefficient query for a Parquet table:

```
SELECT * FROM census_data;
```

Impala would have to read the entire contents of each large data file, and decompress the contents of each column for each row group, negating the I/O optimizations of the column-oriented format. This query might still be faster for a Parquet table than a table with some other file format, but it does not take advantage of the unique strengths of Parquet data files.

Impala can optimize queries on Parquet tables, especially join queries, better when statistics are available for all the tables. Issue the COMPUTE STATS statement for each table after substantial amounts of data are loaded into or appended to it.

The runtime filtering feature works best with Parquet tables. The per-row filtering aspect only applies to Parquet tables.

Impala queries are optimized for files stored in Amazon S3. For Impala tables that use the Parquet file formats, the PARQUET_OBJECT_STORE_SPLIT_SIZE query option determines how Impala divides the I/O work of reading the data files. By default, this value is 256 MB. Impala parallelizes S3 read operations on the files as if they were made up of 256 MB blocks to match the row group size produced by Impala.

Parquet files written by Impala include embedded metadata specifying the minimum and maximum values for each column, within each row group and each data page within the row group. Impala-written Parquet files typically contain a single row group; a row group can contain many data pages. Impala uses this information (currently, only the metadata for each row group) when reading each Parquet data file during a query, to quickly determine whether each row group within the file potentially includes any rows that match the conditions in the WHERE clause.

For example, if the column X within a particular Parquet file has a minimum value of 1 and a maximum value of 100, then a query including the clause WHERE x > 200 can quickly determine that it is safe to skip that particular file, instead of scanning all the associated column values.

This optimization technique is especially effective for tables that use the SORT BY clause for the columns most frequently checked in WHERE clauses, because any INSERT operation on such tables produces Parquet data files with relatively narrow ranges of column values within each file.

To disable Impala from writing the Parquet page index when creating Parquet files, set the PARQUET_WRITE_PAGE_INDEX query option to FALSE.

Partitioning for Parquet Tables

Partitioning is an important performance technique for Impala generally. This section explains some of the performance considerations for partitioned Parquet tables.

The Parquet file format is ideal for tables containing many columns, where most queries only refer to a small subset of the columns. The physical layout of Parquet data files lets Impala read only a small fraction of the data for many queries. The performance benefits of this approach are amplified when you use Parquet tables in combination with partitioning. Impala can skip the data files for certain partitions entirely, based on the comparisons in the WHERE clause that refer to the partition key columns. For example, queries on partitioned tables often analyze data for time intervals based on columns such as YEAR, MONTH, and/or DAY, or for geographic regions.

As Parquet data files use a large block size, when deciding how finely to partition the data, try to find a granularity where each partition contains 256 MB or more of data, rather than creating a large number of smaller files split among many partitions.

Inserting into a partitioned Parquet table can be a resource-intensive operation, because each Impala node could potentially be writing a separate data file to HDFS for each combination of different values for the partition key columns. The large number of simultaneous open files could exceed the HDFS “transceivers” limit. To avoid exceeding this limit, consider the following techniques:

- Load different subsets of data using separate INSERT statements with specific values for the PARTITION clause, such as PARTITION (year=2010).
- Increase the “transceivers” value for HDFS, sometimes spelled “xcievers” (sic). The property value in the hdfs-site.xml configuration file is dfs.datanode.max.transfer.threads.

  For example, if you were loading 12 years of data partitioned by year, month, and day, even a value of 4096 might not be high enough.
- Use the COMPUTE STATS statement to collect column statistics on the source table from which data is being copied, so that the Impala query can estimate the number of different values in the partition key columns and distribute the work accordingly.

Enabling Compression for Parquet Tables

When Impala writes Parquet data files using the INSERT statement, the underlying compression is controlled by the COMPRESSION_CODEC query option. The allowed values for this query option are snappy (the default), gzip, zstd, lz4, and none, the compression codecs that Impala supports for Parquet.

Snappy

By default, the underlying data files for a Parquet table are compressed with Snappy. The combination of fast compression and decompression makes it a good choice for many data sets.

GZip

If you need more intensive compression (at the expense of more CPU cycles for uncompressing during queries), set the COMPRESSION_CODEC query option to gzip before inserting the data.

Zstd

Zstd is a real-time compression algorithm offering a tradeoff between speed and ratio of compression. Compression levels from 1 up to 22 are supported. The lower the level, the faster the speed at the cost of compression ratio.

Lz4

Lz4 is a lossless compression algorithm providing extremely fast and scalable compression and decompression.

None

If your data compresses very poorly, or you want to avoid the CPU overhead of compression and decompression entirely, set the COMPRESSION_CODEC query option to none before inserting the data.

The actual compression ratios, and relative insert and query speeds, will vary depending on the characteristics of the actual data.

Because Parquet data files are typically large, each directory will have a different number of data files and the row groups will be arranged differently.

At the same time, the less aggressive the compression, the faster the data can be decompressed.

For example, using a table with a billion rows, switching from Snappy to GZip compression shrinks the data by an additional 40% or so, while switching from Snappy compression to no compression expands the data also by about 40%. A query that evaluates all the values for a particular column runs faster with no compression than with Snappy compression, and faster with Snappy compression than with Gzip compression.

The data files using the various compression codecs are all compatible with each other for read operations. The metadata about the compression format is written into each data file, and can be decoded during queries regardless of the COMPRESSION_CODEC setting in effect at the time.

Exchanging Parquet Data Files with Other Cloudera Components

You can read and write Parquet data files from other Cloudera components, such as Hive.

Impala supports the scalar data types that you can encode in a Parquet data file, but not composite or nested types such as maps or arrays. Impala can query Parquet data files that include composite or nested types, as long as the query only refers to columns with scalar types.

If you copy Parquet data files between nodes, or even between different directories on the same node, make sure to preserve the block size by using the command hadoop distcp -pb. To verify that the block size was preserved, issue the command hdfs fsck -blocks *HDFS_PATH_OF_IMPALA_TABLE_DIR* and check that the average block size is at or near 256 MB (or whatever other size is defined by the PARQUET_FILE_SIZE query option).. (The hadoop distcp operation typically leaves some directories behind, with names matching _distcp_logs_*, that you can delete from the destination directory afterward.)

Issue the command `hadoop distcp` for details about `distcp` command syntax.

Impala can query Parquet files that use the PLAIN, PLAIN_DICTIONARY, BIT_PACKED, and RLE encodings. As of CDP 7.2.8, Impala supports decodingRLE_DICTIONARY encoded pages. This encoding is identical to the already-supported PLAIN_DICTIONARY encoding but the PLAIN enum value is used for the dictionary pages and the RLE_DICTIONARY enum value is used for the data pages. When creating files outside of Impala for use by Impala, make sure to use one of the supported encodings.

In particular, for MapReduce jobs, parquet.writer.version must not be defined (especially as PARQUET_2_0) for writing the configurations of Parquet MR jobs.

Data using the version 2.0 of Parquet writer might not be consumable by Impala, due to use of the RLE_DICTIONARY encoding.

Use the default version of the Parquet writer and refrain from overriding the default writer version by setting the parquet.writer.version property or via WriterVersion.PARQUET_2_0 in the Parquet API.

How Parquet Data Files Are Organized

Although Parquet is a column-oriented file format, Parquet keeps all the data for a row within the same data file, to ensure that the columns for a row are always available on the same node for processing. Parquet sets a large HDFS block size and a matching maximum data file size to ensure that I/O and network transfer requests apply to large batches of data.

Within that data file, the data for a set of rows is rearranged so that all the values from the first column are organized in one contiguous block, then all the values from the second column, and so on. Putting the values from the same column next to each other lets Impala use effective compression techniques on the values in that column.

Note:

Impala INSERT statements write Parquet data files using an HDFS block size that matches the data file size, to ensure that each data file is represented by a single HDFS block, and the entire file can be processed on a single node without requiring any remote reads.

If you create Parquet data files outside of Impala, such as through a MapReduce or Pig job, ensure that the HDFS block size is greater than or equal to the file size, so that the “one file per block” relationship is maintained. Set the dfs.block.size or the dfs.blocksize property large enough that each file fits within a single HDFS block, even if that size is larger than the normal HDFS block size.

If the block size is reset to a lower value during a file copy, you will see lower performance for queries involving those files, and the PROFILE statement will reveal that some I/O is being done suboptimally, through remote reads.

When Impala retrieves or tests the data for a particular column, it opens all the data files, but only reads the portion of each file containing the values for that column. The column values are stored consecutively, minimizing the I/O required to process the values within a single column. If other columns are named in the SELECT list or WHERE clauses, the data for all columns in the same row is available within that same data file.

If an INSERT statement brings in less than one Parquet block's worth of data, the resulting data file is smaller than ideal. Thus, if you do split up an ETL job to use multiple INSERT statements, try to keep the volume of data for each INSERT statement to approximately 256 MB, or a multiple of 256 MB.

RLE and Dictionary Encoding for Parquet Data Files

Parquet uses some automatic compression techniques, such as run-length encoding (RLE) and dictionary encoding, based on analysis of the actual data values. Once the data values are encoded in a compact form, the encoded data can optionally be further compressed using a compression algorithm. Parquet data files created by Impala can use Snappy, GZip, or no compression; the Parquet spec also allows LZO compression, but currently Impala does not support LZO-compressed Parquet files.

RLE and dictionary encoding are compression techniques that Impala applies automatically to groups of Parquet data values, in addition to any Snappy or GZip compression applied to the entire data files. These automatic optimizations can save you time and planning that are normally needed for a traditional data warehouse. For example, dictionary encoding reduces the need to create numeric IDs as abbreviations for longer string values.

Run-length encoding condenses sequences of repeated data values. For example, if many consecutive rows all contain the same value for a country code, those repeating values can be represented by the value followed by a count of how many times it appears consecutively.

Dictionary encoding takes the different values present in a column, and represents each one in compact 2-byte form rather than the original value, which could be several bytes. (Additional compression is applied to the compacted values, for extra space savings.) This type of encoding applies when the number of different values for a column is less than 2**16 (16,384). It does not apply to columns of data type BOOLEAN, which are already very short. TIME

STAMP columns sometimes have a unique value for each row, in which case they can quickly exceed the 2**16 limit on distinct values. The 2**16 limit on different values within a column is reset for each data file, so if several different data files each contained 10,000 different city names, the city name column in each data file could still be condensed using dictionary encoding.

Compacting Data Files for Parquet Tables

If you reuse existing table structures or ETL processes for Parquet tables, you might encounter a “many small files” situation, which is suboptimal for query efficiency.

Here are techniques to help you produce large data files in Parquet INSERT operations, and to compact existing too-small data files:

- When inserting into a partitioned Parquet table, use statically partitioned INSERT statements where the partition key values are specified as constant values. Ideally, use a separate INSERT statement for each partition.
- You might set the NUM_NODES option to 1 briefly, during INSERT or CREATE TABLE AS SELECT statements. Normally, those statements produce one or more data files per data node. If the write operation involves small amounts of data, a Parquet table, and/or a partitioned table, the default behavior could produce many small files when intuitively you might expect only a single output file. SET NUM_NODES=1 turns off the “distributed” aspect of the write operation, making it more likely to produce only one or a few data files.
- Be prepared to reduce the number of partition key columns from what you are used to with traditional analytic database systems.
- Do not expect Impala-written Parquet files to fill up the entire Parquet block size. Impala estimates on the conservative side when figuring out how much data to write to each Parquet file. Typically, the uncompressed data in memory is substantially reduced on disk by the compression and encoding techniques in the Parquet file format. The final data file size varies depending on the compressibility of the data. Therefore, it is not an indication of a problem if 256 MB of text data is turned into 2 Parquet data files, each less than 256 MB.
- If you accidentally end up with a table with many small data files, consider using one or more of the preceding techniques and copying all the data into a new Parquet table, either through CREATE TABLE AS SELECT or INSERT ... SELECT statements.

  To avoid rewriting queries to change table names, you can adopt a convention of always running important queries against a view. Changing the view definition immediately switches any subsequent queries to use the new underlying tables:

Schema Evolution for Parquet Tables

Schema evolution refers to using the statement ALTER TABLE ... REPLACE COLUMNS to change the names, data type, or number of columns in a table. You can perform schema evolution for Parquet tables as follows:

- The Impala ALTER TABLE statement never changes any data files in the tables. From the Impala side, schema evolution involves interpreting the same data files in terms of a new table definition. Some types of schema changes make sense and are represented correctly. Other types of changes cannot be represented in a sensible way, and produce special result values or conversion errors during queries.
- The INSERT statement always creates data using the latest table definition. You might end up with data files with different numbers of columns or internal data representations if you do a sequence of INSERT and ALTER TABLE ... REPLACE COLUMNS statements.
- If you use ALTER TABLE ... REPLACE COLUMNS to define additional columns at the end, when the original data files are used in a query, these final columns are considered to be all NULL values.
- If you use ALTER TABLE ... REPLACE COLUMNS to define fewer columns than before, when the original data files are used in a query, the unused columns still present in the data file are ignored.

- Parquet represents the TINYINT, SMALLINT, and INT types the same internally, all stored in 32-bit integers.
  - That means it is easy to promote a TINYINT column to SMALLINT or INT, or a SMALLINT column to INT. The numbers are represented exactly the same in the data file, and the columns being promoted would not contain any out-of-range values.
  - If you change any of these column types to a smaller type, any values that are out-of-range for the new type are returned incorrectly, typically as negative numbers.
  - You cannot change a TINYINT, SMALLINT, or INT column to BIGINT, or the other way around. Although the ALTER TABLE succeeds, any attempt to query those columns results in conversion errors.
  - Any other type conversion for columns produces a conversion error during queries. For example, INT to STRI NG, FLOAT to DOUBLE, TIMESTAMP to STRING, DECIMAL(9,0) to DECIMAL(5,2), and so on.

You might find that you have Parquet files where the columns do not line up in the same order as in your Impala table. For example, you might have a Parquet file that was part of a table with columns C1,C2,C3,C4, and now you want to reuse the same Parquet file in a table with columns C4,C2. By default, Impala expects the columns in the data file to appear in the same order as the columns defined for the table, making it impractical to do some kinds of file reuse or schema evolution.

The query option PARQUET_FALLBACK_SCHEMA_RESOLUTION=name lets Impala resolve columns by name, and therefore handle out-of-order or extra columns in the data file.

Using ORC Data Files

Impala can read ORC data files.

Creating ORC Tables and Loading Data

To create a table in the ORC format, use the STORED AS ORC clause in the CREATE TABLE statement.

Because Impala can query ORC tables but cannot currently write to, after creating ORC tables, use the Hive shell to load the data.

After loading data into a table through Hive or other mechanism outside of Impala, issue a REFRESH *TABLE_NAME* statement the next time you connect to the Impala node, before querying the table, to make Impala recognize the new data.

For example, here is how you might create some ORC tables in Impala (by specifying the columns explicitly, or cloning the structure of another table), load data through Hive, and query them through Impala:

```
$ impala-shell -i localhost
[localhost:21000] default> CREATE TABLE orc_table (x INT) STORED AS ORC;
[localhost:21000] default> CREATE TABLE orc_clone LIKE some_other_table STO
RED AS ORC;
[localhost:21000] default> quit;

$ hive
hive> INSERT INTO TABLE orc_table SELECT x FROM some_other_table;
3 Rows loaded to orc_table
Time taken: 4.169 seconds
hive> quit;
$ impala-shell -i localhost
[localhost:21000] default> -- Make Impala recognize the data loaded through
 Hive;
[localhost:21000] default> REFRESH orc_table;
[localhost:21000] default> SELECT * FROM orc_table;

Fetched 3 row(s) in 0.11s
```

Data Type Considerations for ORC Tables

The ORC format defines a set of data types whose names differ from the names of the corresponding Impala data types. The following figure lists the ORC-defined types and the equivalent types in Impala.

Primitive types:

ORC type	Impala type
BINARY	STRING
BOOLEAN	BOOLEAN
DOUBLE	DOUBLE
FLOAT	FLOAT
TINYINT	TINYINT
SMALLINT	SMALLINT
INT	INT
BIGINT	BIGINT
TIMESTAMP	TIMESTAMP
DATE	Not supported

Complex types:

Impala supports the complex types ARRAY, STRUCT, and MAP in ORC files. Because Impala has better performance on Parquet than ORC, if you plan to use complex types, become familiar with the performance and storage aspects of Parquet first.

Enabling Compression for ORC Tables

ORC tables are in ZLIB (Deflate in Impala) compression by default. The supported compressions for ORC tables are NONE, ZLIB, and SNAPPY.

Set the compression when you load ORC tables in Hive.

Using Avro Data Files

Impala supports creating and querying Avro tables. You need to use Hive to insert data into Avro tables.

Creating Avro Tables

To create a new table using the Avro file format, use the STORED AS AVRO clause in the CREATE TABLE statement. If you create the table through Impala, you must include column definitions that match the fields specified in the Avro schema. With Hive, you can omit the columns and just specify the Avro schema.

The following examples demonstrate creating an Avro table in Impala, using either an inline column specification or one taken from a JSON file stored in HDFS:

```
[localhost:21000] > CREATE TABLE avro_only_sql_columns
                  > (col1 BOOLEAN, col2 INT)
                  > STORED AS AVRO;

[localhost:21000] > CREATE TABLE impala_avro_table
                  > (col1 BOOLEAN, col2 INT)
                  > STORED AS AVRO
```

```
                    > TBLPROPERTIES ('avro.schema.literal'='{
                    >    "name": "my_record",
                    >    "type": "record",
                    >    "fields": [
                    >       {"name":"col1", "type":"boolean"},
                    >       {"name":"col2", "type":"int"}]}');
[localhost:21000] > CREATE TABLE avro_examples_of_all_types
                      (col1 BOOLEAN, col2 INT)
                    >   STORED AS AVRO
                    >   TBLPROPERTIES ('avro.schema.url'='hdfs://localhost:802
0/avro_schemas/alltypes.json');
```

Each field of the record becomes a column of the table. Note that any other information, such as the record name, is ignored.

Note: For nullable Avro columns, make sure to put the "null" entry before the actual type name. In Impala, all columns are nullable; Impala currently does not have a NOT NULL clause. Any non-nullable property is only enforced on the Avro side.

If you create the table through Hive, switch back to `impala-shell` and issue an INVALIDATE METADATA *TABLE_NAME* statement. Then you can run queries for that table through `impala-shell`.

In rare instances, a mismatch could occur between the Avro schema and the column definitions in the Metastore database. Impala checks for such inconsistencies during a CREATE TABLE statement and each time it loads the metadata for a table (for example, after INVALIDATE METADATA). Impala uses the following rules to determine how to treat mismatching columns, a process known as *schema reconciliation*:

- If there is a mismatch in the number of columns, Impala uses the column definitions from the Avro schema.
- If there is a mismatch in column name or type, Impala uses the column definition from the Avro schema. Because a CHAR or VARCHAR column in Impala maps to an Avro STRING, this case is not considered a mismatch and the column is preserved as CHAR or VARCHAR in the reconciled schema.
- An Impala TIMESTAMP column definition maps to an Avro STRING and is presented as a STRING in the reconciled schema, because Avro has no binary TIMESTAMP representation.

Specifying the Avro Schema through JSON:

While you can embed a schema directly in your CREATE TABLE statement, as shown above, column width restrictions in the Hive Metastore limit the length of schema you can specify. If you encounter problems with long schema literals, try storing your schema as a JSON file in HDFS instead. Specify your schema in HDFS using table properties in the following format using the avro.schema.url in TBLPROPERTIES clause.

```
TBLPROPERTIES ('avro.schema.url'='hdfs//your-name-node:port/path/to/schema.j
son');
```

Data Type Considerations for Avro Tables

The Avro format defines a set of data types whose names differ from the names of the corresponding Impala data types. If you are preparing Avro files using other Hadoop components such as Pig or MapReduce, you might need to work with the type names defined by Avro. The following figure lists the Avro-defined types and the equivalent types in Impala.

Primitive types:

Avro type	Impala type
STRING	STRING
STRING	CHAR
STRING	VARCHAR
INT	INT

Avro type	Impala type
BOOLEAN	BOOLEAN
LONG	BIGINT
FLOAT	FLOAT
DOUBLE	DOUBLE

The Avro specification allows string values up to 2**64 bytes in length:

- Impala queries for Avro tables use 32-bit integers to hold string lengths.
- Impala truncates CHAR and VARCHAR values in Avro tables to (2**31)-1 bytes.
- If a query encounters a STRING value longer than (2**31)-1 bytes in an Avro table, the query fails.

Logical types:

Avro type	Impala type
BYTES annotated	DECIMAL
INT32 annotated	DATE

Avro types not supported by Impala

- RECORD
- MAP
- ARRAY
- UNION
- ENUM
- FIXED
- NULL

Impala types not supported by Avro:

- TIMESTAMP

Impala issues warning messages if there are any mismatches between the types specified in the SQL column definitions and the underlying types; for example, any TINYINT or SMALLINT columns are treated as INT in the underlying Avro files, and therefore are displayed as INT in any DESCRIBE or SHOW CREATE TABLE output.

Note:

Currently, Avro tables cannot contain TIMESTAMP columns. If you need to store date and time values in Avro tables, as a workaround you can use a STRING representation of the values, convert the values to BIGINT with the UNIX_TIMESTAMP() function, or create separate numeric columns for individual date and time fields using the EXTRACT() function.

Using a Hive-Created Avro Table in Impala

If you have an Avro table created through Hive, you can use it in Impala as long as it contains only Impala-compatible data types. It cannot contain Avro types not supported by Impala, such as ENUM and FIXED Because Impala and Hive share the same metastore database, Impala can directly access the table definitions and data for tables that were created in Hive.

If you create an Avro table in Hive, issue an INVALIDATE METADATA in Impala. This is a one-time operation to make Impala aware of the new table. You can issue the statement while connected to any Impala node, and the catalog service broadcasts the change to all other Impala nodes.

If you load new data into an Avro table through Hive, either through a Hive LOAD DATA or INSERT statement, or by manually copying or moving files into the data directory for the table, issue a REFRESH *TABLE_NAME* statement the next time you connect to Impala through `impala-shell`.

If you issue the LOAD DATA statement through Impala, you do not need a REFRESH afterward.

Impala only supports fields of type BOOLEAN, INT, LONG, FLOAT, DOUBLE, and STRING, or unions of these types with null, for example, ["string", "null"]. Unions with null essentially create a nullable type.

Loading Data into Avro Tables

Currently, Impala cannot write Avro data files. Therefore, an Avro table cannot be used as the destination of an Impala INSERT statement or CREATE TABLE AS SELECT.

To copy data from another table, issue any INSERT statements through Hive.

After loading data into a table through Hive or other mechanism outside of Impala, issue a REFRESH *TABLE_NAME* statement the next time you connect to the Impala node, before querying the table, to make Impala recognize the new data.

If you already have data files in Avro format, you can also issue LOAD DATA in either Impala or Hive. Impala can move existing Avro data files into an Avro table, it just cannot create new Avro data files.

Enabling Compression for Avro Tables

To enable compression for Avro tables, specify settings in the Hive shell to enable compression and to specify a codec, then issue a CREATE TABLE statement as in the preceding examples. Impala supports the snappy and deflate codecs for Avro tables.

For example:

```
hive> set hive.exec.compress.output=true;
hive> set avro.output.codec=snappy;
```

Handling Avro Schema Evolution

Impala can handle with Avro data files that employ *schema evolution*, where different data files within the same table use slightly different type definitions. (You would perform the schema evolution operation by issuing an ALTER TABLE statement in the Hive shell.) The old and new types for any changed columns must be compatible, for example a column might start as an INT and later change to a BIGINT or FLOAT.

As with any other tables where the definitions are changed or data is added outside of the current `impalad` node, ensure that Impala loads the latest metadata for the table if the Avro schema is modified through Hive. Issue a REFRESH *TABLE_NAME* or INVALIDATE METADATA *TABLE_NAME* statement. REFRESH reloads the metadata immediately, INVALIDATE METADATA reloads the metadata the next time the table is accessed.

When Avro data files or columns are not consulted during a query, Impala does not check for consistency. Thus, if you issue SELECT c1, c2 FROM t1, Impala does not return any error if the column c3 changed in an incompatible way. If a query retrieves data from some partitions but not others, Impala does not check the data files for the unused partitions.

In the Hive DDL statements, you can specify an avro.schema.literal table property (if the schema definition is short) or an avro.schema.url property (if the schema definition is long, or to allow convenient editing for the definition).

Query Performance for Avro Tables

In general, expect query performance with Avro tables to be faster than with tables using text data, but slower than with Parquet tables.

Using RCFile Data Files

Impala supports creating and querying RCFile tables.

Creating RCFile Tables and Loading Data

To create a table in the RCFile format, use the STORED AS RCFILE clause in the CREATE TABLE statement.

Because Impala can query RCFile tables but cannot currently write to, after creating RCFile tables, use the Hive shell to load the data.

After loading data into a table through Hive or other mechanism outside of Impala, issue a REFRESH *TABLE_NAME* statement the next time you connect to the Impala node, before querying the table, to make Impala recognize the new data.

Query Performance for RCFile Tables

In general, expect query performance with RCFile tables to be faster than with tables using text data, but slower than with Parquet tables.

Enabling Compression for RCFile Tables

You may want to enable compression on existing tables. Enabling compression provides performance gains in most cases and is supported for RCFile tables.

The compression type is specified in the SET mapred.output.compression.codec command.

For example, to enable Snappy compression, you would specify the following additional settings when loading data through the Hive shell.

```
SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCode
c;
```

Using SequenceFile Data Files

Impala supports creating and querying SequenceFile tables.

Creating SequenceFile Tables and Loading Data

To create a table in the SequenceFile format, use the STORED AS SEQUENCEFILE clause in the CREATE T ABLE statement.

Because Impala can query SequenceFile tables but cannot currently write to, after creating SequenceFile tables, use the Hive shell to load the data.

After loading data into a table through Hive or other mechanism outside of Impala, issue a REFRESH *TABLE_NAME* statement the next time you connect to the Impala node, before querying the table, to make Impala recognize the new data.

Enabling Compression for SequenceFile Tables

You may want to enable compression on existing tables. Enabling compression provides performance gains in most cases and is supported for SequenceFile tables.

The compression type is specified in the SET mapred.output.compression.codec command.

For example, to enable Snappy compression, you would specify the following additional settings when loading data through the Hive shell.

```
SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCode
c;
```

Query Performance for SequenceFile Tables

In general, expect query performance with SequenceFile tables to be faster than with tables using text data, but slower than with Parquet tables.

Storage Systems Supports

This section describes the storage systems that Impala supports.

Impala with Azure Data Lake Store (ADLS)

You can use Impala to query data residing on the Azure Data Lake Store (ADLS) filesystem and Azure Blob File System (ABFS). This capability allows convenient access to a storage system that is remotely managed, accessible from anywhere, and integrated with various cloud-based services.

Impala can query files in any supported file format from ADLS. The ADLS storage location can be for an entire table or individual partitions in a partitioned table.

The default Impala tables use data files stored on HDFS, which are ideal for bulk loads and queries using full-table scans. In contrast, queries against ADLS data are less performant, making ADLS suitable for holding “cold” data that is only queried occasionally, while more frequently accessed “hot” data resides in HDFS. In a partitioned table, you can set the LOCATION attribute for individual partitions to put some partitions on HDFS and others on ADLS, typically depending on the age of the data.

Impala requires that the default filesystem for the cluster be HDFS. You cannot use ADLS as the only filesystem in the cluster.

To be able to access ADLS, first set up an Azure account, configure an ADLS store, and configure your cluster with appropriate credentials.

Creating Impala Databases, Tables, and Partitions for Data Stored on ADLS

To create a table that resides on ADLS, specify the ADLS details in the LOCATION clause of the CREATE TABLE or ALTER TABLE statement. The syntax for the LOCATION clause is:

- For ADLS Gen1:

```
LOCATION 'adl://ACCOUNT.azuredatalakestore.net/PATH/FILE'
```

- For ADLS Gen2:

```
LOCATION 'abfs://CONTAINER@ACCOUNT.dfs.core.windows.net/PATH/FILE'
```

  or

```
LOCATION 'abfss://CONTAINER@ACCOUNT.dfs.core.windows.net/PATH/FILE'
```

CONTAINER denotes the parent location that holds the files and folders, which is the Containers in the Azure Storage Blobs service.

ACCOUNT is the name given for your storage account.

Any reference to an ADLS location must be fully qualified. (This rule applies when ADLS is not designated as the default filesystem.)

Once a table or partition is designated as residing on ADLS, the SELECT statement transparently accesses the data files from the appropriate storage layer.

ALTER TABLE can also set the LOCATION property for an individual partition so that some data in a table resides on ADLS and other data in the same table resides on HDFS.

Note:

By default, TLS is enabled both with abfs:// and abfss://.

When you set the fs.azure.always.use.https=false property, TLS is disabled with abfs://, and TLS is enabled with abfss://

For a partitioned table, either specify a separate LOCATION clause for each new partition, or specify a base LOCATION for the table and set up a directory structure in ADLS to mirror the way Impala partitioned tables are structured in HDFS.

Although, strictly speaking, ADLS filenames do not have directory paths, Impala treats ADLS filenames with / characters the same as HDFS pathnames that include directories.

To point a nonpartitioned table or an individual partition at ADLS, specify a single directory path in ADLS, which could be any arbitrary directory.

To replicate the structure of an entire Impala partitioned table or database in ADLS requires more care, with directories and subdirectories nested and named to match the equivalent directory tree in HDFS. Consider setting up an empty staging area if necessary in HDFS, and recording the complete directory structure so that you can replicate it in ADLS.

For example, the following session creates a partitioned table where only a single partition resides on ADLS. The partitions for years 2013 and 2014 are located on HDFS. The partition for year 2015 includes a LOCATION attribute with an adl:// URL, and so refers to data residing on ADLS, under a specific path underneath the store impalademo.

```
CREATE TABLE mostly_on_hdfs (x INT) PARTITIONED BY (year INT);
ALTER TABLE mostly_on_hdfs ADD PARTITION (year=2013);
ALTER TABLE mostly_on_hdfs ADD PARTITION (year=2014);
ALTER TABLE mostly_on_hdfs ADD PARTITION (year=2015)
   LOCATION 'adl://impalademo.azuredatalakestore.net/dir1/dir2/dir3/t1';
```

When working with multiple tables with data files stored in ADLS, you can create a database with the LOCATION attribute pointing to an ADLS path. Specify a URL of the form as shown above. Any tables created inside that database automatically create directories underneath the one specified by the database LOCATION attribute.

Use the standard ADLS file upload methods to actually put the data files into the right locations. You can also put the directory paths and data files in place before creating the associated Impala databases or tables, and Impala automatically uses the data from the appropriate location after the associated databases and tables are created.

You can switch whether an existing table or partition points to data in HDFS or ADLS. For example, if you have an Impala table or partition pointing to data files in HDFS or ADLS, and you later transfer those data files to the other filesystem, use an ALTER TABLE statement to adjust the LOCATION attribute of the corresponding table or partition to reflect that change. This location-switching technique is not practical for entire databases that have a custom LOCATION attribute.

You cannot use the ALTER TABLE ... SET CACHED statement for tables or partitions that are located in ADLS.

Internal and External Tables Located on ADLS

Just as with tables located on HDFS storage, you can designate ADLS-based tables as either internal (managed by Impala) or external, by using the syntax CREATE TABLE or CREATE EXTERNAL TABLE respectively.

When you drop an internal table, the files associated with the table are removed, even if they are on ADLS storage. When you drop an external table, the files associated with the table are left alone, and are still available for access by other tools or components.

If the data on ADLS is intended to be long-lived and accessed by other tools in addition to Impala, create any associated ADLS tables with the CREATE EXTERNAL TABLE syntax, so that the files are not deleted from ADLS when the table is dropped.

If the data on ADLS is only needed for querying by Impala and can be safely discarded once the Impala workflow is complete, create the associated ADLS tables using the CREATE TABLE syntax so that dropping the table also deletes the corresponding data files on ADLS.

Loading Data into ADLS for Impala Queries

If your ETL pipeline involves moving data into ADLS and then querying through Impala, you can either use Impala DML statements to create, move, or copy the data, or use the same data loading techniques as you would for non-Impala data.

Using Impala DML Statements for ADLS Data:

The Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in the Azure Data Lake Store (ADLS) or ADLS Gen2.

Manually Loading Data into Impala Tables on ADLS:

You can use the Microsoft-provided methods to bring data files into ADLS for querying through Impala. See the Microsoft ADLS documentation for details.

After you upload data files to a location already mapped to an Impala table or partition, or if you delete files in ADLS from such a location, issue the REFRESH statement to make Impala aware of the new set of data files.

Running and Queries for Data Stored on ADLS

Once the appropriate LOCATION attributes are set up at the table or partition level, you query data stored in ADLS the same as data stored on HDFS or in HBase:

- Queries against ADLS data support all the same file formats as for HDFS data.
- Tables can be unpartitioned or partitioned. For partitioned tables, either manually construct paths in ADLS corresponding to the HDFS directories representing partition key values, or use ALTER TABLE ... ADD PARTITION to set up the appropriate paths in ADLS.
- HDFS, Kudu, and HBase tables can be joined to ADLS tables, or ADLS tables can be joined with each other.
- Authorization to control access to databases, tables, or columns works the same whether the data is in HDFS or in ADLS.
- The `catalogd` daemon caches metadata for both HDFS and ADLS tables. Use REFRESH and INVALIDATE METADATA for ADLS tables in the same situations where you would issue those statements for HDFS tables.
- Queries against ADLS tables are subject to the same kinds of admission control and resource management as HDFS tables.
- Metadata about ADLS tables is stored in the same Metastore database as for HDFS tables.
- You can set up views referring to ADLS tables, the same as for HDFS tables.
- The COMPUTE STATS, SHOW TABLE STATS, and SHOW COLUMN STATS statements support ADLS tables.

Query Performance for ADLS Data

Although Impala queries for data stored in ADLS might be less performant than queries against the equivalent data stored in HDFS, you can still do some tuning. Here are techniques you can use to interpret explain plans and profiles for queries against ADLS data, and tips to achieve the best performance possible for such queries.

All else being equal, performance is expected to be lower for queries running against data on ADLS rather than HDFS. The actual mechanics of the SELECT statement are somewhat different when the data is in ADLS. Although the work is still distributed across the datanodes of the cluster, Impala might parallelize the work for a distributed query differently for data on HDFS and ADLS. ADLS does not have the same block notion as HDFS, so Impala uses heuristics to determine how to split up large ADLS files for processing in parallel. Because all hosts can access any ADLS data file with equal efficiency, the distribution of work might be different than for HDFS data, where the data blocks are physically read using short-circuit local reads by hosts that contain the appropriate block replicas. Although the I/O to read the ADLS data might be spread evenly across the hosts of the cluster, the fact that all data

is initially retrieved across the network means that the overall query performance is likely to be lower for ADLS data than for HDFS data.

Because data files written to ADLS do not have a default block size the way HDFS data files do, any Impala INSERT or CREATE TABLE AS SELECT statements use the PARQUET_FILE_SIZE query option setting to define the size of Parquet data files. (Using a large block size is more important for Parquet tables than for tables that use other file formats.)

When optimizing aspects of for complex queries such as the join order, Impala treats tables on HDFS and ADLS the same way.

In query profile reports, the numbers for BytesReadLocal, BytesReadShortCircuit, BytesReadDataNodeCached, and BytesReadRemoteUnexpected are blank because those metrics come from HDFS.

All the I/O for ADLS tables involves remote reads, and they will appear as “remote read” operations in the query profile.

Impala with Amazon S3

You can use Impala to query data residing on the Amazon S3 object store. This capability allows convenient access to a storage system that is remotely managed, accessible from anywhere, and integrated with various cloud-based services.

Impala can query files in any supported file format from S3. The S3 storage location can be for an entire table, or individual partitions in a partitioned table.

Best Practices for Using Impala with S3

The following guidelines summarize the best practices described in the rest of this topic:

- Any reference to an S3 location must be fully qualified when S3 is not designated as the default storage, for example, s3a:://[s3-bucket-name].
- DROP TABLE .. PURGE is much faster than the default DROP TABLE. The same applies to ALTER TABLE ... DROP PARTITION PURGE versus the default DROP PARTITION operation. However, due to the eventually consistent nature of S3, the files for that table or partition could remain for some unbounded time when using PURGE. The default DROP TABLE/PARTITION is slow because Impala copies the files to the S3A trashcan, and Impala waits until all the data is moved. DROP TABLE/PARTITION .. PURGE is a fast delete operation, and the Impala statement finishes quickly even though the change might not have propagated fully throughout S3.
- INSERT statements are faster than INSERT OVERWRITE for S3. The S3_SKIP_INSERT_STAGING query option, which is set to true by default, skips the staging step for regular INSERT (but not INSERT OVERWRITE). This makes the operation much faster, but consistency is not guaranteed: if a node fails during execution, the table could end up with inconsistent data. Set this option to false if stronger consistency is required, but this setting will make the INSERT operations slower.
  - For Impala-ACID tables, both INSERT and INSERT OVERWRITE tables for S3 are fast, regardless of the setting of S3_SKIP_INSERT_STAGING. Plus, consistency is guaranteed with ACID tables.
- Too many files in a table can make metadata loading and updating slow in S3. If too many requests are made to S3, S3 has a back-off mechanism and responds slower than usual.
  - If you have many small files due to over-granular partitioning, configure partitions with many megabytes of data so that even a query against a single partition can be parallelized effectively.
  - If you have many small files because of many small INSERT queries, use bulk INSERTs so that more data is written to fewer files.

Loading Data into S3 for Impala Queries

If your ETL pipeline involves moving data into S3 and then querying through Impala, you can either use Impala DML statements to create, move, or copy the data, or use the same data loading techniques as you would for non-Impala data.

Using Impala DML Statements for S3 Data:

Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in S3.

Because of differences between S3 and traditional filesystems, DML operations for S3 tables can take longer than for tables on HDFS. For example, both the LOAD DATA statement and the final stage of the INSERT and CREATE TABLE AS SELECT statements involve moving files from one directory to another. (In the case of INSERT and CREATE TABLE AS SELECT, the files are moved from a temporary staging directory to the final destination directory.) Because S3 does not support a “rename” operation for existing objects, in these cases, Impala copies the data files from one location to another and then removes the original files.

Manually Loading Data into Impala Tables in S3:

You can use the Amazon-provided methods to bring data files into S3 for querying through Impala.

After you upload data files to a location already mapped to an Impala table or partition, or if you delete files in S3 from such a location outside of Impala, issue the REFRESH statement to make Impala aware of the new set of data files.

Specifying Impala Credentials to Access S3

Specify an Impala credential to access data in Amazon S3.

Procedure

1. In Cloudera Manager, navigate to Administration External Accounts .
2. In the AWS Credentials tab, click Add Access Key Credentials.
3. Enter a Name of your choosing for this account.
4. Enter the AWS Access Key ID.
5. Enter the AWS Secret Key.
6. Click Add.
7. Click Save to finish adding the AWS Credential.
8. Select Cluster Access to S3.

Configure Impala Daemon to spill to S3

Impala occasionally needs to use persistent storage for writing intermediate files during large sorts, joins, aggregations, or analytic function operations. If your workload results in large volumes of intermediate data being written, it is recommended to configure the heavy spilling queries to use a remote storage location rather than the local one. The advantage of using remote storage for scratch space is that it is elastic and can handle any amount of spilling.

Before you begin

Identify the URL for an S3 bucket to which you want your new Impala to write the temporary data. If you use the S3 bucket that is associated with the environment, navigate to the S3 bucket and copy the URL. If you want to use an external S3 bucket, you must first configure your CDP environment to use the external S3 bucket with the correct read/write permissions.

Configuring the Start-up Option in Impala daemon

You can use the Impalad start option scratch_dirs to specify the locations of the intermediate files. The format of the option is scratch_dirs= remote_dir, local_buffer_dir(, local_dir…).

With the option specified above:

- You can specify only one remote directory.

  When you configure a remote directory, you must specify a local buffer directory as the buffer. However you can use multiple local directories with the remote directory. If you specify multiple local directories, the first local directory would be used as the local buffer directory.
- If you configure both remote and local directories, the remote directory is only used when the local directories are fully utilized.
- The size of a remote intermediate file could affect the query performance, and the value can be set by remote_tmp_file_size in the start-up option. The default size of a remote intermediate file is 16MB while the maximum is 256MB.

Examples

- A remote scratch dir with one local buffer dir, file size 64MB.

```
##scratch_dirs="s3a://remote_dir, /local_buffer_dir"
    ##remote_tmp_file_size=64M
```

- A remote scratch dir with one local buffer dir, and one local dir.

```
##scratch_dirs="s3a://remote_dir, /local_buffer_dir, /local_dir"
```

- A remote scratch dir with one local buffer dir, and multiple local dirs.

```
##scratch_dirs="s3a://remote_dir, /local_buffer_dir, /local_dir_1,
    /local_dir_2"
```

Ports Used by Impala

Impala uses the TCP ports listed in the following table.

The following table contains the ports used by an Impala VW. Ensure that they are accessible and not blocked.

Scope	Port Details	Default Port
Impala Coordinator	HiveServer2 HTTP Port	28000
Impala Coordinator	Impala Coordinator debug Web UI	25000
Impala Statestore	Impala Statestore debug Web UI	25010
Impala Catalog	Impala Catalog debug Web UI	25020

SQL transactions in Impala

A transaction is a single logical operation on the data. Impala supports transactions that satisfy a level of consistency that improves the integrity and reliability of the data before and after a transaction.

Specifically, Impala provides atomicity and isolation of insert operations on transactional tables. A single table insert is either committed in full or not committed, and the results of the insert operation are not visible to other query operations until the operation is committed.

For single table, the inserts are ordered, so if Impala doesn't see a committed insert, it won't see any insert committed after it.

For insert-only transactional tables, you can perform the following statements: CREATE TABLE, DROP TABLE, TRUNCATE, INSERT, SELECT

All transactions in Impala automatically commit at the end of the statement. Currently, Impala does not support multi-statement transactions.

Insert-only tables must be the managed and file-format based tables, such as Parquet, Avro, and text.

Note: Impala does not support changing transactional properties of tables. For example, you cannot alter a transactional table to a non-transactional table.

1.0.0

Apache Impala SQL Reference

Date published: 2020-11-30
Date modified: 2024-12-05

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Impala SQL Overview **6**
Impala schema objects **6**
Aliases 6
Impala databases 7
Overview of Impala functions 8
Identifiers 9
Impala tables 10
Impala views 12
Impala SQL data types **13**
ARRAY complex type 13
BIGINT data type 16
Binary data type 17
BOOLEAN data type 19
CHAR data type 21
DATE data type 22
DECIMAL data type 23
DOUBLE data type 28
FLOAT data type 30
INT data type 31
MAP complex type 32
REAL data type 36
SMALLINT data type 36
STRING data type 37
STRUCT complex type 39
TIMESTAMP data type 45
Customizing time zones 50
TINYINT data type 51
VARCHAR data type 52
Complex types 54
Querying arrays 87
Zipping unnest on arrays from views 89
Impala SQL literals **89**
Impala SQL operators **93**
Impala SQL comments **105**
Impala SQL statements **105**
ROLE statements 105
DDL statements 106
DML statements 107
ALTER DATABASE statement 108
ALTER TABLE statement 108
ALTER VIEW statement 122
COMMENT statement 124
COMPUTE STATS statement 124
CREATE DATABASE statement 132
CREATE FUNCTION statement 134
CREATE ROLE statement 140
CREATE TABLE statement 141
CREATE VIEW statement 157
DELETE statement 158
DESCRIBE statement 160

DROP DATABASE statement....172
DROP FUNCTION statement....174
DROP ROLE statement....176
DROP STATS statement....176
DROP TABLE statement....181
DROP VIEW statement....182
EXPLAIN statement....183
GRANT statement....186
GRANT ROLE statement....188
INSERT statement....189
INVALIDATE METADATA statement....198
LOAD DATA statement....199
REFRESH statement....203
REFRESH AUTHORIZATION statement....205
REFRESH FUNCTIONS statement....205
REVOKE statement....205
REVOKE ROLE statement....206
SELECT statement....206
Joins in Impala SELECT statements....209
ORDER BY clause....215
GROUP BY clause....219
HAVING clause....223
LIMIT clause....223
OFFSET clause....225
UNION, INTERSECT, and EXCEPT clauses....226
Subqueries in Impala SELECT statements....229
TABLESAMPLE clause....232
WITH clause....238
DISTINCT operator....238
SET statement....239
SHOW statement....240
SHOW ROLES statement....245
SHOW CURRENT ROLES statement....246
SHOW ROLE GRANT GROUP statement....246
SHUTDOWN statement....246
TRUNCATE TABLE statement....247
UPDATE statement....250
UPSERT statement....251
USE statement....252
VALUES statement....253
Optimizer hints in Impala....254
Query options....259
Virtual column....302
Built-in functions....303
Impala mathematical functions....308
Impala bit functions....326
Conversion functions....336
Impala date and time functions....344
Impala conditional functions....363
Impala string functions....368
Impala miscellaneous functions....385
Impala aggregate functions....388
APPX_MEDIAN function....389
AVG function....390
COUNT function....394
GROUPING() and GROUPING_ID() functions....399

GROUP_CONCAT function.... 401
MAX function.... 402
MIN function.... 406
NDV function.... 410
STDDEV, STDDEV_SAMP, STDDEV_POP functions.... 413
SUM function.... 415
VARIANCE, VARIANCE_SAMP, VARIANCE_POP, VAR_SAMP, VAR_POP functions.... 419
Impala analytic functions.... 421
OVER.... 422
WINDOW.... 423
AVG.... 425
COUNT.... 425
CUME_DIST.... 425
DENSE_RANK.... 427
FIRST_VALUE.... 429
LAG.... 431
LAST_VALUE.... 433
LEAD.... 434
MAX.... 435
MIN.... 435
NTILE.... 435
PERCENT_RANK.... 436
RANK.... 437
ROW_NUMBER.... 440
SUM.... 441
User-defined functions (UDFs).... 441
UDF concepts.... 442
Runtime environment for UDFs.... 445
Installing the UDF development package.... 446
Writing UDFs.... 446
Writing user-defined aggregate functions (UDAFs).... 449
Building and deploying UDFs.... 450
Performance considerations for UDFs.... 452
Examples of creating and using UDFs.... 452
Security considerations for UDFs.... 459
Limitations and restrictions for Impala UDFs.... 459
SQL transactions in Impala.... 459
Impala reserved words.... 460
SQL differences between Impala and Hive.... 474
Porting SQL from other database systems to Impala.... 476
UTF-8 Support.... 480

Apache Impala SQL Overview

You can use Apache Impala SQL to manage and access data in Hadoop storage. Review the basic concepts related to Apache Impala SQL such as schema objects you can use to store and manipulate data, supported data types, literals, operators, built-in functions, and so on.

Impala provides a high degree of compatibility with the Hive Query Language. Impala uses the same metadata store as Hive to record information about table structure and properties. Therefore, Impala can access tables defined through the built-in Impala CREATE TABLE command, or tables created using the Hive data definition language (DDL).

Impala schema objects

You must understand what schema objects are, what they include, and where they are stored. The conceptual information related to these objects help you to understand the various ways to specify their names and work with them.

Schema objects are logical structures that you use to store and manipulate data. Schema objects include databases, tables, views, functions, and so on.

Impala aliases

You must be aware of certain guidelines and rules about assigning an alias to the names of tables, columns, or column expressions in SQL statements and using the alias when referencing the table or column in the same statement.

Aliases rules

The following rules apply to aliases:

- You typically specify aliases that are shorter, easier to remember, or both than the original names. The aliases are printed in the query header, making them useful for self-documenting output.

  To set up an alias, add the AS *ALIAS* clause immediately after any table, column, or expression name in the SELE CT list or FROM list of a query. The AS keyword is optional.

  You can specify the alias immediately after the original name.
- You can specify column aliases with or without the AS keyword, and with no quotation marks, single quotation marks, or double quotation marks.
- Aliases are not case sensitive.
- Aliases can be up to the maximum length of a Java string, 2147483647.
- Aliases can include additional characters such as spaces and dashes when they are quoted using backtick characters (``).
- To use an alias name that matches one of the Impala reserved keywords, surround the identifier with either single or double quotation marks, or backtick characters.
- Aliases are allowed at the top level of the GROUP BY, HAVING, and ORDER BY clauses, e.g. GROUP BY *ALIAS*.
- Aliases are not allowed in subexpressions of the GROUP BY, HAVING, and ORDER BY clauses, e,g. GROUP BY *ALIAS/5*.

From Impala 3.0, the alias substitution logic in the GROUP BY, HAVING, and ORDER BY clauses has become more consistent with standard SQL behavior, as follows. Aliases are now only legal at the top level, and not in subexpressions. The following statements are allowed:

```
SELECT int_col / 2 AS x
FROM t
GROUP BY x;

SELECT int_col / 2 AS x
FROM t
ORDER BY x;
SELECT NOT bool_col AS nb
FROM t
GROUP BY nb
HAVING nb;
```

And the following statements are NOT allowed:

```
SELECT int_col / 2 AS x
FROM t
GROUP BY x / 2;

SELECT int_col / 2 AS x
FROM t
ORDER BY -x;

SELECT int_col / 2 AS x
FROM t
GROUP BY x
HAVING x > 3;
```

Note: Impala does not allow column aliases in the WHERE clause.

Related Information

Impala reserved words

Impala databases

Provides information about how to manage Impala databases that are logical containers for a group of tables.

Each database defines a separate namespace. Creating a database is a lightweight operation. There are minimal database-specific properties to configure, such as LOCATION and COMMENT.

You can change the owner of a database with the ALTER DATABASE statement.

Typically, you create a separate database for each project or application, to avoid naming conflicts between tables and to make clear which tables are related to each other.

The USE statement lets you switch between databases. Unqualified references to tables, views, and functions refer to objects within the current database. You can also refer to objects in other databases by using qualified names of the form *DBNAME.OBJECT_NAME*.

Each database is physically represented by a directory in HDFS. When you do not specify a LOCATION attribute, the directory is located in the Impala data directory with the associated tables managed by Impala. When you do specify a LOCATION attribute, any read and write operations for tables in that database are relative to the specified HDFS directory.

Impala includes two predefined databases:

- default

  By default, you use the default database when you connect to Impala. Tables created in the default database are physically located at one level higher in HDFS than all the user-created databases.
- _impala_builtins

  All Impala built-in functions reside in this database.

Related Information

CREATE DATABASE statement

Overview of Impala functions

Provides information about using Functions to apply arithmetic, string, or other computations and transformations to Impala data.

You can use Functions in SELECT lists and WHERE clauses to filter and format query results so that the result set is exactly what you want, with no further processing needed on the application side.

Scalar functions return a single result for each input row.

```
[localhost:21000] > select name, population from country where continent = '
North America' order by population desc limit 4;
[localhost:21000] > select upper(name), population from country where conti
nent = 'North America' order by population desc limit 4;
+-------------+------------+
| upper(name) | population |
+-------------+------------+
| USA         | 320000000  |
| MEXICO      | 122000000  |
| CANADA      | 25000000   |
| GUATEMALA   | 16000000   |
+-------------+------------+
```

Aggregate functions combine the results from multiple rows: either a single result for the entire table, or a separate result for each group of rows. Aggregate functions are frequently used in combination with GROUP BY and HAVI NG clauses in the SELECT statement.

```
[localhost:21000] > select continent, sum(population) as howmany from cou
ntry group by continent order by howmany desc;
+---------------+------------+
| continent     | howmany    |
+---------------+------------+
| Asia          | 4298723000 |
| Africa        | 1110635000 |
| Europe        | 742452000  |
| North America | 565265000  |
| South America | 406740000  |
| Oceania       | 38304000   |
+---------------+------------+
```

User-defined functions (UDFs) let you code your own logic. They can be either scalar or aggregate functions. UDFs let you implement important business or scientific logic using high-performance code for Impala to automatically parallelize. You can also use UDFs to implement convenience functions to simplify reporting or porting SQL from other database systems.

```
[localhost:21000] > select rot13('Hello world!') as 'Weak obfuscation';
+------------------+
| weak obfuscation |
+------------------+
| Uryyb jbeyq!     |
```

```
+------------------+
[localhost:21000] > select likelihood_of_new_subatomic_particle(sensor1,
 sensor2, sensor3) as probability
                  > from experimental_results group by experiment;
```

Each function is associated with a specific database. For example, if you issue a USE somedb statement followed by CREATE FUNCTION somefunc, the new function is created in the somedb database, and you could refer to it through the fully qualified name somedb.somefunc. You could then issue another USE statement and create a function with the same name in a different database.

Impala built-in functions are associated with a special database named _impala_builtins, which lets you refer to them from any database without qualifying the name.

```
[localhost:21000] > show databases;
+-------------------------+
| name                    |
+-------------------------+
| _impala_builtins        |
| analytic_functions      |
| avro_testing            |
| data_file_size          |
...
[localhost:21000] > show functions in _impala_builtins like '*subs*';
+-------------+-----------------------------------+
| return type | signature                         |
+-------------+-----------------------------------+
| STRING      | substr(STRING, BIGINT)            |
| STRING      | substr(STRING, BIGINT, BIGINT)    |
| STRING      | substring(STRING, BIGINT)         |
| STRING      | substring(STRING, BIGINT, BIGINT) |
+-------------+-----------------------------------+
```

Related Information

Impala built-in functions

Impala identifiers

Provides information about using Identifiers as the names of databases, tables, or columns when creating the objects.

The following rules apply to identifiers in Impala:

- The minimum length of an identifier is 1 character.
- The maximum length of an identifier is currently 128 characters, enforced by the Metastore database.
- An identifier must start with an alphanumeric or underscore character. The remainder can contain any combination of alphanumeric characters and underscores.

  Quoting the identifier with backticks has no effect on the allowed characters in the name.
- An identifier can contain only ASCII characters.
- To use an identifier name that matches one of the Impala reserved keywords, surround the identifier with `` characters (backticks). Quote the reserved word even if it is part of a fully qualified name.

  The following example shows how a reserved word can be used as a column name if it is quoted with backticks in the CREATE TABLE statement, and how the column name must also be quoted with backticks in a query:

  Important: Because the list of reserved words grows over time as new SQL syntax is added, consider adopting coding conventions (especially for any automated scripts or in packaged applications) to always quote all identifiers with backticks. Quoting all identifiers protects your SQL from compatibility issues if new reserved words are added in later releases.
- Impala identifiers are case-insensitive.

Related Information

Impala reserved words

Impala tables

Provides information about Tables that are the primary containers for data in Impala.

Logically, each table has a structure based on the definition of its columns, partitions, and other properties.

Physically, each table that uses HDFS storage is associated with a directory in HDFS. The table data consists of all the data files underneath that directory:

- Internal tables are managed by Impala, and use directories inside the designated Impala work area.
- External tables use arbitrary HDFS directories, where the data files are typically shared between different Hadoop components.
- Large-scale data is usually handled by partitioned tables, where the data files are divided among different HDFS subdirectories.

Impala queries ignore files with extensions commonly used for temporary work files by Hadoop tools. Any files with extensions .tmp or .copying are not considered part of the Impala table. The suffix matching is case-insensitive, so for example Impala ignores both .copying and .COPYING suffixes.

When you create a table in Impala, you can create an internal table or an external table.

To see whether a table is internal or external, and its associated HDFS location, issue the statement DESCRIBE FORMATTED *TABLE_NAME*. The Table Type field displays MANAGED_TABLE for internal tables and EXTERNAL_TABLE for external tables. The Location field displays the path of the table directory as an HDFS URI.

Internal Tables

The default kind of table produced by the CREATE TABLE statement is known as an internal table.

- Impala creates a directory in HDFS to hold the data files.
- You can create data in internal tables by issuing INSERT or LOAD DATA statements.
- If you add or replace data using HDFS operations, issue the REFRESH command in `impala-shell` so that Impala recognizes the changes in data files, block locations, and so on.
- When you issue a DROP TABLE statement, Impala physically removes all the data files from the directory.
- To see whether a table is internal or external, and its associated HDFS location, issue the statement DESCRIBE FORMATTED *TABLE_NAME*. The Table Type field displays MANAGED_TABLE for internal tables and EXTERNAL_TABLE for external tables. The Location field displays the path of the table directory as an HDFS URI.
- When you issue an ALTER TABLE statement to rename an internal table, all data files are moved into the new HDFS directory for the table. The files are moved even if they were formerly in a directory outside the Impala data directory, for example in an internal table with a LOCATION attribute pointing to an outside HDFS directory.

External Tables

The syntax CREATE EXTERNAL TABLE sets up an Impala table that points at existing data files, potentially in HDFS locations outside the normal Impala data directories.. This operation saves the expense of importing the data into a new table when you already have the data files in a known location in HDFS, in the desired file format.

- You can use Impala to query the data in this table.
- You can create data in external tables by issuing INSERT or LOAD DATA statements.
- If you add or replace data using HDFS operations, issue the REFRESH command in `impala-shell` so that Impala recognizes the changes in data files, block locations, and so on.

- When you issue a DROP TABLE statement in Impala, that removes the connection that Impala has with the associated data files, but does not physically remove the underlying data. You can continue to use the data files with other Hadoop components and HDFS operations.
- When you issue an ALTER TABLE statement to rename an external table, all data files are left in their original locations.
- You can point multiple external tables at the same HDFS directory by using the same LOCATION attribute for each one. The tables could have different column definitions, as long as the number and types of columns are compatible with the schema evolution considerations for the underlying file type. For example, for text data files, one table might define a certain column as a STRING while another defines the same column as a BIGINT.

You can switch a table between internal to external by using the ALTER TABLE statement with the SET TBLPROPERTIES clause. For example:

```
ALTER TABLE TABLE_NAME SET TBLPROPERTIES('EXTERNAL'='TRUE');
```

If the Kudu service is integrated with the Hive Metastore, the above operations are not supported.

File formats

Each table has an associated file format, which determines how Impala interprets the associated data files.

You set the file format during the CREATE TABLE statement, or change it later using the ALTER TABLE statement.

Partitioned tables can have a different file format for individual partitions, allowing you to change the file format used in your ETL process for new data without going back and reconverting all the existing data in the same table.

Any INSERT statements produce new data files with the current file format of the table.

For existing data files, changing the file format of the table does not automatically do any data conversion. You must use TRUNCATE TABLE or INSERT OVERWRITE to remove any previous data files that use the old file format. Then you use the LOAD DATA statement, INSERT ... SELECT, or other mechanism to put data files of the correct format into the table.

The default file format, Parquet, offers the highest query performance and uses compression to reduce storage requirements; therefore, where practical, use Parquet for Impala tables with substantial amounts of data. Also, the complex types (ARRAY, STRUCT, and MAP) available in Impala 2.3 and higher are currently only supported with the Parquet file type.

Kudu tables

By default, tables stored in Apache Kudu are treated specially, because Kudu manages its data independently of HDFS files.

All metadata that Impala needs is stored in the HMS.

When Kudu is not integrated with the HMS, when you create a Kudu table through Impala, the table is assigned an internal Kudu table name of the form impala::*DB_NAME.TABLE_NAME*. You can see the Kudu-assigned name in the output of DESCRIBE FORMATTED, in the kudu.table_name field of the table properties.

For Impala-Kudu managed tables, ALTER TABLE ... RENAME renames both the Impala and the Kudu table.

For Impala-Kudu external tables, ALTER TABLE ... RENAME renames just the Impala table. To change the Kudu table that an Impala external table points to, use ALTER TABLE *IMPALA_NAME* SET TBLPROPERTIES('kudu.table_name' = '*DIFFERENT_KUDU_TABLE_NAME*'). The underlying Kudu table must already exist.

In practice, external tables are typically used to access underlying Kudu tables that were created outside of Impala, that is, through the Kudu API.

The SHOW TABLE STATS output for a Kudu table shows Kudu-specific details about the layout of the table. Instead of information about the number and sizes of files, the information is divided by the Kudu tablets. For each

tablet, the output includes the fields # Rows (although this number is not currently computed), Start Key, Stop Key, Leader Replica, and # Replicas. The output of SHOW COLUMN STATS, illustrating the distribution of values within each column, is the same for Kudu tables as for HDFS-backed tables.

If the Kudu service is not integrated with the Hive Metastore, the distinction between internal and external tables has some special details for Kudu tables. Tables created entirely through Impala are internal tables. The table name as represented within Kudu includes notation such as an impala:: prefix and the Impala database name. External Kudu tables are those created by a non-Impala mechanism, such as a user application calling the Kudu APIs. For these tables, the CREATE EXTERNAL TABLE syntax lets you establish a mapping from Impala to the existing Kudu table:

```
CREATE EXTERNAL TABLE impala_name STORED AS KUDU
  TBLPROPERTIES('kudu.table_name' = 'original_kudu_name');
```

External Kudu tables differ in one important way from other external tables: adding or dropping a column or range partition changes the data in the underlying Kudu table, in contrast to an HDFS-backed external table where existing data files are left untouched.

Related Information

Hadoop file formats supported

Storage systems supported

Impala views

Provides information about using Views as lightweight logical constructs that can act as aliases for queries. Using a view, you can issue complicated reporting queries with compact and simple syntax and reduce maintenance by avoiding the duplication of complicated queries across multiple applications in multiple languages.

You can specify a view name in a query (a SELECT statement or the SELECT portion of an INSERT statement) where you would usually specify a table name.

```
CREATE VIEW v2 AS SELECT t1.c1, t1.c2, t2.c3 FROM t1 JOIN t2 ON (t1.id = t2.
id);
-- This simple query is safer to embed in reporting applications than the l
onger query above.
-- The view definition can remain stable even if the structure of the und
erlying tables changes.
SELECT c1, c2, c3 FROM v2;
```

A view lets you to:

- Build a new, more refined query on top of the original query by adding new clauses, select-list expressions, function calls, and so on: This technique lets you build up several more or less granular variations of the same query, and switch between them when appropriate.
- Set up aliases with intuitive names for tables, columns and result sets from joins.
- Avoid coding lengthy subqueries and repeating the same subquery text in many other queries.

To create, alter, or drop views, use the CREATE VIEW, the ALTER VIEW, and the DROP VIEW statements respectively.

You cannot insert into views.

To see the definition of a view, issue a DESCRIBE FORMATTED statement, which shows the query from the original CREATE VIEW statement

Impala SQL data types

Impala supports a set of data types that you can use for table columns, expression values, and function arguments and return values. You must be aware of the supported data types, the purpose each data type serves, and the casting behaviour for each data type, prior to using them.

Note: Currently, Impala supports only scalar types, not composite or nested types. Accessing a table containing any columns with unsupported types causes an error.

For the notation to write literals of each of these data types, see Literals.

Impala supports a limited set of implicit casts to avoid undesired results from unexpected casting behavior.

- Impala does not implicitly cast between string and numeric or Boolean types. Always use CAST() for these conversions.
- Impala does perform implicit casts among the numeric types, when going from a smaller or less precise type to a larger or more precise one. For example, Impala will implicitly convert a SMALLINT to a BIGINT or FLOAT, but to convert from DOUBLE to FLOAT or INT to TINYINT requires a call to CAST() in the query.
- Impala does perform implicit casts from STRING to TIMESTAMP. Impala has a restricted set of literal formats for the TIMESTAMP data type and the FROM_UNIXTIME() format string; see TIMESTAMP data type on page 45 for details.

See the topics under this section for full details on implicit and explicit casting for each data type, and see Impala type conversion functions on page 336 for details about the CAST() function.

Related Information

Impala SQL literals

SQL differences between Impala and Hive

ARRAY complex type

Review the syntax, usage, and restrictions related to the ARRAY data type which is a complex data type that can represent an arbitrary number of ordered elements. The elements can be scalars or another complex type (ARRAY, STRUCT, or MAP).

Syntax:

```
COLUMN_NAME ARRAY < TYPE >

type ::= PRIMITIVE_TYPE | COMPLEX_TYPE
```

Usage:

Because complex types are often used in combination, for example an ARRAY of STRUCT elements, if you are unfamiliar with the Impala complex types, start with Complex types for background information and usage examples.

The elements of the array have no names. You refer to the value of the array item using the ITEM pseudocolumn, or its position in the array with the POS pseudocolumn.

Each row can have a different number of elements (including none) in the array for that row.

When an array contains items of scalar types, you can use aggregation functions on the array elements without using join notation. For example, you can find the COUNT(), AVG(), SUM(), and so on of numeric array elements, or the MAX() and MIN() of any scalar array elements by referring to *TABLE_NAME.ARRAY_COLUMN* in the FROM clause of the query. When you need to cross-reference values from the array with scalar values from the same row, such as by including a GROUP BY clause to produce a separate aggregated result for each row, then the join clause is required.

A common usage pattern with complex types is to have an array as the top-level type for the column: an array of structs, an array of maps, or an array of arrays. For example, you can model a denormalized table by creating a column that is an ARRAY of STRUCT elements; each item in the array represents a row from a table that would normally be used in a join query. This kind of data structure lets you essentially denormalize tables by associating multiple rows from one table with the matching row in another table.

You typically do not create more than one top-level ARRAY column, because if there is some relationship between the elements of multiple arrays, it is convenient to model the data as an array of another complex type element (either STRUCT or MAP).

You can pass a multi-part qualified name to DESCRIBE to specify an ARRAY, STRUCT, or MAP column and visualize its structure as if it were a table. For example, if table T1 contains an ARRAY column A1, you could issue the statement DESCRIBE t1.a1. If table T1 contained a STRUCT column S1, and a field F1 within the STRUCT was a MAP, you could issue the statement DESCRIBE t1.s1.f1. An ARRAY is shown as a two-column table, with ITEM and POS columns. A STRUCT is shown as a table with each field representing a column in the table. A MAP is shown as a two-column table, with KEY and VALUE columns.

Restrictions:

- Columns with this data type can only be used in tables or partitions with the Parquet file format.
- Columns with this data type cannot be used as partition key columns in a partitioned table.
- The COMPUTE STATS statement does not produce any statistics for columns of this data type.
- The maximum length of the column definition for any complex type, including declarations for any nested types, is 4000 characters.
- See the *Limitations and restrictions for complex types* topic for a full list of limitations and associated guidelines about complex type columns.

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Examples:

Note: Many of the complex type examples refer to tables such as CUSTOMER and REGION adapted from the tables used in the TPC-H benchmark. See Sample Schema and Data for Experimenting with Impala Complex Types for the table definitions.

The following example shows how to construct a table with various kinds of ARRAY columns, both at the top level and nested within other complex types. Whenever the ARRAY consists of a scalar value, such as in the PETS column or the CHILDREN field, you can see that future expansion is limited. For example, you could not easily evolve the schema to record the kind of pet or the child's birthday alongside the name. Therefore, it is more common to use an ARRAY whose elements are of STRUCT type, to associate multiple fields with each array element.

Note: Practice the CREATE TABLE and query notation for complex type columns using empty tables, until you can visualize a complex data structure and construct corresponding SQL statements reliably.

```
CREATE TABLE array_demo
(
  id BIGINT,
  name STRING,
-- An ARRAY of scalar type as a top-level column.
  pets ARRAY <STRING>,

-- An ARRAY with elements of complex type (STRUCT).
  places_lived ARRAY < STRUCT <
    place: STRING,
    start_year: INT
  >>,

-- An ARRAY as a field (CHILDREN) within a STRUCT.
-- (The STRUCT is inside another ARRAY, because it is rare
-- for a STRUCT to be a top-level column.)
```

```
  marriages ARRAY < STRUCT <
    spouse: STRING,
    children: ARRAY <STRING>
  >>,

-- An ARRAY as the value part of a MAP.
-- The first MAP field (the key) would be a value such as
-- 'Parent' or 'Grandparent', and the corresponding array would
-- represent 2 parents, 4 grandparents, and so on.
  ancestors MAP < STRING, ARRAY <STRING> >
)
STORED AS PARQUET;
```

The following example shows how to examine the structure of a table containing one or more ARRAY columns by using the DESCRIBE statement. You can visualize each ARRAY as its own two-column table, with columns ITEM and POS.

```
DESCRIBE array_demo;
+--------------+--------------------------+
| name         | type                     |
+--------------+--------------------------+
| id           | bigint                   |
| name         | string                   |
| pets         | array<string>            |
| marriages    | array<struct<            |
|              |   spouse:string,         |
|              |   children:array<string> |
|              | >>                       |
| places_lived | array<struct<            |
|              |   place:string,          |
|              |   start_year:int         |
|              | >>                       |
| ancestors    | map<string,array<string>> |
+--------------+--------------------------+

DESCRIBE array_demo.pets;
+------+--------+
| name | type   |
+------+--------+
| item | string |
| pos  | bigint |
+------+--------+
DESCRIBE array_demo.marriages;
+------+--------------------------+
| name | type                     |
+------+--------------------------+
| item | struct<                  |
|      |   spouse:string,         |
|      |   children:array<string> |
|      | >                        |
| pos  | bigint                   |
+------+--------------------------+
DESCRIBE array_demo.places_lived;
+------+------------------+
| name | type             |
+------+------------------+
| item | struct<          |
|      |   place:string,  |
|      |   start_year:int |
|      | >                |
| pos  | bigint           |
+------+------------------+
```

```
DESCRIBE array_demo.ancestors;
+-------+---------------+
| name  | type          |
+-------+---------------+
| key   | string        |
| value | array<string> |
+-------+---------------+
```

The following example shows queries involving ARRAY columns containing elements of scalar or complex types. You “unpack” each ARRAY column by referring to it in a join query, as if it were a separate table with ITEM and POS columns. If the array element is a scalar type, you refer to its value using the ITEM pseudocolumn. If the array element is a STRUCT, you refer to the STRUCT fields using dot notation and the field names. If the array element is another ARRAY or a MAP, you use another level of join to unpack the nested collection elements.

```
-- Array of scalar values.
-- Each array element represents a single string, plus we know its position
 in the array.
SELECT id, name, pets.pos, pets.item FROM array_demo, array_demo.pets;
-- Array of structs.
-- Now each array element has named fields, possibly of different types.
-- You can consider an ARRAY of STRUCT to represent a table inside another
 table.
SELECT id, name, places_lived.pos, places_lived.item.place, places_lived.it
em.start_year
FROM array_demo, array_demo.places_lived;
-- The .ITEM name is optional for array elements that are structs.
-- The following query is equivalent to the previous one, with .ITEM
-- removed from the column references.
SELECT id, name, places_lived.pos, places_lived.place, places_lived.start_ye
ar
  FROM array_demo, array_demo.places_lived;

-- To filter specific items from the array, do comparisons against the .POS
or .ITEM
-- pseudocolumns, or names of struct fields, in the WHERE clause.
SELECT id, name, pets.item FROM array_demo, array_demo.pets
  WHERE pets.pos in (0, 1, 3);

SELECT id, name, pets.item FROM array_demo, array_demo.pets
  WHERE pets.item LIKE 'Mr. %';

SELECT id, name, places_lived.pos, places_lived.place, places_lived.start_ye
ar
  FROM array_demo, array_demo.places_lived
WHERE places_lived.place like '%California%';
```

BIGINT data type

An 8-byte integer data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME BIGINT
```

Range: -9223372036854775808 .. 9223372036854775807. There is no UNSIGNED subtype.

Conversions: Impala automatically converts to a floating-point type (FLOAT or DOUBLE) automatically. Use CAST () to convert to TINYINT, SMALLINT, INT, STRING, or TIMESTAMP. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestam p_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Examples:

```
CREATE TABLE t1 (x BIGINT);
SELECT CAST(1000 AS BIGINT);
```

Usage notes:

BIGINT is a convenient type to use for column declarations because you can use any kind of integer values in INSE RT statements and they are promoted to BIGINT where necessary. However, BIGINT also requires the most bytes of any integer type on disk and in memory, meaning your queries are not as efficient and scalable as possible if you overuse this type. Therefore, prefer to use the smallest integer type with sufficient range to hold all input values, and CAST() when necessary to the appropriate type.

For a convenient and automated way to check the bounds of the BIGINT type, call the functions MIN_BIGINT() and MAX_BIGINT().

If an integer value is too large to be represented as a BIGINT, use a DECIMAL instead with sufficient digits of precision.

NULL considerations: Casting any non-numeric value to this type produces a NULL value.

Partitioning: Prefer to use this type for a partition key column. Impala can process the numeric type more efficiently than a STRING representation of the value.

HBase considerations: This data type is fully compatible with HBase tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as an 8-byte value.

Added in: Available in all versions of Impala.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Binary data type

This release adds support for BINARY columns for all table formats with the exception of Kudu.

Overview

The BINARY data type is an arbitrary-length byte array data type that can be used in CREATE TABLE and SELECT statements. This data type can contain data like images. The BINARY data type is very similar to the STRING data type except that the BINARY data type is not affected by UTF8_mode.

Syntax

The BINARY data type is defined in the column as shown here:

```
create table binary_table (v varchar, b binary);
```

Length

Take the following considerations for BINARY lengths:

- The hard limit on the size of a BINARY and the total size of a row is 512KB. If a query tries to process or create a BINARY larger than this limit, it will return an error to the user.
- The limit is 1 GB on BINARY when writing to Parquet files.
- Queries operating on BINARYs with 32 KB or less will work reliably and will not cause significant performance or memory problems (unless you have very complex queries, very many columns, etc.)
- Performance and memory consumption may degrade with BINARYs larger than 32 KB.
- The row size, i.e. the total size of all BINARY and other columns, is subject to lower limits at various points in query execution that support spill-to-disk. A few examples for lower row size limits are:
  - Rows coming from the right side of any hash join
  - Rows coming from either side of a hash join that spills to disk
  - Rows being sorted by the SORT operator without a limit
  - Rows in a grouping aggregation

Conversions:The only possible conversion for binary is explicit CAST from/to STRING.

Supported built-in functions

The following list contains some of the supported built-in functions.

- Scalar functions: length, concat, typeof, murmur_hash
- Conditional functions: if, coalesce
- Aggregate functions: min, max, count

Note: If the query option UTF8_MODE is set to FALSE, then the built-ins function the same way for STRING and BINARY.

Restrictions on BINARY columns

- For both table types PARQUET and ORC min/max stat filtering can not be used for BINARY columns, while it can be used for STRING columns.
- Currently, this data type cannot be used with Kudu tables.

Text table considerations:In textfiles Impala writes BINARY columns base64 encoded.

PARQUET considerations:BINARY columns are stored as BYTE_ARRAY in Parquet similar to string. For BINARY columns converted_type/logical type are never set, while they are set for STRINGs if the query option parquet_annotate_strings_utf8 is set to true.

ORC considerations:In ORC file format, there is an exclusive BINARY column type but Impala reads STRING/VARCHAR/CHAR ORC columns also as BINARY.

KUDU considerations:Currently, this data type cannot be used with Kudu tables.

Differences between Impala and Hive

As part of this release, behavior for Impala BINARY type is made to be consistent with Hive in most cases. However, some differences were not resolved to be backward compatible with older releases. You must be aware of these changes to run Hive queries, if needed, without modifying them.

Impala	Hive
Impala doesn't treat STRINGs as UTF8, so BINARY and STRING become nearly identical.	In Hive, the main difference between STRING and BINARY is that STRING is assumed to be UTF8 encoded, while BINARY can be any byte array.
The only possible conversion for binary is explicit CAST from/to STRING.	BINARY can be only implicitly cast from/to STRING
INSERT ... VALUES () string literals need to be explicitly cast to BINARY in Impala.	In Hive, no explici t conversion needed

Impala supports a small subset of built-in STRING functions for the BINARY type. If the query option UTF8_MODE is set to FALSE, then the STRING values behave like BINARY values. So you can store binary values as STRINGs and have all built-in functions available for them.	Only a small subset of built-in STRING functions support BINARY.
In textfiles Impala writes BINARY columns base64 encoded	In several file formats (e.g. text) BINARY is base64 encoded.
No NDV is calculated during COMPUTE STATISTICS.	No NDV is calculated during COMPUTE STATISTICS.
All kinds of UDFs (native, Hive legacy, Hive generic) support BINARY	In Hive UDFs is only supported if the argument and return types are set explicitly

Example

The following example demonstrates calls to string manipulation functions to concatenate binary columns.

```
select concat( cast("a" as binary), cast("b" as binary));
```

The following examples demonstrate how to use the aggregate functions. These functions calculate a return value across all the items in a result set, so they require a FROM clause in the query.

```
select count(a) from product_catalog;
```

```
select max(height), avg(height) from census_data where age > 20;
```

BOOLEAN data type

A data type used in CREATE TABLE and ALTER TABLE statements, representing a single true/false choice.

Syntax: In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME BOOLEAN
```

Range: TRUE or FALSE. Do not use quotation marks around the TRUE and FALSE literal values. You can write the literal values in uppercase, lowercase, or mixed case. The values queried from a table are always returned in lowercase, true or false.

Conversions: Impala does not automatically convert any other type to BOOLEAN. All conversions must use an explicit call to the CAST() function.

You can use CAST() to convert any integer or floating-point type to BOOLEAN: a value of 0 represents false, and any non-zero value is converted to true.

```
SELECT CAST(42 AS BOOLEAN) AS nonzero_int, CAST(99.44 AS BOOLEAN) AS nonzero
_decimal,
  CAST(000 AS BOOLEAN) AS zero_int, CAST(0.0 AS BOOLEAN) AS zero_decimal;
+-------------+-----------------+----------+--------------+
| nonzero_int | nonzero_decimal | zero_int | zero_decimal |
+-------------+-----------------+----------+--------------+
| true        | true            | false    | false        |
+-------------+-----------------+----------+--------------+
```

When you cast the opposite way, from BOOLEAN to a numeric type, the result becomes either 1 or 0:

```
SELECT CAST(true AS INT) AS true_int, CAST(true AS DOUBLE) AS true_double,
  CAST(false AS INT) AS false_int, CAST(false AS DOUBLE) AS false_double;
+----------+-------------+-----------+--------------+
| true_int | true_double | false_int | false_double |
```

```
+----------+-------------+-----------+--------------+
| 1        | 1           | 0         | 0            |
+----------+-------------+-----------+--------------+
```

You can cast DECIMAL values to BOOLEAN, with the same treatment of zero and non-zero values as the other numeric types. You cannot cast a BOOLEAN to a DECIMAL.

You cannot cast a STRING value to BOOLEAN, although you can cast a BOOLEAN value to STRING, returning '1' for true values and '0' for false values.

Although you can cast a TIMESTAMP to a BOOLEAN or a BOOLEAN to a TIMESTAMP, the results are unlikely to be useful. Any non-zero TIMESTAMP (that is, any value other than 1970-01-01 00:00:00) becomes TRUE when converted to BOOLEAN, while 1970-01-01 00:00:00 becomes FALSE. A value of FALSE becomes 1970-01-01 00:00:00 when converted to BOOLEAN, and TRUE becomes one second past this epoch date, that is, 1970-01-01 00:00:01.

NULL considerations: An expression of this type produces a NULL value if any argument of the expression is NULL.

Partitioning:

Do not use a BOOLEAN column as a partition key. Although you can create such a table, subsequent operations produce errors:

```
[localhost:21000] > create table truth_table (assertion string) partitioned
by (truth boolean);
[localhost:21000] > insert into truth_table values ('Pigs can fly',false);
ERROR: AnalysisException: INSERT into table with BOOLEAN partition column
 (truth) is not supported: partitioning.truth_table
```

Examples:

```
SELECT 1 < 2;
SELECT 2 = 5;
SELECT 100 < NULL, 100 > NULL;
CREATE TABLE assertions (claim STRING, really BOOLEAN);
INSERT INTO assertions VALUES
  ("1 is less than 2", 1 < 2),
  ("2 is the same as 5", 2 = 5),
  ("Grass is green", true),
  ("The moon is made of green cheese", false);
SELECT claim FROM assertions WHERE really = TRUE;
```

HBase considerations: This data type is fully compatible with HBase tables.

Parquet considerations: This type is fully compatible with Parquet tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Related Information

Impala SQL operators

Impala conditional functions

CHAR data type

A fixed-length character type, padded with trailing spaces if necessary to achieve the specified length. If values are longer than the specified length, Impala truncates any trailing characters.

Performance consideration

Note: For performance consideration, we recommend using VARCHAR or STRING over CHAR as the performance gain of Codegen outweighs the benefits of fixed width CHAR. The CHAR type currently does not have the Impala Codegen support.

Syntax: In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME CHAR(LENGTH)
```

The maximum *LENGTH* you can specify is 255.

Semantics of trailing spaces:

- When you store a CHAR value shorter than the specified length in a table, queries return the value padded with trailing spaces if necessary; the resulting value has the same length as specified in the column definition.
- Leading spaces in CHAR are preserved within the data file.
- If you store a CHAR value containing trailing spaces in a table, those trailing spaces are not stored in the data file. When the value is retrieved by a query, the result could have a different number of trailing spaces. That is, the value includes however many spaces are needed to pad it to the specified length of the column.
- If you compare two CHAR values that differ only in the number of trailing spaces, those values are considered identical.
- When comparing or processing CHAR values:
  - CAST() truncates any longer string to fit within the defined length. For example:

    ```
    SELECT CAST('x' AS CHAR(4)) = CAST('x        ' AS CHAR(4)); -- Returns T
    RUE.
    ```
  - If a CHAR value is shorter than the specified length, it is padded on the right with spaces until it matches the specified length.
  - CHAR_LENGTH() returns the length including any trailing spaces.
  - LENGTH() returns the length excluding trailing spaces.
  - CONCAT() returns the length including trailing spaces.

Partitioning: This type can be used for partition key columns. Because of the efficiency advantage of numeric values over character-based values, if the partition key is a string representation of a number, prefer to use an integer type with sufficient range (INT, BIGINT, and so on) where practical.

HBase considerations: This data type cannot be used with HBase tables.

Parquet considerations:

- This type can be read from and written to Parquet files.
- There is no requirement for a particular level of Parquet.
- Parquet files generated by Impala and containing this type can be freely interchanged with other components such as Hive and MapReduce.
- Any trailing spaces, whether implicitly or explicitly specified, are not written to the Parquet data files.
- Parquet data files might contain values that are longer than allowed by the CHAR(*N*) length limit. Impala ignores any extra trailing characters when it processes those values during a query.

Text table considerations: Text data files might contain values that are longer than allowed for a particular CHAR(*N*) column. Any extra trailing characters are ignored when Impala processes those values during a query. Text data files can also contain values that are shorter than the defined length limit, and Impala pads them with trailing spaces up to

the specified length. Any text data files produced by Impala INSERT statements do not include any trailing blanks for CHAR columns.

Avro considerations: The Avro specification allows string values up to 2**64 bytes in length. Impala queries for Avro tables use 32-bit integers to hold string lengths. In Impala 2.5 and higher, Impala truncates CHAR and VARCHAR values in Avro tables to (2**31)-1 bytes. If a query encounters a STRING value longer than (2**31)-1 bytes in an Avro table, the query fails. In earlier releases, encountering such long values in an Avro table could cause a crash.

Compatibility: This type is available using Impala 2.0.0 or higher.

Some other database systems make the length specification optional. For Impala, the length is required.

Internal details: Represented in memory as a byte array with the same size as the length specification. Values that are shorter than the specified length are padded on the right with trailing spaces.

Added in: Impala 2.0.0

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

UDF considerations: This type cannot be used for the argument or return type of a user-defined function (UDF) or user-defined aggregate function (UDA).

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Restrictions: Because the blank-padding behavior requires allocating the maximum length for each value in memory, for scalability reasons, you should avoid declaring CHAR columns that are much longer than typical values in that column.

All data in CHAR and VARCHAR columns must be in a character encoding that is compatible with UTF-8. If you have binary data from another database system (that is, a BLOB type), use a STRING column to hold it.

When an expression compares a CHAR with a STRING or VARCHAR, the CHAR value is implicitly converted to STRING first, with trailing spaces preserved.

This behavior differs from other popular database systems. To get the expected result of TRUE, cast the expressions on both sides to CHAR values of the appropriate length. For example:

```
SELECT CAST("foo  " AS CHAR(5)) = CAST('foo' AS CHAR(3)); -- Returns TRUE.
```

This behavior is subject to change in future releases.

DATE data type

Use the DATE data type to store date values.

Use the DATE data type to store date values. The DATE type is supported for HBase, Text, Avro, and Parquet.

Range:

0001-01-01 to 9999-12-31

Literals and expressions:

The DATE literals are in the form of DATE'YYYY-MM-DD'. For example, DATE '2013-01-01'

Parquet and Avro considerations:

Parquet and Avro use DATE logical type for dates. The DATE logical type annotates an INT32 that stores the number of days from the Unix epoch, January 1, 1970. This representation introduces an interoperability issue between Impala and older versions of Hive:

If Hive versions lower than 3.1 wrote dates earlier than 1582-10-15 to a Parquet or Avro table, those dates will be read back incorrectly by Impala and vice versa. In Hive 3.1 and higher, this is no longer an issue.

Explicit casting between DATE and other data types:

DATE type can only be converted to/from DATE, TIMESTAMP, or STRING types as described below.

Cast from	Cast to	Result
TIMESTAMP	DATE	The date component of the TIMESTAMP is returned, and the time of the day component of the TIMESTAMP is ignored.
STRING	DATE	The DATE value of yyyy-MM-dd is returned. The STRING value must be in the yyyy-MM-dd or yyyy-MM-dd HH:mm:ss.SSSS SSSSS pattern. If the time component is present in STRING, it is silently ignored. If the STRING value does not match the above formats, an error is returned.
DATE	TIMESTAMP	The year, month, and day of the DATE is returned along with the time of day component set to 00:00:00.
DATE	STRING	The STRING value, 'yyyy-MM-dd', is returned.

Implicit casting between DATE and other types:

Implicit casting is supported:

- From STRING to DATE if the source STRING value is in the yyyy-MM-dd or yyyy-MM-dd HH:mm:ss.SSSS SSSSS pattern.
- From DATE to TIMESTAMP.

Added in:

The DATE type is available in Impala 3.3 and higher.

Kudu considerations:

You can read and write DATE values to Kudu tables. For example:

```
create table kudu_date_key (fdatekey date primary key, val string)
stored as kudu
insert into kudu_date_key values (DATE '1970-01-01', 'Unix epoch'), (DATE
 '2019-12-12', 'today')
```

DECIMAL data type

The DECIMAL data type is a numeric data type with fixed scale and precision. The data type is useful for storing and doing operations on precise decimal values.

Syntax:

```
DECIMAL[(PRECISION[, SCALE])]
```

Precision:

PRECISION represents the total number of digits that can be represented regardless of the location of the decimal point.

This value must be between 1 and 38, specified as an integer literal.

The default precision is 9.

Scale:

SCALE represents the number of fractional digits.

This value must be less than or equal to the precision, specified as an integer literal.

The default scale is 0.

When the precision and the scale are omitted, a DECIMAL is treated as DECIMAL(9, 0).

Range:

The range of DECIMAL type is -10^38 +1 through 10^38 –1.

The largest value is represented by DECIMAL(38, 0).

The most precise fractional value (between 0 and 1, or 0 and -1) is represented by DECIMAL(38, 38), with 38 digits to the right of the decimal point. The value closest to 0 would be .0000...1 (37 zeros and the final 1). The value closest to 1 would be .999... (9 repeated 38 times).

Memory and disk storage:

Only the precision determines the storage size for DECIMAL values, and the scale setting has no effect on the storage size. The following table describes the in-memory storage once the values are loaded into memory.

Precision	In-memory storage
1 - 9	4 bytes
10 - 18	8 bytes
19 - 38	10 bytes

The on-disk representation varies depending on the file format of the table.

Text, RCFile, and SequenceFile tables use ASCII-based formats as below:

- Leading zeros are not stored.
- Trailing zeros are stored.
- Each DECIMAL value takes up as many bytes as the precision of the value, plus:
  - One extra byte if the decimal point is present.
  - One extra byte for negative values.

Parquet and Avro tables use binary formats and offer more compact storage for DECIMAL values. In these tables, Impala stores each value in fewer bytes where possible depending on the precision specified for the DECIMAL column. To conserve space in large tables, use the smallest-precision DECIMAL type.

Precision and scale in arithmetic operations:

For all arithmetic operations, the resulting precision is at most 38.

If the resulting precision would be greater than 38, Impala truncates the result from the back, but keeps at least 6 fractional digits in scale and rounds.

For example, DECIMAL(38, 20) * DECIMAL(38, 20) returns DECIMAL(38, 6). According to the table below, the resulting precision and scale would be (77, 40), but they are higher than the maximum precision and scale for DECI MAL. So, Impala sets the precision to the maximum allowed 38, and truncates the scale to 6.

When you use DECIMAL values in arithmetic operations, the precision and scale of the result value are determined as follows. For better readability, the following terms are used in the table below:

- P1, P2: Input precisions
- S1, S2: Input scales
- L1, L2: Leading digits in input DECIMALs, i.e., L1 = P1 - S1 and L2 = P2 - S2

Operation	Resulting precision	Resulting scale
Addition and Subtraction	max (L1, L2) + max (S1, S2) + 1 1 is for carry-over.	max (S1, S2)
Multiplication	P1 + P2 + 1	S1 + S2
Division	L1 + S2 + max (S1 + P2 + 1, 6)	max (S1 + P2 + 1, 6)
Modulo	min (L1, L2) + max (S1, S2)	max (S1, S2)

Precision and scale in functions:

When you use DECIMAL values in built-in functions, the precision and scale of the result value are determined as follows:

- The result of the SUM aggregate function on a DECIMAL value is:
  - Precision: 38
  - Scale: The same scale as the input column
- The result of AVG aggregate function on a DECIMAL value is:
  - Precision: 38
  - Scale: max(Scale of input column, 6)

Implicit conversions in DECIMAL assignments:

Impala enforces strict conversion rules in decimal assignments like in INSERT and UNION statements, or in functions like COALESCE.

If there is not enough precision and scale in the destination, Impala fails with an error.

Impala performs implicit conversions between DECIMAL and other numeric types as below:

- DECIMAL is implicitly converted to DOUBLE or FLOAT when necessary even with a loss of precision. It can be necessary, for example when inserting a DECIMAL value into a DOUBLE column. For example:

  ```
  CREATE TABLE flt(c FLOAT);
  INSERT INTO flt SELECT CAST(1e37 AS DECIMAL(38, 0));
  SELECT CAST(c AS DECIMAL(38, 0)) FROM flt;

  Result: 99999999338158125107115063762579619 84
  ```

  The result has a loss of information due to implicit casting. This is why we discourage using the DOUBLE and FLOAT types in general.
- DOUBLE and FLOAT cannot be implicitly converted to DECIMAL. An error is returned.
- DECIMAL is implicitly converted to DECIMAL if all digits fit in the resulting DECIMAL.

  For example, the following query returns an error because the resulting type that guarantees that all digits fit cannot be determined .

  ```
  SELECT GREATEST (CAST(1 AS DECIMAL(38, 0)), CAST(2 AS DECIMAL(38, 37)));
  ```

- Integer values can be implicitly converted to DECIMAL when there is enough room in the DECIMAL to guarantee that all digits fit. The integer types require the following numbers of digits to the left of the decimal point when converted to DECIMAL:
  - BIGINT: 19 digits
  - INT: 10 digits
  - SMALLINT: 5 digits
  - TINYINT: 3 digits

  For example:

  ```
  CREATE TABLE decimals_10_8 (x DECIMAL(10, 8));
  INSERT INTO decimals_10_8 VALUES (CAST(1 AS TINYINT));
  ```

  The above INSERT statement fails because TINYINT requires room for 3 digits to the left of the decimal point in the DECIMAL.

  ```
  CREATE TABLE decimals_11_8(x DECIMAL(11, 8));
  INSERT INTO decimals_11_8 VALUES (CAST(1 AS TINYINT));
  ```

  The above INSERT statement succeeds because there is enough room for 3 digits to the left of the decimal point that TINYINT requires.

In UNION, the resulting precision and scales are determined as follows.

- Precision: max (L1, L2) + max (S1, S2)

  If the resulting type does not fit in the DECIMAL type, an error is returned. See the first example below.
- Scale: max (S1, S2)

Examples for UNION:

- DECIMAL(20, 0) UNION DECIMAL(20, 20) would require a DECIMAL(40, 20) to fit all the digits. Since this is larger than the max precision for DECIMAL, Impala returns an error. One way to fix the error is to cast both operands to the desired type, for example DECIMAL(38, 18).
- DECIMAL(20, 2) UNION DECIMAL(8, 6) returns DECIMAL(24, 6).
- INT UNION DECIMAL(9, 4) returns DECIMAL(14, 4).

  INT has the precision 10 and the scale 0, so it is treated as DECIMAL(10, 0) UNION DECIMAL(9. 4).

Casting between DECIMAL and other data types:

To avoid potential conversion errors, use CAST to explicitly convert between DECIMAL and other types in decimal assignments like in INSERT and UNION statements, or in functions like COALESCE:

- You can cast the following types to DECIMAL: FLOAT, TINYINT, SMALLINT, INT, BIGINT, STRING
- You can cast DECIMAL to the following types: FLOAT, TINYINT, SMALLINT, INT, BIGINT, STRING, BOOLEAN, TIMESTAMP

Impala performs CAST between DECIMAL and other numeric types as below:

- Precision: If you cast a value with bigger precision than the precision of the destination type, Impala returns an error. For example, CAST(123456 AS DECIMAL(3,0)) returns an error because all digits do not fit into DECI MAL(3, 0)
- Scale: If you cast a value with more fractional digits than the scale of the destination type, the fractional digits are rounded. For example, CAST(1.239 AS DECIMAL(3, 2)) returns 1.24.

Casting STRING to DECIMAL:

You can cast STRING of numeric characters in columns, literals, or expressions to DECIMAL as long as number fits within the specified target DECIMAL type without overflow.

- If scale in STRING > scale in DECIMAL, the fractional digits are rounded to the DECIMAL scale.

  For example, CAST('98.678912' AS DECIMAL(15, 1)) returns 98.7.
- If # leading digits in STRING > # leading digits in DECIMAL, an error is returned.

  For example, CAST('123.45' AS DECIMAL(2, 2)) returns an error.

Exponential notation is supported when casting from STRING.

For example, CAST('1.0e6' AS DECIMAL(32, 0)) returns 1000000.

Casting any non-numeric value, such as 'ABC' to the DECIMAL type returns an error.

Casting DECIMAL to TIMESTAMP:

Casting a DECIMAL value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970).

DECIMAL vs FLOAT consideration:

The FLOAT and DOUBLE types can cause problems or unexpected behavior due to inability to precisely represent certain fractional values, for example dollar and cents values for currency. You might find output values slightly different than you inserted, equality tests that do not match precisely, or unexpected values for GROUP BY columns. The DECIMAL type can help reduce unexpected behavior and rounding errors, but at the expense of some performance overhead for assignments and comparisons.

Literals and expressions:

- Numeric literals without a decimal point
  - The literals are treated as the smallest integer that would fit the literal. For example, 111 is a TINYINT, and 1111 is a SMALLINT.
  - Large literals that do not fit into any integer type are treated as DECIMAL.
  - The literals too large to fit into a DECIMAL(38, 0) are treated as DOUBLE.
- Numeric literals with a decimal point
  - The literal with less than 38 digits are treated as DECIMAL.
  - The literals with 38 or more digits are treated as a DOUBLE.
- Exponential notation is supported in DECIMAL literals.
- To represent a very large or precise DECIMAL value as a literal, for example one that contains more digits than can be represented by a BIGINT literal, use a quoted string or a floating-point value for the number and CAST the string to the desired DECIMAL type.

  For example: CAST('999999999999999999999999999999' AS DECIMAL(38, 5)))

File format considerations:

The DECIMAL data type can be stored in any of the file formats supported by Impala:

- Impala can query Avro, RCFile, or SequenceFile tables that contain DECIMAL columns, created by other Hadoop components.
- Impala can query and insert into Kudu tables that contain DECIMAL columns. Kudu supports the DECIMAL type.
- The DECIMAL data type is fully compatible with HBase tables.
- The DECIMAL data type is fully compatible with Parquet tables.
- Values of the DECIMAL data type are potentially larger in text tables than in tables using Parquet or other binary formats.

UDF consideration:

When writing a C++ UDF, use the DecimalVal data type defined in /usr/include/impala_udf/udf.h.

Changing precision and scale:

You can issue an ALTER TABLE ... REPLACE COLUMNS statement to change the precision and scale of an existing DECIMAL column.

- For text-based formats (text, RCFile, and SequenceFile tables)
  - If the values in the column fit within the new precision and scale, they are returned correctly by a query.
  - If any values that do not fit within the new precision and scale:
    - Impala returns an error if the query option ABORT_ON_ERROR is set to true.
    - Impala returns a NULL and warning that conversion failed if the query option ABORT_ON_ERROR is set to false.
  - Leading zeros do not count against the precision value, but trailing zeros after the decimal point do.
- For binary formats (Parquet and Avro tables)
  - Although an ALTER TABLE ... REPLACE COLUMNS statement that changes the precision or scale of a DECIMAL column succeeds, any subsequent attempt to query the changed column results in a fatal error. This is because the metadata about the columns is stored in the data files themselves, and ALTER TABLE does not actually make any updates to the data files. The other unaltered columns can still be queried successfully.
  - If the metadata in the data files disagrees with the metadata in the metastore database, Impala cancels the query.

Partitioning:

Using a DECIMAL column as a partition key provides you a better match between the partition key values and the HDFS directory names than using a DOUBLE or FLOAT partitioning column.

Column statistics considerations:

Because the DECIMAL type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Compatibility with older version of DECIMAL:

This version of DECIMAL type is the default in Impala 3.0 and higher. In Impala 2.12 and lower, decimal_v2=false. And in Impala 3.0 or higher, decimal_v2=true. The key differences between this version of DECIMAL and the previous DECIMAL V1 in Impala 2.x include the following:

Property	Impala 3.0 or higher	Impala 2.12 or lower
Overall behavior	Returns either the result or an error.	Returns either the result or NULL with a warning.
Overflow behavior	Aborts with an error.	Issues a warning and returns NULL.
Truncation / rounding behavior in arithmetic	Truncates and rounds digits from the back.	Truncates digits from the front.
String cast	Truncates from the back and rounds.	Truncates from the back.

If you need to continue using the first version of the DECIMAL type for the backward compatibility of your queries, set the DECIMAL_V2 query option to FALSE:

```
SET DECIMAL_V2=FALSE;
```

Compatibility with other databases:

Use the DECIMAL data type in Impala for applications where you used the NUMBER data type in Oracle.

The Impala DECIMAL type does not support the Oracle idioms of * for scale.

The Impala DECIMAL type does not support negative values for precision.

DOUBLE data type

A double precision floating-point data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME DOUBLE
```

Range: 4.94065645841246544e-324d .. 1.79769313486231570e+308, positive or negative

Precision: 15 to 17 significant digits, depending on usage. The number of significant digits does not depend on the position of the decimal point.

Representation: The values are stored in 8 bytes, using the IEEE 754 Double Precision Binary Floating Point format.

Conversions: Impala does not automatically convert DOUBLE to any other type. You can use CAST() to convert DOUBLE values to FLOAT, TINYINT, SMALLINT, INT, BIGINT, STRING, TIMESTAMP, or BOOLEAN. You can use exponential notation in DOUBLE literals or when casting from STRING, for example 1.0e6 to represent one million. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Usage notes:

The data type REAL is an alias for DOUBLE.

Impala does not evaluate NaN (not a number) as equal to any other numeric values, including other NaN values. For example, the following statement, which evaluates equality between two NaN values, returns false:

```
SELECT CAST('nan' AS DOUBLE)=CAST('nan' AS DOUBLE);
```

Examples:

```
CREATE TABLE t1 (x DOUBLE);
SELECT CAST(1000.5 AS DOUBLE);
```

Partitioning: Because fractional values of this type are not always represented precisely, when this type is used for a partition key column, the underlying HDFS directories might not be named exactly as you expect. Prefer to partition on a DECIMAL column instead.

HBase considerations: This data type is fully compatible with HBase tables.

Parquet considerations: This type is fully compatible with Parquet tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as an 8-byte value.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Restrictions:

Due to the way arithmetic on FLOAT and DOUBLE columns uses high-performance hardware instructions, and distributed queries can perform these operations in different order for each query, results can vary slightly for aggregate function calls such as SUM() and AVG() for FLOAT and DOUBLE columns, particularly on large data sets where millions or billions of values are summed or averaged. For perfect consistency and repeatability, use the DECIMAL data type for such operations instead of FLOAT or DOUBLE.

The inability to exactly represent certain floating-point values means that DECIMAL is sometimes a better choice than DOUBLE or FLOAT when precision is critical, particularly when transferring data from other database systems that use different representations or file formats.

Kudu considerations:

Currently, the data types BOOLEAN, FLOAT, and DOUBLE cannot be used for primary key columns in Kudu tables.

Related Information

Impala mathematical functions
FLOAT data type
Impala SQL literals

FLOAT data type

A single precision floating-point data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME FLOAT
```

Range: 1.40129846432481707e-45 .. 3.40282346638528860e+38, positive or negative

Precision: 6 to 9 significant digits, depending on usage. The number of significant digits does not depend on the position of the decimal point.

Representation: The values are stored in 4 bytes, using the *IEEE 754 Single Precision Binary Floating Point* format.

Conversions: Impala automatically converts FLOAT to more precise DOUBLE values, but not the other way around. You can use CAST() to convert FLOAT values to TINYINT, SMALLINT, INT, BIGINT, STRING, TIMESTAMP, or BOOLEAN. You can use exponential notation in FLOAT literals or when casting from STRING, for example 1.0e6 to represent one million. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Usage notes:

Impala does not evaluate NaN (not a number) as equal to any other numeric values, including other NaN values. For example, the following statement, which evaluates equality between two NaN values, returns false:

```
SELECT CAST('nan' AS FLOAT)=CAST('nan' AS FLOAT);
```

Examples:

```
CREATE TABLE t1 (x FLOAT);
SELECT CAST(1000.5 AS FLOAT);
```

Partitioning: Because fractional values of this type are not always represented precisely, when this type is used for a partition key column, the underlying HDFS directories might not be named exactly as you expect. Prefer to partition on a DECIMAL column instead.

HBase considerations: This data type is fully compatible with HBase tables.

Parquet considerations: This type is fully compatible with Parquet tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as a 4-byte value.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Restrictions:

Due to the way arithmetic on FLOAT and DOUBLE columns uses high-performance hardware instructions, and distributed queries can perform these operations in different order for each query, results can vary slightly for aggregate function calls such as SUM() and AVG() for FLOAT and DOUBLE columns, particularly on large data sets where millions or billions of values are summed or averaged. For perfect consistency and repeatability, use the DECIMAL data type for such operations instead of FLOAT or DOUBLE.

The inability to exactly represent certain floating-point values means that DECIMAL is sometimes a better choice than DOUBLE or FLOAT when precision is critical, particularly when transferring data from other database systems that use different representations or file formats.

Kudu considerations:

Currently, the data types BOOLEAN, FLOAT, and DOUBLE cannot be used for primary key columns in Kudu tables.

Related Information

IEEE 754 Single Precision Binary Floating Point

Impala mathematical functions

Impala SQL literals

INT data type

A 4-byte integer data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME INT
```

Range: -2147483648 .. 2147483647. There is no UNSIGNED subtype.

Conversions: Impala automatically converts to a larger integer type (BIGINT) or a floating-point type (FLOAT or DOUBLE) automatically. Use CAST() to convert to TINYINT, SMALLINT, STRING, or TIMESTAMP. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Usage notes:

The data type INTEGER is an alias for INT.

For a convenient and automated way to check the bounds of the INT type, call the functions MIN_INT() and MAX_INT().

If an integer value is too large to be represented as a INT, use a BIGINT instead.

NULL considerations: Casting any non-numeric value to this type produces a NULL value.

Examples:

```
CREATE TABLE t1 (x INT);
SELECT CAST(1000 AS INT);
```

Partitioning: Prefer to use this type for a partition key column. Impala can process the numeric type more efficiently than a STRING representation of the value.

HBase considerations: This data type is fully compatible with HBase tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as a 4-byte value.

Added in: Available in all versions of Impala.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

MAP complex type

A complex data type representing an arbitrary set of key-value pairs. The key part is a scalar type, while the value part can be a scalar or another complex type (ARRAY, STRUCT, or MAP).

Syntax:

```
COLUMN_NAME MAP < PRIMITIVE_TYPE, TYPE >

type ::= PRIMITIVE_TYPE | COMPLEX_TYPE
```

Usage notes:

Because complex types are often used in combination, for example an ARRAY of STRUCT elements, if you are unfamiliar with the Impala complex types, start with Complex types for background information and usage examples.

The MAP complex data type represents a set of key-value pairs. Each element of the map is indexed by a primitive type such as BIGINT or STRING, letting you define sequences that are not continuous or categories with arbitrary names. You might find it convenient for modelling data produced in other languages, such as a Python dictionary or Java HashMap, where a single scalar value serves as the lookup key.

In a big data context, the keys in a map column might represent a numeric sequence of events during a manufacturing process, or TIMESTAMP values corresponding to sensor observations. The map itself is inherently unordered, so you choose whether to make the key values significant (such as a recorded TIMESTAMP) or synthetic (such as a random global universal ID).

Note: Behind the scenes, the MAP type is implemented in a similar way as the ARRAY type. Impala does not enforce any uniqueness constraint on the KEY values, and the KEY values are processed by looping through the elements of the MAP rather than by a constant-time lookup. Therefore, this type is primarily for ease of understanding when importing data and algorithms from non-SQL contexts, rather than optimizing the performance of key lookups.

You can pass a multi-part qualified name to DESCRIBE to specify an ARRAY, STRUCT, or MAP column and visualize its structure as if it were a table. For example, if table T1 contains an ARRAY column A1, you could issue the statement DESCRIBE t1.a1. If table T1 contained a STRUCT column S1, and a field F1 within the STRUCT was a MAP, you could issue the statement DESCRIBE t1.s1.f1. An ARRAY is shown as a two-column table, with ITEM and POS columns. A STRUCT is shown as a table with each field representing a column in the table. A MAP is shown as a two-column table, with KEY and VALUE columns.

Added in: Impala 2.3.0

Restrictions:

- Columns with this data type can only be used in tables or partitions with the Parquet file format.
- Columns with this data type cannot be used as partition key columns in a partitioned table.
- The COMPUTE STATS statement does not produce any statistics for columns of this data type.
- The maximum length of the column definition for any complex type, including declarations for any nested types, is 4000 characters.
- See the *Limitations and restrictions for complex types* topic for a full list of limitations and associated guidelines about complex type columns.

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Examples:

Note: Many of the complex type examples refer to tables such as CUSTOMER and REGION adapted from the tables used in the TPC-H benchmark.

The following example shows a table with various kinds of MAP columns, both at the top level and nested within other complex types. Each row represents information about a specific country, with complex type fields of various levels of nesting to represent different information associated with the country: factual measurements such as area and population, notable people in different categories, geographic features such as cities, points of interest within each city, and mountains with associated facts. Practice the CREATE TABLE and query notation for complex type columns using empty tables, until you can visualize a complex data structure and construct corresponding SQL statements reliably.

```
create TABLE map_demo
(
  country_id BIGINT,

-- Numeric facts about each country, looked up by name.
-- For example, 'Area':1000, 'Population':999999.
-- Using a MAP instead of a STRUCT because there could be
-- a different set of facts for each country.
  metrics MAP <STRING, BIGINT>,
-- MAP whose value part is an ARRAY.
-- For example, the key 'Famous Politicians' could represent an array of 10
 elements,
-- while the key 'Famous Actors' could represent an array of 20 elements.
  notables MAP <STRING, ARRAY <STRING>>,

-- MAP that is a field within a STRUCT.
-- (The STRUCT is inside another ARRAY, because it is rare
-- for a STRUCT to be a top-level column.)
-- For example, city #1 might have points of interest with key 'Zoo',
-- representing an array of 3 different zoos.
-- City #2 might have completely different kinds of points of interest.
-- Because the set of field names is potentially large, and most entries c
ould be blank,
-- a MAP makes more sense than a STRUCT to represent such a sparse data s
tructure.
  cities ARRAY < STRUCT <
    name: STRING,
    points_of_interest: MAP <STRING, ARRAY <STRING>>
  >>,

-- MAP that is an element within an ARRAY. The MAP is inside a STRUCT field
 to associate
-- the mountain name with all the facts about the mountain.
-- The "key" of the map (the first STRING field) represents the name of s
ome fact whose value
-- can be expressed as an integer, such as 'Height', 'Year First Climbed',
and so on.
  mountains ARRAY < STRUCT < name: STRING, facts: MAP <STRING, INT > > >
)
STORED AS PARQUET;
```

```
DESCRIBE map_demo;
+------------+--------------------------------------------------+
| name       | type                                             |
+------------+--------------------------------------------------+
| country_id | bigint                                           |
```

```
| metrics    | map<string,bigint>                                |
| notables   | map<string,array<string>>                         |
| cities     | array<struct<                                     |
|            |   name:string,                                    |
|            |   points_of_interest:map<string,array<string>> |
|            | >>                                                |
| mountains  | array<struct<                                     |
|            |   name:string,                                    |
|            |   facts:map<string,int>                           |
|            | >>                                                |
+------------+---------------------------------------------------+

DESCRIBE map_demo.metrics;
+-------+--------+
| name  | type   |
+-------+--------+
| key   | string |
| value | bigint |
+-------+--------+

DESCRIBE map_demo.notables;
+-------+---------------+
| name  | type          |
+-------+---------------+
| key   | string        |
| value | array<string> |
+-------+---------------+

DESCRIBE map_demo.notables.value;
+------+--------+
| name | type   |
+------+--------+
| item | string |
| pos  | bigint |
+------+--------+

DESCRIBE map_demo.cities;
+------+-------------------------------------------------+
| name | type                                            |
+------+-------------------------------------------------+
| item | struct<                                         |
|      |   name:string,                                  |
|      |   points_of_interest:map<string,array<string>> |
|      | >                                               |
| pos  | bigint                                          |
+------+-------------------------------------------------+

DESCRIBE map_demo.cities.item.points_of_interest;
+-------+---------------+
| name  | type          |
+-------+---------------+
| key   | string        |
| value | array<string> |
+-------+---------------+

DESCRIBE map_demo.cities.item.points_of_interest.value;
+------+--------+
| name | type   |
+------+--------+
| item | string |
| pos  | bigint |
+------+--------+

DESCRIBE map_demo.mountains;
```

```
+------+-------------------------+
| name | type                    |
+------+-------------------------+
| item | struct<                 |
|      |   name:string,          |
|      |   facts:map<string,int> |
|      | >                       |
| pos  | bigint                  |
+------+-------------------------+

DESCRIBE map_demo.mountains.item.facts;
+-------+--------+
| name  | type   |
+-------+--------+
| key   | string |
| value | int    |
+-------+--------+
```

The following example shows a table that uses a variety of data types for the MAP “key” field. Typically, you use BIGINT or STRING to use numeric or character-based keys without worrying about exceeding any size or length constraints.

```
CREATE TABLE map_demo_obscure
(
  id BIGINT,
  m1 MAP <INT, INT>,
  m2 MAP <SMALLINT, INT>,
  m3 MAP <TINYINT, INT>,
  m4 MAP <TIMESTAMP, INT>,
  m5 MAP <BOOLEAN, INT>,
  m6 MAP <CHAR(5), INT>,
  m7 MAP <VARCHAR(25), INT>,
  m8 MAP <FLOAT, INT>,
  m9 MAP <DOUBLE, INT>,
  m10 MAP <DECIMAL(12,2), INT>
)
STORED AS PARQUET;
```

```
CREATE TABLE celebrities (name STRING, birth_year MAP < STRING, SMALLINT >)
STORED AS PARQUET;
-- A typical row might represent values with 2 different birth years, such
 as:
-- ("Joe Movie Star", { "real": 1972, "claimed": 1977 })

CREATE TABLE countries (name STRING, famous_leaders MAP < INT, STRING >) STO
RED AS PARQUET;
-- A typical row might represent values with different leaders, with key val
ues corresponding to their numeric sequence, such as:
-- ("United States", { 1: "George Washington", 3: "Thomas Jefferson", 16:
"Abraham Lincoln" })

CREATE TABLE airlines (name STRING, special_meals MAP < STRING, MAP < STRI
NG, STRING > >) STORED AS PARQUET;
-- A typical row might represent values with multiple kinds of meals, each
 with several components:
-- ("Elegant Airlines",
--   {
--     "vegetarian": { "breakfast": "pancakes", "snack": "cookies", "dinne
r": "rice pilaf" },
```

```
--     "gluten free": { "breakfast": "oatmeal", "snack": "fruit", "dinner":
"chicken" }
--   } )
```

REAL data type

An alias for the DOUBLE data type.

Examples:

These examples show how you can use the type names REAL and DOUBLE interchangeably, and behind the scenes Impala treats them always as DOUBLE.

```
[localhost:21000] > create table r1 (x real);
[localhost:21000] > describe r1;
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| x    | double |         |
+------+--------+---------+
[localhost:21000] > insert into r1 values (1.5), (cast (2.2 as double));
[localhost:21000] > select cast (1e6 as real);
+---------------------------+
| cast(1000000.0 as double) |
+---------------------------+
| 1000000                   |
+---------------------------+
```

Related Information

DOUBLE data type

SMALLINT data type

A 2-byte integer data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME SMALLINT
```

Range: -32768 .. 32767. There is no UNSIGNED subtype.

Conversions: Impala automatically converts to a larger integer type (INT or BIGINT) or a floating-point type (FLOAT or DOUBLE) automatically. Use CAST() to convert to TINYINT, STRING, or TIMESTAMP. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Usage notes:

For a convenient and automated way to check the bounds of the SMALLINT type, call the functions MIN_SMALLINT() and MAX_SMALLINT().

If an integer value is too large to be represented as a SMALLINT, use an INT instead.

NULL considerations: Casting any non-numeric value to this type produces a NULL value.

Examples:

```
CREATE TABLE t1 (x SMALLINT);
SELECT CAST(1000 AS SMALLINT);
```

Parquet considerations:

Physically, Parquet files represent TINYINT and SMALLINT values as 32-bit integers. Although Impala rejects attempts to insert out-of-range values into such columns, if you create a new table with the CREATE TABLE ... LIKE PARQUET syntax, any TINYINT or SMALLINT columns in the original table turn into INT columns in the new table.

Partitioning: Prefer to use this type for a partition key column. Impala can process the numeric type more efficiently than a STRING representation of the value.

HBase considerations: This data type is fully compatible with HBase tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as a 2-byte value.

Added in: Available in all versions of Impala.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

STRING data type

A data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE and ALTER TABLE statements:

```
COLUMN_NAME STRING
```

Length:

If you need to manipulate string values with precise or maximum lengths, in Impala 2.0 and higher you can declare columns as VARCHAR(*MAX_LENGTH*) or CHAR(*LENGTH*), but for best performance use STRING where practical.

Take the following considerations for STRING lengths:

- The hard limit on the size of a STRING and the total size of a row is 2 GB.

  If a query tries to process or create a string larger than this limit, it will return an error to the user.
- The limit is 1 GB on STRING when writing to Parquet files.
- Queries operating on strings with 32 KB or less will work reliably and will not cause significant performance or memory problems (unless you have very complex queries, very many columns, etc.)
- Performance and memory consumption may degrade with strings larger than 32 KB.

- The row size, i.e. the total size of all string and other columns, is subject to lower limits at various points in query execution that support spill-to-disk. A few examples for lower row size limits are:
  - Rows coming from the right side of any hash join
  - Rows coming from either side of a hash join that spills to disk
  - Rows being sorted by the SORT operator without a limit
  - Rows in a grouping aggregation

  The max row size is configurable on a per-query basis with the MAX_ROW_SIZE query option. Rows up to MAX_ROW_SIZE (which defaults to 512 KB) can always be processed in the above cases. Rows larger than MAX_ROW_SIZE are processed on a best-effort basis.

Character sets:

For full support in all Impala subsystems, restrict string values to the ASCII character set. Although some UTF-8 character data can be stored in Impala and retrieved through queries, UTF-8 strings containing non-ASCII characters are not guaranteed to work properly in combination with many SQL aspects, including but not limited to:

- String manipulation functions.
- Comparison operators.
- The ORDER BY clause.
- Values in partition key columns.

For any national language aspects such as collation order or interpreting extended ASCII variants such as ISO-8859-1 or ISO-8859-2 encodings, Impala does not include such metadata with the table definition. If you need to sort, manipulate, or display data depending on those national language characteristics of string data, use logic on the application side.

Conversions:

- Impala does not automatically convert STRING to any numeric type. Impala does automatically convert STRING to TIMESTAMP if the value matches one of the accepted TIMESTAMP formats.
- You can use CAST() to convert STRING values to TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, or TIMESTAMP.
- You cannot directly cast a STRING value to BOOLEAN. You can use a CASE expression to evaluate string values such as 'T', 'true', and so on and return Boolean true and false values as appropriate.
- You can cast a BOOLEAN value to STRING, returning '1' for true values and '0' for false values.

Partitioning:

Although it might be convenient to use STRING columns for partition keys, even when those columns contain numbers, for performance and scalability it is much better to use numeric columns as partition keys whenever practical. Although the underlying HDFS directory name might be the same in either case, the in-memory storage for the partition key columns is more compact, and computations are faster, if partition key columns such as YEAR, MONTH, DAY and so on are declared as INT, SMALLINT, and so on.

Zero-length strings: For purposes of clauses such as DISTINCT and GROUP BY, Impala considers zero-length strings (""), NULL, and space to all be different values.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Avro considerations: The Avro specification allows string values up to 2**64 bytes in length. Impala queries for Avro tables use 32-bit integers to hold string lengths. In Impala 2.5 and higher, Impala truncates CHAR and VARCHAR values in Avro tables to (2**31)-1 bytes. If a query encounters a STRING value longer than (2**31)-1 bytes in an Avro table, the query fails. In earlier releases, encountering such long values in an Avro table could cause a crash.

Column statistics considerations: Because the values of this type have variable size, none of the column statistics fields are filled in until you run the COMPUTE STATS statement.

Examples:

The following examples demonstrate double-quoted and single-quoted string literals, and required escaping for quotation marks within string literals:

```
SELECT 'I am a single-quoted string';
SELECT "I am a double-quoted string";
SELECT 'I\'m a single-quoted string with an apostrophe';
SELECT "I\'m a double-quoted string with an apostrophe";
SELECT 'I am a "short" single-quoted string containing quotes';
SELECT "I am a \"short\" double-quoted string containing quotes";
```

The following examples demonstrate calls to string manipulation functions to concatenate strings, convert numbers to strings, or pull out substrings:

```
SELECT CONCAT("Once upon a time, there were ", CAST(3 AS STRING), ' little p
igs.');
SELECT SUBSTR("hello world",7,5);
```

The following examples show how to perform operations on STRING columns within a table:

```
CREATE TABLE t1 (s1 STRING, s2 STRING);
INSERT INTO t1 VALUES ("hello", 'world'), (CAST(7 AS STRING), "wonders");
SELECT s1, s2, length(s1) FROM t1 WHERE s2 LIKE 'w%';
```

Related Information

MAX_ROW_SIZE query option

TIMESTAMP data type

STRUCT complex type

A complex data type, representing multiple fields of a single item. Frequently used as the element type of an ARRAY or the VALUE part of a MAP.

Syntax:

```
COLUMN_NAME STRUCT < NAME : TYPE [COMMENT 'COMMENT_STRING'], ... >

type ::= PRIMITIVE_TYPE | COMPLEX_TYPE
```

The names and number of fields within the STRUCT are fixed. Each field can be a different type. A field within a STRUCT can also be another STRUCT, or an ARRAY or a MAP, allowing you to create nested data structures with a maximum nesting depth of 100.

A STRUCT can be the top-level type for a column, or can itself be an item within an ARRAY or the value part of the key-value pair in a MAP.

When a STRUCT is used as an ARRAY element or a MAP value, you use a join clause to bring the ARRAY or MAP elements into the result set, and then refer to *ARRAY_NAME*.ITEM.*FIELD* or *MAP_NAME*.VALUE.*FIELD*. In the case of a STRUCT directly inside an ARRAY or MAP, you can omit the .ITEM and .VALUE pseudocolumns and refer directly to *ARRAY_NAME.FIELD* or *MAP_NAME.FIELD*.

Usage notes:

Because complex types are often used in combination, for example an ARRAY of STRUCT elements, if you are unfamiliar with the Impala complex types, start with Complex types for background information and usage examples.

A STRUCT is similar conceptually to a table row: it contains a fixed number of named fields, each with a predefined type. To combine two related tables, while using complex types to minimize repetition, the typical way to represent that data is as an ARRAY of STRUCT elements.

Because a STRUCT has a fixed number of named fields, it typically does not make sense to have a STRUCT as the type of a table column. In such a case, you could just make each field of the STRUCT into a separate column of the table. The STRUCT type is most useful as an item of an ARRAY or the value part of the key-value pair in a MAP. A nested type column with a STRUCT at the lowest level lets you associate a variable number of row-like objects with each row of the table.

The STRUCT type is straightforward to reference within a query. You do not need to include the STRUCT column in a join clause or give it a table alias, as is required for the ARRAY and MAP types. You refer to the individual fields using dot notation, such as *STRUCT_COLUMN_NAME.FIELD_NAME*, without any pseudocolumn such as ITEM or VALUE.

You can pass a multi-part qualified name to DESCRIBE to specify an ARRAY, STRUCT, or MAP column and visualize its structure as if it were a table. For example, if table T1 contains an ARRAY column A1, you could issue the statement DESCRIBE t1.a1. If table T1 contained a STRUCT column S1, and a field F1 within the STRUCT was a MAP, you could issue the statement DESCRIBE t1.s1.f1. An ARRAY is shown as a two-column table, with ITEM and POS columns. A STRUCT is shown as a table with each field representing a column in the table. A MAP is shown as a two-column table, with KEY and VALUE columns.

Internal details:

Within the Parquet data file, the values for each STRUCT field are stored adjacent to each other, so that they can be encoded and compressed using all the Parquet techniques for storing sets of similar or repeated values. The adjacency applies even when the STRUCT values are part of an ARRAY or MAP. During a query, Impala avoids unnecessary I/O by reading only the portions of the Parquet data file containing the requested STRUCT fields.

Added in: Impala 2.3.0

Restrictions:

- Columns with this data type can only be used in tables or partitions with the Parquet file format.
- Columns with this data type cannot be used as partition key columns in a partitioned table.
- The COMPUTE STATS statement does not produce any statistics for columns of this data type.
- The maximum length of the column definition for any complex type, including declarations for any nested types, is 4000 characters.
- See the *Limitations and restrictions for complex types* topic for a full list of limitations and associated guidelines about complex type columns.

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Examples:

Note: Many of the complex type examples refer to tables such as CUSTOMER and REGION adapted from the tables used in the TPC-H benchmark.

The following example shows a table with various kinds of STRUCT columns, both at the top level and nested within other complex types. Practice the CREATE TABLE and query notation for complex type columns using empty tables, until you can visualize a complex data structure and construct corresponding SQL statements reliably.

```
CREATE TABLE struct_demo
(
  id BIGINT,
  name STRING,

-- A STRUCT as a top-level column. Demonstrates how the table ID column
-- and the ID field within the STRUCT can coexist without a name conflict.
  employee_info STRUCT < employer: STRING, id: BIGINT, address: STRING >,

-- A STRUCT as the element type of an ARRAY.
  places_lived ARRAY < STRUCT <street: STRING, city: STRING, country: STR
ING >>,
-- A STRUCT as the value portion of the key-value pairs in a MAP.
```

```
  memorable_moments MAP < STRING, STRUCT < year: INT, place: STRING, deta
ils: STRING >>,
-- A STRUCT where one of the fields is another STRUCT.
  current_address STRUCT < street_address: STRUCT <street_number: INT, stree
t_name: STRING, street_type: STRING>, country: STRING, postal_code: STRING >
)
STORED AS PARQUET;
```

The following example shows how to examine the structure of a table containing one or more STRUCT columns by using the DESCRIBE statement. You can visualize each STRUCT as its own table, with columns named the same as each field of the STRUCT. If the STRUCT is nested inside another complex type, such as ARRAY, you can extend the qualified name passed to DESCRIBE until the output shows just the STRUCT fields.

```
DESCRIBE struct_demo;
+-------------------+--------------------------+
| name              | type                     |
+-------------------+--------------------------+
| id                | bigint                   |
| name              | string                   |
| employee_info     | struct<                  |
|                   |   employer:string,       |
|                   |   id:bigint,             |
|                   |   address:string         |
|                   | >                        |
| places_lived      | array<struct<            |
|                   |   street:string,         |
|                   |   city:string,           |
|                   |   country:string         |
|                   | >>                       |
| memorable_moments | map<string,struct<       |
|                   |   year:int,              |
|                   |   place:string,          |
|                   |   details:string         |
|                   | >>                       |
| current_address   | struct<                  |
|                   |   street_address:struct< |
|                   |     street_number:int,   |
|                   |     street_name:string,  |
|                   |     street_type:string   |
|                   |   >,                     |
|                   |   country:string,        |
|                   |   postal_code:string     |
|                   | >                        |
+-------------------+--------------------------+
```

The top-level column EMPLOYEE_INFO is a STRUCT. Describing *TABLE_NAME.STRUCT_NAME* displays the fields of the STRUCT as if they were columns of a table:

```
DESCRIBE struct_demo.employee_info;
+----------+--------+
| name     | type   |
+----------+--------+
| employer | string |
| id       | bigint |
| address  | string |
+----------+--------+
```

Because PLACES_LIVED is a STRUCT inside an ARRAY, the initial DESCRIBE shows the structure of the ARRAY:

```
DESCRIBE struct_demo.places_lived;
```

```
+------+------------------+
| name | type             |
+------+------------------+
| item | struct<          |
|      |   street:string, |
|      |   city:string,   |
|      |   country:string |
|      | >                |
| pos  | bigint           |
+------+------------------+
```

Ask for the details of the ITEM field of the ARRAY to see just the layout of the STRUCT:

```
DESCRIBE struct_demo.places_lived.item;
+---------+--------+
| name    | type   |
+---------+--------+
| street  | string |
| city    | string |
| country | string |
+---------+--------+
```

Likewise, MEMORABLE_MOMENTS has a STRUCT inside a MAP, which requires an extra level of qualified name to see just the STRUCT part:

```
DESCRIBE struct_demo.memorable_moments;
+-------+------------------+
| name  | type             |
+-------+------------------+
| key   | string           |
| value | struct<          |
|       |   year:int,      |
|       |   place:string,  |
|       |   details:string |
|       | >                |
+-------+------------------+
```

For a MAP, ask to see the VALUE field to see the corresponding STRUCT fields in a table-like structure:

```
DESCRIBE struct_demo.memorable_moments.value;
+---------+--------+
| name    | type   |
+---------+--------+
| year    | int    |
| place   | string |
| details | string |
+---------+--------+
```

For a STRUCT inside a STRUCT, we can see the fields of the outer STRUCT:

```
DESCRIBE struct_demo.current_address;
+----------------+-----------------------+
| name           | type                  |
+----------------+-----------------------+
| street_address | struct<               |
|                |   street_number:int,  |
|                |   street_name:string, |
|                |   street_type:string  |
|                | >                     |
| country        | string                |
| postal_code    | string                |
```

```
+----------------+-----------------------+
```

Then we can use a further qualified name to see just the fields of the inner STRUCT:

```
DESCRIBE struct_demo.current_address.street_address;
+---------------+--------+
| name          | type   |
+---------------+--------+
| street_number | int    |
| street_name   | string |
| street_type   | string |
+---------------+--------+
```

The following example shows how to examine the structure of a table containing one or more STRUCT columns by using the DESCRIBE statement. You can visualize each STRUCT as its own table, with columns named the same as each field of the STRUCT. If the STRUCT is nested inside another complex type, such as ARRAY, you can extend the qualified name passed to DESCRIBE until the output shows just the STRUCT fields.

```
DESCRIBE struct_demo;
+-------------------+--------------------------+---------+
| name              | type                     | comment |
+-------------------+--------------------------+---------+
| id                | bigint                   |         |
| name              | string                   |         |
| employee_info     | struct<                  |         |
|                   |   employer:string,       |         |
|                   |   id:bigint,             |         |
|                   |   address:string         |         |
|                   | >                        |         |
| places_lived      | array<struct<            |         |
|                   |   street:string,         |         |
|                   |   city:string,           |         |
|                   |   country:string         |         |
|                   | >>                       |         |
| memorable_moments | map<string,struct<       |         |
|                   |   year:int,              |         |
|                   |   place:string,          |         |
|                   |   details:string         |         |
|                   | >>                       |         |
| current_address   | struct<                  |         |
|                   |   street_address:struct< |         |
|                   |     street_number:int,   |         |
|                   |     street_name:string,  |         |
|                   |     street_type:string   |         |
|                   |   >,                     |         |
|                   |   country:string,        |         |
|                   |   postal_code:string     |         |
|                   | >                        |         |
+-------------------+--------------------------+---------+
SELECT id, employee_info.id FROM struct_demo;

SELECT id, employee_info.id AS employee_id FROM struct_demo;

SELECT id, employee_info.id AS employee_id, employee_info.employer
  FROM struct_demo;

SELECT id, name, street, city, country
  FROM struct_demo, struct_demo.places_lived;

SELECT id, name, places_lived.pos, places_lived.street, places_lived.city, p
laces_lived.country
  FROM struct_demo, struct_demo.places_lived;
```

```
SELECT id, name, pl.pos, pl.street, pl.city, pl.country
  FROM struct_demo, struct_demo.places_lived AS pl;

SELECT id, name, places_lived.pos, places_lived.street, places_lived.city, p
laces_lived.country
  FROM struct_demo, struct_demo.places_lived;

SELECT id, name, pos, street, city, country
  FROM struct_demo, struct_demo.places_lived;

SELECT id, name, memorable_moments.key,
  memorable_moments.value.year,
  memorable_moments.value.place,
  memorable_moments.value.details
FROM struct_demo, struct_demo.memorable_moments
WHERE memorable_moments.key IN ('Birthday','Anniversary','Graduation');

SELECT id, name, mm.key, mm.value.year, mm.value.place, mm.value.details
  FROM struct_demo, struct_demo.memorable_moments AS mm
WHERE mm.key IN ('Birthday','Anniversary','Graduation');
SELECT id, name, memorable_moments.key, memorable_moments.value.year,
  memorable_moments.value.place, memorable_moments.value.details
FROM struct_demo, struct_demo.memorable_moments
WHERE key IN ('Birthday','Anniversary','Graduation');

SELECT id, name, key, value.year, value.place, value.details
  FROM struct_demo, struct_demo.memorable_moments
WHERE key IN ('Birthday','Anniversary','Graduation');
SELECT id, name, key, year, place, details
  FROM struct_demo, struct_demo.memorable_moments
WHERE key IN ('Birthday','Anniversary','Graduation');
SELECT id, name,
  current_address.street_address.street_number,
  current_address.street_address.street_name,
  current_address.street_address.street_type,
  current_address.country,
  current_address.postal_code
FROM struct_demo;
```

For example, this table uses a struct that encodes several data values for each phone number associated with a person. Each person can have a variable-length array of associated phone numbers, and queries can refer to the category field to locate specific home, work, mobile, and so on kinds of phone numbers.

```
CREATE TABLE contact_info_many_structs
(
  id BIGINT, name STRING,
  phone_numbers ARRAY < STRUCT <category:STRING, country_code:STRING, are
a_code:SMALLINT, full_number:STRING, mobile:BOOLEAN, carrier:STRING > >
) STORED AS PARQUET;
```

Because structs are naturally suited to composite values where the fields have different data types, you might use them to decompose things such as addresses:

```
CREATE TABLE contact_info_detailed_address
(
  id BIGINT, name STRING,
  address STRUCT < house_number:INT, street:STRING, street_type:STRING, a
partment:STRING, city:STRING, region:STRING, country:STRING >
);
```

In a big data context, splitting out data fields such as the number part of the address and the street name could let you do analysis on each field independently. For example, which streets have the largest number range of addresses, what are the statistical properties of the street names, which areas have a higher proportion of “Roads”, “Courts” or “Boulevards”, and so on.

TIMESTAMP data type

In Impala, the TIMESTAMP data type holds a value of date and time. It can be decomposed into year, month, day, hour, minute and seconds fields, but with no time zone information available, it does not correspond to any specific point in time.

Internally, the resolution of the time portion of a TIMESTAMP value is in nanoseconds.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME TIMESTAMP
```

```
TIMESTAMP [+ | -] INTERVAL INTERVAL
DATE_ADD (TIMESTAMP, INTERVAL INTERVAL TIME_UNIT)
```

Range: 1400-01-01 to 9999-12-31

Out of range TIMESTAMP values are converted to NULL.

The range of Impala TIMESTAMP is different from the Hive TIMESTAMP type.

INTERVAL expressions:

You can perform date arithmetic by adding or subtracting a specified number of time units, using the INTERVAL keyword and the + operator, the - operator, date_add() or date_sub().

The following units are supported for time_unit in the INTERVAL clause:

- YEAR[S]
- MONTH[S]
- WEEK[S]
- DAY[S]
- HOUR[S]
- MINUTE[S]
- SECOND[S]
- MILLISECOND[S]
- MICROSECOND[S]
- NANOSECOND[S]

You can only specify one time unit in each interval expression, for example INTERVAL 3 DAYS or INTERVAL 25 HOURS, but you can produce any granularity by adding together successive INTERVAL values, such as *TIMESTAMP_VALUE* + INTERVAL 3 WEEKS - INTERVAL 1 DAY + INTERVAL 10 MICROSECONDS.

Internal details: Represented in memory as a 16-byte value.

Time zones:

By default, Impala stores and interprets TIMESTAMP values in UTC time zone when writing to data files, reading from data files, or converting to and from system time values through functions.

When you set the --use_local_tz_for_unix_timestamp_conversions startup flag to TRUE, Impala treats the TIME STAMP values specified in the local time zone. The local time zone is determined in the following order with the TIMESTAMP query option takes the highest precedence:

1. The TIMESTAMP query option
2. $TZ environment variable
3. System time zone where the impalad coordinator runs

The --use_local_tz_for_unix_timestamp_conversions setting can be used to fix discrepancy in INTERVAL operations. For example, a TIMESTAMP + INTERVAL *N-HOURS* can be affected by Daylight Saving Time, which Impala does not consider by default as these operations are applied as if the timestamp was in UTC. You can use the --use_local_tz_for_unix_timestamp_conversions setting to fix the issue.

See Customizing time zones on page 50 for configuring to use custom time zone database and aliases.

See Impala date and time functions for the list of functions affected by the --use_local_tz_for_unix_timestamp_co nversions setting.

Time zone handling between Impala and Hive:

Interoperability between Hive and Impala is different depending on the file format.

- Text

  For text tables, TIMESTAMP values can be written and read interchangeably by Impala and Hive as Hive reads and writes TIMESTAMP values without converting with respect to time zones.
- Parquet

  Note: This section only applies to INT96 TIMESTAMP.

  When Hive writes to Parquet data files, the TIMESTAMP values are normalized to UTC from the local time zone of the host where the data was written. On the other hand, Impala does not make any time zone adjustment when it writes or reads INT96 TIMESTAMP values to Parquet files. This difference in time zone handling can cause potentially inconsistent results when Impala processes TIMESTAMP values in the Parquet files written by Hive.

  To avoid incompatibility problems or having to code workarounds, you can specify one or both of these impalad startup flags:

  - --use_local_tz_for_unix_timestamp_conversions=true
  - --convert_legacy_hive_parquet_utc_timestamps=true

  When the ##convert_legacy_hive_parquet_utc_timestamps setting is enabled, Impala recognizes the Parquet data files written by Hive, and applies the same UTC-to-local-timezone conversion logic during the query as Hive does.

  In Impala 3.0 and lower, the ##convert_legacy_hive_parquet_utc_timestamps setting had a severe impact on multi-threaded performance. The new time zone implementation in Impala 3.1 eliminated most of the performance overhead and made Impala scale well to multiple threads. The ##convert_legacy_hive_parquet_utc_timestamps setting is turned off by default for a performance reason. To avoid unexpected incompatibility problems, you should turn on the option when processing TIMESTAMP columns in Parquet files written by Hive.

  Hive currently cannot write INT64 TIMESTAMP values.

  In Impala 3.2 and higher, INT64 TIMESTAMP values annotated with the TIMESTAMP_MILLIS or TIME STAMP_MICROS OriginalType are assumed to be always UTC normalized, so the UTC to local conversion will be always done. INT64 TIMESTAMP annotated with the TIMESTAMP LogicalType specifies whether UTC to local conversion is necessary depending on the Parquet metadata.

Conversions:

Impala automatically converts STRING literals of the correct format into TIMESTAMP values. Timestamp values are accepted in the format 'yyyy#MM#dd HH:mm:ss.SSSSSS', and can consist of just the date, or just the time, with or without the fractional second portion. For example, you can specify TIMESTAMP values such as '1966#07#30', '08:30:00', or '1985#09#25 17:45:30.005'.

Leading zeroes are not required in the numbers representing the date component, such as month and date, or the time component, such as hour, minute, and second. For example, Impala accepts both '2018#1#1 01:02:03' and '2018-01-01 1:2:3' as valid.

In STRING to TIMESTAMP conversions, leading and trailing white spaces, such as a space, a tab, a newline, or a carriage return, are ignored. For example, Impala treats the following as equivalent: '1999#12#01 01:02:03 ', ' 1999#12#01 01:02:03', '1999#12#01 01:02:03\r\n\t'.

When you convert or cast a STRING literal to TIMESTAMP, you can use the following separators between the date part and the time part:

- One or more space characters

  Example: CAST('2001-01-09 01:05:01' AS TIMESTAMP)
- The character “T”

  Example: CAST('2001-01-09T01:05:01' AS TIMESTAMP)

Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

In Impala 1.3 and higher, the FROM_UNIXTIME() and UNIX_TIMESTAMP() functions allow a wider range of format strings, with more flexibility in element order, repetition of letter placeholders, and separator characters. In Impala 2.3 and higher, the UNIX_TIMESTAMP() function also allows a numeric timezone offset to be specified as part of the input string. See Impala date and time functions on page 344 for details.

In Impala 2.2.0 and higher, built-in functions that accept or return integers representing TIMESTAMP values use the BIGINT type for parameters and return values, rather than INT. This change lets the date and time functions avoid an overflow error that would otherwise occur on January 19th, 2038 (known as the “Year 2038 problem” or “Y2K38 problem”). This change affects the FROM_UNIXTIME() and UNIX_TIMESTAMP() functions. You might need to change application code that interacts with these functions, change the types of columns that store the return values, or add CAST() calls to SQL statements that call these functions.

Partitioning:

Although you cannot use a TIMESTAMP column as a partition key, you can extract the individual years, months, days, hours, and so on and partition based on those columns. Because the partition key column values are represented in HDFS directory names, rather than as fields in the data files themselves, you can also keep the original TIME STAMP values if desired, without duplicating data or wasting storage space. See *Partition Key Columns* for more details on partitioning with date and time values.

```
[localhost:21000] > create table timeline (event string) partitioned by (hap
pened timestamp);
ERROR: AnalysisException: Type 'TIMESTAMP' is not supported as partition-c
olumn type in column: happened
```

NULL considerations: Casting any unrecognized STRING value to this type produces a NULL value.

HBase considerations: This data type is fully compatible with HBase tables.

Parquet consideration: INT96 and INT64 encoded Parquet timestamps are supported in Impala.

Parquet considerations: This type is fully compatible with Parquet tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Kudu considerations:

In Impala 2.9 and higher, you can include TIMESTAMP columns in Kudu tables, instead of representing the date and time as a BIGINT value. The behavior of TIMESTAMP for Kudu tables has some special considerations:

- Any nanoseconds in the original 96-bit value produced by Impala are not stored, because Kudu represents date/time columns using 64-bit values. The nanosecond portion of the value is rounded, not truncated. Therefore, a TIMESTAMP value that you store in a Kudu table might not be bit-for-bit identical to the value returned by a query.
- The conversion between the Impala 96-bit representation and the Kudu 64-bit representation introduces some performance overhead when reading or writing TIMESTAMP columns. You can minimize the overhead during writes by performing inserts through the Kudu API. Because the overhead during reads applies to each query, you might continue to use a BIGINT column to represent date/time values in performance-critical applications.
- The Impala TIMESTAMP type has a narrower range for years than the underlying Kudu data type. Impala can represent years 1400-9999. If year values outside this range are written to a Kudu table by a non-Impala client, Impala returns NULL by default when reading those TIMESTAMP values during a query. Or, if the ABORT_ON_ERROR query option is enabled, the query fails when it encounters a value with an out-of-range year.

Restrictions:

If you cast a STRING with an unrecognized format to a TIMESTAMP, the result is NULL rather than an error. Make sure to test your data pipeline to be sure any textual date and time values are in a format that Impala TIMESTAMP can recognize.

Currently, Avro tables cannot contain TIMESTAMP columns. If you need to store date and time values in Avro tables, as a workaround you can use a STRING representation of the values, convert the values to BIGINT with the UNIX_TIMESTAMP() function, or create separate numeric columns for individual date and time fields using the EXTRACT() function.

Examples:

The following examples demonstrate using TIMESTAMP values with built-in functions:

```
select cast('1966-07-30' as timestamp);
select cast('1985-09-25 17:45:30.005' as timestamp);
select cast('08:30:00' as timestamp);
select hour('1970-01-01 15:30:00');         -- Succeeds, returns 15.
select hour('1970-01-01 15:30');            -- Returns NULL because seconds
field required.
select hour('1970-01-01 27:30:00');         -- Returns NULL because hour val
ue out of range.
select dayofweek('2004-06-13');             -- Returns 1, representing Sund
ay.
select dayname('2004-06-13');               -- Returns 'Sunday'.
select date_add('2004-06-13', 365);         -- Returns 2005-06-13 with zeros
 for hh:mm:ss fields.
select day('2004-06-13');                   -- Returns 13.
select datediff('1989-12-31','1984-09-01'); -- How many days between these 2
 dates?
select now();                               -- Returns current date and time
 in local timezone.
```

The following examples demonstrate using TIMESTAMP values with HDFS-backed tables:

```
create table dates_and_times (t timestamp);
insert into dates_and_times values
  ('1966-07-30'), ('1985-09-25 17:45:30.005'), ('08:30:00'), (now());
```

The following examples demonstrate using TIMESTAMP values with Kudu tables:

```
create table timestamp_t (x int primary key, s string, t timestamp, b bigint
)
  partition by hash (x) partitions 16
  stored as kudu;
```

```
-- The default value of now() has microsecond precision, so the final 3 d
igits
-- representing nanoseconds are all zero.
insert into timestamp_t values (1, cast(now() as string), now(), unix_timest
amp(now()));

-- Values with 1-499 nanoseconds are rounded down in the Kudu TIMESTAMP co
lumn.
insert into timestamp_t values (2, cast(now() + interval 100 nanoseconds as
 string), now() + interval 100 nanoseconds, unix_timestamp(now() + interval
 100 nanoseconds));
insert into timestamp_t values (3, cast(now() + interval 499 nanoseconds as
string), now() + interval 499 nanoseconds, unix_timestamp(now() + interval 4
99 nanoseconds));

-- Values with 500-999 nanoseconds are rounded up in the Kudu TIMESTAMP c
olumn.
insert into timestamp_t values (4, cast(now() + interval 500 nanoseconds as
 string), now() + interval 500 nanoseconds, unix_timestamp(now() + interval
 500 nanoseconds));
insert into timestamp_t values (5, cast(now() + interval 501 nanoseconds as
 string), now() + interval 501 nanoseconds, unix_timestamp(now() + interval
501 nanoseconds));

-- The string representation shows how underlying Impala TIMESTAMP can have
nanosecond precision.
-- The TIMESTAMP column shows how timestamps in a Kudu table are rounded to
 microsecond precision.
-- The BIGINT column represents seconds past the epoch and so if not affect
ed much by nanoseconds.
select s, t, b from timestamp_t order by t;
+-------------------------------+-------------------------------+-----------
-+
| s                             | t                             | b
  |
+-------------------------------+-------------------------------+---------
---+
| 2017-05-31 15:30:05.107157000 | 2017-05-31 15:30:05.107157000 | 1496244605
 |
| 2017-05-31 15:30:28.868151100 | 2017-05-31 15:30:28.868151000 | 1496244628
 |
| 2017-05-31 15:34:33.674692499 | 2017-05-31 15:34:33.674692000 | 1496244
873 |
| 2017-05-31 15:35:04.769166500 | 2017-05-31 15:35:04.769167000 | 1496244904
 |
| 2017-05-31 15:35:33.033082501 | 2017-05-31 15:35:33.033083000 | 1496244933
 |
+-------------------------------+-------------------------------+-----------
-+
```

Added in: Available in all versions of Impala.

Related information:

- To convert to or from different date formats, or perform date arithmetic, use the date and time functions described in Impala date and time functions on page 344. In particular, the from_unixtime() function requires a case-sensitive format string such as "yyyy-MM-dd HH:mm:ss.SSSS", matching one of the allowed variations of a TIMESTAMP value (date plus time, only date, only time, optional fractional seconds).
- See SQL differences between Impala and Hive on page 474 for details about differences in TIMESTAMP handling between Impala and Hive.

Related Information

Hive data types
DATE data type
Partitioning

Customizing time zones

Impala 3.1 onward, you can customize the time zone definitions used in Impala.

- By default, Impala uses the OS's time zone database located in /usr/share/zoneinfo. This directory contains the IANA timezone database in a compiled binary format. The contents of the zoneinfo directory is controlled by the OS's package manager.
- Use the following start-up flags managed as impalad safety valves in Cloudera Manager.
  - ##hdfs_zone_info_zip: This flag allows Impala administrators to specify a custom timezone database. The flag should be set to a shared (not necessarily HDFS) path that points to a zip archive of a custom IANA timezone database. The timezone database is expected to be in a compiled binary format. If the startup flag is set, Impala will use the specified timezone database instead of the default /usr/share/zoneinfo database. The timezone db upgrade process is described in detail below.
  - ##hdfs_zone_alias_conf: This flag allows Impala administrators to specify definitions for custom timezone aliases. The flag should be set to a shared (not necessarily HDFS) path that specifies a config file containing custom timezone alias definitions. This config file can be used as a workaround for users who want to keep using their legacy timezone names. Configuring custom aliases is described in detail below.

Upgrading custom IANA time zone database:

1. Download latest IANA time zone database distribution:

   ```
   git clone https://github.com/eggert/tz
   ```

   Alternatively, download a specific tzdb version from:

   ```
    https://www.iana.org/time-zones/repository
   ```

2. Build timezone tools:

   ```
   cd tz
   make TOPDIR=tzdata install
   ```

3. Generate the compiled binary time zone database:

   ```
   ./zic -d ./tzdata/etc/zoneinfo africa antarctica asia australasia backward
    backzone etcetera europe factory northamerica pacificnew southamerica sys
   temv
   ```

4. Create zip archive:

   ```
   pushd ./tzdata/etc
   zip -r zoneinfo.zip zoneinfo
   popd
   ```

5. Copy the time zone database to HDFS:

   ```
   hdfs dfs -mkdir -p /tzdb/latest
   hdfs dfs -copyFromLocal ./tzdata/etc/zoneinfo.zip /tzdb/latest
   ```

6. Set the ##hdfs_zone_info_zip startup flag to /tzdb/latest/zoneinfo.zip as an impalad safety valve.
7. Perform a full restart of Impala service.

Configuring custom time zone aliases:

1. Create a tzalias.conf config file that contains time zone alias definitions formatted as ALIAS = DEFINITION. For example:

```
#
# Define aliases for existing timezone names:
#
Universal Coordinated Time = UTC
Mideast/Riyadh89 = Asia/Riyadh
PDT = America/Los_Angeles
#
# Define aliases as UTC offsets in seconds:
#
GMT-01:00 = 3600
GMT+01:00 = -3600
```

2. Copy the config file to HDFS:

```
hdfs dfs -mkdir -p /tzdb
hdfs dfs -copyFromLocal tzalias.conf /tzdb
```

3. Set the ##hdfs_zone_alias_conf startup flag to /tzdb/tzalias.conf as an impalad safety valve.
4. Perform a full restart of Impala service.

Added in: Impala 3.1.

TINYINT data type

A 1-byte integer data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME TINYINT
```

Range: -128 .. 127. There is no UNSIGNED subtype.

Conversions: Impala automatically converts to a larger integer type (SMALLINT, INT, or BIGINT) or a floating-point type (FLOAT or DOUBLE) automatically. Use CAST() to convert to STRING or TIMESTAMP. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Impala does not return column overflows as NULL, so that customers can distinguish between NULL data and overflow conditions similar to how they do so with traditional database systems. Impala returns the largest or smallest value in the range for the type. For example, valid values for a tinyint range from -128 to 127. In Impala, a tinyint with a value of -200 returns -128 rather than NULL. A tinyint with a value of 200 returns 127.

Usage notes:

For a convenient and automated way to check the bounds of the TINYINT type, call the functions MIN_TINYINT() and MAX_TINYINT().

If an integer value is too large to be represented as a TINYINT, use a SMALLINT instead.

NULL considerations: Casting any non-numeric value to this type produces a NULL value.

Examples:

```
CREATE TABLE t1 (x TINYINT);
SELECT CAST(100 AS TINYINT);
```

Parquet considerations:

Physically, Parquet files represent TINYINT and SMALLINT values as 32-bit integers. Although Impala rejects attempts to insert out-of-range values into such columns, if you create a new table with the CREATE TABLE ... LIKE PARQUET syntax, any TINYINT or SMALLINT columns in the original table turn into INT columns in the new table.

HBase considerations: This data type is fully compatible with HBase tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as a 1-byte value.

Added in: Available in all versions of Impala.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

VARCHAR data type

A variable-length character type, truncated during processing if necessary to fit within the specified length.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME VARCHAR(MAX_LENGTH)
```

The maximum length you can specify is 65,535.

Partitioning: This type can be used for partition key columns. Because of the efficiency advantage of numeric values over character-based values, if the partition key is a string representation of a number, prefer to use an integer type with sufficient range (INT, BIGINT, and so on) where practical.

HBase considerations: This data type cannot be used with HBase tables.

Parquet considerations:

- This type can be read from and written to Parquet files.
- There is no requirement for a particular level of Parquet.
- Parquet files generated by Impala and containing this type can be freely interchanged with other components such as Hive and MapReduce.
- Parquet data files can contain values that are longer than allowed by the VARCHAR(*N*) length limit. Impala ignores any extra trailing characters when it processes those values during a query.

Text table considerations:

Text data files can contain values that are longer than allowed by the VARCHAR(*N*) length limit. Any extra trailing characters are ignored when Impala processes those values during a query.

Avro considerations: The Avro specification allows string values up to 2**64 bytes in length. Impala queries for Avro tables use 32-bit integers to hold string lengths. In Impala 2.5 and higher, Impala truncates CHAR and VARCHAR values in Avro tables to (2**31)-1 bytes. If a query encounters a STRING value longer than (2**31)-1 bytes in an Avro table, the query fails. In earlier releases, encountering such long values in an Avro table could cause a crash.

Schema evolution considerations:

You can use ALTER TABLE ... CHANGE to switch column data types to and from VARCHAR. You can convert from STRING to VARCHAR(*N*), or from VARCHAR(*N*) to STRING, or from CHAR(*N*) to VARCHAR(*N*), or from VARCHAR(*N*) to CHAR(*N*). When switching back and forth between VARCHAR and CHAR, you can also change the length value. This schema evolution works the same for tables using any file format. If a table contains values

longer than the maximum length defined for a VARCHAR column, Impala does not return an error. Any extra trailing characters are ignored when Impala processes those values during a query.

Compatibility:

This type is available in Impala version 2.0 or higher.

Internal details: Represented in memory as a byte array with the minimum size needed to represent each value.

Column statistics considerations: Because the values of this type have variable size, none of the column statistics fields are filled in until you run the COMPUTE STATS statement.

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Restrictions:

All data in CHAR and VARCHAR columns must be in a character encoding that is compatible with UTF-8. If you have binary data from another database system (that is, a BLOB type), use a STRING column to hold it.

Examples:

The following examples show how long and short VARCHAR values are treated. Values longer than the maximum specified length are truncated by CAST(), or when queried from existing data files. Values shorter than the maximum specified length are represented as the actual length of the value, with no extra padding as seen with CHAR values.

```
create table varchar_1 (s varchar(1));
create table varchar_4 (s varchar(4));
create table varchar_20 (s varchar(20));

insert into varchar_1 values (cast('a' as varchar(1))), (cast('b' as varchar
(1))), (cast('hello' as varchar(1))), (cast('world' as varchar(1)));
insert into varchar_4 values (cast('a' as varchar(4))), (cast('b' as varcha
r(4))), (cast('hello' as varchar(4))), (cast('world' as varchar(4)));
insert into varchar_20 values (cast('a' as varchar(20))), (cast('b' as var
char(20))), (cast('hello' as varchar(20))), (cast('world' as varchar(20)));
select * from varchar_1;
+---+
| s |
+---+
| a |
| b |
| h |
| w |
+---+
select * from varchar_4;
+------+
| s    |
+------+
| a    |
| b    |
| hell |
| worl |
+------+
[localhost:21000] > select * from varchar_20;
+-------+
| s     |
+-------+
| a     |
| b     |
| hello |
| world |
+-------+
select concat('[',s,']') as s from varchar_20;
+---------+
```

```
| s       |
+---------+
| [a]     |
| [b]     |
| [hello] |
| [world] |
+---------+
```

The following example shows how identical VARCHAR values compare as equal, even if the columns are defined with different maximum lengths. Both tables contain 'a' and 'b' values. The longer 'hello' and 'world' values from the VARCHAR_20 table were truncated when inserted into the VARCHAR_1 table.

```
select s from varchar_1 join varchar_20 using (s);
+-------+
| s     |
+-------+
| a     |
| b     |
+-------+
```

The following examples show how VARCHAR values are freely interchangeable with STRING values in contexts such as comparison operators and built-in functions:

```
select length(cast('foo' as varchar(100))) as length;
+--------+
| length |
+--------+
| 3      |
+--------+
select cast('xyz' as varchar(5)) > cast('abc' as varchar(10)) as greater;
+---------+
| greater |
+---------+
| true    |
+---------+
```

UDF considerations: This type cannot be used for the argument or return type of a user-defined function (UDF) or user-defined aggregate function (UDA).

Complex types

Complex types (also referred to as *nested types*) let you represent multiple data values within a single row/column position. They differ from the familiar column types such as BIGINT and STRING, known as *scalar types* or *primitive types*, which represent a single data value within a given row/column position. Impala supports the complex types ARRAY, MAP, and STRUCT in Impala 2.3 and higher. The Hive UNION type is not currently supported.

Once you understand the basics of complex types, refer to the individual type topics when you need to refresh your memory about syntax and examples:

- ARRAY complex type on page 13
- STRUCT complex type on page 39
- MAP complex type on page 32

Benefits of Impala complex types

The reasons for using Impala complex types include the following:

- You already have data produced by Hive or other non-Impala component that uses the complex type column names. You might need to convert the underlying data to Parquet to use it with Impala.

- Your data model originates with a non-SQL programming language or a NoSQL data management system. For example, if you are representing Python data expressed as nested lists, dictionaries, and tuples, those data structures correspond closely to Impala ARRAY, MAP, and STRUCT types.
- Your analytic queries involving multiple tables could benefit from greater locality during join processing. By packing more related data items within each HDFS data block, complex types let join queries avoid the network overhead of the traditional Hadoop shuffle or broadcast join techniques.

The Impala complex type support produces result sets with all scalar values, and the scalar components of complex types can be used with all SQL clauses, such as GROUP BY, ORDER BY, all kinds of joins, subqueries, and inline views. The ability to process complex type data entirely in SQL reduces the need to write application-specific code in Java or other programming languages to deconstruct the underlying data structures.

Overview of Impala complex types

The ARRAY and MAP types are closely related: they represent collections with arbitrary numbers of elements, where each element is the same type. In contrast, STRUCT groups together a fixed number of items into a single element. The parts of a STRUCT element (the *fields*) can be of different types, and each field has a name.

The elements of an ARRAY or MAP, or the fields of a STRUCT, can also be other complex types. You can construct elaborate data structures with up to 100 levels of nesting. For example, you can make an ARRAY whose elements are STRUCTs. Within each STRUCT, you can have some fields that are ARRAY, MAP, or another kind of STRUCT. The Impala documentation uses the terms complex and nested types interchangeably; for simplicity, it primarily uses the term complex types, to encompass all the properties of these types.

When visualizing your data model in familiar SQL terms, you can think of each ARRAY or MAP as a miniature table, and each STRUCT as a row within such a table. By default, the table represented by an ARRAY has two columns, POS to represent ordering of elements, and ITEM representing the value of each element. Likewise, by default, the table represented by a MAP encodes key-value pairs, and therefore has two columns, KEY and VALUE.

The ITEM and VALUE names are only required for the very simplest kinds of ARRAY and MAP columns, ones that hold only scalar values. When the elements within the ARRAY or MAP are of type STRUCT rather than a scalar type, then the result set contains columns with names corresponding to the STRUCT fields rather than ITEM or VALUE.

You write most queries that process complex type columns using familiar join syntax, even though the data for both sides of the join resides in a single table. The join notation brings together the scalar values from a row with the values from the complex type columns for that same row. The final result set contains all scalar values, allowing you to do all the familiar filtering, aggregation, ordering, and so on for the complex data entirely in SQL or using business intelligence tools that issue SQL queries.

Behind the scenes, Impala ensures that the processing for each row is done efficiently on a single host, without the network traffic involved in broadcast or shuffle joins. The most common type of join query for tables with complex type columns is INNER JOIN, which returns results only in those cases where the complex type contains some elements. Therefore, most query examples in this section use either the INNER JOIN clause or the equivalent comma notation.

Note:

Although Impala can query complex types that are present in Parquet files, Impala currently cannot create new Parquet files containing complex types. Therefore, the discussion and examples presume that you are working with existing Parquet data produced through Hive, Spark, or some other source. See Constructing Parquet / ORC files with complex columns using Hive for examples of constructing Parquet data files with complex type columns.

For learning purposes, you can create empty tables with complex type columns and practice query syntax, even if you do not have sample data with the required structure.

Design considerations for complex types

When planning to use Impala complex types, and designing the Impala schema, first learn how this kind of schema differs from traditional table layouts from the relational database and data warehousing fields. Because you might

have already encountered complex types in a Hadoop context while using Hive for ETL, also learn how to write high-performance analytic queries for complex type data using Impala SQL syntax.

How complex types differ from traditional data warehouse schemas

Complex types let you associate arbitrary data structures with a particular row. If you are familiar with schema design for relational database management systems or data warehouses, a schema with complex types has the following differences:

- Logically, related values can now be grouped tightly together in the same table.

  In traditional data warehousing, related values were typically arranged in one of two ways:

  - Split across multiple normalized tables. Foreign key columns specified which rows from each table were associated with each other. This arrangement avoided duplicate data and therefore the data was compact, but join queries could be expensive because the related data had to be retrieved from separate locations. (In the case of distributed Hadoop queries, the joined tables might even be transmitted between different hosts in a cluster.)
  - Flattened into a single denormalized table. Although this layout eliminated some potential performance issues by removing the need for join queries, the table typically became larger because values were repeated. The extra data volume could cause performance issues in other parts of the workflow, such as longer ETL cycles or more expensive full-table scans during queries.

  Complex types represent a middle ground that addresses these performance and volume concerns. By physically locating related data within the same data files, complex types increase locality and reduce the expense of join queries. By associating an arbitrary amount of data with a single row, complex types avoid the need to repeat lengthy values such as strings. Because Impala knows which complex type values are associated with each row, you can save storage by avoiding artificial foreign key values that are only used for joins. The flexibility of the STRU CT, ARRAY, and MAP types lets you model familiar constructs such as fact and dimension tables from a data warehouse, and wide tables representing sparse matrices.

Physical storage for complex types in Parquet

Physically, the scalar and complex columns in each row are located adjacent to each other in the same Parquet data file, ensuring that they are processed on the same host rather than being broadcast across the network when cross-referenced within a query. This co-location simplifies the process of copying, converting, and backing all the columns up at once. Because of the column-oriented layout of Parquet files, you can still query only the scalar columns of a table without imposing the I/O penalty of reading the (possibly large) values of the composite columns.

Within each Parquet data file, the constituent parts of complex type columns are stored in column-oriented format:

- Each field of a STRUCT type is stored like a column, with all the scalar values adjacent to each other and encoded, compressed, and so on using the Parquet space-saving techniques.
- For an ARRAY containing scalar values, all those values (represented by the ITEM pseudocolumn) are stored adjacent to each other.
- For a MAP, the values of the KEY pseudocolumn are stored adjacent to each other. If the VALUE pseudocolumn is a scalar type, its values are also stored adjacent to each other.
- If an ARRAY element, STRUCT field, or MAP VALUE part is another complex type, the column-oriented storage applies to the next level down (or the next level after that, and so on for deeply nested types) where the final elements, fields, or values are of scalar types.

The numbers represented by the POS pseudocolumn of an ARRAY are not physically stored in the data files. They are synthesized at query time based on the order of the ARRAY elements associated with each row.

File format support for Impala complex types

Currently, Impala queries support complex type data in the Parquet and ORC file formats. See *Using Parquet data files* for details about the performance benefits and physical layout of Parquet file format.

Because Impala does not parse the data structures containing nested types for unsupported formats such as text, Avro, SequenceFile, or RCFile, you cannot use data files in these formats with Impala, even if the query does not refer to the nested type columns. Also, if a table using an unsupported format originally contained nested type columns, and then those columns were dropped from the table using ALTER TABLE ... DROP COLUMN, any existing data files in the table still contain the nested type data and Impala queries on that table will generate errors.

The one exception to the preceding rule is COUNT(*) queries on RCFile tables that include complex types. Such queries are allowed in Impala 2.6 and higher.

You can perform DDL operations for tables involving complex types in most file formats other than Parquet or ORC. You cannot create tables in Impala with complex types using text files.

You can have a partitioned table with complex type columns that uses a format other than Parquet or ORC, and use ALTER TABLE to change the file format to Parquet/ORC for individual partitions. When you put Parquet/ORC files into those partitions, Impala can run queries against that data as long as the query does not involve any of the non-Parquet and non-ORC partitions.

If you use the `parquet-tools` command to examine the structure of a Parquet data file that includes complex types, you see that both ARRAY and MAP are represented as a Bag in Parquet terminology, with all fields marked Optional because Impala allows any column to be nullable.

Impala supports either 2-level and 3-level encoding within each Parquet data file. When constructing Parquet data files outside Impala, use either encoding style but do not mix 2-level and 3-level encoding within the same data file.

Choosing between complex types and normalized tables

Choosing between multiple normalized fact and dimension tables, or a single table containing complex types, is an important design decision.

- If you are coming from a traditional database or data warehousing background, you might be familiar with how to split up data between tables. Your business intelligence tools might already be optimized for dealing with this kind of multi-table scenario through join queries.
- If you are pulling data from Impala into an application written in a programming language that has data structures analogous to the complex types, such as Python or Java, complex types in Impala could simplify data interchange and improve understandability and reliability of your program logic.
- You might already be faced with existing infrastructure or receive high volumes of data that assume one layout or the other. For example, complex types are popular with web-oriented applications, for example to keep information about an online user all in one place for convenient lookup and analysis, or to deal with sparse or constantly evolving data fields.
- If some parts of the data change over time while related data remains constant, using multiple normalized tables lets you replace certain parts of the data without reloading the entire data set. Conversely, if you receive related data all bundled together, such as in JSON files, using complex types can save the overhead of splitting the related items across multiple tables.
- From a performance perspective:
  - In Parquet or ORC tables, Impala can skip columns that are not referenced in a query, avoiding the I/O penalty of reading the embedded data. When complex types are nested within a column, the data is physically divided at a very granular level; for example, a query referring to data nested multiple levels deep in a complex type column does not have to read all the data from that column, only the data for the relevant parts of the column type hierarchy.
  - Complex types avoid the possibility of expensive join queries when data from fact and dimension tables is processed in parallel across multiple hosts. All the information for a row

containing complex types is typically to be in the same data block, and therefore does not need to be transmitted across the network when joining fields that are all part of the same row.

- The tradeoff with complex types is that fewer rows fit in each data block. Whether it is better to have more data blocks with fewer rows, or fewer data blocks with many rows, depends on the distribution of your data and the characteristics of your query workload. If the complex columns are rarely referenced, using them might lower efficiency. If you are seeing low parallelism due to a small volume of data (relatively few data blocks) in each table partition, increasing the row size by including complex columns might produce more data blocks and thus spread the work more evenly across the cluster. See *Scalability considerations* for more on this advanced topic.

Differences between Impala and Hive complex types

Impala can query Parquet and ORC tables containing ARRAY, STRUCT, and MAP columns produced by Hive. There are some differences to be aware of between the Impala SQL and Hive SQL syntax for complex types, primarily for queries.

Impala supports a subset of the syntax that Hive supports for specifying ARRAY, STRUCT, and MAP types in the CREATE TABLE statements.

Because Impala STRUCT columns include user-specified field names, you use the NAMED_STRUCT() constructor in Hive rather than the STRUCT() constructor when you populate an Impala STRUCT column using a Hive INSERT statement.

The Hive UNION type is not currently supported in Impala.

While Impala usually aims for a high degree of compatibility with Hive SQL query syntax, Impala syntax differs from Hive for queries involving complex types. The differences are intended to provide extra flexibility for queries involving these kinds of tables.

- Impala uses dot notation for referring to element names or elements within complex types, and join notation for cross-referencing scalar columns with the elements of complex types within the same row, rather than the LATERAL VIEW clause and EXPLODE() function of Hive SQL.
- Using join notation lets you use all the kinds of join queries with complex type columns. For example, you can use a LEFT OUTER JOIN, LEFT ANTI JOIN, or LEFT SEMI JOIN query to evaluate different scenarios where the complex columns do or do not contain any elements.
- You can include references to collection types inside subqueries and inline views. For example, you can construct a FROM clause where one of the “tables” is a subquery against a complex type column, or use a subquery against a complex type column as the argument to an IN or EXISTS clause.
- The Impala pseudocolumn POS lets you retrieve the position of elements in an array along with the elements themselves, equivalent to the POSEXPLODE() function of Hive SQL. You do not use index notation to retrieve a single array element in a query; the join query loops through the array elements and you use WHERE clauses to specify which elements to return.
- Join clauses involving complex type columns do not require an ON or USING clause. Impala implicitly applies the join key so that the correct array entries or map elements are associated with the correct row from the table.
- Impala does not currently support the UNION complex type.

Limitations and restrictions for complex types

Complex type columns can only be used in tables or partitions with the Parquet or ORC file format.

Complex type columns cannot be used as partition key columns in a partitioned table.

When you use complex types with the ORDER BY, GROUP BY, HAVING, or WHERE clauses, you cannot refer to the column name by itself. Instead, you refer to the names of the scalar values within the complex type, such as the ITEM, POS, KEY, or VALUE pseudocolumns, or the field names from a STRUCT.

The maximum depth of nesting for complex types is 100 levels.

The maximum length of the column definition for any complex type, including declarations for any nested types, is 4000 characters.

For ideal performance and scalability, use small or medium-sized collections, where all the complex columns contain at most a few hundred megabytes per row. Remember, all the columns of a row are stored in the same HDFS data block, whose size in Parquet files typically ranges from 256 MB to 1 GB.

Including complex type columns in a table introduces some overhead that might make queries that do not reference those columns somewhat slower than Impala queries against tables without any complex type columns. Expect at most a 2x slowdown compared to tables that do not have any complex type columns.

Currently, the COMPUTE STATS statement does not collect any statistics for columns containing complex types. Impala uses heuristics to construct execution plans involving complex type columns.

Currently, Impala built-in functions and user-defined functions cannot accept complex types as parameters or produce them as function return values. (When the complex type values are materialized in an Impala result set, the result set contains the scalar components of the values, such as the POS or ITEM for an ARRAY, the KEY or VALUE for a MAP, or the fields of a STRUCT; these scalar data items can be used with built-in functions and UDFs as usual.)

Impala currently cannot write new data files containing complex type columns. Therefore, although the SELECT statement works for queries involving complex type columns, you cannot use a statement form that writes data to complex type columns, such as CREATE TABLE AS SELECT or INSERT ... SELECT. To create data files containing complex type data, use the Hive INSERT statement, or another ETL mechanism such as MapReduce jobs, Spark jobs, Pig, and so on.

Currently, Impala can query complex type columns only from Parquet/ORC tables or Parquet/ORC partitions within partitioned tables. Although you can use complex types in tables with Avro, text, and other file formats as part of your ETL pipeline, for example as intermediate tables populated through Hive, doing analytics through Impala requires that the data eventually ends up in a Parquet/ORC table. The requirement for Parquet/ORC data files means that you can use complex types with Impala tables hosted on other kinds of file storage systems such as Isilon and Amazon S3, but you cannot use Impala to query complex types from HBase tables.

Using complex types in SQL

When using complex types through SQL in Impala, you learn the notation for < > delimiters for the complex type columns in CREATE TABLE statements, and how to construct join queries to “unpack” the scalar values nested inside the complex data structures. You might need to condense a traditional RDBMS or data warehouse schema into a smaller number of Parquet tables, and use Hive, Spark, Pig, or other mechanism outside Impala to populate the tables with data.

Complex type syntax for DDL statements

The definition of *DATA_TYPE*, as seen in the CREATE TABLE and ALTER TABLE statements, now includes complex types in addition to primitive types:

```
  primitive_type
| array_type
| map_type
| struct_type
```

Unions are not currently supported.

Array, struct, and map column type declarations are specified in the CREATE TABLE statement. You can also add or change the type of complex columns through the ALTER TABLE statement.

Currently, Impala queries allow complex types only in tables that use the Parquet or ORC format. If an Impala query encounters complex types in a table or partition using any another file format, the query returns a runtime error.

You can use ALTER TABLE ... SET FILEFORMAT PARQUET to change the file format of an existing table containing complex types to Parquet, after which Impala can query it. Make sure to load Parquet files into the table after changing the file format, because the ALTER TABLE ... SET FILEFORMAT statement does not convert existing data to the new file format.

Partitioned tables can contain complex type columns. All the partition key columns must be scalar types.

Because use cases for Impala complex types require that you already have Parquet/ORC data files produced outside of Impala, you can use the Impala CREATE TABLE LIKE PARQUET syntax to produce a table with columns that match the structure of an existing Parquet file, including complex type columns for nested data structures. Remember to include the STORED AS PARQUET clause in this case, because even with CREATE TABLE LIKE PARQUET, the default file format of the resulting table is still text.

You cannot use the CREATE TABLE AS SELECT syntax to create a table with nested type columns because the complex columns are omitted from the result set of an Impala SELECT * or SELECT *COL_NAME* query, and because Impala currently does not support writing Parquet files with complex type columns,

Note:

Once you have a table set up with complex type columns, use the DESCRIBE and SHOW CREATE TABLE statements to see the correct notation with < and > delimiters and comma and colon separators within the complex type definitions. If you do not have existing data with the same layout as the table, you can query the empty table to practice with the notation for the SELECT statement. In the SELECT list, you use dot notation and pseudocolumns such as ITEM, KEY, and VALUE for referring to items within the complex type columns. In the FROM clause, you use join notation to construct table aliases for any referenced ARRAY and MAP columns.

For example, when defining a table that holds contact information, you might represent phone numbers differently depending on the expected layout and relationships of the data, and how well you can predict those properties in advance.

Here are different ways that you might represent phone numbers in a traditional relational schema, with equivalent representations using complex types.

Traditional relational representation of phone numbers: single table

The traditional, simplest way to represent phone numbers in a relational table is to store all contact info in a single table, with all columns having scalar types, and each potential phone number represented as a separate column. In this example, each person can only have these 3 types of phone numbers. If the person does not have a particular kind of phone number, the corresponding column is NULL for that row.

```
CREATE TABLE contacts_fixed_phones
(
    id BIGINT
  , name STRING
  , address STRING
  , home_phone STRING
  , work_phone STRING
  , mobile_phone STRING
) STORED AS PARQUET;
```

An array of phone numbers

Using a complex type column to represent the phone numbers adds some extra flexibility. Now there could be an unlimited number of phone numbers. Because the array elements have an order but not symbolic names, you could decide in advance that phone_number[0] is the home number, [1] is the work number, [2] is the mobile number, and so on. (In subsequent examples, you will see how to create a more flexible naming scheme using other complex type variations, such as a MAP or an ARRAY where each element is a STRUCT.)

```
CREATE TABLE contacts_array_of_phones
(
    id BIGINT
  , name STRING
  , address STRING
  , phone_number ARRAY < STRING >
) STORED AS PARQUET;
```

A map of phone numbers

Another way to represent an arbitrary set of phone numbers is with a MAP column. With a MAP, each element is associated with a key value that you specify, which could be a numeric, string, or other scalar type. This example uses a STRING key to give each phone number a name, such as 'home' or 'mobile'. A query could filter the data based on the key values, or display the key values in reports.

```
CREATE TABLE contacts_unlimited_phones
(
  id BIGINT, name STRING, address STRING, phone_number MAP < S
TRING,STRING >
) STORED AS PARQUET;
```

Traditional relational representation of phone numbers: normalized tables

If you are an experienced database designer, you already know how to work around the limitations of the single-table schema from *Traditional relational representation of phone numbers: single table*. By normalizing the schema, with the phone numbers in their own table, you can associate an arbitrary set of phone numbers with each person, and associate additional details with each phone number, such as whether it is a home, work, or mobile phone.

The flexibility of this approach comes with some drawbacks. Reconstructing all the data for a particular person requires a join query, which might require performance tuning on Hadoop because the data from each table might be transmitted from a different host. Data management tasks such as backups and refreshing the data require dealing with multiple tables instead of a single table.

This example illustrates a traditional database schema to store contact info normalized across 2 tables. The fact table establishes the identity and basic information about person. A dimension table stores information only about phone numbers, using an ID value to associate each phone number with a person ID from the fact table. Each person can have 0, 1, or many phones; the categories are not restricted to a few predefined ones; and the phone table can contain as many columns as desired, to represent all sorts of details about each phone number.

```
CREATE TABLE fact_contacts (id BIGINT, name STRING, address STR
ING) STORED AS PARQUET;
CREATE TABLE dim_phones
(
    contact_id BIGINT
  , category STRING
  , international_code STRING
```

```
  , area_code STRING
  , exchange STRING
  , extension STRING
  , mobile BOOLEAN
  , carrier STRING
  , current BOOLEAN
  , service_start_date TIMESTAMP
  , service_end_date TIMESTAMP
)
STORED AS PARQUET;
```

Phone Numbers Represented as an Array of Structs

To represent a schema equivalent to the one from *Traditional relational representation of phone numbers: normalized tables* using complex types, this example uses an ARRAY where each array element is a STRUCT. As with the earlier complex type examples, each person can have an arbitrary set of associated phone numbers. Making each array element into a STRUCT lets us associate multiple data items with each phone number, and give a separate name and type to each data item. The STRUCT fields of the ARRAY elements reproduce the columns of the dimension table from the previous example.

You can do all the same kinds of queries with the complex type schema as with the normalized schema from the previous example. The advantages of the complex type design are in the areas of convenience and performance. Now your backup and ETL processes only deal with a single table. When a query uses a join to cross-reference the information about a person with their associated phone numbers, all the relevant data for each row resides in the same HDFS data block, meaning each row can be processed on a single host without requiring network transmission.

```
CREATE TABLE contacts_detailed_phones
(
  id BIGINT, name STRING, address STRING
    , phone ARRAY < STRUCT <
        category: STRING
      , international_code: STRING
      , area_code: STRING
      , exchange: STRING
      , extension: STRING
      , mobile: BOOLEAN
      , carrier: STRING
      , current: BOOLEAN
      , service_start_date: TIMESTAMP
      , service_end_date: TIMESTAMP
    >>
) STORED AS PARQUET;
```

SQL statements that support complex types

The Impala SQL statements that support complex types are currently CREATE_TABLE, ALTER_TABLE, DESCRIBE, LOAD_DATA, and SELECT. That is, currently Impala can create or alter tables containing complex type columns, examine the structure of a table containing complex type columns, import existing data files containing complex type columns into a table, and query Parquet/ORC tables containing complex types.

Impala currently cannot write new data files containing complex type columns. Therefore, although the SELECT statement works for queries involving complex type columns, you cannot use a statement form that writes data to complex type columns, such as CREATE TABLE AS SELECT or INSERT ... SELECT. To create data files containing complex type data, use the Hive INSERT statement, or another ETL mechanism such as MapReduce jobs, Spark jobs, Pig, and so on.

DDL statements and complex types

Column specifications for complex or nested types use < and > delimiters:

```
-- What goes inside the < > for an ARRAY is a single type, either
 a scalar or another
-- complex type (ARRAY, STRUCT, or MAP).
CREATE TABLE array_t
(
  id BIGINT,
  a1 ARRAY <STRING>,
  a2 ARRAY <BIGINT>,
  a3 ARRAY <TIMESTAMP>,
  a4 ARRAY <STRUCT <f1: STRING, f2: INT, f3: BOOLEAN>>
)
STORED AS PARQUET;

-- What goes inside the < > for a MAP is two comma-separated t
ypes specifying the types of the key-value pair:
-- a scalar type representing the key, and a scalar or complex
type representing the value.
CREATE TABLE map_t
(
  id BIGINT,
  m1 MAP <STRING, STRING>,
  m2 MAP <STRING, BIGINT>,
  m3 MAP <BIGINT, STRING>,
  m4 MAP <BIGINT, BIGINT>,
  m5 MAP <STRING, ARRAY <STRING>>
)
STORED AS PARQUET;

-- What goes inside the < > for a STRUCT is a comma-separated lis
t of fields, each field defined as
-- name:type. The type can be a scalar or a complex type. The fie
ld names for each STRUCT do not clash
-- with the names of table columns or fields in other STRUCTs. A
 STRUCT is most often used inside
-- an ARRAY or a MAP rather than as a top-level column.
CREATE TABLE struct_t
(
  id BIGINT,
  s1 STRUCT <f1: STRING, f2: BIGINT>,
  s2 ARRAY <STRUCT <f1: INT, f2: TIMESTAMP>>,
  s3 MAP <BIGINT, STRUCT <name: STRING, birthday: TIMESTAMP>>
)
STORED AS PARQUET;
```

Queries and complex types

The following example shows how referring directly to a complex type column returns an error, while SELECT * on the same table succeeds, but only retrieves the scalar columns.

Note: Many of the complex type examples refer to tables such as CUSTOMER and REGION adapted from the tables used in the TPC-H benchmark.

```
SELECT c_orders FROM customer LIMIT 1;
ERROR: AnalysisException: Expr 'c_orders' in select list returns
a complex type 'ARRAY<STRUCT<o_orderkey:BIGINT,o_orderstatus:STR
ING, ... l_receiptdate:STRING,l_shipinstruct:STRING,l_shipmode:S
TRING,l_comment:STRING>>>>'.
Only scalar types are allowed in the select list.

-- Original column has several scalar and one complex column.
```

```
DESCRIBE customer;
+--------------+-------------------------------------+
| name         | type                                |
+--------------+-------------------------------------+
| c_custkey    | bigint                              |
| c_name       | string                              |
...
| c_orders     | array<struct<                       |
|              |   o_orderkey:bigint,                |
|              |   o_orderstatus:string,             |
|              |   o_totalprice:decimal(12,2),       |
...
|              | >>                                  |
+--------------+-------------------------------------+

-- When we SELECT * from that table, only the scalar columns come
 back in the result set.
CREATE TABLE select_star_customer STORED AS PARQUET AS SELECT *
 FROM customer;
+------------------------+
| summary                |
+------------------------+
| Inserted 150000 row(s) |
+------------------------+
-- The c_orders column, being of complex type, was not included
 in the SELECT * result set.
DESC select_star_customer;
+--------------+---------------+
| name         | type          |
+--------------+---------------+
| c_custkey    | bigint        |
| c_name       | string        |
| c_address    | string        |
| c_nationkey  | smallint      |
| c_phone      | string        |
| c_acctbal    | decimal(12,2) |
| c_mktsegment | string        |
| c_comment    | string        |
+--------------+---------------+
```

References to fields within STRUCT columns use dot notation. If the field name is unambiguous, you can omit qualifiers such as table name, column name, or even the ITEM or VALUE pseudocolumn names for STRUCT elements inside an ARRAY or a MAP.

```
SELECT id, address.city FROM customers WHERE address.zip = 94305;
```

References to elements within ARRAY columns use the ITEM pseudocolumn:

```
select r_name, r_nations.item.n_name from region, region.r_natio
ns limit 7;
+--------+----------------+
| r_name | item.n_name    |
+--------+----------------+
| EUROPE | UNITED KINGDOM |
| EUROPE | RUSSIA         |
| EUROPE | ROMANIA        |
| EUROPE | GERMANY        |
| EUROPE | FRANCE         |
| ASIA   | VIETNAM        |
| ASIA   | CHINA          |
+--------+----------------+
```

References to fields within MAP columns use the KEY and VALUE pseudocolumns. In this example, once the query establishes the alias MAP_FIELD for a MAP column with a STRING key and an INT value, the query can refer to MAP_FIELD.KEY and MAP_FIELD.VALUE, which have zero, one, or many instances for each row from the containing table.

```
DESCRIBE table_0;
+---------+-----------------------+
| name    | type                  |
+---------+-----------------------+
| field_0 | string                |
| field_1 | map<string,int>       |
...

SELECT field_0, map_field.key, map_field.value
  FROM table_0, table_0.field_1 AS map_field
WHERE length(field_0) = 1
LIMIT 10;
+---------+-----------+-------+
| field_0 | key       | value |
+---------+-----------+-------+
| b       | gshsgkvd  | NULL  |
| b       | twrtcxj6  | 18    |
| b       | 2vp5      | 39    |
| b       | fh0s      | 13    |
| v       | 2         | 41    |
| v       | 8b58mz    | 20    |
| v       | hw        | 16    |
| v       | 65l388pyt | 29    |
| v       | 03k68g91z | 30    |
| v       | r2hlg5b   | NULL  |
+---------+-----------+-------+
```

When complex types are nested inside each other, you use a combination of joins, pseudocolumn names, and dot notation to refer to specific fields at the appropriate level. This is the most frequent form of query syntax for complex columns, because the typical use case involves two levels of complex types, such as an ARRAY of STRUCT elements.

```
SELECT id, phone_numbers.area_code FROM contact_info_many_structs
 INNER JOIN contact_info_many_structs.phone_numbers phone_numbers
 LIMIT 3;
```

You can express relationships between ARRAY and MAP columns at different levels as joins. You include comparison operators between fields at the top level and within the nested type columns so that Impala can do the appropriate join operation.

Note: Many of the complex type examples refer to tables such as CUSTOMER and REGION adapted from the tables used in the TPC-H benchmark.

For example, the following queries work equivalently. They each return customer and order data for customers that have at least one order.

```
SELECT c.c_name, o.o_orderkey FROM customer c, c.c_orders o LIMIT
 5;
+--------------------+------------+
| c_name             | o_orderkey |
+--------------------+------------+
| Customer#000072578 | 558821     |
| Customer#000072578 | 2079810    |
| Customer#000072578 | 5768068    |
| Customer#000072578 | 1805604    |
```

```
| Customer#000072578 | 3436389    |
+--------------------+------------+

SELECT c.c_name, o.o_orderkey FROM customer c INNER JOIN c.c_o
rders o LIMIT 5;
+--------------------+------------+
| c_name             | o_orderkey |
+--------------------+------------+
| Customer#000072578 | 558821     |
| Customer#000072578 | 2079810    |
| Customer#000072578 | 5768068    |
| Customer#000072578 | 1805604    |
| Customer#000072578 | 3436389    |
+--------------------+------------+
```

The following query using an outer join returns customers that have orders, plus customers with no orders (no entries in the C_ORDERS array):

```
SELECT c.c_custkey, o.o_orderkey
  FROM customer c LEFT OUTER JOIN c.c_orders o
LIMIT 5;
+-----------+------------+
| c_custkey | o_orderkey |
+-----------+------------+
| 60210     | NULL       |
| 147873    | NULL       |
| 72578     | 558821     |
| 72578     | 2079810    |
| 72578     | 5768068    |
+-----------+------------+
```

The following query returns only customers that have no orders. (With LEFT ANTI JOIN or LEFT SEMI JOIN, the query can only refer to columns from the left-hand table, because by definition there is no matching information in the right-hand table.)

```
SELECT c.c_custkey, c.c_name
  FROM customer c LEFT ANTI JOIN c.c_orders o
LIMIT 5;
+-----------+--------------------+
| c_custkey | c_name             |
+-----------+--------------------+
| 60210     | Customer#000060210 |
| 147873    | Customer#000147873 |
| 141576    | Customer#000141576 |
| 85365     | Customer#000085365 |
| 70998     | Customer#000070998 |
+-----------+--------------------+
```

You can also perform correlated subqueries to examine the properties of complex type columns for each row in the result set.

Count the number of orders per customer. Note the correlated reference to the table alias C. The COUNT(*) operation applies to all the elements of the C_ORDERS array for the corresponding row, avoiding the need for a GROUP BY clause.

```
select c_name, howmany FROM customer c, (SELECT COUNT(*) howmany
 FROM c.c_orders) v limit 5;
+--------------------+---------+
| c_name             | howmany |
+--------------------+---------+
| Customer#000030065 | 15      |
| Customer#000065455 | 18      |
```

```
| Customer#000113644 | 21      |
| Customer#000111078 | 0       |
| Customer#000024621 | 0       |
+--------------------+---------+
```

Count the number of orders per customer, ignoring any customers that have not placed any orders:

```
SELECT c_name, howmany_orders
FROM
  customer c,
  (SELECT COUNT(*) howmany_orders FROM c.c_orders) subq1
WHERE howmany_orders > 0
LIMIT 5;
+--------------------+----------------+
| c_name             | howmany_orders |
+--------------------+----------------+
| Customer#000072578 | 7              |
| Customer#000046378 | 26             |
| Customer#000069815 | 11             |
| Customer#000079058 | 12             |
| Customer#000092239 | 26             |
+--------------------+----------------+
```

Count the number of line items in each order. The reference to C.C_ORDERS in the FROM clause is needed because the O_ORDERKEY field is a member of the elements in the C_ORDERS array. The subquery labelled SUBQ1 is correlated: it is re-evaluated for the C_ORDERS.O_LINEITEMS array from each row of the CUSTOMERS table.

```
SELECT c_name, o_orderkey, howmany_line_items
FROM
  customer c,
  c.c_orders t2,
  (SELECT COUNT(*) howmany_line_items FROM c.c_orders.o_lineit
ems) subq1
WHERE howmany_line_items > 0
LIMIT 5;
+--------------------+------------+--------------------+
| c_name             | o_orderkey | howmany_line_items |
+--------------------+------------+--------------------+
| Customer#000020890 | 1884930    | 95                 |
| Customer#000020890 | 4570754    | 95                 |
| Customer#000020890 | 3771072    | 95                 |
| Customer#000020890 | 2555489    | 95                 |
| Customer#000020890 | 919171     | 95                 |
+--------------------+------------+--------------------+
```

Get the number of orders, the average order price, and the maximum items in any order per customer. For this example, the subqueries labelled SUBQ1 and SUBQ2 are correlated: they are re-evaluated for each row from the original CUSTOMER table, and only apply to the complex columns associated with that row.

```
SELECT c_name, howmany, average_price, most_items
FROM
  customer c,
  (SELECT COUNT(*) howmany, AVG(o_totalprice) average_price FROM
 c.c_orders) subq1,
  (SELECT MAX(l_quantity) most_items FROM c.c_orders.o_lineitem
s ) subq2
LIMIT 5;
+--------------------+---------+---------------+------------+
| c_name             | howmany | average_price | most_items |
+--------------------+---------+---------------+------------+
```

```
| Customer#000030065 | 15      | 128908.34     | 50.00      |
| Customer#000088191 | 0       | NULL          | NULL       |
| Customer#000101555 | 10      | 164250.31     | 50.00      |
| Customer#000022092 | 0       | NULL          | NULL       |
| Customer#000036277 | 27      | 166040.06     | 50.00      |
+--------------------+---------+---------------+------------+
```

For example, these queries show how to access information about the ARRAY elements within the CUSTOMER table from the “nested TPC-H” schema, starting with the initial ARRAY elements and progressing to examine the STRUCT fields of the ARRAY, and then the elements nested within another ARRAY of STRUCT:

```
-- How many orders does each customer have?
-- The type of the ARRAY column doesn't matter, this is just cou
nting the elements.
SELECT c_custkey, count(*)
  FROM customer, customer.c_orders
GROUP BY c_custkey
LIMIT 5;
+-----------+----------+
| c_custkey | count(*) |
+-----------+----------+
| 61081     | 21       |
| 115987    | 15       |
| 69685     | 19       |
| 109124    | 15       |
| 50491     | 12       |
+-----------+----------+

-- How many line items are part of each customer order?
-- Now we examine a field from a STRUCT nested inside the ARRAY.
SELECT c_custkey, c_orders.o_orderkey, count(*)
  FROM customer, customer.c_orders c_orders, c_orders.o_lineitems
GROUP BY c_custkey, c_orders.o_orderkey
LIMIT 5;
+-----------+------------+----------+
| c_custkey | o_orderkey | count(*) |
+-----------+------------+----------+
| 63367     | 4985959    | 7        |
| 53989     | 1972230    | 2        |
| 143513    | 5750498    | 5        |
| 17849     | 4857989    | 1        |
| 89881     | 1046437    | 1        |
+-----------+------------+----------+

-- What are the line items in each customer order?
-- One of the STRUCT fields inside the ARRAY is another
-- ARRAY containing STRUCT elements. The join finds
-- all the related items from both levels of ARRAY.
SELECT c_custkey, o_orderkey, l_partkey
  FROM customer, customer.c_orders, c_orders.o_lineitems
LIMIT 5;
+-----------+------------+-----------+
| c_custkey | o_orderkey | l_partkey |
+-----------+------------+-----------+
| 113644    | 2738497    | 175846    |
| 113644    | 2738497    | 27309     |
| 113644    | 2738497    | 175873    |
| 113644    | 2738497    | 88559     |
| 113644    | 2738497    | 8032      |
+-----------+------------+-----------+
```

Pseudocolumns for ARRAY and MAP types

Each element in an ARRAY type has a position, indexed starting from zero, and a value. Each element in a MAP type represents a key-value pair. Impala provides pseudocolumns that let you retrieve this metadata as part of a query, or filter query results by including such things in a WHERE clause. You refer to the pseudocolumns as part of qualified column names in queries:

- ITEM: The value of an array element. If the ARRAY contains STRUCT elements, you can refer to either *ARRAY_NAME*.ITEM.*FIELD_NAME* or use the shorthand *ARRAY_NAME*.*FIELD_NAME*.
- POS: The position of an element within an array.
- KEY: The value forming the first part of a key-value pair in a map. It is not necessarily unique.
- VALUE: The data item forming the second part of a key-value pair in a map. If the VALUE part of the MAP element is a STRUCT, you can refer to either *MAP_NAME*.VALUE.*FIELD_NAME* or use the shorthand *MAP_NAME*.*FIELD_NAME*.

ITEM and POS pseudocolumns

When an ARRAY column contains STRUCT elements, you can refer to a field within the STRU CT using a qualified name of the form *ARRAY_COLUMN*.*FIELD_NAME*. If the ARRAY contains scalar values, Impala recognizes the special name *ARRAY_COLUMN*.ITEM to represent the value of each scalar array element. For example, if a column contained an ARRAY where each element was a STRING, you would use *ARRAY_NAME*.ITEM to refer to each scalar value in the SELECT list, or the WHERE or other clauses.

This example shows a table with two ARRAY columns whose elements are of the scalar type STRI NG. When referring to the values of the array elements in the SELECT list, WHERE clause, or ORDER BY clause, you use the ITEM pseudocolumn because within the array, the individual elements have no defined names.

```
create TABLE persons_of_interest
(
person_id BIGINT,
aliases ARRAY <STRING>,
associates ARRAY <STRING>,
real_name STRING
)
STORED AS PARQUET;

-- Get all the aliases of each person.
SELECT real_name, aliases.ITEM
  FROM persons_of_interest, persons_of_interest.aliases
ORDER BY real_name, aliases.item;

-- Search for particular associates of each person.
SELECT real_name, associates.ITEM
  FROM persons_of_interest, persons_of_interest.associates
WHERE associates.item LIKE '% MacGuffin';
```

Because an array is inherently an ordered data structure, Impala recognizes the special name *ARRAY_COLUMN*.POS to represent the numeric position of each element within the array. The POS pseudocolumn lets you filter or reorder the result set based on the sequence of array elements.

The following example uses a table from a flattened version of the TPC-H schema. The REGION table only has a few rows, such as one row for Europe and one for Asia. The row for each region represents all the countries in that region as an ARRAY of STRUCT elements:

```
[localhost:21000] > desc region;
+-------------+-----------------------------------------------
--------------------+
```

```
| name        | type
                  |
+-------------+-------------------------------------------------
------------------+
| r_regionkey | smallint
                  |
| r_name      | string
                  |
| r_comment   | string
                  |
| r_nations   | array<struct<n_nationkey:smallint,n_name:string,n
_comment:string>> |
+-------------+-------------------------------------------------
-------------------+
```

To find the countries within a specific region, you use a join query. To find out the order of elements in the array, you also refer to the POS pseudocolumn in the select list:

```
[localhost:21000] > SELECT r1.r_name, r2.n_name, r2.POS
                  > FROM region r1 INNER JOIN r1.r_nations r2
                  > WHERE r1.r_name = 'ASIA';
+--------+-----------+-----+
| r_name | n_name    | pos |
+--------+-----------+-----+
| ASIA   | VIETNAM   | 0   |
| ASIA   | CHINA     | 1   |
| ASIA   | JAPAN     | 2   |
| ASIA   | INDONESIA | 3   |
| ASIA   | INDIA     | 4   |
+--------+-----------+-----+
```

Once you know the positions of the elements, you can use that information in subsequent queries, for example to change the ordering of results from the complex type column or to filter certain elements from the array:

```
[localhost:21000] > SELECT r1.r_name, r2.n_name, r2.POS
                  > FROM region r1 INNER JOIN r1.r_nations r2
                  > WHERE r1.r_name = 'ASIA'
                  > ORDER BY r2.POS DESC;
+--------+-----------+-----+
| r_name | n_name    | pos |
+--------+-----------+-----+
| ASIA   | INDIA     | 4   |
| ASIA   | INDONESIA | 3   |
| ASIA   | JAPAN     | 2   |
| ASIA   | CHINA     | 1   |
| ASIA   | VIETNAM   | 0   |
+--------+-----------+-----+
[localhost:21000] > SELECT r1.r_name, r2.n_name, r2.POS
                  > FROM region r1 INNER JOIN r1.r_nations r2
                  > WHERE r1.r_name = 'ASIA' AND r2.POS BETWEEN 1
 and 3;
+--------+-----------+-----+
| r_name | n_name    | pos |
+--------+-----------+-----+
| ASIA   | CHINA     | 1   |
| ASIA   | JAPAN     | 2   |
| ASIA   | INDONESIA | 3   |
+--------+-----------+-----+
```

KEY and VALUE pseudocolumns

The MAP data type is suitable for representing sparse or wide data structures, where each row might only have entries for a small subset of named fields. Because the element names (the map keys) vary depending on the row, a query must be able to refer to both the key and the value parts of each key-value pair. The KEY and VALUE pseudocolumns let you refer to the parts of the key-value pair independently within the query, as *MAP_COLUMN*.KEY and *MAP_COLUMN*.VALUE.

The KEY must always be a scalar type, such as STRING, BIGINT, or TIMESTAMP. It can be NULL. Values of the KEY field are not necessarily unique within the same MAP. You apply any required DISTINCT, GROUP BY, and other clauses in the query, and loop through the result set to process all the values matching any specified keys.

The VALUE can be either a scalar type or another complex type. If the VALUE is a STRUCT, you can construct a qualified name *MAP_COLUMN*.VALUE.*STRUCT_FIELD* to refer to the individual fields inside the value part. If the VALUE is an ARRAY or another MAP, you must include another join condition that establishes a table alias for *MAP_COLUMN*.VALUE, and then construct another qualified name using that alias, for example *TABLE_ALIAS*.ITEM or *TABLE_ALIAS*.KEY and *TABLE_ALIAS*.VALUE

The following example shows different ways to access a MAP column using the KEY and VALUE pseudocolumns. The DETAILS column has a STRING first part with short, standardized values such as 'Recurring', 'Lucid', or 'Anxiety'. This is the “key” that is used to look up particular kinds of elements from the MAP. The second part, also a STRING, is a longer free-form explanation. Impala gives you the standard pseudocolumn names KEY and VALUE for the two parts, and you apply your own conventions and interpretations to the underlying values.

Note: If you find that the single-item nature of the VALUE makes it difficult to model your data accurately, the solution is typically to add some nesting to the complex type. For example, to have several sets of key-value pairs, make the column an ARRAY whose elements are MAP. To make a set of key-value pairs that holds more elaborate information, make a MAP column whose VALUE part contains an ARRAY or a STRUCT.

```
CREATE TABLE dream_journal
(
  dream_id BIGINT,
  details MAP <STRING,STRING>
)
STORED AS PARQUET;

-- What are all the types of dreams that are recorded?
SELECT DISTINCT details.KEY FROM dream_journal, dream_journal.de
tails;

-- How many lucid dreams were recorded?
-- Because there is no GROUP BY, we count the 'Lucid' keys across
 all rows.
SELECT COUNT(details.KEY)
  FROM dream_journal, dream_journal.details
WHERE details.KEY = 'Lucid';

-- Print a report of a subset of dreams, filtering based on both
the lookup key
-- and the detailed value.
SELECT dream_id, details.KEY AS "Dream Type", details.VALUE AS
 "Dream Summary"
  FROM dream_journal, dream_journal.details
WHERE
  details.KEY IN ('Happy', 'Pleasant', 'Joyous')
  AND details.VALUE LIKE '%childhood%';
```

The following example shows a more elaborate version of the previous table, where the VALUE part of the MAP entry is a STRUCT rather than a scalar type. Now instead of referring to the VALUE pseudocolumn directly, you use dot notation to refer to the STRUCT fields inside it.

```
CREATE TABLE better_dream_journal
(
  dream_id BIGINT,
  details MAP <STRING,STRUCT <summary: STRING, when_happened: TIM
ESTAMP, duration: DECIMAL(5,2), woke_up: BOOLEAN> >
)
STORED AS PARQUET;


-- Do more elaborate reporting and filtering by examining multipl
e attributes within the same dream.
SELECT dream_id, details.KEY AS "Dream
 Type", details.VALUE.summary AS "Dream
 Summary", details.VALUE.duration AS "Duration"
  FROM better_dream_journal, better_dream_journal.details
WHERE
  details.KEY IN ('Anxiety', 'Nightmare')
  AND details.VALUE.duration > 60
  AND details.VALUE.woke_up = TRUE;

-- Remember that if the ITEM or VALUE contains a STRUCT, you can
reference
-- the STRUCT fields directly without the .ITEM or .VALUE qual
ifier.
SELECT dream_id, details.KEY AS "Dream Type", details.summary AS
 "Dream Summary", details.duration AS "Duration"
  FROM better_dream_journal, better_dream_journal.details
WHERE
  details.KEY IN ('Anxiety', 'Nightmare')
  AND details.duration > 60
  AND details.woke_up = TRUE;
```

Loading data containing complex types

Because the Impala INSERT statement does not currently support creating new data with complex type columns, or copying existing complex type values from one table to another, you primarily use Impala to query Parquet/ORC tables with complex types where the data was inserted through Hive, or create tables with complex types where you already have existing Parquet/ORC data files.

Because the Impala INSERT statement does not currently support creating new data with complex type columns, or copying existing complex type values from one table to another, you primarily use Impala to query Parquet/ORC tables with complex types where the data was inserted through Hive, or create tables with complex types where you already have existing Parquet/ORC data files.

If you have existing Parquet data files containing complex types, located outside of any Impala or Hive table, such as data files created by Spark jobs, you can use an Impala CREATE TABLE ... STORED AS PARQUET statement, followed by an Impala LOAD DATA statement to move the data files into the table. As an alternative, you can use an Impala CREATE EXTERNAL TABLE statement to create a table pointing to the HDFS directory that already contains the Parquet or ORC data files.

The simplest way to get started with complex type data is to take a denormalized table containing duplicated values, and use an INSERT ... SELECT statement to copy the data into a Parquet table and condense the repeated values into complex types. With the Hive INSERT statement, you use the COLLECT_LIST(), NAMED_STRUCT(), and MAP() constructor functions within a GROUP BY query to produce the complex type values. COLLECT_LIST() turns a sequence of values into an ARRAY. NAMED_STRUCT() uses the first, third, and so on arguments as the field names for a STRUCT, to match the field names from the CREATE TABLE statement.

Note: Because Hive currently cannot construct individual rows using complex types through the INSERT ... VALUES syntax, you prepare the data in flat form in a separate table, then copy it to the table with complex columns using INSERT ... S ELECT and the complex type constructors.

Using complex types as nested types

The ARRAY, STRUCT, and MAP types can be the top-level types for “nested type” columns. That is, each of these types can contain other complex or scalar types, with multiple levels of nesting to a maximum depth of 100. For example, you can have an array of structures, a map containing other maps, a structure containing an array of other structures, and so on. At the lowest level, there are always scalar types making up the fields of a STRUCT, elements of an ARRAY, and keys and values of a MAP.

Schemas involving complex types typically use some level of nesting for the complex type columns.

For example, to model a relationship like a dimension table and a fact table, you typically use an ARRAY where each array element is a STRUCT. The STRUCT fields represent what would traditionally be columns in a separate joined table. It makes little sense to use a STRUCT as the top-level type for a column, because you could just make the fields of the STRUCT into regular table columns.

Perhaps the only use case for a top-level STRUCT would be to to allow STRUCT fields with the same name as columns to coexist in the same table. The following example shows how a table could have a column named ID, and two separate STRUCT fields also named ID. Because the STRUCT fields are always referenced using qualified names, the identical ID names do not cause a conflict.

```
CREATE TABLE struct_namespaces
(
  id BIGINT
  , s1 STRUCT < id: BIGINT, field1: STRING >
  , s2 STRUCT < id: BIGINT, when_happened: TIMESTAMP >
)
STORED AS PARQUET;

select id, s1.id, s2.id from struct_namespaces;
```

It is common to make the value portion of each key-value pair in a MAP a STRUCT, ARRAY of STRUCT, or other complex type variation. That way, each key in the MAP can be associated with a flexible and extensible data structure. The key values are not predefined ahead of time (other than by specifying their data type). Therefore, the MAP can accommodate a rapidly evolving schema, or sparse data structures where each row contains only a few data values drawn from a large set of possible choices.

Although you can use an ARRAY of scalar values as the top-level column in a table, such a simple array is typically of limited use for analytic queries. The only property of the array elements, aside from the element value, is the ordering sequence available through the POS pseudocolumn. To record any additional item about each array element, such as a TIMESTAMP or a symbolic name, you use an ARRAY of STRUCT rather than of scalar values.

If you are considering having multiple ARRAY or MAP columns, with related items under the same position in each ARRAY or the same key in each MAP, prefer to use a STRUCT to group all the related items into a single ARRAY or MAP. Doing so avoids the additional storage overhead and potential duplication of key values from having an extra complex type column. Also, because each ARRAY or MAP that you reference in the query SELECT list requires an additional join clause, minimizing the number of complex type columns also makes the query easier to read and maintain, relying more on dot notation to refer to the relevant fields rather than a sequence of join clauses.

For example, here is a table with several complex type columns all at the top level and containing only scalar types. To retrieve every data item for the row requires a separate join for each ARRAY or MAP column. The fields of the STRUCT can be referenced using dot notation, but there is no

real advantage to using the STRUCT at the top level rather than just making separate columns FIEL D1 and FIELD2.

```
CREATE TABLE complex_types_top_level
(
  id BIGINT,
  a1 ARRAY<INT>,
  a2 ARRAY<STRING>,
  s STRUCT<field1: INT, field2: STRING>,
-- Numeric lookup key for a string value.
  m1 MAP<INT,STRING>,
-- String lookup key for a numeric value.
  m2 MAP<STRING,INT>
)
STORED AS PARQUET;

describe complex_types_top_level;
+------+-----------------+
| name | type            |
+------+-----------------+
| id   | bigint          |
| a1   | array<int>      |
| a2   | array<string>   |
| s    | struct<         |
|      |   field1:int,   |
|      |   field2:string |
|      | >               |
| m1   | map<int,string> |
| m2   | map<string,int> |
+------+-----------------+
select
  id,
  a1.item,
  a2.item,
  s.field1,
  s.field2,
  m1.key,
  m1.value,
  m2.key,
  m2.value
from
  complex_types_top_level,
  complex_types_top_level.a1,
  complex_types_top_level.a2,
  complex_types_top_level.m1,
  complex_types_top_level.m2;
```

For example, here is a table with columns containing an ARRAY of STRUCT, a MAP where each key value is a STRUCT, and a MAP where each key value is an ARRAY of STRUCT.

```
CREATE TABLE nesting_demo
(
  user_id BIGINT,
  family_members ARRAY < STRUCT < name: STRING, email: STRING,
 date_joined: TIMESTAMP >>,
  foo map < STRING, STRUCT < f1: INT, f2: INT, f3: TIMESTAMP, f4:
 BOOLEAN >>,
  gameplay MAP < STRING , ARRAY < STRUCT <
    name: STRING, highest: BIGINT, lives_used: INT, total_spent:
 DECIMAL(16,2)
  >>>
)
```

```
STORED AS PARQUET;
```

The DESCRIBE statement rearranges the < and > separators and the field names within each STRU CT for easy readability:

```
DESCRIBE nesting_demo;
+----------------+-----------------------------+
| name           | type                        |
+----------------+-----------------------------+
| user_id        | bigint                      |
| family_members | array<struct<               |
|                |   name:string,              |
|                |   email:string,             |
|                |   date_joined:timestamp     |
|                | >>                          |
| foo            | map<string,struct<          |
|                |   f1:int,                   |
|                |   f2:int,                   |
|                |   f3:timestamp,             |
|                |   f4:boolean                |
|                | >>                          |
| gameplay       | map<string,array<struct<    |
|                |   name:string,              |
|                |   highest:bigint,           |
|                |   lives_used:int,           |
|                |   total_spent:decimal(16,2) |
|                | >>>                         |
+----------------+-----------------------------+
```

To query the complex type columns, you use join notation to refer to the lowest-level scalar values. If the value is an ARRAY element, the fully qualified name includes the ITEM pseudocolumn. If the value is inside a MAP, the fully qualified name includes the KEY or VALUE pseudocolumn. Each reference to a different ARRAY or MAP (even if nested inside another complex type) requires an additional join clause.

```
SELECT
-- The lone scalar field doesn't require any dot notation or join
 clauses.
    user_id
-- Retrieve the fields of a STRUCT inside an ARRAY.
-- The FAMILY_MEMBERS name refers to the FAMILY_MEMBERS table
alias defined later in the FROM clause.
  , family_members.item.name
  , family_members.item.email
  , family_members.item.date_joined
-- Retrieve the KEY and VALUE fields of a MAP, with the value bei
ng a STRUCT consisting of more fields.
-- The FOO name refers to the FOO table alias defined later in th
e FROM clause.
  , foo.key
  , foo.value.f1
  , foo.value.f2
  , foo.value.f3
  , foo.value.f4
-- Retrieve the KEY fields of a MAP, and expand the VALUE part in
to ARRAY items consisting of STRUCT fields.
-- The GAMEPLAY name refers to the GAMEPLAY table alias defined
later in the FROM clause (referring to the MAP item).
-- The GAME_N name refers to the GAME_N table alias defined la
ter in the FROM clause (referring to the ARRAY
-- inside the MAP item's VALUE part.)
  , gameplay.key
```

```
  , game_n.name
  , game_n.highest
  , game_n.lives_used
  , game_n.total_spent
FROM
    nesting_demo
  , nesting_demo.family_members AS family_members
  , nesting_demo.foo AS foo
  , nesting_demo.gameplay AS gameplay
  , nesting_demo.gameplay.value AS game_n;
```

Once you understand the notation to refer to a particular data item in the SELECT list, you can use the same qualified name to refer to that data item in other parts of the query, such as the WHERE clause, ORDER BY or GROUP BY clauses, or calls to built-in functions. For example, you might frequently retrieve the VALUE part of each MAP item in the SELECT list, while choosing the specific MAP items by running comparisons against the KEY part in the WHERE clause.

Allowing embedding complex types into other complex types

Before this release, collections and structs were supported in the select list, but nesting was only supported for the same kinds of complex types, i.e., structs in structs and collections in collections (including the case of maps in arrays and arrays in maps). Mixing different kinds of complex types with structs in collections or vice versa, was not supported. This release adds support for mixed complex types in the select list.

Limitations

- Zipping unnest is not supported for mixed complex types
- Structs containing collections are not (yet) supported in case of sorting

Note: Sorting of collections was not supported before this release. This release adds support for sorting collections that only contain fixed length types, but not for collections in structs. Sorting by the complex types is not supported yet.

Examples of querying a mixed complex type in the SELECT list

```
select
id,
struct_contains_arr,
struct_contains_map,
arr_contains_struct,
arr_contains_nested_struct,
struct_contains_nested_arr,
all_mix
from collection_struct_mix;
```

```
select id, nested_struct from complextypestbl;
```

Accessing complex type data in flattened form using views

The layout of complex and nested types is largely a physical consideration. The complex type columns reside in the same data files rather than in separate normalized tables, for your convenience in managing related data sets and performance in querying related data sets. You can use views to treat tables with complex types as if they were flattened. By putting the join logic and references to the complex type columns in the view definition, you can query the same tables using existing queries intended for tables containing only scalar columns. This technique also lets you use tables with complex types with BI tools that are not aware of the data types and query notation for accessing complex type columns.

For example, the variation of the TPC-H schema containing complex types has a table REGION. This table has 5 rows, corresponding to 5 regions such as NORTH AMERICA and AFRICA. Each

row has an ARRAY column, where each array item is a STRUCT containing details about a country in that region.

```
DESCRIBE region;
+-------------+-------------------------+
| name        | type                    |
+-------------+-------------------------+
| r_regionkey | smallint                |
| r_name      | string                  |
| r_comment   | string                  |
| r_nations   | array<struct<           |
|             |   n_nationkey:smallint, |
|             |   n_name:string,        |
|             |   n_comment:string      |
|             | >>                      |
+-------------+-------------------------+
```

The same data could be represented in traditional denormalized form, as a single table where the information about each region is repeated over and over, alongside the information about each country. The nested complex types let us avoid the repetition, while still keeping the data in a single table rather than normalizing across multiple tables.

To use this table with a JDBC or ODBC application that expected scalar columns, we could create a view that represented the result set as a set of scalar columns (three columns from the original table, plus three more from the STRUCT fields of the array elements). In the following examples, any column with an R_* prefix is taken unchanged from the original table, while any column with an N_* prefix is extracted from the STRUCT inside the ARRAY.

```
CREATE VIEW region_view AS
  SELECT
    r_regionkey,
    r_name,
    r_comment,
    array_field.item.n_nationkey AS n_nationkey,
    array_field.item.n_name AS n_name,
    array_field.n_comment AS n_comment
FROM
  region, region.r_nations AS array_field;
```

Then we point the application queries at the view rather than the original table. From the perspective of the view, there are 25 rows in the result set, one for each nation in each region, and queries can refer freely to fields related to the region or the nation.

```
-- Retrieve info such as the nation name from the original R_NAT
IONS array elements.
select n_name from region_view where r_name in ('EUROPE', 'ASIA
');
+----------------+
| n_name         |
+----------------+
| UNITED KINGDOM |
| RUSSIA         |
| ROMANIA        |
| GERMANY        |
| FRANCE         |
| VIETNAM        |
| CHINA          |
| JAPAN          |
| INDONESIA      |
| INDIA          |
+----------------+
```

```
-- UNITED STATES in AMERICA and UNITED KINGDOM in EUROPE.
SELECT DISTINCT r_name FROM region_view WHERE n_name LIKE 'UNITED
%';
+---------+
| r_name  |
+---------+
| AMERICA |
| EUROPE  |
+---------+

-- For conciseness, we only list some view columns in the SELECT
list.
-- SELECT * would bring back all the data, unlike SELECT *
-- queries on the original table with complex type columns.
SELECT r_regionkey, r_name, n_nationkey, n_name FROM region_view
 LIMIT 7;
+-------------+--------+-------------+----------------+
| r_regionkey | r_name | n_nationkey | n_name         |
+-------------+--------+-------------+----------------+
| 3           | EUROPE | 23          | UNITED KINGDOM |
| 3           | EUROPE | 22          | RUSSIA         |
| 3           | EUROPE | 19          | ROMANIA        |
| 3           | EUROPE | 7           | GERMANY        |
| 3           | EUROPE | 6           | FRANCE         |
| 2           | ASIA   | 21          | VIETNAM        |
| 2           | ASIA   | 18          | CHINA          |
+-------------+--------+-------------+----------------+
```

Tutorials and examples for complex types

The following examples illustrate the query syntax for some common use cases involving complex type columns:

Sample schema and data for experimenting with Impala complex types

The tables used for earlier examples of complex type syntax are trivial ones with no actual data. The more substantial examples of the complex type feature use these tables, adapted from the schema used for TPC-H testing:

```
SHOW TABLES;
+----------+
| name     |
+----------+
| customer |
| part     |
| region   |
| supplier |
+----------+

DESCRIBE customer;
+--------------+-----------------------------------+
| name         | type                              |
+--------------+-----------------------------------+
| c_custkey    | bigint                            |
| c_name       | string                            |
| c_address    | string                            |
| c_nationkey  | smallint                          |
| c_phone      | string                            |
| c_acctbal    | decimal(12,2)                     |
| c_mktsegment | string                            |
| c_comment    | string                            |
| c_orders     | array<struct<                     |
|              |   o_orderkey:bigint,              |
```

```
|               |   o_orderstatus:string,              |
|               |   o_totalprice:decimal(12,2),        |
|               |   o_orderdate:string,                |
|               |   o_orderpriority:string,            |
|               |   o_clerk:string,                    |
|               |   o_shippriority:int,                |
|               |   o_comment:string,                  |
|               |   o_lineitems:array<struct<          |
|               |     l_partkey:bigint,                |
|               |     l_suppkey:bigint,                |
|               |     l_linenumber:int,                |
|               |     l_quantity:decimal(12,2),        |
|               |     l_extendedprice:decimal(12,2),   |
|               |     l_discount:decimal(12,2),        |
|               |     l_tax:decimal(12,2),             |
|               |     l_returnflag:string,             |
|               |     l_linestatus:string,             |
|               |     l_shipdate:string,               |
|               |     l_commitdate:string,             |
|               |     l_receiptdate:string,            |
|               |     l_shipinstruct:string,           |
|               |     l_shipmode:string,               |
|               |     l_comment:string                 |
|               |   >>                                 |
|               | >>                                   |
+---------------+--------------------------------------+

DESCRIBE part;
+---------------+---------------+
| name          | type          |
+---------------+---------------+
| p_partkey     | bigint        |
| p_name        | string        |
| p_mfgr        | string        |
| p_brand       | string        |
| p_type        | string        |
| p_size        | int           |
| p_container   | string        |
| p_retailprice | decimal(12,2) |
| p_comment     | string        |
+---------------+---------------+

DESCRIBE region;
+-------------+-------------------------------------------------------
------------------+
| name        | type
                  |
+-------------+-------------------------------------------------
--------------------+
| r_regionkey | smallint
                  |
| r_name      | string
                  |
| r_comment   | string
                  |
| r_nations   | array<struct<n_nationkey:smallint,n_name:string
,n_comment:string>> |
+-------------+-----------------------------------------------
---------------------+

DESCRIBE supplier;
+-------------+-----------------------------------------------+
| name        | type                                          |
+-------------+-----------------------------------------------+
```

```
| s_suppkey   | bigint                                          |
| s_name      | string                                          |
| s_address   | string                                          |
| s_nationkey | smallint                                        |
| s_phone     | string                                          |
| s_acctbal   | decimal(12,2)                                   |
| s_comment   | string                                          |
| s_partsupps | array<struct<ps_partkey:bigint,                 |
|             | ps_availqty:int,ps_supplycost:decimal(12,2), |
|             | ps_comment:string>>                             |
+-------------+-------------------------------------------------+
```

The volume of data used in the following examples is:

```
SELECT count(*) FROM customer;
+----------+
| count(*) |
+----------+
| 150000   |
+----------+
SELECT count(*) FROM part;
+----------+
| count(*) |
+----------+
| 200000   |
+----------+

SELECT count(*) FROM region;
+----------+
| count(*) |
+----------+
| 5        |
+----------+
SELECT count(*) FROM supplier;
+----------+
| count(*) |
+----------+
| 10000    |
+----------+
```

Constructing Parquet / ORC files with complex columns using Hive

The following examples demonstrate the Hive syntax to transform flat data (tables with all scalar columns) into Parquet/ORC tables where Impala can query the complex type columns. Each example shows the full sequence of steps, including switching back and forth between Impala and Hive. Although the source table can use any file format, the destination table must use the Parquet/ORC file format. We take Parquet in the following examples. You can replace Parquet with ORC to do the same things in ORC file format.

Create table with ARRAY in Impala, load data in Hive, query in Impala:

This example shows the cycle of creating the tables and querying the complex data in Impala, and using Hive (either the hive shell or beeline) for the data loading step. The data starts in flattened, denormalized form in a text table. Hive writes the corresponding Parquet data, including an ARRAY column. Then Impala can run analytic queries on the Parquet table, using join notation to unpack the ARRAY column.

```
/* Initial DDL and loading of flat, denormalized data happens in
 impala-shell */CREATE TABLE flat_array (country STRING, city ST
RING);INSERT INTO flat_array VALUES
```

```
    ('Canada', 'Toronto') , ('Canada', 'Vancouver') , ('Canada',
 "St. John\'s")
  , ('Canada', 'Saint John') , ('Canada', 'Montreal') , ('Canada
', 'Halifax')
  , ('Canada', 'Winnipeg') , ('Canada', 'Calgary') , ('Canada',
 'Saskatoon')
  , ('Canada', 'Ottawa') , ('Canada', 'Yellowknife') , ('France',
 'Paris')
  , ('France', 'Nice') , ('France', 'Marseilles') , ('France',
 'Cannes')
  , ('Greece', 'Athens') , ('Greece', 'Piraeus') , ('Greece', 'Ha
nia')
  , ('Greece', 'Heraklion') , ('Greece', 'Rethymnon') , ('Gree
ce', 'Fira');

CREATE TABLE complex_array (country STRING, city ARRAY <STRING>)
STORED AS PARQUET;
```

```
/* Conversion to Parquet and complex and/or nested columns happe
ns in Hive */

INSERT INTO complex_array SELECT country, collect_list(city) FROM
 flat_array GROUP BY country;
Query ID = dev_20151108160808_84477ff2-82bd-4ba4-9a77-554fa7b8
c0cb
Total jobs = 1
Launching Job 1 out of 1
...
```

```
/* Back to impala-shell again for analytic queries */

REFRESH complex_array;
SELECT country, city.item FROM complex_array, complex_array.city
+---------+-------------+
| country | item        |
+---------+-------------+
| Canada  | Toronto     |
| Canada  | Vancouver   |
| Canada  | St. John's  |
| Canada  | Saint John  |
| Canada  | Montreal    |
| Canada  | Halifax     |
| Canada  | Winnipeg    |
| Canada  | Calgary     |
| Canada  | Saskatoon   |
| Canada  | Ottawa      |
| Canada  | Yellowknife |
| France  | Paris       |
| France  | Nice        |
| France  | Marseilles  |
| France  | Cannes      |
| Greece  | Athens      |
| Greece  | Piraeus     |
| Greece  | Hania       |
| Greece  | Heraklion   |
| Greece  | Rethymnon   |
| Greece  | Fira        |
+---------+-------------+
```

Create table with STRUCT and ARRAY in Impala, load data in Hive, query in Impala:

This example shows the cycle of creating the tables and querying the complex data in Impala, and using Hive (either the hive shell or beeline) for the data loading step. The data starts in flattened, denormalized form in a text table. Hive writes the corresponding Parquet data, including a STRUCT column with an ARRAY field. Then Impala can run analytic queries on the Parquet table, using join notation to unpack the ARRAY field from the STRUCT column.

```
/* Initial DDL and loading of flat, denormalized data happens in
 impala-shell */

CREATE TABLE flat_struct_array (continent STRING, country STRI
NG, city STRING);
INSERT INTO flat_struct_array VALUES
    ('North America', 'Canada', 'Toronto') , ('North America', 'C
anada', 'Vancouver')
  , ('North America', 'Canada', "St. John\'s") , ('North Americ
a', 'Canada', 'Saint John')
  , ('North America', 'Canada', 'Montreal') , ('North America',
'Canada', 'Halifax')
  , ('North America', 'Canada', 'Winnipeg') , ('North America',
 'Canada', 'Calgary')
  , ('North America', 'Canada', 'Saskatoon') , ('North America',
 'Canada', 'Ottawa')
  , ('North America', 'Canada', 'Yellowknife') , ('Europe', 'Fran
ce', 'Paris')
  , ('Europe', 'France', 'Nice') , ('Europe', 'France', 'Marse
illes')
  , ('Europe', 'France', 'Cannes') , ('Europe', 'Greece', 'Athen
s')
  , ('Europe', 'Greece', 'Piraeus') , ('Europe', 'Greece', 'Hani
a')
  , ('Europe', 'Greece', 'Heraklion') , ('Europe', 'Greece', 'Re
thymnon')
  , ('Europe', 'Greece', 'Fira');
CREATE TABLE complex_struct_array (continent STRING, country ST
RUCT <name: STRING, city: ARRAY <STRING> >) STORED AS PARQUET;
```

```
/* Conversion to Parquet and complex and/or nested columns happe
ns in Hive */

INSERT INTO complex_struct_array SELECT continent, named_struct('
name', country, 'city', collect_list(city)) FROM flat_array_array
 GROUP BY continent, country;
Query ID = dev_20151108163535_11a4fa53-0003-4638-97e6-ef13cdb8e09
e
Total jobs = 1
Launching Job 1 out of 1
...
```

```
/* Back to impala-shell again for analytic queries */

REFRESH complex_struct_array;
SELECT t1.continent, t1.country.name, t2.item
  FROM complex_struct_array t1, t1.country.city t2
+---------------+--------------+-------------+
| continent     | country.name | item        |
+---------------+--------------+-------------+
| Europe        | France       | Paris       |
| Europe        | France       | Nice        |
```

```
| Europe        | France       | Marseilles  |
| Europe        | France       | Cannes      |
| Europe        | Greece       | Athens      |
| Europe        | Greece       | Piraeus     |
| Europe        | Greece       | Hania       |
| Europe        | Greece       | Heraklion   |
| Europe        | Greece       | Rethymnon   |
| Europe        | Greece       | Fira        |
| North America | Canada       | Toronto     |
| North America | Canada       | Vancouver   |
| North America | Canada       | St. John's  |
| North America | Canada       | Saint John  |
| North America | Canada       | Montreal    |
| North America | Canada       | Halifax     |
| North America | Canada       | Winnipeg    |
| North America | Canada       | Calgary     |
| North America | Canada       | Saskatoon   |
| North America | Canada       | Ottawa      |
| North America | Canada       | Yellowknife |
+---------------+--------------+-------------+
```

Flattening normalized tables into a single table with complex types

One common use for complex types is to embed the contents of one table into another. The traditional technique of denormalizing results in a huge number of rows with some column values repeated over and over. With complex types, you can keep the same number of rows as in the original normalized table, and put all the associated data from the other table in a single new column.

In this flattening scenario, you might frequently use a column that is an ARRAY consisting of STRUCT elements, where each field within the STRUCT corresponds to a column name from the table that you are combining.

The following example shows a traditional normalized layout using two tables, and then an equivalent layout using complex types in a single table.

```
/* Traditional relational design */
-- This table just stores numbers, allowing us to look up details
 about the employee
-- and details about their vacation time using a three-table j
oin query.
CREATE table employee_vacations
(
  employee_id BIGINT,
  vacation_id BIGINT
)
STORED AS PARQUET;

-- Each kind of information to track gets its own "fact table".
CREATE table vacation_details
(
  vacation_id BIGINT,
  vacation_start TIMESTAMP,
  duration INT
)
STORED AS PARQUET;
-- Any time we print a human-readable report, we join with this
 table to
-- display info about employee #1234.
CREATE TABLE employee_contact
(
  employee_id BIGINT,
```

```
  name STRING,
  address STRING,
  phone STRING,
  email STRING,
  address_type STRING /* 'home', 'work', 'remote', etc. */
)
STORED AS PARQUET;

/* Equivalent flattened schema using complex types */

-- For analytic queries using complex types, we can bundle the di
mension table
-- and multiple fact tables into a single table.
CREATE TABLE employee_vacations_nested_types
(
-- We might still use the employee_id for other join queries.
-- The table needs at least one scalar column to serve as an ide
ntifier
-- for the complex type columns.
  employee_id BIGINT,

-- Columns of the VACATION_DETAILS table are folded into a STRUC
T.
-- We drop the VACATION_ID column because Impala doesn't need
-- synthetic IDs to join a complex type column.
-- Each row from the VACATION_DETAILS table becomes an array el
ement.
  vacation ARRAY < STRUCT <
    vacation_start: TIMESTAMP,
    duration: INT
  >>,

-- The ADDRESS_TYPE column, with a small number of predefined val
ues that are distinct
-- for each employee, makes the EMPLOYEE_CONTACT table a good
candidate to turn into a MAP,
-- with each row represented as a STRUCT. The string value from
 ADDRESS_TYPE becomes the
-- "key" (the anonymous first field) of the MAP.
  contact MAP < STRING, STRUCT <
    address: STRING,
    phone: STRING,
    email: STRING
  >>
)
STORED AS PARQUET;
```

Interchanging complex type tables and data files with Hive and other components

You can produce Parquet data files through several Hadoop components and APIs.

If you have a Hive-created Parquet table that includes ARRAY, STRUCT, or MAP columns, Impala can query that same table in Impala 2.3 and higher, subject to the usual restriction that all other columns are of data types supported by Impala, and also that the file type of the table must be Parquet.

If you have a Parquet data file produced outside of Impala, Impala can automatically deduce the appropriate table structure using the syntax CREATE TABLE ... LIKE PARQUET

'*HDFS_PATH_OF_PARQUET_FILE*'. In Impala 2.3 and higher, this feature works for Parquet files that include ARRAY, STRUCT, or MAP types.

```
/* In impala-shell, find the HDFS data directory of the original
 table.
DESCRIBE FORMATTED tpch_nested_parquet.customer;
...
| Location: | hdfs://localhost:20500/test-warehouse/tpch_nested
_parquet.db/customer   | NULL |
...
# In the Unix shell, find the path of any Parquet data file in
that HDFS directory.
$ hdfs dfs -ls hdfs://localhost:20500/test-warehouse/tpch_neste
d_parquet.db/customer
Found 4 items
-rwxr-xr-x   3 dev supergroup  171298918 2015-09-22 23:30 hdfs://
localhost:20500/blah/tpch_nested_parquet.db/customer/000000_0
...

/* Back in impala-shell, use the HDFS path in a CREATE TABLE LIKE
 PARQUET statement. */
CREATE TABLE customer_ctlp
  LIKE PARQUET 'hdfs://localhost:20500/blah/tpch_nested_parquet.d
b/customer/000000_0'
  STORED AS PARQUET;

/* Confirm that old and new tables have the same column layout, i
ncluding complex types. */
DESCRIBE tpch_nested_parquet.customer
+--------------+-----------------------------------+---------+
| name         | type                              | comment |
+--------------+-----------------------------------+---------+
| c_custkey    | bigint                            |         |
| c_name       | string                            |         |
| c_address    | string                            |         |
| c_nationkey  | smallint                          |         |
| c_phone      | string                            |         |
| c_acctbal    | decimal(12,2)                     |         |
| c_mktsegment | string                            |         |
| c_comment    | string                            |         |
| c_orders     | array<struct<                     |         |
|              |   o_orderkey:bigint,              |         |
|              |   o_orderstatus:string,           |         |
|              |   o_totalprice:decimal(12,2),     |         |
|              |   o_orderdate:string,             |         |
|              |   o_orderpriority:string,         |         |
|              |   o_clerk:string,                 |         |
|              |   o_shippriority:int,             |         |
|              |   o_comment:string,               |         |
|              |   o_lineitems:array<struct<       |         |
|              |     l_partkey:bigint,             |         |
|              |     l_suppkey:bigint,             |         |
|              |     l_linenumber:int,             |         |
|              |     l_quantity:decimal(12,2),     |         |
|              |     l_extendedprice:decimal(12,2), |         |
|              |     l_discount:decimal(12,2),     |         |
|              |     l_tax:decimal(12,2),          |         |
|              |     l_returnflag:string,          |         |
|              |     l_linestatus:string,          |         |
|              |     l_shipdate:string,            |         |
|              |     l_commitdate:string,          |         |
|              |     l_receiptdate:string,         |         |
|              |     l_shipinstruct:string,        |         |
```

```
|              |     l_shipmode:string,             |         |
|              |     l_comment:string               |         |
|              |   >>                               |         |
|              | >>                                 |         |
+--------------+------------------------------------+---------+

describe customer_ctlp;
+--------------+------------------------------------+---------
--------------------+
| name         | type                               | comment
                |
+--------------+------------------------------------+---------
--------------------+
| c_custkey    | bigint                             | Inferred
 from Parquet file. |
| c_name       | string                             | Inferred
 from Parquet file. |
| c_address    | string                             | Inferred
 from Parquet file. |
| c_nationkey  | int                                | Inferred
 from Parquet file. |
| c_phone      | string                             | Inferred
 from Parquet file. |
| c_acctbal    | decimal(12,2)                      | Inferred
 from Parquet file. |
| c_mktsegment | string                             | Inferred
 from Parquet file. |
| c_comment    | string                             | Inferred
 from Parquet file. |
| c_orders     | array<struct<                      | Inferred
 from Parquet file. |
|              |   o_orderkey:bigint,               |
                |
|              |   o_orderstatus:string,            |
                |
|              |   o_totalprice:decimal(12,2),      |
                |
|              |   o_orderdate:string,              |
                |
|              |   o_orderpriority:string,          |
                |
|              |   o_clerk:string,                  |
                |
|              |   o_shippriority:int,              |
                |
|              |   o_comment:string,                |
                |
|              |   o_lineitems:array<struct<        |
                |
|              |     l_partkey:bigint,              |
                |
|              |     l_suppkey:bigint,              |
                |
|              |     l_linenumber:int,              |
                |
|              |     l_quantity:decimal(12,2),      |
                |
|              |     l_extendedprice:decimal(12,2), |
                |
|              |     l_discount:decimal(12,2),      |
                |
|              |     l_tax:decimal(12,2),           |
                |
```

```
|               |     l_returnflag:string,           |
                |
|               |     l_linestatus:string,           |
                |
|               |     l_shipdate:string,             |
                |
|               |     l_commitdate:string,           |
                |
|               |     l_receiptdate:string,          |
                |
|               |     l_shipinstruct:string,         |
                |
|               |     l_shipmode:string,             |
                |
|               |     l_comment:string               |
                |
|               |   >>                               |
                |
|               | >>                                 |
                |
+---------------+------------------------------------+---------
--------------------+
```

Querying arrays

Describes how to use UNNEST function to query arrays. ARRAY data types represent collections with arbitrary numbers of elements, where each element is the same type.

Querying arrays using JOIN and UNNEST

You can query arrays by making a join between the table and the array inside the table. This approach is improved with the introduction of the UNNEST function in the SELECT list or in the FROM clause in the SELECT statement. When you use UNNEST, you can provide more than one array in the SELECT statement. If you use JOINs for querying arrays it will yield a “joining unnest” however the latter will provide a “zipping unnest”.

Example of querying arrays using JOIN

Use JOIN in cases where you must join unnest of multiple arrays. However if you must zip unnest then use the newly implemented UNNEST function.

Here is an example of a SELECT statement that uses JOINs to query an array.

```
SELECT id, arr1.item, arr2.item FROM tbl_name tbl, tbl.arr1, tbl.arr2;
ID, ARR1.ITEM, ARR2.ITEM
```

```
[1, 1, 10]
[1, 1, 11]
[1, 2, 10]
[1, 2, 11]
[1, 3, 10]
[1, 3, 11]
```

Note:

The test data used in this example is ID: 1, arr1: {1, 2, 3}, arr2: {10, 11}

Examples of querying arrays using UNNEST

You can use one of the two different syntaxes shown here to unnest multiple arrays in one query. This results in the items of the arrays being zipped together instead of joining.

- ISO:SQL 2016 compliant syntax:

```
SELECT a1.item, a2.item
FROM complextypes_arrays t, UNNEST(t.arr1, t.arr2) AS (a1, a2);
```

- Postgres compatible syntax:

```
SELECT UNNEST(arr1), UNNEST(arr2) FROM complextypes_arrays;
```

Unnest operator in SELECT list

```
SELECT id, unnest(arr1), unnest(arr2) FROM tbl_name;
```

Unnest operator in FROM clause

```
SELECT id, arr1.item, arr2.item FROM tbl_name tbl_alias, UNNEST(tbl_alias.ar
r1, tbl_alias.arr2);
```

This new functionality would zip the arrays next to each other as shown here.

```
ID, ARR1.ITEM, ARR2.ITEM
[1, 1, 10]
[1, 2, 11]
[1, 3, NULL]
```

Note, that arr2 is shorter than arr1 so the "missing" items in its column will be filled with NULLs.

Note: The test data used in this example is ID: 1, arr1: {1, 2, 3}, arr2: {10, 11}

Limitations in Using UNNEST

- Only arrays from the same table can be zipping unnested
- The old (joining) and the new (zipping) unnests cannot be used together
- You can add a WHERE filter on an unnested item only if you add a wrapper SELECT and do the filtering

  Example:

```
SELECT id, arr1_unnest FROM (SELECT id, unnest(arr1) as arr1_unnest FROM
 tbl_name) WHERE arr1_unnest < 10;
```

Using ARRAY columns in the SELECT list

Prior to this release to look into the content of an array you had to unnest the array either by the joining syntax or by using the zipping UNNEST operator as shown in the following example:

```
SELECT unnest(IDs), unnest(NAMES) FROM table_name;
```

This release adds support to return ARRAYs as STRINGs (JSON arrays) in the SELECT list, for example:

```
select id, int_array from functional_parquet.complextypestbl where id = 1;
returns: 1, “[1,2,3]”
```

Returning ARRAYs from inline or HMS views is also supported. These arrays can be used both in the select list or as relative table references.

```
select id, int_array from (select id, int_array from complextypestbl) s;
```

Though STRUCTs are already supported, ARRAYs and STRUCTs nested within each other are not supported yet. Using them as non-relative table references is also not supported yet.

Zipping unnest on arrays from views

Before this release, zipping unnest functionality worked for arrays only in "tables" but did not support "views" as a source. This release improves that support further by accepting inputs from views as well.

UNNEST() on array columns

You can use UNNEST() on array columns in two ways. Using one of these two ways results in the items of the arrays being zipped together instead of joining.

- ISO:SQL 2016 compliant syntax

```
SELECT a1.item, a2.item
FROM complextypes_arrays t, UNNEST(t.arr1, t.arr2) AS (a1, a2);
```

- Postgres compatible syntax

```
SELECT UNNEST(arr1), UNNEST(arr2) FROM complextypes_arrays;
```

When unnesting multiple arrays with zipping unnest, the i'th item of one array will be placed next to the i'th item of the other arrays in the results. If the size of the arrays is not equal then the shorter arrays will be filled with NULL values up to the size of the longest array as shown in the following example:

The test data used in this example is arr1: {1, 2, 3}, arr2: {11, 12}

After running any of the queries listed in the examples, the result will be as shown here:

```
arr1 arr2
[1, 11]
[2, 12]
[3, NULL]
```

Example of an UNNEST() in an inline view using SELECT/FILTER of the inline view

In the following example the filter is not in the SELECT query that creates the inline view but a level above that.

```
SELECT id, ac1, ac2 FROM
    (SELECT id, UNNEST(array_col1) AS ac1, UNNEST(array_col2) AS ac2 FROM
some_view)
WHERE id <10;
```

Impala SQL literals

Each of the Impala data types has corresponding notation for literal values of that type. You specify literal values in SQL statements, such as in the SELECT list or WHERE clause of a query, or as an argument to a function call.

Numeric literals

To write literals for the integer types (TINYINT, SMALLINT, INT, and BIGINT), use a sequence of digits with optional leading zeros.

To write literals for the floating-point types (DECIMAL, FLOAT, and DOUBLE), use a sequence of digits with an optional decimal point (. character). To preserve accuracy during arithmetic expressions, Impala interprets floating-point literals as the DECIMAL type with the smallest appropriate precision and scale, until required by the context to convert the result to FLOAT or DOUBLE.

Integer values are promoted to floating-point when necessary, based on the context.

You can also use exponential notation by including an e character. For example, 1e6 is 1 times 10 to the power of 6 (1 million). A number in exponential notation is always interpreted as floating-point.

When Impala encounters a numeric literal, it considers the type to be the “smallest” that can accurately represent the value. The type is promoted to larger or more accurate types if necessary, based on subsequent parts of an expression.

STRING literals

String literals are quoted using either single or double quotation marks. You can use either kind of quotes for string literals, even both kinds for different literals within the same statement.

Quoted literals are considered to be of type STRING. To use quoted literals in contexts requiring a CHAR or VARC HAR value, CAST() the literal to a CHAR or VARCHAR of the appropriate length.

Escaping special characters:

To encode special characters within a string literal, precede them with the backslash (\) escape character:

- \t represents a tab.
- \n represents a newline or linefeed. This might cause extra line breaks in `impala-shell` output.
- \r represents a carriage return. This might cause unusual formatting (making it appear that some content is overwritten) in `impala-shell` output.
- \b represents a backspace. This might cause unusual formatting (making it appear that some content is overwritten) in `impala-shell` output.
- \0 represents an ASCII nul character (not the same as a SQL NULL). This might not be visible in `impala-shell` output.
- \Z represents a DOS end-of-file character. This might not be visible in `impala-shell` output.
- \% and _ can be used to escape wildcard characters within the string passed to the LIKE operator.
- \ followed by 3 octal digits represents the ASCII code of a single character; for example, \101 is ASCII 65, the character A.
- Use two consecutive backslashes (\\) to prevent the backslash from being interpreted as an escape character.
- Use the backslash to escape single or double quotation mark characters within a string literal, if the literal is enclosed by the same type of quotation mark.
- If the character following the \ does not represent the start of a recognized escape sequence, the character is passed through unchanged.

Quotes within quotes:

To include a single quotation character within a string value, enclose the literal with either single or double quotation marks, and optionally escape the single quote as a \' sequence. Earlier releases required escaping a single quote inside double quotes. Continue using escape sequences in this case if you also need to run your SQL code on older versions of Impala.

To include a double quotation character within a string value, enclose the literal with single quotation marks, no escaping is necessary in this case. Or, enclose the literal with double quotation marks and escape the double quote as a \" sequence.

```
[localhost:21000] > select "What\'s happening?" as single_within_double,
                  >        'I\'m not sure.' as single_within_single,
```

```
                    >        "Homer wrote \"The Iliad\"." as double_within_d
ouble,
                    >        'Homer also wrote "The Odyssey".' as double_wi
thin_single;
+----------------------+----------------------+-------------------------+-
--------------------------------+
| single_within_double | single_within_single | double_within_double    |
 double_within_single           |
+----------------------+----------------------+-------------------------
+--------------------------------+
| What's happening?    | I'm not sure.        | Homer wrote "The Iliad". | H
omer also wrote "The Odyssey". |
+----------------------+----------------------+-------------------------+-
--------------------------------+
```

Field terminator character in CREATE TABLE:

Note: The CREATE TABLE clauses FIELDS TERMINATED BY, ESCAPED BY, and LINES TERMINATED BY have special rules for the string literal used for their argument, because they all require a single character. You can use a regular character surrounded by single or double quotation marks, an octal sequence such as '\054' (representing a comma), or an integer in the range '-127'..'128' (with quotation marks but no backslash), which is interpreted as a single-byte ASCII character. Negative values are subtracted from 256; for example, FIELDS TERMINATED BY '-2' sets the field delimiter to ASCII code 254, the “Icelandic Thorn” character used as a delimiter by some data formats.

impala-shell considerations:

When dealing with output that includes non-ASCII or non-printable characters such as linefeeds and backspaces, use the `impala-shell` options to save to a file, turn off pretty printing, or both rather than relying on how the output appears visually.

BOOLEAN literals

For BOOLEAN values, the literals are TRUE and FALSE, with no quotation marks and case-insensitive.

TIMESTAMP literals

Impala automatically converts STRING literals of the correct format into TIMESTAMP values. Timestamp values are accepted in the format 'yyyy#MM#dd HH:mm:ss.SSSSSS', and can consist of just the date, or just the time, with or without the fractional second portion. For example, you can specify TIMESTAMP values such as '1966#07#30', '08:30:00', or '1985#09#25 17:45:30.005'.

Leading zeroes are not required in the numbers representing the date component, such as month and date, or the time component, such as hour, minute, and second. For example, Impala accepts both '2018#1#1 01:02:03' and '2018-01-01 1:2:3' as valid.

In STRING to TIMESTAMP conversions, leading and trailing white spaces, such as a space, a tab, a newline, or a carriage return, are ignored. For example, Impala treats the following as equivalent: '1999#12#01 01:02:03 ', ' 1999#12#01 01:02:03', '1999#12#01 01:02:03\r\n\t'.

When you convert or cast a STRING literal to TIMESTAMP, you can use the following separators between the date part and the time part:

- One or more space characters

  Example: CAST('2001-01-09 01:05:01' AS TIMESTAMP)
- The character “T”

  Example: CAST('2001-01-09T01:05:01' AS TIMESTAMP)

You can also use INTERVAL expressions to add or subtract from timestamp literal values, such as CAST('1966#07#30' AS TIMESTAMP) + INTERVAL 5 YEARS + INTERVAL 3 DAYS.

Depending on your data pipeline, you might receive date and time data as text, in notation that does not exactly match the format for Impala TIMESTAMP literals. See *Impala Date and Time Functions* for functions that can convert between a variety of string literals (including different field order, separators, and timezone notation) and equivalent TIMESTAMP or numeric values.

DATE literals

The DATE literals are in the form of DATE'YYYY-MM-DD'. For example, DATE '2013-01-01'

NULL

The notion of NULL values is familiar from all kinds of database systems, but each SQL dialect can have its own behavior and restrictions on NULL values. For Big Data processing, the precise semantics of NULL values are significant: any misunderstanding could lead to inaccurate results or misformatted data, that could be time-consuming to correct for large data sets.

- NULL is a different value than an empty string. The empty string is represented by a string literal with nothing inside, "" or ''.
- In a delimited text file, the NULL value is represented by the special token \N.
- When Impala inserts data into a partitioned table, and the value of one of the partitioning columns is NULL or the empty string, the data is placed in a special partition that holds only these two kinds of values. When these values are returned in a query, the result is NULL whether the value was originally NULL or an empty string. This behavior is compatible with the way Hive treats NULL values in partitioned tables. Hive does not allow empty strings as partition keys, and it returns a string value such as __HIVE_DEFAULT_PARTITION__ instead of NULL when such values are returned from a query. For example:

```
create table t1 (i int) partitioned by (x int, y string);
-- Select an INT column from another table, with all rows going into a spe
cial HDFS subdirectory
-- named __HIVE_DEFAULT_PARTITION__. Depending on whether one or both of
the partitioning keys
-- are null, this special directory name occurs at different levels of the
 physical data directory
-- for the table.
insert into t1 partition(x=NULL, y=NULL) select c1 from some_other_table;
insert into t1 partition(x, y=NULL) select c1, c2 from some_other_table;
insert into t1 partition(x=NULL, y) select c1, c3  from some_other_table;
```

- There is no NOT NULL clause when defining a column to prevent NULL values in that column.
- There is no DEFAULT clause to specify a non-NULL default value.
- If an INSERT operation mentions some columns but not others, the unmentioned columns contain NULL for all inserted rows.
- In Impala 1.2.1 and higher, all NULL values come at the end of the result set for ORDER BY ... ASC queries, and at the beginning of the result set for ORDER BY ... DESC queries. In effect, NULL is considered greater than all other values for sorting purposes. The original Impala behavior always put NULL values at the end, even for ORDER BY ... DESC queries. The new behavior in Impala 1.2.1 makes Impala more compatible with other popular database systems. In Impala 1.2.1 and higher, you can override or specify the sorting behavior for NULL by adding the clause NULLS FIRST or NULLS LAST at the end of the ORDER BY clause.

  Note: Because the NULLS FIRST and NULLS LAST keywords are not currently available in Hive queries, any views you create using those keywords will not be available through Hive.

- In all other contexts besides sorting with ORDER BY, comparing a NULL to anything else returns NULL, making the comparison meaningless. For example, 10 > NULL produces NULL, 10 < NULL also produces NULL, 5 BETWEEN 1 AND NULL produces NULL, and so on.

Several built-in functions serve as shorthand for evaluating expressions and returning NULL, 0, or some other substitution value depending on the expression result: ifnull(), isnull(), nvl(), nullif(), nullifzero(), and zeroifnull().

Kudu considerations:

Columns in Kudu tables have an attribute that specifies whether or not they can contain NULL values. A column with a NULL attribute can contain nulls. A column with a NOT NULL attribute cannot contain any nulls, and an INSE RT, UPDATE, or UPSERT statement will skip any row that attempts to store a null in a column designated as NOT NULL. Kudu tables default to the NULL setting for each column, except columns that are part of the primary key.

In addition to columns with the NOT NULL attribute, Kudu tables also have restrictions on NULL values in columns that are part of the primary key for a table. No column that is part of the primary key in a Kudu table can contain any NULL values.

Related Information

Impala SQL data types

Impala Shell configuration options

Impala SQL operators

SQL operators are used primarily in the WHERE clause to perform operations, such as comparison operations and arithmetic operations.

Arithmetic operators

The arithmetic operators use expressions with a left-hand argument, the operator, and then (in most cases) a right-hand argument.

Syntax:

```
LEFT_HAND_ARG BINARY_OPERATOR RIGHT_HAND_ARG
UNARY_OPERATOR SINGLE_ARG
```

- + and -: Can be used either as unary or binary operators.
  - With unary notation, such as +5, -2.5, or -*COL_NAME*, they multiply their single numeric argument by +1 or -1. Therefore, unary + returns its argument unchanged, while unary - flips the sign of its argument. Although you can double up these operators in expressions such as ++5 (always positive) or -+2 or +-2 (both always negative), you cannot double the unary minus operator because -- is interpreted as the start of a comment. (You can use a double unary minus operator if you separate the - characters, for example with a space or parentheses.)
  - With binary notation, such as 2+2, 5-2.5, or *COL1* + *COL2*, they add or subtract respectively the right-hand argument to (or from) the left-hand argument. Both arguments must be of numeric types.
- * and /: Multiplication and division respectively. Both arguments must be of numeric types.

  When multiplying, the shorter argument is promoted if necessary (such as SMALLINT to INT or BIGINT, or FLOAT to DOUBLE), and then the result is promoted again to the next larger type. Thus, multiplying a TINY INT and an INT produces a BIGINT result. Multiplying a FLOAT and a FLOAT produces a DOUBLE result. Multiplying a FLOAT and a DOUBLE or a DOUBLE and a DOUBLE produces a DECIMAL(38,17), because DECIMAL values can represent much larger and more precise values than DOUBLE.

  When dividing, Impala always treats the arguments and result as DOUBLE values to avoid losing precision. If you need to insert the results of a division operation into a FLOAT column, use the CAST() function to convert the result to the correct type.
- DIV: Integer division. Arguments are not promoted to a floating-point type, and any fractional result is discarded. For example, 13 DIV 7 returns 1, 14 DIV 7 returns 2, and 15 DIV 7 returns 2. This operator is the same as the QUOTIENT() function.
- %: Modulo operator. Returns the remainder of the left-hand argument divided by the right-hand argument. Both arguments must be of one of the integer types.

- &, |, ~, and ^: Bitwise operators that return the logical AND, logical OR, NOT, or logical XOR (exclusive OR) of their argument values. Both arguments must be of one of the integer types. If the arguments are of different type, the argument with the smaller type is implicitly extended to match the argument with the longer type.

You can chain a sequence of arithmetic expressions, optionally grouping them with parentheses.

The arithmetic operators generally do not have equivalent calling conventions using functional notation. For example, prior to Impala 2.2, there is no MOD() function equivalent to the % modulo operator. Conversely, there are some arithmetic functions that do not have a corresponding operator. For example, for exponentiation you use the POW() function, but there is no ** exponentiation operator. See *Impala Mathematical Functions* for the arithmetic functions you can use.

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
| EUROPE      | 19               |
| EUROPE      | 22               |
```

```
| EUROPE      | 23                |
| MIDDLE EAST | 4                 |
| MIDDLE EAST | 10                |
| MIDDLE EAST | 11                |
| MIDDLE EAST | 13                |
| MIDDLE EAST | 20                |
+-------------+-------------------+

select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

The following example shows how to do an arithmetic operation using a numeric field of a STRUCT type that is an item within an ARRAY column. Once the scalar numeric value R_NATIONKEY is extracted, it can be used in an arithmetic expression, such as multiplying by 10:

```
-- The SMALLINT is a field within an array of structs.
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+
```

```
-- When we refer to the scalar value using dot notation,
-- we can use arithmetic and comparison operators on it
-- like any other number.
select r_name, nation.item.n_name, nation.item.n_nationkey * 10
  from region, region.r_nations as nation
where nation.item.n_nationkey < 5;
+-------------+-------------+-----------------------------+
| r_name      | item.n_name | nation.item.n_nationkey * 10 |
+-------------+-------------+-----------------------------+
| AMERICA     | CANADA      | 30                          |
| AMERICA     | BRAZIL      | 20                          |
| AMERICA     | ARGENTINA   | 10                          |
| MIDDLE EAST | EGYPT       | 40                          |
| AFRICA      | ALGERIA     | 0                           |
+-------------+-------------+-----------------------------+
```

BETWEEN operator

In a WHERE clause, compares an expression to both a lower and upper bound. The comparison is successful is the expression is greater than or equal to the lower bound, and less than or equal to the upper bound. If the bound values are switched, so the lower bound is greater than the upper bound, does not match any values.

Syntax:

```
EXPRESSION BETWEEN LOWER_BOUND AND UPPER_BOUND
```

Data types: Typically used with numeric data types. Works with any data type, although not very practical for BOOLEAN values. (BETWEEN false AND true will match all BOOLEAN values.) Use CAST() if necessary to ensure the lower and upper bound values are compatible types. Call string or date/time functions if necessary to extract or transform the relevant portion to compare, especially if the value can be transformed into a number.

Usage notes:

Be careful when using short string operands. A longer string that starts with the upper bound value will not be included, because it is considered greater than the upper bound. For example, BETWEEN 'A' and 'M' would not match the string value 'Midway'. Use functions such as upper(), lower(), substr(), trim(), and so on if necessary to ensure the comparison works as expected.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

Examples:

The following example shows how to do a BETWEEN comparison using a numeric field of a STRUCT type that is an item within an ARRAY column. Once the scalar numeric value R_NATIONKEY is extracted, it can be used in a comparison operator:

```
-- The SMALLINT is a field within an array of structs.
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
```

```
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

-- When we refer to the scalar value using dot notation,
-- we can use arithmetic and comparison operators on it
-- like any other number.
select r_name, nation.item.n_name, nation.item.n_nationkey
from region, region.r_nations as nation
where nation.item.n_nationkey between 3 and 5
+-------------+-------------+------------------+
| r_name      | item.n_name | item.n_nationkey |
+-------------+-------------+------------------+
| AMERICA     | CANADA      | 3                |
| MIDDLE EAST | EGYPT       | 4                |
| AFRICA      | ETHIOPIA    | 5                |
+-------------+-------------+------------------+
```

Comparison operators

Impala supports the familiar comparison operators for checking equality and sort order for the column data types:

Syntax:

```
LEFT_HAND_EXPRESSION COMPARISON_OPERATOR RIGHT_HAND_EXPRESSION
```

- =, !=, <>: apply to all scalar types.
- <, <=, >, >=: apply to all scalar types; for BOOLEAN, TRUE is considered greater than FALSE.

Alternatives:

The IN and BETWEEN operators provide shorthand notation for expressing combinations of equality, less than, and greater than comparisons with a single operator.

Because comparing any value to NULL produces NULL rather than TRUE or FALSE, use the IS NULL and IS NOT NULL operators to check if a value is NULL or not.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

The following example shows how to do an arithmetic operation using a numeric field of a STRUCT type that is an item within an ARRAY column. Once the scalar numeric value R_NATIONKEY is extracted, it can be used with a comparison operator such as <:

```
-- The SMALLINT is a field within an array of structs.
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
```

```
+-------------+-------------------------+---------+

-- When we refer to the scalar value using dot notation,
-- we can use arithmetic and comparison operators on it
-- like any other number.
select r_name, nation.item.n_name, nation.item.n_nationkey
from region, region.r_nations as nation
where nation.item.n_nationkey < 5
+-------------+-------------+------------------+
| r_name      | item.n_name | item.n_nationkey |
+-------------+-------------+------------------+
| AMERICA     | CANADA      | 3                |
| AMERICA     | BRAZIL      | 2                |
| AMERICA     | ARGENTINA   | 1                |
| MIDDLE EAST | EGYPT       | 4                |
| AFRICA      | ALGERIA     | 0                |
+-------------+-------------+------------------+
```

EXISTS operator

The EXISTS operator tests whether a subquery returns any results. You typically use it to find values from one table that have corresponding values in another table.

The converse, NOT EXISTS, helps to find all the values from one table that do not have any corresponding values in another table.

Syntax:

```
EXISTS (SUBQUERY)
NOT EXISTS (SUBQUERY)
```

Usage notes:

The subquery can refer to a different table than the outer query block, or the same table. For example, you might use EXISTS or NOT EXISTS to check the existence of parent/child relationships between two columns of the same table.

You can also use operators and function calls within the subquery to test for other kinds of relationships other than strict equality. For example, you might use a call to COUNT() in the subquery to check whether the number of matching values is higher or lower than some limit. You might call a UDF in the subquery to check whether values in one table matches a hashed representation of those same values in a different table.

NULL considerations:

If the subquery returns any value at all (even NULL), EXISTS returns TRUE and NOT EXISTS returns false.

Restrictions:

Correlated subqueries used in EXISTS and IN operators cannot include a LIMIT clause.

Prior to Impala 2.6, the NOT EXISTS operator required a correlated subquery. In Impala 2.6 and higher, NOT EXIS TS works with uncorrelated queries also.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

ILIKE operator

A case-insensitive comparison operator for STRING data, with basic wildcard capability using _ to match a single character and % to match any characters. The argument expression must match the entire string value. Typically, it is more efficient to put any % wildcard match at the end of the string.

This operator, available in Impala 2.5 and higher, is the equivalent of the LIKE operator, but with case-insensitive comparisons.

Syntax:

```
STRING_EXPRESSION ILIKE WILDCARD_EXPRESSION
STRING_EXPRESSION NOT ILIKE WILDCARD_EXPRESSION
```

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

IN operator

The IN operator compares an argument value to a set of values, and returns TRUE if the argument matches any value in the set. The NOT IN operator reverses the comparison, and checks if the argument value is not part of a set of values.

Syntax:

```
EXPRESSION IN (EXPRESSION [, EXPRESSION])
EXPRESSION IN (SUBQUERY)

EXPRESSION NOT IN (EXPRESSION [, EXPRESSION])
EXPRESSION NOT IN (SUBQUERY)
```

The left-hand expression and the set of comparison values must be of compatible types.

The left-hand expression must consist only of a single value, not a tuple. Although the left-hand expression is typically a column name, it could also be some other value. For example, the WHERE clauses WHERE id IN (5) and WHERE 5 IN (id) produce the same results.

The set of values to check against can be specified as constants, function calls, column names, or other expressions in the query text. The maximum number of expressions in the IN list is 9999. (The maximum number of elements of a single expression is 10,000 items, and the IN operator itself counts as one.)

In Impala 2.0 and higher, the set of values can also be generated by a subquery. IN can evaluate an unlimited number of results using a subquery.

Usage notes:

Any expression using the IN operator could be rewritten as a series of equality tests connected with OR, but the IN syntax is often clearer, more concise, and easier for Impala to optimize. For example, with partitioned tables, queries frequently use IN clauses to filter data by comparing the partition key columns to specific values.

NULL considerations:

If there really is a matching non-null value, IN returns TRUE:

If the searched value is not found in the comparison values, and the comparison values include NULL, the result is NULL:

If the left-hand argument is NULL, IN always returns NULL. This rule applies even if the comparison values include NULL.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

Restrictions:

Correlated subqueries used in EXISTS and IN operators cannot include a LIMIT clause.

IREGEXP operator

Tests whether a value matches a regular expression, using case-insensitive string comparisons. Uses the POSIX regular expression syntax where ^ and $ match the beginning and end of the string, . represents any single character, * represents a sequence of zero or more items, + represents a sequence of one or more items, ? produces a non-greedy match, and so on.

This operator, available in Impala 2.5 and higher, is the equivalent of the REGEXP operator, but with case-insensitive comparisons.

Syntax:

```
STRING_EXPRESSION IREGEXP REGULAR_EXPRESSION
```

Usage notes:

The | symbol is the alternation operator, typically used within () to match different sequences. The () groups do not allow backreferences. To retrieve the part of a value matched within a () section, use the regexp_extract() built-in function. (Currently, there is not any case-insensitive equivalent for the regexp_extract() function.)

In Impala 1.3.1 and higher, the REGEXP and RLIKE operators now match a regular expression string that occurs anywhere inside the target string, the same as if the regular expression was enclosed on each side by .*. Previously, these operators only succeeded when the regular expression matched the entire target string. This change improves compatibility with the regular expression support for popular database systems. There is no change to the behavior of the regexp_extract() and regexp_replace() built-in functions.

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

IS DISTINCT FROM operator

The IS DISTINCT FROM operator, and its converse the IS NOT DISTINCT FROM operator, test whether or not values are identical. IS NOT DISTINCT FROM is similar to the = operator, and IS DISTINCT FROM is similar to the != operator, except that NULL values are treated as identical. Therefore, IS NOT DISTINCT FROM returns true rather than NULL, and IS DISTINCT FROM returns false rather than NULL, when comparing two NULL values. If one of the values being compared is NULL and the other is not, IS DISTINCT FROM returns true and IS NOT DISTINCT FROM returns false, again instead of returning NULL in both cases.

Syntax:

```
EXPRESSION1 IS DISTINCT FROM EXPRESSION2

EXPRESSION1 IS NOT DISTINCT FROM EXPRESSION2
EXPRESSION1 <=> EXPRESSION2
```

The operator <=> is an alias for IS NOT DISTINCT FROM. It is typically used as a NULL-safe equality operator in join queries. That is, A <=> B is true if A equals B or if both A and B are NULL.

Usage notes:

This operator provides concise notation for comparing two values and always producing a true or false result, without treating NULL as a special case. Otherwise, to unambiguously distinguish between two values requires a compound expression involving IS [NOT] NULL tests of both operands in addition to the = or != operator.

The <=> operator, used like an equality operator in a join query, is more efficient than the equivalent clause: IF (A IS NULL OR B IS NULL, A IS NULL AND B IS NULL, A = B). The <=> operator can use a hash join, while the IF expression cannot.

IS NULL operator

The IS NULL operator, and its converse the IS NOT NULL operator, test whether a specified value is NULL. Because using NULL with any of the other comparison operators such as = or != also returns NULL rather than TRUE or FALSE, you use a special-purpose comparison operator to check for this special condition.

In Impala 2.1.1 and higher, you can use the operators IS UNKNOWN and IS NOT UNKNOWN as synonyms for IS NULL and IS NOT NULL, respectively.

Syntax:

```
EXPRESSION IS NULL
EXPRESSION IS NOT NULL

EXPRESSION IS UNKNOWN
EXPRESSION IS NOT UNKNOWN
```

Usage notes:

In many cases, NULL values indicate some incorrect or incomplete processing during data ingestion or conversion. You might check whether any values in a column are NULL, and if so take some followup action to fill them in.

With sparse data, often represented in “wide” tables, it is common for most values to be NULL with only an occasional non-NULL value. In those cases, you can use the IS NOT NULL operator to identify the rows containing any data at all for a particular column, regardless of the actual value.

With a well-designed database schema, effective use of NULL values and IS NULL and IS NOT NULL operators can save having to design custom logic around special values such as 0, -1, 'N/A', empty string, and so on. NULL lets you distinguish between a value that is known to be 0, false, or empty, and a truly unknown value.

Complex type considerations:

The IS [NOT] UNKNOWN operator, as with the IS [NOT] NULL operator, is not applicable to complex type columns (STRUCT, ARRAY, or MAP). Using a complex type column with this operator causes a query error.

IS TRUE operator

This variation of the IS operator tests for truth or falsity, with right-hand arguments [NOT] TRUE, [NOT] FALSE, and [NOT] UNKNOWN.

Syntax:

```
EXPRESSION IS TRUE
EXPRESSION IS NOT TRUE

EXPRESSION IS FALSE
EXPRESSION IS NOT FALSE
```

Usage notes:

This IS TRUE and IS FALSE forms are similar to doing equality comparisons with the Boolean values TRUE and FALSE, except that IS TRUE and IS FALSE always return either TRUE or FALSE, even if the left-hand side expression returns NULL

These operators let you simplify Boolean comparisons that must also check for NULL, for example X != 10 AND X IS NOT NULL is equivalent to (X != 10) IS TRUE.

In Impala 2.1.1 and higher, you can use the operators IS [NOT] TRUE and IS [NOT] FALSE as equivalents for the built-in functions ISTRUE(), ISNOTTRUE(), ISFALSE(), and ISNOTFALSE().

Complex type considerations:

The IS [NOT] TRUE and IS [NOT] FALSE operators are not applicable to complex type columns (STRUCT, ARRAY, or MAP). Using a complex type column with these operators causes a query error.

LIKE operator

A comparison operator for STRING data, with basic wildcard capability using the underscore (_) to match a single character and the percent sign (%) to match any characters. The argument expression must match the entire string value. Typically, it is more efficient to put any % wildcard match at the end of the string.

Syntax:

```
STRING_EXPRESSION LIKE WILDCARD_EXPRESSION
STRING_EXPRESSION NOT LIKE WILDCARD_EXPRESSION
```

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

Examples:

```
select distinct c_last_name from customer where c_last_name like 'Mc%' or c_
last_name like 'Mac%';
select count(c_last_name) from customer where c_last_name like 'M%';
select c_email_address from customer where c_email_address like '%.edu';

-- We can find 4-letter names beginning with 'M' by calling functions...
select distinct c_last_name from customer where length(c_last_name) = 4 and
 substr(c_last_name,1,1) = 'M';
-- ...or in a more readable way by matching M followed by exactly 3 charac
ters.
select distinct c_last_name from customer where c_last_name like 'M___';
```

For case-insensitive comparisons, see the ILIKE operator. For a more general kind of search operator using regular expressions, see the REGEXP operator or its case-insensitive counterpart the IREGEXP operator.

Logical operators

Logical operators return a BOOLEAN value, based on a binary or unary logical operation between arguments that are also BOOLEAN values. Typically, the argument expressions use comparison operators.

Syntax:

```
BOOLEAN_EXPRESSION BINARY_LOGICAL_OPERATOR BOOLEAN_EXPRESSION
UNARY_LOGICAL_OPERATOR BOOLEAN_EXPRESSION
```

The Impala logical operators are:

- AND: A binary operator that returns true if its left-hand and right-hand arguments both evaluate to true, NULL if either argument is NULL, and false otherwise.
- OR: A binary operator that returns true if either of its left-hand and right-hand arguments evaluate to true, NULL if one argument is NULL and the other is either NULL or false, and false otherwise.
- NOT: A unary operator that flips the state of a Boolean expression from true to false, or false to true. If the argument expression is NULL, the result remains NULL. (When NOT is used this way as a unary logical operator, it works differently than the IS NOT NULL comparison operator, which returns true when applied to a NULL.)

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

The following example shows how to do an arithmetic operation using a numeric field of a STRUCT type that is an item within an ARRAY column. Once the scalar numeric value R_NATIONKEY is extracted, it can be used in an arithmetic expression, such as multiplying by 10:

```
-- The SMALLINT is a field within an array of structs.
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

-- When we refer to the scalar value using dot notation,
-- we can use arithmetic and comparison operators on it
-- like any other number.
select r_name, nation.item.n_name, nation.item.n_nationkey
  from region, region.r_nations as nation
where
  nation.item.n_nationkey between 3 and 5
  or nation.item.n_nationkey < 15;
+-------------+----------------+------------------+
| r_name      | item.n_name    | item.n_nationkey |
+-------------+----------------+------------------+
| EUROPE      | UNITED KINGDOM | 23               |
| EUROPE      | RUSSIA         | 22               |
| EUROPE      | ROMANIA        | 19               |
| ASIA        | VIETNAM        | 21               |
| ASIA        | CHINA          | 18               |
| AMERICA     | UNITED STATES  | 24               |
| AMERICA     | PERU           | 17               |
| AMERICA     | CANADA         | 3                |
| MIDDLE EAST | SAUDI ARABIA   | 20               |
| MIDDLE EAST | EGYPT          | 4                |
| AFRICA      | MOZAMBIQUE     | 16               |
| AFRICA      | ETHIOPIA       | 5                |
+-------------+----------------+------------------+
```

REGEXP operator

Tests whether a value matches a regular expression. Uses the POSIX regular expression syntax where ^ and $ match the beginning and end of the string, . represents any single character, * represents a sequence of zero or more items, + represents a sequence of one or more items, ? produces a non-greedy match, and so on.

Syntax:

```
STRING_EXPRESSION REGEXP REGULAR_EXPRESSION
```

Usage notes:

The RLIKE operator is a synonym for REGEXP.

The | symbol is the alternation operator, typically used within () to match different sequences. The () groups do not allow backreferences. To retrieve the part of a value matched within a () section, use the regexp_extract() built-in function.

In Impala 1.3.1 and higher, the REGEXP and RLIKE operators now match a regular expression string that occurs anywhere inside the target string, the same as if the regular expression was enclosed on each side by .*. Previously, these operators only succeeded when the regular expression matched the entire target string. This change improves compatibility with the regular expression support for popular database systems. There is no change to the behavior of the regexp_extract() and regexp_replace() built-in functions.

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator.
You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

Example:

```
- Find all customers whose first name starts with 'J', followed by 0 or more
 of any character.
select c_first_name, c_last_name from customer where c_first_name regexp
'^J.*';
-- Match multiple character sequences, either 'Mac' or 'Mc'.
select c_first_name, c_last_name from customer where c_last_name regexp '^
(Mac|Mc)donald$';
```

RLIKE operator

It is a synonym for the REGEXP operator.

Related Information

Impala mathematical functions

RE2 documentation on GitHub

Impala SQL comments

Impala supports the syntax of standard SQL comments.

- All text from a -- sequence to the end of the line is considered a comment and ignored. This type of comment can occur on a single line by itself, or after all or part of a statement.
- All text from a /* sequence to the next */ sequence is considered a comment and ignored. This type of comment can stretch over multiple lines. This type of comment can occur on one or more lines by itself, in the middle of a statement, or before or after a statement.

Impala SQL statements

The Impala SQL dialect supports a range of standard elements, plus some extensions for Big Data use cases related to data loading and data warehousing.

Note:

In the `impala-shell` interpreter, a semicolon at the end of each statement is required. Since the semicolon is not actually part of the SQL syntax, we do not include it in the syntax definition of each statement, but we do show it in examples intended to be run in `impala-shell`.

The following sections show the major SQL statements that you work with in Impala:

ROLE statements in Impala integrated with Ranger

After upgrading or migrating your workload from CDH to you will have started using Apache Ranger as the authorization provider in Impala. This replaces Apache Sentry. There are some differences in Impala's behavior which you must be aware of when using Ranger as the authorization provider.

Access via Roles in Impala

Impala with Sentry revolved around granting privileges to ROLES, and ROLES to GROUPS. Until this release, Impala's integration with Ranger did not support ROLE related DDL statements. So as a workaround you had to migrate the ROLE-based authorization policies, manage them using Ranger's web UI, to handle them correctly in Impala. You will no longer need to use the Ranger's web UI to manage the ROLEs in 7.1.6 since Impala now supports ROLE management through ROLE related statements.

Note: ROLES can be still managed using the old approach via Ranger's web UI.

Managing ROLES using Ranger's WEB UI

You can perform the following steps to grant to a normal user the privileges on a resource via ROLEs.

1. Assign a user to a group, e.g. using your corporate LDAP provider.
2. In the Ranger web UI, create a ROLE that includes the group consisting of the user we want to grant the privileges.
3. In the Ranger web UI,
   - create a policy for the corresponding resource, e.g., a table,
   - in the 'Select Role' field under the section of 'Allow Conditions', add the ROLE just created above,
   - in the 'Permissions' field under the section of 'Allow Conditions', add the privileges we want to grant to the ROLE associated with the group consisting of that normal user.

ROLE related statements in Impala

To bridge the gap between CDH Impala and Impala in terms of ROLE-related operations, 7.1.6 introduces ROLE related statements in Impala integrated with Ranger.

```
CREATE ROLE <role_name>
DROP ROLE <role_name>
GRANT ROLE <role_name> TO GROUP <group_name>
REVOKE ROLE <role_name> FROM GROUP <group_name>
GRANT <privilege> ON <resource> TO ROLE <role_name>
REVOKE <privilege> ON <resource> FROM ROLE <role_name>
SHOW GRANT ROLE <role_name> ON <resource>
SHOW ROLES
SHOW CURRENT ROLES
SHOW ROLE GRANT GROUP <group_name>
```

Differences in Impala's behavior

The following list describes the major differences in Impala's behavior when using Ranger as the authorization provider in place of Sentry.

- Before dropping a role in Ranger, you must remove all the privileges granted to the role in advance, which was not the case when Sentry was the authorization provider.
- You must specify the resource for the SHOW GRANT ROLE <role_name> ON <resource> statement which is different when using Sentry as the authorization provider. This is due to the fact that there is no API provided by Ranger that allows Impala to directly retrieve the list of all privileges granted to a specified role.

DDL statements

DDL refers to "Data Definition Language", a subset of SQL statements that change the structure of the database schema in some way, typically by creating, deleting, or modifying schema objects such as databases, tables, and views. Most Impala DDL statements start with the keywords CREATE, DROP, or ALTER.

The Impala DDL statements are:

- ALTER TABLE statement on page 108
- ALTER VIEW statement on page 122
- COMPUTE STATS statement on page 124
- CREATE DATABASE statement on page 132
- CREATE FUNCTION statement on page 134
- CREATE TABLE statement on page 141
- CREATE VIEW statement on page 157
- DROP DATABASE statement on page 172
- DROP FUNCTION statement on page 174
- DROP TABLE statement on page 181
- DROP VIEW statement on page 182
- GRANT statement on page 186
- REVOKE statement on page 205

After Impala executes a DDL command, information about available tables, columns, views, partitions, and so on is automatically synchronized between all the Impala nodes in a cluster. (Prior to Impala 1.2, you had to issue a REFR ESH or an INVALIDATE METADATA statement manually on the other nodes to make them aware of the changes.)

If the timing of metadata updates is significant, for example if you use round-robin scheduling where each query could be issued through a different Impala node, you can enable the SYNC_DDL query option to make the DDL statement wait until all nodes have been notified about the metadata changes.

See *Using Impala with the Amazon S3 Filesystem* for details about how Impala DDL statements interact with tables and partitions stored in the Amazon S3 filesystem.

Although INSERT is classified as a DML statement, when the SYNC_DDL option is enabled, INSERT statements also delay their completion until all the underlying data and metadata changes are propagated to all Impala nodes and this option applies to all filesystem-based tables. Internally, Impala inserts have similarities with DDL statements in traditional database systems, because they create metadata needed to track HDFS block locations for new files and they potentially add new partitions to partitioned tables.

Because the SYNC_DDL query option makes each DDL operation take longer than normal, you might only enable it before the last DDL operation in a sequence. For example, if you are running a script that issues multiple of DDL operations to set up an entire new schema, add several new partitions, and so on, you might minimize the performance overhead by enabling the query option only before the last CREATE, DROP, ALTER, or INSERT statement. The script only finishes when all the relevant metadata changes are recognized by all the Impala nodes, so you could connect to any node and issue queries through it.

The classification of DDL, DML, and other statements is not necessarily the same between Impala and Hive. Impala organizes these statements in a way intended to be familiar to people familiar with relational databases or data warehouse products. Statements that modify the metastore database, such as COMPUTE STATS, are classified as DDL. Statements that only query the metastore database, such as SHOW or DESCRIBE, are put into a separate category of utility statements.

Note: The query types shown in the Impala debug web user interface might not match exactly the categories listed here. For example, currently the USE statement is shown as DDL in the debug web UI. The query types shown in the debug web UI are subject to change, for improved consistency.

DML statements

DML refers to “Data Manipulation Language”, a subset of SQL statements that modify the data stored in tables. Because Impala focuses on query performance and leverages the append-only nature of HDFS storage, currently Impala only supports a small set of DML statements.

Impala supports the following DML statements:

- DELETE: Works for Kudu tables only
- INSERT
- LOAD DATA: Does not apply for HBase or Kudu tables
- UPDATE: Works for Kudu tables only
- UPSERT: Works for Kudu tables only

INSERT in Impala is primarily optimized for inserting large volumes of data in a single statement, to make effective use of the multi-megabyte HDFS blocks. This is the way in Impala to create new data files. If you intend to insert one or a few rows at a time, such as using the INSERT ... VALUES syntax, that technique is much more efficient for Impala tables stored in HBase.

LOAD DATA moves existing data files into the directory for an Impala table, making them immediately available for Impala queries. This is one way in Impala to work with data files produced by other Hadoop components. (CREATE EXTERNAL TABLE is the other alternative; with external tables, you can query existing data files, while the files remain in their original location.)

In Impala 2.8 and higher, Impala does support the UPDATE, DELETE, and UPSERT statements for Kudu tables. For HDFS or S3 tables, to simulate the effects of an UPDATE or DELETE statement in other database systems, typically you use INSERT or CREATE TABLE AS SELECT to copy data from one table to another, filtering out or changing the appropriate rows during the copy operation.

You can also achieve a result similar to UPDATE by using Impala tables stored in HBase. When you insert a row into an HBase table, and the table already contains a row with the same value for the key column, the older row is hidden, effectively the same as a single-row UPDATE.

Impala can perform DML operations for tables or partitions stored in the Amazon S3 filesystem.

ALTER DATABASE statement

The ALTER DATABASE statement changes the characteristics of a database.

Use the SET OWNER clause to transfer the ownership of the database from the current owner to another user or a role.

The database owner is originally set to the user who creates the database. When object ownership is enabled in Ranger, an owner of a database can have the ALL with GRANT or ALL without GRANT privilege. The term OWNER is used to differentiate between the ALL privilege that is explicitly granted via the GRANT statement and a privilege that is implicitly granted by the CREATE DATABASE statement.

Syntax:

```
ALTER DATABASE DATABASE_NAME SET OWNER USER user_name;
ALTER DATABASE DATABASE_NAME SET OWNER ROLE role_name;
```

Statement type: DDL

Cancellation: Cannot be cancelled.

Added in: Impala 3.1 release.

ALTER TABLE statement

The ALTER TABLE statement changes the structure or properties of an existing Impala table.

In Impala, this is primarily a logical operation that updates the table metadata in the metastore database that Impala shares with Hive. Most ALTER TABLE operations do not actually rewrite, move, and so on the actual data files. (The RENAME TO clause is the one exception; it can cause HDFS files to be moved to different paths.) When you do an ALTER TABLE operation, you typically need to perform corresponding physical filesystem operations, such as rewriting the data files to include extra fields, or converting them to a different file format.

Syntax:

```
ALTER TABLE [OLD_DB_NAME.]OLD_TABLE_NAME RENAME TO
 [NEW_DB_NAME.]NEW_TABLE_NAME

ALTER TABLE NAME ADD [IF NOT EXISTS] COLUMNS (COL_SPEC[, COL_SPEC ...])
ALTER TABLE NAME REPLACE COLUMNS (COL_SPEC[, COL_SPEC ...])

ALTER TABLE NAME ADD COLUMN [IF NOT EXISTS] COL_SPEC
ALTER TABLE NAME DROP [COLUMN] COLUMN_NAME
ALTER TABLE NAME CHANGE COLUMN_NAME COL_SPEC

ALTER TABLE NAME SET OWNER USER USER_NAME
ALTER TABLE NAME SET OWNER ROLE ROLE_NAME


-- Kudu tables only.
ALTER TABLE NAME ALTER [COLUMN] COLUMN_NAME
  { SET KUDU_STORAGE_ATTR ATTR_VALUE
    | DROP DEFAULT }

kudu_storage_attr ::= { DEFAULT | BLOCK_SIZE | ENCODING | COMPRESSION }

-- Non-Kudu tables only.
ALTER TABLE NAME ALTER [COLUMN] COLUMN_NAME
  SET COMMENT 'COMMENT_TEXT'
```

```
ALTER TABLE NAME ADD [IF NOT EXISTS] PARTITION (PARTITION_SPEC)
  [LOCATION_SPEC]
  [CACHE_SPEC]
ALTER TABLE NAME ADD [IF NOT EXISTS] RANGE PARTITION KUDU_PARTITION_SPEC

ALTER TABLE NAME DROP [IF EXISTS] PARTITION (PARTITION_SPEC)
  [PURGE]
ALTER TABLE NAME DROP [IF EXISTS] RANGE PARTITION KUDU_PARTITION_SPEC

ALTER TABLE NAME RECOVER PARTITIONS

ALTER TABLE NAME [PARTITION (PARTITION_SPEC)]
  SET { FILEFORMAT FILE_FORMAT
  | ROW FORMAT ROW_FORMAT
  | LOCATION 'HDFS_PATH_OF_DIRECTORY'
  | TBLPROPERTIES (TABLE_PROPERTIES)
  | SERDEPROPERTIES (SERDE_PROPERTIES) }

ALTER TABLE NAME COLNAME
  ('STATSKEY'='VAL, ...)

statsKey ::= numDVs | numNulls | avgSize | maxSize

ALTER TABLE NAME [PARTITION (PARTITION_SPEC)] SET { CACHED IN
 'POOL_NAME' [WITH REPLICATION = INTEGER] | UNCACHED }

NEW_NAME ::= [NEW_DATABASE.]NEW_TABLE_NAME

COL_SPEC ::= COL_NAME TYPE_NAME  COMMENT 'COLUMN-COMMENT' [KUDU_ATTRIBUTES]

KUDU_ATTRIBUTES ::= { [NOT] NULL | ENCODING CODEC | COMPRESSION ALGORITHM |
  DEFAULT CONSTANT | BLOCK_SIZE NUMBER }

PARTITION_SPEC ::= SIMPLE_PARTITION_SPEC | COMPLEX_PARTITION_SPEC

SIMPLE_PARTITION_SPEC ::= PARTITION_COL=CONSTANT_VALUE

COMPLEX_PARTITION_SPEC ::= COMPARISON_EXPRESSION_ON_PARTITION_COL

KUDU_PARTITION_SPEC ::= CONSTANT RANGE_OPERATOR
 VALUES RANGE_OPERATOR CONSTANT | VALUE = CONSTANT

cache_spec ::= CACHED IN 'POOL_NAME' [WITH REPLICATION = INTEGER] | UNCACHED

location_spec ::= LOCATION 'HDFS_PATH_OF_DIRECTORY'

TABLE_PROPERTIES ::= 'NAME'='VALUE'[, 'NAME'='VALUE' ...]

SERDE_PROPERTIES ::= 'NAME'='VALUE'[, 'NAME'='VALUE' ...]

FILE_FORMAT ::= { PARQUET | TEXTFILE | RCFILE | SEQUENCEFILE | AVRO }

ROW_FORMAT ::= DELIMITED
  [FIELDS TERMINATED BY 'CHAR' [ESCAPED BY 'CHAR']]
  [LINES TERMINATED BY 'CHAR']
```

Statement type: DDL

Complex type considerations:

In Impala 2.3 and higher, the ALTER TABLE statement can change the metadata for tables containing complex types (ARRAY, STRUCT, and MAP). For example, you can use an ADD COLUMNS, DROP COLUMN, or CHANGE clause to modify the table layout for complex type columns. Although Impala queries only work for complex type

columns in Parquet tables, the complex type support in the ALTER TABLE statement applies to all file formats. For example, you can use Impala to update metadata for a staging table in a non-Parquet file format where the data is populated by Hive. Or you can use ALTER TABLE SET FILEFORMAT to change the format of an existing table to Parquet so that Impala can query it. Remember that changing the file format for a table does not convert the data files within the table; you must prepare any Parquet data files containing complex types outside Impala, and bring them into the table using LOAD DATA or updating the table's LOCATION property.

Usage notes:

Whenever you specify partitions in an ALTER TABLE statement, through the PARTITION (*PARTITION_SPEC*) clause, you must include all the partitioning columns in the specification.

Most of the ALTER TABLE operations work the same for internal tables (managed by Impala) as for external tables (with data files located in arbitrary locations). The exception is renaming a table; for an external table, the underlying data directory is not renamed or moved.

To drop or alter multiple partitions:

In Impala 2.8 and higher, the expression for the partition clause with a DROP or SET operation can include comparison operators such as <, IN, or BETWEEN, and Boolean operators such as AND and OR.

For example, you might drop a group of partitions corresponding to a particular date range after the data “ages out”:

```
alter table historical_data drop partition (year < 1995);
alter table historical_data drop partition (year = 1996 and month between 1
 and 6);
```

For tables with multiple partition keys columns, you can specify multiple conditions separated by commas, and the operation only applies to the partitions that match all the conditions (similar to using an AND clause):

```
alter table historical_data drop partition (year < 1995, last_name like 'A%
');
```

This technique can also be used to change the file format of groups of partitions, as part of an ETL pipeline that periodically consolidates and rewrites the underlying data files in a different file format:

```
alter table fast_growing_data partition (year = 2016, month in (10,11,12))
set fileformat parquet;
```

Note:

The extended syntax involving comparison operators and multiple partitions applies to the SET FILEFORMAT, SET TBLPROPERTIES, SET SERDEPROPERTIES, and SET [UN]CACHED clauses. You can also use this syntax with the PARTITION clause in the COMPUTE INCREMENTAL STATS statement, and with the PARTITION clause of the SHOW FILES statement. Some forms of ALTER TABLE still only apply to one partition at a time: the SET LOCATION and ADD PARTITION clauses. The PARTITION clauses in the LOAD DATA and INSERT statements also only apply to one partition at a time.

A DDL statement that applies to multiple partitions is considered successful (resulting in no changes) even if no partitions match the conditions. The results are the same as if the IF EXISTS clause was specified.

The performance and scalability of this technique is similar to issuing a sequence of single-partition ALTER TABLE statements in quick succession. To minimize bottlenecks due to communication with the metastore database, or causing other DDL operations on the same table to wait, test the effects of performing ALTER TABLE statements that affect large numbers of partitions.

Amazon S3 considerations:

You can specify an s3a:// prefix on the LOCATION attribute of a table or partition to make Impala query data from the Amazon S3 filesystem. In Impala 2.6 and higher, Impala automatically handles creating or removing the associated folders when you issue ALTER TABLE statements with the ADD PARTITION or DROP PARTITION clauses.

In Impala 2.6 and higher, Impala DDL statements such as CREATE DATABASE, CREATE TABLE, DROP DATABASE CASCADE, DROP TABLE, and ALTER TABLE [ADD|DROP] PARTITION can create or remove folders as needed in the Amazon S3 system. Prior to Impala 2.6, you had to create folders yourself and point Impala database, tables, or partitions at them, and manually remove folders when no longer needed. See Related Information for details about reading and writing S3 data with Impala.

HDFS caching (CACHED IN clause):

If you specify the CACHED IN clause, any existing or future data files in the table directory or the partition subdirectories are designated to be loaded into memory with the HDFS caching mechanism.

In Impala 2.2 and higher, the optional WITH REPLICATION clause for CREATE TABLE and ALTER TABLE lets you specify a *replication factor*, the number of hosts on which to cache the same data blocks. When Impala processes a cached data block, where the cache replication factor is greater than 1, Impala randomly selects a host that has a cached copy of that data block. This optimization avoids excessive CPU usage on a single host when the same cached data block is processed multiple times. Cloudera recommends specifying a value greater than or equal to the HDFS block replication factor.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

The following sections show examples of the use cases for various ALTER TABLE clauses.

To rename a table (RENAME TO clause):

The RENAME TO clause lets you change the name of an existing table, and optionally which database it is located in.

For internal tables, this operation physically renames the directory within HDFS that contains the data files; the original directory name no longer exists. By qualifying the table names with database names, you can use this technique to move an internal table (and its associated data directory) from one database to another. For example:

```
create database d1;
create database d2;
create database d3;
use d1;
create table mobile (x int);
use d2;
-- Move table from another database to the current one.
alter table d1.mobile rename to mobile;
use d1;
-- Move table from one database to another.
alter table d2.mobile rename to d3.mobile;
```

To change the owner of a table:

```
ALTER TABLE NAME SET OWNER USER USER_NAME;
ALTER TABLE NAME SET OWNER ROLE ROLE_NAME;
```

The table owner is originally set to the user who creates the table. When object ownership is enabled in Ranger, an owner of a table can have the ALL with GRANT or ALL without GRANT privilege. The term OWNER is used to differentiate between the ALL privilege that is explicitly granted via the GRANT statement and a privilege that is implicitly granted by the CREATE TABLE statement.

Use the ALTER TABLE SET OWNER to transfer the ownership from the current owner to another user or a role.

To change the physical location where Impala looks for data files associated with a table or partition:

```
ALTER TABLE TABLE_NAME [PARTITION (PARTITION_SPEC)] SET LOCATION
 'HDFS_PATH_OF_DIRECTORY';
```

The path you specify is the full HDFS path where the data files reside, or will be created. Impala does not create any additional subdirectory named after the table. Impala does not move any data files to this new location or change any data files that might already exist in that directory.

To set the location for a single partition, include the PARTITION clause. Specify all the same partitioning columns for the table, with a constant value for each, to precisely identify the single partition affected by the statement:

```
create table p1 (s string) partitioned by (month int, day int);
-- Each ADD PARTITION clause creates a subdirectory in HDFS.
alter table p1 add partition (month=1, day=1);
alter table p1 add partition (month=1, day=2);
alter table p1 add partition (month=2, day=1);
alter table p1 add partition (month=2, day=2);
-- Redirect queries, INSERT, and LOAD DATA for one partition
-- to a specific different directory.
alter table p1 partition (month=1, day=1) set location '/usr/external_data/n
ew_years_day';
```

Note: If you are creating a partition for the first time and specifying its location, for maximum efficiency, use a single ALTER TABLE statement including both the ADD PARTITION and LOCATION clauses, rather than separate statements with ADD PARTITION and SET LOCATION clauses.

To automatically detect new partition directories added through Hive or HDFS operations:

In Impala 2.3 and higher, the RECOVER PARTITIONS clause scans a partitioned table to detect if any new partition directories were added outside of Impala, such as by Hive ALTER TABLE statements or by `hdfs dfs` or `hadoop fs` commands. The RECOVER PARTITIONS clause automatically recognizes any data files present in these new directories, the same as the REFRESH statement does.

For example, here is a sequence of examples showing how you might create a partitioned table in Impala, create new partitions through Hive, copy data files into the new partitions with the `hdfs` command, and have Impala recognize the new partitions and new data:

In Impala, create the table, and a single partition for demonstration purposes:

```
create database recover_partitions;
use recover_partitions;
create table t1 (s string) partitioned by (yy int, mm int);
insert into t1 partition (yy = 2016, mm = 1) values ('Partition exists');
show files in t1;
+-----------------------------------------------------------------------+---
---+--------------+
| Path                                                                  | Size
 | Partition    |
+-----------------------------------------------------------------------+---
---+--------------+
| /user/hive/warehouse/recover_partitions.db/t1/yy=2016/mm=1/data.txt | 17B
 | yy=2016/mm=1 |
+-----------------------------------------------------------------------+---
---+--------------+
quit;
```

In Hive, create some new partitions. In a real use case, you might create the partitions and populate them with data as the final stages of an ETL pipeline.

```
hive> use recover_partitions;
OK
hive> alter table t1 add partition (yy = 2016, mm = 2);
OK
hive> alter table t1 add partition (yy = 2016, mm = 3);
OK
hive> quit;
```

For demonstration purposes, manually copy data (a single row) into these new partitions, using manual HDFS operations:

```
$ hdfs dfs -ls /user/hive/warehouse/recover_partitions.db/t1/yy=2016/
Found 3 items
drwxr-xr-x - impala   hive 0 2016-05-09 16:06 /user/hive/warehouse/recover
_partitions.db/t1/yy=2016/mm=1
drwxr-xr-x - jrussell hive 0 2016-05-09 16:14 /user/hive/warehouse/recove
r_partitions.db/t1/yy=2016/mm=2
drwxr-xr-x - jrussell hive 0 2016-05-09 16:13 /user/hive/warehouse/recover_p
artitions.db/t1/yy=2016/mm=3

$ hdfs dfs -cp /user/hive/warehouse/recover_partitions.db/t1/yy=2016/mm=1/
data.txt \
  /user/hive/warehouse/recover_partitions.db/t1/yy=2016/mm=2/data.txt
$ hdfs dfs -cp /user/hive/warehouse/recover_partitions.db/t1/yy=2016/mm=1/d
ata.txt \
  /user/hive/warehouse/recover_partitions.db/t1/yy=2016/mm=3/data.txt
```

```
hive> select * from t1;
OK
Partition exists  2016  1
Partition exists  2016  2
Partition exists  2016  3
hive> quit;
```

In Impala, initially the partitions and data are not visible. Running ALTER TABLE with the RECOVER PARTITIONS clause scans the table data directory to find any new partition directories, and the data files inside them:

```
select * from t1;
+------------------+------+----+
| s                | yy   | mm |
+------------------+------+----+
| Partition exists | 2016 | 1  |
+------------------+------+----+

alter table t1 recover partitions;
select * from t1;
+------------------+------+----+
```

```
| s                | yy   | mm |
+------------------+------+----+
| Partition exists | 2016 | 1  |
| Partition exists | 2016 | 3  |
| Partition exists | 2016 | 2  |
+------------------+------+----+
```

To change the key-value pairs of the TBLPROPERTIES and SERDEPROPERTIES fields:

```
ALTER TABLE TABLE_NAME SET TBLPROPERTIES ('KEY1'='VALUE1',
 'KEY2'='VALUE2'[, ...]);
ALTER TABLE TABLE_NAME SET SERDEPROPERTIES ('KEY1'='VALUE1',
 'KEY2'='VALUE2'[, ...]);
```

The TBLPROPERTIES clause is primarily a way to associate arbitrary user-specified data items with a particular table.

The SERDEPROPERTIES clause sets up metadata defining how tables are read or written, needed in some cases by Hive but not used extensively by Impala. You would use this clause primarily to change the delimiter in an existing text table or partition, by setting the 'serialization.format' and 'field.delim' property values to the new delimiter character. The SERDEPROPERTIES clause does not change the existing data in the table. The change only affects the future inserts into the table.

Use the DESCRIBE FORMATTED statement to see the current values of these properties for an existing table.

To manually set or update table or column statistics:

Although for most tables the COMPUTE STATS or COMPUTE INCREMENTAL STATS statement is all you need to keep table and column statistics up to date for a table, sometimes for a very large table or one that is updated frequently, the length of time to recompute all the statistics might make it impractical to run those statements as often as needed. As a workaround, you can use the ALTER TABLE statement to set table statistics at the level of the entire table or a single partition, or column statistics at the level of the entire table.

You can set the numrows value for table statistics by changing the TBLPROPERTIES setting for a table or partition. For example:

```
create table analysis_data stored as parquet as select * from raw_data;
Inserted 1000000000 rows in 181.98s
compute stats analysis_data;
insert into analysis_data select * from smaller_table_we_forgot_before;
Inserted 1000000 rows in 15.32s
-- Now there are 1001000000 rows. We can update this single data point in t
he stats.
alter table analysis_data set tblproperties('numRows'='1001000000', 'STATS
_GENERATED_VIA_STATS_TASK'='true');
```

```
-- If the table originally contained 1 million rows, and we add another part
ition with 30 thousand rows,
-- change the numRows property for the partition and the overall table.
alter table partitioned_data partition(year=2009, month=4) set tblproperties
 ('numRows'='30000', 'STATS_GENERATED_VIA_STATS_TASK'='true');
alter table partitioned_data set tblproperties ('numRows'='1030000', 'STATS_
GENERATED_VIA_STATS_TASK'='true');
```

In Impala 2.6 and higher, you can use the SET COLUMN STATS clause to set a specific stats value for a particular column.

You specify a case-insensitive symbolic name for the kind of statistics: numDVs, numNulls, avgSize, maxSize. The key names and values are both quoted. This operation applies to an entire table, not a specific partition. For example:

```
create table t1 (x int, s string);
insert into t1 values (1, 'one'), (2, 'two'), (2, 'deux');
show column stats t1;
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| x      | INT    | -1               | -1     | 4        | 4        |
| s      | STRING | -1               | -1     | -1       | -1       |
+--------+--------+------------------+--------+----------+----------+
alter table t1 set column stats x ('numDVs'='2','numNulls'='0');
alter table t1 set column stats s ('numdvs'='3','maxsize'='4');
show column stats t1;
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| x      | INT    | 2                | 0      | 4        | 4        |
| s      | STRING | 3                | -1     | 4        | -1       |
+--------+--------+------------------+--------+----------+----------+
```

To reorganize columns for a table:

You can add multiple columns at a time using the ALTER TABLE statement. If you specify the IF NOT EXISTS clause, Impala silently ignores the ADD request and does not return an error if a column with the same name exists in the table.

When you replace columns, all the original column definitions are discarded.

You might use these statements if you receive a new set of data files with different data types or columns in a different order. The data files are retained, so if the new columns are incompatible with the old ones, use INSERT OVERWRITE or LOAD DATA OVERWRITE to replace all the data before issuing any further queries.

For example, here is how you might add columns to an existing table. The first ALTER TABLE adds two new columns, and the second ALTER TABLE adds one new column. A single Impala query reads both the old and new data files, containing different numbers of columns. For any columns not present in a particular data file, all the column values are considered to be NULL.

```
create table t1 (x int);
insert into t1 values (1), (2);

alter table t1 add columns (s string, t timestamp);
insert into t1 values (3, 'three', now());

alter table t1 add columns (b boolean);
insert into t1 values (4, 'four', now(), true);

select * from t1 order by x;
+---+-------+-------------------------------+------+
| x | s     | t                             | b    |
+---+-------+-------------------------------+------+
| 1 | NULL  | NULL                          | NULL |
| 2 | NULL  | NULL                          | NULL |
| 3 | three | 2016-05-11 11:19:45.054457000 | NULL |
| 4 | four  | 2016-05-11 11:20:20.260733000 | true |
+---+-------+-------------------------------+------+
```

You might use the CHANGE clause to rename a single column, or to treat an existing column as a different type than before, such as to switch between treating a column as STRING and TIMESTAMP, or between INT and BIGINT. You can only drop a single column at a time; to drop multiple columns, issue multiple ALTER TABLE statements, or define the new set of columns with a single ALTER TABLE ... REPLACE COLUMNS statement.

The following examples show some safe operations to drop or change columns. Dropping the final column in a table lets Impala ignore the data causing any disruption to existing data files. Changing the type of a column works if

existing data values can be safely converted to the new type. The type conversion rules depend on the file format of the underlying table. For example, in a text table, the same value can be interpreted as a STRING or a numeric value, while in a binary format such as Parquet, the rules are stricter and type conversions only work between certain sizes of integers.

```
create table optional_columns (x int, y int, z int, a1 int, a2 int);
insert into optional_columns values (1,2,3,0,0), (2,3,4,100,100);
-- When the last column in the table is dropped, Impala ignores the
-- values that are no longer needed. (Dropping A1 but leaving A2
-- would cause problems, as we will see in a subsequent example.)
alter table optional_columns drop column a2;
alter table optional_columns drop column a1;

select * from optional_columns;
+---+---+---+
| x | y | z |
+---+---+---+
| 1 | 2 | 3 |
| 2 | 3 | 4 |
+---+---+---+
```

```
create table int_to_string (s string, x int);
insert into int_to_string values ('one', 1), ('two', 2);
-- What was an INT column will now be interpreted as STRING.
-- This technique works for text tables but not other file formats.
-- The second X represents the new name of the column, which we keep the s
ame.
alter table int_to_string change x x string;

-- Once the type is changed, we can insert non-integer values into the X
column
-- and treat that column as a string, for example by uppercasing or conca
tenating.
insert into int_to_string values ('three', 'trois');
select s, upper(x) from int_to_string;
+-------+----------+
| s     | upper(x) |
+-------+----------+
| one   | 1        |
| two   | 2        |
| three | TROIS    |
+-------+----------+
```

Remember that Impala does not actually do any conversion for the underlying data files as a result of ALTER TA BLE statements. If you use ALTER TABLE to create a table layout that does not agree with the contents of the underlying files, you must replace the files yourself, such as using LOAD DATA to load a new set of data files, or INSERT OVERWRITE to copy from another table and replace the original data.

The following example shows what happens if you delete the middle column from a Parquet table containing three columns. The underlying data files still contain three columns of data. Because the columns are interpreted based on their positions in the data file instead of the specific column names, a SELECT * query now reads the first and second columns from the data file, potentially leading to unexpected results or conversion errors. For this reason, if you expect to someday drop a column, declare it as the last column in the table, where its data can be ignored by queries after the column is dropped. Or, re-run your ETL process and create new data files if you drop or change the type of a column in a way that causes problems with existing data files.

```
-- Parquet table showing how dropping a column can produce unexpected resul
ts.
```

```
create table p1 (s1 string, s2 string, s3 string) stored as parquet;

insert into p1 values ('one', 'un', 'uno'), ('two', 'deux', 'dos'),
  ('three', 'trois', 'tres');
select * from p1;
+-------+-------+------+
| s1    | s2    | s3   |
+-------+-------+------+
| one   | un    | uno  |
| two   | deux  | dos  |
| three | trois | tres |
+-------+-------+------+

alter table p1 drop column s2;
-- The S3 column contains unexpected results.
-- Because S2 and S3 have compatible types, the query reads
-- values from the dropped S2, because the existing data files
-- still contain those values as the second column.
select * from p1;
+-------+-------+
| s1    | s3    |
+-------+-------+
| one   | un    |
| two   | deux  |
| three | trois |
+-------+-------+
```

```
-- Parquet table showing how dropping a column can produce conversion error
s.
create table p2 (s1 string, x int, s3 string) stored as parquet;
insert into p2 values ('one', 1, 'uno'), ('two', 2, 'dos'), ('three', 3, 't
res');
select * from p2;
+-------+---+------+
| s1    | x | s3   |
+-------+---+------+
| one   | 1 | uno  |
| two   | 2 | dos  |
| three | 3 | tres |
+-------+---+------+
alter table p2 drop column x;
select * from p2;
WARNINGS:
File 'HDFS_FILENAME' has an incompatible Parquet schema for column 'add_col
umns.p2.s3'.
Column type: STRING, Parquet schema:
optional int32 x [i:1 d:1 r:0]

File 'HDFS_FILENAME' has an incompatible Parquet schema for column 'add_col
umns.p2.s3'.
Column type: STRING, Parquet schema:
optional int32 x [i:1 d:1 r:0]
```

In Impala 2.6 and higher, if an Avro table is created without column definitions in the CREATE TABLE statement, and columns are later added through ALTER TABLE, the resulting table is now queryable. Missing values from the newly added columns now default to NULL.

To change the file format that Impala expects data to be in, for a table or partition:

Use an ALTER TABLE ... SET FILEFORMAT clause. You can include an optional PARTITION (*COL1=VAL1*, *COL2=VAL2*, ... clause so that the file format is changed for a specific partition rather than the entire table.

Because this operation only changes the table metadata, you must do any conversion of existing data using regular Hadoop techniques outside of Impala. Any new data created by the Impala INSERT statement will be in the new format. You cannot specify the delimiter for Text files; the data files must be comma-delimited.

To set the file format for a single partition, include the PARTITION clause. Specify all the same partitioning columns for the table, with a constant value for each, to precisely identify the single partition affected by the statement:

```
create table p1 (s string) partitioned by (month int, day int);
-- Each ADD PARTITION clause creates a subdirectory in HDFS.
alter table p1 add partition (month=1, day=1);
alter table p1 add partition (month=1, day=2);
alter table p1 add partition (month=2, day=1);
alter table p1 add partition (month=2, day=2);
-- Queries and INSERT statements will read and write files
-- in this format for this specific partition.
alter table p1 partition (month=2, day=2) set fileformat parquet;
```

To change the row format with different delimiter characters:

Use the SET ROW FORMAT DELIMITED clause to ingest data files that use a different delimiter character or a different line end character. When specifying delimiter and line end characters with the FIELDS TERMINATED BY, ESCAPED BY, and LINES TERMINATED BY clauses, you can use the following:

- A regular ASCII character surrounded by single or double quotation marks.
- An octal sequence, such as '\054' representing a comma or '\0' for ASCII null (hex 00).
- Special characters, such as:
  - '\t' for tab
  - '\n' for newline or linefeed
  - '\r' for carriage return
- An integer in the range '-127'..'128' (with quotation marks but no backslash)

  Negative values are subtracted from 256. For example, FIELDS TERMINATED BY '-2' sets the field delimiter to ASCII code 254.

For the ESCAPED BY clause, choose an escape character that is not used anywhere else in the file. The character following the escape character is taken literally as part of a field value.

Surrounding field values with quotation marks does not help Impala to parse fields with embedded delimiter characters as the quotation marks are considered to be part of the column value.

If you want to use \ as the escape character, specify the clause in `impala-shell` as ESCAPED BY '\\'.

To add or drop partitions for a table, the table must already be partitioned (that is, created with a PARTITIONED BY clause). The partition is a physical directory in HDFS, with a name that encodes a particular column value (the partition key). The Impala INSERT statement already creates the partition if necessary, so the ALTER TABLE ... ADD PARTITION is primarily useful for importing data by moving or copying existing data files into the HDFS directory corresponding to a partition. (You can use the LOAD DATA statement to move files into the partition directory, or ALTER TABLE ... PARTITION (...) SET LOCATION to point a partition at a directory that already contains data files.

The DROP PARTITION clause is used to remove the HDFS directory and associated data files for a particular set of partition key values; for example, if you always analyze the last 3 months worth of data, at the beginning of each month you might drop the oldest partition that is no longer needed. Removing partitions reduces the amount of metadata associated with the table and the complexity of calculating the optimal query plan, which can simplify and speed up queries on partitioned tables, particularly join queries. Here is an example showing the ADD PARTITION and DROP PARTITION clauses.

To avoid errors while adding or dropping partitions whose existence is not certain, add the optional IF [NOT] EXISTS clause between the ADD or DROP keyword and the PARTITION keyword. That is, the entire clause becomes ADD IF NOT EXISTS PARTITION or DROP IF EXISTS PARTITION. The following example shows how partitions can be created automatically through INSERT statements, or manually through ALTER TABLE statements.

The IF [NOT] EXISTS clauses let the ALTER TABLE statements succeed even if a new requested partition already exists, or a partition to be dropped does not exist.

Inserting 2 year values creates 2 partitions:

```
create table partition_t (s string) partitioned by (y int);
insert into partition_t (s,y) values ('two thousand',2000), ('nineteen nine
ty',1990);
show partitions partition_t;
+-------+-------+--------+------+--------------+-------------------+--------
+-------------------+
| y     | #Rows | #Files | Size | Bytes Cached | Cache Replication | Format
 | Incremental stats |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
| 1990  | -1    | 1      | 16B  | NOT CACHED   | NOT CACHED        | TEXT
 | false |
| 2000  | -1    | 1      | 13B  | NOT CACHED   | NOT CACHED        | TEXT
| false |
| Total | -1    | 2      | 29B  | 0B           |                   |
 |       |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
```

Without the IF NOT EXISTS clause, an attempt to add a new partition might fail:

```
alter table partition_t add partition (y=2000);
ERROR: AnalysisException: Partition spec already exists: (y=2000).
```

The IF NOT EXISTS clause makes the statement succeed whether or not there was already a partition with the specified key value:

```
alter table partition_t add if not exists partition (y=2000);
alter table partition_t add if not exists partition (y=2010);
show partitions partition_t;
+-------+-------+--------+------+--------------+-------------------+--------
+-------------------+
| y     | #Rows | #Files | Size | Bytes Cached | Cache Replication | Format
| Incremental stats |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
| 1990  | -1    | 1      | 16B  | NOT CACHED   | NOT CACHED        | TEXT
| false |
| 2000  | -1    | 1      | 13B  | NOT CACHED   | NOT CACHED        | TEXT
 | false |
| 2010  | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT
| false |
| Total | -1    | 2      | 29B  | 0B           |                   |
 |       |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
```

Likewise, the IF EXISTS clause lets DROP PARTITION succeed whether or not the partition is already in the table:

```
alter table partition_t drop if exists partition (y=2000);
alter table partition_t drop if exists partition (y=1950);
show partitions partition_t;
```

```
+-------+-------+--------+------+--------------+-------------------+--------
+-------------------+
| y     | #Rows | #Files | Size | Bytes Cached | Cache Replication | Format
 | Incremental stats |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
| 1990  | -1    | 1      | 16B  | NOT CACHED   | NOT CACHED        | TEXT
 | false |
| 2010  | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT
| false |
| Total | -1    | 1      | 16B  | 0B           |                   |
 |       |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
```

The optional PURGE keyword, available in Impala 2.3 and higher, is used with the DROP PARTITION clause to remove associated HDFS data files immediately rather than going through the HDFS trashcan mechanism. Use this keyword when dropping a partition if it is crucial to remove the data as quickly as possible to free up space, or if there is a problem with the trashcan, such as the trash cannot being configured or being in a different HDFS encryption zone than the data files.

```
-- Create an empty table and define the partitioning scheme.
create table part_t (x int) partitioned by (month int);
-- Create an empty partition into which you could copy data files from some
 other source.
alter table part_t add partition (month=1);
-- After changing the underlying data, issue a REFRESH statement to make
the data visible in Impala.
refresh part_t;
-- Later, do the same for the next month.
alter table part_t add partition (month=2);

-- Now you no longer need the older data.
alter table part_t drop partition (month=1);
-- If the table was partitioned by month and year, you would issue a state
ment like:
-- alter table part_t drop partition (year=2003,month=1);
-- which would require 12 ALTER TABLE statements to remove a year's worth of
 data.
-- If the data files for subsequent months were in a different file format,
-- you could set a different file format for the new partition as you create
 it.
alter table part_t add partition (month=3) set fileformat=parquet;
```

The value specified for a partition key can be an arbitrary constant expression, without any references to columns. For example:

```
alter table time_data add partition (month=concat('Decem','ber'));
alter table sales_data add partition (zipcode = cast(9021 * 10 as string));
```

Note:

An alternative way to reorganize a table and its associated data files is to use CREATE TABLE to create a variation of the original table, then use INSERT to copy the transformed or reordered data to the new table. The advantage of ALTER TABLE is that it avoids making a duplicate copy of the data files, allowing you to reorganize huge volumes of data in a space-efficient way using familiar Hadoop techniques.

To switch a table between internal and external:

You can switch a table from internal to external, or from external to internal, by using the ALTER TABLE statement:

```
-- Switch a table from internal to external.
ALTER TABLE TABLE_NAME SET TBLPROPERTIES('EXTERNAL'='TRUE');

-- Switch a table from external to internal.
ALTER TABLE TABLE_NAME SET TBLPROPERTIES('EXTERNAL'='FALSE');
```

If the Kudu service is integrated with the Hive Metastore, the above operations are not supported.

Cancellation: Cannot be cancelled.

HDFS permissions:

Most ALTER TABLE clauses do not actually read or write any HDFS files, and so do not depend on specific HDFS permissions. For example, the SET FILEFORMAT clause does not actually check the file format existing data files or convert them to the new format, and the SET LOCATION clause does not require any special permissions on the new location. (Any permission-related failures would come later, when you actually query or insert into the table.)

In general, ALTER TABLE clauses that do touch HDFS files and directories require the same HDFS permissions as corresponding CREATE, INSERT, or SELECT statements. The permissions allow the user ID that the `impalad` daemon runs under, typically the impala user, to read or write files or directories, or (in the case of the execute bit) descend into a directory. The RENAME TO clause requires read, write, and execute permission in the source and destination database directories and in the table data directory, and read and write permission for the data files within the table. The ADD PARTITION and DROP PARTITION clauses require write and execute permissions for the associated partition directory.

Kudu considerations:

Because of the extra constraints and features of Kudu tables, such as the NOT NULL and DEFAULT attributes for columns, ALTER TABLE has specific requirements related to Kudu tables:

- In an ADD COLUMNS operation, you can specify the NULL, NOT NULL, and DEFAULT *DEFAULT_VALUE* column attributes.
- In Impala 2.9 and higher, you can also specify the ENCODING, COMPRESSION, and BLOCK_SIZE attributes when adding a column.
- If you add a column with a NOT NULL attribute, it must also have a DEFAULT attribute, so the default value can be assigned to that column for all existing rows.
- The DROP COLUMN clause works the same for a Kudu table as for other kinds of tables.
- Although you can change the name of a column with the CHANGE clause, you cannot change the type of a column in a Kudu table.
- You cannot change the nullability of existing columns in a Kudu table.
- In Impala 2.10, you can change the default value, encoding, compression, or block size of existing columns in a Kudu table by using the SET clause.
- You cannot use the REPLACE COLUMNS clause with a Kudu table.
- The RENAME TO clause for a Kudu table only affects the name stored in the metastore database that Impala uses to refer to the table. To change which underlying Kudu table is associated with an Impala table name, you must change the TBLPROPERTIES property of the table: SET TBLPROPERTIES('kudu.table_name '='*KUDU_TBL_NAME*'). You can only change underlying Kudu tables for the external tables.

The following are some examples of using the ADD COLUMNS clause for a Kudu table:

```
CREATE TABLE t1 ( x INT, PRIMARY KEY (x) )
  PARTITION BY HASH (x) PARTITIONS 16
  STORED AS KUDU

ALTER TABLE t1 ADD COLUMNS (y STRING ENCODING prefix_encoding);
ALTER TABLE t1 ADD COLUMNS (z INT DEFAULT 10);
ALTER TABLE t1 ADD COLUMNS (a STRING NOT NULL DEFAULT '', t TIMESTAMP COM
PRESSION default_compression);
```

The following are some examples of modifying column defaults and storage attributes for a Kudu table:

```
create table kt (x bigint primary key, s string default 'yes', t timestamp)
  stored as kudu;

-- You can change the default value for a column, which affects any rows
-- inserted after this change is made.
alter table kt alter column s set default 'no';
-- You can remove the default value for a column, which affects any rows
-- inserted after this change is made. If the column is nullable, any
-- future inserts default to NULL for this column. If the column is marked
-- NOT NULL, any future inserts must specify a value for the column.
alter table kt alter column s drop default;
insert into kt values (1, 'foo', now());
-- Because of the DROP DEFAULT above, omitting S from the insert
-- gives it a value of NULL.
insert into kt (x, t) values (2, now());

select * from kt;
+---+------+-------------------------------+
| x | s    | t                             |
+---+------+-------------------------------+
| 2 | NULL | 2017-10-02 00:03:40.652156000 |
| 1 | foo  | 2017-10-02 00:03:04.346185000 |
+---+------+-------------------------------+

-- Other storage-related attributes can also be changed for columns.
-- These changes take effect for any newly inserted rows, or rows
-- rearranged due to compaction after deletes or updates.
alter table kt alter column s set encoding prefix_encoding;
-- The COLUMN keyword is optional in the syntax.
alter table kt alter x set block_size 2048;
alter table kt alter column t set compression zlib;

desc kt;
+------+-----------+---------+-------------+----------+---------------+--
---------------+---------------------+------------+
| name | type      | comment | primary_key | nullable | default_value | enco
ding        | compression         | block_size |
+------+-----------+---------+-------------+----------+---------------+----
-------------+---------------------+------------+
| x    | bigint    |         | true        | false    |               | AU
TO_ENCODING   | DEFAULT_COMPRESSION | 2048       |
| s    | string    |         | false       | true     |               | P
REFIX_ENCODING | DEFAULT_COMPRESSION | 0          |
| t    | timestamp |         | false       | true     |               | AUTO
_ENCODING   | ZLIB                | 0          |
+------+-----------+---------+-------------+----------+---------------+----
-------------+---------------------+------------+
```

Kudu tables all use an underlying partitioning mechanism. The partition syntax is different than for non-Kudu tables. You can use the ALTER TABLE statement to add and drop *range partitions* from a Kudu table. Any new range must not overlap with any existing ranges. Dropping a range removes all the associated rows from the table.

Related Information

Impala with Amazon S3

ALTER VIEW statement

The ALTER VIEW statement changes the characteristics of a view.

Because a view is a logical construct, an alias for a query, with no physical data behind it, ALTER VIEW only involves changes to metadata in the metastore database, not any data files in HDFS.

To see the definition of the updated view, issue a DESCRIBE FORMATTED statement.

Syntax:

```
ALTER VIEW [DATABASE_NAME.]VIEW_NAME
   [(COLUMN_NAME [COMMENT 'COLUMN_COMMENT'][, ...])]
   AS SELECT_STATEMENT;

ALTER VIEW [DATABASE_NAME.]VIEW_NAME
   RENAME TO [DATABASE_NAME.]VIEW_NAME;

ALTER VIEW [DATABASE_NAME.]VIEW_NAME SET OWNER USER user_name;
ALTER VIEW [DATABASE_NAME.]VIEW_NAME SET OWNER ROLE role_name;

ALTER VIEW [DATABASE_NAME.]VIEW_NAME
    SET TBLPROPERTIES ('NAME' = 'VALUE'[, 'NAME' = 'VALUE' ...]);

ALTER VIEW [DATABASE_NAME.]VIEW_NAME
    UNSET TBLPROPERTIES ('NAME'[, ...]);
```

- The AS clause associates the view with a different query.

  An optional list of column names can be specified with or without the column-level comments.

  For example:

  ```
  ALTER VIEW v1 AS SELECT x, UPPER(s) s FROM t2;
  ALTER VIEW v1 (c1, c2) AS SELECT x, UPPER(s) s FROM t2;
  ALTER VIEW v7 (c1 COMMENT 'Comment for c1', c2) AS SELECT t1.c1, t1.c2 F
  ROM t1;
  ```

- The RENAME TO clause changes the name of the view, moves the view to a different database, or both.

  For example:

  ```
  ALTER VIEW db1.v1 RENAME TO db2.v2; -- Move the view to a different data
  base with a new name.
  ALTER VIEW db1.v1 RENAME TO db1.v2;  -- Rename the view in the same databa
  se.
  ALTER VIEW db1.v1 RENAME TO db2.v1; -- Move the view to a difference dat
  abase with the same view name.
  ```

- The SET OWNER clause transfers the ownership of the view from the current owner to another user or a role.

  The view owner is originally set to the user who creates the view. When object ownership is enabled in Ranger, an owner of a view can have the ALL with GRANT or ALL without GRANT privilege. The term OWNER is used to differentiate between the ALL privilege that is explicitly granted via the GRANT statement and a privilege that is implicitly granted by the CREATE VIEW statement.

- The SET TBLPROPERTIES clause is primarily a way to associate arbitrary user-specified data items with a particular view.

  You can associate arbitrary items of metadata with a table by specifying the TBLPROPERTIES clause. This clause takes a comma-separated list of key-value pairs and stores those items in the metastore database. You can also unset the view properties later with an UNSET TBLPROPERTIES clause.

  For example:

  ```
  ALTER VIEW v1 SET TBLPROPERTIES ('tblp1' = '1', 'tblp2' = '2');
  ALTER VIEW v1 UNSET TBLPROPERTIES ('tblp1', 'tblp2');
  ```

Statement type: DDL

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

COMMENT statement

The COMMENT statement adds, changes, or removes a comment about a database, a table, or a column.

You can alternatively use the CREATE and ALTER statements to add comments to the objects.

You can view the comment on a database, a table, or a column using the SHOW or DESCRIBE statement.

Syntax:

```
COMMENT ON DATABASE DB_NAME IS {'COMMENT' | NULL}
```

```
COMMENT ON TABLE [DB_NAME.]TABLE_NAME IS {'COMMENT' | NULL}
```

```
COMMENT ON COLUMN [DB_NAME.]TABLE_NAME.COLUMN_NAME IS {'COMMENT' | NULL}
```

Parameters:

- *DB_NAME*: Specify the database name if not for the current database.
- NULL: If given for the comment, removes the existing comment.
- The *COMMENT* string can be up to 256 characters long.

Privileges required:

To add a comment, the ALTER privilege on the object is required.

To view a comment, the SELECT, INSERT, or REFRESH on the object is required.

Usage notes:

Added in: Impala 3.1 release.

COMPUTE STATS statement

The COMPUTE STATS statement gathers information about volume and distribution of data in a table and all associated columns and partitions. The information is stored in the metastore database, and used by Impala to help optimize queries.

For example, if Impala can determine that a table is large or small, or has many or few distinct values it can organize and parallelize the work appropriately for a join query or insert operation. For details about the kinds of information gathered by this statement, see the *Table and column statistics* topic.

Syntax:

```
COMPUTE STATS [DB_NAME.]TABLE_NAME  [ ( COLUMN_LIST ) ] [TABLESAMPLE
 SYSTEM(PERCENTAGE) [REPEATABLE(SEED)]]

COLUMN_LIST ::= COLUMN_NAME [ , COLUMN_NAME, ... ]
COMPUTE INCREMENTAL STATS [DB_NAME.]TABLE_NAME [PARTITION (PARTITION_SPEC)]

PARTITION_SPEC ::= PARTITION_COL=CONSTANT_VALUE

PARTITION_SPEC ::= SIMPLE_PARTITION_SPEC | COMPLEX_PARTITION_SPEC

SIMPLE_PARTITION_SPEC ::= PARTITION_COL=CONSTANT_VALUE

COMPLEX_PARTITION_SPEC ::= COMPARISON_EXPRESSION_ON_PARTITION_COL
```

The PARTITION clause is only allowed in combination with the INCREMENTAL clause. It is optional for COMPUTE INCREMENTAL STATS, and required for DROP INCREMENTAL STATS. Whenever you specify partitions through the PARTITION (*PARTITION_SPEC*) clause in a COMPUTE INCREMENTAL STATS or DROP INCREMENTAL STATS statement, you must include all the partitioning columns in the specification, and specify constant values for all the partition key columns.

Usage notes:

Originally, Impala relied on users to run the Hive ANALYZE TABLE statement, but that method of gathering statistics proved unreliable and difficult to use. The Impala COMPUTE STATS statement was built to improve the reliability and user-friendliness of this operation. COMPUTE STATS does not require any setup steps or special configuration. You only run a single Impala COMPUTE STATS statement to gather both table and column statistics, rather than separate Hive ANALYZE TABLE statements for each kind of statistics.

For non-incremental COMPUTE STATS statement, the columns for which statistics are computed can be specified with an optional comma-separate list of columns.

If no column list is given, the COMPUTE STATS statement computes column-level statistics for all columns of the table. This adds potentially unneeded work for columns whose stats are not needed by queries. It can be especially costly for very wide tables and unneeded large string fields.

COMPUTE STATS returns an error when a specified column cannot be analyzed, such as when the column does not exist, the column is of an unsupported type for COMPUTE STATS, e.g. colums of complex types, or the column is a partitioning column.

If an empty column list is given, no column is analyzed by COMPUTE STATS.

In Impala 2.12 and higher, an optional TABLESAMPLE clause immediately after a table reference specifies that the COMPUTE STATS operation only processes a specified percentage of the table data. For tables that are so large that a full COMPUTE STATS operation is impractical, you can use COMPUTE STATS with a TABLESAMPLE clause to extrapolate statistics from a sample of the table data. See *Table and column statistics* about the experimental stats extrapolation and sampling features.

The COMPUTE INCREMENTAL STATS variation is a shortcut for partitioned tables that works on a subset of partitions rather than the entire table. The incremental nature makes it suitable for large tables with many partitions, where a full COMPUTE STATS operation takes too long to be practical each time a partition is added or dropped.

Important:

For a particular table, use either COMPUTE STATS or COMPUTE INCREMENTAL STATS. The two kinds of stats do not interoperate with each other at the table level. Without dropping the stats, if you run COMPUTE INCREMENTAL STATS it will overwrite the full compute stats or if you run COMPUTE STATS it will drop all incremental stats for consistency.

When you run COMPUTE INCREMENTAL STATS on a table for the first time, the statistics are computed again from scratch regardless of whether the table already has statistics. Therefore, expect a one-time resource-intensive operation for scanning the entire table when running COMPUTE INCREMENTAL STATS for the first time on a given table.

In Impala 3.0 and lower, approximately 400 bytes of metadata per column per partition are needed for caching. Tables with a big number of partitions and many columns can add up to a significant memory overhead as the metadata must be cached on the `catalogd` host and on every `impalad` host that is eligible to be a coordinator. If this metadata for all tables exceeds 2 GB, you might experience service downtime. In Impala 3.1 and higher, the issue was alleviated with an improved handling of incremental stats.

COMPUTE INCREMENTAL STATS only applies to partitioned tables. If you use the INCREMENTAL clause for an unpartitioned table, Impala automatically uses the original COMPUTE STATS statement. Such tables display false under the Incremental stats column of the SHOW TABLE STATS output.

Note:

Because many of the most performance-critical and resource-intensive operations rely on table and column statistics to construct accurate and efficient plans, COMPUTE STATS is an important step at the end of your ETL process. Run COMPUTE STATS on all tables as your first step during performance tuning for slow queries, or troubleshooting for out-of-memory conditions:

- Accurate statistics help Impala construct an efficient query plan for join queries, improving performance and reducing memory usage.
- Accurate statistics help Impala distribute the work effectively for insert operations into Parquet tables, improving performance and reducing memory usage.
- Accurate statistics help Impala estimate the memory required for each query, which is important when you use resource management features, such as admission control and the YARN resource management framework. The statistics help Impala to achieve high concurrency, full utilization of available memory, and avoid contention with workloads from other Hadoop components.
- In Impala 2.8 and higher, when you run the COMPUTE STATS or COMPUTE INCREMENTAL STATS statement against a Parquet table, Impala automatically applies the query option setting MT_DOP=4 to increase the amount of intra-node parallelism during this CPU-intensive operation. See MT_DOP query option for details about what this query option does and how to use it with CPU-intensive SELECT statements.

Computing stats for groups of partitions:

In Impala 2.8 and higher, you can run COMPUTE INCREMENTAL STATS on multiple partitions, instead of the entire table or one partition at a time. You include comparison operators other than = in the PARTITION clause, and the COMPUTE INCREMENTAL STATS statement applies to all partitions that match the comparison expression.

For example, the INT_PARTITIONS table contains 4 partitions. The following COMPUTE INCREMENTAL STATS statements affect some but not all partitions, as indicated by the Updated *N* partition(s) messages. The partitions that are affected depend on values in the partition key column X that match the comparison expression in the PARTITION clause.

```
show partitions int_partitions;
+-------+-------+--------+------+--------------+-------------------+-------
--+...
| x     | #Rows | #Files | Size | Bytes Cached | Cache Replication | Format
  |...
+-------+-------+--------+------+--------------+-------------------+-------
--+...
```

```
| 99    | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | PARQ
UET |...
| 120   | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT
 |...
| 150   | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT
  |...
| 200   | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT
 |...
| Total | -1    | 0      | 0B   | 0B           |                   |
  |...
+-------+-------+--------+------+--------------+-------------------+-------
--+...

compute incremental stats int_partitions partition (x < 100);
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 1 column(s). |
+-----------------------------------------+

compute incremental stats int_partitions partition (x in (100, 150, 200));
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 2 partition(s) and 1 column(s). |
+-----------------------------------------+

compute incremental stats int_partitions partition (x between 100 and 175);
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 2 partition(s) and 1 column(s). |
+-----------------------------------------+

compute incremental stats int_partitions partition (x in (100, 150, 200) or
x < 100);
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 3 partition(s) and 1 column(s). |
+-----------------------------------------+

compute incremental stats int_partitions partition (x != 150);
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 3 partition(s) and 1 column(s). |
+-----------------------------------------+
```

Complex type considerations:

Currently, the statistics created by the COMPUTE STATS statement do not include information about complex type columns. The column stats metrics for complex columns are always shown as -1. For queries involving complex type columns, Impala uses heuristics to estimate the data distribution within such columns.

HBase considerations:

COMPUTE STATS works for HBase tables also. The statistics gathered for HBase tables are somewhat different than for HDFS-backed tables, but that metadata is still used for optimization when HBase tables are involved in join queries.

Amazon S3 considerations:

COMPUTE STATS also works for tables where data resides in the Amazon Simple Storage Service (S3).

Performance considerations:

The statistics collected by COMPUTE STATS are used to optimize join queries INSERT operations into Parquet tables, and other resource-intensive kinds of SQL statements.

For large tables, the COMPUTE STATS statement itself might take a long time and you might need to tune its performance. The COMPUTE STATS statement does not work with the EXPLAIN statement, or the SUMMARY command in `impala-shell`. You can use the PROFILE statement in `impala-shell` to examine timing information for the statement as a whole. If a basic COMPUTE STATS statement takes a long time for a partitioned table, consider switching to the COMPUTE INCREMENTAL STATS syntax so that only newly added partitions are analyzed each time.

Examples:

This example shows two tables, T1 and T2, with a small number distinct values linked by a parent-child relationship between T1.ID and T2.PARENT. T1 is tiny, while T2 has approximately 100K rows. Initially, the statistics includes physical measurements such as the number of files, the total size, and size measurements for fixed-length columns such as with the INT type. Unknown values are represented by -1. After running COMPUTE STATS for each table, much more information is available through the SHOW STATS statements. If you were running a join query involving both of these tables, you would need statistics for both tables to get the most effective optimization for the query.

```
[localhost:21000] > show table stats t1;
Query: show table stats t1
+-------+--------+------+--------+
| #Rows | #Files | Size | Format |
+-------+--------+------+--------+
| -1    | 1      | 33B  | TEXT   |
+-------+--------+------+--------+
Returned 1 row(s) in 0.02s
[localhost:21000] > show table stats t2;
Query: show table stats t2
+-------+--------+----------+--------+
| #Rows | #Files | Size     | Format |
+-------+--------+----------+--------+
| -1    | 28     | 960.00KB | TEXT   |
+-------+--------+----------+--------+
Returned 1 row(s) in 0.01s
[localhost:21000] > show column stats t1;
Query: show column stats t1
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| id     | INT    | -1               | -1     | 4        | 4        |
| s      | STRING | -1               | -1     | -1       | -1       |
+--------+--------+------------------+--------+----------+----------+
Returned 2 row(s) in 1.71s
[localhost:21000] > show column stats t2;
Query: show column stats t2
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| parent | INT    | -1               | -1     | 4        | 4        |
| s      | STRING | -1               | -1     | -1       | -1       |
+--------+--------+------------------+--------+----------+----------+
Returned 2 row(s) in 0.01s
[localhost:21000] > compute stats t1;
Query: compute stats t1
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
```

```
| Updated 1 partition(s) and 2 column(s). |
+-----------------------------------------+
Returned 1 row(s) in 5.30s
[localhost:21000] > show table stats t1;
Query: show table stats t1
+-------+--------+------+--------+
| #Rows | #Files | Size | Format |
+-------+--------+------+--------+
| 3     | 1      | 33B  | TEXT   |
+-------+--------+------+--------+
Returned 1 row(s) in 0.01s
[localhost:21000] > show column stats t1;
Query: show column stats t1
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| id     | INT    | 3                | -1     | 4        | 4        |
| s      | STRING | 3                | -1     | -1       | -1       |
+--------+--------+------------------+--------+----------+----------+
Returned 2 row(s) in 0.02s
[localhost:21000] > compute stats t2;
Query: compute stats t2
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 2 column(s). |
+-----------------------------------------+
Returned 1 row(s) in 5.70s
[localhost:21000] > show table stats t2;
Query: show table stats t2
+-------+--------+----------+--------+
| #Rows | #Files | Size     | Format |
+-------+--------+----------+--------+
| 98304 | 1      | 960.00KB | TEXT   |
+-------+--------+----------+--------+
Returned 1 row(s) in 0.03s
[localhost:21000] > show column stats t2;
Query: show column stats t2
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| parent | INT    | 3                | -1     | 4        | 4        |
| s      | STRING | 6                | -1     | 14       | 9.3      |
+--------+--------+------------------+--------+----------+----------+
Returned 2 row(s) in 0.01s
```

The following example shows how to use the INCREMENTAL clause, available in Impala 2.1.0 and higher. The COMPUTE INCREMENTAL STATS syntax lets you collect statistics for newly added or changed partitions, without rescanning the entire table.

```
-- Initially the table has no incremental stats, as indicated
-- 'false' under Incremental stats.
show table stats item_partitioned;
+-------------+-------+--------+----------+--------------+---------+-------
-----------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Increme
ntal stats
+-------------+-------+--------+----------+--------------+---------+-----
-------------
| Books       | -1    | 1      | 223.74KB | NOT CACHED   | PARQUET | false
| Children    | -1    | 1      | 230.05KB | NOT CACHED   | PARQUET | false
| Electronics | -1    | 1      | 232.67KB | NOT CACHED   | PARQUET | false
| Home        | -1    | 1      | 232.56KB | NOT CACHED   | PARQUET | false
```

```
| Jewelry     | -1    | 1      | 223.72KB | NOT CACHED   | PARQUET | false
| Men         | -1    | 1      | 231.25KB | NOT CACHED   | PARQUET | false
| Music       | -1    | 1      | 237.90KB | NOT CACHED   | PARQUET | false
| Shoes       | -1    | 1      | 234.90KB | NOT CACHED   | PARQUET | false
| Sports      | -1    | 1      | 227.97KB | NOT CACHED   | PARQUET | false
| Women       | -1    | 1      | 226.27KB | NOT CACHED   | PARQUET | false
| Total       | -1    | 10     | 2.25MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+-------
-----------
-- After the first COMPUTE INCREMENTAL STATS,
-- all partitions have stats. The first
-- COMPUTE INCREMENTAL STATS scans the whole
-- table, discarding any previous stats from
-- a traditional COMPUTE STATS statement.
compute incremental stats item_partitioned;
+-------------------------------------------+
| summary                                   |
+-------------------------------------------+
| Updated 10 partition(s) and 21 column(s). |
+-------------------------------------------+
show table stats item_partitioned;
+-------------+-------+--------+----------+--------------+---------+--------
----------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Incr
emental stats
+-------------+-------+--------+----------+--------------+---------+------
------------
| Books       | 1733  | 1      | 223.74KB | NOT CACHED   | PARQUET | true
| Children    | 1786  | 1      | 230.05KB | NOT CACHED   | PARQUET | true
| Electronics | 1812  | 1      | 232.67KB | NOT CACHED   | PARQUET | true
| Home        | 1807  | 1      | 232.56KB | NOT CACHED   | PARQUET | true
| Jewelry     | 1740  | 1      | 223.72KB | NOT CACHED   | PARQUET | true
| Men         | 1811  | 1      | 231.25KB | NOT CACHED   | PARQUET | true
| Music       | 1860  | 1      | 237.90KB | NOT CACHED   | PARQUET | true
| Shoes       | 1835  | 1      | 234.90KB | NOT CACHED   | PARQUET | true
| Sports      | 1783  | 1      | 227.97KB | NOT CACHED   | PARQUET | true
| Women       | 1790  | 1      | 226.27KB | NOT CACHED   | PARQUET | true
| Total       | 17957 | 10     | 2.25MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+------
------------

-- Add a new partition...
alter table item_partitioned add partition (i_category='Camping');
-- Add or replace files in HDFS outside of Impala,
-- rendering the stats for a partition obsolete.
!import_data_into_sports_partition.sh
refresh item_partitioned;
drop incremental stats item_partitioned partition (i_category='Sports');
-- Now some partitions have incremental stats
-- and some do not.
show table stats item_partitioned;
+-------------+-------+--------+----------+--------------+---------+-------
-----------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Increme
ntal stats
+-------------+-------+--------+----------+--------------+---------+-----
-------------
| Books       | 1733  | 1      | 223.74KB | NOT CACHED   | PARQUET | true
| Camping     | -1    | 1      | 408.02KB | NOT CACHED   | PARQUET | false
| Children    | 1786  | 1      | 230.05KB | NOT CACHED   | PARQUET | true
| Electronics | 1812  | 1      | 232.67KB | NOT CACHED   | PARQUET | true
| Home        | 1807  | 1      | 232.56KB | NOT CACHED   | PARQUET | true
| Jewelry     | 1740  | 1      | 223.72KB | NOT CACHED   | PARQUET | true
| Men         | 1811  | 1      | 231.25KB | NOT CACHED   | PARQUET | true
```

```
| Music       | 1860  | 1      | 237.90KB | NOT CACHED   | PARQUET | true
| Shoes       | 1835  | 1      | 234.90KB | NOT CACHED   | PARQUET | true
| Sports      | -1    | 1      | 227.97KB | NOT CACHED   | PARQUET | false
| Women       | 1790  | 1      | 226.27KB | NOT CACHED   | PARQUET | true
| Total       | 17957 | 11     | 2.65MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+-----
-------------

-- After another COMPUTE INCREMENTAL STATS,
-- all partitions have incremental stats, and only the 2
-- partitions without incremental stats were scanned.
compute incremental stats item_partitioned;
+------------------------------------------+
| summary                                  |
+------------------------------------------+
| Updated 2 partition(s) and 21 column(s). |
+------------------------------------------+
show table stats item_partitioned;
+-------------+-------+--------+----------+--------------+---------+------
------------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Increm
ental stats
+-------------+-------+--------+----------+--------------+---------+--------
----------
| Books       | 1733  | 1      | 223.74KB | NOT CACHED   | PARQUET | true
| Camping     | 5328  | 1      | 408.02KB | NOT CACHED   | PARQUET | true
| Children    | 1786  | 1      | 230.05KB | NOT CACHED   | PARQUET | true
| Electronics | 1812  | 1      | 232.67KB | NOT CACHED   | PARQUET | true
| Home        | 1807  | 1      | 232.56KB | NOT CACHED   | PARQUET | true
| Jewelry     | 1740  | 1      | 223.72KB | NOT CACHED   | PARQUET | true
| Men         | 1811  | 1      | 231.25KB | NOT CACHED   | PARQUET | true
| Music       | 1860  | 1      | 237.90KB | NOT CACHED   | PARQUET | true
| Shoes       | 1835  | 1      | 234.90KB | NOT CACHED   | PARQUET | true
| Sports      | 1783  | 1      | 227.97KB | NOT CACHED   | PARQUET | true
| Women       | 1790  | 1      | 226.27KB | NOT CACHED   | PARQUET | true
| Total       | 17957 | 11     | 2.65MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+------
------------
```

File format considerations:

The COMPUTE STATS statement works with tables created with any of the file formats supported by Impala. The following considerations apply to COMPUTE STATS depending on the file format of the table.

The COMPUTE STATS statement works with text tables with no restrictions. These tables can be created through either Impala or Hive.

The COMPUTE STATS statement works with Parquet tables. These tables can be created through either Impala or Hive.

The COMPUTE STATS statement works with Avro tables without restriction in Impala 2.2 and higher. In earlier releases, COMPUTE STATS worked only for Avro tables created through Hive, and required the CREATE TABLE statement to use SQL-style column names and types rather than an Avro-style schema specification.

The COMPUTE STATS statement works with RCFile tables with no restrictions. These tables can be created through either Impala or Hive.

The COMPUTE STATS statement works with SequenceFile tables with no restrictions. These tables can be created through either Impala or Hive.

The COMPUTE STATS statement works with partitioned tables, whether all the partitions use the same file format, or some partitions are defined through ALTER TABLE to use different file formats.

Statement type: DDL

Cancellation: Certain multi-stage statements (CREATE TABLE AS SELECT and COMPUTE STATS) can be cancelled during some stages, when running INSERT or SELECT operations internally. To cancel this statement, use Ctrl-C from the `impala-shell` interpreter, the Cancel button from the Watch page in Hue, Actions > Cancel from the Queries list in Cloudera Manager, or Cancel from the list of in-flight queries (for a particular node) on the Queries tab in the Impala web UI (port 25000).

Restrictions:

Note: Prior to Impala 1.4.0, COMPUTE STATS counted the number of NULL values in each column and recorded that figure in the metastore database. Because Impala does not currently use the NULL count during query planning, Impala 1.4.0 and higher speeds up the COMPUTE STATS statement by skipping this NULL counting.

Internal details:

Behind the scenes, the COMPUTE STATS statement executes two statements: one to count the rows of each partition in the table (or the entire table if unpartitioned) through the COUNT(*) function, and another to count the approximate number of distinct values in each column through the NDV() function. You might see these queries in your monitoring and diagnostic displays. The same factors that affect the performance, scalability, and execution of other queries (such as parallel execution, memory usage, admission control, and timeouts) also apply to the queries run by the COMPUTE STATS statement.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read permission for all affected files in the source directory: all files in the case of an unpartitioned table or a partitioned table in the case of COMPUTE STATS; or all the files in partitions without incremental stats in the case of COMPUTE INCREMEN TAL STATS. It must also have read and execute permissions for all relevant directories holding the data files. (Essentially, COMPUTE STATS requires the same permissions as the underlying SELECT queries it runs against the table.)

Kudu considerations:

The COMPUTE STATS statement applies to Kudu tables. Impala does not compute the number of rows for each partition for Kudu tables. Therefore, you do not need to re-run the operation when you see -1 in the # Rows column of the output from SHOW TABLE STATS. That column always shows -1 for all Kudu tables.

CREATE DATABASE statement

The CREATE DATABASE statement is used to create a new database.

In Impala, a database is both:

- A logical construct for grouping together related tables, views, and functions within their own namespace. You might use a separate database for each application, set of related tables, or round of experimentation.
- A physical construct represented by a directory tree in HDFS. Tables (internal tables), partitions, and data files are all located under this directory. You can perform HDFS-level operations such as backing it up and measuring space usage, or remove it with a DROP DATABASE statement.

Syntax:

```
CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] DATABASE_NAME[COMMENT
 'DATABASE_COMMENT']
  [LOCATION HDFS_PATH];
```

Statement type: DDL

Usage notes:

A database is physically represented as a directory in HDFS, with a filename extension .db, under the main Impala data directory. If the associated HDFS directory does not exist, it is created for you. All databases and their associated directories are top-level objects, with no physical or logical nesting.

After creating a database, to make it the current database within an `impala-shell` session, use the USE statement. You can refer to tables in the current database without prepending any qualifier to their names.

When you first connect to Impala through `impala-shell`, the database you start in (before issuing any CREATE DATABASE or USE statements) is named default.

Impala includes another predefined database, _impala_builtins, that serves as the location for the built-in functions. To see the built-in functions, use a statement like the following:

```
show functions in _impala_builtins;
show functions in _impala_builtins like '*SUBSTRING*';
```

After creating a database, your `impala-shell` session or another `impala-shell` connected to the same node can immediately access that database. To access the database through the Impala daemon on a different node, issue the INVALIDATE METADATA statement first while connected to that other node.

Setting the LOCATION attribute for a new database is a way to work with sets of files in an HDFS directory structure outside the default Impala data directory, as opposed to setting the LOCATION attribute for each individual table.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Hive considerations:

When you create a database in Impala, the database can also be used by Hive. When you create a database in Hive, issue an INVALIDATE METADATA statement in Impala to make Impala permanently aware of the new database.

The SHOW DATABASES statement lists all databases, or the databases whose name matches a wildcard pattern. In Impala 2.5 and higher, the SHOW DATABASES output includes a second column that displays the associated comment, if any, for each database.

Amazon S3 considerations:

To specify that any tables created within a database reside on the Amazon S3 system, you can include an s3a:// prefix on the LOCATION attribute. In Impala 2.6 and higher, Impala automatically creates any required folders as the databases, tables, and partitions are created, and removes them when they are dropped.

In Impala 2.6 and higher, Impala DDL statements such as CREATE DATABASE, CREATE TABLE, DROP DATABASE CASCADE, DROP TABLE, and ALTER TABLE [ADD|DROP] PARTITION can create or remove folders as needed in the Amazon S3 system. Prior to Impala 2.6, you had to create folders yourself and point Impala database, tables, or partitions at them, and manually remove folders when no longer needed.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have write permission for the parent HDFS directory under which the database is located.

Examples:

```
create database first_db;
use first_db;
create table t1 (x int);

create database second_db;
use second_db;
-- Each database has its own namespace for tables.
-- You can reuse the same table names in each database.
create table t1 (s string);

create database temp;

-- You can either USE a database after creating it,
```

```
-- or qualify all references to the table name with the name of the datab
ase.
-- Here, tables T2 and T3 are both created in the TEMP database.

create table temp.t2 (x int, y int);
use database temp;
create table t3 (s string);

-- You cannot drop a database while it is selected by the USE statement.
drop database temp;
ERROR: AnalysisException: Cannot drop current default database: temp

-- The always-available database 'default' is a convenient one to USE
-- before dropping a database you created.
use default;
-- Before dropping a database, first drop all the tables inside it,
-- or in Impala 2.3 and higher use the CASCADE clause.
drop database temp;
ERROR: ImpalaRuntimeException: Error making 'dropDatabase' RPC to Hive Meta
store:
CAUSED BY: InvalidOperationException: Database temp is not empty
show tables in temp;
+------+
| name |
+------+
| t3   |
+------+

-- Impala 2.3 and higher:
drop database temp cascade;

-- Earlier releases:
drop table temp.t3;
drop database temp;
```

Related Information

Impala with Amazon S3

CREATE FUNCTION statement

Creates a user-defined function (UDF), which you can use to implement custom logic during SELECT or INSERT operations.

Syntax:

The syntax is different depending on whether you create a scalar UDF, which is called once for each row and implemented by a single function, or a user-defined aggregate function (UDA), which is implemented by multiple functions that compute intermediate results across sets of rows.

In Impala 2.5 and higher, the syntax is also different for creating or dropping scalar Java-based UDFs. The statements for Java UDFs use a new syntax, without any argument types or return type specified. Java-based UDFs created using the new syntax persist across restarts of the Impala catalog server, and can be shared transparently between Impala and Hive.

To create a persistent scalar C++ UDF with CREATE FUNCTION:

```
CREATE FUNCTION [IF NOT EXISTS]
 [DB_NAME.]FUNCTION_NAME([ARG_TYPE[, ARG_TYPE...]])
  RETURNS RETURN_TYPE
  LOCATION 'HDFS_PATH_TO_DOT_SO'
  SYMBOL='SYMBOL_NAME'
```

To create a persistent Java UDF with CREATE FUNCTION:

```
CREATE FUNCTION [IF NOT EXISTS] [DB_NAME.]FUNCTION_NAME
  LOCATION 'HDFS_PATH_TO_JAR'
  SYMBOL='CLASS_NAME'
```

To create a persistent UDA, which must be written in C++, issue a CREATE AGGREGATE FUNCTION statement:

```
CREATE [AGGREGATE] FUNCTION [IF NOT EXISTS]
 [DB_NAME.]FUNCTION_NAME([ARG_TYPE[, ARG_TYPE...])
  RETURNS RETURN_TYPE
  [INTERMEDIATE TYPE_SPEC]
  LOCATION 'HDFS_PATH'
  [INIT_FN='FUNCTION]
  UPDATE_FN='FUNCTION
  MERGE_FN='FUNCTION
  [PREPARE_FN='FUNCTION]
  [CLOSEFN='FUNCTION]
  [SERIALIZE_FN='FUNCTION]
  [FINALIZE_FN='FUNCTION]
```

Statement type: DDL

Varargs notation:

Note:

Variable-length argument lists are supported for C++ UDFs, but currently not for Java UDFs.

If the underlying implementation of your function accepts a variable number of arguments:

- The variable arguments must go last in the argument list.
- The variable arguments must all be of the same type.
- You must include at least one instance of the variable arguments in every function call invoked from SQL.
- You designate the variable portion of the argument list in the CREATE FUNCTION statement by including ... immediately after the type name of the first variable argument. For example, to create a function that accepts an INT argument, followed by a BOOLEAN, followed by one or more STRING arguments, your CREATE FUNCTION statement would look like:

  ```
  CREATE FUNCTION FUNC_NAME (INT, BOOLEAN, STRING ...)
    RETURNS TYPE LOCATION 'PATH' SYMBOL='ENTRY_POINT';
  ```

Scalar and aggregate functions:

The simplest kind of user-defined function returns a single scalar value each time it is called, typically once for each row in the result set. This general kind of function is what is usually meant by UDF. User-defined aggregate functions (UDAs) are a specialized kind of UDF that produce a single value based on the contents of multiple rows. You usually use UDAs in combination with a GROUP BY clause to condense a large result set into a smaller one, or even a single row summarizing column values across an entire table.

You create UDAs by using the CREATE AGGREGATE FUNCTION syntax. The clauses INIT_FN, UPDATE_FN, MERGE_FN, SERIALIZE_FN, FINALIZE_FN, and INTERMEDIATE only apply when you create a UDA rather than a scalar UDF.

The *_FN clauses specify functions to call at different phases of function processing.

- Initialize: The function you specify with the INIT_FN clause does any initial setup, such as initializing member variables in internal data structures. This function is often a stub for simple UDAs. You can omit this clause and a default (no-op) function will be used.
- Update: The function you specify with the UPDATE_FN clause is called once for each row in the original result set, that is, before any GROUP BY clause is applied. A separate instance of the function is called for each

different value returned by the GROUP BY clause. The final argument passed to this function is a pointer, to which you write an updated value based on its original value and the value of the first argument.

- Merge: The function you specify with the MERGE_FN clause is called an arbitrary number of times, to combine intermediate values produced by different nodes or different threads as Impala reads and processes data files in parallel. The final argument passed to this function is a pointer, to which you write an updated value based on its original value and the value of the first argument.
- Serialize: The function you specify with the SERIALIZE_FN clause frees memory allocated to intermediate results. It is required if any memory was allocated by the Allocate function in the Init, Update, or Merge functions, or if the intermediate type contains any pointers. See the *UDA code samples* for details.
- Finalize: The function you specify with the FINALIZE_FN clause does any required teardown for resources acquired by your UDF, such as freeing memory, closing file handles if you explicitly opened any files, and so on. This function is often a stub for simple UDAs. You can omit this clause and a default (no-op) function will be used. It is required in UDAs where the final return type is different than the intermediate type. or if any memory was allocated by the Allocate function in the Init, Update, or Merge functions. See the *UDA code samples* for details.

If you use a consistent naming convention for each of the underlying functions, Impala can automatically determine the names based on the first such clause, so the others are optional.

Complex type considerations:

Currently, Impala UDFs cannot accept arguments or return values of the Impala complex types (STRUCT, ARRAY, or MAP).

Usage notes:

- When authorization is enabled, the CREATE FUNCTION statement requires:
  - The CREATE privilege on the database.
  - The ALL privilege on two URIs where the URIs are:
    - The JAR file on the file system. For example:

      ```
      GRANT ALL ON URI 'file:///PATH_TO_MY.JAR' TO ROLE MY_ROLE;
      ```

    - The JAR on HDFS. For example:

      ```
      GRANT ALL ON URI 'hdfs:///PATH/TO/JAR' TO ROLE MY_ROLE
      ```

- You can write Impala UDFs in either C++ or Java. C++ UDFs are new to Impala, and are the recommended format for high performance utilizing native code. Java-based UDFs are compatible between Impala and Hive, and are most suited to reusing existing Hive UDFs. (Impala can run Java-based Hive UDFs but not Hive UDAs.)
- Impala 2.5 introduces UDF improvements to persistence for both C++ and Java UDFs, and better compatibility between Impala and Hive for Java UDFs.
- The body of the UDF is represented by a .so or .jar file, which you store in HDFS and the CREATE FUNCTION statement distributes to each Impala node.
- Impala calls the underlying code during SQL statement evaluation, as many times as needed to process all the rows from the result set. All UDFs are assumed to be deterministic, that is, to always return the same result when passed the same argument values. Impala might or might not skip some invocations of a UDF if the result value is already known from a previous call. Therefore, do not rely on the UDF being called a specific number of times, and do not return different result values based on some external factor such as the current time, a random number function, or an external data source that could be updated while an Impala query is in progress.
- The names of the function arguments in the UDF are not significant, only their number, positions, and data types.
- You can overload the same function name by creating multiple versions of the function, each with a different argument signature. For security reasons, you cannot make a UDF with the same name as any built-in function.
- In the UDF code, you represent the function return result as a struct. This struct contains 2 fields. The first field is a boolean representing whether the value is NULL or not. (When this field is true, the return value is interpreted

as NULL.) The second field is the same type as the specified function return type, and holds the return value when the function returns something other than NULL.

- In the UDF code, you represent the function arguments as an initial pointer to a UDF context structure, followed by references to zero or more structs, corresponding to each of the arguments. Each struct has the same 2 fields as with the return value, a boolean field representing whether the argument is NULL, and a field of the appropriate type holding any non-NULL argument value.
- For sample code and build instructions for UDFs, see the *UDA code samples*.
- Because the file representing the body of the UDF is stored in HDFS, it is automatically available to all the Impala nodes. You do not need to manually copy any UDF-related files between servers.
- Because Impala currently does not have any ALTER FUNCTION statement, if you need to rename a function, move it to a different database, or change its signature or other properties, issue a DROP FUNCTION statement for the original function followed by a CREATE FUNCTION with the desired properties.
- Because each UDF is associated with a particular database, either issue a USE statement before doing any CREATE FUNCTION statements, or specify the name of the function as *DB_NAME.FUNCTION_NAME*.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Compatibility:

Impala can run UDFs that were created through Hive, as long as they refer to Impala-compatible data types (not composite or nested column types). Hive can run Java-based UDFs that were created through Impala, but not Impala UDFs written in C++.

The Hive current_user() function cannot be called from a Java UDF through Impala.

Persistence:

In Impala 2.5 and higher, Impala UDFs and UDAs written in C++ are persisted in the metastore database. Java UDFs are also persisted, if they were created with the new CREATE FUNCTION syntax for Java UDFs, where the Java function argument and return types are omitted. Java-based UDFs created with the old CREATE FUNCTION syntax do not persist across restarts because they are held in the memory of the `catalogd` daemon. Until you re-create such Java UDFs using the new CREATE FUNCTION syntax, you must reload those Java-based UDFs by running the original CREATE FUNCTION statements again each time you restart the `catalogd` daemon. Prior to Impala 2.5 the requirement to reload functions after a restart applied to both C++ and Java functions.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Examples:

The following example shows how to take a Java jar file and make all the functions inside one of its classes into UDFs under a single (overloaded) function name in Impala. Each CREATE FUNCTION or DROP FUNCTION statement applies to all the overloaded Java functions with the same name. This example uses the signatureless syntax for CREATE FUNCTION and DROP FUNCTION, which is available in Impala 2.5 and higher.

At the start, the jar file is in the local filesystem. Then it is copied into HDFS, so that it is available for Impala to reference through the CREATE FUNCTION statement and queries that refer to the Impala function name.

```
$ jar -tvf udf-examples-cdh570.jar
     0 Mon Feb 22 04:06:50 PST 2016 META-INF/
   122 Mon Feb 22 04:06:48 PST 2016 META-INF/MANIFEST.MF
     0 Mon Feb 22 04:06:46 PST 2016 com/
     0 Mon Feb 22 04:06:46 PST 2016 com/cloudera/
     0 Mon Feb 22 04:06:46 PST 2016 com/cloudera/impala/
  2460 Mon Feb 22 04:06:46 PST 2016 com/cloudera/impala/IncompatibleUdfTe
st.class
```

```
   541 Mon Feb 22 04:06:46 PST 2016 com/cloudera/impala/TestUdfException.cl
ass
  3438 Mon Feb 22 04:06:46 PST 2016 com/cloudera/impala/JavaUdfTest.class
  5872 Mon Feb 22 04:06:46 PST 2016 com/cloudera/impala/TestUdf.class
...
$ hdfs dfs -put udf-examples-cdh570.jar /user/impala/udfs
$ hdfs dfs -ls /user/impala/udfs
Found 2 items
-rw-r--r--   3 jrussell supergroup        853 2015-10-09 14:05 /user/impala/
udfs/hello_world.jar
-rw-r--r--   3 jrussell supergroup       7366 2016-06-08 14:25 /user/impala/
udfs/udf-examples-cdh570.jar
```

In `impala-shell`, the CREATE FUNCTION refers to the HDFS path of the jar file and the fully qualified class name inside the jar. Each of the functions inside the class becomes an Impala function, each one overloaded under the specified Impala function name.

```
[localhost:21000] > create function testudf location '/user/impala/udfs/udf-
examples-cdh570.jar' symbol='com.cloudera.impala.TestUdf';
[localhost:21000] > show functions;
+-------------+-------------------------------------+-------------+-------
--------+
| return type | signature                           | binary type | is p
ersistent |
+-------------+-------------------------------------+-------------+-------
--------+
| BIGINT      | testudf(BIGINT)                     | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN)                    | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN)           | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN, BOOLEAN)  | JAVA        | true
        |
| DOUBLE      | testudf(DOUBLE)                     | JAVA        | true
        |
| DOUBLE      | testudf(DOUBLE, DOUBLE)             | JAVA        | true
        |
| DOUBLE      | testudf(DOUBLE, DOUBLE, DOUBLE)     | JAVA        | true
        |
| FLOAT       | testudf(FLOAT)                      | JAVA        | true
        |
| FLOAT       | testudf(FLOAT, FLOAT)               | JAVA        | true
        |
| FLOAT       | testudf(FLOAT, FLOAT, FLOAT)        | JAVA        | true
        |
| INT         | testudf(INT)                        | JAVA        | true
        |
| DOUBLE      | testudf(INT, DOUBLE)                | JAVA        | true
        |
| INT         | testudf(INT, INT)                   | JAVA        | true
        |
| INT         | testudf(INT, INT, INT)              | JAVA        | true
        |
| SMALLINT    | testudf(SMALLINT)                   | JAVA        | true
        |
| SMALLINT    | testudf(SMALLINT, SMALLINT)         | JAVA        | true
        |
| SMALLINT    | testudf(SMALLINT, SMALLINT, SMALLINT) | JAVA        | true
        |
| STRING      | testudf(STRING)                     | JAVA        | true
        |
```

```
| STRING      | testudf(STRING, STRING)               | JAVA        | true
        |
| STRING      | testudf(STRING, STRING, STRING)       | JAVA        | true
        |
| TINYINT     | testudf(TINYINT)                      | JAVA        | true
        |
+-------------+---------------------------------------+-------------+-------
--------+
```

These are all simple functions that return their single arguments, or sum, concatenate, and so on their multiple arguments. Impala determines which overloaded function to use based on the number and types of the arguments.

```
insert into bigint_x values (1), (2), (4), (3);
select testudf(x) from bigint_x;
+-----------------+
| udfs.testudf(x) |
+-----------------+
| 1               |
| 2               |
| 4               |
| 3               |
+-----------------+

insert into int_x values (1), (2), (4), (3);
select testudf(x, x+1, x*x) from int_x;
+-------------------------------+
| udfs.testudf(x, x + 1, x * x) |
+-------------------------------+
| 4                             |
| 9                             |
| 25                            |
| 16                            |
+-------------------------------+
select testudf(x) from string_x;
+-----------------+
| udfs.testudf(x) |
+-----------------+
| one             |
| two             |
| four            |
| three           |
+-----------------+
select testudf(x,x) from string_x;
+--------------------+
| udfs.testudf(x, x) |
+--------------------+
| oneone             |
| twotwo             |
| fourfour           |
| threethree         |
+--------------------+
```

The previous example used the same Impala function name as the name of the class. This example shows how the Impala function name is independent of the underlying Java class or function names. A second CREATE FUNCTION statement results in a set of overloaded functions all named my_func, to go along with the overloaded functions all named testudf.

```
create function my_func location '/user/impala/udfs/udf-examples-cdh570.jar'
  symbol='com.cloudera.impala.TestUdf';
```

```
show functions;
+-------------+-------------------------------------+-------------+----
-----------+
| return type | signature                           | binary type | is pe
rsistent |
+-------------+-------------------------------------+-------------+----
-----------+
| BIGINT      | my_func(BIGINT)                     | JAVA        | true
        |
| BOOLEAN     | my_func(BOOLEAN)                    | JAVA        | true
        |
| BOOLEAN     | my_func(BOOLEAN, BOOLEAN)           | JAVA        | true
        |
...
| BIGINT      | testudf(BIGINT)                     | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN)                    | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN)           | JAVA        | true
        |
...
```

The corresponding DROP FUNCTION statement with no signature drops all the overloaded functions with that name.

```
drop function my_func;
show functions;
+-------------+-------------------------------------+-------------+-------
--------+
| return type | signature                           | binary type | is p
ersistent |
+-------------+-------------------------------------+-------------+-------
--------+
| BIGINT      | testudf(BIGINT)                     | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN)                    | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN)           | JAVA        | true
        |
...
```

The signatureless CREATE FUNCTION syntax for Java UDFs ensures that the functions shown in this example remain available after the Impala service (specifically, the Catalog Server) are restarted.

Related Information

UDA sample code

CREATE ROLE statement

The CREATE ROLE statement creates a role to which privileges can be granted. Privileges can be granted to roles, which can then be assigned to a group and the users belonging to the group will be assigned the role. A user that has been assigned a role will only be able to exercise the privileges of that role. Only users with administrative privileges can create/drop roles. By default, the Impala users do not have administrative privileges in Ranger. For Impala users to have administrative privileges in Ranger you must use Ranger Web UI and configure.

Syntax:

```
CREATE ROLE ROLE_NAME
```

Required privileges:

Only administrative users can use this statement.

Compatibility:

Impala makes use of any roles and privileges specified by the GRANT and REVOKE statements in Hive, and Hive makes use of any roles and privileges specified by the GRANT and REVOKE statements in Impala. The Impala GRANT and REVOKE statements for privileges do not require the ROLE keyword to be repeated before each role name, unlike the equivalent Hive statements.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

CREATE TABLE statement

The CREATE TABLE statement creates a new table with the specified properties.

While creating a table, you can optionally specify aspects such as:

- Whether the table is internal or external.
- The columns and associated data types.
- The columns used for physically partitioning the data.
- The file format for data files.
- The HDFS directory where the data files are located.

Syntax:

The general syntax for creating a table and specifying its columns is as follows:

Explicit column definitions:

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
  (COL_NAME DATA_TYPE
    [COMMENT 'COL_COMMENT']
    [, ...]
  )
  [PARTITIONED BY (COL_NAME DATA_TYPE [COMMENT 'COL_COMMENT'], ...)]
  [SORT BY ([COLUMN [, COLUMN ...]])]
  [COMMENT 'TABLE_COMMENT']
  [ROW FORMAT ROW_FORMAT]
  [WITH SERDEPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
  [STORED AS FILE_FORMAT]
  [LOCATION 'HDFS_PATH']
  [CACHED IN 'POOL_NAME' [WITH REPLICATION = INTEGER] | UNCACHED]
  [TBLPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
```

CREATE TABLE AS SELECT:

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] DB_NAME.]TABLE_NAME
    [PARTITIONED BY (COL_NAME[, ...])]
    [SORT BY ([COLUMN [, COLUMN ...]])]
    [COMMENT 'TABLE_COMMENT']
+   [ROW FORMAT ROW_FORMAT]
    [WITH SERDEPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
+   [STORED AS CTAS_FILE_FORMAT]
    [LOCATION 'HDFS_PATH']
+     [CACHED IN 'POOL_NAME' [WITH REPLICATION = INTEGER] | UNCACHED]
    [TBLPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
  AS
    SELECT_STATEMENT
```

Note: If creating a partitioned table, the partition key columns must be listed last in the SELECT columns list, in the same order as in the PARTITIONED BY clause. Otherwise, you will receive an error about a column name mismatch.

```
primitive_type:
    TINYINT
  | SMALLINT
  | INT
  | BIGINT
  | BOOLEAN
  | FLOAT
  | DOUBLE
  | DECIMAL
  | STRING
  | CHAR
  | VARCHAR
  | TIMESTAMP

complex_type:
    struct_type
  | array_type
  | map_type

struct_type: STRUCT < NAME : PRIMITIVE_OR_COMPLEX_TYPE [COMMENT
 'COMMENT_STRING'], ... >

array_type: ARRAY < PRIMITIVE_OR_COMPLEX_TYPE >

map_type: MAP < PRIMITIVE_TYPE, PRIMITIVE_OR_COMPLEX_TYPE >

row_format:
  DELIMITED [FIELDS TERMINATED BY 'CHAR' [ESCAPED BY 'CHAR']]
  [LINES TERMINATED BY 'CHAR']

file_format:
    PARQUET
  | TEXTFILE
  | AVRO
  | SEQUENCEFILE
  | RCFILE

ctas_file_format:
    PARQUET
  | TEXTFILE
```

Column definitions inferred from data file:

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
  LIKE PARQUET 'HDFS_PATH_OF_PARQUET_FILE'
  [PARTITIONED BY (COL_NAME DATA_TYPE [COMMENT 'COL_COMMENT'], ...)]
  [SORT BY ([COLUMN [, COLUMN ...]])]
  [COMMENT 'TABLE_COMMENT']
  [ROW FORMAT ROW_FORMAT]
  [WITH SERDEPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
  [STORED AS FILE_FORMAT]
  [LOCATION 'HDFS_PATH']
  [CACHED IN 'POOL_NAME' [WITH REPLICATION = INTEGER] | UNCACHED]
  [TBLPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
data_type:
    PRIMITIVE_TYPE
  | array_type
  | map_type
```

```
  | struct_type
```

Kudu tables:

```
CREATE TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
  (COL_NAME DATA_TYPE
    [KUDU_COLUMN_ATTRIBUTE ...]
    [COMMENT 'COL_COMMENT']
    [, ...]
    [PRIMARY KEY (COL_NAME[, ...])]
  )
  [PARTITION BY KUDU_PARTITION_CLAUSE]
  [COMMENT 'TABLE_COMMENT']
  STORED AS KUDU
  [TBLPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
```

Kudu column attributes:

```
  PRIMARY KEY
| [NOT] NULL
| ENCODING CODEC
| COMPRESSION ALGORITHM
| DEFAULT CONSTANT
| BLOCK_SIZE NUMBER
```

kudu_partition_clause:

```
kudu_partition_clause ::= [ HASH_CLAUSE [, ...]] [, RANGE_CLAUSE ]

hash_clause ::=
  HASH [ (PK_COL [, ...]) ]
    PARTITIONS N

range_clause ::=
  RANGE [ (PK_COL [, ...]) ]
  (
    {
      PARTITION CONSTANT_EXPRESSION RANGE_COMPARISON_OPERATOR VAL
UES RANGE_COMPARISON_OPERATOR CONSTANT_EXPRESSION
      | PARTITION VALUE = CONSTANT_EXPRESSION_OR_TUPLE
    }
   [, ...]
  )

range_comparison_operator ::= { < | <= }
```

External Kudu tables:

In Impala 3.4 and earlier, you can create an external Kudu table based on a pre-existing Kudu schema using the table property 'kudu.table_name'='internal_kudu_name'.

```
CREATE EXTERNAL TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
      [COMMENT 'COL_COMMENT']
      STORED AS KUDU
      [TBLPROPERTIES ('KUDU.TABLE.NAME'='INTERNAL_KUDU_NAME',
 'KEY1'='VALUE1',...)]
```

Alternatively, in Impala 3.4 and higher, you can also create an external Kudu table as follows:

```
CREATE EXTERNAL TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
```

```
      (COL_NAME DATA_TYPE
      [KUDU_COLUMN_ATTRIBUTE ...]
      [COMMENT 'COL_COMMENT']
      [, ...]
      [PRIMARY KEY (COL_NAME[, ...])]
      )
      [PARTITION BY KUDU_PARTITION_CLAUSE]
      [COMMENT 'TABLE_COMMENT']
      STORED AS KUDU
      [TBLPROPERTIES ('external.table.purge'='true', 'KEY1'='VALUE1',...)]
```

- Use a Hive metastore (HMS) 3 or later.
- Provide column specifications to define the schema when you create the table, similar to creating an internal table.
- Omit the kudu.table_name table property as there is no pre-existing schema.
- Include the required external.table.purge property.

Only the schema metadata is stored in HMS when you create an external table; however, using this create table syntax, drop table on the Kudu external table deletes the data stored outside HMS in Kudu as well as the metadata (schema) inside HMS. Likewise, renaming the table changes the name of the table in HMS and in Kudu. Kudu synchronizes changes to the actual data and metadata; consequently, operations such as dropping a table or altering a table name simulate internal table operations.

CREATE TABLE AS SELECT for Kudu tables:

```
CREATE TABLE [IF NOT EXISTS] DB_NAME.]TABLE_NAME
  [PRIMARY KEY (COL_NAME[, ...])]
  [PARTITION BY KUDU_PARTITION_CLAUSE]
  [COMMENT 'TABLE_COMMENT']
  STORED AS KUDU
  [TBLPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
AS
  SELECT_STATEMENT
```

Statement type: DDL

Column definitions:

Depending on the form of the CREATE TABLE statement, the column definitions are required or not allowed.

With the CREATE TABLE AS SELECT and CREATE TABLE LIKE syntax, you do not specify the columns at all; the column names and types are derived from the source table, query, or data file.

With the basic CREATE TABLE syntax, you must list one or more columns, its name, type, optionally constraints, and optionally a comment, in addition to any columns used as partitioning keys. There is one exception where the column list is not required: when creating an Avro table with the STORED AS AVRO clause, you can omit the list of columns and specify the same metadata as part of the TBLPROPERTIES clause.

Constraints:

Constraints are advisory and intended for estimating cardinality during query planning in a future release; there is no attempt to enforce constraints. Add primary and foreign key information after column definitions. Do not include a constraint name; the constraint name is generated internally as a UUID. The following constraint states are supported:

- DISABLE
- NOVALIDATE
- RELY

The ENABLE, VALIDATE, and NORELY options are not supported. The foreign key must be defined as the primary key in the referenced table.

Constraint examples:

```
CREATE TABLE pk(col1 INT, col2 STRING, PRIMARY KEY(col1, col2));
```

```
CREATE TABLE fk(id INT, col1 INT, col2 STRING, PRIMARY KEY(id),
        FOREIGN KEY(col1, col2) REFERENCES pk(col1, col2));
```

```
CREATE TABLE pk(id INT, PRIMARY KEY(id) DISABLE, NOVALIDATE, RELY);
```

```
CREATE TABLE fk(id INT, col1 INT, col2 STRING, PRIMARY KEY(id),
        FOREIGN KEY(col1, col2) REFERENCES pk(col1, col2));
```

Complex type considerations:

The Impala complex types (STRUCT, ARRAY, or MAP) are available in Impala 2.3 and higher. Because you can nest these types (for example, to make an array of maps or a struct with an array field), these types are also sometimes referred to as nested types.

Impala can create tables containing complex type columns, with any supported file format. Because currently Impala can only query complex type columns in Parquet tables, creating tables with complex type columns and other file formats such as text is of limited use. For example, you might create a text table including some columns with complex types with Impala, and use Hive as part of your to ingest the nested type data and copy it to an identical Parquet table. Or you might create a partitioned table containing complex type columns using one file format, and use ALTER TABLE to change the file format of individual partitions to Parquet; Impala can then query only the Parquet-format partitions in that table.

Partitioned tables can contain complex type columns. All the partition key columns must be scalar types.

Internal and external tables (EXTERNAL and LOCATION clauses):

By default, Impala creates an “internal” table, where Impala manages the underlying data files for the table, and physically deletes the data files when you drop the table. If you specify the EXTERNAL clause, Impala treats the table as an “external” table, where the data files are typically produced outside Impala and queried from their original locations in HDFS, and Impala leaves the data files in place when you drop the table.

Typically, for an external table you include a LOCATION clause to specify the path to the HDFS directory where Impala reads and writes files for the table. For example, if your data pipeline produces Parquet files in the HDFS directory /user/etl/destination, you might create an external table as follows:

```
CREATE EXTERNAL TABLE external_parquet (c1 INT, c2 STRING, c3 TIMESTAMP)
  STORED AS PARQUET LOCATION '/user/etl/destination';
```

Although the EXTERNAL and LOCATION clauses are often specified together, LOCATION is optional for external tables, and you can also specify LOCATION for internal tables. The difference is all about whether Impala “takes control” of the underlying data files and moves them when you rename the table, or deletes them when you drop the table. For more about internal and external tables and how they interact with the LOCATION attribute, see *Overview of Impala tables*.

Partitioned tables (PARTITIONED BY clause):

The PARTITIONED BY clause divides the data files based on the values from one or more specified columns. Impala queries can use the partition metadata to minimize the amount of data that is read from disk or transmitted across the network, particularly during join queries.

Note:

All Kudu tables require partitioning, which involves different syntax than non-Kudu tables. See the PARTITION BY clause, rather than PARTITIONED BY, for Kudu tables.

In Impala 2.10 and higher, the PARTITION BY clause is optional for Kudu tables. If the clause is omitted, Impala automatically constructs a single partition that is not connected to any column. Because such a table cannot take advantage of Kudu features for parallelized queries and query optimizations, omitting the PARTITION BY clause is only appropriate for small lookup tables.

Prior to Impala 2.5, you could use a partitioned table as the source and copy data from it, but could not specify any partitioning clauses for the new table. In Impala 2.5 and higher, you can now use the PARTITIONED BY clause with a CREATE TABLE AS SELECT statement. See the examples under the following discussion of the CREATE TABLE AS SELECT syntax variation.

Sorted tables (SORT BY clause):

The optional SORT BY clause lets you specify zero or more columns that are sorted in the data files created by each Impala INSERT or CREATE TABLE AS SELECT operation. Creating data files that are sorted is most useful for Parquet tables, where the metadata stored inside each file includes the minimum and maximum values for each column in the file. (The statistics apply to each row group within the file; for simplicity, Impala writes a single row group in each file.) Grouping data values together in relatively narrow ranges within each data file makes it possible for Impala to quickly skip over data files that do not contain value ranges indicated in the WHERE clause of a query, and can improve the effectiveness of Parquet encoding and compression.

This clause is not applicable for Kudu tables or HBase tables. Although it works for other HDFS file formats besides Parquet, the more efficient layout is most evident with Parquet tables, because each Parquet data file includes statistics about the data values in that file.

The SORT BY columns cannot include any partition key columns for a partitioned table, because those column values are not represented in the underlying data files.

Because data files can arrive in Impala tables by mechanisms that do not respect the SORT BY clause, such as LOAD DATA or ETL tools that create HDFS files, Impala does not guarantee or rely on the data being sorted. The sorting aspect is only used to create a more efficient layout for Parquet files generated by Impala, which helps to optimize the processing of those Parquet files during Impala queries. During an INSERT or CREATE TABLE AS SELECT operation, the sorting occurs when the SORT BY clause applies to the destination table for the data, regardless of whether the source table has a SORT BY clause.

For example, when creating a table intended to contain census data, you might define sort columns such as last name and state. If a data file in this table contains a narrow range of last names, for example from Smith to Smythe, Impala can quickly detect that this data file contains no matches for a WHERE clause such as WHERE last_name = 'Jones' and avoid reading the entire file.

```
CREATE TABLE census_data (last_name STRING, first_name STRING, state STRING,
 address STRING)
  SORT BY (last_name, state)
  STORED AS PARQUET;
```

Likewise, if an existing table contains data without any sort order, you can reorganize the data in a more efficient way by using INSERT or CREATE TABLE AS SELECT to copy that data into a new table with a SORT BY clause:

```
CREATE TABLE sorted_census_data
  SORT BY (last_name, state)
  STORED AS PARQUET
  AS SELECT last_name, first_name, state, address
    FROM unsorted_census_data;
```

The metadata for the SORT BY clause is stored in the TBLPROPERTIES fields for the table. Other SQL engines that can interoperate with Impala tables, such as Hive and Spark SQL, do not recognize this property when inserting into a table that has a SORT BY clause.

Transactional tables:

In the version 3.3 and higher, when integrated with Hive 3, Impala can create, read, and insert into transactional tables.

To create a table that supports transactions, use the TBLPROPERTIES clause and set the 'transactional' and 'transac tional_properties' as below. Currently, Impala only supports insert-only transactional tables.

```
TBLPROPERTIES('transactional'='true', 'transactional_properties'='insert_onl
y')
```

When integrated with Hive3 and the DEFAULT_TRANSACTIONAL_TYPE query option is set to INSERT_ONLY, tables are created as insert-only transactional table by default.

Transactional tables are not supported for Kudu and HBase.

Kudu considerations:

Because Kudu tables do not support clauses related to HDFS and S3 data files and partitioning mechanisms, the syntax associated with the STORED AS KUDU clause is shown separately in the above syntax descriptions. Kudu tables have their own syntax for CREATE TABLE, CREATE EXTERNAL TABLE, and CREATE TABLE AS SELECT. Prior to Impala 2.10, all internal Kudu tables require a PARTITION BY clause, different than the PART ITIONED BY clause for HDFS-backed tables.

Here are some examples of creating empty Kudu tables:

```
-- Single partition. Only for  Impala 2.10 and higher.
-- Only suitable for small lookup tables.
CREATE TABLE kudu_no_partition_by_clause
  (
    id bigint PRIMARY KEY, s STRING, b BOOLEAN
  )
  STORED AS KUDU;

-- Single-column primary key.
CREATE TABLE kudu_t1 (id BIGINT PRIMARY key, s STRING, b BOOLEAN)
  PARTITION BY HASH (id) PARTITIONS 20 STORED AS KUDU;

-- Multi-column primary key.
CREATE TABLE kudu_t2 (id BIGINT, s STRING, b BOOLEAN, PRIMARY KEY (id,s))
  PARTITION BY HASH (s) PARTITIONS 30 STORED AS KUDU;

-- Meaningful primary key column is good for range partitioning.
CREATE TABLE kudu_t3 (id BIGINT, year INT, s STRING,
    b BOOLEAN, PRIMARY KEY (id,year))
  PARTITION BY HASH (id) PARTITIONS 20,
  RANGE (year) (PARTITION 1980 <= VALUES < 1990,
    PARTITION 1990 <= VALUES < 2000,
    PARTITION VALUE = 2001,
    PARTITION 2001 < VALUES)
  STORED AS KUDU;
```

Here is an example of creating an external Kudu table:

```
-- Inherits column definitions from original table.
-- For tables created through Impala, the kudu.table_name property
-- comes from DESCRIBE FORMATTED output from the original table.
CREATE EXTERNAL TABLE external_t1 STORED AS KUDU
  TBLPROPERTIES ('kudu.table_name'='kudu_tbl_created_via_api');
```

In Impala 3.4 and higher, by default HMS implicitly translates internal Kudu tables to external Kudu tables with the 'external.table.purge' property set to true. You can explicitly create such external Kudu tables similar to the way you create internal Kudu tables. You must set the table property 'external.table.purge' to true. Here is an example of creating an external Kudu table:

```
CREATE EXTERNAL TABLE myextkudutbl (
      id int PRIMARY KEY,
      name string)
      PARTITION BY HASH PARTITIONS 8
      STORED AS KUDU
      TBLPROPERTIES ('external.table.purge'='true');
```

Operations on the resulting external table in Impala, HMS, and Kudu table metadata is synchronized. HMS-Kudu integration does not need to be enabled for external table synchronization. Such synchronized tables behave similar to internal tables. For example, dropping a table removes the underlying Kudu table data as well as the table metadata in HMS.

If you want to drop only the HMS metadata and not drop the Kudu table, you can set external.table.purge to false, as shown in the following example:

```
      ALTER TABLE myextkudutbl set tblproperties('external.table.purge'='fa
lse');
```

Here is an example of CREATE TABLE AS SELECT syntax for a Kudu table:

```
-- The CTAS statement defines the primary key and partitioning scheme.
-- The rest of the column definitions are derived from the select list.
CREATE TABLE ctas_t1
  PRIMARY KEY (id) PARTITION BY HASH (id) PARTITIONS 10
  STORED AS KUDU
  AS SELECT id, s FROM kudu_t1;
```

The following CREATE TABLE clauses are not supported for Kudu tables:

- PARTITIONED BY (Kudu tables use the clause PARTITION BY instead)
- LOCATION
- ROWFORMAT
- CACHED IN | UNCACHED
- WITH SERDEPROPERTIES

Partitioning for Kudu tables (PARTITION BY clause)

For Kudu tables, you specify logical partitioning across one or more columns using the PARTITION BY clause. In contrast to partitioning for HDFS-based tables, multiple values for a partition key column can be located in the same partition. The optional HASH clause lets you divide one or a set of partition key columns into a specified number of buckets. You can use more than one HASH clause, specifying a distinct set of partition key columns for each. The optional RANGE clause further subdivides the partitions, based on a set of comparison operations for the partition key columns.

Here are some examples of the PARTITION BY HASH syntax:

```
-- Apply hash function to 1 primary key column.
create table hash_t1 (x bigint, y bigint, s string, primary key (x,y))
  partition by hash (x) partitions 10
  stored as kudu;
```

```
-- Apply hash function to a different primary key column.
create table hash_t2 (x bigint, y bigint, s string, primary key (x,y))
  partition by hash (y) partitions 10
  stored as kudu;

-- Apply hash function to both primary key columns.
-- In this case, the total number of partitions is 10.
create table hash_t3 (x bigint, y bigint, s string, primary key (x,y))
  partition by hash (x,y) partitions 10
  stored as kudu;

-- When the column list is omitted, apply hash function to all primary key c
olumns.
create table hash_t4 (x bigint, y bigint, s string, primary key (x,y))
  partition by hash partitions 10
  stored as kudu;

-- Hash the X values independently from the Y values.
-- In this case, the total number of partitions is 10 x 20.
create table hash_t5 (x bigint, y bigint, s string, primary key (x,y))
  partition by hash (x) partitions 10, hash (y) partitions 20
  stored as kudu;
```

Here are some examples of the PARTITION BY RANGE syntax:

```
-- Create partitions that cover every possible value of X.
-- Ranges that span multiple values use the keyword VALUES between
-- a pair of < and <= comparisons.
create table range_t1 (x bigint, s string, s2 string, primary key (x, s))
  partition by range (x)
    (
      partition 0 <= values <= 49, partition 50 <= values <= 100,
      partition values < 0, partition 100 < values
    )
  stored as kudu;
-- Create partitions that cover some possible values of X.
-- Values outside the covered range(s) are rejected.
-- New range partitions can be added through ALTER TABLE.
create table range_t2 (x bigint, s string, s2 string, primary key (x, s))
  partition by range (x)
    (
      partition 0 <= values <= 49, partition 50 <= values <= 100
    )
  stored as kudu;

-- A range can also specify a single specific value, using the keyword VALUE
-- with an = comparison.
create table range_t3 (x bigint, s string, s2 string, primary key (x, s))
  partition by range (s)
    (
      partition value = 'Yes', partition value = 'No', partition value = 'M
aybe'
    )
  stored as kudu;

-- Using multiple columns in the RANGE clause and tuples inside the parti
tion spec
-- only works for partitions specified with the VALUE= syntax.
create table range_t4 (x bigint, s string, s2 string, primary key (x, s))
  partition by range (x,s)
    (
```

```
      partition value = (0,'zero'), partition value = (1,'one'), partition v
alue = (2,'two')
    )
  stored as kudu;
```

Here are some examples combining both HASH and RANGE syntax for the PARTITION BY clause:

```
-- Values from each range partition are hashed into 10 associated buckets.
-- Total number of partitions in this case is 10 x 2.
create table combined_t1 (x bigint, s string, s2 string, primary key (x, s))
  partition by hash (x) partitions 10, range (x)
    (
      partition 0 <= values <= 49, partition 50 <= values <= 100
    )
  stored as kudu;

-- The hash partitioning and range partitioning can apply to different colum
ns.
-- But all the columns used in either partitioning scheme must be from the p
rimary key.
create table combined_t2 (x bigint, s string, s2 string, primary key (x, s))
  partition by hash (s) partitions 10, range (x)
    (
      partition 0 <= values <= 49, partition 50 <= values <= 100
    )
  stored as kudu;
```

Specifying file format (STORED AS and ROW FORMAT clauses):

The STORED AS clause identifies the format of the underlying data files. Currently, Impala can query more types of file formats than it can create or insert into. Use Hive to perform any create or data load operations that are not currently available in Impala. For example, Impala can create an Avro, SequenceFile, or RCFile table but cannot insert data into it. There are also Impala-specific procedures for using compression with each kind of file format. For details about working with data files of various formats, see *How Impala works with Hadoop fle formats*.

Note: In Impala 1.4.0 and higher, Impala can create Avro tables, which formerly required doing the CREATE TABLE statement in Hive. See *Using the Avro file format with Impala Tables* for details and examples.

By default (when no STORED AS clause is specified), data files in Impala tables are created as text files with Ctrl-A (hex 01) characters as the delimiter. Specify the ROW FORMAT DELIMITED clause to produce or ingest data files that use a different delimiter character such as tab or |, or a different line end character such as carriage return or newline. When specifying delimiter and line end characters with the FIELDS TERMINATED BY and LINES TERMINATED BY clauses, use '\t' for tab, '\n' for newline or linefeed, '\r' for carriage return, and \0 for ASCII nul (hex 00).

The ESCAPED BY clause applies both to text files that you create through an INSERT statement to an Impala TEXT FILE table, and to existing data files that you put into an Impala table directory. (You can ingest existing data files either by creating the table with CREATE EXTERNAL TABLE ... LOCATION, the LOAD DATA statement, or through an HDFS operation such as hdfs dfs -put *FILE HDFS_PATH*.) Choose an escape character that is not used anywhere else in the file, and put it in front of each instance of the delimiter character that occurs within a field value. Surrounding field values with quotation marks does not help Impala to parse fields with embedded delimiter characters; the quotation marks are considered to be part of the column value. If you want to use \ as the escape character, specify the clause in `impala-shell` as ESCAPED BY '\\'.

Note: The CREATE TABLE clauses FIELDS TERMINATED BY, ESCAPED BY, and LINES TERMINATED BY have special rules for the string literal used for their argument, because they all require a single character. You can use a regular character surrounded by single or double quotation marks, an octal sequence such as '\054' (representing a comma), or an integer in the range '-127'..'128' (with quotation marks but no backslash), which is interpreted as a single-byte ASCII character. Negative values are subtracted from 256; for example, FIELDS TERMINATED BY '-2' sets the field delimiter to ASCII code 254, the “Icelandic Thorn” character used as a delimiter by some data formats.

Cloning tables (LIKE clause):

To create an empty table with the same columns, comments, and other attributes as another table, use the following variation. The CREATE TABLE ... LIKE form allows a restricted set of clauses, currently only the LOCATION, COMMENT, and STORED AS clauses.

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
  LIKE { [DB_NAME.]TABLE_NAME | PARQUET 'HDFS_PATH_OF_PARQUET_FILE' }
  [COMMENT 'TABLE_COMMENT']
  [STORED AS FILE_FORMAT]
  [LOCATION 'HDFS_PATH']
```

Note:

To clone the structure of a table and transfer data into it in a single operation, use the CREATE TABLE AS SELECT syntax described in the next subsection.

When you clone the structure of an existing table using the CREATE TABLE ... LIKE syntax, the new table keeps the same file format as the original one, so you only need to specify the STORED AS clause if you want to use a different file format, or when specifying a view as the original table. (Creating a table “like” a view produces a text table by default.)

Although normally Impala cannot create an HBase table directly, Impala can clone the structure of an existing HBase table with the CREATE TABLE ... LIKE syntax, preserving the file format and metadata from the original table.

There are some exceptions to the ability to use CREATE TABLE ... LIKE with an Avro table. For example, you cannot use this technique for an Avro table that is specified with an Avro schema but no columns. When in doubt, check if a CREATE TABLE ... LIKE operation works in Hive; if not, it typically will not work in Impala either.

If the original table is partitioned, the new table inherits the same partition key columns. Because the new table is initially empty, it does not inherit the actual partitions that exist in the original one. To create partitions in the new table, insert data or issue ALTER TABLE ... ADD PARTITION statements.

Prior to Impala 1.4.0, it was not possible to use the CREATE TABLE LIKE *VIEW_NAME* syntax. In Impala 1.4.0 and higher, you can create a table with the same column definitions as a view using the CREATE TABLE LIKE technique. Although CREATE TABLE LIKE normally inherits the file format of the original table, a view has no underlying file format, so CREATE TABLE LIKE *VIEW_NAME* produces a text table by default. To specify a different file format, include a STORED AS *FILE_FORMAT* clause at the end of the CREATE TABLE LIKE statement.

Because CREATE TABLE ... LIKE only manipulates table metadata, not the physical data of the table, issue INSERT INTO TABLE statements afterward to copy any data from the original table into the new one, optionally converting the data to a new file format. For some file formats, Impala can do a CREATE TABLE ... LIKE to create the table, but Impala cannot insert data in that file format; in these cases, you must load the data in Hive.

CREATE TABLE AS SELECT:

The CREATE TABLE AS SELECT syntax is a shorthand notation to create a table based on column definitions from another table, and copy data from the source table to the destination table without issuing any separate INSERT statement. This idiom is so popular that it has its own acronym, “CTAS”.

The following examples show how to copy data from a source table T1 to a variety of destinations tables, applying various transformations to the table properties, table layout, or the data itself as part of the operation:

```
-- Sample table to be the source of CTAS operations.
CREATE TABLE t1 (x INT, y STRING);
INSERT INTO t1 VALUES (1, 'one'), (2, 'two'), (3, 'three');

-- Clone all the columns and data from one table to another.
CREATE TABLE clone_of_t1 AS SELECT * FROM t1;
+-------------------+
| summary           |
+-------------------+
| Inserted 3 row(s) |
+-------------------+

-- Clone the columns and data, and convert the data to a different file form
at.
CREATE TABLE parquet_version_of_t1 STORED AS PARQUET AS SELECT * FROM t1;
+-------------------+
| summary           |
+-------------------+
| Inserted 3 row(s) |
+-------------------+
-- Copy only some rows to the new table.
CREATE TABLE subset_of_t1 AS SELECT * FROM t1 WHERE x >= 2;
+-------------------+
| summary           |
+-------------------+
| Inserted 2 row(s) |
+-------------------+
-- Same idea as CREATE TABLE LIKE: clone table layout but do not copy any d
ata.
CREATE TABLE empty_clone_of_t1 AS SELECT * FROM t1 WHERE 1=0;
+-------------------+
| summary           |
+-------------------+
| Inserted 0 row(s) |
+-------------------+

-- Reorder and rename columns and transform the data.
CREATE TABLE t5 AS SELECT upper(y) AS s, x+1 AS a, 'Entirely new column' AS
n FROM t1;
+-------------------+
| summary           |
+-------------------+
| Inserted 3 row(s) |
+-------------------+
SELECT * FROM t5;
+-------+---+---------------------+
| s     | a | n                   |
+-------+---+---------------------+
| ONE   | 2 | Entirely new column |
| TWO   | 3 | Entirely new column |
| THREE | 4 | Entirely new column |
+-------+---+---------------------+
```

See the SELECT statement on page 206 topic for details about query syntax for the SELECT portion of a CREATE TABLE AS SELECT statement.

The newly created table inherits the column names that you select from the original table, which you can override by specifying column aliases in the query. Any column or table comments from the original table are not carried over to the new table.

Note: When using the STORED AS clause with a CREATE TABLE AS SELECT statement, the destination table must be a file format that Impala can write to: currently, text or Parquet. You cannot specify an Avro, SequenceFile, or RCFile table as the destination table for a CTAS operation.

Prior to Impala 2.5 you could use a partitioned table as the source and copy data from it, but could not specify any partitioning clauses for the new table. In Impala 2.5 and higher, you can now use the PARTITIONED BY clause with a CREATE TABLE AS SELECT statement. The following example demonstrates how you can copy data from an unpartitioned table in a CREATE TABLE AS SELECT operation, creating a new partitioned table in the process. The main syntax consideration is the column order in the PARTITIONED BY clause and the select list: the partition key columns must be listed last in the select list, in the same order as in the PARTITIONED BY clause. Therefore, in this case, the column order in the destination table is different from the source table. You also only specify the column names in the PARTITIONED BY clause, not the data types or column comments.

```
create table partitions_no (year smallint, month tinyint, s string);
insert into partitions_no values (2016, 1, 'January 2016'),
  (2016, 2, 'February 2016'), (2016, 3, 'March 2016');

-- Prove that the source table is not partitioned.
show partitions partitions_no;
ERROR: AnalysisException: Table is not partitioned: ctas_partition_by.partit
ions_no
-- Create new table with partitions based on column values from source tabl
e.
create table partitions_yes partitioned by (year, month)
  as select s, year, month from partitions_no;
+--------------------+
| summary            |
+--------------------+
| Inserted 3 row(s) |
+--------------------+

-- Prove that the destination table is partitioned.
show partitions partitions_yes;
+-------+-------+-------+--------+------+...
| year  | month | #Rows | #Files | Size |...
+-------+-------+-------+--------+------+...
| 2016  | 1     | -1    | 1      | 13B  |...
| 2016  | 2     | -1    | 1      | 14B  |...
| 2016  | 3     | -1    | 1      | 11B  |...
| Total |       | -1    | 3      | 38B  |...
+-------+-------+-------+--------+------+...
```

The most convenient layout for partitioned tables is with all the partition key columns at the end. The CTAS PARTITIONED BY syntax requires that column order in the select list, resulting in that same column order in the destination table.

```
describe partitions_no;
+-------+----------+---------+
| name  | type     | comment |
+-------+----------+---------+
| year  | smallint |         |
| month | tinyint  |         |
| s     | string   |         |
+-------+----------+---------+

-- The CTAS operation forced us to put the partition key columns last.
-- Having those columns last works better with idioms such as SELECT *
-- for partitioned tables.
describe partitions_yes;
+-------+----------+---------+
| name  | type     | comment |
+-------+----------+---------+
| s     | string   |         |
| year  | smallint |         |
```

```
| month | tinyint  |         |
+-------+----------+---------+
```

Attempting to use a select list with the partition key columns not at the end results in an error due to a column name mismatch:

```
-- We expect this CTAS to fail because non-key column S
-- comes after key columns YEAR and MONTH in the select list.
create table partitions_maybe partitioned by (year, month)
  as select year, month, s from partitions_no;
ERROR: AnalysisException: Partition column name mismatch: year != month
```

As part of a CTAS operation, you can convert the data to any file format that Impala can write (currently, TEXTFILE and PARQUET). You cannot specify the lower-level properties of a text table, such as the delimiter.

Sorting considerations: Although you can specify an ORDER BY clause in an INSERT ... SELECT statement, any ORDER BY clause is ignored and the results are not necessarily sorted. An INSERT ... SELECT operation potentially creates many different data files, prepared by different executor Impala daemons, and therefore the notion of the data being stored in sorted order is impractical.

CREATE TABLE LIKE PARQUET:

The variation CREATE TABLE ... LIKE PARQUET '*HDFS_PATH_OF_PARQUET_FILE*' lets you skip the column definitions of the CREATE TABLE statement. The column names and data types are automatically configured based on the organization of the specified Parquet data file, which must already reside in HDFS. You can use a data file located outside the Impala database directories, or a file from an existing Impala Parquet table; either way, Impala only uses the column definitions from the file and does not use the HDFS location for the LOCATION attribute of the new table. (Although you can also specify the enclosing directory with the LOCATION attribute, to both use the same schema as the data file and point the Impala table at the associated directory for querying.)

The following considerations apply when you use the CREATE TABLE LIKE PARQUET technique:

- Any column comments from the original table are not preserved in the new table. Each column in the new table has a comment stating the low-level Parquet field type used to deduce the appropriate SQL column type.
- If you use a data file from a partitioned Impala table, any partition key columns from the original table are left out of the new table, because they are represented in HDFS directory names rather than stored in the data file. To preserve the partition information, repeat the same PARTITION clause as in the original CREATE TABLE statement.
- The file format of the new table defaults to text, as with other kinds of CREATE TABLE statements. To make the new table also use Parquet format, include the clause STORED AS PARQUET in the CREATE TABLE LIKE PARQUET statement.
- If the Parquet data file comes from an existing Impala table, currently, any TINYINT or SMALLINT columns are turned into INT columns in the new table. Internally, Parquet stores such values as 32-bit integers.
- When the destination table uses the Parquet file format, the CREATE TABLE AS SELECT and INSERT ... SELECT statements always create at least one data file, even if the SELECT part of the statement does not match any rows. You can use such an empty Parquet data file as a template for subsequent CREATE TABLE LIKE PARQUET statements.

For more details about creating Parquet tables, and examples of the CREATE TABLE LIKE PARQUET syntax, see *Using the Parquet file format with Impala tables*.

Visibility and Metadata (TBLPROPERTIES and WITH SERDEPROPERTIES clauses):

You can associate arbitrary items of metadata with a table by specifying the TBLPROPERTIES clause. This clause takes a comma-separated list of key-value pairs and stores those items in the metastore database. You can also change the table properties later with an ALTER TABLE statement. You can observe the table properties for different delimiter and escape characters using the DESCRIBE FORMATTED command, and change those settings for an existing table with ALTER TABLE ... SET TBLPROPERTIES.

You can also associate SerDes properties with the table by specifying key-value pairs through the WITH SERDEPROPERTIES clause. This metadata is not used by Impala, which has its own built-in serializer and deserializer for the file formats it supports. Particular property values might be needed for Hive compatibility with certain variations of file formats, particularly Avro.

Some DDL operations that interact with other Hadoop components require specifying particular values in the SERDEPROPERTIES or TBLPROPERTIES fields, such as creating an Avro table or an HBase table. (You typically create HBase tables in Hive, because they require additional clauses not currently available in Impala.)

To see the column definitions and column comments for an existing table, for example before issuing a CREATE TABLE ... LIKE or a CREATE TABLE ... AS SELECT statement, issue the statement DESCRIBE *TABLE_NAME*. To see even more detail, such as the location of data files and the values for clauses such as ROW FORMAT and STORED AS, issue the statement DESCRIBE FORMATTED *TABLE_NAME*. DESCRIBE FORMATTED is also needed to see any overall table comment (as opposed to individual column comments).

After creating a table, your `impala-shell` session or another `impala-shell` connected to the same node can immediately query that table. There might be a brief interval (one statestore heartbeat) before the table can be queried through a different Impala node. To make the CREATE TABLE statement return only when the table is recognized by all Impala nodes in the cluster, enable the SYNC_DDL query option.

HDFS caching (CACHED IN clause):

If you specify the CACHED IN clause, any existing or future data files in the table directory or the partition subdirectories are designated to be loaded into memory with the HDFS caching mechanism.

In Impala 2.2 and higher, the optional WITH REPLICATION clause for CREATE TABLE and ALTER TABLE lets you specify a *replication factor*, the number of hosts on which to cache the same data blocks. When Impala processes a cached data block, where the cache replication factor is greater than 1, Impala randomly selects a host that has a cached copy of that data block. This optimization avoids excessive CPU usage on a single host when the same cached data block is processed multiple times. Cloudera recommends specifying a value greater than or equal to the HDFS block replication factor.

Column order:

If you intend to use the table to hold data files produced by some external source, specify the columns in the same order as they appear in the data files.

If you intend to insert or copy data into the table through Impala, or if you have control over the way externally produced data files are arranged, use your judgment to specify columns in the most convenient order:

- If certain columns are often NULL, specify those columns last. You might produce data files that omit these trailing columns entirely. Impala automatically fills in the NULL values if so.
- If an unpartitioned table will be used as the source for an INSERT ... SELECT operation into a partitioned table, specify last in the unpartitioned table any columns that correspond to partition key columns in the partitioned table, and in the same order as the partition key columns are declared in the partitioned table. This technique lets you use INSERT ... SELECT * when copying data to the partitioned table, rather than specifying each column name individually.
- If you specify columns in an order that you later discover is suboptimal, you can sometimes work around the problem without recreating the table. You can create a view that selects columns from the original table in a permuted order, then do a SELECT * from the view. When inserting data into a table, you can specify a permuted order for the inserted columns to match the order in the destination table.

Hive considerations:

Impala queries can make use of metadata about the table and columns, such as the number of rows in a table or the number of different values in a column. Prior to Impala 1.2.2, to create this metadata, you issued the ANALYZE TABLE statement in Hive to gather this information, after creating the table and loading representative data into it. In Impala 1.2.2 and higher, the COMPUTE STATS statement produces these statistics within Impala, without needing to use Hive at all.

HBase considerations:

Note:

The Impala CREATE TABLE statement cannot create an HBase table, because it currently does not support the STORED BY clause needed for HBase tables. Create such tables in Hive, then query them through Impala.

Amazon S3 considerations:

To create a table where the data resides in the Amazon Simple Storage Service (S3), specify a s3a:// prefix LOCATION attribute pointing to the data files in S3.

In Impala 2.6 and higher, you can use this special LOCATION syntax as part of a CREATE TABLE AS SELECT statement.

In Impala 2.6 and higher, Impala DDL statements such as CREATE DATABASE, CREATE TABLE, DROP DATABASE CASCADE, DROP TABLE, and ALTER TABLE [ADD|DROP] PARTITION can create or remove folders as needed in the Amazon S3 system. Prior to Impala 2.6, you had to create folders yourself and point Impala database, tables, or partitions at them, and manually remove folders when no longer needed.

Sorting considerations: Although you can specify an ORDER BY clause in an INSERT ... SELECT statement, any ORDER BY clause is ignored and the results are not necessarily sorted. An INSERT ... SELECT operation potentially creates many different data files, prepared by different executor Impala daemons, and therefore the notion of the data being stored in sorted order is impractical.

HDFS considerations:

The CREATE TABLE statement for an internal table creates a directory in HDFS. The CREATE EXTERNAL TABLE statement associates the table with an existing HDFS directory, and does not create any new directory in HDFS. To locate the HDFS data directory for a table, issue a DESCRIBE FORMATTED *TABLE* statement. To examine the contents of that HDFS directory, use an OS command such as hdfs dfs -ls hdfs://*PATH*, either from the OS command line or through the shell or `!` commands in `impala-shell`.

The CREATE TABLE AS SELECT syntax creates data files under the table data directory to hold any data copied by the INSERT portion of the statement. (Even if no data is copied, Impala might create one or more empty data files.)

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have both execute and write permission for the database directory where the table is being created.

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Cancellation: Certain multi-stage statements (CREATE TABLE AS SELECT and COMPUTE STATS) can be cancelled during some stages, when running INSERT or SELECT operations internally. To cancel this statement, use Ctrl-C from the `impala-shell` interpreter, the Cancel button from the Watch page in Hue, Actions > Cancel from the Queries list in Cloudera Manager, or Cancel from the list of in-flight queries (for a particular node) on the Queries tab in the Impala web UI (port 25000).

Related Information

Impala tables

Partitioning

Hadoop file formats supported

SQL transactions in Impala

CREATE VIEW statement

The CREATE VIEW statement lets you create a shorthand abbreviation for a more complicated query. The base query can involve joins, expressions, reordered columns, column aliases, and other SQL features that can make a query hard to understand or maintain.

Because a view is purely a logical construct (an alias for a query) with no physical data behind it, ALTER VIEW only involves changes to metadata in the metastore database, not any data files in HDFS.

Syntax:

```
CREATE VIEW [IF NOT EXISTS] VIEW_NAME
    [(COLUMN_NAME [COMMENT 'COLUMN_COMMENT'][, ...])]
    [COMMENT 'VIEW_COMMENT']
    [TBLPROPERTIES ('NAME' = 'VALUE'[, ...])]
  AS SELECT_STATEMENT
```

Statement type: DDL

Usage notes:

The CREATE VIEW statement can be useful in scenarios such as the following:

- To turn even the most lengthy and complicated SQL query into a one-liner. You can issue simple queries against the view from applications, scripts, or interactive queries in `impala-shell`. For example:

  ```
  select * from VIEW_NAME;
  select * from VIEW_NAME order by c1 desc limit 10;
  ```

  The more complicated and hard-to-read the original query, the more benefit there is to simplifying the query using a view.
- To hide the underlying table and column names, to minimize maintenance problems if those names change. In that case, you re-create the view using the new names, and all queries that use the view rather than the underlying tables keep running with no changes.
- To experiment with optimization techniques and make the optimized queries available to all applications. For example, if you find a combination of WHERE conditions, join order, join hints, and so on that works the best for a class of queries, you can establish a view that incorporates the best-performing techniques. Applications can then make relatively simple queries against the view, without repeating the complicated and optimized logic over and over. If you later find a better way to optimize the original query, when you re-create the view, all the applications immediately take advantage of the optimized base query.
- To simplify a whole class of related queries, especially complicated queries involving joins between multiple tables, complicated expressions in the column list, and other SQL syntax that makes the query difficult to understand and debug. For example, you might create a view that joins several tables, filters using several WHERE conditions, and selects several columns from the result set. Applications might issue queries against this view that only vary in their LIMIT, ORDER BY, and similar simple clauses.

For queries that require repeating complicated clauses over and over again, for example in the select list, ORDER BY, and GROUP BY clauses, you can use the WITH clause as an alternative to creating a view.

You can optionally specify the table-level and the column-level comments as in the CREATE TABLE statement.

Complex type considerations:

For tables containing complex type columns (ARRAY, STRUCT, or MAP), you typically use join queries to refer to the complex values. You can use views to hide the join notation, making such tables seem like traditional denormalized tables, and making those tables queryable by business intelligence tools that do not have built-in support for those complex types.

Because you cannot directly issue SELECT *COL_NAME* against a column of complex type, you cannot use a view or a WITH clause to “rename” a column by selecting it with a column alias.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Examples:

```
-- Create a view that is exactly the same as the underlying table.
CREATE VIEW v1 AS SELECT * FROM t1;

-- Create a view that includes only certain columns from the underlying tabl
e.
CREATE VIEW v2 AS SELECT c1, c3, c7 FROM t1;
-- Create a view that filters the values from the underlying table.
CREATE VIEW v3 AS SELECT DISTINCT c1, c3, c7 FROM t1 WHERE c1 IS NOT NULL A
ND c5 > 0;

-- Create a view that that reorders and renames columns from the underlying
table.
CREATE VIEW v4 AS SELECT c4 AS last_name, c6 AS address, c2 AS birth_date
 FROM t1;

-- Create a view that runs functions to convert or transform certain colu
mns.
CREATE VIEW v5 AS SELECT c1, CAST(c3 AS STRING) c3, CONCAT(c4,c5) c5, TRIM(
c6) c6, "Constant" c8 FROM t1;

-- Create a view that hides the complexity of a view query.
CREATE VIEW v6 AS SELECT t1.c1, t2.c2 FROM t1 JOIN t2 ON t1.id = t2.id;
-- Create a view with a column comment and a table comment.
CREATE VIEW v7 (c1 COMMENT 'Comment for c1', c2) COMMENT 'Comment for v7' AS
 SELECT t1.c1, t1.c2 FROM t1;

-- Create a view with tblproperties.
CREATE VIEW v7 (c1 , c2) TBLPROPERTIES ('tblp1' = '1', 'tblp2' = '2') AS SE
LECT t1.c1, t1.c2 FROM t1;
```

DELETE statement

The DELETE statement deletes an arbitrary number of rows from a Kudu table. This statement only works for Impala tables that use the Kudu storage engine.

Syntax:

```
DELETE [FROM] [DATABASE_NAME.]TABLE_NAME [ WHERE WHERE_CONDITIONS ]

DELETE TABLE_REF FROM [JOINED_TABLE_REFS] [ WHERE WHERE_CONDITIONS ]
```

The first form evaluates rows from one table against an optional WHERE clause, and deletes all the rows that match the WHERE conditions, or all rows if WHERE is omitted.

The second form evaluates one or more join clauses, and deletes all matching rows from one of the tables. The join clauses can include non-Kudu tables, but the table from which the rows are deleted must be a Kudu table. The FROM keyword is required in this case, to separate the name of the table whose rows are being deleted from the table names of the join clauses.

Usage notes:

The conditions in the WHERE clause are the same ones allowed for the SELECT statement.

The conditions in the WHERE clause can refer to any combination of primary key columns or other columns. Referring to primary key columns in the WHERE clause is more efficient than referring to non-primary key columns.

If the WHERE clause is omitted, all rows are removed from the table.

Because Kudu currently does not enforce strong consistency during concurrent DML operations, be aware that the results after this statement finishes might be different than you intuitively expect:

- If some rows cannot be deleted because their some primary key columns are not found, due to their being deleted by a concurrent DELETE operation, the statement succeeds but returns a warning.
- A DELETE statement might also overlap with INSERT, UPDATE, or UPSERT statements running concurrently on the same table. After the statement finishes, there might be more or fewer rows than expected in the table because it is undefined whether the DELETE applies to rows that are inserted or updated while the DELETE is in progress.

The number of affected rows is reported in an `impala-shell` message and in the query profile.

Statement type: DML

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMPUTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

The following examples show how to delete rows from a specified table, either all rows or rows that match a WHERE clause:

```
-- Deletes all rows. The FROM keyword is optional.
DELETE FROM kudu_table;
DELETE kudu_table;

-- Deletes 0, 1, or more rows.
-- (If c1 is a single-column primary key, the statement could only
-- delete 0 or 1 rows.)
DELETE FROM kudu_table WHERE c1 = 100;

-- Deletes all rows that match all the WHERE conditions.
DELETE FROM kudu_table WHERE
  (c1 > c2 OR c3 IN ('hello','world')) AND c4 IS NOT NULL;
DELETE FROM t1 WHERE
  (c1 IN (1,2,3) AND c2 > c3) OR c4 IS NOT NULL;
DELETE FROM time_series WHERE
  year = 2016 AND month IN (11,12) AND day > 15;
-- WHERE condition with a subquery.
DELETE FROM t1 WHERE
  c5 IN (SELECT DISTINCT other_col FROM other_table);
-- Does not delete any rows, because the WHERE condition is always false.
DELETE FROM kudu_table WHERE 1 = 0;
```

The following examples show how to delete rows that are part of the result set from a join:

```
-- Remove _all_ rows from t1 that have a matching X value in t2.
DELETE t1 FROM t1 JOIN t2 ON t1.x = t2.x;
-- Remove _some_ rows from t1 that have a matching X value in t2.
DELETE t1 FROM t1 JOIN t2 ON t1.x = t2.x
  WHERE t1.y = FALSE and t2.z > 100;

-- Delete from a Kudu table based on a join with a non-Kudu table.
DELETE t1 FROM kudu_table t1 JOIN non_kudu_table t2 ON t1.x = t2.x;

-- The tables can be joined in any order as long as the Kudu table
-- is specified as the deletion target.
DELETE t2 FROM non_kudu_table t1 JOIN kudu_table t2 ON t1.x = t2.x;
```

Related Information

SELECT statement

DESCRIBE statement

The DESCRIBE statement displays metadata about a table, such as the column names and their data types. In Impala 2.3 and higher, you can specify the name of a complex type column, which takes the form of a dotted path. The path might include multiple components in the case of a nested type definition. In Impala 2.5 and higher, the DESCRIBE DATABASE form can display information about a database.

Syntax:

```
DESCRIBE [DATABASE] [FORMATTED|EXTENDED] OBJECT_NAME

object_name ::=
    [DB_NAME.]TABLE_NAME[.COMPLEX_COL_NAME ...]
  | DB_NAME
```

You can use the abbreviation DESC for the DESCRIBE statement.

The DESCRIBE FORMATTED variation displays additional information, in a format familiar to users of Apache Hive. The extra information includes low-level details such as whether the table is internal or external, when it was created, the file format, the location of the data in HDFS, whether the object is a table or a view, and (for views) the text of the query from the view definition.

Note: The Compressed field is not a reliable indicator of whether the table contains compressed data. It typically always shows No, because the compression settings only apply during the session that loads data and are not stored persistently with the table metadata.

Describing databases:

By default, the DESCRIBE output for a database includes the location and the comment, which can be set by the LOCATION and COMMENT clauses on the CREATE DATABASE statement.

The additional information displayed by the FORMATTED or EXTENDED keyword includes the HDFS user ID that is considered the owner of the database, and any optional database properties. The properties could be specified by the WITH DBPROPERTIES clause if the database is created using a Hive CREATE DATABASE statement. Impala currently does not set or do any special processing based on those properties.

The following examples show the variations in syntax and output for describing databases. This feature is available in Impala 2.5 and higher.

```
describe database default;
+---------+----------------------+-----------------------+
```

```
| name    | location             | comment               |
+---------+----------------------+-----------------------+
| default | /user/hive/warehouse | Default Hive database |
+---------+----------------------+-----------------------+

describe database formatted default;
+---------+----------------------+-----------------------+
| name    | location             | comment               |
+---------+----------------------+-----------------------+
| default | /user/hive/warehouse | Default Hive database |
| Owner:  |                      |                       |
|         | public               | ROLE                  |
+---------+----------------------+-----------------------+

describe database extended default;
+---------+----------------------+-----------------------+
| name    | location             | comment               |
+---------+----------------------+-----------------------+
| default | /user/hive/warehouse | Default Hive database |
| Owner:  |                      |                       |
|         | public               | ROLE                  |
+---------+----------------------+-----------------------+
```

Describing tables:

If the DATABASE keyword is omitted, the default for the DESCRIBE statement is to refer to a table.

If you have the SELECT privilege on a subset of the table columns and no other relevant table/database/server-level privileges, DESCRIBE returns the data from the columns you have access to.

If you have the SELECT privilege on a subset of the table columns and no other relevant table/database/server-level privileges, DESCRIBE FORMATTED/EXTENDED does not return the LOCATION field. The LOCATION data is shown if you have any privilege on the table, the containing database or the server.

```
-- By default, the table is assumed to be in the current database.
describe my_table;
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| x    | int    |         |
| s    | string |         |
+------+--------+---------+

-- Use a fully qualified table name to specify a table in any database.
describe my_database.my_table;
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| x    | int    |         |
| s    | string |         |
+------+--------+---------+

-- The formatted or extended output includes additional useful information.
-- The LOCATION field is especially useful to know for DDL statements and HD
FS commands
-- during ETL jobs. (The LOCATION includes a full hdfs:// URL, omitted here
for readability.)
describe formatted my_table;
+------------------------------+----------------------------------------
-----+----------------------+
| name                         | type
  | comment              |
```

```
+------------------------------+-----------------------------------------
---+----------------------+
| # col_name                   | data_type
  | comment              |
|                              | NULL
   | NULL                 |
| x                            | int
  | NULL                 |
| s                            | string
  | NULL                 |
|                              | NULL
  | NULL                 |
| # Detailed Table Information | NULL
   | NULL                 |
| Database:                    | my_database
  | NULL                 |
| Owner:                       | jrussell
  | NULL                 |
| CreateTime:                  | Fri Mar 18 15:58:00 PDT 2016
  | NULL                 |
| LastAccessTime:              | UNKNOWN
   | NULL                 |
| Protect Mode:                | None
  | NULL                 |
| Retention:                   | 0
  | NULL                 |
| Location:                    | /user/hive/warehouse/my_database.db/my_tabl
e | NULL                 |
| Table Type:                  | MANAGED_TABLE
   | NULL                 |
| Table Parameters:            | NULL
  | NULL                 |
|                              | transient_lastDdlTime
  | 1458341880           |
|                              | NULL
  | NULL                 |
| # Storage Information        | NULL
   | NULL                 |
| SerDe Library:               | org. ... .LazySimpleSerDe
  | NULL                 |
| InputFormat:                 | org.apache.hadoop.mapred.TextInputFormat
  | NULL                 |
| OutputFormat:                | org. ... .HiveIgnoreKeyTextOutputFormat
  | NULL                 |
| Compressed:                  | No
   | NULL                 |
| Num Buckets:                 | 0
  | NULL                 |
| Bucket Columns:              | []
  | NULL                 |
| Sort Columns:                | []
  | NULL                 |
+------------------------------+----------------------------------------
-----+----------------------+
```

Complex type considerations:

Because the column definitions for complex types can become long, particularly when such types are nested, the DESCRIBE statement uses special formatting for complex type columns to make the output readable.

For the ARRAY, STRUCT, and MAP types available in Impala 2.3 and higher, the DESCRIBE output is formatted to avoid excessively long lines for multiple fields within a STRUCT, or a nested sequence of complex types.

You can pass a multi-part qualified name to DESCRIBE to specify an ARRAY, STRUCT, or MAP column and visualize its structure as if it were a table. For example, if table T1 contains an ARRAY column A1, you could issue the statement DESCRIBE t1.a1. If table T1 contained a STRUCT column S1, and a field F1 within the STRUCT was a MAP, you could issue the statement DESCRIBE t1.s1.f1. An ARRAY is shown as a two-column table, with ITEM and POS columns. A STRUCT is shown as a table with each field representing a column in the table. A MAP is shown as a two-column table, with KEY and VALUE columns.

For example, here is the DESCRIBE output for a table containing a single top-level column of each complex type:

```
create table t1 (x int, a array<int>, s struct<f1: string, f2: bigint>, m ma
p<string,int>) stored as parquet;

describe t1;
+------+-----------------+---------+
| name | type            | comment |
+------+-----------------+---------+
| x    | int             |         |
| a    | array<int>      |         |
| s    | struct<         |         |
|      |   f1:string,    |         |
|      |   f2:bigint     |         |
|      | >               |         |
| m    | map<string,int> |         |
+------+-----------------+---------+
```

Here are examples showing how to “drill down” into the layouts of complex types, including using multi-part names to examine the definitions of nested types. The < > delimiters identify the columns with complex types; these are the columns where you can descend another level to see the parts that make up the complex type. This technique helps you to understand the multi-part names you use as table references in queries involving complex types, and the corresponding column names you refer to in the SELECT list. These tables are from the “nested TPC-H” schema, shown in detail in *Sample schema and data for experimenting with Impala complex types*.

The REGION table contains an ARRAY of STRUCT elements:

- The first DESCRIBE specifies the table name, to display the definition of each top-level column.
- The second DESCRIBE specifies the name of a complex column, REGION.R_NATIONS, showing that when you include the name of an ARRAY column in a FROM clause, that table reference acts like a two-column table with columns ITEM and POS.
- The final DESCRIBE specifies the fully qualified name of the ITEM field, to display the layout of its underlying STRUCT type in table format, with the fields mapped to column names.

```
-- #1: The overall layout of the entire table.
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

-- #2: The ARRAY column within the table.
describe region.r_nations;
+------+------------------------+---------+
| name | type                   | comment |
```

```
+------+-------------------------+---------+
| item | struct<                 |         |
|      |   n_nationkey:smallint, |         |
|      |   n_name:string,        |         |
|      |   n_comment:string      |         |
|      | >                       |         |
| pos  | bigint                  |         |
+------+-------------------------+---------+

-- #3: The STRUCT that makes up each ARRAY element.
--     The fields of the STRUCT act like columns of a table.
describe region.r_nations.item;
+-------------+----------+---------+
| name        | type     | comment |
+-------------+----------+---------+
| n_nationkey | smallint |         |
| n_name      | string   |         |
| n_comment   | string   |         |
+-------------+----------+---------+
```

The CUSTOMER table contains an ARRAY of STRUCT elements, where one field in the STRUCT is another ARRAY of STRUCT elements:

- Again, the initial DESCRIBE specifies only the table name.
- The second DESCRIBE specifies the qualified name of the complex column, CUSTOMER.C_ORDERS, showing how an ARRAY is represented as a two-column table with columns ITEM and POS.
- The third DESCRIBE specifies the qualified name of the ITEM of the ARRAY column, to see the structure of the nested ARRAY. Again, it has has two parts, ITEM and POS. Because the ARRAY contains a STRUCT, the layout of the STRUCT is shown.
- The fourth and fifth DESCRIBE statements drill down into a STRUCT field that is itself a complex type, an ARRAY of STRUCT. The ITEM portion of the qualified name is only required when the ARRAY elements are anonymous. The fields of the STRUCT give names to any other complex types nested inside the STRUCT. Therefore, the DESCRIBE parameters CUSTOMER.C_ORDERS.ITEM.O_LINEITEMS and CUSTOMER.C_ORDERS.O_LINEITEMS are equivalent. (For brevity, leave out the ITEM portion of a qualified name when it is not required.)
- The final DESCRIBE shows the layout of the deeply nested STRUCT type. Because there are no more complex types nested inside this STRUCT, this is as far as you can drill down into the layout for this table.

```
-- #1: The overall layout of the entire table.
describe customer;
+--------------+-----------------------------------+
| name         | type                              |
+--------------+-----------------------------------+
| c_custkey    | bigint                            |
... more scalar columns ...
| c_orders     | array<struct<                     |
|              |   o_orderkey:bigint,              |
|              |   o_orderstatus:string,           |
|              |   o_totalprice:decimal(12,2),     |
|              |   o_orderdate:string,             |
|              |   o_orderpriority:string,         |
|              |   o_clerk:string,                 |
|              |   o_shippriority:int,             |
|              |   o_comment:string,               |
|              |   o_lineitems:array<struct<       |
|              |     l_partkey:bigint,             |
|              |     l_suppkey:bigint,             |
|              |     l_linenumber:int,             |
|              |     l_quantity:decimal(12,2),     |
|              |     l_extendedprice:decimal(12,2), |
```

```
|              |     l_discount:decimal(12,2),      |
|              |     l_tax:decimal(12,2),           |
|              |     l_returnflag:string,           |
|              |     l_linestatus:string,           |
|              |     l_shipdate:string,             |
|              |     l_commitdate:string,           |
|              |     l_receiptdate:string,          |
|              |     l_shipinstruct:string,         |
|              |     l_shipmode:string,             |
|              |     l_comment:string               |
|              |   >>                               |
|              | >>                                 |
+--------------+------------------------------------+

-- #2: The ARRAY column within the table.
describe customer.c_orders;
+------+-----------------------------------+
| name | type                              |
+------+-----------------------------------+
| item | struct<                           |
|      |   o_orderkey:bigint,              |
|      |   o_orderstatus:string,           |
... more struct fields ...
|      |   o_lineitems:array<struct<       |
|      |     l_partkey:bigint,             |
|      |     l_suppkey:bigint,             |
... more nested struct fields ...
|      |     l_comment:string              |
|      |   >>                              |
|      | >                                 |
| pos  | bigint                            |
+------+-----------------------------------+

-- #3: The STRUCT that makes up each ARRAY element.
--     The fields of the STRUCT act like columns of a table.
describe customer.c_orders.item;
+-----------------+---------------------------------+
| name            | type                            |
+-----------------+---------------------------------+
| o_orderkey      | bigint                          |
| o_orderstatus   | string                          |
| o_totalprice    | decimal(12,2)                   |
| o_orderdate     | string                          |
| o_orderpriority | string                          |
| o_clerk         | string                          |
| o_shippriority  | int                             |
| o_comment       | string                          |
| o_lineitems     | array<struct<                   |
|                 |   l_partkey:bigint,             |
|                 |   l_suppkey:bigint,             |
... more struct fields ...
|                 |   l_comment:string              |
|                 | >>                              |
+-----------------+---------------------------------+

-- #4: The ARRAY nested inside the STRUCT elements of the first ARRAY.
describe customer.c_orders.item.o_lineitems;
+------+---------------------------------+
| name | type                            |
+------+---------------------------------+
| item | struct<                         |
|      |   l_partkey:bigint,             |
|      |   l_suppkey:bigint,             |
... more struct fields ...
```

```
|      |   l_comment:string                |
|      | >                                 |
| pos  | bigint                            |
+------+-----------------------------------+

-- #5: Shorter form of the previous DESCRIBE. Omits the .ITEM portion of the
 name
--     because O_LINEITEMS and other field names provide a way to refer to
 things
--     inside the ARRAY element.
describe customer.c_orders.o_lineitems;
+------+-----------------------------------+
| name | type                              |
+------+-----------------------------------+
| item | struct<                           |
|      |   l_partkey:bigint,               |
|      |   l_suppkey:bigint,               |
... more struct fields ...
|      |   l_comment:string                |
|      | >                                 |
| pos  | bigint                            |
+------+-----------------------------------+
-- #6: The STRUCT representing ARRAY elements nested inside
--     another ARRAY of STRUCTs. The lack of any complex types
--     in this output means this is as far as DESCRIBE can
--     descend into the table layout.
describe customer.c_orders.o_lineitems.item;
+-----------------+---------------+
| name            | type          |
+-----------------+---------------+
| l_partkey       | bigint        |
| l_suppkey       | bigint        |
... more scalar columns ...
| l_comment       | string        |
+-----------------+---------------+
```

Usage notes:

After the `impalad` daemons are restarted, the first query against a table can take longer than subsequent queries, because the metadata for the table is loaded before the query is processed. This one-time delay for each table can cause misleading results in benchmark tests or cause unnecessary concern. To “warm up” the Impala metadata cache, you can issue a DESCRIBE statement in advance for each table you intend to access later.

When you are dealing with data files stored in HDFS, sometimes it is important to know details such as the path of the data files for an Impala table, and the hostname for the namenode. You can get this information from the DESC RIBE FORMATTED output. You specify HDFS URIs or path specifications with statements such as LOAD DATA and the LOCATION clause of CREATE TABLE or ALTER TABLE. You might also use HDFS URIs or paths with Linux commands such as `hadoop` and `hdfs` to copy, rename, and so on, data files in HDFS.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Each table can also have associated table statistics and column statistics. To see these categories of information, use the SHOW TABLE STATS *TABLE_NAME* and SHOW COLUMN STATS *TABLE_NAME* statements. See the *SHOW statement* topic for details.

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMP UTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

The following example shows the results of both a standard DESCRIBE and DESCRIBE FORMATTED for different kinds of schema objects:

- DESCRIBE for a table or a view returns the name, type, and comment for each of the columns. For a view, if the column value is computed by an expression, the column name is automatically generated as _c0, _c1, and so on depending on the ordinal number of the column.
- A table created with no special format or storage clauses is designated as a MANAGED_TABLE (an “internal table” in Impala terminology). Its data files are stored in an HDFS directory under the default Hive data directory. By default, it uses Text data format.
- A view is designated as VIRTUAL_VIEW in DESCRIBE FORMATTED output. Some of its properties are NULL or blank because they are inherited from the base table. The text of the query that defines the view is part of the DESCRIBE FORMATTED output.
- A table with additional clauses in the CREATE TABLE statement has differences in DESCRIBE FORMATTED output. The output for T2 includes the EXTERNAL_TABLE keyword because of the CREATE EXTERNAL TABLE syntax, and different InputFormat and OutputFormat fields to reflect the Parquet file format.

```
[localhost:21000] > create table t1 (x int, y int, s string);
Query: create table t1 (x int, y int, s string)
[localhost:21000] > describe t1;
Query: describe t1
Query finished, fetching results ...
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| x    | int    |         |
| y    | int    |         |
| s    | string |         |
+------+--------+---------+
Returned 3 row(s) in 0.13s
[localhost:21000] > describe formatted t1;
Query: describe formatted t1
Query finished, fetching results ...
+------------------------------+-------------------------------------------
+------------+
| name                         | type
 | comment    |
+------------------------------+-------------------------------------------
+------------+
| # col_name                   | data_type
| comment    |
|                              | NULL
 | NULL       |
| x                            | int
 | None       |
| y                            | int
 | None       |
| s                            | string
| None       |
|                              | NULL
 | NULL       |
| # Detailed Table Information | NULL
 | NULL       |
| Database:                    | describe_formatted
 | NULL       |
| Owner:                       | doc_demo
| NULL       |
| CreateTime:                  | Mon Jul 22 17:03:16 EDT 2013
 | NULL       |
| LastAccessTime:              | UNKNOWN
 | NULL       |
```

```
| Protect Mode:                | None
 | NULL       |
| Retention:                   | 0
| NULL       |
| Location:                    | hdfs://127.0.0.1:8020/user/hive/warehouse/
 |            |
|                              |   describe_formatted.db/t1
 | NULL       |
| Table Type:                  | MANAGED_TABLE
 | NULL       |
| Table Parameters:            | NULL
| NULL       |
|                              | transient_lastDdlTime
 | 1374526996 |
|                              | NULL
 | NULL       |
| # Storage Information        | NULL
 | NULL       |
| SerDe Library:               | org.apache.hadoop.hive.serde2.lazy.
|            |
|                              |   LazySimpleSerDe
 | NULL       |
| InputFormat:                 | org.apache.hadoop.mapred.TextInputFormat
 | NULL       |
| OutputFormat:                | org.apache.hadoop.hive.ql.io.
 |            |
|                              |   HiveIgnoreKeyTextOutputFormat
| NULL       |
| Compressed:                  | No
 | NULL       |
| Num Buckets:                 | 0
 | NULL       |
| Bucket Columns:              | []
 | NULL       |
| Sort Columns:                | []
| NULL       |
+------------------------------+-------------------------------------------
+------------+
Returned 26 row(s) in 0.03s
[localhost:21000] > create view v1 as select x, upper(s) from t1;
Query: create view v1 as select x, upper(s) from t1
[localhost:21000] > describe v1;
Query: describe v1
Query finished, fetching results ...
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| x    | int    |         |
| _c1  | string |         |
+------+--------+---------+
Returned 2 row(s) in 0.10s
[localhost:21000] > describe formatted v1;
Query: describe formatted v1
Query finished, fetching results ...
+------------------------------+------------------------------+------------
----------+
| name                         | type                         | comment
        |
+------------------------------+------------------------------+----------
------------+
| # col_name                   | data_type                    | comment
        |
|                              | NULL                         | NULL
        |
```

```
| x                             | int                           | None
        |
| _c1                           | string                        | None
         |
|                               | NULL                          | NULL
        |
| # Detailed Table Information | NULL                          | NULL
        |
| Database:                     | describe_formatted            | NULL
        |
| Owner:                        | doc_demo                      | NULL
         |
| CreateTime:                   | Mon Jul 22 16:56:38 EDT 2013 | NULL
        |
| LastAccessTime:               | UNKNOWN                       | NULL
        |
| Protect Mode:                 | None                          | NULL
        |
| Retention:                    | 0                             | NULL
         |
| Table Type:                   | VIRTUAL_VIEW                  | NULL
        |
| Table Parameters:             | NULL                          | NULL
        |
|                               | transient_lastDdlTime         | 1374526598
        |
|                               | NULL                          | NULL
         |
| # Storage Information         | NULL                          | NULL
        |
| SerDe Library:                | null                          | NULL
        |
| InputFormat:                  | null                          | NULL
        |
| OutputFormat:                 | null                          | NULL
         |
| Compressed:                   | No                            | NULL
        |
| Num Buckets:                  | 0                             | NULL
        |
| Bucket Columns:               | []                            | NULL
        |
| Sort Columns:                 | []                            | NULL
         |
|                               | NULL                          | NULL
        |
| # View Information            | NULL                          | NULL
        |
| View Original Text:           | SELECT x, upper(s) FROM t1    | NULL
        |
| View Expanded Text:           | SELECT x, upper(s) FROM t1    | NULL
         |
+-------------------------------+-------------------------------+-----------
-----------+
Returned 28 row(s) in 0.03s
[localhost:21000] > create external table t2 (x int, y int, s string) stored
 as parquet location '/user/doc_demo/sample_data';
[localhost:21000] > describe formatted t2;
Query: describe formatted t2
Query finished, fetching results ...
+-------------------------------+-----------------------------------------
---------+------------+
| name                          | type
        | comment    |
```

```
+------------------------------+----------------------------------------
-----------+------------+
| # col_name                   | data_type
        | comment    |
|                              | NULL
        | NULL       |
| x                            | int
        | None       |
| y                            | int
         | None       |
| s                            | string
        | None       |
|                              | NULL
        | NULL       |
| # Detailed Table Information | NULL
        | NULL       |
| Database:                    | describe_formatted
         | NULL       |
| Owner:                       | doc_demo
        | NULL       |
| CreateTime:                  | Mon Jul 22 17:01:47 EDT 2013
        | NULL       |
| LastAccessTime:              | UNKNOWN
        | NULL       |
| Protect Mode:                | None
         | NULL       |
| Retention:                   | 0
        | NULL       |
| Location:                    | hdfs://127.0.0.1:8020/user/doc_demo/sample
_data    | NULL       |
| Table Type:                  | EXTERNAL_TABLE
        | NULL       |
| Table Parameters:            | NULL
         | NULL       |
|                              | EXTERNAL
        | TRUE       |
|                              | transient_lastDdlTime
        | 1374526907 |
|                              | NULL
        | NULL       |
| # Storage Information        | NULL
         | NULL       |
| SerDe Library:               | org.apache.hadoop.hive.serde2.lazy.LazySi
mpleSerDe | NULL       |
| InputFormat:                 | com.cloudera.impala.hive.serde.ParquetInpu
tFormat  | NULL       |
| OutputFormat:                | com.cloudera.impala.hive.serde.ParquetOutpu
tFormat | NULL       |
| Compressed:                  | No
         | NULL       |
| Num Buckets:                 | 0
        | NULL       |
| Bucket Columns:              | []
        | NULL       |
| Sort Columns:                | []
        | NULL       |
+------------------------------+----------------------------------------
-----------+------------+
Returned 27 row(s) in 0.17s
```

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all directories that are part of the table. (A table could span multiple different HDFS directories if it is partitioned. The directories could be widely scattered because a partition can reside in an arbitrary HDFS directory based on its LOCATION attribute.)

Cloudera Manager considerations:

A change in the behavior of metadata loading in Impala 2.9 could lead to certain long-running statements being left out of the Cloudera Manager list of Impala queries until the statements are completed.

Prior to Impala 2.9, statements such as DESCRIBE could cause the Impala web UI, and Cloudera Manager monitoring pages that rely on information from the web UI, to become unresponsive while they ran. The first time Impala references a large table, for example one with thousands of partitions, the statement might take longer than normal while the metadata for the table is loaded. In Impala 2.9 and higher, the Impala web UI and associated Cloudera Manager monitoring pages are more responsive while a metadata loading operation is in progress.

Although the statement that loads the metadata shows up on the Impala web UI /queries page immediately, it does not show up in the Cloudera Manager list of queries until the metadata is finished loading. For example, the first DESC RIBE of a large partitioned table might take 30 minutes due to metadata loading, and the statement does not show up in Cloudera Manager during those 30 minutes.

Kudu considerations:

The information displayed for Kudu tables includes the additional attributes that are only applicable for Kudu tables:

- Whether or not the column is part of the primary key. Every Kudu table has a true value here for at least one column. There could be multiple true values, for tables with composite primary keys.
- Whether or not the column is nullable. Specified by the NULL or NOT NULL attributes on the CREATE TABLE statement. Columns that are part of the primary key are automatically non-nullable.
- The default value, if any, for the column. Specified by the DEFAULT attribute on the CREATE TABLE statement. If the default value is NULL, that is not indicated in this column. It is implied by nullable being true and no other default value specified.
- The encoding used for values in the column. Specified by the ENCODING attribute on the CREATE TABLE statement.
- The compression used for values in the column. Specified by the COMPRESSION attribute on the CREATE T ABLE statement.
- The block size (in bytes) used for the underlying Kudu storage layer for the column. Specified by the BLOCK_SI ZE attribute on the CREATE TABLE statement.

The following example shows DESCRIBE output for a simple Kudu table, with a single-column primary key and all column attributes left with their default values:

```
describe million_rows;
+------+--------+---------+-------------+----------+---------------+--------
-------+---------------------+------------+
| name | type   | comment | primary_key | nullable | default_value | encodin
g      | compression         | block_size |
+------+--------+---------+-------------+----------+---------------+--------
-------+---------------------+------------+
| id   | string |         | true        | false    |               | AUTO_EN
CODING | DEFAULT_COMPRESSION | 0          |
| s    | string |         | false       | false    |               | AUTO_EN
CODING | DEFAULT_COMPRESSION | 0          |
+------+--------+---------+-------------+----------+---------------+--------
-------+---------------------+------------+
```

The following example shows DESCRIBE output for a Kudu table with a two-column primary key, and Kudu-specific attributes applied to some columns:

```
create table kudu_describe_example
(
  c1 int, c2 int,
  c3 string, c4 string not null, c5 string default 'n/a', c6 string default
'',
  c7 bigint not null, c8 bigint null default null, c9 bigint default -1 enco
ding bit_shuffle,
  primary key(c1,c2)
)
partition by hash (c1, c2) partitions 10 stored as kudu;

describe kudu_describe_example;
+------+--------+---------+-------------+----------+---------------+-----
----------+---------------------+------------+
| name | type   | comment | primary_key | nullable | default_value | enco
ding      | compression         | block_size |
+------+--------+---------+-------------+----------+---------------+-----
----------+---------------------+------------+
| c1   | int    |         | true        | false    |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c2   | int    |         | true        | false    |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c3   | string |         | false       | true     |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c4   | string |         | false       | false    |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c5   | string |         | false       | true     | n/a           | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c6   | string |         | false       | true     |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c7   | bigint |         | false       | false    |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c8   | bigint |         | false       | true     |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c9   | bigint |         | false       | true     | -1            | BIT_
SHUFFLE   | DEFAULT_COMPRESSION | 0          |
+------+--------+---------+-------------+----------+---------------+-----
----------+---------------------+------------+
```

Related Information

SHOW statement

DROP DATABASE statement

The DROP DATABASE statement removes a database from the system. The physical operations involve removing the metadata for the database from the metastore, and deleting the corresponding *.db directory from HDFS.

Syntax:

```
DROP (DATABASE|SCHEMA) [IF EXISTS] DATABASE_NAME [RESTRICT | CASCADE];
```

Statement type: DDL

Usage notes:

By default, the database must be empty before it can be dropped, to avoid losing any data.

In Impala 2.3 and higher, you can include the CASCADE clause to make Impala drop all tables and other objects in the database before dropping the database itself. The RESTRICT clause enforces the original requirement that the database be empty before being dropped. Because the RESTRICT behavior is still the default, this clause is optional.

The automatic dropping resulting from the CASCADE clause follows the same rules as the corresponding DROP TABLE, DROP VIEW, and DROP FUNCTION statements. In particular, the HDFS directories and data files for any external tables are left behind when the tables are removed.

When you do not use the CASCADE clause, drop or move all the objects inside the database manually before dropping the database itself:

- Use the SHOW TABLES statement to locate all tables and views in the database, and issue DROP TABLE and DROP VIEW statements to remove them all.
- Use the SHOW FUNCTIONS and SHOW AGGREGATE FUNCTIONS statements to locate all user-defined functions in the database, and issue DROP FUNCTION and DROP AGGREGATE FUNCTION statements to remove them all.
- To keep tables or views contained by a database while removing the database itself, use ALTER TABLE and ALTER VIEW to move the relevant objects to a different database before dropping the original database.

You cannot drop the current database, that is, the database your session connected to either through the USE statement or the -d option of `impala-shell`. Issue a USE statement to switch to a different database first. Because the default database is always available, issuing USE default is a convenient way to leave the current database before dropping it.

Hive considerations:

When you drop a database in Impala, the database can no longer be used by Hive.

Examples:

See *CREATE DATABASE* statement for examples covering CREATE DATABASE, USE, and DROP DATABASE.

Amazon S3 considerations:

In Impala 2.6 and higher, Impala DDL statements such as CREATE DATABASE, CREATE TABLE, DROP DATABASE CASCADE, DROP TABLE, and ALTER TABLE [ADD|DROP] PARTITION can create or remove folders as needed in the Amazon S3 system. Prior to Impala 2.6, you had to create folders yourself and point Impala database, tables, or partitions at them, and manually remove folders when no longer needed.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have write permission for the directory associated with the database.

Examples:

```
create database first_db;
use first_db;
create table t1 (x int);

create database second_db;
use second_db;
-- Each database has its own namespace for tables.
-- You can reuse the same table names in each database.
create table t1 (s string);

create database temp;

-- You can either USE a database after creating it,
-- or qualify all references to the table name with the name of the datab
ase.
-- Here, tables T2 and T3 are both created in the TEMP database.

create table temp.t2 (x int, y int);
use database temp;
create table t3 (s string);
```

```
-- You cannot drop a database while it is selected by the USE statement.
drop database temp;
ERROR: AnalysisException: Cannot drop current default database: temp

-- The always-available database 'default' is a convenient one to USE
-- before dropping a database you created.
use default;
-- Before dropping a database, first drop all the tables inside it,
-- or in Impala 2.3 and higher use the CASCADE clause.
drop database temp;
ERROR: ImpalaRuntimeException: Error making 'dropDatabase' RPC to Hive Meta
store:
CAUSED BY: InvalidOperationException: Database temp is not empty
show tables in temp;
+------+
| name |
+------+
| t3   |
+------+

-- Impala 2.3 and higher:
drop database temp cascade;

-- Earlier releases:
drop table temp.t3;
drop database temp;
```

Related Information

CREATE DATABASE statement

Impala with Amazon S3

DROP FUNCTION statement

The DROP FUNCTION statement removes a user-defined function (UDF), so that it is not available for execution during Impala SELECT or INSERT operations.

Syntax:

To drop C++ UDFs and UDAs:

```
DROP [AGGREGATE] FUNCTION [IF EXISTS]
 [DB_NAME.]FUNCTION_NAME(TYPE[, TYPE...])
```

Note:

The preceding syntax, which includes the function signature, also applies to Java UDFs that were created using the corresponding CREATE FUNCTION syntax that includes the argument and return types. After upgrading to Impala 2.5 or higher, consider re-creating all Java UDFs with the CREATE FUNCTION syntax that does not include the function signature. Java UDFs created this way are now persisted in the metastore database and do not need to be re-created after an Impala restart.

To drop Java UDFs (created using the CREATE FUNCTION syntax with no function signature):

```
DROP FUNCTION [IF EXISTS] [DB_NAME.]FUNCTION_NAME
```

Statement type: DDL

Usage notes:

Because the same function name could be overloaded with different argument signatures, you specify the argument types to identify the exact function to drop.

Restrictions:

In Impala 2.5 and higher, Impala UDFs and UDAs written in C++ are persisted in the metastore database. Java UDFs are also persisted, if they were created with the new CREATE FUNCTION syntax for Java UDFs, where the Java function argument and return types are omitted. Java-based UDFs created with the old CREATE FUNCTION syntax do not persist across restarts because they are held in the memory of the `catalogd` daemon. Until you re-create such Java UDFs using the new CREATE FUNCTION syntax, you must reload those Java-based UDFs by running the original CREATE FUNCTION statements again each time you restart the `catalogd` daemon. Prior to Impala 2.5 the requirement to reload functions after a restart applied to both C++ and Java functions.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, does not need any particular HDFS permissions to perform this statement. All read and write operations are on the metastore database, not HDFS files and directories.

Examples:

The following example shows how to drop Java functions created with the signatureless CREATE FUNCTION syntax in Impala 2.5 and higher. Issuing DROP FUNCTION *FUNCTION_NAME* removes all the overloaded functions under that name. See the *CREATE FUNCTION* statement for a longer example showing how to set up such functions in the first place.

```
create function my_func location '/user/impala/udfs/udf-examples-cdh570.jar'
  symbol='com.cloudera.impala.TestUdf';

show functions;
+-------------+-------------------------------------+-------------+----
-----------+
| return type | signature                           | binary type | is pe
rsistent |
+-------------+-------------------------------------+-------------+----
-----------+
| BIGINT      | my_func(BIGINT)                     | JAVA        | true
        |
| BOOLEAN     | my_func(BOOLEAN)                    | JAVA        | true
         |
| BOOLEAN     | my_func(BOOLEAN, BOOLEAN)           | JAVA        | true
        |
...
| BIGINT      | testudf(BIGINT)                     | JAVA        | true
         |
| BOOLEAN     | testudf(BOOLEAN)                    | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN)           | JAVA        | true
         |
...

drop function my_func;
show functions;
+-------------+-------------------------------------+-------------+------
---------+
| return type | signature                           | binary type | is
persistent |
+-------------+-------------------------------------+-------------+------
---------+
| BIGINT      | testudf(BIGINT)                     | JAVA        | true
         |
| BOOLEAN     | testudf(BOOLEAN)                    | JAVA        | true
        |
```

```
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN)             | JAVA        | true
         |
...
```

Related Information

CREATE FUNCTION statement

DROP ROLE statement

The DROP ROLE statement removes a role from the metastore database. Once dropped, the role is revoked for all groups it was previously assigned. Queries that are already executing are not affected. Impala verifies the role information approximately every 60 seconds, so the effects of DROP ROLE might not take effect for new Impala queries for a brief period.

Syntax:

```
DROP ROLE ROLE_NAME
```

Note: Before dropping a role in Ranger, you must remove all the privileges granted to the role in advance.

Required privileges:

Only administrative users can use this statement.

Compatibility:

Impala makes use of any roles and privileges specified by the GRANT and REVOKE statements in Hive, and Hive makes use of any roles and privileges specified by the GRANT and REVOKE statements in Impala. The Impala GRANT and REVOKE statements for privileges do not require the ROLE keyword to be repeated before each role name, unlike the equivalent Hive statements.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

DROP STATS statement

The DROP STATS statement removes the specified statistics from a table or partition. The statistics were originally created by the COMPUTE STATS or COMPUTE INCREMENTAL STATS statement.

Syntax:

```
DROP STATS [DATABASE_NAME.]TABLE_NAME
DROP INCREMENTAL STATS [DATABASE_NAME.]TABLE_NAME PARTITION (PARTITION_SPEC)

PARTITION_SPEC ::= PARTITION_COL=CONSTANT_VALUE
```

The PARTITION clause is only allowed in combination with the INCREMENTAL clause. It is optional for COMPUTE INCREMENTAL STATS, and required for DROP INCREMENTAL STATS. Whenever you specify partitions through the PARTITION (*PARTITION_SPEC*) clause in a COMPUTE INCREMENTAL STATS or DROP INCREMENTAL STATS statement, you must include all the partitioning columns in the specification, and specify constant values for all the partition key columns.

DROP STATS removes all statistics from the table, whether created by COMPUTE STATS or COMPUTE INCREMENTAL STATS.

DROP INCREMENTAL STATS only affects incremental statistics for a single partition, specified through the PART ITION clause. The incremental stats are marked as outdated, so that they are recomputed by the next COMPUTE INCREMENTAL STATS statement.

Usage notes:

You typically use this statement when the statistics for a table or a partition have become stale due to data files being added to or removed from the associated HDFS data directories, whether by manual HDFS operations or INSERT, INSERT OVERWRITE, or LOAD DATA statements, or adding or dropping partitions.

When a table or partition has no associated statistics, Impala treats it as essentially zero-sized when constructing the execution plan for a query. In particular, the statistics influence the order in which tables are joined in a join query. To ensure proper query planning and good query performance and scalability, make sure to run COMPUTE STATS or COMPUTE INCREMENTAL STATS on the table or partition after removing any stale statistics.

Dropping the statistics is not required for an unpartitioned table or a partitioned table covered by the original type of statistics. A subsequent COMPUTE STATS statement replaces any existing statistics with new ones, for all partitions, regardless of whether the old ones were outdated. Therefore, this statement was rarely used before the introduction of incremental statistics.

Dropping the statistics is required for a partitioned table containing incremental statistics, to make a subsequent COMPUTE INCREMENTAL STATS statement rescan an existing partition. See the *Tables and columns topic* for information about incremental statistics, a new feature available in Impala 2.1.0 and higher.

Statement type: DDL

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, does not need any particular HDFS permissions to perform this statement. All read and write operations are on the metastore database, not HDFS files and directories.

Examples:

The following example shows a partitioned table that has associated statistics produced by the COMPUTE INCR EMENTAL STATS statement, and how the situation evolves as statistics are dropped from specific partitions, then the entire table.

Initially, all table and column statistics are filled in.

```
show table stats item_partitioned;
+-------------+-------+--------+----------+--------------+---------+-----
------------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Increm
ental stats
+-------------+-------+--------+----------+--------------+---------+--------
---------
| Books       | 1733  | 1      | 223.74KB | NOT CACHED   | PARQUET | true
| Children    | 1786  | 1      | 230.05KB | NOT CACHED   | PARQUET | true
| Electronics | 1812  | 1      | 232.67KB | NOT CACHED   | PARQUET | true
| Home        | 1807  | 1      | 232.56KB | NOT CACHED   | PARQUET | true
| Jewelry     | 1740  | 1      | 223.72KB | NOT CACHED   | PARQUET | true
| Men         | 1811  | 1      | 231.25KB | NOT CACHED   | PARQUET | true
| Music       | 1860  | 1      | 237.90KB | NOT CACHED   | PARQUET | true
| Shoes       | 1835  | 1      | 234.90KB | NOT CACHED   | PARQUET | true
| Sports      | 1783  | 1      | 227.97KB | NOT CACHED   | PARQUET | true
| Women       | 1790  | 1      | 226.27KB | NOT CACHED   | PARQUET | true
| Total       | 17957 | 10     | 2.25MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+-----
------------
show column stats item_partitioned;
+------------------+-----------+------------------+--------+----------+----
----------
```

```
| Column            | Type      | #Distinct Values | #Nulls | Max Size | Avg
 Size
+------------------+-----------+------------------+--------+----------+-----
---------
| i_item_sk         | INT       | 19443            | -1     | 4        | 4
| i_item_id         | STRING    | 9025             | -1     | 16       | 16
| i_rec_start_date | TIMESTAMP | 4                | -1     | 16       | 16
| i_rec_end_date   | TIMESTAMP | 3                | -1     | 16       | 16
| i_item_desc       | STRING    | 13330            | -1     | 200      | 100
.302803039
| i_current_price  | FLOAT     | 2807             | -1     | 4        | 4
| i_wholesale_cost | FLOAT     | 2105             | -1     | 4        | 4
| i_brand_id        | INT       | 965              | -1     | 4        | 4
| i_brand           | STRING    | 725              | -1     | 22       | 16.
1776008605
| i_class_id        | INT       | 16               | -1     | 4        | 4
| i_class           | STRING    | 101              | -1     | 15       | 7.7
6749992370
| i_category_id     | INT       | 10               | -1     | 4        | 4
| i_manufact_id     | INT       | 1857             | -1     | 4        | 4
| i_manufact        | STRING    | 1028             | -1     | 15       | 1
1.3295001983
| i_size            | STRING    | 8                | -1     | 11       | 4.3
3459997177
| i_formulation     | STRING    | 12884            | -1     | 20       | 1
9.9799995422
| i_color           | STRING    | 92               | -1     | 10       | 5.3
8089990615
| i_units           | STRING    | 22               | -1     | 7        | 4
.18690013885
| i_container       | STRING    | 2                | -1     | 7        | 6.9
9259996414
| i_manager_id      | INT       | 105              | -1     | 4        | 4
| i_product_name    | STRING    | 19094            | -1     | 25       | 18.
0233001708
| i_category        | STRING    | 10               | 0      | -1       | -1
+------------------+-----------+------------------+--------+----------+---
-----------
```

To remove statistics for particular partitions, use the DROP INCREMENTAL STATS statement. After removing statistics for two partitions, the table-level statistics reflect that change in the #Rows and Incremental stats fields. The counts, maximums, and averages of the column-level statistics are unaffected.

Note: (It is possible that the row count might be preserved in future after a DROP INCREMENTAL STATS statement. Check the resolution of the issue IMPALA-1615.)

```
drop incremental stats item_partitioned partition (i_category='Sports');
drop incremental stats item_partitioned partition (i_category='Electronics'
);

show table stats item_partitioned
+-------------+-------+--------+----------+--------------+---------+------
------------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Increm
ental stats
+-------------+-------+--------+----------+--------------+---------+--------
---------
| Books       | 1733  | 1      | 223.74KB | NOT CACHED   | PARQUET | true
| Children    | 1786  | 1      | 230.05KB | NOT CACHED   | PARQUET | true
| Electronics | -1    | 1      | 232.67KB | NOT CACHED   | PARQUET | false
| Home        | 1807  | 1      | 232.56KB | NOT CACHED   | PARQUET | true
| Jewelry     | 1740  | 1      | 223.72KB | NOT CACHED   | PARQUET | true
```

```
| Men          | 1811  | 1      | 231.25KB | NOT CACHED   | PARQUET | true
| Music        | 1860  | 1      | 237.90KB | NOT CACHED   | PARQUET | true
| Shoes        | 1835  | 1      | 234.90KB | NOT CACHED   | PARQUET | true
| Sports       | -1    | 1      | 227.97KB | NOT CACHED   | PARQUET | false
| Women        | 1790  | 1      | 226.27KB | NOT CACHED   | PARQUET | true
| Total        | 17957 | 10     | 2.25MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+-------
----------
show column stats item_partitioned
+------------------+-----------+------------------+--------+----------+---
-----------
| Column           | Type      | #Distinct Values | #Nulls | Max Size | Avg
Size
+------------------+-----------+------------------+--------+----------+----
----------
| i_item_sk        | INT       | 19443            | -1     | 4        | 4
| i_item_id        | STRING    | 9025             | -1     | 16       | 16
| i_rec_start_date | TIMESTAMP | 4                | -1     | 16       | 16
| i_rec_end_date   | TIMESTAMP | 3                | -1     | 16       | 16
| i_item_desc      | STRING    | 13330            | -1     | 200      | 10
0.302803039
| i_current_price  | FLOAT     | 2807             | -1     | 4        | 4
| i_wholesale_cost | FLOAT     | 2105             | -1     | 4        | 4
| i_brand_id       | INT       | 965              | -1     | 4        | 4
| i_brand          | STRING    | 725              | -1     | 22       | 16
.1776008605
| i_class_id       | INT       | 16               | -1     | 4        | 4
| i_class          | STRING    | 101              | -1     | 15       | 7.
76749992370
| i_category_id    | INT       | 10               | -1     | 4        | 4
| i_manufact_id    | INT       | 1857             | -1     | 4        | 4
| i_manufact       | STRING    | 1028             | -1     | 15       | 11.3
295001983
| i_size           | STRING    | 8                | -1     | 11       | 4.
33459997177
| i_formulation    | STRING    | 12884            | -1     | 20       | 19.9
799995422
| i_color          | STRING    | 92               | -1     | 10       | 5.
38089990615
| i_units          | STRING    | 22               | -1     | 7        | 4.18
690013885
| i_container      | STRING    | 2                | -1     | 7        | 6.
99259996414
| i_manager_id     | INT       | 105              | -1     | 4        | 4
| i_product_name   | STRING    | 19094            | -1     | 25       | 18
.0233001708
| i_category       | STRING    | 10               | 0      | -1       | -1
+------------------+-----------+------------------+--------+----------+--
------------
```

To remove all statistics from the table, whether produced by COMPUTE STATS or COMPUTE INCREMENTAL STATS, use the DROP STATS statement without the INCREMENTAL clause). Now, both table-level and column-level statistics are reset.

```
drop stats item_partitioned;

show table stats item_partitioned
+-------------+-------+--------+----------+--------------+---------+--------
----------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Incr
emental stats
+-------------+-------+--------+----------+--------------+---------+------
------------
```

```
| Books        | -1    | 1      | 223.74KB | NOT CACHED   | PARQUET | false
| Children     | -1    | 1      | 230.05KB | NOT CACHED   | PARQUET | false
| Electronics  | -1    | 1      | 232.67KB | NOT CACHED   | PARQUET | false
| Home         | -1    | 1      | 232.56KB | NOT CACHED   | PARQUET | false
| Jewelry      | -1    | 1      | 223.72KB | NOT CACHED   | PARQUET | false
| Men          | -1    | 1      | 231.25KB | NOT CACHED   | PARQUET | false
| Music        | -1    | 1      | 237.90KB | NOT CACHED   | PARQUET | false
| Shoes        | -1    | 1      | 234.90KB | NOT CACHED   | PARQUET | false
| Sports       | -1    | 1      | 227.97KB | NOT CACHED   | PARQUET | false
| Women        | -1    | 1      | 226.27KB | NOT CACHED   | PARQUET | false
| Total        | -1    | 10     | 2.25MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+--------
----------
show column stats item_partitioned
+------------------+-----------+------------------+--------+----------+---
-------+
| Column           | Type      | #Distinct Values | #Nulls | Max Size | Avg
 Size |
+------------------+-----------+------------------+--------+----------+-----
-----+
| i_item_sk        | INT       | -1               | -1     | 4        | 4
     |
| i_item_id        | STRING    | -1               | -1     | -1       | -1
     |
| i_rec_start_date | TIMESTAMP | -1               | -1     | 16       | 16
     |
| i_rec_end_date   | TIMESTAMP | -1               | -1     | 16       | 16
     |
| i_item_desc      | STRING    | -1               | -1     | -1       | -1
     |
| i_current_price  | FLOAT     | -1               | -1     | 4        | 4
     |
| i_wholesale_cost | FLOAT     | -1               | -1     | 4        | 4
     |
| i_brand_id       | INT       | -1               | -1     | 4        | 4
     |
| i_brand          | STRING    | -1               | -1     | -1       | -1
     |
| i_class_id       | INT       | -1               | -1     | 4        | 4
     |
| i_class          | STRING    | -1               | -1     | -1       | -1
     |
| i_category_id    | INT       | -1               | -1     | 4        | 4
     |
| i_manufact_id    | INT       | -1               | -1     | 4        | 4
     |
| i_manufact       | STRING    | -1               | -1     | -1       | -1
     |
| i_size           | STRING    | -1               | -1     | -1       | -1
     |
| i_formulation    | STRING    | -1               | -1     | -1       | -1
     |
| i_color          | STRING    | -1               | -1     | -1       | -1
     |
| i_units          | STRING    | -1               | -1     | -1       | -1
     |
| i_container      | STRING    | -1               | -1     | -1       | -1
     |
| i_manager_id     | INT       | -1               | -1     | 4        | 4
     |
| i_product_name   | STRING    | -1               | -1     | -1       | -1
     |
| i_category       | STRING    | 10               | 0      | -1       | -1
     |
```

```
+------------------+-----------+------------------+--------+----------+----
------+
```

Related Information

Table and column statistics

DROP TABLE statement

The DROP TABLE statement removes an Impala table. Also removes the underlying HDFS data files for internal tables, although not for external tables.

Syntax:

```
DROP TABLE [IF EXISTS] [DB_NAME.]TABLE_NAME [PURGE]
```

IF EXISTS clause:

The optional IF EXISTS clause makes the statement succeed whether or not the table exists. If the table does exist, it is dropped; if it does not exist, the statement has no effect. This capability is useful in standardized setup scripts that remove existing schema objects and create new ones. By using some combination of IF EXISTS for the DROP statements and IF NOT EXISTS clauses for the CREATE statements, the script can run successfully the first time you run it (when the objects do not exist yet) and subsequent times (when some or all of the objects do already exist).

PURGE clause:

The optional PURGE keyword, available in Impala 2.3 and higher, causes Impala to remove the associated HDFS data files immediately, rather than going through the HDFS trashcan mechanism. Use this keyword when dropping a table if it is crucial to remove the data as quickly as possible to free up space, or if there is a problem with the trashcan, such as the trash cannot being configured or being in a different HDFS encryption zone than the data files.

Statement type: DDL

Usage notes:

By default, Impala removes the associated HDFS directory and data files for the table. If you issue a DROP TABLE and the data files are not deleted, it might be for the following reasons:

- If the table was created with the EXTERNAL clause, Impala leaves all files and directories untouched. Use external tables when the data is under the control of other Hadoop components, and Impala is only used to query the data files from their original locations.
- Impala might leave the data files behind unintentionally, if there is no HDFS location available to hold the HDFS trashcan for the impala user.

Make sure that you are in the correct database before dropping a table, either by issuing a USE statement first or by using a fully qualified name *DB_NAME.TABLE_NAME*.

If you intend to issue a DROP DATABASE statement, first issue DROP TABLE statements to remove all the tables in that database.

Examples:

```
create database temporary;
use temporary;
create table unimportant (x int);
create table trivial (s string);
-- Drop a table in the current database.
drop table unimportant;
-- Switch to a different database.
use default;
-- To drop a table in a different database...
drop table trivial;
ERROR: AnalysisException: Table does not exist: default.trivial
```

```
-- ...use a fully qualified name.
drop table temporary.trivial;
```

Amazon S3 considerations:

The DROP TABLE statement can remove data files from S3 if the associated S3 table is an internal table. In Impala 2.6 and higher, as part of improved support for writing to S3, Impala also removes the associated folder when dropping an internal table that resides on S3.

For compatibility with the S3 write support in Impala, follow these steps for querying table through Impala:

1. Use native Hadoop techniques, such as hadoop fs -cp or INSERT in Impala or Hive to create data files in S3.
2. Use the PURGE clause with DROP TABLE when dropping internal (managed) tables.

By default, when you do not include the PURGE clause in the statement, the data files are moved to the S3A trashcan. This operation is expensive. When you do include the PURGE clause, the data files are deleted immediately, skipping the expensive S3A trashcan operation.

In Impala 2.6 and higher, Impala DDL statements such as CREATE DATABASE, CREATE TABLE, DROP DATABASE CASCADE, DROP TABLE, and ALTER TABLE [ADD|DROP] PARTITION can create or remove folders as needed in the Amazon S3 system. Prior to Impala 2.6, you had to create folders yourself and point Impala database, tables, or partitions at them, and manually remove folders when no longer needed.

Cancellation: Cannot be cancelled.

HDFS permissions:

For an internal table, the user ID that the `impalad` daemon runs under, typically the impala user, must have write permission for all the files and directories that make up the table.

For an external table, dropping the table only involves changes to metadata in the metastore database. Because Impala does not remove any HDFS files or directories when external tables are dropped, no particular permissions are needed for the associated HDFS files or directories.

Kudu considerations:

Kudu tables can be managed or external, the same as with HDFS-based tables. For a managed table, the underlying Kudu table and its data are removed by DROP TABLE. For an external table, the underlying Kudu table and its data remain after a DROP TABLE.

Related Information

CREATE DATABASE statement

DROP VIEW statement

The DROP VIEW statement removes the specified view, which was originally created by the CREATE VIEW statement. Because a view is purely a logical construct (an alias for a query) with no physical data behind it, DROP VIEW only involves changes to metadata in the metastore database, not any data files in HDFS.

Syntax:

```
DROP VIEW [IF EXISTS] [DB_NAME.]VIEW_NAME
```

Statement type: DDL

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Examples:

The following example creates a series of views and then drops them. These examples illustrate how views are associated with a particular database, and both the view definitions and the view names for CREATE VIEW and DROP VIEW can refer to a view in the current database or a fully qualified view name.

```
-- Create and drop a view in the current database.
CREATE VIEW few_rows_from_t1 AS SELECT * FROM t1 LIMIT 10;
DROP VIEW few_rows_from_t1;

-- Create and drop a view referencing a table in a different database.
CREATE VIEW table_from_other_db AS SELECT x FROM db1.foo WHERE x IS NOT N
ULL;
DROP VIEW table_from_other_db;
USE db1;
-- Create a view in a different database.
CREATE VIEW db2.v1 AS SELECT * FROM db2.foo;
-- Switch into the other database and drop the view.
USE db2;
DROP VIEW v1;

USE db1;
-- Create a view in a different database.
CREATE VIEW db2.v1 AS SELECT * FROM db2.foo;
-- Drop a view in the other database.
DROP VIEW db2.v1;
```

EXPLAIN statement

The EXPLAIN statement returns the execution plan for a statement, showing the low-level mechanisms that Impala will use to read the data, divide the work among nodes in the cluster, and transmit intermediate and final results across the network.

Use explain followed by a complete SELECT query. For example:

Syntax:

```
EXPLAIN { SELECT_QUERY | CTAS_STMT | INSERT_STMT }
```

The *SELECT_QUERY* is a SELECT statement, optionally prefixed by a WITH clause.

The *INSERT_STMT* is an INSERT statement that inserts into or overwrites an existing table. It can use either the INSERT ... SELECT or INSERT ... VALUES syntax.

The *CTAS_STMT* is a CREATE TABLE statement using the AS SELECT clause, typically abbreviated as a “CTAS” operation.

Usage notes:

You can interpret the output to judge whether the query is performing efficiently, and adjust the query and/or the schema if not. For example, you might change the tests in the WHERE clause, add hints to make join operations more efficient, introduce subqueries, change the order of tables in a join, add or change partitioning for a table, collect column statistics and/or table statistics in Hive, or any other performance tuning steps.

The EXPLAIN output reminds you if table or column statistics are missing from any table involved in the query. These statistics are important for optimizing queries involving large tables or multi-table joins. See COMPUTE STATS statement on page 124 for how to gather statistics, and Table and column statistics for how to use this information for query tuning.

Read the EXPLAIN plan from bottom to top:

- The last part of the plan shows the low-level details such as the expected amount of data that will be read, where you can judge the effectiveness of your partitioning strategy and estimate how long it will take to scan a table based on total data size and the size of the cluster.
- As you work your way up, next you see the operations that will be parallelized and performed on each Impala node.
- At the higher levels, you see how data flows when intermediate result sets are combined and transmitted from one node to another.
- The EXPLAIN_LEVEL query option lets you customize how much detail to show in the EXPLAIN plan depending on whether you are doing high-level or low-level tuning, dealing with logical or physical aspects of the query.

If you come from a traditional database background and are not familiar with data warehousing, keep in mind that Impala is optimized for full table scans across very large tables. The structure and distribution of this data is typically not suitable for the kind of indexing and single-row lookups that are common in OLTP environments. Seeing a query scan entirely through a large table is common, not necessarily an indication of an inefficient query. Of course, if you can reduce the volume of scanned data by orders of magnitude, for example by using a query that affects only certain partitions within a partitioned table, then you might be able to optimize a query so that it executes in seconds rather than minutes.

Extended EXPLAIN output:

For performance tuning of complex queries, and capacity planning (such as using the admission control and resource management features), you can enable more detailed and informative output for the EXPLAIN statement. In the `impala-shell` interpreter, issue the command SET EXPLAIN_LEVEL=*LEVEL*, where *LEVEL* is an integer from 0 to 3 or corresponding mnemonic values minimal, standard, extended, or verbose.

When extended EXPLAIN output is enabled, EXPLAIN statements print information about estimated memory requirements, minimum number of virtual cores, and so on.

Starting in Impala 3.2, if the EXPLAIN_LEVEL option is set to EXTENDED level or VERBOSE, the output contains the following additional information.

- The analyzed query, in the output header.

  The analyzed query may have been rewritten to include various optimizations and implicit casts. See the example below.
- The predicates in the plan output includes the same implicit casts and literals printed with a cast to show the type.

Examples:

This example shows how the standard EXPLAIN output moves from the lowest (physical) level to the higher (logical) levels. The query begins by scanning a certain amount of data; each node performs an aggregation operation (evaluating COUNT(*)) on some subset of data that is local to that node; the intermediate results are transmitted back to the coordinator node (labelled here as the EXCHANGE node); lastly, the intermediate results are summed to display the final result.

```
[impalad-host:21000] > explain select count(*) from customer_address;
+----------------------------------------------------------+
| Explain String                                           |
+----------------------------------------------------------+
| Estimated Per-Host Requirements: Memory=42.00MB VCores=1 |
|                                                          |
| 03:AGGREGATE [MERGE FINALIZE]                            |
| |  output: sum(count(*))                                 |
| |                                                        |
| 02:EXCHANGE [PARTITION=UNPARTITIONED]                    |
| |                                                        |
| 01:AGGREGATE                                             |
| |  output: count(*)                                      |
| |                                                        |
| 00:SCAN HDFS [default.customer_address]                  |
|    partitions=1/1 size=5.25MB                            |
```

```
+-----------------------------------------------------------+
```

The following example shows an extended EXPLAIN output. Note that the analyzed query was rewritten to include:

- The 'constant folding' optimization, which simplified the expression in the original query, '1000 / 100' to '10'.
- The implicit casts in the WHERE clause.

```
EXPLAIN SELECT * FROM functional_kudu.alltypestiny WHERE bigint_col < 1000 /
 100;
+-----------------------------------------------------------+
| Explain String                                            |
+-----------------------------------------------------------+
| ...
| Analyzed query: SELECT * FROM mytable WHERE CAST(bigint_col AS DOUBLE) <
 CAST(10 AS DOUBLE)
| ...
| 00:SCAN KUDU [functional_kudu.alltypestiny]
| predicates: CAST(bigint_col AS DOUBLE) < CAST(10 AS DOUBLE)
...
```

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all applicable directories in all source tables for the query that is being explained. (A SELECT operation could read files from multiple different HDFS directories if the source table is partitioned.)

Kudu considerations:

The EXPLAIN statement displays equivalent plan information for queries against Kudu tables as for queries against HDFS-based tables.

To see which predicates Impala can “push down” to Kudu for efficient evaluation, without transmitting unnecessary rows back to Impala, look for the kudu predicates item in the scan phase of the query. The label kudu predicates indicates a condition that can be evaluated efficiently on the Kudu side. The label predicates in a SCAN KUDU node indicates a condition that is evaluated by Impala. For example, in a table with primary key column X and non-primary key column Y, you can see that some operators in the WHERE clause are evaluated immediately by Kudu and others are evaluated later by Impala:

```
EXPLAIN SELECT x,y from kudu_table WHERE
  x = 1 AND y NOT IN (2,3) AND z = 1
  AND a IS NOT NULL AND b > 0 AND length(s) > 5;
+----------------
| Explain String
+----------------
...
| 00:SCAN KUDU [kudu_table]
|    predicates: y NOT IN (2, 3), length(s) > 5
|    kudu predicates: a IS NOT NULL, b > 0, x = 1, z = 1
```

Only binary predicates, IS NULL and IS NOT NULL (in Impala 2.9 and higher), and IN predicates containing literal values that exactly match the types in the Kudu table, and do not require any casting, can be pushed to Kudu.

Related Information

EXPLAIN_LEVEL query option

Table and column statistics
Understanding performance using Explain plan

GRANT statement

The GRANT statement grants a privilege on a specified object to a user or to a group.

Statement

Syntax:

```
      GRANT PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
      TO USER USER_NAME
      GRANT PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
      TO GROUP GROUP_NAME

      privilege ::= ALL | ALTER | CREATE | DROP | INSERT | REFRESH | SELECT
 | SELECT(COLUMN_NAME)
      object_type ::= SERVER | URI | DATABASE | TABLE
```

Typically, the *OBJECT_NAME* is an identifier. For URIs, it is a string literal.

The ability to grant or revoke SELECT privilege on specific columns is available in Impala 2.3 and higher.

Usage notes:

You can only grant the ALL privilege to the URI object. Finer-grained privileges mentioned below on a URI are not supported.

The table below lists the minimum level of privileges and the scope required to execute SQL statements. The following notations are used:

- ANY denotes the SELECT, INSERT, CREATE, or REFRESH privilege.
- ALL privilege denotes the SELECT, INSERT, CREATE, and REFRESH privileges.
- The owner of an object effectively has the ALL privilege on the object.
- The parent levels of the specified scope are implicitly supported. For example, if a privilege is listed with the TABLE scope, the same privilege granted on DATABASE and SERVER will allow the user to execute that specific SQL statement on TABLE.

SQL Statement	Privileges	Scope
SELECT	SELECT	TABLE
WITH SELECT	SELECT	TABLE
EXPLAIN SELECT	SELECT	TABLE
INSERT	INSERT	TABLE
EXPLAIN INSERT	INSERT	TABLE
TRUNCATE	INSERT	TABLE
LOAD	INSERT	TABLE
	ALL	URI
CREATE DATABASE	CREATE	SERVER
CREATE DATABASE LOCATION	CREATE	SERVER
	ALL	URI
CREATE TABLE	CREATE	DATABASE

CREATE TABLE LIKE	CREATE	DATABASE
	SELECT, INSERT, or REFRESH	TABLE
CREATE TABLE AS SELECT	CREATE	DATABASE
	INSERT	DATABASE
	SELECT	TABLE
EXPLAIN CREATE TABLE AS SELECT	CREATE	DATABASE
	INSERT	DATABASE
	SELECT	TABLE
CREATE TABLE LOCATION	CREATE	TABLE
	ALL	URI
CREATE VIEW	CREATE	DATABASE
	SELECT	TABLE
ALTER DATABASE SET OWNER	ALL WITH GRANT	DATABASE
ALTER TABLE	ALL	TABLE
ALTER TABLE SET LOCATION	ALL	TABLE
	ALL	URI
ALTER TABLE RENAME	CREATE	DATABASE
	ALL	TABLE
ALTER TABLE SET OWNER	ALL WITH GRANT	TABLE
ALTER VIEW	ALL	TABLE
	SELECT	TABLE
ALTER VIEW RENAME	CREATE	DATABASE
	ALL	TABLE
ALTER VIEW SET OWNER	ALL WITH GRANT	VIEW
DROP DATABASE	ALL	DATABASE
DROP TABLE	ALL	TABLE
DROP VIEW	ALL	TABLE
CREATE FUNCTION	CREATE	DATABASE
	ALL	URI
DROP FUNCTION	ALL	DATABASE
COMPUTE STATS	ALL	TABLE
DROP STATS	ALL	TABLE
INVALIDATE METADATA	REFRESH	SERVER
INVALIDATE METADATA <table>	REFRESH	TABLE
REFRESH <table>	REFRESH	TABLE
REFRESH AUTHORIZATION	REFRESH	SERVER
REFRESH FUNCTIONS	REFRESH	DATABASE
COMMENT ON DATABASE	ALL	DATABASE
COMMENT ON TABLE	ALL	TABLE
COMMENT ON VIEW	ALL	TABLE

COMMENT ON COLUMN	ALL	TABLE
DESCRIBE DATABASE	SELECT, INSERT, or REFRESH	DATABASE
DESCRIBE <table/view>	SELECT, INSERT, or REFRESH	TABLE
If the user has the SELECT privilege at the COLUMN level, only the columns the user has access will show.	SELECT	COLUMN
USE	ANY	TABLE
SHOW DATABASES	ANY	TABLE
SHOW TABLES	ANY	TABLE
SHOW FUNCTIONS	SELECT, INSERT, or REFRESH	DATABASE
SHOW PARTITIONS	SELECT, INSERT, or REFRESH	TABLE
SHOW TABLE STATS	SELECT, INSERT, or REFRESH	TABLE
SHOW COLUMN STATS	SELECT, INSERT, or REFRESH	TABLE
SHOW FILES	SELECT, INSERT, or REFRESH	TABLE
SHOW CREATE TABLE	SELECT, INSERT, or REFRESH	TABLE
SHOW CREATE VIEW	SELECT, INSERT, or REFRESH	TABLE
SHOW CREATE FUNCTION	SELECT, INSERT, or REFRESH	DATABASE
SHOW RANGE PARTITIONS (Kudu only)	SELECT, INSERT, or REFRESH	TABLE
UPDATE (Kudu only)	ALL	TABLE
EXPLAIN UPDATE (Kudu only)	ALL	TABLE
UPSERT (Kudu only)	ALL	TABLE
WITH UPSERT (Kudu only)	ALL	TABLE
EXPLAIN UPSERT (Kudu only)	ALL	TABLE
DELETE (Kudu only)	ALL	TABLE
EXPLAIN DELETE (Kudu only)	ALL	TABLE

Compatibility:

- The Impala GRANT and REVOKE statements are available in Impala 2.0 and later.
- In Impala 1.4 and later, Impala can make use of any privileges specified by the GRANT and REVOKE statements in Hive.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

GRANT ROLE statement

The GRANT statement assigns a role to a group.

Syntax

```
GRANT ROLE ROLE_NAME TO GROUP GROUP_NAME
```

INSERT statement

Impala supports inserting into tables and partitions that you create with the Impala CREATE TABLE statement, or pre-defined tables and partitions created through Hive.

Syntax:

```
[WITH_CLAUSE]
  INSERT [HINT_CLAUSE] { INTO | OVERWRITE } [TABLE] TABLE_NAME
  [(COLUMN_LIST)]
  [ PARTITION (PARTITION_CLAUSE)]
{
    [HINT_CLAUSE] SELECT_STATEMENT
  | VALUES (VALUE [, VALUE ...]) [, (VALUE [, VALUE ...]) ...]
}

partition_clause ::= COL_NAME [= CONSTANT] [, COL_NAME [= CONSTANT] ...]

hint_clause ::=
  HINT_WITH_DASHES |
  HINT_WITH_CSTYLE_DELIMITERS |
  HINT_WITH_BRACKETS

hint_with_dashes ::= -- +SHUFFLE | -- +NOSHUFFLE -- +CLUSTERED

hint_with_cstyle_comments ::= /* +SHUFFLE */ | /* +NOSHUFFLE */ | /*
 +CLUSTERED */

hint_with_brackets ::= [SHUFFLE] | [NOSHUFFLE]
  (With this hint format, the square brackets are part of the syntax.)
```

Note: The square bracket style of hint is now deprecated and might be removed in a future release. For that reason, any newly added hints are not available with the square bracket syntax.

Appending or replacing (INTO and OVERWRITE clauses):

The INSERT INTO syntax appends data to a table. The existing data files are left as-is, and the inserted data is put into one or more new data files.

The INSERT OVERWRITE syntax replaces the data in a table. Currently, the overwritten data files are deleted immediately; they do not go through the HDFS trash mechanism.

Complex type considerations:

The INSERT statement currently does not support writing data files containing complex types (ARRAY, STRUCT, and MAP). To prepare Parquet data for such tables, you generate the data files outside Impala and then use LOAD DATA or CREATE EXTERNAL TABLE to associate those data files with the table. Currently, such tables must use the Parquet file format.

Kudu considerations:

Currently, the INSERT OVERWRITE syntax cannot be used with Kudu tables.

Kudu tables require a unique primary key for each row. If an INSERT statement attempts to insert a row with the same values for the primary key columns as an existing row, that row is discarded and the insert operation continues. When rows are discarded due to duplicate primary keys, the statement finishes with a warning, not an error. (This is a change from early releases of Kudu where the default was to return in error in such cases, and the syntax INSERT IGNORE was required to make the statement succeed. The IGNORE clause is no longer part of the INSERT syntax.)

For situations where you prefer to replace rows with duplicate primary key values, rather than discarding the new data, you can use the UPSERT statement instead of INSERT. UPSERT inserts rows that are entirely new, and for

rows that match an existing primary key in the table, the non-primary-key columns are updated to reflect the values in the “upserted” data.

If you really want to store new rows, not replace existing ones, but cannot do so because of the primary key uniqueness constraint, consider recreating the table with additional columns included in the primary key.

Usage notes:

Impala currently supports:

- Copy data from another table using SELECT query. In Impala 1.2.1 and higher, you can combine CREATE TABLE and INSERT operations into a single step with the CREATE TABLE AS SELECT syntax, which bypasses the actual INSERT keyword.
- An optional WITH clause before the INSERT keyword, to define a subquery referenced in the SELECT portion.
- Create one or more new rows using constant expressions through VALUES clause. (The VALUES clause was added in Impala 1.0.1.)
- By default, the first column of each newly inserted row goes into the first column of the table, the second column into the second column, and so on.

  You can also specify the columns to be inserted, an arbitrarily ordered subset of the columns in the destination table, by specifying a column list immediately after the name of the destination table. This feature lets you adjust the inserted columns to match the layout of a SELECT statement, rather than the other way around. (This feature was added in Impala 1.1.)

  The number of columns mentioned in the column list (known as the “column permutation”) must match the number of columns in the SELECT list or the VALUES tuples. The order of columns in the column permutation can be different than in the underlying table, and the columns of each input row are reordered to match. If the number of columns in the column permutation is less than in the destination table, all unmentioned columns are set to NULL.
- An optional hint clause immediately either before the SELECT keyword or after the INSERT keyword, to fine-tune the behavior when doing an INSERT ... SELECT operation into partitioned Parquet tables. The hint clause cannot be specified in multiple places. The hint keywords are [SHUFFLE] and [NOSHUFFLE], including the square brackets. Inserting into partitioned Parquet tables can be a resource-intensive operation because it potentially involves many files being written to HDFS simultaneously, and separate large memory buffers being allocated to buffer the data for each partition.

Note:

- Insert commands that partition or add files result in changes to Hive metadata. Because Impala uses Hive metadata, such changes may necessitate a metadata refresh. For more information, see the REFRESH function.
- Currently, Impala can only insert data into tables that use the text and Parquet formats. For other file formats, insert the data using Hive and use Impala to query it.
- As an alternative to the INSERT statement, if you have existing data files elsewhere in HDFS, the LOAD DATA statement can move those files into a table. This statement works with tables of any file format.

Statement type: DML (but still affected by the SYNC_DDL query option)

Usage notes:

When you insert the results of an expression, particularly of a built-in function call, into a small numeric column such as INT, SMALLINT, TINYINT, or FLOAT, you might need to use a CAST() expression to coerce values into the appropriate type. Impala does not automatically convert from a larger type to a smaller one. For example, to insert cosine values into a FLOAT column, write CAST(COS(angle) AS FLOAT) in the INSERT statement to make the conversion explicit.

File format considerations:

Because Impala can read certain file formats that it cannot write, the INSERT statement does not work for all kinds of Impala tables. See *How Impala works with Hadoop file formats* for details about what file formats are supported by the INSERT statement.

Any INSERT statement for a Parquet table requires enough free space in the HDFS filesystem to write one block. Because Parquet data files use a block size of 1 GB by default, an INSERT might fail (even for a very small amount of data) if your HDFS is running low on space.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMPUTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

The following example sets up new tables with the same definition as the TAB1 table using different file formats, and demonstrates inserting data into the tables created with the STORED AS TEXTFILE and STORED AS PARQUET clauses:

```
CREATE DATABASE IF NOT EXISTS file_formats;
USE file_formats;

DROP TABLE IF EXISTS text_table;
CREATE TABLE text_table
( id INT, col_1 BOOLEAN, col_2 DOUBLE, col_3 TIMESTAMP )
STORED AS TEXTFILE;

DROP TABLE IF EXISTS parquet_table;
CREATE TABLE parquet_table
( id INT, col_1 BOOLEAN, col_2 DOUBLE, col_3 TIMESTAMP )
STORED AS PARQUET;
```

With the INSERT INTO TABLE syntax, each new set of inserted rows is appended to any existing data in the table. This is how you would record small amounts of data that arrive continuously, or ingest new batches of data alongside the existing data. For example, after running 2 INSERT INTO TABLE statements with 5 rows each, the table contains 10 rows total:

```
[localhost:21000] > insert into table text_table select * from default.tab1;
Inserted 5 rows in 0.41s

[localhost:21000] > insert into table text_table select * from default.tab1;
Inserted 5 rows in 0.46s

[localhost:21000] > select count(*) from text_table;
+----------+
| count(*) |
+----------+
| 10       |
+----------+
Returned 1 row(s) in 0.26s
```

With the INSERT OVERWRITE TABLE syntax, each new set of inserted rows replaces any existing data in the table. This is how you load data to query in a data warehousing scenario where you analyze just the data for a particular day, quarter, and so on, discarding the previous data each time. You might keep the entire set of data in one raw table, and transfer and transform certain rows into a more compact and efficient form to perform intensive analysis on that subset.

For example, here we insert 5 rows into a table using the INSERT INTO clause, then replace the data by inserting 3 rows with the INSERT OVERWRITE clause. Afterward, the table only contains the 3 rows from the final INSERT statement.

```
[localhost:21000] > insert into table parquet_table select * from default.ta
b1;
Inserted 5 rows in 0.35s

[localhost:21000] > insert overwrite table parquet_table select * from def
ault.tab1 limit 3;
Inserted 3 rows in 0.43s
[localhost:21000] > select count(*) from parquet_table;
+----------+
| count(*) |
+----------+
| 3        |
+----------+
Returned 1 row(s) in 0.43s
```

The VALUES clause lets you insert one or more rows by specifying constant values for all the columns. The number, types, and order of the expressions must match the table definition.

Note: The INSERT ... VALUES technique is not suitable for loading large quantities of data into HDFS-based tables, because the insert operations cannot be parallelized, and each one produces a separate data file. Use it for setting up small dimension tables or tiny amounts of data for experimenting with SQL syntax, or with HBase tables. Do not use it for large ETL jobs or benchmark tests for load operations. Do not run scripts with thousands of INSERT ... VALUES statements that insert a single row each time. If you do run INSE RT ... VALUES operations to load data into a staging table as one stage in an ETL pipeline, include multiple row values if possible within each VALUES clause, and use a separate database to make cleanup easier if the operation does produce many tiny files.

The following example shows how to insert one row or multiple rows, with expressions of different types, using literal values, expressions, and function return values:

```
create table val_test_1 (c1 int, c2 float, c3 string, c4 boolean, c5 timesta
mp);
insert into val_test_1 values (100, 99.9/10, 'abc', true, now());
create table val_test_2 (id int, token string);
insert overwrite val_test_2 values (1, 'a'), (2, 'b'), (-1,'xyzzy');
```

These examples show the type of “not implemented” error that you see when attempting to insert data into a table with a file format that Impala currently does not write to:

```
DROP TABLE IF EXISTS sequence_table;
CREATE TABLE sequence_table
( id INT, col_1 BOOLEAN, col_2 DOUBLE, col_3 TIMESTAMP )
STORED AS SEQUENCEFILE;

DROP TABLE IF EXISTS rc_table;
CREATE TABLE rc_table
( id INT, col_1 BOOLEAN, col_2 DOUBLE, col_3 TIMESTAMP )
STORED AS RCFILE;

[localhost:21000] > insert into table rc_table select * from default.tab1;
Remote error
Backend 0:RC_FILE not implemented.

[localhost:21000] > insert into table sequence_table select * from default.t
ab1;
Remote error
Backend 0:SEQUENCE_FILE not implemented.
```

The following examples show how you can copy the data in all the columns from one table to another, copy the data from only some columns, or specify the columns in the select list in a different order than they actually appear in the table:

```
-- Start with 2 identical tables.
create table t1 (c1 int, c2 int);
create table t2 like t1;

-- If there is no () part after the destination table name,
-- all columns must be specified, either as * or by name.
insert into t2 select * from t1;
insert into t2 select c1, c2 from t1;

-- With the () notation following the destination table name,
-- you can omit columns (all values for that column are NULL
-- in the destination table), and/or reorder the values
-- selected from the source table. This is the "column permutation" feature.
insert into t2 (c1) select c1 from t1;
insert into t2 (c2, c1) select c1, c2 from t1;

-- The column names can be entirely different in the source and destination
 tables.
-- You can copy any columns, not just the corresponding ones, from the so
urce table.
-- But the number and type of selected columns must match the columns mentio
ned in the () part.
alter table t2 replace columns (x int, y int);
insert into t2 (y) select c1 from t1;
```

Sorting considerations: Although you can specify an ORDER BY clause in an INSERT ... SELECT statement, any ORDER BY clause is ignored and the results are not necessarily sorted. An INSERT ... SELECT operation potentially creates many different data files, prepared by different executor Impala daemons, and therefore the notion of the data being stored in sorted order is impractical.

Concurrency considerations: Each INSERT operation creates new data files with unique names, so you can run multiple INSERT INTO statements simultaneously without filename conflicts. While data is being inserted into an Impala table, the data is staged temporarily in a subdirectory inside the data directory; during this period, you cannot issue queries against that table in Hive. If an INSERT operation fails, the temporary data file and the subdirectory could be left behind in the data directory. If so, remove the relevant subdirectory and any data files it contains manually, by issuing an hdfs dfs -rm -r command, specifying the full path of the work subdirectory, whose name ends in _dir.

VALUES clause

The VALUES clause is a general-purpose way to specify the columns of one or more rows, typically within an INSERT statement.

Note: The INSERT ... VALUES technique is not suitable for loading large quantities of data into HDFS-based tables, because the insert operations cannot be parallelized, and each one produces a separate data file. Use it for setting up small dimension tables or tiny amounts of data for experimenting with SQL syntax, or with HBase tables. Do not use it for large ETL jobs or benchmark tests for load operations. Do not run scripts with thousands of INSERT ... VALUES statements that insert a single row each time. If you do run INSERT ... VALUES operations to load data into a staging table as one stage in an ETL pipeline, include multiple row values if possible within each VALUES clause, and use a separate database to make cleanup easier if the operation does produce many tiny files.

The following examples illustrate:

- How to insert a single row using a VALUES clause.
- How to insert multiple rows using a VALUES clause.

- How the row or rows from a VALUES clause can be appended to a table through INSERT INTO, or replace the contents of the table through INSERT OVERWRITE.
- How the entries in a VALUES clause can be literals, function results, or any other kind of expression. See Impala SQL literals on page 89 for the notation to use for literal values, especially for quoting and escaping conventions for strings. See Impala SQL operators on page 93 and Impala built-in functions on page 303 for other things you can include in expressions with the VALUES clause.

```
[localhost:21000] > describe val_example;
Query: describe val_example
Query finished, fetching results ...
+-------+---------+---------+
| name  | type    | comment |
+-------+---------+---------+
| id    | int     |         |
| col_1 | boolean |         |
| col_2 | double  |         |
+-------+---------+---------+

[localhost:21000] > insert into val_example values (1,true,100.0);
Inserted 1 rows in 0.30s
[localhost:21000] > select * from val_example;
+----+-------+-------+
| id | col_1 | col_2 |
+----+-------+-------+
| 1  | true  | 100   |
+----+-------+-------+
[localhost:21000] > insert overwrite val_example values (10,false,pow(2,5)),
 (50,true,10/3);
Inserted 2 rows in 0.16s
[localhost:21000] > select * from val_example;
+----+-------+-------------------+
| id | col_1 | col_2             |
+----+-------+-------------------+
| 10 | false | 32                |
| 50 | true  | 3.333333333333333 |
+----+-------+-------------------+
```

When used in an INSERT statement, the Impala VALUES clause can specify some or all of the columns in the destination table, and the columns can be specified in a different order than they actually appear in the table. To specify a different set or order of columns than in the table, use the syntax:

```
INSERT INTO DESTINATION
  (COL_X, COL_Y, COL_Z)
  VALUES
  (VAL_X, VAL_Y, VAL_Z);
```

Any columns in the table that are not listed in the INSERT statement are set to NULL.

HDFS considerations:

Impala physically writes all inserted files under the ownership of its default user, typically impala. Therefore, this user must have HDFS write permission in the corresponding table directory.

The permission requirement is independent of the authorization performed by the Ranger framework. (If the connected user is not authorized to insert into a table, Ranger blocks that operation immediately, regardless of the privileges available to the impala user.) Files created by Impala are not owned by and do not inherit permissions from the connected user.

The number of data files produced by an INSERT statement depends on the size of the cluster, the number of data blocks that are processed, the partition key columns in a partitioned table, and the mechanism Impala uses for dividing the work in parallel. Do not assume that an INSERT statement will produce some particular number of output files. In case of performance issues with data written by Impala, check that the output files do not suffer from

issues such as many tiny files or many tiny partitions. (In the Hadoop context, even files or partitions of a few tens of megabytes are considered “tiny”.)

The INSERT statement has always left behind a hidden work directory inside the data directory of the table. Formerly, this hidden work directory was named .impala_insert_staging . In Impala 2.0.1 and later, this directory name is changed to _impala_insert_staging . (While HDFS tools are expected to treat names beginning either with underscore and dot as hidden, in practice names beginning with an underscore are more widely supported.) If you have any scripts, cleanup jobs, and so on that rely on the name of this work directory, adjust them to use the new name.

HBase considerations:

You can use the INSERT statement with HBase tables as follows:

- You can insert a single row or a small set of rows into an HBase table with the INSERT ... VALUES syntax. This is a good use case for HBase tables with Impala, because HBase tables are not subject to the same kind of fragmentation from many small insert operations as HDFS tables are.
- You can insert any number of rows at once into an HBase table using the INSERT ... SELECT syntax.
- If more than one inserted row has the same value for the HBase key column, only the last inserted row with that value is visible to Impala queries. You can take advantage of this fact with INSERT ... VALUES statements to effectively update rows one at a time, by inserting new rows with the same key values as existing rows. Be aware that after an INSERT ... SELECT operation copying from an HDFS table, the HBase table might contain fewer rows than were inserted, if the key column in the source table contained duplicate values.
- You cannot INSERT OVERWRITE into an HBase table. New rows are always appended.
- When you create an Impala or Hive table that maps to an HBase table, the column order you specify with the INSERT statement might be different than the order you declare with the CREATE TABLE statement. Behind the scenes, HBase arranges the columns based on how they are divided into column families. This might cause a mismatch during insert operations, especially if you use the syntax INSERT INTO *HBASE_TABLE* SELECT * FROM *HDFS_TABLE*. Before inserting data, verify the column order by issuing a DESCRIBE statement for the table, and adjust the order of the select list in the INSERT statement.

Amazon S3 considerations:

In Impala 2.6 and higher, the Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in the Amazon Simple Storage Service (S3). The syntax of the DML statements is the same as for any other tables, because the S3 location for tables and partitions is specified by an s3a:// prefix in the LOCATION attribute of CREATE TABLE or ALTER TABLE statements. If you bring data into S3 using the normal S3 transfer mechanisms instead of Impala DML statements, issue a REFRESH statement for the table before using Impala to query the S3 data.

Because of differences between S3 and traditional filesystems, DML operations for S3 tables can take longer than for tables on HDFS. For example, both the LOAD DATA statement and the final stage of the INSERT and CREATE TABLE AS SELECT statements involve moving files from one directory to another. (In the case of INSERT and CREATE TABLE AS SELECT, the files are moved from a temporary staging directory to the final destination directory.) Because S3 does not support a “rename” operation for existing objects, in these cases Impala actually copies the data files from one location to another and then removes the original files. In Impala 2.6, the S3_SKIP_INSERT_STAGING query option provides a way to speed up INSERT statements for S3 tables and partitions, with the tradeoff that a problem during statement execution could leave data in an inconsistent state. It does not apply to INSERT OVERWRITE or LOAD DATA statements.

ADLS considerations:

In Impala 2.9 and higher, the Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in the Azure Data Lake Store (ADLS). ADLS Gen2 is supported in Impala 3.1 and higher.

In theCREATE TABLE or ALTER TABLE statements, specify the ADLS location for tables and partitions with the adl:// prefix for ADLS Gen1 and abfs:// or abfss:// for ADLS Gen2 in the LOCATION attribute.

If you bring data into ADLS using the normal ADLS transfer mechanisms instead of Impala DML statements, issue a REFRESH statement for the table before using Impala to query the ADLS data.

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Cancellation: Can be cancelled. To cancel this statement, use Ctrl-C from the `impala-shell` interpreter, the Cancel button from the Watch page in Hue, Actions > Cancel from the Queries list in Cloudera Manager, or Cancel from the list of in-flight queries (for a particular node) on the Queries tab in the Impala web UI (port 25000).

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read permission for the files in the source directory of an INSERT ... SELECT operation, and write permission for all affected directories in the destination table. (An INSERT operation could write files to multiple different HDFS directories if the destination table is partitioned.) This user must also have write permission to create a temporary work directory in the top-level HDFS directory of the destination table. An INSERT OVERWRITE operation does not require write permission on the original data files in the table, only on the table directories themselves.

Restrictions:

For INSERT operations into CHAR or VARCHAR columns, you must cast all STRING literals or expressions returning STRING to to a CHAR or VARCHAR type with the appropriate length.

Related startup options:

By default, if an INSERT statement creates any new subdirectories underneath a partitioned table, those subdirectories are assigned default HDFS permissions for the impala user. To make each subdirectory have the same permissions as its parent directory in HDFS, specify the ##insert_inherit_permissions startup option for the `impalad` daemon.

Inserting into partitioned tables with PARTITION clause

For a partitioned table, the optional PARTITION clause identifies which partition or partitions the values are inserted into.

All examples in this section will use the table declared as below:

```
CREATE TABLE t1 (w INT) PARTITIONED BY (x INT, y STRING);
```

Static partition inserts

In a static partition insert where a partition key column is given a constant value, such as PART ITION (year=2012, month=2), the rows are inserted with the same values specified for those partition key columns.

The number of columns in the SELECT list must equal the number of columns in the column permutation.

The PARTITION clause must be used for static partitioning inserts.

Example:

The following statement will insert the some_other_table.c1 values for the w column, and all the rows inserted will have the same x value of 10, and the same y value of ‘a’.

```
INSERT INTO t1 PARTITION (x=10, y='a')
            SELECT c1 FROM some_other_table;
```

Dynamic partition inserts

In a dynamic partition insert where a partition key column is in the INSERT statement but not assigned a value, such as in PARTITION (year, region)(both columns unassigned) or PARTITIO N(year, region='CA') (year column unassigned), the unassigned columns are filled in with the final

columns of the SELECT or VALUES clause. In this case, the number of columns in the SELECT list must equal the number of columns in the column permutation plus the number of partition key columns not assigned a constant value.

The following rules apply to dynamic partition inserts.

- The columns are bound in the order they appear in the INSERT statement.

  The table below shows the values inserted with the INSERT statements of different column orders.

	Column w Value	Column x Value	Column y Value
INSERT INTO t1 (w, x, y) VALUES (1, 2, 'c');	1	2	‘c’
INSERT INTO t1 (x,w) PARTITION (y) VALUES (1, 2, 'c');	2	1	‘c’

- When a partition clause is specified but the non-partition columns are not specified in the INSERT statement, as in the first example below, the non-partition columns are treated as though they had been specified before the PARTITION clause in the SQL.

  Example: These three statements are equivalent, inserting 1 to w, 2 to x, and ‘c’ to y columns.

```
INSERT INTO t1 PARTITION (x,y) VALUES (1, 2, ‘c’);
INSERT INTO t1 (w) PARTITION (x, y) VALUES (1, 2, ‘c’);
INSERT INTO t1 PARTITION (x, y='c') VALUES (1, 2);
```

- The PARTITION clause is not required for dynamic partition, but all the partition columns must be explicitly present in the INSERT statement in the column list or in the PARTITION clause. The partition columns cannot be defaulted to NULL.

  Example:

  The following statements are valid because the partition columns, x and y, are present in the INSERT statements, either in the PARTITION clause or in the column list.

```
INSERT INTO t1 PARTITION (x,y) VALUES (1, 2, ‘c’);
INSERT INTO t1 (w, x) PARTITION (y) VALUES (1, 2, ‘c’);
```

  The following statement is not valid for the partitioned table as defined above because the partition columns, x and y, are not present in the INSERT statement.

```
INSERT INTO t1 VALUES (1, 2, 'c');
```

- If partition columns do not exist in the source table, you can specify a specific value for that column in the PARTITION clause.

  Example: The source table only contains the column w and y. The value, 20, specified in the PARTITION clause, is inserted into the x column.

```
INSERT INTO t1 PARTITION (x=20, y) SELECT * FROM source;
```

TRUNCATE on Insert-Only Transactional Tables

The TRUNCATE table statement on Insert-Only Transactional tables creates new empty ACID base directories and does not remove the files. To maintain transactional isolation TRUNCATE statement will NOT delete the previous base and delta directories.

Related Information

Reading and writing ADLS data with Impala

Examples and performance characteristics of static and dynamic partitioned inserts

INVALIDATE METADATA statement

The INVALIDATE METADATA statement marks the metadata for one or all tables as stale. The next time the Impala service performs a query against a table whose metadata is invalidated, Impala reloads the associated metadata before the query proceeds. As this is a very expensive operation compared to the incremental metadata update done by the REFRESH statement, when possible, prefer REFRESH rather than INVALIDATE METADATA.

INVALIDATE METADATA is required when the following changes are made outside of Impala, in Hive and other Hive client, such as SparkSQL:

- Metadata of existing tables changes.
- New tables are added, and Impala will use the tables.
- The SERVER or DATABASE level privileges are changed from outside of Impala.
- Block metadata changes, but the files remain the same (HDFS rebalance).
- UDF jars change.
- Some tables are no longer queried, and you want to remove their metadata from the catalog and coordinator caches to reduce memory requirements.

No INVALIDATE METADATA is needed when the changes are made by impalad.

Once issued, the INVALIDATE METADATA statement cannot be cancelled.

Syntax:

```
INVALIDATE METADATA [[DB_NAME.]TABLE_NAME]
```

If there is no table specified, the cached metadata for all tables is flushed and synced with Hive Metastore (HMS). If tables were dropped from the HMS, they will be removed from the catalog, and if new tables were added, they will show up in the catalog.

If you specify a table name, only the metadata for that one table is flushed and synced with the HMS.

Usage notes:

To return accurate query results, Impala need to keep the metadata current for the databases and tables queried. Therefore, if some other entity modifies information used by Impala in the metastore, the information cached by Impala must be updated via INVALIDATE METADATA or REFRESH.

INVALIDATE METADATA and REFRESH are counterparts:

- INVALIDATE METADATA is an asynchronous operations that simply discards the loaded metadata from the catalog and coordinator caches. After that operation, the catalog and all the Impala coordinators only know about the existence of databases and tables and nothing more. Metadata loading for tables is triggered by any subsequent queries.
- REFRESH reloads the metadata synchronously. REFRESH is more lightweight than doing a full metadata load after a table has been invalidated. REFRESH cannot detect changes in block locations triggered by operations like HDFS balancer, hence causing remote reads during query execution with negative performance implications.

Use REFRESH after invalidating a specific table to separate the metadata load from the first query that's run against that table.

Examples:

This example illustrates creating a new database and new table in Hive, then doing an INVALIDATE METADATA statement in Impala using the fully qualified table name, after which both the new table and the new database are visible to Impala.

Before the INVALIDATE METADATA statement was issued, Impala would give a “not found” error if you tried to refer to those database or table names.

```
$ hive
hive> CREATE DATABASE new_db_from_hive;
hive> CREATE TABLE new_db_from_hive.new_table_from_hive (x INT);
hive> quit;
$ impala-shell
> REFRESH new_db_from_hive.new_table_from_hive;
ERROR: AnalysisException: Database does not exist: new_db_from_hive
> INVALIDATE METADATA new_db_from_hive.new_table_from_hive;

> SHOW DATABASES LIKE 'new*';
+--------------------+
| new_db_from_hive   |
+--------------------+

> SHOW TABLES IN new_db_from_hive;
+---------------------+
| new_table_from_hive |
+---------------------+
```

Use the REFRESH statement for incremental metadata update.

```
> REFRESH new_table_from_hive;
```

HDFS considerations:

By default, the INVALIDATE METADATA command checks HDFS permissions of the underlying data files and directories, caching this information so that a statement can be cancelled immediately if for example the impala user does not have permission to write to the data directory for the table. (This checking does not apply when the `catalogd` configuration option --load_catalog_in_background is set to false, which it is by default.) Impala reports any lack of write permissions as an INFO message in the log file.

If you change HDFS permissions to make data readable or writeable by the Impala user, issue another INVALIDATE METADATA to make Impala aware of the change.

Kudu considerations:

By default, much of the metadata for Kudu tables is handled by the underlying storage layer. Kudu tables have less reliance on the Metastore database, and require less metadata caching on the Impala side. For example, information about partitions in Kudu tables is managed by Kudu, and Impala does not cache any block locality metadata for Kudu tables. If the Kudu service is not integrated with the Hive Metastore, Impala will manage Kudu table metadata in the Hive Metastore.

The REFRESH and INVALIDATE METADATA statements are needed less frequently for Kudu tables than for HDFS-backed tables. Neither statement is needed when data is added to, removed, or updated in a Kudu table, even if the changes are made directly to Kudu through a client program using the Kudu API. Run REFRESH *TABLE_NAME* or INVALIDATE METADATA *TABLE_NAME* for a Kudu table only after making a change to the Kudu table schema, such as adding or dropping a column.

LOAD DATA statement

The LOAD DATA statement streamlines the ETL process for an internal Impala table by moving a data file or all the data files in a directory from an HDFS location into the Impala data directory for that table.

Syntax:

```
LOAD DATA INPATH 'HDFS_FILE_OR_DIRECTORY_PATH' [OVERWRITE] INTO T
ABLE TABLENAME
```

```
[PARTITION (PARTCOL1=VAL1, PARTCOL2=VAL2 ...)]
```

When the LOAD DATA statement operates on a partitioned table, it always operates on one partition at a time. Specify the PARTITION clauses and list all the partition key columns, with a constant value specified for each.

Statement type: DML (but still affected by the SYNC_DDL query option)

Usage notes:

- The loaded data files are moved, not copied, into the Impala data directory.
- You can specify the HDFS path of a single file to be moved, or the HDFS path of a directory to move all the files inside that directory. You cannot specify any sort of wildcard to take only some of the files from a directory. When loading a directory full of data files, keep all the data files at the top level, with no nested directories underneath.
- Currently, the Impala LOAD DATA statement only imports files from HDFS, not from the local filesystem. It does not support the LOCAL keyword of the Hive LOAD DATA statement. You must specify a path, not an hdfs :// URI.
- In the interest of speed, only limited error checking is done. If the loaded files have the wrong file format, different columns than the destination table, or other kind of mismatch, Impala does not raise any error for the LOAD DATA statement. Querying the table afterward could produce a runtime error or unexpected results. Currently, the only checking the LOAD DATA statement does is to avoid mixing together uncompressed and LZO-compressed text files in the same table.
- When you specify an HDFS directory name as the LOAD DATA argument, any hidden files in that directory (files whose names start with a .) are not moved to the Impala data directory.
- The operation fails if the source directory contains any non-hidden directories. Prior to Impala 2.5 if the source directory contained any subdirectory, even a hidden one such as _impala_insert_staging, the LOAD DATA statement would fail. In Impala 2.5 and higher, LOAD DATA ignores hidden subdirectories in the source directory, and only fails if any of the subdirectories are non-hidden.
- The loaded data files retain their original names in the new location, unless a name conflicts with an existing data file, in which case the name of the new file is modified slightly to be unique. (The name-mangling is a slight difference from the Hive LOAD DATA statement, which replaces identically named files.)
- By providing an easy way to transport files from known locations in HDFS into the Impala data directory structure, the LOAD DATA statement lets you avoid memorizing the locations and layout of HDFS directory tree containing the Impala databases and tables. (For a quick way to check the location of the data files for an Impala table, issue the statement DESCRIBE FORMATTED *TABLE_NAME*.)
- The PARTITION clause is especially convenient for ingesting new data for a partitioned table. As you receive new data for a time period, geographic region, or other division that corresponds to one or more partitioning columns, you can load that data straight into the appropriate Impala data directory, which might be nested several levels down if the table is partitioned by multiple columns. When the table is partitioned, you must specify constant values for all the partitioning columns.

Complex type considerations:

Because Impala currently cannot create Parquet data files containing complex types (ARRAY, STRUCT, and MAP), the LOAD DATA statement is especially important when working with tables containing complex type columns. You create the Parquet data files outside Impala, then use either LOAD DATA, an external table, or HDFS-level file operations followed by REFRESH to associate the data files with the corresponding table.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMPUTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

First, we use a trivial Python script to write different numbers of strings (one per line) into files stored in the doc_demo HDFS user account. (Substitute the path for your own HDFS user account when doing `hdfs dfs` operations like these.)

```
random_strings.py 1000 | hdfs dfs -put - /user/doc_demo/thousand_strings.txt
random_strings.py 100 | hdfs dfs -put - /user/doc_demo/hundred_strings.txt
random_strings.py 10 | hdfs dfs -put - /user/doc_demo/ten_strings.txt
```

Next, we create a table and load an initial set of data into it. Remember, unless you specify a STORED AS clause, Impala tables default to TEXTFILE format with Ctrl-A (hex 01) as the field delimiter. This example uses a single-column table, so the delimiter is not significant. For large-scale ETL jobs, you would typically use binary format data files such as Parquet or Avro, and load them into Impala tables that use the corresponding file format.

```
[localhost:21000] > create table t1 (s string);
[localhost:21000] > load data inpath '/user/doc_demo/thousand_strings.txt' i
nto table t1;
Query finished, fetching results ...
+----------------------------------------------------------+
| summary                                                  |
+----------------------------------------------------------+
| Loaded 1 file(s). Total files in destination location: 1 |
+----------------------------------------------------------+
Returned 1 row(s) in 0.61s
[kilo2-202-961.cs1cloud.internal:21000] > select count(*) from t1;
Query finished, fetching results ...
+------+
| _c0  |
+------+
| 1000 |
+------+
Returned 1 row(s) in 0.67s
[localhost:21000] > load data inpath '/user/doc_demo/thousand_strings.txt'
 into table t1;
ERROR: AnalysisException: INPATH location '/user/doc_demo/thousand_strings.
txt' does not exist.
```

As indicated by the message at the end of the previous example, the data file was moved from its original location. The following example illustrates how the data file was moved into the Impala data directory for the destination table, keeping its original filename:

```
$ hdfs dfs -ls /user/hive/warehouse/load_data_testing.db/t1
Found 1 items
-rw-r--r--   1 doc_demo doc_demo      13926 2013-06-26 15:40 /user/hive/wa
rehouse/load_data_testing.db/t1/thousand_strings.txt
```

The following example demonstrates the difference between the INTO TABLE and OVERWRITE TABLE clauses. The table already contains 1000 rows. After issuing the LOAD DATA statement with the INTO TABLE clause, the table contains 100 more rows, for a total of 1100. After issuing the LOAD DATA statement with the OVERWRITE INTO TABLE clause, the former contents are gone, and now the table only contains the 10 rows from the just-loaded data file.

```
[localhost:21000] > load data inpath '/user/doc_demo/hundred_strings.txt' in
to table t1;
Query finished, fetching results ...
+----------------------------------------------------------+
| summary                                                  |
+----------------------------------------------------------+
| Loaded 1 file(s). Total files in destination location: 2 |
+----------------------------------------------------------+
Returned 1 row(s) in 0.24s
[localhost:21000] > select count(*) from t1;
```

```
Query finished, fetching results ...
+------+
| _c0  |
+------+
| 1100 |
+------+
Returned 1 row(s) in 0.55s
[localhost:21000] > load data inpath '/user/doc_demo/ten_strings.txt' overwr
ite into table t1;
Query finished, fetching results ...
+-----------------------------------------------------------+
| summary                                                   |
+-----------------------------------------------------------+
| Loaded 1 file(s). Total files in destination location: 1 |
+-----------------------------------------------------------+
Returned 1 row(s) in 0.26s
[localhost:21000] > select count(*) from t1;
Query finished, fetching results ...
+-----+
| _c0 |
+-----+
| 10  |
+-----+
Returned 1 row(s) in 0.62s
```

Amazon S3 considerations:

In Impala 2.6 and higher, the Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in the Amazon Simple Storage Service (S3). The syntax of the DML statements is the same as for any other tables, because the S3 location for tables and partitions is specified by an s3a:// prefix in the LOCATION attribute of CREATE TABLE or ALTER TABLE statements. If you bring data into S3 using the normal S3 transfer mechanisms instead of Impala DML statements, issue a REFRESH statement for the table before using Impala to query the S3 data.

Because of differences between S3 and traditional filesystems, DML operations for S3 tables can take longer than for tables on HDFS. For example, both the LOAD DATA statement and the final stage of the INSERT and CREATE TABLE AS SELECT statements involve moving files from one directory to another. (In the case of INSERT and CREATE TABLE AS SELECT, the files are moved from a temporary staging directory to the final destination directory.) Because S3 does not support a “rename” operation for existing objects, in these cases Impala actually copies the data files from one location to another and then removes the original files. In Impala 2.6, the S3_SKIP_INSERT_STAGING query option provides a way to speed up INSERT statements for S3 tables and partitions, with the tradeoff that a problem during statement execution could leave data in an inconsistent state. It does not apply to INSERT OVERWRITE or LOAD DATA statements.

ADLS considerations:

In Impala 2.9 and higher, the Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in the Azure Data Lake Store (ADLS). ADLS Gen2 is supported in Impala 3.1 and higher.

In theCREATE TABLE or ALTER TABLE statements, specify the ADLS location for tables and partitions with the adl:// prefix for ADLS Gen1 and abfs:// or abfss:// for ADLS Gen2 in the LOCATION attribute.

If you bring data into ADLS using the normal ADLS transfer mechanisms instead of Impala DML statements, issue a REFRESH statement for the table before using Impala to query the ADLS data.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and write permissions for the files in the source directory, and write permission for the destination directory.

Kudu considerations:

The LOAD DATA statement cannot be used with Kudu tables.

HBase considerations:

The LOAD DATA statement cannot be used with HBase tables.

Related information:

The LOAD DATA statement is an alternative to the INSERT statement. Use LOAD DATA when you have the data files in HDFS but outside of any Impala table.

The LOAD DATA statement is also an alternative to the CREATE EXTERNAL TABLE statement. Use LOAD DATA when it is appropriate to move the data files under Impala control rather than querying them from their original location.

Related Information

Reading and writing ADLS data with Impala

REFRESH statement

The REFRESH statement reloads the metadata for the table from the metastore database and does an incremental reload of the file and block metadata from the HDFS NameNode. REFRESH is used to avoid inconsistencies between Impala and external metadata sources, namely Hive Metastore (HMS) and NameNodes.

The REFRESH statement is only required if you load data from outside of Impala. Updated metadata, as a result of running REFRESH, is broadcast to all Impala coordinators.

Syntax:

```
REFRESH [DB_NAME.]TABLE_NAME [PARTITION (KEY_COL1=VAL1 [, KEY_COL2=VAL2...]
)]
```

Usage notes:

The table name is a required parameter, and the table must already exist and be known to Impala.

Only the metadata for the specified table is reloaded.

Use the REFRESH statement to load the latest metastore metadata for a particular table after one of the following scenarios happens outside of Impala:

- Deleting, adding, or modifying files.

  For example, after loading new data files into the HDFS data directory for the table, appending to an existing HDFS file, inserting data from Hive via INSERT or LOAD DATA.
- Deleting, adding, or modifying partitions.

  For example, after issuing ALTER TABLE or other table-modifying SQL statement in Hive

Note:

In Impala 2.3 and higher, the ALTER TABLE *TABLE_NAME* RECOVER PARTITIONS statement maybe a faster alternative to REFRESH in some scenarios such as when you are only adding new partition directories through Hive or manual HDFS operations however the REFRESH query will NOT detect the new partitions unless the new partitions are in HMS already. You must be aware of the following limitations before you choose between REFRESH and ALTER TABLE.

- REFRESH does not detect new partition directories that are added by HDFS.
- RECOVER PARTITIONS does not detect changes in already existing partitions.
- RECOVER PARTITIONS can cause performance issues if SYNC_DDL=1.

INVALIDATE METADATA and REFRESH are counterparts:

- INVALIDATE METADATA is an asynchronous operations that simply discards the loaded metadata from the catalog and coordinator caches. After that operation, the catalog and all the Impala coordinators only know about the existence of databases and tables and nothing more. Metadata loading for tables is triggered by any subsequent queries.
- REFRESH reloads the metadata synchronously. REFRESH is more lightweight than doing a full metadata load after a table has been invalidated. REFRESH cannot detect changes in block locations triggered by operations like HDFS balancer, hence causing remote reads during query execution with negative performance implications.

Refreshing a single partition:

In Impala 2.7 and higher, the REFRESH statement can apply to a single partition at a time, rather than the whole table. Include the optional PARTITION (*PARTITION_SPEC*) clause and specify values for each of the partition key columns.

The following rules apply:

- The PARTITION clause of the REFRESH statement must include all the partition key columns.
- The order of the partition key columns does not have to match the column order in the table.
- Specifying a nonexistent partition does not cause an error.
- The partition can be one that Impala created and is already aware of, or a new partition created through Hive.

The following examples demonstrates the above rules.

```
-- Partition doesn't exist.
REFRESH p2 PARTITION (y=0, z=3);
REFRESH p2 PARTITION (y=0, z=-1)

-- Key columns specified in a different order than the table definition.
REFRESH p2 PARTITION (z=1, y=0)

-- Incomplete partition spec causes an error.
REFRESH p2 PARTITION (y=0)
ERROR: AnalysisException: Items in partition spec must exactly match the
partition columns in the table definition: default.p2 (1 vs 2)
```

Related impala-shell options:

Due to the expense of reloading the metadata for all tables, the `impala-shell` -r option is not recommended.

HDFS considerations:

All HDFS and Ranger permissions and privilege requirements are the same whether you refresh the entire table or a single partition.

The REFRESH statement checks HDFS permissions of the underlying data files and directories, caching this information so that a statement can be cancelled immediately if for example the impala user does not have permission to write to the data directory for the table. Impala reports any lack of write permissions as an INFO message in the log file.

If you change HDFS permissions to make data readable or writeable by the Impala user, issue another REFRESH to make Impala aware of the change.

Kudu considerations:

By default, much of the metadata for Kudu tables is handled by the underlying storage layer. Kudu tables have less reliance on the Metastore database, and require less metadata caching on the Impala side. For example, information about partitions in Kudu tables is managed by Kudu, and Impala does not cache any block locality metadata for Kudu tables. If the Kudu service is not integrated with the Hive Metastore, Impala will manage Kudu table metadata in the Hive Metastore.

The REFRESH and INVALIDATE METADATA statements are needed less frequently for Kudu tables than for HDFS-backed tables. Neither statement is needed when data is added to, removed, or updated in a Kudu table, even if

the changes are made directly to Kudu through a client program using the Kudu API. Run REFRESH *TABLE_NAME* or INVALIDATE METADATA *TABLE_NAME* for a Kudu table only after making a change to the Kudu table schema, such as adding or dropping a column.

REFRESH AUTHORIZATION statement

The REFRESH AUTHORIZATION statement explicitly refreshes authorization data, including privileges and principals. When there is an external update to authorization metadata, use this statement to force Impala to refresh its authorization data without having to wait for the Ranger polling or run INVALIDATE METADATA.

Once issued, the REFRESH AUTHORIZATION statement cannot be cancelled.

Syntax:

```
REFRESH AUTHORIZATION
```

Usage notes: If authorization is not enabled, Impala returns an error.

Added in: Impala 3.2

Related Information

Enabling and using authorization in Impala

REFRESH FUNCTIONS statement

In Impala 2.9 and higher, you can run the REFRESH FUNCTIONS statement to refresh the user-defined functions (UDFs) created outside of Impala. For example, you can add Java-based UDFs to the metastore database through the Hive CREATE FUNCTION statements and make those UDFs visible to Impala at the database level by subsequently running REFRESH FUNCTIONS.

The REFRESH FUNCTIONS statement is only required if you create or modify UDFs from outside of Impala. Updated metadata, as a result of running REFRESH FUNCTIONS, is broadcast to all Impala coordinators.

Once issued, the REFRESH FUNCTIONS statement cannot be cancelled.

Syntax:

REFRESH FUNCTIONS *DB_NAME*

REVOKE statement

The REVOKE statement revokes privileges on a specified object from groups or users.

Syntax:

```
      REVOKE PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
      FROM USER USER_NAME
      REVOKE PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
      FROM GROUP GROUP_NAME

        privilege ::= ALL | ALTER | CREATE | DROP | INSERT | REFRESH |
SELECT | SELECT(COLUMN_NAME)

      object_type ::= SERVER | URI | DATABASE | TABLE
```

See the GRANT statement for the required privileges and the scope for SQL operations.

The ALL privilege is a distinct privilege and not a union of all other privileges.

The following syntax is supported when Impala is using Ranger to manage authorization.

```
REVOKE PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
  FROM USER USER_NAME

REVOKE PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
  FROM GROUP GROUP_NAME


  privilege ::= ALL | ALTER | CREATE | DROP | INSERT | REFRESH | SELECT |
 SELECT(COLUMN_NAME)

object_type ::= SERVER | URI | DATABASE | TABLE
```

You cannot revoke a privilege granted with the WITH GRANT OPTION. If a privilege is granted with the WITH GRANT OPTION, first revoke the grant option, and then revoke the privilege.

Typically, the object name is an identifier. For URIs, it is a string literal.

The ability to grant or revoke SELECT privilege on specific columns is available in Impala 2.3 and higher.

Required privileges:

Only administrative users those with ALL privileges on the server can use this statement.

Compatibility:

- The Impala REVOKE statement is available in Impala 2.0 and higher.
- In Impala 1.4 and higher, Impala makes use of any privileges specified by the GRANT and REVOKE statements in Hive, when your system is configured to use the Ranger service.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Related Information

GRANT statement

REVOKE ROLE statement

The REVOKE statement REMOVES the role assigned to the group.

Syntax

```
REVOKE ROLE ROLE_NAME FROM GROUP GROUP_NAME
```

SELECT statement

The SELECT statement performs queries, retrieving data from one or more tables and producing result sets consisting of rows and columns.

The Impala INSERT statement also typically ends with a SELECT statement, to define data to copy from one table to another.

Syntax:

```
[WITH name AS (select_expression) [, ...] ]
SELECT
  [ALL | DISTINCT]
  [STRAIGHT_JOIN]
  expression [, expression ...]
FROM table_reference [, table_reference ...]
[[FULL | [LEFT | RIGHT] INNER | [LEFT | RIGHT] OUTER | [LEFT | RIGHT] SEMI |
 [LEFT | RIGHT] ANTI | CROSS]
  JOIN table_reference
  [ON join_equality_clauses | USING (COL1[, COL2 ...]] ...
WHERE conditions
GROUP BY { column | expression [, ...] }
HAVING conditions
ORDER BY { column | expression [ASC | DESC] [NULLS FIRST | NULLS LAST] [,
...] }
LIMIT expression [OFFSET expression]
[UNION [ALL] select_statement] ...]

table_reference := { TABLE_NAME | (SUBQUERY) }
  [ TABLESAMPLE SYSTEM(PERCENTAGE) [REPEATABLE(SEED)] ]
```

Impala SELECT queries support:

- SQL scalar data types: BOOLEAN, TINYINT, SMALLINT, INT, BIGINT, DECIMAL FLOAT, DOUBLE, TIMESTAMP, STRING, VARCHAR, CHAR.
- The complex data types ARRAY, STRUCT, and MAP, are available in Impala 2.3 and higher. Queries involving these types typically involve special qualified names using dot notation for referring to the complex column fields, and join clauses for bringing the complex columns into the result set.
- An optional WITH clause before the SELECT keyword, to define a subquery whose name or column names can be referenced from later in the main query. This clause lets you abstract repeated clauses, such as aggregation functions, that are referenced multiple times in the same query.
- Subqueries in a FROM clause. In Impala 2.0 and higher, subqueries can also go in the WHERE clause, for example with the IN(), EXISTS, and NOT EXISTS operators.
- WHERE, GROUP BY, HAVING clauses.
- ORDER BY. Prior to Impala 1.4.0, Impala required that queries using an ORDER BY clause also include a LIMIT clause. In Impala 1.4.0 and higher, this restriction is lifted; sort operations that would exceed the Impala memory limit automatically use a temporary disk work area to perform the sort.
- You can refer to SELECT-list items by their ordinal position. Impala supports ordinals in the GROUP BY, HAVI NG, and ORDER BY clauses. From Impala 3.0, ordinals can only be used at the top level. For example, the following statements are allowed:

  ```
  SELECT int_col / 2, sum(x)
  FROM t
  GROUP BY 1;

  SELECT int_col / 2
  FROM t
  ORDER BY 1;

  SELECT NOT bool_col
  FROM t
  GROUP BY 1
  HAVING 1;
  ```

  Numbers in subexpressions are not interpreted as ordinals:

```
  SELECT int_col / 2, sum(x)
  FROM t
  GROUP BY 1 * 2;
The above parses OK, however GROUP BY 1 * 2 has no effect.
  SELECT int_col / 2
  FROM t
  ORDER BY 1 + 2;
The above parses OK, however ORDER BY 1 + 2 has no effect.
  SELECT NOT bool_col
  FROM t
  GROUP BY 1
  HAVING not 1;
The above raises an error at parse-time.
```

- Impala supports a wide variety of JOIN clauses. Left, right, semi, full, and outer joins are supported in all Impala versions. The CROSS JOIN operator is available in Impala 1.2.2 and higher. During performance tuning, you can override the reordering of join clauses that Impala does internally by including the keyword STRAIGHT_JOIN immediately after the SELECT and any DISTINCT or ALL keywords.
- UNION ALL.
- LIMIT.
- External tables.
- Relational operators such as greater than, less than, or equal to.
- Arithmetic operators such as addition or subtraction.
- Logical/Boolean operators AND, OR, and NOT. Impala does not support the corresponding symbols &&, ||, and !.
- Common SQL built-in functions such as COUNT, SUM, CAST, LIKE, IN, BETWEEN, and COALESCE.
- On optional TABLESAMPLE clause immediately after a table reference, to specify that the query only processes a specified percentage of the table data.

Impala queries ignore files with extensions commonly used for temporary work files by Hadoop tools. Any files with extensions .tmp or .copying are not considered part of the Impala table. The suffix matching is case-insensitive, so for example Impala ignores both .copying and .COPYING suffixes.

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Amazon S3 considerations:

In Impala 2.6 and higher, Impala queries are optimized for files stored in Amazon S3. For Impala tables that use the file formats Parquet, ORC, RCFile, SequenceFile, Avro, and uncompressed text, the setting fs.s3a.block.size in the core-site.xml configuration file determines how Impala divides the I/O work of reading the data files. This configuration setting is specified in bytes. By default, this value is 33554432 (32 MB), meaning that Impala parallelizes S3 read operations on the files as if they were made up of 32 MB blocks. For example, if your S3 queries primarily access Parquet files written by MapReduce or Hive, increase fs.s3a.block.size to 134217728 (128 MB) to match the row group size of those files. If most S3 queries involve Parquet files written by Impala, increase fs.s3a.block.size to 268435456 (256 MB) to match the row group size produced by Impala.

Cancellation: Can be cancelled. To cancel this statement, use Ctrl-C from the `impala-shell` interpreter, the Cancel button from the Watch page in Hue, Actions > Cancel from the Queries list in Cloudera Manager, or Cancel from the list of in-flight queries (for a particular node) on the Queries tab in the Impala web UI (port 25000).

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read permissions for the files in all applicable directories in all source tables, and read and execute permissions for the relevant data directories. (A SELECT operation could read files from multiple different HDFS directories if the source table is partitioned.) If a query attempts to read a data file and is unable to because of an HDFS permission error, the query halts and does not return any further results.

Related information:

The SELECT syntax is so extensive that it forms its own category of statements: queries. The other major classifications of SQL statements are data definition language and data manipulation language.

Because the focus of Impala is on fast queries with interactive response times over huge data sets, query performance and scalability are important considerations.

Joins in Impala SELECT statements

A join query is a SELECT statement that combines data from two or more tables, and returns a result set containing items from some or all of those tables. It is a way to cross-reference and correlate related data that is organized into multiple tables, typically using identifiers that are repeated in each of the joined tables.

Syntax:

Impala supports a wide variety of JOIN clauses. Left, right, semi, full, and outer joins are supported in all Impala versions. The CROSS JOIN operator is available in Impala 1.2.2 and higher. During performance tuning, you can override the reordering of join clauses that Impala does internally by including the keyword STRAIGHT_JOIN immediately after the SELECT and any DISTINCT or ALL keywords.

```
SELECT SELECT_LIST FROM
  TABLE_OR_SUBQUERY1 [INNER] JOIN TABLE_OR_SUBQUERY2 |
  TABLE_OR_SUBQUERY1 {LEFT [OUTER] | RIGHT [OUTER] | FULL [OUTER]} J
OIN TABLE_OR_SUBQUERY2 |
  TABLE_OR_SUBQUERY1 {LEFT | RIGHT} SEMI JOIN TABLE_OR_SUBQUERY2 |
  TABLE_OR_SUBQUERY1 {LEFT | RIGHT} ANTI JOIN TABLE_OR_SUBQUERY2 |
    [ ON COL1 = COL2 [AND COL3 = COL4 ...] |
      USING (COL1 [, COL2 ...]) ]
  [OTHER_JOIN_CLAUSE ...]
[ WHERE WHERE_CLAUSES ]

SELECT SELECT_LIST FROM
  TABLE_OR_SUBQUERY1, TABLE_OR_SUBQUERY2 [, TABLE_OR_SUBQUERY3 ...]
  [OTHER_JOIN_CLAUSE ...]
WHERE
    COL1 = COL2 [AND COL3 = COL4 ...]

SELECT SELECT_LIST FROM
  TABLE_OR_SUBQUERY1 CROSS JOIN TABLE_OR_SUBQUERY2
  [OTHER_JOIN_CLAUSE ...]
[ WHERE WHERE_CLAUSES ]
```

SQL-92 and SQL-89 Joins:

Queries with the explicit JOIN keywords are known as SQL-92 style joins, referring to the level of the SQL standard where they were introduced. The corresponding ON or USING clauses clearly show which columns are used as the join keys in each case:

```
SELECT t1.c1, t2.c2 FROM t1 JOIN t2
  ON t1.id = t2.id and t1.type_flag = t2.type_flag
  WHERE t1.c1 > 100;

SELECT t1.c1, t2.c2 FROM t1 JOIN t2
  USING (id, type_flag)
  WHERE t1.c1 > 100;
```

The ON clause is a general way to compare columns across the two tables, even if the column names are different. The USING clause is a shorthand notation for specifying the join columns, when the column names are the same in both tables. You can code equivalent WHERE clauses that compare the columns, instead of ON or USING clauses, but that practice is not recommended because mixing the join comparisons with other filtering clauses is typically less readable and harder to maintain.

Queries with a comma-separated list of tables and subqueries are known as SQL-89 style joins. In these queries, the equality comparisons between columns of the joined tables go in the WHERE clause alongside other kinds of comparisons. This syntax is easy to learn, but it is also easy to accidentally remove a WHERE clause needed for the join to work correctly.

```
SELECT t1.c1, t2.c2 FROM t1, t2
  WHERE
  t1.id = t2.id AND t1.type_flag = t2.type_flag
  AND t1.c1 > 100;
```

Self-joins:

Impala can do self-joins, for example to join on two different columns in the same table to represent parent-child relationships or other tree-structured data. There is no explicit syntax for this; just use the same table name for both the left-hand and right-hand table, and assign different table aliases to use when referring to the fully qualified column names:

```
-- Combine fields from both parent and child rows.
SELECT lhs.id, rhs.parent, lhs.c1, rhs.c2 FROM tree_data lhs, tree_data rhs
 WHERE lhs.id = rhs.parent;
```

Inner and outer joins:

An inner join is the most common and familiar type: rows in the result set contain the requested columns from the appropriate tables, for all combinations of rows where the join columns of the tables have identical values. If a column with the same name occurs in both tables, use a fully qualified name or a column alias to refer to the column in the select list or other clauses. Impala performs inner joins by default for both SQL-89 and SQL-92 join syntax:

```
-- The following 3 forms are all equivalent.
SELECT t1.id, c1, c2 FROM t1, t2 WHERE t1.id = t2.id;
SELECT t1.id, c1, c2 FROM t1 JOIN t2 ON t1.id = t2.id;
SELECT t1.id, c1, c2 FROM t1 INNER JOIN t2 ON t1.id = t2.id;
```

An outer join retrieves all rows from the left-hand table, or the right-hand table, or both; wherever there is no matching data in the table on the other side of the join, the corresponding columns in the result set are set to NULL. To perform an outer join, include the OUTER keyword in the join operator, along with either LEFT, RIGHT, or FULL:

```
SELECT * FROM t1 LEFT OUTER JOIN t2 ON t1.id = t2.id;
SELECT * FROM t1 RIGHT OUTER JOIN t2 ON t1.id = t2.id;
SELECT * FROM t1 FULL OUTER JOIN t2 ON t1.id = t2.id;
```

For outer joins, Impala requires SQL-92 syntax; that is, the JOIN keyword instead of comma-separated table names. Impala does not support vendor extensions such as (+) or *= notation for doing outer joins with SQL-89 query syntax.

Equijoins and Non-Equijoins:

By default, Impala requires an equality comparison between the left-hand and right-hand tables, either through ON, USING, or WHERE clauses. These types of queries are classified broadly as equijoins. Inner, outer, full, and semi joins can all be equijoins based on the presence of equality tests between columns in the left-hand and right-hand tables.

In Impala 1.2.2 and higher, non-equijoin queries are also possible, with comparisons such as != or < between the join columns. These kinds of queries require care to avoid producing huge result sets that could exceed resource limits. Once you have planned a non-equijoin query that produces a result set of acceptable size, you can code the query using the CROSS JOIN operator, and add the extra comparisons in the WHERE clause:

```
SELECT * FROM t1 CROSS JOIN t2 WHERE t1.total > t2.maximum_price;
```

In Impala 2.3 and higher, additional non-equijoin queries are possible due to the addition of nested loop joins. These queries typically involve SEMI JOIN, ANTI JOIN, or FULL OUTER JOIN clauses. Impala sometimes also uses nested loop joins internally when evaluating OUTER JOIN queries involving complex type columns. Query phases involving nested loop joins do not use the spill-to-disk mechanism if they exceed the memory limit. Impala decides internally when to use each join mechanism; you cannot specify any query hint to choose between the nested loop join or the original hash join algorithm.

```
SELECT * FROM t1 LEFT OUTER JOIN t2 ON t1.int_col < t2.int_col;
```

Semi-joins:

Semi-joins are a relatively rarely used variation. With the left semi-join, only data from the left-hand table is returned, for rows where there is matching data in the right-hand table, based on comparisons between join columns in ON or WHERE clauses. Only one instance of each row from the left-hand table is returned, regardless of how many matching rows exist in the right-hand table. A right semi-join (available in Impala 2.0 and higher) reverses the comparison and returns data from the right-hand table.

```
SELECT t1.c1, t1.c2, t1.c2 FROM t1 LEFT SEMI JOIN t2 ON t1.id = t2.id;
```

Natural joins (not supported):

Impala does not support the NATURAL JOIN operator, again to avoid inconsistent or huge result sets. Natural joins do away with the ON and USING clauses, and instead automatically join on all columns with the same names in the left-hand and right-hand tables. This kind of query is not recommended for rapidly evolving data structures such as are typically used in Hadoop. Thus, Impala does not support the NATURAL JOIN syntax, which can produce different query results as columns are added to or removed from tables.

If you do have any queries that use NATURAL JOIN, make sure to rewrite them with explicit USING clauses, because Impala could interpret the NATURAL keyword as a table alias:

```
-- 'NATURAL' is interpreted as an alias for 't1' and Impala attempts an inne
r join,
-- resulting in an error because inner joins require explicit comparisons be
tween columns.
SELECT t1.c1, t2.c2 FROM t1 NATURAL JOIN t2;
ERROR: NotImplementedException: Join with 't2' requires at least one conjunc
tive equality predicate.
  To perform a Cartesian product between two tables, use a CROSS JOIN.
-- If you expect the tables to have identically named columns with matching
 values,
-- list the corresponding column names in a USING clause.
SELECT t1.c1, t2.c2 FROM t1 JOIN t2 USING (id, type_flag, name, address);
```

Anti-joins (Impala 2.0 and higher only):

Impala supports the LEFT ANTI JOIN and RIGHT ANTI JOIN clauses in Impala 2.0 and higher. The LEFT or RIGHT keyword is required for this kind of join. For LEFT ANTI JOIN, this clause returns those values from the left-hand table that have no matching value in the right-hand table. RIGHT ANTI JOIN reverses the comparison and returns values from the right-hand table. You can express this negative relationship either through the ANTI JOIN clause or through a NOT EXISTS operator with a subquery.

Complex type considerations:

When referring to a column with a complex type (STRUCT, ARRAY, or MAP) in a query, you use join notation to “unpack” the scalar fields of the struct, the elements of the array, or the key-value pairs of the map. (The join notation is not required for aggregation operations, such as COUNT() or SUM() for array elements.) Because Impala recognizes which complex type elements are associated with which row of the result set, you use the same syntax as for a cross or cartesian join, without an explicit join condition.

Usage notes:

You typically use join queries in situations like these:

- When related data arrives from different sources, with each data set physically residing in a separate table. For example, you might have address data from business records that you cross-check against phone listings or census data.

  Note: Impala can join tables of different file formats, including Impala-managed tables and HBase tables. For example, you might keep small dimension tables in HBase, for convenience of single-row lookups and updates, and for the larger fact tables use Parquet or other binary file format optimized for scan operations. Then, you can issue a join query to cross-reference the fact tables with the dimension tables.
- When data is normalized, a technique for reducing data duplication by dividing it across multiple tables. This kind of organization is often found in data that comes from traditional relational database systems. For example, instead of repeating some long string such as a customer name in multiple tables, each table might contain a numeric customer ID. Queries that need to display the customer name could "join" the table that specifies which customer ID corresponds to which name.
- When certain columns are rarely needed for queries, so they are moved into separate tables to reduce overhead for common queries. For example, a biography field might be rarely needed in queries on employee data. Putting that field in a separate table reduces the amount of I/O for common queries on employee addresses or phone numbers. Queries that do need the biography column can retrieve it by performing a join with that separate table.
- In Impala 2.3 or higher, when referring to complex type columns in queries.

When comparing columns with the same names in ON or WHERE clauses, use the fully qualified names such as *DB_NAME.TABLE_NAME*, or assign table aliases, column aliases, or both to make the code more compact and understandable:

```
select t1.c1 as first_id, t2.c2 as second_id from
  t1 join t2 on first_id = second_id;
select fact.custno, dimension.custno from
  customer_data as fact join customer_address as dimension
  using (custno)
```

Note:

Performance for join queries is a crucial aspect for Impala, because complex join queries are resource-intensive operations. An efficient join query produces much less network traffic and CPU overhead than an inefficient one. For best results:

- Make sure that both table and column statistics are available for all the tables involved in a join query, and especially for the columns referenced in any join conditions. Impala uses the statistics to automatically deduce an efficient join order. Use SHOW TABLE STATS *TABLE_NAME* and SHOW COLUMN STATS *TABLE_NAME* to check if statistics are already present. Issue the COMPUTE STATS *TABLE_NAME* for a nonpartitioned table, or (in Impala 2.1.0 and higher) COMPUTE INCREMENTAL STATS *TABLE_NAME* for a partitioned table, to collect the initial statistics at both the table and column levels, and to keep the statistics up to date after any substantial INSERT or LOAD DATA operations.
- If table or column statistics are not available, join the largest table first. You can check the existence of statistics with the SHOW TABLE STATS *TABLE_NAME* and SHOW COLUMN STATS *TABLE_NAME* statements.
- If table or column statistics are not available, join subsequent tables according to which table has the most selective filter, based on overall size and WHERE clauses. Joining the table with the most selective filter results in the fewest number of rows being returned.

To control the result set from a join query, include the names of corresponding column names in both tables in an ON or USING clause, or by coding equality comparisons for those columns in the WHERE clause.

```
[localhost:21000] > select c_last_name, ca_city from customer join customer_
address where c_customer_sk = ca_address_sk;
+-------------+-----------------+
| c_last_name | ca_city         |
+-------------+-----------------+
| Lewis       | Fairfield       |
```

```
| Moses        | Fairview         |
| Hamilton     | Pleasant Valley |
| White        | Oak Ridge        |
| Moran        | Glendale         |
...
| Richards     | Lakewood          |
| Day          | Lebanon           |
| Painter      | Oak Hill          |
| Bentley      | Greenfield        |
| Jones        | Stringtown        |
+-------------+------------------+
Returned 50000 row(s) in 9.82s
```

One potential downside of joins is the possibility of excess resource usage in poorly constructed queries. Impala imposes restrictions on join queries to guard against such issues. To minimize the chance of runaway queries on large data sets, Impala requires every join query to contain at least one equality predicate between the columns of the various tables. For example, if T1 contains 1000 rows and T2 contains 1,000,000 rows, a query SELECT *COLUMNS* FROM t1 JOIN t2 could return up to 1 billion rows (1000 * 1,000,000); Impala requires that the query include a clause such as ON t1.c1 = t2.c2 or WHERE t1.c1 = t2.c2.

Because even with equality clauses, the result set can still be large, as we saw in the previous example, you might use a LIMIT clause to return a subset of the results:

```
[localhost:21000] > select c_last_name, ca_city from customer, customer_addr
ess where c_customer_sk = ca_address_sk limit 10;
+-------------+-----------------+
| c_last_name | ca_city         |
+-------------+-----------------+
| Lewis       | Fairfield       |
| Moses       | Fairview        |
| Hamilton    | Pleasant Valley |
| White       | Oak Ridge       |
| Moran       | Glendale        |
| Sharp       | Lakeview        |
| Wiles       | Farmington      |
| Shipman     | Union           |
| Gilbert     | New Hope        |
| Brunson     | Martinsville    |
+-------------+-----------------+
Returned 10 row(s) in 0.63s
```

Or you might use additional comparison operators or aggregation functions to condense a large result set into a smaller set of values:

```
[localhost:21000] > -- Find the names of customers who live in one particula
r town.
[localhost:21000] > select distinct c_last_name from customer, customer_addr
ess where
  c_customer_sk = ca_address_sk
  and ca_city = "Green Acres";
+---------------+
| c_last_name   |
+---------------+
| Hensley       |
| Pearson       |
| Mayer         |
| Montgomery    |
| Ricks         |
...
| Barrett       |
| Price         |
| Hill          |
```

```
| Hansen        |
| Meeks         |
+---------------+
Returned 332 row(s) in 0.97s

[localhost:21000] > -- See how many different customers in this town have
 names starting with "A".
[localhost:21000] > select count(distinct c_last_name) from customer, cust
omer_address where
  c_customer_sk = ca_address_sk
  and ca_city = "Green Acres"
  and substr(c_last_name,1,1) = "A";
+-----------------------------+
| count(distinct c_last_name) |
+-----------------------------+
| 12                          |
+-----------------------------+
Returned 1 row(s) in 1.00s
```

Because a join query can involve reading large amounts of data from disk, sending large amounts of data across the network, and loading large amounts of data into memory to do the comparisons and filtering, you might do benchmarking, performance analysis, and query tuning to find the most efficient join queries for your data set, hardware capacity, network configuration, and cluster workload.

The two categories of joins in Impala are known as partitioned joins and broadcast joins. If inaccurate table or column statistics, or some quirk of the data distribution, causes Impala to choose the wrong mechanism for a particular join, consider using query hints as a temporary workaround.

Handling NULLs in join columns:

By default, join key columns do not match if either one contains a NULL value. To treat such columns as equal if both contain NULL, you can use an expression such as A = B OR (A IS NULL AND B IS NULL). In Impala 2.5 and higher, the <=> operator (shorthand for IS NOT DISTINCT FROM) performs the same comparison in a concise and efficient form. The <=> operator is more efficient in for comparing join keys in a NULL-safe manner, because the operator can use a hash join while the OR expression cannot.

Examples:

The following examples refer to these simple tables containing small sets of integers:

```
[localhost:21000] > create table t1 (x int);
[localhost:21000] > insert into t1 values (1), (2), (3), (4), (5), (6);

[localhost:21000] > create table t2 (y int);
[localhost:21000] > insert into t2 values (2), (4), (6);
[localhost:21000] > create table t3 (z int);
[localhost:21000] > insert into t3 values (1), (3), (5);
```

The following example demonstrates an anti-join, returning the values from T1 that do not exist in T2 (in this case, the odd numbers 1, 3, and 5):

```
[localhost:21000] > select x from t1 left anti join t2 on (t1.x = t2.y);
+---+
| x |
+---+
| 1 |
| 3 |
| 5 |
+---+
```

Related Information

Optimizer hints in Impala

Performance considerations for join queries

SHOW statement

ORDER BY clause

The ORDER BY clause of a SELECT statement sorts the result set based on the values from one or more columns.

First, data is sorted locally by each impalad daemon, then streamed to the coordinator daemon, which merges the sorted result sets. For distributed queries, this is a relatively expensive operation and can require more memory capacity than a query without ORDER BY. Even if the query takes approximately the same time to finish with or without the ORDER BY clause, subjectively it can appear slower because no results are available until all processing is finished, rather than results coming back gradually as rows matching the WHERE clause are found. Therefore, if you only need the first N results from the sorted result set, also include the LIMIT clause, which reduces network overhead and the memory requirement on the coordinator node.

Syntax:

The full syntax for the ORDER BY clause is:

```
ORDER BY COL_REF [, COL_REF ...] [ASC | DESC] [NULLS FIRST | NULLS LAST]

col_ref ::= COLUMN_NAME | INTEGER_LITERAL
```

Although the most common usage is ORDER BY *COLUMN_NAME*, you can also specify ORDER BY 1 to sort by the first column of the result set, ORDER BY 2 to sort by the second column, and so on. The number must be a numeric literal, not some other kind of constant expression. (If the argument is some other expression, even a STRING value, the query succeeds but the order of results is undefined.)

ORDER BY *COLUMN_NUMBER* can only be used when the query explicitly lists the columns in the SELECT list, not with SELECT * queries.

Ascending and descending sorts:

The default sort order (the same as using the ASC keyword) puts the smallest values at the start of the result set, and the largest values at the end. Specifying the DESC keyword reverses that order.

Sort order for NULL values:

See the NULL on page 92 section for details about how NULL values are positioned in the sorted result set, and how to use the NULLS FIRST and NULLS LAST clauses. (The sort position for NULL values in ORDER BY ... DESC queries is changed in Impala 1.2.1 and higher to be more standards-compliant, and the NULLS FIRST and NULLS LAST keywords are new in Impala 1.2.1.)

Prior to Impala 1.4.0, Impala required any query including an ORDER_BY clause to also use a LIMIT clause. In Impala 1.4.0 and higher, the LIMIT clause is optional for ORDER BY queries. In cases where sorting a huge result set requires enough memory to exceed the Impala memory limit for a particular executor Impala daemon, Impala automatically uses a temporary disk work area to perform the sort operation.

Complex type considerations:

In Impala 2.3 and higher, the complex data types STRUCT, ARRAY, and MAP are available. These columns cannot be referenced directly in the ORDER BY clause. When you query a complex type column, you use join notation to “unpack” the elements of the complex type, and within the join query you can include an ORDER BY clause to control the order in the result set of the scalar elements from the complex type.

The following query shows how a complex type column cannot be directly used in an ORDER BY clause:

```
CREATE TABLE games (id BIGINT, score ARRAY <BIGINT>) STORED AS PARQUET;
...use LOAD DATA to load externally created Parquet files into the table...
SELECT id FROM games ORDER BY score DESC;
ERROR: AnalysisException: ORDER BY expression 'score' with complex type 'A
RRAY<BIGINT>' is not supported.
```

Examples:

The following query retrieves the user ID and score, only for scores greater than one million, with the highest scores for each user listed first. Because the individual array elements are now represented as separate rows in the result set, they can be used in the ORDER BY clause, referenced using the ITEM pseudo-column that represents each array element.

```
SELECT id, item FROM games, games.score
  WHERE item > 1000000
ORDER BY id, item desc;
```

The following queries use similar ORDER BY techniques with variations of the GAMES table, where the complex type is an ARRAY containing STRUCT or MAP elements to represent additional details about each game that was played. For an array of structures, the fields of the structure are referenced as ITEM.*FIELD_NAME*. For an array of maps, the keys and values within each array element are referenced as ITEM.KEY and ITEM.VALUE.

```
CREATE TABLE games2 (id BIGINT, play array < struct <game_name: string, scor
e: BIGINT, high_score: boolean> >) STORED AS PARQUET
...use LOAD DATA to load externally created Parquet files into the table...
SELECT id, item.game_name, item.score FROM games2, games2.play
  WHERE item.score > 1000000
ORDER BY id, item.score DESC;

CREATE TABLE games3 (id BIGINT, play ARRAY < MAP <STRING, BIGINT> >) STORED
AS PARQUET;
...use LOAD DATA to load externally created Parquet files into the table...
SELECT id, info.key AS k, info.value AS v from games3, games3.play AS plays,
 games3.play.item AS info
  WHERE info.KEY = 'score' AND info.VALUE > 1000000
ORDER BY id, info.value desc;
```

Usage notes:

Although the LIMIT clause is now optional on ORDER BY queries, if your query only needs some number of rows that you can predict in advance, use the LIMIT clause to reduce unnecessary processing. For example, if the query has a clause LIMIT 10, each executor Impala daemon sorts its portion of the relevant result set and only returns 10 rows to the coordinator node. The coordinator node picks the 10 highest or lowest row values out of this small intermediate result set.

If an ORDER BY clause is applied to an early phase of query processing, such as a subquery or a view definition, Impala ignores the ORDER BY clause. To get ordered results from a subquery or view, apply an ORDER BY clause to the outermost or final SELECT level.

ORDER BY is often used in combination with LIMIT to perform “top-N” queries:

```
SELECT user_id AS "Top 10 Visitors", SUM(page_views) FROM web_stats
  GROUP BY page_views, user_id
  ORDER BY SUM(page_views) DESC LIMIT 10;
```

ORDER BY is sometimes used in combination with OFFSET and LIMIT to paginate query results, although it is relatively inefficient to issue multiple queries like this against the large tables typically used with Impala:

```
SELECT page_title AS "Page 1 of search results", page_url FROM search_conten
t
  WHERE LOWER(page_title) LIKE '%game%')
  ORDER BY page_title LIMIT 10 OFFSET 0;
SELECT page_title AS "Page 2 of search results", page_url FROM search_conten
t
  WHERE LOWER(page_title) LIKE '%game%')
  ORDER BY page_title LIMIT 10 OFFSET 10;
SELECT page_title AS "Page 3 of search results", page_url FROM search_conte
nt
```

```
  WHERE LOWER(page_title) LIKE '%game%')
  ORDER BY page_title LIMIT 10 OFFSET 20;
```

Internal details:

Impala sorts the intermediate results of an ORDER BY clause in memory whenever practical. In a cluster of N executor Impala daemons, each daemon sorts roughly 1/Nth of the result set, the exact proportion varying depending on how the data matching the query is distributed in HDFS.

If the size of the sorted intermediate result set on any executor Impala daemon would cause the query to exceed the Impala memory limit, Impala sorts as much as practical in memory, then writes partially sorted data to disk. (This technique is known in industry terminology as “external sorting” and “spilling to disk”.) As each 8 MB batch of data is written to disk, Impala frees the corresponding memory to sort a new 8 MB batch of data. When all the data has been processed, a final merge sort operation is performed to correctly order the in-memory and on-disk results as the result set is transmitted back to the coordinator node. When external sorting becomes necessary, Impala requires approximately 60 MB of RAM at a minimum for the buffers needed to read, write, and sort the intermediate results. If more RAM is available on the Impala daemon, Impala will use the additional RAM to minimize the amount of disk I/O for sorting.

This external sort technique is used as appropriate on each Impala daemon (possibly including the coordinator node) to sort the portion of the result set that is processed on that node. When the sorted intermediate results are sent back to the coordinator node to produce the final result set, the coordinator node uses a merge sort technique to produce a final sorted result set without using any extra resources on the coordinator node.

Sorting considerations:

Although you can specify an ORDER BY clause in an INSERT ... SELECT statement, any ORDER BY clause is ignored, and the results are not necessarily sorted.

An ORDER BY clause without an additional LIMIT clause is ignored in any view definition. If you need to sort the entire result set from a view, use an ORDER BY clause in the SELECT statement that queries the view. You can still make a simple “top 10” report by combining the ORDER BY and LIMIT clauses in the same view definition:

```
[localhost:21000] > create table unsorted (x bigint);
[localhost:21000] > insert into unsorted values (1), (9), (3), (7), (5), (
8), (4), (6), (2);
[localhost:21000] > create view sorted_view as select x from unsorted order
 by x;
[localhost:21000] > select x from sorted_view; -- ORDER BY clause in view h
as no effect.
+---+
| x |
+---+
| 1 |
| 9 |
| 3 |
| 7 |
| 5 |
| 8 |
| 4 |
| 6 |
| 2 |
+---+
[localhost:21000] > select x from sorted_view order by x; -- View query requ
ires ORDER BY at outermost level.
+---+
| x |
+---+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
```

```
| 6 |
| 7 |
| 8 |
| 9 |
+---+
[localhost:21000] > create view top_3_view as select x from unsorted order b
y x limit 3;
[localhost:21000] > select x from top_3_view; -- ORDER BY and LIMIT together
 in view definition are preserved.
+---+
| x |
+---+
| 1 |
| 2 |
| 3 |
+---+
```

With the lifting of the requirement to include a LIMIT clause in every ORDER BY query (in Impala 1.4 and higher):

- Now the use of scratch disk space raises the possibility of an “out of disk space” error on a particular Impala daemon, as opposed to the previous possibility of an “out of memory” error. Make sure to keep at least 1 GB free on the filesystem used for temporary sorting work.

In Impala 1.2.1 and higher, all NULL values come at the end of the result set for ORDER BY ... ASC queries, and at the beginning of the result set for ORDER BY ... DESC queries. In effect, NULL is considered greater than all other values for sorting purposes. The original Impala behavior always put NULL values at the end, even for ORDER BY ... DESC queries. The new behavior in Impala 1.2.1 makes Impala more compatible with other popular database systems. In Impala 1.2.1 and higher, you can override or specify the sorting behavior for NULL by adding the clause NULLS FIRST or NULLS LAST at the end of the ORDER BY clause.

```
[localhost:21000] > create table numbers (x int);
[localhost:21000] > insert into numbers values (1), (null), (2), (null), (
3);
[localhost:21000] > select x from numbers order by x nulls first;
+------+
| x    |
+------+
| NULL |
| NULL |
| 1    |
| 2    |
| 3    |
+------+
[localhost:21000] > select x from numbers order by x desc nulls first;
+------+
| x    |
+------+
| NULL |
| NULL |
| 3    |
| 2    |
| 1    |
+------+
[localhost:21000] > select x from numbers order by x nulls last;
+------+
| x    |
+------+
| 1    |
| 2    |
| 3    |
| NULL |
| NULL |
+------+
```

```
[localhost:21000] > select x from numbers order by x desc nulls last;
+------+
| x    |
+------+
| 3    |
| 2    |
| 1    |
| NULL |
| NULL |
+------+
```

Using ORDER BY for collections of fixed length types in SELECT list

You can now use collections of fixed length types in the sorting tuple. However, you cannot sort by these collection columns but they can be in the SELECT list along with other column(s) by which you sort.

Example of a supported query:

```
select id, int_array from complextypestbl order by id;
```

Example of an unsupported query:

```
select int_array from complextypestbl order by int_array;
```

Also note that structs that contain any type of collection (even fixed length) are not allowed in the sorting tuple.

Examples:

Sort a collection.

```
select id, int_array from complextypestbl order by id;
```

Sort collection from HMS view.

```
select id, int_array from complextypes_arrays_only_view order by id;
```

Sort collection from WITH-clause inline view.

```
 with v as (select id, int_array from complextypestbl)
 select id, int_array from v order by id;
```

Sort collection from nested query inline view.

```
select id, int_array
from (select id, int_array from complextypestbl) v
order by id;
```

Related information:

See the SELECT statement for further examples of queries with the ORDER BY clause.

Analytic functions use the ORDER BY clause in a different context to define the sequence in which rows are analyzed. See Impala analytic functions on page 421 for details.

GROUP BY clause

Specify the GROUP BY clause in queries that use aggregation functions, such as COUNT(), SUM(), AVG(), MIN(), and MAX(). Specify in the GROUP BY clause the names of all the columns that do not participate in the aggregation operation.

Complex type considerations:

In Impala 2.3 and higher, the complex data types STRUCT, ARRAY, and MAP are available. These columns cannot be referenced directly in the ORDER BY clause. When you query a complex type column, you use join notation to “unpack” the elements of the complex type, and within the join query you can include an ORDER BY clause to control the order in the result set of the scalar elements from the complex type.

Zero-length strings: For purposes of clauses such as DISTINCT and GROUP BY, Impala considers zero-length strings (""), NULL, and space to all be different values.

Examples:

For example, the following query finds the 5 items that sold the highest total quantity (using the SUM() function, and also counts the number of sales transactions for those items (using the COUNT() function). Because the column representing the item IDs is not used in any aggregation functions, we specify that column in the GROUP BY clause.

```
select
  ss_item_sk as Item,
  count(ss_item_sk) as Times_Purchased,
  sum(ss_quantity) as Total_Quantity_Purchased
from store_sales
  group by ss_item_sk
  order by sum(ss_quantity) desc
  limit 5;
+-------+-----------------+--------------------------+
| item  | times_purchased | total_quantity_purchased |
+-------+-----------------+--------------------------+
| 9325  | 372             | 19072                    |
| 4279  | 357             | 18501                    |
| 7507  | 371             | 18475                    |
| 5953  | 369             | 18451                    |
| 16753 | 375             | 18446                    |
+-------+-----------------+--------------------------+
```

The HAVING clause lets you filter the results of aggregate functions, because you cannot refer to those expressions in the WHERE clause. For example, to find the 5 lowest-selling items that were included in at least 100 sales transactions, we could use this query:

```
select
  ss_item_sk as Item,
  count(ss_item_sk) as Times_Purchased,
  sum(ss_quantity) as Total_Quantity_Purchased
from store_sales
  group by ss_item_sk
      having count(ss_item_sk) >= 100
  order by sum(ss_quantity)
  limit 5;
+-------+-----------------+--------------------------+
| item  | times_purchased | total_quantity_purchased |
+-------+-----------------+--------------------------+
| 13943 | 105             | 4087                     |
| 2992  | 101             | 4176                     |
| 4773  | 107             | 4204                     |
| 14350 | 103             | 4260                     |
| 11956 | 102             | 4275                     |
+-------+-----------------+--------------------------+
```

When performing calculations involving scientific or financial data, remember that columns with type FLOAT or DOUBLE are stored as true floating-point numbers, which cannot precisely represent every possible fractional value. Thus, if you include a FLOAT or DOUBLE column in a GROUP BY clause, the results might not precisely match literal values in your query or from an original Text data file. Use rounding operations, the BETWEEN operator, or another arithmetic technique to match floating-point values that are “near” literal values you expect. For example, this

query on the ss_wholesale_cost column returns cost values that are close but not identical to the original figures that were entered as decimal fractions.

```
select ss_wholesale_cost, avg(ss_quantity * ss_sales_price) as avg_revenue_p
er_sale
  from sales
  group by ss_wholesale_cost
  order by avg(ss_quantity * ss_sales_price) desc
  limit 5;
+-------------------+----------------------+
| ss_wholesale_cost | avg_revenue_per_sale |
+-------------------+----------------------+
| 96.94000244140625 | 4454.351539300434    |
| 95.93000030517578 | 4423.119941283189    |
| 98.37999725341797 | 4332.516490316291    |
| 97.97000122070312 | 4330.480601655014    |
| 98.52999877929688 | 4291.316953108634    |
+-------------------+----------------------+
```

Notice how wholesale cost values originally entered as decimal fractions such as 96.94 and 98.38 are slightly larger or smaller in the result set, due to precision limitations in the hardware floating-point types. The imprecise representation of FLOAT and DOUBLE values is why financial data processing systems often store currency using data types that are less space-efficient but avoid these types of rounding errors.

ROLLUP

The GROUP BY ROLLUP clause creates a group for each combination of column expressions. When you run a query with the “rollup” clause on a group of columns, it is run from right to left decreasing the number of column expressions over which it creates groups and the aggregation(s).

For example, GROUP BY ROLLUP (col1, col2, col3) creates groups for each combination of column expressions in the following lists.

- col1, col2, col3
- col1, col2, col1NULL
- col1, NULL, NULL
- NULL, NULL, NULL This is the grand total

Example:

The following query analyzes sales data, to study the amount of sales that is occurring for different products, in different states and regions using the ROLLUP feature.

```
select
 region, state, product, sum(sales) as total_sales,
from sales_history
 group by rollup (region, state, product);
```

Semantically the above query is equivalent to:

```
select region, state, product, sum(sales) as total_sales
      from sales_history
      group by region, state, product;

      union all

      select region, state, null, sum(sales) as  total_sales
      from sales_history
      group by region, state;

      union all
```

```
select region, null, null, sum(sales) as  total_sales
from sales_history
group by region;

union all

select null, null, null, sum(sales) as total_sales
from sales_history;
```

The query result has the same aggregations as the simple GROUP BY without the ROLLUP. In addition, it creates subtotals for each value of Region. Finally, it gives a grand total for all rows. The result looks like this:

```
+--------+-------+---------+-------------+
| region | state | product | total_sales |
+--------+-------+---------+-------------+
| NULL   | NULL  | NULL    | 6200        |
| EAST   | MA    | BOATS   | 100         |
| EAST   | MA    | CARS    | 1500        |
| EAST   | MA    | NULL    | 1600        |
| EAST   | NY    | BOATS   | 150         |
| EAST   | NY    | CARS    | 1000        |
| EAST   | NY    | NULL    | 1150        |
| EAST   | NULL  | NULL    | 2750        |
| WEST   | CA    | CARS    | 500         |
| WEST   | CA    | BOATS   | 750         |
| WEST   | CA    | NULL    | 1250        |
| WEST   | AZ    | BOATS   | 2000        |
| WEST   | AZ    | CARS    | 200         |
| WEST   | AZ    | NULL    | 2200        |
| WEST   | NULL  | NULL    | 3450        |
+--------+-------+---------+-------------+
```

CUBE

This creates groups for all possible combinations of columns. For e.g., GROUP BY CUBE for (col1, col2) creates groups for:

- col1, col2
- col1, NULL
- NULL, col2
- NULL, NULL

Example:

```
select region, state, sum(sales) as total_sales
       from sales_history
       group by cube (region, state);
```

This will produce results for the following combinations (region, state), (NULL, state), (Region, NULL), and (NULL, NULL).

GROUPING SETS

A grouping set is a set of columns by which you group using the GROUP BY clause.

Example:

```
        select region, state, sum(sales) as total_sales
```

```
        from sales_history
        group by grouping sets ((region, state), (region), (state), ());
```

This lets you define four grouping sets (region, state), (region), (state), and (). Semantically the above query is equivalent to:

```
        select region, state, sum(sales) as total_sales
        from sales_history
        group by region, state;

        union all

        select region, null, sum(sales) as total_sales
        from sales_history
        group by region;

        union all

        select null, state, sum(sales) as total_sales
        from sales_history
        group by state;

        union all

        select null, null, sum(sales) as total_sales
        from sales_history;
```

HAVING clause

Performs a filter operation on a SELECT query, by examining the results of aggregation functions rather than testing each individual table row. Therefore, it is always used in conjunction with a function such as COUNT(), SUM(), AVG(), MIN(), or MAX(), and typically with the GROUP BY clause also.

Restrictions:

The filter expression in the HAVING clause cannot include a scalar subquery.

LIMIT clause

The LIMIT clause in a SELECT query sets a maximum number of rows for the result set. Pre-selecting the maximum size of the result set helps Impala to optimize memory usage while processing a distributed query.

Syntax:

```
LIMIT CONSTANT_INTEGER_EXPRESSION
```

The argument to the LIMIT clause must evaluate to a constant value. It can be a numeric literal, or another kind of numeric expression involving operators, casts, and function return values. You cannot refer to a column or use a subquery.

Usage notes:

This clause is useful in contexts such as:

- To return exactly N items from a top-N query, such as the 10 highest-rated items in a shopping category or the 50 hostnames that refer the most traffic to a web site.
- To demonstrate some sample values from a table or a particular query. (To display some arbitrary items, use a query with no ORDER BY clause. An ORDER BY clause causes additional memory and/or disk usage during the query.)

- To keep queries from returning huge result sets by accident if a table is larger than expected, or a WHERE clause matches more rows than expected.

Originally, the value for the LIMIT clause had to be a numeric literal. In Impala 1.2.1 and higher, it can be a numeric expression.

Prior to Impala 1.4.0, Impala required any query including an ORDER_BY clause to also use a LIMIT clause. In Impala 1.4.0 and higher, the LIMIT clause is optional for ORDER BY queries. In cases where sorting a huge result set requires enough memory to exceed the Impala memory limit for a particular executor Impala daemon, Impala automatically uses a temporary disk work area to perform the sort operation.

In Impala 1.2.1 and higher, you can combine a LIMIT clause with an OFFSET clause to produce a small result set that is different from a top-N query, for example, to return items 11 through 20. This technique can be used to simulate “paged” results. Because Impala queries typically involve substantial amounts of I/O, use this technique only for compatibility in cases where you cannot rewrite the application logic. For best performance and scalability, wherever practical, query as many items as you expect to need, cache them on the application side, and display small groups of results to users using application logic.

Restrictions:

Correlated subqueries used in EXISTS and IN operators cannot include a LIMIT clause.

Examples:

The following example shows how the LIMIT clause caps the size of the result set, with the limit being applied after any other clauses such as WHERE.

```
[localhost:21000] > create database limits;
[localhost:21000] > use limits;
[localhost:21000] > create table numbers (x int);
[localhost:21000] > insert into numbers values (1), (3), (4), (5), (2);
Inserted 5 rows in 1.34s
[localhost:21000] > select x from numbers limit 100;
+---+
| x |
+---+
| 1 |
| 3 |
| 4 |
| 5 |
| 2 |
+---+
Returned 5 row(s) in 0.26s
[localhost:21000] > select x from numbers limit 3;
+---+
| x |
+---+
| 1 |
| 3 |
| 4 |
+---+
Returned 3 row(s) in 0.27s
[localhost:21000] > select x from numbers where x > 2 limit 2;
+---+
| x |
+---+
| 3 |
| 4 |
+---+
Returned 2 row(s) in 0.27s
```

For top-N and bottom-N queries, you use the ORDER BY and LIMIT clauses together:

```
[localhost:21000] > select x as "Top 3" from numbers order by x desc limit 3
;
+-------+
| top 3 |
+-------+
| 5     |
| 4     |
| 3     |
+-------+
[localhost:21000] > select x as "Bottom 3" from numbers order by x limit 3;
+----------+
| bottom 3 |
+----------+
| 1        |
| 2        |
| 3        |
+----------+
```

You can use constant values besides integer literals as the LIMIT argument:

```
-- Other expressions that yield constant integer values work too.
SELECT x FROM t1 LIMIT 1e6;                       -- Limit is one million.
SELECT x FROM t1 LIMIT length('hello world');     -- Limit is 11.
SELECT x FROM t1 LIMIT 2+2;                       -- Limit is 4.
SELECT x FROM t1 LIMIT cast(truncate(9.9) AS INT); -- Limit is 9.
```

OFFSET clause

The OFFSET clause in a SELECT query causes the result set to start some number of rows after the logical first item. The result set is numbered starting from zero, so OFFSET 0 produces the same result as leaving out the OFFSET clause. Always use this clause in combination with ORDER BY (so that it is clear which item should be first, second, and so on) and LIMIT (so that the result set covers a bounded range, such as items 0-9, 100-199, and so on).

In Impala 1.2.1 and higher, you can combine a LIMIT clause with an OFFSET clause to produce a small result set that is different from a top-N query, for example, to return items 11 through 20. This technique can be used to simulate “paged” results. Because Impala queries typically involve substantial amounts of I/O, use this technique only for compatibility in cases where you cannot rewrite the application logic. For best performance and scalability, wherever practical, query as many items as you expect to need, cache them on the application side, and display small groups of results to users using application logic.

Examples:

The following example shows how you could run a “paging” query originally written for a traditional database application. Because typical Impala queries process megabytes or gigabytes of data and read large data files from disk each time, it is inefficient to run a separate query to retrieve each small group of items. Use this technique only for compatibility while porting older applications, then rewrite the application code to use a single query with a large result set, and display pages of results from the cached result set.

```
[localhost:21000] > create table numbers (x int);
[localhost:21000] > insert into numbers select x from very_long_sequence;
Inserted 1000000 rows in 1.34s
[localhost:21000] > select x from numbers order by x limit 5 offset 0;
+----+
| x  |
+----+
| 1  |
| 2  |
| 3  |
| 4  |
| 5  |
```

```
+----+
[localhost:21000] > select x from numbers order by x limit 5 offset 5;
+----+
| x  |
+----+
| 6  |
| 7  |
| 8  |
| 9  |
| 10 |
+----+
```

UNION, INTERSECT, and EXCEPT clauses

The UNION, INTERSECT, and EXCEPT clauses allow you to combine rows from two or more tables. You may use them when you must combine the result sets of multiple queries into one single result.

UNION Clause

The UNION clause lets you combine the result sets of multiple queries. By default, the result sets are combined as if the DISTINCT operator was applied.

Syntax:

```
QUERY_1 UNION [DISTINCT | ALL] QUERY_2
```

Usage notes:

The UNION keyword by itself is the same as UNION DISTINCT. Because eliminating duplicates can be a memory-intensive process for a large result set, prefer UNION ALL where practical. (That is, when you know the different queries in the union will not produce any duplicates, or where the duplicate values are acceptable.)

When an ORDER BY clause applies to a UNION ALL or UNION query, in Impala 1.4 and higher, the LIMIT clause is no longer required. To make the ORDER BY and LIMIT clauses apply to the entire result set, turn the UNION query into a subquery, SELECT from the subquery, and put the ORDER BY clause at the end, outside the subquery.

Examples:

First, set up some sample data, including duplicate 1 values:

```
[localhost:21000] > create table few_ints (x int);
[localhost:21000] > insert into few_ints values (1), (1), (2), (3);
[localhost:21000] > set default_order_by_limit=1000;
```

This example shows how UNION ALL returns all rows from both queries, without any additional filtering to eliminate duplicates. For the large result sets common with Impala queries, this is the most memory-efficient technique.

```
[localhost:21000] > select x from few_ints order by x;
+---+
| x |
+---+
| 1 |
| 1 |
| 2 |
| 3 |
+---+
Returned 4 row(s) in 0.41s
[localhost:21000] > select x from few_ints union all select x from few_ints;
+---+
| x |
+---+
```

```
| 1 |
| 1 |
| 2 |
| 3 |
| 1 |
| 1 |
| 2 |
| 3 |
+---+
Returned 8 row(s) in 0.42s
[localhost:21000] > select * from (select x from few_ints union all select x
 from few_ints) as t1 order by x;
+---+
| x |
+---+
| 1 |
| 1 |
| 1 |
| 1 |
| 2 |
| 2 |
| 3 |
| 3 |
+---+
Returned 8 row(s) in 0.53s
[localhost:21000] > select x from few_ints union all select 10;
+----+
| x  |
+----+
| 10 |
| 1  |
| 1  |
| 2  |
| 3  |
+----+
Returned 5 row(s) in 0.38s
```

This example shows how the UNION clause without the ALL keyword condenses the result set to eliminate all duplicate values, making the query take more time and potentially more memory. The extra processing typically makes this technique not recommended for queries that return result sets with millions or billions of values.

```
[localhost:21000] > select x from few_ints union select x+1 from few_ints;
+---+
| x |
+---+
| 3 |
| 4 |
| 1 |
| 2 |
+---+
Returned 4 row(s) in 0.51s
[localhost:21000] > select x from few_ints union select 10;
+----+
| x  |
+----+
| 2  |
| 10 |
| 1  |
| 3  |
+----+
Returned 4 row(s) in 0.49s
[localhost:21000] > select * from (select x from few_ints union select x fr
om few_ints) as t1 order by x;
```

```
+---+
| x |
+---+
| 1 |
| 2 |
| 3 |
+---+
Returned 3 row(s) in 0.53s
```

EXCEPT and INTERSECT Clauses

The EXCEPT and INTERSECT clause let you combine the result sets of multiple queries and return distinct rows by comparing the results of two queries. EXCEPT returns distinct rows from the left input query that are not part of the result set of the right input query. INTERSECT returns distinct rows that are output by both the left and right input queries operator.

Syntax:

```
QUERY_1 EXCEPT [DISTINCT] QUERY_2
```

```
QUERY_1 INTERSECT [DISTINCT] QUERY_2
```

Usage notes:

When you use EXCEPT or INTERSECT to combine the result sets of two queries the number and the order of the columns must be the same in all queries and the data types must be compatible.

Note: INTERSECT ALL and EXCEPT ALL are not currently supported.

Examples:

This example shows the number of Boeing planes leaving from DFW in 2015. And the subquery shown here uses an INTERSECT on the tail number (tailnum), the unique identification of a plane.

```
SELECT COUNT(*) FROM (
                 SELECT tailnum FROM flights
                 WHERE
                     flights.origin='DFW'
                     AND flights.year = 2005
                 INTERSECT
                 SELECT tailnum FROM planes
                 WHERE
                      planes.manufacturer = 'BOEING') subq;
```

The following example shows the number of non-Boeing planes leaving from DFW. This is done by executing the above query using the INTERSECT operator and then removing the planes that are not from the BOEING manufacturer using the EXCEPT clause.

```
SELECT COUNT(*) FROM (
                 SELECT tailnum FROM flights
                 WHERE
                     flights.origin='DFW'
                     AND flights.year = 2005
                 INTERSECT
                 SELECT tailnum FROM planes
                 EXCEPT
                 SELECT tailnum FROM planes
                 WHERE
                     planes.manufacturer = 'BOEING') subq;
```

Subqueries in Impala SELECT statements

A *subquery* is a query that is nested within another query. Subqueries let queries on one table dynamically adapt based on the contents of another table. This technique provides great flexibility and expressive power for SQL queries.

A subquery can return a result set for use in the FROM or WITH clauses, or with operators such as IN or EXISTS.

A *scalar subquery* produces a result set with a single row containing a single column, typically produced by an aggregation function such as MAX() or SUM(). This single result value can be substituted in scalar contexts such as arguments to comparison operators. If the result set is empty, the value of the scalar subquery is NULL. For example, the following query finds the maximum value of T2.Y and then substitutes that value into the WHERE clause of the outer block that queries T1:

```
SELECT x FROM t1 WHERE x > (SELECT MAX(y) FROM t2);
```

Uncorrelated subqueries do not refer to any tables from the outer block of the query. The same value or set of values produced by the subquery is used when evaluating each row from the outer query block. In this example, the subquery returns an arbitrary number of values from T2.Y, and each value of T1.X is tested for membership in that same set of values:

```
SELECT x FROM t1 WHERE x IN (SELECT y FROM t2);
```

Uncorrelated subqueries are now supported in the SELECT list statement.

Example:

This query returns a row for every sale, along with the corresponding employee information.

```
      SELECT EmpID, EmpName, sum(SaleAmount) as EmpSales,
      sum(SaleAmount) / (SELECT sum(SaleAmount) FROM Sales) as PortionOfT
otalSales
      FROM Employee
      JOIN Sales on Employee.EmpID = Sales.EmpID GROUP BY EmpID, EmpName;
```

In this example, the subquery “(SELECT sum(SaleAmount) FROM Sales)” is an uncorrelated subquery that's evaluated separately and returns the sum across all sales, not grouped by any Employee details.

Then the first and second sum(saleAmount) will be the total of all sales for that employment.

Impala also supports *Uncorrelated subqueries* in the HAVING clause.

Example:

This query finds all the departments with the average salary greater than the average salary across all departments.

```
      SELECT dept_id, avg(salary) FROM employees GROUP BY dept_id HAVING
      avg(salary) > (SELECT avg(salary) FROM employees);
```

Correlated subqueries compare one or more values from the outer query block to values referenced in the WHERE clause of the subquery. Each row evaluated by the outer WHERE clause can be evaluated using a different set of values. These kinds of subqueries are restricted in the kinds of comparisons they can do between columns of the inner and outer tables. (See the following Restrictions item.)

For example, the following query finds all the employees with salaries that are higher than average for their department. The subquery potentially computes a different AVG() value for each employee.

```
SELECT employee_name, employee_id FROM employees one WHERE
  salary > (SELECT avg(salary) FROM employees two WHERE one.dept_id = two
.dept_id);
```

Syntax:

Subquery in the FROM clause:

```
SELECT SELECT_LIST FROM TABLE_REF [, TABLE_REF ...]

TABLE_REF ::= TABLE_NAME | (SELECT_STATEMENT)
```

Subqueries in WHERE clause:

```
WHERE VALUE COMPARISON_OPERATOR (SCALAR_SELECT_STATEMENT)
WHERE VALUE [NOT] IN (SELECT_STATEMENT)
WHERE [NOT] EXISTS (CORRELATED_SELECT_STATEMENT)
WHERE NOT EXISTS (CORRELATED_SELECT_STATEMENT)
```

comparison_operator is a numeric comparison such as =, <=, !=, and so on, or a string comparison operator such as LIKE or REGEXP.

Although you can use non-equality comparison operators such as < or >=, the subquery must include at least one equality comparison between the columns of the inner and outer query blocks.

All syntax is available for both correlated and uncorrelated queries, except that the NOT EXISTS clause cannot be used with an uncorrelated subquery.

Impala subqueries can be nested arbitrarily deep.

Standards compliance: Introduced in SQL:1999.

Examples:

This example illustrates how subqueries can be used in the FROM clause to organize the table names, column names, and column values by producing intermediate result sets, especially for join queries.

```
SELECT avg(t1.x), max(t2.y) FROM
  (SELECT id, cast(a AS DECIMAL(10,5)) AS x FROM raw_data WHERE a BETWEEN 0
 AND 100) AS t1
  JOIN
  (SELECT id, length(s) AS y FROM raw_data WHERE s LIKE 'A%') AS t2;
  USING (id);
```

These examples show how a query can test for the existence of values in a separate table using the EXISTS() operator with a subquery.

The following examples show how a value can be compared against a set of values returned by a subquery.

```
SELECT count(x) FROM t1 WHERE EXISTS(SELECT 1 FROM t2 WHERE t1.x = t2.y * 10
);

SELECT x FROM t1 WHERE x IN (SELECT y FROM t2 WHERE state = 'CA');
```

The following examples demonstrate scalar subqueries. When a subquery is known to return a single value, you can substitute it where you would normally put a constant value.

```
SELECT x FROM t1 WHERE y = (SELECT max(z) FROM t2);
SELECT x FROM t1 WHERE y > (SELECT count(z) FROM t2);
```

Usage notes:

If the same table is referenced in both the outer and inner query blocks, construct a table alias in the outer query block and use a fully qualified name to distinguish the inner and outer table references:

```
SELECT * FROM t1 one WHERE id IN (SELECT parent FROM t1 two WHERE t1.parent
= t2.id);
```

The STRAIGHT_JOIN hint affects the join order of table references in the query block containing the hint. It does not affect the join order of nested queries, such as views, inline views, or WHERE-clause subqueries. To use this hint for performance tuning of complex queries, apply the hint to all query blocks that need a fixed join order.

Internal details:

Internally, subqueries involving IN, NOT IN, EXISTS, or NOT EXISTS clauses are rewritten into join queries. Depending on the syntax, the subquery might be rewritten to an outer join, semi join, cross join, or anti join.

A query is processed differently depending on whether the subquery calls any aggregation functions. There are correlated and uncorrelated forms, with and without calls to aggregation functions. Each of these four categories is rewritten differently.

Column statistics considerations:

Because queries that include correlated and uncorrelated subqueries in the WHERE clause are written into join queries, to achieve best performance, follow the same guidelines for running the COMPUTE STATS statement as you do for tables involved in regular join queries. Run the COMPUTE STATS statement for each associated tables after loading or substantially changing the data in that table.

Restrictions:

The initial Impala support for nested subqueries addresses the most common use cases. Some restrictions remain:

- Although you can use subqueries in a query involving UNION or UNION ALL in Impala 2.1.0 and higher, currently you cannot construct a union of two subqueries (for example, in the argument of an IN or EXISTS operator).
- Subqueries returning scalar values cannot be used with the operators ANY or ALL. (Impala does not currently have a SOME operator, but if it did, the same restriction would apply.)
- For the EXISTS and NOT EXISTS clauses, any subquery comparing values from the outer query block to another table must use at least one equality comparison, not exclusively other kinds of comparisons such as less than, greater than, BETWEEN, or !=.
-
- Expressions inside a subquery, for example in the WHERE clause, can use OR conjunctions but the OR conjunction cannot be combined with more than one subquery.
- Scalar subqueries are only supported in numeric contexts.
- You cannot use subqueries with the CASE function to generate the comparison value, the values to be compared against, or the return value.
- Only uncorrelated subqueries are supported in the filter condition for the HAVING clause. Similarly only a SELE CT uncorrelated scalar subquery is supported.
- You must use a fully qualified name (*TABLE_NAME.COLUMN_NAME* or *DATABASE_NAME.TABLE_NAME.COLUMN_NAME*) when referring to any column from the outer query block within a subquery.
- The TABLESAMPLE clause of the SELECT statement does not apply to a table reference derived from a view, a subquery, or anything other than a real base table. This clause only works for tables backed by HDFS or HDFS-like data files, therefore it does not apply to Kudu or HBase tables.

Complex type considerations:

For the complex types (ARRAY, STRUCT, and MAP) available in Impala 2.3 and higher, the join queries that “unpack” complex type columns often use correlated subqueries in the FROM clause. For example, if the first table in the join clause is CUSTOMER, the second join clause might have a subquery that selects from the column CUST

OMER.C_ORDERS, which is an ARRAY. The subquery re-evaluates the ARRAY elements corresponding to each row from the CUSTOMER table.

Related Information

Table and column statistics

TABLESAMPLE clause

Specify the TABLESAMPLE clause when you need to explore the data distribution within the table, the table is very large, and it is impractical or unnecessary to process all the data from the table or selected partitions.

The clause makes the query process a randomized set of data files from the table, so that the total volume of data is greater than or equal to the specified percentage of data bytes within that table. (Or the data bytes within the set of partitions that remain after partition pruning is performed.)

Syntax:

```
TABLESAMPLE SYSTEM(PERCENTAGE) [REPEATABLE(SEED)]
```

The TABLESAMPLE clause comes immediately after a table name or table alias.

The SYSTEM keyword represents the sampling method. Currently, Impala only supports a single sampling method named SYSTEM.

The *PERCENTAGE* argument is an integer literal from 0 to 100. A percentage of 0 produces an empty result set for a particular table reference, while a percentage of 100 uses the entire contents. Because the sampling works by selecting a random set of data files, the proportion of sampled data from the table may be greater than the specified percentage, based on the number and sizes of the underlying data files. See the usage notes for details.

The optional REPEATABLE keyword lets you specify an arbitrary positive integer seed value that ensures that when the query is run again, the sampling selects the same set of data files each time. REPEATABLE does not have a default value. If you omit the REPEATABLE keyword, the random seed is derived from the current time.

See the COMPUTE STATS statement for the TABLESAMPLE clause used in the COMPUTE STATS statement.

Usage notes:

You might use this clause with aggregation queries, such as finding the approximate average, minimum, or maximum where exact precision is not required. You can use these findings to plan the most effective strategy for constructing queries against the full table or designing a partitioning strategy for the data.

Some other database systems have a TABLESAMPLE clause. The Impala syntax for this clause is modeled on the syntax for popular relational databases, not the Hive TABLESAMPLE clause. For example, there is no BUCKETS keyword as in Hive SQL.

The precision of the *PERCENTAGE* threshold depends on the number and sizes of the underlying data files. Impala brings in additional data files, one at a time, until the number of bytes exceeds the specified percentage based on the total number of bytes for the entire set of table data. The precision of the percentage threshold is higher when the table contains many data files with consistent sizes. See the code listings later in this section for examples.

When you estimate characteristics of the data distribution based on sampling a percentage of the table data, be aware that the data might be unevenly distributed between different files. Do not assume that the percentage figure reflects the percentage of rows in the table. For example, one file might contain all blank values for a STRING column, while another file contains long strings in that column; therefore, one file could contain many more rows than another. Likewise, a table created with the SORT BY clause might contain narrow ranges of values for the sort columns, making it impractical to extrapolate the number of distinct values for those columns based on sampling only some of the data files.

Because a sample of the table data might not contain all values for a particular column, if the TABLESAMPLE is used in a join query, the key relationships between the tables might produce incomplete result sets compared to joins using all the table data. For example, if you join 50% of table A with 50% of table B, some values in the join columns might not match between the two tables, even though overall there is a 1:1 relationship between the tables.

The REPEATABLE keyword makes identical queries use a consistent set of data files when the query is repeated. You specify an arbitrary integer key that acts as a seed value when Impala randomly selects the set of data files to use in the query. This technique lets you verify correctness, examine performance, and so on for queries using the TABLESAMPLE clause without the sampled data being different each time. The repeatable aspect is reset (that is, the set of selected data files may change) any time the contents of the table change. The statements or operations that can make sampling results non-repeatable are:

- INSERT.
- TRUNCATE TABLE.
- LOAD DATA.
- REFRESH or INVALIDATE METADATA after files are added or removed by a non-Impala mechanism.

This clause is similar in some ways to the LIMIT clause, because both serve to limit the size of the intermediate data and final result set. LIMIT 0 is more efficient than TABLESAMPLE SYSTEM(0) for verifying that a query can run without producing any results. TABLESAMPLE SYSTEM(*N*) often makes query processing more efficient than using a LIMIT clause by itself, because all phases of query execution use less data overall. If the intent is to retrieve some representative values from the table in an efficient way, you might combine TABLESAMPLE, ORDER BY, and LIMIT clauses within a single query.

Partitioning:

When you query a partitioned table, any partition pruning happens before Impala selects the data files to sample. For example, in a table partitioned by year, a query with WHERE year = 2017 and a TABLESAMPLE SYSTEM(10) clause would sample data files representing at least 10% of the bytes present in the 2017 partition.

Amazon S3 considerations:

This clause applies to S3 tables the same way as tables with data files stored on HDFS.

ADLS considerations:

This clause applies to ADLS tables the same way as tables with data files stored on HDFS.

Kudu considerations:

This clause does not apply to Kudu tables.

HBase considerations:

This clause does not apply to HBase tables.

Performance considerations:

From a performance perspective, the TABLESAMPLE clause is especially valuable for exploratory queries on text, Avro, or other file formats other than Parquet. Text-based or row-oriented file formats must process substantial amounts of redundant data for queries that derive aggregate results such as MAX(), MIN(), or AVG() for a single column. Therefore, you might use TABLESAMPLE early in the ETL pipeline, when data is still in raw text format and has not been converted to Parquet or moved into a partitioned table.

Restrictions:

This clause applies only to tables that use a storage layer with underlying raw data files, such as HDFS, Amazon S3, or Microsoft ADLS.

This clause does not apply to table references that represent views. A query that applies the TABLESAMPLE clause to a view or a subquery fails with a semantic error.

Because the sampling works at the level of entire data files, it is by nature coarse-grained. It is possible to specify a small sample percentage but still process a substantial portion of the table data if the table contains relatively few data files, if each data file is very large, or if the data files vary substantially in size. Be sure that you understand the data distribution and physical file layout so that you can verify if the results are suitable for extrapolation. For example, if the table contains only a single data file, the “sample” will consist of all the table data regardless of the percentage you specify. If the table contains data files of 1 GiB, 1 GiB, and 1 KiB, when you specify a sampling percentage of 50 you would either process slightly more than 50% of the table (1 GiB + 1 KiB) or almost the entire table (1 GiB + 1 GiB), depending on which data files were selected for sampling.

If data files are added by a non-Impala mechanism, and the table metadata is not updated by a REFRESH or INVALIDATE METADATA statement, the TABLESAMPLE clause does not consider those new files when computing the number of bytes in the table or selecting which files to sample.

If data files are removed by a non-Impala mechanism, and the table metadata is not updated by a REFRESH or INVALIDATE METADATA statement, the query fails if the TABLESAMPLE clause attempts to reference any of the missing files.

Examples:

The following examples demonstrate the TABLESAMPLE clause. These examples intentionally use very small data sets to illustrate how the number of files, size of each file, and overall size of data in the table interact with the percentage specified in the clause.

These examples use an unpartitioned table, containing several files of roughly the same size:

```
create table sample_demo (x int, s string);

insert into sample_demo values (1, 'one');
insert into sample_demo values (2, 'two');
insert into sample_demo values (3, 'three');
insert into sample_demo values (4, 'four');
insert into sample_demo values (5, 'five');

show files in sample_demo;
+---------------------+------+-----------+
| Path                | Size | Partition |
+---------------------+------+-----------+
| 991213608_data.0.   | 7B   |           |
| 982196806_data.0.   | 6B   |           |
| _2122096884_data.0. | 8B   |           |
| _586325431_data.0.  | 6B   |           |
| 1894746258_data.0.  | 7B   |           |
+---------------------+------+-----------+
show table stats sample_demo;
+-------+--------+------+--------+-------------------------+
| #Rows | #Files | Size | Format | Location                |
+-------+--------+------+--------+-------------------------+
| -1    | 5      | 34B  | TEXT   | /tsample.db/sample_demo |
+-------+--------+------+--------+-------------------------+
```

A query that samples 50% of the table must process at least 17 bytes of data. Based on the sizes of the data files, we can predict that each such query uses 3 arbitrary files. Any 1 or 2 files are not enough to reach 50% of the total data in the table (34 bytes), so the query adds more files until it passes the 50% threshold:

```
select distinct x from sample_demo tablesample system(50);
+---+
| x |
+---+
| 4 |
| 1 |
| 5 |
+---+

select distinct x from sample_demo tablesample system(50);
+---+
| x |
+---+
| 5 |
| 4 |
| 2 |
+---+
```

```
select distinct x from sample_demo tablesample system(50);
+---+
| x |
+---+
| 5 |
| 3 |
| 2 |
+---+
```

To help run reproducible experiments, the REPEATABLE clause causes Impala to choose the same set of files for each query. Although the data set being considered is deterministic, the order of results varies (in the absence of an ORDER BY clause) because of the way distributed queries are processed:

```
select distinct x from sample_demo
  tablesample system(50) repeatable (12345);
+---+
| x |
+---+
| 3 |
| 2 |
| 1 |
+---+

select distinct x from sample_demo
  tablesample system(50) repeatable (12345);
+---+
| x |
+---+
| 2 |
| 1 |
| 3 |
+---+
```

The following examples show how uneven data distribution affects which data is sampled. Adding another data file containing a long string value changes the threshold for 50% of the total data in the table:

```
insert into sample_demo values
(1000, 'Boyhood is the longest time in life for a boy. The last term of the
 school-year is made of decades, not of weeks, and living through them is
like waiting for the millennium. Booth Tarkington');

show files in sample_demo;
+---------------------+------+-----------+
| Path                | Size | Partition |
+---------------------+------+-----------+
| 991213608_data.0.   | 7B   |           |
| 982196806_data.0.   | 6B   |           |
| _253317650_data.0.  | 196B |           |
| _2122096884_data.0. | 8B   |           |
| _586325431_data.0.  | 6B   |           |
| 1894746258_data.0.  | 7B   |           |
+---------------------+------+-----------+

show table stats sample_demo;
+-------+--------+------+--------+-------------------------+
| #Rows | #Files | Size | Format | Location                |
+-------+--------+------+--------+-------------------------+
| -1    | 6      | 230B | TEXT   | /tsample.db/sample_demo |
+-------+--------+------+--------+-------------------------+
```

Even though the queries do not refer to the S column containing the long value, all the sampling queries include the data file containing the column value X=1000, because the query cannot reach the 50% threshold (115 bytes) without including that file. The large file might be considered first, in which case it is the only file processed by the query. Or an arbitrary set of other files might be considered first.

```
select distinct x from sample_demo tablesample system(50);
+------+
| x    |
+------+
| 1000 |
| 3    |
| 1    |
+------+

select distinct x from sample_demo tablesample system(50);
+------+
| x    |
+------+
| 1000 |
+------+

select distinct x from sample_demo tablesample system(50);
+------+
| x    |
+------+
| 1000 |
| 4    |
| 2    |
| 1    |
+------+
```

The following examples demonstrate how the TABLESAMPLE clause interacts with other table aspects, such as partitioning and file format:

```
create table sample_demo_partitions (x int, s string) partitioned by (n int)
 stored as parquet;
insert into sample_demo_partitions partition (n = 1) select * from sample_d
emo;
insert into sample_demo_partitions partition (n = 2) select * from sample_d
emo;
insert into sample_demo_partitions partition (n = 3) select * from sample_d
emo;

show files in sample_demo_partitions;
+--------------------------------+--------+-----------+
| Path                           | Size   | Partition |
+--------------------------------+--------+-----------+
| 000000_364262785_data.0.parq   | 1.24KB | n=1       |
| 000001_973526736_data.0.parq   | 566B   | n=1       |
| 0000000_1300598134_data.0.parq | 1.24KB | n=2       |
| 0000001_689099063_data.0.parq  | 568B   | n=2       |
| 0000000_1861371709_data.0.parq | 1.24KB | n=3       |
| 0000001_1065507912_data.0.parq | 566B   | n=3       |
+--------------------------------+--------+-----------+
show table stats tablesample_demo_partitioned;
+-------+-------+--------+--------+---------+------------------------------
---------------+
| n     | #Rows | #Files | Size   | Format  | Location
               |
+-------+-------+--------+--------+---------+----------------------------
-----------------+
```

```
| 1     | -1    | 2      | 1.79KB | PARQUET | /tsample.db/tablesample_dem
o_partitioned/n=1 |
| 2     | -1    | 2      | 1.80KB | PARQUET | /tsample.db/tablesample_demo_p
artitioned/n=2 |
| 3     | -1    | 2      | 1.79KB | PARQUET | /tsample.db/tablesample_demo_
partitioned/n=3 |
| Total | -1    | 6      | 5.39KB |         |
                |
+-------+-------+--------+--------+---------+---------------------------
------------------+
```

If the query does not involve any partition pruning, the sampling applies to the data volume of the entire table:

```
-- 18 rows total.
select count(*) from sample_demo_partitions;
+----------+
| count(*) |
+----------+
| 18       |
+----------+
-- The number of rows per data file is not
-- perfectly balanced, therefore the count
-- is different depending on which set of files
-- is considered.
select count(*) from sample_demo_partitions
  tablesample system(75);
+----------+
| count(*) |
+----------+
| 14       |
+----------+
select count(*) from sample_demo_partitions
  tablesample system(75);
+----------+
| count(*) |
+----------+
| 16       |
+----------+
```

If the query only processes certain partitions, the query computes the sampling threshold based on the data size and set of files only from the relevant partitions:

```
select count(*) from sample_demo_partitions
  tablesample system(50) where n = 1;
+----------+
| count(*) |
+----------+
| 6        |
+----------+

select count(*) from sample_demo_partitions
  tablesample system(50) where n = 1;
+----------+
| count(*) |
+----------+
| 2        |
+----------+
```

Related Information

COMPUTE STATS statement

WITH clause

A clause that can be added before a SELECT statement, to define aliases for complicated expressions that are referenced multiple times within the body of the SELECT. Similar to CREATE VIEW, except that the table and column names defined in the WITH clause do not persist after the query finishes, and do not conflict with names used in actual tables or views. Also known as “subquery factoring”.

You can rewrite a query using subqueries to work the same as with the WITH clause. The purposes of the WITH clause are:

- Convenience and ease of maintenance from less repetition with the body of the query. Typically used with queries involving UNION, joins, or aggregation functions where the similar complicated expressions are referenced multiple times.
- SQL code that is easier to read and understand by abstracting the most complex part of the query into a separate block.
- Improved compatibility with SQL from other database systems that support the same clause (primarily Oracle Database).

Note:

The Impala WITH clause does not support recursive queries in the WITH, which is supported in some other database systems.

Standards compliance: Introduced in SQL:1999.

Examples:

```
-- Define 2 subqueries that can be referenced from the body of a longer quer
y.
with t1 as (select 1), t2 as (select 2) insert into tab select * from t1
union all select * from t2;
-- Define one subquery at the outer level, and another at the inner level as
 part of the
-- initial stage of the UNION ALL query.
with t1 as (select 1) (with t2 as (select 2) select * from t2) union all
select * from t1;
```

DISTINCT operator

The DISTINCT operator in a SELECT statement filters the result set to remove duplicates.

```
-- Returns the unique values from one column.
-- NULL is included in the set of values if any rows have a NULL in this c
olumn.
SELECT DISTINCT c_birth_country FROM customer;

-- Returns the unique combinations of values from multiple columns.
SELECT DISTINCT c_salutation, c_last_name FROM customer;
```

You can use DISTINCT in combination with an aggregation function, typically COUNT(), to find how many different values a column contains.

```
-- Counts the unique values from one column.
-- NULL is not included as a distinct value in the count.
SELECT COUNT(DISTINCT c_birth_country) FROM customer;

-- Counts the unique combinations of values from multiple columns.
SELECT COUNT(DISTINCT c_salutation, c_last_name) FROM customer;
```

Zero-length strings: For purposes of clauses such as DISTINCT and GROUP BY, Impala considers zero-length strings (""), NULL, and space to all be different values.

Note:

In contrast with some database systems that always return DISTINCT values in sorted order, Impala does not do any ordering of DISTINCT values. Always include an ORDER BY clause if you need the values in alphabetical or numeric sorted order.

SET statement

The SET statement specifies values for query options that control the runtime behavior of other statements within the same session.

When issued in impala-shell, the SET command is interpreted as an impala-shell command that has differences from the SQL SET statement.

Syntax:

```
SET
SET ALL
SET QUERY_OPTION=OPTION_VALUE
SET QUERY_OPTION=""
```

SET and SET ALL with no arguments return a result set consisting of all the applicable query options and their current values.

The *QUERY_OPTION* and *OPTION_VALUE* are case-insensitive.

Unlike the impala-shell command version of SET, when used as a SQL statement, the string values for *OPTION_VALUE* need to be quoted, e.g. SET option="new_value".

The SET *QUERY_OPTION* = "" statement unsets the value of the *QUERY_OPTION* in the current session, reverting it to the default state. In impala-shell, use the UNSET command to set a query option to its default.

Each query option has a specific allowed notation for its arguments. See *Query options for the SET statement* for the details of each query option.

Usage notes:

The options are divided into groups: Regular Query Options, Advanced Query Options, Development Query Op tions, and Deprecated Query Options.

- The advanced options are intended for use in specific kinds of performance tuning and debugging scenarios.
- The development options are related to internal development of Impala or features that are not yet finalized. These options might be changed or removed without notice.
- The deprecated options are related to features that are removed or changed so that the options no longer have any purpose. These options might be removed in future versions.
- By default, only the first two groups, regular and advanced, are displayed by the SET command. Use SET ALL to see all groups of options.
- `impala-shell` options and user-specified variables are always displayed at the end of the list of query options, after all appropriate option groups.

SET has always been available as an `impala-shell` command. Promoting it to a SQL statement lets you use this feature in client applications through the JDBC and ODBC APIs.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Cloudera Manager: You can set any of the query options globally in Cloudera Manager to affect all the queries in the cluster.

1. Navigate to *IMPALA CLUSTER* > Configuration > Impala Daemon Query Options Advanced Configuration Snippet (Safety Valve)
2. In the field, type a key-value pair of a query option and the value, e.g. EXPLAIN_LEVEL=2.

3. Click + to enter an additional option.
4. Clive Save Changes.

Related Information

Query options
SET command in impala-shell

SHOW statement

Use the SHOW statement to get information about different types of Impala objects.

Syntax:

```
SHOW DATABASES [[LIKE] 'PATTERN']
SHOW SCHEMAS [[LIKE] 'PATTERN'] - an alias for SHOW DATABASES
SHOW TABLES [IN DATABASE_NAME] [[LIKE] 'PATTERN']
SHOW [AGGREGATE | ANALYTIC] FUNCTIONS [IN DATABASE_NAME] [[LIKE] 'PATTERN']
SHOW CREATE TABLE [DATABASE_NAME].TABLE_NAME
SHOW CREATE VIEW [DATABASE_NAME].VIEW_NAME
SHOW TABLE STATS [DATABASE_NAME.]TABLE_NAME
SHOW COLUMN STATS [DATABASE_NAME.]TABLE_NAME
SHOW [RANGE] PARTITIONS [DATABASE_NAME.]TABLE_NAME
SHOW FILES IN [DATABASE_NAME.]TABLE_NAME [PARTITION (KEY_COL_EXPRESSION
 [, KEY_COL_EXPRESSION]]
SHOW VIEWS IN functional LIKE 'alltypes*'
SHOW VIEWS [IN database_name] [[LIKE] 'pattern']

SHOW GRANT USER USER_NAME ON SERVER
SHOW GRANT USER USER_NAME ON DATABASE DATABASE_NAME
SHOW GRANT USER USER_NAME ON TABLE TABLE_NAME
SHOW GRANT USER USER_NAME ON TABLE [DATABASE_NAME].TABLE_NAME.COLUMN_NAME
SHOW GRANT USER USER_NAME ON URI URI


SHOW GRANT GROUP GROUP_NAME ON SERVER
SHOW GRANT GROUP GROUP_NAME ON DATABASE DATABASE_NAME
SHOW GRANT GROUP GROUP_NAME ON TABLE TABLE_NAME
SHOW GRANT GROUP GROUP_NAME ON TABLE [DATABASE_NAME].TABLE_NAME.COLUMN_NAME
SHOW GRANT GROUP GROUP_NAME ON URI URI
```

Note: SHOW GRANT USER/GROUP is not supported without specifying a resource.

Issue a SHOW *OBJECT_TYPE* statement to see the appropriate objects in the current database, or SHOW *OBJECT_TYPE* IN *DATABASE_NAME* to see objects in a specific database.

The optional *PATTERN* argument is a quoted string literal, using Unix-style * wildcards and allowing | for alternation. The preceding LIKE keyword is also optional. All object names are stored in lowercase, so use all lowercase letters in the pattern string. For example:

```
show databases 'a*';
show databases like 'a*';
show tables in some_db like '*fact*';
use some_db;
show tables '*dim*|*fact*';
```

Cancellation: Cannot be cancelled.

SHOW FILES statement

The SHOW FILES statement displays the files that constitute a specified table, or a partition within a partitioned table. This syntax is available in Impala 2.2 and higher only. The output includes the names of the files, the size of each file, and the applicable partition for a partitioned table. The size includes a suffix of B for bytes, MB for megabytes, and GB for gigabytes.

In Impala 2.8 and higher, you can use general expressions with operators such as <, IN, LIKE, and BETWEEN in the PARTITION clause, instead of only equality operators. For example:

```
show files in sample_table partition (j < 5);
show files in sample_table partition (k = 3, l between 1 and 10);
show files in sample_table partition (month like 'J%');
```

Note: This statement applies to tables and partitions stored on HDFS, or in the Amazon Simple Storage System (S3). It does not apply to views. It does not apply to tables mapped onto HBase or Kudu, because those data management systems do not use the same file-based storage layout.

Usage notes:

You can use this statement to verify the results of your ETL process: that is, that the expected files are present, with the expected sizes. You can examine the file information to detect conditions such as empty files, missing files, or inefficient layouts due to a large number of small files. When you use INSERT statements to copy from one table to another, you can see how the file layout changes due to file format conversions, compaction of small input files into large data blocks, and multiple output files from parallel queries and partitioned inserts.

The output from this statement does not include files that Impala considers to be hidden or invisible, such as those whose names start with a dot or an underscore, or that end with the suffixes .copying or .tmp.

The information for partitioned tables complements the output of the SHOW PARTITIONS statement, which summarizes information about each partition. SHOW PARTITIONS produces some output for each partition, while SHOW FILES does not produce any output for empty partitions because they do not include any data files.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read permission for all the table files, read and execute permission for all the directories that make up the table, and execute permission for the database directory and all its parent directories.

SHOW GRANT USER statement

The SHOW GRANT USER statement shows the list of privileges for a given user. This statement is only allowed for administrative users. However, the current user can run SHOW GRANT USER for themselves.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

SHOW DATABASES

The SHOW DATABASES statement is often the first one you issue when connecting to an instance for the first time. You typically issue SHOW DATABASES to see the names you can specify in a USE *DB_NAME* statement, then after switching to a database you issue SHOW TABLES to see the names you can specify in SELECT and INSERT statements.

In Impala 2.5 and higher, the output includes a second column showing any associated comment for each database.

The output of SHOW DATABASES includes the special _impala_builtins database, which lets you view definitions of built-in functions, as described under SHOW FUNCTIONS.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

SHOW TABLES statement

Displays the names of tables. By default, lists tables in the current database, or with the IN clause, in a specified database. By default, lists all tables, or with the LIKE clause, only those whose name match a pattern with * wildcards.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all directories that are part of the table. (A table could span multiple different HDFS directories if it is partitioned. The directories could be widely scattered because a partition can reside in an arbitrary HDFS directory based on its LOCATION attribute.)

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

SHOW CREATE TABLE statement

As a schema changes over time, you might run a CREATE TABLE statement followed by several ALTER TABLE statements. To capture the cumulative effect of all those statements, SHOW CREATE TABLE displays a CREATE TABLE statement that would reproduce the current structure of a table. You can use this output in scripts that set up or clone a group of tables, rather than trying to reproduce the original sequence of CREATE TABLE and ALTER TABLE statements. When creating variations on the original table, or cloning the original table on a different system, you might need to edit the SHOW CREATE TABLE output to change things such as the database name, LOCATION field, and so on that might be different on the destination system.

If you specify a view name in the SHOW CREATE TABLE, it returns a CREATE VIEW statement with column names and the original SQL statement to reproduce the view. You need the VIEW_METADATA privilege on the view and SELECT privilege on all underlying views and tables to successfully run the SHOW CREATE VIEW statement for a view. The SHOW CREATE VIEW is available as an alias for SHOW CREATE TABLE.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Kudu considerations:

For Kudu tables:

- The column specifications include attributes such as NULL, NOT NULL, ENCODING, and COMPRESSION. If you do not specify those attributes in the original CREATE TABLE statement, the SHOW CREATE TABLE output displays the defaults that were used.
- The specifications of any RANGE clauses are not displayed in full. To see the definition of the range clauses for a Kudu table, use the SHOW RANGE PARTITIONS statement.
- The TBLPROPERTIES output reflects the Kudu master address and the internal Kudu name associated with the Impala table.

SHOW CREATE VIEW statement

The SHOW CREATE VIEW, it returns a CREATE VIEW statement with column names and the original SQL statement to reproduce the view. You need the VIEW_METADATA privilege on the view and SELECT privilege on all underlying views and tables to successfully run the SHOW CREATE VIEW statement for a view.

The SHOW CREATE VIEW is an alias for SHOW CREATE TABLE.

SHOW VIEWS statement

The SHOW VIEWSstatement streamlines the process of listing all views within a specified schema/database. This command proves invaluable for swiftly identifying and reviewing views, contributing to optimized performance by minimizing metadata scan operations.

Usage:

- To list views within a specific schema/database.
- If no schema is specified, the command returns views from the current schema database.

Before you begin:

Before executing the SHOW VIEWS command, it is crucial to ensure that the catalogd is initiated with the --pull_table_types_and_comments=true flag. This setting is pivotal for instructing catalogd to consistently load table types on a per-table basis, significantly enhancing performance. By default, this flag is set to FALSE.

Enabling the --pull_table_types_and_comments=true flag prompts catalogd to load table types and comments during startup and when executing INVALIDATE METADATA commands, ensuring that the metadata remains up-to-date.

These steps are integral to leveraging the efficiency and functionality of the SHOW VIEWS command while maintaining optimal performance through thoughtful metadata management.

Examples:

```
SHOW VIEWS IN functional LIKE 'alltypes*'
SHOW VIEWS [IN database_name] [[LIKE] 'pattern']
```

SHOW TABLE STATS statement

The SHOW TABLE STATS and SHOW COLUMN STATS variants are important for tuning performance and diagnosing performance issues, especially with the largest tables and the most complex join queries.

Any values that are not available (because the COMPUTE STATS statement has not been run yet) are displayed as -1.

SHOW TABLE STATS provides some general information about the table, such as the number of files, overall size of the data, whether some or all of the data is in the HDFS cache, and the file format, that is useful whether or not you have run the COMPUTE STATS statement. A -1 in the #Rows output column indicates that the COMPUTE STATS statement has never been run for this table. If the table is partitioned, SHOW TABLE STATS provides this information for each partition. (It produces the same output as the SHOW PARTITIONS statement in this case.)

The output of SHOW COLUMN STATS is primarily only useful after the COMPUTE STATS statement has been run on the table. A -1 in the #Distinct Values output column indicates that the COMPUTE STATS statement has never been run for this table. Currently, Impala always leaves the #Nulls column as -1, even after COMPUTE STATS has been run.

These SHOW statements work on actual tables only, not on views.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

Kudu considerations:

Because Kudu tables do not have characteristics derived from HDFS, such as number of files, file format, and HDFS cache status, the output of SHOW TABLE STATS reflects different characteristics that apply to Kudu tables. If the Kudu table is created with the clause PARTITIONS 20, then the result set of SHOW TABLE STATS consists of 20 rows, each representing one of the numbered partitions.

Impala does not compute the number of rows for each partition for Kudu tables. Therefore, you do not need to re-run COMPUTE STATS when you see -1 in the # Rows column of the output from SHOW TABLE STATS. That column always shows -1 for all Kudu tables.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all directories that are part of the table. (A table could span multiple different HDFS directories if it is partitioned. The directories could be widely scattered because a partition can reside in an arbitrary HDFS directory based on its LOCATION attribute.) The Impala user must also have execute permission for the database directory, and any parent directories of the database directory in HDFS.

SHOW COLUMN STATS statement

The SHOW TABLE STATS and SHOW COLUMN STATS variants are important for tuning performance and diagnosing performance issues, especially with the largest tables and the most complex join queries.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

Kudu considerations:

The output for SHOW COLUMN STATS includes the relevant information for Kudu tables. The information for column statistics that originates in the underlying Kudu storage layer is also represented in the metastore database that Impala uses.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all directories that are part of the table. (A table could span multiple different HDFS directories if it is partitioned. The directories could be widely scattered because a partition can reside in an arbitrary HDFS directory based on its LOCATION attribute.) The Impala user must also have execute permission for the database directory, and any parent directories of the database directory in HDFS.

SHOW PARTITIONS statement

SHOW PARTITIONS displays information about each partition for a partitioned table. (The output is the same as the SHOW TABLE STATS statement, but SHOW PARTITIONS only works on a partitioned table.) Because it displays table statistics for all partitions, the output is more informative if you have run the COMPUTE STATS statement after creating all the partitions. For example, on a CENSUS table partitioned on the YEAR column:

Because Kudu tables are all considered to be partitioned, the SHOW PARTITIONS statement works for any Kudu table. The default output is the same as for SHOW TABLE STATS, with the same Kudu-specific columns in the result set.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

Kudu considerations:

The optional RANGE clause only applies to Kudu tables. It displays only the partitions defined by the RANGE clause of CREATE TABLE or ALTER TABLE.

Although you can specify < or <= comparison operators when defining range partitions for Kudu tables, Kudu rewrites them if necessary to represent each range as *LOW_BOUND* <= VALUES < *HIGH_BOUND*. This rewriting might involve incrementing one of the boundary values or appending a \0 for string values, so that the partition covers the same range as originally specified.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all directories that are part of the table. (A table could span multiple different HDFS directories if it is partitioned. The directories could be widely scattered because a partition can reside in an arbitrary HDFS directory based on its LOCATION attribute.) The Impala user must also have execute permission for the database directory, and any parent directories of the database directory in HDFS.

SHOW FUNCTIONS statement

By default, SHOW FUNCTIONS displays user-defined functions (UDFs) and SHOW AGGREGATE FUNCTIONS displays user-defined aggregate functions (UDAFs) associated with a particular database. The output from SHOW FUNCTIONS includes the argument signature of each function. You specify this argument signature as part of the DROP FUNCTION statement. You might have several UDFs with the same name, each accepting different argument data types.

Usage notes:

In Impala 2.5 and higher, the SHOW FUNCTIONS output includes a new column, labelled is persistent. This property is true for Impala built-in functions, C++ UDFs, and Java UDFs created using the new CREATE FUNCTION syntax with no signature. It is false for Java UDFs created using the old CREATE FUNCTION syntax that includes the types for the arguments and return value. Any functions with false shown for this property must be created again by the CREATE FUNCTION statement each time the Impala catalog server is restarted. See CREATE FUNCTION for information on switching to the new syntax, so that Java UDFs are preserved across restarts. Java UDFs that are persisted this way are also easier to share across Impala and Hive.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

SHOW ROLES statement

The SHOW ROLES statement displays all the roles created in the Ranger server.

Syntax

```
SHOW ROLES
```

Note: Only a Ranger administrator is allowed to run this statement.

SHOW CURRENT ROLES statement

The SHOW CURRENT ROLES statement displays the roles assigned to the groups of the requesting user.

Syntax

```
SHOW ROLES
```

SHOW ROLE GRANT GROUP statement

The SHOW ROLE GRANT GROUP statement displays the roles assigned to the group.

Syntax

```
SHOW ROLE GRANT GROUP GROUP_NAME
```

Note: To run this statement, the requesting user must belong to the group <group>, or the requesting user has to be a Ranger administrator.

SHUTDOWN statement

The SHUTDOWN statement performs a graceful shutdown of Impala Daemon. The Impala daemon will notify other Impala daemons that it is shutting down, wait for a grace period, then shut itself down once no more queries or fragments are executing on that daemon. The --shutdown_grace_period_s flag determines the duration of the grace period in seconds.

Syntax:

```
:SHUTDOWN()
:SHUTDOWN([HOST_NAME[:PORT_NUMBER] )
:SHUTDOWN(DEADLINE)
:SHUTDOWN([HOST_NAME[:PORT_NUMBER], DEADLINE)
```

Usage notes:

All arguments are optional for SHUTDOWN.

Argument	Type	Default	Description
HOST_NAME	STRING	The current impalad host to whom the SHUTDOWN statement is submitted.	Address of the impalad to be shut down.

Argument	Type	Default	Description
PORT_NUMBER	INT	• In Impala 3.1, the current impalad's port used for the thrift based communication with other impalads (by default, 22000). • In Impala 3.2 and higher, the current impalad's port used for the KRPC based communication with other impalads (by default, 27000).	Specifies the port by which the impalad can be contacted. • In Impala 3.1, use the same impalad port used for the thrift based inter-Impala communication. • In Impala 3.2 and higher, use the same impalad port used for the KRPC based inter-Impala communication.
DEADLINE	INT	The value of the ##shutdown_deadline_s flag, which defaults to 1 hour.	*DEADLINE* must be a non-negative number, specified in seconds. The value, 0, for *DEADLINE* specifies an immediate shutdown.

Take the following points into consideration when running the SHUTDOWN statement:

- A client can shut down the coordinator impalad that it is connected to via :SHUTDOWN().
- A client can remotely shut down any impalad via :SHUTDOWN('*HOSTNAME*').
- The shutdown time limit can be overridden to force a quicker or slower shutdown by specifying a deadline. The default deadline is determined by the --shutdown_deadline_s flag, which defaults to 1 hour.
- The executors can be shut down without disrupting running queries. Short-running queries will finish, and long-running queries will continue until the *DEADLINE* is reached.
- If queries are submitted to a coordinator after shutdown of that coordinator has started, they will fail.
- Long running queries or other issues, such as stuck fragments, will slow down but not prevent eventual shutdown.

Security considerations:

The ALL privilege is required on the server.

Cancellation: Cannot be cancelled.

Examples:

```
:SHUTDOWN(); -- Shut down the current impalad  with the default deadline.
:SHUTDOWN('hostname'); --  Shut down impalad running on hostname  with the
 default deadline.
:SHUTDOWN(\"hostname:1234\"); -- Shut down impalad running on host at port
 1234  with the default deadline.
:SHUTDOWN(10); - Shut down the current impalad after 10 seconds.
:SHUTDOWN('hostname', 10); - Shut down impalad running on hostname when all
queries running on hostname finish, or after 10 seconds.
:SHUTDOWN('hostname:11', 10 * 60); -- Shut down impalad running on hostname
 at port 11 when all queries running on hostname finish, or after 600 second
s.
:SHUTDOWN(0); -- Perform an immdediate shutdown of the current impalad.
```

Added in: Impala 3.1

TRUNCATE TABLE statement

The TRUNCATE TABLE statement removes the data from an Impala table while leaving the table itself.

Syntax:

```
TRUNCATE [TABLE] [IF EXISTS] [DB_NAME.]TABLE_NAME
```

Statement type: DDL

Usage notes:

Often used to empty tables that are used during ETL cycles, after the data has been copied to another table for the next stage of processing. This statement is a low-overhead alternative to dropping and recreating the table, or using INSERT OVERWRITE to replace the data during the next ETL cycle.

This statement removes all the data and associated data files in the table. It can remove data files from internal tables, external tables, partitioned tables, and tables mapped to HBase or the Amazon Simple Storage Service (S3). The data removal applies to the entire table, including all partitions of a partitioned table.

Any statistics produced by the COMPUTE STATS statement are reset when the data is removed.

Make sure that you are in the correct database before truncating a table, either by issuing a USE statement first or by using a fully qualified name *DB_NAME.TABLE_NAME*.

The optional TABLE keyword does not affect the behavior of the statement.

The optional IF EXISTS clause makes the statement succeed whether or not the table exists. If the table does exist, it is truncated; if it does not exist, the statement has no effect. This capability is useful in standardized setup scripts that are might be run both before and after some of the tables exist. This clause is available in Impala 2.5 and higher.

Amazon S3 considerations:

Although Impala cannot write new data to a table stored in the Amazon S3 filesystem, the TRUNCATE TABLE statement can remove data files from S3.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have write permission for all the files and directories that make up the table.

Kudu considerations:

Currently, the TRUNCATE TABLE statement cannot be used with Kudu tables.

Examples:

The following example shows a table containing some data and with table and column statistics. After the TRUN CATE TABLE statement, the data is removed and the statistics are reset.

```
CREATE TABLE truncate_demo (x INT);
INSERT INTO truncate_demo VALUES (1), (2), (4), (8);
SELECT COUNT(*) FROM truncate_demo;
+----------+
| count(*) |
+----------+
| 4        |
+----------+
COMPUTE STATS truncate_demo;
+------------------------------------------+
| summary                                  |
+------------------------------------------+
| Updated 1 partition(s) and 1 column(s). |
+------------------------------------------+
SHOW TABLE STATS truncate_demo;
+-------+--------+------+--------------+-------------------+--------+----
---------------+
| #Rows | #Files | Size | Bytes Cached | Cache Replication | Format | Incre
mental stats |
+-------+--------+------+--------------+-------------------+--------+----
---------------+
| 4     | 1      | 8B   | NOT CACHED   | NOT CACHED        | TEXT   | false
           |
```

```
+-------+--------+------+--------------+-------------------+--------+----
---------------+
SHOW COLUMN STATS truncate_demo;
+--------+------+------------------+--------+----------+----------+
| Column | Type | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+------+------------------+--------+----------+----------+
| x      | INT  | 4                | -1     | 4        | 4        |
+--------+------+------------------+--------+----------+----------+

-- After this statement, the data and the table/column stats will be gone.
TRUNCATE TABLE truncate_demo;
SELECT COUNT(*) FROM truncate_demo;
+----------+
| count(*) |
+----------+
| 0        |
+----------+
SHOW TABLE STATS truncate_demo;
+-------+--------+------+--------------+-------------------+--------+-----
--------------+
| #Rows | #Files | Size | Bytes Cached | Cache Replication | Format | Increm
ental stats |
+-------+--------+------+--------------+-------------------+--------+-----
--------------+
| -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT   | false
            |
+-------+--------+------+--------------+-------------------+--------+-----
--------------+
SHOW COLUMN STATS truncate_demo;
+--------+------+------------------+--------+----------+----------+
| Column | Type | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+------+------------------+--------+----------+----------+
| x      | INT  | -1               | -1     | 4        | 4        |
+--------+------+------------------+--------+----------+----------+
```

The following example shows how the IF EXISTS clause allows the TRUNCATE TABLE statement to be run without error whether or not the table exists:

```
CREATE TABLE staging_table1 (x INT, s STRING);
Fetched 0 row(s) in 0.33s

SHOW TABLES LIKE 'staging*';
+----------------+
| name           |
+----------------+
| staging_table1 |
+----------------+
Fetched 1 row(s) in 0.25s
-- Our ETL process involves removing all data from several staging tables
-- even though some might be already dropped, or not created yet.

TRUNCATE TABLE IF EXISTS staging_table1;
Fetched 0 row(s) in 5.04s

TRUNCATE TABLE IF EXISTS staging_table2;
Fetched 0 row(s) in 0.25s

TRUNCATE TABLE IF EXISTS staging_table3;
Fetched 0 row(s) in 0.25s
```

TRUNCATE on Insert-Only Transactional Tables

The TRUNCATE table statement on Insert-Only Transactional tables creates new empty ACID base directories and does not remove the files. To maintain transactional isolation TRUNCATE statement will NOT delete the previous base and delta directories.

Related Information

Impala with Amazon S3

UPDATE statement

The UPDATE statement updates an arbitrary number of rows in a Kudu table. This statement only works for Impala tables that use the Kudu storage engine.

Syntax:

```
UPDATE [DATABASE_NAME.]TABLE_NAME SET COL = VAL [, COL = VAL ... ]
  [ FROM JOINED_TABLE_REFS ]
  [ WHERE WHERE_CONDITIONS ]
```

Usage notes:

None of the columns that make up the primary key can be updated by the SET clause.

The conditions in the WHERE clause are the same ones allowed for the SELECT statement.

If the WHERE clause is omitted, all rows in the table are updated.

The conditions in the WHERE clause can refer to any combination of primary key columns or other columns. Referring to primary key columns in the WHERE clause is more efficient than referring to non-primary key columns.

Because Kudu currently does not enforce strong consistency during concurrent DML operations, be aware that the results after this statement finishes might be different than you intuitively expect:

- If some rows cannot be updated because their some primary key columns are not found, due to their being deleted by a concurrent DELETE operation, the statement succeeds but returns a warning.
- An UPDATE statement might also overlap with INSERT, UPDATE, or UPSERT statements running concurrently on the same table. After the statement finishes, there might be more or fewer matching rows than expected in the table because it is undefined whether the UPDATE applies to rows that are inserted or updated while the UPDATE is in progress.

The number of affected rows is reported in an `impala-shell` message and in the query profile.

The optional FROM clause lets you restrict the updates to only the rows in the specified table that are part of the result set for a join query. The join clauses can include non-Kudu tables, but the table from which the rows are deleted must be a Kudu table.

Statement type: DML

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMPUTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

The following examples show how to perform a simple update on a table, with or without a WHERE clause:

```
-- Set all rows to the same value for column c3.
-- In this case, c1 and c2 are primary key columns
```

```
-- and so cannot be updated.
UPDATE kudu_table SET c3 = 'not applicable';

-- Update only the rows that match the condition.
UPDATE kudu_table SET c3 = NULL WHERE c1 > 100 AND c3 IS NULL;

-- Does not update any rows, because the WHERE condition is always false.
UPDATE kudu_table SET c3 = 'impossible' WHERE 1 = 0;

-- Change the values of multiple columns in a single UPDATE statement.
UPDATE kudu_table SET c3 = upper(c3), c4 = FALSE, c5 = 0 WHERE c6 = TRUE;
```

The following examples show how to perform an update using the FROM keyword with a join clause:

```
-- Uppercase a column value, only for rows that have
-- an ID that matches the value from another table.
UPDATE kudu_table SET c3 = upper(c3)
  FROM kudu_table JOIN non_kudu_table
  ON kudu_table.id = non_kudu_table.id;

-- Same effect as previous statement.
-- Assign table aliases in FROM clause, then refer to
-- short names elsewhere in the statement.
UPDATE t1 SET c3 = upper(c3)
  FROM kudu_table t1 JOIN non_kudu_table t2
  ON t1.id = t2.id;

-- Same effect as previous statements, but more efficient.
-- Use WHERE clause to skip updating values that are
-- already uppercase.
UPDATE t1 SET c3 = upper(c3)
  FROM kudu_table t1 JOIN non_kudu_table t2
  ON t1.id = t2.id
  WHERE c3 != upper(c3);
```

UPSERT statement

The UPSERT statement acts as a combination of the INSERT and UPDATE statements.

For each row processed by the UPSERT statement:

- If another row already exists with the same set of primary key values, the other columns are updated to match the values from the row being “UPSERTed”.
- If there is not any row with the same set of primary key values, the row is created, the same as if the INSERT statement was used.

This statement only works for Impala tables that use the Kudu storage engine.

Syntax:

```
UPSERT [HINT_CLAUSE] INTO [TABLE] [DB_NAME.]TABLE_NAME
  [(COLUMN_LIST)]
{
    [HINT_CLAUSE] SELECT_STATEMENT
  | VALUES (VALUE [, VALUE ...]) [, (VALUE [, VALUE ...]) ...]
}

hint_clause ::= [SHUFFLE] | [NOSHUFFLE]
  (Note: the square brackets are part of the syntax.)
```

The *SELECT_STATEMENT* clause can use the full syntax, such as WHERE and JOIN clauses.

Statement type: DML

Usage notes:

If you specify a column list, any omitted columns in the inserted or updated rows are set to their default value (if the column has one) or NULL (if the column does not have a default value). Therefore, if a column is not nullable and has no default value, it must be included in the column list for any UPSERT statement. Because all primary key columns meet these conditions, all the primary key columns must be specified in every UPSERT statement.

Because Kudu tables can efficiently handle small incremental changes, the VALUES clause is more practical to use with Kudu tables than with HDFS-based tables.

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMPUTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

```
UPSERT INTO kudu_table (pk, c1, c2, c3) VALUES (0, 'hello', 50, true), (1,
'world', -1, false);
UPSERT INTO production_table SELECT * FROM staging_table;
UPSERT INTO production_table SELECT * FROM staging_table WHERE c1 IS NOT
NULL AND c2 > 0;
```

USE statement

The USE statement switches the current session to a specified database. The *current database* is where any CREATE TABLE, INSERT, SELECT, or other statements act when you specify a table or other object name, without prefixing it with a database name. The new current database applies for the duration of the session or unti another USE statement is executed.

Syntax:

```
USE DB_NAME
```

By default, when you connect to an Impala instance, you begin in a database named default.

Usage notes:

Switching the default database is convenient in the following situations:

- To avoid qualifying each reference to a table with the database name. For example, SELECT * FROM t1 JOIN t2 rather than SELECT * FROM db.t1 JOIN db.t2.
- To do a sequence of operations all within the same database, such as creating a table, inserting data, and querying the table.

To start the `impala-shell` interpreter and automatically issue a USE statement for a particular database, specify the option -d *DB_NAME* for the `impala-shell` command. The -d option is useful to run SQL scripts, such as setup or test scripts, against multiple databases without hardcoding a USE statement into the SQL source.

Examples:

See the *CREATE DATABASE* statement for examples covering CREATE DATABASE, USE, and DROP DATABASE.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Related Information

CREATE DATABASE statement

VALUES statement

The VALUES clause can be used as stand-alone statement, in the INSERT statement, and in the SELECT statement to construct a data set.

Syntax:

```
VALUES (ROW)[, (ROW), ...];

SELECT SELECT_LIST FROM (VALUES (ROW)[, (ROW), ...]) AS ALIAS;

ROW ::= COLUMN [[AS ALIAS], COLUMN [AS ALIAS], ...]
```

- The VALUES keyword is followed by a comma separated list of one or more *ROW*s.
- *ROW* is a comma-separated list of one or more *COLUMN*s.
- Each *ROW* must have the same number of *COLUMN*s.
- *COLUMN* can be a constant, a variable, or an expression.
- The corresponding *COLUMN*s must have compatible data types in all *ROW*s. See the third query in the Examples section below.
- By default, the first row is used to name columns. But using the AS keyword, you can optionally give the column an *ALIAS*.
- If used in the SELECT statement, the AS keyword with an *ALIAS* is required.
- *SELECT_LIST* is the columns to be selected for the result set.

Examples:

```
> SELECT * FROM (VALUES(4,5,6),(7,8,9)) AS t;
+---+---+---+
| 4 | 5 | 6 |
+---+---+---+
| 4 | 5 | 6 |
| 7 | 8 | 9 |
+---+---+---+

> SELECT * FROM (VALUES(1 AS c1, true AS c2, 'abc' AS c3),(100,false,'xyz
')) AS t;
+-----+-------+-----+
| c1  | c2    | c3  |
+-----+-------+-----+
| 1   | true  | abc |
| 100 | false | xyz |
+-----+-------+-----+

> VALUES (CAST('2019-01-01' AS TIMESTAMP)), ('2019-02-02');
+---------------------------------+
| cast('2019-01-01' as timestamp) |
+---------------------------------+
| 2019-01-01 00:00:00             |
| 2019-02-02 00:00:00             |
+---------------------------------+
```

Related Information

SELECT statement

Optimizer hints in Impala

The Impala SQL dialect supports query hints, for fine-tuning the inner workings of queries. Specify hints as a temporary workaround for expensive queries, where missing statistics or other factors cause inefficient performance.

Hints are most often used for the most resource-intensive kinds of Impala queries:

- Join queries involving large tables, where intermediate result sets are transmitted across the network to evaluate the join conditions.
- Inserting into partitioned Parquet tables, where many memory buffers could be allocated on each host to hold intermediate results for each partition.

Syntax:

In Impala 2.0 and higher, you can specify the hints inside comments that use either the /* */ or -- notation. Specify a + symbol immediately before the hint name. Recently added hints are only available using the /* */ and -- notation. For clarity, the /* */ and -- styles are used in the syntax and examples throughout this section. With the /* */ or -- notation for hints, specify a + symbol immediately before the first hint name. Multiple hints can be specified separated by commas, for example /* +clustered,shuffle */

```
SELECT STRAIGHT_JOIN SELECT_LIST FROM
JOIN_LEFT_HAND_TABLE
  JOIN /* +BROADCAST|SHUFFLE */
JOIN_RIGHT_HAND_TABLE
REMAINDER_OF_QUERY;

SELECT SELECT_LIST FROM
JOIN_LEFT_HAND_TABLE
  JOIN -- +BROADCAST|SHUFFLE
JOIN_RIGHT_HAND_TABLE
REMAINDER_OF_QUERY;

INSERT INSERT_CLAUSES
  /* +SHUFFLE|NOSHUFFLE */
  SELECT REMAINDER_OF_QUERY;

INSERT INSERT_CLAUSES
  -- +SHUFFLE|NOSHUFFLE
  SELECT REMAINDER_OF_QUERY;

INSERT /* +SHUFFLE|NOSHUFFLE */
  INSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

INSERT -- +SHUFFLE|NOSHUFFLE
  INSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

UPSERT /* +SHUFFLE|NOSHUFFLE */
  UPSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

UPSERT -- +SHUFFLE|NOSHUFFLE
  UPSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;
```

```
SELECT SELECT_LIST
 FROM
TABLE_REF
  /* +{SCHEDULE_CACHE_LOCAL | SCHEDULE_DISK_LOCAL | SCHEDULE_REMOTE}
    [,RANDOM_REPLICA] */
REMAINDER_OF_QUERY;

INSERT INSERT_CLAUSES
  -- +CLUSTERED
  SELECT REMAINDER_OF_QUERY;

INSERT INSERT_CLAUSES
  /* +CLUSTERED */
  SELECT REMAINDER_OF_QUERY;

INSERT -- +CLUSTERED
  INSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

INSERT /* +CLUSTERED */
  INSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

UPSERT -- +CLUSTERED
  UPSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

UPSERT /* +CLUSTERED */
  UPSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

CREATE /* +SHUFFLE|NOSHUFFLE */
  TABLE_CLAUSES
  AS SELECT REMAINDER_OF_QUERY;

CREATE -- +SHUFFLE|NOSHUFFLE
  TABLE_CLAUSES
  AS SELECT REMAINDER_OF_QUERY;

CREATE /* +CLUSTERED|NOCLUSTERED */
  TABLE_CLAUSES
  AS SELECT REMAINDER_OF_QUERY;

CREATE -- +CLUSTERED|NOCLUSTERED
  TABLE_CLAUSES
  AS SELECT REMAINDER_OF_QUERY;
```

The square bracket style hints are supported for backward compatibility, but the syntax is deprecated and will be removed in a future release. For that reason, any newly added hints are not available with the square bracket syntax.

```
SELECT STRAIGHT_JOIN SELECT_LIST FROM
JOIN_LEFT_HAND_TABLE
  JOIN [{ /* +BROADCAST */ | /* +SHUFFLE */ }]
JOIN_RIGHT_HAND_TABLE
REMAINDER_OF_QUERY;

INSERT INSERT_CLAUSES
  [{ /* +SHUFFLE */ | /* +NOSHUFFLE */ }]
  [/* +CLUSTERED */]
  SELECT REMAINDER_OF_QUERY;

UPSERT [{ /* +SHUFFLE */ | /* +NOSHUFFLE */ }]
  [/* +CLUSTERED */]
```

```
UPSERT_CLAUSES
SELECT REMAINDER_OF_QUERY;
```

Usage notes:

With both forms of hint syntax, include the STRAIGHT_JOIN keyword immediately after the SELECT and any DISTINCT or ALL keywords to prevent Impala from reordering the tables in a way that makes the join-related hints ineffective.

The STRAIGHT_JOIN hint affects the join order of table references in the query block containing the hint. It does not affect the join order of nested queries, such as views, inline views, or WHERE-clause subqueries. To use this hint for performance tuning of complex queries, apply the hint to all query blocks that need a fixed join order.

To reduce the need to use hints, run the COMPUTE STATS statement against all tables involved in joins, or used as the source tables for INSERT ... SELECT operations where the destination is a partitioned Parquet table. Do this operation after loading data or making substantial changes to the data within each table. Having up-to-date statistics helps Impala choose more efficient query plans without the need for hinting.

To see which join strategy is used for a particular query, examine the EXPLAIN output for that query.

Hints for join queries:

The /* +BROADCAST */ and /* +SHUFFLE */ hints control the execution strategy for join queries. Specify one of the following constructs immediately after the JOIN keyword in a query:

- /* +SHUFFLE */ makes that join operation use the “partitioned” technique, which divides up corresponding rows from both tables using a hashing algorithm, sending subsets of the rows to other nodes for processing. (The keyword SHUFFLE is used to indicate a “partitioned join”, because that type of join is not related to “partitioned tables”.) Since the alternative “broadcast” join mechanism is the default when table and index statistics are unavailable, you might use this hint for queries where broadcast joins are unsuitable; typically, partitioned joins are more efficient for joins between large tables of similar size.
- /* +BROADCAST */ makes that join operation use the “broadcast” technique that sends the entire contents of the right-hand table to all nodes involved in processing the join. This is the default mode of operation when table and index statistics are unavailable, so you would typically only need it if stale metadata caused Impala to mistakenly choose a partitioned join operation. Typically, broadcast joins are more efficient in cases where one table is much smaller than the other. (Put the smaller table on the right side of the JOIN operator.)

Hints for INSERT ... SELECT and CREATE TABLE AS SELECT (CTAS):

When inserting into partitioned tables, such as using the Parquet file format, you can include a hint in the INSERT or CREATE TABLE AS SELECT(CTAS) statements to fine-tune the overall performance of the operation and its resource usage.

You would only use hints if an INSERT or CTAS into a partitioned table was failing due to capacity limits, or if such an operation was succeeding but with less-than-optimal performance.

- /* +SHUFFLE */ and /* +NOSHUFFLE */ Hints
  - /* +SHUFFLE */ adds an exchange node, before writing the data, which re-partitions the result of the SELE CT based on the partitioning columns of the target table. With this hint, only one node writes to a partition at a time, minimizing the global number of simultaneous writes and the number of memory buffers holding data for individual partitions. This also reduces fragmentation, resulting in fewer files. Thus it reduces overall resource usage of the INSERT or CTAS operation and allows some operations to succeed that otherwise would fail. It does involve some data transfer between the nodes so that the data files for a particular partition are all written on the same node.

    Use /* +SHUFFLE */ in cases where an INSERT or CTAS statement fails or runs inefficiently due to all nodes attempting to write data for all partitions.

    If the table is unpartitioned or every partitioning expression is constant, then /* +SHUFFLE */ will cause every write to happen on the coordinator node.

- /* +NOSHUFFLE */ does not add exchange node before inserting to partitioned tables and disables re-partitioning. So the selected execution plan might be faster overall, but might also produce a larger number of small data files or exceed capacity limits, causing the INSERT or CTAS operation to fail.

  Impala automatically uses the /* +SHUFFLE */ method if any partition key column in the source table, mentioned in the SELECT clause, does not have column statistics. In this case, use the /* +NOSHUFFLE */ hint if you want to override this default behavior.
- If column statistics are available for all partition key columns in the source table mentioned in the INSERT ... SELECT or CTAS query, Impala chooses whether to use the /* +SHUFFLE */ or /* +NOSHUFFLE */ technique based on the estimated number of distinct values in those columns and the number of nodes involved in the operation. In this case, you might need the /* +SHUFFLE */ or the /* +NOSHUFFLE */ hint to override the execution plan selected by Impala.

- /* +CLUSTERED */ and /* +NOCLUSTERED */ Hints
  - /* +CLUSTERED */ sorts data by the partition columns before inserting to ensure that only one partition is written at a time per node. Use this hint to reduce the number of files kept open and the number of buffers kept in memory simultaneously. This technique is primarily useful for inserts into Parquet tables, where the large block size requires substantial memory to buffer data for multiple output files at once. This hint is available in Impala 2.8 or higher.

    Starting in Impala 3.0, /* +CLUSTERED */ is the default behavior for HDFS tables.
  - /* +NOCLUSTERED */ does not sort by primary key before insert. This hint is available in Impala 2.8 or higher.

    Use this hint when inserting to Kudu tables.

    In the versions lower than Impala 3.0, /* +NOCLUSTERED */ is the default in HDFS tables.

Hint for Table cardinalities

Currently, as part of query planning, you use the 'COMPUTE STATS' command to compute table stats. Without these stats, a query plan may not be optimal. However, these stats may not be available, up to date, or valid. To workaround this problem, this release adds a new query hint 'TABLE_NUM_ROWS' to be used in cases where stats were missing or invalid. You can use this new hint for a HDFS or a Kudu table in a query as shown in the following example.

```
select col from t /* +TABLE_NUM_ROWS(1000) */;
```

Impala will use this value as table scanned rows when a table has no stats or has corrput stats. This hint value will not be valid if table stats is normal.

Kudu consideration:

Starting from Impala 2.9, the INSERT or UPSERT operations into Kudu tables automatically add an exchange and a sort node to the plan that partitions and sorts the rows according to the partitioning/primary key scheme of the target table (unless the number of rows to be inserted is small enough to trigger single node execution). Since Kudu partitions and sorts rows on write, pre-partitioning and sorting takes some of the load off of Kudu and helps large INSERT operations to complete without timing out. However, this default behavior may slow down the end-to-end performance of the INSERT or UPSERT operations. Starting from Impala 2.10, you can use the /* +NOCLUSTERED */ and /* +NOSHUFFLE */ hints together to disable partitioning and sorting before the rows are sent to Kudu. Additionally, since sorting may consume a large amount of memory, consider setting the MEM_LIMIT query option for those queries.

Hints for scheduling of scan ranges (HDFS data blocks or Kudu tablets):

The hints /* +SCHEDULE_CACHE_LOCAL */, /* +SCHEDULE_DISK_LOCAL */, and /* +SCHEDULE_REMOTE */ have the same effect as specifying the REPLICA_PREFERENCE query option with the respective option settings of CACHE_LOCAL, DISK_LOCAL, or REMOTE.

Specifying the replica preference as a query hint always overrides the query option setting.

The hint /* +RANDOM_REPLICA */ is the same as enabling the SCHEDULE_RANDOM_REPLICA query option.

You can use these hints in combination by separating them with commas, for example, /* +SCHEDULE_CACHE_LOCAL,RANDOM_REPLICA */.

Specifying either the SCHEDULE_RANDOM_REPLICA query option or the corresponding RANDOM_REPLICA query hint enables the random tie-breaking behavior when processing data blocks during the query.

Suggestions versus directives:

In early Impala releases, hints were always obeyed and so acted more like directives. Once Impala gained join order optimizations, sometimes join queries were automatically reordered in a way that made a hint irrelevant. Therefore, the hints act more like suggestions in Impala 1.2.2 and higher.

To force Impala to follow the hinted execution mechanism for a join query, include the STRAIGHT_JOIN keyword in the SELECT statement. When you use this technique, Impala does not reorder the joined tables at all, so you must be careful to arrange the join order to put the largest table (or subquery result set) first, then the smallest, second smallest, third smallest, and so on. This ordering lets Impala do the most I/O-intensive parts of the query using local reads on the DataNodes, and then reduce the size of the intermediate result set as much as possible as each subsequent table or subquery result set is joined.

Restrictions:

Queries that include subqueries in the WHERE clause can be rewritten internally as join queries. Currently, you cannot apply hints to the joins produced by these types of queries.

Because hints can prevent queries from taking advantage of new metadata or improvements in query planning, use them only when required to work around performance issues, and be prepared to remove them when they are no longer required, such as after a new Impala release or bug fix.

In particular, the /* +BROADCAST */ and /* +SHUFFLE */ hints are expected to be needed much less frequently in Impala 1.2.2 and higher, because the join order optimization feature in combination with the COMPUTE STATS statement now automatically choose join order and join mechanism without the need to rewrite the query and add hints.

Compatibility:

The hints embedded within -- comments are compatible with Hive queries. The hints embedded within /* */ comments or [ ] square brackets are not recognized by or not compatible with Hive. For example, Hive raises an error for Impala hints within /* */ comments because it does not recognize the Impala hint names.

Considerations for views:

If you use a hint in the query that defines a view, the hint is preserved when you query the view. Impala internally rewrites all hints in views to use the -- comment notation, so that Hive can query such views without errors due to unrecognized hint names.

Examples:

For example, this query joins a large customer table with a small lookup table of less than 100 rows. The right-hand table can be broadcast efficiently to all nodes involved in the join. Thus, you would use the /* +broadcast */ hint to force a broadcast join strategy:

```
select straight_join customer.address, state_lookup.state_name
  from customer join /* +broadcast */ state_lookup
  on customer.state_id = state_lookup.state_id;
```

This query joins two large tables of unpredictable size. You might benchmark the query with both kinds of hints and find that it is more efficient to transmit portions of each table to other nodes for processing. Thus, you would use the /* +shuffle */ hint to force a partitioned join strategy:

```
select straight_join weather.wind_velocity, geospatial.altitude
  from weather join /* +shuffle */ geospatial
  on weather.lat = geospatial.lat and weather.long = geospatial.long;
```

For joins involving three or more tables, the hint applies to the tables on either side of that specific JOIN keyword. The STRAIGHT_JOIN keyword ensures that joins are processed in a predictable order from left to right. For example, this query joins t1 and t2 using a partitioned join, then joins that result set to t3 using a broadcast join:

```
select straight_join t1.name, t2.id, t3.price
  from t1 join /* +shuffle */ t2 join /* +broadcast */ t3
  on t1.id = t2.id and t2.id = t3.id;
```

Table Level Hint

If you attach this table level hint convert_limit_to_sample(n) to a table either in the main query block or within a view/subquery and if the query satisfies the simple limit optimization conditions, the limit is converted to a table sample. The parameter 'n' in parenthesis specifies the sample percentage and must be an integer between 1 and 100.

```
set optimize_simple_limit = true;


CREATE VIEW v1 as SELECT * FROM T [convert_limit_to_sample(5)]
WHERE [always_true] <predicate>;
  SELECT * FROM v1 LIMIT 10;
```

In this case, the limit 10 is applied on top of a 5 percent sample of T which is applied after partition pruning.

Related Information

Background information about join queries

Performance considerations for join queries

Query options

Query options

Specify query options in the SET statement to apply the settings to the subsequently issued queries.

Some query options are useful in day-to-day operations for improving usability, performance, or flexibility.

Other query options control special-purpose aspects of Impala operation and are intended primarily for advanced debugging or troubleshooting.

Options with Boolean parameters can be set to 1 or true to enable, or 0 or false to turn off.

You can set query options directly through the JDBC and ODBC interfaces by using the SET statement. Formerly, SET was only available as a command within the `impala-shell` interpreter.

You can set query options for an `impala-shell` session by specifying one or more command-line arguments of the form --query_option=*OPTION*=*VALUE*.

Impala supports the following query options:

- ABORT_ON_ERROR
- AGG_MEM_CORRELATION_FACTOR
- ALLOW_ERASURE_CODED_FILES
- APPX_COUNT_DISTINCT
- BATCH_SIZE
- BROADCAST_BYTES_LIMIT
- BUFFER_POOL_LIMIT

- COMPRESSION_CODEC
- COMPUTE_STATS_MIN_SAMPLE_SIZE
- CONVERT_LEGACY_HIVE_PARQUET_UTC_TIMESTAMPS
- DEBUG_ACTION
- DECIMAL_V2
- DEFAULT_FILE_FORMAT
- DEFAULT_HINTS_INSERT_STATEMENT
- DEFAULT_JOIN_DISTRIBUTION_MODE
- DEFAULT_NDV_SCALE
- DEFAULT_SPILLABLE_BUFFER_SIZE
- DEFAULT_TRANSACTIONAL_TYPE
- DELETE_STATS_IN_TRUNCATE
- DISABLE_CODEGEN
- DISABLE_CODEGEN_ROWS_THRESHOLD
- DISABLE_HBASE_NUM_ROWS_ESTIMATE
- DISABLE_ROW_RUNTIME_FILTERING
- DISABLE_STREAMING_PREAGGREGATIONS
- DISABLE_UNSAFE_SPILLS
- ENABLE_DISTINCT_SEMI_JOIN_OPTIMIZATION
- ENABLE_EXPR_REWRITES
- EXEC_SINGLE_NODE_ROWS_THRESHOLD
- EXEC_TIME_LIMIT_S
- EXPAND_COMPLEX_TYPES
- EXPLAIN_LEVEL
- FETCH_ROWS_TIMEOUT_MS
- HBASE_CACHE_BLOCKS
- HBASE_CACHING
- IDLE_SESSION_TIMEOUT
- JOIN_ROWS_PRODUCED_LIMIT
- JOIN_SELECTIVITY_CORRELATION_FACTOR
- KUDU_READ_MODE
- KUDU_REPLICA_SELECTION
- LARGE_AGG_MEM_THRESHOLD
- LIVE_PROGRESS
- LIVE_SUMMARY
- MAX_CNF_EXPRS
- MAX_ERRORS
- MAX_FRAGMENT_INSTANCES_PER_NODE
- MAX_MEM_ESTIMATE_FOR_ADMISSION
- MAX_NUM_RUNTIME_FILTERS
- MAX_RESULT_SPOOLING_MEM
- MAX_ROW_SIZE
- MAX_SCAN_RANGE_LENGTH
- MAX_SPILLED_RESULT_SPOOLING_MEM
- MEM_LIMIT
- MEM_LIMIT_COORDINATORS
- MIN_SPILLABLE_BUFFER_SIZE
- MT_DOP
- NUM_NODES
- NUM_ROWS_PRODUCED_LIMIT

- NUM_SCANNER_THREADS
- OPTIMIZE_SIMPLE_LIMIT
- OPTIMIZE_PARTITION_KEY_SCANS
- ORC_SCHEMA_RESOLUTION
- PARQUET_COMPRESSION_CODEC
- PARQUET_ANNOTATE_STRINGS_UTF8
- PARQUET_ARRAY_RESOLUTION
- PARQUET_DICTIONARY_FILTERING
- PARQUET_FALLBACK_SCHEMA_RESOLUTION
- PARQUET_FILE_SIZE
- PARQUET_OBJECT_STORE_SPLIT_SIZE
- PARQUET_PAGE_ROW_COUNT_LIMIT
- PARQUET_READ_PAGE_INDEX
- PARQUET_READ_STATISTICS
- PARQUET_WRITE_PAGE_INDEX
- PREFETCH_MODE
- QUERY_TIMEOUT_S
- REFRESH_UPDATED_HMS_PARTITIONS
- REPLICA_PREFERENCE
- REQUEST_POOL
- RESOURCE_TRACE_RATIO
- RETRY_FAILED_QUERIES
- RUNTIME_BLOOM_FILTER_SIZE
- RUNTIME_FILTER_MAX_SIZE
- RUNTIME_FILTER_MIN_SIZE
- RUNTIME_FILTER_MODE
- RUNTIME_FILTER_WAIT_TIME_MS
- S3_SKIP_INSERT_STAGING
- SCAN_BYTES_LIMIT
- SCHEDULE_RANDOM_REPLICA
- SCRATCH_LIMIT
- SHUFFLE_DISTINCT_EXPRS
- SPOOL_QUERY_RESULTS
- SPOOL_ALL_RESULTS_FOR_RETRIES
- SUPPORT_START_OVER
- SYNC_DDL
- THREAD_RESERVATION_AGGREGATE_LIMIT
- THREAD_RESERVATION_LIMIT
- TIMEZONE
- TOPN_BYTES_LIMIT
- USE_DOP_FOR_COSTING
- #unique_34/unique_34_Connect_42_UTF8_MODE

ABORT_ON_ERROR query option

When this option is enabled, Impala cancels a query immediately when any of the nodes encounters an error, rather than continuing and possibly returning incomplete results. This option is disabled by default, to help gather maximum diagnostic information when an error occurs, for example, whether the same problem occurred on all nodes or only a single node. Currently, the errors that Impala can skip over involve data corruption, such as a column that contains a string value when expected to contain an integer value.

To control how much logging Impala does for non-fatal errors when ABORT_ON_ERROR is turned off, use the MAX_ERRORS option.

Type: BOOLEAN

Default: FALSE (0)

AGG_MEM_CORRELATION_FACTOR and LARGE_AGG_MEM_THRESHOLD query options

These query options are an improvement in memory estimation for aggregation node and do not replace the existing query option PREAGG_BYTES_LIMIT.

Prior to this release, you used the query option PREAGG_BYTES_LIMIT to control memory usage of preagg node. But it has to be manually tweaked to be more conservative with memory and to directly limit the memory consumption of the preaggregations. When PREAGG_BYTES_LIMIT is not set, the memory estimation of each streaming preaggregation node in query plan can be expanded to consume nearly all of the memory reservation on a host, starving the merge aggregation or possibly other streaming aggregation. This is because the Planner often overestimates aggregation node memory estimates since it uses simple multiplication of NDVs of contributing grouping columns.

To reduce this large memory estimation issue, this patch includes two new query options AGG_MEM_CORRELATION_FACTOR and LARGE_AGG_MEM_THRESHOLD to the existing query option to control memory estimation. If the estimated memory from the default NDV multiplication method exceed LARGE_AGG_MEM_THRESHOLD, recompute the estimated memory again by comparing against the max(NDV) & AGG_MEM_CORRELATION_FACTOR method.

Default values:

LARGE_AGG_MEM_THRESHOLD = 512 MB

AGG_MEM_CORRELATION_FACTOR = 0.5

Note: You can revert to the old behavior by setting the query options to the following values LARGE_AGG_MEM_THRESHOLD=0 or -1, or AGG_MEM_CORRELATION_FACTOR=0.0

ALLOW_ERASURE_CODED_FILES query option

Use the ALLOW_ERASURE_CODED_FILES query option to enable or disable the support of erasure coded files in Impala. Until Impala is fully tested and certified with erasure coded files, this query option is set to FALSE by default.

When the ALLOW_ERASURE_CODED_FILES query option is set to FALSE, Impala returns an error when a query requires scanning an erasure coded file.

Type: BOOLEAN

Default: FALSE (0)

Added in: Impala 3.1

APPX_COUNT_DISTINCT query option

When the APPX_COUNT_DISTINCT query option is set to TRUE, Impala implicitly converts COUNT(DISTINCT) operations to the NDV() function calls. The resulting count is approximate rather than precise. Enable the query option when a tolerable amount of error is acceptable in order to obtain faster query results than with a COUNT (DISTINCT) queries.

Type: BOOLEAN

Default: FALSE (0)

BATCH_SIZE query option

Number of rows evaluated at a time by SQL operators. Unspecified or a size of 0 uses a predefined default size. Using a large number improves responsiveness, especially for scan operations, at the cost of a higher memory footprint.

This option is primarily for testing during Impala development, or for use under the direction of Cloudera support.

Type: Number

Default: 0 (meaning the predefined default of 1024)

BROADCAST_BYTES_LIMIT query option

Sets the limit for the size of the broadcast input based on estimated size. The Impala planner may in rare cases make a bad choice to broadcast a large table or intermediate result and encounter performance problems due to high memory pressure. Setting this limit will make the planner pick a partition based hash join instead of broadcast and avoid such performance problems.

Type: INT

Default: The default value is 34359738368 (32 GB). A value of 0 causes the option to be ignored.

Usage notes:

```
Set this option in bytes. For example:
-- Change the limit to 16GB.
set broadcast_bytes_limit=17179869184;

-- Disable the BROADCAST_BYTES_LIMIT.
set broadcast_bytes_limit=0;
```

BUFFER_POOL_LIMIT query option

Defines a limit on the amount of memory that a query can allocate from the internal buffer pool. The value for this limit applies to the memory on each host, not the aggregate memory across the cluster.

Typically not changed by users, except during diagnosis of out-of-memory errors during queries.

You can set it to an absolute value, e.g. 8GB, or a relative value, e.g. 80%, based on the MEM_LIMIT setting.

Type: INT

Default: The default setting for this option is the lower of 80% of the MEM_LIMIT setting, or the MEM_LIMIT setting minus 100 MB.

Usage notes:

If queries encounter out-of-memory errors, consider decreasing the BUFFER_POOL_LIMIT setting to less than 80% of the MEM_LIMIT setting.

COMPRESSION_CODEC query option

When Impala writes Parquet data files and other supported data files using the INSERT statement, the underlying compression is controlled by the COMPRESSION_CODEC query option.

For the ZSTD compression, the optional compression level can be specified as shown in the syntax below.

Note: Prior to Impala 2.0, this option was named PARQUET_COMPRESSION_CODEC. In Impala 2.0 and later, the PARQUET_COMPRESSION_CODEC name is not recognized. Use the more general name COMPRESSION_CODEC for new code.

Syntax:

```
SET COMPRESSION_CODEC=CODEC_NAME; // Supported for all codecs.
SET COMPRESSION_CODEC=CODEC_NAME:COMPRESSION_LEVEL; // Only su
pported for ZSTD.
```

The allowed values for this query option are SNAPPY (the default), GZIP, ZSTD, LZ4, and NONE.

ZSTD also supports setting a compression level. The lower the level, the faster the speed at the cost of compression ratio. Compression levels from 1 up to 22 are supported for ZSTD. The default compression level 3 is used, if one is not passed using the compression_codec query option.

Note: A Parquet file created with COMPRESSION_CODEC=NONE is still typically smaller than the original data, due to encoding schemes such as run-length encoding and dictionary encoding that are applied separately from compression.

The option value is not case-sensitive.

If the option is set to an unrecognized value, all kinds of queries will fail due to the invalid option setting, not just queries involving Parquet tables. (The value BZIP2 is also recognized, but is not compatible with Parquet tables.)

Type: STRING

Default: SNAPPY

COMPUTE_STATS_MIN_SAMPLE_SIZE query option

The COMPUTE_STATS_MIN_SAMPLE_SIZE query option specifies the minimum number of bytes that will be scanned in COMPUTE STATS TABLESAMPLE, regardless of the user-supplied sampling percent. This query option prevents sampling for very small tables where accurate stats can be obtained cheaply without sampling because the minimum sample size is required to get meaningful stats.

Type: INT

Default: 1GB

CONVERT_LEGACY_HIVE_PARQUET_UTC_TIMESTAMPS query option

When Hive writes to Parquet data files, the TIMESTAMP values are normalized to UTC from the local time zone of the host where the data was written. On the other hand, Impala does not make any time zone adjustment when it writes or reads INT96 TIMESTAMP values to Parquet files. This difference in time zone handling can cause potentially inconsistent results when Impala processes TIMESTAMP values in the Parquet files written by Hive.

To avoid incompatibility problems or having to code workarounds, you can specify this CONVERT_LEGACY_HIVE_PARQUET_UTC_TIMESTAMPS query option.

Type: BOOLEAN

Note: This query option can also be set as a flag on the Coordinator or by using the --default_query_options configuration. When set as a query option, it will take precedence over the old flag. The preferred method is to set this as a query option because it offers more flexibility.

DEBUG_ACTION query option

Introduces artificial problem conditions within queries. For internal Cloudera debugging and troubleshooting.

Type: STRING

Default: An empty string

DECIMAL_V2 query option

A query option that changes behavior related to the DECIMAL data type. Set this option to FALSE for backward compatibility to Impala 2.x.

Type: Boolean

Default: TRUE

DEFAULT_FILE_FORMAT query option

Use the DEFAULT_FILE_FORMAT query option to set the default table file format. The following values are supported:

- TEXT (0)
- RC_FILE (1)
- SEQUENCE_FILE (2)
- AVRO (3)
- PARQUET (4)
- KUDU (5)
- ORC (6)

In impala-shell or Hue, the SET DEFAULT_FILE_FORMAT statement will not return an error when the option was set to an unsupported value. Impala validates the value and returns an error when you submitted the next query.

Type: Enum as specified above

Default: TEXT (0)

Added in: Impala 3.3

DEFAULT_HINTS_INSERT_STATEMENT query option

The DEFAULT_HINTS_INSERT_STATEMENT query option sets the default hints for the INSERT statements with no optimizer hint specified.

When there are hints specified in an INSERT statement, these default hints are ignored.

You can specify multiple hints separated by ':'.

For example:

```
SET DEFAULT_HINTS_INSERT_STATEMENT=CLUSTERED;
SET DEFAULT_HINTS_INSERT_STATEMENT=SHUFFLE;
SET DEFAULT_HINTS_INSERT_STATEMENT=NOCLUSTERED:NOSHUFFLE;
```

The default hints apply to the HDFS and Kudu table formats and are ignored for the HBase table format.

Type: STRING

Default: No default hint

Added in: Impala 3.3

DEFAULT_JOIN_DISTRIBUTION_MODE query option

Impala optimizes join queries based on the presence of table statistics, which are produced by the Impala COMPUTE STATS statement. By default, when a table involved in the join query does not have statistics, Impala uses the “broadcast” technique that transmits the entire contents of the table to all executor nodes participating in the query. If one table involved in a join has statistics and the other does not, the table without statistics is broadcast. If both tables are missing statistics, the table on the right-hand side of the join is broadcast. This behavior is appropriate when the table involved is relatively small, but can lead to excessive network, memory, and CPU overhead if the table being broadcast is large.

Because Impala queries frequently involve very large tables, and suboptimal joins for such tables could result in spilling or out-of-memory errors, the setting DEFAULT_JOIN_DISTRIBUTIO N_MODE=SHUFFLE lets you override the default behavior. The shuffle join mechanism divides the corresponding rows of each table involved in a join query using a hashing algorithm, and transmits subsets of the rows to other nodes for processing. Typically, this kind of join is more efficient for joins between large tables of similar size.

The setting DEFAULT_JOIN_DISTRIBUTION_MODE=SHUFFLE is recommended when setting up and deploying new clusters, because it is less likely to result in serious consequences such as spilling or out-of-memory errors if the query plan is based on incomplete information. This setting is not the default, to avoid changing the performance characteristics of join queries for clusters that are already tuned for their existing workloads.

The allowed values are BROADCAST (0) or SHUFFLE (1).

Type: INT

Examples:

The following examples demonstrate appropriate scenarios for each setting of this query option.

```
-- Create a billion-row table.
create table big_table stored as parquet
  as select * from huge_table limit 1e9;

-- For a big table with no statistics, the
-- shuffle join mechanism is appropriate.
set default_join_distribution_mode=shuffle;
...join queries involving the big table...
```

```
-- Create a hundred-row table.
create table tiny_table stored as parquet
  as select * from huge_table limit 100;

-- For a tiny table with no statistics, the
-- broadcast join mechanism is appropriate.
set default_join_distribution_mode=broadcast;

...join queries involving the tiny table...
```

```
compute stats tiny_table;
compute stats big_table;
-- Once the stats are computed, the query option has
-- no effect on join queries involving these tables.
-- Impala can determine the absolute and relative sizes
-- of each side of the join query by examining the
-- row size, cardinality, and so on of each table.

...join queries involving both of these tables...
```

DEFAULT_NDV_SCALE query option

Improving the precision of NDV()

As a cluster admin, you can improve the precision of NDV() using this new query option DEFA ULT_NDV_SCALE. You may adjust the value in this query option to change the default precision setting for NDV() so that the SQL writers do not have to rewrite the SQLs to adjust NDV’s precision. This new option allows the cluster admin to adjust precision for each Admission Control queue according to cluster's resource usage.

Controlling the NDV scale

Until this release, the scale used in NDV(<expr>) functions was fixed at 2. Now the scale can be provided using the newly added query option. This option will not influence the NDV scale for SQL function NDV(<expr>, <scale>) in which the NDV scale is provided by the 2nd argument <scale>

Type: Number

Allowed value: Range [1 .. 10]

Default: 2

Example:

The following is an example of a typical workload with different NDV scales and with a cardinality of 40,090,650.

```
SET DEFAULT_NDV_SCALE = 10;
SET DEFAULT_NDV_SCALE = 5;
SET DEFAULT_NDV_SCALE = 2;
```

Metric	Count Distinct	NDV2	NDV5	NDV10
Memory(GB)	3.83	1.84	1.85	1.89
Duration(s)	182.89	30.22	29.72	29.24
ErrorRate	0%	1.8%	1.17%	0.06%

DEFAULT_SPILLABLE_BUFFER_SIZE query option

Specifies the default size for a memory buffer used when the spill-to-disk mechanism is activated, for example for queries against a large table with no statistics, or large join operations.

Type: INT

Default: 2097152 (2 MB)

Unit: A numeric argument represents a size in bytes; you can also use a suffix of m or mb for megabytes, or g or gb for gigabytes. If you specify a value with unrecognized formats, subsequent queries fail with an error.

Usage notes:

This query option sets an upper bound on the size of the internal buffer size that can be used during spill-to-disk operations. The actual size of the buffer is chosen by the query planner.

If overall query performance is limited by the time needed for spilling, consider increasing the DEFAULT_SPILLABLE_BUFFER_SIZE setting. Larger buffer sizes result in Impala issuing larger I/O requests to storage devices, which might result in higher throughput, particularly on rotational disks.

The tradeoff with a large value for this setting is increased memory usage during spill-to-disk operations. Reducing this value may reduce memory consumption.

To determine if the value for this setting is having an effect by capping the spillable buffer size, you can see the buffer size chosen by the query planner for a particular query. EXPLAIN the query while the setting EXPLAIN_LEVEL=2 is in effect.

DEFAULT_TRANSACTIONAL_TYPE query option

Use the DEFAULT_TRANSACTIONAL_TYPE query option to create insert-only transactional tables by default.

Note: The DEFAULT_TRANSACTIONAL_TYPE query option applies only when integrated with Hive 3.

The supported values are:

- NONE: The table will not be created as transactional.
- INSERT_ONLY: The table will be created as transactional.

If either table properties, transactional or transactional_properties, are set, this query option is ignored.

Note: The DEFAULT_TRANSACTIONAL_TYPE query option does not affect external, Kudu, or HBase tables as these cannot be transactional.

Type: Enum as specified above

Default: NONE

Added in: Impala 3.3

DELETE_STATS_IN_TRUNCATE query option

This query option DELETE_STATS_IN_TRUNCATE can be used to delete or retain table statistics. The default value of this option is 1 or true which means table statistics will be deleted as part of truncate operation. However the default may not be an ideal solution in case of non-transactional tables when the table and column statistics for the table are also deleted by default. This can be an expensive operation especially when many truncate table commands are running concurrently. In cases where truncate operation is used to remove the existing data and then reload new data, it is likely that you will compute stats again as soon as the new data is reloaded. This would overwrite the existing statistics and hence the additional time spent by the truncate operation to delete column and table statistics becomes obsolete. To avoid this expensive operation and to improve performance, you can set this query option to false or 0. When set to false, a truncate operation will not delete the table and column statistics for the table.

Type:BOOLEAN; recognized values are 1 and 0, or true and false;

Default:TRUE (shown as 1 in output of SET statement).

DISABLE_CODEGEN query option

The DISABLE_CODEGEN is a debug option, and it's used to work around any issues with Impala's runtime code generation. If a query fails with an “illegal instruction” or other hardware-specific message, try setting DISABLE_CODEGEN=true and running the query again. If the query succeeds only when the DISABLE_CODEGEN option is turned on, submit the problem to Cloudera support and include that detail in the problem report.

Most queries will run significantly slower with DISABLE_CODEGEN=true.

In Impala 2.10 and higher, the DISABLE_CODEGEN_ROWS_THRESHOLD optimisation automatically disables codegen for small queries because short-running queries may run faster without the overhead of codegen.

The following values are supported:

- TRUE or 1: Disables codegen.
- FALSE or 0: Enables codegen.

Type: BOOLEAN

Default: FALSE (0)

DISABLE_CODEGEN_ROWS_THRESHOLD query option

Specify the number of rows processed per Impala daemon which is the cutoff point below which Impala disables native code generation for the whole query. Native code generation is beneficial for queries that process many rows because it reduces the time taken to process of each row. However, generating the native code adds latency to query startup. Therefore, automatically disabling codegen for queries that process relatively small amounts of data can improve query response time.

This setting applies to queries where the number of rows processed can be accurately determined, either through table and column statistics, or by the presence of a LIMIT clause. If Impala cannot accurately estimate the number of rows, then this setting does not apply.

If a query uses the complex data types STRUCT, ARRAY, or MAP, then codegen is never automatically disabled regardless of the DISABLE_CODEGEN_ROWS_THRESHOLD setting.

Type: Number

Default: 50000

Usage notes: Typically, you increase the default value to make this optimization apply to more queries. If incorrect or corrupted table and column statistics cause Impala to apply this optimization incorrectly to queries that actually involve substantial work, you might see the queries being slower as a result of codegen being disabled. In that case, recompute statistics with the COMPUTE STATS or COMPUTE INCREMENTAL STATS statement. If there is a problem collecting accurate statistics, you can turn this feature off by setting the value to 0.

DISABLE_HBASE_NUM_ROWS_ESTIMATE

Use the DISABLE_HBASE_NUM_ROWS_ESTIMATE query option to disable key sampling of HBase tables in row count and row size estimation.

While generating a plan for an HBase query, the planner samples the underlying HBase tables to estimate their row count and row size, and the sampling can negatively impact the planning time. When the HBase table stats do not change much in a short time, disable the sampling by setting the DISABLE_HBASE_NUM_ROWS_ESTIMATE query option to TRUE. And Impala planner will fall back to using Hive Metastore (HMS) table stats instead.

When DISABLE_HBASE_NUM_ROWS_ESTIMATE query option is set to TRUE, you need to update the HMS table stats by running COMPUTE STATS. Alternatively, you can manually set table statistics by running ALTER TABLE.

The following values are supported:

- TRUE or 1: Disables the normal key sampling of HBase tables and uses HMS table stats for estimation.
- FALSE or 0: Enables the normal sampling of HBase tables.

Type: BOOLEAN

Default: FALSE

DISABLE_ROW_RUNTIME_FILTERING query

The DISABLE_ROW_RUNTIME_FILTERING query option reduces the scope of the runtime filtering feature. Queries still dynamically prune partitions, but do not apply the filtering logic to individual rows within partitions.

Only applies to queries against HDFS-based tables using the Parquet file format. For other file formats, Impala only prunes at the level of partitions, not individual rows.

This query option only applies to queries against

Type: BOOLEAN

Default: FALSE (0)

Usage notes:

Impala automatically evaluates whether the per-row filters are being effective at reducing the amount of intermediate data. Therefore, this option is typically only needed for the rare case where Impala cannot accurately determine how effective the per-row filtering is for a query.

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables.

Kudu consideration:

When applied to a query involving a Kudu table, this option turns off all runtime filtering for the Kudu table.

DISABLE_STREAMING_PREAGGREGATIONS query option

Turns off the “streaming preaggregation” optimization that is available in Impala 2.5 and higher. This optimization reduces unnecessary work performed by queries that perform aggregation operations on columns with few or no duplicate values, for example DISTINCT *ID_COLUMN* or GROUP BY *UNIQUE_COLUMN*. If the optimization causes regressions in existing queries that use aggregation functions, you can turn it off as needed by setting this query option.

Type: BOOLEAN

Default: FALSE (0)

Usage notes:

Typically, queries that would require enabling this option involve very large numbers of aggregated values, such as a billion or more distinct keys being processed on each worker node.

DISABLE_UNSAFE_SPILLS query option

Enable this option if you prefer to have queries fail when they exceed the Impala memory limit, rather than write temporary data to disk.

Queries that “spill” to disk typically complete successfully, when in earlier Impala releases they would have failed. However, queries with exorbitant memory requirements due to missing statistics or inefficient join clauses could become so slow as a result that you would rather have them cancelled automatically and reduce the memory usage through standard Impala tuning techniques.

This option prevents only “unsafe” spill operations, meaning that one or more tables are missing statistics or the query does not include a hint to set the most efficient mechanism for a join or INSE RT ... SELECT into a partitioned table. These are the tables most likely to result in suboptimal execution plans that could cause unnecessary spilling. Therefore, leaving this option enabled is a good way to find tables on which to run the COMPUTE STATS statement.

Type: BOOLEAN

Default: FALSE (0)

ENABLE_DISTINCT_SEMI_JOIN_OPTIMIZATION query option

When this query option is enabled the query planner considers inserting a distinct aggregation on the inner side of the join based on whether that aggregation would reduce the number of rows by more than 75%. This optimization reduces the number of potential plan regressions.

ENABLE_EXPR_REWRITES query option

The ENABLE_EXPR_REWRITES query option controls whether to enable or disable the query compile time optimizations. These optimizations rewrite the expression trees to a more compact and optimized form that helps avoid redundant expression evaluation at run time. Performance optimizations controlled by this query option include:

- Constant folding
- Extracting common conjuncts from disjunctions
- Simplify conditionals with constant conditions

Set the option to FALSE or 0 to disable the performance optimizations.

Type: BOOLEAN

Default: TRUE (1)

EXEC_SINGLE_NODE_ROWS_THRESHOLD query option

Specify the number of rows scanned as the cutoff point below which Impala treats a query as a "small" query, turning off optimizations such as parallel execution and native code generation. The overhead for these optimizations is applicable for queries involving substantial amounts of data, but it makes sense to skip them for queries involving tiny amounts of data. Reducing the overhead for small queries allows Impala to complete them more quickly, keeping admission control slots, CPU, memory, and so on available for resource-intensive queries.

This setting applies to queries where the number of rows processed can be accurately determined, either through table and column statistics, or by the presence of a LIMIT clause. If Impala cannot accurately estimate the number of rows, then this setting does not apply.

In Impala 2.3 and higher, where Impala supports the complex data types STRUCT, ARRAY, and MAP, if a query refers to any column of those types, the small-query optimization is turned off for that query regardless of the EXEC_SINGLE_NODE_ROWS_THRESHOLD setting.

For a query that is determined to be "small", all work is performed on the coordinator node. This might result in some I/O being performed by remote reads. The savings from not distributing the query work and not generating native code are expected to outweigh any overhead from the remote reads.

Type: Number

Default: 100

Usage notes: Typically, you increase the default value to make this optimization apply to more queries. If incorrect or corrupted table and column statistics cause Impala to apply this optimization incorrectly to queries that actually involve substantial work, you might see the queries being slower as a result of remote reads. In that case, recompute statistics with the COMPUTE STATS or COMPUTE INCREMENTAL STATS statement. If there is a problem collecting accurate statistics, you can turn this feature off by setting the value to -1.

EXEC_TIME_LIMIT_S query option

The EXEC_TIME_LIMIT_S query option sets a time limit on query execution in seconds. If a query is still executing when time limit expires, it is automatically canceled. The option is intended to prevent runaway queries that run for much longer than intended.

For example, an Impala administrator could set a default value of EXEC_TIME_LIMIT_S=3600 for a resource pool to automatically stop queries that run for longer than one hour. Then, if a user accidentally runs a large query that executes for more than one hour, it will be automatically killed after the time limit expires to free up resources. Users can override the default value per query or per session if they do not want the default EXEC_TIME_LIMIT_S value to apply to a specific query or a session.

Note:

The time limit only starts once the query is executing. Time spent planning the query, scheduling the query, or in admission control is not counted towards the execution time limit. SELECT statements are eligible for automatic cancellation until the client has fetched all result rows. DML queries are eligible for automatic cancellation until the DML statement has finished.

Type: Number

Default: 0 (no time limit)

EXPAND_COMPLEX_TYPES Query Option

Older versions of Impala did not support complex types in the select list, and therefore a SELECT * statement only expanded to primitive types even when the table contained complex-typed columns. Support for complex types in the select list has since been added, but for backward

compatibility the behaviour that “*” expressions skip complex types has been kept as the default. If the EXPAND_COMPLEX_TYPES query option is set to true, “*” expressions will include complex types as well.

Note: Even if EXPAND_COMPLEX_TYPES is false, complex-typed columns that are listed explicitly in the select list (i.e. not as a “*” expression) will be included in the result.

Type:BOOLEAN

Default:FALSE

Added in:Impala 4.2

EXPLAIN_LEVEL query option

Controls the amount of detail provided in the output of the EXPLAIN statement. The basic output can help you identify high-level performance issues such as scanning a higher volume of data or more partitions than you expect. The higher levels of detail show how intermediate results flow between nodes and how different SQL operations such as ORDER BY, GROUP BY, joins, and WHERE clauses are implemented within a distributed query.

Type: STRING or INT

Default: 1

Arguments:

The allowed range of numeric values for this option is 0 to 3:

- 0 or MINIMAL: A barebones list, one line per operation. Primarily useful for checking the join order in very long queries where the regular EXPLAIN output is too long to read easily.
- 1 or STANDARD: The default level of detail, showing the logical way that work is split up for the distributed query.
- 2 or EXTENDED: Includes additional detail about how the query planner uses statistics in its decision-making process, to understand how a query could be tuned by gathering statistics, using query hints, adding or removing predicates, and so on. In Impala 3.2 and higher, the output also includes the analyzed query with the cast information in the output header, and the implicit cast info in the Predicate section.
- 3 or VERBOSE: The maximum level of detail, showing how work is split up within each node into “query fragments” that are connected in a pipeline. This extra detail is primarily useful for low-level performance testing and tuning within Impala itself, rather than for rewriting the SQL code at the user level.

Changing the value of this option controls the amount of detail in the output of the EXPLAIN statement. The extended information from level 2 or 3 is especially useful during performance tuning, when you need to confirm whether the work for the query is distributed the way you expect, particularly for the most resource-intensive operations such as join queries against large tables, queries against tables with large numbers of partitions, and insert operations for Parquet tables. The extended information also helps to check estimated resource usage when you use the admission control or resource management features.

Usage notes:

Read the EXPLAIN output from bottom to top. The lowest lines represent the initial work of the query (scanning data files), the lines in the middle represent calculations done on each node and how intermediate results are transmitted from one node to another, and the topmost lines represent the final results being sent back to the coordinator node.

The numbers in the left column are generated internally during the initial planning phase and do not represent the actual order of operations, so it is not significant if they appear out of order in the EXPLAIN output.

At all EXPLAIN levels, the plan contains a warning if any tables in the query are missing statistics. Use the COMPUTE STATS statement to gather statistics for each table and suppress this warning. See *Tables and column statistics* for details about how the statistics help query performance.

The PROFILE command in `impala-shell` always starts with an explain plan showing full detail, the same as with EXPLAIN_LEVEL=3. After the explain plan comes the executive summary, the same output as produced by the SUMMARY command in `impala-shell`.

FETCH_ROWS_TIMEOUT_MS query option

Use the FETCH_ROWS_TIMEOUT_MS query option to control how long Impala waits for query results when clients fetch rows.

When this query option is set to 0, fetch requests wait indefinitely.

The timeout applies both when query result spooling is enabled and disabled:

- When result spooling is disabled (SPOOL_QUERY_RESULTS = FALSE), the timeout controls how long a client waits for a single row batch to be produced by the coordinator.
- When result spooling is enabled ((SPOOL_QUERY_RESULTS = TRUE), a client can fetch multiple row batches at a time, so this timeout controls the total time a client waits for row batches to be produced.

The timeout also applies to fetch requests issued against queries in the 'RUNNING' state. A 'RUNNING' query has no rows available, so any fetch request will wait until the query transitions to the 'FINISHED' state and for it to fetch all requested rows. A query in the 'FINISHED' state means that the rows are available to be fetched.

Type: INT

Default: 10000 (10 seconds)

Added in: Impala 3.4

HBASE_CACHE_BLOCKS query option

Setting this option is equivalent to calling the setCacheBlocks method of the class org.apache.hadoop.hbase.client.Scan in an HBase Java application. Helps to control the memory pressure on the HBase RegionServer, in conjunction with the HBASE_CACHING query option.

Type: BOOLEAN

Default: FALSE (0)

HBASE_CACHING query option

Setting this option is equivalent to calling the setCaching method of the class org.apache.hadoop.hbase.client.Scan in an HBase Java application. Helps to control the memory pressure on the HBase RegionServer, in conjunction with the HBASE_CACHE_BLOCKS query option.

Type: BOOLEAN

Default: FALSE (0)

IDLE_SESSION_TIMEOUT query option

Specify the time in seconds after which an idle session is cancelled. A session is idle when no activity is occurring for any of the queries in that session, and the session has not started any new queries. Once a session is expired, you cannot issue any new query requests to it. The session remains open, but the only operation you can perform is to close it.

The IDLE_SESSION_TIMEOUT query option overrides the ##idle_session_timeout startup option. See *Setting timeout periods for daemons, queries, and sessions* for the ##idle_session_timeout startup option.

The IDLE_SESSION_TIMEOUT query option allows JDBC/ODBC connections to set the session timeout as a query option with the SET statement.

Type: Number

Default: 0

When this option is set to default:

- If ##idle_session_timeout is not set, the session never expires.
- If ##idle_session_timeout is set, use that timeout value.

JOIN_ROWS_PRODUCED_LIMIT query option

The JOIN_ROWS_PRODUCED_LIMIT is an advanced query option that limits the number of join rows produced by a join node and is used to prevent runaway join queries. A query is canceled when any one of the joins in the query produces more rows (totaled across all instances of that join) than the specified limit set by the JOIN_ROWS_PRODUCED_LIMIT option.

This limit is checked periodically, so it is possible that the actual number of rows produced may be slightly higher than the limit.

The default value of 0 specifies that there is no limit on the number of join rows produced.

Type: BIGINT

Allowed values: 0 or positive numbers

Default: 0

JOIN_SELECTIVITY_CORRELATION_FACTOR

Any conjuncts in an inner or outer join, similar to the following example, will be used only when no other equijoin conjunct is present.

```
SELECT * FROM t1 inner join
  (SELECT a2, MAX(b2) as max_b2 FROM t2 GROUP BY a2) s2
  ON t1.a1 = s2.a2 AND t1.b1 = s2.max_b2
```

In this example, the stats for the second conjunct involving the MAX expression gets added to the 'otherEqJoinStats' list. These stats were being used only when no other equijoin conjunct (involving base columns) was present leading to over-estimation.

To mitigate this issue, an improvement is made to the cardinality estimate for such cases by considering the conjuncts in the 'otherEqJoinStats list' in combination with the equijoin conjuncts.

A new query option join_selectivity_correlation_factor which is a floating point number between 0 and 1 inclusive is introduced in this release. It defaults to 0.0 which preserves the existing behavior of using the Minimum selectivity of the conjuncts. Given multiple join conjuncts C1, C2 ....Cn having comparable selectivity, you may notice over-estimation by several orders of magnitude.

Setting the above query option to a value higher than 0 first computes the product of the selectivities: sel(C1) * sel(C2) .. sel(Cn) and then scales it by dividing by the join_selectivity_correlation_factor.

Note that this setting applies to all the joins in the query. Another approach is to use per-join hints, but there are limitations in the hints approach too.

Default: "0.0"

KUDU_READ_MODE query option

The KUDU_READ_MODE query option allows you to set a desired consistency level for scans of Kudu tables.

The following values are supported for the query option:

- "DEFAULT": The value of the startup flag, --kudu_read_mode, is used.
- "READ_LATEST": Commonly known as the Read Committed isolation mode, in this mode, Kudu provides no consistency guarantees for this mode, except that all returned rows were committed at some point.
- "READ_AT_SNAPSHOT": Kudu will take a snapshot of the current state of the data and perform the scan over the snapshot, possibly after briefly waiting for ongoing writes to complete. This provides "Read Your Writes" consistency within a single Impala session, except in the case of a Kudu leader change. See the Kudu documentation for more details.

Type: String

Default: "DEFAULT"

Added in: Impala 3.1

KUDU_REPLICA_SELECTION query option

With the location-aware placement policy for tablet replicas in Kudu, the replicas are spread across locations so that the failure of tablet servers in one location does not make tablets unavailable. With this advanced placement policy the replicas of a tablet on Kudu tablet servers are assigned with the role as a leader or a follower. By default when Impala attempts to read from the tablet server, it is targeted to the closest tablet server that is to choose "nearby" replicas on the tablet server for scanning. However using this new query option KUDU_REPLICA_SELECTION, the queries can be targeted to the leader-only replica and skip non-leader replicas while scanning. When the new query option KUDU_REPLICA_SELECTION is set as LEADER_ONLY, Impala planner will generate a query plan that enables Impala to scan Kudu tables at the leader-only replicas.

LIVE_PROGRESS query option

When the LIVE_PROGRESS query option is set to TRUE, Impala displays an interactive progress bar showing roughly what percentage of processing has been completed for queries submitted through the `impala-shell` command. When the query finishes, the progress bar is erased from the `impala-shell` console output.

You can enable this query option within `impala-shell` by starting the shell with the --live_progress command-line option. You can still turn this setting off and on again within the shell through the SET command.

Starting in Impala 3.1, the summary output also includes the queuing status consisting of whether the query was queued and what was the latest queuing reason.

Type: BOOLEAN

Default: FALSE (0)

Usage notes:

The output from this query option is printed to standard error. The output is only displayed in interactive mode, that is, not when the -q or -f options are used.

Restrictions:

Because the percentage complete figure is calculated using the number of issued and completed “scan ranges”, which occur while reading the table data, the progress bar might reach 100% before the query is entirely finished. For example, the query might do work to perform aggregations after all the table data has been read. If many of your queries fall into this category, consider using the LIVE_SUMMARY option instead for more granular progress reporting.

The LIVE_PROGRESS and LIVE_SUMMARY query options currently do not produce any output during COMPUTE STATS operations.

Because the LIVE_PROGRESS query option is available only within the `impala-shell` interpreter, you cannot change the query option through the SQL SET statement using the JDBC

or ODBC interfaces. The SET command in `impala-shell` recognizes the name as a shell-only option.

LIVE_SUMMARY query option

When the LIVE_SUMMARY query option is set to TRUE, Impala displays the same output as the SUMMARY command for queries submitted through the `impala-shell` command, with the measurements updated in real time as the query progresses. When the query finishes, the final SUMMARY output remains visible in the `impala-shell` console output.

You can enable this query option within `impala-shell` by starting the shell with the --live_summary command-line option. You can still turn this setting off and on again within the shell through the SET command.

Starting in Impala 3.1, the summary output also includes the queuing status consisting of whether the query was queued and what was the latest queuing reason.

Type: Boolean

Default: FALSE (0)

Usage notes:

The output from this query option is printed to standard error. The output is only displayed in interactive mode, that is, not when the -q or -f options are used.

Only a single report is displayed at any one time, with each update overwriting the previous numbers.

The live summary output can be useful for evaluating long-running queries, to evaluate which phase of execution takes up the most time, or if some hosts take much longer than others for certain operations, dragging overall performance down. By making the information available in real time, this feature lets you decide what action to take even before you cancel a query that is taking much longer than normal.

For example, you might see the HDFS scan phase taking a long time, and therefore revisit performance-related aspects of your schema design such as constructing a partitioned table, switching to the Parquet file format, running the COMPUTE STATS statement for the table, and so on. Or you might see a wide variation between the average and maximum times for all hosts to perform some phase of the query, and therefore investigate if one particular host needed more memory or was experiencing a network problem.

Restrictions:

The LIVE_PROGRESS and LIVE_SUMMARY query options currently do not produce any output during COMPUTE STATS operations.

Because the LIVE_SUMMARY query option is available only within the `impala-shell` interpreter, you cannot change the query option through the SQL SET statement using the JDBC or ODBC interfaces. The SET command in `impala-shell` recognizes the name as a shell-only option.

MAX_CNF_EXPRS query option

To optimize the performance of queries, the predicates are reorganized so that they can be applied earlier in the data processing. For example, some predicates can be pushed down to the scan node. This helps improve performance for some complex queries. However the CNF rewrite can lead to significant frontend memory usage and eventually OutOfMemory for a complex query that contains many predicates. To avoid this memory problem while maintaining performance improvement in basic circumstances and avoiding performance regressions on complex queries, this release lowered the default value of MAX_CNF_EXPRS from unlimited to 200.

MAX_ERRORS query option

Maximum number of non-fatal errors for any particular query that are recorded in the Impala log file. For example, if a billion-row table had a non-fatal data error in every row, you could diagnose the problem without all billion errors being logged. Unspecified or 0 indicates the built-in default value of 100.

This option only controls how many errors are reported. To specify whether Impala continues or halts when it encounters such errors, use the ABORT_ON_ERROR option.

Type: Number

Default: 0 (meaning 100 errors)

MAX_FRAGMENT_INSTANCES_PER_NODE query option

When you executed a query with COMPUTE_PROCESSING_COST=1, Impala relied on the MT_DOP option to decide the degree of parallelism of the scan fragment. Now scan node's processing cost is used as another factor to consider raising scan parallelism beyond MT_DOP.

Scan node cost now includes the number of effective scan ranges. Each scan range is given a weight of (0.5% * min_processing_per_thread), which roughly means that one scan node instance can handle at most 200 scan ranges. The new query option MAX_FRAGMENT_INSTANCES_PER_NODE is introduced to cap the maximum number of fragment instances per node. This newly introduced query option works in conjunction with PROCESSING_COST_MIN_THREADS.

Query option MAX_FRAGMENT_INSTANCES_PER_NODE is added as an upper bound on scan parallelism if COMPUTE_PROCESSING_COST=true. If the number of scan ranges is fewer than the maximum parallelism allowed by the scan node's processing cost, then the processing cost will be clamped down to (min_processing_per_thread / number of scan ranges). Lowering MAX_FRAGMENT_INSTANCES_PER_NODE can also clamp down the scan processing cost. For interior fragments, a combination of MAX_FRAGMENT_INSTANCES_PER_NODE, PROCESSING_COST_MIN_THREADS, and the number of available cores per node is accounted for to determine maximum fragment parallelism per node. For the scan fragments, only the first two are considered to encourage frontend to choose a larger executor group as needed.

Type: INT

Default: 128

MAX_MEM_ESTIMATE_FOR_ADMISSION query option

Use the MAX_MEM_ESTIMATE_FOR_ADMISSION query option to set an upper limit on the memory estimates of a query as a workaround for over-estimates precluding a query from being admitted.

The query option takes effect when all of the below conditions are met:

- Memory-based admission control is enabled for the pool.
- The MEM_LIMIT query option is not set at the query, session, resource pool, or global level.

When the above conditions are met, MIN(MAX_MEM_ESTIMATE_FOR_ADMISSION, mem_estimate) is used for admission control.

Setting the MEM_LIMIT query option is usually a better option. Use the MAX_MEM_ESTIMATE_FOR_ADMISSION query option when it is not feasible to set MEM_LIMIT for each individual query.

Type: INT

Added in: Impala 3.1

MAX_NUM_RUNTIME_FILTERS query option

The MAX_NUM_RUNTIME_FILTERS query option sets an upper limit on the number of runtime filters that can be produced for each query.

Type: INT

Default: 10

Usage notes:

Each runtime filter imposes some memory overhead on the query. Depending on the setting of the RUNTIME_BLOOM_FILTER_SIZE query option, each filter might consume between 1 and 16 megabytes per plan fragment. There are typically 5 or fewer filters per plan fragment.

Impala evaluates the effectiveness of each filter, and keeps the ones that eliminate the largest number of partitions or rows. Therefore, this setting can protect against potential problems due to excessive memory overhead for filter production, while still allowing a high level of optimization for suitable queries.

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables.

Kudu consideration:

This query option affects only Bloom filters, not the min/max filters that are applied to Kudu tables. Therefore, it does not affect the performance of queries against Kudu tables.

MAX_RESULT_SPOOLING_MEM query option

Use the MAX_RESULT_SPOOLING_MEM query option to set the maximum amount of memory used when spooling query results.

If the amount of memory exceeds this value when spooling query results, all memory will most likely be spilled to disk.

The MAX_RESULT_SPOOLING_MEM query option is applicable only when query result spooling is enabled with the SPOOL_QUERY_RESULTS query option set to TRUE.

Setting the option to 0 or -1 means the memory is unbounded.

You cannot set this query option to values below -1.

Type: INT

Default: 100 * 1024 * 1024 (100 MB)

Added in: Impala 3.4

MAX_ROW_SIZE query option

Ensures that Impala can process rows of at least the specified size. (Larger rows might be successfully processed, but that is not guaranteed.) Applies when constructing intermediate or final rows in the result set. This setting prevents out-of-control memory use when accessing columns containing huge strings.

A numeric argument represents a size in bytes; you can also use a suffix of m or mb for megabytes, or g or gb for gigabytes. If you specify a value with unrecognized formats, subsequent queries fail with an error.

Type: INT

Default: 524288 (512 KB)

Usage notes:

If a query fails because it involves rows with long strings and/or many columns, causing the total row size to exceed MAX_ROW_SIZE bytes, increase the MAX_ROW_SIZE setting to accommodate the total bytes stored in the largest row. Examine the error messages for any failed queries to see the size of the row that caused the problem.

Impala attempts to handle rows that exceed the MAX_ROW_SIZE value where practical, so in many cases, queries succeed despite having rows that are larger than this setting.

Specifying a value that is substantially higher than actually needed can cause Impala to reserve more memory than is necessary to run the query.

In a Hadoop cluster with highly concurrent workloads and queries that process high volumes of data, traditional SQL tuning advice about minimizing wasted memory is worth remembering. For example, if a table has STRING columns where a single value might be multiple megabytes, make sure that the SELECT lists in queries only refer to columns that are actually needed in the result set, instead of using the SELECT * shorthand.

MAX_SCAN_RANGE_LENGTH query option

Maximum length of the scan range. Interacts with the number of HDFS blocks in the table to determine how many CPU cores across the cluster are involved with the processing for a query. (Each core processes one scan range.)

Lowering the value can sometimes increase parallelism if you have unused CPU capacity, but a too-small value can limit query performance because each scan range involves extra overhead.

Only applicable to HDFS tables. Has no effect on Parquet tables. Unspecified or 0 indicates backend default, which is the same as the HDFS block size for each table.

Although the scan range can be arbitrarily long, Impala internally uses an 8 MB read buffer so that it can query tables with huge block sizes without allocating equivalent blocks of memory.

The value can include unit specifiers, such as 100m or 100mb.

Type: Number

Default: 0

MAX_SPILLED_RESULT_SPOOLING_MEM query option

Use the MAX_SPILLED_RESULT_SPOOLING_MEM query option to set the maximum amount of memory that can be spilled when spooling query results.

If the amount of memory exceeds this value when spooling query results, the coordinator fragment will block until the client has consumed enough rows to free up more memory.

The MAX_SPILLED_RESULT_SPOOLING_MEM query option is applicable only when query result spooling is enabled with the SPOOL_QUERY_RESULTS query option set to TRUE.

The value must be greater than or equal to the value of MAX_RESULT_SPOOLING_MEM.

Setting the option to 0 or -1 means the memory is unbounded.

Values below -1 are not allowed for this query option.

Type: INT

Default: 1024 * 1024 * 1024 (1 GB)

Added in: Impala 3.4

MEM_LIMIT query option

The MEM_LIMIT query option defines the maximum amount of memory a query can allocate on each node. The total memory that can be used by a query is the MEM_LIMIT times the number of nodes.

There are two levels of memory limit for Impala. The ##mem_limit startup option sets an overall limit for the `impalad` process (which handles multiple queries concurrently). That process memory limit can be expressed either as a percentage of RAM available to the process such as -mem_limit=70% or as a fixed amount of memory, such as ##mem_limit=100gb. The memory available to the process is based on the host's physical memory and memory limits from Linux Control Groups. For example, if an `impalad` process is running in a Docker container on a host with 100 GB of memory, the memory available is 100 GB or the Docker container's memory limit, whichever is less.

The MEM_LIMIT query option, which you set through `impala-shell` or the SET statement in a JDBC or ODBC application, applies to each individual query. The MEM_LIMIT query option is usually expressed as a fixed size such as 10gb, and must always be less than the `impalad` memory limit.

If query processing approaches the specified memory limit on any node, either the per-query limit or the `impalad` limit, then the SQL operations will start to reduce their memory consumption, for example by writing the temporary data to disk (known as spilling to disk). The result is a query that completes successfully, rather than failing with an out-of-memory error. The tradeoff is decreased performance due to the extra disk I/O to write the temporary data and read it back in. The slowdown could potentially be significant. Thus, while this feature improves reliability, you should optimize your queries, system parameters, and hardware configuration to make this spilling a rare occurrence.

Type: Number

Units: A numeric argument represents memory size in bytes; you can also use a suffix of m or mb for megabytes, or more commonly g or gb for gigabytes. If you specify a value with unrecognized formats, subsequent queries fail with an error.

Default: 0 (unlimited)

Usage notes:

The MEM_LIMIT setting is primarily useful for production workloads. Impala's Admission Controller can be configured to automatically assign memory limits to queries and limit memory consumption of resource pools.

Use the output of the SUMMARY command in `impala-shell` to get a report of memory used for each phase of your most heavyweight queries on each node, and then set a MEM_LIMIT somewhat higher than that. See Understanding Performance using SUMMARY Report for usage information about the SUMMARY command.

The following examples show how to set the MEM_LIMIT query option using a fixed number of bytes, or suffixes representing gigabytes or megabytes.

```
[localhost:21000] > set mem_limit=3000000000;
MEM_LIMIT set to 3000000000
[localhost:21000] > select 5;
Query: select 5
+---+
| 5 |
+---+
| 5 |
+---+

[localhost:21000] > set mem_limit=3g;
MEM_LIMIT set to 3g
[localhost:21000] > select 5;
Query: select 5
+---+
| 5 |
+---+
| 5 |
+---+

[localhost:21000] > set mem_limit=3gb;
MEM_LIMIT set to 3gb
[localhost:21000] > select 5;
+---+
| 5 |
+---+
| 5 |
+---+
```

```
[localhost:21000] > set mem_limit=3m;
MEM_LIMIT set to 3m
[localhost:21000] > select 5;
+---+
| 5 |
+---+
| 5 |
+---+
[localhost:21000] > set mem_limit=3mb;
MEM_LIMIT set to 3mb
[localhost:21000] > select 5;
+---+
| 5 |
+---+
```

The following examples show how unrecognized MEM_LIMIT values lead to errors for subsequent queries.

```
[localhost:21000] > set mem_limit=3pb;
MEM_LIMIT set to 3pb
[localhost:21000] > select 5;
ERROR: Failed to parse query memory limit from '3pb'.

[localhost:21000] > set mem_limit=xyz;
MEM_LIMIT set to xyz
localhost:21000] > select 5;
Query: select 5
ERROR: Failed to parse query memory limit from 'xyz'.
```

The following examples shows the automatic query cancellation when the MEM_LIMIT value is exceeded on any host involved in the Impala query. First it runs a successful query and checks the largest amount of memory used on any node for any stage of the query. Then it sets an artificially low MEM_LIMIT setting so that the same query cannot run.

```
[localhost:21000] > select count(*) from customer;
Query: select count(*) from customer
+----------+
| count(*) |
+----------+
| 150000   |
+----------+

[localhost:21000] > select count(distinct c_name) from customer;
Query: select count(distinct c_name) from customer
+------------------------+
| count(distinct c_name) |
+------------------------+
| 150000                 |
+------------------------+

[localhost:21000] > summary;
+--------------+--------+--------+----------+----------+---------
+------------+----------+---------------+---------------+
| Operator     | #Hosts | #Inst  | Avg Time | Max Time | #Rows
 | Est. #Rows | Peak Mem | Est. Peak Mem | Detail        |
+--------------+--------+--------+----------+----------+---------
+------------+----------+---------------+---------------+
| 06:AGGREGATE | 1      | 1      | 230.00ms | 230.00ms | 1
 | 1          | 16.00 KB | -1 B          | FINALIZE      |
| 05:EXCHANGE  | 1      | 1      | 43.44us  | 43.44us  | 1
| 1          | 0 B      | -1 B          | UNPARTITIONED |
```

```
| 02:AGGREGATE | 1      | 1      | 227.14ms | 227.14ms | 1
 | 1          | 12.00 KB | 10.00 MB      |               |
| 04:AGGREGATE | 1      | 1      | 126.27ms | 126.27ms | 150.00K
 | 150.00K    | 15.17 MB | 10.00 MB      |               |
| 03:EXCHANGE  | 1      | 1      | 44.07ms  | 44.07ms  | 150.00K
 | 150.00K    | 0 B      | 0 B           | HASH(c_name)  |
| 01:AGGREGATE | 1      | 1      | 361.94ms | 361.94ms | 150.00K
 | 150.00K    | 23.04 MB | 10.00 MB      |               |
| 00:SCAN HDFS | 1      | 1      | 43.64ms  | 43.64ms  | 150.00K
 | 150.00K    | 24.19 MB | 64.00 MB      | tpch.customer |
+--------------+--------+--------+----------+----------+---------
+------------+----------+---------------+---------------+

[localhost:21000] > set mem_limit=15mb;
MEM_LIMIT set to 15mb
[localhost:21000] > select count(distinct c_name) from customer;
Query: select count(distinct c_name) from customer
ERROR:
Rejected query from pool default-pool: minimum memory reservation
 is greater than memory available to the query
for buffer reservations. Memory reservation needed given the curr
ent plan: 38.00 MB. Adjust either the mem_limit
or the pool config (max-query-mem-limit, min-query-mem-limit) for
 the query to allow the query memory limit to be
at least 70.00 MB. Note that changing the mem_limit may also c
hange the plan. See the query profile for more
information about the per-node memory requirements.
```

MEM_LIMIT_COORDINATORS query option

The new MEM_LIMIT_COORDINATORS query option functions similarly to the MEM_LIMIT option but sets the query memory limit only on coordinators. This new option addresses the issue related to MEM_LIMIT and is recommended in scenarios where the query needs higher or lower memory on coordinators compared to the planner estimates.

Note: the MEM_LIMIT_COORDINATORS option does not work with MEM_LIMIT. If you set both, only MEM_LIMIT applies.

MIN_SPILLABLE_BUFFER_SIZE query option

Specifies the minimum size for a memory buffer used when the spill-to-disk mechanism is activated, for example for queries against a large table with no statistics, or large join operations.

Type: INT

Default: 65536 (64 KB)

Units: A numeric argument represents memory size in bytes; you can also use a suffix of m or mb for megabytes, or more commonly g or gb for gigabytes. If you specify a value with unrecognized formats, subsequent queries fail with an error.

Usage notes:

This query option sets a lower bound on the size of the internal buffer size that can be used during spill-to-disk operations. The actual size of the buffer is chosen by the query planner.

If overall query performance is limited by the time needed for spilling, consider increasing the MIN_SPILLABLE_BUFFER_SIZE setting. Larger buffer sizes result in Impala issuing larger I/O requests to storage devices, which might result in higher throughput, particularly on rotational disks.

The tradeoff with a large value for this setting is increased memory usage during spill-to-disk operations. Reducing this value may reduce memory consumption.

To determine if the value for this setting is having an effect by capping the spillable buffer size, you can see the buffer size chosen by the query planner for a particular query. EXPLAIN the query while the setting EXPLAIN_LEVEL=2 is in effect.

MT_DOP query option

Sets the degree of parallelism used for certain operations that can benefit from multithreaded execution. You can specify values higher than zero to find the ideal balance of response time, memory usage, and CPU usage during statement processing.

Note:

The Impala execution engine is being revamped incrementally to add additional parallelism within a single host for certain statements and kinds of operations. The setting MT_DOP=0 uses the “old” code path with limited intra-node parallelism.

Higher value of MT_DOP is directly proportional to resource requirements.

Currently, MT_DOP support varies by statement type:

- COMPUTE [INCREMENTAL] STATS. Impala automatically sets MT_DOP=4 for COMPUTE STATS and COMPUTE INCREMENTAL STATS statements on Parquet tables.
- SELECT statements. MT_DOP is 0 by default for SELECT statements but can be set to a value greater than 0 to control intra-node parallelism. This may be useful to tune query performance and in particular to reduce execution time of long-running, CPU-intensive queries.
- DML statements. MT_DOP values greater than zero are not currently supported for DML statements. DML statements will produce an error if MT_DOP is set to a non-zero value.
- In Impala 3.4 (Cloudera 1.2) and earlier, not all SELECT statements support setting MT_DOP. Specifically, only scan and aggregation operators, and local joins that do not need data exchanges (such as for nested types) are supported. Other SELECT statements produce an error if MT_DOP is set to a non-zero value.

Type: INT

Default: 0

Because COMPUTE STATS and COMPUTE INCREMENTAL STATS statements for Parquet tables benefit substantially from extra intra-node parallelism, Impala automatically sets MT_DOP=4 when computing stats for Parquet tables.

Range: 0 to 64

Examples:

Note:

Any timing figures in the following examples are on a small, lightly loaded development cluster. Your mileage may vary. Speedups depend on many factors, including the number of rows, columns, and partitions within each table.

The following example shows how to run a COMPUTE STATS statement against a Parquet table with or without an explicit MT_DOP setting:

```
-- Explicitly setting MT_DOP to 0 selects the old code path.
set mt_dop = 0;
MT_DOP set to 0

-- The analysis for the billion rows is distributed among hosts,
-- but uses only a single core on each host.
compute stats billion_rows_parquet;
```

```
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 2 column(s). |
+-----------------------------------------+

drop stats billion_rows_parquet;

-- Using 4 logical processors per host is faster.
set mt_dop = 4;
MT_DOP set to 4

compute stats billion_rows_parquet;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 2 column(s). |
+-----------------------------------------+

drop stats billion_rows_parquet;

-- Unsetting the option reverts back to its default.
-- Which for COMPUTE STATS and a Parquet table is 4,
-- so again it uses the fast path.
unset MT_DOP;
Unsetting option MT_DOP

compute stats billion_rows_parquet;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 2 column(s). |
+-----------------------------------------+
```

The following example shows the effects of setting MT_DOP for a query on a Parquet table:

```
set mt_dop = 0;
MT_DOP set to 0

-- COUNT(DISTINCT) for a unique column is CPU-intensive.
select count(distinct id) from billion_rows_parquet;
+--------------------+
| count(distinct id) |
+--------------------+
| 1000000000         |
+--------------------+
Fetched 1 row(s) in 67.20s

set mt_dop = 16;
MT_DOP set to 16

-- Introducing more intra-node parallelism for the aggregation
-- speeds things up, and potentially reduces memory overhead by
-- reducing the number of scanner threads.
select count(distinct id) from billion_rows_parquet;
+--------------------+
| count(distinct id) |
+--------------------+
| 1000000000         |
+--------------------+
```

```
Fetched 1 row(s) in 17.19s
```

The following example shows how queries that are not compatible with non-zero MT_DOP settings produce an error when MT_DOP is set:

```
set mt_dop=1;
MT_DOP set to 1

insert into a1
select * from a2;
ERROR: NotImplementedException: MT_DOP not supported for DML st
atements.
```

Related Information:

Compute Stats Statement

Impala Aggregate Functions

NUM_NODES query option

Limit the number of nodes that process a query, typically during debugging.

Type: Number

Allowed values: Only accepts the values 0 (meaning all nodes) or 1 (meaning all work is done on the coordinator node).

Default: 0

Usage notes:

If you are diagnosing a problem that you suspect is due to a timing issue due to distributed query processing, you can set NUM_NODES=1 to verify if the problem still occurs when all the work is done on a single node.

You might set the NUM_NODES option to 1 briefly, during INSERT or CREATE TABLE AS SELECT statements. Normally, those statements produce one or more data files per data node. If the write operation involves small amounts of data, a Parquet table, and/or a partitioned table, the default behavior could produce many small files when intuitively you might expect only a single output file. SET NUM_NODES=1 turns off the “distributed” aspect of the write operation, making it more likely to produce only one or a few data files.

Warning:

Because this option results in increased resource utilization on a single host, it could cause problems due to contention with other Impala statements or high resource usage. Symptoms could include queries running slowly, exceeding the memory limit, or appearing to stop responding. Use it only in a single-user development/test environment; do not use it in a production environment or in a cluster with a high-concurrency or high-volume or performance-critical workload.

NUM_ROWS_PRODUCED_LIMIT query option

The NUM_ROWS_PRODUCED_LIMIT query option limits the number of rows produced by a query. A query is canceled when its execution produces more rows than the specified limit set by the NUM_ROWS_PRODUCED_LIMIT option.

This limit only applies when the results are returned to a client, for example, to a SELECT query, but not to an INSERT query.

The default value of 0 specifies that there is no limit on the number of rows produced.

Type: INT

Allowed values: 0 or positive numbers

Default: 0

NUM_SCANNER_THREADS query option

Maximum number of scanner threads (on each node) used for each query. By default, Impala uses as many cores as are available (one thread per core). You might lower this value if queries are using excessive resources on a busy cluster. Impala imposes a maximum value automatically, so a high value has no practical effect.

Type: Number

Default: 0

OPTIMIZE_PARTITION_KEY_SCANS query option

Enables a fast code path for queries that apply simple aggregate functions to partition key columns: MIN(*KEY_COLUMN*), MAX(*KEY_COLUMN*), or COUNT(DISTINCT *KEY_COLUMN*).

Type: BOOLEAN

Default: FALSE (0)

Usage notes:

This optimization speeds up common “introspection” operations over partition key columns, for example determining the distinct values of partition keys.

This optimization does not apply to SELECT statements that reference columns that are not partition keys. It also only applies when all the partition key columns in the SELECT statement are referenced in one of the following contexts:

- Within a MAX() or MAX() aggregate function or as the argument of any aggregate function with the DISTINCT keyword applied.
- Within a WHERE, GROUP BY or HAVING clause.

This optimization is enabled by a query option because it skips some consistency checks and therefore can return slightly different partition values if partitions are in the process of being added, dropped, or loaded outside of Impala. Queries might exhibit different behavior depending on the setting of this option in the following cases:

- If files are removed from a partition using HDFS or other non-Impala operations, there is a period until the next REFRESH of the table where regular queries fail at run time because they detect the missing files. With this optimization enabled, queries that evaluate only the partition key column values (not the contents of the partition itself) succeed, and treat the partition as if it still exists.
- If a partition contains any data files, but the data files do not contain any rows, a regular query considers that the partition does not exist. With this optimization enabled, the partition is treated as if it exists.

  If the partition includes no files at all, this optimization does not change the query behavior: the partition is considered to not exist whether or not this optimization is enabled.

OPTIMIZE_SIMPLE_LIMIT query option

This new planner query option OPTIMIZE_SIMPLE_LIMIT optimizes the planning time for simple limit queries by only considering a minimal set of partitions whose file descriptors add up to N (the specified limit). Each file is conservatively estimated to contain 1 row.

This reduces the number of partitions processed by the planner which, according to query profiling, has been the main contributor to the planning time especially for large numbers of partitions.

Further, within each partition, only the number of non-empty files that brings the total to N will be considered. For the optimization to be considered, this option should be always set to "true".

This is an opt-in optimization and also applies to subqueries and view.

ORC_SCHEMA_RESOLUTION query option

The ORC_SCHEMA_RESOLUTION query option allows Impala to look up columns within ORC files by column names, rather than by column indices, when necessary.

The allowed values are:

- POSITION (0)
- NAME (1)

Type: Enum of the above-allowed values

Usage notes:

By default, Impala looks up columns within an ORC file based on the order of columns in the table.

Note: Hive only writes column names to ORC files when deployed in Cloudera. So this feature is only required when deployed in Cloudera.

Impact of column renaming

Position-based resolution is resilient to column renaming however, a name-based resolution will fail when you rename a column. The following example demonstrates the impact of column renaming on position-based and name-based resolution.

1. View the details of the table test_orc

```
select * from test_orc;
+---------+------------+------------+
| int_col | string_col | double_col |
+---------+------------+------------+
| 0       | A          | 0.5        |
| 1       | B          | 1.5        |
| 2       | C          | 2.5        |
+---------+------------+------------+
```

2. Rename the column “double_col” to “new_double_col”

```
alter table test_orc change double_col new_double_col double;
```

3. Set resolving ORC file columns by Index

```
set orc_schema_resolution=POSITION;
```

4. View the table

```
select int_col, string_col, new_double_col from test_orc;
+---------+------------+----------------+
| int_col | string_col | new_double_col |
+---------+------------+----------------+
| 0       | A          | 0.5            |
| 1       | B          | 1.5            |
| 2       | C          | 2.5            |
+---------+------------+----------------+
```

The above query results show that renaming a column did not impact when you resolved the ORC file columns by POSITION. However, if you resolve ORC file columns by NAME, set orc_sche ma_resolution=NAME, viewing the table details will fail to fetch the renamed columns.

```
select int_col, string_col, new_double_col from test_orc;
```

```
+---------+------------+----------------+
| int_col | string_col | new_double_col |
+---------+------------+----------------+
| 0       | A          | NULL           |
| 1       | B          | NULL           |
| 2       | C          | NULL           |
+---------+------------+----------------+
```

PARQUET_COMPRESSION_CODEC query option

Deprecated. Use COMPRESSION_CODEC in Impala 2.0 and later.

PARQUET_ANNOTATE_STRINGS_UTF8 query option

Causes Impala INSERT and CREATE TABLE AS SELECT statements to write Parquet files that use the UTF-8 annotation for STRING columns.

Type: BOOLEAN

Default: FALSE (0)

Usage notes:

By default, Impala represents a STRING column in Parquet as an unannotated binary field.

Impala always uses the UTF-8 annotation when writing CHAR and VARCHAR columns to Parquet files. An alternative to using the query option is to cast STRING values to VARCHAR.

This option is to help make Impala-written data more interoperable with other data processing engines. Impala itself currently does not support all operations on UTF-8 data. Although data processed by Impala is typically represented in ASCII, it is valid to designate the data as UTF-8 when storing on disk, because ASCII is a subset of UTF-8.

PARQUET_ARRAY_RESOLUTION query option

The PARQUET_ARRAY_RESOLUTION query option controls the behavior of the indexed-based resolution for nested arrays in Parquet.

In Parquet, you can represent an array using a 2-level or 3-level representation. The modern, standard representation is 3-level. The legacy 2-level scheme is supported for compatibility with older Parquet files. However, there is no reliable metadata within Parquet files to indicate which encoding was used. It is even possible to have mixed encodings within the same file if there are multiple arrays. The PARQUET_ARRAY_RESOLUTION option controls the process of resolution that is to match every column/field reference from a query to a column in the Parquet file.

The supported values for the query option are:

- THREE_LEVEL: Assumes arrays are encoded with the 3-level representation, and does not attempt the 2-level resolution.
- TWO_LEVEL: Assumes arrays are encoded with the 2-level representation, and does not attempt the 3-level resolution.
- TWO_LEVEL_THEN_THREE_LEVEL: First tries to resolve assuming a 2-level representation, and if unsuccessful, tries a 3-level representation.

All of the above options resolve arrays encoded with a single level.

A failure to resolve a column/field reference in a query with a given array resolution policy does not necessarily result in a warning or error returned by the query. A mismatch might be treated like a missing column (returns NULL values), and it is not possible to reliably distinguish the 'bad resolution' and 'legitimately missing column' cases.

The name-based policy generally does not have the problem of ambiguous array representations. You specify to use the name-based policy by setting the PARQUET_FALLBACK_SCHEMA_RESOLUTION query option to NAME.

Type: Enum of TWO_LEVEL, TWO_LEVEL_THEN_THREE_LEVEL, and THREE_LEVEL

Default: THREE_LEVEL

Examples:

EXAMPLE A: The following Parquet schema of a file can be interpreted as a 2-level or 3-level:

```
ParquetSchemaExampleA {
  optional group single_element_groups (LIST) {
    repeated group single_element_group {
      required int64 count;
    }
  }
}
```

The following table schema corresponds to a 2-level interpretation:

```
CREATE TABLE t (col1 array<struct<f1: bigint>>) STORED AS PARQU
ET;
```

Successful query with a 2-level interpretation:

```
SET PARQUET_ARRAY_RESOLUTION=TWO_LEVEL;
SELECT ITEM.f1 FROM t.col1;
```

The following table schema corresponds to a 3-level interpretation:

```
CREATE TABLE t (col1 array<bigint>) STORED AS PARQUET;
```

Successful query with a 3-level interpretation:

```
SET PARQUET_ARRAY_RESOLUTION=THREE_LEVEL;
SELECT ITEM FROM t.col1
```

EXAMPLE B: The following Parquet schema of a file can be only be successfully interpreted as a 2-level:

```
ParquetSchemaExampleB {
  required group list_of_ints (LIST) {
    repeated int32 list_of_ints_tuple;
  }
}
```

The following table schema corresponds to a 2-level interpretation:

```
CREATE TABLE t (col1 array<int>) STORED AS PARQUET;
```

Successful query with a 2-level interpretation:

```
SET PARQUET_ARRAY_RESOLUTION=TWO_LEVEL;
SELECT ITEM FROM t.col1
```

Unsuccessful query with a 3-level interpretation. The query returns NULLs as if the column was missing in the file:

```
SET PARQUET_ARRAY_RESOLUTION=THREE_LEVEL;
SELECT ITEM FROM t.col1
```

PARQUET_DICTIONARY_FILTERING query option

The PARQUET_DICTIONARY_FILTERING query option controls whether Impala uses dictionary filtering for Parquet files.

To efficiently process a highly selective scan query, when this option is enabled, Impala checks the values in the Parquet dictionary page and determines if the whole row group can be thrown out.

A column chunk is purely dictionary encoded and can be used by dictionary filtering if any of the following conditions are met:

1. If the encoding_stats is in the Parquet file, dictionary filtering uses it to determine if there are only dictionary encoded pages (i.e. there are no data pages with an encoding other than PLAIN_DICTIONARY).
2. If the encoding stats are not present, dictionary filtering looks at the encodings. The column is purely dictionary encoded if both of the conditions satisfy:
   - PLAIN_DICTIONARY is present.
   - Only PLAIN_DICTIONARY, RLE, or BIT_PACKED encodings are listed.
3. Dictionary filtering works for the Parquet dictionaries with less than 40000 values if the file was written by Impala 2.9.

In the query runtime profile output for each Impalad instance, the NumDictFilteredRowGroups field in the SCAN node section shows the number of row groups that were skipped based on dictionary filtering.

Note that row groups can be filtered out by Parquet statistics, and in such cases, dictionary filtering will not be considered.

The supported values for the query option are:

- TRUE (1): Use dictionary filtering.
- FALSE (0): Do not use dictionary filtering
- Any other values are treated as FALSE.

Type: BOOLEAN

Default: TRUE (1)

PARQUET_FALLBACK_SCHEMA_RESOLUTION query option

The PARQUET_FALLBACK_SCHEMA_RESOLUTION query option allows Impala to look up columns within Parquet files by column name, rather than column order, when necessary. The allowed values are:

- POSITION (0)
- NAME (1)

Type: Enum of the above allowed values

Usage notes:

By default, Impala looks up columns within a Parquet file based on the order of columns in the table. The name setting for this option enables behavior for Impala queries similar to the Hive setting parquet.column.index access=false. It also allows Impala to query Parquet files created by Hive with the parquet.column.index.access=false setting in effect.

PARQUET_FILE_SIZE query option

Specifies the maximum size of each Parquet data file produced by Impala INSERT statements in bytes, or with a trailing m or g character to indicate megabytes or gigabytes.

With tables that are small or finely partitioned, the default Parquet block size (formerly 1 GB, now 256 MB in Impala 2.0 and later) could be much larger than needed for each data file. For INSERT operations into such tables, you can increase parallelism by specifying a smaller PARQUET_FILE_SIZE value, resulting in more HDFS blocks that can be processed by different nodes.

Type: Number, with optional unit specifier

Important:

Currently, the maximum value for this setting is 1 gigabyte (1g). Setting a value higher than 1 gigabyte could result in errors during an INSERT operation.

Default: 0 (produces files with a target size of 256 MB; files might be larger for very wide tables)

PARQUET_OBJECT_STORE_SPLIT_SIZE

Use the PARQUET_OBJECT_STORE_SPLIT_SIZE query option to control the Parquet split sizes on non-block based stores, e.g. S3, ADLS, etc.

Type: INT

Allowed values: The value must be greater than or equal to 1 MB.

Default: 256 MB

Added in: Impala 3.4 / Cloudera 1.2

PARQUET_PAGE_ROW_COUNT_LIMIT query option

Use the PARQUET_PAGE_ROW_COUNT_LIMIT query option to set the maximum number of rows that can be written on a single Parquet data page. By default there is no row count limit.

Type: INT

Allowed values: Positive integers

Added in: Impala 3.3

PARQUET_READ_PAGE_INDEX query option

Use the PARQUET_READ_PAGE_INDEX query option to disable or enable using the Parquet page index during scans. The page index contains min/max statistics at the page-level granularity. It can be used to skip pages and rows that do not match the conditions in the WHERE clause.

This option enables the same optimization as the PARQUET_READ_STATISTICS at the finer grained page level.

Impala supports filtering based on Parquet statistics:

- Of the types: BOOLEAN, INT, DECIMAL, STRING, TIMESTAMP, DATE
- For simple predicates of the forms: <slot> <op> <constant> or <constant> <op> <slot>, where <op> is LT, LE, GE, GT, and EQ

The supported values for the query option are:

- TRUE (1): Read the page-level statistics from the Parquet page index during query processing and filter out pages based on the statistics.
- FALSE (0): Do not use the Parquet page index.
- Any other values are treated as false.

Type: BOOLEAN

Default: TRUE (1)

PARQUET_READ_STATISTICS query option

The PARQUET_READ_STATISTICS query option controls whether to read statistics from Parquet files and use them during query processing.

Parquet stores min/max stats which can be used to skip reading row groups if they don't qualify a certain predicate. When this query option is set to true, Impala reads the Parquet statistics and skips reading row groups that do not match the conditions in the WHERE clause.

Impala supports filtering based on Parquet statistics:

- Of the numerical types for the old version of the statistics: BOOLEAN, INTEGER, FLOAT
- Of the types for the new version of the statistics (starting in impala 2.8): BOOLEAN, INTEGER, FLOAT, DECIMAL, STRING, TIMESTAMP, DATE
- For simple predicates of the forms: <slot> <op> <constant> or <constant> <op> <slot>, where <op> is LT, LE, GE, GT, and EQ

The PARQUET_READ_STATISTICS option provides a workaround when dealing with files that have corrupt Parquet statistics and unknown errors.

In the query runtime profile output for each Impalad instance, the NumStatsFilteredRowGroups field in the SCAN node section shows the number of row groups that were skipped based on Parquet statistics.

The supported values for the query option are:

- TRUE (1): Read statistics from Parquet files and use them in query processing.
- FALSE (0): Do not use Parquet read statistics.
- Any other values are treated as FALSE.

Type: BOOLEAN

Default: TRUE

PARQUET_WRITE_PAGE_INDEX query option

The PARQUET_WRITE_PAGE_INDEX query option disables or enables the Parquet page index writing.

Impala writes page-level statistics into the Parquet page index of the types: INT, DECIMAL, STRING, TIMESTAMP, DATE

The supported values for the query option are:

- TRUE (1): Write the Parquet page index when creating Parquet files.
- FALSE (0): Do not write the Parquet page index when creating Parquet files.
- Any other values are treated as false.

Type: BOOLEAN

Default: TRUE (1)

Added in: Impala 3.3

PREFETCH_MODE query option

Determines whether the prefetching optimization is applied during join query processing.

Allowed values are:

- NONE (0)
- HT_BUCKET (1): Hash table buckets are prefetched during join query processing.

Type: Enum of the above allowed values

Default: 1 (HT_BUCKET)

QUERY_TIMEOUT_S query option

Sets the idle query timeout value for the session, in seconds. Queries that sit idle for longer than the timeout value are automatically cancelled. If the system administrator specified the --idle_query _timeout startup option, QUERY_TIMEOUT_S must be smaller than or equal to the --idle_query _timeout value.

Note:

The timeout clock for queries and sessions only starts ticking when the query or session is idle.

For queries, this means the query has results ready but is waiting for a client to fetch the data. A query can run for an arbitrary time without triggering a timeout, because the query is computing results rather than sitting idle waiting for the results to be fetched. The timeout period is intended to prevent unclosed queries from consuming resources and taking up slots in the admission count of running queries, potentially preventing other queries from starting.

For sessions, this means that no query has been submitted for some period of time.

Type: Number

Default: 0 (no timeout if --idle_query_timeout not in effect; otherwise, use --idle_query_timeout value)

REFRESH_UPDATED_HMS_PARTITIONS query option

As the name implies the query option REFRESH_UPDATED_HMS_PARTITIONS is used to refresh any updated HMS partitions.

This option is disabled by default so that the performance is not compromised when refreshing a table. However, for certain corner case scenarios refresh table command does not detect changed partitions. In case of the default refresh, catalogd detects, adds any new partitions and removes any partitions which are not present in HMS anymore. However, it does not update any partitions that changed (eg. change of location). When this query option is enabled, the refresh table command will detect certain changes to the partitions and update them accordingly. Currently, catalogd will update the partitions if any of the following StorageDescriptor properties have been modified. 1. Partition Location 2. Partition Fileformat. 3 SerdeInfo. 4. Partition schema changes. 5. Partition bucketInfo.

Type:BOOLEAN; recognized values are 1 and 0, or true and false; any other value will be interpreted as false.

Default:FALSE (shown as 0 in output of SET statement).

REPLICA_PREFERENCE query option

The REPLICA_PREFERENCE query option lets you distribute the work more evenly if hotspots and bottlenecks persist. It causes the access cost of all replicas of a data block to be considered equal to or worse than the configured value. This allows Impala to schedule reads to suboptimal replicas (e.g. local in the presence of cached ones) in order to distribute the work across more executor nodes.

Allowed values are:

- CACHE_LOCAL (0)
- DISK_LOCAL (2)
- REMOTE (4)

Type: Enum

Default: CACHE_LOCAL (0)

Usage Notes:

By default Impala selects the best replica it can find in terms of access cost. The preferred order is cached, local, and remote. With REPLICA_PREFERENCE, the preference of all replicas are

capped at the selected value. For example, when REPLICA_PREFERENCE is set to DISK_LOCAL, cached and local replicas are treated with the equal preference. When set to REMOTE, all three types of replicas, cached, local, remote, are treated with equal preference.

REQUEST_POOL query option

The pool or queue name that queries should be submitted to. Only applies when you enable the Impala admission control feature. Specifies the name of the pool used by requests from Impala to the resource manager.

Type: STRING

Default: An empty string (use the user-to-pool mapping defined by an `impalad` startup option in the Impala configuration file)

RESOURCE_TRACE_RATIO query option

The RESOURCE_TRACE_RATIO query option specifies the ratio of queries where the CPU usage info will be included in the profiles. Collecting CPU usage and sending it around adds a slight overhead during query execution. This query option lets you control whether to collect additional information to diagnose the resource usage.

For example, setting RESOURCE_TRACE_RATIO=1 adds a trace of the CPU usage to the profile of each query.

Setting RESOURCE_TRACE_RATIO=0.5 means that randomly selected half of all queries will have that information collected by the coordinator and included in the profiles.

Setting RESOURCE_TRACE_RATIO=0 means that CPU usage will not be tracked and included in the profiles.

Values from 0 to 1 are allowed.

Type: Number

Default: 0

Added in: Impala 3.2

RETRY_FAILED_QUERIES query option

Use the RETRY_FAILED_QUERIES query option to control whether or not queries are transparently retried on cluster membership changes.

Cluster membership changes typically occur when an impalad crashes, or if the node is blacklisted by the Impala Coordinator. If a SELECT query fails due to a cluster membership change, the Coordinator will cancel and unregister the running query and then launch a retry of the query. For example, if one of the executor nodes fails during query execution, the query fails but is transparently re-executed, either with the executor node immediately replaced, or with a temporarily reduced number of executor nodes. This feature supports retrying the entire query and NOT the individual query fragments. INSERT and DDL queries will NOT be retried.

Note: Query retry will be skipped if the query returned any results to the client. To avoid this, enable Result Spooling and spool_all_results_for_retries.

Type: BOOLEAN

Default: TRUE

RUNTIME_BLOOM_FILTER_SIZE query option

Size (in bytes) of Bloom filter data structure used by the runtime filtering feature.

Important:

In Impala 2.6 and higher, this query option only applies as a fallback, when statistics are not available. By default, Impala estimates the optimal size of the Bloom filter structure regardless of the setting for this option. (This is a change from the original behavior in Impala 2.5.)

In Impala 2.6 and higher, when the value of this query option is used for query planning, it is constrained by the minimum and maximum sizes specified by the RUNTIME_FILTER_MIN_SIZE and RUNTIME_FILTER_MAX_SIZE query options. The filter size is adjusted upward or downward if necessary to fit within the minimum/maximum range.

Type: INTEGER

Default: 1048576 (1 MB)

Maximum: 16 MB

Usage notes:

This setting affects optimizations for large and complex queries, such as dynamic partition pruning for partitioned tables, and join optimization for queries that join large tables. Larger filters are more effective at handling higher cardinality input sets, but consume more memory per filter.

If your query filters on high-cardinality columns (for example, millions of different values) and you do not get the expected speedup from the runtime filtering mechanism, consider doing some benchmarks with a higher value for RUNTIME_BLOOM_FILTER_SIZE. The extra memory devoted to the Bloom filter data structures can help make the filtering more accurate.

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables.

Because the effectiveness of this setting depends so much on query characteristics and data distribution, you typically only use it for specific queries that need some extra tuning, and the ideal value depends on the query. Consider setting this query option immediately before the expensive query and unsetting it immediately afterward.

Kudu consideration:

This query option affects only Bloom filters, not the min/max filters that are applied to Kudu tables. Therefore, it does not affect the performance of queries against Kudu tables.

RUNTIME_FILTER_MAX_SIZE query option

The RUNTIME_FILTER_MAX_SIZE query option adjusts the settings for the runtime filtering feature. This option defines the maximum size for a filter, no matter what the estimates produced by the planner are. This value also overrides any lower number specified for the RUNTIME_BLOOM_FILTER_SIZE query option. Filter sizes are rounded up to the nearest power of two.

Type: INTEGER

Default: 1048576 (1 MB)

Usage notes:

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables.

Kudu consideration:

This query option affects only Bloom filters, not the min/max filters that are applied to Kudu tables. Therefore, it does not affect the performance of queries against Kudu tables.

RUNTIME_FILTER_MIN_SIZE query option

The RUNTIME_FILTER_MIN_SIZE query option adjusts the settings for the runtime filtering feature. This option defines the minimum size for a filter, no matter what the estimates produced by the planner are. This value also overrides any lower number specified for the RUNTIME_BLOOM_FILTER_SIZE query option. Filter sizes are rounded up to the nearest power of two.

Type: INTEGER

Default: 0 (meaning use the value from the corresponding `impalad` startup option)

Usage notes:

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables.

Kudu consideration:

This query option affects only Bloom filters, not the min/max filters that are applied to Kudu tables. Therefore, it does not affect the performance of queries against Kudu tables.

RUNTIME_FILTER_MODE query option

The RUNTIME_FILTER_MODE query option adjusts the settings for the runtime filtering feature. It turns this feature on and off, and controls how extensively the filters are transmitted between hosts.

Allowed values are:

- OFF (0)
- LOCAL (1)
- GLOBAL (2)

Type: Enum of the above allowed values

Default: GLOBAL (2)

Usage notes:

The default is GLOBAL setting is recommended for a wide variety of workloads, to provide best performance with “out of the box” settings.

The lowest setting of LOCAL does a similar level of optimization (such as partition pruning) as in earlier Impala releases. This setting was the default in Impala 2.5, to allow for a period of post-upgrade testing for existing workloads. This setting is suitable for workloads with non-performance-critical queries, or if the coordinator node is under heavy CPU or memory pressure.

You might change the setting to OFF if your workload contains many queries involving partitioned tables or joins that do not experience a performance increase from the runtime filters feature. If the overhead of producing the runtime filters outweighs the performance benefit for queries, you can turn the feature off entirely.

RUNTIME_FILTER_WAIT_TIME_MS query option

The RUNTIME_FILTER_WAIT_TIME_MS query option adjusts the settings for the runtime filtering feature. It specifies a time in milliseconds that each scan node waits for runtime filters to be produced by other plan fragments.

Type: INTEGER

Default: 0 (meaning use the value from the corresponding `impalad` startup option)

Usage notes:

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables

S3_SKIP_INSERT_STAGING query option

Speeds up INSERT operations on tables or partitions residing on the Amazon S3 filesystem. The tradeoff is the possibility of inconsistent data left behind if an error occurs partway through the operation.

By default, Impala write operations to S3 tables and partitions involve a two-stage process. Impala writes intermediate files to S3, then (because S3 does not provide a “rename” operation) those intermediate files are copied to their final location, making the process more expensive as on a filesystem that supports renaming or moving files. This query option makes Impala skip the intermediate files, and instead write the new data directly to the final destination.

Type: BOOLEAN

Default: TRUE (1)

Usage notes:

Important:

If a host that is participating in the INSERT operation fails partway through the query, you might be left with a table or partition that contains some but not all of the expected data files. Therefore, this option is most appropriate for a development or test environment where you have the ability to reconstruct the table if a problem during INSERT leaves the data in an inconsistent state.

The timing of file deletion during an INSERT OVERWRITE operation makes it impractical to write new files to S3 and delete the old files in a single operation. Therefore, this query option only affects regular INSERT statements that add to the existing data in a table, not INSERT OVERWRITE statements. Use TRUNCATE TABLE if you need to remove all contents from an S3 table before performing a fast INSERT with this option enabled.

Performance improvements with this option enabled can be substantial. The speed increase might be more noticeable for non-partitioned tables than for partitioned tables.

SCAN_BYTES_LIMIT query option

The SCAN_BYTES_LIMIT query option sets a limit on the bytes scanned by HDFS and HBase SCAN operations. If a query is still executing when the query’s coordinator detects that it has exceeded the limit, the query is terminated with an error. The option is intended to prevent runaway queries that scan more data than is intended.

For example, an Impala administrator could set a default value of SCAN_BYTES_LIMIT=100GB for a resource pool to automatically stop queries that scan more than 100 GB of data (see Impala Admission Control and Query Queuing for information about default query options). If a user accidentally omits a partition filter in a WHERE clause and runs a large query that scans a lot of data, the query will be automatically terminated after it scans more data than the SCAN_BYTES_LIMIT.

You can override the default value per-query or per-session, in the same way as other query options, if you do not want the default SCAN_BYTES_LIMIT value to apply to a specific query or session.

Note:

- Only data actually read from the underlying storage layer is counted towards the limit. E.g. Impala’s Parquet scanner employs several techniques to skip over data in a file that is not relevant to a specific query, so often only a fraction of the file size is counted towards SCAN_BYTES_LIMIT.
- As of Impala 3.1, bytes scanned by Kudu tablet servers are not counted towards the limit.

Because the checks are done periodically, the query may scan over the limit at times.

Type: Number

Units:

- A numeric argument represents memory size in bytes.
- Specify a suffix of m or mb for megabytes.
- Specify a suffix of g or gb for gigabytes.
- If you specify a suffix with unrecognized formats, subsequent queries fail with an error.

Default: 0 (no limit)

Added in: Impala 3.1

SCHEDULE_RANDOM_REPLICA query option

The SCHEDULE_RANDOM_REPLICA query option fine-tunes the scheduling algorithm for deciding which host processes each HDFS data block or Kudu tablet to reduce the chance of CPU hotspots.

By default, Impala estimates how much work each host has done for the query, and selects the host that has the lowest workload. This algorithm is intended to reduce CPU hotspots arising when the same host is selected to process multiple data blocks / tablets. Use the SCHEDULE_RANDOM_REPLICA query option if hotspots still arise for some combinations of queries and data layout.

The SCHEDULE_RANDOM_REPLICA query option only applies to tables and partitions that are not enabled for the HDFS caching.

Type: BOOLEAN

Default: FALSE

SCRATCH_LIMIT query option

Specifies the maximum amount of disk storage, in bytes, that any Impala query can consume on any host using the “spill to disk” mechanism that handles queries that exceed the memory limit.

Specify the size in bytes, or with a trailing m or g character to indicate megabytes or gigabytes.

A value of zero turns off the spill to disk feature for queries in the current session, causing them to fail immediately if they exceed the memory limit.

The amount of memory used per host for a query is limited by the MEM_LIMIT query option.

The more DataNodes in the cluster, the less memory is used on each host, and therefore also less scratch space is required for queries that exceed the memory limit.

Type: Number, with optional unit specifier

Default: -1 (amount of spill space is unlimited)

SHUFFLE_DISTINCT_EXPRS query option

The SHUFFLE_DISTINCT_EXPRS query option controls the shuffling behavior when a query has both grouping and distinct expressions. Impala can optionally include the distinct expressions in the hash exchange to spread the data among more nodes. However, this plan requires one more hash exchange phase.

It is recommended that you turn off this option if the NDVs of the grouping expressions are high.

Type: BOOLEAN

Default: FALSE

SPOOL_QUERY_RESULTS query Option

Use the SPOOL_QUERY_RESULTS query option to enable query result spooling, which is disabled by default.

Query result spooling controls how rows are returned to the client.

- When query result spooling is disabled (SPOOL_QUERY_RESULTS = FALSE), Impala relies on clients to fetch results to trigger the generation of more result row batches until all the result rows have been produced. If a client issues a query without fetching all the results, the query fragments will continue to consume the resources until the query is cancelled and unregistered, potentially tying up resources and cause other queries to wait for extended period of time in admission control.
- When query result spooling is enabled (SPOOL_QUERY_RESULTS = TRUE), the result sets of queries are eagerly fetched and spooled, either in memory or on disk.

  Once all result rows have been fetched and stored in the spooling location, the resources are freed up. Incoming client fetches can get the data from the spooled results.

Type: INT

Default: FALSE

Added in: Impala 3.4

SPOOL_ALL_RESULTS_FOR _RETRIES

Leave this query option SPOOL_ALL_RESULTS_FOR _RETRIES at its default value “true” for queries that can be retried. This new query option only applies if the query options RETRY_FA ILED_QUERIES and SPOOL_QUERY_RESULTS are enabled.

Currently, to avoid incorrect results, query retry will be skipped if any results are returned to the client in the original query. This new query option is introduced, for queries that can be retried, to control spooling all query results before returning any to the client. If the query results cannot be contained in the allocated result-spooling space, results will be returned and query retry will be disabled on the query.

If you set SPOOL_ALL_RESULTS_FOR _RETRIES to false, it will fall back to the original behavior - the client can fetch results when they are ready. For queries that cannot be retried, this option does not impact.

Type: BOOLEAN

Default: TRUE

SUPPORT_START_OVER query option

Leave this setting at its default value. It is a read-only setting, tested by some client applications such as Hue.

If you accidentally change it through `impala-shell`, subsequent queries encounter errors until you undo the change by issuing UNSET support_start_over.

Type: BOOLEAN

Default: FALSE

SYNC_DDL query option

When enabled, causes any DDL operation such as CREATE TABLE or ALTER TABLE to return only when the changes have been propagated to all other Impala nodes in the cluster by the Impala catalog service. That way, if you issue a subsequent CONNECT statement in `impala-shell` to connect to a different node in the cluster, you can be sure that other node will already recognize any added or changed tables. (The catalog service automatically broadcasts the DDL changes to all nodes automatically, but without this option there could be a period of inconsistency if you quickly switched to another node, such as by issuing a subsequent query through a load-balancing proxy.)

Although INSERT is classified as a DML statement, when the SYNC_DDL option is enabled, INSE RT statements also delay their completion until all the underlying data and metadata changes are propagated to all Impala nodes. Internally, Impala inserts have similarities with DDL statements in traditional database systems, because they create metadata needed to track HDFS block locations for new files and they potentially add new partitions to partitioned tables.

Note: Because this option can introduce a delay after each write operation, if you are running a sequence of CREATE DATABASE, CREATE TABLE, ALTER TABLE, INSERT, and similar statements within a setup script, to minimize the overall delay you can enable the SYNC_DDL query option only near the end, before the final DDL statement.

Type: BOOLEAN

Default: FALSE

THREAD_RESERVATION_AGGREGATE_LIMIT query option

The THREAD_RESERVATION_AGGREGATE_LIMIT query option limits the number of reserved threads for a query across all nodes on which it is executing. The option is intended to prevent execution of complex queries that can consume excessive CPU or operating system resources on a cluster. Queries that have more threads than this threshold are rejected by Impala's admission controller before they start executing.

For example, an Impala administrator could set a default value of THREAD_RESERVATION_AGGREGATE_LIMIT=2000 for a resource pool on a 100 node where they expect only relatively simple queries with less than 20 threads per node to run. This will reject queries that require more than 2000 reserved threads across all nodes, for example a query with 21 fragments running on all 100 nodes of the cluster.

You can override the default value per-query or per-session, in the same way as other query options, if you do not want the default THREAD_RESERVATION_AGGREGATE_LIMIT value to apply to a specific query or session.

Type: Number

Default: 0 (no limit)

Added in: Impala 3.1

THREAD_RESERVATION_LIMIT query option

The THREAD_RESERVATION_LIMIT query option limits the number of reserved threads for a query on each node. The option is intended to prevent execution of complex queries that can consume excessive CPU or operating system resources on a single node. Queries that have more threads per node than this threshold are rejected by Impala's admission controller before they start executing. You can see the number of reserved threads for a query in its explain plan in the "Per-Host Resource Reservation" line.

For example, an Impala administrator could set a default value of THREAD_RESERVATION_LIMIT=100 for a resource pool where they expect only relatively simple queries to run. This will reject queries that require more than 100 reserved threads on a node, for example, queries with more than 100 fragments.

You can override the default value per-query or per-session, in the same way as other query options, if you do not want the default THREAD_RESERVATION_LIMIT value to apply to a specific query or session.

Note: The number of reserved threads on a node may be lower than the maximum value in the explain plan if not all fragments of that query are scheduled on every node.

Type: Number

Default: 3000

Added in: Impala 3.1

TIMEZONE query option

The TIMEZONE query option defines the timezone used for conversions between UTC and the local time. If not set, Impala uses the system time zone where the Coordinator Impalad runs. As

query options are not sent to the Coordinator immediately, the timezones are validated only when the query runs.

Impala takes the timezone into a consideration in the following cases:

- When calling the NOW() function
- When converting between Unix time and timestamp if the use_local_tz_for_unix_timestamp_conversions flag is TRUE
- When reading Parquet timestamps written by Hive if the convert_legacy_hive_parquet_utc_timestamps flag is TRUE

Syntax:

```
SET TIMEZONE=TIME ZONE
```

TIME ZONE can be a canonical code or a time zone name defined in IANA Time Zone Database. The value is case-sensitive.

Leading/trailing quotes (') and double quotes (") are stripped.

If *TIME ZONE* is an empty string, the time zone for the query is set to the default time zone of the Impalad Coordinator.

If *TIME ZONE* is NULL or a space character, Impala returns an error when the query is executed.

Type: STRING

Default: The system time zone where the Coordinator impalad runs

Examples:

```
SET TIMEZONE=UTC;
SET TIMEZONE="Europe/Budapest";
```

Added in: Impala 3.1

TOPN_BYTES_LIMIT query option

The TOPN_BYTES_LIMIT query option places a limit on the amount of estimated memory that Impala can process for top-N queries.

Top-N queries are the queries that include both ORDER BY and LIMIT clauses. Top-N queries don't spill to disk so have to keep all rows they process in memory, and those queries can cause out-of-memory issues when running with a large limit and an offset. If the Impala planner estimates that a top-N operator will process more bytes than the TOPN_BYTES_LIMIT value, it will replace the top-N operator with the sort operator. Switching to the sort operator allows Impala to spill to disk, thus requiring less memory than top-N, but potentially with performance penalties.

The option has no effect when set to 0 or -1.

Syntax:

```
SET TOPN_BYTES_LIMIT=LIMIT
```

Type: Number

Default: 536870912 (512 MB)

Added in: Impala 3.1

UTF8_MODE query option

UTF-8 support allows string functions to recognize the UTF-8 characters, thus processing strings in a compatible way as other engines.

You can use the new query option, UTF8_MODE, to turn on/off the UTF-8 aware behavior. The query option can be set globally, or at per session level. Only queries with UTF8_MODE=true will have UTF-8 aware behaviors. If the query option UTF8_MODE is turned on globally, existing queries that depend on the original binary behavior need to be explicitly set to UTF8_MODE=false.

Type:BOOLEAN

Default:FALSE

Added in:Impala 4.1

Related Information

Query Join Performance
Compute Stats Statement
Impala Aggregate Functions

Virtual column

You can include a virtual column in a standard SELECT statement, select INPUT__FILE__NAME from <tablename>, to retrieve the data file name that stores the actual row in a table.

List of use cases

You can use this virtual column INPUT__FILE__NAME in multiple ways.

- To view the file names corresponding to each record.
- To extract the metadata using virtual columns if external source files ingested to HDFS contain metadata like a timestamp or source system identifier, and if you prefer to use it in subsequent operations.
- To extract the timestamp of the data derived from the fully qualified file name.
- To forward the technical timeline in a data warehouse scenario (for example, the "LoadDateTS" in a hybrid Data Vault architecture).

Supported operations

You can perform ceratin operations on this virtual column. The following list contains some of the supported operations:

- Invoke “Functions”
- Filter rows
- "GROUP BY" operation

Example

This example provides the file name corresponding to each record.

```
select INPUT__FILE__NAME, id, name from users where dt='20140210' group by
          INPUT__FILE__NAME;
```

Result

The above query returns results similar to the following

```
hdfs://localhost.localdomain:8020/user/hive/warehouse/users/users1.txt 2 use
r2
```

```
hdfs://localhost.localdomain:8020/user/hive/warehouse/users/users2.txt 42 jo
hn.doe
```

You can use the supported string functions to extract information from the URL.

Impala built-in functions

Impala supports several categories of built-in functions. These functions let you perform mathematical calculations, string manipulation, date calculations, and other kinds of data transformations directly in SQL statements.

Supported functions

The categories of built-in functions supported by Impala are:

- Impala mathematical functions on page 308
- Impala type conversion functions on page 336
- Impala date and time functions on page 344
- Impala conditional functions on page 363
- Impala string functions on page 368
- Impala aggregate functions on page 388.
- Impala analytic functions on page 421
- Impala bit functions on page 326
- Impala miscellaneous functions on page 385

The following is a complete list of built-in functions supported in Impala:

ABS
ACOS
ADD_MONTHS
ADDDATE
APPX_MEDIAN
ASCII
ASIN
ATAN
ATAN2
AVG
AVG - Analytic Function
BASE64DECODE
BASE64ENCODE
BITAND
BIN
BITNOT
BITOR
BITXOR
BTRIM
CASE
CASE WHEN
CAST
CEIL, CEILING, DCEIL
CHAR_LENGTH

CHR
COALESCE
CONCAT
CONCAT_WS
CONV
COS
COSH
COT
COUNT
COUNT - Analytic Function
COUNTSET
CUME_DIST
CURRENT_DATABASE
CURRENT_TIMESTAMP
DATE_ADD
DATE_PART
DATE_SUB
DATE_TRUNC
DATEDIFF
DAY
DAYNAME
DAYOFWEEK
DAYOFYEAR
DAYS_ADD
DAYS_SUB
DECODE
DEGREES
DENSE_RANK
E
EFFECTIVE_USER
EXP
EXTRACT
FACTORIAL
FIND_IN_SET
FIRST_VALUE
FLOOR, DFLOOR
FMOD
FNV_HASH
GET_JSON_OBJECT
FROM_UNIXTIME

FROM_TIMESTAMP
FROM_UTC_TIMESTAMP
GETBIT
GREATEST
GROUP_CONCAT
GROUP_CONCAT - Analytic Function
HEX
HOUR
HOURS_ADD
HOURS_SUB
IF
IFNULL
INITCAP
INSTR
INT_MONTHS_BETWEEN
IS_INF
IS_NAN
ISFALSE
ISNOTFALSE
ISNOTTRUE
ISNULL
ISTRUE
JARO_DISTANCE, JARO_DIST
JARO_SIMILARITY, JARO_SIM
JARO_WINKER_DISTANCE, JW_DST
JARO_WINKER_SIMILARITY, JW_SIM
LAG
LAST_VALUE
LEAD
LEAST
LEFT
LENGTH
LN
LOCATE
LOG
LOG10
LOG2
LOWER, LCASE
LPAD
LTRIM

MAX
MAX - Analytic Function
MAX_INT, MAX_TINYINT, MAX_SMALLINT, MAX_BIGINT
MICROSECONDS_ADD
MICROSECONDS_SUB
MILLISECOND
MILLISECONDS_ADD
MILLISECONDS_SUB
MIN
MIN - Analytic Function
MIN_INT, MIN_TINYINT, MIN_SMALLINT, MIN_BIGINT
MINUTE
MINUTES_ADD
MINUTES_SUB
MOD
MONTH
MONTHNAME
MONTHS_ADD
MONTHS_BETWEEN
MONTHS_SUB
MURMUR_HASH
NANOSECONDS_ADD
NANOSECONDS_SUB
NDV
NEGATIVE
NEXT_DAY
NONNULLVALUE
NOW
NTILE
NULLIF
NULLIFZERO
NULLVALUE
NVL
NVL2
OVER Clause
PARSE_URL
PERCENT_RANK
PI
PID
PMOD

POSITIVE
POW, POWER, DPOW, FPOW
PRECISION
QUARTER
QUOTIENT
RADIANS
RAND, RANDOM
RANK
REGEXP_ESCAPE
REGEXP_EXTRACT
REGEXP_LIKE
REGEXP_REPLACE
REPEAT
REPLACE
REVERSE
RIGHT
ROTATELEFT
ROTATERIGHT
ROUND, DROUND
ROW_NUMBER
RPAD
RTRIM
SCALE
SECOND
SECONDS_ADD
SECONDS_SUB
SETBIT
SHIFTLEFT
SHIFTRIGHT
SIGN
SIN
SINH
SLEEP
SPACE
SPLIT_PART
SQRT
STDDEV, STDDEV_SAMP, STDDEV_POP
STRLEFT
STRRIGHT
SUBDATE

SUBSTR, SUBSTRING
SUM
SUM - Analytic Function
TAN
TANH
TIMEOFDAY
TIMESTAMP_CMP
TO_DATE
TO_TIMESTAMP
TO_UTC_TIMESTAMP
TRANSLATE
TRIM
TRUNC
TRUNCATE, DTRUNC, TRUNC
TYPEOF
UNHEX
UNIX_TIMESTAMP
UPPER, UCASE
USER
UTC_TIMESTAMP
UUID
VARIANCE, VARIANCE_SAMP, VARIANCE_POP, VAR_SAMP, VAR_POP
VERSION
WEEKOFYEAR
WEEKS_ADD
WEEKS_SUB
WIDTH_BUCKET
YEAR
YEARS_ADD
YEARS_SUB
ZEROIFNULL

Impala mathematical functions

Mathematical functions, or arithmetic functions, perform numeric calculations that are typically more complex than basic addition, subtraction, multiplication, and division. For example, these functions include trigonometric, logarithmic, and base conversion operations.

Note: In Impala, exponentiation uses the POW() function rather than an exponentiation operator such as **.

Related information:

The mathematical functions operate mainly on these data types:

- INT
- BIGINT
- SMALLINT
- TINYINT
- DOUBLE
- FLOAT
- DECIMAL

For the operators that perform the standard operations such as addition, subtraction, multiplication, and division, see Arithmetic operators on page 93.

Functions that perform bitwise operations are explained in Impala bit functions on page 326.

Function reference:

Impala supports the following mathematical functions:

- ABS
- ACOS
- ASIN
- ATAN
- ATAN2
- BIN
- CEIL, CEILING, DCEIL
- CONV
- COS
- COSH
- COT
- DEGREES
- E
- EXP
- FACTORIAL
- FLOOR, DFLOOR
- FMOD
- FNV_HASH
- GREATEST
- HEX
- IS_INF
- IS_NAN
- LEAST
- LN
- LOG
- LOG10
- LOG2
- MAX_INT, MAX_TINYINT, MAX_SMALLINT, MAX_BIGINT
- MIN_INT, MIN_TINYINT, MIN_SMALLINT, MIN_BIGINT
- MOD
- MURMUR_HASH
- NEGATIVE
- PI
- PMOD
- POSITIVE
- POW, POWER, DPOW, FPOW

- PRECISION
- QUOTIENT
- RADIANS
- RAND, RANDOM
- ROUND, DROUND
- SCALE
- SIGN
- SIN
- SINH
- SQRT
- TAN
- TANH
- TRUNCATE, DTRUNC, TRUNC
- UNHEX
- WIDTH_BUCKET

ABS(numeric_type a)

Purpose: Returns the absolute value of the argument.

Return type: Same as the input value

Usage notes: Use this function to ensure all return values are positive. This is different than the positive() function, which returns its argument unchanged (even if the argument was negative).

ACOS(DOUBLE a)

Purpose: Returns the arccosine of the argument.

Return type: DOUBLE

ASIN(DOUBLE a)

Purpose: Returns the arcsine of the argument.

Return type: DOUBLE

ATAN(DOUBLE a)

Purpose: Returns the arctangent of the argument.

Return type: DOUBLE

ATAN(DOUBLE a, DOUBLE b)

Purpose: Returns the arctangent of the two arguments, with the signs of the arguments used to determine the quadrant of the result.

Return type: DOUBLE

BIN(BIGINT a)

Purpose: Returns the binary representation of an integer value, that is, a string of 0 and 1 digits.

Return type: STRING

CEIL(DOUBLE a), CEIL(DECIMAL(p,s) a), CEILING(DOUBLE a), CEILING(DECIMAL(p,s) a), DCEIL(DOUBLE a), DCEIL(DECIMAL(p,s) a)

Purpose: Returns the smallest integer that is greater than or equal to the argument.

Return type: Same as the input type

CONV(BIGINT n, INT from_base, INT to_base), CONV(STRING s, INT from_base, INT to_base)

Purpose: Returns a string representation of the first argument converted from from_base to to_base. The first argument can be specified as a number or a string. For example, conv(100, 2, 10) and conv('100', 2, 10) both return '4'.

Return type: STRING

Usage notes:

If to_base is negative, the first argument is treated as signed, and otherwise, it is treated as unsigned. For example:

- conv(-17, 10, -2) returns '-10001', -17 in base 2.
- conv(-17, 10, 10) returns '18446744073709551599'. -17 is interpreted as an unsigned, 2^64-17, and then the value is returned in base 10.

The function returns NULL when the following invalid arguments are specified:

- Any argument is NULL.
- from_base or to_base is below -36 or above 36.
- from_base or to_base is -1, 0, or 1.
- The first argument represents a positive number and from_base is a negative number.

If the first argument represents a negative number and from_base is a negative number, the function returns 0.

If the first argument represents a number larger than the maximum bigint, the function returns:

- The string representation of -1 in to_base if to_base is negative.
- The string representation of 18446744073709551615' (2^64 - 1) in to_base if to_base is positive.

If the first argument does not represent a valid number in from_base, e.g. 3 in base 2 or '1a23' in base 10, the digits in the first argument are evaluated from left-to-right and used if a valid digit in from_base. The invalid digit and the digits to the right are ignored.

For example:

- conv(445, 5, 10) is converted to conv(44, 5, 10) and returns '24'.
- conv('1a23', 10, 16) is converted to conv('1', 10 , 16) and returns '1'.

COS(DOUBLE a)

Purpose: Returns the cosine of the argument.

Return type: DOUBLE

COSH(DOUBLE a)

Purpose: Returns the hyperbolic cosine of the argument.

Return type: DOUBLE

COT(DOUBLE a)

Purpose: Returns the cotangent of the argument.

Return type: DOUBLE

Added in: Impala 2.3.0

DEGREES(DOUBLE a)

Purpose: Converts argument value from radians to degrees.

Return type: DOUBLE

E()

Purpose: Returns the mathematical constant e.

Return type: DOUBLE

EXP(DOUBLE a), DEXP(DOUBLE a)

Purpose: Returns the mathematical constant e raised to the power of the argument.

Return type: DOUBLE

FACTORIAL(integer_type a)

Purpose: Computes the factorial of an integer value. It works with any integer type.

Added in: Impala 2.3.0

Usage notes: You can use either the factorial() function or the ! operator. The factorial of 0 is 1. Likewise, the factorial() function returns 1 for any negative value. The maximum positive value for the input argument is 20; a value of 21 or greater overflows the range for a BIGINT and causes an error.

Return type: BIGINT

Added in: Impala 2.3.0

```
select factorial(5);
+--------------+
| factorial(5) |
+--------------+
| 120          |
+--------------+

select 5!;
+-----+
| 5!  |
+-----+
| 120 |
+-----+
```

FLOOR(DOUBLE a), FLOOR(DECIMAL(p,s) a), DFLOOR(DOUBLE a), DFLOOR(DECIMAL(p,s) a)

Purpose: Returns the largest integer that is less than or equal to the argument.

Return type: Same as the input type

FMOD(DOUBLE a, DOUBLE b), FMOD(FLOAT a, FLOAT b)

Purpose: Returns the modulus of a floating-point number.

Return type: FLOAT or DOUBLE, depending on type of arguments

Added in: Impala 1.1.1

Usage notes:

Because this function operates on DOUBLE or FLOAT values, it is subject to potential rounding errors for values that cannot be represented precisely. Prefer to use whole numbers, or values that you know can be represented precisely by the DOUBLE or FLOAT types.

Examples:

The following examples show equivalent operations with the fmod() function and the % arithmetic operator, for values not subject to any rounding error.

```
select fmod(10,3);
+-------------+
| fmod(10, 3) |
+-------------+
| 1           |
+-------------+

select fmod(5.5,2);
+--------------+
| fmod(5.5, 2) |
+--------------+
| 1.5          |
+--------------+
```

```
select 10 % 3;
+--------+
| 10 % 3 |
+--------+
| 1      |
+--------+
select 5.5 % 2;
+---------+
| 5.5 % 2 |
+---------+
| 1.5     |
+---------+
```

The following examples show operations with the fmod() function for values that cannot be represented precisely by the DOUBLE or FLOAT types, and thus are subject to rounding error. fmod(9.9,3.0) returns a value slightly different than the expected 0.9 because of rounding. fmod(9.9 ,3.3) returns a value quite different from the expected value of 0 because of rounding error during intermediate calculations.

```
select fmod(9.9,3.0);
+--------------------+
| fmod(9.9, 3.0)     |
+--------------------+
| 0.8999996185302734 |
+--------------------+

select fmod(9.9,3.3);
+-------------------+
| fmod(9.9, 3.3)    |
+-------------------+
| 3.299999713897705 |
+-------------------+
```

FNV_HASH(type v)

Purpose: Returns a consistent 64-bit value derived from the input argument, for convenience of implementing hashing logic in an application.

Return type: BIGINT

Usage notes:

You might use the return value in an application where you perform load balancing, bucketing, or some other technique to divide processing or storage.

Because the result can be any 64-bit value, to restrict the value to a particular range, you can use an expression that includes the ABS() function and the % (modulo) operator. For example, to produce a hash value in the range 0-9, you could use the expression ABS(FNV_HASH(x)) % 10.

This function implements the same algorithm that Impala uses internally for hashing, on systems where the CRC32 instructions are not available.

This function implements the Fowler–Noll–Vo hash function, in particular the FNV-1a variation. This is not a perfect hash function: some combinations of values could produce the same result value. It is not suitable for cryptographic use.

Similar input values of different types could produce different hash values, for example the same numeric value represented as SMALLINT or BIGINT, FLOAT or DOUBLE, or DECIMAL(5,2) or DECIMAL(20,5).

Examples:

```
[localhost:21000] > create table h (x int, s string);
```

```
[localhost:21000] > insert into h values (0, 'hello'), (1,'wor
ld'), (1234567890,'antidisestablishmentarianism');
[localhost:21000] > select x, fnv_hash(x) from h;
+------------+---------------------+
| x          | fnv_hash(x)         |
+------------+---------------------+
| 0          | -2611523532599129963 |
| 1          | 4307505193096137732  |
| 1234567890 | 3614724209955230832  |
+------------+---------------------+
[localhost:21000] > select s, fnv_hash(s) from h;
+-------------------------------+---------------------+
| s                             | fnv_hash(s)         |
+-------------------------------+---------------------+
| hello                         | 6414202926103426347 |
| world                         | 6535280128821139475 |
| antidisestablishmentarianism | -209330013948433970 |
+-------------------------------+---------------------+
[localhost:21000] > select s, abs(fnv_hash(s)) % 10 from h;
+-------------------------------+-------------------------+
| s                             | abs(fnv_hash(s)) % 10.0 |
+-------------------------------+-------------------------+
| hello                         | 8                       |
| world                         | 6                       |
| antidisestablishmentarianism | 4                       |
+-------------------------------+-------------------------+
```

For short argument values, the high-order bits of the result have relatively low entropy:

```
[localhost:21000] > create table b (x boolean);
[localhost:21000] > insert into b values (true), (true), (false),
 (false);
[localhost:21000] > select x, fnv_hash(x) from b;
+-------+---------------------+
| x     | fnv_hash(x)         |
+-------+---------------------+
| true  | 2062020650953872396 |
| true  | 2062020650953872396 |
| false | 2062021750465500607 |
| false | 2062021750465500607 |
+-------+---------------------+
```

Added in: Impala 1.2.2

GREATEST(BIGINT a[, BIGINT b ...]), GREATEST(DOUBLE a[, DOUBLE b ...]), GREATEST(DECIMAL(p,s) a[, DECIMAL(p,s) b ...]), GREATEST(STRING a[, STRING b ...]), GREATEST(TIMESTAMP a[, TIMESTAMP b ...])

Purpose: Returns the largest value from a list of expressions.

Return type: same as the initial argument value, except that integer values are promoted to BIGINT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

HEX(BIGINT a), HEX(STRING a)

Purpose: Returns the hexadecimal representation of an integer value, or of the characters in a string.

Return type: STRING

IS_INF(DOUBLE a)

Purpose: Tests whether a value is equal to the special value “inf”, signifying infinity.

Return type: BOOLEAN

Usage notes:

Infinity and NaN can be specified in text data files as inf and nan respectively, and Impala interprets them as these special values. They can also be produced by certain arithmetic expressions; for example, 1/0 returns Infinity and pow(-1, 0.5) returns NaN. Or you can cast the literal values, such as CAST('nan' AS DOUBLE) or CAST('inf' AS DOUBLE).

IS_NAN(DOUBLE a)

Purpose: Tests whether a value is equal to the special value “NaN”, signifying “not a number”.

Return type: BOOLEAN

Usage notes:

Infinity and NaN can be specified in text data files as inf and nan respectively, and Impala interprets them as these special values. They can also be produced by certain arithmetic expressions; for example, 1/0 returns Infinity and pow(-1, 0.5) returns NaN. Or you can cast the literal values, such as CAST('nan' AS DOUBLE) or CAST('inf' AS DOUBLE).

LEAST(BIGINT a[, BIGINT b ...]), LEAST(DOUBLE a[, DOUBLE b ...]), LEAST(DECIMAL(p,s) a[, DECIMAL(p,s) b ...]), LEAST(STRING a[, STRING b ...]), LEAST(TIMESTAMP a[, TIMESTAMP b ...])

Purpose: Returns the smallest value from a list of expressions.

Return type: same as the initial argument value, except that integer values are promoted to BIGINT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

LN(DOUBLE a), DLOG1(DOUBLE a)

Purpose: Returns the natural logarithm of the argument.

Return type: DOUBLE

LOG(DOUBLE base, DOUBLE a)

Purpose: Returns the logarithm of the second argument to the specified base.

Return type: DOUBLE

LOG10(DOUBLE a), DLOG10(DOUBLE a)

Purpose: Returns the logarithm of the argument to the base 10.

Return type: DOUBLE

LOG2(DOUBLE a)

Purpose: Returns the logarithm of the argument to the base 2.

Return type: DOUBLE

MAX_INT(), MAX_TINYINT(), MAX_SMALLINT(), MAX_BIGINT()

Purpose: Returns the largest value of the associated integral type.

Return type: The same as the integral type being checked.

Usage notes: Use the corresponding min_ and max_ functions to check if all values in a column are within the allowed range, before copying data or altering column definitions. If not, switch to the next higher integral type or to a DECIMAL with sufficient precision.

MIN_INT(), MIN_TINYINT(), MIN_SMALLINT(), MIN_BIGINT()

Purpose: Returns the smallest value of the associated integral type (a negative number).

Return type: The same as the integral type being checked.

Usage notes: Use the corresponding min_ and max_ functions to check if all values in a column are within the allowed range, before copying data or altering column definitions. If not, switch to the next higher integral type or to a DECIMAL with sufficient precision.

MOD(numeric_type a, same_type b)

Purpose: Returns the modulus of a number. Equivalent to the % arithmetic operator. Works with any size integer type, any size floating-point type, and DECIMAL with any precision and scale.

Return type: Same as the input value

Added in: Impala 2.20

Usage notes:

Because this function works with DECIMAL values, prefer it over fmod() when working with fractional values. It is not subject to the rounding errors that make fmod() problematic with floating-point numbers.

Query plans shows the MOD() function as the % operator.

Examples:

The following examples show how the mod() function works for whole numbers and fractional values, and how the % operator works the same way. In the case of mod(9.9,3), the type conversion for the second argument results in the first argument being interpreted as DOUBLE, so to produce an accurate DECIMAL result requires casting the second argument or writing it as a DECIMAL literal, 3.0.

```
select mod(10,3);
+-------------+
| mod(10, 3) |
+-------------+
| 1           |
+-------------+

select mod(5.5,2);
+--------------+
| mod(5.5, 2) |
+--------------+
| 1.5          |
+--------------+

select 10 % 3;
+--------+
| 10 % 3 |
+--------+
| 1      |
+--------+
select 5.5 % 2;
+---------+
| 5.5 % 2 |
+---------+
| 1.5     |
+---------+

select mod(9.9,3.3);
+---------------+
| mod(9.9, 3.3) |
+---------------+
| 0.0           |
+---------------+

select mod(9.9,3);
+--------------------+
| mod(9.9, 3)        |
+--------------------+
| 0.8999996185302734 |
+--------------------+
```

```
select mod(9.9, cast(3 as decimal(2,1)));
+-----------------------------------+
| mod(9.9, cast(3 as decimal(2,1))) |
+-----------------------------------+
| 0.9                               |
+-----------------------------------+

select mod(9.9,3.0);
+---------------+
| mod(9.9, 3.0) |
+---------------+
| 0.9           |
+---------------+
```

MURMUR_HASH(type v)

Purpose: Returns a consistent 64-bit value derived from the input argument, for convenience of implementing MurmurHash2 non-cryptographic hash function.

Return type: BIGINT

Usage notes:

You might use the return value in an application where you perform load balancing, bucketing, or some other technique to divide processing or storage. This function provides a good performance for all kinds of keys such as number, ascii string and UTF-8. It can be recommended as general-purpose hashing function.

Regarding comparison of murmur_hash with fnv_hash, murmur_hash is based on Murmur2 hash algorithm and fnv_hash function is based on FNV-1a hash algorithm. Murmur2 and FNV-1a can show very good randomness and performance compared with well known other hash algorithms, but Murmur2 slightly show better randomness and performance than FNV-1a.

Similar input values of different types could produce different hash values, for example the same numeric value represented as SMALLINT or BIGINT, FLOAT or DOUBLE, or DECIMAL(5,2) or DECIMAL(20,5).

Examples:

```
[localhost:21000] > create table h (x int, s string);
[localhost:21000] > insert into h values (0, 'hello'), (1,'wor
ld'), (1234567890,'antidisestablishmentarianism');
[localhost:21000] > select x, murmur_hash(x) from h;
+------------+----------------------+
| x          | murmur_hash(x)       |
+------------+----------------------+
| 0          | 6960269033020761575  |
| 1          | -7806115816811537 83 |
| 1234567890 | -5754914572385924334 |
+------------+----------------------+
[localhost:21000] > select s, murmur_hash(s) from h;
+-------------------------------+----------------------+
| s                             | murmur_hash(s)       |
+-------------------------------+----------------------+
| hello                         | 2191231550387646743  |
| world                         | 5568329560871645431  |
| antidisestablishmentarianism | -2261804666958489663 |
+-------------------------------+----------------------+
```

For short argument values, the high-order bits of the result have relatively higher entropy than fnv_hash:

```
[localhost:21000] > create table b (x boolean);
```

```
[localhost:21000] > insert into b values (true), (true), (false),
 (false);
[localhost:21000] > select x, murmur_hash(x) from b;
+-------+----------------------+
| x     | murmur_hash(x)       |
+-------+----------------------++
| true  | -5720937396023583481 |
| true  | -5720937396023583481 |
| false | 6351753276682545529  |
| false | 6351753276682545529  |
+-------+---------------------+-+
```

Added in: Impala 2.12.0

NEGATIVE(numeric_type a)

Purpose: Returns the argument with the sign reversed; returns a positive value if the argument was already negative.

Return type: Same as the input value

Usage notes: Use -abs(a) instead if you need to ensure all return values are negative.

PI()

Purpose: Returns the constant pi.

Return type: double

PMOD(BIGINT a, BIGINT b), PMOD(DOUBLE a, DOUBLE b)

Purpose: Returns the positive modulus of a number. Primarily for Hive SQL compatibility.

Return type: INT or DOUBLE, depending on type of arguments

Examples:

The following examples show how the fmod() function sometimes returns a negative value depending on the sign of its arguments, and the pmod() function returns the same value as fmod(), but sometimes with the sign flipped.

```
select fmod(-5,2);
+-------------+
| fmod(-5, 2) |
+-------------+
| -1          |
+-------------+

select pmod(-5,2);
+-------------+
| pmod(-5, 2) |
+-------------+
| 1           |
+-------------+

select fmod(-5,-2);
+--------------+
| fmod(-5, -2) |
+--------------+
| -1           |
+--------------+

select pmod(-5,-2);
+--------------+
| pmod(-5, -2) |
+--------------+
| -1           |
```

```
+--------------+

select fmod(5,-2);
+-------------+
| fmod(5, -2) |
+-------------+
| 1           |
+-------------+

select pmod(5,-2);
+-------------+
| pmod(5, -2) |
+-------------+
| -1          |
+-------------+
```

POSITIVE(numeric_type a)

Purpose: Returns the original argument unchanged (even if the argument is negative).

Return type: Same as the input value

Usage notes: Use abs() instead if you need to ensure all return values are positive.

POW(DOUBLE a, double p), POWER(DOUBLE a, DOUBLE p), DPOW(DOUBLE a, DOUBLE p), FPOW(DOUBLE a, DOUBLE p)

Purpose: Returns the first argument raised to the power of the second argument.

Return type: DOUBLE

PRECISION(numeric_expression)

Purpose: Computes the precision (number of decimal digits) needed to represent the type of the argument expression as a DECIMAL value.

Usage notes:

Typically used in combination with the scale() function, to determine the appropriate DECI MAL(*PRECISION*,*SCALE*) type to declare in a CREATE TABLE statement or CAST() function.

Return type: INT

Examples:

The following examples demonstrate how to check the precision and scale of numeric literals or other numeric expressions. Impala represents numeric literals in the smallest appropriate type. 5 is a TINYINT value, which ranges from -128 to 127, therefore 3 decimal digits are needed to represent the entire range, and because it is an integer value there are no fractional digits. 1.333 is interpreted as a DECIMAL value, with 4 digits total and 3 digits after the decimal point.

```
[localhost:21000] > select precision(5), scale(5);
+--------------+----------+
| precision(5) | scale(5) |
+--------------+----------+
| 3            | 0        |
+--------------+----------+
[localhost:21000] > select precision(1.333), scale(1.333);
+------------------+--------------+
| precision(1.333) | scale(1.333) |
+------------------+--------------+
| 4                | 3            |
+------------------+--------------+
[localhost:21000] > with t1 as
  ( select cast(12.34 as decimal(20,2)) x union select cast(1 as
 decimal(8,6)) x )
  select precision(x), scale(x) from t1 limit 1;
```

```
+--------------+----------+
| precision(x) | scale(x) |
+--------------+----------+
| 24           | 6        |
+--------------+----------+
```

QUOTIENT(BIGINT numerator, BIGINT denominator), QUOTIENT(DOUBLE numerator, DOUBLE denominator)

Purpose: Returns the first argument divided by the second argument, discarding any fractional part. Avoids promoting integer arguments to DOUBLE as happens with the / SQL operator. Also includes an overload that accepts DOUBLE arguments, discards the fractional part of each argument value before dividing, and again returns BIGINT. With integer arguments, this function works the same as the DIV operator.

Return type: BIGINT

RADIANS(DOUBLE a)

Purpose: Converts argument value from degrees to radians.

Return type: DOUBLE

RAND(), RAND(BIGINT seed), RANDOM(), RANDOM(BIGINT seed)

Purpose: Returns a random value between 0 and 1. After rand() is called with a seed argument, it produces a consistent random sequence based on the seed value.

Return type: DOUBLE

Usage notes: Currently, the random sequence is reset after each query, and multiple calls to rand() within the same query return the same value each time. For different number sequences that are different for each query, pass a unique seed value to each call to rand(). For example, select rand(unix_timestamp()) from ...

Examples:

The following examples show how rand() can produce sequences of varying predictability, so that you can reproduce query results involving random values or generate unique sequences of random values for each query. When rand() is called with no argument, it generates the same sequence of values each time, regardless of the ordering of the result set. When rand() is called with a constant integer, it generates a different sequence of values, but still always the same sequence for the same seed value. If you pass in a seed value that changes, such as the return value of the expression unix_timestamp(now()), each query will use a different sequence of random values, potentially more useful in probability calculations although more difficult to reproduce at a later time. Therefore, the final two examples with an unpredictable seed value also include the seed in the result set, to make it possible to reproduce the same random sequence later.

```
select x, rand() from three_rows;
+---+-----------------------+
| x | rand()                |
+---+-----------------------+
| 1 | 0.0004714746030380365 |
| 2 | 0.5895895192351144    |
| 3 | 0.4431900859080209    |
+---+-----------------------+
select x, rand() from three_rows order by x desc;
+---+-----------------------+
| x | rand()                |
+---+-----------------------+
| 3 | 0.0004714746030380365 |
| 2 | 0.5895895192351144    |
| 1 | 0.4431900859080209    |
+---+-----------------------+
```

```
select x, rand(1234) from three_rows order by x;
+---+----------------------+
| x | rand(1234)           |
+---+----------------------+
| 1 | 0.7377511392057646   |
| 2 | 0.009428468537250751 |
| 3 | 0.208117277924026    |
+---+----------------------+

select x, rand(1234) from three_rows order by x desc;
+---+----------------------+
| x | rand(1234)           |
+---+----------------------+
| 3 | 0.7377511392057646   |
| 2 | 0.009428468537250751 |
| 1 | 0.208117277924026    |
+---+----------------------+
select x, unix_timestamp(now()), rand(unix_timestamp(now()))
  from three_rows order by x;
+---+-----------------------+-----------------------------+
| x | unix_timestamp(now()) | rand(unix_timestamp(now())) |
+---+-----------------------+-----------------------------+
| 1 | 1440777752            | 0.002051228658320023        |
| 2 | 1440777752            | 0.5098743483004506          |
| 3 | 1440777752            | 0.9517714925817081          |
+---+-----------------------+-----------------------------+
select x, unix_timestamp(now()), rand(unix_timestamp(now()))
  from three_rows order by x desc;
+---+-----------------------+-----------------------------+
| x | unix_timestamp(now()) | rand(unix_timestamp(now())) |
+---+-----------------------+-----------------------------+
| 3 | 1440777761            | 0.9985985015512437          |
| 2 | 1440777761            | 0.3251255333074953          |
| 1 | 1440777761            | 0.02422675025846192         |
+---+-----------------------+-----------------------------+
```

ROUND(DOUBLE a), ROUND(DOUBLE a, INT d), ROUND(DECIMAL a, int_type d), DROUND(DOUBLE a), DROUND(DOUBLE a, INT d), DROUND(DECIMAL(p,s) a, int_type d)

Purpose: Rounds a floating-point value. By default (with a single argument), rounds to the nearest integer. Values ending in .5 are rounded up for positive numbers, down for negative numbers (that is, away from zero). The optional second argument specifies how many digits to leave after the decimal point; values greater than zero produce a floating-point return value rounded to the requested number of digits to the right of the decimal point.

Return type: Same as the input type

SCALE(numeric_expression)

Purpose: Computes the scale (number of decimal digits to the right of the decimal point) needed to represent the type of the argument expression as a DECIMAL value.

Usage notes:

Typically used in combination with the precision() function, to determine the appropriate DECIMAL(*PRECISION*,*SCALE*) type to declare in a CREATE TABLE statement or CAST() function.

Return type: int

Examples:

The following examples demonstrate how to check the precision and scale of numeric literals or other numeric expressions. Impala represents numeric literals in the smallest appropriate type. 5 is a TINYINT value, which ranges from -128 to 127, therefore 3 decimal digits are needed to represent

the entire range, and because it is an integer value there are no fractional digits. 1.333 is interpreted as a DECIMAL value, with 4 digits total and 3 digits after the decimal point.

```
[localhost:21000] > select precision(5), scale(5);
+--------------+----------+
| precision(5) | scale(5) |
+--------------+----------+
| 3            | 0        |
+--------------+----------+
[localhost:21000] > select precision(1.333), scale(1.333);
+------------------+--------------+
| precision(1.333) | scale(1.333) |
+------------------+--------------+
| 4                | 3            |
+------------------+--------------+
[localhost:21000] > with t1 as
  ( select cast(12.34 as decimal(20,2)) x union select cast(1 as
 decimal(8,6)) x )
  select precision(x), scale(x) from t1 limit 1;
+--------------+----------+
| precision(x) | scale(x) |
+--------------+----------+
| 24           | 6        |
+--------------+----------+
```

SIGN(DOUBLE a)

Purpose: Returns -1, 0, or 1 to indicate the signedness of the argument value.

Return type: INT

SIN(DOUBLE a)

Purpose: Returns the sine of the argument.

Return type: DOUBLE

SINH(DOUBLE a)

Purpose: Returns the hyperbolic sine of the argument.

Return type: DOUBLE

SQRT(DOUBLE a), DSQRT(DOUBLE a)

Purpose: Returns the square root of the argument.

Return type: DOUBLE

TAN(DOUBLE a)

Purpose: Returns the tangent of the argument.

Return type: DOUBLE

TANH(DOUBLE a)

Purpose: Returns the hyperbolic tangent of the argument.

Return type: DOUBLE

TRUNCATE(DOUBLE_or_DECIMAL a[, digits_to_leave]), DTRUNC(DOUBLE_or_DECIMAL a[, digits_to_leave]), TRUNC(DOUBLE_or_DECIMAL a[, digits_to_leave])

Purpose: Removes some or all fractional digits from a numeric value.

Arguments: With a single floating-point argument, removes all fractional digits, leaving an integer value. The optional second argument specifies the number of fractional digits to include in the return value, and only applies when the argument type is DECIMAL. A second argument of 0

truncates to a whole integer value. A second argument of negative N sets N digits to 0 on the left side of the decimal

Scale argument: The scale argument applies only when truncating DECIMAL values. It is an integer specifying how many significant digits to leave to the right of the decimal point. A scale argument of 0 truncates to a whole integer value. A scale argument of negative N sets N digits to 0 on the left side of the decimal point.

TRUNCATE(), DTRUNC(), and TRUNC() are aliases for the same function.

Return type: Same as the input type

Added in: The TRUNC() alias was added in Impala 2.10.0.

Usage notes:

You can also pass a DOUBLE argument, or DECIMAL argument with optional scale, to the DTRUNC() or TRUNCATE functions. Using the TRUNC() function for numeric values is common with other industry-standard database systems, so you might find such TRUNC() calls in code that you are porting to Impala.

The TRUNC() function also has a signature that applies to TIMESTAMP values.

Examples:

The following examples demonstrate the TRUNCATE() and DTRUNC() signatures for this function:

```
select truncate(3.45);
+----------------+
| truncate(3.45) |
+----------------+
| 3              |
+----------------+

select truncate(-3.45);
+-----------------+
| truncate(-3.45) |
+-----------------+
| -3              |
+-----------------+

select truncate(3.456,1);
+--------------------+
| truncate(3.456, 1) |
+--------------------+
| 3.4                |
+--------------------+

select dtrunc(3.456,1);
+------------------+
| dtrunc(3.456, 1) |
+------------------+
| 3.4              |
+------------------+

select truncate(3.456,2);
+--------------------+
| truncate(3.456, 2) |
+--------------------+
| 3.45               |
+--------------------+

select truncate(3.456,7);
+--------------------+
```

```
| truncate(3.456, 7) |
+--------------------+
| 3.4560000          |
+--------------------+
```

The following examples demonstrate using TRUNC() with DECIMAL or DOUBLE values, and with an optional scale argument for DECIMAL values. (The behavior is the same for the TRUNCATE() and DTRUNC() aliases also.)

```
create table t1 (d decimal(20,7));
-- By default, no digits to the right of the decimal point.
insert into t1 values (1.1), (2.22), (3.333), (4.4444), (5.5555
5);
select trunc(d) from t1 order by d;
+----------+
| trunc(d) |
+----------+
| 1        |
| 2        |
| 3        |
| 4        |
| 5        |
+----------+

-- 1 digit to the right of the decimal point.
select trunc(d,1) from t1 order by d;
+-------------+
| trunc(d, 1) |
+-------------+
| 1.1         |
| 2.2         |
| 3.3         |
| 4.4         |
| 5.5         |
+-------------+

-- 2 digits to the right of the decimal point,
-- including trailing zeroes if needed.
select trunc(d,2) from t1 order by d;
+-------------+
| trunc(d, 2) |
+-------------+
| 1.10        |
| 2.22        |
| 3.33        |
| 4.44        |
| 5.55        |
+-------------+
insert into t1 values (9999.9999), (8888.8888);

-- Negative scale truncates digits to the left
-- of the decimal point.
select trunc(d,-2) from t1 where d > 100 order by d;
+--------------+
| trunc(d, -2) |
+--------------+
| 8800         |
| 9900         |
+--------------+

-- The scale of the result is adjusted to match the
-- scale argument.
```

```
select trunc(d,2),
  precision(trunc(d,2)) as p,
  scale(trunc(d,2)) as s
from t1 order by d;
+-------------+----+---+
| trunc(d, 2) | p  | s |
+-------------+----+---+
| 1.10        | 15 | 2 |
| 2.22        | 15 | 2 |
| 3.33        | 15 | 2 |
| 4.44        | 15 | 2 |
| 5.55        | 15 | 2 |
| 8888.88     | 15 | 2 |
| 9999.99     | 15 | 2 |
+-------------+----+---+
```

```
create table dbl (d double);

insert into dbl values
  (1.1), (2.22), (3.333), (4.4444), (5.55555),
  (8888.8888), (9999.9999);

-- With double values, there is no optional scale argument.
select trunc(d) from dbl order by d;
+----------+
| trunc(d) |
+----------+
| 1        |
| 2        |
| 3        |
| 4        |
| 5        |
| 8888     |
| 9999     |
+----------+
```

UNHEX(STRING a)

Purpose: Returns a string of characters with ASCII values corresponding to pairs of hexadecimal digits in the argument.

Return type: STRING

WIDTH_BUCKET(DECIMAL expr, DECIMAL min_value, DECIMAL max_value, INT num_buckets)

Purpose: Returns the bucket number in which the expr value would fall in the histogram where its range between min_value and max_value is divided into num_buckets buckets of identical sizes.

The function returns:

- NULL if any argument is NULL.
- 0 if expr < min_value.
- num_buckets + 1 if expr >= max_val.
- If none of the above, the bucket number where expr falls.

Arguments:The following rules apply to the arguments.

- min_val is the minimum value of the histogram range.
- max_val is the maximum value of the histogram range.
- num_buckets must be greater than 0.
- min_value must be less than max_value.

Usage notes:

Each bucket contains values equal to or greater than the base value of that bucket and less than the base value of the next bucket. For example, with width_bucket(8, 1, 10, 3), the bucket ranges are actually the 0th "underflow bucket" with the range (-infinity to 0.999...), (1 to 3.999...), (4, to 6.999...), (7 to 9.999...), and the "overflow bucket" with the range (10 to infinity).

Return type: BIGINT

Added in: Impala 3.1.

Examples:

The below function creates 3 buckets between the range of 1 and 20 with the bucket width of 6.333, and returns 2 for the bucket #2 where the value 8 falls in:

```
WIDTH_BUCKET(8, 1, 20, 3)
```

The below statement returns a list of accounts with the energy spending and the spending bracket each account falls in, between 0 and 11. Bucket 0 (underflow bucket) will be assigned to the accounts whose energy spendings are less than $50. Bucket 11 (overflow bucket) will be assigned to the accounts whose energy spendings are more than or equal to $1000.

```
SELECT account, invoice_amount, WIDTH_BUCKET(invoice_amount,50,1
000,10)
FROM invoices_june2018
ORDER BY 3;
```

Related Information

Mathematical constant e

Factorial

Fowler–Noll–Vo hash function

Natural logarithm

MurmurHash2

More Hash Function Tests

Hash functions: An empirical comparison

Impala bit functions

Bit manipulation functions perform bitwise operations involved in scientific processing or computer science algorithms. For example, these functions include setting, clearing, or testing bits within an integer value, or changing the positions of bits with or without wraparound.

If a function takes two integer arguments that are required to be of the same type, the smaller argument is promoted to the type of the larger one if required. For example, BITAND(1,4096) treats both arguments as SMALLINT, because 1 can be represented as a TINYINT but 4096 requires a SMALLINT.

Remember that all Impala integer values are signed. Therefore, when dealing with binary values where the most significant bit is 1, the specified or returned values might be negative when represented in base 10.

Whenever any argument is NULL, either the input value, bit position, or number of shift or rotate positions, the return value from any of these functions is also NULL

Related information:

The bit functions operate on all the integral data types:

- INT
- BIGINT
- SMALLINT
- TINYINT

Function reference:

Impala supports the following bit functions:

- BITAND
- BITNOT
- BITOR
- BITXOR
- COUNTSET
- GETBIT
- ROTATELEFT
- ROTATERIGHT
- SETBIT
- SHIFTLEFT
- SHIFTRIGHT

BITAND(integer_type a, same_type b)

Purpose: Returns an integer value representing the bits that are set to 1 in both of the arguments. If the arguments are of different sizes, the smaller is promoted to the type of the larger.

Usage notes: The BITAND() function is equivalent to the & binary operator.

Return type: Same as the input value

Examples:

The following examples show the results of ANDing integer values. 255 contains all 1 bits in its lowermost 7 bits. 32767 contains all 1 bits in its lowermost 15 bits. You can use the bin() function to check the binary representation of any integer value, although the result is always represented as a 64-bit value. If necessary, the smaller argument is promoted to the type of the larger one.

```
select bitand(255, 32767); /* 0000000011111111 & 0111111111111111
 */
+--------------------+
| bitand(255, 32767) |
+--------------------+
| 255                |
+--------------------+
select bitand(32767, 1); /* 0111111111111111 & 0000000000000001
 */
+-------------------+
| bitand(32767, 1) |
+-------------------+
| 1                 |
+-------------------+

select bitand(32, 16); /* 00010000 & 00001000 */
+----------------+
| bitand(32, 16) |
+----------------+
| 0              |
+----------------+

select bitand(12,5); /* 00001100 & 00000101 */
+---------------+
| bitand(12, 5) |
+---------------+
| 4             |
+---------------+
select bitand(-1,15); /* 11111111 & 00001111 */
+----------------+
| bitand(-1, 15) |
```

```
+----------------+
| 15             |
+----------------+
```

BITNOT(integer_type a)

Purpose: Inverts all the bits of the input argument.

Usage notes: The BITNOT() function is equivalent to the ~ unary operator.

Examples:

These examples illustrate what happens when you flip all the bits of an integer value. The sign always changes. The decimal representation is one different between the positive and negative values.

```
select bitnot(127); /* 01111111 -> 10000000 */
+-------------+
| bitnot(127) |
+-------------+
| -128        |
+-------------+

select bitnot(16); /* 00010000 -> 11101111 */
+------------+
| bitnot(16) |
+------------+
| -17        |
+------------+

select bitnot(0); /* 00000000 -> 11111111 */
+-----------+
| bitnot(0) |
+-----------+
| -1        |
+-----------+

select bitnot(-128); /* 10000000 -> 01111111 */
+--------------+
| bitnot(-128) |
+--------------+
| 127          |
+--------------+
```

BITOR(integer_type a, same_type b)

Purpose: Returns an integer value representing the bits that are set to 1 in either of the arguments. If the arguments are of different sizes, the smaller is promoted to the type of the larger.

Usage notes: The BITOR() function is equivalent to the | binary operator.

Return type: Same as the input value

Examples:

The following examples show the results of ORing integer values.

```
select bitor(1,4); /* 00000001 | 00000100 */
+-------------+
| bitor(1, 4) |
+-------------+
| 5           |
+-------------+

select bitor(16,48); /* 00001000 | 00011000 */
+---------------+
```

```
| bitor(16, 48) |
+---------------+
| 48            |
+---------------+

select bitor(0,7); /* 00000000 | 00000111 */
+-------------+
| bitor(0, 7) |
+-------------+
| 7           |
+-------------+
```

BITXOR(integer_type a, same_type b)

Purpose: Returns an integer value representing the bits that are set to 1 in one but not both of the arguments. If the arguments are of different sizes, the smaller is promoted to the type of the larger.

Usage notes: The BITXOR() function is equivalent to the ^ binary operator.

Return type: Same as the input value

Examples:

The following examples show the results of XORing integer values. XORing a non-zero value with zero returns the non-zero value. XORing two identical values returns zero, because all the 1 bits from the first argument are also 1 bits in the second argument. XORing different non-zero values turns off some bits and leaves others turned on, based on whether the same bit is set in both arguments.

```
select bitxor(0,15); /* 00000000 ^ 00001111 */
+---------------+
| bitxor(0, 15) |
+---------------+
| 15            |
+---------------+

select bitxor(7,7); /* 00000111 ^ 00000111 */
+--------------+
| bitxor(7, 7) |
+--------------+
| 0            |
+--------------+

select bitxor(8,4); /* 00001000 ^ 00000100 */
+--------------+
| bitxor(8, 4) |
+--------------+
| 12           |
+--------------+

select bitxor(3,7); /* 00000011 ^ 00000111 */
+--------------+
| bitxor(3, 7) |
+--------------+
| 4            |
+--------------+
```

COUNTSET(integer_type a [, INT zero_or_one])

Purpose: By default, returns the number of 1 bits in the specified integer value. If the optional second argument is set to zero, it returns the number of 0 bits instead.

Usage notes:

In discussions of information theory, this operation is referred to as the “population count” or “popcount”.

Return type: Same as the input value

Examples:

The following examples show how to count the number of 1 bits in an integer value.

```
select countset(1); /* 00000001 */
+-------------+
| countset(1) |
+-------------+
| 1           |
+-------------+

select countset(3); /* 00000011 */
+-------------+
| countset(3) |
+-------------+
| 2           |
+-------------+

select countset(16); /* 00010000 */
+--------------+
| countset(16) |
+--------------+
| 1            |
+--------------+

select countset(17); /* 00010001 */
+--------------+
| countset(17) |
+--------------+
| 2            |
+--------------+

select countset(7,1); /* 00000111 = 3 1 bits; the function counts
 1 bits by default */
+----------------+
| countset(7, 1) |
+----------------+
| 3              |
+----------------+

select countset(7,0); /* 00000111 = 5 0 bits; third argument can
only be 0 or 1 */
+----------------+
| countset(7, 0) |
+----------------+
| 5              |
+----------------+
```

GETBIT(integer_type a, INT position)

Purpose: Returns a 0 or 1 representing the bit at a specified position. The positions are numbered right to left, starting at zero. The position argument cannot be negative.

Usage notes:

When you use a literal input value, it is treated as an 8-bit, 16-bit, and so on value, the smallest type that is appropriate. The type of the input value limits the range of the positions. Cast the input value to the appropriate type if you need to ensure it is treated as a 64-bit, 32-bit, and so on value.

Return type: Same as the input value

Examples:

The following examples show how to test a specific bit within an integer value.

```
select getbit(1,0); /* 00000001 */
+--------------+
| getbit(1, 0) |
+--------------+
| 1            |
+--------------+
select getbit(16,1) /* 00010000 */
+---------------+
| getbit(16, 1) |
+---------------+
| 0             |
+---------------+

select getbit(16,4) /* 00010000 */
+---------------+
| getbit(16, 4) |
+---------------+
| 1             |
+---------------+
select getbit(16,5) /* 00010000 */
+---------------+
| getbit(16, 5) |
+---------------+
| 0             |
+---------------+

select getbit(-1,3); /* 11111111 */
+---------------+
| getbit(-1, 3) |
+---------------+
| 1             |
+---------------+

select getbit(-1,25); /* 11111111 */
ERROR: Invalid bit position: 25

select getbit(cast(-1 as int),25); /* 11111111111111111111111111
111111 */
+-----------------------------+
| getbit(cast(-1 as int), 25) |
+-----------------------------+
| 1                           |
+-----------------------------+
```

ROTATELEFT(integer_type a, INT positions)

Purpose: Rotates an integer value left by a specified number of bits. As the most significant bit is taken out of the original value, if it is a 1 bit, it is “rotated” back to the least significant bit. Therefore, the final value has the same number of 1 bits as the original value, just in different positions. In computer science terms, this operation is a “circular shift”.

Usage notes:

Specifying a second argument of zero leaves the original value unchanged. Rotating a -1 value by any number of positions still returns -1, because the original value has all 1 bits and all the 1 bits are preserved during rotation. Similarly, rotating a 0 value by any number of positions still returns 0. Rotating a value by the same number of bits as in the value returns the same value. Because this is a circular operation, the number of positions is not limited to the number of bits in the input value.

For example, rotating an 8-bit value by 1, 9, 17, and so on positions returns an identical result in each case.

Return type: Same as the input value

Examples:

```
select rotateleft(1,4); /* 00000001 -> 00010000 */
+------------------+
| rotateleft(1, 4) |
+------------------+
| 16               |
+------------------+
select rotateleft(-1,155); /* 11111111 -> 11111111 */
+----------------------+
| rotateleft(-1, 155) |
+----------------------+
| -1                   |
+----------------------+

select rotateleft(-128,1); /* 10000000 -> 00000001 */
+----------------------+
| rotateleft(-128, 1) |
+----------------------+
| 1                    |
+----------------------+

select rotateleft(-127,3); /* 10000001 -> 00001100 */
+----------------------+
| rotateleft(-127, 3) |
+----------------------+
| 12                   |
+----------------------+
```

ROTATERIGHT(integer_type a, INT positions)

Purpose: Rotates an integer value right by a specified number of bits. As the least significant bit is taken out of the original value, if it is a 1 bit, it is “rotated” back to the most significant bit. Therefore, the final value has the same number of 1 bits as the original value, just in different positions. In computer science terms, this operation is a “circular shift”.

Usage notes:

Specifying a second argument of zero leaves the original value unchanged. Rotating a -1 value by any number of positions still returns -1, because the original value has all 1 bits and all the 1 bits are preserved during rotation. Similarly, rotating a 0 value by any number of positions still returns 0. Rotating a value by the same number of bits as in the value returns the same value. Because this is a circular operation, the number of positions is not limited to the number of bits in the input value. For example, rotating an 8-bit value by 1, 9, 17, and so on positions returns an identical result in each case.

Return type: Same as the input value

Examples:

```
select rotateright(16,4); /* 00010000 -> 00000001 */
+--------------------+
| rotateright(16, 4) |
+--------------------+
| 1                  |
+--------------------+
select rotateright(-1,155); /* 11111111 -> 11111111 */
+-----------------------+
```

```
| rotateright(-1, 155) |
+----------------------+
| -1                   |
+----------------------+

select rotateright(-128,1); /* 10000000 -> 01000000 */
+----------------------+
| rotateright(-128, 1) |
+----------------------+
| 64                   |
+----------------------+

select rotateright(-127,3); /* 10000001 -> 00110000 */
+----------------------+
| rotateright(-127, 3) |
+----------------------+
| 48                   |
+----------------------+
```

SETBIT(integer_type a, INT position [, INT zero_or_one])

Purpose: By default, changes a bit at a specified position to a 1, if it is not already. If the optional third argument is set to zero, the specified bit is set to 0 instead.

Usage notes:

If the bit at the specified position was already 1 (by default) or 0 (with a third argument of zero), the return value is the same as the first argument. The positions are numbered right to left, starting at zero. (Therefore, the return value could be different from the first argument even if the position argument is zero.) The position argument cannot be negative.

When you use a literal input value, it is treated as an 8-bit, 16-bit, and so on value, the smallest type that is appropriate. The type of the input value limits the range of the positions. Cast the input value to the appropriate type if you need to ensure it is treated as a 64-bit, 32-bit, and so on value.

Return type: Same as the input value

Examples:

```
select setbit(0,0); /* 00000000 -> 00000001 */
+--------------+
| setbit(0, 0) |
+--------------+
| 1            |
+--------------+
select setbit(0,3); /* 00000000 -> 00001000 */
+--------------+
| setbit(0, 3) |
+--------------+
| 8            |
+--------------+

select setbit(7,3); /* 00000111 -> 00001111 */
+--------------+
| setbit(7, 3) |
+--------------+
| 15           |
+--------------+

select setbit(15,3); /* 00001111 -> 00001111 */
+---------------+
| setbit(15, 3) |
+---------------+
| 15            |
```

```
+---------------+

select setbit(0,32); /* By default, 0 is a TINYINT with only 8
 bits. */
ERROR: Invalid bit position: 32

select setbit(cast(0 as bigint),32); /* For BIGINT, the position
can be 0..63. */
+-------------------------------+
| setbit(cast(0 as bigint), 32) |
+-------------------------------+
| 4294967296                    |
+-------------------------------+

select setbit(7,3,1); /* 00000111 -> 00001111; setting to 1 is t
he default */
+-----------------+
| setbit(7, 3, 1) |
+-----------------+
| 15              |
+-----------------+

select setbit(7,2,0); /* 00000111 -> 00000011; third argument of
0 clears instead of sets */
+-----------------+
| setbit(7, 2, 0) |
+-----------------+
| 3               |
+-----------------+
```

SHIFTLEFT(integer_type a, INT positions)

Purpose: Shifts an integer value left by a specified number of bits. As the most significant bit is taken out of the original value, it is discarded and the least significant bit becomes 0. In computer science terms, this operation is a “logical shift”.

Usage notes:

The final value has either the same number of 1 bits as the original value, or fewer. Shifting an 8-bit value by 8 positions, a 16-bit value by 16 positions, and so on produces a result of zero.

Specifying a second argument of zero leaves the original value unchanged. Shifting any value by 0 returns the original value. Shifting any value by 1 is the same as multiplying it by 2, as long as the value is small enough; larger values eventually become negative when shifted, as the sign bit is set. Starting with the value 1 and shifting it left by N positions gives the same result as 2 to the Nth power, or pow(2,*N*).

Return type: Same as the input value

Examples:

```
select shiftleft(1,0); /* 00000001 -> 00000001 */
+-----------------+
| shiftleft(1, 0) |
+-----------------+
| 1               |
+-----------------+

select shiftleft(1,3); /* 00000001 -> 00001000 */
+-----------------+
| shiftleft(1, 3) |
+-----------------+
| 8               |
+-----------------+
```

```
select shiftleft(8,2); /* 00001000 -> 00100000 */
+-----------------+
| shiftleft(8, 2) |
+-----------------+
| 32              |
+-----------------+
select shiftleft(127,1); /* 01111111 -> 11111110 */
+-------------------+
| shiftleft(127, 1) |
+-------------------+
| -2                |
+-------------------+

select shiftleft(127,5); /* 01111111 -> 11100000 */
+-------------------+
| shiftleft(127, 5) |
+-------------------+
| -32               |
+-------------------+

select shiftleft(-1,4); /* 11111111 -> 11110000 */
+------------------+
| shiftleft(-1, 4) |
+------------------+
| -16              |
+------------------+
```

SHIFTRIGHT(integer_type a, INT positions)

Purpose: Shifts an integer value right by a specified number of bits. As the least significant bit is taken out of the original value, it is discarded and the most significant bit becomes 0. In computer science terms, this operation is a “logical shift”.

Usage notes:

Therefore, the final value has either the same number of 1 bits as the original value, or fewer. Shifting an 8-bit value by 8 positions, a 16-bit value by 16 positions, and so on produces a result of zero.

Specifying a second argument of zero leaves the original value unchanged. Shifting any value by 0 returns the original value. Shifting any positive value right by 1 is the same as dividing it by 2. Negative values become positive when shifted right.

Return type: Same as the input value

Examples:

```
select shiftright(16,0); /* 00010000 -> 00010000 */
+-------------------+
| shiftright(16, 0) |
+-------------------+
| 16                |
+-------------------+

select shiftright(16,4); /* 00010000 -> 00000001 */
+-------------------+
| shiftright(16, 4) |
+-------------------+
| 1                 |
+-------------------+

select shiftright(16,5); /* 00010000 -> 00000000 */
+-------------------+
```

```
| shiftright(16, 5) |
+-------------------+
| 0                 |
+-------------------+
select shiftright(-1,1); /* 11111111 -> 01111111 */
+-------------------+
| shiftright(-1, 1) |
+-------------------+
| 127               |
+-------------------+

select shiftright(-1,5); /* 11111111 -> 00000111 */
+-------------------+
| shiftright(-1, 5) |
+-------------------+
| 7                 |
+-------------------+
```

Related Information

Hamming weight (population count)

Circular shift

Logical shift

Impala type conversion functions

Conversion functions are usually used in combination with other functions, to explicitly pass the expected data types. Impala has strict rules regarding data types for function parameters. For example, Impala does not automatically convert a DOUBLE value to FLOAT, a BIGINT value to INT, or other conversion where precision could be lost or overflow could occur. Also, for reporting or dealing with loosely defined schemas in big data contexts, you might frequently need to convert values to or from the STRING type.

Note: Although in Impala 2.3, the SHOW FUNCTIONS output for database _IMPALA_BUILTINS contains some function signatures matching the pattern castto*, these functions are not intended for public use and are expected to be deprecated in future.

Function reference:

Impala supports the following type conversion functions:

- CAST
- TYPEOF

CAST(expression AS type)

Purpose: Returns *EXPRESSION* converted to the *TYPE* data type.

If the *EXPRESSION* value is of a type that cannot be converted to the target *TYPE*:

- Of DECIMAL, DATE, and BOOLEAN, the function returns an error.
- Of all other types, the function returns NULL.

Usage notes:

Use CAST when passing a column value or literal to a function that expects a parameter with a different type. Frequently used in SQL operations such as CREATE TABLE AS SELECT and INSERT ... VALUES to ensure that values from various sources are of the appropriate type for the destination columns.

Where practical, do a one-time CAST() operation during the ingestion process to make each column into the appropriate type, rather than using many CAST() operations in each query; doing type

conversions for each row during each query can be expensive for tables with millions or billions of rows.

The way this function deals with time zones when converting to or from TIMESTAMP values is affected by the ##use_local_tz_for_unix_timestamp_conversions startup flag for the `impalad` daemon. See TIMESTAMP Data type for details about how Impala handles time zone considerations for the TIMESTAMP data type.

CAST(expression AS type FORMAT pattern)

Purpose: Returns *EXPRESSION* converted to the *TYPE* data type based on the *PATTERN* format string. This signature of CAST() with the FORMAT clause is supported only for casts between STRING / CHAR / VARCHAR types and TIMESTAMP / DATE types.

Note: The patterns allowed in the FORMAT clause support ISO SQL:2016 standard patterns. Those patterns are not the same format patterns used with the other Impala conversion functions, e.g. TO_TIMESTAMP() and FROM_TIMESTAMP().

The following rules apply to *PATTERN*. Any exceptions to these rules are noted in the Details column of the table below.

- *PATTERN* is a case-insensitive STRING unless noted otherwise in the table below.
- If *PATTERN* is NULL, an empty string, or a number, an error returns.
- A fewer digits in *EXPRESSION* than specified in the *PATTERN* is accepted if a separator is correctly specified in the *PATTERN*. For example, CAST('5-01-2017' AS DATE FORMAT 'MM-dd-yyyy') returns DATE'2017-05-01'.
- If fewer number of digits are in *EXPRESSION* than specified in the *PATTERN*, the current date is used to complete the year pattern. For example, CAST('19/05' AS DATE FORMAT 'YYYY/MM') will return DATE'2019-05-01' when executed on August 8, 2019.

The following format patterns are supported in the FORMAT clause.

Pattern	Description	Details
YYYY	4-digit year.	
YYY	Last 3 digits of a year.	
YY	Last 2 digits of a year.	
Y	Last digit of a year	
RRRR	4-digit round year	If 1, 3 or 4-digit year values are provided in *EXPRESSION*, treated as YYYY. If 2-digit years are provided in *EXPRESSION*, treated as RR. For date/time to string conversions, treated as YYYY. If YYYY, YYY, YY, Y, or RR is given in the same *PATTERN* for a string to date/time conversion, an error returns.

Pattern	Description	Details

RR	2-digit round year.	For date/time to string conversion, same as YY.
		For string to date/time conversions, the first 2 digits of the year in the return value depends on the specified two-digit year and the last two digits of the current year as follows:
		• If the specified 2-digit year is 00 to 49:
		  • If the last 2 digits of the current year are 00 to 49, the returned year has the same first 2 digits as the current year.
		  • If the last 2 digits of the current year are 50 to 99, the first 2 digits of the returned year are 1 greater than the first 2 digits of the current year.
		• If the specified 2-digit year is 50 to 99:
		  • If the last 2 digits of the current year are 00 to 49, the first 2 digits of the returned year are 1 less than the first 2 digits of the current year.
		  • If the last 2 digits of the current year are 50 to 99, the returned year has the same first 2 digits as the current year.
		If YYYY, YYY, YY, Y, or RR is given in the same *PATTERN* for a string to date/time conversion, an error returns.
		If 1-digit year values are provided in *EXPRESSION*, it is treated as YYYY.

Q	Quarter of year (1-4)	Not supported in string to date/time conversions.
MM	Month	In date/time to string conversions, 1-digit months are prefixed with a zero.
MONTH Month month	Full month name	In string to date/time conversions: • Converts a textual month name to a 2-digit month number. • The input strings are expected without trailing spaces, e.g. "June", "August". • If the FX modifier is active for the token, a 9-character-long full month name with trailing spaces is expected for the input string. In date/time to string conversions: • Converts to a 9-character-long full month name with trailing spaces. • If the FM modifier is active for this token, converts to a month name without trailing spaces. This pattern token is case sensitive. MONTH, Month, month would require different casing of the input in string to date/time conversions and would produce cased outputs in date/time to string conversions, For example, MONTH, Month, month would convert respectively JULY, July, and july.
MON Mon mon	Abbreviated 3-character month name	This pattern token is case sensitive. MON, Mon, mon would require different casing of the input in string to date/time conversions and would produce cased outputs in date/time to string conversions, For example, MON, Mon, mon would convert respectively JUL, Jul, and jul.
WW	Week of year (1-53)	Not supported in string to date/time conversions. The 1st week begins on January 1st and ends on January 7th.
W	Week of month (1-5)	Week of month (1-5)
D	Day of week (1-7)	Day of week (1-7)
DD	Day of month (1-31)	In date/time to string conversions, a 1-digit day is prefixed with a zero.
DDD	Day of year (1-366)	In string to date/time conversions, providing MM and DD along with DDD results an error, e.g. YYYY-MM-DDD.

Pattern	Description	Details
DAY Day day	Day name	Not supported in string to date/time conversions. In date/time to string conversions: • Converts to a 9-character-long full day name with trailing spaces. • If the FM modifier is active for this token, converts to a day name without trailing spaces. This pattern token is case sensitive. DAY, Day, day would produce cased outputs in date/time to string conversions, For example, DAY, Day, day would convert respectively MONDAY, Monday, and monday.
	DY Dy dy	Not supported in string to date/time conversions. This pattern token is case sensitive. DY, Dy, dy would produce cased outputs in date/time to string conversions, For example, DY, Dy, dy would convert respectively MON, Mon, and mon.
HH HH12	Hour of day (1-12)	In date/time to string conversions, 1-digit hours are prefixed with a zero. If provided hour in *EXPRESSION* is not between 1 and 12, returns an error. If no AM/PM is provided in *EXPRESSION*, the default is AM.
HH HH12	Hour of day (1-12)	In date/time to string conversions, 1-digit hours are prefixed with a zero. If provided hour in *EXPRESSION* is not between 1 and 12, returns an error. If no AM/PM is provided in *EXPRESSION*, the default is AM.
HH24	Hour of day (0-23)	In string to date/time conversions, if HH12, AM, PM are given in the same *PATTERN*, an error returns.
MI	Minute of hour (0-59)	In date/time to string conversions, 1-digit minutes are prefixed with a zero.
SS	Second of minute (0-59)	In date/time to string conversions, 1-digit seconds are prefixed with a zero.
SSSSS	Second of Day (0-86399)	In string to timestamp conversions, if SS, HH, HH12, HH24, MI, AM/PM are given in the same *PATTERN*, an error returns.
FF FF1, ..., FF9	Fractional second	A number, 1 to 9, can be used to indicate the number of digits. FF specifies a 9 digits fractional second.
AM PM A.M. P.M.	Meridiem indicator	For date/time to string conversions, AM and PM are treated as synonyms. For example, casting '2019-01-01 11:00 am' to TIMESTAMP with the 'YYYY-MM-DD HH12:MI PM' pattern returns 01-JAN-19 11.00.00.000000 AM. For string to date/time conversion, HH24 in the same *PATTERN* returns an error.
TZH	Timezone offset hour	An optional sign, + or -, and 2 digits for the value of signed numbers are allowed for the source *EXPRESSION*, e.g. +10, -05, 04. The allowed values are from -15 to +5.
TZM	Timezone offset minute	Unsigned numbers are allowed for the source *EXPRESSION*. The allowed values are from 0 to 59.

Pattern	Description	Details
- . / , ' ; : <space>	Separator	For string to date/time conversions, any separator character in the *PATTERN* string would match any separator character in the input *EXPRESSION*. For example, CAST(“20191010” AS DATE FORMAT “YYYY-MM-DD”) returns an error, but CAST("2019-.;10 10" AS DATE FORMAT "YYYY-MM-DD") succeeds.
T	Separates the date from the time.	This pattern is used for accepting ISO 8601 date/time formats. Example: YYYY-MM-DDTHH24:MI:SS.FF9Z
Z	Indicates the zero hour offset from UTC.	This pattern is used for accepting ISO 8601 date/time formats.
FX	Forces an exact match between the format pattern, *PATTERN*, and the input argument, *EXPRESSION*.	Must be specified at the beginning of the format *PATTERN* and is valid for the whole *PATTERN*. In string to date/time conversions: • Forces strict separator matching, including the separator character. • Expects all the tokens to have the maximum possible length. In date/time to string conversions, the outputs are padded as follows: • Numeric outputs are left padded with zeros. • Text outputs are right padded with spaces.
FM	Overrides FX.	In a date/time to string conversions, FM suppresses blank padding for the element immediately following the FM in the *PATTERN* string, e.g. "2010-1-9". In string to date/time conversions, FM is used to override the effect of FX for certain tokens as follows: • The length of the token in the input argument can be shorter than the max length of that token type if followed by a separator, e.g. 1-digit month, less than 4-digit year. • FM modifier skips the separators and affects the next non-separator token. For example, CAST('1999-10-2' AS TIMESTAMP FORMAT 'FXYYYY-MMFM-DD') returns 1999-10-02 00:00:00 as FM is applied to DD and not to the separator(-).

Pattern	Description	Details
Free text	Nested string	You can specify a free text with surrounding double quotes (") in *PATTERN* where the same text is expected in the input argument without surrounding double quotes. In date/time to string conversions, the string is included in the output with the characters' case preserved. In string to date/time conversions, the nested string is not included in the resulting date/time object. However, the nested string has to match the related part of the input string, including the case. The following rules apply: • If the *PATTERN* is surrounded by double quotes, double quotes surrounding the free text token must be escaped with a single backslash (\"). For example: CAST('"Date:"2019-11-10' AS DATE FORMAT "\"Date:\"YYYY-MM-DD") • If the *PATTERN* is surrounded by double quotes, and there is an escaped double quotes inside a text token that is itself surrounded by escaped double quotes, escape the double quotes in the free text token with a triple backslash: (\\\") For example: CAST("1985 some \"text 11-28" AS DATE"' FORMAT "YYYY\" some \\\"text \"MM-DD") • If the *PATTERN* is surrounded by single quotes, the free text token can be surrounded by (non-escaped) double quotes. To include double quotes within the free text token, those double quotes must be escaped by a single backslash. For example: CAST('"Date:"2019-11-10' AS DATE FORMAT '"\"Date:\""YYYY-MM-DD') • Literal single quotes/apostrophes (') in the nested string must be escaped with a single backslash if the whole pattern string is delimited by single quotes. • If a free text token contains any separator characters in the beginning, and the text token is right after a separator token sequence, an error returns as it's not trivial to find where the separator sequence ends and where the free text starts. In this case, use FX for strict matching. For example: CAST("1985-11- ' 10" AS DATE FORMAT "YYYY-MM-\" ' \"DD") returns an error, but removing the dash before the text token makes this succeed.

Examples:

Input	Output
`CAST("02-11-2014" AS DATE FORMAT 'dd-mm-yyyy')`	2014-11-02
`CAST("365 2014" AS DATE FORMAT 'DDD-YYYY')`	2014-12-31
`CAST("5-01-26" AS DATE FORMAT 'YY-MM-DD')`	Executed at 2019-01-01 11:11:11: 2015-01-26
`CAST('2018-11-10T15:11:04Z' AS TIMESTAMP FORMAT 'YYYY-MM-DDTHH24:MI:SSZ')`	2018-11-10 15:11:04
`CAST("95-01-28" AS DATE FORMAT 'YYY-MM-DD')`	Executed at 2019-01-01 11:11:11: 2095-01-28
`CAST("49-01-15" AS DATE FORMAT 'RR-MM-DD')`	Round year when last 2 digits of current year is greater than 49. Executed at 2099-01-01 11:11:11: 2149-01-15
`CAST('2019.10.10 13:30:40.123456 +01:30' AS TIMESTAMP FORMAT 'YYYY-MM-DD HH24:MI:SS.FF9 TZH:TZM')`	2019-10-10 13:30:40.123456000
`CAST('2001-03-01 03:10:15.123456 -01:30' AS TIMESTAMP FORMAT 'FXYYYY-MM-DD HH12:MI:SS.FF6 TZH:TZM')`	2001-03-01 03:10:15.123456000
`CAST('2001-03-02 03:10:15' AS TIMESTAMP FORMAT 'FXYYYY MM-DD HH12:MI:SS')`	NULL because the separator between the year and month do not match.
`CAST('2001-3-05' AS TIMESTAMP FORMAT 'FXYYYY-MM-DD')`	NULL because the length of the month token does not match the *PATTERN*.
`CAST('2001-3-11 3:15:00.12345' AS TIMESTAMP FORMAT 'FXYYYY-FMMM-DD FMHH12:MI:SS.FMFF')`	2001-03-11 03:15:00.123450000 The multiple FM modifiers for the month and hour override the FX and suppress padding.
`CAST('2019-01-01 01:01:01' AS TIMESTAMP`	1:01:1

Allow implicit casts between numeric and string types when inserting into table

The current implementation requires explicit casts for numeric and string-based literals, however this release relaxes the implicit casting rules for these cases. This is controlled through a query option allow_unsafe_casts and is turned off by default. This query option allows implicit casting between some numeric types and string types.

Possibilities:

From	To
String	Float
String	Double, Real
String	Tinyint
String	Smallint
String	Int
String	Bigint
Float	String
Double, Real	String
Tinyint	String
Smallint	String
Int	String
Bigint	String

Example

In the following example, numeric and string literals are inserted into a FLOAT column. The assumption is that all available numeric types and string types should be interchangeable when they are listed as literals in insert statements.

- String to Float column:
  - INSERT INTO example(float_col) VALUES ("041241");
- Float to String column:
  - INSERT INTO example(string_col) VALUES (CAST(123.0 AS FLOAT);
- Integer to String column:
  - INSERT INTO example(string_col) VALUES (10000000);
- String and Integer literals to String column:
  - INSERT INTO example(STRING_COL) VALUES (123), ("example");

Casting rules

A hand-written query containing different types qualifies for implicit casting, but a referenced column does not. All available numeric types and string types are interchangeable when they are listed as literals in insert statements.

The insert statements and set operations accept unsafe implicitly casted expressions when the source expressions are constant.

The following implicit type casts are enabled in unsafe mode:

- String -> Float, Double
- String -> Tinyint, Smallint, Int, Bigint
- Float, Double -> String
- Tinyint, Smallint, Int, Bigint -> String

TYPEOF(expression)

Purpose: Returns the name of the data type corresponding to *EXPRESSION*. For types with extra attributes, such as length for CHAR and VARCHAR, or precision and scale for DECIMAL, includes the full specification of the type.

Return type: STRING

Usage notes: Typically used in interactive exploration of a schema, or in application code that programmatically generates schema definitions such as CREATE TABLE statements, for example, to get the type of an expression such as col1 / col2 or CONCAT(col1, col2, col3). This function is especially useful for arithmetic expressions involving DECIMAL types because the precision and scale of the result is can be different than that of the operands.

Examples:

TYPEOF(2) returns TINYINT.

TYPEOF(NOW()) returns TIMESTAMP.

Related Information

TIMESTAMP data type

Impala date and time functions

The underlying Impala data type for date and time data is TIMESTAMP and DATE.

Some of the datetime functions are affected by the setting of the ##use_local_tz_for_unix_timestamp_conversions startup flag for the `impalad` daemon.

- This setting is off by default, meaning that functions such as FROM_UNIXTIME() and UNIX_TIMESTAMP() consider the input values to always represent the UTC time zone.
- This setting also applies when you CAST() a BIGINT value to TIMESTAMP, or a TIMESTAMP value to BIGINT. When this setting is enabled, these functions and operations convert to and from values representing the local time zone. See TIMESTAMP data type on page 45 for details about how Impala handles time zone considerations for the TIMESTAMP data type.

Function reference:

Impala supports the following data and time functions:

- ADD_MONTHS
- ADDDATE
- CURRENT_TIMESTAMP
- DATE_ADD
- DATE_PART
- DATE_SUB
- DATE_TRUNC
- DATEDIFF
- DAY
- DAYNAME
- DAYOFWEEK
- DAYOFYEAR
- DAYS_ADD
- DAYS_SUB
- EXTRACT
- FROM_TIMESTAMP
- FROM_UNIXTIME
- FROM_UTC_TIMESTAMP

- HOUR
- HOURS_ADD
- HOURS_SUB
- INT_MONTHS_BETWEEN
- MICROSECONDS_ADD
- MICROSECONDS_SUB
- MILLISECOND
- MILLISECONDS_ADD
- MILLISECONDS_SUB
- MINUTE
- MINUTES_ADD
- MINUTES_SUB
- MONTH
- MONTHNAME
- MONTHS_ADD
- MONTHS_BETWEEN
- MONTHS_SUB
- NANOSECONDS_ADD
- NANOSECONDS_SUB
- NEXT_DAY
- NOW
- QUARTER
- SECOND
- SECONDS_ADD
- SECONDS_SUB
- SUBDATE
- TIMEOFDAY
- TIMESTAMP_CMP
- TO_DATE
- TO_TIMESTAMP
- TO_UTC_TIMESTAMP
- TRUNC
- UNIX_TIMESTAMP
- UTC_TIMESTAMP
- WEEKOFYEAR
- WEEKS_ADD
- WEEKS_SUB
- YEAR
- YEARS_ADD
- YEARS_SUB

ADD_MONTHS(TIMESTAMP / DATE date, INT months), ADD_MONTHS(TIMESTAMP / DATE date, BIGINT months)

Purpose: Adds *MONTHS* to *DATE* and returns the new date value.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

Usage notes:

Same as MONTHS_ADD(). Available in Impala 1.4 and higher. For compatibility when porting code with vendor extensions.

ADDDATE(TIMESTAMP / DATE date, INT / BIGINT days)

Purpose: Adds *DAYS* to *DATE* and returns the new date value.

The *DAYS* value can be negative, which gives the same result as the SUBDATE() function.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

CURRENT_DATE()

Purpose: Returns the current date.

Any references to the CURRENT_DATE() function are evaluated at the start of a query. All calls to CURRENT_DATE() within the same query return the same value, and the value does not depend on how long the query takes.

Return type: DATE

CURRENT_TIMESTAMP()

Purpose: Alias for the NOW() function.

Return type: TIMESTAMP

DATE_ADD(TIMESTAMP / DATE date, INT / BIGINT days), DATE_ADD(TIMESTAMP / DATE date, interval_expression)

Purpose: Adds a specified number of days to the *DATE* argument.

With an INTERVAL expression as the second argument, you can calculate a delta value using other units such as weeks, years, hours, seconds, and so on.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

The following examples show the shorthand notation of an INTERVAL expression, instead of specifying the precise number of days. The INTERVAL notation also lets you work with units smaller than a single day.

```
select now() as right_now, date_add(now(), interval 3 weeks) as
 in_3_weeks;
+-------------------------------+-------------------------------+
| right_now                     | in_3_weeks                    |
+-------------------------------+-------------------------------+
| 2016-05-20 11:05:39.173331000 | 2016-06-10 11:05:39.173331000 |
+-------------------------------+-------------------------------+

select now() as right_now, date_add(now(), interval 6 hours) as
in_6_hours;
+-------------------------------+-------------------------------+
| right_now                     | in_6_hours                    |
+-------------------------------+-------------------------------+
| 2016-05-20 11:13:51.492536000 | 2016-05-20 17:13:51.492536000 |
+-------------------------------+-------------------------------+
```

Like all date/time functions that deal with months, date_add() handles nonexistent dates past the end of a month by setting the date to the last day of the month. The following example shows how the nonexistent date April 31st is normalized to April 30th:

```
select date_add(cast('2016-01-31' as timestamp), interval 3 mon
ths) as 'april_31st';
+---------------------+
| april_31st          |
+---------------------+
| 2016-04-30 00:00:00 |
+---------------------+
```

DATE_CMP(DATE date1, DATE date2)

Purpose: Compares *DATE1* and *DATE2* and returns:

- 0 if the dates are identical.
- 1 if *DATE1* > *DATE2*.
- -1 if *DATE1* < *DATE2*.
- NULL if *DATE1* or *DATE2* is NULL.

Return type: INT

DATE_PART(STRING part, TIMESTAMP / DATE date)

Purpose: Similar to EXTRACT(), with the argument order reversed. Supports the same date and time units as EXTRACT(). For compatibility with SQL code containing vendor extensions.

Return type: BIGINT

DATE_SUB(TIMESTAMP startdate, INT days), DATE_SUB(TIMESTAMP startdate, interval_expression)

Purpose: Subtracts a specified number of days from a TIMESTAMP value. With an INTERVAL expression as the second argument, you can calculate a delta value using other units such as weeks, years, hours, seconds, and so on.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

Examples:

The following example shows the simplest usage, of subtracting a specified number of days from a TIMESTAMP value:

```
select now() as right_now, date_sub(now(), 7) as last_week;
+-------------------------------+-------------------------------+
| right_now                     | last_week                     |
+-------------------------------+-------------------------------+
| 2016-05-20 11:21:30.491011000 | 2016-05-13 11:21:30.491011000 |
+-------------------------------+-------------------------------+
```

The following examples show the shorthand notation of an INTERVAL expression, instead of specifying the precise number of days. The INTERVAL notation also lets you work with units smaller than a single day.

```
select now() as right_now, date_sub(now(), interval 3 weeks) as
 3_weeks_ago;
+-------------------------------+-------------------------------+
| right_now                     | 3_weeks_ago                   |
```

```
+-------------------------------+-------------------------------+
| 2016-05-20 11:23:05.176953000 | 2016-04-29 11:23:05.176953000 |
+-------------------------------+-------------------------------+
select now() as right_now, date_sub(now(), interval 6 hours) as
 6_hours_ago;
+-------------------------------+-------------------------------+
| right_now                     | 6_hours_ago                   |
+-------------------------------+-------------------------------+
| 2016-05-20 11:23:35.439631000 | 2016-05-20 05:23:35.439631000 |
+-------------------------------+-------------------------------+
```

Like all date/time functions that deal with months, date_add() handles nonexistent dates past the end of a month by setting the date to the last day of the month. The following example shows how the nonexistent date April 31st is normalized to April 30th:

```
select date_sub(cast('2016-05-31' as timestamp), interval 1 mon
ths) as 'april_31st';
+---------------------+
| april_31st          |
+---------------------+
| 2016-04-30 00:00:00 |
+---------------------+
```

DATE_TRUNC(STRING unit, TIMESTAMP / DATE ts)

Purpose: Returns the *TS* value truncated to the specified *UNIT*.

Argument: The unit argument is not case-sensitive. This argument string can be one of:

Unit	Supported for TIMESTAMP	Supported for DATE
'MICROSECONDS'	Yes	No
'MILLISECONDS'	Yes	No
'SECOND'	Yes	No
'MINUTE'	Yes	No
'HOUR'	Yes	No
'DAY'	Yes	Yes
'WEEK'	Yes	Yes
'MONTH'	Yes	Yes
'YEAR'	Yes	Yes
'DECADE'	Yes	Yes
'CENTURY'	Yes	Yes
'MILLENNIUM'	Yes	Yes

Usage notes:

Although this function is similar to calling TRUNC() with a TIMESTAMP or DATE argument, the order of arguments and the recognized units are different between TRUNC() and DATE_TRUNC(). Therefore, these functions are not interchangeable.

This function is typically used in GROUP BY queries to aggregate results from the same hour, day, week, month, quarter, and so on. You can also use this function in an INSERT ... SELECT into a partitioned table to divide TIMESTAMP or DATE values into the correct partition.

Return type:

- TIMESTAMP if the second argument, *TS*, is TIMESTAMP.

- DATE if the second argument, *TS*, is DATE.

Examples:

DATE_TRUNC('HOUR', NOW()) returns 2017-12-05 13:00:00.

DATE_TRUNC('MILLENNIUM', DATE'2019-08-02') returns 2001-01-01.

DATEDIFF(TIMESTAMP / DATE enddate, TIMESTAMP / DATE startdate)

Purpose: Returns the number of days from *STARTDATE* to *ENDDATE*.

If *ENDDATE* > *STARTDATE*, the return value is positive.

If *ENDDATE* < *STARTDATE*, the return value is negative.

If *ENDDATE* = *STARTDATE*, the return value is zero.

Return type: INT

Usage notes:

The time portions of the *ENDDATE* and *STARTDATE* values are ignored. For example, 11:59 PM on one day and 12:01 AM on the next day represent a DATEDIFF() of -1 because the date/time values represent different days, even though the TIMESTAMP values differ by only 2 minutes.

DAY(TIMESTAMP / DATE date), DAYOFMONTH(TIMESTAMP / DATE date)

Purpose: Returns the day value from the *DATE* argument. The value represents the day of the month, therefore is in the range 1-31, or less for months without 31 days.

Returns NULL for nonexistent dates, e.g. Feb 30, or misformatted date strings, e.g. '1999-02-013'.

Return type: INT

DAYNAME(TIMESTAMP / DATE date)

Purpose: Returns the day name of the *DATE* argument. The range of return values is 'Sunday' to 'Sat urday'. Used in report-generating queries, as an alternative to calling DAYOFWEEK() and turning that numeric return value into a string using a CASE expression.

Return type: STRING

DAYOFWEEK(TIMESTAMP / DATE date)

Purpose: Returns the day field of the *DATE* arguement, corresponding to the day of the week. The range of return values is 1 (Sunday) to 7 (Saturday).

Return type: INT

DAYOFYEAR(TIMESTAMP / DATE date)

Purpose: Returns the day field from the *DATE* argument, corresponding to the day of the year. The range of return values is 1 (January 1) to 366 (December 31 of a leap year).

Return type: INT

DAYS_ADD(TIMESTAMP / DATE date, INT / BIGINT days)

Purpose: Returns the value with the number of *DAYS* added to *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

DAYS_SUB(TIMESTAMP / DATE date, INT / BIGINT days)

Purpose: Returns the value with the number of *DAYS* subtracted from *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

EXTRACT(TIMESTAMP / DATE ts, STRING unit), EXTRACT(unit FROM TIMESTAMP / DATE ts)

Purpose: Returns one of the numeric date or time fields, specified by *UNIT*, from *TS*.

Argument: The unit argument value is not case-sensitive. The unit string can be one of:

Unit	Supported for TIMESTAMP ts	Supported for DATE ts
'EPOCH'	Yes	No
'MILLISECOND'	Yes	No
'SECOND'	Yes	No
'MINUTE'	Yes	No
'HOUR'	Yes	No
'DAY'	Yes	Yes
'MONTH'	Yes	Yes
'QUARTER'	Yes	Yes
'YEAR'	Yes	Yes

Usage notes:

Typically used in GROUP BY queries to arrange results by hour, day, month, and so on. You can also use this function in an INSERT ... SELECT statement to insert into a partitioned table to split up TIMESTAMP values into individual parts, if the partitioned table has separate partition key columns representing year, month, day, and so on. If you need to divide by more complex units of time, such as by week or by quarter, use the TRUNC() function instead.

Return type: BIGINT

Examples:

EXTRACT(DAY FROM DATE'2019-08-17') returns 17.

If you specify 'MILLISECOND' for the *UNIT* argument, the function returns the seconds component and the milliseconds component.

EXTRACT(CAST('2006-05-12 18:27:28.123456789' AS TIMESTAMP), 'MILLISECOND') returns 28123.

FROM_TIMESTAMP(TIMESTAMP datetime, STRING pattern), FROM_TIMESTAMP(STRING datetime, STRING pattern)

Purpose: Converts a TIMESTAMP value into a string representing the same value.

Return type: STRING

Usage notes:

The FROM_TIMESTAMP() function provides a flexible way to convert TIMESTAMP values into arbitrary string formats for reporting purposes.

Because Impala implicitly converts string values into TIMESTAMP, you can pass date/time values represented as strings (in the standard yyyy-MM-dd HH:mm:ss.SSS format) to this function. The result is a string using different separator characters, order of fields, spelled-out month names, or other variation of the date/time string representation.

The allowed tokens for the pattern string are the same as for the FROM_UNIXTIME() function.

FROM_UNIXTIME(BIGINT unixtime[, STRING pattern])

Purpose: Converts the number of seconds from the Unix epoch to the specified time into a string in the local time zone.

Return type: STRING

The *PATTERN* string supports the following subset of Java SimpleDateFormat.

Pattern	Description
y	Year
M	Month
d	Day
H	Hour
m	Minute
s	Second
S	Fractional second
+/-hh:mm	Time zone offset
+/-hhmm	Time zone offset
+/-hh	Time zone offset

The following rules apply to the *PATTERN* string:

- The *PATTERN* string is case-sensitive.
- All fields are variable length, and thus must use separators to specify the boundaries of the fields, with the exception of the time zone values.
- Time zone offset formats must be at the end of the *PATTERN* string.
- Formatting character groups can appear in any order along with any separators except for the time zone offset. For example:
  - yyyy/MM/dd
  - dd-MMM-yy
  - (dd)(MM)(yyyy) HH:mm:ss
  - yyyy-MM-dd HH:mm:ss+hh:mm

In Impala 1.3 and later, you can switch the order of elements, use alternative separator characters, and use a different number of placeholders for each unit. Adding more instances of y, d, H, and so on produces output strings zero-padded to the requested number of characters. The exception is M for months, where M produces a non-padded value such as 3, MM produces a zero-padded value such as 03, MMM produces an abbreviated month name such as Mar, and sequences of 4 or more M are not allowed.

A date string including all fields could be 'yyyy-MM-dd HH:mm:ss.SSSSSS', 'dd/MM/yyyy HH:mm:ss.SSSSSS', 'MMM dd, yyyy HH.mm.ss (SSSSSS)' or other combinations of placeholders and separator characters.

Usage notes:

The way this function deals with time zones when converting to or from TIMESTAMP values is affected by the ##use_local_tz_for_unix_timestamp_conversions startup flag for the `impalad` daemon.

FROM_UTC_TIMESTAMP(TIMESTAMP ts, STRING timezone)

Purpose: Converts a specified UTC timestamp value into the appropriate value for a specified time zone.

Return type: TIMESTAMP

Usage notes: Often used to translate UTC time zone data stored in a table back to the local date and time for reporting. The opposite of the TO_UTC_TIMESTAMP() function.

To determine the time zone of the server you are connected to, you can call the TIMEOFDAY() function, which includes the time zone specifier in its return value. Remember that with cloud computing, the server you interact with might be in a different time zone than you are, or different

sessions might connect to servers in different time zones, or a cluster might include servers in more than one time zone.

See discussion of time zones in TIMESTAMP data type on page 45 for information about using this function for conversions between the local time zone and UTC.

HOUR(TIMESTAMP ts)

Purpose: Returns the hour field from a TIMESTAMP field.

Return type: INT

HOURS_ADD(TIMESTAMP date, INT hours), HOURS_ADD(TIMESTAMP date, BIGINT hours)

Purpose: Returns the specified date and time plus some number of hours.

Return type: TIMESTAMP

HOURS_SUB(TIMESTAMP date, INT hours), HOURS_SUB(TIMESTAMP date, BIGINT hours)

Purpose: Returns the specified date and time minus some number of hours.

Return type: TIMESTAMP

INT_MONTHS_BETWEEN(TIMESTAMP / DATE enddate, TIMESTAMP / DATE startdate)

Purpose: Returns the number of months from *STARTDATE* to *ENDDATE*, representing only the full months that passed.

Return type: INT

Usage notes:

Typically used in business contexts, for example to determine whether a specified number of months have passed or whether some end-of-month deadline was reached.

The method of determining the number of elapsed months includes some special handling of months with different numbers of days that creates edge cases for dates between the 28th and 31st days of certain months. See MONTHS_BETWEEN() for details. The INT_MONTHS_BETWEEN() result is essentially the FLOOR() of the MONTHS_BETWEEN() result.

If either value is NULL, which could happen for example when converting a nonexistent date string such as '2015-02-29' to a TIMESTAMP, the result is also NULL.

If the first argument represents an earlier time than the second argument, the result is negative.

LAST_DAY(TIMESTAMP / DATE ts)

Purpose: Returns the beginning of the last calendar day in the same month of *TS*.

Return type:

- Returns TIMESTAMP if *TS* is of the TIMESTAMP type.
- Returns DATE if *TS* is of the DATE type.

Usage notes:

If the input argument does not represent a valid Impala TIMESTAMP including both date and time portions, the function returns NULL. For example, if the input argument is a string that cannot be implicitly cast to TIMESTAMP, does not include a date portion, or is out of the allowed range for Impala TIMESTAMP values, the function returns NULL.

MICROSECONDS_ADD(TIMESTAMP date, INT microseconds), MICROSECONDS_ADD(TIMESTAMP date, BIGINT microseconds)

Purpose: Returns the specified date and time plus some number of microseconds.

Return type: TIMESTAMP

MICROSECONDS_SUB(TIMESTAMP date, INT microseconds), MICROSECONDS_SUB(TIMESTAMP date, BIGINT microseconds)

Purpose: Returns the specified date and time minus some number of microseconds.

Return type: TIMESTAMP

MILLISECOND(TIMESTAMP ts)

Purpose: Returns the millisecond portion of a TIMESTAMP value.

Return type: INT

Usage notes:

The millisecond value is truncated, not rounded, if the TIMESTAMP value contains more than 3 significant digits to the right of the decimal point.

MILLISECONDS_ADD(TIMESTAMP date, INT milliseconds), MILLISECONDS_ADD(TIMESTAMP date, BIGINT milliseconds)

Purpose: Returns the specified date and time plus some number of milliseconds.

Return type: TIMESTAMP

MILLISECONDS_SUB(TIMESTAMP date, INT milliseconds), MILLISECONDS_SUB(TIMESTAMP date, BIGINT milliseconds)

Purpose: Returns the specified date and time minus some number of milliseconds.

Return type: TIMESTAMP

MINUTE(TIMESTAMP date)

Purpose: Returns the minute field from a TIMESTAMP value.

Return type: INT

MINUTES_ADD(TIMESTAMP date, INT minutes), MINUTES_ADD(TIMESTAMP date, BIGINT minutes)

Purpose: Returns the specified date and time plus some number of minutes.

Return type: TIMESTAMP

MINUTES_SUB(TIMESTAMP date, INT minutes), MINUTES_SUB(TIMESTAMP date, BIGINT minutes)

Purpose: Returns the specified date and time minus some number of minutes.

Return type: TIMESTAMP

MONTH(TIMESTAMP / DATE date)

Purpose: Returns the month field, represented as an integer, from the *DATE* argument.

Return type: INT

MONTHNAME(TIMESTAMP / DATE date)

Purpose: Returns the month name of the *DATE* argument.

Return type: STRING

MONTHS_ADD(TIMESTAMP / DATE date, INT / BIGINT months)

Purpose: Returns the value with the number of *MONTHS* added to *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

Usage notes:

If *DATE* is the last day of a month, the return date will fall on the last day of the target month, e.g. MONTHS_ADD(DATE'2019-01-31', 1) returns DATE'2019-02-28'.

MONTHS_BETWEEN(TIMESTAMP / DATE enddate, TIMESTAMP / DATE startdate)

Purpose: Returns the number of months from *STARTDATE* to *ENDDATE*.

This result can include a fractional part representing extra days in addition to the full months between the dates. The fractional component is computed by dividing the difference in days by 31 (regardless of the month).

Return type: DOUBLE

Usage notes:

Typically used in business contexts, for example to determine whether a specified number of months have passed or whether some end-of-month deadline was reached.

If the only consideration is the number of full months and any fractional value is not significant, use INT_MONTHS_BETWEEN() instead.

The method of determining the number of elapsed months includes some special handling of months with different numbers of days that creates edge cases for dates between the 28th and 31st days of certain months.

If either value is NULL, which could happen for example when converting a nonexistent date string such as '2015-02-29' to a TIMESTAMP, the result is also NULL.

If the first argument represents an earlier time than the second argument, the result is negative.

The time portion of the input arguements are ignored.

MONTHS_SUB(TIMESTAMP / DATE date, INT / BIGINT months)

Purpose: Returns the value with the number of *MONTHS* subtracted from *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

Usage notes:

If *DATE* is the last day of a month, the return date will fall on the last day of the target month, e.g. MONTHS_SUB(DATE'2019-02-28', 1) returns DATE'2019-01-31'.

NANOSECONDS_ADD(TIMESTAMP date, INT nanoseconds), NANOSECONDS_ADD(TIMESTAMP date, BIGINT nanoseconds)

Purpose: Returns the specified date and time plus some number of nanoseconds.

Return type: TIMESTAMP

Kudu considerations:

The nanosecond portion of an Impala TIMESTAMP value is rounded to the nearest microsecond when that value is stored in a Kudu table.

NANOSECONDS_SUB(TIMESTAMP date, INT nanoseconds), NANOSECONDS_SUB(TIMESTAMP date, BIGINT nanoseconds)

Purpose: Returns the specified date and time minus some number of nanoseconds.

Return type: TIMESTAMP

Kudu considerations:

The nanosecond portion of an Impala TIMESTAMP value is rounded to the nearest microsecond when that value is stored in a Kudu table.

NEXT_DAY(TIMESTAMP / DATE date, STRING weekday)

Purpose: Returns the date of the *WEEKDAY* that follows the specified *DATE*.

Argument: The *WEEKDAY* is not case-sensitive.

The following values are accepted for *WEEKDAY*: "Sunday"/"Sun", "Monday"/"Mon", "Tue sday"/"Tue", "Wednesday"/"Wed", "Thursday"/"Thu", "Friday"/"Fri", "Saturday"/"Sat"

Return type:

- Returns TIMESTAMP if *DATE* is of the TIMESTAMP type.
- Returns DATE if *DATE* is of the DATE type.

Examples:

NEXT_DAY('2013-12-25','Saturday') returns '2013-12-28 00:00:00' which is the first Saturday after December 25, 2013.

NOW()

Purpose: Returns the current date and time (in the local time zone) as a TIMESTAMP value.

Return type: TIMESTAMP

Usage notes:

To find a date/time value in the future or the past relative to the current date and time, add or subtract an INTERVAL expression to the return value of NOW().

To produce a TIMESTAMP representing the current date and time that can be shared or stored without interoperability problems due to time zone differences, use the TO_UTC_TIMESTAMP() function and specify the time zone of the server. When TIMESTAMP data is stored in UTC form, any application that queries those values can convert them to the appropriate local time zone by calling the inverse function, FROM_UTC_TIMESTAMP().

To determine the time zone of the server you are connected to, you can call the timeofday() function, which includes the time zone specifier in its return value. Remember that with cloud computing, the server you interact with might be in a different time zone than you are, or different sessions might connect to servers in different time zones, or a cluster might include servers in more than one time zone.

Any references to the NOW() function are evaluated at the start of a query. All calls to NOW() within the same query return the same value, and the value does not depend on how long the query takes.

QUARTER(TIMESTAMP / DATE date)

Purpose: Returns the quarter in the input *DATE* argument as an integer value, 1, 2, 3, or 4, where 1 represents January 1 through March 31.

Return type: INT

SECOND(TIMESTAMP date)

Purpose: Returns the second field from a TIMESTAMP value.

Return type: INT

SECONDS_ADD(TIMESTAMP date, INT seconds), SECONDS_ADD(TIMESTAMP date, BIGINT seconds)

Purpose: Returns the specified date and time plus some number of seconds.

Return type: TIMESTAMP

SECONDS_SUB(TIMESTAMP date, INT seconds), SECONDS_SUB(TIMESTAMP date, BIGINT seconds)

Purpose: Returns the specified date and time minus some number of seconds.

Return type: TIMESTAMP

SUBDATE(TIMESTAMP / DATE date, INT / BIGINT days)

Purpose: Subtracts *DAYS* from *DATE* and returns the new date value.

The *DAYS* value can be negative, which gives the same result as the ADDDATE() function.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

TIMEOFDAY()

Purpose: Returns a string representation of the current date and time, according to the time of the local system, including any time zone designation.

Return type: STRING

Usage notes:

The result value represents similar information as the now() function, only as a STRING type and with somewhat different formatting. For example, the day of the week and the time zone identifier are included. This function is intended primarily for compatibility with SQL code from other systems that also have a timeofday() function. Prefer to use now() if practical for any new Impala code.

TIMESTAMP_CMP(TIMESTAMP t1, TIMESTAMP t2)

Purpose: Tests if one TIMESTAMP value is newer than, older than, or identical to another TIME STAMP

- If the first argument represents a later point in time than the second argument, the result is 1.
- If the first argument represents an earlier point in time than the second argument, the result is -1.
- If the first and second arguments represent identical points in time, the result is 0.
- If either argument is NULL, the result is NULL.

Return type: INT (either -1, 0, 1, or NULL)

Usage notes:

A comparison function for TIMESTAMP values that only tests whether the date and time increases, decreases, or stays the same. Similar to the SIGN() function for numeric values.

TO_DATE(TIMESTAMP ts)

Purpose: Returns a string representation of the date field from the *TS* argument.

Return type: STRING

TO_TIMESTAMP(BIGINT unixtime), TO_TIMESTAMP(STRING date, STRING pattern)

Purpose: Converts an integer or string representing a date/time value into the corresponding TIME STAMP value.

Return type: TIMESTAMP

Usage notes:

An integer argument represents the number of seconds past the epoch (midnight on January 1, 1970). It is the converse of the UNIX_TIMESTAMP() function, which produces a BIGINT representing the number of seconds past the epoch.

A string argument, plus another string argument representing the pattern, turns an arbitrary string representation of a date and time into a true TIMESTAMP value. The ability to parse many kinds of date and time formats allows you to deal with temporal data from diverse sources, and if desired to convert to efficient TIMESTAMP values during your ETL process. Using TIMESTAMP directly in queries and expressions lets you perform date and time calculations without the overhead of extra function calls and conversions each time you reference the applicable columns.

Examples:

The following examples demonstrate how to convert an arbitrary string representation to TIME STAMP based on a pattern string:

```
select to_timestamp('Sep 25, 1984', 'MMM dd, yyyy');
+-----------------------------------------------+
| to_timestamp('sep 25, 1984', 'mmm dd, yyyy') |
+-----------------------------------------------+
| 1984-09-25 00:00:00                           |
+-----------------------------------------------+

select to_timestamp('1984/09/25', 'yyyy/MM/dd');
+-----------------------------------------+
| to_timestamp('1984/09/25', 'yyyy/mm/dd') |
+-----------------------------------------+
| 1984-09-25 00:00:00                     |
+-----------------------------------------+
```

The following examples show how to convert a BIGINT representing seconds past epoch into a TIMESTAMP value:

```
-- One day past the epoch.
select to_timestamp(24 * 60 * 60);
+-----------------------------+
| to_timestamp(24 * 60 * 60) |
+-----------------------------+
| 1970-01-02 00:00:00        |
+-----------------------------+

-- 60 seconds in the past.
select now() as 'current date/time',
  unix_timestamp(now()) 'now in seconds',
  to_timestamp(unix_timestamp(now()) - 60) as '60 seconds ago';
+-------------------------------+----------------+-------------
--------+
| current date/time             | now in seconds | 60 seconds
ago      |
+-------------------------------+----------------+---------------
------+
| 2017-10-01 22:03:46.885624000 | 1506895426     | 2017-10-01 22
:02:46 |
+-------------------------------+----------------+-------------
--------+
```

TO_UTC_TIMESTAMP(TIMESTAMP ts, STRING timezone)

Purpose: Converts a specified timestamp value in a specified time zone into the corresponding value for the UTC time zone.

Return type: TIMESTAMP

Usage notes:

Often used in combination with the now() function, to translate local date and time values to the UTC time zone for consistent representation on disk. The opposite of the FROM_UTC_TIMESTA MP() function.

See discussion of time zones in TIMESTAMP data type on page 45 for information about using this function for conversions between the local time zone and UTC.

Examples:

The simplest use of this function is to turn a local date/time value to one with the standardized UTC time zone. Because the time zone specifier is not saved as part of the Impala TIMESTAMP value, all applications that refer to such data must agree in advance which time zone the values represent. If different parts of the ETL cycle, or different instances of the application, occur in different time zones, the ideal reference point is to convert all TIMESTAMP values to UTC for storage.

```
select now() as 'Current time in California USA',
  to_utc_timestamp(now(), 'PDT') as 'Current time in Greenwich UK
';
+--------------------------------+------------------------------
+
| current time in california usa | current time in greenwich uk
 |
+--------------------------------+------------------------------
+
| 2016-06-01 15:52:08.980072000  | 2016-06-01 22:52:08.980072000
 |
+--------------------------------+------------------------------
+
```

Once a value is converted to the UTC time zone by TO_UTC_TIMESTAMP(), it can be converted back to the local time zone with FROM_UTC_TIMESTAMP(). You can combine these functions using different time zone identifiers to convert a TIMESTAMP between any two time zones. This example starts with a TIMESTAMP value representing Pacific Daylight Time, converts it to UTC, and converts it to the equivalent value in Eastern Daylight Time.

```
select now() as 'Current time in California USA',
  from_utc_timestamp
  (
    to_utc_timestamp(now(), 'PDT'),
    'EDT'
  ) as 'Current time in New York, USA';
+--------------------------------+------------------------------
+
| current time in california usa | current time in new york, usa
 |
+--------------------------------+------------------------------
+
| 2016-06-01 18:14:12.743658000  | 2016-06-01 21:14:12.743658000
|
+--------------------------------+------------------------------
+
```

TRUNC(TIMESTAMP / DATE ts, STRING unit)

Purpose: Returns the *TS* truncated to the *UNIT* specified.

Argument: The *UNIT* argument is not case-sensitive. This argument string can be one of:

Unit	Supported for TIMESTAMP ts	Supported for DATE ts	Description
'SYYYY' 'YYYY' 'YEAR' 'SYEAR' 'YYY' 'YY' 'Y'	Yes	Yes	Year
'Q'	Yes	Yes	Quarter
'MONTH' 'MON' 'MM' 'RM'	Yes	Yes	Month
'WW'	Yes	Yes	The most recent date that is the same day of the week as the first day of the year
'W'	Yes	Yes	The most recent date that is the same day of the week as the first day of the month
'DDD' 'DD' 'J'	Yes	Yes	Day
'DAY' 'DY' 'D'	Yes	Yes	Starting day of the week (Monday)
'HH' 'HH12' 'HH24'	Yes	No	Hour. A TIMESTAMP value truncated to the hour is always represented in 24-hour notation, even for the HH12 argument string.
'MI'	Yes	No	Minute

Usage notes:

The TIMESTAMP form is typically used in GROUP BY queries to aggregate results from the same hour, day, week, month, quarter, and so on. You can also use this function in an INSERT ... S ELECT into a partitioned table to divide TIMESTAMP values into the correct partition.

Return type:

- TIMESTAMP if the first argument, *TS*, is TIMESTAMP.
- DATE if the first argument, *TS*, is DATE.

Example:

TRUNC(DATE'2019-05-08','YEAR') returns 2019-01-01.

TRUNC(DATE'2019-05-08', 'QUARTER') returns 2019-04-01.

UNIX_TIMESTAMP(), UNIX_TIMESTAMP(STRING datetime), UNIX_TIMESTAMP(STRING datetime, STRING pattern), UNIX_TIMESTAMP(TIMESTAMP datetime)

Purpose: Returns a Unix time, which is a number of seconds elapsed since '1970-01-01 00:00:00' UTC. If called with no argument, the current date and time is converted to its Unix time. If called with arguments, the first argument represented as the TIMESTAMP or STRING is converted to its Unix time.

Return type: BIGINT

Usage notes:

See FROM_UNIXTIME() for details about the patterns you can use in the *PATTERN* string to represent the position of year, month, day, and so on in the date string. In Impala 1.3 and higher, you have more flexibility to switch the positions of elements and use different separator characters.

You can include a trailing uppercase Z qualifier to indicate “Zulu” time, a synonym for UTC.

You can include a timezone offset specified as minutes and hours, provided you also specify the details in the *PATTERN* string argument. The offset is specified in the *PATTERN* string as a plus or minus sign followed by hh:mm, hhmm, or hh. The hh must be lowercase, to distinguish it from the HH represent hours in the actual time value. Currently, only numeric timezone offsets are allowed, not symbolic names.

Built-in functions that accept or return integers representing TIMESTAMP values use the BIGINT type for parameters and return values, rather than INT. This change lets the date and time functions avoid an overflow error that would otherwise occur on January 19th, 2038 (known as the “Year 2038 problem” or “Y2K38 problem”). This change affects the FROM_UNIXTIME() and UNIX_TIMESTAMP() functions. You might need to change application code that interacts with these functions, change the types of columns that store the return values, or add CAST() calls to SQL statements that call these functions.

UNIX_TIMESTAMP() and FROM_UNIXTIME() are often used in combination to convert a TIMESTAMP value into a particular string format. For example, FROM_UNIXTIME(UNIX_TIMESTAMP(NOW() + interval 3 days), 'yyyy/MM/dd HH:mm').

The way this function deals with time zones when converting to or from TIMESTAMP values is affected by the ##use_local_tz_for_unix_timestamp_conversions startup flag for the `impalad` daemon. See TIMESTAMP data type on page 45 for details about how Impala handles time zone considerations for the TIMESTAMP data type.

Examples:

The following examples show different ways of turning the same date and time into an integer value. A *PATTERN* string that Impala recognizes by default is interpreted as a UTC date and time. The trailing Z is a confirmation that the timezone is UTC. If the date and time string is formatted differently, a second argument specifies the position and units for each of the date and time values.

The final two examples show how to specify a timezone offset of Pacific Daylight Saving Time, which is 7 hours earlier than UTC. You can use the numeric offset -07:00 and the equivalent suffix of -hh:mm in the *PATTERN* string, or specify the mnemonic name for the time zone in a call to TO_UTC_TIMESTAMP(). This particular date and time expressed in PDT translates to a different number than the same date and time expressed in UTC.

```
-- 3 ways of expressing the same date/time in UTC and converting
 to an integer.

select unix_timestamp('2015-05-15 12:00:00');
+----------------------------------------+
| unix_timestamp('2015-05-15 12:00:00') |
+----------------------------------------+
| 1431691200                             |
+----------------------------------------+

select unix_timestamp('2015-05-15 12:00:00Z');
```

```
+----------------------------------------+
| unix_timestamp('2015-05-15 12:00:00z') |
+----------------------------------------+
| 1431691200                             |
+----------------------------------------+

select unix_timestamp
(
  'May 15, 2015 12:00:00',
  'MMM dd, yyyy HH:mm:ss'
) as may_15_month_day_year;
+------------------------+
| may_15_month_day_year |
+------------------------+
| 1431691200            |
+------------------------+

-- 2 ways of expressing the same date and time but in a differ
ent timezone.
-- The resulting integer is different from the previous examples.
select unix_timestamp
(
  '2015-05-15 12:00:00-07:00',
  'yyyy-MM-dd HH:mm:ss-hh:mm'
) as may_15_year_month_day;
+------------------------+
| may_15_year_month_day |
+------------------------+
| 1431716400            |
+------------------------+
select unix_timestamp
  (to_utc_timestamp(
    '2015-05-15 12:00:00',
    'PDT')
  ) as may_15_pdt;
+------------+
| may_15_pdt |
+------------+
| 1431716400 |
+------------+
```

UTC_TIMESTAMP()

Purpose: Returns a TIMESTAMP corresponding to the current date and time in the UTC time zone.

Return type: TIMESTAMP

Examples:

Similar to the NOW() or CURRENT_TIMESTAMP() functions, but does not use the local time zone as those functions do. Use UTC_TIMESTAMP() to record TIMESTAMP values that are interoperable with servers around the world, in arbitrary time zones, without the need for additional conversion functions to standardize the time zone of each value representing a date/time.

For working with date/time values represented as integer values, you can convert back and forth between TIMESTAMP and BIGINT with the UNIX_MICROS_TO_UTC_TIMESTAMP() and UTC_TO_UNIX_MICROS() functions. The integer values represent the number of microseconds since the Unix epoch (midnight on January 1, 1970).

Examples:

The following example shows how NOW() and CURRENT_TIMESTAMP() represent the current date/time in the local time zone (in this case, UTC-7), while utc_timestamp() represents the same date/time in the standardized UTC time zone:

```
select now(), utc_timestamp();
+-------------------------------+-------------------------------+
| now()                         | utc_timestamp()               |
+-------------------------------+-------------------------------+
| 2017-10-01 23:33:58.919688000 | 2017-10-02 06:33:58.919688000 |
+-------------------------------+-------------------------------+

select current_timestamp(), utc_timestamp();
+-------------------------------+-------------------------------+
| current_timestamp()           | utc_timestamp()               |
+-------------------------------+-------------------------------+
| 2017-10-01 23:34:07.400642000 | 2017-10-02 06:34:07.400642000 |
+-------------------------------+-------------------------------+
```

WEEK(TIMESTAMP / DATE date), WEEKOFYEAR(TIMESTAMP / DATE date)

Purpose: Returns the corresponding week (1-53) from the *DATE* argument.

Return type: INT

WEEKS_ADD(TIMESTAMP / DATE date, INT / BIGINT weeks)

Purpose: Returns the value with the number of *WEEKS* added to *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

WEEKS_SUB(TIMESTAMP / DATE date, INT / BIGINT weeks)

Purpose: Returns the value with the number of *WEEKS* subtracted from *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

YEAR(TIMESTAMP / DATE date)

Purpose: Returns the year field from the *DATE* argument.

Return type: INT

YEARS_ADD(TIMESTAMP / DATE date, INT / BIGINT years)

Purpose: Returns the value with the number of *YEARS* added to *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

Usage notes:

If the equivalent date does not exist in the year of the result due to a leap year, the date is changed to the last day of the appropriate month.

YEARS_SUB(TIMESTAMP / DATE date, INT / BIGINT years)

Purpose: Returns the value with the number of *YEARS* subtracted from *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.

- If *DATE* is DATE, returns DATE.

Usage notes:

If the equivalent date does not exist in the year of the result due to a leap year, the date is changed to the last day of the appropriate month.

Impala conditional functions

Impala supports the following conditional functions for testing equality, comparison operators, and nullity.

- CASE
- CASE2
- COALESCE
- DECODE
- IF
- IFNULL
- ISFALSE
- ISNOTFALSE
- ISNOTTRUE
- ISNULL
- ISTRUE
- NONNULLVALUE
- NULLIF
- NULLIFZERO
- NULLVALUE
- NVL
- NVL2
- ZEROIFNULL

CASE a WHEN b THEN c [WHEN d THEN e]... [ELSE f] END

Purpose: Compares an expression to one or more possible values, and returns a corresponding result when a match is found.

Return type: same as the initial argument value, except that integer values are promoted to BIGI NT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

Usage notes:

In this form of the CASE expression, the initial value A being evaluated for each row it typically a column reference, or an expression involving a column. This form can only compare against a set of specified values, not ranges, multi-value comparisons such as BETWEEN or IN, regular expressions, or NULL.

Examples:

Although this example is split across multiple lines, you can put any or all parts of a CASE expression on a single line, with no punctuation or other separators between the WHEN, ELSE, and END clauses.

```
select case x
    when 1 then 'one'
    when 2 then 'two'
    when 0 then 'zero'
    else 'out of range'
  end
```

```
    from t1;
```

CASE WHEN a THEN b [WHEN c THEN d]... [ELSE e] END

Purpose: Tests whether any of a sequence of expressions is TRUE, and returns a corresponding result for the first TRUE expression.

Return type: same as the initial argument value, except that integer values are promoted to BIGI NT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

Usage notes:

CASE expressions without an initial test value have more flexibility. For example, they can test different columns in different WHEN clauses, or use comparison operators such as BETWEEN, IN and IS NULL rather than comparing against discrete values.

CASE expressions are often the foundation of long queries that summarize and format results for easy-to-read reports. For example, you might use a CASE function call to turn values from a numeric column into category strings corresponding to integer values, or labels such as “Small”, “Medium” and “Large” based on ranges. Then subsequent parts of the query might aggregate based on the transformed values, such as how many values are classified as small, medium, or large. You can also use CASE to signal problems with out-of-bounds values, NULL values, and so on.

By using operators such as OR, IN, REGEXP, and so on in CASE expressions, you can build extensive tests and transformations into a single query. Therefore, applications that construct SQL statements often rely heavily on CASE calls in the generated SQL code.

Because this flexible form of the CASE expressions allows you to perform many comparisons and call multiple functions when evaluating each row, be careful applying elaborate CASE expressions to queries that process large amounts of data. For example, when practical, evaluate and transform values through CASE after applying operations such as aggregations that reduce the size of the result set; transform numbers to strings after performing joins with the original numeric values.

Examples:

Although this example is split across multiple lines, you can put any or all parts of a CASE expression on a single line, with no punctuation or other separators between the WHEN, ELSE, and END clauses.

```
select case
    when dayname(now()) in ('Saturday','Sunday') then 'result un
defined on weekends'
    when x > y then 'x greater than y'
    when x = y then 'x and y are equal'
    when x is null or y is null then 'one of the columns is null'
    else null
  end
    from t1;
```

COALESCE(type v1, type v2, ...)

Purpose: Returns the first specified argument that is not NULL, or NULL if all arguments are NULL.

Return type: same as the initial argument value, except that integer values are promoted to BIGI NT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

DECODE(type expression, type search1, type result1 [, type search2, type result2 ...] [, type default])

Purpose: Compares the first argument, expression, to the search expressions using the IS NOT D ISTINCT operator, and returns:

- The corresponding result when a match is found.

- The first corresponding result if there are more than one matching search expressions.
- The default expression if none of the search expressions matches the first argument expression.
- NULL if the final default expression is omitted and none of the search expressions matches the first argument.

Return type: Same as the first argument with the following exceptions:

- Integer values are promoted to BIGINT.
- Floating-point values are promoted to DOUBLE.
- Use CAST() when inserting into a smaller numeric column.

Usage notes:

- Can be used as shorthand for a CASE expression.
- The first argument, expression, and the search expressions must be of the same type or convertible types.
- The result expression can be a different type, but all result expressions must be of the same type.
- Returns a successful match if the first argument is NULL and a search expression is also NULL.
- NULL can be used as a search expression.

Examples:

The following example translates numeric day values into weekday names, such as 1 to Monday, 2 to Tuesday, etc.

```
SELECT event, DECODE(day_of_week, 1, "Monday", 2, "Tuesday", 3,
"Wednesday",
  4, "Thursday", 5, "Friday", 6, "Saturday", 7, "Sunday", "Unkn
own day")
  FROM calendar;
```

IF(BOOLEAN condition, type ifTrue, type ifFalseOrNull)

Purpose: Tests an expression and returns a corresponding result depending on whether the result is TRUE, FALSE, or NULL.

Return type: Same as the ifTrue argument value

IFNULL(type a, type ifNull)

Purpose: Alias for the ISNULL() function, with the same behavior. To simplify porting SQL with vendor extensions to Impala.

Added in: Impala 1.3.0

ISFALSE(BOOLEAN expression)

Purpose: Returns TRUE if the expression is FALSE. Returns FALSE if the expression is TRUE or NULL.

Same as the IS FALSE operator.

Similar to ISNOTTRUE(), except it returns the opposite value for a NULL argument.

Return type: BOOLEAN

Usage notes:

In Impala 2.1.1 and higher, you can use the operators IS [NOT] TRUE and IS [NOT] FALSE as equivalents for the built-in functions ISTRUE(), ISNOTTRUE(), ISFALSE(), and ISNOTFALSE().

ISNOTFALSE(BOOLEAN expression)

Purpose: Tests if a Boolean expression is not FALSE (that is, either TRUE or NULL). Returns TRUE if so. If the argument is NULL, returns TRUE.

Same as the IS NOT FALSE operator.

Similar to ISTRUE(), except it returns the opposite value for a NULL argument.

Return type: BOOLEAN

Usage notes: Primarily for compatibility with code containing industry extensions to SQL.

Usage notes:

In Impala 2.1.1 and higher, you can use the operators IS [NOT] TRUE and IS [NOT] FALSE as equivalents for the built-in functions ISTRUE(), ISNOTTRUE(), ISFALSE(), and ISNOTFALSE().

ISNOTTRUE(BOOLEAN expression)

Purpose: Tests if a Boolean expression is not TRUE (that is, either FALSE or NULL). Returns TRUE if so. If the argument is NULL, returns TRUE.

Same as the IS NOT TRUE operator.

Similar to ISFALSE(), except it returns the opposite value for a NULL argument.

Return type: BOOLEAN

Usage notes:

In Impala 2.1.1 and higher, you can use the operators IS [NOT] TRUE and IS [NOT] FALSE as equivalents for the built-in functions ISTRUE(), ISNOTTRUE(), ISFALSE(), and ISNOTFALSE().

ISNULL(type a, type ifNull)

Purpose: Tests if an expression is NULL, and returns the expression result value if not. If the first argument is NULL, returns the second argument.

Compatibility notes: Equivalent to the NVL() function from Oracle Database or IFNULL() from MySQL. The NVL() and IFNULL() functions are also available in Impala.

Return type: Same as the first argument value

ISTRUE(BOOLEAN expression)

Purpose: Returns TRUE if the expression is TRUE. Returns FALSE if the expression is FALSE or NULL.

Same as the IS TRUE operator.

Similar to ISNOTFALSE(), except it returns the opposite value for a NULL argument.

Return type: BOOLEAN

Usage notes: Primarily for compatibility with code containing industry extensions to SQL.

Usage notes:

In Impala 2.1.1 and higher, you can use the operators IS [NOT] TRUE and IS [NOT] FALSE as equivalents for the built-in functions ISTRUE(), ISNOTTRUE(), ISFALSE(), and ISNOTFALSE().

NONNULLVALUE(type expression)

Purpose: Tests if an expression (of any type) is NULL or not. Returns FALSE if so. The converse of nullvalue().

Return type: BOOLEAN

Usage notes: Primarily for compatibility with code containing industry extensions to SQL.

NULLIF(expr1, expr2)

Purpose: Returns NULL if the two specified arguments are equal. If the specified arguments are not equal, returns the value of *EXPR1*. The data types of the expressions must be compatible, according to the conversion rules from Impala SQL data types on page 13. You cannot use an expression that evaluates to NULL for *EXPR1*; that way, you can distinguish a return value of NULL from an argument value of NULL, which would never match *EXPR2*.

Usage notes: This function is effectively shorthand for a CASE expression of the form:

```
CASE
  WHEN EXPR1 = EXPR2 THEN NULL
  ELSE EXPR1
END
```

It is commonly used in division expressions, to produce a NULL result instead of a divide-by-zero error when the divisor is equal to zero:

```
select 1.0 / nullif(c1,0) as reciprocal from t1;
```

You might also use it for compatibility with other database systems that support the same NULLIF() function.

Return type: same as the initial argument value, except that integer values are promoted to BIGINT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

NULLIFZERO(numeric_expr)

Purpose: Returns NULL if the numeric expression evaluates to 0, otherwise returns the result of the expression.

Usage notes: Used to avoid error conditions such as divide-by-zero in numeric calculations. Serves as shorthand for a more elaborate CASE expression, to simplify porting SQL with vendor extensions to Impala.

Return type: Same type as the input argument

NULLVALUE(expression)

Purpose: Tests if an expression (of any type) is NULL or not. Returns TRUE if so. The converse of nonnullvalue().

Return type: BOOLEAN

Usage notes: Primarily for compatibility with code containing industry extensions to SQL.

NVL(type a, type ifNull)

Purpose: Alias for the ISNULL() function. Returns the first argument if the first argument is not NULL. Returns the second argument if the first argument is NULL.

Equivalent to the NVL() function in Oracle Database or IFNULL() in MySQL.

Return type: Same as the first argument value

NVL2(type a, type ifNotNull, type ifNull)

Purpose: Returns the second argument, ifNotNull, if the first argument is not NULL. Returns the third argument, ifNull, if the first argument is NULL.

Equivalent to the NVL2() function in Oracle Database.

Return type: Same as the first argument value

Examples:

```
SELECT NVL2(NULL, 999, 0); -- Returns 0
SELECT NVL2('ABC', 'Is Not Null', 'Is Null'); -- Returns 'Is Not
 Null'
```

ZEROIFNULL(numeric_expr)

Purpose: Returns 0 if the numeric expression evaluates to NULL, otherwise returns the result of the expression.

Usage notes: Used to avoid unexpected results due to unexpected propagation of NULL values in numeric calculations. Serves as shorthand for a more elaborate CASE expression, to simplify porting SQL with vendor extensions to Impala.

Return type: Same type as the input argument

Impala string functions

String functions are classified as those primarily accepting or returning STRING, VARCHAR, or CHAR data types, for example to measure the length of a string or concatenate two strings together.

List of supported string functions

- All the functions that accept STRING arguments also accept the VARCHAR and CHAR types introduced in Impala 2.0.
- Whenever VARCHAR or CHAR values are passed to a function that returns a string value, the return type is normalized to STRING. For example, a call to concat() with a mix of STRING, VARCHAR, and CHAR arguments produces a STRING result.

Function reference:

Impala supports the following string functions:

- ASCII
- BASE64DECODE
- BASE64ENCODE
- BTRIM
- BYTES
- CHAR_LENGTH
- CHR
- CONCAT
- CONCAT_WS
- FIND_IN_SET
- GROUP_CONCAT
- INITCAP
- INSTR
- JARO_DISTANCE, JARO_DIST
- JARO_SIMILARITY, JARO_SIM
- JARO_WINKER_DISTANCE, JW_DST
- JARO_WINKER_SIMILARITY, JW_SIM
- LEFT
- LENGTH
- LEVENSHTEIN, LE_DST
- LOCATE
- LOWER, LCASE
- LPAD
- LTRI
- PARSE_URL
- REGEXP_ESCAPE
- REGEXP_EXTRACT
- REGEXP_LIKE
- REGEXP_REPLACE
- REPEAT
- REPLACE

- REVERSE
- RIGHT
- RPAD
- RTRIM
- SPACE
- SPLIT_PART
- STRLEFT
- STRRIGHT
- SUBSTR, SUBSTRING
- TRANSLATE
- TRIM
- UPPER, UCASE

ASCII(STRING str)

Purpose: Returns the numeric ASCII code of the first character of the argument.

Return type: INT

BASE64DECODE(STRING str)

Purpose:

Return type: STRING

Usage notes:

The functions BASE64ENCODE() and BASE64DECODE() are typically used in combination, to store in an Impala table string data that is problematic to store or transmit. For example, you could use these functions to store string data that uses an encoding other than UTF-8, or to transform the values in contexts that require ASCII values, such as for partition key columns. Keep in mind that base64-encoded values produce different results for string functions such as LENGTH(), MAX(), and MIN() than when those functions are called with the unencoded string values.

The set of characters that can be generated as output from BASE64ENCODE(), or specified in the argument string to BASE64DECODE(), are the ASCII uppercase and lowercase letters (A-Z, a-z), digits (0-9), and the punctuation characters +, /, and =.

All return values produced by BASE64ENCODE() are a multiple of 4 bytes in length. All argument values supplied to BASE64DECODE() must also be a multiple of 4 bytes in length. If a base64-encoded value would otherwise have a different length, it can be padded with trailing = characters to reach a length that is a multiple of 4 bytes.

If the argument string to BASE64DECODE() does not represent a valid base64-encoded value, subject to the constraints of the Impala implementation such as the allowed character set, the function returns NULL.

Examples:

The following examples show how to use BASE64ENCODE() and BASE64DECODE() together to store and retrieve string values:

```
-- An arbitrary string can be encoded in base 64.
-- The length of the output is a multiple of 4 bytes,
-- padded with trailing = characters if necessary.
select base64encode('hello world') as encoded,
  length(base64encode('hello world')) as length;
+------------------+--------+
| encoded          | length |
+------------------+--------+
| aGVsbG8gd29ybGQ= | 16     |
+------------------+--------+
```

```
-- Passing an encoded value to base64decode() produces
-- the original value.
select base64decode('aGVsbG8gd29ybGQ=') as decoded;
+-------------+
| decoded     |
+-------------+
| hello world |
+-------------+
```

These examples demonstrate incorrect encoded values that produce NULL return values when decoded:

```
-- The input value to base64decode() must be a multiple of 4 by
tes.
-- In this case, leaving off the trailing = padding character
-- produces a NULL return value.
select base64decode('aGVsbG8gd29ybGQ') as decoded;
+---------+
| decoded |
+---------+
| NULL    |
+---------+
WARNINGS: UDF WARNING: Invalid base64 string; input length is 15,
  which is not a multiple of 4.

-- The input to base64decode() can only contain certain charac
ters.
-- The $ character in this case causes a NULL return value.
select base64decode('abc$');
+-----------------------+
| base64decode('abc$') |
+-----------------------+
| NULL                  |
+-----------------------+
WARNINGS: UDF WARNING: Could not base64 decode input in space 4;
 actual output length 0
```

These examples demonstrate “round-tripping” of an original string to an encoded string, and back again. This technique is applicable if the original source is in an unknown encoding, or if some intermediate processing stage might cause national characters to be misrepresented:

```
select 'circumflex accents: â, ê, î, ô, û' as original,
  base64encode('circumflex accents: â, ê, î, ô, û') as encoded;
+-----------------------------------+-------------------------
----------------------------+
| original                          | encoded
                            |
+-----------------------------------+----------------------------
--------------------------+
| circumflex accents: â, ê, î, ô, û | Y2lyY3VtZmxleCBhY2NlbnRzOi
DDoiwgw6osIMOuLCDDtCwgw7s= |
+-----------------------------------+-------------------------
----------------------------+

select base64encode('circumflex accents: â, ê, î, ô, û') as encod
ed,
  base64decode(base64encode('circumflex accents: â, ê, î, ô, û'))
 as decoded;
```

```
+-------------------------------------------------------+------
-----------------------------+
| encoded                                               | decoded
                          |
+-------------------------------------------------------+--------
--------------------------+
| Y2lyY3VtZmxleCBhY2NlbnRzOiDDoiwgw6osIMOuLCDDtCwgw7s= | circum
flex accents: â, ê, î, ô, û |
+-------------------------------------------------------+------
----------------------------+
```

BASE64ENCODE(STRING str)

Purpose:

Return type: STRING

Usage notes:

The functions BASE64ENCODE() and BASE64DECODE() are typically used in combination, to store in an Impala table string data that is problematic to store or transmit. For example, you could use these functions to store string data that uses an encoding other than UTF-8, or to transform the values in contexts that require ASCII values, such as for partition key columns. Keep in mind that base64-encoded values produce different results for string functions such as LENGTH(), MAX(), and MIN() than when those functions are called with the unencoded string values.

The set of characters that can be generated as output from BASE64ENCODE(), or specified in the argument string to BASE64DECODE(), are the ASCII uppercase and lowercase letters (A-Z, a-z), digits (0-9), and the punctuation characters +, /, and =.

All return values produced by BASE64ENCODE() are a multiple of 4 bytes in length. All argument values supplied to BASE64DECODE() must also be a multiple of 4 bytes in length. If a base64-encoded value would otherwise have a different length, it can be padded with trailing = characters to reach a length that is a multiple of 4 bytes.

Examples:

The following examples show how to use BASE64ENCODE() and BASE64DECODE() together to store and retrieve string values:

```
-- An arbitrary string can be encoded in base 64.
-- The length of the output is a multiple of 4 bytes,
-- padded with trailing = characters if necessary.
select base64encode('hello world') as encoded,
  length(base64encode('hello world')) as length;
+------------------+--------+
| encoded          | length |
+------------------+--------+
| aGVsbG8gd29ybGQ= | 16     |
+------------------+--------+

-- Passing an encoded value to base64decode() produces
-- the original value.
select base64decode('aGVsbG8gd29ybGQ=') as decoded;
+-------------+
| decoded     |
+-------------+
| hello world |
+-------------+
```

These examples demonstrate incorrect encoded values that produce NULL return values when decoded:

```
-- The input value to base64decode() must be a multiple of 4 by
tes.
-- In this case, leaving off the trailing = padding character
-- produces a NULL return value.
select base64decode('aGVsbG8gd29ybGQ') as decoded;
+---------+
| decoded |
+---------+
| NULL    |
+---------+
WARNINGS: UDF WARNING: Invalid base64 string; input length is 15,
  which is not a multiple of 4.

-- The input to base64decode() can only contain certain charac
ters.
-- The $ character in this case causes a NULL return value.
select base64decode('abc$');
+----------------------+
| base64decode('abc$') |
+----------------------+
| NULL                 |
+----------------------+
WARNINGS: UDF WARNING: Could not base64 decode input in space 4;
 actual output length 0
```

These examples demonstrate “round-tripping” of an original string to an encoded string, and back again. This technique is applicable if the original source is in an unknown encoding, or if some intermediate processing stage might cause national characters to be misrepresented:

```
select 'circumflex accents: â, ê, î, ô, û' as original,
  base64encode('circumflex accents: â, ê, î, ô, û') as encoded;
+----------------------------------+-------------------------
----------------------------+
| original                         | encoded
                            |
+----------------------------------+----------------------------
--------------------------+
| circumflex accents: â, ê, î, ô, û | Y2lyY3VtZmxleCBhY2NlbnRzOi
DDoiwgw6osIMOuLCDDtCwgw7s= |
+----------------------------------+-------------------------
----------------------------+

select base64encode('circumflex accents: â, ê, î, ô, û') as encod
ed,
  base64decode(base64encode('circumflex accents: â, ê, î, ô, û'))
 as decoded;
+-------------------------------------------------------+------
----------------------------+
| encoded                                               | decoded
                          |
+-------------------------------------------------------+--------
--------------------------+
| Y2lyY3VtZmxleCBhY2NlbnRzOiDDoiwgw6osIMOuLCDDtCwgw7s= | circum
flex accents: â, ê, î, ô, û |
+-------------------------------------------------------+------
----------------------------+
```

BTRIM(STRING a), BTRIM(STRING a, STRING chars_to_trim)

Purpose: Removes all instances of one or more characters from the start and end of a STRING value. By default, removes only spaces. If a non-NULL optional second argument is specified, the function removes all occurrences of characters in that second argument from the beginning and end of the string.

Return type: STRING

Examples:

The following examples show the default btrim() behavior, and what changes when you specify the optional second argument. All the examples bracket the output value with [ ] so that you can see any leading or trailing spaces in the btrim() result. By default, the function removes and number of both leading and trailing spaces. When the second argument is specified, any number of occurrences of any character in the second argument are removed from the start and end of the input string; in this case, spaces are not removed (unless they are part of the second argument) and any instances of the characters are not removed if they do not come right at the beginning or end of the string.

```
-- Remove multiple spaces before and one space after.
select concat('[',btrim('    hello '),']');
+---------------------------------------+
| concat('[', btrim('    hello '), ']') |
+---------------------------------------+
| [hello]                               |
+---------------------------------------+
-- Remove any instances of x or y or z at beginning or end. Leave
 spaces alone.
select concat('[',btrim('xy    hello zyzzxx','xyz'),']');
+-------------------------------------------------------+
| concat('[', btrim('xy    hello zyzzxx', 'xyz'), ']') |
+-------------------------------------------------------+
| [    hello ]                                          |
+-------------------------------------------------------+
-- Remove any instances of x or y or z at beginning or end.
-- Leave x, y, z alone in the middle of the string.
select concat('[',btrim('xyhelxyzlozyzzxx','xyz'),']');
+-----------------------------------------------------+
| concat('[', btrim('xyhelxyzlozyzzxx', 'xyz'), ']') |
+-----------------------------------------------------+
| [helxyzlo]                                          |
+-----------------------------------------------------+
```

BYTES

Purpose: Returns the number of bytes contained in the specified byte string.

Syntax: BYTES (byte_expression)

Where:

byte_expression is the byte string to check the number of bytes.

Usage notes:

The BYTES function is similar to the LENGTH() function except that it always returns the number of bytes regardless of the status of UTF-8 mode, ON or OFF.

The following is the list of supported string data types allowed in byte_expression:

- STRING
- VARCHAR
- CHAR

Note: The BYTES (byte_expression) function counts the trailing zeros because trailing zero bytes are considered bytes. In the case of a fixed length column, the length of the value is always equal to the length defined for the column whereas the length of the value in a variable length column is always equal to the number of bytes, including any trailing zero bytes, contained in that value.

Example:

The following example obtains the number of bytes from “cloudera” by applying the BYTES function to the column “cloudera”, which is type VARCHAR.

```
SELECT bytes(cast('cloudera' as varchar(20))) FROM xyz;
            +----------------------------------------+
            | bytes(cast('cloudera' as varchar(20))) |
            +----------------------------------------+
            | 8                                      |
            +----------------------------------------+
```

Note: Use the TRIM function on the byte_expression to exclude the trailing blanks included in the byte count for a data value.

CHAR_LENGTH(STRING a), CHARACTER_LENGTH(STRING a)

Purpose: Returns the length in characters of the argument string. Aliases for the length() function.

Return type: INT

CHR(INT character_code)

Purpose: Returns a character specified by a decimal code point value. The interpretation and display of the resulting character depends on your system locale. Because consistent processing of Impala string values is only guaranteed for values within the ASCII range, only use this function for values corresponding to ASCII characters. In particular, parameter values greater than 255 return an empty string.

Return type: STRING

Usage notes: Can be used as the inverse of the ascii() function, which converts a character to its numeric ASCII code.

Examples:

```
SELECT chr(65);
+---------+
| chr(65) |
+---------+
| A       |
+---------+
SELECT chr(97);
+---------+
| chr(97) |
+---------+
| a       |
+---------+
```

CONCAT(STRING a, STRING b...)

Purpose: Returns a single string representing all the argument values joined together.

If any argument is NULL, the function returns NULL.

Return type: STRING

Usage notes: concat() and concat_ws() are appropriate for concatenating the values of multiple columns within the same row, while group_concat() joins together values from different rows.

CONCAT_WS(STRING sep, STRING a, STRING b...)

Purpose: Returns a single string representing the second and following argument values joined together, delimited by a specified separator.

Return type: STRING

Usage notes: concat() and concat_ws() are appropriate for concatenating the values of multiple columns within the same row, while group_concat() joins together values from different rows.

FIND_IN_SET(STRING str, STRING strList)

Purpose: Returns the position (starting from 1) of the first occurrence of a specified string within a comma-separated string. Returns NULL if either argument is NULL, 0 if the search string is not found, or 0 if the search string contains a comma.

Return type: INT

GROUP_CONCAT(STRING s [, STRING sep])

Purpose: Returns a single string representing the argument value concatenated together for each row of the result set. If the optional separator string is specified, the separator is added between each pair of concatenated values.

Return type: STRING

Usage notes: concat() and concat_ws() are appropriate for concatenating the values of multiple columns within the same row, while group_concat() joins together values from different rows.

By default, returns a single string covering the whole result set. To include other columns or values in the result set, or to produce multiple concatenated strings for subsets of rows, include a GROUP BY clause in the query.

Strictly speaking, group_concat() is an aggregate function, not a scalar function like the others in this list. For additional details and examples, see the GROUP_CONCAT function.

INITCAP(STRING str)

Purpose: Returns the input string with the first letter of each word capitalized and all other letters in lowercase.

Return type: STRING

Example:

INITCAP("i gOt mY ChiCkeNs in tHe yard.") returns "I Got My Chickens In The Yard.".

INSTR(STRING str, STRING substr [, BIGINT position [, BIGINT occurrence]])

Purpose: Returns the position (starting from 1) of the first occurrence of a substring within a longer string.

Return type: INT

Usage notes:

If the *SUBSTR* is not present in *STR*, the function returns 0.

The optional third and fourth arguments let you find instances of the *SUBSTR* other than the first instance starting from the left.

- The third argument, *POSITION*, lets you specify a starting point within the *STR* other than 1.

```
-- Restricting the search to positions 7..end,
-- the first occurrence of 'b' is at position 9.
select instr('foo bar bletch', 'b', 7);
+---------------------------------+
| instr('foo bar bletch', 'b', 7) |
+---------------------------------+
```

```
| 9                                |
+----------------------------------+
```

- If there are no more occurrences after the specified position, the result is 0.
- If *POSITION* is negative, the search works right-to-left starting that many characters from the right. The return value still represents the position starting from the left side of *STR*.

```
-- Scanning right to left, the first occurrence of 'o'
-- is at position 8. (8th character from the left.)
select instr('hello world','o',-1);
+-------------------------------+
| instr('hello world', 'o', -1) |
+-------------------------------+
| 8                             |
+-------------------------------+
```

- The fourth argument, *OCCURRENCE*, lets you specify an occurrence other than the first.

```
-- 2nd occurrence of 'b' is at position 9.
select instr('foo bar bletch', 'b', 1, 2);
+------------------------------------+
| instr('foo bar bletch', 'b', 1, 2) |
+------------------------------------+
| 9                                  |
+------------------------------------+
```

- If *OCCURRENCE* is greater than the number of matching occurrences, the function returns 0.
- *OCCURRENCE* cannot be negative or zero. A non-positive value for this argument causes an error.
- If either of the optional arguments, *POSITION* or *OCCURRENCE*, is NULL, the function also returns NULL.

JARO_DISTANCE(STRING str1, STRING str2)
JARO_DST(STRING str1, STRING str2)

Purpose: Returns the Jaro distance between two input strings. The Jaro distance is a measure of similarity between two strings and is the complementary of JARO_SIMILARITY(), i.e. (1 - JARO _SIMILARITY()).

Return type: DOUBLE

Usage notes:

If the two input strings are identical, the function returns 0.0.

If there is no matching character between the input strings, the function returns 1.0.

If either input strings is NULL, the function returns NULL.

If the length of either input string is bigger than 255 characters, the function returns an error.

JARO_SIMILARITY(STRING str1, STRING str2)
JARO_SIM(STRING str1, STRING str2)

Purpose: Returns the Jaro similarity of two strings. The higher the Jaro similarity for two strings is, the more similar the strings are.

Return type: DOUBLE

Usage notes:

If the two input strings are identical, the function returns 1.0.

If there is no matching character between the input strings, the function returns 0.0.

If either input strings is NULL, the function returns NULL.

If the length of either input string is bigger than 255 characters, the function returns an error.

JARO_WINKLER_DISTANCE(STRING str1, STRING str2[, DOUBLE scaling_factor, DOUBLE boost_threshold])
JW_DST(STRING str1, STRING str2[, DOUBLE scaling_factor, DOUBLE boost_threshold])

Purpose: Returns the Jaro-Winkler distance of two input strings. It is the complementary of JARO_WINKLER_SIMILARITY(), i.e. 1 - JARO_WINKLER_SIMILARITY().

Return type: DOUBLE

Usage notes:

If the two input strings are identical, the function returns 0.0.

If there is no matching character between the input strings, the function returns 1.0.

The function returns an error in the following cases:

- The length of either input string is bigger than 255 characters.
- *SCALING_FACTOR* < 0.0 or *SCALING_FACTOR* > 0.25
- *BOOST_THRESHOLD* < 0.0 or *BOOST_THRESHOLD* > 1.0

If either input strings is NULL, the function returns NULL.

The default *SCALING FACTOR* is 0.1.

The prefix weight will only be applied if the Jaro-distance exceeds the optional *BOOST_THRESHOLD*. By default, the *BOOST_THRESHOLD* value is 0.7.

JARO_WINKLER_SIMILARITY(STRING str1, STRING str2[, DOUBLE scaling_factor, DOUBLE boost_threshold])
JARO_SIM(STRING str1, STRING str2[, DOUBLE scaling_factor, DOUBLE boost_threshold])

Purpose: Returns the Jaro-Winkler Similarity between two input strings. The Jaro-Winkler similarity uses a prefix weight, specified by *SCALING FACTOR*, which gives more favorable ratings to strings that match from the beginning for a set prefix length, up to a maximum of four characters.

Use Jaro or Jaro-Winkler functions to perform fuzzy matches on relatively short strings, e.g. to scrub user inputs of names against the records in the database.

Return type: DOUBLE

Usage notes:

If the two input strings are identical, the function returns 1.0.

If there is no matching character between the input strings, the function returns 0.0.

The function returns an error in the following cases:

- The length of either input string is bigger than 255 characters.
- *SCALING_FACTOR* < 0.0 or *SCALING_FACTOR* > 0.25
- *BOOST_THRESHOLD* < 0.0 or *BOOST_THRESHOLD* > 1.0

If either input strings is NULL, the function returns NULL.

The default *SCALING FACTOR* is 0.1.

The prefix weight will only be applied if the Jaro-similarity exceeds the optional *BOOST_THRESHOLD*. By default, the *BOOST_THRESHOLD* value is 0.7.

LEFT(STRING a, INT num_chars)

See the STRLEFT() function.

LENGTH(STRING a)

Purpose: Returns the length in characters of the argument string.

Return type: INT

LEVENSHTEIN(STRING str1, STRING str2), LE_DST(STRING str1, STRING str2)

Purpose: Returns the Levenshtein distance between two input strings. The Levenshtein distance between two strings is the minimum number of single-character edits required to transform one string to other. The function indicates how different the input strings are.

Return type: INT

Usage notes:

If input strings are equal, the function returns 0.

If either input exceeds 255 characters, the function returns an error.

If either input string is NULL, the function returns NULL.

If the length of one input string is zero, the function returns the length of the other string.

Example:

LEVENSHTEIN ('welcome', 'We come') returns 2, first change to replace 'w' to 'W', and then to replace 'l' to a space character.

LOCATE(STRING substr, STRING str[, INT pos])

Purpose: Returns the position (starting from 1) of the first occurrence of a substring within a longer string, optionally after a particular position.

Return type: INT

LOWER(STRING a), LCASE(STRING a)

Purpose: Returns the argument string converted to all-lowercase.

Return type: STRING

Usage notes:

In Impala 2.5 and higher, you can simplify queries that use many UPPER() and LOWER() calls to do case-insensitive comparisons, by using the ILIKE or IREGEXP operators instead.

LPAD(STRING str, INT len, STRING pad)

Purpose: Returns a string of a specified length, based on the first argument string. If the specified string is too short, it is padded on the left with a repeating sequence of the characters from the pad string. If the specified string is too long, it is truncated on the right.

Return type: STRING

LTRIM(STRING a [, STRING chars_to_trim])

Purpose: Returns the argument string with all occurrences of characters specified by the second argument removed from the left side. Removes spaces if the second argument is not specified.

Return type: STRING

PARSE_URL(STRING urlString, STRING partToExtract [, STRING keyToExtract])

Purpose: Returns the portion of a URL corresponding to a specified part. The part argument can be 'PROTOCOL', 'HOST', 'PATH', 'REF', 'AUTHORITY', 'FILE', 'USERINFO', or 'QUERY'. Uppercase is required for these literal values. When requesting the QUERY portion of the URL, you can optionally specify a key to retrieve just the associated value from the key-value pairs in the query string.

Return type: STRING

Usage notes: This function is important for the traditional Hadoop use case of interpreting web logs. For example, if the web traffic data features raw URLs not divided into separate table columns, you

can count visitors to a particular page by extracting the 'PATH' or 'FILE' field, or analyze search terms by extracting the corresponding key from the 'QUERY' field.

REGEXP_ESCAPE(STRING source)

Purpose: The REGEXP_ESCAPE function returns a string escaped for the special character in RE2 library so that the special characters are interpreted literally rather than as special characters. The following special characters are escaped by the function:

```
.\+*?[^]$(){}=!<>|:-
```

Return type: string

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Because the `impala-shell` interpreter uses the \ character for escaping, use \\ to represent the regular expression escape character in any regular expressions that you submit through `impala-shell`. You might prefer to use the equivalent character class names, such as [[:digit:]] instead of \d which you would have to escape as \\d.

Examples:

This example shows escaping one of special characters in RE2.

```
+-------------------------------------------------------+
| regexp_escape('Hello.world')                          |
+-------------------------------------------------------+
| Hello\.world                                          |
+-------------------------------------------------------+
```

This example shows escaping all the special characters in RE2.

```
+-------------------------------------------------------------+
| regexp_escape('a.b\\c+d*e?f[g]h$i(j)k{l}m=n!o<p>q|r:s-t')  |
+-------------------------------------------------------------+
| a\.b\\c\+d\*e\?f\[g\]h\$i\(j\)k\{l\}m\=n\!o\<p\>q\|r\:s\-t |
+-------------------------------------------------------------+
```

REGEXP_EXTRACT(STRING subject, STRING pattern, INT index)

Purpose: Returns the specified () group from a string based on a regular expression pattern. Group 0 refers to the entire extracted string, while group 1, 2, and so on refers to the first, second, and so on (...) portion.

Return type: STRING

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Because the `impala-shell` interpreter uses the \ character for escaping, use \\ to represent the regular expression escape character in any regular expressions that you submit through `impala-shell`. You might prefer to use the equivalent character class names, such as [[:digit:]] instead of \d which you would have to escape as \\d.

Examples:

This example shows how group 0 matches the full pattern string, including the portion outside any () group:

```
[localhost:21000] > select regexp_extract('abcdef123ghi456jkl','
.*?(\\d+)',0);
+-------------------------------------------------------+
| regexp_extract('abcdef123ghi456jkl', '.*?(\\d+)', 0) |
+-------------------------------------------------------+
| abcdef123ghi456                                       |
+-------------------------------------------------------+
Returned 1 row(s) in 0.11s
```

This example shows how group 1 matches just the contents inside the first () group in the pattern string:

```
[localhost:21000] > select regexp_extract('abcdef123ghi456jkl','
.*?(\\d+)',1);
+-------------------------------------------------------+
| regexp_extract('abcdef123ghi456jkl', '.*?(\\d+)', 1) |
+-------------------------------------------------------+
| 456                                                   |
+-------------------------------------------------------+
Returned 1 row(s) in 0.11s
```

Unlike in earlier Impala releases, the regular expression library used in Impala 2.0 and later supports the .*? idiom for non-greedy matches. This example shows how a pattern string starting with .*? matches the shortest possible portion of the source string, returning the rightmost set of lowercase letters. A pattern string both starting and ending with .*? finds two potential matches of equal length, and returns the first one found (the leftmost set of lowercase letters).

```
[localhost:21000] > select regexp_extract('AbcdBCdefGHI','.*?([[
:lower:]]+)',1);
+-----------------------------------------------------------+
| regexp_extract('abcdbcdefghi', '.*?([[:lower:]]+)', 1) |
+-----------------------------------------------------------+
| def                                                       |
+-----------------------------------------------------------+
[localhost:21000] > select regexp_extract('AbcdBCdefGHI','.*?([[
:lower:]]+).*?',1);
+-------------------------------------------------------------+
| regexp_extract('abcdbcdefghi', '.*?([[:lower:]]+).*?', 1) |
+-------------------------------------------------------------+
| bcd                                                         |
+-------------------------------------------------------------+
```

REGEXP_LIKE(STRING source, STRING pattern[, STRING options])

Purpose: Returns true or false to indicate whether the source string contains anywhere inside it the regular expression given by the pattern. The optional third argument consists of letter flags that change how the match is performed, such as i for case-insensitive matching.

Syntax:

The flags that you can include in the optional third argument are:

- c: Case-sensitive matching (the default).

- i: Case-insensitive matching. If multiple instances of c and i are included in the third argument, the last such option takes precedence.
- m: Multi-line matching. The ^ and $ operators match the start or end of any line within the source string, not the start and end of the entire string.
- n: Newline matching. The . operator can match the newline character. A repetition operator such as .* can match a portion of the source string that spans multiple lines.

Return type: BOOLEAN

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Because the `impala-shell` interpreter uses the \ character for escaping, use \\ to represent the regular expression escape character in any regular expressions that you submit through `impala-shell`. You might prefer to use the equivalent character class names, such as [[:digit:]] instead of \d which you would have to escape as \\d.

Examples:

This example shows how REGEXP_LIKE() can test for the existence of various kinds of regular expression patterns within a source string:

```
-- Matches because the 'f' appears somewhere in 'foo'.
select regexp_like('foo','f');
+-------------------------+
| regexp_like('foo', 'f') |
+-------------------------+
| true                    |
+-------------------------+

-- Does not match because the comparison is case-sensitive by de
fault.
select regexp_like('foo','F');
+-------------------------+
| regexp_like('foo', 'f') |
+-------------------------+
| false                   |
+-------------------------+

-- The 3rd argument can change the matching logic, such as 'i' me
aning case-insensitive.
select regexp_like('foo','F','i');
+------------------------------+
| regexp_like('foo', 'f', 'i') |
+------------------------------+
| true                         |
+------------------------------+
-- The familiar regular expression notations work, such as ^ and
 $ anchors...
select regexp_like('foo','f$');
+--------------------------+
| regexp_like('foo', 'f$') |
+--------------------------+
| false                    |
+--------------------------+
select regexp_like('foo','o$');
```

```
+--------------------------+
| regexp_like('foo', 'o$') |
+--------------------------+
| true                     |
+--------------------------+

-- ...and repetition operators such as * and +
select regexp_like('foooooobar','fo+b');
+-----------------------------------+
| regexp_like('foooooobar', 'fo+b') |
+-----------------------------------+
| true                              |
+-----------------------------------+

select regexp_like('foooooobar','fx*y*o*b');
+---------------------------------------+
| regexp_like('foooooobar', 'fx*y*o*b') |
+---------------------------------------+
| true                                  |
+---------------------------------------+
```

REGEXP_REPLACE(STRING initial, STRING pattern, STRING replacement)

Purpose: Returns the initial argument with the regular expression pattern replaced by the final argument string.

Return type: STRING

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Because the `impala-shell` interpreter uses the \ character for escaping, use \\ to represent the regular expression escape character in any regular expressions that you submit through `impala-shell`. You might prefer to use the equivalent character class names, such as [[:digit:]] instead of \d which you would have to escape as \\d.

Examples:

These examples show how you can replace parts of a string matching a pattern with replacement text, which can include backreferences to any () groups in the pattern string. The backreference numbers start at 1, and any \ characters must be escaped as \\.

Replace a character pattern with new text:

```
[localhost:21000] > select regexp_replace('aaabbbaaa','b+','xyz'
);
+-----------------------------------------+
| regexp_replace('aaabbbaaa', 'b+', 'xyz') |
+-----------------------------------------+
| aaaxyzaaa                               |
+-----------------------------------------+
Returned 1 row(s) in 0.11s
```

Replace a character pattern with substitution text that includes the original matching text:

```
[localhost:21000] > select regexp_replace('aaabbbaaa','(b+)','<\
\1>');
```

```
+---------------------------------------------+
| regexp_replace('aaabbbaaa', '(b+)', '<\\1>') |
+---------------------------------------------+
| aaa<bbb>aaa                                 |
+---------------------------------------------+
Returned 1 row(s) in 0.11s
```

Remove all characters that are not digits:

```
[localhost:21000] > select regexp_replace('123-456-789','[^[:dig
it:]]','');
+----------------------------------------------------+
| regexp_replace('123-456-789', '[^[:digit:]]', '') |
+----------------------------------------------------+
| 123456789                                          |
+----------------------------------------------------+
Returned 1 row(s) in 0.12s
```

REPEAT(STRING str, INT n)

Purpose: Returns the argument string repeated a specified number of times.

Return type: STRING

REPLACE(STRING initial, STRING target, STRING replacement)

Purpose: Returns the initial argument with all occurrences of the target string replaced by the replacement string.

Return type: STRING

Usage notes:

Because this function does not use any regular expression patterns, it is typically faster than REGE XP_REPLACE() for simple string substitutions.

If any argument is NULL, the return value is NULL.

Matching is case-sensitive.

If the replacement string contains another instance of the target string, the expansion is only performed once, instead of applying again to the newly constructed string.

Examples:

```
-- Replace one string with another.
select replace('hello world','world','earth');
+-----------------------------------------+
| replace('hello world', 'world', 'earth') |
+-----------------------------------------+
| hello earth                             |
+-----------------------------------------+
-- All occurrences of the target string are replaced.
select replace('hello world','o','0');
+----------------------------------+
| replace('hello world', 'o', '0') |
+----------------------------------+
| hell0 w0rld                      |
+----------------------------------+

-- If no match is found, the original string is returned unchang
ed.
select replace('hello world','xyz','abc');
+--------------------------------------+
| replace('hello world', 'xyz', 'abc') |
+--------------------------------------+
```

```
| hello world                          |
+--------------------------------------+
```

REVERSE(STRING a)

Purpose: Returns the argument string with characters in reversed order.

Return type: STRING

RIGHT(STRING a, INT num_chars)

See the STRRIGHT function.

RPAD(STRING str, INT len, STRING pad)

Purpose: Returns a string of a specified length, based on the first argument string. If the specified string is too short, it is padded on the right with a repeating sequence of the characters from the pad string. If the specified string is too long, it is truncated on the right.

Return type: STRING

RTRIM(STRING a [, STRING chars_to_trim])

Purpose: Returns the argument string with all occurrences of characters specified by the second argument removed from the right side. Removes spaces if the second argument is not specified.

Return type: STRING

SPACE(INT n)

Purpose: Returns a concatenated string of the specified number of spaces. Shorthand for repeat(' ',*N*).

Return type: STRING

SPLIT_PART(STRING source, STRING delimiter, BIGINT index)

Purpose: Returns the requested indexth part of the input *SOURCE* string split by the *DELIMITER*.

- If *INDEX* is a positive number, returns the *INDEX*th part from the left within the *SOURCE* string.
- If *INDEX* is a negative number, returns the *INDEX*th part from the right within the *SOURCE* string.
- If *INDEX* is 0, returns an error.

The *DELIMITER* can consist of multiple characters, not just a single character.

All matching of the delimiter is done exactly, not using any regular expression patterns.

Return type: STRING

Examples:

SPLIT_PART('x,y,z',',',2) returns 'y'.

SPLIT_PART('one***two***three','***',2) returns 'two'.

SPLIT_PART('abc@@def@@ghi', '@@', 3) returns 'ghi'.

SPLIT_PART('abc@@def@@ghi', '@@', -3) returns 'abc'.

STRLEFT(STRING a, INT num_chars)

Purpose: Returns the leftmost characters of the string. Shorthand for a call to substr() with 2 arguments.

Return type: STRING

STRRIGHT(STRING a, INT num_chars)

Purpose: Returns the rightmost characters of the string. Shorthand for a call to substr() with 2 arguments.

Return type: STRING

SUBSTR(STRING a, INT start [, INT len]), SUBSTRING(STRING a, INT start [, INT len])

Purpose: Returns the portion of the string starting at a specified point, optionally with a specified maximum length. The characters in the string are indexed starting at 1.

Return type: STRING

TRANSLATE(STRING input, STRING from, STRING to)

Purpose: Returns the input string with each character in the from argument replaced with the corresponding character in the to argument. The characters are matched in the order they appear in from and to.

For example: translate ('hello world','world','earth') returns 'hetta earth'.

Return type: STRING

Usage notes:

If from contains more characters than to, the from characters that are beyond the length of to are removed in the result.

For example:

translate('abcdedg', 'bcd', '1') returns 'a1eg'.

translate('Unit Number#2', '# ', '_') returns 'UnitNumber_2'.

If from is NULL, the function returns NULL.

If to contains more characters than from, the extra characters in to are ignored.

If from contains duplicate characters, the duplicate character is replaced with the first matching character in to.

For example: translate ('hello','ll','67') returns 'he66o'.

TRIM(STRING a)

Purpose: Returns the input string with both leading and trailing spaces removed. The same as passing the string through both LTRIM() and RTRIM().

Usage notes: Often used during data cleansing operations during the ETL cycle, if input values might still have surrounding spaces. For a more general-purpose function that can remove other leading and trailing characters besides spaces, see BTRIM().

Return type: STRING

UPPER(STRING a), UCASE(STRING a)

Purpose: Returns the argument string converted to all-uppercase.

Return type: STRING

Usage notes:

In Impala 2.5 and higher, you can simplify queries that use many UPPER() and LOWER() calls to do case-insensitive comparisons, by using the ILIKE or IREGEXP operators instead.

Related Information

General information on Base64 encoding

GROUP_CONCAT function

Impala miscellaneous functions

Impala supports some utility functions that do not operate on a particular column or data type.

For example:

- CURRENT_DATABASE
- EFFECTIVE_USER
- GET_JSON_OBJECT
- LOGGED_IN_USER
- PID
- SLEEP
- USER
- UUID
- VERSION

CURREN_DATABASE()

Purpose: Returns the database that the session is currently using, either default if no database has been selected, or whatever database the session switched to through a USE statement or the `impalad` -d option.

Return type: STRING

EFFECTIVE_USER()

Purpose: Typically returns the same value as USER(), except if delegation is enabled, in which case it returns the ID of the delegated user.

Return type: STRING

GET_JSON_OBJECT(STRING json_str, STRING selector)

Purpose: Extracts JSON object from the *JSON_STR* based on the *SELECTOR* JSON path and returns the string of the extracted JSON object.

The function returns NULL if the input *JSON_STR* is invalid or if nothing is selected based on the *SELECTOR* JSON path.

The following characters are supported in the *SELECTOR* JSON path:

- $: Denotes the root object
- . : Denotes the child operator
- [] : Denotes the subscript operator for array
- * : Denotes the wildcard for [] or .

Return type: STRING

Examples:

```
---- QUERY
SELECT GET_JSON_OBJECT ('{"a":true, "b":false, "c":true}', '$.*')
;
---- RESULTS
[true,false,true]
```

```
---- QUERY
SELECT GET_JSON_OBJECT(t.json, '$.a.b.c') FROM (VALUES (
 ('{"a": {"b": {"c": 1}}}' AS json),
 ('{"a": {"b": {"c": 2}}}'),
 ('{"a": {"b": {"c": 3}}}')
)) t
---- RESULTS
'1'
'2'
```

```
'3'
```

```
---- QUERY
SELECT GET_JSON_OBJECT(t.json, '$.a'),
 GET_JSON_OBJECT(t.json, '$.b'),
 GET_JSON_OBJECT(t.json, '$.c')
FROM (VALUES (
 ('{"a":1, "b":2, "c":3}' AS json),
 ('{"b":2, "c":3}'),
 ('{"c":3}')
)) t
---- RESULTS
'1','2','3'
'NULL','2','3'
'NULL','NULL','3'
```

```
---- QUERY
SELECT GET_JSON_OBJECT(t.json, '$[1]'),
 GET_JSON_OBJECT(t.json, '$[*]')
FROM (VALUES (
 ('["a", "b", "c"]' AS json),
 ('["a", "b"]'),
 ('["a"]')
)) t
---- RESULTS
'b','["a","b","c"]'
'b','["a","b"]'
'NULL','a'
```

Added in: Impala 3.1

LOGGED_IN_USER()

Purpose: Typically returns the same value as USER(). If delegation is enabled, it returns the ID of the delegated user.

LOGGED_IN_USER() is an alias of EFFECTIVE_USER().

Return type: STRING

Added in: Impala 3.1

PID()

Purpose: Returns the process ID of the `impalad` daemon that the session is connected to. You can use it during low-level debugging, to issue Linux commands that trace, show the arguments, and so on the `impalad` process.

Return type: INT

SLEEP(INT ms)

Purpose: Pauses the query for a specified number of milliseconds. For slowing down queries with small result sets enough to monitor runtime execution, memory usage, or other factors that otherwise would be difficult to capture during the brief interval of query execution. When used in the SELECT list, it is called once for each row in the result set; adjust the number of milliseconds accordingly. For example, a query SELECT *, sleep(5) FROM table_with_1000_rows would take at least 5 seconds to complete (5 milliseconds * 1000 rows in result set). To avoid an excessive number of concurrent queries, use this function for troubleshooting on test and development systems, not for production queries.

Return type: N/A

USER()

Purpose: Returns the username of the Linux user who is connected to the `impalad` daemon. Typically called a single time, in a query without any FROM clause, to understand how authorization settings apply in a security context; once you know the logged-in username, you can check which groups that user belongs to, and from the list of groups you can check which roles are available to those groups through the authorization policy file.

Impala 2.0 and later, USER() returns the full Kerberos principal string, such as user@example.com, in a Kerberized environment.

When delegation is enabled, consider calling the effective_user() function instead.

Return type: STRING

UUID()

Purpose: Returns a universal unique identifier, a 128-bit value encoded as a string with groups of hexadecimal digits separated by dashes.

Each call to UUID() produces a new arbitrary value.

If you get a UUID for each row of a result set, you can use it as a unique identifier within a table, or even a unique ID across tables.

Return type: STRING

Usage notes:

Ascending numeric sequences of type BIGINT are often used as identifiers within a table, and as join keys across multiple tables. The uuid() value is a convenient alternative that does not require storing or querying the highest sequence number. For example, you can use it to quickly construct new unique identifiers during a data import job, or to combine data from different tables without the likelihood of ID collisions.

VERSION()

Purpose: Returns information such as the precise version number and build date for the impalad daemon that you are currently connected to. Typically used to confirm that you are connected to the expected level of Impala to use a particular feature, or to connect to several nodes and confirm they are all running the same level of `impalad`.

Return type: STRING (with one or more embedded newlines)

COORDINATOR()

Purpose: Returns the name of the host which is running the impalad daemon that is acting as the coordinator for the current query.

Return type: STRING

Added in: Impala 3.1

Impala aggregate functions

Aggregate functions are a special category with different rules. These functions calculate a return value across all the items in a result set, so they require a FROM clause in the query.

```
select count(product_id) from product_catalog;
select max(height), avg(height) from census_data where age > 20;
```

Aggregate functions also ignore NULL values rather than returning a NULL result. For example, if some rows have NULL for a particular column, those rows are ignored when computing the AVG() for that column. Likewise, specifying COUNT(*COL_NAME*) in a query counts only those rows where *COL_NAME* contains a non-NULL value.

APPX_MEDIAN function

An aggregate function that returns a value that is approximately the median (midpoint) of values in the set of input values.

Syntax:

```
APPX_MEDIAN([DISTINCT | ALL] EXPRESSION)
```

This function works with any input type, because the only requirement is that the type supports less-than and greater-than comparison operators.

Usage notes:

Because the return value represents the estimated midpoint, it might not reflect the precise midpoint value, especially if the cardinality of the input values is very high. If the cardinality is low (up to approximately 20,000), the result is more accurate because the sampling considers all or almost all of the different values.

Return type: Same as the input value, except for CHAR and VARCHAR arguments which produce a STRING result

The return value is always the same as one of the input values, not an “in-between” value produced by averaging.

Restrictions:

This function cannot be used in an analytic context. That is, the OVER() clause is not allowed at all with this function.

The APPX_MEDIAN function returns only the first 10 characters for string values (string, varchar, char). Additional characters are truncated.

Examples:

The following example uses a table of a million random floating-point numbers ranging up to approximately 50,000. The average is approximately 25,000. Because of the random distribution, we would expect the median to be close to this same number. Computing the precise median is a more intensive operation than computing the average, because it requires keeping track of every distinct value and how many times each occurs. The APPX_MEDIAN() function uses a sampling algorithm to return an approximate result, which in this case is close to the expected value. To make sure that the value is not substantially out of range due to a skewed distribution, subsequent queries confirm that there are approximately 500,000 values higher than the APPX_MEDIAN() value, and approximately 500,000 values lower than the APPX_MEDIAN() value.

```
[localhost:21000] > select min(x), max(x), avg(x) from million_numbers;
+-------------------+-------------------+-------------------+
| min(x)            | max(x)            | avg(x)            |
+-------------------+-------------------+-------------------+
| 4.725693727250069 | 49994.56852674231 | 24945.38563793553 |
+-------------------+-------------------+-------------------+
[localhost:21000] > select appx_median(x) from million_numbers;
+----------------+
| appx_median(x) |
+----------------+
| 24721.6        |
+----------------+
[localhost:21000] > select count(x) as higher from million_numbers where x >
 (select appx_median(x) from million_numbers);
+--------+
| higher |
+--------+
| 502013 |
+--------+
[localhost:21000] > select count(x) as lower from million_numbers where x <
 (select appx_median(x) from million_numbers);
+--------+
| lower  |
+--------+
```

```
| 497987 |
+--------+
```

The following example computes the approximate median using a subset of the values from the table, and then confirms that the result is a reasonable estimate for the midpoint.

```
[localhost:21000] > select appx_median(x) from million_numbers where x betwe
en 1000 and 5000;
+-------------------+
| appx_median(x)    |
+-------------------+
| 3013.107787358159 |
+-------------------+
[localhost:21000] > select count(x) as higher from million_numbers where x b
etween 1000 and 5000 and x > 3013.107787358159;
+--------+
| higher |
+--------+
| 37692  |
+--------+
[localhost:21000] > select count(x) as lower from million_numbers where x
 between 1000 and 5000 and x < 3013.107787358159;
+-------+
| lower |
+-------+
| 37089 |
+-------+
```

AVG function

An aggregate function that returns the average value from a set of numbers or TIMESTAMP values. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to AVG are NULL, AVG returns NULL.

Syntax:

```
AVG([DISTINCT | ALL] EXPRESSION) [OVER (ANALYTIC_CLAUSE)]
```

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Return type: DOUBLE for numeric values; TIMESTAMP for TIMESTAMP values

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
```

```
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
| EUROPE      | 19               |
| EUROPE      | 22               |
| EUROPE      | 23               |
| MIDDLE EAST | 4                |
| MIDDLE EAST | 10               |
| MIDDLE EAST | 11               |
| MIDDLE EAST | 13               |
| MIDDLE EAST | 20               |
+-------------+------------------+

select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
```

```
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Examples:

```
-- Average all the non-NULL values in a column.
insert overwrite avg_t values (2),(4),(6),(null),(null);
-- The average of the above values is 4: (2+4+6) / 3. The 2 NULL values are
 ignored.
select avg(x) from avg_t;
-- Average only certain values from the column.
select avg(x) from t1 where month = 'January' and year = '2013';
-- Apply a calculation to the value of the column before averaging.
select avg(x/3) from t1;
-- Apply a function to the value of the column before averaging.
-- Here we are substituting a value of 0 for all NULLs in the column,
-- so that those rows do factor into the return value.
select avg(isnull(x,0)) from t1;
-- Apply some number-returning function to a string column and average the
results.
-- If column s contains any NULLs, length(s) also returns NULL and those ro
ws are ignored.
select avg(length(s)) from t1;
-- Can also be used in combination with DISTINCT and/or GROUP BY.
-- Return more than one result.
select month, year, avg(page_visits) from web_stats group by month, year;
-- Filter the input to eliminate duplicates before performing the calcula
tion.
select avg(distinct x) from t1;
-- Filter the output after performing the calculation.
select avg(x) from t1 group by y having avg(x) between 1 and 20;
```

The following examples show how to use AVG() in an analytic context. They use a table containing integers from 1 to 10. Notice how the AVG() is reported for each input value, as opposed to the GROUP BY clause which condenses the result set.

```
select x, property, avg(x) over (partition by property) as avg from int_t wh
ere property in ('odd','even');
+----+----------+-----+
| x  | property | avg |
+----+----------+-----+
| 2  | even     | 6   |
| 4  | even     | 6   |
| 6  | even     | 6   |
| 8  | even     | 6   |
| 10 | even     | 6   |
| 1  | odd      | 5   |
| 3  | odd      | 5   |
| 5  | odd      | 5   |
| 7  | odd      | 5   |
| 9  | odd      | 5   |
+----+----------+-----+
```

Adding an ORDER BY clause lets you experiment with results that are cumulative or apply to a moving set of rows (the “window”). The following examples use AVG() in an analytic context (that is, with an OVER() clause) to produce a running average of all the even values, then a running average of all the odd values. The basic ORDER BY x clause implicitly activates a window clause of RANGE BETWEEN UNBOUNDED PRECEDING AND CURREN

T ROW, which is effectively the same as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, therefore all of these examples produce the same results:

```
select x, property,
  avg(x) over (partition by property order by x) as 'cumulative average'
  from int_t where property in ('odd','even');
+----+----------+--------------------+
| x  | property | cumulative average |
+----+----------+--------------------+
| 2  | even     | 2                  |
| 4  | even     | 3                  |
| 6  | even     | 4                  |
| 8  | even     | 5                  |
| 10 | even     | 6                  |
| 1  | odd      | 1                  |
| 3  | odd      | 2                  |
| 5  | odd      | 3                  |
| 7  | odd      | 4                  |
| 9  | odd      | 5                  |
+----+----------+--------------------+

select x, property,
  avg(x) over
  (
    partition by property
    order by x
    range between unbounded preceding and current row
  ) as 'cumulative average'
from int_t where property in ('odd','even');
+----+----------+--------------------+
| x  | property | cumulative average |
+----+----------+--------------------+
| 2  | even     | 2                  |
| 4  | even     | 3                  |
| 6  | even     | 4                  |
| 8  | even     | 5                  |
| 10 | even     | 6                  |
| 1  | odd      | 1                  |
| 3  | odd      | 2                  |
| 5  | odd      | 3                  |
| 7  | odd      | 4                  |
| 9  | odd      | 5                  |
+----+----------+--------------------+
select x, property,
  avg(x) over
  (
    partition by property
    order by x
    rows between unbounded preceding and current row
  ) as 'cumulative average'
  from int_t where property in ('odd','even');
+----+----------+--------------------+
| x  | property | cumulative average |
+----+----------+--------------------+
| 2  | even     | 2                  |
| 4  | even     | 3                  |
| 6  | even     | 4                  |
| 8  | even     | 5                  |
| 10 | even     | 6                  |
| 1  | odd      | 1                  |
| 3  | odd      | 2                  |
| 5  | odd      | 3                  |
| 7  | odd      | 4                  |
```

```
| 9  | odd      | 5                  |
+----+----------+--------------------+
```

The following examples show how to construct a moving window, with a running average taking into account 1 row before and 1 row after the current row, within the same partition (all the even values or all the odd values). Because of a restriction in the Impala RANGE syntax, this type of moving window is possible with the ROWS BETWEEN clause but not the RANGE BETWEEN clause:

```
select x, property,
  avg(x) over
  (
    partition by property
    order by x
    rows between 1 preceding and 1 following
  ) as 'moving average'
  from int_t where property in ('odd','even');
+----+----------+----------------+
| x  | property | moving average |
+----+----------+----------------+
| 2  | even     | 3              |
| 4  | even     | 4              |
| 6  | even     | 6              |
| 8  | even     | 8              |
| 10 | even     | 9              |
| 1  | odd      | 2              |
| 3  | odd      | 3              |
| 5  | odd      | 5              |
| 7  | odd      | 7              |
| 9  | odd      | 8              |
+----+----------+----------------+

-- Doesn't work because of syntax restriction on RANGE clause.
select x, property,
  avg(x) over
  (
    partition by property
    order by x
    range between 1 preceding and 1 following
  ) as 'moving average'
from int_t where property in ('odd','even');
ERROR: AnalysisException: RANGE is only supported with both the lower and
upper bounds UNBOUNDED or one UNBOUNDED and the other CURRENT ROW.
```

Restrictions:

Due to the way arithmetic on FLOAT and DOUBLE columns uses high-performance hardware instructions, and distributed queries can perform these operations in different order for each query, results can vary slightly for aggregate function calls such as SUM() and AVG() for FLOAT and DOUBLE columns, particularly on large data sets where millions or billions of values are summed or averaged. For perfect consistency and repeatability, use the DECIMAL data type for such operations instead of FLOAT or DOUBLE.

COUNT function

An aggregate function that returns the number of rows, or the number of non-NULL rows.

Syntax:

```
COUNT([DISTINCT | ALL] EXPRESSION) [OVER (ANALYTIC_CLAUSE)]
```

Depending on the argument, COUNT() considers rows that meet certain conditions:

- The notation COUNT(*) includes NULL values in the total.

- The notation COUNT(*COLUMN_NAME*) only considers rows where the column contains a non-NULL value.
- You can also combine COUNT with the DISTINCT operator to eliminate duplicates before counting, and to count the combinations of values across multiple columns.

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Return type: BIGINT

Usage notes:

If you frequently run aggregate functions such as MIN(), MAX(), and COUNT(DISTINCT) on partition key columns, consider enabling the OPTIMIZE_PARTITION_KEY_SCANS query option, which optimizes such queries. See the *OPTIMIZE_PARTITION_KEY_SCANS query option* topic for the kinds of queries that this option applies to, and slight differences in how partitions are evaluated when this query option is enabled.

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
```

```
| EUROPE      | 19                |
| EUROPE      | 22                |
| EUROPE      | 23                |
| MIDDLE EAST | 4                 |
| MIDDLE EAST | 10                |
| MIDDLE EAST | 11                |
| MIDDLE EAST | 13                |
| MIDDLE EAST | 20                |
+-------------+-------------------+

select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Examples:

```
-- How many rows total are in the table, regardless of NULL values?
select count(*) from t1;
-- How many rows are in the table with non-NULL values for a column?
select count(c1) from t1;
-- Count the rows that meet certain conditions.
-- Again, * includes NULLs, so COUNT(*) might be greater than COUNT(col).
select count(*) from t1 where x > 10;
select count(c1) from t1 where x > 10;
-- Can also be used in combination with DISTINCT and/or GROUP BY.
-- Combine COUNT and DISTINCT to find the number of unique values.
-- Must use column names rather than * with COUNT(DISTINCT ...) syntax.
-- Rows with NULL values are not counted.
select count(distinct c1) from t1;
-- Rows with a NULL value in _either_ column are not counted.
select count(distinct c1, c2) from t1;
-- Return more than one result.
select month, year, count(distinct visitor_id) from web_stats group by mon
th, year;
```

The following examples show how to use COUNT() in an analytic context. They use a table containing integers from 1 to 10. Notice how the COUNT() is reported for each input value, as opposed to the GROUP BY clause which condenses the result set.

```
select x, property, count(x) over (partition by property) as count from int_
t where property in ('odd','even');
+----+----------+-------+
| x  | property | count |
+----+----------+-------+
| 2  | even     | 5     |
| 4  | even     | 5     |
| 6  | even     | 5     |
| 8  | even     | 5     |
| 10 | even     | 5     |
| 1  | odd      | 5     |
| 3  | odd      | 5     |
| 5  | odd      | 5     |
| 7  | odd      | 5     |
| 9  | odd      | 5     |
+----+----------+-------+
```

Adding an ORDER BY clause lets you experiment with results that are cumulative or apply to a moving set of rows (the “window”). The following examples use COUNT() in an analytic context (that is, with an OVER() clause) to produce a running count of all the even values, then a running count of all the odd values. The basic ORDER BY x clause implicitly activates a window clause of RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, which is effectively the same as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, therefore all of these examples produce the same results:

```
select x, property,
  count(x) over (partition by property order by x) as 'cumulative count'
  from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative count |
+----+----------+------------------+
| 2  | even     | 1                |
| 4  | even     | 2                |
| 6  | even     | 3                |
| 8  | even     | 4                |
| 10 | even     | 5                |
| 1  | odd      | 1                |
| 3  | odd      | 2                |
| 5  | odd      | 3                |
| 7  | odd      | 4                |
| 9  | odd      | 5                |
+----+----------+------------------+

select x, property,
  count(x) over
  (
    partition by property
    order by x
    range between unbounded preceding and current row
  ) as 'cumulative total'
from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative count |
+----+----------+------------------+
| 2  | even     | 1                |
| 4  | even     | 2                |
| 6  | even     | 3                |
| 8  | even     | 4                |
| 10 | even     | 5                |
```

```
| 1  | odd      | 1                |
| 3  | odd      | 2                |
| 5  | odd      | 3                |
| 7  | odd      | 4                |
| 9  | odd      | 5                |
+----+----------+------------------+

select x, property,
  count(x) over
  (
    partition by property
    order by x
    rows between unbounded preceding and current row
  ) as 'cumulative total'
  from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative count |
+----+----------+------------------+
| 2  | even     | 1                |
| 4  | even     | 2                |
| 6  | even     | 3                |
| 8  | even     | 4                |
| 10 | even     | 5                |
| 1  | odd      | 1                |
| 3  | odd      | 2                |
| 5  | odd      | 3                |
| 7  | odd      | 4                |
| 9  | odd      | 5                |
+----+----------+------------------+
```

The following examples show how to construct a moving window, with a running count taking into account 1 row before and 1 row after the current row, within the same partition (all the even values or all the odd values). Therefore, the count is consistently 3 for rows in the middle of the window, and 2 for rows near the ends of the window, where there is no preceding or no following row in the partition. Because of a restriction in the Impala RANGE syntax, this type of moving window is possible with the ROWS BETWEEN clause but not the RANGE BETWEEN clause:

```
select x, property,
  count(x) over
  (
    partition by property
    order by x
    rows between 1 preceding and 1 following
  ) as 'moving total'
  from int_t where property in ('odd','even');
+----+----------+--------------+
| x  | property | moving total |
+----+----------+--------------+
| 2  | even     | 2            |
| 4  | even     | 3            |
| 6  | even     | 3            |
| 8  | even     | 3            |
| 10 | even     | 2            |
| 1  | odd      | 2            |
| 3  | odd      | 3            |
| 5  | odd      | 3            |
| 7  | odd      | 3            |
| 9  | odd      | 2            |
+----+----------+--------------+
-- Doesn't work because of syntax restriction on RANGE clause.
select x, property,
  count(x) over
  (
    partition by property
```

```
    order by x
    range between 1 preceding and 1 following
  ) as 'moving total'
from int_t where property in ('odd','even');
ERROR: AnalysisException: RANGE is only supported with both the lower and up
per bounds UNBOUNDED or one UNBOUNDED and the other CURRENT ROW.
```

GROUPING() and GROUPING_ID() functions

These aggregate functions are used in conjunction with the GROUP BY ROLLUP, CUBE, and GROUPING SETS clauses.

GROUPING()

GROUPING (column) function indicates whether the column in a GROUP BY list is aggregated or not. This function can be used only on a single column. Grouping returns 1 if the data is aggregated across the specified column or 0 for not aggregated in the result set. So these functions can be used to distinguish between a NULL value in the raw data vs the TOTAL across all values that are returned by ROLLUP, CUBE or GROUPING SETS for a specific column.

Example:

The following example shows GROUPING function applied to the columns "country" and "city".

```
Select country, city, sum(sales) as total_sales
      GROUPING(country) as GP_country,
      GROUPING(city) as GP_city
      from sales_history
      GROUP BY ROLLUP (country, city);
```

```
+--------+----------+-------------+-------------+------------+
| country| city     | total_sales | GP_country  | GP_city    |
+--------+----------+-------------+-------------+------------+
| US     | San Jose | 1000        | 0           | 0          |
| US     | Fremont  | 2000        | 0           | 0          |
| US     | NULL     | 3000        | 0           | 1          |
| Japan  | Hiroshima| 5000        | 0           | 0          |
| Japan  | Tokyo    | 3000        | 0           | 0          |
| Japan  | NULL     | 4000        | 0           | 0          |
| Japan  | NULL     | 12000       | 0           | 1          |
| NULL   | NULL     | 15000       | 1           | 1          |
+--------+----------+-------------+-------------+------------+
```

In the above example the value “NULL” in the third row for the column "city" represents the total sales for both cities (San jose and Fremont) in the US. This is the summary row added by the ROLLUP operation on the column "city". Since this is an aggregated value the result set for the GROUPING (city) is indicated by 1 in the GROUPING alias GP_city column for the corresponding row. However the NULL value in the 6th row for the column city shows the sales for an UNKNOWN city in Japan. SInce this is an actual NULL value in the raw data and does not indicate an aggregate the corresponding GROUPING alias GP_city column is indicated by 0.

GROUPING_ID()

GROUPING_ID function computes the level of grouping and returns an integer value which is unique for each combination of grouping values. The integer value can be interpreted as a bit vector where bits in the binary representation of the integer correspond to each GROUPING() value.

```
GROUPING_ID(col1, col2, col3) = GROUPING(col1) * pow(2,2) + GROUPING(col2) *
 pow(2,1) + GROUPING(col3) * pow(2,0)
```

This function can be used with multiple columns however they must match with the GROUP BY column list.

Return type:

GROUPING_ID function performs the binary to decimal conversion and returns a BIGINT.

Note: GROUPING_ID() with no arguments is equivalent to GROUPING_ID() with all of the grouping columns specified in the group by clause. This function can be used only in the SELECT list, HAVING, or ORDER BY clauses when GROUP BY is specified.

Example:

The following example shows GROUPING_ID applied to the column “country, city” and also matches the "column_expression" in the GROUP BY list.

```
Select country, city, sum(sales) as total_sales
      GROUPING_ID(country, city) as GPID,
     from sales_history
      GROUP BY ROLLUP (country, city);
```

In this example the GROUPING_ID(country, city) function is equivalent to GROUPING(country) * pow(2,1) + GROUPING(city). However, GROUPING_ID(country, city) returns an integer value whereas GROUPING(country) + GROUPING(city) returns a binary string.

```
      +--------+----------+-------------+----------------+
      | country| city     | total_sales | GPID (integer) |
      +--------+----------+-------------+----------------+
      | US     | San Jose | 1000        | 0              |
      | US     | Fremont  | 2000        | 0              |
      | US     | NULL     | 3000        | 1              |
      | Japan  | Hiroshima| 5000        | 0              |
      | Japan  | Tokyo    | 3000        | 0              |
      | Japan  | NULL     | 4000        | 0              |
      | Japan  | NULL     | 12000       | 1              |
      | NULL   | NULL     | 15000       | 3              |
      +--------+----------+-------------+----------------+
```

The following table shows the binary equivalent of the GROUPING_ID () function GPID calcualted in the above example.

```
    +-------------------------------------------+-------------------------
----------------------------------------------------------------------+
    | GROUPING_ID(country, city) GPID in integer | Binary equivalent of GRO
UPING_ID(country, city) = LPAD(BIN(GROUPING_ID(country, city)), 2, ‘0’)|
    +-------------------------------------------+-------------------------
----------------------------------------------------------------------+
    | 0                                          | 00
                                                                       |
    | 0                                          | 00
                                                                       |
    | 1                                          | 01
                                                                       |
    | 0                                          | 00
                                                                       |
    | 0                                          | 00
                                                                       |
    | 0                                          | 00
                                                                       |
    | 1                                          | 01
                                                                       |
```

```
    | 3                                          | 11
                                                                          |
    +-------------------------------------------+-------------------------
-----------------------------------------------------------------------+
```

GROUP_CONCAT function

An aggregate function that returns a single string representing the argument value concatenated together for each row of the result set. If the optional separator string is specified, the separator is added between each pair of concatenated values. The default separator is a comma followed by a space.

Syntax:

```
GROUP_CONCAT([ALL | DISTINCT] EXPRESSION [, SEPARATOR])
```

Usage notes: concat() and concat_ws() are appropriate for concatenating the values of multiple columns within the same row, while group_concat() joins together values from different rows.

By default, returns a single string covering the whole result set. To include other columns or values in the result set, or to produce multiple concatenated strings for subsets of rows, include a GROUP BY clause in the query.

Return type: STRING

This function cannot be used in an analytic context. That is, the OVER() clause is not allowed at all with this function.

Currently, Impala returns an error if the result value grows larger than 1 GiB.

Examples:

The following examples illustrate various aspects of the GROUP_CONCAT() function.

You can call the function directly on a STRING column. To use it with a numeric column, cast the value to STRING.

```
[localhost:21000] > create table t1 (x int, s string);
[localhost:21000] > insert into t1 values (1, "one"), (3, "three"), (2, "
two"), (1, "one");
[localhost:21000] > select group_concat(s) from t1;
+----------------------+
| group_concat(s)      |
+----------------------+
| one, three, two, one |
+----------------------+
[localhost:21000] > select group_concat(cast(x as string)) from t1;
+---------------------------------+
| group_concat(cast(x as string)) |
+---------------------------------+
| 1, 3, 2, 1                      |
+---------------------------------+
```

Specify the DISTINCT keyword to eliminate duplicate values from the concatenated result:

```
[localhost:21000] > select group_concat(distinct s) from t1;
+--------------------------+
| group_concat(distinct s) |
+--------------------------+
| three, two, one          |
+--------------------------+
```

The optional separator lets you format the result in flexible ways. The separator can be an arbitrary string expression, not just a single character.

```
[localhost:21000] > select group_concat(s,"|") from t1;
+----------------------+
| group_concat(s, '|') |
+----------------------+
| one|three|two|one    |
+----------------------+
[localhost:21000] > select group_concat(s,'---') from t1;
+-------------------------+
| group_concat(s, '---')  |
+-------------------------+
| one---three---two---one |
+-------------------------+
```

The default separator is a comma followed by a space. To get a comma-delimited result without extra spaces, specify a delimiter character that is only a comma.

```
[localhost:21000] > select group_concat(s,',') from t1;
+----------------------+
| group_concat(s, ',') |
+----------------------+
| one,three,two,one    |
+----------------------+
```

Including a GROUP BY clause lets you produce a different concatenated result for each group in the result set. In this example, the only X value that occurs more than once is 1, so that is the only row in the result set where GROUP_CONCAT() returns a delimited value. For groups containing a single value, GROUP_CONCAT() returns the original value of its STRING argument.

```
[localhost:21000] > select x, group_concat(s) from t1 group by x;
+---+-----------------+
| x | group_concat(s) |
+---+-----------------+
| 2 | two             |
| 3 | three           |
| 1 | one, one        |
+---+-----------------+
```

MAX function

An aggregate function that returns the maximum value from a set of numbers. Opposite of the MIN function. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to MAX are NULL, MAX returns NULL.

Syntax:

```
MAX([DISTINCT | ALL] EXPRESSION) [OVER (ANALYTIC_CLAUSE)]
```

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Restrictions: In Impala 2.0 and higher, this function can be used as an analytic function, but with restrictions on any window clause. For MAX() and MIN(), the window clause is only allowed if the start bound is UNBOUNDED PRECEDING.

Return type: Same as the input value, except for CHAR and VARCHAR arguments which produce a STRING result

Usage notes:

If you frequently run aggregate functions such as MIN(), MAX(), and COUNT(DISTINCT) on partition key columns, consider enabling the OPTIMIZE_PARTITION_KEY_SCANS query option, which optimizes such queries. See the *OPTIMIZE_PARTITION_KEY_SCANS query option* topic for the kinds of queries that this option applies to, and slight differences in how partitions are evaluated when this query option is enabled.

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
| EUROPE      | 19               |
| EUROPE      | 22               |
| EUROPE      | 23               |
| MIDDLE EAST | 4                |
| MIDDLE EAST | 10               |
| MIDDLE EAST | 11               |
| MIDDLE EAST | 13               |
| MIDDLE EAST | 20               |
+-------------+------------------+
```

```
select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Examples:

```
-- Find the largest value for this column in the table.
select max(c1) from t1;
-- Find the largest value for this column from a subset of the table.
select max(c1) from t1 where month = 'January' and year = '2013';
-- Find the largest value from a set of numeric function results.
select max(length(s)) from t1;
-- Can also be used in combination with DISTINCT and/or GROUP BY.
-- Return more than one result.
select month, year, max(purchase_price) from store_stats group by month,
year;
-- Filter the input to eliminate duplicates before performing the calculat
ion.
select max(distinct x) from t1;
```

The following examples show how to use MAX() in an analytic context. They use a table containing integers from 1 to 10. Notice how the MAX() is reported for each input value, as opposed to the GROUP BY clause which condenses the result set.

```
select x, property, max(x) over (partition by property) as max from int_t wh
ere property in ('odd','even');
+----+----------+-----+
| x  | property | max |
+----+----------+-----+
| 2  | even     | 10  |
| 4  | even     | 10  |
| 6  | even     | 10  |
| 8  | even     | 10  |
| 10 | even     | 10  |
```

```
| 1  | odd      | 9   |
| 3  | odd      | 9   |
| 5  | odd      | 9   |
| 7  | odd      | 9   |
| 9  | odd      | 9   |
+----+----------+-----+
```

Adding an ORDER BY clause lets you experiment with results that are cumulative or apply to a moving set of rows (the “window”). The following examples use MAX() in an analytic context (that is, with an OVER() clause) to display the smallest value of X encountered up to each row in the result set. The examples use two columns in the ORDER BY clause to produce a sequence of values that rises and falls, to illustrate how the MAX() result only increases or stays the same throughout each partition within the result set. The basic ORDER BY x clause implicitly activates a window clause of RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, which is effectively the same as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, therefore all of these examples produce the same results:

```
select x, property,
  max(x) over (order by property, x desc) as 'maximum to this point'
from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | maximum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 7                     |
| 3 | prime    | 7                     |
| 2 | prime    | 7                     |
| 9 | square   | 9                     |
| 4 | square   | 9                     |
| 1 | square   | 9                     |
+---+----------+-----------------------+
select x, property,
  max(x) over
  (
    order by property, x desc
    rows between unbounded preceding and current row
  ) as 'maximum to this point'
from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | maximum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 7                     |
| 3 | prime    | 7                     |
| 2 | prime    | 7                     |
| 9 | square   | 9                     |
| 4 | square   | 9                     |
| 1 | square   | 9                     |
+---+----------+-----------------------+
select x, property,
  max(x) over
  (
    order by property, x desc
    range between unbounded preceding and current row
  ) as 'maximum to this point'
from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | maximum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 7                     |
| 3 | prime    | 7                     |
| 2 | prime    | 7                     |
```

```
| 9 | square   | 9                    |
| 4 | square   | 9                    |
| 1 | square   | 9                    |
+---+----------+----------------------+
```

The following examples show how to construct a moving window, with a running maximum taking into account all rows before and 1 row after the current row. Because of a restriction in the Impala RANGE syntax, this type of moving window is possible with the ROWS BETWEEN clause but not the RANGE BETWEEN clause. Because of an extra Impala restriction on the MAX() and MIN() functions in an analytic context, the lower bound must be UNBO UNDED PRECEDING.

```
select x, property,
  max(x) over
  (
    order by property, x
    rows between unbounded preceding and 1 following
  ) as 'local maximum'
from int_t where property in ('prime','square');
+---+----------+---------------+
| x | property | local maximum |
+---+----------+---------------+
| 2 | prime    | 3             |
| 3 | prime    | 5             |
| 5 | prime    | 7             |
| 7 | prime    | 7             |
| 1 | square   | 7             |
| 4 | square   | 9             |
| 9 | square   | 9             |
+---+----------+---------------+
-- Doesn't work because of syntax restriction on RANGE clause.
select x, property,
  max(x) over
  (
    order by property, x
    range between unbounded preceding and 1 following
  ) as 'local maximum'
from int_t where property in ('prime','square');
ERROR: AnalysisException: RANGE is only supported with both the lower and u
pper bounds UNBOUNDED or one UNBOUNDED and the other CURRENT ROW.
```

MIN function

An aggregate function that returns the minimum value from a set of numbers. Opposite of the MAX function. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to MIN are NULL, MIN returns NULL.

Syntax:

```
MIN([DISTINCT | ALL] EXPRESSION) [OVER (ANALYTIC_CLAUSE)]
```

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Restrictions: In Impala 2.0 and higher, this function can be used as an analytic function, but with restrictions on any window clause. For MAX() and MIN(), the window clause is only allowed if the start bound is UNBOUNDED PR ECEDING.

Return type: Same as the input value, except for CHAR and VARCHAR arguments which produce a STRING result

Usage notes:

If you frequently run aggregate functions such as MIN(), MAX(), and COUNT(DISTINCT) on partition key columns, consider enabling the OPTIMIZE_PARTITION_KEY_SCANS query option, which optimizes such queries. See the

OPTIMIZE_PARTITION_KEY_SCANS query option topic for the kinds of queries that this option applies to, and slight differences in how partitions are evaluated when this query option is enabled.

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
| EUROPE      | 19               |
| EUROPE      | 22               |
| EUROPE      | 23               |
| MIDDLE EAST | 4                |
| MIDDLE EAST | 10               |
| MIDDLE EAST | 11               |
| MIDDLE EAST | 13               |
| MIDDLE EAST | 20               |
+-------------+------------------+

select
```

```
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Examples:

```
-- Find the smallest value for this column in the table.
select min(c1) from t1;
-- Find the smallest value for this column from a subset of the table.
select min(c1) from t1 where month = 'January' and year = '2013';
-- Find the smallest value from a set of numeric function results.
select min(length(s)) from t1;
-- Can also be used in combination with DISTINCT and/or GROUP BY.
-- Return more than one result.
select month, year, min(purchase_price) from store_stats group by month, y
ear;
-- Filter the input to eliminate duplicates before performing the calculati
on.
select min(distinct x) from t1;
```

The following examples show how to use MIN() in an analytic context. They use a table containing integers from 1 to 10. Notice how the MIN() is reported for each input value, as opposed to the GROUP BY clause which condenses the result set.

```
select x, property, min(x) over (partition by property) as min from int_t wh
ere property in ('odd','even');
+----+----------+-----+
| x  | property | min |
+----+----------+-----+
| 2  | even     | 2   |
| 4  | even     | 2   |
| 6  | even     | 2   |
| 8  | even     | 2   |
| 10 | even     | 2   |
| 1  | odd      | 1   |
| 3  | odd      | 1   |
```

```
| 5  | odd      | 1   |
| 7  | odd      | 1   |
| 9  | odd      | 1   |
+----+----------+-----+
```

Adding an ORDER BY clause lets you experiment with results that are cumulative or apply to a moving set of rows (the “window”). The following examples use MIN() in an analytic context (that is, with an OVER() clause) to display the smallest value of X encountered up to each row in the result set. The examples use two columns in the ORDER BY clause to produce a sequence of values that rises and falls, to illustrate how the MIN() result only decreases or stays the same throughout each partition within the result set. The basic ORDER BY x clause implicitly activates a window clause of RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, which is effectively the same as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, therefore all of these examples produce the same results:

```
select x, property, min(x) over (order by property, x desc) as 'minimum to
this point'
  from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | minimum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 5                     |
| 3 | prime    | 3                     |
| 2 | prime    | 2                     |
| 9 | square   | 2                     |
| 4 | square   | 2                     |
| 1 | square   | 1                     |
+---+----------+-----------------------+

select x, property,
  min(x) over
  (
    order by property, x desc
    range between unbounded preceding and current row
  ) as 'minimum to this point'
from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | minimum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 5                     |
| 3 | prime    | 3                     |
| 2 | prime    | 2                     |
| 9 | square   | 2                     |
| 4 | square   | 2                     |
| 1 | square   | 1                     |
+---+----------+-----------------------+
select x, property,
  min(x) over
  (
    order by property, x desc
    rows between unbounded preceding and current row
  ) as 'minimum to this point'
from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | minimum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 5                     |
| 3 | prime    | 3                     |
| 2 | prime    | 2                     |
| 9 | square   | 2                     |
```

```
| 4 | square   | 2                     |
| 1 | square   | 1                     |
+---+----------+-----------------------+
```

The following examples show how to construct a moving window, with a running minimum taking into account all rows before and 1 row after the current row. Because of a restriction in the Impala RANGE syntax, this type of moving window is possible with the ROWS BETWEEN clause but not the RANGE BETWEEN clause. Because of an extra Impala restriction on the MAX() and MIN() functions in an analytic context, the lower bound must be UNBO UNDED PRECEDING.

```
select x, property,
  min(x) over
  (
    order by property, x desc
    rows between unbounded preceding and 1 following
  ) as 'local minimum'
from int_t where property in ('prime','square');
+---+----------+---------------+
| x | property | local minimum |
+---+----------+---------------+
| 7 | prime    | 5             |
| 5 | prime    | 3             |
| 3 | prime    | 2             |
| 2 | prime    | 2             |
| 9 | square   | 2             |
| 4 | square   | 1             |
| 1 | square   | 1             |
+---+----------+---------------+
-- Doesn't work because of syntax restriction on RANGE clause.
select x, property,
  min(x) over
  (
    order by property, x desc
    range between unbounded preceding and 1 following
  ) as 'local minimum'
from int_t where property in ('prime','square');
ERROR: AnalysisException: RANGE is only supported with both the lower and u
pper bounds UNBOUNDED or one UNBOUNDED and the other CURRENT ROW.
```

NDV function

An aggregate function that returns an approximate value similar to the result of COUNT(DISTINCT *COL*), the “number of distinct values”. It is much faster than the combination of COUNT and DISTINCT, and uses a constant amount of memory and thus is less memory-intensive for columns with high cardinality.

Syntax:

```
NDV([DISTINCT | ALL] EXPRESSION [,scale])
```

Note: The optional argument scale must be an integer and can be in the range from 1 to 10 and maps to a precision used by the HyperLogLog (HLL) algorithm with the following mapping formula:

```
precision = scale + 8
```

Therefore a scale of 1 is mapped to a precision of 9 and a scale of 10 is mapped to a precision of 18.

Without the optional argument, the precision which determines the total number of different estimators in the HLL algorithm will be still 10.

A large precision value generally produces a better estimation with less error than a small precision value. This is due to the extra number of estimators involved. The expense is at the need of extra memory. For a given precision p, the amount of memory used by the HLL algorithm is in the order of 2^p bytes.

When provided a scale of 10 against a total of 22 distinct data sets loaded into external Impala tables, the error will be computed as abs(<true_unique_value> - <estimated_unique_value>) / <true_unique_value>

The scale of 10, mapped to the precision of 18, yielded the worst estimation error at 0.42% (for one set of 10 million integers), and average error no more than 0.17%. This was at the cost of 256Kb of memory for the internal data structure per evaluation of the HLL algorithm.

Usage notes:

This is the mechanism used internally by the COMPUTE STATS statement for computing the number of distinct values in a column.

Because this number is an estimate, it might not reflect the precise number of different values in the column, especially if the cardinality is very low or very high. If the estimated number is higher than the number of rows in the table, Impala adjusts the value internally during query planning.

Return type: DOUBLE in Impala 2.0 and higher; STRING in earlier releases

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
```

```
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
| EUROPE      | 19               |
| EUROPE      | 22               |
| EUROPE      | 23               |
| MIDDLE EAST | 4                |
| MIDDLE EAST | 10               |
| MIDDLE EAST | 11               |
| MIDDLE EAST | 13               |
| MIDDLE EAST | 20               |
+-------------+------------------+

select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Restrictions:

This function cannot be used in an analytic context. That is, the OVER() clause is not allowed at all with this function.

Examples:

The following example queries a billion-row table to illustrate the relative performance of COUNT(DISTINCT) and NDV(). It shows how COUNT(DISTINCT) gives a precise answer, but is inefficient for large-scale data where an approximate result is sufficient. The NDV() function gives an approximate result but is much faster.

```
select count(distinct col1) from sample_data;
+---------------------+
| count(distinct col1)|
+---------------------+
| 100000              |
```

```
+---------------------+
Fetched 1 row(s) in 20.13s

select cast(ndv(col1) as bigint) as col1 from sample_data;
+----------+
| col1     |
+----------+
| 139017   |
+----------+
Fetched 1 row(s) in 8.91s
```

The following example shows how you can code multiple NDV() calls in a single query, to easily learn which columns have substantially more or fewer distinct values. This technique is faster than running a sequence of queries with COUNT(DISTINCT) calls.

```
select cast(ndv(col1) as bigint) as col1, cast(ndv(col2) as bigint) as col2,
    cast(ndv(col3) as bigint) as col3, cast(ndv(col4) as bigint) as col4
  from sample_data;
+----------+-----------+------------+-----------+
| col1     | col2      | col3       | col4      |
+----------+-----------+------------+-----------+
| 139017   | 282       | 46         | 145636240 |
+----------+-----------+------------+-----------+
Fetched 1 row(s) in 34.97s

select count(distinct col1) from sample_data;
+---------------------+
| count(distinct col1)|
+---------------------+
| 100000              |
+---------------------+
Fetched 1 row(s) in 20.13s

select count(distinct col2) from sample_data;
+----------------------+
| count(distinct col2) |
+----------------------+
| 278                  |
+----------------------+
Fetched 1 row(s) in 20.09s
select count(distinct col3) from sample_data;
+-----------------------+
| count(distinct col3)  |
+-----------------------+
| 46                    |
+-----------------------+
Fetched 1 row(s) in 19.12s
select count(distinct col4) from sample_data;
+----------------------+
| count(distinct col4) |
+----------------------+
| 147135880            |
+----------------------+
Fetched 1 row(s) in 266.95s
```

STDDEV, STDDEV_SAMP, STDDEV_POP functions

An aggregate function that returns the standard deviation of a set of numbers.

Syntax:

```
{ STDDEV | STDDEV_SAMP | STDDEV_POP } ([DISTINCT | ALL] EXPRESSION)
```

This function works with any numeric data type.

Return type: DOUBLE in Impala 2.0 and higher; STRING in earlier releases

This function is typically used in mathematical formulas related to probability distributions.

The STDDEV_POP() and STDDEV_SAMP() functions compute the population standard deviation and sample standard deviation, respectively, of the input values. (STDDEV() is an alias for STDDEV_SAMP().) Both functions evaluate all input rows matched by the query. The difference is that STDDEV_SAMP() is scaled by 1/(N-1) while STDDEV_POP() is scaled by 1/N.

If no input rows match the query, the result of any of these functions is NULL. If a single input row matches the query, the result of any of these functions is "0.0".

Examples:

This example demonstrates how STDDEV() and STDDEV_SAMP() return the same result, while STDDEV_POP() uses a slightly different calculation to reflect that the input data is considered part of a larger “population”.

```
[localhost:21000] > select stddev(score) from test_scores;
+---------------+
| stddev(score) |
+---------------+
| 28.5          |
+---------------+
[localhost:21000] > select stddev_samp(score) from test_scores;
+--------------------+
| stddev_samp(score) |
+--------------------+
| 28.5               |
+--------------------+
[localhost:21000] > select stddev_pop(score) from test_scores;
+-------------------+
| stddev_pop(score) |
+-------------------+
| 28.4858           |
+-------------------+
```

This example demonstrates that, because the return value of these aggregate functions is a STRING, you must currently convert the result with CAST.

```
[localhost:21000] > create table score_stats as select cast(stddev(score) as
 decimal(7,4)) `standard_deviation`, cast(variance(score) as decimal(7,4)) `
variance` from test_scores;
+-------------------+
| summary           |
+-------------------+
| Inserted 1 row(s) |
+-------------------+
[localhost:21000] > desc score_stats;
+--------------------+--------------+---------+
| name               | type         | comment |
+--------------------+--------------+---------+
| standard_deviation | decimal(7,4) |         |
| variance           | decimal(7,4) |         |
+--------------------+--------------+---------+
```

Restrictions:

This function cannot be used in an analytic context. That is, the OVER() clause is not allowed at all with this function.

Related information:

The STDDEV(), STDDEV_POP(), and STDDEV_SAMP() functions compute the standard deviation (square root of the variance) based on the results of VARIANCE(), VARIANCE_POP(), and VARIANCE_SAMP() respectively. See VARIANCE, VARIANCE_SAMP, VARIANCE_POP, VAR_SAMP, VAR_POP functions on page 419 for details about the variance property.

Related Information

Standard deviation

SUM function

An aggregate function that returns the sum of a set of numbers. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to MIN are NULL, SUM returns NULL.

Syntax:

```
SUM([DISTINCT | ALL] EXPRESSION) [OVER (ANALYTIC_CLAUSE)]
```

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Return type: BIGINT for integer arguments, DOUBLE for floating-point arguments

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
```

```
| AMERICA     | 24                |
| ASIA        | 8                 |
| ASIA        | 9                 |
| ASIA        | 12                |
| ASIA        | 18                |
| ASIA        | 21                |
| EUROPE      | 6                 |
| EUROPE      | 7                 |
| EUROPE      | 19                |
| EUROPE      | 22                |
| EUROPE      | 23                |
| MIDDLE EAST | 4                 |
| MIDDLE EAST | 10                |
| MIDDLE EAST | 11                |
| MIDDLE EAST | 13                |
| MIDDLE EAST | 20                |
+-------------+-------------------+

select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Examples:

The following example shows how to use SUM() to compute the total for all the values in the table, a subset of values, or the sum for each combination of values in the GROUP BY clause:

```
-- Total all the values for this column in the table.
select sum(c1) from t1;
-- Find the total for this column from a subset of the table.
select sum(c1) from t1 where month = 'January' and year = '2013';
-- Find the total from a set of numeric function results.
select sum(length(s)) from t1;
-- Often used with functions that return predefined values to compute a s
core.
```

```
select sum(case when grade = 'A' then 1.0 when grade = 'B' then 0.75 else 0)
 as class_honors from test_scores;
-- Can also be used in combination with DISTINCT and/or GROUP BY.
-- Return more than one result.
select month, year, sum(purchase_price) from store_stats group by month,
year;
-- Filter the input to eliminate duplicates before performing the calculat
ion.
select sum(distinct x) from t1;
```

The following examples show how to use SUM() in an analytic context. They use a table containing integers from 1 to 10. Notice how the SUM() is reported for each input value, as opposed to the GROUP BY clause which condenses the result set.

```
select x, property, sum(x) over (partition by property) as sum from int_t w
here property in ('odd','even');
+----+----------+-----+
| x  | property | sum |
+----+----------+-----+
| 2  | even     | 30  |
| 4  | even     | 30  |
| 6  | even     | 30  |
| 8  | even     | 30  |
| 10 | even     | 30  |
| 1  | odd      | 25  |
| 3  | odd      | 25  |
| 5  | odd      | 25  |
| 7  | odd      | 25  |
| 9  | odd      | 25  |
+----+----------+-----+
```

Adding an ORDER BY clause lets you experiment with results that are cumulative or apply to a moving set of rows (the “window”). The following examples use SUM() in an analytic context (that is, with an OVER() clause) to produce a running total of all the even values, then a running total of all the odd values. The basic ORDER BY x clause implicitly activates a window clause of RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, which is effectively the same as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, therefore all of these examples produce the same results:

```
select x, property,
  sum(x) over (partition by property order by x) as 'cumulative total'
  from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative total |
+----+----------+------------------+
| 2  | even     | 2                |
| 4  | even     | 6                |
| 6  | even     | 12               |
| 8  | even     | 20               |
| 10 | even     | 30               |
| 1  | odd      | 1                |
| 3  | odd      | 4                |
| 5  | odd      | 9                |
| 7  | odd      | 16               |
| 9  | odd      | 25               |
+----+----------+------------------+

select x, property,
  sum(x) over
  (
    partition by property
    order by x
    range between unbounded preceding and current row
```

```
  ) as 'cumulative total'
from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative total |
+----+----------+------------------+
| 2  | even     | 2                |
| 4  | even     | 6                |
| 6  | even     | 12               |
| 8  | even     | 20               |
| 10 | even     | 30               |
| 1  | odd      | 1                |
| 3  | odd      | 4                |
| 5  | odd      | 9                |
| 7  | odd      | 16               |
| 9  | odd      | 25               |
+----+----------+------------------+

select x, property,
  sum(x) over
  (
    partition by property
    order by x
    rows between unbounded preceding and current row
  ) as 'cumulative total'
  from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative total |
+----+----------+------------------+
| 2  | even     | 2                |
| 4  | even     | 6                |
| 6  | even     | 12               |
| 8  | even     | 20               |
| 10 | even     | 30               |
| 1  | odd      | 1                |
| 3  | odd      | 4                |
| 5  | odd      | 9                |
| 7  | odd      | 16               |
| 9  | odd      | 25               |
+----+----------+------------------+
```

Changing the direction of the ORDER BY clause causes the intermediate results of the cumulative total to be calculated in a different order:

```
select sum(x) over (partition by property order by x desc) as 'cumulative
 total'
  from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative total |
+----+----------+------------------+
| 10 | even     | 10               |
| 8  | even     | 18               |
| 6  | even     | 24               |
| 4  | even     | 28               |
| 2  | even     | 30               |
| 9  | odd      | 9                |
| 7  | odd      | 16               |
| 5  | odd      | 21               |
| 3  | odd      | 24               |
| 1  | odd      | 25               |
+----+----------+------------------+
```

The following examples show how to construct a moving window, with a running total taking into account 1 row before and 1 row after the current row, within the same partition (all the even values or all the odd values). Because

of a restriction in the Impala RANGE syntax, this type of moving window is possible with the ROWS BETWEEN clause but not the RANGE BETWEEN clause:

```
select x, property,
  sum(x) over
  (
    partition by property
    order by x
    rows between 1 preceding and 1 following
  ) as 'moving total'
  from int_t where property in ('odd','even');
+----+----------+--------------+
| x  | property | moving total |
+----+----------+--------------+
| 2  | even     | 6            |
| 4  | even     | 12           |
| 6  | even     | 18           |
| 8  | even     | 24           |
| 10 | even     | 18           |
| 1  | odd      | 4            |
| 3  | odd      | 9            |
| 5  | odd      | 15           |
| 7  | odd      | 21           |
| 9  | odd      | 16           |
+----+----------+--------------+
-- Doesn't work because of syntax restriction on RANGE clause.
select x, property,
  sum(x) over
  (
    partition by property
    order by x
    range between 1 preceding and 1 following
  ) as 'moving total'
from int_t where property in ('odd','even');
ERROR: AnalysisException: RANGE is only supported with both the lower and up
per bounds UNBOUNDED or one UNBOUNDED and the other CURRENT ROW.
```

Restrictions:

Due to the way arithmetic on FLOAT and DOUBLE columns uses high-performance hardware instructions, and distributed queries can perform these operations in different order for each query, results can vary slightly for aggregate function calls such as SUM() and AVG() for FLOAT and DOUBLE columns, particularly on large data sets where millions or billions of values are summed or averaged. For perfect consistency and repeatability, use the DECIMAL data type for such operations instead of FLOAT or DOUBLE.

VARIANCE, VARIANCE_SAMP, VARIANCE_POP, VAR_SAMP, VAR_POP functions

An aggregate function that returns the variance of a set of numbers. This is a mathematical property that signifies how far the values spread apart from the mean. The return value can be zero (if the input is a single value, or a set of identical values), or a positive number otherwise.

Syntax:

```
{ VARIANCE | VAR[IANCE]_SAMP | VAR[IANCE]_POP } ([DISTINCT | ALL
] EXPRESSION)
```

This function works with any numeric data type.

Return type: DOUBLE in Impala 2.0 and higher; STRING in earlier releases

This function is typically used in mathematical formulas related to probability distributions.

The VARIANCE_SAMP() and VARIANCE_POP() functions compute the sample variance and population variance, respectively, of the input values. (VARIANCE() is an alias for VARIANCE_SAMP().) Both functions evaluate all input rows matched by the query. The difference is that STDDEV_SAMP() is scaled by 1/(N-1) while STDDEV_POP() is scaled by 1/N.

The functions VAR_SAMP() and VAR_POP() are the same as VARIANCE_SAMP() and VARIANCE_POP(), respectively. These aliases are available in Impala 2.0 and later.

If no input rows match the query, the result of any of these functions is NULL. If a single input row matches the query, the result of any of these functions is "0.0".

Examples:

This example demonstrates how VARIANCE() and VARIANCE_SAMP() return the same result, while VARIANCE_POP() uses a slightly different calculation to reflect that the input data is considered part of a larger “population”.

```
[localhost:21000] > select variance(score) from test_scores;
+-----------------+
| variance(score) |
+-----------------+
| 812.25          |
+-----------------+
[localhost:21000] > select variance_samp(score) from test_scores;
+----------------------+
| variance_samp(score) |
+----------------------+
| 812.25               |
+----------------------+
[localhost:21000] > select variance_pop(score) from test_scores;
+---------------------+
| variance_pop(score) |
+---------------------+
| 811.438             |
+---------------------+
```

This example demonstrates that, because the return value of these aggregate functions is a STRING, you convert the result with CAST if you need to do further calculations as a numeric value.

```
[localhost:21000] > create table score_stats as select cast(stddev(score) as
 decimal(7,4)) `standard_deviation`, cast(variance(score) as decimal(7,4)) `
variance` from test_scores;
+-------------------+
| summary           |
+-------------------+
| Inserted 1 row(s) |
+-------------------+
[localhost:21000] > desc score_stats;
+--------------------+--------------+---------+
| name               | type         | comment |
+--------------------+--------------+---------+
| standard_deviation | decimal(7,4) |         |
| variance           | decimal(7,4) |         |
+--------------------+--------------+---------+
```

Restrictions:

This function cannot be used in an analytic context. That is, the OVER() clause is not allowed at all with this function.

Related information:

The STDDEV(), STDDEV_POP(), and STDDEV_SAMP() functions compute the standard deviation (square root of the variance) based on the results of VARIANCE(), VARIANCE_POP(), and VARIANCE_SAMP() respectively.

See STDDEV, STDDEV_SAMP, STDDEV_POP functions on page 413 for details about the standard deviation property.

Related Information

Mathematical variance

Impala analytic functions

Analytic functions (also known as window functions) are a special category of built-in functions. Like aggregate functions, they examine the contents of multiple input rows to compute each output value. However, rather than being limited to one result value per GROUP BY group, they operate on *windows* where the input rows are ordered and grouped using flexible conditions expressed through an OVER() clause.

Some functions, such as LAG() and RANK(), can only be used in this analytic context. Some aggregate functions do double duty: when you call the aggregation functions such as MAX(), SUM(), AVG(), and so on with an OVER() clause, they produce an output value for each row, based on computations across other rows in the window.

Although analytic functions often compute the same value you would see from an aggregate function in a GROUP BY query, the analytic functions produce a value for each row in the result set rather than a single value for each group. This flexibility lets you include additional columns in the SELECT list, offering more opportunities for organizing and filtering the result set.

Analytic function calls are only allowed in the SELECT list and in the outermost ORDER BY clause of the query. During query processing, analytic functions are evaluated after other query stages such as joins, WHERE, and GROUP BY,

The rows that are part of each partition are analyzed by computations across an ordered or unordered set of rows. For example, COUNT() and SUM() might be applied to all the rows in the partition, in which case the order of analysis does not matter. The ORDER BY clause might be used inside the OVER() clause to defines the ordering that applies to functions such as LAG() and FIRST_VALUE().

Analytic functions are frequently used in fields such as finance and science to provide trend, outlier, and bucketed analysis for large data sets. You might also see the term “window functions” in database literature, referring to the sequence of rows (the “window”) that the function call applies to, particularly when the OVER clause includes a ROWS or RANGE keyword.

Related Information

OVER
WINDOW
AVG
COUNT
CUME_DIST
DENSE_RANK
FIRST_VALUE
LAG
LAST_VALUE
LEAD
MAX
MIN
NTILE
PERCENT_RANK
RANK
ROW_NUMBER
SUM

OVER

The OVER clause is required for calls to pure analytic functions such as LEAD(), RANK(), and FIRST_VALUE(). When you include an OVER clause with calls to aggregate functions such as MAX(), COUNT(), or SUM(), they operate as analytic functions.

Syntax:

```
function(ARGS) OVER([PARTITION_BY_CLAUSE] [ORDER_BY_CLAUSE [WINDOW_CLAUSE]])
partition_by_clause ::= PARTITION BY EXPR [, EXPR ...]
order_by_clause ::= ORDER BY EXPR  [ASC | DESC] [NULLS FIRST | NULLS LAST]
              [, EXPR [ASC | DESC] [NULLS FIRST | NULLS LAST] ...]
window_clause: See Window clause
```

PARTITION BY clause:

The PARTITION BY clause acts much like the GROUP BY clause in the outermost block of a query. It divides the rows into groups containing identical values in one or more columns. These logical groups are known as *partitions*. Throughout the discussion of analytic functions, “partitions” refers to the groups produced by the PARTITION BY clause, not to partitioned tables. However, note the following limitation that applies specifically to analytic function calls involving partitioned tables.

In queries involving both analytic functions and partitioned tables, partition pruning only occurs for columns named in the PARTITION BY clause of the analytic function call. For example, if an analytic function query has a clause such as WHERE year=2016, the way to make the query prune all other YEAR partitions is to include PART ITION BY year in the analytic function call; for example, OVER (PARTITION BY year,*OTHER_COLUMNS OTHER_ANALYTIC_CLAUSES*).

The sequence of results from an analytic function “resets” for each new partition in the result set. That is, the set of preceding or following rows considered by the analytic function always come from a single partition. Any MAX(), SUM(), ROW_NUMBER(), and so on apply to each partition independently. Omit the PARTITION BY clause to apply the analytic operation to all the rows in the table.

ORDER BY clause:

The ORDER BY clause works much like the ORDER BY clause in the outermost block of a query. It defines the order in which rows are evaluated for the entire input set, or for each group produced by a PARTITION BY clause. You can order by one or multiple expressions, and for each expression optionally choose ascending or descending order and whether nulls come first or last in the sort order. Because this ORDER BY clause only defines the order in which rows are evaluated, if you want the results to be output in a specific order, also include an ORDER BY clause in the outer block of the query.

When the ORDER BY clause is omitted, the analytic function applies to all items in the group produced by the PART ITION BY clause. When the ORDER BY clause is included, the analysis can apply to all or a subset of the items in the group, depending on the optional window clause.

The order in which the rows are analyzed is only defined for those columns specified in ORDER BY clauses.

One difference between the analytic and outer uses of the ORDER BY clause: inside the OVER clause, ORDER BY 1 or other integer value is interpreted as a constant sort value (effectively a no-op) rather than referring to column 1.

Window clause:

The window clause is only allowed in combination with an ORDER BY clause. If the ORDER BY clause is specified but the window clause is not, the default window is RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

HBase considerations:

Because HBase tables are optimized for single-row lookups rather than full scans, analytic functions using the OVER () clause are not recommended for HBase tables. Although such queries work, their performance is lower than on comparable tables using HDFS data files.

Parquet considerations:

Analytic functions are very efficient for Parquet tables. The data that is examined during evaluation of the OVER() clause comes from a specified set of columns, and the values for each column are arranged sequentially within each data file.

Text table considerations:

Analytic functions are convenient to use with text tables for exploratory business intelligence. When the volume of data is substantial, prefer to use Parquet tables for performance-critical analytic queries.

Examples:

The following example shows how to synthesize a numeric sequence corresponding to all the rows in a table. The new table has the same columns as the old one, plus an additional column ID containing the integers 1, 2, 3, and so on, corresponding to the order of a TIMESTAMP column in the original table.

```
CREATE TABLE events_with_id AS
  SELECT
    row_number() OVER (ORDER BY date_and_time) AS id,
    c1, c2, c3, c4
  FROM events;
```

The following example shows how to determine the number of rows containing each value for a column. Unlike a corresponding GROUP BY query, this one can analyze a single column and still return all values (not just the distinct ones) from the other columns.

```
SELECT x, y, z,
  count() OVER (PARTITION BY x) AS how_many_x
FROM t1;
```

Restrictions:

You cannot directly combine the DISTINCT operator with analytic function calls. You can put the analytic function call in a WITH clause or an inline view, and apply the DISTINCT operator to its result set.

```
WITH t1 AS (SELECT x, sum(x) OVER (PARTITION BY x) AS total FROM t1)
  SELECT DISTINCT x, total FROM t1;
```

WINDOW

Certain analytic functions accept an optional *window clause*, which makes the function analyze only certain rows “around” the current row rather than all rows in the partition. For example, you can get a moving average by specifying some number of preceding and following rows, or a running count or running total by specifying all rows up to the current position. This clause can result in different analytic results for rows within the same partition.

The window clause is supported with the AVG(), COUNT(), FIRST_VALUE(), LAST_VALUE(), and SUM() functions. For MAX() and MIN(), the window clause only allowed if the start bound is UNBOUNDED PRECED ING

Syntax:

```
ROWS BETWEEN [ { M | UNBOUNDED } PRECEDING | CURRENT ROW] [ AND [CURRENT ROW
 | { UNBOUNDED | N } FOLLOWING] ]
RANGE BETWEEN [ {M | UNBOUNDED } PRECEDING | CURRENT ROW] [ AND [CURRENT ROW
 | { UNBOUNDED | N } FOLLOWING] ]
```

ROWS BETWEEN defines the size of the window in terms of the indexes of the rows in the result set. The size of the window is predictable based on the clauses the position within the result set.

RANGE BETWEEN does not currently support numeric arguments to define a variable-size sliding window.

Currently, Impala supports only some combinations of arguments to the RANGE clause:

- RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW (the default when ORDER BY is specified and the window clause is omitted)
- RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
- RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

When RANGE is used, CURRENT ROW includes not just the current row but all rows that are tied with the current row based on the ORDER BY expressions.

Examples:

The following examples show financial data for a fictional stock symbol JDR. The closing price moves up and down each day.

```
create table stock_ticker (stock_symbol string, closing_price decimal(8,2),
closing_date timestamp);
...load some data...
select * from stock_ticker order by stock_symbol, closing_date
+--------------+---------------+---------------------+
| stock_symbol | closing_price | closing_date        |
+--------------+---------------+---------------------+
| JDR          | 12.86         | 2014-10-02 00:00:00 |
| JDR          | 12.89         | 2014-10-03 00:00:00 |
| JDR          | 12.94         | 2014-10-04 00:00:00 |
| JDR          | 12.55         | 2014-10-05 00:00:00 |
| JDR          | 14.03         | 2014-10-06 00:00:00 |
| JDR          | 14.75         | 2014-10-07 00:00:00 |
| JDR          | 13.98         | 2014-10-08 00:00:00 |
+--------------+---------------+---------------------+
```

The queries use analytic functions with window clauses to compute moving averages of the closing price. For example, ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING produces an average of the value from a 3-day span, producing a different value for each row. The first row, which has no preceding row, only gets averaged with the row following it. If the table contained more than one stock symbol, the PARTITION BY clause would limit the window for the moving average to only consider the prices for a single stock.

```
select stock_symbol, closing_date, closing_price,
  avg(closing_price) over (partition by stock_symbol order by closing_date
    rows between 1 preceding and 1 following) as moving_average
  from stock_ticker;
+--------------+---------------------+---------------+----------------+
| stock_symbol | closing_date        | closing_price | moving_average |
+--------------+---------------------+---------------+----------------+
| JDR          | 2014-10-02 00:00:00 | 12.86         | 12.87          |
| JDR          | 2014-10-03 00:00:00 | 12.89         | 12.89          |
| JDR          | 2014-10-04 00:00:00 | 12.94         | 12.79          |
| JDR          | 2014-10-05 00:00:00 | 12.55         | 13.17          |
| JDR          | 2014-10-06 00:00:00 | 14.03         | 13.77          |
| JDR          | 2014-10-07 00:00:00 | 14.75         | 14.25          |
| JDR          | 2014-10-08 00:00:00 | 13.98         | 14.36          |
+--------------+---------------------+---------------+----------------+
```

The clause ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW produces a cumulative moving average, from the earliest data up to the value for each day.

```
select stock_symbol, closing_date, closing_price,
  avg(closing_price) over (partition by stock_symbol order by closing_date
    rows between unbounded preceding and current row) as moving_average
  from stock_ticker;
+--------------+---------------------+---------------+----------------+
| stock_symbol | closing_date        | closing_price | moving_average |
+--------------+---------------------+---------------+----------------+
| JDR          | 2014-10-02 00:00:00 | 12.86         | 12.86          |
```

```
| JDR           | 2014-10-03 00:00:00 | 12.89          | 12.87          |
| JDR           | 2014-10-04 00:00:00 | 12.94          | 12.89          |
| JDR           | 2014-10-05 00:00:00 | 12.55          | 12.81          |
| JDR           | 2014-10-06 00:00:00 | 14.03          | 13.05          |
| JDR           | 2014-10-07 00:00:00 | 14.75          | 13.33          |
| JDR           | 2014-10-08 00:00:00 | 13.98          | 13.42          |
+--------------+---------------------+---------------+----------------+
```

AVG

You can include an OVER clause with a call to this function to use it as an analytic function.

Related Information

AVG function

COUNT

You can include an OVER clause with a call to this function to use it as an analytic function.

Related Information

COUNT function

CUME_DIST

Returns the cumulative distribution of a value. The value for each row in the result set is greater than 0 and less than or equal to 1.

Syntax:

```
CUME_DIST (EXPR)
  OVER ([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

Within each partition of the result set, the CUME_DIST() value represents an ascending sequence that ends at 1. Each value represents the proportion of rows in the partition whose values are less than or equal to the value in the current row.

If the sequence of input values contains ties, the CUME_DIST() results are identical for the tied values.

Impala only supports the CUME_DIST() function in an analytic context, not as a regular aggregate function.

Examples:

This example uses a table with 9 rows. The CUME_DIST() function evaluates the entire table because there is no PARTITION BY clause, with the rows ordered by the weight of the animal. the sequence of values shows that 1/9 of the values are less than or equal to the lightest animal (mouse), 2/9 of the values are less than or equal to the second-lightest animal, and so on up to the heaviest animal (elephant), where 9/9 of the rows are less than or equal to its weight.

```
create table animals (name string, kind string, kilos decimal(9,3));
insert into animals values
  ('Elephant', 'Mammal', 4000), ('Giraffe', 'Mammal', 1200), ('Mouse', 'Mamm
al', 0.020),
  ('Condor', 'Bird', 15), ('Horse', 'Mammal', 500), ('Owl', 'Bird', 2.5),
  ('Ostrich', 'Bird', 145), ('Polar bear', 'Mammal', 700), ('Housecat', '
Mammal', 5);

select name, cume_dist() over (order by kilos) from animals;
+------------+-----------------------+
| name       | cume_dist() OVER(...) |
+------------+-----------------------+
| Elephant   | 1                     |
```

```
| Giraffe    | 0.8888888888888888    |
| Polar bear | 0.7777777777777778    |
| Horse      | 0.6666666666666666    |
| Ostrich    | 0.5555555555555556    |
| Condor     | 0.4444444444444444    |
| Housecat   | 0.3333333333333333    |
| Owl        | 0.2222222222222222    |
| Mouse      | 0.1111111111111111    |
+------------+-----------------------+
```

Using a PARTITION BY clause produces a separate sequence for each partition group, in this case one for mammals and one for birds. Because there are 3 birds and 6 mammals, the sequence illustrates how 1/3 of the “Bird” rows have a kilos value that is less than or equal to the lightest bird, 1/6 of the “Mammal” rows have a kilos value that is less than or equal to the lightest mammal, and so on until both the heaviest bird and heaviest mammal have a CUME_DIS T() value of 1.

```
select name, kind, cume_dist() over (partition by kind order by kilos) from
animals
+------------+--------+-----------------------+
| name       | kind   | cume_dist() OVER(...) |
+------------+--------+-----------------------+
| Ostrich    | Bird   | 1                     |
| Condor     | Bird   | 0.6666666666666666    |
| Owl        | Bird   | 0.3333333333333333    |
| Elephant   | Mammal | 1                     |
| Giraffe    | Mammal | 0.8333333333333334    |
| Polar bear | Mammal | 0.6666666666666666    |
| Horse      | Mammal | 0.5                   |
| Housecat   | Mammal | 0.3333333333333333    |
| Mouse      | Mammal | 0.1666666666666667    |
+------------+--------+-----------------------+
```

We can reverse the ordering within each partition group by using an ORDER BY ... DESC clause within the OVER() clause. Now the lightest (smallest value of kilos) animal of each kind has a CUME_DIST() value of 1.

```
select name, kind, cume_dist() over (partition by kind order by kilos desc)
from animals
+------------+--------+-----------------------+
| name       | kind   | cume_dist() OVER(...) |
+------------+--------+-----------------------+
| Owl        | Bird   | 1                     |
| Condor     | Bird   | 0.6666666666666666    |
| Ostrich    | Bird   | 0.3333333333333333    |
| Mouse      | Mammal | 1                     |
| Housecat   | Mammal | 0.8333333333333334    |
| Horse      | Mammal | 0.6666666666666666    |
| Polar bear | Mammal | 0.5                   |
| Giraffe    | Mammal | 0.3333333333333333    |
| Elephant   | Mammal | 0.1666666666666667    |
+------------+--------+-----------------------+
```

The following example manufactures some rows with identical values in the kilos column, to demonstrate how the results look in case of tie values. For simplicity, it only shows the CUME_DIST() sequence for the “Bird” rows. Now with 3 rows all with a value of 15, all of those rows have the same CUME_DIST() value. 4/5 of the rows have a value for kilos that is less than or equal to 15.

```
insert into animals values ('California Condor', 'Bird', 15), ('Andean Condo
r', 'Bird', 15)
select name, kind, cume_dist() over (order by kilos) from animals where kind
 = 'Bird';
+-------------------+------+----------------------+
```

```
| name               | kind | cume_dist() OVER(...) |
+--------------------+------+-----------------------+
| Ostrich            | Bird | 1                     |
| Condor             | Bird | 0.8                   |
| California Condor  | Bird | 0.8                   |
| Andean Condor      | Bird | 0.8                   |
| Owl                | Bird | 0.2                   |
+--------------------+------+-----------------------+
```

The following example shows how to use an ORDER BY clause in the outer block to order the result set in case of ties. Here, all the “Bird” rows are together, then in descending order by the result of the CUME_DIST() function, and all tied CUME_DIST() values are ordered by the animal name.

```
select name, kind, cume_dist() over (partition by kind order by kilos) as or
dering
  from animals
where
  kind = 'Bird'
order by kind, ordering desc, name;
+--------------------+------+----------+
| name               | kind | ordering |
+--------------------+------+----------+
| Ostrich            | Bird | 1        |
| Andean Condor      | Bird | 0.8      |
| California Condor  | Bird | 0.8      |
| Condor             | Bird | 0.8      |
| Owl                | Bird | 0.2      |
+--------------------+------+----------+
```

DENSE_RANK

Returns an ascending sequence of integers, starting with 1. The output sequence produces duplicate integers for duplicate values of the ORDER BY expressions. After generating duplicate output values for the “tied” input values, the function continues the sequence with the next higher integer. Therefore, the sequence contains duplicates but no gaps when the input contains duplicates. Starts the sequence over for each group produced by the PARTITIONED BY clause.

Syntax:

```
DENSE_RANK() OVER([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The PARTITION BY clause is optional. The ORDER BY clause is required. The window clause is not allowed.

Usage notes:

Often used for top-N and bottom-N queries. For example, it could produce a “top 10” report including all the items with the 10 highest values, even if several items tied for 1st place.

Similar to ROW_NUMBER and RANK. These functions differ in how they treat duplicate combinations of values.

Examples:

The following example demonstrates how the DENSE_RANK() function identifies where each value “places” in the result set, producing the same result for duplicate values, but with a strict sequence from 1 to the number of groups. For example, when results are ordered by the X column, both 1 values are tied for first; both 2 values are tied for second; and so on.

```
select x, dense_rank() over(order by x) as rank, property from int_t;
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 1  | 1    | square   |
| 1  | 1    | odd      |
```

```
| 2  | 2    | even     |
| 2  | 2    | prime    |
| 3  | 3    | prime    |
| 3  | 3    | odd      |
| 4  | 4    | even     |
| 4  | 4    | square   |
| 5  | 5    | odd      |
| 5  | 5    | prime    |
| 6  | 6    | even     |
| 6  | 6    | perfect  |
| 7  | 7    | lucky    |
| 7  | 7    | lucky    |
| 7  | 7    | lucky    |
| 7  | 7    | odd      |
| 7  | 7    | prime    |
| 8  | 8    | even     |
| 9  | 9    | square   |
| 9  | 9    | odd      |
| 10 | 10   | round    |
| 10 | 10   | even     |
+----+------+----------+
```

The following examples show how the DENSE_RANK() function is affected by the PARTITION property within the ORDER BY clause.

Partitioning by the PROPERTY column groups all the even, odd, and so on values together, and DENSE_RANK() returns the place of each value within the group, producing several ascending sequences.

```
select x, dense_rank() over(partition by property order by x) as rank, prope
rty from int_t;
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 2  | 1    | even     |
| 4  | 2    | even     |
| 6  | 3    | even     |
| 8  | 4    | even     |
| 10 | 5    | even     |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 1  | 1    | odd      |
| 3  | 2    | odd      |
| 5  | 3    | odd      |
| 7  | 4    | odd      |
| 9  | 5    | odd      |
| 6  | 1    | perfect  |
| 2  | 1    | prime    |
| 3  | 2    | prime    |
| 5  | 3    | prime    |
| 7  | 4    | prime    |
| 10 | 1    | round    |
| 1  | 1    | square   |
| 4  | 2    | square   |
| 9  | 3    | square   |
+----+------+----------+
```

Partitioning by the X column groups all the duplicate numbers together and returns the place each value within the group; because each value occurs only 1 or 2 times, DENSE_RANK() designates each X value as either first or second within its group.

```
select x, dense_rank() over(partition by x order by property) as rank, prope
rty from int_t;
```

```
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 1  | 1    | odd      |
| 1  | 2    | square   |
| 2  | 1    | even     |
| 2  | 2    | prime    |
| 3  | 1    | odd      |
| 3  | 2    | prime    |
| 4  | 1    | even     |
| 4  | 2    | square   |
| 5  | 1    | odd      |
| 5  | 2    | prime    |
| 6  | 1    | even     |
| 6  | 2    | perfect  |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 2    | odd      |
| 7  | 3    | prime    |
| 8  | 1    | even     |
| 9  | 1    | odd      |
| 9  | 2    | square   |
| 10 | 1    | even     |
| 10 | 2    | round    |
+----+------+----------+
```

The following example shows how DENSE_RANK() produces a continuous sequence while still allowing for ties. In this case, Croesus and Midas both have the second largest fortune, while Crassus has the third largest. (In the *RANK function* section, you see a similar query with the RANK() function that shows that while Crassus has the third largest fortune, he is the fourth richest person.)

```
select dense_rank() over (order by net_worth desc) as placement, name, net_w
orth from wealth order by placement, name;
+-----------+---------+---------------+
| placement | name    | net_worth     |
+-----------+---------+---------------+
| 1         | Solomon | 2000000000.00 |
| 2         | Croesus | 1000000000.00 |
| 2         | Midas   | 1000000000.00 |
| 3         | Crassus | 500000000.00  |
| 4         | Scrooge | 80000000.00   |
+-----------+---------+---------------+
```

FIRST_VALUE

Returns the expression value from the first row in the window. If your table has null values, you can use the IGNORE NULLS clause to return the first non-null value from the window. This same value is repeated for all result rows for the group. The return value is NULL if the input expression is NULL.

Syntax:

```
FIRST_VALUE(EXPR [IGNORE NULLS]) OVER([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE
 [WINDOW_CLAUSE])
```

The PARTITION BY clause is optional. The ORDER BY clause is required. The window clause is optional.

Usage notes:

If any duplicate values occur in the tuples evaluated by the ORDER BY clause, the result of this function is not deterministic. Consider adding additional ORDER BY columns to ensure consistent ordering.

Examples:

The following example shows a table with a wide variety of country-appropriate greetings. For consistency, we want to standardize on a single greeting for each country. The FIRST_VALUE() function helps to produce a mail merge report where every person from the same country is addressed with the same greeting.

```
select name, country, greeting from mail_merge;
+---------+---------+--------------+
| name    | country | greeting     |
+---------+---------+--------------+
| Pete    | USA     | Hello        |
| John    | USA     | Hi           |
| Boris   | Germany | Guten tag    |
| Michael | Germany | Guten morgen |
| Bjorn   | Sweden  | Hej          |
| Mats    | Sweden  | Tja          |
+---------+---------+--------------+

select country, name,
  first_value(greeting)
    over (partition by country order by name, greeting) as greeting
  from mail_merge;
+---------+---------+-----------+
| country | name    | greeting  |
+---------+---------+-----------+
| Germany | Boris   | Guten tag |
| Germany | Michael | Guten tag |
| Sweden  | Bjorn   | Hej       |
| Sweden  | Mats    | Hej       |
| USA     | John    | Hi        |
| USA     | Pete    | Hi        |
+---------+---------+-----------+
```

Changing the order in which the names are evaluated changes which greeting is applied to each group.

```
select country, name,
  first_value(greeting)
    over (partition by country order by name desc, greeting) as greeting
  from mail_merge;
+---------+---------+--------------+
| country | name    | greeting     |
+---------+---------+--------------+
| Germany | Michael | Guten morgen |
| Germany | Boris   | Guten morgen |
| Sweden  | Mats    | Tja          |
| Sweden  | Bjorn   | Tja          |
| USA     | Pete    | Hello        |
| USA     | John    | Hello        |
+---------+---------+--------------+
```

If you introduce null values in the mail_merge table, the FIRST_VALUE() function will produce a different result with the IGNORE NULLS clause.

```
select * from mail_merge;
+---------+---------+--------------+
| name    | country | greeting     |
+---------+---------+--------------+
| Boris   | Germany | Guten tag    |
| Peng    | China   | Nihao        |
| Mats    | Sweden  | Tja          |
| Bjorn   | Sweden  | Hej          |
| Kei     | Japan   | NULL         |
| Li      | China   | NULL         |
| John    | USA     | Hi           |
```

```
| Pete    | USA     | Hello        |
| Michael | Germany | Guten morgen |
+---------+---------+--------------+

select country, name,
  first_value(greeting ignore nulls)
    over (partition by country order by name,greeting) as greeting
  from mail_merge;
+---------+---------+-----------+
| country | name    | greeting  |
+---------+---------+-----------+
| Japan   | Kei     | NULL      |
| Germany | Boris   | Guten tag |
| Germany | Michael | Guten tag |
| China   | Li      | NULL      |
| China   | Peng    | Nihao     |
| Sweden  | Bjorn   | Hej       |
| Sweden  | Mats    | Hej       |
| USA     | John    | Hi        |
| USA     | Pete    | Hi        |
+---------+---------+-----------+
```

Changing the order in which the names are evaluated changes the result because null values are now encountered in a different order.

```
select country, name,
  first_value(greeting ignore nulls)
    over (partition by country order by name desc, greeting) as greeting
  from mail_merge
+---------+---------+--------------+
| country | name    | greeting     |
+---------+---------+--------------+
| Japan   | Kei     | NULL         |
| China   | Peng    | Nihao        |
| China   | Li      | Nihao        |
| Sweden  | Mats    | Tja          |
| Sweden  | Bjorn   | Tja          |
| USA     | Pete    | Hello        |
| USA     | John    | Hello        |
| Germany | Michael | Guten morgen |
| Germany | Boris   | Guten morgen |
+---------+---------+--------------+
```

LAG

This function returns the value of an expression using column values from a preceding row. You specify an integer offset, which designates a row position some number of rows previous to the current row. Any column references in the expression argument refer to column values from that prior row. Typically, the table contains a time sequence or numeric sequence column that clearly distinguishes the ordering of the rows.

Syntax:

```
LAG (EXPR [, OFFSET] [, DEFAULT])
  OVER ([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

Sometimes used an an alternative to doing a self-join.

Examples:

The following example uses the same stock data created in the *Window clause* section. For each day, the query prints the closing price alongside the previous day's closing price. The first row for each stock symbol has no previous row, so that LAG() value is NULL.

```
select stock_symbol, closing_date, closing_price,
    lag(closing_price,1) over (partition by stock_symbol order by closing_
date) as "yesterday closing"
  from stock_ticker
    order by closing_date;
+--------------+---------------------+---------------+-------------------+
| stock_symbol | closing_date        | closing_price | yesterday closing |
+--------------+---------------------+---------------+-------------------+
| JDR          | 2014-09-13 00:00:00 | 12.86         | NULL              |
| JDR          | 2014-09-14 00:00:00 | 12.89         | 12.86             |
| JDR          | 2014-09-15 00:00:00 | 12.94         | 12.89             |
| JDR          | 2014-09-16 00:00:00 | 12.55         | 12.94             |
| JDR          | 2014-09-17 00:00:00 | 14.03         | 12.55             |
| JDR          | 2014-09-18 00:00:00 | 14.75         | 14.03             |
| JDR          | 2014-09-19 00:00:00 | 13.98         | 14.75             |
+--------------+---------------------+---------------+-------------------+
```

The following example does an arithmetic operation between the current row and a value from the previous row, to produce a delta value for each day. This example also demonstrates how ORDER BY works independently in the different parts of the query. The ORDER BY closing_date in the OVER clause makes the query analyze the rows in chronological order. Then the outer query block uses ORDER BY closing_date DESC to present the results with the most recent date first.

```
select stock_symbol, closing_date, closing_price,
    cast(
      closing_price - lag(closing_price,1) over
        (partition by stock_symbol order by closing_date)
      as decimal(8,2)
    )
    as "change from yesterday"
  from stock_ticker
    order by closing_date desc;
+--------------+---------------------+---------------+---------------------
--+
| stock_symbol | closing_date        | closing_price | change from yesterday
 |
+--------------+---------------------+---------------+-------------------
----+
| JDR          | 2014-09-19 00:00:00 | 13.98         | -0.76
 |
| JDR          | 2014-09-18 00:00:00 | 14.75         | 0.72
 |
| JDR          | 2014-09-17 00:00:00 | 14.03         | 1.47
 |
| JDR          | 2014-09-16 00:00:00 | 12.55         | -0.38
  |
| JDR          | 2014-09-15 00:00:00 | 12.94         | 0.04
 |
| JDR          | 2014-09-14 00:00:00 | 12.89         | 0.03
 |
| JDR          | 2014-09-13 00:00:00 | 12.86         | NULL
 |
+--------------+---------------------+---------------+-------------------
----+
```

Related information:

This function is the converse of the LEAD function.

LAST_VALUE

Returns the expression value from the last row in the window. If your table has null values, you can use the IGNORE NULLS clause to return the last non-null value from the window. This same value is repeated for all result rows for the group. The return value is NULL if the input expression is NULL.

Syntax:

```
LAST_VALUE(EXPR [IGNORE NULLS]) OVER([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE
 [WINDOW_CLAUSE])
```

The PARTITION BY clause is optional. The ORDER BY clause is required. The window clause is optional.

Usage notes:

If any duplicate values occur in the tuples evaluated by the ORDER BY clause, the result of this function is not deterministic. Consider adding additional ORDER BY columns to ensure consistent ordering.

Examples:

The following example uses the same MAIL_MERGE table as in the example for the FIRST_VALUE function. Because the default window when ORDER BY is used is BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, the query requires the UNBOUNDED FOLLOWING to look ahead to subsequent rows and find the last value for each country.

```
select country, name,
  last_value(greeting) over (
    partition by country order by name, greeting
    rows between unbounded preceding and unbounded following
  ) as greeting
  from mail_merge
+---------+---------+--------------+
| country | name    | greeting     |
+---------+---------+--------------+
| Germany | Boris   | Guten morgen |
| Germany | Michael | Guten morgen |
| Sweden  | Bjorn   | Tja          |
| Sweden  | Mats    | Tja          |
| USA     | John    | Hello        |
| USA     | Pete    | Hello        |
+---------+---------+--------------+
```

Introducing null values into the MAIL_MERGE table as in the example for the FIRST VALUE function, the result set changes when you use the IGNORE NULLS clause.

```
select * from mail_merge;
+---------+---------+--------------+
| name    | country | greeting     |
+---------+---------+--------------+
| Kei     | Japan   | NULL         |
| Boris   | Germany | Guten tag    |
| Li      | China   | NULL         |
| Michael | Germany | Guten morgen |
| Bjorn   | Sweden  | Hej          |
| Peng    | China   | Nihao        |
| Pete    | USA     | Hello        |
| Mats    | Sweden  | Tja          |
| John    | USA     | Hi           |
+---------+---------+--------------+

select country, name,
  last_value(greeting ignore nulls) over (
    partition by country order by name, greeting
    rows between unbounded preceding and unbounded following
```

```
  ) as greeting
  from mail_merge;
+---------+---------+--------------+
| country | name    | greeting     |
+---------+---------+--------------+
| Japan   | Kei     | NULL         |
| Germany | Boris   | Guten morgen |
| Germany | Michael | Guten morgen |
| China   | Li      | Nihao        |
| China   | Peng    | Nihao        |
| Sweden  | Bjorn   | Tja          |
| Sweden  | Mats    | Tja          |
| USA     | John    | Hello        |
| USA     | Pete    | Hello        |
+---------+---------+--------------+
```

LEAD

This function returns the value of an expression using column values from a following row. You specify an integer offset, which designates a row position some number of rows after to the current row. Any column references in the expression argument refer to column values from that later row. Typically, the table contains a time sequence or numeric sequence column that clearly distinguishes the ordering of the rows.

Syntax:

```
LEAD (EXPR [, OFFSET] [, DEFAULT])
  OVER ([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

Sometimes used an an alternative to doing a self-join.

Examples:

The following example uses the same stock data created in the Window clause. The query analyzes the closing price for a stock symbol, and for each day evaluates if the closing price for the following day is higher or lower.

```
select stock_symbol, closing_date, closing_price,
  case
    (lead(closing_price,1)
      over (partition by stock_symbol order by closing_date)
        - closing_price) > 0
    when true then "higher"
    when false then "flat or lower"
  end as "trending"
from stock_ticker
  order by closing_date;
+--------------+---------------------+---------------+---------------+
| stock_symbol | closing_date        | closing_price | trending      |
+--------------+---------------------+---------------+---------------+
| JDR          | 2014-09-13 00:00:00 | 12.86         | higher        |
| JDR          | 2014-09-14 00:00:00 | 12.89         | higher        |
| JDR          | 2014-09-15 00:00:00 | 12.94         | flat or lower |
| JDR          | 2014-09-16 00:00:00 | 12.55         | higher        |
| JDR          | 2014-09-17 00:00:00 | 14.03         | higher        |
| JDR          | 2014-09-18 00:00:00 | 14.75         | flat or lower |
| JDR          | 2014-09-19 00:00:00 | 13.98         | NULL          |
+--------------+---------------------+---------------+---------------+
```

Related information:

This function is the converse of the LAG function.

MAX

You can include an OVER clause with a call to this function to use it as an analytic function.

Related Information

MAX function

MIN

You can include an OVER clause with a call to this function to use it as an analytic function.

Related Information

MIN function

NTILE

Returns the “bucket number” associated with each row, between 1 and the value of an expression. For example, creating 100 buckets puts the lowest 1% of values in the first bucket, while creating 10 buckets puts the lowest 10% of values in the first bucket. Each partition can have a different number of buckets.

Syntax:

```
NTILE (EXPR [, OFFSET ...]
  OVER ([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

The “ntile” name is derived from the practice of dividing result sets into fourths (quartile), tenths (decile), and so on. The NTILE() function divides the result set based on an arbitrary percentile value.

The number of buckets must be a positive integer.

The number of items in each bucket is identical or almost so, varying by at most 1. If the number of items does not divide evenly between the buckets, the remaining N items are divided evenly among the first N buckets.

If the number of buckets N is greater than the number of input rows in the partition, then the first N buckets each contain one item, and the remaining buckets are empty.

Examples:

The following example shows divides groups of animals into 4 buckets based on their weight. The ORDER BY ... DESC clause in the OVER() clause means that the heaviest 25% are in the first group, and the lightest 25% are in the fourth group. (The ORDER BY in the outermost part of the query shows how you can order the final result set independently from the order in which the rows are evaluated by the OVER() clause.) Because there are 9 rows in the group, divided into 4 buckets, the first bucket receives the extra item.

```
create table animals (name string, kind string, kilos decimal(9,3));

insert into animals values
  ('Elephant', 'Mammal', 4000), ('Giraffe', 'Mammal', 1200), ('Mouse', 'Mam
mal', 0.020),
  ('Condor', 'Bird', 15), ('Horse', 'Mammal', 500), ('Owl', 'Bird', 2.5),
  ('Ostrich', 'Bird', 145), ('Polar bear', 'Mammal', 700), ('Housecat', 'Mam
mal', 5);

select name, ntile(4) over (order by kilos desc) as quarter
  from animals
order by quarter desc;
+------------+---------+
| name       | quarter |
+------------+---------+
| Owl        | 4       |
| Mouse      | 4       |
| Condor     | 3       |
```

```
| Housecat   | 3       |
| Horse      | 2       |
| Ostrich    | 2       |
| Elephant   | 1       |
| Giraffe    | 1       |
| Polar bear | 1       |
+------------+---------+
```

The following examples show how the PARTITION clause works for the NTILE() function. Here, we divide each kind of animal (mammal or bird) into 2 buckets, the heavier half and the lighter half.

```
select name, kind, ntile(2) over (partition by kind order by kilos desc) as
half
  from animals
order by kind;
+------------+--------+------+
| name       | kind   | half |
+------------+--------+------+
| Ostrich    | Bird   | 1    |
| Condor     | Bird   | 1    |
| Owl        | Bird   | 2    |
| Elephant   | Mammal | 1    |
| Giraffe    | Mammal | 1    |
| Polar bear | Mammal | 1    |
| Horse      | Mammal | 2    |
| Housecat   | Mammal | 2    |
| Mouse      | Mammal | 2    |
+------------+--------+------+
```

Again, the result set can be ordered independently from the analytic evaluation. This next example lists all the animals heaviest to lightest, showing that elephant and giraffe are in the “top half” of mammals by weight, while housecat and mouse are in the “bottom half”.

```
select name, kind, ntile(2) over (partition by kind order by kilos desc) as
half
  from animals
order by kilos desc;
+------------+--------+------+
| name       | kind   | half |
+------------+--------+------+
| Elephant   | Mammal | 1    |
| Giraffe    | Mammal | 1    |
| Polar bear | Mammal | 1    |
| Horse      | Mammal | 2    |
| Ostrich    | Bird   | 1    |
| Condor     | Bird   | 1    |
| Housecat   | Mammal | 2    |
| Owl        | Bird   | 2    |
| Mouse      | Mammal | 2    |
+------------+--------+------+
```

PERCENT_RANK

The PERCENT_RANK() is a window function that calculates the percentile ranking of rows in a result set.

Syntax:

```
PERCENT_RANK (EXPR)
  OVER ([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

Calculates the rank, expressed as a percentage, of each row within a group of rows. If rank is the value for that same row from the RANK() function (from 1 to the total number of rows in the partition group), then the PERCENT_

RANK() value is calculated as (*RANK* - 1) / (*ROWS_IN_GROUP* - 1) . If there is only a single item in the partition group, its PERCENT_RANK() value is 0.

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

This function is similar to the RANK and CUME_DIST() functions: it returns an ascending sequence representing the position of each row within the rows of the same partition group. The actual numeric sequence is calculated differently, and the handling of duplicate (tied) values is different.

The return values range from 0 to 1 inclusive. The first row in each partition group always has the value 0. A NULL value is considered the lowest possible value. In the case of duplicate input values, all the corresponding rows in the result set have an identical value: the lowest PERCENT_RANK() value of those tied rows. (In contrast to CUME_DIST(), where all tied rows have the highest CUME_DIST() value.)

Examples:

The following example uses the same ANIMALS table as the examples for CUME_DIST() and NTILE(), with a few additional rows to illustrate the results where some values are NULL or there is only a single row in a partition group.

```
insert into animals values ('Komodo dragon', 'Reptile', 70);
insert into animals values ('Unicorn', 'Mythical', NULL);
insert into animals values ('Fire-breathing dragon', 'Mythical', NULL);
```

As with CUME_DIST(), there is an ascending sequence for each kind of animal. For example, the “Birds” and “Mammals” rows each have a PERCENT_RANK() sequence that ranges from 0 to 1. The “Reptile” row has a PERCENT_RANK() of 0 because that partition group contains only a single item. Both “Mythical” animals have a PERCENT_RANK() of 0 because a NULL is considered the lowest value within its partition group.

```
select name, kind, percent_rank() over (partition by kind order by kilos) fr
om animals;
+-----------------------+----------+--------------------------+
| name                  | kind     | percent_rank() OVER(...) |
+-----------------------+----------+--------------------------+
| Mouse                 | Mammal   | 0                        |
| Housecat              | Mammal   | 0.2                      |
| Horse                 | Mammal   | 0.4                      |
| Polar bear            | Mammal   | 0.6                      |
| Giraffe               | Mammal   | 0.8                      |
| Elephant              | Mammal   | 1                        |
| Komodo dragon         | Reptile  | 0                        |
| Owl                   | Bird     | 0                        |
| California Condor     | Bird     | 0.25                     |
| Andean Condor         | Bird     | 0.25                     |
| Condor                | Bird     | 0.25                     |
| Ostrich               | Bird     | 1                        |
| Fire-breathing dragon | Mythical | 0                        |
| Unicorn               | Mythical | 0                        |
+-----------------------+----------+--------------------------+
```

RANK

Returns an ascending sequence of integers, starting with 1. The output sequence produces duplicate integers for duplicate values of the ORDER BY expressions. After generating duplicate output values for the “tied” input values, the function increments the sequence by the number of tied values. Therefore, the sequence contains both duplicates and gaps when the input contains duplicates. Starts the sequence over for each group produced by the PARTITIONED BY clause.

Syntax:

```
RANK() OVER([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The PARTITION BY clause is optional. The ORDER BY clause is required. The window clause is not allowed.

Usage notes:

Often used for top-N and bottom-N queries. For example, it could produce a “top 10” report including several items that were tied for 10th place.

Similar to ROW_NUMBER and DENSE_RANK. These functions differ in how they treat duplicate combinations of values.

Examples:

The following example demonstrates how the RANK() function identifies where each value “places” in the result set, producing the same result for duplicate values, and skipping values in the sequence to account for the number of duplicates. For example, when results are ordered by the X column, both 1 values are tied for first; both 2 values are tied for third; and so on.

```
select x, rank() over(order by x) as rank, property from int_t;
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 1  | 1    | square   |
| 1  | 1    | odd      |
| 2  | 3    | even     |
| 2  | 3    | prime    |
| 3  | 5    | prime    |
| 3  | 5    | odd      |
| 4  | 7    | even     |
| 4  | 7    | square   |
| 5  | 9    | odd      |
| 5  | 9    | prime    |
| 6  | 11   | even     |
| 6  | 11   | perfect  |
| 7  | 13   | lucky    |
| 7  | 13   | lucky    |
| 7  | 13   | lucky    |
| 7  | 13   | odd      |
| 7  | 13   | prime    |
| 8  | 18   | even     |
| 9  | 19   | square   |
| 9  | 19   | odd      |
| 10 | 21   | round    |
| 10 | 21   | even     |
+----+------+----------+
```

The following examples show how the RANK() function is affected by the PARTITION property within the ORDER BY clause.

Partitioning by the PROPERTY column groups all the even, odd, and so on values together, and RANK() returns the place of each value within the group, producing several ascending sequences.

```
select x, rank() over(partition by property order by x) as rank, property fr
om int_t;
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 2  | 1    | even     |
| 4  | 2    | even     |
| 6  | 3    | even     |
| 8  | 4    | even     |
| 10 | 5    | even     |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
```

```
| 1  | 1    | odd      |
| 3  | 2    | odd      |
| 5  | 3    | odd      |
| 7  | 4    | odd      |
| 9  | 5    | odd      |
| 6  | 1    | perfect  |
| 2  | 1    | prime    |
| 3  | 2    | prime    |
| 5  | 3    | prime    |
| 7  | 4    | prime    |
| 10 | 1    | round    |
| 1  | 1    | square   |
| 4  | 2    | square   |
| 9  | 3    | square   |
+----+------+----------+
```

Partitioning by the X column groups all the duplicate numbers together and returns the place each value within the group; because each value occurs only 1 or 2 times, RANK() designates each X value as either first or second within its group.

```
select x, rank() over(partition by x order by property) as rank, property fr
om int_t;
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 1  | 1    | odd      |
| 1  | 2    | square   |
| 2  | 1    | even     |
| 2  | 2    | prime    |
| 3  | 1    | odd      |
| 3  | 2    | prime    |
| 4  | 1    | even     |
| 4  | 2    | square   |
| 5  | 1    | odd      |
| 5  | 2    | prime    |
| 6  | 1    | even     |
| 6  | 2    | perfect  |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 4    | odd      |
| 7  | 5    | prime    |
| 8  | 1    | even     |
| 9  | 1    | odd      |
| 9  | 2    | square   |
| 10 | 1    | even     |
| 10 | 2    | round    |
+----+------+----------+
```

The following example shows how a magazine might prepare a list of history's wealthiest people. Croesus and Midas are tied for second, then Crassus is fourth.

```
select rank() over (order by net_worth desc) as rank, name, net_worth from w
ealth order by rank, name;
+------+---------+---------------+
| rank | name    | net_worth     |
+------+---------+---------------+
| 1    | Solomon | 2000000000.00 |
| 2    | Croesus | 1000000000.00 |
| 2    | Midas   | 1000000000.00 |
| 4    | Crassus | 500000000.00  |
| 5    | Scrooge | 80000000.00   |
```

```
+------+---------+---------------+
```

ROW_NUMBER

Returns an ascending sequence of integers, starting with 1. Starts the sequence over for each group produced by the PARTITIONED BY clause. The output sequence includes different values for duplicate input values. Therefore, the sequence never contains any duplicates or gaps, regardless of duplicate input values.

Syntax:

```
ROW_NUMBER() OVER([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

Often used for top-N and bottom-N queries where the input values are known to be unique, or precisely N rows are needed regardless of duplicate values.

Because its result value is different for each row in the result set (when used without a PARTITION BY clause), ROW_NUMBER() can be used to synthesize unique numeric ID values, for example for result sets involving unique values or tuples.

Similar to RANK and DENSE_RANK. These functions differ in how they treat duplicate combinations of values.

Examples:

The following example demonstrates how ROW_NUMBER() produces a continuous numeric sequence, even though some values of X are repeated.

```
select x, row_number() over(order by x, property) as row_number, property fr
om int_t;
+----+------------+----------+
| x  | row_number | property |
+----+------------+----------+
| 1  | 1          | odd      |
| 1  | 2          | square   |
| 2  | 3          | even     |
| 2  | 4          | prime    |
| 3  | 5          | odd      |
| 3  | 6          | prime    |
| 4  | 7          | even     |
| 4  | 8          | square   |
| 5  | 9          | odd      |
| 5  | 10         | prime    |
| 6  | 11         | even     |
| 6  | 12         | perfect  |
| 7  | 13         | lucky    |
| 7  | 14         | lucky    |
| 7  | 15         | lucky    |
| 7  | 16         | odd      |
| 7  | 17         | prime    |
| 8  | 18         | even     |
| 9  | 19         | odd      |
| 9  | 20         | square   |
| 10 | 21         | even     |
| 10 | 22         | round    |
+----+------------+----------+
```

The following example shows how a financial institution might assign customer IDs to some of history's wealthiest figures. Although two of the people have identical net worth figures, unique IDs are required for this purpose. ROW_NUMBER() produces a sequence of five different values for the five input rows.

```
select row_number() over (order by net_worth desc) as account_id, name, net_
worth
  from wealth order by account_id, name;
+------------+---------+---------------+
| account_id | name    | net_worth     |
+------------+---------+---------------+
| 1          | Solomon | 2000000000.00 |
| 2          | Croesus | 1000000000.00 |
| 3          | Midas   | 1000000000.00 |
| 4          | Crassus | 500000000.00  |
| 5          | Scrooge | 80000000.00   |
+------------+---------+---------------+
```

SUM

You can include an OVER clause with a call to this function to use it as an analytic function.

Related Information

SUM function

User-defined functions (UDFs)

User-defined functions (frequently abbreviated as UDFs) let you code your own application logic for processing column values during an Impala query. For example, a UDF could perform calculations using an external math library, combine several column values into one, do geospatial calculations, or other kinds of tests and transformations that are outside the scope of the built-in SQL operators and functions.

You can use UDFs to simplify query logic when producing reports, or to transform data in flexible ways when copying from one table to another with the INSERT ... SELECT syntax.

You might be familiar with this feature from other database products, under names such as stored functions or stored routines.

Impala support for UDFs is available in Impala 1.2 and higher:

- In Impala 1.1, using UDFs in a query required using the Hive shell. (Because Impala and Hive share the same metastore database, you could switch to Hive to run just those queries requiring UDFs, then switch back to Impala.)
- Starting in Impala 1.2, Impala can run both high-performance native code UDFs written in C++, and Java-based Hive UDFs that you might already have written.
- Impala can run scalar UDFs that return a single value for each row of the result set, and user-defined aggregate functions (UDAFs) that return a value based on a set of rows. Currently, Impala does not support user-defined table functions (UDTFs) or window functions.

Related Information

Sample UDF and UDAs for Impala

Memoization

UDF concepts

Runtime environment for UDFs

Installing the UDF development package

Writing UDFs

Writing user-defined aggregate functions (UDAFs)

Building and deploying UDFs

Performance considerations for UDFs
Examples of creating and using UDFs
Security considerations for UDFs
Limitations and restrictions for Impala UDFs

UDF concepts

Depending on your use case, you might write all-new functions, reuse Java UDFs that you have already written for Hive, or port Hive Java UDF code to higher-performance native Impala UDFs in C++. You can code either scalar functions for producing results one row at a time, or more complex aggregate functions for doing analysis across. The following sections discuss these different aspects of working with UDFs.

UDFs and UDAFs

Depending on your use case, the user-defined functions (UDFs) you write might accept or produce different numbers of input and output values:

- The most general kind of user-defined function (the one typically referred to by the abbreviation UDF) takes a single input value and produces a single output value. When used in a query, it is called once for each row in the result set. For example:

```
select customer_name, is_frequent_customer(customer_id) from
 customers;
select obfuscate(sensitive_column) from sensitive_data;
```

- A user-defined aggregate function (UDAF) accepts a group of values and returns a single value. You use UDAFs to summarize and condense sets of rows, in the same style as the built-in COUNT, MAX(), SUM(), and AVG() functions. When called in a query that uses the GROU P BY clause, the function is called once for each combination of GROUP BY values. For example:

```
-- Evaluates multiple rows but returns a single value.
select closest_restaurant(latitude, longitude) from places;

-- Evaluates batches of rows and returns a separate value for
 each batch.
select most_profitable_location(store_id, sales, expenses, tax
_rate, depreciation) from franchise_data group by year;
```

- Currently, Impala does not support other categories of user-defined functions, such as user-defined table functions (UDTFs) or window functions.

Native Impala UDFs

Impala supports UDFs written in C++, in addition to supporting existing Hive UDFs written in Java. Cloudera recommends using C++ UDFs because the compiled native code can yield higher performance, with UDF execution time often 10x faster for a C++ UDF than the equivalent Java UDF.

Using Hive UDFs with Impala

Impala can run Java-based user-defined functions (UDFs), originally written for Hive, with no changes, subject to the following conditions:

- The parameters and return value must all use scalar data types supported by Impala. For example, complex or nested types are not supported.
- Hive/Java UDFs must extend org.apache.hadoop.hive.ql.exec.UDF class.
- Currently, Hive UDFs that accept or return the TIMESTAMP type are not supported.
- Prior to Impala 2.5 the return type must be a “Writable” type such as Text or IntWritable, rather than a Java primitive type such as String or int. Otherwise, the UDF returns NULL. In Impala

2.5 and higher, this restriction is lifted, and both UDF arguments and return values can be Java primitive types.
- Hive UDAFs and UDTFs are not supported.
- Typically, a Java UDF will run several times slower in Impala than the equivalent native UDF written in C++.
- In Impala 2.5 and higher, you can transparently call Hive Java UDFs through Impala, or call Impala Java UDFs through Hive. This feature does not apply to built-in Hive functions. Any Impala Java UDFs created with older versions must be re-created using new CREATE FUNCTION syntax, without any signature for arguments or the return value.

To take full advantage of the Impala architecture and performance features, you can also write Impala-specific UDFs in C++.

For background about Java-based Hive UDFs, see the Hive documentation for UDF. For examples or tutorials for writing such UDFs, search the web for related blog posts.

The ideal way to understand how to reuse Java-based UDFs (originally written for Hive) with Impala is to take some of the Hive built-in functions (implemented as Java UDFs) and take the applicable JAR files through the UDF deployment process for Impala, creating new UDFs with different names:

1. Take a copy of the Hive JAR file containing the Hive built-in functions.
2. Use jar tf *JAR_FILE* to see a list of the classes inside the JAR. You will see names like org/apache/hadoop/hive/ql/udf/UDFLower.class and org/apache/hadoop/hive/ql/udf/UDFOPNegative.class. Make a note of the names of the functions you want to experiment with. When you specify the entry points for the Impala CREATE FUNCTION statement, change the slash characters to dots and strip off the .class suffix, for example org.apache.hadoop.hive.ql.udf.UDFLower and org.apache.hadoop.hive.ql.udf.UDFOPNegative.
3. Copy that file to an HDFS location that Impala can read. (In the examples here, we renamed the file to hive-builtins.jar in HDFS for simplicity.)
4. For each Java-based UDF that you want to call through Impala, issue a CREATE FUNCTION statement, with a LOCATION clause containing the full HDFS path of the JAR file, and a SYMBOL clause with the fully qualified name of the class, using dots as separators and without the .class extension. Remember that user-defined functions are associated with a particular database, so issue a USE statement for the appropriate database first, or specify the SQL function name as *DB_NAME.FUNCTION_NAME*. Use completely new names for the SQL functions, because Impala UDFs cannot have the same name as Impala built-in functions.
5. Call the function from your queries, passing arguments of the correct type to match the function signature. These arguments could be references to columns, arithmetic or other kinds of expressions, the results of CAST functions to ensure correct data types, and so on.

Note:

In Impala 2.9 and higher, you can refresh the user-defined functions (UDFs) that Impala recognizes, at the database level, by running the REFRESH FUNCTIONS statement with the database name as an argument. Java-based UDFs can be added to the metastore database through Hive CREATE FUNCTION statements, and made visible to Impala by subsequently running REFRESH FUNCTIONS. For example:

```
CREATE DATABASE shared_udfs;
USE shared_udfs;
...use CREATE FUNCTION statements in Hive to create so
me Java-based UDFs
   that Impala is not initially aware of...
REFRESH FUNCTIONS shared_udfs;
SELECT udf_created_by_hive(c1) FROM ...
```

Java UDF example: Reusing lower() function

For example, the following `impala-shell` session creates an Impala UDF my_lower() that reuses the Java code for the Hive lower(): built-in function. We cannot call it lower() because Impala does not allow UDFs to have the same name as built-in functions. From SQL, we call the function in a basic way (in a query with no WHERE clause), directly on a column, and on the results of a string expression:

```
[localhost:21000] > create database udfs;
[localhost:21000] > use udfs;
localhost:21000] > create function lower(string) returns string
location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.hi
ve.ql.udf.UDFLower';
ERROR: AnalysisException: Function cannot have the same name as a
 builtin: lower
[localhost:21000] > create function my_lower(string) returns s
tring location '/user/hive/udfs/hive.jar' symbol='org.apache.had
oop.hive.ql.udf.UDFLower';
[localhost:21000] > select my_lower('Some String NOT ALREADY LOWE
RCASE');
+----------------------------------------------------+
| udfs.my_lower('some string not already lowercase') |
+----------------------------------------------------+
| some string not already lowercase                  |
+----------------------------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > create table t2 (s string);
[localhost:21000] > insert into t2 values ('lower'),('UPPER'),('I
nit cap'),('CamelCase');
Inserted 4 rows in 2.28s
[localhost:21000] > select * from t2;
+-----------+
| s         |
+-----------+
| lower     |
| UPPER     |
| Init cap  |
| CamelCase |
+-----------+
Returned 4 row(s) in 0.47s
[localhost:21000] > select my_lower(s) from t2;
+------------------+
| udfs.my_lower(s) |
+------------------+
| lower            |
| upper            |
| init cap         |
| camelcase        |
+------------------+
Returned 4 row(s) in 0.54s
[localhost:21000] > select my_lower(concat('ABC ',s,' XYZ')) f
rom t2;
+-----------------------------------------+
| udfs.my_lower(concat('abc ', s, ' xyz')) |
+-----------------------------------------+
| abc lower xyz                           |
| abc upper xyz                           |
| abc init cap xyz                        |
| abc camelcase xyz                       |
+-----------------------------------------+
Returned 4 row(s) in 0.22s
```

Java UDF example: Reusing negative() function

Here is an example that reuses the Hive Java code for the negative() built-in function. This example demonstrates how the data types of the arguments must match precisely with the function signature. At first, we create an Impala SQL function that can only accept an integer argument. Impala cannot find a matching function when the query passes a floating-point argument, although we can call the integer version of the function by casting the argument. Then we overload the same function name to also accept a floating-point argument.

```
[localhost:21000] > create table t (x int);
[localhost:21000] > insert into t values (1), (2), (4), (100);
Inserted 4 rows in 1.43s
[localhost:21000] > create function my_neg(bigint) returns bigin
t location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.
hive.ql.udf.UDFOPNegative';
[localhost:21000] > select my_neg(4);
+----------------+
| udfs.my_neg(4) |
+----------------+
| -4             |
+----------------+
[localhost:21000] > select my_neg(x) from t;
+----------------+
| udfs.my_neg(x) |
+----------------+
| -2             |
| -4             |
| -100           |
+----------------+
Returned 3 row(s) in 0.60s
[localhost:21000] > select my_neg(4.0);
ERROR: AnalysisException: No matching function with signature:
 udfs.my_neg(FLOAT).
[localhost:21000] > select my_neg(cast(4.0 as int));
+--------------------------------+
| udfs.my_neg(cast(4.0 as int)) |
+--------------------------------+
| -4                             |
+--------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > create function my_neg(double) returns double
 location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.h
ive.ql.udf.UDFOPNegative';
[localhost:21000] > select my_neg(4.0);
+------------------+
| udfs.my_neg(4.0) |
+------------------+
| -4               |
+------------------+
Returned 1 row(s) in 0.11s
```

You can find the sample files mentioned here in the Impala github repo.

Runtime environment for UDFs

By default, Impala copies UDFs into /tmp, and you can configure this location through the --local_library_dir startup flag for the `impalad` daemon.

Installing the UDF development package

To develop UDFs for Impala, download and install the impala-udf-devel package (RHEL-based distributions). This package contains header files, sample source, and build configuration files.

- In Cloudera Archive (at https://archive.cloudera.com/cdh7/), locate the appropriate package installation file for your operating system version.

  Access to Cloudera Runtime packages for production purposes requires authentication.

Note: The UDF development code does not rely on Impala being installed on the same machine. You can write and compile UDFs on a minimal development system, then deploy them on a different one for use with Impala. If you develop UDFs on a server managed by Cloudera Manager through the parcel mechanism, you still install the UDF development kit through the package mechanism; this small standalone package does not interfere with the parcels containing the main Impala code.

When you are ready to start writing your own UDFs, download the sample code and build scripts from the Cloudera sample UDF github. Then see Writing UDFs on page 446 for how to code UDFs, and Examples of creating and using UDFs on page 452 for how to build and run UDFs.

Writing UDFs

Before starting UDF development, make sure to install the development package and download the UDF code samples.

When writing UDFs:

- Keep in mind the data type differences as you transfer values from the high-level SQL to your lower-level UDF code. For example, in the UDF code you might be much more aware of how many bytes different kinds of integers require.
- Use best practices for function-oriented programming: choose arguments carefully, avoid side effects, make each function do a single thing, and so on.

Getting started with UDF coding

To understand the layout and member variables and functions of the predefined UDF data types, examine the header file /usr/include/impala_udf/udf.h:

```
// This is the only Impala header required to develop UDFs and U
DAs. This header
// contains the types that need to be used and the FunctionCont
ext object. The context
// object serves as the interface object between the UDF/UDA and
 the impala process.
```

For the basic declarations needed to write a scalar UDF, see the header file udf-sample.h within the sample build environment, which defines a simple function named AddUdf():

```
#ifndef IMPALA_UDF_SAMPLE_UDF_H
#define IMPALA_UDF_SAMPLE_UDF_H
#include <impala_udf/udf.h>

using namespace impala_udf;

IntVal AddUdf(FunctionContext* context, const IntVal& arg1, const
 IntVal& arg2);
#endif
```

For sample C++ code for a simple function named AddUdf(), see the source file udf-sample.cc within the sample build environment:

```
#include "udf-sample.h"
// In this sample we are declaring a UDF that adds two ints and
 returns an int.
IntVal AddUdf(FunctionContext* context, const IntVal& arg1, const
 IntVal& arg2) {
  if (arg1.is_null || arg2.is_null) return IntVal::null();
  return IntVal(arg1.val + arg2.val);
}

// Multiple UDFs can be defined in the same file
```

Data types for function arguments and return values

Each value that a user-defined function can accept as an argument or return as a result value must map to a SQL data type that you could specify for a table column.

Currently, Impala UDFs cannot accept arguments or return values of the Impala complex types (STRUCT, ARRAY, or MAP).

Each data type has a corresponding structure defined in the C++ and Java header files, with two member fields and some predefined comparison operators and constructors:

- is_null indicates whether the value is NULL or not. val holds the actual argument or return value when it is non-NULL.
- Each struct also defines a null() member function that constructs an instance of the struct with the is_null flag set.
- The built-in SQL comparison operators and clauses such as <, >=, BETWEEN, and ORDER BY all work automatically based on the SQL return type of each UDF. For example, Impala knows how to evaluate BETWEEN 1 AND udf_returning_int(col1) or ORDER BY udf_returning_string(col2) without you declaring any comparison operators within the UDF itself.

  For convenience within your UDF code, each struct defines == and != operators for comparisons with other structs of the same type. These are for typical C++ comparisons within your own code, not necessarily reproducing SQL semantics. For example, if the is_null flag is set in both structs, they compare as equal. That behavior of null comparisons is different from SQL (where NULL == NULL is NULL rather than true), but more in line with typical C++ behavior.
- Each kind of struct has one or more constructors that define a filled-in instance of the struct, optionally with default values.
- Impala cannot process UDFs that accept the composite or nested types as arguments or return them as result values. This limitation applies both to Impala UDFs written in C++ and Java-based Hive UDFs.
- You can overload functions by creating multiple functions with the same SQL name but different argument types. For overloaded functions, you must use different C++ or Java entry point names in the underlying functions.

The data types defined on the C++ side (in /usr/include/impala_udf/udf.h) are:

- IntVal represents an INT column.
- BigIntVal represents a BIGINT column. Even if you do not need the full range of a BIGINT value, it can be useful to code your function arguments as BigIntVal to make it convenient to call the function with different kinds of integer columns and expressions as arguments. Impala automatically casts smaller integer types to larger ones when appropriate, but does not implicitly cast large integer types to smaller ones.
- SmallIntVal represents a SMALLINT column.
- TinyIntVal represents a TINYINT column.
- StringVal represents a STRING column. It has a len field representing the length of the string, and a ptr field pointing to the string data. It has constructors that create a new StringVal struct

based on a null-terminated C-style string, or a pointer plus a length; these new structs still refer to the original string data rather than allocating a new buffer for the data. It also has a constructor that takes a pointer to a FunctionContext struct and a length, that does allocate space for a new copy of the string data, for use in UDFs that return string values.

- BooleanVal represents a BOOLEAN column.
- FloatVal represents a FLOAT column.
- DoubleVal represents a DOUBLE column.
- TimestampVal represents a TIMESTAMP column. It has a date field, a 32-bit integer representing the Gregorian date, that is, the days past the epoch date. It also has a time_of_day field, a 64-bit integer representing the current time of day in nanoseconds.

Variable-length argument lists

UDFs typically take a fixed number of arguments, with each one named explicitly in the signature of your C++ function. Your function can also accept additional optional arguments, all of the same type. For example, you can concatenate two strings, three strings, four strings, and so on. Or you can compare two numbers, three numbers, four numbers, and so on.

To accept a variable-length argument list, code the signature of your function like this:

```
StringVal Concat(FunctionContext* context, const StringVal& sepa
rator,
  int num_var_args, const StringVal* args);
```

In the CREATE FUNCTION statement, after the type of the first optional argument, include ... to indicate it could be followed by more arguments of the same type. For example, the following function accepts a STRING argument, followed by one or more additional STRING arguments:

```
[localhost:21000] > create function my_concat(string, string ...
) returns string location '/user/test_user/udfs/sample.so' symbo
l='Concat';
```

The call from the SQL query must pass at least one argument to the variable-length portion of the argument list.

When Impala calls the function, it fills in the initial set of required arguments, then passes the number of extra arguments and a pointer to the first of those optional arguments.

Handling NULL values

For correctness, performance, and reliability, it is important for each UDF to handle all situations where any NULL values are passed to your function. For example, when passed a NULL, UDFs typically also return NULL. In an aggregate function, which could be passed a combination of real and NULL values, you might make the final value into a NULL (as in CONCAT()), ignore the NULL value (as in AVG()), or treat it the same as a numeric zero or empty string.

Each parameter type, such as IntVal or StringVal, has an is_null Boolean member. Test this flag immediately for each argument to your function, and if it is set, do not refer to the val field of the argument structure. The val field is undefined when the argument is NULL, so your function could go into an infinite loop or produce incorrect results if you skip the special handling for NULL.

If your function returns NULL when passed a NULL value, or in other cases such as when a search string is not found, you can construct a null instance of the return type by using its null() member function.

Memory allocation for UDFs

By default, memory allocated within a UDF is deallocated when the function exits, which could be before the query is finished. The input arguments remain allocated for the lifetime of the function, so you can refer to them in the expressions for your return values. If you use temporary variables

to construct all-new string values, use the StringVal() constructor that takes an initial FunctionContext* argument followed by a length, and copy the data into the newly allocated memory buffer.

Thread-safe work area for UDFs

One way to improve performance of UDFs is to specify the optional PREPARE_FN and CLOSE_FN clauses on the CREATE FUNCTION statement. The “prepare” function sets up a thread-safe data structure in memory that you can use as a work area. The “close” function deallocates that memory. Each subsequent call to the UDF within the same thread can access that same memory area. There might be several such memory areas allocated on the same host, as UDFs are parallelized using multiple threads.

Within this work area, you can set up predefined lookup tables, or record the results of complex operations on data types such as STRING or TIMESTAMP. Saving the results of previous computations rather than repeating the computation each time is an optimization known as Memoization. For example, if your UDF performs a regular expression match or date manipulation on a column that repeats the same value over and over, you could store the last-computed value or a hash table of already-computed values, and do a fast lookup to find the result for subsequent iterations of the UDF.

Each such function must have the signature:

```
void FUNCTION_NAME(impala_udf::FunctionContext*, impala_udf::FunctionContext::FunctionScope)
```

Currently, only THREAD_SCOPE is implemented, not FRAGMENT_SCOPE. See udf.h for details about the scope values.

Error handling for UDFs

To handle errors in UDFs, you call functions that are members of the initial FunctionContext* argument passed to your function.

A UDF can record one or more warnings, for conditions that indicate minor, recoverable problems that do not cause the query to stop. The signature for this function is:

```
bool AddWarning(const char* warning_msg);
```

For a serious problem that requires cancelling the query, a UDF can set an error flag that prevents the query from returning any results. The signature for this function is:

```
void SetError(const char* error_msg);
```

Related Information

Building and deploying UDFs

Writing user-defined aggregate functions (UDAFs)

User-defined aggregate functions (UDAFs or UDAs) are a powerful and flexible category of user-defined functions. If a query processes N rows, calling a UDAF during the query condenses the result set, anywhere from a single value (such as with the SUM or MAX functions), or some number less than or equal to N (as in queries using the GROUP BY or HAVING clause).

The underlying functions for a UDA

A UDAF must maintain a state value across subsequent calls, so that it can accumulate a result across a set of calls, rather than derive it purely from one set of arguments. For that reason, a UDAF is represented by multiple underlying functions:

- An initialization function that sets any counters to zero, creates empty buffers, and does any other one-time setup for a query.

- An update function that processes the arguments for each row in the query result set and accumulates an intermediate result for each node. For example, this function might increment a counter, append to a string buffer, or set flags.
- A merge function that combines the intermediate results from two different nodes.
- A serialize function that flattens any intermediate values containing pointers, and frees any memory allocated during the init, update, and merge phases.
- A finalize function that either passes through the combined result unchanged, or does one final transformation.

In the SQL syntax, you create a UDAF by using the statement CREATE AGGREGATE FUNCTION. You specify the entry points of the underlying C++ functions using the clauses INIT_FN, UPDATE_FN, MERGE_FN, SERIALIZE_FN, and FINALIZE_FN.

For convenience, you can use a naming convention for the underlying functions and Impala automatically recognizes those entry points. Specify the UPDATE_FN clause, using an entry point name containing the string update or Update. When you omit the other _FN clauses from the SQL statement, Impala looks for entry points with names formed by substituting the update or Update portion of the specified name.

uda-sample.h:

See this file online at: uda-sample.h

uda-sample.cc:

See this file online at: uda-sample.cc

Intermediate results for UDAs

A user-defined aggregate function might produce and combine intermediate results during some phases of processing, using a different data type than the final return value. For example, if you implement a function similar to the built-in AVG() function, it must keep track of two values, the number of values counted and the sum of those values. Or, you might accumulate a string value over the course of a UDA, then in the end return a numeric or Boolean result.

In such a case, specify the data type of the intermediate results using the optional INTERMEDIATE *TYPE_NAME* clause of the CREATE AGGREGATE FUNCTION statement. If the intermediate data is a typeless byte array (for example, to represent a C++ struct or array), specify the type name as CHAR(*N*), with *N* representing the number of bytes in the intermediate result buffer.

For an example of this technique, see the trunc_sum() aggregate function, which accumulates intermediate results of type DOUBLE and returns BIGINT at the end. View the appropriate CREATE FUNCTION statement and the implementation of the underlying TruncSum*() functions on Github.

- test_udfs.py
- test-udas.cc

Building and deploying UDFs

This section explains the steps to compile Impala UDFs from C++ source code, and deploy the resulting libraries for use in Impala queries.

Impala UDF development package ships with a sample build environment for UDFs, that you can study, experiment with, and adapt for your own use.

To build the sample environment:

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. From the **Overview** page, go to **Resources and Downloads** and click See More Impala UDF SDK .

3. Download the impala-udf-devel zip file.
4. SSH into the host on which you want to install the install the UDF environment.
5. Run the following commands:

```
cmake .
make
```

The `cmake` configuration command reads the file CMakeLists.txt and generates a Makefile customized for your particular directory paths. Then the `make` command runs the actual build steps based on the rules in the Makefile.

Impala loads the shared library from an HDFS location. After building a shared library containing one or more UDFs, use hdfs dfs or hadoop fs commands to copy the binary file to an HDFS location readable by Impala.

The final step in deployment is to issue a CREATE FUNCTION statement in the `impala-shell` interpreter to make Impala aware of the new function. Because each function is associated with a particular database, always issue a USE statement to the appropriate database before creating a function, or specify a fully qualified name, that is, CREATE FUNCTION *DB_NAME.FUNCTION_NAME*.

As you update the UDF code and redeploy updated versions of a shared library, use DROP FUNCTION and CREATE FUNCTION to let Impala pick up the latest version of the code.

Note:

In Impala 2.5 and higher, Impala UDFs and UDAs written in C++ are persisted in the metastore database. Java UDFs are also persisted, if they were created with the new CREATE FUNCTION syntax for Java UDFs, where the Java function argument and return types are omitted. Java-based UDFs created with the old CREATE FUNCTION syntax do not persist across restarts because they are held in the memory of the `catalogd` daemon. Until you re-create such Java UDFs using the new CREATE FUNCTION syntax, you must reload those Java-based UDFs by running the original CREATE FUNCTION statements again each time you restart the `catalogd` daemon. Prior to Impala 2.5 the requirement to reload functions after a restart applied to both C++ and Java functions.

See CREATE FUNCTION statement on page 134 and DROP FUNCTION statement on page 174 for the new syntax for the persistent Java UDFs.

Prerequisites for the build environment are:

1. Install the packages using the appropriate package installation command for your Linux distribution.

```
sudo yum install gcc-c++ cmake boost-devel
sudo yum install impala-udf-devel
# The package name on Ubuntu and Debian is impala-udf-dev.
```

2. Download the UDF sample code:

```
git clone https://github.com/cloudera/impala-udf-samples
cd impala-udf-samples &amp;amp;&amp;amp; cmake . &amp;amp;&amp;amp; make
```

3. Unpack the sample code in udf_samples.tar.gz and use that as a template to set up your build environment.

To build the original samples:

```
# Process CMakeLists.txt and set up appropriate Makefiles.
cmake .
# Generate shared libraries from UDF and UDAF sample code,
# udf_samples/libudfsample.so and udf_samples/libudasample.so
make
```

The sample code to examine, experiment with, and adapt is in these files:

- udf-sample.h: Header file that declares the signature for a scalar UDF (AddUDF).

- udf-sample.cc: Sample source for a simple UDF that adds two integers. Because Impala can reference multiple function entry points from the same shared library, you could add other UDF functions in this file and add their signatures to the corresponding header file.
- udf-sample-test.cc: Basic unit tests for the sample UDF.
- uda-sample.h: Header file that declares the signature for sample aggregate functions. The SQL functions will be called COUNT, AVG, and STRINGCONCAT. Because aggregate functions require more elaborate coding to handle the processing for multiple phases, there are several underlying C++ functions such as CountInit, AvgU pdate, and StringConcatFinalize.
- uda-sample.cc: Sample source for simple UDAFs that demonstrate how to manage the state transitions as the underlying functions are called during the different phases of query processing.
  - The UDAF that imitates the COUNT function keeps track of a single incrementing number; the merge functions combine the intermediate count values from each Impala node, and the combined number is returned verbatim by the finalize function.
  - The UDAF that imitates the AVG function keeps track of two numbers, a count of rows processed and the sum of values for a column. These numbers are updated and merged as with COUNT, then the finalize function divides them to produce and return the final average value.
  - The UDAF that concatenates string values into a comma-separated list demonstrates how to manage storage for a string that increases in length as the function is called for multiple rows.
- uda-sample-test.cc: basic unit tests for the sample UDAFs.

Performance considerations for UDFs

Because a UDF typically processes each row of a table, potentially being called billions of times, the performance of each UDF is a critical factor in the speed of the overall ETL or ELT pipeline. Tiny optimizations you can make within the function body can pay off in a big way when the function is called over and over when processing a huge result set.

Examples of creating and using UDFs

This section demonstrates how to create and use all kinds of user-defined functions (UDFs).

For downloadable examples that you can experiment with, adapt, and use as templates for your own functions, see the Cloudera sample UDF github. You must have already installed the appropriate header files, as explained in *Building and deploying UDFs*.

Sample C++ UDFs: HasVowels, CountVowels, StripVowels

This example shows 3 separate UDFs that operate on strings and return different data types. In the C++ code, the functions are HasVowels() (checks if a string contains any vowels), CountVowels() (returns the number of vowels in a string), and StripVowels() (returns a new string with vowels removed).

First, we add the signatures for these functions to udf-sample.h in the demo build environment:

```
BooleanVal HasVowels(FunctionContext* context, const StringVal&
input);
IntVal CountVowels(FunctionContext* context, const StringVal& ar
g1);
StringVal StripVowels(FunctionContext* context, const StringVal&
 arg1);
```

Then, we add the bodies of these functions to udf-sample.cc:

```
BooleanVal HasVowels(FunctionContext* context, const StringVal&
input)
```

```
{
        if (input.is_null) return BooleanVal::null();

        int index;
        uint8_t *ptr;

        for (ptr = input.ptr, index = 0; index <= input.len; i
ndex++, ptr++)
        {
                uint8_t c = tolower(*ptr);
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o'
 || c == 'u')
                {
                        return BooleanVal(true);
                }
        }
        return BooleanVal(false);
}

IntVal CountVowels(FunctionContext* context, const StringVal&
arg1)
{
        if (arg1.is_null) return IntVal::null();

        int count;
        int index;
        uint8_t *ptr;

        for (ptr = arg1.ptr, count = 0, index = 0; index <= arg1.
len; index++, ptr++)
        {
                uint8_t c = tolower(*ptr);
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o'
 || c == 'u')
                {
                        count++;
                }
        }
        return IntVal(count);
}

StringVal StripVowels(FunctionContext* context, const StringVal&
 arg1)
{
        if (arg1.is_null) return StringVal::null();

        int index;
        std::string original((const char *)arg1.ptr,arg1.len);
        std::string shorter("");

        for (index = 0; index < original.length(); index++)
        {
                uint8_t c = original[index];
                uint8_t l = tolower(c);

                if (l == 'a' || l == 'e' || l == 'i' || l == 'o'
 || l == 'u')
                {
                        ;
                }
                else
                {
                    shorter.append(1, (char)c);
                }
```

```
        }
// The modified string is stored in 'shorter', which is destroyed
 when this function ends. We need to make a string val
// and copy the contents.
        StringVal result(context, shorter.size()); // Only the ve
rsion of the ctor that takes a context object allocates new memo
ry
        memcpy(result.ptr, shorter.c_str(), shorter.size());
        return result;
}
```

We build a shared library, libudfsample.so, and put the library file into HDFS where Impala can read it:

```
$ make
[  0%] Generating udf_samples/uda-sample.ll
[ 16%] Built target uda-sample-ir
[ 33%] Built target udasample
[ 50%] Built target uda-sample-test
[ 50%] Generating udf_samples/udf-sample.ll
[ 66%] Built target udf-sample-ir
Scanning dependencies of target udfsample
[ 83%] Building CXX object CMakeFiles/udfsample.dir/udf-sample.o
Linking CXX shared library udf_samples/libudfsample.so
[ 83%] Built target udfsample
Linking CXX executable udf_samples/udf-sample-test
[100%] Built target udf-sample-test
$ hdfs dfs -put ./udf_samples/libudfsample.so /user/hive/udfs/li
budfsample.so
```

Finally, we go into the `impala-shell` interpreter where we set up some sample data, issue CREATE FUNCTION statements to set up the SQL function names, and call the functions in some queries:

```
[localhost:21000] > create database udf_testing;
[localhost:21000] > use udf_testing;

[localhost:21000] > create function has_vowels (string) returns b
oolean location '/user/hive/udfs/libudfsample.so' symbol='HasVow
els';
[localhost:21000] > select has_vowels('abc');
+------------------------+
| udfs.has_vowels('abc') |
+------------------------+
| true                   |
+------------------------+
Returned 1 row(s) in 0.13s
[localhost:21000] > select has_vowels('zxcvbnm');
+----------------------------+
| udfs.has_vowels('zxcvbnm') |
+----------------------------+
| false                      |
+----------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select has_vowels(null);
+-----------------------+
| udfs.has_vowels(null) |
+-----------------------+
| NULL                  |
+-----------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > select s, has_vowels(s) from t2;
```

```
+-----------+--------------------+
| s         | udfs.has_vowels(s) |
+-----------+--------------------+
| lower     | true               |
| UPPER     | true               |
| Init cap  | true               |
| CamelCase | true               |
+-----------+--------------------+
Returned 4 row(s) in 0.24s
[localhost:21000] > create function count_vowels (string) retur
ns int location '/user/hive/udfs/libudfsample.so' symbol='CountV
owels';
[localhost:21000] > select count_vowels('cat in the hat');
+-------------------------------------+
| udfs.count_vowels('cat in the hat') |
+-------------------------------------+
| 4                                   |
+-------------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select s, count_vowels(s) from t2;
+-----------+----------------------+
| s         | udfs.count_vowels(s) |
+-----------+----------------------+
| lower     | 2                    |
| UPPER     | 2                    |
| Init cap  | 3                    |
| CamelCase | 4                    |
+-----------+----------------------+
Returned 4 row(s) in 0.23s
[localhost:21000] > select count_vowels(null);
+--------------------------+
| udfs.count_vowels(null) |
+--------------------------+
| NULL                     |
+--------------------------+
Returned 1 row(s) in 0.12s

[localhost:21000] > create function strip_vowels (string) returns
 string location '/user/hive/udfs/libudfsample.so' symbol='Strip
Vowels';
[localhost:21000] > select strip_vowels('abcdefg');
+------------------------------+
| udfs.strip_vowels('abcdefg') |
+------------------------------+
| bcdfg                        |
+------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > select strip_vowels('ABCDEFG');
+------------------------------+
| udfs.strip_vowels('abcdefg') |
+------------------------------+
| BCDFG                        |
+------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select strip_vowels(null);
+--------------------------+
| udfs.strip_vowels(null) |
+--------------------------+
| NULL                     |
+--------------------------+
Returned 1 row(s) in 0.16s
[localhost:21000] > select s, strip_vowels(s) from t2;
+-----------+----------------------+
| s         | udfs.strip_vowels(s) |
```

```
+-----------+----------------------+
| lower     | lwr                  |
| UPPER     | PPR                  |
| Init cap  | nt cp                |
| CamelCase | CmlCs                |
+-----------+----------------------+
Returned 4 row(s) in 0.24s
```

Sample C++ UDA: SumOfSquares

```
[localhost:21000] > insert overwrite sos values (1, 1), (2, 0),
(3, 1), (4, 0);
Inserted 4 rows in 1.24s

[localhost:21000] > -- Compute 1 squared + 3 squared, and 2 sq
uared + 4 squared;
[localhost:21000] > select y, sum_of_squares(x) from sos group by
 y;
+---+------------------------+
| y | udfs.sum_of_squares(x) |
+---+------------------------+
| 1 | 10                     |
| 0 | 20                     |
+---+------------------------+
Returned 2 row(s) in 0.43s
```

This example demonstrates a user-defined aggregate function (UDA) that produces the sum of the squares of its input values.

The coding for a UDA is a little more involved than a scalar UDF, because the processing is split into several phases, each implemented by a different function. Each phase is relatively straightforward: the “update” and “merge” phases, where most of the work is done, read an input value and combine it with some accumulated intermediate value.

As in our sample UDF from the previous example, we add function signatures to a header file (in this case, uda-sample.h). Because this is a math-oriented UDA, we make two versions of each function, one accepting an integer value and the other accepting a floating-point value.

```
void SumOfSquaresInit(FunctionContext* context, BigIntVal* val);
void SumOfSquaresInit(FunctionContext* context, DoubleVal* val);

void SumOfSquaresUpdate(FunctionContext* context, const BigIntVal
& input, BigIntVal* val);
void SumOfSquaresUpdate(FunctionContext* context, const Double
Val& input, DoubleVal* val);

void SumOfSquaresMerge(FunctionContext* context, const BigIntV
al& src, BigIntVal* dst);
void SumOfSquaresMerge(FunctionContext* context, const DoubleV
al& src, DoubleVal* dst);

BigIntVal SumOfSquaresFinalize(FunctionContext* context, const Bi
gIntVal& val);
DoubleVal SumOfSquaresFinalize(FunctionContext* context, const Do
ubleVal& val);
```

We add the function bodies to a C++ source file (in this case, uda-sample.cc):

```
void SumOfSquaresInit(FunctionContext* context, BigIntVal* val) {
  val->is_null = false;
  val->val = 0;
}
```

```
void SumOfSquaresInit(FunctionContext* context, DoubleVal* val) {
  val->is_null = false;
  val->val = 0.0;
}

void SumOfSquaresUpdate(FunctionContext* context, const BigIntVal
& input, BigIntVal* val) {
  if (input.is_null) return;
  val->val += input.val * input.val;
}
void SumOfSquaresUpdate(FunctionContext* context, const DoubleVal
& input, DoubleVal* val) {
  if (input.is_null) return;
  val->val += input.val * input.val;
}

void SumOfSquaresMerge(FunctionContext* context, const BigIntVal&
 src, BigIntVal* dst) {
  dst->val += src.val;
}
void SumOfSquaresMerge(FunctionContext* context, const DoubleV
al& src, DoubleVal* dst) {
  dst->val += src.val;
}
BigIntVal SumOfSquaresFinalize(FunctionContext* context, const
BigIntVal& val) {
  return val;
}
DoubleVal SumOfSquaresFinalize(FunctionContext* context, const
 DoubleVal& val) {
  return val;
}
```

As with the sample UDF, we build a shared library and put it into HDFS:

```
$ make
[  0%] Generating udf_samples/uda-sample.ll
[ 16%] Built target uda-sample-ir
Scanning dependencies of target udasample
[ 33%] Building CXX object CMakeFiles/udasample.dir/uda-sample.o
Linking CXX shared library udf_samples/libudasample.so
[ 33%] Built target udasample
Scanning dependencies of target uda-sample-test
[ 50%] Building CXX object CMakeFiles/uda-sample-test.dir/uda-s
ample-test.o
Linking CXX executable udf_samples/uda-sample-test
[ 50%] Built target uda-sample-test
[ 50%] Generating udf_samples/udf-sample.ll
[ 66%] Built target udf-sample-ir
[ 83%] Built target udfsample
[100%] Built target udf-sample-test
$ hdfs dfs -put ./udf_samples/libudasample.so /user/hive/udfs/li
budasample.so
```

To create the SQL function, we issue a CREATE AGGREGATE FUNCTION statement and specify the underlying C++ function names for the different phases:

```
[localhost:21000] > use udf_testing;

[localhost:21000] > create table sos (x bigint, y double);
[localhost:21000] > insert into sos values (1, 1.1), (2, 2.2),
(3, 3.3), (4, 4.4);
Inserted 4 rows in 1.10s
```

```
[localhost:21000] > create aggregate function sum_of_squares(b
igint) returns bigint
  > location '/user/hive/udfs/libudasample.so'
  > init_fn='SumOfSquaresInit'
  > update_fn='SumOfSquaresUpdate'
  > merge_fn='SumOfSquaresMerge'
  > finalize_fn='SumOfSquaresFinalize';
[localhost:21000] > -- Compute the same value using literals or
 the UDA;
[localhost:21000] > select 1*1 + 2*2 + 3*3 + 4*4;
+-------------------------------+
| 1 * 1 + 2 * 2 + 3 * 3 + 4 * 4 |
+-------------------------------+
| 30                            |
+-------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select sum_of_squares(x) from sos;
+------------------------+
| udfs.sum_of_squares(x) |
+------------------------+
| 30                     |
+------------------------+
Returned 1 row(s) in 0.35s
```

Until we create the overloaded version of the UDA, it can only handle a single data type. To allow it to handle DOUBLE as well as BIGINT, we issue another CREATE AGGREGATE FUNCTION statement:

```
[localhost:21000] > select sum_of_squares(y) from sos;
ERROR: AnalysisException: No matching function with signature: ud
fs.sum_of_squares(DOUBLE).

[localhost:21000] > create aggregate function sum_of_squares(dou
ble) returns double
  > location '/user/hive/udfs/libudasample.so'
  > init_fn='SumOfSquaresInit'
  > update_fn='SumOfSquaresUpdate'
  > merge_fn='SumOfSquaresMerge'
  > finalize_fn='SumOfSquaresFinalize';

[localhost:21000] > -- Compute the same value using literals or t
he UDA;
[localhost:21000] > select 1.1*1.1 + 2.2*2.2 + 3.3*3.3 + 4.4*4.4;
+-----------------------------------------------+
| 1.1 * 1.1 + 2.2 * 2.2 + 3.3 * 3.3 + 4.4 * 4.4 |
+-----------------------------------------------+
| 36.3                                          |
+-----------------------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select sum_of_squares(y) from sos;
+------------------------+
| udfs.sum_of_squares(y) |
+------------------------+
| 36.3                   |
+------------------------+
Returned 1 row(s) in 0.35s
```

Typically, you use a UDA in queries with GROUP BY clauses, to produce a result set with a separate aggregate value for each combination of values from the GROUP BY clause. Let's change our sample table to use 0 to indicate rows containing even values, and 1 to flag rows containing

odd values. Then the GROUP BY query can return two values, the sum of the squares for the even values, and the sum of the squares for the odd values:

Related Information

Building and deploying UDFs

Security considerations for UDFs

When the Impala authorization feature is enabled:

- To call a UDF in a query, you must have the required read privilege for any databases and tables used in the query.
- The CREATE FUNCTION statement requires:
  - The CREATE privilege on the database.
  - The ALL privilege on two URIs where the URIs are:
    - The JAR file on the file system. For example:

```
GRANT ALL ON URI 'file:///PATH_TO_MY.JAR' TO ROLE MY_ROLE;
```

    - The JAR on HDFS. For example:

```
GRANT ALL ON URI 'hdfs:///PATH/TO/JAR' TO ROLE MY_ROLE
```

Limitations and restrictions for Impala UDFs

The following limitations and restrictions apply to Impala UDFs in the current release.

- Impala does not support Hive UDFs that accept or return composite or nested types, or other types not available in Impala tables.
- The Hive current_user() function cannot be called from a Java UDF through Impala.
- All Impala UDFs must be deterministic, that is, produce the same output each time when passed the same argument values. For example, an Impala UDF must not call functions such as rand() to produce different values for each invocation. It must not retrieve data from external sources, such as from disk or over the network.
- An Impala UDF must not spawn other threads or processes.
- Prior to Impala 2.5 when the `catalogd` process is restarted, all UDFs become undefined and must be reloaded. In Impala 2.5 and higher, this limitation only applies to older Java UDFs. Re-create those UDFs using the new CREATE FUNCTION syntax for Java UDFs, which excludes the function signature, to remove the limitation entirely.
- Impala currently does not support user-defined table functions (UDTFs).
- The CHAR and VARCHAR types cannot be used as input arguments or return values for UDFs.

SQL transactions in Impala

A transaction is a single logical operation on the data. Impala supports transactions that satisfy a level of consistency that improves the integrity and reliability of the data before and after a transaction.

Specifically, Impala provides atomicity and isolation of insert operations on transactional tables. A single table insert is either committed in full or not committed, and the results of the insert operation are not visible to other query operations until the operation is committed.

For single table, the inserts are ordered, so if Impala doesn't see a committed insert, it won't see any insert committed after it.

For insert-only transactional tables, you can perform the following statements: CREATE TABLE, DROP TABLE, TRUNCATE, INSERT, SELECT

All transactions in Impala automatically commit at the end of the statement. Currently, Impala does not support multi-statement transactions.

Insert-only tables must be the managed and file-format based tables, such as Parquet, Avro, and text.

Note: Impala does not support changing transactional properties of tables. For example, you cannot alter a transactional table to a non-transactional table.

Related Information

CREATE TABLE statement

Impala reserved words

Impala reserves a set of keywords that cannot be used directly as identifiers in Impala SQL. You must be aware of the list of reserved words to avoid name conflicts with any of your table, columns, or other object names.

If you need to use it as an identifier, you must quote it with backticks. For example:

- CREATE TABLE select (x INT): fails
- CREATE TABLE `select` (x INT): succeeds

Because different database systems have different sets of reserved words, and the reserved words change from release to release, carefully consider database, table, and column names to ensure maximum compatibility between products and versions.

Also consider whether your object names are the same as any Hive keywords, and rename or quote any that conflict as you might switch between Impala and Hive when doing analytics and ETL.

To future-proof your code, you should avoid additional words in case they become reserved words if Impala adds features in later releases. This kind of planning can also help to avoid name conflicts in case you port SQL from other systems that have different sets of reserved words. The Future Keyword column in the table below indicates those additional words that you should avoid for table, column, or other object names, even though they are not currently reserved by Impala.

The following is a summary of the process for deciding whether a particular SQL 2016 word is to be reserved in Impala.

- By default, Impala targets to have the same list of reserved words as SQL 2016.
- At the same time, to be compatible with earlier versions of Impala and to avoid breaking existing tables/workloads, Impala built-in function names are removed from the reserved words list, e.g. COUNT, AVG, as Impala generally does not need to reserve the names of built-in functions for parsing to work.
- For those remaining SQL 2016 reserved words, if a word is likely to be in-use by users of older Impala versions and if there is a low chance of Impala needing to reserve that word in the future, then the word is not reserved.
- Otherwise, the word is reserved in Impala.

The following table lists the reserved words in Impala.

Keyword	Reserved in SQL:2016	Reserved in Impala 3.0 and higher	Future Keyword
abs	X		

acos	X		
add		X	
aggregate		X	
all	X	X	
allocate	X	X	
alter	X	X	
analytic		X	
and	X	X	
anti		X	
any	X	X	
api_version		X	
are	X	X	
array	X	X	
array_agg	X	X	
array_max_cardinality	X	X	
as	X	X	
asc		X	
asensitive	X	X	
asin	X		
asymmetric	X	X	
at	X	X	
atan	X		
atomic	X	X	
authorization	X	X	
avg	X		
avro		X	
backup			X
begin	X		X
begin_frame	X	X	
begin_partition	X	X	
between	X	X	
bigint	X	X	
binary	X	X	
blob	X	X	
block_size		X	
boolean	X	X	
both	X	X	
break			X
browse			X
bulk			X

by	X	X	
cache			
cached		X	
call	X		
called	X	X	
cardinality	X	X	
cascade		X	
cascaded	X	X	
case	X	X	
cast	X	X	
ceil	X		
ceiling	X		
change		X	
char	X	X	
char_length	X		
character	X	X	
character_length	X		
check	X		X
checkpoint			X
class		X	
classifier	X		
clob	X	X	
close	X		X
close_fn		X	
clustered			X
coalesce	X		X
collate	X	X	
collect	X	X	
column	X	X	
columns		X	
comment		X	
commit	X	X	
compression		X	
compute		X	
condition	X	X	
conf			
connect	X	X	
constraint	X	X	
contains	X	X	
continue			X

convert	X	X	
copy	X	X	
corr	X	X	
corresponding	X	X	
cos	X		
cosh	X		
count	X		
covar_pop	X	X	
covar_samp	X	X	
create	X	X	
cross	X	X	
cube	X	X	
cume_dist	X		
current	X	X	
current_catalog	X		
current_date	X	X	
current_default_transform_group	X	X	
current_path	X	X	
current_role	X	X	
current_row	X	X	
current_schema	X	X	
current_time	X	X	
current_timestamp	X		X
current_transform_group_for_type	X	X	
current_user	X		X
cursor	X	X	
cycle	X	X	
data		X	
database		X	
databases		X	
date	X	X	
datetime		X	
day	X		
dayofweek			
dbcc			X
deallocate	X	X	
dec	X	X	
decfloat	X	X	
decimal	X	X	
declare	X	X	

default	X	X	
define	X	X	
delete	X	X	
delimited		X	
dense_rank	X		
deny			X
deref	X	X	
desc		X	
describe	X	X	
deterministic	X	X	
disconnect	X	X	
disk			X
distinct	X	X	
distributed			X
div		X	
double	X	X	
drop	X	X	
dump			X
dynamic	X	X	
each	X	X	
element	X	X	
else	X	X	
empty	X	X	
encoding		X	
end	X	X	
end-exec	X	X	
end_frame	X	X	
end_partition	X	X	
equals	X	X	
errlvl			X
escape	X	X	
escaped		X	
every	X	X	
except	X	X	
exchange			
exec	X	X	
execute	X	X	
exists	X	X	
exit			X
exp	X		

explain		X	
extended		X	
external	X	X	
extract	X		
false	X	X	
fetch	X	X	
fields		X	
file			X
filefactor			X
fileformat		X	
files		X	
filter	X	X	
finalize_fn		X	
first		X	
first_value	X		
float	X	X	
floor	X		
following		X	
for	X	X	
foreign	X	X	
format		X	
formatted		X	
frame_row	X	X	
free	X	X	
freetext			X
from	X	X	
full	X	X	
function	X	X	
functions		X	
fusion	X	X	
get	X	X	
global	X	X	
goto			X
grant	X	X	
group	X	X	
grouping	X	X	
groups	X	X	
hash		X	
having	X	X	
hold	X	X	

holdlock			X
hour	X		
iceberg		X[1]	
identity	X		X
if		X	
ignore		X	
ilike		X	
import			
in	X	X	
incremental		X	
index			X
indicator	X	X	
init_fn		X	
initial	X	X	
inner	X	X	
inout	X	X	
inpath		X	
insensitive	X	X	
insert	X	X	
int	X	X	
integer	X	X	
intermediate		X	
intersect	X	X	
intersection	X	X	
interval	X	X	
into	X	X	
invalidate		X	
iregexp		X	
is	X	X	
join	X	X	
json_array	X	X	
json_arrayagg	X	X	
json_exists	X	X	
json_object	X	X	
json_objectagg	X	X	
json_query	X	X	
json_table	X	X	
json_table_primitive	X	X	
json_value	X	X	

[1] Impala 4.0 and higher

key			X
kill			X
kudu		X	
lag	X		
language	X		
large	X	X	
last		X	
last_value	X		
lateral	X	X	
lead	X		
leading	X	X	
left	X	X	
less			
like	X	X	
like_regex	X	X	
limit		X	
lineno			X
lines		X	
listagg	X	X	
ln	X		
load		X	
local	X	X	
localtime	X		
localtimestamp	X	X	
location		X	
log	X		
log10	X	X	
lower	X		
macro			
map		X	
match	X	X	
match_number	X	X	
match_recognize	X	X	
matches	X	X	
max	X		
member	X		
merge	X	X	
merge_fn		X	
metadata		X	
method	X	X	

min	X		
minute	X		
mod	X		
modifies	X	X	
module	X		
month	X		
more			
multiset	X	X	
national	X	X	
natural	X	X	
nchar	X	X	
nclob	X	X	
new	X		
no	X	X	
nocheck			X
nonclustered			X
none	X	X	
normalize	X	X	
not	X	X	
nth_value	X	X	
ntile	X		
null	X	X	
nullif	X		X
nulls		X	
numeric	X	X	
occurrences_regex	X	X	
octet_length	X	X	
of	X	X	
off			X
offset	X	X	
offsets			X
old	X		
omit	X	X	
on	X	X	
one	X	X	
only	X	X	
open	X		X
option			X
or	X	X	
order	X	X	

out	X	X	
outer	X	X	
over	X	X	
overlaps	X	X	
overlay	X	X	
overwrite		X	
parameter	X		
parquet		X	
parquetfile		X	
partialscan			
partition	X	X	
partitioned		X	
partitions		X	
pattern	X	X	
per	X	X	
percent	X	X	
percent_rank	X		
percentile_cont	X	X	
percentile_disc	X	X	
period	X		
pivot			X
plan			X
portion	X	X	
position	X	X	
position_regex	X	X	
power	X		
precedes	X	X	
preceding		X	
precision	X		X
prepare	X	X	
prepare_fn		X	
preserve			
primary	X	X	
print			X
proc			X
procedure	X	X	
produced		X	
ptf	X	X	
public			X
purge		X	

raiseerror			X
range	X	X	
rank	X		
rcfile		X	
read			X
reads	X	X	
readtext			X
real	X	X	
reconfigure			X
recover		X	
recursive	X	X	
reduce			
ref	X	X	
references	X	X	
referencing	X	X	
refresh		X	
regexp		X	
regr_avgx	X	X	
regr_avgy	X	X	
regr_count	X	X	
regr_intercept	X	X	
regr_r2	X	X	
regr_slope	X	X	
regr_sxx	X	X	
regr_sxy	X	X	
regr_syy	X	X	
release	X	X	
rename		X	
repeatable		X	
replace		X	
replication		X	
restore			X
restrict		X	
result	X		
return	X		X
returns	X	X	
revert			X
revoke	X	X	
right	X	X	
rlike		X	

role		X	
roles		X	
rollback	X	X	
rollup	X	X	
row	X	X	
row_number	X		
rowcount			X
rows	X	X	
rule			X
running	X	X	
save			X
savepoint	X	X	
schema		X	
schemas		X	
scope	X	X	
scroll	X	X	
search	X	X	
second	X		
securityaudit			X
seek	X	X	
select	X	X	
semi		X	
sensitive	X	X	
sequencefile		X	
serdeproperties		X	
serialize_fn		X	
session_user	X		X
set	X	X	
setuser			X
show	X	X	
shutdown			X
similar	X	X	
sin	X		
sinh	X		
skip	X	X	
smallint	X	X	
some	X	X	
sort		X	
specific	X	X	
specifictype	X	X	

sql	X		
sqlexception	X	X	
sqlstate	X	X	
sqlwarning	X	X	
sqrt	X		
start	X		
static	X	X	
statistics			X
stats		X	
stddev_pop	X		
stddev_samp	X		
stored		X	
straight_join		X	
string		X	
struct		X	
submultiset	X	X	
subset	X	X	
substring	X		
substring_regex	X	X	
succeeds	X	X	
sum	X		
symbol		X	
symmetric	X	X	
system	X		
system_time	X	X	
system_user	X	X	
table	X	X	
tables		X	
tablesample	X	X	
tan	X		
tanh	X		
tblproperties		X	
terminated		X	
textfile		X	
textsize			X
then	X	X	
time	X		
timestamp	X	X	
timezone_hour	X	X	
timezone_minute	X	X	

tinyint		X	
to	X	X	
top			X
trailing	X	X	
tran			X
transform			
translate	X		
translate_regex	X	X	
translation	X	X	
treat	X	X	
trigger	X	X	
trim	X		
trim_array	X	X	
true	X	X	
truncate	X	X	
try_convert			X
uescape	X	X	
unbounded		X	
uncached		X	
union	X	X	
unique	X	X	
uniquejoin			
unknown	X	X	
unnest	X	X	
unpivot			X
update	X	X	
update_fn		X	
updatetext			X
upper	X		
upsert		X	
use		X	
user	X		X
using	X	X	
utc_tmestamp			
value	X		
value_of	X	X	
values	X	X	
var_pop	X		
var_samp	X		
varbinary	X	X	

varchar	X	X	
varying	X	X	
versioning	X	X	
view		X	
views			
waitfor			X
when	X	X	
whenever	X	X	
where	X	X	
while			X
width_bucket	X	X	
window	X	X	
with	X	X	
within	X	X	
without	X	X	
writetext			X
year	X		

SQL differences between Impala and Hive

Impala's SQL syntax follows the SQL-92 standard, and includes extensions, such as built-in functions. Because Impala and Hive share the same Metastore database and their tables are often used interchangeably, this topic covers differences between Impala and Hive in detail.

The current release of Impala does not support the following SQL features that available in Hive SQL:

- Extensibility mechanisms such as TRANSFORM, custom file formats, or custom SerDes.
- XML functions.
- BINARY data type.
- Certain aggregate functions from Hive SQL: covar_pop, covar_samp, corr, percentile, percentile_approx, histogram_numeric, collect_set. Impala supports the set of aggregate functions and analytic functions.
- Lateral views. In Impala 2.3 and higher, Impala supports queries on complex types (STRUCT, ARRAY, or MAP), using join notation rather than the EXPLODE() keyword.

User-defined functions (UDFs) are supported from Impala 1.2 onward.

- Impala supports high-performance UDFs written in C++, as well as reusing some Java-based Hive UDFs.
- Impala supports scalar UDFs and user-defined aggregate functions (UDAFs). Impala does not currently support user-defined table generating functions (UDTFs).
- Only Impala-supported column types are supported in Java-based UDFs.
- The Hive current_user() function cannot be called from a Java UDF through Impala.

Impala does not currently support these Hive SQL statements:

- ANALYZE TABLE (the Impala equivalent is COMPUTE STATS)
- DESCRIBE COLUMN
- DESCRIBE DATABASE
- EXPORT TABLE
- IMPORT TABLE

- SHOW TABLE EXTENDED
- SHOW TBLPROPERTIES
- SHOW INDEXES
- SHOW COLUMNS
- INSERT OVERWRITE DIRECTORY; use INSERT OVERWRITE *TABLE_NAME* or CREATE TABLE AS SELECT to materialize query results into the HDFS directory associated with an Impala table.

Impala respects the serialization.null.format table property only for TEXT tables and ignores the property for Parquet and other formats. Hive respects the serialization.null.format property for Parquet and other formats and converts matching values to NULL during the scan. See the *Data files for text tables* section for using the table property in Impala.

Semantic differences between Impala and Hive SQL features

This section covers instances where Impala and Hive have similar functionality, sometimes including the same syntax, but there are differences in the runtime semantics of those features.

SQL statements and clauses:

- Impala uses different syntax and names for query hints, [SHUFFLE] and [NOSHUFFLE] rather than MapJoin or StreamJoin.
- Impala does not expose MapReduce specific features of SORT BY, DISTRIBUTE BY, or CLUSTER BY.
- Impala does not require queries to include a FROM clause.

Data types:

- Impala supports a limited set of implicit casts. This can help avoid undesired results from unexpected casting behavior.
  - Impala does not implicitly cast between string and numeric or Boolean types. Always use CAST() for these conversions.
  - Impala does perform implicit casts among the numeric types, when going from a smaller or less precise type to a larger or more precise one. For example, Impala will implicitly convert a SMALLINT to a BIGINT or FLOAT, but to convert from DOUBLE to FLOAT or INT to TINYINT requires a call to CAST() in the query.
  - Impala does perform implicit casts from STRING to TIMESTAMP. Impala has a restricted set of literal formats for the TIMESTAMP data type and the FROM_UNIXTIME() format string.

  See the topics under Impala SQL data types on page 13 for full details on implicit and explicit casting for all types, and Impala type conversion functions on page 336 for details about the CAST() function.
- Impala does not store or interpret TIMESTAMP values using the local timezone, to avoid undesired results from unexpected time zone issues. Timestamps are stored and interpreted relative to UTC. This difference can produce different results for some calls to similarly named date/time functions between Impala and Hive. See TIMESTAMP data type on page 45 for a discussion of how Impala handles time zones, and configuration options you can use to make Impala match the Hive behavior more closely when dealing with Parquet-encoded TIMESTAMP data or when converting between the local time zone and UTC.
- The Impala TIMESTAMP type can represent dates ranging from 1400-01-01 to 9999-12-31. This is different from the Hive date range, which is 0000-01-01 to 9999-12-31.
- Impala does not return column overflows as NULL, so that customers can distinguish between NULL data and overflow conditions similar to how they do so with traditional database systems. Impala returns the largest or smallest value in the range for the type. For example, valid values for a tinyint range from -128 to 127. In Impala, a tinyint with a value of -200 returns -128 rather than NULL. A tinyint with a value of 200 returns 127.

Miscellaneous features:

- Impala does not provide virtual columns.
- Impala does not expose locking.
- Impala does not expose some configuration properties.

Related Information

Adapting SQL code from a variety of database systems to Impala

Using text files in Impala

Porting SQL from other database systems to Impala

Although Impala uses standard SQL for queries, you might need to modify SQL source when bringing applications to Impala, due to variations in data types, built-in functions, vendor language extensions, and Hadoop-specific syntax. Even when SQL is working correctly, you might make further minor modifications for best performance.

Porting DDL and DML statements

When adapting SQL code from a legacy database system to Impala, expect to find a number of differences in the DDL statements that you use to set up the schema. Clauses related to physical layout of files, tablespaces, and indexes have no equivalent in Impala. You might restructure your schema considerably to account for the Impala partitioning scheme and Hadoop file formats.

Expect SQL queries to have a much higher degree of compatibility. With modest rewriting to address vendor extensions and features not yet supported in Impala, you might be able to run identical or almost-identical query text on both systems.

Therefore, consider separating out the DDL into a separate Impala-specific setup script. Focus your reuse and ongoing tuning efforts on the code for SQL queries.

Porting data types from other database systems

- Change any VARCHAR, VARCHAR2, and CHAR columns to STRING. Remove any length constraints from the column declarations; for example, change VARCHAR(32) or CHAR(1) to STRING. Impala is very flexible about the length of string values; it does not impose any length constraints or do any special processing (such as blank-padding) for STRING columns. (In Impala 2.0 and higher, there are data types VARCHAR and CHAR, with length constraints for both types and blank-padding for CHAR. However, for performance reasons, it is still preferable to use STRING columns where practical.)
- For national language character types such as NCHAR, NVARCHAR, or NCLOB, be aware that while Impala can store and query UTF-8 character data, currently some string manipulation operations only work correctly with ASCII data.
- Change any DATETIME or TIME columns to DATE or TIMESTAMP. Remove any precision constraints. Remove any timezone clauses, and make sure your application logic or ETL process accounts for the fact that Impala expects all TIMESTAMP values to be in the Coordinated Universal Time (UTC). See TIMESTAMP data type on page 45 for information about the TIMESTAMP data type, and Impala date and time functions on page 344 for conversion functions for different date and time formats.

  You might also need to adapt date- and time-related literal values and format strings to use the supported Impala date and time formats. If you have date and time literals with different separators or different numbers of YY, MM, and so on placeholders than Impala expects, consider using calls to regexp_replace() to transform those values to the Impala-compatible format. See TIMESTAMP data type on page 45 for information about the allowed formats for date and time literals, and Impala string functions on page 368 for string conversion functions such as regexp_replace().

  Instead of SYSDATE, call the function NOW().

  Instead of adding or subtracting directly from a date value to produce a value *N* days in the past or future, use an INTERVAL expression, for example NOW() + INTERVAL 30 DAYS.
- Although Impala supports INTERVAL expressions for datetime arithmetic, as shown in TIMESTAMP data type on page 45, INTERVAL is not available as a column data type in Impala. For any INTERVAL values stored in tables, convert them to numeric values that you can add or subtract using the functions in Impala date and time

functions on page 344. For example, if you had a table DEADLINES with an INT column TIME_PERIOD, you could construct dates N days in the future like so:

```
SELECT NOW() + INTERVAL time_period DAYS from deadlines;
```

- For YEAR columns, change to the smallest Impala integer type that has sufficient range. See Impala SQL data types on page 13 for details about ranges, casting, and so on for the various numeric data types.
- Change any DECIMAL and NUMBER types. If fixed-point precision is not required, you can use FLOAT or DOUBLE on the Impala side depending on the range of values. For applications that require precise decimal values, such as financial data, you might need to make more extensive changes to table structure and application logic, such as using separate integer columns for dollars and cents, or encoding numbers as string values and writing UDFs to manipulate them.
- FLOAT, DOUBLE, and REAL types are supported in Impala. Remove any precision and scale specifications. (In Impala, REAL is just an alias for DOUBLE; columns declared as REAL are turned into DOUBLE behind the scenes.)
- Most integer types from other systems have equivalents in Impala, perhaps under different names such as BIGINT instead of INT8. For any that are unavailable, for example MEDIUMINT, switch to the smallest Impala integer type that has sufficient range. Remove any precision specifications.
- Remove any UNSIGNED constraints. All Impala numeric types are signed.
- For any types holding bitwise values, use an integer type with enough range to hold all the relevant bits within a positive integer.

  For example, TINYINT has a maximum positive value of 127, not 256, so to manipulate 8-bit bitfields as positive numbers switch to the next largest type SMALLINT. For example, CAST(127*2 AS SMALLINT).

  Impala does not support notation such as b'0101' for bit literals.
- For BLOB values, use STRING to represent CLOB or TEXT types (character based large objects) up to 32 KB in size. Binary large objects such as BLOB, RAW BINARY, and VARBINARY do not currently have an equivalent in Impala.
- For Boolean-like types such as BOOL, use the Impala BOOLEAN type.
- Because Impala currently does not support composite or nested types, any spatial data types in other database systems do not have direct equivalents in Impala. You could represent spatial values in string format and write UDFs to process them. Where practical, separate spatial types into separate tables so that Impala can still work with the non-spatial data.
- Take out any DEFAULT clauses. Impala can use data files produced from many different sources, such as Pig, Hive, or MapReduce jobs. The fast import mechanisms of LOAD DATA and external tables mean that Impala is flexible about the format of data files, and Impala does not necessarily validate or cleanse data before querying it. When copying data through Impala INSERT statements, you can use conditional functions such as CASE or NVL to substitute some other value for NULL fields.
- Take out any constraints from your CREATE TABLE and ALTER TABLE statements, for example PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, UNSIGNED, or CHECK constraints. Impala can use data files produced from many different sources, such as Pig, Hive, or MapReduce jobs. Therefore, Impala expects initial data validation to happen earlier during the ETL or ELT cycle. After data is loaded into Impala tables, you can perform queries to test for NULL values. When copying data through Impala INSERT statements, you can use conditional functions such as CASE or NVL to substitute some other value for NULL fields.

  Do as much verification as practical before loading data into Impala. After data is loaded into Impala, you can do further verification using SQL queries to check if values have expected ranges, if values are NULL or not, and so on. If there is a problem with the data, you will need to re-run earlier stages of the ETL process, or do an INSERT ... SELECT statement in Impala to copy the faulty data to a new table and transform or filter out the bad values.
- Take out any CREATE INDEX, DROP INDEX, and ALTER INDEX statements, and equivalent ALTER TABLE statements. Remove any INDEX, KEY, or PRIMARY KEY clauses from CREATE TABLE and ALTER TABLE statements. Impala is optimized for bulk read operations for data warehouse-style queries, and therefore does not support indexes for its tables.
- Calls to built-in functions with out-of-range or otherwise incorrect arguments, return NULL in Impala as opposed to raising exceptions. (This rule applies even when the ABORT_ON_ERROR=true query option is in effect.) Run

small-scale queries using representative data to doublecheck that calls to built-in functions are returning expected values rather than NULL. For example, unsupported CAST operations do not raise an error in Impala:

- For any other type not supported in Impala, you could represent their values in string format and write UDFs to process them.
- To detect the presence of unsupported or unconvertable data types in data files, do initial testing with the ABORT_ON_ERROR=true query option in effect. This option causes queries to fail immediately if they encounter disallowed type conversions.

SQL statements to remove or adapt

The following SQL statements or clauses are not currently supported or supported with limitations in Impala:

- Impala supports the DELETE statement only for Kudu tables.

  Impala is intended for data warehouse-style operations where you do bulk moves and transforms of large quantities of data. When not using Kudu tables, instead of DELETE, use INSERT OVERWRITE to entirely replace the contents of a table or partition, or use INSERT ... SELECT to copy a subset of data (everything but the rows you intended to delete) from one table to another.
- Impala supports the UPDATE statement only for Kudu tables.

  When not using Kudu tables, instead of UPDATE, do all necessary transformations early in the ETL process, such as in the job that generates the original data, or when copying from one table to another to convert to a particular file format or partitioning scheme.
- Impala has no transactional statements, such as COMMIT or ROLLBACK.

  Impala effectively works like the AUTOCOMMIT mode in some database systems, where changes take effect as soon as they are made.
- If your database, table, column, or other names conflict with Impala reserved words, use different names or quote the names with backticks.

  Conversely, if you use a keyword that Impala does not recognize, it might be interpreted as a table or column alias.

  For example, in SELECT * FROM t1 NATURAL JOIN t2, Impala does not recognize the NATURAL keyword and interprets it as an alias for the table t1. If you experience any unexpected behavior with queries, check the list of reserved words to make sure all keywords in join and WHERE clauses are supported keywords in Impala.
- Impala has some restrictions on subquery support.
- Impala supports UNION and UNION ALL set operators, but not INTERSECT.

  Prefer UNION ALL over UNION when you know the data sets are disjoint or duplicate values are not a problem; UNION ALL is more efficient because it avoids materializing and sorting the entire result set to eliminate duplicate values.
- Impala requires query aliases for the subqueries used as inline views in the FROM clause.

  For example, without the alias contents_of_t1 at the end, the following query gives a syntax error:

  ```
  SELECT COUNT(*) FROM (SELECT * FROM t1) contents_of_t1;
  ```

  Aliases are not required for the subqueries used in other parts of queries. For example:

  ```
  SELECT * FROM functional.alltypes WHERE id = (SELECT MIN(id) FROM functi
  onal.alltypes);
  ```

- When an alias is declared for an expression in a query, that alias cannot be referenced again within the same SELECT list.

  For example, the average alias cannot be referenced twice in the SELECT list as below. You will receive an error:

  ```
  SELECT AVG(x) AS average, average+1 FROM t1 GROUP BY x;
  ```

  An alias can be referenced again in the same query if not in the SELECT list. For example, the average alias can be referenced twice as shown below:

  ```
  SELECT AVG(x) AS average FROM t1 GROUP BY x HAVING average > 3;
  ```

- Impala does not support NATURAL JOIN, and it does not support the USING clause in joins.
- Impala supports a limited choice of partitioning types.

  Partitions are defined based on each distinct combination of values for one or more partition key columns. Impala does not redistribute or check data to create evenly distributed partitions. You must choose partition key columns based on your knowledge of the data volume and distribution. Adapt any tables that use range, list, hash, or key partitioning to use the Impala partition syntax for CREATE TABLE and ALTER TABLE statements.

  Impala partitioning is similar to range partitioning where every range has exactly one value, or key partitioning where the hash function produces a separate bucket for every combination of key values.

  Note: Because the number of separate partitions is potentially higher than in other database systems, keep a close eye on the number of partitions and the volume of data in each one; scale back the number of partition key columns if you end up with too many partitions with a small volume of data in each one.

  To distribute work for a query across a cluster, you need at least one HDFS block per node. HDFS blocks are typically multiple megabytes, especially for Parquet files. Therefore, if each partition holds only a few megabytes of data, you are unlikely to see much parallelism in the query because such a small amount of data is typically processed by a single node.
- For the “top-N” queries, Impala uses the LIMIT clause rather than comparing against a pseudo column named ROWNUM or ROW_NUM.

SQL constructs to double-check

Some SQL constructs that are supported have behavior or defaults more oriented towards convenience than optimal performance. Also, sometimes machine-generated SQL, perhaps issued through JDBC or ODBC applications, might have inefficiencies or exceed internal Impala limits. As you port SQL code, examine and possibly update the following where appropriate:

- A CREATE TABLE statement with no STORED AS clause creates data files in plain text format, which is convenient for data interchange but not a good choice for high-volume data with high-performance queries.
- Adapting tables that were already partitioned in a different database system could produce an Impala table with a high number of partitions and not enough data in each one, leading to underutilization of Impala's parallel query features.
- The INSERT ... VALUES syntax is suitable for setting up toy tables with a few rows for functional testing when used with HDFS. Each such statement creates a separate tiny file in HDFS, and it is not a scalable technique for loading megabytes or gigabytes (let alone petabytes) of data.

  Consider revising your data load process to produce raw data files outside of Impala, then setting up Impala external tables or using the LOAD DATA statement to use those data files instantly in Impala tables, with no conversion or indexing stage.

  INSERT works fine for Kudu tables even though not particularly fast.
- If your ETL process is not optimized for Hadoop, you might end up with highly fragmented small data files, or a single giant data file that cannot take advantage of distributed parallel queries or partitioning. In this case, use an

INSERT ... SELECT statement to copy the data into a new table and reorganize into a more efficient layout in the same operation.

You can do INSERT ... SELECT into a table with a more efficient file format or from an unpartitioned table into a partitioned one.

- Complex queries may have high codegen time. As a workaround, set the query option DISABLE_CODEGEN= true if queries fail for this reason.
- If practical, rewrite UNION queries to use the UNION ALL operator instead. Prefer UNION ALL over UNION when you know the data sets are disjoint or duplicate values are not a problem; UNION ALL is more efficient because it avoids materializing and sorting the entire result set to eliminate duplicate values.

Next porting steps after verifying syntax and semantics

Some of the decisions you make during the porting process can have an impact on performance. After your SQL code is ported and working correctly, examine the performance-related aspects of your schema design, physical layout, and queries to make sure that the ported application is taking full advantage of Impala's parallelism, performance-related SQL features, and integration with Hadoop components. The following are a few of the areas you should examine:

- For the optimal performance, we recommend that you run COMPUTE STATS on all tables.
- Use the most efficient file format for your data volumes, table structure, and query characteristics.
- Partition on columns that are often used for filtering in WHERE clauses.
- Your ETL process should produce a relatively small number of multi-megabyte data files rather than a huge number of small files.

Related Information

Impala conditional functions

ABORT_ON_ERROR query option

How to use specific file formats for compact data and high-performance queries

Most heavily optimized file format for large-scale data warehouse queries

UTF-8 Support

Impala has traditionally offered a single-byte binary character set for STRING data type with the character data encoded in an ASCII character set. This release provides a UTF-8 aware behavior for Impala STRING type to get consistent behavior with Hive on UTF-8 strings using a query option.

Turning ON the UTF-8 behavior

Before this release, Impala was incompatible with Hive in some functions applying on non-ASCII strings. For example, the length() function in Impala returns the length of bytes of the string, while the length() function in Hive returns the length of UTF-8 characters of the string. UTF-8 characters (code points) assemble in variant-length bytes (1~4 bytes), so the results differ for non-ASCII characters in the string.

UTF-8 support enables you to read and write UTF-8 from standard formats like Parquet and ORC, thus improving interoperability with other engines supporting those standard formats.

You can use the new query option, UTF8_MODE, to enable and disable the UTF-8 aware behavior. You can set it globally or at a session level. Only queries with UTF8_MODE=true will have UTF-8-related behaviors.

Note: If you enable the UTF8_MODE query option globally, you must explicitly set UTF8_MODE=false for the existing queries that depend on the past binary behavior.

List of STRING functions

The new query option enables the UTF-8 aware behavior of the following string functions:

- LENGTH(STRING a)
  - returns the number of UTF-8 characters instead of bytes
- SUBSTR(STRING a, INT start [, INT len])
- SUBSTRING(STRING a, INT start [, INT len])()
  - the substring start position and length are counted as UTF-8 characters instead of bytes
- REVERSE(STRING a)
  - the unit of the operation is a UTF-8 character. That is, it does not reverse bytes inside a UTF-8 character.

    Note: The results of reverse("###SQL##") used to be "####LQS#####" and now "##LQS###".
- INSTR(STRING str, STRING substr[, BIGINT position[, BIGINT occurrence]])
- LOCATE(STRING substr, STRING str[, INT pos])
  - These functions have an optional position argument. The return values are also positions in the string. In UTF-8 mode, these positions are counted as UTF-8 characters instead of bytes.
- mask functions
  - The unit of the operation is a UTF-8 character. That is, they do not mask the string byte-to-byte.
- upper/lower/initcap
  - These functions recognize non-ascii characters and transform them based on the current locale used by the Impala process.

Limitations

- Use the UTF8_MODE option only when needed because the performance of UTF-8 is not optimized yet. It is an experimental feature.
- UTF-8 support for CHAR and VARCHAR types is not implemented yet. So VARCHAR(N) returns N bytes instead of N UTF-8 characters.

1.0.0

Starting and Stopping Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-05

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring Client Access to Impala....4
Impala Shell Tool....4
Impala Shell Configuration Options....4
Impala Shell Configuration File....7
Running Commands and SQL Statements in Impala Shell....8
Impala Shell Command Reference....9
Setting up ODBC connection from a BI tool....10
Configuring Impyla for Impala....12
Connecting to Impala Virtual Warehouse....14
Configuring Delegation for Clients....14
Spooling Impala Query Results....15
Setting Timeouts in Impala....16
Setting Timeout and Retries for Thrift Connections to Backend Client....16
Increasing StateStore Timeout....17
Setting the Idle Query and Idle Session Timeouts....18
Adjusting Heartbeat TCP Timeout Interval....18

Configuring Client Access to Impala

Application developers have a number of options to interface with Impala.

The core development language with Impala is SQL. You can also use Java or other languages to interact with Impala through the standard JDBC and ODBC interfaces used by many business intelligence tools. For specialized kinds of analysis, you can supplement the Impala built-in functions by writing user-defined functions in C++ or Java.

You can connect and submit requests to the Impala through:

- The impala-shell interactive command interpreter
- The Hue web-based user interface
- JDBC
- ODBC
- Impyla

Impala clients can connect to any Coordinator Impala Daemon (impalad) via HiveServer2 over HTTP or over the TCP binary. You can control client connections to Impala using the Impala coordinator role available under the Configurations tab for a selected Virtual Warehouse.

Impala Shell Tool

You can use the Impala shell tool (impala-shell) to set up databases and tables, insert data, and issue queries.

For ad-hoc queries and exploration, you can submit SQL statements in an interactive session. To automate your work, you can specify command-line options to process a single statement or a script file. The `impala-shell` accepts all the same SQL statements, plus some shell-only commands that you can use for tuning performance and diagnosing problems.

You can install the Impala shell on a local computer and use it as a client to connect with an Impala Virtual Warehouse instance. If you are connecting from a node that is already a part of a CDH or CDP cluster, you already have Impala shell and do not need to install it. For more information on installing Impala shell, see the link provided under Related Information.

Impala Shell Configuration Options

You can specify the following options when starting impala-shell to control how shell commands are executed. You can specify options on the command line or in the impala-shell configuration file.

Command-Line Option	Configuration File Setting	Explanation
-B or --delimited	write_delimited=true	Causes all query results to be printed in plain format as a delimited text file. Useful for producing data files to be used with other Hadoop components. Also useful for avoiding the performance overhead of pretty-printing all output, especially when running benchmark tests using queries returning large result sets. Specify the delimiter character with the --output_delimiter option. Store all query results in a file rather than printing to the screen with the -B option.
-E or --vertical	vertical=true	Causes all query results to print in vertical format. In this mode, impala-shell will print each row in the following style. • The first line will contain the line number followed by the row's columns in separate lines. Each column line will have a prefix with its name and a colon. To enable this mode, use the shell option '-E' or '--vertical', or 'set VERTICAL= true' in the interactive mode. To disable it in interactive mode, 'set VERTICAL=false'. NOTE: This vertical option will be disabled if the '-B' option or 'set WRITE_DELIMITED=true' is specified.

Command-Line Option	Configuration File Setting	Explanation
--connect_max_tries	--connect_max_tries=4	Sets the maximum number of attempts to connect to a coordinator. Currently, the maximum number of attempts to connect to a coordinator is hard coded to 4 in hs2-HTTP mode. From this release, you can configure the maximum number of attempts impala-shell can make using this option --connect_max_tries. The default value of this option is 4. After the number of attempts specified through this option, impala-shell returns with error "Not connected to Impala"
--hs2_fp_format	hs2_fp_format=<a Python-based format specification expression which will get parsed and applied to floating-pointcolumn values>	Sets the printing format specification for floating point values when using HS2 protocol. The default behaviour makes the values handled by Python's str() built-in method. Use '16G' to match Beeswax protocol's floating-point output format.
--http_cookie_names	http_cookie_names=KNOX_BACKEND,IMPALA	This parameter has to be set to include the cookie that Knox uses for stateful connections. This is what the Knox enableStickySession=true uses. When using impala-shell and http mode, you must update the client connection string to include the cookie being used for session persistence using the `-http_cookie_names`. This config is needed for Active-Active HA in Impala.
--live_progress	live_progress=true	Prints a progress bar showing roughly the percentage complete for each query. Information is updated interactively as the query progresses.
--disable_live_progress	live_progress=false	Disables live_progress in the interactive mode.
--print_header	print_header=true	
-o *FILENAME* or --output_file *FILENAME*	output_file=*FILENAME*	Stores all query results in the specified file. Typically used to store the results of a single query issued from the command line with the -q option. Also works for interactive sessions; you see the messages such as number of rows fetched, but not the actual result set. To suppress these incidental messages when combining the -q and -o options, redirect stderr to /dev/null.
--output_delimiter= *CHARACTER*	output_delimiter=*CHARACTER*	Specifies the character to use as a delimiter between fields when query results are printed in plain format by the -B option. Defaults to tab ('\t'). If an output value contains the delimiter character, that field is quoted, escaped by doubling quotation marks, or both.
-p or --show_profiles	show_profiles=true	Displays the query execution plan (same output as the EXPLAIN statement) and a more detailed low-level breakdown of execution steps, for every query executed by the shell.
--profile_format= text | json | prettyjson	N/A	json and prettyjson output the JSON representation of each profile in a dense single-line form and in a human-readable multi-line form respectively.
-h or --help	N/A	Displays help information.
N/A	history_max=1000	Sets the maximum number of queries to store in the history file.
-i *HOSTNAME* or --impalad=*HOSTNAME*[:*PORTNUM*]	impalad=*HOSTNAME*[:*PORTNUM*]	Connects to the impalad daemon on the specified host. The default port of 21050 is assumed unless you provide another value. You can connect to any host in your cluster that is running impalad. If you connect to an instance of impalad that was started with an alternate port specified by the --fe_port flag, provide that alternative port.
-q *QUERY* or --query=*QUERY*	query=*QUERY*	Passes a query or other `impala-shell` command from the command line. The `impala-shell` interpreter immediately exits after processing the statement. It is limited to a single statement, which could be a SELECT, CREATE TABLE, SHOW TABLES, or any other statement recognized in impala-shell. Because you cannot pass a USE statement and another query, fully qualify the names for any tables outside the default database. (Or use the -f option to pass a file with a USE statement followed by other queries.)

Command-Line Option	Configuration File Setting	Explanation
-f *QUERY_FILE* or --query_file=*QUERY_FILE*	query_file=*PATH_TO_QUERY_FILE*	Passes a SQL query from a file. Multiple statements must be semicolon (;) delimited.
--query_option= "*OPTION=VALUE* -Q "*OPTION=VALUE*"	Header line [impala.query_options], followed on subsequent lines by *OPTION=VALUE*, one option per line.	Sets default query options for an invocation of the `impala-shell` command. To set multiple query options at once, use more than one instance of this command-line option. The query option names are not case-sensitive.
-V or --verbose	verbose=true	Enables verbose output.
--quiet	verbose=false	Disables verbose output.
-v or --version	version=true	Displays version information.
-c	ignore_query_failure=true	Continues on query failure.
-d *DEFAULT_DB* or --database=*DEFAULT_DB*	default_db=*DEFAULT_DB*	Specifies the database to be used on startup. Same as running the USE statement after connecting. If not specified, a database named DEFAULT is used.
--ssl	ssl=true	Enables TLS/SSL for `impala-shell`.
--http_socket_timeout_s	http_socket_timeout_s=*HTTP_SOCKET_TIMEOUT_S*	Sets the timeout in seconds after which the socket will time out if the associated operation cannot be completed. Set to None to disable any timeout. This configurable option is only supported for hs2-http mode and the DEFAULT is NONE.
PATH_TO_CERTIFICATE	ca_cert=*PATH_TO_CERTIFICATE*	The local pathname pointing to the third-party CA certificate, or to a copy of the server certificate for self-signed server certificates. If --ca_cert is not set, `impala-shell` enables TLS/SSL, but does not validate the server certificate. This is useful for connecting to a known-good Impala that is only running over TLS/SSL, when a copy of the certificate is not available (such as when debugging customer installations).
-l	use_ldap=true	Enables LDAP authentication.
-u	user=*USER_NAME*	Supplies the username, when LDAP authentication is enabled by the -l option. (Specify the short username, not the full LDAP distinguished name.) The shell then prompts interactively for the password.
--ldap_password_cmd= *COMMAND*	N/A	Specifies a command to run to retrieve the LDAP password, when LDAP authentication is enabled by the -l option. If the command includes space-separated arguments, enclose the command and its arguments in quotation marks.
-j, --jwt	N/A	Indicates that JWT authentication will be used when this command line option is specified in the impala-shell command.
--jwt_cmd	N/A	Shell command to run to retrieve the JWT to be used for authentication.
--config_file= *PATH_TO_CONFIG_FILE*	N/A	Specifies the path of the file containing `impala-shell` configuration settings. The default is /etc/impalarc. This setting can only be specified on the command line.
--live_progress	N/A	Prints a progress bar showing roughly the percentage complete for each query. The information is updated interactively as the query progresses.
--live_summary	N/A	Prints a detailed report, similar to the SUMMARY command, showing progress details for each phase of query execution. The information is updated interactively as the query progresses.
--var= *VARIABLE_NAME=* *VALUE*	N/A	Defines a substitution variable that can be used within the `impala-shell` session. The variable can be substituted into statements processed by the -q or -f options, or in an interactive shell session. Within a SQL statement, you substitute the value by using the notation ${var:*VARIABLE_NAME*}.

Command-Line Option	Configuration File Setting	Explanation
--auth_creds_ok_in_clear	N/A	Allows LDAP authentication to be used with an insecure connection to the shell. WARNING: This will allow authentication credentials to be sent unencrypted, and hence may be vulnerable to an attack.
--protocol= *PROTOCOL*	N/A	Protocol to use for the connection to Impala. Valid *PROTOCOL* values are: • 'hs2-http': Impala-shell uses HTTP transport to speak to the Impala Daemon via the HiveServer2 protocol.

Impala Shell Configuration File

You can store a set of default settings for `impala-shell` in the `impala-shell` configuration file.

The global `impala-shell` configuration file is located in /etc/impalarc.

The user-level `impala-shell` configuration file is located in ~/.impalarc.

Note that the global-level file name is different from the user-level file name. The global-level file name does not include a dot (.) in the file name.

The default path of the global configuration file can be changed by setting the $IMPALA_SHELL_GLOBAL_CONFIG_FILE environment variable.

To specify a different file name or path for the user-level configuration file, start `impala-shell` with the --config_file `impala-shell` option set to the path of the configuration file.

Typically, an administrator creates the global configuration file for the `impala-shell`, and if the user-level configuration file exists, the options set in the user configuration file take precedence over those in the global configuration file.

In turn, any options you specify on the `impala-shell` command line override any corresponding options within the configuration file.

The `impala-shell` configuration file (global or user) must contain a header label [impala], followed by the options specific to `impala-shell`.

The `impala-shell` configuration file consists of key-value pairs, one option per line. Everything after the # character on a line is treated as a comment and ignored.

The names of the options in the configuration file are similar (although not necessarily identical) to the long-form command-line arguments to the `impala-shell` command. For the supported options in the configuration file, see Impala Shell Configuration Options on page 4.

You can specify key-value pair options using keyval, similar to the --var command-line option. For example, keyval=*VARIABLE1=VALUE1*.

The query options specified in the [impala] section override the options specified in the [impala.query_options] section.

The following example shows a configuration file that you might use during benchmarking tests. It sets verbose mode, so that the output from each SQL query is followed by timing information. `impala-shell` starts inside the database containing the tables with the benchmark data, avoiding the need to issue a USE statement or use fully qualified table names.

In this example, the query output is formatted as delimited text rather than enclosed in ASCII art boxes, and is stored in a file rather than printed to the screen. Those options are appropriate for benchmark situations, so that the overhead of `impala-shell` formatting and printing the result set does not factor into the timing measurements. It also enables the show_profiles option. That option prints detailed performance information after each query, which might be valuable in understanding the performance of benchmark queries.

```
[impala]
verbose=true
default_db=tpc_benchmarking
```

```
write_delimited=true
output_delimiter=,
output_file=/home/tester1/benchmark_results.csv
show_profiles=true
keyval=msg1=hello,keyval=msg2=world
```

The following example shows a configuration file that connects to a specific remote Impala node, runs a single query within a particular database, then exits. Any query options predefined under the [impala.query_options] section in the configuration file take effect during the session.

You would typically use this kind of single-purpose configuration setting with the `impala-shell` command-line option --config_file=*PATH_TO_CONFIG_FILE*, to easily select between many predefined queries that could be run against different databases, hosts, or even different clusters. To run a sequence of statements instead of a single query, specify the configuration option query_file=*PATH_TO_QUERY_FILE* instead.

```
[impala]
impalad=impala-test-node1.example.com
default_db=site_stats
# Issue a predefined query and immediately exit.
query=select count(*) from web_traffic where event_date = trunc(now(),'dd')

[impala.query_options]
mem_limit=32g
```

Running Commands and SQL Statements in Impala Shell

This topic provides the commonly used syntax and shortcut keys in impala-shell.

The following are a few of the key syntax and usage rules for running commands and SQL statements in impala-shell.

- To see the full set of available commands, press TAB twice.
- To cycle through and edit previous commands, click the up-arrow and down-arrow keys.
- Use the standard set of keyboard shortcuts in GNU Readline library for editing and cursor movement, such as Ctrl-A for the beginning of line and Ctrl-E for the end of line.
- Commands and SQL statements must be terminated by a semi-colon.
- Commands and SQL statements can span multiple lines.
- Use -- to denote a single-line comment and /* */ to denote a multi-line comment.

  A comment is considered part of the statement it precedes, so when you enter a -- or /* */ comment, you get a continuation prompt until you finish entering a statement ending with a semicolon. For example:

  ```
  [impala] > -- This is a test comment
                   > SHOW TABLES LIKE 't*';
  ```

- If a comment contains the ${*VARIABLE_NAME*} and it is not for a variable substitution, the $ character must be escaped, e.g. -- \${hello}.

Variable Substitution in impala-shell

You can define substitution variables to be used within SQL statements processed by `impala-shell`.

1. You specify the variable and its value as below.
   - On the command line, you specify the option --var=*VARIABLE_NAME*=*VALUE*
   - Within an interactive session or a script file processed by the -f option, use the SET VAR:*VARIABLE_NAME*=*VALUE* command.
2. Use the above variable in SQL statements in the impala-shell session using the notation: ${VAR:*VARIABLE_NAME*}.

For example, here are some `impala-shell` commands that define substitution variables and then use them in SQL statements executed through the -q and -f options. Notice how the -q argument strings are single-quoted to

prevent shell expansion of the ${var:value} notation, and any string literals within the queries are enclosed by double quotation marks.

```
$ impala-shell --var=tname=table1 --var=colname=x --var=coltype=string -q '
CREATE TABLE ${var:tname} (${var:colname} ${var:coltype}) STORED AS PARQUET'
Query: CREATE TABLE table1 (x STRING) STORED AS PARQUET
```

The below example shows a substitution variable passed in by the --var option, and then referenced by statements issued interactively. Then the variable is reset with the SET command.

```
$ impala-shell --quiet --var=tname=table1

[impala] > SELECT COUNT(*) FROM ${var:tname};

[impala] > SET VAR:tname=table2;
[impala] > SELECT COUNT(*) FROM ${var:tname};
```

When you run a query, the live progress bar appears in the output of a query. The bar shows roughly the percentage of completed processing. When the query finishes, the live progress bar disappears from the console output.

Impala Shell Command Reference

Use the following commands within impala-shell to pass requests to the impalad daemon that the shell is connected to. You can enter a command interactively at the prompt or pass it as the argument to the -q option of impala-shell.

Command	Explanation
Impala SQL statements	You can issue valid SQL statements to be executed.
connect	Connects to the specified instance of impalad. The default port of 21050 is assumed unless you provide another value. You can connect to any host in your cluster that is running impalad. If you connect to an instance of impalad that was started with an alternate port specified by the --fe_port flag, you must provide that alternate port.
help	Help provides a list of all available commands and options.
history	Maintains an enumerated cross-session command history. This history is stored in the ~/.impalahistory file.
profile	Displays low-level information about the most recent query. Used for performance diagnosis and tuning. The report starts with the same information as produced by the EXPLAIN statement and the SUMMARY command.
quit	Exits the shell. Remember to include the final semicolon so that the shell recognizes the end of the command.
rerun or @	Executes a previous `impala-shell` command again, from the list of commands displayed by the history command. These could be SQL statements, or commands specific to `impala-shell` such as quit or profile. Specify an integer argument. A positive integer N represents the command labelled N in the output of the HIST ORY command. A negative integer -N represents the Nth command from the end of the list, such as -1 for the most recent command. Commands that are executed again do not produce new entries in the HISTORY output list.
set	Manages query options for an `impala-shell` session. These options are used for query tuning and troubleshooting. Issue SET with no arguments to see the current query options, either based on the `impalad` defaults, as specified by you at `impalad` startup, or based on earlier SET statements in the same session. To modify option values, issue commands with the syntax set *OPTION=VALUE*. To restore an option to its default, use the unset command.
shell	Executes the specified command in the operating system shell without exiting impala-shell. You can use the ! character as shorthand for the shell command. **Note:** Quote any instances of the -- or /* tokens to avoid them being interpreted as the start of a comment. To embed comments within source or ! commands, use the shell comment character # before the comment portion of the line.

Command	Explanation
source or src	Executes one or more statements residing in a specified file from the local filesystem. Allows you to perform the same kinds of batch operations as with the -f option, but interactively within the interpreter. The file can contain SQL statements and other `impala-shell` commands, including additional SOURCE commands to perform a flexible sequence of actions. Each command or statement, except the last one in the file, must end with a semicolon.
summary	Summarizes the work performed in various stages of a query. It provides a higher-level view of the information displayed by the EXPLAIN command. Added in Impala 1.4.0. The time, memory usage, and so on reported by SUMMARY only include the portions of the statement that read data, not when data is written. Therefore, the PROFILE command is better for checking the performance and scalability of INSERT statements. You can see a continuously updated report of the summary information while a query is in progress.
unset	Removes any user-specified value for a query option and returns the option to its default value. You can also use it to remove user-specified substitution variables using the notation UNSET VAR:*VARIABLE_NAME*.
use	Indicates the database against which to run subsequent commands. Lets you avoid using fully qualified names when referring to tables in databases other than default. Not effective with the -q option, because that option only allows a single statement in the argument.
version	Returns Impala version information.

Setting up ODBC connection from a BI tool

Describes how to connect to Impala Virtual Warehouses using ODBC with your BI tool, with Tableau as an example.

Before you begin

Before you can use Tableau with Impala Virtual Warehouses, you must have created a Database Catalog that is populated with data. You have the option to populate your Database Catalog with sample data when you create it. You must also create an Impala Virtual Warehouse, which is configured to connect to the Database Catalog that is populated with data.

Procedure

1. Download the latest version of the Impala ODBC driver from Cloudera Downloads page or alternatively, on the **Virtual Warehouses** page, click the options menu for the warehouse you want to connect to your BI tool, and select Download JDBC/ODBC Driver and install it.
2. Install the driver on the local host where you intend to use Tableau Desktop.
3. Log in to the Cloudera web interface and navigate to the **Data Warehouse** service.
4. Click Virtual Warehouse in the left navigation panel.
5. On the **Virtual Warehouses** page, click for the Impala warehouse you want to connect to with Tableau, and select Copy JDBC URL:

   This copies the JDBC URL to your system clipboard.
6. Paste the copied JDBC URL into a text file. It should look similar to the following:

   ```
   jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<dwx.company.co
   m>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
   ```
7. From the text file where you just pasted the URL, copy the host name from the JDBC URL to your system clipboard. For example, in the URL shown in Step 6, the host name is:

   ```
   <your-virtual-warehouse>.<your-environment>.<dwx.company.com>
   ```

8. Start Tableau and navigate to ConnectMore…Cloudera Hadoop :

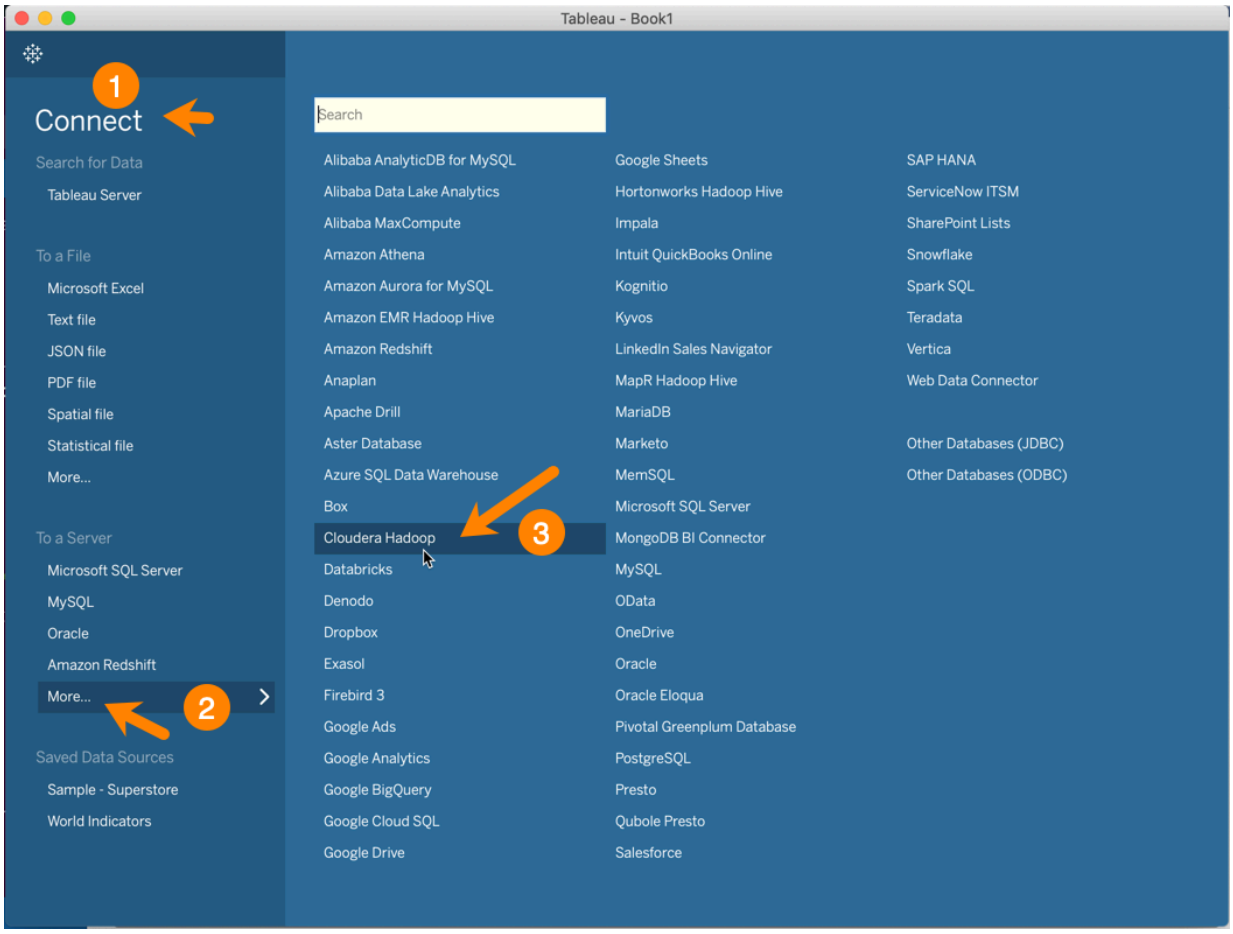

This launches the Cloudera Hadoop dialog box.

9. In the Tableau Cloudera Hadoop dialog box, paste the host name you copied to your clipboard in Step 7 into the Server field:

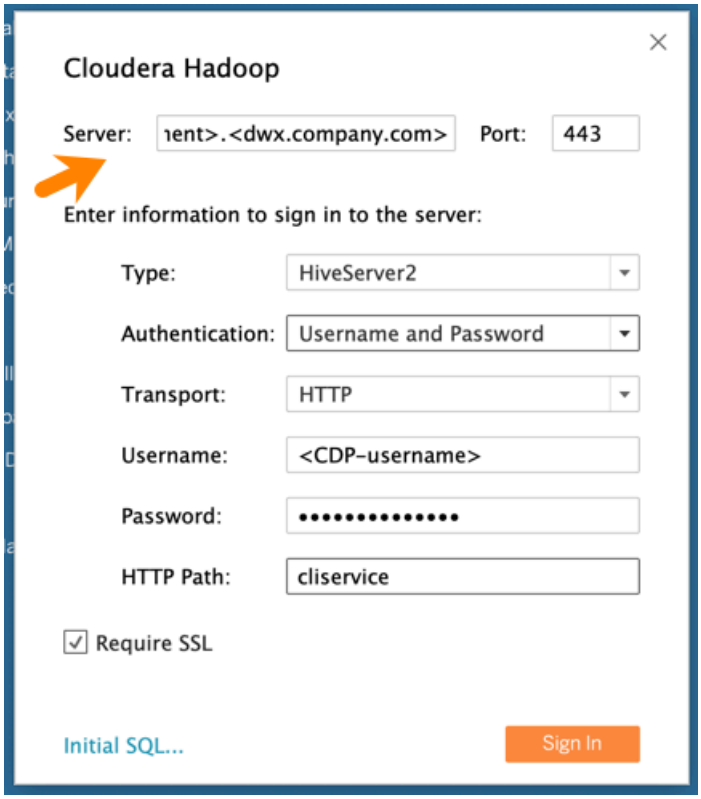

10. Then in the Tableau Cloudera Hadoop dialog box, set the following options:
    - Port: 443
    - Type: HiveServer2
    - Authentication: Username and Password
    - Transport: HTTP
    - Username: Username to connect to the CDP Data Warehouse service.
    - Password: Password to connect to the CDP Data Warehouse service.
    - HTTP Path: cliservice
    - Require SSL: Make sure this is selected.
11. Click Sign In.

Configuring Impyla for Impala

Explains how to install Impyla to connect to and submit SQL queries to Impala. Impyla is a Python client wrapper around the HiveServer2 Thrift Service. It connects to Impala and implements Python DB API 2.0.

About this task

Impyla releases are available at pypi.org. To get the available releases, check Release history.

Note: Cloudera will not support versions of Impyla that are built manually from source code.

Key Features of Impyla

- HiveServer2 compliant.
- Works with Impala including nested data.

- DB API 2.0 (PEP 249)-compliant Python client (similar to sqlite or MySQL clients) supporting Python 2.6+ and Python 3.3+.
- Works with LDAP, SSL.
- SQLAlchemy connector.
- Converts to pandas DataFrame, allowing easy integration into the Python data stack (including scikit-learn and matplotlib); see the Ibis project for a richer experience.
- For more information, see here.

Before you begin

Different systems require different packages to be installed to enable SASL support in Impyla. The following list shows some examples of how to install the packages on different distributions.

You must have the following installed in your environment before installing impyla. Python 2.6+ or 3.3+ and the pip packages six, bitarray, thrift and thriftpy2 will be automatically installed as dependencies when installing impyla. However if you clone the impyla repo and run their local copy, you must install these pip packages manually.

- Install the latest pip and setuptools:

```
python -m pip install --upgrade pip setuptools
```

- RHEL/CentOS:

```
sudo yum install gcc-c++ cyrus-sasl-md5 cyrus-sasl-plain cyrus-sasl-gssapi
 cyrus-sasl-devel
```

- Ubuntu:

```
sudo apt install g++ libsasl2-dev libsasl2-2 libsasl2-modules-gssapi-mit
```

Procedure

1. Using pip you can install the latest release: pip install impyla
2. You also need to pip-install pandas for conversion to DataFrame objects or sqlalchemy for the SQLAlchemy engine.

Example

Sample codes

Impyla implements the Python DB API v2.0 (PEP 249) database interface (refer to it for API details):

```
from impala.dbapi import connect

conn = connect(host = "my.host.com", port = 21050)
cursor = conn.cursor()
cursor.execute("SELECT * FROM mytable LIMIT 100")
print(cursor.description)  # prints the result set's schema
results = cursor.fetchall()
cursor.close()
conn.close()
```

The Cursorobject also exposes the iterator interface, which is buffered (controlled by cursor.arraysize):

```
cursor.execute("SELECT * FROM mytable LIMIT 100")
for row in cursor:
    print(row)
```

Furthermore the Cursor object returns you information about the columns returned in the query. This is useful to export your data as a csv file.

```
import csv

cursor.execute("SELECT * FROM mytable LIMIT 100")
columns = [datum[0] for datum in cursor.description]
targetfile = "/tmp/foo.csv"

with open(targetfile, "w", newline = "") as outcsv:
    writer = csv.writer(
        outcsv,
        delimiter = ",",
        quotechar = '"',
        quoting = csv.QUOTE_ALL,
        lineterminator = "\n")
writer.writerow(columns)
for row in cursor:
    writer.writerow(row)
```

You can also get back a pandas DataFrame object

```
from impala.util import as_pandas

# carry df through scikit-learn, for example
df = as_pandas(cur)
```

Connecting to Impala Virtual Warehouse

Lists an example code to connect to Impala VW with LDAP over http using LDAP as the authentication mechanism.

Example

Sample code

```
from impala.dbapi import connect

conn = connect(
    host = "Impala VW endpoint",  # could be coordinator or impala-proxy
    port = 443,
    auth_mechanism = "LDAP",
    use_ssl = True,
    use_http_transport = True,
    http_path = "cliservice",
    user = "ldap_userId",
    password = "ldap_password")
cursor = conn.cursor()
cursor.execute("SELECT * FROM default.emax_temp")
for row in cursor:
    print(row)
cursor.close()
conn.close()
```

Configuring Delegation for Clients

Impala supports user delegation for client connections.

About this task

When users submit Impala queries through a separate application, such as Hue or a business intelligence tool, typically all requests are treated as coming from the same user. Impala supports “delegation” where users whose names you specify can delegate the execution of a query to another user. The query runs with the privileges of the delegated user, not the original authenticated user.

The name of the delegated user is passed using the HiveServer2 protocol configuration property impala.doas.user when the client connects to Impala.

When the client connects over HTTP, the doAs parameter can be specified in the HTTP path. For example:

```
/?doAs=DELEGATED_USER
```

Currently, the delegation feature is available only for Impala queries submitted through application interfaces such as Hue and BI tools. For example, Impala cannot issue queries using the privileges of the HDFS user.

Attention:

- When the delegation is enabled in Impala, the Impala clients should take an extra caution to prevent unauthorized access for the delegate-able users.
- Impala requires Apache Ranger on the cluster to enable delegation. Without Ranger installed, the delegation feature will fail with an error.

Procedure

To enable delegation:

1. Log in to the CDP web interface and navigate to the Data Warehouse service.
2. In the Data Warehouse service, click Virtual Warehouses in the left navigation panel.
3. Select the Impala Virtual Warehouse, click options for the warehouse you want to configure Proxy User.
4. Select Edit.
5. In the Configuration tab, click Impala Coordinator.
6. In the Proxy User Configuration field, type the a semicolon-separated list of key=value pairs of authorized proxy users to the user(s) they can impersonate.

   The list of delegated users are delimited with a comma, e.g. hue=user1, user2.
7. Click Apply to save the changes.

Spooling Impala Query Results

In Impala, you can control how query results are materialized and returned to clients, e.g. impala-shell, Hue, JDBC apps.

Result spooling is turned ON by default and can be controlled using the SPOOL_QUERY_RESULTS query option.

- Since the query result spooling is enabled by default, result sets of queries are eagerly fetched and spooled in the spooling location, either in memory or on disk.

  Once all result rows have been fetched and stored in the spooling location, the resources are freed up. Incoming client fetches can get the data from the spooled results.

Admission Control and Result Spooling

Query results spooling collects and stores query results in memory that is controlled by admission control. Use the following query options to calibrate how much memory to use and when to spill to disk.
MAX_RESULT_SPOOLING_MEM

The maximum amount of memory used when spooling query results. If this value is exceeded when spooling results, all memory will most likely be spilled to disk. Set to 100 MB by default.

MAX_SPILLED_RESULT_SPOOLING_MEM

The maximum amount of memory that can be spilled to disk when spooling query results. Must be greater than or equal to MAX_RESULT_SPOOLING_MEM. If this value is exceeded, the coordinator fragment will block until the client has consumed enough rows to free up more memory. Set to 1 GB by default.

Fetch Timeout

Resources for a query are released when the query completes its execution. To prevent clients from indefinitely waiting for query results, use the FETCH_ROWS_TIMEOUT_MS query option to set the timeout when clients fetch rows. Timeout applies both when query result spooling is enabled and disabled:

- When result spooling is disabled (SPOOL_QUERY_RESULTS = FALSE), the timeout controls how long a client waits for a single row batch to be produced by the coordinator.
- When result spooling is enabled ((SPOOL_QUERY_RESULTS = TRUE), a client can fetch multiple row batches at a time, so this timeout controls the total time a client waits for row batches to be produced.

Explain Plans

Below is the part of the EXPLAIN plan output for result spooling.

```
F01:PLAN FRAGMENT [UNPARTITIONED] hosts=1 instances=1
     |  Per-Host Resources: mem-estimate=4.02MB mem-reservation=4.00MB thr
ead-reservation=1
     PLAN-ROOT SINK
     |  mem-estimate=4.00MB mem-reservation=4.00MB spill-buffer=2.00MB thr
ead-reservation=0
```

- The mem-estimate for the PLAN-ROOT SINK is an estimate of the amount of memory needed to spool all the rows returned by the query.
- The mem-reservation is the number and size of the buffers necessary to spool the query results. By default, the read and write buffers are 2 MB in size each, which is why the default is 4 MB.

PlanRootSink

In Impala, the PlanRootSink class controls the passing of batches of rows to the clients and acts as a queue of rows to be sent to clients.

- When result spooling is disabled, a single batch or rows is sent to the PlanRootSink, and then the client must consume that batch before another one can be sent.
- When result spooling is enabled, multiple batches of rows can be sent to the PlanRootSink, and multiple batches can be consumed by the client.

Setting Timeouts in Impala

Depending on how busy your cluster is, you might increase or decrease various timeout values. Increase timeouts if Impala is cancelling operations prematurely, when the system is responding slower than usual but the operations are still successful if given extra time. Decrease timeouts if operations are idle or hanging for long periods, and the idle or hung operations are consuming resources and reducing concurrency.

Setting Timeout and Retries for Thrift Connections to Backend Client

Impala connections to the backend client are subject to failure in cases when the network is momentarily overloaded.

About this task

To avoid failed queries due to transient network problems, you can configure the number of Thrift connection retries using the following option:

Procedure

1. Log in to the CDP web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, click Virtual Warehouses in the left navigation panel.
3. Select the Impala Virtual Warehouse, click options for the warehouse you want to set the timeout and retry options.
4. Click Edit and navigate to Impala Coordinatorunder Configurationstab.
5. Using the + sign, specify the following if the options are not already added.

   To avoid failed queries due to transient network problems, you can configure the number of Thrift connection retries using the following option:

   - The --backend_client_connection_num_retries option specifies the number of times Impala will try connecting to the backend client after the first connection attempt fails. By default, `impalad` will attempt three re-connections before it returns a failure.

   You can configure timeouts for sending and receiving data from the backend client. Therefore, if for some reason a query does not respond, instead of waiting indefinitely for a response, Impala will terminate the connection after a configurable timeout.

   - The --backend_client_rpc_timeout_ms option can be used to specify the number of milliseconds Impala should wait for a response from the backend client before it terminates the connection and signals a failure. The default value for this property is 300000 milliseconds, or 5 minutes.
6. Click Apply and restart Impala.

Increasing StateStore Timeout

If you have an extensive Impala schema, for example, with hundreds of databases, tens of thousands of tables, you might encounter timeout errors during startup as the Impala catalog service broadcasts metadata to all the Impala nodes using the StateStore service. To avoid such timeout errors on startup, increase the StateStore timeout value from its default of 10 seconds.

About this task

Increase the timeout value of the StateStore service if you see messages in the impalad log such as:

```
Connection with state-store lost
Trying to re-register with state-store
```

Procedure

1. Log in to the CDP web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, click Virtual Warehouses in the left navigation panel.
3. Select the Impala Virtual Warehouse, click options for the warehouse you want to set the timeout and retry options.
4. Click Edit and navigate to Impala Coordinatorunder Configurationstab.
5. Using the + sign, specify the following if the option is not already added.

   Specify a new timeout value larger than the current value using the option StateStoreSubscriber Timeout.

6. Click Apply and restart Impala.

Setting the Idle Query and Idle Session Timeouts

To keep long-running queries or idle sessions from tying up cluster resources, you can set timeout intervals for both individual queries, and entire sessions.

About this task

Procedure

1. Log in to the CDP web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, click Virtual Warehouses in the left navigation panel.
3. Select the Impala Virtual Warehouse, click options for the warehouse you want to set the timeout and retry options.
4. Click Edit and navigate to Impala Coordinatorunder Configurationstab.
5. In the search field, type idle.
6. In the Idle Query Timeout field, specify the time in seconds after which an idle query is cancelled.

   This could be a query whose results were all fetched but was never closed, or one whose results were partially fetched and then the client program stopped requesting further results. This condition is most likely to occur in a client program using the JDBC or ODBC interfaces, rather than in the interactive `impala-shell` interpreter. Once a query is cancelled, the client program cannot retrieve any further results from the query.

   You can reduce the idle query timeout by using the QUERY_TIMEOUT_S query option at the query level. Any non-zero value specified in this field serves as an upper limit for the QUERY_TIMEOUT_S query option.

   The value of 0 disables query timeouts.
7. In the Idle Session Timeout field, specify the time in seconds after which an idle session expires.

   A session is idle when no activity is occurring for any of the queries in that session, and the session has not started any new queries. Once a session is expired, you cannot issue any new query requests to it. The session remains open, but the only operation you can perform is to close it.

   The default value of 0 specifies sessions never expire.

   You can override this setting with the IDLE_SESSION_TIMEOUT query option at the session or query level.
8. Click Apply and restart Impala.

Results

Impala checks periodically for idle sessions and queries to cancel. The actual idle time before cancellation might be up to 50% greater than the specified configuration setting. For example, if the timeout setting was 60, the session or query might be cancelled after being idle between 60 and 90 seconds.

Adjusting Heartbeat TCP Timeout Interval

Using the TCP flag, you can prevent the Statestore from waiting indefinitely for a response from the subscribers that fail to respond to the heartbeat RPC within the set period.

About this task

This flag statestore_heartbeat_tcp_timeout_seconds defines the time that may elapse before a heartbeat RPC connection request from a Statestore server to an Impalad or a Catalog server (subscribers) should be considered dead.

You can increase the flag value if you see intermittent heartbeat RPC timeouts listed in the statestore's log. You may find the max value of "statestore.priority-topic-update-durations" metric on the statestore to get an idea of a reasonable value to be used in this flag.

Note: The priority topic updates are only small amounts of data that take little time to process, similar to the heartbeat complexity.

Procedure

1. Log in to the CDP web interface and navigate to the Data Warehouse service.
2. In the Cloudera Data Warehouse service, click Virtual Warehouses in the left navigation panel.
3. Select the Impala Virtual Warehouse, click options for the warehouse you want to adjust the Heartbeat TCP Timeout.
4. Click Edit and navigate to Impala Statestoredunder Configurationstab.
5. Using the + sign, specify the following if the options are not already added.
6. In the Add Customs Configuration field, add the flag statestore_heartbeat_tcp_timeout_seconds with an appropriate value.
7. You can also control the maximum number of consecutive heartbeat messages an impalad can miss before being declared failed by the statestore by adding this flag statestore_max_missed_heartbeats. Typically, you will not have to change this value.
8. Click Apply and restart Impala.

1.0.0

Troubleshooting Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-05

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Troubleshooting Impala **4**
Access Impala Workload Logs **4**
Locations of Impala Log Files in S3 5
Locations of Impala Log Files in Azure 6

Troubleshooting Impala

This topic describes the general troubleshooting procedures to diagnose some of the commonly encountered issues in Impala.

Symptom	Explanation	Recommendation
Impala takes a long time to start.	Impala instances with large numbers of tables, partitions, or data files take longer to start because the metadata for these objects is broadcast to all `impalad` nodes and cached.	Adjust timeout and synchronicity settings.
Query rejected with the default pool-defined memory limit settings.	Some complex queries fail because the minimum memory reservation per host is greater than the memory available to the query for buffer reservations.	Increase VW t-shirt size so that there are more hosts in the executor group and less memory is needed per host.
Joins fail to complete.	There may be insufficient memory. During a join, data from the second, third, and so on sets to be joined is loaded into memory. If Impala chooses an inefficient join order or join mechanism, the query could exceed the total memory available.	Start by gathering statistics with the COMPUTE STATS statement for each table involved in the join. Consider specifying the [SHUFFLE] hint so that data from the joined tables is split up between nodes rather than broadcast to each node. If tuning at the SQL level is not sufficient, add more memory to your system or join smaller data sets.
Queries return incorrect results.	Impala metadata may be outdated after changes are performed in Hive.	After inserting data, adding a partition, or other operation in Hive, refresh the metadata for the table with the REFRESH statement.
Attempts to complete Impala tasks such as executing INSERT SELECT statements fail. The Impala logs include notes that files could not be opened due to permission denied.	This can be the result of permissions issues. For example, you could use the Hive shell as the hive user to create a table. After creating this table, you could attempt to complete some action, such as an INSERT SELECT on the table. Because the table was created using one user and the INSERT SELECT is attempted by another, this action may fail due to permissions issues.	Ensure the Impala user has sufficient permissions to the table that the Hive user created.
Impala fails to start up, with the `impalad` logs referring to errors connecting to the statestored service and attempts to re-register.	A large number of databases, tables, partitions, and so on can require metadata synchronization, particularly on startup, that takes longer than the default timeout for the statestore service.	Configure the statestore timeout value and possibly other settings related to the frequency of statestore updates and metadata loading.

Access Impala Workload Logs

Describes how to locate Impala logs in S3 or Azure to diagnose some of the commonly encountered issues in Impala.

Using Impala Logs

The Impala logs record information about:

- Any errors Impala encountered.
- How Impala is configured.
- Jobs Impala has completed.

However, you can use the logs record information to troubleshoot only if the relevant logs are downloaded and then uploaded to a location where you can access them. To download the logs from S3 or Azure you must first identify the locations.

Locations of Impala Log Files in S3

This topic describes how to identify the Amazon S3 locations of Impala logs for the different Impala components.

About this task

The Cloudera Data Warehouse service collects logs from Impala Virtual Warehouses and uploads them to an Amazon S3 location. This S3 log location is configured under an external warehouse directory so that the logs are preserved even if the Virtual Warehouse they are collected from is destroyed.

Before you begin

To identify the location of the logs in S3, you must have the environment name, impala ID identifiers, and S3 bucket name.

Procedure

Finding the environment name, and impala ID identifiers

1. In the Data Warehouse service, expand the Environment column by clicking ⋮ .
2. From the Overview page, note down the environment name, and impala ID identifiers.

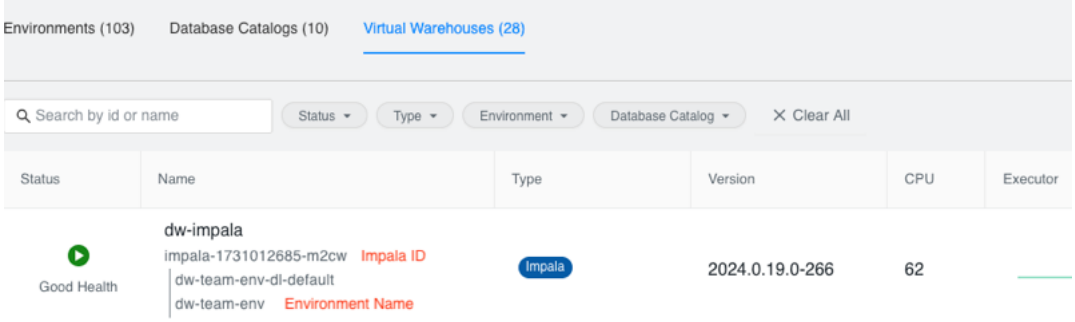

Identifying the bucket name

3. In the Management console navigate to the Environment page.
4. On the Environment page, click on your Environment and then on the Summary tab.
5. On the Logs Storage and Audits section. You can find your log bucket and prefix under the Storage Location. This name is required to identify the S3 location of the logs.

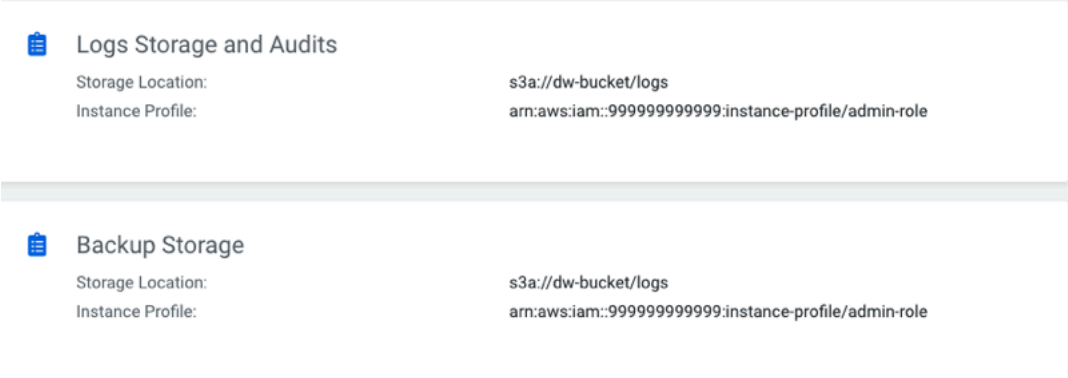

Log locations in S3

6. Now that you have identified the S3 bucket name, environment name, and impala ID identifiers, use the following prefix to find the logs generated by specific components in the following directories. Use the different directories listed here to view Impala/Hue logs.

```
PREFIX =
s3://<s3_bucket_name>/<prefix>/cdw/clusters/<environment_name>/logs/dt=
<date_stamp>/ns=<impala_ID>/
```

Impala component	S3 directory location
impalad	PREFIX + “app=impala-executor-log”
catalogd	PREFIX + “app=catalogd-log”
coordinator	PREFIX + “app=coordinator-log”
auto-scaler	PREFIX + “app=impala-autoscaler-log”
Hue	PREFIX + “app=huebackend-log” PREFIX + “app=hue-huedb-create-job-log” PREFIX + “app=huefrontend-log”
statestored	PREFIX + “app=statestored-log”

The impalad executor logs for 25 August 2024 are located in the following S3 location:

```
PREFIX = s3://<s3_bucket_name>/<prefix>/cdw/clusters/<environment_name>/
logs/dt=2024-08-25/app=impala-executor-log/
```

In the above location, you can find multiple logs that were generated on the specified day.

Impala Minidumps

7. Impala minidumps can be found under the debug-artifacts/impala directory

```
/cdw/clusters/<environment_name>/logs/debug-artifacts/impala/<impala_ID>/
minidump/<pod_name>/
```

Impala Query Profiles

8. Impala query profiles are written in thrift encoded format in this location:

Impala component	S3 directory location
Impala query profiles	PREFIX + “app=impala-profiles”

Use the binary tool to decode thrift to text. This binary tool is provided with the upstream runtime Impala 4.0 as a docker image. Run the following command to use this tool.

```
docker run -i apache/impala:4.0.0-impala_profile_tool < name of the thrift
 encoded file to decode
```

You can use the docker image available here to use this decoding tool.

Locations of Impala Log Files in Azure

This topic describes how to identify the Azure locations of Impala logs for the different Impala components.

About this task

The Cloudera Data Warehouse service collects logs from Impala Virtual Warehouses and uploads them to the Azure storage account that was provided while registering the Environment. This ABFS log location is configured under an

external warehouse directory so that the logs are preserved even if the Virtual Warehouse they are collected from is destroyed.

Before you begin

To identify the location of the logs in Azure, you must have the environment name and impala ID identifiers and to access the logs from the Azure Portal you must know your storage account name.

Procedure

Finding the environment name, impala ID identifiers

1. In the Cloudera Data Warehouse service, go to the Virtual Warehouse tab.
2. From the Overview page, note down the environment name and impala ID identifiers.

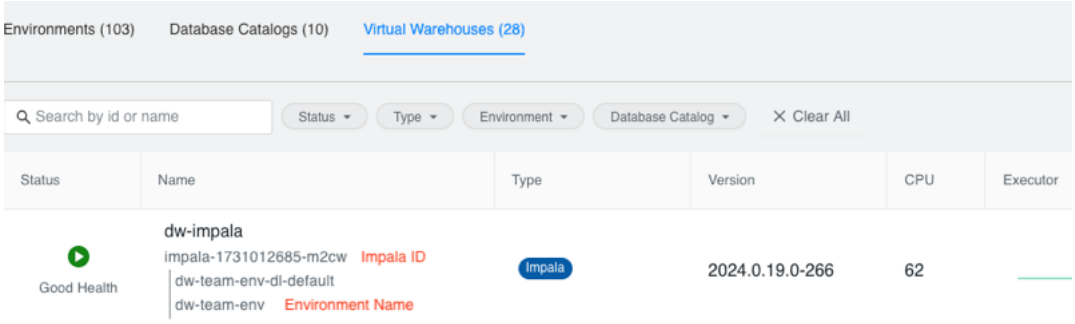

Retrieving your storage account name

3. In the Management Console navigate to the Environments page.
4. On the Environments page, click on your Environment and click on the Summary tab.
5. Scroll down to the Logs Storage and Audits section.

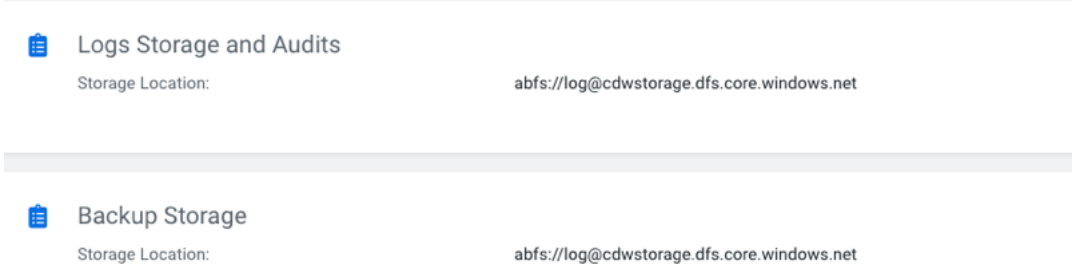

Note down your storage account name.

Accessing the different directories in the Azure Portal

6. Log in to the Azure Portal and search for your storage account name using the Search bar.
7. On the Overview page of your storage account, click on the Containers menu.

8. Click on the file system you used during the Environment registration.

   Note: You need to enable the firewall rules, click on the Firewalls and virtual networks menu, and set Allow access to “All networks”, then save the changes to access the file system.

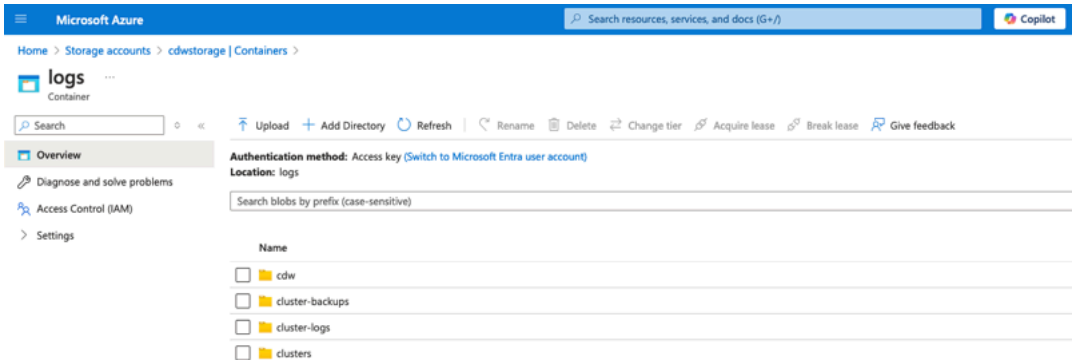

Log locations in ABFS

9. Use the environment name, and impala ID identifiers, in the following prefix to find the logs generated by specific components in the following directories. Use the different directories listed here to view Impala/Hue logs

```
PREFIX =
/cdw/clusters/<environment_name>/logs/dt=<date_stamp>/ns=<impala_ID>/
```

Impala component	ABFS directory location
impalad	PREFIX + “app=impala-executor-log”
catalogd	PREFIX + “app=catalogd-log”
coordinator	PREFIX + “app=coordinator-log”
auto-scaler	PREFIX + “app=impala-autoscaler-log”
Hue	PREFIX + “app=huebackend-log” PREFIX + “app=hue-huedb-create-job-log” PREFIX + “app=huefrontend-log”
statestored	PREFIX + “app=statestored-log”

The impalad executor logs for 25 August 2024 are located in the following ABFS location:

```
PREFIX = /cdw/clusters/<environment_name>/logs/dt=2024-08-25/app=impala-
executor-log/
```

In the above location, you can find multiple logs that were generated on the specified day.

Impala Minidumps

10. Impala minidumps can be found under the debug-artifacts/impala directory

```
/cdw/clusters/<environment_name>/logs/debug-artifacts/impala/<impala_ID>/
minidump/<pod_name>/
```

Impala Query Profiles

11. Impala query profiles are written in thrift encoded format in this location:

Impala component	S3 directory location

Impala query profiles	PREFIX + “app=impala-profiles”

Use the binary tool to decode thrift to text. This binary tool is provided with the upstream runtime Impala 4.0 as a docker image. Run the following command to use this tool.

```
docker run -i apache/impala:4.0.0-impala_profile_tool < name of the thrift
 encoded file to decode
```

You can use the docker image available here to use this decoding tool.

Cloudera Runtime 7.2.11

Integrating Apache Hive with BI tools

Date published: 2019-08-21
Date modified: 2021-09-08

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Connecting Hive to BI tools using a JDBC/ODBC driver in Cloudera Data Warehouse..... 4

Downloading a JDBC driver from Cloudera Data Warehouse.....4

Connecting Hive to BI tools using a JDBC/ODBC driver in Cloudera Data Warehouse

To query, analyze, and visualize data stored in Cloudera, you use drivers provided by Cloudera to connect Apache Hive to Business Intelligence (BI) tools.

About this task

How you connect to Hive depends on a number of factors: the location of Hive inside or outside the cluster, the HiveServer deployment, the type of transport, transport-layer security, and authentication. HiveServer is the server interface that enables remote clients to run queries against Hive and retrieve the results using a JDBC or ODBC connection.

Before you begin

- Choose a Hive authorization model.
- Configure authenticated users for querying Hive through JDBC or ODBC driver. For example, set up a Ranger policy.

Downloading a JDBC driver from Cloudera Data Warehouse

To use third-party BI tools, your client users need a JDBC JAR to connect your BI tool and the service. You learn how to download the JDBC JAR to give to your client, and general instructions about how to use the JDBC JAR.

Before you begin

Before you can use your BI tool with the Cloudera Data Warehouse service:

- You created a Database Catalog.

  You have the option to populate your Database Catalog with sample data when you create it.
- You created a Virtual Warehouse and configured it to connect to the Database Catalog.

Of course, to query tables in the Virtual Warehouse from a client, you must have populated the Virtual Warehouse with some data.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.
2. In the **Overview** page of the Cloudera Data Warehouse service, click See More in the Resources and Downloads tile.
3. Click Hive JDBC Jar to download the Apache Hive JDBC JAR file.
4. Provide the JAR file you downloaded to your JDBC client.

   On most clients, add the JAR file under the Libraries folder. Refer to your client documentation for information on the location to add the JAR file.
5. In the Cloudera Data Warehouse service **Overview** page, for the Virtual Warehouse you want to connect to the client, click ⋮ and select Copy JDBC URL.

   A URL is copied to your system clipboard in the following format:

   ```
   jdbc:hive2://<your_virtual_warehouse>.<your_environment>.<dwx.company.co
   m>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
   ```

6. Paste the URL into a text editor and configure your client BI tool to connect to the Virtual Warehouse using the following portion of the URL, represents the server name of the Virtual Warehouse:

```
<your_virtual_warehouse>.<your_environment>.<dwx.company.com>
```

7. In your JDBC client, set the following other options:

   Authentication: Username and Password

   Username: Username you use to connect to the Cloudera Data Warehouse service.

   Password: Password you use to connect to the Cloudera Data Warehouse service.

1.0.0

Managing Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-05

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ACID Operation **4**
Concepts Used in FULL ACID v2 Tables 4
Key Differences between INSERT-ONLY and FULL ACID Tables 5
Compaction of Data in FULL ACID Transactional Table 6
Using HLL Datasketch Algorithms in Impala **6**
Using KLL Datasketch Algorithms in Impala **10**
Automatic Invalidation/Refresh of Metadata **13**

READ Support for FULL ACID ORC Tables

FULL ACID v2 transactional tables are readable in Impala without modifying any configurations. You must have connection to Hive Metastore server in order to READ from FULL ACID tables.

There are two types of transactional tables available with Hive ACID.

- INSERT-ONLY
- FULL ACID

Until this release, Impala in Cloudera supported INSERT-ONLY transactional tables allowing both READ and WRITE operations. The latest version of Impala in Cloudera now also supports READ of FULL ACID ORC tables.

By default tables created in Impala are INSERT-ONLY managed tables whereas the default tables in Hive are managed tables that are FULL-ACID.

Limitations

- Impala cannot CREATE or WRITE to FULL ACID transactional tables yet. You can CREATE and WRITE FULL ACID transactional tables with transaction scope at the row level via HIVE and use Impala to READ these tables.
- Impala does not support ACID v1.

Concepts Used in FULL ACID v2 Tables

Before beginning to use FULL ACID v2 tables you must be aware of these new concepts like transactions, WriteIds, rowIDs, delta delete directories, locks, etc. that are added to FULL ACID tables to achieve ACID semantics.

Write IDs

For every transaction, both read and write, Hive will assign a globally unique ID. For transactional writes like INSERT and DELETE, it will also assign a table-wise unique ID, a write ID. The write ID range will be encoded in the delta and delete directory names. Results of a DML transactional query are allocated to a location under partition/table. This location is derived by Write ID allocated to the transaction. This provides Isolation of DML queries and such queries can run in parallel without interfering with each other.

New Sub-directories

New data files resulting from a DML query are written to a unique location derived from WriteId of the transaction. You can find the results of an INSERT query in delta directories under partition/table location. Depending on the operation type there can be two types of delta directories:

- Delta Directory: This type is created for the results of INSERT statements and is named delta_<writeId>_<writeId> under partition/table location.
- Delete Delta Directory: This delta directory is created for results of DELETE statements and is named delete_delta_<writeId>_<writeId> under partition/table location.

UPDATE operations create both delete and delta directories.

Row IDs

rowId is the auto-generated unique ID within the transaction and bucket. This is added to each row to identify each row in a table. RowID is used during a DELETE operation. When a record is deleted from a table, the rowId of the deleted row will be written to the delete_delta directory. So for all subsequent READ operations all rows will be read except these rows.

Schematic differences between INSERT-ONLY and FULL ACID tables

INSERT-ONLY tables do not have a special schema. They store the data just like plain original files from the non-ACID world. However, their files are organized differently. For every INSERT statement the created files are put into a transactional directory which has transactional information in its name.

Full ACID tables do have a special schema. They have row identifiers to support row-level DELETEs. So a row in Full ACID format looks like this:

```
{
   "operation": 0,
   "originalTransaction": 1,
   "bucket": 536870912,
   "rowId": 0,
   "currentTransaction": 1,
   "row": {"i": 1}
   }
```

- The green columns are the hidden/system ACID columns.
- Field “row” holds the user data.
- operation 0 means INSERT, 1 UPDATE, and 2 DELETE. UPDATE will not appear because of the split-update technique (INSERT + DELETE).
- originalTransaction is the write ID of the INSERT operation that created this row.
- bucket is a 32-bit integer defined by BucketCodec class.
- rowId is the auto-generated unique ID within the transaction and bucket.
- currentTransaction is the current write ID. For INSERT, it is the same as currentTransaction. For DELETE, it is the write ID when this record is first created.
- row contains the actual data. For DELETE, row will be null.

Key Differences between INSERT-ONLY and FULL ACID Tables

Before beginning to use FULL ACID v2 tables you must be aware of the key differences between the INSERT-ONLY and FULL-ACID tables.

This table highlights some of the differences between the INSERT-ONLY and FULL ACID tables.

	INSERT-ONLY	FULL ACID
Schema	There is no special data schema. They store the data just like plain original files from the non-ACID world.	Data is in special format, i.e. there are synthetic columns with transactional information in addition to actual data.
Transactional information	Transactional information is encoded in directory names.	Full ACID tables also use the same directory structure as INSERT-only tables. Transactional information is encoded in the directory names. Directory name and filename are the source of transactional information.
Table properties	'transactional'='true', 'transactional_properties'='insert_only'	'transactional'='true'
Supported operations	INSERT-ONLY tables only support insertion of data. UPDATES and DELETES are not supported. These tables also provide CREATE TABLE, DROP TABLE, TRUNCATE, INSERT, SELECT operations.	FULL ACID ORC tables can be READ using IMPALA. These tables also provide UPDATE and DELETE operations at the row level using HIVE. This is achieved using transactions like Insert-Only Tables along with changes in ORC Reader to support deletes.
WRITE operation	WRITE operations are atomic and the results of the insert operation are not visible to other query operations until the operation is committed.	WRITE operations are atomic - The operation either succeeds completely or fails; it does not result in partial data.

	INSERT-ONLY	FULL ACID
INSERT operation	For every INSERT statement the created files are added to a transactional directory which has transactional information in its name.	INSERT operation is done through HIVE and this statement is executed in a single transaction. This operation creates a delta directory containing information about this transaction and its data.
DELETE operation	N/A	DELETE operation is done through HIVE and this event creates a special “delete delta” directory.
UPDATE operation	N/A	UPDATE operation is done through HIVE. This operation is split into an INSERT and DELETE operation. This operation creates a delta dir followed by a delete dir.
READ operation	READ operations always read a consistent snapshot of the data.	READ operations always read a consistent snapshot of the data.
Supported file format	Supports any file formats.	Supports only ORC.
Compactions	Minor and major compactions are supported.	Minor compactions can be created, which means several delta and delete directories can be compacted into one delta and delete directory. Major compactions are also supported.

File structure of FULL ACID transational table

Hive 3 achieves atomicity and isolation of operations on transactional tables by using techniques in write, read, insert, create, delete, and update operations that involve delta files, which can provide query status information and help you troubleshoot query problems.

Compaction of Data in FULL ACID Transactional Table

As administrator, you need to manage compaction of delta files that accumulate during data ingestion. Compaction is a process that performs critical cleanup of files.

Hive creates a set of delta files for each transaction that alters a table or partition and stores them in a separate delta directory. When the number of delta and delete directories in the table grow, the read performance will be impacted, since reading is a process of merging the results of valid transactions. To avoid any compromise on the read performance, occasionally Hive performs compaction, namely minor and major. This process merges these directories while preserving the transaction information.

To initiate automatic compaction, you must enable it using Cloudera Manager. For more information on managing the compaction process, see the link provided under Related Information.

Using HLL Datasketch Algorithms in Impala

You can use Datasketch algorithms (HLL) for queries that take too long to calculate exact results due to very large data sets (e.g. number of distinct values).

You may use data sketches (HLL algorithms) to generate approximate results that are much faster to retrieve. HLL is an algorithm that gives approximate answers for computing the number of distinct values in a column. The value returned by this algorithm is similar to the result of COUNT(DISTINCT col) and the NDV function integrated with Impala. However, HLL algorithm is much faster than COUNT(DISTINCT col) and the NDV function and is less memory-intensive for columns with high cardinality.

These HLL algorithms create “sketches”, which are data structures containing an approximate calculation of some facet of the data and which are compatible regardless of which system reads or writes them. For e.g., This particular algorithm works in two phases. First it creates a sketch from the dataset provided for the algorithm and then as the second step in the process you can read the estimate from the sketch. Using this approach it’s also possible to create granular sketches for a specific dataset (e.g. one sketch per partition) and later on use them to provide answers for different count(distinct) queries without reading the actual data as they can also be merged together. Using HLL you can create sketches by Hive and read it with Impala for getting the estimate and vice versa.

Usage Notes

Query results produced by the HLL algorithm are approximate but within well defined error bounds. Because the number of distinct values in a column returned by this HLL algorithm is an estimate, it might not reflect the precise number of different values in the column, especially if the cardinality is very high.

Data Types

The following is a list of currently supported data types of the datasets that these HLL functions are compatible with.

Supported:

1. Numeric Types:

   TINYINT, INT, BIGINT, FLOAT, DOUBLE
2. String types:

   STRING, CHAR, VARCHAR
3. Other types:

   BINARY

Unsupported:

1. Complex types
2. Some of the scalar types: DECIMAL, TIMESTAMP, BOOLEAN, SMALLINT, DATE

File Formats

The currently supported file formats are ORC and Parquet. Since Impala does not yet have read support for materialized views, it is required for Hive to write the sketches into regular tables (i.e. not into a materialized view), using the BINARY type. Then Impala can be used to read these sketches and calculate the estimate as Hive does.

Limitations:

You can write HLL sketches into a text table, however it may not work as the serialized sketches can contain characters that are special for text format.

Internal Configuration

The configuration is hard coded and the level of compression is HLL_4 and the number of buckets in the HLL array is k=12 (2 power of 12).

Using HLL Sketches

This datasketch algorithm works in two phases.

- First it creates a sketch from the dataset provided for the algorithm.
- As the second step in the process the function provides an estimate from the sketch.

The following example shows how to create a sketch and to create an estimate from the sketch.

Creating a sketch

This query creates a sketch using the ds_hll_sketch function from a column date_string_col containing the supported data type.

```
Select ds_hll_sketch(date_string_col) from tableName;
```

This query receives a primitive expression and returns a serialized HLL sketch that is not in a readble format.

Estimating the number of distinct values in a column

The following query returns a cardinality estimate (similarly to ndv()) for that particular column (date_string_col) by using the HLL function ds_hll_estimate.

```
Select ds_hll_estimate(ds_hll_sketch(date_string_col)) from tableName;
```

This query receives a primitive expression (column name) and then creates a sketch from the column, and then passes the sketch to the outer function that gives a cardinality estimate.

```
+-------------------------------------------------+
| ds_hll_estimate(ds_hll_sketch(date_string_col))|
+-------------------------------------------------+
| 729                                             |
+-------------------------------------------------+
Fetched 1 row(s) in 0.25s
```

The above result is similar to the results received using the count(distinct col) or the ndv() approach.

```
Select count(distinct date_string_col) from tableName;
+-------------------------------------------------+
| count (distinct date_string_col)                |
+-------------------------------------------------+
| 730                                             |
+-------------------------------------------------+
```

```
Select ndv(distinct date_string_col) from tableName;
+-------------------------------------------------+
| ndv (date_string_col)                           |
+-------------------------------------------------+
| 736                                             |
+-------------------------------------------------+
```

Inserting the derived sketch into a table

You can create a sketch and save it for later use. If you save the sketches to a table or view with the same granularity (e.g. one sketch for each partition in a table) then later you can simply read the sketch and get the estimate almost for free as the real cost here is just the sketch creation.

The following example shows the creation of a sketch for the column (date_string_col) from the table “tableName” and saving the created sketch into another table called sketch_table and later reading the sketch and its estimate from the saved table called sketch_table.

```
   select year, month, ds_hll_estimate(ds_hll_sketch(date_string_col)) from
 tableName group by year, month;
   +-------------------------------------------------------------+
   | year | month | ds_hll_estimate(ds_hll_sketch(date_string_col))|
   +------+-------+-------------------------------------------------+
   | 2010 | 5     | 31                                              |
   | 2010 | 11    | 30                                              |
   | 2010 | 6     | 30                                              |
   | 2010 | 2     | 28                                              |
   | 2010 | 10    | 31                                              |
   | 2009 | 11    | 30                                              |
   | 2009 | 7     | 31                                              |
   | 2009 | 10    | 31                                              |
```

```
| 2010 | 11    | 31                                                |
| 2009 | 7     | 31                                                |
| 2010 | 10    | 31                                                |
| 2010 | 8     | 30                                                |
| 2010 | 5     | 31                                                |
| 2009 | 7     | 30                                                |
| 2010 | 4     | 31                                                |
| 2010 | 12    | 30                                                |
| 2009 | 9     | 31                                                |
| 2009 | 8     | 30                                                |
| 2010 | 6     | 28                                                |
| 2010 | 3     | 31                                                |
| 2010 | 4     | 31                                                |
| 2010 | 2     | 30                                                |
| 2009 | 1     | 31                                                |
| 2009 | 12    | 30                                                |
+-------------------------------------------------------------+
```

Insert the created sketch into another table:

```
insert into sketch_table select year, month, ds_hll_sketch(date_string_
col) from tableName group by year, month;
```

Verify the number of rows modified using count function in the table:

```
select count(1) from sketch_table;
```

You can also verify it by getting the estimates from the sketches stored in the table using the following query:

```
select year, month, ds_hll_estimate(sketch_col) from sketch_table;
```

Combining Sketches

You can combine sketches to allow complex queries to be computed from a few number of simple sketches.

```
select ds_hll_estimate(ds_hll_union(sketch_col)) from sketch_table;
```

Non-deterministic vs Deterministic Results

Based on the characteristics of a query, DataSketches HLL might return results in a non-deterministic manner. During the cache populating phase it returns exact results but in the approximating phase the algorithm can be sensitive to the order of data it receives. As a result you might see the results differ in consecutive runs but all of them will be within the error range of the algorithm.

Due to the characteristics of the algorithm, and to how it was implemented in Impala, the results might be non-deterministic when the query is executed on one node and the input data set is big enough that the algorithm starts approximating instead of giving exact results.

Inter-opreability between Hive and Impala

From the above examples you will notice that the NDV function is much faster than the Datasketches. But NDV is used only by Impala and not by Hive whereas the Datasketches are implemented in Hive as well as in Impala. In

order to maintain the inter-operability between Hive and Impala we recommend to use Datasketches to calculate the estimate. A sketch created by Hive using Datasketches can be fed to Impala to produce an estimate.

Using KLL Datasketch Algorithms in Impala

You can use the stochastic streaming algorithm (KLL) that uses the percentile/quantile functions to statistically analyze the approximate distribution of comparable data from a very large stream.

Use the ds_kll_quantiles() or its inverse functions the Probability Mass Function ds_kll_PMF() and the Cumulative Distribution Function ds_kll_CDF() to obtain the analysis. Query results produced by this KLL algorithm are approximate but within well defined error bounds.

Data Types and File Formats

KLL function currently supports only floats. If you sketch an int data the KLL function converts it into float.

You can sketch data from any file formats. However it is recommended to save the sketches into parquet tables. It is not recommended to save the sketches to text format.

Using KLL Sketches

This datasketch algorithm works in two phases.

- First it creates a sketch from the dataset provided for the algorithm.
- As the second step in the process you can read the estimate from the sketch.

Using this approach you can create granular sketches for a specific dataset (e.g. one sketch per partition) and later use them to provide answers for different quantile queries without reading the actual data as they can also be merged together. Using KLL you can create sketches by Hive and read it with Impala for getting the estimate and vice versa.

Creating a sketch

The following example shows how to create a sketch using the ds_kll_sketch function from a column (float_col) containing the supported data type. This created sketch is a data structure containing an approximate calculation of some facet of the data and which are compatible regardless of which system reads or writes them.

```
select ds_kll_sketch(float_col) from tableName;
```

where ds_kll_sketch() is an aggregate function that receives a float parameter (e.g. a float column of a table) and returns a serialized Apache DataSketches KLL sketch of the input data set wrapped into STRING type. Since the serialized data might contain unsupported characters this query may error out. So another approach to this method is to insert the derived sketch into a table or a view and later use for quantile approximations.

Inserting the derived sketch into a table

If you save the sketches to a table or view with the same granularity (e.g. one sketch for each partition in a table) then later you can simply read the sketch for quantile approximation.

```
insert into table_name select ds_kll_sketch(float_col) from tableName;
```

Debugging the created sketch

To troubleshoot the sketch created by the ds_kll_sketch function, use ds_kll_stringify on the sketch. ds_kll_stringify () receives a string that is a serialized Apache DataSketches sketch and returns its stringified format by invoking the related function on the sketch's interface.

```
select ds_kll_stringify(ds_kll_sketch(float_col)) from tableName;
```

```
+-------------------------------------------+
| ds_kll_stringify(ds_kll_sketch(float_col))|
+-------------------------------------------+
| ### KLL sketch summary:                   |
|    K                     : 200            |
|    min K                 : 200            |
|    M                     : 8              |
|    N                     : 100            |
|    Epsilon               : 1.33%          |
|    Epsilon PMF           : 1.65%          |
|    Empty                 : false          |
|    Estimation mode       : false          |
|    Levels                : 1              |
|    Sorted                : false          |
|    Capacity items        : 200            |
|    Retained items        : 100            |
|    Storage bytes         : 432            |
|    Min value             : 0              |
|    Max value             : 9              |
| ### End sketch summary                    |
+-------------------------------------------+
```

Determining the number of datasets sketched

To determine the number of records sketched into this sketch, use ds_kll_n function.

```
select ds_kll_n(ds_kll_sketch(float_col)) from tableName;
```

```
+-------------------------------------+
| ds_kll_n(ds_kll_sketch(float_col)) |
+-------------------------------------+
| 100                                 |
+-------------------------------------+
```

Calculate Quantile

This function ds_kll_quantile() function receives two parameters, a STRING parameter that contains a serialized KLL sketch and a DOUBLE (0.5) that represents the rank of the quantile in the range of [0,1]. E.g. rank=0.5 means the approximate median of all the sketched data.

```
+-------------------------------------------------+
| ds_kll_quantile(ds_kll_sketch(float_col), 0.5) |
+-------------------------------------------------+
| 4.0                                             |
+-------------------------------------------------+
```

This query returns a data (4.0) that is bigger than the 50% of the data.

Calculating quantiles

To calculate the quantiles for multiple rank parameters, use the function ds_kll_quantiles_as_string () that is very similar to ds_kll_quantile() but this function receives any number of rank parameters and returns a comma separated string that holds the results for all of the given ranks.

```
    select ds_kll_quantiles_as_string(ds_kll_sketch(float_col), 0.5, 0.6, 1)
 from tableName;
```

```
+-------------------------------------------------------------------+
| ds_kll_quantiles_as_string(ds_kll_sketch(float_col), 0.5, 0.6, 1) |
+-------------------------------------------------------------------+
| 4, 5, 9                                                           |
+-------------------------------------------------------------------+
```

Calculate Rank

This rank function ds_kll_rank() receives two parameters, a STRING that represents a serialized DataSketches KLL sketch and a float to provide a probing value in the sketch.

```
select ds_kll_rank(ds_kll_sketch(float_col), 4) from tableName;
```

```
+-------------------------------------------------+
| ds_kll_rank(ds_kll_sketch(float_col), 4)        |
+-------------------------------------------------+
| 0.48                                            |
+-------------------------------------------------+
```

This query returns a DOUBLE that is the rank of the given probing value in the range of [0,1]. E.g. a return value of 0.48 means that the probing value given as parameter is greater than 48% of all the values in the sketch.

Note: This is only an approximate calculation.

Calculate Probability Mass Function (PMF)

This Probabilistic Mass Function (PMF) ds_kll_pmf_as_string()receives a serialized KLL sketch and one or more float values to represent ranges in the sketched values. In the following example, the float values [1, 3, 4, 8] represent the following ranges: (-inf, 1), [1, 3), [3, 4), [4, 8) [8, +inf)

Note: The input values for the ranges have to be unique and monotonically increasing.

```
select ds_kll_pmf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) from table
Name
```

```
+-----------------------------------------------------------+
| ds_kll_pmf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) |
+-----------------------------------------------------------+
| 0.12, 0.24, 0.12, 0.36, 0.16                              |
+-----------------------------------------------------------+
```

This query returns a comma separated string where each value in the string is a number in the range of [0,1] and shows what percentage of the data is in the particular ranges.

Calculate Cumulative Distribution Function (CDF)

This Cumulative Distribution Function (CDF) ds_kll_cmf_as_string()receives a serialized KLL sketch and one or more float values to represent ranges in the sketched values. In the following example, the float values [1, 3, 4, 8] represents the following ranges: (-inf, 1), (-inf, 3), (-inf, 4), (-inf, 8), (-inf, +inf)

Note: The input values for the ranges have to be unique and monotonically increasing.

```
select ds_kll_cdf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) from table
Name;
```

```
    +-------------------------------------------------------------+
    | ds_kll_cdf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) |
    +-------------------------------------------------------------+
    | 0.12, 0.36, 0.48, 0.84, 1                                   |
    +-------------------------------------------------------------+
```

This query returns a comma separated string where each value in the string is a number in the range of [0,1] and shows what percentage of the data is in the particular ranges.

Calculate the Union

To take advantage of the UNION function, create granular sketches for a specific dataset (one sketch per partition), write these sketches to a separate table and based on the partition you are interested in, you can UNION the relevant sketches together to get an estimate.

```
insert into sketch_tbl select ds_kll_sketch(float_col) from tableName;
```

```
select ds_kll_quantile(ds_kll_union(sketch_col), 0.5) from sketch_tbl
    where partition_col=1 OR partition_col=5;
```

This function ds_kll_union() receives a set of serialized Apache DataSketches KLL sketches produced by ds_kll_s ketch() and merges them into a single sketch.

Automatic Invalidation/Refresh of Metadata

In this release, you can invalidate or refresh metadata automatically after changes to databases, tables or partitions render metadata stale. You control the synching of tables or database metadata by basing the process on events. You learn how to access metrics and state information about the event processor.

When tools such as Hive and Spark are used to process the raw data ingested into Hive tables, new HMS metadata (database, tables, partitions) and filesystem metadata (new files in existing partitions/tables) are generated. In previous versions of Impala, in order to pick up this new information, Impala users needed to manually issue an INVALIDATE or REFRESH commands.

When automatic invalidate/refresh of metadata is enabled,, the Catalog Server polls Hive Metastore (HMS) notification events at a configurable interval and automatically applies the changes to Impala catalog.

Impala Catalog Server polls and processes the following changes.

- Invalidates the tables when it receives the ALTER TABLE event.
- Refreshes the partition when it receives the ALTER, ADD, or DROP partitions.
- Adds the tables or databases when it receives the CREATE TABLE or CREATE DATABASE events.

- Removes the tables from catalogd when it receives the DROP TABLE or DROP DATABASE events.
- Refreshes the table and partitions when it receives the INSERT events.

  If the table is not loaded at the time of processing the INSERT event, the event processor does not need to refresh the table and skips it.
- Changes the database and updates catalogd when it receives the ALTER DATABASE events. The following changes are supported. This event does not invalidate the tables in the database.
  - Change the database properties
  - Change the comment on the database
  - Change the owner of the database
  - Change the default location of the database

    Changing the default location of the database does not move the tables of that database to the new location. Only the new tables which are created subsequently use the default location of the database in case it is not provided in the create table statement.

This feature is controlled by the ##hms_event_polling_interval_s flag. Start the catalogd with the ##hms_event_polling_interval_s flag set to a positive integer to enable the feature and set the polling frequency in seconds. We recommend the value to be less than 5 seconds.

The following use cases are not supported:

- When you bypass HMS and add or remove data into table by adding files directly on the filesystem, HMS does not generate the INSERT event, and the event processor will not invalidate the corresponding table or refresh the corresponding partition.

  It is recommended that you use the LOAD DATA command to do the data load in such cases, so that event processor can act on the events generated by the LOAD command.
- The Spark API that saves data to a specified location does not generate events in HMS, thus is not supported. For example:

  ```
  Seq((1, 2)).toDF("i", "j").write.save("/user/hive/warehouse/spark_etl.db/
  customers/date=01012019")
  ```

Disable Event Based Automatic Metadata Sync

When the ##hms_event_polling_interval_s flag is set to a non-zero value for your catalogd, the event-based automatic invalidation is enabled for all databases and tables. If you wish to have the fine-grained control on which tables or databases need to be synced using events, you can use the impala.disableHmsSync property to disable the event processing at the table or database level.

This feature can be turned off by setting the ##hms_event_polling_interval_s flag set to 0.

When you add the DBPROPERTIES or TBLPROPERTIES with the impala.disableHmsSync key, the HMS event based sync is turned on or off. The value of the impala.disableHmsSync property determines if the event processing needs to be disabled for a particular table or database.

- If 'impala.disableHmsSync'='true', the events for that table or database are ignored and not synced with HMS.
- If 'impala.disableHmsSync'='false' or if impala.disableHmsSync is not set, the automatic sync with HMS is enabled if the ##hms_event_polling_interval_s global flag is set to non-zero.
- To disable the event based HMS sync for a new database, set the impala.disableHmsSync database properties in Hive as currently, Impala does not support setting database properties:

  ```
  CREATE DATABASE <name> WITH DBPROPERTIES ('impala.disableHmsSync'='true');
  ```
- To enable or disable the event based HMS sync for a table:

  ```
  CREATE TABLE <name> ... TBLPROPERTIES ('impala.disableHmsSync'='true' |
  'false');
  ```

- To change the event based HMS sync at the table level:

```
ALTER TABLE <name> SET TBLPROPERTIES ('impala.disableHmsSync'='true' | '
false');
```

When both table and database level properties are set, the table level property takes precedence. If the table level property is not set, then the database level property is used to evaluate if the event needs to be processed or not.

If the property is changed from true (meaning events are skipped) to false (meaning events are not skipped), you need to issue a manual INVALIDATE METADATA command to reset event processor because it doesn't know how many events have been skipped in the past and cannot know if the object in the event is the latest. In such a case, the status of the event processor changes to NEEDS_INVALIDATE.

Metrics for Event Based Automatic Metadata Sync

You can use the web UI of the catalogd to check the state of the automatic invalidate event processor.

By default, the debug web UI of catalogd is at http://*IMPALA-SERVER-HOSTNAME*:25020 (non-secure cluster) or https://*IMPALA-SERVER-HOSTNAME*:25020 (secure cluster).

Under the web UI, there are two pages that presents the metrics for HMS event processor that is responsible for the event based automatic metadata sync.

- /metrics#events
- /events

  This provides a detailed view of the metrics of the event processor, including min, max, mean, median, of the durations and rate metrics for all the counters listed on the /metrics#events page.

The /metrics#events page provides the following metrics about the HMS event processor.

Name	Description
events-processor.avg-events-fetch-duration	Average duration to fetch a batch of events and process it.
events-processor.avg-events-process-duration	Average time taken to process a batch of events received from the Metastore.
events-processor.events-received	Total number of the Metastore events received.
events-processor.events-received-15min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 15 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-received-1min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 1 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-received-5min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 5 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-skipped	Total number of the Metastore events skipped. Events can be skipped based on certain flags are table and database level. You can use this metric to make decisions, such as: • If most of the events are being skipped, see if you might just turn off the event processing. • If most of the events are not skipped, see if you need to add flags on certain databases.

Name	Description
events-processor.status	Metastore event processor status to see if there are events being received or not. Possible states are: • PAUSED The event processor is paused because catalog is being reset concurrently. • ACTIVE The event processor is scheduled at a given frequency. • ERROR • The event processor is in error state and event processing has stopped. • NEEDS_INVALIDATE The event processor could not resolve certain events and needs a manual INVALIDATE command to reset the state. • STOPPED The event processing has been shutdown. No events will be processed. • DISABLED The event processor is not configured to run.

Cloudera Runtime 1.0.0

Managing Apache Hive

Date published: 2021-12-01
Date modified: 2024-12-05

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ACID operations in Cloudera Data Warehouse.....4
Viewing transactions in Cloudera Data Warehouse.....4
Viewing transaction locks.....4

Data compaction.....5
Compaction tasks.....6
Starting compaction manually.....6
Viewing compaction progress.....7
Monitoring compaction.....7
Disabling automatic compaction.....8
Configuring compaction using table properties.....8
Configuring the compaction check interval.....9

Query vectorization.....9
Vectorization default.....10

ACID operations in Cloudera Data Warehouse

Apache Hive supports ACID (atomicity, consistency, isolation, and durability) v2 transactions at the row level without any configuration. Knowing what this support entails helps you determine the table type you create.

By default, managed tables are ACID tables. You cannot disable ACID transactions on managed tables, but you can change the Hive default behavior to create external tables by default to mimic legacy releases. Application development and operations are simplified with strong transactional guarantees and simple semantics for SQL commands. You do not need to bucket ACID v2 tables, so maintenance is easier. With improvements in transactional semantics, advanced optimizations, such as materialized view rewrites and automatic query cache, are available. With these optimizations, you can deploy new Hive application types.

A Hive operation is atomic. The operation either succeeds completely or fails; it does not result in partial data. A Hive operation is also consistent: After an application performs an operation, the results are visible to the application in every subsequent operation. Hive operations are isolated. Your operations do not cause unexpected side effects for other users. Finally, a Hive operation is durable. A completed operation is preserved in the event of a failure.

Hive operations are atomic at the row level instead of the table or partition level. A Hive client can read from a partition at the same time another client adds rows to the partition. Transaction streaming rapidly inserts data into Hive tables and partitions.

Viewing transactions in Cloudera Data Warehouse

As Administrator, you can view a list of open and aborted transactions.

Procedure

Enter a query to view transactions.
SHOW TRANSACTIONS
The following information appears in the output:

- Transaction ID
- Transaction state
- Hive user who initiated the transaction
- Host machine or virtual machine where transaction was initiated

Viewing transaction locks

As a Hive administrator, you can get troubleshooting information about locks on a table, partition, or schema.

Procedure

1. Enter a Hive query to check table locks.

```
SHOW LOCKS mytable EXTENDED;
```

2. Check partition locks.

```
SHOW LOCKS mytable PARTITION(ds='2018-05-01', hr='12') EXTENDED;
```

3. Check schema locks.

```
SHOW LOCKS SCHEMA mydatabase;
```

Output generally appears as follows.

- Database name
- Table name
- Partition, if the table is partitioned
- Lock state:
  - Acquired - transaction initiator hold the lock
  - Waiting - transaction initiator is waiting for the lock
  - Aborted - the lock has timed out but has not yet been cleaned
- Lock type:
  - Exclusive - the lock cannot be shared
  - Shared_read - the lock cannot be shared with any number of other shared_read locks
  - Shared_write - the lock may be shared by any number of other shared_read locks but not with other shared_write locks
- Transaction ID associated with the lock, if one exists
- Last time lock holder sent a heartbeat
- Time the lock was acquired, if it has been acquired
- Hive user who requested the lock
- Host machine or virtual machine on which the Hive user is running a Hive client
- Blocked By ID - ID of the lock causing current lock to be in Waiting mode, if the lock is in this mode

Related Information

Apache wiki transaction configuration documentation

Data compaction

As administrator, you need to manage compaction of delta files that accumulate during data ingestion. Compaction is a process that performs critical cleanup of files.

Hive creates a set of delta files for each transaction that alters a table or partition. By default, compaction of delta and base files occurs at regular intervals. Compactions occur in the background without affecting concurrent reads and writes.

There are two types of compaction:

- Minor

  Rewrites a set of delta files to a single delta file for a bucket.
- Major

  Rewrites one or more delta files and the base file as a new base file for a bucket.

  Carefully consider the need for a major compaction as this process can consume significant system resources and take a long time. Base and delta files for a table or partition are compacted.

  You can configure automatic compactions or do manual compactions. Start a major compaction during periods of low traffic. You use an ALTER TABLE statement to start compaction manually. A manual compaction either returns the accepted compaction request ID or shows the ID (and current state) of a compaction request for the very same target. The request is stored in the COMPACTION_QUEUE table.

The compactor initiator must run on only one HMS instance at a time.

Related Information

Apache Wiki transactions and compaction documentation

Compaction tasks

Compaction in Hive goes hand-in-hand with Hive ACID. Compaction is not, however, necessarily required for Hive ACID. You need to understand when you want, or do not want, compaction to occur.

If you confine your ACID operations tables to full reloads and some delta merges, the performance is steady. As tables are always rewritten (dropped and recreated), there is no need for compaction. Consider disabling compaction by

Compaction occurs for the following reasons:

- You explicitly trigger compaction on a table or partition.
- Automatic compaction finds something to compact.

You run an ALTER TABLE statement to start explicit compaction. Automatic compaction happens without your intervention when Hive periodically crawls through the transaction information, finds tables/partitions affected and those fit for the pre-defined conditions, and marks them for compaction. Conditions are based on the number of deltas, amount of change, and so on.

Compaction of sorted tables is not supported.

In a data pipeline, creating staging or temporary tables can significantly increase the compaction throughput. Avoid compaction of these tables.

Starting compaction manually

You manually start compaction when automatic compaction fails for some reason. You can start compaction by running a Hive statement.

About this task

You can run compaction pseudo-synchronously using the AND WAIT clause. Compaction actually occurs asynchronously, but seems synchronous. The compaction request is recorded and queued, and remains in a waiting cycle, querying the status of the compaction in the background until a failure, success, or timeout occurs. The hive.compactor.wait.timeout (default: 300s) property sets the timeout.

Start compaction using a query

You use the following syntax to issue a query that starts compaction:

```
 ALTER TABLE tablename [PARTITION (partition_key='partition_value' [,...])]
COMPACT 'compaction_type'
```

Required role: DWAdmin

Before you begin

- Files you are compacting must be in the ORC format.
- Compaction must be enabled (initiator hive.compactor.initiator.on=true)

Procedure

1. Run a query to start a major compaction of a table.

   ```
   ALTER TABLE mytable COMPACT 'major'
   ```

   Use the COMPACT 'minor' clause to run a minor compaction. ALTER TABLE compacts tables even if the NO_AUTO_COMPACTION table property is set.
2. Start compaction in a pseudo-synchronous way.
   ALTER TABLE mydb.mytable PARTITION (mypart='myval') COMPACT 'MAJOR' AND WAIT;

Viewing compaction progress

You view the progress of compactions by running Hive queries.

About this task

The ‘SHOW COMPACTIONS’ statement is not authorized. Every user can view compactions and can see the current state of compactions.

Procedure

Enter the query to view the progress of compactions.
SHOW COMPACTIONS;

- Unique internal ID
- Database name
- Table name
- Partition name
- Major or minor compaction
- Compaction state:
  - Initiated - waiting in queue
  - Working - currently compacting
  - Ready for cleaning - compaction completed and old files scheduled for removal
  - Failed - the job failed. Details are printed to the metastore log.
  - Succeeded
  - Attempted - initiator attempted to schedule a compaction but failed. Details are printed to the metastore log.
- Thread ID
- Start time of compaction
- Duration
- Job ID - ID of the submitted MapReduce job

Monitoring compaction

You can query the SYS database to monitor compactions.

About this task

Querying the COMPACTIONS table provides information about completed and queued compactions.

Procedure

Run a query to find failed compactions.

```
SELECT * FROM SYS.COMPACTIONS WHERE C_STATE IN ("failed", "did not initiate"
);
```

Disabling automatic compaction

You can disable automatic compaction of a particular Hive ACID table by setting a Hive table property. By default, compaction is enabled, so you must enter an ALTER TABLE command to disable it.

About this task

Compaction of a full ACID table is skipped under the following conditions:

- Another compaction is either running or already initiated for the target.
- The compaction target table is already dropped.
- Table is explicitly configured to be skipped by the auto-compaction

Compaction of an insert-only, ACID table is skipped if hive.compactor.compact.insert.only is set to false (turned off). The default is true. Although you can disable automatic compaction, tables still can be compacted if you explicitly request compaction.Disabling automatic compaction does not prevent you from performing manual compaction.

The compaction auto-initiator can be disabled on service instance level (disabled by default). You can independently enable or disable compaction workers on service instance level. Compaction merges only the bucket files with the same index and keeps the same number of bucket files in base. Compaction does not rebalance the rows between the buckets.

Procedure

At the Hive JDBC client prompt, in the database of the target table, alter the TBLPROPERTIES.

```
ALTER TABLE my_t SET TBLPROPERTIES ('NO_AUTO_COMPACTION'='true');
```

Configuring compaction using table properties

You see how to configure compaction using table properties and learn about the advantage of using this method of configuration.

About this task

You can configure compaction using table properties. Using table properties, you can group all table and partition compactions into a specific queue to match your use case. You can also size the compactor job based on the tables parameters like size and compression.

Procedure

Set table properties to adjust the compaction initiator properties.

```
ALTER TABLE mydb.mytable
SET TBLPROPERTIES (
'compactorthreshold.hive.compactor.delta.pct.threshold'='0.2f',
'compactorthreshold.hive.compactor.delta.num.threshold'='20');
```

These properties change thresholds, as the names imply: the deltas/base size percentage override threshold and the number of deltas threshold.

Configuring the compaction check interval

You need to know when and how to control the compaction process checking, performed in the background, of the file system for changes that require compaction.

When you turn on the compaction initiator, consider setting the hive.compactor.check.interval property. This property determines how often the initiator should search for possible tables, partitions, or compaction. By default, the value for this property is set to 300 seconds. Decreasing hive.compactor.check.interval has the following effect:

- Reduces the time it takes for compaction to be started for a table or partition that requires compaction.
- Requires several calls to the file system for each table or partition that has undergone a transaction since the last major compaction, resulting in increases to the load on the filesystem.

The compaction initiator first checks the completed transactions (COMPLETED_TXN_COMPONENTS), excluding those that already have completed compactions, searching for potential compaction targets. The search of the first iteration includes all transactions. Further searching of iterations are limited to the time-frame since the last iteration.

To configure the compaction check interval, set the hive.compactor.check.interval. For example:

From the Cloudera Data Warehouse service, go to the corresponding Database Catalog CONFIGURATIONS Metastore and set the value for the hive.compactor.check.interval property under the hive-site configuration file.

Query vectorization

You can use vectorization to improve instruction pipelines for certain data and queries and to optimize how Hive uses the cache. Vectorization processes batches of primitive types on the entire column rather than one row at a time.

Unsupported functionality on vectorized data

Some functionality is not supported on vectorized data:

- DDL queries
- DML queries other than single table, read-only queries
- Formats other than Optimized Row Columnar (ORC)

Supported functionality on vectorized data

The following functionality is supported on vectorized data:

- Single table, read-only queries

  Selecting, filtering, and grouping data is supported.
- Partitioned tables
- The following expressions:
  - Comparison: >, >=, <, <=, =, !=
  - Arithmetic plus, minus, multiply, divide, and modulo
  - Logical AND and OR
  - Aggregates sum, avg, count, min, and max

Supported data types

You can query data of the following types using vectorized queries:

- tinyint
- smallint
- int
- bigint

- date
- boolean
- float
- double
- timestamp
- stringchar
- varchar
- binary

Vectorization default

Vectorized query execution can affect performance. You need to be aware of the Boolean default value of hive.vectorized.execution.enabled.

Vectorized query execution is enabled by default (true). Vectorized query execution processes Hive data in batch, channeling a large number of rows of data into columns, foregoing intermediate results. This technique is more efficient than the MapReduce execution process that stores temporary files.

Cloudera Runtime 7.2.11

Apache Hive Materialized View Commands

Date published: 2019-08-21
Date modified: 2021-09-08

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ALTER MATERIALIZED VIEW REBUILD..... 4

ALTER MATERIALIZED VIEW REWRITE..... 4

CREATE MATERIALIZED VIEW.....5

DESCRIBE EXTENDED and DESCRIBE FORMATTED.....6

DROP MATERIALIZED VIEW.....8

SHOW MATERIALIZED VIEWS.....8

ALTER MATERIALIZED VIEW REBUILD

You must rebuild the materialized view to keep it up-to-date when changes to the data occur.

Syntax

```
ALTER MATERIALIZED VIEW [db_name.]materialized_view_name REBUILD;
```

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by the name of the materialized view in dot notation.

Description

A rewrite of a query based on a stale materialized view does not occur automatically. If you want a rewrite of a stale or possibly stale materialized view, you can force a rewrite. For example, you might want to use the contents of a materialized view of a non-transactional table because the freshness of such a table is unknown. To enable rewriting of a query based on a stale materialized view, you can run the rebuild operation periodically and set the following property: hive.materializedview.rewriting.time.window. For example, SET hive.materializedview.rewriting.time.window=10min;

Example

ALTER MATERIALIZED VIEW mydb.mv1 REBUILD;

Related Information

Using materialized views

Periodically rebuild a materialized view

ALTER MATERIALIZED VIEW REWRITE

You can enable or disable the rewriting of queries based on a particular materialized view.

Syntax

```
ALTER MATERIALIZED VIEW [db_name.]materialized_view_name ENABLE|DISABLE REWR
ITE;
```

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by the name for the materialized view in dot notation.

Description

To optimize performance, by default, Hive rewrites a query based on materialized views. You can change this behavior to manage query planning and execution manually. By setting the hive.materializedview.rewriting global property, you can manage query rewriting based on materialized views for all queries.

Example

ALTER MATERIALIZED VIEW mydb.mv1 DISABLE REWRITE;

Related Information

Using materialized views

CREATE MATERIALIZED VIEW

If you are familiar with the CREATE TABLE AS SELECT (CTAS) statement, you can quickly master how to create a materialized view.

Syntax

```
CREATE MATERIALIZED VIEW [IF NOT EXISTS] [db_name.]materialized_view_name
   [DISABLE REWRITE]
   [COMMENT materialized_view_comment]
   [PARTITIONED ON (column_name, ...)]
   [
     [ROW FORMAT row_format]
     [STORED AS file_format]
        | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (ser
de_property_name=serde_property_value, ...)]
   ]
   [LOCATION file_path]
   [TBLPROPERTIES (tbl_property_name=tbl_property_value, ...)]
   AS
   <query>;
```

Required Parameters

QUERY

The query to run for results that populate the contents of the materialized view.

Optional Parameters

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by a name, unique among materialized view names, for the materialized view. The name must be a valid a table name, including case-insensitive alphanumeric and underscore characters.

MATERIALIZED_VIEW_COMMENT

A string literal enclosed in single quotation marks.

COLUMN_NAME

A key that determines how to do the partitioning, which divides the view of the table into parts.

'STORAGE.HANDLER.CLASS.NAME'

The name of a storage handler, such as org.apache.hadoop.hive.druid.DruidStorageHandler, that conforms to the Apache Hive specifications for storage handlers in a table definition that uses the STORED BY clause. The default is hive.materializedview.fileformat.

SERDE_PROPERTY_NAME

A property supported by SERDEPROPERTIES that you specify as part of the STORED BY clause. The property is passed to the serde provided by the storage handler. When not specified, Hive uses the default hive.materializedview.serde.

SERDE_PROPERTY_VALUE

A value of the SERDEPROPERTIES property.

FILE_PATH

The location on the file system for storing the materialized view.

TBL_PROPERTY_NAME

A key that conforms to the Apache Hive specification for TBLPROPERTIES keys in a table.

TBL_PROPERTY_VALUE

The value of a TBLPROPERTIES key.

Usage

The materialized view creation statement meets the criteria of being atomic: it does not return incomplete results. By default, the optimizer uses materialized views to rewrite the query. You can store a materialized view in an external storage system using the STORED AS clause followed by a valid storage handler class name. You can set the DISABLE REWRITE option to alter automatic rewriting of the query at materialized view creation time. The table on which you base the materialized view, src in the example below, must be an ACID, managed table.

Examples

```
CREATE MATERIALIZED VIEW druid_t
  STORED BY 'org.apache.hadoop.hive.druid.DruidStorageHandler'
  AS SELECT a, b, c
  FROM src;
```

```
CREATE MATERIALIZED VIEW mv4
  LOCATION '/user/csso_max'
  AS SELECT empid, deptname, hire_date
  FROM emps JOIN depts
  ON (emps.deptno = depts.deptno)
  WHERE hire_date >= '2017-01-01';
```

Related Information

Apache Hive Wiki Hive Data Definition Language > Create Table and CTAS

Apache Hive Wiki StorageHandlers > DDL

Using materialized views

DESCRIBE EXTENDED and DESCRIBE FORMATTED

You can get extensive formatted and unformatted information about a materialized view.

Syntax

```
DESCRIBE [EXTENDED | FORMATTED] [db_name.]materialized_view_name;
```

DB_NAME

The database name.

MATERIALIZED_VIEW_NAME

The name of the materialized view.

Examples

Get summary, details, and formatted information about the materialized view in the default database and its partitions.

```
DESCRIBE FORMATTED default.partition_mv_1;
```

Example output is:

col_name	data_type	comment
# col_name	data_type	comment

col_name	data_type	comment
name	varchar(256)	
	NULL	NULL
# Partition Information	NULL	NULL
# col_name	data_type	comment
deptno	int	
	NULL	NULL
# Detailed Table Information	NULL	NULL
Database:	default	NULL
OwnerType:	USER	NULL
Owner:	hive	NULL
CreateTime:	Wed Aug 22 19:46:08 UTC 2018	NULL
LastAccessTime:	UNKNOWN	NULL
Retention:	0	NULL
Location:	hdfs://myserver:8020/warehouse/ tables pace/managed/hive/partition_mv_1	NULL
Table Type:	MATERIALIZED_VIEW	NULL
Table Parameters:	NULL	NULL
	COLUMN_STATS_ACCURATE	{\"BASIC_STATS\":\"true\"}
	bucketing_version	2
	numFiles	2
	numPartitions	2
	numRows	4
	rawDataSize	380
	totalSize	585
	transient_lastDdlTime	1534967168
	NULL	NULL
# Storage Information	NULL	NULL
SerDe Library:	org.apache.hadoop.hive.ql.io.orc.OrcSerde	NULL
InputFormat:	org.apache.hadoop.hive.ql.io.orc.OrcInputFor mat	NULL
OutputFormat:	org.apache.hadoop.hive.ql.io.orc.OrcOutputFo rmat	NULL
Compressed:	No	NULL
Num Buckets:	-1	NULL
Bucket Columns:	[]	NULL
Sort Columns:	[]	NULL
	NULL	NULL
# Materialized View Information	NULL	NULL
Original Query:	SELECT hire_date, deptno FROM emps W HERE deptno > 100 AND deptno < 200	NULL

col_name	data_type	comment
Expanded Query:	SELECT `hire_date`, `deptno` FROM (S ELECT `emps`.`hire_date`, `emps`.`dept no` FROM `default`.`emps` WHERE `e mps`.`deptno` > 100 AND `emps`.`deptno ` < 200) `default.partition_mv_1`	NULL
Rewrite Enabled:	Yes	NULL
Outdated for Rewriting:	No	NULL

Related Information

Using materialized views

DROP MATERIALIZED VIEW

You can avoid making a table name unusable by dropping a dependent materialized view before dropping a table.

Syntax

```
DROP MATERIALIZED VIEW [db_name.]materialized_view_name;
```

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by a name for the materialized view in dot notation.

Description

Dropping a table that is used by a materialized view is not allowed and prevents you from creating another table of the same name. You must drop the materialized view before dropping the tables.

Example

DROP MATERIALIZED VIEW mydb.mv1;

Related Information

Using materialized views

SHOW MATERIALIZED VIEWS

You can list all materialized views in the current database or in another database. You can filter a list of materialized views in a specified database using regular expression wildcards.

Syntax

```
SHOW MATERIALIZED VIEWS [IN db_name];
```

DB_NAME

The database name.

'IDENTIFIER_WITH_WILDCARDS'

The name of the materialized view or a regular expression consisting of part of the name plus wildcards. The asterisk and pipe (* and |) wildcards are supported. Use single quotation marks to enclose the identifier.

Examples

SHOW MATERIALIZED VIEWS;

```
+-----------------+--------------------+-----------------+
|     mv_name     |  rewrite_enabled   |      mode       |
+-----------------+--------------------+-----------------+
| # MV Name       | Rewriting Enabled  | Mode            |
| partition_mv_1  | Yes                | Manual refresh  |
| partition_mv_2  | Yes                | Manual refresh  |
| partition_mv_3  | Yes                | Manual refresh  |
+-----------------+--------------------+-----------------+
```

SHOW MATERIALIZED VIEWS '*1';

```
+-----------------+--------------------+-----------------+
|     mv_name     |  rewrite_enabled   |      mode       |
+-----------------+--------------------+-----------------+
| # MV Name       | Rewriting Enabled  | Mode            |
| partition_mv_1  | Yes                | Manual refresh  |
|                 | NULL               | NULL            |
+-----------------+--------------------+-----------------+
```

Related Information

Using materialized views

1.0.0

Monitoring Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-05

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Impala lineage **4**
Debug Web UI for Query Timeline **4**
Improvement in Catalog Observability **5**

Impala lineage

You can use the Atlas lineage graph to understand the source and impact of data and changes to data over time and across all your data.

Atlas collects metadata from Impala to represent the lineage among data assets. The Atlas lineage graph shows the input and output processes that the current entity participated in. Entities are included if they were inputs to processes that lead to the current entity or they are output from processes for which the current entity was an input. Impala processes follow this pattern.

Note that lineage is not updated between a table and views that the table is a part of when an Impala ALTER TABLE operation runs on the table.

Debug Web UI for Query Timeline

For a detailed report on how a query was executed and to understand the detailed performance characteristics of a query, you can use the built-in web server's UI and look at the Gantt chart.

The Gantt chart is an alternative to the profile command and is a graphical display in the WebUI that renders timing information and dependencies. This chart allows you to determine where queries are spending time instead of looking for timing information in the profile. Once you identify which fragments or operators are consuming the most time you may need to look at the summary profile or full profile to drill down further.

In subsequent releases, the timeline will be enhanced to display additional details so that looking at the profiles will not be necessary.

Here is the screenshot of the Gantt chart that shows the Query timeline of a TPC_DS query.

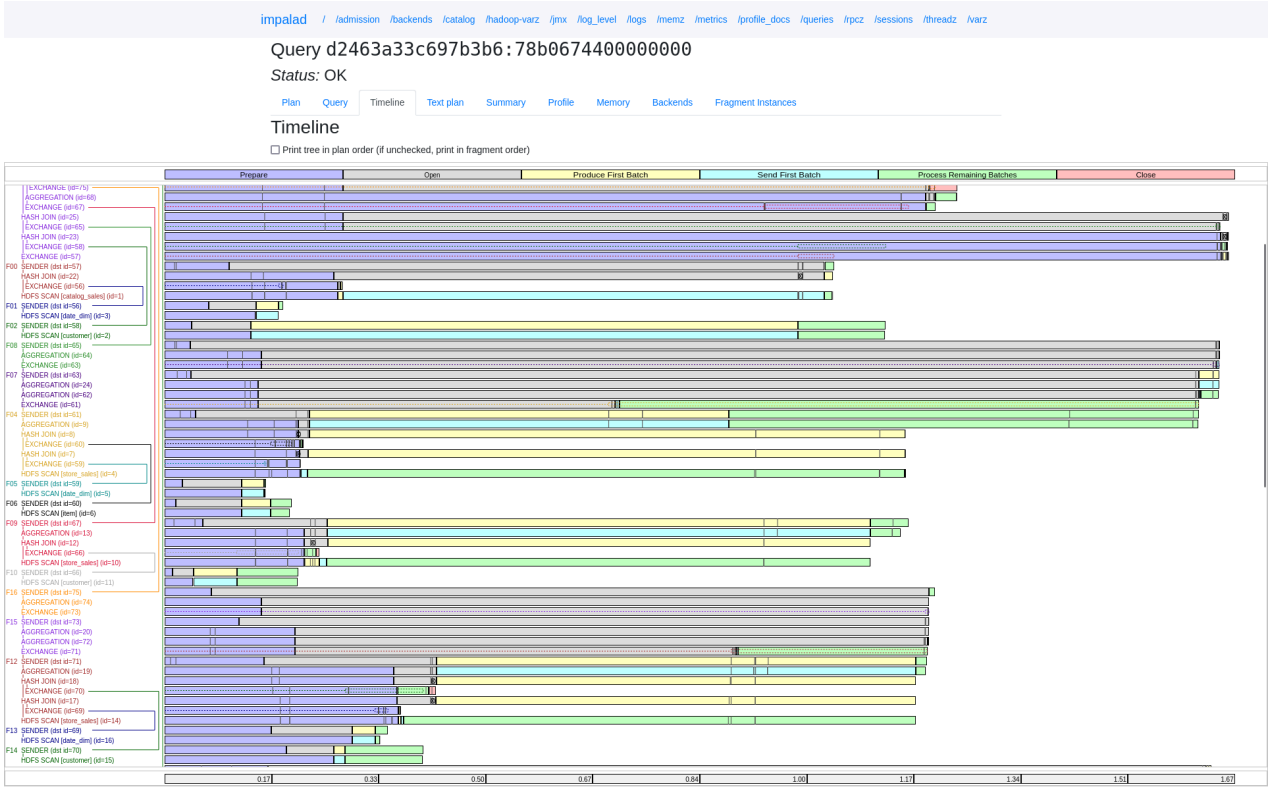

Improvement in Catalog Observability

This release brings significant enhancements to the Impala Catalog Web UI, addressing performance issues related to delays in processing Hive Metastore (HMS) events, which can lead to queries using outdated metadata

These improvements specifically target slow Data Definition Language (DDL) and Data Manipulation Language (DML) operations such as INSERT, LOAD DATA statements, and REFRESH operations. Additionally, it addresses operations stuck in various stages such as startup, among other scenarios. Prior to this release, troubleshooting these issues required analyzing logs, but now the process has been streamlined by enhancing the observability of the catalog across multiple dimensions, including the catalogd WebUI, DDL/DML profiling, and additional log reports on sluggish operations.

Key enhancements include:

- Catalogd WebUI: Utilize the catalogd WebUI to monitor catalog operations.
- DDL/DML Profiling: Monitor DDL/DML profile for detailed execution insights within catalogd.
- Log Reports: Analyze logs for a comprehensive review of slow operations, offering valuable insights into the root causes of performance issues.

These improvements empower users with a more user-friendly and effective means of diagnosing and resolving catalog-related performance challenges.

1.0.0

Securing Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-05

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Impala Authorization.....4
Row-level filtering in Impala with Ranger policies.....11

Impala Authorization

Authorization determines which users are allowed to access which resources, and what operations they are allowed to perform. You use Apache Ranger to enable and manage authorization in Impala.

Supported Ranger Features in Impala

- Resource-based and tag-based authorization policies
- Resource-based and tag-based column masking
- Row-level filtering is enabled by default

Using Resource-based Authorization Policies for Impala

In the Ranger Service Manager, you can use the Hadoop SQL preloaded resource-based and tag-based services and policies to authorize access to Impala:

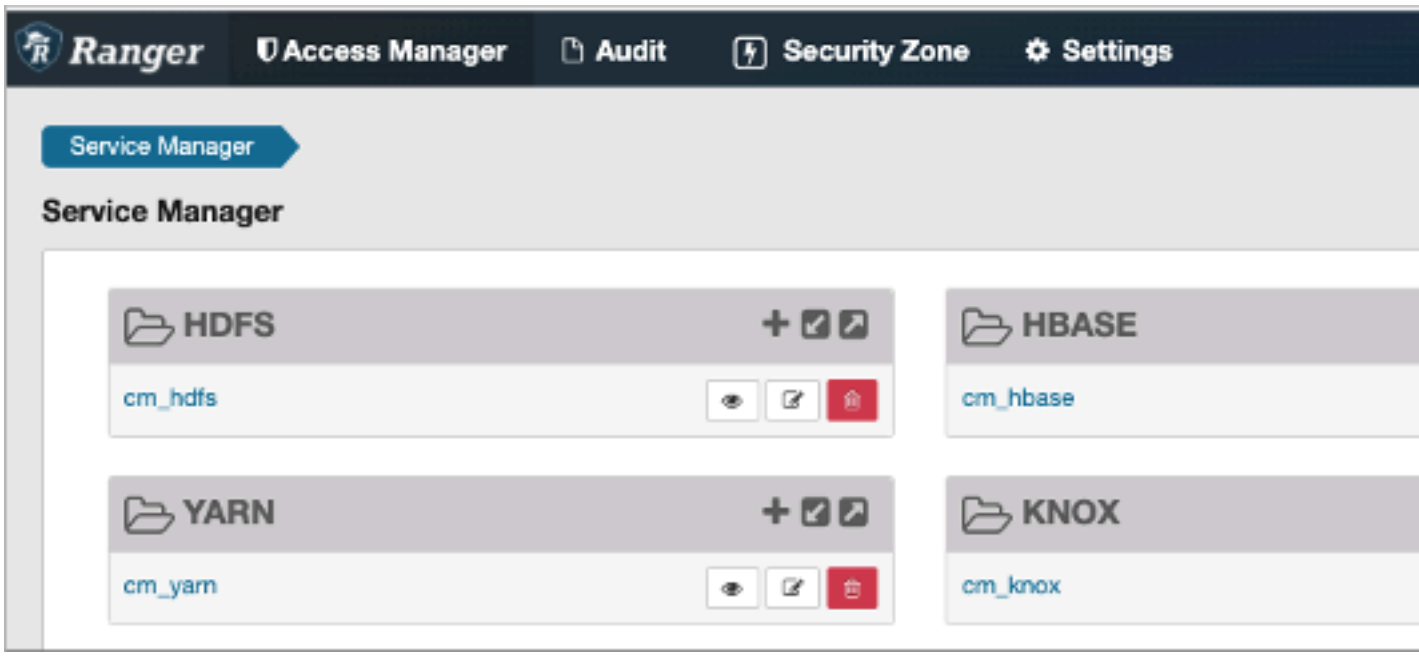

You can configure services for Impala as described in Using Ranger to Provide Authorization in Cloudera.

For example, you can edit the all-global policy for an Impala user:

For information about using tag-based policies, see Tag-based column masking in Hive with Ranger policies.

Privilege Model

You set up privileges through the GRANT and REVOKE statements in either Impala or Hive. Then both components use those same privileges automatically.

By default, when authorization is not enabled, Impala does all read and write operations with the privileges of the impala user, which is suitable for a development/test environment but not for a secure production environment. When authorization is enabled, Impala uses the OS user ID of the user who runs `impala-shell` or other client programs, and associates various privileges with each user.

Privileges can be granted on different resources in the schema and are associated with a level in the resource hierarchy. A privilege on a particular resource automatically inherits the same privilege of its parent.

The resource hierarchy is:

```
Server
    URI
    Database
        Table
            Column
        Function
```

The table-level privileges apply to views as well. Anywhere you specify a table name, you can specify a view name instead.

You can specify privileges for individual columns.

The table below lists the minimum level of privileges and the scope required to run SQL statements in Impala. The following notations are used:

- The SERVER resource type in Ranger implies all databases, all tables, all columns, all UDFs, and all URLs.
  - **Note:** For Impala server level access, you must add the user to the below listed default policies in Ranger:

    all - database, table, column

    all - database, udf

    all - url
- ANY denotes the CREATE, ALTER, DROP, SELECT, INSERT, or REFRESH privilege.
- ALL privilege denotes the SELECT, INSERT, CREATE, ALTER, DROP, and REFRESH privileges.
- VIEW_METADATA privilege denotes the SELECT, INSERT, or REFRESH privileges.
- The parent levels of the specified scope are implicitly supported. For example, if a privilege is listed with the TABLE scope, the same privilege granted on DATABASE and SERVER will allow the user to run that specific SQL statement on TABLE.

Note: For an "impala" user to access any managed tables under the warehouse root directory or external tables with directories outside the warehouse root directory, the "impala" user must have HDFS RW access to the underlying paths that are referenced in a query.

Without this HDFS RW access, you may see Impala queries failing with HDFS errors such as this:

Error(13): Permission denied

Root cause: RemoteException: Permission denied: user=impala, access=WRITE, inode="/path/external/table":hive:supergroup:drwxr-xr-x

For example, to be able to run CREATE VIEW, you need the CREATE privilege on the database and the SELECT privilege on the source table.

SQL Statement	Privileges	Object Type / Resource Type
SELECT	SELECT	TABLE

WITH SELECT	SELECT	TABLE
EXPLAIN SELECT	SELECT	TABLE
INSERT	INSERT	TABLE
EXPLAIN INSERT	INSERT	TABLE
TRUNCATE	INSERT	TABLE
LOAD	INSERT	TABLE
	ALL	URI
CREATE DATABASE	CREATE	SERVER
CREATE DATABASE LOCATION	CREATE	SERVER
	ALL	URI
CREATE TABLE	CREATE	DATABASE
CREATE TABLE LIKE	CREATE	DATABASE
	VIEW_METADATA	TABLE
CREATE TABLE AS SELECT	CREATE	DATABASE
	INSERT	DATABASE
	SELECT	TABLE
EXPLAIN CREATE TABLE AS SELECT	CREATE	DATABASE
	INSERT	DATABASE
	SELECT	TABLE
CREATE TABLE LOCATION	CREATE	TABLE
	ALL	URI
CREATE VIEW	CREATE	DATABASE
	SELECT	TABLE
ALTER DATABASE SET OWNER	ALL WITH GRANT	DATABASE
ALTER TABLE	ALL	TABLE
ALTER TABLE SET LOCATION	ALL	TABLE
	ALL	URI
ALTER TABLE RENAME	CREATE	DATABASE
	ALL	TABLE
ALTER TABLE SET OWNER	ALL WITH GRANT	TABLE
ALTER VIEW	ALL	TABLE
	SELECT	TABLE
ALTER VIEW RENAME	CREATE	DATABASE
	ALL	TABLE
ALTER VIEW SET OWNER	ALL WITH GRANT	VIEW
DROP DATABASE	ALL	DATABASE
DROP TABLE	ALL	TABLE
DROP VIEW	ALL	TABLE
CREATE FUNCTION	CREATE	DATABASE
	ALL	URI

DROP FUNCTION	ALL	DATABASE
COMPUTE STATS	ALL	TABLE
DROP STATS	ALL	TABLE
INVALIDATE METADATA	REFRESH	SERVER
INVALIDATE METADATA <table>	REFRESH	TABLE
REFRESH <table>	REFRESH	TABLE
REFRESH AUTHORIZATION	REFRESH	SERVER
REFRESH FUNCTIONS	REFRESH	DATABASE
COMMENT ON DATABASE	ALL	DATABASE
COMMENT ON TABLE	ALL	TABLE
COMMENT ON VIEW	ALL	TABLE
COMMENT ON COLUMN	ALL	TABLE
DESCRIBE DATABASE	VIEW_METADATA	DATABASE
DESCRIBE <table/view>	VIEW_METADATA	TABLE
If the user has the SELECT privilege at the COLUMN level, only the columns the user has access will show.	SELECT	COLUMN
USE	ANY	TABLE
SHOW DATABASES	ANY	TABLE
SHOW TABLES	ANY	TABLE
SHOW FUNCTIONS	VIEW_METADATA	DATABASE
SHOW PARTITIONS	VIEW_METADATA	TABLE
SHOW TABLE STATS	VIEW_METADATA	TABLE
SHOW COLUMN STATS	VIEW_METADATA	TABLE
SHOW FILES	VIEW_METADATA	TABLE
SHOW CREATE TABLE	VIEW_METADATA	TABLE
SHOW CREATE VIEW	VIEW_METADATA	TABLE
SHOW CREATE FUNCTION	VIEW_METADATA	DATABASE
SHOW RANGE PARTITIONS (Kudu only)	VIEW_METADATA	TABLE
UPDATE (Kudu only)	ALL	TABLE
EXPLAIN UPDATE (Kudu only)	ALL	TABLE
UPSERT (Kudu only)	ALL	TABLE
WITH UPSERT (Kudu only)	ALL	TABLE
EXPLAIN UPSERT (Kudu only)	ALL	TABLE
DELETE (Kudu only)	ALL	TABLE
EXPLAIN DELETE (Kudu only)	ALL	TABLE

The privileges not listed in the table above will be silently ignored by Impala.

Changing Privileges in Impala

Privileges are managed via the GRANT and REVOKE SQL statements that require the Ranger service enabled.

Privileges can be also managed in Ranger UI. Especially, for attribute-based access control, Ranger UI is required to manage authorization.

Impala authorization policies are listed in the Hive service section in Ranger UI.

REFRESH AUTHORIZATION is not required when you make the changes to privileges within Impala. The changes are automatically propagated.

Changing Privileges from Outside of Impala

If you make a change to privileges in Ranger from outside of Impala, e.g. adding a user, removing a user, modifying privileges, there are two options to propagate the change:

- Use the ranger.plugin.hive.policy.pollIntervalMs property to specify how often to do a Ranger refresh. The property is specified in ranger-impala-security.xml in the conf directory under your Impala home directory.
- Run the REFRESH AUTHORIZATION statement to force a refresh.

Warning: As INVALIDATE METADATA is an expensive operation, you should use it judiciously.

Granting Privileges on URI

URIs represent the file paths you specify as part of statements such as CREATE EXTERNAL TABLE and LOAD DATA. Typically, you specify what look like UNIX paths, but these locations can also be prefixed with hdfs:// to make clear that they are really URIs. To set privileges for a URI, specify the name of a directory, and the privilege applies to all the files in that directory.

URIs must start with hdfs://, s3a://, adl://, or file://. If a URI starts with an absolute path, the path will be appended to the default filesystem prefix. For example, if you specify:

```
GRANT ALL ON URI '/tmp';
```

The above statement effectively becomes the following where the default filesystem is HDFS.

```
GRANT ALL ON URI 'hdfs://localhost:20500/tmp';
```

When defining URIs for HDFS, you must also specify the NameNode. For example:

```
GRANT ALL ON URI file:///path/to/dir TO <principal>
          GRANT ALL ON URI hdfs://namenode:port/path/to/dir TO <principal>
```

Warning: Because the NameNode host and port must be specified, it is strongly recommended that you use High Availability (HA). This ensures that the URI will remain constant even if the NameNode changes. For example:

```
GRANT ALL ON URI hdfs://ha-nn-uri/path/to/dir TO <principal>
```

Note:

```
<principal> can be a user, or a group
```

Note:

- If a user uses Impala SQL engine to access the resource of the specified URL/URI provided as part of the SQL statement and as an admin if you must deny access permissions to this location for this particular user, you must add the permission "All" in the "HADOOP SQL" policy.
- However, if the user is using Hive as the execution engine and to deny access permission to a location, then you must add both the “Read” and “Write” permissions in the field of ‘Permissions’ for the corresponding deny conditions.
- If the same Ranger policy is shared by both Hive and Impala, then you must add “All”, “Read”, and “Write” to the field of ‘Permissions’ to enforce the policy.

Object Ownership

Object ownership for tables, views and databases is enabled by default in Impala.

To define owner specific privileges, go to Ranger UI and define appropriate policies on the {OWNER} user.

The CREATE statements implicitly make the user running the statement the owner of the object. For example, if *USER A* creates a database, *FOO*, via the CREATE DATABASE statement, *USER A* now owns the *FOO* database and is authorized to perform any operation on the *FOO* database.

An ownership can be transferred to another user or role via the ALTER DATABASE, ALTER TABLE, or ALTER VIEW with the SET OWNER clause.

Note: Currently, due to a known issue (IMPALA-8937), until the ownership information is fully loaded in the coordinator catalog cache, the owner of a table might not be able to see the table when executing the SHOW TABLES statement. The owner can still query the table.

Ranger Column Masking for Impala

Ranger column masking hides sensitive columnar data in Impala query output. For example, you can define a policy that reveals only the first or last four characters of column data. Column masking is enabled by default. To disable column masking, modify the following configuration property in all coordinators:

```
--enable_column_masking=false
```

This flag might be removed in a future release. The Impala behavior mimics Hive behavior with respect to column masking.

The following table lists all Impala-supported, built-in mask types for defining column masking in a policy using the Ranger REST API or Ranger UI:

Type	Name	Description	Transformer
MASK	Redact	Replace lowercase with 'x', uppercase with 'X', digits with '0'	mask({col})
MASK_SHOW_LAST_4	Partial mask: show last 4	Show last 4 characters; replace rest with 'x'	mask_show_last_n({col}, 4, 'x', 'x', 'x', -1, '1')
MASK_SHOW_FIRST_4	Partial mask: show first 4	Show first 4 characters; replace rest with 'x'	mask_show_first_n({col}, 4, 'x', 'x', 'x', -1, '1')
MASK_HASH	Hash	Hash the value	mask_hash({col})
MASK_NULL	Nullify	Replace with NULL	N/A
MASK_NONE	Unmasked (retain original value)	No masking	N/A
MASK_DATE_SHOW_YEAR	Date: show only year	Date: show only year	mask({col}, 'x', 'x', 'x', -1, '1', 1, 0, -1)
CUSTOM	Custom	Custom	N/A

The table includes the mask name as it appears in the Ranger UI.

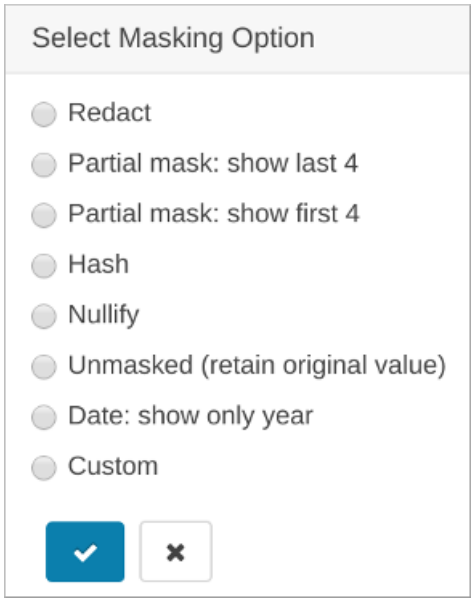

Ranger Column Masking Limitations in Impala

- Column masking introduces unused columns during the query analysis and, consequently, additional SELECT privileges checks on all columns of the masked table.
- Impala might produce more than one audit log entry for a column involved in a column masking policy under all of these conditions: the column appears in multiple places in a query, the query is rewritten, or the query is re-analyzed.
- Column masking policies, shared between Hive and Impala, might be affected by SQL/UDF differences between Hive and Impala, as shown in the following example.

  For instance, UTF-8 strings containing non-ASCII characters are not guaranteed to work properly. Suppose a column masking policy masks the last two characters of a string: s => mask_last_n(s, 2, 'x', 'x', 'x', 'x'). Applying this policy, Hive properly masks SQL## to SQLxx, but the Impala masking produces SQL##xx because each Chinese character is encoded in 3 bytes. Impala and Hive do not handle different lengths in the same way.

Limitations on Mask Functions

The mask functions in Hive are implemented through GenericUDFs. Even though Impala users can call Hive UDFs, Impala does not yet support Hive GenericUDFs, so you cannot use Hive's mask functions in Impala. However, Impala has builtin mask functions that are implemented through overloads. In Impala, when using mask functions, not all parameter combinations are supported. These mask functions are introduced in Impala 3.4

The following list includes all the implemented overloads.

- Overloads used by Ranger default masking policies,
- Overloads with simple arguments,
- Overload with all arguments in int type for full functionality. Char argument needs to be converted to their ASCII value.

To list the available overloads, use the following query:

```
show functions in _impala_builtins like "mask*";
```

Note: An error message that states "No matching function with signature: mask..." implies that Impala does not contain the corresponding overload.

Related Information

GRANT statement

REVOKE statement
Apache Ranger documentation

Row-level filtering in Impala with Ranger policies

Row-level filtering policies are similar to other Ranger access policies. Apache Ranger row-level filtering policy allows to set access policies for rows when reading from a table.

You can use Apache Ranger row-level filtering policies to set access policies for rows when reading from a table. You can set filters for specific users, groups, and conditions. This release adds a new feature flag enable_row_filtering which is set to be true by default. To enable row-filtering feature you must have set the column masking flag enable_column_masking to true since the row-level filtering depends on the column masking implementation. You can use this flag enable_row_filtering to disable this experimental feature as required.

Note: Note that row filtering policies apply prior to any column masking policies, because column masking policies apply only on result data of the target table/view.

The following limitations apply when using row-level filters:

- Row filtering policies on nested tables can't be applied when nested collection columns(e.g. array, map columns) are used directly in the FROM clause. Such queries are currently forbidden.

  This is an example of the currently supported version of a query on the table my_tbl (id int, int_array array<int>) that has a row filter "id = 0".

  ```
  select a.item from my_tbl t, t.int_array a
  ```

  However an equivalent query below is not currently supported since it uses the int_array column non-relatively.

  ```
  select item from my_tbl.int_array
  ```

  Such queries are forbidden and you must rewrite them to the supported format until further notice.
- For information on the steps to set the row-level filtering using Apache Ranger, see the link provided under Related Information.

  Note: Hive features that are not supported by Impala should not be used in the filter since Impala can't evaluate them.

Cloudera Runtime 7.1.0

Securing Apache Hive

Date published: 2019-08-21
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Hive access authorization.....4

Transactional table access..... 5

External table access..... 5

Token-based authentication for Cloudera Data Warehouse integrations..... 5

Hive access authorization

As administrator, you need to understand that the Hive default authorization for running Hive queries is insecure and what you need to do to secure your data. You need to set up Apache Ranger.

To limit Apache Hive access to approved users, Cloudera recommends and supports only Ranger. Authorization is the process that checks user permissions to perform select operations, such as creating, reading, and writing data, as well as editing table metadata. Apache Ranger provides centralized authorization for all Cloudera Runtime Services.

You can set up Ranger to protect managed, ACID tables or external tables using a Hadoop SQL policy. You can protect external table data on the file system by using an HDFS policy in Ranger.

Preloaded Ranger Policies

In Ranger, preloaded Hive policies are available by default. Users covered by these policies can perform Hive operations. All users need to use the default database, perform basic operations such as listing database names, and query the information schema. To provide this access, preloaded default database tables columns and information_schema database policies are enabled for group public (all users). Keeping these policies enabled for group public is recommended. For example, if the default database tables columns policy is disabled preventing use of the default database, the following error appears:

```
hive> USE default;
Error: Error while compiling statement: FAILED: HiveAccessControlException
Permission denied: user [hive] does not have [USE] privilege on [default]
```

Apache Ranger policy authorization

Apache Ranger provides centralized policy management for authorization and auditing of all Cloudera Runtime services, including Hive. All Cloudera Runtime services are installed with a Ranger plugin used to intercept authorization requests for that service, as shown in the following illustration.

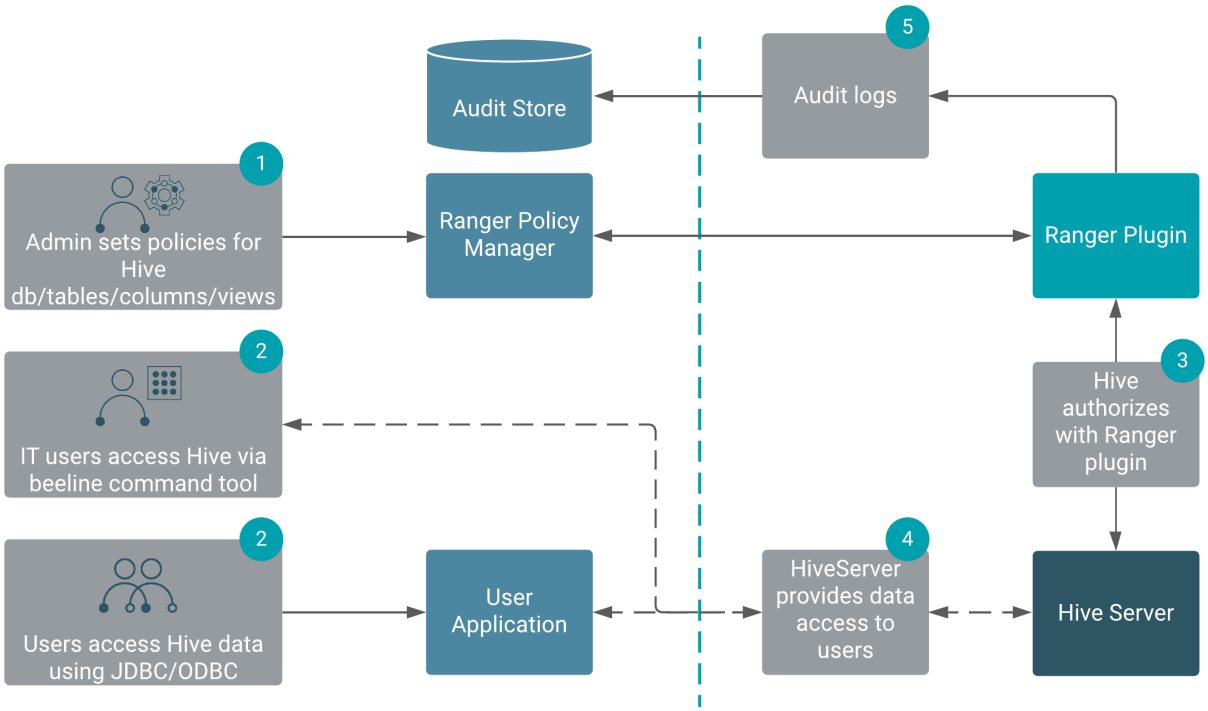

The following table compares authorization models:

Authorization model	Secure?	Fine-grained authorization (column, row level)	Privilege management using GRANT/REVOKE statements	Centralized management GUI
Apache Ranger	Secure	Yes	Yes	Yes
Hive default	Not secure. No restriction on which users can run GRANT statements	Yes	Yes	No

When you run grant/revoke commands and Apache Ranger is enabled, a Ranger policy is created/removed.

Transactional table access

As administrator, you must enable the Apache Ranger service to authorize users who want to work with transactional tables. These types of tables are the default, ACID-compliant tables in Hive 3 and later.

ACID tables reside by default in /warehouse/tablespace/managed/hive. Only the Hive service can own and interact with files in this directory. Ranger is the only available authorization mechanism that Cloudera recommends for ACID tables.

External table access

As administrator, you must set up Apache Ranger to allow users to access external tables.

External tables reside by default in /warehouse/tablespace/external on your object store. To specify some other location of the external table, you need to include the specification in the table creation statement as shown in the following example:

```
CREATE EXTERNAL TABLE my_external_table (a string, b string)
LOCATION '/users/andrena';
```

Hive assigns a default permission of 777 to the hive user, sets a umask to restrict subdirectories, and provides a default ACL to give Hive read and write access to all subdirectories. External tables must be secured using Ranger.

Token-based authentication for Cloudera Data Warehouse integrations

Using a token, you can sign on to use Hive and Impala in Cloudera Data Warehouse for a period of time instead of entering your single-sign on (SSO) credentials every time you need to run a query. This feature is in a technical preview state. Contact your account team for more information.

Cloudera Runtime 1.0.0

Securing Hue

Date published: 2021-12-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

User management in Hue **4**
Understanding Hue users and groups 4
Finding the list of Hue superusers 5
Creating a Hue user in 6
Creating a group in Hue 6
Managing Hue permissions 7

User authentication in Hue **7**
Authenticating Hue users with SAML 7
SAML properties 8

Securing sessions **9**

Specifying HTTP request methods **12**

Restricting supported ciphers for Hue **12**

Specifying domains or pages to which Hue can redirect users **12**

User management in Hue

Hue is a gateway to cluster services and both have completely separate permissions. Being a Hue superuser does not grant access to HDFS, Hive, and so on.

Users who log on to the Hue UI must have permission to use Hue and to each service accessible within Hue.

A common configuration is for “Hue users” to be authenticated with an LDAP server and “ users” with Kerberos. These users can differ. For example, services do not authenticate each user who logs on to Hue. Rather, they authenticate “Hue” and trust that Hue has authenticated “its” users.

Once Hue is authenticated by a service such as Hive, Hue impersonates the user requesting use of that service. For example, to create a Hive table. The service uses Apache Ranger to ensure the group to which that user belongs is authorized for that action.

Hue user permissions are at the application level only. For example, a Hue superuser can filter Hue user access to a service but cannot authorize the use of its features. Again, Ranger does that.

Understanding Hue users and groups

There are two types of users in Hue - superusers and general users referred to as users, each with specific privileges. These users can be a part of certain groups. Groups enable you to control which Hue applications and features your users can view and access when they log into Hue.

On a non-secure cluster, the first user logging into Hue after the initial installation becomes the first superuser. Superusers have the permissions to perform the following administrative functions:

- Add and delete users
- Add and delete groups
- Assign permissions to groups
- Change a user into a superuser
- Import users and groups from an LDAP server

If a user is part of the superuser LDAP group in Hue, then that user is also a part of the group of superusers in Hue.

Users can only change their name, e-mail address, and password. They can log in to Hue and run Hue applications, subject to the permissions provided by the Hue groups to which they belong. This is different from how perceives the Hue application when you submit a Hive or an Impala query from the Hue user interface (UI). Hue is a server between the users and the services. Hue is considered as a single ‘hue’ user by the other services in the cluster.

For example, when a user ‘bob’ submits a query from Hue, Hue also sends the username of this user to the corresponding service in . The HIVE_ON_TEZ service in considers ‘bob’ as the owner of the query and not ‘hue’. This is illustrated in the following graphic:

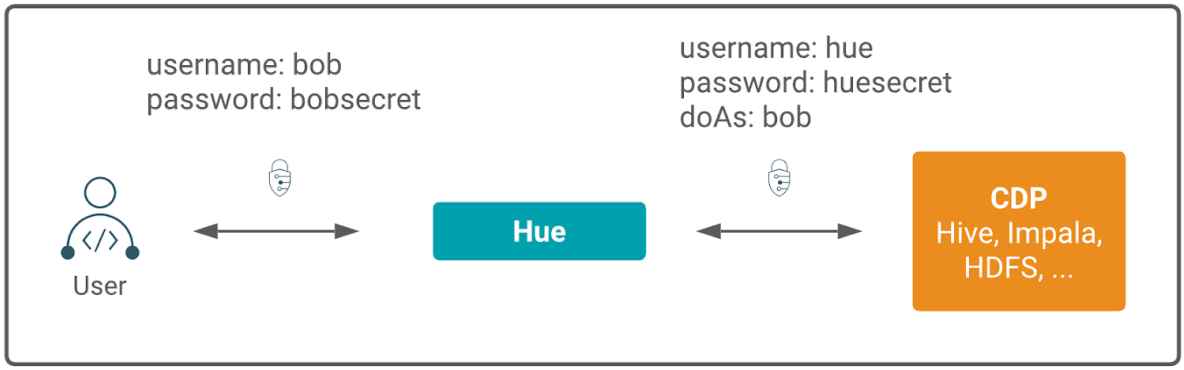

Hue is a gateway to cluster services and both have separate permissions. A Hue superuser is not granted access to HDFS, Hive, and other cluster services. Apache Ranger governs access to the cluster services.

Note: Groups in Hue are different from groups in Ranger.

Hue user permissions are at the application level only. For example, a Hue superuser can filter Hue user access to a service but cannot authorize the use of its features. Users who log on to the Hue UI must have permission to use Hue and to each service accessible within Hue.

Finding the list of Hue superusers

You can fetch the list of superusers by using the Hue shell with Python code or by running a SQL query on the auth_user table.

Using the Hue shell and Python code to find Hue superusers

1. Connecting to Hue shell by running the following command:

```
/opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue shell --cm-managed
```

2. Enter the Python code as follows:

```
from django.contrib.auth.models import User
print "%s" % User.objects.filter(is_superuser = True)
```

Sample output:

```
<QuerySet [<User: admin>]>
```

Running a SQL query on the auth_user table to find Hue superusers

1. Connect to Hue database shell by running the following command:

```
/opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue dbshell --cm-managed
```

2. Run the following SQL query:

```
select username, is_superuser from auth_user where is_superuser=1;
```

Sample output:

```
----------------------+
username is_superuser
----------------------+

admin 1
----------------------+
1 row in set (0.00 sec)
```

Creating a Hue user in

You can create new Hue users and superusers from the Hue web UI and assign them to groups so that they can view and access Hue as per the permissions granted to them.

Procedure

1. Log in to Hue as a superuser.
2. From the left assist panel, point your cursor to the user profile icon and click Administer Users.
3. On the **User Admin** page, click Add/Sync LDAP user.
4. On the **Hue User - Add/Sync LDAP user** page, specify the username and select the Create home directory option if you want to create a separate Hue home directory for this user.
5. Go to the Step 2. Profile and Groups tab.
6. Enter the first and last name of the user, their email address, and add them to the required groups.

   A user can be a part of more than one group.
7. Go to the Step 3. Advanced tab.
8. Ensure that the user is active by selecting the Active option.
9. If you want to make this user a superuser, then select the Superuser status option.
10. Click Add user.
    The new user is displayed on the **Users** page.

Creating a group in Hue

By creating groups, you can club certain permissions that you want to assign to specific users in your organization.

Procedure

1. Sign in to the Hue UI as a superuser.
2. From the left assist panel, point your cursor to the user profile icon and click Administer Users.

   The **User Admin** page is displayed.
3. Go to the Groups tab.

   The **Groups** page displays the list of existing groups, if any.
4. Click Add group.

   If you are using LDAP, then you can directly add LDAP groups by clicking Add/Sync LDAP group.

5. To add or synchronize an LDAP group:
   a) Specify the name of your LDAP group in the Name field.
   b) If you are using LDAP distinguished names, then you can select the Distinguished name option.
   c) Select Import new members to automatically import new members to Hue when you add them to your LDAP group.
   d) Select the Create home directories option if you want to create home directories for your users.
   e) Select the Import new members from all subgroups option to import new members from your subgroups.
   f) Click Add/Sync group.

      The newly added group is displayed on the **Groups** page along with the list of members and permissions associated with it.

Managing Hue permissions

Permissions for Hue applications are granted to groups, with users gaining permissions based on their group membership. Group permissions define the Hue applications visible to group members when they log in to Hue and the application features available to them. There is a fixed set of Hue permissions. You cannot add or modify permissions. However, you can apply permission to group(s).

Procedure

1. Sign in to the Hue UI as a superuser.
2. From the left assist panel, point your cursor to the user profile icon and click Administer Users.

   The **User Admin** page is displayed.
3. From the **User Admin** page, go to the Permissions tab.

   The **Permissions** page displays the list of all the available permissions.
4. Click a permission that you want to assign to a group(s).

   The **Edit [permission name]** page is displayed.
5. Select the group(s) on which you want to apply the permission and click Update permission.

   The “Permission information updated successfully” message is displayed.

User authentication in Hue

supports authenticating users to Hue using SAML.

After Hue is authenticated by a service such as Hive, Hue impersonates the user requesting the use of that service, for example, to create a Hive table. In this case, the Hive service uses Apache Ranger to ensure that the group to which the user belonged is authorized for that action (to create a Hive table).

Note: By default, the Hue session uses a secure cookie protocol.

Authenticating Hue users with SAML

Hue supports SAML (Security Assertion Markup Language) for Single Sign-on (SSO) authentication.

The SAML 2.0 Web Browser SSO profile has three components:

- User Agent - Browser that represents you, the user, seeking resources.
- Service Provider (SP) - Service (Hue) that sends authentication requests to SAML.
- Identity Provider (IdP) - SAML service that authenticates users.

When a user requests access to an application, the Service Provider (Hue) sends an authentication request from the User Agent (browser) to the identity provider. The identity provider authenticates the user, sends a response, and redirects the browser back to Hue as shown in the following diagram:

Figure 1: SAML SSO protocol flow in a web browser

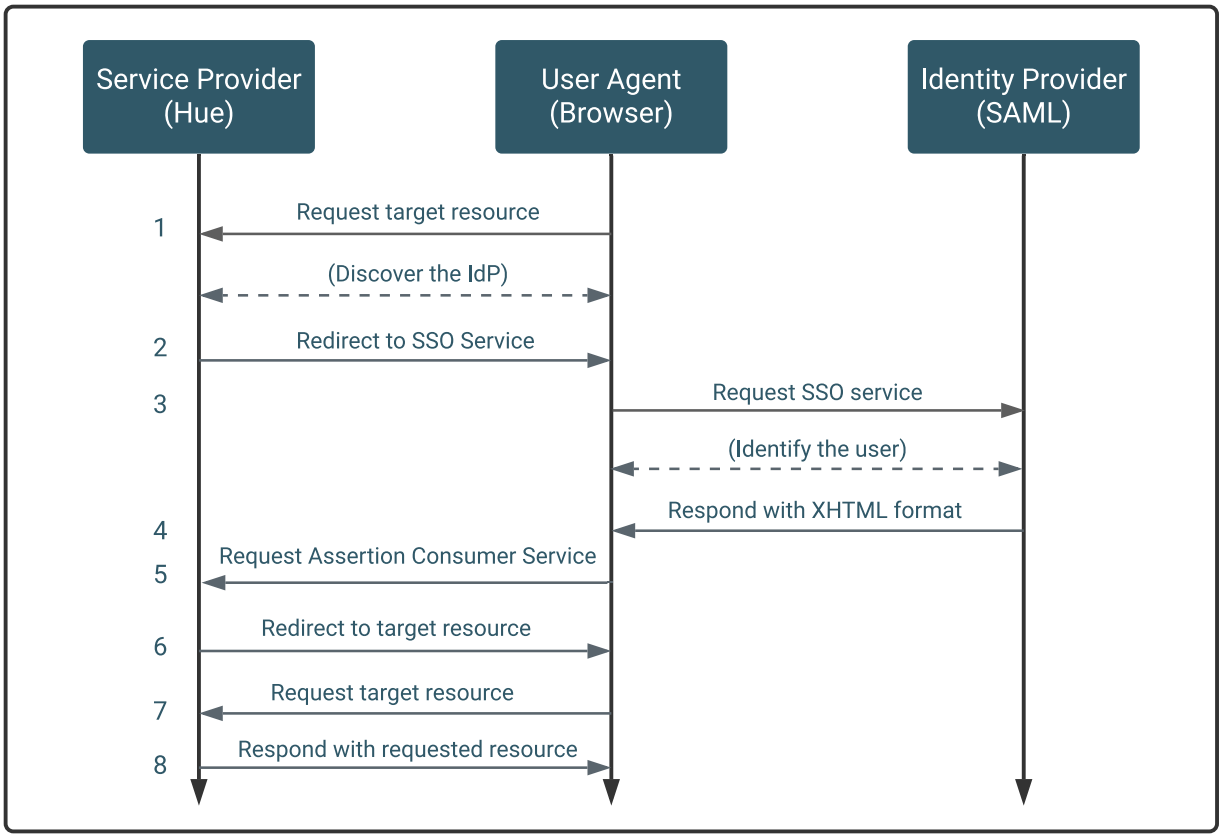

The Service Provider (Hue) and the identity provider use a metadata file to confirm each other's identity. Hue stores metadata from the SAML server, and the identity provider stores metadata from the Hue server.

In , SSO with SAML is automatically configured. You need not configure anything manually.

SAML properties

In , SSO with SAML is automatically configured. However, if you need to configure a certain parameter, you can set the properties in the hue-safety-valve field.

Table 1: Table of SAML parameters

SAML parameter	Description
authn_requests_signed	Boolean, that when True, signs Hue-initiated authentication requests with X.509 certificate.
backend	Hard-coded value set to SAML backend library packaged with Hue (libsaml.backend.SAML2Backend).
base_url	URL that SAML Identity Provider uses for responses. Typically used in Load balanced Hue environments.
cert_file	Path to X.509 certificate sent with encrypted metadata. File format must be .PEM.
create_users_on_login	Boolean, that when True, creates users from OpenId, upon successful login.
entity_id	Service provider ID. Can also accept pattern where '<base_url>' is replaced with server URL base.

SAML parameter	Description
key_file	Path to private key used to encrypt metadata. File format must be .PEM.
key_file_password	Password used to decrypt the X.509 certificate in memory.
logout_enabled	Boolean, that when True, enables single logout.
logout_requests_signed	Boolean, that when True, signs Hue-initiated logout requests with an X.509 certificate.
metadata_file	Path to readable metadata XML file copied from Identity Provider.
name_id_format	Format of NameID that Hue requests from SAML server.
optional_attributes	Comma-separated list of optional attributes that Hue requests from Identity Provider.
required_attributes	Comma-separated list of required attributes that Hue requests from Identity Provider. For example, uid and email.
redirect_whitelist	Fully qualified domain name of SAML server: "^\/.*$,^https:\/\/<SAML_server_FQDN>\/.*$".
user_attribute_mapping	Map of Identity Provider attributes to Hue django user attributes. For example, {'uid':'username', 'email':'email'}.
username_source	Declares source of username as nameid or attributes.
want_response_signed	A boolean parameter, when set to True, requires SAML response wrapper returned by an IdP to be digitally signed by the IdP. The default value is False.
want_assertions_signed	A boolean parameter, when set to True, requires SAML assertions returned by an IdP to be digitally signed by the IdP. The default value is False.
xmlsec_binary	Path to xmlsec_binary that signs, verifies, encrypts/decrypts SAML requests and assertions. Must be executable by user, hue.

Securing sessions

When a Hue session expires, the screen blurs and the user is automatically logged out of the Hue web interface. Logging back on returns the user to the same location in the application.

Session timeout

User sessions are controlled with the ttl (time-to-live) property, which is set in Virtual Warehouses Edit CONFIGURATIONS Hue Configuration files hue-safety-valve property as follows:

```
[desktop]
     [[session]]
     ttl=[***NUMBER-OF-SECONDS***]
```

The default setting for ttl is 1,209,600 seconds, which equals two weeks. The ttl property determines the length of time that the cookie with the user's session ID lives before expiring. After the ttl setting is reached, the user's session expires whether it is active or not.

Idle session timeout

Idle sessions are controlled with the idle_session_timeout property, which is set in Virtual Warehouses Edit CONFIGURATIONS Hue Configuration files hue-safety-valve property as follows:

```
[desktop]
     [[auth]]
     idle_session_timeout=[***NUMBER-OF-SECONDS***]
```

Sessions expire that are idle for the number of seconds set for this property. For example, if you set idle_session_timeout=900, sessions expire after being idle for 15 minutes. You can disable the property by setting it to a negative value, like idle-session_timeout=-1.

Secure session login

Session login properties are set under the [desktop] [[auth]] section in Virtual Warehouses Edit CONFIGURATIONS Hue Configuration files hue-safety-valve property as follows:

```
[desktop]
     [[auth]]
     [***SET-SESSION-LOGIN-PARAMETERS-HERE***]
```

Note: These configuration settings are based on django-axes 1.5.0.

Use the following properties to configure session login behavior:

change_default_password	Valid values: true | false
	If this property is set to true, users must change their passwords on first login attempt.
	Example:
	`[desktop]`
	`[[auth]]`
	`change_default_password=true`
	To use this property, you must enable the AllowFirstUserDjangoBackend in Hue. For example:
	`[desktop]`
	`[[auth]]`
	`backend=desktop.auth.backend.AllowFirstUserDjangoBackend`

expires_after	Use this property to configure the number of seconds after logout that user accounts are disabled. For example, user accounts are disabled 900 seconds or 15 minutes after logout with the following configuration:
	`[desktop]`
	`[[auth]]`
	`expires_after=900`
	If you set this property to a negative value, user sessions never expire. For example, expires_after=-1.

expire_superusers	Use to expire superuser accounts after the specified number of seconds after logout. For example, expire_superusers=900 causes superuser accounts to expire 15 minutes after logging out.
login_cooloff_time	Sets the number of seconds after which failed logins are forgotten. For example, if you set login_cooloff_time=900, a failed login attempt is forgotten after 15 minutes.
login_failure_limit	Sets the number of login attempts allowed before a failed login record is created. For example, if you set login_failure_limit=3, a failed login record is created after 3 login attempts.
login_lock_out_at_failure	Valid values: true | false If set to true: • The IP address that is attempting to log in is locked out after exceeding the limit set for login_failure_limit. • If login_lock_out_by_combination_user_and_ip is also set to true, both the IP address and the user are locked out after exceeding the limit set for login_failure_limit. • If login_lock_out_use_user_agent is also set to true, both the IP address and the agent application (such as a browser) are locked out after exceeding the limit set for login_failure_limit.

login_lock_out_by_combination_user_and_ip	Valid values: true | false If set to true, both the IP address and the user are locked out after exceeding the limit set for login_failure_limit.
login_lock_out_use_user_agent	Valid values: true | false If set to true, the agent application (such as a browser) is locked out after exceeding the limit set for login_failure_limit.

Secure session cookies

Session cookie properties are set under the [desktop] [[session]] section in Virtual Warehouses Edit CONFIGURATIONS Hue Configuration files hue-safety-valve property as follows:

```
[desktop]
     [[session]]
     [***SET-SESSION-COOKIE-PROPERTIES-HERE***]
```

Use the following properties to configure session cookie behavior:

secure	Valid values: true | false
	If this property is set to true, the user session ID is secured.
	Important: To use this property, HTTPS must be enabled.
	Example:
	`[desktop]`
	`[[session]]`
	`secure=true`
	By default this property is set to false.

http_only	Valid values: true | false
	If this property is set to true, the cookie with the user session ID uses the HTTP only flag.
	Example:
	`[desktop]`
	`[[session]]`
	`http_only=true`
	Important: If the HttpOnly flag is included in the HTTP response header, the cookie cannot be accessed through a client side script.
	By default this property is set to true.

expire_at_browser_close	Valid values: true | false
	If this property is set to true, only session-length cookies are used. Users are automatically logged out when the browser window is closed.
	Example:
	`[desktop]`
	`[[session]]`
	`expire_at_browser_close=true`
	By default this property is set to false.

Specifying HTTP request methods

You can specify the HTTP request methods that the Hue server responds to.

Use the http_allowed_methods property under the [desktop] section in Virtual Warehouses Edit CONFIGURATIONS Hue Configuration files hue-safety-valve property.

By default, the http_allowed_methods property is set to options, get, head, post, put, delete, connect.

Restricting supported ciphers for Hue

You can configure the list of ciphers that Hue supports with HTTPS.

Use the ssl_cipher_list property under the [desktop] section in Virtual Warehouses Edit CONFIGURATIONS Hue Configuration files hue-safety-valve property:

```
[desktop]
ssl_cipher_list=[***LIST-OF-ACCEPTED-CIPHERS***]
```

By default, the ssl_cipher_list property is set to !aNULL:!eNULL:!LOW:!EXPORT:!SSLv2. Specify ciphers using the cipher list format described at OpenSSL Cryptograpy and SSL/TLS Toolkit Manpages by selecting the SSL version, and then going to Commands ciphers .

Specifying domains or pages to which Hue can redirect users

You can restrict the domains or pages to which Hue can redirect users.

Use the redirect_whitelist property under the [desktop] section in Virtual Warehouses Edit CONFIGURATIONS Hue Configuration files hue-safety-valve property:

```
[desktop]
redirect_whitelist=[***REDIRECT-URL***]
```

Specify the redirect_whitelist value with a comma-separated list of regular expressions that match the redirect URL. For example, to restrict redirects to your local domain and fully-qualified domain name (FQDN), use the following value:

```
redirect_whitelist=^\/.*$,^http:\/\/www.mydomain.com\/.*$
```

Cloudera Runtime 7.2.11

Starting Apache Hive

Date published: 2019-08-21
Date modified: 2021-09-08

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Running a query in Cloudera Data Warehouse.....4

Running a query in Cloudera Data Warehouse

You simply open Hue and submit your query. You do not need to manually start beeline or any other shell.

About this task

As a DWUser, you open Hue from a Virtual Warehouse that you set up, and run the query. The SQL engine reads from and writes to the same metastore, regardless of the type Virtual Warehouse.

Before you begin

- Required role: DW User
- You obtained permission to run SQL queries from the EnvironmentAdmin, who added you to a Hadoop SQL policy.

Procedure

1. Log into the Cloudera Data Warehouse service as DWUser.
2. Go to the **Virtual Warehouses** tab, locate the Virtual Warehouse using which you want to run queries, and click HUE.

   The Hue query editor opens in a new browser tab.
3. To run a query:
   a) Click a database to view the tables it contains.

      When you click a database, it sets it as the target of your query in the main query editor panel.

   b) Type a query in the editor panel and click ▶ to run the query.

1.0.0

Tuning Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-05

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Data Cache for Remote Reads..... 4

Data Cache for Remote Reads

In Cloudera Data Warehouse, frequently accessed data is cached in a storage layer on SSD so that it can be quickly retrieved for subsequent queries.

The ability to quickly retrieve data for subsequent queries boosts performance. Each executor can have up to 200 GB of cache. For example, a Medium-sized Virtual Warehouse can keep 200 * 20 = 4 TB of data in its cache. For columnar formats, such as ORC, data in the cache is decompressed, but not decoded. If the expected size of the hot dataset is 6 TB, which requires about 30 executors, you can over-provision by choosing a Large-sized warehouse to ensure full cache coverage. A case can also be made to under-provision by choosing a Medium-sized warehouse to reduce costs at the expense of having a lower cache hit rate. To offset this, keep in mind that columnar formats allow optimizations such as column projection and predicate push-down, which can significantly reduce cache requirements. In these cases, under-provisioning the cache might have no negative performance effect.

Cloudera Runtime 1.0.0

Tuning Hue

Date published: 2022-04-27
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring the Hue Query Processor scan frequency..... 4

Configuring the Hue Query Processor scan frequency

You can configure the frequency at which the Hue Query Processor fetches the data from a Hadoop Compatible File System by setting the time in milliseconds (ms) in the “hue.query-processor.event-pipeline.folder-scan-delay-millis” property in the Database Catalog configurations.

About this task

By default, the value of the “hue.query-processor.event-pipeline.folder-scan-delay-millis” property is 5 * 60 * 1000 ms (5 minutes). For a faster data refresh rate, you can set this value to less than 5 * 60 * 1000 ms (5 minutes). But this can result in higher storage read costs. However, if you want stronger performance, and a higher refresh rate is not a priority, then you can set it to a higher value to reduce storage read costs.

Procedure

1. Sign in to the service.
2. Go to Database Catalog Edit CONFIGURATIONS Hue query processor and select hue-query-processor.json from the Configuration files drop-down menu.
3. Change the value of the “hue.query-processor.event-pipeline.folder-scan-delay-millis” property under the “dasC onf” section.
4. Click Apply Changes.

Cloudera Runtime 1.0.0

Using Apache Hive

Date published: 2021-12-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Hive 3 tables.....5
Refer to a table using dot notation.....6
Creating a CRUD transactional table.....6
Creating an insert-only transactional table.....7
Converting a managed non-transactional table to external.....8
External tables based on a non-default schema.....8
Using your schema in MariaDB.....8
Using your schema in MS SQL.....9
Using your schema in Oracle.....10
Using your schema in PostgreSQL.....11
Using constraints.....12
Determining the table type.....14

Apache Hive 3 ACID transactions.....14

Apache Hive query basics.....17
Querying the information_schema database.....17
Inserting data into a table.....19
Updating data in a table.....20
Merging data in tables.....20
Deleting data from a table.....20
Using a subquery.....21
Subquery restrictions.....21
Use wildcards with SHOW DATABASES.....21
Aggregating and grouping data.....22
Querying correlated data.....22
Using common table expressions.....23
Use a CTE in a query.....23
Comparing tables using ANY/SOME/ALL.....24
Escaping an invalid identifier.....24
CHAR data type support.....25
ORC vs Parquet formats.....25

Creating a default directory for managed tables.....26

Generating surrogate keys.....26

Partitions and performance.....28
Repairing partitions manually using MSCK repair.....29

Query scheduling.....30
Enabling scheduled queries.....30
Periodically rebuilding a materialized view.....31

Getting scheduled query information and monitor the query....32

Materialized views....33

Creating and using a materialized view....34
Creating the tables and view....34
Using optimizations from a subquery....35
Dropping a materialized view....36
Showing materialized views....36
Describing a materialized view....37
Managing query rewrites....39
Purposely using a stale materialized view....39
Creating and using a partitioned materialized view....40

Cloudera Data Warehouse....42

Setting up a Cloudera Data Warehouse client....43
Creating a function....44
Using the cursor to return record sets....45
Stored procedure examples....46

Apache Hive 3 tables

Table type definitions and a diagram of the relationship of table types to ACID properties clarifies Hive tables. The location of a table depends on the table type. You might choose a table type based on its supported storage format.

You can create ACID (atomic, consistent, isolated, and durable) tables for unlimited transactions or for insert-only transactions. These tables are Hive managed tables. Alternatively, you can create an external table for non-transactional use. Because Hive control of the external table is weak, the table is not ACID compliant.

The following diagram depicts the Hive table types.

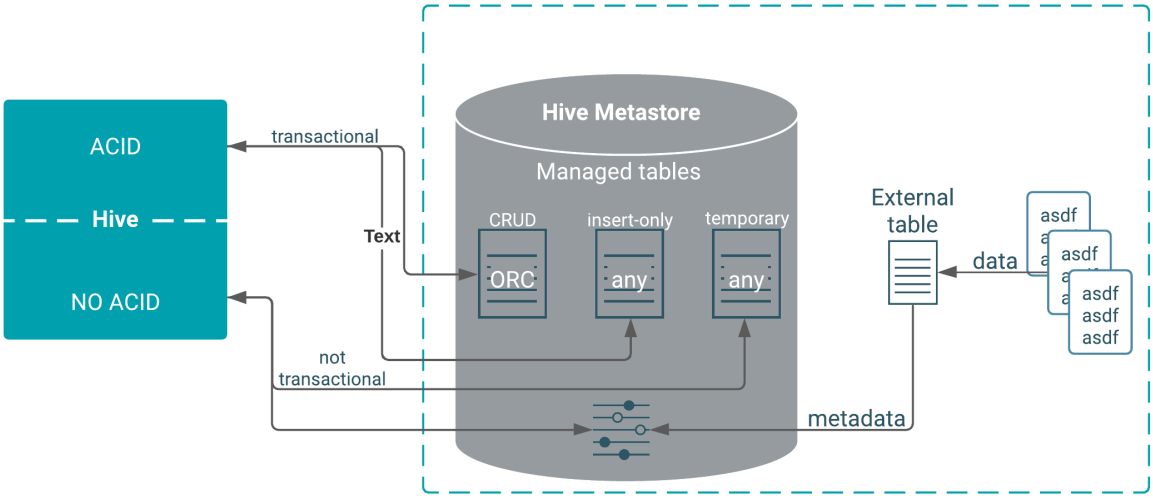

The following matrix includes the types of tables you can create using Hive, whether or not ACID properties are supported, required storage format, and key SQL operations.

Table Type	ACID	File Format	INSERT	UPDATE/DELETE
Managed: CRUD transactional	Yes	ORC	Yes	Yes
Managed: Insert-only transactional	Yes	Any	Yes	No
Managed: Temporary	No	Any	Yes	No
External	No	Any	Yes	No

Although you cannot use the SQL UPDATE or DELETE statements to delete data in some types of tables, you can use DROP PARTITION on any table type to delete the data.

Table storage formats

The data in CRUD tables must be in ORC format. Implementing a storage handler that supports AcidInputFormat and AcidOutputFormat is equivalent to specifying ORC storage.

Insert-only tables support all file formats.

The managed table storage type is Optimized Row Column (ORC) by default. If you accept the default by not specifying any storage during table creation, or if you specify ORC storage, you get an ACID table with insert, update, and delete (CRUD) capabilities. If you specify any other storage type, such as text, CSV, AVRO, or JSON, you get an insert-only ACID table. You cannot update or delete columns in the insert-only table.

Transactional tables

Transactional tables are ACID tables that reside in the Hive warehouse. To achieve ACID compliance, Hive has to manage the table, including access to the table data. Only through Hive can you access and change the data in managed tables. Because Hive has full control of managed tables, Hive can optimize these tables extensively.

Hive is designed to support a relatively low rate of transactions, as opposed to serving as an online analytical processing (OLAP) system. You can use the SHOW TRANSACTIONS command to list open and aborted transactions.

Transactional tables in Hive 3 are on a par with non-ACID tables. No bucketing or sorting is required in Hive 3 transactional tables. Bucketing does not affect performance. These tables are compatible with native cloud storage.

Hive supports one statement per transaction, which can include any number of rows, partitions, or tables.

External tables

External table data is not owned or controlled by Hive. You typically use an external table when you want to access data directly at the file level, using a tool other than Hive.

Hive 3 does not support the following capabilities for external tables:

- Query cache
- Materialized views, except in a limited way
- Automatic runtime filtering
- File merging after insert
- ARCHIVE, UNARCHIVE, TRUNCATE, MERGE, and CONCATENATE. These statements only work for Hive Managed tables.

When you run DROP TABLE on an external table, by default Hive drops only the metadata (schema). If you want the DROP TABLE command to also remove the actual data in the external table, as DROP TABLE does on a managed table, you need to set the external.table.purge property to true as described later.

Refer to a table using dot notation

Hive 3.1 changes to table references using dot notation might require changes to your Hive scripts.

About this task

Hive 3.1 in Cloudera includes SQL compatibility (Hive-16907), which rejects `db.table` in SQL queries. The dot (.) is not allowed in table names. To reference the database and table in a table name, enclosed both in backticks as follows:

```
`db`.`table`
```

Creating a CRUD transactional table

You create a CRUD transactional table having ACID (atomic, consistent, isolated, and durable) properties when you need a managed table that you can update, delete, and merge. You learn by example how to determine the table type.

About this task

In this task, you create a CRUD transactional table. You cannot sort this type of table. To create a CRUD transactional table, you must accept the default ORC format by not specifying any storage during table creation, or by specifying ORC storage explicitly.

Procedure

1. Start Hive.
   For example, start Hive from a JDBC client:

   ```
   beeline -u jdbc:hive2://myhiveserver.com:10000 -n hive -p
   ```

2. Enter your user name and password.
   The Hive 3 connection message, followed by the Hive prompt for entering SQL queries on the command line, appears.
3. Create a CRUD transactional table named T having two integer columns, a and b:

   ```
   CREATE TABLE T(a int, b int);
   ```

4. Confirm that you created a managed, ACID table.

   ```
   DESCRIBE FORMATTED T;
   ```

   The table type says MANAGED_TABLE and transactional = true.

Related Information

HMS storage

Creating an insert-only transactional table

You can create a transactional table using any storage format if you do not require update and delete capability. This type of table has ACID properties, is a managed table, and accepts insert operations only. The storage format of an insert-only table is not restricted to ORC.

About this task

In this task, you create an insert-only transactional table for storing text. In the CREATE TABLE statement, specifying a storage type other than ORC, such as text, CSV, AVRO, or JSON, results in an insert-only ACID table. You can explicitly specify insert-only in the table properties clause.

Procedure

1. Start Hive.
   For example, start Hive from a JDBC client:

   ```
   beeline -u jdbc:hive2://myhiveserver.com:10000 -n hive -p
   ```

2. Enter your user name and password.
   The Hive 3 connection message, followed by the Hive prompt for entering SQL queries on the command line, appears.
3. Create a insert-only transactional table named T2 having two integer columns, a and b:

   ```
   CREATE TABLE T2(a int, b int)
     STORED AS ORC
     TBLPROPERTIES ('transactional'='true',
     'transactional_properties'='insert_only');
   ```

   The 'transactional_properties'='insert_only' is required; otherwise, a CRUD table results. The STORED AS ORC clause is optional (default = ORC).
4. Create an insert-only transactional table for text data.

   ```
   CREATE TABLE T3(a int, b int)
   ```

```
STORED AS TEXTFILE;
```

The 'transactional_properties'='insert_only' is not required because the storage format is other than ORC.

Related Information

HMS storage

Converting a managed non-transactional table to external

You can easily convert a managed table, if it is not an ACID (transactional) table, to external using the ALTER TABLE statement. You might have a non-ACID, managed table after an upgrade from Hive 1 or 2.

About this task

The following pseudo-code changes a managed table, if it is not transactional, to external. The data and metadata is dropped when the table is dropped.

```
ALTER TABLE ... SET TBLPROPERTIES('EXTERNAL'='TRUE','external.table.purge'='
true')
```

Related Information

Before and After Upgrading Table Type Comparison

External tables based on a non-default schema

If you define a schema for external tables, you need to know how to create the table using the hive.sql.schema table property to prevent problems with client connections to the tables. Examples for widely-used databases show you how to create such tables.

The handling of schema differs from DBMS to DBMS. In this task, you follow examples to create external tables that the following databases can understand.

- MariaDB
- MS SQL
- Oracle
- PostgreSQL

Using your schema in MariaDB

You follow an example of how to create an external table in MariaDB using your own schema.

Before you begin

GRANT INSERT ANY TABLE TO bob;

GRANT INSERT ANY TABLE TO alice;

Procedure

Using MariaDB, create an external table based on a user-defined schema.

```
CREATE SCHEMA bob;
CREATE TABLE bob.country
(
    id   int,
    name varchar(20)
);
insert into bob.country
```

```
values (1, 'India');
insert into bob.country
values (2, 'Russia');
insert into bob.country
values (3, 'USA');

CREATE SCHEMA alice;
CREATE TABLE alice.country
(
    id   int,
    name varchar(20)
);
insert into alice.country
values (4, 'Italy');
insert into alice.country
values (5, 'Greece');
insert into alice.country
values (6, 'China');
insert into alice.country
values (7, 'Japan');
```

Using your schema in MS SQL

You follow an example of how to create an external table in MS SQL using your own schema.

Procedure

1. Using MS SQL, create an external table based on a user-defined schema.

```
CREATE DATABASE world;
USE world;

CREATE SCHEMA bob;
CREATE TABLE bob.country
(
    id   int,
    name varchar(20)
);

insert into bob.country
values (1, 'India');
insert into bob.country
values (2, 'Russia');
insert into bob.country
values (3, 'USA');

CREATE SCHEMA alice;
CREATE TABLE alice.country
(
    id   int,
    name varchar(20)
);

insert into alice.country
values (4, 'Italy');
insert into alice.country
values (5, 'Greece');
insert into alice.country
values (6, 'China');
insert into alice.country
values (7, 'Japan');
```

2. In MS SQL, create a user and associate them with a default schema.

```
CREATE LOGIN greg WITH PASSWORD = 'GregPass123!$';
CREATE USER greg FOR LOGIN greg WITH DEFAULT_SCHEMA=bob;
```

3. Allow the user to connect to the database and run queries. For example:

```
GRANT CONNECT, SELECT TO greg;
```

Using your schema in Oracle

You follow an example of how to create an external table in Oracle using your own schema.

About this task

In Oracle, dividing the tables into different namespaces/schemas is achieved through different users. The CREATE SCHEMA statement exists in Oracle, but has different semantics from those defined by SQL Standard and those adopted in other DBMS.

To create "local" users in Oracle you need to be connected to the Pluggable Database (PDB), not to the Container Database (CDB). The following example was tested in Oracle XE edition, using only PDB -- XEPDB1.

Procedure

1. Using Oracle XE edition, connect to the PDB.

```
ALTER SESSION SET CONTAINER = XEPDB1;
```

2. Create the bob schema/user and give appropriate connections to be able to connect to the database.

```
CREATE USER bob IDENTIFIED BY bobpass;
ALTER USER bob QUOTA UNLIMITED ON users;
GRANT CREATE SESSION TO bob;

CREATE USER bob IDENTIFIED BY bobpass;
ALTER USER bob QUOTA UNLIMITED ON users;
GRANT CREATE SESSION TO bob;
```

3. Create the alice schema/user, give appropriate connections to be able to connect to the database, and create an external table.

```
CREATE USER alice IDENTIFIED BY alicepass;
ALTER USER alice QUOTA UNLIMITED ON users;

GRANT CREATE SESSION TO alice;

CREATE TABLE alice.country
(
    id   int,
    name varchar(20)
);

insert into alice.country
values (4, 'Italy');
insert into alice.country
values (5, 'Greece');
insert into alice.country
values (6, 'China');
insert into alice.country
values (7, 'Japan');
```

4. Grant the SELECT ANY privilege to client users.

   Without the SELECT ANY privilege, a user cannot see the tables/views of another user. When a user connects to the database using a specific user and schema it is not possible to refer to tables in another user/schema -- namespace.

```
GRANT SELECT ANY TABLE TO bob;
GRANT SELECT ANY TABLE TO alice;
```

5. Allow the users to perform inserts on any table/view in the database, not only those present on their own schema.

```
GRANT INSERT ANY TABLE TO bob;
GRANT INSERT ANY TABLE TO alice;
```

Using your schema in PostgreSQL

You follow an example of how to create an external table in PostgreSQL using your own schema.

Procedure

1. Using Postgres, create external tables based on a user-defined schema.

```
CREATE SCHEMA bob;
CREATE TABLE bob.country
(
    id   int,
    name varchar(20)
);
insert into bob.country
values (1, 'India');
insert into bob.country
values (2, 'Russia');
insert into bob.country
values (3, 'USA');

CREATE SCHEMA alice;
CREATE TABLE alice.country
(
    id   int,
    name varchar(20)
);
insert into alice.country
values (4, 'Italy');
insert into alice.country
values (5, 'Greece');
insert into alice.country
values (6, 'China');
insert into alice.country
values (7, 'Japan');
```

2. Create a user and associate them with a default schema <=> search_path.

```
CREATE ROLE greg WITH LOGIN PASSWORD 'GregPass123!$';
ALTER ROLE greg SET search_path TO bob;
```

3. Grant the necessary permissions to be able to access the schema.

```
GRANT USAGE ON SCHEMA bob TO greg;
GRANT SELECT ON ALL TABLES IN SCHEMA bob TO greg;
```

Using constraints

You can use SQL constraints to enforce data integrity and improve performance. Using constraints, the optimizer can simplify queries. Constraints can make data predictable and easy to locate. Using constraints and supported modifiers, you can follow examples to constrain queries to unique or not null values, for example.

Hive enforces DEFAULT, NOT NULL and CHECK only, not PRIMARY KEY, FOREIGN KEY, and UNIQUE.

You can use the constraints listed below in your queries. Hive enforces DEFAULT, NOT NULL and CHECK only, not PRIMARY KEY, FOREIGN KEY, and UNIQUE. DEFAULT even if enforced, does not support complex types (array,map,struct). Constraint enforcement is limited to the metadata level. This limitation aids integration with third party tools and optimization of constraints declarations, such as materialized view rewriting.

CHECK

Limits the range of values you can place in a column.

DEFAULT

Ensures a value exists, which is useful in offloading data from a data warehouse.

PRIMARY KEY

Identifies each row in a table using a unique identifier.

FOREIGN KEY

Identifies a row in another table using a unique identifier.

UNIQUE KEY

Checks that values stored in a column are different.

NOT NULL

Ensures that a column cannot be set to NULL.

Supported modifiers

You can use the following optional modifiers:

ENABLE

Ensures that all incoming data conforms to the constraint.

DISABLE

Does not ensure that all incoming data conforms to the constraint.

VALIDATE

Checks that all existing data in the table conforms to the constraint.

NOVALIDATE

Does not check that all existing data in the table conforms to the constraint.

ENFORCED

Maps to ENABLE NOVALIDATE.

NOT ENFORCED

Maps to DISABLE NOVALIDATE.

RELY

Specifies abiding by a constraint; used by the optimizer to apply further optimizations.

NORELY

Specifies not abiding by a constraint.

You use modifiers as shown in the following syntax:

```
( ( ( (ENABLE | DISABLE) (VALIDATE | NOVALIDATE) ) | ( ENFORCED | NOT ENFORC
ED ) ) (RELY | NORELY) )
```

Note: For external tables, the RELY constraint is the only supported constraint.

Default modfiers

The following default modifiers are in place:

- The default modifier for ENABLE is NOVALIDATE RELY.
- The default modifier for DISABLE is NOVALIDATE NORELY.
- If you do not specify a modifier when you declare a constraint, the default is ENABLE NOVALIDATE RELY. The following constraints do not support ENABLE:
  - PRIMARY KEY
  - FOREIGN KEY
  - UNIQUE KEY

  To prevent an error, specify a modfier when using these constraints to override the default.

Constraints examples

The optimizer uses the constraint information to make smart decisions. The following examples show the use of constraints.

The following example shows how to create a table that declares the NOT NULL in-line constraint to constrain a column.

```
CREATE TABLE t(a TINYINT, b SMALLINT NOT NULL ENABLE, c INT);
```

The constrained column b accepts a SMALLINT value as shown in the first INSERT statement.

```
INSERT INTO t values(2,45,5667);
    ...
    ----------------------------------------------------------
    1 row affected ...
```

The constrained column b will not accept a NULL value.

```
INSERT INTO t values(2,NULL,5667);
    Error: Error running query: org.apache.hadoop.hive.ql.exec.errors.Dat
aConstraintViolationError: /
    Either CHECK or NOT NULL constraint violated! (state=,code=0)
```

The following examples shows how to declare the FOREIGN KEY constraint out of line. You can specify a constraint name, in this case fk, in an out-of-line constraint

```
CREATE TABLE Persons (
     ID INT NOT NULL,
     Name STRING NOT NULL,
     Age INT,
     Creator STRING DEFAULT CURRENT_USER(),
     CreateDate DATE DEFAULT CURRENT_DATE(),
     PRIMARY KEY (ID) DISABLE NOVALIDATE);

     CREATE TABLE BusinessUnit (
```

```
     ID INT NOT NULL,
     Head INT NOT NULL,
     Creator STRING DEFAULT CURRENT_USER(),
     CreateDate DATE DEFAULT CURRENT_DATE(),
     PRIMARY KEY (ID) DISABLE NOVALIDATE,
     CONSTRAINT fk FOREIGN KEY (Head) REFERENCES Persons(ID) DISABLE NOVA
LIDATE
     );
```

Determining the table type

You can determine the type of a Hive table, whether it has ACID properties, the storage format, such as ORC, and other information. Knowing the table type is important for a number of reasons, such as understanding how to store data in the table or to completely remove data from the cluster.

Procedure

1. In the Hive shell, get an extended description of the table.
   For example: DESCRIBE EXTENDED mydatabase.mytable;
2. Scroll to the bottom of the command output to see the table type.
   The following output says the table type is managed. transaction=true indicates that the table has ACID properties:

```
...
| Detailed Table Information  | Table(tableName:t2, dbName:mydatabase, o
wner:hdfs, createTime:1538152187, lastAccessTime:0, retention:0, sd:Stor
ageDescriptor(cols:[FieldSchema(name:a, type:int, comment:null), FieldSc
hema(name:b, type:int, comment:null)], ...
```

Related Information

HMS storage

Apache Hive 3 ACID transactions

Hive 3 achieves atomicity and isolation of operations on transactional tables by using techniques in write, read, insert, create, delete, and update operations that involve delta files. You can obtain query status information from these files and use the files to troubleshoot query problems.

Write and read operations

Hive 3 write and read operations improve the ACID qualities and performance of transactional tables. Transactional tables perform as well as other tables. Hive supports all TPC Benchmark DS (TPC-DS) queries.

Hive 3 and later extends atomic operations from simple writes and inserts to support the following operations:

- Writing to multiple partitions
- Using multiple insert clauses in a single SELECT statement

A single statement can write to multiple partitions or multiple tables. If the operation fails, partial writes or inserts are not visible to users. Operations remain fast even if data changes often, such as one percent per hour. Hive 3 and later does not overwrite the entire partition to perform update or delete operations.

Hive compacts ACID transaction files automatically without impacting concurrent queries. Automatic compaction improves query performance and the metadata footprint when you query many small, partitioned files.

Read semantics consist of snapshot isolation. Hive logically locks in the state of the warehouse when a read operation starts. A read operation is not affected by changes that occur during the operation.

Atomicity and isolation in insert-only tables

When an insert-only transaction begins, the transaction manager gets a transaction ID. For every write, the transaction manager allocates a write ID. This ID determines a path to which data is actually written. The following code shows an example of a statement that creates insert-only transactional table:

```
CREATE TABLE tm (a int, b int) TBLPROPERTIES
('transactional'='true', 'transactional_properties'='insert_only')
```

Assume that three insert operations occur, and the second one fails:

```
INSERT INTO tm VALUES(1,1);
INSERT INTO tm VALUES(2,2); // Fails
INSERT INTO tm VALUES(3,3);
```

For every write operation, Hive creates a delta directory to which the transaction manager writes data files. Hive writes all data to delta files, designated by write IDs, and mapped to a transaction ID that represents an atomic operation. If a failure occurs, the transaction is marked aborted, but it is atomic:

```
tm
___ delta_0000001_0000001_0000
### 000000_0
___ delta_0000002_0000002_0000       //Fails
### 000000_0
___ delta_0000003_0000003_0000
### 000000_0
```

During the read process, the transaction manager maintains the state of every transaction. When the reader starts, it asks for the snapshot information, represented by a high watermark. The watermark identifies the highest transaction ID in the system followed by a list of exceptions that represent transactions that are still running or are aborted.

The reader looks at deltas and filters out, or skips, any IDs of transactions that are aborted or still running. The reader uses this technique with any number of partitions or tables that participate in the transaction to achieve atomicity and isolation of operations on transactional tables.

Atomicity and isolation in CRUD tables

You create a full CRUD (create, retrieve, update, delete) transactional table using the following SQL statement:

```
CREATE TABLE acidtbl (a INT, b STRING);
```

Running SHOW CREATE TABLE acidtbl provides information about the defaults: transactional (ACID) and the ORC data storage format:

```
        +----------------------------------------------------+
        |                   createtab_stmt                   |
        +----------------------------------------------------+
        | CREATE TABLE `acidtbl`(                            |
        |   `a` int,                                         |
        |   `b` string)                                      |
        | ROW FORMAT SERDE                                   |
        |   'org.apache.hadoop.hive.ql.io.orc.OrcSerde'      |
        | STORED AS INPUTFORMAT                              |
        |   'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'  |
        | OUTPUTFORMAT                                       |
        |   'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat' |
        | LOCATION                                           |
        |   's3://myserver.com:8020/warehouse/tablespace/managed/hive/acidtb
l' |
        | TBLPROPERTIES (                                    |
```

```
|   'bucketing_version'='2',                          |
|   'transactional'='true',                           |
|   'transactional_properties'='default',             |
|   'transient_lastDdlTime'='1555090610')             |
+-----------------------------------------------------+
```

Tables that support updates and deletions require a slightly different technique to achieve atomicity and isolation. Hive runs in append-only mode, which means Hive does not perform in-place updates or deletions. Isolation of readers and writers cannot occur in the presence of in-place updates or deletions. In this situation, a lock manager or some other mechanism, is required for isolation. These mechanisms create a problem for long-running queries.

Instead of in-place updates, Hive decorates every row with a row ID. The row ID is a struct that consists of the following information:

- The write ID that maps to the transaction that created the row
- The bucket ID, a bit-backed integer with several bits of information, of the physical writer that created the row
- The row ID, which numbers rows as they were written to a data file

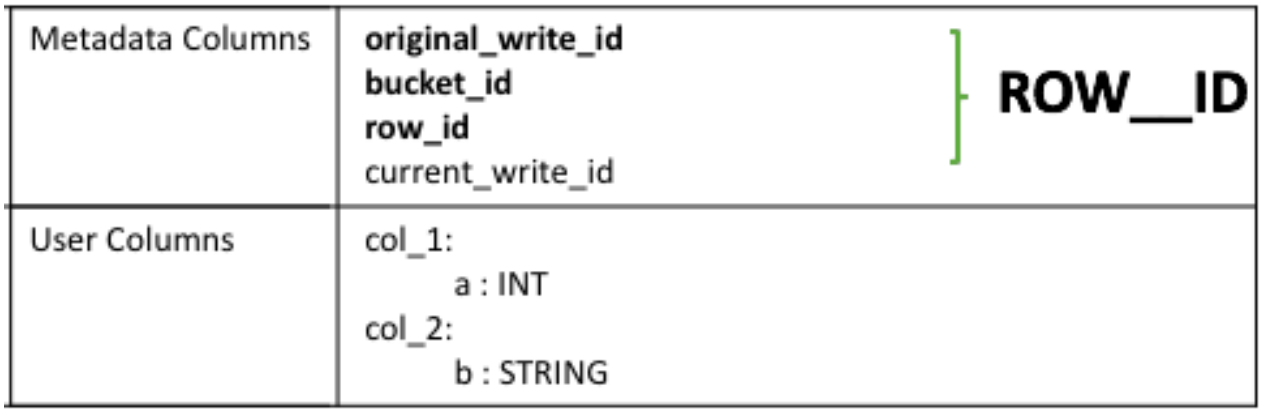

Instead of in-place deletions, Hive appends changes to the table when a deletion occurs. The deleted data becomes unavailable and the compaction process takes care of the garbage collection later.

Create operation

The following example inserts several rows of data into a full CRUD transactional table, creates a delta file, and adds row IDs to a data file.

```
INSERT INTO acidtbl (a,b) VALUES (100, "oranges"), (200, "apples"), (300, "b
ananas");
```

This operation generates a directory and file, delta_00001_00001/bucket_0000, that have the following data:

ROW_ID	a	b
{1,0,0}	100	"oranges"
{1,0.1}	200	"apples"
{1,0,2}	300	"bananas"

Delete operation

A delete statement that matches a single row also creates a delta file, called the delete-delta. The file stores a set of row IDs for the rows that match your query. At read time, the reader looks at this information. When it finds a delete

event that matches a row, it skips the row and that row is not included in the operator pipeline. The following example deletes data from a transactional table:

```
DELETE FROM acidTbl where a = 200;
```

This operation generates a directory and file, delete_delta_00002_00002/bucket_0000 that have the following data:

ROW_ID	a	b
{1,0,1}	null	null

Update operation

An update combines the deletion and insertion of new data. The following example updates a transactional table:

```
UPDATE acidTbl SET b = "pears" where a = 300;
```

One delta file contains the delete event, and the other, the insert event:

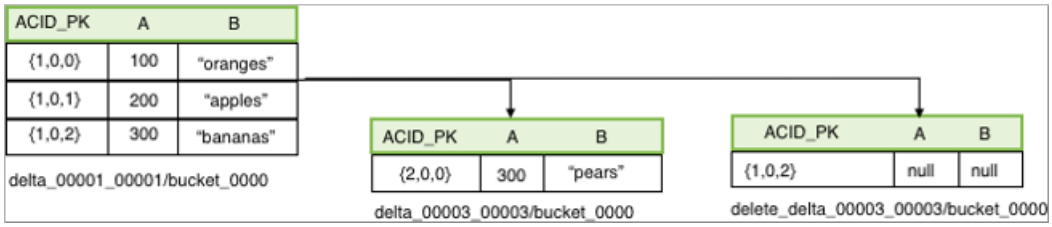

The reader, which requires the AcidInputFormat, applies all the insert events and encapsulates all the logic to handle delete events. A read operation first gets snapshot information from the transaction manager based on which it selects files that are relevant to that read operation. Next, the process splits each data file into the number of pieces that each process has to work on. Relevant delete events are localized to each processing task. Delete events are stored in a sorted ORC file. The compressed, stored data is minimal, which is a significant advantage of Hive 3. You no longer need to worry about saturating the network with insert events in delta files.

Apache Hive query basics

Using Apache Hive, you can query distributed data storage. You need to know ANSI SQL to view, maintain, or analyze Hive data. Examples of the basics, such as how to insert, update, and delete data from a table, helps you get started with Hive.

Hive supports ANSI SQL and atomic, consistent, isolated, and durable (ACID) transactions. For updating data, you can use the MERGE statement, which meets ACID standards. Materialized views optimize queries based on access patterns. Hive supports tables up to 300PB in Optimized Row Columnar (ORC) format. Other file formats are also supported. You can create tables that resemble those in a traditional relational database. You use familiar insert, update, delete, and merge SQL statements to query table data. The insert statement writes data to tables. Update and delete statements modify and delete values already written to Hive. The merge statement streamlines updates, deletes, and changes data capture operations by drawing on co-existing tables. These statements support auto-commit that treats each statement as a separate transaction and commits it after the SQL statement is executed.

Related Information

ORC Language Manual on the Apache wiki

Querying the information_schema database

Hive supports the ANSI-standard information_schema database, which you can query for information about tables, views, columns, and your Hive privileges. The information_schema data reveals the state of the system, similar to

sys database data, but in a user-friendly, read-only way. You can use joins, aggregates, filters, and projections in information_schema queries.

About this task

One of the following steps involves changing the time interval for synchronization between HiveServer and the policy. HiveServer responds to any policy changes within this time interval. You can query the information_schema database for only your own privilege information.

Procedure

1. Open Ranger Access Manager, and check that the preloaded default database tables columns and information_schema database policies are enabled for group public.

Policy ID	Policy Name	Policy Labels	Status	Audit Logging	Roles	Groups	Users
7	all - global	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
8	all - database, table, column	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
9	all - database, table	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
10	all - database	--	Enabled	Enabled	--	public	hive beacon dpprofiler hue + More..
11	all - hiveservice	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
12	all - database, udf	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
13	all - url	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
14	default database tables columns	--	Enabled	Enabled	--	public	--
15	Information_schema database ...	--	Enabled	Enabled	--	public	--

The information schema database is synchronized every half hour by default.

2. From the Beeline shell, start Hive, and check for the information_schema database:

```
SHOW DATABASES;
...
+---------------------+
|    database_name    |
+---------------------+
| default             |
| information_schema  |
| sys                 |
+---------------------+
```

3. Use the information_schema database to list tables in the database.

```
USE information_schema;
...
SHOW TABLES;
...
+--------------------+
|      tab_name      |
+--------------------+
```

```
| column_privileges  |
| columns            |
| schemata           |
| table_privileges   |
| tables             |
| views              |
+--------------------+
```

4. Query the information_schema database to see, for example, information about tables into which you can insert values.

```
SELECT * FROM information_schema.tables WHERE is_insertable_into='YES' l
imit 2;
...
+--------------------+-------------------+-----------------
|tables.table_catalog|tables.table_schema|tables.table_name
+--------------------+-------------------+-----------------
|default             |default            |students2
|default             |default            |t3
```

Inserting data into a table

To insert data into a table you use a familiar ANSI SQL statement. A simple example shows you have to accomplish this basic task.

About this task

To insert data into an ACID table, use the Optimized Row Columnar (ORC) storage format. To insert data into a non-ACID table, you can use other supported formats. You can specify partitioning as shown in the following syntax:

INSERT INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)] VALUES values_row [, values_row...]

where

values_row is (value [, value]) .

A value can be NULL or any SQL literal.

Procedure

1. Create an ACID table to contain student information.
   CREATE TABLE students (name VARCHAR(64), age INT, gpa DECIMAL(3,2));
2. Insert name, age, and gpa values for a few students into the table.
   INSERT INTO TABLE students VALUES ('fred flintstone', 35, 1.28), ('barney rubble', 32, 2.32);
3. Create a table called pageviews and assign null values to columns you do not want to assign a value.

```
CREATE TABLE pageviews (userid VARCHAR(64), link STRING, from STRING) PA
RTITIONED BY (datestamp STRING) CLUSTERED BY (userid) INTO 256 BUCKETS;
INSERT INTO TABLE pageviews PARTITION (datestamp = '2014-09-23') VALUES
('jsmith', 'mail.com', 'sports.com'), ('jdoe', 'mail.com', null);
INSERT INTO TABLE pageviews PARTITION (datestamp) VALUES ('tjohnson', 'sp
orts.com', 'finance.com', '2014-09-23'), ('tlee', 'finance.com', null, '
2014-09-21');
```

   The ACID data resides in the warehouse.

Updating data in a table

The syntax describes the UPDATE statement you use to modify data already stored in a table. An example shows how to apply the syntax.

About this task

You construct an UPDATE statement using the following syntax:

UPDATE tablename SET column = value [, column = value ...] [WHERE expression];

Depending on the condition specified in the optional WHERE clause, an UPDATE statement might affect every row in a table. The expression in the WHERE clause must be an expression supported by a SELECT clause. Subqueries are not allowed on the right side of the SET statement. Partition columns cannot be updated.

Before you begin

You must have SELECT and UPDATE privileges to use the UPDATE statement.

Procedure

Create a statement that changes the values in the name column of all rows where the gpa column has the value of 1.0.
UPDATE students SET name = null WHERE gpa <= 1.0;

Merging data in tables

A sample statement shows how you can conditionally insert existing data in Hive tables using the ACID MERGE statement. Additional merge operations are mentioned.

About this task

The MERGE statement is based on ANSI-standard SQL.

Procedure

1. Construct a query to update the customers' names and states in customer table to match the names and states of customers having the same IDs in the new_customer_stage table.
2. Enhance the query to insert data from new_customer_stage table into the customer table if none already exists.

   Update or delete data using MERGE in a similar manner.

```
MERGE INTO customer USING (SELECT * FROM new_customer_stage) sub ON sub.id
 = customer.id
WHEN MATCHED THEN UPDATE SET name = sub.name, state = sub.new_state
WHEN NOT MATCHED THEN INSERT VALUES (sub.id, sub.name, sub.state);
```

Related Information

Merge documentation on the Apache wiki

Deleting data from a table

You use the DELETE statement to delete data already written to an ACID table.

About this task

Use the following syntax to delete data from a Hive table. DELETE FROM tablename [WHERE expression];

Procedure

Delete any rows of data from the students table if the gpa column has a value of 1 or 0.
DELETE FROM students WHERE gpa <= 1,0;

Using a subquery

Hive supports subqueries in FROM clauses and WHERE clauses that you can use for many Apache Hive operations, such as filtering data from one table based on contents of another table.

About this task

A subquery is a SQL expression in an inner query that returns a result set to the outer query. From the result set, the outer query is evaluated. The outer query is the main query that contains the inner subquery. A subquery in a WHERE clause includes a query predicate and predicate operator. A predicate is a condition that evaluates to a Boolean value. The predicate in a subquery must also contain a predicate operator. The predicate operator specifies the relationship tested in a predicate query.

Procedure

Select all the state and net_payments values from the transfer_payments table if the value of the year column in the table matches a year in the us_census table.

```
SELECT state, net_payments
FROM transfer_payments
WHERE transfer_payments.year IN (SELECT year FROM us_census);
```

The predicate starts with the first WHERE keyword. The predicate operator is the IN keyword.

The predicate returns true for a row in the transfer_payments table if the year value in at least one row of the us_census table matches a year value in the transfer_payments table.

Subquery restrictions

To construct queries efficiently, you must understand the restrictions of subqueries in WHERE clauses.

- Subqueries must appear on the right side of an expression.
- Nested subqueries are not supported.
- A single query can have only one subquery expression.
- Subquery predicates must appear as top-level conjuncts.
- Subqueries support four logical operators in query predicates: IN, NOT IN, EXISTS, and NOT EXISTS.
- The IN and NOT IN logical operators may select only one column in a WHERE clause subquery.
- The EXISTS and NOT EXISTS operators must have at least one correlated predicate.
- The left side of a subquery must qualify all references to table columns.
- References to columns in the parent query are allowed only in the WHERE clause of the subquery.
- Subquery predicates that reference a column in a parent query must use the equals (=) predicate operator.
- Subquery predicates may not refer only to columns in the parent query.
- Correlated subqueries with an implied GROUP BY statement may return only one row.
- All unqualified references to columns in a subquery must resolve to tables in the subquery.
- Correlated subqueries cannot contain windowing clauses.

Use wildcards with SHOW DATABASES

In a SHOW DATABASES LIKE statement, you can use wildcards, and in this release of Hive, specify any character or a single character.

About this task

SHOW DATABASES or SHOW SCHEMAS lists all of the databases defined in Hive metastore. You can use the following wildcards:

%

Matches any single character or multiple characters

_

Matches any single character

|

Matches either the part of the pattern on the left or the right side of the pipe.

For example, 'students', 'stu%', 'stu_ents' match the database named students.

Aggregating and grouping data

You use AVG, SUM, or MAX functions to aggregate data, and the GROUP BY clause to group data query results in one or more table columns..

About this task

The GROUP BY clause explicitly groups data. Hive supports implicit grouping, which occurs when aggregating the table in full.

Procedure

1. Construct a query that returns the average salary of all employees in the engineering department grouped by year.

```
SELECT year, AVG(salary)
FROM Employees
WHERE Department = 'engineering' GROUP BY year;
```

2. Construct an implicit grouping query to get the highest paid employee.

```
SELECT MAX(salary) as highest_pay,
AVG(salary) as average_pay
FROM Employees
WHERE Department = 'engineering';
```

Querying correlated data

You can query one table relative to the data in another table.

About this task

A correlated query contains a query predicate with the equals (=) operator. One side of the operator must reference at least one column from the parent query and the other side must reference at least one column from the subquery. An uncorrelated query does not reference any columns in the parent query.

Procedure

Select all state and net_payments values from the transfer_payments table for years during which the value of the state column in the transfer_payments table matches the value of the state column in the us_census table.

```
SELECT state, net_payments
FROM transfer_payments
WHERE EXISTS
   (SELECT year
```

```
    FROM us_census
    WHERE transfer_payments.state = us_census.state);
```

This query is correlated because one side of the equals predicate operator in the subquery references the state column in the transfer_payments table in the parent query and the other side of the operator references the state column in the us_census table.

This statement includes a conjunct in the WHERE clause.

A conjunct is equivalent to the AND condition, while a disjunct is the equivalent of the OR condition The following subquery contains a conjunct:

... WHERE transfer_payments.year = "2018" AND us_census.state = "california"

The following subquery contains a disjunct:

... WHERE transfer_payments.year = "2018" OR us_census.state = "california"

Using common table expressions

Using common table expression (CTE), you can create a temporary view that repeatedly references a subquery.

A CTE is a set of query results obtained from a simple query specified within a WITH clause that immediately precedes a SELECT or INSERT keyword. A CTE exists only within the scope of a single SQL statement and not stored in the metastore. You can include one or more CTEs in the following SQL statements:

- SELECT
- INSERT
- CREATE TABLE AS SELECT
- CREATE VIEW AS SELECT

Recursive queries are not supported and the WITH clause is not supported within subquery blocks.

Use a CTE in a query

You can use a common table expression (CTE) to simplify creating a view or table, selecting data, or inserting data.

Procedure

1. Use a CTE to create a table based on another table that you select using the CREATE TABLE AS SELECT (CTAS) clause.

```
CREATE TABLE s2 AS WITH q1 AS (SELECT key FROM src WHERE key = '4') SELECT
 * FROM q1;
```

2. Use a CTE to create a view.

```
CREATE VIEW v1 AS WITH q1 AS (SELECT key FROM src WHERE key='5') SELECT *
 from q1;
```

3. Use a CTE to select data.

```
WITH q1 AS (SELECT key from src where key = '5')
  SELECT * from q1;
```

4. Use a CTE to insert data.

```
CREATE TABLE s1 LIKE src;
WITH q1 AS (SELECT key, value FROM src WHERE key = '5') FROM q1 INSERT OV
ERWRITE TABLE s1 SELECT *;
```

Comparing tables using ANY/SOME/ALL

You learn how to use quantified comparison predicates (ANY/SOME/ALL) in non-correlated subqueries according to the SQL standard. SOME is any alias for ANY.

About this task

You can use one of the following operators with a comparison predicate:

- >
- <
- >=
- <=
- <>
- =

ALL:

- If the table is empty, or the comparison is true for every row in subquery table, the predicate is true for that predicand.
- If the comparison is false for at least one row, the predicate is false.

SOME or ANY:

- If the comparison is true for at least one row in the subquery table, the predicate is true for that predicand.
- If the table is empty or the comparison is false for each row in subquery table, the predicate is false.

If the comparison is neither true nor false, the result is undefined.

For example, you run the following query to match any value in c2 of tbl equal to any value in c1 from the same tbl:

```
select c1 from tbl where c1 = ANY (select c2 from tbl);
```

You run the following query to match all values in c1 of tbl not equal to any value in c2 from the same tbl.

```
select c1 from tbl where c1 <> ALL (select c2 from tbl);
```

Escaping an invalid identifier

When you need to use reserved words, special characters, or a space in a column or partition name, enclose it in backticks (`).

About this task

An identifier in SQL is a sequence of alphanumeric and underscore (_) characters enclosed in backtick characters. In Hive, these identifiers are called quoted identifiers and are case-insensitive. You can use the identifier instead of a column or table partition name.

Before you begin

You have set the following parameter to column in the hive-site.xml file to enable quoted identifiers:

Set the hive.support.quoted.identifiers configuration parameter to column in the hive-site.xml file to enable quoted identifiers in column names. Valid values are none and column. For example, in Hive execute the following command: SET hive.support.quoted.identifiers = column.

Procedure

1. Create a table named test that has two columns of strings specified by quoted identifiers:
   CREATE TABLE test (`x+y` String, `a?b` String);
2. Create a table that defines a partition using a quoted identifier and a region number:
   CREATE TABLE partition_date-1 (key string, value string) PARTITIONED BY (`dt+x` date, region int);
3. Create a table that defines clustering using a quoted identifier:
   CREATE TABLE bucket_test(`key?1` string, value string) CLUSTERED BY (`key?1`) into 5 buckets;

CHAR data type support

Knowing how Hive supports the CHAR data type compared to other databases is critical during migration.

Table 1: Trailing Whitespace Characters on Various Databases

Data Type	Hive	Oracle	SQL Server	MySQL	Teradata
CHAR	Ignore	Ignore	Ignore	Ignore	Ignore
VARCHAR	Compare	Compare	Configurable	Ignore	Ignore
STRING	Compare	N/A	N/A	N/A	N/A

ORC vs Parquet formats

The differences between Optimized Row Columnar (ORC) file format for storing data in SQL engines are important to understand. Query performance improves when you use the appropriate format for your application.

ORC and Parquet capabilities comparison

The following table compares SQL engine support for ORC and Parquet.

Table 2:

Capability	Cloudera Data Warehouse	ORC	Parquet	SQL Engine
Read non-transactional data	Apache Hive	#	#	Hive
Read non-transactional data	Apache Impala	#	#	Impala
Read/Write Full ACID tables	Apache Hive	#		Hive
Read Full ACID tables	Apache Impala	#		Impala & HMS
Read Insert-only managed tables	Apache Impala	#	#	Impala & HMS
Column index	Apache Hive	#	#	Hive & HMS
Column index	Apache Impala		#	Impala & HMS
CBO uses column metadata	Apache Hive	#		Hive & HMS
Recommended format	Apache Hive	#		Hive & HMS
Recommended format	Apache Impala		#	Impala & HMS
Vectorized reader	Apache Hive	#	#	Hive & HMS

Capability	Cloudera Data Warehouse	ORC	Parquet	SQL Engine
Read complex types	Apache Impala	#	#	Impala & HMS
Read/write complex types	Apache Hive	#	#	Hive & HMS

Creating a default directory for managed tables

You can specify a top-level directory for managed tables when creating a Hive database.

About this task

Create a default directory for managed tables only after limiting CREATE DATABASE and ALTER DATABASE statements to users having the Admin role, which has hive service user permissions. Permissions to the managed directory must be limited to the hive service user. In addition to restricting permissions to the hive user, you can further secure managed tables using Ranger fine-grained permissions, such as row-level filtering and column masking.

As Admin, you specify a managed location within the default location specified by the hive.metastore.warehouse.dir configuration property to give managed tables a common location for governance policies. The managed location designates a single root directory for all tenant tables, managed and external.

Use the following syntax to create a database that specifies a location for managed tables:

```
CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name
    [COMMENT database_comment]
    [LOCATION external_table_path]
    [MANAGEDLOCATION managed_table_directory_path]
    [WITH DBPROPERTIES (property_name=property_value, ...)];
```

Do not set LOCATION and MANAGEDLOCATION to the same file system path.

Use the following syntax to set or change a location for managed tables.

```
ALTER (DATABASE|SCHEMA) database_name SET MANAGEDLOCATION [managed_table_dir
ectory_path];
```

Procedure

1. Create a database mydatabase that specifies a top level directory named sales for managed tables.

   ```
   CREATE DATABASE mydatabase MANAGEDLOCATION '/warehouse/tablespace/managed/
   hive/sales';
   ```

2. Change the abc_sales database location to the same location as mydatabase.

   ```
   ALTER DATABASE abc_sales SET MANAGEDLOCATION '/warehouse/tablespace/mana
   ged/hive/sales';
   ```

Generating surrogate keys

You can use the built-in SURROGATE_KEY user-defined function (UDF) to automatically generate numerical Ids for rows as you enter data into a table. The generated surrogate keys can replace wide, multiple composite keys.

Before you begin

Hive supports the surrogate keys on ACID tables only, as described in the following matrix of table types:

Table Type	ACID	Surrogate Keys	File Format	INSERT	UPDATE/DELETE
Managed: CRUD transactional	Yes	Yes	ORC	Yes	Yes
Managed: Insert-only transactional	Yes	Yes	Any	Yes	No
Managed: Temporary	No	No	Any	Yes	No
External	No	No	Any	Yes	No

The table you want to join using surrogate keys cannot have column types that need casting. These data types must be primitives, such as INT or STRING.

About this task

Joins using the generated keys are faster than joins using strings. Using generated keys does not force data into a single node by a row number. You can generate keys as abstractions of natural keys. Surrogate keys have an advantage over UUIDs, which are slower and probabilistic.

The SURROGATE_KEY UDF generates a unique Id for every row that you insert into a table. It generates keys based on the execution environment in a distributed system, which includes a number of factors, such as internal data structures, the state of a table, and the last transaction id. Surrogate key generation does not require any coordination between compute tasks.

The UDF takes either no arguments or two arguments:

- Write Id bits
- Task Id bits

Procedure

1. Create a students table in the default ORC format that has ACID properties.

```
CREATE TABLE students (row_id INT, name VARCHAR(64), dorm INT);
```

2. Insert data into the table. For example:

```
INSERT INTO TABLE students VALUES (1, 'fred flintstone', 100), (2, 'barney
 rubble', 200);
```

3. Create a version of the students table using the SURROGATE_KEY UDF.

```
CREATE TABLE students_v2
(`ID` BIGINT DEFAULT SURROGATE_KEY(),
 row_id INT,
 name VARCHAR(64),
 dorm INT,
 PRIMARY KEY (ID) DISABLE NOVALIDATE);
```

4. Insert data, which automatically generates surrogate keys for the primary keys.

```
INSERT INTO students_v2 (row_id, name, dorm) SELECT * FROM students;
```

5. Take a look at the surrogate keys.

```
SELECT * FROM students_v2;
```

```
+-----------------+--------------------+------------------+-----------
--------+
| students_v2.id  | students_v2.row_id  | students_v2.name  | students_v2.
dorm  |
+-----------------+--------------------+------------------+-----------
--------+
| 1099511627776   | 1                  | fred flintstone   | 100
      |
| 1099511627777   | 2                  | barney rubble     | 200
      |
+-----------------+--------------------+------------------+-------------
------+
```

6. Add the surrogate keys as a foreign key to another table, such as a student_grades table, to speed up subsequent joins of the tables.

```
ALTER TABLE student_grades ADD COLUMNS (gen_id BIGINT);

MERGE INTO student_grades g USING students_v2 s ON g.row_id = s.row_id
WHEN MATCHED THEN UPDATE SET gen_id = s.id;
```

Now you can achieve fast joins on the surrogate keys.

Partitions and performance

A brief description of partitions and the performance benefits includes characters you must avoid when creating a partition. Examples of creating a partition and inserting data in a partition introduce basic partition syntax. Best practices for partitioning are mentioned.

A table you create without partitioning puts the data in a single directory. Partitioning divides the data into multiple directories. Queries of one or more columns based on the directories can run faster. Lengthy full table scans are avoided. Only data in the relevant directory is scanned. For example, a school_records table partitioned on a year column, segregates values by year into separate directories. A WHERE condiition such as YEAR=2020, YEAR IN (2020,2019), or YEAR BETWEEN 2001 AND 2010 scans only the data in the appropriate directory to resolve the query. Using partitions typically improves query performance.

In a SQL query, you define the partition as shown in the following example:

```
CREATE TABLE sale(id in, amount decimal) PARTITIONED BY (xdate string, state
 string);
```

To insert data into this table, you specify the partition key for fast loading:

```
INSERT INTO sale (xdate='2016-03-08', state='CA') SELECT * FROM staging_tabl
e WHERE xdate='2016-03-08' AND state='CA';
```

You do not need to specify dynamic partition columns. Hive generates a partition specification if you enable dynamic partitions.

Examples of a query on partitioned data

```
INSERT INTO sale (xdate, state)
SELECT * FROM staging_table;
```

Follow these best practices when you partition tables and query partitioned tables:

- Never partition on a unique ID.
- Size partitions to greater than or equal to 1 GB on average.
- Design queries to process not more than 1000 partitions.

Invalid Characters in a Partition Name

When you create a partition, do not use the following characters in a partition name:

- colon
- question mark
- percent

If you use these characters in a partition name, your directories will be named using the URL encoding of these characters, as described in "Why some special characters should not be used in a partition name in Hive/Impala."

Related Information

Why some special characters should not be used in a partition name in Hive/Impala

Repairing partitions manually using MSCK repair

The MSCK REPAIR TABLE command was designed to manually add partitions that are added to or removed from the file system, but are not present in the Hive metastore.

About this task

This task assumes you created a partitioned external table named emp_part that stores partitions outside the warehouse. You remove one of the partition directories on the file system. This action renders the metastore inconsistent with the file system. You repair the discrepancy manually to synchronize the metastore with the file system.

Procedure

1. Remove the dept=sales object from the file system.
2. From the Hive command line, look at the emp_part table partitions.

   ```
   SHOW PARTITIONS emp_part;
   ```

   The list of partitions is stale; it still includes the dept=sales directory.

   ```
   +----------------+
   |   partition    |
   +----------------+
   | dept=finance   |
   | dept=sales     |
   | dept=service   |
   +----------------+
   ```

3. Repair the partition manually.

   ```
   MSCK REPAIR TABLE emp_part DROP PARTITIONS;
   ```

Query scheduling

Apache Hive scheduled queries is a simple, secure way to create, manage, and monitor scheduled jobs. You can replace OS-level schedulers like cron, Apache Oozie, or Apache Airflow with scheduled queries.

Using SQL statements, you can schedule Hive queries to run on a recurring basis, monitor query progress, and optionally disable a query schedule. You can run queries to ingest data periodically, refresh materialized views, replicate data, and perform other repetitive tasks. For example, you can insert data from a stream into a transactional table every 10 minutes, refresh a materialized view used for BI reporting every hour, and replicate data from one cluster to another on a daily basis.

A Hive scheduled query consists of the following parts:

- A unique name for the schedule
- The SQL statement to be executed
- The execution schedule defined by a Quartz cron expression.

Quartz cron expressions are expressive and flexible. For instance, expressions can describe simple schedules such as every 10 minutes, but also an execution happening at 10 AM on the first Sunday of the month in January, February in 2021, 2022. You can describe common schedules in an easily comprehensible format, for example every 20 minutes or every day at ‘3:25:00’.

Operation

A scheduled query belongs to a namespace, which is a collection of HiveServer (HS2) instances that are responsible for executing the query. Scheduled queries are stored in the Hive metastore. The metastore stores scheduled queries, the status of ongoing and previously executed statements, and other information. HiveServer periodically polls the metastore to retrieve scheduled queries that are due to be executed. If you run multiple HiveServer roles, the metastore guarantees that only one of them executes a certain scheduled query at any given time.

You create, alter, and drop scheduled queries using dedicated SQL statements.

Related Information

Apache Hive Language Manual--Scheduled Queries

Enabling scheduled queries

You need to know how to enable and disable scheduled queries and understand how the default state can prevent you from running a query unintentionally.

About this task

Scheduled queries are created in disabled mode by default in Cloudera. This default helps prevent you from running new scheduled queries inadvertantly. You must explicitly enable new scheduled queries. A scheduled query can keep the cluster awake at the wrong time. To enable a particular schedule, for example schedule1, you run the ALTER SCHEDULED QUERY statement:

```
ALTER SCHEDULED QUERY schedule1 ENABLE;
```

To disable this schedule: ALTER SCHEDULED QUERY schedule1 DISABLE;

Related Information

Apache Hive Language Manual--Scheduled Queries

Periodically rebuilding a materialized view

Using materialized views can enhance query performance. You need to refresh materialized view contents when new data is added to the underlying table. Instead of rebuilding the materialized view manually, you can schedule this task. Rebuilding occurs periodically and transparently to users.

About this task

This task assumes you created the following schemas for storing employee and departmental information:

```
CREATE TABLE emps (
empid INTEGER,
deptno INTEGER,
name VARCHAR(256),
salary FLOAT,
hire_date TIMESTAMP);

CREATE TABLE depts (
deptno INTEGER,
deptname VARCHAR(256),
locationid INTEGER);
```

Imagine that you add data for a number of employees to the table. Assume many users of your database issue queries to access to data about the employees hired during last year including the department they belong to.

You perform the steps below to create a materialized view of the table to address these queries. Imagine new employees are hired and you add their records to the table. These changes render the materialized view contents outdated. You need to refresh its contents. You create a scheduled query to perform this task. The scheduled rebuilding will not occur unless there are changes to the input tables. You test the scheduled query by bypassing the schedule and executing the schedule immediately. Finally, you change the schedule to rebuild less often.

Procedure

1. To handle many queries to access recently hired employee and departmental data, create a materialized view.

```
CREATE MATERIALIZED VIEW mv_recently_hired AS
  SELECT empid, name, deptname, hire_date FROM emps
  JOIN depts ON (emps.deptno = depts.deptno)
  WHERE hire_date >= '2020-01-01 00:00:00';
```

2. Use the materialized view by querying the employee data.

```
SELECT empid, name FROM emps
JOIN depts  ON (emps.deptno = depts.deptno)
WHERE hire_date >= '2020-03-01 00:00:00' AND deptname = 'finance';
```

3. Assuming new hiring occurred and you added new records to the emps table, rebuild the materialized view.

```
ALTER MATERIALIZED VIEW mv_recently_hired REBUILD;
```

   The rebuilding refreshes the contents of the materialized view.

4. Create a scheduled query to invoke the rebuild statement every 10 minutes.

```
CREATE SCHEDULED QUERY scheduled_rebuild
EVERY 10 MINUTES AS
```

```
ALTER MATERIALIZED VIEW mv_recently_hired REBUILD;
```

A rebuild executes every 10 minutes, assuming changes to the emp table occur within that period. If a materialized view can be rebuilt incrementally, the scheduled rebuild does not occur unless there are changes to the input tables.

5. To test the schedule, run a scheduled query immediately.

```
ALTER SCHEDULED QUERY scheduled_rebuild EXECUTE;
```

6. Change the frequency of the rebuilding.

```
ALTER SCHEDULED QUERY scheduled_rebuild EVERY 20 MINUTES;
```

Related Information

Apache Hive Language Manual--Scheduled Queries

Getting scheduled query information and monitor the query

After you create a scheduled query you can access information about it in the scheduled_queries table of the Hive information schema. You can also use the information schema to monitor scheduled query execution.

Procedure

1. Query the information schema to get information about a schedule.

```
SELECT *
FROM information_schema.scheduled_queries
WHERE schedule_name = 'scheduled_rebuild';
```

The following information appears about the scheduled query:

scheduled_query_id
: Unique numeric identifier for a scheduled query.

schedule_name
: Name of the scheduled query.

enabled
: Whether the scheduled query is currently enabled or not.

cluster_namespace
: Namespace that the scheduled query belongs to.

schedule
: Schedule described as a Quartz cron expression.

user
: Owner of the scheduled query.

query
: SQL query to be executed.

next_execution
: When the next execution of this scheduled query is due.

2. Monitor the most recent scheduled query execution.

```
SELECT *
```

```
FROM information_schema.scheduled_executions;
```

You can configure the retention period for this information in the Hive metastore.

scheduled_execution_id

Unique numeric identifier for a scheduled query execution.

schedule_name

Name of the scheduled query associated with this execution.

executor_query_id

Query ID assigned to the execution by HiveServer (HS2).

state

One of the following phases of execution.

- STARTED. A scheduled query is due and a HiveServer instance has retrieved its information.
- EXECUTING. HiveServer is executing the query and reporting progress in configurable intervals.
- FAILED. The query execution was stopped due to an error or exception.
- FINISHED. The query execution was successful.
- TIMED_OUT. HiveServer did not provide an update on the query status for more than a configurable timeout.

start_time

Start time of execution.

end_time

End time of execution.

elapsed

Difference between start and end time.

error_message

If the scheduled query failed, it contains the error message associated with its failure.

last_update_time

Time of the last update of the query status by HiveServer.

Related Information

Apache Hive Language Manual--Scheduled Queries

Materialized views

A materialized view is a Hive-managed database object that holds a query result you can use to speed up the execution of a query workload. If your queries are repetitive, you can reduce latency and resource consumption by using materialized views. You create materialized views to optimize your queries automatically.

Using a materialized view, the optimizer can compare old and new tables, rewrite queries to accelerate processing, and manage maintenance of the materialized view when data updates occur. The optimizer can use a materialized view to fully or partially rewrite projections, filters, joins, and aggregations.

In Unified Analytics, you can perform the following materialized view operations in either Hive or Impala:

- Create a materialized view of queries or subqueries
- Drop a materialized view
- Show materialized views
- Describe a materialized view

- Enable or disable query rewriting based on a materialized view
- Globally enable or disable rewriting based on any materialized view
- Use partitioning to improve the performance of materialized views

When you enable BI mode to use Apache DataSketches approximations, a materialized view automatically calls DataSketches for some types of operations.

Related Information

Materialized view commands

Creating and using a materialized view

You can create a materialized view of a query to calculate and store results of an expensive operation, such as a particular join, on a managed, ACID table that you repeatedly run. When you issue queries specified by that materialized view, the optimizer rewrites the query based on it. This action saves reprocessing. Query performance improves.

About this task

In the tasks that follow, first you create, or use an existing, Hive Virtual Warehouse. You create and populate example tables. The tables are managed tables. You cannot create a materialized view of an external table. You create a materialized view of a join of the tables. Subsequently, you run a query to join the tables, and the query plan takes advantage of the precomputed join to accelerate processing. These over-simplified tasks show the syntax and output of a materialized view, and do not demonstrate accelerated processing that occurs in a real-world task, processing a large amount of data.

Before you begin

- You have access to an existing Hive Virtual Warehouse, or you created a new one.
-

Related Information

Materialized view commands

Creating the tables and view

You see how to create simple tables, insert the data, and join the tables using a materialized view. You run the query, and the optimizer takes advantage of the precomputation performed by the materialized view to speed response time.

Procedure

1. Create two ACID tables:

```
CREATE TABLE emps (
empid INT,
deptno INT,
name VARCHAR(256),
salary FLOAT,
hire_date TIMESTAMP);

CREATE TABLE depts (
deptno INT,
deptname VARCHAR(256),
locationid INT);
```

2. Insert some data into the tables for example purposes:

```
INSERT INTO TABLE emps VALUES (10001,101,'jane doe',250000,'2018-01-10');
```

```
INSERT INTO TABLE emps VALUES (10002,100,'somporn klailee',210000,'2017-12
-25');
INSERT INTO TABLE emps VALUES (10003,200,'jeiranan thongnopneua',175000,'
2018-05-05');

INSERT INTO TABLE depts VALUES (100,'HR',10);
INSERT INTO TABLE depts VALUES (101,'Eng',11);
INSERT INTO TABLE depts VALUES (200,'Sup',20);
```

Tables must be ACID (managed) tables.

3. Create a materialized view to join the tables:

```
CREATE MATERIALIZED VIEW mv1
AS SELECT empid, deptname, hire_date
FROM emps JOIN depts
ON (emps.deptno = depts.deptno)
WHERE hire_date >= '2017-01-01';
```

4. Run a query that takes advantage of the precomputation performed by the materialized view:

```
SELECT empid, deptname
FROM emps
JOIN depts
ON (emps.deptno = depts.deptno)
WHERE hire_date >= '2017-01-01'
AND hire_date <= '2019-01-01';
```

Output is:

```
+--------+-----------+
| empid  | deptname  |
+--------+-----------+
| 10003  | Sup       |
| 10002  | HR        |
| 10001  | Eng       |
+--------+-----------+
```

Using optimizations from a subquery

You can create a query having a subquery that the optimizer rewrites based on a materialized view. You create a materialized view, and then run a query that uses the materialized view.

About this task

In this task, you create a materialized view and use it in a subquery to return the number of destination-origin pairs. Suppose the data resides in a table named flights_data that has the following columns:

c_id	dest	origin
1	Chicago	Hyderabad
2	London	Moscow
...		

Procedure

1. Create a table schema definition named flights_data for destination and origin data.

```
CREATE TABLE flights_data(
  c_id INT,
  dest VARCHAR(256),
```

```
  origin VARCHAR(256));
```

2. Create a materialized view that counts destinations and origins.

```
CREATE MATERIALIZED VIEW mv1
AS
  SELECT dest, origin, count(*)
  FROM flights_data
  GROUP BY dest, origin;
```

3. Take advantage of the materialized view to speed your queries when you have to count destinations and origins again.

   For example, use a subquery to select the number of destination-origin pairs like the materialized view.

```
SELECT count(*)/2
FROM(
  SELECT dest, origin, count(*)
  FROM flights_data
  GROUP BY dest, origin
) AS t;
```

   Transparently, the SQL engine uses the work already in place since creation of the materialized view instead of reprocessing.

Related Information

Materialized view commands

Dropping a materialized view

You must understand when to drop a materialized view to successfully drop related tables.

About this task

Drop a materialized view before performing a DROP TABLE operation on a related table. You cannot drop a table that has a relationship with a materialized view.

In this task, you drop a materialized view named mv1 from the database named default.

Procedure

Drop a materialized view in my_database named mv1 .
DROP MATERIALIZED VIEW default.mv;

Showing materialized views

You can list all materialized views in the current database or in another database. You can filter a list of materialized views in a specified database using regular expression wildcards.

About this task

You can use regular expression wildcards to filter the list of materialized views you want to see. The following wildcards are supported:

- Asterisk (*)

  Represents one or more characters.
- Pipe symbol (|)

  Represents a choice.

For example, mv_q* and *mv|q1* match the materialized view mv_q1. Finding no match does not cause an error.

Procedure

1. List materialized views in the current database.
   SHOW MATERIALIZED VIEWS;
2. List materialized views in a particular database.
   SHOW MATERIALIZED VIEWS IN another_database;

Describing a materialized view

You can get summary, detailed, and formatted information about a materialized view.

About this task

This task builds on the task that creates a materialized view named mv1.

Procedure

1. Get summary information about the materialized view named mv1.

```
DESCRIBE mv1;
```

```
+------------+---------------+----------+
|  col_name  |   data_type   | comment  |
+------------+---------------+----------+
| empid      | int           |          |
| deptname   | varchar(256)  |          |
| hire_date  | timestamp     |          |
+------------+---------------+----------+
```

2. Get detailed information about the materialized view named mv1.

```
DESCRIBE EXTENDED mv1;
```

```
+----------------------------+---------------------------------...
|          col_name          |                      data_type   ...
+----------------------------+---------------------------------...
| empid                      | int                                 ...
| deptname                   | varchar(256)                        ...
| hire_date                  | timestamp                           ...
|                            | NULL                                ...
| Detailed Table Information |Table(tableName:mv1, dbName:default, own
er:hive, createTime:1532466307, lastAccessTime:0, retention:0, sd:Storag
eDescriptor(cols:[FieldSchema(name:empid, type:int, comment:null), Field
Schema(name:deptname, type:varchar(256), comment:null), FieldSchema(name
:hire_date, type:timestamp, comment:null)], location:hdfs://myserver.com
:8020/warehouse/tablespace/managed/hive/mv1, inputFormat:org.apache.hado
op.hive.ql.io.orc.OrcInputFormat, outputFormat:org.apache.hadoop.hive.ql
.io.orc.OrcOutputFormat, compressed:false, numBuckets:-1, serdeInfo:SerD
eInfo(name:null, serializationLib:org.apache.hadoop.hive.ql.io.orc.OrcSe
rde, parameters:{}), bucketCols:[], sortCols:[], parameters:{}, skewedIn
fo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocat
ionMaps:{}), storedAsSubDirectories:false), partitionKeys:[], parameters
:{totalSize=488, numRows=4, rawDataSize=520, COLUMN_STATS_ACCURATE={\"BA
SIC_STATS\":\"true\"}, numFiles=1, transient_lastDdlTime=1532466307, buc
keting_version=2}, viewOriginalText:SELECT empid, deptname, hire_date\nF
ROM emps2 JOIN depts\nON (emps2.deptno = depts.deptno)\nWHERE hire_date >=
 '2017-01-17', viewExpandedText:SELECT `emps2`.`empid`, `depts`.`deptname`
, `emps2`.`hire_date`\nFROM `default`.`emps2` JOIN `default`.`depts`\nON
 (`emps2`.`deptno` = `depts`.`deptno`)\nWHERE `emps2`.`hire_date` >= '20
17-01-17', tableType:MATERIALIZED_VIEW, rewriteEnabled:true, creationMet
adata:CreationMetadata(catName:hive, dbName:default, tblName:mv1, tables
```

```
Used:[default.depts, default.emps2], validTxnList:53$default.depts:2:922
3372036854775807::$default.emps2:4:9223372036854775807::, materializatio
nTime:1532466307861), catName:hive, ownerType:USER)
```

3. Get formatting details about the materialized view named mv1.

```
DESCRIBE FORMATTED mv1;
```

```
+-------------------------------+--------------------------------...
|           col_name            |                     data_type  ...
+-------------------------------+--------------------------------...
| # col_name                    | data_type                      ...
| empid                         | int                            ...
| deptname                      | varchar(256)                   ...
| hire_date                     | timestamp                      ...
|                               | NULL                           ...
| # Detailed Table Information  | NULL                           ...
| Database:                     | default                        ...
| OwnerType:                    | USER                           ...
| Owner:                        | hive                           ...
| CreateTime:                   | Tue Jul 24 21:05:07 UTC 2019   ...
| LastAccessTime:               | UNKNOWN                        ...
| Retention:                    | 0                              ...
| Location:                     | hdfs://myserver...
| Table Type:                   | MATERIALIZED_VIEW              ...
| Table Parameters:             | NULL                           ...

|                               | COLUMN_STATS_ACCURATE          ...

|                               | bucketing_version              ...

|                               | numFiles                       ...

|                               | numRows                        ...

|                               | rawDataSize                    ...

|                               | totalSize                      ...

|                               | transient_lastDdlTime          ...
|                               | NULL                           ...

| # Storage Information         | NULL                           ...

| SerDe Library:                | org.apache.hadoop.hive.ql.io.or...

| InputFormat:                  | org.apache.hadoop.hive.ql.io.or...
| OutputFormat:                 | org.apache.hadoop.hive.ql.io.or...
| Compressed:                   | No                             ...

| Num Buckets:                  | -1                             ...

| Bucket Columns:               | []                             ...

| Sort Columns:                 | []                             ...
| # View Information            | NULL                           ...

| View Original Text:           | SELECT empid, deptname, hire_da...

| View Expanded Text:           | SELECT `emps2`.`empid`, `depts`...
| View Rewrite Enabled:         | Yes                            ...
```

Related Information

Materialized view commands

Managing query rewrites

After changes to base tables, the data in a materialized view is stale. You need to know how to prevent the SQL optimizer from rewriting queries in this situation. If you want a query executed without regard to a materialized view, for example to measure the execution time difference, you can disable rewriting and then enable it again.

About this task

As administrator, you can globally enable or disable all query rewrites based on materialized views. By default, the optimizer rewrites a query based on a materialized view.

Procedure

1. Disable rewriting of a query based on a materialized view named mv1 in the default database.

```
ALTER MATERIALIZED VIEW default.mv1 DISABLE REWRITE;
```

2. Enable rewriting of a query based on materialized view mv1.

```
ALTER MATERIALIZED VIEW default.mv1 ENABLE REWRITE;
```

3. Globally enable rewriting of queries based on materialized views by setting a global property.

```
SET hive.materializedview.rewriting=true;
```

Related Information

Materialized view commands

Purposely using a stale materialized view

A rewrite of a query based on a stale materialized view does not occur automatically. If you want a rewrite of a stale or possibly stale materialized view, you can force a rewrite.

About this task

For example, you might want to use the contents of a materialized view of a non-transactional table because the freshness of such a table is unknown. The optimizer cannot determine the data freshness if you use external tables. You can purposely rewrite a query based on a stale materialized views using these techniques:

- Schedule the materialized view for rebuilding. For example, schedule a rebuild to occur every x minutes.
- Adjust the rewriting time window to use stale or possibly stale data for a period of time. For example, schedule the window within which to use stale data for x + y minutes.

Procedure

1. Create a scheduled query to invoke the rebuild statement every 10 minutes.

```
CREATE SCHEDULED QUERY scheduled_rebuild
EVERY 10 MINUTES AS
ALTER MATERIALIZED VIEW mv_recently_hired REBUILD;
```

2. Define the window of time for using stale data.

```
SET hive.materializedview.rewriting.time.window=10min;
```

Creating and using a partitioned materialized view

When creating a materialized view, you can partition selected columns to improve performance. Partitioning separates the view of a table into parts, which often improves query rewrites of partition-wise joins of materialized views with tables or other materialized views.

About this task

This task assumes you created a materialized view of the emps and depts tables and assumes you created these tables. The emps table contains the following data:

empid	deptno	name	salary	hire_date
10001	101	jane doe	250000	2018-01-10
10005	100	somporn klailee	210000	2017-12-25
10006	200	jeiranan thongnopneua	175000	2018-05-05

The depts table contains the following data:

deptno	deptname	locationid
100	HR	10
101	Eng	11
200	Sup	20

In this task, you create two materialized views: one partitions data on department; the other partitions data on hire date. You select data, filtered by department,from the original table, not from either one of the materialized views. The explain plan shows that Hive rewrites your query for efficiency to select data from the materialized view that partitions data by department. In this task, you also see the effects of rebuilding a materialized view.

Procedure

1. Create a materialized view of the emps table that partitions data into departments.

```
CREATE MATERIALIZED VIEW partition_mv_1 PARTITIONED ON (deptno)
AS SELECT hire_date, deptno FROM emps WHERE deptno > 100 AND deptno < 200;
```

2. Create a second materialized view that partitions the data on the hire date instead of the department number.

```
CREATE MATERIALIZED VIEW partition_mv_2 PARTITIONED ON (hire_date)
  AS SELECT deptno, hire_date FROM emps where deptno > 100 AND deptno < 2
00;
```

3. Generate an extended explain plan by selecting data for department 101 directly from the emps table without using the materialized view.

```
EXPLAIN EXTENDED SELECT deptno, hire_date FROM emps where deptno = 101;
```

The explain plan shows that Hive rewrites your query for efficiency, using the better of the two materialized views for the job: partition_mv_1.

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| OPTIMIZED SQL: SELECT CAST(101 AS INTEGER) AS `deptno`, `hire_date` |
| FROM `default`.`partition_mv_1`                    |
| WHERE 101 = `deptno`                               |
| STAGE DEPENDENCIES:                                |
|   Stage-0 is a root stage
```

```
...
```

4. Correct Jane Doe's hire date to February 12, 2018, rebuild one of the materialized views, but not the other, and compare contents of both materialized views.

```
INSERT INTO emps VALUES (10001,101,'jane doe',250000,'2018-02-12');
ALTER MATERIALIZED VIEW partition_mv_1 REBUILD;
SELECT * FROM partition_mv_1 where deptno = 101;
SELECT * FROM partition_mv_2 where deptno = 101;
```

The output of selecting the rebuilt partition_mv_1 includes the original row and newly inserted row because INSERT does not perform in-place updates (overwrites).

```
+----------------------------+------------------------+
| partition_mv_1.hire_date   | partition_mv_1.deptno  |
+----------------------------+------------------------+
| 2018-01-10 00:00:00.0      | 101                    |
| 2018-02-12 00:00:00.0      | 101                    |
+----------------------------+------------------------+
```

The output from the other partition is stale because you did not rebuild it:

```
+------------------------+----------------------------+
| partition_mv_2.deptno  | partition_mv_2.hire_date   |
+------------------------+----------------------------+
| 101                    | 2018-01-10 00:00:00.0      |
+------------------------+----------------------------+
```

5. Create a second employees table and a materialized view of the tables joined on the department number.

```
CREATE TABLE emps2 AS SELECT * FROM emps;

CREATE MATERIALIZED VIEW partition_mv_3 PARTITIONED ON (deptno) AS
  SELECT emps.hire_date, emps.deptno FROM emps, emps2
  WHERE emps.deptno = emps2.deptno
  AND emps.deptno > 100 AND emps.deptno < 200;
```

6. Generate an explain plan that joins tables emps and emps2 on department number using a query that omits the partitioned materialized view.

```
EXPLAIN EXTENDED SELECT emps.hire_date, emps.deptno FROM emps, emps2
  WHERE emps.deptno = emps2.deptno
  AND emps.deptno > 100 AND emps.deptno < 200;
```

The output shows that Hive rewrites the query to use the partitioned materialized view partition_mv_3 even though your query omitted the materialized view.

7. Verify that the partition_mv_3 sets up the partition for deptno=101 for partition_mv_3.

```
SHOW PARTITIONS partition_mv_3;
```

Output is:

```
+-------------+
|  partition  |
+-------------+
| deptno=101  |
+-------------+
```

Related Information

Creating and using a materialized view

Materialized view commands

Cloudera Data Warehouse

Cloudera Data Warehouse) supports Hive Hybrid Procedural SQL (HPL/SQL). HPL/SQL is an Apache open source procedural extension for SQL for Hive users. You connect over JDBC to Cloudera Data Warehouse from a client to run HPL/SQL queries. You can also run stored procedures from Hue in Cloudera Data Warehouse.

HPL/SQL includes imperative programming structures (variables, procedures, control flow, and exceptions), and is typically used for ETL. The HPL/SQL language understands the syntax and semantics of most procedural SQL dialects, such as Oracle PL/SQL.

HPL/SQL has its own grammar. For more information about the HPL/SQL language, see the HPL/SQL Reference.

HPL/SQL architecture

HPL/SQL has been re-architected from a command line tool to an integrated part of HiveServer (HS2). From a JDBC client, such as Beeline, you connect to HiveServer through Cloudera Data Warehouse in Cloudera Public Cloud. The interpreter executes the abstract syntax tree (AST) from the parser. Hive metastore securely stores the function and procedure code permanently. The procedure is loaded and cached on demand to the interpreter's memory when needed. You can close the session or restart Hive without losing the definitions.

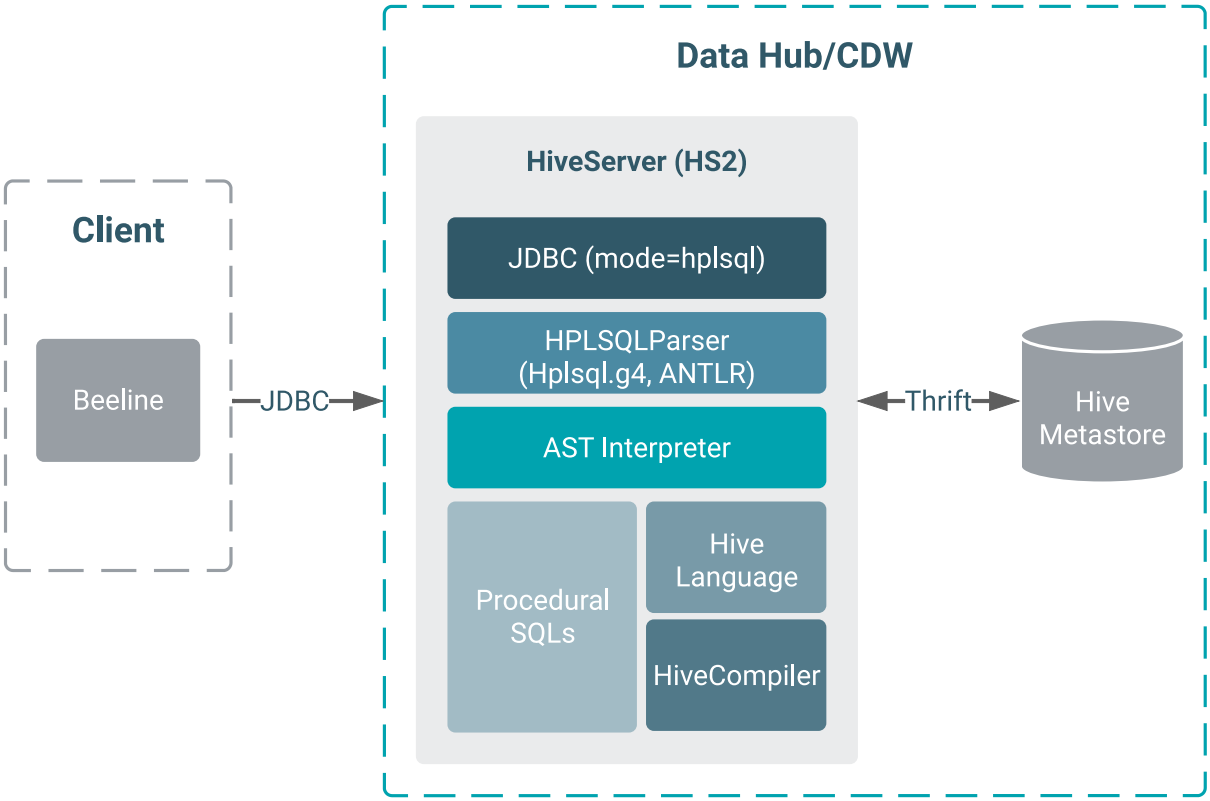

You can enable and use HPL/SQL from any host or third-party tool that can make a JDBC connection to HiveServer. Beeline is a popular client for use with HPL/SQL because other third-party tools do not show you some of the error messages about syntax mistakes.

Enabling HPL/SQL in the beeline connection string

After setting up a client to connect to Cloudera Data Warehouse, you append mode=hplsql to the JDBC URL that connects the client to Cloudera Data Warehouse:

```
beeline -u "jdbc:hive2://<HiveServer host>:10000/default;mode=hplsql"
```

When the client connects to HiveServer in Cloudera Data Warehouse configuring this mode, HPL/SQL is enabled; otherwise, HPL/SQL is disabled.

HPL/SQL key features

- Flow of Control Statements (FOR, WHILE, IF, CASE, LOOP, LEAVE, RETURN)
- Functions, procedures, and packages
- Built-in functions (string manipulations, datetime functions, conversions)
- Exception handling and conditions
- Constants and variable, assignment (DECLARE count INT := 1)
- Processing results using a CURSOR
- UDF to run HPL/SQL scripts from Hive queries

  ```
  (SELECT hplsql('mycustomfunc(:1)', name) FROM users;)
  ```

- Bulk data processing with BULK COLLECT statement

HPL/SQL limitations

- Some of the Hive specific CREATE TABLE parameters are missing.
- No colon syntax to parametrize SQL strings.
- No quoted string literals.
- No GOTO and Label.
- EXECUTE does not have output parameters.
- Some complex data types, such as Arrays and Records are not supported.
- No object-oriented extension.
- Data Analytics Studio (DAS) does not support HPL/SQL.

Related Information

HPL/SQL Reference

Setting up a Cloudera Data Warehouse client

As a Hive user who connects to Cloudera Data Warehouse from a client, you need to know how to set up the client for using HPL/SQL.

Before you begin

- You have created a Database Catalog that is populated with data.

  You can select an option to load sample airline data when you create the catalog.
- You have an existing Hive Virtual Warehouse, or you added a new one, that is configured to connect to the Database Catalog with data.

Procedure

1. On the client end, download the latest version of the Hive JDBC driver from Cloudera Downloads page.
2. Install the driver on the client end.

   Typically, you add the JAR file to the Libraries folder.

3. Log into the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
4. In the Cloudera Data Warehouse service, click Virtual Warehouse in the left navigation panel.
5. On the Virtual Warehouses page, click the options menu of the Hive VW you want to connect to, and select Copy JDBC URL.

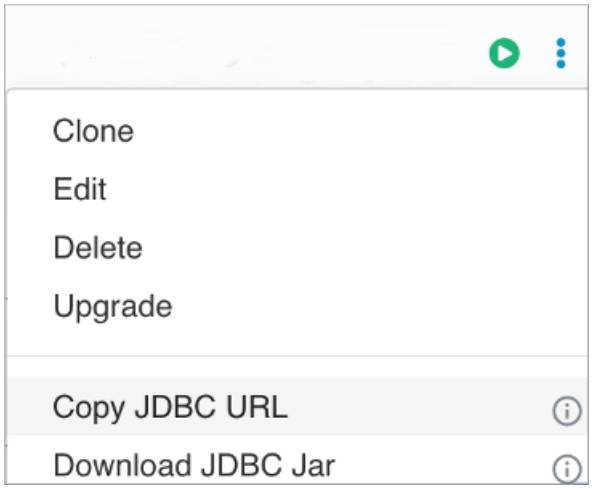

6. Paste the copied JDBC URL into a text file.

```
jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<dwx.company.co
m>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
```

7. Append mode=hplsql to the end of the JDBC URL for connecting to Cloudera Data Warehouse from the client.

```
jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<dwx.company.co
m>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3;mod
e=hplsql
```

8. On the client end, use the URL in Beeline to connect to Cloudera Data Warehouse and enable HPL/SQL.

```
beeline -n <csso_username> -p <password> -u "jdbc:hive2://<your-virtual-
warehouse>.<your-environment>.<dwx.company.com>/default;transportMode=ht
tp;httpPath=cliservice;ssl=true;retries=3;mode=hplsql"
```

At the Hive prompt, you can run HPL/SQL. You can use the forward slash (/) as a statement delimiter because a HPL/SQL statement can have multiple rows consisting of multiple SQL statements.

Related Information

HPL/SQL Reference

Adding a new Virtual Warehouse

Creating a function

You need to know the syntax of HPL/SQL, which closely resembles Oracle's PL/SQL. An example of creating a function and calling it in a Hive SELECT statement demonstrates the HPL/SQL basics.

Procedure

1. Create a table and populate it with some numbers.

```
create table numbers (n int);

for i in 1..10 loop
  insert into numbers values(i);
```

```
end loop;
```

2. Create a function called fizzbuzz to return numbers.

```
create function fizzbuzz(n int) returns string
begin
  if mod(n, 15) == 0 then
     return 'FIZZBUZZ';
  elseif mod(n, 5) == 0 then
     return 'BUZZ';
  elseif mod(n, 3) == 0 then
     return 'FIZZ';
  else
     return n;
  end if;
end;
```

3. Call the function from a Hive select statement.

```
select fizzbuzz(n) from numbers;
```

Output looks something like this:

```
...
1
2
FIZZ
4
BUZZ
1
2
FIZZ
4
BUZZ
FIZZ
7
8
FIZZ
BUZZ
```

Related Information

HPL/SQL Reference

Using the cursor to return record sets

You use a cursor to return recordsets from stored procedures and functions. A cursor defines a set of rows and row-by-row operations on the set. You need to know how to open a cursor, fetch data into local variables row-by-row, and close a cursor.

About this task

A function called fizzbuzz2 calls another function fizzbuzz, created in the previous topic. The new function declares a cursor of type SYS_REFCURSOR to fetch the contents of the numbers table, also created in the previous topic.

Before you begin

- You created the fizzbuzz function.
- You created the numbers table.

Procedure

Use an OPEN-FOR statement to select the number in each row of the numbers table, iteratively FETCH the numerical content of the each row INTO cursor variable num, close the cursor, and print the results.

```
create function fizzbuzz2() returns string
begin
 declare num int = 0;
 declare result string = '';
 declare curs SYS_REFCURSOR;
 open curs for select n from numbers;
 fetch curs into num;
 while (SQLCODE = 0) do
    result = result || fizzbuzz(num) || ' ';
    fetch curs INTO num;
 end while;
 close curs;
 return result;
end;
print fizzbuzz2();
```

The output looks something like this:

```
...
1 2 FIZZ 4 BUZZ 1 2 FIZZ 4 BUZZ FIZZ 7 8 FIZZ BUZZ
No rows affected
```

Related Information

HPL/SQL Reference

Creating a function

Stored procedure examples

You see by example some of the frequently used and useful HPL/SQL code. HPL/SQL is an Apache open source procedural extension for SQL for Hive users.

Greeting function

The following example creates a function that takes the input of your name and returns "hello <name>":

```
CREATE PROCEDURE greet(name STRING)
BEGIN
  PRINT 'Hello ' || name;
END;
```

Pass a cursor to return records

This example of a procedure, called from another procedure, takes a cursor parameter declared OUT; an OPEN-FOR statement opens the cursor and executes the SELECT query. The query returns a subset of the records to the test_even procedure from the numbers table, created in a previous topic.

```
CREATE PROCEDURE even(cur OUT SYS_REFCURSOR)
BEGIN
OPEN cur FOR
SELECT n FROM numbers
WHERE MOD(n, 2) == 0;
END;
```

The test_even procedure below calls the even procedure above, passing the cursor of type SYS_REFCURSOR to fetch each row containing an even number.

```
CREATE PROCEDURE test_even()
BEGIN
 DECLARE curs SYS_REFCURSOR;
 DECLARE n INT = 0;
 DECLARE result STRING = 'Even numbers are: ';
 even(curs);
 FETCH curs INTO n;
 WHILE (SQLCODE = 0) DO
    result = result || n || ' ';
    FETCH curs INTO n;
 END WHILE;
 CLOSE curs;
 PRINT result;
END;
```

Using table types to index fields in records

You can set up table types and reference fields in records using indexing. This example assumes you created the emp table.

```
-- CREATE TABLE emp (name string, age int);
TYPE emp_type IS TABLE OF emp%ROWTYPE INDEX BY BINARY_INTEGER;
TYPE emp_age_type IS TABLE OF emp.age%TYPE INDEX BY BINARY_INTEGER;
TYPE emp_name_type IS TABLE OF STRING INDEX BY BINARY_INTEGER;

DECLARE rows emp_type;
DECLARE ages emp_age_type;

SELECT * INTO rows(1) FROM emp WHERE name = 'alice';
PRINT 'name=' || rows(1).name || ' age=' || rows(1).age;

SELECT age INTO ages(1) FROM emp WHERE name = 'alice';
PRINT 'age=' || ages(1);
```

BULK COLLECT

Using BULK COLLECT, you can retrieve multiple rows in a single fetch quickly.

```
TYPE emp_type IS TABLE OF emp%ROWTYPE INDEX BY BINARY_INTEGER;
    DECLARE rows emp_type;

    SELECT * BULK COLLECT INTO rows FROM emp;


    DECLARE idx INT = rows.FIRST;
    WHILE idx IS NOT NULL LOOP
    PRINT rows(idx).name || ' = ' || rows(idx).age;
    idx = rows.NEXT(idx);
    END LOOP;
```

Related Information

HPL/SQL Reference

Cloudera Runtime 1.0.0

Using Hue

Date published: 2020-07-28
Date modified: 2024-12-05

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

About using Hue.....4
Accessing and using Hue in Cloudera Data Warehouse.....4
Viewing Hive query details.....6
Viewing Hive query history.....6
Viewing Hive query information.....6
Viewing explain plan for a Hive query.....7
Viewing Hive query timeline.....8
Viewing configurations for a Hive query.....8
Viewing DAG information for a Hive query.....10
Viewing Impala query details.....13
Viewing Impala query history.....13
Viewing Impala query information.....14
Viewing the Impala query execution plan.....14
Viewing the Impala query metrics.....15
Terminating Hive queries.....16
Comparing Hive and Impala queries in Hue.....16
How to run a stored procedure from Hue in Cloudera Data WarehouseHow to run a stored procedure from Hue in Cloudera Data Hub.....19
Enabling stored procedures for Hive in Cloudera Data Warehouse.....19
Enabling the SQL editor autocompleter.....20
Using governance-based data discovery.....21
Searching metadata tags.....21
Using Amazon S3 with Hue.....22
Accessing S3 bucket from Hue in Cloudera Data Warehouse with RAZ.....22
Accessing S3 bucket from Hue in Cloudera Data Warehouse without RAZ.....24
Creating tables by importing CSV files from AWS S3 in 26
Using Azure Data Lake Storage Gen2 with Hue.....31
Accessing ADLS Gen2 containers from Hue in Cloudera Data Warehouse with RAZ.....31
Accessing ADLS Gen2 containers from Hue in Cloudera Data Warehouse without RAZ.....32
Creating tables by importing CSV files from ABFS.....34
Granting permission to access S3 and ABFS File Browser in Hue.....35
Creating tables in Hue by importing files.....36
Uploading files with Hue task server enabled.....37
List of supported non-alphanumeric characters for file and directory names in Hue.....37
Unsupported features in Hue.....38
Known limitations in Hue.....38

About using Hue

Hue provides a one-stop querying experience in Cloudera that leverages Hive, Impala, and Unified Analytics SQL engines.

Accessing and using Hue in Cloudera Data Warehouse

Get started using Hue by analyzing and visualizing your data with Impala and Hive SQL query engines.

About this task

To try Hue without having an account, try running sample queries on http://demo.gethue.com/.

Before you begin

Hue uses your LDAP credentials that you have configured for the Cloudera cluster.

Procedure

1. Log into the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, navigate to the **Overview** page.

   Note: You can also launch Hue from the **Virtual Warehouse** page using the same steps.

3. To run Impala queries:
   a) On the **Overview** page under Virtual Warehouses, click on the Hue button.
      The query editor is displayed:

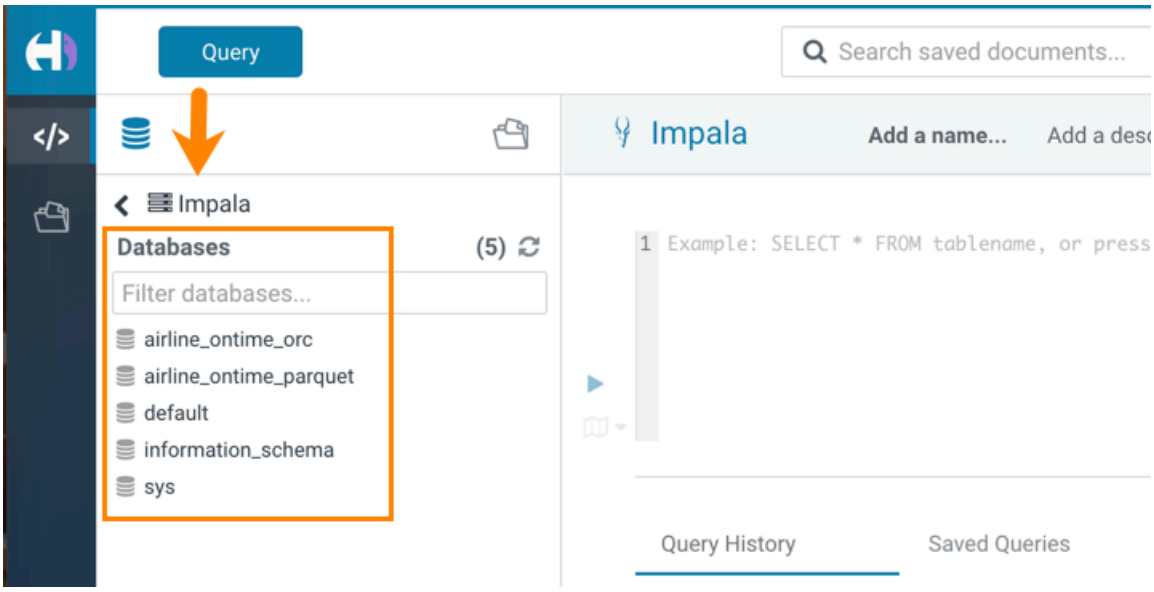

   b) Click a database to view the tables it contains.
      When you click a database, it sets it as the target of your query in the main query editor panel.
   c) Type a query in the editor panel and click the run icon ▶ to run the query.

      Note: Use the Impala language reference to get information about syntax in addition to the SQL auto-complete feature that is built in. To view the language reference, click the book icon to the right of the query editor panel.

4. To run Hive queries:
   a) On the **Overview** page under Virtual Warehouses, click on the Hue button.
      The Hive query editor is displayed:

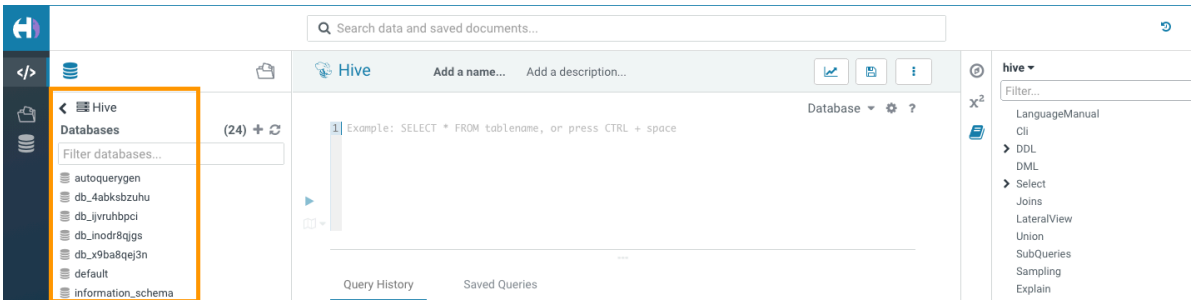

   b) Click a database to view the tables it contains.
      When you click a database, it sets it as the target of your query in the main query editor panel.
   c) Type a query in the editor panel and click the run icon ▶ to run the query.

      Note: Use the Hive language reference to get information about syntax in addition to the SQL auto-complete feature that is built in. To view the language reference, click the book icon to the right of the query editor panel.

Viewing Hive query details

You can search Hive query history, compare two queries, download debug bundles for troubleshooting, and view query details, a graphical representation of the query execution plan, and DAG information on the Job Browser page in Hue.

Viewing Hive query history

The Queries tab on the Job Browser page in Hue displays all the queries that were run on all Hive Virtual Warehouses within a Database Catalog from various query interfaces, such as Beeline, Hive Warehouse Connector (HWC), Tableau, Hue, and other JDBC BI clients and tools.

About this task

Only Query Processor Administrators can view historical queries of all users to monitor resource utilization and control costs from the Hue Job Browser. Non-admin users can view only their queries.

Queries are retained in the backend database for 30 days by default, after which they are cleaned up. You can change the clean-up interval from the Database Catalog configurations.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Click Queries.

   The Hive queries that were run for the past seven days are displayed. You can select the time period for which you want to view the historical data.

   You can also filter queries by their status.

Related Information

Adding Query Store Administrator users in Cloudera Data Warehouse

Viewing Hive query information

The Query Info tab provides information such as, the Hive query ID, the user who executed the query, the start time, the end time, the total time taken to execute the query, the tables that were read and written, application ID, Directed Acyclic Graph (DAG) IDs, session ID, LLAP app ID, thread ID, and the queue against which the query was run.

Procedure

1. Go to the Cloudera Data Warehouseweb interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.

3. Go to the **Queries** tab and click on the query for which you want to view the query details.

   The following image shows the **Query Info** tab on the Hue web interface:

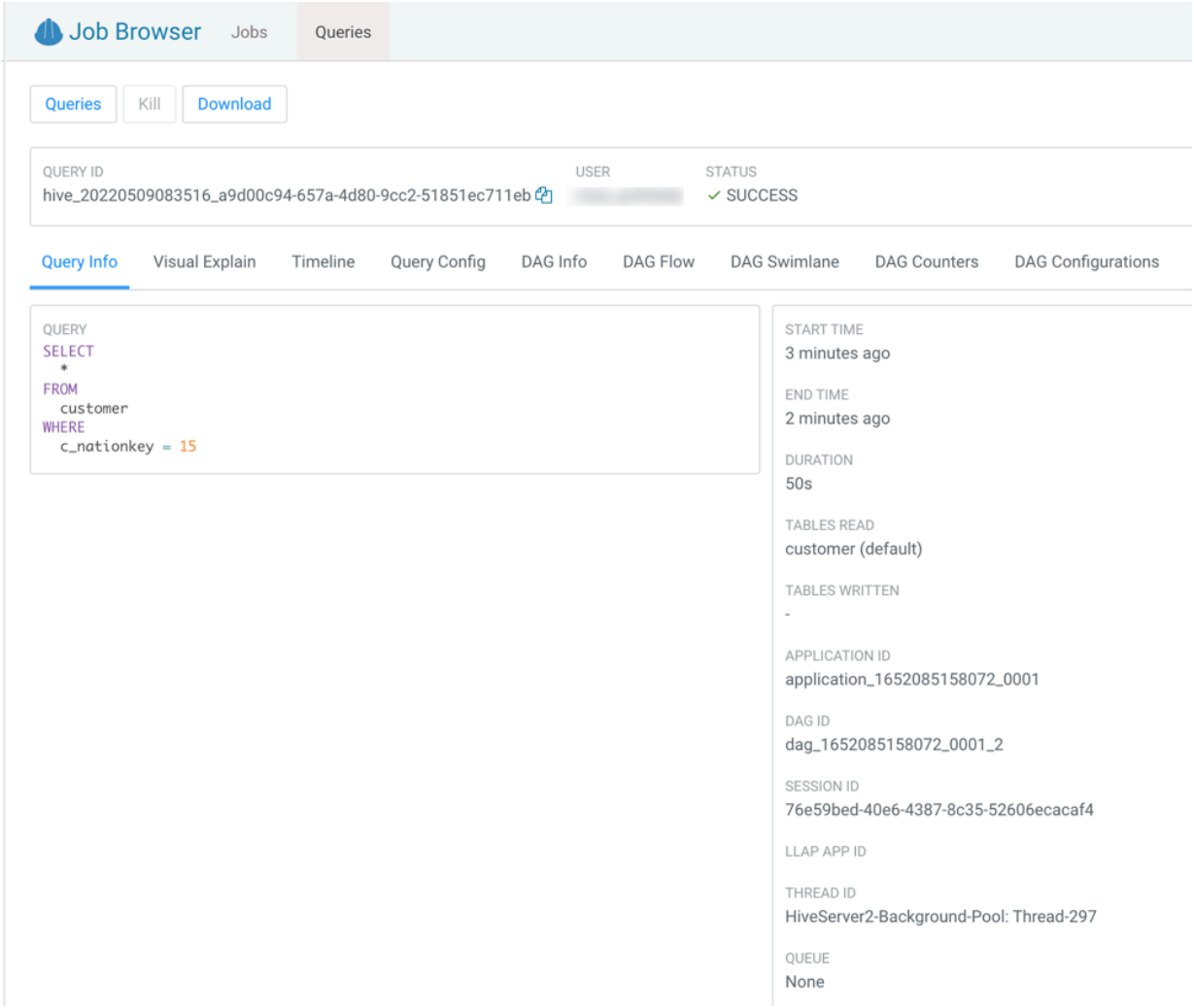

Viewing explain plan for a Hive query

The Visual Explain feature provides a graphical representation of the query execution plan. The Explain plan is read from right to left. It provides details about every stage of query execution.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Queries** tab and click on the query for which you want to view the query details.

4. Click on Visual Explain.

   The following image shows the **Visual Explain** tab on the Hue web interface:

5. (Optional) Click to download the query explain plan in JSON format.

Viewing Hive query timeline

The Timeline tab provides a visual representation of Hive performance logs and shows the time taken by each stage of the query execution.

About this task

Following are the stages in which a query is executed:

- Pre-execution and DAG construction: It is the first phase of query execution and is executed on the Hive engine. It constitutes the time taken to compile, parse, and build the Directed Acyclic Graph (DAG) for the next phase of the query execution.
- DAG submission: It is the second phase in which the DAG that was generated in Hive is submitted to the Tez engine for execution.
- DAG runtime: It shows the time taken by the Tez engine to execute the DAG.
- Post-execution: It is the last phase of query execution in which the files in S3/ABFS are moved or renamed.

Duration data about each phase are distilled into more granular metrics based on query execution logs.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Queries** tab and click on the query for which you want to view the query details.
4. Click on Timeline.

   The following image shows the **Timeline** tab on the Hue web interface:

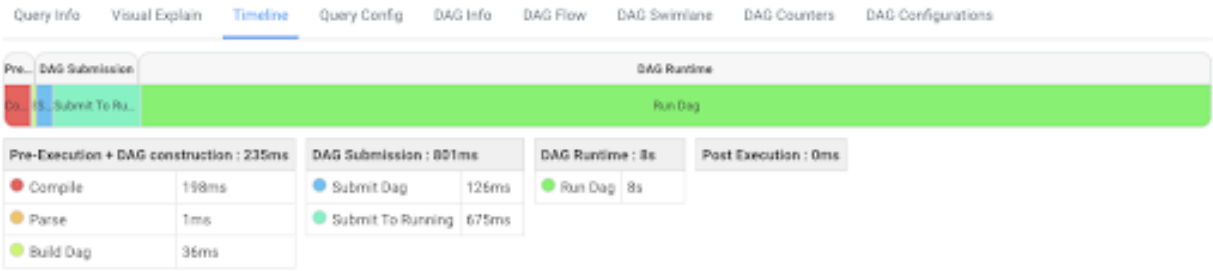

Viewing configurations for a Hive query

The Query Config tab provides the configuration properties and settings that are used in a Hive query. You can use this tab to verify that configuration property values align with your expectations.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.
   The **Job Browser** page is displayed.
3. Go to the **Queries** tab and click on the query for which you want to view the query details.
4. Click on Query Config.
   The following image shows the **Query Config** tab on the Hue web interface:

Query Info | Visual Explain | Timeline | Query Config | DAG Info | DAG Flow | DAG Swimlane

Config Name	Config Value
hadoop.security.group.mapping.ldap.posix.attr...	uidNumber
dfs.block.invalidate.limit	1000
yarn.admin.acl	*
hive.repl.dump.metadata.only.for.external.table	true
hive.exec.stagingdir	.hive-staging
hive.druid.rollup	true
yarn.federation.enabled	false
yarn.app.mapreduce.am.job.committer.cancel-...	60000
hive.druid.broker.address.default	localhost:8082
dfs.disk.balancer.max.disk.throughputInMBper...	10
dfs.qjournal.select-input-streams.timeout.ms	20000
hive.llap.io.orc.time.counters	true
hive.repl.retain.prev.dump.dir	false
hive.vectorized.execution.mapjoin.native.fast....	true
dfs.provided.aliasmap.inmemory.leveldb.dir	/tmp
yarn.nodemanager.process-kill-wait.ms	5000
yarn.minicluster.use-rpc	false
io.map.index.interval	128

Viewing DAG information for a Hive query

Directed Acyclic Graph (DAG) is created by the Hive engine every time you query the Hive Virtual Warehouse. The Hive SQL queries are compiled and converted into a Tez execution graph also known as a DAG. DAG is a collection of vertices where each vertex executes a fragment of the query or script. Hue provides a web interface to view detailed information about DAGs.

About this task

Directed connections between vertices determine the order in which they are executed. For example, the vertex to read a table must be run before a filter can be applied to the rows of that table. As another example, consider a vertex that reads a user table that is very large and distributed across multiple computers and multiple racks. Reading the table is achieved by running many tasks in parallel.

Important: The DAG information tabs (**DAG Info**, **DAG Flow**, **DAG Swimlane**, **DAG Counters**, **DAG Configurations**) are displayed only if the Tez engine is used for query execution. The Tez engine is typically utilized for complex queries.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Queries** tab and click on the query for which you want to view the query details.
4. Click DAG Info to see the DAG ID, DAG name, the status of the query, the time taken to execute the DAG, start time, and end time.

   The following image shows the **DAG Info** tab on the Hue web interface:

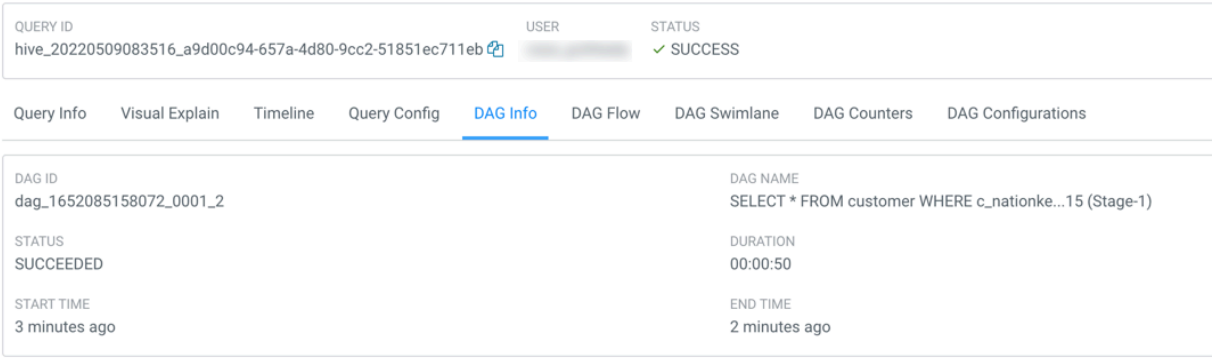

The following table lists and describes the status of the Tez job:

Status	Description
Submitted	The DAG is submitted to Tez but is not running
Running	The DAG is currently running
Succeeded	The DAG was completed successfully
Failed	The DAG failed to complete successfully
Killed	The DAG was stopped manually
Error	An internal error occurred when executing the DAG

5. Click DAG Flow to see the DAG in the form of a flowchart.

   You can gain insight into the complexity and the progress of executing jobs, and investigate the vertices that have failures or are taking a long time to complete.

   The following image shows the **DAG Flow** tab on the Hue web interface::

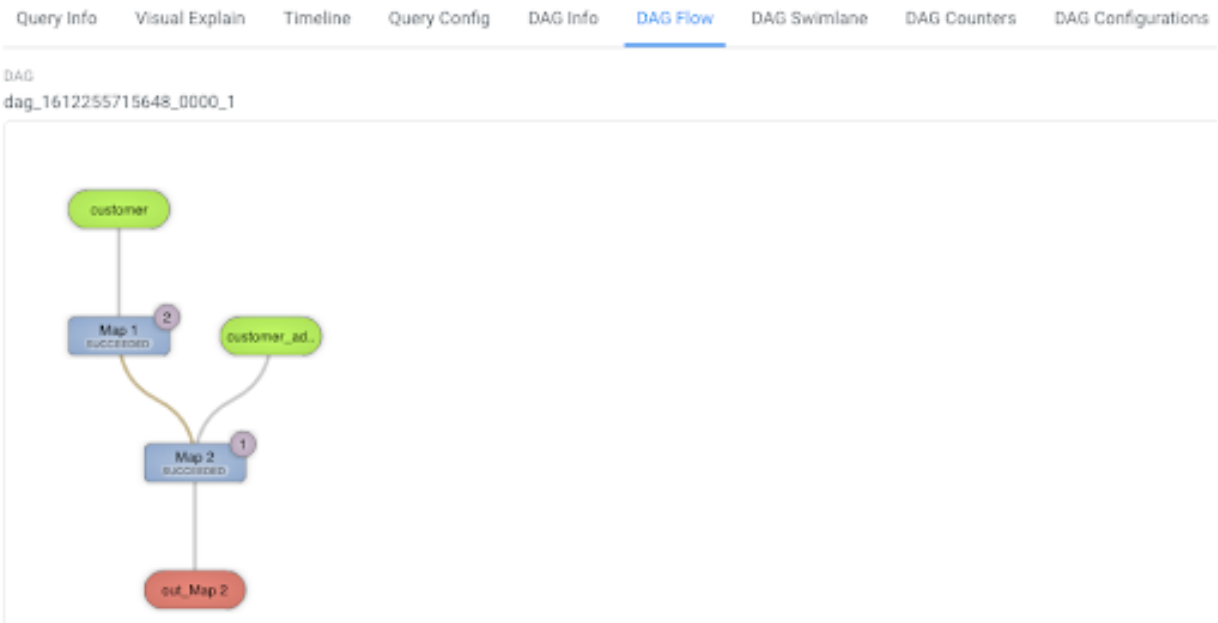

   Here, the input to vertices Map 1 and Map 2 are the tables displayed in green boxes. Next, Map 2 depends on the result set generated by Map 1. Map 2 is the last vertex in the DAG flow and after it completes its execution, the query output is written to a file in a filesystem such as S3 or ABFS.

   There are a few options to change the layout of the DAG flow. You can hide the input and the output nodes to view only the task vertices by clicking the Toggle source/sink visibility button. You can switch between the horizontal and vertical orientation by clicking the Toggle orientation button.

6. Click DAG Swimlane to see the DAG of the vertices against time.

   Each mapping and reducing task is a vertex. Each horizontal bar of the swimlane represents the total time taken by the vertex to complete the execution. The vertical lines indicate the time when the vertex was initialized, the time when the vertex started, the time when the first task started, the time when the last task was completed, and the time when the vertex finished its execution. When you mouse over the vertical line, the bubble displays the stage of the vertex execution and provides a timestamp. The vertical lines connecting two vertices denote the dependency of a vertex on another vertex.

   The following image shows the **DAG Swimlane** tab on the Hue web interface:

   In this example, Map 1 depends on the results of Map 5. Map 1 will finish its execution only when Map 5 finishes its execution successfully. Similarly, Reducer 2 depends on Map 1 to complete its execution.

   The consolidated timeline shows the percentage of time each vertex took to complete executing.

7. Click DAG Counters to see details such as the number of bytes read and written, number of tasks that initiated and ran successfully, amount of CPU and memory consumed, and so on.

   The **DAG Counters** tab provides a way to measure the progress or the number of operations that occur within a generated DAG. Counters are used to gather statistics for quality control purposes or problem diagnosis.

   The following image shows the **DAG Counters** tab on the Hue web interface:

Query Info | Visual Explain | Timeline | Query Config | DAG Info | DAG Flow | DAG Swimlane | DAG Counters | DAG Configurations

Group Name	Counter Name	DAG : dag_1652085158072_0001_2
org.apache.tez.common.counters.DAGCounter	NUM_SUCCEEDED_TASKS	54
org.apache.tez.common.counters.DAGCounter	TOTAL_LAUNCHED_TASKS	54
org.apache.tez.common.counters.DAGCounter	DATA_LOCAL_TASKS	54
org.apache.tez.common.counters.DAGCounter	AM_CPU_MILLISECONDS	5890
org.apache.tez.common.counters.DAGCounter	AM_GC_TIME_MILLIS	22
org.apache.tez.common.counters.FileSystem...	FILE_BYTES_WRITTEN	1074416910
org.apache.tez.common.counters.FileSystem...	S3A_BYTES_READ	6742239795
org.apache.tez.common.counters.FileSystem...	S3A_READ_OPS	1124
org.apache.tez.common.counters.TaskCounter	TASK_DURATION_MILLIS	759357
org.apache.tez.common.counters.TaskCounter	INPUT_RECORDS_PROCESSED	146519
org.apache.tez.common.counters.TaskCounter	INPUT_SPLIT_LENGTH_BYTES	12387638515
HIVE	CREATED_FILES	36
HIVE	RECORDS_IN_Map_1	150000000
HIVE	RECORDS_OUT_0	6003115
HIVE	RECORDS_OUT_OPERATOR_FIL_5	6003115
HIVE	RECORDS_OUT_OPERATOR_FS_7	6003115
HIVE	RECORDS_OUT_OPERATOR_SEL_6	6003115
HIVE	RECORDS_OUT_OPERATOR_TS_0	150000000

8. Click DAG Configurations to see the Tez configuration details for a query that has a DAG associated with it.
   The following image shows the **DAG Configurations** tab on the Hue web interface:

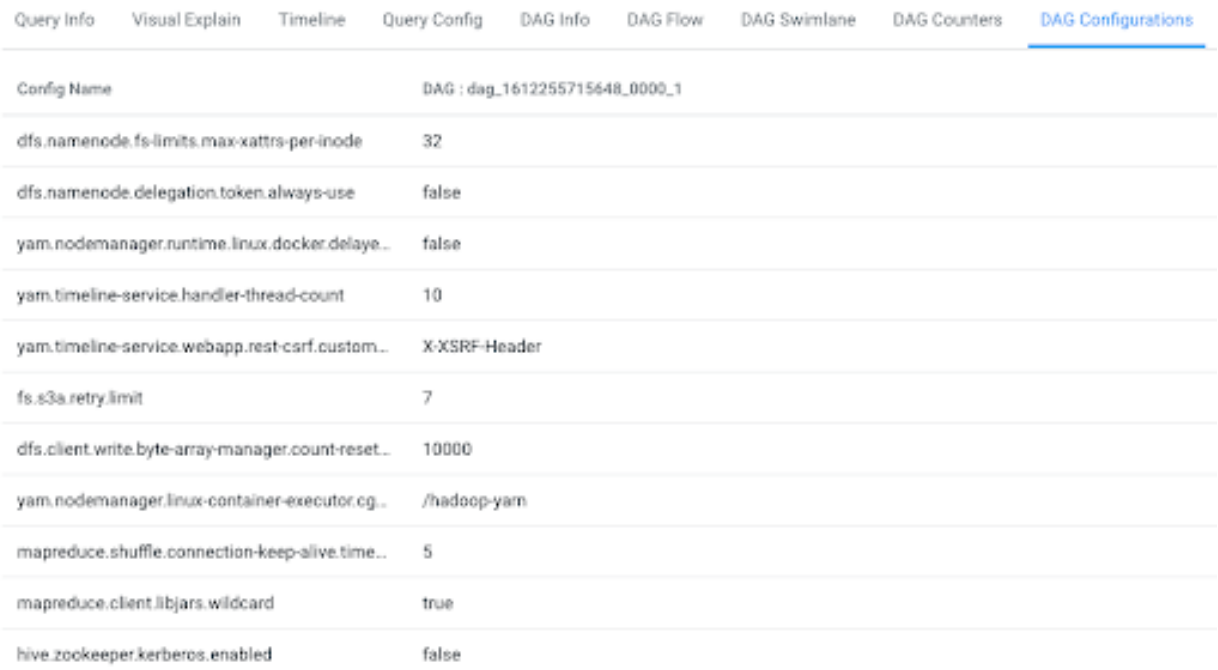

Viewing Impala query details

You can view Impala query details, query plan, execution summary, and query metrics on the new Impala Queries tab on the Job Browser page in Hue, and use this information to tune and optimize your queries.

Viewing Impala query history

The Impala Queries tab on the Job Browser page in Hue displays all the queries that were run on all Impala Virtual Warehouses within a Database Catalog from various query interfaces, such as Impala-shell, Impyla, Hue, and other JDBC BI clients and tools.

About this task

Only Query Processor Administrators can view historical queries of all users to monitor resource utilization and control costs from the Hue Job Browser. Non-admin users can view only their queries.

Queries are retained in the backend database for 30 days by default, after which they are cleaned up. You can change the clean-up interval from the Database Catalog configurations.

Note: Impala queries may take up to 25 minutes to appear on the **Impala Queries** tab after they are run. This is a known limitation in Cloudera Data Warehouse.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.
   The **Job Browser** page is displayed.

3. Click Queries.

   The Hive queries that were run for the past seven days are displayed. You can select the time period for which you want to view the historical data.

   You can also search using the query ID, sort queries by various parameters such as duration, peak memory, and so on, and filter queries by their status.

Viewing Impala query information

The Query Info tab in Hue provides information such as, the Impala query ID, the user who executed the query, the start time, the end time, the total time taken to execute the query, the coordinator that received the query, CPU time, rows produced, peak memory, and HDFS bytes read.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Impala Queries** tab and click on the query for which you want to view the query details.

   The following image shows the **Query Info** tab on the Hue web interface:

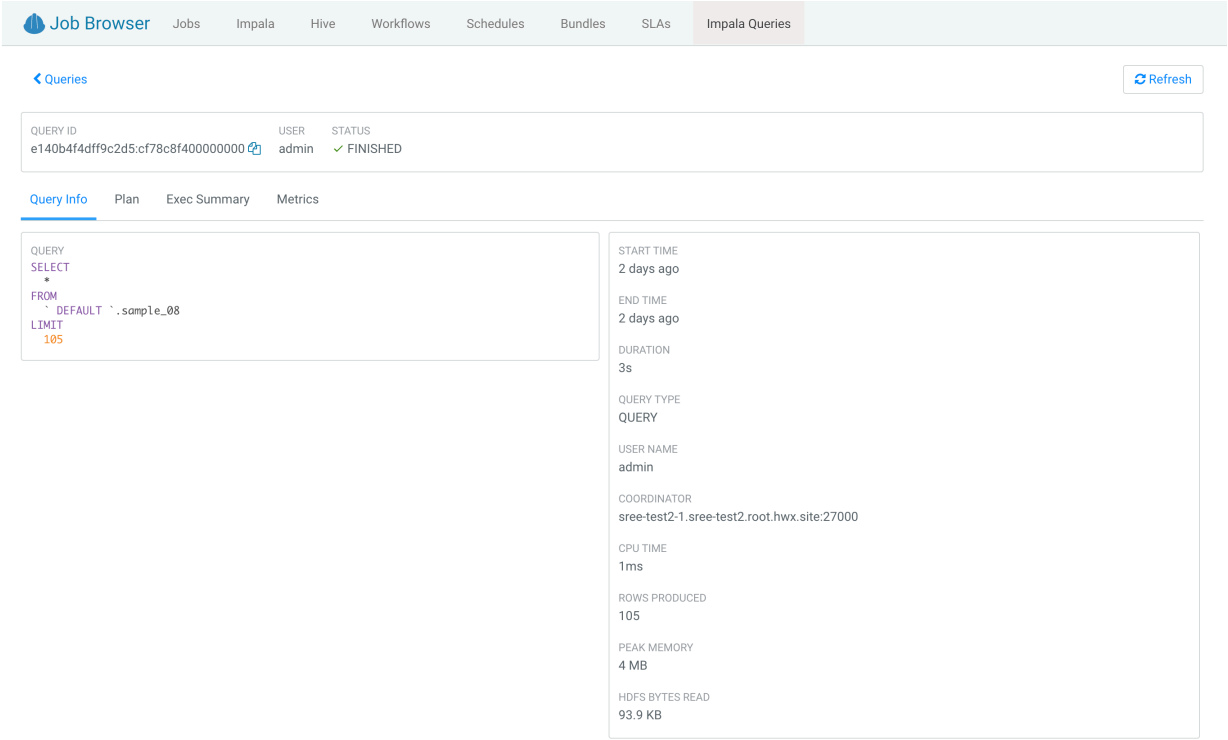

Viewing the Impala query execution plan

The query execution plan in Hue provides details on how the query will be executed, the operators involved, and other information before the query is submitted to the Impala engine.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.

3. Go to the **Impala Queries** tab and click on the query for which you want to view the execution plan.

   The following image shows the **Plan** tab on the Hue web interface:

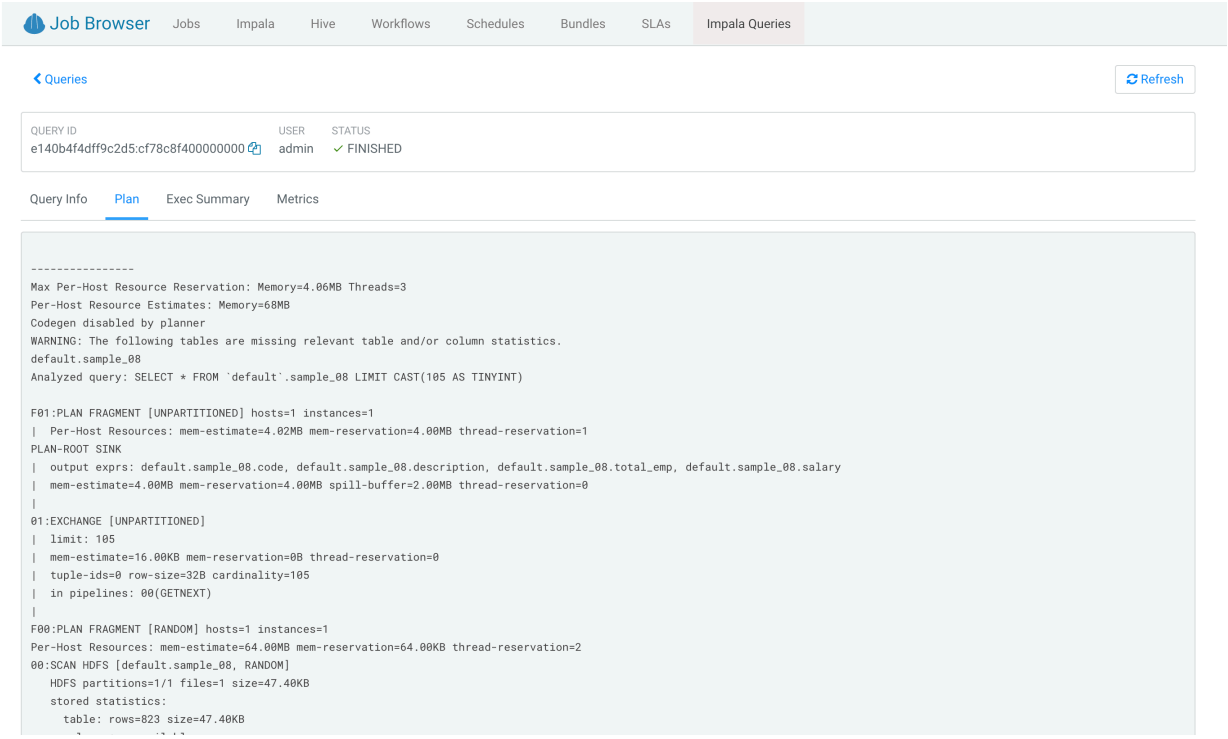

Viewing the Impala query metrics

You can view detailed, aggregated metrics for various counters such as hdfs_bytes_read, memory_per_node_peak, thread_cpu_time, and so on, on the Metrics tab in Hue.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.

3. Go to the **Impala Queries** tab and click on the query for which you want to view the query metrics.

   The following image shows the **Metrics** tab on the Hue web interface:

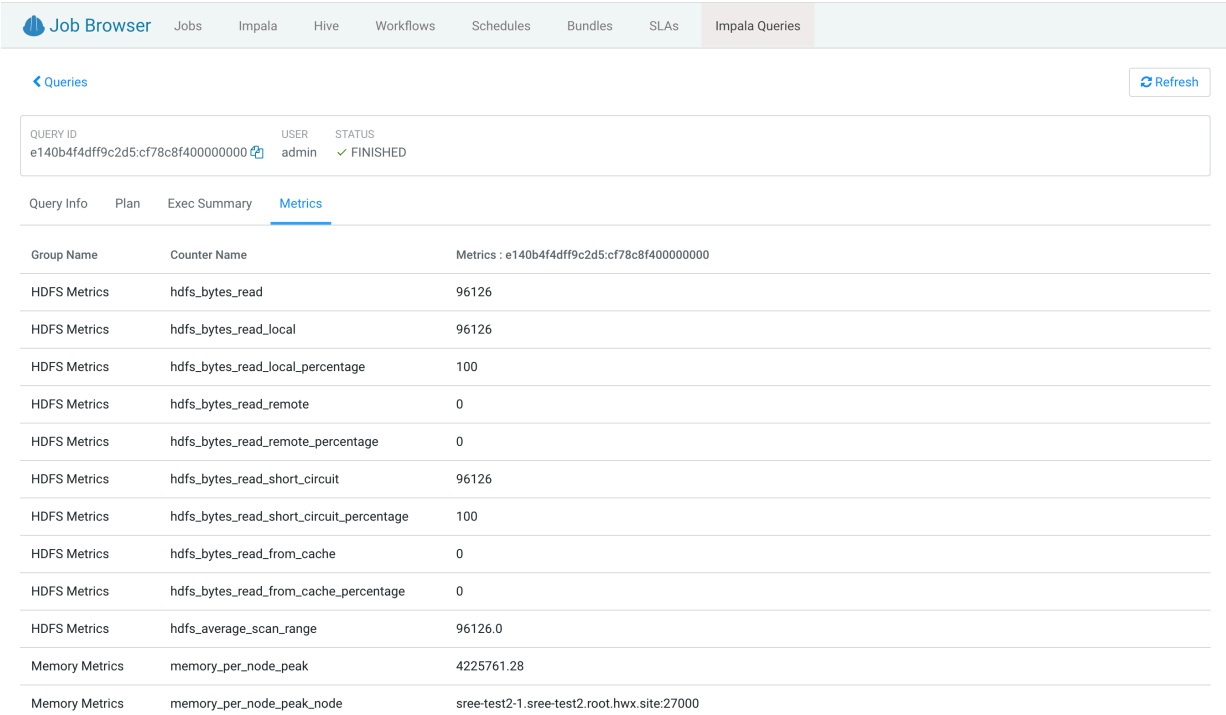

Group Name	Counter Name	Metrics : e140b4f4dff9c2d5:cf78c8f400000000
HDFS Metrics	hdfs_bytes_read	96126
HDFS Metrics	hdfs_bytes_read_local	96126
HDFS Metrics	hdfs_bytes_read_local_percentage	100
HDFS Metrics	hdfs_bytes_read_remote	0
HDFS Metrics	hdfs_bytes_read_remote_percentage	0
HDFS Metrics	hdfs_bytes_read_short_circuit	96126
HDFS Metrics	hdfs_bytes_read_short_circuit_percentage	100
HDFS Metrics	hdfs_bytes_read_from_cache	0
HDFS Metrics	hdfs_bytes_read_from_cache_percentage	0
HDFS Metrics	hdfs_average_scan_range	96126.0
Memory Metrics	memory_per_node_peak	4225761.28
Memory Metrics	memory_per_node_peak_node	sree-test2-1.sree-test2.root.hwx.site:27000

Terminating Hive queries

If a query is running for longer than expected, or you have accidentally triggered it, then you can stop the query to free up the resources. Hue also allows you to stop multiple queries at once.

About this task

Note: This feature is available only for Hive queries. Only admin users or Hue superusers can stop running queries.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Queries** tab.

   A list of queries that were run is displayed.
4. Select the queries that you want to stop and click Kill.

Comparing Hive and Impala queries in Hue

You can compare two queries to know how each query is performing in terms of speed and cost-effectiveness. Hue compares various aspects of the two queries, based on which you can identify what changed between the executions of those two queries, and you can debug performance-related issues between different runs of the same query.

About this task

The query comparison report provides you a detailed side-by-side comparison of your queries.

For Hive queries, it includes recommendations for optimizing each query, metadata about the queries, visual explain for each query, query timeline, query configuration, Directed Acyclic Graph (DAG) information, DAG flows, DAG swimlanes, DAG counters, and DAG configurations.

For Impala queries, the query comparison report includes query details, execution plan details, and the aggregated metrics for both the queries and provides a variance between the two.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Virtual Warehouse.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Queries** tab.

   A list of queries that were run is displayed.

4. Select the two queries you want to compare and click Compare.

   Query comparison report for Hive queries:

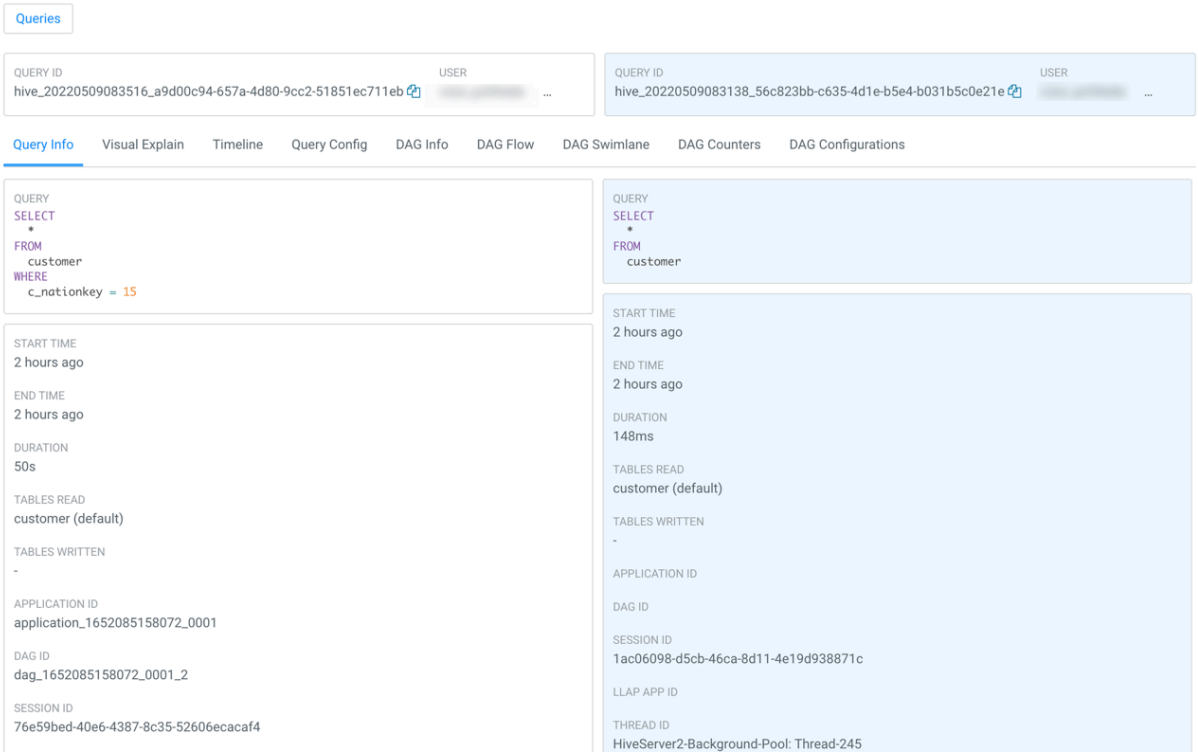

   Query comparison report for Impala queries:

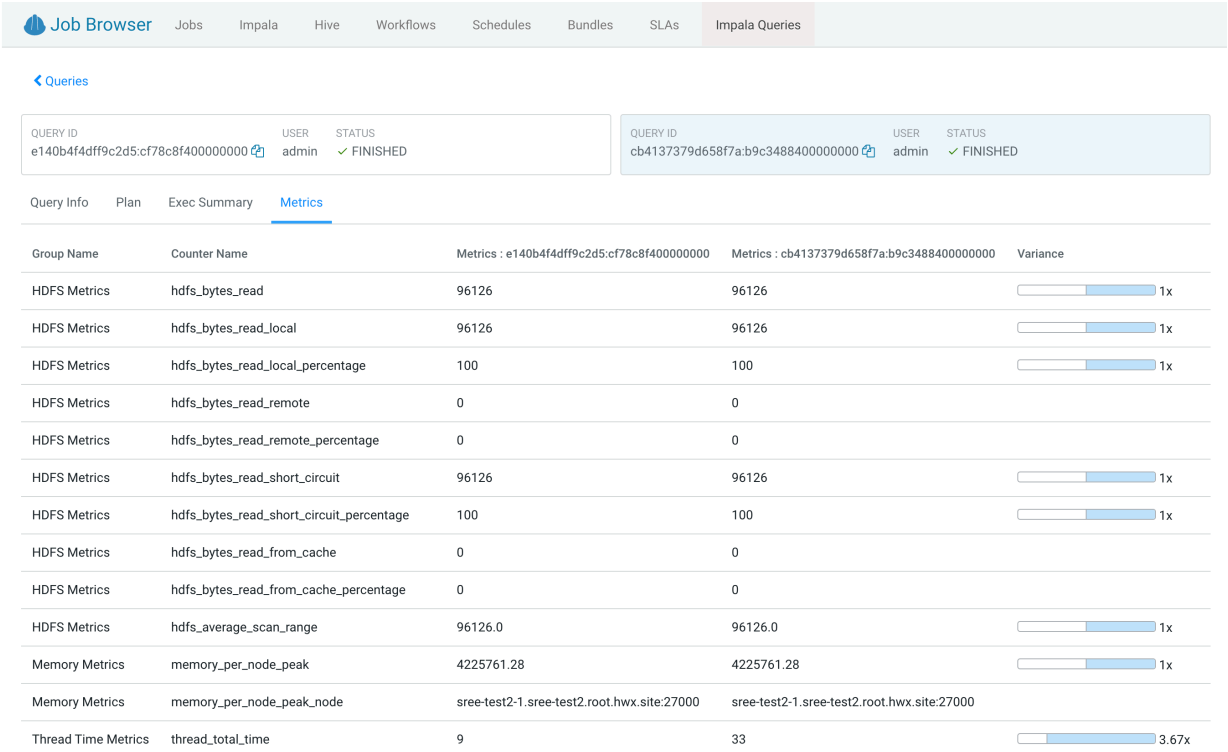

Group Name	Counter Name	Metrics : e140b4f4dff9c2d5:cf78c8f400000000	Metrics : cb4137379d658f7a:b9c3488400000000	Variance
HDFS Metrics	hdfs_bytes_read	96126	96126	1x
HDFS Metrics	hdfs_bytes_read_local	96126	96126	1x
HDFS Metrics	hdfs_bytes_read_local_percentage	100	100	1x
HDFS Metrics	hdfs_bytes_read_remote	0	0	
HDFS Metrics	hdfs_bytes_read_remote_percentage	0	0	
HDFS Metrics	hdfs_bytes_read_short_circuit	96126	96126	1x
HDFS Metrics	hdfs_bytes_read_short_circuit_percentage	100	100	1x
HDFS Metrics	hdfs_bytes_read_from_cache	0	0	
HDFS Metrics	hdfs_bytes_read_from_cache_percentage	0	0	
HDFS Metrics	hdfs_average_scan_range	96126.0	96126.0	1x
Memory Metrics	memory_per_node_peak	4225761.28	4225761.28	1x
Memory Metrics	memory_per_node_peak_node	sree-test2-1.sree-test2.root.hwx.site:27000	sree-test2-1.sree-test2.root.hwx.site:27000	
Thread Time Metrics	thread_total_time	9	33	3.67x

How to run a stored procedure from Hue in Cloudera Data WarehouseHow to run a stored procedure from Hue in Cloudera Data Hub

HPL/SQL allows you to implement business logic using variables, expressions, flow-of-control statements, and iterations. HPL/SQL makes SQL-on-Hadoop more dynamic. You can leverage your existing procedural SQL skills, and use functions and statements to make your typical ETL development more productive. In Cloudera Data Warehouse, Hue provides a smart interface to run stored procedures.

Note: This feature is available only for Hive queries.

To run stored procedures from Hue, create a Hive Virtual Warehouse in Cloudera Data Warehouse and enable the hplsql option in the hue-safety-valve field.

The following example creates a procedure and returns records by passing a cursor:

```
print 'Hello world';/
CREATE PROCEDURE greet(name STRING)
BEGIN
  PRINT 'Hello ' || name;
END;/
CREATE PROCEDURE even(cur OUT SYS_REFCURSOR)
BEGIN
  OPEN cur FOR
SELECT n FROM NUMBERS
WHERE MOD(n, 2) == 0;
END;/
CREATE PROCEDURE set_message(IN name STRING, OUT result STRING)
BEGIN
 SET result = 'Hello, ' || name || '!';
END;
-- Call the procedure and print the results
DECLARE str STRING;
CALL set_message('world', str);
PRINT str;
```

Attention: In the hplsql mode, you must terminate the commands using the forward slash (/). The semicolon (;) is used throughout procedure declarations and can no longer be relied upon to terminate a query in the editor.

Note: HPL/SQL does not support all types of Hive statements, such as JOIN or EXPLAIN. Refer to the HPL/SQL Reference for more information.

Related Information

Enabling stored procedures for Hive in Cloudera Data Warehouse

Enabling stored procedures for Hive in Cloudera Data Warehouse

To create, edit, and drop procedures and functions that are written in Hive Hybrid Procedural SQL (HPL/SQL) using the Hue query editor in Cloudera Data Warehouse, you must enable the hplsql option in the hue-safety-valve field.

About this task

Important: Hue enables you to switch between Hive and HPL/SQL interpreters. By default, the regular Hive interpreter is enabled when you create a Hive Virtual Warehouse. To enable the HPL/SQL interpreter, you must update the configuration in the hue-safety-valve field in your Hive Virtual Warehouse. However, updating the hue-safety-valve overrides the default configuration. Therefore, to use both Hive and HPL/SQL interpreters, you must enable both by updating the configuration in the hue-safety-valve field.

Procedure

1. Log in to the Cloudera Data Warehouse service as an administrator.
2. Go to Virtual Warehouse Edit CONFIGURATIONS Hue and select hue-safety-valve from the Configuration files drop-down list.
3. Add the following lines in the hue-safety-valve:

```
[notebook]
  [[interpreters]]
    [[[hive]]]
      name=Hive
      interface=hiveserver2
    [[[hplsql]]]
      name=Hplsql
      interface=hiveserver2
```

4. Click APPLY.
5. Restart the Virtual Warehouse.

Enabling the SQL editor autocompleter

Autocompleter provides finely tuned SQL suggestions for Hive and Impala dialects while you enter queries into the editor window. See Brand new Autocompleter for Hive and Impala in the Hue blog.

About this task

Autocompleter is enabled by default. To manually enable or disable it, open the editor configuration panel and edit settings as follows:

Procedure

1. Log in to Hue and go to either the Hive or Impala editor.
2. Place your cursor in the editor window and then use one of the following keyboard shortcuts to open the editor configuration panel:
   - On a Mac system, use the Command key followed by a hyphen and then a comma:

     Command-,
   - On a Windows system, use the Ctrl key followed by a hyphen and then a comma:

     Ctrl-,

     Tip: Type a question mark (?) anywhere but in the active editor window to open a menu of editor keyboard shortcuts.

3. To enable autocompletion, check the box adjacent to Enable Autocompleter. When you check Enable Autocompleter, Enable Live Autocompletion is automatically enabled as well. Place your cursor in the editor window to close the configuration panel.

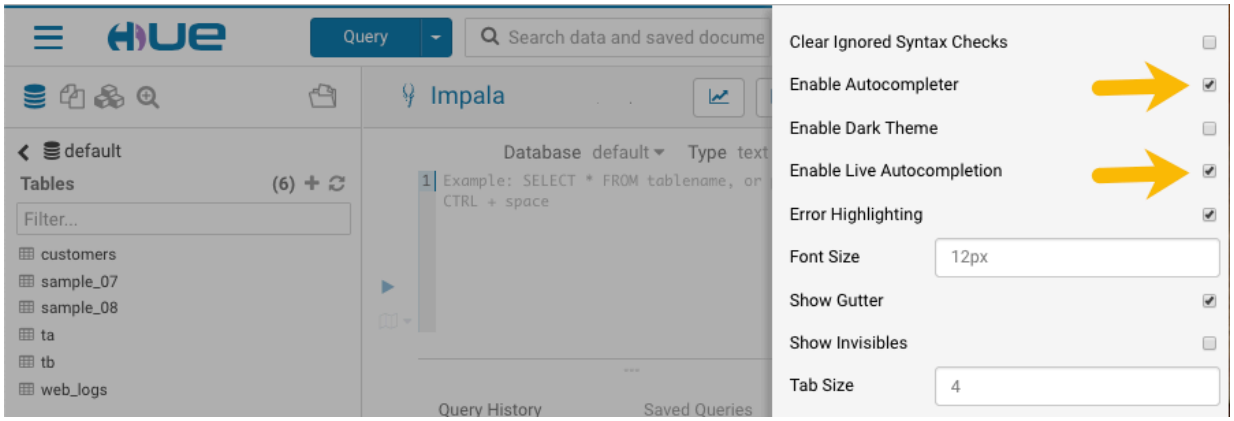

4. To disable autocompletion:
   - Uncheck Enable Live Autocompletion but leave Enable Autocompleter checked, and then place your cursor in the editor window to close the configuration panel. This disables live autocompletion, but if you want to use autocompletion while building your queries in the editor, enter the following key stroke sequence to activate autocompletion: Ctrl + Space Key
   - Uncheck both Enable Autocompleter and Enable Live Autocompletion, and then click in the editor to close the configuration panel. This disables all autocompletion functionality.

Using governance-based data discovery

Hue can use the metadata tagging, indexing, and search features available in Apache Atlas data management. After integrating Hue with Atlas, classifications and indexed entities can be accessed and viewed in Hue. This topic shows you how to use metadata classifications in Hue.

Integration between Hue and Atlas is enabled by default, but if your administrator has disabled it, it must be re-enabled before you can use governance-based data discovery.

In Cloudera Data Warehouse, you can only view tags that are created in Atlas in Hue. You must create tags in Atlas.

Searching metadata tags

The SQL Editor in Hue provides a search text box where you can search on the metadata tags or classifications that are associated with your databases, tables, and columns.

About this task

You can search for tags or classifications in either the Hive or the Impala editors.

Note: On clusters that use Apache Ranger for role-based access control, the Search mechanism does not display counts of popular values. Ranger ensures that Hue users can view only entities to which their user role (as configured and managed by Ranger) has been granted specific permissions.

Procedure

1. Go to Query Editor Impala or Hive.

2. To locate the tags or classifications in Apache Atlas, in the metadata search box located just to the right of the Query drop-down menu, type a tag: or classification: facet followed by its name. For example, type classification: wine as shown in the following image:

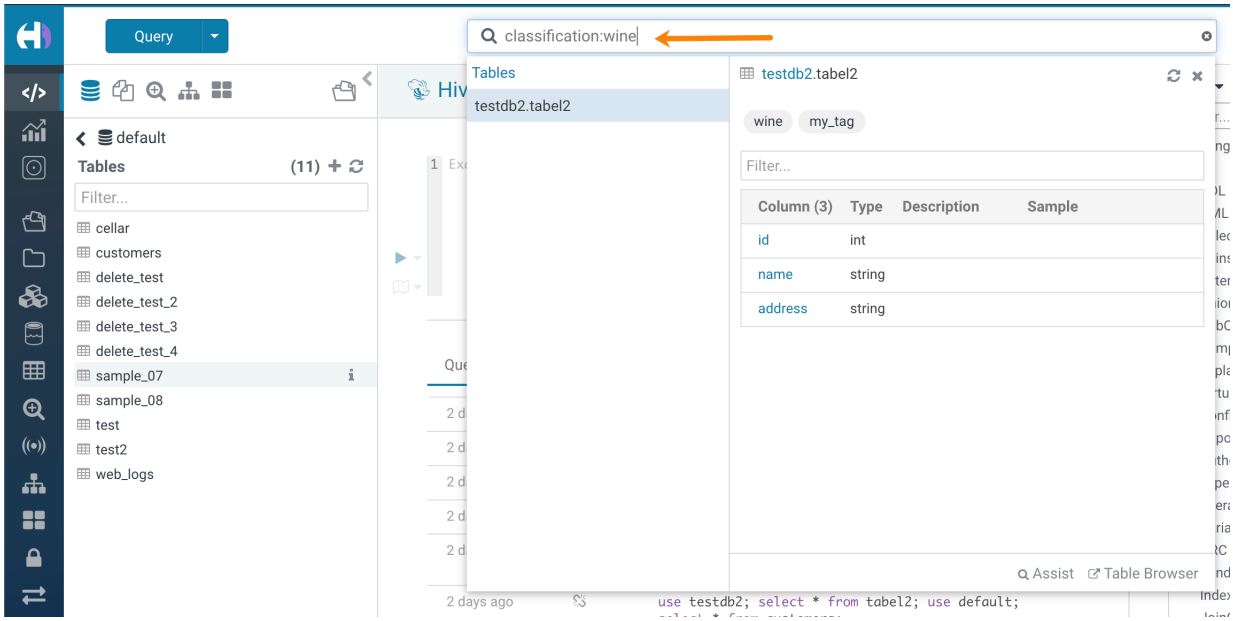

After you type the search facet and the tag or classification name in the search box, the *<DATABASE>.<TABLE>* where the tag or classification is found is returned. Click the *<DATABASE>.<TABLE>* to view the tags and classifications that have been defined for it.

Using Amazon S3 with Hue

Hue can read to and write to an Amazon S3 bucket.

Note:

In Cloudera Data Warehouse, you can browse Amazon S3 buckets from Hue in the following ways:

- With Ranger Authorization Service (RAZ)
- Without Ranger Authorization Service (RAZ)

If you have registered your Cloudera Public Cloud environment using RAZ, policies attached to the Ranger RAZ Service role at the Data Lake-level can control access to external S3 buckets. If your Virtual Warehouse predates the capability to use RAZ configurations and policies, then you must manually enable RAZ for Cloudera Data Warehouse and then configure Hue to access S3 buckets. Your Data Lake must be RAZ-enabled to manually enable RAZ exclusively for Cloudera Data Warehouse.

Related Information

Registering a RAZ-enabled AWS environment

Enabling RAZ manually in Cloudera Data Warehouse Public Cloud

Accessing S3 bucket from Hue in Cloudera Data Warehouse with RAZ

Hue offers you the capability to browse S3 buckets, upload files to S3, and create tables by importing files from S3. With Ranger Authorization (RAZ), you can grant fine-grained access to per-user home directories.

About this task

If you have enabled RAZ while registering your AWS environment with Cloudera, then Hue uses RAZ as the default mechanism for enabling the S3 File Browser. Before you can enable the S3 File Browser in Hue, you must complete the following prerequisites:

Procedure

1. Follow the instructions listed in Introduction to RAZ on AWS environments to register an AWS environment with the Enable Ranger authorization for AWS S3 option enabled. You can use the Cloudera web interface or the Cloudera CLI to complete this task.

   Note: You must enable RAZ while registering your environment with Cloudera.

2. Log in to the Cloudera Management Console as a DWAdmin or DWUser and go to the Cloudera Data Warehouse service.
3. Click ⋮ Open Ranger on your Database Catalog.
4. Create the following Ranger policies:
   a) Hadoop SQL policy (all - database, table, column, all - url).

      You must grant permissions to individual users or groups in these Ranger policies. To grant permissions to all users, you can specify {USER} in the Permission section.

   b) S3 (cm_S3) policy (Default: User Home)

      You must grant permissions to the following users in the Permissions section for the user home directory (/user/{USER}): {USER}.

      Specify the bucket name in the S3 Bucket field and the directory path in the Path field of the cm_S3 Ranger policy.

   c) S3 (cm_S3) policy (Default: user)

      You must grant permissions to the following users in the Permissions section for the root directory (/user/): hive, impala.
5. You must also grant appropriate permissions to the users in Cloudera User Management Service (UMS). For example, EnvironmentUser.

Enabling the S3 File Browser for Hue in Cloudera Data Warehouse with RAZ

The S3 File Browser in Hue is enabled by default. However, you must set the path to your S3 directory in the hue-safety-valve field to avoid a 403 error when you click on the S3 File Browser.

Procedure

1. Sign in to Cloudera Data Warehouse DWAdmin or DWUser.
2. Go to the Virtual Warehouse from which you want to access the S3 buckets and click ✎.
3. Go to CONFIGURATIONS Hue and select hue-safety-valve from the Configuration files drop-down menu.
4. Add the path to your S3 bucket under the [filebrowser] section as follows:

   ```
   [filebrowser]
   remote_storage_home=s3a://[***S3-BUCKET-NAME***]/user
   ```

   (Optional) Per-user home directories are created by default. To disable automatic user directory creation, you can add the following lines in the hue-safety-valve as follows:

   ```
   [desktop]
   [[raz]]
   autocreate_user_dir=false
   ```

5. Click APPLY.
   You should be able to view the icon for the S3 File Browser on the left assist panel on the Hue web interface.

Accessing S3 bucket from Hue in Cloudera Data Warehouse without RAZ

To enable access to S3 buckets from Hue without RAZ, you must have onboarded to Cloudera Public Cloud and must meet the requirements listed in this section.

Only Hue superusers can view and access the S3 File Browser.

Creating roles and synchronizing users to FreeIPA

You must be an EnvironmentAdmin or EnvironmentUser to browse S3 buckets and create tables by importing CSV files. You must also synchronize users to FreeIPA. This is required for users accessing the Data Lake.

About this task

To assign roles to users, see Assigning resource roles to users in the *Cloudera Management Console* documentation.

To synchronize users to FreeIPA, see Performing user sync in the *Cloudera Management Console* documentation.

Related Information

Assigning resource roles to users
Performing user sync

Adding an external S3 bucket to your Cloudera Data Warehouse environment

If you try to access an external S3 bucket from the Hue web interface without adding it to the Cloudera Data Warehouse environment, then Impala or Hive may display the “AccessDeniedException 403” exception. Make sure that your Cloudera Data Warehouse environment has access to the S3 buckets that you want to access from Hue.

About this task

When you create a Virtual Warehouse in the Cloudera Data Warehouse service, a cluster is created in your AWS account. This cluster has two buckets. One bucket is used for managed data and the other is used for external data. Access to these two buckets is controlled by AWS instance profiles.

To add read/write access to external S3 buckets that reside in the same AWS account as the Cloudera Data Warehouse service cluster or that are different from the account where the Cloudera Data Warehouse service cluster resides, see the corresponding links in the Related information section.

Procedure

1. Sign in to the Cloudera Management Console as an administrator.
2. Go to Cloudera Data Warehouse Environments and click the More… menu.
3. Search and locate the environment in which you want to add the S3 bucket and click the edit icon.
   The **Environment Details** page is displayed.
4. Specify the name of the S3 bucket you want to configure access to in the Add External S3 Bucket text box.
   If the bucket belongs to another AWS account, then select the Bucket belongs to different AWS account option.
5. Select the access mode.
   Read-only access is sufficient to import data in Hue.
6. Click Add Bucket to save the configuration.
   A success message is displayed.
7. Click APPLY to update the Cloudera Data Warehouse environment.

Tip: If you configure read only access to an external S3 bucket, there is no need to restart Virtual Warehouses. However, if you configure read/write access to an external S3 bucket, you must restart Virtual Warehouses by suspending them and starting them again.

Related Information

Adding Cloudera Data Warehouse cluster access to external S3 buckets in the same AWS account
Adding Cloudera Data Warehouse cluster access to external S3 buckets in a different AWS account
AWS instance profiles

Adding users to Hadoop SQL Ranger policies

You must grant the Hadoop SQL Ranger permissions to enable your users to access specific tables and secure your data from unauthorized access.

Procedure

1. Sign in to Cloudera Data Warehouse.
2. Click the ⋮ Open Ranger option on your Database Catalog.

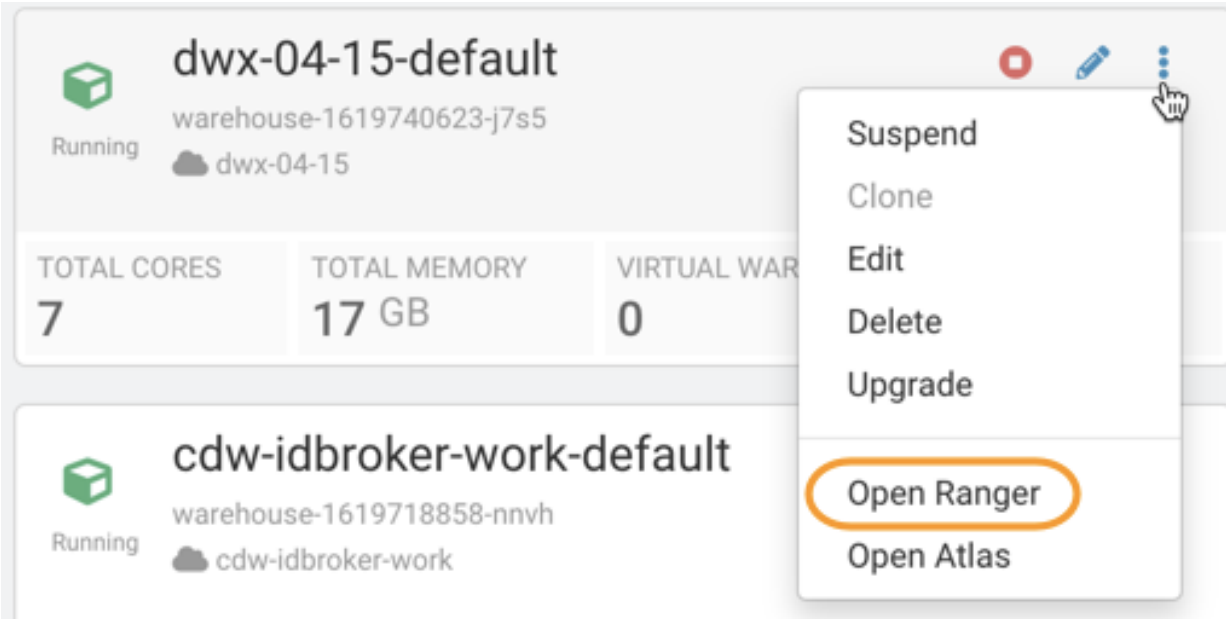

3. On the **Ranger Service Manager** page, click Hadoop SQL.
4. Select the all - url policy.

   The **Edit Policy** page is displayed.
5. Under the Add Conditions section, add the users under the Select User column and add permissions such as Create, Alter, Drop, Select, and so on from the Permissions column.

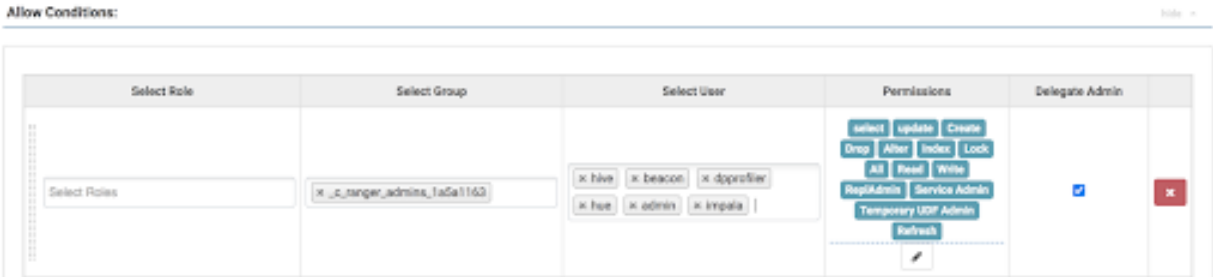

Tip: To grant permissions to all users, you can specify {USER} in the Select User column.

6. Scroll to the bottom of the page and click Save.

Enabling the S3 File Browser for Hue in Cloudera Data Warehouse without RAZ

To enable access to S3 buckets from the Hue web interface in a non-RAZ environment, you must add the AWS environment details in the hue-safety-valve configuration from your Virtual Warehouse. After enabling the S3 File Browser, you can browse the S3 buckets, create folders, and upload files from your computer, and import files to create tables.

Procedure

1. Sign in to Cloudera Data Warehouse.
2. Go to the Virtual Warehouse from which you want to access the S3 buckets and click .
3. On the **Virtual Warehouses** detail page, click the Hue tab and select hue-safety-valve from the drop-down menu.
4. Add the following configuration for Hive or Impala Virtual Warehouse in the space provided:

   For the Hive Virtual Warehouse:

   ```
   [desktop]
   # Remove the file browser from the blocked list of apps.
   # Tweak the app_blacklist property to suit your app configuration.
   app_blacklist=oozie,search,hbase,security,pig,sqoop,spark,impala

   [aws]
   [[aws_accounts]]
   [[[default]]]
   access_key_id=[***AWS-ACCESS-KEY***]
   secret_access_key=[***SECRET-ACCESS-KEY***]
   region=[***AWS-REGION***]
   [filebrowser]
   # (Optional) To set a specific home directory path:
   remote_storage_home=s3a://[***S3-BUCKET-NAME***]
   ```

   For Impala Virtual Warehouse:

   ```
   [desktop]
   # Remove the file browser from the blocked list of apps.
   # Tweak the app_blacklist property to suit your app configuration.
   app_blacklist=spark,zookeeper,hive,hbase,search,oozie,jobsub,pig,sqoop,sec
   urity

   [aws]
   [[aws_accounts]]
   [[[default]]]
   access_key_id=[***AWS-ACCESS-KEY***]
   secret_access_key=[***SECRET-ACCESS-KEY***]
   region=[***AWS-REGION***]

   [filebrowser]
   # (Optional) To set a specific home directory path:
   remote_storage_home=s3a://[***S3-BUCKET-NAME***]
   ```

5. Click APPLY.

   The S3 File Browser icon appears on the left Assist panel on the Hue web interface after the Virtual Warehouse restarts.

Creating tables by importing CSV files from AWS S3 in

You can create tables in Hue by importing CSV files stored in S3 buckets. Hue automatically detects the schema and the column types, thus helping you to create tables without using the CREATE TABLE syntax.

About this task

The maximum file size supported is three gigabytes.

(Non-RAZ deployment) Only Hue Superusers can access S3 buckets and import files to create tables. To create tables by importing files from S3, you must assign and authorize use of a specific bucket on S3 bucket for your environment. The bucket then appears like a home directory on the Hue web interface.

Procedure

1. Log in to the service.
2. On the **Overview** page, select the Virtual Warehouse in which you want to create the table and click on Hue.
3. From the left assist panel, click on Importer.
4. On the Importer screen, click .. at the end of the Path field:

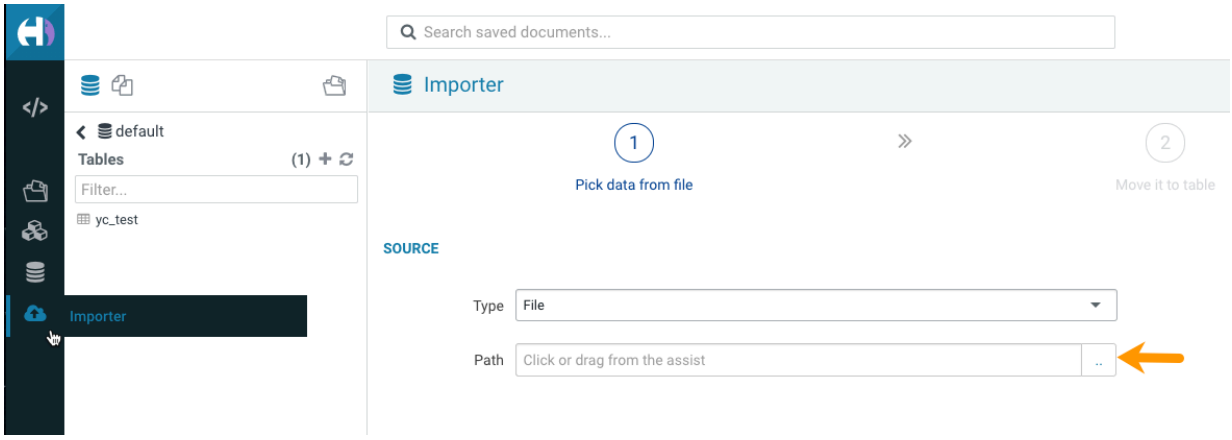

Choose a file pop-up is displayed.

5. (Non-RAZ deployment) Type s3a:// in the address text box and press enter.

   The S3 buckets associated with the environment are displayed. You can narrow down the list of results using the search option.

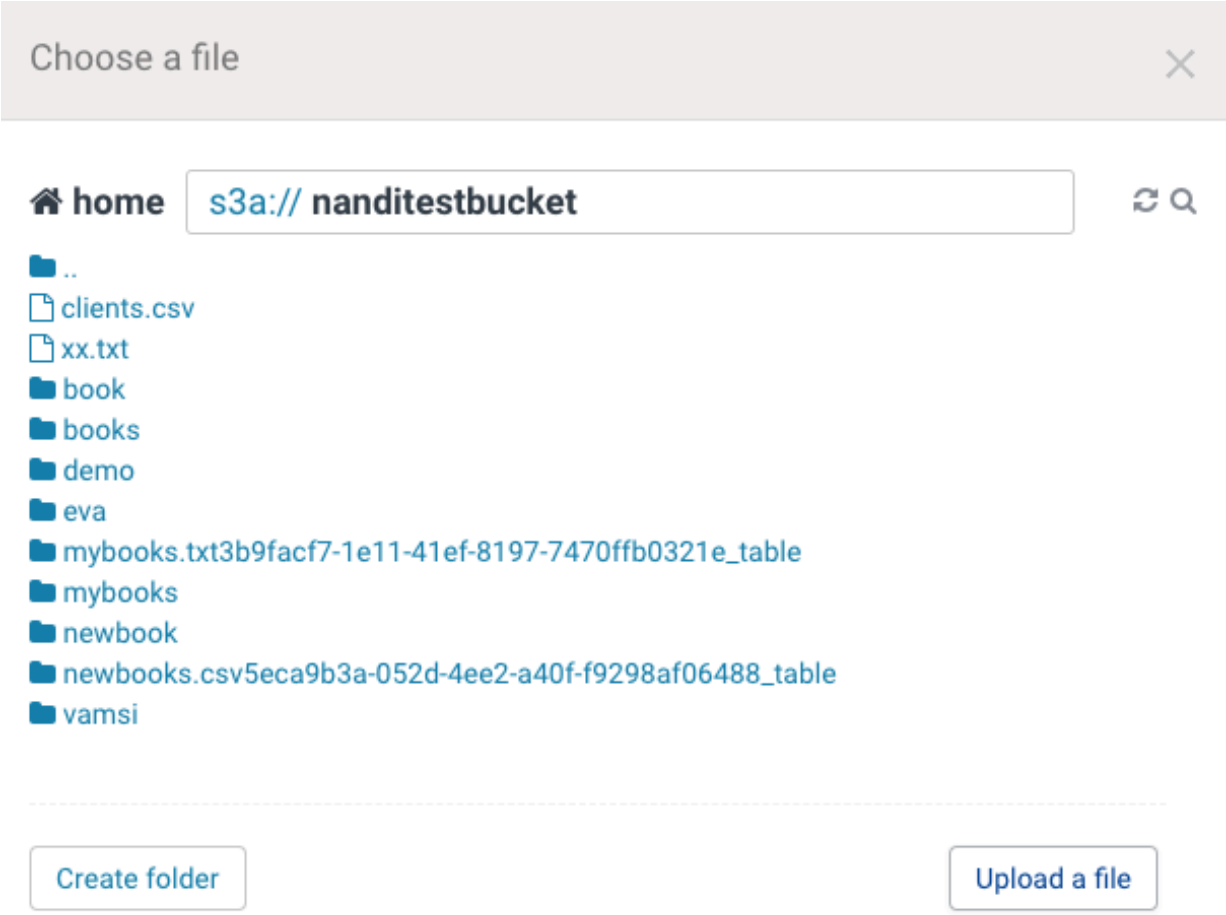

   If the file is present on your computer, then you can upload it to S3 by clicking Upload a file. To do this, you must have enabled read/write access to the S3 bucket from the environment.

6. Select the CSV file that you want to import into Hue.
   Hue displays the preview of the table along with the format:

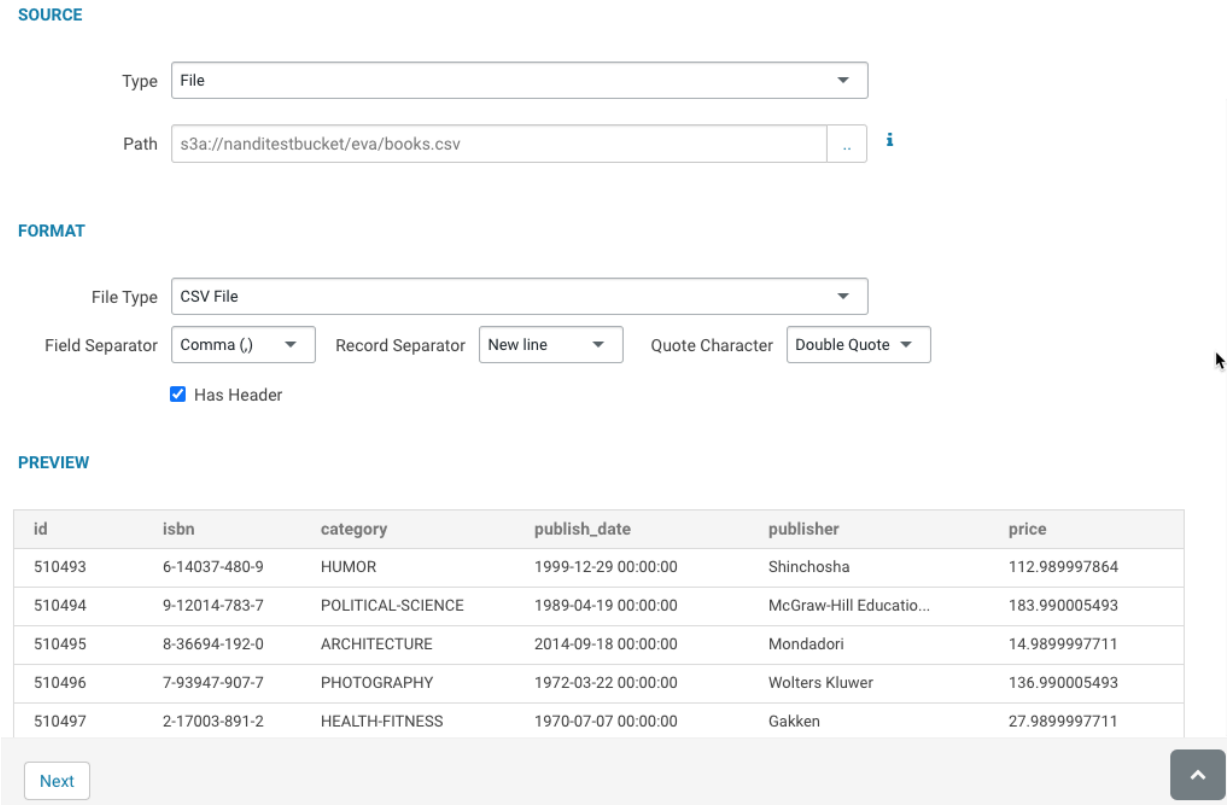

Hue automatically detects the field separator, record separator, and the quote character from the CSV file. If you want to override a specific setting, then you can change it by selecting a different value from the drop-down menu.

7. Click Next.

   On this page, you can set the table destination, partitions, and change the column data types.

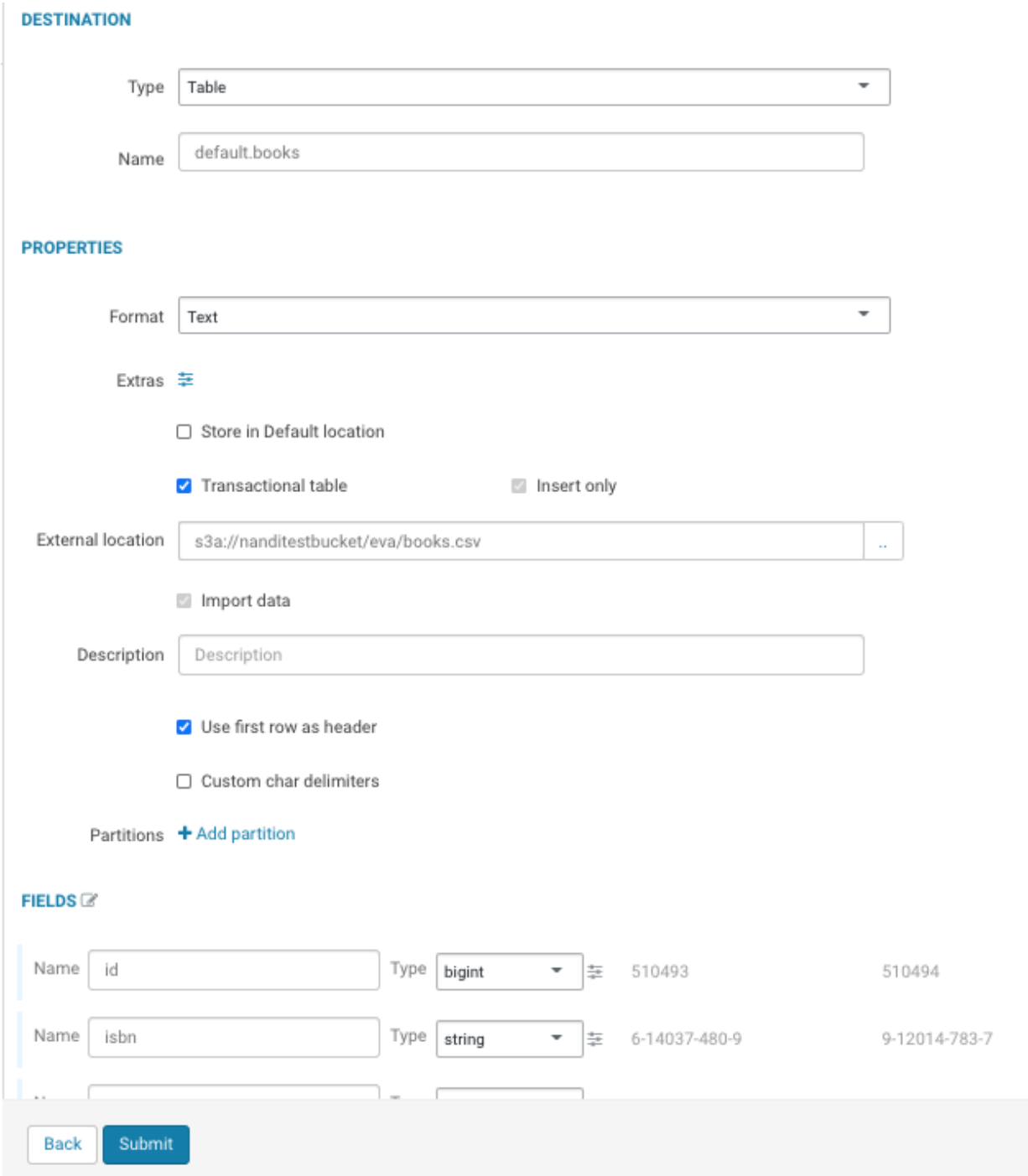

8. Verify the settings and click Submit to create the table.

   The CREATE TABLE query is triggered:

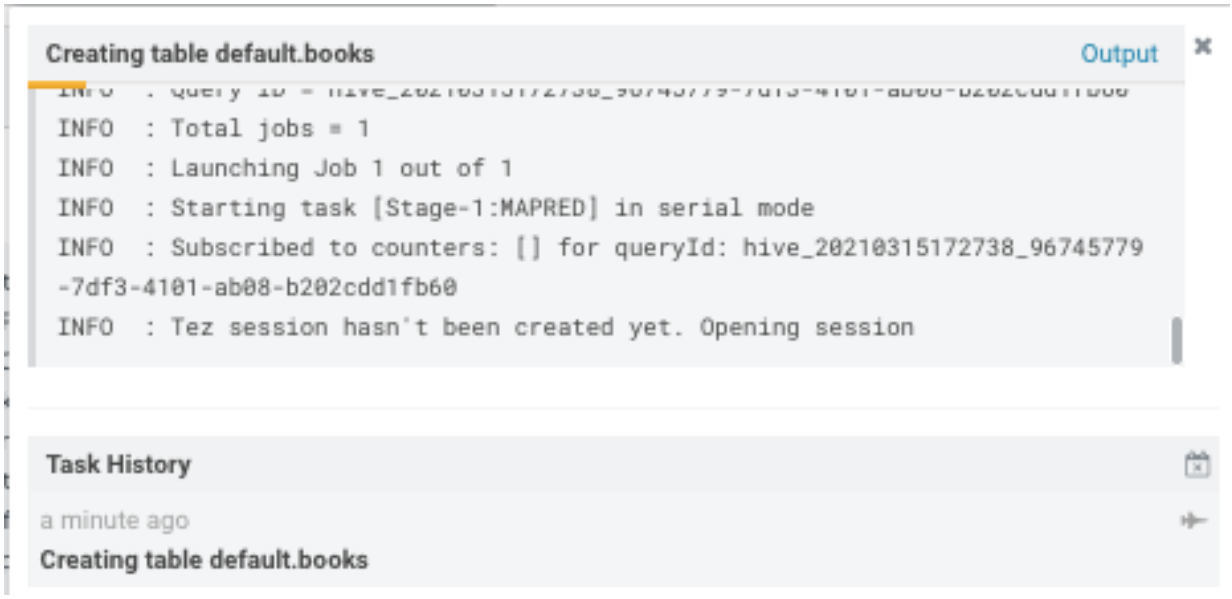

   Hue displays the logs and opens the **Table Browser** from which you can view the newly created table when the operation completes successfully.

Using Azure Data Lake Storage Gen2 with Hue

Hue can read to and write to an Azure Data Lake Storage (ADLS) Gen2.

In Cloudera Data Warehouse, you can browse ADLS Gen2 storage from Hue in the following ways:

- With Ranger Authorization Service (RAZ)
- Without Ranger Authorization Service (RAZ)

Related Information

Registering a RAZ-enabled Azure environment

Accessing ADLS Gen2 containers from Hue in Cloudera Data Warehouse with RAZ

Hue offers you the capability to browse ADLS Gen2 containers, upload files to ADLS Gen2 containers, and create tables by importing files from ABFS. With Ranger Authorization (RAZ), you can grant fine-grained access to per-user home directories.

About this task

If you have enabled RAZ while registering your AWS environment with Cloudera, then Hue uses RAZ as the default mechanism for enabling the ABFS File Browser. Before you can enable the ABFS File Browser in Hue, you must complete the following prerequisites:

Procedure

1. Follow the instructions listed in Introduction to RAZ on Azure environments to register an Azure environment with the Enable Ranger authorization for ADLS Gen2 option enabled. You can use the Cloudera web interface or the Cloudera CLI to complete this task.

   Note: You must enable RAZ while registering your environment with Cloudera.

2. Log in to the Cloudera Management Console as a DWAdmin or DWUser and go to the Cloudera Data Warehouse service.
3. Click ⋮ Open Ranger on your Database Catalog.
4. Create the following Ranger policies:
   a) Hadoop SQL policy (all - database, table, column, all - url).

      You must grant permissions to individual users or groups in these Ranger policies. To grant permissions to all users, you can specify {USER} in the Permission section.
   b) ABFS (cm_ADLS) policy (Default: User Home)

      You must grant permissions to the following users in the Permissions section for the user home directory: {USER}.
   c) ABFS (cm_ADLS) policy (Default: user)

      You must grant permissions to the following users in the Permissions section for the root directory (/user/): hive, impala.
5. You must also grant appropriate permissions to the users in Cloudera User Management Service (UMS). For example, EnvironmentUser.
6. Specify the storage account name in the Storage Account field and the directory path of the container and its sub-directories in the Storage Account Container field of the cm_ADLS Ranger policy.

Enabling the ABFS File Browser for Hue in Cloudera Data Warehouse with RAZ

The ABFS File Browser in Hue is enabled by default. However, you must set the path to your ADLS Gen2 container in the hue-safety-valve field to avoid a 403 error when you click on the ABFS File Browser.

Procedure

1. Sign in to Cloudera Data Warehouse DWAdmin or DWUser.
2. Go to the Virtual Warehouse from which you want to access the ADLS Gen2 containers and click ✎.
3. Go to CONFIGURATIONS Hue and select hue-safety-valve from the Configuration files drop-down menu.
4. Add the path to your ADLS Gen2 container under the [filebrowser] section as follows:

```
[filebrowser]
remote_storage_home==abfs://[***CONTAINER-FOR-DATA-ACCESS***]/user
```

   (Optional) Per-user home directories are created by default. To disable automatic user directory creation, you can add the following lines in the hue-safety-valve as follows:

```
[desktop]
[[raz]]
autocreate_user_dir=false
```

5. Click APPLY.
   You should be able to view the icon for the ABFS File Browser on the left assist panel on the Hue web interface.

Accessing ADLS Gen2 containers from Hue in Cloudera Data Warehouse without RAZ

To enable access to Azure Data Lake Storage (ADLS) Gen2 containers from Hue, you must have onboarded to Cloudera Public Cloud and must meet the requirements listed in this section.

Only Hue superusers can view and access the ABFS File Browser.

Creating an Azure storage account

You need an Azure storage account to use ABFS with Hue.

Procedure

1. Sign in to the Microsoft Azure portal as an administrator.
2. On the Create storage account Advanced page, enable Data Lake Storage Gen2 so that the objects and files within your account can be organized into a hierarchy of directories and nested subdirectories in the same way that the file system on your computer is organized.

Setting storage location base

You must specify the Storage Location Base to configure a default ADLS Gen2 base storage location for the Cloudera environment when you register your Azure environment with Cloudera.

About this task

While registering an Azure environment Cloudera Management Console, set the Storage Location Base in the Data Access section as follows:

```
abfs://storage-fs@[***AZURE-STORAGE-ACCOUNT-NAME***].dfs.core.windows.net
```

This location is used to read and store data.

Enabling the ABFS File Browser

To enable access to ADLS Gen2 containers from the Hue web interface, you must add the Azure environment details in the hue-safety-valve configuration from your Virtual Warehouse. After enabling the ABFS File Browser, you can browse the ADLS Gen2 containers, create folders, and upload files from your computer, and import files to create tables.

Procedure

1. Sign in to Cloudera Data Warehouse.
2. Go to the Virtual Warehouse from which you want to access the ADLS Gen2 containers and click .
3. On the **Virtual Warehouses** detail page, click the Hue tab and select hue-safety-valve from the drop-down menu.
4. Add the following configuration for Hive or Impala Virtual Warehouse in the space provided:

   For the Hive Virtual Warehouse:

```
[desktop]
# Remove the file browser from the blocked list of apps.
# Tweak the app_blacklist property to suit your app configuration.
app_blacklist=oozie,search,hbase,security,pig,sqoop,spark,impala
[azure]
  [[azure_accounts]]
    [[[default]]]
      client_id=[***AZURE-ACCOUNT-CLIENT-ID***]
      client_secret=[***AZURE-ACCOUNT-CLIENT-SECRET***]
      tenant_id=[***AZURE-ACCOUNT-TENANT-ID***]

    [[abfs_clusters]]
      [[[default]]]
  fs_defaultfs=abfs://[***CONTAINER-NAME***]@[***AZURE-STORAGE-ACCOUNT-N
AME***]>.dfs.core.windows.net
              webhdfs_url=https://[***AZURE-STORAGE-ACCOUNT-NAME***].dfs.c
ore.windows.net/
```

   For Impala Virtual Warehouse:

```
[desktop]
# Remove the file browser from the blocked list of apps.
# Tweak the app_blacklist property to suit your app configuration.
app_blacklist=spark,zookeeper,hive,hbase,search,oozie,jobsub,pig,sqoop,sec
urity
```

```
[azure]
  [[azure_accounts]]
    [[[default]]]
      client_id=[***AZURE-ACCOUNT-CLIENT-ID***]
      client_secret=[***AZURE-ACCOUNT-CLIENT-SECRET***]
      tenant_id=[***AZURE-ACCOUNT-TENANT-ID***]

    [[abfs_clusters]]
      [[[default]]]
  fs_defaultfs=abfs://[***CONTAINER-NAME***]@[***AZURE-STORAGE-ACCOUNT-NA
ME***]>.dfs.core.windows.net
              webhdfs_url=https://[***AZURE-STORAGE-ACCOUNT-NAME***].df
s.core.windows.net/
```

Make sure that the container name and the Azure storage account name that you specify under the abfs_clusters section is same as what you specified under Data Access Storage Location Base while activating the Azure environment, so that Hive or Impala has permission to access the uploaded files.

5. Click APPLY.
   The ABFS File Browser icon appears on the left Assist panel on the Hue web interface after the Virtual Warehouse restarts.

Creating tables by importing CSV files from ABFS

You can create tables in Hue by importing CSV files stored in ABFS. Hue automatically detects the schema and the column types, thus helping you to create tables without using the CREATE TABLE syntax.

About this task

The maximum file size supported is three gigabytes.

(Non-RAZ deployment) Only Hue Superusers can access ADLS Gen2 containers and import files to create tables. To create tables by importing files from ADLS, you must assign and authorize use of a specific bucket on ADLS Gen2 containers for your environment. The bucket then appears like a home directory on the Hue web interface.

Procedure

1. In the service **Overview** page, select the Virtual Warehouse in which you want to create the table, click the options menu in the upper right corner and click Open Hue.
2. From the left assist panel, click on Importer.
3. On the **Importer** screen, click .. at the end of the Path field:

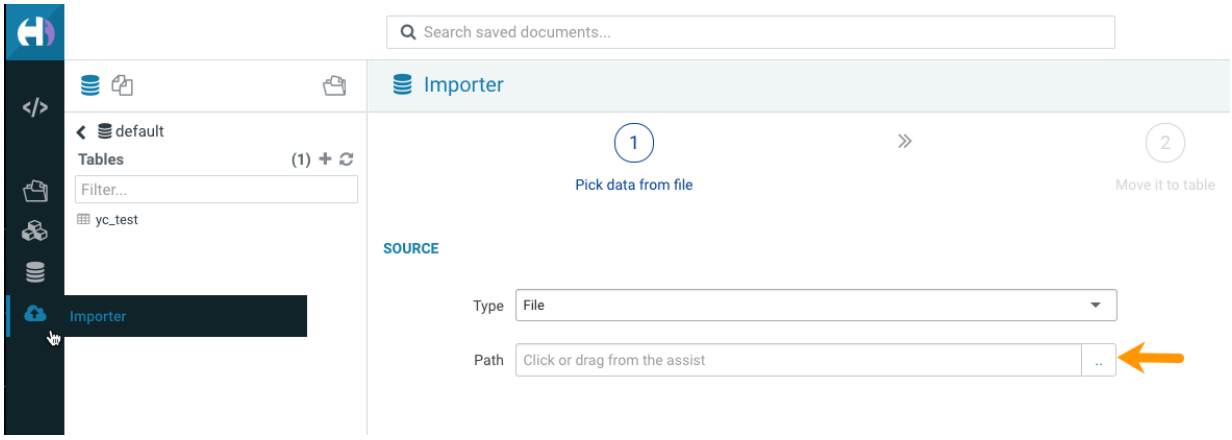

Choose a file pop-up is displayed.

4. (Non-RAZ deployment) Type abfs://[***CONTAINER-NAME***] in the address text box and press enter.

   The ABFS containers created under the Azure storage account are displayed.

   You can narrow down the list of results using the search option.

   If the file is present on your computer, then you can upload it to ABFS by clicking Upload a file.
5. Select the CSV file that you want to import into Hue.

   Hue displays the preview of the table along with the format.

   Hue automatically detects the field separator, record separator, and the quote character from the CSV file. If you want to override a specific setting, then you can change it by selecting a different value from the drop-down menu.
6. Click Next.

   On this page, you can set the table destination, partitions, and change the column data types.
7. Verify the settings and click Submit to create the table.

   The CREATE TABLE query is triggered.

   Hue displays the logs and opens the **Table Browser** from which you can view the newly created table when the operation completes successfully.

Granting permission to access S3 and ABFS File Browser in Hue

Only admin users can view and access S3 or ABFS File Browser in Hue after enabling it. You must manually grant application permissions to non-admin users and groups for them to be able to view and access S3 and ABFS File Browsers in Hue.

About this task

The following table lists the application permissions for each cloud storage type:

Cloud storage	Hue application permission
S3	filebrowser.s3_access:Access to S3 from filebrowser and filepicker.
ABFS (ADLS Gen2)	filebrowser.abfs_access:Access to ABFS from filebrowser and filepicker.
ADLS (Gen1)	filebrowser.adls_access:Access to ADLS from filebrowser and filepicker.
GS	Access to GS from filebrowser and filepicker.

Before you begin

You can only assign Hue application permissions to groups or users within a group. Add the users to a group to whom you want to grant access to the S3 or ABFS File Browsers.

Important: The "default" group in Hue does not have the permissions required to view S3 or ABFS File Browsers, by default. If your users belong to the default group, then you must manually grant the required permissions.

Procedure

1. Open Hue from Cloudera Data Warehouse Virtual Warehouse as an EnvironmentAdmin.
2. Go to admin Manage Users Groups .
3. Click on the group to whom you want to grant the filebrowser application permissions.
4. On the **Edit [***GROUP-NAME***]** page, select the required permission under the permission section and click Update group.

Creating tables in Hue by importing files

Using Hue Importer, you can create Hive, Impala, and Iceberg tables from CVS and XLSX files. After enabling the File Browser for your cloud provider, you can import the file into Hue to create tables.

About this task

You can upload and import a 200 KB file from your local computer, or import a file up to 3 GB from cloud storage (S3, ADLS Gen2, and Google Cloud Storage (GS) buckets) using the Importer.

Note: To enable uploading a small local file from your computer, you must go to Cloudera Manager Clusters Hue Configuration and add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[indexer]
  # Flag to turn on the direct upload of a small file.
  enable_direct_upload=true
```

You can create managed and external tables. When you create managed tables, the data is moved to HDFS in the Cloudera Data Warehouse workspace. When you create external tables, your data remains in its original location.

Note: Superusers can browse all directories up to the root. Other users can only browse and access buckets and directories that they are granted access to.

Before you begin

- Enable user access to cloud storage buckets/containers in Ranger
- Enable the File Browser for your cloud provider

Procedure

1. Log in to the Hue web interface.
2. Click Importer from the left-assist panel.

   Alternatively, click + to create tables under the Databases Tables list view. This opens the Hue Importer.
3. Under SOURCE, select Remote File from the Type drop-down menu.
4. Click .. at the end of the Path field.

   The **Choose a file** modal is displayed.
5. (Non-RAZ deployment only) Click on your cloud provider, type the following depending on your cloud provider, and press enter:

   (AWS): s3a://[***BUCKET-NAME***]

   (Azure): abfs://[***CONTAINER-NAME***]

   (GCS): gs://[***BUCKET-NAME***]
6. Browse and select the file you want to use to create a table.

   Hue displays the preview of the table along with the format.

   Hue automatically detects the field separator, record separator, and the quote character from the file. If you want to override a specific setting, then you can change it by selecting a different value from the drop-down menu.
7. Click Next.

   The table destination and properties are displayed.
8. Under Extras, deselect Store in Default location and Transaction table options.

   This is required to create external tables.

9. Select the Copy file option.

   Selecting this option retains the source file in the original location and creates a copy of the file to create a table.

   If you do not select this option, then Hue moves the file from the source location to a new folder, making it unavailable in the original location.
10. Set the table destination, partitions, and change the column data types.
11. Verify the settings and click Submit to create the table.

    The CREATE TABLE query is triggered.

    Hue displays the logs and opens the Table Browser from which you can view the newly created table when the operation completes successfully.

Uploading files using Hue with the task server enabled in Cloudera Data Warehouse

The task server is enabled by default. When task server is in the enabled state, the files are uploaded using an asynchronous task queue or job queue. This improves performance and allows you to upload multiple files as large as 5 GB each in parallel.

Before you begin

You must have access to the cloud storage (AWS S3 or Azure ADLS Gen2).

Procedure

1. Log in to the Hue web interface as a normal user.
2. Go to the Hue File Browser.
3. Click Schedule Upload.
4. Click Select files to browse the files from your local system, and then click Upload.

   Alternatively, you can drag and drop the file into the file upload dialogue box.

   Note: 5 GB is the maximum supported upload file size per file.

List of supported non-alphanumeric characters for file and directory names in Hue

Auto-generated files may often introduce non-alphanumeric characters in the filenames which are not supported by Hue. This might cause the files or directories to not appear on the Hue File Browser. Review the list of non-alphanumeric characters supported in Hue to avoid running into this issue.

Table 1: Non-alphanumeric characters supported in Hue

Special character symbol	Description
~	Tilde
@	Ampersat
#	Hash
$	Dollar sign
&	Ampersand
(	Left paranthesis
)	Right paranthesis

Special character symbol	Description
*	Asterisk
!	Exclamation mark
+	Plus
=	Equal
:	Colon Not supported with Knox.
;	Semicolon
,	Comma
.	Period
?	Question mark Not supported with Knox.
/	Forward slash Not supported with Knox.
\	Backslash
'	Apostrophe or single quote

Unsupported features in Hue

Learn about the Hue features that are not supported by Cloudera.

Unsupported options in Hue Importer

The following options are displayed on the Hue **Importer** page under SOURCE Path , but are not supported:

- External Database

  Creating an external database using the Hue Importer is not supported. Cloudera recommends that you create a database using a SQL query.
- Manually

Known limitations in Hue

Review the known limitations in Hue.

Hue has the following limitations:

- Node depth for graphing Oozie workflows because of performance issues. See Improved Oozie Workflow display of large Graphs.
- You must use the Cloudera-provided Apache Load balancer to serve static content, because:
  - It serves static JavaScript, CSS, and Webpack files for client requests and reduces the load from the backend Python web server.
  - The Hue load balancer uses a sticky cookie session to route requests to the same backend as the Python web server, which talks to the same coordinator.
- Hue can only show logs from either Spark1 or Spark2, not both at a time.
- Spark notebook is not supported.
- External RDBMS in the query editor is not supported out of the box by default. Cloudera support will assist on a best-effort basis. Cloudera recommends that you raise issues in the open-source github community.
- Impala queries stay in the “executing” state so that Hue can display results when users are ready

- We need to limit the amount of data available to download from Hive/Impala because massive downloads cause performance degradation. Multiple simultaneous downloads of result sets could also degrade performance.
- Upstream features and connectors may not function properly in Cloudera. Cloudera recommends that you raise issues in the open-source github community.

CDP Public Cloud / Data Catalog

Cloudera Data Catalog Overview

Date published: 2019-11-14
Date modified: 2024-11-12

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

About Cloudera Data Catalog....4
Profiler architecture in VM-based environments....5
Profiler architecture in Compute Cluster enabled environment....6

Cloudera Data Catalog terminology....7

Before you start....7
Prerequisites to access Cloudera Data Catalog....8
Providing role access....9
Authorization for viewing assets....11
Restricting access for certain users of Cloudera Data Catalog....12

Cloudera Data Catalog Dashboard....14

Understanding datasets....15

Understanding the Cloudera Data Catalog Profilers....16
Understanding the Cluster Sensitivity Profiler....16
Understanding the Hive Column Profiler....18
Understanding the Ranger Audit Profiler....19

About Cloudera Data Catalog

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across the enterprise.
Cloudera Data Catalog helps you understand data lying in your data lake. You can locate relevant data of interest based on various parameters. Using Cloudera Data Catalog, you can understand how data is interpreted for use, how it is created and modified, and how data access is secured and protected.

Note: Cloudera Data Catalog is supported in the EU and APAC regional Control Plane. You can ensure that you manage, secure, collaborate, and govern data assets across multiple clusters and environments within the EU region or APAC region where your organization operates as per the data protection regulatory requirements.

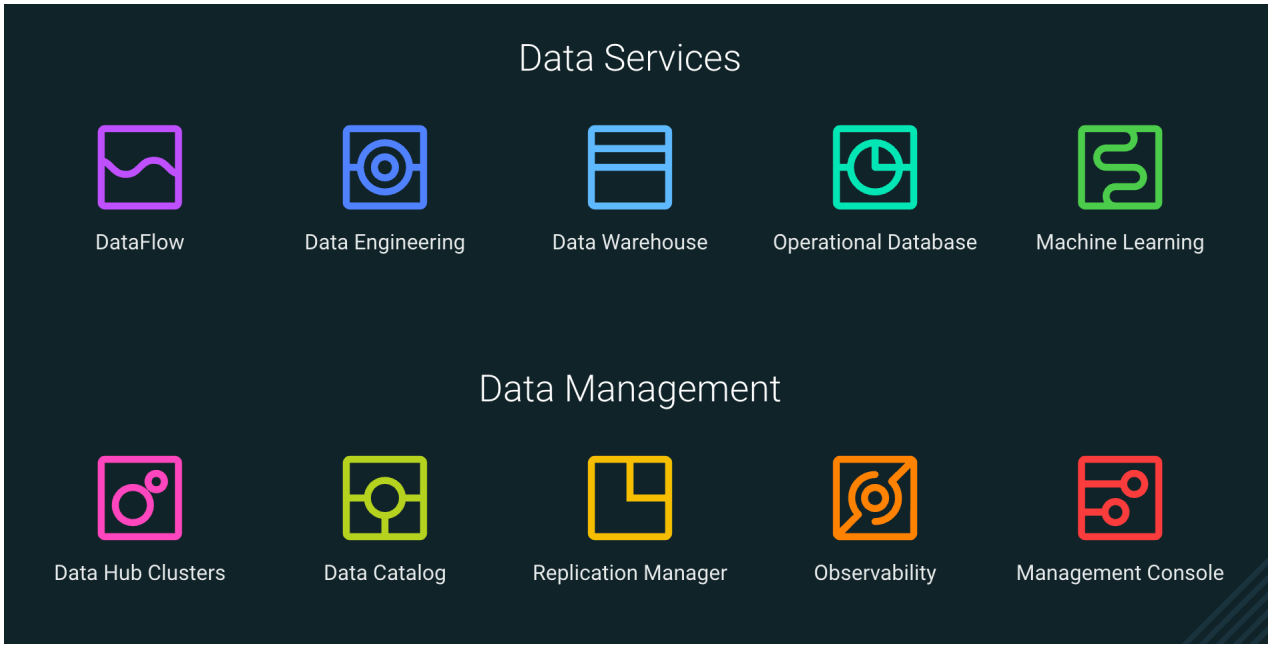

Cloudera Data Catalog enables data stewards across the enterprise to work with data assets in the following ways:

- Organize and curate data globally
  - Organize data based on business classifications, purpose, protections needed, etc.
  - Promote responsible collaboration across enterprise data workers
- Understand where relevant data is located
  - Catalog and search to locate relevant data of interest (sensitive data, commonly used, high risk data, etc.)
  - Understand what types of sensitive personal data exists and where it is located
- Understand how data is interpreted for use
  - View basic descriptions: schema, classifications (business cataloging), and encodings
  - View statistical models and parameters
  - View user annotations, wrangling scripts, view definitions etc.
- Understand how data is created and modified
  - Visualize upstream lineage and downstream impact
  - Understand how schema or data evolve
  - View and understand data supply chain (pipelines, versioning, and evolution)

- Understand how data access is secured, protected, and audited
  - - Understand who can see which data and metadata (for example, based on business classifications) and under what conditions (security policies, data protection, anonymization)
    - View who has accessed what data from a forensic audit or compliance perspective
    - Visualize access patterns and identify anomalies

Related Information

Cloudera Data Catalog Terminology

Profiler architecture in VM-based environments

In a VM-based environment, the Cloudera Data Catalog Profiler architecture uses a Cloudera Data Hub workload cluster.

Figure 1: VM-based profiler architecture

After registering a VM-based environment, you have to launch a Cloudera Data Hub cluster for each data lake to provide the resources and services required for a profiler workload. This can be handled by Cloudera Data Catalog. For more information, see Launch profiler Cluster.

Note: In comparison to a Kubernetes pod in a Compute Cluster enabled environment, a Cloudera Data Hub workload cluster reserves compute resources even when a profiler task is not running. Also, more services are required to be included in the Cloudera Data Hub cluster template in contrast to the default compute cluster:

Zookeeper

For configuration information, naming, synchronization and group services over large clusters in distributed systems

Yarn

For resource management

1. Cloudera Data Hub uses the internal service called Cloudbreak to start the necessary services in the Profiler Cloudera Data Hub cluster. It is also used to access data about profilers and the data lake. In comparison, the Cluster Proxy provides the connection between the Cloudera Data Hub UI service and the rest of the Cloudera Data Catalog services.
2. An additional Amazon Relational Database (PostgreSQL) is used to store data required for the profiling process, such as, Custom Sensitivity Profiler Rules, profiler-data lake mappings and datasets.
3. Knox is used to authenticate services between your and Cloudera’s environment

4. Livy is used together with a dedicated Scheduler Service to start the individual profiler instances with Spark jobs.
5. The Cloudera Data Hub cluster manages the different services responsible for the profiling.
   a. Profiler Admin service is similar to an interface for Profilers. It allows Cloudera Data Hub to fetch information from the workload Cloudera Data Hub about scheduled jobs, profiler configurations and so on.

      Profiler Metrics is responsible for the metrics calculation and synching it to Cloudera Data Hub database and Atlas.
   b. The profilers use a cloud storage called Profiler output bucket as a temporary storage to aggregate all their collected data, such as profiler snapshots, which help to continue the profiling by saving interim data.
6. The final profiler results are stored in an attached cloud storage.

Related Information

Cloudera Data Hub

Introduction to Data Lakes

Cloudera Management Console

Profiler architecture in Compute Cluster enabled environment

Next to the Cloudera Data Hub based profiler cluster, Cloudera Data Catalog offers the possibility to run profilers as a containerized service in a standardized Kubernetes base cluster called Externalized Compute Cluster. This consumes far less resources and provides auto-scaling.

Figure 2: Profiler architecture in Compute Cluster enabled environment

1. Once the container-ready environment is set up, a default Kubernetes cluster (Externalized Compute Cluster) is also created in this environment.

   Note: The Kubernetes jobs and API server offers the same API and UI interface capabilities for you as the VM-based Cloudera Data Hub, therefore, there is no difference in use.
2. The Profiler Launcher Service (PLS) internal to Cloudera Data Catalog schedules Kubernetes jobs, cron jobs in the compute cluster using HTTP API calls. Each type of a profiler has its own Kubernetes cron-jobs for handling scheduled profilers.
3. Once the time of the schedule is reached the Kubernetes job will launch a pod that will start profiling a data lake or ranger audit logs. The configuration for the jobs are received via the Cloudera Data Catalog API.

4. Using these settings the profiler connects to a data lake, identifies all the assets present in the data lake then starts profiling.
5. The results will be synced to Atlas and Cloudera Data Catalog using their respective APIs.

Related Information

Cloudera Data Hub
Cloudera Management Console
Using Compute Clusters in AWS environments
Using Compute Clusters in Azure environments

Cloudera Data Catalog terminology

An overview of terminology used in Cloudera Data Catalog.

Profiler

Enables the Cloudera Data Catalog service to gather and view information about different relevant characteristics of data such as shape, distribution, quality, and sensitivity which are important to understand and use the data effectively. For example, view the distribution between males and females in column Gender, or min/max/mean/null values in a column named avg_income. Profiled data is generated on a periodic basis from the profilers, which run at regularly scheduled intervals. Works with data sourced from Apache Ranger Audit Logs, Apache Atlas Metadata Store, and Hive.

Data Lake

A trusted and governed data repository that stores, processes, and provides access to many kinds of enterprise data to support data discovery, data preparation, analytics, insights, and predictive analytics. In the context of Cloudera, a Data Lake can be realized in practice with an Cloudera Manager enabled Cloudera Shared Data Experience cluster that runs Apache Atlas for metadata and governance services, Apache Knox, and Apache Ranger for security services.

Data Asset

A data asset is a physical asset located in the Cloudera ecosystem such as a Hive table which contains business or technical data. A data asset could include a specific instance of an Apache Hive database, table, or column. An asset can belong to multiple asset collections. Data assets are equivalent to “entities” in Apache Atlas.

Datasets

Datasets allow users of Cloudera Data Catalog to manage and govern various kinds of data objects as a single unit through a unified interface. Asset collections help organize and curate information about many assets based on many facets including data content and metadata, such as size/schema/tags/alterations, lineage, and impact on processes and downstream objects in addition to the display of security and governance policies.

Related Information

About Cloudera Data Catalog
Introduction to Data Lakes

Before you start

Before you access Cloudera Data Catalog on cloud, you must deploy the service.

Before deploying Cloudera Data Catalog, make sure you have reviewed and compiled with the requirements in the installation guide for your environment.

Cloudera Data Catalog is supported on Compute Cluster enabled and VM-based environments. For more information, see the related documents.

For VM-based environment

- Creating and managing Cloudera deployments
- Register your first environment

For Compute Cluster enabled environment

- AWS environments:
  - Using Compute clusters
  - Enabling default Compute Cluster for an existing environment
- Azure environments:
  - Using Compute clusters
  - Enabling default Compute Cluster for an existing environment

Prerequisites to access Cloudera Data Catalog

To access the Cloudera Data Catalog, you must have the required credentials.

Follow these instructions to provide the required access to the Cloudera Data Catalog users.

Cloudera Data Catalog users must have either an EnvironmentAdmin or EnvironmentUser role assigned.

Note: To launch and delete profilers you must be a PowerUser.

Using DataCatalogCspRuleViewer role, you can list and view the existing Custom Sensitivity Profiler rules.

The PowerUser must provide the requisite access to subscribers who plan to use Cloudera Data Catalog, either as EnvironmentAdmin or EnvironmentUser.

EnvironmentAdmin	EnvironmentUser
Can perform similar actions as EnvironmentUser.	Can create Dataset and related actions (Add assets, remove assets, tag assets, tag asset columns, and few others) for data lakes.
Additionally, in Cloudera Management Console, can perform the following: • Delete Environments. • Stop Environments. • Upgrade data lake.	Connect to Atlas and Ranger entities for data lakes for which there is access.
	Search for assets on the search page.
	Bookmark any dataset (even with no data lakes or data lakes to which access is not available).
	Access profilers.
	Create custom tags.
	Launch profilers on a data lake from the search page. Ability to launch the Workload cluster (Must have the Power User role assigned).
	Use Filters on the **Search** page of Cloudera Data Catalog to filter results.
	Launch profilers on a data lake from the search page.

Additionally, using Cloudera Manager, you must configure Apache Atlas and Apache Ranger services. Use the following instructions to complete the process.

Providing role access

You must provide the required role access to use Cloudera Data Catalog.

Procedure

1. From Cloudera Management Console Environments > Select an environment > select the Actions drop-down > Manage Access.

2. Search for the user who requires Cloudera Data Catalog access > Select the check-box, either EnvironmentAdmin or EnvironmentUser > Click Update Roles.

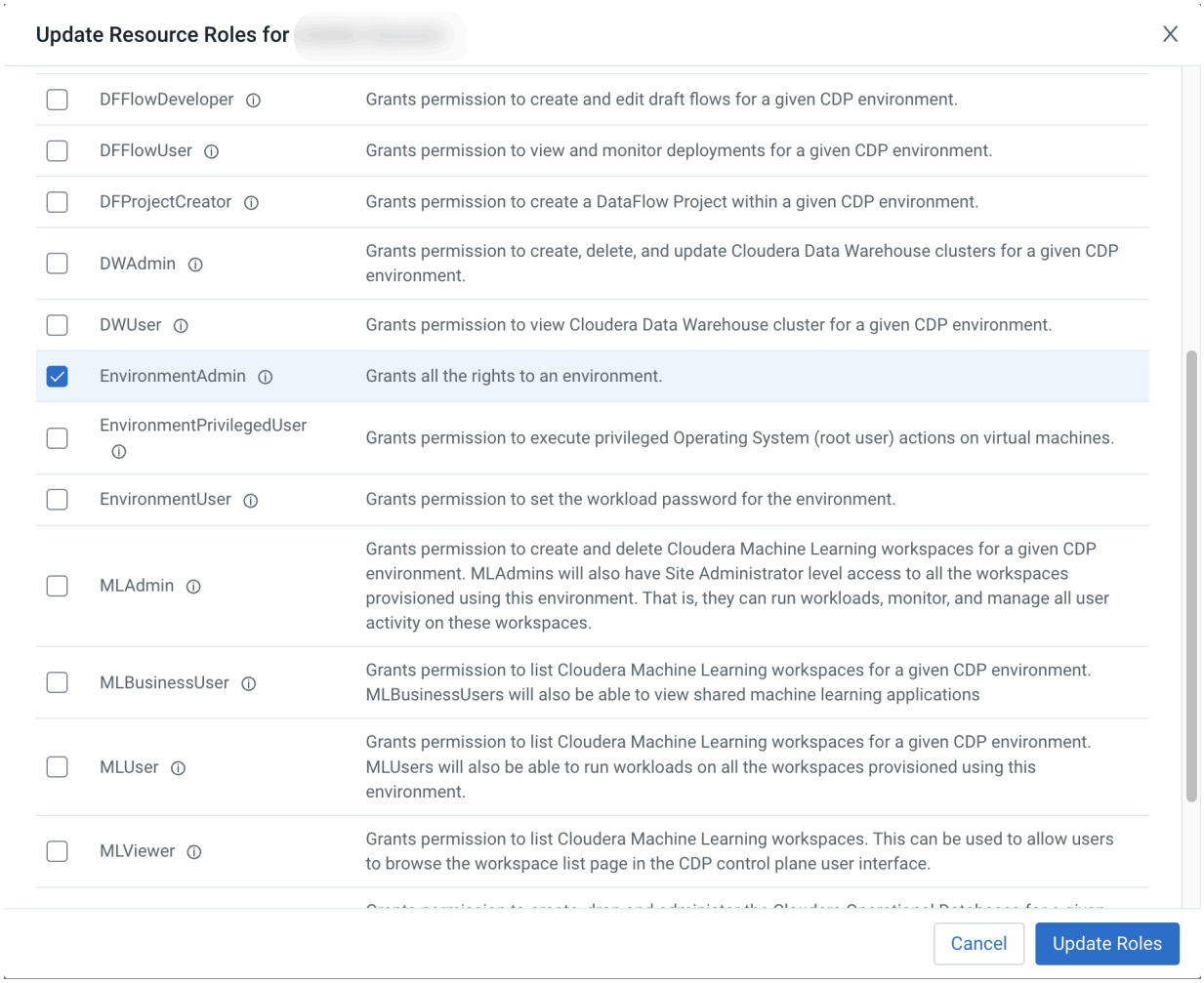

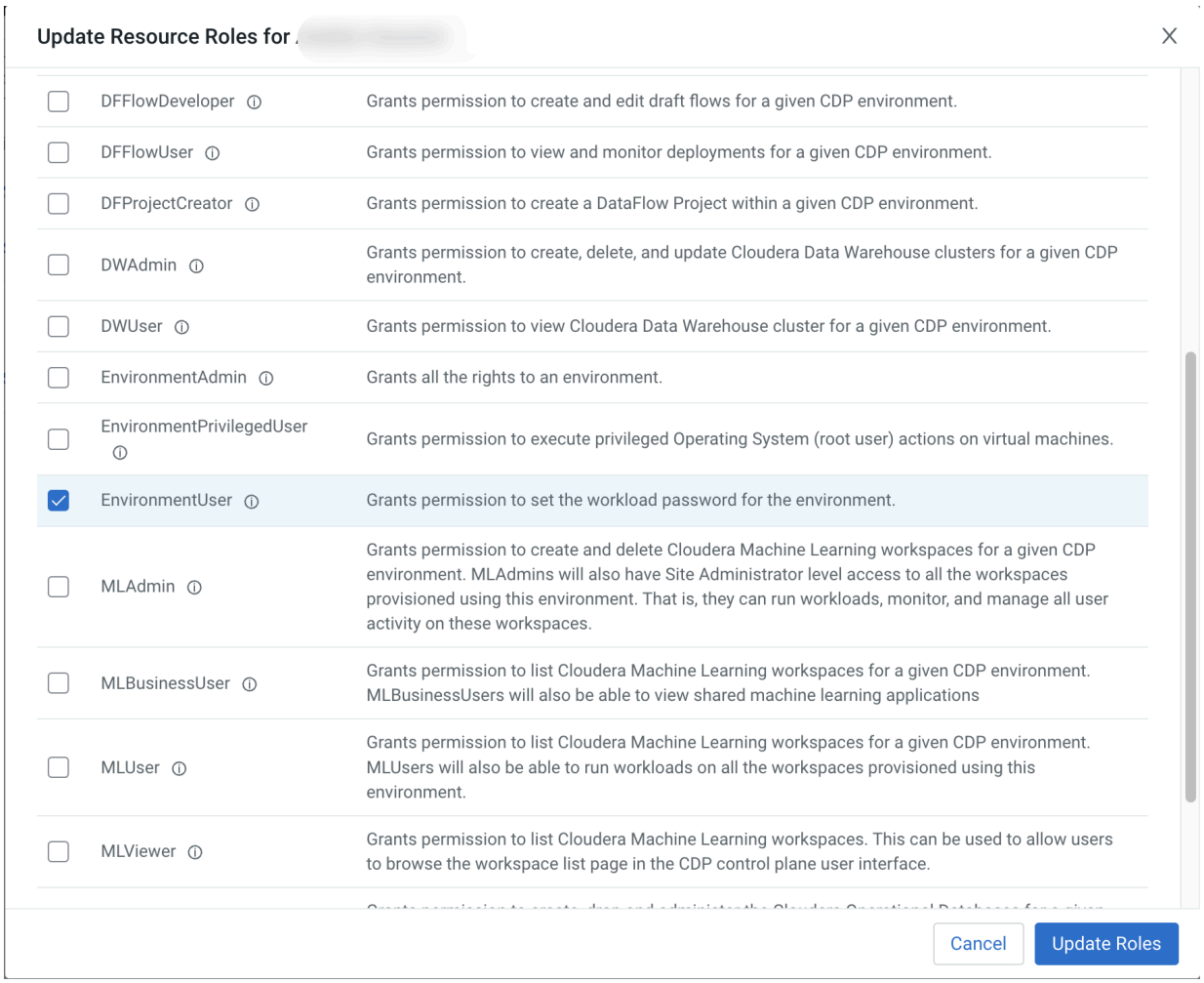

3. Navigate back to the **Clusters** page and select Actions > select Synchronize Users
   Allow the sync operation to complete and the changes to take effect.

   Attention: For tenant specific roles and related permissions, see Account roles.

Related Information

Authorization for viewing Assets

Authorization for viewing assets

Cloudera Data Catalog users must have appropriate authorization set up in Apache Ranger to view certain asset types.

Hive Ranger Policy - You must set up Hive Ranger resource-based policies as per your work requirements for Hive assets in Cloudera Data Catalog.

For example, the following diagram provides a sample Hive Ranger policy with all permissions.

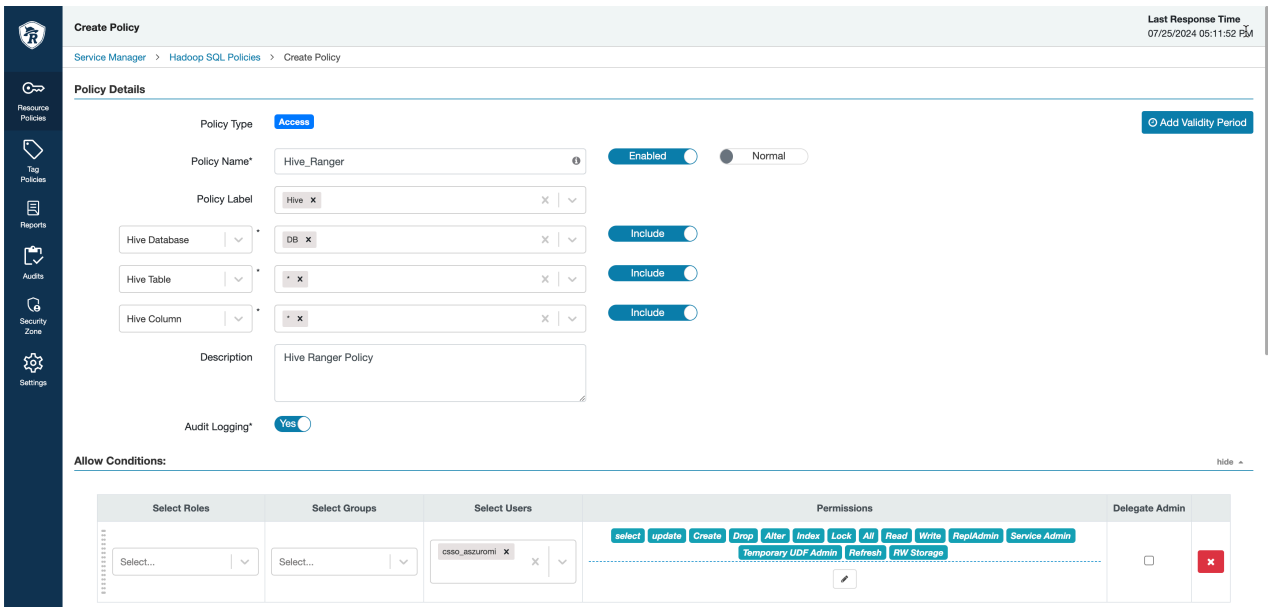

Atlas Ranger Policy- You must set-up Ranger policies for Atlas in order to work with asset search and tag flow management.

For example, the following diagram provides a sample Atlas Ranger policy.

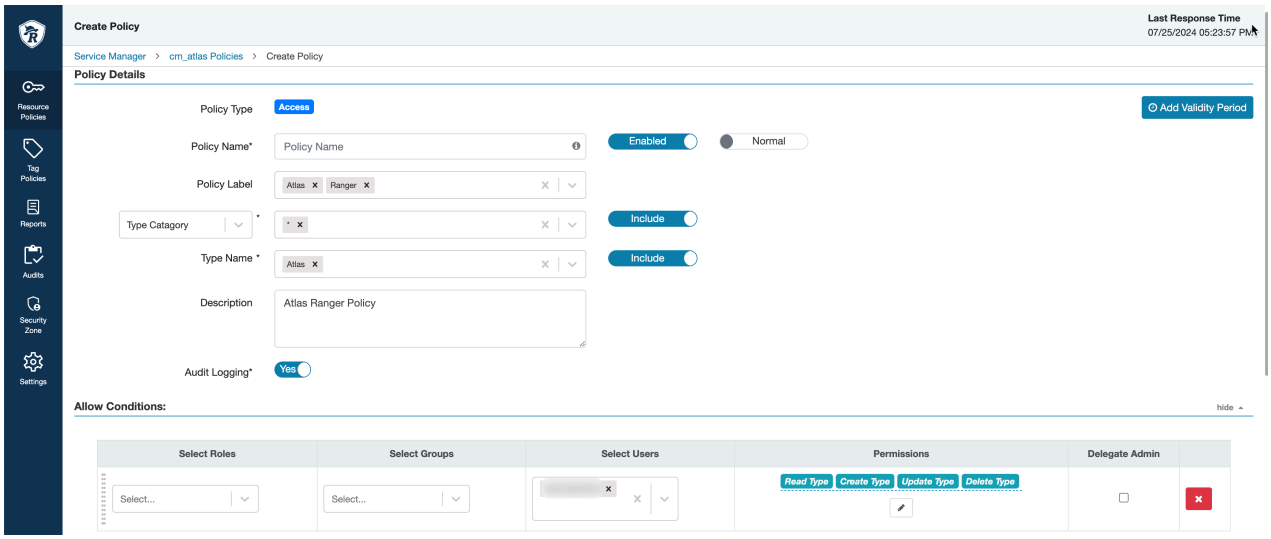

Related Information

Providing role access

Ranger Policies Overview

Configuring resource-based services

Restricting access for certain users of Cloudera Data Catalog

Restricting access for certain users of Cloudera Data Catalog

To have a fine-grained access to the same user from accessing the assets in Cloudera Data Catalog, you can perform some additional changes. For example, if you want to restrict some users from accessing specific table information, you must set-up a Ranger policy such that these users will not have access to the asset details.

Note: When a subscriber is provided with user access either as EnvironmentAdmin or EnvironmentUser, by default, the user is synchronized with the default Atlas policies in Ranger. This implies that the same user has access to all the datasets in the environment.

To create the Ranger policy to restrict users from accessing asset details, for example, with a specific classification, refer to the following images:

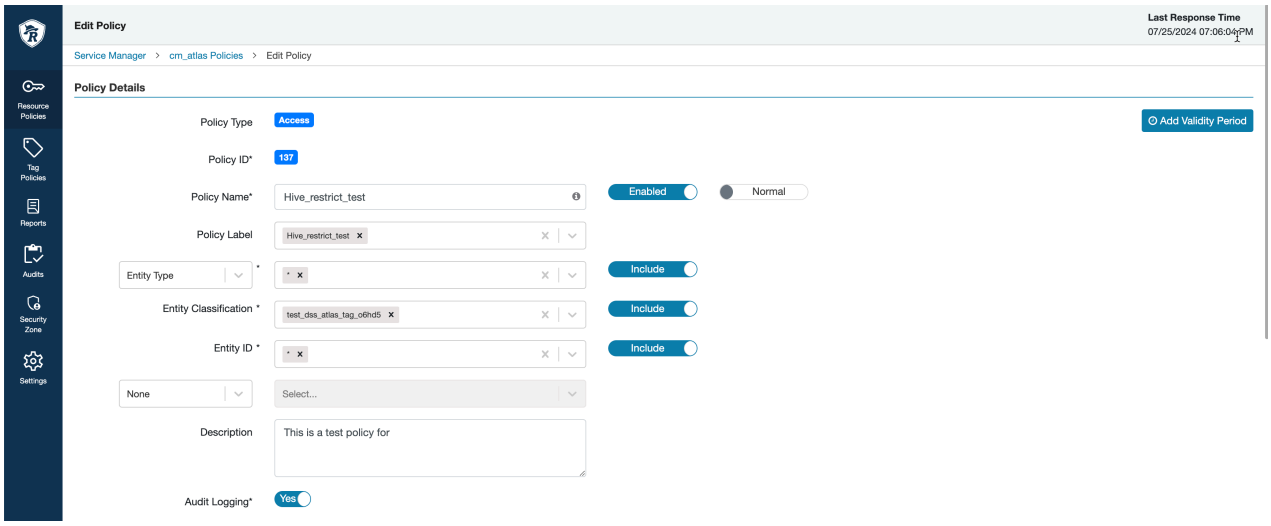

The following image displays the **Deny Conditions** set for the specific user.

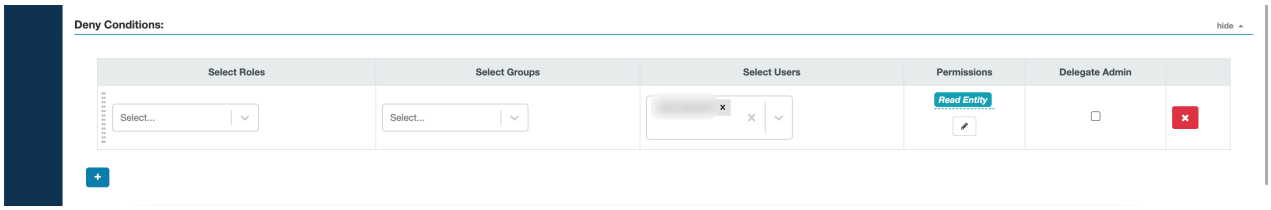

The result is depicted in the following image, where the user has no permissions to access the specified dataset.

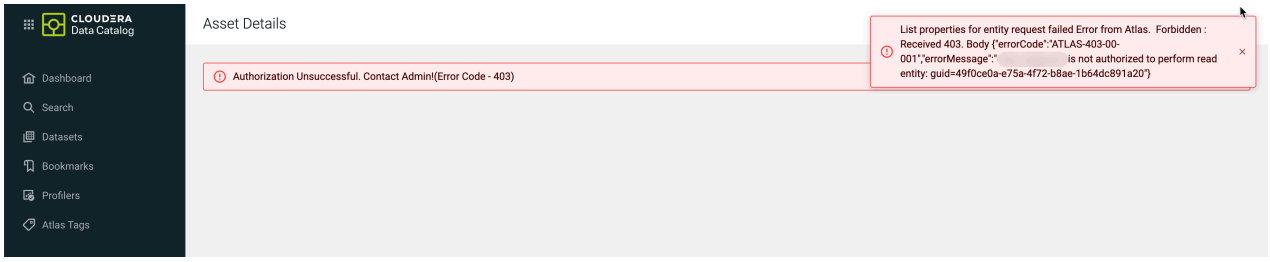

Reducing resource consumption with restricted users

Additionally, when you plan to restrict data access, please note the following:

- Audit summarization for the asset evolves from the Ranger Audit Profiler and Metrics service.
- Various Hive Column Statistical metrics for columns of the asset evolves from Atlas as part of the profile_data of a column.

To ensure that the data related to audit summary and Hive Column Statistics are not visible to the subscribers, you must make sure to turn off the audit profiler and the Hive Column Profiler respectively.

Related Information

Authorization for viewing Assets

Cloudera Data Catalog Dashboard

The Dashboard provides quick access to vital service information at a glance, in the form of visual, actionable navigation for multiple operations. The user-friendly navigation enables viewing, filtering, and acting upon data quickly and in a simple manner.

Data Stewards can view the **Dashboard** at a glance, and also focus on the most important tasks, enabling faster decision making as well as immediate action. The application lets you perform multiple actions for different types of content that helps in visualizing information with ease.

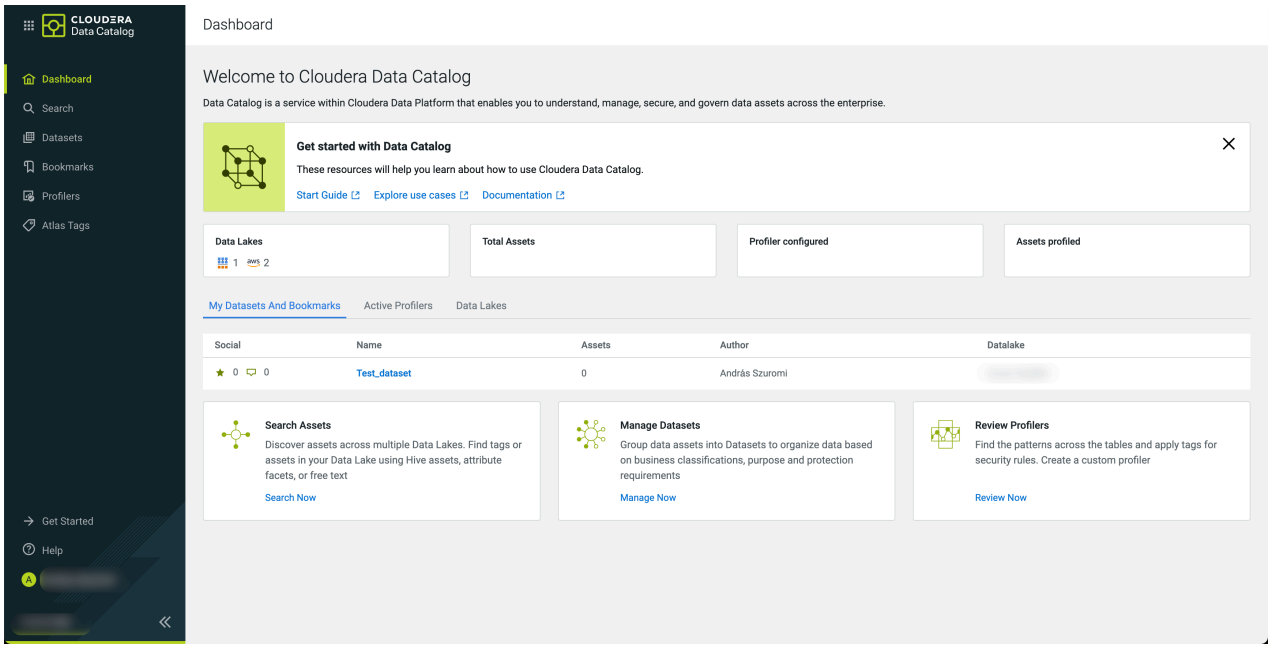

The displayed cards and tabs are fully interactive, with clickable areas for easy navigation to relevant parts of applications. Users can access individual sections and narrow down the information displayed.

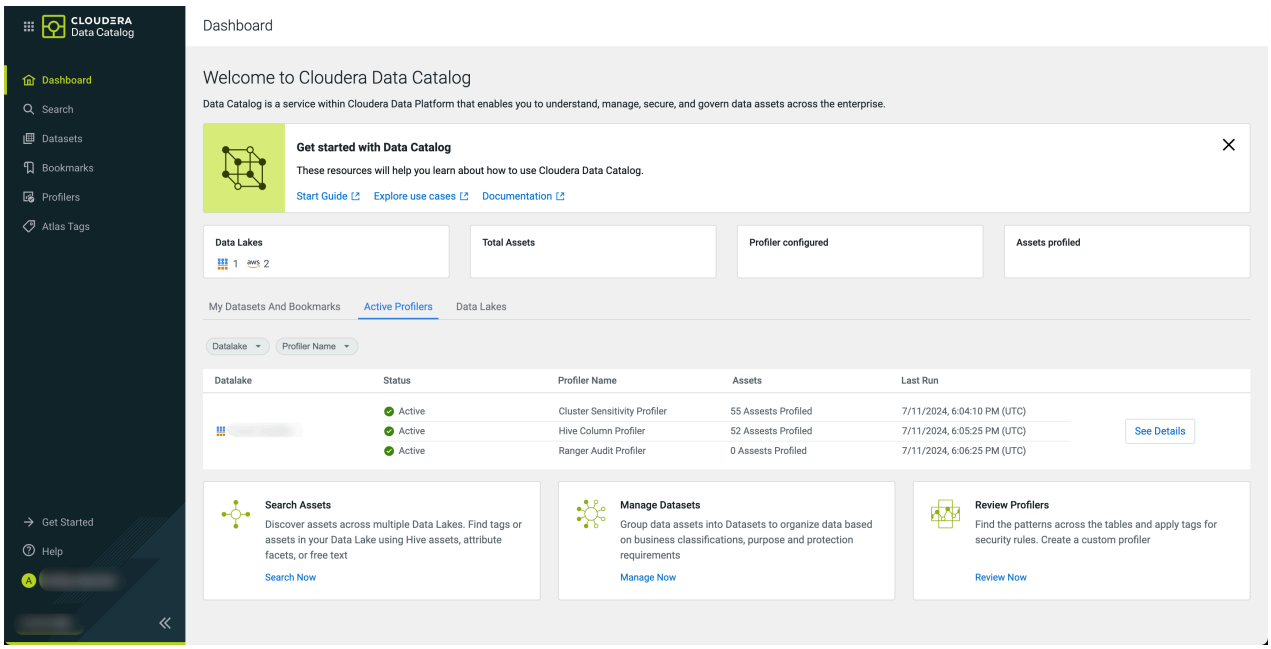

The **Dashboard** page contains information on the data lakes, the total number of assets that are profiled, along with the assets that are scanned for data.

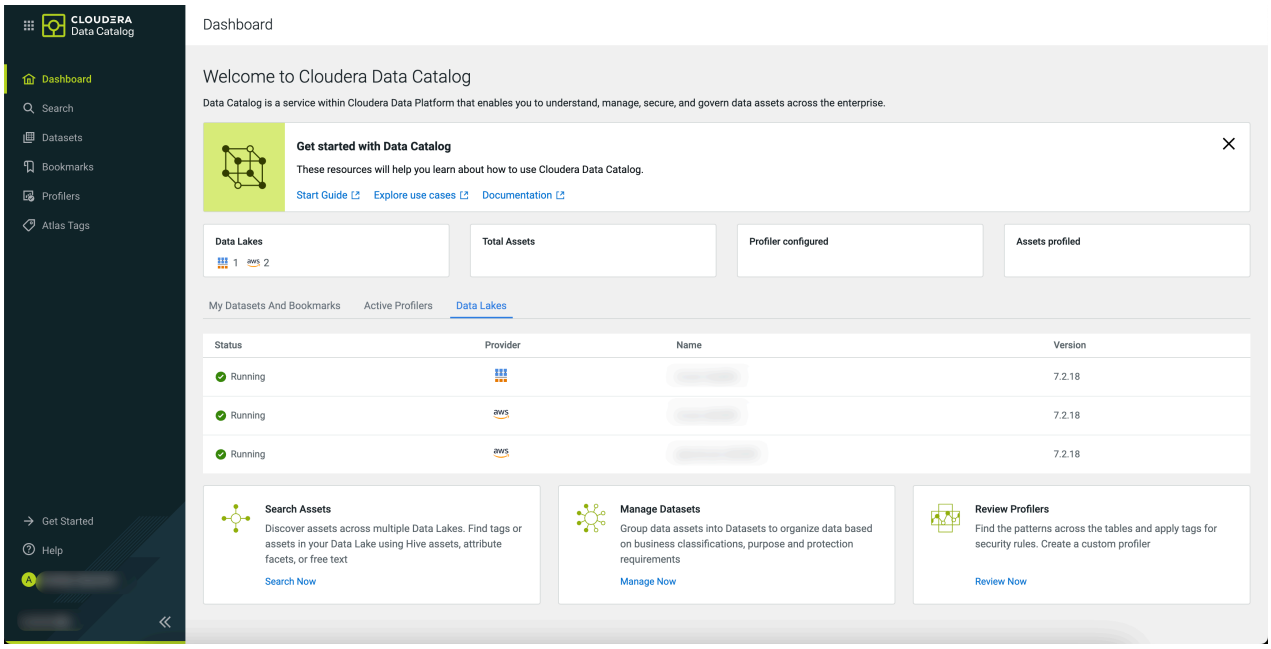

Additionally, you can manage the datasets created and bookmarked by you, check the status of profilers, and search or discover assets.

Understanding datasets

A dataset is a group of assets that fit search criteria so that you can manage and administer them collectively.

Asset collections enable you to perform the following tasks when working with your data:

- Organize

  Group data assets into datasets based on business classifications, purpose, protections, relevance, etc.
- Search

  Find tags or assets in your data lake using Hive assets, attribute facets, or free text.

  Advanced asset search uses facets of technical and business metadata about the assets, such as those captured in Apache Atlas, to help users define and build collections of interest. Advanced search conditions are a subset of attributes for the Apache Atlas type hive_table.
- Understand

  Audit data asset security and use for anomaly detection, forensic audit and compliance, and proper control mechanisms.

You can edit datasets after you create them and the assets contained within the collection will be updated. CRUD (Create, Read, Update, Delete) is supported for datasets.

Related Information

Understanding the Data Catalog Profiler

Understanding the Cloudera Data Catalog Profilers

Profilers create metadata annotations that summarize the content and shape characteristics of the data assets (such as distribution of values in a box plot or histogram).

Cloudera Data Catalog supports different environments:

- When using the VM-based environment, you can create a Cloudera Data Hub cluster for a profiler engine to run data profiling operations as a pipeline on data located in one of your data lakes. You can install the profiler agent in a data lake and set up a specific schedule to generate various types of data profiles.
- When using the Compute Cluster enabled environment, after launching a profiler cluster, an internal service provisions new Kubernetes pods, scheduling and runs profiler jobs on-demand.

Note:

- In a VM-based Cloudera Data Hub cluster environment, you can launch the profiler engine (Cloudera Data Hub cluster) for a data lake after selecting the data lake and clicking Launch Profilers. Note that, if the selected data lake already has the profiler engine attached, then the Launch Profilers option is not displayed.
- Also, the Cloudera Data Hub for the profiler can be set up only by accessing the Cloudera Data Catalog UI and is not supported through the Cloudera Management Console.

Profiler Name	Description
Cluster Sensitivity Profiler	A sensitive data profiler- PII, PCI, HIPAA and others.
Ranger Audit Profiler	A Ranger audit log summarizer.
Hive Column Profiler	Provides summary statistics like Maximum, Minimum, Mean, Unique, and Null values at the Hive column level.

Important: In VM-based environments (with Cloudera Data Hub workflows), profilers do not support Iceberg tables. However, they are discoverable by Cloudera Data Catalog.

Related Information

Understanding Datasets

Understanding the Cluster Sensitivity Profiler

The Cluster Sensitivity Profiler automatically performs context and content inspection to detect various types of sensitive data and suggest suitable classifications or tags based on the type of sensitive content detected or discovered.

Auto-detected data types

Type of data

- Bank account
- Credit card
- Driver number (UK)
- Email

- IBAN number
  - Austria (AUT)
  - Belgium (BEL)
  - Bulgaria (BGR)
  - Switzerland (CHE)
  - Cyprus (CYP)
  - Czech Republic (CZE)
  - Germany (DEU)
  - Denmark (DNK)
  - Spain (ESP)
  - Estonia (EST)
  - Finland (FIN)
  - France (FRA)
  - United Kingdom (GBR)
  - Greece (GRC)
  - Croatia (HRV)
  - Hungary (HUN)
  - Ireland (IRL)
  - Iceland (ISL)
  - Italy (ITA)
  - Liechtenstein (LIE)
  - Lithuania (LTU)
  - Latvia (LVA)
  - Luxembourg (LUX)
  - Malta (MLT)
  - Netherlands (NLD)
  - Norway (NOR)
  - Poland (POL)
  - Portugal (PRT)
  - Romania (ROU)
  - Slovakia (SVK)
  - Slovenia (SVN)
  - Sweden (SWE)
- IP address
- NPI
- Name

- National ID number
  - Bulgaria (BGR)
  - Switzerland (CHE)
  - Czech Republic (CZE)
  - Denmark (DNK)
  - Spain (ESP)
  - Estonia (EST)
  - Finland (FIN)
  - Greece (GRC)
  - Ireland (IRL)
  - Iceland (ISL)
  - Italy (ITA)
  - Lithuania (LTU)
  - Latvia (LVA)
  - Norway (NOR)
  - Poland (POL)
  - Portugal (PRT)
  - Romania (ROU)
  - Slovakia (SVK)
  - Sweden (SWE)
- National insurance number (UK)
- Passport number
  - Austria (AUT)
  - Belgium (BEL)
  - Switzerland (CHE)
  - Germany (DEU)
  - Spain (ESP)
  - Finland (FIN)
  - France (FRA)
  - Greece (GRC)
  - Ireland (IRL)
  - Italy (ITA)
  - Poland (POL)
  - United Kingdom (UK)
- Bank Routing Number
- US Social Security Number
- Society for Worldwide Interbank Financial Telecommunication (SWIFT)
- Telephone

Understanding the Hive Column Profiler

You can view the shape or distribution characteristics of the columnar data within a Hive table based on the Hive Column Profiler.

There are different charts available to help visualize the shape and distribution of the data within the column as well as summary statistics (such as means, null count, and cardinality of the data) for a column. The profiler computes column univariate statistics that are displayed using an appropriate chart in the **Schema** tab.

Pie charts are presented for categorical data with limited number of categories or classes. Examples include data such as eye colors that only have a fixed list of values (categories or labels).

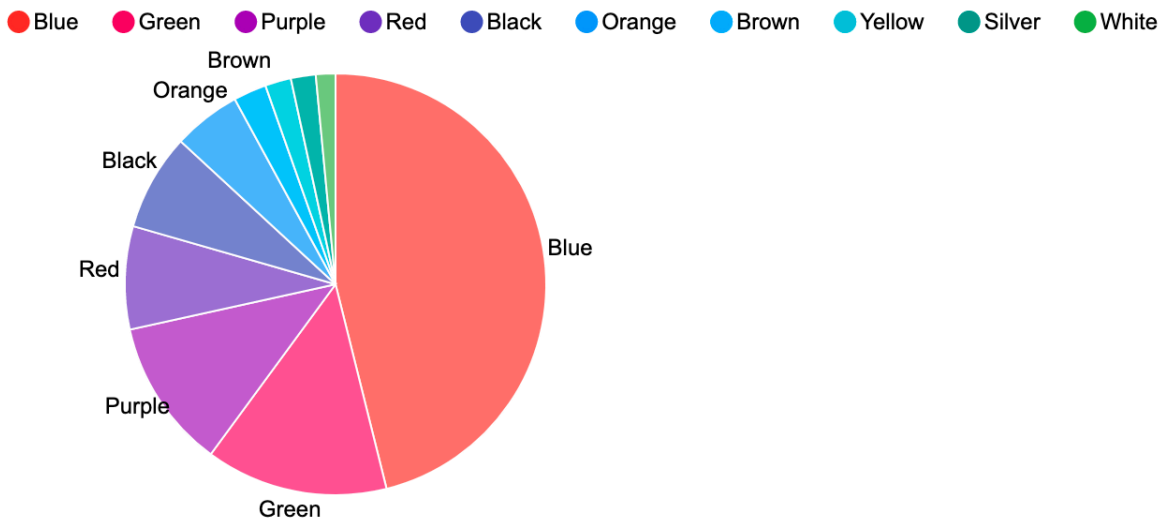

When the data within columns is numeric, a histogram of the distribution of values organized into 10 groups (decile frequency histogram) and a box plot with a five-number summary (mean, median, quartiles, maximum, and minimum values) are shown for the column.

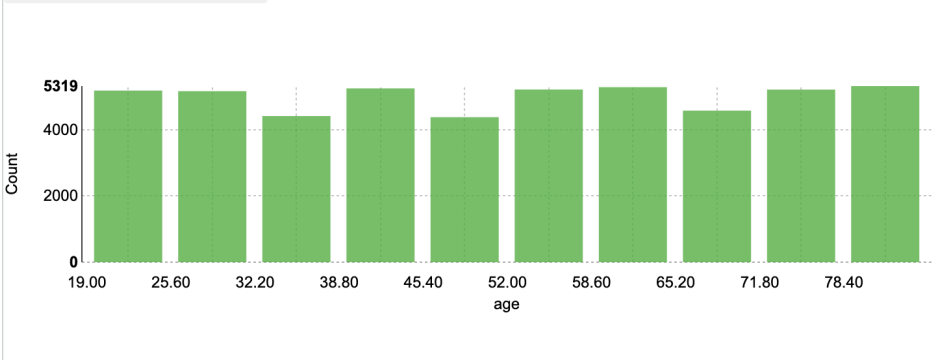

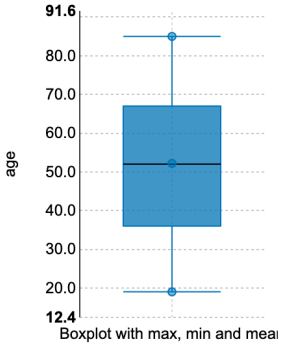

Related Information

Understanding the Ranger Audit Profiler

Understanding the Ranger Audit Profiler

You can view who has accessed which data from a forensic audit or compliance perspective, visualize access patterns, and identify anomalies in access patterns using the Ranger Audit Profiler.

The audit profiler uses the Apache Ranger audit logs to show the most recent raw audit event data as well as summarized views of audits by type of access and access outcomes (allowed/denied). Such summarized views are obtained by profiling audit records in the data lake with the audit profiler.

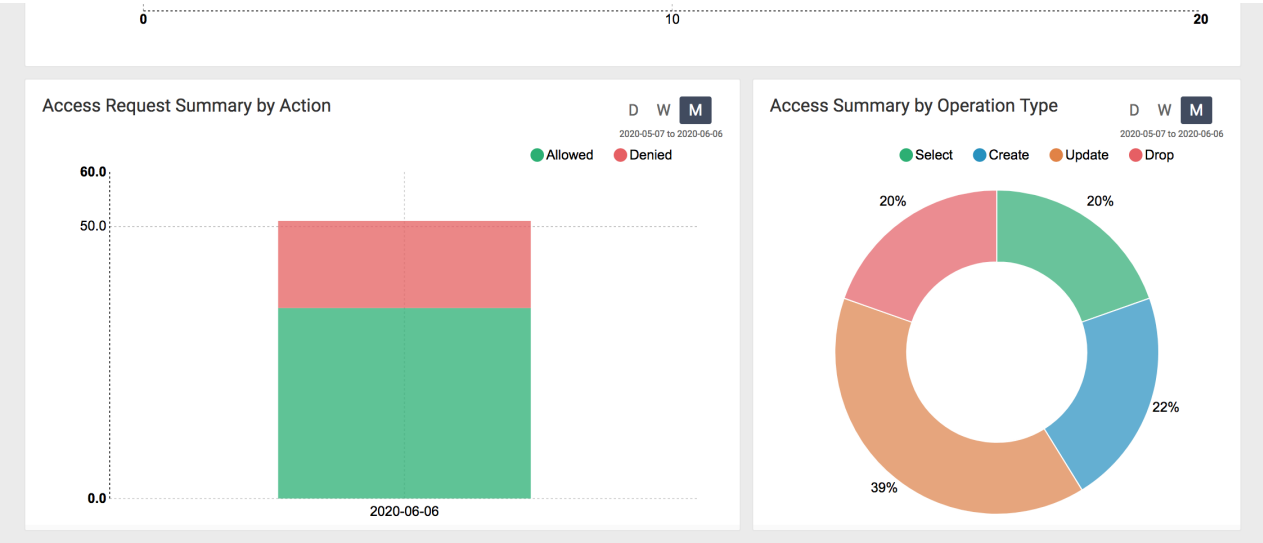

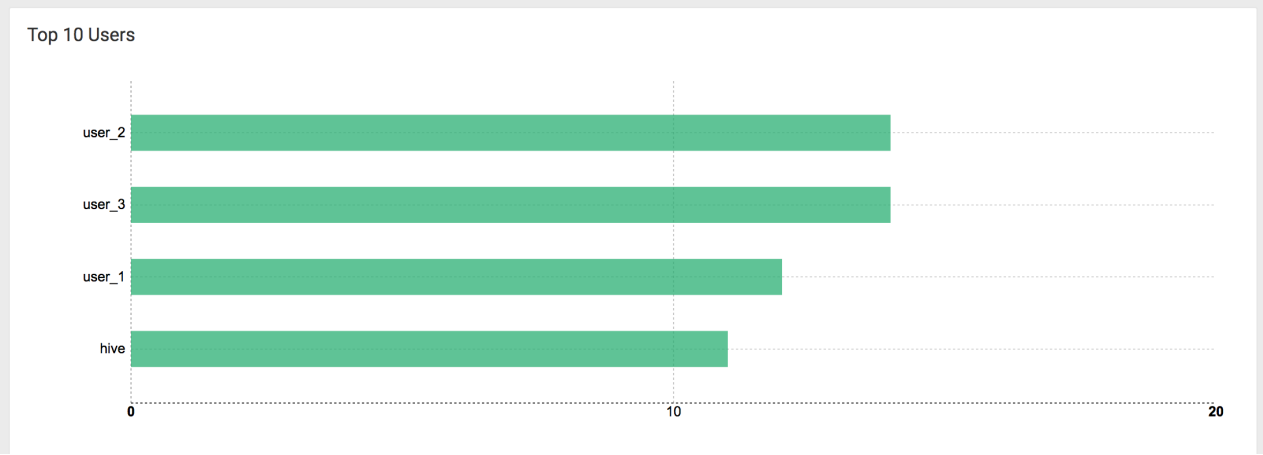

Related Information

Understanding the Hive Column Profiler

CDP Public Cloud / Data Catalog

Cloudera Data Catalog Operations

Date published: 2019-11-14
Date modified: 2025-01-16

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing Datasets.....5
Creating Datasets.....5
Editing Datasets.....7
Deleting Datasets.....7

Collaborate with other users.....7

Search for assets.....8
Filters.....9
Accessing Data Lakes.....10
Download search results as CSV files.....11
Integrating Cloudera Data Catalog with AWS Glue Data Catalog.....12
Setting up AWS Glue Catalog with Cloudera Data Catalog.....13
Working with AWS Glue.....17
Accessing AWS Glue.....17
Prerequisites for accessing Hue tables and databases.....21
Accessing Hue tables and databases.....22
Accessing Hue tables and databases from the Search page.....22
Accessing Hue tables and databases from the Asset Details page.....23
Accessing Hue assets.....24
Searching for assets using Atlas glossaries.....26
Using terms in Cloudera Data Catalog.....26
Mapping glossary terms.....27
Searching for assets using glossary terms.....29
Additional search options for asset types.....30
Searching for assets using additional search options.....32
Accessing tables based on Ranger policies.....33
Creating classifications for selected assets.....33
Additional entity type selection for searching assets.....35

Viewing Data Asset details.....36
Supported Data Asset fields for entities.....37
Navigation in Asset Details.....39
View Data Asset Access Audit Logs.....40
Navigation support for hive entities within Lineage.....41
Adding assets to one or more datasets.....42
Viewing Atlas entity audits.....44
Viewing Ranger access audits.....46
Viewing Ranger policies.....48

Managing Profilers.....48
Profiler data testing.....49
Launching profilers.....49
Launching profilers using the command-line.....53
Deleting profilers.....55

On-Demand Profilers....58
Profiling table data in non-default buckets....59
Tracking profiler jobs....60
Viewing profiler configurations....62
Ranger Audit Profiler configuration....62
Cluster Sensitivity Profiler configuration....64
Associating Cluster Sensitivity Profiler tags to Hive assets....66
Hive Column Profiler configuration....66
Understanding the Cron Expression generator....68
Backing up and restoring the profiler database....69
About the back up script....70
Running the back up script....70

Enable or disable profilers....72

Profiler Tag Rules....72

Tag management....74

Propagated asset tagging....76

Creating custom profiler rules....77
Adding custom regular expressions....78
Using DSL grammar....79

Managing Datasets

You can view, create, edit, and delete datasets to manage and govern various kinds of data objects as a single unit through a unified interface.

In Cloudera Data Catalog, click Datasets to view all the datasets.

Search for Datasets

On the **Datasets** page, enter a search string in the search box to view all asset collections with names that contain the search string.

Filter Datasets by Tags

You can view and filter datasets with tags added during dataset creation. Select the tag from the drop down list or enter the tag in the filter box. Any dataset with the filter tag assigned to a column will appear in the results.

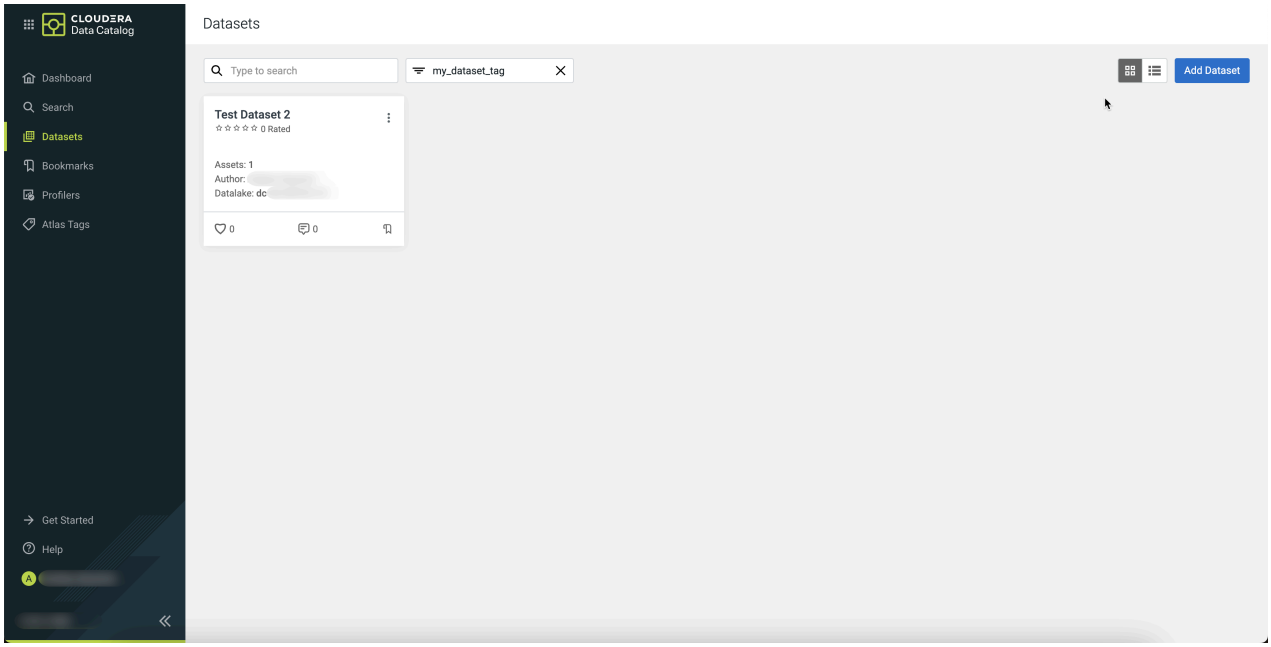

Note: The tags added during dataset creation are not synchronized to Atlas. They can be used for organization only in .

Related Information

Understanding datasets

Creating Datasets

You can group data assets into datasets. This enables you to organize data based on business classifications, purpose, protection requirements, or more. Examples of datasets are: customer profiles, sales assets, financials, PII, and HR data.

Procedure

1. From the **Datasets** page, click Add Datasets.

   The **Add** page appears.

2. Enter the following information.

Field Name	Description	Example Values
Name	Enter an appropriate dataset name. This name cannot be duplicated across the system. (Mandatory)	Customer Profiles, Sales Assets, Financials
Description	Describe the purpose or intent of the dataset. (Mandatory)	Contains customer profiles: data assets for US and WW.
Data Lake	Assign the dataset to a data lake. Choose from a list of available data lakes. (Mandatory)	dss_bbsh_clust3
Tags	Add tags to your dataset for context and subsequent lookup. Tags enable you to quickly catalog, search and retrieve asset collections in Cloudera Data Catalog, as well as, share such information with others in the future. (Optional)	se, pii, geo, finance
Public/Private	Select Public if you want other users to have access to this dataset. Select Private if only you want to have access to this dataset. **Note:** You can change the status of the asset collection later. Click the lock icon on the **Dataset Details** page to change the access state of the dataset.	Public/Private

3. Click Next.

   The **Dataset Details** page appears for the new dataset.
4. Click Add Assets to add related data assets into your dataset.

   The **Asset Search** page appears.
5. Search for assets using the search bar.
   a) Use filters to search for specific assets based on the attributes of assets. Click Filter to display the filters available.
      - **Created**: Select the time to refine the search on the basis of when the asset has been created.
      - **Owner**: Enter the name of the owner to refine the search on the basis of the owners of the assets.
      - **DB Name**: Enter the name of the database.

        Note: The data base name filter uses the "begins with" logic.
      - **Tag**: Enter the names of the tags after selecting its type (Table/Column).

   b) Select one more than one filter if needed.
   c) Click Search to view the assets.
   d) Click Reset to reset the filters and search again.
   e) From the list, click to select the assets that you like to add to your dataset.
6. Search for assets using the **Advanced** tab, if needed. Advanced search uses facets of technical and business metadata about the assets, such as those captured in Apache Atlas, to help users define and build collections of interest. Advanced search conditions are a subset of attributes for the Apache Atlas type hive_table.
7. Click Add.

   The assets are added to the dataset and the **Search** page is refreshed.
8. Close the Search tab by clicking Done.

   The Datasets Details page appears.
9. Click Save to finish editing your dataset.

Editing Datasets

You can edit datasets by adding or removing assets and changing the access state of the datasets.

Procedure

1. Click a dataset in the list to edit it. The **Details** page of that dataset appears.
2. On the **Assets** tab, click Edit to edit the content of this dataset. The dataset appears in edit mode.

   If another user is editing this dataset, an error message will appear saying that this dataset is being edited by another user and you cannot edit it.
3. Add or remove assets in the dataset.
   a) Click Add to add new assets to this dataset.
   b) Select one or more assets and click Remove to remove assets from this dataset.
4. Click Save to save the changes that you made to the dataset.
5. Click Cancel to undo any changes that you made to this dataset.

   Note: You also can edit the metadata (name, description, and tags) of the datasets. Being an owner of specific datasets, and making them private, you can update the name, description, and tags.

Deleting Datasets

You might want to delete a datasets if you no longer need to track those datasets, or if you want to reassign those assets to another dataset. You can delete datasets at any time. Deleting datasets does not delete the assets contained therein, it only disassembles the datasets. You can recreate datasets or reassign assets to new datasets.

Procedure

1. From **Datasets** page, click the ⋮ icon beside the name of the dataset you want to delete.
2. Click Delete.

   Note: Datasets can be deleted only by their creators.
3. Click Confirm.
   You are returned to the Datasets home page.

Collaborate with other users

You can collaborate and share insights with other users in the enterprise regarding various datasets.

You can rate datasets and view the average rating of a dataset. This can help other users to find datasets with higher ratings easily. You can also add your knowledge and insights about the asset collection by adding comments. Other users can respond to your comments or add their comments about each data asset collection.

On the right hand side of each dataset **Details** page, you can see additional details about the dataset. The collaboration details are also displayed in this tab. The tab displays the following details:

- The average rating for the asset collection
- The number of likes
- The number of comments
- The bookmark icon indicating if the dataset is bookmarked by the current user or not

You can perform the following collaboration actions for each dataset.

Like a dataset

You can let other users know that you like a dataset. The icon on the dataset **Details** page displays the total number of likes received by this dataset.

Click the icon to add the dataset to your list of liked collections.

Comment and discuss about a dataset

You might want to share your knowledge or insights about this dataset with other users. Cloudera Data Catalog allows you to collaborate with other users by adding comments.

Click the icon to add a comment about this dataset. The **Collaborate** tab expands. Click icon to reply to an existing comment. You can continue to add comments for each dataset.

Bookmark the dataset

In addition to sharing with other users, you can also bookmark datasets for easy access in the future.

Click the icon to add the dataset to your list of bookmarks. This dataset will appear in the list of bookmarks when you click the Bookmarks link in the left navigation menu.

Rate the dataset

You can also rate the datasets on a scale of one to five. Click the icon to rate the open dataset. The **Collaborate** tab expands.

Click the stars to provide your own rating. The rating on the **Datasets** page shows the average of the rating provided by various users. The **Rating** section also displays the number of votes given for this dataset.

View the tags of an dataset

You can add tags while creating the dataset. You can also click on the tags to search for datasets with similar tags. There are two types of tags. System tags are automatically generated based on the details of the assets in the datasets. You can add more tags that appear in the list of user generated tags.

Note: These tags do not synchronize to Atlas.

Search for assets

On the Cloudera Data Catalog **Search** page, select a data lake and enter a search string in the search box to view all the assets with details that contain the search string.

When you enter the search terms **Search**, you are looking up names, types, descriptions, and other metadata collected by Cloudera Data Catalog. The search index includes metadata (not data) about your environment and cluster data assets and operations. You can make the search more powerful by associating your own information (business metadata) to the stored assets.

Note:

For the selected data lake, click the Atlas and Ranger links to navigate to the respective base cluster services in a new browser tab.

Related Information

Understanding datasets

Filters

Use filters to refine the overview of all your available assets.

You must have access to at least one data lake to search and filter your results. By default, a data lake is already selected for you if you have access to it.

You can further refine your search results using filters as follows:

Owner

From all the owner names that appear, you can select the owner to further refine the results and display those search results with the selected owner.

Type

Select an entity type to view all the assets stored in that type of database.

- Azure BLOB
- Azure Container
- Azure Directory
- AWS S3 Bucket
- AWS S3 Object
- AWS S3 Pseudo Dir
- AWS S3 V2 Bucket
- AWS S3 V2 Directory
- AWS S3 V2 Object
- Hbase Column Family
- Hbase Namesspace
- Hbase Table
- HDFS path
- Hive Column
- Hive DB
- Hive Table
- Iceberg Column
- Iceberg Table
- Impala Column Lineage
- Impala Process
- Impala Process Execution
- Kafka topic
- ML Model Build
- ML Model Deployment
- ML Project
- RDBMS Column
- RDBMS DB
- RDBMS Foreign key
- RDBMS Index
- RDBMS Table
- Spark Application
- Spark Column
- Spark Column Lineage
- Spark DB
- Spark ML Directory
- Spark ML Model
- Spark ML Pipeline

- Spark Process
- Spark Process Execution
- Spark Table

Note: After selecting an entity type, further filters related to that type will be available under the More filter. For example, selecting the Hive Table type will enable the Column Tag filter.

Entity Tag

Use entity tags to refine your search results. You can add business metadata as entity tags in Atlas as classifications, or in the **Atlas Tags** menu. Use these tags to refine your search results and view the details of the required data asset.

Time Range

You can filter your assets by the **Created On** date (if provided by Atlas) after selecting and asset Type. Use the calendar widget to select a range and click Apply.

Glossary Terms

You can filter assets based on business glossary terms. You can search for any asset without any entity type restrictions.

Note: This filter appears only if Atlas has terms set up.

Click Cancel for any filter to clear the selection or Clear All to reset all your filters.

In the resulting list of your matching assets, you can click a row and see the following:

- **Qualified name**
- **Database**
- **Classification**
- **Terms**

Clicking the Name of the entity will open its **Asset Details**.

Accessing Data Lakes

In the **Search** page, the accessible data lakes are displayed in a drop-down.

Users have access to the lakes based on the permissions that are granted. You can choose the available lake by selecting the appropriate radio button.

For example, in the following diagram, the logged in user has access to all the listed data lakes.

Note: You can search the assets of one data lake at a time.

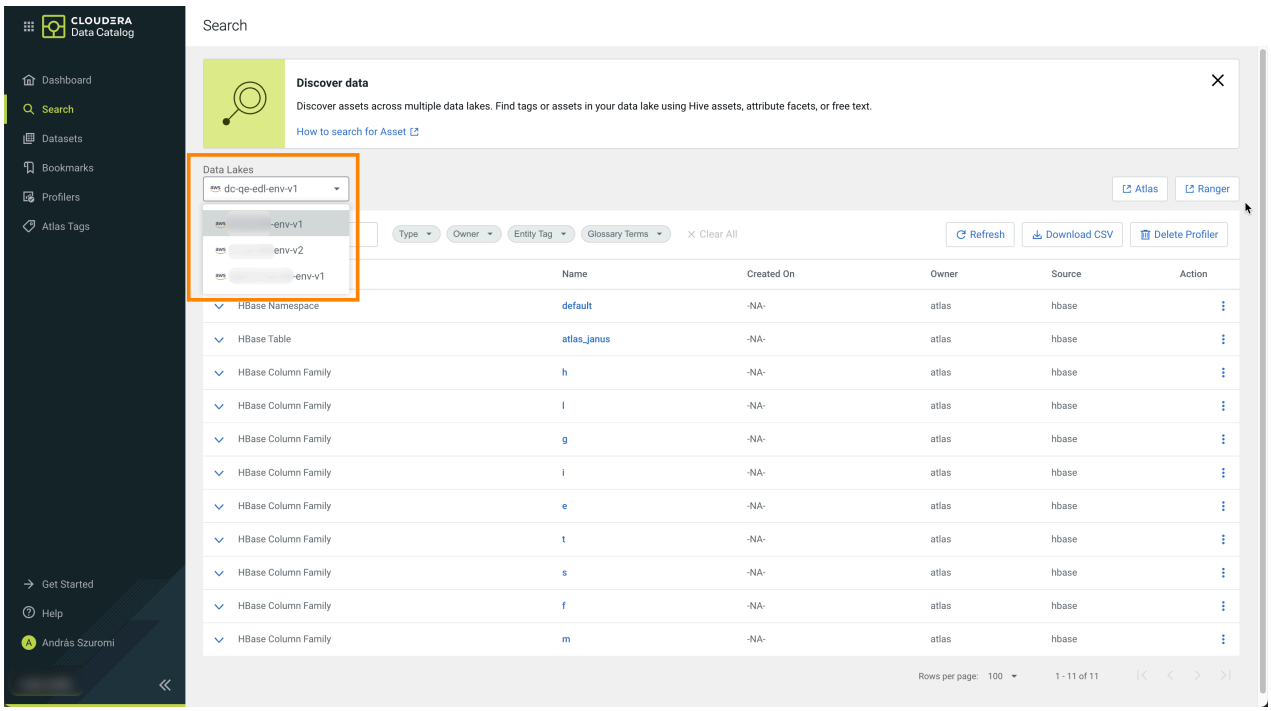

Related Information

Introduction to data lakes

Understanding data lake details

Download search results as CSV files

You can download the search result for the current query with the selected data lake. The feature allows you to download up to 10000 rows for the current search query.

The CSV file format does not conform to any specific order or continuation in the downloaded results. For example, a user can download 10000 assets and later downloads the results for the same query again, then the downloaded CSV files may not contain the search results in the same order as it was downloaded previously.

Click Download CSV to start your download:

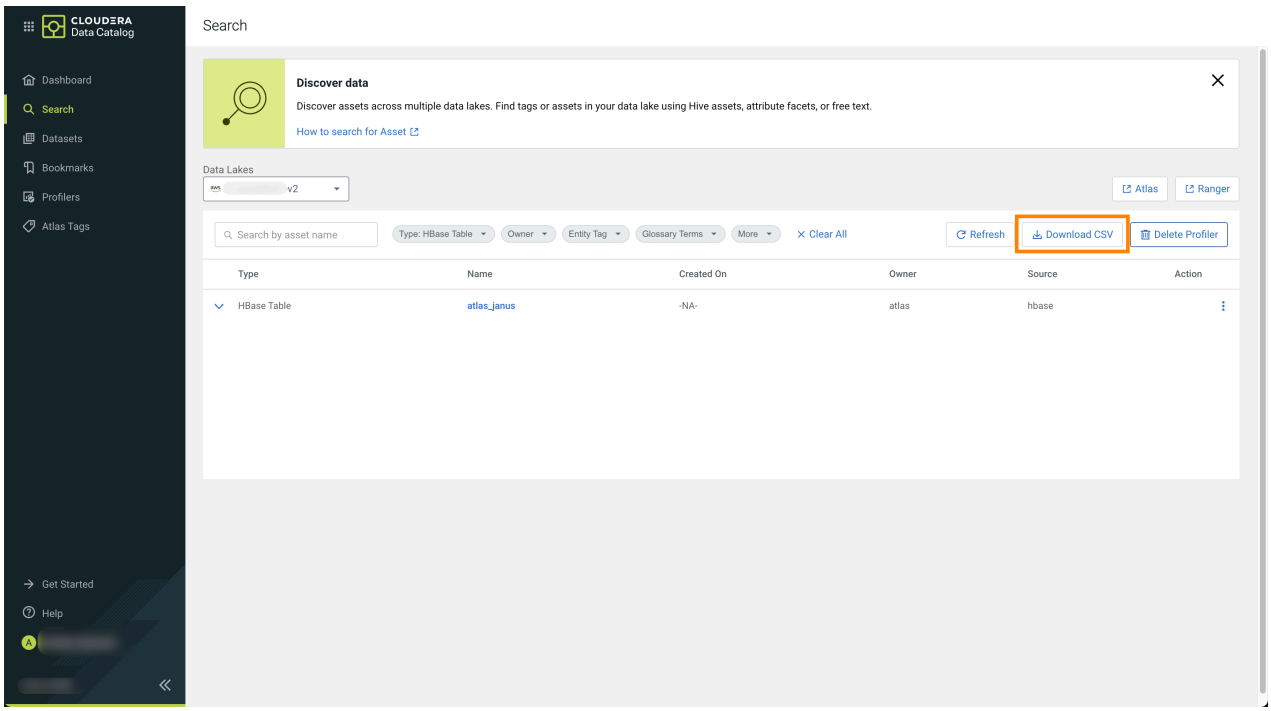

Integrating Cloudera Data Catalog with AWS Glue Data Catalog

Integrating Cloudera Data Catalog with AWS Glue Catalog enables the users to browse and discover data as well as register data into Cloudera Shared Data Experience (through metadata translation or copy), so that it can be used with Cloudera Data Hub and other relevant experiences.

Note:

- This feature is only available in VM-based environments.
- This feature is available but is not ready for production deployment. Cloudera encourages you to explore this technical preview feature in non-production environments and provide feedback on your experiences through the Cloudera Community Forums or through your account managers and field team contacts.

While using AWS Glue in Cloudera Data Catalog, you will be able to experience a complete snapshot metadata view, along with other visible attributes that can power your data governance capabilities.

How integration works

Assuming that the Cloudera Shared Data Experience is running in the users' AWS account (that contains the same AWS account which has Glue Data Catalog and the data that has to be discovered), the credentials with the Exte rnalDataDiscoveryService (which is hosted in Cloudera Shared Data Experience) must be shared, so that these two entities can interact with each other. These credentials are used to launch Cloudera Shared Data Experience and other workload clusters on the users' AWS account.

Prerequisites:

- You must have full access to AWS Glue Catalog and also have access to the EMR cluster's Hive Metastore instance.
- You must set up the Cloudera platform.
- You must have access to your AWS IT Admin and CDP Admin user credentials, which is required to enable the Cloudera platform to access AWS/EMR managed data in Cloudera.

Note: AWS policies are managed by AWS IAM and the AWS roles are added to the Cloudera Management Console. Refer to Using Instance Profile in AWS and Using credentials in Cloudera Management Console. For more information about the AWS access, see AWS Environments.

Setting up AWS Glue Catalog with Cloudera Data Catalog

You must map your Cloudera Data Catalog instance with AWS Glue Catalog.

Procedure

1. Enable the entitlement for your Cloudera Data Catalog instance by running the following command on your Cloudera environment. For example:

```
$ cdp coreadmin grant-entitlement --entitlement-name
DATA_CATALOG_ENABLE_AWS_GLUE --account-id {account_id}
```

2. Add the relevant permissions in the corresponding AWS account:
   a) Include permission to access Glue Catalog service by editing the policy accordingly.

      Make a note of the Assumer Instance Profile role that you intend to use and include full access authorization for AWS Glue.

      Refer to the following images as a guidance to complete the set up.

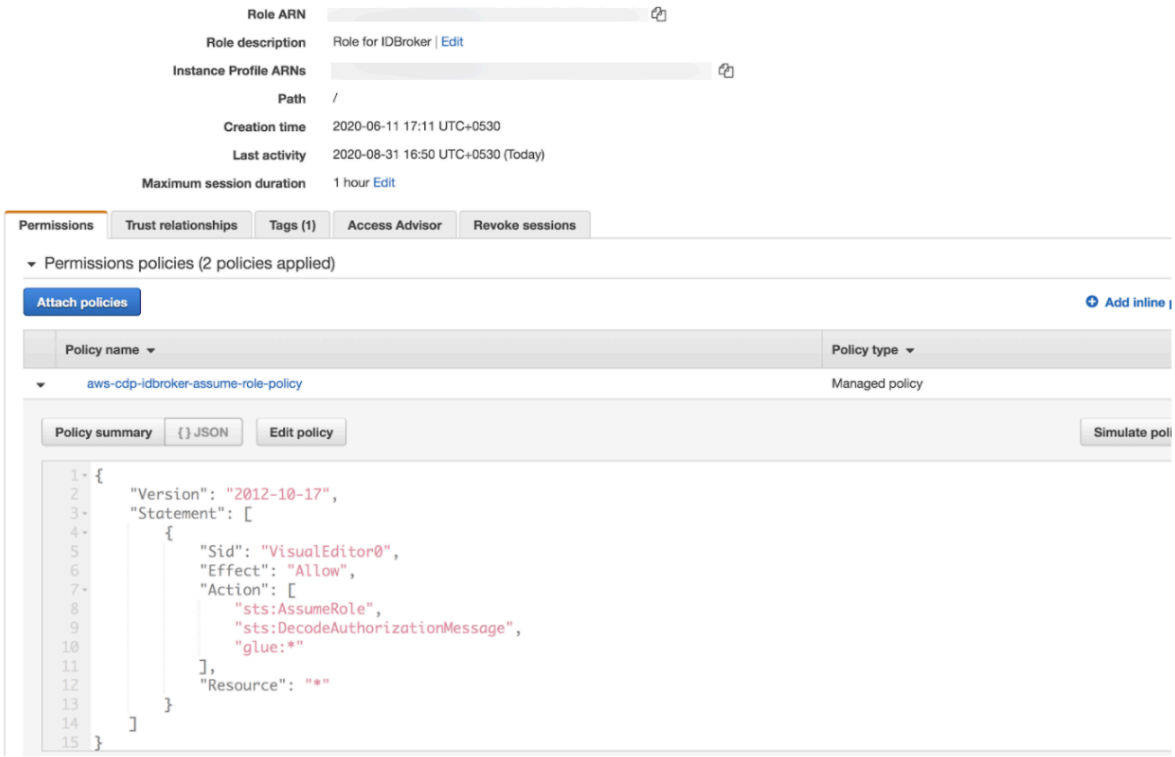

Note: For Role ARN and Instance Profile ARNs, you must include the appropriate account number and role respectively.

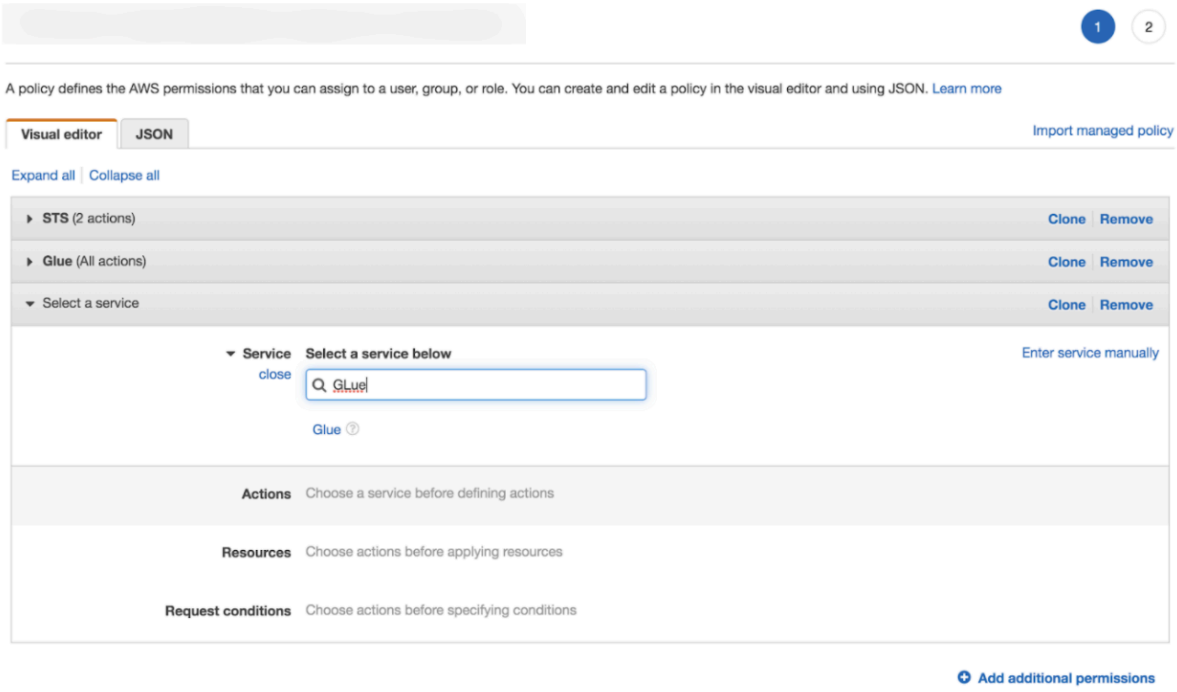

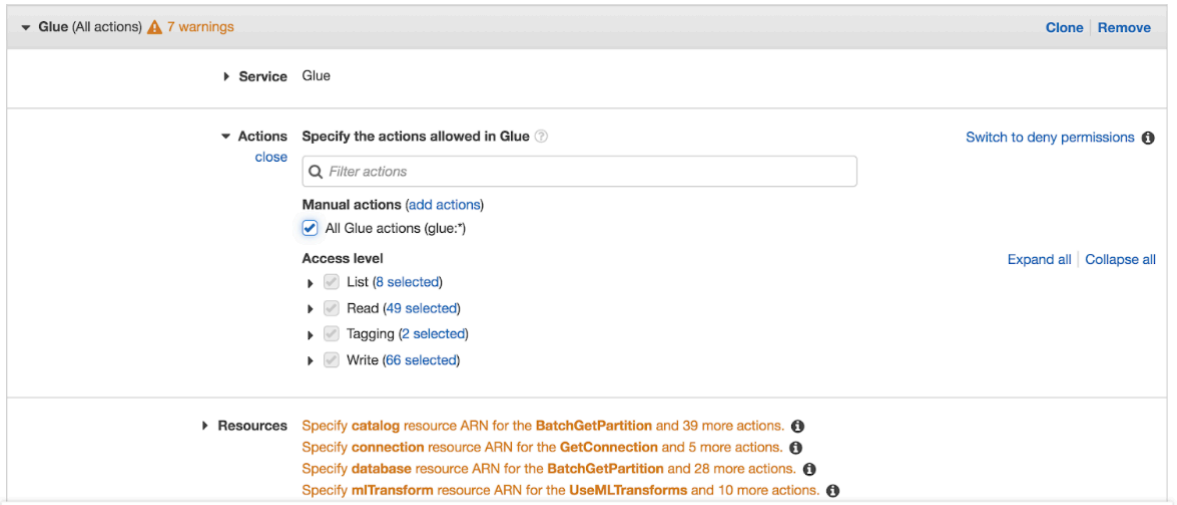

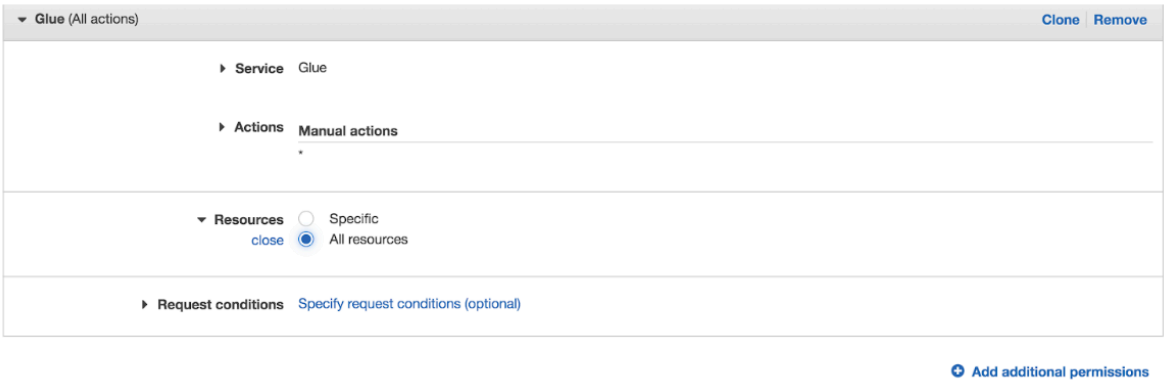

b) Search for the role attached to the Instance Profile of the Cloudera environment. Use the Instance Profile that you have configured above with Glue related policy in your AWS Environment creation command.

Use the following examples to setup AWS environment and AWS data lake as part of the Glue setup:

```
cdp environments create-aws-environment --profile default --cli-input-js
on '
{"environmentName”:”ab-ds-cli-7321”,
 "credentialName”:”cd2d-1234”,
 "Region":"us-region-2”,
 "securityAccess":{-insert the value--"},
 "Authentication":{---insert the value---"},
 "logStorage":{"storageLocationBase":"s3a://demo-e2e-test-state-bucket/
ab-ds-cli-7321/logs","instanceProfile":"arn:aws:iam::<xxxxxxxxxxx>:insta
nce-profile/<role-name>"},
 "vpcId":"vpc-0123456”,
 "subnetIds":["subnet-04fe923b902aa5cf2","subnet-099c7a631f0ebed3c"],
 "s3GuardTableName":"dc-pro-cli-7210",
 "Description":"ab-ds-cli-7321",
"enableTunnel":false,
 "workloadAnalytics":false,
 "freeIpa":{"instanceCountByGroup":1},
 }'

cdp environments set-id-broker-mappings \
--environment-name "ab-ds-cli-7321" \
--profile default \
--set-empty-mappings \
--data-access-role arn:aws:iam::<xxxxxxxxxxxx>:role/add-role \
--ranger-audit-role arn:aws:iam::<xxxxxxxxxxxx>:role/add-role
```

Similarly, while setting up the data lake use the Instance Profile that you configured above with Glue related policy in your data lake creation command:

```
cdp datalake create-aws-datalake --profile default --runtime 7.2.12 --cl
i-input-json '
{"datalakeName":"ab-ds-cli-7321-sdx",
 "environmentName":"ab-ds-cli-7321",
 "cloudProviderConfiguration":{"instanceProfile":"arn:aws:iam::<xxxxxx
xxxxx>:instance-profile/<role-name>","storageBucketLocation":"s3a://demo
-e2e-test-state-bucket/ab-ds-cli-7321"},
 "scale":"LIGHT_DUTY",
 }'
```

For more information, see Creating an AWS environment with a medium duty data lake using the CLI.

c) Navigate to the attached policy for the role.

d) When you manually create tables in AWS Glue Data Catalog, you must set the **fully qualified path** for the table location.

For example: s3://my-aws-server-node-1/something/something.amazonaws.com/dc-pro-721-storage/glue/

3. 3. You must set up the AWS Glue Data Catalog. For more information, see Populating the Glue Data Catalog. You must select only the CSV format which is currently supported for Cloudera Data Catalog and the delimiter which is used in the data.

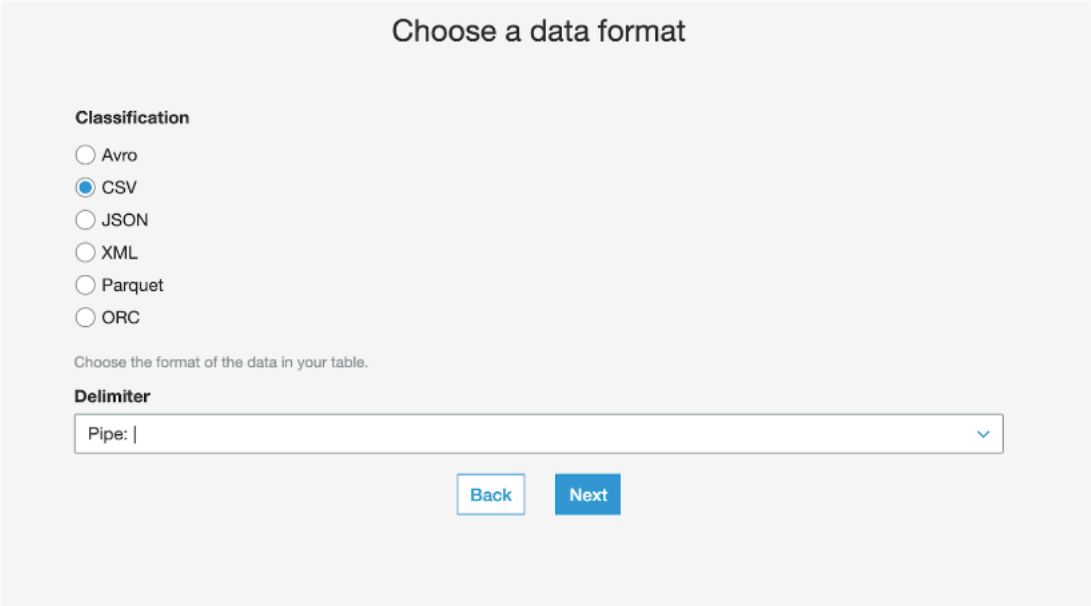

4. While creating tables in AWS Glue Data Catalog manually, set the fully qualified path for location. For example: s3://my-aws-server-node-1/something/something/dc-pro-721-storage/glue/

What to do next

AWS Glue metadata must be registered with Cloudera Data Catalog.

Working with AWS Glue

The AWS Glue metadata must be registered with the Cloudera Data Catalog. The Glue contains the metadata that is synchronized with Cloudera Data Catalog. The Glue metadata is accessed using the data lake option in the Cloudera Data Catalog. After setting up AWS Glue with Cloudera Data Catalog and once the Glue synchronization with Cloudera Data Platform is complete, the AWS Glue data lake appears.

Note: For each AWS environment, there would be a separate listing under the data lake option. The Glue data lake name / identifier in Cloudera Data Catalog follows the format: <glue: Data Lake Name>. The Glue assets are of the type: GLUE EXTERNAL TABLE

Accessing AWS Glue

To access the Glue metadata in Cloudera Data Catalog, you must note the following in your instance.

- List the Glue Metadata by selecting the Glue data lake.
- Select one or more Glue assets and register the same with Cloudera platform.
- Verify if the registered Glue assets are listed in the Cloudera Data Catalog owned data lake.
- Select the registered Glue asset and click to open the **Asset Details** page.

Listing the Glue assets

In Cloudera Data Catalog, when you select the AWS Glue data lake, you can view the list of Glue assets. These metadata assets are directly sourced from Glue.

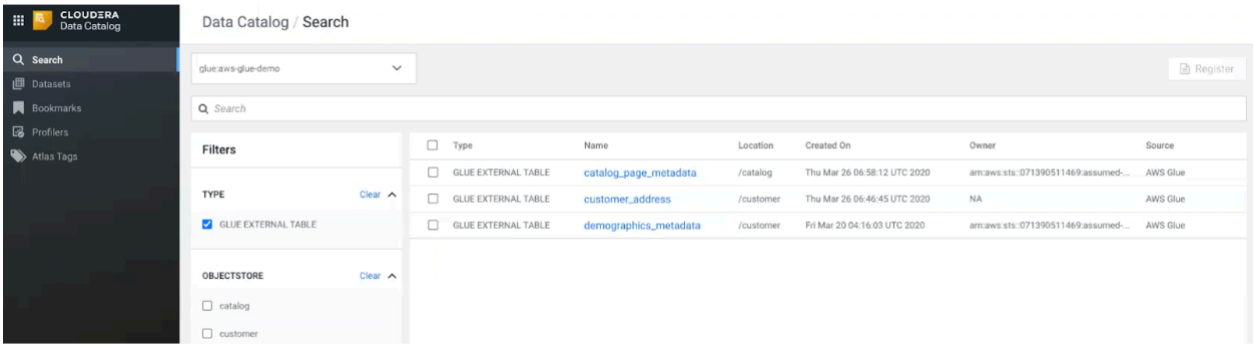

When you click on one of the assets, the **Asset Details** page is displayed.

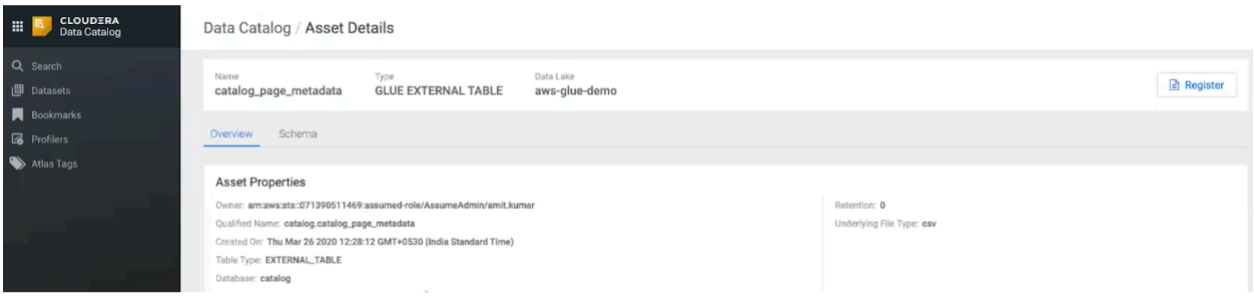

Next, on the main Cloudera Data Catalog page, you must select the Glue data lake and select one of the Glue assets and register the asset to the Cloudera platform. Click Register.

Optionally, you directly click on the Glue asset and register the asset on the **Asset Details** page.

Note: You can select one or more Glue assets on the asset listing page to register them in Cloudera.

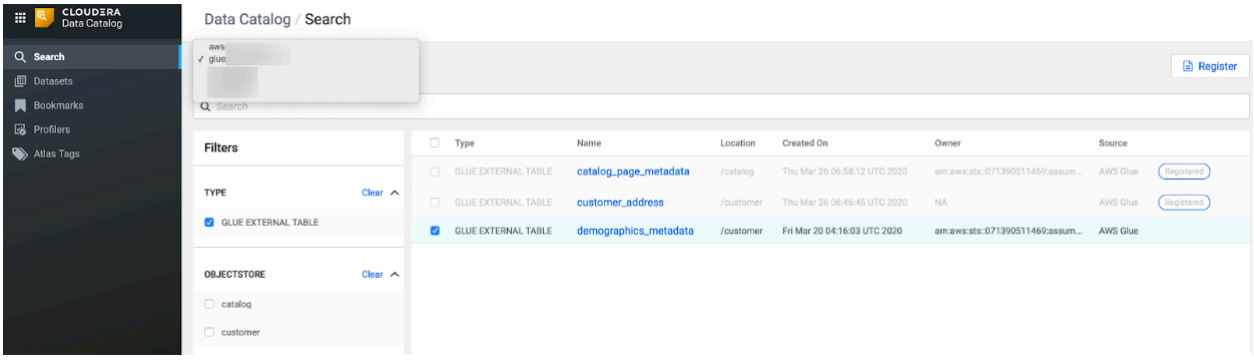

Once the Glue asset is registered, the asset is imported into Cloudera.

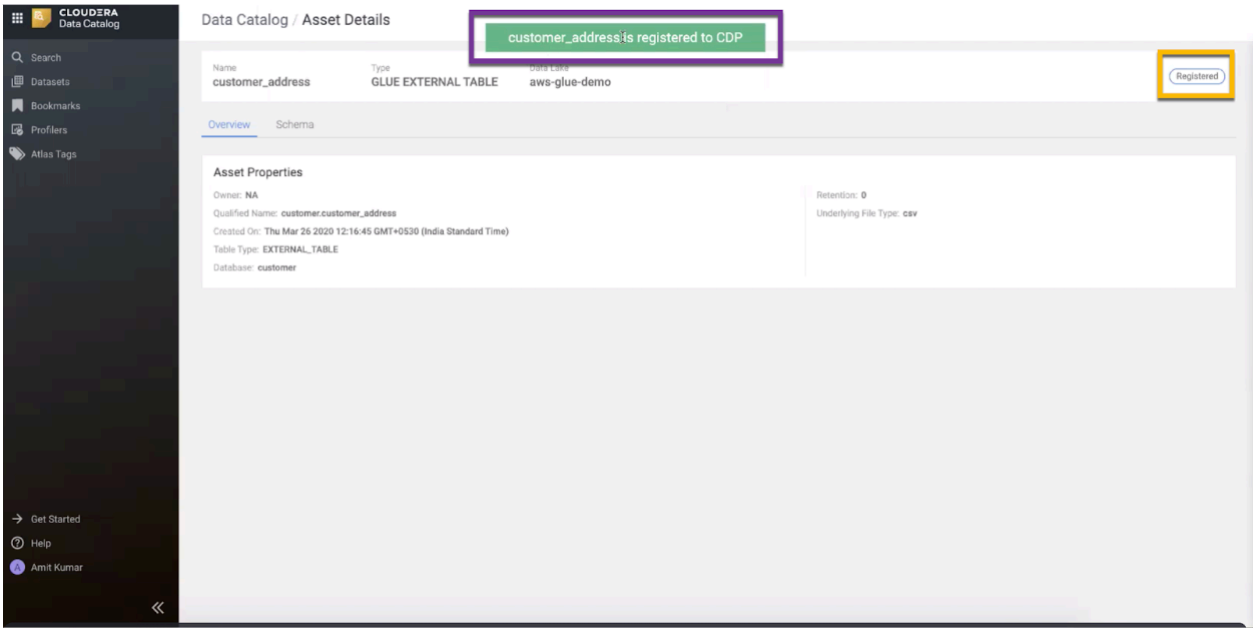

Next, navigate back to the Cloudera Data Catalog main page and select the Cloudera Data Catalog owned data lake and select the type as Hive Table. The search results lists all the Hive table assets and you can view the Glue registered asset(s) as well. The registered Glue asset can be identified using

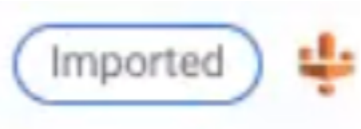

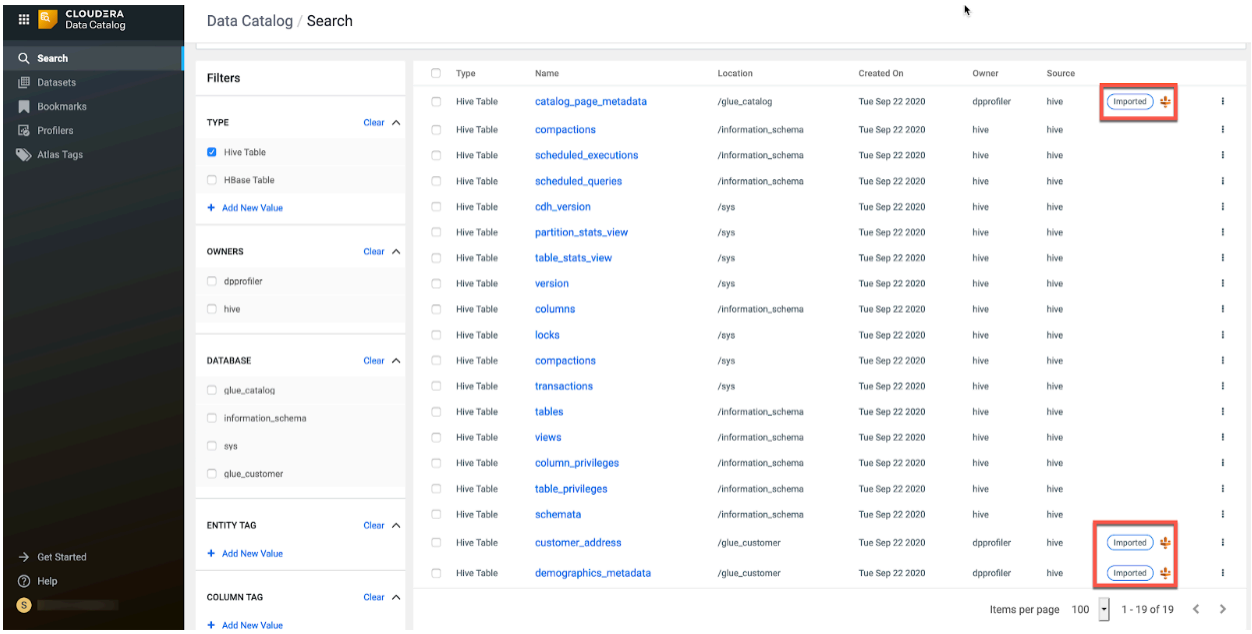

Note: The entries in Cloudera Data Catalog for Glue assets are created in the Hive Metastore.

Click on the Glue registered asset and you can view the **Asset Details** page for the selected Glue asset.

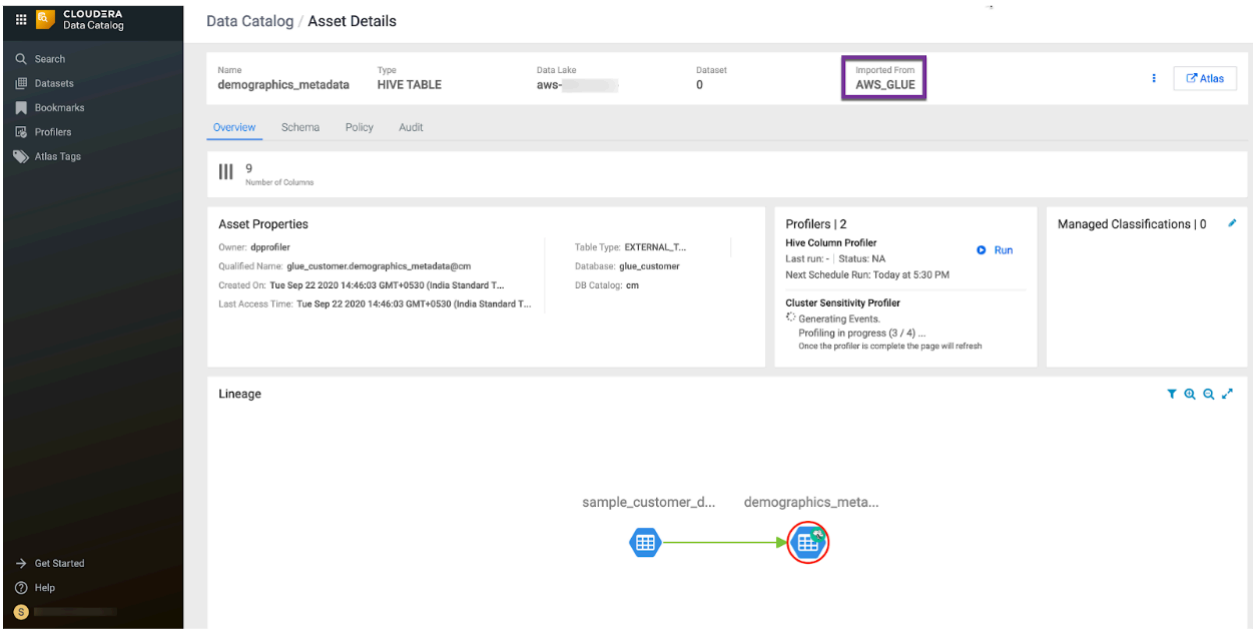

The **Asset Details** page for the Glue asset is populated by Atlas. While registering the Glue data, the data was written to the Hive Metastore and later Atlas synchronised the metadata.

Go back to the main Cloudera Data Catalog page and select the Glue data lake. Note that the registered Glue asset(s) are greyed out or cannot be selected again.

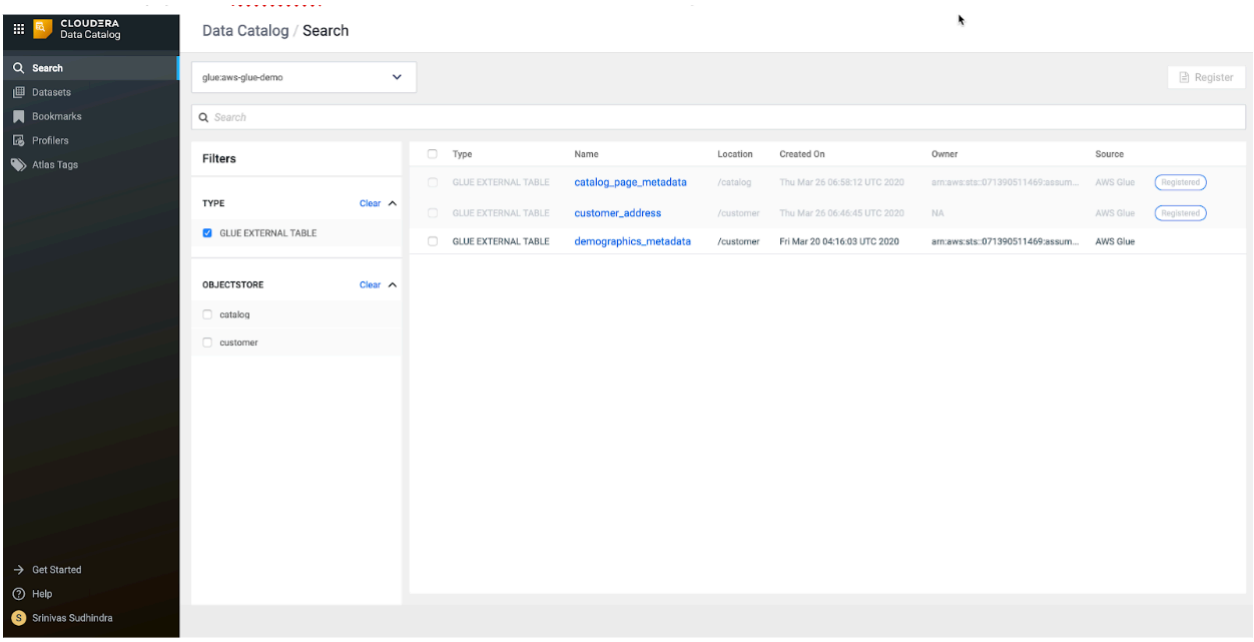

You can still view the registered Glue assets (powered by Atlas) by clicking on the same and it navigates to the **Asset Details** page as seen above in the image.

Working with Ranger Authorization Service (RAZ) enabled AWS environment

For RAZ enabled AWS environment, you must employ the following permission settings to work with Cloudera Data Catalog - Glue integration.

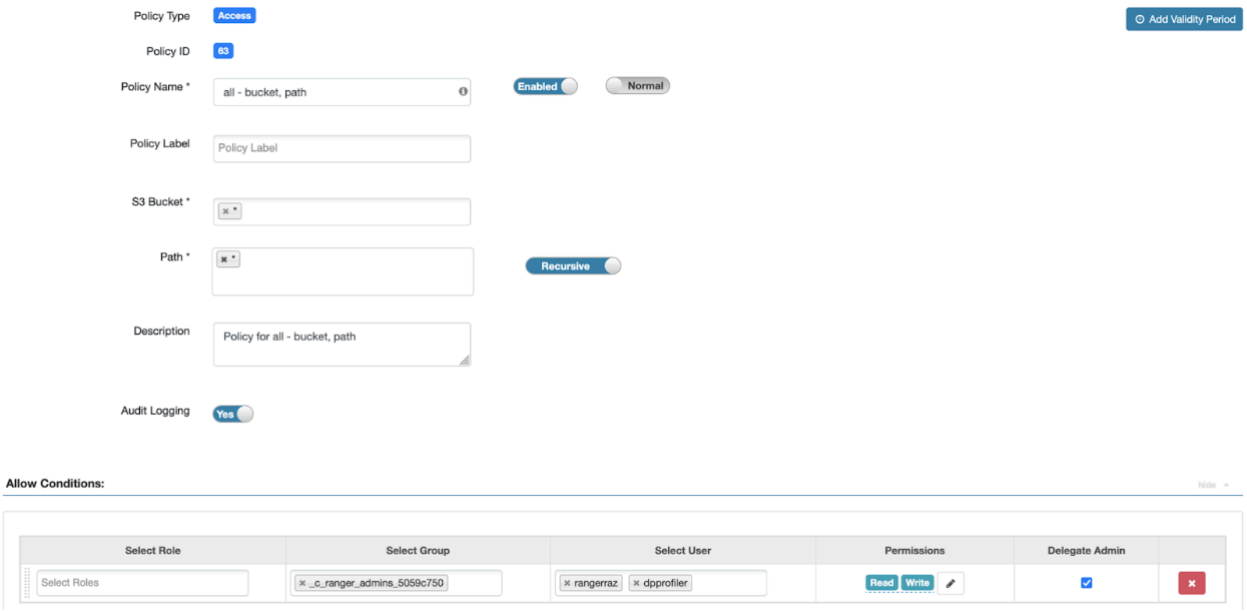

Note: By default, the dpprofiler user is not included in the allowed users list. You must manually add the dpprofiler user in the allowed users list.

Prerequisites for accessing Hue tables and databases

Apache Hue is a part of the Cloudera Data Warehouse service. Before you can access Apache Hue from Cloudera Data Catalog, you must set up the Cloudera Data Warehouse service.

Before you begin

You must have the DWAdmin role to perform these tasks. For more information, see Granting CDP users and groups access to CDW.

Procedure

1. Set up an AWS or Azure environment in Cloudera Data Warehouse that has been activated and registered in the Cloudera Management Console:

   Activating an AWS or Azure environment in Cloudera Data Warehouse enables using the existing Data Lakes that are present in the environment. This also creates a default Database Catalog that is used by a Virtual Warehouse.

   For more information, see Activating AWS environments or Activating Azure environments. Currently, Cloudera Data Warehouse only supports AWS and Azure Cloud environments
2. Set up a Hive or an Impala Virtual Warehouse created using Cloudera Data Warehouse. Depending on your data set type, create a Hive or an Impala Virtual Warehouse using Cloudera Data Warehouse.

   For more information, see Adding a new Virtual Warehouse.
3. Select the asset type.

   The asset type must be either a Hive Table or Hive DB.

   After these conditions are met, Apache Hue links for the selected assets are displayed in Cloudera Data Catalog.

Related Information

Giving access to Data Warehouse users

Accessing Hue tables and databases

When you log into Cloudera Data Catalog, you can access web links to the Hue instances from the Search page or the Asset Details page.

Before you begin

You can access the Hue tables and databases by using one of the following options.

Procedure

1. Log into Cloudera Data Catalog.
2. When you navigate to the entity search page, you can click on the icon present at the end of the row which opens the Link to Experiences dialog.
3. Directly on the selected asset's **Asset Details** page, you can click Link to Experiences.

Accessing Hue tables and databases from the Search page

When you log into Cloudera Data Catalog, you can access web links to the Apache Hue instances from the Search page.

Procedure

1. Log into Cloudera Data Catalog.
2. Select the Data Lake associated with your environment.
3. Select the asset type under Filters, for example Hive Table.
   Hive tables present within the Data Lake are displayed.
4. Click the icon at the end of the row for a table that you want to display in Hue.

   A Link to Experiences pop-up appears. It contains a list of Hue instances within the virtual warehouses that are associated with this particular table.
5. Click the link to open Hue web interface.
   The Hive table is displayed on the Hue Table Browser.

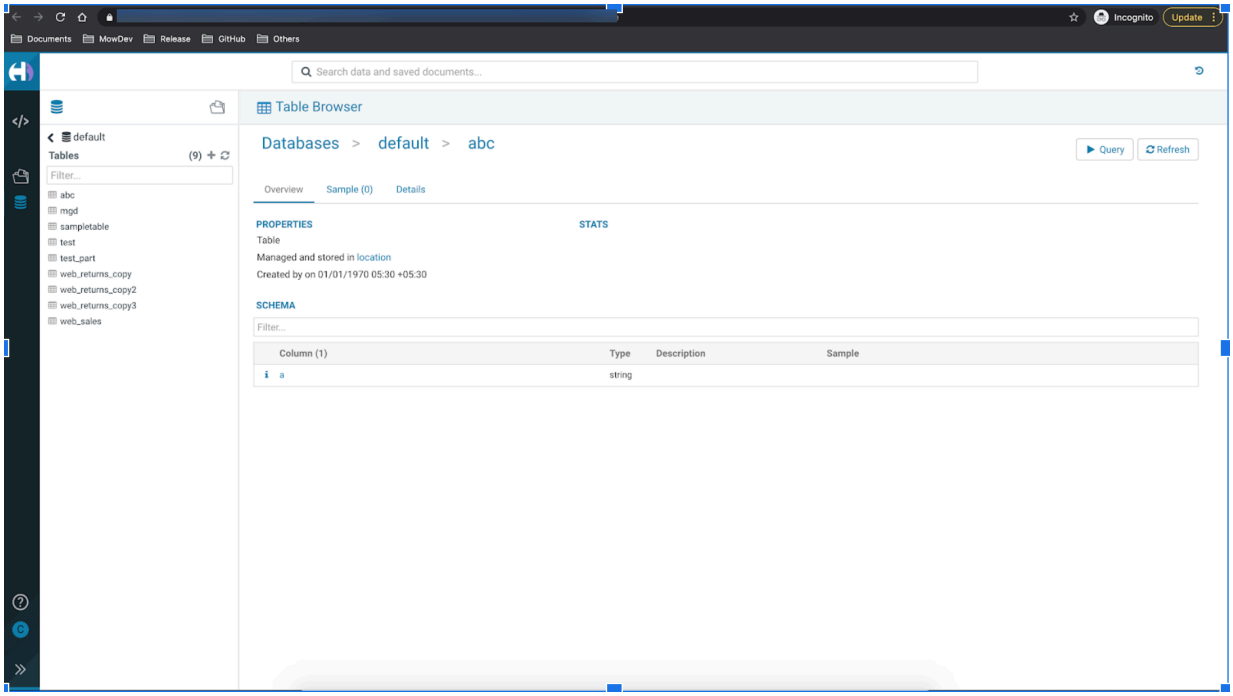

Accessing Hue tables and databases from the Asset Details page

When you log into Cloudera Data Catalog, you can access web links to the Apache Hue instances from the Asset Details page.

Procedure

1. Log into Cloudera Data Catalog.
2. Select the Data Lake associated with your environment.
3. Select the asset type under Filters, for example Hive Table.
4. Click on the asset name to open the **Asset Details** page. The **Asset Details** page contains the Link to Experience button as shown in the following image:

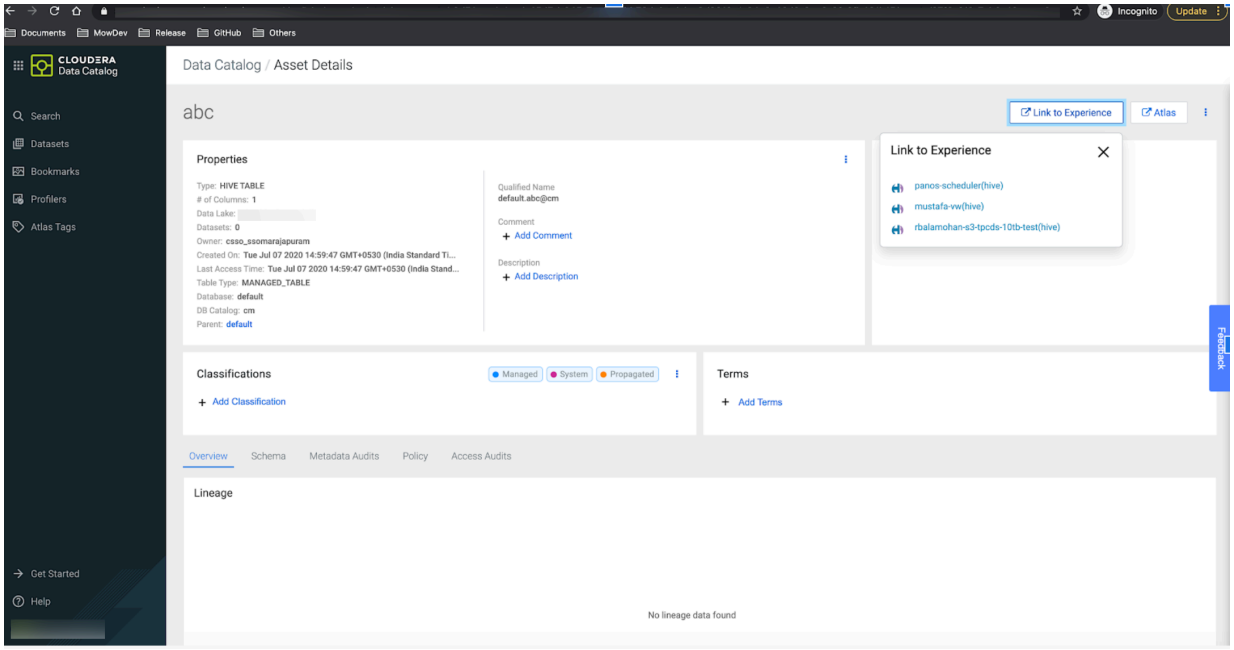

5. Click Link to Experience.
   The pop-up contains a list of Hue instances within the Virtual Warehouses that are associated with this particular table.

6. Click the link to open Hue web interface.
   The Hive table is displayed on the Hue Table Browser.

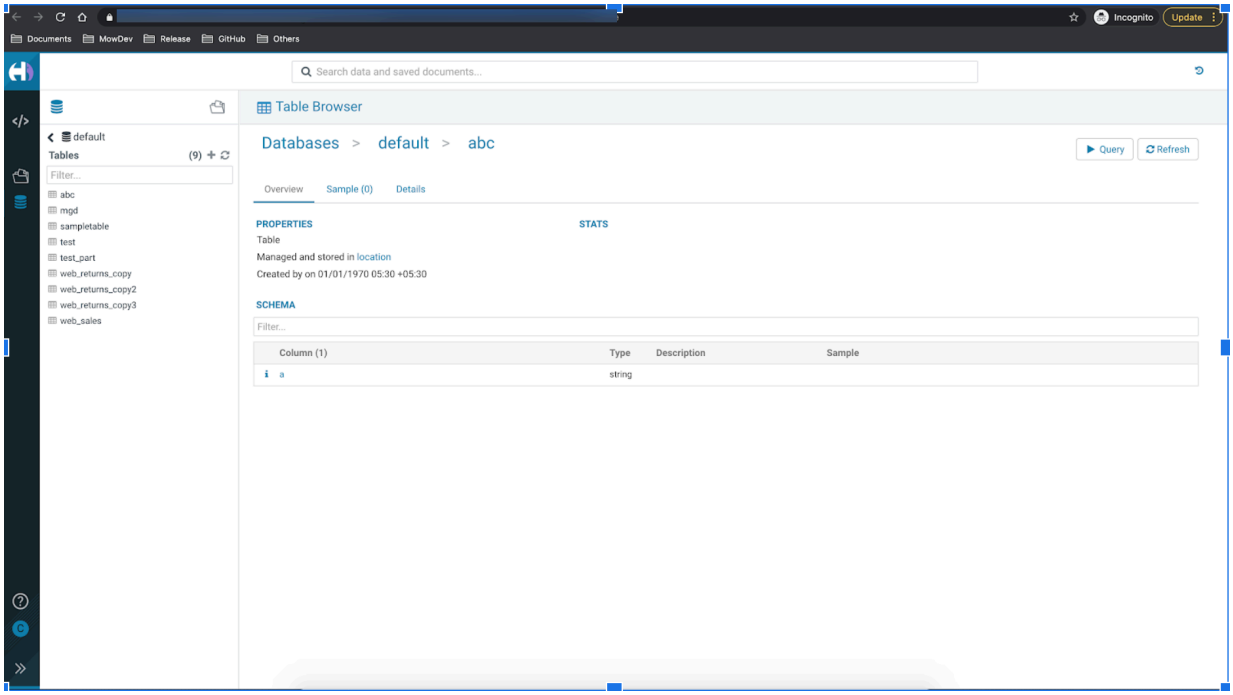

Accessing Hue assets

Cloudera Data Catalog helps you to explore assets within your Data Lake. Apache Hue allows you to run SQL queries on Hive assets. The integration between Cloudera Data Catalog and Cloudera Data Warehouse service provides a direct link to the Hue instance from the Search page or the Asset Details.

Before you begin

You must fulfill all conditions listed in Prerequisites for accessing Hue tables and databases on page 21 for viewing the external link button.

You must activate Cloudera Data Warehouse on your selected environment.

Procedure

1. Go to Cloudera Data Warehouse.
2. On the Overview page, search for your AWS or Azure environment.

3. Locate the enabled AWS / Azure environment and click Activate.
   This action also creates a default Database Catalogs.

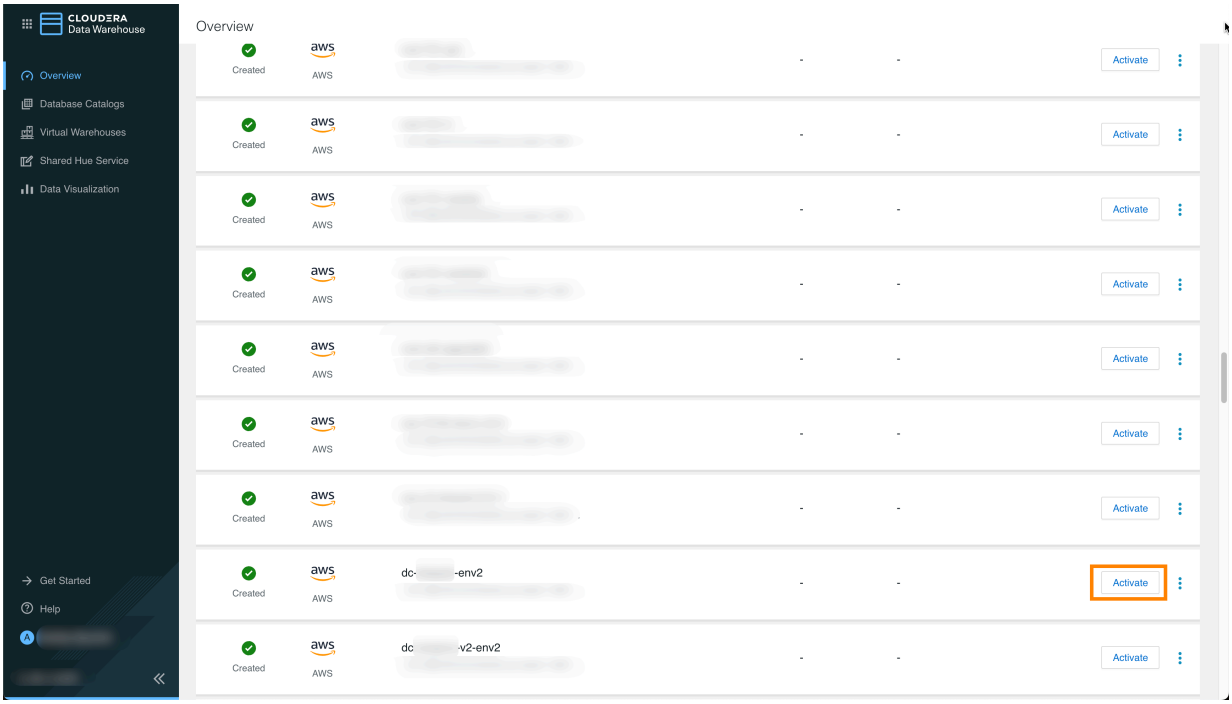

4. Once the environment is activated and a DB Catalog is created and running, you can create a virtual warehouse as per the requirements.

   For more information about AWS and Azure Environments, see AWS environments overview and Azure environments overview.

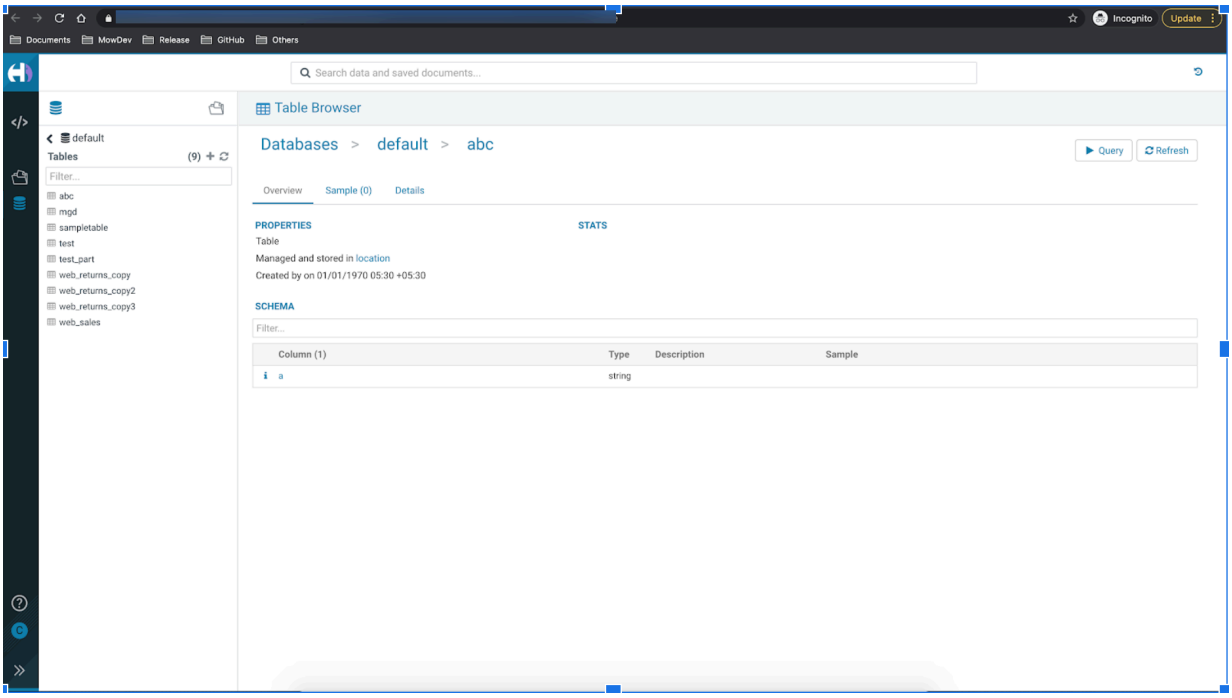

   You will now be able to navigate to the Hue assets on these virtual warehouses using the links provided by Cloudera Data Catalog.

Related Information

Setting up private networking in AWS environments

Searching for assets using Atlas glossaries

Use Apache Atlas glossaries to define a common set of search terms that data users across your organization use to describe their data.

Data can describe a wide variety of content: lists of names or text or columns full of numbers. You can use algorithms to describe data as having a specific pattern, of being within a range or having wide variation, but what's missing from these descriptions is what does the data mean in a given business context and what is it used for? Is this column of integers the count of pallets that entered a warehouse on a given day or number of visitors for each room in a conference center?

The glossary is a way to organize the context information that your business uses to make sense of your data beyond what can be figured out just by looking at the content. The glossary holds the terms you've agreed upon across your organization so business users can use familiar terms to find what they are looking for.

Glossaries enable you to define a hierarchical set of business terms that represents your business domain.

Glossary terms can be thought of as of a flat (but searchable) list of business terms organized by glossaries. Unlike classifications, terms are not propagated through lineage relationships: the context of the term is what's important, so propagation may or may not make sense.

You can search for the datasets using the Glossary Terms filter available on the **Search** page.

Related Information

Glossaries overview

Using terms in Cloudera Data Catalog

You can use the Asset Details page to add or modify Apache Atlas glossary terms for your selected assets.

Use Atlas to define rich glossary vocabularies using the natural terminology (technical terms and/or business terms) of your industry. You can also create semantic relationships between your terms. Then, in Cloudera Data Catalog, use the **Terms** widget in the **Asset Details** page to map assets to glossary terms.

You can use terms in Cloudera Data Catalog to search for entities, filter them by glossary term(s), and also search for entities associated with them in Atlas.

Note: When you work with terms in Cloudera Data Catalog and map them to your assets, you can search for the same datasets in Atlas by using the corresponding terms.

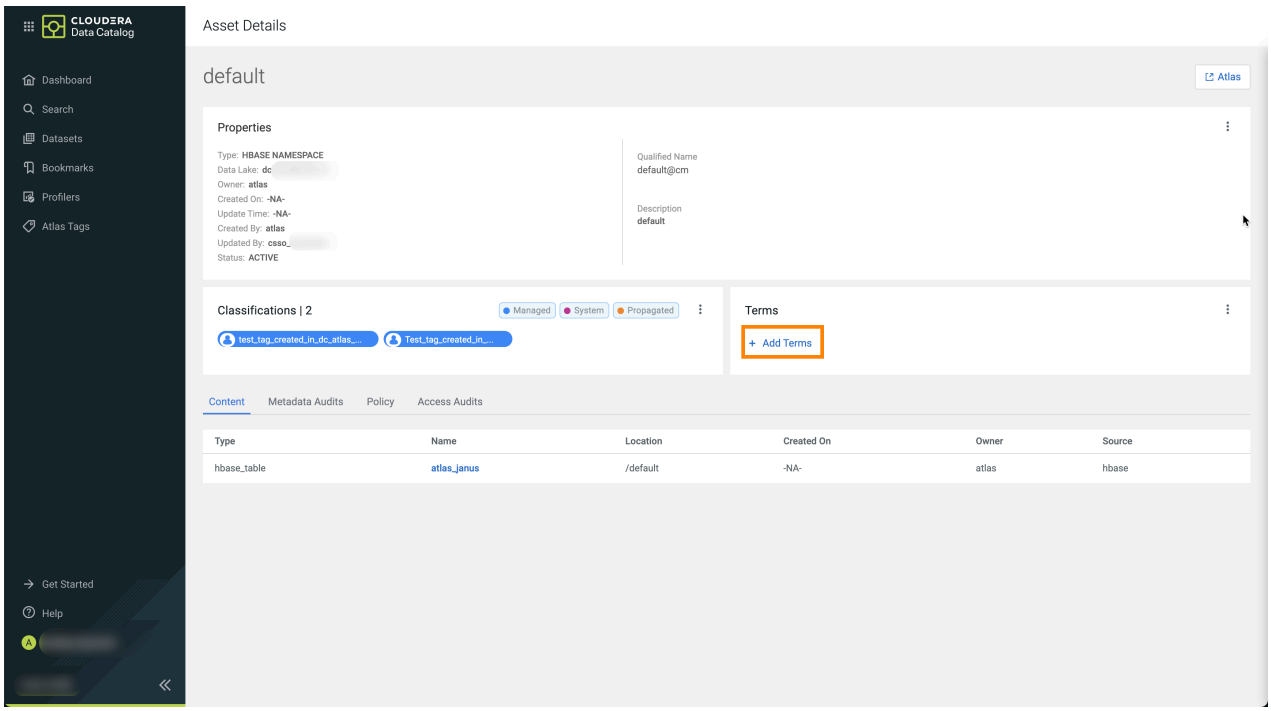

Related Information

Associating terms with entities

Defining related terms

Mapping glossary terms

Cloudera Data Catalog contains the glossary terms that are created in Apache Atlas.

You can search for those terms in Cloudera Data Catalog and map specific terms with assets. You can also search for terms to delete them from the selected asset. The selected asset displays the total number of terms associated or mapped accordingly.

When you map a specific term for your dataset, the term is displayed in the following format:

```
<termname>@glossaryname>
```

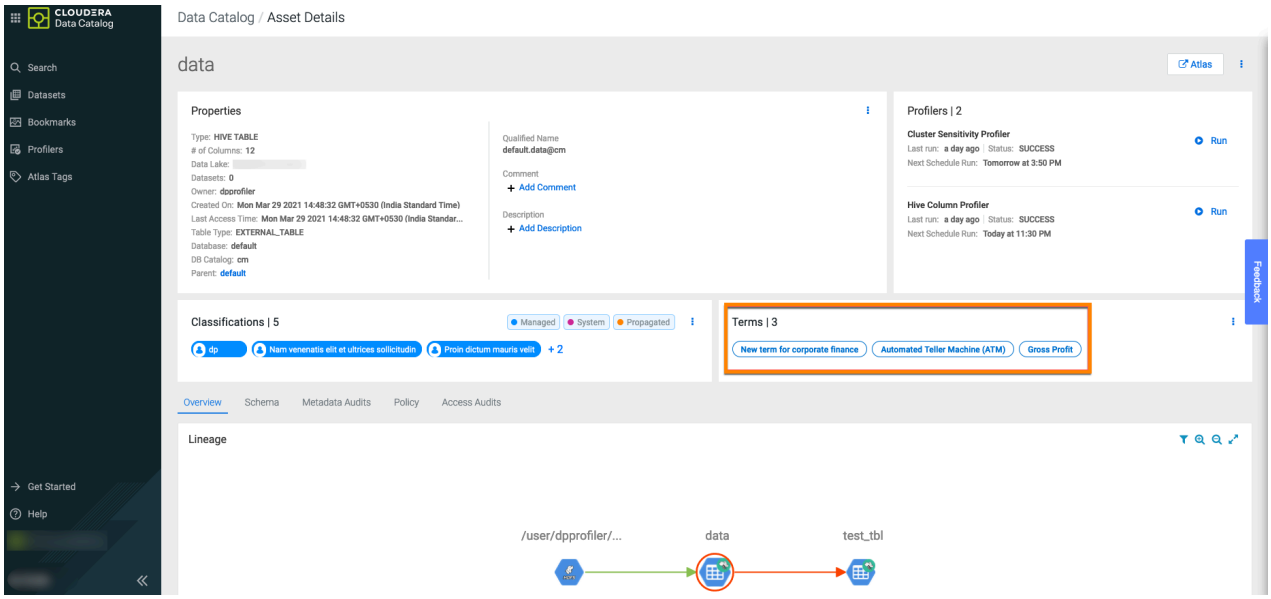

You can use the icon in the **Terms** widget on the **Asset Details** page to add new terms for your assets. Click Save to save the changes.

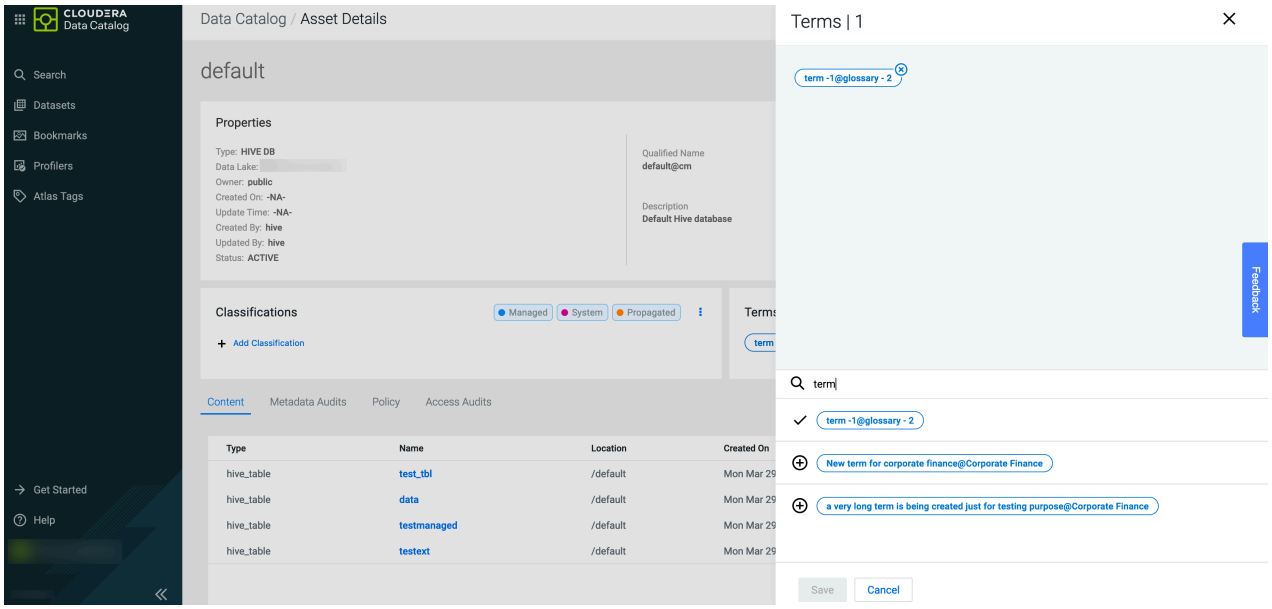

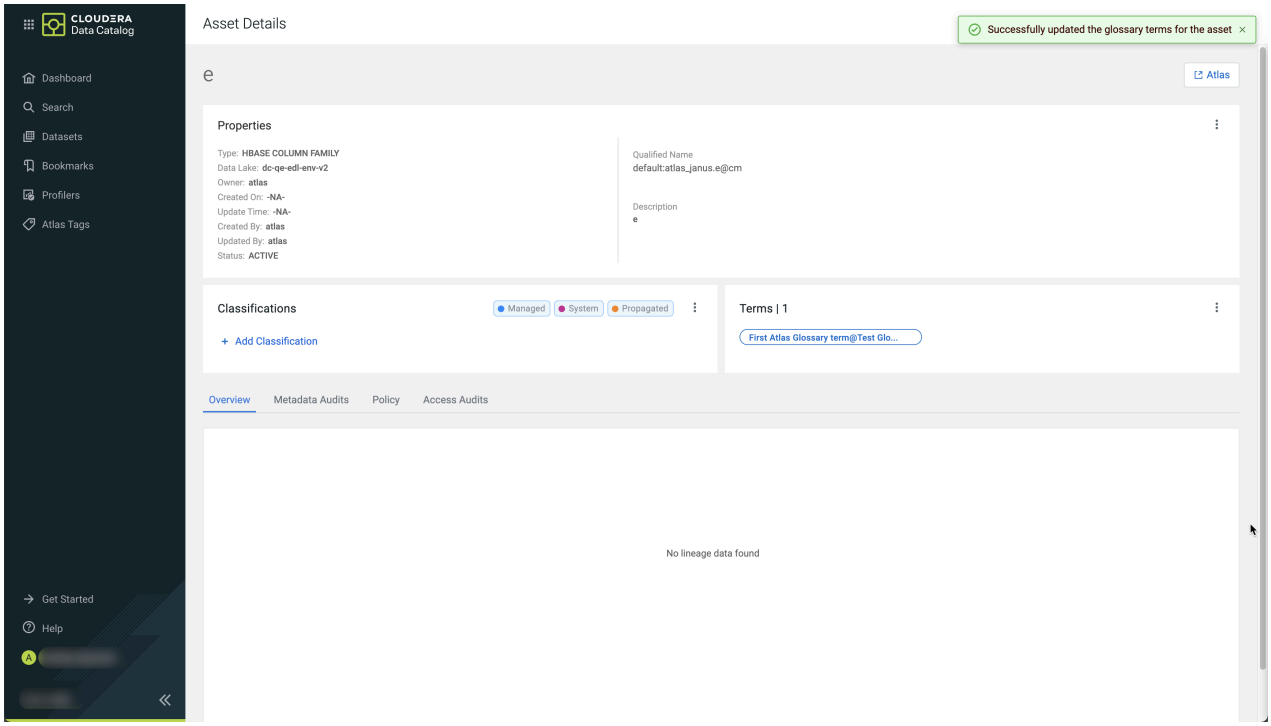

You can search for the same asset in the corresponding Atlas environment as shown in the example image.

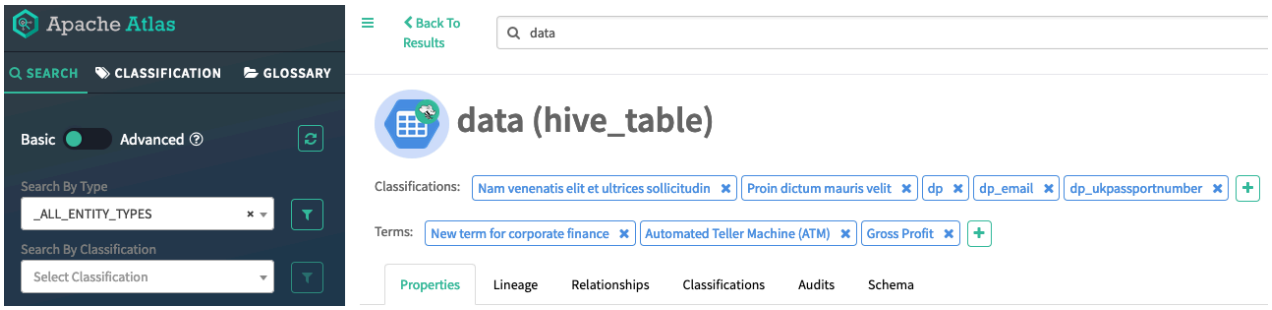

When you select a Hive table asset and navigate to the **Asset Details** page, under the **Schema** tab, you can view the list of terms associated with the asset.

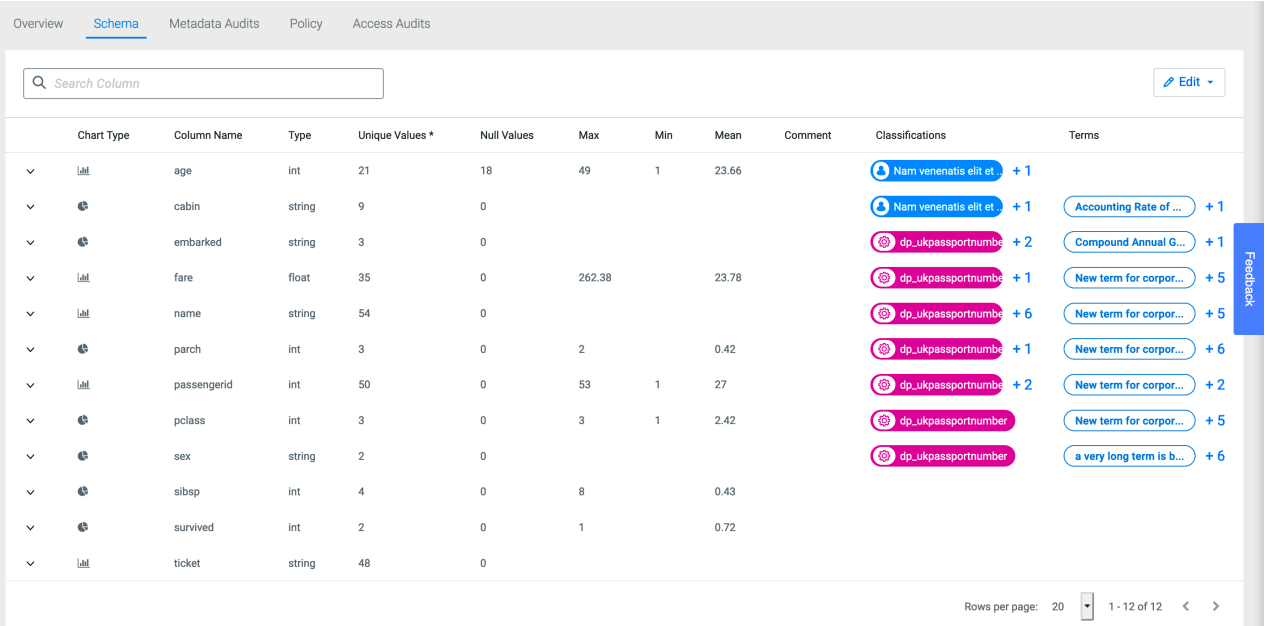

You can add or update the terms for the associated datasets by clicking the Edit button.

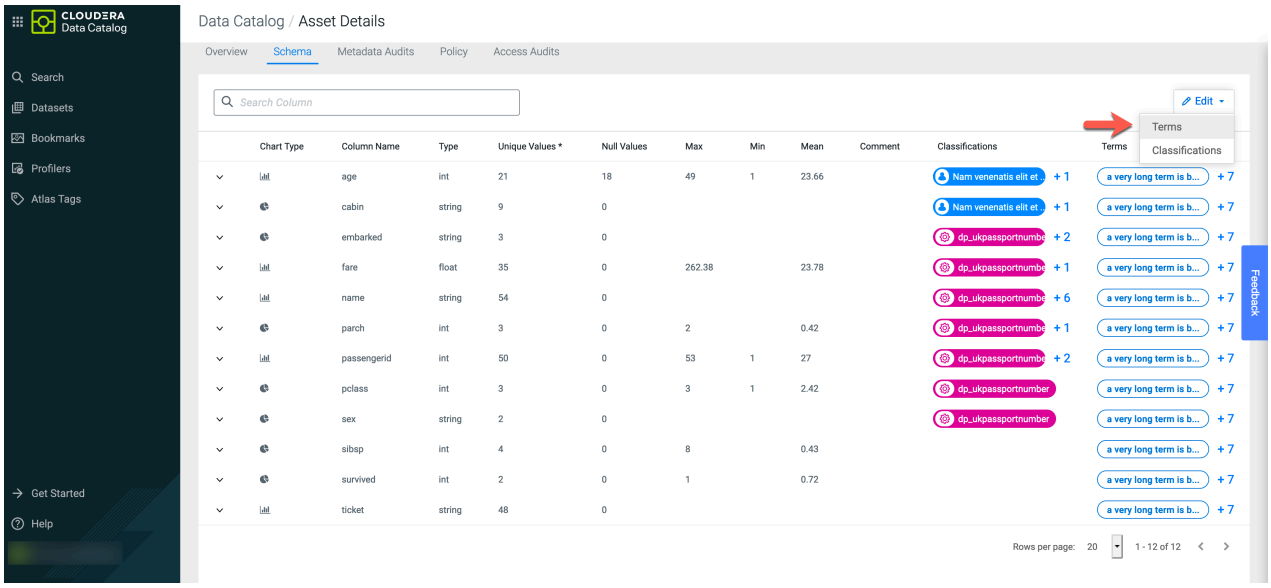

Searching for assets using glossary terms

You can search for the datasets using the Glossary Terms filter available on the Search page.

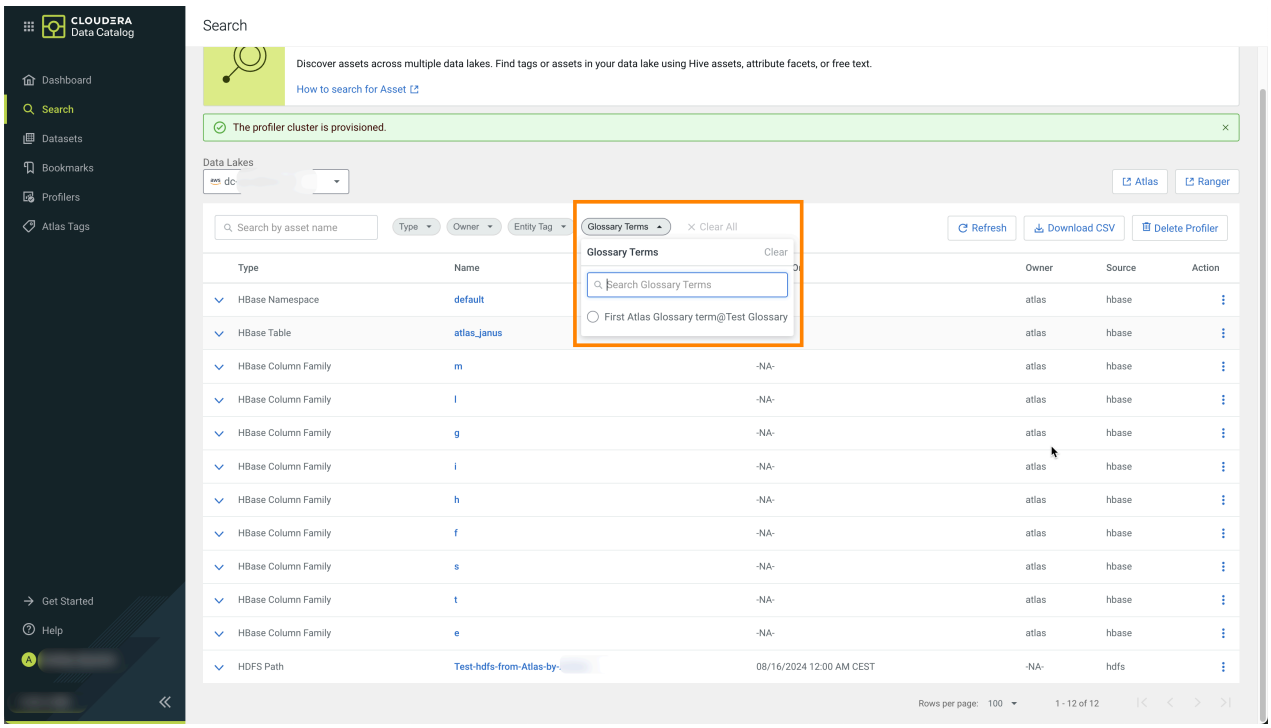

Additional search options for asset types

Using Cloudera Data Catalog, you can add or edit asset description values to search for data assets across both Cloudera Data Catalog and Apache Atlas services by using the asset content.

In the **Asset Details** page for each asset type that you select, you can add or edit **comment** or **description** fields. Including these values for the selected asset helps you to identify your chosen asset.

Using the same set of values (comment or description), you can also search for the asset types in Atlas.

Note: The comment and description options are supported only for Hive table and Hive Column assets. For other asset types, only the description option is supported.

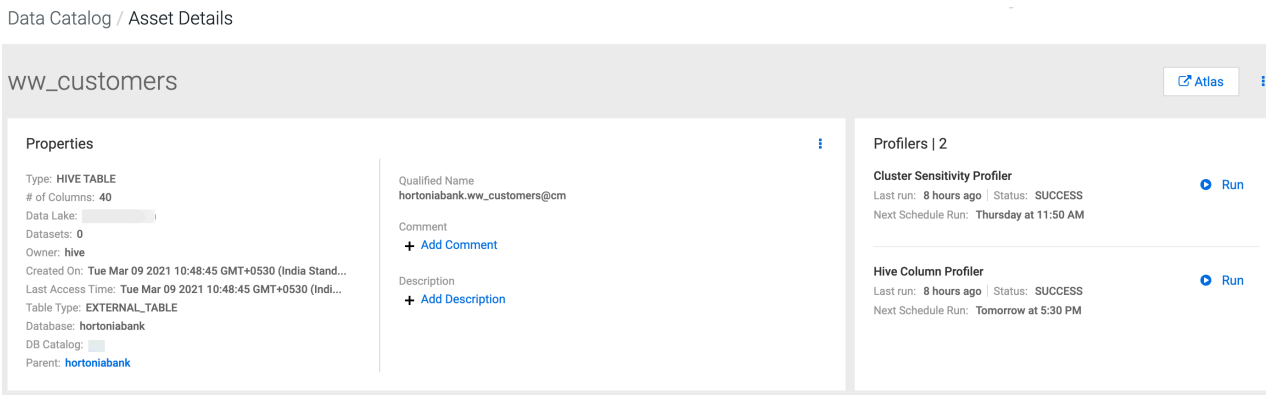

Click + Add Comment or + Add Description fields to include the respective values.

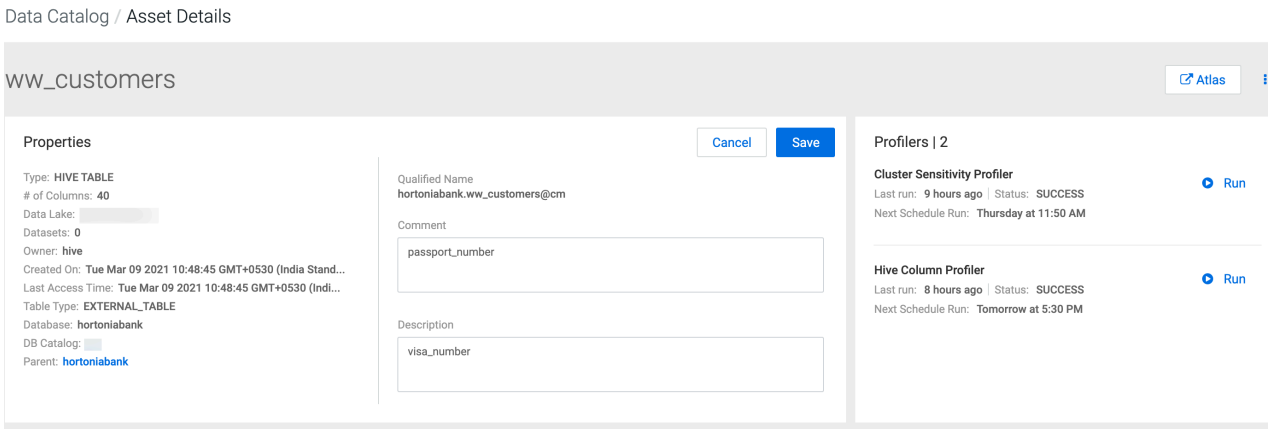

Click Save to save your changes.

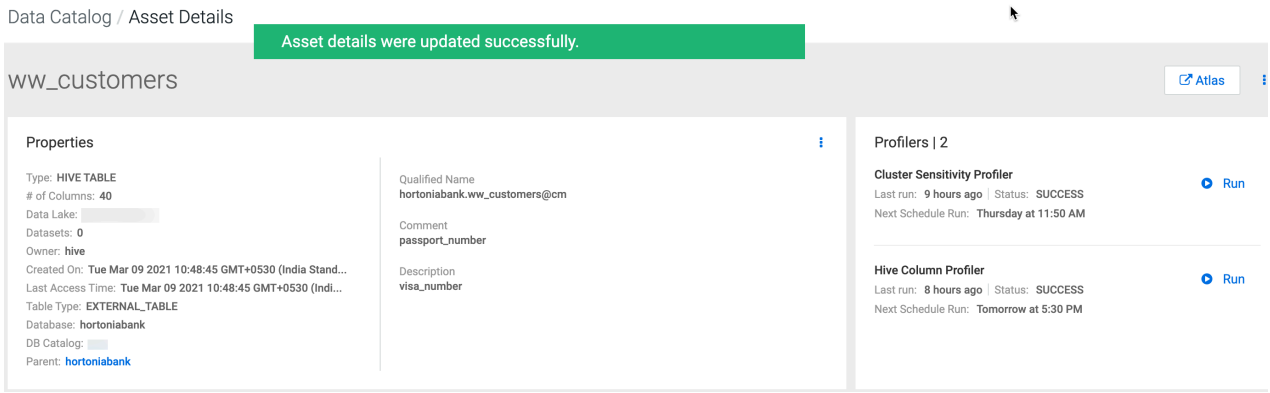

Note: You can also edit the already saved valued by clicking the icon.

Clicking on the Atlas button will navigate to the corresponding Atlas asset page as shown:

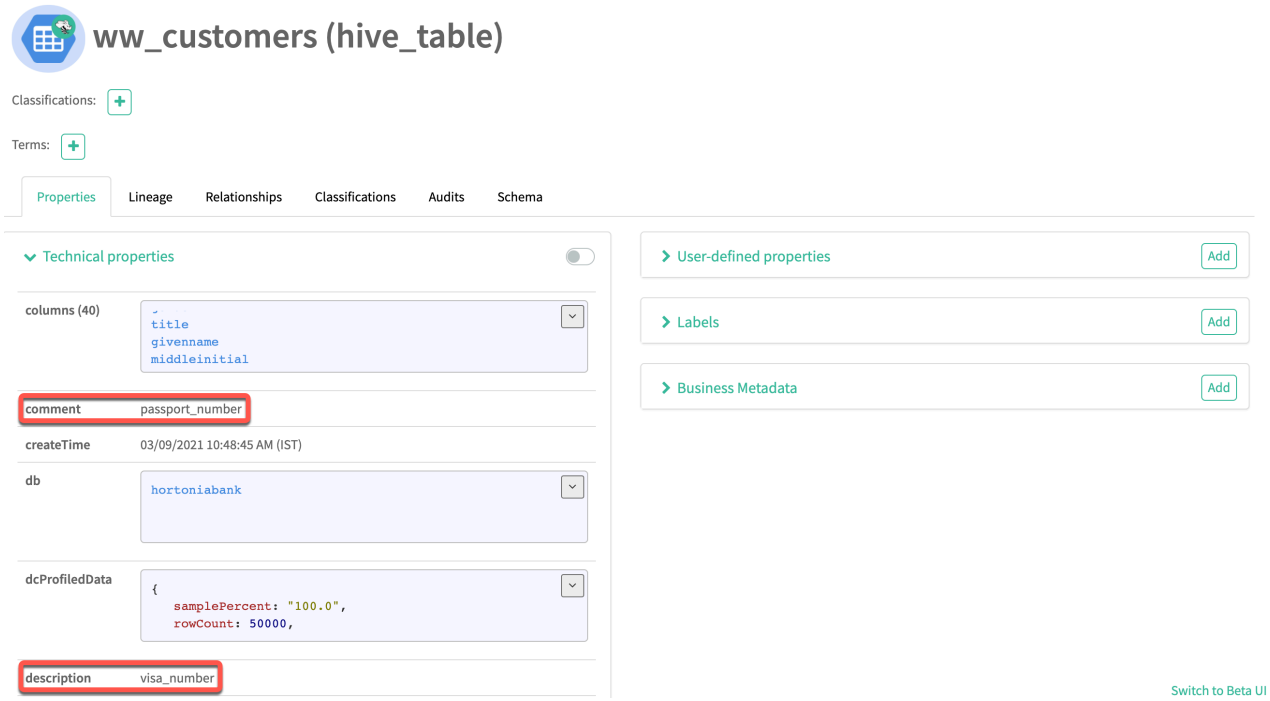

Searching for assets using additional search options

In Cloudera Data Catalog, you can select a data asset type and under the Asset Details page, to insert a comment and to provide a description for the selected asset.

The values of the **Comment** or **Description** fields can be searched in the **Search** menu. The result page displays the assets where you added your comments and descriptions without the use of filters.

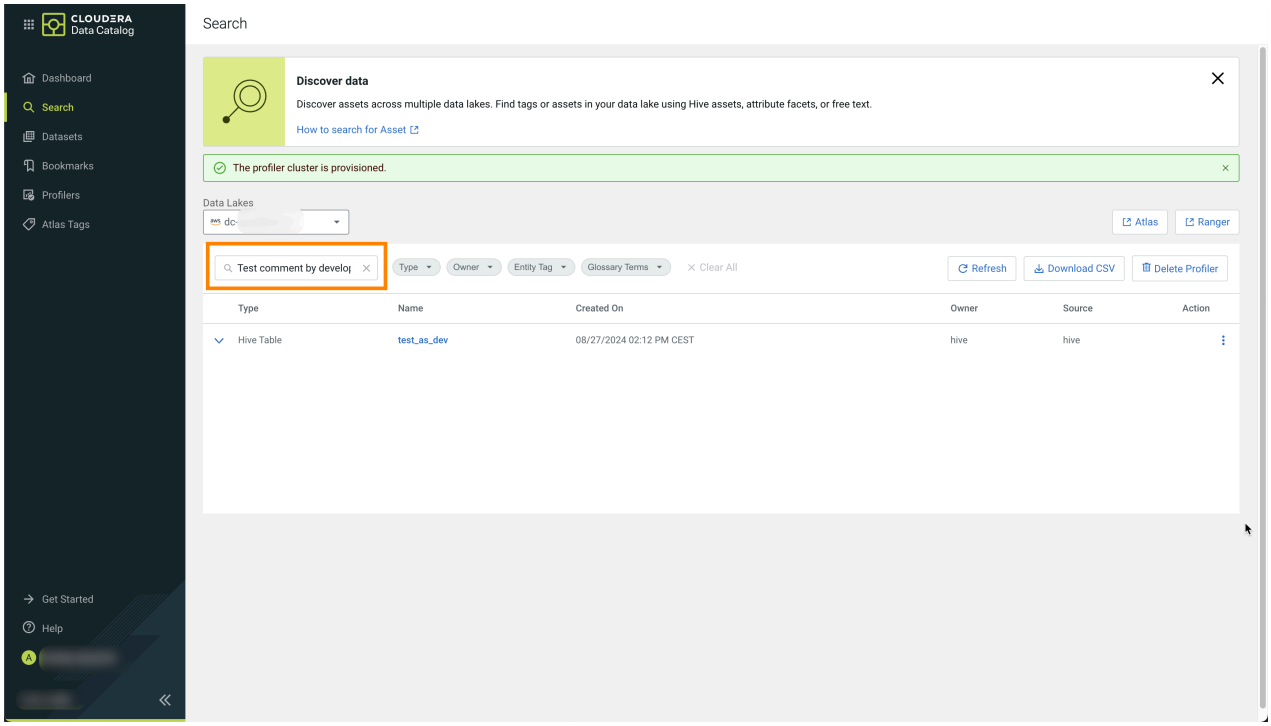

Clicking on the asset type displays the comment and description values.

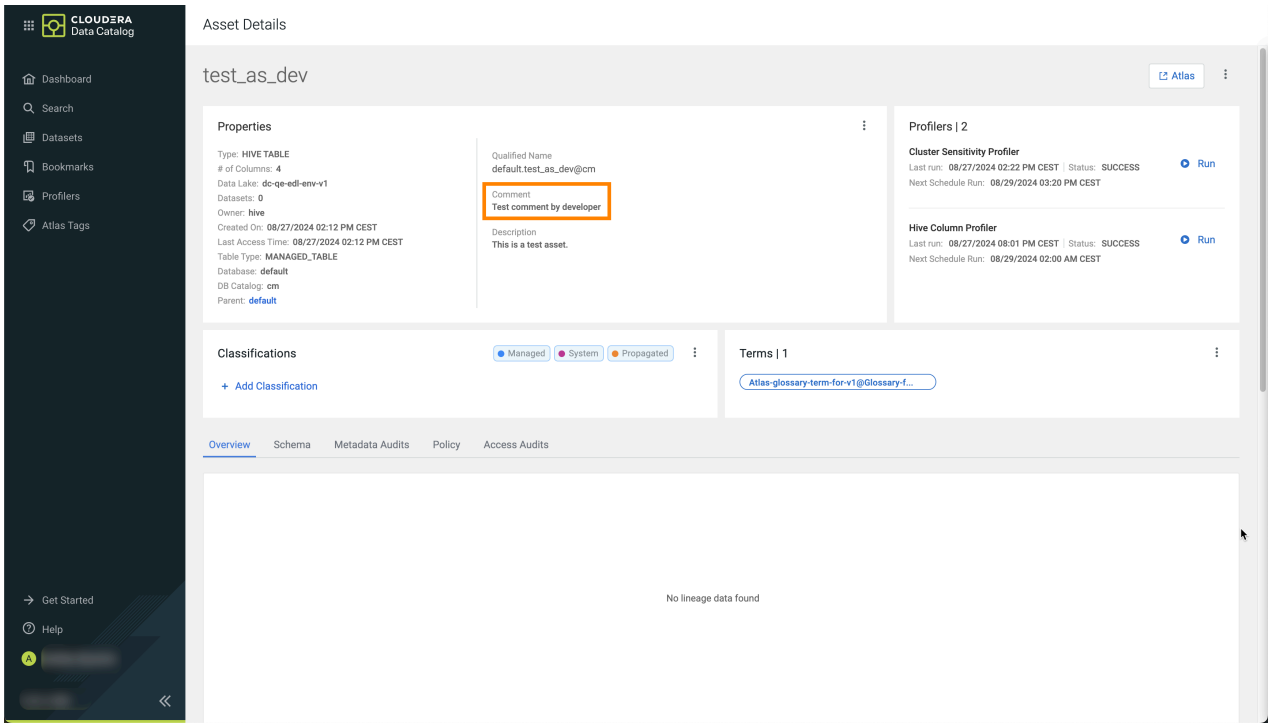

Accessing tables based on Ranger policies

When a table (in blue color link) is clicked, the Asset Details view page is displayed.

If a user is not authorized to click or view table details, it implies that the user permissions have not been set up in the Apache Ranger.

As seen in the following diagram, if users are not able to view the table details, a message appears next to the same table "Some information might not be available to unauthorised users".

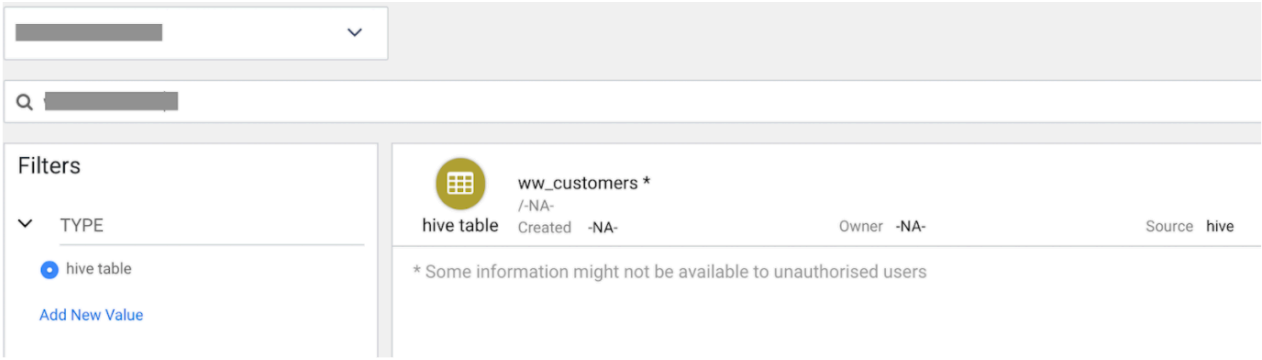

In the next example diagram, tables that have the permissions to view are displayed with a blue color link. The ones that do not have read permissions are visible in grey.

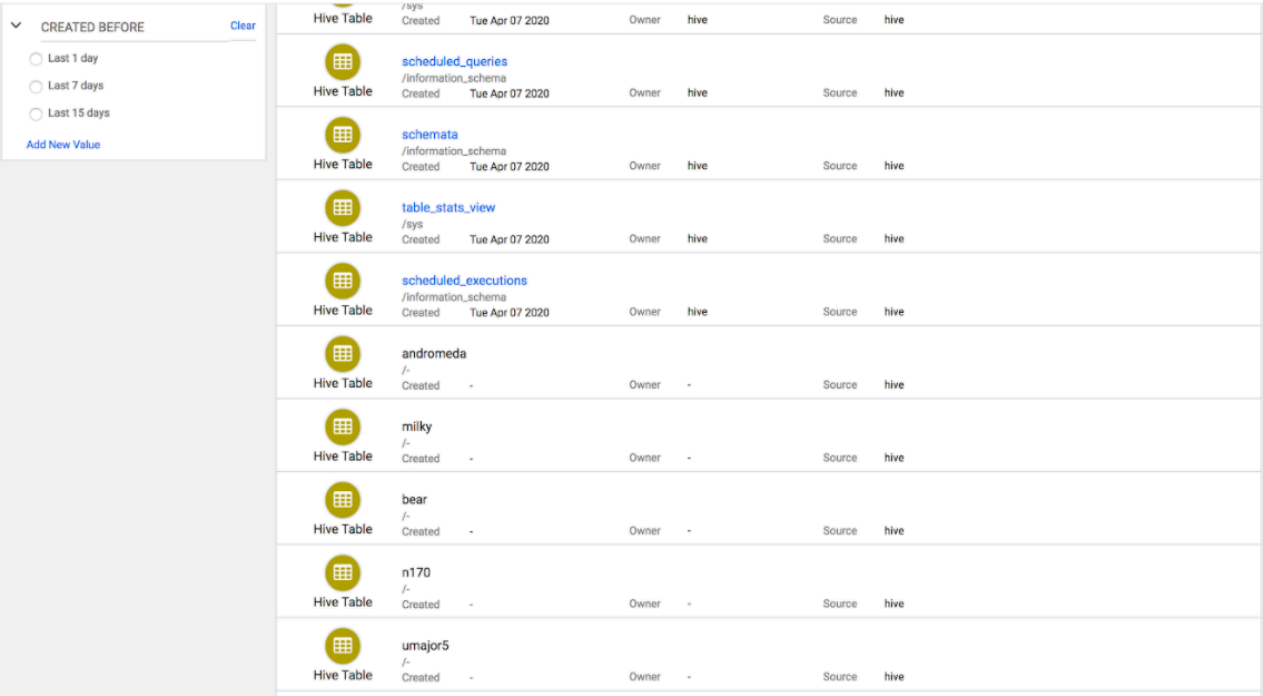

Related Information

Ranger Policies Overview

Configuring resource-based services

Creating classifications for selected assets

You can create classifications in multiple pages. These classifications can be associated with an asset. Then, you can use these classifications to filter your assets both in Cloudera Data Catalog and Apache Atlas.

Creating a classification from the Search page

1. Navigate to the **Search** page.
2. Click the ⋮ icon by an asset, then select Edit Classifications.

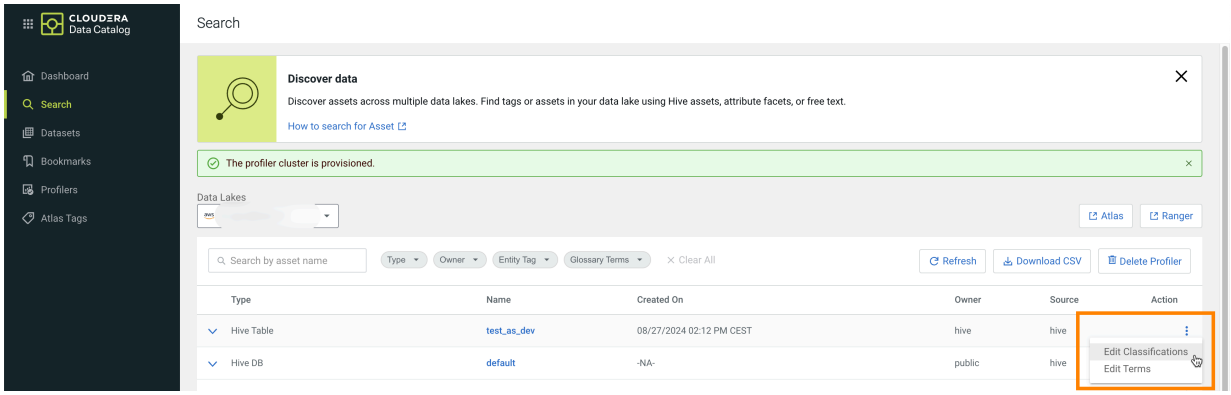

3. Search for a previously created classification or create a new one.
4. Click Save to finalize your changes.

Creating a classification from Asset Details

1. Navigate to the **Asset Details** page of an asset.
2. Click Add Classification or ⋮ icon by an asset, then select Edit.

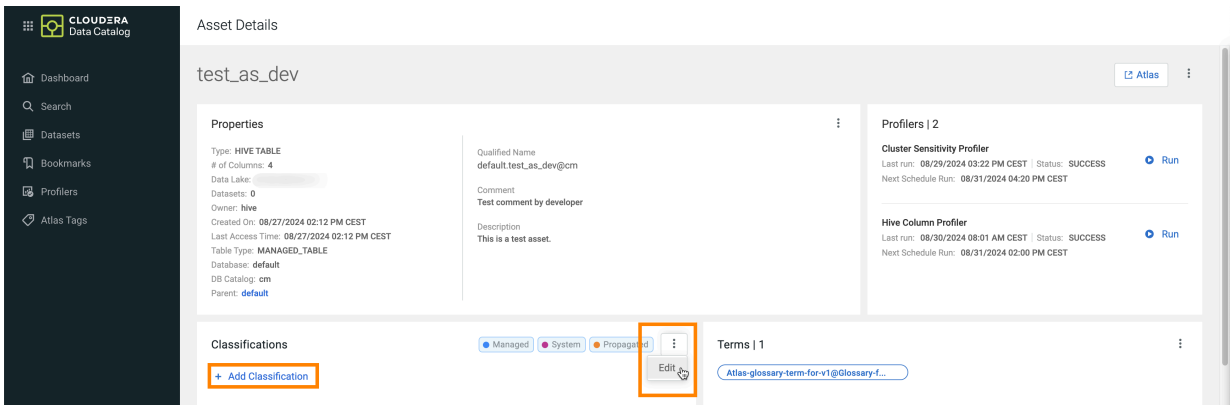

3. Search for a previously created classification or create a new one.
4. Click Save to finalize your changes.

Creating a classification in Atlas Tags

1. Navigate to **Atlas Tags**.
2. Click Add Tag.

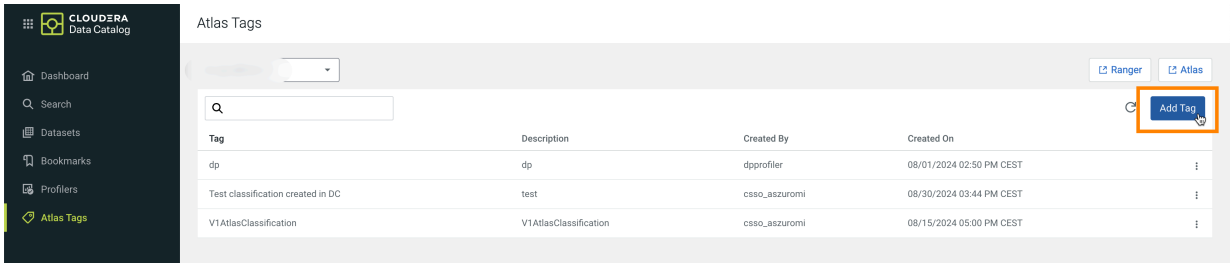

3. Fill in the details and Save your changes.

Note: Your classification still needs to be added to an asset in the **Search** or **Asset Details** menu.

Note: Classifications are synchronized between Apache Atlas and Cloudera Data Catalog.

Related Information

Working with Atlas classifications and labels

Creating classifications

Additional entity type selection for searching assets

Using the Cloudera Data Catalog service, you can search for assets by using entity types.

Cloudera Data Catalog users can search and discover different asset types.

Supported entity types include the following:

- Azure BLOB
- Azure Container
- Azure Directory
- AWS S3 Object
- AWS S3 V2 Object
- AWS S3 Bucket
- AWS S3 V2 Bucket
- AWS S3 Pseudo Dir
- AWS S3 V2 Directory
- HBase Table
- HBase Column Family
- HBase Namespace
- HDFS Path
- Hive DB
- Hive Table
- Hive Column
- ML Project
- ML Model Build
- ML Model Deployment
- NiFi Flow
- NiFi Data
- Iceberg Column[1]
- Iceberg Table[1]
- Impala Process
- Impala Column Lineage
- Impala Process Execution
- Kafka Topic
- RDBMS DB
- RDBMS Column
- RDBMS Foreign Key

[1] Iceberg assets are discoverable in VM-based environments but they can be profiled only in Compute Cluster enabled environments.

- RDBMS Index
- RDBMS Instance
- RDBMS Table
- Spark Process
- Spark Application
- Spark Column
- Spark Column Lineage
- Spark DB
- Spark ML Directory
- Spark ML Model
- Spark ML Pipeline
- Spark Process Execution
- Spark Table

Selecting a type triggers a search query for that type. The **Owner** of the asset is derived from the response received from the type based queries.

The following examples depict the entity type selection search results:

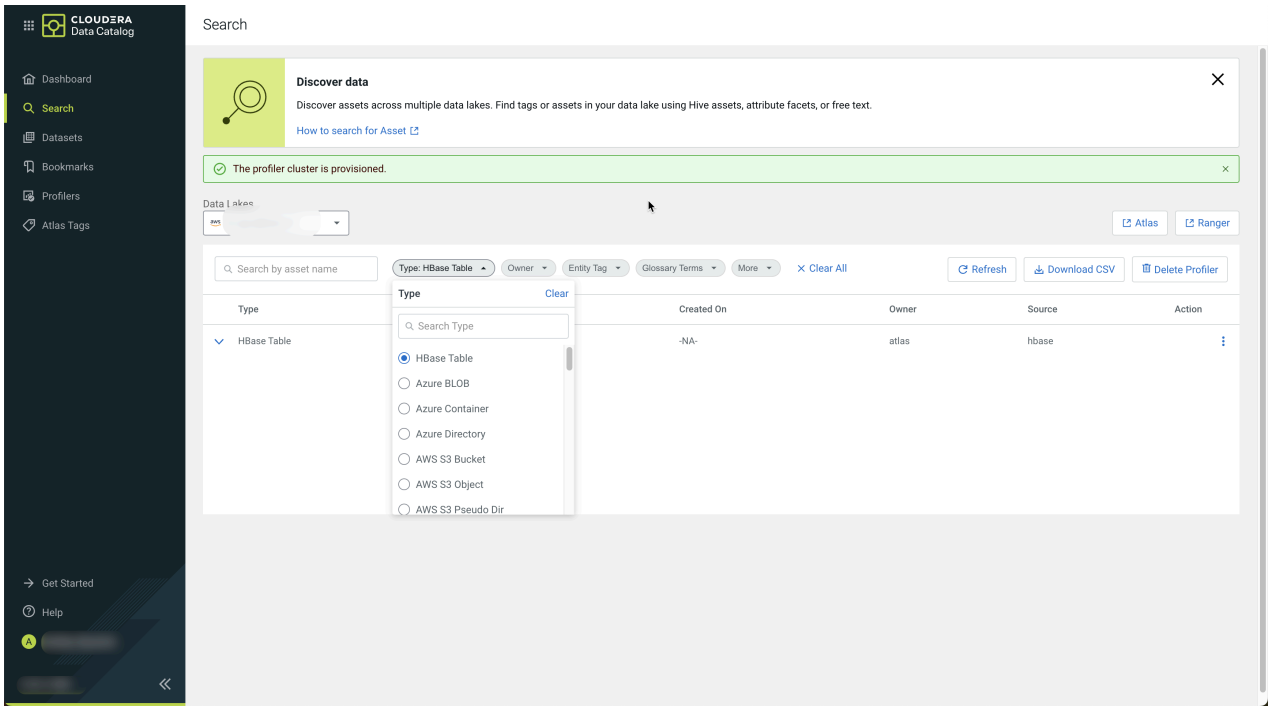

Viewing Data Asset details

Use the Asset Details menu to learn about the specifics of your asset including its relation to other asset, metadata and access updates related to it or Apache Ranger policies affecting it.

To access the **Asset Details** menu, click an asset in the **Search** menu. This brings you to the **Overview** tab.

- **Properties**:
  - **Type**
  - **Number of columns**
  - **Data Lake**
  - **Owner**
  - **Created On** date
  - **Modified Time**
  - **Namespace GUID** (HBase entity specific)
  - **Last Access Time**
  - **Table Type**
  - **DB Catalog**
  - **Parent**
  - **Qualified Name**
  - **Comment**

    Note: This field is available for Hive tables and columns and Iceberg tables and columns.

  - **Description**
- **Profilers**: The list of available on-demand profilers and details of the last run.
- **Classification** and **Terms**: The list of assigned metadata.
- **Overview**: Shows the chain of custody for the data from relevant metadata repositories such as Apache Atlas. Lineage overview shows both upstream paths (lineage) into and downstream paths (impact) out of a given asset.
- **Schema**: Displays the schema of the data asset for structured data (such as Hive tables) from the relevant metadata repositories (such as Atlas).
- **Metadata Audits**: Shows the list of metadata changes.
- **Policy**: Shows security (authorization) policies defined on assets such as those present in Apache Ranger. It includes both resource (physical asset based) as well as classification based policies. This helps you understand how data access is secured and protected: what users can see what data (or metadata) under what conditions (security policies, data protection, and anonymization).
- **Access Audits**: Shows the most recent access audits from Apache Ranger.

Supported Data Asset fields for entities

The Asset Details menu displays all the Apache Atlas metadata associated with a particular data asset.

Supported fields

The following matrix captures the supported fields for different asset types:

Asset Type	Lineage	Tagging	Access Metrics	Schema	Policy	Metadata and Access Audits	Atlas Punch out
aws_S3_bucket	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
aws_S3_Object	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
aws_S3_pseudo_dir	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
aws_s3_v2_object	Yes	Yes	Not Supported	Not Supported	Not Supported	Yes	Yes
aws_s3_v2_directory	Yes	Yes	Not Supported	Not Supported	Not Supported	Yes	Yes

Asset Type	Lineage	Tagging	Access Metrics	Schema	Policy	Metadata and Access Audits	Atlas Punch out
aws_s3_v2_bucket	Yes	Yes	Not Supported	Not Supported	Not Supported	Yes	Yes
adls_gen2_directory	Yes	Yes	Not Supported	Not Supported	Not Supported	Yes	Yes
adls_gen2_blob	Yes	Yes	Not Supported	Not Supported	Not Supported	Yes	Yes
adls_gen2_container	Yes	Yes	Not Supported	Not Supported	Not Supported	Yes	Yes
Hive DB	Not Supported	Yes	Not Supported	Not Supported	Yes	Yes	Yes
Hive Table	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hive Column	Yes	Yes	Not Supported	Not Supported	Yes	Yes	Yes
Hbase Namespace	Yes	Yes	Not Supported	Not Supported	Yes	Yes	Yes
Hbase Table	Yes	Yes	Not Supported	Yes	Yes	Yes	Yes
Hbase Column Family	Yes	Yes	Not Supported	Not Supported	Yes	Yes	Yes
Iceberg Table[2]	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Iceberg Column	Yes	Yes	Yes	Not supported	Yes	Yes	Yes
impala_process	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
impala_column_lineage	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
impala_process_execution	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
ML_Project	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
ML_Model_Build	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
ML_Model_Deploy	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
rdbms_db	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
rdbms_column	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
rdbms_foreign_key	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
rdbms_index	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
rdbms_instance	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
rdbms_table	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes

[2] Iceberg assets are discoverable in VM-based environments but they can be profiled only in Compute Cluster enabled environments.

Asset Type	Lineage	Tagging	Access Metrics	Schema	Policy	Metadata and Access Audits	Atlas Punch out
spark_process	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
spark_application	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
spark_column	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
spark_column_lineage	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
spark_db	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
spark_ml_directory	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
spark_ml_model	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
spark_ml_pipeline	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
spark_process_execution	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes
spark_table	Yes	Yes	Not Supported	Not Supported	Not Supported	Not Supported	Yes

Navigation in Asset Details

A generic Assets Details page is available for container data types like buckets and databases.

The **Contents** tab (similar to the **Schema** tab) lists all the contents of the selected entity. Clicking on any element available in the **Content** tab navigates you to the **Asset Details** page.

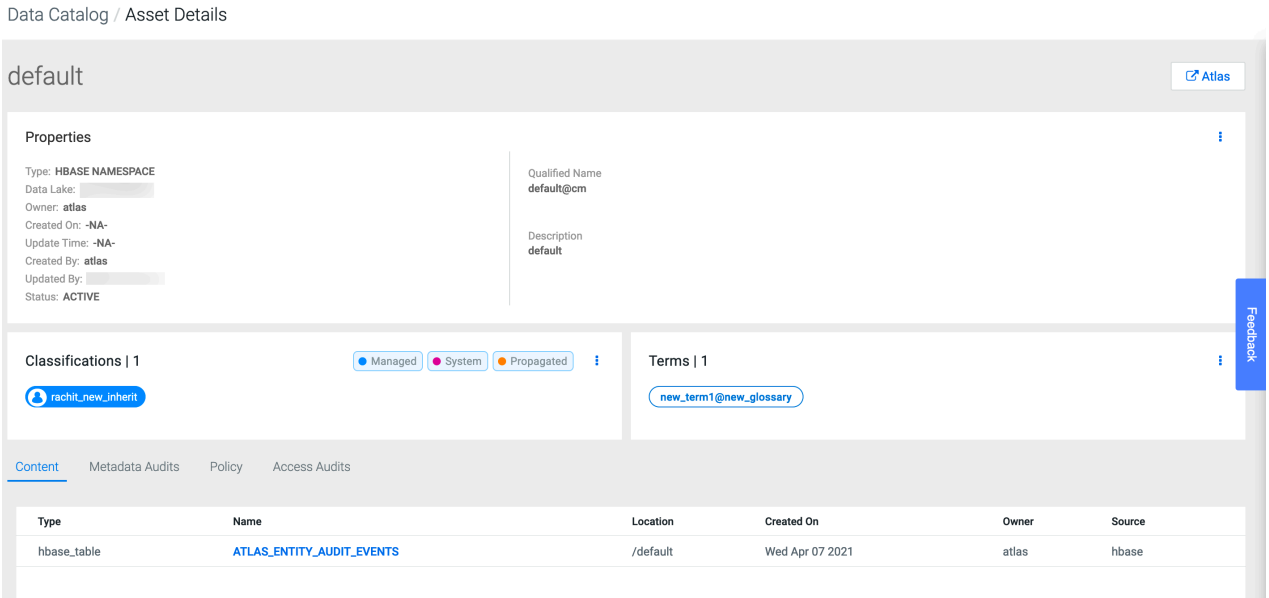

For example, for a database entity having a list of tables, clicking on any listed table navigates to the **Asset Details** page of the same table. This page helps you understand the parent-child relationship between related assets. The **Contents** tab displays entities that are contained within assets of container entity types. The entities in the table of **Contents** tab are clickable, which will allow you to navigate to the **Asset Details** page of these contained assets.

In these contained assets, the Parent row in the **Properties** panel allows you to go one level up in the hierarchy to the container element. In certain elements, such as Hive tables or Iceberg tables, the **Schema** tab contains further elements whose name can be clicked to open individual contained elements.

The following table lists the entity types, their parent, and contents.

Type	Parent	Content
Hive DB	-	Hive Table, Iceberg Table
Hive Table, Iceberg Table[3]	Hive DB	Hive column, Iceberg column
HBase Namespace	-	HBase Table
ML Project	-	ML Model Build
ML Model Build	ML Project	ML Model Deployment
AWS S3 Bucket	-	AWS S3 Pseudo Dir
AWS S3 Pseudo Dir	AWS S3 Bucket	AWS S3 Object
RDBMS Instance	-	RDBMS DB
RDBMS DB	RDBMS Instance	RDBMS Table

View Data Asset Access Audit Logs

The Asset Details Access Audit page displays all the Apache Ranger audit events associated with a particular data asset. This helps you to view who has accessed what data from a forensic audit or compliance perspective, and to visualize access patterns and identify anomalies.

Procedure

1. From the **Search** menu, select a data asset.
   The **Asset Details** opens.
2. Click the **Access Audit** tab.
   The Audit table shows the most recent raw audit event data by type of access and access outcome (authorized/unauthorized).

[3] Iceberg assets are discoverable in VM-based environments but they can be profiled only in Compute Cluster enabled environments.

3. You can filter the audit results by **Access Type** or **Result**.

   The available options are the following for Hive Table:

   Note: Different asset types have different filter options.

   - **Access Type**
     - SELECT
     - UPDATE
     - CREATE
     - DROP
     - ALTER
     - INDEX
     - READ
     - WRITE
   - **Result**
     - ALL
     - ALLOWED
     - DENIED

Navigation support for hive entities within Lineage

The Lineage in the Overview tab shows the chain of custody for the data from relevant metadata repositories such as Apache Atlas. Lineage overview shows both upstream paths (lineage) into and downstream paths (impact) out of a given asset.

When you click a Hive entity within a lineage, the **Asset Details** page of the selected Hive entity is displayed. In the **Overview** tab, you can change the detail level of the lineage with the Depth drop-down and Show Process:

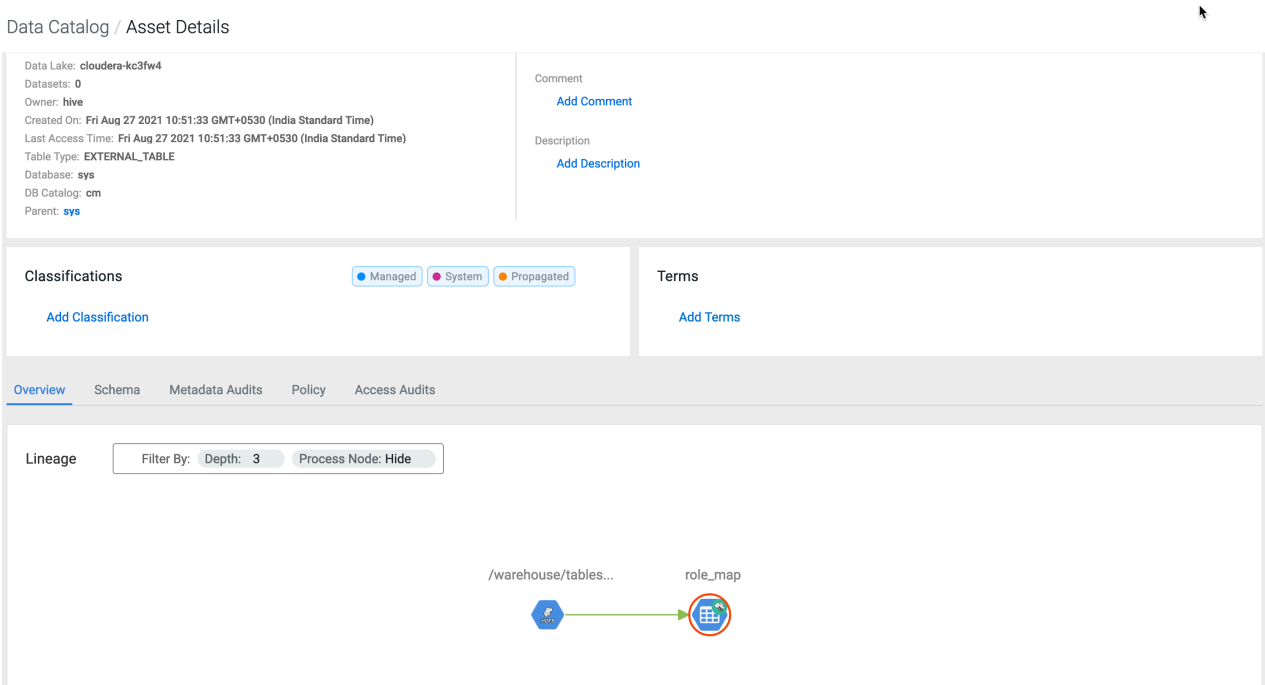

Alternatively, if you do not want to navigate away from the current page and want to view the information with respect to any entity, hover on the entity and click the information icon to view the details.

The screenshot depicts the slider information for the clicked entity:

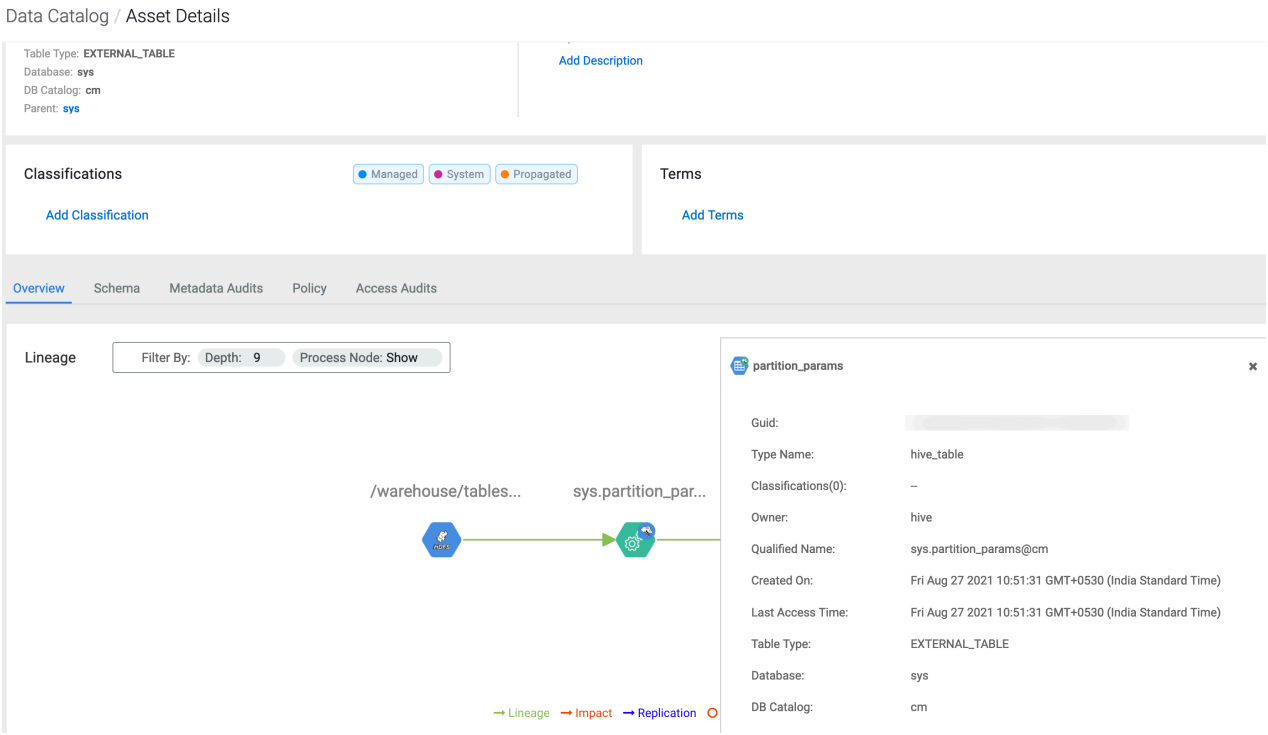

Adding assets to one or more datasets

On the Asset Details screen, users are provided with an option to add the asset to a dataset for easier management and searching.

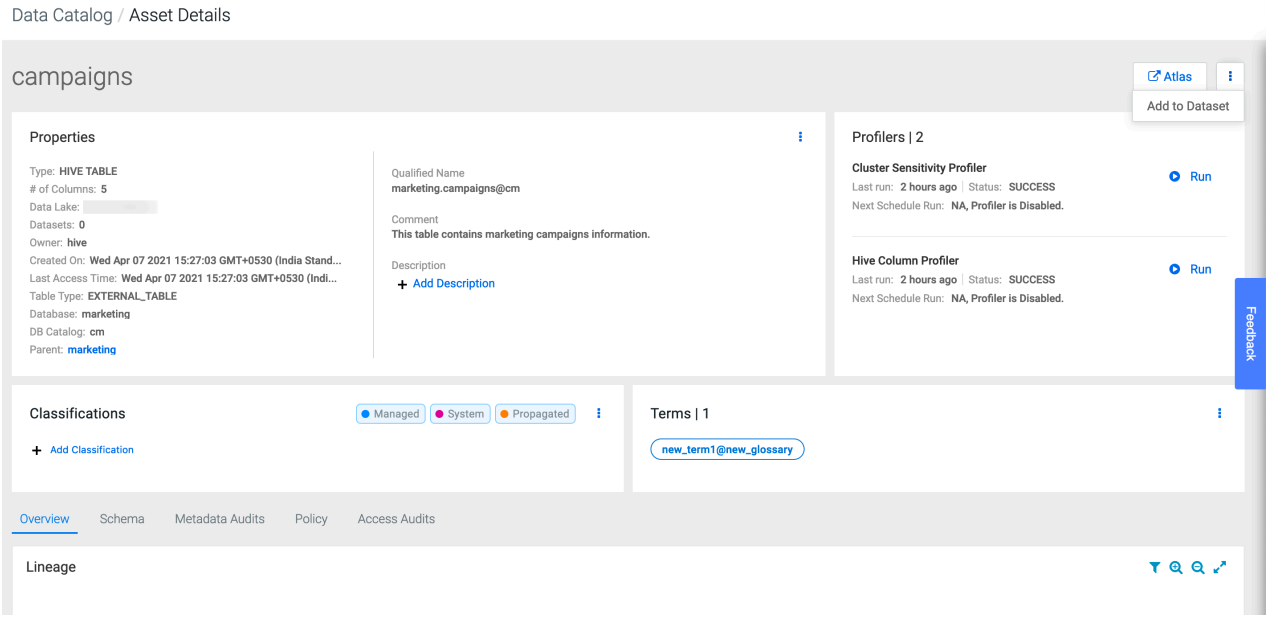

The **Add to Dataset** pane provides an option to add the asset into one or more existing datasets or even create a new one.

Datasets that already contain the asset are disabled and marked as checked. Datasets which are currently in edit state are disabled and marked with the characters a *.

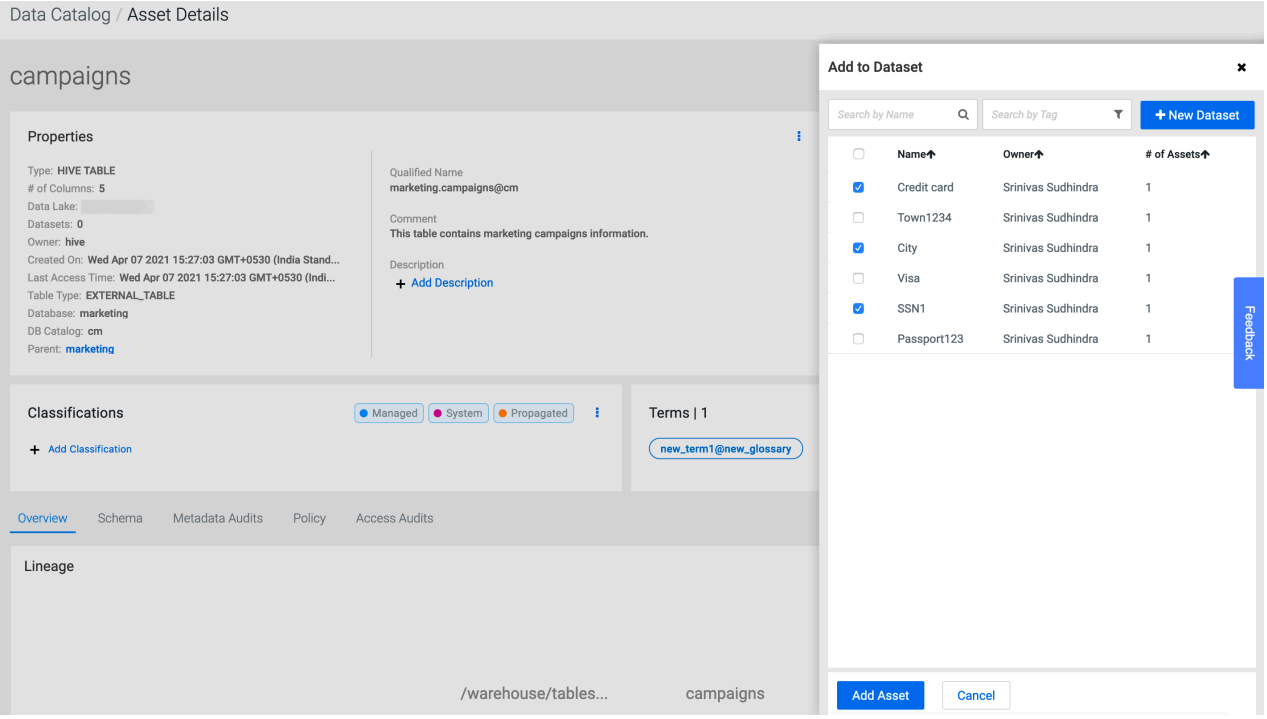

Users can search for an existing dataset by name or by tags applied on the dataset. Users can select one or more datasets from the list and then click on the Add Asset button which adds the asset to these dataset(s).

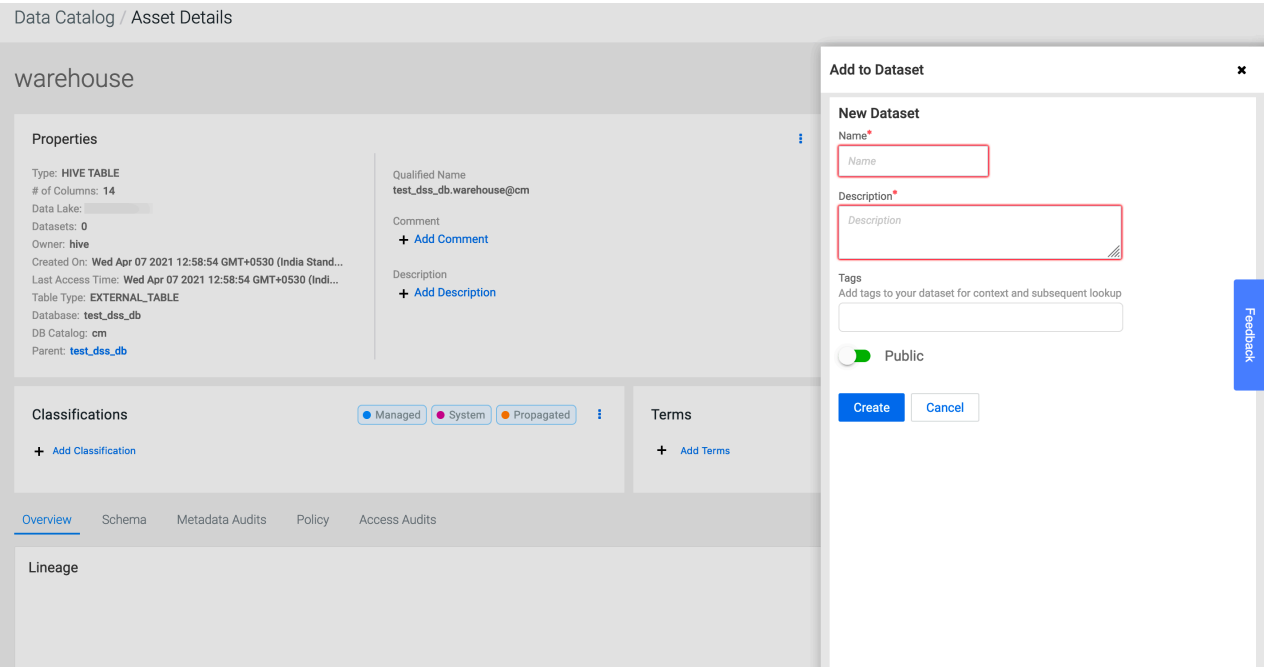

There are instances, where there are no datasets present or the user just wants to create a new dataset to add the asset. In that case, the user can click on the New Dataset button which opens up a new dataset form. Once the user fills in the form and clicks on the Create button, a new dataset with the given properties is created and the asset is added to it automatically. This is reflected in the datasets list where the newly added dataset is highlighted.

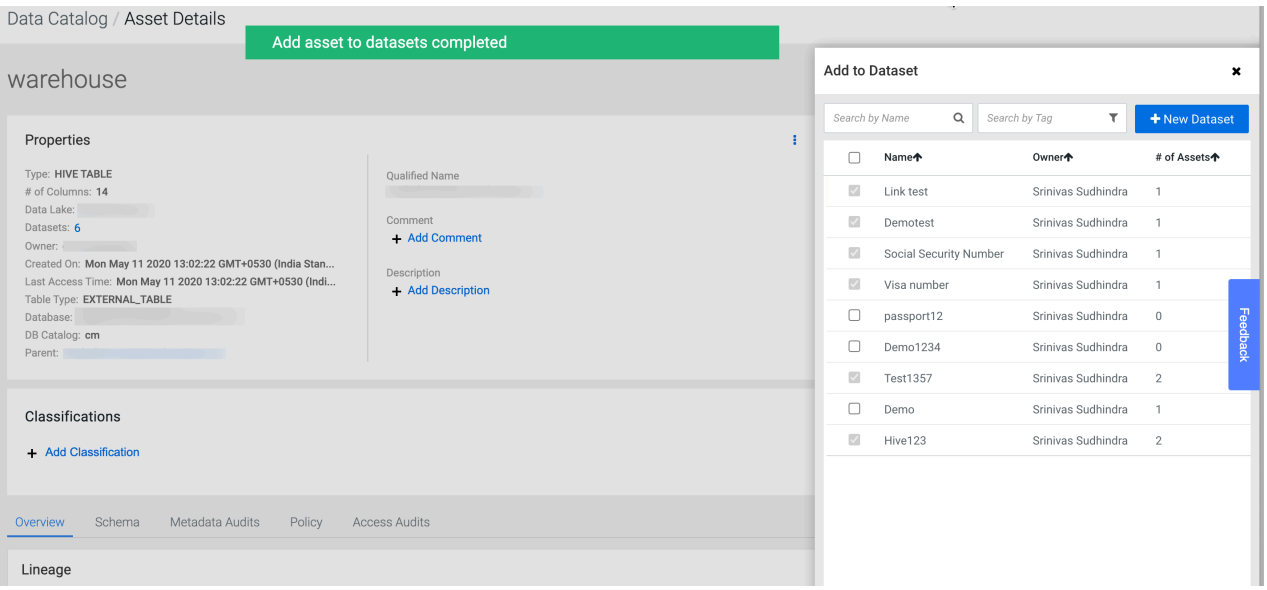

Viewing Atlas entity audits

Atlas audits help Data Stewards to identify and track the entity changes or modifications that are performed over a period of time.

Information about the Apache Atlas entity audit events are displayed for each entity in the **Asset Details**. Using this information, Data Stewards can distinguish between entity audits and data audits that originate from Ranger.

On the **Asset Details** page, the **Metadata Audits** tab displays information related to the selected entity type and about the events that occurred based on the user activities.

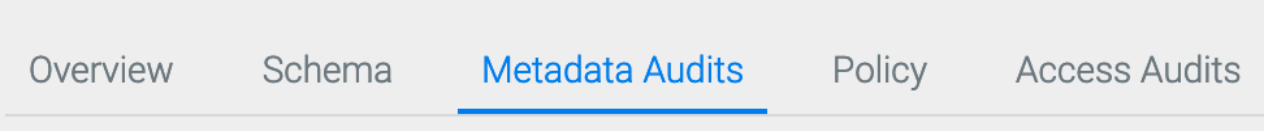

Clicking on **Metadata Audits**, tab, you can view manage information about:

- The user who made the changes to the specific entity
  - Users also include the profilers
- The time when the entity was changed
- The kind of change that was made to the entity
  - For example, adding comments, descriptions and the entity itself being created
- Any other relevant changes pertaining to the audit entries

The changes that can be identified for:

- Created entities and related updates
- Tagged entities
- Labeled entities
- Export and Import operations

For example, the following image displays the Atlas audit entity creation event that is recorded by each Atlas entity that is displayed in the **Asset Details** page.

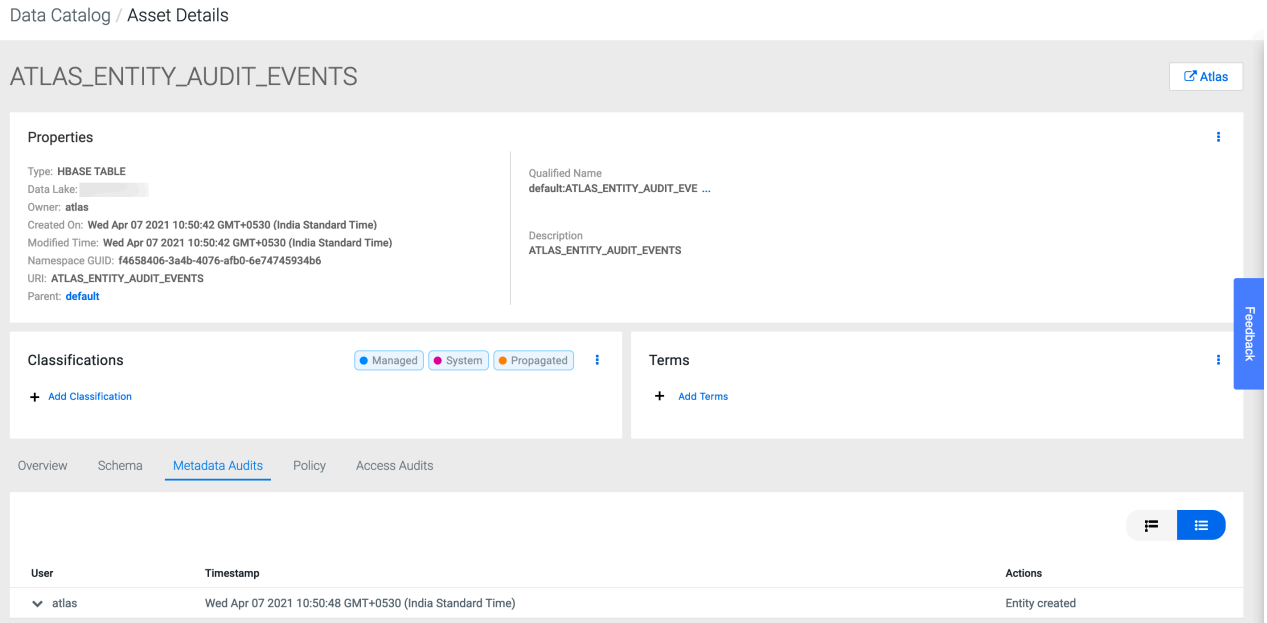

Clicking on any line item displays the change in JSON format, which is directly derived from Atlas. For example, adding an Atlas glossary term:

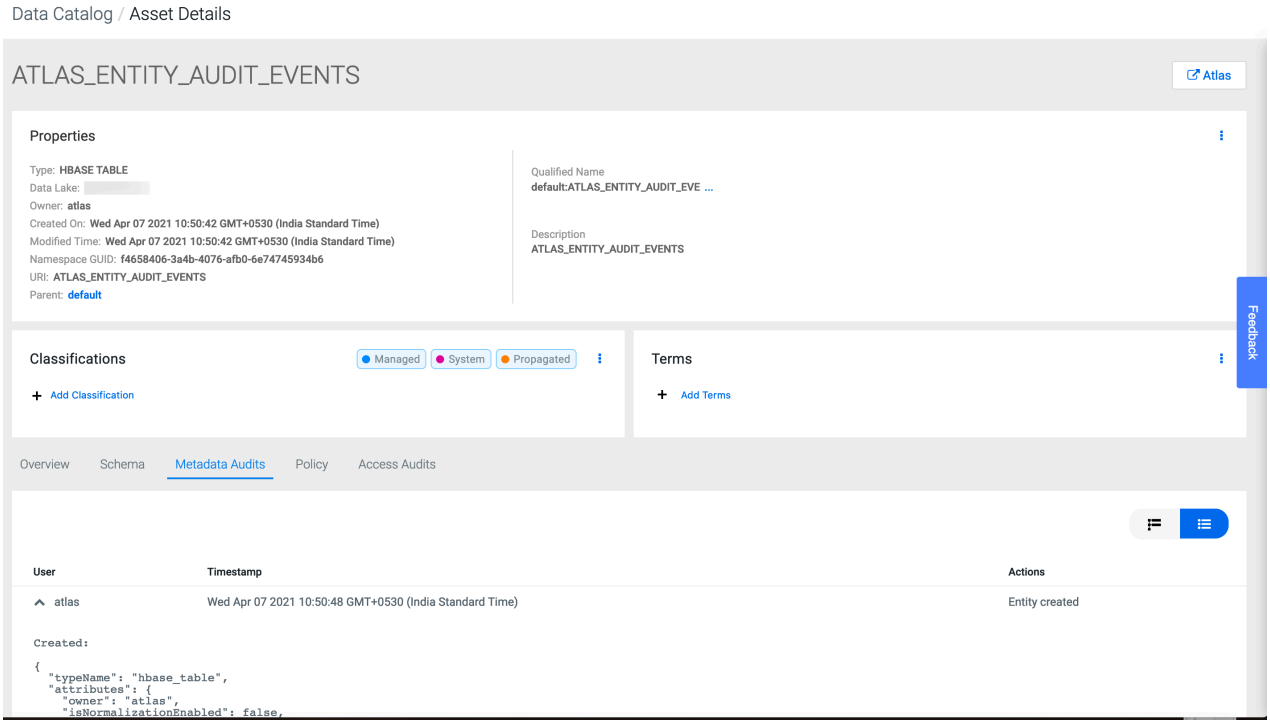

Use the toggle icon (on the top-right corner) for viewing Atlas audits in different formats. By default, you can view the audits in tabular format. When you toggle the view icon, you can view the **Timeline** format.

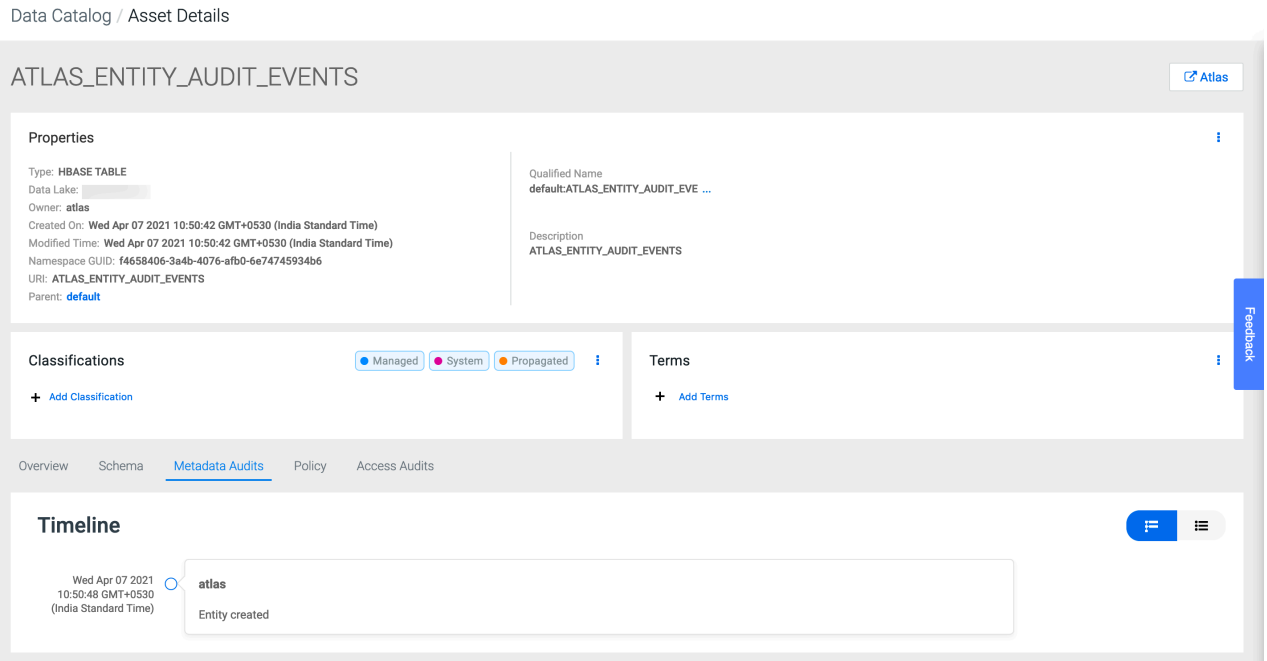

Clicking on a user in the **Timeline** format displays the JSON data.

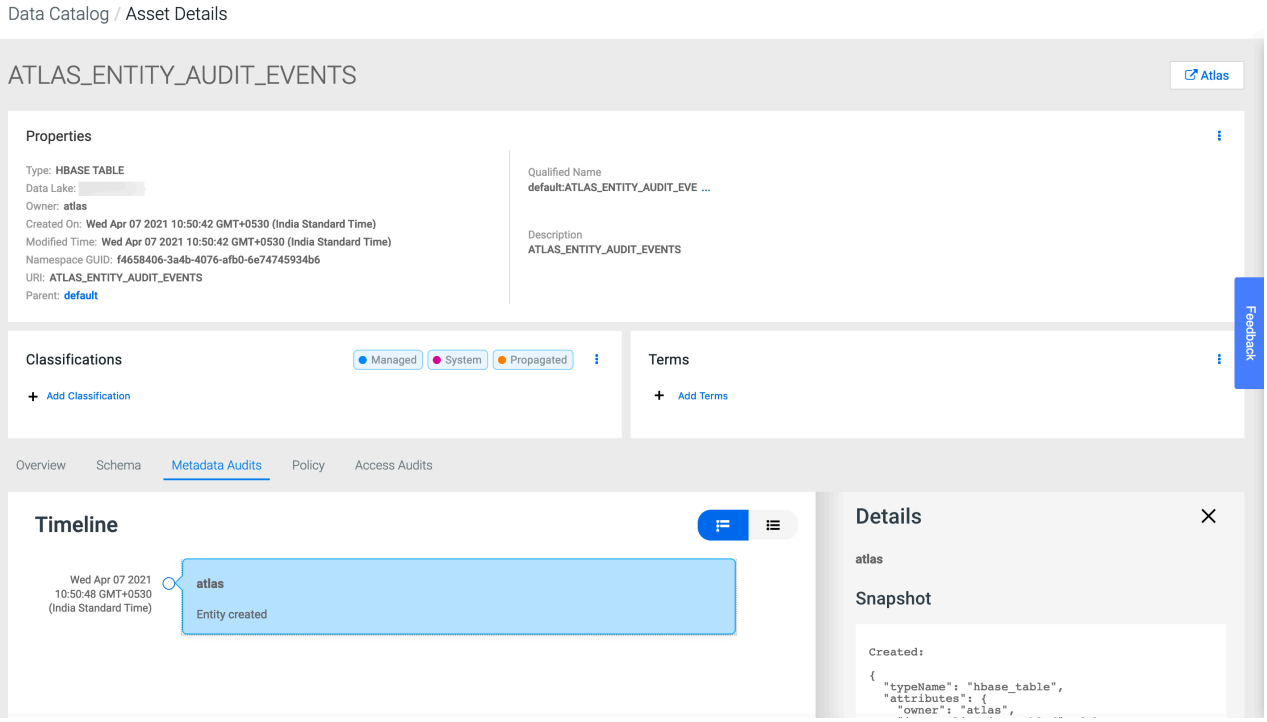

Viewing Ranger access audits

Apache Ranger audits help Data Stewards to identify and track the policies affecting the Apache Atlas entities over a period of time. More specifically, the accesses and the relevant policies can be also identified.

Information about the Apache Ranger policy audit events are displayed for each entity in the **Asset Details**. Using this information, Data Stewards can distinguish between policies originating from Ranger to have an overview of all users accessing (or being prevented from accessing) an entity. They can also use this to troubleshoot access issues.

On the **Asset Details** page, the **Access Audits** tab displays information related to the selected entity type and about the events that occurred based on the user activities trying to access the entity.

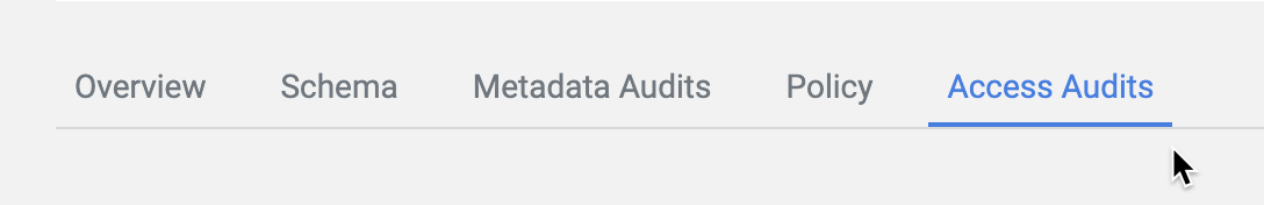

Clicking on **Access Audits** tab, you can view manage information about:

- The relevant Ranger policy ID
- The time when the entity access was attempted
- The user who accessed the specific entity (or was being barred from accessing it)

  Note: Users also include the profilers.

- The resource type accessed
- The access type
  - For example, by Hive entities the following filter options are available
    - SELECT
    - UPDATE
    - CREATE
    - DROP
    - ALTER
    - INDEX
    - READ
    - WRITE
- The access attempt result
  - ALLOWED
  - DENIED
- The client IP

The accesses can be identified for:

- Created entities and related updates
- Tagged entities
- Labeled entities

For example, the following image displays the entity creation event recorded for a Hive table:

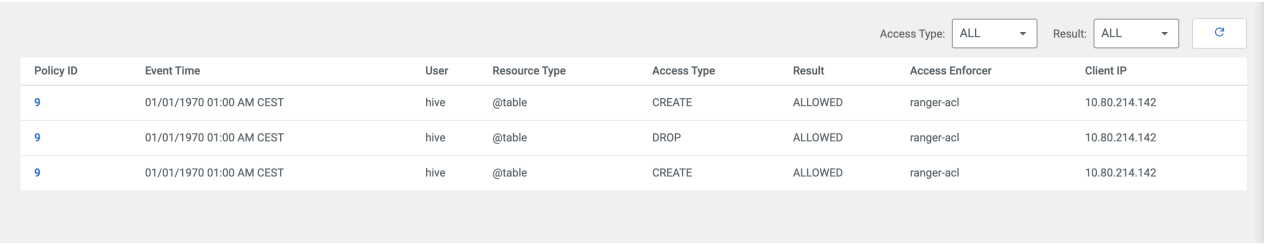

Policy ID	Event Time	User	Resource Type	Access Type	Result	Access Enforcer	Client IP
9	01/01/1970 01:00 AM CEST	hive	@table	CREATE	ALLOWED	ranger-acl	10.80.214.142
9	01/01/1970 01:00 AM CEST	hive	@table	DROP	ALLOWED	ranger-acl	10.80.214.142
9	01/01/1970 01:00 AM CEST	hive	@table	CREATE	ALLOWED	ranger-acl	10.80.214.142

Note: If you click an event row, you can see an overview of the users set for that policy including the set permissions.

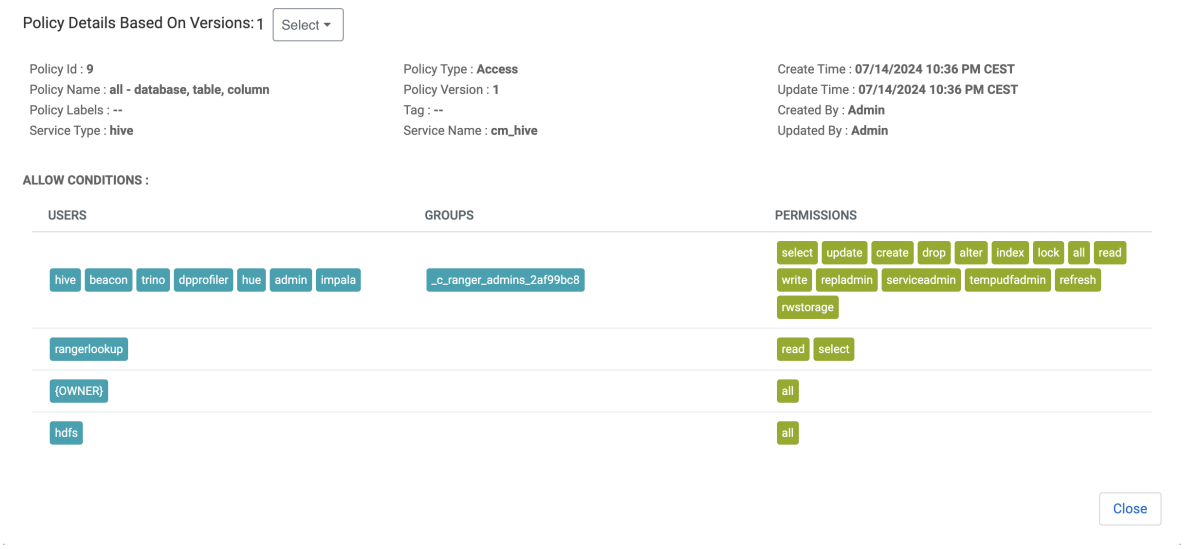

Viewing Ranger policies

Apache Ranger audits help Data Stewards to identify and track the policies affecting the Apache Atlas entities.

In Asset Details Policy , you can have an overview of the Ranger policies currently affecting the selected entity.

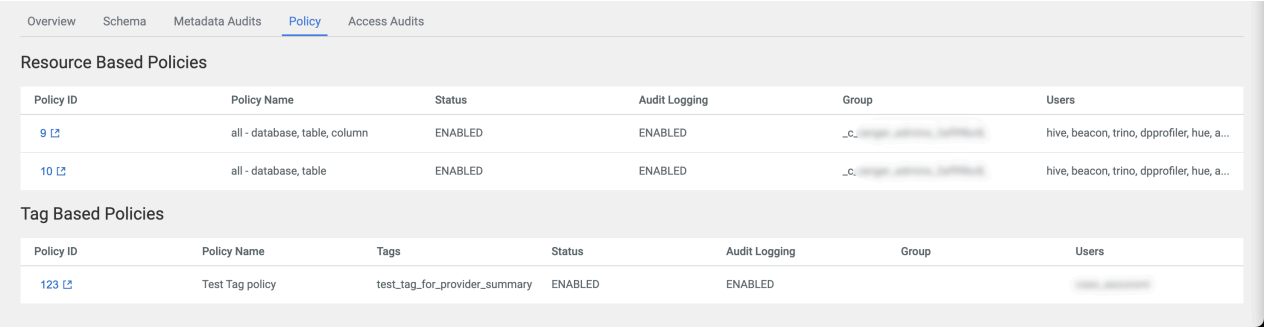

By clicking the Policy ID, you can directly open the relevant policy in Ranger for editing.

Related Information

Resource-based Services and Policies

Tag-based Services and Policies

Managing Profilers

The Cloudera Data Catalog profiler engine runs data profiling operations on data located in multiple data lakes. These profilers create metadata annotations that summarize the content and shape characteristics of the data assets.

Table 1: List of built-in profilers

Profiler Name	Description
Cluster Sensitivity Profiler	A sensitive data profiler- PII, PCI, HIPAA, etc.

Profiler Name	Description
Ranger Audit Profiler	A Ranger audit log summarizer.
Hive Column Profiler	Provides summary statistics like Maximum, Minimum, Mean, Unique, and Null values at the Hive column level.

Limitations

- In VM-based environments, profilers do not support Iceberg Tables. However, Iceberg tables are discoverable. In Compute Cluster enabled environments, Iceberg tables can be profiled.
- In Compute Cluster enabled environments, profilers only support tables which are stored on AWS S3 storage.
- Supported file formats:
  - VM-based environments:
    - CSV
  - Compute Cluster enabled environments:
    - Hive Column Profilers and Cluster Sensitivity Profilers
      - CSV
      - Parquet
      - Iceberg tables

Related Information

Understanding the Cloudera Data Catalog Profiler
Understanding the Cluster Sensitivity Profiler
Understanding the Ranger Audit Profiler

Profiler data testing

You must note the important information about profiler services.

Note: The Cloudera Data Catalog profilers are not tested at par with the Hive scale.

The following dataset has been validated and works as expected for VM-based environments:

- DataHub Master: m5.4xlarge
- Hive tables: 3000 Hive assets
- Total Number of assets (including Hive columns, tables, databases): 1,000,000
- Total Data Size = 1 GB
- Partitions on Hive tables: Around 5000 partitions spread across five tables

Note: For Compute Cluster enabled environments, more detailed testing information will be provided in the next release of Cloudera Data Catalog.

The following dataset has been validated and works as expected for Compute Cluster enabled environments:

- Total Data Size = 300 GB
- Sampling profiler size = 50% (150 GB)

Launching profilers

In VM-based environments, you must first provision the Cloudera Data Hub to launch the profiler cluster to view the profiler results for your assets and datasets. In Compute Cluster enabled environments, after you set up the profiler, the Profiler Launcher Services automatically starts the profiler Kubernetes containers.

Note: You must be a Power User to launch a profiler cluster.

Profiler cluster in VM based environments

The Profiler Services supports enabling the High Availability (HA) feature.

Note: The profiler HA feature is under entitlement. Based on the entitlement, the HA functionality is supported on the Profiler cluster. Contact your Cloudera account representative to activate this feature in your Cloudera environment.

Attention: By default when you launch a profiler cluster, the instance type of the Master node will be the following based on the provider:

- AWS - m5.4xlarge
- Azure - Standard_D16_v3
- GCP - e2-standard-16

Note: This is applicable from the following build of Cloudera Data Catalog: 2.0.17: 2.0.17-b26.

There are two types of Profiler Services:

- Profiler Manager
- Profiler Scheduler

The Profiler Manager service consists of profiler administrators, metrics, and data discovery services. These three entities support HA. The HA feature supports Active-Active mode.

Important: The Profiler Scheduler service does not support the HA functionality.

How to launch the profiler cluster for VM based environments

On the **Search** page, select the data lake from which you want to launch the profiler cluster. Click the Get Started link to proceed.

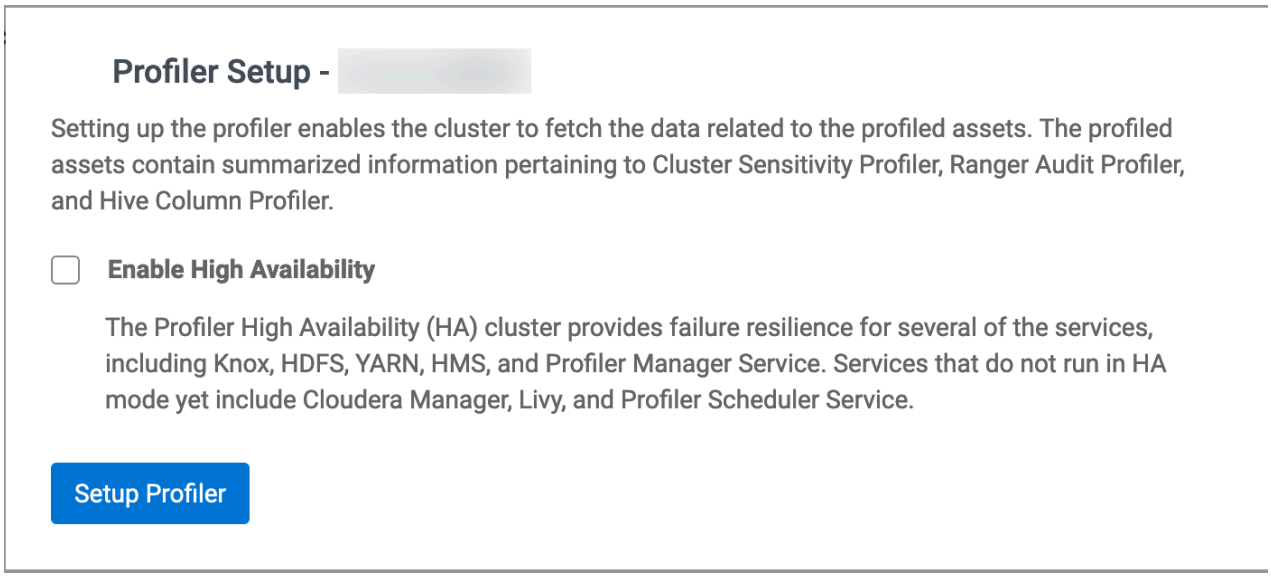

For setting up the profiler, you have the option to enable or disable the HA.

Note: The HA functionality is being supported only from Cloudera Runtime 7.2.10 release onwards. If you are using a Cloudera Runtime version below 7.2.10, you are not able to use the HA feature when launching the profiler services.

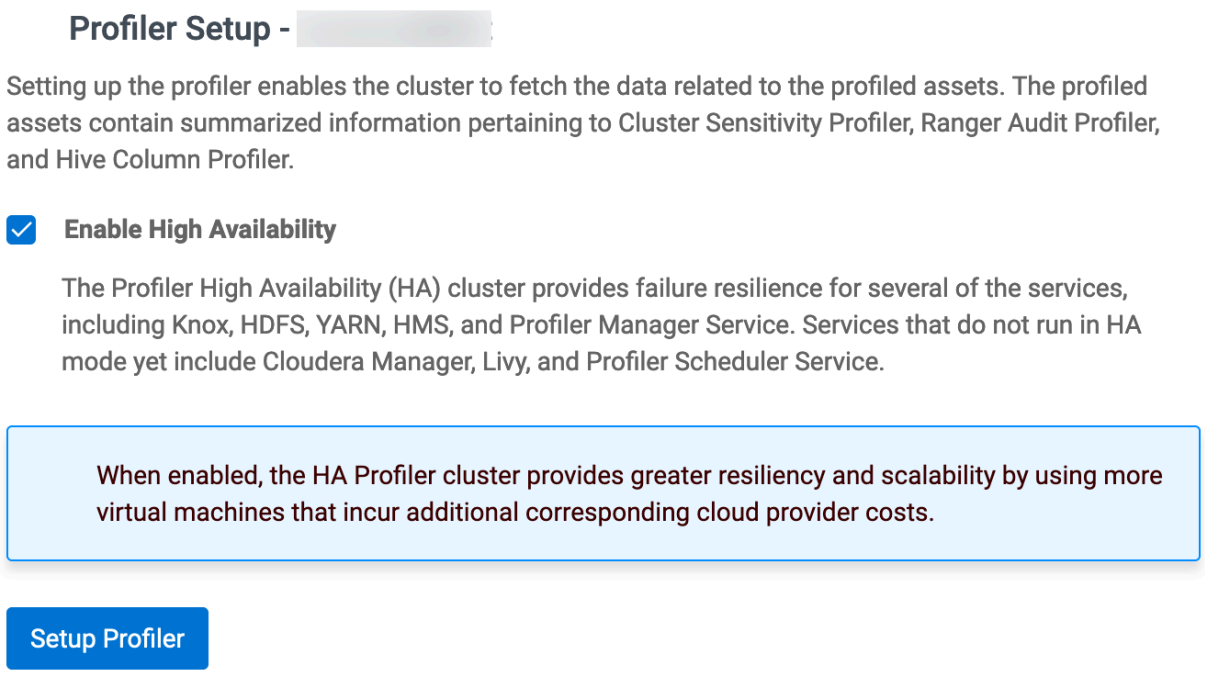

Once you enable HA and click Setup Profiler, Cloudera Data Catalog processes the request and the profiler creation is in progress.

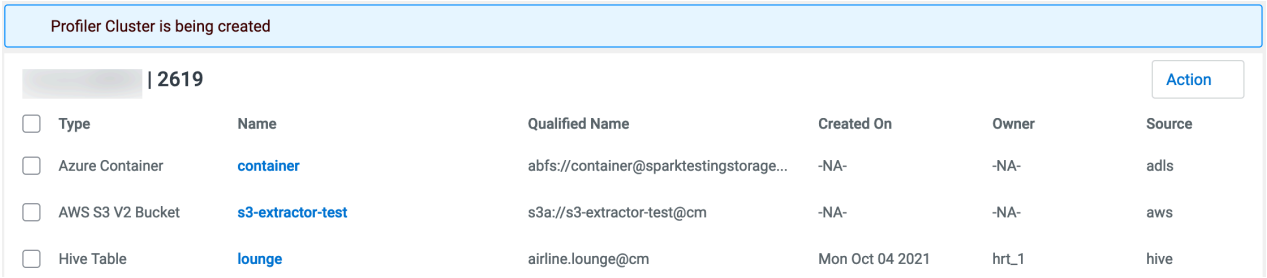

Later, a confirmation message appears that the profiler cluster is created.

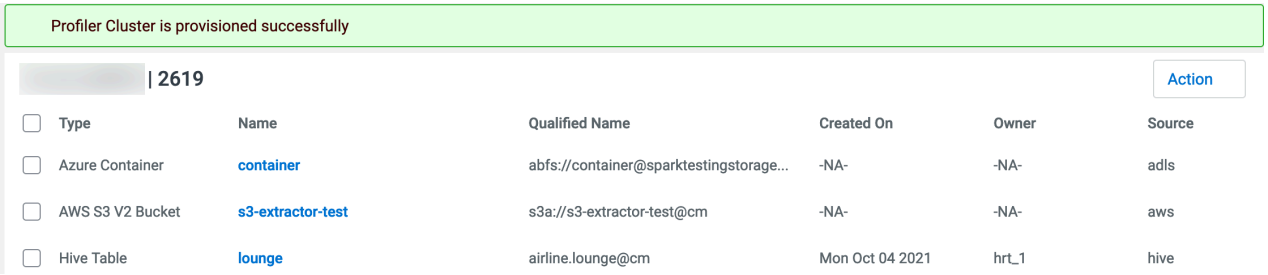

Next, you can verify the profiler cluster creation under Cloudera Management Console Environments Data Hubs pane.

The newly created profiler cluster looks like the following in Cloudera Management Console:

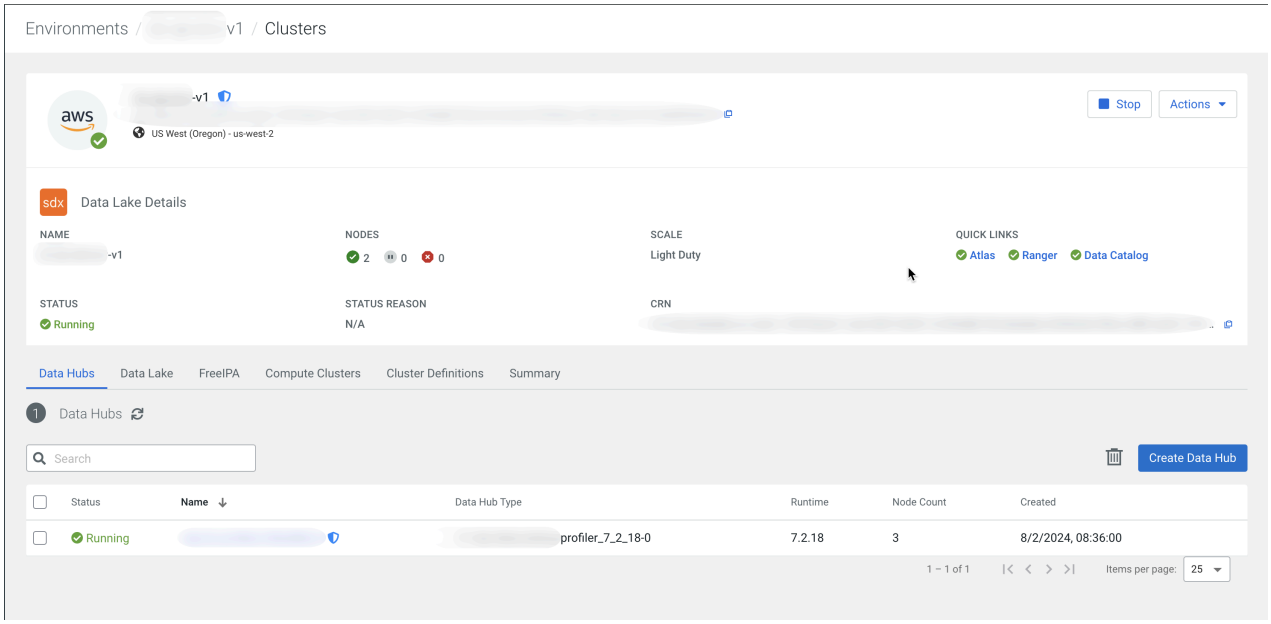

How to launch the profiler for Compute Cluster enabled environments

On the **Search** page, select the data lake from which you want to launch the profiler cluster. Click the Get Started link to proceed.

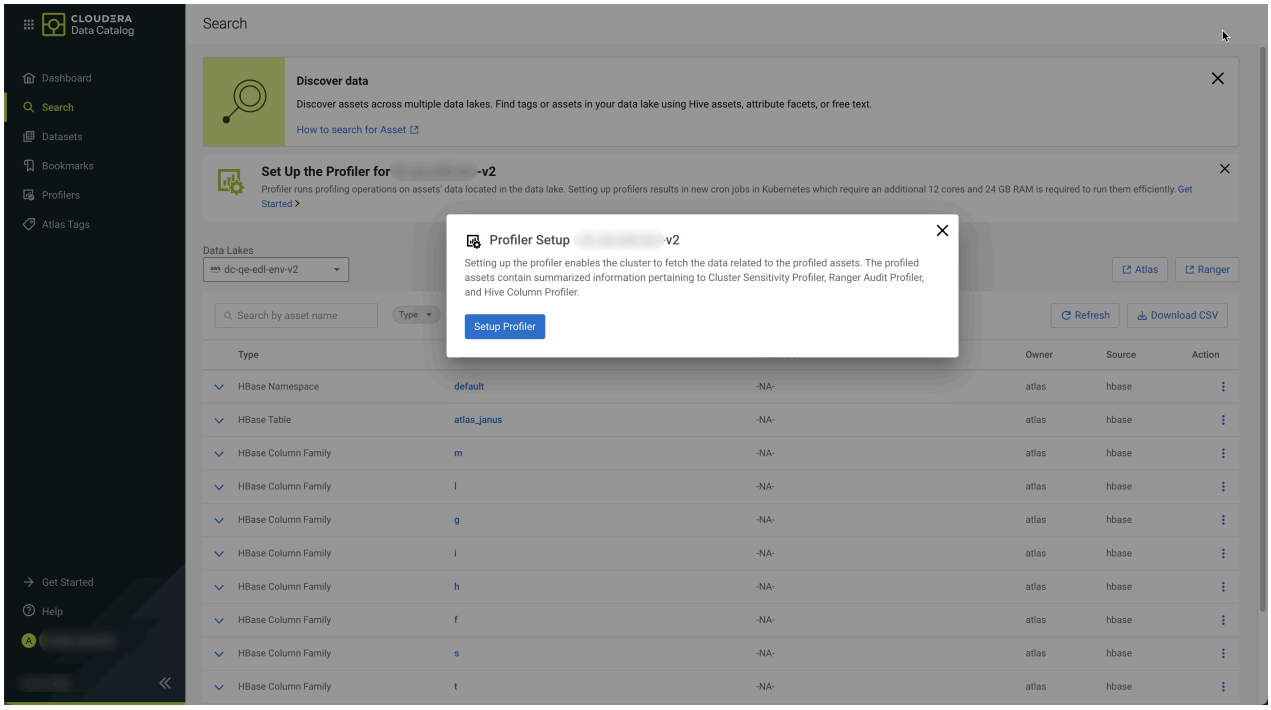

Click Setup Profiler, Cloudera Data Catalog processes the request and the profiler creation will start.

Next, you can verify that the profiler jobs are running under the Cloudera Management Console Environments Compute Clusters Node Groups pane.

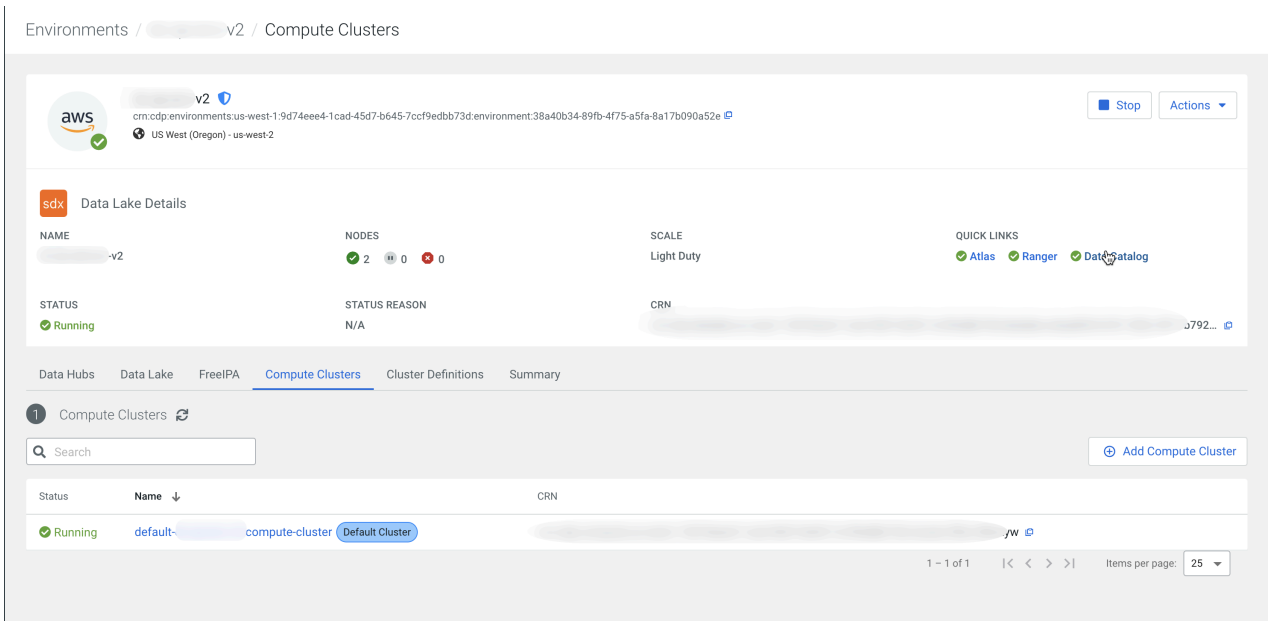

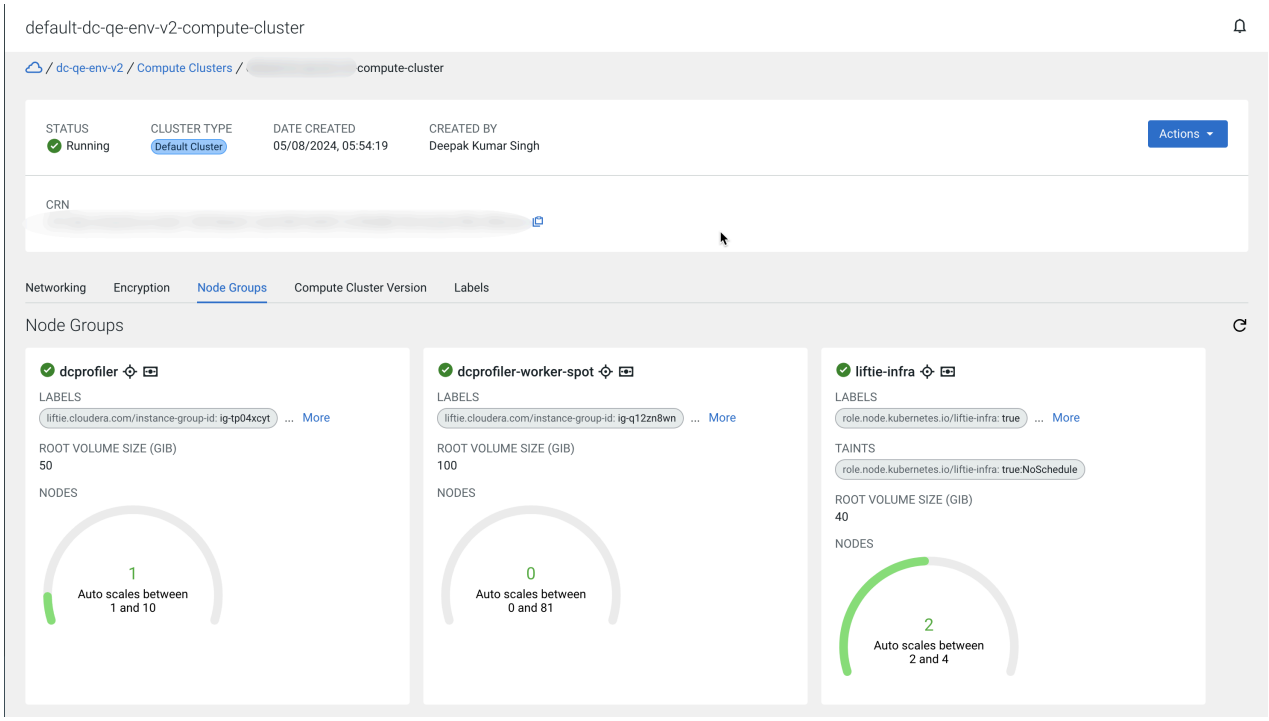

Related Information

Understanding the Cloudera Data Catalog Profiler
Understanding the Cluster Sensitivity Profiler
Understanding the Ranger Audit Profiler

Launching profilers using the command-line

Cloudera Data Catalog supports launching profilers using the Command-Line Interface (CLI) option.

The CLI is one executable and does not have any external dependencies. You can execute some operations in the Cloudera Data Catalog service using the Cloudera CLI commands.

Users must have valid permissions to launch profilers on a data lake.

For more information about the access details, see Prerequisites to access Cloudera Data Catalog.

Prerequisites

You must have the following entitlement granted to use this feature:

DATA_CATALOG_ENABLE_API_SERVICE

For more information about the Cloudera command-line interface and setting up the same, see Cloudera CLI.

The Cloudera Data Catalog CLI

In your Cloudera CLI environment, enter the following command to get started in the CLI mode.

cdp datacatalog --help

This command provides information about the available commands in Cloudera Data Catalog for Cloudera Public Cloud 7.2.18. and earlier versions.

The output is displayed as:

```
NAME
datacatalog
DESCRIPTION
Cloudera Data Catalog Service is a web service, using this service user can
 execute operations like launching profilers in Data Catalog.
AVAILABLE SUBCOMMANDS
launch-profilers
```

You get additional information about this command by using:

cdp datacatalog launch-profilers --help

```
NAME
launch-profilers -
DESCRIPTION
Launches DataCatalog profilers in a given datalake.
```

```
SYNOPSIS
 launch-profilers
 --datalake <value>
 [--cli-input-json <value>]
 [--generate-cli-skeleton]
OPTIONS
 --datalake (string)
The CRN of the Datalake.
--cli-input-json
(string) Performs service operation based on the JSON string provided. The
JSON string follows the format provided by --generate-cli-skeleton. If other
 arguments are provided on the command line, the CLI values will override th
e JSON-provided values.
--generate-cli-skeleton
(boolean) Prints a sample input JSON to standard output. Note the specified
operation is not run if this argument is specified. The sample input can be
used as an argument for --cli-input-json.
```

```
OUTPUT
.
datahubCluster -> (object)
 Information about a cluster.
clusterName -> (string)
 The name of the cluster.
 crn -> (string)
 The CRN of the cluster.
 creationDate -> (datetime)
```

```
 The date when the cluster was created.
 clusterStatus -> (string)
 The status of the cluster.
 nodeCount -> (integer)
 The cluster node count.
 workloadType -> (string)
 The workload type for the cluster. cloudPlatform -> (string) The cloud plat
form.
 imageDetails -> (object)
The details of the image used for cluster instances.
 name -> (string)
The name of the image used for cluster instances.
 id -> (string)
The ID of the image used for cluster instances.
This is internally generated by the cloud provider to Uniquely identify the
 image.
 catalogUrl -> (string)
 The image catalog URL.
 catalogName -> (string)
 The image catalog name.
 environmentCrn -> (string)
 The CRN of the environment.
 credentialCrn -> (string)
 The CRN of the credential.
 datalakeCrn -> (string)
 The CRN of the attached datalake.
 clusterTemplateCrn -> (string)
The CRN of the cluster template used for the cluster creation.
```

Launching the profiler

You can use the following CLI command to launch the data profiler:

```
cdp datacatalog launch-profilers --datalake [***DATALAKE CRN***]
```

Example:

```
cdp datacatalog launch-profilers --datalake crn:cdp:data
lake:DATACENTERNAME:c*****b-ccce-4**d-a**1-8********8:datalake:4*****5e-c**
1-4**2-8**e-1********2
{
    "success": true
}
```

Related Information

Cloudera CLI 0.9.124 documentation / datacatalog command

Deleting profilers

In VM-based environments, deleting the profiler cluster (or in Compute Cluster enabled environments deleting the profiler jobs) removes all the Custom Sensitivity Profiler rules and other updates to the specific cluster. It could also cause loss of data specific to currently applied rules on the deleted profiler cluster.

About this task

To overcome this situation, when you decide to delete the profiler cluster or (in VM-based environments) the profiler jobs by Compute Cluster enabled environments, there is a provision to retain the status of the Custom Sensitivity Profiler rules:

- If your profiler cluster or profiler jobs have rules that are not changed or updated, you can directly delete them or the profiler cluster.
- If the rules were modified or updated, you have an option to download the modified rules along with deletion. The modified rules consist of the suspended system rules and the deployed custom rules. Using the downloaded rules, you can manually add or modify them to your newly added profiler jobs or the profiler cluster.

Note:

- In a Compute Cluster enabled environment, when you delete the scheduled jobs, the associated Kubernetes cron job object is deleted from the Kubernetes cluster.
- The associated data of the profilers from the Cloudera Management Console database is also deleted for the specified data lake.

Procedure

1. On the **Search** page, select the data lake from the drop-down.
2. Click Delete Profiler.

3. If you agree, select the warning message I understand this action cannot be undone.

Figure 1: Deleting a profiler in a VM-based environment

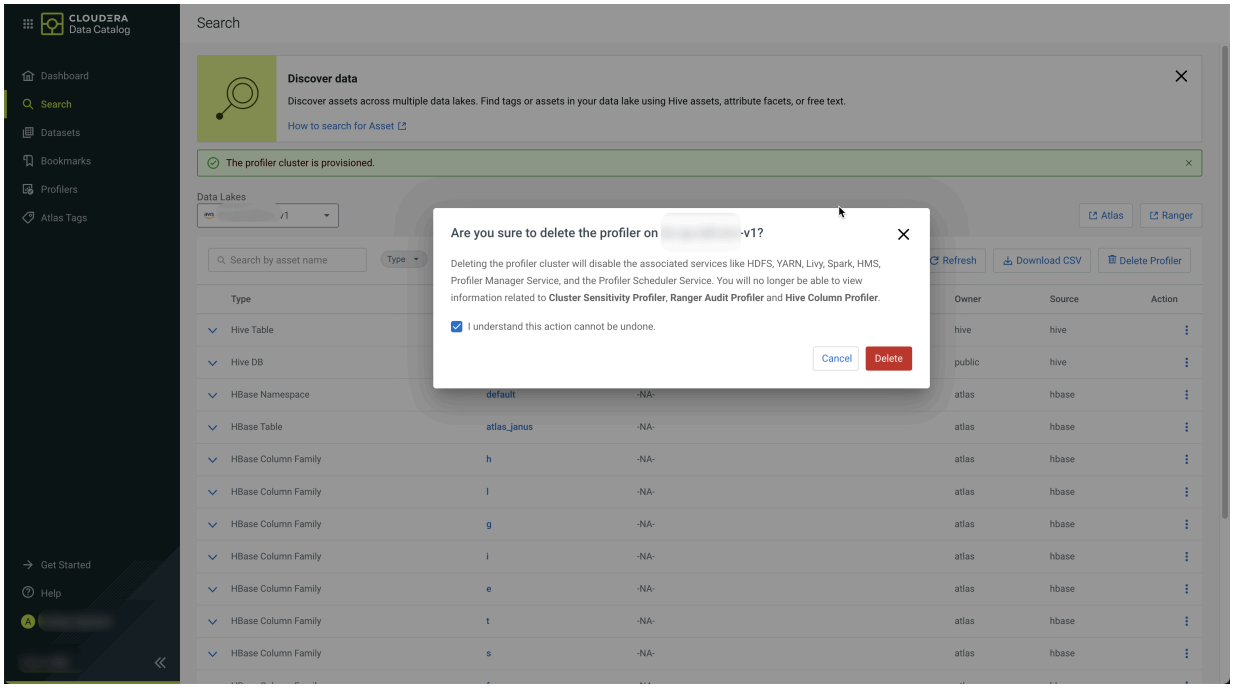

Figure 2: Deleting a profiler in a Compute Cluster enabled environment

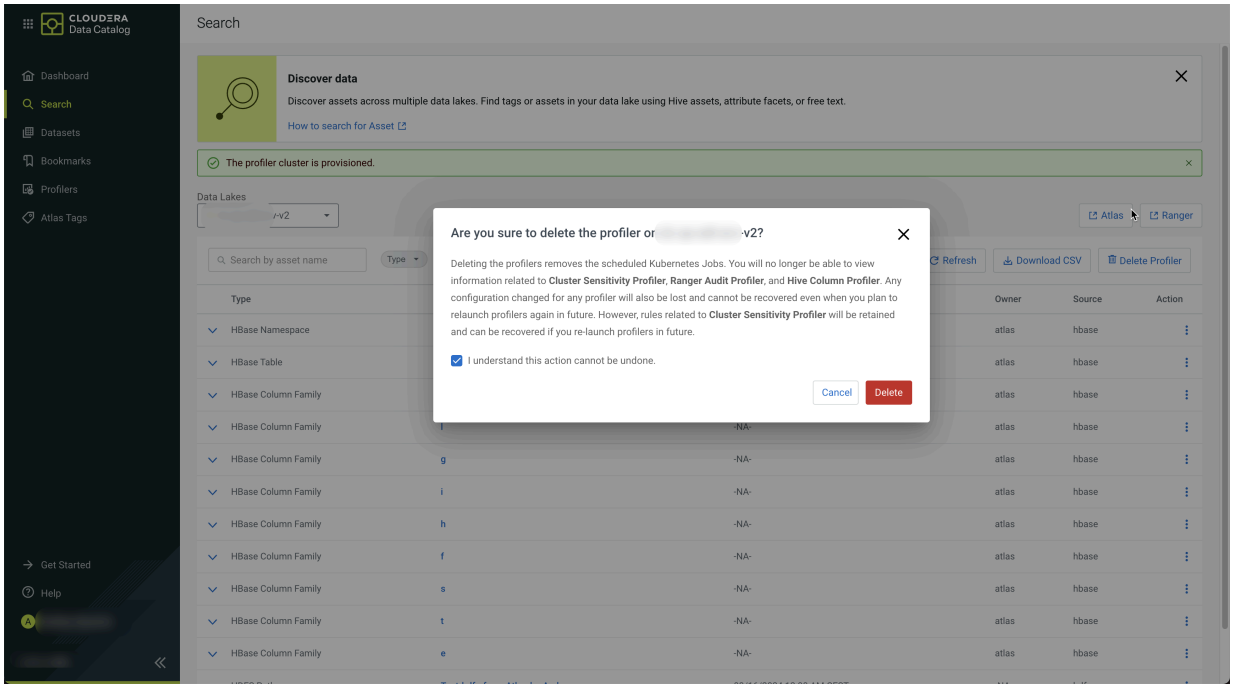

4. Click Delete.

   The application displays the following message.

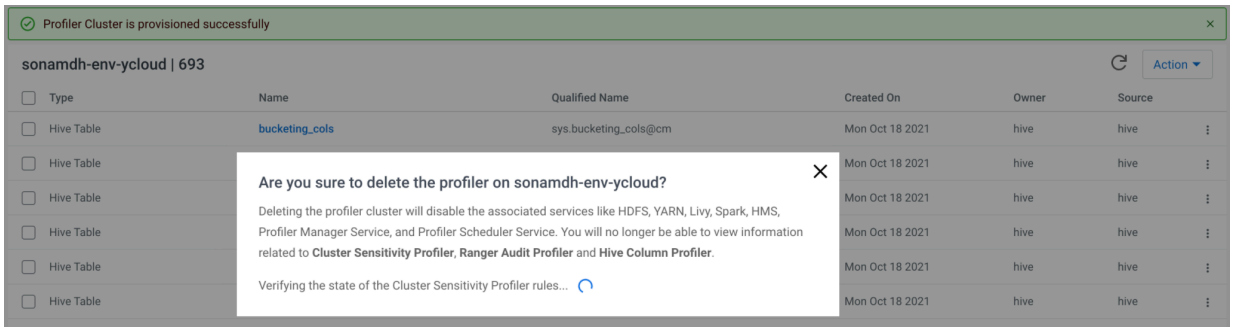

Note: When you launch Cloudera Data Catalog in Cloudera Runtime version 7.2.14, and later if the profiler cluster is deleted, the following message is displayed.

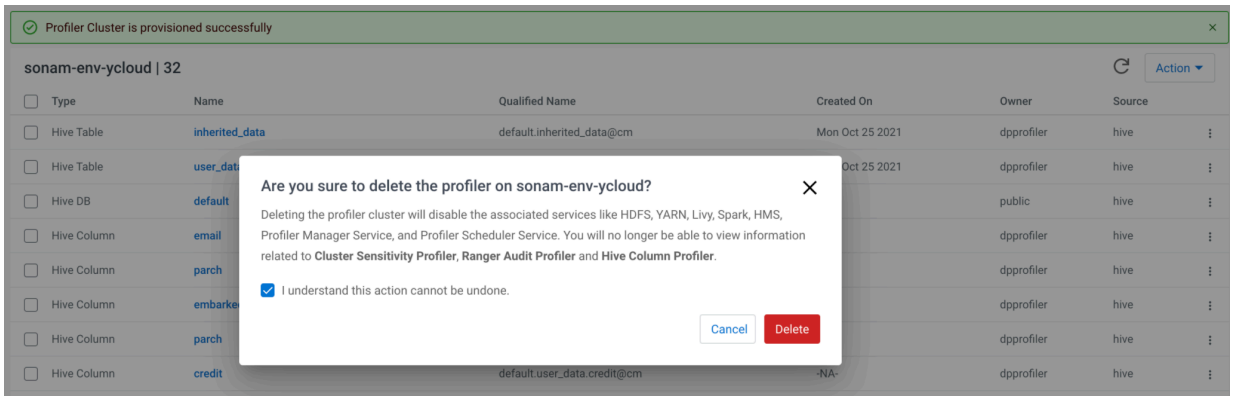

Additionally, note that you can delete the profiler cluster in these situations, when:

- Profiler cluster is up and running
- Profiler cluster is created but stopped
- Profiler cluster creation failed but is registered with the data lake
- Profiler cluster is down and inaccessible

Note: In VM-based environments, if the profiler cluster is not registered with the data lake, Cloudera Data Catalog cannot locate or trace the profiler cluster. You have to delete the profiler cluster from the Cloudera Data Hub page (Cloudera Management Console).

The profiler cluster is deleted successfully.

On-Demand Profilers

You can use On-Demand Profilers to profile specific assets without depending on the cron-based scheduling of profilers jobs. The On-Demand Profiler option is available in the Asset Details of the selected asset.

The following image shows the **Asset Details** page of an asset. You can run an On-Demand Profiler for Hive Column Profiler and Cluster Sensitivity Profiler by clicking on the appropriate Run button. The next scheduled run provides details about the next scheduled profiling for the respective profilers.

Note: You can use the On-Demand Profiler feature to profile both external and managed tables.

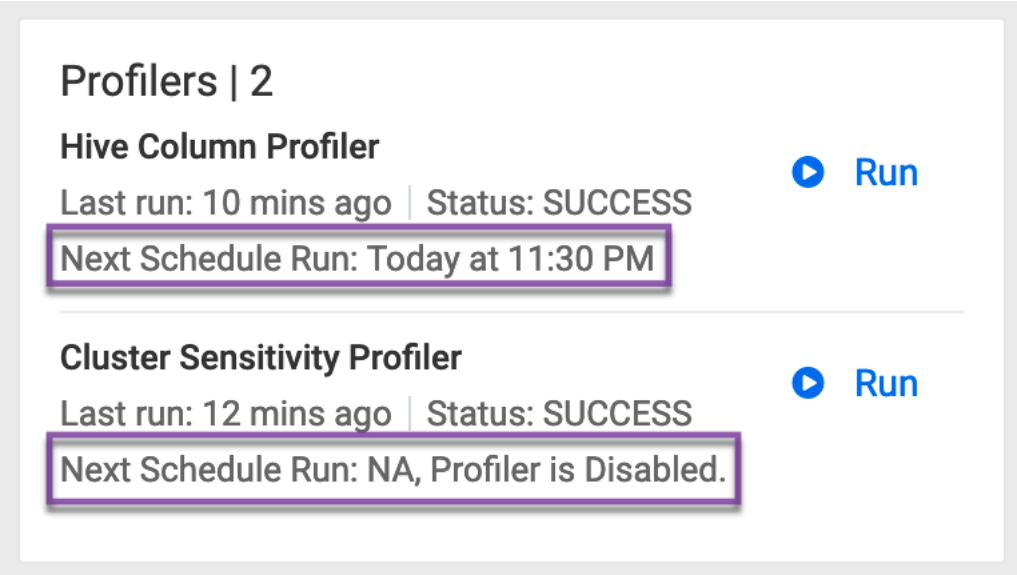

Note: In Compute Cluster-enabled environments, Iceberg tables can be also profiled with the On-Demand Profiler.

Profiling table data in non-default buckets

In VM-based environments, you must configure a parameter in Profiler Scheduler in your instance to profile table data in non-default buckets.

Procedure

1. In Cloudera Data Catalog, make a note of your environment's name in the **Search** menu.
2. Go Cloudera Management Console Environments
3. Search for your environment, then switch to the **Data hubs** tab.
4. Open you Cloudera Data Hub by clicking its name.
5. Open the CM URL under **Cloudera Manager Info**.
6. In Cloudera Manager go to Configuration Configuration Search .
7. Search for the term Profiler Scheduler Spark conf.
   The **Profiler Scheduler Spark conf** configuration snippet appears.

8. Add spark.yarn.access.hadoopFileSystems=s3a://default-bucket,s3a://bucket-1,s3a://bucket-2 to **Profiler Scheduler Spark conf** to enable profiling for bucket-1 and bucket-2 non-default buckets.

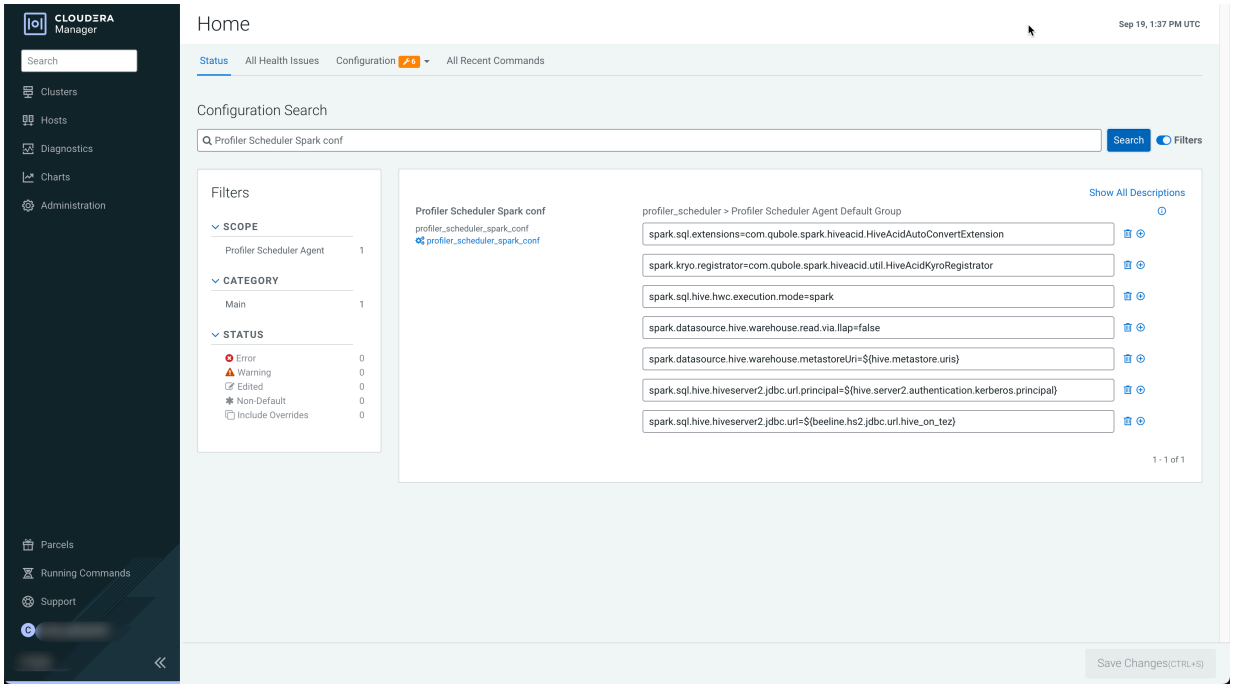

Tracking profiler jobs

Use the Profilers > Jobs page for tracking respective profiler jobs.

Under Profilers Jobs , you can have an overview of your started profiler jobs. By using the D, W, M filters, you can go back up to a day, week or a moth, to see your previous jobs. Use this page to quickly check if your profiler jobs are failing.

Figure 3: Profiling jobs in a Compute Cluster enabled environment

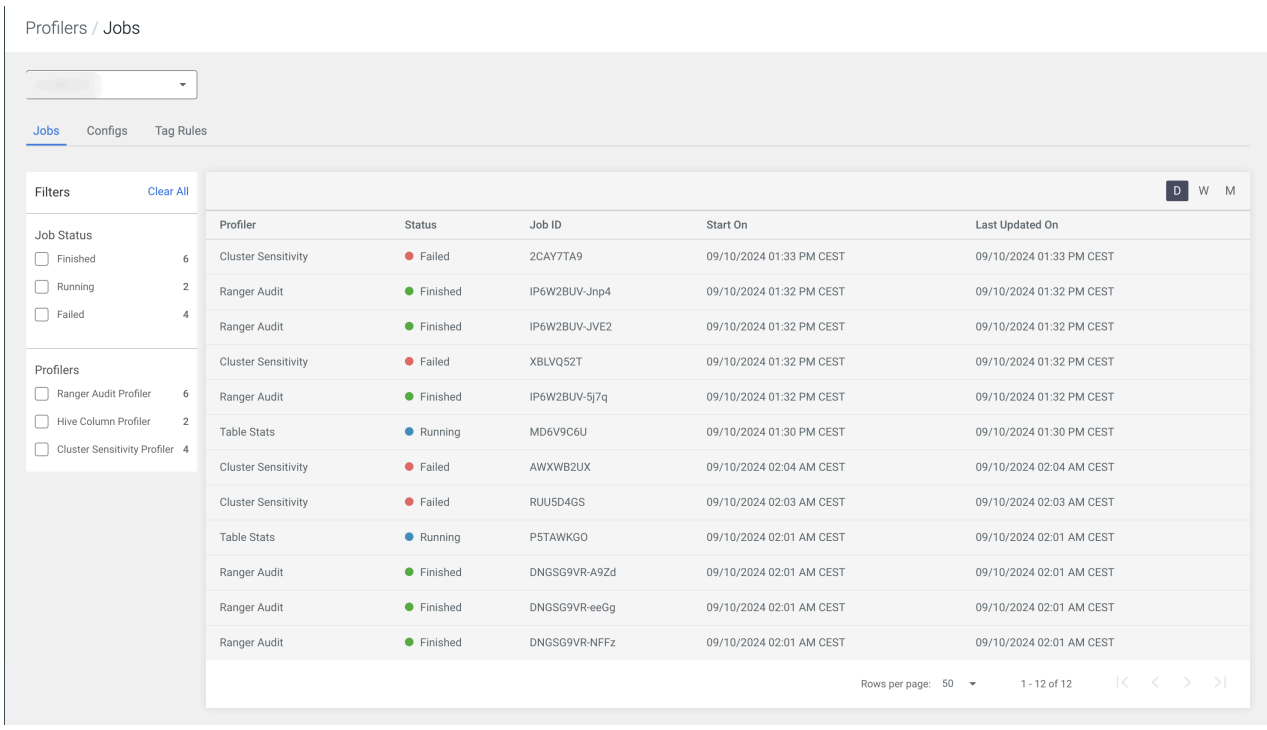

In VM-based environments, Profilers Jobs can show you the current profiling **Stage** based on the relevant service used:

Figure 4: Profiling jobs in a VM-based environment

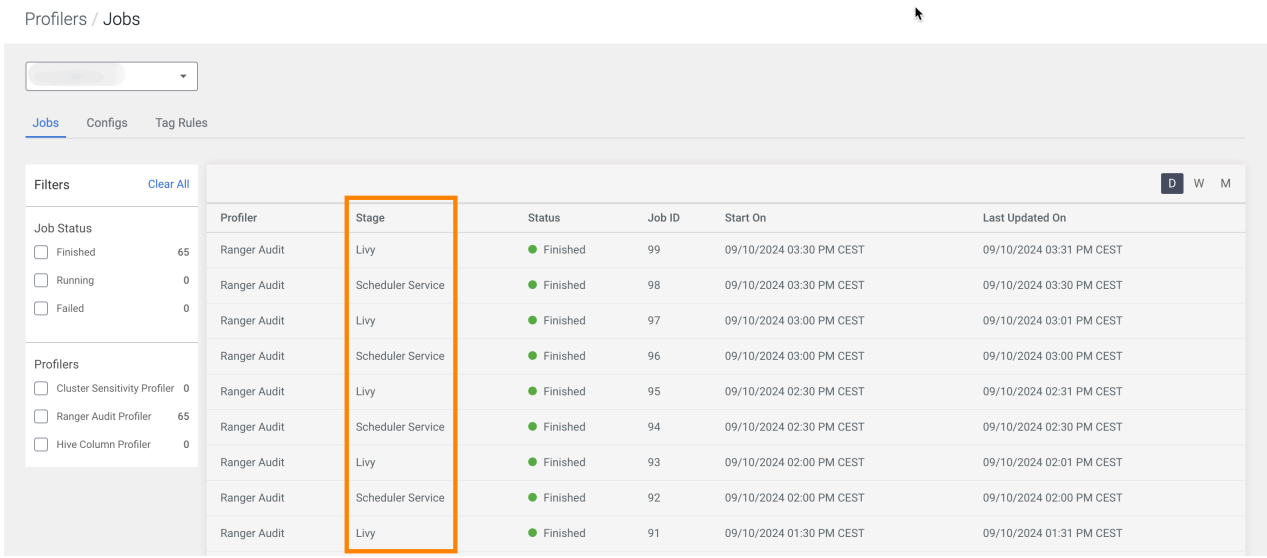

For each profiler job, you can view the details about:

- **Profiler** type
- **Stage** (for VM-based environments)
- Job **Status**
- **Job ID**
- **Start Time**
- **Last Update On** time

Using this data can help you to troubleshoot failed jobs or even understand how the jobs were profiled and other pertinent information that can help you to manage your profiled assets.

In VM-based environments, profiler job runs ins the following phases:

- Scheduler Service - The part of Profiler Admin which queues the profiler requests.
- Livy - This service is managed by YARN and where the actual asset profiling takes place.
- Metrics Service - Reads the profiled data files and publishes them.

Note: More than one occurrence of Scheduler Service or Livy indicates that there could be more assets to be profiled. For example, if an HBase schedule has about 80 assets to be profiled, the first 50 assets would be profiled in the first Livy batch and the other assets get profiled in the next batch.

In case of Ranger Audit profiling, there could be a “NA” status for the total number of assets profiled. It indicates that the auditing that happens is dependent on the Ranger policies. In other words, the Ranger policies are actually profiled and not the assets.

Related Information

Understanding the Cloudera Data Catalog Profiler
Understanding the Cluster Sensitivity Profiler
Understanding the Ranger Audit Profiler

Viewing profiler configurations

You can monitor the last status of individual profilers by viewing them in Profiler > Configs. Also, you can change their resources, sensitivity and scheduling.

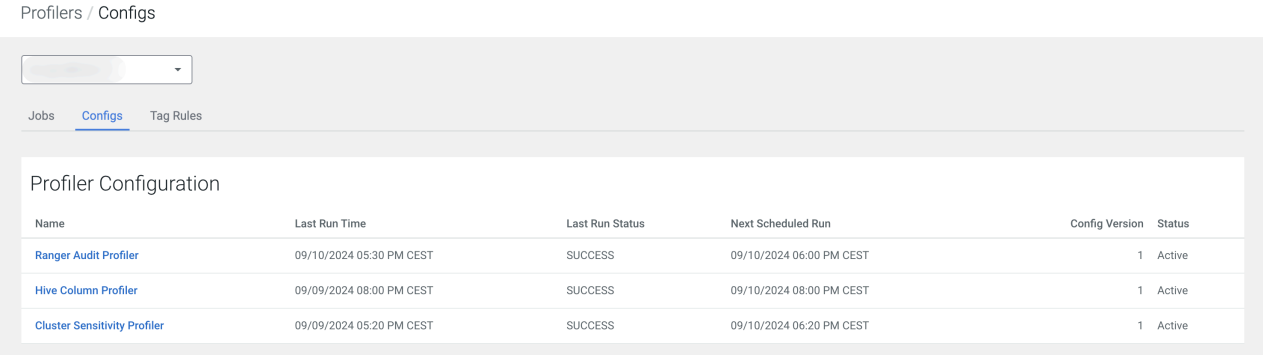

Monitoring the profiler configurations has the following uses:

- Verify which profilers are active or inactive.
- Verify the status of the profiler runs.
- View the last run time and status and the next scheduled run.

Note: You can also filter your profilers by job status, type for the last day, week and month.

Related Information

Understanding the Cloudera Data Catalog Profiler
Understanding the Cluster Sensitivity Profiler
Understanding the Ranger Audit Profiler

Ranger Audit Profiler configuration

In addition to the generic configuration, there are additional parameters for the Ranger Audit Profiler that can optionally be edited.

Procedure

1. Go to Profilers Configs .
2. Select your data lake.
3. Select Ranger Audit Profiler.
   The **Detail** page is displayed.
4. Use the toggle button

   to enable or disable the profiler.
5. Select a schedule to run the profiler. This is implemented as a quartz cron expression.
6. Continue with the resource settings.

For VM-based environment

a. In **Advanced Options**, set the following:

- Number of Executors - Enter the number of executors to launch for running this profiler.
- Executor Cores - Enter the number of cores to be used for each executor.
- Executor Memory - Enter the amount of memory in GB to be used per executor process.
- Driver Cores - Enter the number of cores to be used for the driver process.
- Driver Memory - Enter the memory to be used for the driver processes.

Note: For more information, see Configuring SPARK on YARN Applications and Tuning Resource Allocation.

For Compute Cluster enabled environment

a. In **Pod Configurations**, set the Kubernetes job resources.

Pod configurations specify the resources that would be allocated to a pod when the profiler job starts to run. As all profilers are submitted as Kubernetes jobs, you must decide if you want to add or reduce resources to handle workload of various sizes.

- **Pod CPU Limit**: Indicates the maximum number of cores that can be allocated to a Pod. The accepted values range from one through eight.
- **Pod CPU Requirement**: This is the minimum number of CPUs that will be allocated to a Pod when its provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit. The accepted values range from one through eight.
- **Pod Memory Limit**: Maximum amount of memory can be allocated to a Pod. The accepted values range from 1 through 256.
- **Pod Memory Requirement**: This is the minimum amount of RAM that will be allocated to a Pod when it is provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a

container is not allowed to use more than its resource limit. The accepted values range from 1 through 256.

b. In **Executor Configurations**, update the following:

- **Number of workers**: Indicates the number of processes that are used by the distributed computing framework. The accepted values range from one through eight.
- **Number of threads per worker**: Indicates the number of threads used by each worker to complete the job. The accepted values range from one through eight.
- **Worker Memory limit in GB**: To avoid over utilization of memory, this parameter forces an upper threshold memory usage for a given worker. For example, if you have a 8 GB Pod and 4 threads, the value of this parameter must be 2 GB. The accepted values range from one through four.

Executor configurations are the runtime configurations. These configuration must be changed if you are changing the pod configurations and when there is a requirement for additional compute power.

7. Click Save to apply the configuration changes to the selected profiler.

Related Information

Understanding the cron expression generator

Assign CPU Resources to Containers and Pods

Assign Memory Resources to Containers and Pods

Cluster Sensitivity Profiler configuration

In addition to the generic configuration, there are additional parameters for the Cluster Sensitivity Profiler that can optionally be edited.

Procedure

1. Go to Profilers Configs .

2. Select your data lake.

3. Select Cluster Sensitivity Profiler.
The **Detail** page is displayed which contains the following sections:

4. Use the toggle button

to enable or disable the profiler.

5. Select a schedule to run the profiler. This is implemented as a quartz cron expression.
For more information, see Understanding the Cron Expression generator on page 68.

6. Select Last Run Check and set a period if needed.

Note:

The Last Run Check enables profilers to avoid profiling the same asset on each scheduled run.

If you have scheduled a cron job, for example set to start in about an hour, and have enabled the Last Run Check configuration for two days, this setup ensures that the job scheduler filters out any asset which was already profiled in the last two days.

If the Last Run Check configuration is disabled, assets will be picked up for profiling as per the job cron schedule, honoring the asset filter rules.

7. Set the sample settings for VM-based environments:
   a. Select the **Sample Data Size**.
      1. From the drop down, select the type of sample data size.
      2. Enter the value based on the previously selected type.

      Note: In Compute Cluster enabled environments, skip to step 8 on page 65.

8. Continue with the resource settings.

For VM-based environment

a. In **Advanced Options**, set the following:
   - Number of Executors - Enter the number of executors to launch for running this profiler.
   - Executor Cores - Enter the number of cores to be used for each executor.
   - Executor Memory - Enter the amount of memory in GB to be used per executor process.
   - Driver Cores - Enter the number of cores to be used for the driver process.
   - Driver Memory - Enter the memory to be used for the driver processes.

   Note: For more information, see Configuring SPARK on YARN Applications and Tuning Resource Allocation.

For Compute Cluster enabled environment

a. In **Pod Configurations**, update the following:
   - **Pod CPU Limit**: Indicates the maximum number of cores that can be allocated to a Pod. The accepted values range from one through eight.
   - **Pod CPU Requirement**: This is the minimum number of CPUs that will be allocated to a Pod when its provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit. The accepted values range from one through eight.
   - **Pod Memory Limit**: Maximum amount of memory can be allocated to a Pod. The accepted values range from 1 through 256.
   - **Pod Memory Requirement**: This is the minimum amount of RAM that will be allocated to a Pod when it is provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit. The accepted values range from 1 through 256.

b. In **Executor Configurations**, update the following:
   - **Number of workers**: Indicates the number of processes that are used by the distributed computing framework. The accepted values range from one through eight.
   - **Number of threads per worker**: Indicates the number of threads used by each worker to complete the job. The accepted values range from one through eight.
   - **Worker Memory limit in GB**: To avoid over utilization of memory, this parameter forces an upper threshold memory usage for a given worker. For example, if you have a 8 GB Pod and 4 threads, the value of this parameter must be 2 GB. The accepted values range from one through four.

   Executor configurations are the runtime configurations. These configuration must be changed if you are changing the pod configurations and when there is a requirement for additional compute power.

9. Add **Asset Filter Rules** as needed to customize the selection and deselection of assets which the profiler profiles.
   a) Set your **Deny List** and **Allow-list**.

      The profiler will skip profiling assets that meet any criteria in the **Deny List** and will include assets that meet any criteria in the **Allow List**.

      1. Select the **Deny-list** or **Allow List** tab.
      2. Click Add New to include rules.
      3. Select the key from the drop-down list. You can select from the following:
         - Database name
         - Asset name
         - Asset owner
         - Path to the asset
         - Created date
      4. Select the operator from the drop-down list. Depending on the keys selected, you can select an operator such as equals, contains. For example, you can select the name of assets that contain a particular string.
      5. Enter the value corresponding to the key. For example, you can enter a string as mentioned in the previous example.
      6. Click Done. Once rule is added, you can toggle the state of the new rule to enable it or disable it as needed.
10. Click Save to apply the configuration changes to the selected profiler.

Related Information

Understanding the Cloudera Data Catalog Profiler
Understanding the Cluster Sensitivity Profiler
Understanding the cron expression generator
Assign CPU Resources to Containers and Pods
Assign Memory Resources to Containers and Pods

Associating Cluster Sensitivity Profiler tags to Hive assets

Associating Custom Sensitivity Profiler tags to Hive assets fails on the 7.2.16 Cloudera Runtime cluster.

The issue can be resolved by adding the following Spark configuration:

spark.sensitive.profiler.tableSamplePercentage=50 to profiler scheduler conf by adding the configuration under the key - profiler_scheduler_spark_conf.

The value passed in the configuration must be an integer value ranging from 1 to 100.

This value indicates the percentage of rows to be picked up as a sample for the profiler job.

Hive Column Profiler configuration

In addition to the generic configuration, there are additional parameters for the Hive Column Profiler that can optionally be edited.

Procedure

1. Go to Profilers Configs .
2. Select your data lake.
3. Select Hive Column Profiler.
   The **Detail** page is displayed.
4. Use the toggle button

to enable or disable the profiler.

5. Select a schedule to run the profiler. This is implemented as a quartz cron expression.

   For more information, see Understanding the Cron Expression generator on page 68.

6. Select Last Run Check and set a period if needed.

   Note:

   The Last Run Check enables profilers to avoid profiling the same asset on each scheduled run.

   If you have scheduled a cron job, for example set to start in about an hour, and have enabled the Last Run Check configuration for two days, this setup ensures that the job scheduler filters out any asset which was already profiled in the last two days.

   If the Last Run Check configuration is disabled, assets will be picked up for profiling as per the job cron schedule, honoring the asset filter rules.

7. Set the sample settings:

   a. Select the **Sample Data Size**.

      1. From the drop down, select the type of sample data size.
      2. Enter the value based on the previously selected type.

      Note: In Compute Cluster enabled environments, skip to step 8 on page 67.

8. Continue with the resource settings.

For VM-based environment

a. In **Advanced Options**, set the following:

- Number of Executors - Enter the number of executors to launch for running this profiler.
- Executor Cores - Enter the number of cores to be used for each executor.
- Executor Memory - Enter the amount of memory in GB to be used per executor process.
- Driver Cores - Enter the number of cores to be used for the driver process.
- Driver Memory - Enter the memory to be used for the driver processes.

Note: For more information, see Configuring SPARK on YARN Applications and Tuning Resource Allocation.

For Compute Cluster enabled environment

a. In **Pod Configurations**, update the following:

- **Pod CPU Limit**: Indicates the maximum number of cores that can be allocated to a Pod. The accepted values range from one through eight.
- **Pod CPU Requirement**: This is the minimum number of CPUs that will be allocated to a Pod when its provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit. The accepted values range from one through eight.
- **Pod Memory Limit**: Maximum amount of memory can be allocated to a Pod. The accepted values range from 1 through 256.
- **Pod Memory Requirement**: This is the minimum amount of RAM that will be allocated to a Pod when it is provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a

container is not allowed to use more than its resource limit. The accepted values range from 1 through 256.

b. In **Executor Configurations**, update the following:

- **Number of workers**: Indicates the number of processes that are used by the distributed computing framework. The accepted values range from one through eight.
- **Number of threads per worker**: Indicates the number of threads used by each worker to complete the job. The accepted values range from one through eight.
- **Worker Memory limit in GB**: To avoid over utilization of memory, this parameter forces an upper threshold memory usage for a given worker. For example, if you have a 8 GB Pod and 4 threads, the value of this parameter must be 2 GB. The accepted values range from one through four.

Executor configurations are the runtime configurations. These configuration must be changed if you are changing the pod configurations and when there is a requirement for additional compute power.

9. Add **Asset Filter Rules** as needed to customize the selection and deselection of assets which the profiler profiles.

a) Set your **Deny List** and **Allow-list**.

The profiler will skip profiling assets that meet any criteria in the **Deny List** and will include assets that meet any criteria in the **Allow List**.

1. Select the **Deny-list** or **Allow List** tab.
2. Click Add New to include rules.
3. Select the key from the drop-down list. You can select from the following:
   - Database name
   - Asset name
   - Asset owner
   - Path to the asset
   - Created date
4. Select the operator from the drop-down list. Depending on the keys selected, you can select an operator such as equals, contains. For example, you can select the name of assets that contain a particular string.
5. Enter the value corresponding to the key. For example, you can enter a string as mentioned in the previous example.
6. Click Done. Once rule is added, you can toggle the state of the new rule to enable it or disable it as needed.

10. Click Save to apply the configuration changes to the selected profiler.

Related Information

Understanding the cron expression generator
Assign CPU Resources to Containers and Pods
Assign Memory Resources to Containers and Pods

Understanding the Cron Expression generator

In the Profiler > Configs > Detail page, a cron expression defines when the profiler schedule executes and visualizes the next execution dates of your profiling jobs.

The Unix (in Compute Cluster enabled environments) and quartz (in VM-based environments) cron expression uses the following typical format:

Each * in the cron represents a unique value.

For VM-based environments

The ? character is used for undefined day of the month and the day of the week.

Schedule: * * * * * ? *

For example, consider a cron with the following values:

1 2 3 2 5 ? 2021

This cron expression is scheduled to run the profiler job at: 03:02:01am, on the 2nd day, in May, in 2021.

Note: The ? character is a replacement for the "day-of-the-week". It is not specified on which day of the week the job has to run.

For Compute Cluster enabled environments

Cron Expression: 0 18 * * *

In this format the * characters represent the following units:Minute hour day(month) month day(week)

For example, consider a cron with the following values:

CRON Expression: 30 10 15 5 *

This cron expression is scheduled to run the profiler job at: “At 10:30 on 15th day-of-month in May.”

Note: The * character is a replacement for the "day-of-the-week". It is not specified on which day of the week the job has to run.

Consider another example:

30 10 * 5 7

This cron expression is scheduled to run the profiler job at: “At 10:30 on Sunday in May”.

Note: The * character is a replacement for the "day-of the month". It is not specified on which day of the month the job has to run.

You can change the value of cron as and when it is required depending on how you want to schedule your profiler job.

Related Information

UNIX crontab documentation

Quartz Cron Trigger Tutorial

Backing up and restoring the profiler database

Using certain scripts that can be executed by the root users, you can back up of the profiler databases. Later, if you want to delete the existing Cloudera Data Hub cluster and launch a new cluster, you will have an option to restore the old data.

Important: Backing up and restoring the profiler database is only available in VM-based environments.

Cloudera Data Catalog includes profiler services that run data profiling operations on data that is located in multiple data lakes. In VM-based environments, the profiler services run on a Cloudera Data Hub cluster. When you delete the Cloudera Data Hub cluster, the profiled data and the user configuration information stored in the local databases are lost.

Profiler clusters run on the Cloudera Data Hub cluster using embedded databases:

- profiler_agent
- profiler_metrics

Note: If you download the modified Cluster Sensitivity Profiler rules before deleting the profiler cluster, and later when you create a new profiler cluster, you can restore the state of the rules manually. If the system rules are part of the downloaded files, you must Suspend those rules. If custom rules are part of the downloaded files, you must deploy those rules. This is applicable if the profiler cluster has Cloudera Runtime below 7.2.14 version.

About the back up script

The Backup and Restore script can be used only on Amazon Web Services, Microsoft Azure, and Google Cloud Platform clusters where they support cloud storage.

Scenarios for using the script

- When you upgrade the data lake cluster and want to preserve profiler data in the Cloudera Data Hub cluster.
- When you want to delete the Cloudera Data Hub cluster but preserve the profiler data.
- When you want to relaunch the profiler and access the older processed data.
- **Note:** For users using Cloudera Data Catalog on Cloudera Runtime 7.2.14 version, note the following:
  - No user action or manual intervention needed after the upgrading Cloudera Data Hub cluster to the 7.2.14 version.
  - Also, as an example use case scenario, in case a new profiler cluster is launched that contains Custom Sensitivity Profiler tags and which is deleted and relaunched later, the changes are retained and no further action is required.
  - No user action is required to backup and restore the profiler data. The changes are automatically restored.

When upgrading a Cloudera Runtime version earlier than 7.2.11 to version 7.2.11:

Go to the following locations to pick up your scripts:

Back up

```
bash /opt/cloudera/parcels/PROFILER_MANAGER/profileradmin/scripts/users/backup_db.sh
```

Restore

```
bash /opt/cloudera/parcels/PROFILER_MANAGER/profileradmin/scripts/users/restore_db.sh
```

When upgrading a version below or equal to Cloudera Runtime version 7.2.11 to 7.2.12:

Go to the following locations to pick up your scripts:

Back up

```
bash /opt/cloudera/parcels/PROFILER_MANAGER/profileradmin/scripts/users/backup_db.sh
```

Restore

```
bash /opt/cloudera/parcels/CDH/lib/profiler_manager/profileradmin/scripts/users/restore_db.sh
```

When backing up and restoring for a cluster having the Cloudera Runtime version 7.2.12 and onwards:

Navigate to the following location to pick up your scrips:

Back up

```
bash /opt/cloudera/parcels/CDH/lib/profiler_manager/profileradmin/scripts/users/backup_db.sh
```

Restore

```
bash /opt/cloudera/parcels/CDH/lib/profiler_manager/profileradmin/scripts/users/restore_db.sh
```

Running the back up script

Running the profiler Backup and Restore script has multiple phases.

About this task

First, you need to back up your profiler database and next you can restore it.

Backing up the profiler database

1. Stop the Profiler Manager and Profiler Scheduler services from the Cloudera Manager instance of the Cloudera Data Hub cluster.
2. Use SSH to connect to the node where the Profiler Manager is installed as a root user.
3. Execute the backup_db.sh script:

   Attention: Users of Cloudera Runtime below 7.2.8 version should contact Cloudera Support.

   Note:

   - If the profiler cluster has Cloudera Runtime version 7.2.11 or earlier, you run the following command:

     ```
     bash /opt/cloudera/parcels/PROFILER_MANAGER/profileradmin/scripts/
     users/backup_db.sh
     ```

   - If the profiler cluster has the Cloudera Runtime version 7.2.12 or higher you must run the following command:

     ```
     bash /opt/cloudera/parcels/CDH/lib/profiler_manager/profileradmin/
     scripts/users/backup_db.sh
     ```

4. Delete the Profiler cluster.
5. Install a new version of Profiler cluster:
   - [Scenario-1] When the data lake upgrade is successfully completed.
   - [Scenario-2] When the user decides to launch a new version of the Profiler cluster.

Restoring the profiler database

1. Stop the Profiler Manager and Profiler Scheduler services from the Cloudera Manager instance of the Cloudera Data Hub cluster.
2. Use SSH to connect to the node where Profiler Manager is installed as a root user.
3. Execute the restore_db.sh script.

   Attention: Users of Cloudera Runtime below 7.2.8 version should contact Cloudera Support.

   Note:

   - If the profiler cluster has the Cloudera Runtime version 7.2.11 or earlier, you must run the following command:

     ```
     bash /opt/cloudera/parcels/PROFILER_MANAGER/profileradmin/scripts/
     users/restore_db.sh
     ```

   - If the profiler cluster having the Cloudera Runtime version 7.2.12 or higher, you must run the following command:

     ```
     bash /opt/cloudera/parcels/CDH/lib/profiler_manager/profileradmin/
     scripts/users/restore_db.sh
     ```

4. Start the Profiler Manager and Profiler Scheduler services from Cloudera Manager.

Note: When you upgrade the data lake cluster and a new version of profiler cluster is installed, the profiler configurations that have been modified by users in the older version is replaced with new values as the following:

- Schedule
- Last Run Check
- Number of Executors
- Executor Cores
- Executor Memory (in GB)
- Driver Core
- Driver Memory (in GB)

Enable or disable profilers

By default, profilers are enabled and run every 30 minutes. If you want to disable (or re-enable) a profiler, you can do this by selecting the appropriate profiler from the Configs tab.

Procedure

1. From Profilers Configs
2. Select the profiler to proceed further.

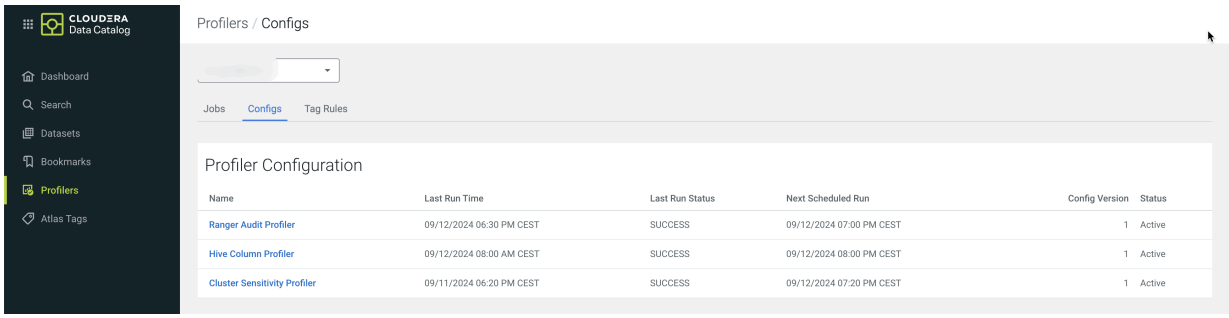

3. Switch the toggle to the desired state.

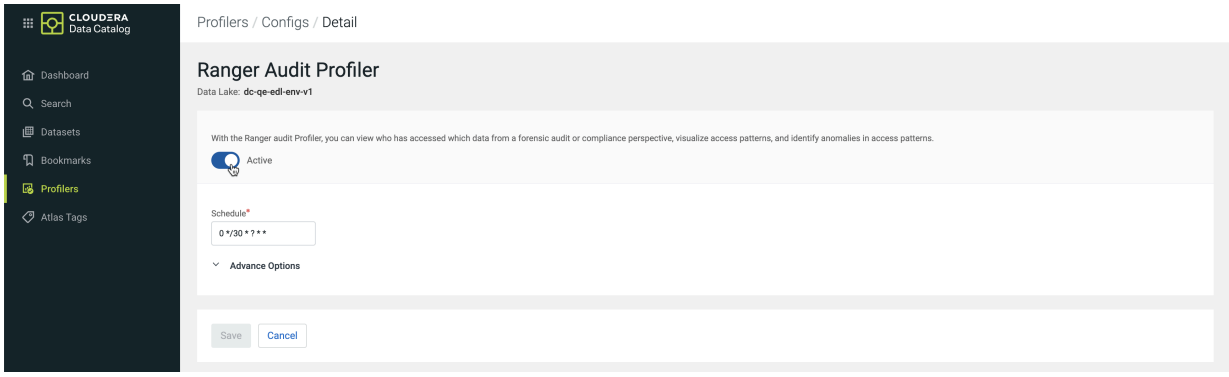

Profiler Tag Rules

You can use preconfigured tag rules or create new rules based on regular expressions to limit the number of assets to be profiled.

Rules are categorized into following groups:

- System Deployed: These are built-in rules that cannot be edited.
- Custom Deployed: Tag rules that you create and deploy on clusters after validation will appear under this category. Hover your mouse over the tag rules to deploy or suspend them as needed. You can also edit these tag rules.
- Custom Draft: You can create new tag rules and save them for later validation and deployment on clusters. Such rules appear under this category.

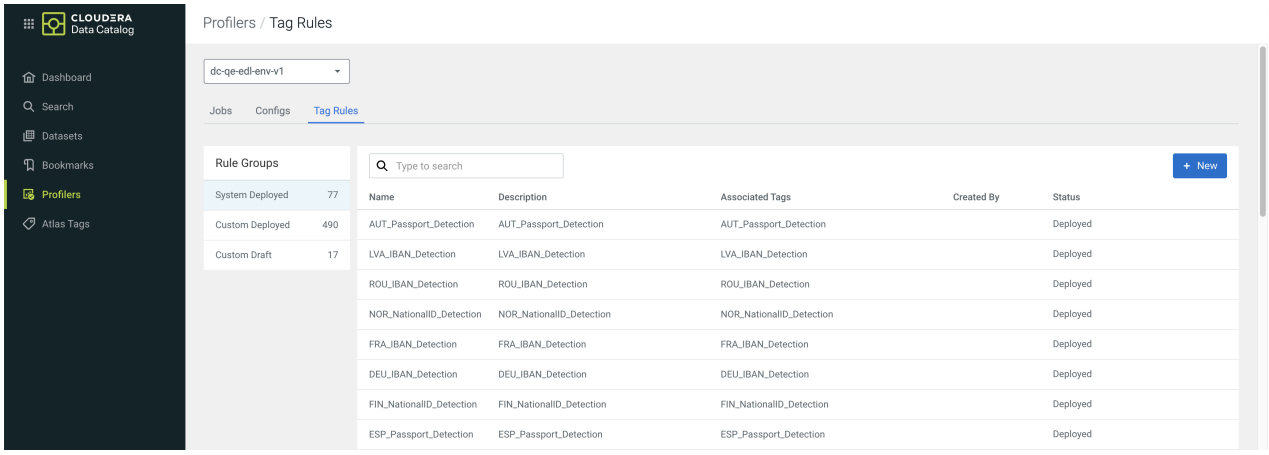

After creating your rule, you have to validate them with test data and, then Deploy them from **Custom Draft**.

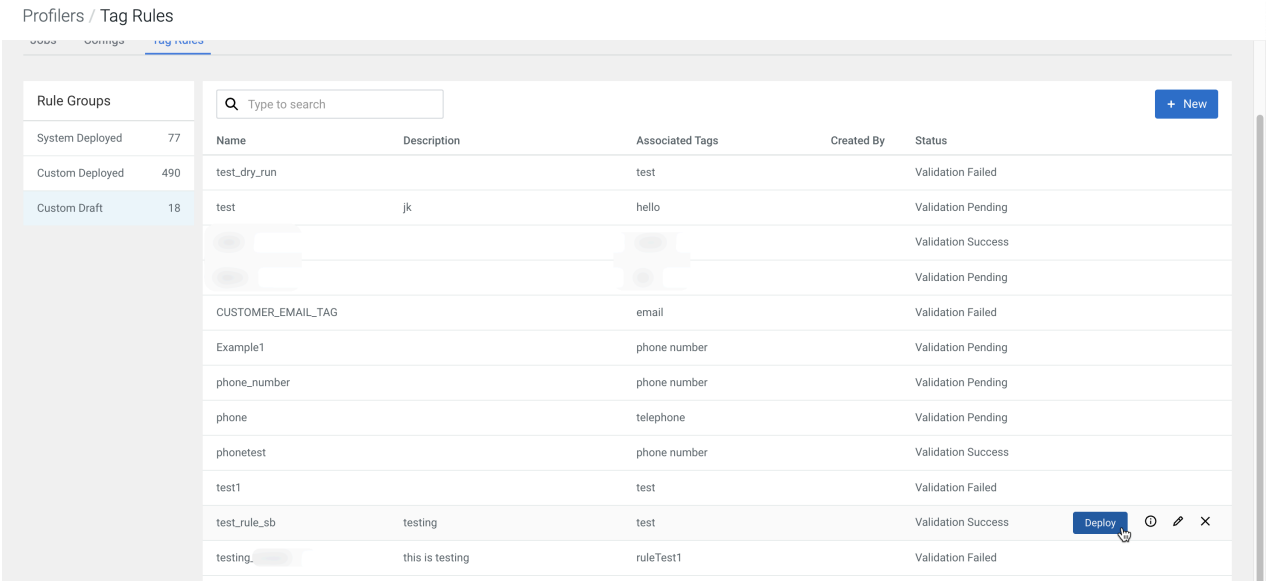

Note: Tag Rules can be temporarily suspended.

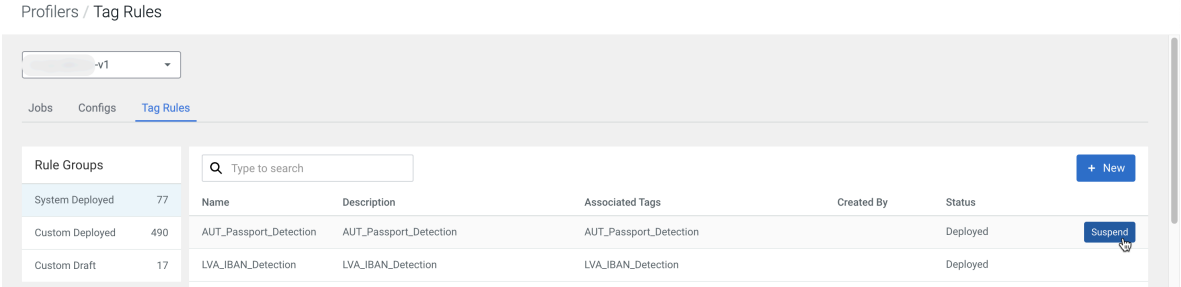

Tag management

From the Atlas Tags menu, you can create, modify, and delete any of the Apache Atlas classifications.

Atlas Tags allows the user to perform the following activities with a selected data lake for tag management:

- Selecting a data lake
- Searching for a tag
- Adding a tag
- Editing a tag
- Deleting a tag

You can create a new Cloudera Data Catalog tag in the **Atlas Tags**, which are synced to Atlas. Click Add Tag to open the **Create a new tag** page.

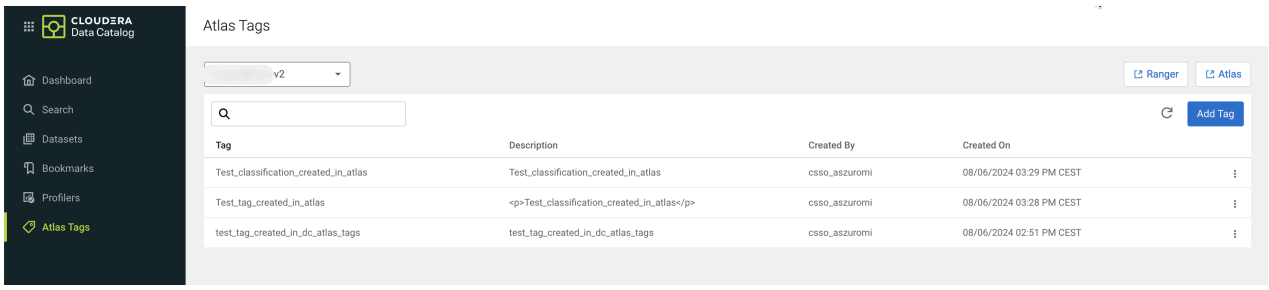

In **Create New Tag**, you can define the tag name, description and the "super-classification" from which the attributes are inherited for the sub-classification (or tag in Cloudera Data Catalog)

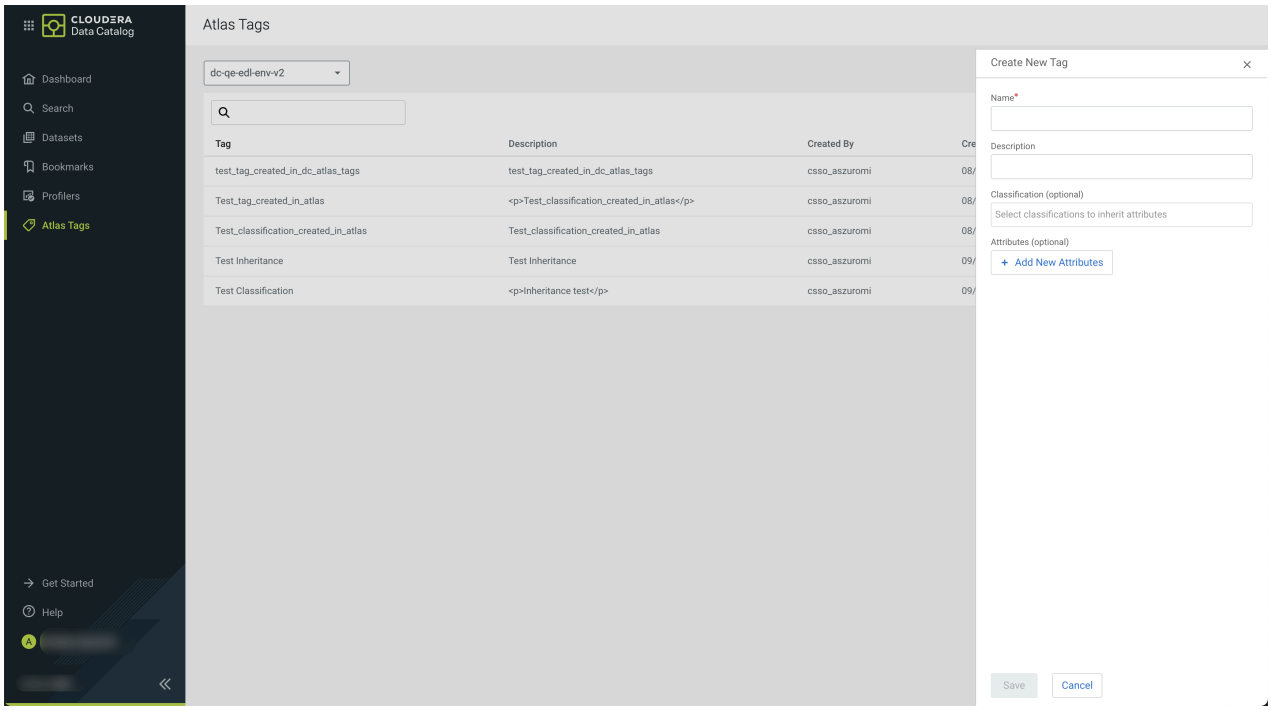

You can add or update Atlas tags. The created or updated tag is highlighted in the tag list as seen in the following diagram.

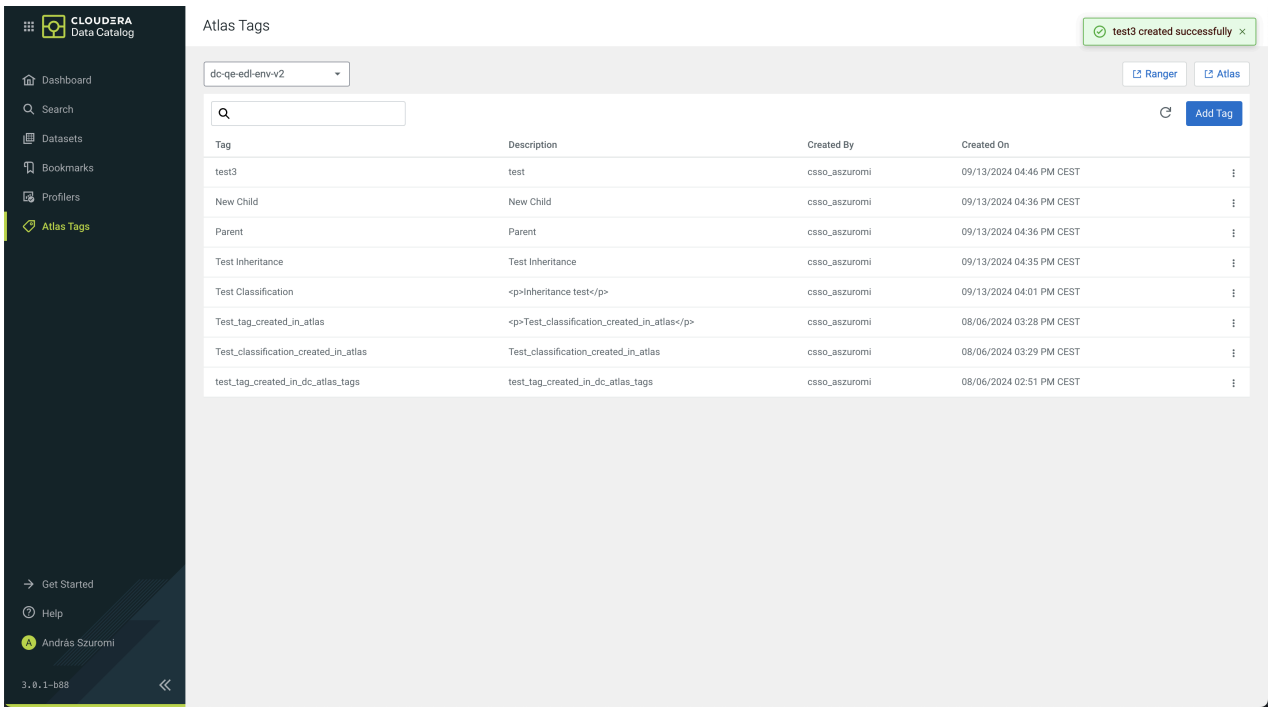

You can also edit or delete the Atlas tag as shown in the image. When you are editing the tag, you can only change the description or add new attributes.

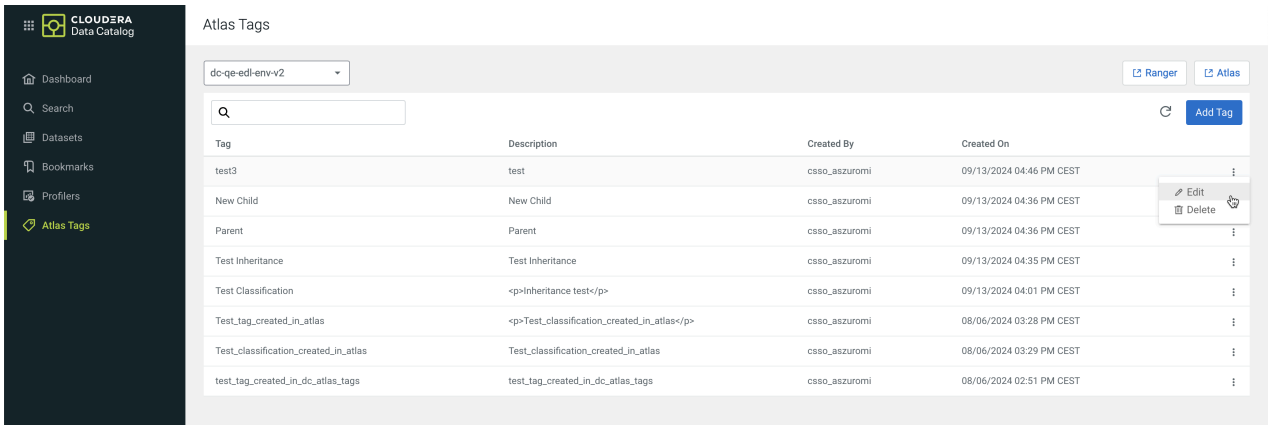

You can delete one Atlas tag at a time. A separate confirmation message appears.

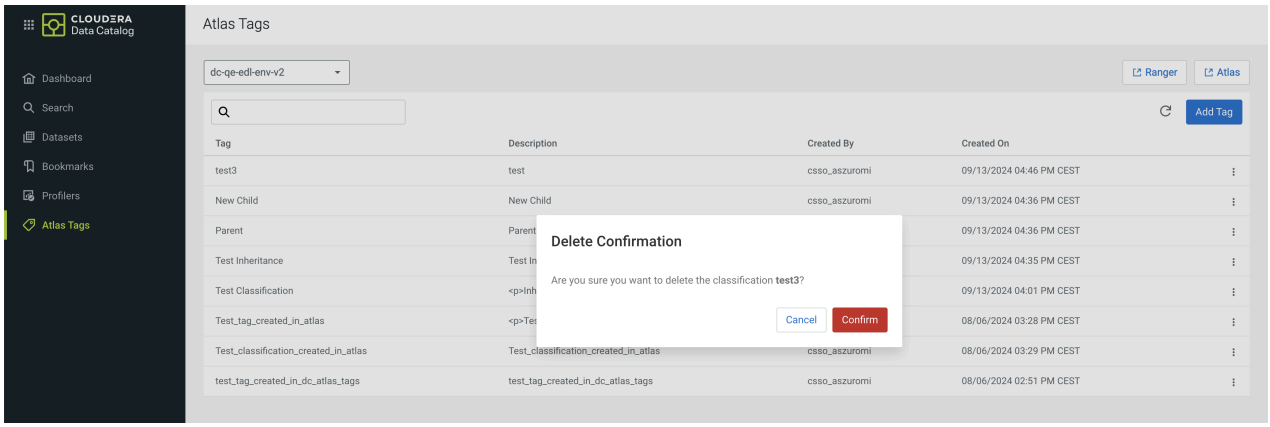

Related Information

Apache Atlas metadata attributes

Working with Atlas classifications and labels

Propagated asset tagging

Cloudera Data Catalog supports the concept of propagated tags. This feature is derived from Apache Atlas.

Whenever you add a new tag in Atlas, you can mark them as propagated and use those tags accordingly while tagging assets.

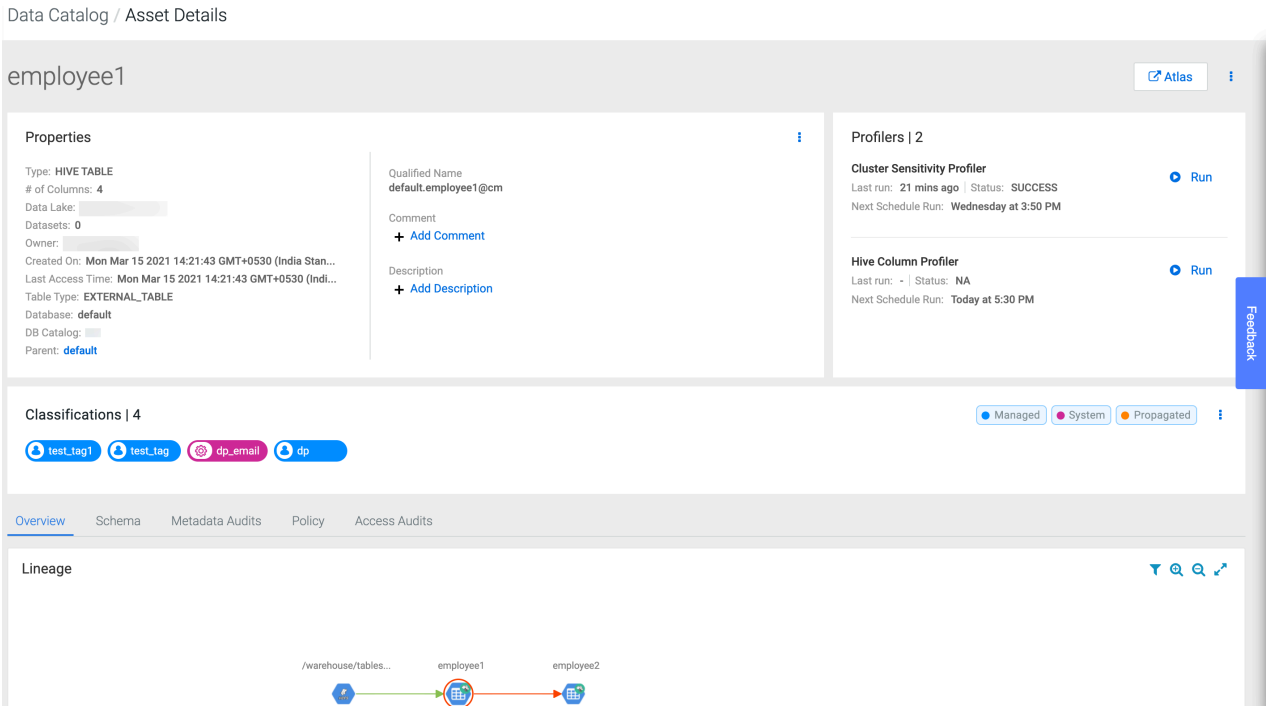

For example, consider table1 as a parent asset and table2 as a child asset. Create a tag and mark that tag as propagated, and later apply the same tag to table1. The tag gets applied to table2 as well. Propagated tag works on the basis of parent -> child tagging relationship.

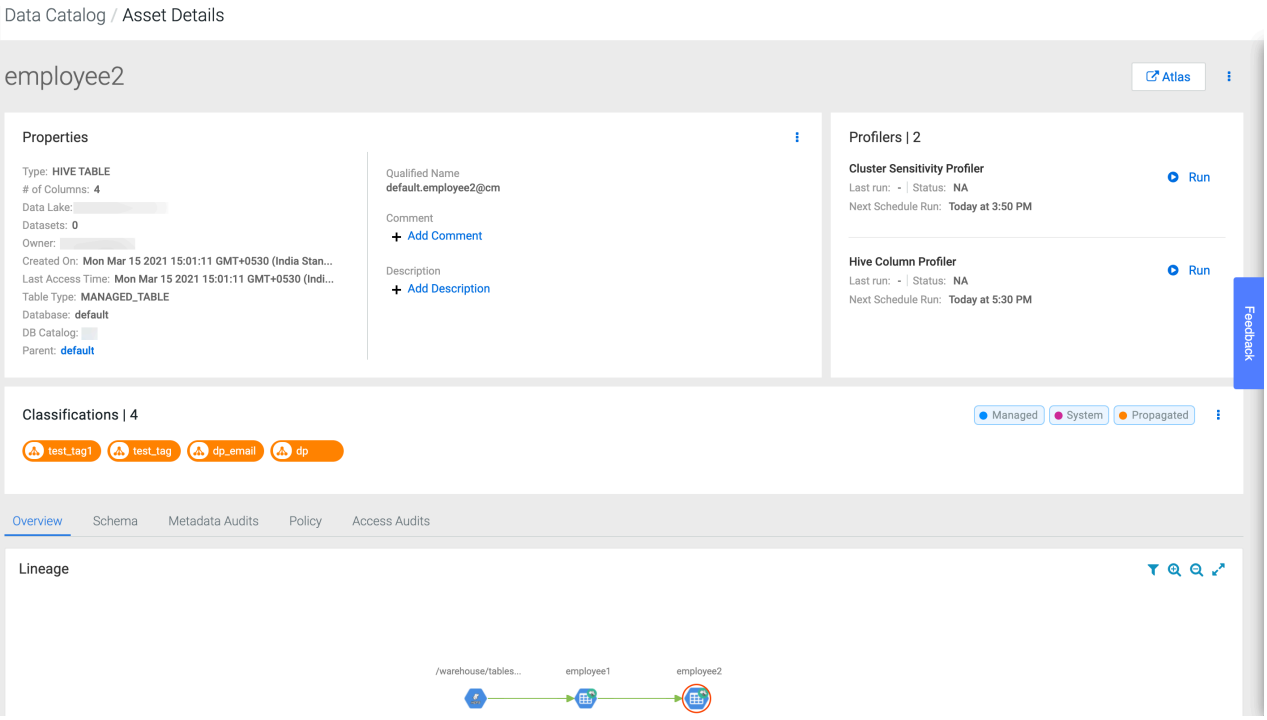

Note: When you delete or remove the propagated tag from the parent asset, the same tag is removed from all the child assets.

Attention: The propagated tag concept is not supported with child -> parent relationships.

Related Information

Propagating classifications through lineage

Creating custom profiler rules

You can create a custom profiler by adding the required tags, regex entries to specific columns within your tables. You can evaluate the tags and regex entries to Boolean values.

Procedure

1. On the **Profilers** page, click Tag Rules.
2. On the Tag Rules tab, click New to create a new profiler tag rule.
3. Enter the name of the new custom profiler tag rule.
4. Enter the description for the custom tag rule.
5. Select the tags to apply to assets which match your regex expressions. You can select tags from the drop-down list and or enter a new value to create a new tag.

   New tags that you create here are added with a dp_ prefix in the list of Atlas tags. For example, if you add a new tag called credit_card, this tag will be added as dp_credit_card in Apache Atlas.
6. Enter the rule for the column name. As you enter the values, regex name and resource names are auto populated. Select the column that is needed for your custom profiler.
7. Enter the column value for the DSL.

   Based on your entry, Cloudera Data Catalog auto-populates values from the entries already available in the **Resources** tab. For more information about behaviors, see Using DSL Grammar.

8. Click Save and Validate.

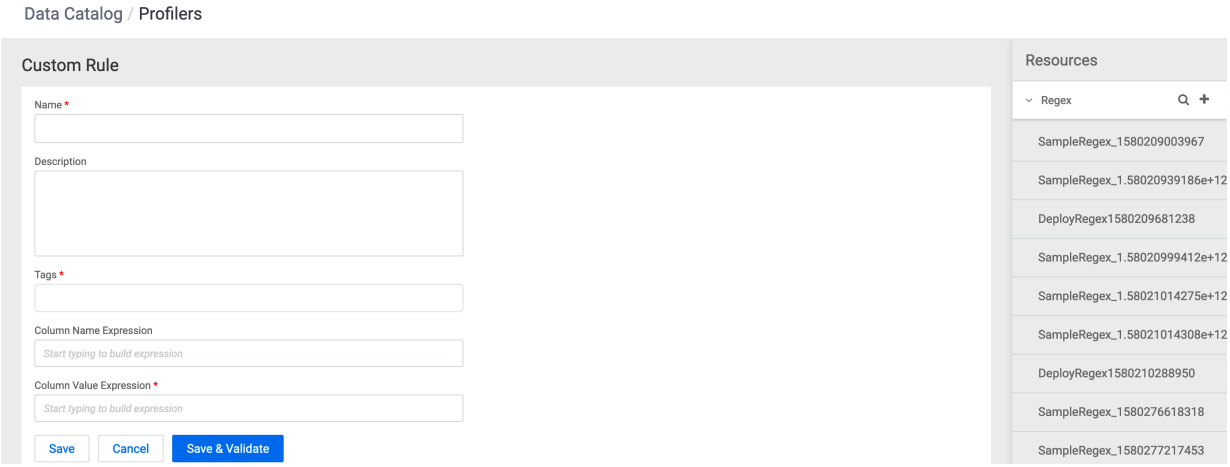

In the validation pop up window that appears, enter data to validate your custom profiler tag rule. Make sure you separate each data entry with a new line.

9. Click Save to create a tag rule and validate and deploy it later.

Adding custom regular expressions

To use custom regex entries within your new custom profiler tag rules, you can also add new regex values. If a match is found, the expression is evaluated to True.

Procedure

1. Go to Profilers Tag Rules .
2. Click New.
3. In the **Resources** in the panel on the **Custom Profiler** page, click +. The **Regular Expression Editor** page appears.
4. Enter the name of the new regular expression.

5. Enter a valid regular expression.

   For example:

```
\b(([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?))\b
```

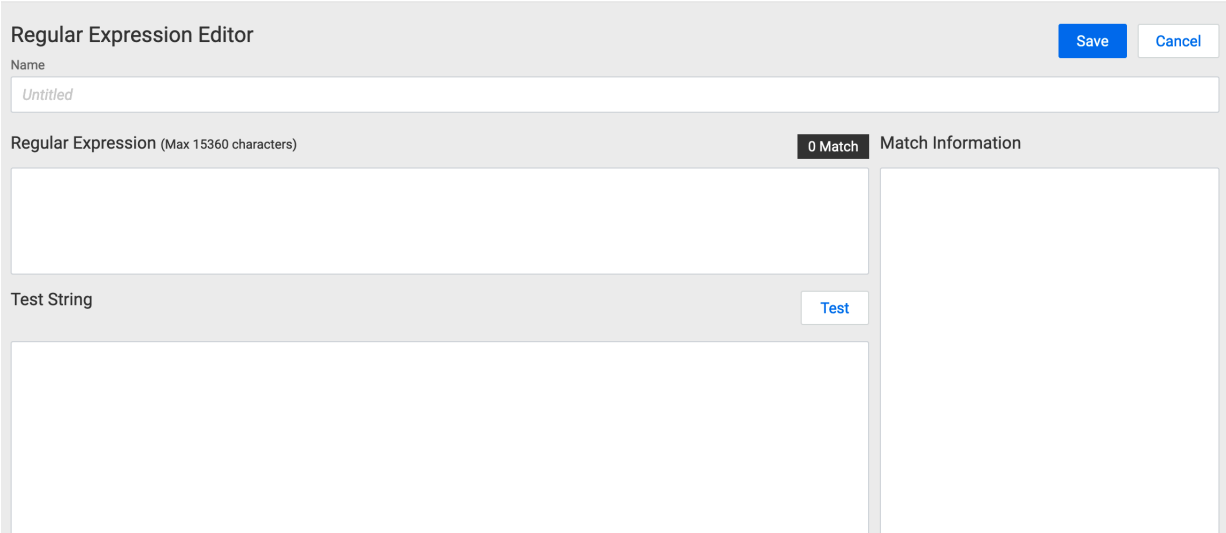

6. Enter the list of test strings to evaluate the new regular expression.

   If the test string is valid, then the match information gets auto populated in the **Match Information** box.
7. Click Save to add the new regular expression to the list of regex Resources.

Using DSL grammar

Using Domain-specific Language (DSL) grammar, you can combine different regex expressions in intuitive ways to bring out new functionality while creating custom profiler rules.

The two behaviors available in this framework are the following:

1. falseIdentity - Always evaluates to false, regardless of the input.
2. trueIdentity - Always evaluates to true, regardless of the input.

These two behaviors are used in the following examples and descriptions.

Binary AND operator

Keyword: AND

AND works the same way it does other languages. Hence following observations.

```
falseIdentity and trueIdentity == falseIdentity
```

```
falseIdentity and falseIdentity == falseIdentity
```

```
trueIdentity and trueIdentity == trueIdentity
```

```
trueIdentity and falseIdentity == falseIdentity
```

Here we are using == to show their equality.

Binary OR operator

The OR operator works the same way it does in other languages.

```
falseIdentity or trueIdentity == trueIdentity
```

```
falseIdentity or falseIdentity == falseIdentity
```

```
trueIdentity or trueIdentity == trueIdentity
```

```
trueIdentity or falseIdentity == trueIdentity
```

Expand DSL to use as follows.

```
val rule1= falseIdentity and trueIdentity and trueIdentity
```

```
val rule2= trueIdentity and trueIdentity and trueIdentity
```

```
val rule3=rule1 and rule2
```

```
rule3 or trueIdentity
```

The above expression evaluates to true.

Unary NOT operator

The NOT operator negates the value of a regex expression (from TRUE to FALSE and vice versa).

```
not(falseIdentity) == trueIdentity
```

```
not(trueIdentity) == falseIdentity
```

Use the not operator as follows.

```
val rule1= falseIdentity and trueIdentity and trueIdentity
```

```
val rule2= trueIdentity and trueIdentity and trueIdentity
```

```
val rule3=rule1 and rule2
```

```
val rule4=rule3 or trueIdentity
```

```
rule4 and not(falseIdentity)
```

The above expression evaluates to true.

CDP Public Cloud / Data Catalog

Cloudera Data Catalog Reference

Date published: 2019-11-14
Date modified: 2025-01-16

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Launching profilers using the command-line.....4

Sample Workflows.....6

Tag management and search workflow..... 6
Auto-tagging workflow with custom Cluster Sensitivity Profiler rules..... 6
Restricting access for certain users of Cloudera Data Catalog.....7

Launching profilers using the command-line

Cloudera Data Catalog supports launching profilers using the Command-Line Interface (CLI) option.

The CLI is one executable and does not have any external dependencies. You can execute some operations in the Cloudera Data Catalog service using the Cloudera CLI commands.

Users must have valid permissions to launch profilers on a data lake.

For more information about the access details, see Prerequisites to access Cloudera Data Catalog.

Prerequisites

You must have the following entitlement granted to use this feature:

DATA_CATALOG_ENABLE_API_SERVICE

For more information about the Cloudera command-line interface and setting up the same, see Cloudera CLI.

The Cloudera Data Catalog CLI

In your Cloudera CLI environment, enter the following command to get started in the CLI mode.

cdp datacatalog --help

This command provides information about the available commands in Cloudera Data Catalog for Cloudera Public Cloud 7.2.18. and earlier versions.

The output is displayed as:

```
NAME
datacatalog
DESCRIPTION
Cloudera Data Catalog Service is a web service, using this service user can
 execute operations like launching profilers in Data Catalog.
AVAILABLE SUBCOMMANDS
launch-profilers
```

You get additional information about this command by using:

cdp datacatalog launch-profilers --help

```
NAME
launch-profilers -
DESCRIPTION
Launches DataCatalog profilers in a given datalake.
```

```
SYNOPSIS
 launch-profilers
 --datalake <value>
 [--cli-input-json <value>]
 [--generate-cli-skeleton]
OPTIONS
 --datalake (string)
The CRN of the Datalake.
--cli-input-json
(string) Performs service operation based on the JSON string provided. The
JSON string follows the format provided by --generate-cli-skeleton. If other
 arguments are provided on the command line, the CLI values will override th
e JSON-provided values.
--generate-cli-skeleton
```

```
(boolean) Prints a sample input JSON to standard output. Note the specified
operation is not run if this argument is specified. The sample input can be
used as an argument for --cli-input-json.
```

```
OUTPUT
.
datahubCluster -> (object)
 Information about a cluster.
clusterName -> (string)
 The name of the cluster.
 crn -> (string)
 The CRN of the cluster.
 creationDate -> (datetime)
 The date when the cluster was created.
 clusterStatus -> (string)
 The status of the cluster.
 nodeCount -> (integer)
 The cluster node count.
 workloadType -> (string)
 The workload type for the cluster. cloudPlatform -> (string) The cloud plat
form.
 imageDetails -> (object)
The details of the image used for cluster instances.
 name -> (string)
The name of the image used for cluster instances.
 id -> (string)
The ID of the image used for cluster instances.
This is internally generated by the cloud provider to Uniquely identify the
 image.
 catalogUrl -> (string)
 The image catalog URL.
 catalogName -> (string)
 The image catalog name.
 environmentCrn -> (string)
 The CRN of the environment.
 credentialCrn -> (string)
 The CRN of the credential.
 datalakeCrn -> (string)
 The CRN of the attached datalake.
 clusterTemplateCrn -> (string)
The CRN of the cluster template used for the cluster creation.
```

Launching the profiler

You can use the following CLI command to launch the data profiler:

```
cdp datacatalog launch-profilers --datalake [***DATALAKE CRN***]
```

Example:

```
cdp datacatalog launch-profilers --datalake crn:cdp:data
lake:DATACENTERNAME:c*****b-ccce-4**d-a**1-8********8:datalake:4*****5e-c**
1-4**2-8**e-1********2
{
    "success": true
}
```

Related Information

Cloudera CLI 0.9.124 documentation / datacatalog command

Sample Workflows

When you work with Cloudera Data Catalog, you can use the workflows described in this section.

Tag management and search workflow

You can create a tag, assign the tag to an asset, and search for an asset using various filters.

About this task

You can create a tag and search for any asset using the tag. You can also apply a tag to a column. Later, you can search assets based on the Entity Tag and Column Tag filters.

Procedure

1. Go to **Atlas Tags** and click Add Tag
2. Enter the name, description for the tag.
3. Inserting a classification to inherit attributes and adding new attributes are optional.
4. Click Save.
   The new tag is created.

   Next, you must assign the newly created tag to one of the existing assets.

   For example, we can assign the tag to the Hive table.
5. Go to **Search** and search for an asset with columns, such as a Hive table, then click on the asset to view the **Asset Details** page.
6. Click +Add Classification, to display the **Classifications** view.
7. Select the tag that you newly created by typing in its name and assign the tag to the asset.
8. Click Save.

   Next, you can add the tag to a column in the selected asset.
9. Return to the previously selected asset: Go to **Search** and search for an asset with columns, such as a Hive table, then click on the asset to view the **Asset Details** page.
10. Select the **Schema** tab and click Edit, then select Classifications.
11. Click + and select the tag by entering its name.

    Note: The tag list contains System Tags and Managed Tags. Managed Tags are selected by default.

    - System Tags are applied by the tag rules in the **System Deployed** section. For example, IBAN, passport number detection rules.
    - Managed Tags are applied by the tag rules in the **Custom Deployed** section. These are user created tag rules. However, custom tag rules can also apply system tags.
12. Select a tag with the + icon and click Save.

Once tags are added to the table and column, you can search for the corresponding assets using the tag name.

13. Go to **Search** and enter the tag name in the Entity Tag or the Column tag filter.
    The asset for which the tag was added is displayed as the search result.

Auto-tagging workflow with custom Cluster Sensitivity Profiler rules

You can auto-tag workflows while working with Cluster Sensitivity Profiler.

About this task

Use the following information to create a custom tag and assign the same to the Cluster Sensitivity Profiler.

Procedure

1. Go to Profilers Tag Rules .
2. Click + New to open the **Custom Rule** window.
3. Under the **Resources** pane, click + to open the **Regular Expression Editor** window.
4. Enter the name and input the regular expression value in such a way that it matches your test string.
   If your regular expression value is [a-z][a-z][a-z][a-z] and the test string is “baby”, there is a match.
5. Click Save.
6. On the Custom Rule window, enter the name and description.
7. Enter the tags value and select the Column Expression Name from the drop-down.

   You must select the same regular expression you had created under the Resources pane.
8. Enter a value for the **Tags** field and select the **Column Value Expression** from the drop-down.

   You must select the same regular expression you had created under the Resources pane.
9. Click Save & Validate.
   The Data For Validation window appears.
10. Enter the sample values to validate if **Column Name Expression** and **Column Value Expression** entities match.
    Make sure that the correct data lake is selected to validate the entries.
11. Click Submit Validation.
    The status for the newly created regular expression validation is displayed on the Tags Rule tab. Once the validation is successful, you can deploy the rule.
12. Click Done.
    On the Rule Groups pane, verify if the rule is available under the Custom Deployed list. You can also suspend the rule by selecting the it from the list.

    Once the Cluster Sensitivity Profiler job or On-Demand Profiler picks up the Hive asset for profiling, the newly set up custom tag gets applied on the Hive column, provided the asset has the column(s) which meet the custom rule criteria.

Restricting access for certain users of Cloudera Data Catalog

To have a fine-grained access to the same user from accessing the assets in Cloudera Data Catalog, you can perform some additional changes. For example, if you want to restrict some users from accessing specific table information, you must set-up a Ranger policy such that these users will not have access to the asset details.

Note: When a subscriber is provided with user access either as EnvironmentAdmin or EnvironmentUser, by default, the user is synchronized with the default Atlas policies in Ranger. This implies that the same user has access to all the datasets in the environment.

To create the Ranger policy to restrict users from accessing asset details, for example, with a specific classification, refer to the following images:

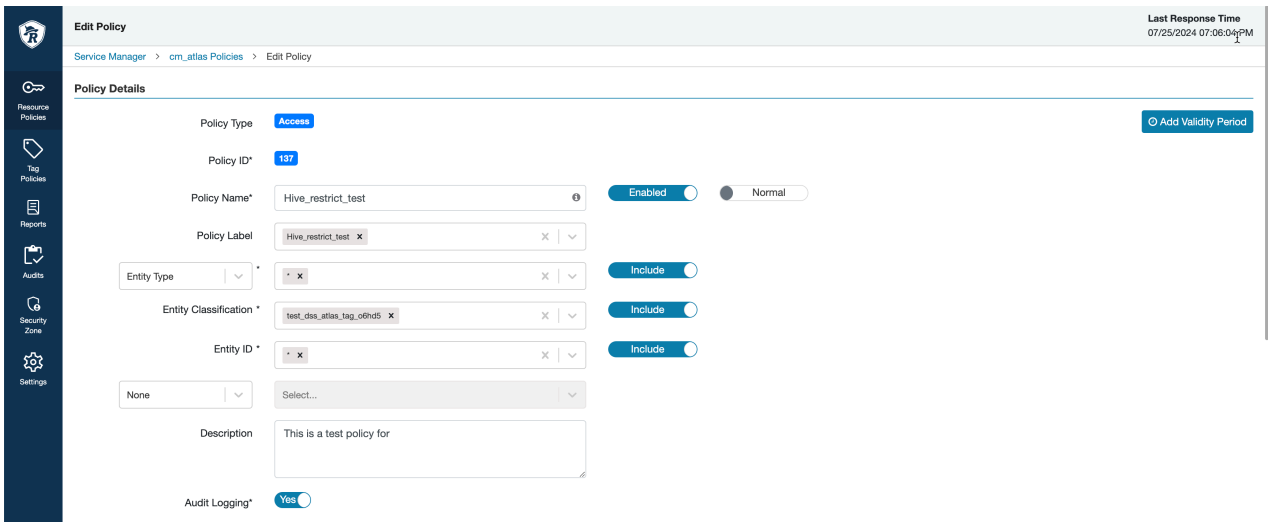

The following image displays the **Deny Conditions** set for the specific user.

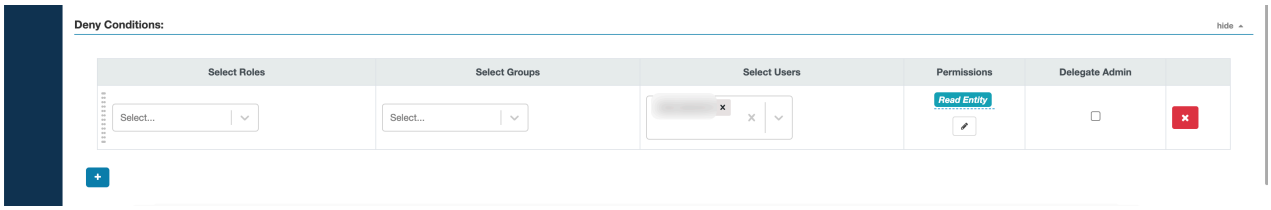

The result is depicted in the following image, where the user has no permissions to access the specified dataset.

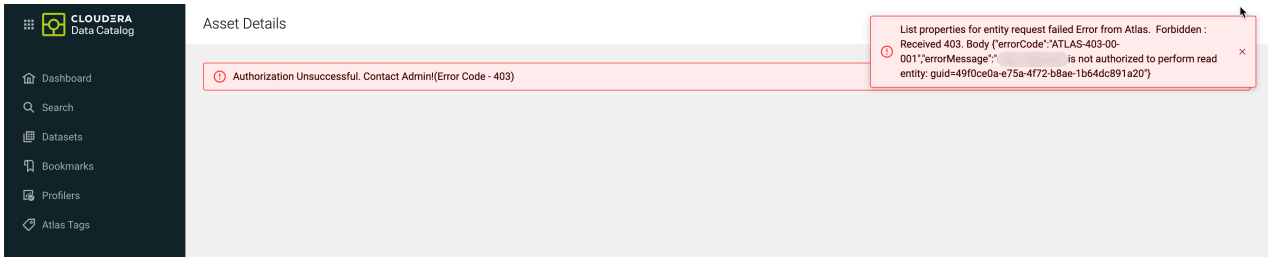

Reducing resource consumption with restricted users

Additionally, when you plan to restrict data access, please note the following:

- Audit summarization for the asset evolves from the Ranger Audit Profiler and Metrics service.
- Various Hive Column Statistical metrics for columns of the asset evolves from Atlas as part of the profile_data of a column.

To ensure that the data related to audit summary and Hive Column Statistics are not visible to the subscribers, you must make sure to turn off the audit profiler and the Hive Column Profiler respectively.

CDP Public Cloud / Data Catalog

Cloudera Data Catalog Release Notes

Date published: 2019-11-14
Date modified: 2025-01-16

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What's new 4
January 16, 2025 4
December 18, 2024 5
June 03, 2024 5
May 16, 2024 5
April 3, 2024 6

Older releases 6
Cloudera Data Catalog releases for 2023 6
December 15, 2023 6
March 13, 2023 7
January 17, 2023 7
Cloudera Data Catalog releases for 2022 7
April 1, 2022 7
March 2, 2022 7
February 16, 2022 8
Cloudera Data Catalog releases for 2021 8
December 9, 2021 8
October 26, 2021 8
August 27, 2021 8
October 5, 2021 8
July 26, 2021 9
May 31, 2021 9
April 5, 2021 9
March 25, 2021 9
March 15, 2021 9
Cloudera Data Catalog releases for 2020 10
December 14, 2020 10
October 29, 2020 10
August 27, 2020 10
August 12, 2020 10
July 25, 2020 11
June 30, 2020 11
June 17, 2020 11
June 5, 2020 11
May 6, 2020 11
April 7, 2020 12
January 6, 2020 12
Cloudera Data Catalog releases for 2019 12
November 14, 2019 12
September 23, 2019 12
August 22, 2019 13

Known issues 13

Fixed issues 15

What's new

This section lists major features and updates for the Cloudera Data Catalog service.

January 16, 2025

This release (2025-M1) of the Cloudera Data Catalog service introduces the following new changes:

Containerized architecture for profilers

Cloudera Data Catalog introduces a new containerized architecture for Profilers for Compute Cluster enabled environments, providing a scalable environment:

- Only the required amount of Kubernetes pods are launched based on the size of the database to be profiled. You need to pay only for the used cloud resources only while they are used by the profilers.
- Also, the deployment of the containerized profiler architecture is more streamlined and quicker than the previous VM-based architecture.
- Moreover, the containerized nature of the architecture means that later upgrades can be carried out easier, without the need for multiple dependencies as in the VM-based architecture which used multiple services.
- Profilers now also support the following file formats:
  - VM-based environments: CSV, ORC
  - Compute Cluster enabled environments: CSV and Parquet
    - Hive Column Profilers and Cluster Sensitivity Profilers also support profiling Iceberg Tables, including with On-Demand Profilers.

Important:

- Currently, Kubernetes based profilers are only supported in AWS environments.
- In Compute Cluster enabled environments, profilers only support tables which are stored on AWS S3 storage.

For more information, see Profiler architecture in Compute Cluster enabled environment.

Note:

The VM-based architecture (using the Cloudera Data Hub Cluster) is deprecated from this release but remains available until Cloudera Public Cloud 7.2.18 is supported (Sept 2025). Therefore, Cloudera Data Hub based profilers will also not be available in Cloudera Public Cloud versions after 7.2.18. Only Compute Cluster enabled environment will be able to run Cloudera Data Catalog profilers after version 7.2.18.

For more information, see Cloudera Support lifecycle policy.

Redesigned Dashboard menu

- A new **Dashboard** is introduced to give a overview of your data lakes and profilers including:
  - Data lake type and status
  - Profiler status
  - Last 10 assets bookmarked by you
  - Last run of profiler
  - Number of assets profiled

Redesigned Search menu

The **Search** menu is reorganized so information is easier to access. You can expand each entity result to see their qualified name, database, classification and assigned terms. You can use these to check if your query returns the expected results.

Improved display of comments in Asset Details

Following this release, you can hover over the **Comment** field for individual schema entries in **Asset Details** to preview longer comments without opening them.

Common time format

Asset Details and other menus will used the same time format for a more readable overview: MM/DD/YYYY hh:mm A.

Removed features

The following features have been removed:

- Tag Rules Luhn check algorithm
- Tag Rules File-based Allow and Deny list
- Tag Rules Lookup files
- Tagging multiple assets in the **Search** menu

December 18, 2024

This release (2.0.28) of the Cloudera Data Catalog service introduces the following new changes:

Column name based tagging in Cluster Sensitivity Profiler

You can override the sampling to profile data in a column based on the column name matching a preset regular expression pattern instead matching the certain percentage of the columns values. This can be used for assets with skewed proportions where relying on the sampling would not result in correct tagging.

For more information, see Cluster Sensitivity Profiler configuration and Setting up column name based tagging.

June 03, 2024

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

This release of the Cloudera Data Catalog service introduces the following new changes:

This release only contains fixes and updates to prepare Cloudera Data Catalog for the changes in the upcoming 3.0.0 release.

May 16, 2024

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

This release of the Cloudera Data Catalog service introduces the following new changes:

Iceberg tables are now supported by the Cloudera Data Catalog service:

- You are able to filter for them in the Search page.
- Iceberg tables can be viewed in the Asset Details page.
- Iceberg tables can be added to a dataset.
- All subcomponents of Cloudera Data Catalog support JDK 17.
- **Important:** Profilers do not support Iceberg tables in this release.

April 3, 2024

This release of the Cloudera Data Catalog service provides you with a notable behavior change which you must note and act accordingly.

While upgrading your cluster from Cloudera Runtime version **7.2.17** to **7.2.18**, and specifically during the OS upgrade step, the cluster goes into the failure state. The following message is seen:

__NODE_FAILURE:

New node(s) could not be added to the cluster. Reason Please find more details on Cloudera Manager UI. Failed co mmand(s): Start(id=1546339088): Failed to start role profc6cf3856-PROFILER_SCHEDULER_AGENT-484032cb8 f17cacf9e684efe50 of service profiler_scheduler in cluster cdp-dc-profilers-258395ef._

Impact on Cloudera Data Catalog profilers:

If the Cloudera Data Hub is not created, then the Cloudera Data Catalog profilers will not be created in Cloudera Runtime 7.2.18 version.

To overcome this scenario, you must use the following process to bring up the Cloudera Data Catalog profilers in the Cloudera Runtime 7.2.18 version.

First you must delete your existing 7.2.17 clusters. For more information, see Deleting profiler cluster.

Next, after you upgrade to the 7.2.18 Data Lake, then you can launch the Cloudera Data Catalog profilers. For more information, see Launch profiler cluster.

Note: There is no data loss expected on the users' side or the Profiler analysis. However, the only loss that could be expected is related to the last runtime value of the profiler and the profiler run history. The Profiler Last Runtime history refers to the records of how many runs of the profiler are displayed on the history page. It includes information on whether the runs were completed successfully or resulted in failures.

Older releases

Overview of new features, enhancements, and changed behavior introduced in earlier releases of Cloudera Data Catalog.

Cloudera Data Catalog releases for 2023

Release summaries:

December 15, 2023

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following features as generally available:

Cloudera Data Catalog is now integrated with AWS Glue Data Catalog which enables the users to browse and discover data as well as register data into Cloudera Shared Data Experience (using metadata translation or copy), so that it can be used with Cloudera Data Hub and other relevant experiences.

March 13, 2023

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following features as generally available:

- Supporting High Availability for profiler services.
- Navigating to tables and databases in Hue using Cloudera Data Catalog.
- Using the Download CSV option to save queried results.

January 17, 2023

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Cloudera Cloudera Data Catalog is now additionally supported in the EU and APAC regional Control Plane.
- Associating Customer Sensitivity Profiler tags to Hive assets fails on the 7.2.16 Cloudera Runtime cluster. You must update the configurations. For more information, see Associating Custom Sensitivity Profiler tags to Hive assets.

Cloudera Data Catalog releases for 2022

Release summaries:

April 1, 2022

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

Profiler cluster launch requires specific instance type of the Master node.

March 2, 2022

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Support for Cloudera Runtime 7.2.14
- No end-user manual process needed for backing up and restoring profiler data
- Post-upgrade (to Cloudera Runtime 7.2.14 version), deleting profiler cluster does not require saving the modified Custom Sensitivity Profiler rules

February 16, 2022

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Deleting profiler cluster with multiple scenarios
- Bug fixes

Cloudera Data Catalog releases for 2021

Release summaries:

December 9, 2021

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Deleting profiler cluster
- Bug fixes

October 26, 2021

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following addition:

Available location paths for the Backup and Restore scripts based on the Cloudera Runtime version.

August 27, 2021

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following addition:

- Bug fixes. Refer to the Fixed Issues section in the Release Notes document.
- Refreshed screenshots for profiler configuration and related pages.
- Profiler cluster creation using the new UI
- The option selected in Depth drop-down and Show Process nodes are now displayed on the upper-left corner of the Lineage module.

October 5, 2021

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following addition:

- Support for deleting the profiler cluster
- Updated asset details
- Bug fixes and refreshed UI

July 26, 2021

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following addition:

- Creating classifications on the search page or on the classification widget.
- Bug fixes. Refer to the Fixed Issues section in the Release Notes document.
- Updated in-help documentation for multiple components in the UI.

May 31, 2021

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following addition:

- Launching Profilers using the Command-line Interface (CLI).
- Bug fixes. Refer to the Fixed Issues section in the Release Notes document.

April 5, 2021

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following addition:

Support for Business Glossary and usage of "terms" to associate datasets.

March 25, 2021

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following addition:

Backing up and Restoring Profiler Database.

March 15, 2021

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Searching for asset types using additional options in Cloudera Data Catalog and Apache Atlas
- Bug fixes

Cloudera Data Catalog releases for 2020

Release summaries:

December 14, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Support for AWS and Azure in Cloudera Runtime 7.2.6 version
- Atlas entity audits in the **Asset Details** page
- Profiler jobs tracking status
- Audit graphs for HBase tables

October 29, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Support for searching assets across multiple data lakes.
- A new tab called Contents is introduced to list the contents of the select entity.
- Support for tagging multiple assets.

August 27, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Multiple entity type selections to search for assets
- Other UI fixes and improvements

August 12, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Multiple filter type selections on the **Search** page
- Other UI fixes and improvements

July 25, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Support for improved user experience for the **Search** page
- Atlas tags option is now directly accessible from the menu bar

June 30, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following additions:

- Tag management
- On-Demand profiler support for Hive Column Statistics Profiler and Cluster Sensitivity Profiler
- Last Run Check Configuration
- Support for adding Hive asset to one or more datasets from the **Asset Details** screen

June 17, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following changes and bug fixes:

- UI improvements for search capabilities
- Ability to search Hive and HBase assets using Column tag

June 5, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following changes:

- About accessing Data Lakes
- Support for prepopulating the Asset owners
- Support for selecting more entity types for searching assets

May 6, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following changes and bug fixes:

Navigation Support for Hive entity within Lineage.

April 7, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following new features:

- Profiler Tag Rules
- Hive Column Statistical Profiler

January 6, 2020

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments.

Cloudera Data Catalog introduces the following new features:

- Ability to run Sensitivity Profiler jobs
- Ability to collaborate with other users - you can now rate datasets, add comments about asset collections and respond to the comments of other users.

Cloudera Data Catalog releases for 2019

Release summaries:

November 14, 2019

This release of the Cloudera Data Catalog service introduces the following new features.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments (on-premises, cloud, and IOT).

Cloudera Data Catalog introduces the following new features:

- Ability to view Profiler jobs
- Ability to edit Profiler configuration
- Ability to toggle a Profiler on and off.

September 23, 2019

This release of the Cloudera Data Catalog service introduces the following new features:

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments (on-premises, cloud, and IOT).

Cloudera Data Catalog enables data stewards across the enterprise to work with data assets in the following ways:

- Organize and curate data globally

- Understand where relevant data is located
- Understand how data is interpreted for use
- Understand how data is created and modified
- Understand how data access is secured and protected, and audit use

August 22, 2019

This release of the Cloudera Data Catalog service introduces the following new features:

This is the first release of the Cloudera Data Catalog service.

Cloudera Data Catalog is a service within Cloudera that enables you to understand, manage, secure, and govern data assets across enterprise data lakes. Cloudera Data Catalog helps you understand data across multiple clusters and across multiple environments (on-premises, cloud, and IOT).

Cloudera Data Catalog enables data stewards across the enterprise to work with data assets in the following ways:

- Organize and curate data globally
- Understand where relevant data is located
- Understand how data is interpreted for use
- Understand how data is created and modified
- Understand how data access is secured and protected, and audit use

Known issues

Learn about the known issues in Cloudera Data Catalog, the impact or changes to the functionality, and the workaround.

CDPDSS-1953: Location field does not load absolute path for entities in contents tab

Problem:After navigating to the Azure Blob which is created within the Azure directory and later clicking the "Content" tab, the path for Azure blob is displayed as "/" instead of the path that mentions: containers/directory/Blob.

CDPDSS-1956: Issue with count on search page as AWS S3 V2 Object

Problem: The count against the Data Lake displays only the count of AWS S3 V2 Object entity type.

CDPDSS-31173: Atlas API on saving the classifications in Cloudera Data Catalog is failing while using Cloudera Runtime version 7.2.12

When the suggested classification by Custom Sensitivity Profiler is saved, Atlas API throws an error response and it is not saved in Cloudera Data Catalog. This behavior is specifically observed in Cloudera Runtime version 7.2.12.

Workaround: If you are using 7.2.12 version, from Cloudera Manager > Atlas > Configuration > search for 'application.prop' > enter the value as "atlas.tasks.enabled=false". Restart Atlas. Later, log into Cloudera Data Catalog to use the classification feature.

CDPDSS-2127: On profiler cluster creation step 'external database in creation' the message on search page is not displayed

The message on Cloudera Data Catalog **Search** page is not displayed in 'blue' section saying "profilers cluster is getting created" .

CDPDSS-2138: ODP fails on an asset imported from Glue to Cloudera on a medium duty cluster

The profiler run fails on the table.

CDPDSS-2142: Livy job in Profiler run has failed but profiling status of assets is shown as started

Problem: The Livy job fails but each assets shows profiling status as started.

CDPD-32000: Atlas intermittently does not send the classification information to Cloudera Data Catalog

On loading of search page filters with 'entity type' filter applied to 'Hive', intermittently the tags/classifications are not loaded in Cloudera Data Catalog App. Due to this, Cloudera Data Catalog does not load the filters for 'entity tag' and column tag.

CDPDSS-2145: Applying column tag filter and typing the same search result in search field gives ‘no results’ found

Problem: On searching the asset which is in the results for a column tag filter, the search results are shown as 'No results found'.

CDPDSS-2236: Issues noticed on data-catalog while performing scale testing

Problem:

- Search page consistently throws 504 gateway time-out error, however on applying global search field or search filters, results are loaded.
- Hive Column profiler goes to “undetermined” state for both ODP or scheduled jobs . To fix this, Profiler must be stopped and restarted. Happens due to Profiler not completing the read events from eventstore in AWS S3 within the time-out limit.(intermittent)
- On **Asset Details** page, when ODP is triggered, on successful completion of jobs and refresh of page, the ODP widget is not displayed and user needs to refresh a couple of times to load it on UI (reproducible). This ODP trigger job is running in parallel to running scheduled jobs.
- Assets cannot be added to dataset as the API fails with gateway timeout (intermittent)

Workaround: For the first and fourth issues listed, the following configuration change done on Cloudera Manager for Atlas:

atlas.graph.index.search.types.max-query-str-length=1536

Heap size = 4 GB

For the third issue listed, you must refresh the **Asset Details** page for about three to four times.

CDPDSS-2488: In MGMT service of DH CM, the telemetry command is failing bringing the master node down

When a Cloudera Data Hub cluster is created, the management service of Cloudera Manager becomes unavailable because the telemetry command fails on Hive. Because of this, the Master node of Cloudera Data Hub goes down causing a Node Failure error on the cluster.

In Cloudera Data Catalog, due to the failure of the master node of Cloudera Data Hub, the **Profiler** and the **On-Demand Profiler** page in the **Asset Details** page does not load.

It is recommended to upgrade to Cloudera Data Catalog3.0.0 to avoid this issue.

CDPDSS-3364: Not able to delete the profilers from DSS App when underlying compute cluster deleted

When deleting the compute cluster in Compute Cluster enabled environments profilers cannot be deleted. The following error is displayed: Failed to delete profilers with error 401.

CDPDSS-3403: Altered ICEBERG table is not available in Compute Cluster enable environment

Iceberg tables modified or renamed with the ALTER operation appears with both old and new name, however, neither entry can be accessed. Apache Atlas synchronizes the table with the altered name with its full qualified name instead of its name shown in the **Search** results.

CDPDSS-3412: Profiling is not working on migrated ICEBERG table

Hive tables migrated to Iceberg with the ALTER TABLE statement cannot be profiled in Compute Cluster enabled environments. Cluster Sensitivity Profilers and Hive Column Profilers will not be able to apply tags to the values.

CDPDSS-3401: Profiler is shown as running in DSS App on deleting default compute cluster

In a Compute Cluster enabled environment, after deleting the default Compute Cluster profiler are displayed as still running in the **Dashboard**. However, profiler jobs will fail even if the default Compute Cluster is recreated.

After recreating the Compute Cluster, delete and relaunch the profilers.

CDPDSS-3396: Occasionally getting Bad Gateway(Error Code - 502) error in DC UI

In a Compute Cluster enabled environment, occasionally the messages Error Occurred while processing: Bad Gateway are displayed in addition to the **Asset Details Overview** tab failing to display any data.

The error is corrected by Cloudera Data Catalog in a few seconds.

CDPDSS-3323: Profiler Jobs are running forever if the primary Kubernetes pod is shut down

If the primary Kubernetes pod is forced shut down, or shuts down because of an error, the Kubernetes scheduler requests new resources for new pods. However, the original pod shut down still appears as a running job in Profiler / Jobs. This job cannot be deleted, as it is falsely marked running.

CDPDSS-3467: [Search Page] Incorrect results are being displayed after navigation

The total number of results in the **Search** page may change when changing the number of results displayed for a page after navigating into **Asset Details** for a result and navigating back to the overview of results. This is due to an incorrect response from Apache Atlas. The total number of results in the **Search** page may also change when using the owner filter and then clearing the applied filters, displaying all available assets.

CDPDSS-3387: Profiling of struct or composite data type are not supported in Cloudera Data Catalog

Profiling of struct or composite data type are not supported in Cloudera Data Catalog. When profiling tables with such data. the profiling job will be stuck in the Undetermined status.

CDPDSS-3561: The number of columns is displayed incorrectly when the DB or table name is too long

In **Asset Details**, the number of columns (**# of Columns**) is displayed as zero when the database or table name is too long.

CDPDSS-3569: [Search Page] Filters are retained when a filter is selected and then a different data lake is selected

When switching data lakes on the **Search** page, the previously selected search filters are not cleared. This can result in an error message.

After switching data lakes, remove the applied filters by clicking Clear All.

Fixed issues

This section lists the issues that have been fixed since the previous version.

CDPDSS-81: Cloudera Data Catalog Issue

Problem: If an error occurs while fetching results, an error message appears briefly and disappears. Once the message disappears, the results pane is stuck in the Loading state.

CDPDSS-81: Cloudera Data Catalog Issue

Problem: Despite having no privileges to add, modify, or delete classifications, users are able to edit the tag flow on the **Asset Details** page. However, users will not be able to save the changes.

CDPDSS-81: Cloudera Data Catalog Issue

Problem: On the **Asset Details** page, the Audit tab fails to load if the logged in user does not have Audit read privilege.

CDPDSS-363: The 403 error message seen in Cloudera Data Catalog > Table / Audit tab

Problem: Error messaging needs to be improved for users accessing Table and Audit tab without sufficient Ranger role.

CDPDSS-437: Unauthorized message for tag management flow does not have column name

Problem: Unauthorized column names and comma visible for tag management flow.

CDPDSS-444: Duplicate tag based policy on Asset Details Policy tab

Problem: When a specific tag is present both on the table as well as on any column of the same table, and a couple of requests to Ranger with the same tag and it displays the policy twice on list.

CDPDSS-527: Cloudera Data Catalog specific user roles not supported.

Problem: The following user roles are not supported:

- DataCatalogCspRuleManager
- DataCatalogCspRuleViewer

Adding or removing these roles in the User Management will not have any impact on Cloudera Data Catalog user access.

CDPDSS-81: Cloudera Data Catalog Issue

Problem: When you navigate to the **Cloudera Data Catalog** page from the **Environments** page or **Data Lakes** page, the context is not retained and the **Cloudera Data Catalog** page loads with the last used context.

Workaround: You must change the context manually by selecting the right data lake.

CDPDSS-302: Profiler Manager server requires a manual restart after installation for DP Tags to reflect in Atlas

Problem: Once the profiler is launched, run the sensitivity profiler, the profiler recognises the sensitive tags but fails to push them to Atlas.

Workaround: You must restart the profiler manager. Note that restarting the profiler manager is a one-time task.

CDPDSS-360: Sensitive and Hive Column Profilers do not support managed Hive tables.

Problem: Profilers cannot access the storage file of the managed table.

CDPDSS-517: Dataset View: Asset list displays loading icon if user is not authorized on Environment on which the dataset is created.

Problem: The dataset view page breaks if user does not have access to the Environment on which the dataset is created.

CDPDSS-570: The Filter type list should be refreshed based on the data lake selection.

Problem: Selecting the filter type from the drop-down does not display the supported entity type and also in the radio button list.

CDPDSS-547: Data lake name missing on the Asset Details page.

Problem: The data lake name was not visible on the Asset Details page .

CDPDSS-594: Database filter and create dataset button gets hidden

Problem: When you clear the query used the clear button, the databaser filter gets hidden and create dataset button on the search page gets disabled.

CDPDSS-636: User able to view assets of a dataset which is available in an inaccessible data lake.

Problem: Users are able to view the assets in a data lake for which they have no permission or access.

CDPDSS-672: On the Asset-Details page, when you make changes and save them, the Scheme page is refreshed, the Schema tab is disabled, and Overview tab becomes operational.

Problem: When you perform some operations on the **Asset Details** page and save the changes, the **Schema** tab gets inactive.

CDPDSS-727: CSP Custom Rule Validation is getting stuck in "Validation Running" status

Problem: The rule gets stuck in the Validation Running status.

Workaround: Keep sensitivity profiler enabled when running CSP rule validation.

CDPDSS-955: Cloudera Data Catalog uses wrong attribute to read createTime of entities and misinterprets lastAccessTime attribute.

Problem: Cloudera Data Catalog does not match the Created On date with Atlas.

CDPDSS-949: Asset registration pertaining to GLUE integration is not possible.

Problem: Asset registration failing with GLUE integration.

CDPDSS-921: Table created few days ago and configured recently is getting backlisted.

Problem:When an asset created date is older than the filter applied for black-listing, the asset still gets blacklisted (not picked during the profiler run).

CDPDSS-1007: On-Demand Profiler remains in "in-progress" state if the profiler job fails.

Problem: The On-Demand Profiler remains in the "in-progress" state if the profiler job fails due to some exception (For example: Permission denied exception).

CDPDSS-144: Filter Results Count Mismatch

Problem: When you search and filter the results using Created Before, the count displayed in the filters panel is incorrect. But, the number of search results is accurate.

CDPDSS-144: Search icon within dataset does not work

Problem: When you try to search for an asset within a dataset and click the search icon, it does not yield any results.

Workaround: Press enter to view the search results after entering a string in the search box.

CDPDSS-143: Edit button appears after creating a new dataset

Problem: After creating a new dataset, when you visit the dataset, the Edit button appears instead of Add Assets.

Workaround: Refresh your browser and reopen the dataset. You will be able to see the Add Assets button.

CDPDSS-81: When you select a table tag, owner filters along with the counts disappear

Problem: When you select a table tag, owner filters along with the counts disappear.

Workaround: Make sure you select table tags before selecting the owner filters.

CDPDSS-789: Adding system tag 'dp' prevents further editing of tags on the assets

Problem: Adding system tag 'dp' prevents further editing of tags on the assets

Workaround: Choose not to add the tag 'dp' on any of the assets.

CDPDSS-795: Search bar in schema tab goes empty, but previous results are shown

Problem: Search results displays previous results.

Workaround: Refresh the page.

CDPDSS-797: Schema search tab displays all results for not existing search string

Problem: Search by terms which have no hits and returns all assets.

CDPDSS-1055: Multiple tagging across different types throws 504 Gateway Timeout error

Problem: 504 Bad Gateway error is displayed.

CDPDSS-1075: When one specific asset is tagged to multiple tags from the search page, an incomplete alert is displayed

Problem: Asset name display is inconsistent. Only the first tag is displayed, remaining tag messages do not display the asset name.

CDPDSS-1084: Confusing terminologies.

Problem: Different usage terminologies. Not consistent.

CDPDSS-1373: "Unauthorised" message is displayed by middleware on the Cloudera Data Catalog UI for every API failing 401

Problem: The Cloudera Data Catalog UI shows a pop-up at the bottom with message: "unauthorised request" and no context.

CDPDSS-1375: Search enhancements by TERM

Problem: In Cloudera Data Catalog, if the term is included in the asset search query, the count of the assets next to the Data Lakes displays NA. It is because, Atlas search with Glossary term is returns approximate count as -1.

CDPDSS-1410: While navigating to another node from the Lineage tags filter, the tool-tip is retained.

Problem: From the lineage tags filter, the tool-tip is retained.

CDPDSS-1414: The "Cancel" button should exit the "Classifications" pane.

Problem: The "Cancel" button does not go away from the the "Classification" pane.

CDPDSS-1415: The Attribute name field in the "add classification" gets truncated when its very long and it cannot be read completely.

Problem: The attribute name field cannot be read fully.

CDPDSS-1416: Intermittent: Clicking add classifications reloads the lineage.

Problem: Clicking "add classifications" reloads the lineage but takes a very long before completely loading it.

CDPDSS-1474: [Glossary Terms] Difference in the Slider panel used in Asset Details and Search pages.

Problem: Indifferent display of slider panels in **Asset Details** and search pages.

CDPDSS-1475: Glossary Term | Allow selection on clicking any part of the row.

Problem: While associating a term to column or entity, the user has to click on the + icon for selection and not just tap on any object within the row.

CDPDSS-1484: Atlas and Ranger services UI link for Cloudera Private Cloud Base clusters is broken.

Problem: The managerUri attribute in Classic Cluster does not contain the Knox gateway URL.

CDPDSS-1498: For AWS Glue assets, users cannot add terms.

Problem: An error is displayed when Glue asset user tries to add terms.

CDPDSS-1501: On applying the term name in the Glossary filter on the search page, the count for data lakes is rendered as NA

Problem: The result count for data lakes displays are NA but the result list loads correctly.

CDPDSS-1664: Inherited attributes are not getting displayed for tags in the Schema tab slider for specific user cases.

Problem: Missing attributes in the Schema tab slider search bar.

CDPDSS-1179: Cloudera Data Catalog search via Atlas's /v2/search/basic API doesn't retrieve results when the search text mentioned in the entity filter criteria (like searching by DB / table name) has special characters (like + - & | ! ( ) { } [ ] ^ " ~ * ? :)

Workaround: No known workaround via Cloudera Data Catalog.

CDPDSS-1346: Asset Details view of HiveTable not displaying all columns

Workaround: N/A

CDPDSS-1276: Error is thrown when advanced configuration values are changed to 0 or left blank

Problem: Error is thrown when advanced configuration values are changed to 0 or left blank and saved

CDPDSS-1506: For specific databases, the terms associated to them are not getting deleted

Workaround: N/A

CDPDSS-1595: Cloudera Data Catalog Profilers are not functioning as expected for Ranger Authorization Service (RAZ) enabled Data Lakes

Problem: When you are using the RAZ enabled clusters, you can launch the Cloudera Data Catalog profilers but the profilers do not function as expected.

Workaround: To enable RAZ with Azure, you must update the existing Ranger policy and also add a couple of new Ranger policies:

- Existing Policy: Default: Hive warehouse locations

  Add dpprofiler user and assign Read + List permission.

Later, add the following policies:

- Policy Name: Profiler locations (Anything that Ranger can accept)

Depending on your storage location, note the following and set them accordingly:

If you have a sample location like:

```
abfs://<Storage account>@<Storage account container>.dfs.core.windows.net/<s
ub_directory>/dpprofiler
Storage account:<Storage account>
Storage account container : <Storage account Container>
Relative path: /*/dpprofiler
```

Select All permissions to allow conditions to take effect.

- Policy Name :Profiler Ranger Audit Logs (Anything that Ranger can accept)

  Depending on your Storage location, note the following and set them accordingly:

  If you have a sample location like:

  ```
  abfs://<Storage account>@<Storage account container>.dfs.core.windows.net/
  <sub_directory>/ranger/audit
  Storage account:<Storage account>
  Storage account container : <Storage account Container>
  Relative path: /*/ranger/audit
  ```

  Select Read + List permissions for allow conditions.

  Restart the Profiler Manager service in the Cloudera Data Hub Cloudera Manager instance.

CDPDSS-1748: Release specific features call fails with 404 error on the search page

Problem: Release specific features call is failing with 404 error for data lakes with no profiler cluster.

CDPDSS-1774: In the edit dataset page, on selecting all the assets and deselecting a few does not work

Problem: In the edit dataset page, on selecting all the assets and deselecting a few of the assets, deselects them in the UI but on hitting the 'delete' (trash icon), all the assets (selected and unselected ones) are deleted.

CDPDSS-1801: Options drop-down for a dataset stays open even though drop-down is opened for another dataset

Problem: The drop-down for all the datasets stay open.

CDPDSS-1807: Overlapping b/w icon and inline text in text boxes

Problem: This overlapping of icons and inline text is removed from all the pages.

CDPDSS-1808: UI Issues with new 'add asset' modal to add assets to dataset N/A

Problem:Directly clicking on 'search' field of 'Table Tag' shows no results found.

Typing "_" in the search field erases the entire search string

'Clear' text button is not working for Table tags , however it works for column tags

CDPDSS-1897: Show process nodes option is displayed for entity types which do not have process nodes

Problem: The toggle option to show the "process nodes" is present even for entity types which do not have process nodes.

Workaround: N/A

CDPDSS-1954: User cannot delete profilers when no entities are displayed in search result

Problem: The "Action" button containing the option to delete profilers is also removed when the search results yeild no entity type.

CDPDSS-1988: Asset names are not clickable for registered assets in Glue Data Lake

Problem: Once the asset is registered observe that the 'asset name' is in blue but not clickable.

CDPDSS-2125: Unable to create profiler cluster for Cloudera Runtime version 7.2.11 and below

CDPDSS-2130: Unregistered Glue tables consume more time to load Asset Details page.

CDPDSS-2045: Duplicate API calls in Profiler job listing screen

CDPDSS-2047: Azure | Profilers HA | Issue with DB connection

The issue can be solved by upgrading the data lake and the profiler cluster to Cloudera Public Cloud 7.2.14 or later.

CDPDSS-2032: Profiler Manager Service call is failing for Classic Clusters

CDPDSS-2120: Test run on CSP tag creation fails with 404 error

CDPDSS-2140: System tags status is marked as 'suspended' on a medium duty cluster with Profiler Management Services on Cloudera Shared Data Experience

Problem: The system tags are in a suspended state instead of default state which is 'deployed'.

CDPDSS-1965: AWS S3 V2 Directory type nodes could not be clicked in the Asset Details page

The issue was fixed by enabling AWS S3 V2 Directory type nodes in a configuration file.

CDPDSS-2141: Delete profiler option should be available in Cloudera Data Catalog UI for all the states

Problem: This functionality is present on CloudBreak UI and hence can be made available from Cloudera Data Catalog App as well.

CDPDSS-2839: Hadoop version upgraded from 7.2.17.200-29 to 7.2.17.301-3

Problem: The Hadoop version is upgraded from 7.2.17.200-29 to 7.2.17.301-3 to remediate CVE-2023-44487 and other CVE fixes are implemented as well.

CDPDSS-2925: The getNextScheduledRun API fails if createtime value is in the Allow List.

The issue was fixed by updating the Cloudera Data Catalog Profiler configuration.

Workaround: Remove asset filter rules with createTime filter and recreate them.

CDPDSS-2926: Profilers page displayed Start Time and Last Updated time in epoch time format

The Profilers page was updated to use the UTC time format.

CDPDSS-2928: Refreshing the Profiler page deletes configuration details

The issue was fixed by updating the Profiler page.

CDPDSS-2929: The Asset Details page reports a 404 error for impala_process_execution type assets.

The issue was fixed by removing the unsupported lineage widget for impala_process_execution type assets.

CDPDSS-2934: Classification and Glossary filters are not loaded on search page intermittently

The issue was fixed by updating the entity search components.

CDPDSS-2948: 400 Bad Request is generated in the Asset details page for Table Type "VIRTUAL_VIEW"

The issue was fixed by adding a tooltip and disabling the ODP run button for the empty location.

CDPDSS-3037: Asset Details page does not highlight link for Datasets

The **Asset Details** now correctly highlights links to Datasets.

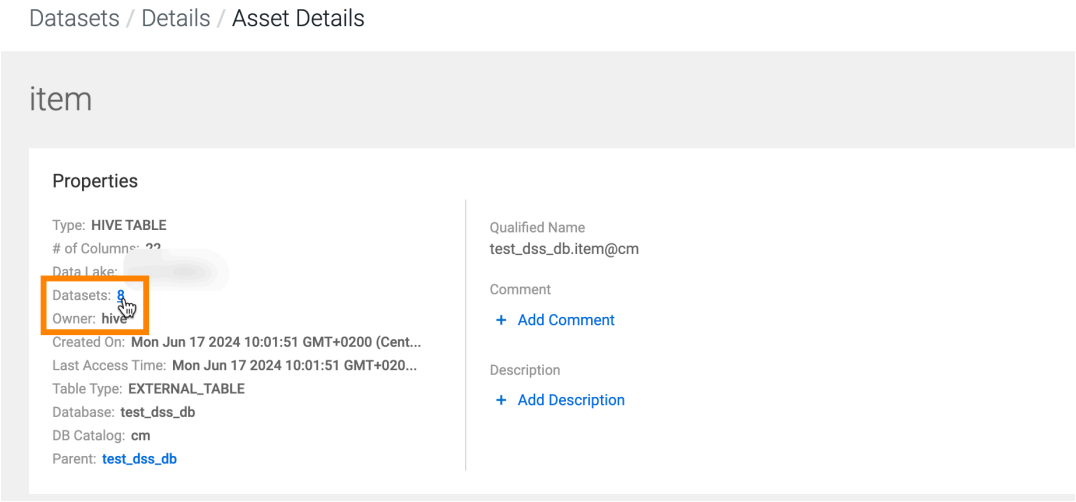

CDPDSS-2958: Schema rows are not ordered as in their database

Rows under the **Schema** tab in **Asset Details** are ordered based on their position in their parent database.

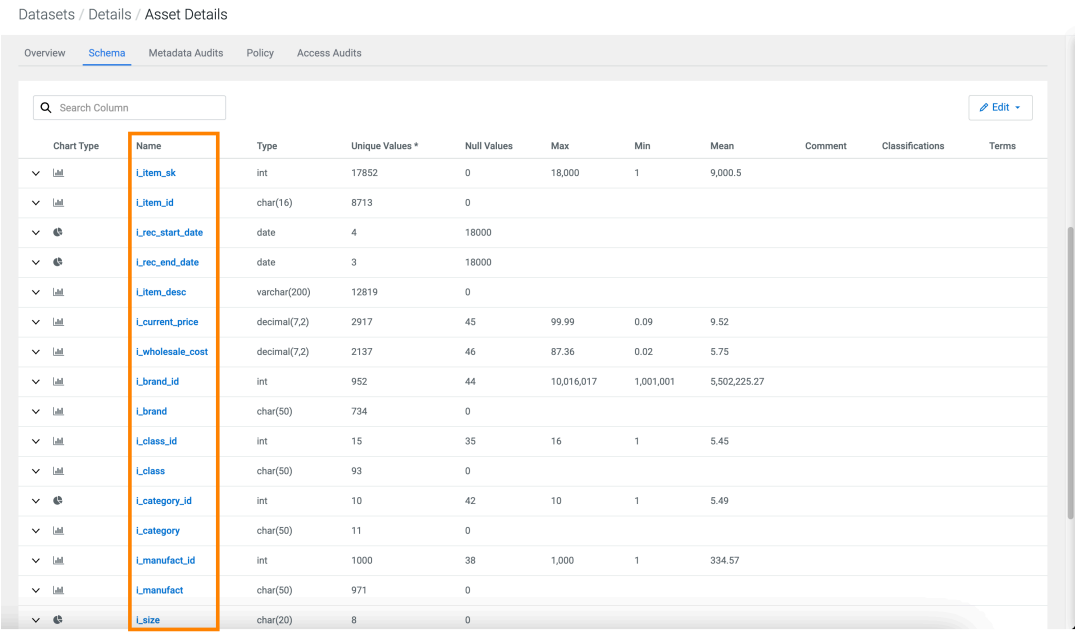

CDPDSS-2959: Asset Details does not provide a link to the parent database

- Now, the parent database can be accessed via a link in **Asset Details**.
- The Qualified Name is no longer trimmed. Longer names are displayed in multiple rows.

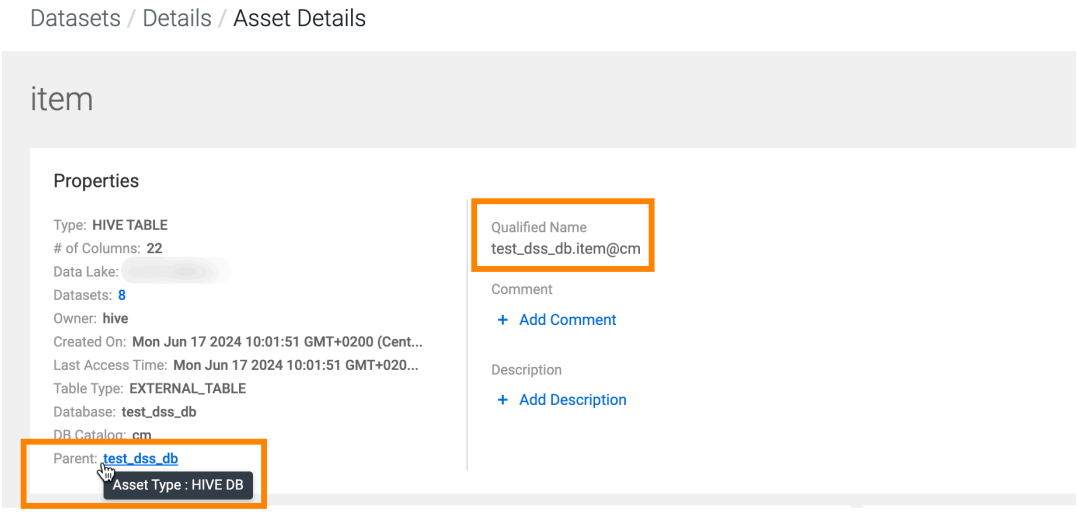

CDPDSS-3042: Edit classification is failing with guid undefined error

Trying to edit the classifications in the **Asset Details** screen no longer results in the "List properties for entity request failed Error from Atlas. Not Found : Received 404. Body {\"errorCode\": \"ATLAS-404-00-005\",\"errorMessage\":\"Given instance guid undefined is invalid/not found\"}" message.

CDPDSS-3049: Expanded graph is displayed on every page of the Schema tab

Previously, in the **Asset Details** page under the **Schema** tab, the bar graph of profiled assets might stayed visible for a selected asset even when moving to the next page in the list. Now, changing the page correctly closes the previously selected asset, and only displays the assets in the active page.

CDPDSS-3050: Username is not visible in Cloudera Data Catalog for the Local Administrator user

The username for local administrator users is no longer hidden. It is shown in the bottom left corner of the navigation sidebar.

CDPDSS-3077: The Datasets Details page fails to load the list of assets

Datasets containing deleted assets no longer fail to render the assets list with the following error: pages.dataset.labels.err-undefined-msg(Error Code - undefined)

CDPDSS-3366: Hbase Table: The Asset Details page is distorted with very long description under Schema tab

After the update, the description length no longer disrupts the **Schema** tab in **Asset Details**.

CDPDSS-3319: No filter options for Access Type for HIVE DB type assets

After the update, all filter options are correctly displayed for Hive DB type assets in Asset Details Access Audits .

CDPDSS-3318: Access Audits doesn't have the filter for Access Type INSERT

After the update, the INSERT filter option is correctly displayed for relevant assets in Asset Details Access Audits .

CDPDSS-3064: Cancel and Suspend button is disabled on revert

In Profilers Tag Rules , when trying to revert a suspended tag rule, the Cancel and Suspend buttons are no longed disabled.

CDPDSS-3371: Asset Details is distorted with very long description

After the update, the very long strings in the **Description** field are trimmed and no longer distort **Asset Details** by being displayed in full length.

CDPDSS-3351: Unable to see comments on the Hive columns in Asset Details

Full comments can be displayed in Asset Details while hovering over the **Comment** field compared to the earlier limited view.

CDPDSS-3169: Asset Details Page - Optimize API to avoid 504 Error

Before the update, by some tables with thousands of columns and one ALTER statement on each column to add comment, the `/api/atlas/v2/entity/guid/` API call took more than one minute and caused the timeout. Optimizing the API calls related to **Asset Details** led to improved performance and avoiding potential 504 errors when opening assets.

CDPDSS-3073: API Status code is 200 even if add classification is failed because of wrong datatype for attribute

When a classification is added to an asset with a mismatching attribute value type, the correct error message is shown, including the correct status code 400.

CDPDSS-3061: Add assets is not displaying all the assets when searched based on Owner/DBName/Tags

When adding assets to a dataset in Dataset Details , all assets will be correctly displayed when filtering the available assets by DB Name.

CDPDSS-3031: Number of Jobs count is not matching in the Profilers/Jobs page

The number of profiler jobs now correctly matches the between the **Filters** panel and list of profiler jobs in ProfilerJobs if the Rows per page: 500 setting is used.

CDPDSS-3030: Rows displayed in the Profilers/Jobs page is not matching with the selected Rows per page value

The list of profilers in Profiler Jobs correctly matches the number set in the The Rows per page setting.

CDPDSS-3417: Incorrect results are being displayed after navigation, when "owners" filter is selected

In the **Search** page, returning to the list of results after opening the **Asset Details** of one the results and repeating the search with the same filter, other entities belonging to owners different than the selected no longer appear. Only entities belonging to the selected owners are displayed.

CDPDSS-2571: Events filtered from Atlas that are not assets or objects

The API call to Apache Atlas creating the default quick search response was streamlined by removing events that are not relevant in the Cloudera Data Catalog context, resulting in less clutter in the default view of the **Search** page.

CDP Public Cloud / Data Catalog

Cloudera Data Catalog Top Use Cases

Date published: 2019-11-14
Date modified: 2025-01-16

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Search for assets **4**
Filters 4
Accessing Data Lakes 5
Searching for assets using Atlas glossaries 6
Using terms in Cloudera Data Catalog 7
Mapping glossary terms 7
Searching for assets using glossary terms 10
Additional search options for asset types 11
Searching for assets using additional search options 13
Accessing tables based on Ranger policies 14
Creating classifications for selected assets 14
Additional entity type selection for searching assets 16

Managing Profilers **17**
Profiler data testing 18
Launching profilers 18
Launching profilers using the command-line 22
Deleting profilers 24
On-Demand Profilers 27
Profiling table data in non-default buckets 28
Tracking profiler jobs 29
Viewing profiler configurations 31
Ranger Audit Profiler configuration 31
Cluster Sensitivity Profiler configuration 33
Hive Column Profiler configuration 35
Understanding the Cron Expression generator 37
Backing up and restoring the profiler database 38
About the back up script 38
Running the back up script 39
Enable or disable profilers 40
Profiler Tag Rules 41

Tag management **42**

Search for assets

On the Cloudera Data Catalog **Search** page, select a data lake and enter a search string in the search box to view all the assets with details that contain the search string.

When you enter the search terms **Search**, you are looking up names, types, descriptions, and other metadata collected by Cloudera Data Catalog. The search index includes metadata (not data) about your environment and cluster data assets and operations. You can make the search more powerful by associating your own information (business metadata) to the stored assets.

Note:

For the selected data lake, click the Atlas and Ranger links to navigate to the respective base cluster services in a new browser tab.

Related Information

Understanding datasets

Filters

Use filters to refine the overview of all your available assets.

You must have access to at least one data lake to search and filter your results. By default, a data lake is already selected for you if you have access to it.

You can further refine your search results using filters as follows:

Owner

From all the owner names that appear, you can select the owner to further refine the results and display those search results with the selected owner.

Type

Select an entity type to view all the assets stored in that type of database.

- Azure BLOB
- Azure Container
- Azure Directory
- AWS S3 Bucket
- AWS S3 Object
- AWS S3 Pseudo Dir
- AWS S3 V2 Bucket
- AWS S3 V2 Directory
- AWS S3 V2 Object
- Hbase Column Family
- Hbase Namesspace
- Hbase Table
- HDFS path
- Hive Column
- Hive DB
- Hive Table
- Iceberg Column
- Iceberg Table
- Impala Column Lineage
- Impala Process

- Impala Process Execution
- Kafka topic
- ML Model Build
- ML Model Deployment
- ML Project
- RDBMS Column
- RDBMS DB
- RDBMS Foreign key
- RDBMS Index
- RDBMS Table
- Spark Application
- Spark Column
- Spark Column Lineage
- Spark DB
- Spark ML Directory
- Spark ML Model
- Spark ML Pipeline
- Spark Process
- Spark Process Execution
- Spark Table

Note: After selecting an entity type, further filters related to that type will be available under the More filter. For example, selecting the Hive Table type will enable the Column Tag filter.

Entity Tag

Use entity tags to refine your search results. You can add business metadata as entity tags in Atlas as classifications, or in the **Atlas Tags** menu. Use these tags to refine your search results and view the details of the required data asset.

Time Range

You can filter your assets by the **Created On** date (if provided by Atlas) after selecting and asset Type. Use the calendar widget to select a range and click Apply.

Glossary Terms

You can filter assets based on business glossary terms. You can search for any asset without any entity type restrictions.

Note: This filter appears only if Atlas has terms set up.

Click Cancel for any filter to clear the selection or Clear All to reset all your filters.

In the resulting list of your matching assets, you can click a row and see the following:

- **Qualified name**
- **Database**
- **Classification**
- **Terms**

Clicking the Name of the entity will open its **Asset Details**.

Accessing Data Lakes

In the **Search** page, the accessible data lakes are displayed in a drop-down.

Users have access to the lakes based on the permissions that are granted. You can choose the available lake by selecting the appropriate radio button.

For example, in the following diagram, the logged in user has access to all the listed data lakes.

Note: You can search the assets of one data lake at a time.

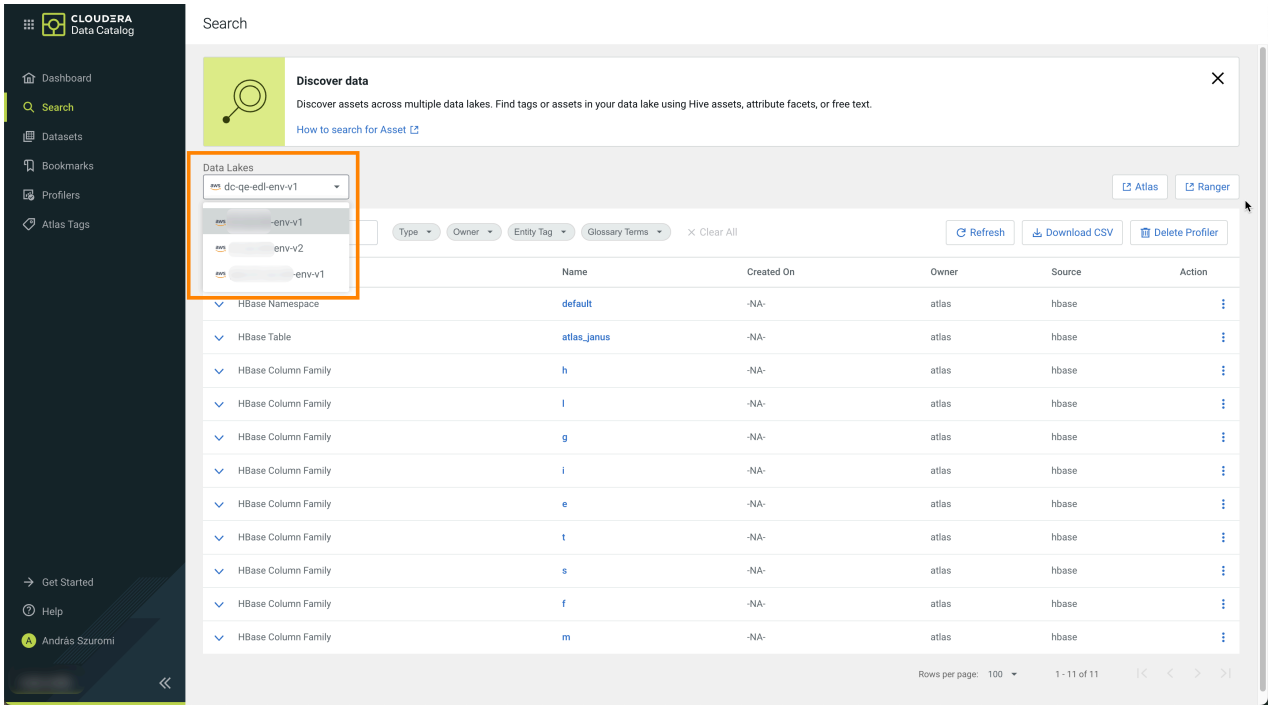

Related Information

Introduction to data lakes

Understanding data lake details

Searching for assets using Atlas glossaries

Use Apache Atlas glossaries to define a common set of search terms that data users across your organization use to describe their data.

Data can describe a wide variety of content: lists of names or text or columns full of numbers. You can use algorithms to describe data as having a specific pattern, of being within a range or having wide variation, but what's missing from these descriptions is what does the data mean in a given business context and what is it used for? Is this column of integers the count of pallets that entered a warehouse on a given day or number of visitors for each room in a conference center?

The glossary is a way to organize the context information that your business uses to make sense of your data beyond what can be figured out just by looking at the content. The glossary holds the terms you've agreed upon across your organization so business users can use familiar terms to find what they are looking for.

Glossaries enable you to define a hierarchical set of business terms that represents your business domain.

Glossary terms can be thought of as of a flat (but searchable) list of business terms organized by glossaries. Unlike classifications, terms are not propagated through lineage relationships: the context of the term is what's important, so propagation may or may not make sense.

You can search for the datasets using the Glossary Terms filter available on the **Search** page.

Using terms in Cloudera Data Catalog

You can use the Asset Details page to add or modify Apache Atlas glossary terms for your selected assets.

Use Atlas to define rich glossary vocabularies using the natural terminology (technical terms and/or business terms) of your industry. You can also create semantic relationships between your terms. Then, in Cloudera Data Catalog, use the **Terms** widget in the **Asset Details** page to map assets to glossary terms.

You can use terms in Cloudera Data Catalog to search for entities, filter them by glossary term(s), and also search for entities associated with them in Atlas.

Note: When you work with terms in Cloudera Data Catalog and map them to your assets, you can search for the same datasets in Atlas by using the corresponding terms.

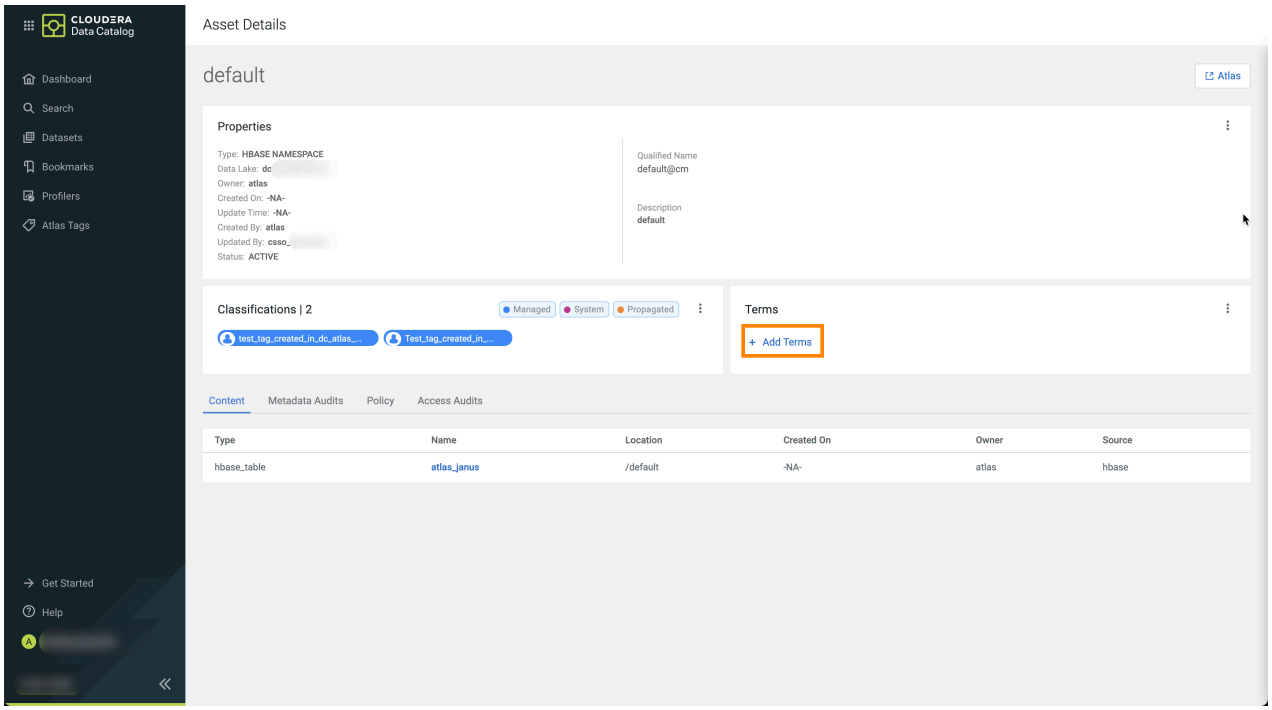

Mapping glossary terms

Cloudera Data Catalog contains the glossary terms that are created in Apache Atlas.

You can search for those terms in Cloudera Data Catalog and map specific terms with assets. You can also search for terms to delete them from the selected asset. The selected asset displays the total number of terms associated or mapped accordingly.

When you map a specific term for your dataset, the term is displayed in the following format:

```
<termname>@glossaryname>
```

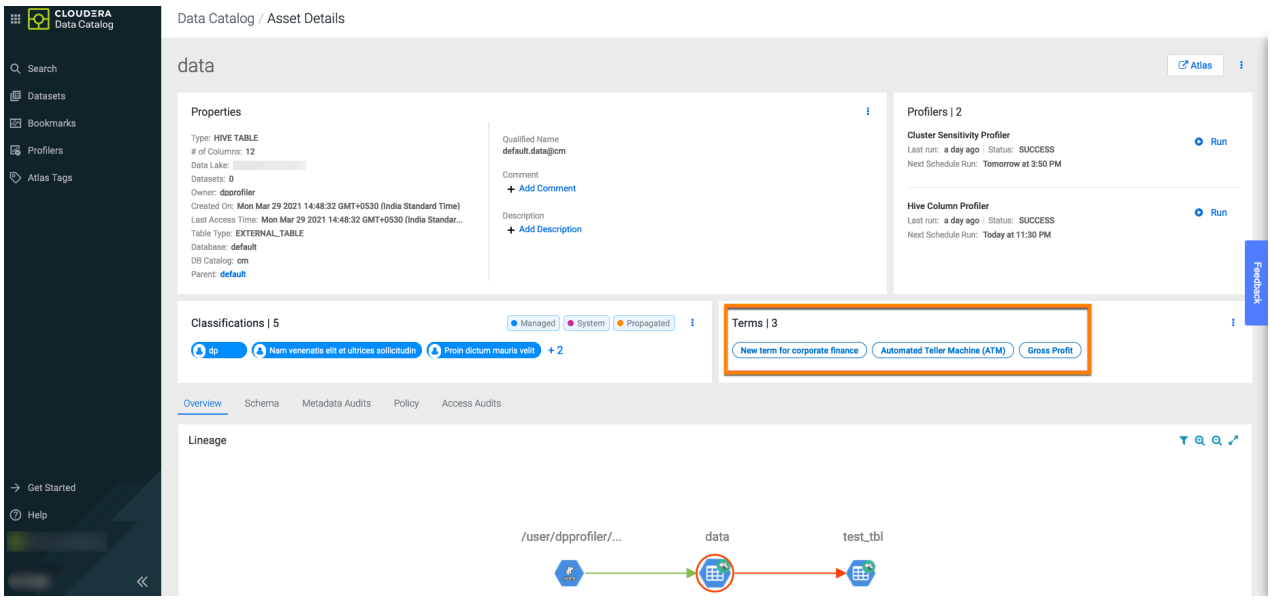

You can use the icon in the **Terms** widget on the **Asset Details** page to add new terms for your assets. Click Save to save the changes.

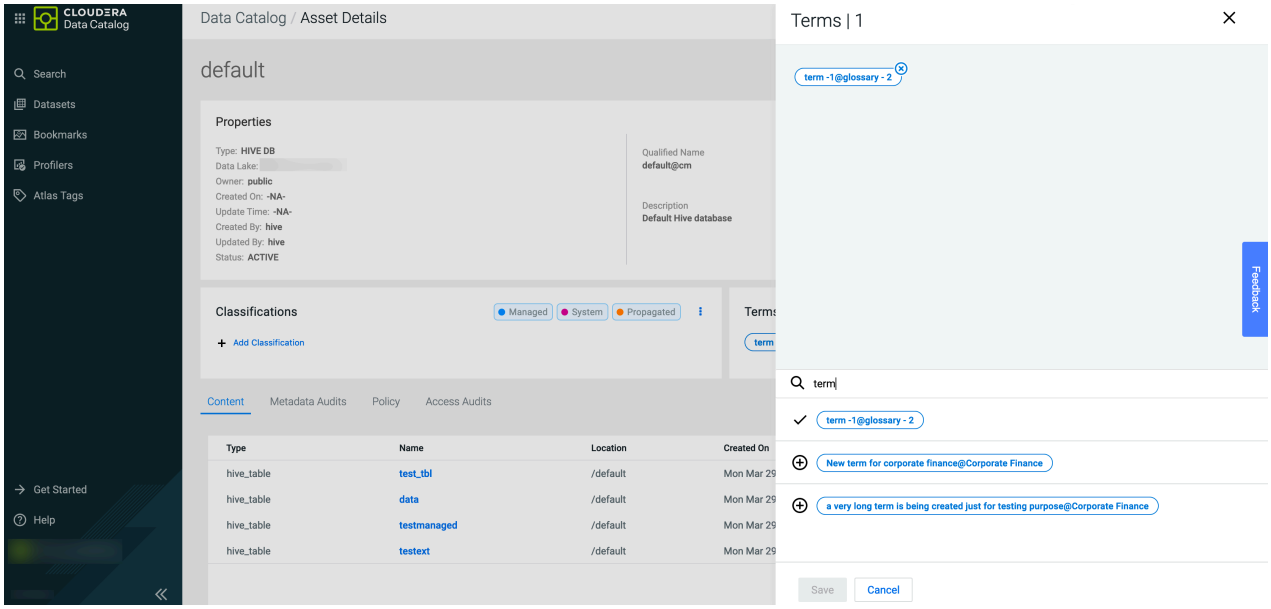

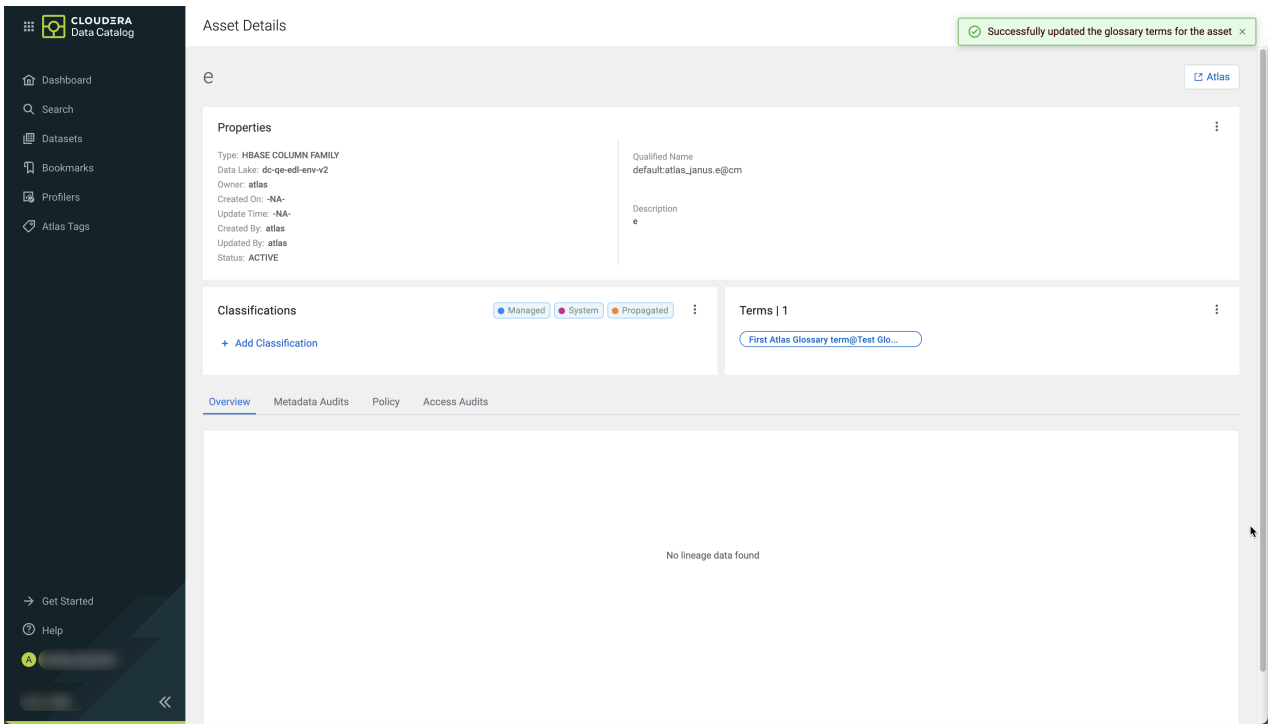

You can search for the same asset in the corresponding Atlas environment as shown in the example image.

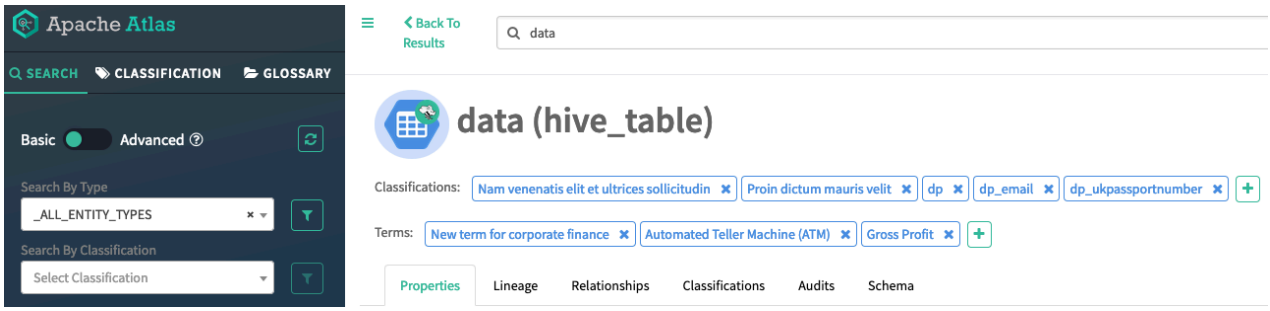

When you select a Hive table asset and navigate to the **Asset Details** page, under the **Schema** tab, you can view the list of terms associated with the asset.

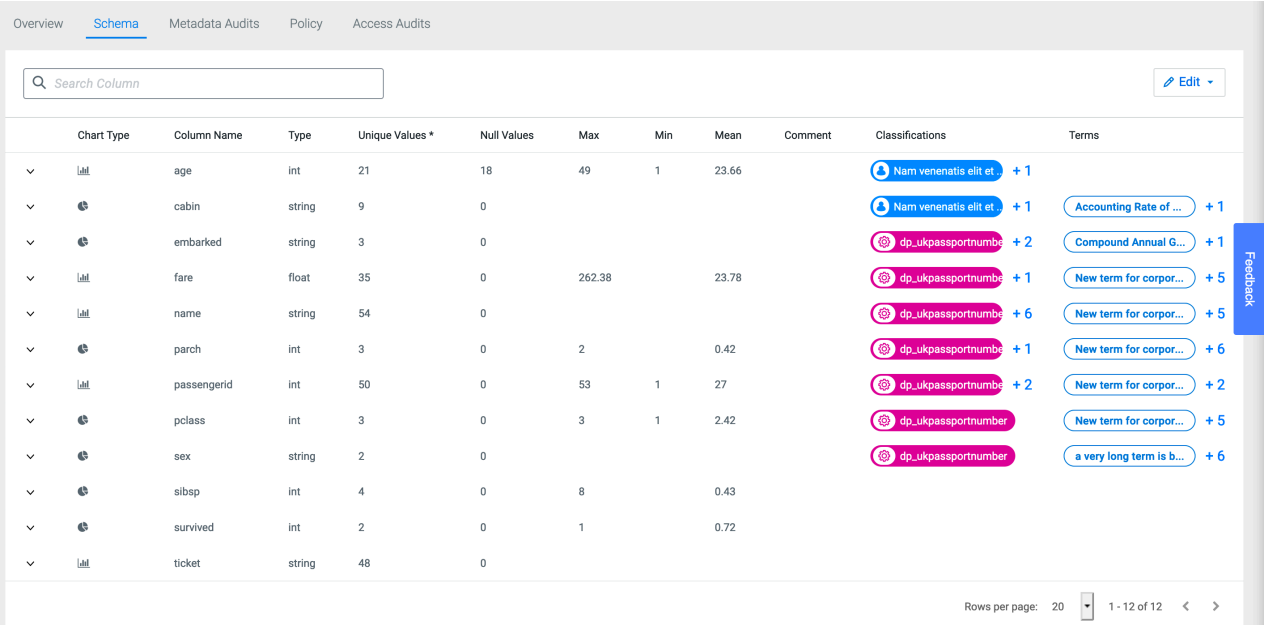

You can add or update the terms for the associated datasets by clicking the Edit button.

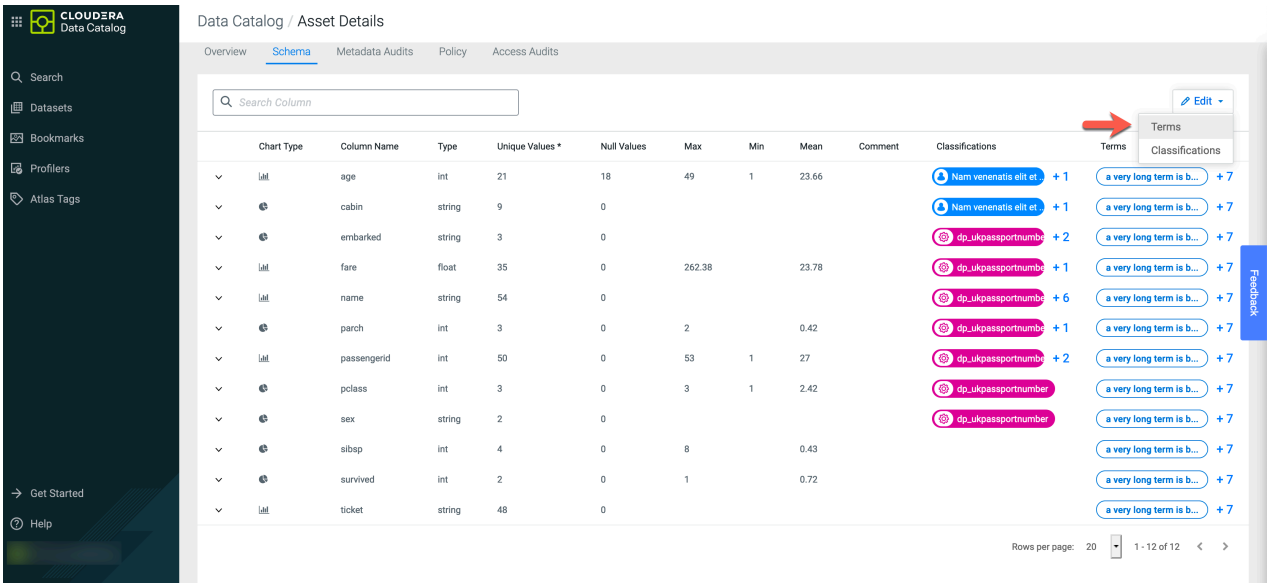

Searching for assets using glossary terms

You can search for the datasets using the Glossary Terms filter available on the Search page.

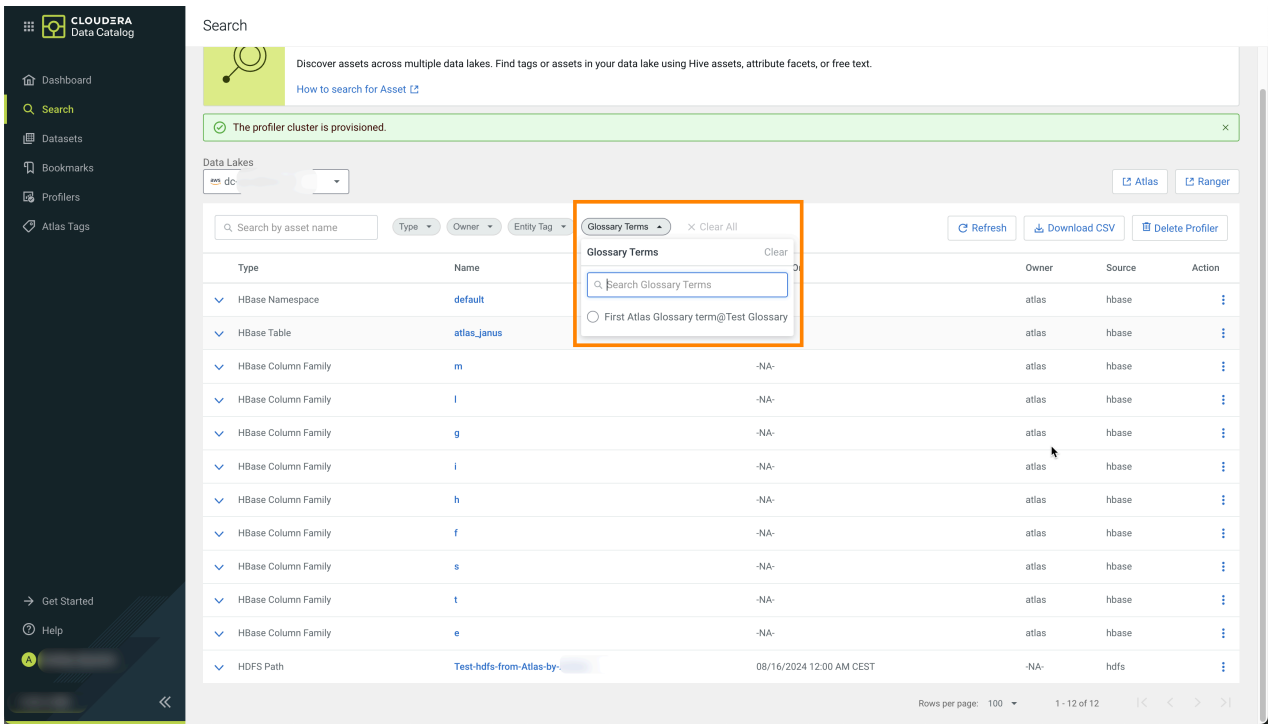

Additional search options for asset types

Using Cloudera Data Catalog, you can add or edit asset description values to search for data assets across both Cloudera Data Catalog and Apache Atlas services by using the asset content.

In the **Asset Details** page for each asset type that you select, you can add or edit **comment** or **description** fields. Including these values for the selected asset helps you to identify your chosen asset.

Using the same set of values (comment or description), you can also search for the asset types in Atlas.

Note: The comment and description options are supported only for Hive table and Hive Column assets. For other asset types, only the description option is supported.

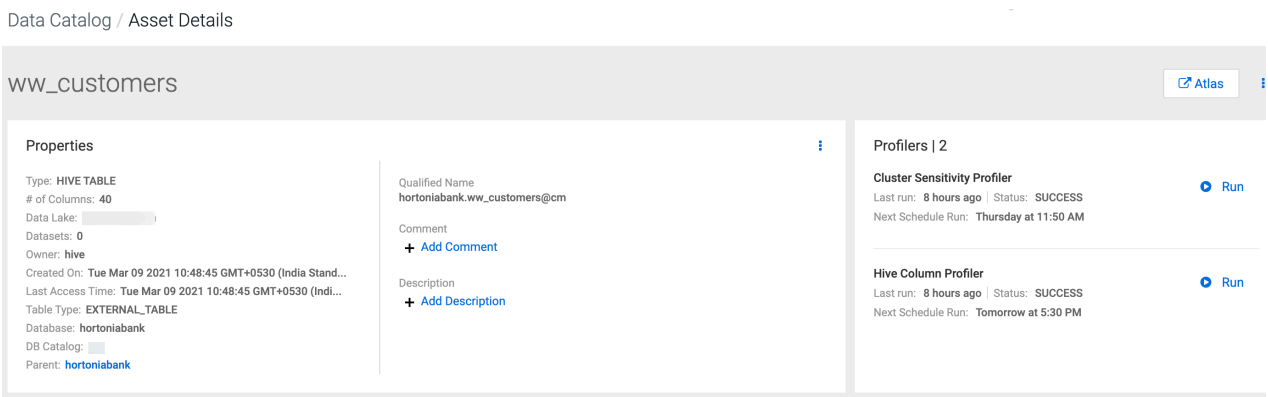

Click + Add Comment or + Add Description fields to include the respective values.

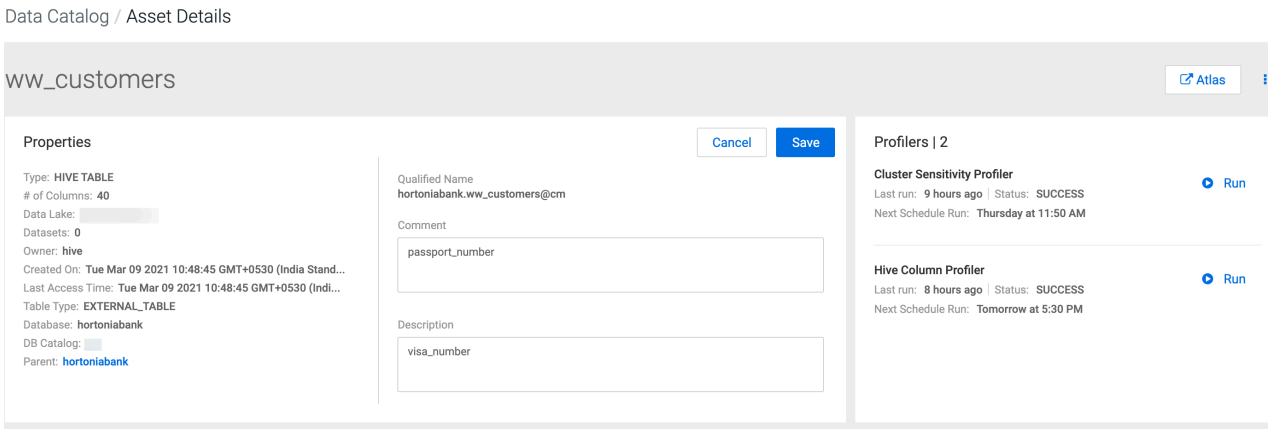

Click Save to save your changes.

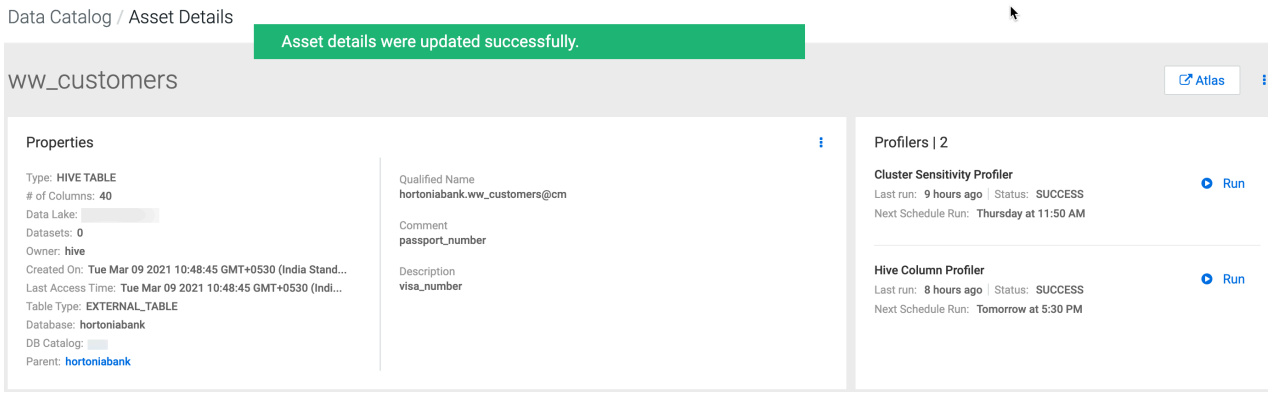

Note: You can also edit the already saved valued by clicking the icon.

Clicking on the Atlas button will navigate to the corresponding Atlas asset page as shown:

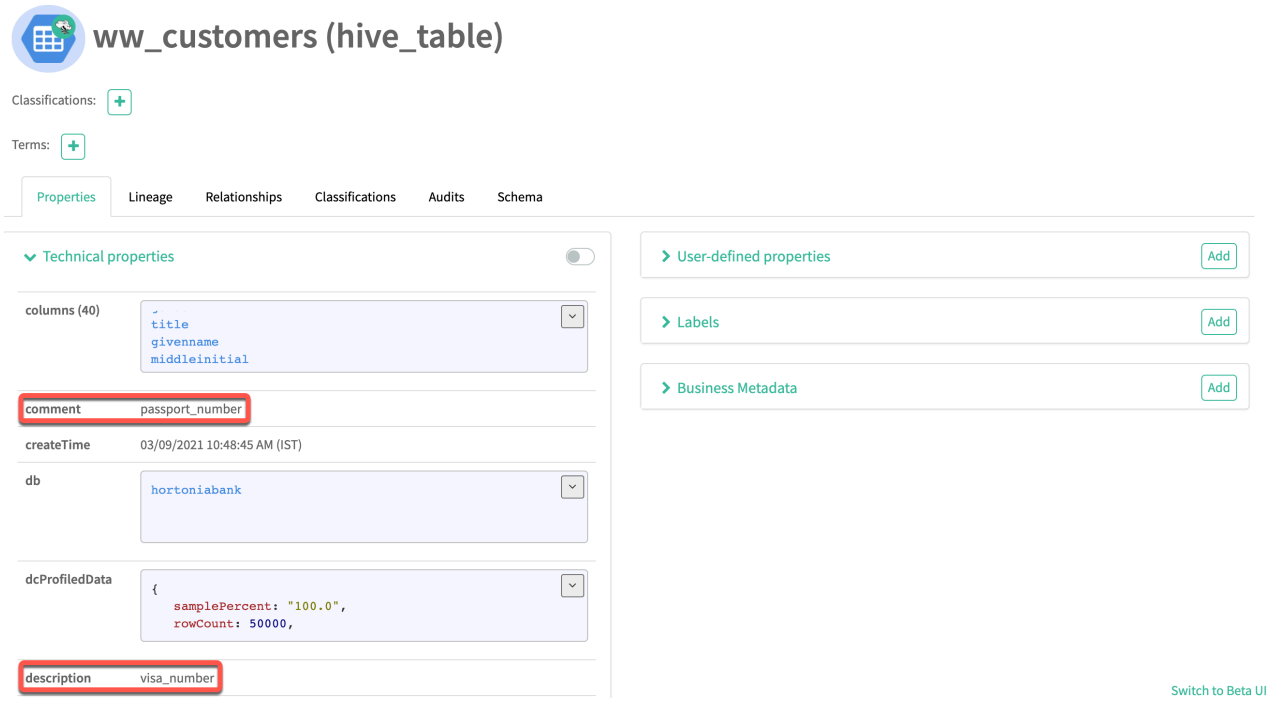

Searching for assets using additional search options

In Cloudera Data Catalog, you can select a data asset type and under the Asset Details page, to insert a comment and to provide a description for the selected asset.

The values of the **Comment** or **Description** fields can be searched in the **Search** menu. The result page displays the assets where you added your comments and descriptions without the use of filters.

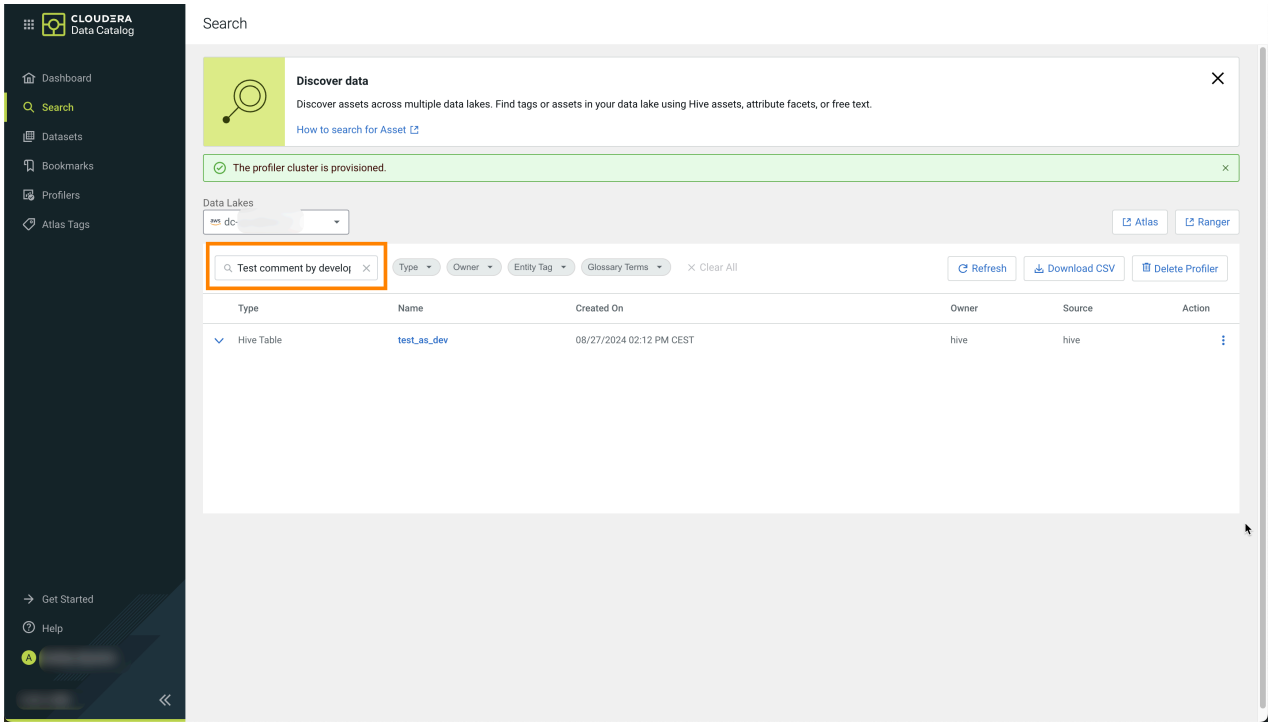

Clicking on the asset type displays the comment and description values.

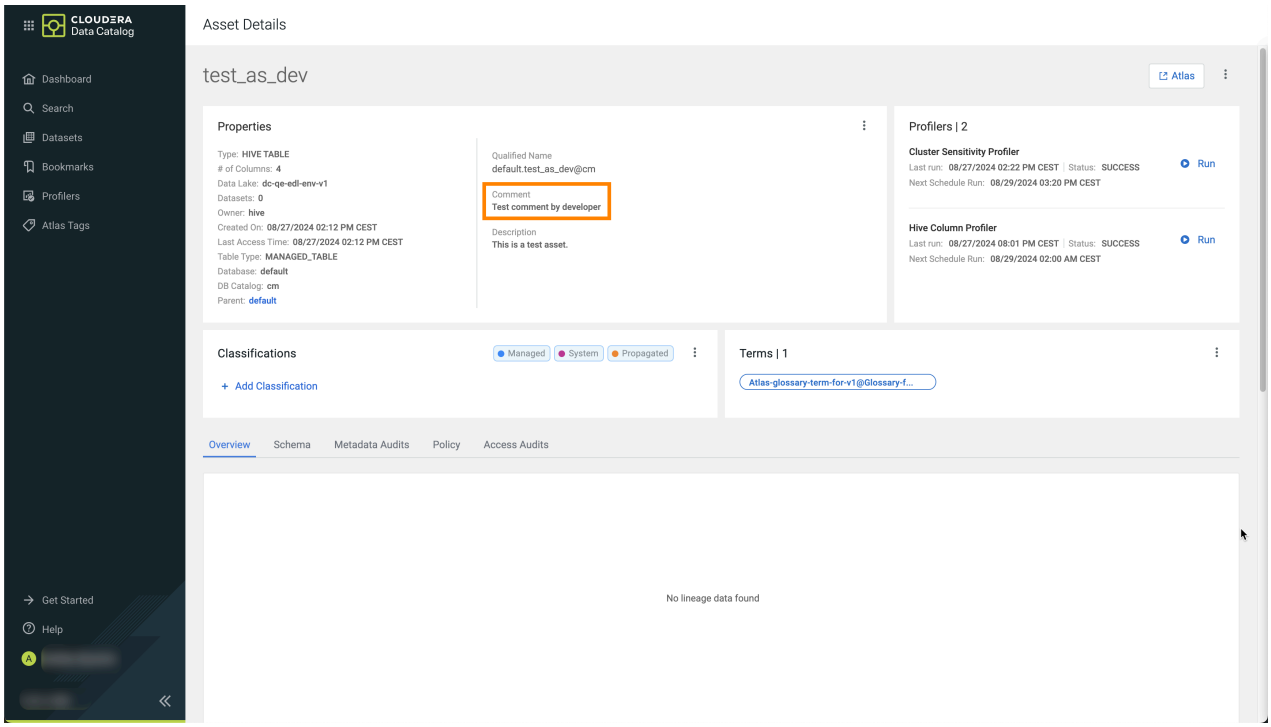

Accessing tables based on Ranger policies

When a table (in blue color link) is clicked, the Asset Details view page is displayed.

If a user is not authorized to click or view table details, it implies that the user permissions have not been set up in the Apache Ranger.

As seen in the following diagram, if users are not able to view the table details, a message appears next to the same table "Some information might not be available to unauthorised users".

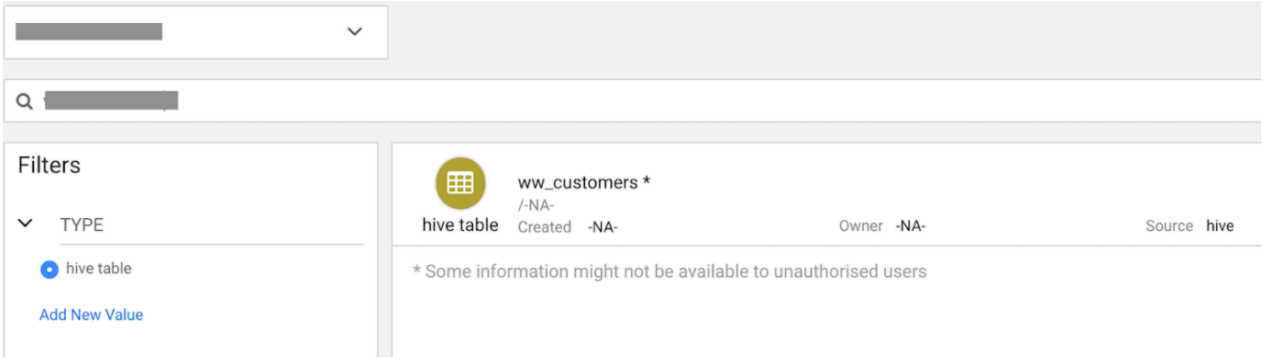

In the next example diagram, tables that have the permissions to view are displayed with a blue color link. The ones that do not have read permissions are visible in grey.

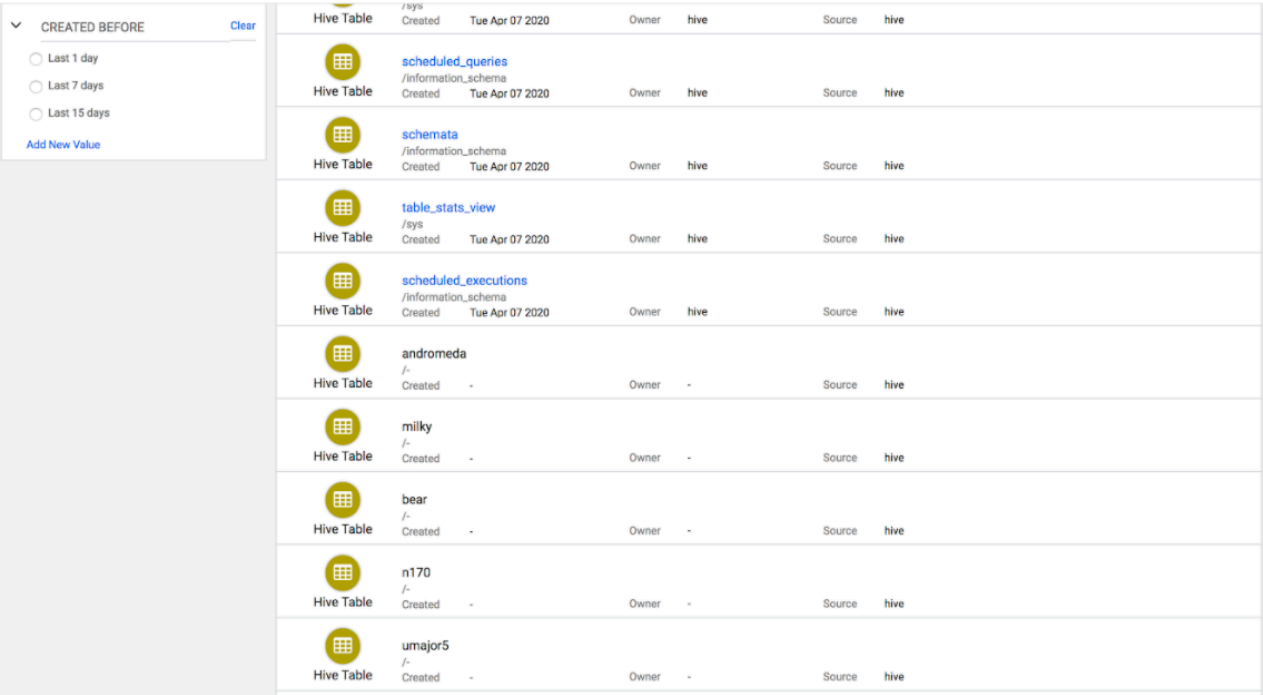

Creating classifications for selected assets

You can create classifications in multiple pages. These classifications can be associated with an asset. Then, you can use these classifications to filter your assets both in Cloudera Data Catalog and Apache Atlas.

Creating a classification from the Search page

1. Navigate to the **Search** page.

2. Click the ⋮ icon by an asset, then select Edit Classifications.

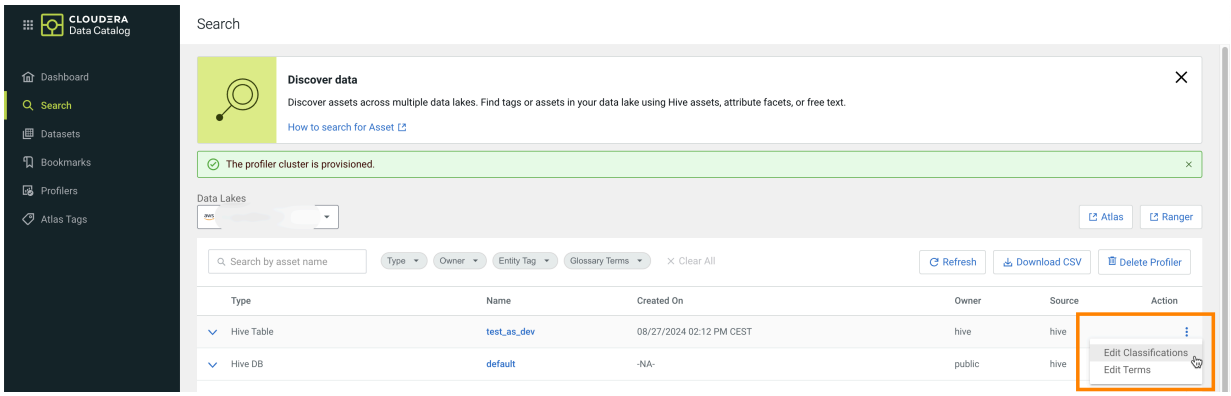

3. Search for a previously created classification or create a new one.
4. Click Save to finalize your changes.

Creating a classification from Asset Details

1. Navigate to the **Asset Details** page of an asset.
2. Click Add Classification or ⋮ icon by an asset, then select Edit.

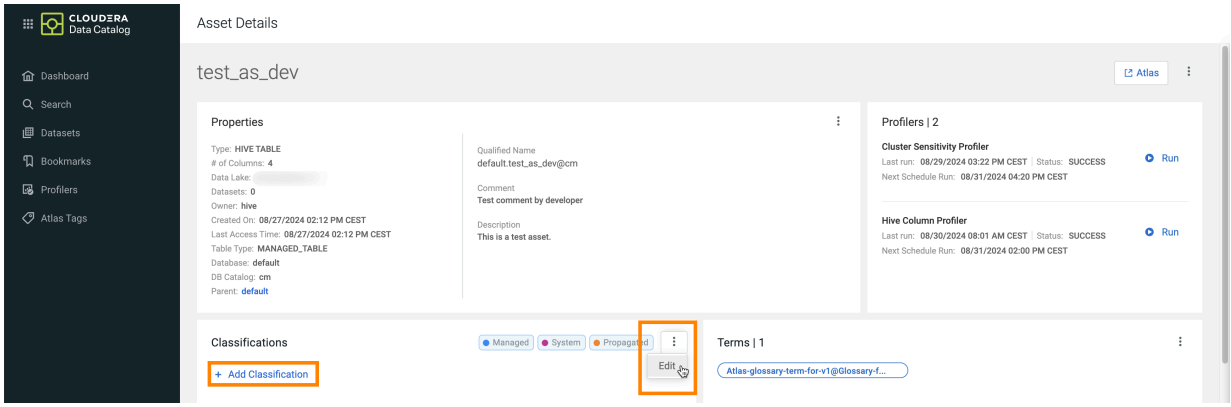

3. Search for a previously created classification or create a new one.
4. Click Save to finalize your changes.

Creating a classification in Atlas Tags

1. Navigate to **Atlas Tags**.
2. Click Add Tag.

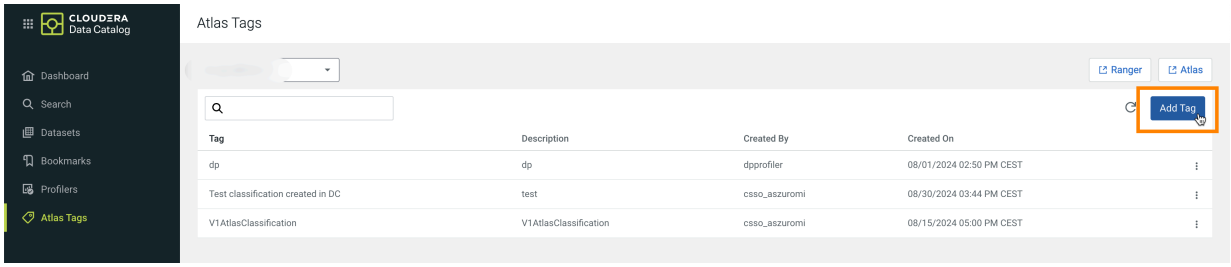

3. Fill in the details and Save your changes.

Note: Your classification still needs to be added to an asset in the **Search** or **Asset Details** menu.

Note: Classifications are synchronized between Apache Atlas and Cloudera Data Catalog.

Additional entity type selection for searching assets

Using the Cloudera Data Catalog service, you can search for assets by using entity types.

Cloudera Data Catalog users can search and discover different asset types.

Supported entity types include the following:

- Azure BLOB
- Azure Container
- Azure Directory
- AWS S3 Object
- AWS S3 V2 Object
- AWS S3 Bucket
- AWS S3 V2 Bucket
- AWS S3 Pseudo Dir
- AWS S3 V2 Directory
- HBase Table
- HBase Column Family
- HBase Namespace
- HDFS Path
- Hive DB
- Hive Table
- Hive Column
- ML Project
- ML Model Build
- ML Model Deployment
- NiFi Flow
- NiFi Data
- Iceberg Column[1]
- Iceberg Table[1]
- Impala Process
- Impala Column Lineage
- Impala Process Execution
- Kafka Topic
- RDBMS DB
- RDBMS Column
- RDBMS Foreign Key
- RDBMS Index
- RDBMS Instance
- RDBMS Table
- Spark Process
- Spark Application
- Spark Column
- Spark Column Lineage
- Spark DB

[1] Iceberg assets are discoverable in VM-based environments but they can be profiled only in Compute Cluster enabled environments.

- Spark ML Directory
- Spark ML Model
- Spark ML Pipeline
- Spark Process Execution
- Spark Table

Selecting a type triggers a search query for that type. The **Owner** of the asset is derived from the response received from the type based queries.

The following examples depict the entity type selection search results:

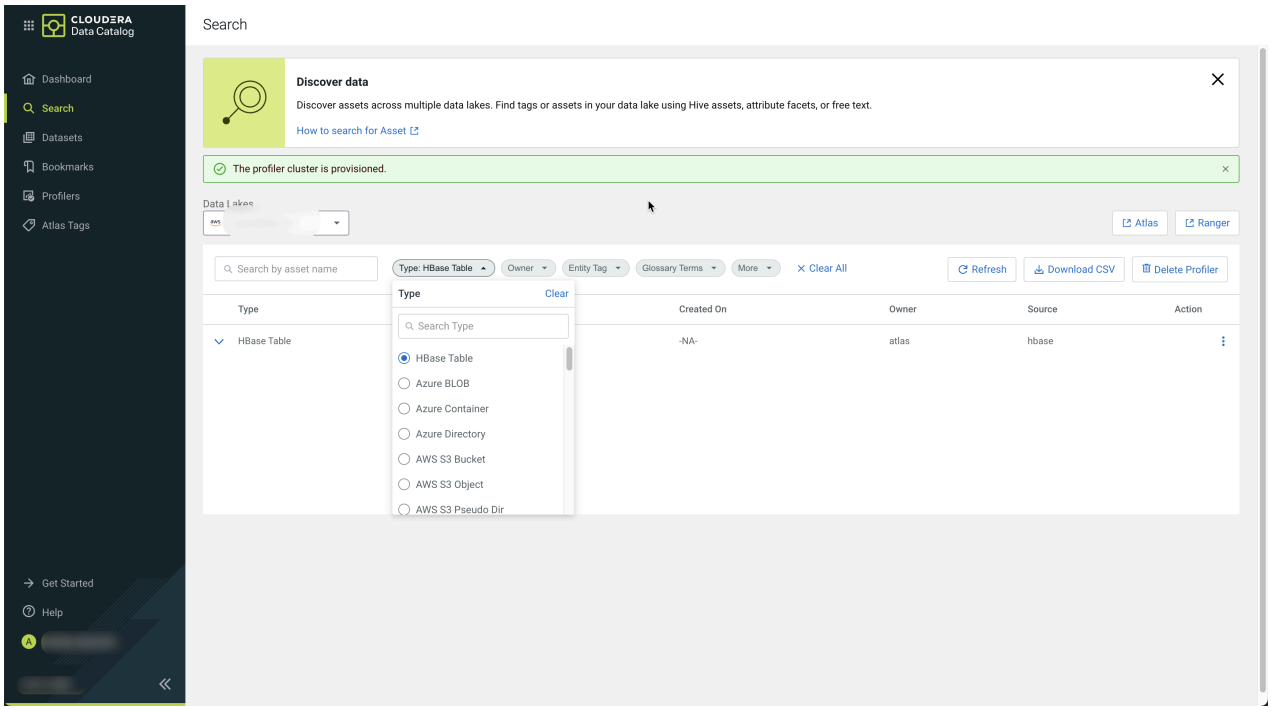

Managing Profilers

The Cloudera Data Catalog profiler engine runs data profiling operations on data located in multiple data lakes. These profilers create metadata annotations that summarize the content and shape characteristics of the data assets.

Table 1: List of built-in profilers

Profiler Name	Description
Cluster Sensitivity Profiler	A sensitive data profiler- PII, PCI, HIPAA, etc.
Ranger Audit Profiler	A Ranger audit log summarizer.
Hive Column Profiler	Provides summary statistics like Maximum, Minimum, Mean, Unique, and Null values at the Hive column level.

Limitations

- In VM-based environments, profilers do not support Iceberg Tables. However, Iceberg tables are discoverable. In Compute Cluster enabled environments, Iceberg tables can be profiled.
- In Compute Cluster enabled environments, profilers only support tables which are stored on AWS S3 storage.

- Supported file formats:
  - VM-based environments:
    - CSV
  - Compute Cluster enabled environments:
    - Hive Column Profilers and Cluster Sensitivity Profilers
      - CSV
      - Parquet
      - Iceberg tables

Related Information

Understanding the Cloudera Data Catalog Profiler
Understanding the Cluster Sensitivity Profiler
Understanding the Ranger Audit Profiler

Profiler data testing

You must note the important information about profiler services.

Note: The Cloudera Data Catalog profilers are not tested at par with the Hive scale.

The following dataset has been validated and works as expected for VM-based environments:

- DataHub Master: m5.4xlarge
- Hive tables: 3000 Hive assets
- Total Number of assets (including Hive columns, tables, databases): 1,000,000
- Total Data Size = 1 GB
- Partitions on Hive tables: Around 5000 partitions spread across five tables

Note: For Compute Cluster enabled environments, more detailed testing information will be provided in the next release of Cloudera Data Catalog.

The following dataset has been validated and works as expected for Compute Cluster enabled environments:

- Total Data Size = 300 GB
- Sampling profiler size = 50% (150 GB)

Launching profilers

In VM-based environments, you must first provision the Cloudera Data Hub to launch the profiler cluster to view the profiler results for your assets and datasets. In Compute Cluster enabled environments, after you set up the profiler, the Profiler Launcher Services automatically starts the profiler Kubernetes containers.

Note: You must be a Power User to launch a profiler cluster.

Profiler cluster in VM based environments

The Profiler Services supports enabling the High Availability (HA) feature.

Note: The profiler HA feature is under entitlement. Based on the entitlement, the HA functionality is supported on the Profiler cluster. Contact your Cloudera account representative to activate this feature in your Cloudera environment.

Attention: By default when you launch a profiler cluster, the instance type of the Master node will be the following based on the provider:

- AWS - m5.4xlarge
- Azure - Standard_D16_v3
- GCP - e2-standard-16

Note: This is applicable from the following build of Cloudera Data Catalog: 2.0.17: 2.0.17-b26.

There are two types of Profiler Services:

- Profiler Manager
- Profiler Scheduler

The Profiler Manager service consists of profiler administrators, metrics, and data discovery services. These three entities support HA. The HA feature supports Active-Active mode.

Important: The Profiler Scheduler service does not support the HA functionality.

How to launch the profiler cluster for VM based environments

On the **Search** page, select the data lake from which you want to launch the profiler cluster. Click the Get Started link to proceed.

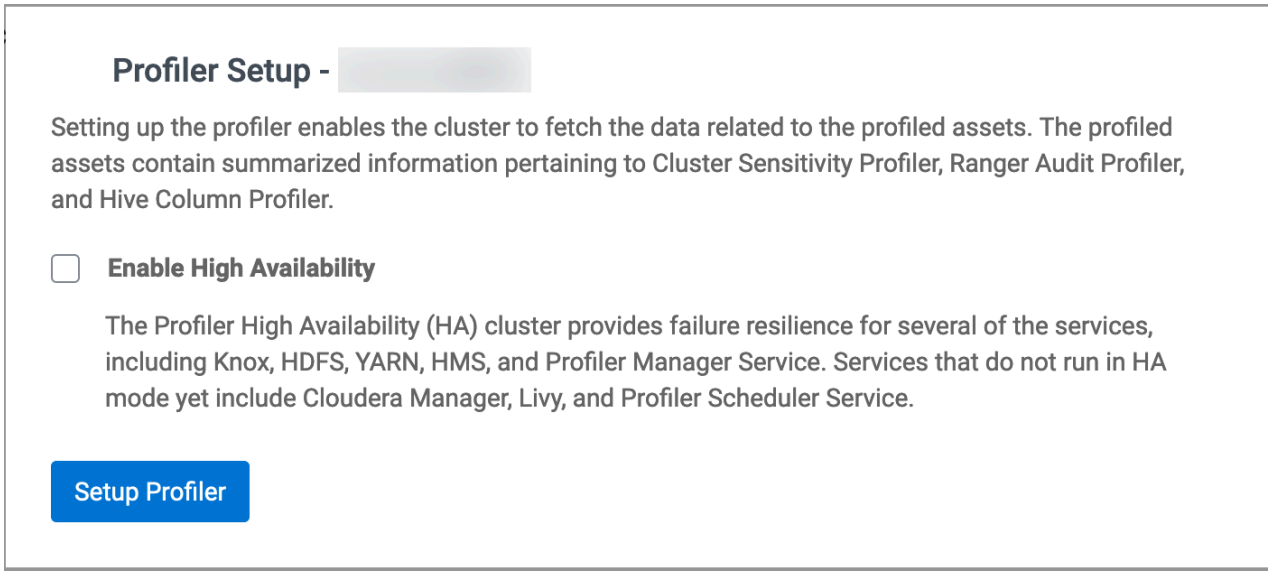

For setting up the profiler, you have the option to enable or disable the HA.

Note: The HA functionality is being supported only from Cloudera Runtime 7.2.10 release onwards. If you are using a Cloudera Runtime version below 7.2.10, you are not able to use the HA feature when launching the profiler services.

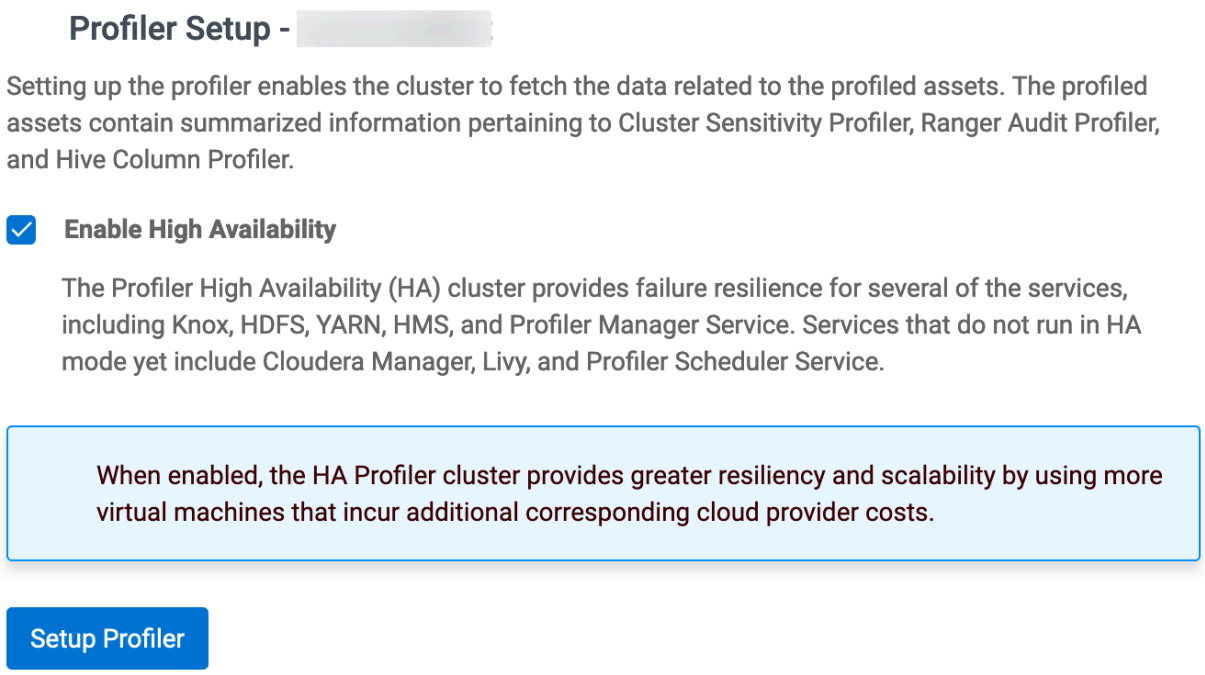

Once you enable HA and click Setup Profiler, Cloudera Data Catalog processes the request and the profiler creation is in progress.

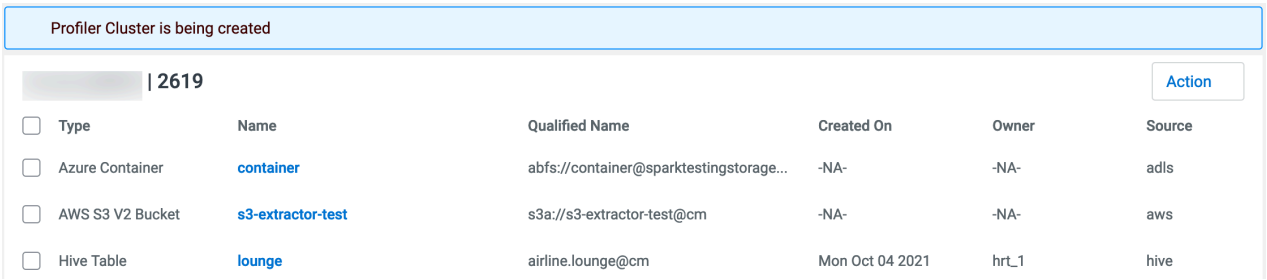

Later, a confirmation message appears that the profiler cluster is created.

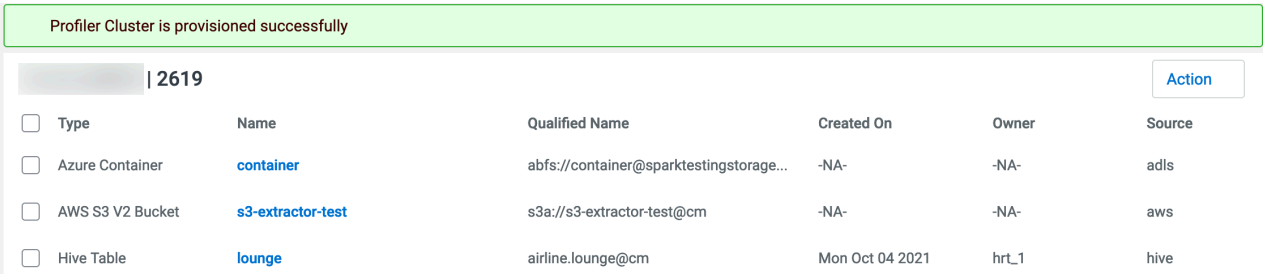

Next, you can verify the profiler cluster creation under Cloudera Management Console Environments Data Hubs pane.

The newly created profiler cluster looks like the following in Cloudera Management Console:

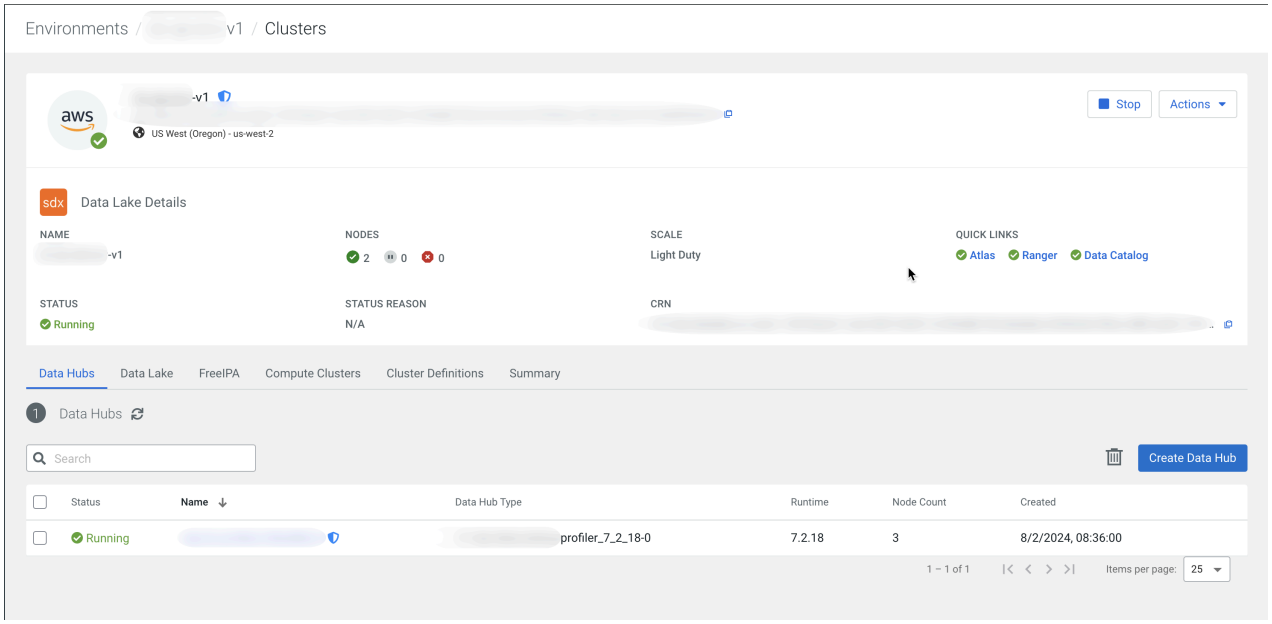

How to launch the profiler for Compute Cluster enabled environments

On the **Search** page, select the data lake from which you want to launch the profiler cluster. Click the Get Started link to proceed.

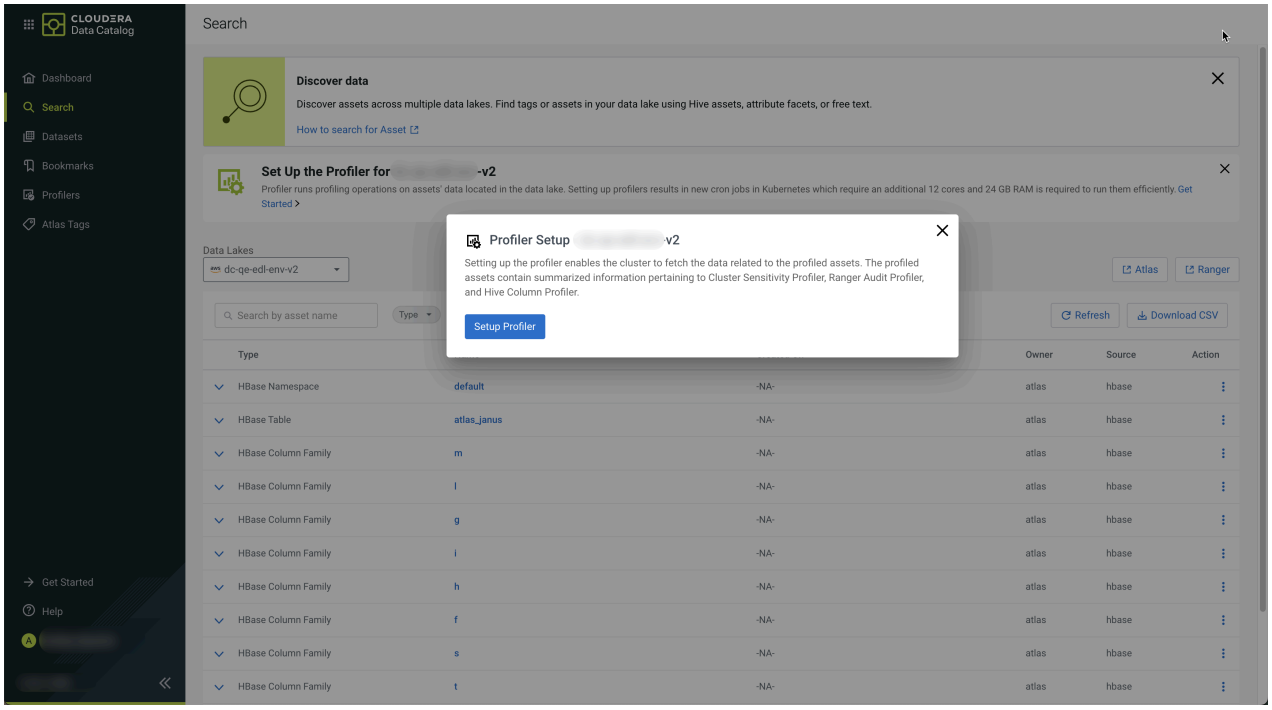

Click Setup Profiler, Cloudera Data Catalog processes the request and the profiler creation will start.

Next, you can verify that the profiler jobs are running under the Cloudera Management Console Environments Compute Clusters Node Groups pane.

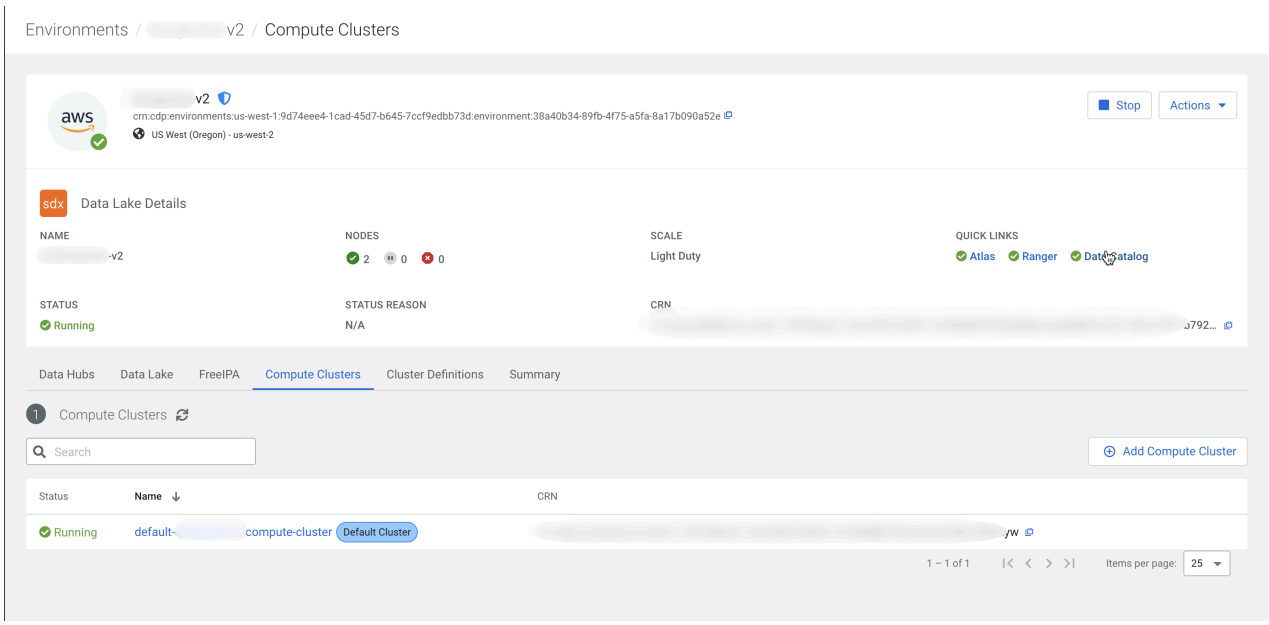

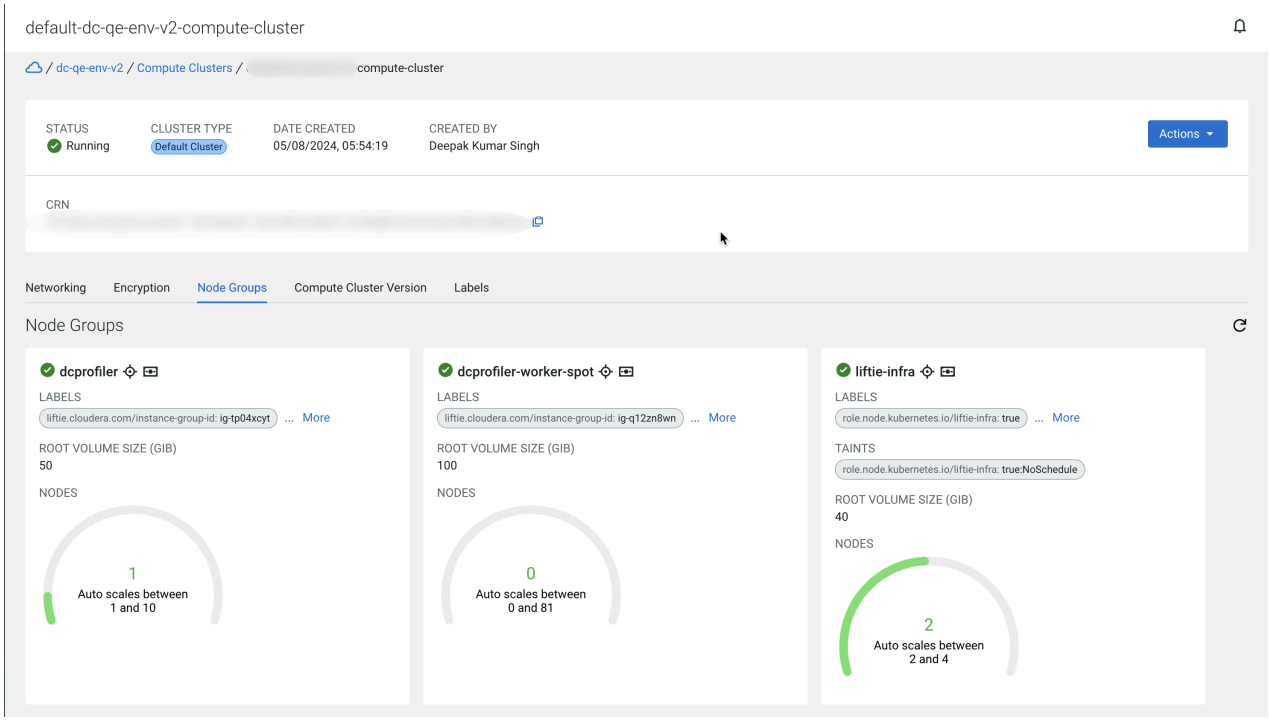

Related Information

Understanding the Cloudera Data Catalog Profiler
Understanding the Cluster Sensitivity Profiler
Understanding the Ranger Audit Profiler

Launching profilers using the command-line

Cloudera Data Catalog supports launching profilers using the Command-Line Interface (CLI) option.

The CLI is one executable and does not have any external dependencies. You can execute some operations in the Cloudera Data Catalog service using the Cloudera CLI commands.

Users must have valid permissions to launch profilers on a data lake.

For more information about the access details, see Prerequisites to access Cloudera Data Catalog.

Prerequisites

You must have the following entitlement granted to use this feature:

DATA_CATALOG_ENABLE_API_SERVICE

For more information about the Cloudera command-line interface and setting up the same, see Cloudera CLI.

The Cloudera Data Catalog CLI

In your Cloudera CLI environment, enter the following command to get started in the CLI mode.

cdp datacatalog --help

This command provides information about the available commands in Cloudera Data Catalog for Cloudera Public Cloud 7.2.18. and earlier versions.

The output is displayed as:

```
NAME
datacatalog
DESCRIPTION
Cloudera Data Catalog Service is a web service, using this service user can
 execute operations like launching profilers in Data Catalog.
AVAILABLE SUBCOMMANDS
launch-profilers
```

You get additional information about this command by using:

cdp datacatalog launch-profilers --help

```
NAME
launch-profilers -
DESCRIPTION
Launches DataCatalog profilers in a given datalake.
```

```
SYNOPSIS
 launch-profilers
 --datalake <value>
 [--cli-input-json <value>]
 [--generate-cli-skeleton]
OPTIONS
 --datalake (string)
The CRN of the Datalake.
--cli-input-json
(string) Performs service operation based on the JSON string provided. The
JSON string follows the format provided by --generate-cli-skeleton. If other
 arguments are provided on the command line, the CLI values will override th
e JSON-provided values.
--generate-cli-skeleton
(boolean) Prints a sample input JSON to standard output. Note the specified
operation is not run if this argument is specified. The sample input can be
used as an argument for --cli-input-json.
```

```
OUTPUT
.
datahubCluster -> (object)
 Information about a cluster.
clusterName -> (string)
 The name of the cluster.
 crn -> (string)
 The CRN of the cluster.
 creationDate -> (datetime)
```

```
 The date when the cluster was created.
 clusterStatus -> (string)
 The status of the cluster.
 nodeCount -> (integer)
 The cluster node count.
 workloadType -> (string)
 The workload type for the cluster. cloudPlatform -> (string) The cloud plat
form.
 imageDetails -> (object)
The details of the image used for cluster instances.
 name -> (string)
The name of the image used for cluster instances.
 id -> (string)
The ID of the image used for cluster instances.
This is internally generated by the cloud provider to Uniquely identify the
 image.
 catalogUrl -> (string)
 The image catalog URL.
 catalogName -> (string)
 The image catalog name.
 environmentCrn -> (string)
 The CRN of the environment.
 credentialCrn -> (string)
 The CRN of the credential.
 datalakeCrn -> (string)
 The CRN of the attached datalake.
 clusterTemplateCrn -> (string)
The CRN of the cluster template used for the cluster creation.
```

Launching the profiler

You can use the following CLI command to launch the data profiler:

```
cdp datacatalog launch-profilers --datalake [***DATALAKE CRN***]
```

Example:

```
cdp datacatalog launch-profilers --datalake crn:cdp:data
lake:DATACENTERNAME:c*****b-ccce-4**d-a**1-8********8:datalake:4*****5e-c**
1-4**2-8**e-1********2
{
    "success": true
}
```

Deleting profilers

In VM-based environments, deleting the profiler cluster (or in Compute Cluster enabled environments deleting the profiler jobs) removes all the Custom Sensitivity Profiler rules and other updates to the specific cluster. It could also cause loss of data specific to currently applied rules on the deleted profiler cluster.

About this task

To overcome this situation, when you decide to delete the profiler cluster or (in VM-based environments) the profiler jobs by Compute Cluster enabled environments, there is a provision to retain the status of the Custom Sensitivity Profiler rules:

- If your profiler cluster or profiler jobs have rules that are not changed or updated, you can directly delete them or the profiler cluster.

- If the rules were modified or updated, you have an option to download the modified rules along with deletion. The modified rules consist of the suspended system rules and the deployed custom rules. Using the downloaded rules, you can manually add or modify them to your newly added profiler jobs or the profiler cluster.

Note:

- In a Compute Cluster enabled environment, when you delete the scheduled jobs, the associated Kubernetes cron job object is deleted from the Kubernetes cluster.
- The associated data of the profilers from the Cloudera Management Console database is also deleted for the specified data lake.

Procedure

1. On the **Search** page, select the data lake from the drop-down.
2. Click Delete Profiler.

3. If you agree, select the warning message I understand this action cannot be undone.

Figure 1: Deleting a profiler in a VM-based environment

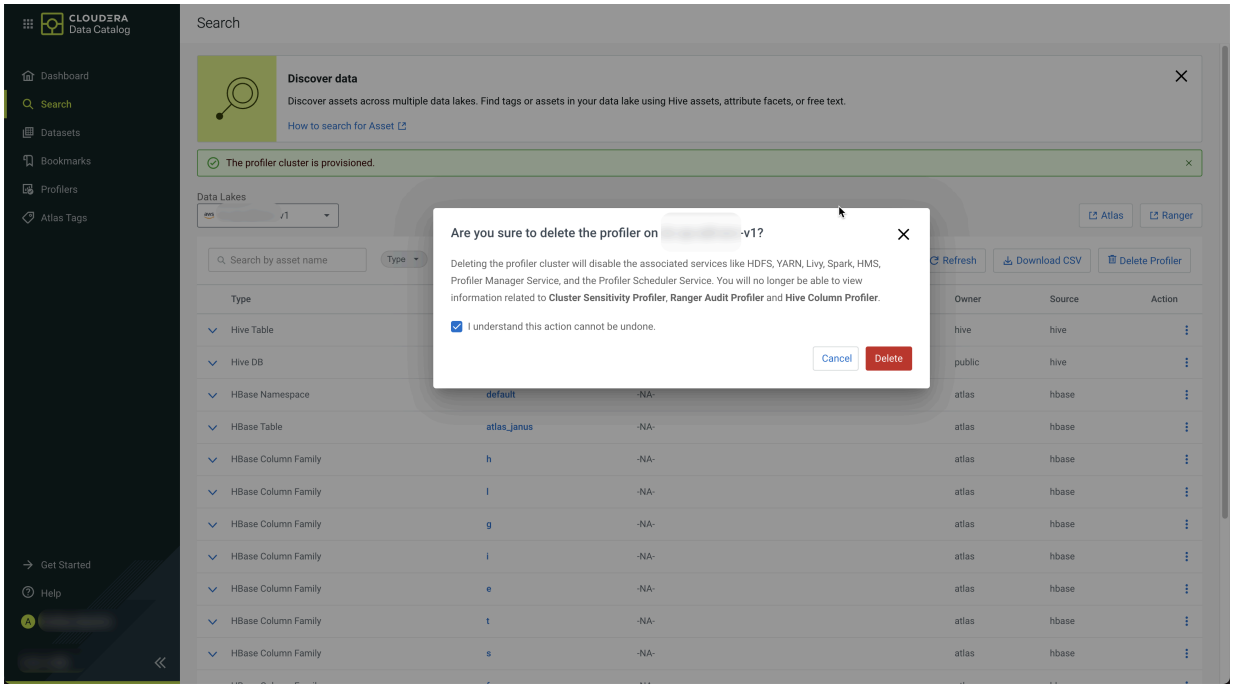

Figure 2: Deleting a profiler in a Compute Cluster enabled environment

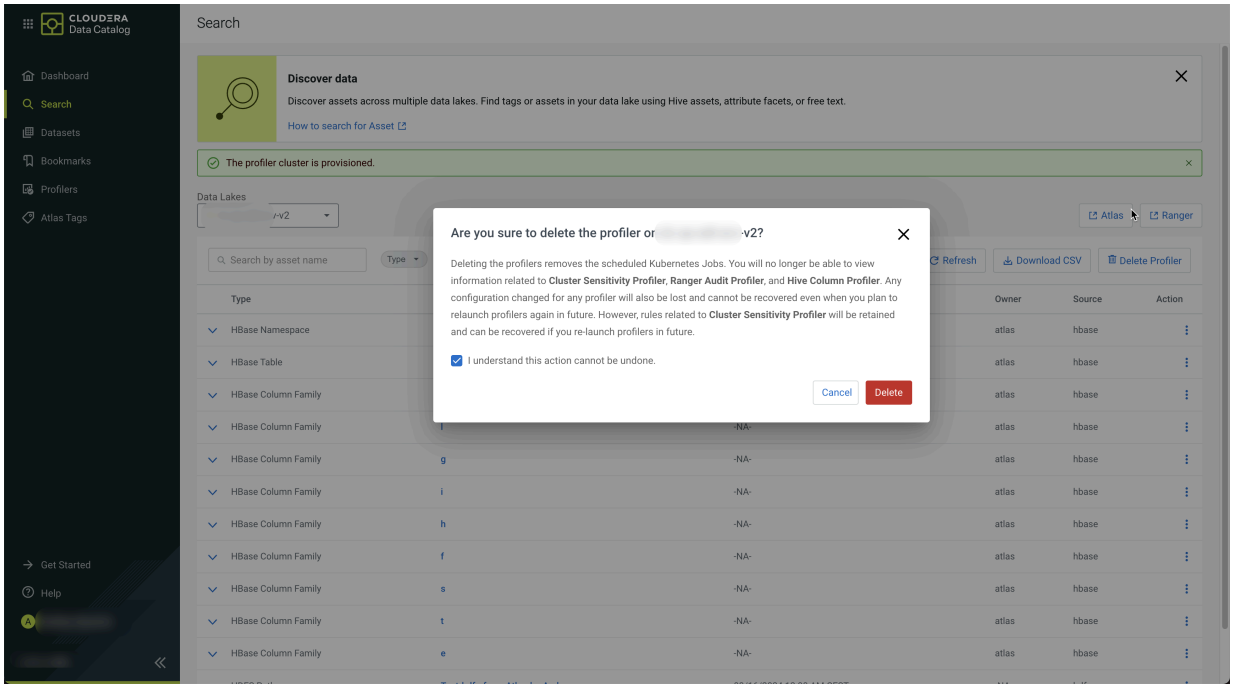

4. Click Delete.

   The application displays the following message.

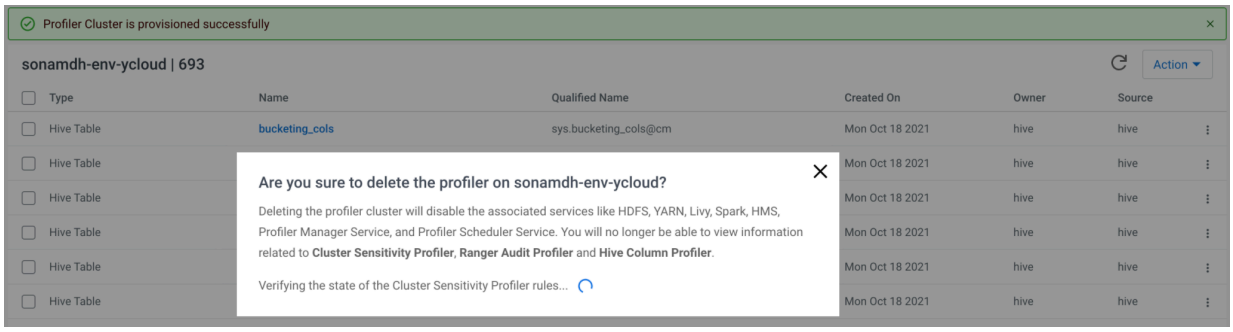

Note: When you launch Cloudera Data Catalog in Cloudera Runtime version 7.2.14, and later if the profiler cluster is deleted, the following message is displayed.

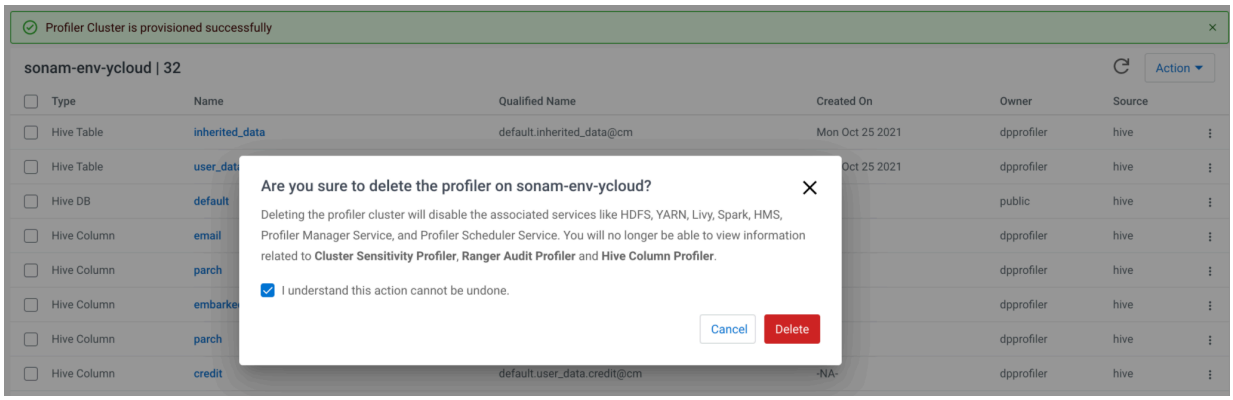

Additionally, note that you can delete the profiler cluster in these situations, when:

- Profiler cluster is up and running
- Profiler cluster is created but stopped
- Profiler cluster creation failed but is registered with the data lake
- Profiler cluster is down and inaccessible

Note: In VM-based environments, if the profiler cluster is not registered with the data lake, Cloudera Data Catalog cannot locate or trace the profiler cluster. You have to delete the profiler cluster from the Cloudera Data Hub page (Cloudera Management Console).

The profiler cluster is deleted successfully.

On-Demand Profilers

You can use On-Demand Profilers to profile specific assets without depending on the cron-based scheduling of profilers jobs. The On-Demand Profiler option is available in the Asset Details of the selected asset.

The following image shows the **Asset Details** page of an asset. You can run an On-Demand Profiler for Hive Column Profiler and Cluster Sensitivity Profiler by clicking on the appropriate Run button. The next scheduled run provides details about the next scheduled profiling for the respective profilers.

Note: You can use the On-Demand Profiler feature to profile both external and managed tables.

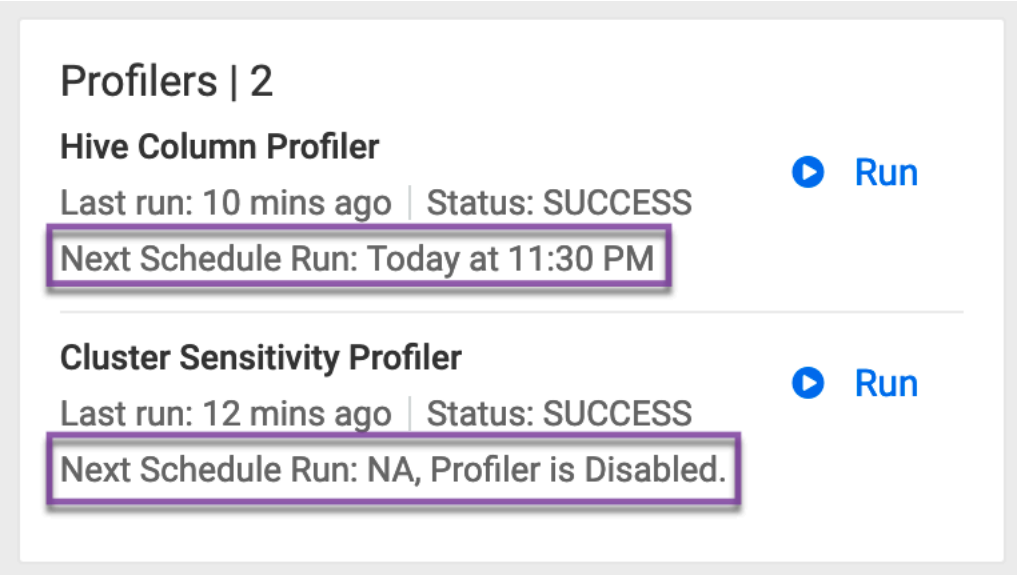

Note: In Compute Cluster-enabled environments, Iceberg tables can be also profiled with the On-Demand Profiler.

Profiling table data in non-default buckets

In VM-based environments, you must configure a parameter in Profiler Scheduler in your instance to profile table data in non-default buckets.

Procedure

1. In Cloudera Data Catalog, make a note of your environment's name in the **Search** menu.
2. Go Cloudera Management Console Environments
3. Search for your environment, then switch to the **Data hubs** tab.
4. Open you Cloudera Data Hub by clicking its name.
5. Open the CM URL under **Cloudera Manager Info**.
6. In Cloudera Manager go to Configuration Configuration Search .
7. Search for the term Profiler Scheduler Spark conf.
   The **Profiler Scheduler Spark conf** configuration snippet appears.

8. Add spark.yarn.access.hadoopFileSystems=s3a://default-bucket,s3a://bucket-1,s3a://bucket-2 to **Profiler Scheduler Spark conf** to enable profiling for bucket-1 and bucket-2 non-default buckets.

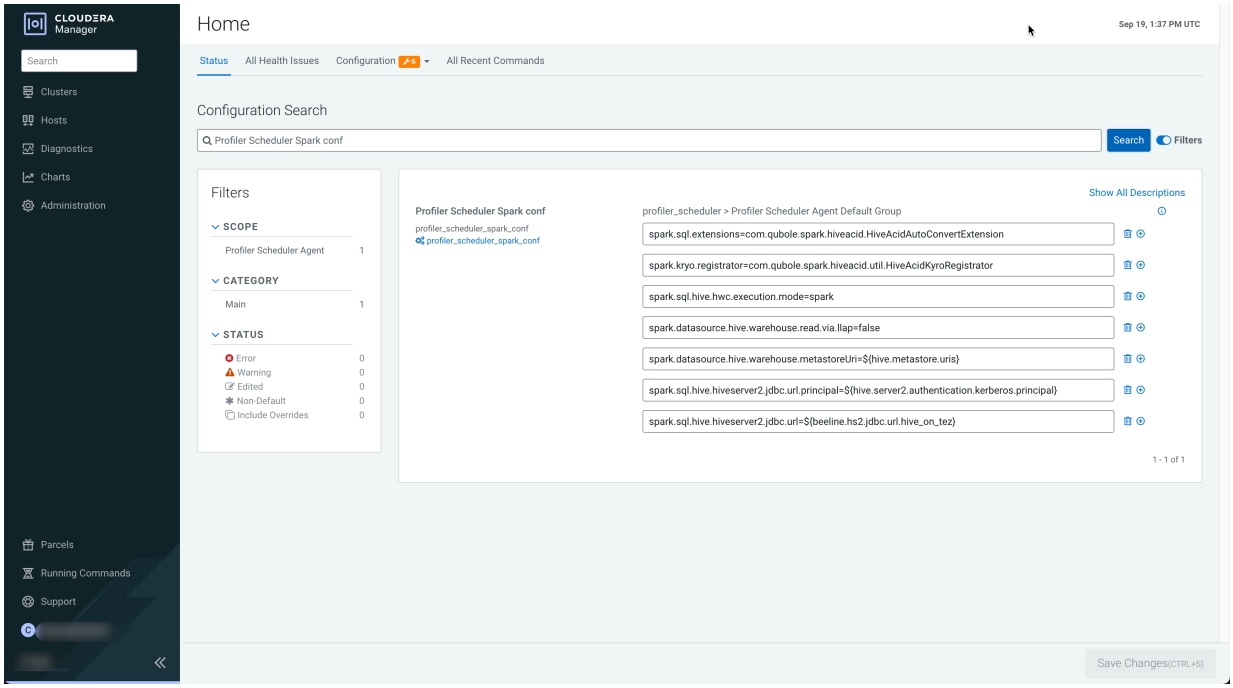

Tracking profiler jobs

Use the Profilers > Jobs page for tracking respective profiler jobs.

Under Profilers Jobs , you can have an overview of your started profiler jobs. By using the D, W, M filters, you can go back up to a day, week or a moth, to see your previous jobs. Use this page to quickly check if your profiler jobs are failing.

Figure 3: Profiling jobs in a Compute Cluster enabled environment

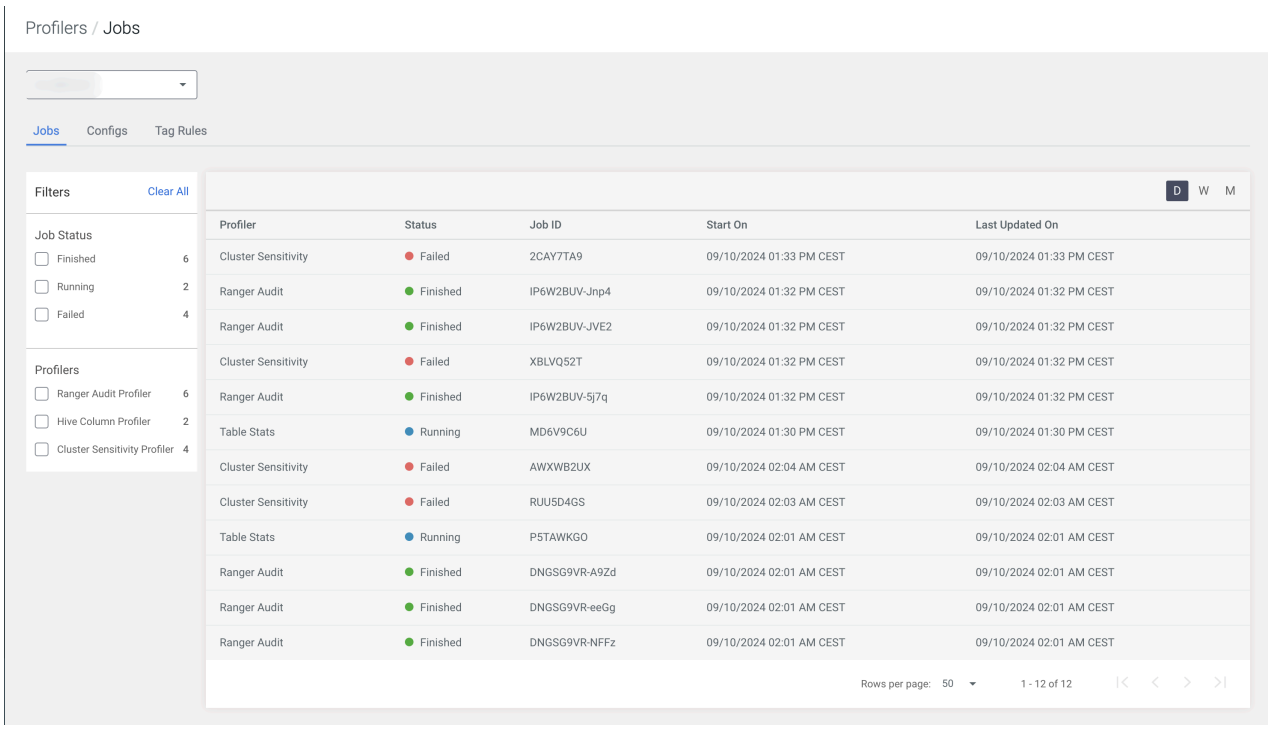

In VM-based environments, Profilers Jobs can show you the current profiling **Stage** based on the relevant service used:

Figure 4: Profiling jobs in a VM-based environment

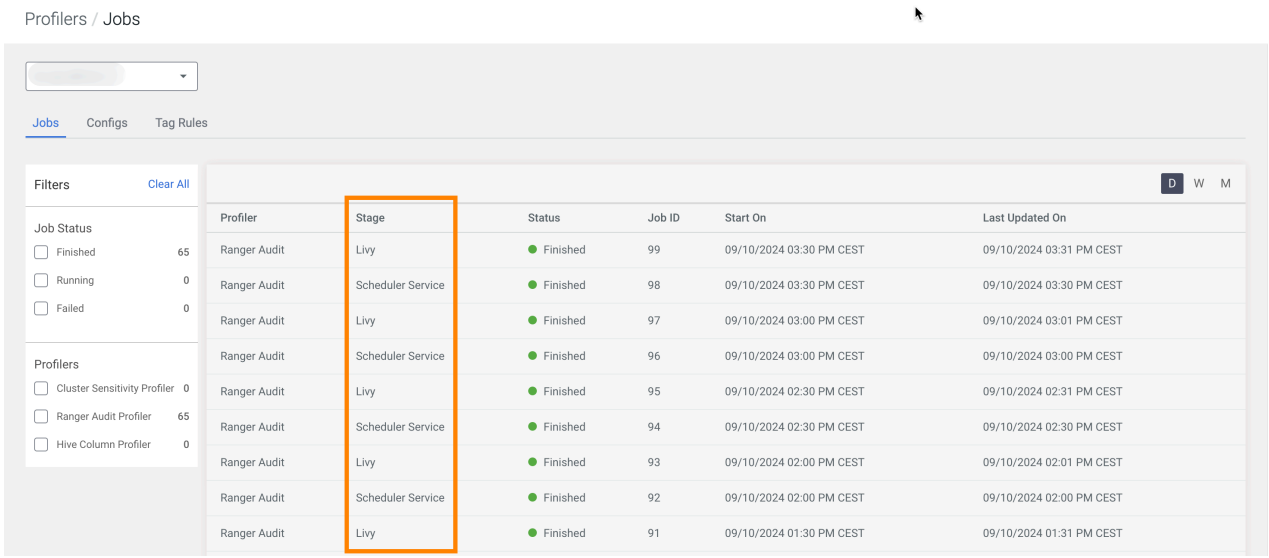

For each profiler job, you can view the details about:

- **Profiler** type
- **Stage** (for VM-based environments)
- Job **Status**
- **Job ID**
- **Start Time**
- **Last Update On** time

Using this data can help you to troubleshoot failed jobs or even understand how the jobs were profiled and other pertinent information that can help you to manage your profiled assets.

In VM-based environments, profiler job runs ins the following phases:

- Scheduler Service - The part of Profiler Admin which queues the profiler requests.
- Livy - This service is managed by YARN and where the actual asset profiling takes place.
- Metrics Service - Reads the profiled data files and publishes them.

Note: More than one occurrence of Scheduler Service or Livy indicates that there could be more assets to be profiled. For example, if an HBase schedule has about 80 assets to be profiled, the first 50 assets would be profiled in the first Livy batch and the other assets get profiled in the next batch.

In case of Ranger Audit profiling, there could be a “NA” status for the total number of assets profiled. It indicates that the auditing that happens is dependent on the Ranger policies. In other words, the Ranger policies are actually profiled and not the assets.

Related Information

Understanding the Cloudera Data Catalog Profiler
Understanding the Cluster Sensitivity Profiler
Understanding the Ranger Audit Profiler

Viewing profiler configurations

You can monitor the last status of individual profilers by viewing them in Profiler > Configs. Also, you can change their resources, sensitivity and scheduling.

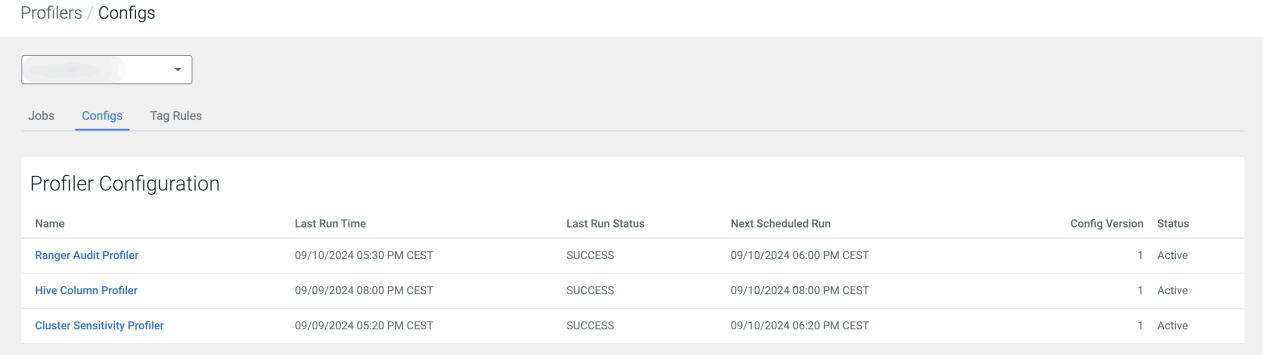

Name	Last Run Time	Last Run Status	Next Scheduled Run	Config Version	Status
Ranger Audit Profiler	09/10/2024 05:30 PM CEST	SUCCESS	09/10/2024 06:00 PM CEST	1	Active
Hive Column Profiler	09/09/2024 08:00 PM CEST	SUCCESS	09/10/2024 08:00 PM CEST	1	Active
Cluster Sensitivity Profiler	09/09/2024 05:20 PM CEST	SUCCESS	09/10/2024 06:20 PM CEST	1	Active

Monitoring the profiler configurations has the following uses:

- Verify which profilers are active or inactive.
- Verify the status of the profiler runs.
- View the last run time and status and the next scheduled run.

Note: You can also filter your profilers by job status, type for the last day, week and month.

Related Information

Understanding the Cloudera Data Catalog Profiler
Understanding the Cluster Sensitivity Profiler
Understanding the Ranger Audit Profiler

Ranger Audit Profiler configuration

In addition to the generic configuration, there are additional parameters for the Ranger Audit Profiler that can optionally be edited.

Procedure

1. Go to Profilers Configs .
2. Select your data lake.
3. Select Ranger Audit Profiler.
   The **Detail** page is displayed.
4. Use the toggle button

   to enable or disable the profiler.
5. Select a schedule to run the profiler. This is implemented as a quartz cron expression.
6. Continue with the resource settings.

For VM-based environment

a. In **Advanced Options**, set the following:

- Number of Executors - Enter the number of executors to launch for running this profiler.
- Executor Cores - Enter the number of cores to be used for each executor.
- Executor Memory - Enter the amount of memory in GB to be used per executor process.
- Driver Cores - Enter the number of cores to be used for the driver process.
- Driver Memory - Enter the memory to be used for the driver processes.

Note: For more information, see Configuring SPARK on YARN Applications and Tuning Resource Allocation.

For Compute Cluster enabled environment

a. In **Pod Configurations**, set the Kubernetes job resources.

Pod configurations specify the resources that would be allocated to a pod when the profiler job starts to run. As all profilers are submitted as Kubernetes jobs, you must decide if you want to add or reduce resources to handle workload of various sizes.

- **Pod CPU Limit**: Indicates the maximum number of cores that can be allocated to a Pod. The accepted values range from one through eight.
- **Pod CPU Requirement**: This is the minimum number of CPUs that will be allocated to a Pod when its provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit. The accepted values range from one through eight.
- **Pod Memory Limit**: Maximum amount of memory can be allocated to a Pod. The accepted values range from 1 through 256.
- **Pod Memory Requirement**: This is the minimum amount of RAM that will be allocated to a Pod when it is provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a

container is not allowed to use more than its resource limit. The accepted values range from 1 through 256.

b. In **Executor Configurations**, update the following:

- **Number of workers**: Indicates the number of processes that are used by the distributed computing framework. The accepted values range from one through eight.
- **Number of threads per worker**: Indicates the number of threads used by each worker to complete the job. The accepted values range from one through eight.
- **Worker Memory limit in GB**: To avoid over utilization of memory, this parameter forces an upper threshold memory usage for a given worker. For example, if you have a 8 GB Pod and 4 threads, the value of this parameter must be 2 GB. The accepted values range from one through four.

Executor configurations are the runtime configurations. These configuration must be changed if you are changing the pod configurations and when there is a requirement for additional compute power.

7. Click Save to apply the configuration changes to the selected profiler.

Cluster Sensitivity Profiler configuration

In addition to the generic configuration, there are additional parameters for the Cluster Sensitivity Profiler that can optionally be edited.

Procedure

1. Go to Profilers Configs .

2. Select your data lake.

3. Select Cluster Sensitivity Profiler.
The **Detail** page is displayed which contains the following sections:

4. Use the toggle button

to enable or disable the profiler.

5. Select a schedule to run the profiler. This is implemented as a quartz cron expression.
For more information, see Understanding the Cron Expression generator on page 37.

6. Select Last Run Check and set a period if needed.

Note:

The Last Run Check enables profilers to avoid profiling the same asset on each scheduled run.

If you have scheduled a cron job, for example set to start in about an hour, and have enabled the Last Run Check configuration for two days, this setup ensures that the job scheduler filters out any asset which was already profiled in the last two days.

If the Last Run Check configuration is disabled, assets will be picked up for profiling as per the job cron schedule, honoring the asset filter rules.

7. Set the sample settings for VM-based environments:

a. Select the **Sample Data Size**.

1. From the drop down, select the type of sample data size.

2. Enter the value based on the previously selected type.

Note: In Compute Cluster enabled environments, skip to step 8 on page 34.

8. Continue with the resource settings.

For VM-based environment

a. In **Advanced Options**, set the following:

- Number of Executors - Enter the number of executors to launch for running this profiler.
- Executor Cores - Enter the number of cores to be used for each executor.
- Executor Memory - Enter the amount of memory in GB to be used per executor process.
- Driver Cores - Enter the number of cores to be used for the driver process.
- Driver Memory - Enter the memory to be used for the driver processes.

Note: For more information, see Configuring SPARK on YARN Applications and Tuning Resource Allocation.

For Compute Cluster enabled environment

a. In **Pod Configurations**, update the following:

- **Pod CPU Limit**: Indicates the maximum number of cores that can be allocated to a Pod. The accepted values range from one through eight.
- **Pod CPU Requirement**: This is the minimum number of CPUs that will be allocated to a Pod when its provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit. The accepted values range from one through eight.
- **Pod Memory Limit**: Maximum amount of memory can be allocated to a Pod. The accepted values range from 1 through 256.
- **Pod Memory Requirement**: This is the minimum amount of RAM that will be allocated to a Pod when it is provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit. The accepted values range from 1 through 256.

b. In **Executor Configurations**, update the following:

- **Number of workers**: Indicates the number of processes that are used by the distributed computing framework. The accepted values range from one through eight.
- **Number of threads per worker**: Indicates the number of threads used by each worker to complete the job. The accepted values range from one through eight.
- **Worker Memory limit in GB**: To avoid over utilization of memory, this parameter forces an upper threshold memory usage for a given worker. For example, if you have a 8 GB Pod and 4 threads, the value of this parameter must be 2 GB. The accepted values range from one through four.

Executor configurations are the runtime configurations. These configuration must be changed if you are changing the pod configurations and when there is a requirement for additional compute power.

9. Add **Asset Filter Rules** as needed to customize the selection and deselection of assets which the profiler profiles.
   a) Set your **Deny List** and **Allow-list**.

      The profiler will skip profiling assets that meet any criteria in the **Deny List** and will include assets that meet any criteria in the **Allow List**.

      1. Select the **Deny-list** or **Allow List** tab.
      2. Click Add New to include rules.
      3. Select the key from the drop-down list. You can select from the following:
         - Database name
         - Asset name
         - Asset owner
         - Path to the asset
         - Created date
      4. Select the operator from the drop-down list. Depending on the keys selected, you can select an operator such as equals, contains. For example, you can select the name of assets that contain a particular string.
      5. Enter the value corresponding to the key. For example, you can enter a string as mentioned in the previous example.
      6. Click Done. Once rule is added, you can toggle the state of the new rule to enable it or disable it as needed.
10. Click Save to apply the configuration changes to the selected profiler.

Related Information

Understanding the Cloudera Data Catalog Profiler
Understanding the Cluster Sensitivity Profiler

Hive Column Profiler configuration

In addition to the generic configuration, there are additional parameters for the Hive Column Profiler that can optionally be edited.

Procedure

1. Go to Profilers Configs .
2. Select your data lake.
3. Select Hive Column Profiler.
   The **Detail** page is displayed.
4. Use the toggle button

to enable or disable the profiler.
5. Select a schedule to run the profiler. This is implemented as a quartz cron expression.

   For more information, see Understanding the Cron Expression generator on page 37.
6. Select Last Run Check and set a period if needed.

   Note:

   The Last Run Check enables profilers to avoid profiling the same asset on each scheduled run.

   If you have scheduled a cron job, for example set to start in about an hour, and have enabled the Last Run Check configuration for two days, this setup ensures that the job scheduler filters out any asset which was already profiled in the last two days.

   If the Last Run Check configuration is disabled, assets will be picked up for profiling as per the job cron schedule, honoring the asset filter rules.

7. Set the sample settings:
   a. Select the **Sample Data Size**.
      1. From the drop down, select the type of sample data size.
      2. Enter the value based on the previously selected type.

Note: In Compute Cluster enabled environments, skip to step 8 on page 36.

8. Continue with the resource settings.

For VM-based environment

a. In **Advanced Options**, set the following:
   - Number of Executors - Enter the number of executors to launch for running this profiler.
   - Executor Cores - Enter the number of cores to be used for each executor.
   - Executor Memory - Enter the amount of memory in GB to be used per executor process.
   - Driver Cores - Enter the number of cores to be used for the driver process.
   - Driver Memory - Enter the memory to be used for the driver processes.

   Note: For more information, see Configuring SPARK on YARN Applications and Tuning Resource Allocation.

For Compute Cluster enabled environment

a. In **Pod Configurations**, update the following:
   - **Pod CPU Limit**: Indicates the maximum number of cores that can be allocated to a Pod. The accepted values range from one through eight.
   - **Pod CPU Requirement**: This is the minimum number of CPUs that will be allocated to a Pod when its provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit. The accepted values range from one through eight.
   - **Pod Memory Limit**: Maximum amount of memory can be allocated to a Pod. The accepted values range from 1 through 256.
   - **Pod Memory Requirement**: This is the minimum amount of RAM that will be allocated to a Pod when it is provisioned. If the node where a Pod is running has enough resources available, it is possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit. The accepted values range from 1 through 256.

b. In **Executor Configurations**, update the following:
   - **Number of workers**: Indicates the number of processes that are used by the distributed computing framework. The accepted values range from one through eight.
   - **Number of threads per worker**: Indicates the number of threads used by each worker to complete the job. The accepted values range from one through eight.
   - **Worker Memory limit in GB**: To avoid over utilization of memory, this parameter forces an upper threshold memory usage for a given worker. For example, if you have a 8 GB Pod and 4 threads, the value of this parameter must be 2 GB. The accepted values range from one through four.

   Executor configurations are the runtime configurations. These configuration must be changed if you are changing the pod configurations and when there is a requirement for additional compute power.

9. Add **Asset Filter Rules** as needed to customize the selection and deselection of assets which the profiler profiles.
   a) Set your **Deny List** and **Allow-list**.

      The profiler will skip profiling assets that meet any criteria in the **Deny List** and will include assets that meet any criteria in the **Allow List**.

      1. Select the **Deny-list** or **Allow List** tab.
      2. Click Add New to include rules.
      3. Select the key from the drop-down list. You can select from the following:
         - Database name
         - Asset name
         - Asset owner
         - Path to the asset
         - Created date
      4. Select the operator from the drop-down list. Depending on the keys selected, you can select an operator such as equals, contains. For example, you can select the name of assets that contain a particular string.
      5. Enter the value corresponding to the key. For example, you can enter a string as mentioned in the previous example.
      6. Click Done. Once rule is added, you can toggle the state of the new rule to enable it or disable it as needed.
10. Click Save to apply the configuration changes to the selected profiler.

Understanding the Cron Expression generator

In the Profiler > Configs > Detail page, a cron expression defines when the profiler schedule executes and visualizes the next execution dates of your profiling jobs.

The Unix (in Compute Cluster enabled environments) and quartz (in VM-based environments) cron expression uses the following typical format:

Each * in the cron represents a unique value.

For VM-based environments

The ? character is used for undefined day of the month and the day of the week.

Schedule: * * * * * ? *

For example, consider a cron with the following values:

1 2 3 2 5 ? 2021

This cron expression is scheduled to run the profiler job at: 03:02:01am, on the 2nd day, in May, in 2021.

Note: The ? character is a replacement for the "day-of-the-week". It is not specified on which day of the week the job has to run.

For Compute Cluster enabled environments

Cron Expression: 0 18 * * *

In this format the * characters represent the following units:Minute hour day(month) month day(week)

For example, consider a cron with the following values:

CRON Expression: 30 10 15 5 *

This cron expression is scheduled to run the profiler job at: “At 10:30 on 15th day-of-month in May.”

Note: The * character is a replacement for the "day-of-the-week". It is not specified on which day of the week the job has to run.

Consider another example:

30 10 * 5 7

This cron expression is scheduled to run the profiler job at: “At 10:30 on Sunday in May”.

Note: The * character is a replacement for the "day-of the month". It is not specified on which day of the month the job has to run.

You can change the value of cron as and when it is required depending on how you want to schedule your profiler job.

Backing up and restoring the profiler database

Using certain scripts that can be executed by the root users, you can back up of the profiler databases. Later, if you want to delete the existing Cloudera Data Hub cluster and launch a new cluster, you will have an option to restore the old data.

Important: Backing up and restoring the profiler database is only available in VM-based environments.

Cloudera Data Catalog includes profiler services that run data profiling operations on data that is located in multiple data lakes. In VM-based environments, the profiler services run on a Cloudera Data Hub cluster. When you delete the Cloudera Data Hub cluster, the profiled data and the user configuration information stored in the local databases are lost.

Profiler clusters run on the Cloudera Data Hub cluster using embedded databases:

- profiler_agent
- profiler_metrics

Note: If you download the modified Cluster Sensitivity Profiler rules before deleting the profiler cluster, and later when you create a new profiler cluster, you can restore the state of the rules manually. If the system rules are part of the downloaded files, you must Suspend those rules. If custom rules are part of the downloaded files, you must deploy those rules. This is applicable if the profiler cluster has Cloudera Runtime below 7.2.14 version.

About the back up script

The Backup and Restore script can be used only on Amazon Web Services, Microsoft Azure, and Google Cloud Platform clusters where they support cloud storage.

Scenarios for using the script

- When you upgrade the data lake cluster and want to preserve profiler data in the Cloudera Data Hub cluster.
- When you want to delete the Cloudera Data Hub cluster but preserve the profiler data.
- When you want to relaunch the profiler and access the older processed data.
- **Note:** For users using Cloudera Data Catalog on Cloudera Runtime 7.2.14 version, note the following:
  - No user action or manual intervention needed after the upgrading Cloudera Data Hub cluster to the 7.2.14 version.
  - Also, as an example use case scenario, in case a new profiler cluster is launched that contains Custom Sensitivity Profiler tags and which is deleted and relaunched later, the changes are retained and no further action is required.
  - No user action is required to backup and restore the profiler data. The changes are automatically restored.

When upgrading a Cloudera Runtime version earlier than 7.2.11 to version 7.2.11:

Go to the following locations to pick up your scripts:

Back up

bash /opt/cloudera/parcels/PROFILER_MANAGER/profileradmin/scripts/users/backup_db.sh

Restore

bash /opt/cloudera/parcels/PROFILER_MANAGER/profileradmin/scripts/users/restore_db.sh

When upgrading a version below or equal to Cloudera Runtime version 7.2.11 to 7.2.12:

Go to the following locations to pick up your scripts:

Back up

bash /opt/cloudera/parcels/PROFILER_MANAGER/profileradmin/scripts/users/backup_db.sh

Restore

bash /opt/cloudera/parcels/CDH/lib/profiler_manager/profileradmin/scripts/users/restore_db.sh

When backing up and restoring for a cluster having the Cloudera Runtime version 7.2.12 and onwards:

Navigate to the following location to pick up your scrips:

Back up

bash /opt/cloudera/parcels/CDH/lib/profiler_manager/profileradmin/scripts/users/backup_db.sh

Restore

bash /opt/cloudera/parcels/CDH/lib/profiler_manager/profileradmin/scripts/users/restore_db.sh

Running the back up script

Running the profiler Backup and Restore script has multiple phases.

About this task

First, you need to back up your profiler database and next you can restore it.

Backing up the profiler database

1. Stop the Profiler Manager and Profiler Scheduler services from the Cloudera Manager instance of the Cloudera Data Hub cluster.
2. Use SSH to connect to the node where the Profiler Manager is installed as a root user.
3. Execute the backup_db.sh script:

   Attention: Users of Cloudera Runtime below 7.2.8 version should contact Cloudera Support.

   Note:

   - If the profiler cluster has Cloudera Runtime version 7.2.11 or earlier, you run the following command:

     ```
     bash /opt/cloudera/parcels/PROFILER_MANAGER/profileradmin/scripts/
     users/backup_db.sh
     ```

   - If the profiler cluster has the Cloudera Runtime version 7.2.12 or higher you must run the following command:

     ```
     bash /opt/cloudera/parcels/CDH/lib/profiler_manager/profileradmin/
     scripts/users/backup_db.sh
     ```

4. Delete the Profiler cluster.
5. Install a new version of Profiler cluster:
   - [Scenario-1] When the data lake upgrade is successfully completed.
   - [Scenario-2] When the user decides to launch a new version of the Profiler cluster.

Restoring the profiler database

1. Stop the Profiler Manager and Profiler Scheduler services from the Cloudera Manager instance of the Cloudera Data Hub cluster.
2. Use SSH to connect to the node where Profiler Manager is installed as a root user.
3. Execute the restore_db.sh script.

Attention: Users of Cloudera Runtime below 7.2.8 version should contact Cloudera Support.

Note:

- If the profiler cluster has the Cloudera Runtime version 7.2.11 or earlier, you must run the following command:

```
bash /opt/cloudera/parcels/PROFILER_MANAGER/profileradmin/scripts/
users/restore_db.sh
```

- If the profiler cluster having the Cloudera Runtime version 7.2.12 or higher, you must run the following command:

```
bash /opt/cloudera/parcels/CDH/lib/profiler_manager/profileradmin/
scripts/users/restore_db.sh
```

4. Start the Profiler Manager and Profiler Scheduler services from Cloudera Manager.

Note: When you upgrade the data lake cluster and a new version of profiler cluster is installed, the profiler configurations that have been modified by users in the older version is replaced with new values as the following:

- Schedule
- Last Run Check
- Number of Executors
- Executor Cores
- Executor Memory (in GB)
- Driver Core
- Driver Memory (in GB)

Enable or disable profilers

By default, profilers are enabled and run every 30 minutes. If you want to disable (or re-enable) a profiler, you can do this by selecting the appropriate profiler from the Configs tab.

Procedure

1. From Profilers Configs
2. Select the profiler to proceed further.

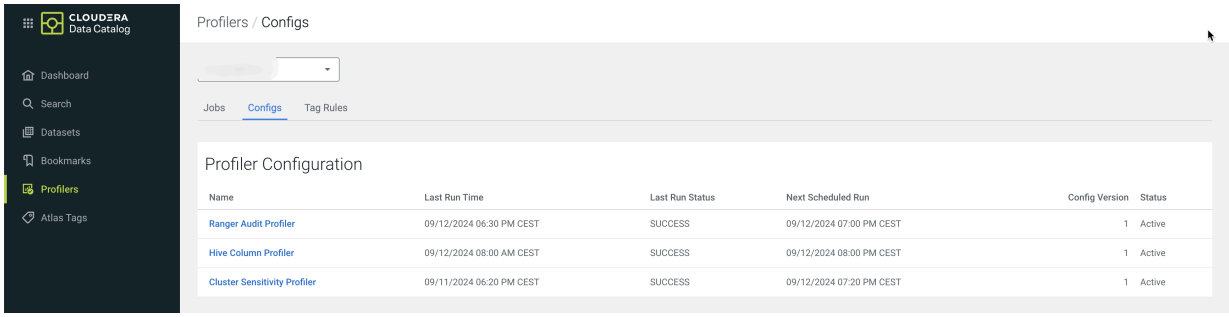

3. Switch the toggle to the desired state.

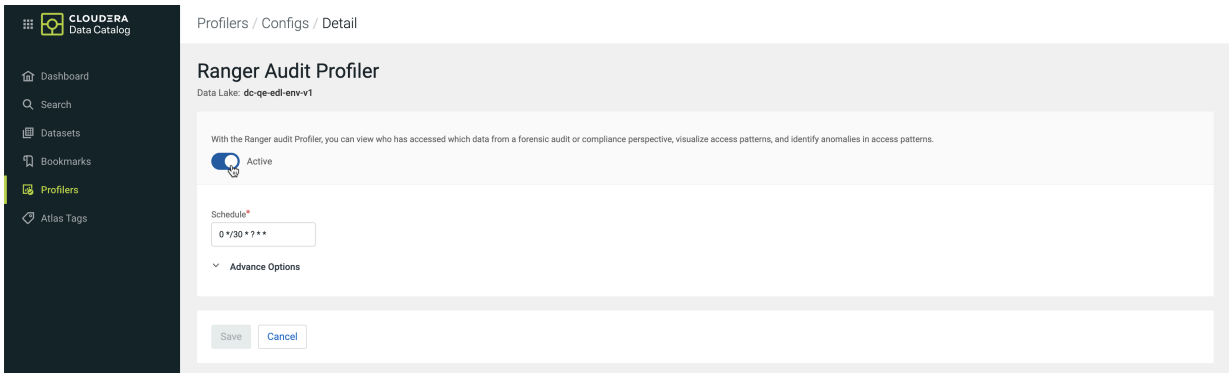

Profiler Tag Rules

You can use preconfigured tag rules or create new rules based on regular expressions to limit the number of assets to be profiled.

Rules are categorized into following groups:

- System Deployed: These are built-in rules that cannot be edited.
- Custom Deployed: Tag rules that you create and deploy on clusters after validation will appear under this category. Hover your mouse over the tag rules to deploy or suspend them as needed. You can also edit these tag rules.
- Custom Draft: You can create new tag rules and save them for later validation and deployment on clusters. Such rules appear under this category.

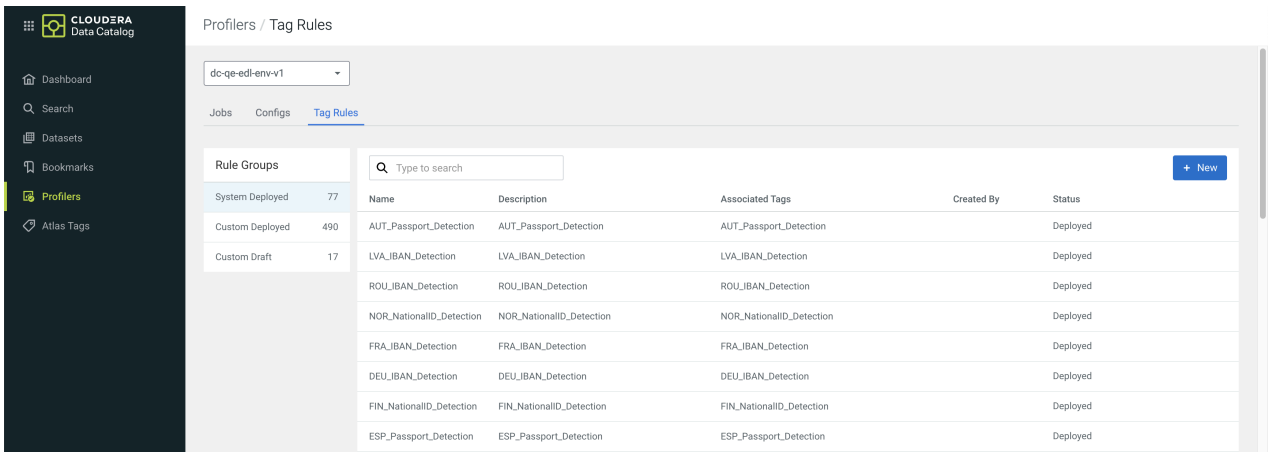

After creating your rule, you have to validate them with test data and, then Deploy them from **Custom Draft**.

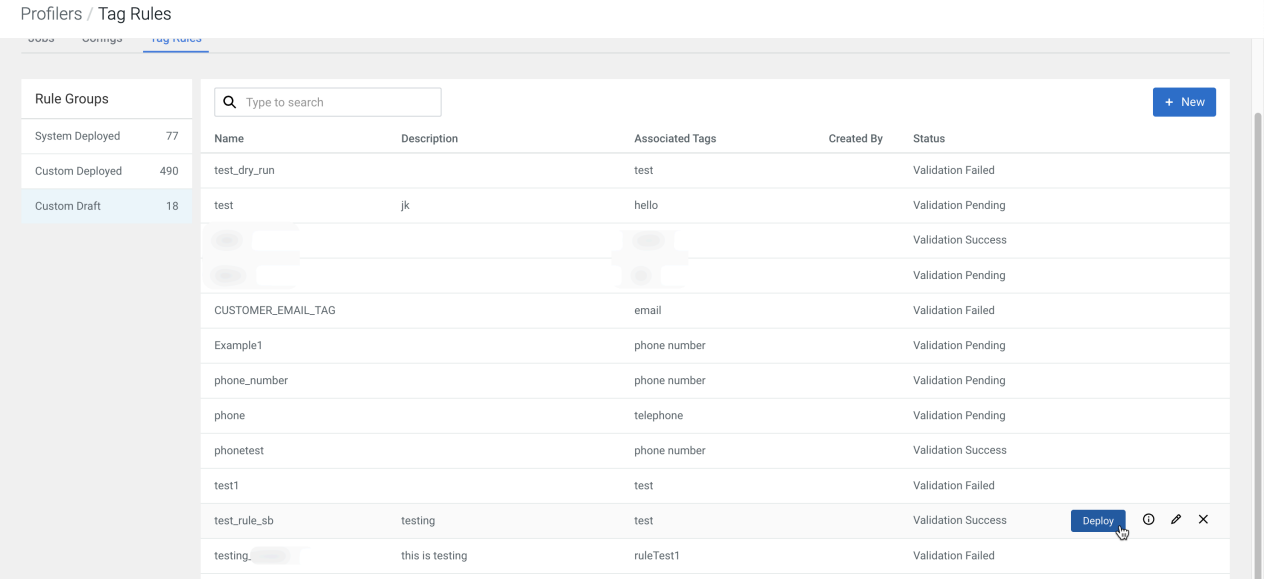

Note: Tag Rules can be temporarily suspended.

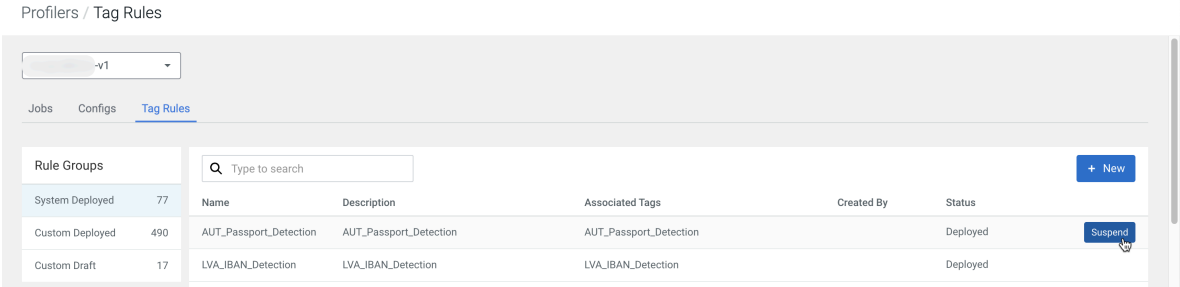

Tag management

From the Atlas Tags menu, you can create, modify, and delete any of the Apache Atlas classifications.

Atlas Tags allows the user to perform the following activities with a selected data lake for tag management:

- Selecting a data lake
- Searching for a tag
- Adding a tag
- Editing a tag
- Deleting a tag

You can create a new Cloudera Data Catalog tag in the **Atlas Tags**, which are synced to Atlas. Click Add Tag to open the **Create a new tag** page.

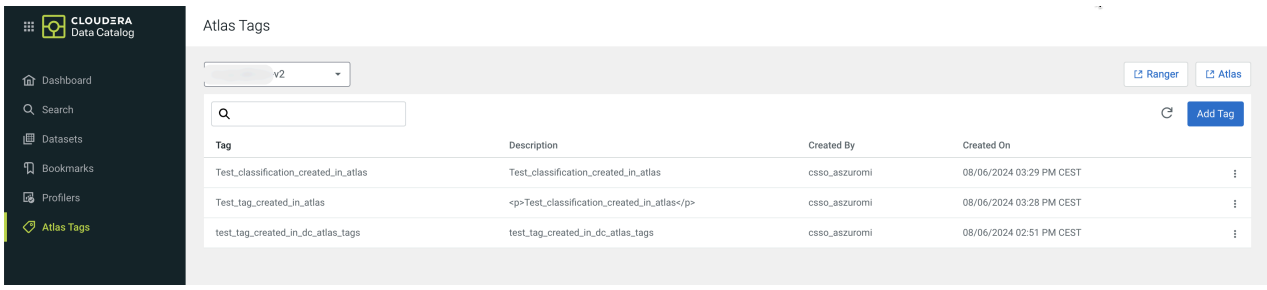

In **Create New Tag**, you can define the tag name, description and the "super-classification" from which the attributes are inherited for the sub-classification (or tag in Cloudera Data Catalog)

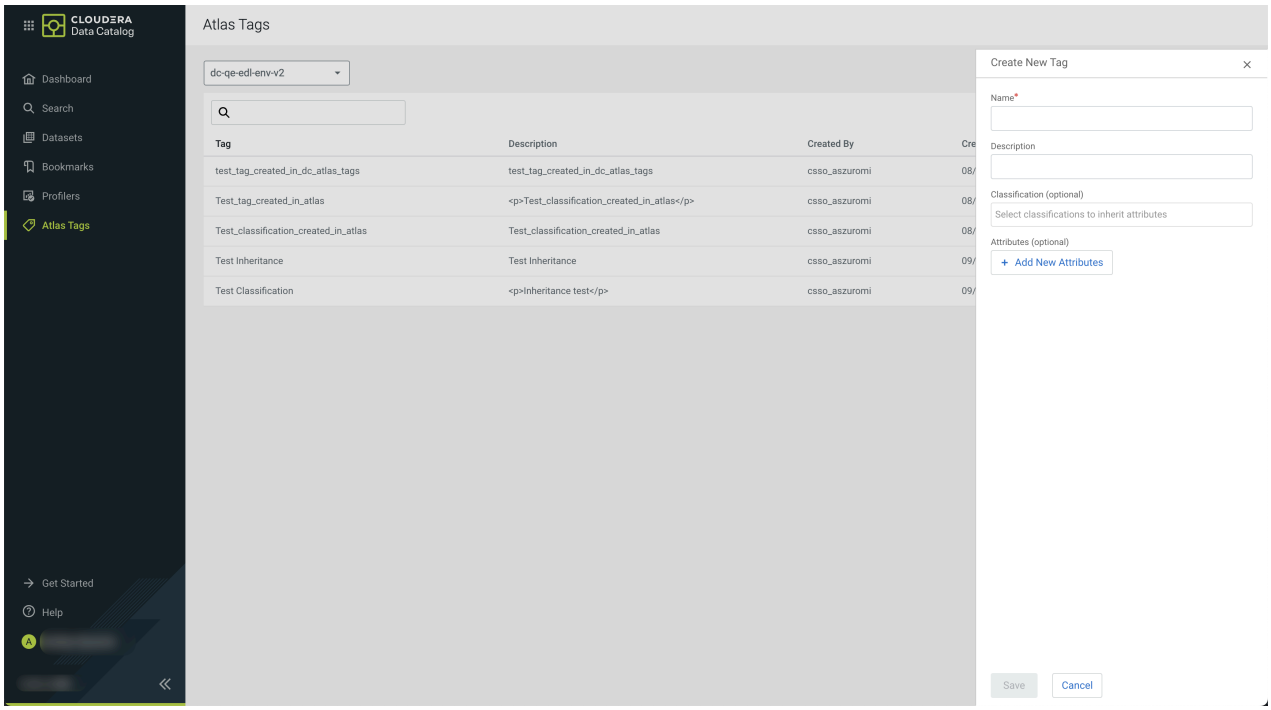

You can add or update Atlas tags. The created or updated tag is highlighted in the tag list as seen in the following diagram.

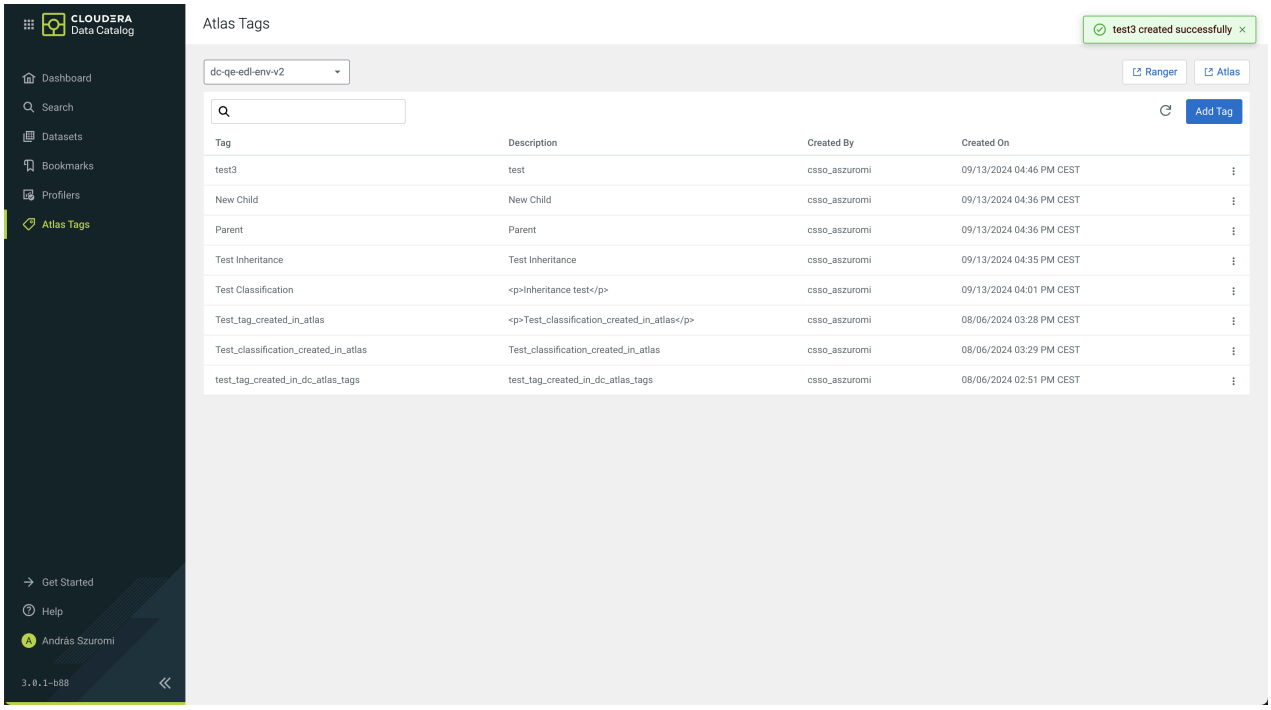

You can also edit or delete the Atlas tag as shown in the image. When you are editing the tag, you can only change the description or add new attributes.

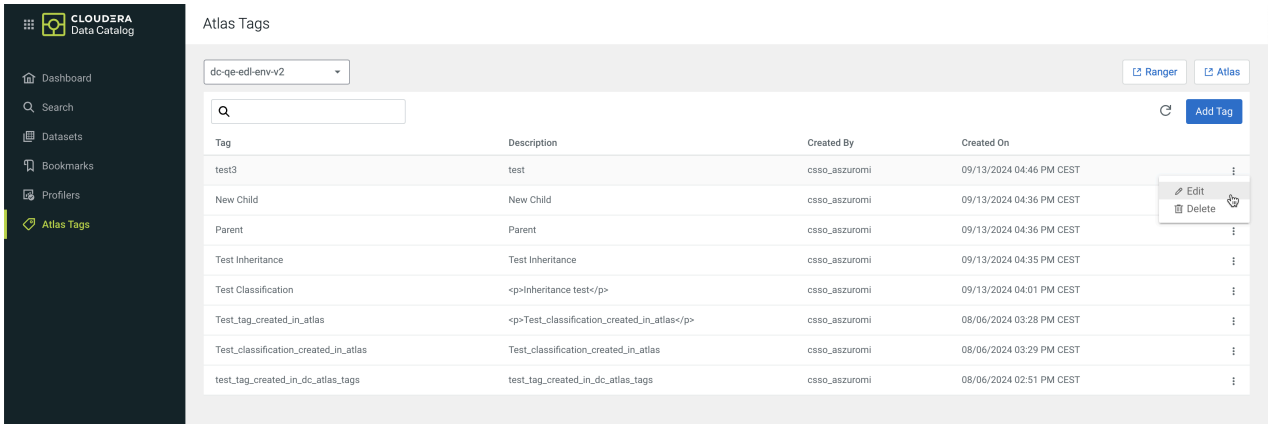

You can delete one Atlas tag at a time. A separate confirmation message appears.

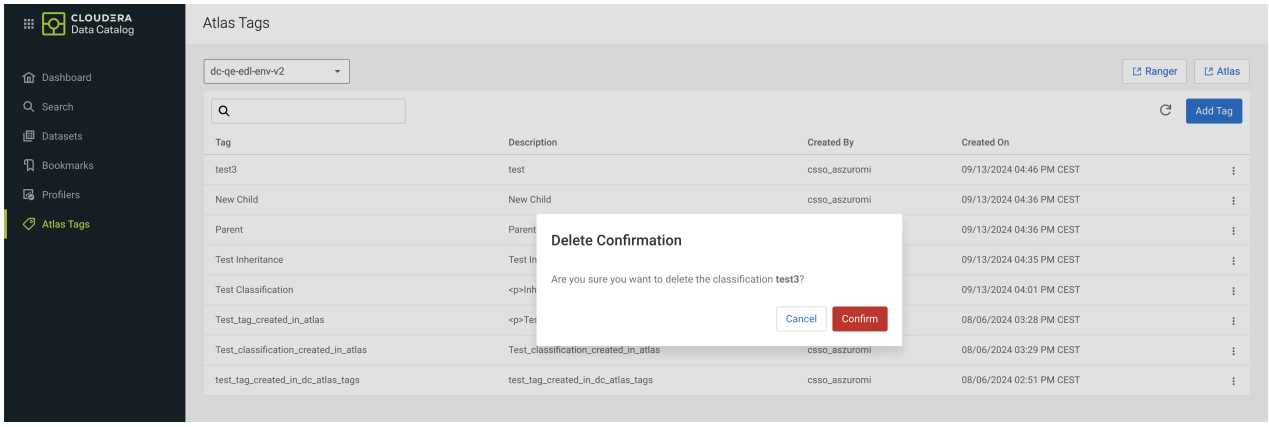

CDP Public Cloud / Data Catalog

Troubleshooting Data Catalog

Date published: 2019-11-14
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

FAQs..... 4

FAQs

This page provides answers to frequently asked questions about problems that you might encounter while you use Data Catalog service.

Issue while accessing Data Catalog

You must note about specific permissions that are needed to access Data Catalog service.

In order to use the Data Catalog UI, however, users have to add themselves to at least one Data Lake as an Environmental Admin / Environmental User. It allows you to search for assets on the Data Lake, add tags, and business glossaries to these assets.

In order to use Profilers (DataHub clusters specific to Data Catalog) related features, you must have a PowerUser role. This runs Spark jobs and provides some valuable insights / suggestions.

Cloudera Data Engineering 1.23.0

Accessing the Cloudera Data Engineering service using the API

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using the Cloudera Data Engineering API....4

Getting a Cloudera Data Engineering API access token....4

Using an access token in Cloudera Data Engineering API calls....5

Managing Cloudera Data Engineering job resources using the API....6

Creating a Cloudera Data Engineering resource using the API.... 6
Deleting a Cloudera Data Engineering resource using the API.... 7

Using custom operators and libraries for Apache Airflow using API....7

Adding or updating custom operators and libraries using API.... 8
Deleting custom operators and libraries using API.... 10
Troubleshooting custom operators and libraries.... 10
Viewing logs for custom operators and libraries using API....10
Cancel Maintenance using API.... 11

Managing workload secrets with Cloudera Data Engineering Spark Jobs using the API....11

Creating a workload secret for Cloudera Data Engineering Spark Jobs using API....11
Listing an existing workload secret for Cloudera Data Engineering Spark Jobs using API....12
Deleting a workload secret for Cloudera Data Engineering Spark Jobs using API....13
Linking a workload secret to the Cloudera Data Engineering Spark Job definitions using API.... 13
Using the workload secret in the Spark application code.... 14

Creating a Cloudera Data Engineering job using the API....15

Listing Cloudera Data Engineering jobs using the API.... 16

Getting Cloudera Data Engineering job information using the API....16

Adding a database and a Storage Account deployed in private DNS zones using the API....17

Managing VC-level Spark configurations using the API.... 18

Using the Cloudera Data Engineering API

Cloudera Data Engineering provides a robust API for integration with your existing continuous integration/continuous delivery platforms.

The Cloudera Data Engineering service API is documented in Swagger. You can view the API documentation and try out individual API calls by accessing the API DOC link in any virtual cluster:

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. In the Virtual Clusters section, click the Cluster Details icon in any of the listed virtual clusters.
3. Click API DOC.

Related Information

Using CLI-API to Automate Access to Cloudera Data Engineering

Using Cloudera Data Engineering CLI

Getting a Cloudera Data Engineering API access token

Cloudera Data Engineering uses JSON Web Tokens (JWT) for API authentication. To interact with a virtual cluster using the API, you must obtain an access token for that cluster.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view Cloudera Data Engineering Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Determine the authentication endpoint for your virtual cluster:

1. Navigate to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera management console.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster you want to interact with.
3. In the Virtual Clusters column on the right, click the Cluster Details icon on the virtual cluster you want to interact with.
4. Click the link under GRAFANA CHARTS. The hostname of the URL in your browser is the base URL, and /gateway/authtkn/knoxtoken/api/v1/token is the endpoint. For example:

```
https://service.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/gateway/authtkn/knoxtoken/api/v1/token
```

Procedure

1. From the client you want to use to access the API, run curl -u <workload_user> <auth_endpoint>. Enter your workload password when prompted.
   For example:

   ```
   curl -u csso_psherman https://service.cde-czlmkz4y.na-01.xvp2-7p8o.cloud
   era.site/gateway/authtkn/knoxtoken/api/v1/token
   ```

   The user account is your Cloudera workload user.
2. In the output, the access_token value is the JWT. For convenience, copy it and set it as an environment variable:

   ```
   export CDE_TOKEN=<ACCESS_TOKEN>
   ```

   Alternatively, you can set the token in a single step using jq to extract the token:

   ```
   export CDE_TOKEN=$(curl -s -u <WORKLOAD_USER> <AUTH_ENDPOINT> | jq -r '.a
   ccess_token')
   ```

What to do next

See Using an access token in Cloudera API calls for instructions on using the token in API calls.

Using an access token in Cloudera Data Engineering API calls

Cloudera Data Engineering uses JSON Web Tokens (JWT) for API authentication.

Before you begin

Get an access token and save it as an environment variable as described in Getting a Cloudera Data Engineering API access token.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view Cloudera Data Engineering Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. Determine the API URL for the virtual cluster you want to access using the API. The API URL for managing jobs is different from the URL used to obtain the access token.
   a) In the Cloudera console, click the Data Engineering tile and click Overview
   b) In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster you want to interact with using the API.
   c) In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to interact with using API.
   d) Click JOBS API URL to copy the link to your clipboard.
      For example: https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1

2. When you make an API call, include the JWT as a bearer token. For example, to list all jobs associated with the virtual cluster, assuming you have saved the token as an environment variable named CDE_TOKEN:

```
curl -H "Authorization: Bearer ${CDE_TOKEN}" -H "Content-Type: applicati
on/json" -X GET "https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.
site/dex/api/v1/jobs"
```

Managing Cloudera Data Engineering job resources using the API

A *resource* in Cloudera Data Engineering is a named collection of files referenced by a job. The files can include application code, configuration files, and even Python virtual environment specifications (requirements.txt). These resources can be managed using the Cloudera Data Engineering API.

Creating a Cloudera Data Engineering resource using the API

A *resource* in Cloudera Data Engineering is a named collection of files referenced by a job. The files can include application code, configuration files, and even Python virtual environment specifications (requirements.txt). You can create a resource using the Cloudera Data Engineering API.

Before you begin

As with all API calls, make sure you have a valid access token. For instructions, see Getting a Cloudera Data Engineering API access token.

Procedure

1. Create a JSON file describing the resource, using the structure for the type of resource you want to create.

   The JSON payload for a resource is structured as follows:

   files resource type

```
{
   "name":"<RESOURCENAME>",
   "type":"files",
   "retentionPolicy":"keep_indefinitely"
}
```

   python-env resource type

```
{
   "name":"<RESOURCENAME>",
   "type":"python-env",
   "retentionPolicy":"keep_indefinitely",
   "pythonEnv":{
      "pythonVersion":"python3"
   }
}
```

   custom-runtime-image resource type

```
{
  "customRuntimeImage": {
    "credential": "string",
    "engine": "string",
    "image": "string",
```

```
    "modified": "string"
  },
  "name": "<RESOURCENAME>",
  "retentionPolicy": "keep_indefinitely",
  "type": "custom-runtime-image"
}
```

2. Create the resource by submitting a POST request to the resources endpoint. The JSON filename is referenced using the @/path/to/filename.json convention. In this example, the JSON filename is fileResource.json, and describes a files type resource named example-job-files.

```
curl <jobs_api_url>/resources
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "Content-Type: application/json" \
-X POST -d @${HOME}/fileResource.json
```

3. Verify that the resource was creating by requesting the resource details from the /resources/<resourceName> endpoint:

```
curl <jobs_api_url>/example-job-files \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "Content-Type: application/json" \
-X GET
```

Deleting a Cloudera Data Engineering resource using the API

A *resource* in Cloudera Data Engineering is a named collection of files referenced by a job. The files can include application code, configuration files, and even Python virtual environment specifications (requirements.txt). You can delete a resource using the Cloudera Data Engineering API.

Before you begin

As with all API calls, make sure you have a valid access token. For instructions, see Getting a Cloudera Data Engineering API access token.

Procedure

1. Make sure that you no longer need the resource before deleting it.
2. Delete the resource by sending a DELETE request to the /resources/<*RESOURCENAME*> endpoint.
   For example, to delete a resource named example-job-files:

```
curl <jobs_api_url>/example-job-files \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "Content-Type: application/json" \
-X DELETE
```

Using custom operators and libraries for Apache Airflow using API

You can install and use custom python packages for Airflow with Cloudera Data Engineering. Cloudera provides access to the open source packages that you can use for your Airflow jobs using the Cloudera Data Engineering API.

Related Information

Using custom operators and libraries for Apache Airflow using Cloudera Data Engineering UI

Adding or updating custom operators and libraries using API

You can add or update custom python packages for Airflow with Cloudera Data Engineering. Cloudera provides access to the open source packages that you can use for your Airflow jobs using the API.

About this task

While you can install the operator, if additional runtime dependencies are required such as additional setup with binaries on the path, environment configuration like Kerberos and Cloud credentials, and so on, then the operator will not work.

Important: You must be in a maintenance session to set up a custom Python environement for Airflow. You can cancel a maintenance session before or after any of these steps in the workflow. After the maintenance session is cancelled, all objects related to the session are deleted.

1. Start a maintenance session by calling the /admin/airflow/env maintenance. It is a synchronous call and the session is created upon success.

```
curl -X POST \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/a
dmin/airflow/env/maintenance" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

2. Create and define the pip repositories by calling the /admin/airflow/env/maintenance/repos. This is a sync operation.

   The JSON payload to create a job is structured as follows:

```
{
"extraPipRepositories": [
 {
   "caCerts": "string",
   "credential": {
       "password": "string",
       "username": "string"
   },
   "skipCertValidation": false,
   "url": "string"
 }
  ],
  "pipRepository": {
 "caCerts": "string",
 "credential": {
     "password": "string",
     "username": "string"
 },
 "skipCertValidation": false,
 "url": "string"
  }
}
```

   For example:

```
curl -X POST \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/a
dmin/airflow/env/maintenance/repos" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json" \
-H "Content-Type: application/json" \
-d '{
```

```
  "pipRepository": {
 "url": "https://pypi.org/simple/"
  }
}'
curl -X GET \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/a
dmin/airflow/env/maintenance/status" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

3. Build the Python environment by calling /admin/airflow/env/maintenance/build.
   a. To monitor results, issue GET /admin/airflow/env/maintenance/status.

Required build starting states	Success state	Failure state
pip-repos-defined built build-failed	built	build-failed

The payload to create a job for a multi-part/form input is as follows:

```
# requirements.txt should be a file on the local file system and a well
formatted python requirements.txt

curl -X POST \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/
admin/airflow/env/maintenance/build" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json" \
--form file="@requirements.txt"

curl -X GET \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/
admin/airflow/env/maintenance/status" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

4. Activate the Python environment by calling /admin/airflow/env/maintenance/activate.
   a. To monitor results, issue GET /admin/airflow/env/maintenance/status.

Required activation starting states	Success state	Failure state
built activation-failed	N/A, request should fail with 404, since maintenance is done /admin/airflow/env/status should have the “activated” status and the description of your environment should be correct and updated	activation-failed

For example:

```
curl -X POST \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/
admin/airflow/env/maintenance/activate" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

```
curl -X GET \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/
admin/airflow/env/maintenance/status" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

You can now use your DAGs with custom code.

Deleting custom operators and libraries using API

If you no longer need your custom operators and libraries, you can reset the runtime environment to Cloudera Data Engineering default environment by deleting the currently active Python environment. To reset the Cloudera Data Engineering Airflow Python environment, complete the following steps.

Note: Ensure that no jobs are using the packages which are meant to be deleted, otherwise they will stop working in the Airflow instance. Also, no other maintenance operations like delete operation, updating the python environment is allowed while the delete operation is in progress.

To delete an environment, call DELETE on /admin/airflow/env:

```
curl -X DELETE \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/adm
in/airflow/env" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

To ensure that the environment has been successfully deleted, the currently active environment status must be checked and the request must fail with a 404. This operation may take a few minutes. This indicates that the environment is not currently in use. For example:

```
curl -X GET \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/ad
min/airflow/env" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

Troubleshooting custom operators and libraries

Learn the different ways that you can troubleshoot the setup of your Airflow Python environment in Cloudera Data Engineering.

Viewing logs for custom operators and libraries using API

You can view the logs of a maintenance session and logs of an environment. You can view the maintenance logs when a maintenance is ongoing and if fails at some step. You can view logs of an active environment if some jobs behave unexpectedly on an active Airflow Python environment.

View logs of a maintenance session:

Note: Logs are cumulative and would contain all actions done within a maintenance session. Therefore, each attempt log (upload retry) will end up in the logs.

```
curl -X GET \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/ad
min/airflow/env/maintenance/logs" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

View the logs of an environment:

```
curl -X GET \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/ad
min/airflow/env/logs" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

Cancel Maintenance using API

You cancel the maintenance of an Airflow Python environment setup and all the objects related to the session will be discarded and deleted.

To cancel a maintenance, call DELETE on /admin/airflow/env/maintenance:

```
curl -X DELETE \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/adm
in/airflow/env/maintenance" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

To ensure that the maintenance has been cancelled, the maintenance status must be checked and the request must fail with a 404. This may take seconds or up to a few minutes. This indicates that no maintenance session is ongoing. For example:

```
curl -X GET \
"https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/ad
min/airflow/env/maintenance/status" \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H "accept: application/json"
```

Managing workload secrets with Cloudera Data Engineering Spark Jobs using the API

This API provides a secure way to create and store workload secrets for Cloudera Data Engineering Spark Jobs. This is a more secure alternative to storing credentials in plain text embedded in your application or job configuration.

Creating a workload secret for Cloudera Data Engineering Spark Jobs using API

Creating a workload secret for Cloudera Data Engineering Spark Jobs using API provides a secure way to create workload secrets. This is a more secure alternative to storing credentials in plain text embedded in your application or job configuration.

Procedure

1. Create Workload Credentials with name workload-cred-:

```
curl -k -X "POST" 'https://<dex-vc-host>/dex/api/v1/credentials' \
  -H 'Accept: application/json' \
  -H 'Connection: keep-alive' \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  --data '{
  "workloadCred": {
```

```
    "aws-secret": "secret123",
    "db-pass": "dbpass123"
  },
  "name": "workload-cred-1",
  "type": "workload-credential",
  "description": "workload credential description"
}'
```

2. Edit the existing workload credentials with new data. For example, edit workload-cred-1:

```
curl -k -X "PATCH" 'https://<dex-vc-host>/dex/api/v1/credentials/workload-
cred-1' \
  -H 'Accept: application/json' \
  -H 'Connection: keep-alive' \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  --data '{
  "workloadCred": {
    "aws-secret": "newsecret123",
    "db-pass": "newdbpass123"
  }
}'
```

Listing an existing workload secret for Cloudera Data Engineering Spark Jobs using API

You can list an existing secret for Cloudera Data Engineering Spark Jobs using API.

Procedure

1. List the existing workload credentials in the virtual cluster by creating a filter with type that is equal to workload-credential

```
curl -k -X "GET" 'https://<dex-vc-host>/dex/api/v1/credentials?filter=type
%5Beq%5Dworkload-credential' \
  -H 'Accept: application/json' \
  -H 'Connection: keep-alive' \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${CDE_TOKEN}"
```

Response:

```
{
  "credentials": [
    {
      "name": "workload-cred-1",
      "type": "workload-credential",
      "description": "workload credential description",
      "created": "2022-10-18T07:26:41Z",
      "modified": "2022-10-18T07:26:41Z"
    }
  ],
  "meta": {
    "hasNext": false,
    "limit": 20,
    "offset": 0,
    "count": 1
  }
}
```

2. Get the existing specific workload credential. For example, workload-cred-1:

```
curl -k -X "GET" 'https://<dex-vc-host>/dex/api/v1/credentials/workload-
cred-1' \
  -H 'Accept: application/json' \
  -H 'Connection: keep-alive' \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${CDE_TOKEN}"
```

Response:

```
{
      "name": "workload-cred-1",
      "type": "workload-credential",
      "description": "worklaod credential description",
      "created": "2022-10-18T07:26:41Z",
      "modified": "2022-10-18T07:26:41Z"
}
```

Deleting a workload secret for Cloudera Data Engineering Spark Jobs using API

You can delete an existing secret for Cloudera Data Engineering Spark Jobs using API.

Procedure

Delete the workload-credential. For example, delete workload-cred-1:

```
curl -k -X "DELETE" 'https://<dex-vc-host>/dex/api/v1/credentials/workload-c
red-1' \
  -H 'Accept: application/json' \
  -H 'Connection: keep-alive' \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${CDE_TOKEN}"
```

Linking a workload secret to the Cloudera Data Engineering Spark Job definitions using API

Once you've created a workload secret, you can link it to the Cloudera Data Engineering Spark Job definitions that you created using API.

Procedure

1. Use 1 or more workload-credentials in a new Spark job. For example, workload-cred-1 and workload-cred-2 in the job below:

```
curl -k -X "POST" 'https://<dex-vc-host>/dex/api/v1/jobs' \
  -H 'Accept: application/json' \
  -H 'Connection: keep-alive' \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  --data '{
    "name": "test-workload-job",
    "spark": {
        "className": "org.apache.spark.examples.SparkPi",
        "file": "spark-examples_2.11-2.4.8.7.2.12.0-195.jar"
    },
```

```
    "mounts": [
        {
            "resourceName": "spark-jar",
            "dirPrefix": ""
        }
    ],
    "type": "spark",
    "workloadCredentials": ["workload-cred-1", "workload-cred-2"]
}'
```

2. Use 1 or more workload-credentials in an existing Spark job. For example, workload-cred-1 and workload-cred-2 in the job below:

```
curl -k -X "PATCH" 'https://<dex-vc-host>/dex/api/v1/jobs/test-workload-
job' \
  -H 'Accept: application/json' \
  -H 'Connection: keep-alive' \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${CDE_TOKEN}" \
  --data '{
        "workloadCredentials": ["workload-cred-1", "workload-cred-2"]
}'
```

Using the workload secret in the Spark application code

To use the workload secret credentials, you can read the credentials that are mounted into the Spark drivers and executors as read-only files.

Procedure

The workload secrets are mounted into the Spark drivers and executors in this path: /etc/dex/secrets/<workload-credential-name>/<credential-key-1> /etc/dex/secrets/<workload-credential-name>/<credential-key-2>

Example workload credentials:

The workload-credential was created with the payload below.

```
{
  "workloadCred": {
    "aws-secret": "secret123",
    "db-pass": "dbpass123"
  },
  "name": "workload-cred-1",
  "type": "workload-credential",
  "description": "workload credential description"
}
```

The secrets can be read as local files from the paths below within the Spark drivers and executors:

/etc/dex/secrets/workload-cred-1/aws-secret

/etc/dex/secrets/workload-cred-1/db-pass

Example of a Pyspark application code to read a secret:

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("Sample DB Connection") \
    .getOrCreate()
```

```
# read the password from the local file
dbPass=open("/etc/dex/secrets/workload-cred-1/db-pass").read()

# use the password in a jdbc connection
jdbcDF= spark.read \
```

Creating a Cloudera Data Engineering job using the API

You can create a job in Cloudera Data Engineering using the Cloudera Data Engineering jobs API endpoint.

Before you begin

Request an access token and save it as an environment variable to use in API calls. For instructions, see Getting a Cloudera Data Engineering API access token.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view Cloudera Data Engineering Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

If you're using secrets or if you want to access workload secrets, refer to Managing workload secrets with Cloudera Data Engineering using the API linked below.

Procedure

1. Determine the API URL for the virtual cluster you want to access using the API:
   a) In the Cloudera console, click the Data Engineering tile and click Overview
   b) In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster you want to interact with using the API.
   c) In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to interact with.
   d) Click JOBS API URL to copy the URL.
   For example: https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1
2. Submit the job creation request using the API to the /jobs endpoint.

   The JSON payload to create a job is structured as follows:

```
{
  "name": "demoJob",
  "spark": {
    "className": "com.example.demoJobMainClass",
    "file": "local:/path/to/demoJobJar"
  },
  "type": "spark"
}
```

```
curl -H "Authorization: Bearer ${CDE_TOKEN}" <JOBS_API_URL>/jobs \
-H "Content-Type: application/json" \
-X POST -d "{\"name\":\"demoJob\",\"spark\":{\"className\":\"com.example.d
emoJobMainClass\",\"file\":\"local:/path/to/demoJobJar\"},\"type\":\"spark
\"}"
```

3. Verify the job was created. You can view job details using the /jobs/<*JOBNAME*> endpoint:

```
curl -H "Authorization: Bearer ${CDE_TOKEN}" -H "Content-Type: applicati
on/json" -X GET "https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.
site/dex/api/v1/jobs/demoJob"
```

Listing Cloudera Data Engineering jobs using the API

You can list Cloudera Data Engineering jobs using the API by issuing a GET request to the /jobs endpoint.

Before you begin

Get an access token and save it as an environment variable as described in Getting a Cloudera Data Engineering API access token.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view Cloudera Data Engineering Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. Determine the API URL for the virtual cluster you want to access using the API:
   a) In the Cloudera console, click the Data Engineering tile and click Overview
   b) In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster you want to interact with using the API.
   c) In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to interact with.
   d) Click JOBS API URL to copy the URL.
   For example: https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1
2. Issue a GET request to the /jobs endpoint:

```
curl -H "Authorization: Bearer ${CDE_TOKEN}" -X GET "https://pmjkrgn5.cde-
czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/jobs"
```

Getting Cloudera Data Engineering job information using the API

You can get Cloudera Data Engineering job information using the API by issuing a GET request to the /job s/<*JOB_NAME*> endpoint.

Before you begin

Get an access token and save it as an environment variable as described in Getting a Cloudera Data Engineering API access token.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view Cloudera Data Engineering Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. Determine the API URL for the virtual cluster you want to access using the API:
   a) In the Cloudera console, click the Data Engineering tile and click Overview
   b) In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster you want to interact with using the API.
   c) Click JOBS API URL to copy the URL.
   For example: https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1
2. Issue a GET request to the /jobs/<*JOBNAME*> endpoint:

```
curl -H "Authorization: Bearer ${CDE_TOKEN}" -X GET "https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.cloudera.site/dex/api/v1/jobs/SparkPi"
```

Configuring an existing private DNS zone for a database and a Storage Account File Share using the API (Technical Preview)

Learn about how to configure existing private DNS zones for a database and a Storage Account File Share using the API.

Before you begin

To learn about the CLI option and about the prerequisites:

- for the database, see Configuring an existing private DNS zone for a database
- for the Storage Account File Share, see Configuring an existing private DNS zone for a Storage Account File Share

Procedure

To configure the private DNS zones for the database and the Storage Account File Share, specify the full resource ID of the private DNS zones where you want to configure them.

To configure the Azure private DNS zone IDs for the database and the Storage Account File Share, in config.properties of the cluster endpoint, post the full resource IDs in the following parameters respectively:

- For the database, use azure.database.privateDNSZoneId
- For the Storage Account, use azure.fileshare.privateDNSZoneId

Request example

Example for the base URL, which changes according to the region: https://console.us-west-1.cdp.cloudera.com

```
curl -H "Cookie: cdp-session-token=${CST}" '[***BASE-URL***]/dex/api/v1/clus
ter' \
-H "Content-Type: application/json" \
-H 'accept: application/json' \
-X POST -d '{
    "name": "[***NAME***]",
    "env": "[***ENVIRONMENT-NAME***]",
    "config": {
        "properties": {
            "loadbalancer.internal": "true",
            "dbus-wxm-client.enabled": "false",
            "kubernetes.api.allowList": "",
            "loadbalancer.allowList": "",
            "kubernetes.api.proxyCIDRList": "",
            "subnets": "",
            "loadBalancerSubnets": "",
            "default-vc.create": "true",
            "cde.deployPreviousVersion": "false",
            "privateNetwork.enabled": "true",
            "azure.fileshare.privateDNSZoneId": "[***AZURE-PRIVATEDNSZONE-RE
SOURCE-ID-FOR-STORAGE-ACCOUNT-FILE-SHARE***]",
            "azure.database.privateDNSZoneId": "[***AZURE-PRIVATEDNSZONE-
RESOURCE-ID-FOR-DATABASE***]"
        },
        "resources": {
            "instance_type": "Standard_D8s_v4",
            "min_spot_instances": "0",
            "max_spot_instances": "0",
            "min_instances": "0",
            "max_instances": "50",
            "initial_instances": "0",
            "initial_spot_instances": "0",
            "root_vol_size": "100",
            "allp_instance_group_details": {
                "instance_type": "Standard_D8s_v4",
                "min_spot_instances": "0",
                "max_spot_instances": "0",
                "min_instances": "0",
                "max_instances": "50",
                "initial_instances": "0",
                "initial_spot_instances": "0",
                "root_vol_size": "100"
            }
        },
        "tags": {}
    },
    "skipValidation": false
}'
```

Managing Virtual Cluster-level Spark configurations using the API (Technical Preview)

Learn about how to manage Virtual Cluster (VC)-level Spark configurations using the API.

Before you begin

For information about managing Virtual Cluster-level Spark configurations using the CLI, see Managing Virtual Cluster-level Spark configurations.

Creating a VC instance with VC-level Spark configurations

On the control plane side, to create a Virtual Cluster (VC) instance with VC-level Spark configurations, run the following command.

Note: The Spark configurations defined in the sparkConfigs field apply to all the Spark jobs within the VC.

```
curl -H "Cookie: cdp-session-token=${CST}" '[***BASE-URL***]/dex/api/v1/clus
ter/[***CLUSTER-ID***]/instance' \
-H "Content-Type: application/json" \
-H 'accept: application/json' \
-X POST -d '{
  "name": "vc-spark-configs-cli",
  "config": {
    "sparkConfigs": {
      "spark.executor.instances": "4",
      "is.config.vc": "true",
      "is.config.job": "false"
    },
    "resources": {
      "cpu_requests": "20",
      "mem_requests": "80Gi"
    }
  }
}'
```

Example for the base URL, which changes according to the region: https://console.us-west-1.cdp.cloudera.com

Payload for creating the Spark configurations for a VC

```
{
  "name": "vc-spark-configs-cli",
  "config": {
    "sparkConfigs": {
      "spark.executor.instances": "4",
      "is.config.vc": "true",
      "is.config.job": "false"
    },
    "resources": {
      "cpu_requests": "20",
      "mem_requests": "80Gi"
    }
  }
}
```

If successful, the 200, OK response is received.

Updating the Spark configurations for a VC instance

On the control plane side, to update the Spark configurations for a VC instance, run the following command.

Note: The Spark configurations defined in the sparkConfigs field override the existing sparkConfigs.

```
curl -H "Cookie: cdp-session-token=${CST}" '[***BASE-URL***]/dex/api/v1/clus
ter/[***CLUSTER-ID***]/instance/[***INSTANCE-ID***]' \
-H "Content-Type: application/json" \
-H 'accept: application/json' \
-X 'PATCH' -d '{
  "config": {
    "sparkConfigs": {
```

```
      "spark.executor.instances": "4",
      "is.config.vc": "true",
      "is.config.job": "false"
    }
  }
}'
```

Example for the base URL, which changes according to the region: https://console.us-west-1.cdp.cloudera.com

Payload for updating the Spark configurations

```
{
  "config": {
    "sparkConfigs": {
      "spark.executor.instances": "4",
      "is.config.vc": "true",
      "is.config.job": "false"
    }
  }
}
```

If successful, the 200, OK response is received.

Getting the Spark configuration information for the Job-runs of a VC instance

On the workload side, to get the Spark configuration information for the Job-runs of a VC instance, run the following command.

In the Job-Runs endpoint, the instanceSparkConfigs field contains the Spark configurations that are applied at VC-level to a particular virtual cluster instance.

1. To get the API URL of the VC you want to access, on the Cloudera Data Engineering UI, navigate to Administration Virtual Clusters and click the Cluster Details icon of the VC you want to interact with.
2. To copy the URL, click Actions COPY JOBS API URL .

```
curl -X GET '[***JOBS-API-URL***]/job-runs/[***JOB-RUN-ID***]' \
-H 'accept: application/json' \
-H "Cookie: cde-csrf-token=${CSRF}" \
-H "Cookie: hadoop-jwt=${JWT}"
```

In the Spark.Conf field, the compiled list of configurations, which include the configurations applied at VC-level and at Job-level, is added with the default configurations of Spark. The configuration information is sent to Livy, which is a job submitter with precedence. The precedence hierarchy, from highest to lowest priority is: Job-Run, Job, and VC-level configurations.

Cloudera Data Engineering 1.23.0

Backing up and restoring Cloudera Data Engineering jobs

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Backing up Cloudera Data Engineering jobs on remote storage.....4

Backing up Cloudera Data Engineering jobs on local storage.....6

Restoring Cloudera Data Engineering jobs from backup.....10

Restoring jobs from a 1.20.3 VC backup.....13

Backing up Cloudera Data Engineering jobs on remote storage

You can back up Cloudera Data Engineering jobs and associated resources. Backups are saved as ZIP files that can be used to restore jobs and their associated resources. Backups and restores are also supported for remote storage (also known as object store).

Note: Currently, the user who runs the cde backup restore command has permissions, by default, to run the jobs. This may cause Cloudera Data Engineering jobs to fail if the workload user differs from the user who runs the jobs on Source Cloudera Data Engineering Service where the backup was performed. This failure is due to the Workload User having different privileges as the user who is expected to run the job. Additionally, backup and restore functionality does not include Airflow metadata related to connections and variable definitions. Those must be manually recreated in the new service.

For CDE CLI

Before you begin

- Download and configure the CDE CLI.

Steps for backing up on remote storage

1. Run the cde backup create command to create a backup of the jobs in the virtual cluster your CDE CLI is configured to interact with. By default, all job configurations in the cluster are backed up, but the resources are not. You can use command flags to change this behavior as follows:

   --continue-on-error

   If set to true, the archive generation continues even if there are errors for some items. It works only for remote storage.

   --credential-filter

   Selects credentials to be backed up using the fieldname[operator]argument syntax. This command flag can be repeated. The name of the job and resource API field is 'fieldname' and the 'operator is

one of the following: 'eq', 'noteq', 'lte', 'lt', 'gte', 'gt', 'in', 'notin', 'like', 'rlike'. You can add multiple filters using and. For example:

```
cde backup create --credential-filter "name[noteq]test" --creden
tial-filter "created[gte]2020-01-01"
```

--include-active-airflow-pyenv

If set to true, it backs up the active Airflow Libraries and Operators with the associated credentials and secret. The default value is false.

--include-credential-secrets

Backs up credential secrets.

Warning: In the archive, the backed up secrets are not encrypted.

--include-credentials

Backs up all virtual cluster credentials. By default, the credential secrets are not included in the backup.

--include-job-resources

Backs up all resources associated with the selected jobs. These resources cannot be filtered out by the --resource-filter parameter.

--include-jobs

Backs up all jobs in the virtual cluster. This is the default behavior.

--include-resource-credentials

Backs up credentials for the resources. You cannot filter out selected credentials using the credenti alfilter parameter. This is the default behavior.

--include-resources

Backs up all resources in the virtual cluster, including those not associated with jobs.

--job-filter <*FILTER_STRING*>

Selects jobs to back up using the fieldname[operator]argument syntax. This command flag can be repeated. The name of the job and resource API field is 'fieldname' and the 'operator is one of the following: 'eq', 'noteq', 'lte', 'lt', 'gte', 'gt', 'in', 'notin', 'like', 'rlike'. You can add multiple filters using and. For example:

```
cde backup create --job-filter "name[noteq]test" --job-filter "c
reated[gte]2020-01-01"
```

--resource-filter <*FILTER_STRING*>

Selects resources to back up using the fieldname[operator]argument syntax. This command flag can be repeated. Filter by adding detail to the filter syntax, for example, filter syntax 'fieldname[operator]argument'. The name of the job and resource API field is 'fieldname' and the

'operator is one of the following: 'eq', 'noteq', 'lte', 'lt', 'gte', 'gt', 'in', 'notin', 'like', 'rlike'. You can add multiple filters using and. For example:

```
'name[noteq]my-resource','created[gte]2020-01-01'
```

--output

The output in text or JSON format. The default is text format.

--remote-storage

Backs up to the remote storage. The default value is false.

--remote-path

Use the remote backup file relative path together with the remote-storage parameter. Do not include the filename in the path. The path is relative to /dex/backup/.

--remote-name

Use the remote backup file name together with the remote-storage parameter. If the file name is not specified, then a default generated value is used.

--validate

After a remote backup archive is created, it validates the archive.

Example for backing up to remote storage all jobs and its related resources, plus all resources whose name contains "data".

```
cde backup create --remote-storage --include-resources --resource-filter
 "name[like]%data%"
```

Example for creating a backup on remote storage:

```
./cde backup create --remote-storage --remote-path test --remote-name ar
chive.zip

# output
{"archiveRelativePath":"test/archive.zip","archiveURL":"s3a://dex-dev-d
efault-aws-storage/datalake/logs/dex/backup/test/archive.zip","code":201}
```

For Web UI

Result

Depending on your browser settings, you are either prompted for a location to save the file, or the file is downloaded to your default download location. The file is a ZIP file named archive-<*TIMESTAMP*>.zip.

To restore a backup file, see Restoring Cloudera Data Engineering jobs from backup.

Backing up Cloudera Data Engineering jobs on local storage

You can back up Cloudera Data Engineering jobs and their associated resources and/or repositories. Backups are saved as ZIP files that can be used to restore jobs and their associated resources and/or repositories. Backups and restores is also supported for remote storage (also known as object store).

Note: Currently, the user who runs the cde backup restore command has permissions, by default, to run the Jobs. This may cause Cloudera Data Engineering jobs to fail if the workload user differs from the user who runs the jobs on Source Cloudera Data Engineering Service where the backup was performed. This failure is due to the Workload User having different privileges as the user who is expected to run the job. Additionally, backup and restore functionality does not include Airflow metadata related to connections and variable definitions. Those must be manually recreated in the new service.

For CDE CLI

Before you begin

- Download and configure the CDE CLI.

Steps for backing up on local storage

1. Run the cde backup create command to create a backup of the jobs in the virtual cluster your CDE CLI is configured to interact with. By default, all job configurations in the cluster are backed up, but the resources are not. You can use command flags to change this behavior as follows:

   --credential-filter

   Selects credentials to be backed up using the fieldname[operator]argument syntax. This command flag can be repeated. The name of the job and resource API field is 'fieldname' and the 'operator' is one of the following: 'eq', 'noteq', 'lte', 'lt', 'gte', 'gt', 'in', 'notin', 'like', 'rlike'. You can add multiple filters using and. For example:

   ```
   cde backup create --credential-filter "name[noteq]test" --creden
   tial-filter "created[gte]2020-01-01"
   ```

   --include-active-airflow-pyenv

   If set to true, it backs up the active Airflow Libraries and Operators with the associated credentials and secret. The default value is false.

   --include-credential-secrets

   Backs up credential secrets.

   Warning: In the archive, the backed up secrets are not encrypted.

   --include-credentials

   Backs up all virtual cluster credentials. By default, the credential secrets are not included in the backup.

   --include-job-resources

   Backs up all resources associated with the selected jobs. These resources cannot be filtered out by the --resource-filter parameter.

   --include-jobs

   Backs up all jobs in the virtual cluster. This is the default behavior.

   --include-resource-credentials

   Backs up credentials for the resources. You cannot filter out selected credentials using the credenti alfilter parameter. This is the default behavior.

   --include-resources

   Backs up all resources and repositories (internally these are also resources only) in the virtual cluster, including those not associated with jobs.

   --job-filter <*FILTER_STRING*>

   Selects jobs to back up using the fieldname[operator]argument syntax. This command flag can be repeated. The name of the job and resource API field is 'fieldname' and the 'operator' is one of the

following: 'eq', 'noteq', 'lte', 'lt', 'gte', 'gt', 'in', 'notin', 'like', 'rlike'. You can add multiple filters using and.

For example: The following command backs up locally all jobs whose name is not test, and which is created later than or equal to 2020-01-01, and all their related resources:

```
cde backup create --job-filter "name[noteq]test" --job-filter "c
reated[gte]2020-01-01"
```

--local-path <*FILENAME*>

Specifies the local file path and name to store the backup. By default, the backup filename is arch ive-<timestamp>.zip.

--output

The output in text or JSON format. The default is text format.

--resource-filter <*FILTER_STRING*>

Selects resources to back up using the fieldname[operator]argument syntax. This command flag can be repeated. The name of the job and resource API field is 'fieldname' and the 'operator' is one of the following: 'eq', 'noteq', 'lte', 'lt', 'gte', 'gt', 'in', 'notin', 'like', 'rlike'. You can add multiple filters using and. For example:

```
cde backup create --resource-filter "name[eq]test" --resource-fi
lter "created[gte]2020-01-01"
```

For example, to backup all jobs containing the string etl and include all resources associated with those jobs:

```
cde backup create --job-filter "name[like]%etl%" --include-job-resources
```

2. Validate the backup using the cde backup list-archive command. For example:

```
cde backup list-archive --local-path archive-2021-11-10T01:24:06.zip
```

Confirm that all jobs and resources that you expected to be backed up are included.

Result

The output of cde backup list-archive is similar to the following:

```
{
  "backup_set": [
    {
      "id": "v1/backupset/v1/f332bb06-7612-4345-8f3b-da4f27f315b3/",
      "cdeVersion": "1.18.0-b59",
      "clusterID": "cluster-2bqdpfrs",
      "appID": "dex-app-82wlpx6d",
      "app_name": "yjtest",
      "user": "csso_yjlu",
      "backupStarted": "2022-10-07T06:39:46.082837691Z"
    }
  ],
  "entries": [
    {
      "backup_set": "v1/backupset/v1/f332bb06-7612-4345-8f3b-da4f27f315b3/
",
      "entityType": "Job",
      "name": "example-a",
      "adjustedName": "example-a",
      "archiveDirectoryPath": "v1/jobs/v1/d7826797-4985-455f-a9c8-2ab1cc
624d9b/",
      "user": "csso_yjlu"
    },
```

```
    {
      "backup_set": "v1/backupset/v1/f332bb06-7612-4345-8f3b-da4f27f315
b3/",
      "entityType": "Resource",
      "name": "example-data",
      "adjustedName": "example-data",
      "archiveDirectoryPath": "v1/resources/v1/41979747-5ad1-40c3-b301-cd5
7111411f9/",
      "user": "csso_yjlu"
    }
  ]
}
```

For Cloudera Data Engineering API

Note: Some of the command examples provided use jq to parse the JSON output and make it more readable.

Before you begin

Request an access token and save it as an environment variable to use in API calls. For instructions, see Getting a Cloudera Data Engineering API access token .

Steps

1. Determine the API URL for the virtual cluster containing the job you want to back up:
   a. Navigate to the Cloudera Data Engineering Overview page.
   b. In the Cloudera Data Engineering Services column, select the service containing the virtual cluster with the jobs you want to back up.
   c. In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster containing the jobs you want to back up.
   d. Copy the URL under JOBS API URL, and set it as an environment variable. For example:

   ```
   export CDE_JOBS_API="https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.c
   loudera.site/dex/api/v1"
   ```

2. Back up jobs using a URL-encoded filter with the syntax name[like]<*QUERY*>, modeled after the SQL like operator. For example, to back up jobs containing the string etl, set jobfilter to name[like]%etl% (URL-encoded as name%5Blike%5D%25etl%25):

   ```
   curl -k \
   -H "Authorization: Bearer ${CDE_TOKEN}" \
   -X GET "${CDE_JOBS_API}/admin/export?exportjobs=true&jobfilter=name%5Bli
   ke%5D%25etl%25&exportjobresources=true&exportresources=false" \
   -H "accept: application/zip" \
   --output cde-etl-jobs-backup.zip
   ```

   To back up all jobs and associated resources, omit the jobfilter parameter:

   ```
   curl -k \
   -H "Authorization: Bearer ${CDE_TOKEN}" \
   -X GET "${CDE_JOBS_API}/admin/export?exportjobs=true&exportjobresources=
   true&exportresources=false" \
   -H "accept: application/zip" \
   --output cde-all-jobs-backup.zip
   ```

   To back up all jobs and associated resources, and the active Airflow Libraries and Operators with the associated credentials and secrets, run:

   ```
   curl -k \
   -H "Authorization: Bearer ${CDE_TOKEN}" \
   ```

```
-X GET "${CDE_JOBS_API}/admin/export?exportjobs=true&exportjobresources=
true&exportresources=false&exportactiveairflowpyenv=true” \
-H "accept: application/zip" \
```

3. (Optional) You can validate a backup file by uploading it to the /admin/list-archive endpoint. For example, for a backup file named cde-all-jobs-backup.zip:

```
curl -k \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-X POST "${CDE_JOBS_API}/admin/list-archive" \
-H "accept: application/json" \
-H "Content-Type: multipart/form-data" \
-F "file=@/path/to/cde-all-jobs-backup.zip;type=application/zip" \
| jq
```

For Web UI

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and higher has been updated. The left-hand menu provides easy access to commonly used pages. The steps below vary slightly, for example, the Overview page has been replaced with the Home page. You can also manage a job by clicking Jobs on the left-hand menu, then selecting a Virtual Cluster from a drop-down at the top of the Jobs page. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Steps

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Jobs in the left navigation menu. The Jobs page displays.
3. From the drop-down in the upper left-hand corner, select the Virtual Cluster with the jobs that you want to back up.
4. Click ⋮ at the top right, and then click Backup Jobs.

Result

Depending on your browser settings, you are either prompted for a location to save the file, or the file is downloaded to your default download location. The file is a ZIP file named archive-<*TIMESTAMP*>.zip.

To restore a backup file, see Restoring Cloudera Data Engineering jobs from backup.

Restoring Cloudera Data Engineering jobs from backup

You can restore Cloudera Data Engineering jobs and associated resources from a backup ZIP file.

Note: The backup and restore functionality does not include Airflow metadata related to connections and variable definitions. Recreate those manually in the new service. If the backup contains Airflow Libraries and Operators, consider the following:

- Restoring a backup with active Airflow Libraries and Operators can take 5 to 60 minutes, depending on the packages.
- You can restore the backup only to a virtual cluster that does not have active or maintenance mode Airflow Libraries and Operators.
- Restore the backup to the same Cloudera Data Engineering version. If you try to restore to a newer Cloudera Data Engineering and Airflow version, the restore can fail due to Python version or PIP constraint changes. In such a case, remove the Airflow Libraries and Operators from the archive.

For CDE CLI

Before you begin

- You must have a valid backup file to restore from. For instructions on backing up Cloudera Data Engineering jobs, see Backing up Cloudera Data Engineering jobs .
- Download and configure the CDE CLI.

Steps

1. Run the cde backup restore command to restore a backup file to the virtual cluster your CDE CLI is configured to interact with. Use the --duplicate-handling flag to select the policy for handling duplicate jobs. The possible values are:

   error

   Return an error if there are duplicate job names, and abort the restore operation. This is the default behavior.

   rename

   If a job name in the backup conflicts with an existing job, keep the existing job and rename the restored job by appending a numerical identifier to the job name.

   keep-original

   If a job name in the backup conflicts with an existing job, keep the existing job and do not restore the backed up job.

   For example:

   ```
   cde backup restore --local-path archive-2021-11-10T01:24:06.zip --duplic
   ate-handling rename
   ```

   --remote-path

   If you use a remote storage, use this flag for the relative remote storage path of the backup to be restored. This restores the archive in the remote object store. Include a filename in the path. The path is relative to /dex/backup/.

   Restore example for remote storage:

   ```
   ./cde backup restore --remote-path test/archive.zip
   # the output is similar to list-archive
   ```

Result

Validate that the jobs and resources were restored by running cde job list and cde resource list.

For Cloudera Data Engineering API

Note: Some of the command examples provided use jq to parse the JSON output and make it more readable.

Before you begin

- You must have a valid backup file to restore from. For instructions on backing up Cloudera Data Engineering jobs, see Backing up Cloudera Data Engineering jobs .
- Request an access token and save it as an environment variable to use in API calls. For instructions, see Getting a Cloudera Data Engineering API access token .

Steps

1. Determine the API URL for the virtual cluster that you want to restore the jobs and resources to:
   a. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
   b. Click Administration on the left navigation menu. The Administration page displays.
   c. In the Services column, select the service containing the virtual cluster where you want to restore the job. Then, in the Virtual Clusters column, click the Cluster Details icon.
   d. Click JOBS API URL. The Jobs API URL is copied to the clipboard.
   e. Paste the URL into a text editor to set the Jobs API URL as an environment variable. For example:

```
export CDE_JOBS_API="https://pmjkrgn5.cde-czlmkz4y.na-01.xvp2-7p8o.c
loudera.site/dex/api/v1"
```

2. Restore jobs from the backup file by uploading the backup file to the /admin/import endpoint. You can choose how to handle duplicate job names using the duplicatehandling=<*POLICY*> parameter. The options are:

   error

   Return an error if there are duplicate job names, and abort the restore operation.

   rename

   If a job name in the backup conflicts with an existing job, keep the existing job and rename the restored job by appending a numerical identifier to the job name.

   keep-original

   If a job name in the backup conflicts with an existing job, keep the existing job and do not restore the backed up job.

   For example, to restore a backup named cde-etl-jobs-backup.zip using the rename duplicate handling policy:

```
curl -k \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-X POST "${CDE_JOBS_API}/admin/import" \
-H "accept: application/json" \
-H "Content-Type: multipart/form-data" \
-F "file=@/path/to/cde-etl-jobs-backup.zip;type=application/zip" \
-F duplicatehandling=rename \
| jq
```

For Web UI

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and higher has been updated. The left-hand menu provides easy access to commonly used pages. The steps below vary slightly, for example, the Overview page has been replaced with the Home page. You can also manage a job by clicking Jobs on the left-hand menu, then selecting a Virtual Cluster from a drop-down list at the top of the Jobs page. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

- You must have a valid backup file to restore from. For instructions on backing up Cloudera Data Engineering jobs, see Backing up Cloudera Data Engineering jobs .

Steps

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Jobs in the left navigation menu. The Jobs page displays.
3. From the drop-down list in the upper left-hand corner, select the Virtual Cluster that you want to restore jobs to.
4. Click ⋮ at the top right, and then click Restore Jobs.
5. Click Choose a zip file.
6. Browse to the ZIP file containing the backup of jobs and resources you want to restore, and then click Open.

7. Click Select to restore the backup.

Result

The jobs and resources from the backup file are restored using the rename duplicate handling policy. If a job name in the backup conflicts with an existing job, the restore operation keeps the existing job and renames the restored job by appending a numerical identifier to the job name. If the backup contains Airflow Libraries and Operators, the Airflow Libraries and Operators are restored as well.

Restoring jobs and Airflow Operators and Libraries from a Cloudera Data Engineering version 1.20.3 VC backup

In Cloudera Data Engineering version 1.20.3, the backup and restore function was not implemented yet for Airflow Operators and Libraries. To restore a Cloudera Data Engineering version 1.20.3 Virtual Cluster (VC) from a backup, first create the VC manually, after that create and activate the Airflow Operators and Libraries with the same requirements.txt file that the original VC used, and restore the jobs, resources, and so on from the archive.

Before you begin

Make sure that you have created a backup on a local storage from a Cloudera Data Engineering version 1.20.3 VC with active state Airflow Operators and Libraries with the following command:

cde backup create --include-jobs --include-resources --include-credential-secrets --include-credentials --local-path backup.zip

Optionally, you can validate the backup.

cde backup list-archive --local-path backup.zip

Procedure

1. Create a VC manually with the same configuration (Spark version, CPU, memory, and so on) as the original VC.

   For more information, see *Creating virtual clusters*.

2. Create and activate the Airflow Operators and Libraries manually. Use the same requirements.txt file that the backed up VC used.

   For more information about obtaining the requirements.txt file, see *In-place upgrade with Airflow Operators and Libraries*.

3. Restore the jobs, resources, and so on from the archive.

   cde backup restore --local-path backup.zip

   Important: If there is a cde-airflow-pyenv-*[***ID***]* resource listed in the output in the Failed items section, ignore it.

   Example:

```
$ cde backup restore --local-path backup.zip
     8.2KB/8.2KB 100% [==============================================]
backup.zip
All items restored successfully! Details:
Successfully restored items:
  Job:helloworld ( Created )
  Job:sparkhello ( Created )
  Resource:files ( Created )
  Resource:cde-airflow-pyenv-1711096070 ( Created )
```

Related Information

Creating virtual clusters

In-place Upgrade with Airflow Operators and Libraries

1.19.2

Best Practices

Date published: 2023-06-22
Date modified: 2023-06-22

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Recommendations for scaling Cloudera Data Engineering deployments.....4

Apache Airflow scaling and tuning considerations.....5

Cloudera Data Engineering performance tuning.....6

Best practices for building Apache Spark applications.....6

Recommendations for scaling Cloudera Data Engineering deployments

Your business might experience a sudden increase or drop in demand due to which your Cloudera Data Engineering deployment needs to autoscale. You can scale your Cloudera Data Engineering deployment by either adding new instances of a Cloudera Data Engineering service or Virtual Cluster, or by adding additional resources to the existing ones.

You can scale your Cloudera Data Engineering deployment:

- Vertically - More resources are provisioned within the same instance of a Cloudera Data Engineering service or Virtual Cluster.
- Horizontally - New instances of Cloudera Data Engineering service or Virtual Cluster are provisioned.

The key components of a typical Cloudera Data Engineering service deployment are:

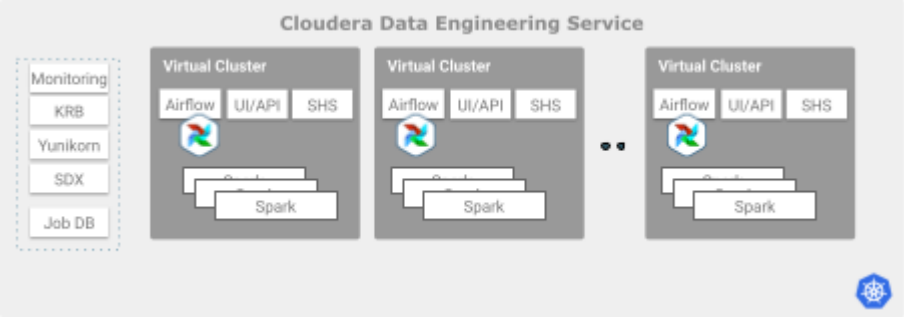

Virtual Clusters provide an isolated autoscaling compute capacity to run Spark and Airflow jobs. You can use Virtual Clusters to isolate individual teams or lines of business by using user-based Access Control Lists (ACLs).

Guidelines for scaling Virtual Clusters

- Each Virtual Cluster requires infrastructure capacity to run various services such as Airflow, API server, and Spark-History-Server (SHS).

  Recommendation: Do not scale horizontally beyond 50 Virtual Clusters within the same Cloudera Data Engineering service.
- Virtual Clusters can actively run hundreds of parallel jobs. In certain scenarios, it might be required to simultaneously submit multiple jobs as per the schedule or due to a burst in demand. In these scenarios, the API server enforces guardrails and limits the number of simultaneous Spark job submissions to 60. Once the jobs move from submission to running state, you can submit more jobs.

  Note: The Airflow jobs are not impacted by the submission limit of 60 simultaneous jobs.

  Recommendation: Distribute simultaneous submission of jobs over time or horizontally scale across multiple Virtual Clusters.

Job submission rate guardrails

When jobs are submitted to the Cloudera Data Engineering API server of a particular Virtual Cluster, it takes time and resources, known as preparing and starting states, before they begin running. This process is called the job submission intake process. To ensure proper resourcing and handling of incoming jobs, guardrails have been set up. By default, the guardrail, or limit is set to 60 simultaneous job submissions. Simultaneous incoming Spark job submissions that exceed 60 return a 429 error message to the client. The example error message is: Failed to submit. Too many requests.

Recommendation:

- Incorporate Cloudera Data Engineering Clients error handling for this error. The CDE CLI exit code is 77. The CDE CLI receives an HTTP 429 response code for the request to the Runtime API Server's REST API. For more information on exit codes, see Cloudera Data Engineering CLI exit codes.
- If needed, increase or decrease the guardrail limit during the Virtual Cluster creation by setting the "limiting.simultaneousJobSubmissions" configuration.

Apache Airflow scaling and tuning considerations

There are certain limitations related to the deployment architecture, and guidelines for scaling and tuning of the deployment, that you must consider while creating or running Airflow jobs (DAGs).

Cloudera Data Engineering deployment architecture

If you use Airflow to schedule and/or develop multi-step pipelines, you must consider the following deployment limitations to scale your environment.

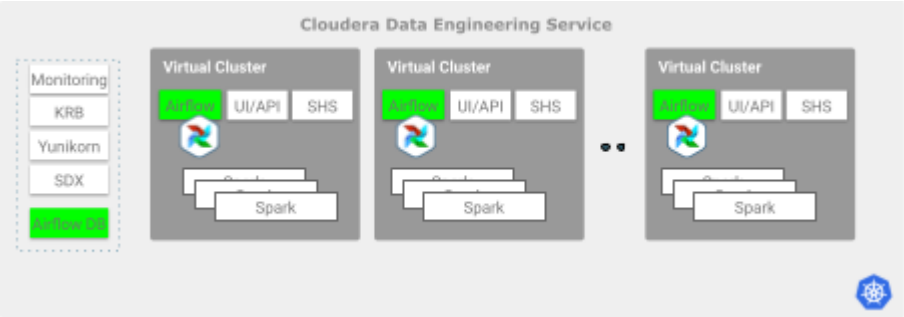

These guidelines describe when to scale horizontally, given total loaded DAGs and/or number of parallel (concurrent) tasks. These apply to all cloud providers.

Cloudera Data Engineering service guidelines

The number of Airflow jobs that can run at the same time is limited by the number of parallel tasks that are triggered by the associated DAGs. The number of tasks can reach up to 250-300 when running in parallel per Cloudera Data Engineering service.

Recommendation: Create more Cloudera Data Engineering services to increase Airflow tasks concurrency beyond the limits noted above.

Virtual Cluster guidelines

- Each Virtual Cluster has a maximum parallel task execution limit of 250. This applies to both Amazon Web Services (AWS) and Azure.

  Note: It is recommended that you do not exceed 300 parallel tasks across all Virtual Clusters.

- For AWS: The recommended maximum number of Airflow jobs loaded per virtual cluster is 1500. As the number of jobs increases, the job page loading time experiences latency along with Airflow job creation time.
- For Azure: The recommended maximum number of Airflow jobs loaded per virtual cluster is 300 to 500. Higher capacity can be achieved but will require manual configuration changes.

  Recommendation: Create more Virtual Clusters to load additional Airflow DAGs.

- The suggested number of Airflow jobs created in parallel is three or less. For jobs that are created in parallel, specify a DAG and/or a task_group for each Airflow operator.

  Recommendation: Distribute the creation of Airflow jobs so that you do not create more than three jobs at the same time.

Cloudera Data Engineering performance tuning

Use local SSD

Using default instances without local SSD does impact performance. Instances with local SSDs generally provide faster performance, but you need to ensure you choose instance types that have large enough local storage for intermediate results.

Avoid instance types with low compute profiles

Make sure there is reasonable headroom (at least 20%) between the per executor memory and core usage and what the instance profile provides.

Disable Spark Analysis

By default, Cloudera Data Engineering collects metrics around CPU, memory, and I/O. This can have a negative impact on performance - especially for long running jobs that take several hours to complete. Profiling metrics are useful in development and testing, but an unnecessary overhead in a production setting.

You can turn this feature off using the Spark Analysis toggle on the job Configuration tab.

Use Spark 3 instead of Spark 2

Spark 3 introduced performance improvements over Spark 2, therefore Cloudera recommends using it whenever possible.

Best practices for building Apache Spark applications

Follow these best practices when building Apache Spark Scala and Java applications:

- Refrain from using withColumn in chain, loop, and calling it multiple times in a single query. Doing so may cause performance issues. To avoid issues, use select() with multiple columns at once. See the Apache Spark API reference linked below for more information.
- Compile your applications against the same version of Spark that you are running.
- Build a single assembly JAR ("Uber" JAR) that includes all dependencies. In Maven, add the Maven assembly plug-in to build a JAR containing all dependencies:

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <configuration>
    <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
    </descriptorRefs>
  </configuration>
  <executions>
    <execution>
      <id>make-assembly</id>
      <phase>package</phase>
      <goals>
```

```
        <goal>single</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

This plug-in manages the merge procedure for all available JAR files during the build. Exclude Spark, Hadoop, and Kafka classes from the assembly JAR, because they are already available on the cluster and contained in the runtime classpath. In Maven, specify Spark, Hadoop, and Kafka dependencies with scope provided. For example:

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.11</artifactId>
  <version>2.4.0.7.0.0.0</version>
  <scope>provided</scope>
</dependency>
```

Cloudera Data Engineering 1.23.0

Accessing the Cloudera Data Engineering service using the CLI

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using the Cloudera Data Engineering command line interface.....5

Downloading the Cloudera Data Engineering command line interface..... 5

Configuring the CLI client..... 5

Cloudera Data Engineering CLI configuration options.....6
Creating and using multiple profiles using CDE CLI..... 7
Cloudera Data Engineering CLI authentication.....8
Cloudera Data Engineering CLI TLS configuration.....10

Cloudera Data Engineering concepts..... 11

Managing Cloudera Data Engineering job resources using the CLI.....12

Creating a Cloudera Data Engineering resource using the CLI..... 12
Uploading files or other assets to a Cloudera Data Engineering resource using the CLI.....13
Deleting a Cloudera Data Engineering resource using the CLI..... 14
Creating and updating Docker credentials.....15
Deleting Docker credentials..... 15
Deleting an Airflow DAG.....16

Managing Cloudera Data Engineering jobs using the CLI.....16

Creating and updating Apache Spark jobs using the CLI..... 16
Creating and updating Apache Airflow jobs using the CLI.....17
Listing jobs using the CLI..... 18
Submitting a Spark job using the CLI.....18
Running raw Scala code in Cloudera Data Engineering..... 19
Submitting an Airflow job using the CLI.....19
Running a Spark job using the CLI.....20
Running a Airflow job using the CLI.....21

Scheduling Spark jobs.....21

Enabling, disabling, and pausing scheduled jobs.....23
Managing the status of scheduled job instances.....23

Managing Sessions in Cloudera Data Engineering using the CLI.....24

Creating a Session using the CDE CLI [Technical Preview].....24
Interacting with a Session using the CDE CLI..... 24
Sessions example for the CDE CLI.....25
Sessions command descriptions..... 26

Cloudera Data Engineering Spark job example.....27

CDE CLI command reference..... 28

CDE CLI Spark flag reference..... 31

CDE CLI Airflow flag reference.....31

CDE CLI list command syntax reference.....32

Managing workload secrets with Cloudera Data Engineering Spark Jobs using the CLI..... 33

Creating a workload secret for Cloudera Data Engineering Spark Jobs using CLI.....33
Updating a workload secret for Cloudera Data Engineering Spark Jobs using CLI.....34
Linking a workload secret to the Cloudera Data Engineering Spark Job definitions using CLI..... 35
Using a workload secret in Spark application code..... 35
Listing an existing workload secret to the Cloudera Data Engineering Spark Job.....36
Deleting a workload secret for Cloudera Data Engineering Spark Jobs using CLI.....36

Using the Cloudera Data Engineering command line interface

Cloudera Data Engineering provides a command line interface (CLI) client. You can use the CLI to create and update jobs, view job details, manage job resources, run jobs, and so on.

Note: The CLI client is not forward compatible. Download the client for the version of the cluster you are accessing. The Cluster Details page for every virtual cluster includes a link to download the CLI client for that cluster version.

The CLI client can also use a password file for non-interactive uses, such as automation frameworks.

Related Information

Using CLI-API to Automate Access to Cloudera Data Engineering

Using Cloudera Data Engineering CLI

Downloading the Cloudera Data Engineering command line interface

Cloudera Data Engineering provides a command line interface (CLI) client.

In addition to the CDE API, you can use the CDE CLI client to access your Cloudera Data Engineering service. Using the CLI, you can manage clusters and applications.

Note: The CLI client is not forward compatible. Download the client for the version of the cluster you are accessing. The Cluster Details page for every virtual cluster includes a link to download the CLI client for that cluster version.

To download the CLI client:

1. Navigate to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera management console.
2. In the Cloudera Data Engineering web console, select an environment.
3. Click the Cluster Details icon for the virtual cluster you want to access.
4. Click the link under CLI TOOL to download the CLI client.
5. Change the permissions on the downloaded cde file to make it executable:

```
chmod +x /PATH/TO/cde
```

Configuring the CLI client

The CDE CLI client uses a configuration file, ~/.cde/config.yaml, to define the default Cloudera Data Engineering virtual cluster to interact with, as well as other configuration parameters.

Before you begin

Make sure that you have downloaded the CDE CLI client.

Procedure

1. Determine the virtual cluster endpoint URL.
   a) Navigate to the Cloudera Data Engineering Overview page.
   b) In the Environments column, select the environment containing the virtual cluster you want to access using the CLI.
   c) In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to access.
   d) Click JOBS API URL to copy the URL to your clipboard.
2. On the host with the CLI client, create or edit the configuration file at ~/.cde/config.yaml. You can create multiple profiles in the ~/.cde/config.yaml file and can be used while running commands. For more information, see Creating and using multiple profiles.

   Note: You can use a custom file location by setting the CDE_CONFIG environment variable. If you include “user: ” in your config.yaml, the CLI tool will not use the access token and will instead keep prompting for a password. If you want to use a credential file, then you must remove the specified user from the profile.
3. In the configuration file, specify the Cloudera user and virtual cluster endpoint as follows:

```
user: <CDP_USER>
vcluster-endpoint: <CDE_VIRTUAL_CLUSTER_ENDPOINT>
```

   The Cloudera user is your workload username.
4. Save the configuration file.
5. If you have not done so already, make sure that the cde file is executable by running chmod +x /*PATH/TO*/cde.
6. Run cde job list to verify your configuration. Enter your workload password when prompted.

   Note: If the directory containing the cde file is not part of your PATH environment variable, you can either add it to your PATH environment variable or use the full path when running the command.

   You can also configure the CLI to use an access token so that you do not need to enter your password each time. For more information, see Cloudera Data Engineering CLI authentication .

What to do next

See for other configuration options.

Cloudera Data Engineering CLI configuration options

The Cloudera Data Engineering CLI can be configured using a configuration file, environment variables, or by command flags.

Configuration Option	Configuration File (~/.cde/config.yaml)	Environment Variable	Command Flag
User	user: <*USERNAME*>	CDE_USER=<*USER*>	--user <*USERNAME*>
Credentials file	credentials-file: </path/to/credentials>	CDE_CREDENTIALS_FILE=</path/to/credentials>	--credentials-file </path/to/credentials>
Skip credentials file detection	skip-credentials-file: true	CDE_SKIP_CREDENTIALS_FILE=true	--skip-credentials-file
Password file	auth-pass-file: <*PASSWORD_FILE*>	CDE_AUTH_PASS_FILE=<*PASSWORD_FILE*>	--auth-pass-file <*PASSWORD_FILE*>
Virtual cluster endpoint	vcluster-endpoint: <*VIRTUAL_CLUSTER*>	CDE_VCLUSTER_ENDPOINT=<*VIRTUAL_CLUSTER*>	--vcluster-endpoint <*VIRTUAL_CLUSTER*>
Disable authentication token caching	auth-no-cache: true	CDE_AUTH_NO_CACHE=true	--auth-no-cache

Configuration Option	Configuration File (~/.cde/config.yaml)	Environment Variable	Command Flag
Authentication token cache file	auth-cache-file: *<TOKEN_CACHE_FILE>*	CDE_AUTH_CACHE_FILE=*<TOKEN_CACHE_FILE>*	--auth-cach-file *<TOKEN_CACHE_FILE>*
CDE configuration profile		CDE_CONFIG_PROFILE=*<PROFILE_NAME>*	--config-profile *<PROFILE_NAME>*

Creating and using multiple profiles using CDE CLI

You can now add a collection of CDE CLI configurations grouped together as profiles, to the config.yaml file. You can use these profiles while running commands. You can set the configurations either at a profile level or at a global level.

About this task

The CDE CLI client uses ~/.cde/config.yaml configuration file to define the default Cloudera Data Engineering virtual cluster to interact with and to define other configuration parameters. CDE CLI now supports the profiling option in the configuration file. A profile is a collection of configurations that you can apply to a CLI command. Multiple named profiles can be stored in the configuration file.

Before you begin

Make sure that you have downloaded the CLI client. For more information, see *Using the Cloudera Data Engineering command line interface*.

Procedure

1. Create or open the ~/.cde/config.yaml file.
2. Add profiles to the config.yaml file based on the following structure:

```
<Global Configurations>
profiles:
- name: <PROFILE NAME 1>
 <PROFILE CONFIGURATIONS>
- name: <PROFILE NAME 2>
 <PROFILE CONFIGURATIONS>
```

Example Configuration file:

```
vcluster-endpoint: https://g7f9bnv2.cde.dev.cldr.work/dex/api/v1
user: cdpuser

profiles:
- name: dev
  user: cdpuser1
  vcluster-endpoint: https://y86gbhv3.cde.dev.cldr.work/dex/api/v1

- name: test
  vcluster-endpoint: https://g7f9bnv2.cde8x.dev.cldr.work/dex/api/v1
```

- Global Configuration: These configurations are set at the global level. The configurations set here are used by default when a profile name is not specified or the configuration is not specified in the profile

- Profile Configuration: These configurations are set at the profile level and overrides the respective configurations set at the global level. You can select the profile either by using the environment variables, or command flags.

  Note: Any profile mentioned under profiles overrides the keys of global configuration and does not replace all the configurations.

  For example, if there are five parameters in global and only three are configured in the selected profile, the final configurations will be three from the selected profile and the remaining from global . In the above test profile example, it does not have the user configuration defined, so if the user selects the test profile, except vcluster-endpoint, which is set in the test profile, all the other configurations are used from global .

  Global configuration is accessed as the default profile name. Hence, you cannot create a profile named default .

3. You can select the profile using the flag or environment variable.

   Flag

   ```
   cde job list --config-profile <PROFILE NAME>
   ```

   Environment variable

   ```
   export CDE_CONFIG_PROFILE=<PROFILE NAME>
   cde job list
   ```

4. [Optional] You can view the active profile in the configuration file.

   ```
   cde profile show-active
   ```

Cloudera Data Engineering CLI authentication

The Cloudera Data Engineering CLI tool supports both interactive and transparent authentication. For interactive authentication, if you have configured the CLI with your workload username, you are prompted for a password. For transparent authentication, the CDE CLI supports a password file, Cloudera access keys, and Cloudera credentials file .

The CDE CLI provides several mechanisms for authentication:

- Cloudera access key stored in a credentials file
- Cloudera access key specified by CLI flag or environment variable
- Interactive prompt for workload password
- Workload password specified by CLI flag or environment variable

Note: If you include “user: ” in your config.yaml, the CLI tool does not use the access token and instead keeps prompting for a password. If you want to use a credential file, then you must remove the specified user from the profile.

Caching

In all cases, the CLI uses the provided credentials to obtain an authentication token for the specified user, and caches it locally in a file on the machine where the CLI is running. You can disable caching of tokens entirely by using the --auth-no-cache CLI flag or the CDE_AUTH_NO_CACHE environment variable.

The cache file location is automatically determined based on the default system user cache:

- Linux: $HOME/.cache/cloudera/cde or $XDG_CACHE_HOME/cloudera/cde/
- macOS: $HOME/Library/Caches/cloudera/cde/
- Windows: %LocalAppData%\cloudera\cde\

If you want to use a custom location, specify it with the --auth-cache-file flag or the CDE_AUTH_CACHE_FILE environment variable. You can use the special string $USERCACHE, which is expanded according to the default system user cache (as listed above, without the /cloudera/cde/ suffix).

Cloudera credentials file

Important: The minimum required roles to obtain an access token are DEUser and EnvironmentUser. EnvironmentAdmin role is not required.

When you generate a Cloudera access key, you can download it to a credentials file:

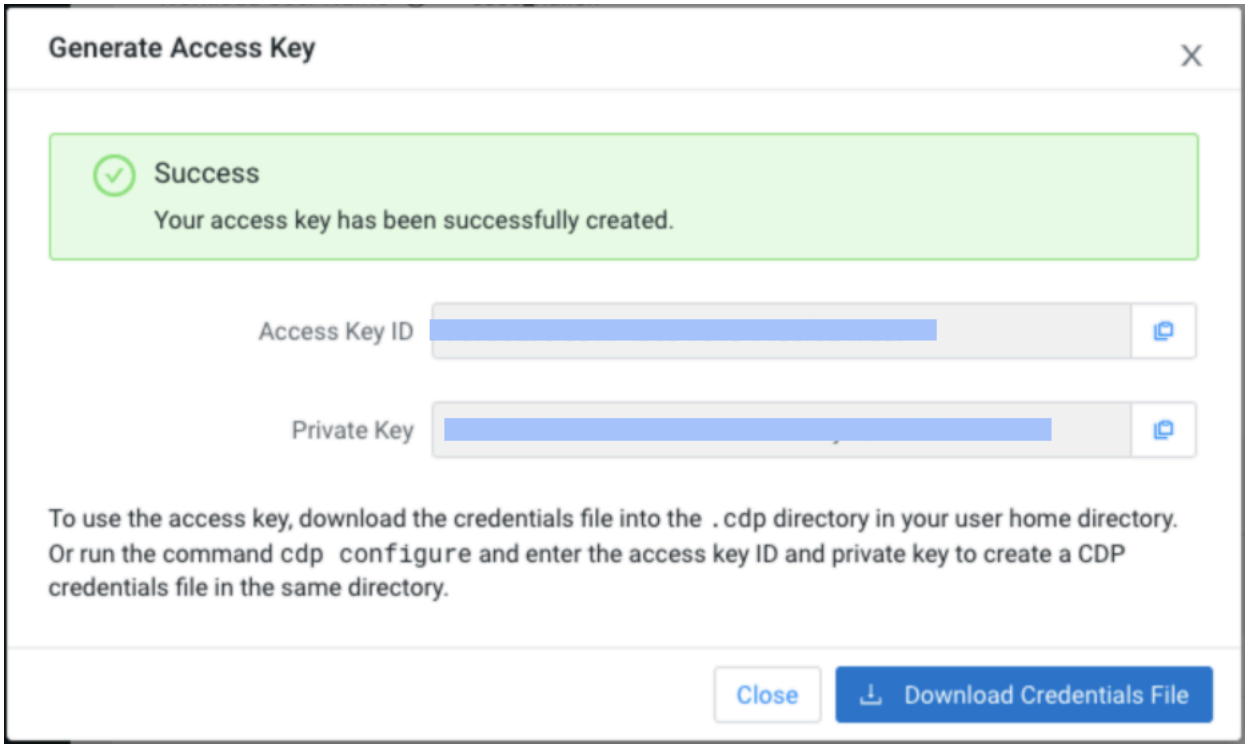

The access key is only displayed and available for download when you first generate it. After you close the dialog, there is no way to recover the key.

Save or copy the credentials file to $HOME/.cdp/credentials on the machine where you are running the CDE CLI. Credentials stored in this file are automatically discovered by both the CDE and CDP CLIs. If a credentials file is found, authentication occurs transparently using the discovered Cloudera access key.

The CDE CLI automatically looks for a Cloudera access key in the following locations in the order given:

1. ./credentials
2. $HOME/.cde/credentials
3. /etc/cde/credentials
4. $HOME/.cdp/credentials

Note: If you include “user: ” in your config.yaml, the CLI tool does not use the access token and instead keeps prompting for a password. If you want to use a credential file, then you must remove the specified user from the profile.

You can override this by using the --credentials-file *</PATH/TO/CREDENTIALS>* CLI flag to specify a different file location.

You can also skip credential discovery by using the --skip-credentials-file flag.

Cloudera access key

If you do not want to use the credentials file, you can specify the access key using environment variables or command line flags as follows:

Table 1: Cloudera access key environment variables and CLI flags

Parameter	Environment variable	CLI flag
Access key ID	CDE_ACCESS_KEY_ID=*<ACCESS_KEY_ID>*	--access-key-id *<ACCESS_KEY_ID>*
Access key secret	CDE_ACCESS_KEY_SECRET=*<ACCESS_KEY_SECRET>*	--access-key-secret string *<ACCESS_KEY_SECRET>*

Workload password prompt

When the CLI requires a new token for a virtual cluster, if you are not using a Cloudera access key, you are prompted for the password for the workload user, identified by the --user CLI flag or the CDE_USER environment variable.

The workload password, for both human and machine users, can be set using the Cloudera User Management console. For more information, see Cloudera workload user.

Workload password file

If you are not using a Cloudera access key, and you do not want to be prompted for your workload password, you can provide a password file. This is not the same as the credentials file discussed before. A password file is a file containing your workload password, and nothing else.

Note: When using a password file, the CLI strips one trailing newline character. If your password actually includes a newline character at the end, add an extra newline at the end of the file.

You can specify the password file by using an environment variable or a command line flag as follows:

Environment variable

CDE_AUTH_PASS_FILE=*</PATH/TO/PASSWORD/FILE>*

Command line flag

--auth-pass-file *</PATH/TO/PASSWORD/FILE>*

Cloudera Data Engineering CLI TLS configuration

All Cloudera Data Engineering virtual cluster endpoints are configured with TLS. In non-production or on-premises environments the TLS certificates are usually signed by a non-production or non-public certificate authority (CA). In these cases, without additional configuration, the CLI tool fails as it attempts to validate the API server's TLS certificate. The CLI provides a TLS configuration when using non-public/non-production CAs.

Specify a file containing the PEM encoded public certificate(s) of the signing CA in one of the following ways:

- add the --tls-ca-certs *[***/PATH/TO/CA.PEM***]* flag on the command line
- define the tls-ca-certs: *[***/PATH/TO/CA.PEM***]* variable in the ~/.cde/config.yaml configuration file
- set the CDE_TLS_CA_CERTS environment variable

Replace *[***/PATH/TO/CA.PEM***]* with the path to a valid ca.pem file.

Certificates for Cloudera Data Engineering are issued and signed by LetsEncrypt and Amazon. If you are using an older system that does not already trust these certificate authorities, you must specify them as trusted using the --tls-ca-certs option. Additionally, if you are using the CDE CLI on Microsoft Windows with the --tls-ca-certs option, the operating system trusted root CAs are ignored. To make sure that your CA certificate file includes all of the necessary certificates, you can download the certificate chain from by doing the following:

1. Go to the Cloudera Data Engineering Overview page.
2. In the Environments column, select the environment containing the virtual cluster you want to interact with.
3. In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to interact with.
4. Click JOBS API URL to copy the link to your clipboard.
5. Paste the link into your browser and press Enter.
6. In your browser, click the lock icon to view the certificate chain.
7. Export the certificate chain as a Base64-encoded X.509 certificate (.CER). Save it as ca.pem.

Tip: When using the CLI on Windows, remember to use path styles such as C:\Users\janeblogs\.cde\ca.pem when referencing local files.

Cloudera Data Engineering concepts

Learn about some basic concepts behind Cloudera Data Engineering service to better understand how you can use the command line interface (CLI).

Cloudera Data Engineering has three main concepts:

job

A 'job' is a definition of something that Cloudera Data Engineering can run. For example, the information required to run a jar file on Spark with specific configurations.

job run

A 'job run' is an execution of a job. For example, one run of a Spark job on a Cloudera Data Engineering cluster.

session

A 'session' is an interactive short-lived development environment for running Spark commands to help you iterate upon and build your Spark workloads.

resource

A 'resource' refers to a job dependency that must be available to jobs at runtime. Currently the following resource types are supported:

- files is a directory of files that you can upload to Cloudera Data Engineering pods into a standard location (/app/mount). This is typically for application (for example, .jar or .py files) and reference files, and not the data that the job run will operate on. Multiple files resources can be referenced in a single job.
- python-env is used to provide custom Python dependencies to the job as a Python virtual environment which is automatically configured. Up to one python-env resource can be specified per job definition.
- custom-runtime-image is a custom Docker container image

  Note: Custom Docker container images is a *Technical Preview* feature. Contact your Cloudera account representative to enable access to this feature.

In addition, to support jobs with custom requirements, Cloudera Data Engineering also allows users to manage credentials which can be used at job run time. Currently, only custom Docker registry credentials are supported.

Submitting versus running a job

The cde spark submit and cde airflow submit commands automatically create a new job and a new resource, submit the job as a job run, and when the job run terminates they delete the job and resources.

A cde job run requires a job and all necessary resources to be created and uploaded to the Cloudera Data Engineering cluster beforehand. The advantage of creating resources and jobs ahead of time is that resources can be reused across jobs, and that jobs can be run using only a job name.

Managing Cloudera Data Engineering job resources using the CLI

A *resource* in Cloudera Data Engineering is a named collection of files or other resources referenced by a job. The files can include application code, configuration files, or any other arbitrary files required by a job. A resource can also be a Python virtual environment, or a custom Docker container image.

Note: Custom Docker container images is a *Technical Preview* feature. Contact your Cloudera account representative to enable access to this feature.

You can think of resources as any supporting files, libraries, or images that a Cloudera Data Engineering job requires to run. Resources can be created and deleted, and files can be added to and deleted from a resource as needed.

A resource can also be a Python virtual environment specification (as a requirements.txt file), or a custom Docker container image.

Before continuing, make sure that you have downloaded and configured the CLI client.

Creating a Cloudera Data Engineering resource using the CLI

A *resource* in Cloudera Data Engineering is a named collection of files or other assets referenced by a job, including application code, configuration files, or any other arbitrary files required by a job. A resource can also be a Python virtual environment, or a custom Docker container image.

Before you begin

Note: Custom Docker container images is a *Technical Preview* feature. Contact your Cloudera account representative to enable access to this feature.

Make sure that you have downloaded and configured the CLI client.

Procedure

1. Create a resource using the cde resource create command.

   The cde resource create syntax is as follows:

   ```
   cde resource create [flags]
   ```

   You can view the list of flags by running cde resource create --help, or you can view the CDE CLI reference documentation.

   Example: Create a file resource

   ```
   cde resource create --name cde-file-resource --type files
   ```

   Example: Create a Python virtual environment resource

   ```
   cde resource create --name cde-python-env-resource --type python-env --p
   ython-version python3
   ```

   Note:

   You can specify a PyPi mirror for a Python virtual environment resource using the --pypi-mirror flag. Note, that this requires network access to the mirror from the Cloudera environment.

   Example: Create a custom Docker container image resource

   ```
   cde resource create --name cde-container-image-resource --type custom-ru
   ntime-image
   ```

2. Verify that the resource was created by running cde resource list.

Uploading files or other assets to a Cloudera Data Engineering resource using the CLI

A *resource* in Cloudera Data Engineering is a named collection of files or other assets referenced by a job, including application code, configuration files, or any other arbitrary files required by a job. A resource can also be a Python virtual environment, or a custom Docker container image.

Before you begin

Note: Custom Docker container images is a *Technical Preview* feature. Contact your Cloudera account representative to enable access to this feature.

Make sure that you have downloaded and configured the CLI client.

Make sure that you have created a resource.

Procedure

1. Upload assets to a resource using the cde resource upload command.

   The cde resource upload syntax is as follows:

   ```
   cde resource upload [flags]
   ```

   You can view the list of flags by running cde resource upload --help, or you can view the CDE CLI reference documentation.

   Note: For Python environment resources, you can only upload a requirements.txt file. Python environment resources do not support arbitrary file upload. If the local file is named something other than requirements.txt, you must add the flag --resource-path requirements.txt to the command.

   Example: Upload a file resource

   ```
   cde resource upload --name [***RESOURCE_NAME***] --local-pat
   h [***LOCAL_PATH***] [--resource-path [***PATH_IN_RESOURCE***]]
   ```

   Use repeated local path flags, and/or */?/[a-z] wildcards, to specify multiple files. Use quotes around the local path when including wildcards, for example, --local-path "*.jar". For a single file --resource-path is used for the resource filename. For multiple files --resource-path is used for the resource directory.

   Example: Upload a Python virtual environment resource

   ```
   cde resource upload --name cde-python-env-resource --local-path ${HOME}/
   requirements.txt
   ```

   Example: Upload a custom Docker container image resource

   ```
   cde resource upload --name cde-container-image-resource --type custom-ru
   ntime-image
   ```

   Example: Upload a file for a file resource

   ```
   cde resource upload --name cde-file-resource --local-path /path/to/local/
   file
   ```

   Example: Upload and extract archive to resource

   Currently supported archive file formats are : .zip and .tar.gz

   ```
   cde resource upload-archive --name cde-file-resource --local-path /path/
   to/local/file
   ```

2. Verify that the file is included in the resource by running cde resource describe --name *<RESOURCE_NAME>*.

Deleting a Cloudera Data Engineering resource using the CLI

A *resource* in Cloudera Data Engineering is a named collection of files or other resources referenced by a job, including application code, configuration files, or any other arbitrary files required by a job. A resource can also be a Python virtual environment, or a custom Docker container image. Resources can be deleted using the CLI.

Before you begin

- Make sure that you have downloaded and configured the CLI client.
- Make sure that the resource you are deleting is no longer needed for any jobs. (Resources cannot be deleted if they are referenced in one or more jobs)

Procedure

1. Run cde resource describe --name <*RESOURCE_NAME*>. View the output and confirm that the resource you want to delete is no longer required, and does not contain any files that you need to retain.
2. Delete the resource by running cde resource delete --name <*RESOURCE_NAME*>
3. Verify that the resource is deleted by running cde resource list and confirming that the resource is no longer listed.

Creating and updating Docker credentials

To allow the use of private Docker registries, Cloudera Data Engineering supports the creation and management of credentials. These are stored securely in the Kubernetes cluster as secrets and cannot be accessed by end users directly. Credentials are attached to job runs automatically by the Cloudera Data Engineering backend.

About this task

Note: Custom Docker container images is a *Technical Preview* feature. Contact your Cloudera account representative to enable access to this feature.

Procedure

1. To create a new Docker credential:

```
cde credential create --name <CRED_NAME> --type docker-basic --docker-se
rver <REGISTRY_URL_OR_HOSTNAME> --docker-username <DOCKER_USER>
```

2. Enter the Docker registry password when you are prompted.

   An optional --description field allows you to annotate the credential with a human readable description.
3. Run cde credential list to verify that the credential was created:

```
cde credential list [--filter <FILTER>]
```

   For more information on filtering syntax, see CDE CLI list command syntax reference on page 32.
4. If you want to update a credential, use the cde credential update command.

   This command allows you to update the secret content, the credential description, or both.

```
cde credential update --name <CRED_NAME> [--docker-serve
r <REGISTRY_URL_OR_HOSTNAME> --docker-username <DOCKER_USER>] [--descrip
tion "<DESC>"]
```

Deleting Docker credentials

To allow the use of private Docker registries, Cloudera Data Engineering supports the creation and management of credentials. These are stored securely in the Kubernetes cluster as secrets and cannot be accessed by end users directly. Credentials are attached to job runs automatically by the Cloudera Data Engineering backend.

Before you begin

- Make sure that you have downloaded and configured the CLI client.
- Make sure that the credential you are deleting is no longer needed for any jobs.

About this task

Note: Custom Docker container images is a *Technical Preview* feature. Contact your Cloudera account representative to enable access to this feature.

Procedure

1. Delete the credential by running cde credential delete --name *<CRED_NAME>*
2. Run cde credential list to verify that the credential was deleted:

```
cde credential list [--filter <FILTER>]
```

For more information on filtering syntax, see CDE CLI list command syntax reference on page 32.

Deleting an Airflow DAG

You can delete unused Airflow DAGs using the Cloudera Data Engineering command line interface (CLI).

About this task

The default process of removing Cloudera Data Engineering resources is to delete them together with the jobs owning them, using the cde job delete command. The cde airflow delete-dag command is a fallback for when Airflow gets into an unexpected situation and you have to remove a DAG with no associated Airflow job.

Procedure

To delete a DAG from Airflow that is not associated with a job, use the cde airflow delete-dag command:

```
cde airflow delete-dag --dag-id <DAG_ID>
```

Managing Cloudera Data Engineering jobs using the CLI

A *job* in Cloudera Data Engineering is a definition of something that Cloudera Data Engineering can run. For example, the information required to run a JAR file on Spark with specific configurations. A 'job run' is an execution of a job. For example, one run of a Spark job on a Cloudera Data Engineering cluster.

Creating and updating Apache Spark jobs using the CLI

The following example demonstrates how to create a Spark application in Cloudera Data Engineering using the command line interface (CLI).

Before you begin

Make sure that you have downloaded the CLI client. For more information, see Using the Cloudera Data Engineering command line interface. While creating a job if you want to use the [--data-connector] flag, you must obtain the name of the data connector from the Cloudera Data Engineering UI by navigating to Administration > click Service Details icon of the Cloudera Data Engineering Service > Data Connectors tab.

Procedure

1. Run the cde job create command as follows:

```
cde job create --application-file <PATH_TO_APPLICATION_JAR> --mount-1-r
esource <YOUR_CDE_RESOURCE> --class <APPLICATION_CLASS> [--default-vari
```

```
able name=value] [--data-connector name] --name <JOB_NAME> --num-execu
tors <NUM_EXECUTORS> --type spark
```

To see the full command syntax and supported options, run cde job create --help.

The application file can either be a local JAR file inside the Spark container or can be mounted from a Cloudera Data Engineering Resource. If it is from a Cloudera Data Engineering Resource, the --application-file points to the file from within the Cloudera Data Engineering Resource. If you want to configure the job to pick up the application file from a Cloudera Data Engineering Repository, provide the name of the repository with the --mount-1-resource flag and provide the --application-file path as you would do with a Cloudera Data Engineering Resource.

With [--default-variable] flags you can replace strings in job values. Currently the supported fields are:

- Spark application name
- Spark arguments
- Spark configurations

For a variable flag name=value any substring {{{name}}} in the value of the supported field gets replaced with value. These can be overriden by the [--variable] flag during the job run.

Using the [--data-connector] flag, you can specify the name of the data connector. Currently, only the Ozone type data connector is supported and it must be created before the job run.

2. Run cde job describe to verify that the job was created:

```
cde job describe --name <JOB_NAME>
```

3. If you want to update the job configuration, use the cde job update command.
   For example, to change the number of executors:

```
cde job update --name test_job --num-executors 15
```

   To see the full command syntax and supported options, run cde job update --help.

4. To verify the updated configuration, run cde job describe again:

```
cde job describe --name <JOB_NAME>
```

Creating and updating Apache Airflow jobs using the CLI

The following example demonstrates how to create an Airflow DAG in Cloudera Data Engineering using the command line interface (CLI).

Before you begin

Make sure that you have downloaded the CLI client. For more information, see Using the Cloudera Data Engineering command line interface.

About this task

Procedure

1. Run the cde job create command as follows:

```
cde job create --name <JOB_NAME> --type airflow --dag-file <DAG_FILE> --m
ount-1-resource <YOUR_DAG_RESOURCE> [OTHER AIRFLOW FLAGS...]
```

<DAG_file>

is a reference to a file within a Cloudera Data Engineering resource

To see the full command syntax and supported options, run cde job create --help.

Note: Airflow DAGs manage their own schedules and so their schedules cannot be set through the CLI.

2. Run cde job describe to verify that the job was created:

```
cde job describe --name <JOB_NAME>
```

3. If you want to update the job configuration, use the cde job update command.
For example, to change the number of executors:

```
cde job update --name test_job
```

To see the full command syntax and supported options, run cde job update --help.

4. To verify the updated configuration, run cde job describe again:

```
cde job describe --name <JOB_NAME>
```

Listing jobs using the CLI

To view existing applications, run cde job list. To view details for a specific application, run cde job describe --name <*JOB_NAME*>

Submitting a Spark job using the CLI

The following example demonstrates how to submit a JAR or Python file to run on Cloudera Data Engineering Spark in Cloudera Data Engineering using the command line interface (CLI).

About this task

Using the cde spark submit command is a quick and efficient way of testing a spark job, as it spares you the task of creating and uploading resources and job definitions before running the job, and cleaning up after running the job.

This command is recommended only for JAR or Python files that need to be run just once, because the the file is removed from Spark at the end of the run. To manage jobs that need to be run more than once, or that contain schedules, use cde job run instead of this command.

Procedure

To submit a JAR or Python file to run on Cloudera Data Engineering Spark, use the CLI command:

```
cde spark submit <JAR/Python file> [args...] [Spark flags...] [--job-name <j
ob name>] [--hide-logs]
```

You can use [--job-name <job name>] to specify the same Cloudera Data Engineering job name for consecutive cde spark submit commands. To see the full command syntax and supported options, run cde spark submit --help.

For example:

To submit a job with a local JAR file:

```
cde spark submit my-spark-app-0.1.0.jar 100 1000 --class com.company.app.spa
rk.Main
```

The CLI displays the job run ID followed by the driver logs, unless you specified the --hide-logs option. The script returns an exit code of 0 for success or 1 for failure.

Running raw Scala code in Cloudera Data Engineering

Cloudera Data Engineering supports running raw Scala code from the command line, without compiling it into a JAR file. You can use the cde spark submit command to run a .scala file. Cloudera Data Engineering recognizes the file as Scala code and runs it using spark-shell in batch mode rather than spark-submit.

Limitations:

- When setting the Log Level from the user interface, the setting is not applied to the raw Scala jobs.
- Do not use package <something> in the raw Scala job file as Raw Scala File is used for Scripting and not for Jar development and packaging.

Note: Cloudera Data Engineering does not currently support interactive sessions. The Scala code runs in batch mode spark-shell.

Run cde spark submit as follows to run a Scala file:

```
cde spark submit filename.scala --jar <JAR_DEPENDENCY_1> --j
ar <JAR_DEPENDENCY_2> ...
```

Submitting an Airflow job using the CLI

The following example demonstrates how to submit a DAG file to immediately run on Cloudera Data Engineering Airflow in Cloudera Data Engineering using the command line interface (CLI).

About this task

Using the cde airflow submit command is a quick and efficient way of testing an Airflow job, as it spares you the task of creating and uploading resources and job definitions before running the job, and cleaning up after running the job.

This command is recommended only for Airflow DAGs that need to be run just once, because the DAG is removed from Airflow at the end of the run. To manage Airflow DAGs that need to be run more than once, or that contain schedules, use cde job run instead of this command.

Procedure

To submit a DAG file to run on Cloudera Data Engineering Airflow, use the CLI command:

```
cde airflow submit <DAG python file> [--config-json <json-string>]* [--job-n
ame <job name>]
```

To see the full command syntax and supported options, run cde airflow submit --help.

For example:

To submit a job with a local DAG file:

```
cde airflow submit my-dag.py
```

When the job has been submitted the CLI displays the job run ID, waits for the job to terminate, and returns an exit code of 0 for success or 1 for failure.

Running a Spark job using the CLI

The following example demonstrates how to run a Cloudera Data Engineering Spark job using the command line interface (CLI).

Before you begin

Make sure that the Spark job has been created and all necessary resources have been created and uploaded.

Note: Custom Docker container images is a *Technical Preview* feature. Contact your Cloudera account representative to enable access to this feature.

About this task

Using the cde job run requires more preparation on the target environment compared to the cde spark submit command. Whereas cde spark submit is a quick and efficient way of testing a Spark job during development, cde job run is suited for production environments where a job is to be run multiple times, therefore removing resources and job definitions after every job run is neither necessary, nor viable.

Procedure

To run a Spark job, run the following command:

```
cde job run --name <job name> [Spark flags...] [--wait] [--variable name=val
ue...]
```

- With [Spark flags...] you can override the corresponding job values. Spark flags that can be repeated replace the original list, except for --conf which only adds or replaces values for the given keys.
- With [--variable] flags you can replace strings in job values. Currently the supported fields are:
  - Spark application name
  - Spark arguments
  - Spark configurations

  For a variable flag name=value any substring {{{name}}} in the value of the supported field gets replaced with value.
- A custom runtime Docker image can be specified for the job using the --runtime-image-resource-name flag, which has to refer to the name of a custom image resource that has already been created.
- GPU Acceleration (Technical Preview): Using [--enable-gpu-acceleration] you can accelerate your Spark jobs using GPUs. You can use [--executor-node-selector "nvidia.com/gpu=A100"] and [--executor-node-tolerat ion "nvidia.com/gpu=true"] options to configure selectors and tolerations if you want to run the job on specific GPU nodes. When this job is run, this particular job will request GPU resources.

  Warning: You must ensure this virtual cluster has been configured with GPU resource quota. Otherwise, the jobs will be in the Pending state as no GPU resource can be allocated to the pod.

  ```
  cde job run --name example-pi \
  --enable-gpu-acceleration \
  --executor-node-selector "nvidia.com/gpu=A100" \
  --executor-node-toleration "nvidia.com/gpu=true"
  ```

By default the command returns the job run ID as soon as the job has been submitted.

Optionally, you can use the --wait switch to wait until the job run ends and returns a non-zero exit code if the job run was not successful.

Running a Airflow job using the CLI

The following example demonstrates how to run a Cloudera Data Engineering Airflow job using the command line interface (CLI).

Before you begin

Make sure that the job has been created and all necessary resources have been created and uploaded.

Note: Custom Docker container images is a *Technical Preview* feature. Contact your Cloudera account representative to enable access to this feature.

About this task

Using the cde job run requires more preparation on the target environment compared to the cde airflow submit command. Whereas cde airflow submit is a quick and efficient way of testing an Airflow job during development, cde job run is suited for production environments where a job is to be run multiple times, therefore removing resources and job definitions after every job run is neither necessary, nor viable.

Procedure

To run an Airflow job, run the following command:

```
cde job run --name <job name> [--config <key=value>]* [--wait]
```

Airflow configs provided at job run time will override the corresponding job configs.

By default the command returns the job run ID as soon as the job has been submitted.

Optionally, you can use the --wait switch to wait until the job run ends and returns a non-zero exit code if the job run was not successful.

Scheduling Spark jobs

Spark jobs can optionally be scheduled so that they are automatically run on an interval. Cloudera Data Engineering uses the Apache Airflow scheduler to create the schedule instances.

About this task

Note:

Airflow DAGs manage their own schedules, therefore Airflow job schedules cannot be set in this way, other than by using the operational commands pause, unpause, clear, mark-success.

Before you begin

Make sure that the Spark job has been created and all necessary resources have been created and uploaded.

Note: Custom Docker container images is a *Technical Preview* feature. Contact your Cloudera account representative to enable access to this feature.

Procedure

1. Define a running interval for your Spark job:

   The schedule interval is defined by a cron expression. Intervals can be regular, such as daily at 3 a.m., or irregular, such as hourly but only between 2 a.m. and 6 a.m. and only on weekdays. You can provide the cron expression directly or you can generate it using flags.

   Note: Scheduled job runs start at the end of the first full schedule interval after the start date, at the end of the scheduled period. For example, if you schedule a job with a daily interval with a start_date of 14:00, the first scheduled run is triggered at the end of the next day, after 23:59:59. However if the start_date is set to 00:00, it is triggered at the end of the same day, after 23:59:59.

   Available schedule interval flags are:

   --cron-expression
   A cron expression that is provided directly to the scheduler. For example, 0 */1 * * *

   --every-minutes
   Running frequency in minutes. Valid values are 0-59. Only a single value is allowed.

   --every-hours
   Running frequency in hours. Valid values are 0-23. Only a single value is allowed.

   --every-days
   Running frequency in days. Valid values are 1-31. Only a single value is allowed.

   --every-months
   Running frequency in months. Valid values are 1-12. Only a single value is allowed.

   --for-minutes-of-hour
   The minutes of the hour to run on. Valid values are 0-59. Single value, range (e.g.: 1-5), or list (e.g.: 5,10) are allowed.

   --for-hours-of-day
   The hours of the day to run on. Valid values are 0-23. Single value, range (e.g.: 1-5), or list (e.g.: 5,10) are allowed.

   --for-days-of-month
   The days of the month to run on. Valid values are 1-31. Single value, range (e.g.: 1-5), or list (e.g.: 5,10) are allowed.

   --for-months-of-year
   The months of the year to run on. Valid values are 1-12 and JAN-DEC. Single value, range (e.g.: 1-5), or list (e.g.: APR,SEP) are allowed.

   --for-days-of-week
   The days of the week to run on. Valid values are SUN-SAT and 0-6. Single value, range (e.g.: 1-5), or list (e.g. TUE,THU) are allowed.

   For example, to set the interval as hourly but only between 2 a.m. and 6 a.m. and only on weekdays, use the command:

   ```
   cde job create --name test_job --schedule-enabled=true --every-hours 1 --
   for-minutes-of-hour 0 --for-hours-of-day 2-6 --for-days-of-week MON-FRI --
   schedule-start 2021-03-09T00:00:00Z
   ```

   Or, equivalently, using a single cron expression:

   ```
   cde job create --name test_job --schedule-enabled=true --cron-expression
    '0 2-6/1 * * MON-FRI'  --schedule-start 2021-03-09T00:00:00Z
   ```

2. Define a time range for your Spark job:

   The schedule also defines the range of time that instances can be created for. The mandatory --schedule-start flag timestamp tells the scheduler the date and time from which the scheduling begins. The optional --schedule-end flag timestamp tells the scheduler the last date and time at which the schedule is active. If --schedule-end is not specified, the job runs at the scheduled interval until it is stopped manually.

   Note: Timestamps must be specified in ISO-8601 UTC format ('yyyy-MM-ddTHH:mm:ssZ'). UTC offsets are not supported.

   For example, to create a schedule that runs at midnight for each day of a single week, use the following command:

```
cde job create --name test_job --schedule-enabled=true --every-days 1 --
for-minutes-of-hour 0 --for-hours-of-day 0 --schedule-start 2021-03-09T0
0:00:00Z --schedule-end 2021-03-15T00:00:00Z
```

Enabling, disabling, and pausing scheduled jobs

Using the Cloudera Data Engineering command line interface (CLI), you can enable, disable, or pause scheduled job runs.

Before you begin

Note:

Disabling the schedule removes all record of prior schedule instances.

Note:

Pausing and unpausing the schedule does not remove the record of prior schedule instances.

Procedure

- To enable or disable a job schedule, use the following command:

```
cde job (create | update) --name <job name> --schedule-enabled=(true | f
alse) ...
```

- To pause a job schedule upon schedule creation:

```
cde job (create | update) --name <job name> --schedule-enabled=true --sc
hedule-paused=true ...
```

- To pause an existing job schedule:

```
cde job schedule pause --name <job name>
```

  or

```
cde job schedule pause-all
```

- To unpause an existing job schedule:

```
cde job schedule unpause --name <job name>
```

Managing the status of scheduled job instances

Using the Cloudera Data Engineering command line interface (CLI), you can clear the statuses of a range of scheduled instances or mark a scheduled job instance as successful.

Procedure

- To clear the status of a range of scheduled instances, run the following command:

```
cde job schedule clear [--schedule-start <start of clear period>] [--sch
edule-end <end of clear period>]
```

- To mark a single scheduled instance as successful, run the following command:

```
cde job schedule mark-success --execution-date <execution date of schedu
led instance>
```

  where <execution date of scheduled instance> is the timestamp that the instance was scheduled for, not when it actually ran.

Managing Sessions in Cloudera Data Engineering using the CLI

A Cloudera Data Engineering Session is an interactive short-lived development environment for running Spark commands to help you iterate upon and build your Spark workloads.

Creating a Session using the CDE CLI [Technical Preview]

The cde session create command allows you to create a new Session.

Before you begin

Ensure that you are using a version of Cloudera Data Engineering 1.19 or higher for your Virtual Cluster and ensure that you have downloaded the CLI client. For more information, see Using the Cloudera Data Engineering command line interface.

Procedure

Run the following command in the CDE CLI:

```
cde session create --name <session-name> --type <pyspark/spark-scala>
```

Interacting with a Session using the CDE CLI

Once your Session has been created, you can interact with it using the cde sessions interact command.

About this task

Below is an example that demonstrates how to interact with a PySpark or Scala Session in Cloudera Data Engineering using the CLI.

Before you begin

Ensure that you are using a version of Cloudera Data Engineering 1.19 or higher for your Virtual Cluster and ensure that you have set up the CDE CLI client and have created a Cloudera Data Engineering Session using the CLI or UI. For more information, see Using the Cloudera Data Engineering command line interface.

Procedure

Run the following command in the CDE CLI:

```
cde session interact --name <session-name>
```

Sessions example for the CDE CLI

In this example, a Session is created using the Cloudera Data Engineering CLI with resources specified during creation. In this example, python environment, files, Git repository, and workload credentials resources are used.

Note: Access files from the /app/mount after they are mounted. Access Secrets from /etc/dex/secrets/<secret -name>.

```
> cde session create --name resources --type pyspark --python-env-resource-n
ame example-virtual-env --runtime-image-resource-name docker-image --mount-1
-resource octocat --mount-2-resource example-files --mount-3-resource exampl
e-data --workload-credential workload-cred --workload-credential workload-cr
ed-2
{
  "name": "resources",
  "type": "pyspark",
  "creator": "csso_surya.balakrishnan",
  "created": "2023-10-06T03:13:03Z",
  "mounts": [
    {
      "dirPrefix": "/",
      "resourceName": "octocat"
    },
    {
      "dirPrefix": "/",
      "resourceName": "example-files"
    },
    {
      "dirPrefix": "/",
      "resourceName": "example-data"
    }
  ],
  "lastStateUpdated": "2023-10-06T03:13:03Z",
  "state": "starting",
  "interactiveSpark": {
    "id": 1,
    "driverCores": 1,
    "executorCores": 1,
    "driverMemory": "1g",
    "executorMemory": "1g",
    "numExecutors": 1,
    "pythonEnvResourceName": "example-virtual-env"
  },
  "workloadCredentials": [
    "workload-cred",
    "workload-cred-2"
  ],
  "runtimeImageResourceName": "docker-image"
}

> ./cde session interact --name resources
Starting REPL...
Waiting for the session to go into an available state...
Connected to Cloudera Data Engineering...
Press Ctrl+D (i.e. EOF) to exit
```

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\
      /_/
Type in expressions to have them evaluated.

>>> os.listdir("/app/mount")
['.git', 'README', 'access-logs-ETL-iceberg.py', 'access-logs-ETL.py', 'a
ccess-logs.txt', 'cdeoperator.py', 'pyspark-batch-job.py', 'pyspark_wordcoun
t.py', 'spark-load-data.py', 'word_count_templates.txt', 'wordcount_input_1.
txt', 'dex-spark-driver-template-txckdpxp.yaml', 'dex-spark-executor-templat
e-txckdpxp.yaml']

>>> sec_path = "/etc/dex/secrets/workload-cred/key1"
>>> with open(sec_path) as f:
...     for line in f:
...         print(line)
value1
>>> sec_path = "/etc/dex/secrets/workload-cred-2/key2"
>>> with open(sec_path) as f:
...     for line in f:
...         print(line)
value2
>>> import pandas

>>> dates = pandas.date_range("20130101", periods=6)

>>> dates
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')
```

Sessions command descriptions

The Cloudera Data Engineering command reference is shown below.

Command	Definition
cde session create	Creates a Cloudera Data Engineering session. Sessions are identified by a user-specified name. Sessions have a type that defines the engine that the Session will run on. The 'spark-scala'[Scala REPL] and 'pyspark'[Python REPL] types are currently supported.
cde session interact	Connects to a running session in a Spark shell similar to the interface and submit statements.
cde session kill	Ends a session. The Spark driver and executor processes are stopped.
cde session delete	Deletes a session and removes all references to the session. Logs will no longer be accessible.
cde session list	Lists all sessions. The --output flag can be used to control the output format. You can use the `--output string output format ("table" or "json") (default "table")` flag to specify whether the Session's output must be in a table or JSON format
cde session statements	Lists session statements. The --output flag can be used to control the output format.
cde session describe –name <session_name>	Describes the session. The command is used as an input to name the session.

Cloudera Data Engineering Spark job example

In this example there is a local Spark jar my-app-0.1.0.jar, and a local reference file my-ref.conf that the Spark job opens locally as part of its execution. The Spark job reads data from the location in the first argument and writes data to the location in the second argument. There is also a custom Spark configuration for tuning performance.

1. Make your job available for running in one of the following ways:

   You can submit the job directly to Cloudera Data Engineering and have it run the job once, using the spark submit command. In this case no permanent resources are created on Cloudera Data Engineering subsequently no cleanup is necessary after the job run. This is ideal when testing a job.

   ```
   cde spark submit my-app-0.1.0.jar \
     --file my-ref.conf \
     --conf spark.sql.shuffle.partitions=1000
   ```

   If you plan to run the same job several times it is is a good idea to create and upload the resource and job and then run it on Cloudera Data Engineering using the job run command. This is the preferable method in production environments.

   ```
   > cde resource create --name my-resource
   > cde resource upload --name my-resource --local-path my-app-0.1.0.jar
    109.7MB/109.7MB 100% [==============================================] my-
   app-0.1.0.jar
   > cde resource upload --name my-resource --local-path my-ref.conf
      135.0b/135.0b 100% [==============================================] my-
   ref.conf
   > cde job create \
     --name my-job \
     --type spark \
     --mount-1-resource my-resource \
     --application-file my-app-0.1.0.jar \
     --conf spark.sql.shuffle.partitions=1000 \
   > cde job run --name my-job
   {
     "id": 1
   }
   > cde run describe --id 1 | jq -r '.status'
   starting
   ...
   > cde run describe --id 1 | jq -r '.status'
   finished
   ```

2. Schedule your job:

   As the above created job stays in Cloudera Data Engineering permanently until you delete it, you can schedule it to run regularly at a predefined time. This example schedules your job to run daily at midnight, starting from January 1, 2021:

   ```
   > cde job update \
     --name my-job \
     --schedule-enabled=true \
     --schedule-start 2021-01-01T00:00:00Z \
     --every-days 1 \
     --for-minutes-of-hour 0 \
     --for-hours-of-day 0
   ```

CDE CLI command reference

The Cloudera Data Engineering command line syntax is shown below. You can view additional syntax help by adding --help after any command.

cde command

```
Usage:
  cde [command]

Available Commands:
  airflow     Airflow commands
  backup      Create and Restore CDE backups
  credential  Manage CDE credentials
  help        Help about any command
  job         Manage CDE jobs
  profile     Manage CDE configuration profiles
  repository  Manage CDE repository
  resource    Manage CDE resources
  run         Manage CDE runs
  session     Manage CDE sessions
  spark       Spark commands
Flags:
      --access-key-id string         access key identifier
      --access-key-secret string     access key secret
      --auth-cache-file string       token file cache location (default "$
USERCACHE/token-cache")
      --auth-no-cache                do not cache authentication tokens
      --auth-pass-file string        authentication password file location
      --cdp-endpoint string          CDP API endpoint (default depends on
 CDP Control Plane region)
      --config-profile string        CDE configuration profile name
      --credentials-file string      CDP credentials file location
      --credentials-profile string   CDP credentials profile name (default
 "default")
  -h, --help                         help for cde
      --hide-progress-bars           hide progress bars for file uploads
      --region string                CDP Control Plane region ("us-west-1",
 "eu-1" or "ap-1") (default "us-west-1")
      --skip-credentials-file        skip CDP credentials file discovery
      --tls-ca-certs string          additional PEM-encoded CA certificates
      --tls-insecure                 skip verification of API server TLS cer
tificate
      --user string                  CDP user to authenticate as
      --vcluster-endpoint string     CDE virtual cluster endpoint
  -v, --verbose                      verbose logging
      --version                      version for cde

Use "cde [command] --help" for more information about a command.
```

cde airflow command

```
Usage:
  cde airflow [command]

Available Commands:
```

```
  activate-pyenv   Deprecated, please use /admin/airflow/env APIs or UI inst
ead. Can be used ONLY for the private cloud setup. Activate an existing envi
ronment for a given VC
  delete-dag       Delete an unassociated DAG from Airflow
  get-active-pyenv Deprecated, please use /admin/airflow/env APIs or UI in
stead. Can be used ONLY for the private cloud setup. Get currently active ai
rflow python env resource name
  reset-pyenv      Deprecated, please use /admin/airflow/env APIs or UI i
nstead. Can be used ONLY for the private cloud setup. Reset Airflow env to f
actory defaults
  submit           Run a DAG on CDE Airflow
```

cde backup command

```
Usage:
  cde backup [command]

Available Commands:
  create       Create a backup of jobs and resources
  list-archive List jobs and resources in an archive
  restore      Restore a backup of jobs and resources
```

cde credential command

```
Usage:
  cde credential [command]

Available Commands:
  create      Create a credential (experimental)
  delete      Delete a credential (experimental)
  describe    Describe a credential (experimental)
  list        List credentials (experimental)
  update      Update a credential (experimental)
```

cde job command

```
Usage:
  cde job [command]

Available Commands:
  clone       Clone an existing job or job run as a new job
  create      Create a job
  delete      Delete a job
  describe    Describe a job
  import      Import a job
  list        List jobs
  run         Run a job
  schedule    Operate CDE job schedules
  update      Update a job
```

cde profile command

```
Usage:
  cde profile [command]

Available Commands:
  show-active Shows the active profile of config file
```

cde repository command

```
Usage:
  cde repository [command]

Available Commands:
  create      Create a repository
  delete      Delete a repository
  describe    Describe repository
  download    Download a file from a repository
  list        List repositories
  sync        Syncs a repository with its remote
```

cde resource command

```
Usage:
  cde resource [command]
Available Commands:
  create           Create a resource
  delete           Delete a resource
  delete-file      Delete a file from a resource
  describe         Describe resource
  download         Download a file from a resource
  download-archive Download resource as archive
  list             List resources
  list-events      List resource events
  upload           Upload one or more files to a resource
  upload-archive   Upload and extract archive to resource
```

cde run command

```
Usage:
  cde run [command]

Available Commands:
  debug       Debug a run
  describe    Describe a run
  fg-status   Get the fine-grained statuses of a run
  kill        Kill a run
  list        List runs
  logs        Retrieve logs for a run
  ui          Open a run in the default browser
  wait        Wait for a run to end
```

cde session command

```
Usage:
  cde session [command]

Available Commands:
  create      Create a session
  delete      Delete a session
  describe    Describe a sessions
  interact    interact with a running session
  kill        Kill a session
  list        List sessions
  logs        Retrieve logs for a session
  statements  List session statements
```

cde spark command

```
Usage:
  cde spark [command]

Available Commands:
  submit      Run a jar/py file on CDE Spark
```

CDE CLI Spark flag reference

The Cloudera Data Engineering command Spark flag reference is shown below.

```
--application-file: application main file
--class: application main class
--arg: Spark argument
--conf: Spark configuration (format key=value) (can be repeated)
--min-executors: minimum number of executors
--max-executors: maximum number of executors
--initial-executors: initial number of executors
--executor-cores: number of cores per executor
--executor-memory: memory per executor
--driver-memory: memory for driver
--driver-cores: number of driver cores
--spark-name: Spark application name
--file: additional file additional file (can be repeated) (will be merged w
ith --files, if provided)
--files: additional files (comma-separated list) (will be merged with all --
file)
--jar: additional jar (can be repeated) (will be merged with --jars, if prov
ided)
--jars: additional jars (comma-separated list) (will be merged with all --
jar)
--py-file: additional Python file (can be repeated) (will be merged with --
py-file, if provided)
--py-files: additional Python files (comma-separated list) (will be merged
 with all --py-file)
--packages: additional dependencies as comma-separated list of Maven coordi
nates
--repositories: additional repositories/resolvers for retrieving the --pac
kages dependencies
--python-env-resource-name: Python environment resource name
--python-version: Python version ("python3" or "python2")
--log-level: log level for Spark containers (TRACE, DEBUG, INFO, WARN, ERR
OR, FATAL, OFF)
--enable-analysis: enables Spark analysis (see 'Analysis' UI tab for a job r
un)
```

CDE CLI Airflow flag reference

The Cloudera Data Engineering command Airflow flag reference is shown below.

```
cde airflow submit --help
Usage:
  cde airflow submit [flags]

Examples:
For a local DAG file 'my-airflow-job.py':
```

```
> cde airflow submit my-airflow-job.py

Flags:
      --airflow-file-mount-N-prefix string     mount directory prefix for
airflow file mount N (defaults to "/airflow-file-mount-N-resource-name")
      --airflow-file-mount-N-resource string   resource name for airflow fi
le mount N
      --config stringArray                     DEPRECATED - DAG configura
tion (format key=value) (can be repeated). Use --config-json option instead.
      --config-json string                     DAG configuration in JSON st
ring format
      --config-json-file string                DAG configuration file locati
on in JSON format
      --dag-file string                        DAG filename, path to the DAG
 within the resource
  -h, --help                                   help for submit
      --job-name string                        name of the generated job
```

CDE CLI list command syntax reference

You can include flags with the Cloudera Data Engineering command line interface (CLI) list command calls to filter the result set.

cde [credential|job|resource|run|...] list [--filter [fieldname[operator]argument]] [--filter [fieldname[operator]argument]] ...

A list command call can include multiple filter flags, where all filters must match for the entry to be returned. You have to enclose filters in quotes.

fieldname

is selected from the top-level fields of the returned entries. Filtering of fields nested within other fields is supported using MySQL 8 JSON path expressions.

operator

is one of: eq, noteq, lte, lt, gte, gt, in, notin, like, rlike. The in and notin operators work on an argument of comma-separated values. The like operator matches using SQL LIKE syntax, e.g. %test %. The rlike operator matches using the SQL REGEXP regular expression syntax.

argument

is the value, list, or expression to match with the operator. If the argument contains commas the filter has to be enclosed in a second set of quotes, for example: '"id[in]12,14,16"'.

Note:

Timestamps must be formatted as MySQL date time literals.

For example:

```
cde run list --filter 'spark.spec.file[rlike]jar'
```

Managing workload secrets with Cloudera Data Engineering Spark Jobs using the CLI

Cloudera Data Engineering provides a secure way to create and store workload secrets for Cloudera Data Engineering Spark Jobs. This is a more secure alternative to storing credentials in plain text embedded in your application or job configuration.

Creating a workload secret for Cloudera Data Engineering Spark Jobs using CLI

You can create workload secrets using interactive mode or using a JSON file. You can use the `--workload-cred-json-file` and the `--workload-cred-key` flags along with the `--type` flag supporting workload-credential.

Before you begin

Make sure that you have downloaded the CLI client. For more information, see Using the Cloudera Data Engineering command line interface.

For Interactive mode

- Specify --workload-cred-key when prompted for secret values multiple times. The values which are sensitive are read as a hidden password field interactively from the CLI.

```
./cde credential create --name <workload-credential-name> --type workloa
d-credential --workload-cred-key <workload-credential_key> --workload-cr
ed-key <workload-credential_key>
```

For example:

```
./cde credential create --name workload-cred-1 --type workload-credentia
l --workload-cred-key db-pass --workload-cred-key aws-secret

Enter Secret value for Workload Cred key "db-pass" :
Re-enter Secret value for Workload Cred key "db-pass" :
Enter Secret value for Workload Cred key "aws-secret" :
Re-enter Secret value for Workload Cred key "aws-secret" :
```

For JSON file

1. Create a JSON file with workload secret keys.

```
sample.json file

{
    "<workload-credential-key>": "<secret_value_of_key>",
    "<workload-credential-key>": "<secret_value_of_key>"
  }
```

For example:

```
sample.json file

{
    "aws-secret": "secret123",
    "db-pass": "dbpass123"
```

```
  }
```

2. Run the following command to create the workload secret:

```
./cde credential create  --name <workload-credential-name> --type worklo
ad-credential --workload-cred-json-file <workload-credential-json-file-n
ame>
```

For example:

```
./cde credential create  --name workload-cred-1 --type workload-credenti
al --workload-cred-json-file sample.json
```

Updating a workload secret for Cloudera Data Engineering Spark Jobs using CLI

You can update an existing secret to use it with the Cloudera Data Engineering Spark Jobs.

For Interactive mode

- Specify --workload-cred-key when prompted for secret values multiple times. The values which are sensitive are read as a hidden password field interactively from the CLI.

```
./cde credential update --name <workload-credential-name> --workload-cre
d-key <workload-credential_key> --workload-cred-key <workload-credential
_key>
```

For example:

```
./cde credential update --name workload-cred-1 --workload-cred-key db-pa
ss --workload-cred-key aws-secret --workload-cred-key api-token
Enter Secret value for Workload Cred key "dbPassword" :
Re-enter Secret value for Workload Cred key "dbPassword" :
Enter Secret value for Workload Cred key "aws-secret" :
Re-enter Secret value for Workload Cred key "aws-secret" :
```

For JSON file

1. Update the JSON file with workload secret keys.

For example:

```
sample.json file – file name

{
    "aws-secret": "secret123",
    "db-pass": "dbpass123"
  }
```

2. Run the following command to create the workload secret with updated parameters:

```
./cde credential update --name <workload-credential-name> --type workloa
d-credential --workload-cred-json-file <workload-credential-json-file-na
me>
```

For example:

```
./cde credential update --name workload-cred-2 --workload-cred-json-file
 sample.json
```

Linking a workload secret to the Cloudera Data Engineering Spark Job definitions using CLI

After you create a workload secret, you can link it to the Cloudera Data Engineering Spark Job definitions that you created using CLI.

```
./cde job create --name <workload-credential-name> --type <workload-credenti
al_type> --application-file <application-file-name> --workload-credential <w
orkload-credential-1> --workload-credential <workload-credential-2>
```

For example:

```
./cde job create --name test-workload-job --type spark --mount-1-resource te
st-workload --application-file test-workload-cred.py --workload-credential w
orkload-cred-1 --workload-credential workload-cred-2
```

Using a workload secret in Spark application code

To use the workload secret credentials, you can read the credentials that are mounted into the Spark drivers and executors as read-only files.

The workload secrets are mounted into the Spark drivers and executors in this path:

```
/etc/dex/secrets/<workload-credential-name>/<credential-key-1> /etc/dex/secr
ets/<workload-credential-name>/<credential-key-2>
```

Example workload credentials to use in the application code:

The workload credential is created with the command below.

```
./cde credential create --name workload-cred-1 --type workload-credential --
workload-cred-key db-pass --workload-cred-key aws-secret
```

The secrets can be read as local files from the paths below within the Spark drivers and executors:

```
/etc/dex/secrets/workload-cred-1/aws-secret
/etc/dex/secrets/workload-cred-1/db-pass
```

Example of a PySpark application code to read a secret:

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("Sample DB Connection") \
```

```
    .getOrCreate()

# read the password from the local file
dbPass=open("/etc/dex/secrets/workload-cred-1/db-pass").read()

# use the password in a jdbc connection
jdbcDF= spark.read \
         .jdbc("jdbc:postgresql:dbserver", "schema.tablename",
         properties={"user": "username", "password": dbPass})
```

Listing an existing workload secret to the Cloudera Data Engineering Spark Job

You can list an existing secret for Cloudera Data Engineering Spark Jobs using CLI.

```
./cde credential list  --filter 'type[eq]workload-credential'
```

Example output:

```
[
    {
      "name": "workload-cred-1",
      "type": "workload-credential",
      "description": "workload credential description",
      "created": "2022-12-18T07:26:41Z",
      "modified": "2022-12-18T07:26:41Z"
    }
  ]
```

Deleting a workload secret for Cloudera Data Engineering Spark Jobs using CLI

You can delete an existing secret for Cloudera Data Engineering Spark Jobs using CLI.

About this task

```
./cde credential delete --name <workload-credential-name>
```

For example:

```
./cde credential delete --name workload-cred-1
```

Cloudera Data Engineering 1.23.0

Managing Cloud Costs in Cloudera Data Engineering

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Data Engineering Spot Instances.....4

Cloudera Data Engineering Spot Instances

Cloudera Data Engineering supports spot instances to facilitate cloud cost savings for workloads that are not SLA-bound. If you have enabled spot instances for your Cloudera Data Engineering service, when you create a virtual cluster, you can specify whether drivers and executors run on spot instances or on-demand instances. The default setting uses on-demand instances for drivers, and spot instances for executors.

Note: Spot Instances are not currently supported for Cloudera Data Engineering on Microsoft Azure.

Spot instances are discounted instances that can be reclaimed at any time by the cloud provider. For fault-tolerant workloads such as Apache Spark, utilizing spot instances can provide significant cost savings. Because spot instances can disappear, impacting job performance, Cloudera recommends only using them for workloads without strict SLA requirements.

For development, testing, and other workloads that have more flexible requirements, Cloudera recommends using on-demand instances for drivers, and spot instances for executors. This reduces the impact of losing a spot instance, because executor failure recovery is more efficient than recovery from a driver failure.

To use spot instances, you must enable them on the Cloudera Data Engineering service. For instructions, see Enabling Cloudera Data Engineering. You cannot enable spot instances on an existing Cloudera Data Engineering service. To use spot instances, create a new service.

After you have enabled spot instances for the Cloudera Data Engineering service, you can configure individual virtual clusters to use on-demand instances only, spot instances only, or the default hybrid model (on-demand for drivers, spot for executors).

For more information on spot instances, see the cloud provider documentation:

Amazon AWS

Amazon EC2 Spot Instances

Cloudera Data Engineering 1.23.0

Data Engineering deployment architecture

Date published: 2020-07-30
Date modified: 2024-11-12

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Recommendations for scaling Cloudera Data Engineering deployments.....4

Apache Airflow scaling and tuning considerations.....5

Guidelines for Virtual Cluster upkeep.....6

Guidelines for database upkeep.....6

Recommendations for scaling Cloudera Data Engineering deployments

Your business might experience a sudden increase or drop in demand due to which your Cloudera Data Engineering deployment needs to autoscale. You can scale your Cloudera Data Engineering deployment by either adding new instances of a Cloudera Data Engineering service or Virtual Cluster, or by adding additional resources to the existing ones.

You can scale your Cloudera Data Engineering deployment:

- Vertically - More resources are provisioned within the same instance of a Cloudera Data Engineering service or Virtual Cluster.
- Horizontally - New instances of Cloudera Data Engineering service or Virtual Cluster are provisioned.

The key components of a typical Cloudera Data Engineering service deployment are:

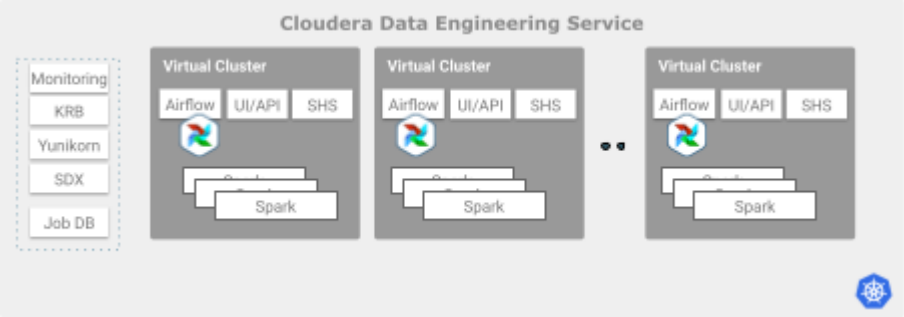

Virtual Clusters provide an isolated autoscaling compute capacity to run Spark and Airflow jobs. You can use Virtual Clusters to isolate individual teams or lines of business by using user-based Access Control Lists (ACLs).

Guidelines for scaling Virtual Clusters

- Each Virtual Cluster requires infrastructure capacity to run various services such as Airflow, API server, and Spark-History-Server (SHS).

  Recommendation: Do not scale horizontally beyond 50 Virtual Clusters within the same Cloudera Data Engineering service.
- Virtual Clusters can actively run hundreds of parallel jobs. In certain scenarios, it might be required to simultaneously submit multiple jobs as per the schedule or due to a burst in demand. In these scenarios, the API server enforces guardrails and limits the number of simultaneous Spark job submissions to 60. Once the jobs move from submission to running state, you can submit more jobs.

  Note: The Airflow jobs are not impacted by the submission limit of 60 simultaneous jobs.

  Recommendation: Distribute simultaneous submission of jobs over time or horizontally scale across multiple Virtual Clusters.

Job submission rate guardrails

When jobs are submitted to the Cloudera Data Engineering API server of a particular Virtual Cluster, it takes time and resources, known as preparing and starting states, before they begin running. This process is called the job submission intake process. To ensure proper resourcing and handling of incoming jobs, guardrails have been set up. By default, the guardrail, or limit is set to 60 simultaneous job submissions. Simultaneous incoming Spark job submissions that exceed 60 return a 429 error message to the client. The example error message is: Failed to submit. Too many requests.

Recommendation:

- Incorporate Cloudera Data Engineering Clients error handling for this error. The CDE CLI exit code is 77. The CDE CLI receives an HTTP 429 response code for the request to the Runtime API Server's REST API. For more information on exit codes, see Cloudera Data Engineering CLI exit codes.
- If needed, increase or decrease the guardrail limit during the Virtual Cluster creation by setting the "limiting.simultaneousJobSubmissions" configuration.

Related Information

Creating virtual clusters

Cloudera Data Engineering CLI exit codes

Apache Airflow scaling and tuning considerations

There are certain limitations related to the deployment architecture, and guidelines for scaling and tuning of the deployment, that you must consider while creating or running Airflow jobs (DAGs).

Cloudera Data Engineering deployment architecture

If you use Airflow to schedule and/or develop multi-step pipelines, you must consider the following deployment limitations to scale your environment.

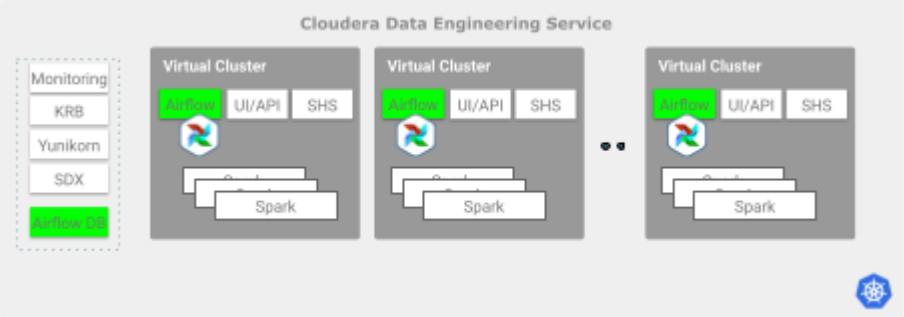

These guidelines describe when to scale horizontally, given total loaded DAGs and/or number of parallel (concurrent) tasks. These apply to all cloud providers.

Cloudera Data Engineering service guidelines

The number of Airflow jobs that can run at the same time is limited by the number of parallel tasks that are triggered by the associated DAGs. The number of tasks can reach up to 250-300 when running in parallel per Cloudera Data Engineering service.

Recommendation: Create more Cloudera Data Engineering services to increase Airflow tasks concurrency beyond the limits noted above.

Virtual Cluster guidelines

- Each Virtual Cluster has a maximum parallel task execution limit of 250. This applies to both Amazon Web Services (AWS) and Azure.

  Note: It is recommended that you do not exceed 300 parallel tasks across all Virtual Clusters.

- For AWS: The recommended maximum number of Airflow jobs loaded per virtual cluster is 1500. As the number of jobs increases, the job page loading time experiences latency along with Airflow job creation time.
- For Azure: The recommended maximum number of Airflow jobs loaded per virtual cluster is 300 to 500. Higher capacity can be achieved but will require manual configuration changes.

  Recommendation: Create more Virtual Clusters to load additional Airflow DAGs.

- The suggested number of Airflow jobs created in parallel is three or less. For jobs that are created in parallel, specify a DAG and/or a task_group for each Airflow operator.

  Recommendation: Distribute the creation of Airflow jobs so that you do not create more than three jobs at the same time.

Guidelines for Virtual Cluster upkeep

There are upkeep guidelines for Cloudera Data Engineering Spark History Server (SHS) that you'll need to consider.

Lifecycle configuration of Cloudera Data Engineering Spark event logs

The number of Spark event logs (spark.eventLog.enabled), that are produced by Spark jobs that run via Cloudera Data Engineering Virtual Cluster, grows indefinitely with each new Cloudera Data Engineering Spark run. These event logs are not automatically deleted and are stored on the object store under <CDP env storage location>/dex/<Service ID>/<VC ID>/eventlog/.

Some examples of the event log location can look like the following:

- For Amazon Web Services (AWS): s3a://dex-storage-bucket/datalake/logs/dex/cluster-2xvl4pfp/rdw8q2sh/eventlog/
- For Azure: abfs://logs@dexstorageaccount.dfs.core.windows.net/dex/cluster-4p54mk8j/22bnm99g/eventlog/

Because the number of event logs continuously increases, the time from when the Cloudera Data Engineering job finishes and when the Spark UI is available for this run on the Virtual Cluster UI may increase. The delay is most apparent in Virtual Clusters with 6,000 or more completed Spark job runs.

To avoid delays in event log availability after Cloudera Data Engineering job runs, you can configure an object store lifecycle policy so that event logs are deleted automatically on the object store. For more information about an Amazon S3 lifecycle policy, see Setting lifecycle configuration on a bucket linked below. For more information about Azure lifecycle management policies, see Configure a lifecycle management policy linked below.

Note: The lifecycle policy rule prefix does not include a bucket or storage account name.

Related Information

Setting lifecycle configuration on a bucket

Configure a lifecycle management policy

Guidelines for database upkeep

There are database upkeep guidelines for Cloudera Data Engineering that you'll need to consider.

Note: Despite the end of service for mySQL v8.0.28 in June 2023, you can continue using Cloudera Data Engineering with mySQL v8.0.28. Although mySQL will continue to work with your databases after end of service, troubleshooting support will not be available. Therefore, it is recommended that you upgrade to a supported version of mySQL.

Upgrade mySQL for AWS

Upgrade mySQL for AWS to ensure you are using a supported version to work with Cloudera Data Engineering.

Note: If you are using an in-place upgrade, your mySQL version will also be updated.

1. In the AWS console, navigate to RDS > cluster-id > Modify.

2. Select the version that you want to upgrade from the Database engine version drop-down list.
3. Click Continue.
4. Click Apply Immediately to apply changes.

Increasing database size on Azure

Azure databases can fill up with logs and entries with every job that runs in Cloudera Data Engineering Virtual Clusters. When the database fills up, increase the size of your database.

1. Ensure that there are no jobs running.
2. Go to the Azure Portal.
3. Locate the Azure SQL database named <cluster id>, for example, cluster-2cmmds8q.
4. Navigate to the Pricing Tier section.
5. Move the slider to the desired value.

   Note: The slider cannot be reduced.

6. Click OK and wait for the scaling to complete.
7. Resume running your jobs.

Increasing database size on Amazon Web Services

Amazon Web Services (AWS) databases can fill up with logs and entries with every job that runs in Cloudera Data Engineering Virtual Clusters. When the database fills up, increase the size of your database.

1. Go to the AWS Console.
2. Navigate to the RDS Service page.
3. Click Databases and use the filter to find your cluster id, for instance, cluster-w8d65nxp.
4. Select the target database.
5. In the Database homepage in the top right-hand corner, click Modify.
6. Scroll down to the database size settings and set the Allocated Storage Property to the desired value.
7. Click Continue.
8. Set the desired timeframe for maintenance.
9. Click Continue and wait for the database status to become available.
10. Resume running your jobs.

Cloudera Data Engineering 1.23.0

Enabling and disabling Cloudera Data Engineering

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

AWS Graviton instances in Cloudera Data Engineering..... 4

Enabling a Cloudera Data Engineering service..... 4

Enabling a fully private network for a Cloudera Data Engineering service for Azure (Tech Preview).....7

Azure private DNS zones in a Cloudera Data Engineering service (Technical Preview).....8

Enabling a semi-private network for a Cloudera Data Engineering service with AWS (Tech Preview)..... 12

Managing a Cloudera Data Engineering Service..... 12

Cloudera Data Engineering cluster types..... 13

Removing a Cloudera Data Engineering service.....15

Limiting Incoming Endpoint Traffic for Cloudera Data Engineering Services For AWS.....16

AWS Graviton instances in Cloudera Data Engineering

AWS Graviton is a general purpose, ARM-based processor family. AWS Graviton delivers currently the best price performance for cloud workloads running in AWS Elastic Compute Cloud (EC2). With AWS Graviton, you can optimize costs and achieve better performance. From the AWS Graviton family, Cloudera Data Engineering supports AWS Graviton 3.

Prerequisites for using AWS Graviton in Cloudera Data Engineering

For information on the prerequisites, such as the Data Lake and the Spark supported versions, see Compatibility for Cloudera Data Engineering and Runtime components.

If your Data Lake version is not supported by Graviton, see Upgrading a Data Lake.

Considerations for using AWS Graviton in your cluster

By default, AWS Graviton is available to all customers for Cloudera Data Engineering service compute nodes if the component version requirements are met.

To use AWS Graviton, when enabling a Cloudera Data Engineering service, select from the available Graviton instances listed under the Workload type drop-down list.

With AWS Graviton, the use of SSD instances is supported, but using spot instances is not supported.

If you select the AWS Graviton workload when creating the service:

- All Virtual Clusters will run AWS Graviton. Running a subset of Virtual Clusters or jobs on non-Graviton instances is not supported.
- To check the list of Spark versions supported by Graviton, see Compatibility for Cloudera Data Engineering and Runtime components.

Enabling a Cloudera Data Engineering service

Before you can use the Cloudera Data Engineering service, you must enable it on each environment that you want to use Cloudera Data Engineering on.

Before you begin

Make sure that you have a working environment for which you want to enable the Cloudera Data Engineering service. For more information about environments, see Environments.

Note: Cloudera Data Engineering on Microsoft Azure does not currently support SSD or Spot instances.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu. The Administration page displays.
3. In the Services column, click the plus icon at the top to enable Cloudera Data Engineering for an environment.
4. Type the name of the service that you want to enable Cloudera Data Engineering for.
5. Select the Environment from the drop-down menu.

6. Select the Workload Type.

   The workload type corresponds to the instance size that will be deployed to run your submitted Spark jobs. When you select a type, the corresponding cloud provider instance size is displayed in the Summary section to the right.

   a) If you want to use SSD storage, check the box labeled Use SSD instances. In this configuration, SSD storage is used for the workload filesystem, such as the Spark local directory. If your workload requires more space than is available in the instance storage, select a larger instance type with sufficient local storage or select an instance type without SSD, and then configure the EBS volume size.

7. Set the Auto-Scale Range under the Capacity & Costs section:

   The range you set here creates an auto scaling group with the specified minimum and maximum number of instances that can be used. The Cloudera Data Engineering service launches and shuts down instances as needed within this range. The instance size is determined by the Workload Type you selected.

   You can use Reserved Instances (RIs) in your account to save cost. Reserved Instances are not physical instances, but a billing discount that is applied to the use of On-Demand Instances in your account. To benefit from the billing discount, the On-Demand Instances in your account must match certain attributes. For more information, see the AWS documentation on Reserved Instances. If you want to use RIs, you must also specify an instance family for them.

   For Azure: Set the On-demand Instances range. This option displays for Core (Tier 1) cluster types during service creation. After the service and cluster is created, you can edit the service and set the All purpose On-demand Instances range under the Capacity & Costs section.

   For AWS: After the service and clusters are created, you can set the All purpose On-demand Instances range and the All purpose Spot Instances range under the Capacity & Costs section. This option displays for the All-purpose (Tier 2) cluster types.

   Important: For both Microsoft Azure and Amazon Web Services, to avoid being charged for All purpose compute nodes, even if you do not create an All purpose Virtual Cluster:

   Navigate to the Service Details page > Configurations tab > Capacity & Costs > Autoscale Range and set the minimum value of All purpose On-demand Instances to 0.

   For more information, see Cloudera Data Engineering cluster types.

8. If you want to use spot instances, check the box labeled Use Spot instances and select a range of spot instances to request. This creates another auto scaling group of spot instances. Spot instances are requested with similar CPU and memory profiles as the instances selected for the Workload Type. For more information, see Cloudera Data Engineering Spot Instances.

   Note: Duplicate auto-scaling groups of the same size are created for supporting tiered pricing. These can be edited at a later time if needed.

9. Optional: Enable a private network (preview feature). This feature ensures that the Cloudera Data Engineering service is deployed with a secure network setup based on the cloud provider such as Microsoft Azure or Amazon Web Services (AWS). For more information see the links below on how to configure this feature as there are prerequisites needed before you see the Enable Private Network option in the Cloudera Data Engineering user interface under Network & Storage.

10. If you create the service in an AWS environment using a non-transparent proxy, you find a Proxy CIDR Source Ranges field. You are only required to enter the proxy CIDR ranges for the proxy instances if you registered your proxies using host names, as the Cloudera Control Plane has no way to resolve those to IPs. If your proxy instances were registered using IP addresses, you can leave this field blank.
    For example, if you have a load balanced proxy with static IPs running on 10.80.199.105 and 10.80.200.45, add "10.80.199.105/32" and "10.80.200.45/32". If your proxy instances are dynamic (behind a load balancer or virtual IP) then you would enter a wider range, e.g.: "10.80.200.0/24".

    Note: Currently non-transparent support is only available on Amazon Web Services.

11. If you want to create a load balancing endpoint in a public subnet, check the box labeled Enable Public Loadbalancer. If you leave this unchecked, the load balancing endpoint will be created in a private subnet, and you will need to configure access manually in your cloud account.

    Important: When you enable the public loadbalancer, you need at least one public subnet configured in the environment.

12. Specify a comma-separated list of CIDRs in API server Authorized IP Ranges that can access the Kubernetes master API server.

    You may specify a comma-separated list of CIDRs that can access the Kubernetes master API server.

    Attention: Leaving this field empty renders the Kubernetes API server open to all traffic.

    Make sure that the provided IP addresses do not overlap with the following ranges:

    - 0.0.0.0 - 0.255.255.255
    - 10.0.0.0 - 10.255.255.255
    - 100.64.0.0 - 100.127.255.255
    - 127.0.0.0 - 127.255.255.255
    - 169.254.0.0 - 169.254.255.255s
    - 172.16.0.0 - 172.31.255.255
    - 192.0.0.0 - 192.0.0.255
    - 192.0.2.0 - 192.0.2.255
    - 192.88.99.0 - 192.88.99.255
    - 192.168.0.0 - 192.168.255.255
    - 198.18.0.0 - 198.19.255.255
    - 198.51.100.0 - 198.51.100.255
    - 203.0.113.0 - 203.0.113.255
    - 224.0.0.0 - 239.255.255.255
    - 240.0.0.0 - 255.255.255.254
    - 255.255.255.255

13. Specify a comma-separated list of client IP ranges in Load Balancer Source Ranges that should be allowed to access the load balancer.
14. Specify which subnets to use for the Kubernetes worker nodes. Select from available Subnets in the drop-down list.
15. Specify which subnets to use for the Load balancer. Select from available Subnets in the drop-down list.
16. Optional: Check the box labeled Enable Observability Analytics if you want diagnostic information about jobs and query execution sent to Cloudera Observability. This helps optimize troubleshooting.
17. Check the box labeled Enable Workload Analytics to automatically send diagnostic information from job execution to Cloudera Workload Manager.
18. Optionally add Tags as needed. Tags are applied to the cloud provider resources associated with the Cloudera Data Engineering service (including virtual clusters created in that service). For more information about tags, see the cloud provider documentation:

    Amazon AWS

    Tagging AWS resources

    Microsoft Azure

    Use tags to organize your Azure resources and management hierarchy

    Note: The following tags are added automatically by Cloudera Data Engineering along with the custom Tags: "cde-cluster-id", "cde-provisioner-id", "cde-owner-email", "Cloudera-Resource-Name", "owner". After Cloudera Data Engineering 1.20.3, the "owner" tag is no longer added. If you ever rely on the "owner" tag, please use "cde-owner-email" instead.

19. Default Virtual Cluster selection is enabled by default to create a default virtual cluster after enabling a Cloudera Data Engineering service. This will help you get a jump start to create your jobs easily, without having to wait to create a Cloudera Data Engineering virtual cluster as mentioned in Creating virtual clusters, making the onboarding smoother. You can turn this toggle off if you do not wish to use a default virtual cluster.
20. Click Enable.

Results

The Cloudera Data Engineering Administration page displays the status of the environment initialization. You can view logs for the environment by clicking on the environment vertical ellipsis menu, and then clicking View Logs.

Related Information

Cluster Connectivity Manager

Enabling Cluster Connectivity Manager in the Management Console

Enabling a fully private network for a Cloudera Data Engineering service for Azure (Tech Preview)

Learn how to enable a fully private network setup for a Cloudera Data Engineering service for Azure services in Cloudera. Additionally, you can learn how to add User Defined Routing (UDR) in the UI or CLI. The UDR helps from exposing public IP addresses in your service.

This feature ensures that all Azure services used by Cloudera Data Engineering are provisioned as private (private Azure Kubernetes Service (AKS), MySQL, and Storage Accounts). The Azure cluster is deployed as a fully private network cluster when you enable a Cloudera Data Engineering service and enables VNet access through private endpoints and private links. Lastly, Cloudera Data Engineering on Microsoft Azure does not currently support SSD or Spot instances.

For Cloudera Data Engineering UI

Before you begin

- Ensure that you have created and enabled a Cloudera Data Engineering service. Additionally, the Cloudera must communicate with the Cloudera Data Engineering service on a private network in order to manage the Cloudera Data Engineering service lifecycle. This communication occurs using the Cluster Connectivity Manager v2; therefore, to enable this feature, the Cloudera environment must be enabled with the CCMv2. Once the CCMv2 is enabled at the Cloudera environment level, the Enable Private Network option displays in the Cloudera Data Engineering user interface when you enable a service. For more information on how to enable a Cloudera Data Engineering service and set up CCMv2, refer to the links below.

  Note: To enable UDR, you must enable a private network flag and you must provide a subnet.

UI steps for enabling a private network and enabling UDR

1. While enabling a Cloudera Data Engineering service for an Azure environment, select Enable Private Network. Optionally, once you've enabled a private network on Microsoft Azure, you can select the User Defined Routing checkbox. Use this to avoid exposing public IP addresses in your service by using a user defined routing (UDR) table. After, you'll need to specify a Subnet.
2. Click Enable.

For CDP CLI

You can enable the user defined routing (UDR) with the CDP CLI using the –network-outbound-type CLI switch with a value of “UDR”. See the example command:

```
./clients/cdpcli/cdp.sh de enable-service --name "test-service-cdpcli" -
-env "dex-priv-env" --instance-type "Standard_D8s_v4" --minimum-instance
```

```
s 0 --maximum-instances 10 --enable-private-network --subnets dex-dev.in
ternal.19.westus2 --network-outbound-type UDR
```

Related Information

Enabling a Cloudera Data Engineering service

Cluster Connectivity Manager

Enabling Cluster Connectivity Manager in the Management Console

Azure private DNS zones in a Cloudera Data Engineering service (Technical Preview)

With the Azure private DNS zones feature, you can optimize costs by leveraging your existing Azure private DNS zone resources. When creating the Cloudera Data Engineering service, you can use Azure private DNS zones for the Azure Kubernetes Service (AKS), the Storage Account File Share, and for the database.

You can create private DNS zones in the same subscription where there is the Cloudera subscription, or you can create private DNS zones in a different subscription. For this reason, you must specify the full Azure resource ID of the Azure private DNS zones for the database and for the Storage Account when adding them to your Cloudera Data Engineering service. You can obtain the full resource ID of a private DNS zone using the JSON view option in the Azure Portal in Private DNS Zones.

Important: Before configuring the Azure private DNS zones, you must create them in the Azure Portal (one for the AKS, one for the database, and one for the Storage Account File Share) and link the private DNS zones to the Azure virtual networks.

Configuring an existing private DNS Zone for an AKS

At the Cloudera-level, when you provision the environment, you can configure an existing Azure Private DNS zone for an Azure Kubernetes Service (AKS). Alternatively, if you do not have an existing private DNS zone configured, the system can create a new private DNS zone for the AKS.

About this task

Important: To configure an existing Azure Private DNS zone for an AKS, you must create the environment using the CDP CLI.

If all the following statements are true, the system uses the private DNS zone that you specify in the aksPrivateDn sZoneId parameter while provisioning the environment for the AKS:

- The private network is enabled (the privateNetwork.enabled parameter is set to true)
- While provisioning the environment using CLI, the aksPrivateDnsZoneId parameter is configured
- The LIFTIE_AKS_DISABLE_PRIVATE_DNS_ZONE parameter is set to false (default)

If all the following statements are true, the system creates and enables a new private DNS zone for the AKS:

- The private network is enabled (the privateNetwork.enabled parameter is set to true)
- The aksPrivateDnsZoneId parameter is not set
- The LIFTIE_AKS_DISABLE_PRIVATE_DNS_ZONE parameter is set to false (default)

Before you begin

To configure an existing Azure Private DNS zone for an AKS, you need the full resource ID of the private DNS zone. To obtain the full resource ID, use the JSON view option in Azure Private DNS Zones . Example for the full resource ID of a private DNS zone for an AKS:

```
/subscriptions/[***SUBSCRIPTIONID***]/resourceGroups/[***RESOURCEGROUP***]/p
roviders/Microsoft.Network/privateDnsZones/privatelink.[***REGION***].azmk8s
.io
```

For information on how to create the environment for Cloudera Data Engineering, see Register an Azure environment from CDP CLI and Obtain CLI commands for registering an environment.

Note: Perform this procedure in the CDP CLI while creating the environment.

Procedure

1. Open the CDP CLI.
2. To configure an existing private DNS zone for AKS, you must create an Azure environment using the CDP CLI. While creating the environment, specify the resource ID in aksPrivateDnsZoneId in the following CLI option:

```
--existing-network-params (object)
          Request object for creating an Azure environment using existing
 VNet
          and subnets.

          networkId -> (string)
             The id of the Azure VNet.

          resourceGroupName -> (string)
             The name of the resource group associated with the VNet.
          subnetIds -> (array)
             One or more subnet ids within the VNet.

          aksPrivateDnsZoneId -> (string)
             The full Azure resource ID of an existing Private DNS zone
used
             for the AKS.

       Shorthand Syntax:

          networkId=string,resourceGroupName=string,subnetIds=string,aksPr
ivateDnsZoneId=string

       JSON Syntax:

          {
            "networkId": "string",
            "resourceGroupName": "string",
            "subnetIds": ["string", ...],
            "aksPrivateDnsZoneId": "string"
          }
```

Note: You can specify the aksPrivateDnsZoneId parameter only through the CDP CLI.

Results

You can use the AKS configured in the provided private DNS zone in a Cloudera Data Engineering service.

Configuring an existing private DNS zone for a database

You can configure an existing Azure private DNS zone for your MySQL Flexi Server-based database to use in your Cloudera Data Engineering service. Alternatively, if you do not have an existing private DNS zone configured, the system can create a new private DNS zone for the database.

About this task

If all the following statements are true, the system uses the private DNS zone that you specify in the azure.database.p rivateDNSZoneId parameter for the database:

- The privateNetwork.enabled parameter is set to true or false
- The azure.database.privateDNSZoneId parameter is set

If all the following statements are true, the system creates and enables a new private DNS zone for the database:

- The privateNetwork.enabled parameter is set to true or false
- The azure.database.privateDNSZoneId parameter is not set

For the database, you can configure a private DNS zone regardless of whether the privateNetwork.enabled parameter is set to true or false. This is because the MySQL Flexi server implementation requires a private DNS zone.

Before you begin

For information on how to configure an existing private DNS zone for a database using the API, see Configuring an existing private DNS zone for a database and a Storage Account File Share using the API.

To configure an existing private DNS zone for a database, you need the full resource ID of the private DNS zone. To obtain the full resource ID, use the JSON view option in Azure Private DNS Zones . Example for the full resource ID of a private DNS zone for a database:

```
/subscriptions/[***SUBSCRIPTIONID***]/resourceGroups/[***RESOURCEGROUP***]/p
roviders/Microsoft.Network/privateDnsZones/privatelink.mysql.database.azure.
com
```

Procedure

To configure an existing private DNS zone for a database, specify the full resource ID of the private DNS zone.

To configure the Azure private DNS zone ID for the database, use the enable-cluster CDP command with the --azure-database-private-dns-zone-id flag.

```
cdp de enable-service
 --name "[***SERVICE NAME***]"
 --env "[***ENVIRONMENT NAME***]"
 --instance-type Standard_D8s_v4
 --minimum-instances 0
 --maximum-instances 50
 --enable-private-network
 --azure-database-private-dns-zone-id "[***AZURE DATABASE PRIVATE DNS ZONE
 RESOURCE ID***]"
 --azure-fileshare-private-dns-zone-id "[***AZURE DATABASE FILE SHARE DNS
ZONE RESOURCE ID***]"
```

Configuring an existing private DNS zone for a Storage Account File Share

You can configure an existing Azure private DNS zone for your Storage Account File Share to use in your Cloudera Data Engineering service. Alternatively, if you do not have an existing private DNS zone configured, the system can create a new private DNS zone for the Storage Account File Share.

About this task

If all the following statements are true, the system uses the private DNS zone that you specify in the azure.fileshare.privateDNSZoneId parameter for the private Storage Account:

- The private network is enabled (the privateNetwork.enabled parameter is set to true)
- The DE_AZURE_PRIVATE_STORAGE parameter is set to true on Cloudera-level
- The azure.fileshare.privateDNSZoneId parameter is set
- The custom Storage Account (also known as Bring Your Own Storage Account) is not used

If all the following statements are true, the system creates and enables a new private DNS zone for the private Storage Account:

- The private network is enabled (the privateNetwork.enabled parameter is set to true)
- The DE_AZURE_PRIVATE_STORAGE parameter is set to true on Cloudera-level

- The azure.fileshare.privateDNSZoneId parameter is not set
- The custom Storage Account (also known as Bring Your Own Storage Account) is not used

Note: You cannot use the private DNS zone for File Share together with a custom Storage Account that is also known as Bring Your Own (BYO) Storage Account. This means that you cannot use the Override Azure File Storage Server UI option with the private DNS zone for Storage Account File Share. The reason for this is that the BYO private custom Storage Account integrates with a private DNS zone from Azure, but Cloudera Data Engineering cannot accept an additional private DNS zone to manage the Storage Account File Share.

Before you begin

For information on how to configure an existing private DNS zone for a Storage Account File Share using the API, see Configuring an existing private DNS zone for a database and a Storage Account File Share using the API.

To configure an existing private DNS zone for a Storage Account File Share, make sure that the following prerequisites are fulfilled:

- The DE_AZURE_PRIVATE_STORAGE parameter is set to true on the Cloudera-side.
- The privateNetwork.enabled parameter is set to true.

To configure an existing private DNS zone for a Storage Account File Share, you need the full resource ID of the private DNS zone. To obtain the full resource ID, use the JSON view option in Azure Private DNS Zones . Example for the full resource ID of a private DNS zone for Storage Account File Share:

```
/subscriptions/[***SUBSCRIPTIONID***]/resourceGroups/[***RESOURCEGROUP***]/p
roviders/Microsoft.Network/privateDnsZones/privatelink.file.core.windows.net
```

Procedure

To configure an existing private DNS zone for a Storage Account File Share, specify the full resource ID of the private DNS zone.

To configure the Azure private DNS zone ID for the Storage Account File Share, use the enable-cluster CDP command with the --azure-fileshare-private-dns-zone-id flag.

```
cdp de enable-service
 --name "[***SERVICE NAME***]"
 --env "[***ENVIRONMENT NAME***]"
 --instance-type Standard_D8s_v4
 --minimum-instances 0
 --maximum-instances 50
 --enable-private-network
 --azure-database-private-dns-zone-id "[***AZURE DATABASE PRIVATE DNS ZONE
 RESOURCE ID***]"
 --azure-fileshare-private-dns-zone-id "[***AZURE DATABASE FILE SHARE DNS
ZONE RESOURCE ID***]"
```

Enabling cross-subscription communication for private DNS zones

If the private DNS zones for the Storage Account File Share and the database are deployed in an Azure subscription that is outside of the Cloudera subscription, to enable the cross-subscription communication, you must configure the role assignment for Cloudera.

About this task

In the subscription of the private DNS zone for the Storage Account Files Share and for the database, you must assign a contributor role to the Azure application that is used for Cloudera.

Procedure

1. In Azure, navigate to the private DNS zone subscription of the database and the Storage Account File Share.
2. Using the Add role assignment option, assign a contributor role to the Azure application that is used for Cloudera.
   For more information, see the Microsoft Azure documentation on assigning Azure roles using the Azure portal.

Enabling a semi-private network for a Cloudera Data Engineering service with AWS (Tech Preview)

Learn how to enable a semi-private network setup for a Cloudera Data Engineering service with Amazon Web Services (AWS) services in Cloudera. When you enable a Cloudera Data Engineering service with this feature, the Amazon Kubernetes Service (EKS) cluster is deployed as a private cluster but some services used by Cloudera Data Engineering such as MySQL and S3 are not provisioned as private.

Note: You need to contact Cloudera to have this feature enabled.

For Cloudera Data Engineering UI

Before you begin

- Ensure that you have created and enabled a Cloudera Data Engineering service. Additionally, Cloudera must communicate with the Cloudera Data Engineering service on a private network in order to manage the Cloudera Data Engineering service lifecycle. This communication occurs using the Cluster Connectivity Manager v2; therefore, to enable this feature, the Cloudera environment must be enabled with the CCMv2. Once the CCMv2 is enabled at the Cloudera environment level, the Enable Private Network option displays in the Cloudera Data Engineering user interface when you enable a service. For more information on how to enable a Cloudera Data Engineering service and set up CCMv2, refer to the links below.

UI steps for enabling a private network

1. While enabling a Cloudera Data Engineering service for an AWS environment, under Network & Storage, select Enable Private Network.
2. Click Enable.

For CDP CLI

You can enable a private network with the CDP CLI with the following commands:

```
cdp de enable-service --name dsp-private-eks-ntp-try1 --env dsp-aws-ntp-
priv --instance-type m5.2xlarge --minimum-instances 0 --maximum-instance
s 4 --initial-instances 0 --root-volume-size 50 --no-skip-validation --e
nable-private-network
```

Related Information

Enabling a Cloudera Data Engineering service
Cluster Connectivity Manager
Enabling Cluster Connectivity Manager in the Management Console

Managing a Cloudera Data Engineering Service

You can view configuration, metrics, and logs of existing Cloudera Data Engineering services. You can use the Edit option to make the configuration changes dynamically.

Note: The active node count has been removed from the following UI pages:

- Administrator Overview
- Administrator Overview Service Details

For real-time node count information, check your Cloud Service Provider's website. For more information, see Checking the node count on your Cloud Service Provider's website.

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu. The Administration page displays.
3. In the Services column, click the Service Details icon for the Cloudera Data Engineering service you want to manage.
4. On the Configuration tab, you can view details about the service, including the service name and Cloudera environment.
5. Optional: Click the Edit option to make the configuration changes dynamically which may take a few minutes to update.

You can switch between the following tabs to view additional information:

Configuration

The Configuration tab lists details about the service name, Cloudera environment, and the CPU and memory capacity. You can modify the CPU and memory capacity dynamically.

Important: For both Microsoft Azure and Amazon Web Services, to avoid being charged for All purpose compute nodes, even if you do not create an All purpose Virtual Cluster:

At Capacity & Costs > Autoscale Range, set the minimum value of All purpose On-demand Instances to 0.

For more information, see Cloudera Data Engineering cluster types.

Charts

The Charts tab displays the charts related to CPU Requests, Memory Requests, Jobs, and Nodes.

Logs

The Logs tab displays the latest log entries for the Cloudera Data Engineering service.

Access

The Access tab displays the option to add Amazon Web Services (AWS) user's Amazon Resource Names (ARNs) for the Cloudera Data Engineering Service. The Access tab also displays the users you have added to the Cloudera Data Engineering Service which will provide them with the following:

- Access to all pods
- Access to all secrets including TGTs
- Bypass Istio security
- Access to tgtgen host keytab with ability to create, delete, and modify users in FreeIPA

Diagnostics

The Diagnostics tab provides option to generate and download diagnostics bundle.

Cloudera Data Engineering cluster types

Learn more about the available cluster types that you can use in your Cloudera Data Engineering service.

On the Cloudera Data Engineering UI, DE and Platform administrators can create new Virtual Clusters and select the required cluster type at Administration Create a Virtual Cluster .

Cloudera Data Engineering offers these cluster types:

- Core
- All purpose

In the same Cloudera Data Engineering Service, you can run some workloads on Core and some workloads on All purpose clusters.

Core cluster functionalities:

- Autoscaling cluster
- Spot instances
- Cloudera Shared Data Experience
- Open Lakehouse with Iceberg
- Job lifecycle management
- Centralized monitoring
- Workflow orchestration (Airflow)

All Purpose cluster functionalities (on top of functionalities available with Core clusters as well):

- Interactive sessions
- External IDE connectivity
- Spark streaming
- JDBC connector (Coming soon)

At Administration Enable a Service Capacity & Costs Autoscale Range , you can set the autoscale range for the On-demand Instances, that is for the Core clusters. The range you set on this page for the On-demand Instances is used also for the All purpose Instances, that is for the All purpose clusters.

At Administration Service Details , you can modify the autoscale value range of:

- On-demand Instances
- All purpose On-demand Instances
- Spot Instances
- All purpose Spot Instances

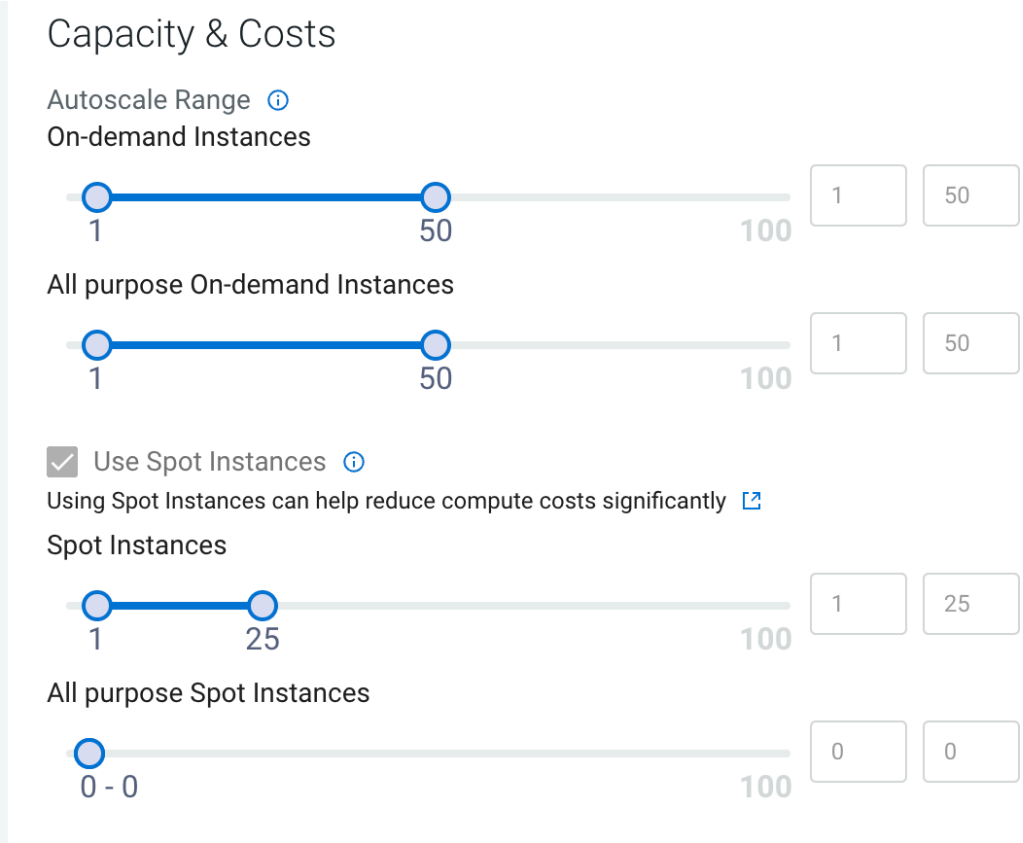

You can use Spot Instances and All purpose Spot Instances to save compute costs.

Important: With All purpose On-demand Instances, consider the following:

- Since version 1.20.3, Cloudera Data Engineering offers price tiering to provide better cost-control for the consumed features. The All purpose On-demand Instances are charged at a higher rate than the On-demand Instances (Core instances). For more information, see the pricing page.
- To avoid being charged for All purpose compute nodes, even if you do not create an All purpose Virtual Cluster, set the minimum value of All purpose On-demand Instances to 0.
- If you set the autoscale range minimum value of All purpose On-demand Instances to 0, if you create a Session, the Session can take several minutes to start.
- If you set the autoscale range minimum value of All purpose On-demand Instances to 1 or higher, you are charged for All purpose compute nodes even if you do not create an All purpose cluster.

Removing a Cloudera Data Engineering service

Disabling an existing Cloudera Data Engineering service stops all jobs, and deletes all associated virtual clusters and virtual cluster metadata. Do not do this unless you are certain that you no longer need any of these. Disabling Cloudera Data Engineering does not delete your Cloudera data. If enabling the Cloudera Data Engineering service on an environment for the first time fails, you must disable the service before you can try again. In this scenario, there are no clusters or jobs, and you can safely perform this procedure.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can remove a Cloudera Data Engineering service by clicking Administration on the left-hand menu, then proceed to step 2 listed below. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. In the Cloudera console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, click the menu icon for the environment for which you want to disable the Cloudera Data Engineering service, and then click Disable Cloudera Data Engineering.

   Warning: Disabling an existing Cloudera Data Engineering service stops all jobs, deletes all associated virtual clusters and virtual cluster metadata. Do not do this unless you are certain that you no longer need any of these. Additionally, if you're prompted to perform a Force Disable, in the event that a Disable is not successful, you must perform a manual cleanup of cloud infrastructures such as Security Group, EBS Volume, and S3 Bucket. A manual cleanup is not required for a standard Disable.
3. Confirm that you want to disable Cloudera Data Engineering by typing the environment name and then clicking Disable.

Results

The Cloudera Data Engineering Administration page displays the status of the environment that is being disabled.

What to do next

If you disabled Cloudera Data Engineering as a result of a failure to enable Cloudera Data Engineering for the first time on an environment, resolve any reported issues, and then try again.

Limiting Incoming Endpoint Traffic for Cloudera Data Engineering Services For AWS

You can limit incoming endpoint traffic for a Cloudera Data Engineering service.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. Note the Cloudera Data Engineering service ID, which you can obtain from the URL in the Cloudera Data Engineering Management Console when the service is highlighted:

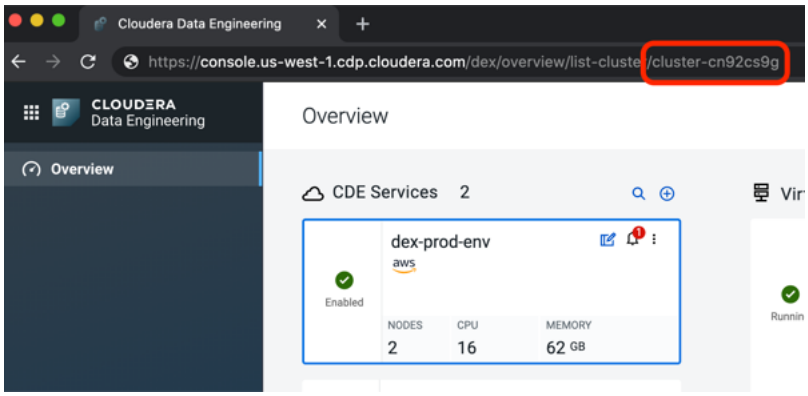

2. Go to the AWS console for the account where the Cloudera Data Engineering service is enabled.
3. Navigate to EC2 -> Load Balancers in the AWS console and enter the following filter:

```
 tag:cde-cluster-id : <id_from_step_1>
, e.g. tag:cde-cluster-id : cluster-cn92cs9g
```

4. Select the Load Balancer instance and then under the Description tab in the Detail window, click on the link to the Source Security Group.
5. In the subsequent view, select the correct security group ID.
6. In the subsequent window, click Edit inbound rules.
7. Modify the "0.0.0.0/0" CIDR ranges for the HTTPS rule to your desired CIDR ranges.

   Add additional ranges and rules as required but note that HTTPS traffic must be enabled for each range. The HTTP (port 80) and ICMP rules can be removed.

Cloudera Data Engineering 1.23.0

Managing virtual clusters

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating virtual clusters 4

Viewing and managing virtual cluster details 6

Applying user and group access for virtual clusters 7
Editing user and group access for virtual clusters 9
Cloudera Data Engineering Virtual Cluster Access Controls Troubleshooting 10
Managing Virtual Cluster-level Spark configurations (Technical Preview) 10

Editing virtual clusters 12

Locking and unlocking a Cloudera Data Engineering Service 15

Deleting virtual clusters 15

Accessing non-public Cloudera Data Engineering virtual clusters via SOCKS proxy 16

Creating virtual clusters

In Cloudera Data Engineering, a virtual cluster is an individual auto-scaling cluster with defined CPU and memory ranges. Jobs are associated with virtual clusters, and virtual clusters are associated with an environment. You can create as many virtual clusters as you need. See Recommendations for scaling Cloudera Data Engineering ceployments linked below.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly Cloudera Data Engineering version prior to 1.17, for example, the Overview page has been replaced with the Home page. The new Home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Note: Virtual Cluster installation fails when Ozone S3 gateway proxy is enabled. Ozone S3 gateway proxy gets enabled when more than one Ozone S3 Gateway is configured in the Cloudera Private Cloud Base cluster.

As a workaround, add the `127.0.0.1 s3proxy-<environment-name>.<private-cloud-control-plane-name>-services.svc.cluster.local` entry in the /etc/hosts file of all nodes in the Cloudera Private Cloud Base cluster where the Ozone S3 gateway is installed. For example, if the private cloud environment name is `cdp-env-1` and private cloud control plane name is `cdp`, then add the following entry in the `/etc/hosts` file:

```
127.0.0.1 s3proxy-cdp-env-1.cdp-services.svc.cluster.local
```

To create a virtual cluster, you must have an environment with Cloudera Data Engineering enabled.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu, select the environment you want to create a virtual cluster in.

3. In the Virtual Clusters column, click ⊕ at the top right to create a new virtual cluster.

   If the environment has no virtual clusters associated with it, the page displays a Create DE Cluster button that launches the same wizard.

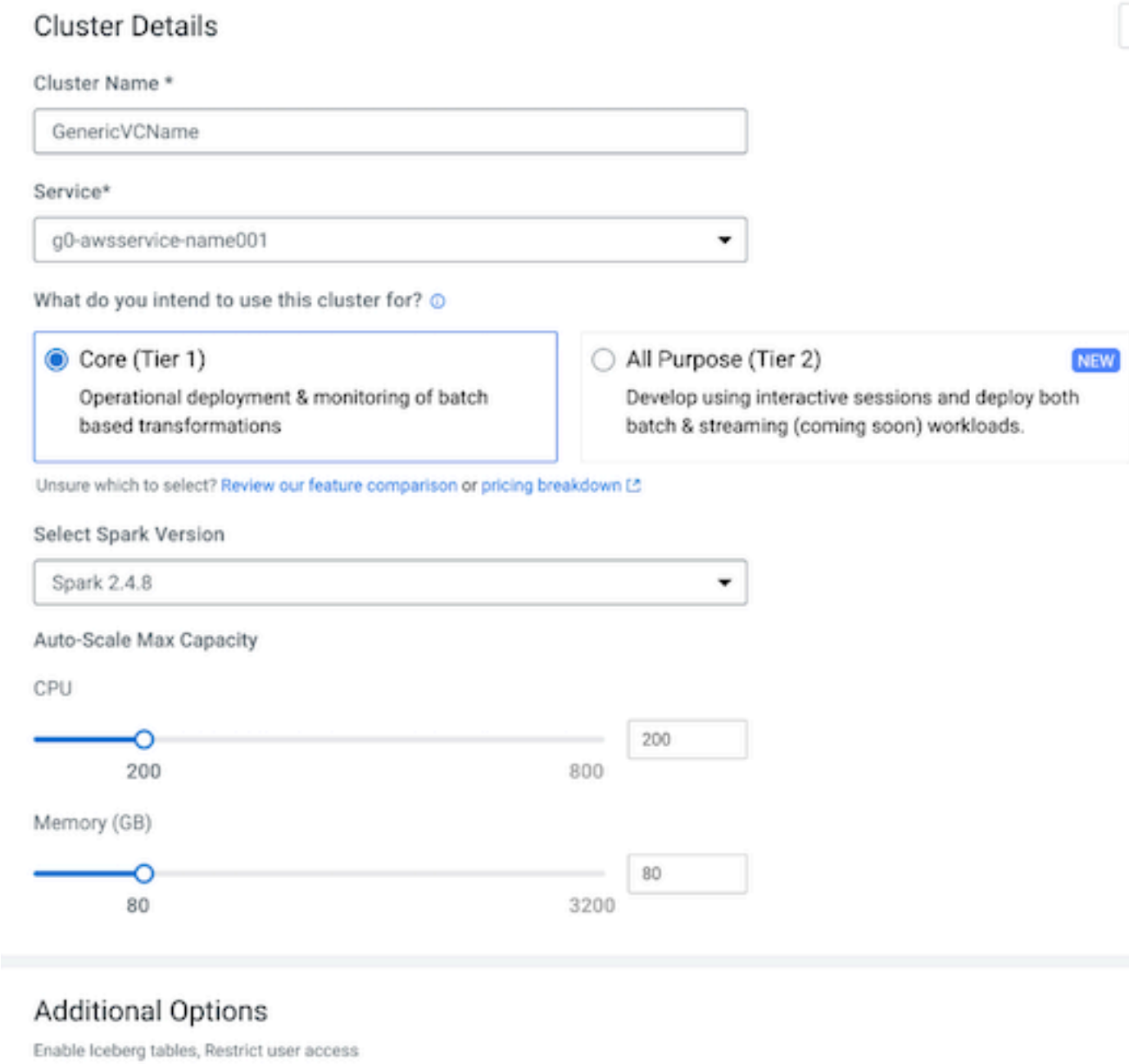

4. Enter a Cluster Name.

   Cluster names must include the following:

   - Begin with a letter
   - Be between 3 and 30 characters (inclusive)
   - Contain only alphanumeric characters and hyphens

5. Select the Service to create the virtual cluster in.

   The environment you selected before launching the wizard is selected by default, but you can use the wizard to create a virtual cluster in a different Cloudera Data Engineering service.

6. Select one of the following Cloudera Data Engineering cluster types:

   Note: These tiers are available in Cloudera Data Engineering 1.19 and above only.

   - Core (Tier 1)
   - All Purpose (Tier 2)

   Important: For both Microsoft Azure and Amazon Web Services, to avoid being charged for All purpose compute nodes, even if you do not create an All purpose Virtual Cluster:

   Navigate to the Service Details page > Configurations tab > Capacity & Costs > Autoscale Range and set the minimum value of All purpose On-demand Instances to 0.

   For more information, see Cloudera Data Engineering cluster types.
7. Select the Spark Version to use in the virtual cluster. You cannot use Spark 2 and Spark 3 in the same virtual cluster, but you can have separate Spark 2 and Spark 3 virtual clusters within the same Cloudera Data Engineering service. You can use multiple Spark 3.x in your virtual cluster.
8. Use the Auto-Scale Max Capacity sliders to set the maximum number of CPU cores and the maximum memory in gigabytes. The cluster will scale up and down as needed to run the submitted Spark applications.
9. Optional for spot instances enabled at the Cloudera Data Engineering service level: From the Driver and Executors will run on drop-down menu, select whether you want to run drivers and executors on spot instances or on-demand instances. By default, the driver runs on on-demand instances, and the executors run on spot instances. For SLA-bound workloads, select On-demand. For non-SLA workloads, Cloudera recommends leaving the default configuration to take advantage of the cost savings afforded by spot instances. For more information, see Cloudera Data Engineering Spot Instances.
10. Select Restrict Access to add access control for the virtual cluster. You can search for users to add by name or email address. You can manage users using the Cloudera Management Console. For more information, see Managing user access and authorization.
11. Click Configure Email Alerting (Technical Preview) if you want to receive notification mails.The email configuration options appear.
    a) You must provide at least Sender Email Address and SMTP Host information.
    b) Test SMTP Configs: Click Test SMTP Configs to test the configurations set for SMTP. This helps you to test the SMTP configuration before creating the cluster.

    Note: You will have the option to enable email alerts when you create or edit a job after the Virtual Cluster is created, but step 13 must be completed first.
12. Click Create.

Results

On the Cloudera Data Engineering Home page, select the environment to view the virtual cluster initialization status. You can also click the three-dot menu for the virtual cluster to view the logs.

Related Information

Recommendations for scaling Cloudera Data Engineering deployments

Viewing and managing virtual cluster details

You can view details such as configuration, metrics, and logs for existing virtual clusters. You can also view jobs associated with a cluster, clone a cluster, and delete a cluster

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

To view cluster details:

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu. The Administration page displays.
3. In the Services column, select the environment containing the virtual cluster you want to manage.
4. In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to manage.
5. On the Configuration tab, you can view details about the cluster, including the cluster name, Cloudera environment, and the CPU and memory auto-scale ranges.

The virtual cluster Overview page provides details about the virtual cluster, and includes links to several resources, such as the API documentation, and the Jobs API URL. You can switch between the following three tabs to view additional information:

Configuration

The Configuration tab lists details about the cluster, including the cluster name, Cloudera environment, and the CPU and memory auto-scale ranges. You can modify the virtual cluster quotas (CPU/memory) dynamically. Cloudera Data Engineering provides an Edit option to make the configuration changes dynamically which may take a few mintutes to update. For more information, see Editing virtual clusters linked below.

Charts

The Charts tab displays charts of virtual cluster metrics, such as network and filesystem usage.

Note: If your browser blocks third-party cookies, Charts may not display correctly. If this happens, add *.cloudera.com as an exception in your browser settings.

Security

The Security tab displays the Access Management that you can set for the virtual cluster. See Applying user and group access for virtual clustersfor more details.

Logs

The Logs tab displays the latest log entries for the virtual cluster.

Related Information

Editing virtual clusters

Applying user and group access for virtual clusters

Learn how to restrict or grant access to a virtual cluster for specific users or groups that you specify in Cloudera Data Engineering.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu. The Administration page displays.
3. In the Services column, select the environment containing the virtual cluster you want to manage.
4. In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to update security for.

5. Click the Security Access Add User or Group .

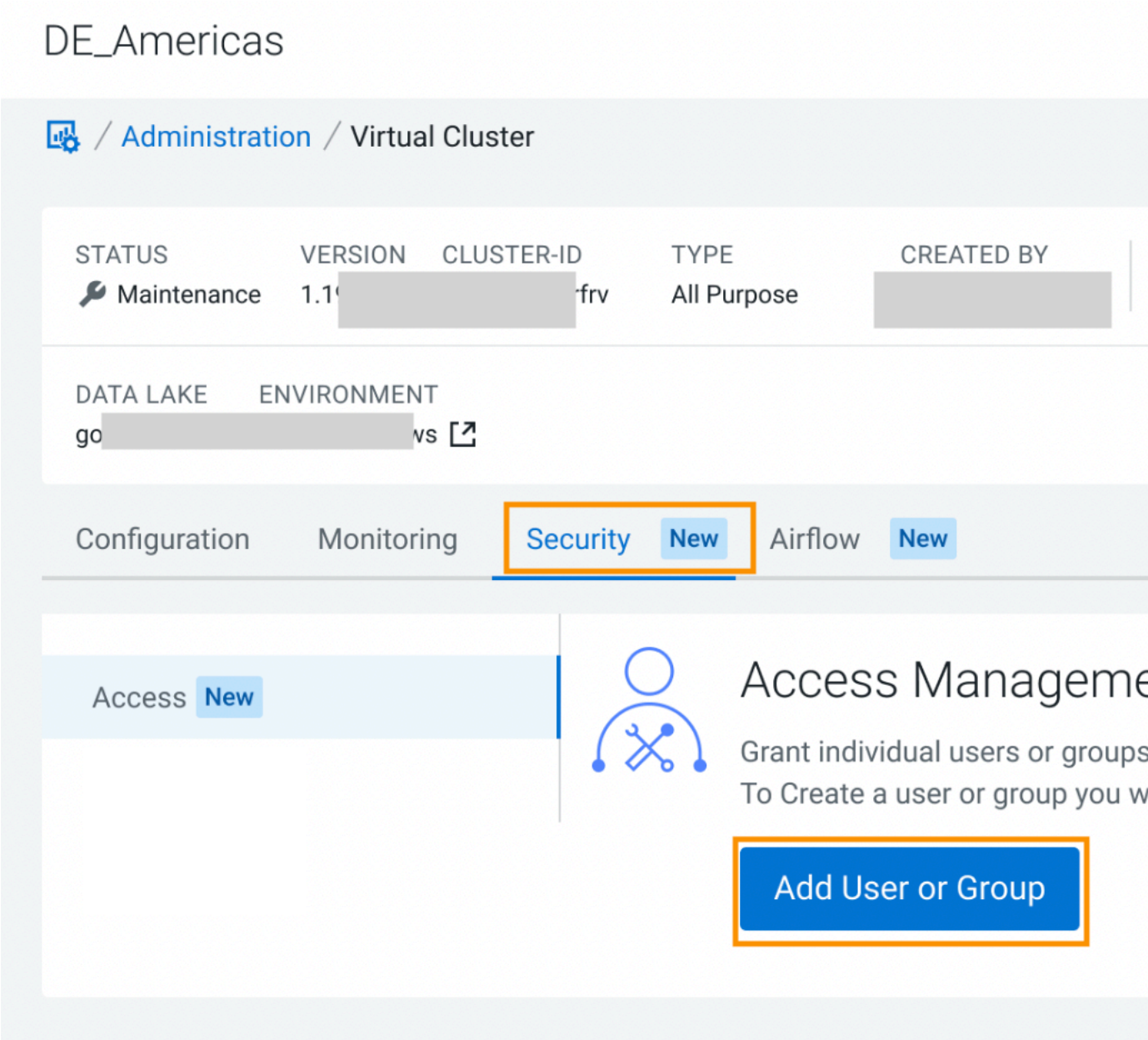

The Add User(s) or Group(s) screen is displayed.

6. In the Search for User(s) or Group(s) field, enter the name or email of the user(s) or group name(s).
7. Under Set Role(s) drop-down, select Read-only Admin or Admin.

   The Admin option can access endpoints including all jobs, resources, and sessions. The Read-only Admin option can only view all cluster activities but cannot modifiy the cluster or activities.
8. Click Add.

   The Access Management screen is displayed with the users and groups that you added. Only one role can be set each time, but selection can be modified by editing once the user or groups are added.
9. To add more users or groups, click Add.

   The Add User(s) or Group(s) screen is displayed.

   a) Repeat steps 6 through 8.

Editing user and group access for virtual clusters

Learn how to edit or delete a user or group access that you have added for a virtual cluster in Cloudera Data Engineering.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu. The Administration page displays.
3. In the Services column, select the environment containing the virtual cluster you want to manage.
4. In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to update security for.
5. Click the Security Access .

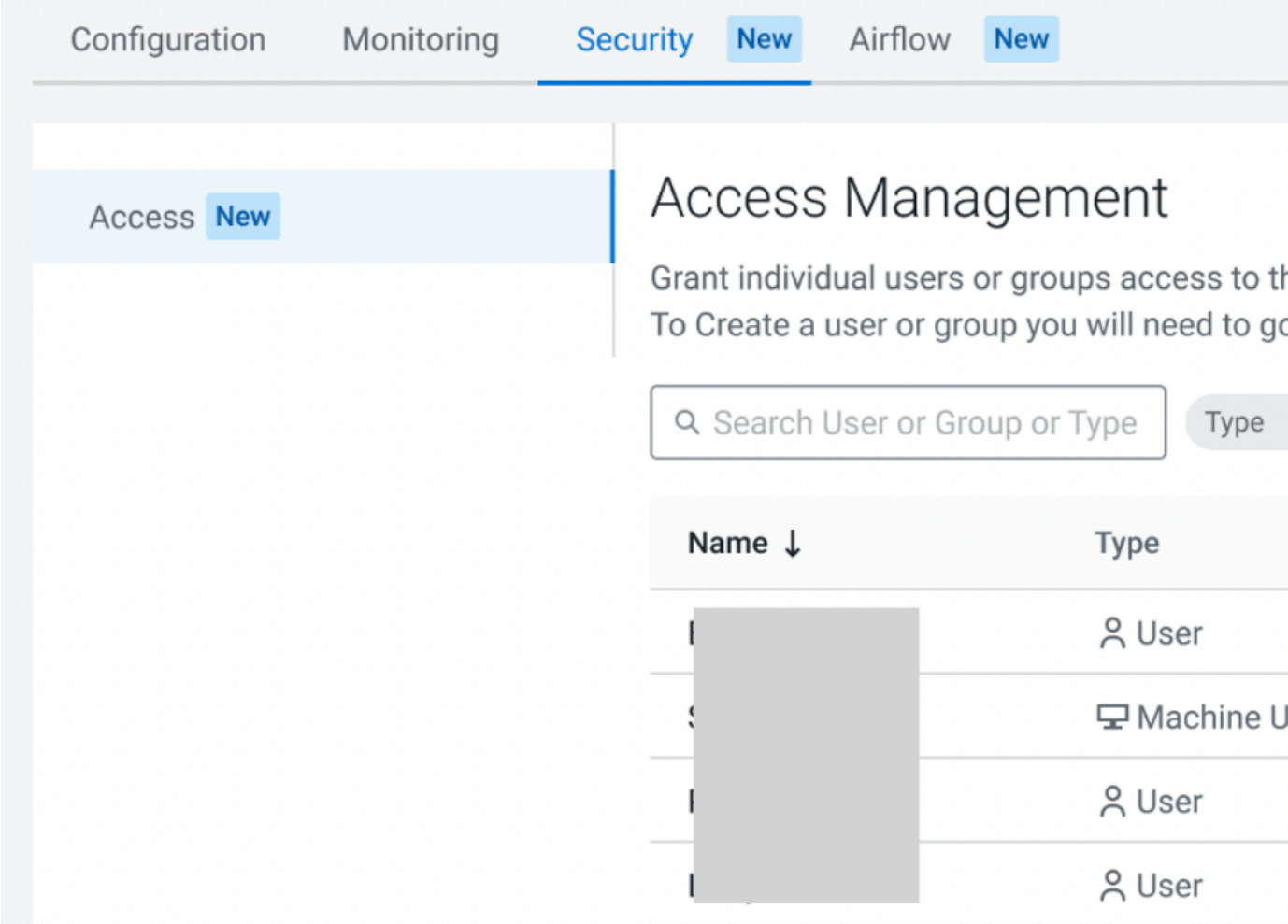

6. Locate the name of the user or group that you want to edit.
7. Click the Pencil icon to edit the current user or group.

   The Add User(s) or Group(s) screen is displayed.
8. If you want to update the user or group, in the Search for User(s) or Group(s) field, enter the name or email of the user(s) or group name(s).
9. If you want to edit the role, under Set Role(s) drop-down, select Read-only Admin or Admin.

   The Admin option can access endpoints including all jobs, resources, and sessions. The Read-only Admin option can only view all cluster activities but cannot modifiy the cluster or activities.

10. If you want to delete the user or group, locate the name or group of the user that you want to delete, and click the Trash icon.

Cloudera Data Engineering Virtual Cluster Access Controls Troubleshooting

Learn about troubleshooting scenarios for user and group access issues in Cloudera Data Engineering Virtual Clusters (VCs).

A user is added to a group with access to a VC, but the user still cannot access the VC

As the user is added to the group, do a user synchronization. For information about synchronizing users in an environment, see Performing user sync.

Note: If the user still cannot access the VC, it might be due to internal caching. Wait for five minutes and try again.

A user or group is not displayed in the drop-down list while adding roles

If the user or group exists in the Cloudera account, but the user is not displayed in the drop-down list, it is very likely that the user is given a role already. Check the current access list.

Adding access controls while creating a VC

From the UI, you can add access controls only after the VC is created. Using CLI or API, you can add access controls during VC creation.

A user is added, but if the user tries to access the VC, the UI shows a 403 error

The user needs to have either a DEUser or a DEAdmin role in the environment irrespective of the roles in the VC. For more information, see Giving access to Data Engineering users.

User access works, but group access does not

If the group access does not work as expected, do a user synchronization in the environment and wait for five minutes. If the user access still does not work, a miscommunication between the authorization server and the LDAP server can be the cause. To find such errors, access the kubernetes cluster and check for logs of the dex-authz pod in the dex-base- namespace.

Managing Virtual Cluster-level Spark configurations (Technical Preview)

Learn how to add, edit, or delete the Spark configurations at a Virtual Cluster (VC)-level so that the configurations apply to all the jobs in the Virtual Cluster, by default. You can edit the Spark configurations using the Cloudera Data Engineering UI, the CDP CLI, or the Cloudera Data Engineering API.

About this task

Earlier, the Spark configurations were added to the jobs at the job-level. If you had to apply the common configurations for all the jobs, then you had to add the configurations for each job individually.

Using this feature, you can apply the Spark configurations in the Virtual Cluster (VC) to all the jobs by default.

Before you begin

For information on how to manage Virtual Cluster-level Spark configurations using the API, see Managing Virtual Cluster-level Spark configurations using the API.

Procedure

Edit the Spark configurations using the Cloudera Data Engineering UI, the CLI, or the Cloudera Data Engineering API.

For Cloudera Data Engineering UI

a. Sign in to Cloudera Data Engineering UI and click Administration.
b. Select the relevant Cloudera Data Engineering Service.
c. Select the relevant Virtual Cluster in the Cloudera Data Engineering Service and click the Cluster Details option.
d. Click Configurations Spark .

A Spark page appears.

e. Under Configurations, Enter Spark Configurations as key-value pairs.

For example

```
is.config.vc=true
is.config.job=false
```

f. Click Apply Changes.

Result: The Spark configurations are updated and applied to the VC and the jobs in the Virtual Cluster. You can validate them by running the jobs and then verifying the Spark configurations in the Configuration tab in that particular job runtime.

For CDP CLI

- To create a VC with the Spark configurations, run the following command:

```
cdp de create-vc \
         --name                [***VIRTUAL-CLUSTER-NAME***] \
         --cluster-id          [***CLUSTER-ID***] \
         --cpu-requests        [NUMBER-OF-CPUS-REQUIRED***] \
         --memory-requests     [NUMBER-OF-MEMORY UNITS-REQUIRED***] \
         --spark-configs   \   [***KEY1***]=[***VALUE1***],[***KEY2***]=
[***VALUE2***]
```

For example

```
cdp de create-vc \
         --name               test-vc \
         --cluster-id         cluster-sq4rpnr2 \
         --cpu-requests       20 \
         --memory-requests     20 \
         --spark-configs   \
spark.executor.instances=2,spark.dynamicAllocation.maxExecutors=15,key1=
val1
```

- To update the Spark configurations of the VC, run the following command:

```
cdp de update-vc \
        --cluster-id        [***CLUSTER-ID***] \
        --vc-id             [***VC-ID***] \
        --spark-configs \   [***KEY1***]=[***VALUE1***][***KEY2***]=[***
VALUE2***]
```

For example

```
cdp de update-vc \
        --cluster-id      cluster-sq4rpnr2 \
        --vc-id           dex-app-w2rhk89r \
        --spark-configs \ spark.executor.instances=2,spark.dynamicAlloc
ation.maxExecutors=15,key1=val1
```

- To discard or delete all the Spark configurations of the VC, run the following command:

```
cdp de update-vc \
```

```
        --cluster-id        [***CLUSTER-ID***] \
        --vc-id             [***VC-ID***] \
        --discard-spark-configs
```

For example

```
cdp de update-vc \
        --cluster-id        cluster-sq4rpnr2 \
        --vc-id             dex-app-w2rhk89r \
        --discard-spark-configs
```

Editing virtual clusters

You can edit virtual cluster configurations in the UI or use a script to apply configurations to multiple jobs at once.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

For Edit in UI

To edit virtual cluster details:

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu. The Administration page displays.
3. In the Services column, select the environment containing the virtual cluster you want to manage.
4. In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to manage.
5. On the Configuration tab, you can modify the virtual cluster quotas (CPU/memory) dynamically. Cloudera Data Engineering provides an Edit option to make the configuration changes dynamically which may take a few minutes to update.

For Edit with Script

As an Administrator, you can further customize the virtual cluster by applying additional configurations all at once using the cde-utils script. Click cdp-cde-utils.sh to download the script. Customizations include applying spark configurations to all jobs, enabling and disabling gang schedueling, and initiating faster autoscaling times for nodes in an autoscaler cluster for Amazon Web Services (AWS). See the following cde-utils script and options to apply these changes:

Important: Before using the script, the user must export the kubeconfig file to ensure that the virtual cluster is accessible via kubectl.

Apply Spark configurations to all new jobs

This script allows users to add and update default configurations for all Spark jobs all at once. The options required are as follows:

- -h <host_name>: You must provide this option, and paste the domain of the URL into a text editor to identify the endpoint host. For example, if the URL is https://xyz.cde-vjgd8ksl.dsp-azur.xcu2-8y8x.dev.cldr.work/dex/ui/jobs, then paste the following :

```
xyz.cde-vjgd8ksl.dsp-azur.xcu2-8y8x.dev.cldr.work
```

- -c <spark_default_configurations>: You must provide option -c <spark_configuration> to add all the spark-configurations inside the single quotes (‘ ’) with a delimiter comma (,) at the virtual cluster level. There must be at least one configuration in the command. For example:

```
./cde-utils.sh add-spark-config-in-virtual-cluster -h <host_
name> -c ‘spark.driver.supervise=“false”,spark.executor.inst
ances=“2”’
```

  The ,user can add multiple configurations as shown above in the example: ‘ “key1”=“val1”, “key2”=“val2” , ...’ . The , <host_name> is is determined by the provided information in the host name option.

Enabling and disabling gang scheduling

This script enables and disables gang scheduling at the virtual cluster level for all jobs that are inside the virtual cluster. The options required are as follows:

- -h <host_name>: You must provide this option, and paste the domain of the URL into a text editor to identify the endpoint host. For example, if the URL is https://xyz.cde-vjgd8ksl.dsp-azur.xcu2-8y8x.dev.cldr.work/dex/ui/jobs, then paste the following:

```
xyz.cde-vjgd8ksl.dsp-azur.xcu2-8y8x.dev.cldr.work
```

- --gang-scheduling <gang_scheduling_status>: You must provide this option to toggle the status of the gang scheduling at the virtual cluster level. You can only use enable or disable options. To disable gang scheduling, see the following example:

```
./cde-utils.sh add-spark-config-in-virtual-cluster -h <host_
name> --gang-scheduling disable
```

  The , <host_name> is determined by the provided information in the host name option.

  Important: You must provide either configurations option (-c) or gang-scheduling option(--gang-scheduling) for this sub-command. You may also provide both options.

Initiating faster autoscaling times

You can increase the speed of autoscaling times for nodes in an autoscaler cluster for Amazon Web Services (AWS) by using a script. This script allows you to speed up the scale down progress for situations where you scale up to hundreds of nodes. This is a cost-saving measure. The options required for scaling down nodes are as follows:

- -i <interactive_menu>: You can use this option to use an interactive menu that uses the arrow keys to navigate through the menu options. For example:

```
./cde-utils.sh edit-cluster-autoscaler -i
```

  Important: In the edit-cluster-autoscaler sub command, you can provide all the other options except -i <interactive_menu>, or only provide -i <interactive_menu> to change the value of other options using the interactive menu.

- --scale-down-delay-after-add: Reduce the value of this option which specifies the interval after scaling up when scale down evaluation resumes. This allows you to speed up the process. The default value is 10 minutes. Permitted values are 10 seconds, 5 minutes, and 1 hour. The format is as follows: <time>[s / m / h]. For Example:

```
./cde-utils.sh edit-cluster-autoscaler --scale-down-delay-af
ter-add 10m
```

- --scale-down-delay-after-delete: You can decrease the value for this option which is the duration of time after node deletion when the scale down evaluation resumes. The default is 10 seconds.

Permitted values are 10 seconds, 5 minutes, and 1 hour. The format is as follows: <time>[s / m / h]. For example:

```
./cde-utils.sh edit-cluster-autoscaler --scale-down-delay-af
ter-delete 10s
```

- --scale-down-delay-after-failure: You can decrease the value for this option which is a duration of time after a scale down failure when the scale down evaluation resumes. The default value is 3 minutes. Permitted values are 10 seconds, 5 minutes, and 1 hour. The format is as follows: <time>[s / m / h]. For example:

```
./cde-utils.sh edit-cluster-autoscaler --scale-down-delay-af
ter-failure 3m
```

- --scale-down-unneeded-time: To speed up the process, the user can decrease the value of this option which specifies the amount of time before a node qualifies for scale down. The default value is 10 minutes. Permitted values are 10 seconds, 5 minutes, and 1 hour. The format is as follows: <time>[s / m / h]. For example:

```
./cde-utils.sh edit-cluster-autoscaler --scale-down-unneeded-
time 10m
```

- --unremovable-node-recheck-timeout: You can reduce the value for this option which is the duration of the timeout before the unremovable node is checked again. The default value is 5 minutes. Permitted values are 10 seconds, 5 minutes, and 1 hour. The format is as follows: <time>[s / m / h]. For example:

```
./cde-utils.sh edit-cluster-autoscaler --unremovable-node-re
check-timeout 5m
```

Example use cases with commands

These example use cases and commands show how you can use the cde-utils script.

- Initiate faster autoscaling times with the command below. The order of options is not significant in this command:

```
./cde-utils.sh edit-cluster-autoscaler --scale-down-delay-af
ter-add 1m --scale-down-delay-after-failure 1m --scale-down-
delay-after-delete 5s
--scale-down-unneeded-time 1m --unremovable-node-recheck-timeo
ut 1m
```

- Add a single Spark configuration at the virtual cluster level:

```
./cde-utils.sh add-spark-config-in-virtual-cluster -h <host_
name> -c spark.executor.instances="2"
```

- Add multiple spark configurations at the virtual cluster level:

```
./cde-utils.sh add-spark-config-in-virtual-cluster -h <host_
name> -c 'spark.driver.supervise="false",spark.executor.inst
ances="2"'
```

- Enable gang scheduling:

```
./cde-utils.sh add-spark-config-in-virtual-cluster -h <host_
name> --gang-scheduling enable
```

Related Information

Accessing the Kubernetes dashboard

Support Video: Enabling kubectl for Cloudera Data Engineering

Locking and unlocking a Cloudera Data Engineering Service

Locking a Cloudera Data Engineering Service freezes the configuration of a Service and its corresponding Virtual Clusters. When a Service is locked, you are unable to edit, add, or delete a Service and its Virtual Clusters.

About this task

You may want to lock a Service in scenarios where you don't want configuration changes to occur during a planned upgrade of Cloudera Data Engineering. The lock and unlock options will display interchangeably, depending on the state that the Service is in. For example, if the Service is unlocked, you will only see the Lock Service option when you complete the steps below. Once your Service is locked, you will see the option to Unlock it as you complete the steps below.

Before you begin

Note that locking a Service is a preview feature starting in Cloudera Data Engineering 1.19. Ensure that you are on Cloudera Data Engineering 1.19 and above.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration.
3. Locate the Service that you want to lock or unlock.
4. Click ⋮ on the Service, and select the Lock Service or Unlock Service option.
5. Enter the name of the Service that you want to lock or unlock as this confirms your choice.
6. Click the appropriate Lock Service or Unlock button.

Deleting virtual clusters

If you no longer need a particular virtual cluster, you can delete it. Deleting a virtual cluster instantly deletes all metadata associated with the cluster, including job configurations and log files. Do not delete a virtual cluster unless you are certain you no longer need access to any of the applications, jobs, or log files. Job run history is maintained separately, and is not deleted even if you delete a virtual cluster.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. On the Cloudera Data Engineering Overview page, select the environment containing the virtual cluster that you want to delete.

2. In the Virtual Clusters column for the selected environment, click the menu icon at the top right of the virtual cluster that you want to delete, and then click Delete.

   Warning: Deleting a virtual cluster deletes all metadata associated with the cluster, including job configurations and log files. Do not do this unless you are certain that you no longer need any of these.

3. Confirm that you want to delete the cluster by entering the cluster name and then clicking Delete.

Accessing non-public Cloudera Data Engineering virtual clusters via SOCKS proxy

A SOCKS proxy server, or jump host, allows your web browser to connect securely to a non-public Cloudera Data Engineering endpoint. When you connect to the Cloudera Data Engineering endpoint using its internal IP, the browser connects to the proxy server, which performs the required SSH tunneling, and forwards the request to the internal Cloudera Data Engineering endpoint.

About this task

This topic describes how to configure a SOCKS proxy to access Cloudera Data Engineering Virtual Clusters on non-publicly routable VPCs.

Note: Using the below approach all the web traffic in the launched browser instance will be routed via the jump host.

Procedure

1. In a VPC with a public subnet, which can also route to the target VPC where the Cloudera Data Engineering Load Balancer is running, create an EC2 instance for your SOCKS server (for example, my-ec2-jump-host) with a public IP and an SSH key-pair (for example, my-key-file.pem).

   The instance does not need to be large, for example on AWS a micro instance is usually sufficient, but may need to be larger depending on how many users are using it concurrently.

   Depending on whether you want multiple users to share the jump host or have everyone create their own server, pick the SSH key pair for the instance accordingly.

   Note: Make sure to apply a locked down security group to your jump host instance to ensure only users from trusted IP ranges can SSH into the server.

   Related AWS Documentation: Getting Started with Amazon EC2 Instances, Amazon EC2 Key Pairs

   a) Set up a SOCKS proxy server via SSH to the EC2 instance using below command:

   ```
   nohup ssh -i "SSH-KEY-PAIR" -CND <SOCKS-PROXY-PORT> ec2-user@<JUMP-HOST-PUBLIC-IP> &
   ```

```
Example: nohup ssh -i "my-key-file.pem" -CND 8157 ec2-user@<public ip for
 my-ec2-jump-host> &
```

where

- nohup(optional) is a POSIX command to ignore the HUP (hangup) signal so that the proxy process is not terminated automatically if the terminal process is later terminated.
- my-key-file.pem is the private key you used to create the EC2 instance where the SOCKS server is running.
- C sets up compression.
- N suppresses any command execution once established.
- D 8157 sets up the SOCKS 5 proxy on the port. The port number 8157 in this example is arbitrary, but must match the port number you specify in your browser configuration in step 2.
- ec2-user is the AMI username for the EC2 instance. The AMI username can be found in the details for the instance displayed in the AWS Management Console on the Instances page under the Usage Instructions tab.
- <public ip for my-ec2-jump-host>is the public IP address of the EC2 instance running the SOCKS server you created above.
- & (optional) causes the SSH connection to run as an operating system background process, independent of the command shell. Without the &, you leave your terminal open while the proxy server is running and use another terminal window to issue other commands.

2. Configure Your Browser to Use the Proxy.
Using Google Chrome

By default, Google Chrome uses system-wide proxy settings on a per-profile basis. To get around that you can start Chrome using the Command Line and specify the following:

- The SOCKS proxy port to use (must be the same value used in step 1, 8157 in example above)
- The profile to use (new profile created in the example below)

a) Run the following commands to create a new profile and launch a new instance of Chrome that does not interfere with any currently running instance.

For Linux:

```
/usr/bin/google-chrome \
--user-data-dir="$HOME/chrome-with-proxy" \
--proxy-server="socks5://localhost:<SOCKS-PROXY-PORT>"
```

For MacOS

```
"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" \
--user-data-dir="$HOME/chrome-with-proxy" \
--proxy-server="socks5://localhost:<SOCKS-PROXY-PORT>"
```

For Windows

```
"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" ^
--user-data-dir="%USERPROFILE%\chrome-with-proxy" ^
--proxy-server="socks5://localhost:<SOCKS-PROXY-PORT>"
```

Results

These steps enable you to navigate to any Cloudera Data Engineering virtual cluster in the browser launched using SOCKS proxy.

Cloudera Data Engineering 1.23.0

Managing Cloudera Data Engineering jobs

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating jobs in Cloudera Data Engineering.....5
Support for Spark Structured Streaming in Cloudera Data Engineering (Technical Preview).....7

Cloudera Data Engineering example jobs and sample data.....8

Accessing Amazon S3 data using Cloudera Data Engineering.....11

Using Apache Iceberg in Cloudera Data Engineering.....12
Prerequisites and limitations for using Iceberg.....13
Accessing Iceberg tables.....14
Editing a storage handler policy to access Iceberg files on the file system.....14
Creating a SQL policy to query an Iceberg table.....16
Creating Virtual Cluster with Spark 3.....18
Creating a new Iceberg table from Spark 3.....18
Configuring Hive Metastore for Iceberg column changes.....19
Importing and migrating Iceberg table in Spark 3.....19
Importing and migrating Iceberg table format v2.....20
Configuring Catalog.....22
Loading data into an unpartitioned table.....22
Querying data in an Iceberg table.....23
Updating Iceberg table data.....23
Iceberg library dependencies for Spark applications.....23

Creating a Git repository in Cloudera Data Engineering (Technical Preview).....24

Managing jobs in Cloudera Data Engineering.....27

Running Jobs in Cloudera Data Engineering.....27

Scheduling jobs in Cloudera Data Engineering.....28

Creating an ad-hoc job in Cloudera Data Engineering.....29

Configuring logging in Cloudera Data Engineering jobs.....30

Deleting Jobs in Cloudera Data Engineering.....31

Best practices for building Apache Spark applications.....31

Creating jobs in Cloudera Data Engineering

A job in Cloudera Data Engineering consists of defined configurations and resources (including application code). Jobs can be run on demand or scheduled.

Before you begin

In Cloudera Data Engineering, jobs are associated with virtual clusters. Before you can create a job, you must create a virtual cluster that can run it. For more information, see Creating virtual clusters.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also create a job by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. To view Cloudera Data Engineering Services, click Administration on the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Important: The Cloudera Data Engineering jobs API implicitly adds the default DataLake filesystem to the Spark configuration to save the user having to do that. If you need to reference other buckets, you can set the spark.yarn.access.hadoopFileSystems parameter with the extra comma-separated buckets needed. If you set this parameter in your application code before creating the session, it might override the default setting, leading to errors.

Procedure

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the service that contains the virtual cluster that you want to create a job for.
3. In the Virtual Clusters column on the right, locate the virtual cluster that you want to use and click the View Jobs icon.
4. In the left hand menu, click Jobs.
5. Click the Create Job button.

6. Provide the Job Details:
   a) Select Spark for the job type. For Airflow job types, see Apache Airflow in Cloudera Data Engineering.
   b) Specify the Name.
   c) Select Resources, URL, or Repository for your application file, and provide or specify the file. You can upload a new file or select a file from an existing resource.

      If you select URL and specify an Amazon AWS S3 URL, add the following configuration to the job:

      config_key: spark.hadoop.fs.s3a.delegation.token.binding

      config_value: org.apache.knox.gateway.cloud.idbroker.s3a.IDBDelegationTokenBinding

      If you want to use files from a repository, you must first create a repository. Once the repository is created, you can select Repository, click Add from Repository and select the file. Then, click Select File.
   d) If your application code is a JAR file, specify the Main Class.
   e) Specify arguments if required. You can click the Add Argument button to add multiple command arguments as necessary.
   f) Enter Configurations if needed. You can click the Add Configuration button to add multiple configuration parameters as necessary.

      Important: For Spark jobs, setting the `spark.app.id` property at the Spark job level configuration or within the Spark application code is not supported in Cloudera Data Engineering.

   g) If your application code is a Python file, select the Python Version, and optionally select a Python Environment.
7. Click Advanced Configurations to display more customizations, such as additional files, initial executors, executor range, driver and executor cores and memory.

   By default, the executor range is set to match the range of CPU cores configured for the virtual cluster. This improves resource utilization and efficiency by allowing jobs to scale up to the maximum virtual cluster resources available, without manually tuning and optimizing the number of executors per job.
8. Click Schedule to display scheduling options.

   You can schedule the application to run periodically using the Basic controls or by specifying a Cron Expression.

   Note: Scheduled job runs start at the end of the first full schedule interval after the start date, at the end of the scheduled period. For example, if you schedule a job with a daily interval with a start_date of 14:00, the first scheduled run is triggered at the end of the next day, after 23:59:59. However if the start_date is set to 00:00, it is triggered at the end of the same day, after 23:59:59.
9. If you provided a schedule, click Schedule to create the job. If you did not specify a schedule, and you do not want the job to run immediately, click the drop-down arrow on Create and Run and select Create. Otherwise, click Create and Run to run the job immediately.

10. Optional: Toggle Alerts to send mail to the email address that you choose. You have the option to select Job Failure to send an email upon job failure, and Job SLE Miss to send an email on a Job service-level agreement miss.

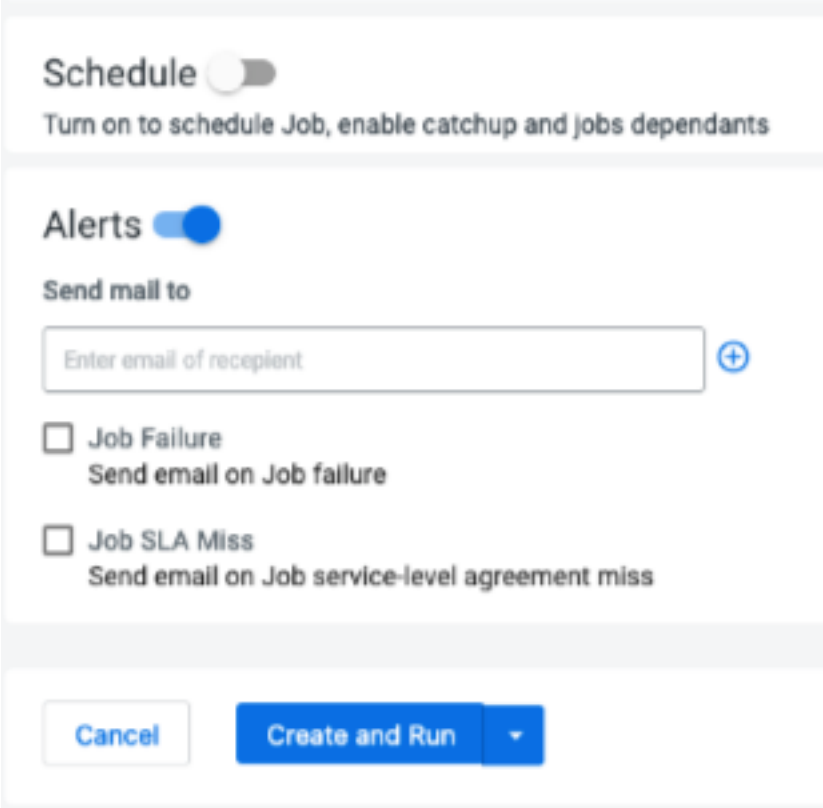

Note: The Alerts option will only display if you have selected Configure Email Alerting during Virtual Cluster creation.

Support for Spark Structured Streaming in Cloudera Data Engineering (Technical Preview)

Understand the supported features and limitations related to Spark Structured Streaming in Cloudera Data Engineering. Spark Structured Streaming is compatible with both Spark 2 and Spark 3.

Note: This feature is in Technical Preview and not recommended for production deployments. Cloudera recommends that you try this feature in test or development environments.

Spark Structured Streaming is supported with the following limitations:

- For checkpointing on S3 for Amazon Web Services (AWS), Cloudera recommends using of the abortable stream based checkpoint manager with the following command:
  - --conf spark.sql.streaming.checkpointFileManagerClass=org.apache.spark.internal.io.cloud.AbortableStreamBasedCheckpointFileManager
- For authentication with delegation tokens that is supported for Kafka, use the following:
  - For Spark 2, pass the following configuration: --conf spark.kafka.bootstrap.servers=<kafka_broker_list>
  - For Spark 3, pass the following configuration where <clusterName> is an arbitrary name used for grouping the configuration: --conf spark.kafka.clusters.<clusterName>.auth.bootstrap.servers=<kafka_broker_list>
- Hive Warehouse Conector (HWC) is not supported.
- Cloudera Observability does not report information on Spark Streaming jobs.

- The visual profiling and Cloudera Data Engineering deep analysis features are not supported.
- Checkpointing with Amazon Elastic File System (EFS) is not supported.
- Spark dynamic allocation is not supported.

  You may use spark.streaming.dynamicAllocation, but this option is only available for Discretized streams (DStreams).
- S3 checkpoints may have issues with Cloudera Runtime versions 7.2.8 or lower.
- While running streaming with Kafka, set the auto.offset.reset parameter to "latest", to avoid out-of-memory errors.
- Schema Registry with Spark 3 is not supported.

Related Information

Spark Streaming
Spark Structured Streaming

Cloudera Data Engineering example jobs and sample data

Cloudera Data Engineering provides a suite of example jobs that operate on example data to showcase its core capabilities and make the onboarding easier. The example jobs are a combination of Spark and Airflow jobs, which include scenarios such as reading and writing from object storage, running an Airflow DAG, and expanding on Python capabilities with custom virtual environments. Once loaded, these jobs can be run on demand or scheduled. The sample data will be loaded into the environment's default Data Lake location.

Before you begin

In Cloudera Data Engineering, jobs are associated with virtual clusters. Before you can create a job, you must register a Cloudera environment and Data Lake, and create a Cloudera Data Engineering Service and virtual cluster. For more information, see Environments, Enabling Cloudera Data Engineering service, and Creating virtual clusters .

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also load an example job by opening a new Virtual Cluster and clicking Load Example Jobs. To view Cloudera Data Engineering Services, click Administration on the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

About this task

You must run the example jobs with a user who is not the Local Administrator, that is, the user must to have been granted DEUser or DEAdmin privileges in the environment associated with your DE workspace. Also ensure you have enough resources to run these example jobs. Below is the description of the different example jobs:

Table 1: Example Jobs

Job	Description
example-load-data	Loads the sample data onto the environment data lake. This job runs only once and is then deleted. **Note:** This will need to be run manually first if the sample jobs are loaded in any user defined virtual clusters. If the example-load-data job fails, contact Cloudera Support to recreate the example-load-data job.

Job	Description
example-virtual-env	Demonstrates Cloudera Data Engineering job configuration that utilizes Python Environment resource type to expand pyspark features via custom virtual env. This example adds pandas support. **Note:** You cannot run this job in an air-gapped environment.
example-resources	Demonstrates Cloudera Data Engineering job configuration utilizing file-based resource type. Resources are mounted on Spark driver and executor pods. This example uses an input file as a data source for a word-count Spark app. The driver stderr log contains the word count.
example-resources-schedules	Demonstrates scheduling functionality for Spark job in Cloudera Data Engineering. This example schedules a job to run at 5:04am UTC each day.
example-spark-pi	Demonstrates how to define a Cloudera Data Engineering job. It runs a SparkPi using a scala example jar located on a s3 bucket. The driver stderr log contains the value of pi. **Note:** You cannot run this job in an air-gapped environment.
example-cdeoperator	Demonstrates job orchestration using Airflow. This example uses a custom Cloudera Data Engineering Operator to run two Spark jobs in sequence, mimicking a pipeline composed of data ingestion and data processing. **Note:** You cannot run this job in an air-gapped environment.
example-object-store	Demonstrates how to access and write data to object store on different form factors: S3, ADLS, and HDFS. This example reads data already staged to object store and makes changes and then saves back the transformed data to object store. The output of the query ran on the object store table can be viewed in the driver stderr log.
example-iceberg	Demonstrates support for iceberg table format. This example reads raw data from object store and saves data in iceberg table format and showcases iceberg metadata inforation, such as snapshots.

Procedure

1. In the Cloudera Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the service containing the virtual cluster where you want to create the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster where you want to create the application.

4. Select Load Example Jobs from the two options that appear.

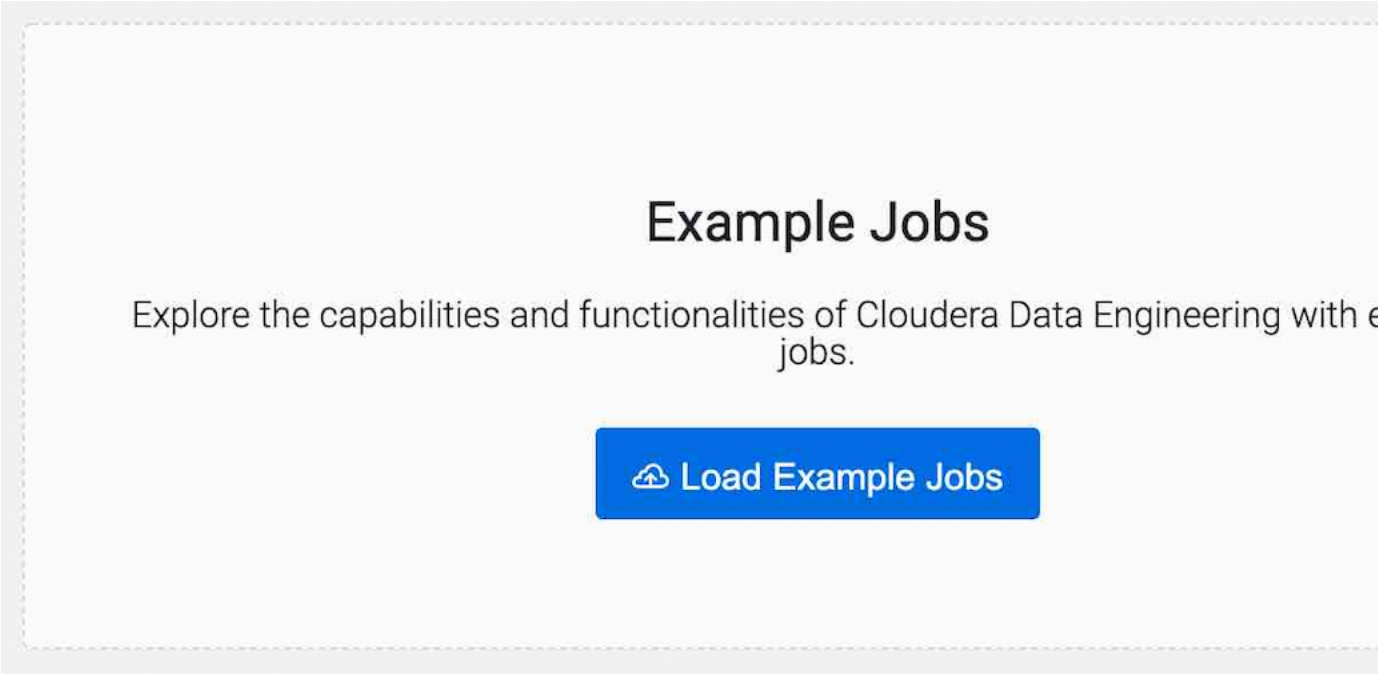

Note: You will see this window only if you have no existing jobs in the virtual cluster.

5. If you have existing jobs in the virtual cluster, click ⋮ on the jobs page to Load Example Jobs.

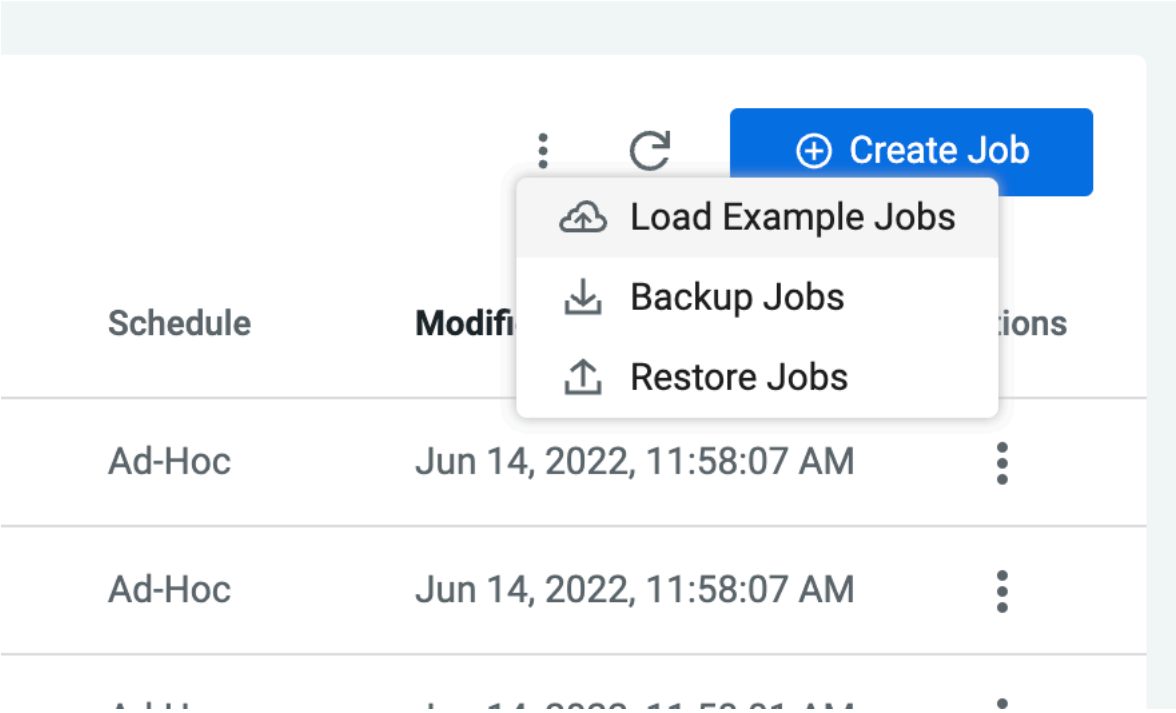

6. A dialog box appears explaining the example jobs and sample data. Click Confirm to load example jobs and sample data.

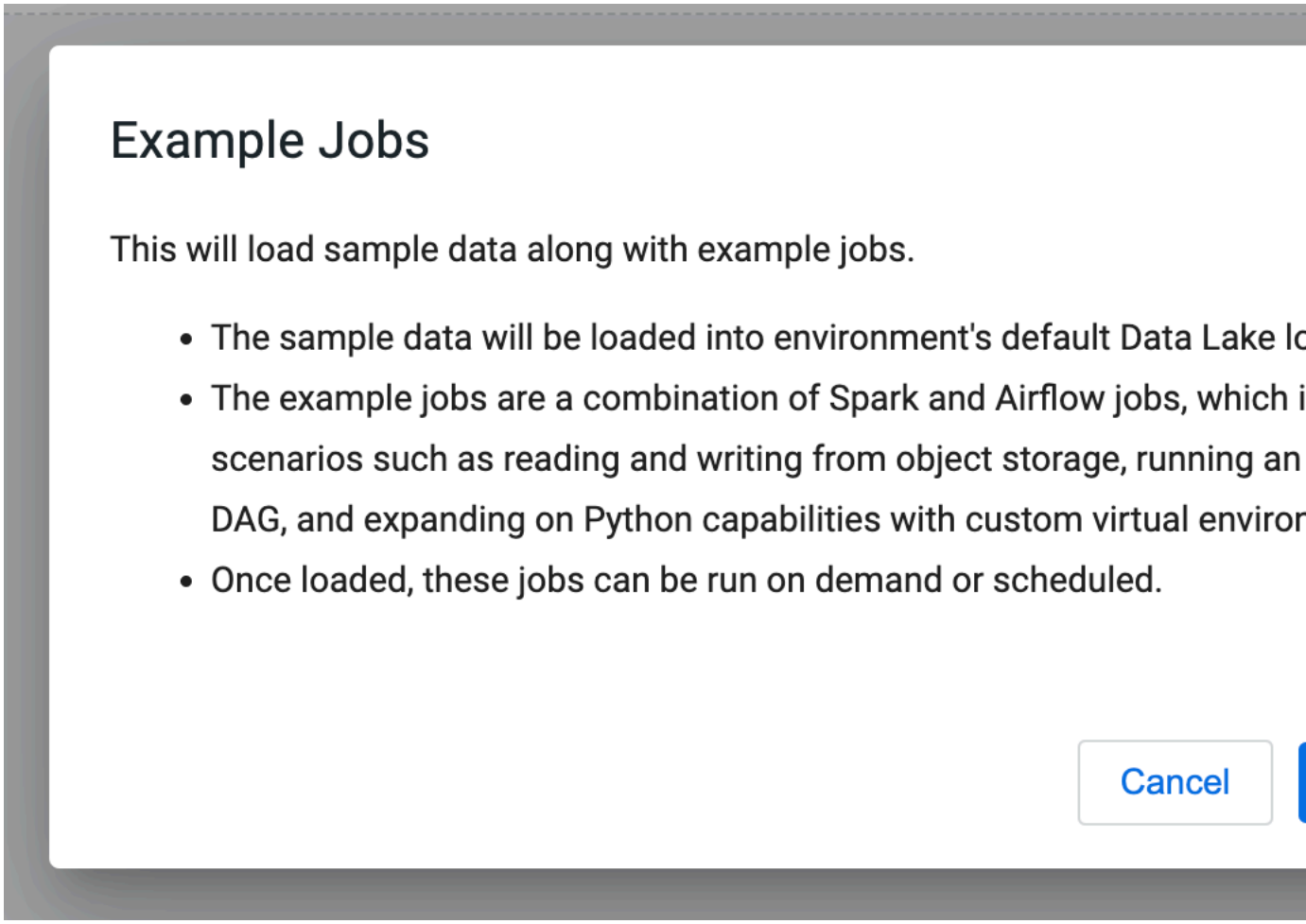

Results

Example jobs will be loaded in the virtual cluster and sample data will be loaded in the environment's Data Lake location.

Accessing Amazon S3 data using Cloudera Data Engineering

Some additional configuration is required to access Amazon S3 buckets from a Cloudera Data Engineering job.

Important: Cloudera components writing data to S3 are constrained by the inherent limitation of Amazon S3 known as *eventual consistency*. For more information, see Limitations of Amazon S3 in the Cloudera Runtime documentation.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also configure a job by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. To view Cloudera Data Engineering Services, click Administration on the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

To access data stored in Amazon S3 from Spark applications, use Hadoop file APIs (SparkContext.hadoopFile, Java HadoopRDD.saveAsHadoopFile, SparkContext.newAPIHadoopRDD, and JavaHadoopRDD.saveAsNewAPIHadoop File) for reading and writing RDDs, providing URLs of the form s3a://*BUCKET_NAME*/path/to/file.

Make sure that your user or group is mapped to an AWS IAM role with access to the buckets you need. For more information, see Onboarding Cloudera users and groups for cloud storage. In particular, make sure that the IAM role you are mapped to has the getBucketLocation permission on the buckets you need access to.

You must also configure the spark.kerberos.access.hadoopFileSystems (for Spark 3 jobs) or spark.yarn.access.ha doopFileSystems (for Spark 2 jobs) parameter to include the buckets you need to access. You can do this using the Cloudera Data Engineering job configuration, or at runtime as a command line parameter.

For example:

Job configuration

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster with the job you want to configure.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster with the job you want to configure.
4. In the left hand menu, click Jobs.
5. Click on the job you want to configure.
6. Go to the Configuration tab.
7. Click Edit.
8. Click the Add Configuration icon.
9. Enter spark.kerberos.access.hadoopFileSystems (for Spark 3 jobs) or spark.yarn.access.hadoop FileSystems (for Spark 2 jobs) in the config_key field.
10. In the config_value field, add a comma-separated list of buckets that the job requires access to, with no spaces after the commas. For example:

```
s3a://bucket1,s3a://bucket2
```

cde spark submit

If you are submitting a job using the CDE CLI, use the --conf command flag. For example:

Spark 3:

```
cde spark submit --conf "spark.kerberos.access.hadoopFileSystems
=s3a://bucket1,s3a://bucket2" ...
```

Spark 2:

```
cde spark submit --conf "spark.yarn.access.hadoopFileSystems=s3a
://bucket1,s3a://bucket2" ...
```

Using Apache Iceberg in Cloudera Data Engineering

Cloudera Data Engineering supports Apache Iceberg which provides a table format for huge analytic datasets in the cloud. Iceberg enables you to work with large tables, especially on object stores, and supports concurrent reads and writes on all storage media. You can use Cloudera Data Engineering virtual clusters running Spark 3 to interact with Apache Iceberg tables.

Prerequisites and limitations for using Iceberg

Learn about the supported versions for Cloudera Data Engineering, Spark, and Data Lake to use with Apache Iceberg Cloudera Data Engineering.

To use Apache Iceberg in Cloudera Data Engineering, you'll need the following prerequisites:

- Spark 3.2 or higher
- A compatible version of Data Lake as listed in Cloudera Data Engineering and Data Lake compatibility linked below
- Cloudera Data Engineering 1.16 or higher
- **Note:** Only AWS is supported in Cloudera Data Engineering 1.16 (which supports Iceberg 0.13)
- AWS or Azure is supported starting in Cloudera Data Engineering 1.17-h1 (which supports Iceberg 0.14)

Limitations

- The use of Iceberg tables as Structured Streaming sources or sinks is not supported.
- PyIceberg is not supported. Using Spark SQL to query Iceberg tables in PySpark is supported.
- Iceberg supports two timestamp types:
  - timestamp (without timezone)
  - timestamptz (with timezone)

  In Spark 3.3 and earlier, Spark SQL supports a single TIMESTAMP type, which maps to the Iceberg timestamptz type. However, Impala is unable to write to Iceberg tables with timestamptz columns. To create Iceberg tables from Spark with timestamp rather than timestamptz columns, set the following configurations to true:

  - spark.sql.iceberg.handle-timestamp-without-timezone
  - spark.sql.iceberg.use-timestamp-without-timezone-in-new-tables

  Configure these properties only on Spark 3.3 and earlier.

  Spark still handles the timestamp column as a timestamp with local timezone. Inconsistent results occur unless Spark is running in UTC.
- Iceberg tables with equality deletes do not support partition evolution or schema evolution on Primary Key columns.

  Users should not do partition evolution on tables with Primary Keys or Identifier Fields available, or do Schema Evolution on Primary Key columns, Partition Columns, or Identifier Fields from Spark.

Iceberg table format version 2

Iceberg table format version 2 (v2) is available starting in Iceberg 0.14. Iceberg table format v2 uses row-level UPDATE and DELETE operations that add deleted files to encoded rows that were deleted from existing data files. The DELETE, UPDATE, and MERGE operations function by writing delete files instead of rewriting the affected data files. Additionally, upon reading the data, the encoded deletes are applied to the affected rows that are read. This functionality is called merge-on-read.

To use Iceberg table format v2, you'll need the following prerequisites:

- Cloudera Data Engineering 1.17-h1 or higher
- Iceberg 0.14
- Spark 3.2 or higher

With Iceberg table format version 1 (v1), the above-mentioned operations are only supported with copy-on-write where data files are rewritten in their entirety when rows in the files are deleted. Merge-on-read is more efficient for writes, while copy-on-write is more efficient for reads.

Note: Unless otherwise indicated, the operations in the subsequent documentation apply to both v1 and v2 formats.

Related Information

Cloudera Data Engineering and Data Lake compatibility versions

Accessing Iceberg tables

Cloudera uses Apache Ranger to provide centralized security administration and management. The Ranger Admin UI is the central interface for security administration. You can use Ranger to create two policies that allow users to query Iceberg tables.

How you open the Ranger Admin UI differs from one Cloudera service to another. In Management Console, you can select your environment, and then click Environment Details Quick Links Ranger .

You log into the Ranger Admin UI, and the Ranger Service Manager appears.

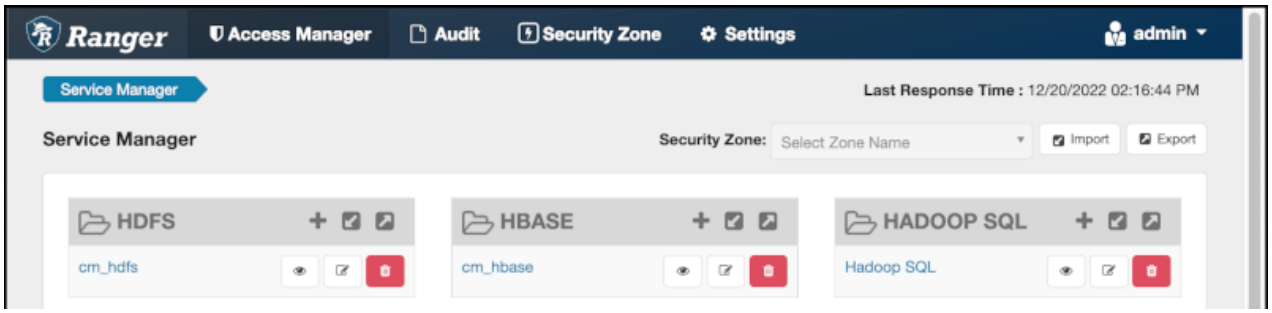

Policies for accessing tables on S3

The default policies that appear differ from service to service. You need to set up two Hadoop SQL policies to query Iceberg tables:

- One to authorize users to access the Iceberg files

  Follow steps in "Editing a policy to access Iceberg files" below.
- One to authorize users to query Iceberg tables

  Follow steps in "Creating a policy to query an Iceberg table" below.

Prerequisites

- Obtain the RangerAdmin role.
- Get the user name and password your Administrator set up for logging into the Ranger Admin.

  The default credentials for logging into the Ranger Admin Web UI are admin/admin123.

Editing a storage handler policy to access Iceberg files on the file system

You learn how to edit the existing default Hadoop SQL Storage Handler policy to access files. This policy is one of the two Ranger policies required to use Iceberg.

About this task

The Hadoop SQL Storage Handler policy allows references to Iceberg table storage location, which is required for creating or altering a table. You use a storage handler when you create a file stored as Iceberg on the file system or object store.

In this task, you specify Iceberg as the storage-type and allow the broadest access by setting the URL to *.

The Hadoop SQL Storage Handler policy supports only the RW Storage permission. A user having the required RW Storage permission on a resource, such as Iceberg, that you specify in the storage-type properties, is allowed only to reference the table location (for create/alter operations) in Iceberg. The RW Storage permission does not provide access to any table data. You need to create the Hadoop SQL policy described in the next topic in addition to this Hadoop SQL Storage Handler policy to access data in tables.

For more information about these policy settings, see Ranger Storage Handler documentation.

Procedure

1. Log into Ranger Admin Web UI.
   The Ranger Service Manager appears:

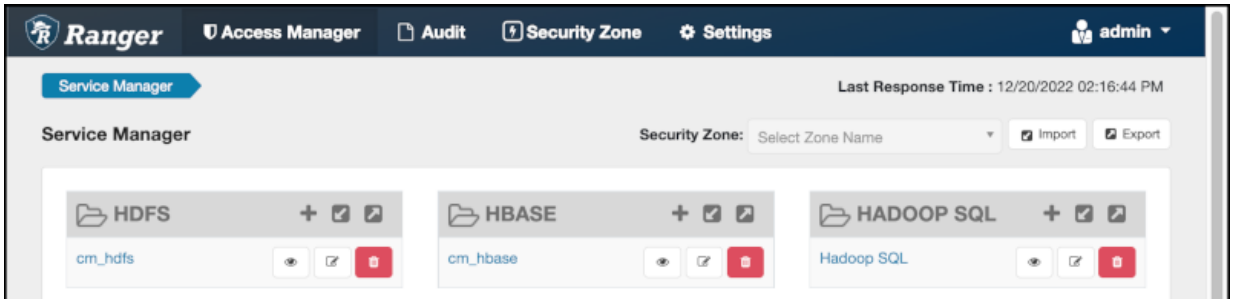

2. In Policy Name, enable the all - storage-type, storage-url policy.

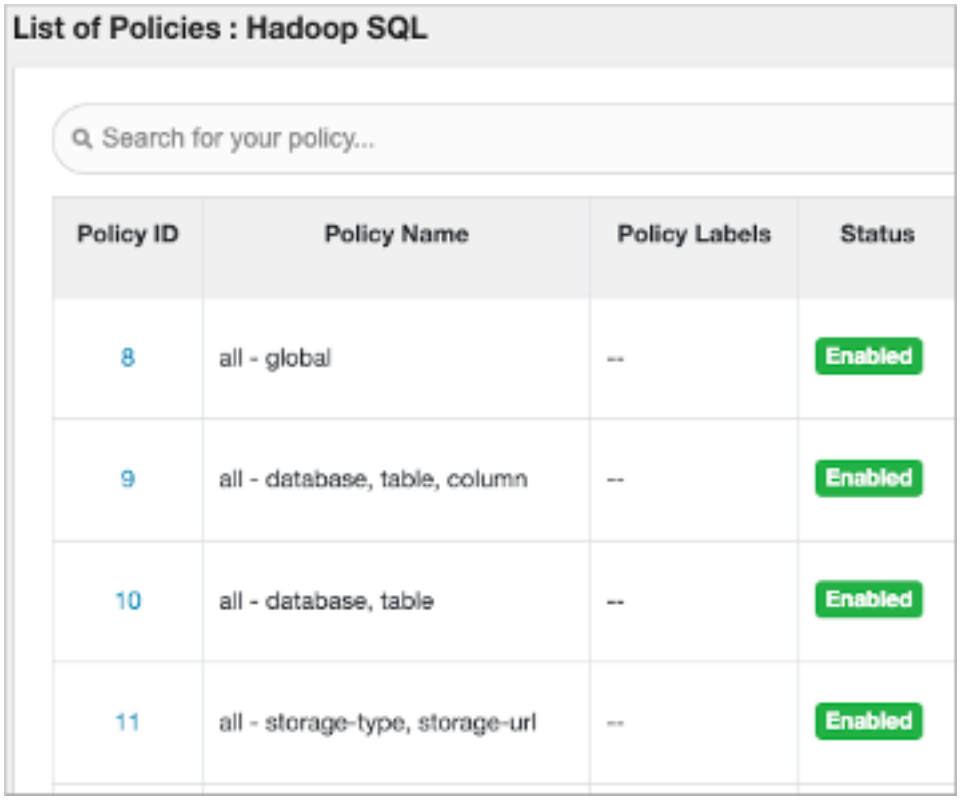

3. In Service Manager, in Hadoop SQL, select Edit and edit the all storage-type, storage-url policy.
4. Below Policy Label, select storage-type, and enter iceberg..

5. In Storage URL, enter the value *, enable Include.

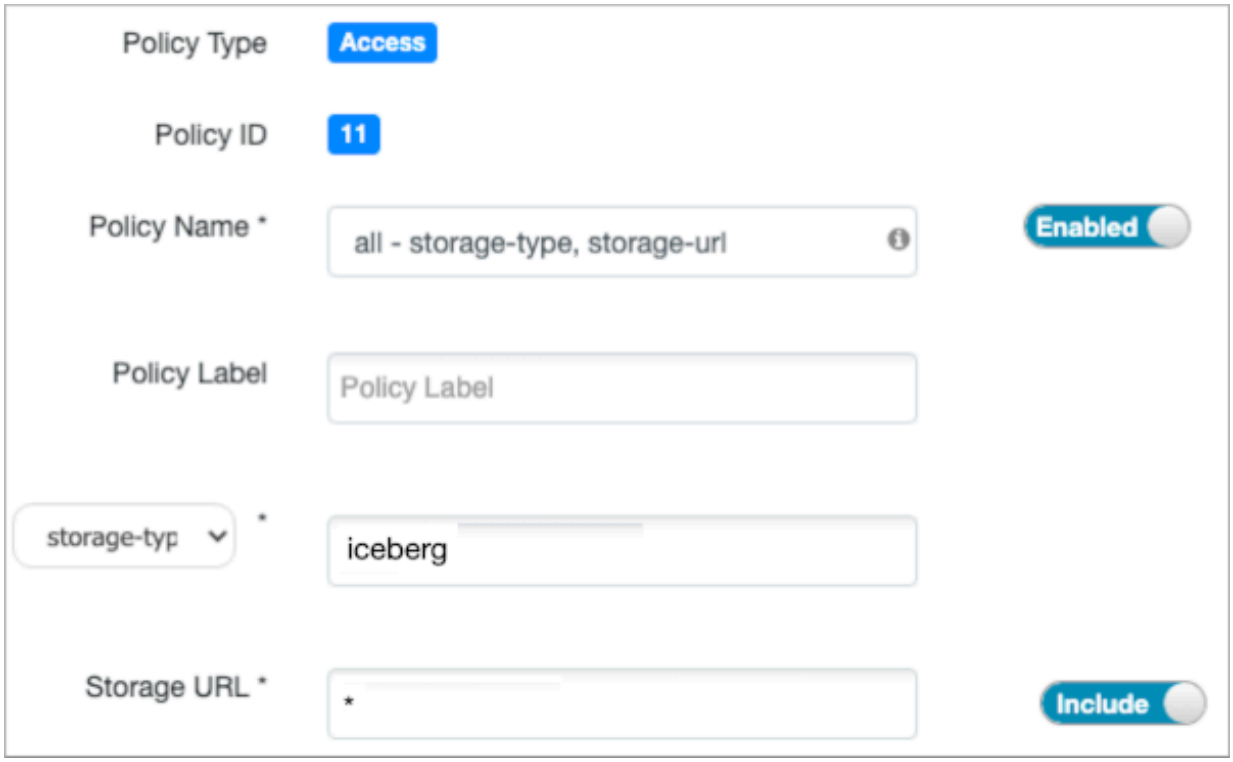

For more information about these policy settings, see Ranger storage handler documentation.

6. In Allow Conditions, specify roles, users, or groups to whom you want to grant RW storage permissions.

   You can specify PUBLIC to grant access to Iceberg tables permissions to all users. Alternatively, you can grant access to one user. For example, add the systest user to the list of users who can access Iceberg:

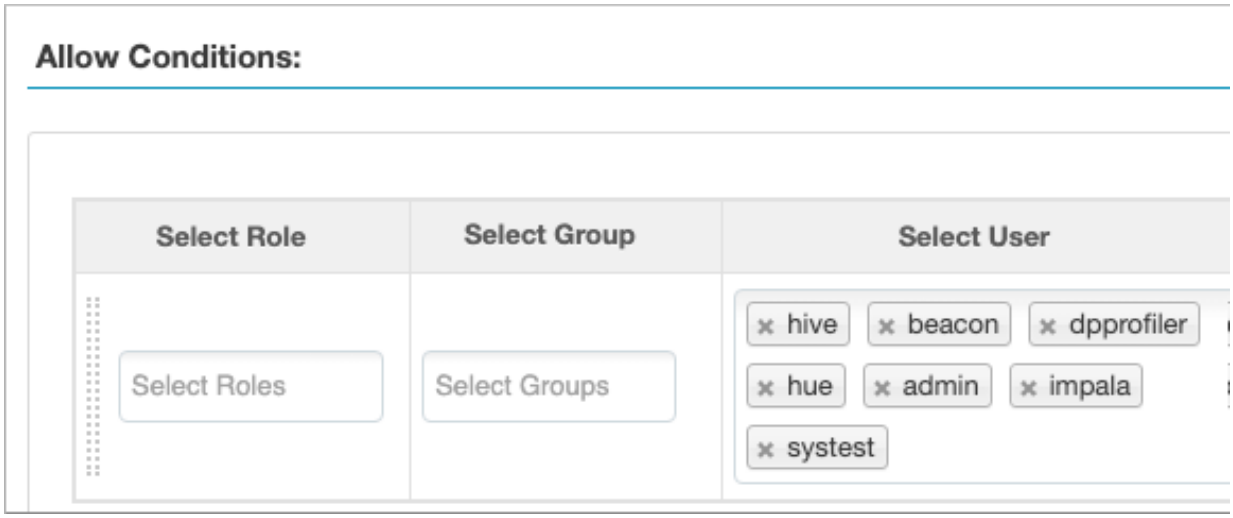

For more information about granting permissions, see Configure a resource-based policy: Hadoop-SQL.

7. Add the RW Storage permission to the policy.
8. Save your changes.

Creating a SQL policy to query an Iceberg table

You learn how to set up the second required policy for using Iceberg. This policy manages SQL query access to Iceberg tables.

About this task

You create a Hadoop SQL policy to allow roles, groups, or users to query an Iceberg table in a database. In this task, you see an example of just one of many ways to configure the policy conditions. You grant (allow) the selected roles, groups, or users the following add or edit permissions on the table: Select, Update, Create, Drop, Alter, and All. You can also deny permissions.

For more information about creating this policy, see Ranger documentation.

Procedure

1. Log into Ranger Admin Web UI.
   The Ranger Service Manager appears.
2. Click Add New Policy.
3. Fill in required fields.
   For example, enter the following required settings:
   - In Policy Name, enter the name of the policy, for example IcebergPolicy1.
   - In database, enter the name of the database controlled by this policy, for example icedb.
   - In table, enter the name of the table controlled by this policy, for example icetable.
   - In columns, enter the name of the column controlled by this policy, for example enter the wildcard asterisk (*) to allow access to all columns of icetable.
   - Accept defaults for other settings.

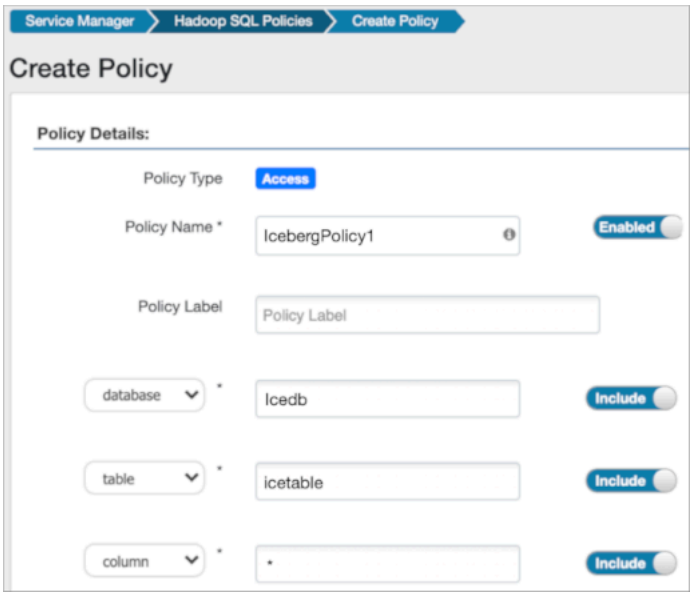

4. Scroll down to Allow Conditions, and select the roles, groups, or users you want to access the table.
   You can use Deny All Other Accesses to deny access to all other roles, groups, or users other than those specified in the allow conditions for the policy.

5. Select permissions to grant.
   For example, select Create, Select, and Alter. Alternatively, to provide the broadest permissions, select All.

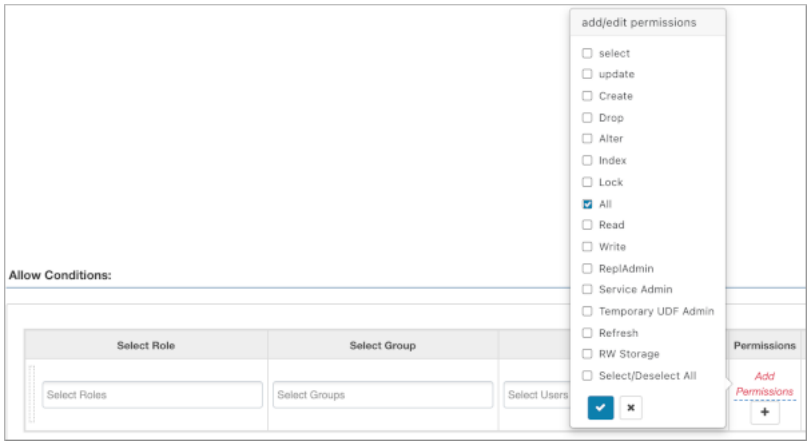

   Ignore RW Storage and other permissions not named after SQL queries. These are for future implementations.
6. Click Add.

Creating Virtual Cluster with Spark 3

Create a virtual cluster with Spark 3 as the Spark version.

For more information on creating virtual clusters, see Creating virtual clusters.

Creating a new Iceberg table from Spark 3

You can create an Iceberg table using Spark SQL.

Note: By default, Iceberg tables are created in the v1 format.

An example Spark SQL creation command to create a new Iceberg table is as follows:

```
spark.sql("""CREATE EXTERNAL TABLE ice_t (idx int, name string, state string
)
USING iceberg
PARTITIONED BY (state)""")
```

For information about creating tables, see the Iceberg documentation.

Creating an Iceberg table format v2

To use the Iceberg table format v2, set the format-version property to 2 as shown below:

```
CREATE TABLE logs (app string, lvl string, message string, event_ts timestam
p) USING iceberg TBLPROPERTIES ('format-version' = '2')
```

<delete-mode> <update-mode> and <merge-mode> can be specified during table creation for modes of the respective operation. If unspecified, they default to merge-on-read.

Unsupported Feature: CREATE TABLE ... LIKE

The CREATE TABLE ... LIKE feature is not supported in Spark:

```
CREATE TABLE <target> LIKE <source> USING iceberg
```

Here, `<source>` is an existing Iceberg table. This operation may appear to succeed and does not display errors and only warnings, but the resulting table is not a usable table.

Configuring Hive Metastore for Iceberg column changes

To make schema changes to an existing column of an Iceberg table, you must configure the Hive Metastore of the Data Lake.

Before you begin

Only set the following Hive Metastore configuration if you are using Data Lake version 7.2.14 or earlier. By default, Data Lake versions 7.2.15.0 and onwards has `metastore.allow.incompatible.col.type.changes.serdes` set to `org.apache.hadoop.hive.kudu.KuduSerDe,org.apache.iceberg.mr.hive.HiveIcebergSerDe` in Hive. For information about creating tables, see the Spark DDL documentation linked below.

Procedure

1. In Cloudera Manager, select the service for the Hive Metastore.
2. Click the Configuration tab.
3. Search for safety valve and find the Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml safety valve.
4. Add the following property:
   - Name: metastore.allow.incompatible.col.type.changes.serdes
   - Value: org.apache.hadoop.hive.kudu.KuduSerDe,org.apache.iceberg.mr.hive.HiveIcebergSerDe
5. Click Save Changes.
6. Restart the service to apply the configuration change.

Related Information

Spark DDL documentation

Importing and migrating Iceberg table in Spark 3

Importing or migrating tables are supported only on existing external Hive tables. When you import a table to Iceberg, the source and destination remain intact and independent. When you migrate a table, the existing Hive table is converted into an Iceberg table. You can use Spark SQL to import or migrate a Hive table to Iceberg.

Importing

Call the snapshot procedure to import a Hive table into Iceberg using a Spark 3 application.

```
spark.sql("CALL
<catalog>.system.snapshot('<src>', '<dest>')")
```

Definitions:

- <src> is the qualified name of the Hive table
- <dest> is the qualified name of the Iceberg table to be created
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.

For example:

```
spark.sql("CALL
spark_catalog.system.snapshot('hive_db.hive_tbl',
```

```
'iceberg_db.iceberg_tbl')")
```

For information on compiling Spark 3 application with Iceberg libraries, see Iceberg library dependencies for Spark applications linked below.

Migrating

When you migrate a Hive table to Iceberg, a backup of the table, named <table_name>_backup_, is created.

Ensure that the TRANSLATED_TO_EXTERNAL property, that is located in TBLPROPERTIES, is set to false before migrating the table. This ensures that a table backup is created by renaming the table in Hive metastore (HMS) instead of moving the physical location of the table. Moving the physical location of the table would entail copying files in Amazon s3.

We recommend that you refrain from dropping the backup table, as doing so will invalidate the newly migrated table.

If you want to delete the backup table, set the following:

```
'external.table.purge'='FALSE'
```

Note: For Cloudera Data Engineering 1.19 and above, the property will be set automatically.

Deleting the backup table in the manner above will prevent underlying data from being deleted, therefore, only the table will be deleted from the metastore.

To undo the migration, drop the migrated table and restore the Hive table from the backup table by renaming it.

Call the migrate procedure to migrate a Hive table to Iceberg.

```
spark.sql("CALL
<catalog>.system.migrate('<src>')")
```

Definitions:

- <src> is the qualified name of the Hive table
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.

For example:

```
spark.sql("CALL
spark_catalog.system.migrate('hive_db.hive_tbl')")
```

Related Concepts

Configuring Catalog

Related reference

Iceberg library dependencies for Spark applications

Importing and migrating Iceberg table format v2

Importing or migrating Hive tables Iceberg table formats v2 are supported only on existing external Hive tables. When you import a table to Iceberg, the source and destination remain intact and independent. When you migrate a table, the existing Hive table is converted into an Iceberg table. You can use Spark SQL to import or migrate a Hive table to Iceberg.

Importing

Call the snapshot procedure to import a Hive table into Iceberg table format v2 using a Spark 3 application.

```
spark.sql("CALL
<catalog>.system.snapshot(source_table => '<src>',
table => '<dest>',
properties => map('format-version', '2', 'write.delete.mode', '<delete-mo
de>',
'write.update.mode', '<update-mode>',
'write.merge.mode', '<merge-mode>'))")
```

Definitions:

- <src> is the qualified name of the Hive table
- <dest> is the qualified name of the Iceberg table to be created
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.
- <delete-mode> <update-mode> and <merge-mode> are the modes that shall be used to perform the respective operation. If unspecified, they default to 'merge-on-read'

For example:

```
spark.sql("CALL
spark_catalog.system.snapshot('hive_db.hive_tbl',
'iceberg_db.iceberg_tbl')")
```

For information on compiling Spark 3 application with Iceberg libraries, see Iceberg library dependencies for Spark applications linked below.

Migrating

Call the migrate procedure to migrate a Hive table to Iceberg.

```
spark.sql("CALL
<catalog>.system.migrate('<src>',
map('format-version', '2',
'write.delete.mode', '<delete-mode>',
'write.update.mode', '<update-mode>',
'write.merge.mode', '<merge-mode>'))")
```

Definitions:

- <src> is the qualified name of the Hive table
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.
- <delete-mode> <update-mode> and <merge-mode> are the modes that shall be used to perform the respective operation. If unspecified, they default to 'merge-on-read'

For example:

```
spark.sql("CALL
spark_catalog.system.migrate('hive_db.hive_tbl',
map('format-version', '2',
'write.delete.mode', 'merge-on-read',
'write.update.mode', 'merge-on-read',
'write.merge.mode', 'merge-on-read'))")
```

Upgrading Iceberg table format v1 to v2

To upgrade an Iceberg table format from v1 to v2, run an `ALTER TABLE` command as follows:

```
spark.sql("ALTER TABLE <table_name> SET TBLPROPERTIES('merge-on-read', '
2')")
```

<delete-mode>,<update-mode>, and <merge-mode> can be specified as the modes that shall be used to perform the respective operation. If unspecified, they default to 'merge-on-read'

Related Concepts

Configuring Catalog

Related reference

Iceberg library dependencies for Spark applications

Configuring Catalog

When using Spark SQL to query an Iceberg table from Spark, you refer to a table using the following dot notation:

```
<catalog_name>.<database_name>.<table_name>
```

The default catalog used by Spark is named spark_catalog. When referring to a table in a database known to spark_catalog, you can omit <catalog_name>. .

Iceberg provides a SparkCatalog property that understands Iceberg tables, and a SparkSessionCatalog property that understands both Iceberg and non-Iceberg tables. The following are configured by default:

```
spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
spark.sql.catalog.spark_catalog.type=hive
```

This replaces Spark's default catalog by Iceberg's SparkSessionCatalog and allows you to use both Iceberg and non-Iceberg tables out of the box.

There is one caveat when using SparkSessionCatalog. Iceberg supports CREATE TABLE … AS SELECT (CTAS) and REPLACE TABLE … AS SELECT (RTAS) as atomic operations when using SparkCatalog. Whereas, the CTAS and RTAS are supported but are not atomic when using SparkSessionCatalog. As a workaround, you can configure another catalog that uses SparkCatalog. For example, to create the catalog named iceberg_catalog, set the following:

```
spark.sql.catalog.iceberg_catalog=org.apache.iceberg.spark.SparkCatalog
spark.sql.catalog.iceberg_catalog.type=hive
```

You can configure more than one catalog in the same Spark job. For more information, see the *Iceberg documentation*.

Related Information

Iceberg documentation

Loading data into an unpartitioned table

You can insert data into an unpartitioned table. The syntax to load data into an iceberg table:

```
INSERT INTO table_identifier [ ( column_list ) ]
    VALUES ( { value | NULL } [ , ... ] ) [ , ( ... ) ]
```

Or

```
INSERT INTO table_identifier [ ( column_list ) ]
```

```
    query
```

Example:

```
INSERT INTO students VALUES
    ('Amy Smith', '123 Park Ave, San Jose', 111111)
INSERT INTO students VALUES
    ('Bob Brown', '456 Taylor St, Cupertino', 222222),
    ('Cathy Johnson', '789 Race Ave, Palo Alto', 333333)
```

Querying data in an Iceberg table

To read the Iceberg table, you can use SparkSQL to query the Iceberg tables.

Example:

```
spark.sql("select * from ice_t").show(1000, false)
```

Updating Iceberg table data

Iceberg table data can be updated using copy-on-write or merge-on-read. The table version you are using will determine how you can update the table data.

v1 format

Iceberg supports bulk updates through MERGE, by defaulting to copy-on-write deletes when using v1 table format.

v2 format

Iceberg table format v2 supports efficient row-level updates and delete operations leveraging merge-on-read.

For more details, refer to *Position Delete Files* linked below.

For updating data examples, see *Spark Writes* linked below.

Related Information

Position Delete Fies

Spark Writes

Iceberg library dependencies for Spark applications

If your Spark application only uses Spark SQL to create, read, or write Iceberg tables, and does not use any Iceberg APIs, you do not need to build it against any Iceberg dependencies. The runtime dependencies needed for Spark to use Iceberg are in the Spark classpath by default. If your code uses Iceberg APIs, then you need to build it against Iceberg dependencies.

Cloudera publishes Iceberg artifacts to a Maven repository with versions matching the Iceberg in Cloudera Data Engineering.

Note: For CDH-7.1.x, there are no iceberg jars in the maven repository. Use 0.14.1.1.17.7215.0-27 iceberg version for compilation. The below iceberg dependencies should only be used for compilation. Including iceberg jars within a Spark application fat jar must be avoided.

```
<dependency>
     <groupId>org.apache.iceberg</groupId>
     <artifactId>iceberg-core</artifactId>
```

```
     <version>${iceberg.version}</version>
     <scope>provided</scope>
</dependency>
<!-- for org.apache.iceberg.hive.HiveCatalog -->
<dependency>
     <groupId>org.apache.iceberg</groupId>
     <artifactId>iceberg-hive-metastore</artifactId>
     <version>${iceberg.version}</version>
     <scope>provided</scope>
</dependency>
<!-- for org.apache.iceberg.spark.* classes if used -->
<dependency>
    <groupId>org.apache.iceberg</groupId>
    <artifactId>iceberg-spark</artifactId>
    <version>${iceberg.version}</version>
    <scope>provided</scope>
</dependency>
```

Alternatively, the following dependency can be used:

```
<dependency>
   <groupId>org.apache.iceberg</groupId>
   <artifactId>iceberg-spark3-runtime</artifactId>
   <version>${iceberg.version}</version>
   <scope>provided</scope>
</dependency>
```

The iceberg-spark3-runtime JAR contains the necessary Iceberg classes for Spark runtime support, and includes the classes from the dependencies above.

After compiling the job, you can create and run Cloudera Data Engineering jobs. For more information see, Creating Spark jobs and Running a Spark job.

Creating a Git repository in Cloudera Data Engineering (Technical Preview)

Git repositories allow teams to collaborate, manage project artifacts, and promote applications from lower to higher environments. Cloudera currently supports Git providers such as GitHub, GitLab, and Bitbucket. Learn how to use Cloudera Data Engineering with version control service.

About this task

Repository files can be accessed when you create a Spark or Airflow job. You can then deploy the job and use Cloudera Data Engineering's centralized monitoring and troubleshooting capabilities to tune and adjust your workloads. Cloudera Data Engineering automatically clones the project files and folders when a repository is created. Metadata such as file size and hash are also available. These files display as a read-only view in the Cloudera Data Engineering UI and users cannot delete or modify the files. This ensures a single source of truth and simplifies promotions.

Supported version control service providers: Cloudera currently supports the following version control service providers:

- GitHub
- GitLab
- Bitbucket

Before you begin

To use a non-public Git repository, you must first create repository credentials using a workload secret for Cloudera Data Engineering using the CDE CLI as follows:

```
cde credential create --type basic --username myuser --name my-credential
```

The command above prompts you for a password where you can either provide your Personal Access Token (PAT) or provide a password for your Git repository account, for example, Github.

Note: When using the password based authentication, make sure your two factor authentication (2FA) for your Github account is turned off. Cloudera recommends turning it off because the 2FA doesn't get the authentication requests when the source is an API.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Home page displays.
2. Click Repositories in the left navigation menu.The Repositories page displays.

3. Click Create Repository. The Create A Repository dialog box displays. Enter the following fields for the repository:

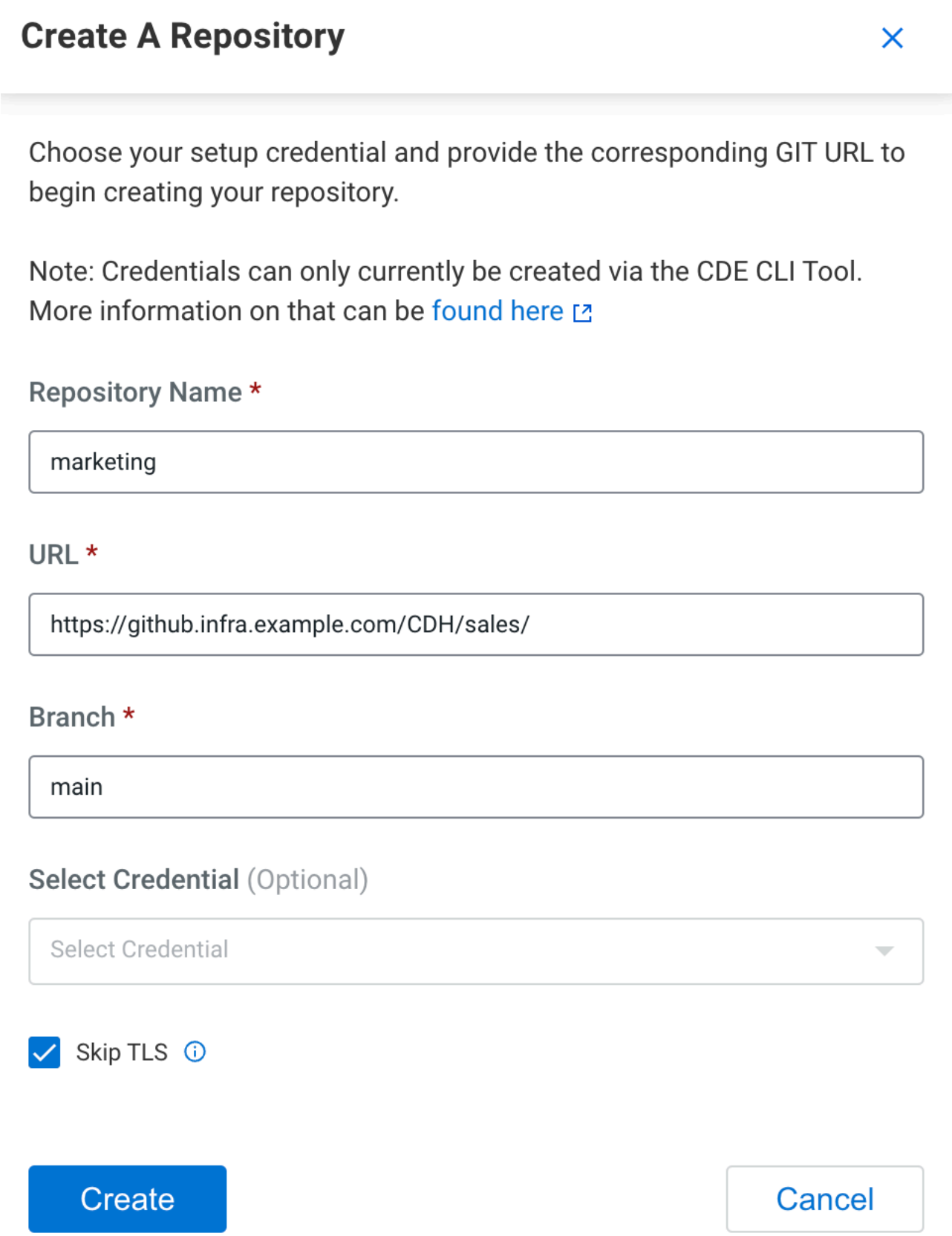

a) Repository Name - Enter a name for the repository.
b) URL - Enter the repository URL (https only).
c) Branch - Enter the name of the git branch.
d) Select a credential from the Select Credential drop-down list. The credentials can be created using the Cloudera Data Engineering CLI.
e) Select Skip TLS. Select this option if the server uses a self-signed CA certificate that Cloudera Data Engineering does not trust. This allows Cloudera Data Engineering to skip the security check and clone the repository.

4. Click Create.

Managing jobs in Cloudera Data Engineering

It is often necessary to modify your Cloudera Data Engineering jobs. Cloudera Data Engineering makes it easy to modify most aspects of your jobs, including replacing the application code and any supplemental files, as well as modifying configuration parameters and the schedule.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also manage a job by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. To view Cloudera Data Engineering Services, click Administration on the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. In the Cloudera Cloudera Management Console, click the Data Engineering tile. The Cloudera Data Engineering Overview page displays.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster that your application is associated with.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the application you want to manage.
4. Select Jobs from the left hand menu.
5. Click on the name of the job you want to modify.
6. The Run History tab lists the recent job executions for the application. Click the Configuration tab to display the job configuration.
7. Click the Configure button to edit the application configuration.
8. Edit the configuration parameters you want to change, including uploading a modified JAR or Python file if necessary.
9. Click Advanced Configuration to see additional parameters and the job schedule. Make any necessary changes, and then click Update.

Running Jobs in Cloudera Data Engineering

Jobs in Cloudera Data Engineering can be run on demand, or scheduled to run on an ongoing basis. The following instructions demonstrate how to run a job in Cloudera Data Engineering.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also manage a job by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. To view Cloudera Data Engineering Services, click Administration on the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster where you want to run the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to run.
4. To run a job immediately, click ⋮ in the Actions column next to the job, and then click Run Now.

   You can cancel a running job by clicking Cancel in the same Actions menu.

   Job Run Notices:The running jobs provide notifications, in the form of a Bell icon next to the job Run ID, when certain conditions are met, without having to parse low level logs, or navigating away to a cloud provider or Kubernetes interface. This will help you to identify why certain job is running slow or stuck, and take actions to rectify this.

Scheduling jobs in Cloudera Data Engineering

Jobs in Cloudera Data Engineering can be run on demand, or scheduled to run on an ongoing basis. The following instructions demonstrate how to create or modify a schedule for an existing job.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also schedule a job from the new Home page by clicking Schedule a Job, or by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. To view Cloudera Data Engineering Services, click Administration on the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. In the Cloudera console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster where you want to schedule the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to schedule.
4. Click the Configure.
5. Click the Advanced Configurations link at the bottom of the page to view additional configuration parameters.
6. Click the Actions menu next to the application, and then click Configuration.

7. Select the Schedule toggle, and then set the Start time, End time, and Cron expression.

   The start and end times designate the time frame for which the schedule is active. The Cron expression uses the cron scheduling syntax to specify when the application should run within the start and end times. For information and examples of the cron syntax, see the Cron entry on Wikipedia.

   Note: Timestamps must be specified in ISO-8601 UTC format ('yyyy-MM-ddTHH:mm:ssZ'). UTC offsets are not supported.

   Note: Scheduled job runs start at the end of the first full schedule interval after the start date, at the end of the scheduled period. For example, if you schedule a job with a daily interval with a start_date of 14:00, the first scheduled run is triggered at the end of the next day, after 23:59:59. However if the start_date is set to 00:00, it is triggered at the end of the same day, after 23:59:59.
8. If you want to start a job immediately, check the Start job box.
9. Click Update to save your changes.
10. Select optional scheduling configurations:
    a) Select Enable Catchup to kick off job runs for any data interval that has not been run since the last data interval. If this option is not selected, only the runs that start after the time that the job was created will be included.
    b) Select Depends on Previous to ensure that each job run is preceeded by a successful job run.
11. Click Schedule.

Creating an ad-hoc job in Cloudera Data Engineering

Ad-hoc runs mimic the behavior of the traditional spark-submit or a single execution of an Airflow DAG, where the job runs once. These runs will not establish a permanent job definition. You can use the ad-hoc job runs for log analysis and future reference.

Before you begin

- Ensure that you have a Virtual Cluster that is ready to use.

For Spark jobs

1. In the Cloudera console, click the Data Engineering tile. The Home page displays.
2. Click See More under Deploy and select Ad-Hoc Run. The Create an Ad-Hoc Spark Job dialog box is displayed.
3. Select a Virtual Cluster.
4. Enter a Job Name.
5. In the Select Application Files drop-down list, select Resource and upload or select the resource or URL and enter the URL that contains the file or Repository and select a file from the repository and the file will be automatically added to the job.
6. Enter a Main Class.
7. Enter Arguments and Configurations.
8. Select a Python Environment.
9. Select a Data Connector.

Steps for advanced options

You can upload additional files, customize the number of executors, drivers, executor cores, and memory.

1. Upload files and resources.
2. Configure Compute options.
3. Set an option for Log Level.
4. Click Enable GPU Accelerations checkbox to enable the GPU acceleration and configure selectors and tolerations if you want to run the job on specific GPU nodes.

5. Click Create and Run.

For Airflow jobs

1. In the Cloudera console, click the Data Engineering tile. The Home page displays.
2. In the Jobs section under Airflow, and click Ad-hoc Run.
3. Select a Virtual Cluster.
4. Enter a Job Name.
5. Upload a DAG file.
6. Click Create and Run.

Configuring logging in Cloudera Data Engineering jobs

You can configure logging in your Cloudera Data Engineering jobs using the Cloudera Data Engineering UI. You can modify log levels, and add or replace customized log4j.properties files.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also view jobs by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. To view Cloudera Data Engineering Services, click Administration on the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. In the Cloudera Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster that your application is associated with.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the application you want to manage.
4. Select Jobs from the left hand menu.
5. Click on the name of the job you want to modify.
6. The Run History tab lists the recent job executions for the application. Click the Configuration tab to display the job configuration.
7. Click Edit to open the job configuration for editing.
8. Click Add Configuration.
   - Enter extraJavaOptions in the config_key field.
   - Enter '-Dlog4j.configuration=file:/app/mount/*[***LOG4J.PROPERTIES FILE***]*' in the config_value field, replacing *[***LOG4J.PROPERTIES FILE***]* with the name of the custom log4j.properties file you want to reference.

   For example:

   ```
   '-Dlog4j.configuration=file:/app/mount/mylog4j.properties'
   ```
9. Open Advanced Options to see additional parameters.
10. Under Other Dependencies click Upload.

    Browse for the custom log4j.properties file you want to upload, then click Open.

11. Scroll down to set Log Level.
    Select a value from the drop-down list.
12. Click Update.

Deleting Jobs in Cloudera Data Engineering

If you no longer need a job, you can delete it. Deleting a job does not delete the job run history.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also view jobs by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. To view Cloudera Data Engineering Services, click Administration on the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster where you want to delete the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to delete.

Best practices for building Apache Spark applications

Follow these best practices when building Apache Spark Scala and Java applications:

- Refrain from using withColumn in chain, loop, and calling it multiple times in a single query. Doing so may cause performance issues. To avoid issues, use select() with multiple columns at once. See the Apache Spark API reference linked below for more information.
- Compile your applications against the same version of Spark that you are running.
- Build a single assembly JAR ("Uber" JAR) that includes all dependencies. In Maven, add the Maven assembly plug-in to build a JAR containing all dependencies:

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <configuration>
    <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
    </descriptorRefs>
  </configuration>
  <executions>
    <execution>
      <id>make-assembly</id>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
    </execution>
  </executions>
```

```
</plugin>
```

This plug-in manages the merge procedure for all available JAR files during the build. Exclude Spark, Hadoop, and Kafka classes from the assembly JAR, because they are already available on the cluster and contained in the runtime classpath. In Maven, specify Spark, Hadoop, and Kafka dependencies with scope provided. For example:

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.11</artifactId>
  <version>2.4.0.7.0.0.0</version>
  <scope>provided</scope>
</dependency>
```

Cloudera Data Engineering 1.23.0

Orchestrating workflows and pipelines with Apache Airflow in Cloudera Data Engineering

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Airflow in Cloudera Data Engineering..... 5

Creating and managing Airflow jobs using Cloudera Data Engineering.....6
Creating Airflow jobs using Cloudera Data Engineering.....6
Creating an Airflow DAG using the Pipeline UI.....6
Running Airflow jobs using Cloudera Data Engineering.....7
Deleting Airflow jobs using Cloudera Data Engineering.....7

Managing an Airflow Pipeline using the CDE CLI..... 8
Creating a pipeline using the CDE CLI..... 8
Creating a basic Airflow pipeline using CDE CLI.....8
Creating a pipeline with additional Airflow configurations using CDE CLI..... 8
Creating an Airflow pipeline with custom files using CDE CLI [Technical Preview].....10
Updating a pipeline using the CDE CLI..... 11
Updating a DAG file using the CDE CLI..... 12
Updating the Airflow job configurations using the CDE CLI.....12
Updating the Airflow file mounts using the CDE CLI [Technical Preview].....12
Deleting an Airflow pipeline using the CDE CLI.....13

Creating and managing Airflow connections..... 13
Creating a connection to Cloudera Data Warehouse or Cloudera Data Hub instance for SQL Operator.....13
(Deprecated) Creating a connection to Cloudera Data Warehouse for Cloudera Data Warehouse Operator..... 15
Creating a connection to run jobs on other Cloudera Data Engineering Virtual Clusters..... 16

Executing SQL queries on Cloudera Data Warehouse or Cloudera Data Hub instance using Apache Airflow in Cloudera Data Engineering..... 17

Running Airflow jobs on other Cloudera Data Engineering Virtual Clusters.....19

Using Cloudera Data Engineering with an external Apache Airflow deployment.....20

Supported Airflow operators and hooks.....22

Using custom operators and libraries for Apache Airflow.....22
Adding custom operators and libraries..... 22
Updating custom operators and libraries..... 23

Deleting custom operators and libraries....24
Troubleshooting custom operators and libraries....24
Viewing logs for custom operators and libraries....24

Apache Airflow in Cloudera Data Engineering

Learn about how Apache Airflow is integrated with Cloudera Data Engineering and how to automate a workflow or data pipeline using Apache Airflow Python DAG files in Cloudera Data Engineering.

Cloudera Data Engineering enables you to automate a workflow or data pipeline using Apache Airflow Python DAG files. Each Cloudera Data Engineering Virtual Cluster includes an embedded instance of Apache Airflow. You can also use Cloudera Data Engineering with your own Airflow deployment. For more information about using your own Cloudera Data Engineering Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

Cloudera Data Engineering currently supports multiple Airflow operators. For example, one for running Cloudera Data Engineering jobs, or one for accessing and executing SQL commands on Cloudera Data Warehouse. For more information about the complete list of installed and supported operators, see Supported Airflow operators and hooks.

You can create and manage Apache Airflow jobs by writing or creating Python DAG files and uploading them using the UI. For more information about Cloudera Data Engineering Airflow job management, see Creating and managing Cloudera Data Engineering Airflow Jobs using the Cloudera Data Engineering UI.

Cloudera Data Engineering provides various ways to connect to Cloudera Data Warehouses or to other Cloudera Data Engineering Virtual Clusters. You must create an Airflow connection containing the target system access details and then refer it in the associated operators to execute the required workloads.

- Creating and managing Airflow connections: To connect to a Cloudera Data Warehouse or Cloudera Data Engineering Virtual Cluster, you must create an Airflow connection.
- Executing SQL queries on Cloudera Data Warehouse: To execute SQL queries on Hive or Impala Virtual Warehouses, you can use the installed SQLExecuteQueryOperator or the CdwExecuteQueryOperator.
- Running jobs on other Cloudera Data Engineering Virtual Clusters: With an existing Cloudera Data Engineering connection, you can use the CDERunJobOperator to execute jobs on other Cloudera Data Engineering Virtual clusters.

You can also install and use custom operators and libraries (Python packages) for Airflow with Cloudera Data Engineering. Cloudera provides a way to extend the installed default packages with the third party or custom Python packages using the Custom Operators and Libraries feature using the Cloudera Data Engineering user interface (UI).

Important:

- Cloudera supports certain Airflow core operators and hooks but does not provide support for any custom extension through provider packages. For more information, see *Supporting Airflow operators and hooks*. Cloudera Support may require you to remove any installed provider packages during troubleshooting. For more information, see *Provider packages*.
- Direct access to the native Airflow API is not supported because bypassing the Cloudera Data Engineering API prevents job tracking, centralized monitoring, and metadata tracking.

Related reference

Supported Airflow operators and hooks

Related Information

Provider packages

Enabling SSO to a Virtual Warehouse

Creating jobs in Cloudera Data Engineering

Using Cloudera Data Engineering with an external Apache Airflow deployment

Creating and managing Airflow jobs using Cloudera Data Engineering

Learn about how to create and manage the Apache Airflow jobs using Cloudera Data Engineering.

Creating Airflow jobs using Cloudera Data Engineering

Learn about how to create Airflow jobs using Cloudera Data Engineering.

About this task

An Airflow job in Cloudera Data Engineering consists of an Airflow DAG file and various optional resources. Jobs can be run on demand or scheduled.

Procedure

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the service that contains the virtual cluster that you want to create a job for.
3. In the Virtual Clusters column, locate the virtual cluster that you want to use and click the View Jobs icon.
4. In the left navigation menu, click Jobs.
5. Click the Create Job button.
6. Provide the Job Details:
   a) Select Airflow for the job type. The available fields on the user interface updates automatically.
   b) Specify the Name.
   c) Click the File option and select the way you want to provide the DAG file. Using the File option, you can do the following:
      - Upload the DAG file to a new resource.
      - Use the DAG file from a previously created resource.
      - Upload any other resources required for the job.

      For more information about the Editor, see Creating an Airflow DAG using the Pipeline UI.
7. If you do not want to run the job immediately, click the Create and Run drop-down menu and then click Create. Otherwise, click Create and Run to run the job immediately.

   Note: If you select the Create and Run option from the Create and Run drop-down menu, the DAG file must not be paused while creating and provide a False value to the is_paused_upon_creation field.

Creating an Airflow DAG using the Pipeline UI

With the Cloudera Data Engineering Pipeline UI, you can create multi-step pipelines with a combination of available operators.

About this task

Note: Cloudera supports all major browsers (Google Chrome, Firefox and Safari) for this feature. If you are using a browser in incognito mode, you have to allow all cookies in your browser settings so that you can view Pipelines, Spark, and Airflow pages.

Procedure

1. Go to Jobs Create Job .

   Under Job details, select Airflow.

   The UI refreshes, only Airflow-specific options remain.
2. Specify a name for the job.
3. Under DAG File select the Editor option.
4. Click Create.
   You are redirected to the job Editor tab.
5. Build your Airflow pipeline.
   - Drag and drop operators to the canvas from the left hand pane.
   - When selecting an operator, you can configure it in the editor pane that opens up.

     On the Configure tab you can provide operator-specific settings. The Advanced tab allows you to make generic settings that are common to all operators, for example execution timeout or retries.
   - Create dependencies between tasks by selecting them and drawing an arrow from one of the four nodes on their edges to another task. If the dependency is valid the task is highlighted in green. If invalid, it is highlighted in red.
   - To modify DAG-level configuration, select Configurations on the upper right.
6. When you are done with building your pipeline, click Save.

Running Airflow jobs using Cloudera Data Engineering

Jobs in Cloudera Data Engineering can be run on demand, or scheduled to run on an ongoing basis. Learn about how to run a job in Cloudera Data Engineering.

Procedure

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster where you want to run the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to run.
4. To run a job immediately, click ⋮ in the Actions column next to the job, and then click Run Now. The job must not be paused. If the Job is paused, there is an option named Resume Schedule in the menu, which allows you unpause it.

   Job Run Notices: The running jobs provide notifications, in the form of a Bell icon next to the Job Run ID, when certain conditions are met, without having to parse low level logs, or navigating away to a cloud provider or Kubernetes interface. This allows you to identify why certain job is running slow or stuck, and take actions to rectify this.

Deleting Airflow jobs using Cloudera Data Engineering

If you no longer need a job, you can delete it. Deleting a job does not delete the job run history.

Before you begin

An Airflow job can't be deleted if it has active runs. Before deleting a job, make sure that every run is either completed or killed.

Procedure

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster where you want to delete the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to delete.
4. Click ⋮ in the Actions column next to the job, and then click Delete.

Managing an Airflow Pipeline using the CDE CLI

Based on your business requirement, you can use Cloudera Data Engineering CLI to create basic Airflow pipelines or multi-step pipelines with a combination of available operators, to enable data-driven decisions. You can update these Airflow pipelines by updating the DAG files and job configurations.

Creating a pipeline using the CDE CLI

You can update the following properties in an Airflow pipeline:

Related Information

Using the Cloudera Data Engineering CLI

Creating a basic Airflow pipeline using CDE CLI

By creating a basic pipeline in Cloudera Data Engineering using the CLI, you can create multi-step pipelines with a combination of available operators.

About this task

To create a basic pipeline in Cloudera Data Engineering, you must upload the Airflow (Directed Acyclic Graph) DAG to a Cloudera Data Engineering resource and create a Cloudera Data Engineering Airflow job from this DAG.

Procedure

In the CDE CLI, run the following command:

```
cde resource create --name my_pipeline_resource
cde resource upload --name my_pipeline_resource --local-path my_pipeline_dag
.py
cde job create --name my_pipeline --type airflow --dag-file my_pipeline_dag.
py --mount-1-resource my_pipeline_resource
```

Related Information

Using the Cloudera Data Engineering CLI

Creating a pipeline with additional Airflow configurations using CDE CLI

By creating a pipeline with additional Airflow configurations using the Cloudera Data Engineering CLI, you can create multi-step pipelines with a combination of available operators. There are two ways to create this type of pipeline. The first method detailed below is recommended approach that we highly suggest customers use. The second is the alternative method that customers have used in the past, but is not recommended.

About this task

Airflow DAGs can be defined with parameters at the DAG-level or Task-level. These parameters can be overridden in the case of a manual run. A manual run is triggered explicitly by the user. It is recommended to use the Params approach so that default values can be used by the scheduled job instances as well.

For Params (Recommended)

An example of a DAG definition with additional Airflow configuration is as follows:

1. Create a configuration such as the example shown below:

```
from airflow import DAG
from airflow.models.param import Param

with DAG(
    "my_dag",
    params={
        # an int with a default value
        "int_param": Param(10, type="integer", minimum=0, maximum
=20),

        # a required param which can be of multiple types
        # a param must have a default value
        "dummy": Param(5, type=["null", "number", "string"]),

        # an enum param, must be one of three values
        "enum_param": Param("foo", enum=["foo", "bar", 42]),

        # a param which uses json-schema formatting
        "email": Param(
            default="example@example.com",
            type="string",
            format="idn-email",
            minLength=5,
            maxLength=255,
        ),
    },
):
  # prints <class 'str'> by default
  # prints <class 'int'> if render_template_as_native_obj=True
  my_operator = PythonOperator(
      task_id="template_type",
      op_args=[
          "{{ params.int_param }}",
      ],
      python_callable=(
          lambda x: print(type(x))
      ),
  )
```

In this case, nothing needs to be done on the cde job create step. Values can be additionally overridden in a manual run, through the --config flag of the cde job run command. For example:

```
cde job run --name my_pipeline --config key1=my_new_value1
```

For Dag run conf (Not recommended)

Note: This is an alternative method to create a pipeline with additional Airflow configurations, but it is not recommended. This method does not work with scheduled job instances.

For historical reasons Cloudera Data Engineering supports the {{ dag_run.conf }} object as well. In this case, the option, --config key=value in the cde job create command, is used to define default values whenever the user triggers a manual run using cde job run without specifying these parameters in the run command. This config option can be repeated to define multiple parameters.

1. Create a configuration such as the example shown below:

```
cde resource create --name my_pipeline_resource
cde resource upload --name my_pipeline_resource --local-path my_pipeline
_dag.py
cde job create --name my_pipeline --type airflow --dag-file my_pipeline_
dag.py --mount-1-resource my_pipeline_resource -–config key1=value1 —-co
nfig key2=value2
```

The configuration can be used in a DAG as shown below:

```
my_bash_task = BashOperator(
 task_id="my_bash_task",
 bash_command="echo key1_value: {{ dag_run.conf[‘key1’] }} key2_value:
 {{ dag_run.conf[‘key2’] }}",
 dag=dag,
)
```

The configuration can also be overridden for manual runs in the same manner as described in the Recommended section on this page.

Related Information

Params

Using the Cloudera Data Engineering CLI

Creating an Airflow pipeline with custom files using CDE CLI [Technical Preview]

By creating a pipeline in Cloudera Data Engineering using the CLI, you can add custom files that are available for tasks. This is a technical preview.

Before you begin

This feature is available in Cloudera Data Engineering 1.19 and above in new Virtual Cluster installations only.

About this task

For use cases where custom files need to be accessed within an Airflow task, you need to first upload the custom files to a Cloudera Data Engineering resource, and then specify it in the job creation parameter using the --airflow-file-mount-<n>-resource option. These files are available only to the jobs in which they are linked.

The general form of the command is:

```
cde job create \
  --name <my_job_name> \
  --type airflow \
  --mount-1-resource <my_dag_resource> \
  --dag-file <my_dag_file.py> \
  --airflow-file-mount-n-resource <my_file_resource> \
  --airflow-file-mount-n-prefix <my_custom_prefix>   # Optional
```

In the --airflow-file-mount-n-resource parameter, n is an integer number (beginning at 1). This allows you to specify multiple ...-resource parameters, to mount multiple resources.

Each resource is mounted at /app/mount/<prefix>. If you do not need to specify a custom prefix, the mount point of your resource will be based on the resource name. For example, if the name of your Cloudera Data Engineering

resource is 'my_resource', the files in the resource will be made available within Airflow under /app/mount/my_resource.

If you do want to specify a custom prefix for your resource's mount point, use the optional --airflow-file-mount-n-prefix parameter, specifying n as the same number as the corresponding --airflow-file-mount-n-resource parameter.

Note: Note that all resource mounts to Airflow are read-only.

Procedure

Run the following commands to upload the custom files to a Cloudera Data Engineering resource, and then create the job:

```
cde resource create --name my_pipeline_resource
cde resource upload --name my_pipeline_resource --local-path my_pipeline_dag
.py
cde resource create --name my_file_resource
cde resource upload --name my_file_resource --local-path my_file.conf

cde job create --name my_pipeline --type airflow --dag-file my_pipeline_dag.
py --mount-1-resource my_pipeline_resource --airflow-file-mount-1-resource m
y_file_resource
```

Example

The files can be reached in Airflow DAGs with the following pattern: /app/mount/<resource_name or resource_alias>/<file_name>, like in the following example:

```
read_conf = BashOperator(
     task_id=read_conf,
     bash_command="cat /app/mount/my_file_resource/my_file.conf"
 )
```

Note: It is possible to change the mount path by specifying the --airflow-file-mount-N-prefix my_custom_prefix option in the job creation command, like in the following example:

```
cde job create --name my_pipeline --type airflow --dag-file my_pipeline_dag.
py --mount-1-resource my_pipeline_resource --airflow-file-mount-1-resource m
y_file_resource --airflow-file-mount-1-prefix my_custom_prefix
```

In this case, the file is available at:

```
read_conf = BashOperator(
     task_id=read_conf,
     bash_command="cat /app/mount/my_custom_prefix/my_file.conf"
 )
```

Note: As a best practice, Cloudera recommends to use the same resource name because it is simpler to follow the DAG without having to look at the job definition. Also, it is possible to use "/" as a value to mount to /app/mount if there is only one Airflow file mounted in the job definition; however, this is not recommended.

Related Information

Using the Cloudera Data Engineering CLI

Updating a pipeline using the CDE CLI

You can update the following properties in an Airflow pipeline:

Updating a DAG file using the CDE CLI

You can update a Directed Acyclic Graph (DAG) file using the CDE CLI for instances where the DAG needs to be overridden. For use cases where the DAG needs to be overridden, first the DAG needs to be uploaded to the resource to override the previous version, then you must update the job.

About this task

Unlike a Spark job, the Airflow job does not automatically pull in the updated resource. Airflow jobs require a forced update by calling the `job update` command, such that the required files are uploaded to Airflow server for processing.

Choose one of the following options in step 1:

Important: Before updating a DAG, it is recommended to ensure the DAG is paused and is not currently running when the update is being done.

Updating a DAG file that needs to be overridden

Run the following command in the CDE CLI:

```
cde resource upload --name my_pipeline_resource --local-path my_pipeline_dag
.py
cde job update --name my_pipeline --dag-file my_pipeline_dag.py --mount-1-r
esource my_pipeline_resource
```

Updating a DAG in scenarios where the DAG must be set to a different one:

First upload the DAG to any resource and then the job needs to be updated. Run the following command in the CDE CLI:

```
cde resource upload --name my_other_pipeline_resource --local-path my_other_
pipeline_dag.py
cde job update --name my_pipeline --dag-file my_other_pipeline_dag.py --mou
nt-1-resource my_other_pipeline_resource
```

Updating the Airflow job configurations using the CDE CLI

In the case where the Airflow job was created with the --config option, the Airflow job configuration can be updated with the following command below. For more information, see Creating a pipeline using the CDE CLI linked below.

Procedure

Run the following command in the CDE CLI:

```
cde job update --name my_pipeline --config key1=new_value1 —-config key2=new
_value2
```

The new configuration is merged with the existing job configuration.

Related Information

Creating a pipeline using the CDE CLI

Updating the Airflow file mounts using the CDE CLI [Technical Preview]

You can update or delete an existing file mount, or add new Airflow file mounts for your pipeline with these commands.

Changing an existing file mount

```
cde job update --name my_pipeline --airflow-file-mount-1-resource my_new_pip
eline_resource
```

Changing a file mount prefix

```
cde job update --name my_pipeline --airflow-file-mount-1-prefix my_new_pipel
ine_resource_prefix
```

Adding a new file mount

Add a new file mount when one already exists:

```
cde job update --name my_pipeline --airflow-file-mount-2-resource my_new_pip
eline_resource
```

Removing an existing file mount

```
cde job update --name my_pipeline --unset-airflow-file-mount-index 2
```

Note: You can only delete one mount at a time. After the mount is deleted, they will be re-indexed.

Deleting an Airflow pipeline using the CDE CLI

You can delete a pipeline in Cloudera Data Engineering using the CLI.

Procedure

To delete a pipeline, give the job name to the delete command:

```
cde job delete --name my_pipeline
```

Creating and managing Airflow connections

You can create multiple connections to various services using the embedded Airflow UI.

Creating a connection to Cloudera Data Warehouse or Cloudera Data Hub instance for SQL Operator

Learn how to create an Airflow connection to an existing Cloudera Data Warehouse or a Cloudera Data Hub instance before running the workloads using the supported Airflow SQL Operators.

About this task

The following steps are for using the Airflow service provided with each Cloudera Data Engineering Virtual Cluster. For information about using your own Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

Before you begin

To determine the Cloudera Data Warehouse hostname to use for the connection, perform the following steps:

1. In the Cloudera Cloudera Management Console, click the Data Warehouse tile and click Overview.
2. In the Virtual Warehouses column, locate the Hive or Impala warehouse you want to connect to.
3. Click ⋮ next to the selected Warehouse, and then click Copy JDBC URL.
4. Paste the URL into a text editor, and make note of the hostname and the httpPath.

   For example,

   ```
   jdbc:hive2://hs2-aws-2-hive.env-k5ip0r.dw.ylcu-atmi.cloudera.site/defaul
   t;transportMode=http;httpPath=cliservice;ssl=true;retries=3;
   ```

   In this JDBC URL, the hostname is `hs2-aws-2-hive.env-k5ip0r.dw.ylcu-atmi.cloudera.site` and the httpPath is `cliservice`.

To determine the Cloudera Data Hub hostname to use for the connection, perform the following steps:

1. In the Cloudera Management Console, navigate to a suitable Cloudera Data Hub instance.
2. Click the Endpoints tab.
3. Select and copy the JDBC URL for the Warehouse.
4. Paste the URL into a text editor, and make note of the hostname and the httpPath parameters. The httpPath parameter is different for each Cloudera Data Hub.

   For example,

   ```
   jdbc:hive2://man-az-1-master0.man-azur.xcu2-8y8x.dev.cldr.work/;ssl=true
   ;transportMode=http;httpPath=man-az-1/cdp-proxy-api/hive
   ```

   In this JDBC URL, the hostname is `man-az-1-master0.man-azur.xcu2-8y8x.dev.cldr.work` and the httpPath is `man-az-1/cdp-proxy-api/hive`.

Procedure

To create a connection to an existing Cloudera Data Warehouse virtual warehouse using the embedded Airflow UI, perform the following steps:

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu and select the service containing the Virtual Cluster that you are using.
3. In the Virtual Clusters column, click Cluster Details for the Virtual Cluster.
4. Click AIRFLOW UI.
5. From the Airflow UI, click the Connection link from the Admin menu.

   Important: The connection credentials that you specify here can be used by any DAG created by any user in the same Virtual Cluster. Cloudera recommends you to use a Cloudera machine user for the connection to Cloudera Data Warehouse and limiting access to the Cloudera Data Engineering Virtual Cluster to trusted users using Virtual Cluster ACLs.

6. Click the plus sign to add a new record and fill the following fields:
   - Conn Id: Create a unique connection identifier. For example, cdw-hive-sql.
   - Conn Type: Select Impala, regardless of the Data Warehouse type whether it is Impala or Hive.
   - Host: Enter the hostname copied from the JDBC connection URL. Do not enter the full JDBC URL.
   - Schema: Enter the schema to be used. The default value is default.
   - Login/Password: Both Impala and Hive support JWT token based authentication if configured accordingly. If JWT token is used, make sure that you have left the Username and Password fields blank. If not, enter your workload username and password.
   - Port: 443.
   - Extra: Extra arguments must contain the following options:

     Note: The httpPath parameter must be the same that was copied in the prerequisites steps. From the preceding examples in the Prerequisites steps, for Cloudera Data Warehouse, the httpPath parameter is cliserviceand for Cloudera Data Hub, the httpPath parameter is man-az-1/cdp-proxy-api/hive.

     - If JWT token based authentication is used, then enter the following arguement:

       ```
       {"auth_mechanism": "JWT", "use_ssl": "True",
        "use_http_transport": "True", "http_path":
       "<***HTTPPATH_COPIED_FROM_THE_PREQUISITES_STEP***>", "jwt": "JWT_TOKE
       N"}
       ```

     - If the workload username and password are used, then enter the following arguement:

       ```
       {"auth_mechanism": "LDAP", "use_ssl": "True"
       , "use_http_transport": "True", "http_path":
        "<***HTTPPATH_COPIED_FROM_THE_PREQUISITES_STEP***>"}
       ```

7. Click Save.

(Deprecated) Creating a connection to Cloudera Data Warehouse for Cloudera Data Warehouse Operator

Learn how to create an Airflow connection to an existing Cloudera Data Warehouse before running the workloads using the Cloudera Data Warehouse Operator.

About this task

Important: Starting from Cloudera Data Engineering version 1.23.0, Cloudera Data Warehouse Operator is deprecated. Cloudera Data Warehouse Operator supports connections to Apache Hive virtual warehouses only. It does not support connections to Impala virtual warehouses. However, SQL Operator supports connections to both Impala and Hive virtual warehouses. Hence, Cloudera recommends you to use SQL Operator instead of Cloudera Data Warehouse Operator. For more information about creating a connection to Cloudera Data Warehouse using SQL Operator, see Creating a connection to Cloudera Data Warehouse or Cloudera Data Hub instance for SQL Operator.

The following steps are for using the Airflow service provided with each Cloudera Data Engineering virtual cluster. For information about using your own Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

Before you begin

To determine the Cloudera Data Warehouse hostname to use for the connection, perform the following steps:

1. In the Cloudera Management Console, click the Data Warehouse tile and click Overview.
2. In the Virtual Warehouses column, locate the Hive or Impala warehouse you want to connect to.
3. Click ⋮ next to the selected Warehouse, and then click Copy JDBC URL.

4. Paste the URL into a text editor, and make note of the hostname.

   For example,

   ```
   jdbc:hive2://hs2-aws-2-hive.env-k5ip0r.dw.ylcu-atmi.cloudera.site/defaul
   t;transportMode=http;httpPath=cliservice;ssl=true;retries=3;
   ```

   In this JDBC URL, the hostname is `hs2-aws-2-hive.env-k5ip0r.dw.ylcu-atmi.cloudera.site`.

Procedure

To create a connection to an existing Cloudera Data Warehouse virtual warehouse using the embedded Airflow UI, perform the following steps:

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu and select the service containing the virtual cluster that you are using.
3. In the Virtual Clusters column, click Cluster Details for the virtual cluster.
4. Click AIRFLOW UI.
5. From the Airflow UI, click the Connection link from the Admin menu.

   Important: The connection credentials that you specify here can be used by any DAG created by any user in the same Virtual Cluster. Cloudera recommends you to use a Cloudera machine user for the connection to Cloudera Data Warehouse and limiting access to the Cloudera Data Engineering virtual cluster to trusted users using virtual cluster ACLs.
6. Click the plus sign to add a new record and fill the following fields:
   - Conn Id: Create a unique connection identifier. For example, cdw-hive-demo.
   - Conn Type: Select Hive Client Wrapper.
   - Host: Enter the hostname copied from the JDBC connection URL. Do not enter the full JDBC URL.
   - Schema: Enter the schema to be used. The default value is default.
   - Login/Password: Enter your workload username and password.
7. Click Save.

Creating a connection to run jobs on other Cloudera Data Engineering Virtual Clusters

Learn how to create a Cloudera Data Engineering type connection to run jobs on other Cloudera Data Engineering Virtual Clusters.

About this task

The following steps are for using the Airflow service provided with each Cloudera Data Engineering Virtual Cluster. For information about using your own Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

To create a connection to an existing Cloudera Data Engineering Virtual Cluster using the embedded Airflow UI, perform the following steps:

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu and select the service containing the Virtual Cluster that you are using.
3. In the Virtual Clusters column, click Cluster Details for the Virtual Cluster.
4. Click AIRFLOW UI.

5. From the Airflow UI, click the Connection link from the Admin menu.
6. Click the plus sign to add a new record and fill the following fields:
   - Conn Id: Create a unique connection identifier. For example, cde_runtime_api.
   - Conn Type: Select Cloudera Data Engineering.
   - Virtual Cluster API endpoint: Enter the target Virtual Clusters Jobs API URL.
   - Cloudera Access Key: Enter the Cloudera access key of the account for running jobs on the Cloudera Data Engineering Virtual Cluster.
   - Cloudera Private Key: Enter the Cloudera private key associated to the Cloudera Access Key that you have entered.
   - Extra: Extra arguments must in JSON format. Available extra parameters are as follows:
     - Proxy: Optional. Translates to https_proxy/HTTPS_PROXY environment variables. The default value is None.
     - Region: Optional. Cloudera Control Plane region ("us-west-1", "eu-1" or "ap-1") is inferred automatically, if not specified.

       For more information about the available extra arguments, see the GitHub page.
7. Click Save.

Executing SQL queries on Cloudera Data Warehouse or Cloudera Data Hub instance using Apache Airflow in Cloudera Data Engineering

About this task

The following steps are for using the Airflow service provided with each Cloudera Data Engineering virtual cluster. For information about using your own Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

Before you begin

To run this job, an existing Airflow connection is required. For more information about how to create an airflow connection, see the following topics:

- For SQL Operators, see Creating a connection to Cloudera Data Warehouse or Cloudera Data Hub instance for SQL Operator.
- For Cloudera Data Warehouse Operators, see Creating a connection to Cloudera Data Warehouse for Cloudera Data Warehouse Operator.

Procedure

Create an Airflow DAG file in Python. Import the required operators and define the tasks and dependencies.

- The following example DAG file uses the connection named “impala-test”, and executes a “`SHOW TABLES`” query on the Impala Virtual Warehouse using the SQLExecuteQueryOperator:

```
import logging
from airflow import DAG
from datetime import datetime, timedelta
from airflow.providers.common.sql.operators.sql import SQLExecuteQueryOp
erator

with DAG(
dag_id="imp_dag",
```

```
start_date=datetime(2024, 2, 9),
schedule_interval=timedelta(days=1),
catchup=False,
) as dag:
  execute_query = SQLExecuteQueryOperator(
    task_id="execute_query",
    conn_id="impala-test",
    sql=f"SHOW TABLES",
    split_statements=True,
    return_last=False,
)
  execute_query
```

For more advanced use cases, see Airflow documentation about the SQL Operators.

- The following example DAG file uses the connection named “cdw-hive”, and executes a “`SHOW TABLES`” query on the Impala Virtual Warehouse using the CdwExecuteQueryOperator:

```
from airflow import DAG
from cloudera.airflow.providers.operators.cdw import CdwExecuteQueryOper
ator
from pendulum import datetime

default_args = {
    'owner': 'dag_owner',
    'depends_on_past': False,
    'start_date':datetime(2024, 2, 9)
}

example_dag = DAG(
    'example-cdwoperator',
    default_args=default_args,
    schedule_interval=None,
    catchup=False,
    is_paused_upon_creation=False
)

cdw_query = """
USE default;
SHOW TABLES;
"""

cdw_step = CdwExecuteQueryOperator(
    task_id='cdw-test',
    dag=example_dag,
    cli_conn_id='cdw-hive',
    hql=cdw_query,
    # The following values `schema`, `query_isolation`
    # are the default values, just presented here for the example.
    schema='default',
    query_isolation=True
)

cdw_step
```

For more information about the CdwExecuteQueryOperator, see the GitHub page.

Running Airflow jobs on other Cloudera Data Engineering Virtual Clusters

You can learn about how to run Airflow jobs on other Cloudera Data Engineering Virtual Clusters.

About this task

The following steps are for using the Apache Airflow service provided with each Cloudera Data Engineering virtual cluster. For more information on using your own Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

Before you begin

To run this job, an existing Airflow connection is required. For more information about how to create an Airflow connection, see Creating a connection to run jobs on other Cloudera Data Engineering Virtual Clusters.

Procedure

Create an Airflow DAG file in Python. Import the required operators and define the tasks and dependencies.

The following example DAG starts a job called “example-scala-pi” on the Cloudera Data Engineering Virtual Cluster defined by the connection named “cde_runtime_api”. The connection “cde_runtime_api” is also the default connection to the same Virtual Cluster the job is launched on. For more information about the operator and its capabilities, see the GitHub page.

```
from datetime import timedelta
from airflow import DAG
from cloudera.airflow.providers.operators.cde import CdeRunJobOperator
import pendulum

default_args = {
    'retry_delay': timedelta(seconds=5),
    'depends_on_past': False,
    'start_date': pendulum.datetime(2021, 1, 1, tz="UTC")
}

example_dag = DAG(
    'example-cdeoperator',
    default_args=default_args,
    schedule_interval='@once',
    catchup=False,
    is_paused_upon_creation=False
)

ingest_step1 = CdeRunJobOperator(
    connection_id='cde_runtime_api',
    task_id='ingest',
    retries=3,
    dag=example_dag,
    job_name='example-scala-pi'
)

prep_step2 = CdeRunJobOperator(
    task_id='data_prep',
    dag=example_dag,
    job_name='example-scala-pi'
)
ingest_step1 >> prep_step2
```

Using Cloudera Data Engineering with an external Apache Airflow deployment

Before you begin

The Cloudera provider for Apache Airflow, available at the Cloudera GitHub repository, provides two Airflow operators for running Cloudera Data Engineering and Cloudera Data Warehouse jobs. You can install the provider on your existing Apache Airflow deployment to integrate.

Important: Cloudera Data Engineering on Cloudera Private Cloud currently supports only the Cloudera Data Engineering job run operator.

- The Cloudera provider for Apache Airflow is for use with existing Airflow deployments. If you want to use the embedded Airflow service provided by Cloudera Data Engineering, see Apache Airflow in Cloudera Data Engineering.
- The provider requires Python 3.6 or higher.
- The provider requires the Python cryptography package version 3.3.2 or higher to address CVE-2020-36242. If an older version is installed, the plugin automatically updates the cryptography library.

About this task

This component provides two Airflow operators to be integrated in your DAGs:

- CdeRunJobOperator, for running Cloudera Data Engineering jobs.
- CDWOperator, for accessing Cloudera Data Warehouse

Procedure

Install Cloudera Airflow provider on your Airflow servers

1. Run the following pip command on each Airflow server: `pip install cloudera-airflow-provider`

Create a connection using the Airflow UI

Before you can run a Cloudera Data Engineering job from your Airflow deployment, you must configure a connection using the Airflow UI.

2. From the Cloudera Data Engineering home page, go to Overview Virtual Clusters Cluster Details of the Virtual Cluster (VC) where you want the Cloudera Data Engineering job to run.
3. Click JOBS API URL to copy the URL.
4. Go to your Airflow web console (where you installed the Cloudera provider).
5. Go to Admin Connection .
6. Click + Add a new record.

7. Fill in connection details:

   Conn Id

   Create a unique connection identifier.

   Conn Type

   The type of the connection. From the drop-down, select

   - HTTP (if you are using Apache Airflow version 1)
   - HTTP or Cloudera Data engineering (if you are using Apache Airflow version 2)

   Host/Virtual API Endpoint

   URL of the host where you want the job to run. Paste here the JOBS API URL you copied in a previous step.

   Login/Cloudera Access Key

   Provide the Cloudera access key of the account for running jobs on the Cloudera Data Engineering VC.

   Password/Cloudera Private Key

   Provide the Cloudera private key of the account for running jobs on the Cloudera Data Engineering VC.

8. Click Save.
9. In the Cloudera Data Engineering Home page, click Jobs in the left navigation menu, and then click Create Job.
10. Fill in the Job Details:

    Job Type

    Select the option matching your use case.

    Name

    Specify a name for the job.

    DAG File

    Provide a DAG file.

    Use the `CdeRunJobOperator` to specify a Cloudera Data Engineering job to run. The job definition in the DAG file must contain:

    connection_id

    The Conn Id you specified on the Airflow UI when creating the connection.

    task_id

    The ID that identifies the job within the DAG.

    dag

    The variable containing the dag object

    job_name

    The name of the Cloudera Data Engineering job to run. This job must exist in the Cloudera Data Engineering virtual cluster you are connecting to.

    For example:

```
from cloudera.cdp.airflow.operators.cde_operator import CdeRunJobOperator
...
t1 = CdeRunJobOperator(
    connection_id='cde-vc01-dev',
    task_id='ingest',
    dag=example_dag,
    job_name='etl-ingest-job'
)
```

11. Click Create and Run to create the job and run it immediately, or click the dropdown button and select Create to create the job.

Supported Airflow operators and hooks

Apache Airflow Python DAG files can be used to automate workflows or data pipelines in Cloudera Data Engineering. Cloudera Data Engineering currently supports a specified list of Airflow operators and hooks that can be used.

Airflow operators

Cloudera Data Engineering supports the following Airflow operators:

- airflow.operators.bash
- airflow.operators.branch
- airflow.operators.datetime
- airflow.operators.email
- airflow.operators.empty
- airflow.operators.generic_transfer
- airflow.operators.latest_only
- airflow.operators.python
- airflow.operators.trigger_dagrun
- airflow.operators.weekday

Airflow hooks

Cloudera Data Engineering supports the following Airflow hooks:

- airflow.hooks.filesystem
- airflow.hooks.package_index
- airflow.hooks.subprocess

Related Information

Apache Airflow in Cloudera Data Engineering

Using custom operators and libraries for Apache Airflow

You can install and use custom python packages for Airflow with Cloudera Data Engineering. Cloudera provides access to the open source packages that you can use for your Airflow jobs using the Cloudera Data Engineering user interface (UI).

Adding custom operators and libraries

You can add custom python packages for Airflow with Cloudera Data Engineering. Cloudera provides access to the open source packages that you can use for your Airflow jobs using the UI.

About this task

While you can install the operator, if additional runtime dependencies are required such as additional setup with binaries on the path, and environment configuration like Kerberos and Cloud credentials, and so on, then the operator will not work. To use an Airflow third-party operator for your custom library and operator package, you must configure the Airflow connection in the Airflow UI.

Note: You can use all officially supported third-party providers for Apache Airflow. Any provider listed in Apache Airflow's official documentation is fully compatible, excluding potential cases of dependency conflicts with your custom package.

Note: The installed operators and libraries will be available to all jobs defined in the chosen Cloudera Data Engineering Virtual Cluster.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering **Home** page displays.
2. Click Administration in the left navigation menu. The **Administration** page displays.
3. Locate the Virtual Cluster that you want to edit, and click Cluster Details.
4. Go to the **Airflow** tab. The **Libraries and Operators** page displays.
5. Under the **Configure Repositories** section, enter the following fields to configure the Python Package Index (PyPi) repositories used to source your custom libraries and operators:
   a) PyPI Repository URL - Enter the Python Package Index (PyPi) URL.
   b) SSL Certificate - Enter the PEM-encoded CA certificate.
   c) Enter Authorization Credentials if you are configuring a Private or Protected PyPi Repository that requires authorization for access.
      - Username
      - Password
6. Click Validate Configurations.
7. Under the **Build** section, upload a requirements.txt file that contains a list of all library and operator packages that you want to enable. Once uploaded, the system will automatically build and install your packages.

   You can specify any Python package which is compatible with the Airflow python constraints, which you can find here:

   https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/constraints-${PYTHON_VERSION}.txt

   Tip: The Airflow and Python versions (${AIRFLOW_VERSION} and ${PYTHON_VERSION} respectively) depend on your Cloudera Data Engineering version.
8. Click Activate. The activation restarts the Airflow server. This may take a few minutes. Once activation is complete, you will see the Installed Packages listed.

   You can now create and run an Airflow job using the custom library and operators that you have acitvated.

Related Information

Third-party Airflow Operators
Requirement files

Updating custom operators and libraries

After creating your custom operator and libraries, you can update and add more configurations to the installed package.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering **Home** page displays.
2. Click Administration in the left navigation menu. The **Administration** page displays.
3. Locate the Virtual Cluster that you want to edit, and click Cluster Details.
4. Go to the **Airflow** tab. The **Libraries and Operators** page displays.
5. Click the Installed Package, and click Actions Update

6. Under **Configure Repositories**, update configurations or click Add Configurations, then click Validate Configurations to confirm your changes.
7. Under **Build**, you can upload a new requirements.txt file if needed.
8. Click Activate to confirm your updates.

Deleting custom operators and libraries

If you no longer need your custom operators and libraries, you can delete it. To delete an installed package containing custom operators and libraries, complete the following steps.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering **Home** page displays.
2. Click Administration in the left navigation menu. The **Administration** page displays.
3. Locate the Virtual Cluster that you want to edit, and click Cluster Details.
4. Go to the **Airflow** tab. The **Libraries and Operators** page displays.
5. Under Installed Package , click Actions Delete Deleting may take a few minutes.

Troubleshooting custom operators and libraries

Learn the different ways that you can troubleshoot the setup of your Airflow Python environment in Cloudera Data Engineering.

Viewing logs for custom operators and libraries

After adding custom operators and libraries, you will see them listed under Instsalled Packages. You can view the logs related to the build and activation process of the installed packages. To view logs, complete the following steps:

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering **Home** page displays.
2. Click Administration in the left navigation menu. The **Administration** page displays.
3. Locate the Virtual Cluster that you want to edit, and click Cluster Details. The **Libraries and Operators** page displays.
4. Under Installed Package, click Actions View Logs . The logs information is displayed in a new window.

Cloudera Data Engineering Overview

Date published:
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Data Engineering service 4

Cloudera Data Engineering auto-scaling 5

Service and virtual cluster auto-scaling 5

Job auto-scaling 6

Cloudera Data Engineering resources 6

Cloudera Data Engineering service

Cloudera Data Engineering is a serverless service for Cloudera that allows you to submit batch jobs to auto-scaling virtual clusters. Cloudera Data Engineering enables you to spend more time on your applications, and less time on infrastructure.

Note: Cloudera Data Engineering is currently available only on Amazon AWS and Microsoft Azure.

Cloudera Data Engineering allows you to create, manage, and schedule Apache Spark jobs without the overhead of creating and maintaining Spark clusters. With Cloudera Data Engineering, you define virtual clusters with a range of CPU and memory resources, and the cluster scales up and down as needed to run your Spark workloads, helping to control your cloud costs.

The Cloudera Data Engineering service involves several components:

Environment

A logical subset of your cloud provider account including a specific virtual network. For more information, see Environments.

Cloudera Data Engineering Service

The long-running Kubernetes cluster and services that manage the virtual clusters. The Cloudera Data Engineering service must be enabled on an environment before you can create any virtual clusters.

Virtual Cluster

An individual auto-scaling cluster with defined CPU and memory ranges. Virtual Clusters in Cloudera Data Engineering can be created and deleted on demand. Jobs are associated with clusters.

Jobs

Application code along with defined configurations and resources. Jobs can be run on demand or scheduled. An individual job execution is called a job run.

Resource

A defined collection of files such as a Python file or application JAR, dependencies, and any other reference files required for a job.

Job run

An individual job run.

The Cloudera Data Engineering service differs from a Cloudera Data Engineering Data Hub cluster in several ways, including the following:

Table 1: Cloudera Data Engineering Service vs. Cloudera Data Engineering Data Hub

Feature	Cloudera Data Engineering	Data Hub DE Template
Cloud providers	Amazon AWS, Microsoft Azure	Amazon AWS, Microsoft Azure
Compute engines	Apache Spark	Apache Spark, Apache Hive
Deployment	Kubernetes	Virtual machines (cloud provider)
Resource management	Yunikorn, Kubernetes	YARN
Troubleshooting	Cloudera Data Engineering deep analysis, Spark History Server	Spark History Server
Portability	Public/private cloud	Public cloud
Job submission	Managed API	Apache Livy

Browser Requirements

Supported browsers:

- Chrome
- Safari

Unsupported browsers:

- Firefox

Cloudera Data Engineering auto-scaling

Cloudera Data Engineering auto-scales at the job level as well as the service and virtual cluster level. Service and virtual cluster autoscaling uses Apache Yunikorn (Incubating) for resource scheduling to improve efficiency and utilization, reducing cloud provider costs. Job auto-scaling is managed by Apache Spark dynamic allocation. Cloudera Data Engineering scales resources up and down as needed for running jobs.

Service and virtual cluster auto-scaling

Service and virtual cluster autoscaling uses Apache Yunikorn (Incubating) for resource scheduling to improve efficiency and utilization

When you create a Cloudera Data Engineering service, you specify an instance type (size) and auto-scale range of instances. Virtual clusters associated with the Cloudera Data Engineering service use CPU and memory resources as needed to run jobs. When more resources are required, virtual machines of the specified instance type are started. When resources are no longer required, instances are terminated.

Virtual clusters also have auto-scaling controls, specified as maximum CPU cores and memory (in gigabytes).

Cloudera Data Engineering takes advantage of YuniKorn resource scheduling and sorting policies, such as gang scheduling and bin packing, to optimize resource utilization and improve cost efficiency.

By default, Cloudera Data Engineering uses the bin packing policy to allocate resources. It sorts the list of nodes ascending by the amount of available resources and the node with the lowest amount of available resources is selected. Sorting is solely based on an array of available memory in MBs and available CPUs in milli CPUs (mCPU), considering absolute numbers only. It disregards resource types and percentage of total resources available. For example, if you have three nodes with the following amounts of available resources:

Node	Available memory in MB	Available CPU in mCPU
nodeA	100,000	4,000
nodeB	4,000	10,000
nodeC	10,000	10,000

YuniKorn makes the following sorting:

nodeB	4,000	10,000
nodeA	4,000	100,000
nodeC	10,000	10,000

If bin packing does not provide optimal results for the applications you run, consider selecting one of the other available sorting policies.

For more information on gang scheduling, see the Cloudera blog post Spark on Kubernetes – Gang Scheduling with YuniKorn.

Job auto-scaling

Job auto-scaling is controlled by Apache Spark dynamic allocation.

Dynamic allocation scales job executors up and down as needed for running jobs. This can provide large performance benefits by allocating as many resources as needed by the running job, and by returning resources when they are not needed so that concurrent jobs can potentially run faster.

Resources are limited by the job configuration (executor range) as well as the virtual cluster auto-scaling parameters. By default, the executor range is set to match the range of CPU cores configured for the virtual cluster. This improves resource utilization and efficiency by allowing jobs to scale up to the maximum virtual cluster resources available, without manually tuning and optimizing the number of executors per job.

Cloudera Data Engineering resources

A *resource* in Cloudera Data Engineering is a named collection of files used by a job or a session. Resources can include application code, configuration files, custom Docker images, and Python virtual environment specifications (requirements.txt).

Note: Custom Docker container images is a *Technical Preview* feature, requiring entitlement. Contact your Cloudera account representative to enable access to this feature. See Cloudera Public Cloud Preview Features below.

Resources are associated with virtual clusters. A resource can be used by multiple jobs, and jobs can use multiple resources.The resource types supported by Cloudera Data Engineering are files, python-env, and custom-runtime-image.

files

An arbitrary collection of files that a job can reference. The application code for the job, including any necessary configuration files or supporting libraries, can be stored in a files resource. Files can be uploaded to and removed from a resource as needed.

python-env

A defined virtual Python environment that a job runs in. The only file that can be uploaded to a python-env resource is a requirements.txt file. When you associate a python-env resource with a job, the job runs within a Python virtual environment built according to the requirements.txt specification.

custom-runtime-image

A Docker container image. When you run a job using a custom-runtime-image resource, the executors that are launched use your custom image.

Related Information

Cloudera Public Cloud Preview Features

Cloudera Data Engineering 1.23.0

Managing Performance in Cloudera Data Engineering

Date published: 2020-07-30
Date modified: 2024-11-12

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Data Engineering performance tuning.....4

Cloudera Data Engineering performance tuning

Use local SSD

Using default instances without local SSD does impact performance. Instances with local SSDs generally provide faster performance, but you need to ensure you choose instance types that have large enough local storage for intermediate results.

Avoid instance types with low compute profiles

Make sure there is reasonable headroom (at least 20%) between the per executor memory and core usage and what the instance profile provides.

Disable Spark Analysis

By default, Cloudera Data Engineering collects metrics around CPU, memory, and I/O. This can have a negative impact on performance - especially for long running jobs that take several hours to complete. Profiling metrics are useful in development and testing, but an unnecessary overhead in a production setting.

You can turn this feature off using the Spark Analysis toggle on the job Configuration tab.

Use Spark 3 instead of Spark 2

Spark 3 introduced performance improvements over Spark 2, therefore Cloudera recommends using it whenever possible.

Cloudera Data Engineering 1.23.0

Cloudera Data Engineering Prerequisites

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Amazon AWS prerequisites for Cloudera Data Engineering..... 4
Enabling Customer Managed Keys (CMK) on Amazon Web Services (AWS).....8

Microsoft Azure prerequisites for Cloudera Data Engineering.....9

Supporting Azure private storage.....12

Using AWS IAM restricted roles and policies for compute and Cloudera Data Engineering..... 14
Create IAM roles and instance profile pair..... 14
Create role and policy used to deploy Cloudera environments for Cloudera Data Engineering.....21

Amazon AWS prerequisites for Cloudera Data Engineering

Amazon Web Services (AWS) prerequisites for Cloudera Data Engineering.

Review AWS account prerequisites for Cloudera

Refer to the Cloudera AWS account requirements and verify that the AWS account you are using for Cloudera has the required resources, and that you have the permissions required to manage these resources.

Review Cloudera Data Engineering-specific AWS resource requirements

Provisioning a Cloudera Data Engineering service and virtual clusters require access to the following AWS resources.

AWS Services used by Cloudera Data Engineering

- Network – Amazon VPC (see below for requirements)
- Compute – Amazon Elastic Kubernetes Service (EKS)
- Load Balancing – Amazon ELB Classic Load Balancer
- Key Management – AWS Key Management Service (KMS)
- DNS – Amazon Route 53 (Cloudera Data Engineering makes use of this but it is hosted in Cloudera's AWS infrastructure)
- Persistent Instance Storage – Amazon Elastic Block Store (EBS)
- Persistent Service and Virtual Cluster Storage – Amazon Elastic File System (EFS)
- Database – Amazon Relational Database Service (RDS)

VPC Requirements

You can use an existing VPC, or allow Cloudera to create one when you create an environment.

Option 1: use your own VPC

Minimum requirements:

- Cloudera Data Engineering requires at least two subnets, each in a different Availability Zone (AZ). If you require a public endpoint for Cloudera Data Engineering, provision at least one public subnet.
- Ensure that the CIDR block for the subnets is sized appropriately. For each Cloudera Data Engineering environment, in addition to ensuring enough IPs to accomodate the maximum number of autoscaling compute instances, allow for a fixed overhead of three instances for core Cloudera Data Engineering services and approximately one instance for every two virtual clusters.
- You must enable DNS for the VPC.

Recommended setup:

- Cloudera recommends that you provision at least three subnets, each in a different Availability Zone (AZ). If you do not require a public endpoint, use three private subnets. If you require a public endpoint, use at least two private subnets and one public subnet.
- Private subnets should have routable IPs over your internal VPN. If IPs are not routable, private Cloudera Data Engineering endpoints must be accessed via a SOCKS. This is not recommended.
- Tag the VPC and the subnets as shared so that Kubernetes can find them. For load balancers to be able to choose the subnets correctly, you are also required to tag private subnets with the kubernetes.io/role/internal-elb:1 tag, and public subnets with the kubernetes.io/role/elb:1 tag.

Note that only the load balancer needs to be on a public subnet for access to Cloudera Data Engineering. By default, if they are available, Cloudera Data Engineering will configure the EKS to run on private subnets.

Option 2: Cloudera creates a new VPC

If you choose to allow Cloudera to create a new VPC, three subnets will be automatically created. One subnet is created for each availability zone assuming three AZs per region; If a region has two AZs instead of three, three subnets are still created, with two in the same AZ.

You will be asked to specify a valid CIDR in IPv4 range that will be used to define the range of private IPs for EC2 instances provisioned into these subnets.

Related AWS documentation: Amazon EKS - Cluster VPC Considerations, Creating a VPC for your Amazon EKS Cluster

Firewall requirements

HTTPS access to Cloudera Data Engineering endpoints is available over port 443 for the following cases:

- Internal only – Should be accessible from your organization's network, but not the public internet.
- Internet facing (public endpoint) – Should be accessible from the public internet as well as your organization's internal network.

If you are using a firewall or a security group setting to prevent egress traffic from the service, make sure that the cont ainer.repository.cloudera.com and docker.repository.cloudera.com URLs on port 443 are allowed at all times.

If egress traffic is blocked to these URLs, then autoscaling cannot pull images, which can result in broken pods. For more information on required outbound access, see Outbound network access destinations for AWS and Security groups.

Do not remove firewall rules added during provisioning. The rules are also required for regular operation. You must also maintain the minimum firewall requirements set by the cloud provider. For more information, see Amazon EKS security group considerations in the Amazon AWS documentation.

If you're using Amazon Relational Database Service (RDS), you'll need to ensure that you are using *.*.rds.amazonaws.com and TCP 5432 / 3306 / 443 ports.

Review the default AWS service limits and your current AWS account limits

By default, AWS imposes certain default limits for AWS services for each user account. Make sure you review your account's current usage status and resource limits before you start provisioning additional resources for Cloudera and Cloudera Data Engineering.

For example, depending on your AWS account, you may only be allowed to provision a certain number of EC2 instances. Be sure to review your AWS service limits before your proceed.

Related AWS documentation: AWS Service Limits, Amazon EC2 Resource Limits.

Supported AWS regions

Cloudera supports the following AWS regions: Supported AWS regions. However, the Cloudera Data Engineering service also requires AWS Elastic Kubernetes Service (EKS). Make sure you select a region that includes EKS.

Related AWS documentation: Region Table.

Set up an AWS Cloud Credential

Create a role-based AWS credential that allows Cloudera to authenticate with your AWS account and has authorization to provision AWS resources on your behalf. Role-based authentication uses an IAM role with an attached IAM policy that has the minimum permissions required to use Cloudera.

Once you have created this IAM policy, register it in Cloudera as a cloud credential. Reference this credential when you register an AWS environment in Cloudera environment as described in the next step.

Instructions: Cross-account access IAM role

Register an AWS Environment in Cloudera

A Cloudera user must have the Environment Creator role in order to register an environment. An environment determines the specific cloud provider region and virtual network in which resources can be provisioned, and includes the credential that should be used to access the cloud provider account.

Cloudera Data Engineering supports deployment into environments with non-transparent proxies. To use this feature, you need to register a proxy and add it to the environment during environment registration. Registering a proxy requires Power User privileges.

Instructions: Register an AWS environment

Cloudera Data Engineering Role Requirements

There are two Cloudera user roles associated with the Cloudera Data Engineering service: DEAdmin and DEUser. Any Cloudera user with the EnvironmentAdmin (or higher) access level must assign these roles to users who require access to the Cloudera Data Engineering console within their environment.

Furthermore, if you want to allow users to log in to provisioned workspaces and run workloads on them, this will need to be configured separately.

Set up the AWS account to run kubectl commands

1. In the AWS console, create an IAM user (for example, kubectl-user) with Programmatic access (you don't need to grant any permissions).
2. Note the User ARN and copy the Access key ID and Secret access key and set up an AWS profile as follows:

```
[kubectl-user]

aws_access_key_id = <Access Key ID>
aws_secret_access_key = <Secret access key>
```

3. Navigate to IAM Roles and edit the cross-account IAM role (note the Role ARN) that was created as part of the Cloudera prerequisites.
4. Navigate to Trust relationships > Edit trust relationships.
5. Add the following to the policy document, then click Update trust policy.

```
 "Effect": "Allow",
 "Principal": {
  "AWS": "User ARN from step 2"
 },
 "Action": "sts:AssumeRole"
 },
```

6. Download the kubeconfig file from the Cloudera Data Engineering UI and save it (~/.kube/cde-env1-kube-config, for example), then run the following shell commands:

```
$ export AWS_PROFILE=kubectl-user
$ unset AWS_ACCESS_KEY_ID AWS_SECRET_ACCESS_KEY AWS_SESSION_TOKEN
$ cred=$(aws sts assume-role --role-arn <Role ARN from step 3> --role-se
ssion-name test | jq .Credentials)
$ export AWS_ACCESS_KEY_ID=$(echo $cred|jq .AccessKeyId|tr -d '"')
$ export AWS_SECRET_ACCESS_KEY=$(echo $cred|jq .SecretAccessKey|tr -d '"')
$ export AWS_SESSION_TOKEN=$(echo $cred|jq .SessionToken|tr -d '"')
$ export KUBECONFIG=~/.kube/cde-env1-kube-config
$ export TILLER_NAMESPACE=tiller
```

7. You should now be able to run kubectl commands.

Using AWS S3 buckets with encryption

You may need to incorporate a policy to use at-rest encryption on your Amazon Web Services (AWS) S3 buckets and telemetry log bucket. Starting with Cloudera Data Engineering 1.18 or higher, telemetry buckets with a customer-managed key is supported. When the policy is used, the data is encrypted before it is saved to your disk in S3 and is decrypted when read. This encryption and decryption takes place in the S3 infrastructure and is transparent to authentiated clients. See server-side encryption listed below under Encrypting Data on S3.

For Cloudera Data Engineering to write and read data to and from an encrypted S3 bucket, you must configure a KMS Key ARN under the Customer Managed Encryption Key for a Cloudera environment before you create a Cloudera Data Engineering service.

Once the KMS KEY ARN is configured, newly created Cloudera Data Engineering services will use that key to access the encrypted bucket. If the key is not configured or is invalid, then Cloudera Data Engineering cant access the encrypted telemetry bucket. This results in service/jobs logs not being stored on S3 and will be available on the Virtual Cluster user interface or for Diagnostics bundles. Additionally, the Spark user interface will not be available for completed applications.

There may be cases when you want the telemetry bucket to be encrypted with a key that is different from the one that is specified under the Customer Managed Encryption Key (see Adding a customer managed encryption key to a Cloudera environment running on AWS linked below) and use it to encrypt the EBS volumes and RDS instances running in the environment. In those cases, it's possible to override KMS KEY ARN via the "telemetry.encryption.key" property during service creation.

Using Customer Managed Keys (CMK) encryption

```
You can use customer managed key (CMK) enabled environments for Cloudera
 Data Engineering services deployed on AWS to use CMK-based data-at-rest
encryption for Amazon Relational Database Service (RDS), Amazon Elastic File
 System (EFS) and Kubernetes secrets. For more information, see Enabling Cus
tomer Managed Keys (CMK) on Amazon Web Services (AWS) linked below.
```

While creating a CMK, if not provided, the AWS Key Management Service (KMS) creates a default key-policy for the newly created KMS key. The default key-policy allows Cloudera cross-account roles to use CMK. Instead of using default key-policy, if you want to change the key-policy, then the key-policy must contain the permission blocks below for Cloudera Data Engineering service to use CMK:

```
{
    "Sid": "AllowAutoscalingAndCDPCrossAccountRoleUseOfTheCMK",
    "Effect": "Allow",
    "Principal":
    {
        "AWS":
        [
            "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/CDP-CROSSACCOUNT-ROLE",
            "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/autoscal
ing.amazonaws.com/AWSServiceRoleForAutoScaling"
        ]
    },
    "Action":
    [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "AllowAutoscalingAndCDPCrossAccountRoleUseOfTheCMKGrants",
```

```
    "Effect": "Allow",
    "Principal":
    {
        "AWS":
        [
            "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/CDP-CROSSACCOUNT-ROLE",
            "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/autoscali
ng.amazonaws.com/AWSServiceRoleForAutoScaling"
        ]
    },
    "Action":
    [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition":
    {
        "Bool":
        {
            "kms:GrantIsForAWSResource": "true"
        }
    }
},
{
    "Sid": "AllowCreateGrantToLiftieCluster",
    "Effect": "Allow",
    "Principal":
    {
        "AWS": "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/CDP-CROSSACCOUNT-ROLE"
    },
    "Action": "kms:CreateGrant",
    "Resource": "*",
    "Condition":
    {
        "StringEquals":
        {
            "aws:CalledViaFirst": "cloudformation.amazonaws.com"
        },
        "ForAllValues:StringEquals":
        {
            "kms:GrantOperations":
            [
                "Encrypt",
                "Decrypt"
            ]
        }
    }
}
```

Related Information

Accessing the Kubernetes dashboard

Enabling CMK on AWS

Encrypting Data on S3

Enabling Customer Managed Keys (CMK) on Amazon Web Services (AWS)

Learn how to use customer managed keys (CMK) enabled environments for Cloudera Data Engineering services deployed on AWS using CMK-based data at rest encryption for Amazon Relational Database Service (RDS), Kubernetes secrets, and data at rest encryption.

Before you begin

To perform these steps, you must be a Cloudera Data Engineering Admin and obtain your Customer-managed key from your Amazon Key Management Service (KMS). This CMK that you obtain will be associated with the environment for the steps below.

Add the CMK

Once you've obtained the CMK, complete the following steps:

1. Go to the Management Console.
2. Click Environments.
3. Click the environment where your Cloudera Data Engineering Service is deployed.
4. In the Customer Managed Encryption Key section, click Edit.
5. Toggle Enable Customer-Managed Keys.
6. Select the Select Encryption Key field.
7. Click Save. Once saved, the key is associated with the environment and you are unable to change the CMK. Now, any Cloudera Data Engineering Service deployed using this CMK enabled environment uses the CMK-based data at rest encryption for EFS.

Microsoft Azure prerequisites for Cloudera Data Engineering

Microsoft Azure prerequisites for Cloudera Data Engineering.

Review Azure account prerequisites for Cloudera

Refer to the Azure subscription requirements, and make sure that the Azure account you are using for Cloudera has the required resources, and that you have the permissions required to manage these resources.

Review Cloudera Data Engineering-specific Azure resource requirements

The following Azure services are required to provision a Cloudera Data Engineering Service and virtual clusters:

Azure Services used by Cloudera Data Engineering

- Network – VNet and Subnets(see below for requirements)
- Database – Azure Database for MySQL server
- Compute – Azure Kubernetes Service (AKS)
- Load Balancing - Azure Load Balancer
- Virtual machine scale set
- Storage account - Cloudera Data Engineering stores workload data and logs in Azure Data Lake Store Gen 2 (ADLS Gen2) environment storage account. The AKS service also generates a separate storage account for use with Azure Files.
- Azure Files - Contain job resources, application code, Apache Airflow DAG files and any other uploaded files. The AKS service generates an ADLS Gen2 storage account for these files.
- Log Analytics workspace

Refer to the Azure resources used by Cloudera to check the Azure resources used by Cloudera.

Vnet and Subnet Requirements

When registering an Azure environment in Cloudera, you will be asked to select a VNet and one or more subnets. Cloudera Data Engineering runs in the VNet registered in Cloudera as part of your Azure environment.

You have two options:

- Use your existing VNet and subnets for provisioning Cloudera resources

- Have Cloudera create a new VNet and subnets

New VNet and subnets

If you would like Cloudera to create a new VNet, you will need to specify a valid CIDR in IPv4 range that will be used to define the range of private IPs for VM instances provisioned into these subnets. This must be a /16 CIDR, but you can customize the IP Range. The default is 10.10.0.0/16.

Cloudera will divide this address range as follows:

- 32 x /24 private subnet for Cloudera AI and Cloudera Data Engineering
- 3 x /19 private subnet for Cloudera Data Warehouse
- 3 x /19 private subnet for Data Lake and Cloudera Data Hub
- 3 x /24 public subnet

Existing VNet and subnets

VNet Requirements

If you would like to use your own VNet, it needs to fulfill the following requirements:

- The VNet has at least one subnet
- VNet should be able to make an outbound connection with the internet or set of CIDRs and ports provided by Cloudera

Subnet Requirements

Each Cloudera Data Engineering service requires its own subnet, and must not share subnets or routing tables with any other Cloudera service or AKS cluster. Cloudera Data Engineering on AKS uses the Kubenet CNI plugin provided by Azure. In order to use Kubenet CNI, you must create multiple smaller subnets when creating an Azure environment. Cloudera recommends partitioning the VNet with subnets that accommodate the maximum number of expected cluster nodes.

If you are not sure what size of subnet to use, Cloudera recommends a 24-bit subnet mask (/24 CIDR prefix), which can accommodate 254 usable IP addresses per subnet. If you want to use a different subnet size, you can calculate the number of IP addresses required per Cloudera Data Engineering service as follows:

- Each Cloudera Data Engineering service can scale up to 100 compute nodes; each node consumes one IP address.
- Each Cloudera Data Engineering service requires 3 IP addresses for the infrastructure nodes
- Each virtual cluster within a Cloudera Data Engineering service requires 2 IP addresses

For more information, see VNet and subnets, VNet and subnet planning

Firewall requirements

HTTPS access to Cloudera Data Engineering endpoints is available over port 443 for the following cases:

- Internal only – Should be accessible from your organization's network, but not the public internet.
- Internet facing (public endpoint) – Should be accessible from the public internet as well as your organization's internal network.

If you are using a firewall or a security group setting to prevent egress traffic from the service, make sure that the cont ainer.repository.cloudera.com and docker.repository.cloudera.com URLs on port 443 are allowed at all times.

If egress traffic is blocked to these URLs, then autoscaling cannot pull images, which can result in broken pods. For more information on required outbound access, see Outbound network access destinations for Azure and Default security group settings on Azure.

Do not remove firewall rules added during provisioning. The rules are also required for regular operation. You must also maintain the minimum firewall requirements set by the cloud provider. For more information, see Use Azure Firewall to protect Azure Kubernetes Service (AKS) Deployments in the Microsoft Azure documentation.

Review the default Azure service limits and your current Azure account limits

Azure portal imposes default limits to the resources available to each user subscription, which may vary for different regions. Make sure you review your Azure subscription's current usage status and resource limits before you start provisioning additional resources for Cloudera and Cloudera Data Engineering.

If you require more resources than the limit set by Azure, you can create a support request on your Azure Portal.

For example, To register an Azure environment in Cloudera, you may need to increase some of these limits for the region(s) that you are planning to use.Cloudera creates resources such as VMs in your Azure subscription. Depending on the number of clusters that Cloudera creates in your Azure subscription, you might need to raise the limits for certain resources such as VMs and vCPUs in your Azure subscription.

Related Azure quotas documentation: Azure subscription and service limits, quotas, and constraints.

Supported Azure regions

Cloudera supports the following Azure regions: Supported Azure regions.

- A single Azure environment registered in Cloudera corresponds to a single VNet located in a specific region, and all the resources deployed by Cloudera on Azure are deployed into that VNet.
- Deploying clusters into the region containing the ADLS Gen2 containers that you want to access for input and output data, speeds up the data access.Therefore, when selecting the region to use, you should consider where your data is located.
- Cloudera requires that the ADLS Gen2 storage location provided during environment registration is in the same region as the region selected for the environment.

If you need to use multiple regions, you need to register multiple environments, one per region.

Related Azure documentation: Azure geographies.

Set up an Azure Cloud Credential

You must create the Azure provisioning credential for Cloudera prior to registering an environment. The credential allows Cloudera to access and provision a set of resources in your Azure account.

When working with an Azure environment, you can use the app-based credential to authenticate your Azure account and obtain authorization to create resources on your behalf. The app-based credential allows you to manually configure the service principal created within the Azure Active Directory.

Instructions: Azure Credentials

Register an Azure Environment in Cloudera

Once you have met cloud provider requirements and have created the Azure provisioning credential for Cloudera, you may proceed to register an Azure Environment.

Important:

If you intend to use the Enable Private Network - create Cloudera Data Engineering with fully private Azure services (AKS, MySQL, etc.) option during environment creation, Cloudera recommends using one resource group per environment. You can accomplish this by selecting a pre-created resource group during Cloudera environment creation. This causes all Cloudera resources to be provisioned into the same resource group, allowing you to create multiple Cloudera Data Engineering services with private network enabled.

If you select creating new resource groups together with the environment, several separate resource groups are created. Since Azure restricts creating private link connections from private DNS zones in different resource groups to the same vnet, in this scenario you can only have a single active private Cloudera Data Engineering service for a given vnet.

As of Cloudera Data Engineering 1.16, private storage accounts are supported.

Instructions: Register an Azure environment

Cloudera Data Engineering Role Requirements

There are two Cloudera user roles associated with the Cloudera Data Engineering service: DEAdmin and DEUser. Any Cloudera user with the EnvironmentAdmin (or higher) access level must assign these roles to users who require access to the Cloudera Data Engineering console within their environment.

Furthermore, if you want to allow users to log in to provisioned workspaces and run workloads on them, this will need to be configured separately.

Set up to run kubectl

1. Go to the three dots on top right side of the Cloudera Data Engineering UI to see a dropdown menu.
2. Click on Download the Kube Config option and save it (For example, ~/.kube/cde-env1-kube-config).

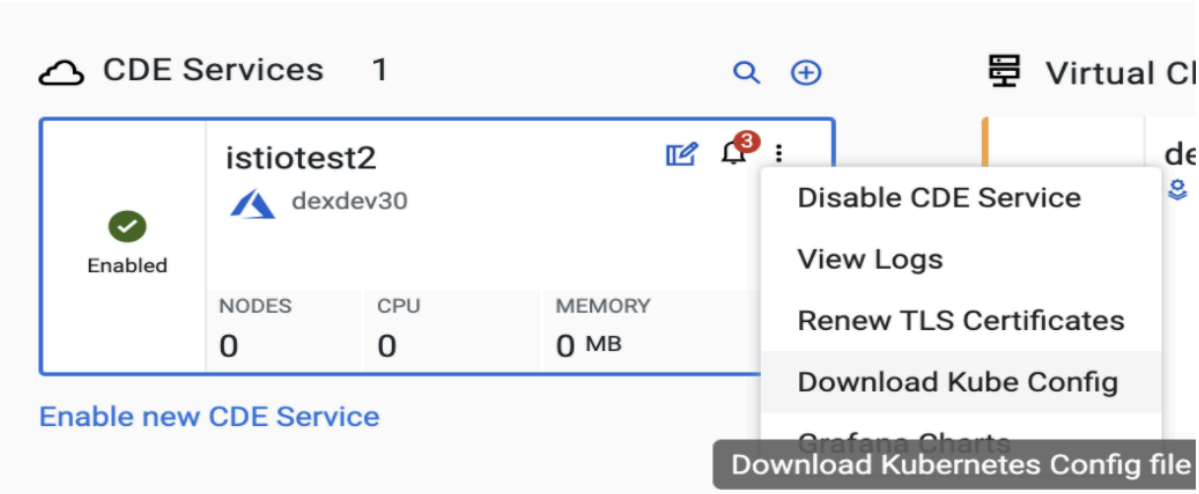

3. Run the following shell command.

```
$ export KUBECONFIG=~/.kube/cde-env1-kube-config
```

4. You should now be able to run Kubectl Commands

Browser Requirements

Supported browsers:

- Chrome
- Safari

Supporting Azure private storage

Cloudera Data Engineering stores all the metadata and application files in Azure Files. By default, Cloudera Data Engineering relies on Azure Kubernetes Service (AKS) to create a public Azure Storage Account in the AKS Node Resource Group. The provisioned Storage Account will be used only for Azure Files in Cloudera Data Engineering and will be deleted once the Cloudera Data Engineering service is deleted.

About this task

With Cloudera Data Engineering 1.16, the Azure Files used for storing Cloudera Data Engineering internal metadata can be overridden with custom private/public Azure Files.

Creating Private Azure files

Before you begin

Create a Private Azure Account. For more information, see Private Endpoints for Azure Storage below. Be sure to create a Private Endpoint for Files as target sub-resource since Cloudera Data Engineering only uses Azure Files. This means that blobs, tables, queues, and so on are not used.

Table 1: Sample creation commands

Action	Command
Entitlement	`Enable : DE_AZURE_PRIVATE_STORAGE for customer tenant Id.`
Export Variables	`RESOURCE_GROUP=TestCDEPrivateResourceGroup \ STORAGE_ACCOUNT_NAME=testcdeprivatestorageaccount \ PRIVATE_ENDPOINT_NAME=private-endpoint-${STORAGE_ACCOUNT_NAME} \ VNET_NAME=cde-vnet \ VNET_RESOURCE_GROUP=TestCDEPrivateResourceGroup \ DNS_ZONE_NAME= ${STORAGE_ACCOUNT_NAME}.privatelink.file.core.windows.net \ VNET_LINK_NAME=vnet-link-${STORAGE_ACCOUNT_NAME}`
Create Private Storage Account	`az storage account create --name ${STORAGE_ACCOUNT_NAME} --resource-group ${RESOURCE_GROUP} --allow-blob-public-access false --https true --public-network-access Disabled --sku Standard_LRS --location westus2`
Get Storage Account ID	`STORAGE_ACCOUNT_ID=$(az storage account show --resource-group ${RESOURCE_GROUP} --name ${STORAGE_ACCOUNT_NAME} | jq -r .id)`
Get Subnet ID	`SUBNET_ID=$(az network vnet subnet list --resource-group ${VNET_RESOURCE_GROUP} --vnet-name ${VNET_NAME} | jq -r '.[0].id')`
Create Private Endpoint	`az network private-endpoint create --connection-name myConnection --name ${PRIVATE_ENDPOINT_NAME} --private-connection-resource-id ${STORAGE_ACCOUNT_ID} --location westus2 --subnet ${SUBNET_ID} --resource-group ${RESOURCE_GROUP} --group-id file`
Get Private Endpoint IP	`PRIVATE_ENDPOINT_IP=$(az network private-endpoint show --name ${PRIVATE_ENDPOINT_NAME} --resource-group ${RESOURCE_GROUP} | jq -r '.customDnsConfigs | .[0].ipAddresses | .[0]')`
Create Private DNS Zone	`az network private-dns zone create --resource-group ${RESOURCE_GROUP} --name ${DNS_ZONE_NAME}`
Create Private DNS Zone record	`az network private-dns record-set a add-record --resource-group ${RESOURCE_GROUP} --zone-name ${DNS_ZONE_NAME} --record-set-name @ --ipv4-address ${PRIVATE_ENDPOINT_IP}`
Get VNet ID	`VNET_ID=$(az network vnet show --resource-group ${VNET_RESOURCE_GROUP} --name ${VNET_NAME} | jq -r .id)`
Link DNS Zone to VNet	`az network private-dns link vnet create --name ${VNET_LINK_NAME} --registration-enabled false --resource-group ${RESOURCE_GROUP} --virtual-network ${VNET_ID} --zone-name ${DNS_ZONE_NAME}`

Enable Cloudera Data Engineering Service for Azure

1. Navigate to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera Management Console.
2. In the Cloudera Data Engineering Environments column, click the plus icon at the top or the Enable new Cloudera Data Engineering link at the bottom to enable Cloudera Data Engineering for an environment.
3. Select an Azure environment.
4. Select the Override Azure Files Storage Server option.
5. Enter the Storage Account name and Resource group of the Storage Account.
6. For Azure Files FQDN, enter the Private Endpoint FQDN that you created in the prerequisites above.
7. Fill out the necessary details and click Create.

Using customer-managed key encryption

You can enable CMK encryption with Cloudera Data Engineering in Azure. Below are high-level instructions to enable CMK encryption. For detailed instructions, see Customer-managed keys for Azure Storage encryption linked below.

1. Create a key vault and grant corresponding permission to the service principle that is used to create Cloudera environment.
2. Associate the private storage account and key vault when enabling the CMK encryption.
3. In their job's code, set up ABFS authentication configurations and access the abfs:// files.

Related Information

Why are two resource groups created with AKS?
Azure Files
Private Endpoints for Azure Storage
Customer-managed keys for Azure storage encryption

Using AWS IAM restricted roles and policies for compute and Cloudera Data Engineering

AWS IAM write permissions are used by the Cloudera Data Engineering compute infrastructure to create and delete roles and instance profiles.

Some customers may not be willing to provide IAM write permission in the role's policy. Instead, customers can set up static pre-created roles andinstance profiles defined and used by the Cloudera Data Engineering compute infrastructure to provision clusters.

Note: The compute infrastructure is only able to use the pre-created roles and instance profile if the entitlement LIFTIE_USE_PRECREATED_IAM_RESOURCES is set for the tenant in use. The pre-created roles and instance profiles will continue to exist for the lifetime of the cluster.

The two main tasks for AWS IAM write permissions are the following:

1. Create roles and an instance profile.
2. Create restricted IAM policies for use by the compute infrastructure.

After the two tasks are completed, you may create a cross-account credential if needed.

See the following topics for the procedures for creating the roles and policies.

Create IAM roles and instance profile pair

This step describes the role and instance profile pair that you will create and attach to EKS master and worker instances at runtime. This step is required in customer environments where write permissions are not provided to Cloudera Data Engineering. The roles created here are used exclusively within the customer's account. This step needs to be performed in the user's AWS console.

Use the following CloudFormation template to create the following:

- IAM role called cdp-eks-master-role
- IAM role and Instance Profile pair called cdp-liftie-instance-profile

Important: For Cloudera Data Engineering 1.19.4 and above, when you use a Restricted Policy, you must use an additional permission of elasticfilesystem:TagResource in the efs-csi policy for the cdp-liftie-instance-profile role.

1. To apply the template, you need to provide values for the following parameters in the AWS console CloudFormation wizard:
   - Stack Name: Provide an appropriate name. (Example: compute-precreated-roles-and-instanceprofile)
   - TelemetryLoggingBucket: Name of the log bucket (just the name, not s3://) (Example : compute-logging-bucket)
   - TelemetryLoggingEnabled: Set it to true
   - TelemetryLoggingRootDir: Keep the default value (which is cluster-logs)
   - TelemetryKMSKeyARN: If the telemetry bucket is encrypted, give the KMS Key ARN. Default value is null.

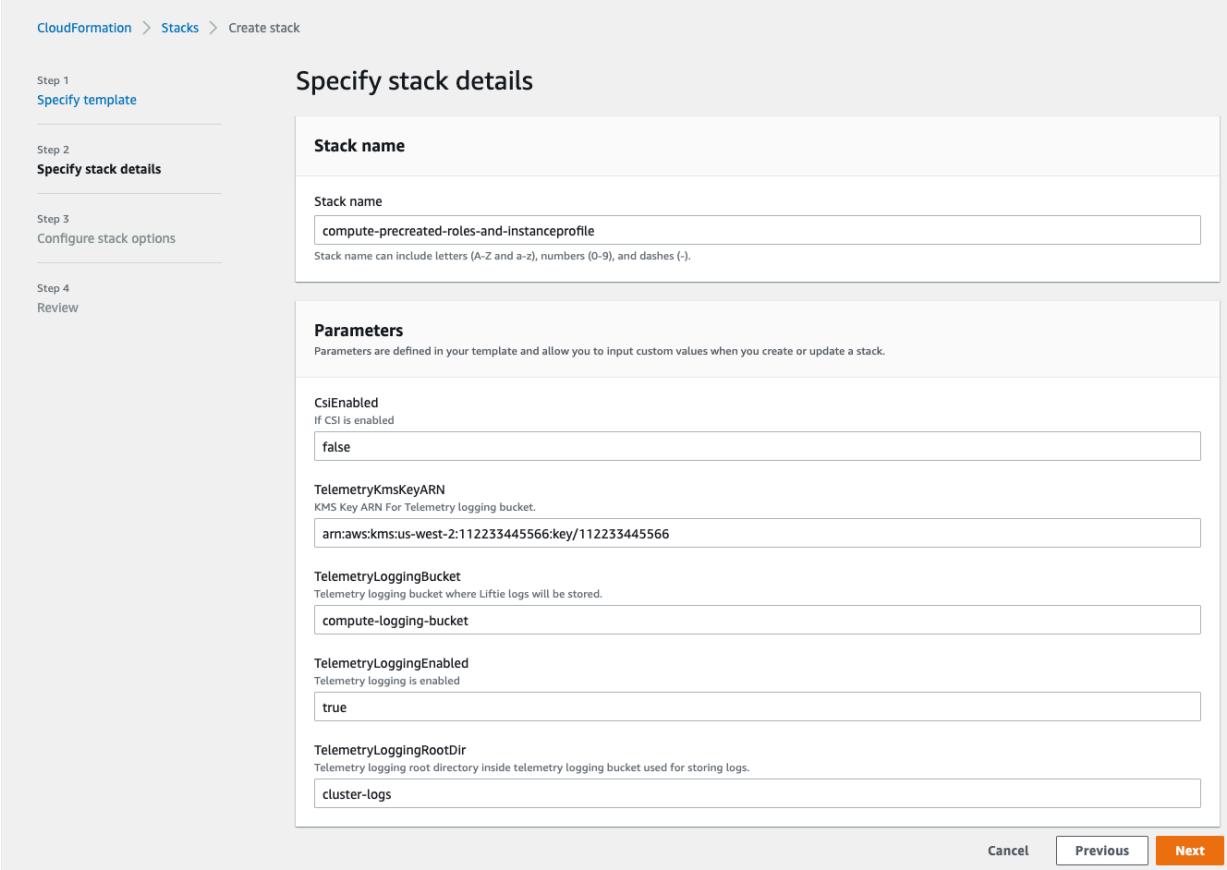

2. On the last page of the wizard, select the checkbox to allow creation of IAM resources with special names. If not selected, CloudFormation prepends the provided name with random prefixes to ensure uniqueness.

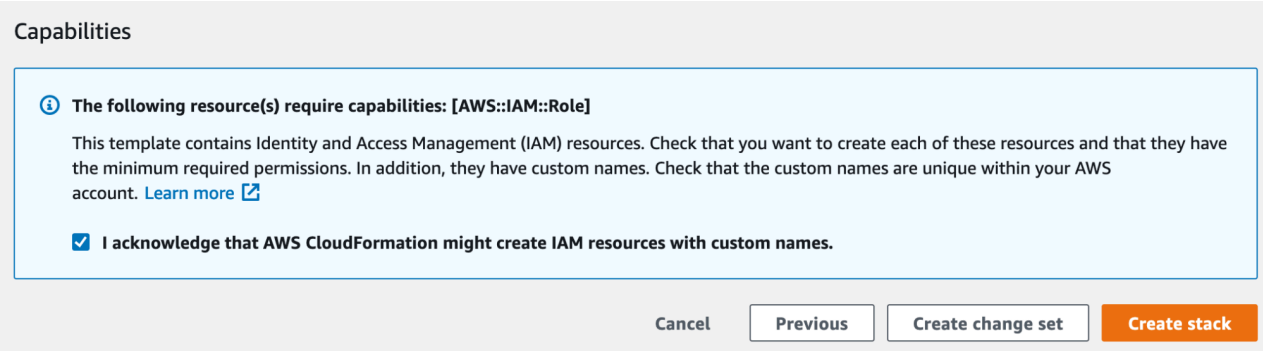

The result of this procedure resembles the following:

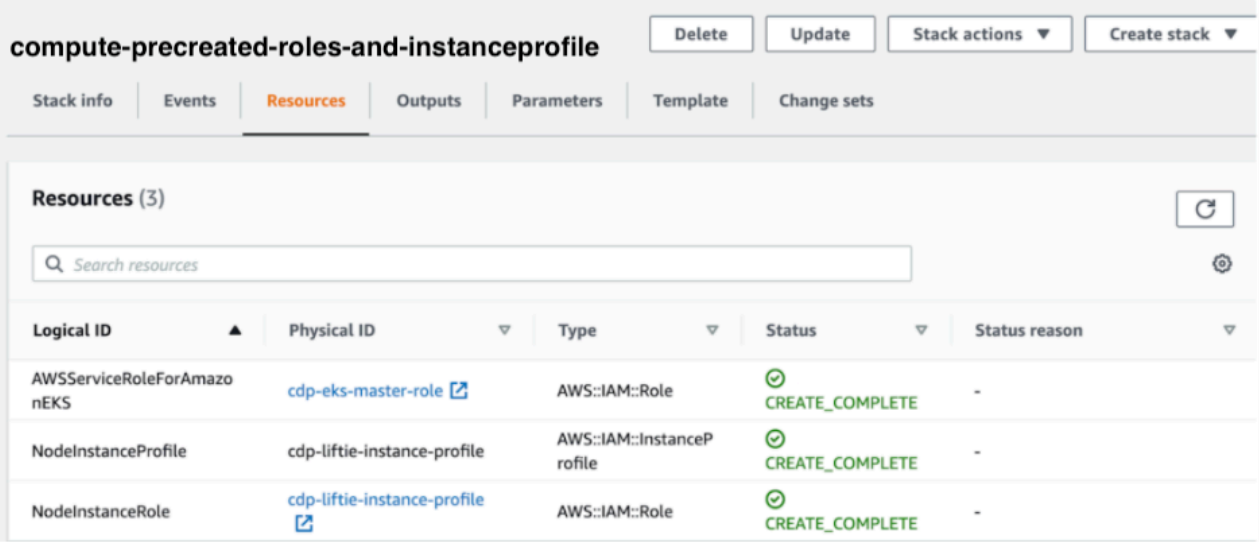

Use the following CloudFormation template for this process.

CloudFormation Template (format: YAML)

```
AWSTemplateFormatVersion: "2010-09-09"

Description: "Creates Liftie IAM resources"
Parameters:

 TelemetryLoggingEnabled:
   Description: Telemetry logging is enabled
   Type: String

 TelemetryLoggingBucket:
   Description: Telemetry logging bucket where Liftie logs will be stored.
   Type: String

 TelemetryKmsKeyARN:
   Description: KMS Key ARN For Telemetry logging bucket.
   Type: String
   Default: ""

 TelemetryLoggingRootDir:
   Description: Telemetry logging root directory inside telemetry logging bu
cket used for storing logs.
   Default: "cluster-logs"
   Type: String
```

```
Conditions:
 TelemetryLoggingEnabled:
   Fn::Equals:
     - {Ref: TelemetryLoggingEnabled}
     - true
 KMSKeyARNForTelemetryLoggingBucketIsEmpty: !Not [!Equals [!Ref TelemetryKms
KeyARN, ""]]

Resources:

 AWSServiceRoleForAmazonEKS:
   Type: AWS::IAM::Role
   Properties:
     AssumeRolePolicyDocument:
       Version: '2012-10-17'
       Statement:
         - Effect: Allow
           Principal:
             Service:
               - eks.amazonaws.com
           Action:
             - sts:AssumeRole
     ManagedPolicyArns:
       - arn:aws:iam::aws:policy/AmazonEKSServicePolicy
       - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
     RoleName: cdp-eks-master-role
 NodeInstanceRole:
   Type: AWS::IAM::Role
   Properties:
     AssumeRolePolicyDocument:
       Version: '2012-10-17'
       Statement:
         - Effect: Allow
           Principal:
             Service:
               - ec2.amazonaws.com
           Action:
             - sts:AssumeRole
     Path: "/"
     ManagedPolicyArns:
       - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
       - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
       - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
     RoleName: cdp-liftie-instance-profile
     Policies:
       - PolicyName: 'ssm-required'
         PolicyDocument:
           Version: '2012-10-17'
           Statement:
             - Effect: Allow
               Action:
                 - ssm:GetParameters
               Resource:
                 - "*"
       - PolicyName: 'cde-specific-permissions'
         PolicyDocument:
           Version: '2012-10-17'
           Statement:
             - Effect: Allow
               Action:
                 - cloudwatch:GetMetricData
               Resource:
                 - "*"
       - PolicyName: 'cluster-autoscaler'
```

```
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action:
                - autoscaling:DescribeAutoScalingGroups
                - autoscaling:DescribeAutoScalingInstances
                - autoscaling:DescribeLaunchConfigurations
                - autoscaling:DescribeScalingActivities
                - autoscaling:DescribeTags
                - ec2:DescribeImages
                - ec2:DescribeInstanceTypes
                - ec2:DescribeLaunchTemplateVersions
                - ec2:GetInstanceTypesFromInstanceRequirements
                - eks:DescribeNodegroup
                Resource:
                - "*"
              - Effect: Allow
                Action:
                  - autoscaling:SetDesiredCapacity
                  - autoscaling:TerminateInstanceInAutoScalingGroup
                Resource:
                  - "*"
                Condition:
                  StringEquals:
                    "aws:ResourceTag/k8s.io/cluster-autoscaler/enabled": "
true"
       - PolicyName: ebs-csi
         PolicyDocument:
           Version: 2012-10-17
           Statement:
             - Effect: Allow
               Action:
                 - ec2:CreateSnapshot
                 - ec2:AttachVolume
                 - ec2:DetachVolume
                 - ec2:ModifyVolume
                 - ec2:DescribeAvailabilityZones
                 - ec2:DescribeInstances
                 - ec2:DescribeSnapshots
                 - ec2:DescribeTags
                 - ec2:DescribeVolumes
                 - ec2:DescribeVolumesModifications
               Resource: "*"
             - Effect: Allow
               Action:
                 - ec2:CreateTags
               Resource:
                 - "arn:aws:ec2:*:*:volume/*"
                 - "arn:aws:ec2:*:*:snapshot/*"
               Condition:
                 StringEquals:
                   "ec2:CreateAction":
                     - CreateVolume
                     - CreateSnapshot
             - Effect: Allow
               Action:
                 - ec2:DeleteTags
               Resource:
                 - "arn:aws:ec2:*:*:volume/*"
                 - "arn:aws:ec2:*:*:snapshot/*"
             - Effect: Allow
               Action:
                 - ec2:CreateVolume
```

```
                Resource: "*"
                Condition:
                  StringLike:
                    "aws:RequestTag/ebs.csi.aws.com/cluster": "true"
              - Effect: Allow
                Action:
                  - ec2:CreateVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "aws:RequestTag/CSIVolumeName": "*"
              - Effect: Allow
                Action:
                  - ec2:CreateVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "aws:RequestTag/kubernetes.io/cluster/*": "owned"
              - Effect: Allow
                Action:
                  - ec2:DeleteVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
              - Effect: Allow
                Action:
                  - ec2:DeleteVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "ec2:ResourceTag/CSIVolumeName": "*"
              - Effect: Allow
                Action:
                  - ec2:DeleteVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "ec2:ResourceTag/kubernetes.io/created-for/pvc/name": "*"
              - Effect: Allow
                Action:
                  - ec2:DeleteSnapshot
                Resource: "*"
                Condition:
                  StringLike:
                    "ec2:ResourceTag/CSIVolumeSnapshotName": "*"
              - Effect: Allow
                Action:
                  - ec2:DeleteSnapshot
                Resource: "*"
                Condition:
                  StringLike:
                    "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
          - PolicyName: efs-csi
            PolicyDocument:
              Version: 2012-10-17
              Statement:
              - Effect: Allow
                Action:
                  - elasticfilesystem:DescribeAccessPoints
                  - elasticfilesystem:DescribeFileSystems
                  - elasticfilesystem:DescribeMountTargets
                  - elasticfilesystem:TagResource
                Resource: "*"
```

```
              - Effect: Allow
                Action:
                  - elasticfilesystem:CreateAccessPoint
                Resource: "*"
                Condition:
                  StringLike:
                    "aws:RequestTag/efs.csi.aws.com/cluster": "true"
              - Effect: Allow
                Action:
                  - elasticfilesystem:DeleteAccessPoint
                Resource: "*"
                Condition:
                  StringEquals:
                    "aws:ResourceTag/efs.csi.aws.com/cluster": "true"
        - !If
          - TelemetryLoggingEnabled
          - PolicyName: telemetry-s3-list-bucket
            PolicyDocument:
              Version: 2012-10-17
              Statement:
                - Effect: Allow
                  Action:
                    - 's3:ListBucket'
                  Resource:
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}'
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}/${Telemetr
yLoggingRootDir}/*'
          - !Ref 'AWS::NoValue'
        - !If
          - TelemetryLoggingEnabled
          - PolicyName: telemetry-s3-read-write
            PolicyDocument:
              Version: 2012-10-17
              Statement:
                - Effect: Allow
                  Action:
                    - 's3:*Object'
                    - 's3:AbortMultipartUpload'
                  Resource:
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}'
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}/${Telemet
ryLoggingRootDir}/*'
          - !Ref 'AWS::NoValue'
        - !If
          - KMSKeyARNForTelemetryLoggingBucketIsEmpty
          - PolicyName: s3-kms-read-write-policy
            PolicyDocument:
              Version: 2012-10-17
              Statement:
                - Effect: Allow
                  Action:
                    - 'kms:Decrypt'
                    - 'kms:GenerateDataKey'
                  Resource:
                    - !Sub ${TelemetryKmsKeyARN}
          - !Ref 'AWS::NoValue'
        - PolicyName: 'calico-cni'
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action:
                  - ec2:ModifyInstanceAttribute
                Resource:
```

```
                  - "*"
                Condition:
                   StringEquals:
                     "ec2:Attribute": "SourceDestCheck"
NodeInstanceProfile:
  Type: AWS::IAM::InstanceProfile
  Properties:
    Path: "/"
    InstanceProfileName: cdp-liftie-instance-profile
    Roles:
      - !Ref NodeInstanceRole
```

Create role and policy used to deploy Cloudera environments for Cloudera Data Engineering

The Cloudera Data Engineering control plane requires a role and policies to create Cloudera environments. In this step, you create a common policy for creating environments, as well as a policy that is specific to Cloudera Data Engineering environments.

The following two policies are created in this step:

- Compute infrastructure restricted IAM policy - A common policy for all data services deployed on Cloudera.
- Cloudera Data Engineering restricted IAM policy - A policy with additional permissions for Cloudera Data Engineering.

There are two options for the timing of attaching the role: during environment creation, or prior to enabling the Cloudera Data Engineering data service.

Option #1: During environment creation

The Cloudbreak environment creation UI should be set up as shown here:

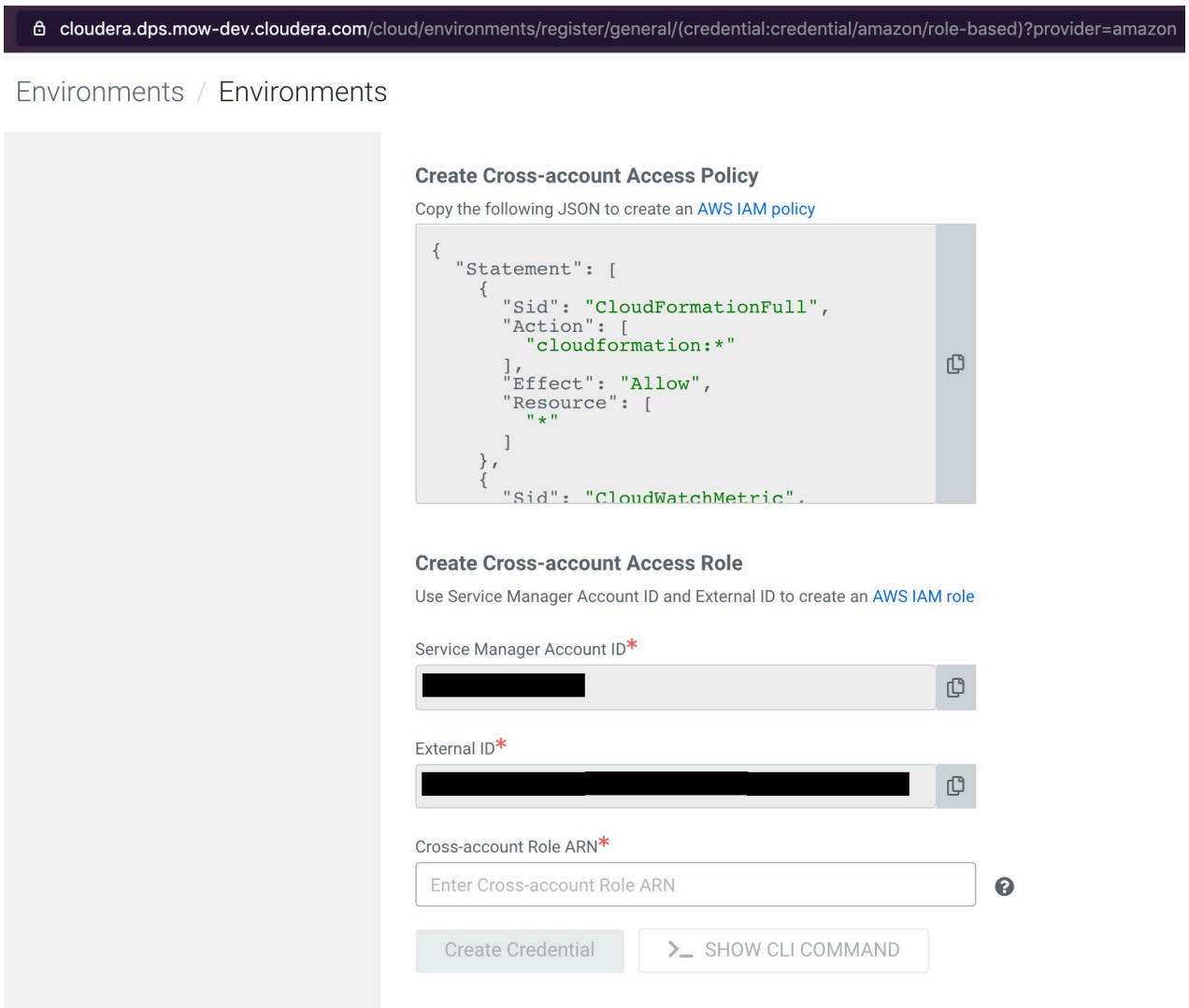

Note:

- For the AWS IAM policy mentioned in the “Create Cross-account Access Policy” section, use the Compute infrastructure Restricted IAM and Cloudera Data Engineering Restricted IAM policies below and create as new policies in AWS IAM. There may be one or more restricted policies already attached to the cross-account role, in addition to the Compute infrastructure and Cloudera Data Engineering restricted policies. For example, there may also be a Cloudera Data Hub restricted policy.
- For the “Create Cross-account Access Role” section, create the cross-account role as instructed (or update the role if one already exists) and attach the newly created Compute infrastructure Restricted IAM policy and Cloudera Data Engineering Restricted IAM policy. Finally, update the cross-account role to use it.

Option #2: Prior to enabling Cloudera Data Engineering data service

If the Cloudbreak environment has already been created, you can create and attach the Compute infrastructure Restricted IAM policy and Cloudera Data Engineering restricted IAM policy to the existing cross-account role associated with the environment.

To view the existing cross-account role, in the Environments section of the Cloudera Management Console, on the Summary tab, see Credentials.

Note: There may be one or more restricted policies already attached to the cross-account role, in addition to the Compute infrastructure and Cloudera Data Engineering restricted policies. For example, there might be a Cloudera Data Hub restricted policy. These should be left in place.

Compute (Liftie) Restricted IAM policy

Replace the following placeholders in the JSON file:

- [YOUR-ACCOUNT-ID] with your account ID in use.
- [YOUR-IAM-ROLE-NAME] with the IAM restricted role associated with this policy.
- [YOUR-SUBNET-ARN-*] supplied during the Cloudbreak Environment(s) creation. Note: Please provide all the subnets present in all the Cloudbreak Environment(s) that you intend to use it for the experience. If at any point a new Cloudbreak Environment is created or an existing one is updated for subnets, the same should be updated here.
- [YOUR-IDBROKER-ROLE-NAME] with the ID Broker Role name in use.
- [YOUR-LOG-ROLE-NAME] with the Log Role name in use.
- [YOUR-KMS-CUSTOMER-MANAGED-KEY-ARN] with KMS key ARN.
- [YOUR-ACCOUNT-REGION] with the AWS region.

```
{
  "Version": "2012-10-17",
  "Id": "ComputePolicy_v12",
  "Statement": [
    {
      "Sid": "SimulatePrincipalPolicy",
      "Effect": "Allow",
      "Action": [
        "iam:SimulatePrincipalPolicy"
      ],
      "Resource": [
        "arn:{{ .ARNPartition }}:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-IAM-ROLE-
NAME]"
      ]
    },
    {
      "Sid": "RestrictedPermissionsViaClouderaRequestTag",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:CreateChangeSet",
        "ec2:createTags",
        "eks:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/Cloudera-Resource-Name": [
            "crn:{{ .CRNPartition }}:*"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedPermissionsViaClouderaResourceTag",
      "Effect": "Allow",
      "Action": [
        "autoscaling:DeleteTags",
        "autoscaling:DetachInstances",
        "autoscaling:ResumeProcesses",
        "autoscaling:SetDesiredCapacity",
        "autoscaling:SuspendProcesses",
        "autoscaling:TerminateInstanceInAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeChangeSet",
```

```
        "cloudformation:DescribeStacks",
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResource",
        "cloudformation:DescribeStackResources",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListStacks",
        "cloudwatch:deleteAlarms",
        "cloudwatch:putMetricAlarm",
        "ec2:AttachVolume",
        "ec2:CreateNetworkInterface",
        "ec2:CreateVolume",
        "ec2:DeleteVolume",
        "ec2:RunInstances",
        "eks:DescribeUpdate",
        "eks:ListUpdates",
        "eks:UpdateClusterConfig",
        "eks:UpdateClusterVersion",
        "iam:GetRolePolicy",
        "iam:ListInstanceProfiles",
        "iam:ListRoleTags",
        "iam:RemoveRoleFromInstanceProfile",
        "iam:TagRole",
        "iam:UntagRole",
        "logs:DescribeLogStreams",
        "logs:FilterLogEvents"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/Cloudera-Resource-Name": [
            "crn:{{ .CRNPartition }}:*"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedPermissionsViaCloudFormation",
      "Effect": "Allow",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:CreateLaunchConfiguration",
        "autoscaling:CreateOrUpdateTags",
        "autoscaling:DeleteAutoScalingGroup",
        "autoscaling:DeleteLaunchConfiguration",
        "autoscaling:DescribeAutoScalingInstances",
        "autoscaling:DescribeLaunchConfigurations",
        "autoscaling:DescribeScalingActivities",
        "autoscaling:DescribeScheduledActions",
        "autoscaling:DescribeTags",
        "dynamodb:DescribeTable",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:CreateLaunchTemplate",
        "ec2:CreateSecurityGroup",
        "ec2:DeleteLaunchTemplate",
        "ec2:DeletePlacementGroup",
        "ec2:DeleteSecurityGroup",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeImages",
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeLaunchTemplateVersions",
```

```
        "ec2:DescribeLaunchTemplates",
        "ec2:DescribePlacementGroups",
        "ec2:DescribeRegions",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVolumes",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "eks:CreateCluster",
        "eks:DeleteCluster"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedEC2PermissionsViaClouderaResourceTag",
      "Effect": "Allow",
      "Action": [
        "ec2:RebootInstances",
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ForAnyValue:StringLike": {
          "ec2:ResourceTag/Cloudera-Resource-Name": [
            "crn:{{ .CRNPartition }}:*"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedIamPermissionsToClouderaResources",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:{{ .ARNPartition }}:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-IDBROKER-
ROLE-NAME]",
        "arn:{{ .ARNPartition }}:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-LOG-ROLE-
NAME]",
        "arn:{{ .ARNPartition }}:iam::[YOUR-ACCOUNT-ID]:role/liftie-*-eks-se
rvice-role",
        "arn:{{ .ARNPartition }}:iam::[YOUR-ACCOUNT-ID]:role/liftie-*-eks-w
orker-nodes",
        "arn:{{ .ARNPartition }}:iam::[YOUR-ACCOUNT-ID]:role/cdp-eks-master-
role",
        "arn:{{ .ARNPartition }}:iam::[YOUR-ACCOUNT-ID]:role/cdp-liftie-ins
tance-profile"
      ]
    },
    {
      "Sid": "RestrictedKMSPermissionsUsingCustomerProvidedKey",
      "Effect": "Allow",
```

```
      "Action": [
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*"
      ],
      "Resource": [
        "[YOUR-KMS-CUSTOMER-MANAGED-KEY-ARN]"
      ]
    },
    {
      "Sid": "AllowCreateDeleteTagsForSubnets",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags",
        "ec2:DeleteTags"
      ],
      "Resource": [
        "arn:{{ .ARNPartition }}:ec2:[YOUR-SUBNET-REGION]:[YOUR-ACCOUNT-ID]:
subnet/*"
      ]
    },
    {
      "Sid": "ModifyInstanceAttribute",
      "Effect": "Allow",
      "Action": [
        "ec2:ModifyInstanceAttribute"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "ec2:Attribute": "SourceDestCheck"
        }
      }
    },
    {
      "Sid": "OtherPermissions",
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateLaunchTemplateVersion",
        "ec2:CreatePlacementGroup",
        "ec2:DeleteKeyPair",
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInstanceTypes",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcs",
        "ec2:ImportKeyPair",
        "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
        "ec2:GetInstanceTypesFromInstanceRequirements",
        "eks:DescribeCluster",
        "elasticloadbalancing:DescribeLoadBalancers",
        "iam:GetRole",
        "iam:ListRoles",
        "iam:GetInstanceProfile"
      ],
```

```
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowSsmParams",
      "Effect": "Allow",
      "Action": [
        "ssm:DescribeParameters",
        "ssm:GetParameter",
        "ssm:GetParameters",
        "ssm:GetParameterHistory",
        "ssm:GetParametersByPath"
      ],
      "Resource": [
        "arn:aws:ssm:*:*:parameter/aws/service/eks/optimized-ami/*"
      ]
    },
    {
      "Sid": "CfDeny",
      "Effect": "Deny",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ForAnyValue:StringLike": {
          "cloudformation:ImportResourceTypes": [
            "*"
          ]
        }
      }
    },
    {
      "Sid": "ForAutoscalingLinkedRole",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:{{ .ARNPartition }}:iam::[YOUR-ACCOUNT-ID]:role/aws-service-ro
le/autoscaling-plans.amazonaws.com/AWSServiceRoleForAutoScalingPlans_EC2Auto
Scaling"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "autoscaling-plans.amazonaws.com"
        }
      }
    },
    {
      "Sid": "ForEksLinkedRole",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:{{ .ARNPartition }}:iam::[YOUR-ACCOUNT-ID]:role/aws-service-
role/eks.amazonaws.com/AWSServiceRoleForEKS"
      ],
      "Condition": {
        "StringLike": {
```

```
            "iam:AWSServiceName": "eks.amazonaws.com"
          }
        }
      }
    ]
}
```

Supporting Customer Managed CMKs

Along with providing the KMS Customer Managed Customer Master Key (CMK) for volume encryption in the policy section with Sid: RestrictedKMSPermissionsUsingCustomerProvidedKey, you need to verify that the policy for the Customer Managed Customer Master Key (CMK) at KMS (this is not an IAM policy) has the following permission blocks defined for AWSServiceRoleForAutoScaling:

```
        {
    "Sid": "AllowAutoscalingAndCDPCrossAccountRoleUseOfTheCMK",
    "Effect": "Allow",
    "Principal":
    {
        "AWS":
        [
            "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/CDP-CROSSACCOUNT-ROLE",
            "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/autoscali
ng.amazonaws.com/AWSServiceRoleForAutoScaling"
        ]
    },
    "Action":
    [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "AllowAutoscalingAndCDPCrossAccountRoleToCreateGrantsOfTheCMK",
    "Effect": "Allow",
    "Principal":
    {
        "AWS":
        [
            "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/CDP-CROSSACCOUNT-ROLE",
            "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/autoscali
ng.amazonaws.com/AWSServiceRoleForAutoScaling"
        ]
    },
    "Action":
    [
        "kms:CreateGrant"
    ],
    "Resource": "*",
    "Condition":
    {
        "Bool":
        {
            "kms:GrantIsForAWSResource": "true"
        }
    }
},
{
```

```
    "Sid": "AllowCreateGrantToLiftieCluster",
    "Effect": "Allow",
    "Principal":
    {
        "AWS": "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/CDP-CROSSACCOUNT-ROLE"
    },
    "Action": "kms:CreateGrant",
    "Resource": "*",
    "Condition":
    {
        "StringEquals":
        {
            "aws:CalledViaFirst": "cloudformation.amazonaws.com"
        },
        "ForAllValues:StringEquals":
        {
            "kms:GrantOperations":
            [
                "Encrypt",
                "Decrypt"
            ]
        }
    }
},
{
    "Sid": "AllowEKSAccessToEBS",
    "Effect": "Allow",
    "Principal":
    {
        "AWS": "*"
    },
    "Action":
    [
        "kms:CreateGrant",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition":
    {
        "StringEquals":
        {
            "kms:CallerAccount": "[YOUR-ACCOUNT-ID]",
            "kms:viaService": "ec2.[YOUR-ACCOUNT-REGION].amazonaws.com"
        }
    }
}
```

After the policy is attached, the KMS service page will show the CMK as having the policy attached, similar to this screenshot:

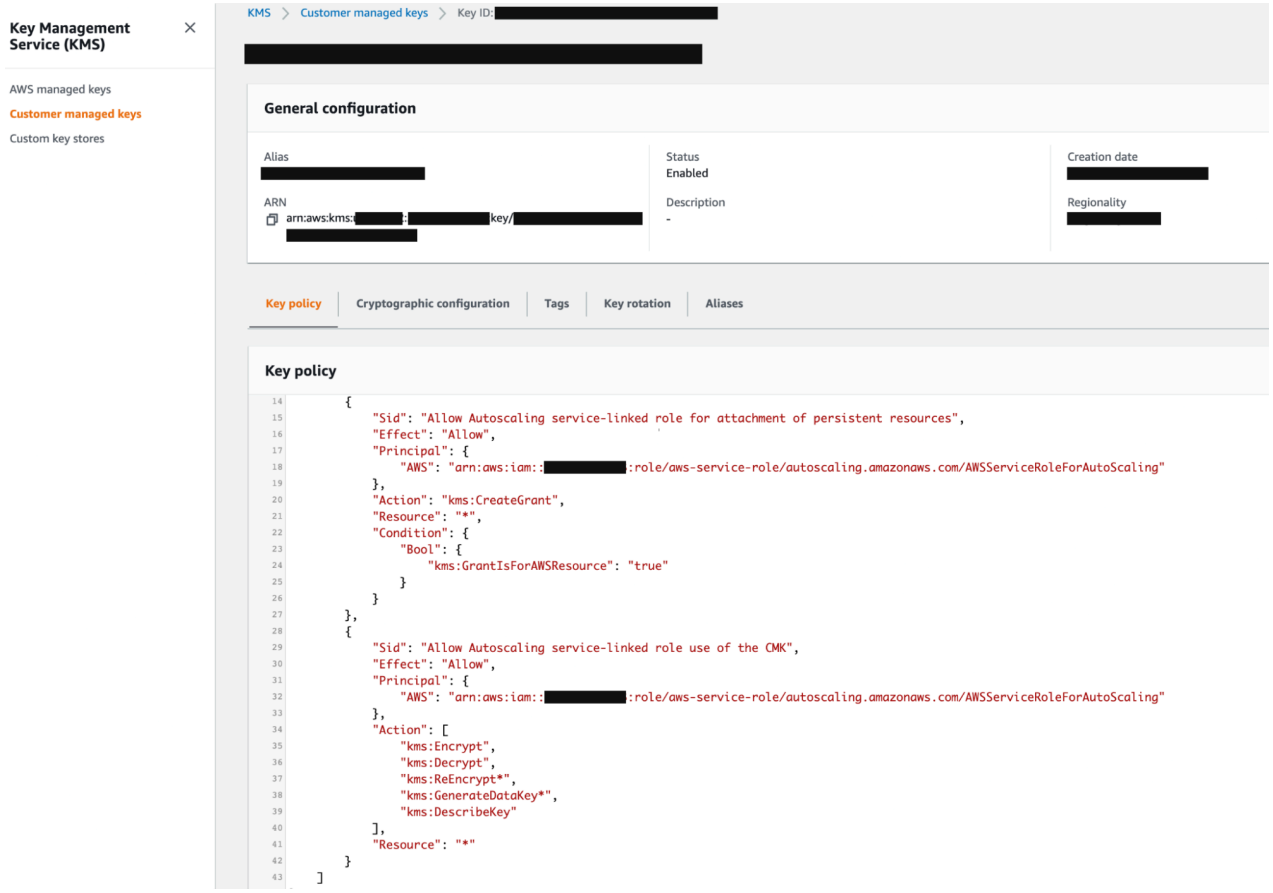

Cloudera Data Engineering restricted IAM policy

```
{
  "Id": "CDEPolicy_v2",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ElasticFileSystem",
      "Action": [
        "elasticfilesystem:CreateMountTarget",
        "elasticfilesystem:DescribeFileSystems",
        "elasticfilesystem:TagResource",
        "elasticfilesystem:ClientMount"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "ElasticFileSystemRequest",
      "Action": [
        "elasticfilesystem:CreateFileSystem",
        "elasticfilesystem:CreateTags"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Condition": {
```

```
        "StringLike": {
          "aws:RequestTag/Cloudera-Resource-Name": [
            "crn:cdp:de:*"
          ]
        }
      }
    },
    {
      "Sid": "ElasticFileSystemResource",
      "Action": [
        "elasticfilesystem:DescribeFileSystemPolicy",
        "elasticfilesystem:DeleteFileSystem",
        "elasticfilesystem:DescribeMountTargets",
        "elasticfilesystem:DeleteMountTarget",
        "elasticfilesystem:PutFileSystemPolicy"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/Cloudera-Resource-Name": [
            "crn:cdp:de:*"
          ]
        }
      }
    },
    {
      "Sid": "CloudWatch",
      "Action": [
        "cloudwatch:GetMetricData"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "ElasticLoadBalancing",
      "Action": [
        "elasticloadbalancing:DescribeTags"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "RelationalDatabaseServiceRequest",
      "Action": [
        "rds:CreateDBCluster",
        "rds:CreateDBSubnetGroup",
        "rds:AddTagsToResource",
        "rds:CreateDBInstance"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "aws:RequestTag/Cloudera-Resource-Name": [
            "crn:cdp:de:*"
```

```
            ]
          }
        }
      },
      {
        "Sid": "RelationalDatabaseServiceResource",
        "Action": [
          "rds:DeleteDBSubnetGroup",
          "rds:DescribeDBInstances",
          "rds:ModifyDBInstance",
          "rds:DeleteDBInstance"
        ],
        "Effect": "Allow",
        "Resource": [
          "*"
        ],
        "Condition": {
          "StringLike": {
            "aws:ResourceTag/Cloudera-Resource-Name": [
              "crn:cdp:de:*"
            ]
          }
        }
      },
      {
        "Sid": "RelationalDatabaseService",
        "Action": [
          "rds:DescribeDBEngineVersions"
        ],
        "Effect": "Allow",
        "Resource": [
          "*"
        ]
      }
    ]
}
```

Cloudera Data Hub restricted policy

To add the Cloudera Data Hub restricted policy, copy the following data in the Create Cross-account Access Policy field:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteTags",
                "ec2:AssociateAddress",
                "ec2:StartInstances",
                "ec2:StopInstances",
                "ec2:AttachVolume",
                "ec2:DescribeAddresses",
                "ec2:TerminateInstances",
                "ec2:DeleteSecurityGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/Cloudera-Resource-Name": [
                        "crn:cdp:*"
```

```
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:DeleteStack",
                "autoscaling:SuspendProcesses",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:ResumeProcesses",
                "autoscaling:DetachInstances",
                "autoscaling:DeleteAutoScalingGroup",
                "rds:StopDBInstance",
                "rds:StartDBInstance"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/Cloudera-Resource-Name": [
                        "crn:cdp:*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:GetTemplate",
                "ec2:CreateTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/Cloudera-Resource-Name": [
                        "crn:cdp:*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteVolume",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteKeyPair",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeImages",
                "ec2:DeleteLaunchTemplate",
                "ec2:DescribeVolumes",
                "ec2:CreateVolume",
                "ec2:DescribeInstances",
                "ec2:DescribeRegions",
                "ec2:DescribeInstanceTypeOfferings",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs",
```

```
                "ec2:DescribeInternetGateways",
                "ec2:DescribeVpcEndpoints",
                "ec2:describeAddresses",
                "ec2:DescribeNatGateways",
                "ec2:DescribeVpcEndpointServices",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:CreatePlacementGroup",
                "ec2:DescribePlacementGroups",
                "ec2:ImportKeyPair",
                "ec2:DescribeLaunchTemplates",
                "ec2:CreateLaunchTemplate",
                "ec2:RunInstances",
                "ec2:DescribeAccountAttributes",
                "sts:DecodeAuthorizationMessage",
                "cloudformation:DescribeStacks",
                "dynamodb:DeleteTable",
                "dynamodb:DescribeTable",
                "iam:ListInstanceProfiles",
                "iam:ListRoles",
                "dynamodb:ListTables",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DescribeAlarms",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "s3:GetBucketLocation",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStackResource",
                "cloudformation:ListStackResources",
                "cloudformation:UpdateStack",
                "cloudformation:GetTemplate",
                "iam:GetInstanceProfile",
                "iam:SimulatePrincipalPolicy",
                "iam:GetRole",
                "rds:AddTagsToResource",
                "rds:CreateDBInstance",
                "rds:CreateDBSubnetGroup",
                "rds:DeleteDBInstance",
                "rds:DeleteDBSubnetGroup",
                "rds:ListTagsForResource",
                "rds:RemoveTagsFromResource",
                "rds:CreateDBParameterGroup",
                "rds:DeleteDBParameterGroup",
                "rds:DescribeEngineDefaultParameters",
                "rds:ModifyDBParameterGroup",
                "rds:DescribeDBParameters",
                "rds:DescribeDBParameterGroups",
```

```
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeDBInstances",
                "rds:ModifyDBInstance",
                "rds:DescribeCertificates",
                "kms:ListKeys",
                "kms:ListAliases",
                "ec2:ModifyInstanceAttribute",
                "ec2:CreateLaunchTemplateVersion"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-IDBROKER-ROLE-NA
ME]"
            ]
        },
        {
            "Sid": "IdentityAccessManagementLimited",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/*"
            ]
        }
    ]
}
```

Cloudera Data Engineering 1.23.0

Cloudera Data Engineering Release Notes

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What's new in Cloudera Data Engineering Public Cloud? 5

November 12, 2024 5
August 27, 2024 6
July 23, 2024 7
June 24, 2024 8
May 31, 2024 8
May 22, 2024 8
March 27, 2024 9
March 20, 2024 9
February 27, 2024 9
December 11, 2023 11
September 14, 2023 11
June 29, 2023 12
May 18, 2023 12
March 30, 2023 13
January 19, 2023 14
November 23, 2022 14
October 12, 2022 15
September 26, 2022 16
July 20, 2022 16
June 30, 2022 17
April 27, 2022 18
February 09, 2022 18
December 21, 2021 19
November 9, 2021 20
Older releases 20
October 18, 2021 20
August 30, 2021 21
August 2, 2021 22
June 23, 2021 22
May 20, 2021 23
April 7, 2021 24
March 9, 2021 24
February 4, 2021 24
December 21, 2020 25
November 9, 2020 26
September 21, 2020 26
July 30, 2020 27

Cloudera Data Engineering fix for CVE-2021-44228 27

Known issues and limitations in Cloudera Data Engineering Public Cloud 27

General known issues with Cloudera Data Engineering 27
Technical service bulletins 39
Limitations 41

Cloudera Data Engineering Runtime end of support.....41

Compatibility for Cloudera Data Engineering and Runtime components.....42

Using the Cloudera Runtime Maven repository for Cloudera Data Engineering.....44

What's new in Cloudera Data Engineering Public Cloud?

This section lists major features and updates for the Cloudera Data Engineering (CDE) service in CDP Public Cloud.

November 12, 2024

This release (1.23.0) of the Cloudera Data Engineering service on Cloudera Public Cloud introduces the following changes.

External IDE connectivity through Spark Connect-based sessions (Technical Preview)

Cloudera Data Engineering supports Spark Connect sessions, which are Cloudera Data Engineering sessions that expose the Spark Connect interface. Spark Connect sessions allow you to run Spark commands from any remote Python environment. For more information, see External IDE connectivity through Spark Connect-based sessions.

In-place upgrade enhancements

In-place upgrade enhancements have been documented to improve handling Airflow version 2.9 and Python version 3.11 changes during the upgrade. For more information, see In-place upgrade with Airflow Operators and Libraries.

Backup-restore-based upgrade enhancements

Backup-restore-based upgrade enhancements have been implemented.

Airflow version upgrade to 2.9

The Airflow version that Cloudera Data Engineering uses is upgraded to 2.9. For more information, see:

- Apache Airflow 2.9 Release Notes
- Compatibility for Cloudera Data Engineering and Runtime components

Python 3.11 support for Apache Airflow

The Python version that Cloudera Data Engineering uses is upgraded to 3.11. For more information, see Compatibility for Cloudera Data Engineering and Runtime components.

Iceberg version upgrade to 1.5

The Iceberg version that Cloudera Data Engineering uses is upgraded to 1.5.2. For more information, see: Compatibility for Cloudera Data Engineering and Runtime components.

Kubernetes version upgrade to 1.30

The Kubernetes version that Cloudera Data Engineering uses is upgraded to 1.30. For more information, see Compatibility for Cloudera Data Engineering and Runtime components.

Deprecation of Cloudera Data Warehouse Operator

Starting from Cloudera Data Engineering service version 1.23.0, Cloudera Data Warehouse Operator is deprecated. Cloudera recommends you to use SQL Operator in place of Cloudera Data Warehouse Operator. For more information, see Creating a connection to Cloudera Data Warehouse for Cloudera Data Warehouse Operator.

Active node count

The active node count has been removed from the following Cloudera Data Engineering UI pages:

- Administrator Overview
- Administrator Overview Service Details

For real-time node count information, check your Cloud Service Provider's website. For more information, see Checking the node count on your Cloud Service provider's website.

Fixed issues

DEX-9617: Cloudera Data Engineering Jobs Page-Airflow UI: Audit Log tab does not allow further navigation

The fix allows you to view the Airflow DAG run audit logs.

August 27, 2024

This release (1.22.0-h1) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

This release does not contain new features, but includes the following fixes:

DEX-13103: Fix dex-runtime-python-builder to use python 3.8 even for Spark 3.3+

Apache Spark now comes with Python 3.8 installed for versions 3.3 and 3.5.

DEX-14027: Spark 3.5.1 on RAZ; few jobs are failing with error 'org.apache.hadoop.fs.s3a.impl.InstantiationIOException'

Jobs running on RAZ-enabled clusters on Apache Spark 3.5.1 failed with the org.apache.hadoop.fs.s3a.impl.InstantiationIOException error.

DEX-14037: Restore tries to create deleted default VC

When you performed the CDE jobs backup and restore operation, CDE tried to restore a default VC as well that had been deleted before the backup was created. This has been corrected, and now, only those VCs are restored that are included in the Backup archive.

DEX-14231: Allow setting Spark configs to empty & multiple '=' separators values in the UI

You can use the empty value and multiple '=' separator values in the UI for Spark configurations in the create-job and VC-level Spark configurations.

DEX-14313: CDE Backup/Restore Upgrade Does Not Update K8 version

When you performed the CDE jobs backup and restore operation, during the upgrade, the Kubernetes version was not upgraded.

DEX-14412: Runtime API sometimes crashes due to panic: interface conversion: interface {} is cache.DeletedFinalStateUnknown, not *v1.Event

The Runtime API sometimes crashed with the panic: interface conversion: interface {} is cache.DeletedFinalStateUnknown, not *v1.Event error.

DEX-14424: TGT Secret name validation and sanitisation incorrect

The TGT Kerberos tickets are saved as a secret with a name that is based on the CDP username. An issue related to the validation and sanitization of the CDP usernames has been fixed.

DEX-14556: Unhide 1.22 CDP CLI changes related to custom private DNS zone

The CDP CLI commands related to private DNS zones have been made available.

July 23, 2024

This release (1.22.0) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

AWS Graviton support (GA)

AWS Graviton is a general purpose, ARM-based processor family. From the AWS Graviton family, CDE supports AWS Graviton 3. AWS Graviton delivers currently the best price performance for cloud workloads running in AWS EC2. With AWS Graviton, you can optimize costs and achieve better performance.

For more information, see AWS Graviton instances in CDE.

Support for in-place upgrade with private networking

If you use services that run on a private network, (for example, Private EKS), your CDE service is eligible for an in-place upgrade.

For more information, see In-place upgrade with Apache Airflow Operators and Libraries

Kubernetes version upgrade to 1.28

The Kubernetes version that CDE uses is upgraded to 1.28. For more information, see Compatibility for Cloudera Data Engineering and Runtime components.

Iceberg version upgrade to 1.4.3

The Iceberg version that CDE uses is upgraded to 1.4.3. For more information, see Compatibility for Cloudera Data Engineering and Runtime components.

Support for Spark version 3.5.1

CDE supports Spark version 3.5.1. For more information, see Compatibility for Cloudera Data Engineering and Runtime components.

Virtual Cluster (VC)-level Spark configurations (Technical Preview)

With the VC-level Spark configurations option, you can create and update Spark configurations that, by default, apply to all Spark jobs that are run in a VC.

For more information, see Managing Virtual Cluster-level Spark configurations

Azure private DNS zones (Technical Preview)

With the Azure private DNS zones feature, you can optimize costs by leveraging existing Azure private DNS zone resources. When creating the CDE service, you can use an Azure private DNS zone for the Azure Kubernetes Service (AKS), the Storage Account File Share, and for the database.

For more information, see Azure private DNS zones in a CDE service.

Fixed issues

- DEX-12763 - Upgraded CDE Service to 1.20.3 and above with SSD Instances unable to run jobs
- DEX-13755 - Azure and AWS service B&R does not restore all the jobs
- DEX-13151 - UI keeps loading when jobs are unpaused if the user has readonly access
- DEX-12634 - VC Security/Access Control allows for duplicate user record
- DEX-14080 - Remove VC cluster maintenance mode configuration on restore
- DEX-13154 - View log for VC will not redirect to Logs tab

- DEX-13742 - UI should send allp details for previous Service version
- DEX-13262 - UI: editing a schedule for the example job fails to send a request
- DEX-13385 - UI does not indicate user does not have access to CLI tool
- DEX-13639 - Without All Purpose details we are showing All Purpose OnDemand and Spot instances on Service Config page
- DEX-13692 - Include sub seconds into execution date for manually triggered Airflow runs
- DEX-13958 - In place upgrade fails from 1.19/1.20 -> 1.21/1.22 if custom runtime images are used with public docker registry

June 24, 2024

This release (1.21.0-h2) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

Azure and AWS service backup and restore

If there were Airflow jobs running on a Virtual Cluster (VC) and you performed a cluster backup and restore, the CDE service and the VC were restored, but the Airflow jobs on the VC were not restored.

This issue has been fixed and now if you perform a backup and restore, the Airflow jobs are restored as well.

In-place upgrade using custom runtime images and public Docker registry

If you performed the in-place upgrade from CDE version 1.19 or version 1.20 to version 1.21 with a custom runtime image that uses a public Docker registry without a credential, the upgrade failed.

This issue has been fixed and now you can perform the in-place upgrade with a custom runtime image that uses a public Docker registry without a credential.

May 31, 2024

This release (1.21.0-h1) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following change.

Listing ACL-based users

Listing users governed by the Access Control List (ACL) for CDE Server versions lower than 1.20.3.

In CDE 1.21, users who had clusters lower than 1.20.3 with ACL-based access control were not able to interact with the ACL. Now, the cluster version is checked and the appropriate ACL-based, or RBAC-based UI is displayed, based on the cluster version.

May 22, 2024

This release (1.21) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following change.

Airflow version upgrade to 2.7

We have upgraded the Airflow version to 2.7. For more information, see:

- Apache Airflow 2.7 Release Notes
- Compatibility for Cloudera Data Engineering and Runtime components

.

March 27, 2024

This release (1.20.3-h2) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

SSD Based Instance used on CDE Service

This release includes a fix for SSD based instance use in the CDE service. Jobs can now be run on an SSD based instances from 1.20.3-h2 and on.

Bug fixes and improvements

This release includes bug fixes and performance improvements for in-place upgrades.

Known issue added

When a customer creates a new CDE service with an SSD Instance enabled on CDE version greater than or equal to 1.19.4, Spark and Airflow jobs do not start at all. For more information on the workaround, see General issues.

March 20, 2024

This release (1.20.3-h1) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following change.

Support for Azure Database for MySQL - Flexible Servers

CDE 1.20.3-H1 moves from Azure Single Server to Azure Flexible Server. This update is due to Azure no longer supporting Azure Single Server as of March 19, 2024.

February 27, 2024

This release (1.20.3) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

Sessions (GA) with enhancements

CDE Sessions is now GA as a default feature. Sessions is an interactive short-lived development environment for running Spark commands to help you iterate upon and build your Spark workloads. The Interaction tab was added so that you can run Java, Impala, and PySpark code in blocks to develop applications. Cloudera currently supports Sessions in the CDE CLI and UI. The Spark UI tab was also added to view active sessions. For more information, see Creating and Managing CDE Sessions and Managing Sessions in CDE using the CLI.

Updated CDE homepage 2.0

CDE now has a revamped landing page with a new design that focuses on a more simplified workflow: Develop, Deploy, and Monitor.

In-place upgrade (GA)

CDE supports upgrades from two CDE versions 1.19.2 and above for AWS and 1.19.4 and above for Azure. Users will need to manually pause, backup, and restore each Virtual Cluster to account for upgrade failures. A way to handle upgrade failures has also been created. In-place upgrade also includes the following:

- Upgrades of CDE core components include: EKS, AKS Services, and Application Services
- Upgrades of dependencies include: Helm, K8s versions, YuniKorn

For more information, see Upgrading CDE and Handling upgrade failures in CDE .

Git repositories (Technical Preview)

You can now use Git repositories to collaborate, manage project artifacts, and promote applications from lower to higher environments. Cloudera currently supports Git providers such as GitHub, GitLab, and Bitbucket. Repository files can be accessed when you create a Spark or Airflow job. You can then deploy the job and use CDE's centralized monitoring and troubleshooting capabilities to tune and adjust your workloads. For more information, see Creating a Git repository in CDE (Technical Preview).

Airflow custom operators and libraries for Python

CDE supports 3rd party python-based plugins and libraries to build custom Airflow pipelines using the CDE UI and API. For more information, see Using custom operators and libraries for Apache Airflow and Using custom operators and libraries for Apache Airflow using API .

New configuration parameters added for Airflow

New parameters were added for Airflow. For more information, see CDE CLI Airflow flag reference and Submitting an Airflow job using the CLI.

Support for Spark Streaming (Technical Preview)

CDE supports Spark Structured Streaming for both Spark 2 and Spark 3. For more information, see Support for Spark Structured Streaming in Cloudera Data Engineering (Technical Preview).

Support for group-based access control for virtual clusters

You can now restrict or grant access to a virtual cluster for specific groups that you specify. For more information, see Applying user and group access for virtual clusters.

Edit all-purpose nodes for AWS and Azure

New sliders to edit all-purpose nodes for AWS and Azure have been added to allow users to control the size of your auto-scaling group. For more information, see Enabling a Cloudera Data Engineering service.

Kubernetes update

CDE now supports K8s 1.27. For more information, see Compatibility for Cloudera Data Engineering and Runtime components.

End of Service Notice

For more information, see Support lifecycle policy.

Support for Airflow 2.6

Support for Airflow 2.6 to version 2.6. For more information, see Compatibility for Cloudera Data Engineering and Runtime components.

Update Automating data pipelines page with Impala VW connections

Impala VWs are supported and the CDWOperator is no longer needed for executing queries. For more information, see Apache Airflow in Cloudera Data Engineering.

Support for Iceberg 1.3

When you upgrade to CDE 1.20.3, ensure that you also upgrade to Iceberg 1.3. For more information, see Compatibility for Cloudera Data Engineering and Runtime components.

Support for setting subnets for the Load Balancer

CDE now supports setting subnets for the Load Balancer during service creation. For more information, see Enabling a Cloudera Data Engineering service.

Enable Observability during service creation

You can select Enable Observability Analytics if you want diagnostic information about jobs and query execution sent to Cloudera Observability. This helps optimize troubleshooting. For more information, see Enabling a Cloudera Data Engineering service

December 11, 2023

This release (1.19.4) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

Kubernetes 1.26 support

- CDE now supports Kubernetes 1.26 for Azure and Amazon Web Services (AWS).
- You can upgrade to the Kubernetes 1.26 cluster through the CDE supported upgrade path.

Amazon Relational Database Service (Amazon RDS) at rest encryption with Customer Managed Keys (CMK) (Technical Preview)

CDE Service deployed on AWS using this CMK enabled environment, will start using CMK based data at rest encryption for RDS.

AWS Kubernetes secret encryption with Customer Managed Keys (CMK) (Technical Preview)

CDE Service deployed using this CMK enabled environment, will start using CMK based encryption for Kubernetes secrets.

Amazon Elastic File System (AWS EFS) data at-rest encryption with Customer Managed Key (CMK) (Technical Preview)

Customer Managed Key is a feature supported by AWS that give customers ownership of their encryption keys.

Amazon Elastic File System (AWS EFS) data in-transit encryption

Support for data in-transit encryption through EFS CSI Driver. The EFS data read/write over the wire are encrypted by TLS.

Amazon Elastic File System (AWS EFS) Anonymous Access restriction

This feature includes security hardening by preventing anonymous user or machines from accessing EFS and its access points.

September 14, 2023

This release (1.19.3) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

Support for Spark 3.3 runtimes

CDE supports Spark 3.3. Note that Spark 3.3.0 is supported on Data Lake 7.2.15. For more information, see Cloudera Data Engineering and Data Lake compatibility.

Support for semi-private network for AWS and fully private network for AKS (Preview)

CDE now supports semi-private network for Amazon Web Services (AWS) and fully private network Azure Kubernetes Service (AKS) with Cluster Connectivity Manager v 2 (CCMv2). For more information, see Enabling a Cloudera Data Engineering service, Enabling a semi-private network on AWS, Enabling a fully private network for a CDE service for Azure (Preview), and Cluster Connectivity Manager.

June 29, 2023

This release (1.19.2) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

Kubernetes update

CDE now supports K8s 1.25 for Azure and Amazon Web Services (AWS).

Support for new AWS region

CDE now supports the EU Milan region for AWS.

Support for user defined routing (UDR)

CDE now supports UDR when you enable a CDE service for Azure. For more information, see Enabling a Cloudera Data Engineering service.

Support for more AMD instances

CDE now includes more AMD instances for the Workload Type drop-down menu when you enable a CDE service.

Workload Secrets

CDE now provides a secure way to create and store workload secrets for Cloudera Data Engineering (CDE) Spark Jobs. This is a more secure alternative to storing credentials in plain text embedded in your application or job configuration. For more information, see Managing workload secrets with CDE Spark Jobs using the API.

May 18, 2023

This release (1.19) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

[Technical Preview] Cloudera Data Engineering Sessions

CDE 1.19 introduces Sessions as an interactive short-lived development environment for running Spark commands to help you iterate upon and build your Spark workloads.

- The Sessions feature is available via CDE user interface and the CDE CLI.
- Python, Scala, and Java are supported session types.

For more information, see Creating Sessions in Cloudera Data Engineering [Technical Preview] and Managing Sessions in Cloudera Data Engineering using the CLI.

[Technical Preview] Lock and unlock

CDE supports locking of a CDE Service which freezes the configuration of a Service and its corresponding Virtual Clusters. When a Service is locked, you are unable to edit, add, or delete a Service and its Virtual Clusters. This will assist during planned upgrades to ensure changes are not made to the Service. For more information, see Locking and unlocking a CDE Service.

[Technical Preview] Airflow file based resource using the CDE CLI

CDE supports Airflow file based resources using the CDE CLI. By creating a pipeline in CDE using the CLI, you can add custom files that are available for tasks. For more information, see Creating an Airflow pipeline with custom files using CDE CLI [technical preview].

Support for new AWS regions

CDE now supports the Hong Kong and Jakarta regions for AWS.

Support for multiple Spark 3 runtimes

CDE supports Spark 3.3. Note that Spark 3.3 must use Data Lake 7.2.16. For more information, see Cloudera Data Engineering and Data Lake compatibility.

Creating and using multiple profiles using CDE CLI

You can now add a collection of CDE CLI configurations grouped together as profiles, to the config.yaml file. You can use these profiles while running commands. You can set the configurations either at a profile level or at a global level. For more information, see Creating and using multiple profiles using CDE CLI.

Using spark-submit drop-in migration tool for migrating Spark workloads to CDE

CDE provides a command line tool cde-env to help migrate your CDP Spark workloads running on CDP Private Cloud Base (“spark-on-YARN”) to CDE without having to completely rewrite your existing spark-submit command-lines. For more information, see Using spark-submit drop-in migration tool for migrating Spark workloads to CDE.

New Virtual Cluster types

CDE now provides a choice between two tiers during Virtual Cluster creation. Administrators will see the following two options:

- Core (Tier 1) - Batch-based transformation and engineering options.
- All-Purpose (Tier 2) - Develop using interactive sessions and deploy both batch and streaming workloads.

For more information, see Creating virtual clusters.

Default setting for external.table.purge property for migrating Hive tables

When migrating Iceberg tables in Spark 3, the external.table.purge property is now set to FALSE by default. For more information, see Importing and migrating Iceberg table in Spark 3.

New exit codes for the CDE CLI

CDE now provides exit codes for the CDE CLI. The exit codes help users better identify the error. For more information, see Cloudera Data Engineering CLI exit codes.

Support for Data Lake

CDE 1.19 supports Data Lake 7.2.14 through 7.2.16. For more information, see Cloudera Data Engineering and Data Lake compatibility.

March 30, 2023

This release (1.18.3) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces a new improvement that is described in this topic.

Kubernetes update

CDE now supports K8s 1.24 for Azure and Amazon Web Services (AWS).

January 19, 2023

This release (1.18.1) of the Cloudera Data Engineering Service on CDP Public Cloud introduces the following changes.

Upgrade to Airflow 2.3.4

CDE 1.18.1 now runs with Airflow 2.3.4. This upgrade includes several fixes to improve performance and stability.

Support for additional Grafana charts

Additional Grafana charts that specify metrics for Livy memory and API server have been added.

Support for Data Lake

CDE 1.18.1 now supports Data Lake 7.2.16. For more information, see Cloudera Data Engineering and Data Lake compatibility.

November 23, 2022

This release (1.18) of the Cloudera Data Engineering Service on CDP Public Cloud introduces the following changes.

Updated CDE user interface

The user interface for CDE 1.17 and above has been updated with easy access to commonly used pages, a new Home page, and a Virtual Cluster drop-down menu that allows you to view relevant content related to each Virtual Cluster that you select. Only users who have a CDE Service on 1.18 and create new Virtual Clusters on 1.18 will see the changes. Users on older versions will continue have access to the old UI. The following user interface changes were made:

- Left-hand menu displays the following:
  - Home- New landing page that displays Virtual Clusters and convenient quick-access links.
  - Jobs - Displays jobs for the Virtual Cluster that you select from the drop-down menu in the upper left-hand corner.
  - Job Runs - Displays the run history of all jobs within a selected Virtual Cluster.
  - Resources - Displays resources created within a selected Virtual Cluster.
  - Administration - Displays services and Virtual Clusters that can be customized (previously known as the Overview page.

  Note: If you're using a browser in incognito mode, you'll need to allow all cookies in your browser settings so that you can view the following CDE pages: Pipelines, Spark, and Airflow.

Airflow performance

Airflow scaling improvements include support for 1500 DAGs on AWS and about 300 to 500 DAGs when deploying on Azure. For more information, see Apache Airflow scaling and tuning considerations.

Support for the eu-1 (Germany) and ap-1 (Australia) regional Control Plane

The eu-1 (Germany) and ap-1 (Australia) regional Control Plane now supports CDE. For the list of all supported services for all supported Control Plane regions, see CDP Control Plane regions.

Java Virtual Machine Debugger (Tech preview)

Attaching a remote debugger (Java virtual machine (JVM) debugger) to a CDE Spark job is now supported as a technical preview feature. For more information, see Using Java virtual machine (JVM) debugger with Apache Spark jobs in Cloudera Data Engineering (Preview) .

Hive Warehouse Connector tables

Hive Warehouse Connector (HWC) tables are now supported in Spark 3 of CDE.

Backup & Restore in object storage

Remote backup storage (object store) is now supported. Previously, only backup to and restore from local storage was supported. This is supported through the CLI and API only. For more information, see Backing up Cloudera Data Engineering jobs and Restoring Cloudera Data Engineering jobs from backup.

Limitations for raw Scala code in CDE

Limitations have been added to the raw Scala code. For limitation details, see Running raw Scala code in Cloudera Data Engineering.

Support for Iceberg V2

Iceberg table format version 2 (v2) is generally available (GA) in CDE. The latest specifications include the following key updates:

- UPDATE and DELETE operations follow the Iceberg format v2 row-level position delete specification and enforces snapshot isolation.
- DELETES, UPDATES, and MERGE operations use the merge-on-read function by default. Merge-on-read is more efficiant than the copy-on-write function because it does not rewrite file data.

For more information, see Prerequisites

October 12, 2022

This release (1.17-h1) of the Cloudera Data Engineering Service on CDP Public Cloud introduces the following changes.

Support for Iceberg 0.14

When you upgrade to CDE 1.17-h1, ensure that you also upgrade to Iceberg 0.14. For more information, see Using Apache Iceberg in Cloudera Data Engineering. The following features are included with Iceberg 0.14:

- MERGE operations allow for bulk updates and DELETES.
- CDE Azure deployments are now able to leverage Iceberg for Lakehouse architecture.
- [Technical Preview] Iceberg table format version 2 (v2) is the latest specification available and includes the following key updates:
  - UPDATE and DELETE operations follow the Iceberg format v2 row-level position delete specification and enforce snapshot isolation.
  - DELETES, UPDATES, and MERGE operations use the merge-on-read function by default. Merge-on-read is more efficient than the copy-on-write function because it does not rewrite file data.

Set default values for the variables in CDE job specification

Using [--default-variable] flags you can now replace strings in job values. For more information, see Creating and updating Apache Spark jobs using the CLI.

September 26, 2022

This release (1.17) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

Service reference architecture

Reference architecture has been published to outline best practices on scaling CDE service, including when running Airflow bases pipelines. For more informarion, see Recommendations for scaling CDE deployments and Apache Airflow scaling and tuning considerations.

Kubernetes dashboard

CDE provides the option to view the Kubernetes dashboard to provide an easy user experience for monitoring your diagnostics. The dashboard is to be used when troubleshooting in coordination with Cloudera Support. For more information, see Accessing the Kubernetes dashboard.

Azure private storage

As of CDE 1.16, Azure private storage is supported. Details around deploying and configuring CDE with Azure private storage are now available. For more information, see Supporting Azure private storage.

SSL Support for Azure DB

For increased security, CDE on Azure will now deploy SSL enabled with TLS 1.2.

Technical Service Bulletin (2022-587)

Technical Service Bulletin (2022-587) has been resolved.

July 20, 2022

This release (1.16) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

Airflow pipeline UI editor (GA)

- Airflow Pipeline UI editor is now GA as a default feature in new Virtual Clusters with support for all major browsers (Firefox, Chrome, and Safari).

Upgrade to Airflow 2.2.5

CDE 1.16 now runs with Airflow 2.2.5.

- Several fixes to improve performance and stability have been bundled with the upgrade.
- New Virtual Clusters will automatically use the new Airflow version.
- This version deprecated the timezone package usage. The DAGs need to be updated to use the pendulum package instead. If your airflow DAGs need to be timezone aware then they should rely on the pendulum timezone library for start and end dates as described here. Otherwise, the backup and restore process will not be able to restore these DAGs. For more information, see CDE known issues.

Spark 3 support for raw scala code

Spark 3 support for raw scala code.

Previously this feature was limited to Spark 2, it is now extended to Spark 3 based Virtual Clusters. This allows you to directly run raw scala via API & CLI in batch-mode without having to compile, similar to what spark-shell supports.

Support for Azure private storage

CDE now supports Azure private storage. Both private ABFS and ADLS gen2 containers are now supported.

Editing VC configurations post creation

You can now modify the virtual settings such as cluster quotas (CPU/memory) dynamically.

Loading example jobs and sample data using new VCs

CDE provides an option to add in-product examples of data & jobs in new virtual clusters to facilitate smoother onboarding and learning for new customers.

Kubernetes update

CDE now supports K8s 1.22.

- The CSP EOS for K8s 1.21 is as follows:

  For Azure: July 2022

  For AWS: February 2023
- Check for removals as per this upgrade:

  Kubernetes API and Feature Removals In 1.22 and Removed APIs by release

Support for creation of a Default Virtual Cluster

CDE now provides support for default virtual clusters. This will help you get a jump start to create your jobs easily, without having to wait to create a CDE virtual cluster, making the onboarding smoother. You have the option to turn this selection off if you do not wish to use a default virtual cluster.

For more information, see Enabling Cloudera Data Engineering service.

[Technical Preview] In-place upgrades

CDE supports upgrades from two CDE versions prior for both AWS and Azure. For example, if the current CDE version is 1.18, then upgrades are supported from CDE 1.16. The upgrades can be triggered by an Admin from CDE UI.

- Users will need to manually pause/backup/restore each Virtual Cluster to account for upgrade failures.
- Upgrades of CDE core components include: EKS, AKS Services, and Application Services
- Upgrades of dependencies include: Helm, K8s versions, YuniKorn

June 30, 2022

This release (1.15-h1) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud has certified support for Apache Iceberg v0.13.

GA of Apache Iceberg

- You can use Cloudera Data Engineering virtual clusters running Spark 3 to interact with the latest version (0.13) of Apache Iceberg tables.
- CDE supports row level updates via copy-on-write MERGE / UPDATES/ DELETES operations. Copy-on-write is helpful in bulk updates in read heavy use-cases.
- Compaction is also supported using Spark Iceberg APIs.
- As support for Atlas lineage is still in progress, users should set the following Spark property in their jobs: spark.lineage.enabled=false.
- For more information, see Using Apache Iceberg in Cloudera Data Engineering.

April 27, 2022

This release (1.15) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

[Technical Preview] Support for Iceberg 0.13

- You can use Cloudera Data Engineering virtual clusters running Spark 3 to interact with the latest version (0.13) of Apache Iceberg tables.
- CDE now supports row level updates via copy-on-write MERGE / UPDATES/ DELETES operations. Copy-on-write is helpful in bulk updates in read heavy use-cases.
- For more information, see Using Apache Iceberg in Cloudera Data Engineering .

[Technical Preview] In-place upgrades

- CDE supports upgrades from two CDE versions prior for both AWS and Azure. For example, if the current CDE version is 1.18, then upgrades are supported from CDE 1.16. The upgrades can be triggered by an Admin from CDE UI.
- Users will need to manually pause/backup/restore each Virtual Cluster to account for upgrade failures.
- Upgrades of CDE core components include: EKS, AKS Services, and Application Services
- Upgrades of dependencies include: Helm, K8s versions, YuniKorn
- For more information, see CDE In-place Upgrades (Preview)

Job email alerts

SLA miss and job failure conditions can be configured for email notifications.

Job runtime notices

Active jobs will now provide notification to the user when certain conditions are met and jobs are not behaving as expected making it easier to understand why jobs might be stuck or not making progress.

For more information, see Running jobs in Cloudera Data Engineering

Spark 3.2

CDE now supports Apache Spark 3.2.

Data Lake upgrades

CDE has now been certified when Data Lake is resized..

February 09, 2022

This release (1.14) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

Improved handling of job resources to reduce EFS utilization

- Recursive copying of frequently used and large file resources can result in very high I/O throughput and can exhaust cloud storage burst credits, leading to poor performance. To avoid excessive file copying, CDE now uses hard linking in AWS by default.

[Technical Preview] Apache Iceberg support

- Apache Iceberg tables are now supported with Spark 3 virtual clusters on AWS. Use tables at petabyte scale without impacting query planning, while benefiting from efficient metadata management, snapshotting, and time-travel.
- Run multi-analytic workloads by accessing those same tables in Cloudera Data Warehouse (CDW) with Hive and Impala for BI and SQL analytics (Expected in an upcoming CDW release).

[Technical Preview] Remote Shuffle Service

- You can now store Spark shuffle data on remote servers. This improves resilience in case of executor loss.
- This feature is available as a Technical Preview. Contact your Cloudera account representative to enable access to this feature.

Unified diagnostic bundle

- A single click now generates one unified bundle containing both service logs and summary status.
- The bundles are stored securely in the object storage of the environment.
- A historical list of previously generated bundles are available for access.

Guardrails to prevent submitting jobs that do not fit resource capacity

- CDE now automatically prevents execution of jobs that do not fit on the available resources.
- CDE takes into account Kubernetes and system reserved resources, daemonset utilized resources, and Spark overhead factors.
- The API returns an error with run failed to start: requested *[***TYPE AND AMOUNT OF RESOURCE***]* is more than *[***THE MAXIMUM AMOUNT OF AVAILABLE RESOURCES OF THAT TYPE***]* allocatable per cluster node
- You can either reduce the Spark executor and driver CPU and/or memory requirements, or deploy on a larger cluster.

Notification email configuration can now be verified

When configuring the optional email alerts feature [Technical Preview] during virtual cluster creation, you can now verify the SMTP settings before creating the virtual cluster.

Streamlined resource creation and re-use during job creation

You can now create a resource on the fly when creating a job. Alternatively, you can select from a list of existing resources, if any, to upload your application or DAG file. This promotes re-usability of project artifacts across jobs.

Kubernetes update

CDE now supports K8s 1.21.

December 21, 2021

This release (1.13.0-h1-b1) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud addresses the Log4j2 security vulnerability CVE-2021-44228.

Fixed Log4j2 security vulnerability CVE-2021-44228

- Removed JndiLookup.class from affected JAR files.

For instructions on upgrading your existing CDE services and virtual clusters, see Cloudera Data Engineering fix for CVE-2021-44228.

November 9, 2021

This release (1.13) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the following changes.

Non-transparent proxy support

- CDE supports deploying into CDP environments using a non-transparent proxy.
- The proxy is registered and enabled during CDE environment creation.
- The proxy configuration is automatically added to the deployed CDE service and virtual clusters (VCs).

UI support for Python virtual environments

- You can now create custom Python resources on the CDE UI, including virtual environments (venvs)
- These custom venvs are selectable in the job creation wizard when creating PySpark jobs.

Support for Airflow core operators

- With Airflow 2, Cloudera now supports all core operators.

Support for Ranger Authorization Service

- CDE now supports Ranger Authorization Service (RAZ) in AWS and Azure environments.
- For more information, see RAZ support requirements

Older releases

Overview of new features, enhancements, and changed behavior introduced in earlier releases of Cloudera Data Engineering.

October 18, 2021

This release (1.12) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Apache Airflow 2

With this release, Apache Airflow 2.1 is the new default managed scheduler in CDE. It comes with governance, security and compute autoscaling enabled out-of-the-box, along with integration with CDE's job management APIs giving users the flexibility to deploy custom DAGs that tap into Cloudera Data Platform (CDP) data services like Spark in CDE and Hive in CDW.

For more information on what's new in Airflow 2, see the upstream documentation.

[Technical Preview] Airflow pipeline authoring UI

With the CDE Pipeline Authoring UI, any CDE user irrespective of their level of Airflow expertise can create multi-step pipelines with a combination of out-of-the-box operators (CDEOperator, CDWOperator, BashOperator, PythonOperator). Nevertheless, you can still deploy your own customer Airflow DAGs (Directed Acyclic Graphs) as before, or use the Pipeline Authoring UI to bootstrap your projects for further customization.

This feature is in Technical Preview and available on new CDE services only. When creating a Virtual Cluster, a new option allows you to enable the Airflow Authoring UI.

For best user experience, Cloudera suggests using Google Chrome for this feature

[Technical Preview] Email alerts

You can now configure email alerts during Virtual Cluster setup and schedule them in custom Airflow DAGs.

Kubernetes update

CDE now supports K8s 1.20.

August 30, 2021

This release (1.11) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

GA support for virtual clusters powered by Apache Spark 3

- Support for virtual clusters powered by Apache Spark 3 is no longer a Technical Preview feature, and is now generally available (GA).
- The following functionalities are not currently supported:
  - Deep analysis (visual profiler)
  - HWC - that is, Hive managed ACID tables (Direct Reader & JDBC mode)
  - Phoenix Connector
  - SparkR
  - Kudu

[Technical Preview] Fully private AKS cluster set up

- Fully private AKS clusters are now supported, for customers who want to restrict resources from being exposed via public IP addresses. This allows securing the Kubernetes cluster even more, an AKS API server can be created with a private IP address which is only accessible to the resources which are running inside of the Azure virtual network (VNet).
- A private AKS is deployed within customers' network and leverages CCMv2/Proxy for accessing the K8s APIs.
- Cloudera recommends using one single resource group per environment. You can accomplish this by selecting a (pre-created) resource group during CDP environment creation.

Gang scheduling enabled by default

- YuniKorn Gang scheduling policy is now enabled by default within CDE.
- For more information on Gang scheduling, see the Spark on Kubernetes – Gang Scheduling with YuniKorn Cloudera Blog post.

[Technical Preview] User-specified IAM roles

- CDE job pods can now run with a user-specified IAM role with the role credentials automatically supplied as instance credentials. This allows transparent usage of cloud SDKs or any code making use of the instance credentials provider. User roles are secured and allocated through the CDP environment IDBroker mappings.
- This feature is available as a Technical Preview. Contact your Cloudera account representative to enable access to this feature.

Spark Analysis disabled by default

- Metric collection from Spark jobs is now disabled by default to provide the most optimal performance.

- During development and testing, you can turn on additional Spark profiling:
  - On the CDE UI:

    After creating the job, go to its Configuration tab and toggle the Spark Analysis option.

    For more information, see Managing jobs in Cloudera Data Engineering.
  - From CLI/API:

    Set the following configuration parameter during job creation: dex.safariEnabled=true

    For more information, see Managing Cloudera Data Engineering jobs using the CLI and Creating a Cloudera Data Engineering job using the API respectively.

August 2, 2021

This release (1.9) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Kubernetes version updated to 1.19

The Kubernetes version has been updated to 1.19.

Dynamic allocation enabled by default

Dynamic allocation is now enabled by default. You can configure the initial number of executors as well as a range of executors per job.

Dynamic allocation scales job executors up and down as needed for running jobs. This can provide large performance benefits by allocating as many resources as needed by the running job, and by returning resources when they are not needed so that concurrent jobs can potentially run faster.

Resources are limited by the job configuration (executor range) as well as the virtual cluster auto-scaling parameters. By default, the executor range is set to match the range of CPU cores configured for the virtual cluster. This improves resource utilization and efficiency by allowing jobs to scale up to the maximum virtual cluster resources available, without manually tuning and optimizing the number of executors per job.

This is a change from the default behavior (static allocation) in older releases. If you restore job configuration from an older release, the restored jobs will use dynamic allocation.

Support for Amazon AWS S3 URLs in jobs

A previous issue with S3 URLs in job configurations has been fixed. You can now specify S3 URLs for your application code and Jar files. For jobs using this functionality, you must also add the following Apache Spark configuration option:

```
spark.hadoop.fs.s3a.delegation.token.binding=org.apache.knox.gateway.cloud.i
dbroker.s3a.IDBDelegationTokenBinding
```

On-demand Python virtual environments

You can now submit a job with a Python requirements.txt file, as follows:

```
cde spark submit my_job.py --python-requirements /PATH/TO/requirements.txt
```

This builds the Python virtual environment resource for the user, attaches it to the job, and sets it to be cleaned up when the job run terminates.

June 23, 2021

This release (1.8) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Python virtual environment improvements

Python virtual environments are now built in dedicated pods, and support C-based Python libraries.

Open source CDE/CDW operators for Apache Airflow

You can now use CDE and CDW operators with your existing Apache Airflow deployment. For more information and instructions, see Using CDE with an external Apache Airflow deployment.

CDE jobs in different virtual clusters within the same DAG file

Airflow DAG files can now trigger CDE jobs in different virtual clusters. For more information, see Apache Airflow in Cloudera Data Engineering.

CIDR notation support for IP whitelist

You can now add IP ranges to the whitelist using CIDR notation.

Subnet selection option

You can now select a subnet to use for CDE when enabling a CDE service.

May 20, 2021

This release (1.7) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

GA support for Microsoft Azure

- Support for Microsoft Azure is no longer a *Technical Preview* feature, and is now generally available (GA).

GA support for gang scheduling

- Gang scheduling is no longer a *Technical Preview* feature, and is now generally available (GA).
- For a detailed explanation on gang scheduling, see the blog post.

Diagnostic bundles

- The Diagnostic page has been enhanced to support granular log selection as well as a snapshot of the service and cluster status.

[Technical Preview] Support for virtual clusters powered by Apache Spark 3

- You can now create virtual clusters powered by Apache Spark version 3. You cannot use Spark 2 and Spark 3 within the same virtual cluster, but you can have multiple Spark 2 and Spark 3 virtual clusters within the same CDE service.
- This features is available as a *Technical Preview*. Contact your Cloudera account representative to enable access to this feature.

[Technical Preview] Bin packing resource scheduling

- Bin packing is a new Apache Yunikorn resource management policy. Bin packing makes more efficient use of available nodes when assigning executors to hosts.
- This features is available as a *Technical Preview*. Contact your Cloudera account representative to enable access to this feature.

Automatic TLS certificate renewal

- Cluster TLS certificates are now renewed automatically.

April 7, 2021

This release (1.6) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

API keys

- CDE users can now use API keys, managed using the CDP user management service (UMS), to interact with the CDE Jobs API using the command line.

Raw Scala code

- Users can now submit jobs with raw Scala code, without compiling. These jobs run spark-shell to process the application file.

Diagnostic bundles

- Admins can now access summary diagnostic logs directly on their local machine.
- A new Diagnostic page has been added to the CDE Service details to generate and download the bundle.

Force TLS certificate renewal

- CDE services older than 90 days will have expired TLS certificates. A new action has been added to the CDE service hamburger menu to renew the certificates and avoid access issues for DE users.

[Technical Preview] GANG scheduling

- GANG is a new resource scheduling policy that overcomes scale-up challenges in situations where high rates of job submission lead to queuing. The new scheduling policy moves jobs off the queue in batches. This clears up the queue, forces scale-up of nodes to process the burst of incoming jobs, and reduces wait and startup time of jobs.
- By default, GANG scheduling is disabled. It can be turned on for specific jobs by adding a new job-level configuration option.

March 9, 2021

This release (1.5) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces technical preview support for deploying CDE on Microsoft Azure.

[Technical Preview] Support for Microsoft Azure

- Cloudera Data Engineering can now be deployed on Microsoft Azure. This functionality is provided as a technical preview, and is not supported for production environments.

February 4, 2021

This release (1.4) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Support for Airflow DAGs

- DE and ML practitioners can now define their own pipelines packaged as an Apache Airflow Python DAG. Currently supported CDP operators include running Spark jobs on CDE and Hive jobs on CDW.
- An embedded Airflow UI within the job & job run details pages gives users a “deep link” to the specific Airflow DAG making it easier to access within the context of the job runs.
- The Schedule page has been removed from the left panel of the virtual cluster jobs UI, and the full Airflow UI is now exposed through the Virtual Cluster details page.

Improved service observability for service troubleshooting

Diagnostic bundles can now be collected through a new API end-point, which includes:

- Cloud resource status: (EKS, RDS, EFS, ELB)
- Helm status(helm version, helm ls -A)
- Kubernetes status (deployments, pods, services, ingresses, config maps)

Virtual Cluster user-based ACL

- By default a Virtual Cluster is accessible to all DEUsers and DEAdmins, which includes the Jobs API, Airflow UI, along with any connections and credentials defined within Airflow.
- Enabling access control will now limit access to the API and UIs of the Virtual Cluster to a subset of users - normal or machine users. Groups are not yet supported.

Kubernetes support

- CDE now supports EKS 1.18

December 21, 2020

This release (1.3) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Better scaling

- Larger compute instance types to accommodate heavier ETL workloads and tweaks to infra services for higher job submission API throughput.
- Better Spark defaults tuned for Kubernetes, improving scale up and stability.

Run profiling analysis on demand for additional tuning metrics

Users can trigger additional profiling analysis for any job, providing memory and CPU utilization, and stage-level CPU flamegraphs.

Workload Manager integration

CDE services can now share Spark application metadata and metrics with Workload Manager for better visibility into aggregate workloads across the entire CDP environment, manage SLAs, and identify additional tuning opportunities.

Easy log configuration & access

- Full logs can now be downloaded as a new download option along with a quick bookmark to the S3 location of the Spark application logs.
- The UI now supports setting the log level from OFF to DEBUG and TRACE.

Lightweight backup and restore

You can use the CDE CLI & API to backup and restore jobs from one virtual cluster into another virtual cluster within the same CDE service or completely new one. Helps support upgrades as it requires enabling a new service.

Create/clone job from existing run for easier debugging

CLI now supports cloning job runs, carrying over configs and parameters, making it easier to troubleshoot runs with failures.

[Tech Preview] Self-authored Airflow DAGs

DE and ML practitioners can now define their own pipelines packaged as an Apache Airflow Python DAG. Currently supported CDP operators include running Spark jobs on CDE and Hive jobs on CDW.

November 9, 2020

This release (1.2) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

CDP CLI Integration

Administrators can now automate the enabling of CDE services and creation of Virtual Clusters through CDP CLI. Jobs will continue to be managed through the CDE CLI shipped with the service.

Multiple CDE Services

It's now easier to enable CDE service multiple times within the same environment (datalake/SDX). This allows admins to set up multiple CDE services with differing instance profiles and allows for easier consumption tracking through AWS tags at the service level.

Python virtual environments

Users can now specify a list of python libraries as dependencies for Pyspark jobs. This can be specified through a requirements.txt file that is uploaded and managed through CLI/API.

CDP Trial Tours

The first trial tour for Data Engineering admins is now available.

September 21, 2020

This release (1.1) of the Cloudera Data Engineering (CDE) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Spot instance support

Users now have an additional knob to control cost, by being able to choose between running on Spot or On-Demand instances, providing up to 90% discount in AWS resources.

Local (native) SSD

For memory and shuffle heavy workloads, CDE now allows using instances with local (native) SSD for intermediate results boosting performance.

Resource tags

Allows administrators to define tags during CDE service creation to track and audit cloud provider resources.

IP Whitelisting

Administrators can lock down access to the EKS control plane components via a CIDR range.

Stability & Security

New CDE service deployment will now use Kubernetes 1.15 and Helm 3; this improves the stability and security of the service moving forward

July 30, 2020

This release (1.0) marks the General Availability (GA) release of the Cloudera Data Engineering (CDE) service on CDP Public Cloud.

Cloudera Data Engineering fix for CVE-2021-44228

On December 21, 2021, Cloudera released Cloudera Data Engineering for Public Cloud version 1.13.0-h1-b1. It addresses the Log4j2 security vulnerabilities listed below. Cloudera urges all customers to upgrade their Data Engineering services to the latest version.

- CVE-2021-44228 which affects Apache Log4j2 versions 2.0 through 2.14.1.
- CVE-2021-45046 which affects Apache Log4j2 version 2.15.0

Upgrade to the latest Data Engineering version

To upgrade your Cloudera Data Engineering service to the latest version, which addresses the log4j2 security vulnerability, follow these steps. These steps provide a comprehensive upgrade and are the recommended approach. To ensure compatibility with the CDP environment, you must also upgrade the environment DataLake to Runtime 7.2.11 or higher.

1. For each existing virtual cluster, create a backup of the jobs and resources.
2. Enable a new Cloudera Data Engineering service for each existing CDE service you have. Make sure to use the same settings as the existing service for each new corresponding service.
3. Within each new Data Engineering service, create a new virtual cluster for each existing virtual cluster in the pre-existing service. Make sure to use the same settings as the existing virtual cluster for each new corresponding virtual cluster.
4. After making sure that you have added CDE services and virtual clusters to match your existing deployment, restore the backup file for each pre-existing virtual cluster to the corresponding new virtual cluster.

Result

Your Cloudera Data Engineering service and all associated virtual clusters are upgraded to the latest version.

Known issues and limitations in Cloudera Data Engineering Public Cloud

This section lists known issues and limitations that you might run into while using the Cloudera Data Engineering (CDE) service in CDP Public Cloud.

General known issues with Cloudera Data Engineering

Learn about the general known issues with the Cloudera Data Engineering service on public clouds, the impact or changes to the functionality, and the workaround.

DEX-12616: Node Count shows zero in /metric request

Cloudera Data Engineering 1.20.3 introduced compatibility with Kubernetes version 1.27. With this update, the kube_state_metrics no longer provides label and annotation metrics by default.

Earlier, Cloudera Data Engineering used label information to calculate the Node Count for both Core and All-Purpose nodes, which was automatically exposed. However, due to the changes in kube_state_metrics, this functionality is no longer available by default. As a result, the Node count shows zero in /metrics, charts, and the user interface.

1. Make sure that you set a kube-config for the Cloudera Data Engineering service.

2. Run the following command, which exposes the label manually in the prometheus-kube-state-metrics container:

```
`kubectl patch deployment monitoring-prometheus-kube-state-m
etrics --namespace monitoring --type='json' -p=\
'[{"op": "add",
"path":"/spec/template/spec/containers/0/args", "value":["--
metric-labels-allowlist=nodes=[role]"]}]'`
```

For more information about how to see the node count, see Checking the node count on your Cloud Service Provider's website.

DEX-11340: Kill all the alive sessions in prepare-for-upgrade phase of stop-gap solution for upgrade

If Spark sessions are running during the Cloudera Data Engineering upgrade, they are not automatically killed, leaving them in an unknown state during and after the upgrade.

Ensure that you do not have any Spark sessions running during the Cloudera Data Engineering upgrade. If they are running during the Cloudera Data Engineering upgrade, kill them before proceeding.

DEX-14094: New pipeline editor can create a cde_job step of its own pipeline job which causes recursion and looping

If you add a Cloudera Data Engineering Job step in the pipeline editor and select the same job as the pipeline job from the Select Job drop-down list while configuring the pipeline job using the editor, then running the pipeline job results in a recursive loop.

For example, You have created a pipeline job named test-dag and selected the same job test-dag from the Select Job drop-down list while adding the Cloudera Data Engineering Job step, then running the pipeline job results in a recursive loop.

DEX-14084: No error response for Airflow Python virtual environment at Virtual Cluster level for view only access user

If a user with a view only role on a Virtual Cluster (VC) tries to create an Airflow Python virtual environment on a VC, the access is blocked with a 403 error. However, the no-access 403 error is not displayed on the UI.

DEX-13465: Cloudera Data Engineering Airflow DAG Code Fetch is not working

The embedded Airflow UI within the Cloudera Data Engineering Job pages does not correctly show the "Code" view.

Open the Jobs page through Virtual Cluster link and use the old Cloudera Data Engineering UI or manually open Airflow UI through Virtual Cluster details page and navigate to the appropriate DAG details.

DEX-11639: "CPU" and "Memory" Should Match Tier 1 and Tier 2 Virtual Clusters AutoScale

CPU and Memory options in the service or cluster edit page display the values for Core (tier 1) and All-Purpose (tier 2) together. However, they must be separate values for Core and All-Purpose.

DEX-12482: [Intermittent] Diagnostic Bundle generation taking several hours to generate

Diagnostics bundles can intermittently take very long to get generated due to low EBS throughput and IOPS of the base node.

Increase the EBS throughput and IOPS values in the CloudFormation template, then trigger new diagnostic bundles.

Note: Any existing job runs will fail when the CloudFormation template is changed as a new node comes up and the existing one is terminated.

DEX-14253: Cloudera Data Engineering Spark Jobs are getting stuck due to the unavailability of the spot instances

The unavailability of AWS spot instances may cause Cloudera Data Engineering Spark jobs to get stuck.

Re-create the Virtual Cluster with on-demand instances.

Note: Cloudera does not recommend using spot instances for creating VCs.

DEX-14192: Some Spark 3.5.1 jobs have slightly higher memory requirements

Some jobs running on Spark 3.5.1 have slightly higher memory requirements, resulting in the driver pods getting killed with a k8s OOMKilled.

Increase the driver pod memory from the default 1GB to 1.2GB in the job's configuration.

DEX-14173: VC Creation is failing with "Helm error: 'timed out waiting for the condition', no events found for chart"

In case of busy k8s clusters, installing VC/Cloudera Data Engineering may fail with an error message showing Helm error: 'timed out waiting for the condition', no events found for chart.

Try installing again. The failure is due to image pulls timing out. The installation will go through as more resources become available.

DEX-14067: Email alerting is not sending email to the recipient even after the job fails

When a Virtual Cluster is created through the UI, SMTP configuration and authentication is empty. Enabling Email Alerting after VC creation allows users to add the SMTP parameters, but the updated password is not fetched when sending email alerts.

Restart the runtime-api pod after adding the SMTP configuration. On restart the secrets are re-fetched and all services are reinitialized with the updated secrets.

1. List all pods in the namespace (for example, dex-app-dtznsgc5):

```
kubectl get pods -n [***NAMESPACE***]
```

2. Copy the runtime-api pod name. (For example, dex-app-dtznsgc5-api-b5f8d7cb9-gdbxb.)
3. Delete the runtime-api pod:

```
kubectl delete pod [***POD NAME***] -n [***NAMESPACE***]
```

4. A new instance of the runtime-api pod will get initialized.
5. Test email alerting. (For example, via a job run failure.)

DEX-14027: Spark 3.5.1 jobs are failing with error 'org.apache.hadoop.fs.s3a.impl.InstantiationIOException'

Spark 3.5.1 RAZ-enabled Cloudera Data Engineering clusters fail to initialize RAZ S3 plugin library due to recent backward incompatible changes in the library, and jobs fail with error org.apache.hadoop.fs.s3a.impl.InstantiationIOException.

Depending on the configuration level, multiple following workarounds are available through the API or CLI, since setting empty parameters in the UI is not possible.

- (preferred) as a VC-level config setting (not applicable to Cloudera Data Engineering sessions), with the API, use:

```
curl '[***CLUSTER APP INSTANCE API ENDPOINT***]'; \
  -X 'PATCH' \
  -H 'Connection: keep-alive' \
  -H 'Content-Type: application/json' \
  -H  'Cookie: [***COOKIE CONTAINING CDP-TOKEN***]' \
  --data-raw '{"config":{"sparkConfigs":{"spark.hadoop.fs.s3
a.http.signer.class":"org.apache.ranger.raz.hook.s3.RazS3Sig
nerPlugin","spark.hadoop.fs.s3a.http.signer.enabled":"true",
```

```
"spark.hadoop.fs.s3a.custom.signers":"","spark.hadoop.fs.s3a
.s3.signing-algorithm":""}}}'
```

- as a per session configuration (VC-level configuration will not be applied), in the CLI, run:

```
./cde --vcluster-endpoint [***CLUSTER APP INSTANCE API
 ENDPOINT***] session create
--name [***CDE SESSION NAME***] \
--conf "spark.hadoop.fs.s3a.custom.signers=" \
--conf "spark.hadoop.fs.s3a.http.signer.class=org.apache.range
r.raz.hook.s3.RazS3SignerPlugin" \
--conf "spark.hadoop.fs.s3a.http.signer.enabled=true" \
--conf "spark.hadoop.fs.s3a.s3.signing-algorithm="
```

- in the CLI, at job creation, run:

```
./cde --vcluster-endpoint [***VIRTUAL CLUSTER JOB API URL***]
 job run --name [***CDE JOB NAME***]> \
--conf "spark.hadoop.fs.s3a.custom.signers=" \
--conf "spark.hadoop.fs.s3a.http.signer.class=org.apache.rang
er.raz.hook.s3.RazS3SignerPlugin" \
--conf "spark.hadoop.fs.s3a.http.signer.enabled=true" \
--conf "spark.hadoop.fs.s3a.s3.signing-algorithm="
```

Note: spark.hadoop.fs.s3a.custom.signers and spark.hadoop.fs.s3a.s3.signing-algor ithm needs to be set to empty to unset them.

DEX-13975: 'spark.catalog.listTables()' command in job is failing in Python Spark for Spark 3.5.1

Using catalog.listTables() with Iceberg tables results in an exception org.apache.spark.sql.catalys t.parser.ParseException: [PARSE_SYNTAX_ERROR] Syntax error at or near end of input.

Set spark.sql.legacy.useV1Command=true for running catalog.listTables().

DEX-13957: Cloudera Data Engineering metrics and graphs show no data

Cloudera Data Engineering versions 1.20.3 and 1.21 use Kubernetes version 1.27. In Kubernetes version 1.27, by default, the kube_state_metrics does not provide label and annotation metrics. For this reason, the node count shows zero for core and all-purpose nodes in the Cloudera Data Engineering UI and in charts.

As a prerequisite, set a kube-config for the Cloudera Data Engineering service and run:

```
kubectl patch deployment monitoring-prometheus-kube-state-metrics
 --namespace monitoring --type='json' -p=\
'[{"op": "add",
"path":"/spec/template/spec/containers/0/args", "value":["--metr
ic-labels-allowlist=nodes=[role]"]}]'
```

DEX-12630: Cloudera Data Engineering Service failed to run jobs on SSD Based clusters

When a customer creates a new Cloudera Data Engineering service with an SSD Instance enabled on Cloudera Data Engineering version greater than or equal to 1.19.4, Spark and Airflow jobs do not start at all. The same problem happens if an existing Cloudera Data Engineering service is upgraded to 1.19.3 or greater and has SSD Instance enabled.

Create a new Cloudera Data Engineering service without SSD Instance enabled until 1.20.3-h1. From 1.20.3-h2, you can create a new Cloudera Data Engineering service with SSD Instance enabled. However, you cannot upgrade existing SSD based service, therefore you must create a new one.

DEX 12451: Service creation fails when "Enable Public Loadbalancer" is selected in an environment with only private subnets

When creating a service with the Enable Public Load Balancer option selected, the service creation fails with the following error:

```
“CDE Service: 1.20.3-b15 ENV: dsp-storage-mow-priv (in mow-priv)
 and dsp-storage-aws-dev-newvpc (mw-dev) – Environment is config
ured only with private subnets, there are no public subnets. dex
-base installation failed, events: [\{"ResourceKind":"Service","
ResourceName":"dex-base-nginx-56g288bq-controller","ResourceName
space":"dex-base-56g288bq","Message":"Error syncing load balance
r: failed to ensure load balancer: could not find any suitable s
ubnets for creating the ELB","Timestamp":"2024-02-08T09:55:28Z"}
]
```

When creating a service and enabling a public load balancer, configure at least one public subnet in the environment. For more information, see Enabling a Cloudera Data Engineering service.

DEX 11498: Spark job failing with error: "Exception in thread "main" org.apache.hadoop.fs.s3a.AWSBadRequestException:"

When users in Milan and Jakarta region use Hadoop s3a client to access AWS s3 storage, that is using s3a://bucket-name/key to access the file, an error may occur. This is a known issue in Hadoop.

Set the region manually as: spark.hadoop.fs.s3a.endpoint.region=<region code> . For region codes, see https://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region.

ENGESC-22921: Livy pod failure (CrashLoopBackoff)

There is not enough Livy-overhead-memory causing the Livy service to crash and trigger the pod to restart.

Increase the readiness/liveness timeout to 60 seconds and Livy pod will start.

DEX-11086: Cancelled statement(s) not canceled by Livy

Currently, Livy statements cant be cancelled immediately after using /sessions/{name}/statements/{id}/cancel. The status is returned as Cancelled but the background job continues to run.

There is a limitation on what can be cancelled. For example, if something is running on the driver exclusively, such as Thread.sleep()), it can not be cancelled.

DEX-9939: Tier 2 Node groups are created with the same size as Tier 1 node groups during service creation. They cannot be edited during service edit

If a service is created with 1 as the minimum on-demand scale limit, two nodes will run for Tier 1 and Tier 2. Even if a service is edited with the minimum reduced to 0, the Tier 2 node will still run. This will be fixed in the Cloudera Data Engineering 1.20 release.

You must manually edit the node group parameters from the AWS or Azure console. First, locate the log titled "Started provisioning a managed cluster, provisionerID: liftie-xxxxxxxx" and locate the Liftie ID for the cluster so that you can continue with the steps below.

In AWS:

1. Log in to the AWS Management Console.
2. Navigate to EC2 > Auto Scaling groups.
3. Find <liftie-id>-spt2-<hash>-NodeGroup and click on the name to open the Instance Group Details page.
4. Under Group Details, click Edit.
5. Update Maximum capacity from 5 to 10, and click Update.
6. Repeat the steps above to update the Maximum capacity of the <liftie-id>-cmp2-5<hash>-NodeGroup from 2 to 5. The Cluster Autoscaler implements the changes and the issue will resolve. The number of simultaneously running jobs will increase.

In Azure:

1. Navigate to the Kubernetes Service Details > Log page.

2. Navigate to the Node Pools page, and locate the Node Pool starting with cmp2.
3. Click on Scale Node pools and edit the capacity.

DEX-9852: FreeIPA certificate mismatch issues for new Spark 3.3 Virtual Clusters

In Cloudera Data Engineering 1.19, when creating a new Virtual Cluster based on Spark 3.3, and submitting any job in the pods, the following error occurs: "start TGT gen failed for user : rpc error: code = Unavailable desc = Operation not allowed while app is in recovery state."

Manually copy over the certificate from the FreeIPA server.

DEX-10147: Grafana issue for virtual clusters with the same name

In Cloudera Data Engineering 1.19, when you have two different Cloudera Data Engineering services with the same name under the same environment, and you click the Grafana charts for the second Cloudera Data Engineering service, metrics for the Virtual Cluster in the first Cloudera Data Engineering service will display.

After you have upgraded Cloudera Data Engineering, you must verify other things in the upgraded Cloudera Data Engineering cluster except the data shown in Grafana. Once you have verified everything in the new upgraded Cloudera Data Engineering service, the old Cloudera Data Engineering service should be deleted and the Grafana issue will be fixed.

DEX-9932: Name length causes Pod creation error for Cloudera Data Engineering Sessions

In Cloudera Data Engineering 1.19, the K8s pod name has a limitation of 63 Characters, and Cloudera Data Engineering Sessions has a name length of 56 maximum characters.

Create a Cloudera Data Engineering Session with a name of less than 56 characters to avoid the pod creation error.

DEX-9895: Cloudera Data Engineering Virtual Cluster API response displays default Spark version as 2.4.7

In Cloudera Data Engineering 1.19, the Spark version 3.2.3 is the expected default in a Cloudera Data Engineering Spark Virtual Cluster, but Spark 2.4.7 displays instead. This issue will be fixed in Cloudera Data Engineering 1.20.

DEX-9112: VC deployment frequently fails when deployed through the CDP CLI

In Cloudera Data Engineering 1.19, when a Virtual Cluster is deployed using the CDP CLI, it fails frequently as the pods fail to start. However, creating a Virtual cluster using the UI is successful.

Ensure that you are using proper units to --memory-requests in "cdp de" CLI, for example "--memory-requests 10Gi".

DEX-9790: Single tab Session support for Virtual Cluster selection

In Cloudera Data Engineering 1.19, the Virtual Cluster selection in the Jobs, Resources, Runs, and Sessions page is not preserved if the user attempts to open Cloudera Data Engineering in another browser tab/window.

When you open Cloudera Data Engineering in another tab, you must re-select the Virtual Cluster that you want to use in the new tab.

DEX-10044: Handle adding tier 2 auto scaling groups during in-place upgrades

Since auto scaling groups (ASGs) are not added or updated during the upgrade, the tier 2 ASGs are not created. This resulted in pods that were unable to be scheduled. This error applies to services created in Cloudera Data Engineering 1.18 and then upgraded to 1.19.

Create a new Cloudera Data Engineering service as this issue won't be seen on a new Cloudera Data Engineering 1.19 service because this error applies only to upgraded clusters.

DEX-10107: Spark 3.3 in Cloudera Data Engineering 1.19 has a limitation of characters for job names

Jobs with longer names over 23 characters can fail in Spark 3.3 with the following exception: 23-05-14 10:14:16 ERROR ExecutorPodsSnapshotsStoreImpl: Going to stop due to IllegalArgument Exception java.lang.IllegalArgumentException: '$JOB_NAME' in spark.kubernetes.executor.p

odNamePrefix is invalid. must conform https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#dns-label-names and the value length <= 47

Change the name of the job:

1. Clone the job with a new name using the Cloudera Data Engineering UI, CLI, or API.
2. Set the app name in the job itself:conf.setAppName("Custom Job Name").

DEX-10055: Interacting with a killed session

When you interact with a long-running killed Spark session, the session might become unresponsive. Refrain from interacting with the long-running killed session. This will be fixed in a future release of Cloudera Data Engineering.

DEX-9879: Infinite while loops not working in Cloudera Data Engineering Sessions

If an infinite while loop is submitted as a statement, the session will be stuck infinitely. This means that new sessions can't be sent and the Session stays in a busy state. Sample input:

```
while(True) {
  print("hello")
}
```

1. Copy and use the DEX_API that can be found on the Virtual Cluster details page to cancel the statement: POST $DEX_API/sessions/{session-name}/statements/{statement-id}/cancel. The Statement ID can be found by running the cde sessions statements command from the CDE CLI.
2. Kill the Session and create a new one.

DEX-9898: CDE CLI input reads break after interacting with a Session

After interacting with a Session through the sessions interact command, input to the CDE CLI on the terminal breaks. In this example below, ^M displays instead of proceeding:

```
> cde session interact --name sparkid-test-6
WARN: Plaintext or insecure TLS connection requested, take care
 before continuing. Continue? yes/no [no]: yes^M
```

Open a new terminal and type your Cloudera Data Engineering commands.

DEX-9881: Multi-line command error for Spark-Scala Session types in the CDE CLI

In Cloudera Data Engineering 1.19, Multi-line input into a Scala session on the CDE CLI will not work as expected, in some cases. The CLI interaction will throw an error before reading the complete input. Sample input:

```
scala> type
     |
```

Use the UI to interact with Scala sessions. A newline is expected in the above situation. In Cloudera Data Engineering 1.19, only unbalanced brackets will generate a new line. In Cloudera Data Engineering 1.20, all valid Scala newline conditions will be handled:

```
scala> customFunc(
     | (
     | )
     | )
     |
```

DEX-9756: Unable to run large raw Scala jobs

Scala code with more than 2000 lines could result in an error.

To avoid the error, increase the stack size. For example, "spark.driver.extraJavaOptions=-Xss4M", "spark.driver.extraJavaOptions=-Xss8M", and so forth.

DEX-8679: Job fails with permission denied on a RAZ environment

When running a job that has access to files is longer than the delegation token renewal time on a RAZ-enabled Cloudera environment, the job will fail with the following error:

```
Failed to acquire a SAS token for get-status on /.../words.txt d
ue to org.apache.hadoop.security.AccessControlException: Permiss
ion denied.
```

DEX-8769: The table entity type on Atlas is spar_tables instead of hive_tables on Spark3 Virtual Clusters

Tables that are created using a Spark3 Virtual Cluster on an AWS setup will have spark_tables type instead of hive_tables on Atlas Entities.

On a Spark3 Virtual Cluster, enableHiveSupport() must be called in the following way: `spark = SparkSession.builder.enableHiveSupport().getOrCreate()` You may also use Spark2 in lieu of Spark3 as this issue does not occur in Spark2.

DEX-8774: Job and run cloning is not fully supported in Cloudera Data Engineering 1.17 through 1.18.1

Currently, cloning job and runs are not supported in Cloudera Data Engineering 1.17 through 1.18.1.

Clone jobs and run operations by navigating to the Administration page, clicking View Jobs on the respective Virtual Cluster.

DEX-3706: The Cloudera Data Engineering home page not displaying for some users

The Cloudera Data Engineering home page will not display Virtual Clusters or a Quick Action bar if the user is part of hundreds of user groups or subgrooups.

The user must access the Administration page and open the Virtual Cluster of choice to perform all Job-related actions. This issue will be fixed in Cloudera Data Engineering 1.18.1

DEX-8515: The Spark History Server user interface is not visible in Cloudera Data Engineering

During job execution in Cloudera Data Engineering 1.18, the Spark History Server user interface is not visible. This error will be fixed in Cloudera Data Engineering 1.18.1.

DEX-6163: Error message with Spark 3.2 and Cloudera Data Engineering

For Cloudera Data Engineering 1.16 through 1.18, if you experience an error message, "Service account may have been revoked" with Spark 3.2 and Cloudera Data Engineering, note that this is not the core issue despite what the error message states. Look for other exceptions as it is a harmless error and only displays after a job fails due to another reason. The error message displays as part of the shutdown process. This issue will be fixed in Cloudera Data Engineering 1.18.1.

DEX-7653: Updating Airflow Job/Dag file throws a 404 error

A 404 error occurs when you update an Airflow Job/Dag file with a modified DAG ID or name when you initiate the following steps:

1. Create an Airflow job using a Simple Dag file. Use the Create Only option.
2. Edit the Airflow Job and delete the existing DAG file.
3. Upload the same DAG file with a modified DAG ID and Name from it's content.
4. Choose a different Resource Folder.
5. Use the Update and Run option.

   The 404 error occurs.

   To avoid this issue, ensure that you do not modify the DAG ID in step 3. If you must change your DAG ID in the dag file, then create a new file.

   This issue will be fixed in Cloudera Data Engineering 1.18.1.

DEX-8283: False Positive Status is appearing for the Raw Scala Syntax issue

Raw Scala jobs that fail due to syntax errors are reported as succeeded by Cloudera Data Engineering as displayed in this example:

```
spark.range(3)..show()
```

The job will fail with the following error and will be logged in the driver stdout log:

```
/opt/spark/optional-lib/exec_invalid.scala:3: error: identifier
expected but '.' found.
    spark.range(3)..show()
                   ^
```

This issue will be fixed in Cloudera Data Engineering 1.18.1.

DEX-8281: Raw Scala Scripts fail due to the use of the case class

Implicit conversions which involve implicit Encoders for case classes, that are usually supported by importing spark.implicits._, don't work in Raw Scala jobs in Cloudera Data Engineering. These include converting Scala objects, including RDD Dataset DataFrame, and Columns. For example, the following operations will fail on Cloudera Data Engineering:

```
import org.apache.spark.sql.Encoders
import spark.implicits._
case class Case(foo:String, bar:String)

// 1: an attempt to obtain schema via the implicit encoder for ca
se class fails
val encoderSchema = Encoders.product[Case].schema
encoderSchema.printTreeString()

// 2: an attempt to convert RDD[Case] to DataFrame fails
val caseDF = sc
 .parallelize(1 to 3)
 .map(i => Case(f"$i", "bar"))
 .toDF

// 3: an attempt to convert DataFrame to Dataset[Case] fails
val caseDS = spark
 .read
 .json(List("""{"foo":"1","bar":"2"}""").toDS)
 .as[Case]
```

Whereas conversions that involve implicit encoders for primitive types are supported:

```
val ds = Seq("I am a Dataset").toDS
val df = Seq("I am a DataFrame").toDF
```

Notice that List, Row, StructField, and createDataFrame are used below instead of case class and .toDF():

```
val bankRowRDD = bankText.map(s => s.split(";")).filter(s => s(0
) != "\"age\"").map(
  s => Row(
    s(0).toInt,
    s(1).replaceAll("\"", ""),
    s(2).replaceAll("\"", ""),
    s(3).replaceAll("\"", ""),
    s(5).replaceAll("\"", "").toInt
  )
)
val bankSchema = List(
  StructField("age", IntegerType, true),
```

```
  StructField("job", StringType, true),
  StructField("marital", StringType, true),
  StructField("education", StringType, true),
  StructField("balance", IntegerType, true)
)

val bank = spark.createDataFrame(
  bankRowRDD,
  StructType(bankSchema)
)


bank.registerTempTable("bank")
```

DEX-7001: When Airflow jobs are run, the privileges of the user who created the job is applied and not the user who submitted the job

If you have an Airflow job (created by User A) that contains Spark jobs, and the Airflow job is run by another user (User B), the Spark jobs are run as User A instead of the user who ran it. Regardless of who submits the Airflow job, the Airflow job is run with the user privileges of the user who created the job. This causes issues when the job submitter has lesser privileges than the job owner who has higher privileges. We recommend that the Spark and Airflow jobs must be created and run by the same user.

CDPD-40396 Iceberg migration fails on partitioned Hive table created by Spark without location

Iceberg provides a migrate procedure to migrate a Parquet/ORC/Avro Hive table to Iceberg. If the table was created using Spark and the location is not specified, and is partitioned, the migration fails.

If you are using Data Lake 7.2.15.2 or higher, the above known-issue will not occur. Otherwise, you'll need to unset the default table property of 'TRANSLATED_TO_EXTERNAL' from 'true' by completing the following:

1. Run 'ALTER TABLE ... UNSET TBLPROPERTIES ('TRANSLATED_TO_EXTERNAL') to unset the property.
2. Run the migrate procedure.

DEX-5857 Persist job owner across Cloudera Data Engineering backup restores

Currently, the user who runs the cde backup restore command has permissions, by default, to run the Jobs. This may cause Cloudera Data Engineering jobs to fail if the workload user differs from the user who runs the Jobs on Source Cloudera Data Engineering Service where the backup was performed. This failure is due to the Workload User having different privileges as the user who is expected to run the job.

Ensure that the cde job restore command is performed by the same user who is running the Cloudera Data Engineering Jobs in the Source Cloudera Data Engineering Cluster where the backup was performed. Additionally, you can ensure the User running the Restore has the same set of Permission as the User running the Job in Source Cloudera Data Engineering Cluster where the Backup was performed.

DEX-7483 User interface bug for in-place upgrade (Tech Preview)

The user interface incorrectly states that the Data Lake version 7.2.15 and above is required. The correct minimum version is 7.2.14.

DEX-6873 Kubernetes 1.21 will fail service account token renewal after 90 days

Cloudera Data Engineering on AWS running version Cloudera Data Engineering 1.14 through 1.16 using Kubernetes 1.21 will observe failed jobs after 90 days of service uptime.

Restart specific components to force regenerate the token using one of the following options:

Option 1) Using kubectl:

1. Setup kubectl for CDE.

2. Delete *CALICO-NODE* pods.

```
kubectl delete pods --selector k8s-app=calico-node --namespace
 kube-system
```

3. Delete Livy pods for all Virtual Clusters.

```
kubectl delete pods --selector app.kubernetes.io/name=livy --
all-namespaces
```

If for some reason only one Livy pod needs to be fixed:

a. Find the virtual cluster ID through the UI under Cluster Details.
b. Delete Livy pod:

```
export VC_ID=<VC ID>
kubectl delete pods --selector app.kubernetes.io/name=livy
--namespace ${VC_ID}
```

Option 2) Using K8s dashboard

1. On the Service Details page copy the RESOURCE SCHEDULER link.
2. Replace yunikorn part with the dashboard and open the resulting link in the browser.
3. In the top left corner find the namespaces dropdown and choose All namespaces.
4. Search for *CALICO-NODE*.
5. For each pod in the Pods table click the Delete option from the hamburger menu.
6. Search for *LIVY*.
7. For each pod in the Pods table click the Delete option from the hamburger menu.
8. If for some reason only one Livy pod needs to be fixed, find the Virtual Cluster ID through the UI under Cluster Details and only delete the pod with the name starting with Virtual Cluster ID.

DEX-7286 In place upgrade (Technical Preview) issue: Certificate expired showing error in browser

Certificates failure after an in-place upgrade from 1.14.

Start the certificate upgrade:

Get cluster ID

1. Navigate to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera Data Platform (Cloudera) Management Console.
2. Edit device details.
3. Copy cluster ID filed into click board.
4. In a terminal set the CID environment variable to this value.

```
export CID=cluster-1234abcd
```

Get session token

1. Navigate to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera Data Platform (Cloudera) Management Console.
2. Right click and select Inspect

3. Click the Application tab.
4. Click Cookies and select the URL of the console.
5. Select cdp-session-token.
6. Double click the displayed cookie value and right click and select Copy.
7. Open a terminal screen.

```
export CST=<Paste value of cookie here>
```

Force TLS certificate update

```
curl -b cdp-session-token=${CST}  -X 'PATCH' -H 'Content-Type: a
pplication/json' -d '{"status_update":"renewTLSCerts"}' 'https://
<URL OF CONSOLE>/dex/api/v1/cluster/$CID'
```

DEX-7051 EnvironmentPrivilegedUser role cannot be used with Cloudera Data Engineering

The role EnvironmentPrivilegedUser cannot currently be used by a user if a user wants to access Cloudera Data Engineering. If a user has this role, then this user will not be able to interact with Cloudera Data Engineering as an "access denied" would occur.

Cloudera recommends to not use or assign the EnvironmentPrivilegedUser role for accessing Cloudera Data Engineering.

CDPD-40396 Iceberg migration fails on partitioned Hive table created by Spark without location

Iceberg provides a migrate procedure for migrating a Parquet/ORC/Avro Hive table to Iceberg. If the table was created using Spark without specifying location and is partitioned, the migration fails.

By default, the table has a TRANSLATED_TO_EXTERNAL property and that is set to true. Unset this property by running ALTER TABLE ... UNSET TBLPROPERTIES ('TRANSLATED_TO_EXTERNAL') and then run the migrate procedure.

Strict DAG declaration in Airflow 2.2.5

Cloudera Data Engineering 1.16 introduces Airflow 2.2.5 which is now stricter about DAG declaration than the previously supported Airflow version in Cloudera Data Engineering. In Airflow 2.2.5, DAG timezone should be a pendulum.tz.Timezone, not datetime.timezone.utc.

If you upgrade to Cloudera Data Engineering 1.16, make sure that you have updated your DAGs according to the Airflow documentation, otherwise your DAGs will not be able to be created in Cloudera Data Engineering and the restore process will not be able to restore these DAGs.

Example of valid DAG:

```
import pendulum
dag = DAG("my_tz_dag", start_date=pendulum.datetime(2016, 1, 1,
 tz="Europe/Amsterdam"))
op = DummyOperator(task_id="dummy", dag=dag)
```

Example of invalid DAG:

```
from datetime import timezone
from dateutil import parser
dag = DAG("my_tz_dag", start_date=parser.isoparse('2020-11-11T
20:20:04.268Z').replace(tzinfo=timezone.utc))
op = DummyOperator(task_id="dummy", dag=dag)
```

COMPX-5494: Yunikorn recovery intermittently deletes existing placeholders

On recovery, Yunikorn may intermittently delete placeholder pods. After recovery, there may be remaining placeholder pods. This may cause unexpected behavior during rescheduling.

There is no workaround for this issue. To avoid any unexpected behavior, Cloudera suggests removing all the placeholders manually before restarting the scheduler.

DWX-8257: Cloudera Data Warehouse Airflow Operator does not support SSO

Although Virtual Warehouse (VW) in Cloudera Data Warehouse supports SSO, this is incompatible with the Cloudera Data Engineering Airflow service as, for the time being, the Airflow Cloudera Data Warehouse Operator only supports workload username/password authentication.

Disable SSO in the VW.

COMPX-7085: Scheduler crashes due to Out Of Memory (OOM) error in case of clusters with more than 200 nodes

Resource requirement of the YuniKorn scheduler pod depends on cluster size, that is, the number of nodes and the number of pods. Currently, the scheduler is configured with a memory limit of 2Gi. When running on a cluster that has more than 200 nodes, the memory limit of 2Gi may not be enough. This can cause the scheduler to crash because of OOM.

Important: This change requires K8s cluster access.

Increase resource requests and limits for the scheduler. Edit the YuniKorn scheduler deployment to increase the memory limit to 16Gi.

For example:

```
resources:
  limits:
    cpu: "4"
    memory: 16Gi
 requests:
    cpu: "2"
    memory: 8Gi
```

COMPX-6949: Stuck jobs prevent cluster scale down

Because of hanging jobs, the cluster is unable to scale down even when there are no ongoing activities. This may happen when some unexpected node removal occurs, causing some pods to be stuck in Pending state. These pending pods prevent the cluster from downscaling.

Terminate the jobs manually.

DEX-3997: Python jobs using virtual environment fail with import error

Running a Python job that uses a virtual environment resource fails with an import error, such as:

```
Traceback (most recent call last):
  File "/tmp/spark-826a7833-e995-43d2-bedf-6c9dbd215b76/app.py",
line 3, in <module>
    from insurance.beneficiary import BeneficiaryData
ModuleNotFoundError: No module named 'insurance'
```

Do not set the spark.pyspark.driver.python configuration parameter when using a Python virtual environment resource in a job.

Technical service bulletins

Learn about the technical service bulletins (TSBs) with the Cloudera Data Engineering service on Public Clouds, the impact or changes to the functionality, and the workaround.

TSB 2024-768: CDE graphs appear empty due to internal Prometheus OOM errors

When using a CDE service that underwent an in-place upgrade from CDE 1.19.x to CDE 1.20.3/1.21.0), charts on the CDE administration page and Grafana dashboard might not show data. This issue occurs when the CDE internal Prometheus server encounters out of memory (OOM) errors, which is caused by the pre-upgrade Prometheus configurations not being carried over during upgrade. The upgrade path from CDE 1.20.3 to CDE 1.21.0 is not affected.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Base article: TSB 2024-768: CDE Graphs appear empty due to internal Prometheus OOM errors

TSB 2024-727: Upgrade YuniKorn in CDE deployments

Apache YuniKorn (YuniKorn) 1.3.0, deployed as part of Cloudera Data Engineering (CDE) versions 1.19.3 and 1.19.4, can stop scheduling pods that cause workloads to get stuck even when cluster resources are available. Issues were found in the YuniKorn 1.3.0 release that get triggered by Spark jobs using gang scheduling in combination with preemption. These issues lead to accounting leaks and can result in YuniKorn stopping the scheduling of all pods in the cluster.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Base article: TSB 2024-727: Upgrade YuniKorn in CDE deployments

TSB 2023-658: Within CDP Public Cloud environments, CDE CLI access key authentication does not work for a[0-9].*, b[1-9].*, c[1-9].* workload domains

Using the Cloudera Data Engineering (CDE) command line interface (CLI) with access key-based authentication may cause authentication errors within CDP Public Cloud environments. The cause of the issue is the discrepancy in the way JSON Web Token (JWT) audience claim value is obtained between different components within the Cloudera Data Platform (CDP) Public Cloud.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Base article: TSB 2023-658: Within CDP Public Cloud environments, CDE CLI access key authentication does not work for a[0-9].*, b[1-9].*, c[1-9].* workload domains

TSB 2022-588: Kubeconfig and new version of aws-iam-authenticator

Regenerate Kubeconfig and in conjunction use a newer version of aws-iam-authenticator on AWS. Kubeconfig in Cloudera Data Platform (CDP) Public Cloud Data Services needs to be regenerated because the Kubeconfig generated before June 15, 2022 uses an old APIVersion (client.authentication.k8s.io/v1alpha1) which is no longer supported. This causes compatibility issues with aws-iam-authenticator starting from v0.5.7. To be able to use the new aws-iam-authenticator, the Kubeconfig needs to be regenerated.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Base article: TSB 2022-588: Kubeconfig and new version of aws-iam-authenticator

TSB 2022-587: CDE 1.14 and above using Kubernetes 1.21 will fail service account token renewal after 90 days

Cloudera Data Engineering (CDE) on Amazon Web Services (AWS) running version CDE 1.14 and above using Kubernetes 1.21 will observe failed jobs after 90 days of service uptime [1].

[1] “For Amazon Elastic Kubernetes Service (EKS) clusters, the extended expiry period is 90 days. Your Amazon EKS cluster's Kubernetes API server rejects requests with tokens older than 90 days.”

Knowledge article

For the latest update on this issue see the corresponding Knowledge Base article: TSB 2022-587: CDE 1.14 and above using Kubernetes 1.21 will fail service account token renewal after 90 days

Limitations

Cloudera Data Engineering has the following limitations.

Tier 2 node groups cannot be edited through the UI or API

Once a Tier 2 node group is created as part of service creation, the Tier 2 node group cannot be edited.

No support for Data Lake resizing

Cloudera Data Engineering does not support Data Lake resizing.

Running raw Scala code in Cloudera Data Engineering

- When setting the Log Level from the user interface, the setting is not applied to the raw Scala jobs.
- Do not use package <something> in the raw Scala job file as Raw Scala File is used for Scripting and not for Jar development and packaging.

Cloudera Data Engineering Runtime end of support

Learn about Cloudera Data Engineering Runtime end of support for Spark.

The following table specifies the planned End of Support (EoS) policy schedule for Spark. All future dates are provided for planning purposes only and are subject to change. In each case, the projected EoS date can be considered to be the last day of the month specified in the Cloudera Data Engineering Runtime end of support information table.

Table 1: Cloudera Data Engineering Runtime end of support information

Public Cloud Runtime version	Data Lake support	End of support	Long-term support	Notes
Spark 2.4.8	Up to 7.2.18	September 2025	Yes	Deprecated (will only receive bug fixes and security patches)
Spark 3.2.3	Up to 7.2.18	September 2025	-	-
Spark 3.3.x	Up to 7.2.18	September 2025	-	-
Spark 3.5.x	Starting from 7.2.18	Will follow Data Lake support policy	Yes	-

Frequently Asked Questions (FAQs)

What is the relationship between Cloudera Data Engineering Spark runtimes and the Data Lake version?

Each Cloudera Data Engineering release is certified against one or more Data Lake versions. This ensures full compatibility between the Spark engine and the underlying platform capabilities, such as the Hive Catalog service and Ranger for security policies. Each Spark version is compatible with a Data Lake version. For more information, see Compatibility for Cloudera Data Engineering and Runtime components.

Important: It is critical to update both Cloudera Data Engineering and Data Lake on a regular basis to ensure continued support and compatibility.

Does Cloudera Data Engineering offer Long-term support (LTS) releases?

Cloudera Data Engineering offers LTS through underlying Spark runtimes. When running Spark jobs within Cloudera Data Engineering, you have the option to choose an older Spark version. Specific versions of Spark are designated as

LTS. This allows you to continue running Spark jobs without any code changes. Since Cloudera Data Engineering job management APIs remain backwards compatible, existing automations are not impacted.

What is the EoS timeline for Spark runtimes designated LTS?

Spark runtimes designated as LTS in the Public Cloud have certification that spans through multiple Data Lake versions and are limited to minor enhancements and critical bug and security enhancements. To use a Spark runtime that is tagged as LTS, you need to run a supported Cloudera Data Engineering version with a corresponding certified and currently supported Data Lake version.

What is the EOS timeline for non-LTS Spark runtimes?

Spark runtimes that are not designated as LTS will be limited to the support lifetime of the corresponding certified Data Lake version. For more information, see Compatibility for Cloudera Data Engineering and Runtime components.

What is the support timeline for a given Data Lake version?

See the Cloudera - Public Cloud section in Support lifecycle policy.

Compatibility for Cloudera Data Engineering and Runtime components

Learn about Cloudera Data Engineering and compatibility for Runtime components across different versions. This document also includes component version compatibility information for AWS Graviton.

Table 2: Cloudera Data Engineering compatibility with Runtime component details

Cloudera Data Engineering Version	Spark 2.4.x	Spark 3.2.x	Spark 3.3.x	Spark 3.5.x	Airflow	Kubernetes
1.23.0	• Spark 2.4.8 • Python 2.7 and 3.6 • Iceberg 1.3.1 • Data Lake 7.2.16-7.2.18	• Spark 3.2.3 • Python 3.6 • Iceberg 1.3.1 • Data Lake 7.2.16-7.2.18	• Spark 3.3.0 • Python 3.8 • Iceberg 1.3.1 • Data Lake 7.2.16-7.2.18	• Spark 3.5.1 • Python 3.8 • Iceberg 1.5.2 • Data Lake 7.2.18	• Airflow 2.9.3 • Python 3.11	1.30
1.22.0	• Spark 2.4.8 • Python 2.7 and 3.6 • Iceberg 1.3.1 • Data Lake 7.2.16-7.2.18	• Spark 3.2.3 • Python 3.6 • Iceberg 1.3.1 • Data Lake 7.2.16-7.2.18	• Spark 3.3.0 • Python 3.8 • Iceberg 1.3.1 • Data Lake 7.2.16-7.2.18	• Spark 3.5.1 • Python 3.8 • Iceberg 1.4.3 • Data Lake 7.2.18	• Airflow 2.7 • Python 3.8	1.28
1.21	• Spark 2.4.8 • Python 2.7 and 3.6 • Iceberg 1.3.1 • Data lake 7.2.16-7.2.18	• Spark 3.2.3 • Python 3.6 • Iceberg 1.3.1 • Data lake 7.2.16-7.2.18	• Spark 3.3.0 • Python 3.8 • Iceberg 1.3.1 • Data lake 7.2.16-7.2.18	N/A	• Airflow 2.7 • Python 3.8	1.27
1.20.3	• Spark 2.4.8 • Python 2.7 • Iceberg 1.3.1 • Data lake 7.2.15-7.2.17	• Spark 3.2.3 • Python 3.6 • Iceberg 1.3.1 • Data lake 7.2.15-7.2.17	• Spark 3.3.0 • Python 3.8 • Iceberg 1.3.1 • Data lake 7.2.15-7.2.17	N/A	• Airflow 2.6.3 • Python 3.8	1.27

Cloudera Data Engineering Version	Spark 2.4.x	Spark 3.2.x	Spark 3.3.x	Spark 3.5.x	Airflow	Kubernetes
1.19.4	• Spark 2.4.8 • Scala 2.11.12 • Java 1.8.0_352 • Python 2.7 • Data lake 7.2.15-7.2.17 • Iceberg 1.1.0	• Spark 3.2.3 • Scala 2.12.10 • Java 11.0.20 • Python 3.6 • Data lake 7.2.15-7.2.17 • Iceberg 1.1.0	• Spark 3.3.0 • Scala 2.12.15 • Java 11.0.20 • Python 3.8 • Data lake 7.2.15-7.2.17 • Iceberg 1.1.0	N/A	• Airflow 2.3.4 • Python 3.8	1.26
1.19.3	• Spark 2.4.8 • Scala 2.11.12 • Java 1.8.0_352 • Python 2.7 • Data lake 7.2.15-7.2.17	• Spark 3.2.3 • Scala 2.12.10 • Java 11.0.20 • Python 3.6 • Data lake 7.2.15-7.2.17	• Spark 3.3.0 • Scala 2.12.15 • Java 11.0.20 • Python 3.8 • Data lake 7.2.15-7.2.17	N/A	• Airflow 2.3.4 • Python 3.8	1.25
1.19.2	• Spark 2.4.8 • Scala 2.11.12 • Java 1.8.0_352 • Python 2.7 • Data lake 7.2.14-7.2.16 • Iceberg 1.1.0	• Spark 3.2.3 • Scala 2.12.10 • Java 11.0.20 • Python 3.6 • Data lake 7.2.14-7.2.16 • Iceberg 1.1.0	• Spark 3.3.0 • Scala 2.12.15 • Java 11.0.20 • Python 3.8 • Data lake: 7.2.16 • Iceberg 1.1.0	N/A	• Airflow 2.3.4 • Python 3.8	1.25
1.19	• Spark 2.4.8 • Scala 2.11.12 • Java 1.8.0_352 • Python 2.7 • Data lake 7.2.14-7.2.16 • Iceberg 1.1.0	• Spark 3.2.3 • Scala 2.12.10 • Java 11.0.20 • Python 3.6 • Data lake 7.2.14-7.2.16 • Iceberg 1.1.0	• Spark 3.3.0 • Scala 2.12.15 • Java 11.0.20 • Python 3.8 • Data lake: 7.2.16 • Iceberg 1.1.0	N/A	• Airflow 2.3.4 • Python 3.8	1.25
1.18.3	• Spark 2.4.8 • Scala: 2.11.12 • Java 1.8.0_352 • Python 2.7 • Data lake 7.2.14-7.2.16 • Iceberg 0.14.1	• Spark 3.2.3 • Scala 2.12.10 • Java 11.0.20 • Python 3.6 • DL: 7.2.14-7.2.16 • Iceberg 0.14.1	N/A	N/A	• Airflow 2.3.4 • Python 3.8	1.24
1.18.1	• Spark 2.4.8 • Scala 2.11.12 • Java 1.8.0_352 • Python 2.7 • Data lake 7.2.14-7.2.16 • Iceberg 0.14.1	• Spark 3.2.0 • Scala 2.12.10 • Java 11.0.20 • Python 3.6 • Data lake 7.2.14-7.2.16 • Iceberg 0.14.1	N/A	N/A	• Airflow 2.3.4 • Python 3.8	1.23

Cloudera Data Engineering Version	Spark 2.4.x	Spark 3.2.x	Spark 3.3.x	Spark 3.5.x	Airflow	Kubernetes
1.18	• Spark 2.4.8 • Scala 2.11.12 • Java 1.8.0_352 • Python 2.7 • Data lake 7.2.14-7.2.15 • Iceberg 0.14.1	• Spark 3.2.0 • Data lake 7.2.14-7.2.15 • Iceberg 0.14.1	N/A	N/A	• Airflow 2.2.5 • Python 3.8	1.23
1.17-h1	• Spark 2.4.8 • Scala 2.11.12 • Java 1.8.0_352 • Python 2.7 • Iceberg 0.14.1	• Spark 3.2.1 • Iceberg 0.14.1	N/A	N/A	• Airflow 2.2.5 • Python 3.8	1.23

AWS Graviton component version compatibility

To use AWS Graviton in Cloudera Data Engineering, you need the following runtime component versions:

- Spark 3.3 and 3.5
- Data Lake 7.2.18

Using the Cloudera Runtime Maven repository for Cloudera Data Engineering

When building Spark applications to run on Cloudera Data Engineering, you can use the following table to determine which artifact versions to use from the Cloudera Runtime Maven repository.

You can determine the Cloudera Data Engineering version for your virtual cluster by clicking the Cluster Details icon for the virtual cluster you want to use.

The following is a sample POM (pom.xml) file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM
/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <repositories>
    <repository>
      <id>cloudera</id>
      <url>https://repository.cloudera.com/artifactory/cloudera-repos/</url>
    </repository>
  </repositories>
</project>
```

Cloudera Data Engineering version	Cloudera Runtime Maven repository
• 1.20.3 • 1.19.4 • 1.19.3 • 1.19.2	• Cloudera Runtime 7.2.15 • Cloudera Runtime 7.2.16

Cloudera Data Engineering version	Cloudera Runtime Maven repository
• 1.19	• Cloudera Runtime 7.2.16 • Cloudera Runtime 7.2.15
• 1.18 • 1.17 • 1.16	Cloudera Runtime 7.2.15
• 1.15	Cloudera Runtime 7.2.14
• 1.14	Cloudera Runtime 7.2.11.0
• 1.13	Cloudera Runtime 7.2.11.0
• 1.12 • 1.11 • 1.9	Cloudera Runtime 7.2.10.0
• 1.8 • 1.7	Cloudera Runtime 7.2.8.0
• 1.6 • 1.5 • 1.4 • 1.3 • 1.2	Cloudera Runtime 7.2.2.0

1.19.

Creating and managing Cloudera Data Engineering Sessions

Date published: 2023-04-05
Date modified: 2023-04-13

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating Sessions in Cloudera Data Engineering..... 4

Interacting with a Session in Cloudera Data Engineering..... 5

Connecting Sessions with the CDE CLI.....6

External IDE connectivity through Spark Connect-based sessions (Technical Preview)..... 6

Configuring external IDE Spark Connect sessions.....7
Sample code to connect to an external IDE Spark Connect session..... 10
Troubleshooting errors when working with an external IDE Spark Connect session..... 12

Viewing logs for Cloudera Data Engineering Sessions..... 12

Viewing Spark UI for Cloudera Data Engineering Sessions..... 13

Creating Sessions in Cloudera Data Engineering

A Cloudera Data Engineering Session is an interactive short-lived development environment for running Spark commands to help you iterate upon and build your Spark workloads.

About this task

The commands that are run in a Cloudera Data Engineering Session are called Statements. You can submit the Statements through the connect CLI command or the Interact tab in the Cloudera Data Engineering UI for a Session. Python and Scala are the supported Session types. Learn how to use Cloudera Data Engineering Sessions using the user interface.

Before you begin

Ensure that you are using a version of Cloudera Data Engineering 1.19 or higher for your Virtual Cluster.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Home page displays.
2. Click Sessions in the left navigation menu and then click Create Session.

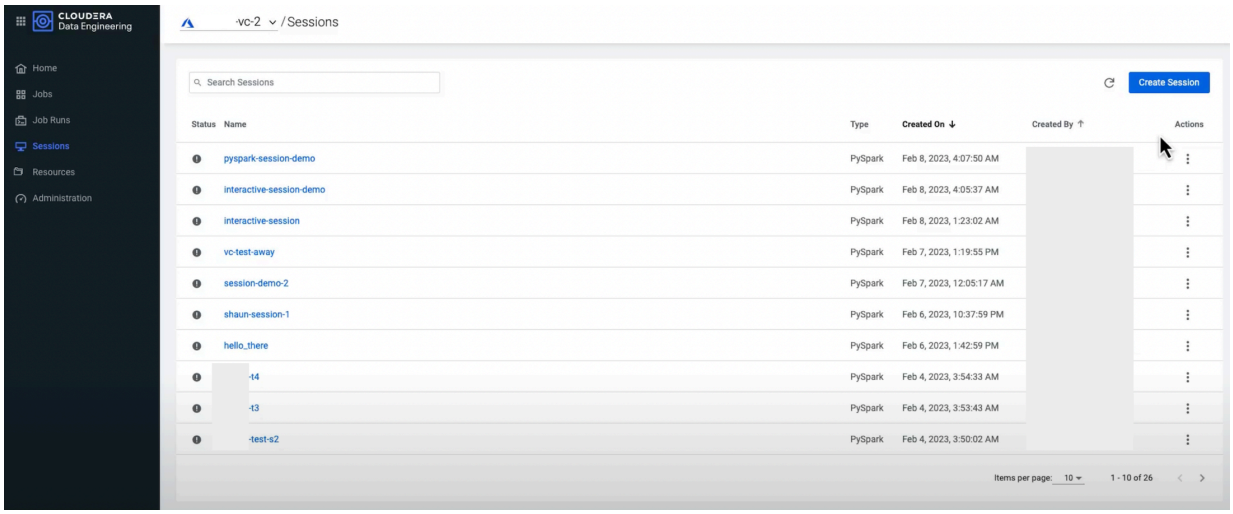

3. Enter a Name for the Session.
4. Select a Type, for example, PySpark, Scala, or Spark Connect.
5. Select a Timeout value.

   The Session will stop after the indicated time has passed.
6. Optionally, enter a Description for the Session.
7. Optionally, enter the Configurations.
8. Set the Compute options.

9. Under **Files and Resources**, you can upload Jar, Python, Egg, Zip, and other files. You can also add a resource, respositories, or a Python environment to be used in this session.

   Files that are uploaded to a session are stored in the app/mount directory.

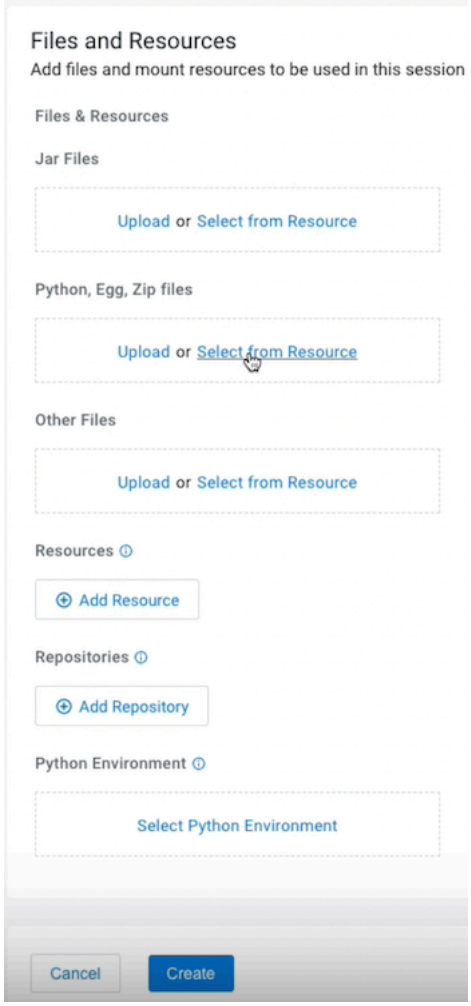

10. Click Create.

    The Connect tab displays a list of connectivity options available to interact with the Session. The Interact tab allows you to interact with the Session, and becomes available once the Session is running.

11. To delete a Session, open the Session and click Delete.

    Note: If you delete a Session, doing so will result in the termination of an active session and the loss of any attached logs and details.

Interacting with a Session in Cloudera Data Engineering

Once a Session is created in Cloudera Data Engineering, you can begin your interaction.

About this task

You can also interact with files that were uploaded to a Session. Those files are stored in the app/mount directory.

Before you begin

Ensure that you are using a version of Cloudera Data Engineering 1.19 or higher for your Virtual Cluster.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Home page displays.
2. Click Sessions in the left navigation menu.
3. Click the Interact tab.
4. In the Input box, enter your commands.
5. Click Run. A history of executed commands displays in the sequence of their execution.

Connecting Sessions with the CDE CLI

Once a Session is created in Cloudera Data Engineering, you can download the CDE CLI from the user interface then use the terminal on your local machine to set up the interactive session using the CDE CLI.

Before you begin

Ensure that you are using a version of Cloudera Data Engineering 1.19 or higher for your Virtual Cluster.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Home page displays.
2. Click Sessions in the left navigation menu.
3. Click the Connect tab.
4. Download the CDE CLI.
5. Use the CDE CLI to run and connect interactive commands:
   a) Copy the command that displays.
   b) Paste the command in the CDE CLI and continue to interact with your session.

External IDE connectivity through Spark Connect-based sessions (Technical Preview)

You can learn what an external IDE Spark Connect session is, certain known limitations, and the supported Runtime component versions.

What an external IDE Spark Connect session is

A session is an interactive short-lived development environment for running Spark commands. A Spark Connect Session is a type of Cloudera Data Engineering Session that exposes the Spark Connect interface. A Spark Connect Session allows you to connect to Spark from any remote Python environment.

Spark Connect allows you to connect remotely to the Spark clusters. Spark Connect is an API that uses the DataFrame API and unresolved logical plans as the protocol. The separation between client and server allows Spark and its open ecosystem to be leveraged from everywhere. It can be embedded in modern data applications, in IDEs and Notebooks. For more information about Spark Connect, identify the Spark version in your Virtual Cluster, and navigate to the relevant *Spark Connect Overview* page linked to that Spark version in the Spark documentation.

Supported versions of Cloudera Runtime components

Ensure that you are using Spark 3.5.1 before you use Spark Connect Sessions.

Limitations

Spark Connect Sessions do not support the following:

- Profile support: Spark Connect does not support profiles in the configuration files even though the Cloudera Data Engineering clients support "Profiles" in the configuration files.
- PySpark: In Spark 3.4, Spark Connect supports most PySpark APIs, including DataFrame, Functions, and Column. Some APIs, such as SparkContext and RDD are not supported. You can check which APIs are currently supported in the Apache Spark API Reference documentation. Supported APIs are labeled "Supports Spark Connect", so before migrating existing code to Spark Connect, you can check whether the APIs you are using are available. For more information, see the Apache Spark documentation.
- Scala: In Spark 3.5, Spark Connect supports most Scala APIs, including Dataset, functions, Column, Catalog, and KeyValueGroupedDataset. For more information, see the Apache Spark documentation.
- User-Defined Functions (UDFs) are supported, by default, for the shell and in standalone applications, with additional setup requirements.
- The majority of the Streaming API is supported, including DataStreamReader, DataStreamWriter, StreamingQuery, and StreamingQueryListener. For more information, see the Apache Spark documentation.
- APIs, such as SparkContext and RDD are deprecated in all Spark Connect versions.

Configuring external IDE Spark Connect sessions

Learn about how to configure a Spark Connect Session with Cloudera Data Engineering.

Before you begin

Before you create a Spark Connect Session, perform the following steps:

1. Enable a Cloudera Data Engineering service .
2. Create a Cloudera Data Engineering Virtual cluster. You must select All Purpose (Tier 2) in the Virtual Cluster option and Spark 3.5.1 as the Spark version.

Procedure

1. Perform the following steps on each user's machine:
   a) Create the ~/.cde/config.yaml configuration file and add the vcluster-endpoint and cdp-endpoint parameters.

      This allows the client machine to identify a virtual cluster.

      Note: The cdp-endpoint value is the same as the Cloudera console URL. From the Cloudera console URL, copy the protocol (https://), the subdomain (console), the second-level domain, and the top-level domain (.com).

      Example: https://console.[***SECOND-LEVEL-DOMAIN***].com

      Figure 1: Getting the Cloudera endpoint URL from the Cloudera console URL

      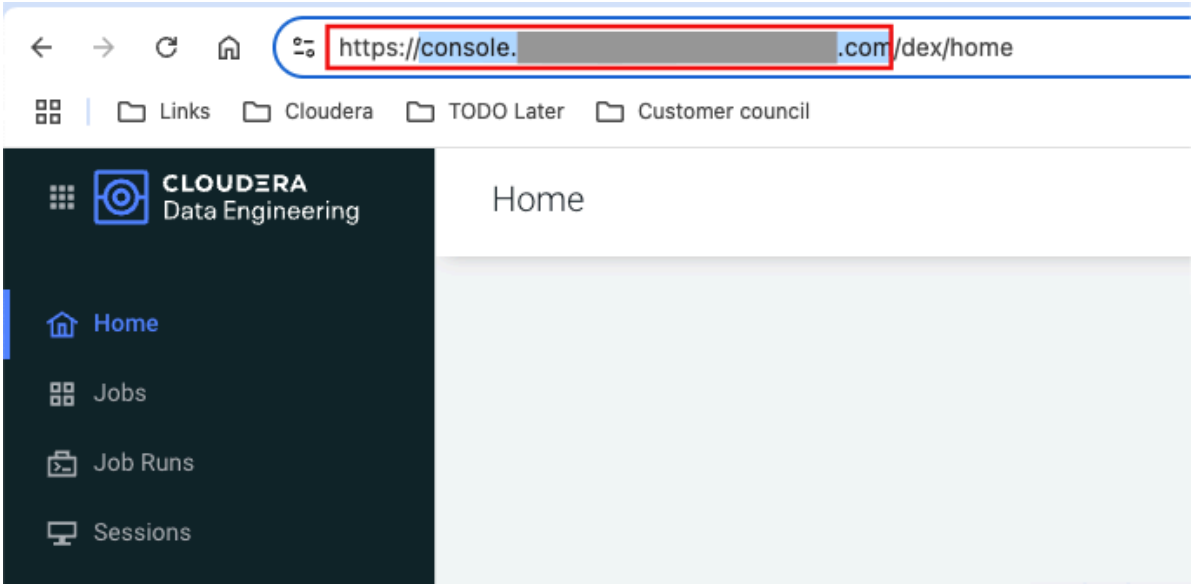

      For more information, see vcluster-endpoint and cdp-endpoint.

      For example,

      ```
      cdp-endpoint: https://console.cdp.apps.example.com
      credentials-file: /Users/user1/.cde/credentials
      vcluster-endpoint: https://ffws6v27.cde-c9b822vr.apps.example.com/dex/
      api/v1
      ```

   b) Create an access key and update the credentials-file parameter in the ~/.cde/config.yaml configuration file with the path where the credentials file is located. This allows the client machine to acquire the short-lived access tokens.

      Note: Access keys configured with the default profile are supported.

      For example,

      ```
      [default]
      cdp_access_key_id=571ff....
      cdp_private_key=dvbYd....
      ```

2. Create a Spark Connect Session using one of the following methods:

Note: You can interact with a Spark Connect session that only you have created.

- Using the UI: Create a new session as per Creating Sessions in Cloudera Data Engineering but when you select the session type, select Spark Connect (Tech Preview) from the Type drop-down list.

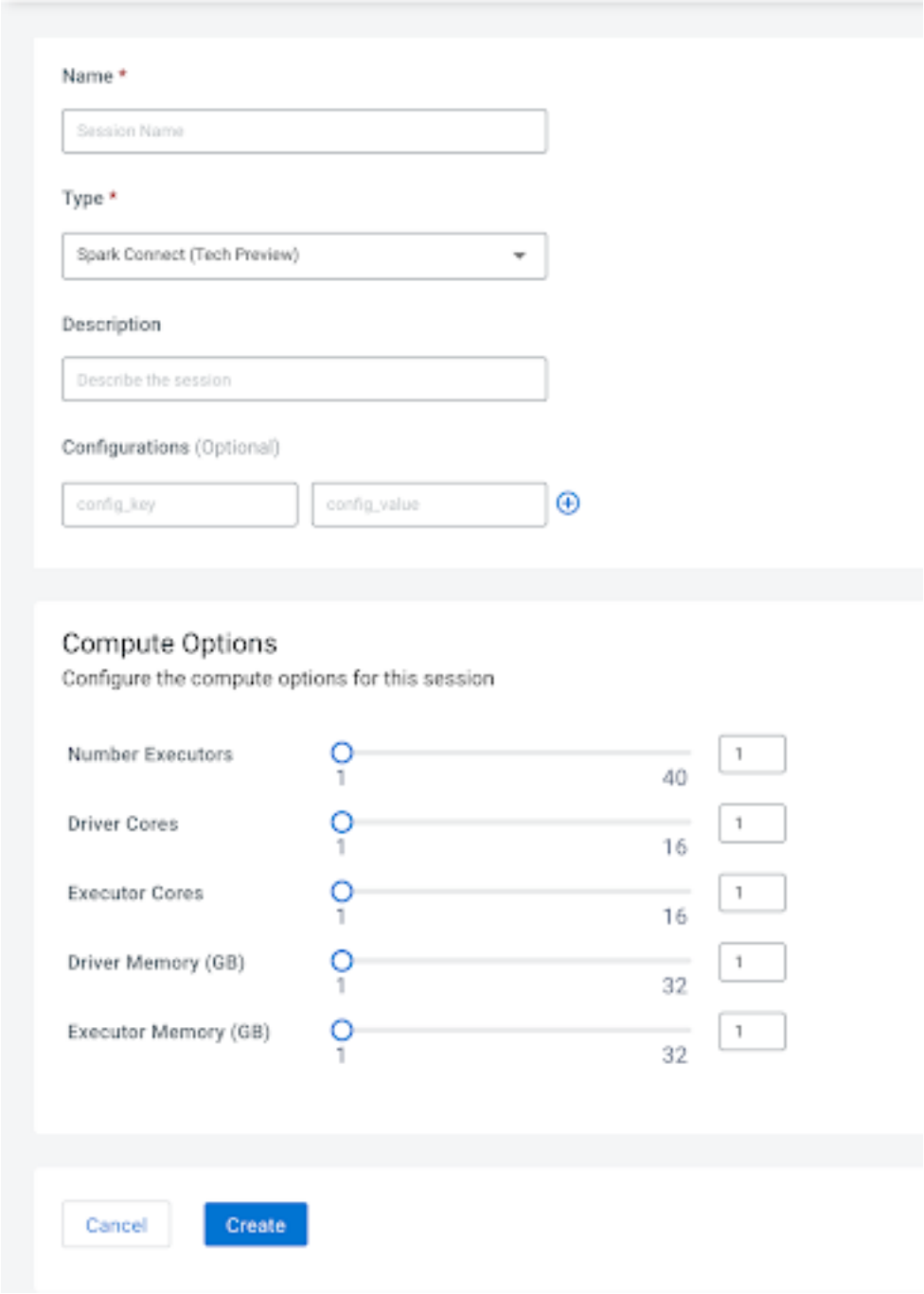

- Using the CLI: Create a Spark Connect Session by running the following command:

```
cde session create --name [***SPARK-SESSION-NAME***] --type spark-connec
t
```

Note:

To get all the attributes of a cde session command, run the `cde session -h` command.

3. On the Cloudera Data Engineering Home page, click Sessions and then select the Spark Connect Session that you have created.
4. Go to the Connect tab and download the required Cloudera Data Engineering TAR file and PySpark TAR file as displayed on the screen.

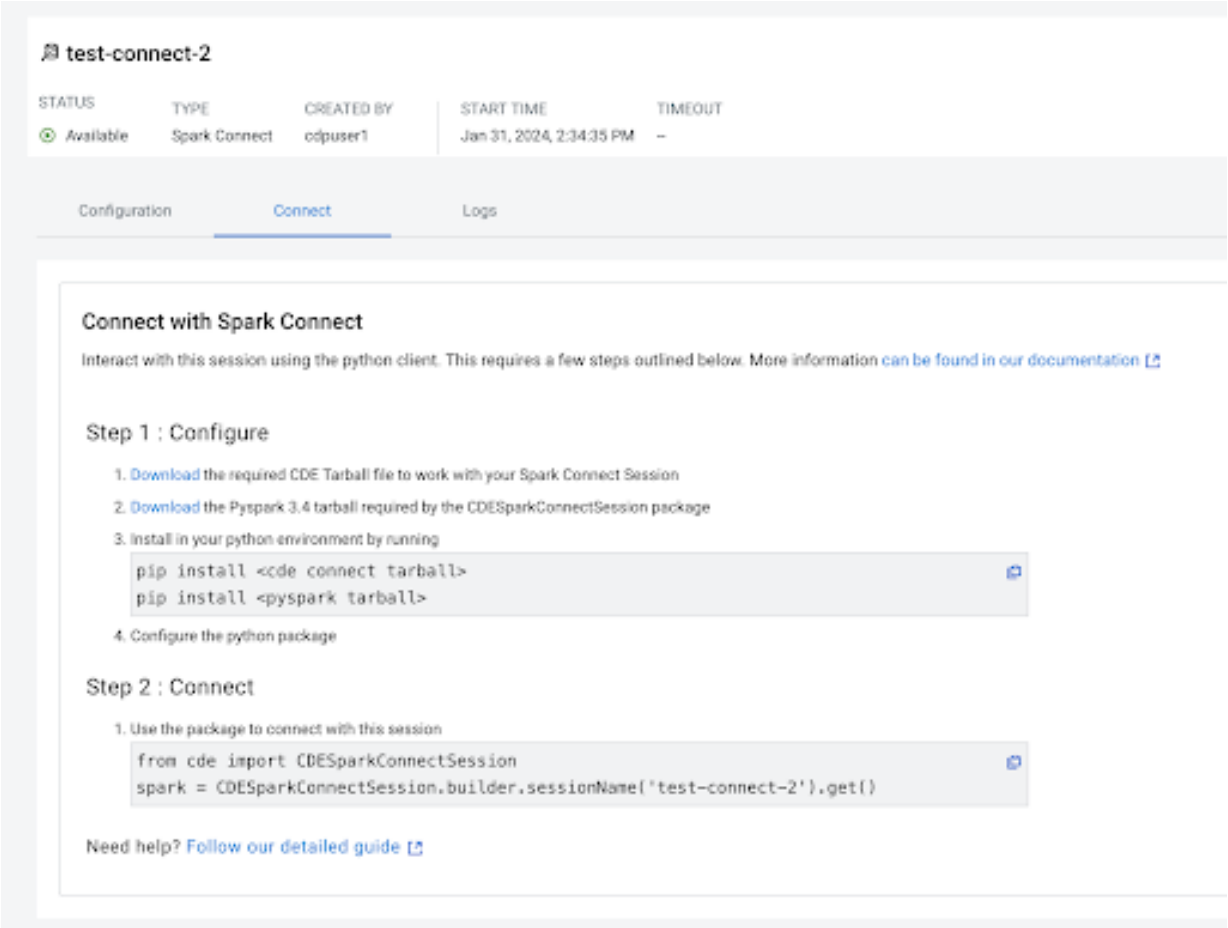

Note:

- The Copy Link option can be used to retrieve a URL and download the client using cURL.
- The PySpark TAR file version must be same as the Virtual Cluster's Spark version.

5. Create a new Python virtual environment or use your existing one and install the TAR file after activating your Python virtual environment.

```
python3 -m venv cdeconnect
. cdeconnect/bin/activate

pip install [***CDECONNECT TARBALL***]
pip install [***PYSPARK TARBALL***]
```

Sample code to connect to an external IDE Spark Connect session

After configuring Spark Connect Sessions, learn how you can run the CLI commands from a remote Python host to connect to a session and execute Spark SQL commands through an example.

You can use the following sample code to connect to the Spark Connect session. Use the spark variable to interact with Spark as you connect to the Cloudera Data Engineering jobs or sessions.

```
> python
Python 3.9.13 (main, Jul 29 2022, 12:22:24)
[Clang 13.0.0 (clang-1300.0.27.3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from cde import CDESparkConnectSession
>>> spark = CDESparkConnectSession.builder.sessionName('connect-session').ge
t()
>>> spark.version
'3.4.1.1.20.7180.0-33'
>>> spark.sql("use retaildb").show()
++
||
++
++

>>> spark.sql("select * from products_external").show()
+----------+-------------------+--------------------+-------------------+
-------------+--------------------+
|product_id|product_category_id|        product_name|product_description|pro
duct_price|       product_image|
+----------+-------------------+--------------------+-------------------+--
-----------+--------------------+
|         1|                  2|Quest Q64 10 FT. ...|                   |
     59.98|http://images.acm...|
|         2|                  2|Under Armour Men'...|                   |
     129.99|http://images.acm...|
|         3|                  2|Under Armour Men'...|                   |
     89.99|http://images.acm...|
|         4|                  2|Under Armour Men'...|                   |
     89.99|http://images.acm...|
|         5|                  2|Riddell Youth Rev...|                   |
    199.99|http://images.acm...|
|         6|                  2|Jordan Men's VI R...|                   |
     134.99|http://images.acm...|
|         7|                  2|Schutt Youth Recr...|                   |
     99.99|http://images.acm...|
|         8|                  2|Nike Men's Vapor ...|                   |
    129.99|http://images.acm...|
|         9|                  2|Nike Adult Vapor ...|                   |
      50.0|http://images.acm...|
|        10|                  2|Under Armour Men'...|                   |
     129.99|http://images.acm...|
|        11|                  2|Fitness Gear 300 ...|                   |
    209.99|http://images.acm...|
|        12|                  2|Under Armour Men'...|                   |
    139.99|http://images.acm...|
|        13|                  2|Under Armour Men'...|                   |
     89.99|http://images.acm...|
|        14|                  2|Quik Shade Summit...|                   |
     199.99|http://images.acm...|
|        15|                  2|Under Armour Kids...|                   |
     59.99|http://images.acm...|
|        16|                  2|Riddell Youth 360...|                   |
    299.99|http://images.acm...|
|        17|                  2|Under Armour Men'...|                   |
    129.99|http://images.acm...|
|        18|                  2|Reebok Men's Full...|                   |
      29.97|http://images.acm...|
|        19|                  2|Nike Men's Finger...|                   |
    124.99|http://images.acm...|
```

```
|        20|                   2|Under Armour Men'...|                   |
    129.99|http://images.acm...|
+----------+-------------------+--------------------+-------------------+-
------------+--------------------+
only showing top 20 rows
```

Troubleshooting errors when working with an external IDE Spark Connect session

While working with the Spark Connect Sessions in Cloudera Data Engineering, you might encounter errors. Learn how you can troubleshoot those errors.

Condition

If the session is killed or the driver exits due to an error when the code is being executed, Spark Connect shows the following error.

```
pyspark.errors.exceptions.connect.SparkConnectGrpcException: <_MultiThreaded
Rendezvous of RPC that terminated with:
 status = StatusCode.UNKNOWN
 details = "Stream removed"
 debug_error_string = "UNKNOWN:Error received from peer  {grpc_message:"Str
eam removed", grpc_status:2, created_time:"2024-01-31T13:28:23.35214+05:30"}
"
>
```

Remedy

Procedure

Check the actual error from the session driver logs using UI or CDE CLI.

Viewing logs for Cloudera Data Engineering Sessions

Once a Session is created in Cloudera Data Engineering, you can view logs of the Spark driver and executors from the Session.

Before you begin

Ensure that you are using a version of Cloudera Data Engineering 1.19 or higher for your Virtual Cluster.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Home page displays.
2. Click Sessions in the left navigation menu.
3. Click the Session name that you want to view the logs for.
4. Click the Logs tab.
5. In the Select Log Type drop-down, select the Driver or Executor that you want to view the logs for.
6. Click Download All Logs to save your log files.

Viewing Spark UI for Cloudera Data Engineering Sessions

You can view the Spark UI for a Cloudera Data Engineering active Session for troubleshooting purposes.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Home page displays.
2. Click Sessions in the left navigation menu.
3. Select the Virtual Cluster from the drop-down menu at the top.
4. Select the Active/Running sessions that you wan to troubleshoot.
5. Click the Spark UI tab.

Cloudera Data Engineering 1.23.0

Spark Connect Sessions (Technical Preview)

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

External IDE connectivity through Spark Connect-based sessions (Technical Preview)..... 4

Configuring external IDE Spark Connect sessions..... 4

Sample code to connect to an external IDE Spark Connect session..... 7

Troubleshooting errors when working with an external IDE Spark Connect session..... 9

External IDE connectivity through Spark Connect-based sessions (Technical Preview)

You can learn what an external IDE Spark Connect session is, certain known limitations, and the supported Runtime component versions.

What an external IDE Spark Connect session is

A session is an interactive short-lived development environment for running Spark commands. A Spark Connect Session is a type of Session that exposes the Spark Connect interface. A Spark Connect Session allows you to connect to Spark from any remote Python environment.

Spark Connect allows you to connect remotely to the Spark clusters. Spark Connect is an API that uses the DataFrame API and unresolved logical plans as the protocol. The separation between client and server allows Spark and its open ecosystem to be leveraged from everywhere. It can be embedded in modern data applications, in IDEs and Notebooks. For more information about Spark Connect, identify the Spark version in your Virtual Cluster, and navigate to the relevant *Spark Connect Overview* page linked to that Spark version in the Spark documentation.

Supported versions of Cloudera Runtime components

Ensure that you are using Spark 3.5.1 before you use Spark Connect Sessions.

Limitations

Spark Connect Sessions do not support the following:

- Profile support: Spark Connect does not support profiles in the configuration files even though the clients support "Profiles" in the configuration files.
- PySpark: In Spark 3.4, Spark Connect supports most PySpark APIs, including DataFrame, Functions, and Column. Some APIs, such as SparkContext and RDD are not supported. You can check which APIs are currently supported in the Apache Spark API Reference documentation. Supported APIs are labeled "Supports Spark Connect", so before migrating existing code to Spark Connect, you can check whether the APIs you are using are available. For more information, see the Apache Spark documentation.
- Scala: In Spark 3.5, Spark Connect supports most Scala APIs, including Dataset, functions, Column, Catalog, and KeyValueGroupedDataset. For more information, see the Apache Spark documentation.
- User-Defined Functions (UDFs) are supported, by default, for the shell and in standalone applications, with additional setup requirements.
- The majority of the Streaming API is supported, including DataStreamReader, DataStreamWriter, StreamingQuery, and StreamingQueryListener. For more information, see the Apache Spark documentation.
- APIs, such as SparkContext and RDD are deprecated in all Spark Connect versions.

Configuring external IDE Spark Connect sessions

Learn about how to configure a Spark Connect Session with .

Before you begin

Before you create a Spark Connect Session, perform the following steps:

1. Enable a service .
2. Create a Virtual cluster. You must select All Purpose (Tier 2) in the Virtual Cluster option and Spark 3.5.1 as the Spark version.

Procedure

1. Perform the following steps on each user's machine:
   a) Create the ~/.cde/config.yaml configuration file and add the vcluster-endpoint and cdp-endpoint parameters.

      This allows the client machine to identify a virtual cluster.

      Note: The cdp-endpoint value is the same as the console URL. From the console URL, copy the protocol (https://), the subdomain (console), the second-level domain, and the top-level domain (.com).

      Example: https://console.[***SECOND-LEVEL-DOMAIN***].com

      Figure 1: Getting the endpoint URL from the console URL

      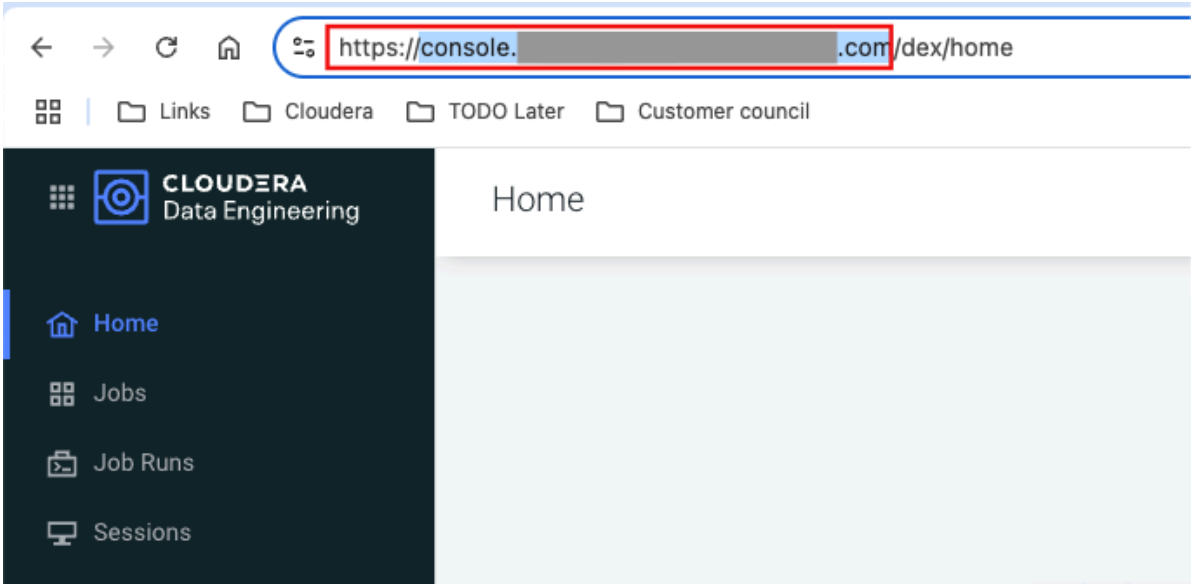

      For more information, see vcluster-endpoint and cdp-endpoint.

      For example,

      ```
      cdp-endpoint: https://console.cdp.apps.example.com
      credentials-file: /Users/user1/.cde/credentials
      vcluster-endpoint: https://ffws6v27.cde-c9b822vr.apps.example.com/dex/
      api/v1
      ```

   b) Create an access key and update the credentials-file parameter in the ~/.cde/config.yaml configuration file with the path where the credentials file is located. This allows the client machine to acquire the short-lived access tokens.

      Note: Access keys configured with the default profile are supported.

      For example,

      ```
      [default]
      cdp_access_key_id=571ff....
      cdp_private_key=dvbYd....
      ```

2. Create a Spark Connect Session using one of the following methods:

Note: You can interact with a Spark Connect session that only you have created.

- Using the UI: Create a new session as per Creating Sessions in but when you select the session type, select Spark Connect (Tech Preview) from the Type drop-down list.

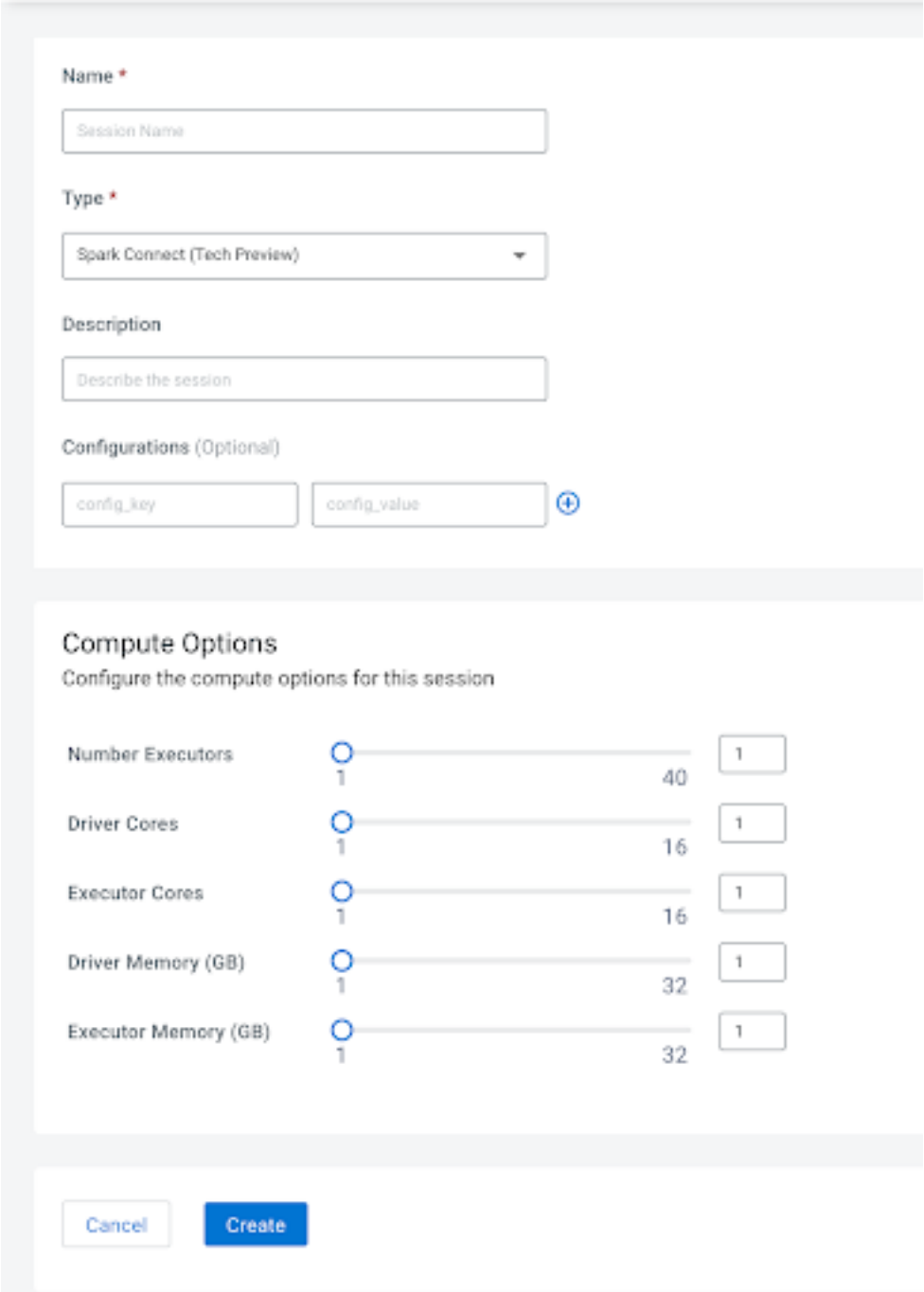

- Using the CLI: Create a Spark Connect Session by running the following command:

```
cde session create --name [***SPARK-SESSION-NAME***] --type spark-connec
t
```

Note:

To get all the attributes of a cde session command, run the `cde session -h` command.

3. On the Home page, click Sessions and then select the Spark Connect Session that you have created.
4. Go to the Connect tab and download the required TAR file and PySpark TAR file as displayed on the screen.

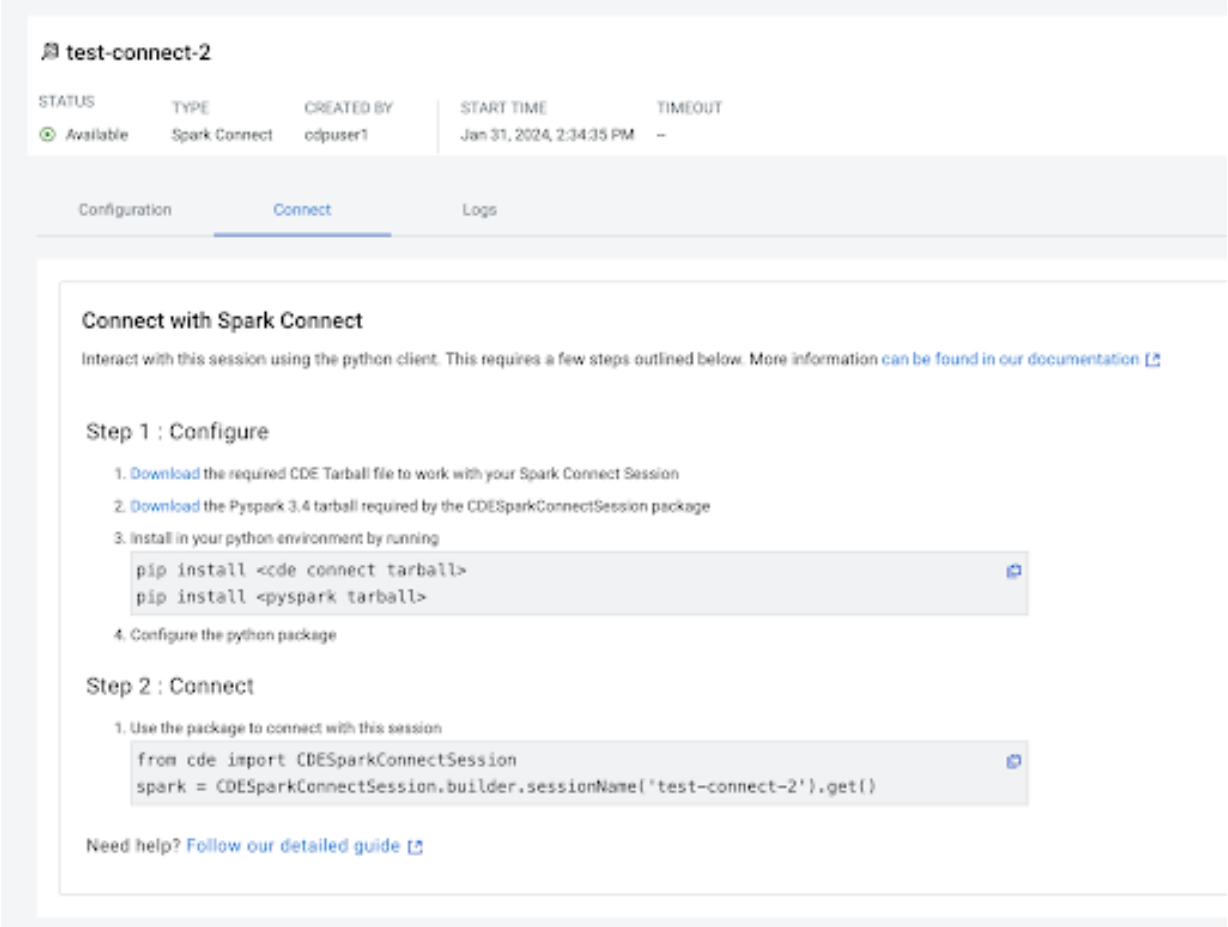

Note:

- The Copy Link option can be used to retrieve a URL and download the client using cURL.
- The PySpark TAR file version must be same as the Virtual Cluster's Spark version.

5. Create a new Python virtual environment or use your existing one and install the TAR file after activating your Python virtual environment.

```
python3 -m venv cdeconnect
. cdeconnect/bin/activate

pip install [***CDECONNECT TARBALL***]
pip install [***PYSPARK TARBALL***]
```

Sample code to connect to an external IDE Spark Connect session

After configuring Spark Connect Sessions, learn how you can run the CLI commands from a remote Python host to connect to a session and execute Spark SQL commands through an example.

You can use the following sample code to connect to the Spark Connect session. Use the spark variable to interact with Spark as you connect to the jobs or sessions.

```
> python
Python 3.9.13 (main, Jul 29 2022, 12:22:24)
[Clang 13.0.0 (clang-1300.0.27.3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from cde import CDESparkConnectSession
>>> spark = CDESparkConnectSession.builder.sessionName('connect-session').ge
t()
>>> spark.version
'3.4.1.1.20.7180.0-33'
>>> spark.sql("use retaildb").show()
++
||
++
++

>>> spark.sql("select * from products_external").show()
+----------+-------------------+--------------------+-------------------+
-------------+--------------------+
|product_id|product_category_id|        product_name|product_description|pro
duct_price|       product_image|
+----------+-------------------+--------------------+-------------------+--
-----------+--------------------+
|         1|                  2|Quest Q64 10 FT. ...|                   |
     59.98|http://images.acm...|
|         2|                  2|Under Armour Men'...|                   |
     129.99|http://images.acm...|
|         3|                  2|Under Armour Men'...|                   |
     89.99|http://images.acm...|
|         4|                  2|Under Armour Men'...|                   |
     89.99|http://images.acm...|
|         5|                  2|Riddell Youth Rev...|                   |
    199.99|http://images.acm...|
|         6|                  2|Jordan Men's VI R...|                   |
     134.99|http://images.acm...|
|         7|                  2|Schutt Youth Recr...|                   |
     99.99|http://images.acm...|
|         8|                  2|Nike Men's Vapor ...|                   |
    129.99|http://images.acm...|
|         9|                  2|Nike Adult Vapor ...|                   |
      50.0|http://images.acm...|
|        10|                  2|Under Armour Men'...|                   |
     129.99|http://images.acm...|
|        11|                  2|Fitness Gear 300 ...|                   |
    209.99|http://images.acm...|
|        12|                  2|Under Armour Men'...|                   |
    139.99|http://images.acm...|
|        13|                  2|Under Armour Men'...|                   |
     89.99|http://images.acm...|
|        14|                  2|Quik Shade Summit...|                   |
     199.99|http://images.acm...|
|        15|                  2|Under Armour Kids...|                   |
     59.99|http://images.acm...|
|        16|                  2|Riddell Youth 360...|                   |
    299.99|http://images.acm...|
|        17|                  2|Under Armour Men'...|                   |
    129.99|http://images.acm...|
|        18|                  2|Reebok Men's Full...|                   |
      29.97|http://images.acm...|
|        19|                  2|Nike Men's Finger...|                   |
    124.99|http://images.acm...|
```

```
|         20|                   2|Under Armour Men'...|                   |
    129.99|http://images.acm...|
+----------+--------------------+--------------------+--------------------+-
------------+--------------------+
only showing top 20 rows
```

Troubleshooting errors when working with an external IDE Spark Connect session

While working with the Spark Connect Sessions in , you might encounter errors. Learn how you can troubleshoot those errors.

Condition

If the session is killed or the driver exits due to an error when the code is being executed, Spark Connect shows the following error.

```
pyspark.errors.exceptions.connect.SparkConnectGrpcException: <_MultiThreaded
Rendezvous of RPC that terminated with:
 status = StatusCode.UNKNOWN
 details = "Stream removed"
 debug_error_string = "UNKNOWN:Error received from peer  {grpc_message:"Str
eam removed", grpc_status:2, created_time:"2024-01-31T13:28:23.35214+05:30"}
"
>
```

Remedy

Procedure

Check the actual error from the session driver logs using UI or CDE CLI.

Cloudera Data Engineering 1.23.0

Using spark-submit drop-in migration tool for migrating Spark workloads to Cloudera Data Engineering

Date published: 2020-07-30
Date modified: 2024-11-12

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using spark-submit drop-in migration tool for migrating Spark workloads to Cloudera Data Engineering 4
Installing and using the migration tool 4
Downloading the cde-env tool 4
Installing the cde-env tool 7
Configuring the cde-env tool 8
Using the cde-env tool 11
Run sample spark-submit command 11
Using the migration tool in a docker container 12
Configuring the cde-env tool 12
Run the migration tool in a docker container 16
Run sample spark-submit command inside the docker container 16
Known Issues and Limitations 16

Using spark-submit drop-in migration tool for migrating Spark workloads to Cloudera Data Engineering

Cloudera Data Engineering provides a command line tool cde-env to help migrate your Cloudera Spark workloads running on Cloudera Private Cloud Base (spark-on-YARN) and Cloudera Data Hub to Cloudera Data Engineering without having to completely rewrite your existing spark-submit command-lines.

Supported platforms

You can use the migration tool in the following platforms:

- Linux
- MacOS
- Windows (Docker only)

You can use the migration tool either by installing it on a gateway host or running it as a docker container.

Installing and using the migration tool

You can install and use the migration tools on a gateway host so that all the users on that machine can use the tool.

Downloading the cde-env tool

You can use the cde-env tool to migrate your spark-on-YARN workloads to Cloudera Data Engineering:

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. In the Home page, click the Migration Tool link under Resources & Guides to download the migration tool.
3. Select the Virtual Cluster.

4. Download the tool for either Linux or Mac.

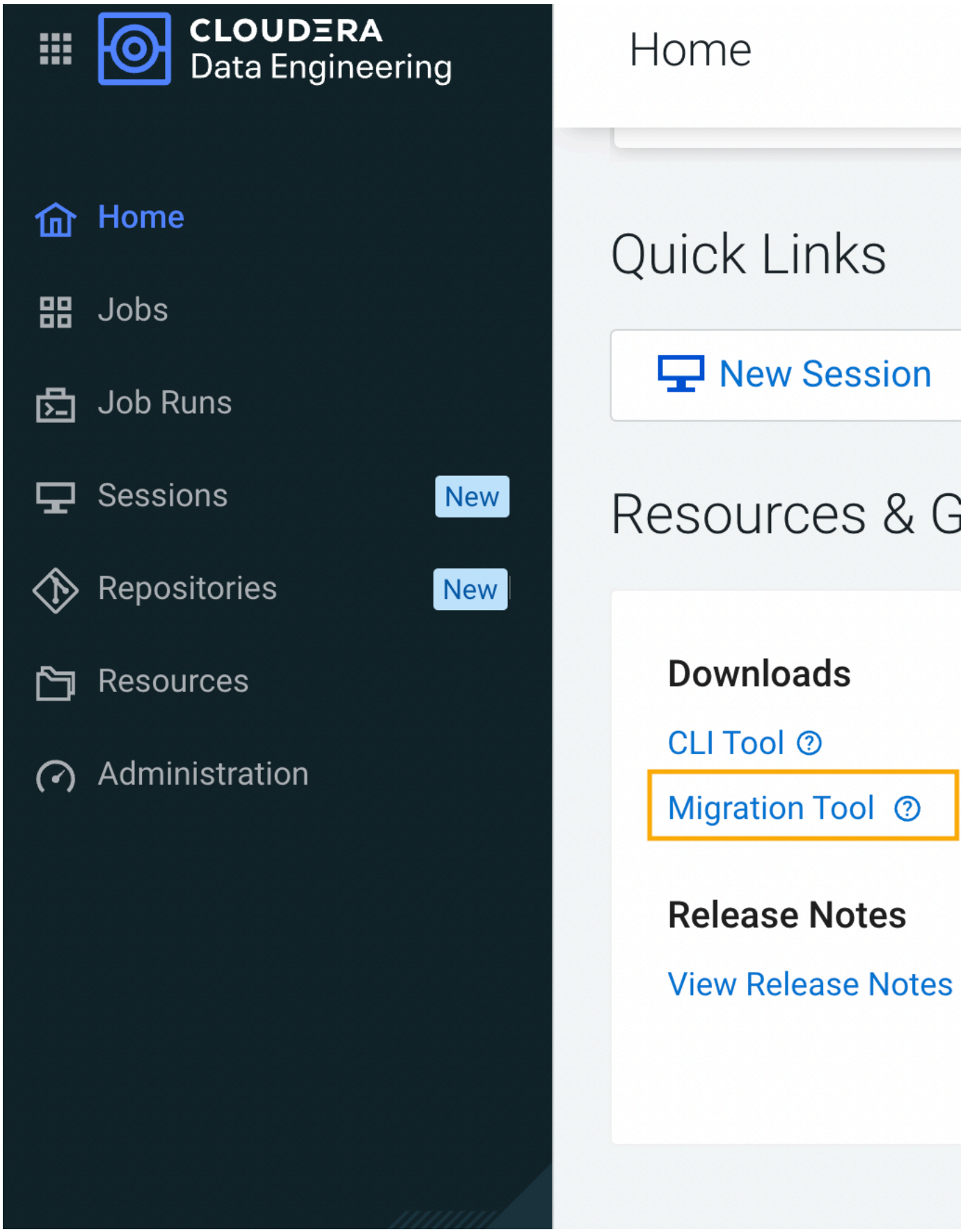
CLOUDERA
Data Engineering
Home
Jobs
Job Runs
Sessions
New
Repositories
New
Resources
Administration
Home
Quick Links
New Session
Resources & G
Downloads
CLI Tool
Migration Tool
Release Notes
View Release Notes

5. Unzip the archive.

   This is a temporary installation package and can be saved in any location. The folder contains README.md, cde, cde-env.sh, spark-submit-cde, and spark3-submit-cde files.

Installing the cde-env tool

You can install the cde-env tool as an Administrator or as a normal user. Cloudera recommends you to install the tool as an Administrator in the /opt/cloudera/bin folder so that all the users in the host can access the tool.

For Administrator

1. Install the tool by copying the required binary and script files to the /opt/cloudera/bin folder so that the migration tool can run in the current gateway host.

   ```
   $ ./cde-env.sh enable-spark-submit-proxy
   ```

   Note: If the host machine has update-alternatives installed, no need to perform the following step. You can check this by running the `update-alternatives` command.

2. Enable the installed binary and script files by granting access to those files for all users on the host using one of the following options.

   Note: If the host machine has `update-alternatives` installed, no need to perform this step. You can check this by running the `update-alternatives` command. Otherwise, perform the following:

   - Link the existing /usr/bin/spark-submit and /usr/bin/spark3-submit.

     ```
     $ ln -s /opt/cloudera/bin/cde /usr/bin/cde
     $ ln -s /opt/cloudera/bin/cde-env.sh /usr/bin/cde-env.sh
     $ ln -s -f /opt/cloudera/bin/spark-submit /usr/bin/spark-submit
     $ ln -s -f /opt/cloudera/bin/spark3-submit /usr/bin/spark3-submit
     ```

     or

   - Update PATH to point to the new installation location at the host level.

     ```
     $ export PATH=/opt/cloudera/bin:$PATH
     ```

For User

1. Install the tool on the host by running the following command:

   By default, the binary and script files will be installed in the $HOME/bin folder. You can change the location by replacing the $HOME/bin folder to the target folder in the following command.

   Linux

   ```
   $ sed -i "s#CLOUDERA_BIN=/opt/cloudera/bin#CLOUDERA_BIN=$HOME/BIN#g"
   cde-env.sh && ./cde-env.sh enable-spark-submit-proxy -f private
   ```

   MacOS

   ```
   $ sed -i '' "s#CLOUDERA_BIN=/opt/cloudera/bin#CLOUDERA_BIN=$HOME/BIN#g"
   cde-env.sh && ./cde-env.sh enable-spark-submit-proxy -f private
   ```

2. Update PATH to give access to those binary and script files.

   ```
   $ export PATH=$HOME/BIN:$PATH
   ```

Configuring the cde-env tool

The CDE env-tool uses the ~/.cde/config.yaml configuration file to manage jobs in Cloudera Data Engineering virtual clusters. You must manually edit the ~/.cde/config.yaml file and update the profiles with the required information.

For more information, see Creating and using multiple profiles .

Prerequisites for setting up the cde-env tool

You must obtain the virtual cluster endpoint URL, Cloudera endpoint URL, and generate user keys for each user whose Spark jobs you are migrating over to Cloudera Data Engineering.

Procedure

1. **Virtual Cluster Endpoint URL**: Determine the virtual cluster endpoint URL.
   a. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
   b. Click Administration in the left navigation menu, select the environment containing the virtual cluster you want to use.
   c. In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to use to migrate your spark jobs to.
   d. Click JOBS API URL to copy the URL to your clipboard.
   e. Paste the URL into a text editor to identify the endpoint host. For example, the URL is similar to the following:

      ```
      https://dfdj6kgx.cde-2cdxw5x5.apps.ecs-demo.example.com/dex/api/v1
      ```

      In the above example, the endpoint host is

      ```
      dfdj6kgx.cde-2cdxw5x5.apps.ecs-demo.example.com
      ```

2. **Cloudera Endpoint URL**: Copy the Cloudera console URL.

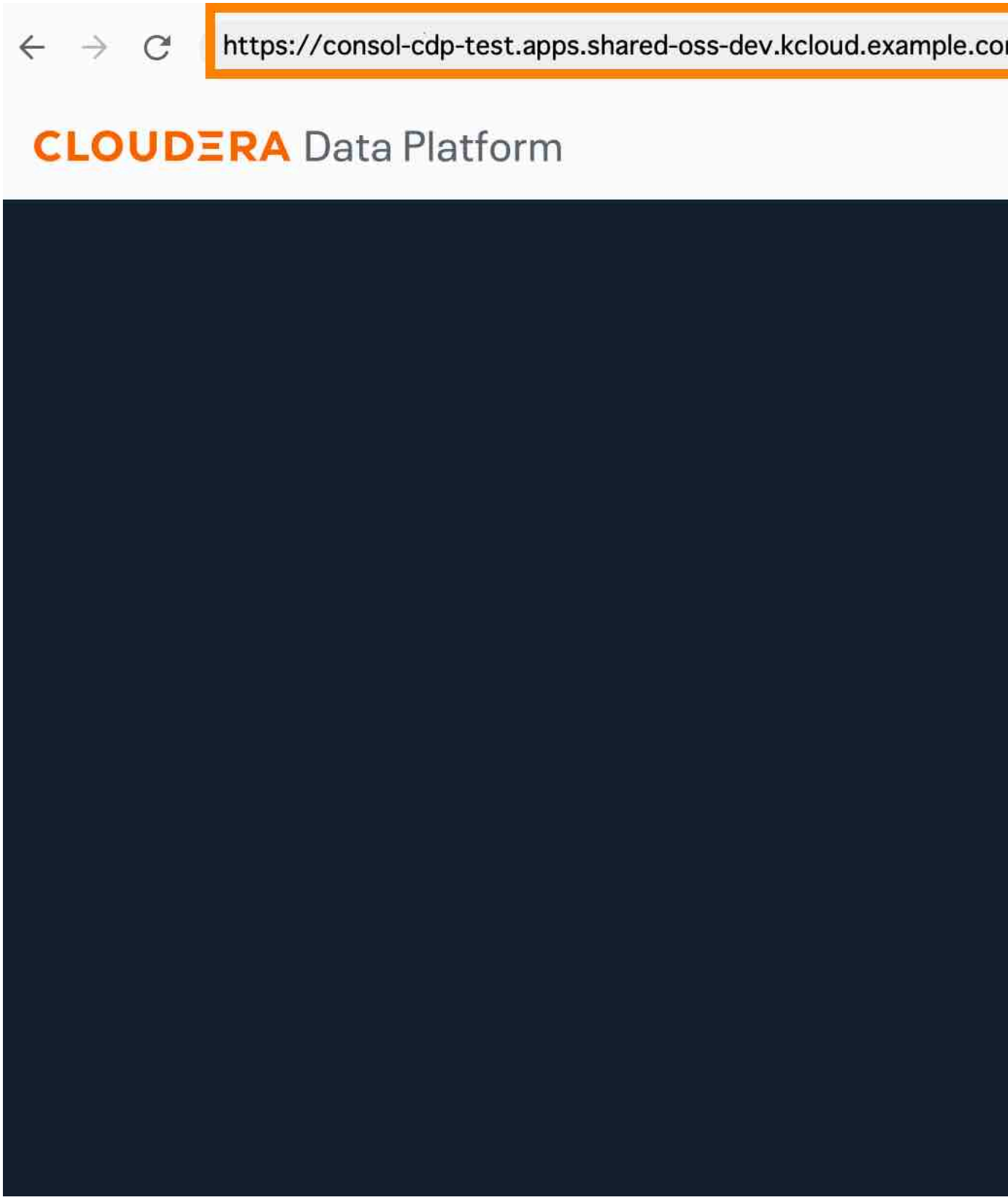

3. **Access Key**: Generate access key for each user whose Spark jobs you are migrating over to Cloudera Data Engineering:
   a. Sign in to the Cloudera console as an Administrator.
   b. In the Cloudera Home page, click Cloudera Management Console.
   c. On the left navigation menu, click Users.
   d. On the Users page, click the name of the user or the machine user account for which you want to generate an access key.
   e. On the user account page, go to the Access Keys section and click Generate Access Key.

      Cloudera creates the key and displays the information on the screen.

Adding profile for each user and creating the Credentials file

You must create a profile for each user in the ~/.cde/config.yaml file and add the access key in the Credentials file.

Procedure

1. Create the config.yaml and credentials files under ~/.cdefolder. If the ~/.cde folder does not exist, create it where the cde-env tool is installed.
2. Edit the config.yaml file. You can create multiple profiles in the ~/.cde/config.yaml file and can be used while running commands.

   Edit the ~/.cde/config.yaml file to add the allow-all-spark-submit-flags: true parameter and update the profiles.

```
# ~/.cde/config.yaml

allow-all-spark-submit-flags: true
credentials-file: <credentials-location>
cdp-endpoint: <CDP-endpoint>
tls-insecure: true

profiles:
- name: <Profile Name1>
  vcluster-endpoint: <VC-endpoint>

- name: <Profile Name2>
  vcluster-endpoint: <VC-endpoint>
```

   Example configuration file:

```
# ~/.cde/config.yaml

allow-all-spark-submit-flags: true
credentials-file: /home/cdpuser1/.cde/credentials
cdp-endpoint: https://console-xhu-141.apps.shared-os-dev-01.kcloud.example
.com
tls-insecure: true

profiles:
- name: vc-2
  vcluster-endpoint: https://5b27g4jm.cde-x6j2nh5j.apps.apps.shared-osdev-
01.kcloud.example.com/dex/api/v1/

- name: spark3-1
  vcluster-endpoint: https://7j92n8q4.cde-smstx27m.apps.apps.shared-osde
v-01.kcloud.example.com/dex/api/v1/
```

3. Add your access key information generated from the Cloudera management console in the credentials file.

   Example credentails file:

```
[default]
```

```
cdp_access_key_id=a4e8f324-5940-454c-a172-5c748e56e4c2
cdp_private_key=qpG0CzVqodKTQYXakm89bjX0606c7fP3EnAcxuy+Rzs=
```

Using the cde-env tool

You can run the spark-submit command after installing the cde-env tool. You can use the cde-env tool even without administrator privileges.

You can activate the specific profile you want to use by running the following command:

```
$ cde-env.sh activate -p <PROFILE-NAME>
```

For example, to alternatively run the the same spark-submit command either against YARN or one of the Cloudera Data Engineering virtual cluster, you can activate the relevant profile:

1. Run spark jobs on YARN by activating the yarn profile. yarn is a reserved profile name here.

```
$ cde-env.sh activate -p YARN
```

2. Run spark jobs on a Cloudera Data Engineering virtual cluster configured by CDE CLI profile named vc-1:

```
$ cde-env.sh activate -p VC-1
```

Switching profiles back and forth this way lets you run the same spark-submit command either against YARN or one of the Cloudera Data Engineering virtual clusters.

Run sample spark-submit command

After you activate the profile using the cdp-env tool, you can run your spark-submit commands on Cloudera Data Engineering without completely rewriting your existing spark-on-yarn command lines.

- Sample spark-submit commands you can run on the Cloudera Data Engineering workloads.

```
$ spark-submit \
--name pt_rpt_streams \
--master=yarn --deploy-mode=cluster \
--driver-memory 4G \
--executor-memory 4G --executor-cores 3 \
--num-executors 4 \
--files "$HOME/spark-sql.py" \
--conf "SPARK.EXECUTOR.EXTRAJAVAOPTIONS=-
DJAVA.SECURITY.AUTH.LOGIN.CONFIG=/HOME/HDPSPARKPRD/SPARK-
HDPSPARKPRDKEYTAB-JAAS.CONF -DJAVA.SECURITY.KRB5.CONF=/ETC/KRB5.CONF -
DJAVAX.SECURITY.AUTH.USESUBJECTCREDSONLY=TRUE" \
--conf "SPARK.DRIVER.EXTRAJAVAOPTIONS=-DJAVA.SECURITY.AUTH.LOGIN.CONFIG=/
HOME/HDPSPARKPRD/SPARK-HDPSPARKPRDKEYTAB-JAAS.CONF
 -DJAVA.SECURITY.KRB5.CONF=/ETC/KRB5.CONF -
DJAVAX.SECURITY.AUTH.USESUBJECTCREDSONLY=TRUE" \
--co
nf "SPARK.IO.COMPRESSION.CODEC=ORG.APACHE.SPARK.IO.LZ4COMPRESSIONCODEC" \
$HOME/spark-sql.py
```

```
$ spark3-submit \
--name pt_rpt_streams \
--master=yarn --deploy-mode=cluster \
--driver-memory 4G \
--executor-memory 4G --executor-cores 3 \
--num-executors 4 \
--files "$HOME/spark-sql.py" \
--conf "SPARK.EXECUTOR.EXTRAJAVAOPTIONS=-
DJAVA.SECURITY.AUTH.LOGIN.CONFIG=/HOME/HDPSPARKPRD/SPARK-
```

```
HDPSPARKPRDKEYTAB-JAAS.CONF -DJAVA.SECURITY.KRB5.CONF=/ETC/KRB5.CONF -
DJAVAX.SECURITY.AUTH.USESUBJECTCREDSONLY=TRUE" \
--conf "SPARK.DRIVER.EXTRAJAVAOPTIONS=-DJAVA.SECURITY.AUTH.LOGIN.CONFIG=/
HOME/HDPSPARKPRD/SPARK-HDPSPARKPRDKEYTAB-JAAS.CONF
 -DJAVA.SECURITY.KRB5.CONF=/ETC/KRB5.CONF -
DJAVAX.SECURITY.AUTH.USESUBJECTCREDSONLY=TRUE" \
--c
onf "SPARK.IO.COMPRESSION.CODEC=ORG.APACHE.SPARK.IO.LZ4COMPRESSIONCODEC" \
$HOME/spark-sql.py
```

- Sample spark-submit commands with an inline profile configuration you can run on the Cloudera Data Engineering workloads.

```
$ CDE_CONFIG_PROFILE=yarn \
spark-submit \
--name pt_rpt_streams --master=yarn \
--deploy-mode=cluster --driver-memory 4G \
--executor-memory 4G --executorcores 3 \
--num-executors 4 --files "$HOME/spark-sql.py" \
--conf "SPARK.EXECUTOR.EXTRAJAVAOPTIONS=-
DJAVA.SECURITY.AUTH.LOGIN.CONFIG=/HOME/HDPSPARKPRD/SPARK-
HDPSPARKPRDKEYTAB-JAAS.CONF -DJAVA.SECURITY.KRB5.CONF=/ETC/KRB5.CONF -
DJAVAX.SECURITY.AUTH.USESUBJECTCREDSONLY=TRUE" \
--conf "SPARK.DRIVER.EXTRAJAVAOPTIONS=-DJAVA.SECURITY.AUTH.LOGIN.CONFIG=/
HOME/HDPSPARKPRD/SPARK-HDPSPARKPRDKEYTAB-JAAS.CONF
 -DJAVA.SECURITY.KRB5.CONF=/ETC/KRB5.CONF -
DJAVAX.SECURITY.AUTH.USESUBJECTCREDSONLY=TRUE" \
--c
onf "SPARK.IO.COMPRESSION.CODEC=ORG.APACHE.SPARK.IO.LZ4COMPRESSIONCODEC" \
$HOME/spark-sql.py
```

```
$ CDE_CONFIG_PROFILE=vc-1 \
spark3-submit \
--name pt_rpt_streams \
--master=yarn --deploy-mode=cluster \
--driver-memory 4G --executor-memory 4G \
--executor-cores 3 --num-executors 4 \
--files "$HOME/spark-sql.py" \
--conf "SPARK.EXECUTOR.EXTRAJAVAOPTIONS=-
DJAVA.SECURITY.AUTH.LOGIN.CONFIG=/HOME/HDPSPARKPRD/SPARK-
HDPSPARKPRDKEYTAB-JAAS.CONF -DJAVA.SECURITY.KRB5.CONF=/ETC/KRB5.CONF -
DJAVAX.SECURITY.AUTH.USESUBJECTCREDSONLY=TRUE" \
--conf
"SPARK.DRIVER.EXTRAJAVAOPTIONS=-DJAVA.SECURITY.AUTH.LOGIN.CONFIG=/HOME/
HDPSPARKPRD/SPARK-HDPSPARKPRDKEYTAB-JAAS.CONF -DJAVA.SECURITY.KRB5.CONF=/
ETC/KRB5.CONF -DJAVAX.SECURITY.AUTH.USESUBJECTCREDSONLY=TRUE" \
--c
onf "SPARK.IO.COMPRESSION.CODEC=ORG.APACHE.SPARK.IO.LZ4COMPRESSIONCODEC" \
$HOME/spark-sql.py
```

Using the migration tool in a docker container

You can run the docker image in an interactive mode after you mount the config.yaml and credentials files into the docker container.

Configuring the cde-env tool

The CDE env-tool uses the ~/.cde/config.yaml configuration file to manage jobs in Cloudera Data Engineering virtual clusters. You must manually edit the ~/.cde/config.yaml file and update the profiles with the required information.

For more information, see Creating and using multiple profiles .

Prerequisites for setting up the cde-env tool

You must obtain the virtual cluster endpoint URL, Cloudera endpoint URL, and generate user keys for each user whose Spark jobs you are migrating over to Cloudera Data Engineering.

Procedure

1. **Virtual Cluster Endpoint URL**: Determine the virtual cluster endpoint URL.
   a. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
   b. Click Administration in the left navigation menu, select the environment containing the virtual cluster you want to use.
   c. In the Virtual Clusters column on the right, click the Cluster Details icon for the virtual cluster you want to use to migrate your spark jobs to.
   d. Click JOBS API URL to copy the URL to your clipboard.
   e. Paste the URL into a text editor to identify the endpoint host. For example, the URL is similar to the following:

      ```
      https://dfdj6kgx.cde-2cdxw5x5.apps.ecs-demo.example.com/dex/api/v1
      ```

      In the above example, the endpoint host is

      ```
      dfdj6kgx.cde-2cdxw5x5.apps.ecs-demo.example.com
      ```

2. **Cloudera Endpoint URL**: Copy the Cloudera console URL.

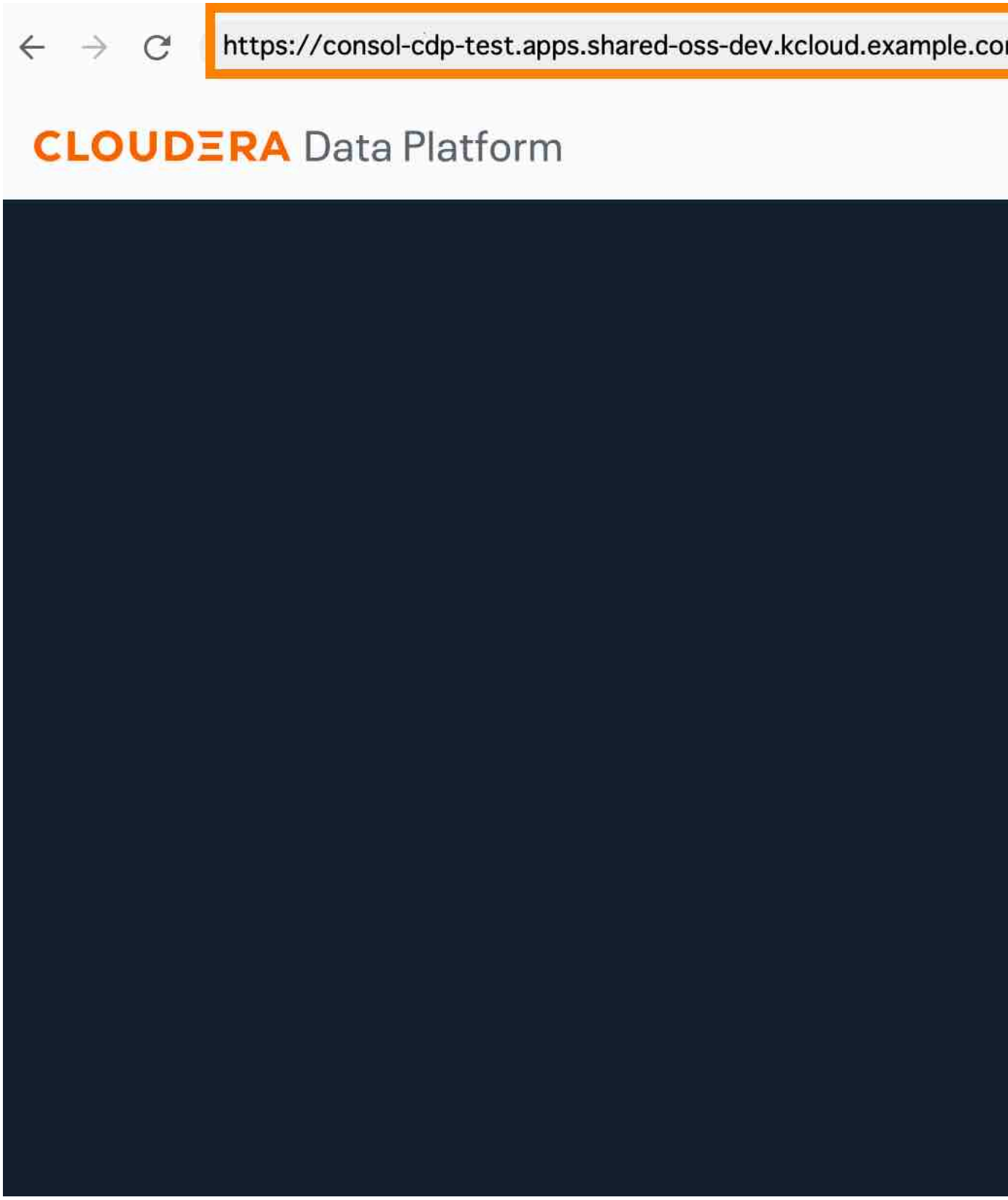

3. **Access Key**: Generate access key for each user whose Spark jobs you are migrating over to Cloudera Data Engineering:
   a. Sign in to the Cloudera console as an Administrator.
   b. In the Cloudera Home page, click Cloudera Management Console.
   c. On the left navigation menu, click Users.
   d. On the Users page, click the name of the user or the machine user account for which you want to generate an access key.
   e. On the user account page, go to the Access Keys section and click Generate Access Key.

      Cloudera creates the key and displays the information on the screen.

Adding profile for each user and creating the Credentials file

You must create a profile for each user in the ~/.cde/config.yaml file and add the access key in the Credentials file.

Procedure

1. Create the config.yaml and credentials files under ~/.cdefolder. If the ~/.cde folder does not exist, create it where the cde-env tool is installed.
2. Edit the config.yaml file. You can create multiple profiles in the ~/.cde/config.yaml file and can be used while running commands.

   Edit the ~/.cde/config.yaml file to add the allow-all-spark-submit-flags: true parameter and update the profiles.

```
# ~/.cde/config.yaml

allow-all-spark-submit-flags: true
credentials-file: <credentials-location>
cdp-endpoint: <CDP-endpoint>
tls-insecure: true

profiles:
- name: <Profile Name1>
  vcluster-endpoint: <VC-endpoint>

- name: <Profile Name2>
  vcluster-endpoint: <VC-endpoint>
```

   Example configuration file:

```
# ~/.cde/config.yaml

allow-all-spark-submit-flags: true
credentials-file: /home/cdpuser1/.cde/credentials
cdp-endpoint: https://console-xhu-141.apps.shared-os-dev-01.kcloud.example
.com
tls-insecure: true

profiles:
- name: vc-2
  vcluster-endpoint: https://5b27g4jm.cde-x6j2nh5j.apps.apps.shared-osdev-
01.kcloud.example.com/dex/api/v1/

- name: spark3-1
  vcluster-endpoint: https://7j92n8q4.cde-smstx27m.apps.apps.shared-osde
v-01.kcloud.example.com/dex/api/v1/
```

3. Add your access key information generated from the Cloudera management console in the credentials file.

   Example credentails file:

```
[default]
```

```
cdp_access_key_id=a4e8f324-5940-454c-a172-5c748e56e4c2
cdp_private_key=qpG0CzVqodKTQYXakm89bjX0606c7fP3EnAcxuy+Rzs=
```

Run the migration tool in a docker container

Mount the config.yaml and credentials files into the docker container and run the docker image in the interactive mode. You have to activate the tool after running the tool before you run `spark-submit` commands.

Procedure

1. Run the docker tool.

```
$ docker run -it \
-v <path-to-yaml-file>/config.yaml:/home/cdpuser1/.cde/config.yaml:ro \
-v <path-to-credential-file>/credentials:/home/cdpuser1/.cde/credentials:
ro \
<customers-docker-registry-for-cdp-private-cloud>/cloudera/dex/dex-migr
ation-tool:<tag>
```

Example:

```
$ docker run -it \
-v /Users/cdp-compute-cluster/cdpuser1/config.yaml:/home/cdpuser1/.cde/co
nfig.yaml:ro \
-v /Users/cdpuser1/credentials:/home/cdpuser1/.cde/credentials:ro \
docker-registry.example.com/cloudera/dex/dex-migration-tool:1.19.1-b185
```

2. Activate the profile.

```
$ cde-env.sh activate -p vc-1
```

Run sample spark-submit command inside the docker container

After you activate the docker image, you can run your `spark-submit` commands on Cloudera Data Engineering without completely rewriting your existing spark-on-yarn command lines inside the docker container.

- Sample spark-submit commands you can run on the Cloudera Data Engineering workloads.

```
$ spark-submit --name pt_rpt_streams5 --master=yarn --deploy-mode=cluste
r --driver-memory 4G --executor-memory 4G --executor-cores 3 --num-execu
tors 4 --conf spark.yarn.queue=hr --conf "spark.executor.extraJavaOption
s=-Djava.security.auth.login.config=/home/hdpsparkprd/spark-hdpsparkprd-
keytab-jaas.conf -Djava.security.krb5.conf=/etc/krb5.conf -Djavax.securi
ty.auth.useSubjectCredsOnly=true" --conf "spark.driver.extraJavaOptions=
-Djava.security.auth.login.config=/home/hdpsparkprd/spark-hdpsparkprd-ke
ytab-jaas.conf -Djava.security.krb5.conf=/etc/krb5.conf -Djavax.security
.auth.useSubjectCredsOnly=true" --conf "spark.io.compression.codec=org.a
pache.spark.io.LZ4CompressionCodec" --class org.apache.spark.examples.Sp
arkPi http://qe-repo.s3.amazonaws.com/dex/app-jar/spark-examples_2.11-2.
4.4.jar 22
```

Known Issues and Limitations

This page lists the current known issues and limitations that you might run into while using the cde-env tool.

- Limited to spark-submit commands and does not include spark-shell, pyspark, and sparksql.
- When activating a profile using the cde-env.sh script, there is no validation yet on whether such profile exists. However, if a profile does not exist, it will display an error when running the spark-submit command.

- The following spark-submit flags are not yet supported in Cloudera Data Engineering:
  - --archives
  - --exclude-packages
  - --driver-class-path
  - --driver-library-path
  - --driver-java-options

  You are instead suggested to create Cloudera Data Engineering jobs to handle the above mentioned scenarios.
- Using the profile yarn is not supported in the container version of the migration tool.

Cloudera Data Engineering Top Tasks

Date published: 2022-09-30
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating Sessions in Cloudera Data Engineering..... 4

Creating jobs in Cloudera Data Engineering..... 5

Creating an Airflow DAG using the Pipeline UI..... 7

Scheduling jobs in Cloudera Data Engineering..... 8

Connecting to Grafana dashboards in Cloudera Data Engineering Public Cloud..... 9

Creating Sessions in Cloudera Data Engineering

A Cloudera Data Engineering Session is an interactive short-lived development environment for running Spark commands to help you iterate upon and build your Spark workloads.

About this task

The commands that are run in a Cloudera Data Engineering Session are called Statements. You can submit the Statements through the connect CLI command or the Interact tab in the Cloudera Data Engineering UI for a Session. Python and Scala are the supported Session types. Learn how to use Cloudera Data Engineering Sessions using the user interface.

Before you begin

Ensure that you are using a version of Cloudera Data Engineering 1.19 or higher for your Virtual Cluster.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Home page displays.
2. Click Sessions in the left navigation menu and then click Create Session.

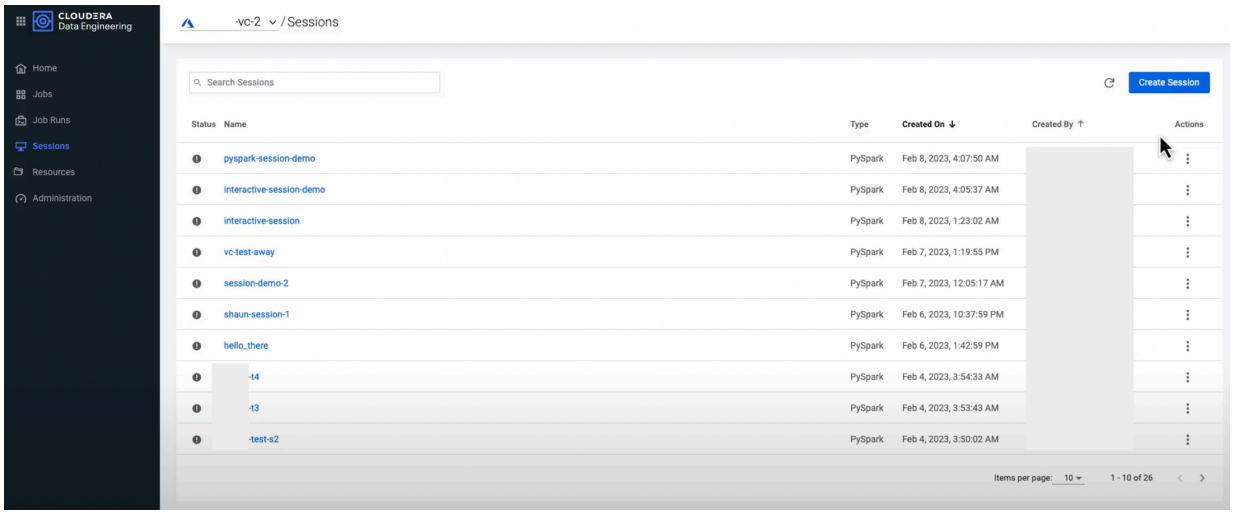

3. Enter a Name for the Session.
4. Select a Type, for example, PySpark, Scala, or Spark Connect.
5. Select a Timeout value.

   The Session will stop after the indicated time has passed.
6. Optionally, enter a Description for the Session.
7. Optionally, enter the Configurations.
8. Set the Compute options.

9. Under **Files and Resources**, you can upload Jar, Python, Egg, Zip, and other files. You can also add a resource, respositories, or a Python environment to be used in this session.

   Files that are uploaded to a session are stored in the app/mount directory.

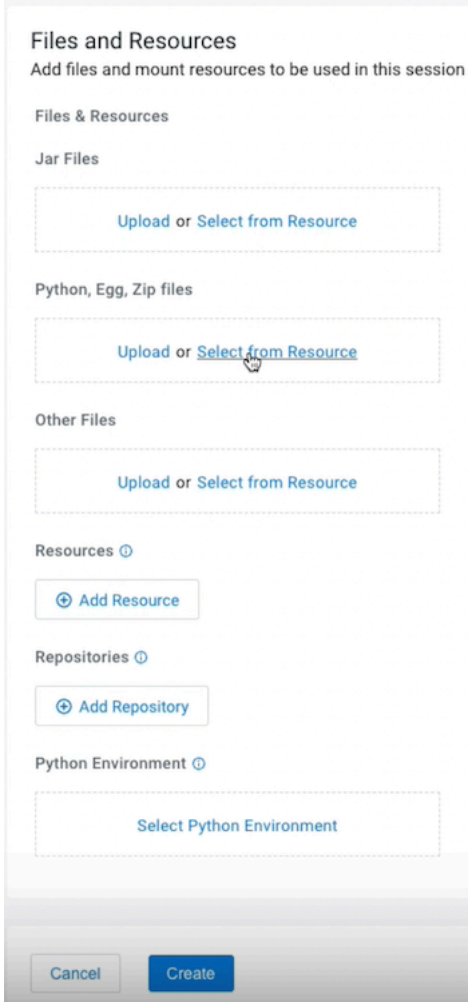

10. Click Create.

    The Connect tab displays a list of connectivity options available to interact with the Session. The Interact tab allows you to interact with the Session, and becomes available once the Session is running.

11. To delete a Session, open the Session and click Delete.

    Note: If you delete a Session, doing so will result in the termination of an active session and the loss of any attached logs and details.

Creating jobs in Cloudera Data Engineering

A job in Cloudera Data Engineering consists of defined configurations and resources (including application code). Jobs can be run on demand or scheduled.

Before you begin

In Cloudera Data Engineering, jobs are associated with virtual clusters. Before you can create a job, you must create a virtual cluster that can run it. For more information, see Creating virtual clusters.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also create a job by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. To view Cloudera Data Engineering Services, click Administration on the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Important: The Cloudera Data Engineering jobs API implicitly adds the default DataLake filesystem to the Spark configuration to save the user having to do that. If you need to reference other buckets, you can set the spark.yarn.access.hadoopFileSystems parameter with the extra comma-separated buckets needed. If you set this parameter in your application code before creating the session, it might override the default setting, leading to errors.

Procedure

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the service that contains the virtual cluster that you want to create a job for.
3. In the Virtual Clusters column on the right, locate the virtual cluster that you want to use and click the View Jobs icon.
4. In the left hand menu, click Jobs.
5. Click the Create Job button.
6. Provide the Job Details:
   a) Select Spark for the job type. For Airflow job types, see Apache Airflow in Cloudera Data Engineering.
   b) Specify the Name.
   c) Select Resources, URL, or Repository for your application file, and provide or specify the file. You can upload a new file or select a file from an existing resource.

      If you select URL and specify an Amazon AWS S3 URL, add the following configuration to the job:

      config_key: spark.hadoop.fs.s3a.delegation.token.binding

      config_value: org.apache.knox.gateway.cloud.idbroker.s3a.IDBDelegationTokenBinding

      If you want to use files from a repository, you must first create a repository. Once the repository is created, you can select Repository, click Add from Repository and select the file. Then, click Select File.
   d) If your application code is a JAR file, specify the Main Class.
   e) Specify arguments if required. You can click the Add Argument button to add multiple command arguments as necessary.
   f) Enter Configurations if needed. You can click the Add Configuration button to add multiple configuration parameters as necessary.

      Important: For Spark jobs, setting the `spark.app.id` property at the Spark job level configuration or within the Spark application code is not supported in Cloudera Data Engineering.

   g) If your application code is a Python file, select the Python Version, and optionally select a Python Environment.
7. Click Advanced Configurations to display more customizations, such as additional files, initial executors, executor range, driver and executor cores and memory.

   By default, the executor range is set to match the range of CPU cores configured for the virtual cluster. This improves resource utilization and efficiency by allowing jobs to scale up to the maximum virtual cluster resources available, without manually tuning and optimizing the number of executors per job.

8. Click Schedule to display scheduling options.

   You can schedule the application to run periodically using the Basic controls or by specifying a Cron Expression.

   Note: Scheduled job runs start at the end of the first full schedule interval after the start date, at the end of the scheduled period. For example, if you schedule a job with a daily interval with a start_date of 14:00, the first scheduled run is triggered at the end of the next day, after 23:59:59. However if the start_date is set to 00:00, it is triggered at the end of the same day, after 23:59:59.

9. If you provided a schedule, click Schedule to create the job. If you did not specify a schedule, and you do not want the job to run immediately, click the drop-down arrow on Create and Run and select Create. Otherwise, click Create and Run to run the job immediately.

10. Optional: Toggle Alerts to send mail to the email address that you choose. You have the option to select Job Failure to send an email upon job failure, and Job SLE Miss to send an email on a Job service-level agreement miss.

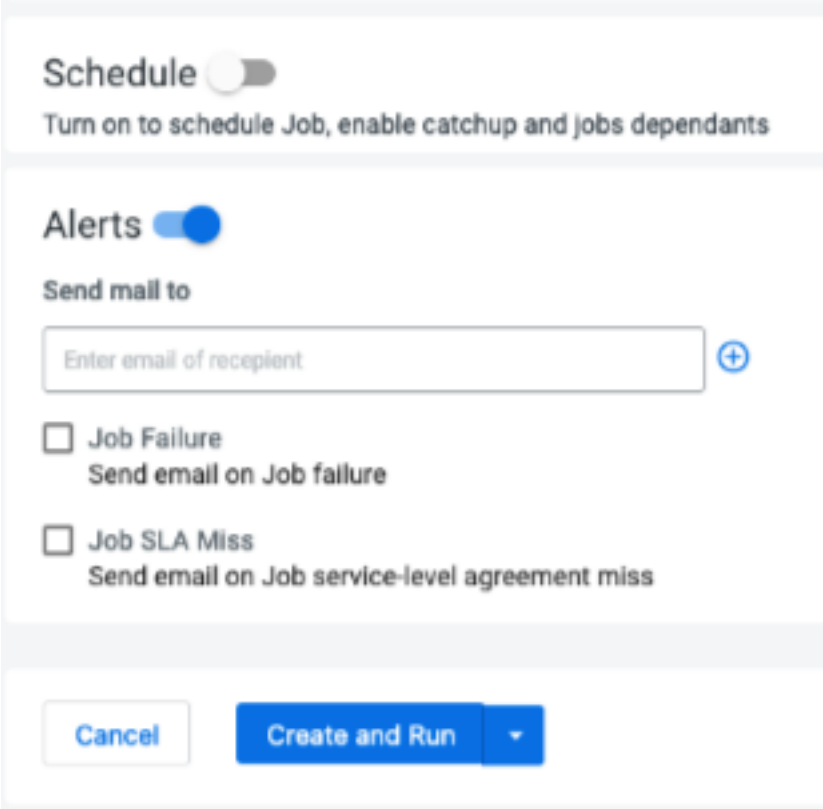

Note: The Alerts option will only display if you have selected Configure Email Alerting during Virtual Cluster creation.

Creating an Airflow DAG using the Pipeline UI

With the Cloudera Data Engineering Pipeline UI, you can create multi-step pipelines with a combination of available operators.

About this task

Note: Cloudera supports all major browsers (Google Chrome, Firefox and Safari) for this feature. If you are using a browser in incognito mode, you have to allow all cookies in your browser settings so that you can view Pipelines, Spark, and Airflow pages.

Procedure

1. Go to Jobs Create Job .

   Under Job details, select Airflow.

   The UI refreshes, only Airflow-specific options remain.
2. Specify a name for the job.
3. Under DAG File select the Editor option.
4. Click Create.
   You are redirected to the job Editor tab.
5. Build your Airflow pipeline.
   - Drag and drop operators to the canvas from the left hand pane.
   - When selecting an operator, you can configure it in the editor pane that opens up.

     On the Configure tab you can provide operator-specific settings. The Advanced tab allows you to make generic settings that are common to all operators, for example execution timeout or retries.
   - Create dependencies between tasks by selecting them and drawing an arrow from one of the four nodes on their edges to another task. If the dependency is valid the task is highlighted in green. If invalid, it is highlighted in red.
   - To modify DAG-level configuration, select Configurations on the upper right.
6. When you are done with building your pipeline, click Save.

Scheduling jobs in Cloudera Data Engineering

Jobs in Cloudera Data Engineering can be run on demand, or scheduled to run on an ongoing basis. The following instructions demonstrate how to create or modify a schedule for an existing job.

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also schedule a job from the new Home page by clicking Schedule a Job, or by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. To view Cloudera Data Engineering Services, click Administration on the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. In the Cloudera console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster where you want to schedule the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to schedule.
4. Click the Configure.
5. Click the Advanced Configurations link at the bottom of the page to view additional configuration parameters.
6. Click the Actions menu next to the application, and then click Configuration.

7. Select the Schedule toggle, and then set the Start time, End time, and Cron expression.

   The start and end times designate the time frame for which the schedule is active. The Cron expression uses the cron scheduling syntax to specify when the application should run within the start and end times. For information and examples of the cron syntax, see the Cron entry on Wikipedia.

   Note: Timestamps must be specified in ISO-8601 UTC format ('yyyy-MM-ddTHH:mm:ssZ'). UTC offsets are not supported.

   Note: Scheduled job runs start at the end of the first full schedule interval after the start date, at the end of the scheduled period. For example, if you schedule a job with a daily interval with a start_date of 14:00, the first scheduled run is triggered at the end of the next day, after 23:59:59. However if the start_date is set to 00:00, it is triggered at the end of the same day, after 23:59:59.
8. If you want to start a job immediately, check the Start job box.
9. Click Update to save your changes.
10. Select optional scheduling configurations:
    a) Select Enable Catchup to kick off job runs for any data interval that has not been run since the last data interval. If this option is not selected, only the runs that start after the time that the job was created will be included.
    b) Select Depends on Previous to ensure that each job run is preceeded by a successful job run.
11. Click Schedule.

Connecting to Grafana dashboards in Cloudera Data Engineering Public Cloud

This topic describes how to access Grafana dashboards for advanced visualization of Virtual Cluster's metrics such as memory and CPU usage in Cloudera Data Engineering Public Cloud.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view Cloudera Data Engineering Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

For Cloudera Data Engineering Service

1. In the Cloudera management console, click the Data Engineering tile and click Overview.

2. In the CDE Services column, click the Service Details button on the environment for which you want to see the Grafana dashboard.

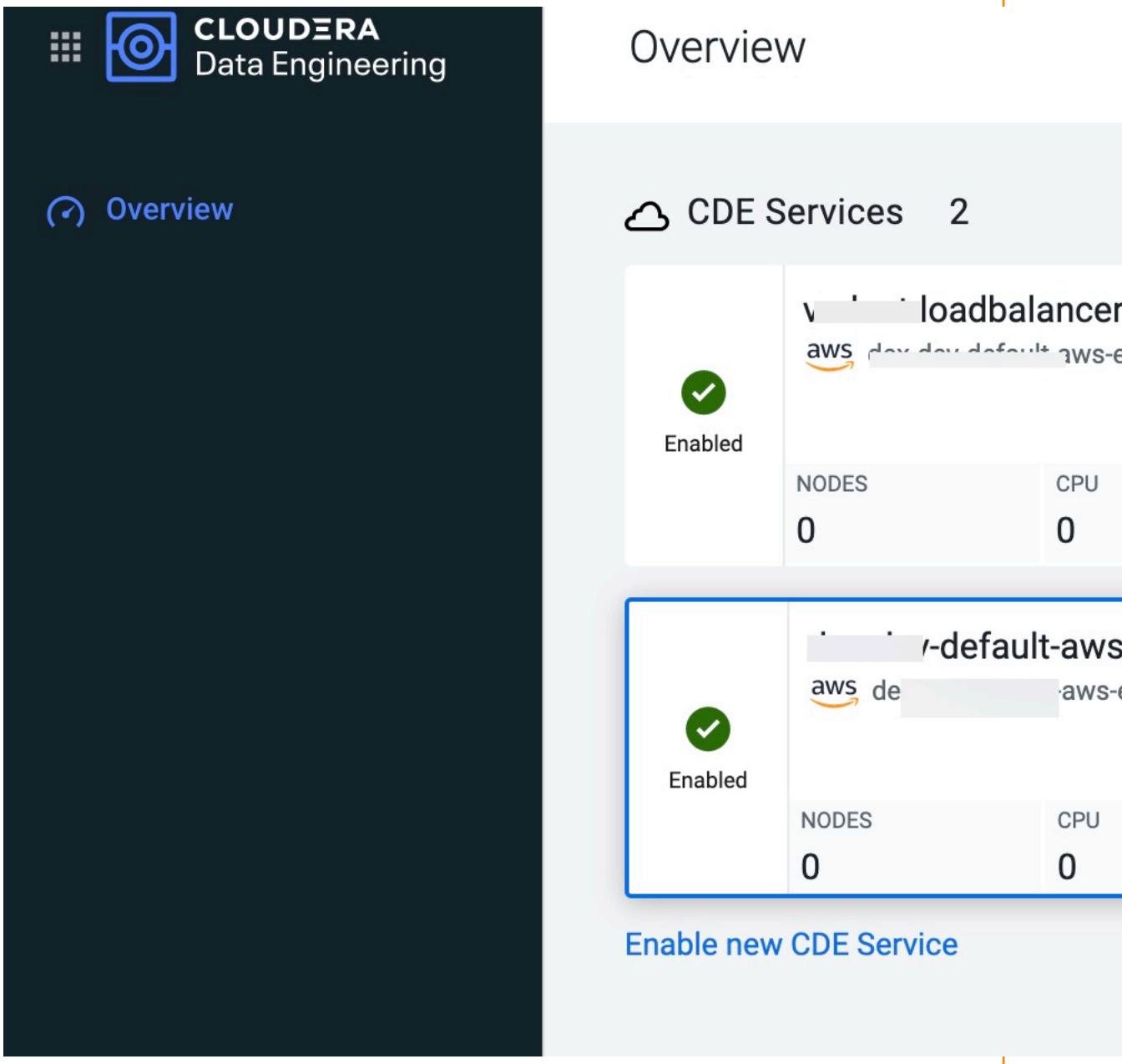

3. In the Service details page, click Grafana Charts in the hamburger menu. A read-only version of the Grafana interface opens in a new tab in your browser.
4. Select Virtual Cluster Metrics under the Dashboards pane.

5. Click on a virtual cluster name from the dropdown list to view the Grafana charts.

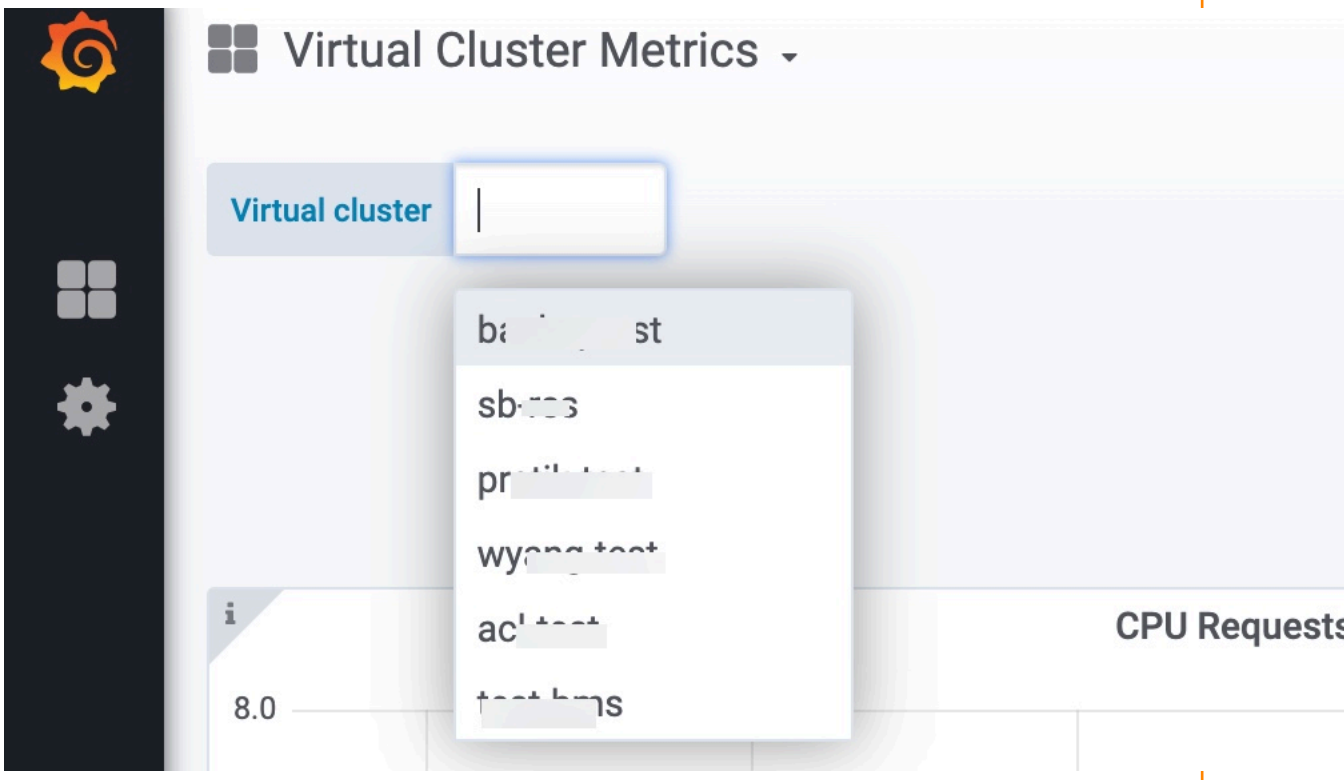

about CPU requests, memory requests, jobs, and other information related to the virtual cluster is displayed.

For Virtual Cluster

1. Navigate to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera management console.
2. In the Service details column, select the environment containing the virtual cluster for which you want to see the Grafana dashboard.
3. In the Virtual Clusters column on the right, click the Cluster Details icon of the virtual cluster.

   The virtual cluster's Overview page is displayed.

4. In the Overview page, click Grafana Charts.

   A read-only version of the Grafana interface opens in a new tab in your browser.

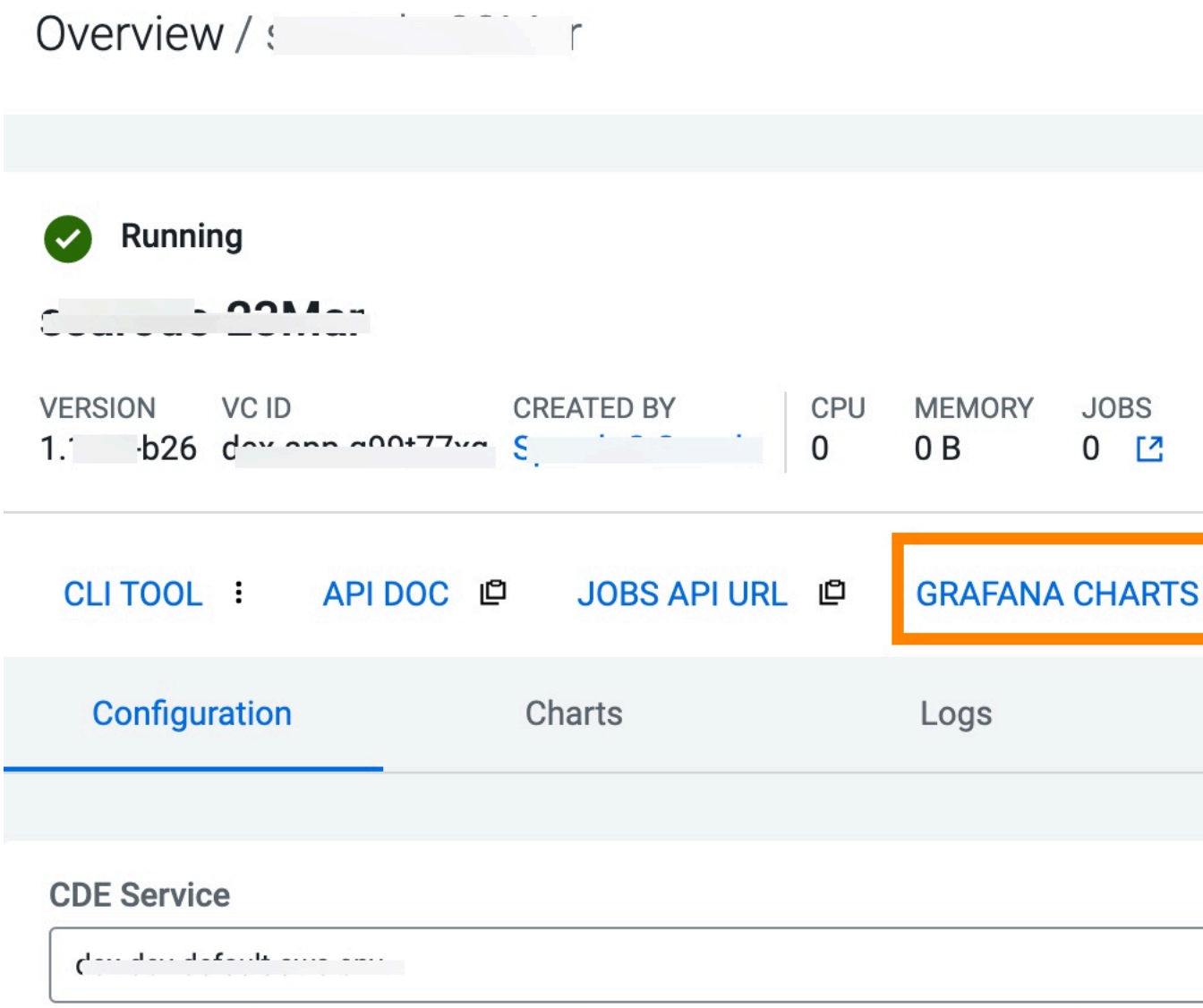

   Information about CPU requests, memory requests, jobs, and other information related to the virtual cluster is displayed.

5. In the Virtual Cluster Metrics page, click on a virtual cluster name from the Virtual Cluster dropdown list to view the Grafana charts of that virtual cluster.

Cloudera Data Engineering 1.23.0

Troubleshooting Cloudera Data Engineering

Date published: 2020-07-30
Date modified: 2024-11-12

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Data Engineering log files....4

Checking the node count on your Cloud Service Provider's website.... 4

Viewing Job run timeline....5

Accessing the Kubernetes dashboard....8

Connecting to Grafana dashboards in Cloudera Data Engineering Public Cloud.... 8

Cloudera Data Engineering diagnostic bundles and summary status....11

Downloading a diagnostic bundle for Cloudera Data Engineering.... 12

Downloading summary status for Cloudera Data Engineering.... 14

Preflight Checks....16

Cloudera Data Engineering CLI exit codes....26

Cloudera Data Engineering Deep Analysis....28

Enabling deep analysis on a Cloudera Data Engineering job from the Cloudera Data Engineering web UI.... 28

Enabling deep analysis on a Cloudera Data Engineering job run using the CLI.... 29

Running deep analysis on a Cloudera Data Engineering job run....29

Cloudera Data Engineering log files

You can view logs for Cloudera Data Engineering using the web console, including Cloudera Data Engineering service logs, virtual cluster logs, and job logs.

To view logs for a Cloudera Data Engineering service or virtual cluster, click the three-dot menu for the service or virtual cluster, and then select View Logs. When the View Logs modal is displayed, you can download the logs or copy them to the clipboard by clicking the associated icon at the top right of the modal.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also manage a job by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

To view logs for a job run:

1. In the Cloudera console, click the Data Engineering tile and click Overview.
2. From the Cloudera Data Engineering Overview page, select the Cloudera Data Engineering service for the job you want to troubleshoot.
3. In the Virtual Clusters column, click the View Jobs icon for the cluster containing the job.
4. Select the job you want to troubleshoot.
5. In the Run History tab, click the Job ID for the job run you want to troubleshoot.
6. Go to the Logs tab.
7. Select the log you want to view using the Select log type drop-down menu and the log file tabs.
8. To download the logs, click the Download menu button. You can download a text file of the currently displayed log, or download a zip file containing all log files.

Checking the node count on your Cloud Service Provider's website

Learn about how to get real-time node count information on Amazon Web Services (AWS) and on Microsoft Azure.

Before you begin

To filter the nodes in the AWS Auto scaling groups (ASGs) or in Azure, obtain one of these identifiers:

- Provisioner ID
- Cluster name
- Cluster ID

Note: To filter using the Cluster name or Cluster ID:

- On AWS: you can filter the ASGs directly using the Cluster name or Cluster ID.
- On Azure, you must use the Add filter option on the Virtual machine scale sets page to query by the Cluster name or Cluster ID.

To obtain the provisioner ID (for example: *LIFTIE-XYZ*), in Cloudera Data Engineering, navigate to:

Administration Service-Details Logs and search for *PROVISIONERID: LIFTIE*.

Procedure

Navigate to your Cloud Service Provider's website.

For AWS

a. In the AWS Management Console, log in as an IAM user, root user, or a similar role.

b. At the top-right corner, select the region in which your cluster is located.

> **Note:** Ensure that you select the correct region, because ASGs are region-specific.

c. In the AWS Management Console Search bar, type EC2, and from the displayed list, select EC2.

The EC2 Dashboard for the selected region is displayed.

d. In the EC2 Dashboard, in the left navigation pane, click Auto Scaling and select Auto Scaling Groups.

On the Auto Scaling Groups page, you can see a list of all ASGs in the selected region.

e. In the search bar, filter the ASGs using one of the following:

- Cluster name
- Cluster ID
- provisioner ID

f. To calculate the number of nodes in your Virtual Cluster (VC), add all the values displayed in the Instances column.

> **Note:** The list includes infrastructure nodes also.

g. Optional: If your cluster operates in multiple regions, switch to another region at the top-right corner of the AWS Management Console and repeat the steps described in this procedure.

For Azure

a. Login to Azure Portal.

b. In the top-right corner, select the Directory and the Subscription where your VC is hosted.

> **Note:** Even though regions are not selected at this step, make sure that you know in which region your VC operates.

c. At the top of the portal, in the Search bar, type Virtual machine scale sets and select it from the displayed list.

The Virtual machine scale sets page is displayed, where all Virtual Machine Scale Sets (VMSS) in your subscription are listed.

d. To narrow down the list, perform one of these options:

- If your cluster resources are grouped within a specific Resource group, use the Resource group filter to narrow down the list.
- To search by the Cluster Name or Cluster ID tags associated with your Cluster or node pools, use the Add filter option. If you add a tag filter, the list of VMSS associated with the queried Cluster ID or Cluster Name is displayed.
- To filter by the Provisioner ID, enter the Provisioner ID in the Search field on the VMSS page.

> **Note:** The list includes infrastructure nodes also.

e. To calculate the number of nodes in your VC, add up all the values displayed in the Instances column.

f. Optional: If your VC spans through multiple regions, make sure that you select the correct Resource group, or use the global search to find resources in the specific region.

Viewing Job run timeline

You can view the intermediate stages of the job run at every stage during its life cycle in real-time.

About this task

In case of a job failure, you can view the specific event and component where the job run failed. This reduces turnaround time during the debugging process for job run failure. You can see the step-by-step advancement of the job run on the UI, including all the granular details instead of reviewing extensive logs to obtain the same insights.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. In the left navigation pane, click Jobs Runs. The Jobs Runs page displays.
3. Click on the Job Id for which you want to see the status.

4. Go to the Timeline tab. It displays the summary of the Job run progression in the reverse chronological order for both primary and subordinate stages.

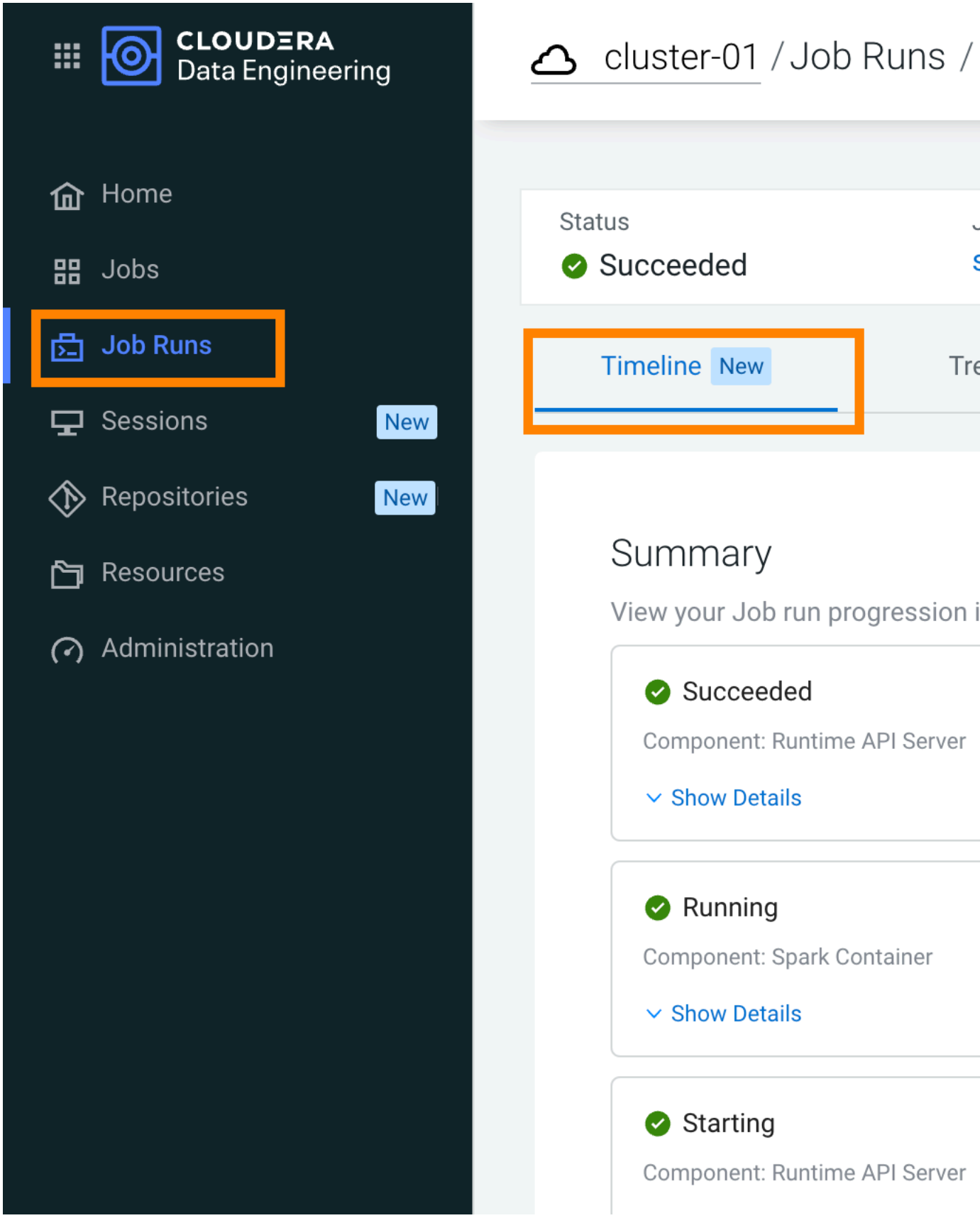

Accessing the Kubernetes dashboard

As a user with the DEAdmin role, you can access the Kubernetes dashboard in Cloudera Data Engineering for troubleshooting and debugging of the cluster deployment.

About this task

The Kubernetes dashboard is to be used for troubleshooting in coordination with Cloudera support. The Kubernetes dashboard allows you to view and download diagnostics and container logs. Additionally, the DEAdmin can initiate limited actions such as restarting pods, modifying configuration maps and deployments which is beneficial for a debugging session.

Note: If you prefer to to use the kubectl tool for troubleshooting instead, see Enabling kubectl for Cloudera Data Engineering linked below. The kubectl tool is also to be used in coordination with Cloudera support.

Before you begin

Ensure that you have the DEAdmin role.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view Cloudera Data Engineering Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

Procedure

1. In the Cloudera console, click the Data Engineering tile and click Overview.
2. In the CDE Services column, click Service Details.
3. Copy the Resource Scheduler URL and open a new browser window.
4. Paste the Resource Scheduler URL in your new browser window and edit the URL as follows:
   a) In the URL, replace the word "YuniKorn" with "dashboard". For example, https://yunikorn. becomes https://dashboard..
   b) Press Enter

Results

The Kubernetes Dashboard displays and provides an easy user experience for monitoring your diagnostics.

Related Information

Deploy and Access the Kubernetes Dashboard
Enabling kubectl for Cloudera Data Engineering

Connecting to Grafana dashboards in Cloudera Data Engineering Public Cloud

This topic describes how to access Grafana dashboards for advanced visualization of Virtual Cluster's metrics such as memory and CPU usage in Cloudera Data Engineering Public Cloud.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view Cloudera Data Engineering Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

For Cloudera Data Engineering Service

1. In the Cloudera management console, click the Data Engineering tile and click Overview.
2. In the CDE Services column, click the Service Details button on the environment for which you want to see the Grafana dashboard.

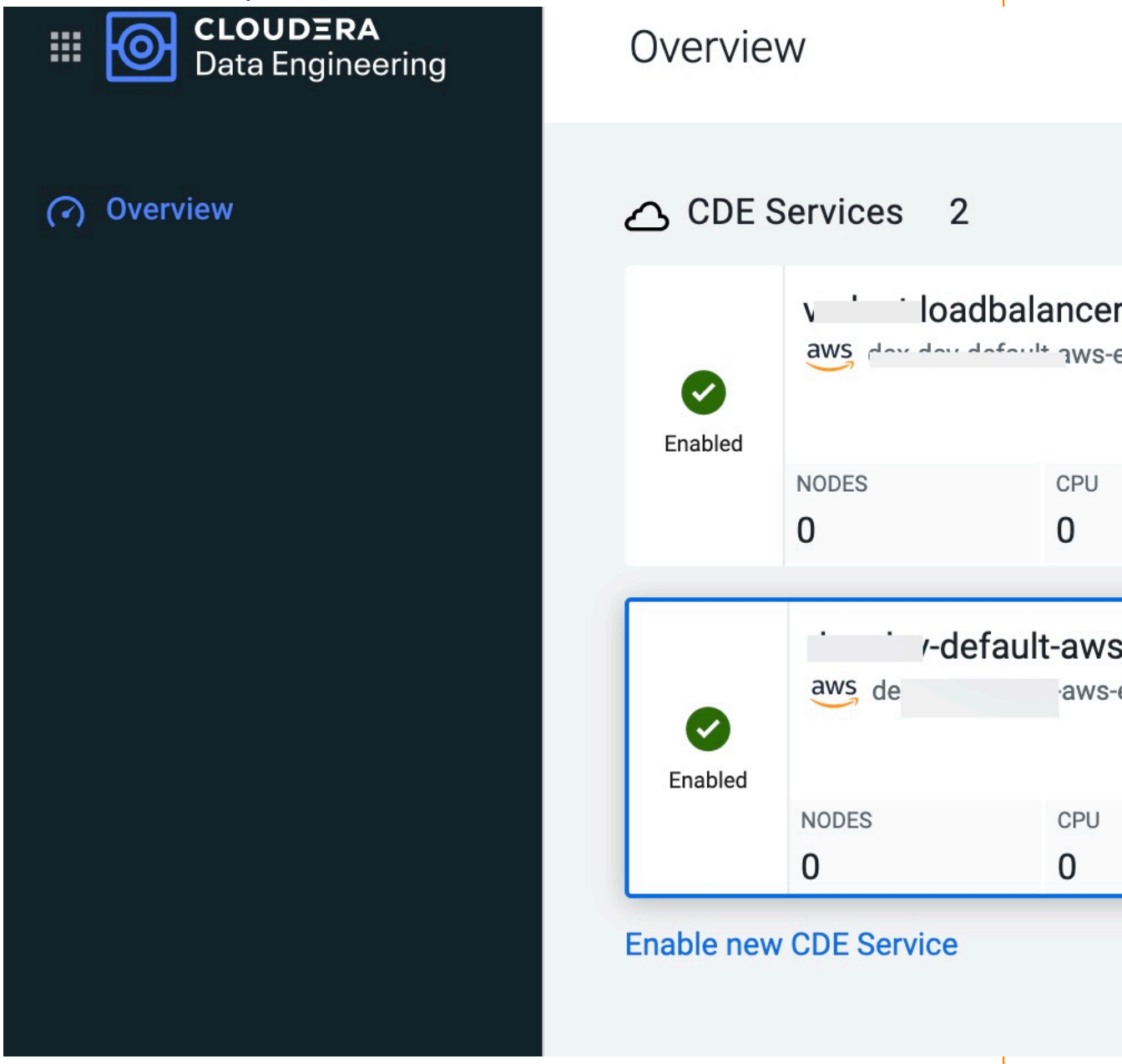

3. In the Service details page, click Grafana Charts in the hamburger menu. A read-only version of the Grafana interface opens in a new tab in your browser.
4. Select Virtual Cluster Metrics under the Dashboards pane.

5. Click on a virtual cluster name from the dropdown list to view the Grafana charts.

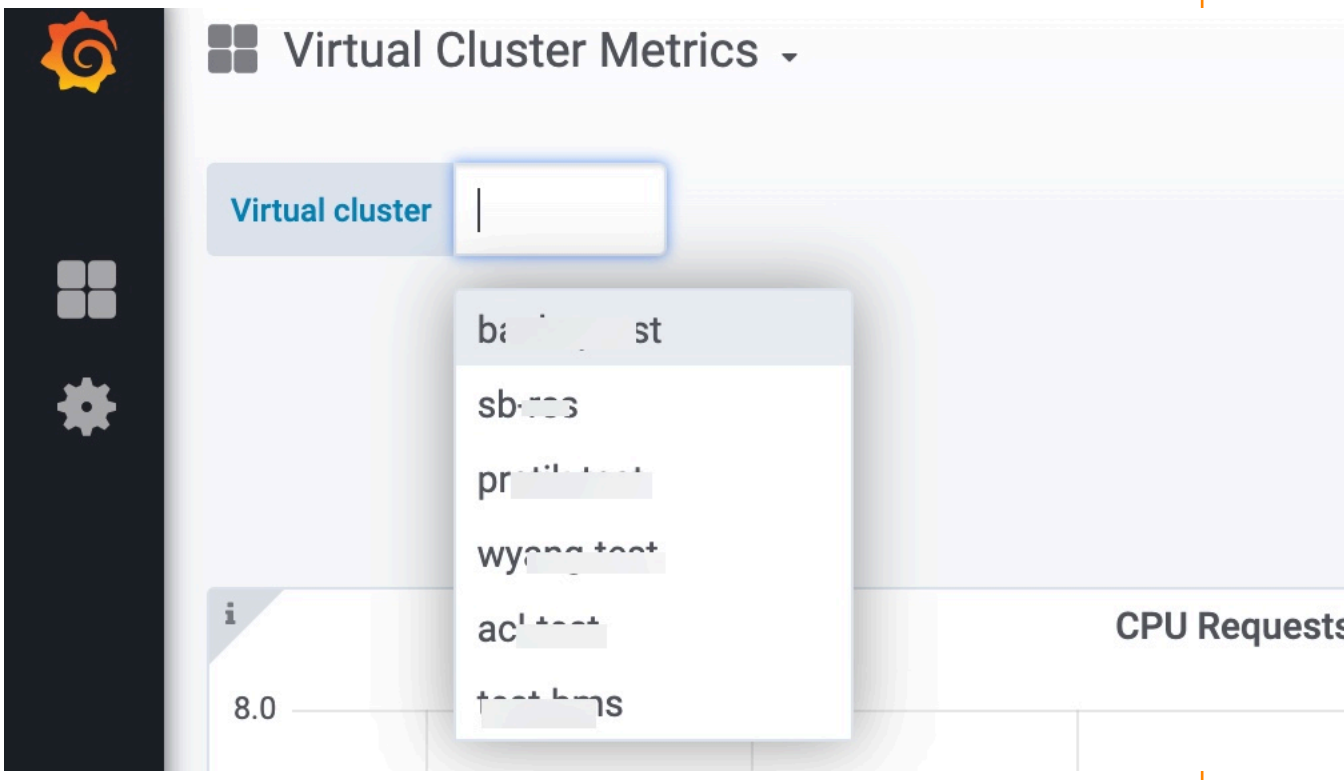

about CPU requests, memory requests, jobs, and other information related to the virtual cluster is displayed.

For Virtual Cluster

1. Navigate to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera management console.
2. In the Service details column, select the environment containing the virtual cluster for which you want to see the Grafana dashboard.
3. In the Virtual Clusters column on the right, click the Cluster Details icon of the virtual cluster.

   The virtual cluster's Overview page is displayed.

4. In the Overview page, click Grafana Charts.

   A read-only version of the Grafana interface opens in a new tab in your browser.

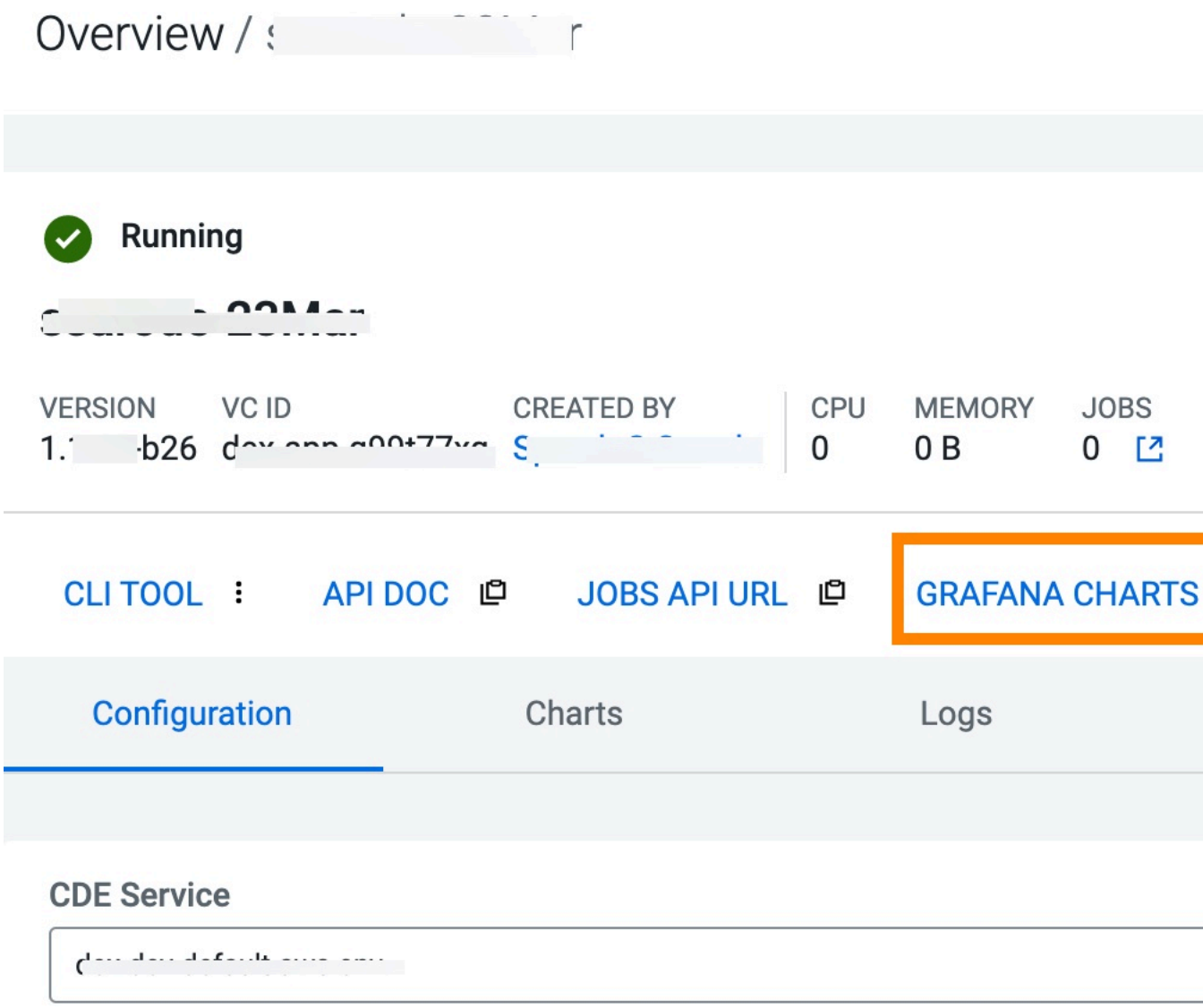

   Information about CPU requests, memory requests, jobs, and other information related to the virtual cluster is displayed.

5. In the Virtual Cluster Metrics page, click on a virtual cluster name from the Virtual Cluster dropdown list to view the Grafana charts of that virtual cluster.

Cloudera Data Engineering diagnostic bundles and summary status

Cloudera Data Engineering provides the ability to download log files, diagnostic data, and a summary status for the running Cloudera Data Engineering services and virtual clusters. You can provide this data to Cloudera Support for assistance troubleshooting an issue.

The following section describes about Cloudera Data Engineering Diagnostic Bundles and Summary Status, and the information collected in each.

Note: A single click now generates one unified bundle containing both service logs and summary status.

Diagnostic Bundle

The diagnostic bundle is a collection of the logs from the Cloudera Data Engineering Services. These logs can be downloaded on-demand from Cloudera Data Engineering UI. Cloudera Data Engineering gives you the functionality to select the sources for which you want to download the logs and you can also select a predefined or custom time range for these logs.

Information Collected in Diagnostic Bundles

- Container logs for all running Cloudera Data Engineering service pods (excluding user compute pods).

Summary Status

The Status Summary shows the status of each service instance being managed by the Cloudera Data Engineering Control Plane. It is a package of JSON files consisting of information related to configuration, monitoring, logging, events and health test results of the service and its instances.

Information Collected in Summary Status

- Status of all cloud resources created during Cloudera Data Engineering provisioning
- Kubernetes resource status for critical infrastructure pods, deployments, pods, services and events for core service infrastructure pods or virtual cluster infrastructure pods

Downloading a diagnostic bundle for Cloudera Data Engineering

This section describes how to download a diagnostic bundle to troubleshoot a Cloudera Data Engineering Service in Cloudera using Cloudera Data Engineering UI.

About this task

To troubleshoot issues with your Cloudera Data Engineering Service, download diagnostic bundles of log files. These diagnostic bundles, in the form of ZIP files, are downloaded to your local machine. This is available on both AWS and Azure environments.

Before you begin

Required Role: DEAdmin

After granting or revoking the role on the environment, run the Sync Users to FreeIPA environment action.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Engineering service
2. In the Cloudera Data Engineering service, click Overview in the left navigation panel, and select the Cloudera Data Engineering Service for which you want to download a diagnostic bundle.

3. In the selected Cloudera Data Engineering service, click on the three-dot menu, and select the Download Diagnostics option.

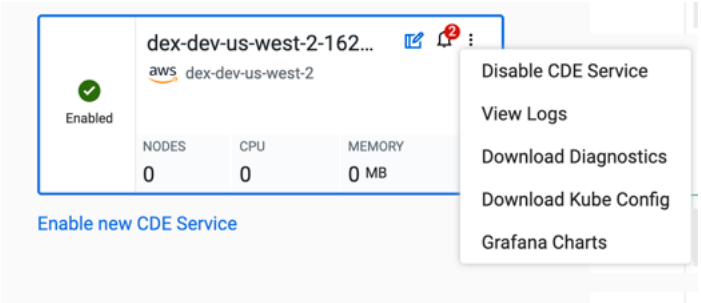

4. On the Diagnostics tab of the Service details, click the Generate Diagnostics Bundle button. The Diagnostic Bundle Options dialog box launches.
5. Set the time range and select the log sources that you want to include:

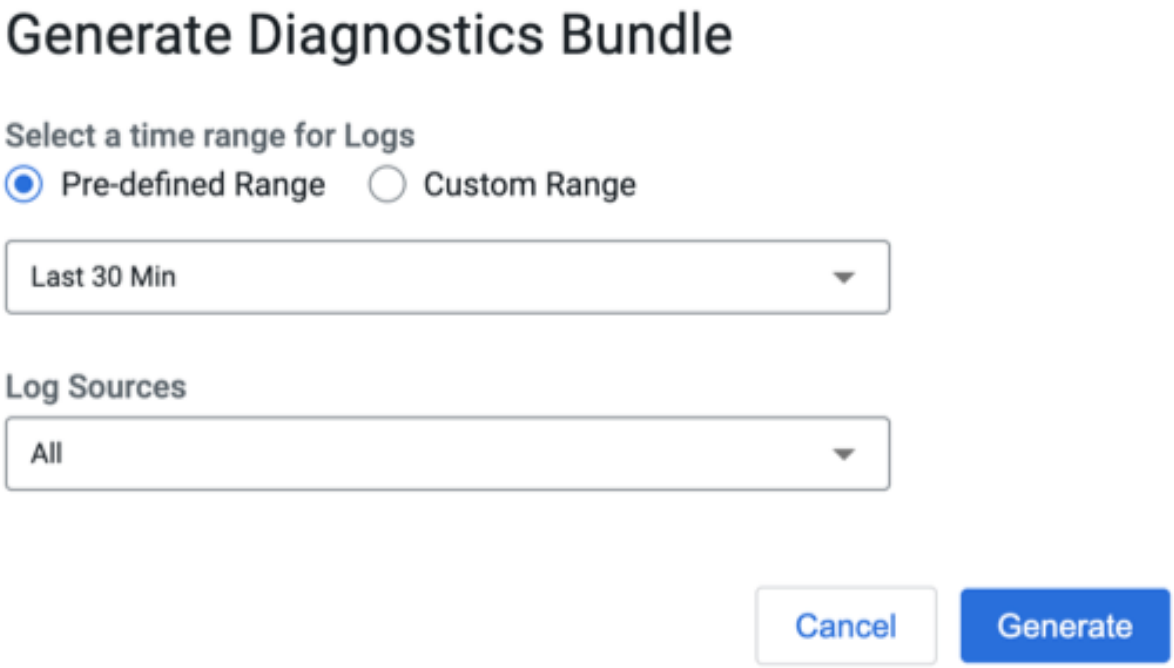

In the dialog box, you can choose or set the following options:

- Pre-defined Range: Select a specific time range of log files to generate from the drop-down list, or you can choose a custom interval in the next option.
- Custom Range: Select the start and end time from the drop-down list to define the specific time interval for the log files in the diagnostic bundle.
- Log Sources: Select All or any of the available categories of log files for the different Services and virtual clusters by clicking the adjacent checkbox.

6. After selecting which log sources you want to include in the diagnostic bundle, click Generate to generate the bundle.
7. When the diagnostic bundle is created, click Download.

Results

When you extract the diagnostic bundle .gz file that you downloaded from Cloudera Data Engineering UI, you find the directories that contain service log files and summary status.

Downloading summary status for Cloudera Data Engineering

This section describes how to download a summary status of a Cloudera Data Engineering Service in Cloudera using Cloudera Data Engineering UI.

About this task

To get a current snapshot of the current Cloudera Data Engineering infrastructure status (cloud resources, Kubernetes pod statuses, pod events and service metadata). This summary status, in the form of a ZIP file, is downloaded to your computer. This is available on both AWS and Azure environments.

Before you begin

Required Role: DEAdmin

After granting or revoking the role on the environment, run the Sync Users to FreeIPA environment action.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu and locate the service in the Services column, and click Service Details on the environment for which you want to download a summary status.
3. In the selected Cloudera Data Engineering service, click ⋮ to see a drop down menu, and select Download Diagnostics.

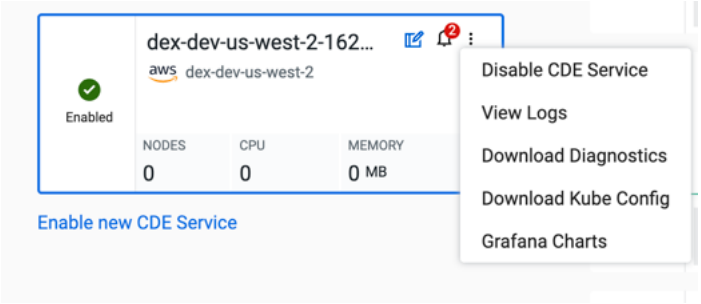

4. It takes you to the Diagnostics tab of the Service details. Click the Download Summary Status button.

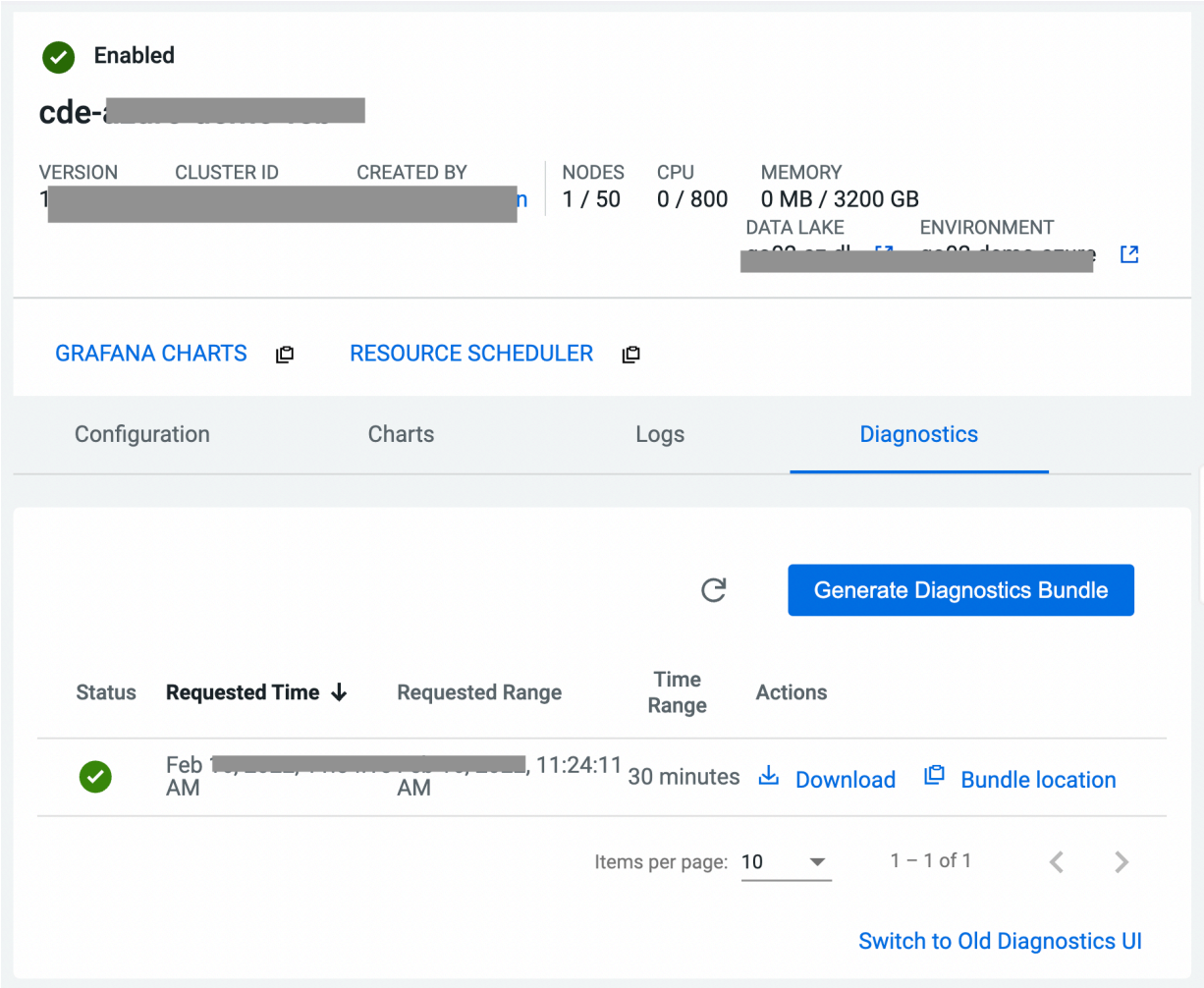

5. Summary Status gets downloaded on your local machine in a Zip file.

Results

When you extract the Summary Status ZIP file that you downloaded from Cloudera Data Engineering UI, you can see the directories that contain files in JSON format.

Below is an example of format of the Zip file for Summary Status:

Note: In the extracted Summary Status file, cluster.json refers to the Cloudera Data Engineering service and 'instances' refer to Cloudera Data Engineering virtual clusters.

```
cde-service-diagnostics-{clusterID}-{timestamp}/
  cluster.json            # output from cluster describe
  cluster-events.json     # output from get cluster events
  cloud-diagnostics.json  # cloud resource status (RDS, EFS, EKS, LB, ...)
  instances/
    {id}.json             # output from instance id describe
    {id}-events.json      # output from get instance id events
  kubernetes/             # kubernetes resource status for all relevant obj
ects and namespaces
    nodes.json
    {namespace}/
      namespace.json
      pods.json
```

```
      events.json
      deployments.json
      replicasets.json
      services.json
      daemonsets.json
    {namespace}/
      namespace.json
      pods.json
      events.json
      deployments.json
      replicasets.json
      services.json
      daemonsets.json
    ...
```

Preflight Checks

Part of provisioning and managing a Kubernetes cluster in the cloud is ensuring that your account and your environment is properly configured before beginning. Cloudera automatically runs a series of preflight checks which can help in determining if there is a problem before creating new resources or adjusting existing ones.

When creating a brand new cluster, preflight validation checks are used to ensure that the resources you are requesting as well as your environment are ready and configured correctly. For existing clusters, many of the infrastructure validations are skipped since they are not needed anymore. Instead, the validations which concern the resources being adjusted are executed.

Results

The result of running a collection of preflight checks is represented as a single aggregated value that lets you know whether it is safe to proceed with your actions. Each individual validation which was run is included in the response.

A preflight validation contains information concerning what it was checking for and what the result of that check was. For example, this is a very simple check to ensure that the type of node pool image is valid in the region:

Name	**Instance type**
Description	Instance groups must have an instance type that exists in the region in which they will be created. For EKS, there is additional verification for EKS support and usage class.
Category	COMMON
Status	FAILED
Message	The instance type validation failed.
Detailed Message	Instance type validation failed for Standard_B2s1 in west us2. Check to ensure that this instance type is valid in that region.
Duration	659 ms

PASSED

The validation successfully passed all criteria.

WARNING

The validation was either unable to fully check all of its criteria or it found a potential issue which could affect the success of the operation. This type of failure will not stop a cluster from being created or modified, but it does require further investigation.

SKIPPED

The validation was skipped because it does not apply to the current request. This can happen for many reasons, such as using a cloud provider that is not applicable to the preflight check.

FAILED

The validation failed its expected criteria and provisioning or updating of a cluster cannot proceed. In this case, there's an identified problem that needs resolution before continuing.

List of Preflight Checks

Node / Agent Pools

The following preflight validation checks pertain to the node / agent instance groups which are created to launch new nodes inside of the Kubernetes cluster.

Name	Description	Cloud	Type
Instance Count	The number of nodes in each instance group must not exceed a predefined threshold. Remediation: Change the requested size of an instance group to be within the boundaries specified by the error message.	All	Create Update
Group Count	The number of distinct node/agent pool groups must not exceed a predefined threshold. Remediation: Change the number of distinct instance groups to be within the boundaries specified by the error message.	All	Create Update
Instance Naming	The name of each instance group is restricted by cloud providers. There are differences in the size and characters allowed by each provider. Remediation: Change the name of the instance group to conform to the cloud provider's requirements. This could include changing the overall length of the name	All	Create Update

Name	Description	Cloud	Type
	or only using certain approved characters.		
Instance Type	Instance image types are not universal across all regions. Some providers further restrict this to service level, and whether they can be used for Kubernetes. Remediation: Choose a different region or service type for the desired instance type. If this is not possible, then a different instance type must be chosen.	All	Create Update
Kubernetes Version	Each cloud provider supports different versions of Kubernetes. CDP has also only been certified to work with particular versions. Remediation: Choose a different version of Kubernetes that is supported by your cloud provider in the region you are deploying in. https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-versions.html	All	Create
Placement Rules	Some instance types are not allowed to be grouped within a single availability zone. Remediation: Remove the restriction on single availability zone, or choose a different instance type.	AWS	Create Update

Infrastructure

The following preflight validation checks pertain to Cloudera Control Plane infrastructure and your specific account within that control plane.

Name	Description	Cloud	Type
Restricted IAM Policies	If your account is trying to provision with restricted IAM policies, then it needs to have those policies defined before deploying the cluster. Remediation: Check Cloudera's documentation on restricted IAM policies to ensure that you have the correctly named policies defined and are accessible.	AWS	Create
Proxy Connectivity	When provisioning a private cluster, your environment must have the cluster proxy enabled and it must be healthy. Remediation: Create a new environment which has the cluster proxy service enabled or check that your existing FreeIPA Virtual Machine is running and healthy. https://docs.cloudera.com/management-console/test/connection-to-private-subnets/topics/mc-ccm-overview.html	All	Create Update
Data Lake Connectivity	A healthy data lake with a functioning FreeIPA DNS server is required in order to provision a new cluster. Remediation: Check to ensure that the FreeIPA DNS server is running and healthy inside of your network. https://docs.cloudera.com/management-console/cloud/data-lakes/topics/mc-data-lake.html https://docs.cloudera.com/management-console/cloud/identity-management/topics/mc-identity-management.html	All	Create

Networking

The following preflight validation checks pertain to specific network configurations inside of the cloud provider.

Name	Description	Cloud	Type
Shared VPC	When a Virtual Private Cloud is shared between multiple subscriptions, access to modify this VPC needs to be granted. Remediation: Check the permission for which roles can make modifications to the VPC https://docs.aws.amazon.com/vpc/latest/userguide/vpc-sharing.html https://docs.cloudera.com/cdp-public-cloud/cloud/requirements-aws/topics/mc-aws-req-vpc.html	AWS	Create
Subnet Availability Zone	All subnets which are part of the environment must be located in at least 2 different Availability Zones. Remediation: Recreate your environment and choose subnets that satisfy the requirement of being in at least 2 different Availability Zones.	AWS	Create
Subnet Load Balancer Tagging	In order for load balancers to choose subnets correctly, a subnet needs to have either the public or private ELB tags defined. Remediation: Tag subnets with either kubernetes.io/role/elb or kubernetes.io/role/internal-elb based on whether they are public or private. https://docs.cloudera.com/cdp-public-cloud/cloud/requirements-aws/topics/mc-aws-req-vpc.html	AWS	Create

Name	Description	Cloud	Type
API Server Access	Validates that there are no conflicting requests between Kubernetes API CIDR ranges and private AKS clusters. Remediation: When using a private AKS cluster, Kubernetes API CIDR ranges are not supported,	All	Create Update
Available Subnets	Subnets cannot be shared when provisioning Kubernetes clusters on Azure. At least one available subnet must exist that is not being used by another AKS cluster and must not have an existing route table with conflicting pod CIDRs. Remediation: Create a new subnet to satisfy this requirement or delete an old and unused cluster to free an existing subnet.	Azure	Create
Delegated Subnet	A subnet which has been delegated for a particular service cannot be used to provision an Azure AKS cluster. Remediation: Choose a different subnet or remove the delegated service from at least one subnet in the environment.	All	Create
Kubernetes API Server Security	Validates that the supplied IP CIDR ranges are valid and do not overlap any reserved IP ranges. Each cloud provider has a limit set on the maximum number of allowed CIDRs. Remediation: Use valid CIDR formats and ranges when limiting access to the Kubernetes API server and limit the number of ranges specified.	All	Create

Name	Description	Cloud	Type
Kubernetes Service CIDR Validation	Validates that the specified service CIDR for Kubernetes services does not overlap any restricted CIDR ranges and is a valid CIDR format. Remediation: Change the service CIDR so that it doesn't conflict with any pod CIDRs or other routes on the subnet.	All	Create
Autoscale Parameters	Azure's built-in autoscaler has limitations on the ranges of values for scale-up and scale-down operations. Remediation: Adjust the specified parameters from the error message which are not within the required ranges.	Azure	Create Update

Example


```
{
   "result": "PASSED",
   "summary": {
       "passed": 8,
       "warning": 0,
       "failed": 0,
       "skipped": 10,
       "total": 18
   },
   "message": "The cluster validation has passed, but some checks were skip
ped",
   "validations": [
       {
           "name": "Instance Count",
           "description": "Each instance count must be between minInstance a
nd maxInstance inclusively. The minInstance and maxInstance of infrastructur
e group should comply with minimum number of infra nodes and maximum number
of infra nodes.",
           "category": "COMMON",
           "status": "PASSED",
           "message": "The minimum and maximum instance counts are correct
 for all instance groups.",
           "detailedMessage": "The minimum and maximum instance counts are c
orrect for all instance groups.",
           "duration": "1µs"
       },
       {
           "name": "Instance Group Count",
           "description": "Total instance group count must be less than or
equal to maximum instance group limit.",
           "category": "COMMON",
```

```
            "status": "PASSED",
            "message": "The number of instance groups in the request is less
 than or equal to the maximum allowed.",
            "detailedMessage": "The total instance group count of 2 is within
 the limit.",
            "duration": "3µs"
        },
        {
            "name": "Instance Group Naming",
            "description": "Each instance group name must conform the rest
rictions of the cloud provider. This includes using valid characters and adh
ering to length restrictions.",
            "category": "COMMON",
            "status": "PASSED",
            "message": "All instance groups meet the naming restrictions for
Azure.",
            "detailedMessage": "All instance groups meet the naming restrict
ions for Azure.",
            "duration": "12µs"
        },
        {
            "name": "Instance Type",
            "description": "Instance groups must have an instance type that
 exists in the region in which they will be created. For EKS, there is addi
tional verifiation for EKS support and usage class.",
            "category": "COMMON",
            "status": "PASSED",
            "message": "All instance groups have valid instance types.",
            "detailedMessage": "The following instance types were validated
 for westus2: Standard_B2s",
            "duration": "599ms"
        },
        {
            "name": "Kubernetes Version",
            "description": "Each cloud provider (Amazon, Azure, Google, etc)
supports different versions of Kubernetes.",
            "category": "COMMON",
            "status": "PASSED",
            "message": "The specified Kubernetes version 1.18 has been resol
ved to 1.18.17 and is valid on Azure",
            "detailedMessage": "The specified Kubernetes version 1.18 has
been resolved to 1.18.17 and is valid on Azure",
            "duration": "388ms"
        },
        {
            "name": "Placement Rule",
            "description": "Instance Types must be allowed by the placement r
ule.",
            "category": "COMMON",
            "status": "SKIPPED",
            "message": "Skipping validation since the cloud platform is Azure
.",
            "detailedMessage": "Skipping validation since the cloud platform
is Azure.",
            "duration": "6µs"
        },
        {
            "name": "Entitlement Check",
            "description": "When the entitlement LIFTIE_USE_PRECREATED_IAM_RE
SOURCES is enabled, the expected profile (cdp-liftie-instance-profile) shoul
d exist and it needs to have the necessary roles attached to it. ",
            "category": "ENTITLEMENTS",
            "status": "SKIPPED",
```

```
            "message": "IAM Resource Entitlement validation skipped for Cl
oud Provider azure.",
            "detailedMessage": "IAM Resource Entitlement validation skipped
 for Cloud Provider azure.",
            "duration": "14µs"
        },
        {
            "name": "Cluster Proxy Connectivity",
            "description": "Verifies connectivity to the cluster proxy ser
vice which is used to register private cluster endpoints.",
            "category": "CONTROL_PLANE",
            "status": "SKIPPED",
            "message": "Connectivity to the cluster connectivity manager will
 be skipped since this is not a private cluster.",
            "detailedMessage": "The cluster being provisioned is not marked
 as private in the provisioning request.",
            "duration": "1µs"
        },
        {
            "name": "Cluster Proxy Enabled",
            "description": "Verifies that the environment was created with th
e cluster proxy service enabled.",
            "category": "CONTROL_PLANE",
            "status": "SKIPPED",
            "message": "Skipping the environment check for cluster proxy c
onnectivity since the cluster is public.",
            "detailedMessage": "Skipping the environment check for cluster
 proxy connectivity since the cluster is public.",
            "duration": "1µs"
        },
        {
            "name": "DataLake Connectivity",
            "description": "Validates whether DataLake connection is reach
able and if FreeIPA is available.",
            "category": "CONTROL_PLANE",
            "status": "PASSED",
            "message": "DataLake validation succeeded.",
            "detailedMessage": "Data lake is healthy and reachable. Service
 Discovery Feature is enabled, verified DNS entries retrieved for Data Lake
s. Datalake URL : localhost:8081  Service discovery URL : localhost:8082 "
        },
        {
            "name": "AWS Shared VPC Access",
            "description": "When a shared VPC is used, proper access should
 be granted.",
            "category": "NETWORK",
            "status": "SKIPPED",
            "message": "Skipping validation since the cloud platform is Azur
e.",
            "detailedMessage": "Skipping validation since the cloud platform
 is Azure.",
            "duration": "3µs"
        },
        {
            "name": "AWS Subnet Availability Zones",
            "description": "When existing AWS subnets are provided for provi
sioning an EKS cluster, the subnets must be in at least 2 different Availabi
lity Zones.",
            "category": "NETWORK",
            "status": "SKIPPED",
            "message": "Skipping validation since the cloud platform is Azure
.",
            "detailedMessage": "Skipping validation since the cloud platform
is Azure.",
```

```
            "duration": "2µs"
        },
        {
            "name": "AWS Subnet Tagging",
            "description": "In order for load balancers to choose subnets co
rrectly a subnet needs to have either the public or private ELB tags defined
.",
            "category": "NETWORK",
            "status": "SKIPPED",
            "message": "Skipping validation since the cloud platform is Azu
re.",
            "detailedMessage": "Skipping validation since the cloud platform
 is Azure.",
            "duration": "26µs"
        },
        {
            "name": "Azure API Access Parameters",
            "description": "Verifies that the security parameters for locking
 down access to the Azure Kubernetes API Server are correct.",
            "category": "NETWORK",
            "status": "PASSED",
            "message": "The API server access parameters specified in the cl
uster request are valid.",
            "detailedMessage": "The cluster will be provisioned as public
with the following whitelist CIDRs: ",
            "duration": "48µs"
        },
        {
            "name": "Azure Available Subnets",
            "description": "When an existing Azure subnet is chosen for pro
visioning an AKS cluster, the subnet must not be in use by any other cluster
. This is a restriction of Kubenet, which is the CNI used on the new cluster
. Although the subnet may have a routing table, it may not have any existing
 IP address associations.",
            "category": "NETWORK",
            "status": "PASSED",
            "message": "At least 1 valid subnet was found and can be used for
 cluster creation.",
            "detailedMessage": "The cluster can be provisioned using subnet
 liftie-dev.internal.2.westus2 in virtual network liftie-dev and resource gr
oup liftie-test",
            "duration": "917ms"
        },
        {
            "name": "Kubernetes API Server CIDR Security",
            "description": "CIDR blocks for whitelisting access to the Kube
rnetes API Server must not overlap restricted IP ranges.",
            "category": "NETWORK",
            "status": "SKIPPED",
            "message": "Skipping CIDR validation for whitelisting because it
 is not enabled.",
            "detailedMessage": "The ability to secure access the Kubernetes
 API server via a list of allowed CIDRs is not enabled. This can be enabled
 either in the controlplane (currently false) or via the provisioning reques
t (currently false).",
            "duration": "11µs"
        },
        {
            "name": "Service CIDR Validation",
            "description": "CIDR blocks that Kubernetes assigns service IP
addresses from should not overlap with any other networks that are peered or
 connected to existing VPC.",
            "category": "NETWORK",
            "status": "SKIPPED",
```

```
            "message": "Service CIDR is missing in Network Profile.",
            "detailedMessage": "VPC validation is executed only if the VPC a
lready exists and a service CIDR is specified in the network profile.",
            "duration": "426µs"
        },
        {
            "name": "Azure Autoscale Parameters",
            "description": "The following autoscale parameters for Azure,
which are specified during provisioning and update, need to be in multiples
of 60s. Autoscale parameters: scaleDownDelayAfterAdd, scaleDownDelayAfterFai
lure, scaleDownUnneededTime, scaleDownUnreadyTime.",
            "category": "DEPLOYMENT",
            "status": "SKIPPED",
            "message": "There are no autoscale parameters specified in the
request.",
            "detailedMessage": "The request did not contain an Autoscaler
structure.",
            "duration": "0s"
        }
    ]
}
```

Cloudera Data Engineering CLI exit codes

When a script that has been executed from the Cloudera Data Engineering CLI ends, a number is displayed in the command prompt window. This number is an exit code. If the script ends unexpectedly, this exit code can help you identify the error. These exit codes are applicable to Cloudera Data Engineering 1.19 or higher.

Exit code	Description
0	Success
1	Unknown Non-Retriable Error
2	Redirection Happened (HTTP 3xx status code received)
3	Bad Request (HTTP Status Code 400)
4	Authorization Error
5	Forbidden (HTTP Status Code 403)
6	Not Found (HTTP Status Code 404)
7	Method Not Allowed (HTTP Status Code 405)
8	Not Acceptable (HTTP Status Code 406)
9	Conflict (HTTP Status Code 409)
10	Gone (HTTP Status Code 410)
11	Length Required (HTTP Status Code 411)
12	Precondition Failed (HTTP Status Code 412)

Exit code	Description
13	Request Entity Too Large (HTTP Status Code 413)
14	URI Too Long (HTTP Status Code 414)
15	Unsupported Media Type (HTTP Status Code 415)
16	Range Not Satisfiable (HTTP Status Code 416)
17	Expectation Failed (HTTP Status Code 417)
18	Unprocessable Entity (HTTP Status Code 422)
19	Failed Dependency (HTTP Status Code 424)
20	Upgrade Required (HTTP Status Code 426)
21	Precondition Required (HTTP Status Code 428)
22	Request Header Fields Too Large (HTTP Status Code 431)
23	Unavailable For Legal Reasons (HTTP Status Code 451)
24	Internal Server Error (HTTP Status Code 500)
25	Not Implemented (HTTP Status Code 501)
26	HTTP Version Not Supported (HTTP Status Code 505)
27	Not Extended (HTTP Status Code 510)
28	Network Authentication Required (HTTP Status Code 511)
70	Unknown Retriable Error
71	Timeout during API call
72	Bad Gateway (HTTP Status Code 502)
73	Service Unavailable (HTTP Status Code 503)
74	Gateway Timeout (HTTP Status Code 504)
75	Request Timeout (HTTP Status Code 408)
76	Too Early (HTTP Status Code 425)
77	Too Many Requests (HTTP Status Code 429)

Cloudera Data Engineering Deep Analysis

You can run an on-demand *deep analysis* on a Cloudera Data Engineering job run. Deep analysis analyzes job logs and generates detailed information for a given job run, including memory utilization and stage-level analysis.

Because deep analysis consumes cluster resources by running an internal Cloudera Data Engineering job, you must run it manually for any job run you want to analyze or troubleshoot.

Enabling deep analysis on a Cloudera Data Engineering job from the Cloudera Data Engineering web UI

Using the Cloudera Data Engineering web UI you can enable an on-demand *deep analysis* on a Cloudera Data Engineering job run to generate detailed information, including memory utilization and stage-level analysis.

Before you begin

Important: Deep analysis is currently supported only for Spark 2 jobs. For Spark 3 jobs, the Spark Analysis toggle is greyed out.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view CDE Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

About this task

Because deep analysis consumes cluster resources by running an internal Cloudera Data Engineering job, you must enable and run it manually for any job run you want to analyze or troubleshoot.

Procedure

1. From the Cloudera Data Engineering Overview page, select the Cloudera Data Engineering service for the job you want to troubleshoot or analyze.
2. In the Virtual Clusters column, click the View Jobs icon for the cluster containing the job.
3. Select the job you want to analyze.
4. Select the Configuration tab and click Edit.
5. Select the Spark Analysis option.

   This enables collecting metrics during future runs of the job that you want to investigate.
6. Click Update.
7. Click Actions Run Now to run the job.
   Metrics are collected for deep analysis.

What to do next

Important: Do not forget to disable Spark Analysis on the Configuration tab once the deep analysis job completes. It adds unnecessary overhead and can have a negative impact on performance.

Enabling deep analysis on a Cloudera Data Engineering job run using the CLI

Using the Cloudera Data Engineering CLI you can enable an on-demand *deep analysis* on a Cloudera Data Engineering job run to generate detailed information, including memory utilization and stage-level analysis.

About this task

Important: Deep analysis is currently supported only for Spark 2 jobs.

Procedure

- You can enable deep analysis for a job run by using the `cde job run` or `cde spark submit` commands with the --enable-analysis flag.

  `cde job run`:

  ```
  cde job run --name [***JOB NAME***] --enable-analysis
  ```

  For example:

  ```
  cde job run --name test_job --enable-analysis
  ```

  `cde spark submit`:

  ```
  cde spark submit [***JAR/PY FILE***] --enable-analysis
  ```

  For example:

  ```
  cde spark submit test_job.jar --enable-analysis
  ```

  Metrics are collected for deep analysis.

Related Information

Running a Spark job using the CLI
Creating and updating Apache Spark jobs using the CLI
Submitting a Spark job using the CLI

Running deep analysis on a Cloudera Data Engineering job run

You can run an on-demand *deep analysis* on a Cloudera Data Engineering job run to generate detailed information, including memory utilization and stage-level analysis.

Before you begin

- You must enable deep analysis for the job you want to analyze and run the job once to collect data.

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. To view Cloudera Data Engineering Services, click Administration in the left-hand menu. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

About this task

You can only analyze job runs that took place after Spark Analysis was enabled for the job.

Procedure

1. From the Cloudera Data Engineering Overview page, select the Cloudera Data Engineering service for the job you want to troubleshoot or analyze.
2. In the Virtual Clusters column, click the View Jobs icon for the cluster containing the job.
3. In the Run History tab, click the Job ID for the job run you want to analyze.
4. Go to the Analysis tab.
5. Click Run Deep Analysis.

Results

After the deep analysis job completes, you can view additional job run information in the Analysis tab for the job run.

What to do next

Important: Do not forget to disable Spark Analysis on the Configuration tab once the deep analysis job completes. It adds unnecessary overhead and can have a negative impact on performance.

Cloudera Data Engineering 1.23.0

Upgrading Cloudera Data Engineering

Date published: 2020-07-30
Date modified: 2024-11-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Data Engineering upgrade version compatibility.....4

Upgrading Cloudera Data Engineering..... 4

In-place upgrade with Airflow Operators and Libraries..... 6

Airflow constraints file.....9
Troubleshooting Python or Airflow compatibility issues.....31
Debugging Airflow DAGs locally..... 33

Using the backup-restore-based upgrade script.....35

Upgrading Airflow if DAGs and packages are compatible with new Airflow version..... 38

Handling upgrade failures for Cloudera Data Engineering.....38

Handling ineligible upgrades for Cloudera Data Engineering.....41

Cloudera Data Engineering upgrade version compatibility

Cloudera Data Engineering provides the in-place upgrade option from the eligible versions to the latest version. If you are on a version that is ineligible for in-place upgrade, the in-place upgrade option is disabled and you can perform a backup and restore.

Checking your upgrade options

To check if your Cloudera Data Engineering version is eligible for the in-place upgrade option, see Support lifecycle policy.

In-place upgrades

To perform an in-place upgrade, you must be on an a version eligible for the in-place upgrade option. The *Cloudera Data Engineering versions eligible for in-place upgrade* table includes the specific AWS and Azure versions from which you can perform an in-place upgrade to the latest Cloudera Data Engineering version.

Table 1: Cloudera Data Engineering versions eligible for in-place upgrade

Supported upgrade versions	Target version
Azure: 1.22.0 and higher AWS: 1.22.0 and higher	1.23.0

Backup and restore

If you are on a version that is not eligible for the in-place upgrade option, to upgrade to the latest version, create a new Cloudera Data Engineering service and transfer your jobs to the new service by following the instructions in *Backing up and restoring Cloudera Data Engineering jobs*.

Related Information

In-place Upgrade with Airflow Operators and Libraries

Backing up Cloudera Data Engineering jobs on remote storage

Upgrading Cloudera Data Engineering

Cloudera Data Engineering supports in-place upgrades on both AWS and Azure. Cloudera Data Engineering provides an in-place upgrade option from the eligible versions to the latest version. If you are on a Cloudera Data Engineering version that is not eligible for the in-place upgrade, the in-place upgrade option is disabled and you can perform a backup and restore. The upgrades can be triggered by an Admin from the Cloudera Data Engineering user interface.

About this task

To check whether your Cloudera Data Engineering version is eligible for the in-place upgrade option, see Support lifecycle policy.

When you upgrade to the latest version in Cloudera Data Engineering, the upgrade process prepares your service for the upgrade with a click of a button. Your information is also backed up in this process. Learn how to complete the upgrade process here.

Important:

- Virtual clusters for this service are unavailable during the upgrade process and the upgrade puts your service into maintenance. During this time, scheduled jobs are paused and any active jobs are killed. The upgrade can take some time so be advised and alert your team of the downtime.
- Before the upgrade, delete any existing Airflow operators and libraries. If you have configured and built any Airflow libraries and operators, activate and then delete them before the upgrade. For more information, see In-place upgrade with Airflow Operators and Libraries and Upgrading Airflow if DAGs and packages are compatible with new Airflow version.

The following are not included in the backup:

- Python-venv resources
- Airflow custom operators and libraries
- Airflow connections
- Airflow variables
- Logs
  - Job Run logs (Driver, Executor, API)
  - Service logs
  - Virtual Cluster event logs
  - Spark Session logs (Including statement history)
  - Airflow DAG logs
- Job Runs
  - Job Run history
- Virtual Cluster end-points
- Resource: Docker runtimes

Before you begin

Ensure that the catchup option is not enabled for any user's Airflow jobs. Before the backup starts, if the Airflow DAG catchup options are enabled, disable them manually.

Procedure

1. In the Cloudera console, click the Data Engineering tile.

   The Cloudera Data Engineering **Home** page displays.
2. In the left navigation menu, click Administration, select a Service, and click Upgrade.
3. Click the Maintenance tab and click Upgrade Service.
4. Click Start Preparation.

   Cloudera Data Engineering prepares the service for you. If the service cannot be prepared, click Retry or Cancel Upgrade. Canceling the upgrade returns your service to the original state. Optionally, you can click Download Logs to view the Diagnostics page where you can generate a diagnostic bundle. If the preparation is successful, proceed to the next step. If the preparation and cancellation fails, you must contact Cloudera support.
5. Click Start Backup.

   Cloudera Data Engineering automatically creates a backup for you. The backup is used for restoring if an irrecoverable failure happens during the upgrade.

   Note: If the backup fails, you have the option to click Retry Upgrade or Contact Support. If the upgrade fails a second time, you have the option to contact support or perform a manual upgrade with jobs backup and restore. If the backup is successful, you can proceed with the upgrade.

6. When the backup is complete, to view the backup that was created, select Backup Library.

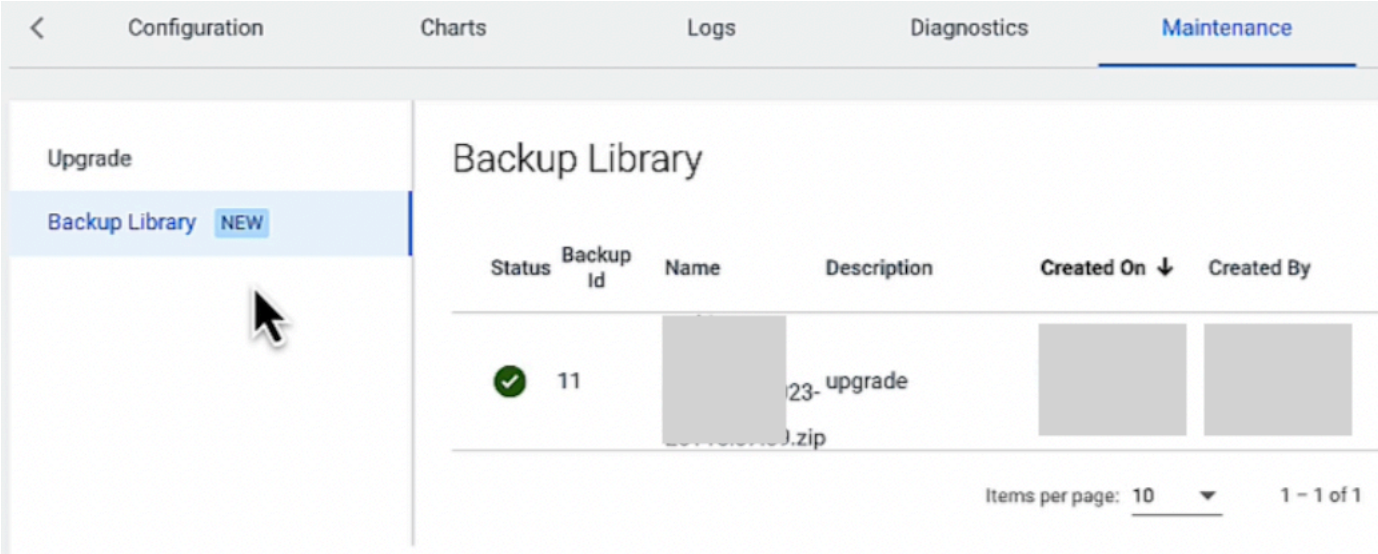

7. Click Start Upgrade.
8. After the upgrade is complete, to enable all paused jobs and sessions, click Resume Service.
   The service restarts.

Results

After the upgrade is complete and you resumed the service, the service operates on the upgraded version of Cloudera Data Engineering. If you cancel the upgrade, your service resumes the previous state, which enables all paused jobs and resumes all sessions.

Related Information

Handling upgrade failures for Cloudera Data Engineering

Cloudera Data Engineering upgrade version compatibility

In-place upgrade with Airflow Operators and Libraries

This section details Airflow considerations to avoid issues during and after the upgrade. Before performing an in-place upgrade of the Airflow operators and libraries, check whether the environment is compatible. This allows you to determine if the DAGs or the Python package dependencies require changes.

About this task

Note:

- Check the compatibility for all used Airflow operators and libraries. If you do not check the compatibility, the recreation of the Airflow operators and libraries, or the DAG execution can fail.

  The following is the recommended procedure. However, if you are confident that the DAGs and the packages in the Airflow Custom Libraries and Operators are compatible with the new Airflow version, instead of following this procedure, you can perform the upgrade as described in Upgrading Airflow if DAGs and packages are compatible with new Airflow version.
- After a Cloudera Data Engineering Upgrade, latest Airflow or Python versions such as Airflow 2.9 or Python 3.11 might cause backward compatibility issues with DAGs. These backward compatibility issues might cause downtime for the affected DAGs until they are fixed. To minimize the impact, the upgrade must be planned. The existing DAGs must be verified if they are compliant with the latest Cloudera Data Engineering version before the upgrade to minimize the number of DAGs that have to be fixed post-upgrade (if any).
- On Microsoft Azure, Cloudera Data Engineering version 1.23.0 does not support in-place upgrade from a version that uses the MySQL Single server to a version with MySQL Flexible server. In such a case, perform the upgrade using the procedure described in Handling upgrade failures for Cloudera Data Engineering.

Before you begin

Ensure that the catchup option is not enabled for any user's Airflow jobs. Before the backup starts, if the Airflow DAG catchup options are enabled, disable them manually.

Verify Airflow operators and libraries and DAGs in the newer Cloudera Data Engineering service:

Procedure

1. Create a new test Cloudera Data Engineering service with the latest Cloudera Data Engineering version.
2. Copy the DAGs from the current Cloudera Data Engineering for local checking:
   a) Update the DAG code for the latest Python version by using the tools mentioned in the Tools for porting Python from version 3.8 to version 3.11 on page 32 section.
   b) Update and test the DAG code for the latest Airflow version as described in the Debugging Airflow DAGs locally on page 33 section.
3. Obtain the requirements.txt for the existing Airflow environment. The requirements.txt is required after the upgrade to restore the environments.

   To obtain the Airflow environment requirements.txt for a virtual cluster, perform the following steps.

   For In Cloudera Data Engineering 1.21 and higher

   a. Obtain the environment information JSON from the Airflow environment API endpoint.

   1. Copy the JOBS API URL. For more information, check the Determine the virtual cluster endpoint URL step in *Configuring the CLI client*.
   2. Use the JOBS API URL as a base and add /admin/airflow/env.

      JOBS API URL example with /admin/airflow/env:

      ```
      https://t5kh5fjp.cde-hfrsp8ww.dex-priv.xcu2-8y8x.dev.cldr.work/dex/
      api/v1/admin/airflow/env
      ```

      Example for the JSON:

      ```
      cat pyenv.json
      {
        "status": "activated",
      ```

```
  "packages": [
 {
   "type": "python-module",
   "name": "mypy",
   "version": "1.2.0",
   "created": "2024-04-03T16:58:07Z"
 },
 {
   "type": "python-module",
   "name": "mypy-extensions",
   "version": "1.0.0",
   "created": "2024-04-03T16:58:07Z"
 },
 {
   "type": "python-module",
   "name": "tomli",
   "version": "2.0.1",
   "created": "2024-04-03T16:58:07Z"
 }
  ],
  "requirements": "mypy\n"
}
```

b. From the returned pyenv.json, generate a requirements.txt file with the jq tool:

```
cat pyenv.json | jq -r '.requirements'> requirements.txt
```

Example for the generated requirements.txt:

```
mypy
```

For In Cloudera Data Engineering 1.20

Use the CDE CLI to find and download the requirements.txt file which belongs to the Airflow environment in a virtual cluster:

a. Find the resource name that belongs to the Python environment. Optionally use jq to get the name:

```
cde resource list --filter "type[eq]airflow-python-env" --filter "st
atus[eq]active" --show-hidden=true | jq '.[0].name'
cde-airflow-pyenv-1712224823
```

b. Download the requirements.txt file:

```
cde resource download --name cde-airflow-pyenv-1712224823 --resource-
path requirements.txt
```

4. Verify the requirements.txt file against an Airflow constraints file.

a) Get the Python version used for Airflow from *Compatibility for Cloudera Data Engineering and Runtime components*.

b) Check requirements.txt against the Cloudera Data Engineering constraint file by creating a temporary Python virtual environment and use the dry-run feature of pip. If pip does not support dry-run, perform the check by installing the packages without the –dry-run flag:

```
# create a temporary local python virtual environment using the same pyt
hon version used in Airflow in Cloudera Data Engineering.
python -m venv
# source the environment
source venv/bin/activate
# check if there are any conflicts between the packages in the requir
ements.txt # file and the packages needed by airflow
```

```
# At least pip 22.2 is needed for the dry-run feature.
(venv) python -m pip install -r requirements.txt -c "https://raw.github
usercontent.com/apache/airflow/constraints-2.9.3/constraints-3.11.txt" -
-dry-run
```

5. Create a compatible Python environment in a test virtual cluster in the test Service.

   Python packages might need to be updated to work with the Airflow service in the new Cloudera Data Engineering version.
6. Backup and restore your Airflow jobs or create all the necessary Airflow jobs in the test Cloudera Data Engineering Service that you created in step 1 on page 7.
7. If the DAG parsing fails, go to the Airflow UI to identify the impacted DAGs.
8. Fix the DAGs or update the dependencies in the requirements.txt to fix the issues.
9. Conduct a test run of these DAGs if they call custom libraries at runtime to ensure that the code defined within the tasks is also valid.
10. If the DAG execution fails, go to the Airflow job logs to identify the impacted DAGs.
11. Fix the DAGs or update the dependencies in the requirements.txt to fix the issues.

    Backport all DAG changes to the Cloudera Data Engineering service to be upgraded, so that DAGs are compatible with the previous and the new version of Airflow. This may require a retest on an older Cloudera Data Engineering service test virtual cluster, if the production environment cannot be updated easily. If changes are needed for the older Cloudera Data Engineering service, test the changes again on the newer Cloudera Data Engineering service. Verify if both services work as expected. To fix the compatibility issues regarding Python version change, see Python upgrade from version 3.8 to version 3.11 on page 32 and to fix issues regarding Airflow, see Airflow upgrade from version 2.7.3 to version 2.9.3 on page 33.
12. Delete the test service.
13. Delete all Airflow operators and libraries in all virtual clusters in the Cloudera Data Engineering service before executing the upgrade.
14. Continue with the upgrade.
15. After a successful upgrade, build and activate the Airflow operators and libraries in the virtual clusters from the updated and fixed requirements.txt.
16. Verify that the Airflow jobs are properly running in the Airflow UI and Cloudera Data Engineering Jobs UI.
17. Remove and retest backward compatible changes made in the DAGs for the previous Cloudera Data Engineering version. Keep the DAGs updated to the latest Airflow version in Cloudera Data Engineering.

Related Information

Configuring the CLI client

Compatibility for Cloudera Data Engineering and Runtime components

Upgrading Airflow if DAGs and packages are compatible with new Airflow version

Airflow constraints file

Airflow constraints file

Before performing an in-place upgrade of the Airflow operators and libraries, check whether the environment is compatible. Use the Airflow constraints file to verify its content against the requirements.txt file that belongs to the Airflow environment in a virtual cluster.

For more information about checking the Airflow constraints file against the requirements.txt file, see step 4 on page 8 in *In-place upgrade with Airflow Operators and Libraries*.

For Upgrade to 1.23

Airflow version 2.9.3 constraints file:

```
apache-airflow==2.9.3
waitress==2.1.1
argparse==1.4.0
Authlib==1.3.1
Babel==2.15.0
ConfigUpdater==3.2
Deprecated==1.2.14
Events==0.5
Flask-AppBuilder==4.5.0
Flask-Babel==2.0.0
Flask-Bcrypt==1.0.1
Flask-Caching==2.3.0
Flask-JWT-Extended==4.6.0
Flask-Limiter==3.7.0
Flask-Login==0.6.3
Flask-SQLAlchemy==2.5.1
Flask-Session==0.5.0
Flask-WTF==1.2.1
Flask==2.2.5
GitPython==3.1.43
JPype1==1.5.0
JayDeBeApi==1.2.3
Jinja2==3.1.4
Js2Py==0.74
Mako==1.3.5
Markdown==3.6
MarkupSafe==2.1.5
PyAthena==3.8.3
PyGithub==2.3.0
PyHive==0.7.0
PyJWT==2.8.0
PyNaCl==1.5.0
PyYAML==6.0.1
Pygments==2.18.0
SQLAlchemy-JSONField==1.0.2
SQLAlchemy-Utils==0.41.2
SQLAlchemy==1.4.52
SecretStorage==3.3.3
Sphinx==5.3.0
WTForms==3.1.2
Werkzeug==2.2.3
adal==1.2.7
adlfs==2024.4.1
aiobotocore==2.13.1
aiofiles==23.2.1
aiohttp==3.9.5
aioitertools==0.11.0
aioresponses==0.7.6
aiosignal==1.3.1
alabaster==0.7.16
alembic==1.13.2
alibabacloud-adb20211201==2.0.0
alibabacloud-tea==0.3.9
alibabacloud_credentials==0.3.4
alibabacloud_endpoint_util==0.0.3
alibabacloud_gateway_spi==0.0.1
alibabacloud_openapi_util==0.2.2
alibabacloud_tea_openapi==0.3.10
alibabacloud_tea_util==0.3.12
alibabacloud_tea_xml==0.0.2
aliyun-python-sdk-core==2.15.1
```

```
aliyun-python-sdk-kms==2.16.3
amqp==5.2.0
analytics-python==1.2.9
annotated-types==0.7.0
ansicolors==1.1.8
anyascii==0.3.2
anyio==4.4.0
apache-airflow-providers-airbyte==3.8.1
apache-airflow-providers-alibaba==2.8.1
apache-airflow-providers-amazon==8.25.0
apache-airflow-providers-apache-beam==5.7.1
apache-airflow-providers-apache-cassandra==3.5.1
apache-airflow-providers-apache-drill==2.7.2
apache-airflow-providers-apache-druid==3.10.1
apache-airflow-providers-apache-flink==1.4.2
apache-airflow-providers-apache-hdfs==4.4.2
apache-airflow-providers-apache-hive==8.1.2
apache-airflow-providers-apache-iceberg==1.0.0
apache-airflow-providers-apache-impala==1.4.1
apache-airflow-providers-apache-kafka==1.5.0
apache-airflow-providers-apache-kylin==3.6.2
apache-airflow-providers-apache-livy==3.8.1
apache-airflow-providers-apache-pig==4.4.1
apache-airflow-providers-apache-pinot==4.4.1
apache-airflow-providers-apache-spark==4.8.2
apache-airflow-providers-apprise==1.3.1
apache-airflow-providers-arangodb==2.5.1
apache-airflow-providers-asana==2.5.1
apache-airflow-providers-atlassian-jira==2.6.1
apache-airflow-providers-celery==3.7.2
apache-airflow-providers-cloudant==3.5.2
apache-airflow-providers-cncf-kubernetes==8.3.3
apache-airflow-providers-cohere==1.2.1
apache-airflow-providers-common-io==1.3.2
apache-airflow-providers-common-sql==1.14.2
apache-airflow-providers-databricks==6.7.0
apache-airflow-providers-datadog==3.6.1
apache-airflow-providers-dbt-cloud==3.9.0
apache-airflow-providers-dingding==3.5.1
apache-airflow-providers-discord==3.7.1
apache-airflow-providers-docker==3.12.2
apache-airflow-providers-elasticsearch==5.4.1
apache-airflow-providers-exasol==4.5.2
apache-airflow-providers-fab==1.2.1
apache-airflow-providers-facebook==3.5.2
apache-airflow-providers-ftp==3.10.0
apache-airflow-providers-github==2.6.2
apache-airflow-providers-google==10.21.0
apache-airflow-providers-grpc==3.5.2
apache-airflow-providers-hashicorp==3.7.1
apache-airflow-providers-http==4.12.0
apache-airflow-providers-imap==3.6.1
apache-airflow-providers-influxdb==2.6.0
apache-airflow-providers-jdbc==4.3.1
apache-airflow-providers-jenkins==3.6.1
apache-airflow-providers-microsoft-azure==10.3.0
apache-airflow-providers-microsoft-mssql==3.7.2
apache-airflow-providers-microsoft-psrp==2.7.1
apache-airflow-providers-microsoft-winrm==3.5.1
apache-airflow-providers-mongo==4.1.2
apache-airflow-providers-mysql==5.6.2
apache-airflow-providers-neo4j==3.6.1
apache-airflow-providers-odbc==4.6.2
apache-airflow-providers-openai==1.2.2
```

```
apache-airflow-providers-openfaas==3.5.1
apache-airflow-providers-openlineage==1.9.1
apache-airflow-providers-opensearch==1.3.0
apache-airflow-providers-opsgenie==5.6.1
apache-airflow-providers-oracle==3.10.3
apache-airflow-providers-pagerduty==3.7.2
apache-airflow-providers-papermill==3.7.2
apache-airflow-providers-pgvector==1.2.1
apache-airflow-providers-pinecone==2.0.1
apache-airflow-providers-postgres==5.11.2
apache-airflow-providers-presto==5.5.2
apache-airflow-providers-qdrant==1.1.1
apache-airflow-providers-redis==3.7.1
apache-airflow-providers-salesforce==5.7.2
apache-airflow-providers-samba==4.7.1
apache-airflow-providers-segment==3.5.1
apache-airflow-providers-sendgrid==3.5.1
apache-airflow-providers-sftp==4.10.2
apache-airflow-providers-singularity==3.5.1
apache-airflow-providers-slack==8.7.1
apache-airflow-providers-smtp==1.7.1
apache-airflow-providers-snowflake==5.6.0
apache-airflow-providers-sqlite==3.8.1
apache-airflow-providers-ssh==3.11.2
apache-airflow-providers-tableau==4.5.2
apache-airflow-providers-tabular==1.5.1
apache-airflow-providers-telegram==4.5.2
apache-airflow-providers-teradata==2.4.0
apache-airflow-providers-trino==5.7.2
apache-airflow-providers-vertica==3.8.2
apache-airflow-providers-weaviate==1.4.2
apache-airflow-providers-yandex==3.11.2
apache-airflow-providers-zendesk==4.7.1
apache-beam==2.57.0
apispec==6.6.1
apprise==1.8.0
argcomplete==3.4.0
asana==3.2.3
asgiref==3.8.1
asn1crypto==1.5.1
astroid==2.15.8
asttokens==2.4.1
asyncssh==2.15.0
atlasclient==1.0.0
atlassian-python-api==3.41.14
attrs==23.2.0
aws-sam-translator==1.89.0
aws-xray-sdk==2.14.0
azure-batch==14.2.0
azure-common==1.1.28
azure-core==1.30.2
azure-cosmos==4.7.0
azure-datalake-store==0.0.53
azure-identity==1.17.1
azure-keyvault-secrets==4.8.0
azure-kusto-data==4.5.1
azure-mgmt-containerinstance==10.1.0
azure-mgmt-containerregistry==10.3.0
azure-mgmt-core==1.4.0
azure-mgmt-cosmosdb==9.5.1
azure-mgmt-datafactory==8.0.0
azure-mgmt-datalake-nspkg==3.0.1
azure-mgmt-datalake-store==0.5.0
azure-mgmt-nspkg==3.0.2
```

```
azure-mgmt-resource==23.1.1
azure-mgmt-storage==21.2.1
azure-nspkg==3.0.2
azure-servicebus==7.12.2
azure-storage-blob==12.20.0
azure-storage-file-datalake==12.15.0
azure-storage-file-share==12.16.0
azure-synapse-artifacts==0.19.0
azure-synapse-spark==0.7.0
backoff==2.2.1
backports.tarfile==1.2.0
bcrypt==4.1.3
beautifulsoup4==4.12.3
billiard==4.2.0
bitarray==2.9.2
black==24.4.2
blinker==1.8.2
boto3==1.34.131
botocore==1.34.131
cachelib==0.9.0
cachetools==5.3.3
cassandra-driver==3.29.1
cattrs==23.2.3
celery==5.4.0
certifi==2024.7.4
cffi==1.16.0
cfgv==3.4.0
cfn-lint==1.6.0
cgroupspy==0.2.3
chardet==5.2.0
charset-normalizer==3.3.2
checksumdir==1.2.0
ciso8601==2.3.1
click-didyoumean==0.3.1
click-plugins==1.1.1
click-repl==0.3.0
click==8.1.7
clickclick==20.10.2
cloudant==2.15.0
cloudpickle==2.2.1
cohere==4.57
colorama==0.4.6
colorlog==4.8.0
comm==0.2.2
confluent-kafka==2.5.0
connexion==2.14.2
coverage==7.6.0
crcmod==1.7
cron-descriptor==1.4.3
croniter==2.0.5
cryptography==41.0.7
curlify==2.2.1
databricks-sql-connector==2.9.6
datadog==0.49.1
db-dtypes==1.2.0
debugpy==1.8.2
decorator==5.1.1
defusedxml==0.7.1
deltalake==0.18.2
diagrams==0.23.4
dill==0.3.1.1
distlib==0.3.8
distro==1.9.0
dnspython==2.6.1
```

```
docker==7.1.0
docopt==0.6.2
docstring_parser==0.16
docutils==0.16
duckdb==1.0.0
elastic-transport==8.13.1
elasticsearch==8.14.0
email_validator==2.2.0
entrypoints==0.4
eralchemy2==1.4.1
et-xmlfile==1.1.0
eventlet==0.36.1
execnet==2.1.1
executing==2.0.1
facebook_business==20.0.0
fastavro==1.9.5
fasteners==0.19
fastjsonschema==2.20.0
filelock==3.15.4
flower==2.0.1
frozenlist==1.4.1
fsspec==2023.12.2
future==1.0.0
gcloud-aio-auth==4.2.3
gcloud-aio-bigquery==7.1.0
gcloud-aio-storage==9.2.0
gcsfs==2023.12.2.post1
geomet==0.2.1.post1
gevent==24.2.1
gitdb==4.0.11
google-ads==24.1.0
google-analytics-admin==0.22.8
google-api-core==2.19.1
google-api-python-client==2.137.0
google-auth-httplib2==0.2.0
google-auth-oauthlib==1.2.1
google-auth==2.32.0
google-cloud-aiplatform==1.59.0
google-cloud-appengine-logging==1.4.4
google-cloud-audit-log==0.2.5
google-cloud-automl==2.13.4
google-cloud-batch==0.17.22
google-cloud-bigquery-datatransfer==3.15.4
google-cloud-bigquery==3.20.1
google-cloud-bigtable==2.24.0
google-cloud-build==3.24.1
google-cloud-compute==1.19.1
google-cloud-container==2.49.0
google-cloud-core==2.4.1
google-cloud-datacatalog==3.19.1
google-cloud-dataflow-client==0.8.11
google-cloud-dataform==0.5.10
google-cloud-dataplex==2.2.1
google-cloud-dataproc-metastore==1.15.4
google-cloud-dataproc==5.10.1
google-cloud-dlp==3.18.1
google-cloud-kms==2.24.1
google-cloud-language==2.13.4
google-cloud-logging==3.10.0
google-cloud-memcache==1.9.4
google-cloud-monitoring==2.22.1
google-cloud-orchestration-airflow==1.13.0
google-cloud-os-login==2.14.5
google-cloud-pubsub==2.22.0
```

```
google-cloud-redis==2.15.4
google-cloud-resource-manager==1.12.4
google-cloud-run==0.10.7
google-cloud-secret-manager==2.20.1
google-cloud-spanner==3.47.0
google-cloud-speech==2.26.1
google-cloud-storage-transfer==1.11.4
google-cloud-storage==2.17.0
google-cloud-tasks==2.16.4
google-cloud-texttospeech==2.16.4
google-cloud-translate==3.15.4
google-cloud-videointelligence==2.13.4
google-cloud-vision==3.7.3
google-cloud-workflows==1.14.4
google-crc32c==1.5.0
google-re2==1.1.20240702
google-resumable-media==2.7.1
googleapis-common-protos==1.63.2
graphql-core==3.2.3
graphviz==0.20.3
greenlet==3.0.3
grpc-google-iam-v1==0.13.1
grpc-interceptor==0.15.4
grpcio-gcp==0.2.2
grpcio-health-checking==1.62.2
grpcio-status==1.62.2
grpcio-tools==1.62.2
grpcio==1.64.1
gssapi==1.8.3
gunicorn==22.0.0
h11==0.14.0
h2==4.1.0
hatch==1.12.0
hatchling==1.25.0
hdfs==2.7.3
hmsclient==0.1.1
hpack==4.0.0
httpcore==1.0.5
httplib2==0.22.0
httpx==0.27.0
humanize==4.10.0
hvac==2.3.0
hyperframe==6.0.1
hyperlink==21.0.0
icdiff==2.0.7
identify==2.6.0
idna==3.7
ijson==3.3.0
imagesize==1.4.1
importlib-metadata==6.11.0
importlib_resources==6.4.0
impyla==0.19.0
incremental==22.10.0
inflection==0.5.1
influxdb-client==1.44.0
iniconfig==2.0.0
ipdb==0.13.13
ipykernel==6.29.5
ipython==8.26.0
isodate==0.6.1
itsdangerous==2.2.0
jaraco.classes==3.4.0
jaraco.context==5.3.0
jaraco.functools==4.0.1
```

```
jedi==0.19.1
jeepney==0.8.0
jmespath==0.10.0
joserfc==0.12.0
json-merge-patch==0.2
jsondiff==2.1.1
jsonpatch==1.33
jsonpath-ng==1.6.1
jsonpickle==3.2.2
jsonpointer==3.0.0
jsonschema-path==0.3.3
jsonschema-specifications==2023.12.1
jsonschema==4.23.0
jupyter_client==8.6.2
jupyter_core==5.7.2
keyring==25.2.1
kombu==5.3.7
krb5==0.5.1
kubernetes==29.0.0
kubernetes_asyncio==29.0.0
kylinpy==2.8.4
lazy-object-proxy==1.10.0
ldap3==2.9.1
limits==3.13.0
linkify-it-py==2.0.3
lockfile==0.12.2
loguru==0.7.2
looker-sdk==24.10.0
lxml==5.2.2
lz4==4.3.3
markdown-it-py==3.0.0
marshmallow-oneofschema==3.1.1
marshmallow-sqlalchemy==0.28.2
marshmallow==3.21.3
matplotlib-inline==0.1.7
mdit-py-plugins==0.4.1
mdurl==0.1.2
mergedeep==1.3.4
methodtools==0.4.7
microsoft-kiota-abstractions==1.3.3
microsoft-kiota-authentication-azure==1.0.0
microsoft-kiota-http==1.3.2
mmhash3==3.0.1
mongomock==4.1.2
more-itertools==10.3.0
moto==5.0.11
mpmath==1.3.0
msal-extensions==1.2.0
msal==1.29.0
msgraph-core==1.1.1
msrest==0.7.1
msrestazure==0.6.4.post1
multi_key_dict==2.0.3
multidict==6.0.5
mypy-boto3-appflow==1.34.0
mypy-boto3-rds==1.34.135
mypy-boto3-redshift-data==1.34.0
mypy-boto3-s3==1.34.138
mypy-extensions==1.0.0
mypy==1.9.0
mysql-connector-python==9.0.0
mysqlclient==2.2.4
nbclient==0.10.0
nbformat==5.10.4
```

```
neo4j==5.22.0
nest-asyncio==1.6.0
networkx==3.3
nh3==0.2.18
nodeenv==1.9.1
numpy==1.26.4
oauthlib==3.2.2
objsize==0.7.0
openai==1.35.13
openapi-schema-validator==0.6.2
openapi-spec-validator==0.7.1
openlineage-integration-common==1.18.0
openlineage-python==1.18.0
openlineage_sql==1.18.0
openpyxl==3.1.5
opensearch-py==2.6.0
opentelemetry-api==1.25.0
opentelemetry-exporter-otlp-proto-common==1.25.0
opentelemetry-exporter-otlp-proto-grpc==1.25.0
opentelemetry-exporter-otlp-proto-http==1.25.0
opentelemetry-exporter-otlp==1.25.0
opentelemetry-exporter-prometheus==0.46b0
opentelemetry-proto==1.25.0
opentelemetry-sdk==1.25.0
opentelemetry-semantic-conventions==0.46b0
opsgenie-sdk==2.1.5
oracledb==2.2.1
ordered-set==4.1.0
orjson==3.10.6
oss2==2.18.6
packaging==24.1
pandas-gbq==0.23.1
pandas-stubs==2.2.2.240603
pandas==2.1.4
papermill==2.6.0
paramiko==3.4.0
parso==0.8.4
pathable==0.4.3
pathspec==0.12.1
pbr==6.0.0
pdpyras==5.2.0
pendulum==3.0.0
pexpect==4.9.0
pgvector==0.3.1
pinecone-client==4.1.2
pinecone-plugin-interface==0.0.7
pinotdb==5.3.0
pipdeptree==2.23.1
pipx==1.6.0
pkginfo==1.10.0
platformdirs==4.2.2
pluggy==1.5.0
ply==3.11
plyvel==1.5.1
portalocker==2.10.0
pprintpp==0.4.0
pre-commit==3.7.1
presto-python-client==0.8.4
prison==0.2.1
prometheus_client==0.20.0
prompt_toolkit==3.0.47
proto-plus==1.24.0
protobuf==4.25.3
psutil==6.0.0
```

```
psycopg2-binary==2.9.9
ptyprocess==0.7.0
pure-eval==0.2.2
pure-sasl==0.6.2
py-partiql-parser==0.5.5
py4j==0.10.9.7
pyOpenSSL==24.1.0
pyarrow-hotfix==0.6
pyarrow==16.1.0
pyasn1-modules==0.3.0
pyasn1==0.5.1
pycountry==24.6.1
pycparser==2.22
pycryptodome==3.20.0
pydantic==2.8.2
pydantic_core==2.20.1
pydata-google-auth==1.8.2
pydot==1.4.2
pydruid==0.6.9
pyenchant==3.2.2
pyexasol==0.26.0
pygraphviz==1.13
pyiceberg==0.6.1
pyjsparser==2.7.1
pykerberos==1.2.4
pymongo==4.8.0
pymssql==2.3.0
pyodbc==5.1.0
pyparsing==3.1.2
pypsrp==0.8.1
pyspark==3.5.1
pyspnego==0.11.0
pytest-asyncio==0.23.7
pytest-cov==5.0.0
pytest-custom-exit-code==0.3.0
pytest-icdiff==0.9
pytest-instafail==0.5.0
pytest-mock==3.14.0
pytest-rerunfailures==14.0
pytest-timeouts==1.2.1
pytest-xdist==3.6.1
pytest==7.4.4
python-arango==8.0.0
python-daemon==3.0.1
python-dateutil==2.9.0.post0
python-dotenv==1.0.1
python-http-client==3.3.7
python-jenkins==1.8.2
python-ldap==3.4.4
python-nvd3==0.16.0
python-slugify==8.0.4
python-telegram-bot==21.3
python3-saml==1.16.0
pytz==2024.1
pywinrm==0.4.3
pyzmq==26.0.3
qdrant-client==1.10.1
reactivex==4.0.4
readme_renderer==43.0
redis==5.0.7
redshift-connector==2.1.2
referencing==0.35.1
regex==2024.5.15
requests-file==2.1.0
```

```
requests-kerberos==0.15.0
requests-mock==1.12.1
requests-oauthlib==1.3.1
requests-toolbelt==1.0.0
requests==2.31.0
requests_ntlm==1.3.0
responses==0.25.3
restructuredtext_lint==1.4.0
rfc3339-validator==0.1.4
rfc3986==2.0.0
rich-argparse==1.5.2
rich-click==1.8.3
rich==13.7.1
rpds-py==0.19.0
rsa==4.9
ruff==0.3.3
s3fs==2023.12.2
s3transfer==0.10.2
scramp==1.4.5
scrapbook==0.5.0
semver==3.0.2
sendgrid==6.11.0
sentinels==1.0.0
sentry-sdk==2.9.0
setproctitle==1.3.3
shapely==2.0.4
shellingham==1.5.4
simple-salesforce==1.12.6
six==1.16.0
slack_sdk==3.31.0
smbprotocol==1.13.0
smmap==5.0.1
sniffio==1.3.1
snowballstemmer==2.2.0
snowflake-connector-python==3.11.0
snowflake-sqlalchemy==1.6.1
sortedcontainers==2.4.0
soupsieve==2.5
sphinx-airflow-theme==0.0.12
sphinx-argparse==0.4.0
sphinx-autoapi==2.1.1
sphinx-copybutton==0.5.2
sphinx-jinja==2.0.2
sphinx-rtd-theme==2.0.0
sphinx_design==0.6.0
sphinxcontrib-applehelp==1.0.8
sphinxcontrib-devhelp==1.0.6
sphinxcontrib-htmlhelp==2.0.5
sphinxcontrib-httpdomain==1.8.1
sphinxcontrib-jquery==4.1
sphinxcontrib-jsmath==1.0.1
sphinxcontrib-qthelp==1.0.7
sphinxcontrib-redoc==1.6.0
sphinxcontrib-serializinghtml==1.1.5
sphinxcontrib-spelling==8.0.0
spython==0.3.13
sqlalchemy-bigquery==1.11.0
sqlalchemy-redshift==0.8.14
sqlalchemy-spanner==1.7.0
sqlalchemy_drill==1.1.5
sqlparse==0.5.0
sshtunnel==0.4.0
stack-data==0.6.3
starkbank-ecdsa==2.2.0
```

```
statsd==4.0.1
std-uritemplate==1.0.3
strictyaml==1.7.3
sympy==1.13.0
tableauserverclient==0.31
tabulate==0.9.0
tenacity==8.5.0
teradatasql==20.0.0.13
teradatasqlalchemy==20.0.0.1
termcolor==2.4.0
text-unidecode==1.3
thrift-sasl==0.4.3
thrift==0.16.0
time-machine==2.14.2
tomli_w==1.0.0
tomlkit==0.13.0
tornado==6.4.1
towncrier==23.11.0
tqdm==4.66.4
traitlets==5.14.3
trino==0.329.0
trove-classifiers==2024.7.2
twine==5.1.1
typed-ast==1.5.5
types-Deprecated==1.2.9.20240311
types-Markdown==3.6.0.20240316
types-PyMySQL==1.1.0.20240524
types-PyYAML==6.0.12.20240311
types-aiofiles==24.1.0.20240626
types-certifi==2021.10.8.3
types-cffi==1.16.0.20240331
types-croniter==2.0.0.20240423
types-docutils==0.21.0.20240711
types-paramiko==3.4.0.20240423
types-protobuf==5.27.0.20240626
types-pyOpenSSL==24.1.0.20240425
types-python-dateutil==2.9.0.20240316
types-python-slugify==8.0.2.20240310
types-pytz==2024.1.0.20240417
types-redis==4.6.0.20240425
types-requests==2.32.0.20240712
types-setuptools==70.3.0.20240710
types-tabulate==0.9.0.20240106
types-termcolor==1.1.6.2
types-toml==0.10.8.20240310
typing_extensions==4.12.2
tzdata==2024.1
tzlocal==5.2
uc-micro-py==1.0.3
unicodecsv==0.14.1
universal_pathlib==0.2.2
uritemplate==4.1.1
urllib3==2.0.7
userpath==1.9.2
uv==0.2.22
validators==0.28.3
vertica-python==1.3.8
vine==5.1.0
virtualenv==20.26.3
watchtower==3.2.0
wcwidth==0.2.13
weaviate-client==4.6.5
websocket-client==1.8.0
wirerope==0.4.7
```

```
wrapt==1.16.0
xmlsec==1.3.13
xmltodict==0.13.0
yamllint==1.35.1
yandex-query-client==0.1.4
yandexcloud==0.291.0
yarl==1.9.4
zeep==4.2.1
zenpy==2.0.49
zipp==3.19.2
zope.event==5.0
zope.interface==6.4.post2
zstandard==0.22.0
```

For Upgrade to 1.22

Airflow version 2.7.3 constraints file:

```
apache-airflow==2.7.3
waitress==2.1.1
argparse==1.4.0
Authlib==1.2.1
Babel==2.13.1
ConfigUpdater==3.1.1
Deprecated==1.2.14
Flask-AppBuilder==4.3.6
Flask-Babel==2.0.0
Flask-Bcrypt==1.0.1
Flask-Caching==2.1.0
Flask-JWT-Extended==4.5.3
Flask-Limiter==3.5.0
Flask-Login==0.6.3
Flask-SQLAlchemy==2.5.1
Flask-Session==0.5.0
Flask-WTF==1.2.1
Flask==2.2.5
GitPython==3.1.40
JPype1==1.4.1
JayDeBeApi==1.2.3
Jinja2==3.1.2
Js2Py==0.74
Mako==1.2.4
Markdown==3.5.1
MarkupSafe==2.1.3
PyGithub==2.1.1
PyHive==0.7.0
PyJWT==2.8.0
PyNaCl==1.5.0
PyYAML==6.0.1
Pygments==2.16.1
SQLAlchemy-JSONField==1.0.1.post0
SQLAlchemy-Utils==0.41.1
SQLAlchemy==1.4.50
SecretStorage==3.3.3
Sphinx==5.3.0
WTForms==3.0.1
Werkzeug==2.2.3
adal==1.2.7
adlfs==2023.10.0
aiobotocore==2.7.0
aiofiles==23.2.1
aiohttp==3.8.6
aioitertools==0.11.0
```

```
aioresponses==0.7.4
aiosignal==1.3.1
alabaster==0.7.13
alembic==1.12.1
alibabacloud-adb20211201==1.0.0
alibabacloud-credentials==0.3.2
alibabacloud-endpoint-util==0.0.3
alibabacloud-gateway-spi==0.0.1
alibabacloud-openapi-util==0.2.2
alibabacloud-tea-openapi==0.3.7
alibabacloud-tea-util==0.3.11
alibabacloud-tea-xml==0.0.2
alibabacloud-tea==0.3.3
aliyun-python-sdk-core==2.14.0
aliyun-python-sdk-kms==2.16.2
amqp==5.1.1
analytics-python==1.4.post1
annotated-types==0.6.0
ansiwrap==0.8.4
anyascii==0.3.2
anyio==4.0.0
apache-airflow-providers-airbyte==3.4.0
apache-airflow-providers-alibaba==2.6.0
apache-airflow-providers-amazon==8.10.0
apache-airflow-providers-apache-beam==5.3.0
apache-airflow-providers-apache-cassandra==3.3.0
apache-airflow-providers-apache-drill==2.5.0
apache-airflow-providers-apache-druid==3.6.0
apache-airflow-providers-apache-flink==1.2.0
apache-airflow-providers-apache-hdfs==4.2.0
apache-airflow-providers-apache-hive==6.2.0
apache-airflow-providers-apache-impala==1.2.0
apache-airflow-providers-apache-kafka==1.2.0
apache-airflow-providers-apache-kylin==3.3.0
apache-airflow-providers-apache-livy==3.6.0
apache-airflow-providers-apache-pig==4.2.0
apache-airflow-providers-apache-pinot==4.2.0
apache-airflow-providers-apache-spark==4.3.0
apache-airflow-providers-apache-sqoop==4.1.0
apache-airflow-providers-apprise==1.1.0
apache-airflow-providers-arangodb==2.3.0
apache-airflow-providers-asana==2.3.0
apache-airflow-providers-atlassian-jira==2.2.0
apache-airflow-providers-celery==3.4.1
apache-airflow-providers-cloudant==3.3.0
apache-airflow-providers-cncf-kubernetes==7.8.0
apache-airflow-providers-common-sql==1.8.0
apache-airflow-providers-daskexecutor==1.1.0
apache-airflow-providers-databricks==4.7.0
apache-airflow-providers-datadog==3.4.0
apache-airflow-providers-dbt-cloud==3.4.0
apache-airflow-providers-dingding==3.3.0
apache-airflow-providers-discord==3.4.0
apache-airflow-providers-docker==3.8.0
apache-airflow-providers-elasticsearch==5.1.0
apache-airflow-providers-exasol==4.3.0
apache-airflow-providers-facebook==3.3.0
apache-airflow-providers-ftp==3.6.0
apache-airflow-providers-github==2.4.0
apache-airflow-providers-google==10.11.0
apache-airflow-providers-grpc==3.3.0
apache-airflow-providers-hashicorp==3.5.0
apache-airflow-providers-http==4.6.0
apache-airflow-providers-imap==3.4.0
```

```
apache-airflow-providers-influxdb==2.3.0
apache-airflow-providers-jdbc==4.1.0
apache-airflow-providers-jenkins==3.4.0
apache-airflow-providers-microsoft-azure==8.1.0
apache-airflow-providers-microsoft-mssql==3.5.0
apache-airflow-providers-microsoft-psrp==2.4.0
apache-airflow-providers-microsoft-winrm==3.3.0
apache-airflow-providers-mongo==3.4.0
apache-airflow-providers-mysql==5.4.0
apache-airflow-providers-neo4j==3.4.0
apache-airflow-providers-odbc==4.1.0
apache-airflow-providers-openfaas==3.3.0
apache-airflow-providers-openlineage==1.2.0
apache-airflow-providers-opensearch==1.0.0
apache-airflow-providers-opsgenie==5.2.0
apache-airflow-providers-oracle==3.8.0
apache-airflow-providers-pagerduty==3.4.0
apache-airflow-providers-papermill==3.4.0
apache-airflow-providers-plexus==3.3.0
apache-airflow-providers-postgres==5.7.1
apache-airflow-providers-presto==5.2.1
apache-airflow-providers-redis==3.4.0
apache-airflow-providers-salesforce==5.5.0
apache-airflow-providers-samba==4.3.0
apache-airflow-providers-segment==3.3.0
apache-airflow-providers-sendgrid==3.3.0
apache-airflow-providers-sftp==4.7.0
apache-airflow-providers-singularity==3.3.0
apache-airflow-providers-slack==8.3.0
apache-airflow-providers-smtp==1.4.1
apache-airflow-providers-snowflake==5.1.0
apache-airflow-providers-sqlite==3.5.0
apache-airflow-providers-ssh==3.8.1
apache-airflow-providers-tableau==4.3.0
apache-airflow-providers-tabular==1.3.0
apache-airflow-providers-telegram==4.2.0
apache-airflow-providers-trino==5.4.0
apache-airflow-providers-vertica==3.6.0
apache-airflow-providers-zendesk==4.4.0
apache-beam==2.51.0
apispec==6.3.0
apprise==1.6.0
argcomplete==3.1.3
arrow==1.3.0
asana==3.2.2
asgiref==3.7.2
asn1crypto==1.5.1
astroid==2.15.8
asttokens==2.4.1
async-timeout==4.0.3
atlasclient==1.0.0
atlassian-python-api==3.41.3
attrs==23.1.0
aws-sam-translator==1.79.0
aws-xray-sdk==2.12.1
azure-batch==14.0.0
azure-common==1.1.28
azure-core==1.29.5
azure-cosmos==4.5.1
azure-datalake-store==0.0.53
azure-identity==1.15.0
azure-keyvault-secrets==4.7.0
azure-kusto-data==4.2.0
azure-mgmt-containerinstance==10.1.0
```

```
azure-mgmt-containerregistry==10.2.0
azure-mgmt-core==1.4.0
azure-mgmt-cosmosdb==9.3.0
azure-mgmt-datafactory==3.1.0
azure-mgmt-datalake-nspkg==3.0.1
azure-mgmt-datalake-store==0.5.0
azure-mgmt-nspkg==3.0.2
azure-mgmt-resource==23.0.1
azure-mgmt-storage==21.1.0
azure-nspkg==3.0.2
azure-servicebus==7.11.3
azure-storage-blob==12.18.3
azure-storage-common==2.1.0
azure-storage-file-datalake==12.13.2
azure-storage-file-share==12.14.2
azure-storage-file==2.1.0
azure-synapse-spark==0.7.0
backcall==0.2.0
backoff==1.10.0
backports.zoneinfo==0.2.1
bcrypt==4.0.1
beautifulsoup4==4.12.2
billiard==4.1.0
bitarray==2.8.2
black==23.10.1
blinker==1.6.3
boto3==1.28.64
botocore==1.31.64
cachelib==0.9.0
cachetools==5.3.2
cassandra-driver==3.28.0
cattrs==23.1.2
celery==5.3.4
certifi==2023.7.22
cffi==1.16.0
cfgv==3.4.0
cfn-lint==0.83.1
cgroupspy==0.2.2
chardet==5.2.0
charset-normalizer==3.3.2
checksumdir==1.2.0
ciso8601==2.3.1
click-didyoumean==0.3.0
click-plugins==1.1.1
click-repl==0.3.0
click==8.1.7
clickclick==20.10.2
cloudant==2.15.0
cloudpickle==2.2.1
colorama==0.4.6
colorlog==4.8.0
confluent-kafka==2.3.0
connexion==2.14.2
coverage==7.3.2
crcmod==1.7
cron-descriptor==1.4.0
croniter==2.0.1
cryptography==41.0.5
curlify==2.2.1
dask==2023.4.1
databricks-sql-connector==2.9.3
datadog==0.47.0
db-dtypes==1.1.1
decorator==5.1.1
```

```
defusedxml==0.7.1
deprecation==2.1.0
dill==0.3.1.1
distlib==0.3.7
distributed==2023.4.1
dnspython==2.4.2
docker==6.1.3
docopt==0.6.2
docutils==0.20.1
duckdb==0.9.1
ecdsa==0.18.0
elastic-transport==8.10.0
elasticsearch==8.10.1
email-validator==1.3.1
entrypoints==0.4
eralchemy2==1.3.8
et-xmlfile==1.1.0
eventlet==0.33.3
exceptiongroup==1.1.3
execnet==2.0.2
executing==2.0.1
facebook-business==18.0.3
fastavro==1.9.0
fasteners==0.19
fastjsonschema==2.18.1
filelock==3.13.1
flower==2.0.1
frozenlist==1.4.0
fsspec==2023.10.0
future==0.18.3
gcloud-aio-auth==4.2.3
gcloud-aio-bigquery==7.0.0
gcloud-aio-storage==9.0.0
gcsfs==2023.10.0
geomet==0.2.1.post1
gevent==23.9.1
gitdb==4.0.11
google-ads==22.1.0
google-api-core==2.12.0
google-api-python-client==2.106.0
google-auth-httplib2==0.1.1
google-auth-oauthlib==1.1.0
google-auth==2.23.4
google-cloud-aiplatform==1.36.0
google-cloud-appengine-logging==1.3.2
google-cloud-audit-log==0.2.5
google-cloud-automl==2.11.3
google-cloud-batch==0.17.2
google-cloud-bigquery-datatransfer==3.12.1
google-cloud-bigquery-storage==2.22.0
google-cloud-bigquery==3.13.0
google-cloud-bigtable==2.21.0
google-cloud-build==3.20.1
google-cloud-compute==1.14.1
google-cloud-container==2.32.0
google-cloud-core==2.3.3
google-cloud-datacatalog==3.16.0
google-cloud-dataflow-client==0.8.5
google-cloud-dataform==0.5.3
google-cloud-dataplex==1.7.0
google-cloud-dataproc-metastore==1.13.0
google-cloud-dataproc==5.6.0
google-cloud-dlp==3.12.3
google-cloud-kms==2.19.2
```

```
google-cloud-language==2.11.1
google-cloud-logging==3.8.0
google-cloud-memcache==1.7.3
google-cloud-monitoring==2.16.0
google-cloud-orchestration-airflow==1.9.2
google-cloud-os-login==2.11.0
google-cloud-pubsub==2.18.4
google-cloud-redis==2.13.2
google-cloud-resource-manager==1.10.4
google-cloud-run==0.10.0
google-cloud-secret-manager==2.16.4
google-cloud-spanner==3.40.1
google-cloud-speech==2.21.1
google-cloud-storage-transfer==1.9.2
google-cloud-storage==2.13.0
google-cloud-tasks==2.14.2
google-cloud-texttospeech==2.14.2
google-cloud-translate==3.12.1
google-cloud-videointelligence==2.11.4
google-cloud-vision==3.4.5
google-cloud-workflows==1.12.1
google-crc32c==1.5.0
google-re2==1.1
google-resumable-media==2.6.0
googleapis-common-protos==1.61.0
graphql-core==3.2.3
graphviz==0.20.1
greenlet==3.0.1
grpc-google-iam-v1==0.12.6
grpcio-gcp==0.2.2
grpcio-status==1.59.2
grpcio==1.59.2
gssapi==1.8.3
gunicorn==21.2.0
h11==0.14.0
hdfs==2.7.3
hmsclient==0.1.1
httpcore==0.16.3
httplib2==0.22.0
httpx==0.23.3
humanize==4.8.0
hvac==2.0.0
identify==2.5.31
idna==3.4
ijson==3.2.3
imagesize==1.4.1
importlib-metadata==6.8.0
importlib-resources==6.1.0
impyla==0.18.0
incremental==22.10.0
inflection==0.5.1
influxdb-client==1.38.0
iniconfig==2.0.0
ipdb==0.13.13
ipython==8.12.3
isodate==0.6.1
itsdangerous==2.1.2
jaraco.classes==3.3.0
jedi==0.19.1
jeepney==0.8.0
jmespath==0.10.0
jschema-to-python==1.2.3
json-merge-patch==0.2
jsondiff==2.0.0
```

```
jsonpatch==1.33
jsonpath-ng==1.6.0
jsonpickle==3.0.2
jsonpointer==2.4
jsonschema-path==0.3.1
jsonschema-specifications==2023.7.1
jsonschema==4.19.2
junit-xml==1.9
jupyter_client==8.5.0
jupyter_core==5.5.0
keyring==24.2.0
kombu==5.3.2
krb5==0.5.1
kubernetes-asyncio==24.2.3
kubernetes==23.6.0
kylinpy==2.8.4
lazy-object-proxy==1.9.0
ldap3==2.9.1
limits==3.6.0
linkify-it-py==2.0.2
locket==1.0.0
lockfile==0.12.2
looker-sdk==23.16.0
lxml==4.9.3
lz4==4.3.2
markdown-it-py==3.0.0
marshmallow-oneofschema==3.0.1
marshmallow-sqlalchemy==0.26.1
marshmallow==3.20.1
matplotlib-inline==0.1.6
mdit-py-plugins==0.4.0
mdurl==0.1.2
mongomock==4.1.2
monotonic==1.6
more-itertools==10.1.0
moto==4.2.7
mpmath==1.3.0
msal-extensions==1.0.0
msal==1.24.1
msgpack==1.0.7
msrest==0.7.1
msrestazure==0.6.4
multi-key-dict==2.0.3
multidict==6.0.4
mypy-boto3-appflow==1.28.42
mypy-boto3-rds==1.28.76
mypy-boto3-redshift-data==1.28.36
mypy-boto3-s3==1.28.55
mypy-extensions==1.0.0
mypy==1.2.0
mysql-connector-python==8.0.29
mysqlclient==2.2.0
nbclient==0.8.0
nbformat==5.9.2
neo4j==5.14.0
networkx==3.1
nh3==0.2.14
nodeenv==1.8.0
numpy==1.24.4
oauthlib==3.2.2
objsize==0.6.1
openapi-schema-validator==0.6.2
openapi-spec-validator==0.7.1
openlineage-integration-common==1.4.1
```

```
openlineage-python==1.4.1
openlineage_sql==1.4.1
openpyxl==3.1.2
opensearch-py==2.3.2
opentelemetry-api==1.20.0
opentelemetry-exporter-otlp-proto-common==1.20.0
opentelemetry-exporter-otlp-proto-grpc==1.20.0
opentelemetry-exporter-otlp-proto-http==1.20.0
opentelemetry-exporter-otlp==1.20.0
opentelemetry-exporter-prometheus==1.12.0rc1
opentelemetry-proto==1.20.0
opentelemetry-sdk==1.20.0
opentelemetry-semantic-conventions==0.41b0
opsgenie-sdk==2.1.5
oracledb==1.4.2
ordered-set==4.1.0
orjson==3.9.10
oscrypto==1.3.0
oss2==2.18.3
packaging==23.2
pandas-gbq==0.19.2
pandas==2.0.3
papermill==2.4.0
paramiko==3.3.1
parso==0.8.3
partd==1.4.1
pathable==0.4.3
pathspec==0.11.2
pbr==5.11.1
pdpyras==5.1.2
pendulum==2.1.2
pexpect==4.8.0
pickleshare==0.7.5
pinotdb==5.1.0
pipdeptree==2.13.0
pipx==1.2.1
pkginfo==1.9.6
pkgutil_resolve_name==1.3.10
platformdirs==3.11.0
pluggy==1.3.0
ply==3.11
plyvel==1.5.0
portalocker==2.8.2
pre-commit==3.5.0
presto-python-client==0.8.4
prison==0.2.1
prometheus-client==0.18.0
prompt-toolkit==3.0.39
proto-plus==1.22.3
protobuf==4.24.4
psutil==5.9.6
psycopg2-binary==2.9.9
ptyprocess==0.7.0
pure-eval==0.2.2
pure-sasl==0.6.2
py-partiql-parser==0.4.1
py4j==0.10.9.7
pyOpenSSL==23.3.0
pyarrow==11.0.0
pyasn1-modules==0.3.0
pyasn1==0.5.0
pycountry==22.3.5
pycparser==2.21
pycryptodome==3.19.0
```

```
pycryptodomex==3.19.0
pydantic==2.4.2
pydantic_core==2.10.1
pydata-google-auth==1.8.2
pydot==1.4.2
pydruid==0.6.5
pyenchant==3.2.2
pyexasol==0.25.2
pygraphviz==1.11
pyjsparser==2.7.1
pykerberos==1.2.4
pymongo==4.5.0
pymssql==2.2.10
pyodbc==5.0.1
pyparsing==3.1.1
pypsrp==0.8.1
pyspark==3.5.0
pyspnego==0.10.2
pytest-asyncio==0.21.1
pytest-cov==4.1.0
pytest-httpx==0.21.3
pytest-instafail==0.5.0
pytest-mock==3.12.0
pytest-rerunfailures==12.0
pytest-timeouts==1.2.1
pytest-xdist==3.3.1
pytest==7.4.3
python-arango==7.7.0
python-daemon==3.0.1
python-dateutil==2.8.2
python-dotenv==1.0.0
python-http-client==3.3.7
python-jenkins==1.7.0
python-jose==3.3.0
python-ldap==3.4.3
python-nvd3==0.15.0
python-slugify==8.0.1
python-telegram-bot==20.2
pytz==2023.3.post1
pytzdata==2020.1
pywinrm==0.4.3
pyzmq==25.1.1
reactivex==4.0.4
readme-renderer==42.0
redis==4.6.0
redshift-connector==2.0.915
referencing==0.30.2
regex==2023.10.3
requests-file==1.5.1
requests-kerberos==0.14.0
requests-mock==1.11.0
requests-ntlm==1.2.0
requests-oauthlib==1.3.1
requests-toolbelt==1.0.0
requests==2.31.0
responses==0.23.3
rfc3339-validator==0.1.4
rfc3986==1.5.0
rich-argparse==1.4.0
rich-click==1.7.1
rich==13.6.0
rpds-py==0.10.6
rsa==4.9
ruff==0.1.3
```

```
s3transfer==0.7.0
sarif-om==1.0.4
scramp==1.4.4
scrapbook==0.5.0
semver==3.0.2
sendgrid==6.10.0
sentinels==1.0.0
sentry-sdk==1.33.1
setproctitle==1.3.3
shapely==2.0.2
simple-salesforce==1.12.5
six==1.16.0
slack-sdk==3.23.0
smbprotocol==1.11.0
smmap==5.0.1
sniffio==1.3.0
snowballstemmer==2.2.0
snowflake-connector-python==3.3.1
snowflake-sqlalchemy==1.5.0
sortedcontainers==2.4.0
soupsieve==2.5
sphinx-airflow-theme==0.0.12
sphinx-argparse==0.4.0
sphinx-autoapi==2.1.1
sphinx-copybutton==0.5.2
sphinx-jinja==2.0.2
sphinx-rtd-theme==1.3.0
sphinxcontrib-applehelp==1.0.4
sphinxcontrib-devhelp==1.0.2
sphinxcontrib-htmlhelp==2.0.1
sphinxcontrib-httpdomain==1.8.1
sphinxcontrib-jquery==4.1
sphinxcontrib-jsmath==1.0.1
sphinxcontrib-qthelp==1.0.3
sphinxcontrib-redoc==1.6.0
sphinxcontrib-serializinghtml==1.1.5
sphinxcontrib-spelling==8.0.0
spython==0.3.1
sqlalchemy-bigquery==1.8.0
sqlalchemy-drill==1.1.4
sqlalchemy-redshift==0.8.14
sqlalchemy-spanner==1.6.2
sqlparse==0.4.4
sshpubkeys==3.3.1
sshtunnel==0.4.0
stack-data==0.6.3
starkbank-ecdsa==2.2.0
statsd==4.0.1
sympy==1.12
tableauserverclient==0.25
tabulate==0.9.0
tblib==3.0.0
tenacity==8.2.3
termcolor==2.3.0
text-unidecode==1.3
textwrap3==0.9.2
thrift-sasl==0.4.3
thrift==0.16.0
time-machine==2.13.0
tomli==2.0.1
tomlkit==0.12.1
toolz==0.12.0
tornado==6.3.3
towncrier==23.10.0
```

```
tqdm==4.66.1
traitlets==5.13.0
trino==0.327.0
twine==4.0.2
types-Deprecated==1.2.9.3
types-Markdown==3.5.0.0
types-PyMySQL==1.1.0.1
types-PyYAML==6.0.12.12
types-certifi==2021.10.8.3
types-croniter==2.0.0.0
types-docutils==0.20.0.3
types-paramiko==3.3.0.0
types-protobuf==4.24.0.4
types-pyOpenSSL==23.3.0.0
types-python-dateutil==2.8.19.14
types-python-slugify==8.0.0.3
types-pytz==2023.3.1.1
types-redis==4.6.0.8
types-requests==2.31.0.6
types-setuptools==68.2.0.0
types-tabulate==0.9.0.3
types-termcolor==1.1.6.2
types-toml==0.10.8.7
types-urllib3==1.26.25.14
typing_extensions==4.8.0
tzdata==2023.3
tzlocal==5.2
uc-micro-py==1.0.2
unicodecsv==0.14.1
uritemplate==4.1.1
urllib3==1.26.18
userpath==1.9.1
vertica-python==1.3.6
vine==5.0.0
virtualenv==20.24.6
watchtower==3.0.1
wcwidth==0.2.9
websocket-client==1.6.4
wrapt==1.15.0
xmltodict==0.13.0
yamllint==1.32.0
yarl==1.9.2
zeep==4.2.1
zenpy==2.0.41
zict==3.0.0
zipp==3.17.0
zope.event==5.0
zope.interface==6.1
zstandard==0.22.0
```

Related Information

In-place Upgrade with Airflow Operators and Libraries

Troubleshooting Python or Airflow compatibility issues

You can learn more about how to identify the compatibility issues related to Python or Airflow versions during Cloudera Data Engineering upgrade and troubleshoot them.

Python upgrade from version 3.8 to version 3.11

- If you are upgrading Python from version 3.8 to version 3.9, for more information, see:
  - What's New in Python 3.9
  - Removed in Python 3.9
  - Porting to Python 3.9
  - Check for DeprecationWarning
- If you are upgrading Python from version 3.9 to version 3.10, for more information, see:
  - What's New In Python 3.10
  - Deprecated in Python 3.10
  - Removed in Python 3.10
  - Porting to Python 3.10
  - Some of the important deprecations, removals, and restrictions in Python 3.10:
    - PEP 644 – Require OpenSSL 1.1.1 or newer
    - PEP 632 – Deprecate distutils module
    - PEP 623 – Remove wstr from Unicode
    - PEP 624 – Remove Py_UNICODE encoder APIs
    - PEP 597 – Add optional EncodingWarning
- If you are upgrading Python from version 3.10 to version 3.11, for more information, see:
  - What's New In Python 3.11
  - Deprecated in Python 3.11
  - Porting to Python 3.11
  - Some of the important deprecations, removals, and restrictions in Python 3.11:
    - PEP 594: Many legacy standard library modules have been deprecated. These will be removed in Python 3.13.
    - PEP 624: Py_UNICODE encoder APIs are removed

Tools for porting Python from version 3.8 to version 3.11

This list contains automated tools that allow you to check the Python source code and modernize it to the latest version if necessary:

- Python development mode: It can be enabled using the `-X dev` command line option or by setting the PYTHONDEVMODE environment variable to 1.
- DeprecationWarnings: To check for deprecation warnings in your code, see DeprecationWarnings.
- refurb: A tool for modernizing Python codebases. For more information, see refurb. For example,

```
refurb --enable-all --python-version 3.8 dag_file.py
```

- pyupgrade: A tool to automatically upgrade syntax for newer versions of Python. For more information, see pyupgrade. For example,

```
# create a temporary git repository for the DAG files and run pyupgrade on
 them
# NOTE: pyupgrade will rewrite the files automatically
pyupgrade --py311-plus *.py
# check the modifications done by pyupgrade before updating the DAG files
 in CDE
```

Airflow upgrade from version 2.7.3 to version 2.9.3

- Significant changes in Airflow version 2.8.0:
  - Raw HTML code in DAG docs and DAG params descriptions are disabled by default (#35460)
  - Account for change in UTC offset when calculating next schedule (#35887)
  - For the complete list of changes, see Airflow 2.8.0 documentation.
- Significant changes in Airflow version 2.8.1:
  - Target version for core dependency pendulum package set to 3 (#36281)
  - Airflow packaging specification follows modern Python packaging standards (#36537)
  - Graphviz dependency is now an optional one, not required one (#36647)
  - Do not let EventsTimetable schedule past events if catchup=False (#36134)
  - For the complete list of changes, see Airflow 2.8.1 documentation.
- Significant changes in Airflow version 2.8.2:
  - The allowed_deserialization_classes flag now follows a glob pattern (#36147)
  - The audit_logs permissions have been updated for heightened security (#37501)
  - AirflowTimeoutError is no longer except by default through Exception (#35653)
  - Revoking audit_log permission from all users except admin (#37501)
  - For the complete list of changes, see Airflow 2.8.2 documentation.
- Significant changes in Airflow version 2.8.3:
  - The smtp provider is now pre-installed when you install Airflow (#37713)
  - For the complete list of changes, see Airflow 2.8.3 documentation.
- Significant changes in Airflow version 2.8.4:
  - There are no significant changes in Airflow 2.8.4. For full list of changes, see Airflow 2.8.4 documentation.
- Significant changes in Airflow version 2.9.0:
  - Listener API methods are considered stable and can be used for production system (They were experimental features in older Airflow versions) (#36376)
  - Dataset URIs are now validated on input (#37005)
  - Add REST API actions to Audit Log events (#37734)
  - Prevent large string objects from being stored in the Rendered Template Fields (#38094)
  - The xcom change allows the storage of larger Xcom values in Airflow metadata 2^32 instead of 2^16 bytes. (#38401)
  - Stronger validation for key parameter defaults in taskflow context variables (#38015)
  - Deprecate smtp configs in airflow settings / local_settings (#37711)
  - For the complete list of changes, see Airflow 2.9.0 documentation.
- Significant changes in Airflow version 2.9.1:
  - Stackdriver logging bugfix requires Google provider 10.17.0 or later (#38071)
  - For the complete list of changes, see Airflow 2.9.1 documentation.
- Significant changes in Airflow version 2.9.2:
  - There are no significant changes in Airflow 2.9.2. For full list of changes, see Airflow 2.9.2 documentation.
- Significant changes in Airflow version 2.9.3:
  - Time unit for scheduled_duration and queued_duration changed (metrics) (#37936)
  - For the complete list of changes, see Airflow 2.9.3 documentation.

Debugging Airflow DAGs locally

You can learn about how to debug Airflow DAGs locally to identify any compatibility issues while upgrading Cloudera Data Engineering.

About this task

The example is based on the Airflow DAG debug documentation.

Procedure

1. Add the following two lines at the end of the DAG file that you want to test:

```
if __name__ == "__main__":
 dag.test()
```

For example,

```
from datetime import timezone
from dateutil import parser
from airflow.operators.bash_operator import BashOperator
from airflow.operators.python_operator import PythonOperator
from airflow import DAG
default_args = {
 'owner': 'test',
 'depends_on_past': False,
 'email': ['airflow@example.com'],
 'email_on_failure': False,
 'start_date':  parser.isoparse('2020-11-11T20:20:04.268Z').replace(tzinfo
=timezone.utc),
 'email_on_retry': False,
 'user': 'notused'
}

dag = DAG(
 'some-dag-3',
 default_args=default_args,
 catchup=False,
 schedule_interval=None,
 is_paused_upon_creation=False,
)

def print_function():
 print("Task executed.")

print_task = PythonOperator(
 task_id='print_task',
 dag=dag,
 python_callable=print_function
)

echo = BashOperator(
 task_id='run_this_first',
 bash_command='echo 1',
 dag=dag,
)

echo >> print_task

if __name__ == "__main__":
 dag.test()
```

2. Prepare a DAG testing environment by running the following commands:

```
python -m venv venv-airflow
source venv-airflow/bin/activate
```

```
 (venv-airflow) airflow pip install 'apache-airflow[http,celery,kubernetes
]==2.9.3' --constraint "https://raw.githubusercontent.com/apache/airflow/
constraints-2.9.3/constraints-3.11.txt"

# check the configured DB used for testing the DAGs (a local sqlite db sho
uld be enough)
airflow config get-value database sql_alchemy_conn
sqlite://///Users/$USER/airflow/airflow.db

# if there are DB errors missing tables or other reset the DB
airflow db reset
```

3. Run the DAG and check for the warnings. For example,

```
python -W default example_dag.py
/Users/$USER/workload/airflow/bashop_pythonop_dag.py:4 DeprecationWarning:
 The `airflow.operators.bash_operator.BashOperator` class is deprecated.
 Please use `'airflow.operators.bash.BashOperator'`.
/Users/$USER/workload/airflow/bashop_pythonop_dag.py:5 DeprecationWarning:
 The `airflow.operators.python_operator.PythonOperator` class is
 deprecated. Please use `'airflow.operators.python.PythonOperator'`.
/Users/$USER/workload/airflow/bashop_pythonop_dag.py:19
 RemovedInAirflow3Warning: Param `schedule_interval` is deprecated and
 will be removed in a future release. Please use `schedule` instead.
[2024-08-22T12:20:33.673+0200] {dag.py:4199} INFO - dagrun id: some-dag-3
[2024-08-22T12:20:33.678+0200] {dag.py:4215} INFO - created dagrun <DagRu
n some-dag-3@2024-08-2210:20:33.616669+00:00:manual__2024-08-22T10:20:33
.616669+00:00,state:ru
...
```

where the -W flag checks for DeprecationWarnings in the code. For more information, see Check for DeprecationWarning in your code.

Using the backup-restore-based upgrade script

Learn about how to use the backup-restore-based upgrade script in Cloudera Data Engineering Public Cloud.

About this task

This procedure applies to Cloudera Data Engineering versions 1.20.3-h2, 1.21.0-h2, 1.22.0, and higher.

If Apache Airflow connections and variables are involved, the original backup-restore-based upgrade is not applicable. This document describes a script-based fallback for performing the backup-restore-based upgrade that you can use when the default upgrade method described in Handling upgrade failures for Cloudera Data Engineering does not work.

Table 2: Contents included in the backup

Artifacts	Included in the automation script
Service config	Y
Service Logs	N
Virtual Cluster config	Y
Virtual Cluster end-points	N
Virtual Cluster event logs	N
Job: Spark	Y
Job: Airflow	Y

Resource: files	Y
Resource: docker runtimes	Y
Resource: python-venv (for Spark)	Y
Resource: python-venv (for Airflow)	Y (Since 1.21.0)
Git Repository	Y
Credentials	Y
Spark Session	N
Spark Session logs (Including statement history)	N
Job Runs	Y (Since 1.22.0)
Job Run logs (Driver, Executor, API)	N
Airflow DAG logs	N
Airflow connections	Y
Airflow variables	Y

Before you begin

Before running the script, you must install the listed tools:

- CDP CLI
- cdpcurl
- CDE CLI
- jq
- kubectl

1. Install CDP CLI. For more information, see Installing Cloudera client.
2. Install cdpcurl. For more information, see cdpcurl.
3. Download the CDE CLI from your Virtual Cluster page and add its path to the *PATH* environment variable with the export PATH="$PATH:[***CLI-PATH***]" command.
4. Install the other tools according to your system requirements.

Procedure

1. Configure the CDP CLI.
   a) On the Cloudera Management Console, navigate to your Profile page.
   b) On the Access Keys tab, click Generate Access Key.
   c) Download the credentials file and copy the contents to ~/.cdp/credentials

      The credentials are set as the default profile. You can also rename the profile. If you do so, set the CDP profile to your preference using one of the listed methods:

      - The CDP_PROFILE environment variable
      - The cde-service-backup-restore-utils.properties file
      - The --cdp-profile command line option

2. Configure the CDE CLI.

   Configure the CDE CLI following the instructions in Configuring the CLI client. Cloudera recommends that you use the Cloudera credentials, as they are needed for CDP CLI too.

   a) Set the CDE CLI using one of the listed methods:

      - The environment variable
      - The cde-service-backup-restore-utils.properties file
      - The ~/.cde/config.yaml file

      Note: Command line options are not available.

   b) Run the following command once to make sure that the configuration works:

      cde job list --vcluster-endpoint [***JOBS-API-URL***]

3. Back up the Cluster.

   a) Download the backup-restore-based upgrade script (cde-service-backup-restore-utils.sh) to your local machine: navigate to https://github.com/cloudera/cde-public/blob/master/scripts/cde-service-backup-restore-utils.sh and download the raw file using the download icon on the right.

   b) Authenticate to kubectl to access your Cluster.

   c) To back up the Cluster, run the following command.

      Note: Using --pre-check is optional, but recommended on the first attempt. With --pre-check, you can check the output of the tools in advance, to make sure the tools are properly installed and configured.

```
./cde-service-backup-restore-utils.sh backup-service -s <cluster-id> -b
<backup-base-directory> --pre-check
```

   Some of the contents are stored remotely and some are stored locally in the directory you specify.

   Note: -b indicates the backup base directory, which is optional. If not specified, the default $HOME/.cde/service-backups value is used. Backup files are saved under [***BASE-DIR***]/[***SERVICE-ID***]/, which means that the same directory is shared between multiple backups with the same base directory and service ID. If you want to preserve multiple backups of the same service, specify a different base directory. When you are sure that the backup is no longer needed, delete the backup files.

4. Restore the Cluster.

   a) Run the following command to provision the Cluster and Virtual Clusters. This can take around an hour.

```
./cde-service-backup-restore-utils.sh restore-service -s <original-clust
er-id> -t <new-cluster-id> -n <new-cluster-name> -b <backup-base-directo
ry> --service-only
```

   b) Authenticate again to kubectl to access your Cluster.

      If the version of the new service is different from the original one, download the CDE CLI tool of the new service and add its path to the *PATH* environment variable.

      Note:

      Ideally, all scheduled jobs are paused before taking the backup. If the scheduled jobs were not paused, add the --pause-jobs flag to the command to pause them on restore. If you add this flag, the script modifies the Airflow DAGs also to stop catch up.

      To restore your contents, run:

```
./cde-service-backup-restore-utils.sh restore-service -s <original-clust
er-id> -t <new-cluster-id> -n <new-cluster-name> -b <backup-base-directo
ry> --contents-only
```

Upgrading Airflow if DAGs and packages are compatible with new Airflow version

You can upgrade Airflow after ensuring the DAGs and the packages in the Airflow Custom Libraries and Operators are compatible with the new Airflow version by using the requirements.txt file.

Before you begin

Note:

To avoid issues after the upgrade, ensure that the DAGs and the packages in the Airflow Custom Libraries and Operators are compatible with the new Airflow version. Fixing the issues after the upgrade process is complete might cause further downtime in performing the job.

Ensure that the catchup option is not enabled for any user's Airflow jobs. Before the backup starts, if the Airflow DAG catchup options are enabled, disable them manually.

Procedure

1. Get the requirements.txt file for the current Airflow Custom Libraries and Operators.

   After the upgrade, you need the requirements.txt file to restore Python environments. For information on how to get the requirements.txt file, see *In-place upgrade with Airflow Operators and Libraries*.
2. Verify the requirements.txt file against an Airflow constraints file.

   For more information, see *In-place upgrade with Airflow Operators and Libraries*.

   a) If you run into errors during the verification, update the dependencies in the requirements.txt file.
3. Disable all Airflow Custom Libraries and Operators in all virtual clusters in the Cloudera Data Engineering service.
4. Perform the upgrade process.
5. Using the saved requirements.txt file from step 2 on page 38, build and activate the Airflow Python environment in the virtual clusters.

   If the build fails because of non-python related dependency issues, you may need to make version adjustments in the requirements.txt file.
6. On the Airflow UI and Cloudera Data Engineering jobs UI, verify that the Airflow jobs are running.

   In the case of DAG-related errors:

   - If the DAG parsing fails, look into the Airflow UI to identify the impacted DAGs.
   - If the DAG execution fails, look into the Airflow job logs to identify the impacted DAGs.
7. Fix the impacted DAGs or update the dependencies in the requirements.txt file.

Related Information

In-place Upgrade with Airflow Operators and Libraries

Handling upgrade failures for Cloudera Data Engineering

If your upgrade of Cloudera Data Engineering fails, you have the option to clone the service with the latest version of Cloudera Data Engineering. Learn how to handle an upgrade failure.

About this task

During a Cloudera Data Engineering upgrade, a backup is created as part of the upgrade preparation process. This procedure uses that backup to be restored in a new cluster.

The list of service backups is available in the Backup Library. To locate the Backup Library, in the left navigation menu of Cloudera Data Engineering select Administration, select Service Details, select the Maintenance tab, and select Backup Library.

To obtain the list of all available backups, in the CDP CLI, run:

```
cdp de list-backups
```

To obtain the list of service backups associated with a specific Cloudera environment, run "cdp de list-backups --filter "environment(eq)*[***CDP ENVIRONMENT NAME***]*"

The Cloudera Data Engineering backup includes the following:

- Cloudera Data Engineering Service configurations
- Virtual cluster names
- Virtual cluster configurations
- Virtual cluster file-based resources
- Spark job definitions
- Airflow job definitions
- Spark Python-env resources

The following are not yet included in the backup:

- Non file-based resources, for example, Python-venv resources and custom runtimes
- Airflow custom operators & libraries
- Logs
- Job run history
- Endpoints

Before you begin

Important: This procedure changes the virtual cluster endpoints.

1. Ensure that the catchup option is not enabled for any user's Airflow jobs.

   Before the backup starts, if the Airflow DAG catchup options are enabled, disable them manually.
2. By default, the restored service receives the name and ID of the original backed-up service. To ensure that the backup does not fail due to name and ID conflicts, perform either of these options:

   a. Delete the original service, which failed to upgrade during upgrading Cloudera Data Engineering.

   b. Rename the service and assign a new ID to it using the --service-id and --service-name options.

   Note:

   Assigning a new ID results in new Fully Qualified Domain Names (FQDNs) for the service and all its virtual clusters, which affects the endpoint stability.

   A valid service ID is:

   - An 8 character-long alphanumeric string
   - Does not contain vowels
   - Unique

Procedure

1. Restore the service from the backup.

```
cdp de restore-service --backup-id <backup-id> --environment-crn
```

```
    <environment-crn>
```

Where:

backup-id

The ID of the backup that you are restoring from.

environment-crn

The Customer Resource Number (CRN) of the Cloudera environment with which a restored Cloudera Data Engineering service is associated. Currently, you can restore the Cloudera Data Engineering service only to the same Cloudera environment to which the backed-up service is associated.

For example:

```
cdp de restore-service --backup-id 2 --environment-crn crn:cdp:environme
nts:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:environment:c67b9089-
2d3b-4579-861d-c0df12a105b1
```

2. To obtain a list of backups, run:

```
cdp de list-backups
```

3. To describe a particular backup, run:

```
cdp de describe-backup --backup-id <backup-id>
```

For example:

```
$ cdp de describe-backup --backup-id 2 --profile priv
{
    "backup": {
        "id": 2,
        "serviceID": "cluster-cf6h74lq",
        "serviceName": "dex-priv-default-azure-env-1689008683873",
        "environmentName": "dex-priv-default-azure-env",
        "environmentCrn": "crn:cdp:environments:us-west-1:9d74eee4-1cad-45
d7-b645-7ccf9edbb73d:environment:c67b9089-2d3b-4579-861d-c0df12a105b1",
        "creator": "crn:altus:iam:us-west-1:9d74eee4-1cad-45d7-b645-7ccf
9edbb73d:user:0f9a97a7-23a7-43bd-bc71-ecdb2aa34ed5",
        "cloudPlatform": "AZURE",
        "status": "completed",
        "created": "2023-07-17T18:02:58.385455Z"
    }
}
```

4. In the case of an Airflow DAG failure, identify the impacted DAG on the Airflow UI and fix it.

   For more information, see the DAG-related steps in *In-place upgrade with Airflow Operators and Libraries*.

Related Information

Upgrading Cloudera Data Engineering

In-place Upgrade with Airflow Operators and Libraries

Handling ineligible upgrades for Cloudera Data Engineering

In certain scenarios, your Cloudera Data Engineering service is not eligible for an in-place upgrade to the latest Cloudera Data Engineering version. In such a scenario, follow a manual upgrade process where administrators create a new service and import all jobs and resources. Learn how to conduct a workaround for each scenario.

Older Cloudera Data Engineering versions

If the version of your Cloudera Data Engineering service is older than six months, it is outside of the upgrade window. Older Cloudera Data Engineering versions do not support in-place upgrade and you cannot use the process listed in *Handling upgrade failures for Cloudera Data Engineering*. To check whether your Cloudera Data Engineering version is eligible for the in-place upgrade option, see Support lifecycle policy.

Cloudera Data Engineering provides the in-place upgrade option from the eligible versions to the latest version. If you are on a release that is ineligible for the in-place upgrade, the in-place upgrade option is disabled and you can perform a backup and restore. For more information, see *Cloudera Data Engineering upgrade version compatibility*.

Solution: A manual upgrade option is available. To upgrade, create a new Cloudera Data Engineering Service and transfer your jobs to the new service by following the instructions in *Backing up and restoring Cloudera Data Engineering jobs*.

Older Data Lake versions

If your Cloudera Data Engineering service uses an old Data Lake version, and therefore it is not compatible with the latest Cloudera Data Engineering version, your Cloudera Data Engineering service is not eligible to perform either an in-place upgrade or a manual upgrade. For information about Cloudera Data Engineering and its compatible Data Lake version, see *Compatibility for Cloudera Data Engineering and Runtime components*.

Solution: Upgrade your Data Lake to a compatible version associated with the Cloudera Data Engineering version. For more information about upgrading Data Lake, see *Upgrading a Data Lake*.

Airflow operators and libraries

If you use Airflow operators and libraries, before you can use the in-place upgrade, follow the procedure described in *In-place upgrade with Airflow Operators and Libraries*.

Solution: See *In-place upgrade with Airflow Operators and Libraries*.

Hotfix version of Cloudera Data Engineering

There is no upgrade available to Cloudera Data Engineering hotfix versions.

Related Information

Handling upgrade failures for Cloudera Data Engineering
Cloudera Data Engineering upgrade version compatibility
Backing up Cloudera Data Engineering jobs on remote storage
Compatibility for Cloudera Data Engineering and Runtime components
Upgrading a Data Lake
In-place Upgrade with Airflow Operators and Libraries

Cloudera Data Engineering 1.23.0

Using Cloudera Data Engineering resources

Date published: 2020-07-30
Date modified: 2024-11-12

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Python virtual environments with Cloudera Data Engineering.....4
Creating a Python virtual environment resource..... 4
Associating a Python virtual environment with a Cloudera Data Engineering job..... 6
Associating a Python virtual environment with a Cloudera Data Engineering Session..... 7
Updating Python virtual environment resources.....7

Using Python virtual environments with Cloudera Data Engineering

Cloudera Data Engineering supports Python virtual environments to manage job dependencies by using the python-e nv *resource* type.

A *resource* in Cloudera Data Engineering is a named collection of files or other resources referenced by a job. The python-env resource type allows you to specify a requirements.txt file that defines a virtual environment that you can then associate with a Cloudera Data Engineering job. You can specify any Python package in the requirements.txt file, including those with C dependencies.

Creating a Python virtual environment resource

After you have created the requirements.txt file, you can create the Python virtual environment resource.

Note:

- For python-env resources, you can only upload a requirements.txt file. Python environment resources do not support arbitrary file upload. If the local file is named something other than requirements.txt, you must add the flag --resource-path requirements.txt to the command, which renames the file to requirements.txt in the resource.

  You can also specify a PyPi mirror for a Python virtual environment resource using the --pypi-mirror flag. This requires network access to the mirror from the Cloudera environment.

- If a Python package specified in the requirements.txt file is not found or does not support the underlying OS architecture amd64/arm64, then the Python virtual environment build fails.

For CDE CLI

Before you begin

- Download and configure the CDE CLI.
- Create a requirements.txt file specifying the Python package and version dependencies required by your Cloudera Data Engineering job.
- Ensure that the following hostnames are reacheable from within the cluster, to install the Python package successfully if no PyPi mirror is configured:
  - pypi.python.org
  - pypi.org
  - pythonhosted.org
  - files.pythonhosted.org

Steps

1. Run the cde resource create command as follows to create a Python virtual environment resource.

   Note: You can also specify a PyPi mirror for a Python virtual environment resource using the --pypi-mirror flag. This requires network access to the mirror from the CDP environment.

   ```
   cde resource create --name cde-python-env-resource --type python-env --p
   ython-version python3
   ```

2. Upload the requirements.txt file to the resource.

   Note: For python-env resources, you can only upload a requirements.txt file. Python environment resources do not support arbitrary file upload. If the local file is named something other than requirements.txt, you must add the flag --resource-path requirements.txt to the command, which renames the file to requirements.txt in the resource.

```
cde resource upload --name cde-python-env-resource --local-path ${HOME}/
requirements.txt
```

Result

When you first create a Python virtual environment resource, Cloudera Data Engineering builds the environment according to the requirements.txt file. During this build time, you cannot run a job associated with the virtual environment. You can check the status of the environment by running cde resource list-events --name <*RESOURCE_NAME*>. For example:

```
cde resource list-events --name cde-python-env-resource
```

The environment is ready when you see a message similar to the following:

```
  {
    "id": 4,
    "message": "Job pp-84kgdgf6-resource-builder-cde-python-env-resourc
e-1634911572 succeeded, marking resource with ready status",
    "created": "2021-10-22T14:09:13Z"
  }
```

For Web UI

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

- Create a requirements.txt file specifying the Python package and version dependencies required by your Cloudera Data Engineering job.
- Ensure that the following hostnames are reacheable from within the cluster, to install the Python package successfully if no PyPi mirror is configured:
  - pypi.python.org
  - pypi.org
  - pythonhosted.org
  - files.pythonhosted.org

Steps

1. In the Cloudera management console, click the Data Engineering tile and click Overview.
2. In the CDE Services column, select the service containing the virtual cluster where you want to create the Python virtual environment.
3. In the Virtual Clusters column on the right, click the View Jobs icon for the virtual cluster where you want to create the Python virtual environment.
4. Click Resources in the left menu.
5. Click Create Resource at the top right.
6. Specify a resource name, and then select Python Environment from the Type drop-down menu.
7. Choose the Python version for the environment and optionally specify the PyPi Mirror URL. The PyPi mirror must be accessible from the Cloudera environment.

8. Click Create.
9. Click Upload File and select the requirements.txt file from your local machine. You can also drag-and-drop the file to the outlined area on the page.

Result

The UI displays Building the resource... while the Python virtual environment is building. After the environment is built, the page displays the Python packages and versions included in the environment.

Associating a Python virtual environment with a Cloudera Data Engineering job

You can associate the Python virtual environment with a Cloudera Data Engineering job at the time of creation, or you can update an existing job.

For CDE CLI

Before you begin

- Download and configure the CDE CLI.
- Create a Python virtual environment Cloudera Data Engineering resource.
- Create a Cloudera Data Engineering job.

Steps

1. Using the CDE CLI, run the cde job update command to associate a Python virtual environment with the job.

```
cde job update --name pyspark-example --python-env-resource-name cde-pyt
hon-env-resource
```

Note: You can specify a Python virtual environment resource at job creation time as well, using the flag --python-env-resource-name. For example:

```
cde job create --type spark --application-file pyspark-example.p
y --python-env-resource-name cde-python-env-resource --name pysp
ark-example
```

For Web UI

Before you begin

Important: The user interface for Cloudera Data Engineering 1.17 and above has been updated. The left-hand menu was updated to provide easy access to commonly used pages. The steps below will vary slightly, for example, the Overview page has been replaced with the Home page. You can also manage a job by clicking Jobs on the left-hand menu, then selecting your desired Virtual Cluster from a drop-down at the top of the Jobs page. The new home page still displays Virtual Clusters, but now includes quick-access links located at the top for the following categories: Jobs, Resources, and Download & Docs.

- Create a Python virtual environment Cloudera Data Engineering resource.
- Create a Cloudera Data Engineering job.

Steps

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Job Runs on the left navigation menu. The Job Runs page displays.
3. Using the dropdown menu, select the virtual cluster containing the application you want to manage.
4. Click the name of the job you want to modify.
5. Go to the Configuration tab.
6. Click Edit.

7. In the Python Environment section, click Select Python Environment.
8. Select the Python virtual environment resource you want to use, and then click Select Resource.
9. At the bottom of the page, click Update and Run to run the job immediately, or click the drop-down arrow on the button and select Update to update the job without running it.

Associating a Python virtual environment with a Cloudera Data Engineering Session

You can associate the Python virtual environment with a Cloudera Data Engineering Session at the time of creation.

For CDE CLI

Before you begin

- Download and configure the CDE CLI.
- Create a Python virtual environment Cloudera Data Engineering resource.

Steps

1. Using the CDE CLI, run the cde session create command to associate a Python virtual environment with the session.

```
cde session create --name python-env-example  --python-env-resource-name
 cde-python-env-resource
```

For Web UI

Before you begin

- Create a Python virtual environment Cloudera Data Engineering resource.
- Create a Cloudera Data Engineering Session.

Steps

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Sessions on the left navigation menu.
3. Using the dropdown menu, select the virtual cluster containing the application you want to manage.
4. Click Create.

   Note: The Python environment can only be specified when you create a Session.

5. In the Python Environment section, click Select Python Environment.
6. Select the Python virtual environment resource that you want to use, and then click Select Resource.
7. Set additional configurations as needed.
8. Click Create.

Updating Python virtual environment resources

Currently, Python virtual environments cannot be updated. Instead, create a new Python virtual environment resource and update the job to reference the new resource.

For CDE CLI

Before you begin

- Download and configure the CDE CLI.

- Create a new requirements.txt file specifying the Python package and version dependencies required by your Cloudera Data Engineering job.

Steps

1. Create a new python-env resource.

```
cde resource create --name new-cde-python-env-resource --type python-env
 --python-version python3
```

2. Upload the new or updated requirements.txt file to the new resource.

```
cde resource upload --name new-cde-python-env-resource --local-path ${HO
ME}/requirements.txt
```

3. Update the Cloudera Data Engineering job to specify the new resource.

```
cde job update --name pyspark-example --python-env-resource-name new-cde
-python-env-resource
```

For Web UI

Before you begin

- Create a new requirements.txt file specifying the Python package and version dependencies required by your Cloudera Data Engineering job.

Steps

1. Create a new Python virtual environment using the new requirements.txt file, following the instructions in Creating a Python virtual environment resource .
2. Update the Cloudera Data Engineering job to reference the new Python virtual environment, following the instructions in Associating a Python virtual environment with a Cloudera Data Engineering job .

Related Information

Creating a Python virtual environment resource

Associating a Python virtual environment with a Cloudera Data Engineering job

Cloudera DataFlow

Auto-scaling

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Auto-scaling flow deployments....4
CPU based auto-scaling....4
Configure CPU based auto-scaling....4
Flow metrics based auto-scaling....5
Enable Flow metrics based auto-scaling....6

Auto-scaling flow deployments

Flow deployments can be configured to automatically scale up or down the number of NiFi nodes depending on the resource utilization in the cluster.

With Auto-scaling, you get the following benefits:

- You can choose a minimum and maximum number of nodes to ensure that the required resources are available while controlling cost.
- Flow deployments support CPU based auto-scaling as well as flow metrics based auto-scaling.
- When a flow deployment scales down, the affected NiFi nodes are being offloaded before they are removed from the cluster to ensure that no data is lost.
- After new nodes have been added to the cluster, new data is automatically distributed to all cluster nodes including the recently provisioned ones.

CPU based auto-scaling

When you turn this feature on, Cloudera DataFlow uses the Kubernetes Horizontal Pod Autoscaler (HPA) to increase or decrease the number of NiFi pods in a deployment based on resource utilization.

HPA monitors CPU utilization across all NiFi pods of a flow deployment. When CPU utilization reaches 75%, a new NiFi pod gets created and is added to the NiFi cluster. HPA continues to add additional NiFi pods until the aggregated CPU utilization for a deployment becomes stable around the 75% threshold. Once the new NiFi pod has joined the NiFi cluster, it starts receiving and processing data.

Note: Data that has already been received by the NiFi cluster and is being processed is not redistributed to new NiFi pods.

A similar logic is applied to automatically scale down and decrease the number of NiFi pods. HPA monitors CPU utilization and when it falls below 75% for five minutes it starts removing NiFi pods. Before the NiFi pod is terminated, Cloudera DataFlow offloads all existing data from the NiFi pod that has been selected for termination. Offloading data distributes data that is currently being processed by the affected node to the remaining nodes in the cluster. This offloading procedure ensures that NiFi pods are terminated gracefully and avoids data loss.

Use cases

CPU based auto-scaling is great for data flows that require CPU intensive transformations like compressing or decompressing data and have to deal with bursts of data every now and then.

Configure CPU based auto-scaling

For UI

Before you begin

When deploying a flow definition, continue to Step 4: Sizing & Scaling and pick your NiFi Node size

Steps

1. Activate the Auto Scaling toggle to turn on CPU based auto-scaling.

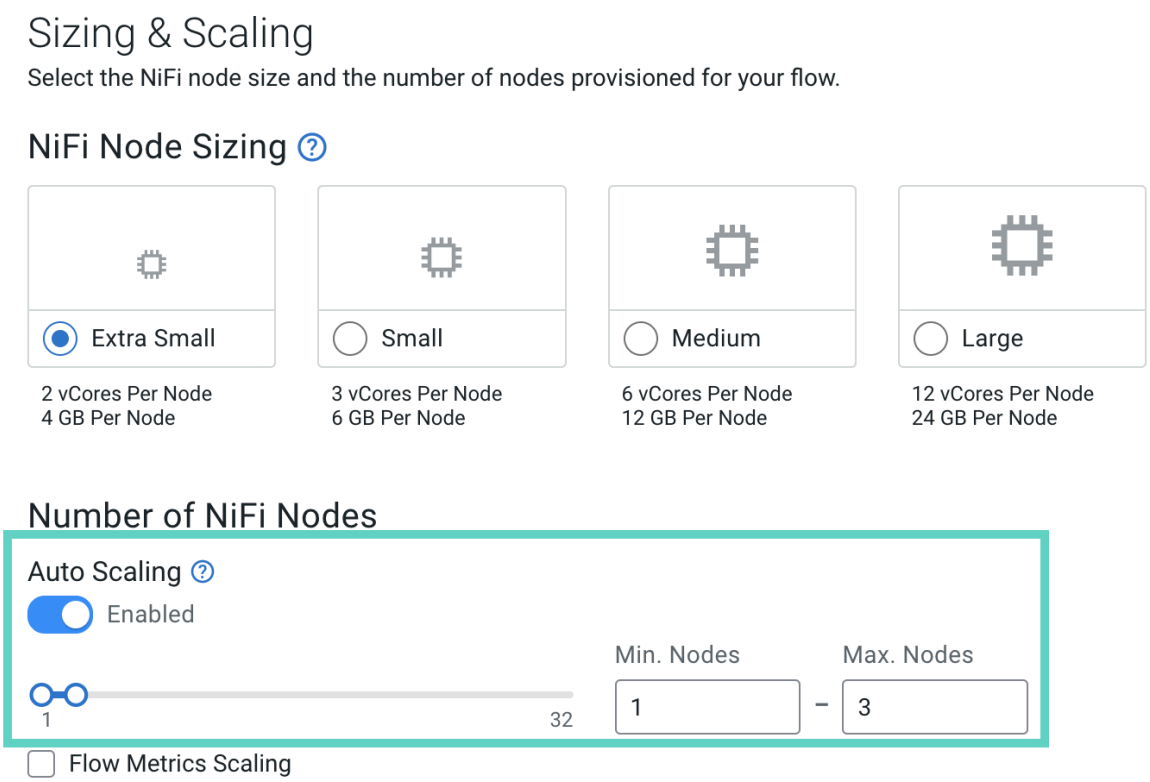

2. Use the slider or text fields to adjust the minimum and maximum number of nodes. Cloudera DataFlow automatically determines the best number of nodes within the specified boundaries.

For CLI

To configure auto-scaling using the CDP CLI, execute the `cdp df create-deployment` command with the following configuration:

```
cdp df create-deployment \
  --service-crn <your DFService CRN> \
  --flow-version-crn "<your flow version CRN>" \
  --deployment-name "<your flow deployment name>" \
  --cfm-nifi-version <your desired NiFi version> \
  --auto-start-flow \
  --cluster-size-name <your desired NiFi Node Size> \
  --auto-scaling-enabled \
  --auto-scale-min-nodes [***MINIMUM NUMBER OF NODES***] \
  --auto-scale-max-nodes [***MAXIMUM NUMBER OF NODES***]
```

Replace *[***MINIMUM NUMBER OF NODES***]* and *[***MAXIMUM NUMBER OF NODES***]* with the minimum and maximum number of nodes you want to allow for this deployment, respectively.

Flow metrics based auto-scaling

Flow metrics based auto-scaling predicts future load on a connection to trigger scaling decisions.

In contrast to CPU based auto-scaling where scaling decisions are made based on infrastructure utilization, flow metrics based auto-scaling works by predicting backpressure on a source connection queue. Source connections are connections attached to processors where data is first introduced into the flow. Backpressure occurs when such

a connection becomes 100% full. Source connections are automatically detected through static analysis of the flow definition. Queue percentage of each source connection is tracked and a linear regression is performed on the histogram for each to predict the fullness of the queue in 20 minutes. Scaling up occurs when that prediction is greater than 80% for 5 minutes.

Flow metrics based auto-scaling can be used to detect the need to scale based on flow performance metrics. In cases where queues fill up without CPU utilization going over the threshold, flow metrics based auto-scaling still triggers a scaling operation.

Enable Flow metrics based auto-scaling

Flow metrics based scaling is an optional setting which can be enabled in addition to CPU based auto-scaling.

For UI

Before you begin

When deploying a flow definition, continue to Step 4: Sizing & Scaling and pick your NiFi Node size

Steps

1. Activate the Auto Scaling toggle to turn on CPU based auto-scaling.

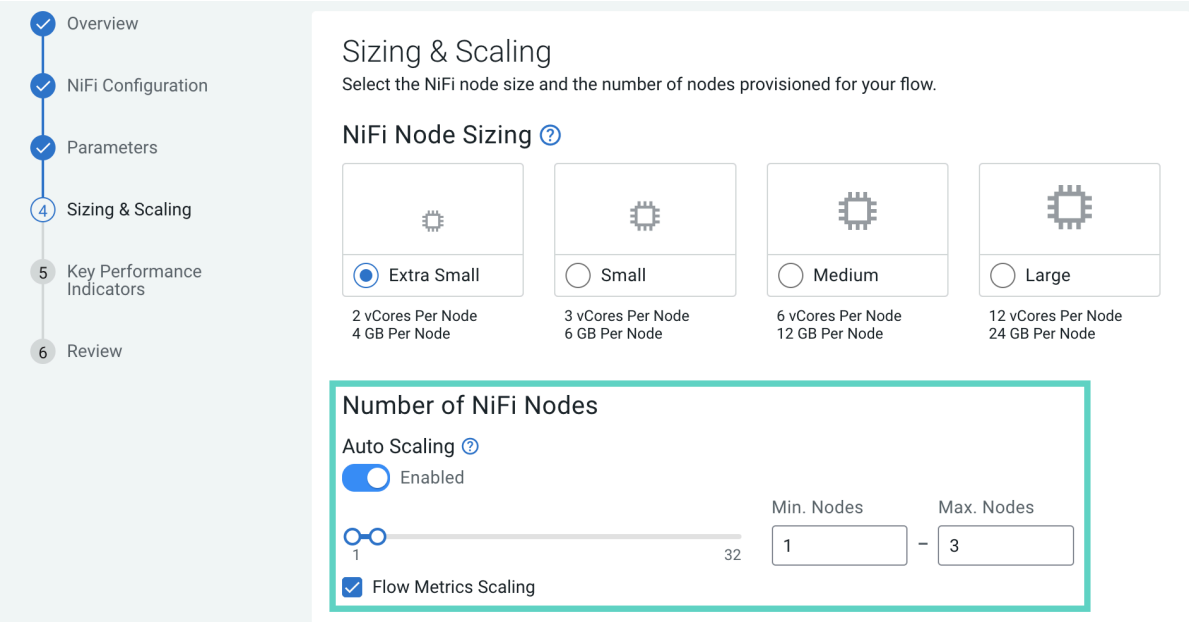

2. Use the slider or text fields to adjust the minimum and maximum number of nodes. Cloudera DataFlow automatically determines the best number of nodes within the specified boundaries.
3. Select Flow Metrics Scaling to enable flow metrics based auto-scaling for your deployment.

For CLI

To configure auto-scaling using the CDP CLI, execute the `cdp df create-deployment` command with the following configuration:

```
cdp df create-deployment \
  --service-crn <your DFService CRN> \
  --flow-version-crn "<your flow version CRN>" \
  --deployment-name "<your flow deployment name>" \
  --cfm-nifi-version <your desired NiFi version> \
  --auto-start-flow \
  --cluster-size-name <your desired NiFi Node Size> \
  --auto-scaling-enabled \
```

```
  --auto-scale-min-nodes 1 \
  --auto-scale-max-nodes 3
  --flow-metrics-scaling-enabled
```

Cloudera DataFlow

Cloudera DataFlow Functions in AWS Lambda

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a Lambda function 5

Configuring your Lambda function 7

Output ports 12

Parameters 13

Cold start 15

Cloud storage 16

Configuring Cloudera VPC to deploy Lambda Functions 18

File system for content repository 20

Output size constraints 21

DataFlow state 21

Configuring Kerberos 21

Handling failures 22

Testing your Lambda function 22

Publishing your Lambda function 23

Monitoring and logs 24

Adjusting logs levels 25

AWS Lambda triggers..... 26

Creating a Lambda function using CLI.....36

Creating a Lambda function

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers.

About this task

Follow these steps to create an AWS Lambda function that is able to run Cloudera DataFlow Functions:

Procedure

1. Navigate to the AWS Lambda page in your cloud account in the region of your choice.

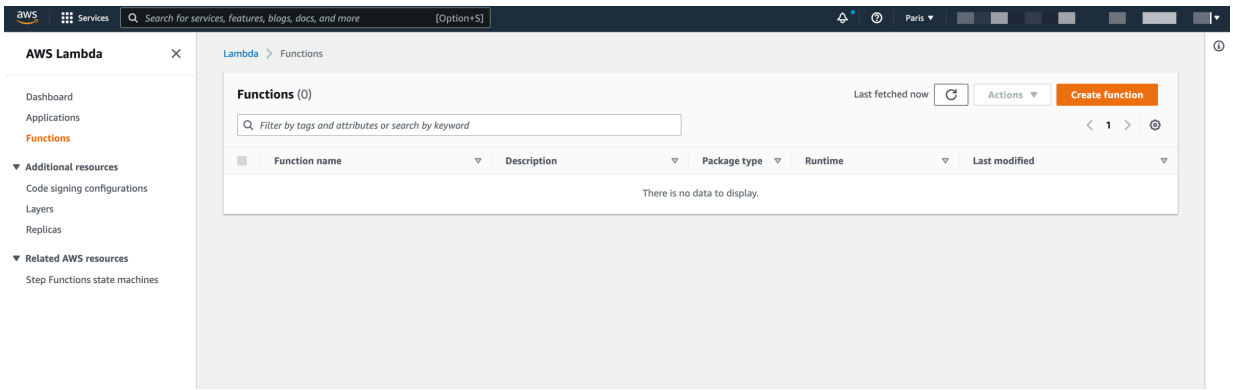

2. Click Create function.

   The Create function page opens.

3. Fill in the Basic information section:
   a) Keep the default Author from scratch option selected.
   b) Give your function a meaningful name for your use case.
   c) Select Java 8 on Amazon Linux 2 for Runtime.
   d) Choose x86_64 for Architecture.
   e) Click Create function.

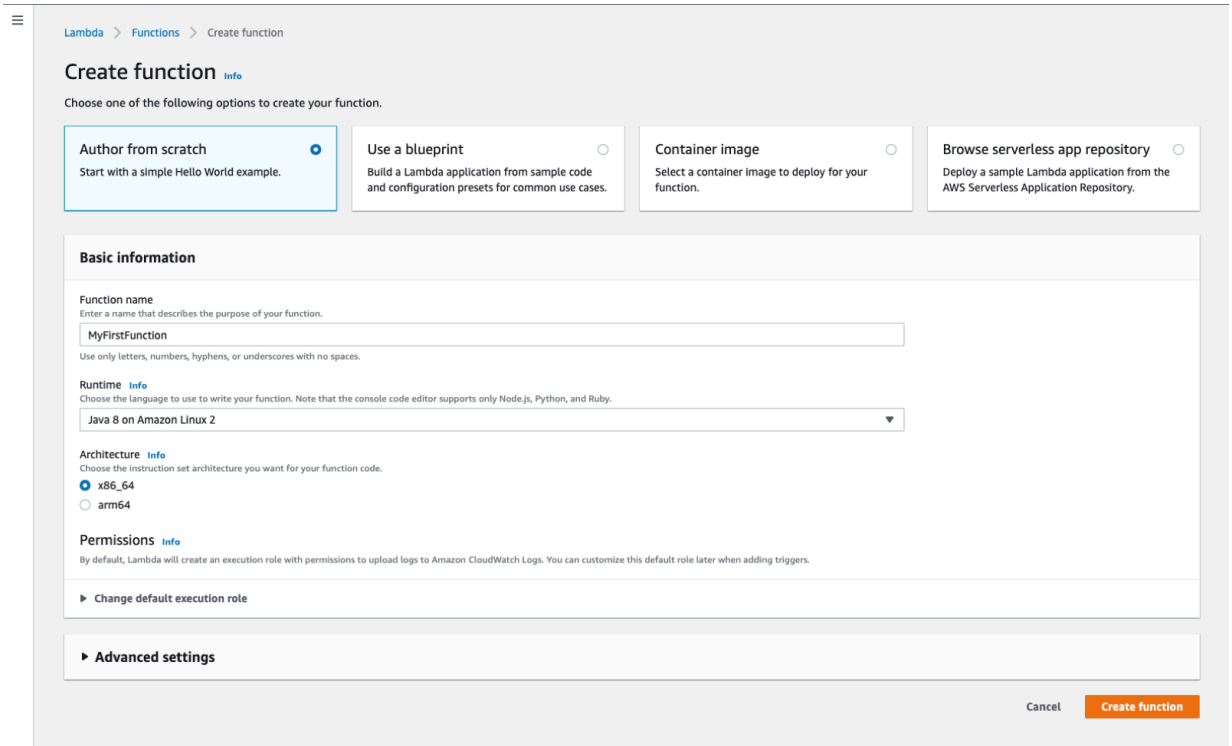

Your function is created and its details are displayed.

4. In the Code source section of the Code tab, click Upload from Amazon S3 location .

   The Upload a file from Amazon S3 modal window appears.

5. Provide the S3 URL of Cloudera DataFlow Function binaries in the textbox.

   Note: You have downloaded the Cloudera DataFlow Function handler libraries as a zip of binaries from Cloudera DataFlow and uploaded them to an S3 bucket when getting ready to run functions. AWS Lambda will use these binaries to run the NiFi flow. For instructions, see *Downloading Lambda Cloudera DataFlow Function binaries and uploading to S3*.

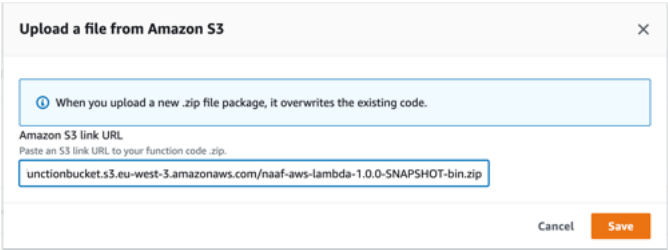

6. Click Save.

7. Navigate to the Runtime Settings on the Code tab and click Edit to change the Handler property to: com.cloudera .naaf.aws.lambda.StatelessNiFiFunctionHandler::handleRequest

   The details of your first function:

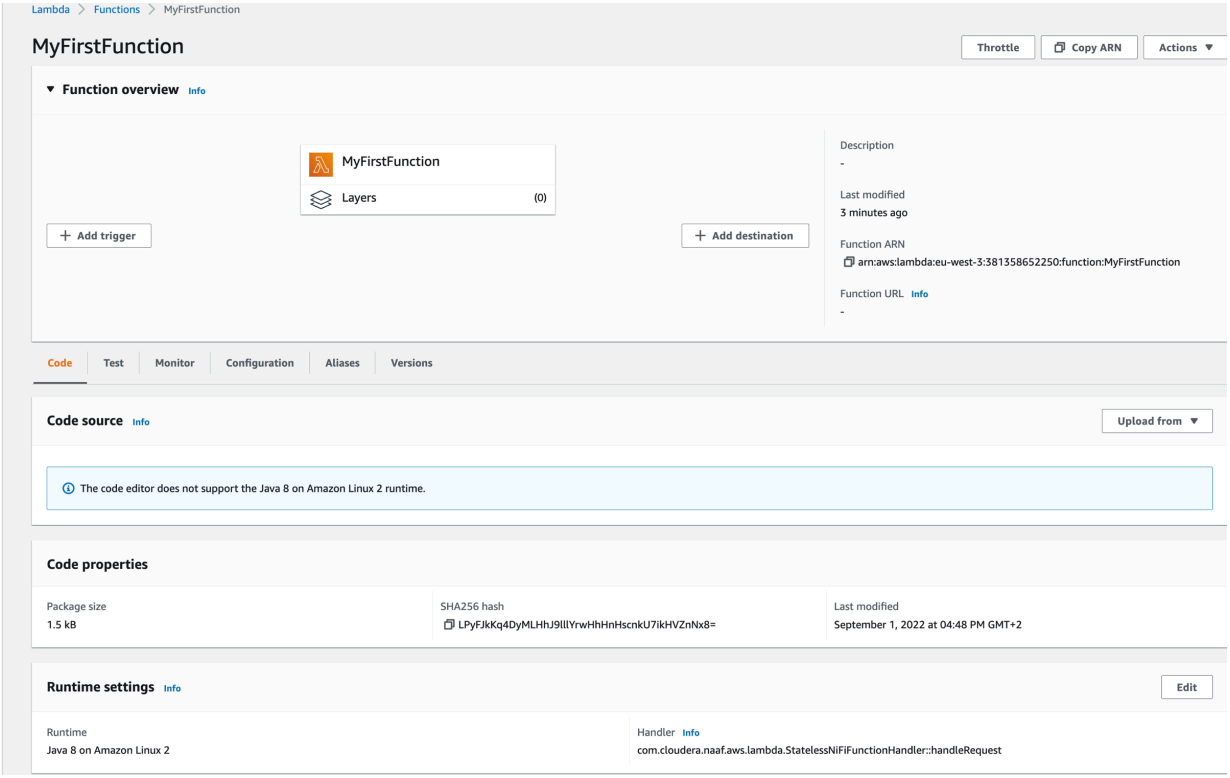

What to do next

You can now start configuring your Lambda specifically for your flow and for your use case.

Related Information

Downloading Lambda Cloudera DataFlowFunction binaries and uploading to S3

Configuring your Lambda function

After you create a function, you can use the built-in configuration options to control its behavior. You can configure additional capabilities, adjust resources associated with your function, such as memory and timeout, or you can also create and edit test events to test your function using the console.

Memory and timeout in runtime configuration

You can configure two very important elements, memory and timeout on the Configuration General configuration tab of the Lambda function details screen.

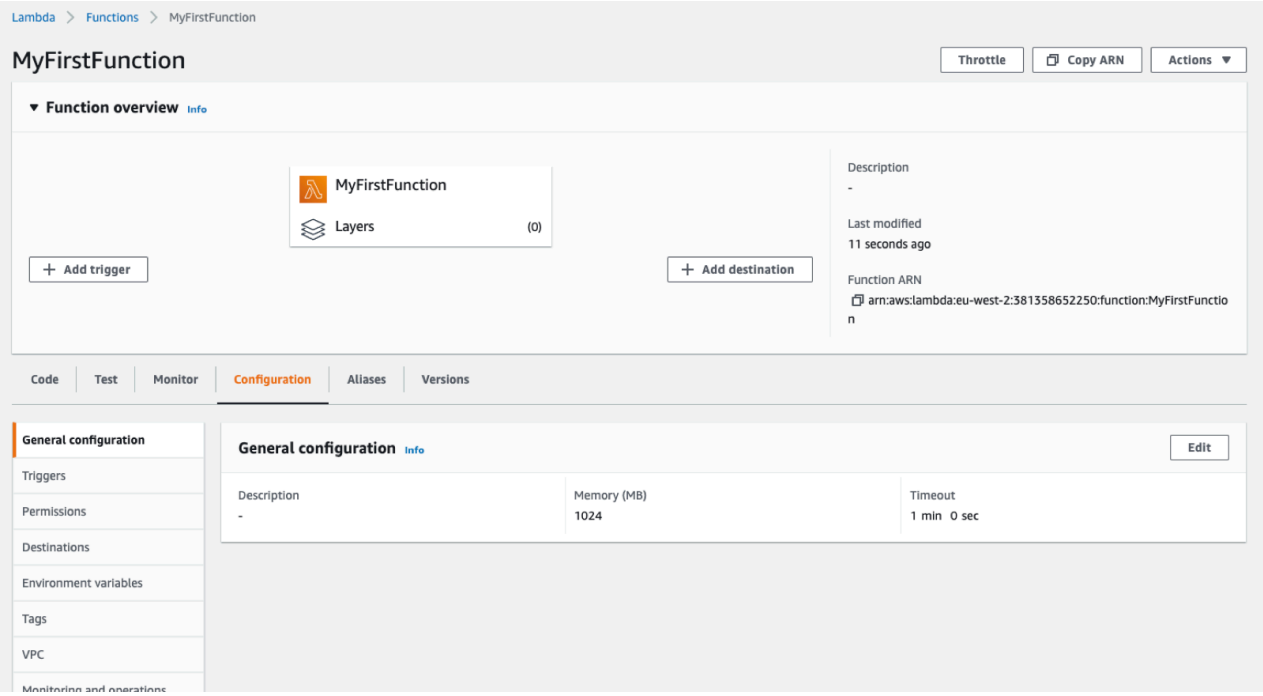

The appropriate values for these two settings depend on the data flow to be deployed. Typically, it is recommended to start with 1536-2048 MB (1.5 - 2 GB) memory allocated to Lambda. This amount of memory may be more than necessary or less than required. The default value of 512 MB may be perfectly fine for many use cases. You can adjust this value later as you see how much memory your function needs.

While Cloudera DataFlow Functions tend to perform very well during a warm start, a cold start that must source extensions (processors, controller services, and so on) may take several seconds to initialize. So it is recommended to set the Timeout to at least 30 seconds. If the data flow to run reaches out to many other services or performs complex computation, it may be necessary to use a larger value, even several minutes.

Click Edit if you want to change the default values.

See the corresponding AWS documentation to learn more about timeout duration and memory sizing, which determines the amount of resources provisioned for your function.

Runtime environment variables

You must configure the function to specify which data flow to run and add any necessary runtime configuration using environment variables.

Procedure

1. Navigate to the Configuration tab and select Environment variables from the menu on the left.
2. Click Edit on the Environment variables pane.

3. Define your environment variables as key-value pairs.

   You can add one or more environment variables to configure the function. The following environment variables are supported:

Variable Name	Description	Required	Default Value
FLOW_CRN	The Cloudera Resource Name (CRN) for the data flow that is to be run. The data flow must be stored in the Cloudera DataFlow Catalog. This CRN should indicate the specific version of the data flow and as such will end with a suffix like /v.1. For more information, see *Retrieving data flow CRN*.	true	--
DF_PRIVATE_KEY	The Private Key for accessing the Cloudera DataFlow service. The Private Key and Access Key are used to authenticate with the Cloudera DataFlow Service and they must provide the necessary authorizations to access the specified data flow. For more information, see *Provisioning Access Key ID and Private Key*.	true	--
DF_ACCESS_KEY	The Access Key for accessing the Cloudera DataFlow service. The Private Key and Access Key are used to authenticate with the Cloudera DataFlow Service and they must provide the necessary authorizations to access the specified data flow. For more information, see *Provisioning Access Key ID and Private Key*.	true	--
INPUT_PORT	The name of the Input Port to use. If the specified data flow has more than one Input Port at the root group level, this environment variable must be specified, indicating the name of the Input Port to queue up the AWS Lambda notification. If there is only one Input Port, this variable is not required. If it is specified, it must properly match the name of the Input Port.	false	--

Variable Name	Description	Required	Default Value
OUTPUT_PORT	The name of the Output Port to retrieve the results from. If no Output Port exists, the variable does not need to be specified and no data will be returned. If at least one Output Port exists in the data flow, this variable can be used to determine the name of the Output Port whose data will be sent along as the output of the function. For more information on how the appropriate Output Port is determined, see *Output ports*.	false	--
FAILURE_PORTS	A comma-separated list of Output Ports that exist at the root group level of the data flow. If any FlowFile is sent to one of these Output Ports, the function invocation is considered a failure. For more information, see *Output ports*.	false	--
DEFAULT_PARAM_CONTEXT	If the data flow uses a Parameter Context, the Lambda function will look for a Secret in the AWS Secrets Manager with the same name. If no Secret can be found with that name, this variable specifies the name of the Secret to default to. For more information, see *Parameters*.	false	--
PARAM_CONTEXT_*	If the data flow makes use of Parameter Contexts, this environment variable provides a mechanism for mapping a Parameter Context to one or more alternate Secrets in the AWS Secrets Manager, in a comma-separated list. For more information, see *Parameters*.	false	--
CONTENT_REPO	The contents of the FlowFiles can be stored either in memory, on the JVM heap, or on disk. If this environment variable is set, it specifies the path to a directory where the content should be stored. If it is not specified, the content is held in memory. For more information, see *File system for content repository*.	false	--

Variable Name	Description	Required	Default Value
EXTENSIONS_*	The directory to look for custom extensions / NiFi Archives (NARs). For more information, see *Providing custom extensions / NARs*.	false	--
DF_SERVICE_URL	The Base URL for the Cloudera DataFlow Service.	false	https://api.us-west-1.cdp.cloudera.com/
NEXUS_URL	The Base URL for a Nexus Repository for downloading any NiFi Archives (NARs) needed for running the data flow.	false	https://repository.cloudera.com/artifactory/cloudera-repos/
STORAGE_BUCKET	An S3 bucket in which to look for custom extensions / NiFi Archives (NARs) and resources. For more information, see *S3 Bucket storage*.	false	--
STORAGE_EXTENSIONS_DIRECTORY	The directory in the S3 bucket to look for custom extensions / NiFi Archives (NARs). For more information, see *S3 Bucket storage*.	false	extensions
STORAGE_RESOURCES_DIRECTORY	The directory in the S3 bucket to look for custom resources. For more information, see *Providing additional resources*.	false	resources
WORKING_DIR	The working directory, where NAR files will be expanded.	false	/tmp/working
EXTENSIONS_DOWNLOAD_DIR	The directory to which missing extensions / NiFi Archives (NARs) will be downloaded. For more information, see *Providing custom extensions / NARs*.	false	/tmp/extensions
EXTENSIONS_DIR_*	It specifies read-only directories that may contain custom extensions / NiFi Archives (NARs). For more information, see *Providing custom extensions / NARs*.	false	--
DISABLE_STATE_PROVIDER	If true, it disables the DynamoDB data flow state provider, even if the data flow has stateful processors.	false	--
DYNAMODB_STATE_TABLE	The DynamoDB table name where the state will be stored	false	nifi_state
DYNAMODB_BILLING_MODE	It sets the DynamoDB Billing Mode (PROVISIONED or PAY_PER_REQUEST).	false	PAY_PER_REQUEST

Variable Name	Description	Required	Default Value
DYNAMODB_WRITE_CAPACITY_UNITS	It sets the DynamoDB Write Capacity Units, when using PROVISIONED Billing Mode.	true if using PROVISIONED Billing Mode	--
DYNAMODB_READ_CAPACITY_UNITS	It sets the DynamoDB Read Capacity Units, when using PROVISIONED Billing Mode.	true if using PROVISIONED Billing Mode	--
KRB5_FILE	It specifies the filename of the krb5.conf file. This is necessary only if connecting to a Kerberos-protected endpoint. For more information, see *Configuring Kerberos*.	false	/etc/krb5.conf

4. When ready, click Save.

Related Information

Retrieving data flow CRN
Provisioning Access Key ID and Private Key
Output ports
Parameters
File system for content repository
Custom NARs in AWS
S3 Bucket storage
Configuring Kerberos

Output ports

Depending on what kind of data flow you want to run, you may or may not want to output the data.

For example, a data flow may be designed to be called in response to an object being added to S3. This data flow may fetch the data from S3, parse the data, transform it, and then push the data into Apache Kafka. For such a data flow, it does not really make sense to output any data, as the data flow itself introduces a Kafka topic as the sink of the data.

You can design a different data flow to be called in response to a message being placed onto an SNS topic. The data flow may take that notification, filter out messages that do not match some criteria, and then send any messages that do match the criteria to an Output Port. That output may then be connected to another destination through an AWS Lambda destination.

Output ports can also be used when using the API Gateway trigger to output the data that should be sent back to the client that made the API call.

No output port

If no Output Port is present in the data flow's root group, no output will be provided from the function invocation.

One output port

If one Output Port is present, and its name is "failure", any data routed to that Output Port will cause the function invocation to fail. No output will be provided as the output of the function invocation.

Two output ports, where one is failure

If two Output Ports are present, and one of them is named "failure", that Output Port will behave as the "failure" port described above. The other Output Port is considered a successful invocation. When the data flow is invoked, if a single FlowFile is routed to this Port, the contents of the

FlowFile will be emitted as the output of the function. If more than one FlowFile is sent to the Output Port, the data flow will be considered a failure, as Lambda requires a single entity to be provided as its output. In this case a MergeContent or MergeRecord Processor may be used to assemble multiple FlowFiles into a single output FlowFile.

Any port with the name "failure" (ignoring case) will be considered a Failure Port. Additionally, if the *FAILURE_PORTS* environment variable is specified, any port whose name is in the comma-separated list will be considered a Failure Port.

Two or more output ports with no failure port

If two or more Output Ports are present and none of them are failure ports, you must provide the *OUTPUT_PORT* environment variable. This environment variable must match the name of an Output Port at the data flow's root group (case is NOT ignored). Anything that is routed to the specified port is considered the output of the function invocation. Anything routed to any other port is considered a failure.

Parameters

An important step in building a data flow that you can run outside of the NiFi instance where it was built is the concept of parameterization. NiFi allows you to define Processor and Controller Service properties at runtime instead of at build time by using Parameter Contexts. The Lambda function handler allows you to specify the parameters in two ways: using environment variables or using the AWS Secrets Manager.

Environment variables

Any parameter can be specified using the environment variables of the AWS Lambda function. When configuring the Lambda function, simply add an environment variable whose name matches the name of a parameter in your Parameter Context.

Note: AWS has reserved environment variables that you cannot use for your parameters (examples: AWS_ACCESS_KEY, AWS_SECRET_ACCESS_KEY, and so on). See the full list here.

AWS Secrets Manager

A more secure mechanism for storing parameters is to use the AWS Secrets Manager.

Procedure

1. Open the AWS Secrets Manager and click Store a new secret.
2. For secret type, select Other type of secret.
3. Provide your secret information, such as credentials and connection details, as key/value pairs.

   Enter one or more key/value pairs to use as parameters. Each key provided to the Secret Manager maps to a parameter in the data flow's Parameter Context with the same name.

   If the name of the secret matches the name of the Parameter Context in the data flow, there is no further configuration needed. The Lambda function will automatically configure the data flow to pull the specified secret's values to represent the Parameter Context.

   Note:

   Keep in mind that there are cases in which this does not work.
4. When you are ready, click Next.
5. Provide a name for your secret.
6. You can configure other options like tags, resource permissions, rotation schedule and others.

7. On the Review page, review your secret details, and click Store.

   Example:

   You developed a data flow with a Parameter Context named AWS_CONTEXT, and you want to deploy this data flow three times. The first time, it should use the values in the "Context_1" secret. For the second deployment, it should use values for the "Context_2" secret. For the third deployment, it should use values from the "Context_3" secret.

   To accomplish this, you can specify an environment variable named PARAM_CONTEXT_AWS_CONTEXT. In this case, the name of the environment variable is PARAM_CONTEXT_ followed by the name of the Parameter Context. The value of the environment variable will be the name of the secret in the AWS Secrets Manager.

   For the first deployment you would use the environment variable PARAM_CONTEXT_AWS_CONTEXT with a value of "Context_1". For the second deployment, you would use an environment variable named PARAM_CONTEXT_AWS_CONTEXT with a value of "Context_2" and so on.

   If your data flow contains multiple Parameter Contexts, you can also map each of them to different secrets. For example, if you have a Parameter Context named AWS_CONTEXT and another one named CLOUDERA_CONTEXT, and you wanted to map those to secrets named "Context_1" and "Cldr_Context" respectively, you could do so by adding two environment variables: PARAM_CONTEXT_AWS_CONTEXT = "Context_1" and PARAM_CONTEXT_CLOUDERA_CONTEXT = "Cldr_Context".

   Additionally, you can simply default all Parameter Contexts whose names do not map any secrets in AWS Secrets Manager to use a default secret by setting an environment variable named DEFAULT_PARAM_CONTEXT. The value of this environment variable should be the name of the secret to use.

   Important: To use this capability, the Parameter Contexts used in the flow must have names that are compatible with AWS Secrets Manager names and Environment Variable names. To ensure that these naming conventions are followed, Parameter Contexts and Parameters should use names that start with a letter and only consist of letters, numbers, and the underscore (_) character. For example, instead of naming a Parameter Context AWS Parameter Context, use a name of AWS_PARAMETER_CONTEXT when building the data flow.

   Note: The role that is used to launch the Lambda function must be assigned to the right policies to list secrets in the AWS Secrets Manager, as well as read the secrets.

   You can also specify that multiple Secrets map to the same Parameter Context by providing a comma-separated list of Secret names. For example, if you want both the "Kafka" and "Common" Secrets to contribute parameters to a Parameter Context named "ALL_KAFKA", you would set PARAM_CONTEXT_ALL_KAFKA = "Kafka, Common".

   Note: Duplicate keys are not supported and will be non-deterministic. Therefore, it is important to ensure that any Secrets mapped to the same Parameter Context do not share identical key names. Otherwise, you are not able to tell which Secret's key is used for the parameter value.

   The same Secret may also provide parameters to multiple different Parameter Contexts. Therefore, a configuration similar to the following is possible:

```
PARAM_CONTEXT_KAFKA = "Kafka, Common"
PARAM_CONTEXT_SOLR = "Solr, Common"
```

   In this scenario, the Common Secret supplies values to both the KAFKA and SOLR Parameter Contexts.

Example

Now you can update the permissions of your Lambda:

1. Navigate to Configuration Permissions .
2. Click the arrow next to the role name.

   Identity and Access Management (IAM) opens on a new browser tab.

3. Click Add permissions to attach a new policy.

   You can attach a policy to grant the right permissions to the specific secret(s) that should be accessed by the Lambda.

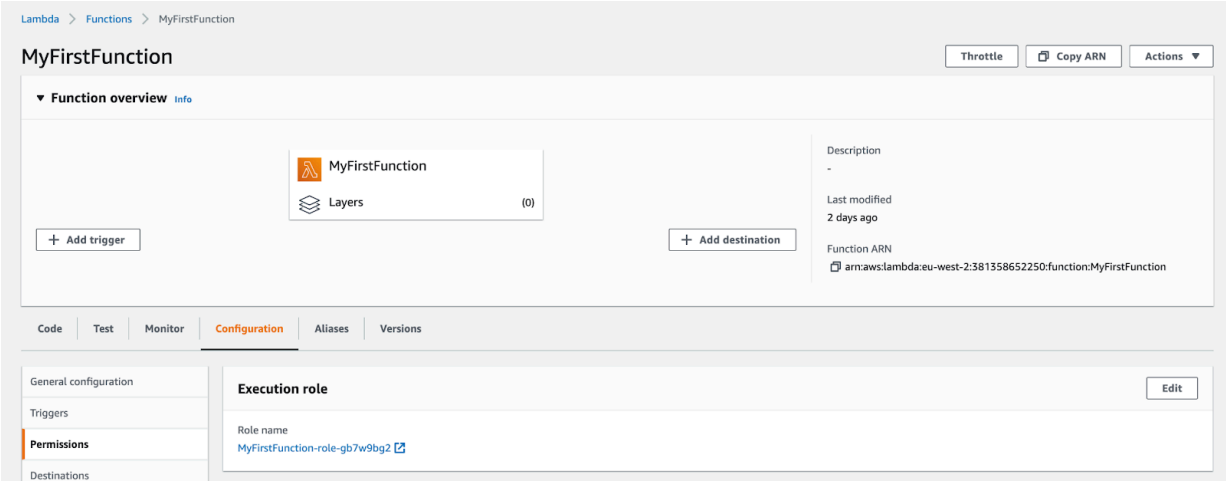

   This is an example of a policy to grant your Lambda access to your secret. Make sure to update the policy with the ARN of your secret.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "secretsmanager:ListSecretVersionIds"
      ],
      "Resource": [
        "ARN:TO:YOUR:SECRET"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "secretsmanager:ListSecrets",
      "Resource": "*"
    }
  ]
}
```

Cold start

Cold start can be defined as the set-up time required to get a serverless application's environment up and running when it is invoked for the first time within a defined period.

What is a cold start?

The concept of cold start is very important when using Function as a Service solutions, especially if you are developing a customer-facing application that needs to operate in real time. A cold start is the first request that a new function instance handles. It happens because your function is not yet running and the cloud provider needs to

provision resources and deploy your code before the processing can begin. This usually means that the processing is going to take much longer than expected.

Before execution, the cloud provider needs to:

- Provision the container that will run the code
- Initialize the container / runtime
- Initialize your function

before the trigger information can be forwarded to the function's handler.

For more information about the concept of cold start, see Operating Lambda: Performance optimization.

How to avoid a cold start?

Cloud providers offer the option to always have at least one instance of the function running to completely remove the occurrence of a cold start. You can read more about predictable start-up times in Provisioned concurrency for AWS Lambda.

Cold start with Cloudera DataFlow Functions

When a cold starts happens with Cloudera DataFlow Functions, the following steps are executed before the trigger payload gets processed:

- The function retrieves the flow definition from the Cloudera DataFlow Catalog.
- The NiFi Stateless runtime is initialized.
- The flow definition is parsed to list the required NAR files for executing the flow.
- The required NAR files are downloaded from the specified repository unless the NARs are made available in the local storage attached to the function.
- The NARs are loaded into the JVM.
- The flow is initialized and components are started.

The step that might take a significant amount of time is the download of the NAR files. In order to reduce the time required for this step as much as possible, the best option is to list the NAR files required for the flow definition and make those files available to the function.

For the related instructions, see Cloud storage on page 16.

Cloud storage

Providing custom extensions / NARs

The binary that is deployed for running AWS Lambda does not include the full NiFi deployment with all NARs / extensions to avoid very long startup times and requiring more expensive configurations for Lambda. Instead, it packages only the NARs necessary to run Stateless NiFi as well as the AWS NAR. If any other extension is needed to run the data flow, it will be automatically downloaded when the Lambda function is initialized on a cold start. The extensions will be downloaded from the Cloudera Maven Repository by default.

However, you can configure an alternate Nexus repository by setting the NEXUS_URL environment variable to the URL of the Nexus Repository where extensions should be downloaded. For example, to use Google's mirror of the Maven Central repository, set the *NEXUS_URL* environment variable to https://maven-central.storage-download.googleapis.com/maven2.

Important: Any configured URL must either be accessible through http or be served over https with a trusted certificate.

This is the recommended approach for downloading extensions that are made available by Cloudera or Apache NiFi. However, if there is a need to provide custom extensions, a simpler route is usually to use Lambda's Layers capability.

1. Create a zip file containing all NAR files that you will need.
2. In the AWS console (or through the CLI), add a layer to your Lambda function, providing the zip file as the source for the layer.
3. Tell Lambda where to look for those extensions by adding another environment variable whose name starts with EXTENSIONS_DIR_.

   You can add multiple extensions, such as EXTENSIONS_DIR_1, EXTENSIONS_DIR_2, EXTENSIONS_DIR_MY_CUSTOM_LIBS. The name of the environment variable is the path where Lambda should look for the extensions. If the zip file provided as a layer does not contain any directories, the value should be /opt. If the zip file is to contain some directory, (for example, extensions/nars), the value for the environment variable would be set to /opt/extensions/nars.

If you add a larger layer, the Lambda configuration takes a bit longer to load on a cold start, and it takes more memory to load the additional classes. So it is not recommended to upload extremely large zip files of extensions if they are not needed.

Providing additional resources

Besides NiFi extensions, some data flows may also require additional resources to work. For example, a JDBC driver for establishing a database connection, or a CSV file for data enrichment. The recommended approach for providing such resources to your Lambda function is through Lambda Layers. You can create a zip file containing the resources that are required for your data flow. This zip file will be extracted into the /opt directory to be accessed by the data flow.

As each deployment of a data flow may need to reference files in a different location depending on its environment, it is a best practice to parameterize resources that need to be accessed by the data flow.

For example, if the data flow contains a DBCPConnectionPool Controller Service for accessing a database, you should use a parameter for the Database Driver Location(s) property of the Controller Service. This way each deployment can specify the location of the JDBC driver independently.

1. Set the Database Driver Location(s) property to #{JDBC_DRIVER_LOCATION}.
2. In Lambda, add a layer that contains our JDBC driver, database-jdbc-driver.jar.
3. Add an Environment Variable named *JDBC_DRIVER_LOCATION* with a value of /opt/database-jdbc-driver.jar.

This way your data flow can be deployed (reused) in many different environments.

S3 Bucket storage

If the extensions or resources of your data flow exceed the size limit of Lambda Layers, you can provide these files in an S3 bucket.

1. Specify the S3 bucket name in the *STORAGE_BUCKET* environment variable.
2. Upload any extensions to a directory in the S3 bucket and specify the directory name in the *STORAGE_EXTENSIONS_DIRECTORY* environment variable.
3. Upload any data flow resources to another directory in the bucket and specify the directory name in the *STORAGE_RESOURCES_DIRECTORY* environment variable.

When your Lambda function does a cold start, it will download any extensions or resources from these locations.

The extensions are directly downloaded in the right location to be loaded in NiFi at startup. The resources are downloaded in /tmp. For example if STORAGE_BUCKET = my-bucket-name, STORAGE_RESOURCES_DIRECTORY = my_resources and there is an hdfs-site.xml file in s3a://my-bucket-name/my_resources/hdfs-site.xml, the file will be available in the Lambda at /tmp/my_resources/hdfs-site.xml.

Note: In order to pull from S3, the Execution Role for the Lambda function must have both the ListBucket and GetObject permissions for the chosen bucket.

Configuring Cloudera VPC to deploy Lambda Functions

In some scenarios, it is required to run the Lambda in a specific Virtual Private Cloud (VPC). For example, if your Lambda function has to interact with services in a Cloudera environment, you must run the Lambda in the same VPC where the Cloudera service is and you must also fulfill some networking prerequisites for the Cloudera DataFlow Function to work as expected.

Before you begin

Cloudera DataFlow Functions running on top of AWS Lambda require access to the Cloudera DataFlow Catalog deployed on the Cloudera Control Plane. This means that Lambda requires outbound internet access to download the flow that is going to be run, and it also requires access to some AWS services like Security Token Service (STS) and AWS Secrets Manager.

This article explains how you can allow a managed, serverless service like AWS Lambda to have internet access. AWS Lambda cannot be deployed on a public subnet in a VPC as it is not possible to grant a Lambda function a public IP address. This means that to have internet access, you need to have at least one private subnet for each availability zone where the Lambda function is going to run and you need to configure a Network Address Translation (NAT) gateway on the public subnets to route its outbound traffic.

Step 1 - Creating VPC private subnets and route tables

A VPC can have multiple subnets, which can be public or private. Depending on the connectivity that you need, you might also have to add gateways and route tables. The Internet Gateway (IG) component enables your instances to connect to the internet through the Amazon EC2 network edge. For more information on basic AWS networking concepts, see How Amazon VPC works.

A public subnet has a direct route to an internet gateway. Resources in a public subnet can access the public internet. By default, the AWS subnets are public, so the instances have public and private IPs and they have a default route table with a route to an internet gateway.

A private subnet does not have a direct route to an internet gateway. Resources in a private subnet require a NAT device to access the public internet, so it cannot have a route to an IG. An EC2 instance in a non-default subnet only has private IPs and does not have internet access.

To allow a private subnet to have internet access you need to use a NAT device, which maps multiple private IPs into a single public IP. So, if you configure the NAT device in a public subnet with a public IP, it can use the internet gateway to route the traffic and get access to the internet.

Use the following procedure to create subnets for your VPC and to create route tables that will be associated with the subnets.

1. Create one private subnet per availability domain of your VPC where you will deploy the Lambda function.
   a. Open the Amazon VPC console.
   b. Click Subnets in the left navigation pane.
   c. Click Create subnet.
   d. For the VPC ID, select the VPC where you want to add the new subnet.

      Note: This should be the Cloudera Environment subnet.

   e. (Optional) For Name, add a name for your subnet.
   f. Define the Availability Zone and the CIDR.
   g. Click Create subnet.

2. Create a route table that will be associated with the subnet.
   a. Click Route tables in the left navigation pane.
   b. Click Create route table.
   c. (Optional) For Name, add a name for your route table.
   d. For VPC, choose your VPC.
   e. (Optional) To add a tag, click Add new tag and add the tag key and tag value.
   f. Click Create route table.

Note: Follow these steps for all subnets and use a common nomenclature to identify the route table of a subnet.

Step 2 - Creating NAT gateways and configuring route tables

Now that you have your subnets, you need to create and configure the Network Address Translation (NAT) gateways that map the private IPs to a public IP and allow the instances in your private subnets to connect to services outside your VPC.

1. Create NAT gateways.
   a. Open the Amazon VPC console.
   b. Click NAT gateways in the left navigation pane.
   c. Click Create NAT gateway.
   d. (Optional) Add a name for the NAT gateway.
   e. Define the subnet where you want to create the NAT gateway.
   f. Define the connectivity type.

      Important: Important: The subnet where the NAT gateway is deployed must be a public subnet to allow internet access, so make sure that the connectivity type is set to public. For more information about the difference between public and private NAT gateways, see NAT gateways.
   g. Allocate an Elastic IP. For more information, see Elastic IP addresses.
2. Configure the route tables of the private subnets created in the previous procedure.
   a. Click Route tables.
   b. Search for the route table.
   c. Click Routes Edit routes .

      There are two routes that need to be defined.

      Route 1

      - Destination: <VPC CIDR>
      - Target: local
      - Status: Active

      Route 2

      - Destination: 0.0.0.0/0
      - Target: <Search for one of the NAT Gateways created>
      - Status: Active
   d. Click Save Changes.
3. Associate each route table with each private subnet.
   a. Click Subnet Associations.
   b. Select the specific subnet.
   c. Save the associations.
   d. Validate that subnets have the route table with it.

Note: Follow these steps for all route tables of the private subnets.

Result

With this, the private subnet components are able to communicate with other components within the same VPC and have internet access.

File system for content repository

If your function is going to process very large files, the data may not be held in memory, which would cause errors. In that case you may want to attach a File System to your Lambda and tell Cloudera DataFlow Functions to use the File System as the Content Repository.

Configuration

If you want to attach a File System to your Lambda, you are required to connect your Lambda to the VPC where the File System is provisioned.

However, by default, Lambda runs your functions in a secure VPC with access to AWS services and the internet. Lambda owns this VPC, which is not connected to your account's default VPC. When you connect a function to a VPC in your account, the function cannot access the internet unless your VPC provides access. Access to the internet is required so that Cloudera DataFlow Functions can request the Cloudera DataFlow Catalog and retrieve the flow definition to be executed.

In such a situation, a recommended option is to create a VPC with a public subnet and multiple private subnets (to increase resiliency across multiple availability zones). You can then create an Internet Gateway and NAT Gateway to give your Lambda access to the internet. For more information, see the AWS Knowledge Center.

Concurrent access

Depending on your Lambda's configuration and the throughput of the events triggering the function, you may have multiple instances of the Lambda running concurrently and each instance would share the same File System. For more information, see this AWS Compute Blog article.

To ensure each instance of the function has its own "content repository", you can use the CONTENT_REPO environment variable when configuring your Lambda and give it the value of the *LOCAL MOUNT PATH* of the attached File System:

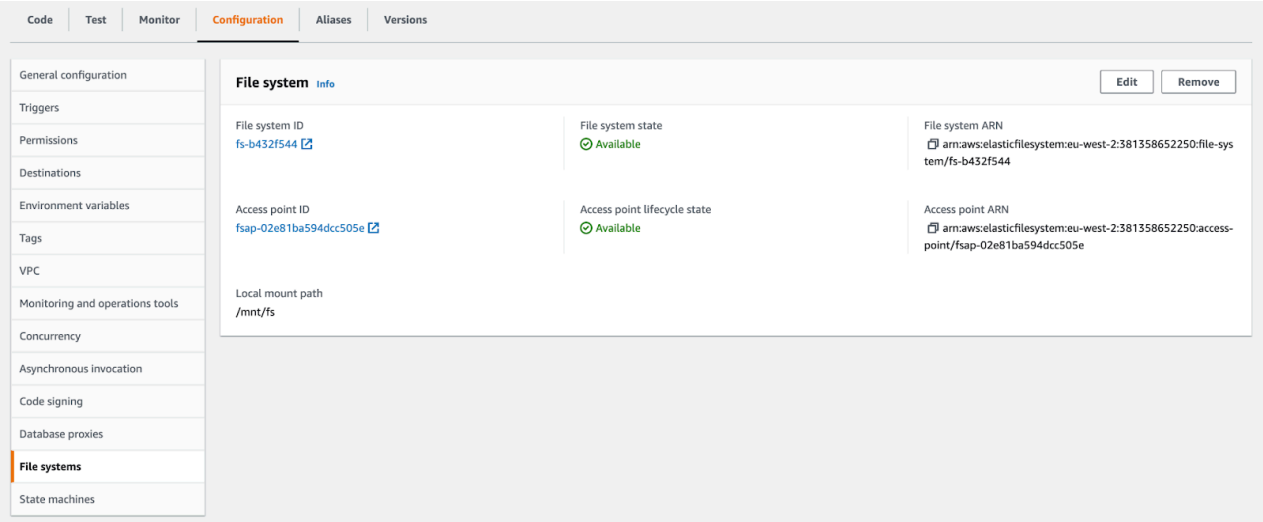

In this case, each instance of the function will create its content repository on the File System under the following path: ${CONTENT_REPO}/<function name>/content_repository/<epoch timestamp>-<UUID>

You can share the same File System across multiple Lambda functions if desired depending on the IO performances you need.

Output size constraints

When you design a data flow that will be deployed within AWS Lambda, you create a Process Group that contains a single Input Port (for example named Input) and two Output Ports (for example named success and failure). If data is routed to the failure port, the entire invocation is failed. If data is routed to the success port, the contents of the FlowFile become the output of the Lambda function. This allows you to set a destination for your AWS Lambda. For example, you can send the output to an SQS queue or another Lambda function.

Consider the following when designing your flow:

- AWS Lambda limits the size of the output. While the maximum output size is configurable, the default is 6 MB. So, if the size of the FlowFile routed to success exceeds 6 MB, the result is a failure, even if you have no destination configured.
- Take into account the size constraints on any downstream system that may be used as the Lambda destination, and do not send a large FlowFile as the output of the Process Group. Instead, you should have the last processor in the data flow auto-terminate the success relationship so that there is no output FlowFile. Or you can use a ReplaceText processor with the Replacement Strategy set to Always Replace. This allows you to set the contents to some predefined text, such as: { "action": "perform dataflow", "success": true, "fileSize": ${fileSize} }

DataFlow state

By default, if your data flow contains any stateful processors (for example, ListSFTP), this state will automatically be stored in a DynamoDB table called nifi_state. This name can be set using the *DYNAMODB_STATE_TABLE* environment variable. If state is to be stored in DynamoDB, the role executing the Lambda function requires permissions to create and describe tables, and permissions to put and get items. If the data flow contains no stateful processors, this provider is not used, and requires no extra permissions.

The default DynamoDB billing mode is PAY_PER_REQUEST. To change this to PROVISIONED, set the *DYNAMODB_BILLING_MODE* environment variable to PROVISIONED, and set the *DYNAMODB_READ_CAPACITY_UNITS* and *DYNAMODB_WRITE_CAPACITY_UNITS* variables accordingly. For more information, see Read/write capacity mode.

The DynamoDB state provider can be disabled even if your data flow contains stateful processors by setting the *DISABLE_STATE_PROVIDER* environment variable to true.

Warning: This will cause the processor state to get lost between function executions.

Configuring Kerberos

Depending on the dataflow that is used, you may need to enable Kerberos authentication in order to interact with some service(s).

To enable Kerberos, you must provide an appropriate krb5.conf file. This file tells the Cloudera DataFlow Function where the Key Distribution Center (KDC) is located and the Kerberos Realm to use.

To specify this, make the krb5.conf available to Cloudera DataFlow Functions. You can do it by using the Lambda Layer or by placing the file in an S3 bucket. For more information, see *S3 Bucket storage*.

Additionally, you must tell Cloudera DataFlow Function where it can find this krb5.conf file. To do this, add an environment variable to your Lambda function. The environment variable must have the name KRB5_FILE and the value must be the fully qualified filename of the krb5.conf file. For example, it might use /opt/krb5.conf.

In order to use Keytabs in your data flow, it is also important to provide the necessary keytab files to the Cloudera DataFlow Function. You can accomplished it by using a Lambda Layer or an S3 bucket. While the krb5.conf file is a system configuration, the keytab is not. The keytab will be configured in the data flow on a per-component basis. For example, one processor may use keytab nifi-keytab-1 while some Controller Service makes use of keytab hive-keytab-4. The location of these files may change from deployment to deployment. For example, for an on-premises deployment, keytabs may be stored in /etc/security/keytabs while in Cloudera DataFlow Function, they may be made available in /opt.

So it is better to use Parameter Contexts to parameterize the location of the keytab files. This is a good practice for any file-based resource that will need to be made available to a deployment. In this case, because there is a parameter referencing the location, you need to provide Cloudera DataFlow Function with a value for that parameter. For more information, see *Parameters*.

Related Information

S3 Bucket storage

Parameters

Handling failures

You can use a failure Output Port to notify the Cloud function that an invocation has failed.

Lambda will automatically retry the invocation, by default, twice. After that, if it still fails, the default option in Lambda is to simply drop the event notification. This could result in data loss, as it means that our Lambda function will not have a chance to process the data again.

Instead, you can also choose to use a "Dead-Letter Queue" for the data:

1. On the Function details page of the AWS Console, navigate to the Configuration Asynchronous invocation tab.
2. Click Edit.

   You can see an option to configure Dead-letter queue service. Use a dead-letter queue to send discarded events to an Amazon SQS queue or Amazon SNS topic. Your function's execution role requires permission to write to the queue or topic.

   You can use None (the default, which drops the notification), or you can configure an SNS or SQS queue to use in the case of failure. This would give you the option to later go back and manually process the data or triage and analyze it to understand why processing failed.

For more information, see the AWS documentation.

Testing your Lambda function

Once you have built and verified your dataflow and uploaded it to the Cloudera DataFlow catalog, you can create your Lambda function.

After creating the Lambda function and configuring all necessary Environment Variables, you should test that you have configured all of the settings correctly before publishing the Lambda.

For this, you can use the Test tab of AWS Lambda Function details screen. Lambda provides you with many options for sample data, so you can easily create an example JSON document to simulate the events that your Lambda function will handle.

Click the Test button to run the data flow. If there are problems, you can inspect the log messages to understand the problem. If the logs do not provide enough details, you may need to adjust the log levels to gain more debugging information. Ror more information, see *Adjusting logs levels*.

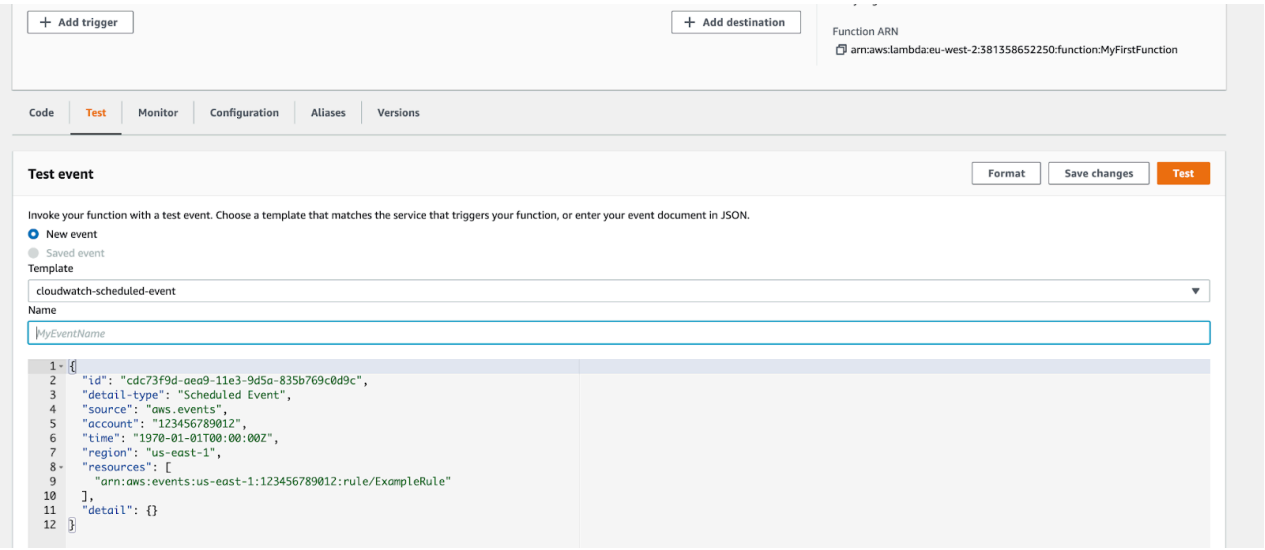

The amount of time the Lambda function takes to run depends heavily on the data flow and the number of extensions it needs. Because the Lambda function may have to download some extensions from Nexus and perform initialization, a cold start may take longer time. Even 20-30 seconds is not uncommon for a data flow with several extensions, while other data flows may complete in 10 seconds.

After you run the function successfully, it may be helpful to run several additional iterations to understand how the function will perform when a cold start is not necessary. Again, this depends heavily upon the configured data flow and other services that it may interact with. A simple data flow may complete in as a short time as 10 milliseconds while a data flow that must perform complex transformations and reach out to one or more web services or databases for enrichment may take several seconds to complete.

It is important to adjust Lambda's timeout configuration on the Configuration General configuration tab if the cold start takes longer than the amount of time allocated.

Related Information

Adjusting logs levels

Publishing your Lambda function

When the Lambda configuration is ready, you can publish a new version of the Lambda function.

For more information about Lambda function versions, see the AWS documentation.

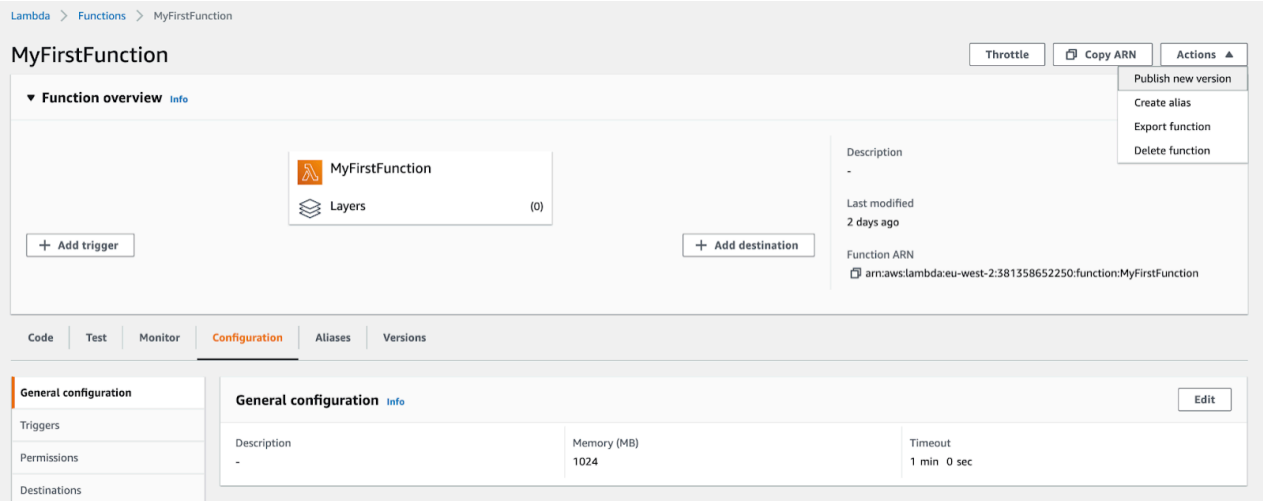

You can define the Triggers/Destinations for the newly published version of your Lambda function and everything will be ready to process the events.

Monitoring and logs

For each invocation of your Lambda, you are able to access information about the duration of the execution and the resources provisioned and used during the function's execution.

Note: With Cloudera DataFlow Functions, a cold start may take a few seconds.

This view is useful in case you want to adjust the amount of memory provisioned with your function and optimize the cost charged by AWS:

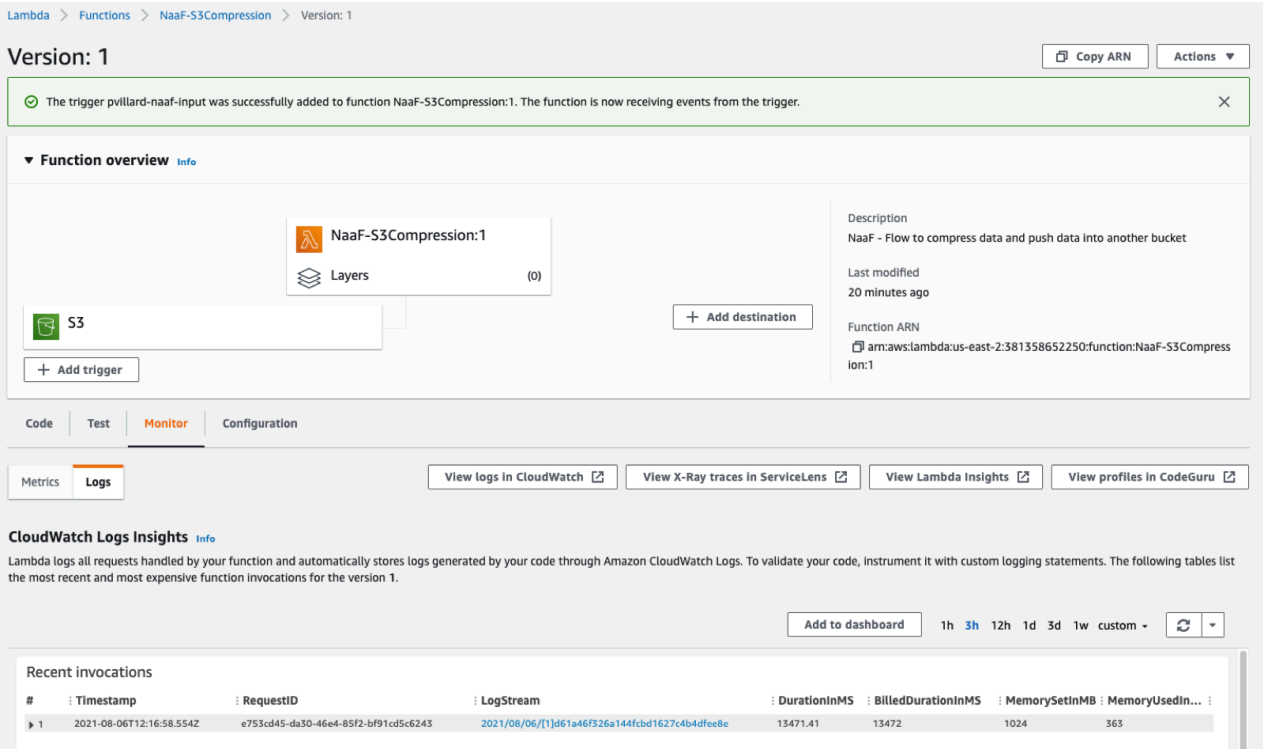

From this view, you can also access the logs of the function's execution:

```
2021-08-06T14:16:54.796+02:00   [main] INFO org.apache.nifi.groups.StandardProcessGroup - Added Connection[ID=1e2410b3-f90c-3cde-80bd-df5843e888fb, Source ID=c28026c1-
2021-08-06T14:16:54.797+02:00   [main] INFO org.apache.nifi.groups.StandardProcessGroup - Added Connection[ID=7fba82c7-c0cd-3b6c-b50a-4b191478c756, Source ID=6b3f9082-
2021-08-06T14:16:54.797+02:00   [main] INFO org.apache.nifi.groups.StandardProcessGroup - Added Connection[ID=2816399e-f373-38f1-82d2-13a31f48a14d, Source ID=60d4c52c-
2021-08-06T14:16:54.814+02:00   [main] INFO org.apache.nifi.groups.StandardProcessGroup - Added Connection[ID=8541305e-082e-313c-99e3-bb77451ffdb4, Source ID=3e5e1010-
2021-08-06T14:16:54.815+02:00   [main] INFO org.apache.nifi.groups.StandardProcessGroup - Updating LocalPort[id=3e5e1010-d6e4-3e57-9478-b2a70d2a7355, type=INPUT_PORT,
2021-08-06T14:16:54.894+02:00   [main] INFO org.apache.nifi.stateless.flow.StandardStatelessDataflowFactory - NiFi Stateless Engine and Dataflow created and initialize
2021-08-06T14:16:54.903+02:00   [main] INFO org.apache.nifi.controller.scheduling.StatelessProcessScheduler - Starting PutS3Object[id=58cbcfbd-84d8-3fe7-a76e-0ee58d930
2021-08-06T14:16:54.903+02:00   [main] INFO org.apache.nifi.controller.StandardProcessorNode - Starting PutS3Object[id=58cbcfbd-84d8-3fe7-a76e-0ee58d9302a5]
2021-08-06T14:16:54.916+02:00   [main] INFO org.apache.nifi.controller.scheduling.StatelessProcessScheduler - Starting FetchS3Object[id=6b3f9082-ee4b-3c9b-a2b7-e55ff0d
2021-08-06T14:16:54.916+02:00   [main] INFO org.apache.nifi.controller.StandardProcessorNode - Starting FetchS3Object[id=6b3f9082-ee4b-3c9b-a2b7-e55ff0db7816]
2021-08-06T14:16:54.916+02:00   [main] INFO org.apache.nifi.controller.scheduling.StatelessProcessScheduler - Starting EvaluateJsonPath[id=d130f25d-197f-3dca-b715-b385
2021-08-06T14:16:54.916+02:00   [main] INFO org.apache.nifi.controller.StandardProcessorNode - Starting EvaluateJsonPath[id=d130f25d-197f-3dca-b715-b385dc6b5676]
2021-08-06T14:16:54.916+02:00   [main] INFO org.apache.nifi.controller.scheduling.StatelessProcessScheduler - Starting SplitJson[id=c28026c1-ae71-3be9-bb8c-45a94ca745f
2021-08-06T14:16:54.916+02:00   [main] INFO org.apache.nifi.controller.StandardProcessorNode - Starting SplitJson[id=c28026c1-ae71-3be9-bb8c-45a94ca745f3]
2021-08-06T14:16:54.917+02:00   [main] INFO org.apache.nifi.controller.scheduling.StatelessProcessScheduler - Starting CompressContent[id=60d4c52c-3f07-30fe-af16-65139
2021-08-06T14:16:54.917+02:00   [main] INFO org.apache.nifi.controller.StandardProcessorNode - Starting CompressContent[id=60d4c52c-3f07-30fe-af16-651397cef438]
2021-08-06T14:16:55.859+02:00   [Component Lifecycle for dataflow S3Compression Thread-2] INFO org.apache.nifi.processors.aws.s3.FetchS3Object - FetchS3Object[id=6b3f9
2021-08-06T14:16:55.860+02:00   [Component Lifecycle for dataflow S3Compression Thread-1] INFO org.apache.nifi.processors.aws.s3.PutS3Object - PutS3Object[id=58cbcfbd-
2021-08-06T14:16:55.874+02:00   [Component Lifecycle for dataflow S3Compression Thread-2] INFO org.apache.nifi.processors.aws.s3.FetchS3Object - FetchS3Object[id=6b3f9
2021-08-06T14:16:55.874+02:00   [Component Lifecycle for dataflow S3Compression Thread-1] INFO org.apache.nifi.processors.aws.s3.PutS3Object - PutS3Object[id=58cbcfbd-
2021-08-06T14:16:57.212+02:00   [main] INFO org.apache.nifi.stateless.flow.StandardStatelessFlow - Successfully initialized components in 2297 millis (4 millis to perf
2021-08-06T14:16:57.392+02:00   [main] INFO org.apache.nifi.aws.lambda.StatelessNiFiFunctionHandler - Enqueued 798 bytes for Input Port events
2021-08-06T14:16:57.533+02:00   [Run Dataflow S3Compression] INFO org.apache.nifi.processors.standard.SplitJson - SplitJson[id=c28026c1-ae71-3be9-bb8c-45a94ca745f3] Sp
2021-08-06T14:16:58.039+02:00   [Run Dataflow S3Compression] INFO org.apache.nifi.processors.aws.s3.FetchS3Object - FetchS3Object[id=6b3f9082-ee4b-3c9b-a2b7-e55ff0db78
2021-08-06T14:16:58.075+02:00   [Run Dataflow S3Compression] INFO org.apache.nifi.processors.standard.CompressContent - CompressContent[id=60d4c52c-3f07-30fe-af16-6513
2021-08-06T14:16:58.541+02:00   [Run Dataflow S3Compression] INFO org.apache.nifi.processors.aws.s3.PutS3Object - PutS3Object[id=58cbcfbd-84d8-3fe7-a76e-0ee58d9302a5]
2021-08-06T14:16:58.552+02:00   [Run Dataflow S3Compression] INFO org.apache.nifi.stateless.flow.StandardStatelessFlow - Ran dataflow in 1146 millis
2021-08-06T14:16:58.553+02:00   [main] INFO org.apache.nifi.aws.lambda.StatelessNiFiFunctionHandler - Successfully triggered dataflow but there were no output FlowFile
2021-08-06T14:16:58.554+02:00   END RequestId: e753cd45-da30-46e4-85f2-bf91cd5c6243
2021-08-06T14:16:58.554+02:00   REPORT RequestId: e753cd45-da30-46e4-85f2-bf91cd5c6243 Duration: 13471.41 ms Billed Duration: 13472 ms Memory Size: 1024 MB Max Memory
```

Additional monitoring capabilities may be enabled in AWS.

Adjusting logs levels

Cloudera DataFlow Functions makes use of the SLF4J Simple Logger for logging purposes. You can update the log levels by adjusting the JVM's system properties.

To do so, you need to set the *JAVA_TOOL_OPTIONS* environment variable.

- To set a log level for a given logger, you can set the JVM system property named org.slf4j.simpleLogger.log.< logger name> to the desired log level.

  For example, if you want to enable DEBUG logging for the AwsSecretsManagerParameterValueProvider to understand whether or not parameters are being fetched from AWS Secrets Manager, you would set the *JAVA_TOOL_OPTIONS* environment variable to -Dorg.slf4j.simpleLogger.log.org.apache.nifi.stateless.parameter.AwsSecretsManagerParameterValueProvider=DEBUG.
- You can also set additional logger levels by adding multiple -D options separated by spaces. For example, to enable DEBUG logs on both the AWS Secrets Manager Parameter Value Provider and the Stateless Bootstrap class, you would set *JAVA_TOOL_OPTIONS* to a value of:

  ```
  -Dorg.slf4j.simpleLogger.log.org.apache.nifi.stateless.parameter.AwsSecr
  etsManagerParameterValueProvider=DEBUG
  -Dorg.slf4j.simpleLogger.log.org.apache.nifi.stateless.bootstrap.Stateles
  sBootstrap=DEBUG
  ```

AWS Lambda triggers

All AWS Lambda triggers are supported with Cloudera DataFlow Functions. In this section you can review the most commonly used triggers and examples of the FlowFile's content that would be generated following a triggering event. You can add triggers on the Configuration Triggers tab.

S3 Trigger

This trigger can be used, for example, to start your function whenever a file is landing into AWS S3:

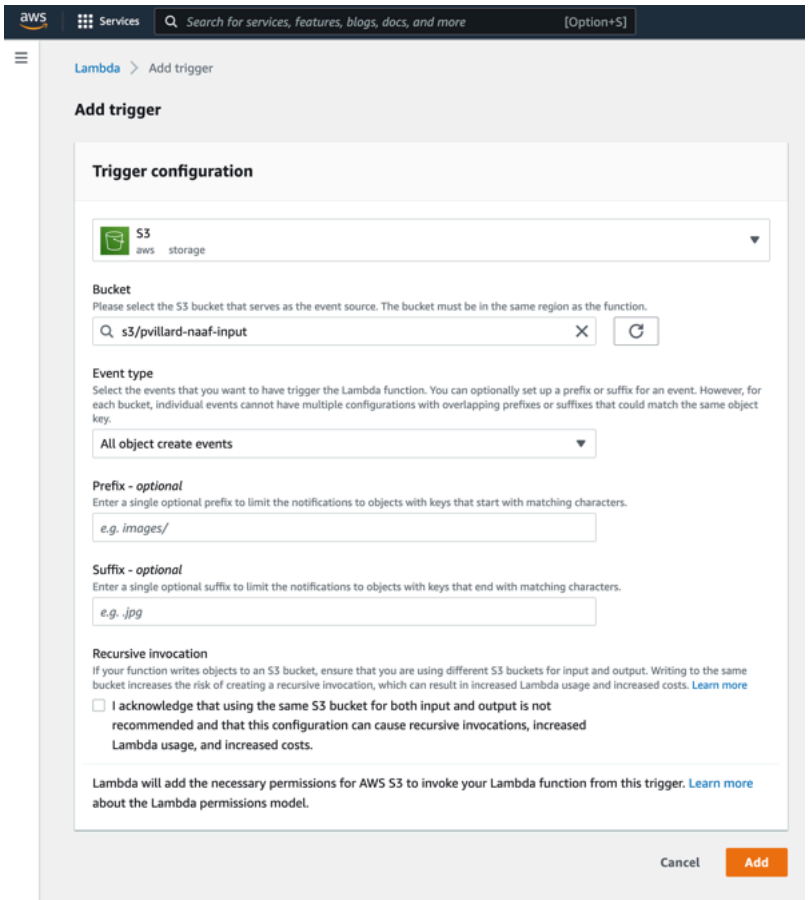

The FlowFile's content would look like this:

```
{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-2",
      "eventTime": "2021-08-06T09:53:57.377Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "AWS:AIDAVRSV7HNNHFQQOE2F4"
      },
      "requestParameters": {
        "sourceIPAddress": "18.222.152.52"
      },
      "responseElements": {
        "x-amz-request-id": "WBPM5W0026J3NWEH",
        "x-amz-id-2": "Cco/DVtbHFYtb+rXQrEMUFJnZSxtebZL61KG79emsACUDqUIyrAziY3frha9AzX+RSLx9syJMl4OMCn06poZv1dJQJg5TgRx"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "ad0ec124-9c9f-476e-9dd1-9e513d9af822",
        "bucket": {
          "name": "pvillard-naaf-input",
          "ownerIdentity": {
            "principalId": "A339RKRQ79D9V"
          },
          "arn": "arn:aws:s3:::pvillard-naaf-input"
        },
        "object": {
          "key": "04f56bcf-cb15-4244-9379-2abc7ac0cd8d",
          "size": 1024,
          "eTag": "265bc4c12162ffca179a58975728a033",
          "sequencer": "00610D06B8BE365D41"
        }
      }
    }
  ]
}
```


API Gateway Trigger

This trigger is used to expose an HTTPS endpoint so that clients can trigger the function with HTTP requests. This is particularly useful to build microservices. It can also be used to create an HTTPS Gateway for ingesting data into Kafka, or to provide an HTTP Gateway to many MiNiFi agents sending data into the cloud. See more information about these use cases in the *next sections*.

Configuration example:

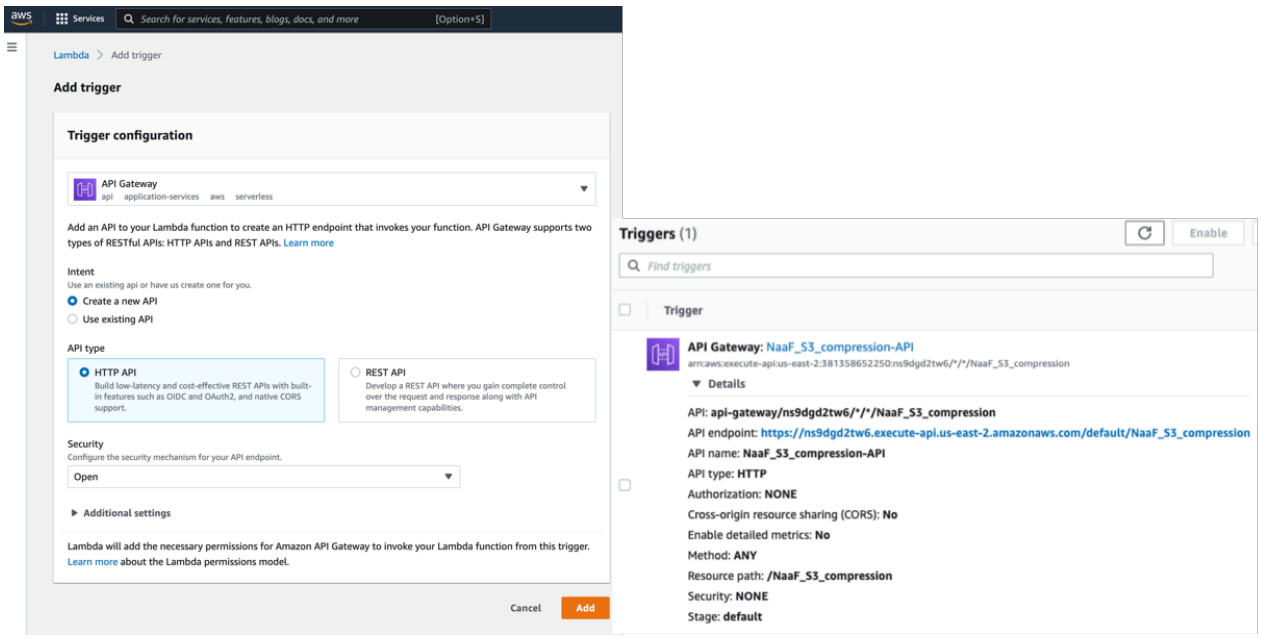

Example of event generated:

```
{
  "version": "2.0",
  "routeKey": "ANY /NaaF_S3_compression",
  "rawPath": "/default/NaaF_S3_compression",
  "rawQueryString": "",
  "headers": {
    "accept-encoding": "gzip",
    "content-length": "34",
    "content-type": "application/json",
    "date": "Fri, 06 Aug 2021 12:35:29 GMT",
    "host": "ns9dgd2tw6.execute-api.us-east-2.amazonaws.com",
    "x-amzn-trace-id": "Root=1-610d2c92-7651f91a5dcbab8671048936",
    "x-forwarded-for": "3.19.60.232",
    "x-forwarded-port": "443",
    "x-forwarded-proto": "https"
  },
  "requestContext": {
    "accountId": "381358652250",
    "apiId": "ns9dgd2tw6",
    "domainName": "ns9dgd2tw6.execute-api.us-east-2.amazonaws.com",
    "domainPrefix": "ns9dgd2tw6",
    "http": {
      "method": "POST",
      "path": "/default/NaaF_S3_compression",
      "protocol": "HTTP/1.1",
      "sourceIp": "3.19.60.232",
      "userAgent": ""
    },
    "requestId": "DpPm3j83iYcEP7w=",
    "routeKey": "ANY /NaaF_S3_compression",
    "stage": "default",
    "time": "06/Aug/2021:12:35:30 +0000",
    "timeEpoch": 1628253330104
  },
  "body": "{\n  \"data\": \"this is my payload\"\n}",
  "isBase64Encoded": false
}
```


EventBridge (CloudWatch Events) - CRON driven functions

This trigger can be used with rules including CRON expressions so that your function is executed at given moments. You can use it, for example, to execute a particular processing once a day.

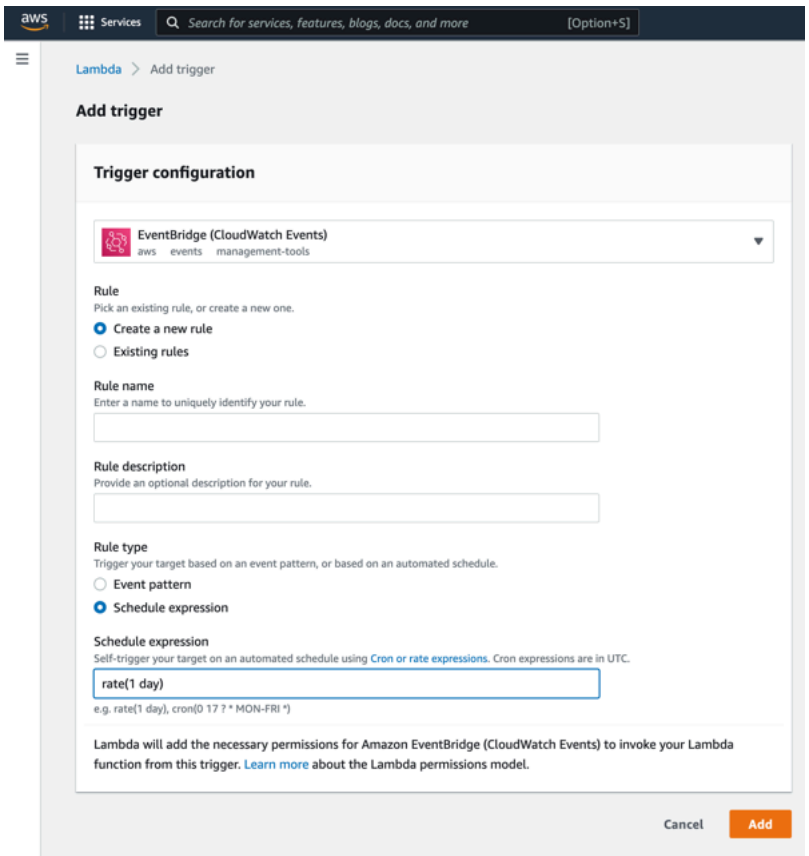

Kinesis

This trigger can be used to process a stream of events using Cloudera DataFlow Functions. The trigger configuration could be:

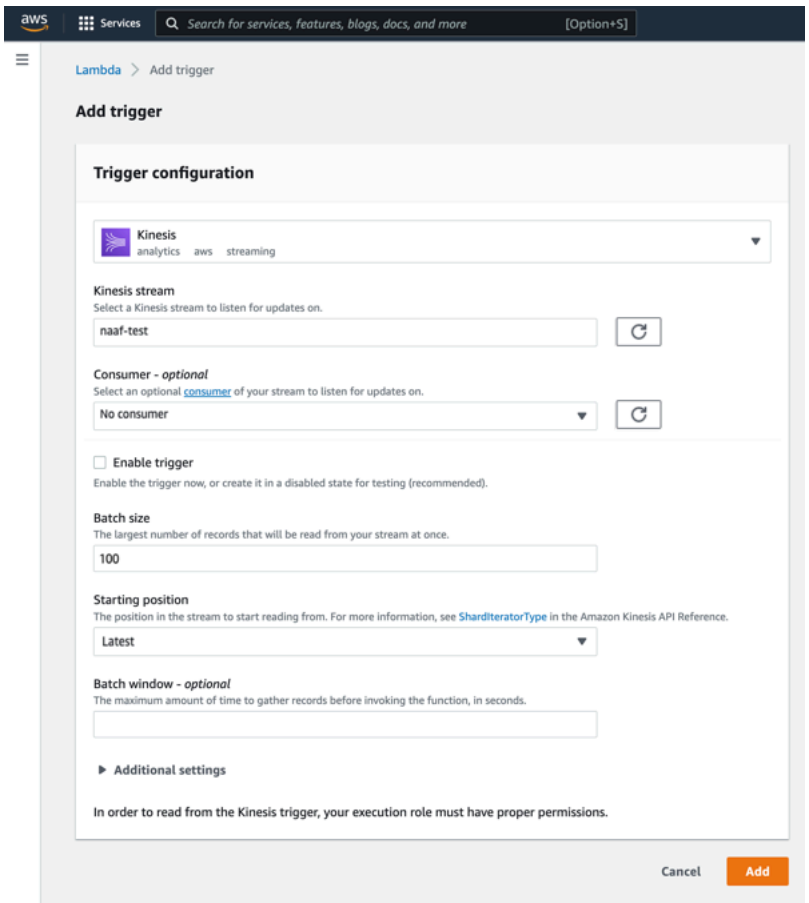

The FlowFile's content would look like this:


```
{
  "Records": [
    {
      "kinesis": {
        "kinesisSchemaVersion": "1.0",
        "partitionKey": "1422215794",
        "sequenceNumber": "49620838107107426836706204605255346319435863317985361922",
        "data": "ewogICJkYXRhIjogInRoaXMgaXMgbXkgcGF5bG9hZCIKfQ==",
        "approximateArrivalTimestamp": 1628255950.71
      },
      "eventSource": "aws:kinesis",
      "eventVersion": "1.0",
      "eventID": "shardId-000000000000:49620838107107426836706204605255346319435863317985361922",
      "eventName": "aws:kinesis:record",
      "invokeIdentityArn": "arn:aws:iam::381358652250:role/service-role/NaaF_S3_compression-role-rvm8p2yc",
      "awsRegion": "us-east-2",
      "eventSourceARN": "arn:aws:kinesis:us-east-2:381358652250:stream/naaf-test"
    }
  ]
}
```


Note that the data associated with a particular record may be base 64 encoded. In this case, you may want to consider the Base64EncodeContent processor in your flow definition to decode the content.

AWS MQ

This trigger can be used to trigger your Lambda based on events received in specific queues of AWS MQ. The trigger configuration may look like this:

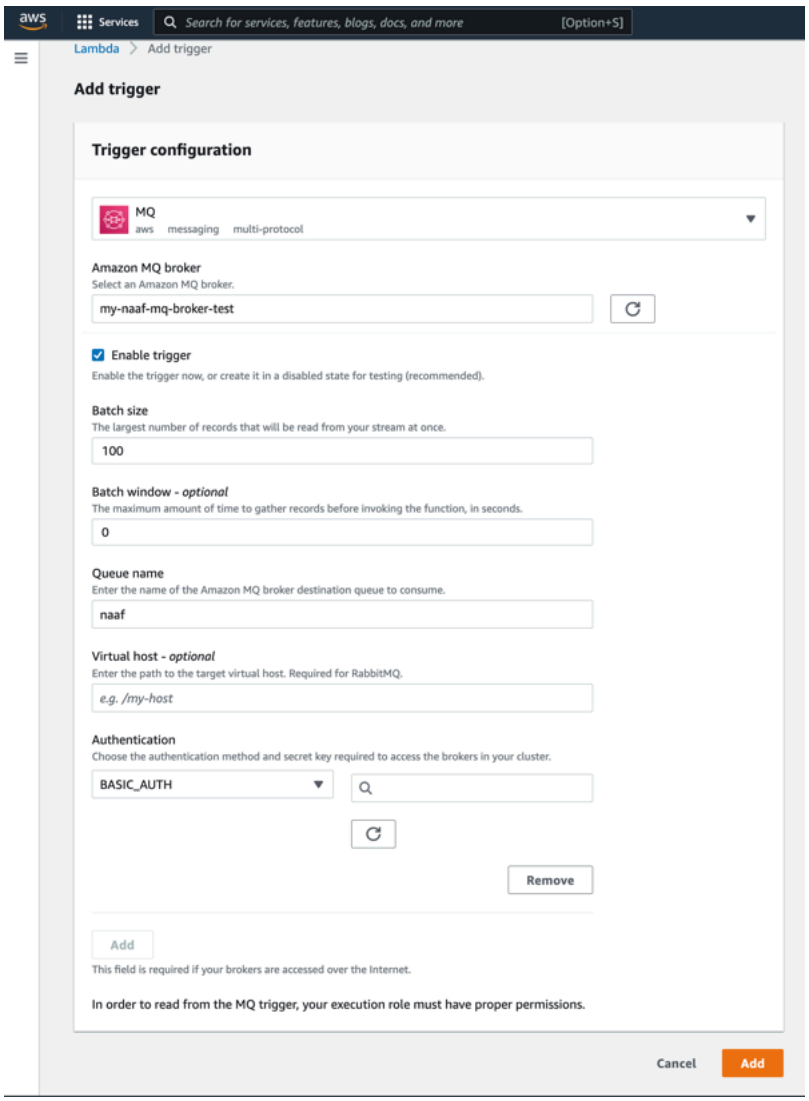

The FlowFile's content would look like this:


```
{
  "eventSource": "aws:mq",
  "eventSourceArn": "arn:aws:mq:us-east-2:381358652250:broker:my-naaf-mq-broker-test:b-7feee146-7a61-40a3-9245-d06419a5c118",
  "messages": [
    {
      "messageID": "ID:b-7feee146-7a61-40a3-9245-d06419a5c118-1.mq.us-east-2.amazonaws.com-42681-1630411308974-4:1:1:1:1",
      "messageType": "jms/text-message",
      "timestamp": 1630412991976,
      "deliveryMode": 1,
      "correlationID": "",
      "replyTo": "null",
      "destination": {
        "physicalName": "naaf"
      },
      "redelivered": false,
      "type": "",
      "expiration": 0,
      "priority": 0,
      "data": "ew0KICAiZGF0YSI6ICJ0aGlzIGlzIG15IHBheWxvYWQiDQp9",
      "brokerInTime": 1630412991976,
      "brokerOutTime": 1630412991980
    }
  ]
}
```

Note that the data associated with a particular message may be base 64 encoded. In this case, you may want to consider the Base64EncodeContent processor in your flow definition to decode the content.

AWS SNS

When using the AWS SNS trigger with a particular topic, the configuration may look like this:

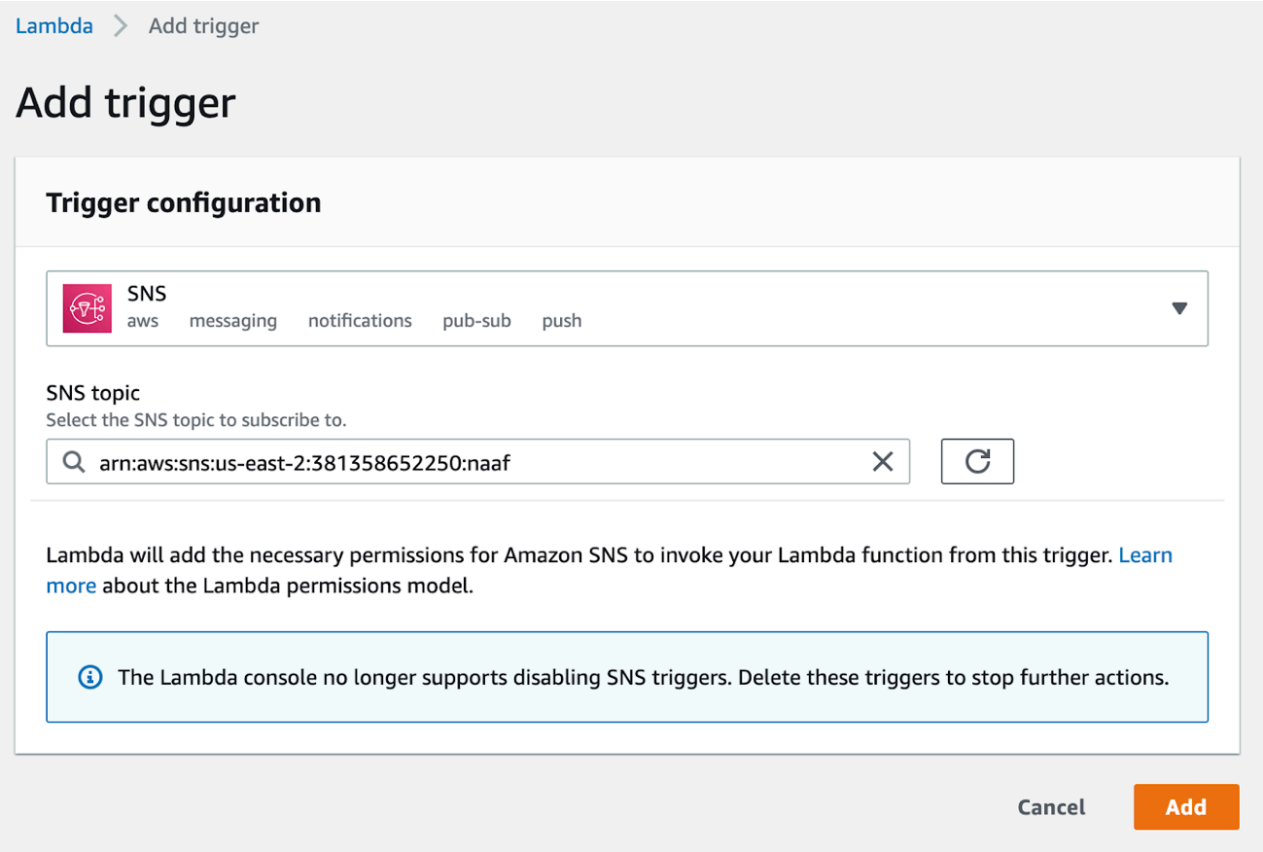

The FlowFile's content would look like this:


```
{
  "Records": [
    {
      "EventSource": "aws:sns",
      "EventVersion": "1.0",
      "EventSubscriptionArn": "arn:aws:sns:us-east-2:381358652250:naaf:d9d324e6-26c9-4c9f-bc17-787422a526e8",
      "Sns": {
        "Type": "Notification",
        "MessageId": "2d8aafed-45fc-5d20-9b73-f64563caaaf6",
        "TopicArn": "arn:aws:sns:us-east-2:381358652250:naaf",
        "Subject": null,
        "Message": "This is my message...",
        "Timestamp": "2021-08-31T12:43:34.806Z",
        "SignatureVersion": "1",
        "Signature":
        "fjxVP2P5wt5jQicwrWDsHORDQQAhyKaPv7NrdRcjosyiKqFc5pXTIw4k2bzKcjtvPaISnKuVYQ42BiX6FPCbWjhGeQ0dPKMjNFTiZg2pFaUFYUHPXmfrBu3Nfj8slEkgQsP9vIoOruNil1jl5kucgz2Pb/
        46adDggMylri928I+4zZ+x3nSTA3GMTt49mIeLTCzw8JUeQsEY3bCqt0sqvuBSM1NyqJYaKoifXy30YLvBZhUPas5jbG32enicrSnX8ZS8+WBDTcI6Z/ubreCKrabTYIK1FojBrVmze8uMHoAjUgcYDvy6T/
        0EL0ZnH/tuueGQ8E7MeyqvvdrwcslBfq==",
        "SigningCertUrl": "https://sns.us-east-2.amazonaws.com/SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
        "UnsubscribeUrl": "https://sns.us-east-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-2:381358652250:naaf:d9d324e6-26c9-4c9f-bc17-
        787422a526e8",
        "MessageAttributes": {
          "naafKeyExample": {
            "Type": "String",
            "Value": "naafValueExample"
          }
        }
      }
    }
  ]
}
```

AWS SQS

When using the AWS SQS trigger with a particular queue, the configuration may look like this:

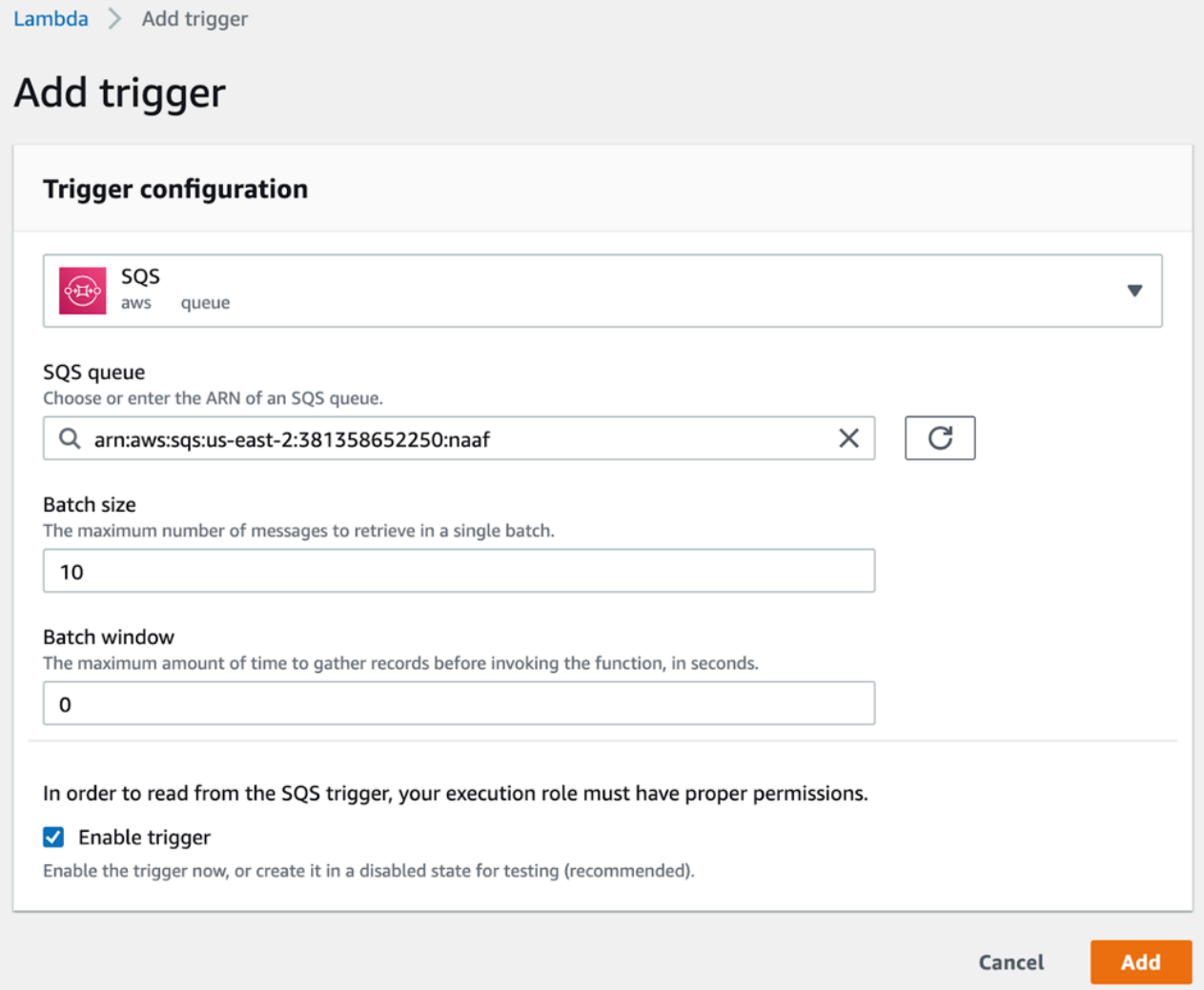

The FlowFile's content would look like this:


```
{
  "Records": [
    {
      "messageId": "2f895bd2-256e-4a37-bf50-e75ecbb9706c",
      "receiptHandle": "AQEBjVQd44ww6p0skNG++ss7ltM63PjL+IGBxeVyQ/3vd9CubJ+6OOjofED168mSrz21oA2fu1gqRneuVVb4c9nColYiy6yeaF6Mo4yqCJDECwscwvU7hfhmCfYk9bNuOchUbf3uIC2z1A2/
      P+RI5hodPc4gX6hQLHSaAv6AtBTPSs7dsyzlmHiCzv2QVb4oE+CAVYIVwzQrXKIjiPgzwU4Jk4s4Go7+VravhOT5xIif5g43WSgO3yxlPKiqFE8L7ESoxLtWG+fV2w9gMU1Ox4xcT5A/d1OZFwjWySVP/
      pWIairgGMxfzNW+oHLfsCzK0EcWN3xFJUHsu2GToIUgfJXx3OlmqcSzeLlNAJ/9jbf/tbBosWpUhN7OxuTFD5LQ6xsP",
      "body": "This is my message...",
      "attributes": {
        "ApproximateReceiveCount": "1",
        "SentTimestamp": "1630414343808",
        "SenderId": "AROAVRSV7HNNJTVJNL3VZ:pvillard@cloudera.com",
        "ApproximateFirstReceiveTimestamp": "1630414343814"
      },
      "messageAttributes": {
        "myKeyExample": {
          "stringValue": "myValueExample",
          "stringListValues": [],
          "binaryListValues": [],
          "dataType": "String"
        }
      },
      "md5OfMessageAttributes": "f170220ddb772bb53d3e06f3fb2e0529",
      "md5OfBody": "fc9ef998398157d220c38142ae53b795",
      "eventSource": "aws:sqs",
      "eventSourceARN": "arn:aws:sqs:us-east-2:381358652250:naaf",
      "awsRegion": "us-east-2"
    }
  ]
}
```

AWS MSK

When using the AWS MSK trigger with a particular Kafka cluster and topic, the configuration may look like this:

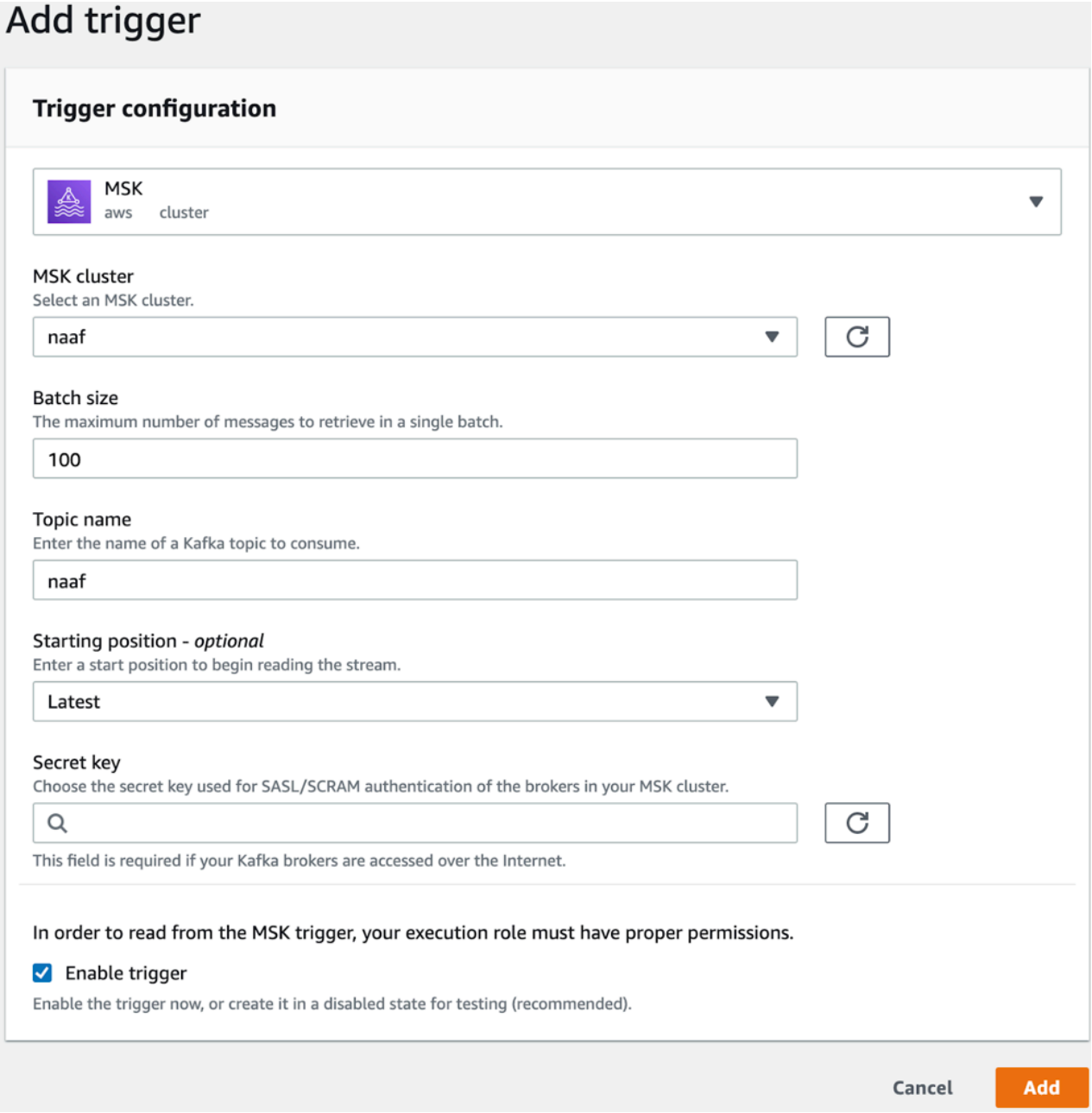

The FlowFile's content would look like this:

```
{
  "eventSource": "aws:kafka",
  "eventSourceArn": "arn:aws:kafka:us-east-2:381358652250:cluster/naaf-msk-cluster/d2a665a0-84a2-473d-bb2d-8853c532cd54-7",
  "bootstrapServers": "b-2.naaf-msk-cluster.vnvbjv.c7.kafka.us-east-2.amazonaws.com:9094,b-1.naaf-msk-cluster.vnvbjv.c7.kafka.us-east-2.amazonaws.com:9094",
  "records": {
    "naaf-0": [
      {
        "topic": "naaf",
        "partition": 0,
        "offset": 0,
        "timestamp": 1630434537574,
        "timestampType": "CREATE_TIME",
        "value": "VGhpcyBpcyBteSBtZXNzYWdlICE=",
        "headers": []
      }
    ]
  }
}
```


Note that the data associated with a particular record may be base 64 encoded (the value field in the above screenshot). In this case, you may want to consider the Base64EncodeContent processor in your flow definition to decode the content.

Creating a Lambda function using CLI

Follow these steps to deploy a Cloudera DataFlow Function using the AWS CLI.

Before you begin

- Your Apache NiFi flow definition is stored in the Cloudera DataFlow Catalog and you have the CRN corresponding to the flow version you want to execute as a function
- You have created a Machine User with the proper role and you have its Access Key and Private Key credentials
- You have installed and configured the AWS CLI on your local machine

Procedure

1. Create a role for your Lambda.

```
aws iam create-role --role-name my_function_role --assume-role-policy-do
cument '{"Version": "2012-10-17","Statement": [{ "Effect": "Allow", "Pri
ncipal": {"Service": "lambda.amazonaws.com"}, "Action": "sts:AssumeRole"
}]}'
```

2. Attach the basic AWS Lambda policy to your role.

```
aws iam attach-role-policy --role-name my_function_role --policy-arn arn
:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
```

3. Copy the following file as a local file named my-function-definition.json.

```
{
    "FunctionName": "first-cli-function",
    "Description": "my first cli function",
    "Timeout": 300,
    "MemorySize": 1024,
    "Environment": {
        "Variables": {
            "DF_ACCESS_KEY": "00000000-0000-0000-0000-000000000000",
            "DF_PRIVATE_KEY": "0000000000000000000000000000000000000000",
            "FLOW_CRN": "crn:cdp:df:us-west-1:00000000-0000-0000-0000-0000
00000000:flow:my-flow/v.1"
        }
    },
    "Tags": {
        "tagKey": "tagValue"
```

```
    },


    "EphemeralStorage": {
        "Size": 2048
    },
    "Code": {
        "S3Bucket": "my-bucket",
        "S3Key": "naaf-aws-lambda-1.0.0-bin.zip"
    },
    "Role": "arn:aws:iam::327162129569:role/service-role/My-Role",

    "Runtime": "java11",
    "Handler": "com.cloudera.naaf.aws.lambda.StatelessNiFiFunctionHandler
::handleRequest",
    "Publish": false,
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ]
}
```

Note:

Executing aws lambda create-function --generate-cli-skeleton gives you additional information that you can add in this JSON payload if you want to further customize your deployment.

4. Update the three sections of the JSON file.

   The JSON file is generally structured into three sections.

   a) The first section is expected to be updated with each function. It includes Function Name, Description, Timeout, Memory Size, Tags, and Environment Variables.

      The Environment Variables must include the following 3 environment variables:

      - DF_ACCESS_KEY: The Cloudera DataFlow Access Key
      - DF_PRIVATE_KEY: The Cloudera DataFlow Private Key
      - FLOW_CRN: The CRN of the specific version of the Cloudera DataFlow in the Cloudera DataFlow Catalog. This always ends with /v.<number>.

      Additionally, you can use environment variables to specify parameters for the dataflow.

   b) The second section of the JSON file contains fields that will likely be updated once when customizing the template for your particular organization. You should add the ARN of the role you created during the first step.

   c) The third section of the JSON file contains fields that will likely remain the same.

5. Create the Lambda function.

```
aws lambda create-function --cli-input-json file://my-function-definitio
n.json
```

   The function is now created. You can check it on the AWS Lambda UI and test the configuration.

6. Use the AWS Lambda UI or CLI to publish a version of the function.

```
aws lambda publish-version --function-name <function name or function ARN>
```

7. Use the AWS Lambda UI or CLI to specify the appropriate trigger for your published version.

Cloudera DataFlow

AWS Resource Panning

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

AWS requirements for Cloudera DataFlow **4**

Cloudera DataFlow networking in AWS **5**

Use your own VPC 6

Allow Cloudera to create a VPC 7

Limitations on AWS **8**

AWS restricted policies **8**

Using Customer Managed Keys with Cloudera DataFlow **8**

Define a new default KMS key for AWS account level EBS encryption 8

Define a new default KMS key for Cloudera environment level EBS encryption 10

AWS requirements for Cloudera DataFlow

As the administrator for your AWS environment, ensure that the environment meets the requirements for Cloudera Public Cloud and Cloudera DataFlow. Then set up your AWS cloud credential and register the environment.

Follow the steps to ensure that your AWS environment meets the Cloudera and Cloudera DataFlow requirements:

Understand your AWS account requirements for Cloudera

- Review the *Cloudera AWS account requirements*. The link is in the *Related information* section below.
- Verify that your AWS account for Cloudera has the required resources.
- Verify that you have the permissions to manage these resources.

Understand the Cloudera DataFlow requirements

- Verify that the following services are available in your environment for Cloudera DataFlow to use:
  - Network – Amazon VPC
  - Compute – Amazon Elastic Kubernetes Service (EKS)
  - Load Balancing – Amazon ELB Classic Load Balancer
  - Persistent Instance Storage – Amazon Elastic Block Store (EBS)
  - Database – Amazon Relational Database Service (RDS)
- Determine your networking option:
  - Use your own VPC
  - Allow Cloudera to create a VPC

  To understand each option, see: *Cloudera DataFlow Networking*. The link is in the *Related information* section below.
- Regions:
  - Select a Cloudera Public Cloud-supported region that also includes the AWS Elastic Kubernetes Service (EKS).

    For more information, see: *Cloudera Supported AWS regions* and the Region Table in *AWS Regional Services*. The links are in the *Related information* section below.
- Ports and outbound network access:
  - Review the port requirements for the Cloudera default security group. See: *Cloudera Management Console - Security groups*. The link is in the *Related information* section below.
  - Configure ports for NiFi to access your source and destination systems in the data flow.
  - If you are using a firewall or a security group setting to prevent egress from the workspace, you must ensure that the outbound destinations required by Cloudera DataFlow are reachable. For more information, see *Outbound network access destinations for AWS*. The link is in the *Related information* section below.
  - If the egress is blocked to these URLs, then autoscaling fails to pull new images and the instances will have broken pods.

    Follow the recommended and minimum required security group settings by AWS. For more information, see *Amazon EKS security group considerations*. The link is in the *Related information* section below.

Set up an AWS Cloud credential

Create a role-based AWS credential that allows Cloudera Public Cloud to authenticate with your AWS account and has authorization to provision AWS resources on your behalf. Role-based authentication uses an IAM role with an attached IAM policy that has the minimum permissions required to use Cloudera.

To set up an AWS Cloud credential, see *Creating a role based provisioning credential for AWS*. The link is in the *Related information* section below.

After you have created this IAM policy, register it in Cloudera as a cloud credential. Reference this credential when you register an AWS environment in Cloudera environment as described in the next step.

Register an AWS environment in Cloudera Public Cloud

A Cloudera user must have the PowerUser role in order to register an environment. An environment determines the specific cloud provider region and virtual network in which resources can be provisioned, and includes the credential that should be used to access the cloud provider account.

To register an AWS environment in Cloudera Public Cloud, see *Cloudera AWS Environments*.

Related Concepts

Cloudera DataFlow networking in AWS

Related Information

Cloudera Public Cloud

Cloudera Public Cloud supported AWS regions

AWS Regional Services

Cloudera Management Console

Amazon EKS security group considerations

Creating a role based provisioning credential for AWS

Cloudera Public Cloud

Outbound network access destinations for AWS

Cloudera DataFlow networking in AWS

Cloudera DataFlow supports different networking options depending on how you have set up your VPC and subnets. If you want Cloudera DataFlow to use specific subnets, make sure that you specify them when registering a Cloudera environment.

If you specified a mix of public and private subnets during environment registration, Cloudera DataFlow by default will provision the Kubernetes nodes in the private subnets. For Cloudera DataFlow to work, the private subnets require outbound internet access. This can be achieved by configuring NAT gateways in separate public subnets and making sure that outbound internet traffic is routed via the NAT gateway. The VPC you are using must have an Internet Gateway set up which ultimately provides internet access to the public subnets. Following this approach allows the Cloudera DataFlow services running on Kubernetes nodes in your private network to connect to the internet while also preventing inbound connections from the internet.

You can configure Cloudera DataFlow to either use a private or public load balancer to allow users to connect to flow deployments. Using a private load balancer is possible when your Cloudera environment contains at least two private subnets. When you are using a private load balancer, you need to ensure connectivity between the client network from where your users are initiating connections and the private subnets in your VPC. This is typically done by setting up VPN access between the private subnets in AWS and the corporate network.

If you want to allow users to connect to flow deployments from the internet you can use the public load balancer option. This option will provision public load balancers in public subnets allowing your users to connect to flow deployments without the need to set up VPN connectivity between the private subnets and your corporate network.

Important: Cloudera recommends that you either use a fully private deployment in private subnets with private load balancers or a mix of private subnets with a public load balancer. Cloudera does not recommend provisioning Cloudera DataFlow in public subnets.

The image below represents a fully private deployment where Kubernetes nodes and load balancers are deployed in the private subnets.

Use your own VPC

If you choose to use your own VPC, verify that it meets the minimum requirements and review Cloudera's recommended setup.

VPCs can be created and managed from the *VPC console on AWS*. For instructions on how to create a new VPC on AWS, refer to *Create and configure your VPC* in the AWS documentation.

Verify that your VPC meets the following requirements and recommendations:

Minimum requirements

- Cloudera DataFlow requires at least two subnets, each in a different Availability Zone (AZ). If you require a public endpoint for Cloudera DataFlow, provision at least one public subnet.
- Ensure that the CIDR block for the subnets is sized appropriately for each Cloudera DataFlow environment. You must have enough IPs to accommodate:
  - The maximum number of autoscaling compute instances.
  - A fixed overhead of 48 IP addresses for three instances for core Cloudera DataFlow services.
- You must enable DNS for the VPC.

Cloudera's recommended setup

- Provision two subnets, each in a different Availability Zone (AZ).
  - If you do not require a public endpoint, use two private subnets.
  - If you require a public endpoint, use one private subnet and one public subnet.
- Private subnets should have routable IPs over your internal VPN. If IPs are not routable, private Cloudera DataFlow endpoints must be accessed via a SOCKS. This is not recommended.
- Tag the VPC and the subnets as shared so that Kubernetes can find them. Also, for load balancers to be able to choose the subnets correctly, you must tag either the private or public subnets.

  A tag in AWS consists of a key and a value.

  - To tag private subnets, enter kubernetes.io/role/internal-elb for the key and 1 for the value.

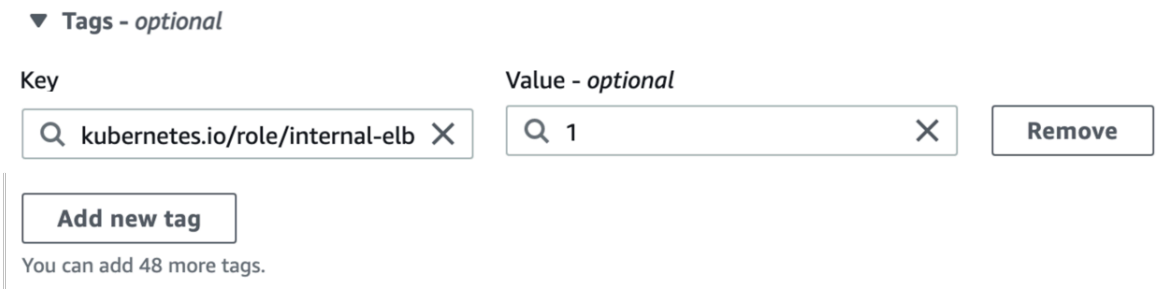

  - To tag public subnets, enter kubernetes.io/role/elb for the key and 1 for the value.

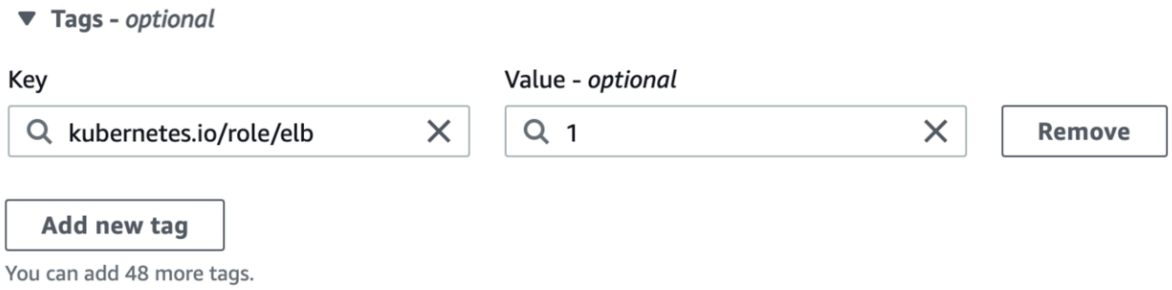

Note: The load balancer must be on a public subnet for access to Cloudera DataFlow. By default, if they are available, Cloudera DataFlow will configure the EKS to run on private subnets.

Related Information

VPC Console on AWS

Create and configure your VPC

Allow Cloudera to create a VPC

You can choose to create a VPC through Cloudera.

If you choose to create a VPC through Cloudera, three subnets will be automatically created.

You will be asked to specify a valid CIDR in IPv4 range that will be used to define the range of private IPs for EC2 instances provisioned into these subnets.

For more information, see the AWS documentation *Amazon EKS - Cluster VPC Considerations* and *Creating a VPC for your Amazon EKS Cluster*. The links are in the *Related information* section below.

Related Information

Amazon EKS - Cluster VPC Considerations

Creating a VPC for your Amazon EKS Cluster

Limitations on AWS

Review the default AWS service limits and your current AWS account limits.

By default, AWS imposes certain default limits for AWS services for each user account. Make sure you review your account's current usage status and resource limits before you start provisioning additional resources for Cloudera and Cloudera DataFlow.

For example, depending on your AWS account, you may only be allowed to provision a certain number of EC2 instances. Be sure to review your AWS service limits before your proceed.

For more information, see the AWS documentation: *AWS Service Limits* and *Amazon EC2 Resource Limits*.

Cloudera DataFlow environments have the following resource limits on AWS:

- Certificate creation (for TLS) uses LetsEncrypt which is limited to 2000 certs/week. As such, a single tenant in Cloudera can create a maximum of 2000 flows per week.

Related Information

AWS Service Limits

Amazon EC2 Resource Limits

ENI Max Pods

AWS restricted policies

Customers with strict security policies beyond what the default Cloudera cross-account policy permits can enable Cloudera DataFlow for a Cloudera Public Cloud environment with more restricted IAM policies.

For information on setting up the required IAM roles and policies, see Setting up Compute Cluster IAM permissions.

Using Customer Managed Keys with Cloudera DataFlow

By default, Cloudera DataFlow uses your account level KMS key for EBS storage encryption. You can optionally secure your data with a custom KMS key.

You have two options to implement Customer Managed Keys (CMKs):

- define a new default KMS key for EBS encryption on AWS account level
- define a key on Cloudera DataFlow environment level

Define a new default KMS key for AWS account level EBS encryption

When you define a new account level default key in AWS, you need to add policies to your key definition that allow for storage provisioning and fulfilling scaling requests.

About this task

Important: Defining a new default key affects all EBS storage encryption within your account.

Procedure

1. Create a custom encryption key on the AWS Management Console.

   The key policy section of the new key must contain additional permissions. Add the three required permission blocks in the example below.

   Replace *[***YOUR ACCOUNT ID***]* and *[***YOUR ACCOUNT REGION***]* with your AWS account ID and with the AWS region where you want to deploy Cloudera DataFlow, respectively.

```
        {
            "Sid": "AllowAutoscalingServiceLinkedRoleForAttachmentOfPer
sistentResources",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::[***YOUR ACCOUNT ID***]:role/aws-se
rvice-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        },
        {
            "Sid": "AllowAutoscalingServiceLinkedRoleUseOfTheCMK",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::[***YOUR ACCOUNT ID***]:role/aws-se
rvice-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow EKS access to EBS.",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
           ],
           "Resource": "*",
           "Condition": {
```

```
                "StringEquals": {
                    "kms:CallerAccount": "[***YOUR ACCOUNT ID***]",
                    "kms:viaService": "ec2.[***YOUR ACCOUNT REGION***].ama
zonaws.com"
                }
            }
        }
```

Important: If you fail to add these permissions, you will encounter failures when enabling Cloudera DataFlow since it will be unable to provision the necessary encrypted storage using the custom key.

2. Set the newly created key as the default KMS key for EBS encryption.

   For more information, see Default KMS key for EBS encryption.

3. If you are also use restricted IAM policies with Cloudera, make sure you provide the KMS CMK for volume encryption when you Create the restricted policies and attach them to the cross-account role .

Define a new default KMS key for Cloudera environment level EBS encryption

When you define a custom KMS key at the Cloudera environment level, you need to add policies to your key definition that allow for storage provisioning and fulfilling scaling requests.

Procedure

1. Create a custom encryption key on the AWS Management Console.

   The key policy section of the new key must contain additional permissions. Add the three required permission blocks in the example below.

   Replace *[***YOUR ACCOUNT ID***]* and *[***YOUR ACCOUNT REGION***]* with your AWS account ID and with the AWS region where you want to deploy Cloudera DataFlow, respectively.

```
        {
            "Sid": "AllowAutoscalingServiceLinkedRoleForAttachmentOfPer
sistentResources",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::[***YOUR ACCOUNT ID***]:role/aws-se
rvice-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        },
        {
            "Sid": "AllowAutoscalingServiceLinkedRoleUseOfTheCMK",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::[***YOUR ACCOUNT ID***]:role/aws-se
rvice-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
```

```
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow EKS access to EBS.",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
           ],
           "Resource": "*",
           "Condition": {
               "StringEquals": {
                   "kms:CallerAccount": "[***YOUR ACCOUNT ID***]",
                   "kms:viaService": "ec2.[***YOUR ACCOUNT REGION***].ama
zonaws.com"
               }
           }
        }
```

Important: If you fail to add these permissions, you will encounter failures when enabling Cloudera DataFlow since it will be unable to provision the necessary encrypted storage using the custom key.

2. When registering your Cloudera environment, follow these steps on the Region, Networking and Security page to assign the custom key:

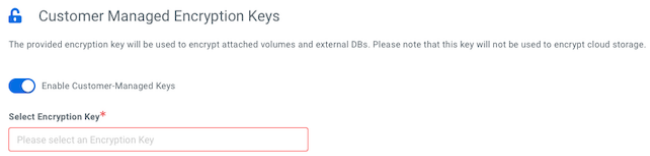

   a. Under Customer-Managed Keys, click Enable Customer-Managed Keys.
   b. Select the CMK you want to enable for this environment from the Select Encryption Key drop-down list.

   For more information on registering a Cloudera environment, see Register an AWS environment from Cloudera UI.

3. If you are also using restricted IAM policies with Cloudera, make sure you provide the KMS CMK for volume encryption when you Create the restricted policies and attach them to the cross-account role .

Cloudera DataFlow

Cloudera DataFlow Functions in Azure Functions

Date published: 2021-04-06
Date modified: 2024-12-03

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating your first Azure Function App 5

General configuration 14

Managed identity 19

Output ports 20

Parameters 20

Cold start 21

Cloud storage 22

Disk storage considerations 25

Data flow state 25

Configuring Kerberos 26

Handling failures 27

Testing your data flow 27

Testing your Azure Function App 28

Securing your Azure Function App 29

Monitoring and logs 30

Adjusting logs levels 30

Azure Function triggers.....31

Creating an Azure function using CLI.....33

Creating your first Azure Function App

Azure Functions is a serverless computer service that allows you to write less code, maintain less infrastructure, and save on costs. Instead of deploying and maintaining servers, the cloud infrastructure provides all the up-to-date resources needed to keep your applications running. In Azure Functions, a function app provides the execution context for your individual functions.

About this task

Follow these steps to create an Azure Function App that is able to run Cloudera DataFlow Functions:

Procedure

1. Navigate to the Azure Portal in your cloud account and click the Function App service.

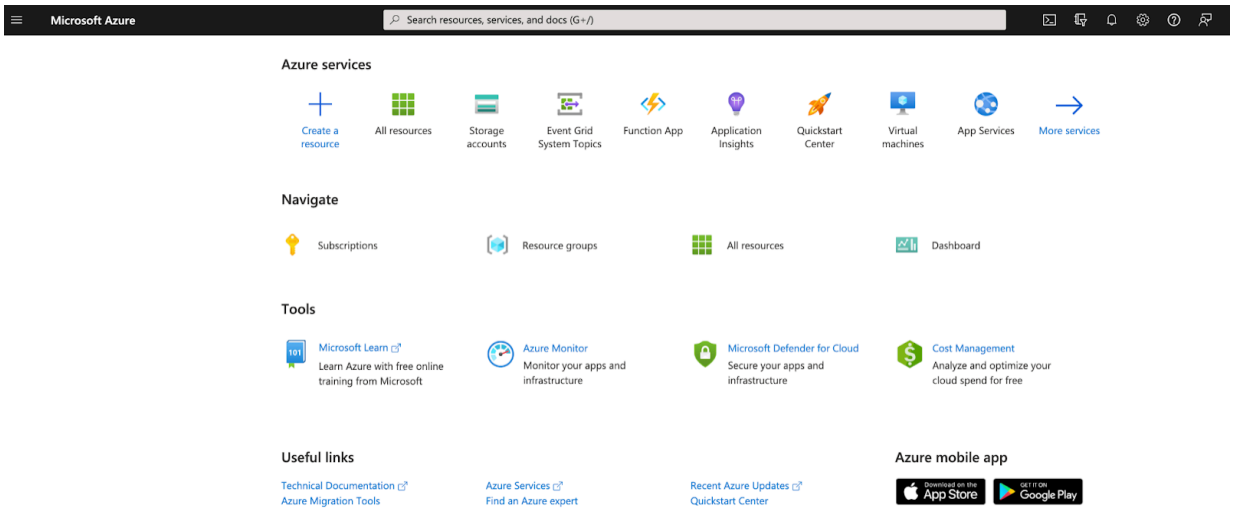

2. Click Create Function App.

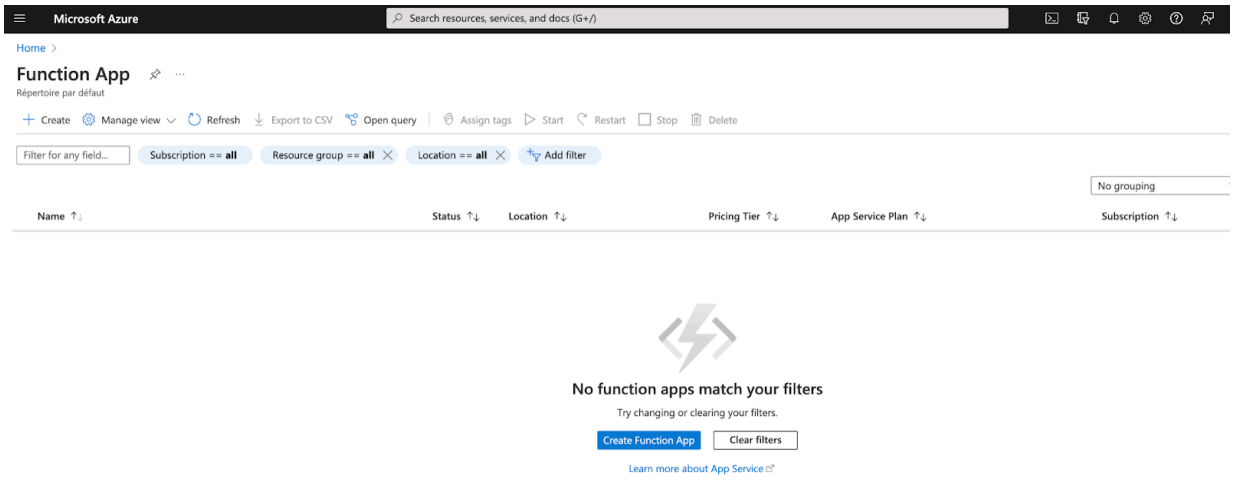

You can now create your Function App:

a) Select the Subscription and the Resource Group that you want to use.

   You can also create a new one, dedicated to this Function App.

b) Provide a name for your Function App.

c) Select Java for Runtime stack.

d) Select 11.0 for Version.

e) Select the Region where you want to deploy your Function App.

f) Select the Operating System.

   You can keep Windows, or you can choose Linux.

   Note: Choosing Linux specifically on a "Consumption" plan is not recommended for Cloudera DataFlow Functions due to a comparative lack of diagnostic, debugging, and logging support.

g) Select the Plan type that you want to use.

   Azure has multiple plan types which should be weighed depending on your use case. The default is Consumption, which means that no dedicated servers are provisioned and your Function App is automatically scaled completely based on consumption. There are many differences between these plans, including memory

limits, timeout limits and networking support. For a full comparison of the plans, see Azure Functions hosting options.

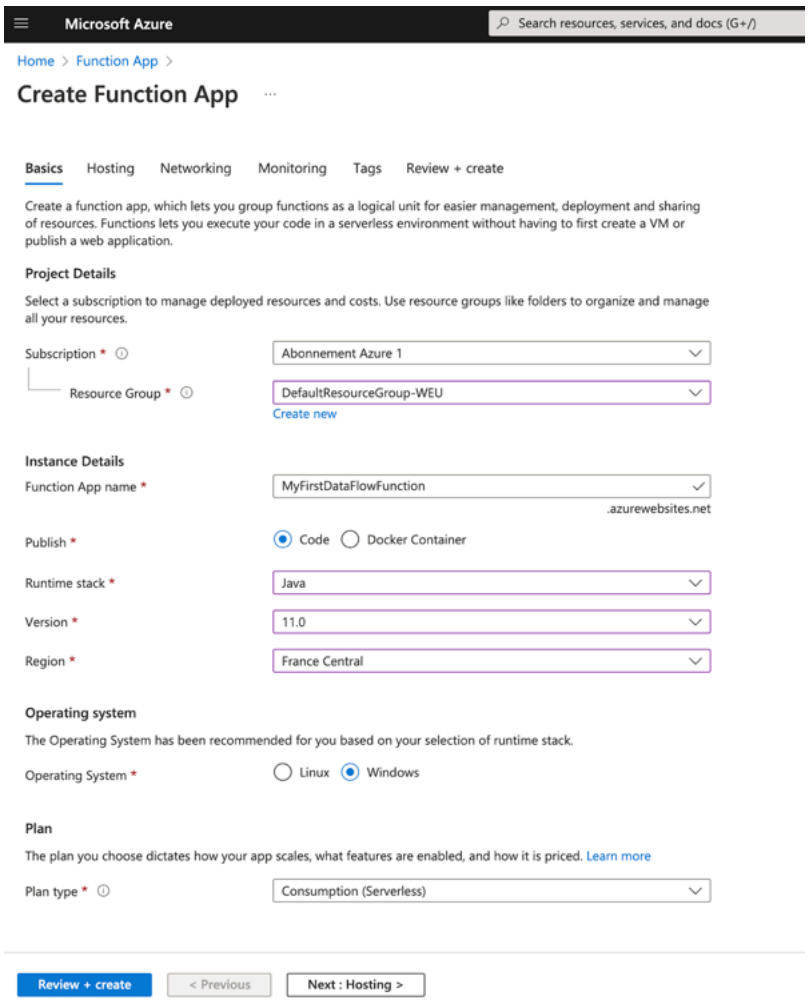

After completing the Basics tab, you can either click Next : Hosting > to provide additional details for your function app, or if you want to skip these configurations, click Review + create.

Note: You are advised to go through the additional configuration options as well, as they cannot be changed later.

3. Click Next : Hosting > to continue creating the function app in the wizard.
   You can specify an existing an existing or create a new Azure Storage account.

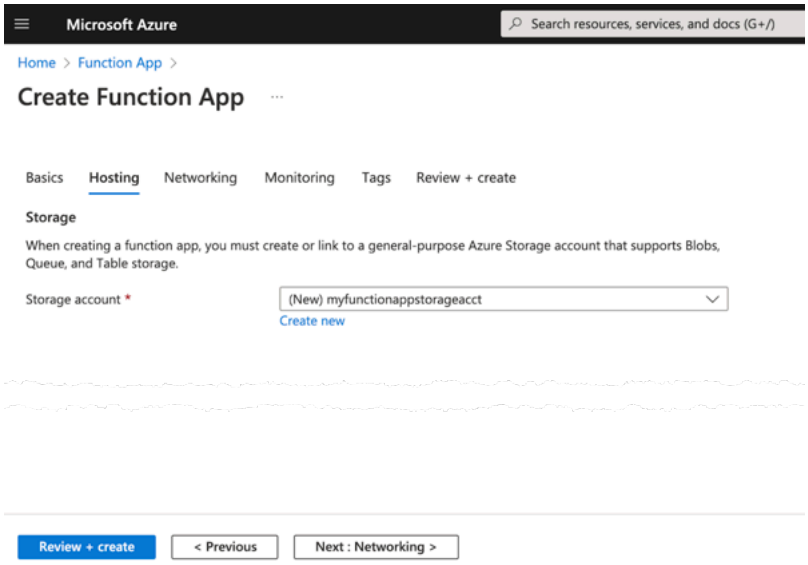

4. Click Next : Networking > to move forward in the wizard.
   Depending on the Service Plan you selected, here you may be able to enable network injection.

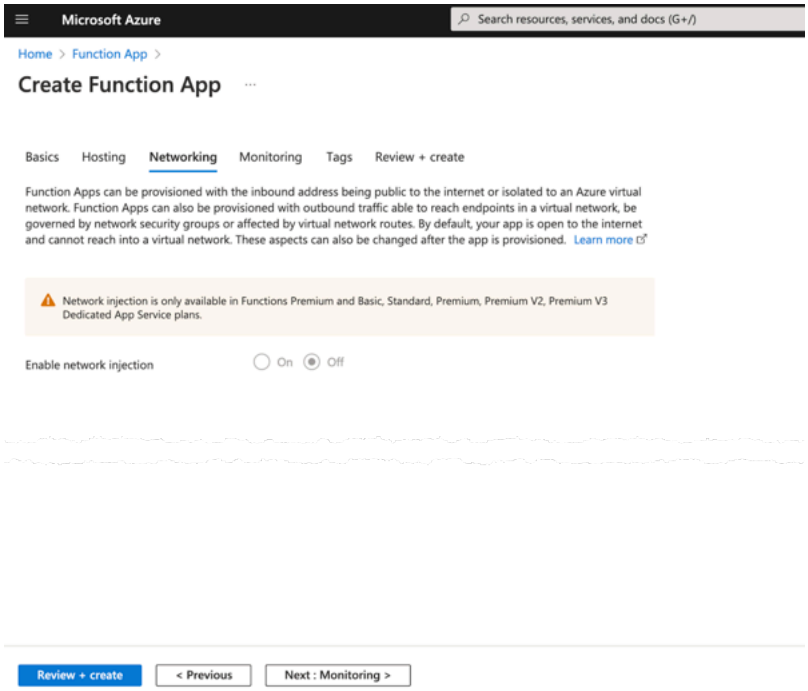

5. Click Next : Monitoring > to go to the next page.

   Application Insights will automatically be integrated unless disabled here.

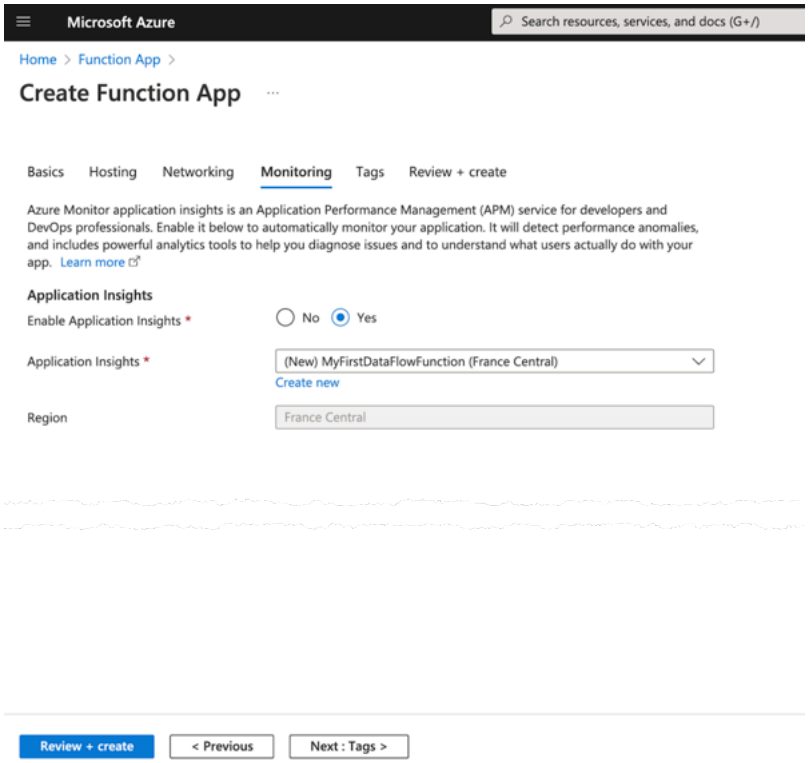

6. Click Review + create.

   At this point, you can safely skip the Tags tab and click Review + create as tags can be set later.

   You will get a summary of your Function App details.

7. Click Create to create your Function App, and wait for the deployment to complete.

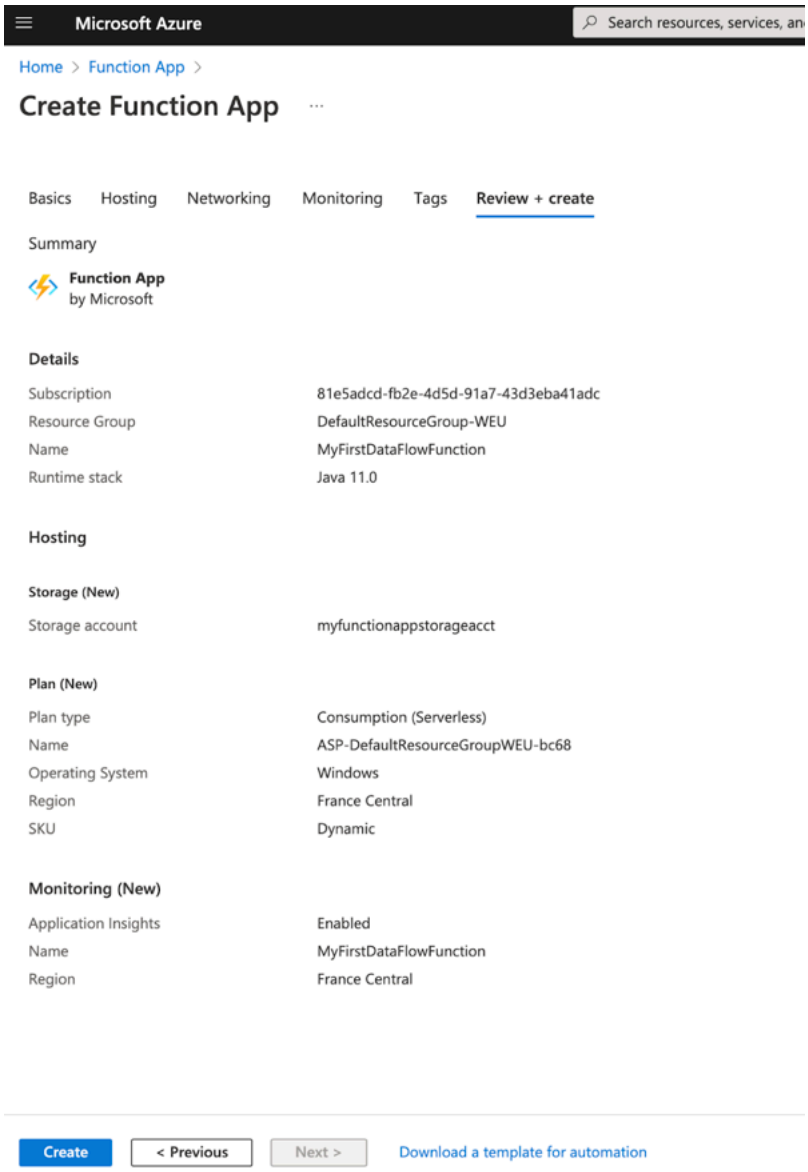

Attention:

At this point, some additional steps are required if you selected the App Service Plan Type:

a. Navigate to the storage account selected or created before, and click Access keys.
b. Click Show keys at the top, and copy the key1 "Connection string" value.
c. Go back to the Function App, click Configuration New application setting to add a new setting named WEBSITE_CONTENTAZUREFILECONNECTIONSTRING, and paste the copied value as the setting value.
d. Add another Application setting named WEBSITE_CONTENTSHARE with a value of "-fileshare".
e. Click Continue Save at the top.

This creates a file share to back your Function App that you can easily access from your storage account.

Attention:

At this point, if you selected a Consumption plan with Linux Operating System, follow these steps to upload the Cloudera DataFlow Functions binary:

a. Follow the instructions in the *Managed Identity* section to create a managed identity for your Function App.
b. Navigate to the new storage account you created, and select the Containers blade.
c. Click + Container on the top left and specify a name for a new Blob Container (e.g., "naaf").
d. Click Create.
e. Click the new container and select the Access Control (IAM) Role assignments .
f. Click Add at the top left, and specify Add role assignment.
g. Select Storage Blob Data Reader and click Next.
h. Change Assign access to to Managed identity.
i. Click the Select members link.
j. Under Managed identity, select Function App and click the name of your Function App.
k. Click Select and Review + assign.
l. Go back to the new Container, click Upload at the top left and upload the NaafAzureFunctions.zip file provided by Cloudera.

For Windows based Function App or non-Consumption plans, follow these steps to upload the Cloudera DataFlow Functions binary to your newly created Function App:

a. Navigate to the storage account selected or created before, and click the File shares blade under Data storage
b. Click the file share name.

You can see that this share has a 5 TB quota. It will be mapped to /home on the Function App filesystem.

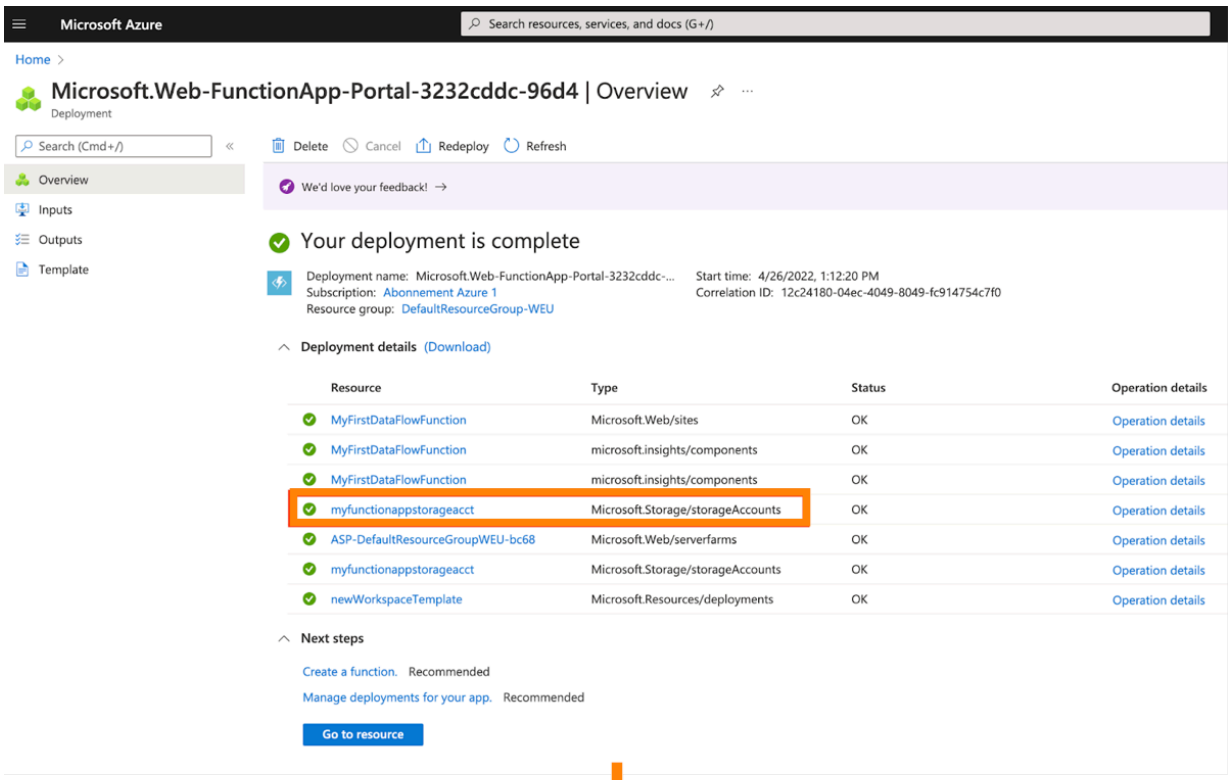
Microsoft Azure
Search resources, services, and docs (G+/)
Home >
Microsoft.Web-FunctionApp-Portal-3232cddc-96d4 | Overview
Deployment
Search (Cmd+/)
Overview
Inputs
Outputs
Template
Delete
Cancel
Redeploy
Refresh
We'd love your feedback! →
Your deployment is complete
Deployment name: Microsoft.Web-FunctionApp-Portal-3232cddc-...
Subscription: Abonnement Azure 1
Resource group: DefaultResourceGroup-WEU
Start time: 4/26/2022, 1:12:20 PM
Correlation ID: 12c24180-04ec-4049-8049-fc914754c7f0
Deployment details (Download)
Resource
Type
Status
Operation details
MyFirstDataFlowFunction
Microsoft.Web/sites
OK
Operation details
MyFirstDataFlowFunction
microsoft.insights/components
OK
Operation details
MyFirstDataFlowFunction
microsoft.insights/components
OK
Operation details
myfunctionappstorageacct
Microsoft.Storage/storageAccounts
OK
Operation details
ASP-DefaultResourceGroupWEU-bc68
Microsoft.Web/serverfarms
OK
Operation details
myfunctionappstorageacct
Microsoft.Storage/storageAccounts
OK
Operation details
newWorkspaceTemplate
Microsoft.Resources/deployments
OK
Operation details
Next steps
Create a function. Recommended
Manage deployments for your app. Recommended
Go to resource

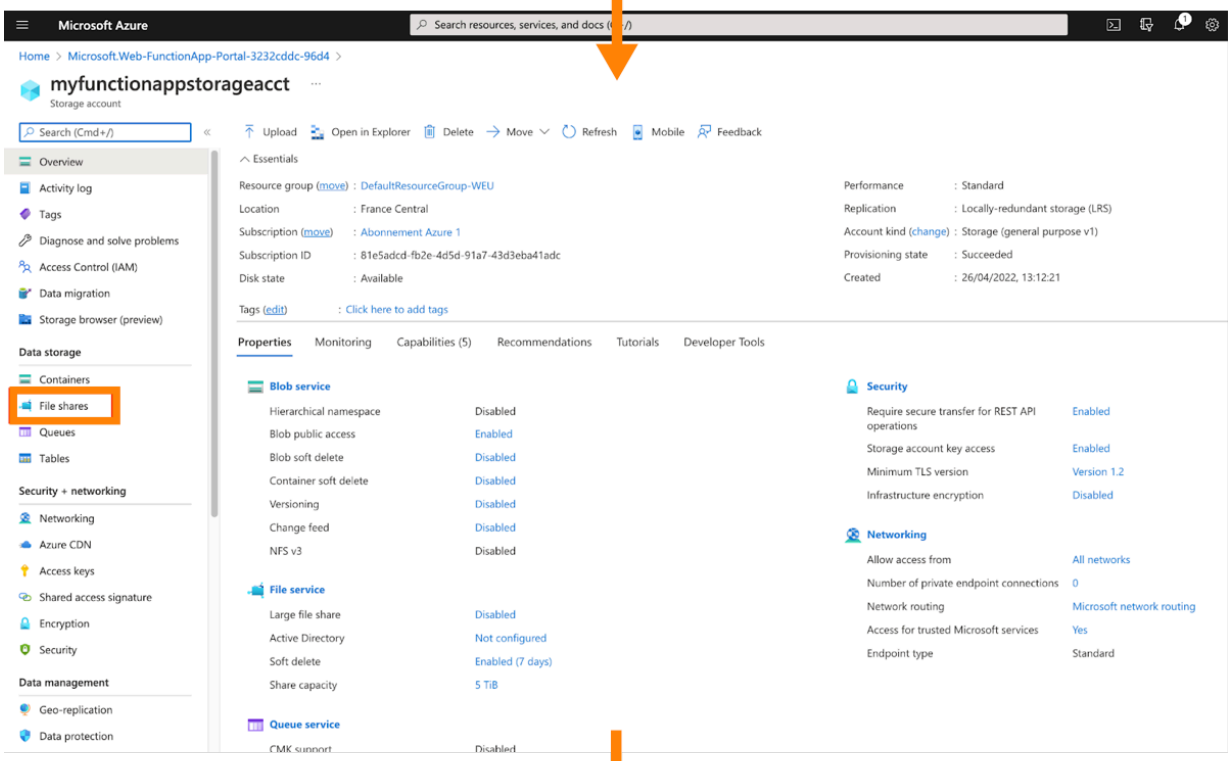
Microsoft Azure
Home > Microsoft.Web-FunctionApp-Portal-3232cddc-96d4 >
myfunctionappstorageacct
Storage account
Search (Cmd+/)
Upload
Open in Explorer
Delete
Move
Refresh
Mobile
Feedback
Overview
Activity log
Tags
Diagnose and solve problems
Access Control (IAM)
Data migration
Storage browser (preview)
Data storage
Containers
File shares
Queues
Tables
Security + networking
Networking
Azure CDN
Access keys
Shared access signature
Encryption
Security
Data management
Geo-replication
Data protection
Essentials
Resource group (move) : DefaultResourceGroup-WEU
Location : France Central
Subscription (move) : Abonnement Azure 1
Subscription ID : 81e5adcd-fb2e-4d5d-91a7-43d3eba41adc
Disk state : Available
Tags (edit) : Click here to add tags
Performance : Standard
Replication : Locally-redundant storage (LRS)
Account kind (change) : Storage (general purpose v1)
Provisioning state : Succeeded
Created : 26/04/2022, 13:12:21
Properties
Monitoring
Capabilities (5)
Recommendations
Tutorials
Developer Tools
Blob service
Hierarchical namespace Disabled
Blob public access Enabled
Blob soft delete Disabled
Container soft delete Disabled
Versioning Disabled
Change feed Disabled
NFS v3 Disabled
File service
Large file share Disabled
Active Directory Not configured
Soft delete Enabled (7 days)
Share capacity 5 TiB
Queue service
Security
Require secure transfer for REST API operations Enabled
Storage account key access Enabled
Minimum TLS version Version 1.2
Infrastructure encryption Disabled
Networking
Allow access from All networks
Number of private endpoint connections 0
Network routing Microsoft network routing
Access for trusted Microsoft services Yes
Endpoint type Standard

Microsoft Azure
Search resources, services, and docs (G+/)
Home > Microsoft.Web-FunctionApp-Portal-3232cddc-96d4 > myfunctionappstorageacct
myfunctionappstorageacct | File shares
Storage account
Search (Cmd+/)
File share
Refresh
Overview
Activity log
Tags
Diagnose and solve problems
Access Control (IAM)
Data migration
Storage browser (preview)
File share settings
Active Directory: Not configured
Soft delete: 7 days
Maximum capacity: 5 TiB
Security: Maximum compatibility
Search file shares by prefix (case-sensitive)
Show deleted shares
Name
Modified
Quota
myfirstdataflowfunctiona25a
4/26/2022, 1:12:58 PM
5 TiB

c. Click the data directory (or create it if it does not yet exist), and click Add directory at the top to create a directory named SitePackages.

d. Enter the new directory and upload the NaafAzureFunctions.zip file.

e. Create a file named packagename.txt with the contents of NaafAzureFunctions.zip and upload it as well.

```
% echo    "NaafAzureFunctions.zip" > packagename.txt
```

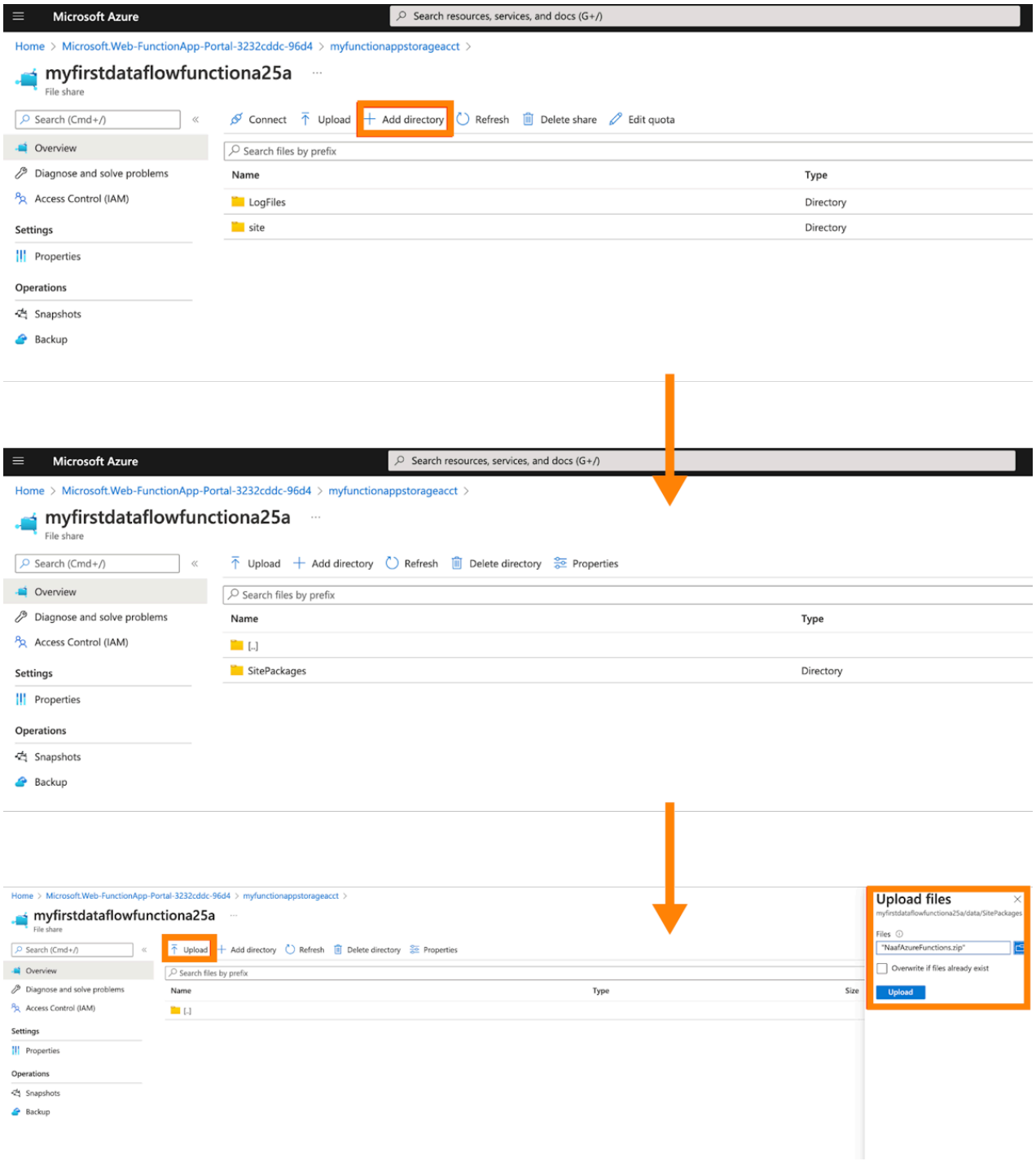

8. The Function app is now created and the binary is in place, but you still need to configure the Application Settings to complete the deployment.

See the *General configuration* section for configuration details.

Related Information

General configuration

General configuration

You must configure the function to specify the dataflow to run and add any necessary runtime configuration using environment variables.

Procedure

1. Click Configuration under Settings on the left of your Function App screen.
2. Select the Application Settings tab.

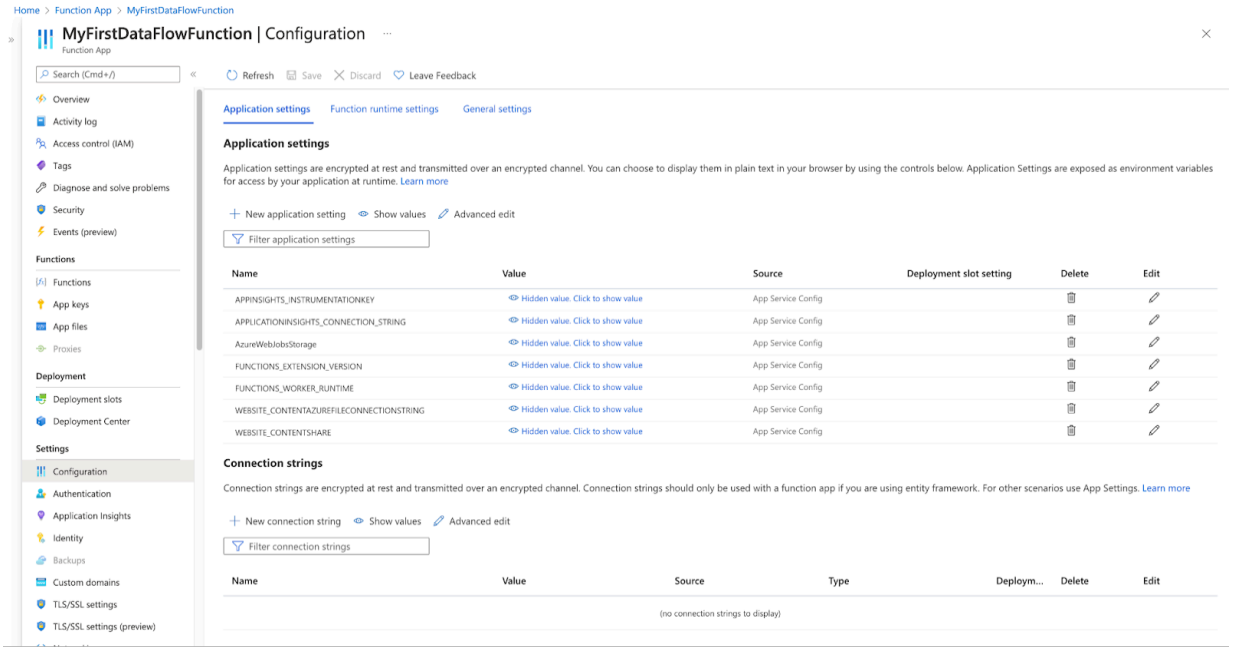

Application settings are exposed as Environment Variables inside the function. Azure pre-populates some of these, which are used by the Function framework itself. You have to add the remaining Application Settings.

3. Click New application setting at the top to add the first setting, which is required for all Cloudera DataFlow Functions deployments.

4. As a name, enter WEBSITE_RUN_FROM_PACKAGE.

   The value is different depending on the Operating System:

   - For Windows, enter 1. This tells the Function App to use the Zip package you uploaded earlier to the File share.
   - For Linux, enter the full path to the NaafAzureFunctions.zip uploaded to the Container.

     a. To get the URL, navigate to the Storage account and click Containers.
     b. Click the container you created before.
     c. Click the NaafAzureFunctions.zip file and click the Copy icon next to the URL.
     d. Back in the Function app, paste the value as the value of the WEBSITE_RUN_FROM_PACKAGE setting.

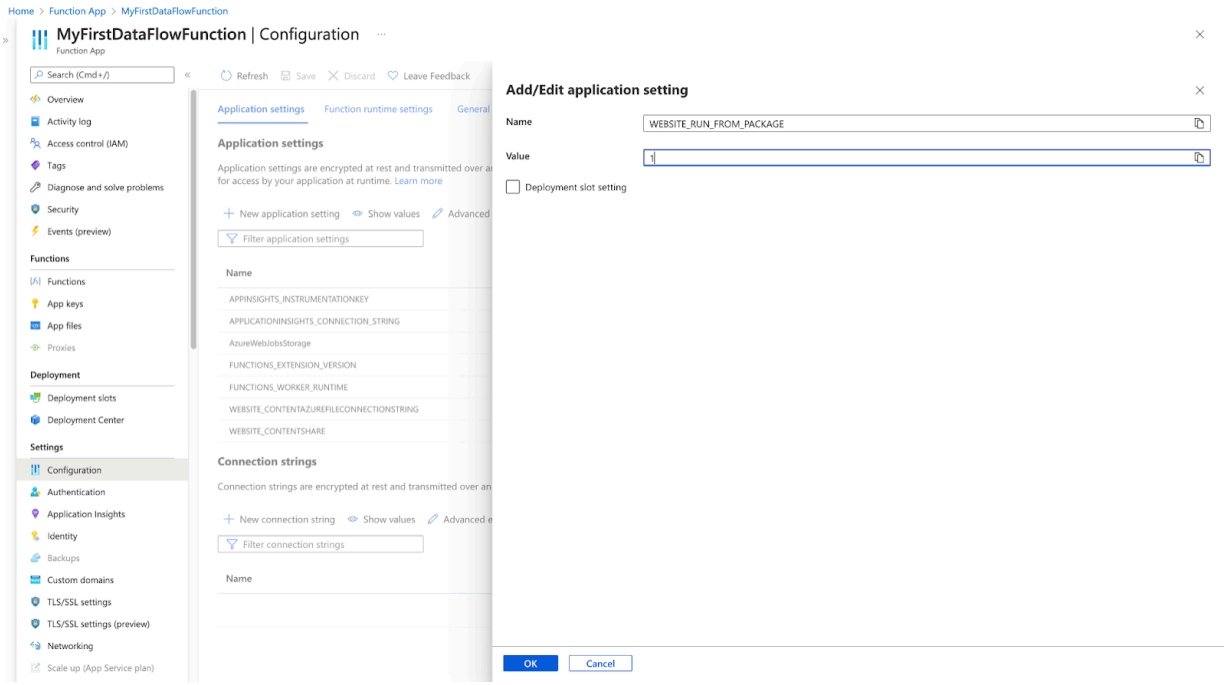

Additionally, the following Application Settings are supported.

Variable Name	Description	Required	Default Value
FUNCTION_NAME	The unique function name for this deployment. It is recommended to simply use the name of the Function App. **Warning:** If more than one Function App has the same value for this variable, data flow state may be corrupted.	true	--
FLOW_CRN	The Cloudera Resource Name (CRN) for the data flow that is to be run. The data flow must be stored in the Cloudera DataFlow Catalog. This CRN should indicate the specific version of the data flow and as such will end with some suffix such as /v.1. For more information, see *Retrieving data flow CRN*.	true	--

Variable Name	Description	Required	Default Value
DF_PRIVATE_KEY	The Private Key for accessing the Cloudera DataFlow service. The Private Key and Access Key are used to authenticate with the Cloudera DataFlow Service and they must provide the necessary authorizations to access the specified data flow. For more information, see *Provisioning Access Key ID and Private Key*. **Note:** It is a best practice to configure the DF_PRIVATE_KEY as a Secret in Key Vault.	true	--
DF_ACCESS_KEY	The Access Key for accessing the Cloudera DataFlow service. The Private Key and Access Key are used to authenticate with the Cloudera DataFlow Service and they must provide the necessary authorizations to access the specified data flow. For more information, see *Provisioning Access Key ID and Private Key*.	true	--
INPUT_PORT	The name of the Input Port to use. If the specified data flow has more than one Input Port at the root group level, this environment variable must be specified, indicating the name of the Input Port to queue up the Cloud Function event. If there is only one Input Port, this variable is not required. If it is specified, it must properly match the name of the Input Port.	false	--
OUTPUT_PORT	The name of the Output Port to retrieve the results from. If no Output Port exists, the variable does not need to be specified and no data will be returned. If at least one Output Port exists in the data flow, this variable can be used to determine the name of the Output Port whose data will be sent along as the output of the function. For more information on how the appropriate Output Port is determined, see *Output ports*.	false	--

Variable Name	Description	Required	Default Value
FAILURE_PORTS	A comma-separated list of Output Ports that exist at the root group level of the data flow. If any FlowFile is sent to one of these Output Ports, the function invocation is considered a failure. For more information, see *Output ports*.	false	--
DF_SERVICE_URL	The Base URL for the Cloudera Dataflow Service.	false	https://api.us-west-1.cdp.cloudera.com/
NEXUS_URL	The Base URL for a Nexus Repository for downloading any NiFi Archives (NARs) needed for running the data flow.	false	https://repository.cloudera.com/artifactory/cloudera-repos/
CONTENT_REPO	The contents of the FlowFiles can be stored either in memory, on the JVM heap, or on disk. If this environment variable is specified, it specifies the path to a directory where the content should be stored. If not specified, the content is held in memory.	false	--
WORKING_DIR	The working directory, where NAR files will be expanded.	false	/tmp/extensions for Linux and C:\local\Temp for Windows
EXTENSIONS_DOWNLOAD_DIR	The directory to which missing extensions / NiFi Archives (NARs) will be downloaded. For more information, see *Providing custom extensions / NARs*.	false	/tmp/extensions for Linux and C:\local\Temp for Windows
EXTENSIONS_DIR_*	Specifies read-only directories that may contain custom extensions / NiFi Archives (NARs). For more information, see the *Providing custom extensions / NARs*.	false	--
STORAGE_ENDPOINT	A Storage account endpoint to look for custom extensions / NiFi Archives (NARs) and resources. For more information, see *Cloud storage*.	false	--
STORAGE_CONTAINER	A Storage account container name in which to look in for custom extensions / NiFi Archives (NARs) and resources. For more information, see *Cloud storage*.	false	--

Variable Name	Description	Required	Default Value
STORAGE_EXTENSIONS_DIRECTORY	The directory in the Storage account Blob container to look in for custom extensions / NiFi Archives (NARs). For more information, see *Cloud storage*.	false	extensions
STORAGE_RESOURCES_DIRECTORY	The directory in the Storage account Blob container to look in for custom resources. For more information, see *Cloud storage*.	false	resources
COSMOSDB_ENDPOINT	The Cosmos DB account URI for the Cosmos DB state provider. It is required if you wish to preserve state in any stateful processors. For more information, see *Data flow state*.	false	--
COSMOSDB_STATE_DATABASE	The Cosmos DB database name for the Cosmos DB state provider. For more information, see *Data flow state*.	false	nifi_state
COSMOSDB_STATE_CONTAINER	The Cosmos DB container name for the Cosmos DB state provider. For more information, see *Data flow state*.	false	nifi_state
WEBSITE_RUN_FROM_PACKAGE	Specifies that the Function App should use the NaaF Zip package as its binary distribution.	true	1
KRB5_FILE	It specifies the filename of the krb5.conf file. This is necessary only if connecting to a Kerberos-protected endpoint. For more information, see *Configuring Kerberos*.	false	/etc/krb5.conf

The following environment variables apply only to HTTP triggers:

Variable Name	Description	Required	Default Value
HEADER_ATTRIBUTE_PATTERN	A Regular Expression capturing all flowfile attributes from the Output Port that should be returned as HTTP headers, if the trigger type is HTTP. For all other trigger types, this variable is ignored.	false	--

Variable Name	Description	Required	Default Value
HTTP_STATUS_CODE_ATTRIBUTE	A flowfile attribute name that will set the response HTTP status code in a successful data flow, if the trigger type is HTTP. For all other trigger types, this variable is ignored. See the section on *HTTP Functions* for more information.	false	--

Related Information

Retrieving data flow CRN
Provisioning Access Key ID and Private Key
Output ports
Providing custom extensions / NARs
Cloud storage
Data flow state
Configuring Kerberos
HTTP Functions

Managed identity

A Function App can use a Managed Identity in order to access other Azure services.

Procedure

1. When configuring your Function App, click the Identity blade on the left.
2. Switch the Status to On and click Save.
3. Click Yes to create a Managed Identity to represent your Function App.

   Your Function App now has an identity in Azure Active Directory, and can be assigned roles and access to specific services like other identities.

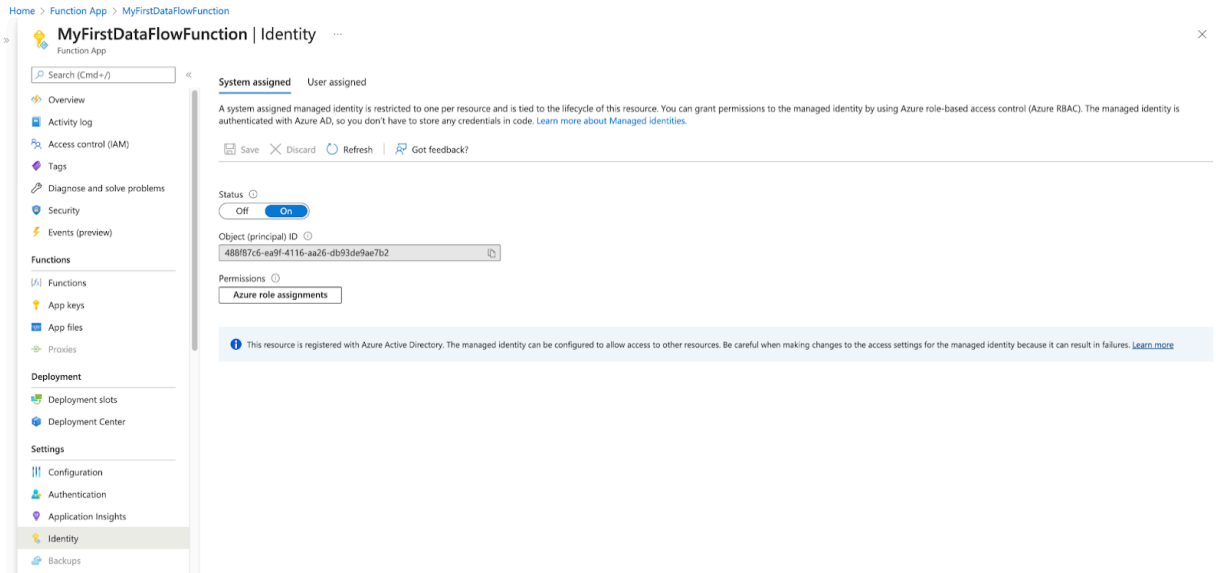

When interacting with Key Vaults, Event Grid, or Storage accounts (besides the File share automatically attached to the Function App), this is the Identity you will use, and it is named after your Function App. You can also grant it permissions to access other Azure services if the data flow itself requires access to them.

Output ports

While all Function trigger types support Failure ports, only HTTP Functions will use the output from non-failure Output Ports. This topic collects different considerations of Output Ports for the different Function trigger types.

Event Grid Function Output Ports

With Event Grid Functions, the Output Port data flow logic is simpler.

If one Output Port is present, and its name is "failure", any data routed to that Output Port will cause the function invocation to fail.

If two Output Ports are present, and one of them is named "failure", that Output Port will behave as the "failure" port above. Any other non-failure Output Port is considered a successful invocation, but the function will do nothing with data routed here.

Any port with the name "failure" (ignoring case) will be considered a Failure Port. Additionally, if the *FAILURE_PORTS* environment variable is specified, any port whose name is in the comma-separated list will be considered a Failure Port.

HTTP Function Output Ports

Since Functions with an HTTP trigger are invoked synchronously, they directly return a response. The Output Port semantics for this case is as follows.

If no Output Port is present in the data flow's root group, no output will be provided from the function invocation, but a default success message will be returned in the HTTP response with a status code of 200.

If one Output Port is present, and its name is "failure", any data routed to that Output Port will cause the function invocation to fail. No output will be provided as the output of the function invocation, and an error message will be returned from the HTTP response, with a status code of 500.

If two Output Ports are present, and one of them is named "failure", that Output Port will behave as the "failure" port above. The other Output Port is considered a successful invocation. When the data flow is invoked, if a single FlowFile is routed to this Port, the contents of the FlowFile will be returned as the HTTP response. If more than one FlowFile is sent to the Output Port, the data flow will be considered a failure, as there is only one HTTP response possible. A MergeContent or MergeRecord Processor may be used in order to assemble multiple FlowFiles into a single output FlowFile if necessary.

Any Port with the name "failure" (ignoring case) will be considered a Failure Port. Additionally, if the *FAILURE_PORTS* environment variable is specified, any port whose name is in the comma-separated list will be considered a Failure Port.

If two or more Output Ports, other than failure ports, are present, the *OUTPUT_PORT* environment variable must be provided. In such a case, this environment variable must match the name of an Output Port at the data flow's root group (case is NOT ignored). Anything that is routed to the specified port is considered the output of the function invocation. Anything routed to any other port is considered a failure.

Note:

In any successful response, the mime.type flowfile attribute is used as the Content-Type of the response.

Parameters

Parameterization is important when building a dataflow that is able to be run outside of the NiFi instance where it was built. NiFi allows users to build a data flow making use of Parameter Contexts. Using Parameter Contexts, you can define Processor and Controller Service properties at runtime instead of at build time. Some parameters are sensitive,

such as a password, while others are not, such as the name of a Kafka topic. The Cloud function allows you to specify these parameters through Application Settings, either directly or mapped from Secrets.

Application Settings

Any parameter can be specified using the Application Settings of the Azure Function App. When configuring the Function App, in the Configuration blade, add an application setting whose name matches the name of a parameter in your Parameter Context.

Key Vault Secret

A more secure mechanism for storing sensitive data flow parameters and other sensitive Application Settings is to use a Key Vault Secret.

1. Navigate to the Key Vaults Azure service and click the Create at the top-left.
2. Select your Function App's resource group, and name the Key Vault. Select the same region as the Function App.
3. Click Next : Access policy, and Add Access Policy.
4. Under Secret permissions, select List and Get.
5. Click the None selected link next to Select principal.
6. Assuming you have created a Managed Identity for your Function App, enter the name of your Function App and select it on the right panel.
7. Click Select Add Review + create Create .
8. Once the deployment is complete, click Go to resource.

   Here you can create any secrets to be referenced.
9. Click the Secrets blade on the left, and the Generate/Import button at the top.
10. Enter the name of the desired Application Setting.

    This does not have to match the actual name of the Application Setting or Parameter Context, and must follow the convention of only alphanumeric characters and dashes.
11. Enter the value of the Application Setting, and click Create.
12. Back in your Function App, create an Application Setting with the appropriate name (the Parameter name, or one of the supported Application Settings from the “Configuration of my Function App” section).
13. Add the following value: @Microsoft.KeyVault(VaultName={vaultName};SecretName={secretName})
14. Replace {vaultName} with the actual Key Vault name, and replace {secretName} with the name of the newly created secret.
15. Click Save.

Azure will update your configuration and then attempt to link this Application Setting to the Key Vault Secret. To see this link, you can click the Refresh button. In the Source column, you should now see Key Vault Reference and a green check mark if the Function App successfully resolved the Secret. If there is a red X, there is likely an issue with either the permissions of the Function App's managed identity or maybe the vault or secret could not be found.

Important: To use this capability for data flow parameters, the Parameter Contexts used in the flow must have names that are compatible with Azure Application Settings names. To ensure that these naming conventions are followed, Parameter Contexts and Parameters should use names that start with a letter and consist only of letters, numbers, and the underscore (_) character. For example, instead of naming a Parameter "Azure Parameter Context" use the name of AZURE_PARAMETER_CONTEXT when building a data flow.

Cold start

Cold start can be defined as the set-up time required to get a serverless application's environment up and running when it is invoked for the first time within a defined period.

What is a cold start?

The concept of cold start is very important when using Function as a Service solutions, especially if you are developing a customer-facing application that needs to operate in real time. A cold start is the first request that a new function instance handles. It happens because your function is not yet running and the cloud provider needs to provision resources and deploy your code before the processing can begin. This usually means that the processing is going to take much longer than expected.

Before execution, the cloud provider needs to:

- Provision the container that will run the code
- Initialize the container / runtime
- Initialize your function

before the trigger information can be forwarded to the function's handler.

For more information about the concept of cold start, see Understanding serverless cold start in Azure Functions.

How to avoid a cold start?

Cloud providers offer the option to always have at least one instance of the function running to completely remove the occurrence of a cold start. You can read more about this in Consumptions plans for Azure Functions.

Cold start with Cloudera DataFlow Functions

When a cold starts happens with Cloudera DataFlow Functions, the following steps are executed before the trigger payload gets processed:

- The function retrieves the flow definition from the Cloudera DataFlow Catalog.
- The NiFi Stateless runtime is initialized.
- The flow definition is parsed to list the required NAR files for executing the flow.
- The required NAR files are downloaded from the specified repository unless the NARs are made available in the local storage attached to the function.
- The NARs are loaded into the JVM.
- The flow is initialized and components are started.

The step that might take a significant amount of time is the download of the NAR files. In order to reduce the time required for this step as much as possible, the best option is to list the NAR files required for the flow definition and make those files available to the function.

For the related instructions, see Cloud storage on page 22.

Cloud storage

Providing custom extensions / NARs

Cloudera Maven Repository

The binary that is deployed for running the Google Cloud function does not include the full NiFi deployment with all NARs / extensions, as it would result in very long startup times and would require more expensive configurations for the Cloud function. Instead, it packages only the NARs necessary to run Stateless NiFi. If any other extension is needed to run the data flow, it is automatically downloaded from the Cloudera Maven Repository by default when the Function App is initialized on a cold start.

Nexus repository

You can configure an alternative Nexus repository by setting the *NEXUS_URL* environment variable to the URL of the Nexus Repository where extensions should be downloaded. For example,

to use Google's mirror of the Maven Central repository, set the *NEXUS_URL* environment variable to: https://maven-central.storage-download.googleapis.com/maven2

Note: Any configured URL must either be accessible via http or be served over https with a trusted certificate.

Providing extensions from file share

This process is different depending on the Plan Type and Operating System. For every type except Linux on a Consumption plan, the file share is already created, and you can simply create a directory in the file share containing all NAR files that are needed.

For Linux on a Consumption plan, you need to create a file share and mount it as follows:

1. In the storage account attached to the Function App, click the File shares blade, and then click + File share on the top left.
2. Name the File share, and click Create.
3. Click the Access keys blade of the Storage account, and then click Show keys.
4. Copy the Key from key1. You will need to paste it in the place of [Copied access key].
5. Click the Cloud Shell icon at the top-right, and accept any prompts needed to set up the shell.
6. When the shell is running, run the following command:

```
az webapp config storage-account add      \
   --resource-group [Resource Group Name] \
   --name [Function App Name]             \
   --custom-id [Pick a unique mount ID]   \
   --storage-type AzureFiles              \
   --share-name [File share Name]         \
   --account-name [Storage account Name]  \
   --mount-path /data                     \
   --access-key [Copied access key]
```

The mount path can be different, but should not be /home or /tmp.

Now you can create a directory in the File share, and upload any NAR files to the directory.

Regardless of OS or plan type, you need to tell the Function App where to look for the extensions. You can do this by adding another Application Setting whose name starts with EXTENSIONS_DIR_. You can add multiple of these, such as EXTENSIONS_DIR_1, EXTENSIONS_DIR_2, EXTENSIONS_DIR_MY_CUSTOM_LIBS. The name of the Application Setting is the path where the Function App should look for the extensions.

- In Linux Consumption, the File share is mounted where you specified it in the Cloud Shell. For example, if you upload the NARs to the File share under a directory called extensions/custom, and the File share is mounted under /data, you can add an Application Setting called EXTENSIONS_DIR_CUSTOM with a value of /data/extensions/custom.
- In all other plans, since the File share is mounted at /home, this should always start with /home. For example, if you upload our NARs to the File share under a directory called extensions/custom, you could add an Application Setting called EXTENSIONS_DIR_CUSTOM with a value of /home/extensions/custom.

Adding many extensions will result in the Function App configuration taking a bit longer to load on a cold start, and it takes more memory to load the additional classes. So it is not recommended to upload an extremely large set of files if they are not needed.

Providing extensions / resources with storage containers

If extensions or resources are to be shared among many Function Apps, it is easier to have a single Container in a Storage account with one copy of the extensions or resources. To integrate with an arbitrary Storage Container, specify the Storage account's endpoint in the STORAGE_ENDPOINT Application Setting (e.g., https://<storageAcount>.blob.core.windows.net), and the Container name in the STORAGE_CONTAINER Application Setting.

Your Function App will need permission to access the container. Assuming you have created a Managed Identity for your Function App:

1. Edit your Function App and click the Identity blade on the left.
2. Click the Azure role assignments button under Permissions, and click Add role assignment.
3. Under Scope, select Storage, and then select the desired Storage account under Resource.

   Storage Blob Data Reader should be sufficient to download Blobs.
4. Click Save.

Next, you need to upload any extensions to a directory in the Container and specify the directory name in the STORAGE_EXTENSIONS_DIRECTORY Application Setting. Upload any data flow resources to another directory in the Container and specify the directory name in the STORAGE_RESOURCES_DIRECTORY Application Setting. Now when your App Function is restarted, it will download any extensions or resources from these locations.

If you are using the Storage Container approach, you may see the following errors in the log at function startup, which can be safely ignored:

```
2022-01-21T18:36:33.976 [Error] [pool-2-thread-4] ERROR com.azure.identity.E
nvironmentCredential -
Azure Identity => ERROR in EnvironmentCredential: Missing required environme
nt variable AZURE_CLIENT_ID
2022-01-21T18:36:34.460 [Error] [pool-2-thread-4] ERROR com.azure.identity.i
mplementation.IntelliJCacheAccessor - IntelliJ Authentication not available.
 Please log in with Azure Tools for IntelliJ plugin in the IDE.
2022-01-21T18:36:34.917 [Error] [parallel-2] ERROR com.azure.identity.Enviro
nmentCredential - EnvironmentCredential authentication unavailable. Environm
ent variables are not fully configured.To mitigate this issue,
please refer to the troubleshooting guidelines here at https://aka.ms/azsdk/
net/identity/environmentcredential/troubleshoot
2022-01-21T18:36:35.779 [Error] [pool-2-thread-4] ERROR com.azure.identity.E
nvironmentCredential - Azure Identity => ERROR in EnvironmentCredential: Mis
sing required environment variable AZURE_CLIENT_ID
2022-01-21T18:36:36.142 [Error] [pool-2-thread-4] ERROR com.azure.identit
y.implementation.IntelliJCacheAccessor - IntelliJ Authentication not availab
le. Please log in with Azure Tools for IntelliJ plugin in the IDE.
2022-01-21T18:36:36.451 [Error] [parallel-2] ERROR com.azure.identity.Env
ironmentCredential - EnvironmentCredential authentication unavailable. Envir
onment variables are not fully configured.To mitigate this issue,
please refer to the troubleshooting guidelines here at https://aka.ms/azsdk/
net/identity/environmentcredential/troubleshoot
```

Providing additional resources

In addition to NiFi extensions, some data flows may also require additional resources in order to function properly. For example, a JDBC driver may be required for establishing a database connection, or a CSV file to provide data enrichment. The recommended approach for providing such resources to your Function App is using the attached Storage account's File share. To set this up, follow the instructions in the File share section.

1. Create a directory for your resources.
2. Upload all resources required for the data flow to run to this directory.

The directory will be mapped under the mapped File share directory for access by the data flow.

Because each deployment of a given data flow may need to reference files in a different location, depending on its environment, it is generally considered a best practice to parameterize any resource that needs to be accessed by the data flow. For example, if the data flow contains a DBCPConnectionPool Controller Service for accessing a database, it is recommended to use a Parameter for the "Database Driver Location(s)" property of the Controller Service. This allows each deployment to specify the location of the JDBC driver independently.

So, for example, you can set the "Database Driver Location(s)" property to a value of #{JDBC_DRIVER_LOCATION}. Then, in your Storage File share, you can create a directory called something like resources, which contains

your JDBC driver database-jdbc-driver.jar. And you can add an Application Setting named JDBC_DRIVER_LOCATION with a value of /home/resources/database-jdbc-driver.jar (or /data/resources/database-jdbc-driver.jar for Linux Consumption plans).

By taking this approach, your data flow is more reusable, as it can be deployed in many different environments.

Disk storage considerations

Azure Function Apps have two basic options for local storage used for downloading and expanding NARs.

1. Temporary storage, under C:\local\Temp - for Consumption plans, the total local disk size is limited to 500 MB, including this Temp directory, making the actual limit somewhat less than 500 MB. This limit can be increased significantly with App Service and Functions Premium plans (for example, the Basic plan starts at 11 GB). This storage is attached directly to the Function App container, and is therefore much faster than the File Share option, discussed next. This storage also requires no additional configuration, but caution must be taken not to fill up the disk. Mitigations for this are discussed below.
2. File Share storage, which is mapped to C:\home. When setting up the initial function, you already attached a File share from your Storage account, so this should already be available to you. The File share has a default limit of 5 TB, but is not locally attached to your Function App container, and so can be significantly slower than writing to the Temp directory.

Although Temporary storage is faster, Cloudera DataFlow Functions has a storage optimization that reuses downloaded and unpacked extension NARs. Therefore, the recommended approach is to use the File share for both the WORKING_DIR and EXTENSIONS_DOWNLOAD_DIR. This will incur a significant time penalty on the very first cold start, but afterwards the downloading and unpacking of NARs can be skipped, greatly reducing the function execution time.

Note: Because of the initial execution time cost, when using the File share for either directory, it is recommended that you perform a single "warming" invocation of your function in order to download and unpack all extension NARs for later reuse.

You can use the File share for storage by doing the following:

1. Navigate to your Storage account and click the File shares blade on the left. Select your file share name on the right.
2. Click the data directory (or create it if it does not yet exist).
3. Click the + Add directory button, name it extensions, and click OK. Repeat this for a directory called working.
4. In your Function App, click the Configuration blade on the left, and add (or set) an Application Setting named EXTENSIONS_DOWNLOAD_DIR with a value of C:\home\data\extensions.
5. In your Function App, click the Configuration blade on the left, and add (or set) an Application Setting named WORKING_DIR with a value of C:\home\data\working.
6. Save the configuration.
7. Optionally (but recommended), perform a single "warming" invocation of your Function App.

Data flow state

By default, if your data flow contains any stateful processors (for example, ListSFTP), this state will automatically be stored in a Cosmos database called nifi_state, though this name can be set using the *COSMOSDB_STATE_DATABASE* environment variable. To integrate with Cosmos DB, some additional steps are required. If you do not perform these steps, the data flow will still run, but the state will not be preserved.

1. In the Azure console, navigate to the Cosmos DB service.
2. Click Create.
3. Click Create under Core (SQL) - Recommended.

4. Select your Resource Group, and enter an account name.
5. All other configurations are optional, and may be different depending on your use case. While provisioned throughput is supported, it is recommended to select Serverless.
6. Click Review + Create and check the function details.
7. Click Create.
8. In your Function App, go to the Identity blade, and note the Object (principal) ID, for the following steps.
9. Click the Cloud Shell icon in the upper right and follow any initial prompts that may set up the web-based cloud shell for you.
10. Run the following commands in the cloud shell:

```
resourceGroup=<your resource group>
accountName=<your CosmosDB account name>
principalId=<your Function App Object principal ID from above>
az cosmosdb sql role assignment create --account-name $accountName \
  --resource-group $resourceGroup --scope "/" \
  --principal-id $principalId --role-definition-id 00000000-0000-0000-0
000-000000000001
az cosmosdb sql role assignment create --account-name $accountName \
  --resource-group $resourceGroup --scope "/" \
  --principal-id $principalId --role-definition-id 00000000-0000-0000-0000
-000000000002
```

11. On the Overview blade, click the Add Container.
12. Specify the database and container IDs.

    If you use nifi_state for both, the COSMOSDB_STATE_DATABASE and COSMOSDB_STATE_CONTAINER Application Settings are not needed.
13. Specify /id for the Partition Key and click OK.
14. In your Function App Configuration, add an Application Setting called COSMOSDB_ENDPOINT with the Cosmos DB URI (from the Overview page).

    If you specified any other database or container name than nifi_state, you will need to specify the COSMOSDB_STATE_DATABASE and COSMOSDB_STATE_CONTAINER Application Settings, respectively.

The Cosmos DB state provider can be disabled even if your data flow contains stateful processors by setting the DISABLE_STATE_PROVIDER Application Setting to true.

Warning: This will cause processor state to be lost between function executions.

Configuring Kerberos

Depending on the dataflow that is used, you may need to enable Kerberos authentication in order to interact with some service(s).

To enable Kerberos, you must provide an appropriate krb5.conf file. This file tells the Cloudera DataFlow Function, among other things, where the Key Distribution Center (KDC) is located and the Kerberos Realm to use.

Depending on the data flow that is used, you may need to enable Kerberos authentication in order to interact with some service(s). To enable Kerberos, you must provide an appropriate krb5.conf file. This file tells the Cloudera DataFlow Function, among other things, where the Key Distribution Center (KDC) is located and the Kerberos Realm to use.

To specify this, the krb5.conf must be made available to Cloudera DataFlow Function. You can do this by using an Azure Storage Container. See the corresponding section to learn more about this.

Additionally, you must tell Cloudera DataFlow Function where it can find this krb5.conf file. To do this, add an environment variable to your Lambda function. The environment variable must have the name KRB5_FILE and the value must be the fully qualified filename of the krb5.conf file. For example, it might use /home/resources/krb5.conf.

In order to use Keytabs in your data flow, it will also be important to provide the necessary keytab files to the Cloudera DataFlow Function. This is accomplished in the same manner, using a Lambda Layer or an S3 bucket. While the krb5.conf file is a system configuration, the keytab is not. The keytab will be configured in the data flow on a per-component basis. For example, one Processor may use keytab nifi-keytab-1 while some Controller Service makes use of keytab hive-keytab-4. The location of these files may change from deployment to deployment. For example, for an on-premises deployment, keytabs may be stored in /etc/security/keytabs while in Cloudera DataFlow Function, they may be made available in /home/resources.

So it is better to use Parameter Contexts in order to parameterize the location of the keytab files. This is a good practice for any file-based resource that will need to be made available to a deployment. In this case, because there is a parameter referencing the location, we need to provide Cloudera DataFlow Function with a value for that parameter. For more information on this, see *Parameters*.

Related Information

Parameters

Handling failures

You can use a failure Output Port to notify the Function that an invocation has failed.

You can use a failure Output Port to notify the Function that an invocation has failed. For HTTP triggers, there is no retry. For Event Grid triggers, Azure Event Grid may automatically retry the invocation, if configured in the Event Subscription, for a default of 30 attempts. After that, if it still fails, the Event Subscription will simply drop the event notification. This could result in data loss, as it means that our Function App will not have a chance to process the data again.

Testing your data flow

It is important to test the data flow you have built before trying to deploy it.

Any data flow that is to be used by Azure Function Apps requires an Input Port and most often contains at least one failure Output Port. You can simply move up to the Parent Process Group within NiFi and create a GenerateFlowFile processor. You can then set the "Custom Text" property to any value that you may wish to feed into the data flow. For example, to simulate an Event Grid event indicating that data was added to a Storage Container:

1. Set the "Custom Text" property to the value:

```
{
  "topic": "/subscriptions/73db9d72-863b-45e9-9a1a-890cff0ac0e6/resourceGr
oups/my-rg/providers/Microsoft.Storage/storageAccounts/mystorageaccount",
  "subject": "/blobServices/default/containers/my-container/blobs/input/te
stfile.json",
  "eventType": "Microsoft.Storage.BlobCreated",
  "id": "d4c93b9a-901e-0078-1dd4-d34b8d06bbad",
  "data": {
    "api": "DeleteBlob",
    "requestId": "9d4c93b9-901e-0078-1dd4-d34b8d000000",
    "eTag": "0xD9A1EBFB6C05A58",
    "contentType": "application/json",
    "contentLength": 136,
    "blobType": "BlockBlob",
    "url": "https://mystorageaccount.blob.core.windows.net/my-container/i
nput/testfile.json",
```

```
    "sequencer": "00000000000000000000000000004F50000000003e3cc87",
    "storageDiagnostics": {
      "batchId": "69a23459-7006-003d-00d4-d39e6e000000"
    }
  },
  "dataVersion": "",
  "metadataVersion": "1",
  "eventTime": "2021-11-07T12:41:55.7182885Z"
}
```

2. Connect this GenerateFlowFile processor to the Input Port of our Process Group that you will run in the Function App.

   The "Run Once" feature of NiFi makes it easy to create this FlowFile and send it to the Process Group.
3. Start the Process Group and ensure that the data processes as expected.

   If there are any issues, you can fix the data flow and trigger the GenerateFlowFile processor again, until you have properly handled the data.

When the test is successful, you are ready to download the flow from NiFi and upload the flow definition to the Cloudera DataFlow Catalog.

Testing your Azure Function App

Once you have built and verified your dataflow and uploaded it to the Cloudera DataFlow Catalog, you can create your Azure Function App.

After deploying the function, you should test if all settings have been properly configured.

1. Click the Functions blade of your Function App.
2. Select the Function you want to test.
3. Click the Code + Test blade on the left, and the Test/Run button at the top.

   This brings up a side panel to input sample data and have the data flow run. If there are problems, you can inspect the log messages to understand the issue. If the logs do not provide enough details, you may need to adjust the log levels to gain more debugging information, and view the complete Log Stream. For more information, see Monitoring and logs.

Note: At this time, the Azure Console cannot submit valid requests in the "Code + Test" view specifically for Event Grid trigger functions.

As a workaround, you can test your Function without generating an actual Event Grid event as follows:

1. Click the Get function URL button at the top of the Code + Test view, and click the Copy button next to the URL.
2. Create a file, for example, named event-grid.json with the contents of the Event Grid schema JSON you want to test.
3. Paste the URL in the place of the URL in the following curl command:

```
curl https://<app-function-name>.azurewebsites.net/runtime/webhooks/Even
tGrid?functionName=StatelessNiFiEventGridTriggerFunction&code=<code> \
     -H 'Content-Type: application/json'\
     -H 'aeg-event-type: Notification'
     --data-binary @event-grid.json
```

For more details on submitting this request, see Azure Event Grid trigger for Azure Functions.

The amount of time the cloud function takes to run depends heavily on the data flow and how many extensions it needs. Because the Cloud function may have to download some extensions from Nexus and perform initialization, a cold start may take many seconds. Even 20-30 seconds is not uncommon for a data flow with several extensions, while other data flows may complete in 10 seconds. After you run the function successfully using the Test button,

it may be helpful to run several additional iterations in order to understand how the function will perform when a cold start is not necessary. Again, this depends heavily upon the configured data flow and other services that it may interact with. A simple data flow may complete in as little as 5 milliseconds for HTTP triggers or 100 milliseconds for Event Grid triggers, while a data flow that must perform complex transformations and reach out to one or more web services or databases for enrichment may take several seconds to complete.

It is important to adjust the function's timeout configuration if the cold start takes longer than the amount of time allocated.

Related Information

Monitoring and logs

Securing your Azure Function App

This section guides you on securing the HTTP endpoint for your App.

Enforce HTTPS

1. In your Function App, click the TLS/SSL Settings blade on the left.
2. Next to HTTPS Only, click On.
3. Ensure that "Minimum TLS Version" is the latest version.
4. You can also configure the server certificate on this page.

Add an Identity Provider

By default, your HTTP Function endpoint is completely public. In order to restrict access to only authorized users, you can add an Identity Provider:

1. In the Authentication blade of your Function App, click the Add Identity Provider button.

   Several options are available. This example uses Microsoft.
2. Leave the App registration type as Create new app registration.
3. Enter the name of your Function App as the registration name.
4. Select the desired Supported account types (for example, "Any Azure AD directory - Multi-tenant")
5. Ensure Restrict access is set to Require authentication.
6. Select the desired Unauthenticated requests value.

   This example uses "HTTP 401 Unauthorized: recommended for APIs".
7. Ensure Token store is selected.
8. Click the Add.

The identity provider appears in the Authentication blade. You can follow the Quickstart link for details on how to use it to authenticate for your HTTP Function. The example presented here shows a common use case.

To access your HTTP Function endpoint programmatically, you can create a new App Registration representing the calling client:

1. In the App Registrations service, click New registration.
2. Give the registration a name representing your client (e.g., myfunction-client).
3. Select the desired Supported account types option, and click Register.
4. In the Certificates & secrets blade, select Client secrets New client secret .
5. Specify a description, and click Add.
6. Copy the secret Value, because it will not be shown again.
7. In the Overview blade, add the Directory (tenant) ID and Application (client) ID for the next step.
8. Add the Application (client) ID of your Function App's App Registration.

9. Request an access token similar to the following:

```
# <tenant-id> is the tenant ID copied above
# <client-id> is the client App Registration's "Application (client) ID"
 copied above
# <client-secret> is the client App Registration's client secret copied ab
ove
# <application-id> is your Function App's App Registration "Application
(client) ID" copied above
curl -X POST https://login.microsoftonline.com/<tenant-id>/oauth2/token -H
 "Content-Type: application/x-www-form-urlencoded" -d \
    'grant_type=client_credentials&client_id=<client-id>&client_secret=<
client-secret>&resource=<application-id>' -X POST
```

This will give you an OAuth 2 response.

10. Copy the value of the "access_token" to use below.
11. Submit a request to your HTTP function now as follows:

```
TOKEN=<access_token> # copied from above
curl https://<your-function-app>.azurewebsites.net/api/StatelessNiFiHtt
pTextTriggerFunction \
    -H "Authorization: Bearer $TOKEN" \
    ... etc.
```

Monitoring and logs

Azure Functions offers an integration with Azure Monitor Logs to monitor functions.

You can view Function logs either directly in the Log Stream blade at the Function App level, or under the Monitoring blade of the individual Function. Function output log files are also available in the Storage account File share under LogFiles/Application/Functions/Host.

Adjusting logs levels

Cloudera DataFlow Functions makes use of the SLF4J Simple Logger for logging purposes. You can update the log levels by adjusting the JVM's system properties.

To do so, you need to set the *JAVA_TOOL_OPTIONS* application setting.

- To set a log level for a given logger, you can set the JVM system property named org.slf4j.simpleLogger.log.< logger name> to the desired log level.

  For example, if you want to enable DEBUG logging for the StandardStatelessNiFiFunction to view the actual data flow JSON being used, you would set the *JAVA_TOOL_OPTIONS* environment variable to -Dorg.slf4j.simpleLo gger.log.com.cloudera.naaf.StandardStatelessNiFiFunction=DEBUG
- You can also set additional logger levels by adding multiple -D options separated by spaces. For example, to enable DEBUG logs on both the Cloudera DataFlow Functions framework and the Stateless Bootstrap class, you would set *JAVA_TOOL_OPTIONS* to a value of:

```
-Dorg.slf4j.simpleLogger.log.com.cloudera.naaf.StandardStatelessNiFiFunc
tion=DEBUG
          -Dorg.slf4j.simpleLogger.log.org.apache.nifi.stateless.bootstra
p.StatelessBootstrap=DEBUG
```

Azure Function triggers

After you fully deploy your Function App, you can go to the Functions blade to view the different Functions that are available in Cloudera DataFlow Functions.

Overview

StatelessNiFiEventGridTriggerFunction

Triggers based on an Azure EventGrid event. This allows a wide range of possible events to trigger your Function App. For more information configuring Event Grid to interact with your Function App, see the section on *Event Grid Functions*.

StatelessNiFiHttpBinaryTriggerFunction

Triggers in response to an HTTP request with a binary payload (Content-Type: application/octet-stream). See the section on *HTTP Functions* for more information on this trigger type.

StatelessNiFiHttpTextTriggerFunction

Identical to the StatelessNiFiHttpBinaryTriggerFunction, but supports text-based payloads such as JSON (Content-Type: application/json), XML (Content-Type: application/xml), and so on.

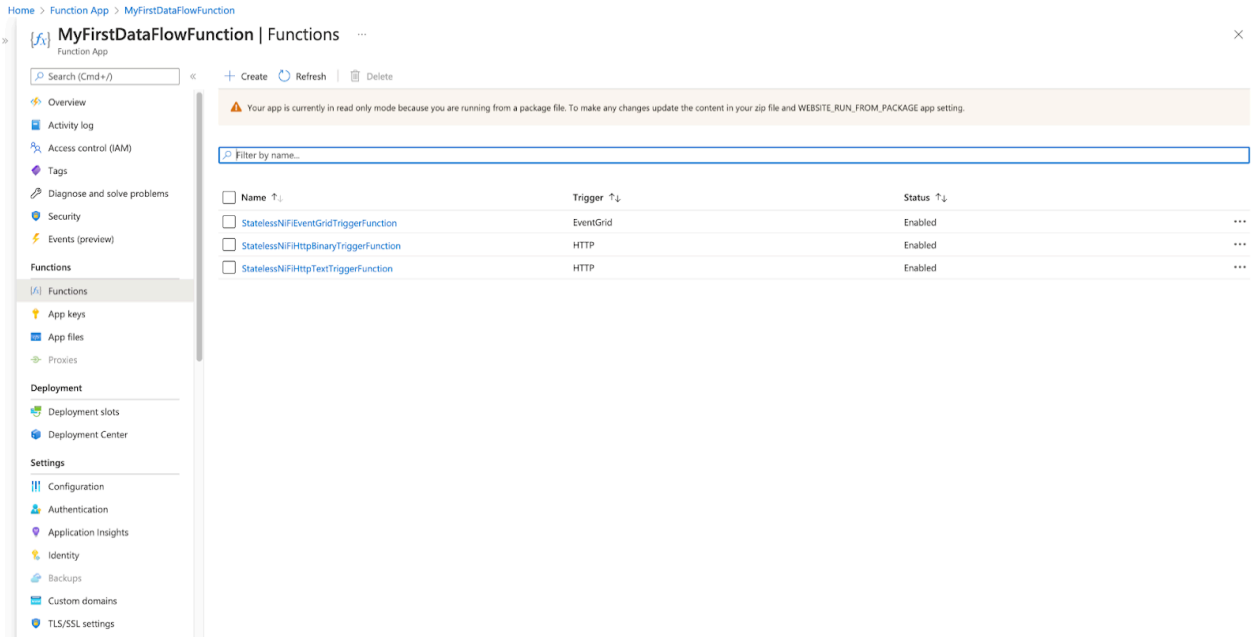

Note:

Your data flow will most likely be written differently between the two basic Function trigger types of Event Grid and HTTP, but all three Functions are enabled by default.

To disable the other functions, click the ... button at the right of the Functions table and then click Disable.

Event Grid Functions

The StatelessNiFiEventGridTriggerFunction can be used to trigger based on an Event Grid event, responding to a variety of Azure signals.

To set up an Event Grid Subscription for your Function App, follow these steps:

1. Navigate to the Event Grid System Topics service, and click Create in the top-left.

2. Select the Topic Type based on what service you want to trigger your Function App.

   For example, if you want your Function to trigger every time a file is dropped into a Storage Container, select Storage Accounts (Blob & GPv2). After that select the Subscription of the Resource you want to target, and select the Resource Group and Resource accordingly.
3. Enter a name for your System Topic, and check the Enable system assigned identity box.
4. Click Review + create, and then Create.
5. When the deployment completes, click Go to resource.

   Although you could have created an Event Grid Subscription directly, creating a System Topic to house the Subscription allows additional diagnostics to be configured, such as routing information about DeliveryFailures to a Storage account, which is particularly useful in troubleshooting. Check the Diagnostic settings blade for more information on configuring this setting.
6. On the Event Grid System Topic Overview blade, click + Event Subscription in the top-left.
7. Enter a name for the subscription.
8. Select the Event schema, which specifies the format of metadata sent as input to your NiFi data flow.

   For details on event grid properties and schema, see Azure Event Grid event schema. For details on the Cloud Event Schema, see CloudEvents v1.0 schema with Azure Event Grid.
9. Optionally, select the Event Types (for example, "Blob Created" for Storage accounts).
10. For Endpoint Details, select Azure Function, and click the Select an endpoint link.

    Your function will likely be automatically selected, but if not, you can select the appropriate Function from the right panel, and then click Confirm Selection.
11. Click Create.

You can now monitor the subscription from the System Topic, and should see any activity in the Overview blade.

HTTP Functions

With HTTP functions, the Function is invoked by submitting an HTTP request, and the Function output from the data flow is returned as the HTTP response, with specific Output Port semantics. For more information, see *HTTP Function Output Ports*.

This is an example of invoking the StatelessNiFiHttpTextTriggerFunction using curl:

```
curl https://<function-app-name>.azurewebsites.net/api/StatelessNiFiHttpText
TriggerFunction \
     -H 'Content-Type: application/json' \
     --data-binary @my-file.json
```

The StatelessNiFiHttpBinaryTriggerFunction invocation is very similar, but allows binary data to be submitted:

```
curl https://<function-app-name>.azurewebsites.net/api/StatelessNiFiHttpBina
ryTriggerFunction \
     -H 'Content-Type: application/octet-stream' \
     --data-binary @my-file.pdf
```

Note: The HTTP request body for Azure HTTP Function triggers is read completely into memory on the Heap, so it is recommended to limit the size of the data sent in the HTTP request. Microsoft's recommendation for working with large data objects is to store the objects in a Storage account, and only pass around pointers to the objects in the HTTP request body. NiFi data flows can make use of this recommendation by using the FetchAzureBlobStorage processor to retrieve the object if needed.

In any successful response, the mime.type flowfile attribute will be used as the Content-Type of the response. Conversely, the HTTP request Content-Type will be added as the mime.type attribute of the input flowfile.

There are two Application Settings that can be configured to customize the HTTP response:

HEADER_ATTRIBUTE_PATTERN

can provide a regular expression matching any flowfile attributes to be included as HTTP response headers. This means that any attributes to be included as headers should be named as the headers themselves.

HTTP_STATUS_CODE_ATTRIBUTE

specifies a flowfile attribute that sets the HTTP status code in the response. This allows a code other than 200 for 'success' or 500 for 'failure' to be returned. If this environment variable is specified but the attribute is not set, the status code in the HTTP response will default to 200.

In the "failure" scenario, the HTTP_STATUS_CODE_ATTRIBUTE is not used. Therefore, if a non-500 status code is required, it must be provided in a successful data flow path. For example, If a 409 Conflict is desired, it could be achieved as follows:

1. In the data flow, include an UpdateAttribute processor that sets an arbitrary attribute, for example http.status.code, to the value 409.
2. Optionally, add a ReplaceText processor that sets the contents of the flowfile using Always Replace to whatever HTTP response body is desired.
3. Route this part of the flow to the main Output Port used for the success case.
4. Set HTTP_STATUS_CODE_ATTRIBUTE to http.status.code.

The same method may also be used to set a non-200 success code.

All HTTP headers sent in the HTTP request will be imported as Flowfile attributes in the data flow, with an attribute name like azure.http.header.<lowercaseHeaderName>. For example, submitting a request with a header X-My-Test-Header: test will result in the flowfile having an attribute named azure.http.header.x-my-test-header with a value of test.

Creating an Azure function using CLI

Follow these steps to deploy a Cloudera DataFlow Functions using the Azure CLI.

Before you begin

- Your NiFi flow definition is stored in the Cloudera DataFlow Catalog and you have the CRN corresponding to the flow version you want to execute as a function
- You have created a Machine User with the proper role and you have its Access Key and Private Key credentials
- You have installed and configured the Azure CLI on your local machine

About this task

These steps provide an example for deploying a Function App that listens for HTTPS requests with a publicly exposed endpoint. Cloudera recommends going through the documented steps in the Azure Console UI to generate the appropriate templates and use those templates for CLI-based deployments.

Procedure

1. Create a Resource Group that you will use for the Function App. You can skip this step if you want to reuse an existing Resource Group.

```
az group create -l francecentral -n MyDFFResourceGroup
```

2. Download the template.json file available here.

```
az deployment group create --resource-group MyDFFResourceGroup \
 --template-file <path to template.json> \
 --parameters \
  subscriptionId=<existing subscription ID to use>  \
  name=<name of your FunctionApp> \
```

```
  location=<region where to deploy your Function App> \
  hostingPlanName=<name of the plan to be created> \
  resourceGroup=<name of the resource group previously created> \
  storageAccountName=<name of the storage account to be created> \
  FLOW_CRN=... \
  DF_ACCESS_KEY=... \
  DF_PRIVATE_KEY=...
```

If you have additional parameters, you can add those in the appSettings section of the template.json file.

Note: For sensitive parameters, Cloudera recommends to use Azure Key Vault as explained in the Parameters documentation. This is not covered in the provided template.

Example:

```
az deployment group create --resource-group MyDFFResourceGroup \
 --template-file /tmp/template.json \
 --parameters \
     subscriptionId=00000000-0000-0000-000000000000 \
     name=DFFResizeImage \
     location=francecentral \
     hostingPlanName=MyHostingPlanDFF \
     resourceGroup=MyDFFResourceGroup \
     storageAccountName=mydffstorageaccountname \
     FLOW_CRN=crn:cdp:df:us-west-1:00000000-0000-0000-0000-000000000000:
flow:my-flow/v.1 \
     DF_ACCESS_KEY=00000000-0000-0000-0000-000000000000 \
     DF_PRIVATE_KEY=0000000000000000000000000000000000
```

3. Retrieve the File Share name that has been created during the previous operation by executing the below command:

```
az storage share list --account-name mydffstorageaccountname
```

You can automate the retrieval with jq using the below command:

```
FILESHARE=`az storage share list --account-name mydffstorageaccountname |
 jq -r '.[0].name'`
```

4. Create the directories in the File Share:

```
az storage directory create --account-name mydffstorageaccountname --sha
re-name dffresizeimageadcb --name data
az storage directory create --account-name mydffstorageaccountname --shar
e-name dffresizeimageadcb --name data/SitePackages
```

5. Upload the binary to the File Share and create the package file:

```
az storage file upload -s dffresizeimageadcb --source /tmp/NaafAzureFunc
tions.zip --path data/SitePackages/ --account-name mydffstorageaccountname

echo "NaafAzureFunctions.zip" > /tmp/packagename.txt && az storage file
upload -s dffresizeimageadcb --source /tmp/packagename.txt --path data/S
itePackages/ --account-name mydffstorageaccountname && rm /tmp/packagena
me.txt
```

Results

Your function is now ready to be used with HTTPS triggers. See the documentation for more details. If another type of trigger is required, you need to create an Event Grid subscription as explained in Azure Function triggers.

Cloudera DataFlow

Azure Resource Planning

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Azure requirements for Cloudera DataFlow....4

Cloudera DataFlow networking in Azure....5

Subnets for Cloudera DataFlow....5
Firewall exceptions for Azure AKS....6
Azure load balancers in DataFlow....6
User defined routing....7

Limitations on Azure....8

Setting up minimum permissions....8

Azure requirements for Cloudera DataFlow

As the administrator for your Azure environment, ensure that the environment meets the requirements listed in this topic to enable the Cloudera DataFlow experience in Cloudera Public Cloud.

Follow the steps to ensure that your Azure environment meets the Cloudera and Cloudera DataFlow requirements:

Understand your Azure account requirements for Cloudera Public Cloud

- Review the *Azure subscription requirements*. The link is in the *Related information* section below.
- Verify that your Azure account for Cloudera Public Cloud has the required resources.
- Verify that you have the permissions to manage these resources.

Understand the Cloudera DataFlow requirements

Networking requirements:

- Determine your networking option:
  - Use your existing VNet and subnets
  - Have Cloudera create a new VNet and subnets
- Review firewall exceptions for Azure AKS
- Determine your load balancer option:
  - public
  - private (internal)

Verify that the following services are available in your environment for Cloudera DataFlow to use:

Azure environments used for the Cloudera DataFlow service must have the following resources available in the specific Azure region where the environment is registered. Currently, there is no cross-regional support for Cloudera DataFlow service.

- Azure Kubernetes Service (AKS)
- Azure Database for PostgreSQL
- Azure Data Lake Storage Gen2
- Virtual machine scale sets
- Dsv4-series
- Azure Availability Zones (optional)

Understand Cloudera DataFlow role requirements

- There are two Cloudera user roles associated with the Cloudera DataFlow service: DFAdmin and DFUser. Any Cloudera user with the EnvironmentAdmin (or higher) access level must assign these roles to users who require access to the Cloudera DataFlow console within their environment.

Register an Azure Environment in Cloudera

Once you have met cloud provider requirements and have created the Azure provisioning credential for Cloudera, you may proceed to register an Azure Environment.

Instructions: Register an Azure environment

Use only app-based credentials

For the Cloudera DataFlow service, you must only use an app-based credential, which requires the Contributor role to create a new service principal. For more information about creating an app-based credential for the environment you want to use for the Cloudera DataFlow service, see Create an app-based credential. If you need to change your

environment credential, see Change environment's credential. Both of these references are in the Management Console documentation.

App must have the Contributor role at the subscription level

For environments that you plan to use for the Cloudera DataFlow service, you must ensure that the application you create in Azure has the built-in Contributor Azure role at the Azure subscription level. For more information, see the description of app-based credentials in Credential options on Azure.

Created Azure app must have access to the storage account used during environment registration

Ensure that the application, which the Azure app-based credentials are attached to, must have access to the ADLS Gen2 storage location that is specified when you register the Azure environment. This is the storage location specified in Step 6 in the Register an Azure environment topic. Also see ADLS Gen2 and managed identities for information about storage accounts for Azure environments. See Minimal setup for cloud storage for further details. These references are in the Cloudera Management Console documentation.

Azure subscription should be in a similar region as the resources

Ensure that your Azure subscription is in a relatively similar region as the region where your resources are deployed. Particularly, be careful that the regions are governed by the same regulatory laws. For more information, see Azure region requirements in the Cloudera Management Console documentation. In that topic it specifies that "Cloudera Public Cloud requires that the ADLS Gen2 storage location provided during environment registration must be in the same region as the region selected for the environment." In addition, please review Azure geographies in the Microsoft documentation.

Related Information

Azure subscription requirements

Cloudera DataFlow networking in Azure

Cloudera DataFlow supports different networking options depending on how you have set up your VNet and subnets. If you want Cloudera DataFlow to use specific subnets, make sure that you specify them when registering a Cloudera environment.

Vnet and Subnet Requirements

When registering an Azure environment in Cloudera, you are asked to select a VNet and one or more subnets. Cloudera DataFlow runs in the VNet registered in Cloudera as part of your Azure environment.

You have two options:

- You use your existing VNet and subnets for provisioning Cloudera resources.
- You let Cloudera create a new VNet and subnets.

Subnets for Cloudera DataFlow

Learn about Cloudera DataFlow subnet requirements within your VNet.

Cloudera DataFlow runs in the VNet registered in Cloudera as part of your Azure environment. The Cloudera DataFlow service requires its own subnet. Cloudera DataFlow on AKS uses the Kubenet CNI plugin provided by Azure. In order to use Kubenet CNI, create multiple smaller subnets when creating an Azure environment.

Cloudera recommends the following:

- Partition the VNet with subnets that are just the right size to fit the expected maximum of nodes in the cluster.

- Use /24 CIDR for these subnets. However, if you prefer a custom range, use the following points to determine the IP addresses for the Cloudera DataFlow service:
  - The Cloudera DataFlow service can scale up to 50 compute nodes.
  - Each node consumes one IP address.
  - Additionally, you must allocate two IPs for the base infra nodes.

Firewall exceptions for Azure AKS

Learn about required firewall exceptions for maintenance tasks and management.

If you need to restrict egress traffic in Azure, then you must reserve a limited number of ports and addresses for cluster maintenance tasks including cluster provisioning. See *Control egress traffic for cluster nodes in Azure Kubernetes Service (AKS)* to prepare your Azure environment for AKS deployment.

Cloudera recommends you safelist the Azure portal URLs on your firewall or proxy server for management purposes. For more information, see *Safelist the Azure portal URLs on your firewall or proxy server*.

Related Information

Control egress traffic for cluster nodes in Azure Kubernetes Service (AKS)

Safelist the Azure portal URLs on your firewall or proxy server

Azure load balancers in DataFlow

Learn about load balancing options in Azure environments.

Azure provides a public and a private (internal) load balancer. DataFlow uses the Standard SKU for the load balancer. You can configure DataFlow to use either private or public load balancer to allow users to connect to flow deployments. By default, DataFlow provisions a private load balancer.

The figure represents a DataFlow deployed with an internal load balacer:

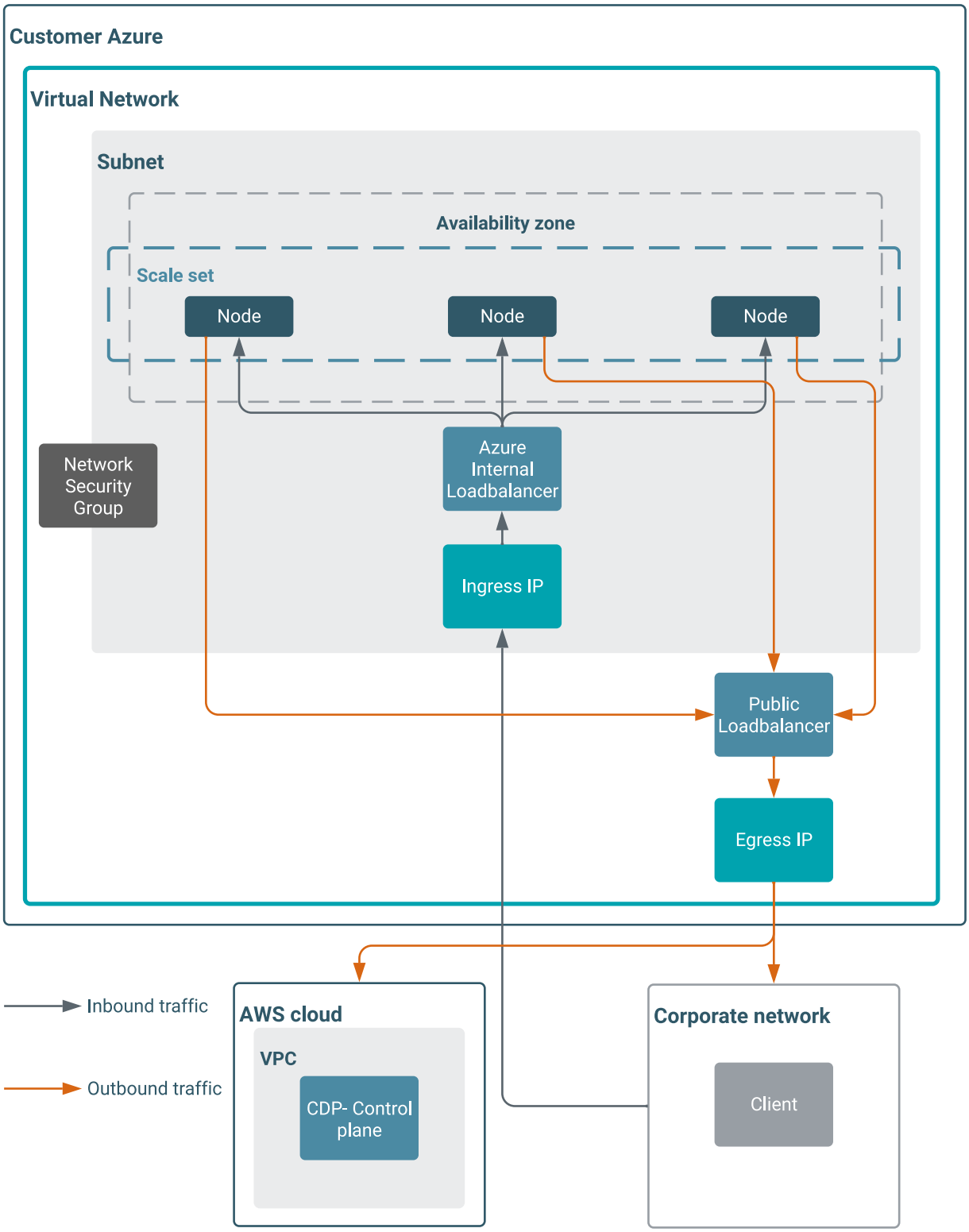

User defined routing

Azure UDR (User-Defined Routing) refers to the capability provided by Microsoft Azure to define and control the flow of network traffic within a virtual network (VNet) using custom routing rules. By default, Azure virtual

networks use system-defined routes to direct network traffic between subnets and to the internet. With UDR, you can override these default routes and specify your own routing preferences.

UDRs allow you to define specific routes for traffic to follow based on various criteria, such as source or destination IP address, protocol, port, or Next Hop type. This gives you granular control over how traffic flows within your Azure infrastructure. With UDR, you can create your own custom routes to control the path of traffic within your Azure virtual network. UDR is implemented using route tables, which contain a set of rules defining the paths for network traffic. Each subnet within a VNet can be associated with a specific route table. UDR allows you to specify the next hop for a route, which can be an IP address, a virtual appliance, or the default Azure Internet Gateway. UDR enables you to segment traffic within your virtual network, directing specific types of traffic through specific network appliances or services. UDR is commonly used in conjunction with Network Virtual Appliances (NVAs), such as firewalls, load balancers, or VPN gateways. By configuring UDR, you can ensure that traffic is properly routed through these appliances.

By leveraging UDR, you can design more complex networking architectures, implement security measures, optimize performance, and meet specific requirements for your Azure deployments.

You can enable UDR when you enable Cloudera DataFlow for an environment.

Limitations on Azure

This section lists some resource limits that Cloudera DataFlow and Azure impose on workloads running in Cloudera DataFlow workspaces.

- There is no ability to grant or revoke remote access (via Kubeconfig) to specific users. Users with the DFAdmin role in the environment can download a Kubeconfig file. The Kubeconfig file will continue to allow access even if the DFAdmin role is later revoked.
- Each Cloudera DataFlow workspace requires a separate subnet. For more information on this issue, see Use kubenet networking with your own IP address ranges in Azure Kubernetes Service (AKS).
- Heavy AKS activity can cause default API rate limits to trigger, causing throttling and eventually failures for AKS clusters. For some examples, see AKS issue 1187 and AKS issue 1413.

Setting up minimum permissions

The minimum permissions for Cloudera DataFlow on Azure govern access control between Azure resources, the Azure storage account, and Cloudera DataFlow. The minimum permissions that allow for enabling/disabling Cloudera DataFlow and deploying/undeploying flows can be set using a custom role.

Before you begin

- You have registered an application on the Azure Portal. For instructions, see *Create an app registration and assign a role to it*.
- You have created an app-based provisioning credential in your Azure subscription. For instructions, see *Create a provisioning credential for Azure*.

Procedure

1. Create a custom role that contains the minimum permissions.

   The following role definition outlines the minimum permissions required to create a custom role for Cloudera DataFlow. The permissions are listed in the Actions section, so that Cloudera DataFlow can access resources and operate correctly.

   When using the role defintiion, replace the following values:

   - *[YOUR-SUBSCRIPTION-ID]*: Your subscription ID in use.
   - *[YOUR-RESTRICTED-ROLE-NAME]*: The custom role name which is assigned to the application. For example: *CLOUDERA DATAFLOW AZURE OPERATOR FOR SINGLE RESOURCE GROUP*
   - *[YOUR-RESOURCE-GROUP-NAME]*: The original resource group name.

```
{
   "properties": {
       "roleName": [YOUR-RESTRICTED-ROLE-NAME],
       "description": "Custom restricted role for liftie",
       "isCustom": true,
       "assignableScopes": [
           "/subscriptions/[YOUR-SUBSCRIPTION-ID]/resourceGroups/[YOUR-
RESOURCE-GROUP-NAME]"
       ],
       "permissions": [
           {
               "actions": [
                    "Microsoft.ContainerService/managedClusters/read",
                    "Microsoft.ContainerService/managedClusters/write",
                    "Microsoft.ContainerService/managedClusters/agentPoo
ls/read",
                    "Microsoft.ContainerService/managedClusters/agentPools
/write",
                    "Microsoft.ContainerService/managedClusters/upgrade
Profiles/read",
                    "Microsoft.ContainerService/managedClusters/agentPoo
ls/delete",
                    "Microsoft.ContainerService/managedClusters/delete",
                    "Microsoft.ContainerService/managedClusters/accessP
rofiles/listCredential/action",
                    "Microsoft.ContainerService/managedClusters/agentPoo
ls/upgradeProfiles/read",
                    "Microsoft.Storage/storageAccounts/read",
                    "Microsoft.Storage/storageAccounts/write",
                    "Microsoft.ManagedIdentity/userAssignedIdentities/ass
ign/action",
                    "Microsoft.Compute/virtualMachineScaleSets/write",
                    "Microsoft.Network/virtualNetworks/subnets/join/acti
on",
                    "Microsoft.Network/virtualNetworks/subnets/read",
                    "Microsoft.Insights/diagnosticSettings/write",
                    "Microsoft.Insights/metrics/read",
                    "Microsoft.Insights/metricDefinitions/read",
                    "Microsoft.DBforPostgreSQL/flexibleServers/delete",
                    "Microsoft.DBforPostgreSQL/flexibleServers/read",
                    "Microsoft.DBforPostgreSQL/flexibleServers/write",
                    "Microsoft.DBforPostgreSQL/flexibleServers/configurat
ions/write",
                    "Microsoft.DBforPostgreSQL/flexibleServers/firewall
Rules/write",
                    "Microsoft.DBforPostgreSQL/flexibleServers/privateEnd
pointConnectionsApproval/action",
                    "Microsoft.DBforPostgreSQL/flexibleServers/start/actio
n",
```

```
                    "Microsoft.DBforPostgreSQL/flexibleServers/stop/acti
on",
                    "Microsoft.MarketplaceOrdering/offertypes/publishers/
offers/plans/agreements/write",
                    "Microsoft.MarketplaceOrdering/offerTypes/publishers/o
ffers/plans/agreements/read",
                    "Microsoft.ManagedIdentity/userAssignedIdentities/fede
ratedIdentityCredentials/*"
                ],
                "notActions": [],
                "dataActions": [],
                "notDataActions": []
            }
        ]
    }
}
```

2. Assign the custom role to the app registration that you earlier created on the Azure Portal. For instructions, see *Create an app registration and assign a role to it*.

Related Information

Create an app registration and assign a role to it

Create a provisioning credential for Azure

Cloudera DataFlow

Using custom processors

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Best practices for building custom components.....4

Best practices for packaging custom Python processors [Technical Preview].....5

Preparing cloud storage to deploy custom processors.....6

Best practices for building custom components

Learn about general guidelines concerning the creation of custom NiFi archives (NARs).

The goal is to build your code once against a baseline version of Apache NiFi and it can be deployed to any flow you need using any version of Cloudera DataFlow powered by any NiFi deployment equal to or greater than the version it was built against, bar major version changes.

Apache NiFi extensions are packaged in NARs. A NAR allows several components and their dependencies to be packaged together into a single package. NiFi provides Maven archetypes for creating custom processor and controller service bundle project structures. For detailed information, see the Maven Projects for Extensions Apache NiFi wiki page.

- If directly specifying the nifi-nar-maven-plugin, ensure you use the most recent version when compiling your custom code.
- If inheriting the nifi-nar-maven-plugin, Cloudera recommends that the parent version of nifi-nar-bundles has the same major and minor version as the selected NiFi runtime version. For example, If the CFM Version is 1.18.0.2.3.7.1 (NiFi Major Version: 1, NiFi Minor Version: 18) the recommended compilation version is 1.18.0 (The first two version numbers must be equal to, or less than the CFM version).

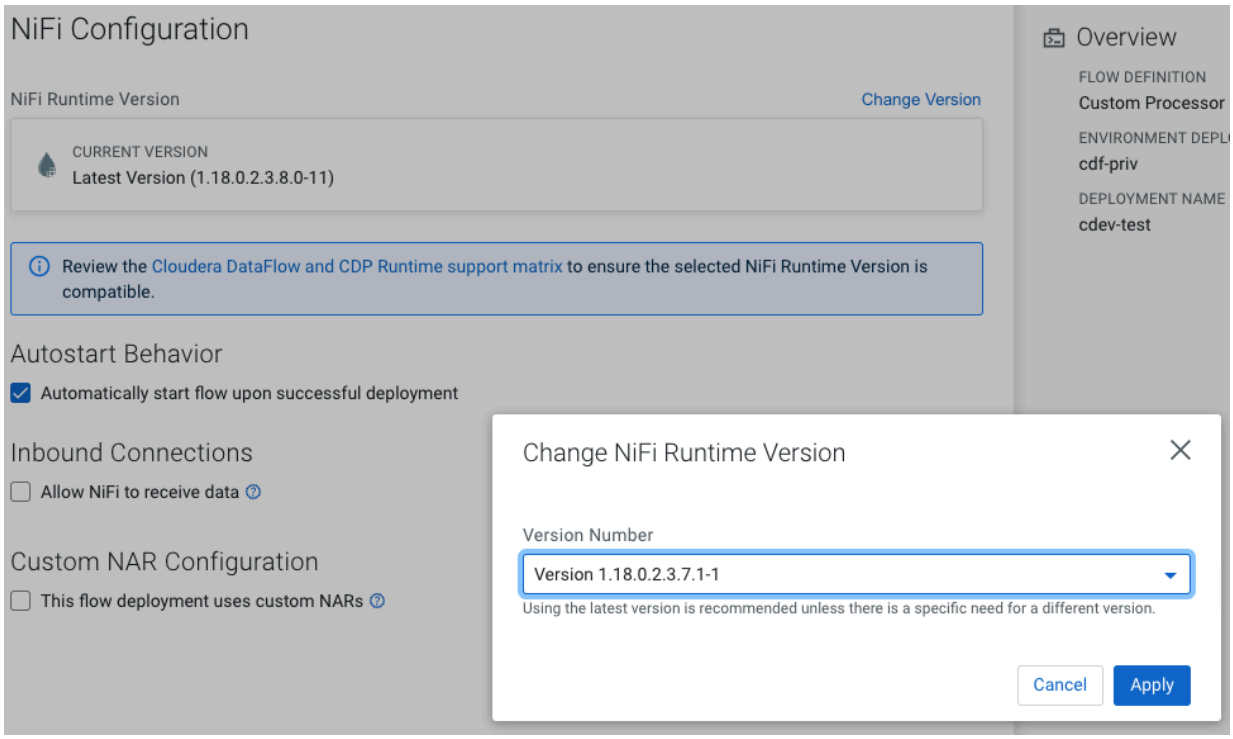


```
<parent>
       <groupId>org.apache.nifi</groupId>
       <artifactId>nifi-nar-bundles</artifactId>
       <version>1.18.0.2.3.7.1-1</version>
   </parent>
```

- Ensure your NAR pom only depends on another NAR pom for a controller service API NAR. Generally, do not extend from implementation NARs like nifi-standard-nar.
- Ensure your components jar pom marks API dependencies as provided in order to obtain them at runtime through the appropriate NAR dependency.

Best practices for packaging custom Python processors [Technical Preview]

Depending on complexity and possible shared dependencies, you need to decide whether to create your custom processor as a single file or as a package.

This documentation describes your options on packaging your custom Python processor and making it available for flow deployments in Cloudera DataFlow, and is based on the official Apache NiFi 2 documentation. For additional best practices on writing a custom Python processor for a NiFi 2.x flow, consult the official Apache NiFi documentation. Python processors can be packaged either as a single Python file, or as a Python package.

Single Python File

If the processor is simple and does not share dependencies with any other custom processor, it is easiest to have a single Python file named after the processor, like CreateFlowFile.py. In the single Python file format dependencies are specified directly in the processor.

For example:

```
class PandaProcessor(FlowFileTransform)
    class Java:
        implements = ['org.apache.nifi.pythonprocessor.FlowFileTransform']
    class ProcessorDetails:
        version = '0.0.1-SNAPSHOT',
        dependencies = ['pandas', 'numpy==1.20.0']
```

Python package

If more than one custom Python processor uses the same dependencies, or if you have a helper module that you want to use in one or more Python processors, a Python package is required. Structure your code as follows:

```
my-python-package/
#
## __init__.py
#
## ProcessorA.py
#
## ProcessorB.py
#
## HelperModule.py
#
## requirements.txt
```

In this example, all requirements across the processors and helper modules appear in requirements.txt, and both ProcessorA and ProcessorB can reference code in the helper module in a way similar to the following:

```
from HelperModule import my_helper_function
```

When uploading a Python package to cloud storage for use in Cloudera DataFlow, add the package directory (my-python-package in this example) directly inside the cloud storage directory that you are going to specify during deployment.

For example, if you specify s3a://bucket-name/custom-python as your cloud storage directory in the wizard, the following files should exist in cloud storage:

```
s3://my-bucket/custom-python/my-python-package/__init__.py
s3://my-bucket/custom-python/my-python-package/ProcessorA.py
```

```
s3://my-bucket/custom-python/my-python-package/ProcessorB.py
s3://my-bucket/custom-python/my-python-package/HelperModule.py
s3://my-bucket/custom-python/my-python-package/requirements.txt
```

Making the processor available for Cloudera DataFlow

In order to make a custom Python processor available to Cloudera DataFlow, upload it to cloud storage as described in Preparing cloud storage to deploy custom processors.

In the deployment wizard, specify this directory as Custom Python processor Storage Location.

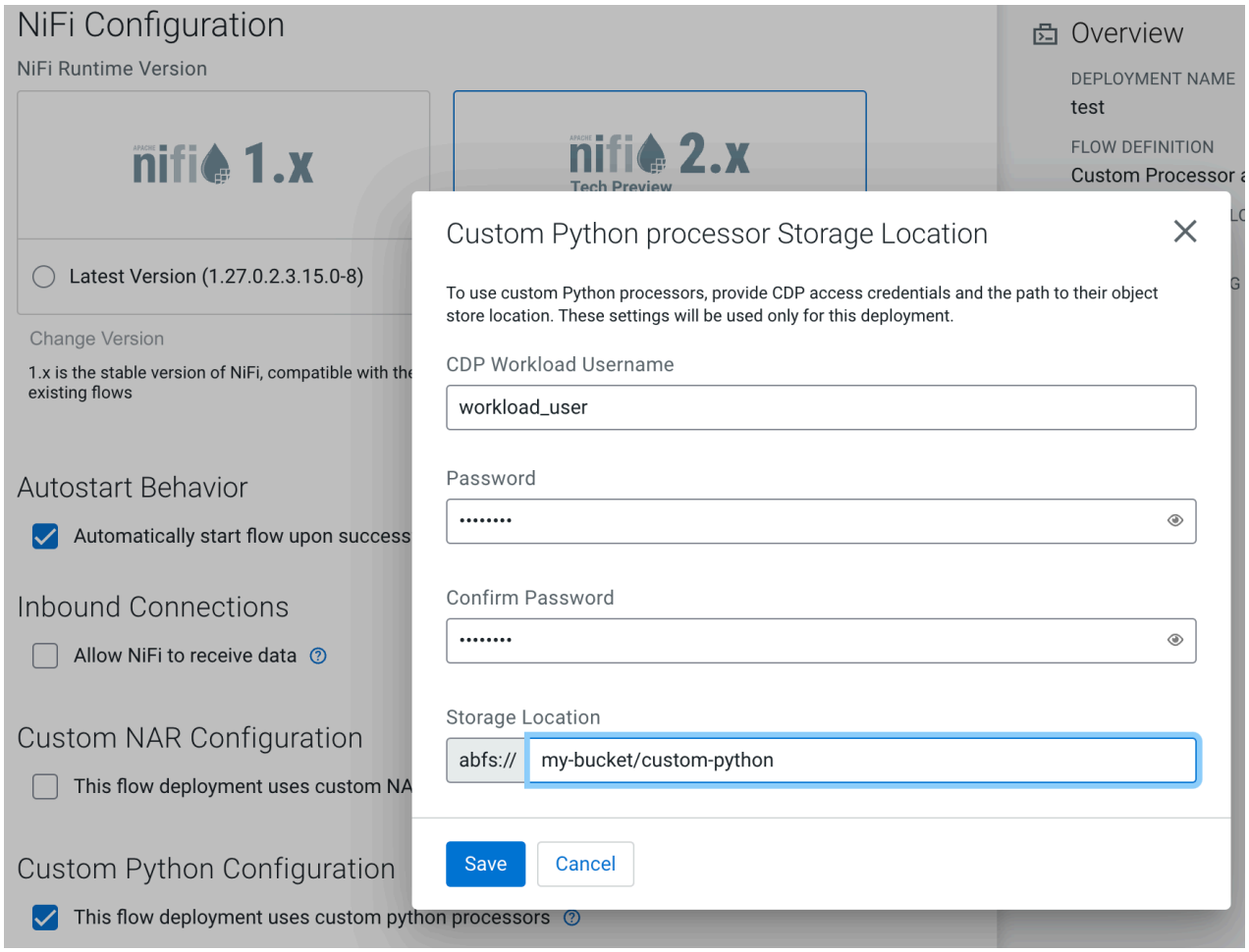

Related Tasks

Preparing cloud storage to deploy custom processors

Related Information

Deploying a flow definition using the wizard

NiFi Python Developer's Guide

Preparing cloud storage to deploy custom processors

To use a custom Apache NiFi processor or controller service in one of your Cloudera DataFlow flow deployments, add the NiFi Archive (NAR), Python file, or Python package containing the custom processor or controller service to a cloud storage location for later use during a flow deployment.

Procedure

1. Create your cloud storage location.
2. Upload your NAR file, Python file, or Python package to the cloud storage location.
3. Configure access to your cloud provider storage in one of two ways:
   - You have configured access to S3 buckets using ID Broker mapping.

     If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

     a. Access IDBroker mappings.

        1. To access IDBroker mappings in your environment, click Actions Manage Access .
        2. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.

     b. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

        Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.

     c. Click Save and Sync.
   - You have configured access to S3 buckets with a RAZ enabled environment.

     It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

     a. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
     b. From the Ranger UI, navigate to the S3 repository.
     c. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

        Tip:

        The Path field must begin with a forward slash (/).

     d. Add the machine user that you have created for your ingest workflow to the policy you just created.

     For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
4. Note the workload user name and password, and cloud storage location to use in the Deployment Wizard.

   Tip: If you want to provide a machine user as CDP Workload Username during flow deployment, make sure to note the full workload user name including the srv_ prefix.

What to do next

Once you have added the NAR files to a cloud storage location, you are ready to launch the Deployment Wizard and deploy a flow.

Cloudera DataFlow

Creating flow deployments

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Deploying a flow definition using the wizard.....4
Select the flow definition version you want to deploy from the catalog.....4
Launch the deployment wizard.....4
Name your flow deployment and assign it to a project.....5
Configure NiFi.....6
Provide parameter values.....7
Configure sizing and scaling.....7
Set Key performance indicators.....7
Verify your settings and initiate deployment.....8

Deploying a flow definition using the CLI.....8

Deploying a flow definition using the wizard

Deploy a flow definition to run Apache NiFi flows as flow deployments in Cloudera DataFlow. To do this, launch the Deployment wizard and specify your environment, parameters, sizing, and KPIs.

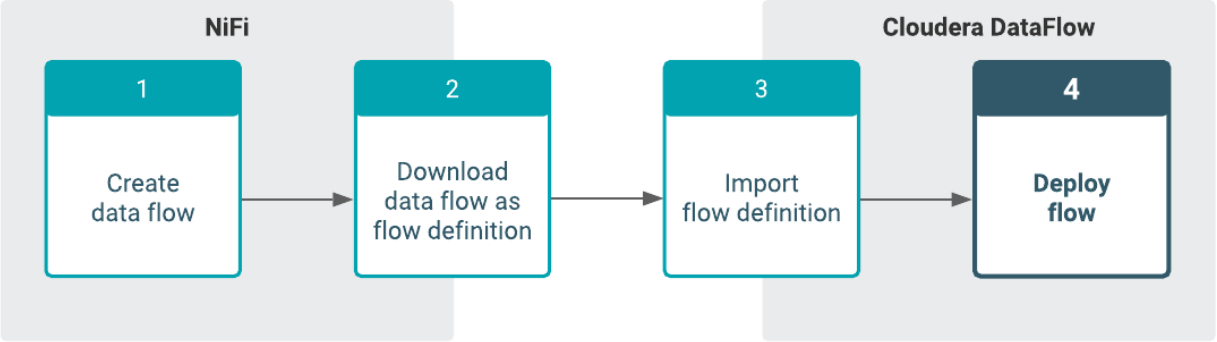

Before you begin

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog.
- The flow definition you want to deploy has been added to the Catalog by someone with DFCatalogAdmin role.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- You have been assigned DFProjectMember role for the Project where you want to deploy the flow definition.
- If you are deploying custom processors or controller services, you may need to meet additional prerequisites.

Select the flow definition version you want to deploy from the catalog

The Catalog is where you manage the flow definition lifecycle, from initial import, to versioning, to deploying a flow definition.

Procedure

1. In Cloudera DataFlow, select Catalog from the left navigation pane.

   Flow definitions available for you to deploy are displayed, one definition per row.
2. Select a row to display the flow definition details and available versions.

   The flow details pane opens on the right.

Launch the deployment wizard

After selecting a flow definition version from the catalog, you need to select an environment, provide a deployment name and assign it to a project using the deployment wizard.

Procedure

1. Click Deploy to launch the Deployment wizard.

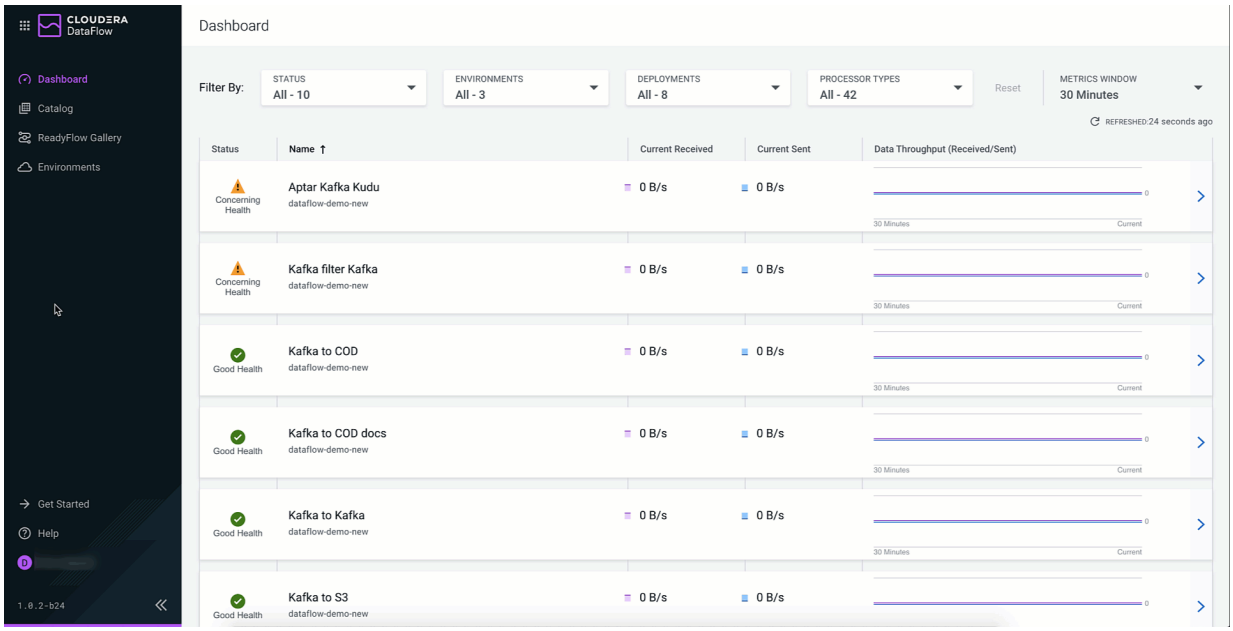

2. Select the environment where you want to deploy the flow.

   Note:

   Only environments which have been enabled for Cloudera DataFlow, are in a healthy state, and to which you have access show up in the dropdown. Once you have selected the target environment, you cannot change it later in the flow deployment process without starting over.

3. Click Deploy.

Name your flow deployment and assign it to a project

After selecting the flow version and an environment, the deployment wizard takes you to the Overview page. Here you need to provide a name for your flow deployment and assign it to a project. At this point you can also import a previously exported deployment configuration, auto-filling configuration values and thus speeding up deployment.

Procedure

1. Give your flow a unique Deployment Name.

   You can use this name to distinguish between different versions of a flow definition, flow definitions deployed to different environments, and similar.

   Note:

   Flow Deployment names need to be unique within an Environment. The Deployment wizard indicates whether a name is valid by displaying a green check below the Deployment Name text box.

2. Select a Target Project for your flow deployment from the list of Projects available to you.

   - If you do not want to assign the deployment to any of the available Projects, select Unassigned. Unassigned deployments are accessible to every user with DFFlowUser role in the environment.
   - This field is automatically populated if you import a configuration and the Project referenced there exists in your environment, and you have access to it.

3. If you have previously exported a deployment configuration that closely aligns with the one you are about to deploy, you can import it under Import Configuration to auto-fill as much of the wizard as possible.

   You can later manually modify auto-filled configuration values during deployment.
4. Click Next.

Configure NiFi

After selecting the target environment, project, and naming your flow, you need to set Apache NiFi version, possible inbound connections, and custom processors. Depending on the flow definition, you may also need to provide values for a number of configuration parameters. Finally, you need to set the capacity of the NiFi cluster servicing your deployment.

Procedure

1. Pick a NiFi Runtime Version for your flow deployment.

   Select if you want to use Apache NiFi 1.x or 2.x with your deployment.

   Important: NiFi 2.x is currently provided as a technical preview feature, do not use it for deployments in production environments.

   Cloudera recommends that you always use the latest available version within the 1.x and 2.x lines, if possible.
2. Specify whether you want the flow deployment to auto-start once deployed.
3. Specify whether you want to use Inbound Connections that allow your flow deployment receiving data from an external data source.

   If yes, specify the endpoint host name and listening port(s) where your flow deployment listens to incoming data.

   See *Creating an inbound connection endpoint* for complete information on endpoint configuration options.
4. Specify whether you want to use NiFi Archives (NARs) to deploy custom NiFi processors or controller services.

   If yes, specify the CDP Workload Username, password, and cloud storage location you used when preparing to deploy custom processors.

   Tip: If you want to provide a machine user as CDP Workload Username during flow deployment, make sure to note the full workload user name including the srv_ prefix.

   Make sure that you click the Apply button specific to Custom NAR Configuration before proceeding.
5. If you selected to run your flow with NiFi 2.x [Technical Preview], specify whether you want to use custom Python processors with your flow deployment.

   If yes, specify the CDP Workload Username, password, and cloud storage location where the processors are stored.

   Tip: Create a dedicated directory in your object store where you keep all your Python processors. Create one Python script per processor and store it in this directory.

   Tip: If you want to provide a machine user as CDP Workload Username during flow deployment, make sure to note the full workload user name including the srv_ prefix.

   Make sure that you click the Apply button specific to Custom Python Processors before proceeding.
6. Click Next.

Related Information

Inbound connections

Provide parameter values

Depending on the flow you deploy, you may need to specify parameter values like connection strings, usernames and similar, and upload files like truststores, JARs, and similar.

Procedure

1. Provide values to parameters required for your flow deployment.

   You have to provide values for all parameters. You can filter for the still empty fields by selecting the No value checkbox.

   Tip: If you are deploying a ReadyFlow, you can learn about required parameters and instructions on how to obtain parameter values by checking *Prerequisites* and *Required parameters* in the documentation of the respective ReadyFlow.
2. When you finished setting configuration parameters, click Next.

Configure sizing and scaling

Set the size and number of Apache NiFi nodes, auto-scaling, and the type of storage to be used.

Procedure

1. Specify NiFi node size.

   Select one of the following options:

   - Extra Small: 2 vCores per Node, 4 GB per Node
   - Small: 3 vCores per Node, 6 GB per Node
   - Medium: 6 vCores per Node, 12 GB per Node
   - Large: 12 vCores per Node, 24 GB per Node
2. Set the number of NiFi nodes and auto-scaling.

   - You can set whether you want to automatically scale your cluster according to flow deployment capacity requirements. When you enable auto-scaling, the minimum number of NiFi nodes are used for initial size and the workload scales up or down depending on resource demands.
   - You can set the number of nodes between 1 and 32.
   - You can set whether you want to enable Flow Metrics Scaling.
3. Select storage type.

   Select whether you want your deployment to use storage optimized for cost or for performance.

   - Standard: 512 GB Content Repo Size, 512 GB Provenance Repo Size, 256 GB Flow File Repo Size, 2300 IOPS, 150 MB/s Max Throughput
   - Performance: 1024 GB Content Repo Size, 1024 GB Provenance Repo Size, 256 GB Flow File Repo Size, 5000 IOPS, 200 MB/s Max Throughput
4. Click Next.

Related Information

Auto-scaling

Set Key performance indicators

Optionally add key performance indicators to help you track the performance of your flow deployment then review your settings and launch the deployment process.

Procedure

1. From KPIs, you may choose to identify key performance indicators (KPIs), the metrics to track those KPIs, and when and how to receive alerts about the KPI metrics tracking.

   Tip:

   You can reorder the KPIs by dragging them up or down the list. The order you configure here will be reflected in the detailed monitoring view of the resulting deployment.

   See *Working with KPIs* for complete information about the KPIs available to you and how to monitor them.
2. Click Next.

Related Information

Working with KPIs

Verify your settings and initiate deployment

Review deployment settings, make any necessary changes, and start deployment.

Procedure

1. Review a summary of the information provided and make any necessary edits by clicking Previous.
2. When you are finished, complete your flow deployment by clicking Deploy.

   Tip:

   Click View CLI Command to see the equivalent Cloudera CLI syntax in a help pane.

Results

After you click Deploy, you are redirected to the **Alerts** tab in the **Flow Details** where you can track how the deployment progresses.

Deploying a flow definition using the CLI

Deploy a flow definition to run NiFi flows as flow deployments in Cloudera DataFlow. To do this, launch the CDP CLI and specify your environment, parameters, sizing, and KPIs.

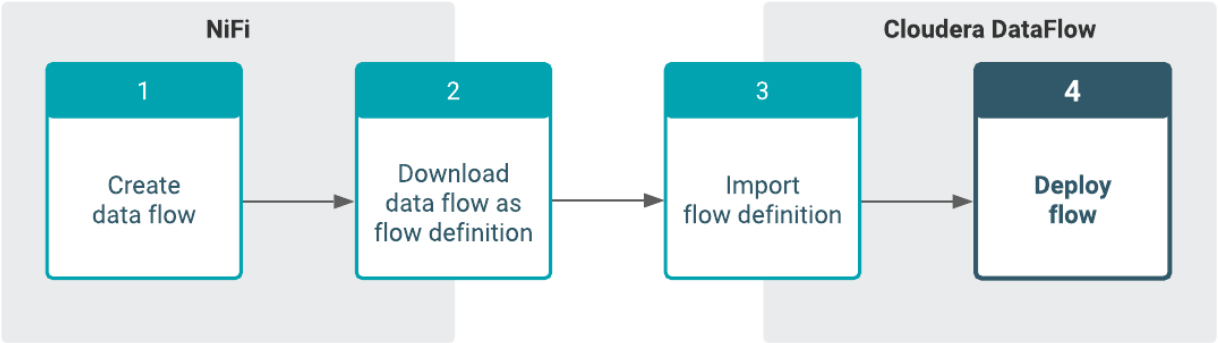

Before you begin

- You have installed CDP CLI.
- Run cdp df list-services to get the service-crn value.
- Run cdp df list-flows to get the flow-version-crn value.

Procedure

To deploy a flow, enter:

```
cdp df create-deployment
          --service-crn [***SERVICE-CRN-VALUE***]
          --flow-version-crn [***FLOW-VERSION-CRN-VALUE***]
          --deployment-name [***FLOW-DEPLOYMENT-NAME***]
          [--cluster-size-name [EXTRA_SMALL|SMALL|MEDIUM|LARGE]]
          [--static-node-count [***VALUE***]]
          [--auto-scaling-enabled | --no-auto-scaling-enabled]
          [--auto-scale-min-nodes [***NUMBER-MIN-NODES***]]
          [--auto-scale-max-nodes [***NUMBER-MAX-NODES***]]
          [--cfm-nifi-version [***NIFI-VERSION***]]
          [--auto-start-flow | --no-auto-start-flow]
          [--parameter-groups [***JSON-FILE-LOCATIONE***]]
          [--kpis [***JSON-FILE-LOCATIONE***]]
          [---node-storage-profile-name [STANDARD_AWS|STANDARD_AZURE|PERFO
RMANCE_AWS|PERFORMANCE_AZURE]]
```

Where:

--service-crn

Specifies the service-crn value you obtained when completing the prerequisites.

--flow-version-crn

Specifies the flow-version-crn value you obtained when completing the prerequisites.

--deployment-name

Specifies a unique name for your flow deployment.

[--cluster-size-name]

Specifies the cluster size. Valid values are:

- EXTRA_SMALL
- SMALL
- MEDIUM
- LARGE

The default value is EXTRA_SMALL.

[--static-node-count]

Specifies the number of NiFi nodes when autoscaling is not enabled. You can select between 1 and 32 nodes. The default value is 1.

[--auto-scaling-enabled | --no-auto-scaling-enabled]

Specifies whether you want to enable autoscaling. The default is to disable autoscaling.

[--auto-scale-min-nodes]

Specifies the minimum nodes when you have autoscaling enabled. If you have autoscaling enabled, this parameter is required.

[--auto-scale-max-nodes]

Specifies the maximum nodes when autoscaling is enabled. If you have autoscaling enabled, this parameter is required.

[--cfm-nifi-version]

Specifies the NiFi runtime version. The default is the latest version.

[--auto-start-flow | --no-auto-start-flow]

Specifies whether you want to automatically start your flow once it has been deployed. The default is to enable the automatic start.

[--parameter-groups]

Specifies the location of the parameter group JSON file, if you are using one for this flow deployment.

[--kpis]

Specifies the location of the KPIs JSON file, if you are providing KPIs for this flow.

--node-storage-profile-name

Specifies the storage profile. Valid values are:

- STANDARD_AWS
- STANDARD_AZURE
- PERFORMANCE_AWS
- PERFORMANCE_AZURE

The default values are STANDARD_AWS and STANDARD_AZURE, depending on the cloud provider.

Example parameter group file:

Note: The JSON file you develop for parameter group will be different depending on your flow objectives and requirements. This is an example of the parameter group file format.

```
[
  {
    "name": "kafka-filter-to-kafka",
    "parameters": [
      {
        "name": "CDP Workload User",
        "assetReferences": [],
        "value": "srv_nifi-machine-ingest"
      },
      {
        "name": "CDP Workload User Password",
        "assetReferences": [],
        "value": "<<CDP_MISSING_SENSITIVE_VALUE>>"
      },
      {
        "name": "CSV Delimiter",
        "assetReferences": [],
        "value": ","
      },
      {
        "name": "Data Input Format",
        "assetReferences": [],
        "value": "CSV"
      },
      {
        "name": "Data Output Format",
        "assetReferences": [],
        "value": "JSON"
      },
      {
        "name": "Filter Rule",
        "assetReferences": [],
        "value": "SELECT * FROM FLOWFILE"
      },
      {
        "name": "Kafka Broker Endpoint",
        "assetReferences": [],
        "value": "streams-messaging-broker0.pm-sandb.a465-9q4k.cloudera.sit
e:9093"
```

```
      },
      {
        "name": "Kafka Consumer Group ID",
        "assetReferences": [],
        "value": "cdf"
      },
      {
        "name": "Kafka Destination Topic",
        "assetReferences": [],
        "value": "MachineDataJSON"
      },
      {
        "name": "Kafka Producer ID",
        "assetReferences": [],
        "value": "cdf"
      },
      {
        "name": "Kafka Source Topic",
        "assetReferences": [],
        "value": "MachineDataCSV"
      },
      {
        "name": "Schema Name",
        "assetReferences": [],
        "value": "SensorReading"
      },
      {
        "name": "Schema Registry Hostname",
        "assetReferences": [],
        "value": "streams-messaging-master0.pm-sandb.a465-9q4k.cloudera.site
"
      }
    ]
  }
]
```

Example KPI file:

Note: The JSON file you develop for KPIs will be different depending on your flow objectives and requirements. This is an example of the KPI file format.

```
[
  {
    "metricId": "rateBytesReceived",
    "alert": {
      "thresholdLessThan": {
        "unitId": "kilobytesPerSecond",
        "value": 150
      },
      "frequencyTolerance": {
        "unit": {
          "id": "MINUTES"
        },
        "value": 1
      }
    }
  },
  {
    "metricId": "processorAmountBytesSent",
    "alert": {},
    "componentId": "a7f7df1c-a32d-3c25-9b09-e1d1036dcc04;a33a1b48-005b-32dd-
bb88-b63230bb8525"
  }
```

```
]
```

Results

Successfully deploying a flow results in output similar to:

```
{
  "crn": "deployment-crn"
}
```

Cloudera DataFlow

Getting Ready to Run Functions

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Developing a NiFi flow for Cloudera DataFlow Functions..... 4
Flow design principles.....4
Flow testing principles..... 7

Downloading a flow from Apache NiFi.....7

Uploading an Apache NiFi flow to the Catalog.....8

Downloading Lambda Cloudera DataFlow Function binaries and uploading to S3..... 8

Retrieving data flow CRN..... 9

Creating Cloudera service account.....10

Generating Access Key ID and Private Key.....10

Developing a NiFi flow for Cloudera DataFlow Functions

A flow definition represents the data flow logic that you can download from NiFi, import to the Cloudera DataFlow Catalog and run in serverless mode. You can develop this data flow for your function in any development environment using Apache NiFi and then deploy the function on a Function as a Service (FAAS) solution on AWS, Azure, or the Google Cloud Platform (GCP).

You have three main options for developing your flow:

- Use Cloudera DataFlow Flow Designer and publish your draft as a flow definition in the Catalog. For more information on using the Flow Designer, see *Creating a new draft*. In this case you can skip the downloading flow and importing flow to Catalog steps as you have already added your flow to the Cloudera DataFlow Catalog.
- Use Cloudera Public Cloud Data Hub with the Flow Management template, if you are a Cloudera customer who has a Cloudera Public Cloud Data Lake.

  For more information on how to set up a managed and secured Flow Management cluster in Cloudera Public Cloud, see *Setting up your Flow Management cluster*.
- Develop the data flow in your local development environment using open source Apache NiFi.

Once you have developed and tested your NiFi flow, you can deploy it as a Cloudera DataFlow function in serverless mode using one of the three cloud providers function services: AWS Lambda, Google Cloud Functions, and Azure Functions.

To make sure that your NiFi data flow can be deployed as a function in Cloudera DataFlow, review your traditional NiFi flow development process and follow the best practices outlined in the next sections.

Related Information

Creating a new draft

Setting up your Flow Management cluster

Flow design principles

To make sure that your Apache NiFi data flow can be deployed as a function in Cloudera DataFlow, follow the best practices outlined in this section.

Procedure

1. Create a Process Group and associate a Parameter Context to it.

   The Parameter Context can itself inherit or not, from other Parameter Contexts. Make sure that the name of your Parameter Contexts starts with a letter and it only consists of letters, numbers, and the underscore (_) character.
2. Go into the Process Group and add an Input Port.

   This is the start of your flow where the Cloudera DataFlow Functions code will “generate” a FlowFile containing the payload generated by the function’s trigger. See the corresponding sections of each cloud provider in the Cloudera DataFlow Functions documentation to see what the FlowFile’s content would be.
3. Add your processors to extract the required data from the FlowFile and perform the required actions.
4. Once the flow is designed, parameterize all processor and controller service properties that should be externalized for configuration during function creation.

   Make sure that all parameters have names that start with a letter and only consist of letters, numbers, and the underscore (_) character.

S3 trigger

In this example, use the EvaluateJsonPath processor to extract the bucket name and the object key, and continue the flow as you wish, for example fetch the data, compress it and send the result to a different location.

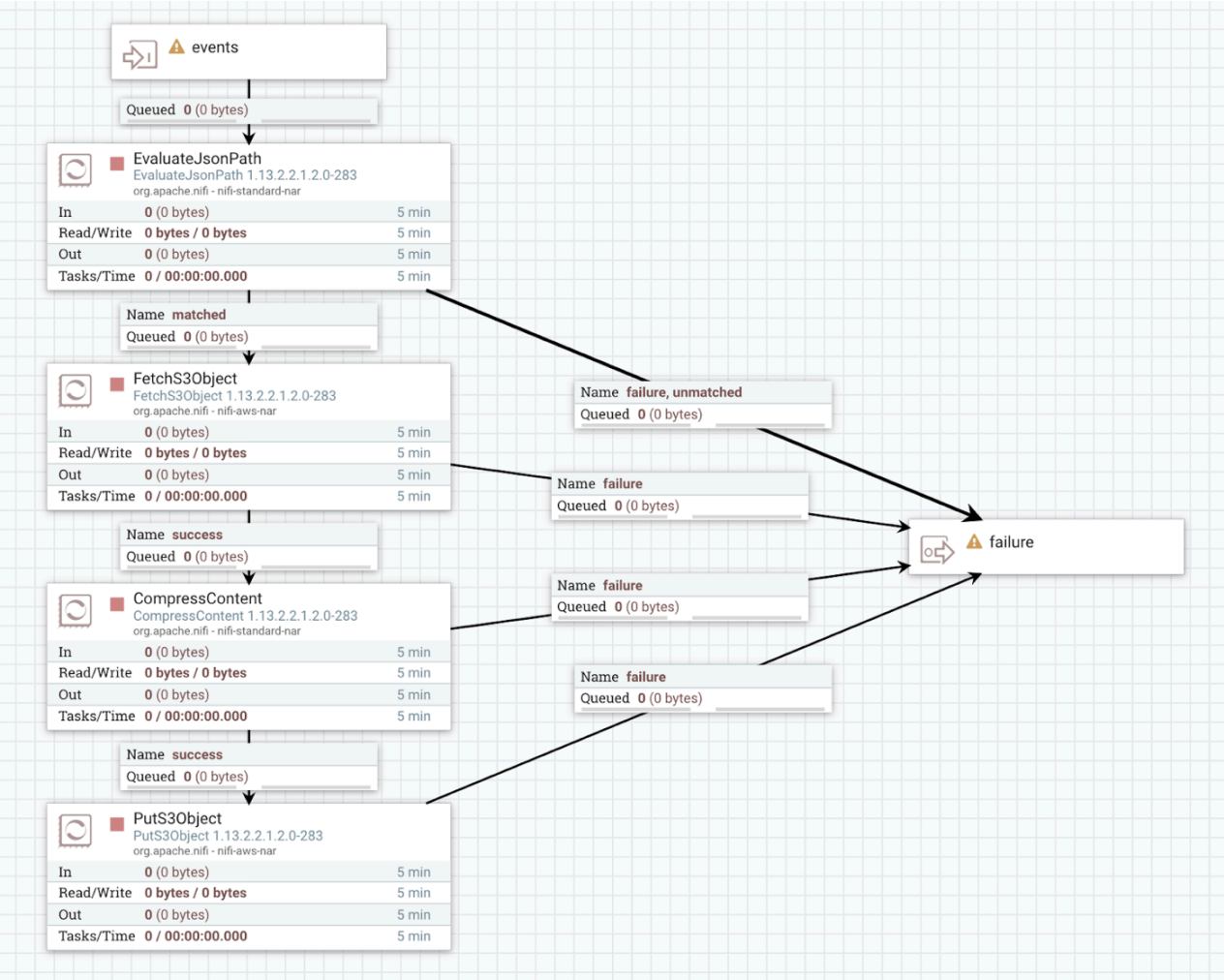

When using an AWS Lambda with an S3 trigger, the generated FlowFile would have a payload similar to this:

```
{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-2",
      "eventTime": "2021-08-06T09:53:57.377Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "AWS:AIDAVRSV7HNNHFQQOE2F4"
      },
      "requestParameters": {
        "sourceIPAddress": "18.222.152.52"
      },
      "responseElements": {
        "x-amz-request-id": "WBPM5W0026J3NWEH",
        "x-amz-id-2": "Cco/DVtbHFYtb+rXQrEMUFJnZSxtebZL61KG79emsACUDqUIyrAziY3frha9AzX+RSLx9syJMl4OMCn06poZv1dJQJg5TgRx"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "ad0ec124-9c9f-476e-9dd1-9e513d9af822",
        "bucket": {
          "name": "pvillard-naaf-input",
          "ownerIdentity": {
            "principalId": "A339RKRQ79D9V"
          },
          "arn": "arn:aws:s3:::pvillard-naaf-input"
        },
        "object": {
          "key": "04f56bcf-cb15-4244-9379-2abc7ac0cd8d",
          "size": 1024,
          "eTag": "265bc4c12162ffca179a58975728a033",
          "sequencer": "00610D06B8BE365D41"
        }
      }
    }
  ]
}
```


Getting the payload associated with a trigger

You can design a flow that would just push as-is the event generated by the function's trigger into S3. This can be useful when you do not know beforehand what would be generated by a given trigger.

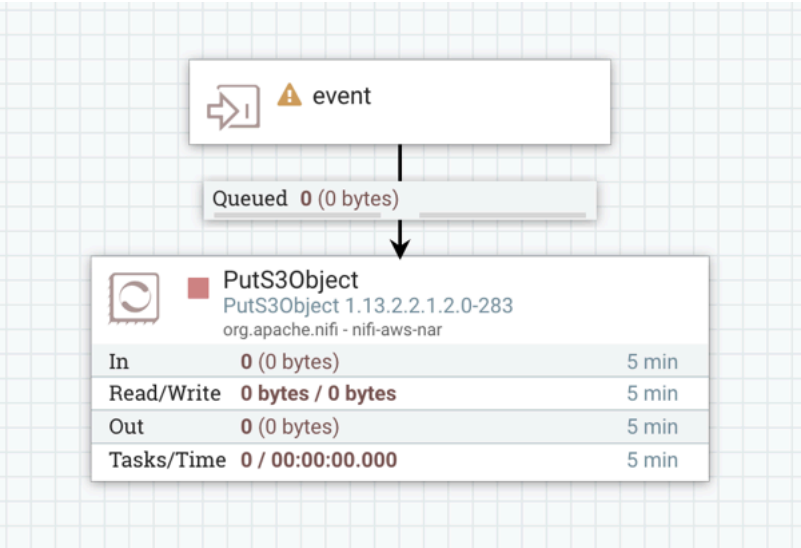

Flow testing principles

As with any data flow that you build, it is important to test it before deployment. Data flows to be used by AWS Lambda require an Input Port and most often will also contain at least one Output Port. This makes it quite easy to test these data flows in NiFi.

Procedure

1. Go to the Parent Process Group within Apache NiFi and add a GenerateFlowFile processor.
2. Set the Custom Text property to any value that you want to feed into your data flow.

   For example, to simulate an event indicating that data was added to an S3 bucket, you would set the Custom Text property to the following:

```
{
  "Records": [
    {
      "awsRegion": "us-east-1",
      "s3": {
        "bucket": {
          "name": "my-nifi-logs",
          "arn": "arn:aws:s3:::example-bucket"
        },
        "object": {
          "key": "nifi-app_2021-10-12_18.36.log.gz",
          "size": 1024,
        }
      }
    }
  ]
}
```

3. Connect the GenerateFlowFile processor to the Input Port of the Process Group that you will run in Lambda.

   The Run Once feature of NiFi makes it easy to create this FlowFile and send it to the Process Group.
4. Start the Process Group and ensure that the data processes as expected.

   If not, you can fix the data flow and trigger the GenerateFlowFile processor again, until you have properly handled the data.

What to do next

When ready, you can download the Flow Definition from NiFi and upload it to the Catalog in Cloudera DataFlow.

Downloading a flow from Apache NiFi

When ready, you can download the Flow Definition from Apache NiFi, import it to the Cloudera DataFlow Catalog, and deploy the function on AWS, Azure, or the Google Cloud Platform (GCP) to run it in serverless mode. In Cloudera DataFlow Functions, only flows versioned in the Cloudera DataFlow Catalog can be used as functions.

The Download flow definition option in NiFi allows you to download the flow definition of a Process Group as a JSON file. After you have downloaded your flow from NiFi, you can import it into Cloudera DataFlow. For detailed instructions, see *Downloading a flow definition from NiFi* .

Related Information

Download a flow definition from NiFi

Uploading an Apache NiFi flow to the Catalog

Before an Apache NiFi flow can be run in serverless mode on a cloud provider function service, you must register it as a Cloudera DataFlow function by importing it into the Cloudera DataFlow Catalog. With Cloudera DataFlow Functions, only flows versioned in the Cloudera DataFlow Catalog can be used as functions.

If you want to use a NiFi flow in Cloudera DataFlow, you must import it to Cloudera DataFlow as a flow definition. When imported, the flow definition is added to the Catalog and you can start using it. For instructions, see *Importing a flow definition*.

Related Information

Importing a flow definition

Downloading Lambda Cloudera DataFlow Function binaries and uploading to S3

To be able to run the NiFi flow in AWS Lambda, you need to add the Cloudera DataFlow function handler libraries to S3.

Procedure

1. Click Functions in the left navigation pane and download the Cloudera DataFlow Function binaries for AWS Lambda.

   AWS Lambda will use these binaries to run the NiFi flow.

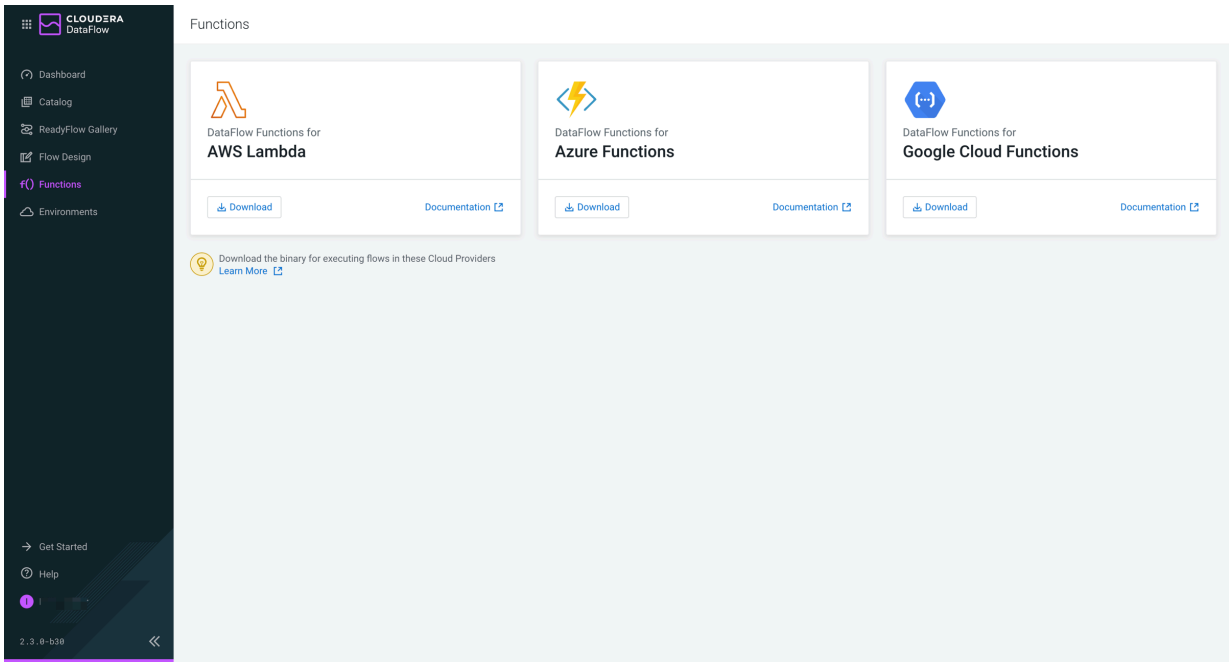

2. Upload this binary to an S3 bucket that you will later reference when creating the function in AWS Lambda.

   The S3 bucket needs to be in the same region as where the Lambda is being created/deployed.

3. Copy the S3 URI for later use.

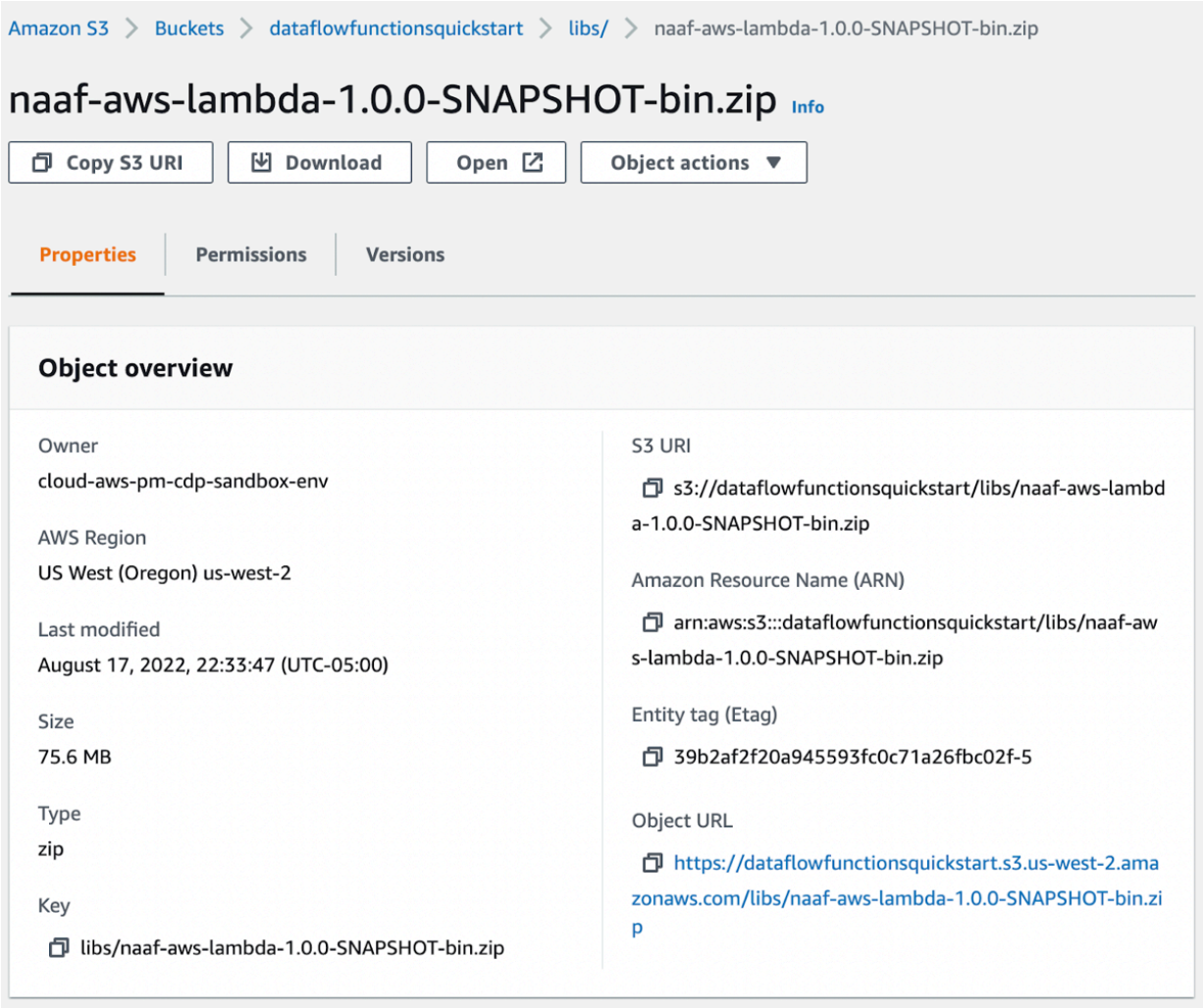

Retrieving data flow CRN

When configuring your function on the cloud provider service page, you need to provide the Customer Resource Number (CRN) of the flow to be executed.

You can retrieve the CRN by checking the flow in the Cloudera DataFlow Catalog.

Note: The CRN must include the specific version of the flow that should be executed, so it should end with some version suffix such as /v.1.

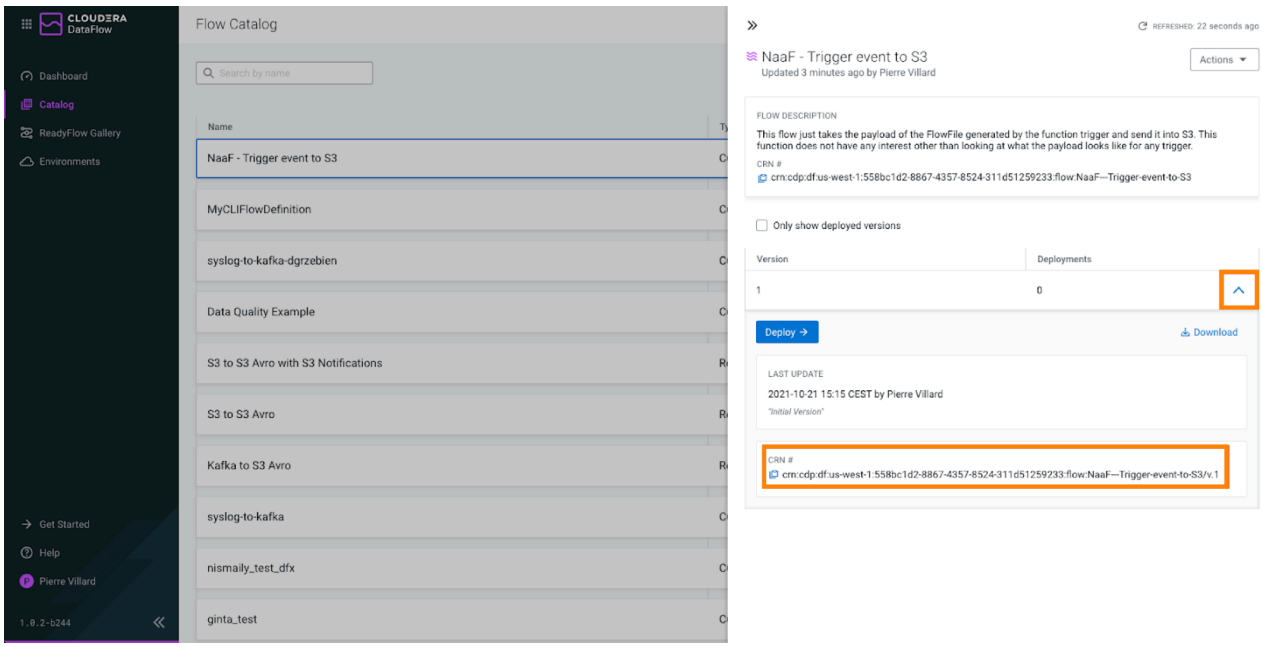

Creating Cloudera service account

The first step of executing Cloudera DataFlow Functions code in your cloud environment is fetching the flow definition to be executed from the Cloudera DataFlow Catalog. For this, you need to create a service account and provision an access key.

Procedure

1. Navigate to the Management Console User Management Users .
2. From the Actions menu, select Create Machine User.
3. Provide a name and click Create.

Note: Machine user names cannot start with a double underscore ("__").

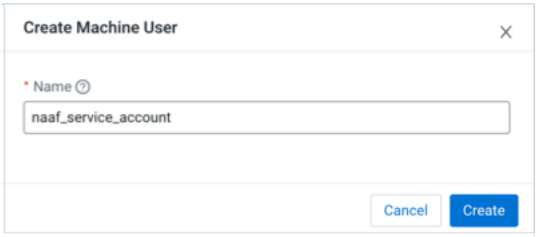

The user details page is displayed showing information about the user.

Generating Access Key ID and Private Key

A Cloudera machine user must have API access credentials to access Cloudera services through the Cloudera CLI or API.

Procedure

1. Click the Roles tab on the user account details page.
2. Click Update Roles.

   The Update Roles pane is displayed.
3. Select the DFFunctionMachineUser role to assign it to your user.
4. Click Update.

   When the role is added, you should see:

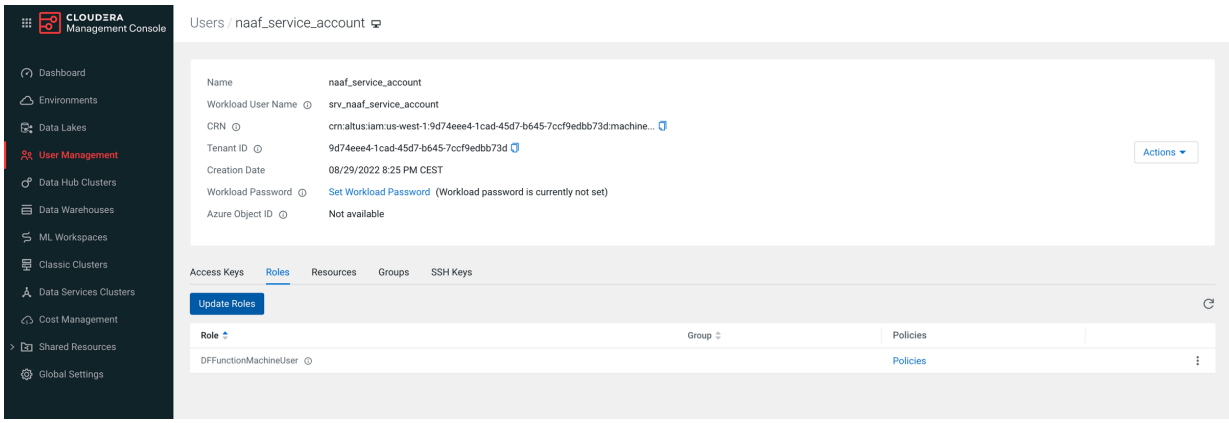

5. Select the Access Keys tab on the user account details page.
6. Click Generate Access Key.

   The Generate Access Key modal window is displayed. it gives you an Access Key ID and a Private Key, which will be required when configuring your function.
7. Click Generate Access Key.

   A message is displayed that your access key has been successfully created.

   You can copy your Access Key ID and your Private Key. You will need these when configuring your function.
8. You can also download the credentials file into the .cdp directory in your user home directory. Or run the command cdp configure and enter the access key ID and private key to create a Cloudera credentials file in the same directory.

Results

When ready, you should see something like this:

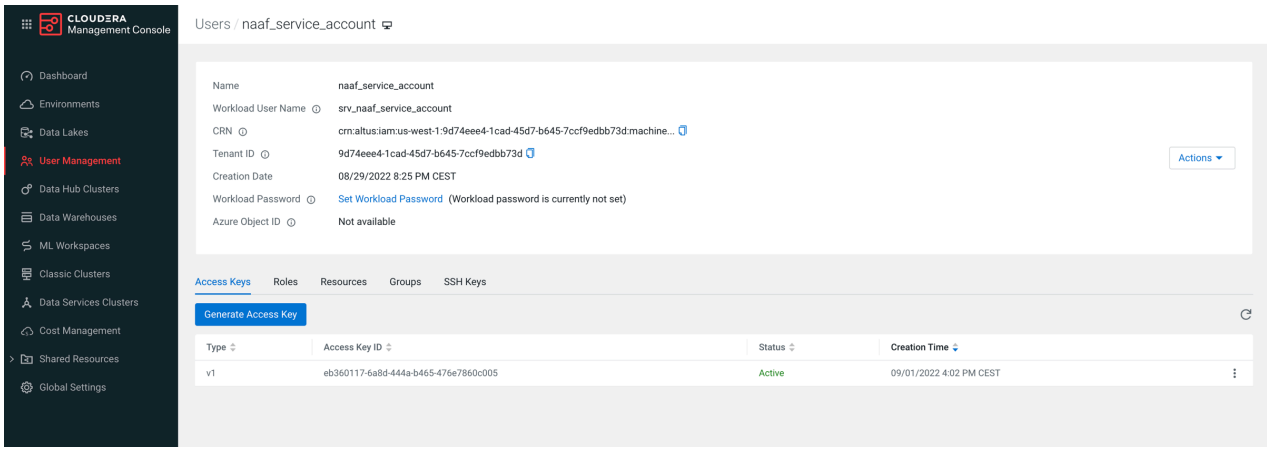

Cloudera DataFlow

Deploying a ReadyFlow

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Add ReadyFlow to the Catalog.....4

Deploy a ReadyFlow.....4

Add ReadyFlow to the Catalog

ReadyFlows are out of box flow definitions, designed to make getting started with Cloudera DataFlow quick and easy. To use them, just add the ReadyFlow to the Catalog.

For UI

Steps

1. From the ReadyFlow Gallery page, identify the ReadyFlow you want to use.
2. Review the ReadyFlow details by clicking View Details.
3. Click Add to Catalog to add the ReadyFlow into the Cloudera DataFlow Catalog and make it ready for deployment.
4. Click Add to confirm that you want to add the ReadyFlow to the Catalog.

For CLI

Before you begin

- You have installed Cloudera CLI.
- Run cdp df list-ready-flows to get the crn.

Steps

1. To add a ReadyFlow to the Catalog, enter:

```
cdp df add-readyflow
--crn [***CRN***]
```

   Where:

   - --crn specifies the ReadyFlow CRN value you retrieved when running cdp df list-ready-flows.

Result

The ReadyFlow is added to the Catalog and is ready for deployment.

Related Information

Available ReadyFlows

Deploy a ReadyFlow

Learn how to deploy a ReadyFlow using the information you collected using the prerequisites check list.

Before you begin

- You have added the ReadyFlow you want to deploy to the Cloudera DataFlow Catalog.
- You have met *Prerequisites* and collected values for *Required parameters* of the selected ReadyFlow.

For information on deploying a flow using either the deployment wizard or the CLI, see Creating flow deployments.

Cloudera DataFlow

Developing Flow Definitions Using NiFi

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Best Practices for developing flow definitions.....4

Considerations when developing flow definitions..... 4

Creating parameter context for flow definitions.....9

Creating controller service for flow definitions.....9

Creating a flow definition in NiFi.....10

Adding Inbound Connection support to a NiFi flow..... 13

Download a flow definition from NiFi..... 14

Best Practices for developing flow definitions

Before you can deploy a flow definition in Cloudera DataFlow, you need to develop your data flow logic in a development environment using Apache NiFi. To make sure that your Apache NiFi data flow can be deployed in Cloudera DataFlow, follow the best practices outlined in this section.

What is a flow definition?

A flow definition represents data flow logic which was developed in Apache NiFi and exported by using the Download Flow Definition action on a NiFi process group or the root canvas. Flow definitions typically leverage parameterization to make them portable between different environments such as development or production NiFi environments.

To run one of your existing NiFi data flows in Cloudera DataFlow you have to export it as a flow definition and upload it to the Catalog.

What is flow isolation?

Cloudera DataFlow is the first cloud service allowing NiFi users to easily isolate data flows from each other and guarantee a set of resources to each one without requiring administrators to create additional NiFi clusters.

Flow isolation describes the ability to treat NiFi process groups which typically run on a shared cluster on shared resources as independent, deployable artifacts which can be exported as flow definitions from NiFi.

Flow isolation is useful when

- You want to guarantee a set of resources for a specific data flow.
- You want to isolate failure domains.

Considerations when developing flow definitions

As you are building your flow definition in your NiFi development environment, you should build the flow definition with ease of Cloudera DataFlow export and isolation in mind.

Controller Services

In traditional NiFi data flows, Controller Services are considered shared services and often used by multiple data flows, or by multiple Processors in different Process Groups to avoid redundancies and promote reusability. In Cloudera DataFlow, you should plan for the possibility that you will run your process group in isolation and consider how you want to handle shared Controller Services.

When you download a flow definition from NiFi, you can choose whether you want to include external controller services (Requires NiFi 1.16 / 1.17.0 or newer). A controller service is considered external when it is referenced by components in the selected process group but is defined outside the process group scope (e.g. controller services in a parent group). Select “With external services” if you want to include referenced controller services in the resulting flow definition and select “without external services” if you only want to include controller services which are defined in the selected process group.

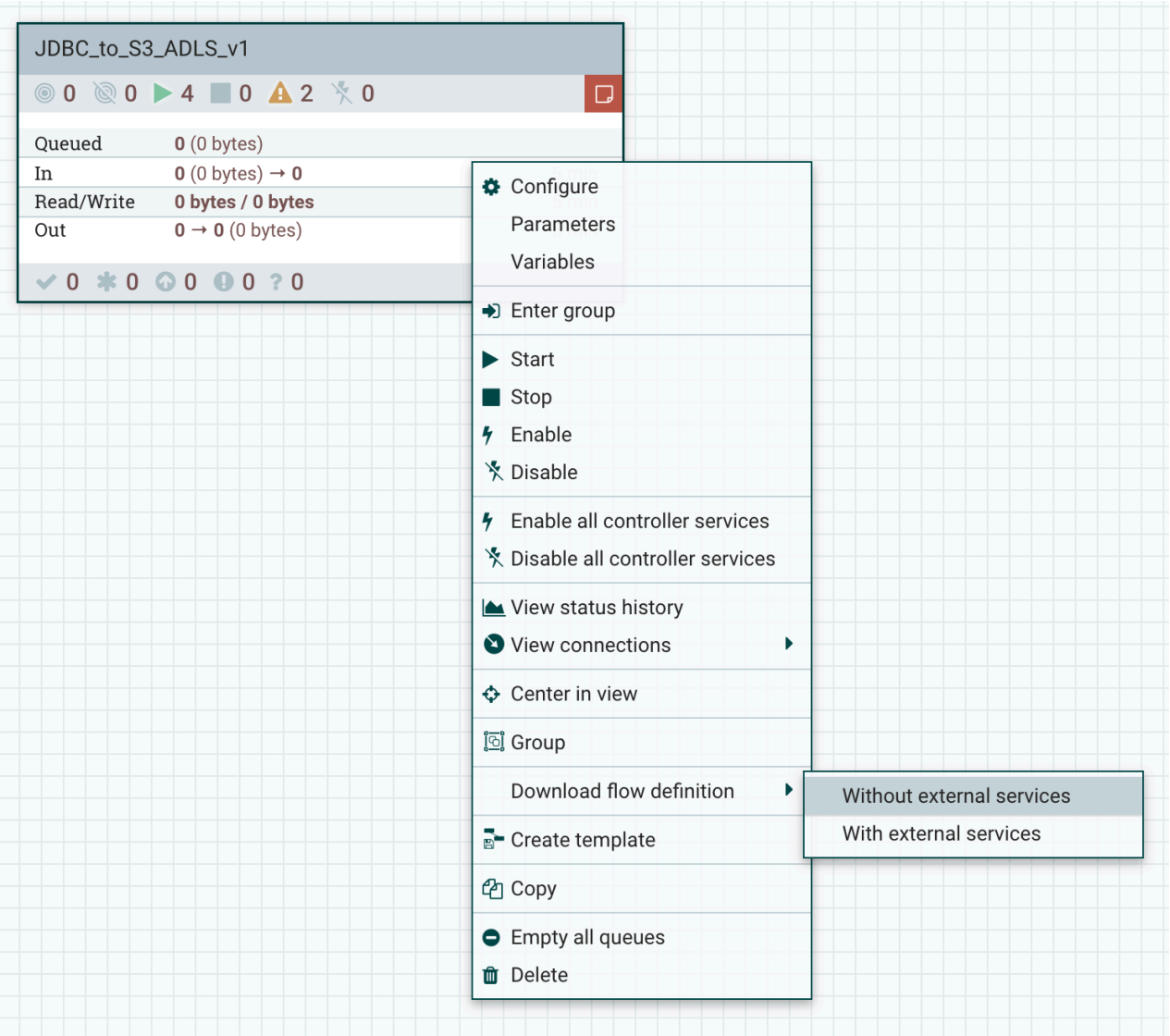

Note:

If use an older version of NiFi which does not support including external services in flow definitions, review the Controller Services defined outside your Process Group and if needed recreate them in the Process Group you are planning to download.

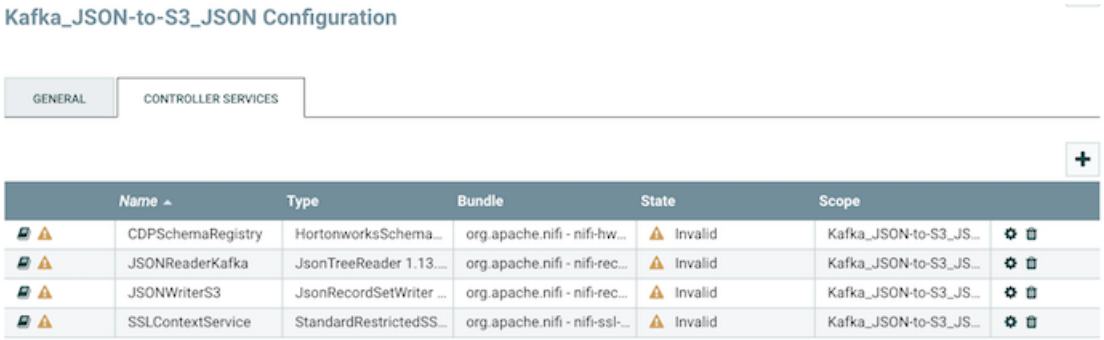

Tip:

An easy way to check whether a controller service is considered external is to compare its scope with the process group name you are planning to download. If they do not match, it is considered an external Controller Service.

Default NiFi SSL Context Service

For data flows that interact with other Cloudera Public Cloud experiences like Streams Messaging or Operational Database Data Hub clusters, you can reference an external NiFi controller service called Default NiFi SSL Context Service in your NiFi flow to automatically obtain the correct truststore configuration for your target Cloudera Public Cloud environment.

Figure 1: Using the Default NiFi SSL Context Service in a processor configuration

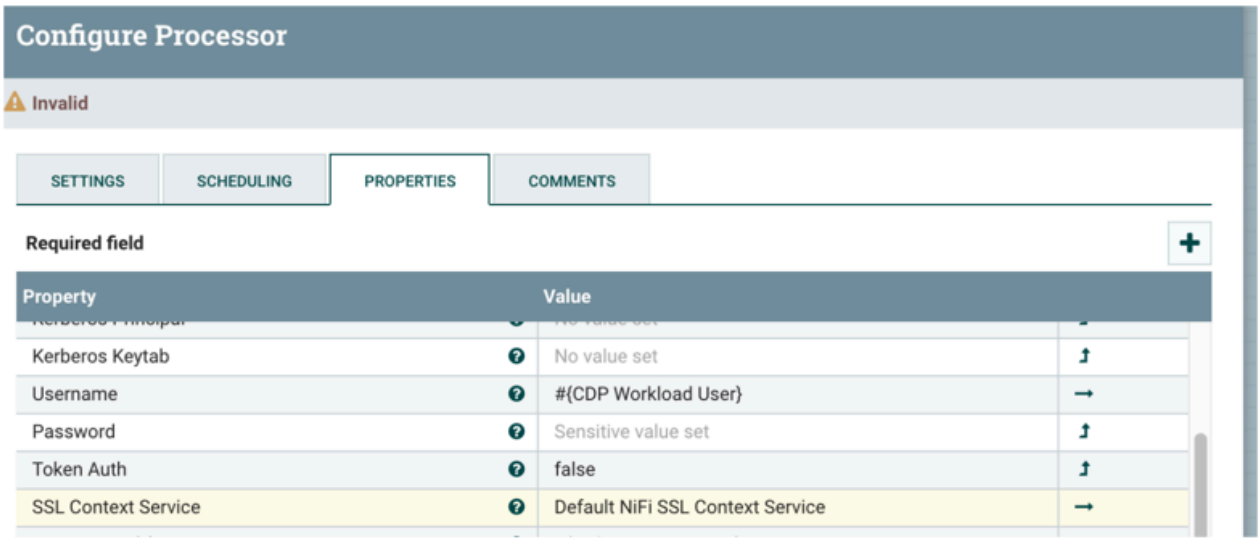

A Default NiFi SSL Context Service is already set up for you and can be used when you are developing NiFi flows on Flow Management clusters in Data Hub. Any flow that uses the Default NiFi SSL Context Service in Data Hub can be exported and deployed through Cloudera DataFlow.

If you are developing your NiFi flows outside of Data Hub, you can create a controller service called Default NiFi SSL Context Service yourself and reference it in any processor that requires an SSL Context Service configuration. You have to define the Default NiFi SSL Context Service as an external Controller Service outside of the process group that you plan to export and run in Cloudera DataFlow.

Figure 2: If you are creating the Default NiFi SSL Context Service yourself, make sure that the Scope it is created in is a parent to the process group that you plan to export

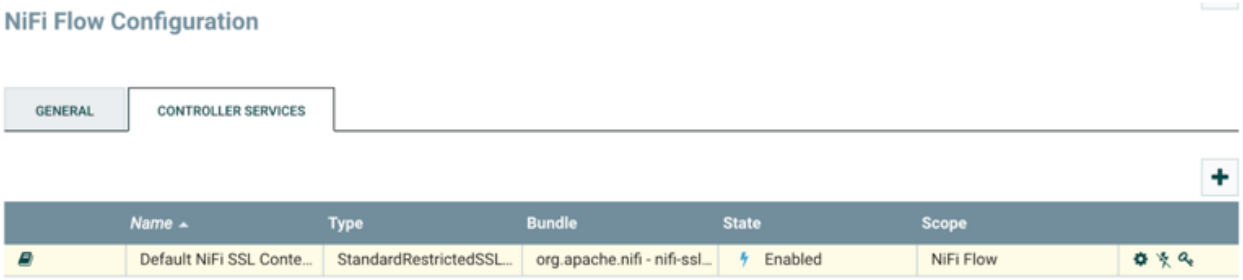

Review the following information if you developed a flow in a Cloudera Flow Management Data Hub cluster and the flow uses the SSLContextService that was defined outside the Process Group:

- If the name of the SSLContextService in the flow definition is Default NiFi SSL Context Service, then at the time of deployment, Cloudera DataFlow automatically creates a new SSLContextService in the Root Process Group with the required information of the target environment.
- Specifically, Cloudera DataFlow imports the Truststore certificates, creates the Truststore, puts the information on a shared space that all NiFi nodes can access, and updates the Truststore filename property with the correct file path.
- When the flow definition is deployed, the flow references the Default NiFi SSL Context Service in the Root Process Group and the deployment is successful.

If your flow interacts with non-Cloudera Public Cloud services that require a TLS/SSL connection, then you must do the following:

- Define a new SSLContext Service in the Process Group you are planning to export.
- Parameterize the Truststore Filename property so that Cloudera DataFlow allows you to upload a custom Truststore when you deploy the flow definition using the Flow Deployment Wizard.

Parameterize Processor and Controller Service configurations

Ensure that your Process Group is portable by parameterizing your processor and controller services configurations. This allows you to deploy a flow in different environments without having to update Processor configuration details.

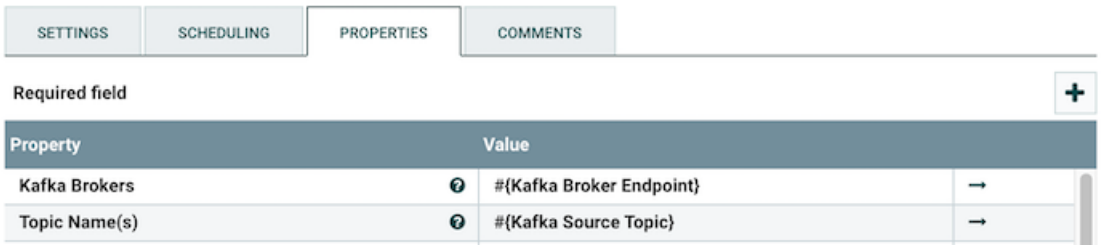

Customize your Processor and Connection names

To ensure that you are able to distinguish Processors and Connections when defining KPIs in your Cloudera DataFlow flow deployment ensure that you specify a custom name for them when developing your data flow.

For example:

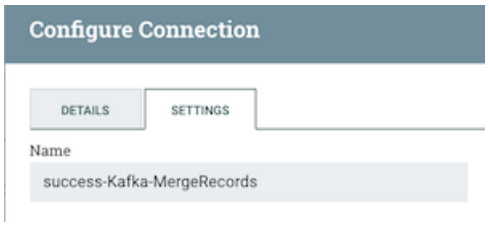

Review Reporting Tasks

Reporting tasks are not exported to Cloudera DataFlow when you are exporting your flow definition. When you are designing your flow definition, you should be aware of your monitoring and report needs and plan to address these needs with KPIs and Alerts specified as part of your flow deployment.

Using the CDPEnvironment parameter to get Hadoop configuration resources

Cloudera DataFlow makes it easy to use HDFS processors to read/write data from/to S3. Using HDFS processors to access data in S3 allows you to leverage Cloudera Public Cloud's IDBroker authentication method and doesn't require you to specify any S3 secrets in the processor configuration. HDFS processors however require Hadoop configuration files - specifically the core-site.xml file. Cloudera DataFlow offers a special parameter called CDPEnvironment for you to use whenever you are working with processors that require Hadoop configuration files.

Simply use the parameter #{CDPEnvironment} for the Hadoop configuration file property and Cloudera DataFlow will automatically obtain the required files during the flow deployment process.

Note:

The value of the #{CDPEnvironment} parameter is not automatically provided when using it in your NiFi development environment. While developing your flow, make sure to add the CDPEnvironment parameter to your parameter context with the correct value for your environment.

For example, in DataHub set the value for this parameter to /etc/hadoop/conf/core-site.xml.

Providing the correct value for your environment allows you to successfully test the flow.

Required field

Property	Value
Hadoop Configuration Resources	#{CDPEnvironment}

Using CDPEnvironment for the Hadoop Configuration Resources property.

CDPEnvironment

core-site.xml

ssl-client.xml

hive-site.xml

DataFlow automatically adds all required configuration files to interact with Data Lake services. Unnecessary files that are added won't impact the deployment process.

The Cloudera DataFlow Deployment Wizard detects usage of the CDPEnvironment parameter and automatically obtains the required Hadoop configuration files.

Resource consumption

When deciding how to export your process groups for deployment on DFX, review the data flow resource consumption to help you decide how to isolate your data flows within Cloudera DataFlow.

Once you have developed your data flow in a NiFi development environment, you start to export your flow definition

- Root canvas
- Whole process group level
- Part of the process group

Inter process group communication

If you have process groups that exchange data between each other, you should treat them as one flow definition and therefore download their parent process group as a flow definition.

Configuring object store access

When you are developing flows for cloud provider object stores, `CDPObjectStore` are the preferred processors. Available processors are:

- `ListCDPObjectStore`
- `FetchCDPObjectStore`
- `PutCDPObjectStore`

- `DeleteCDPObjectStore`

`CDPObjectStore` processors are equipped with the latest best practices for accessing cloud provider object stores.

Configuring inbound connection support

For more information, see *Adding Inbound Connection support to a NiFi flow*.

Related Information

Adding Inbound Connection support to a NiFi flow

Creating parameter context for flow definitions

About this task

As you are building your data flow in your NiFi development environment, follow these steps to create a parameter context, add parameters to it, and assign the new context to your process group:

Procedure

1. Select Parameter Contexts from the Global menu of the NiFi UI.
2. In the NiFi Parameter Contexts dialog, click the (+) button to create a new parameter context.
3. Add a name for your parameter context.
4. Select the Parameters tab to add the parameters you need for configuring your data flow components.
5. Click Apply to save the parameter context.
6. To assign the parameter context to your process group, click Configure from the process group context menu and select the parameter context in the Process Group Parameter Context drop-down menu.

Note: To add parameters to an existing parameter context, open the Parameter Contexts dialog from the Global menu and click the (Pencil) button in the row of the chosen parameter context. You can also add new parameters to the parameter context when you are configuring the components in your data flow. For more information about parameters and parameter contexts, see the Apache NiFi User Guide.

Creating controller service for flow definitions

About this task

As you are building your data flow in your NiFi development environment, follow these steps to create a controller service:

Procedure

1. Select your process group and click Configure from either the Operate Palette or the process group context menu of the NiFi UI.

   The Process Group Configuration dialog is displayed.
2. Select the Controller Services tab.
3. Click the (+) button to add a new controller service.

   The Add Controller Service dialog is displayed.
4. Select the required controller service and click Add.
5. Click the (Configure) icon in the right column and set the necessary options.
6. Click Apply to save the changes.

7. Click the (Flash) button to enable the controller service.

Important: You must define the controller services for your data flow in the configuration of the process group where they will be used. This ensures that the controller services are included in the process group flow definition.

Creating a flow definition in NiFi

Before you can run a data flow in Cloudera DataFlow, you need to (1) create the flow in Apache NiFi, (2) download the NiFi flow as a flow definition, (3) import it to Cloudera DataFlow and finally, (4) deploy the flow. The flow definition acts as a configuration logic for your flow deployments. It enables you to deploy your data flow without the need to maintain cluster infrastructure. Also, you can deploy the same flow to multiple environments in Cloudera DataFlow.

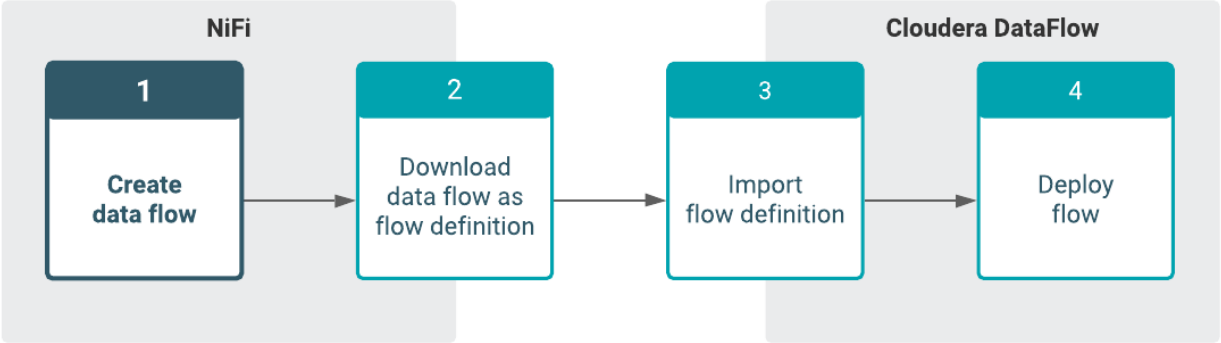

Before you begin

When you want to develop an Apache NiFi flow that you intend to use in Cloudera DataFlow, review and adjust your traditional NiFi flow development process to make sure that you can create portable data flows that will work in the Cloudera DataFlow environment. Before you get started with flow development, it is useful to understand where you need special attention and what adjustments you have to make in your development workflow.

You can create and download flow definitions starting with version 1.11 of Apache NiFi. Cloudera provides the following Apache NiFi based products:

- Cloudera DataFlow for Cloudera Data Hub
- Cloudera Flow Management 1.1.0 and higher
- HDF 3.5

Cloudera recommends that you develop your flow definitions using Cloudera Data Hub Flow Management clusters. For more information on how to set up a managed and secured Flow Management cluster in Cloudera Public Cloud, see *Setting up your Flow Management cluster*.

For more information on planning and preparing your NiFi flows for Cloudera DataFlow, see *Best Practices for Developing Flow Definitions*.

Procedure

1. Create a process group that will contain your NiFi flow.
   a) Drag and drop the process group icon onto the canvas.
   b) Add a name for the process group.

   Once you have the new process group available on the canvas, you can interact with it by right-clicking it and selecting an option from the context menu. The available options vary depending on the privileges assigned to you.

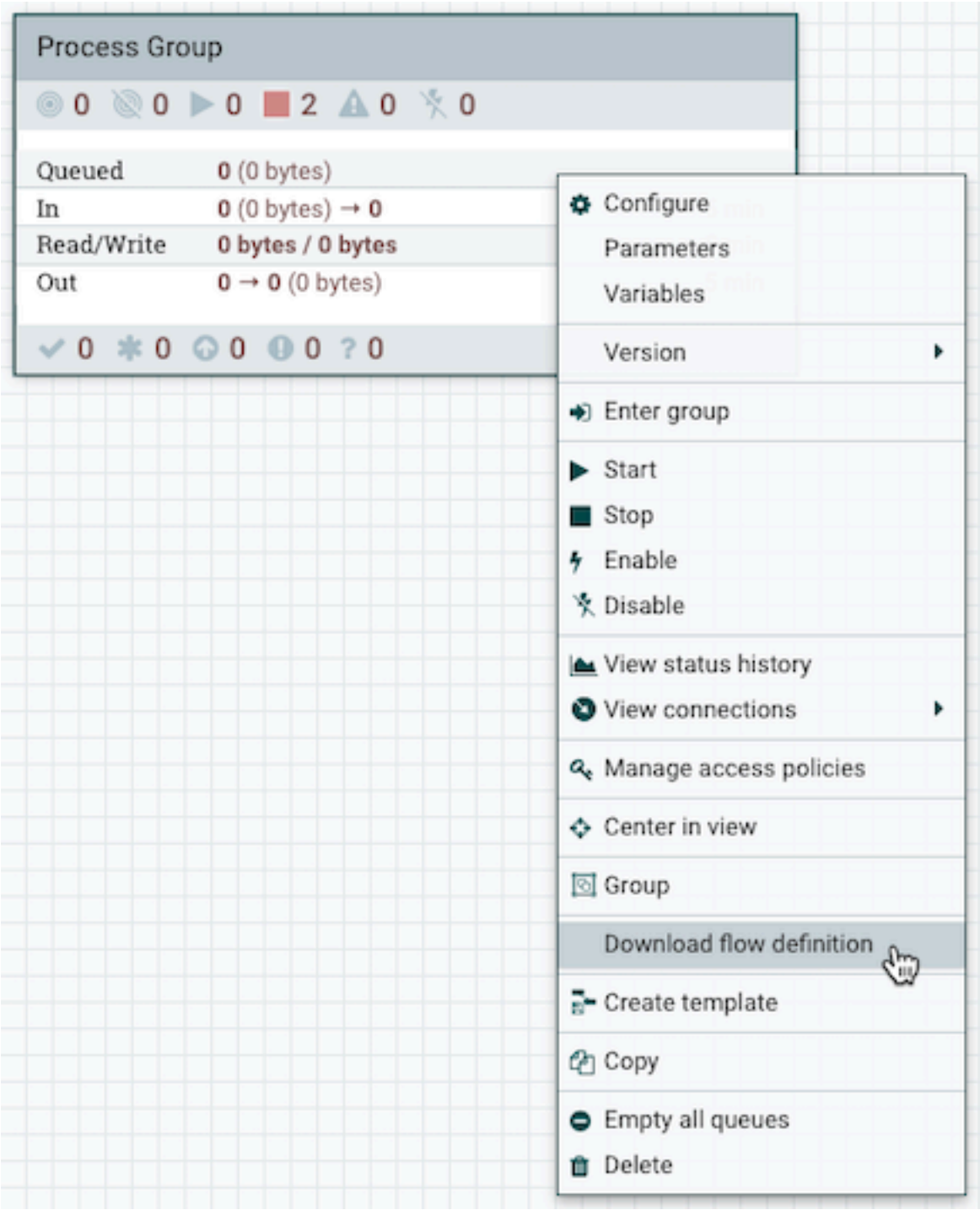

2. Enter the process group by double-clicking it.

   Alternatively, you can select the Enter group option from the context menu.

3. Add the appropriate flow components to the NiFi canvas.

   You can add processors and other components to build your data flow. To add a processor to your flow, drag the processor icon to the canvas and select the name of the processor from the list.

4. Configure the components in your data flow.

   Make sure that you externalize the component properties where values change depending on the environment in which the data flow is running. For more information on parameterizing your processor configurations, see *Best Practices for Flow Definition Development*.

   a) Create a parameter context for your data flow and add parameters to it.

      Using parameters for certain properties (for example, connection information, truststores, or drivers) makes the flow portable. You can download your data flow from NiFi and then import it to Cloudera DataFlow as a flow definition. When you deploy the flow definition, you can specify values or upload files for these parameters in the deployment wizard to adjust them to your needs.

      For instructions on how to set up your parameters, see *Best Practices for Flow Definition Development*.

   b) Create controller services for your flow.

      If you want to add controller services to your flow that you will later use in Cloudera DataFlow, you must define the controller services in the configuration of the process group that you will download as a flow definition. In this case the services will be available to all processors in that process group and will be available in Cloudera DataFlow as well, when you import your flow definition.

      Important:

      The only exception to the rule that controller services must be defined in the process group that you plan to export, is when you use the Default NiFi SSL Context Service controller service in Cloudera Data Hub Flow Management clusters. If a property references a controller service with that exact name, Default NiFi SSL Context Service, DataFlow creates a matching controller service when deploying the flow. The controller service points to a truststore that is setup with the environment's FreeIPA root certificate. This allows you to reference this controller service when interacting with Cloudera Public Cloud services that require TLS and are running in the same environment.

      For more information and instructions on how to set up your controller services, see *Best Practices for Flow Definition Development*.

   c) Configure each processor (and any other components) in your flow with the required values by double-clicking it.

      Alternatively, you can right-click the processor and select the Configure option from the processor's context menu.

      Parameterize component properties and use controller services in the configuration where needed. You can use the parameters you previously created or you can also create new parameters as you configure the components in your flow. To create a new parameter for a property, select the (Convert to Parameter) icon in the property's row.

      Note: To reference a parameter, the process group must first be assigned a parameter context. Processors and controller services of a process group can only reference parameters within the parameter context assigned to the process group.

      Important: A process group can only be assigned one parameter context, while a given parameter context can be assigned to multiple process groups.

   d) After configuring the processors and other data flow components, click Apply.

5. Connect the components in the data flow and configure the connections.
6. Check your data flow to make sure that configuration of all components and connections is valid.

Results

Your NiFi flow is ready to be downloaded as a JSON file.

What to do next

Download the data flow as a flow definition from NiFi and import it to Cloudera DataFlow.

Related Information

Setting up your Flow Management cluster
Best Practices for Developing Flow Definitions
Downloading a Flow Definition
Importing Flow Definitions
Deploying Flows
Parameters in Apache NiFi User Guide

Adding Inbound Connection support to a NiFi flow

You can enable a dataflow to listen on a public endpoint to data sources that are outside your Cloudera Public Cloud environment by adding inbound connection support to your flow

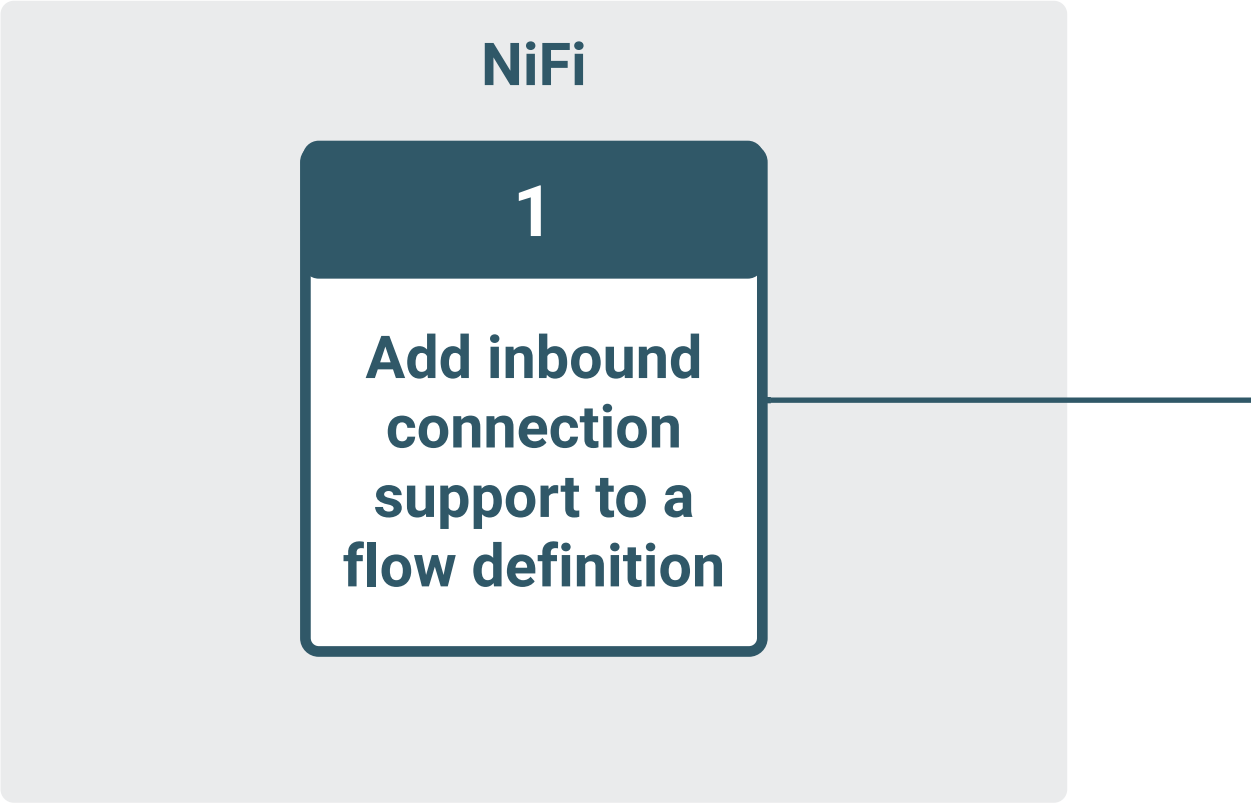

definition.

To enable your dataflow to use listen processors with inbound connection endpoint support, make the following addition to your flow definition:

1. In NiFi, open the flow definition where you want to enable inbound connection support.
2. Add the required listen processor to your flow definintion.

   Cloudera DataFlow supports all listen processors, including Custom Processors.

3. Configure the processor to enable inbound connections.

   Port

   Provide a port number where the flow deployment listens for incoming data. You can either parameterize it and set the actual port number during flow deployment, or you can set an explicit port number when designing the flow.

   SSL Context Service

   Create an external StandardRestrictedSSLContextService for your processor. You must name this context service Inbound SSL Context Service. No other configuration is required. The SSL context service will be created during cluster deployment and all other properties will be populated with values generated for that NiFi cluster.

   Client Auth

   Set to “REQUIRED” to use mTLS

   Note: Most listen processors only support TCP. ListenSyslog supports both TCP and UDP. As dataflow deployments do not support mixed protocols for listening ports, and UDP does not support SSL Context Service and Client authentication, Cloudera advises to configure your ListenSyslog processor to use TCP protocol. This allows your Cloudera DataFlow dataflow to use SSL Context Service for authentication and to listen to different data sources on different ports.

   If this is not possible, create separate flows that listen on UDP and TCP respectively.

4. Save and download the updated flow definition.

Download a flow definition from NiFi

Before you can run an Apache NiFi flow in Cloudera DataFlow, you need to download the NiFi flow as a flow definition. The Download flow definition option in NiFi allows you to download the flow definition of a process group as a JSON file. The file can be imported into Cloudera DataFlow using the Import Flow Definition option.

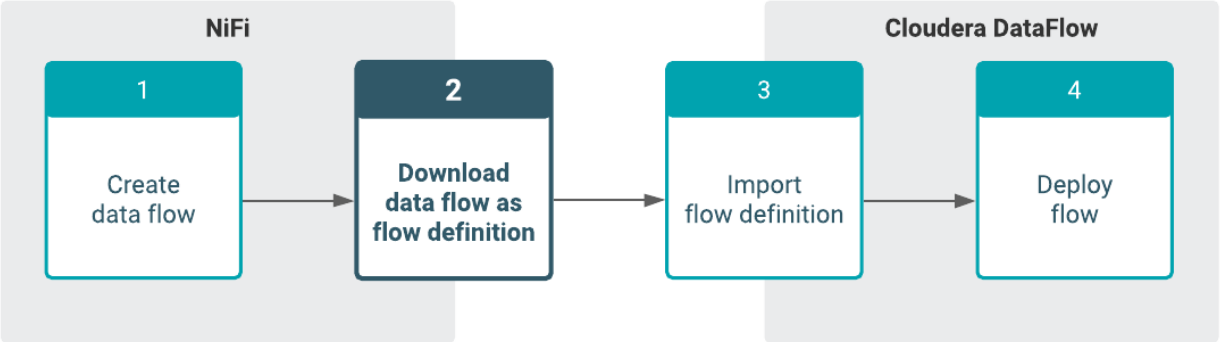

Before you begin

- Review *Best Practices for Flow Definition Development* to ensure that you have a good understanding of flow isolation best practices.
- Develop your data flow in NiFi.

Important:

When you download your data flow from NiFi:

- Default values are exported with the flow definition. When you import this flow definition to Cloudera DataFlow and want to deploy your flow, the default values are carried over into the deployment wizard.
- Sensitive values are removed and you have to set them again during flow deployment in Cloudera DataFlow.
- Empty strings remain empty.

Procedure

1. Select the process group that you want to transfer to Cloudera DataFlow.

   For recommendations on how to export your process groups for deployment in Cloudera DataFlow, see *Best Practices for Flow Definition Development*.
2. Right-click the process group.

3. Select Download flow definition.

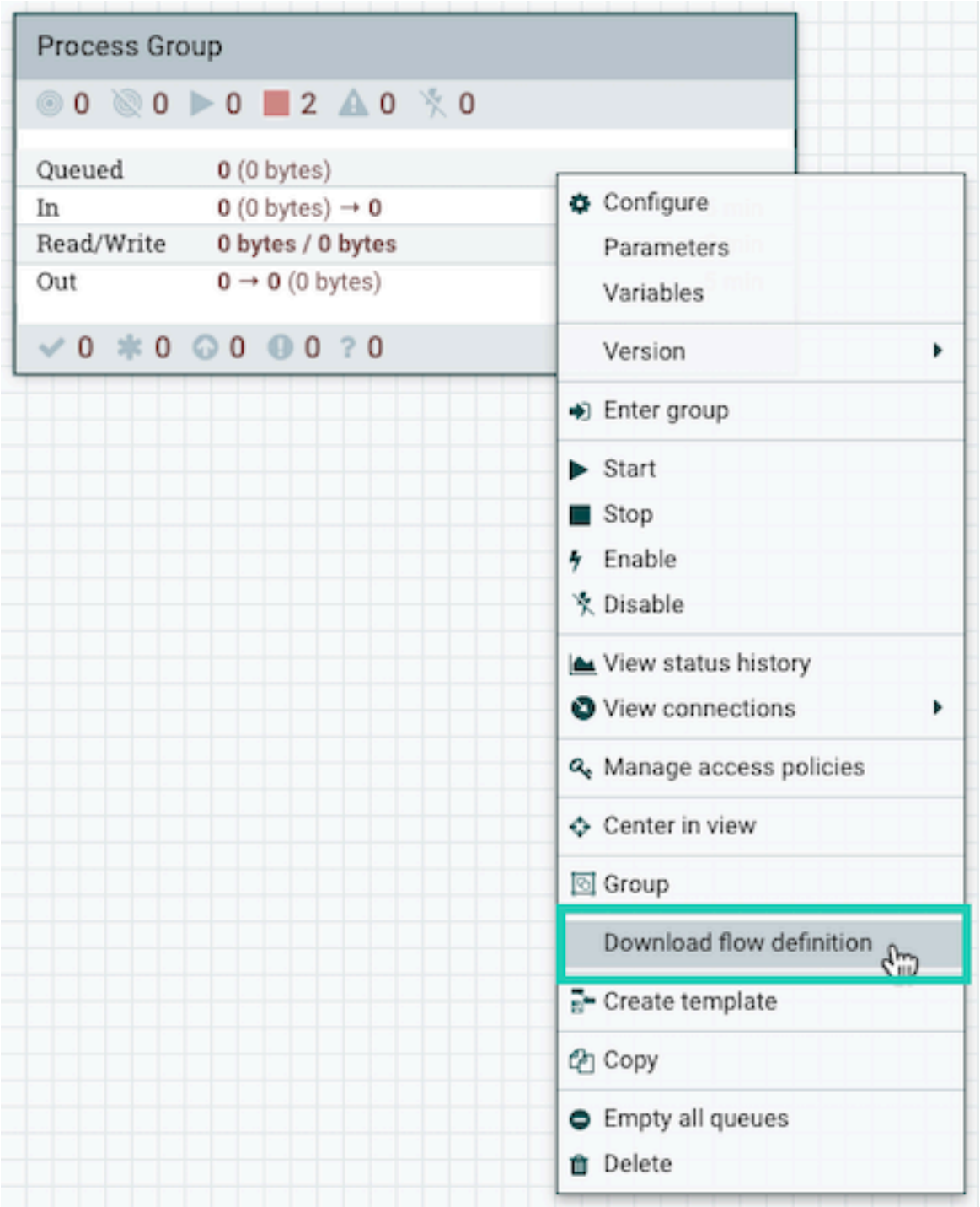

Note:

You can download flow definitions from NiFi at any level of your process group hierarchy. If you download one particular process group, your package contains the flow(s) included in the selected process group. When you download your flow definition from the root canvas level, you basically export all your data flows running on a cluster in one package, as the root canvas is the top level in the process group hierarchy.

If you download a versioned process group, the versioning information is not included in the downloaded JSON file.

After the download, you can upload your flow to Cloudera DataFlow.

For more information on how to plan and organize your NiFi data flows, see *Best Practices for Flow Definition Development*.

Results

The data flow is downloaded as a JSON file. You have a portable NiFi flow ready to be imported into Cloudera DataFlow.

What to do next

Import the NiFi flow to Cloudera DataFlow as a flow definition.

Related Information

Creating a Flow Definition

Best Practices for Flow Definition Development

Importing Flow Definitions

Deploying Flows

Cloudera DataFlow

Enabling DataFlow for an Environment

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enabling Cloudera DataFlow for an environment.....4

Setting up service and deployment notifications.....8

Setting up notifications while enabling Cloudera DataFlow for an environment.....9
Setting up notifications in a running service.....11
Setting up announcements for notifications.....12
Sharing a subscription link with users without the DFAdmin role.....12

Enabling Cloudera DataFlow for an environment

Enabling Cloudera DataFlow for an environment is your first step in getting started with Cloudera DataFlow. To do this, ensure that you have met the prerequisites, and then launch the Enable Environment window to walk you through the process.

Before you begin

- You have an AWS or an Azure account.
- You have prepared your infrastructure and network. For more information on requirements, see the *AWS Resource Planning* or the *Azure Resource Planning* documentation respectively.
- You have created and registered a Cloudera Public Cloud environment. For more information, see the *Data Hub documentation*.
- FreeIPA is running and healthy.
- Your Data Lake is started and healthy.
- You have the DFAdmin role for the environment you want to enable. For more information, see the *Cloudera DataFlow Security* documentation

For UI

Steps

1. From the Cloudera Public Cloud home page, click Cloudera DataFlow, then click Environments.
2. Find the environment you want to enable, and click Enable to launch the Environments / Enable DataFlow window.

   If the Enable button is grayed out, hover over the Not Enabled icon for more details about the problem.

   Note:

   If you have previously disabled the environment but preserved event history, go to Actions and select Enable Environment.
3. Configure **DataFlow Capacity**.

   This defines the minimum and maximum size of the Kubernetes (K8s) cluster. Your Cloudera DataFlow cluster automatically scales between the minimum and maximum cluster size that you specify here.

   Note: You can specify a maximum capacity of 50 K8s nodes.

4. Configure **Networking**.
   a. Specify whether to use a Public Endpoint.

      Select this option when you want to allow users to connect to workload side UIs like the Cloudera DataFlow Deployment Manager or the actual NiFi UI through the public Internet.

      - If checked when enabling an AWS environment, this option provisions an endpoint (load balancer) in a public subnet.

        Note: A public endpoint requires public subnets. If you want to place a public endpoint in private subnets, you must explicitly select them.

        If checked when enabling an Azure environment, this option provisions an endpoint (load balancer) in a public subnet and you cannot explicitly specify subnets. The Load Balancer Subnet Use option is not available.
      - If unchecked, Cloudera Public Cloud creates an endpoint in a private subnet, and you must set up access to the endpoint manually in your cloud account to allow user access to workload side UIs.

   b. Specify Load Balancer Subnets.

      Select from Available Subnets. Explicitly specifying subnets overrides the automatic, tag-based subnet selection process and ensures load balancer provisioning in the specified subnets. If no subnets are specified, Cloudera DataFlow provisions a load balancer according to how the subnets have been tagged. For more information, see *VPC and subnets* in the *Related information* section below.

   c. Specify Worker Node Subnets.

      Select one of the Available Subnets. If you do not make a subnet selection, Cloudera DataFlow considers any available subnet that has been registered with the environment for worker placement. Worker nodes are only placed in public subnets if no private subnets are available.

   d. Specify Load Balancer Endpoint Access.

      Specify a set of IP address ranges that will be allowed to access the Cloudera DataFlow load balancer endpoint. Providing no IP address ranges makes the load balancer endpoint open to all traffic.

5. Configure **Kubernetes**.
   a. Configure API Server Endpoint Access.

      Cloudera Public Cloud environments with Cluster Connectivity Manager enabled support the ability to create a fully private cluster, which disallows all access to the Kubernetes API Server Endpoint.

      - Specify whether to use a Private Cluster.

        If you select Private Cluster IP based access is not applicable, and the option to specify a set of IP address ranges is not available.

        If you do not select Private Cluster you can specify a set of IP address ranges that are allowed to access the Kubernetes API Server Endpoint. Providing no Classless Inter-Domain Routing (CIDR) makes the Kubernetes API Server Endpoint open to all traffic.

        In either case, any user who needs access to the Kubernetes API Server must be granted remote access to the underlying Kubernetes cluster. This can be configured after Cloudera DataFlow has been enabled successfully.

        Note: If you select Private Cluster while enabling an Azure environment, the UDR Mode option becomes available. If you enable UDR mode, make sure that under Networking Worker Node Subnets you selected the specific worker node subnet where UDR mode has been configured.

   b. Configure Pod CIDR Range.

      The CIDR notation IP range from which to assign IPs to pods in your kubernetes cluster. This address should be a large address space that is not in use elsewhere in your network environment, accommodating the number of nodes that you expect to scale up to. The default value is 10.244.0.0/16. The value is used to assign a /24 address space to each node in the cluster. For example, the first node would be assigned 10.2

44.0.0/24, the second node 10.244.1.0/24, the third node 10.244.2.0/24. As the cluster scales or upgrades, the platform continues to assign a pod IP address range to each new node.

c. Configure Service CIDR Range.

The CIDR notation IP range from which to assign IPs to internal services in your kubernetes cluster. This IP address range should be an address space that is not in use elsewhere in your network environment, accommodating the amount of kubernetes services that you intend to use. The default value is 10.0.0.0/16.

6. Configure Tags as keys and their values.

Tags are added to Cloudera DataFlow resources at the time of enablement. These tags are included in addition to those set by the Cloudera Public Cloud service.

7. Configure Notifications.

Select the types of events for which you want to receive notifications on the Cloudera Management Console.

DataFlow Service

If you enable this option, you will receive notifications about enablement progress, scaling operations, and health for this Cloudera DataFlow service. Select the event severity levels you want to receive notifications for (Info, Warning, Error). Cloudera recommends you keep notifications enabled for all event severities.

All Deployments

If you enable this option, you will receive notifications for deployment progress, scaling operations, and health for all deployments in this workspace. Select the event severity levels you want to receive notifications for (Info, Warning, Error). Cloudera recommends you keep notifications enabled for all event severities.

You can configure email notifications for the events you have subscribed to after the environment is enabled for Cloudera DataFlow.

8. Click Enable. This may take up to 45 minutes.

Result

Your cluster status changes from Not Enabled to Enabling.

- Hover over Enabling for environment enablement event messages to display.
- Click the Alerts tab to see environment enablement event messages.
- Click anywhere in your environment row to see your environment details.

For CLI

Before you begin

- You have installed the CDP CLI.
- You have run cdp environments list-environments # to obtain the environment-crn.

Steps

1. To enable Cloudera DataFlow for an environment, enter:

```
cdp df enable-service
--environment-crn [***ENVIRONMENT_CRN***]
--min-k8s-node-count [***MIN_K8S_NODE_COUNT***]
--max-k8s-node-count [***MAX_K8S_NODE_COUNT***]
[--use-public-load-balancer] [--no-use-public-load-balancer]
[--private-cluster] [--no-private-cluster]
[--kube-api-authorized-ip-ranges
 [[***KUBE_API_AUTHORIZED_IP_RANGES***[ [[***KUBE_API_AUTHORIZED_IP_RANGES***[ ..
.]]]
[--tags [***TAGS***]]
[--load-balancer-authorized-ip-ranges
 [[***LOAD_BALANCER_AUTHORIZED_IP_RANGES***]
 [[***LOAD_BALANCER_AUTHORIZED_IP_RANGES***] ...]]]
```

```
[--cluster-subnets [[***CLUSTER_SUBNETS***] [[***CLUSTER_SUBNETS***] ...
]]]
[--load-balancer-subnets [[***LOAD_BALANCER_SUBNETS***]
 [[***LOAD_BALANCER_SUBNETS***] ...]]]
[--send-events-to-notification] [--no-send-events-to-notification]
[ --notification-severity-levels [[***NOTIFICATION_SEVERITY_LEVELS***]
 [[***NOTIFICATION_SEVERITY_LEVELS***] ...]]]
[--send-collective-flows-events-to-notification][--no-send-collective-fl
ows-events-to-notification]
 [--collective-flows-notification-severity-levels
 [[***COLLECTIVE_FLOWS_NOTIFICATION_SEVERITY_LEVELS***]
 [[***COLLECTIVE_FLOWS_NOTIFICATION_SEVERITY_LEVELS***] ...]]]
```

Where:

- --environment-crn specifies the environment-crn you obtained while completing the pre-requisites.
- --min-k8s-node-count and --max-k8s-node-count specify your Cloudera DataFlow capacity. Your Cloudera DataFlow cluster automatically scales between the minimum and maximum cluster size that you specify here.
- [--use-public-load-balancer] [--no-use-public-load-balancer]
- [--private-cluster] [--no-private-cluster]
- --kube-api-authorized-ip-ranges specifies a set of IP address ranges that will be allowed to access the Kubernetes API Server Endpoint. Providing no IP address ranges makes the Kubernetes API Server Endpoint open to all traffic.
- --tags specifies any tags you want to add to Cloudera DataFlow resources at the time of enablement. These tags are in addition to those set by the Cloudera service.
- --load-balancer-authorized-ip-ranges
- --cluster-subnets
- --load-balancer-subnets
- [--send-events-to-notification], [--no-send-events-to-notification] define whether you want to receive notifications on Cloudera Management Console about environment level events and --notification-severity-levels defines the types of events you want to receive.
- [--send-collective-flows-events-to-notification], [--no-send-collective-flows-events-to-notification] define whether you want to receive event notifications about all deployments in the environment and --collective-flows-notification-severity-levels defines the types of events you want to receive.

Example

Successfully enabling Cloudera DataFlow for an AWS environment results in output similar to:

```
{
  "service": {
    "kubeApiAuthorizedIpRanges": [],
    "validActions": [
      "ENABLE"
    ],
    "loadBalancerAuthorizedIpRanges": [],
    "clusterSubnets": [],
    "loadBalancerSubnets": [],
    "tags": {},
    "crn": "crn:cdp:df:us-west-1:CLOUDERA:service:96827a6b-bd8b-42fb-85ad-
bdb4fc7ca43d",
    "environmentCrn": "crn:cdp:environments:us-west-1:CLOUDERA:environment
:dev-east-local-k8s",
    "name": "dev-east-local-k8s",
    "cloudPlatform": "LOCAL_K8S",
    "region": "local-k8s",
    "deploymentCount": 0,
    "minK8sNodeCount": 3,
    "maxK8sNodeCount": 4,
```

```
    "status": {
      "state": "ENABLING",
      "message": "Enabling DataFlow crn:cdp:environments:us-west-1:CLOUDER
A:environment:dev-east-local-k8s",
      "detailedState": "NEW"
    },
    "runningK8sNodeCount": 0,
    "instanceType": "m5.xlarge",
    "dfLocalUrl": "",
    "activeWarningAlertCount": "0",
    "activeErrorAlertCount": "0",
    "clusterId": "",
    "clusterUsable": false,
    "usePublicLoadBalancer": false,
    "usePrivateCluster": false,
    "creatingK8sNodeCount": 0,
    "terminatingK8sNodeCount": 0
  }
}
```


What to do next

After the environment is enabled for Cloudera DataFlow, you can subscribe to receiving email notifications on events you have configured during enablement. For more information, see the Cloudera Management Console documentation.

Once you have enabled your Cloudera DataFlow environment, you are ready to deploy your first flow definition from the Catalog. For instructions, see *Deploying a flow definition*.

For more information on managing and monitoring Cloudera DataFlow, see *Managing Cloudera DataFlow in an environment*.

Related Information

AWS Resource Planning

Azure Resource Planning

Data Hub documentation

Cloudera DataFlow security

VPC and subnets in AWS environments

VNet and subnets in Azure environments

Deploying a flow definition

Managing Cloudera DataFlow in an environment

Setting up service and deployment notifications

You can configure to send notifications about Cloudera DataFlow service and deployment events to the Cloudera Management Console. You can also subscribe to email notifications on these events.

You can enable notifications for a Cloudera DataFlow service, and for all deployments within that service. These notifications are independent of each other, you can configure just one or both of them. You can either configure notifications while enabling Cloudera DataFlow for an environment, or at a later time, for a running environment.

Configuring notifications only enables Cloudera DataFlow to send the configured event type notifications to the Cloudera Management Console that you can view in-app. If you want to receive email announcements of these notifications, you need to additionally set up a subscription.

Both the description of the notification in the Cloudera Management Console and the email announcement contains a URL that allows you to easily navigate to your Cloudera DataFlow deployment or service.

Note:

The URL only works for you if you have access to the resource that the alert originated from.

Related Information

Receiving notifications | Management Console

Setting up notifications while enabling Cloudera DataFlow for an environment

Learn how you can configure to send notifications of Cloudera DataFlow service and deployment events to the Cloudera Management Console while enabling Cloudera DataFlow for an environment.

Before you begin

To set up notifications you must have the DFAdmin role for the environment.

Procedure

- During the environment enablement process, select the types of events for which you want to receive notifications on the Cloudera Management Console.

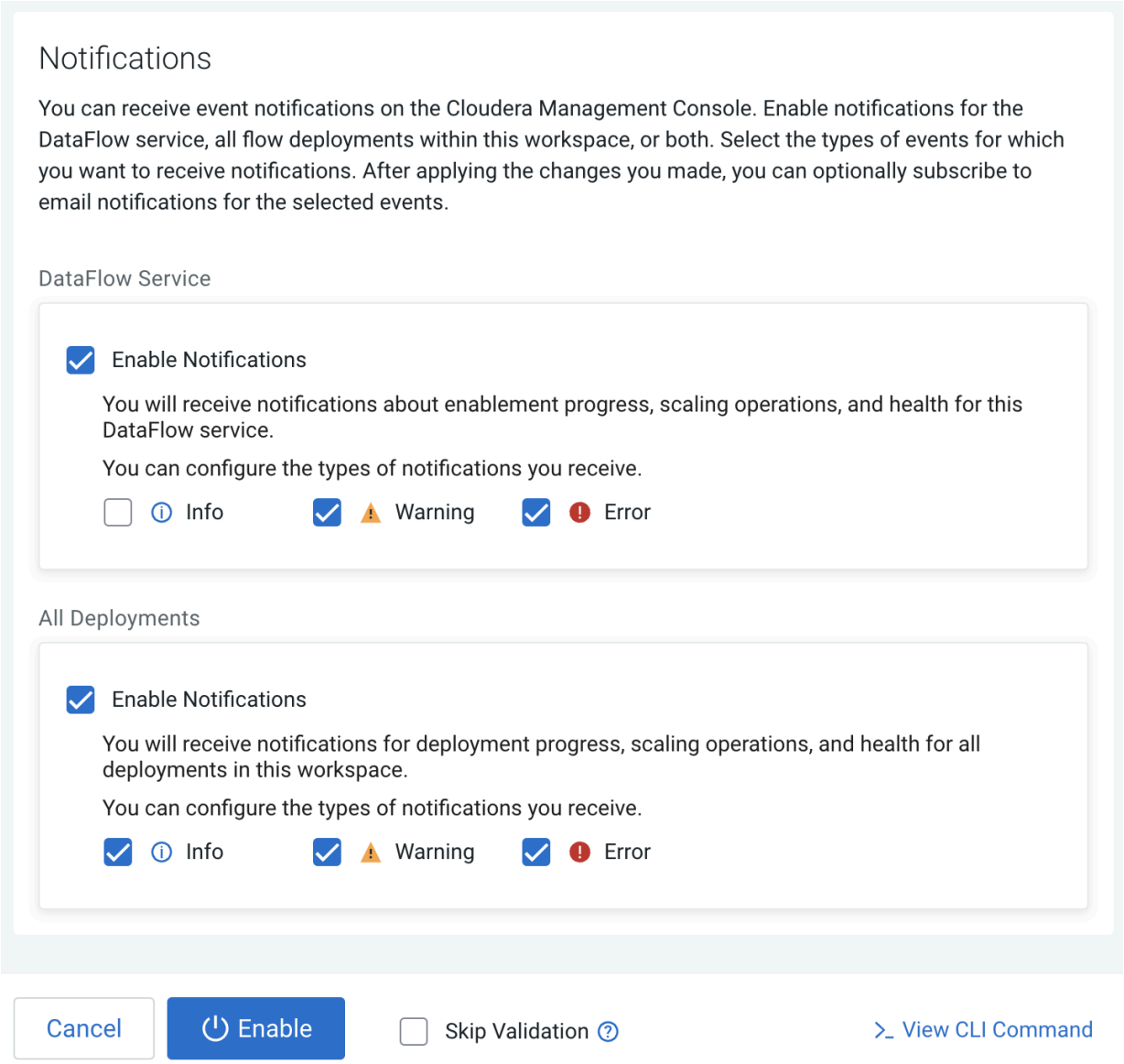

DataFlow Service

If you enable this option, you will receive notifications about all other alerts related to this Cloudera DataFlow service. Select the event severity levels you want to receive notifications for (Info, Warning, Error).

All Deployments

If you enable this option, you will receive notifications for all the alerts related to all of the NiFi flow deployments in this Cloudera DataFlow service. Select the event severity levels you want to receive notifications for (Info, Warning, Error).

What to do next

After the environment is enabled for Cloudera DataFlow, you can configure receiving email notifications.

Setting up notifications in a running service

Learn how you can configure to send notifications of Cloudera DataFlow service and deployment events to the Cloudera Management Console in a running Cloudera DataFlow service.

Before you begin

You have the DFAdmin role for the environment.

Procedure

1. In Cloudera DataFlow, from the **Environments** page, select the enabled Cloudera DataFlow service where you want to set up notifications.
2. Click Manage DataFlow from the **Environment Details** pane.

   You are redirected to the **Manage DataFlow** page.
3. Under **DataFlow Settings**, select the **Notifications** tab.

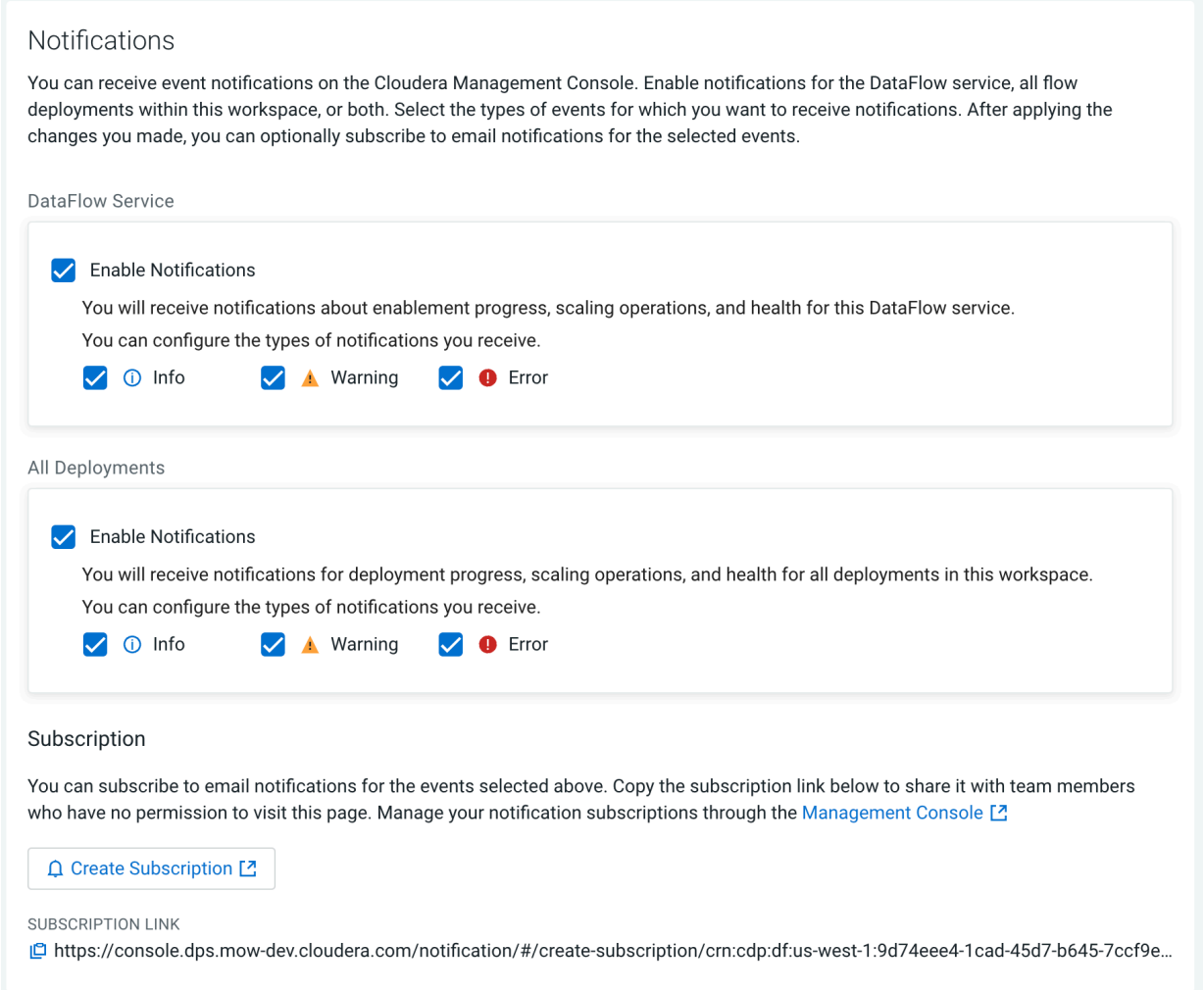

4. Select the types of events for which you want to receive notifications on the Cloudera Management Console.

   DataFlow Service

   If you enable this option, you will receive notifications about all other alerts related to this Cloudera DataFlow environment. Select the event severity levels you want to receive notifications for (Info, Warning, Error).

   All Deployments

   If you enable this option, you will receive notifications for all the alerts related to all of the Apache NiFi flow deployments in this environment. Select the event severity levels you want to receive notifications for (Info, Warning, Error).

5. Click Apply Changes.
6. Click Create Subscription if you want to create an email subscription now.

   You are redirected to the Cloudera Management Console where you can set up and configure your email subscription.

Setting up announcements for notifications

Learn about how to receive email announcements for events that you have subscribed to.

Before you begin

- You have the DFAdmin role for the environment.
- You have the NotificationSubscriber role for the organization.

Procedure

1. In Cloudera DataFlow, from the **Environments** page, select the environment where you want to set up notifications.
2. Click Manage DataFlow from the **Environment Details** pane.

   You are redirected to the **Manage DataFlow** page.
3. Under **DataFlow Settings**, select the **Notifications** tab.
4. Click Create Subscription.

   You are redirected to the Cloudera Management Console where you can configure email announcements.

   Note:

   You only receive notifications for events that occur after you have updated your preferences.

Related Tasks

Sharing a subscription link with users without the DFAdmin role

Sharing a subscription link with users without the DFAdmin role

Learn about providing access to email announcements for team members that have no access to the Environments view.

Before you begin

- To access the **Environments** view and share the subscription link, you must have the DFAdmin role for the environment.
- The users you share the subscription link with must have the NotificationSubscriber role for the organization.

Procedure

1. In Cloudera DataFlow, from the **Environments** page, select the environment for which you want to share notifications.
2. Click Manage DataFlow on the **Environment Details** pane.

   You are redirected to the **Manage DataFlow** page.
3. Under **DataFlow Settings**, select the **Notifications** tab.
4. Copy the SUBSCRIPTION LINK and share it with team members who do not have the DFAdmin role and therefore cannot access this page.

   They can subscribe to notifications as described in the Cloudera Management Console documentation.

Cloudera DataFlow

Tutorial: Build a New Flow

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Tutorial: Building a new flow from scratch..... 4
Create a new flow..... 4
Create controller services..... 8
Build your draft flow.....13
Start a test session..... 25
Publish your flow definition to the Catalog..... 27

Tutorial: Building a new flow from scratch

If you are new to flow design and have never used NiFi before, this tutorial is for you. Learn how to build a draft adding and configuring components, connecting them, creating Controller Services, and testing your flow while creating it.

About this task

This tutorial walks you through the creation of a simple flow design that retrieves the latest changes from Wikipedia through invoking the Wikipedia API. It converts JSON events to Avro, before filtering and routing them to two different processors which merge events together before a file is written to local disk.

You will learn about:

- Creating a draft
- Creating a Controller Service
- Adding processors to your draft
- Configuring processors
- Adding a user-defined property to a processor configuration
- Connecting processors to create relationships between them
- Running a Test Session
- Publishing a draft to the Catalog as a flow definition.

Before you begin

The flow you are about to build can be deployed without any external dependencies and does not require any parameter values during deployment. Still, there are prerequisites you have to meet before you can start building your first draft.

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFDeveloper role granting you access to the Flow Designer.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog. You will need this authorization to publish your draft as a flow definition to the Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.

Create a new flow

Create a new flow in a Flow Designer Workspace and give it a name.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.

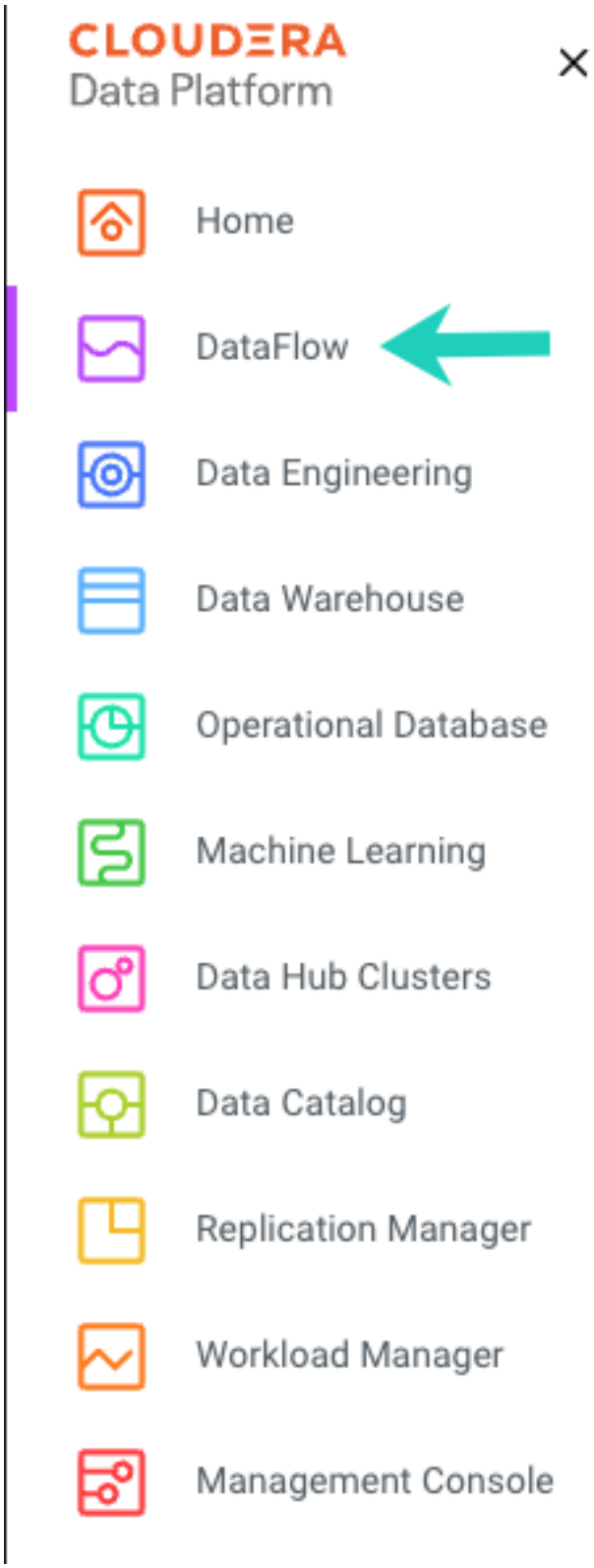

2. Click Flow Design in the left navigation pane.

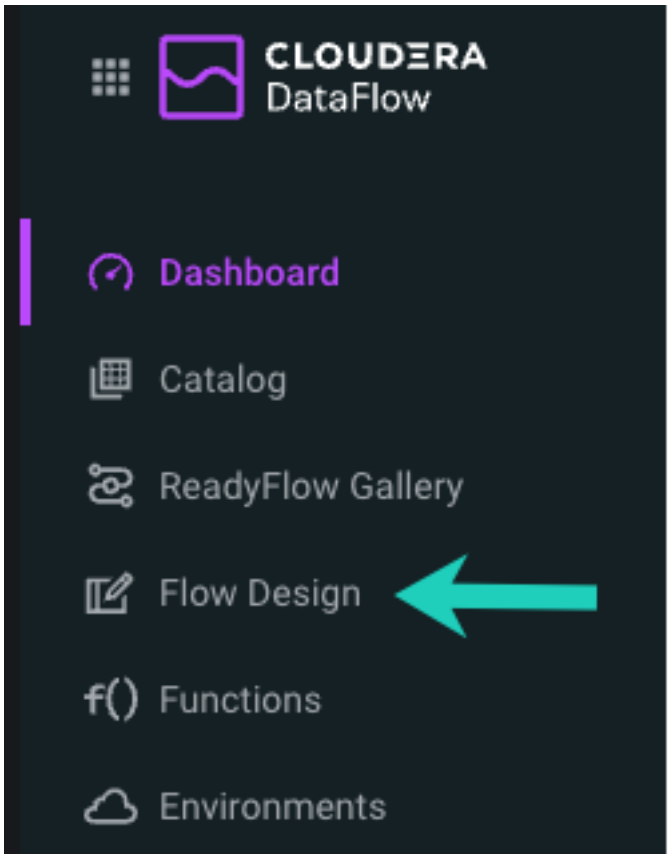

You are redirected to the **Flow Design** page, where previously created draft flows are displayed, one flow per row.

3. Click Create New Flow.

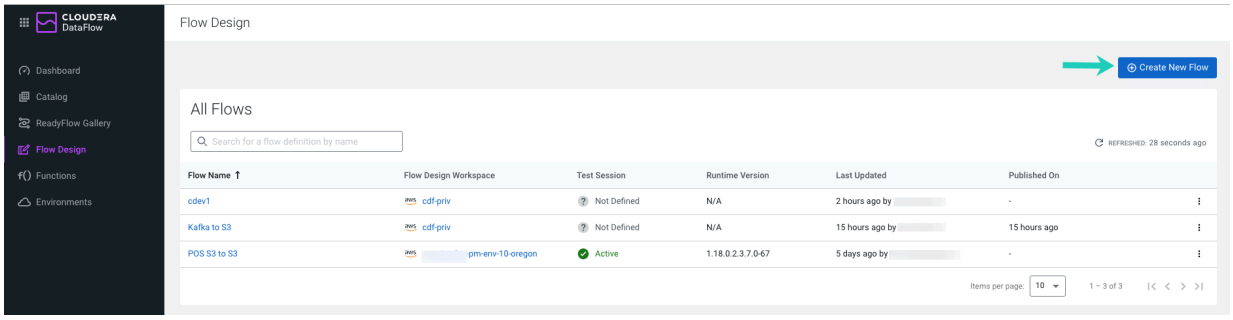

4. Select a Target Workspace where you want to create the draft.

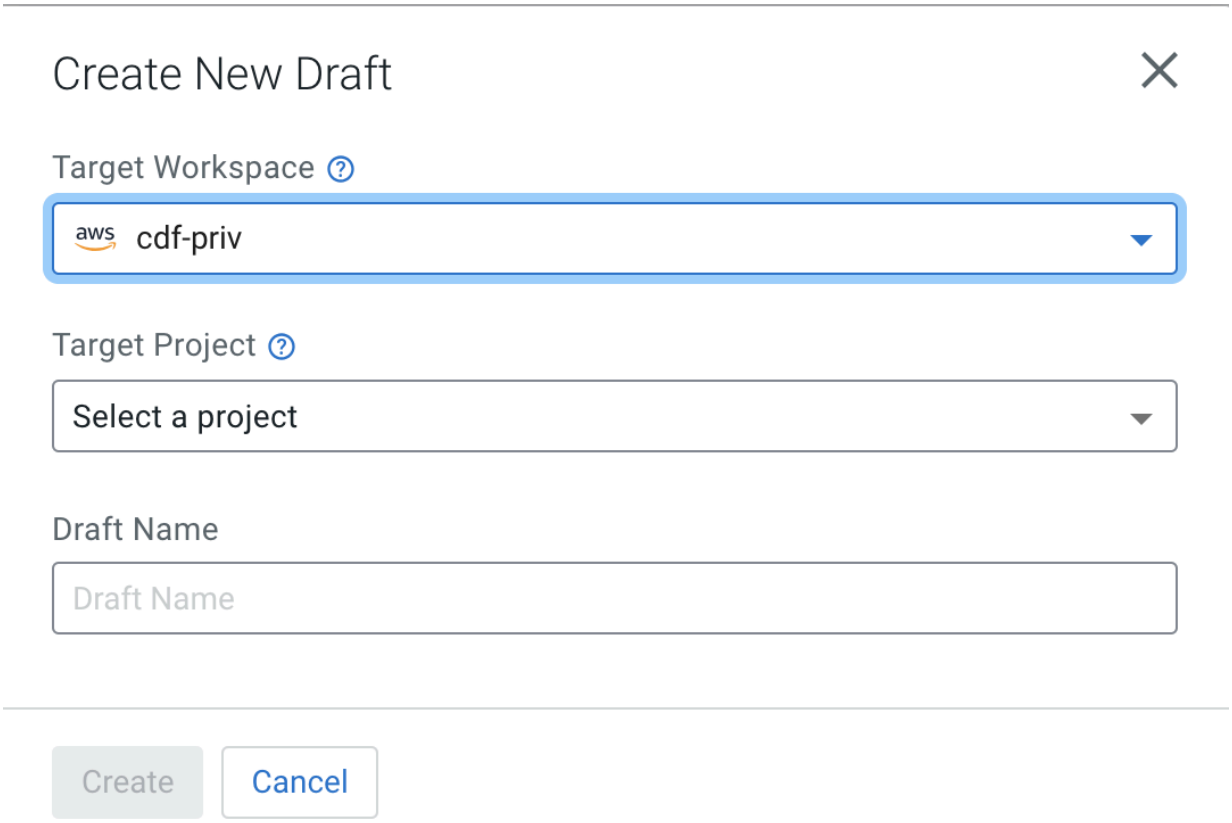

5. Under Target Project select Unassigned.
6. Provide a Draft Name.
   Provide Hello World.

7. Click Create.
   Flow Designer creates a default Process Group with the Draft Name you provided, 'Hello World' in this case, and you are redirected to the **Flow Design** canvas. The **Configuration** pane on the right displays configuration options for the default Process Group.

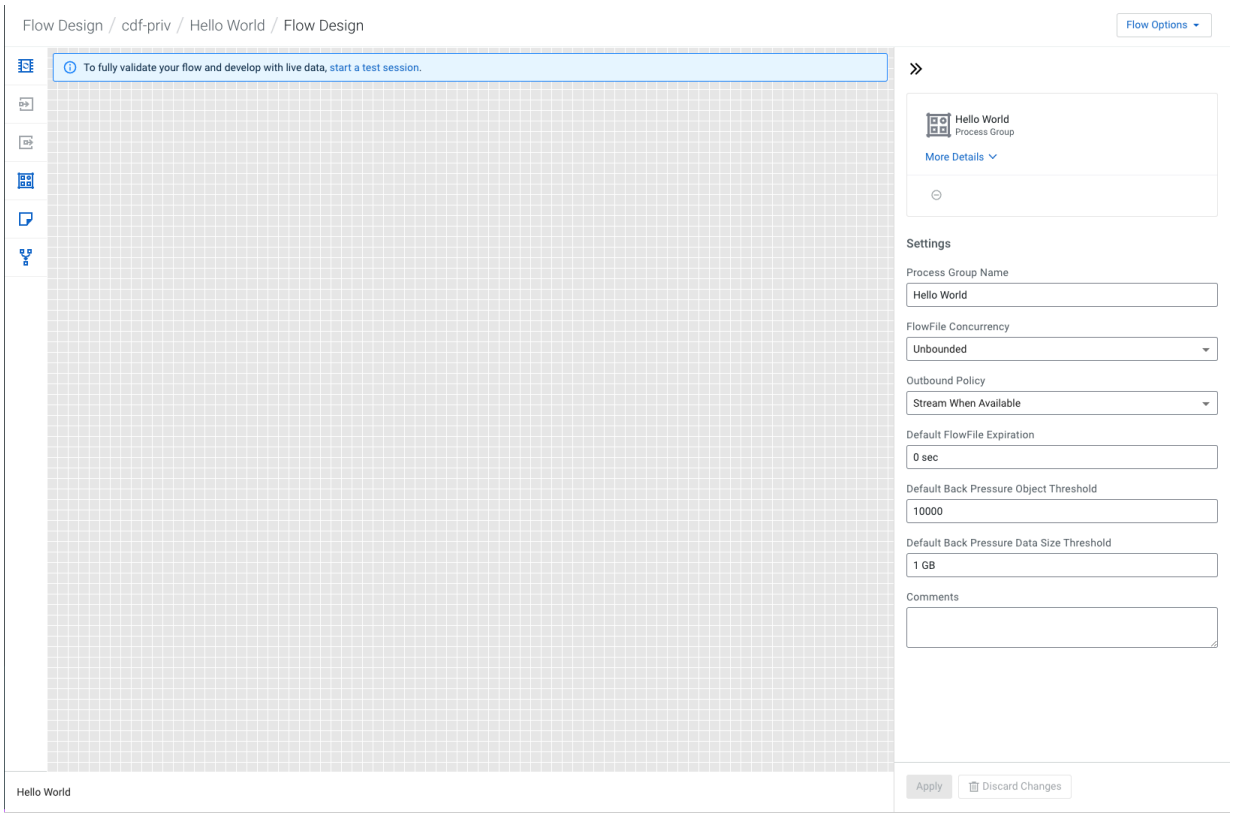

What to do next

Proceed to creating controller services.

Related Tasks

Create controller services

Create controller services

Learn about creating Controller Services in Cloudera DataFlow Flow Designer.

About this task

Controller Services are extension points that provide information for use by other components (such as processors or other controller services). The idea is that, rather than configure this information in every processor that might need it, the controller service provides it for any processor to use as needed.

You will use the controller services you create now to configure the behavior of several processors you will add to your flow as you are building it.

Procedure

1. Go to Flow Options Services .

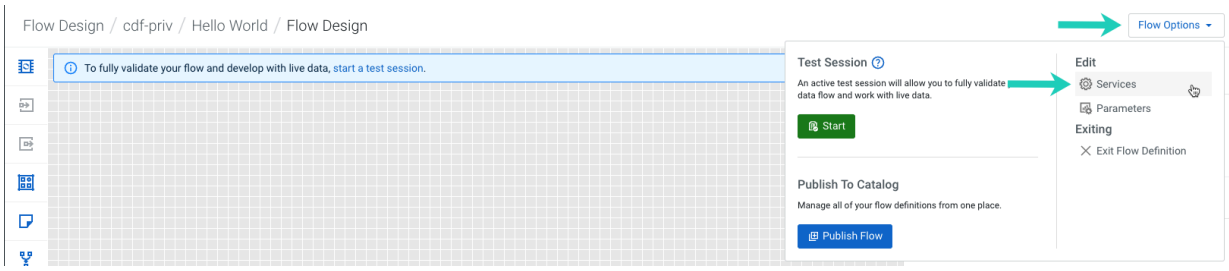

2. Click Add Service.
   The Add Service page opens.

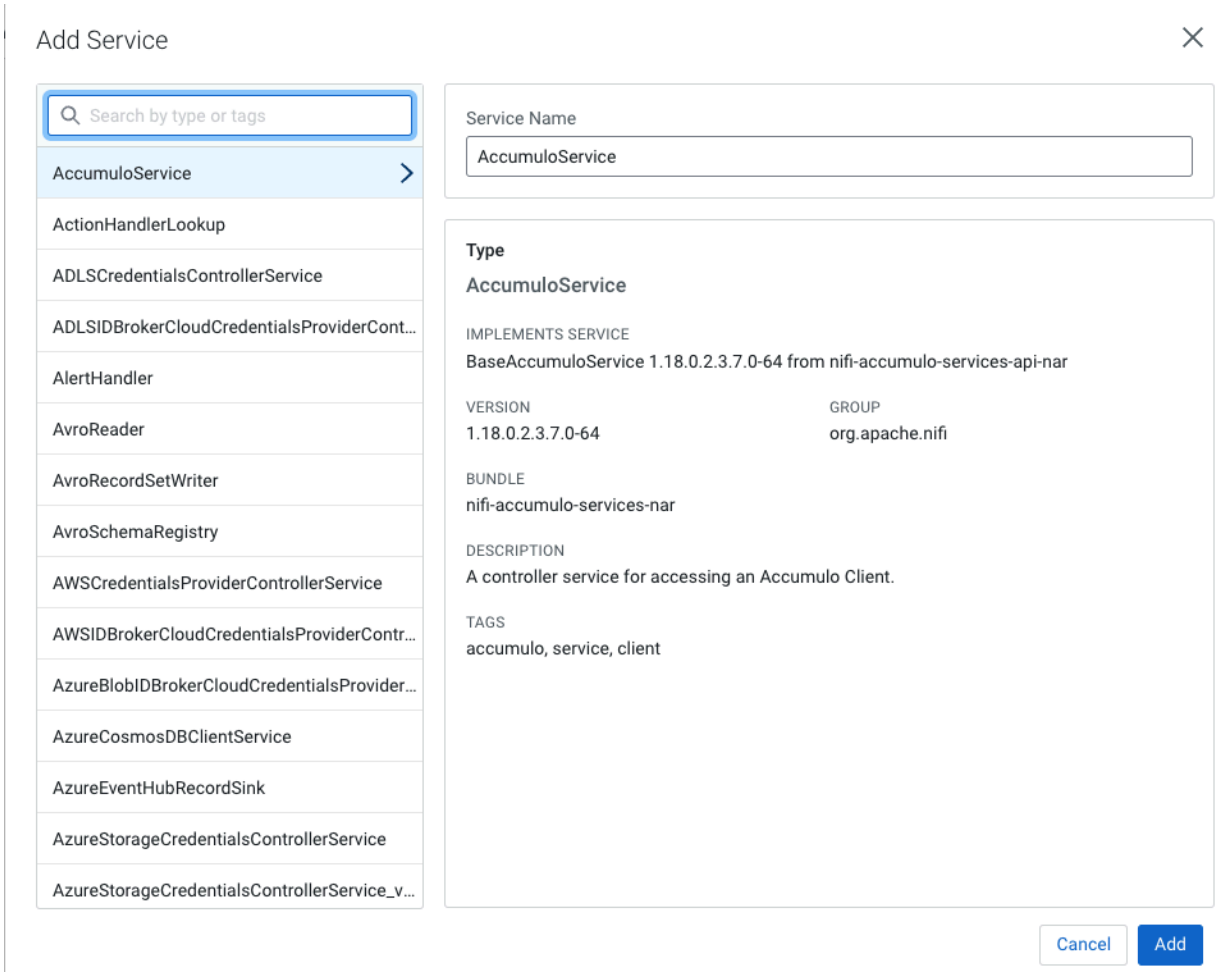

3. In the text box, filter for JsonTreeReader.

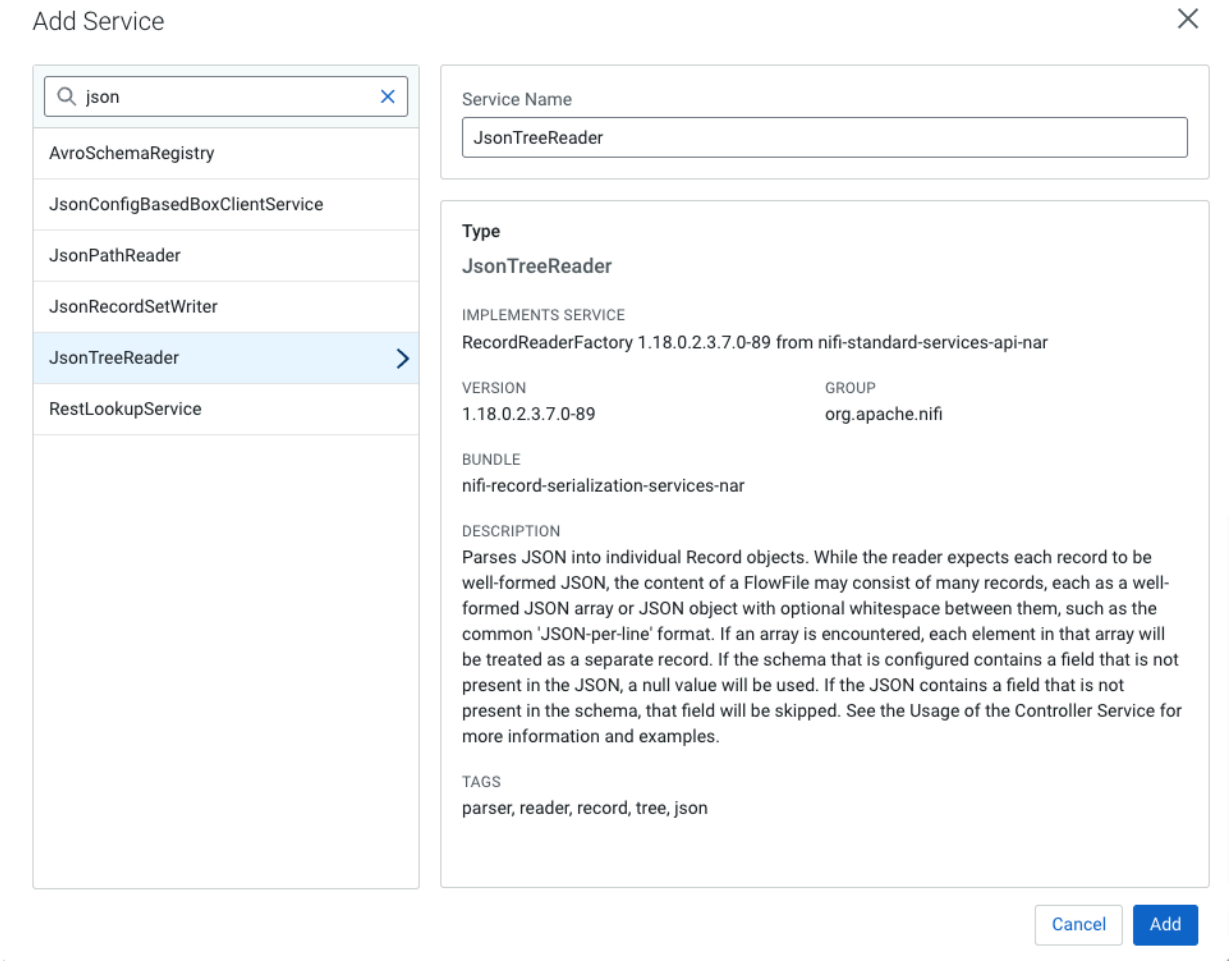

4. Provide Service Name: JSON_Reader_Recent_Changes.
5. Click Add.

6. Configure the JSON_Reader_Recent_Changes service.

   Set the following Properties:

   Starting Field Strategy

   Nested Field

   Starting Field Name

   recentchanges

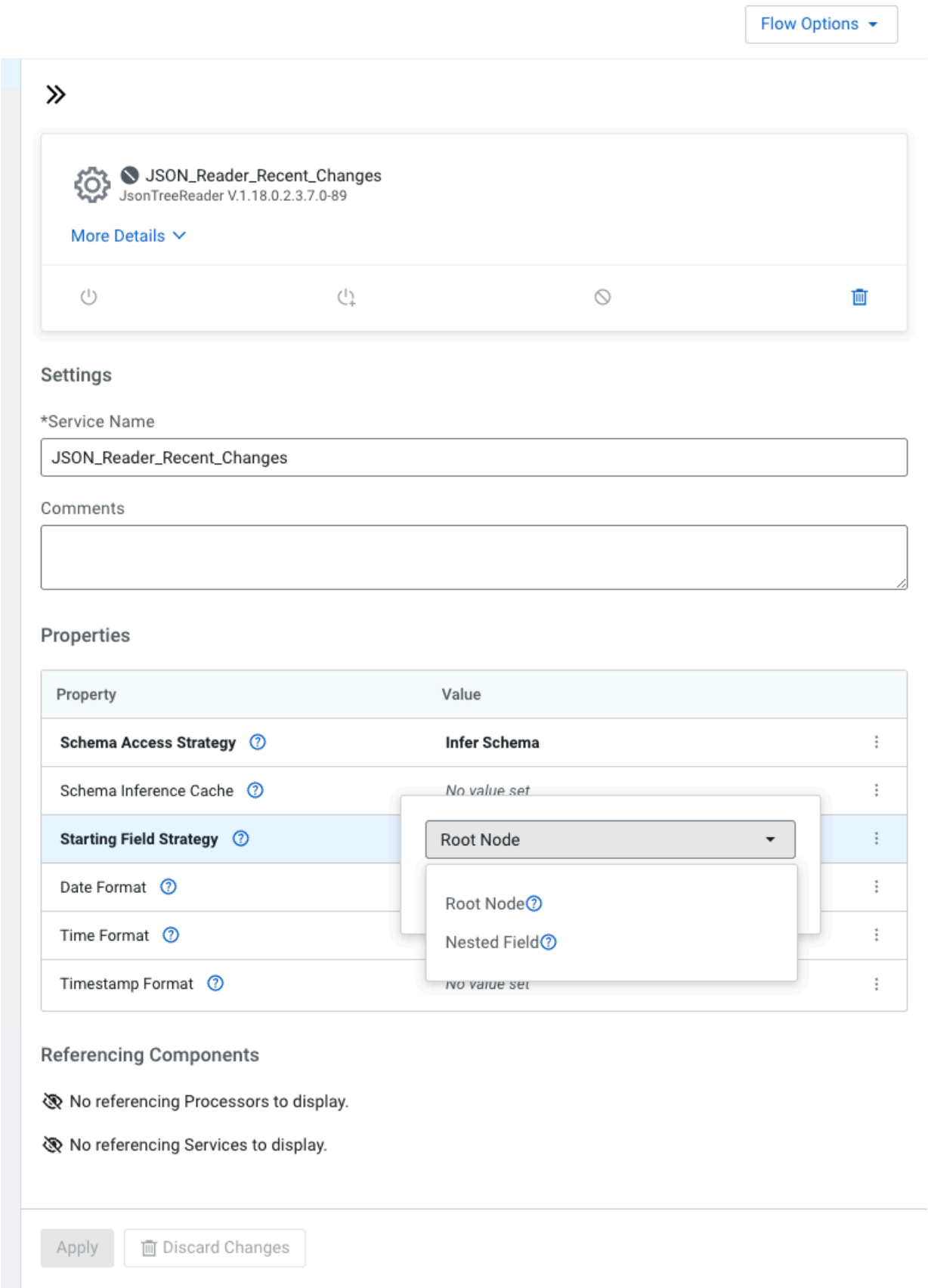

7. Click Apply.

8. Click Add Service to create another service.
9. In the text box, filter for AvroRecordSetWriter.
10. Provide Service Name: AvroWriter_Recent_Changes.
11. Click Add.

    You do not need to configure the AvroWriter_Recent_Changes service. You can leave all properties with their default values.
12. Click Add Service to create a third service.
13. In the text box, filter for AvroReader.
14. Provide Service Name: AvroReader_Recent_Changes.
15. Click Add.

    You do not need to configure the AvroReader_Recent_Changes service. You can leave all properties with their default values.
16. Click Back To Flow Designer to return to the flow design Canvas.

What to do next

After creating the necessary Controller Services, you can start building and configuring your flow.

Related Tasks

Build your draft flow

Build your draft flow

Start building your draft flow by adding components to the Canvas and configuring them.

Procedure

1. Add an InvokeHTTP processor to the canvas.
   a) Drag a Processor from the Components sidebar to the canvas.

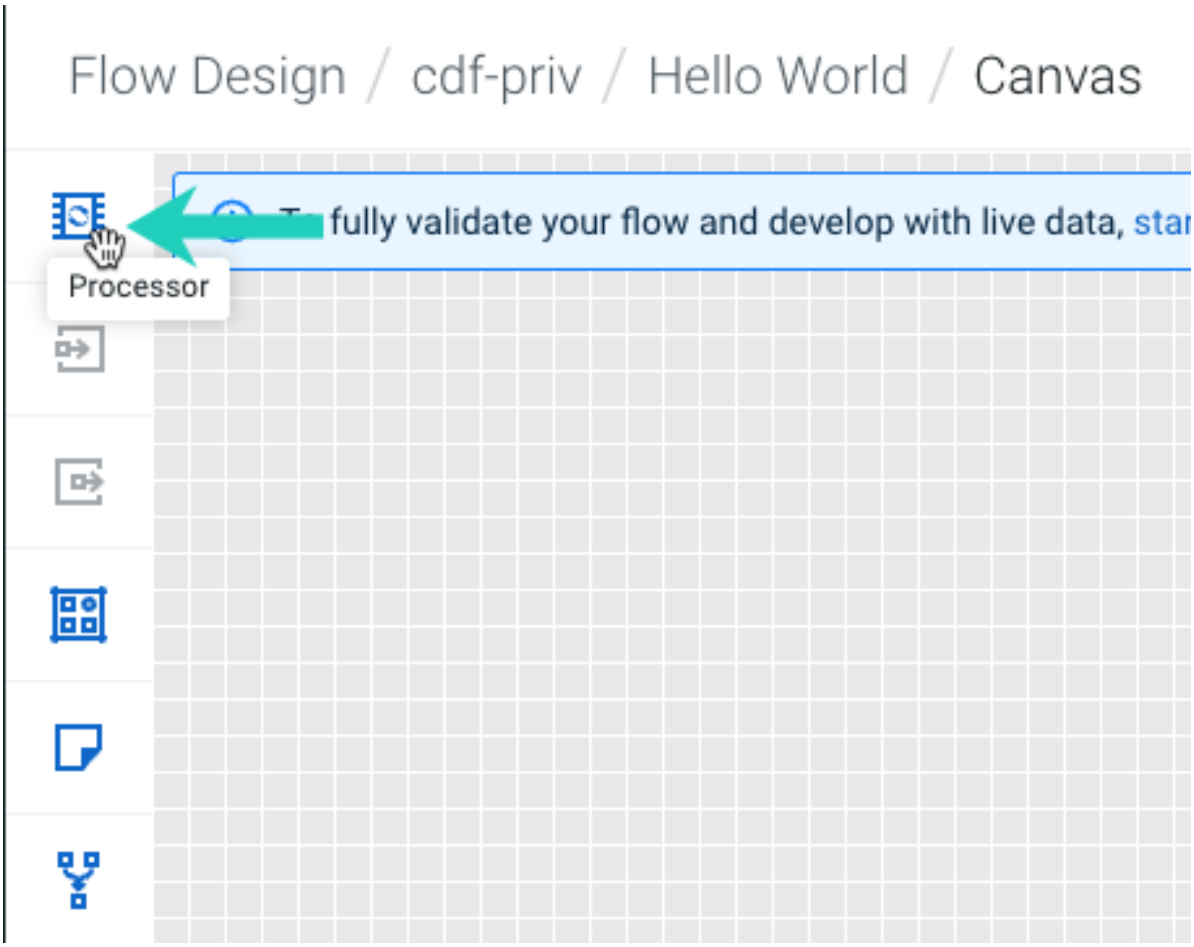

   b) In the text box, filter for InvokeHTTP.

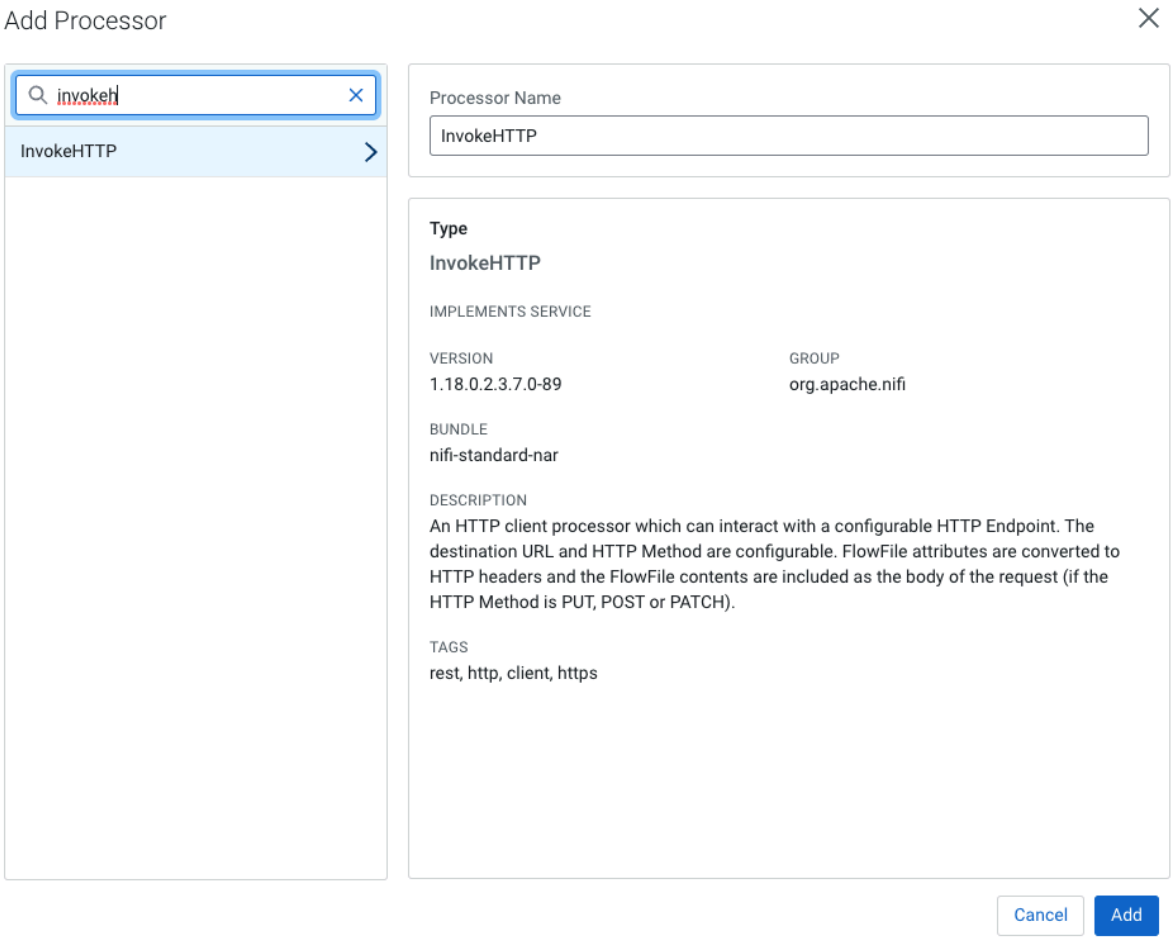

c) Change the Processor Name to Get Recent Wikipedia Changes.
d) Click Add.

After configuration, this processor calls the Wikipedia API to retrieve the latest changes.

2. Configure the Get Recent Wikipedia Changes processor.

   Properties

   HTTP URL

   provide https://en.wikipedia.org/w/api.php?action=query&list=recentchanges&format=json&rcprop=user%7Ccomment%7Cparsedcomment%7Ctimestamp%7Ctitle%7Csizes%7Ctags

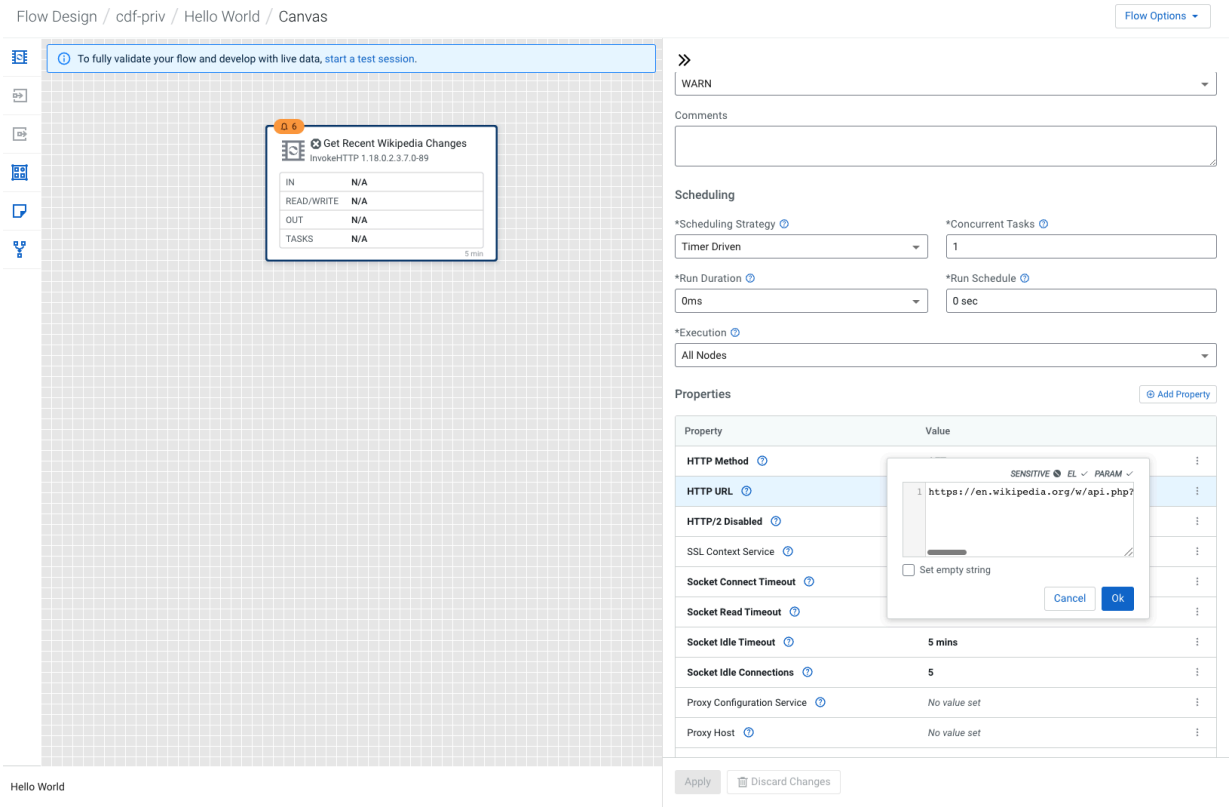

   Relationships

   Select the following relationships:

   - Original - Terminate
   - Failure - Terminate, Retry
   - Retry - Terminate
   - No Retry - Terminate

3. Click Apply.

Notice: Note the orange Notification Pill in the upper left corner of your processor. It is there to warn you about possible issues with a component.

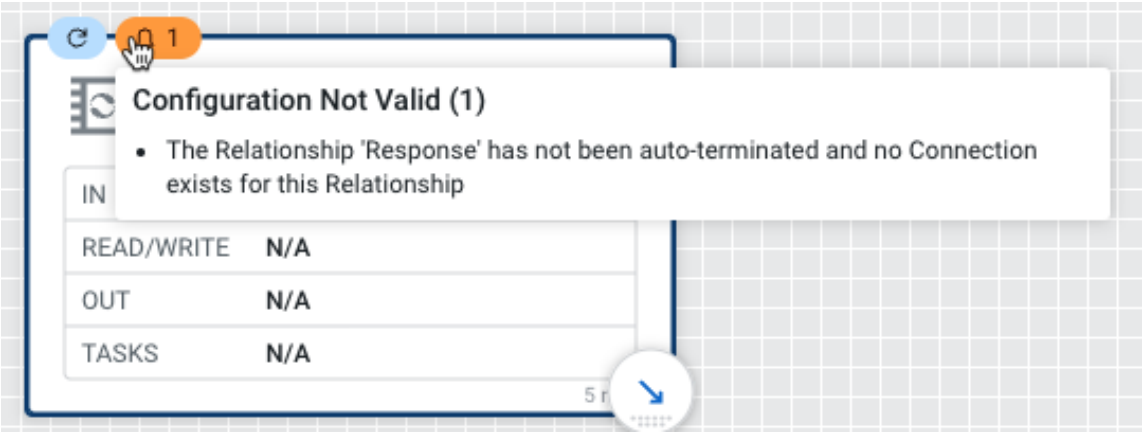

In this particular case, it warns you about one of the Relationships not being addressed when configuring your processor. Do not worry, it will disappear when you create a connection to another processor for the 'Response' relationship.

4. Add a ConvertRecord processor to the canvas.
   a) Drag a Processor from the Components sidebar to the canvas.
   b) In the text box, filter for ConvertRecord.
   c) Change the Processor Name to Convert JSON to AVRO.
   d) Click Add.

   This processor converts the JSON response to AVRO format. It uses RecordReaders and RecordWriters to accomplish this. It infers the JSON schema starting from the recent changes field.
5. Configure the Convert JSON to AVRO processor.

   Properties

   Record Reader

   Select the JSON_Reader_Recent_Changes controller service you have created from the drop-down list.

   Record Writer

   Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

   Relationships

   Select the following relationships:

   failure - Terminate, Retry
6. Click Apply.

7. Connect the Get Recent Wikipedia Changes and Convert JSON to AVRO processors by hovering over the lower-right corner of the Get Recent Wikipedia Changes processor, clicking the arrow that appears and dragging it to the Convert JSON to AVRO processor.

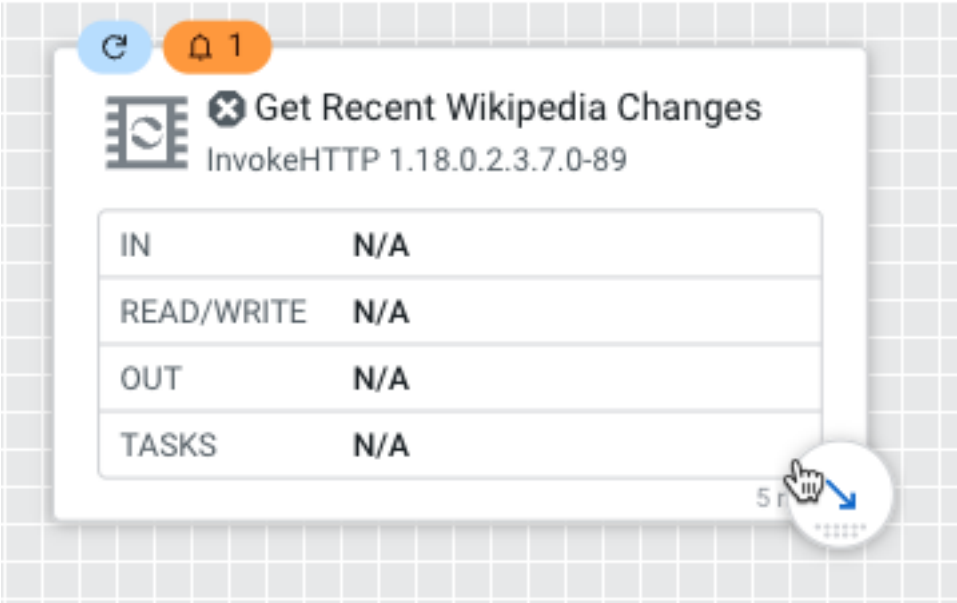

8. In the configuration popup, select the Response relationship and click Add.

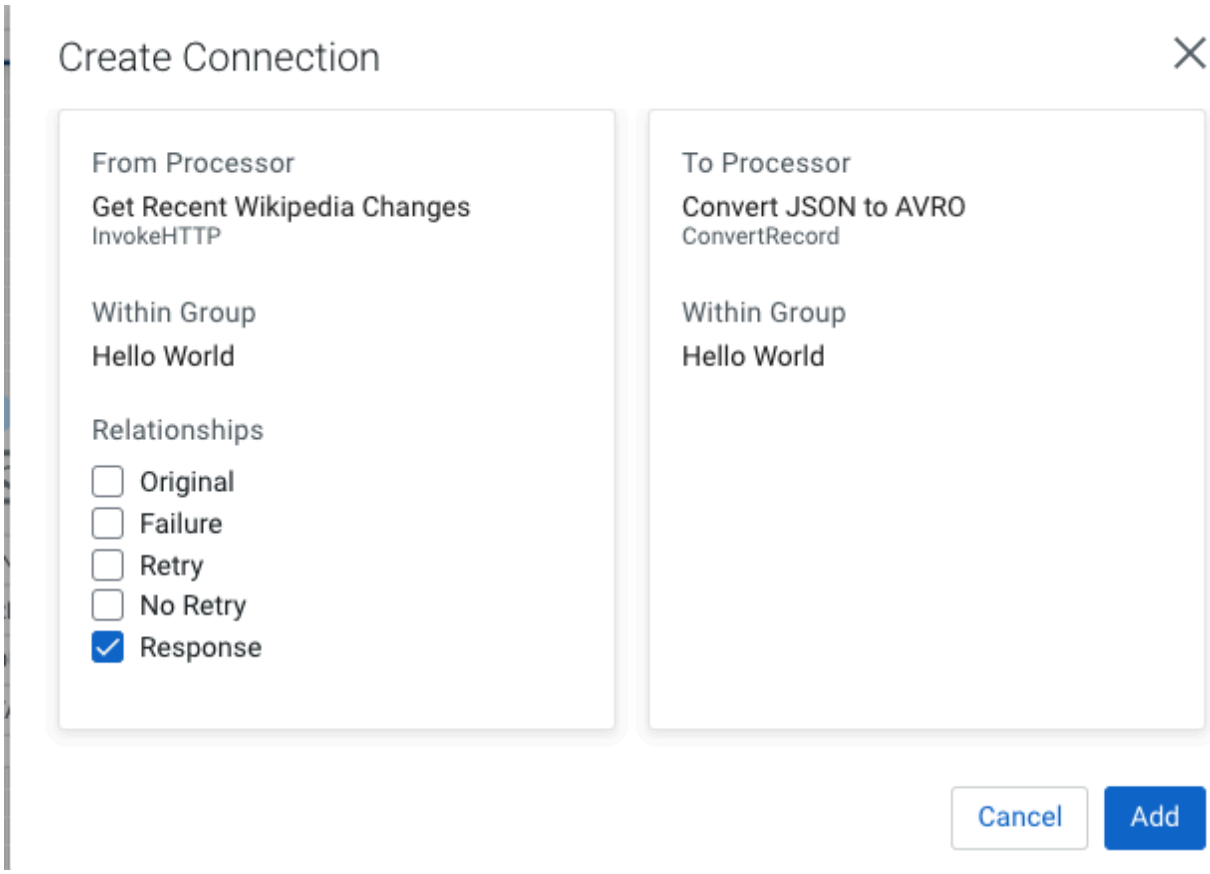

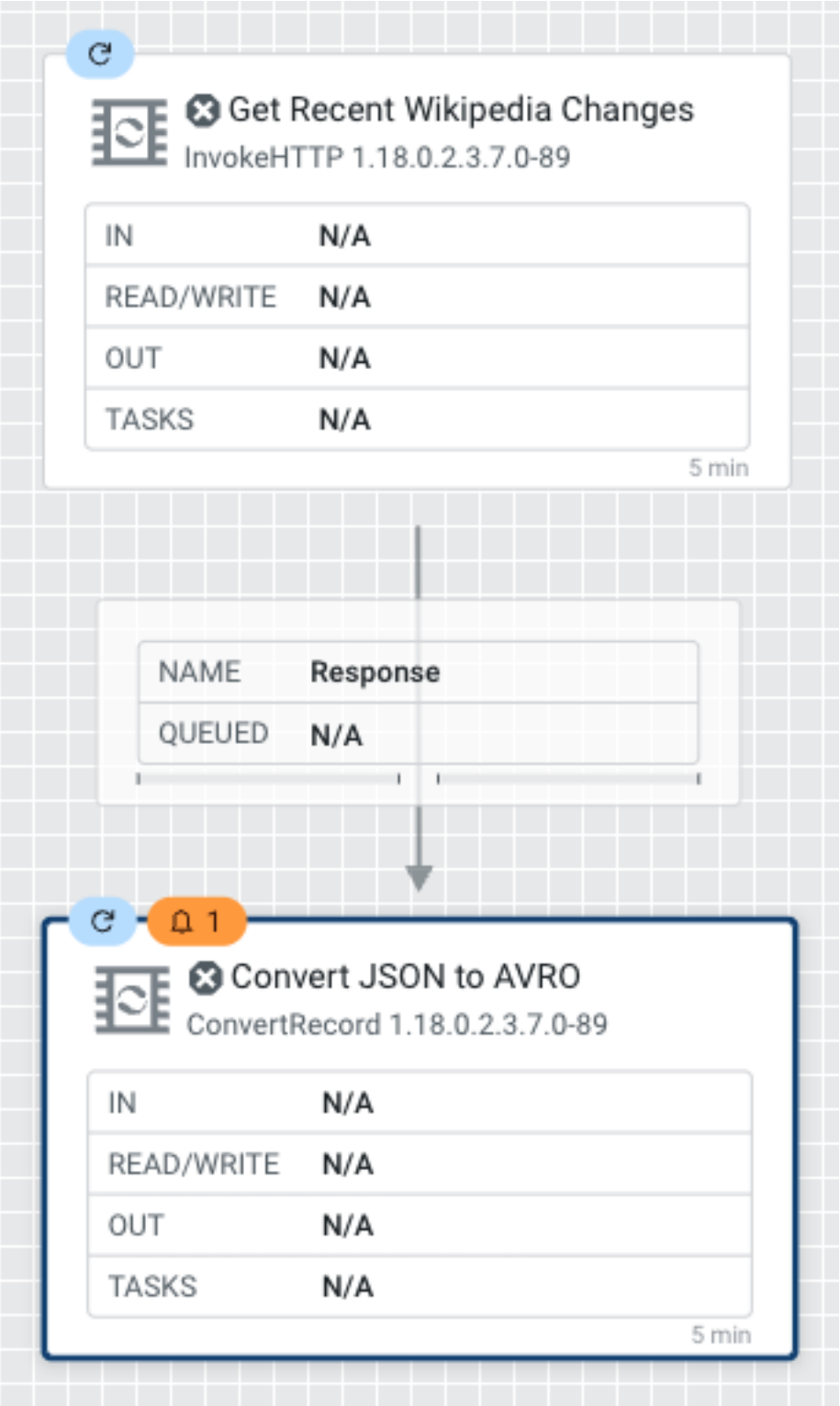

Notice: Note, that the Notification Pill warning about the unconfigured 'Response' relationship disappeared from your Get Recent Wikipedia Changes processor.

9. Add a QueryRecord processor.
   a) Drag a Processor from the Components sidebar to the canvas.
   b) In the text box, filter for QueryRecord.
   c) Name it Filter Edits.
   d) Click Add.

   This processor filters out anything but actual page edits. To achieve this, it's running a query that selects all FlowFiles (events) of the type edit.
10. Configure the Filter Edits processor.

    Properties

    Record Reader

    Select the AvroReader_Recent_Changes controller service you have created from the drop-down list.

    Record Writer

    Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

    Relationships

    Select the following relationship:

    - failure - Terminate
    - original - Terminate
11. For the Filter Edits processor you also need to add a user-defined property. Click Add Property.
    a) Provide Filtered edits as Name
    b) Provide Select * from FLOWFILE where type='edit' as Value.
    c) Click Apply.
12. Connect the Convert JSON to AVRO and Filter Edits processors by hovering over the lower-right corner of the Convert JSON to AVRO processor, clicking the arrow that appears and dragging it to the Filter Edits processor.
13. In the configuration pane, select the success and failure relationships and click Add.
14. Add a second QueryRecord processor.
    a) Drag a Processor from the Components sidebar to the canvas.
    b) In the text box, filter for QueryRecord.
    c) Name it Route on Content Size.
    d) Click Add.

    This processor uses two SQL statements to separate edit events that resulted in a longer article from edit events that resulted in a shorter article.
15. Configure the Route on Content Size processor.

    Properties

    Record Reader

    Select the AvroReader_Recent_Changes controller service you have created from the drop-down list.

    Record Writer

    Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

    Relationships

    Select the following relationships:

    - failure - Terminate, Retry
    - original - Terminate

16. For the Route on Content Size processor you also need to add two user-defined properties.
    a) Click Add Property.
    b) Provide Added content as Name
    c) Provide Select * from FLOWFILE where newlen>=oldlen as Value.
    d) Click Add.
    e) Click Add Property, to create the second property.
    f) Provide Removed content as Name.
    g) Provide Select * from FLOWFILE where newlen<oldlen as Value.
    h) Click Add.
    i) Click Apply.
17. Connect the Filter Edits and Route on Content Size processors by hovering over the lower-right corner of the Filter Edits processor, clicking the arrow that appears and drawing it to Route on Content Size.

    In the Create Connection pop up select the Filtered edits relation and click Add.
18. Add two MergeRecord processors.
    a) Drag a Processor from the Components sidebar to the canvas.
    b) In the text box, filter for MergeRecord.
    c) Name it Merge Edit Events.
    d) Click Add.
    e) Repeat the above steps to add another identical processor.

    These processors are configured to merge at least 100 records into one flowfile to avoid writing lots of small files. The MaxBinAge property is set to 2 minutes which makes the processors merge records after two minutes even if less than 100 records have arrived.
19. Configure the two Merge Edit Events processors.

    Properties

    Record Reader

    Select the AvroReader_Recent_Changes controller service you have created from the drop-down list.

    Record Writer

    Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

    Max Bin Age

    Set to two minutes by providing a value of 2 min.

    Relationships

    Select the following relationships:

    - failure - Terminate
    - original - Terminate

    Note: Do not forget to perform configuration for both Merge Edit Events processors.

20. Connect the Route on Content Size processor to both of the Merge Edit Events processors.

a) For the first Merge Edit Events processor, select Added content from Relationships and click Add.

b) For the second Merge Edit Events processor, select Removed content from Relationships and click Add.

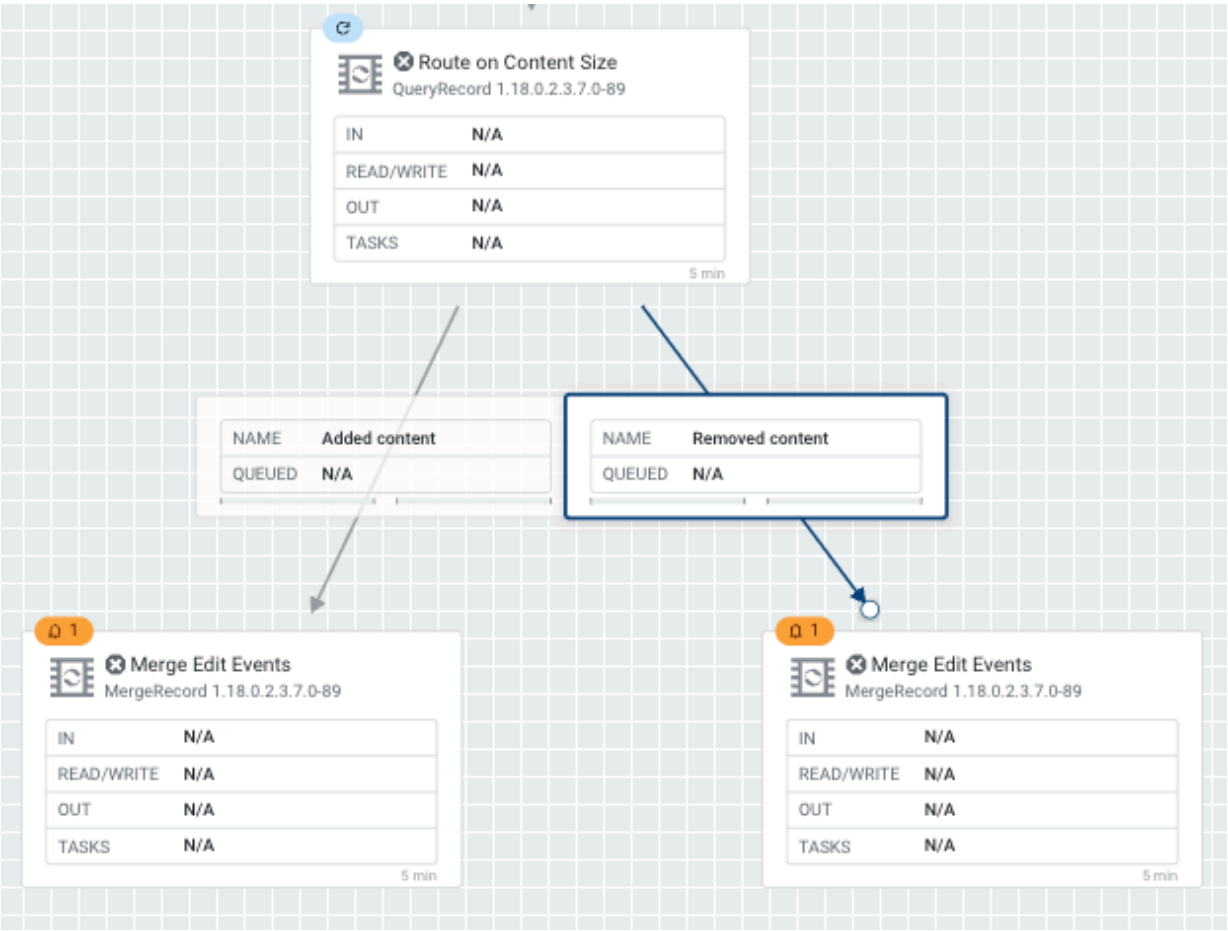

21. Add two PutFile processors to the canvas.

a) Drag a Processor from the Components sidebar to the canvas.

b) In the text box, filter for PutFile.

c) Name it Write "Added Content" Events To File.

d) Click Add.

e) Repeat the above steps to add another identical processor, naming this second PutFile processor Write "Removed Content" Events To File.

These processors write the filtered, routed edit events to two different locations on the local disk. In Cloudera DataFlow, you would typically not write to local disk but replace these processors with processors that resemble your destination system (Kafka, Database, Object Store etc.)

22. Configure the Write "Added Content" Events To File processor.

    Properties:

    Directory

    Provide /tmp/larger_edits

    Maximum File Count

    Provide 500

    Relationships

    Select the following relationships:

    - SUCCESS - Terminate
    - failure - Terminate
23. Click Apply.
24. Configure the Write "Removed Content" Events To File processor.

    Properties:

    Directory

    Provide /tmp/smaller_edits

    Maximum File Count

    Provide 500

    Relationships

    Select the following relationships:

    - SUCCESS - Terminate
    - failure - Terminate
25. Click Apply.
26. Connect the Merge Edit Events processor with the Added content connection to the Write "Added Content" Events To File processor.

    In the Create Connection pop up select merged and click Add.
27. Connect the Merge Edit Events processor with the Removed content connection to the Write "Removed Content" Events To File processor.

    In the Create Connection pop up select merged and click Add.

Results

Congratulations, you have created your first draft flow. Now proceed to testing it by launching a Test Session.

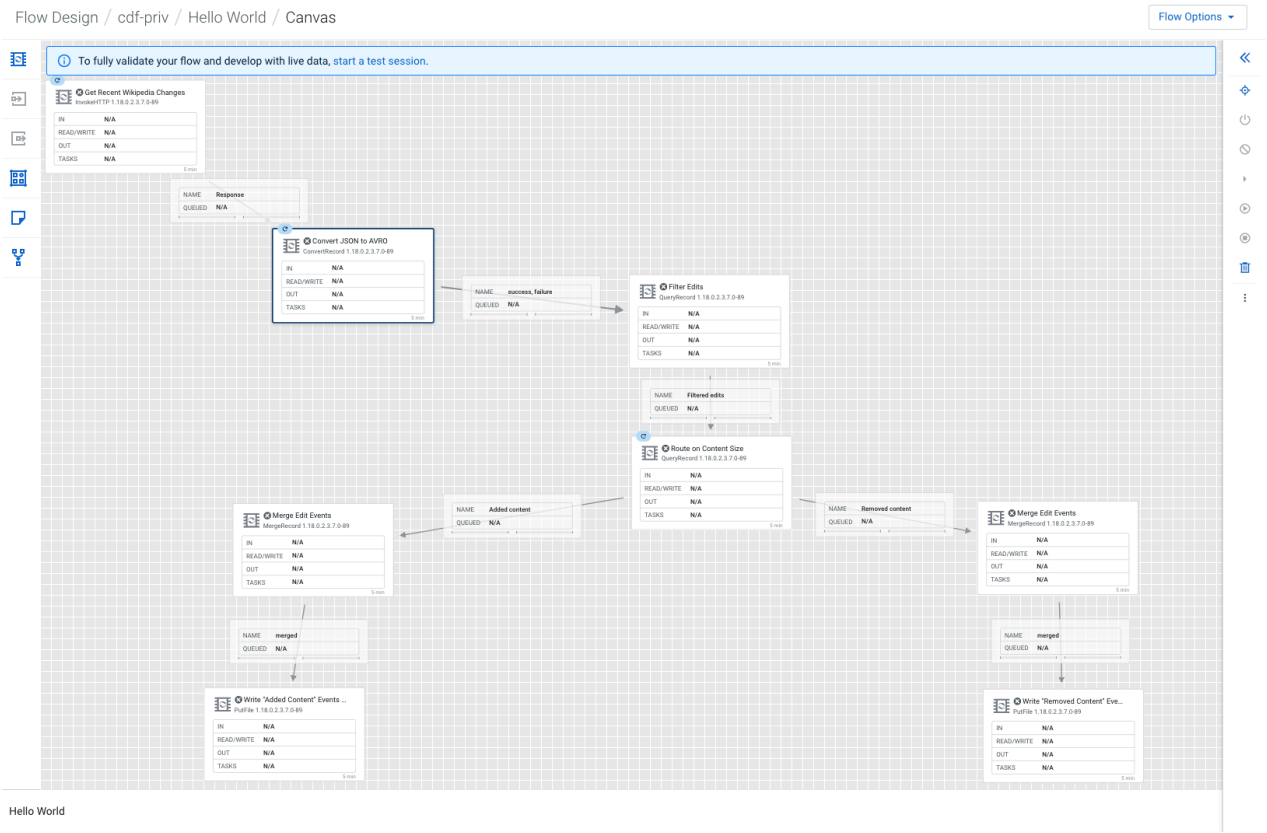

Start a test session

To validate your draft, start a test session. This provisions an Apache NiFi cluster where you can test your draft.

About this task

Starting a Test Session provisions NiFi resources, acting like a development sandbox for a particular draft. It allows you to work with live data to validate your data flow logic while updating your draft. You can suspend a test session any time and change the configuration of the NiFi cluster then resume testing with the updated configuration.

Procedure

1. Click start a test session in the banner on top of the Canvas.

2. Click Start Test Session.

   Test Session status Initializing Test Session... appears on top of the page.

3. Wait for the status to change to

   Active Test Session

   This may take several minutes.

4. Click Flow Options Services to enable Controller Services.

5. Select a service you want to enable, then click Enable Service and Referencing Components.

   This option does not only enable the controller service, but also any component that references it. This way, you do not need to enable the component separately to run the test session. In the context of this tutorial, enabling the 'AvroReader_Recent_Changes' controller service will also enable 'Filter Edits', 'Route on Content Size', and 'Merge Edit Events' processors as well.

   Repeat this step for all Controller Services.
6. Click Back To Flow Designer to return to the flow design Canvas.
7. Start the Get Recent Wikipedia Changes, Write "Added Content" Events To File, and Write "Removed Content" Events To File components by selecting them on the Canvas then clicking Start.

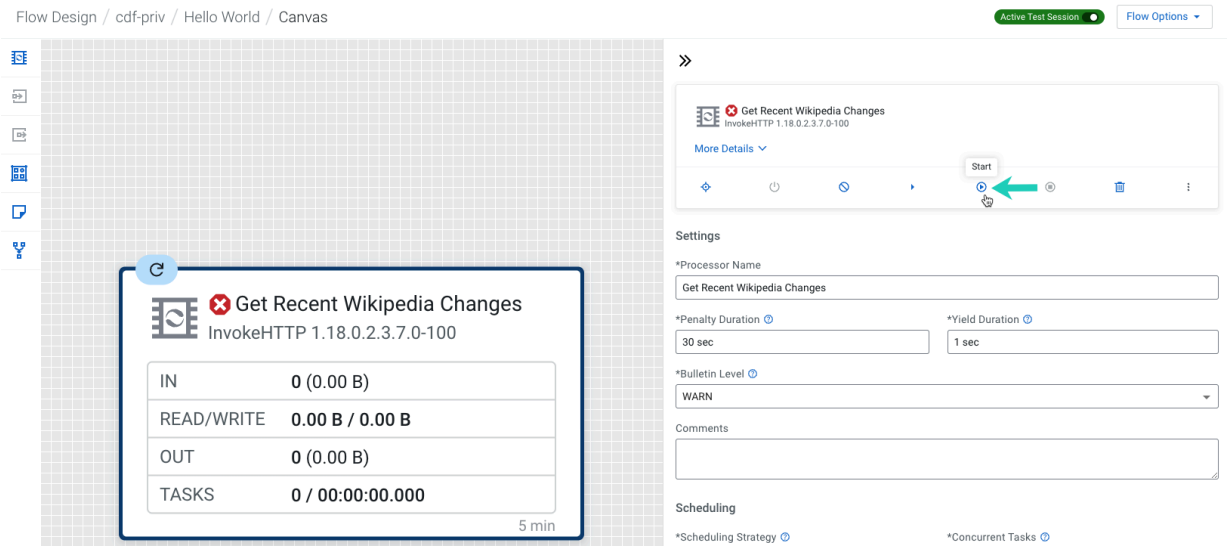

All other components were auto-started when you selected the Enable Service and Referencing Components option.

8. Observe your first draft flow processing data.

   On the Flow Design Canvas you can observe statistics on your processors change as they consume and process data from Wikipedia. You can also observe one or more blue Notification Pills, providing information about the current task.

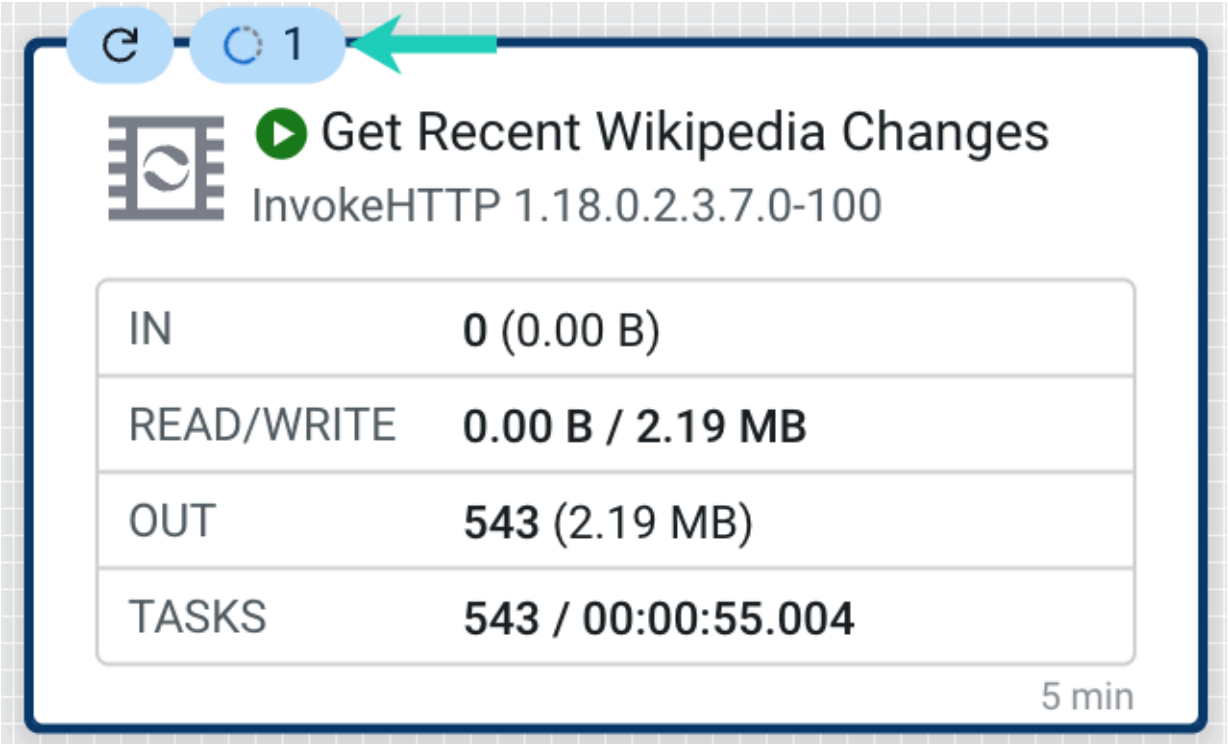

Publish your flow definition to the Catalog

Now that you have tested your draft and it works fine, you can go on and publish it to the Catalog as a flow definition so that you can create a Cloudera DataFlow deployment.

Procedure

1. On the **Flow Designer** canvas, click Flow Options Publish To Catalog Publish .
2. Fill in the fields in the **Publish A New Flow** box.
   - Provide a Flow Name for your flow definition.

     You can only provide a name when you publish your flow for the first time.
   - Optionally provide a Flow Description.

     You can only provide a description when you publish your flow for the first time.
   - Optionally provide Custom Tags.

     You can filter flow definition versions by tags in the Catalog.
   - Optionally provide Version Comments.
3. Click Publish.

Results

Your draft is published to the Catalog as a flow definition.

Related Information

Deploying a flow definition

Cloudera DataFlow

Tutorial: Customize a ReadyFlow

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Tutorial: customizing a ReadyFlow.....4
Open the Hello World ReadyFlow as a template for your draft.....4
Create new services.....5
Swap the controller services in your draft.....7
Start a test session.....9
Publish your flow definition to the Catalog.....11

Tutorial: customizing a ReadyFlow

Learn how to create a draft using a ReadyFlow as a template.

About this task

This tutorial shows you how you can open a ReadyFlow from the ReadyFlow Gallery and create a customized flow design using the ReadyFlow as a template. This tutorial uses the 'Hello World' ReadyFlow, a simple flow design that retrieves the latest changes from Wikipedia through invoking the Wikipedia API. It converts JSON events to Avro, before filtering and routing them to two different processors which merge events together before a file is written to local disk.

You will learn about:

- Opening a ReadyFlow as a draft on the Flow Design Canvas.
- Creating a Controller Service
- Changing the configuration of processors
- Running a Test Session
- Publishing a draft to the Catalog as a flow definition.

Before you begin

The 'Hello World' ReadyFlow that you are about to customize can be deployed without any external dependencies and does not require any parameter values during deployment. Still, there are prerequisites you have to meet before you can start building your first draft.

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFDeveloper role granting you access to the Flow Designer.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog. You will need this authorization to publish your draft as a flow definition to the Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.

Open the Hello World ReadyFlow as a template for your draft

About this task

ReadyFlows are read-only, therefore you cannot publish a new version of them. When you are done with creating your customized version of the given ReadyFlow, you can only publish your work to the Catalog as a new flow definition.

Before you begin

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFDeveloper role granting you access to the Flow Designer.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog. You will need this authorization to publish your draft flow as a flow definition to the Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.

2. Select ReadyFlow Gallery in the left navigation pane.
3. Select the Hello World ReadyFlow.
4. Click Create New Draft.
5. Select the target Workspace where you want to create the draft flow.
6. Provide a valid Flow Name for the draft flow.

   Flow names must be unique within a workspace. If a draft with the provided name already exists, you need to provide a different name.
7. Click Create.
   Hello World opens as a draft in the designated Flow Designer workspace with the Flow Name you provided on the **Flow Design** canvas.

What to do next

Proceed with creating the necessary Controller Services.

Related Tasks

Create new services

Create new services

Learn about creating Controller Services in Cloudera DataFlow Flow Designer.

About this task

Services are extension points that provide information for use by other components (such as processors or other controller services). The idea is that, rather than configure this information in every processor that might need it, the service provides it for any processor to use as needed.

You will use the services you create now to configure the behavior of several processors you will add to your flow as you are building it.

Procedure

1. Go to Flow Options Services .

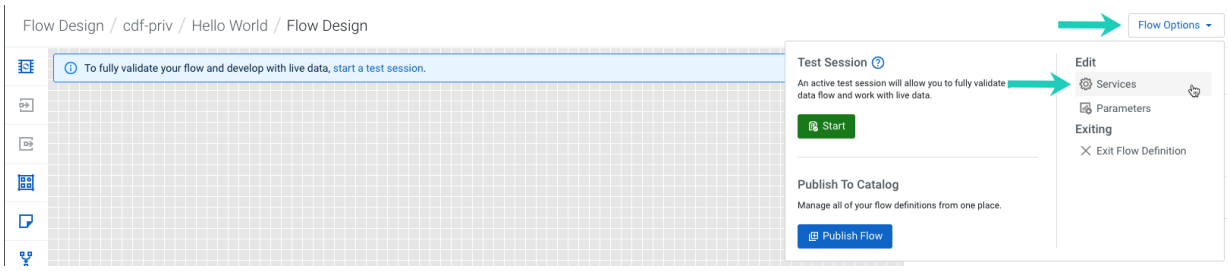

2. Click Add Service.
   The Add Service page opens.

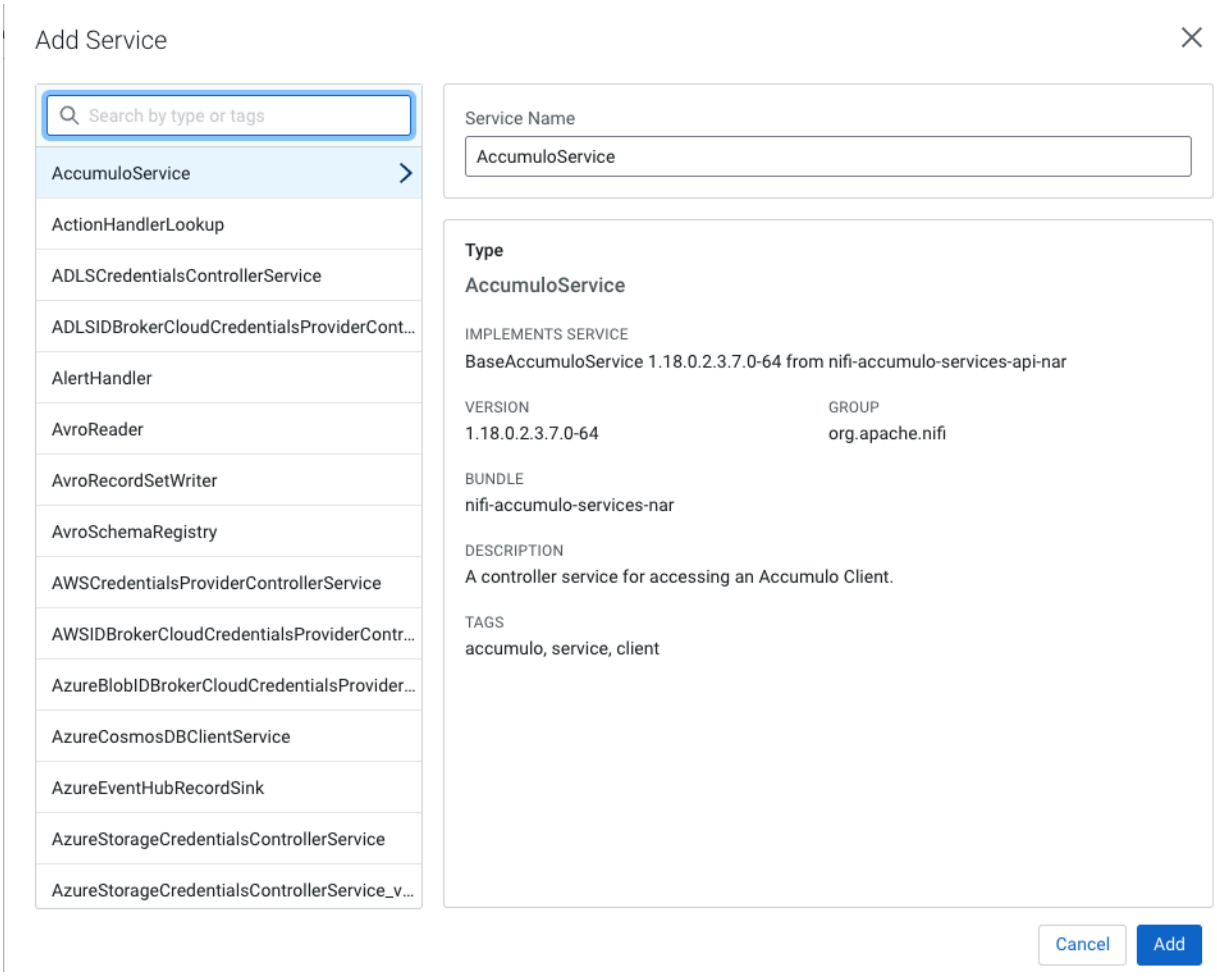

3. In the text box, filter for CSVReader.
4. Provide Service Name: CSVReader_Recent_Changes.
5. Click Add.

   You do not need to configure the CSVReader_Recent_Changes service. You can leave all properties with their default values.
6. Click Add Service to create another service.
7. In the text box, filter for CSVRecordSetWriter.
8. Provide Service Name: CSVRecordSetWriter_Recent_Changes.
9. Click Add.

   You do not need to configure the CSVRecordSetWriter_Recent_Changes service. You can leave all properties with their default values.
10. Select the AvroReader_Recent_Changes service and check the list of Referencing Components in the Service Details pane on the right.

    Make a note of the listed components, because that is where you will need to replace AvroReader_Recent_Changes with the CSVReader_Recent_Changes service.

11. Select the AvroWriter_Recent_Changes service and check the list of Referencing Components in the Service Details pane on the right.

Make a note of the listed components, because that is where you will need to replace AvroWriter_Recent_Changes with the CSVRecordSetWriter_Recent_Changes service.

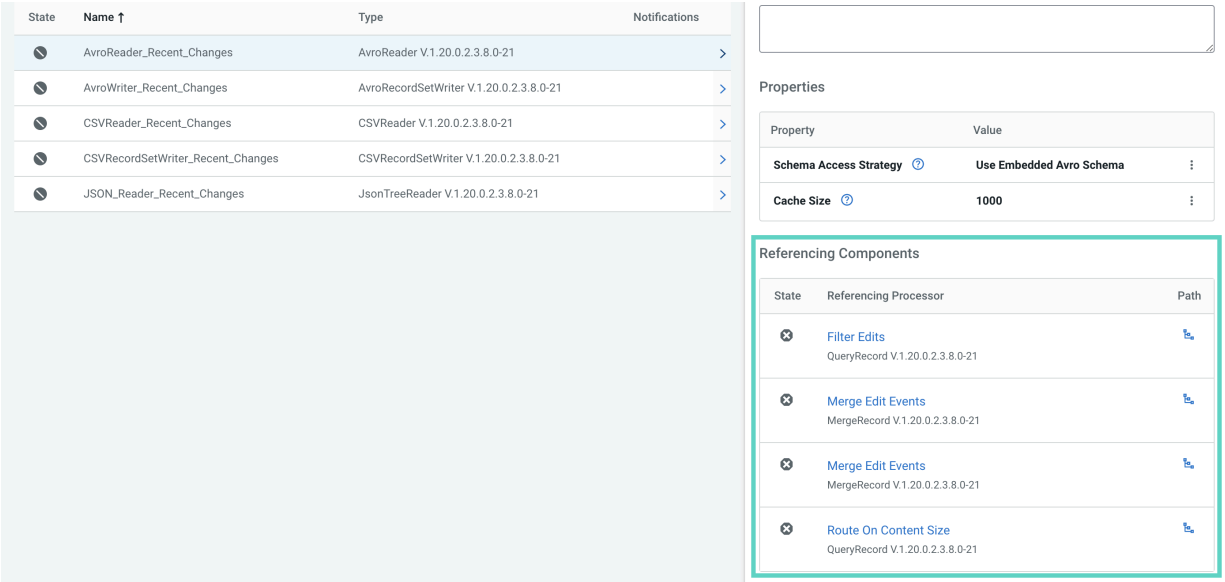

12. Click Back To Flow Designer to return to the flow design Canvas.

What to do next

After creating the necessary services, you can start customizing your flow.

Related Tasks

Swap the controller services in your draft

Swap the controller services in your draft

Customize your draft by replacing services to change the destination file format.

Procedure

1. Select the Convert JSON TO AVRO processor by clicking on it.

2. In the Configuration pane scroll down to Properties, click the drop-down next to Record Writer and select CSVRecordSetWriter_Recent_Changes.

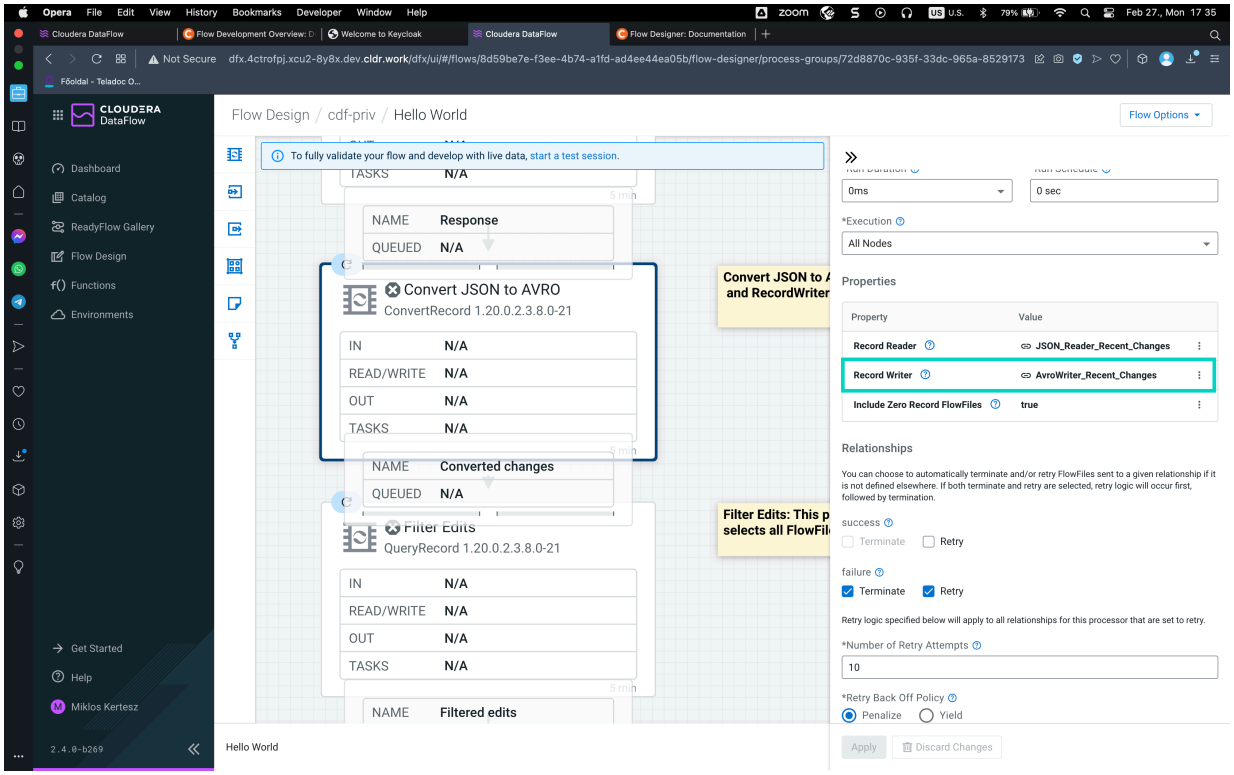

3. Click OK.
4. Rename the processor from Convert JSON to AVRO to Convert JSON to CSV.
5. Click Apply.
6. Select the Filter Edits processor by clicking on it.
7. In the Configuration pane scroll down to Properties, click the drop-down next to Record Reader and select CSVReader_Recent_Changes.
8. Click OK.
9. Click the drop-down next to Record Writer and select the CSVRecordSetWriter_Recent_Changes service.
10. Click OK.
11. Click Apply.
12. Proceed with updating the remaining Record Reader and Record Writer references.

    You need to modify the configurations of the Route on Content Size and the two Merge Edit Events processors.

13. Check that you have updated all the processors.
    a) Select Flow Options Services .
    b) Select the CSVReader_Recent_Changes service and check Referencing Components.

       You should see:

       - Filer Edits
       - Merge Edit Events
       - Merge Edit Events
       - Route On Content Size
    c) Select the CSVRecordSetWriter_Recent_Changes service and check Referencing Components.

       You should see:

       - Convert JSON to AVRO
       - Filer Edits
       - Merge Edit Events
       - Merge Edit Events
       - Route On Content Size
    d) If you check the AvroReader_Recent_Changes and AvroWriter_Recent_Changes services, you should see No referencing Processors to display.
14. Delete the AvroReader_Recent_Changes and AvroWriter_Recent_Changes services by selecting the service and clicking Delete.

    In the confirmation pop-up select Delete.
15. Click Back To Flow Designer to return to the flow design Canvas.

What to do next

Start a Test Session to validate your draft.

Start a test session

To validate your draft, start a test session. This provisions an Apache NiFi cluster where you can test your draft.

About this task

Starting a Test Session provisions NiFi resources, acting like a development sandbox for a particular draft. It allows you to work with live data to validate your data flow logic while updating your draft. You can suspend a test session any time and change the configuration of the NiFi cluster then resume testing with the updated configuration.

Procedure

1. Click start a test session in the banner on top of the Canvas.

2. Click Start Test Session.

   Test Session status

   Initializing Test Session... appears on top of the page.
3. Wait for the status to change to

   Active Test Session

   This may take several minutes.
4. Click Flow Options Services to enable Controller Services.

5. Select a service you want to enable, then click Enable Service and Referencing Components.

   This option does not only enable the controller service, but also any component that references it. This way, you do not need to enable the component separately to run the test session. In the context of this tutorial, enabling the 'AvroReader_Recent_Changes' controller service will also enable 'Filter Edits', 'Route on Content Size', and 'Merge Edit Events' processors as well.

   Repeat this step for all Controller Services.
6. Click Back To Flow Designer to return to the flow design Canvas.
7. Start the Get Recent Wikipedia Changes, Write "Added Content" Events To File, and Write "Removed Content" Events To File components by selecting them on the Canvas then clicking Start.

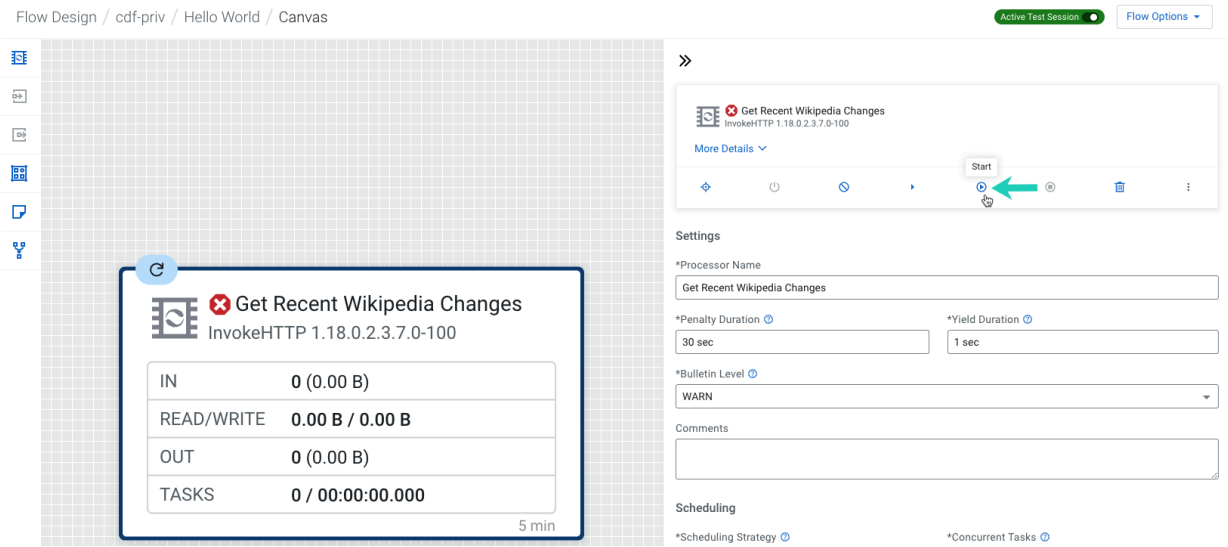

All other components were auto-started when you selected the Enable Service and Referencing Components option.

8. Observe your first draft flow processing data.

   On the Flow Design Canvas you can observe statistics on your processors change as they consume and process data from Wikipedia. You can also observe one or more blue Notification Pills, providing information about the current task.

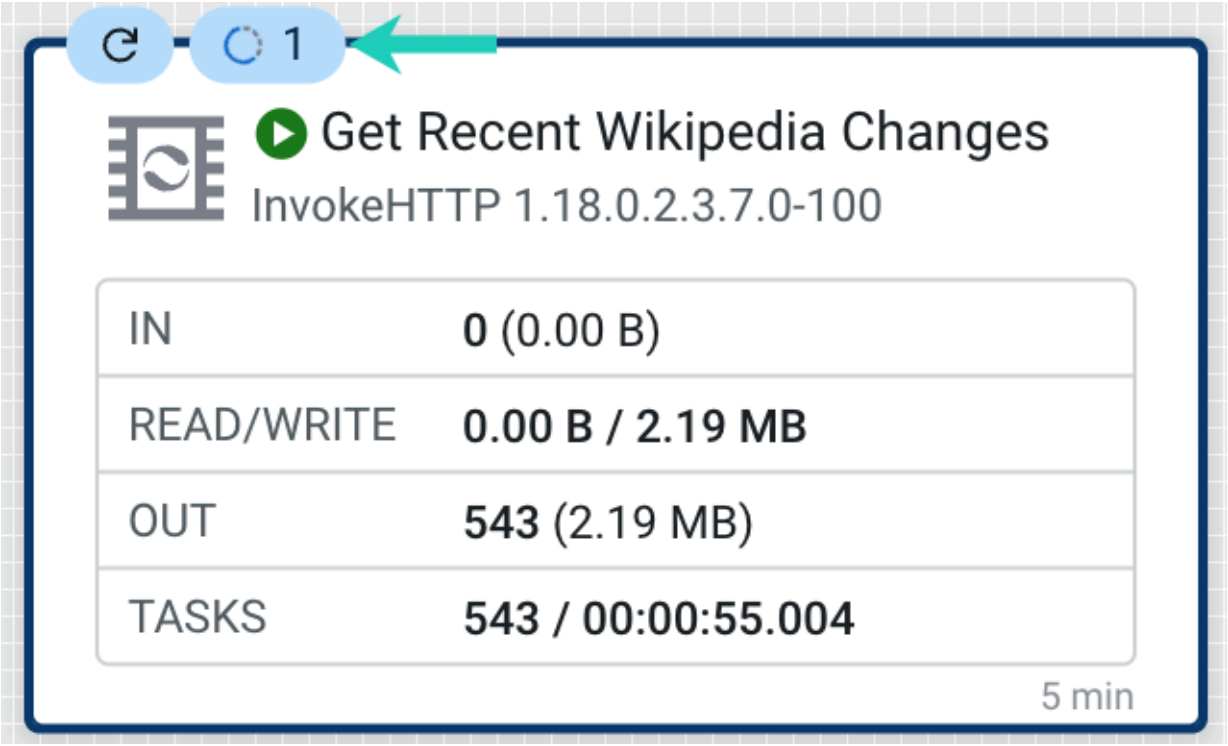

Publish your flow definition to the Catalog

Now that you have tested your draft and it works fine, you can go on and publish it to the Catalog as a flow definition so that you can create a Cloudera DataFlow deployment.

Procedure

1. On the **Flow Designer** canvas, click Flow Options Publish To Catalog Publish .
2. Fill in the fields in the **Publish A New Flow** box.
   - Provide a Flow Name for your flow definition.

     You can only provide a name when you publish your flow for the first time.
   - Optionally provide a Flow Description.

     You can only provide a description when you publish your flow for the first time.
   - Optionally provide Custom Tags.

     You can filter flow definition versions by tags in the Catalog.
   - Optionally provide Version Comments.
3. Click Publish.

Results

Your draft is published to the Catalog as a flow definition.

Related Information

Deploying a flow definition

Cloudera DataFlow

Creating Drafts

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a new draft **4**

Creating a draft from a flow definition in the Catalog **5**
Matching parameter contexts to shared parameter groups during draft creation 6

Creating a draft from a ReadyFlow **10**

Parameters **11**
Adding a parameter to your draft 13
Adding a parameter from Flow Options 14
Creating a parameter from the Configuration pane 16
Importing a parameter group 17
Exporting draft parameters to a parameter group 18

Configuring advanced usage of the UpdateAttribute processor **20**
Add a Rule 22
Reorder Rules 22

Testing JoltTrasformJSON processor configuration **22**

Adding a service to your draft **24**
Adding a service from Flow Options 24
Creating a service from Configuration 24

Adding inbound connection support to a draft **26**

Copying and pasting processors **27**

Creating a new draft

Learn about creating a draft in Cloudera DataFlow Flow Designer.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Open Cloudera DataFlowby selecting the DataFlow tile in the Cloudera sidebar.
2. Select Flow Design in the left navigation pane.

   You are redirected to the Flow Design page, where previously created drafts are displayed, one per row.
3. Select Create Draft.
4. Select the Target Workspace where you want to create the draft.
5. Select the Target Project to which you want to assign the draft.
6. Provide a valid Draft Name.

   Draft names must be unique within a workspace. If a draft with the provided name already exists, you need to provide a different name.
7. Select NiFi Major Version.

   Depending on the NiFi version you select, the list of available processors will be different. Currently you cannot change the the NiFi major version of an existing draft.

   Note: NiFi 2.x is provided as a technical preview in this release of Cloudera DataFlow.
8. Click Create.
   Flow Designer creates a draft and a default Process Group with the Draft Name you provided and you are redirected to the **Flow Design** canvas. The **Configuration** pane on the right displays configuration options for the default Process Group.
9. Start building your draft by dragging components to the canvas from the Components sidebar on the left.

   The **Configuration** sidebar always displays configuration options for the highlighted component. To select a component for configuration, click on it.

   Tip:
   - To create a Controller Service, click Flow Options Services Add Service .
   - To create a Parameter, click Flow Options Parameters Add Parameter .

     Unlike in NiFi, there is one default parameter group auto-created for each draft, where you can add parameters. This parameter group is then automatically bound to each Process Group you create within your draft.

10. You can connect two processors by hovering the cursor over the lower-right corner of the source processor, clicking the arrow that appears and dragging it to the target processor.

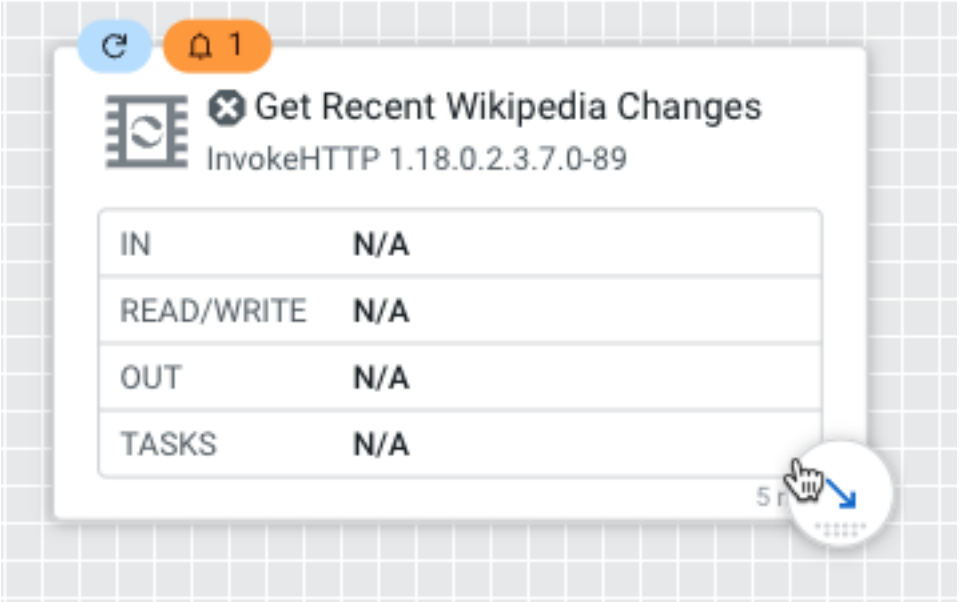

11. To test your draft, start a Test Session by clicking Flow Options Test Session Start Start Test Session .

    This provisions a NiFi sandbox and allows you to validate your draft and work with live data by starting and stopping components.

    Tip: You can check the status of your Test Session in the upper right corner of your workspace. That is where you can also deactivate the Test Session.

12. Once you are confident that your flow design is ready for production, click Flow Options Publish To Catalog Publish .

    This step turns your draft to a flow definition and adds it to the Catalog. From there, you can deploy your flow to any environment where you have DFFlowAdmin authorization.

Related Tasks

Adding a parameter to your draft

Adding a service to your draft

Related Information

Cloudera DataFlow authorization

Starting a test session

Publishing to the Catalog

Creating a draft from a flow definition in the Catalog

You can open flow definitions in the Catalog as drafts in Flow Designer. Depending on whether the flow definition has associated flow drafts or not, you may have more than one option.

Before you begin

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFDeveloper role granting you access to the Flow Designer.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog. You will need this authorization to view and open flow definitions, and to publish your draft as a flow definition to the Catalog.

- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.

About this task

If you have sufficient rights, you can create a new draft from existing flow definitions in the Catalog. This gives you two options:

- You can update the existing flow definition by creating a draft and then publishing your work as a new version of the source flow definition.
- You can create a draft using the source flow definition as a template and then publish your work as a new flow definition to the Catalog.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Select Catalog in the left navigation pane.
3. Select the flow that you want to open as a draft.
4. Depending on whether there already is one or more associated draft for the selected flow, you may have two options to select from:
   - If there is one or more ASSOCIATED DRAFT, you can select that and the draft opens for editing in the Flow Design view.
   - If there are no associated drafts or you want to create a new flow based on the existing one, select Create New Draft.

     Note: If there is more than one version available, select the version that you want to use as a template for your draft.
5. Click Create.

   If the source flow definition contains more than one parameter context, the **Map Parameter Contexts** page opens. If parameter matching is not necessary, the draft opens in the Flow Designer.
6. If the source flow definition contains more than one parameter context, the **Map Parameter Contexts** page opens. This is the step where you can review the suggested shared parameter groups to replace parameter contexts in your flow definitions.

   For more information on parameter matching, see Parameter matching .
7. Click Create Flow Draft.

What to do next

Proceed according to *Creating a draft*.

Related Concepts

Parameters

Related Tasks

Exporting draft parameters to a parameter group

Matching parameter contexts to shared parameter groups during draft creation

If the source flow definition contains more than one parameter context attached to the root process group, the Flow Designer tries to match those contexts to shared parameter groups that are assigned to the project where the draft is created.

Depending on the structure of the source flow definition, parameter matching may or may not be a separate step during draft creation:

- If the flow definition has a single parameter context attached to its root process group, the draft is created without the parameter matching step.
- If the source flow definition has parameter contexts attached to different process groups, the flow definition cannot be converted into a flow draft within Flow Designer, subsequently parameter matching does not happen.
- If the source flow definition has several parameter contexts atttached to its root process group, parameter matching

The figure *Parameter matching scenarios* illustrates these cases.

Figure 1: Parameter matching scenarios

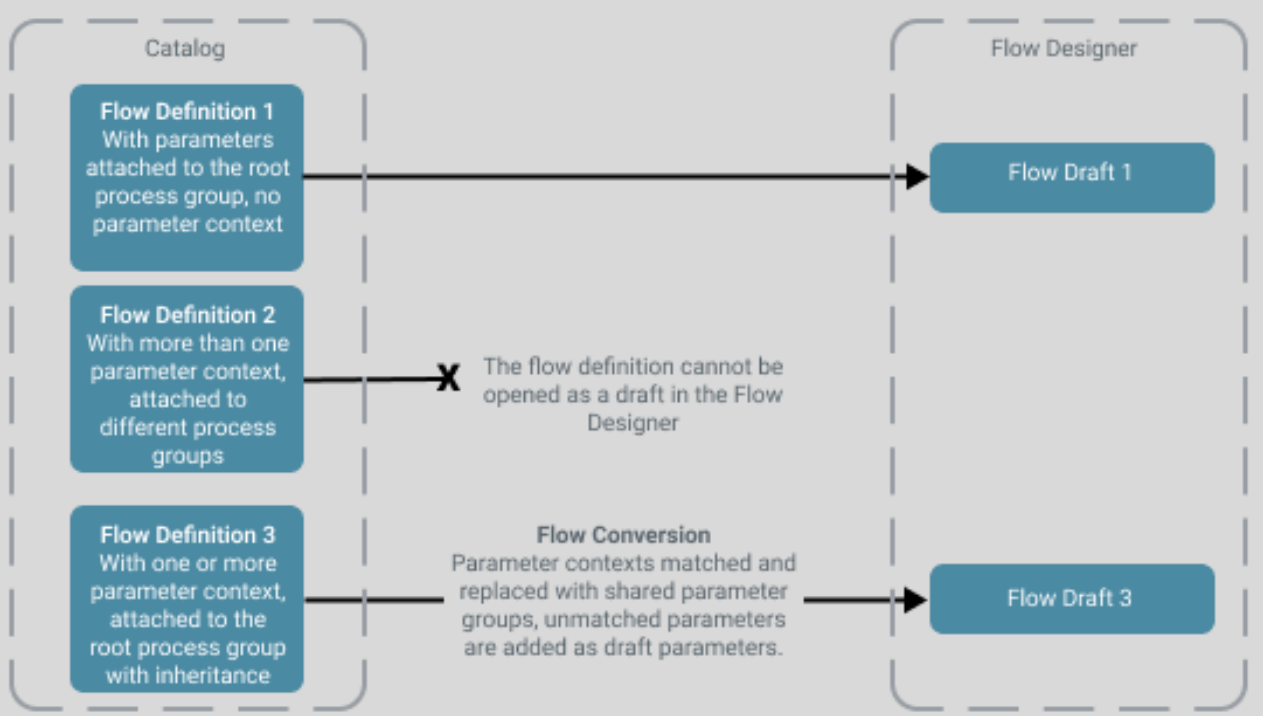

Parameter matching happens by matching parameter context names to shared parameter group names, and you are provided with a table of the results on the **Map Parameter Contexts** page.

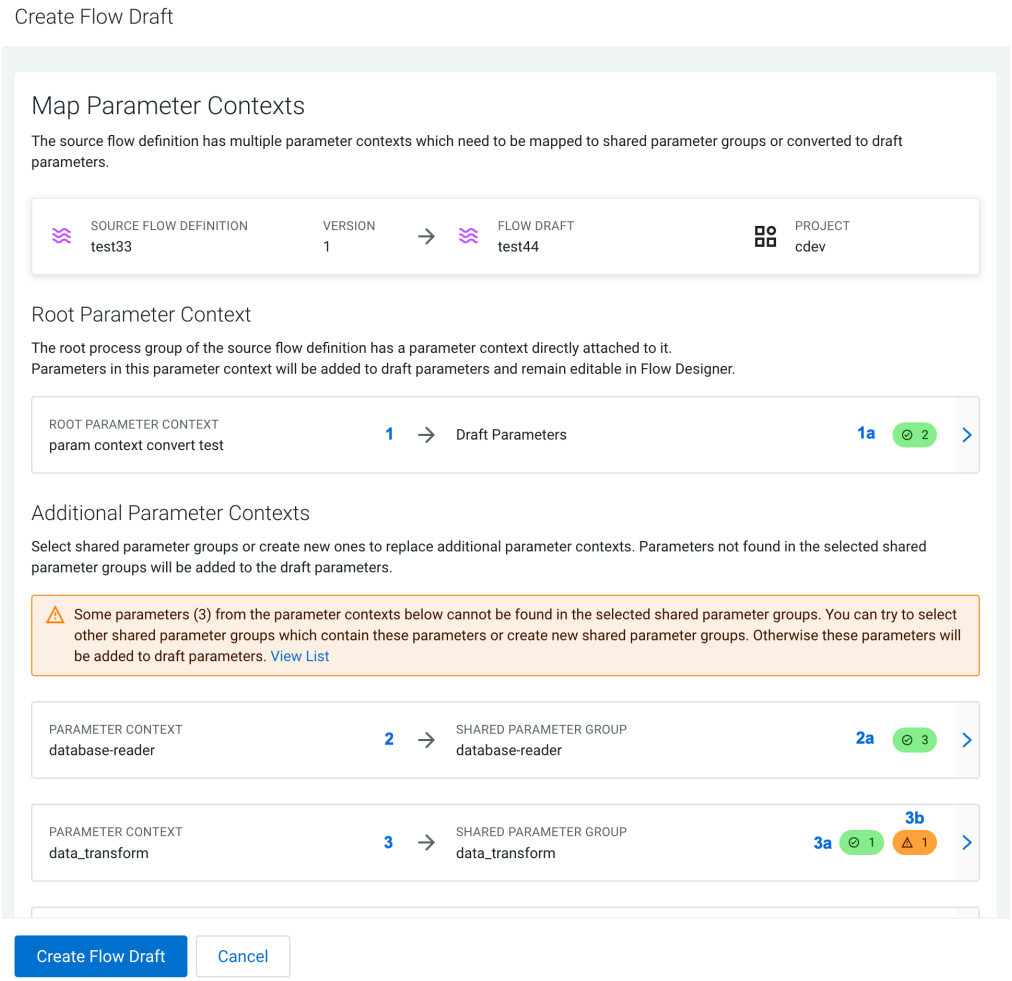

In this step, you can examine the mapping of parameter contexts to parameter groups and make modifications to this mapping as necessary.

1. Parameters attached to the source flow definition's root process group are added to the root process group of the flow draft as draft parameters and remain editable within Flow Designer.

   a. The green pill with a number, 3 in this example, displayed on the right shows the number of parameters attached to the root process group of the source flow definition added to the root process group of the flow draft as draft parameters.

   You can click on the > icon to view the draft parameters added to te flow draft. Draft parameters can be edited within the Flow Designer.

2. There is a one-to-one match between the source parameter context and the suggested shared parameter group. This means that Flow Designer found a shared parameter group where both the name and the contents match with the given rparameter context.

   a. The green pill with a number, 3 in this example, displayed on the right shows the number of parameters in the parameter context replaced by parameters in the shared parameter group.

   Note: Any parameters in the shared parameter group that have no matching pairs in the parameter context are still kept in the draft but, as there are no components referencing them, they will play no role whatsoever.

   Click on the > icon to view the parameters within the shared group. Parameters in a shared group cannot be modified in the Flow Designer, only through the **Workspace Resources** page.

3. There is a partial match. A partial match means that there is a shared parameter group with a name that matches the name of the given parameter context, but their contents are not identical.
   a. The green pill marks the number of matching parameters. These parameters are part of the suggested shared parameter group and can only be modified through the **Workspace Resources** page.
   b. The orange pill marks the number of parameters in the parameter context that were not matched in the shared parameter group. These parameters are added to the root process group of the flow draft as draft parameters.

   Note: Any parameters in the shared parameter group that have no matching pairs in the parameter context are still kept in the draft but, as there are no components referencing them, they will play no role whatsoever.

   Click on the > icon to view the parameter matching details. Parameters in a shared group cannot be modified in this view, only through the **Workspace Resources** page. You can select a different Shared parameter group from the drop-down, e.g.: if you know there is a shared group under a different name, with the required parameters.

   Note: Any parameters in the shared parameter group that have no matching pairs in the parameter context are still kept in the draft but, as there are no components referencing them, they will play no role whatsoever.

Modifying the results of parameter matching

Click on the > icon to view the parameter matching details.

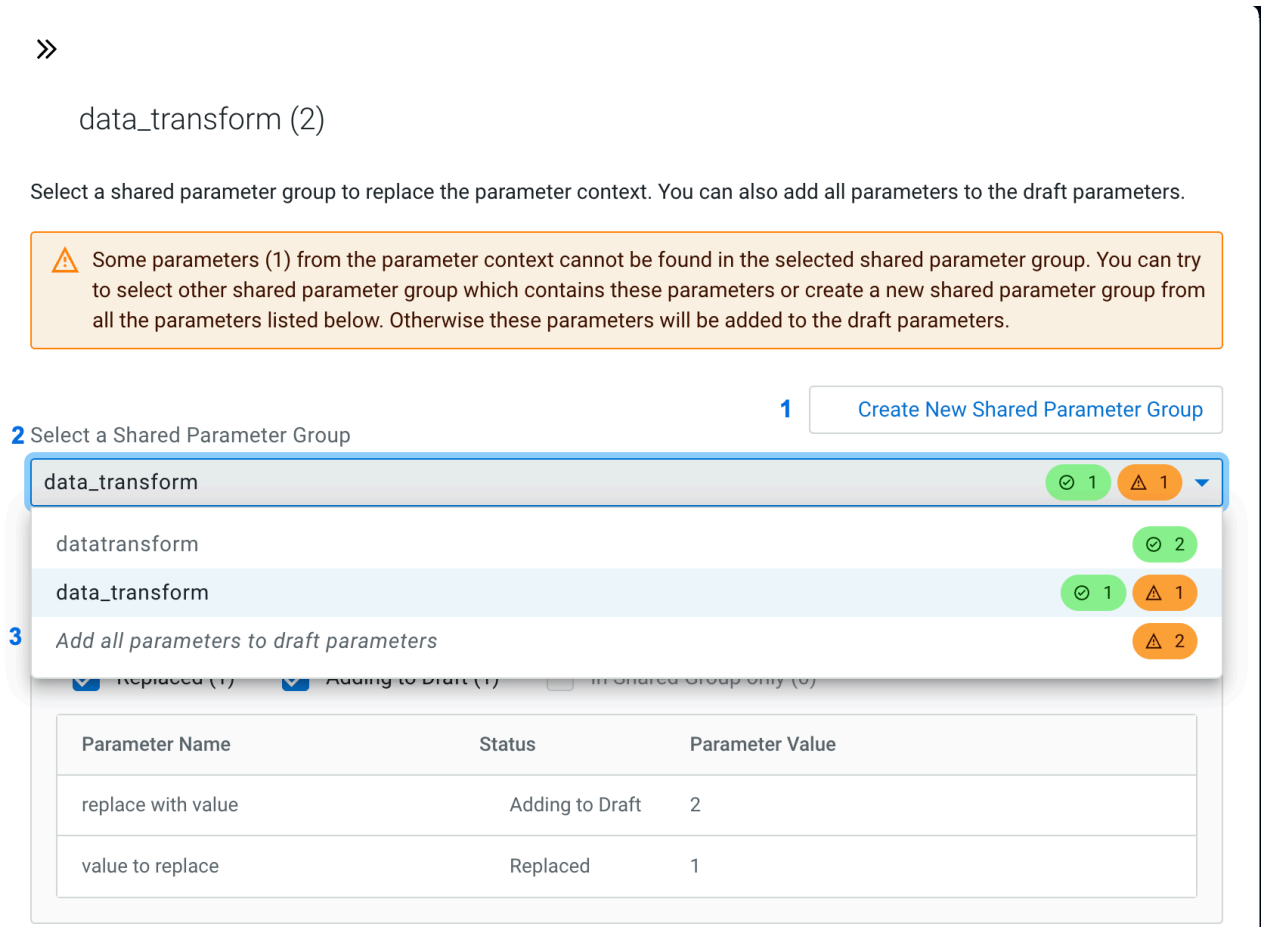

In this view you can:

1. You can Create New Shared Parameter Group that includes all the parameters in the parameter context of the source flow definition.

2. Select a Shared Parameter Group from the drop-down, for example, if you know there is a shared group under a different name, which actually matches the parameter context in the source flow definition better. In the example on the figure, the shared paramete group data_transform was selected by Flow Designer based on name matching however, datatransform is a better match, as it contains all two parameters present in the source parameter context.
3. Add all parameters to the root process group of the flow draft as draft parameters by selecting Select a Shared Parameter Group Add all parameters to draft parameters .

Outcome of parameter matching

Precedence

- Draft parameters always take precedence over shared parameters.
- In case of shared parameter groups, their order defines precedence. By changing the order of parameter groups, you can redefine precedence. As a result of this, if a parameter is present in several shared parameter groups, the value of the one in the topmost group takes precedence.

Value retention

- Values of both non-sensitive and sensitive parameters that were matched with a shared parameter group take the values from the shared parameter group.
- Non-sensitive parameters that were added to draft parameters retain the values they had in the source flow definition.
- Sensitive parameters that were added to draft parameters lose their original values.

Creating a draft from a ReadyFlow

You can use ReadyFlows as the starting point of your flow design. Select an available ReadyFlow from the ReadyFlow Gallery, open it for editing on the Canvas, and publish it to the Catalog as your own custom flow definition.

Before you begin

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFDeveloper role granting you access to the Flow Designer.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog. You will need this authorization to publish your draft as a flow definition to the Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.

About this task

ReadyFlows are read-only, therefore you cannot add a new version to them. When you are done with creating your customized version of the given ReadyFlow, you can only publish your work to the Catalog as a new flow definition.

Procedure

1. Open Cloudera DataFlow by selecting the DataFlow tile in the Cloudera Public Cloud sidebar.
2. Select ReadyFlow Gallery in the left navigation pane.
3. Select the ReadyFlow you want to use as a template for your flow design.
4. Click Create New Draft.
5. Select the Target Workspace where you want to create the draft.
6. Select the Target Project to which you want to assign the draft.

7. Provide a valid Draft Name for the draft.

   Flow names must be unique within a workspace. If a draft with the provided name already exists, you need to provide a different name.

8. Click Create.
   The selected ReadyFlow opens as a draft in the designated Flow Designer workspace with the Draft Name you provided, on the **Flow Design** canvas.

Parameters

The values of properties in the flow, including sensitive properties, can be parameterized. You can either create new parameters for your draft or import existing ones as a parameter group.

The values of properties in the flow, including sensitive properties, can be parameterized using Parameters. Parameters are created and configured within the Flow Designer. Any property can be configured to reference a Parameter with the following conditions:

- A sensitive property can only reference a sensitive parameter
- A non-sensitive property can only reference a non-sensitive parameter
- Parameters cannot be referenced in reporting tasks or in management controller services

Draft parameters

Any parameter defined in the root process group of the flow draft is a draft parameter. These parameters can be directly modified in the Flow Designer, their values always take precedence over any parameter in a shared parameter group.

Parameter groups

You can create groups of parameters that can be shared within a project. This allows you to fill parameter values for several flow definitions, without having to copy and paste those values every time they change. Furthermore, changing the value of a parameter in a parameter group is instantly reflected in any running test session which has components referencing the changed parameter. For example, you create a group of sensitive parameters for credentials, and you change the password. This change is disseminated in real time, and instead of failing, all test sessions keep on running without manual intervention.

Assets (file type parameters) are also available in parameter groups, and they have a very important advantage: you do not need a running test session to upload the asset, which is a limitation of draft parameters.

To prevent crashes by accidental or malicious deletion,, users cannot delete parameters in a parameter group that has a running component referencing it.

When creating a flow draft in **Flow Designer**, you encounter parameter groups in the following contexts:

While opening a flow definition from the Catalog

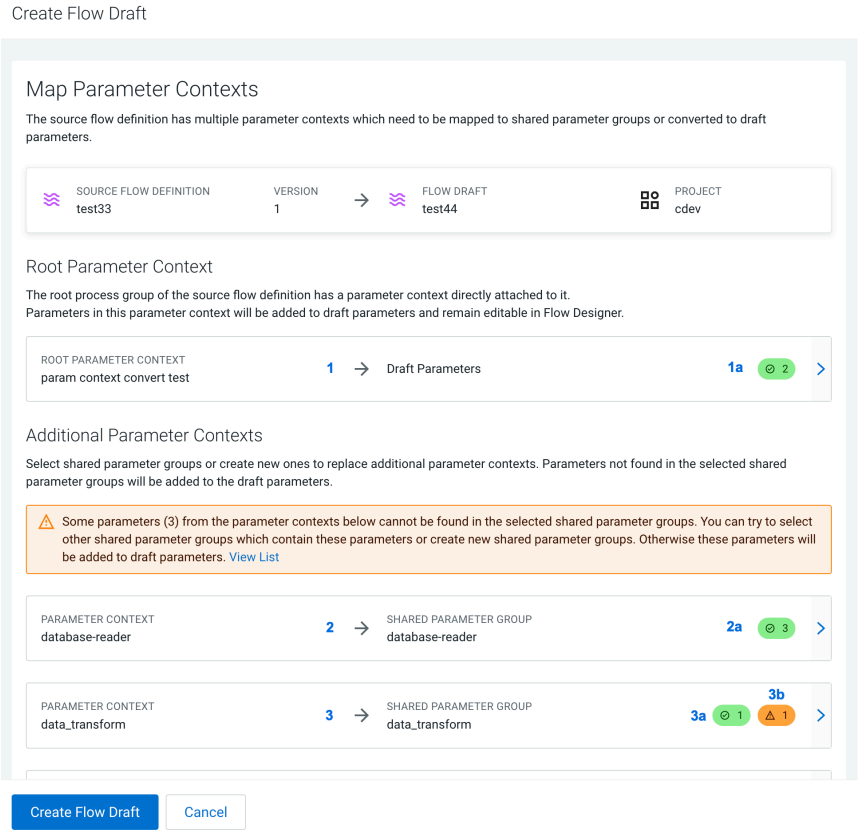

If the source flow definition contains more than one parameter context attached to the root process group, the Flow Designer matches those contexts to shared parameter groups that are assigned to the same project where the draft is created (In this respect ‘unassigned’ is treated as a project of its own: an unassigned draft is matched to unassigned parameter groups). Matching happens by name, and you are provided with a table of the results on the **Convert Parameters** page. In this view you can examine and modify the results of parameter matching. You can manually replace parameter contexts of the source flow definition with other shared parameter groups available in the project that you find more suitable to your needs. You can also create parameter groups from draft parameters.

While building a flow draft

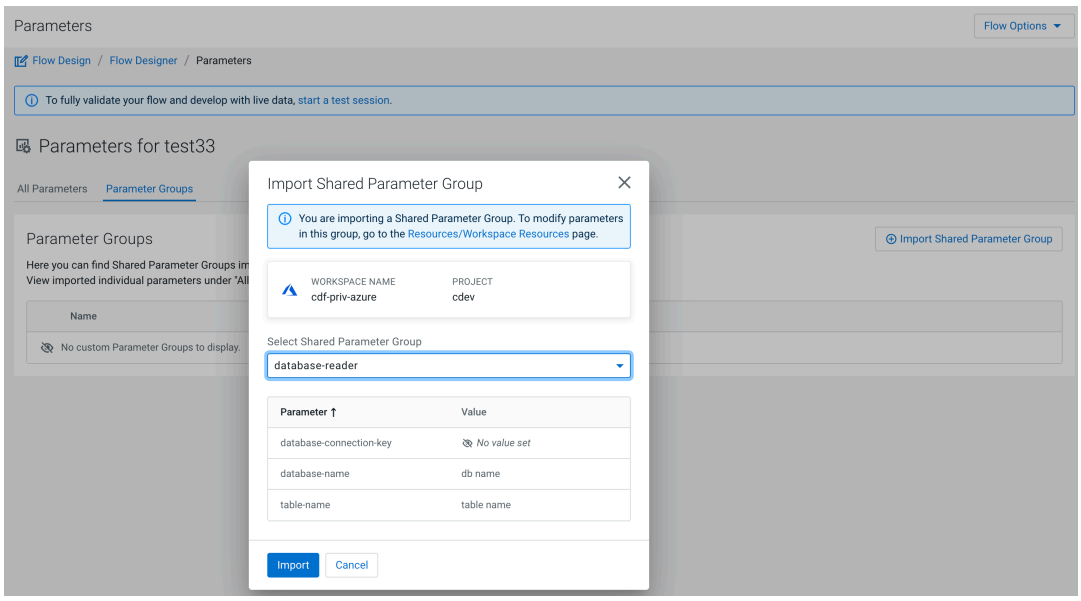

In the Parameters view of the Flow Designer you can import existing shared parameter groups and you can also export a number of draft parameters to a parameter group which can then be reused in multiple flows.

Note:

Draft parameters always take precedence over parameters in imported parameter groups. The precedence of parameters with overlapping values in imported parameter groups is defined by their order in the Flow Design Flow Designer Parameters Parameter Groups view. To change the order of precedence, simply reorder the imported parameter groups.

Relationship between shared parameters and draft parameters

Precedence

- The values of draft parameters always take precedence over imported parameters.
- In case of imported parameter groups, their order defines precedence. By changing the order of parameter groups, you can redefine precedence. As a result of this, if a parameter is present in several shared parameter groups, the value of the one in the topmost group takes precedence.

Value retention

- Both non-sensitive and sensitive parameters that are part of an imported parameter group keep the values they have in the shared parameter group. These values cannot be changed within the Flow Designer, only in the **Resources** view.
- Non-sensitive parameters that are added as draft parameters to the root parameter context during draft creation retain the default values they had in the source flow definition.
- Sensitive parameters that are added as draft parameters to the root parameter context during draft creation are added without values.

Adding a parameter to your draft

The values of properties in the flow, including sensitive properties, can be parameterized. Learn how to add a parameter to your draft.

About this task

Note:

Parameters are created within Parameter Contexts. In Flow Designer, one default Parameter Context is auto-created when you create a new draft. You can then add parameters to this one context, you cannot create additional ones. This default context is automatically bound to each Process Group you create within your draft, making all parameters available to be used in any process group.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow authorization.
- You have created a draft in Cloudera DataFlow Flow Designer.

Related Information

Cloudera DataFlow authorization

For more information on Parameters, see the NiFi User Guide.

Adding a parameter from Flow Options

Learn about creating parameters from the Flow Options menu.

Procedure

1. Click Flow Options Parameters Add Parameter Add Parameter or Add Sensitive Parameter.

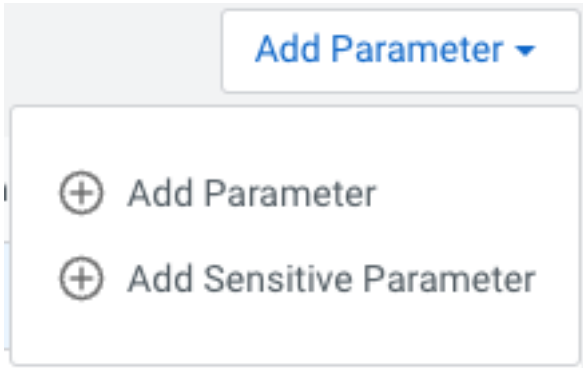

Note: A sensitive parameter cannot be later changed to non-sensitive and this is also true the other way around. Keep this in mind when creating parameters.

2. Provide the parameter Name, define a Value (or select Set empty string) and optionally provide a parameter Description.

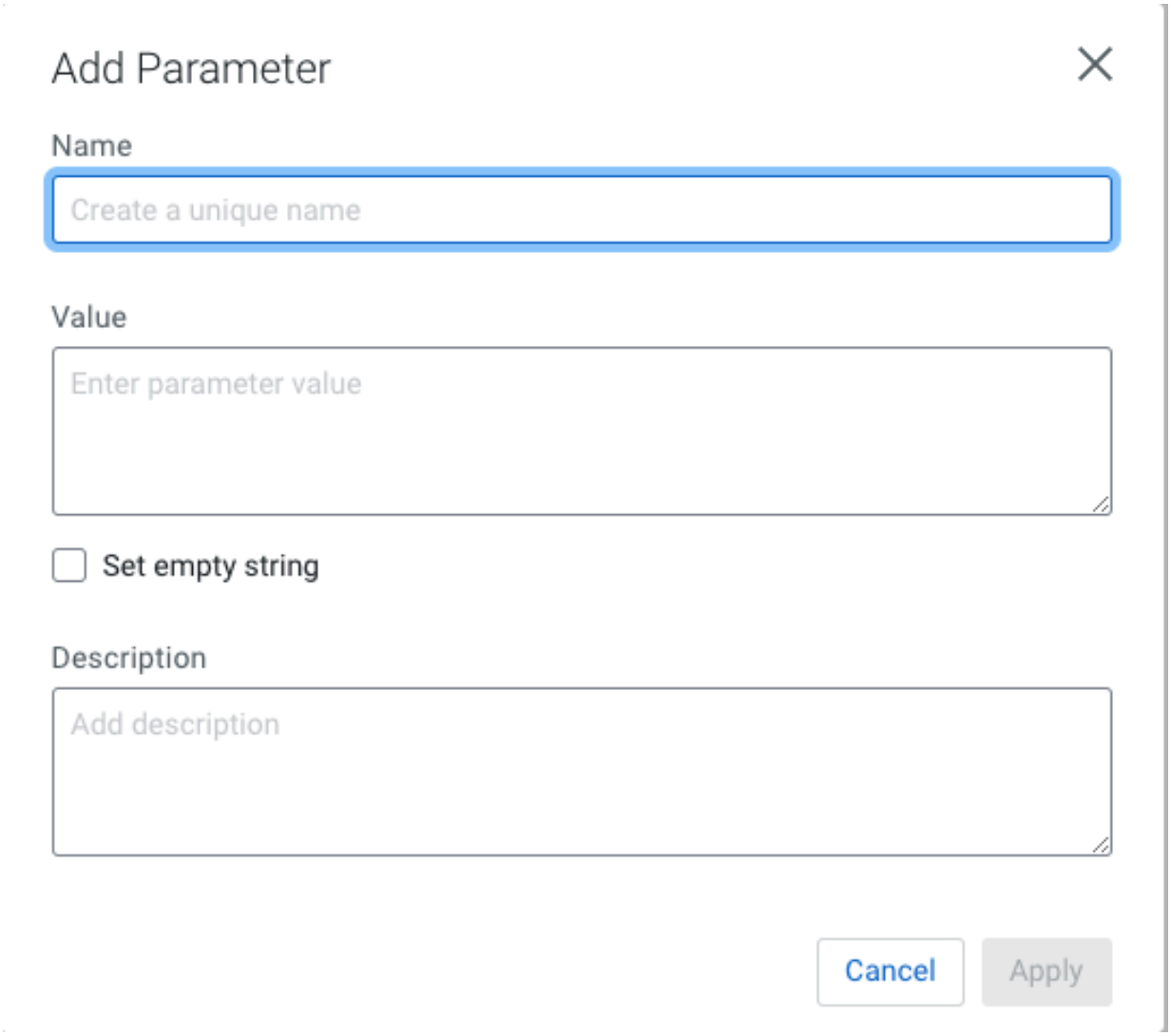

3. When you are done, click Apply.

4. For your parameter changes to take effect, click Apply Changes.

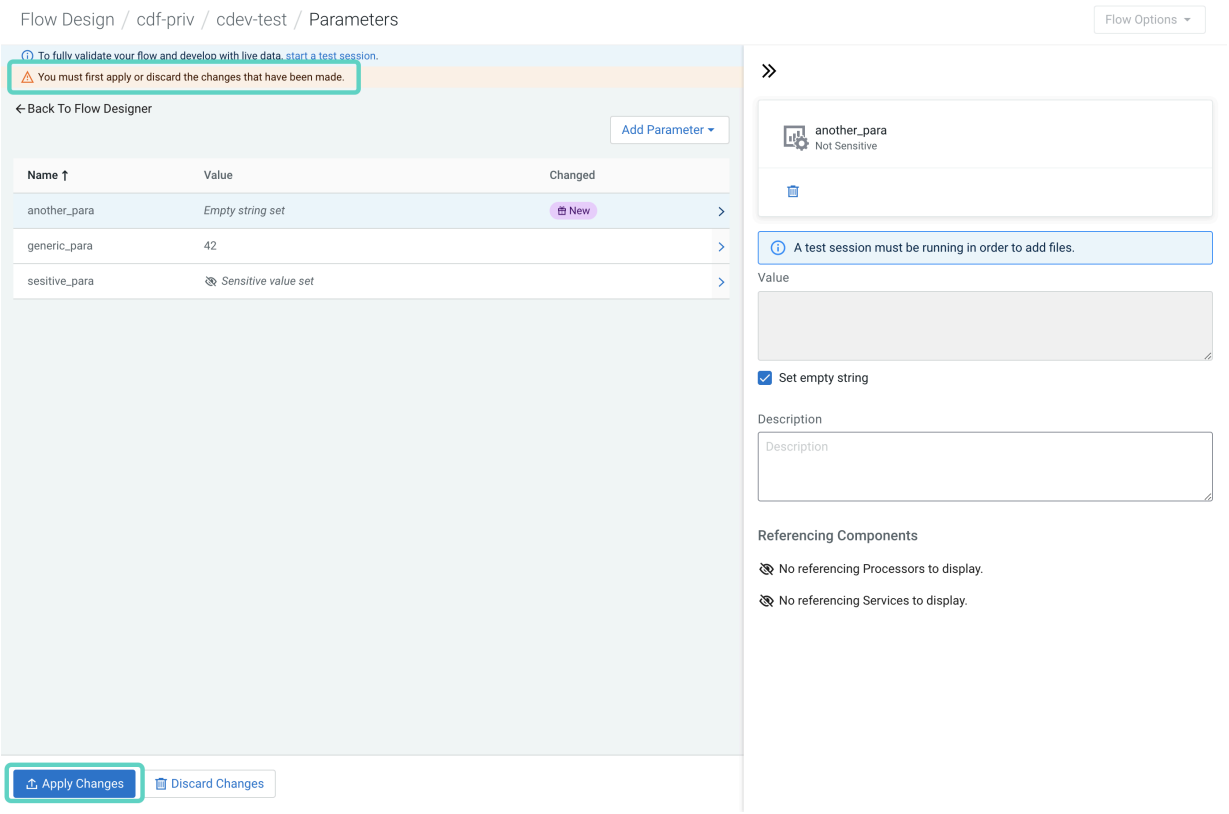

What to do next

Click ← Back to Flow Designer to return to the canvas when you are done adding parameters.

Creating a parameter from the Configuration pane

Learn about converting a property of a component into a parameter from the Configuration pane.

Procedure

1. Select the component with the Property that you want to convert to a parameter.
   Click the component on the Canvas to access its properties in the Configuration pane.

2. Locate the property that you want to turn into a parameter and click ⋮ Options ⋮ More Convert To Parameter .

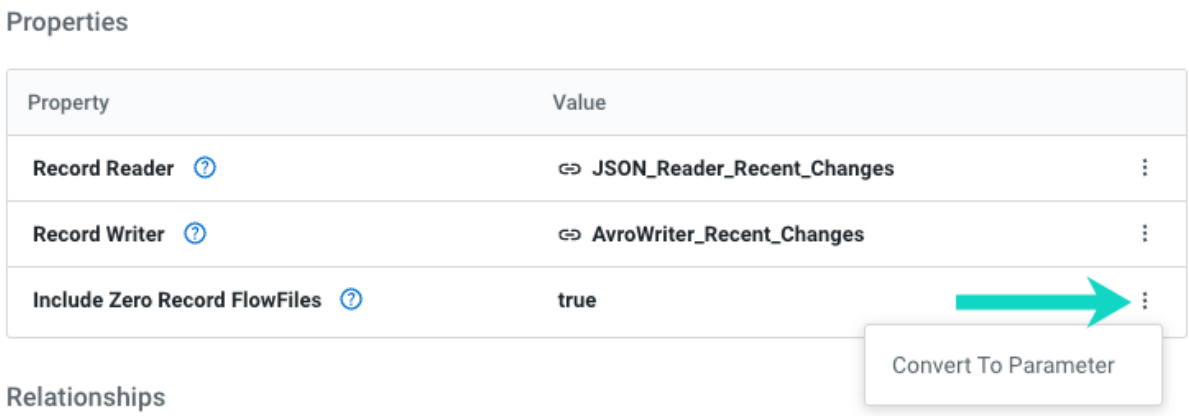

3. Edit the parameter Name, Value, and optionally add a description.
4. Click Add.
   The Value of the Property is automatically updated to a reference to the newly created parameter.
5. Click Apply to save the change you made to your component.

Note: Even if you click Discard Changes at this stage, the parameter is created, but the Property is reset to its previous Value.

To delete an unwanted parameter:

a. Go to Flow Options Parameters .
b. Click > in the row of the parameter to be deleted.
c. Click Delete.

Importing a parameter group

You can import existing parameter groups into your flow draft so you do not have to individually copy and paste values that are common in an environment, such as credentials or access paths.

About this task

One use case is to import a parameter group that was created earlier to reuse it in your draft. This is useful, for example, to share credentials, access paths, and so on between several flows within a project. Another use case is exporting draft parameters as a parameter group from the flow draft you are working on, then immediately importing the same group back to your draft to replace flow-specific draft parameters with ones that can be shared and reused.

Procedure

1. On the **Flow Design** canvas, select Flow Options Parameters .

2. Select the Parameter Groups tab and click Import Parameter Group.

   The **Import Parameter Group** modal opens.

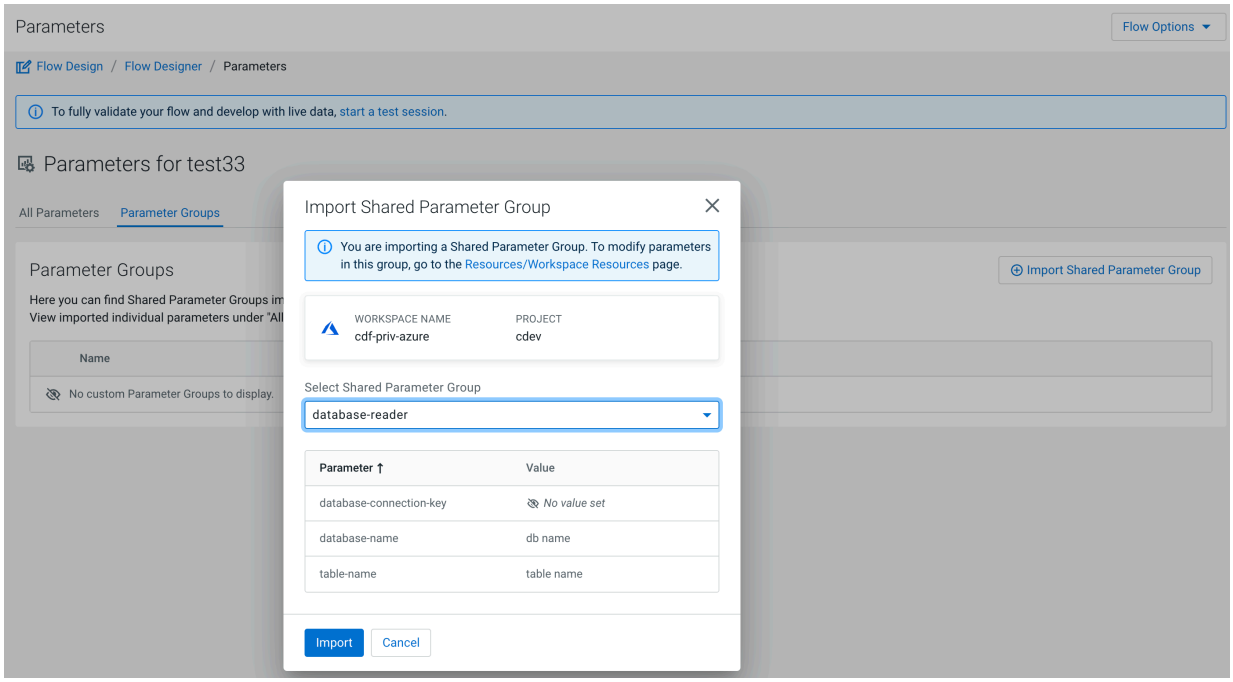

3. From the Select Parameter Group drop-down select the group you want to import.

   The list displays only the parameter groups available to you, that is, parameter groups that are either in a project where you have at least the DFProjectMember permission or parameters in the same workspace that are not assigned to any project.

4. Click Import.

5. To finalize the changes you made, click Apply Changes.

What to do next

After importing the parameter group, go to the **All Parameters** tab. There you can remove unnecessary draft parameters that have overlapping values with shared parameters. This is useful, because draft parameters always have precedence over parameters in an imported parameter group.

Exporting draft parameters to a parameter group

You can select a group of parameters and make them available for reuse in a workspace by exporting them as a parameter group.

About this task

When you want to share a group of parameters for reuse, use the Export Parameter Group option. This allows for the reuse of those parameters within the same project, without the need for copying and pasting them individually.

Before you begin

You need the DFFlowDeveloper role to manage flow drafts in the workload environment.

Procedure

1. On the **Flow Design** canvas, select Flow Options Parameters .

2. Select the All Parameters tab and click Export Parameter Group.
   The **Export to a Shared Parameter Group** modal opens.

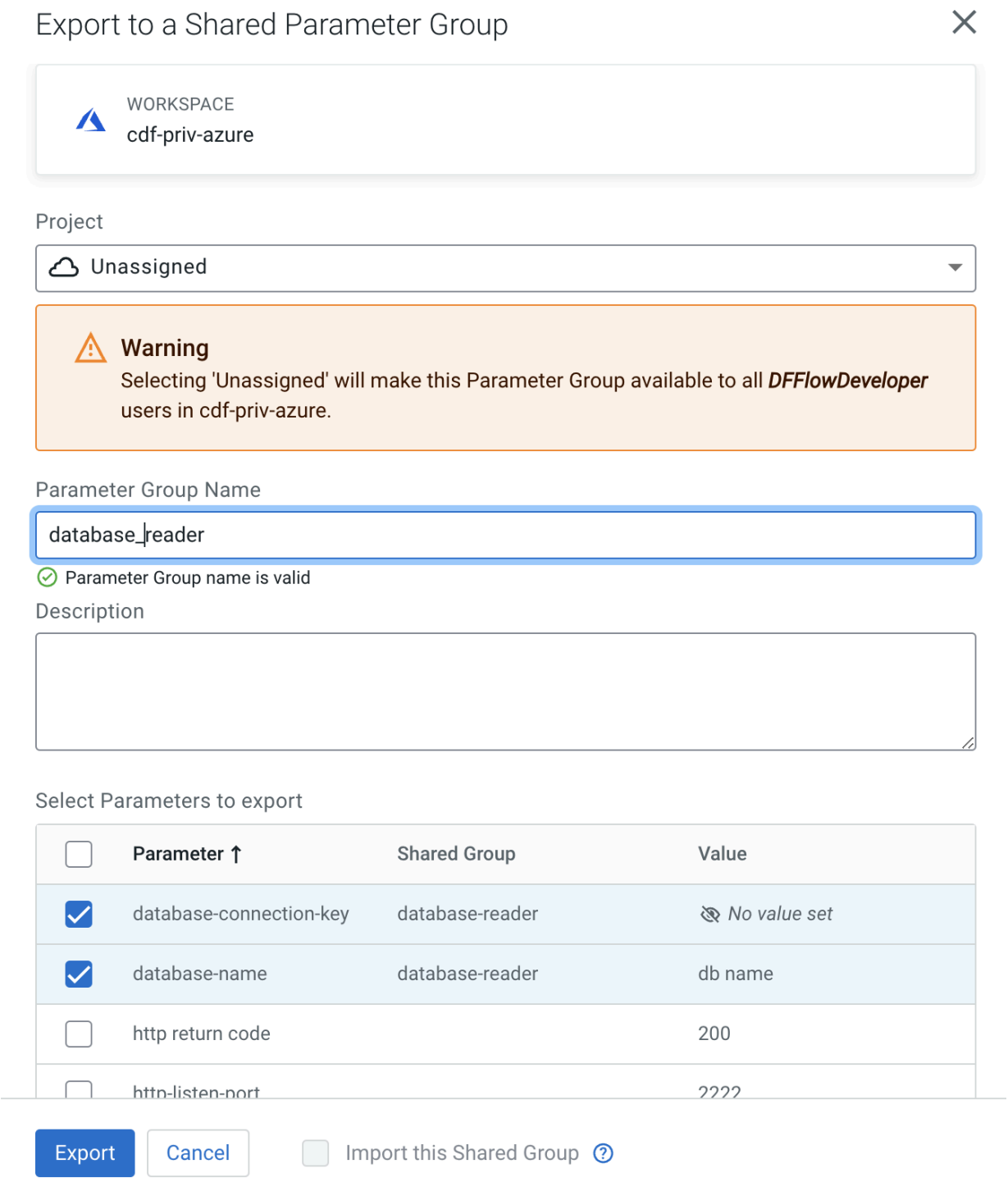

3. From the Project drop-down select an available project where you want to assign the new group, or leave it unassigned.

   Note: Leaving the parameter group unassigned will make it available to anyone with DFFlowDevoper role in the workspace.

4. Provide a Parameter Group Name. The name must be unique within a workspace, Cloudera DataFlow automatically validates the name.
5. You can optionally provide a Description of the parameter group.
6. Select Parameters to export by checking the boxes next to them. Select the checkbox on top to select all parameters.
7. If you select the Import this Shared Group option the newly created shared parameter group will be automatically imported to the draft from which the parameters were originally exported, as the parameter group with the highest priority. Draft parameters with matching names will be replaced with parameters in the shared parameter group.
8. Click Export.

Configuring advanced usage of the UpdateAttribute processor

The UpdateAttribute processor may be used to make conditional changes, based on user defined rules.

About this task

Rules are a set of Conditions and associated Actions. Each rule basically says, "If these conditions are met, then do this action." One or more conditions may be used in a given rule, and they all must be met in order for the designated action(s) to be taken.

In this topic we use two rules as an example to demonstrate how adding rules work: CheckForLargeFiles and Chec kForGiantFiles.

The CheckForLargeFiles rule has the following conditions:

- ${filename:equals('fileOfInterest')}
- ${fileSize:toNumber():ge(1048576)}
- ${fileSize:toNumber():lt(1073741824)}

The associated action for the filename attribute:

- ${filename}.meg

Taken together, this rule says:

- If the value of the filename attribute is fileOfInterest, and
- If the fileSize is greater than or equal to (ge) one megabyte (1,048,576 bytes), and
- If the fileSize is less than (lt) one gigabyte (1,073,741,824 bytes)
- Then change the value of the filename attribute by appending ".meg" to the filename.

Continuing with this example, we can add another rule to check for files that are larger than one gigabyte. When we add this second rule, we can use the previous rule as a template, so to speak, by selecting the Expand Duplicate option for the CheckForLargeFiles rule.

In this example, the CheckForGiantFiles rule has these conditions:

- ${filename:equals('fileOfInterest')}
- ${fileSize:toNumber():gt(1073741824)}

Then it has this action for the filename attribute:

- ${filename}.gig

Taken together, this rule says:

- If the value of the filename attribute is fileOfInterest, and

- If the fileSize is greater than (gt) one gigabyte (1,073,741,824 bytes)
- Then change the value of the filename attribute by appending ".gig" to the filename.

Procedure

1. To add rules to the UpdateAttribute processor and to define behavior when multiple rules match, go to « Configuration Pane Define Rules .

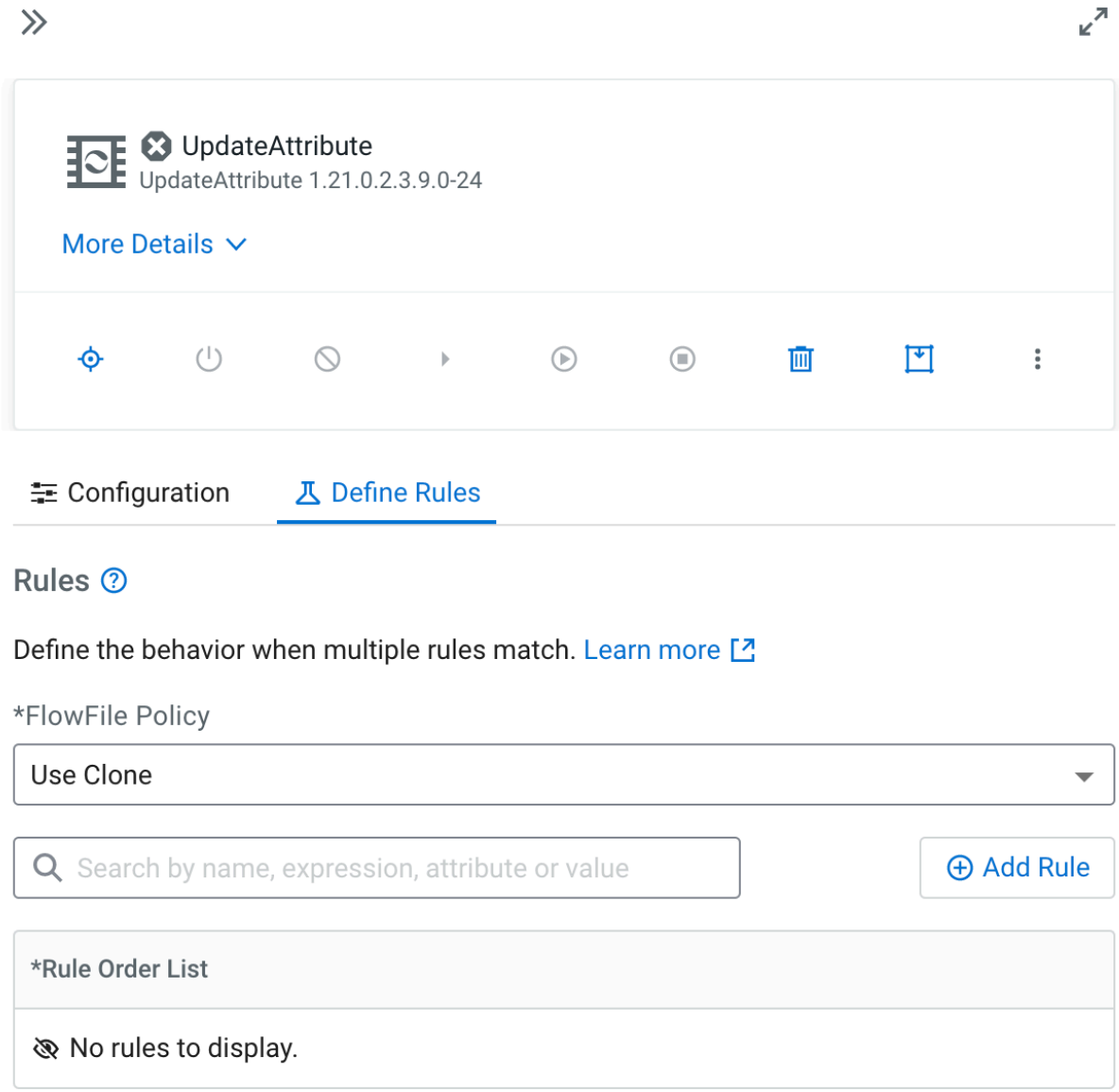

2. Select a Flow File Policy:

   Use Clone

   Matching rules are executed with a copy of the original flow file. This is the default behavior.

   Use Original

   Matching rules are executed with the original flow file, in the order that you specify under Rule Order List.

Add a Rule

Procedure

1. You can add rules by clicking Add Rule.
2. Specify a Rule Name.
   Specify CheckForLargeFiles
3. Add Conditions by clicking Add Condition.

   Note: All conditions you add here must be met for the rule to match.

   In the **Add Condition** modal window you can specify a condition using regular expressions.
   Add the ${filename:equals('fileOfInterest')}
   Click Add Condition in the modal window to save your changes.
4. To add Actions, click Add Action.
   In the **Add Action** modal window specify an Attribute name and a Value.
   For example, specify filename as Attribute and ${filename}.meg as Value.
   Click Add Action to save your changes.
5. Once you are done with creating or updating a Rule, click Apply to save your changes.

   Tip: As long as your rule configuration is invalid, the Apply button is inactive and you can only Discard Changes.

Reorder Rules

Procedure

1. To reorder Rules, grab one by the Drag Handle and drag it to the required position.
2. Click Apply.

Testing JoltTrasformJSON processor configuration

You can test the transformations a given Jolt Specification effects on a sample JSON file without starting a test session.

Procedure

1. To test Jolt Transform when using the JoltTransformJSON processor, go to Configure Test Jolt Transform .

2. You may modify the Jolt Transformation DSL and Jolt Spec fields. If you modify them here, click Apply to confirm the changes.

   The values for fields Jolt Transformation DSL and Jolt Spec are automatically retrieved from the processor Configuration if already set.

   Important: Any configuration changes you apply here overwrite the processor configuration.

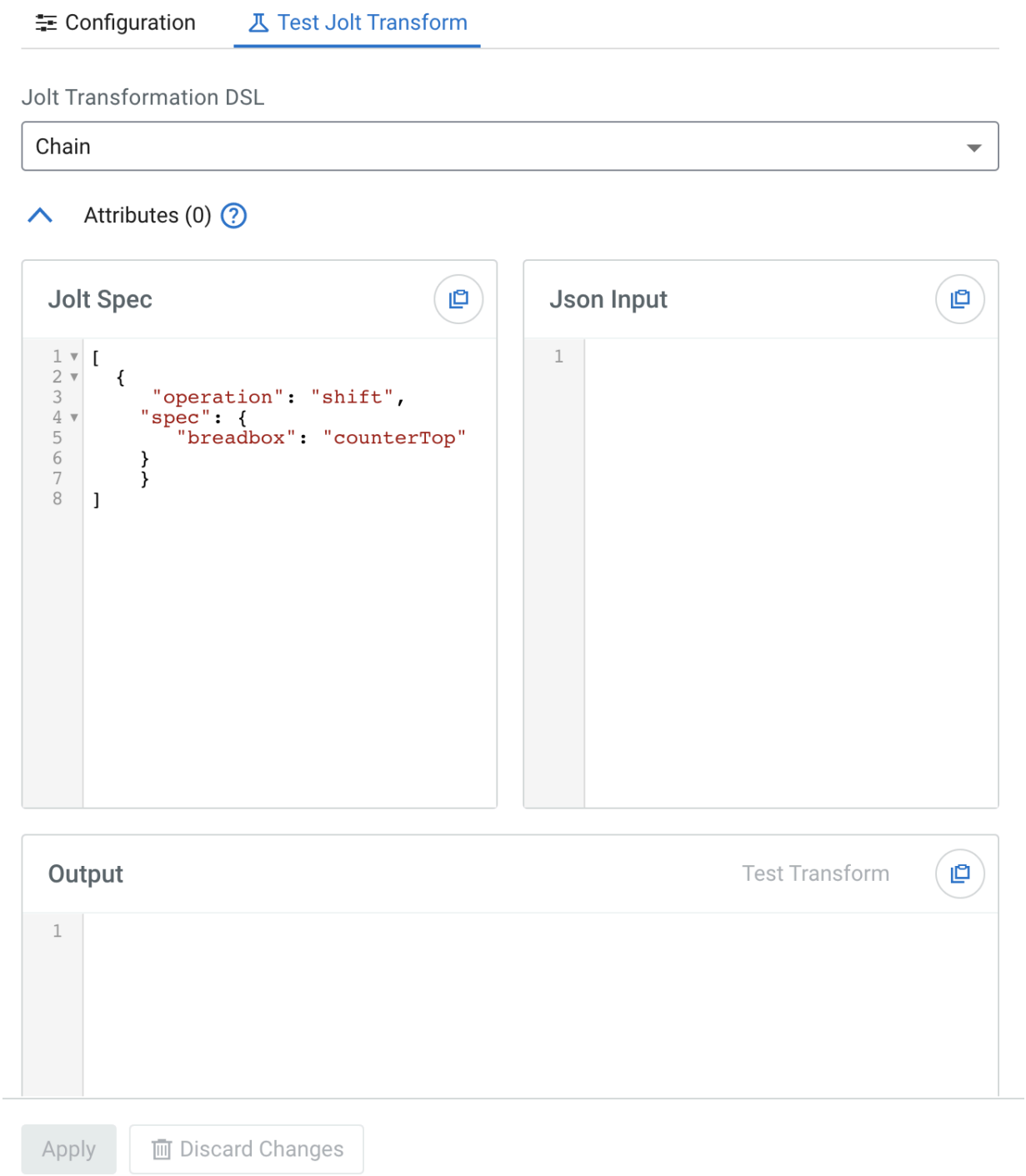

3. You may add attributes and values to reference within your specification for testing input. To do so, click ^ Attributes.

   Important: Attribute values entered here are only available during testing, but not when executing the actual flow.

4. Paste the input file contents into the Json Input box.
5. Click Test Transform.
   The result of the transformation is printed to the Output box.

Adding a service to your draft

Learn how you can add a service in Cloudera DataFlow Flow Designer.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow authorization.
- You have created a draft flow in Cloudera DataFlow Flow Designer.

Related Information

Cloudera DataFlow authorization

Adding a service from Flow Options

Learn about creating services from the Flow Options menu.

Procedure

1. To create a service for your flow, click Flow Options Services Add Service .
2. In the Add Service box you can select from the available services by scrolling down the list on the left or by starting to type the name of the service you are looking for.

   You can optionally change the name of the service.
3. Click Add.

   The Configuration pane opens.
4. Make the required settings to your service, then click Apply.

What to do next

Click Back to Flow Designer to return to the canvas when you are done adding services.

Creating a service from Configuration

Learn how you can create services directly from the Configuration pane for a property that expects a controller service as a value.

Procedure

1. Select the component with the Property for which you want to create the controller service.

   Click the component on the Canvas to access its properties on the Configuration pane.

2. Locate the property for which you want to create the service and click ⋮ More Add Service .

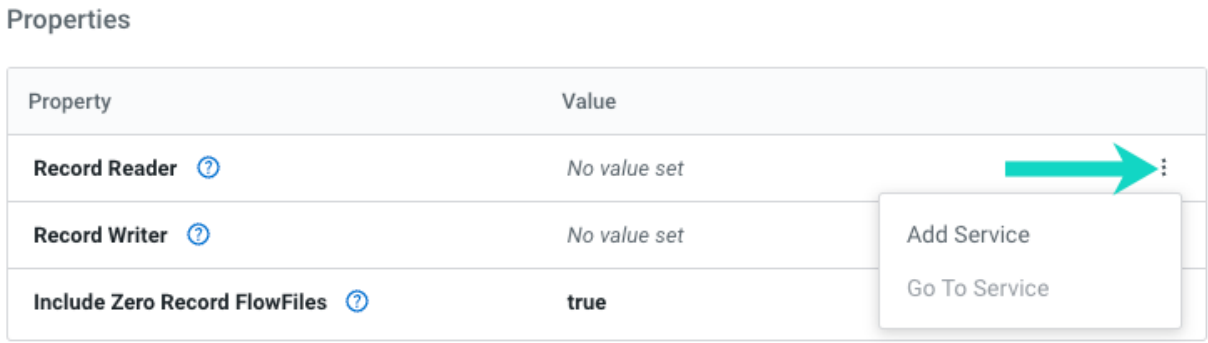

3. In the Add Service box you can select from the available services by scrolling down the list on the left or by starting to type the name of the service you are looking for.

   You can optionally change the name of the service.
4. Click Add.

   The Configuration pane opens.
5. Make the required settings to your service, then click Apply.

What to do next

Click Back to Flow Designer to return to the canvas when you are done adding services.

Adding inbound connection support to a draft

You can enable a dataflow to listen on a public endpoint to data sources that are outside your Cloudera Public Cloud environment by adding inbound connection support to your

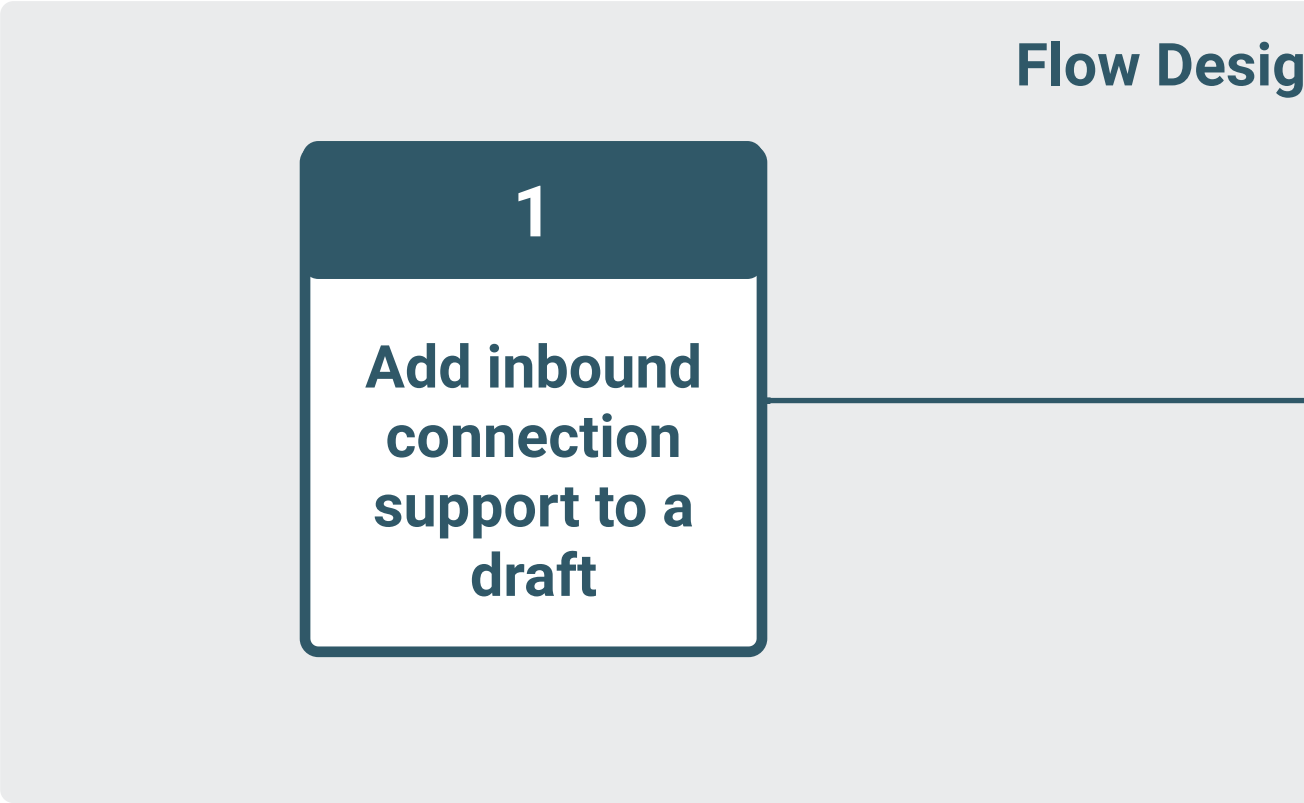

draft.

To enable your dataflow to use listen processors with inbound connection endpoint support, make the following addition to your draft:

1. Open the draft where you want to enable inbound connection support.
2. Add the required listen processor to your draft.

   Cloudera DataFlow supports all listen processors, including custom processors.
3. Configure the processor to enable inbound connections.

   Port

   Provide a port number where the flow deployment listens for incoming data. You can either add it as a parameter by clicking ⋮ Convert To Parameter to make the actual port number configurable during flow deployment, or you can set an explicit port number when creating the draft.

   SSL Context Service

   Create an external StandardRestrictedSSLContextService for your processor, by clicking ⋮ Add Service and filtering for StandardRestrictedSSLContextService. You must name this context service Inbound SSL Context Service. No other configuration is required. The SSL context service

will be created during cluster deployment and all other properties will be populated with values generated for that NiFi cluster.

Client Auth

Set to “REQUIRED” to use mTLS.

Note: Most listen processors only support TCP. ListenSyslog supports both TCP and UDP. As dataflow deployments do not support mixed protocols for listening ports, and UDP does not support SSL Context Service and Client authentication, Cloudera advises to configure your ListenSyslog processor to use TCP protocol. This allows your data flow to use SSL Context Service for authentication and to listen to different data sources on different ports.

If this is not possible, create separate flows that listen on UDP and TCP respectively.

Copying and pasting processors

In Flow Designer you can copy processors together with their configuration and then paste them to the same or another draft, or to an external application.

About this task

Copying a processor copies the entire processor configuration to the clipboard, even sensitive properties are kept. If you paste the processor to a different draft, you may still get an error as, for example, services are not copied together with your processor and you will have to recreate those in the new draft.

Procedure

1. Select the processor you want to copy then either right-click on it and select Copy to clipboard or press #+C.
2. To paste the processor, press #+V.

Tip: You can also paste processor configurations in JSON format to external applications, such as text editors and easily share them for troubleshooting purposes.

Cloudera DataFlow

Managing Drafts

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Publishing a draft as a new flow definition.....4

Publishing a draft as a new version of an existing flow definition.....5

Reassigning a flow draft.....5

Publishing a draft as a new flow definition

When you publish a new draft for the first time, or you have opened an existing flow definition from the Catalog as a draft, you can publish your updates as a new flow definition.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.
- Make sure you have DFCatalogAdmin permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.
- You have created a draft from an existing flow definition in Flow Designer.

Procedure

1. When you publish your draft as a new flow definition, you have two options: publishing from the Flow Designer Canvas view, and publishing from the All Workspace Drafts view.

   For Canvas

   a. In the Flow Designer Canvas view, click Flow Options Publish To Catalog .

   b. Depending on your use case, select one of the following:

   - Select Publish As if you have updated an exisiting flow definition and want to publish your work as a new flow.
   - Select Publish if you want to publish a new draft for the first time. (In this case, the Publish As option is inactive)

   For Workspace

   a. In the All Workspace Drafts view, click on the right of the row representing the draft you want to publish.

   b. Depending on your use case, select one of the following:

   - Select Publish As if you have updated an exisiting flow definition and want to publish your work as a new flow.
   - Select Publish if you want to publish a new draft for the first time. (In this case, the Publish As option is inactive)

2. Fill in the fields in the **Publish A New Flow** box.
   - Provide a Flow Name for your flow definition.

     You can only provide a name when you publish your flow for the first time.
   - Optionally provide a Flow Description.

     You can only provide a description when you publish your flow for the first time.
   - Optionally provide Version Comments.
3. Click Publish.

Results

Your draft is published to the Catalog as a new flow definition.

Publishing a draft as a new version of an existing flow definition

If you opened an existing flow definition from the Catalog as a draft, you have the option to publish your updates as a new version of the source flow definition.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.
- Make sure you have DFCatalogAdmin permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.
- You have created a draft from an existing flow definition in Flow Designer.

Procedure

1. When you publish your draft as a new version of an existing flow definition, you have two options: publishing from the Flow Design Canvas view, and publishing from the All Workspace Drafts view.

   For Canvas

   a. In the Flow Design Canvas view, click Flow Options Publish To Catalog Publish .

   For Workspace

   a. In the All Workspace Drafts view, click on the right of the row representing the draft you want to publish.

   b. Select Publish

2. Provide Version Comments in the **Publish A New Version** box.

   When publishing a new version to an existing flow definition, you cannot modify the Flow Name or Flow Description.

3. Click Publish.

Results

Your draft is published to the Catalog as a new version of the source flow definition.

Reassigning a flow draft

Cloudera DataFlow

CDF Flow Designer Overview

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Concepts 4

Flow design lifecycle 5

Flow Design landing page 5

Flow Design canvas 6

NiFi component documentation embedded in Flow Designer 7

Concepts

Learn about basic concepts of flow design and how those concepts relate to their NiFi counterparts.

Controller Service

Controller Services are extension points that provide information for use by other components (such as processors or other controller services). The idea is that, rather than configure this information in every processor that might need it, the controller service provides it for any processor to use as needed.

Draft

A draft is a flow definition in development, and its lifecycle is tied to the workspace it is created in. You can edit, update, and continuously test drafts by starting a Test Session. When your draft is ready to be deployed in production, you can publish it as a flow definition to the Cloudera DataFlow catalog.

Workspace

A workspace is where your drafts are stored. You can manage and delete drafts from this view. The workspace is automatically created when Cloudera DataFlow is enabled for an environment and is tied to the lifecycle of your Cloudera DataFlow service.

Environment

A logical environment defined with a specific virtual network and region on a customer's cloud provider account. After enabling Cloudera DataFlow for an environment, its service components run in an environment.

Parameter group

One default parameter group is auto-created when you create a new draft. You can then add parameters this one group, you cannot create additional ones. When you initiate a test session, a number of parameters are auto-generated in this parameter group. Depending on the configuration options you chose, these are necessary for your NiFi sandbox deployment, and also make it possible to integrate your NiFi instance with Cloudera Public Cloud components or external data sources.

This default group is automatically bound to each Process Group you create within your draft, making all parameters available to be used in any process group.

Processor

The Processor is the NiFi component that is used to listen for incoming data; pull data from external sources; publish data to external sources; and route, transform, or extract information from FlowFiles.

For more information, see Apache NiFi User Guide.

Process group

When a data flow becomes complex, it often is beneficial to reason about the data flow at a higher, more abstract level. NiFi allows multiple components, such as Processors, to be grouped together into a Process Group.

In Cloudera DataFlow Flow Designer, creating a draft automatically creates a process group with the name you specified when creating the draft, acting as the root process group. You can create child process groups as you see fit to organize your data processing logic.

For more information, see Apache NiFi User Guide.

Service

A logical environment defined with a specific virtual network and region on a customer's cloud provider account. Cloudera DataFlow service components run in an environment. In the Cloudera

DataFlow Service, you can provision a workload. This workload allows you to create a number of NiFi Deployments.

Test session

Starting a test session provisions NiFi resources, acting like a development sandbox for a particular draft flow. It allows you to work with live data to validate your data flow logic while updating your draft. You can suspend a test session any time and change the configuration of the NiFi cluster then resume testing with the updated configuration.

Flow design lifecycle

Drafts have a set lifecycle. They are created, built, and tested in Flow Designer, published to the Catalog and finally deployed through the Deployment Wizard or the CLI as flow deployments.

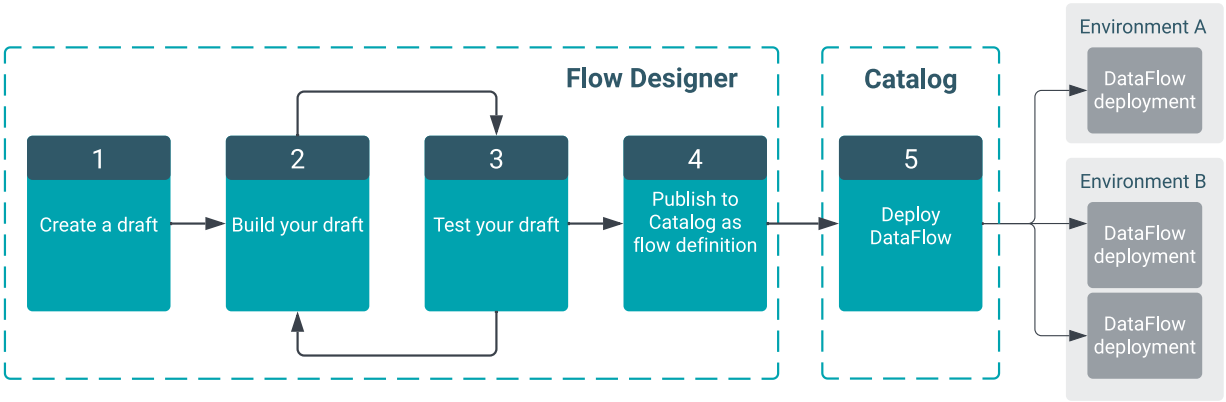

Flows created with Flow Designer have a set lifecycle. You start with creating a draft in Flow Designer. You can start from scratch, but you can also open an existing flow definition from the catalog, or a ReadyFlow from the gallery to use it as a template for your draft. You then build up your flow on the flow design canvas by adding and configuring components, establishing relationships between them, creating services, and so on. Whenever you feel the need, you can start a test session to verify what you have built so far. Starting a test session commissions a NiFi sandbox allowing you to test and simultaneously update your draft. Once you deem your draft to be production ready, you publish it to the catalog as a flow definition. Once published, you can keep updating your draft and publish it as a new version of your flow definition. Should this draft be lost, for example because the workspace holding it got deleted, you can always create a new one from the flow definition you have published the catalog.

Flow Design landing page

The Flow Design landing page provides a read-only view of all drafts you have access to.

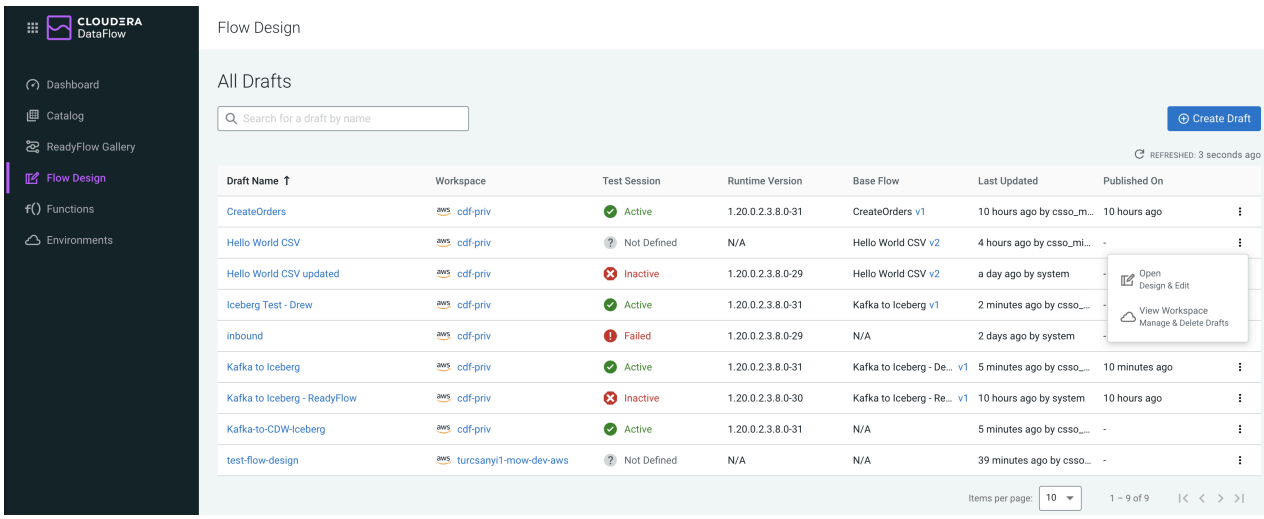

From the Cloudera Public Cloud home page, click Cloudera DataFlow, then click Flow Design. You are redirected to the Flow Design landing page. On the landing page you see all drafts that are both located in an environment where you have the DFDeveloper role, and are either unassigned or assigned to a Project where you have the DFProjectMember role.

- To create a new draft, click ⊕ Create Draft and follow the instructions.
- To interact with an existing draft, you can
  - click the name of the draft to open it on the **Flow Designer** Canvas.
  - click ⋮ Actions and select Open to open it on the **Flow Designer** Canvas, or select View Workspace. In the **All Workspace Drafts** view you can manage (edit, publish as a flow definition, delete) all drafts that are located in that given workspace.

Related Information

Projects

Flow Design canvas

Learn about getting around and performing basic actions on the Flow Design Canvas.

Components sidebar

You can add components to your draft by dragging them from the Components sidebar and dropping them onto the Canvas.

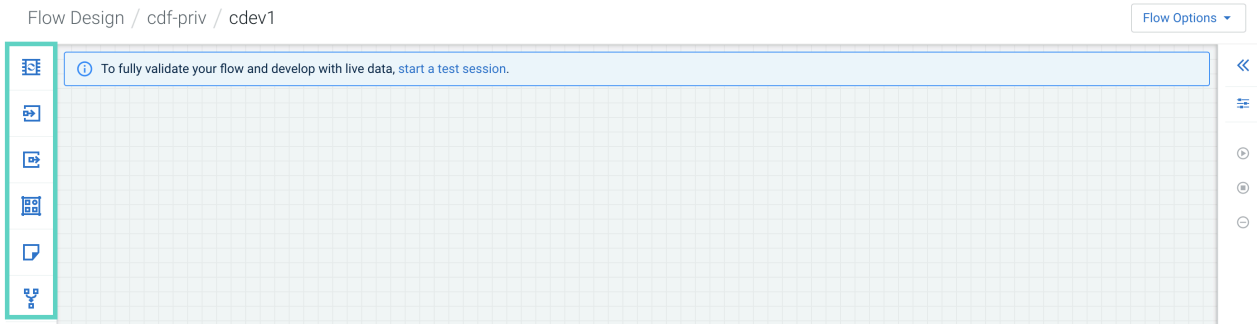

Configuration pane

To access the Configuration pane, click « Expand in the upper-right corner of the workspace. The Configuration pane is context-sensitive. It always displays the settings belonging to the element selected on the Canvas. When you create a new draft and the Canvas is empty, it defaults to the Process Group that was auto-created with the draft.

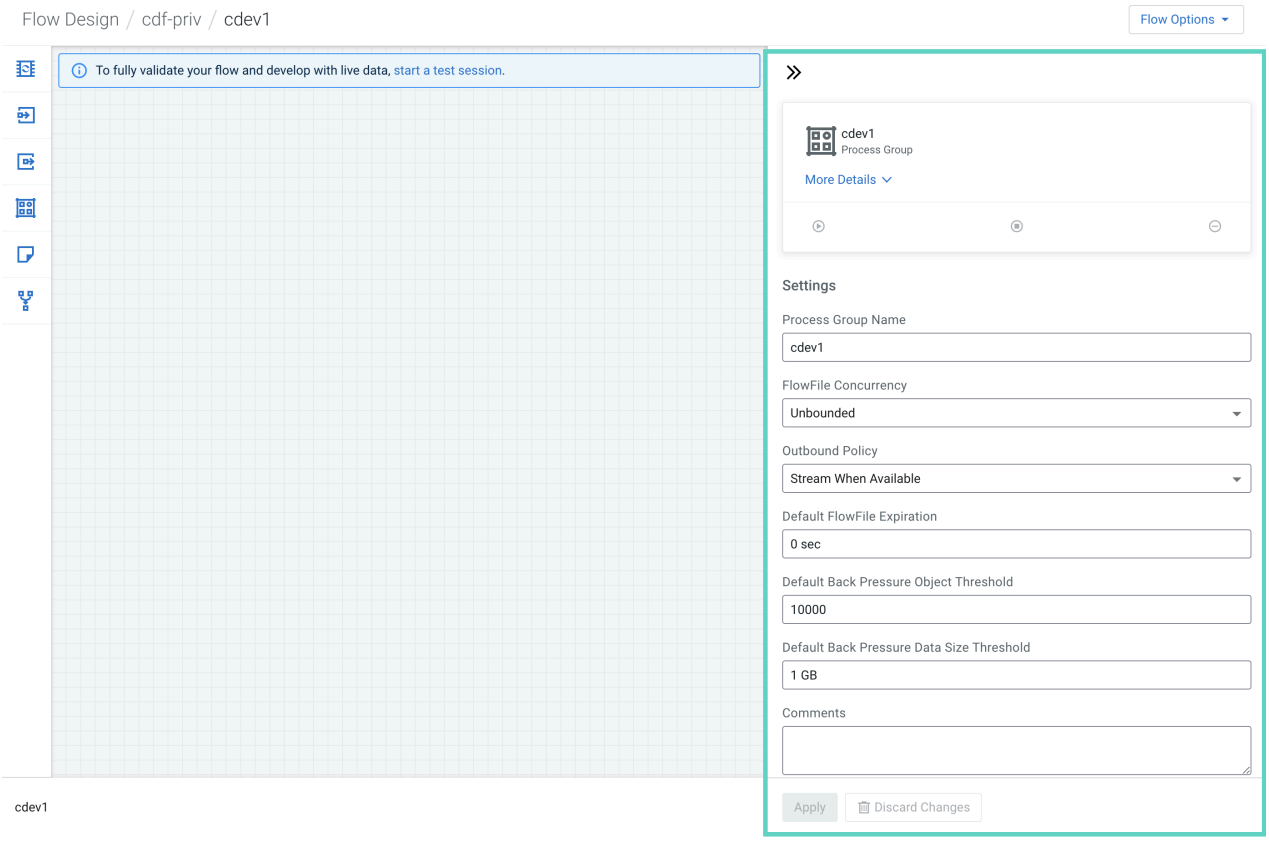

To hide the Configuration pane, click » Hide.

Tip: You can also open the Configuration pane by double-clicking the component that you want to configure on the Canvas.

NiFi component documentation embedded in Flow Designer

The standard Apache NiFi component documentation is directly available in Flow Designer.

To access the component documentation, right-click on a component in the Canvas and select Documentation.

Note: The embedded component documentation is retrieved from Apache NiFi and may contain information that is not valid in the context of Cloudera DataFlow Flow Designer.

Cloudera DataFlow

Test Sessions

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Test sessions 4

Starting a test session 4

Connecting an external application to a test session 5

Stopping a test session 6

Restarting a test session 7

Verifying flow file content 7

Queue management 12

Emptying a queue 12

Emptying all queues 13

Test sessions

Starting a Test Session provisions NiFi resources, acting like a development sandbox for a particular draft. It allows you to work with live data to validate your data flow logic while updating your draft. You can suspend a test session any time and change the configuration of the NiFi cluster, then resume testing with the updated configuration.

Starting a test session

To validate your draft flow, start a test session, provisioning a NiFi cluster.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.
- You have created a draft in Flow Designer.

Procedure

1. Select Flow Options Test Session .
   - Add Custom NAR Configuration if your flow requires custom processors that you have built.
   - Select Inbound Connections Allow NiFi to receive data if your your draft uses listen processors with inbound connection endpoint support.

     The deployment wizard auto synchronizes port information with the configuration settings you made for your listen processor(s).

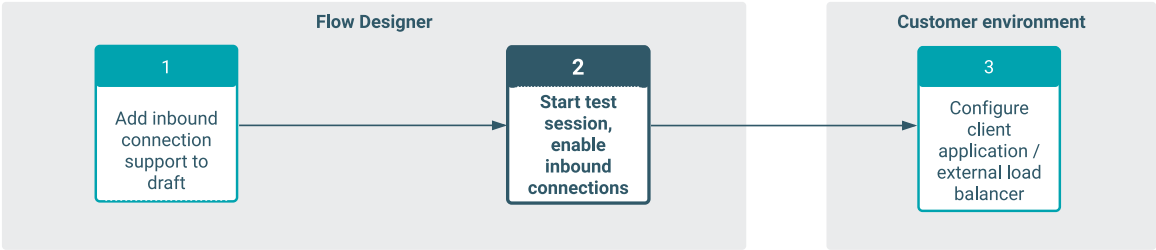

   Note: You can also change NiFi version but, unless you have a specific reason to select an earlier one, Cloudera recommends using the latest version.

2. Click Start Test Session.

   Test Session status changes to

Initializing Test Session...
3. Wait for the status to change to

Active Test Session.
4. Click Flow Options Services to enable Services.
5. Select a service you want to enable, then click ⏻ Enable.

   Repeat this step for all Services.

   Tip: You can also select Enable Service and Referencing Components. It will not only enable the controller service, but also any component that references it. This way, you do not need to enable the component separately to run the test session.

6. Click Back To Flow Designer to return to the flow design Canvas.
7. Start all processors that are not running (you see a Stopped symbol next to the processor name) by selecting the processor on the Canvas then clicking Start.

Results

The flow starts executing. On the Flow Design Canvas you can observe statistics on your processors change as they consume data and execute their respective tasks. You can also observe one or more blue Notification Pill, providing information about the current task. Potential issues that need to be addressed are called out in an orange Notification Pill on the individual processors.

Connecting an external application to a test session

Once a test session with inbound connection is running, you can go on and connect an external application to start sending data.

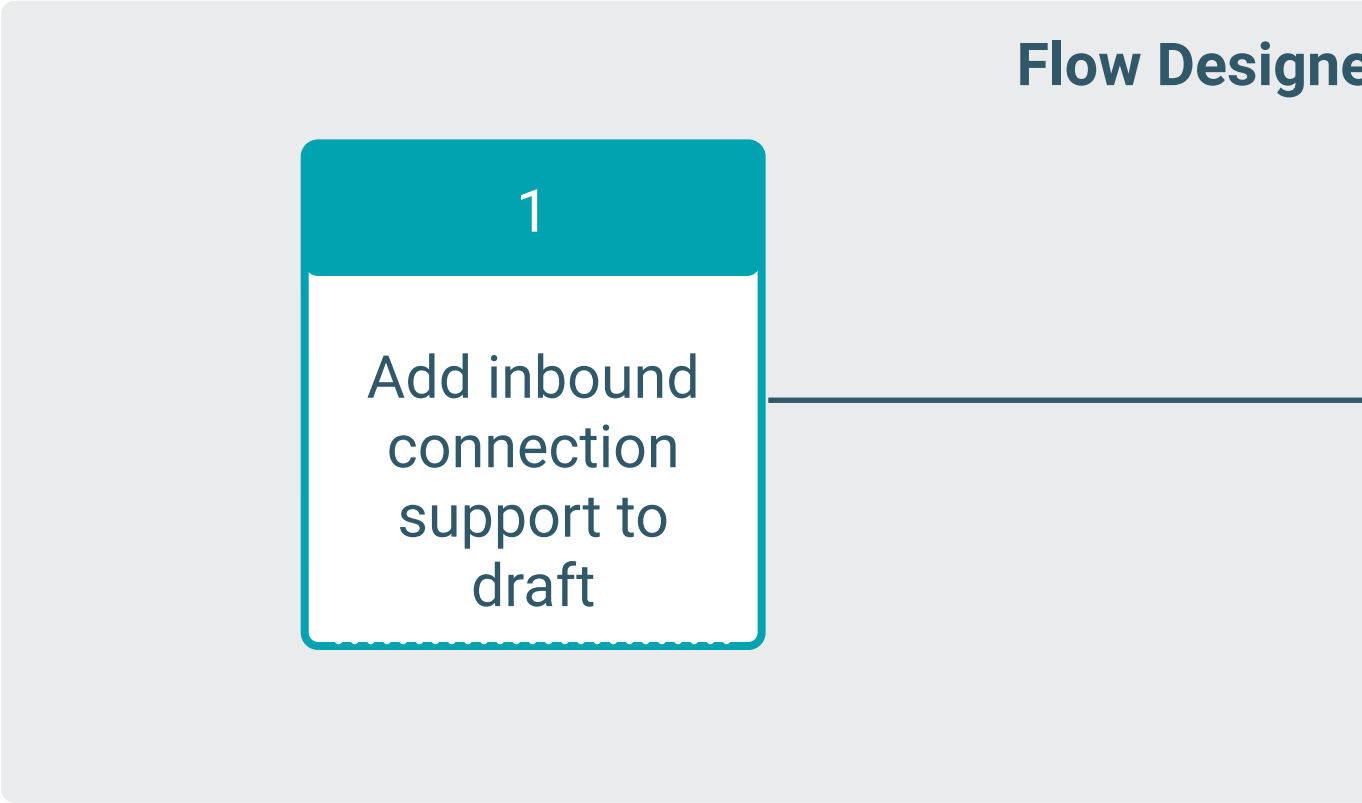

Before you begin

- A test session with inbound connection configured is running.
- A network connection through which the client can reach the NiFi endpoint is available.
- You have been assigned at least the DFFlowDeveloper role for the environment to which you want to configure the inbound connection.

Procedure

1. Select the draft where you want to send data and go to Flow Options Inbound Connection Details .
2. Make a note of the ENDPOINT HOSTNAME and LISTENING PORTS.
3. Click Download Client Certificate.

   The X.509 client certificate downloads to your computer in PEM format.
4. Click Download Private Key to obtain the RSA Private Key.

   The unencrypted RSA Private Key encoded with PKCS8 downloads to your computer in PEM format.
5. Depending on your client, you may have to convert the certificate and the private key to a different format.

   For example, to convert PEM to PKCS12 format, use the following command:

   ```
   openssl pkcs12 -export -in [***DOWNLOADED PEM CERT FILE***] -in
   key [***DOWNLOADED PEM PRIVATE KEY***] -out certificate.p12
   ```

   To further convert the PKCS12 file to JKS format for a Java client, run the following command:

   ```
   keytool -importkeystore -srckeystore [***CERTIFICATE NAME***].p12 -srcsto
   retype pkcs12 -destkeystore [***DESTINATION KEYSTORE***].jks
   ```
6. Add the certificate file and the private key files to the keystore of your application.
7. Configure your application to stream data to the Endpoint Hostname, port, and protocol of the NiFi instance behind the Test Session.

Stopping a test session

You can stop a test session to reconfigure your NiFi cluster. You can continue editing your draft flow in the meantime.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.
- You have an active test session.

Procedure

1. You have two options to end an active test session:
   - Click the toggle next to the Active Test Session notification.
   - Click Flow Options then click Stop under Test Session.
2. Click End.

   Test session status first changes to Ending Test Session... Ending Test Session... then, after the NiFi cluster shuts down, it changes to Test Session Not Active Test Session Not Active.

What to do next

Once the test session stops, you can update the configuration of your NiFi testbed cluster. To do so, select Flow Options Test Session Edit Configuration .

Restarting a test session

Learn about restarting an inactive test session.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.
- You have an inactive test session.

Procedure

1. You can optionally update the configuration of your NiFi development sandbox before restarting it. Click Flow Options then click Edit Configuration under Test Session.
2. You have two options to restart an inactive test session:
   - Click the toggle next to the Test Session Not Active Test Session Not Active notification.
   - Click Flow Options then click Restart under Test Session.

Verifying flow file content

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.
- You have an active test session.

Procedure

1. On the Flow Design Canvas, click the connection where you want to inspect the flow files.

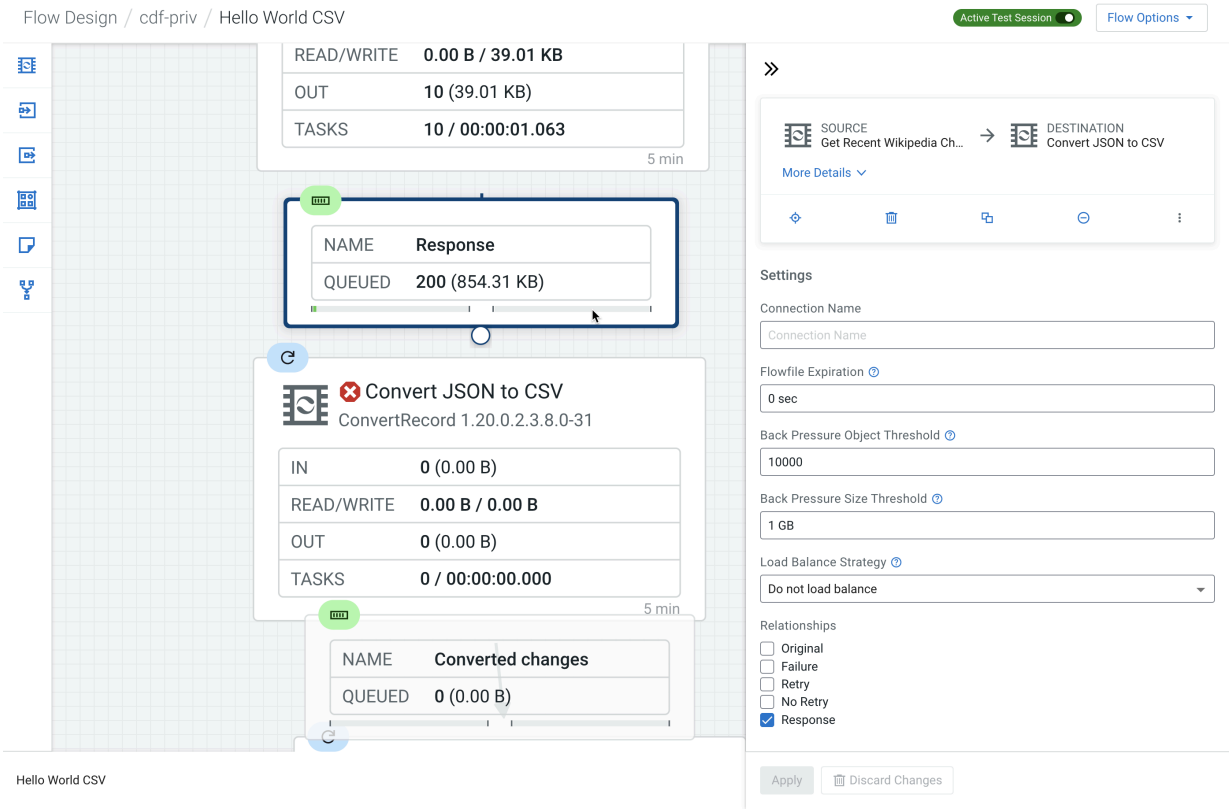

2. In the Configuration pane select ⋮ List Queue .
In the List Queue view, All Queued Flow Files for [***RELATIONSHIP NAME***] Relationship opens.

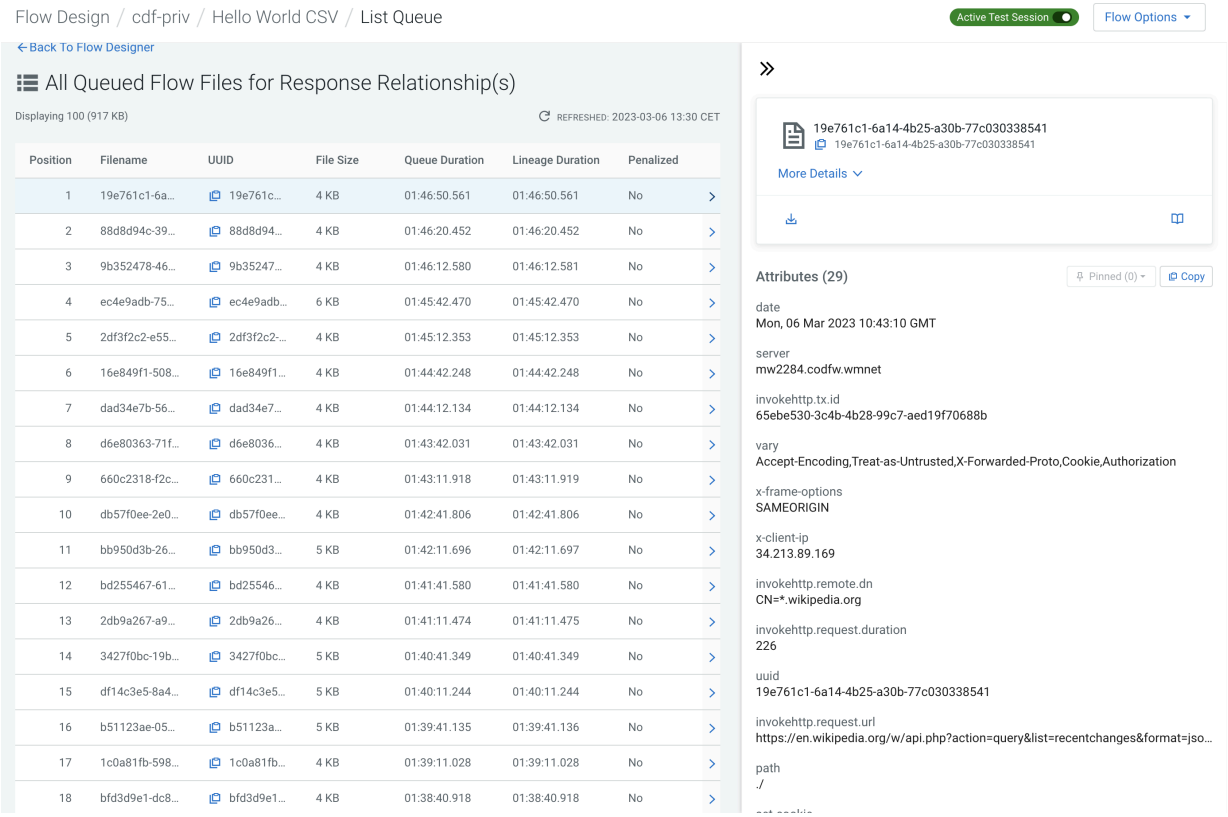

3. Select a flow file by clicking it to view its Attributes in the right-hand pane.

4.

If there is an attribute that is of special interest to you, hover the cursor over it and select Pin this attribute The selected attribute is pinned to the top of the list of attributes. Pinning persists, if you select a different flow file, or return to the Canvas and decide to view flow files later, you will find the pinned attribute(s) at the top of the list.

19e761c1-6a14-4b25-a30b-77c030338541
19e761c1-6a14-4b25-a30b-77c030338541

More Details

Attributes (29)

Pinned (1) Copy

invokehttp.status.code
200

date
Mon, 06 Mar 2023 10:43:10 GMT

server
mw2284.codfw.wmnet

invokehttp.tx.id
65ebe530-3c4b-4b28-99c7-aed19f70688b

vary
Accept-Encoding,Treat-as-Untrusted,X-Forwarded-Proto,Cookie,Authorization

x-frame-options
SAMEORIGIN

x-client-ip
34.213.89.169

invokehttp.remote.dn
CN=*.wikipedia.org

invokehttp.request.duration
226

uuid
19e761c1-6a14-4b25-a30b-77c030338541

invokehttp.request.url
https://en.wikipedia.org/w/api.php?action=query&list=recentchanges&format=jso...

5. To view flow file content, click Open in Data Viewer on top of the Attributes pane.

Tip:

- Click More details to view file metadata.
- To format the file printout, select an available grammar from the View as drop-down.

The contents of the flow file is displayed in a new tab in your browser.

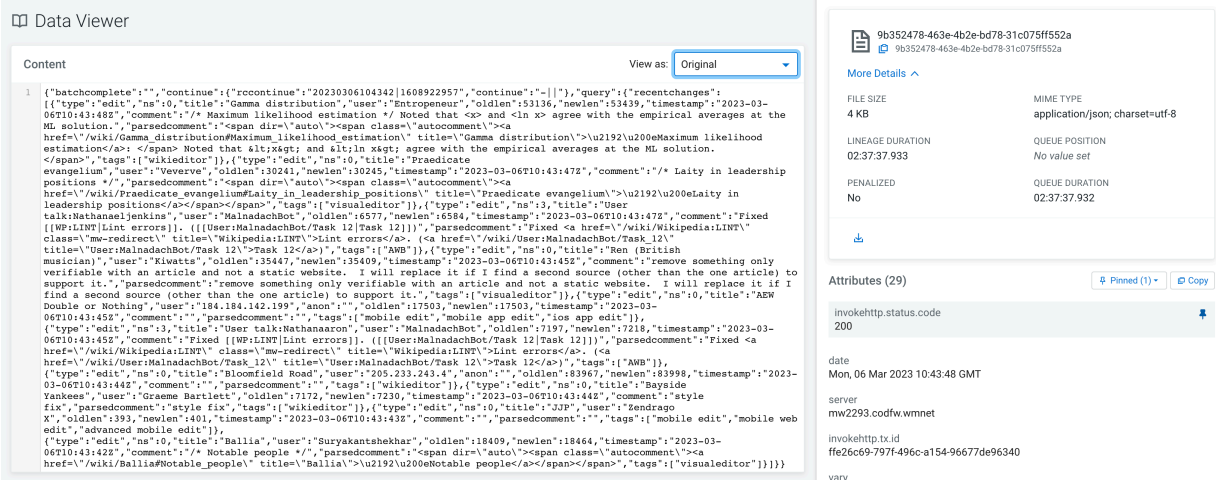

Queue management

You can remove all flow files from a selected connection or all flow files on process group level.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.
- You have an active test session.

Procedure

- To view the queue of a connection, right click on the connection and select List Queue.
  The top 100 flow files are listed on the **List Queue** page.

Emptying a queue

Learn how to delete enqueued flow files from a connection.

Procedure

1. Right click on the connection where you want to delete the enqueued flow files.
2. Select Empty Queue.
3. Click Empty.

   All FlowFiles waiting on the selected connection at the time of the request are removed.

Emptying all queues

Learn how to delete enqueued flow files from all connections on Process Group level.

Procedure

1. Click on an empty area of the Canvas.
   The **Configuration** pane on the right displays configuration options for the process group you are currently in.
2. Click Empty All Queues.
3. Click Empty.

   All FlowFiles waiting on all connections within the process group at the time of the request are removed.

Cloudera DataFlow

DataFlow Functions in Google Cloud Functions

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating your first Google Cloud function....4

Configuring your Google Cloud function.... 6

Service account....10

Output ports.... 11

Parameters.... 11

Cold start.... 12

Cloud storage....13

Data flow state.... 14

Configuring Kerberos....14

Handling failures....15

Testing your Google Cloud function.... 15

Monitoring and logs.... 16

Adjusting logs levels.... 17

Google Cloud Function triggers....17

Creating a Google Cloud function using CLI....21

Creating your first Google Cloud function

Google Cloud Functions is a serverless execution environment for building and connecting cloud services. You can write simple, single-purpose functions that are attached to events emitted from your cloud infrastructure and services. Your function is triggered when an event being watched is fired.

About this task

Follow these steps to create a Google Cloud function that is able to run Cloudera DataFlow Functions:

Procedure

1. Navigate to the Google Cloud Function page in your cloud account in the region of your choice.

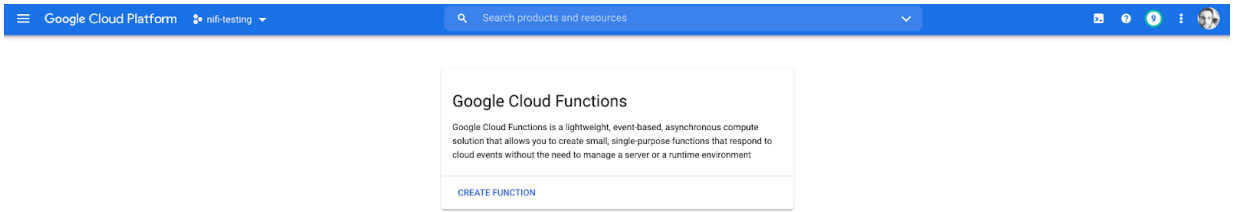

2. Click CREATE FUNCTION.

   You can now create your function.

   a) Give your function a meaningful name for your use case.
   b) Select the region where your function should be running.
   c) Select the trigger that you want to use with your function and configure it.

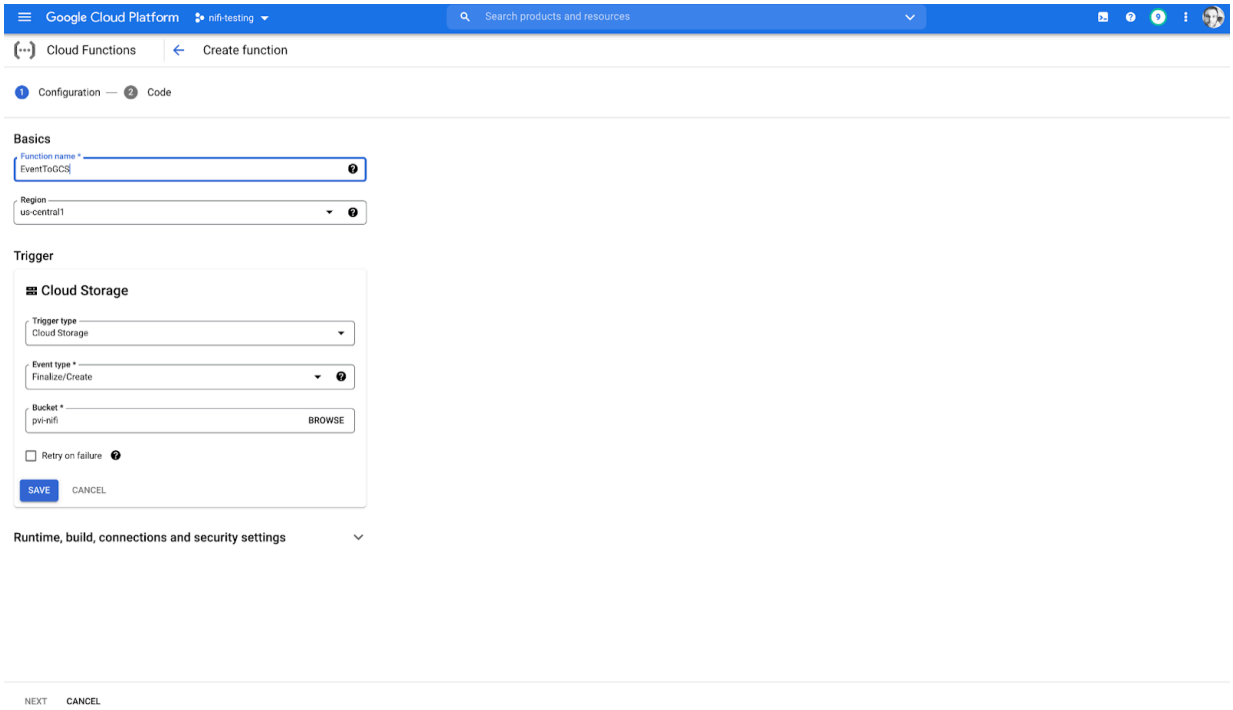

3. Click SAVE.
4. Expand the Runtime, build, connections and security settings section.

5. Configure Memory allocation and Timeout on the Runtime tab.

   For more information on recommendations, see the *Memory and timeout in runtime configuration* section in *Configuring your Google Cloud function*.

6. Select a Runtime service account.

   For more information, see *Service account*.

7. Configure the Runtime environment variables. For more information, see the *Runtime environment variables* section in *Configuring your Google Cloud function*.

8. You may also want to switch to the Security tab and reference a secret for sensitive parameters.

   For more information on a secure mechanism for storing parameters, see the *Google Cloud Secret Manager* section in *Parameters*.

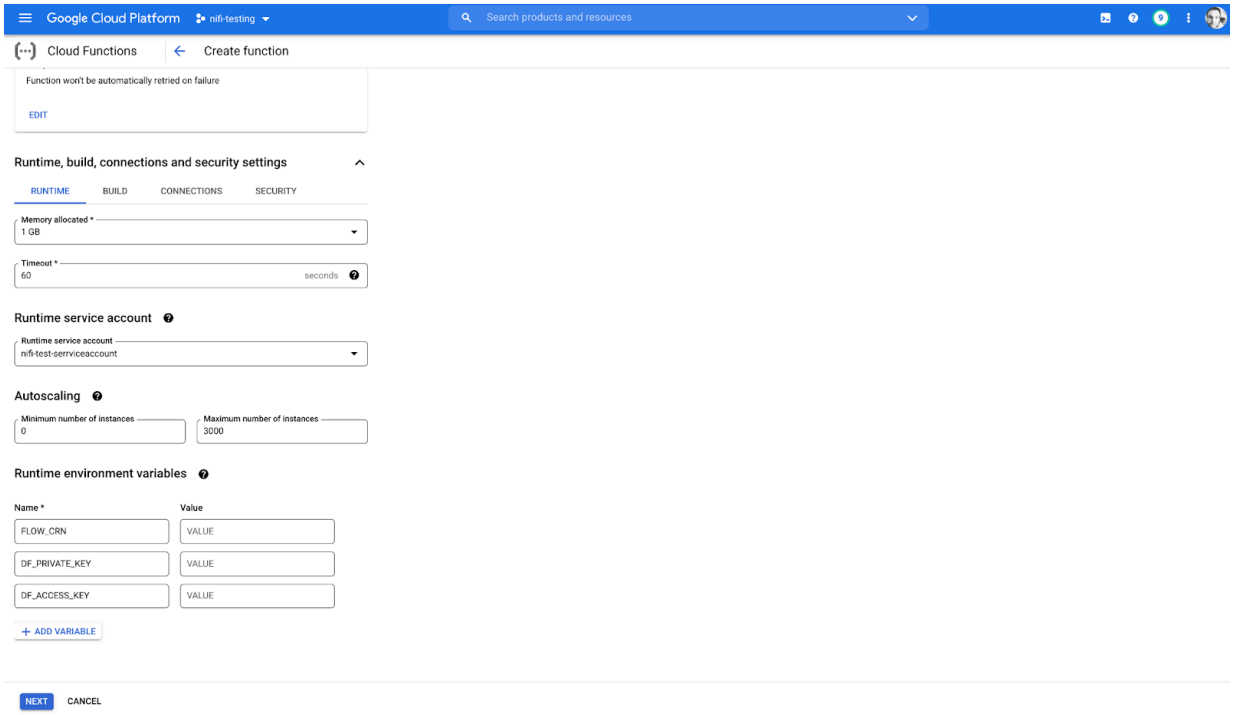

9. Click NEXT.

10. In the Code section, select Java 11 for the Runtime.

    If it is not already enabled, you have to enable the Cloud Build API.

11. In the Source code drop-down menu, select ZIP Upload and upload the ZIP file that was provided to you by Cloudera.

    Provide a Google Cloud Storage Stage bucket where the code will be staged during the deployment of the function. Alternatively, you may choose ZIP from Cloud Storage and provide a location to the ZIP file.

12. Set the Entry point to:

For HTTP triggers:

com.cloudera.naaf.gcp.cloud.functions.StatelessNiFiHttpFunction

For background triggers with asynchronous calls:

com.cloudera.naaf.gcp.cloud.functions.StatelessNiFiBackgroundFunction

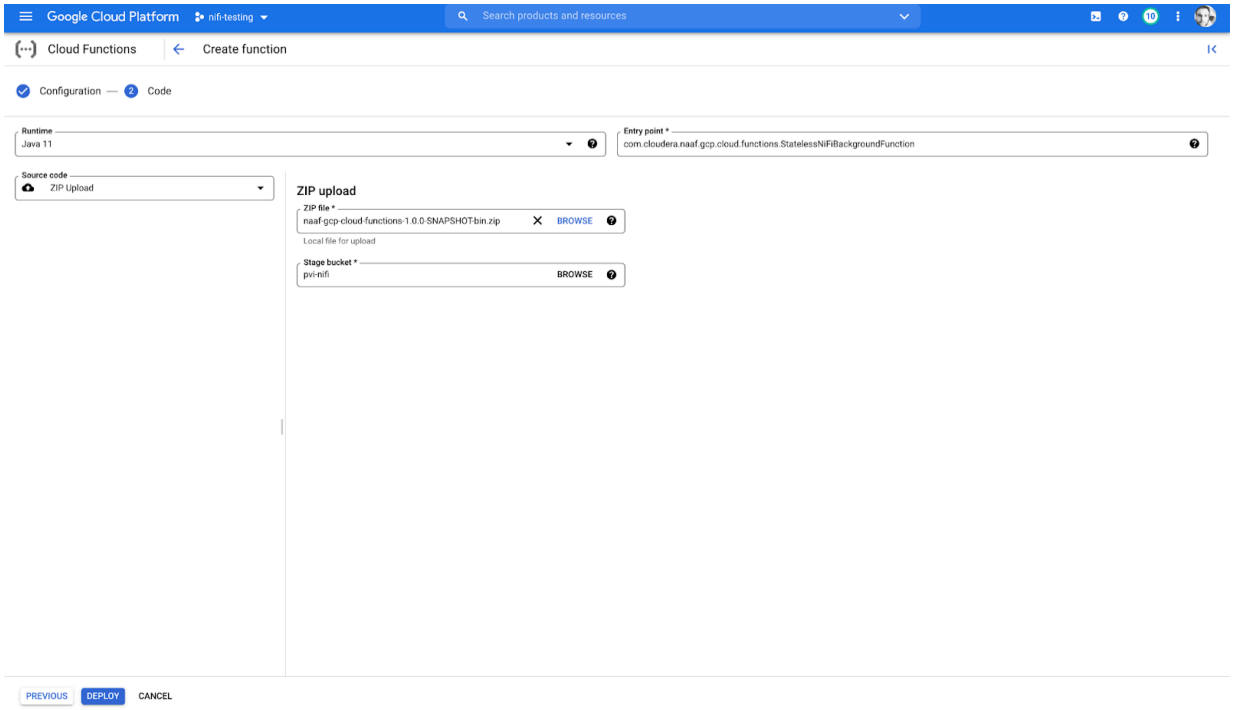

13. Click DEPLOY.

Once your function is deployed, it looks similar to this:

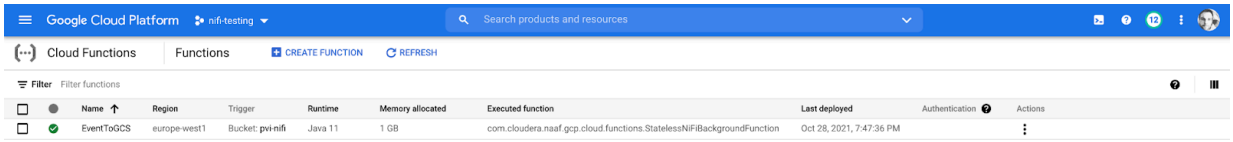

Related Information

Memory and timeout in runtime configuration

Service account

Runtime environment variables

Google Cloud Secret Manager

Configuring your Google Cloud function

After you have created a function, you can use the built-in configuration options to control its behavior. You can configure additional capabilities, adjust resources associated with your function, such as memory and timeout, or you can also create and edit test events to test your function using the console.

Memory and timeout in runtime configuration

You can configure two important elements on the Runtime Configuration tab, timeout and memory allocated to the cloud function. The appropriate values for these two settings depend on the data flow to be deployed. Typically, it is recommended to start with allocating 1 GB - 2 GB memory. You can adjust this value later as you see how much memory your function needs.

Note: Any extensions or resources added to the Google Cloud Function are placed under /tmp, which is an in-memory mount. This means that they consume memory resources provisioned for the function. For more information, see *Cloud Storage*.

While Cloudera DataFlow Functions performs very well during a warm start, a cold start that must source extensions may take several seconds to initialize. As a result, it is recommended to set the timeout to at least 60 seconds. If the dataflow to run reaches out to many other services or performs complex computation, it may be necessary to use a larger value (even several minutes).

Runtime environment variables

You must configure the function to specify the data flow to run and add any necessary runtime configuration using environment variables.

Procedure

1. Navigate to the Runtime tab and choose Runtime Environment variables in the left-hand menu.
2. Click Add variable to add the first environment variable.

   You can add one or more environment variables to configure the function. The following environment variables are supported:

Variable Name	Description	Required	Default Value
FLOW_CRN	The Cloudera Resource Name (CRN) for the data flow that is to be run. The data flow must be stored in the Cloudera DataFlow Catalog. This CRN should indicate the specific version of the data flow and as such will end with a suffix like /v.1. For more information, see *Retrieving data flow CRN*.	true	--
DF_PRIVATE_KEY	The Private Key for accessing the Cloudera DataFlow service. The Private Key and Access Key are used to authenticate with the Cloudera DataFlow Service and they must provide the necessary authorizations to access the specified data flow. For more information, see *Provisioning Access Key ID and Private Key*.	true	--
DF_ACCESS_KEY	The Access Key for accessing the Cloudera DataFlow service. The Private Key and Access Key are used to authenticate with the Cloudera DataFlow Service and they must provide the necessary authorizations to access the specified data flow. For more information, see *Provisioning Access Key ID and Private Key*.	true	--

Variable Name	Description	Required	Default Value
INPUT_PORT	The name of the Input Port to use. If the specified data flow has more than one Input Port at the root group level, this environment variable must be specified, indicating the name of the Input Port to queue up the Cloud Function event. If there is only one Input Port, this variable is not required. If it is specified, it must properly match the name of the Input Port.	false	--
OUTPUT_PORT	The name of the Output Port to retrieve the results from. If no Output Port exists, the variable does not need to be specified and no data will be returned. If at least one Output Port exists in the data flow, this variable can be used to determine the name of the Output Port whose data will be sent along as the output of the function. For more information on how the appropriate Output Port is determined, see *Output ports*.	false	--
FAILURE_PORTS	A comma-separated list of Output Ports that exist at the root group level of the data flow. If any FlowFile is sent to one of these Output Ports, the function invocation is considered a failure. For more information, see *Output ports*.	false	--
DF_SERVICE_URL	The Base URL for the Cloudera Dataflow Service.	false	https://api.us-west-1.cdp.cloudera.com/
NEXUS_URL	The Base URL for a Nexus Repository for downloading any NiFi Archives (NARs) needed for running the data flow.	false	https://repository.cloudera.com/artifactory/cloudera-repos/
WORKING_DIR	The working directory, where NAR files will be expanded.	false	/tmp/working
EXTENSIONS_DOWNLOAD_DIR	The directory to which missing extensions / NiFi Archives (NARs) will be downloaded. For more information, see *Providing Custom Extensions / NARs*.	false	/tmp/extensions
STORAGE_BUCKET	A Cloud Storage bucket in which to look for custom extensions / NiFi Archives (NARs) and resources. For more information, see *Cloud storage*.	false	--

Variable Name	Description	Required	Default Value
STORAGE_EXTENSIONS_DIRECTORY	The directory in the Cload Storage bucket to look for custom extensions / NiFi Archives (NARs). For more information, see *Cloud storage*.	false	extensions
STORAGE_RESOURCES_DIRECTORY	The directory in the Cloud Storage bucket to look for custom resources. For more information, see *Providing additional resources*.	false	resources
DISABLE_STATE_PROVIDER	If true, it disables the Firestore data flow state provider, even if the data flow has stateful processors.	false	--
FIRESTORE_STATE_COLLECTION	The Firestore collection name where the state will be stored.	false	nifi_state
KRB5_FILE	It specifies the filename of the krb5.conf file. This is necessary only if connecting to a Kerberos-protected endpoint. For more information, see *Configuring Kerberos*.	false	/etc/krb5.conf

The following environment variables apply only to HTTP triggers:

Variable Name	Description	Required	Default Value
HEADER_ATTRIBUTE_PATTERN	A Regular Expression capturing all flowfile attributes from the Output Port that should be returned as HTTP headers, if the trigger type is HTTP. For all other trigger types, this variable is ignored.	false	--
HTTP_STATUS_CODE_ATTRIBUTE	A flowfile attribute name that will set the response HTTP status code in a successful data flow, if the trigger type is HTTP. For all other trigger types, this variable is ignored. See the *HTTP trigger* section of *Google Cloud Function triggers* for more information.	false	--

Related Information

Cloud storage
Retrieving data flow CRN
Provisioning Access Key ID and Private Key
Output ports
Providing custom extensions / NARs
Providing additional resources
Configuring Kerberos
HTTP trigger

Service account

Cloud functions require a service account to use during runtime, granting the function permissions to use other Google Cloud Platform (GCP) services. There are some permissions commonly required for Cloudera DataFlow Functions.

Creating/adding service account

When configuring your Cloud function:

1. Locate the Runtime service account section of the Runtime, build, connections and security settings.
2. If you already have a service account for this purpose, you may select it here. Otherwise, select Create new service account.
3. Specify the account name and ID.
4. Click Create service account.

   You will need to add permissions to the Service Account separately.

When configuring Secrets accessible to the Cloud function, the console automatically prompts you to grant the necessary permissions to the Runtime Service Account. For more information on this, see the *Google Cloud Secret Manager* section in *Parameters*.

Adding additional role to service account

When providing custom extensions or resources from a Cloud Storage bucket, you need to add an additional role to the Service Account:

1. Navigate to the IAM & Admin GCP service, and select Roles on the left sidebar.
2. Click Create role.
3. Name the role.
4. Click Add permissions.
5. In the Filter section, add storage.buckets.get for Enter property name or value.
6. Click Add.
7. Do the same for the storage.buckets.list permission.
8. Click Create.
9. Select IAM from the left sidebar.
10. Locate your Cloud function's Runtime Service Account, and click the Edit pencil icon on the far right.
11. In the Role drop-down menu, select Custom your newly created Role on the right side .
12. Click Save.

For additional information, see *Cloud Storage*.

Your Service Account now allows your Cloud function to list and get objects from buckets.

Granting service account access to specific bucket

You need to grant your Service Account access to the specific bucket configured in the *CLOUD_STORAGE_BUCKET* environment variable:

1. Navigate to the Cloud Storage GCP service, and click your bucket name.
2. Select the Permissions tab under the bucket name.
3. In the Permissions section, click Add.
4. Provide the fully qualified name of the Runtime Service Account.

   For example: naaf@projectname.iam.gserviceaccount.com
5. In Select a role, select Cloud Storage on the left, and Storage Object Viewer on the right.

6. Click Save.

Related Information

Google Cloud Secret Manager

Cloud storage

Output ports

Google Cloud functions may or may not allow the function to return a result.

The only Google Cloud Function trigger type that can directly return a result is an HTTP trigger. In this case, the output of the data flow, is written as the HTTP response to the trigger, as described in the *HTTP trigger* section of *Google Cloud Function triggers* .

Related Information

HTTP trigger

Parameters

The concept of parameterization is an important step in building a dataflow that you can run outside of the NiFi instance where it was built. NiFi allows you to define Processor and Controller Service properties at runtime instead of at build time by using Parameter Contexts.

Environment variables

Any parameter can be specified using the environment variables of the Google Cloud Function. When configuring the function, in the Runtime environment variables section of the Runtime, build, connections and security settings, add an environment variable with a name that matches the name of a parameter in your Parameter Context.

Google Cloud Secret Manager

A more secure mechanism for storing parameters is to use the Google Cloud Secret Manager, which is recommended for the sensitive properties of a data flow.

1. Navigate to the Secret Manager GCP service and click Create secret at the top-left.
2. Name the secret descriptively (this does not have to be the same name as a data flow Parameter), and enter the Secret value (or upload a file, which is most useful for mounted secrets).
3. Click Create secret at the bottom.
4. Repeat the above steps for all needed secrets.
5. In the Runtime, build, connections and security settings section of Cloud function configuration, select the Security tab.
6. Click Reference a secret.

   This allows you to select an existing Secret. The console will prompt you to grant permissions to use the secret to the Runtime Service Account, which you should do.
7. If your secret is an uploaded file, select Mounted as volume for the Reference method.
8. For parameters, select Exposed as environment variable as the Reference method. Here you should enter the actual data flow Parameter Context name in the environment variable name, since this is how parameters are mapped.
9. Optionally, select latest from the Version drop-down menu.

   This allows the secret value to be automatically updated if the function is redeployed.

Important:

To use this capability, the Parameter Contexts used in the flow must have names that are compatible with Google Cloud Secret Manager names and Environment Variable names. To ensure that these naming conventions are followed, Parameter Contexts and Parameters should use names that start with a letter and consist only of letters, numbers, and the underscore (_) character.

For example, instead of naming a Parameter Context GCP Parameter Context, use the name GCP_PARAMETER_CONTEXT when building the data flow.

Cold start

Cold start can be defined as the set-up time required to get a serverless application's environment up and running when it is invoked for the first time within a defined period.

What is a cold start?

The concept of cold start is very important when using Function as a Service solutions, especially if you are developing a customer-facing application that needs to operate in real time. A cold start is the first request that a new function instance handles. It happens because your function is not yet running and the cloud provider needs to provision resources and deploy your code before the processing can begin. This usually means that the processing is going to take much longer than expected.

Before execution, the cloud provider needs to:

- Provision the container that will run the code
- Initialize the container / runtime
- Initialize your function

before the trigger information can be forwarded to the function’s handler.

For more information about the concept of cold start, see Cold starts in Cloud Functions execution environment.

How to avoid a cold start?

Cloud providers offer the option to always have at least one instance of the function running to completely remove the occurrence of a cold start. You can read more about how Cloud Functions min instances reduce cold starts in Minimum number of instances for Google Cloud Functions.

Cold start with Cloudera DataFlow Functions

When a cold starts happens with Cloudera DataFlow Functions, the following steps are executed before the trigger payload gets processed:

- The function retrieves the flow definition from the Cloudera DataFlow Catalog.
- The NiFi Stateless runtime is initialized.
- The flow definition is parsed to list the required NAR files for executing the flow.
- The required NAR files are downloaded from the specified repository unless the NARs are made available in the local storage attached to the function.
- The NARs are loaded into the JVM.
- The flow is initialized and components are started.

The step that might take a significant amount of time is the download of the NAR files. In order to reduce the time required for this step as much as possible, the best option is to list the NAR files required for the flow definition and make those files available to the function.

For the related instructions, see Cloud storage on page 13.

Cloud storage

Providing custom extensions / NARs

Cloudera Maven Repository

The binary that is deployed for running the Google Cloud function does not include the full NiFi deployment with all NARs / extensions, as it would result in very long startup times and would require more expensive configurations for the Cloud function. Instead, it packages only the NARs necessary to run Stateless NiFi. If any other extension is needed to run the data flow, it is automatically downloaded from the Cloudera Maven Repository by default when the Cloud function is initialized on a cold start.

Nexus repository

You can configure an alternative Nexus repository by setting the *NEXUS_URL* environment variable to the URL of the Nexus Repository where extensions should be downloaded. For example, to use Google's mirror of the Maven Central repository, set the *NEXUS_URL* environment variable to: https://maven-central.storage-download.googleapis.com/maven2

Note: Any configured URL must either be accessible through http or be served over https with a trusted certificate.

Cloud Storage bucket

If there is a need to provide custom extensions, you can use a Cloud Storage bucket:

1. Create a Cloud Storage bucket and create a directory with a descriptive name, like extensions.
2. Upload all NAR files that will need to be used to this directory.
3. Add the following environment variables to tell the Cloud function where to look for the extensions:
   - *STORAGE_BUCKET* containing the name of the bucket
   - *STORAGE_EXTENSIONS_DIRECTORY* with the full directory path of the above extensions directory

     Note: If you do not specify this variable, the directory defaults to extensions.

Note: If you add a lot of extensions, it will take longer for the Cloud function to load on a cold start, and it will take more memory to load the additional files and classes. Although extensions are supported, Cloudera does not recommend uploading hundreds of MBs or more of extensions.

Providing additional resources

In addition to NiFi extensions, some data flows may also require additional resources to function. For example, a JDBC driver may be required for establishing a database connection, or a CSV file to provide data enrichment. The recommended approach for providing such resources to your Cloud function is using the Cloud Storage bucket.

1. Create a Cloud Storage bucket, if you do not already have one and create a directory with a descriptive name, like "resources".
2. Upload all resources required for the data flow to run to this directory.

   These resources will appear in the Cloud function's /tmp/resources directory to be accessed by the data flow.

Because each deployment of a given data flow may need to reference files in a different location, depending on its environment, it is generally considered a best practice to parameterize all resources that need to be accessed by the data flow. For example, if the data flow contains a DBCPConnectionPool controller service for accessing a database, it is recommended to use a Parameter for the "Database Driver Location(s)" property of the Controller Service. This allows each deployment to specify the location of the JDBC driver independently.

For example:

1. You can set the Database Driver Location(s) property to a value of #{JDBC_DRIVER_LOCATION}.
2. In the Cloud Storage, you can upload a file that contains your JDBC driver, database-jdbc-driver.jar.
3. You can add an Environment Variable named *JDBC_DRIVER_LOCATION* with a value of /tmp/resources/database-jdbc-driver.jar.

By taking this approach, your data flow is more reusable, as it can be deployed in many different environments.

Data flow state

About this task

By default, if your data flow contains any stateful processors (e.g. ListSFTP), this state is automatically stored in a Firestore collection called nifi_state. You can set this name using the *FIRESTORE_STATE_COLLECTION* environment variable. In order to integrate with Firestore, some additional steps are required. Without these steps, the data flow will still run, but the state will not be preserved.

Procedure

1. In the GCP console, navigate to the Firestore service.
2. Click Select Native Mode.
3. Select the appropriate region, and click Create database.
4. Navigate to the IAM service, and click the Edit (pencil) button on the service account principal that runs your Cloud Function.
5. Click Add Role, and add the Firebase Admin role.
6. Click Save.

   The Firestore state provider can be disabled even if your data flow contains stateful processors by setting the *DISABLE_STATE_PROVIDER* Environment Variable to true.

   Warning: Disabling the Firestore state provider will cause processor state to be lost between function executions.

Configuring Kerberos

Depending on the data flow that is used, you may need to enable Kerberos authentication to interact with some service(s).

To enable Kerberos, you must provide an appropriate krb5.conf file. This file tells the Cloudera DataFlow function where the Key Distribution Center (KDC) is located and which Kerberos Realm to use.

To specify this, make the krb5.conf file available to Cloudera DataFlow Function. You can do it by using a GCS bucket. See the corresponding section to learn more about this.

Additionally, you must tell Cloudera DataFlow Function where it can find this krb5.conf file. To do this, add an environment variable to your Lambda function. The environment variable must have the name KRB5_FILE and the value must be the fully qualified file name of the krb5.conf file. For example, it might use /tmp/resources/krb5.conf.

In order to use Keytabs in your data flow, it is also important to provide the necessary keytab files to the Cloudera DataFlow Function. You can accomplish it by using a GCS bucket. While the krb5.conf file is a system configuration, the keytab is not. The keytab is configured in the data flow on a per-component basis. For example, one Processor may use keytab nifi-keytab-1 while some Controller Service makes use of keytab hive-keytab-4. The location of these files may change from deployment to deployment. For example, for an on-premises deployment, keytabs may be stored in /etc/security/keytabs while in Cloudera DataFlow Function, they may be made available in /tmp/resources.

So it is better to use Parameter Contexts to parameterize the location of the keytab files. This is a good practice for any file-based resource that needs to be made available to a deployment. In this case, because there is a parameter referencing the location, you need to provide Cloudera DataFlow Function with a value for that parameter. For more information, see *Parameters*.

Related Information

Parameters

Handling failures

You can use a failure Output Port to notify the Cloud function that an invocation has failed. At that point, the Cloud function may automatically retry the invocation, if configured in the Trigger section, for up to seven days. After that, if it still fails, the Cloud function will simply drop the event notification. This could result in data loss, as it means that your Cloud function will not have a chance to process the data again.

Testing your Google Cloud function

Once you have built and verified your dataflow and uploaded it to the Cloudera DataFlow Catalog, you can create your Cloud function.

After deploying the function, you should test that you have configured all of the settings correctly.

1. Use the Testing tab of the Cloud function to input the relevant JSON structure.

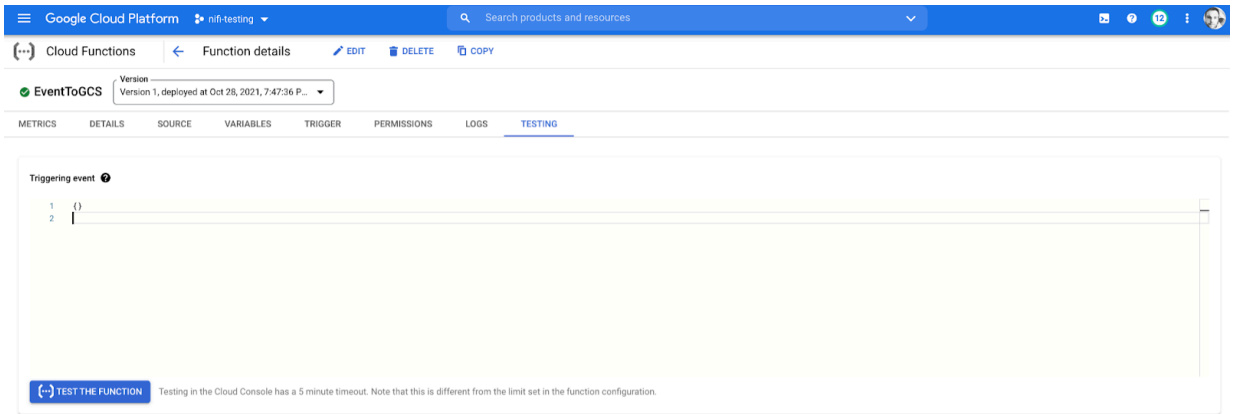

2. Click Test the function to run the data flow.

If there are problems, you can inspect the log messages to understand the problem. If the logs do not provide enough details, you may need to adjust the log levels to gain more debugging information. See the *Adjusting logs levels* for instructions.

The amount of time the cloud function takes to run depends heavily on the data flow and the number of extensions it needs. Because the cloud function may have to download some extensions from Nexus and perform initialization, a cold start may take several seconds. Even 20-30 seconds is not uncommon for a data flow with several extensions, while other data flows may complete in 10 seconds. After you run the function successfully using the Test option, it may be helpful to run several additional iterations to understand how the function will perform when a cold start is not necessary. This depends heavily on the configured data flow and other services that it may interact with. A simple data flow may complete in as a short time as 10 milliseconds, while a data flow that must perform complex transformations and reach out to one or more web services or databases for enrichment may take several seconds to complete.

It is important to adjust the Cloud function's timeout configuration in the Runtime tab of the Runtime build, connections and security settings, if the cold start takes longer than the amount of time allocated.

Related Information

Adjusting log levels

Monitoring and logs

For each invocation of your Google Cloud Function, you have access to information about the resources provisioned and used during the function's execution.

Note: With Cloudera DataFlow Functions, a cold start may take a few seconds.

This view is useful in case you want to adjust the amount of memory provisioned with your function and optimize the cost charged by Google Cloud:

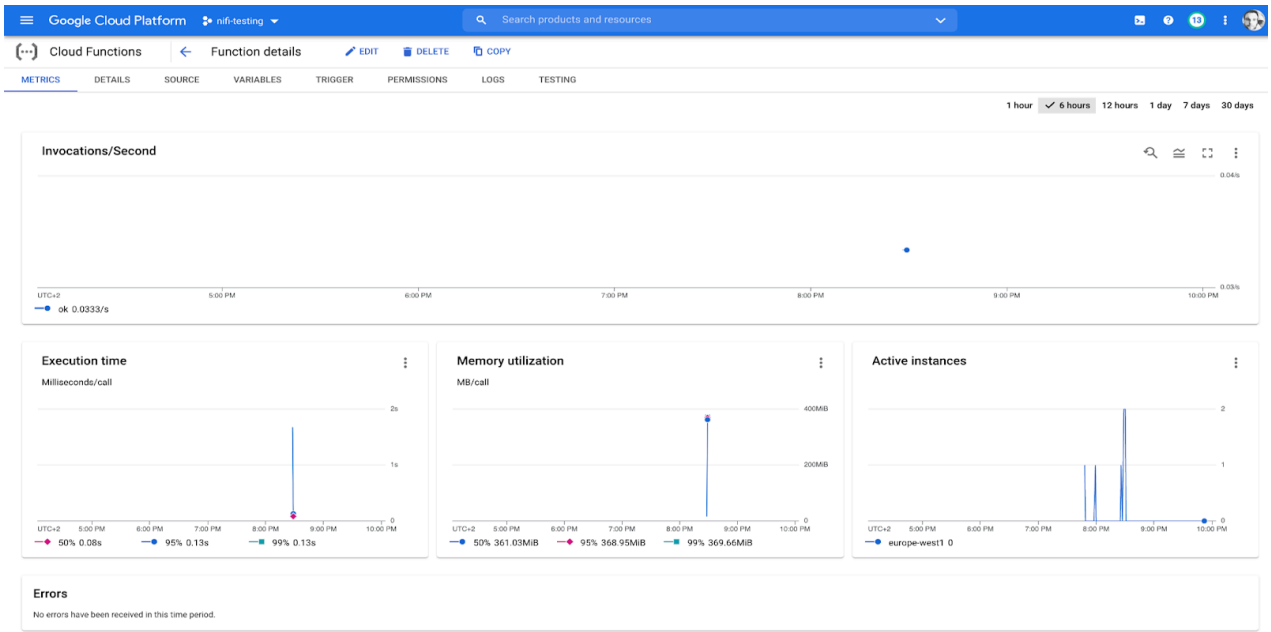

From this view, you can also access the logs of the function's executions:

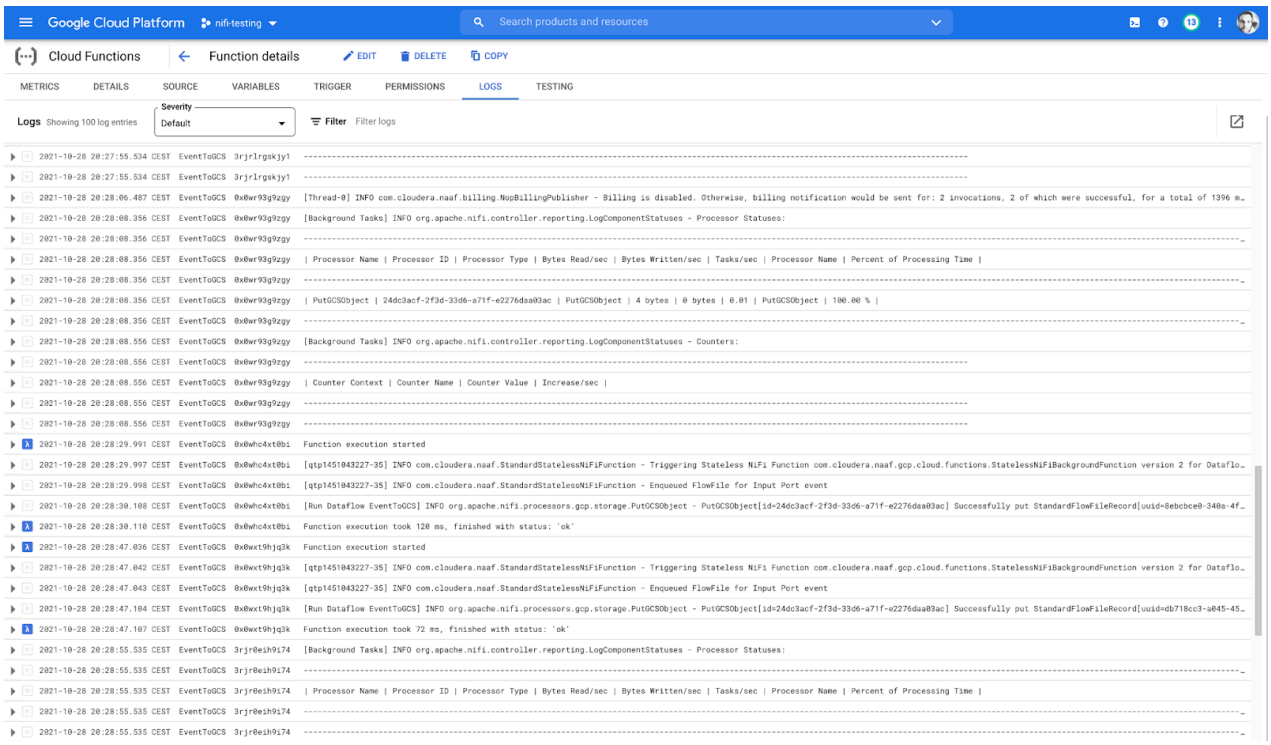

You can enable additional monitoring capabilities in Google Cloud.

Adjusting logs levels

Cloudera DataFlow Functions makes use of the SLF4J Simple Logger for logging purposes. You can update the log levels by adjusting the JVM's system properties.

To do so, you need to set the *JAVA_TOOL_OPTIONS* environment variable.

- To set a log level for a given logger, you can set the JVM system property named org.slf4j.simpleLogger.log.< logger name> to the desired log level.

  For example, if you want to enable DEBUG logging for the StandardStatelessNiFiFunction to view the actual data flow JSON being used, you would set the *JAVA_TOOL_OPTIONS* environment variable to -Dorg.slf4j.simpleLogger.log.com.cloudera.naaf.StandardStatelessNiFiFunction=DEBUG
- You can also set additional logger levels by adding multiple -D options separated by spaces. For example, to enable DEBUG logs on both the Cloudera DataFlow Functions framework and the Stateless Bootstrap class, you would set *JAVA_TOOL_OPTIONS* to a value of:

```
-Dorg.slf4j.simpleLogger.log.com.cloudera.naaf.StandardStatelessNiFiFunc
tion=DEBUG
           -Dorg.slf4j.simpleLogger.log.org.apache.nifi.stateless.bootstra
p.StatelessBootstrap=DEBUG
```

Google Cloud Function triggers

All Google Cloud Function triggers are supported with Cloudera DataFlow Functions. In this section you can review the most commonly used triggers and examples of the FlowFile’s content that would be generated following a triggering event.

Background triggers

The Output Port data flow logic is simple with background functions.

If one Output Port is present, and its name is "failure", any data routed to that Output Port will cause the function invocation to fail.

If two Output Ports are present, and one of them is named "failure", that Output Port will behave as the "failure" port above. Any other Output Port is considered a successful invocation, but the function will do nothing with data routed here.

Any Port with the name "failure" (ignoring case) is considered a Failure Port. Additionally, if the *FAILURE_PORTS* environment variable is specified, any port whose name is in the comma-separated list will be considered a Failure Port.

Google Cloud Storage

You can use this trigger, for example, to start your function whenever a file is landing into Google Cloud Storage:

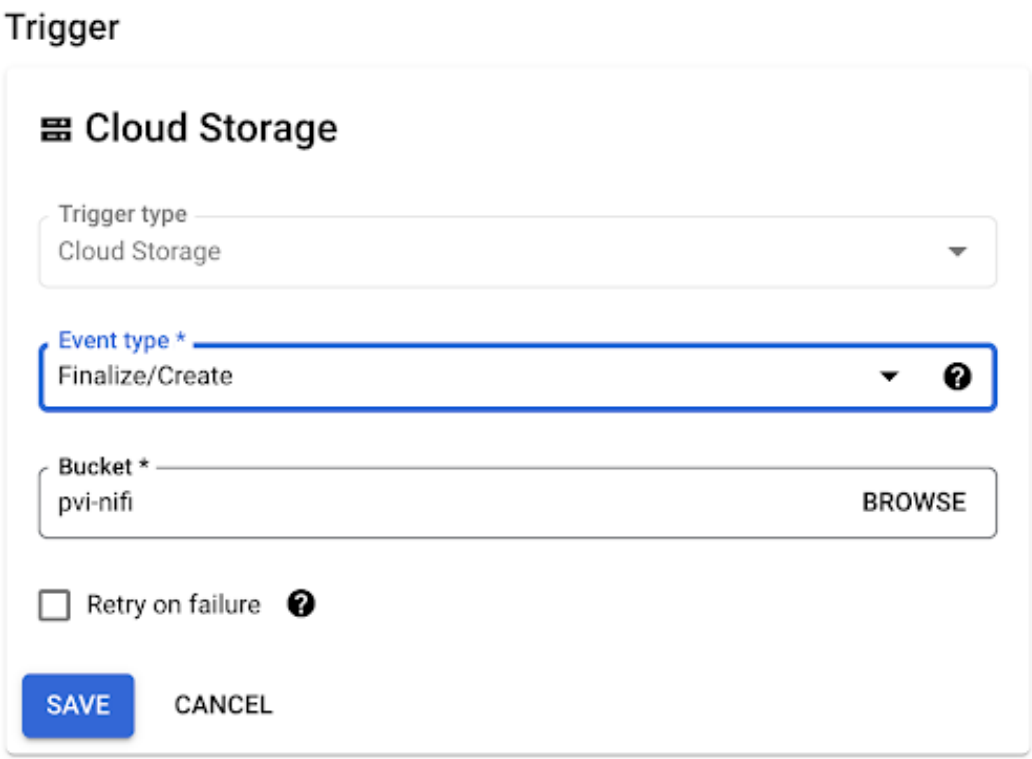

The FlowFile's content would look like this:


```
{
  "bucket": "pvi-nifi",
  "contentType": "text/plain",
  "crc32c": "LnPMaQ==",
  "etag": "CPH+l4ze7fMCEAE=",
  "generation": "1635445725593457",
  "id": "pvi-nifi/dummyfile.txt/1635445725593457",
  "kind": "storage#object",
  "md5Hash": "hKh3F0pE5OJl6kqON7D00A==",
  "mediaLink": "https://www.googleapis.com/download/storage/v1/b/pvi-nifi/o/dummyfile.txt?generation=1635445725593457&alt=media",
  "metageneration": "1",
  "name": "dummyfile.txt",
  "selfLink": "https://www.googleapis.com/storage/v1/b/pvi-nifi/o/dummyfile.txt",
  "size": "7042",
  "storageClass": "STANDARD",
  "timeCreated": "2021-10-28T18:28:45.602Z",
  "timeStorageClassUpdated": "2021-10-28T18:28:45.602Z",
  "updated": "2021-10-28T18:28:45.602Z"
}
```

Google Cloud Pub/Sub

You can use this trigger for processing events received in a Pub/Sub topic:

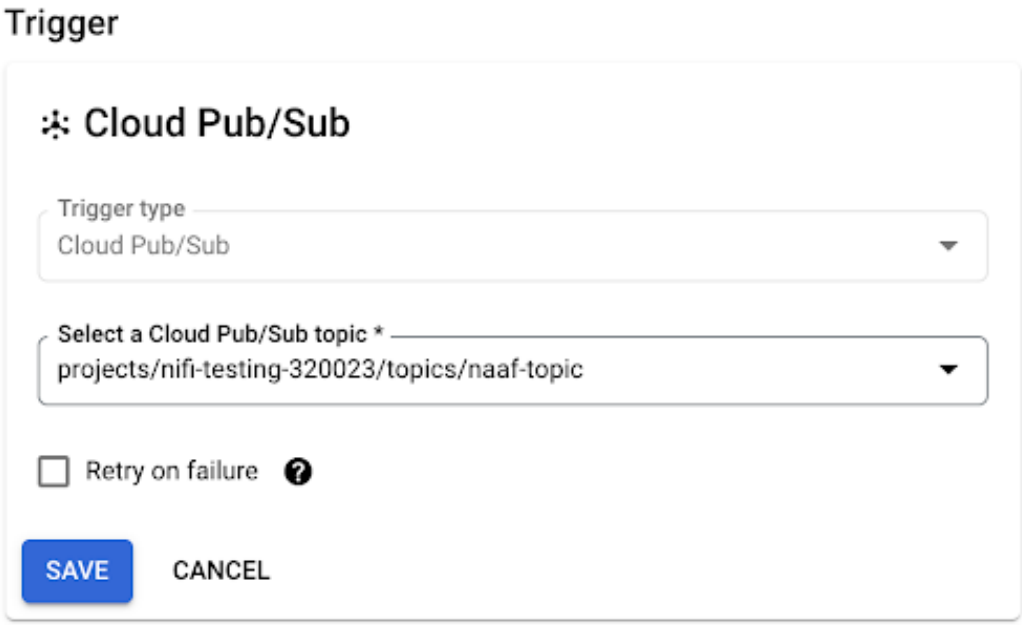

The FlowFile's content would look like this:

```
{
  "@type": "type.googleapis.com/google.pubsub.v1.PubsubMessage",
  "attributes": {
    "myAttributeKey": "myAttributeValue"
  },
  "data": "eyJteU1lc3NhZ2UiOiJUaGlzIGlzIG15IG1lc3NhZ2UgcGF5bG9hZCEifQ\u003d\u003d"
}
```

Note:

It is not possible to batch messages with this trigger. The function is executed once for each message in the Pub/Sub topic.

The payload of the message is Base 64 encoded.

HTTP trigger

This trigger can be used to expose an HTTP endpoint (with or without required authentication) allowing for clients to send data to the function.

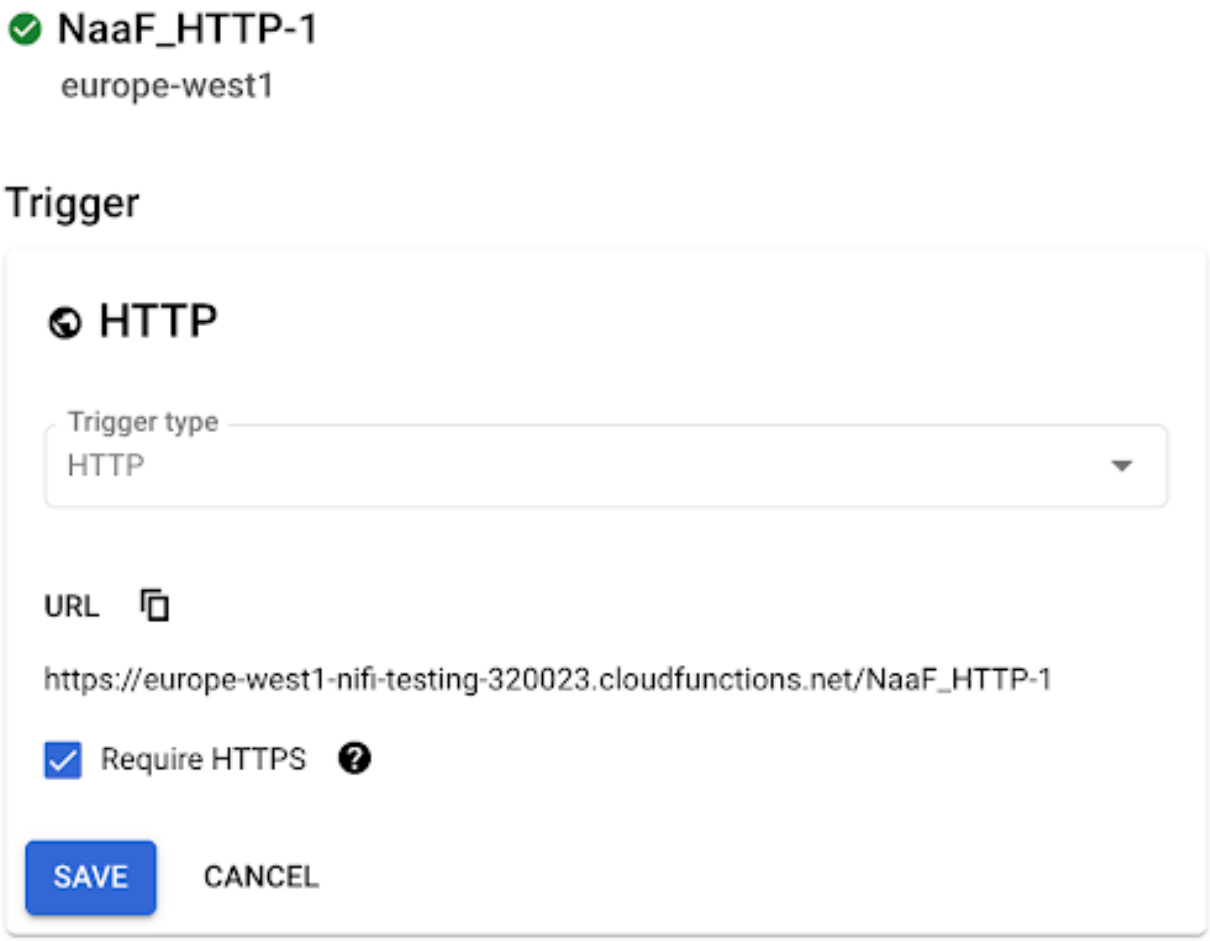

This trigger would have Google Cloud creating and exposing an endpoint such as: https://europe-west1-nifi-testing-320023.cloudfunctions.net/NaaF_HTTP-1

The FlowFile created from the client call will be as follow:

- The content of the FlowFile is the exact body of the client request
- Attributes of the FlowFile are created for each HTTP headers of the client request. Each attribute is prefixed: gcp.http.header.<header key> = <header value>

Since Cloud Functions with an HTTP trigger are invoked synchronously, they directly return a response.

The Output Port semantics are the following.

If no Output Port is present in the data flow's root group, no output will be provided from the function invocation, but a default success message (DataFlow completed successfully) is returned in the HTTP response with a status code of 200.

If one Output Port is present, and its name is "failure", any data routed to that Output Port will cause the function invocation to fail. No output is provided as the output of the function invocation, and an error message (DataFlow completed successfully: + error message) is returned from the HTTP response, with a status code of 500.

If two Output Ports are present, and one of them is named "failure", that Output Port will behave as the "failure" port above. The other Output Port is considered a successful invocation. When the data flow is invoked, if a single FlowFile is routed to this Port, the contents of the FlowFile are emitted as the output of the function. If more than one FlowFile is sent to the Output Port, the data flow is considered a failure, as GCP requires that a single entity be provided as its output. A MergeContent or MergeRecord Processor may be used in order to assemble multiple FlowFiles into a single output FlowFile if necessary.

Any Port with the name "failure" (ignoring case) will be considered a Failure Port. Additionally, if the *FAILURE_PORTS* environment variable is specified, any port whose name is in the comma-separated list will be considered a Failure Port.

If two or more Output Ports, other than failure ports, are present, the *OUTPUT_PORT* environment variable must be provided. In such a case, this environment variable must match the name of an Output Port at the data flow's root group (case is NOT ignored). Anything that is routed to the specified port is considered the output of the function invocation. Anything routed to any other port is considered a failure.

Note: In any successful response, the mime.type flowfile attribute is used as the Content-Type of the response.

Additionally, there are two environment variables that can be configured to customize the HTTP response:

- *HEADER_ATTRIBUTE_PATTERN* can provide a regular expression matching any flowfile attributes to be included as HTTP response headers. This means that any attributes to be included as headers should be named as the headers themselves.
- *HTTP_STATUS_CODE_ATTRIBUTE* specifies a flowfile attribute that sets the HTTP status code in the response. This allows a code other than 200 for 'success' or 500 for 'failure' to be returned. If this environment variable is specified but the attribute is not set, the status code in the HTTP response will default to 200.

In the "failure" scenario, the *HTTP_STATUS_CODE_ATTRIBUTE* is not used. So, if a non-500 status code is required, it must be provided in a successful data flow path. For example, if a 409 Conflict is desired in a particular case, you can achieve it as follows:

1. In the data flow, include an UpdateAttribute processor that sets an arbitrary attribute, for example http.status.code, to the value 409.
2. Optionally, add a ReplaceText processor that sets the contents of the flowfile using Always Replace to whatever HTTP response body is desired.
3. Route this part of the flow to the main Output Port used for the success case.
4. In the GCP Cloud function, set the Runtime environment variable *HTTP_STATUS_CODE_ATTRIBUTE* to http .status.code, and deploy it.

You may also use this same method to set a non-200 success code.

Creating a Google Cloud function using CLI

Google Cloud Functions offers two product versions: Cloud Functions 1st Generation, the original version and Google Cloud Functions 2nd Generation, a new version built on Cloud Run and Eventarc to provide an enhanced feature set. See the below examples on how to deploy a Cloudera DataFlow Function in both versions of Google Cloud Functions, using the Google Cloud CLI.

Before you begin

- Your NiFi flow definition is stored in the Cloudera DataFlow Catalog and you have the CRN corresponding to the flow version you want to execute as a function
- You have created a Machine User with the proper role and you have its Access Key and Private Key credentials
- You have installed and configured the Google Cloud CLI on your local machine

1st generation (1st gen)

HTTP Trigger

To deploy a Google Cloud Function with an unauthenticated HTTP trigger, you can use the following example:

```
gcloud functions deploy myfunctionname \
 --region=europe-west1 \
 --runtime=java11 \
 --source=gs://my-bucket/naaf-gcp-cloud-functions-1.0.0-bin.zip \
 --memory=1024MB \
```

```
 --timeout=300 \
 --trigger-http \
 --allow-unauthenticated \
 --entry-point=com.cloudera.naaf.gcp.cloud.functions.StatelessNiFiHttpFunct
ion \
 --update-labels=mylabelkey=mylabelvalue \
--set-env-vars=FLOW_CRN=crn:cdp:df:us-west-1:00000000-0000-0000-0000-000000
000000:flow:my-flow/v.1,DF_PRIVATE_KEY=0000000000000000000000000000000000,D
F_ACCESS_KEY=00000000-0000-0000-0000-000000000000
```

GCS Trigger

To deploy a Google Cloud Function with a trigger on a GCS bucket, you can use the following example:

```
gcloud functions deploy myfunctionname \
 --region=europe-west1 \
 --runtime=java11 \
 --source=gs://my-bucket/naaf-gcp-cloud-functions-1.0.0-bin.zip \
 --memory=1024MB \
 --timeout=300 \
 --trigger-bucket=my-trigger-bucket
 --entry-point=com.cloudera.naaf.gcp.cloud.functions.StatelessNiFiBackground
Function \
 --update-labels=mylabelkey=mylabelvalue \
--set-env-vars=FLOW_CRN=crn:cdp:df:us-west-1:00000000-0000-0000-0000-00000
0000000:flow:my-flow/v.1,DF_PRIVATE_KEY=0000000000000000000000000000000000,
DF_ACCESS_KEY=00000000-0000-0000-0000-000000000000
```

Pub/Sub Trigger

To deploy a Google Cloud Function with a trigger on a Pub/Sub topic, you can use the following example:

```
gcloud functions deploy myfunctionname \
 --region=europe-west1 \
 --runtime=java11 \
 --source=gs://my-bucket/naaf-gcp-cloud-functions-1.0.0-bin.zip \
 --memory=1024MB \
 --timeout=300 \
 --trigger-topic=my-trigger-topic
 --entry-point=com.cloudera.naaf.gcp.cloud.functions.StatelessNiFiBackgrou
ndFunction \
 --update-labels=mylabelkey=mylabelvalue \
--set-env-vars=FLOW_CRN=crn:cdp:df:us-west-1:00000000-0000-0000-0000-0000000
00000:flow:my-flow/v.1,DF_PRIVATE_KEY=0000000000000000000000000000000000,DF
_ACCESS_KEY=00000000-0000-0000-0000-000000000000
```

2nd generation (2nd gen)

For Google Cloud Functions 2nd gen, only HTTPS triggers are supported and a Google Eventarc trigger can be added to trigger the function through HTTPS calls according to configurable events (files added to a bucket, event published to a Pub/Sub topic, and so on).

You can use the following example:

```
gcloud functions deploy myfunctionname \
 --gen2 \
 --region=europe-west1 \
 --runtime=java11 \
 --source=gs://my-bucket/naaf-gcp-cloud-functions-1.0.0-bin.zip \
 --memory=1024MB \
 --timeout=300 \
 --trigger-http \
```

```
 --entry-point=com.cloudera.naaf.gcp.cloud.functions.StatelessNiFiHttpFuncti
on \
 --update-labels=mylabelkey=mylabelvalue \
--set-env-vars=FLOW_CRN=crn:cdp:df:us-west-1:00000000-0000-0000-0000-0000000
00000:flow:my-flow/v.1,DF_PRIVATE_KEY=0000000000000000000000000000000000000,DF
_ACCESS_KEY=00000000-0000-0000-0000-000000000000
```

It is then possible to create an Eventarc trigger and link it to the deployed function. This is an example with Pub/Sub:

```
gcloud eventarc triggers create pubsub \
--location=europe-west1 \
--service-account=0000000000-compute@developer.gserviceaccount.com \
--transport-topic=projects/my-project/topics/my-topic \
--destination-run-service=myfunctionname \
--destination-run-region=europe-west1 \
--destination-run-path="/" \
--event-filters="type=google.cloud.pubsub.topic.v1.messagePublished"
```

Cloudera DataFlow

Importing Flow Definitions

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Importing flow definitions from Apache NiFi.....4
Importing using the webUI.....4
Importing using the CLI.....5

Importing a new version of an existing flow definition.....6

Importing flow definitions from Apache NiFi

If you want to use a NiFi flow in Cloudera DataFlow, you must import it as a flow definition. When imported, the flow definition is added to the flow Catalog.

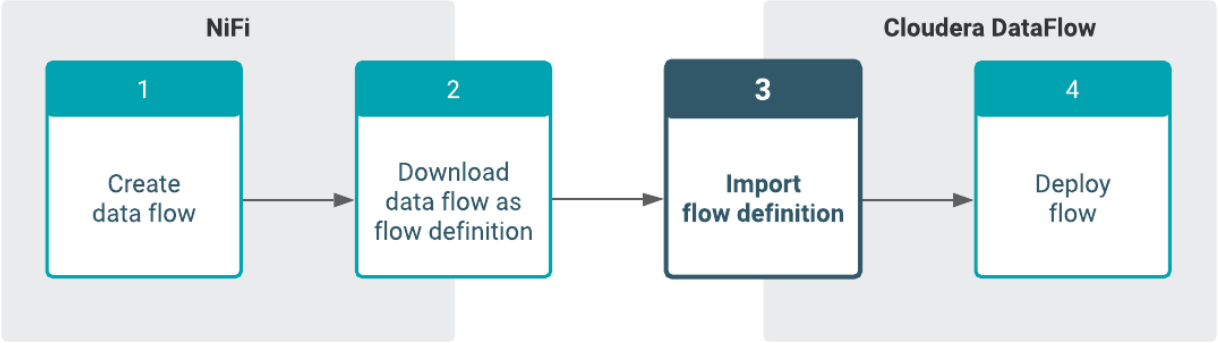

Note: For information on publishing a draft from Flow Designer, see *Managing Drafts*.

Related Information

Creating a Flow Definition in NiFi
Managing Flow Definitions
Deploying Flows
Cloudera DataFlow Authorization
Managing drafts

Importing using the webUI

Learn about importing a flow definition developed in Apache NiFi using the Cloudera DataFlow webUI.

Before you begin

- Create a data flow in Apache NiFi and download it as a flow definition JSON file. For more information, see *Creating a Flow Definition*.
- Make sure that you have DFCatalogAdmin permission to perform this task. For information on account and resource roles, see *Cloudera DataFlow Authorization*.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera Public Cloud sidebar.
2. Click Catalog from the left navigation pane.

   The Flow Catalog page is displayed. Previously imported flow definitions are displayed, one definition per row.

3. Click Import Flow Definition.

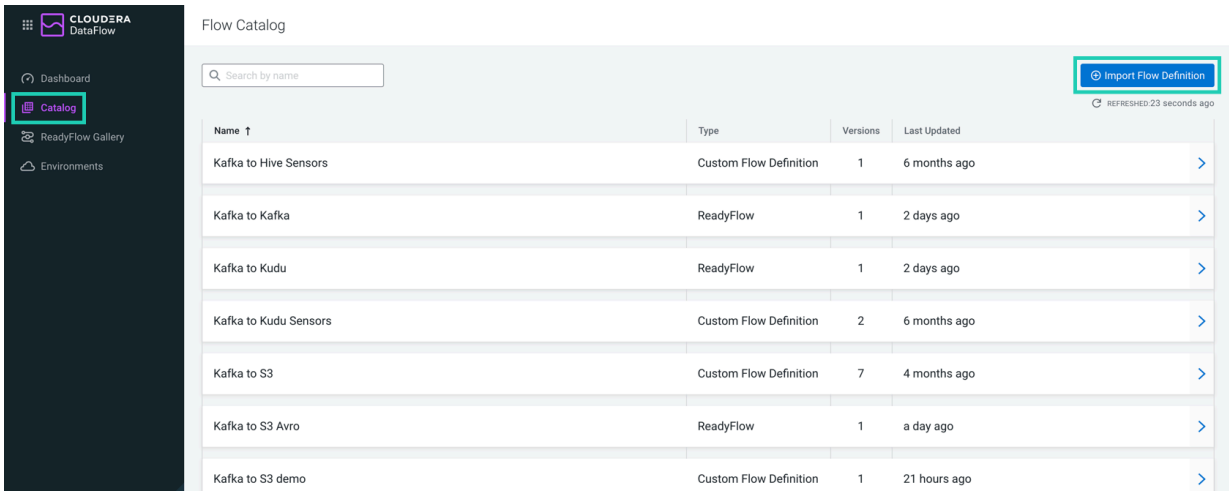

4. Provide a name (Flow Name) and a description (Flow Description) for the flow definition.

   Note: Flow name and flow description cannot be changed later.

5. In the NiFi Flow Configuration field, select the JSON file you downloaded from NiFi.

   Alternatively, you can drop the JSON file into this field to select it.

6. You can add comments to the flow definition version you are importing (Version Comments).

   Note: When you are importing the first version of a flow definition, the Version Comments field contains Initial Version by default. You can change this comment if needed.

   Note: Version comments cannot be changed later.

7. You can add custom tags to the flow definition version you are importing (Custom Tag).

   Note: Custom tags can also be added later, in the Catalog. You cannot modify custom tags, but they can be deleted. They are a good option to introduce dynamic information to your flow definition versions (for example, 'preview' or 'obsolete').

8. Click Import.

Results

You have successfully imported your NiFi flow to Cloudera DataFlow. It is available in the Catalog as the first version of your flow definition.

What to do next

Deploy the flow definition to an environment. For more information, see *Deploying flows*.

Importing using the CLI

Learn about importing a flow definition developed in Apache NiFi using the CDP CLI.

Before you begin

- Create a data flow in Apache NiFi and download it as a flow definition JSON file. For more information, see *Creating a Flow Definition*.

- Make sure that you have DFCatalogAdmin permission to perform this task. For information on account and resource roles, see *Cloudera DataFlow Authorization*.
- You have installed the CDP CLI

Procedure

To import a flow definition, enter:

```
cdp df import-flow-definition
--file [***FILE***]
--name [***NAME***]
[--description [***DESCRIPTION***]]
[--comments [***COMMENTS***]]
```

Where:

- --file specifies the path to the flow definition file you want to import.
- --name is a distinct name for your flow definition.
- --description is a short description you can use to help identify the flow definition within the Cloudera DataFlow Catalog.
- --comments optionally includes any additional information for your flow definition.

Results

Successfully importing a new flow definition results in output similar to:

```
cdpcli % ./cdp.sh --profile priv df import-flow-definition --file /Users/use
r1/Downloads/test-flow.json --name EmptyRelName
{
    "crn": "crn:cdp:df:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:flow
:EmptyRelName",
    "name": "EmptyRelName",
    "versionCount": 1,
    "createdTimestamp": 1632782902156,
    "modifiedTimestamp": 1632782902156,
    "versions": [
        {
            "crn": "crn:cdp:df:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb
73d:flow:EmptyRelName/v.1",
            "bucketIdentifier": "https://s3.us-west-2.amazonaws.com/dfx-flow
-artifacts.mow-priv.mow-dev.cloudera.com/dswift/9d74eee4-1cad-45d7-b645-7ccf
9edbb73d/18b3b7aa-da62-49a9-bf23-61da8b7a36dc",
            "author": "David Swift",
            "version": 1,
            "timestamp": 1632782902170,
            "deploymentCount": 0
        }
    ]
}
```

What to do next

Deploy the flow definition to an environment. For more information, see *Deploying Flows*.

Importing a new version of an existing flow definition

You can add new versions to an existing flow definition using either the Cloudera DataFlow UI or the Cloudera CLI.

Before you begin

- You have imported at least one flow definition.

For UI

Steps

1. In the Cloudera DataFlow UI, from the left navigation pane, click Catalog.

   Flow definitions available for you to deploy are displayed, one definition per row.
2. Identify the flow definition you wish to upload a new version for, and click the row to display the flow definition details and versions.
3. From the Actions drop-down, select Import New Version.

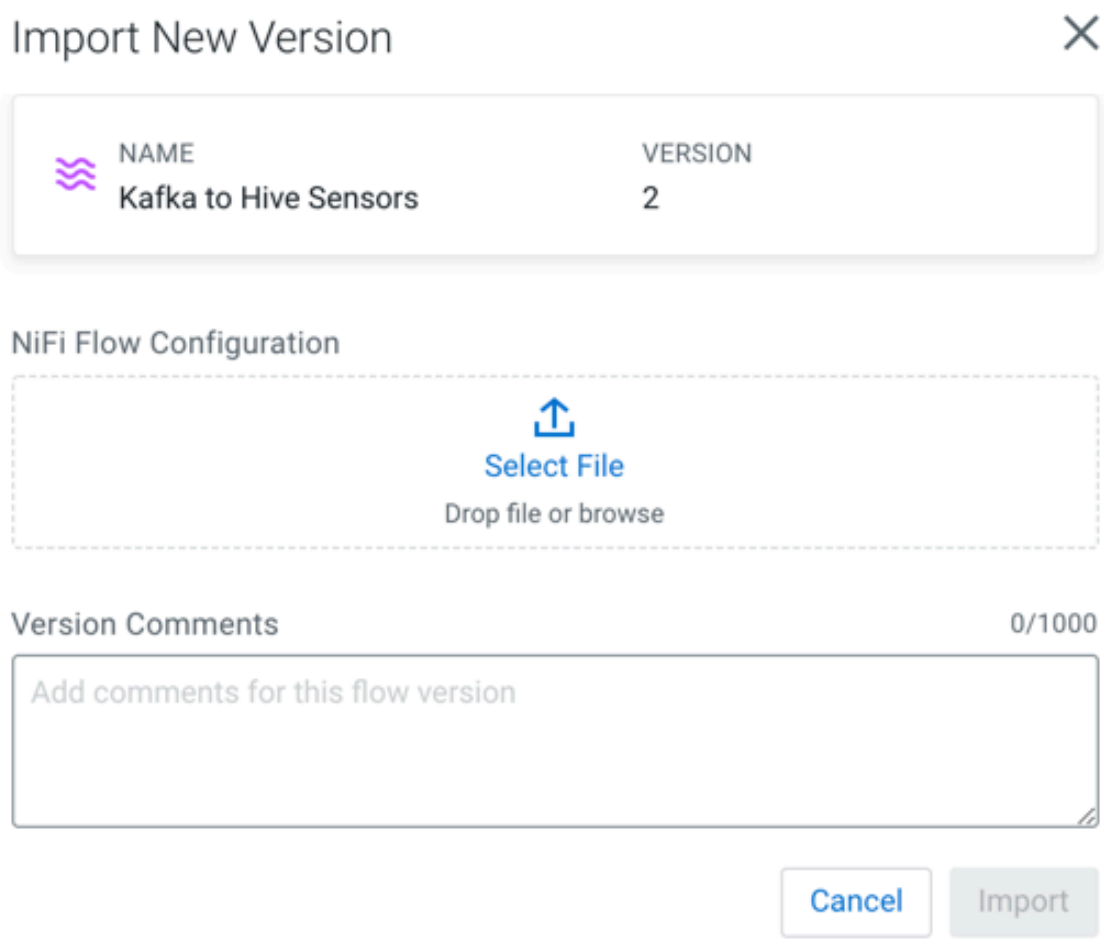

4. Select the new version file of the NiFi flow configuration, and click Import.

For CLI

Before you begin

- You have installed the CDP CLI.
- You have run cdp df list-flows to retrieve the flow-crn.

Steps

1. To import a new flow definition version, enter:

```
cdp df import-flow-definition-version
```

```
--flow-crn [***FLOW_CRN***]
--file [***FILE***]
[--comments [[***COMMENTS***]]
```

Where:

- --flow-crn specifies the flow-crn you obtained when completing the pre-requisites.
- --file specifies the path to the new flow definition version you want to import.
- --comments allows for option comments about the new version of the flow definition.

Result

Successfully importing a new flow definition version results in output similar to:

Cloudera DataFlow

Connecting to an Inbound Connection Endpoint

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Connecting applications to an endpoint.....4
TLS keys and certificates.....5

Tutorial: MiNiFi to Cloudera DataFlow flow deployment.....5

Tutorial: Invoking an HTTP endpoint with curl.....16

Using Inbound Connections with an external load balancer.....21
Configure an Application Gateway in Azure.....22

Connecting applications to an endpoint

Once a Cloudera DataFlow flow deployment with inbound connection is available, you can go on and connect an external application to start sending data.

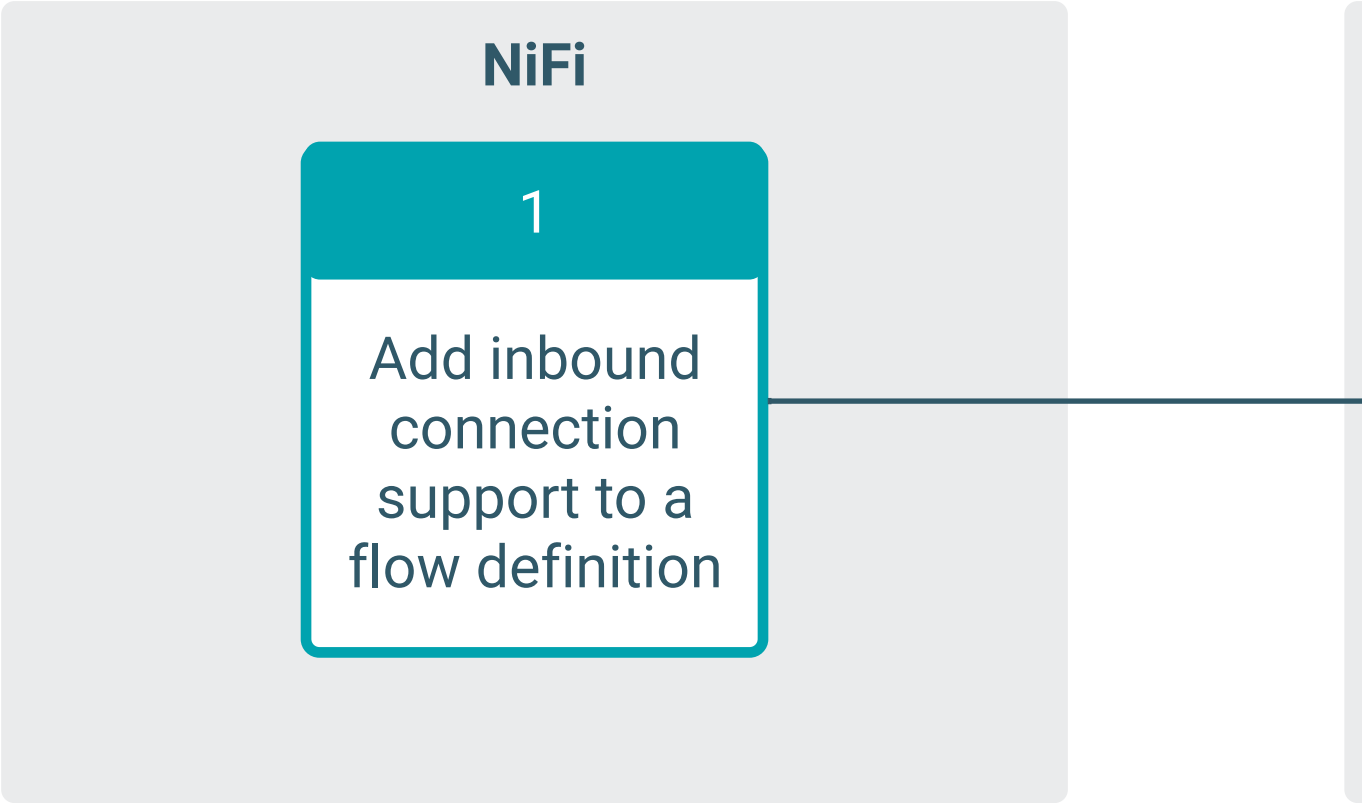

Before you begin

- A flow deployment with inbound connection is available.
- A network connection through which the client can reach the flow deployment endpoint is available.
- You have been assigned at least the DFFlowUser role for the environment to which you want to configure the inbound connection.

Procedure

1. Select the flow deployment where you want to send data and go to Deployment Manager Deployment Settings .
2. Select the NiFi Configuration tab.
3. Make a note of the Endpoint Hostname and port.
4. Click Download Client Certificate.

   The X.509 client certificate downloads to your computer in PEM format.
5. Click Download Client Private Key to obtain the RSA Private Key.

   The unencrypted RSA Private Key encoded with PKCS8 downloads to your computer in PEM format.

6. Depending on your client, you may have to convert the certificate and the private key to a different format.

   For example, to convert PEM to PKCS12 format, use the following command:

   ```
   openssl pkcs12 -export -in [***DOWNLOADED PEM CERT FILE***] -in
   key [***DOWNLOADED PEM PRIVATE KEY***] -out certificate.p12
   ```

   To further convert the PKCS12 file to JKS format for a Java client, run the following command:

   ```
   keytool -importkeystore -srckeystore [***CERTIFICATE NAME***].p12 -srcsto
   retype pkcs12 -destkeystore [***DESTINATION KEYSTORE***].jks
   ```

7. Add the certificate file and the private key files to the keystore of your application.
8. Configure your application to stream data to the Endpoint Hostname, port, and protocol of the flow deployment.

TLS keys and certificates

When using Inbound Connection Endpoints, sensitive information is sent over the network between Cloudera Data Flow (CDF) and external data sources including configuration files that contain passwords. To secure this transfer, Cloudera strongly recommends that you configure mutual Transport Layer Security (TLS) encryption.

TLS is an industry standard set of cryptographic protocols for securing communications over a network.

Configuring TLS involves creating a private key and a public key for use by server and client processes to negotiate an encrypted connection. In addition, TLS can use certificates to verify the trustworthiness of keys presented during the negotiation to prevent spoofing and mitigate other potential security issues.

Tutorial: MiNiFi to Cloudera DataFlow flow deployment

This tutorial walks you through creating an inbound connection endpoint in Cloudera DataFlow used by a flow deployment to receive data from one or more MiNiFi agents managed by Edge Flow Manager.

Before you begin

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Cloudera DataFlow Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- You have an enabled and healthy Cloudera Edge Management (CEM) environment.
- You have sufficient rights to configure the MiNiFi Agent in EFM.

Procedure

1. In a development NiFi environment, create a Controller Service of type StandardRestrictedSSLContextService at the root canvas level and name it Inbound SSL Context Service.

   a. In the Operate palette click Configuration Controller Services Create a new controller service
   b. Filter for ssl, select StandardRestrictedSSLContextService then click Add.
   c. Click Configure.
   d. On the **Settings** tab change the Name to Inbound SSL Context Service, then click Apply.

   You do not need to make further configuration on this Controller Service; it acts as a placeholder and will be created with a managed SSL Context when deployed by Cloudera DataFlow.

   Tip:

   For testing the dataflow during development, the SSL Context may be configured with test keys and certificates for your development NiFi environment.

2. Create a Process Group on the root canvas to hold your flow definition and give it a name.

   This tutorial uses the name ListenHTTP Flow.

3. Enter the process group.
4. Inside the Process Group, add a listen processor.

   This tutorial uses ListenHTTP.

5. Configure the listen processor:

   Base Path

   This tutorial uses the default contentListener.

   Listening Port

   Define a value that is valid for your use case. This tutorial uses port 9000.

   SSL Context Service

   Select Inbound SSL Context Service.

   Client Authentication

   Select REQUIRED.

   Click Apply.

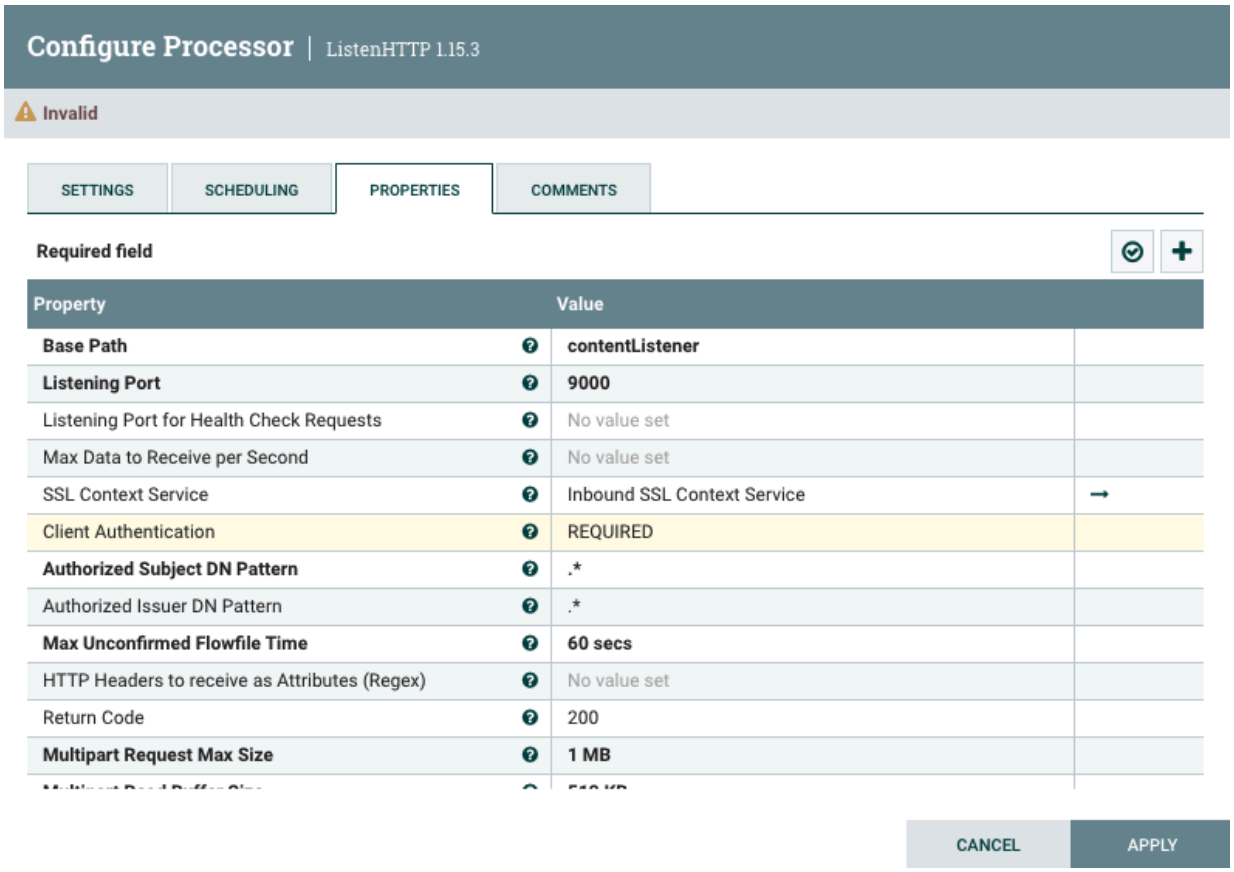

6. Connect the ListenHTTP processor to a downstream processor of your choice.

   This tutorial uses LogAttribute, where all relationships terminate.

7. From the root canvas, right click on the Process Group and select Download flow definition Without external controller services .

8. Upload the flow definition JSON to the Flow Catalog of your Cloudera DataFlow deployment.

9. Deploy the flow.

a) At the NiFi Configuration step of the Deployment wizard, select Inbound Connections Allow NiFi to Receive Data to enable inbound connections.

Accept the automatically created endpoint hostname and automatically discovered port by clicking Next.

b) At Parameters, click Next.

c) At Sizing & Scaling select the Extra Small NiFi Node Size then click Next.

d) Add a KPI on the ListenHTTP processor to monitor how many bytes it is receiving, by clicking Add new KPI.

Make the following settings:

KPI Scope

Processor

Processor Name

ListenHTTP

Metric to Track

Bytes Received

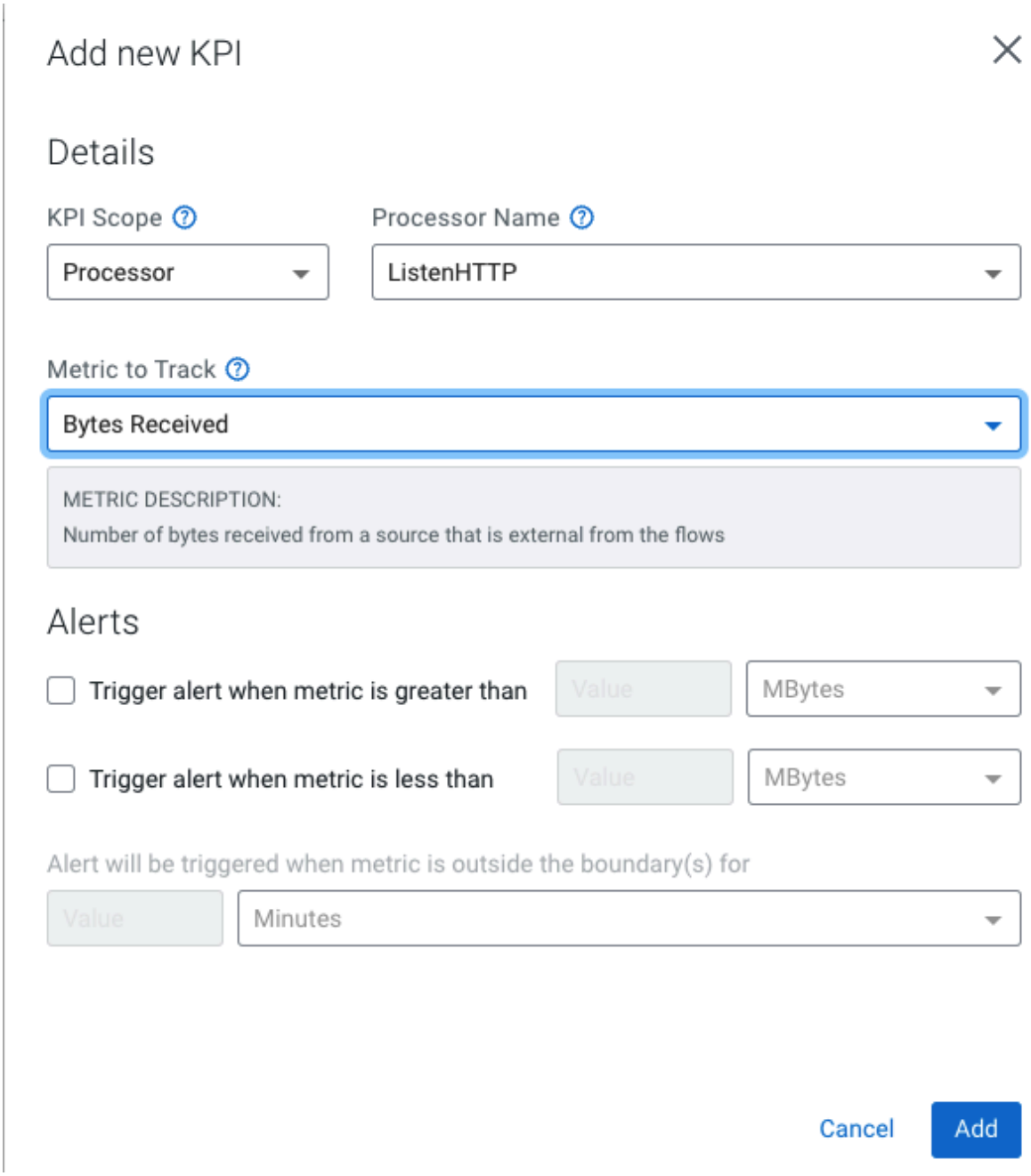

e) Review the information provided and click Deploy.

Soon after the flow deployment has started, the client certificate and private key required for sending data to the NiFi flow become available for the flow deployment that is being created.

10. Collect the information required to configure your load balancer.

a. Once the deployment has been created successfully, select it in the Deployments view and click Manage Deployment.

b. In the Deployment Settings section, navigate to the **NiFi Configuration** tab to find information about the associated inbound connection endpoint.

c. Copy the endpoint hostname and port and download the certificate and private key.

11. Start designing your MiNiFi flow in EFM.

For C++

To design a flow for your MiNiFi C++ agent class:

a. Copy the downloaded client-private-key-encoded key and client-certificate-encoded.cer certificate files to the host with the running MiNiFi C++ agent, so they are accessible by filepath from the agent.

b. Create a Service of type SSL Context Service with the following configuration:

Service Name

Specify a name for this service. This tutorial uses Client SSL Context Service.

CA Certificate

Leave it empty. As Cloudera DataFlow uses Let's Encrypt as a Certificate Authority, the certificate will be accepted automatically, without additional configuration.

Client Certificate

*[***/PATH/TO/***]*client-certificate-encoded.cer

For example, /opt/minifi/minifi-test/client-certs/client-certificate-encoded.cer.

Passphrase

Set no value.

Private Key

*[***PATH/TO/***]*client-private-key-encoded

For example, /opt/minifi/minifi-test/client-certs/client-private-key-encoded

Use System Cert Store

Keep the default False value.

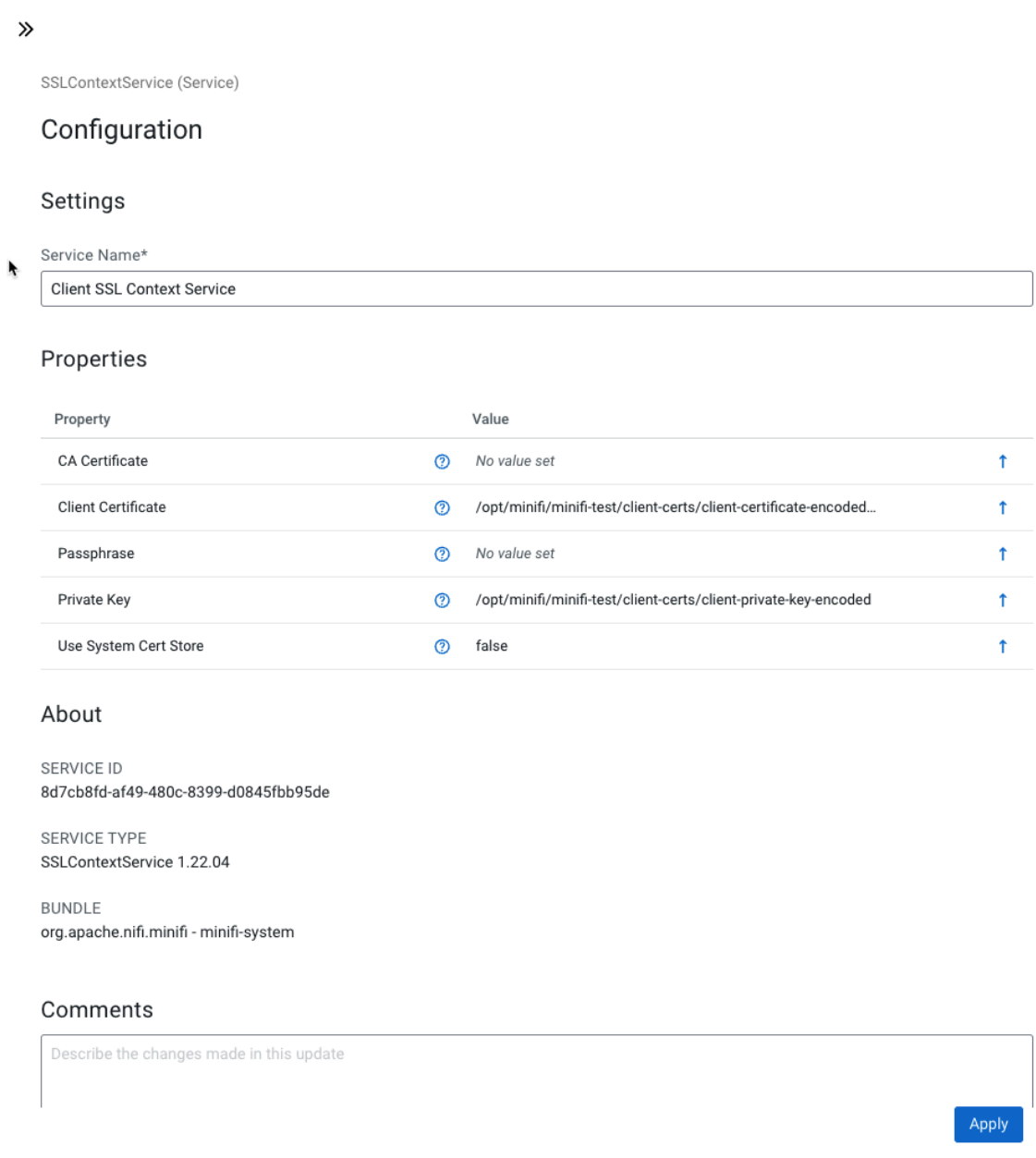

c. Click Apply.

d. Create an InvokeHTTP processor named Send to CDF with the following configuration:

Automatically Terminated Relationships

Select all relationships.

Content-type

Depends on your flow file data type. This tutorial uses text/plain.

HTTP Method

POST

Remote URL

https://[***ENDPOINT HOSTNAME COPIED FROM CLOUDERADATAFLOW FLOW DEPLOYMENT MANAGER***]:9000/contentListener

For example, https://my-flow.inbound.my-dfx.c94x5i9m.xcu2-8y8z.mycompany.test:9000/contentListener

SSL Context Service

Client SSL Context Service

Leave all other settings with their default values.

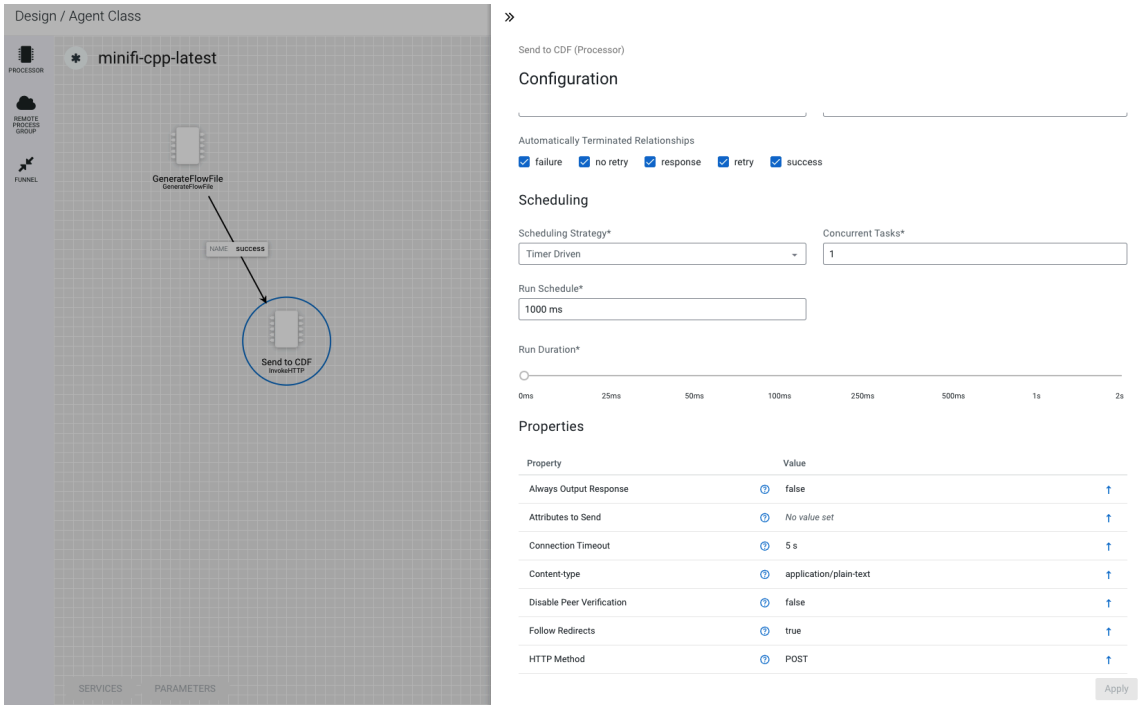

For Java

To design a flow for your MiNiFi Java agent class:

a. Convert the downloaded client-private-key-encoded key and client-certificate-encoded.cer certificate files to a JKS Keystore:

1. Create a PKCS12 keystore:

   openssl pkcs12 -export -in client-certificate-encoded -inkey client-private-key-encoded -out client-keystore.p12

2. Convert the PKCS12 keystore to a JKS keystore:

   keytool -importkeystore -srckeystore client-keystore.p12 -srcstoretype pkcs12 -destkeystore client-keystore.jks

b. Copy the resulting client-keystore.jks file to the host with the running MiNiFi Java agent, so they are accessible by filepath from the agent.

c. Obtain the CA root cert and add it to truststore client-truststore.jks, by running the following commands:

```
wget https://letsencrypt.org/certs/isrgrootx1.pem
```

```
keytool -import -file isrgrootx1.pem -alias isrgrootx1 -keystore c
lient-truststore.jks
```

MiNiFi Java requires you to specify an explicit truststore for inbound connections. Remember the password you used for creating client-truststore.jks, as you will need it .

d. Create a Service of type Restricted SSL Context Service with the following configuration:

Service Name

Specify a name for this service. This tutorial uses Client SSL Context Service.

Keystore Filename

*[***/PATH/TO/***]*client-truststore.jks

Keystore Password

*[***THE PASSWORD YOU PROVIDED WHEN CREATING THE JKS STORE***]*

Key Password

*[***THE PASSWORD YOU PROVIDED WHEN CREATING THE JKS STORE***]*

Keystore Type

JKS

Truststore Filename

client-truststore.jks

Truststore Type

JKS

Truststore Password

*[***THE PASSWORD YOU PROVIDED WHEN CREATING THE CLIENT TRUSTSTORE***]*

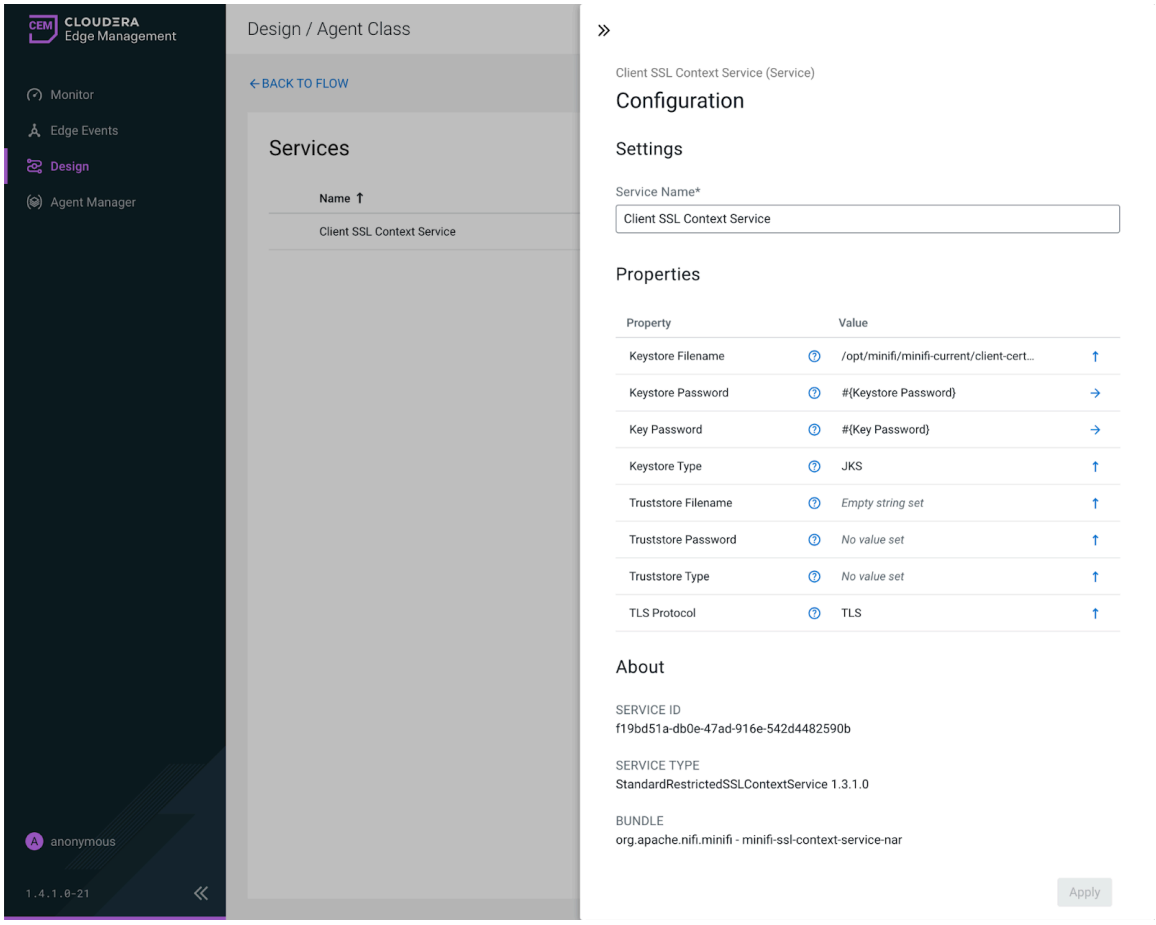

e. Click Apply.

f. Create an InvokeHTTP processor named Send to CDF with the following configuration:

Automatically Terminated Relationships

Select all relationships.

Content-type

Depends on your flow file data type. This tutorial uses text/plain.

HTTP Method

POST

Remote URL

https://[***ENDPOINT HOSTNAME COPIED FROM CLOUDERA DATAFLOW FLOW DEPLOYMENT MANAGER***]:9000/contentListener

For example, https://my-flow.inbound.my-dfx.c94x5i9m.xcu2-8y8z.mycompany.test:9000/contentListener

SSL Context Service

Client SSL Context Service

Leave all other settings with their default values.

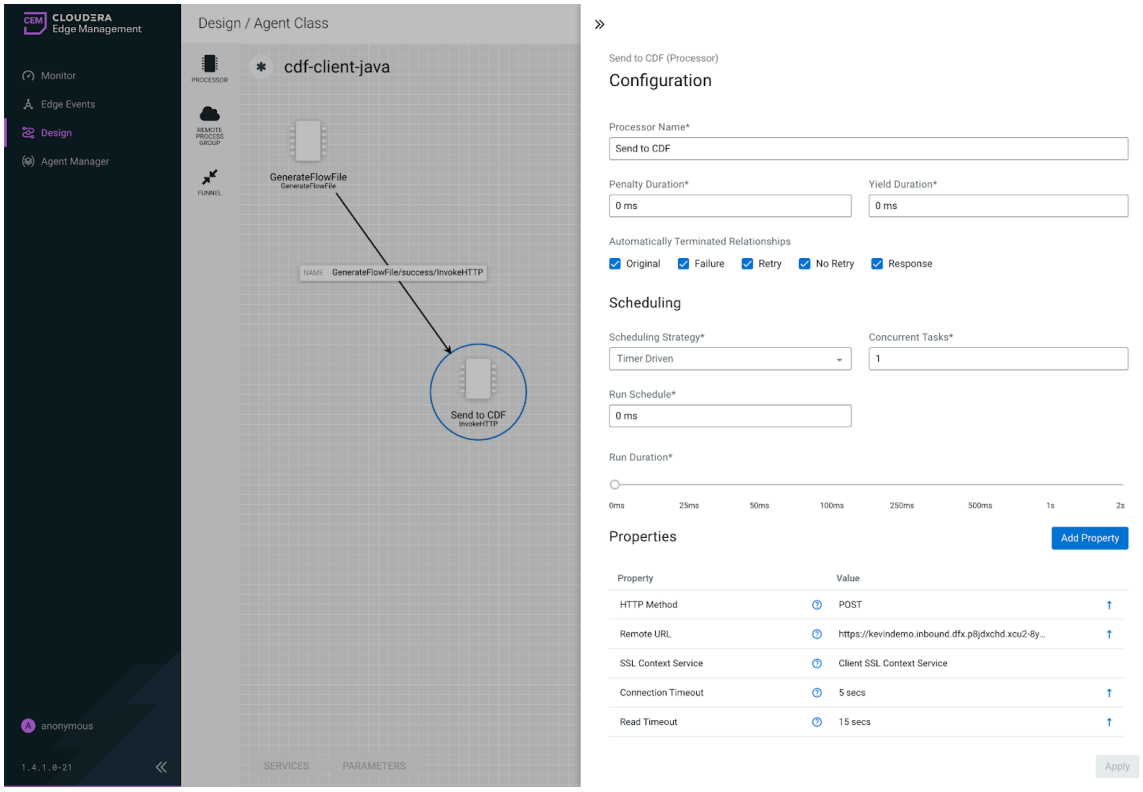

12. Build the rest of your data flow to read data and send to your Cloudera DataFlow flow deployment using InvokeHTTP. As a simple example, this tutorial uses the GenerateFlowFile processor, with the following settings:

Run Schedule

Set to 10000 ms (10 seconds).

Custom Text

The message you type here will be sent to the ListenHTTP Flow you have created, with the frequency specified by Run Schedule. For example, Hello DFX! This is MiNiFi.

Data Format

Set to Text.

Unique FlowFiles

Set to false.

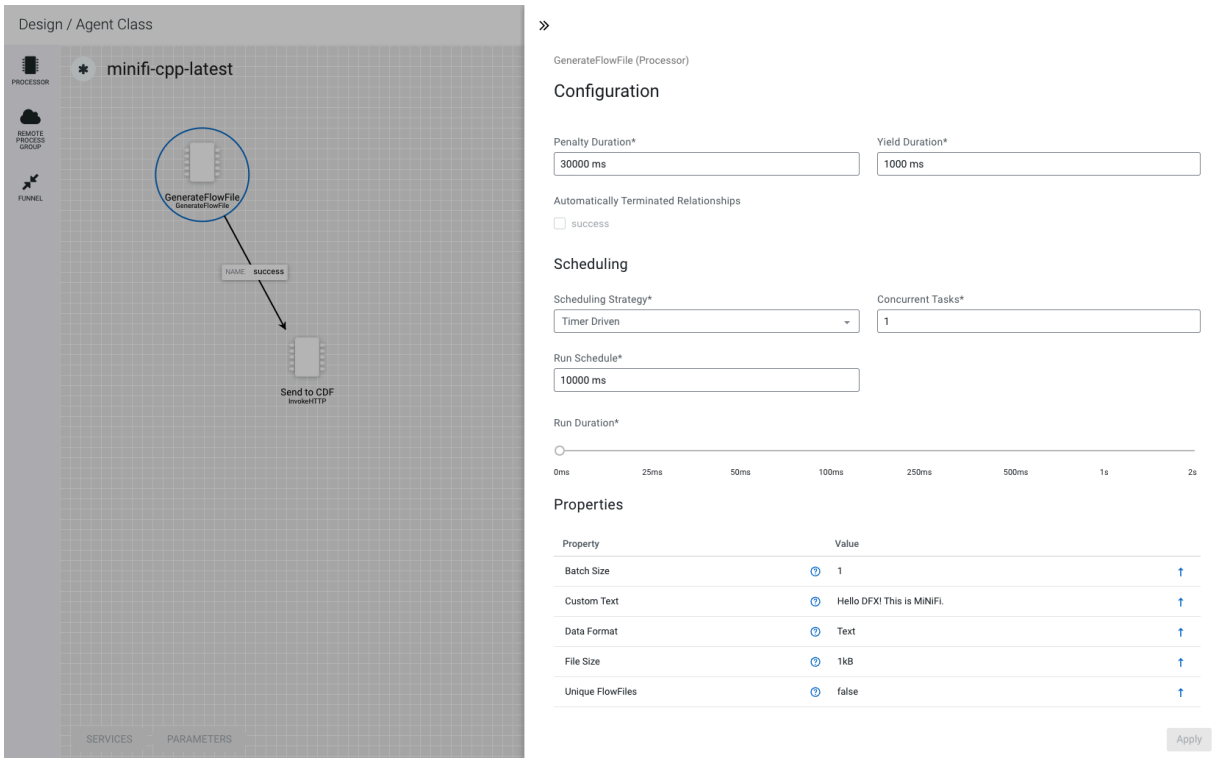

13. Connect the GenerateFlowFile processor to the InvokeHTTP processor.

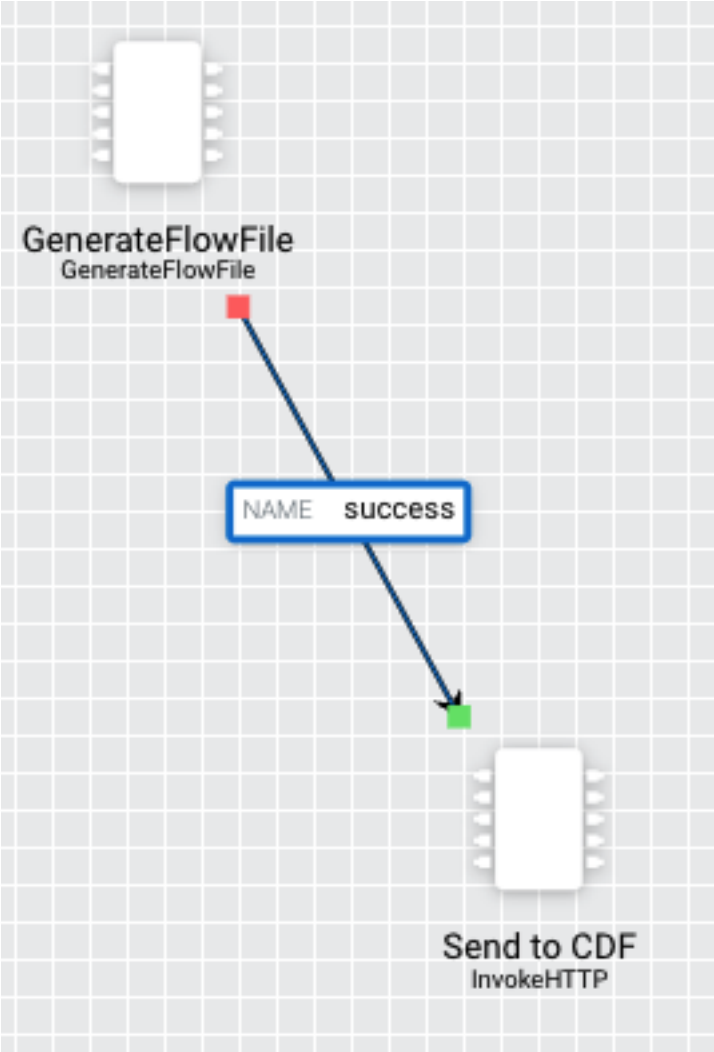

14. Click Actions Publish... to publish the flow and start it on your MiNiFi agent.

15. Select your flow deployment in the Cloudera DataFlow Dashboard and click KPIs.

Observe that your Cloudera DataFlow flow deployment is now receiving data from MiNiFi.

Related Information

Configure an external application as a client to an Inbound Connection Endpoint
Creating a flow definition in NiFi
Deploying a flow definition
Building a dataflow in CEM
Publishing a dataflow in CEM

Tutorial: Invoking an HTTP endpoint with curl

This tutorial walks you through invoking an HTTP Inbound Connection Endpoint with curl using the ListenHTTP filter to Kafka ReadyFlow from the ReadyFlow Gallery.

Before you begin

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Cloudera DataFlow Catalog.
- You have added the ListenHTTP filter to Kafka ReadyFlow to the Catalog. Adding a ReadyFlow to the Catalog requires the DFCatalogAdmin role.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- You have network connectivity established from your curl client to the virtual network or VPC where your Flow Deployment will run.
- You have downloaded and installed curl.

Procedure

1. Deploy the ListenHTTP filter to Kafka Ready Flow.
   a. Navigate to the ReadyFlow Gallery, locate the ListenHTTP filter to Kafka ReadyFlow and click View Added Flow Definition.
   b. Click Deploy and select your target environment to start the Deployment Wizard for the latest version of this ReadyFlow.
   c. Specify a deployment name, for example, Inbound Connections curl and click Next.
   d. Select Allow NiFi to receive data checkbox to configure an endpoint host.
   e. Accept the automatically created endpoint hostname and automatically discovered port by clicking Next.
   f. Optional: This ReadyFlow performs schema validation for incoming events using Cloudera's Schema Registry before sending the events to a Kafka topic. If you have a Streams Messaging cluster available, fill in the Kafka and Schema Registry connection properties.

      If you only want to validate inbound connection endpoint connectivity, enter dummy values for the empty parameters, set the Input and Output format to JSON while keeping the Listening Port set to 7001.
   g. At Sizing & Scaling select the Extra Small NiFi Node Size and click Next.
   h. Add a KPI on the ListenHTTP processor to monitor how many bytes it is receiving, by clicking Add new KPI.

      Make the following settings:

      KPI Scope

      Processor

      Processor Name

      ListenHTTP

      Metric to Track

      Bytes Received

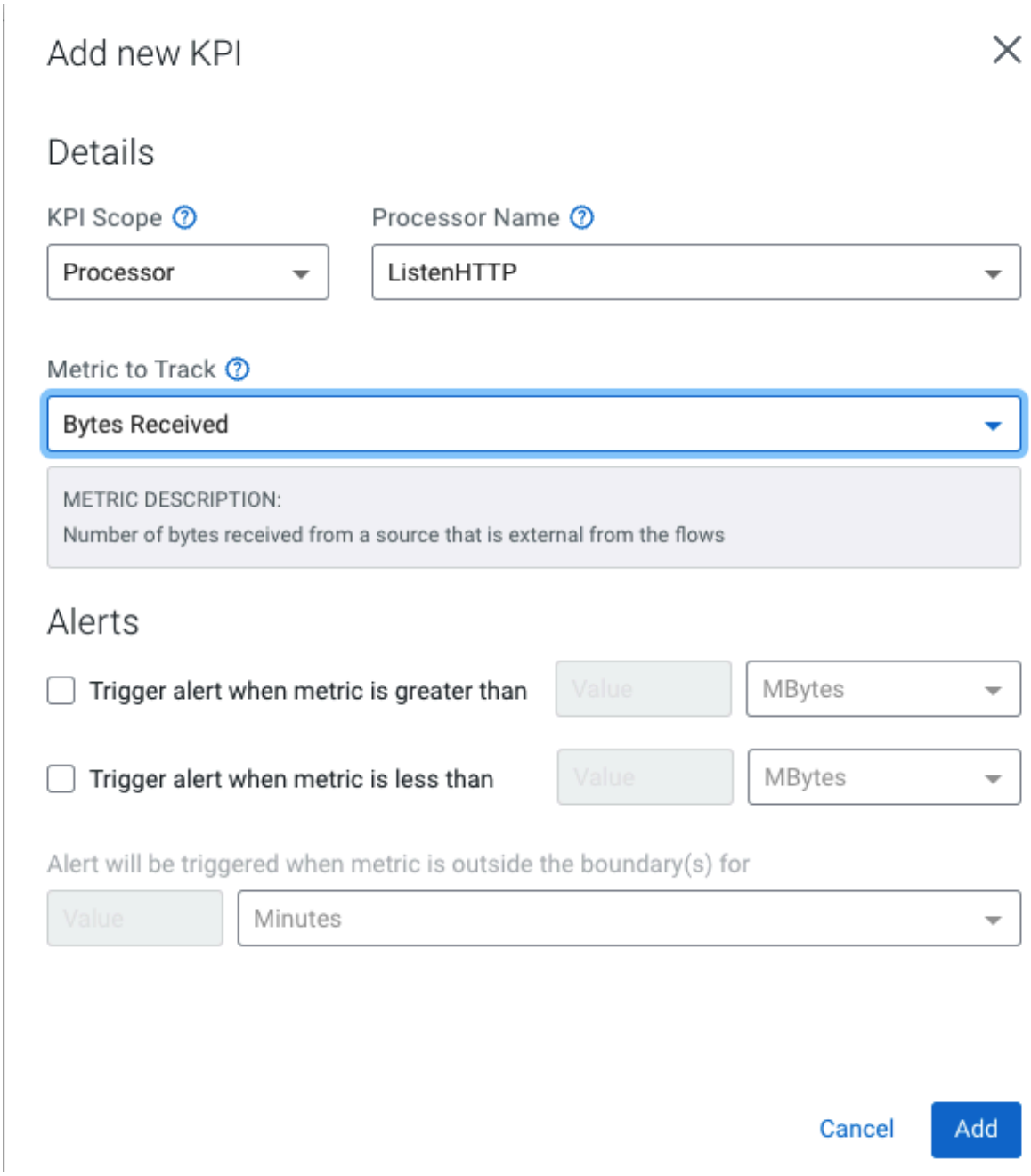

i. Click Next.
j. Review the information provided and click Deploy.

   Soon after the flow deployment has started, the client certificate and private key required for sending data to the NiFi flow become available for the flow deployment that is being created.

2. Collect the information required to configure your load balancer.
   a. Once the deployment has been created successfully, select it in the Deployments view and click Manage Deployment.
   b. In the Deployment Settings section, navigate to the **NiFi Configuration** tab to find information about the associated inbound connection endpoint.
   c. Copy the endpoint hostname and port and download the certificate and private key.

3. Create the curl request to validate connectivity to the HTTP inbound connection endpoint.

   Using the endpoint hostname, port, client certificate and private key you can now construct a curl command to call the endpoint and validate connectivity:

   curl -v -X POST https://*[***ENDPOINT HOSTNAME***]*:7001/contentListener --key *[***/PATH/TO/***]*client-private-key-encoded --cert *[***/PATH/TO/***]*client-certificate-encoded

   You receive an HTTP 200 response code in a similar message, indicating that your client was able to securely connect to the inbound connection endpoint:

```
*   Trying 10.36.84.149:7001...
* Connected to [***ENDPOINT HOSTNAME***] (10.36.84.149) port 7001 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*  CAfile: /Users/mkohs/letsencrypt-stg-root-x1.pem
*  CApath: none
* (304) (OUT), TLS handshake, Client hello (1):
* (304) (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Request CERT (13):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS handshake, CERT verify (15):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
*  subject: CN=dfx.dtefgqis.xcu2-8y8x.dev.cldr.work
*  start date: May 11 21:02:20 2022 GMT
*  expire date: Aug  9 21:02:19 2022 GMT
*  subjectAltName: host "[***ENDPOINT HOSTNAME***]" matched cert's
 "[***ENDPOINT HOSTNAME***]"
*  issuer: C=US; O=(STAGING) Let's Encrypt; CN=(STAGING) Artificial Aprico
t R3
*  SSL certificate verify ok.
> POST /contentListener HTTP/1.1
> Host: [***ENDPOINT HOSTNAME***]:7001
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 12 May 2022 00:45:57 GMT
< Content-Type: text/plain
< Content-Length: 0
< Server: Jetty(9.4.46.v20220331)
<
* Connection #0 to host [***ENDPOINT HOSTNAME***] left intact
```

4. Use curl to post data to the HTTP inbound connection endpoint.

   If you want to post events to the NiFi deployment you can add header and content definition to the curl request.

   This example sends JSON data following a simple schema to the endpoint:

```
curl -v -X POST [***ENDPOINT HOSTNAME***]:7001/contentListener \
 --key [***/PATH/TO/***]client-private-key-encoded \
 --cert [***/PATH/TO/***]client-certificate-encoded \
```

```
 -H 'Content-Type: application/json' \
 -d '{"created_at":6453,"id":6453,"text":"This is my test event","timest
amp_ms":34534,"id_store":12}'
```

The NiFi deployment tries to validate the schema against the schema name and Schema Registry that you provided when deploying the ReadyFlow. If you provided dummy values, you receive a response indicating that the flow was unable to look up the schema.

Related Information

ReadyFlow overview: ListenHTTP filter to Kafka

Using Inbound Connections with an external load balancer

Once a Cloudera DataFlow deployment with an Inbound Connection Endpoint is available, you can go on and connect an external load balancer to start sending data.

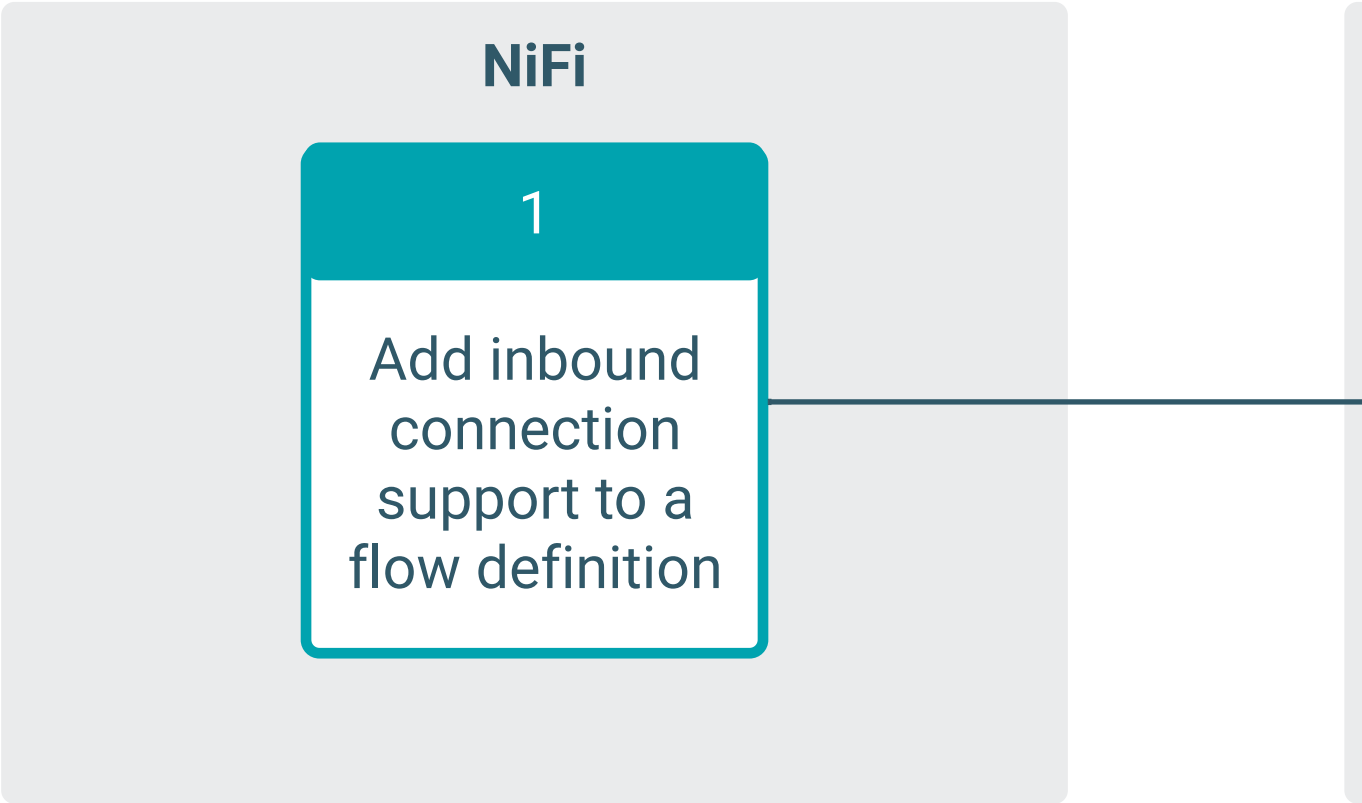

Inbound Connection Endpoints are created in Cloudera DataFlow with an internal Layer 4 (L4) load balancer (LB). Nevertheless, it is also possible to use your own native Layer 7 (L7) LB (Application Gateway on Azure, Application Load Balancer on AWS, respectively) in front of the Cloudera managed L4 LB.

Cloudera recommends achieving this by configuring your L7 LB to use the Cloudera DataFlow deployment LB as a backend. Enabling TLS between your LB and the Cloudera DataFlow LB is recommended, but mTLS is not possible for the backend connection. This means that your Listen Processor (e.g., ListenHTTP) in your NiFi flow cannot be configured with Client Auth = Required when using an external LB as a gateway.

You may configure the listening side of your LB and routing rules according to the requirements of your organization.

Alternatively, you may be required to use a L4 LB provided by your organization in front of the Cloudera managed LB. This is also possible, although Cloudera recommends directly using the Cloudera managed L4 LB when possible.

Typically, when using an external load balancer to act as a gateway, the internal managed load balancer should stay private. This can be accomplished by deselecting the “Use Public Endpoint” option when enabling Cloudera DataFlow for your environment, which limits Cloudera DataFlow to only use private subnets for all resources. If public access is needed, that would be done by exposing private resources via the external gateway load balancer.

Configuration workflow

Currently, an Inbound Connection Endpoint can only be created during flow deployment, and cannot be reassigned without terminating the flow deployment for which it was created.

To configure an external load balancer, you need to go through the following steps:

- Design a flow to accept inbound connections.
  - If the flow will be used with a Layer 7 LB, a compatible Layer 7 processor (such as ListenHTTP or HandleHttpRequest) must be used.
  - Decide if TLS will be used between your LB and your flow deployment
    - Cloudera recommends enabling TLS, which is done by configuring your listen processor to use an SSL Context Service named Inbound SSL Context Service. Cloudera DataFlow will manage the certificates for this context service when deployed. mTLS between cloud native load balancers is generally not supported, so unless you know that your Gateway LB supports client certificates and mTLS for the backend connection, it is recommended to configure the Client Auth property for your Listen processor to something other than REQUIRED (that is, use NONE, AUTO, or WANT)
    - If you will terminate TLS at your Gateway LB, you can optionally choose to use an unencrypted connection on the backend, which you can do by configuring your listen processor with no SSL Context Service.
- Add the flow to the Cloudera DataFlow Catalog.
- Create a flow deployment, with an Inbound Connection Endpoint.

  Note: If you have an existing endpoint left over after terminating a previous flow deployment, you can reuse it within the same environment.

- Download the client certificate and client key, note down the Endpoint Host Name, port number and protocol from the flow deployment where you want to send data using the external LB.

  1. Once the deployment has been created successfully, select it in the Deployments view and click Manage Deployment.
  2. In the Deployment Settings section, navigate to the **NiFi Configuration** tab to find information about the associated inbound connection endpoint.
  3. Copy the endpoint hostname and port and download the certificate and private key.

  Note: Ports are flow-specific, all the other attributes are specific to the individual endpoint host.

- Create a Layer 7 Load Balancer in your environment and configure its backend using the endpoint information obtained from the flow deployment.

Configure an Application Gateway in Azure

Learn about the settings required to set up an Azure Application Gateway to communicate with an Inbound Connection Endpoint.

About this task

Create an Azure Application Gateway service (you find it in the Networking services category) using the following settings:

Procedure

1. Make the following Backend Pool settings:

   Backend Pool without targets

   Set to No.

   Backend Targets

   IP address or FQDN

   Set to the *[*** INBOUND CONNECTION ENDPOINT HOSTNAME ***]* acquired from the NiFi settings of the flow deployment where you want to connect with your gateway.

   For example, my-endpoint.inbound.dfx.p8jdxchd.xcu2-8y8x.cloudera.site.

   For all other settings you can keep the default values.

   Tip:

   You can use a single gateway for multiple flow deployments, with rules for routing traffic to the correct flow deployment. For this scenario, setup multiple backend pools, one for each flow deployment.

2. Make the following Backend Settings:

   - If your flow listen processor uses TLS (recommended):

     Backend protocol:

     HTTPS

     Trusted root certificate

     Yes

     Backend port

     Match the port of your HTTP Listen Processor.

     For all other settings you can keep the default values.

   - If your flow listen processor does not use TLS:

     Backend protocol

     HTTP

     Backend port

     Match the port of your HTTP Listen Processor.

     For all other settings you can keep the default values.

   Tip: You can use a single Application Gateway for multiple HTTP listen processors, with rules to route traffic to the correct listen processor port. For this scenario, setup multiple backend settings, one for each listen processor.

Related Information

Create an application gateway

Cloudera DataFlow

Creating an Inbound Connection Endpoint

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Create an Inbound Connection Endpoint during flow deployment..... 4

Create an Inbound Connection Endpoint during flow deployment

During flow deployment, you can enable an endpoint that allows NiFi to listen to external data

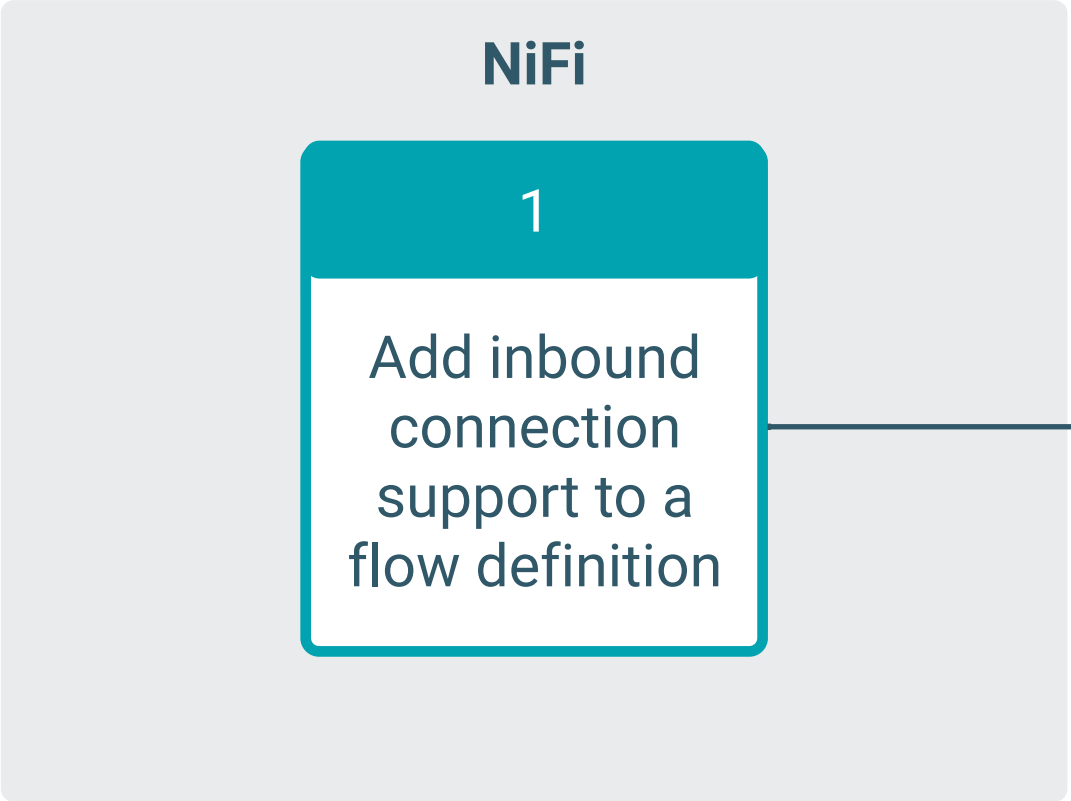

sources.

About this task

You can create flow deployments with an Inbound Connection Endpoint during flow deployment, using the Cloudera DataFlow Deployment wizard.

Note: Currently you cannot create flow deployments with Inbound Connection Endpoints using the CLI.

Before you begin

- A flow definition supporting inbound connections has been created and added to the Catalog.
- You have DFCatalogViewer role to view flow definitions in the Catalog.
- You have DFFlowAdmin right for the environment where you want to create the endpoint.

Procedure

1. During flow deployment, at the NiFi Configuration step, select Inbound Connections Allow NiFi to Receive Data .

   An Endpoint Hostname, generated from the Deployment Name you provided at the Overview step is offered and port configuration options become available.

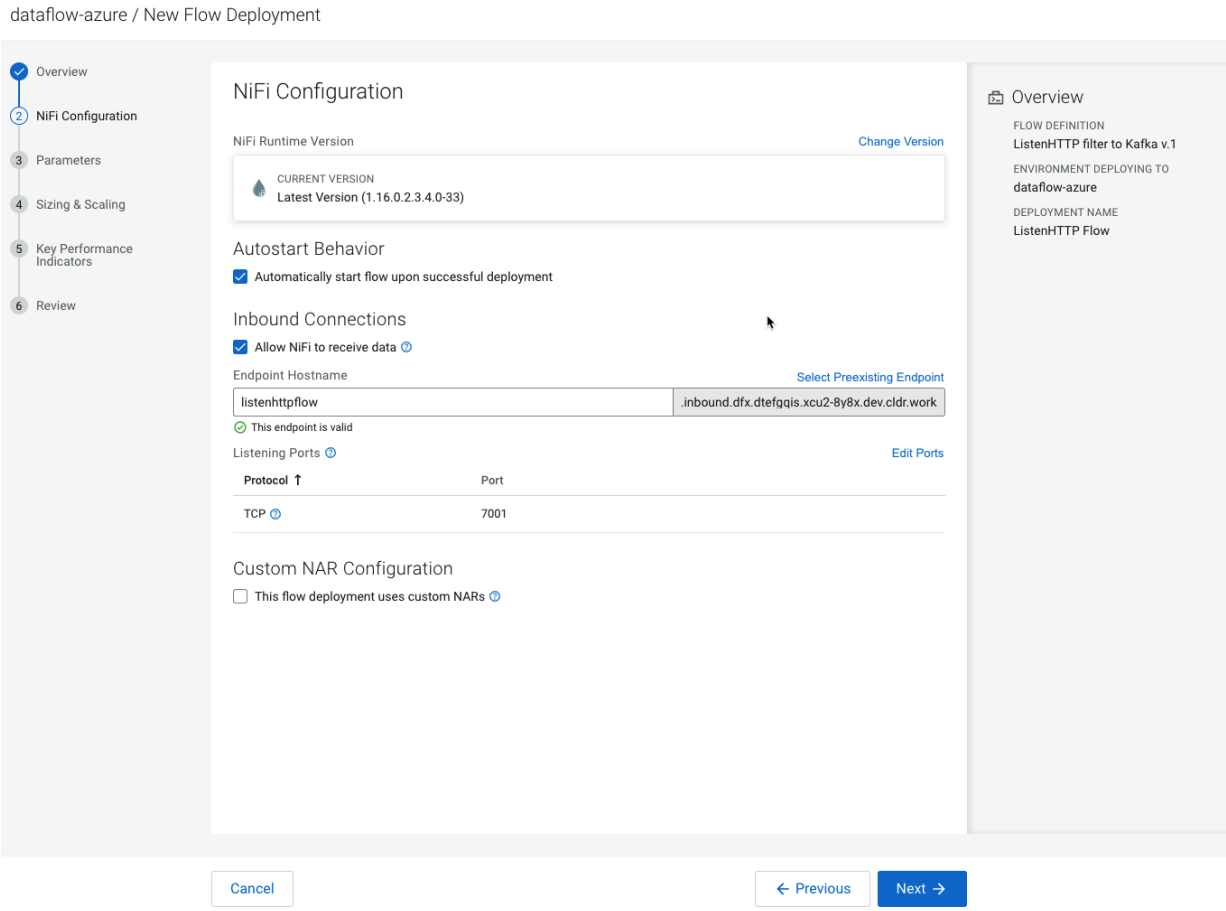

2. You can accept the generated hostname as is, or you can change the prefix. You can also click Select Preexisting Endpoint to select an available endpoint within your environment.

   Endpoints are unavailable if they are currently used by another flow deployment.

3. Select Listening Ports.

   Provide the protocol and port or ports where your flow will listen to incoming data. Cloudera DataFlow automatically suggests ports based on listen processors detected in the flow definition. Custom Processors, if applicable, are not detected and must be added manually.

   Using ports with mixed protocols (TCP and UDP) is not allowed. As long as the suggested ports use mixed protocols, you are not allowed to add new ports or to proceed to the next step in the Wizard.

   Note: If you want your dataflow to use HTTP, select TCP protocol.

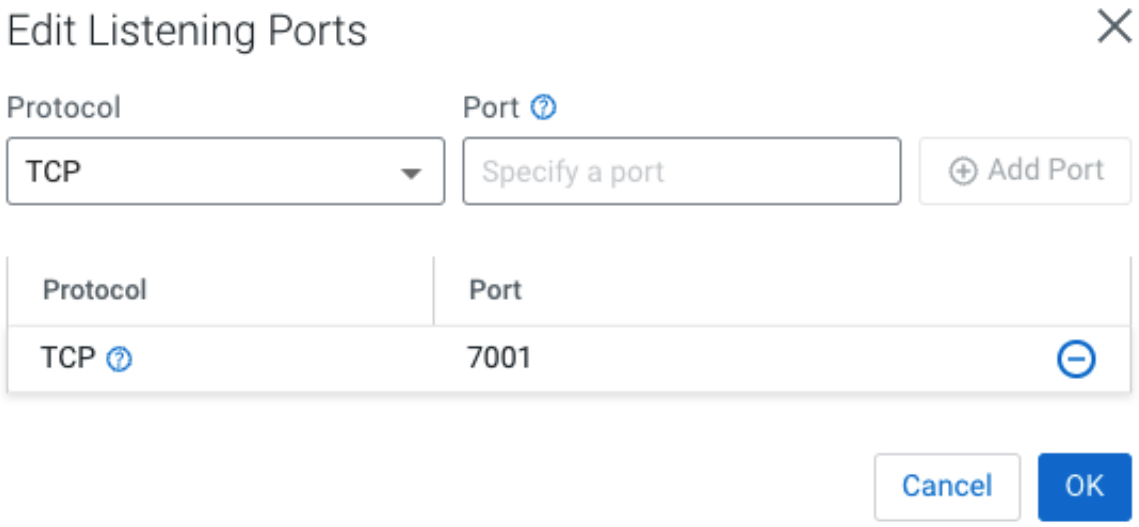

4. Fill in Parameters.

   If the listening port in your flow definition is parameterized instead of having a pre-defined value, you have to fill in the same listening port numbers you provided in the NiFi Configuration step. The listening ports you specified in the NiFi Configuration step are visible in the Overview pane.
5. Configure Sizing & Scaling, then click Next.
6. Add Key Performance Indicators, then click Next.
7. Review the information provided and click Deploy.

   As part of the flow deployment, a Layer-4 Load Balancer (LB) is created to forward traffic to the flow deployment's NiFi cluster on the configured ports and protocols. DNS records are created for Inbound Connection Endpoint hostname to resolve to the Layer-4 LB.

   Note: There is an initial delay for the cloud provider (AWS or Azure) to fully provision and start the LB resources, during which the flow deployment appears in the Cloudera DataFlow monitoring dashboard as healthy, but it is not yet able to receive data from clients. The load balancer provisioning normally completes within 10 minutes of the flow deployment being finished, but can take up to three hours.

Related Information

Deploying a flow definition

Cloudera DataFlow

Managing Inbound Connection Endpoints

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Deleting and reassigning Inbound Connection Endpoints..... 4

Renewing certificates for Outbound Connection Endpoints.....4

Renew certificates manually for an Inbound Connection Endpoint..... 4

Deleting and reassigning Inbound Connection Endpoints

Learn about deleting and reassigning Inbound Connection Endpoints.

Note: Inbound Connection Endpoints exist within the environment where they were created. They cannot be moved between environments. If you delete an environment, endpoints get destroyed as well and they cannot be reused.

- To delete an Inbound Connection Endpoint, you need to terminate the flow deployment to which it is currently assigned, selecting the Delete assigned endpoint hostname option during termination.
- To reassign an existing Inbound Connection Endpoint, you need to delete the flow deployment to which it is currently assigned, making sure that the Delete assigned endpoint hostname option is not selected. You can reuse existing, unassigned endpoints during flow deployment.

Renewing certificates for Outbound Connection Endpoints

Learn about certificate renewal for Inbound Connection Endpoints.

For security reasons, the cerificates generated for Inbound Connection Endpoints need to be renewed after a certain period:

- NiFi Inbound SSL Context Service - 90 days
- Client SSL Context - 1 year
- Client CA - 5 years

Cloudera DataFlow polls certificate status, and they are automatically renewed 30 days before expiration. At this point you receive a notification on the Alerts tab, asking you to restart the deployment so that the new certificate takes effect.

If it becomes necessary, for example, because a certificate is compromised, you can also renew certificates manually.

Renew certificates manually for an Inbound Connection Endpoint

If you need to replace an X.509 certificate for an inbound connection endpoint before it expires, you can do so manually.

Before you begin

You need DFFlowAdmin privilege to perform this action.

Procedure

1. On the Dashboard select the Cloudera DataFlow deployment for which you want to renew the certifiacate. The Deployment Details pane opens.
2. On the Deployment Details pane click Manage DataFlow.
3. On the Deployment Manager page from Deployment Settings pane select NiFi Configuration.

4. Click Renew Certificates.
   - To renew the server certificate, select NiFi Inbound SSL Context Service.

     Note: Each server certificate is limited to five renewals in a 7 day sliding window.
   - To renew the client certificate, select Client SSL Context.
   - If you leave Revoke previously issued client certificates unchecked, existing client certificates remain valid and existing clients can continue to connect to your deployment using it. By selecting the Revoke previously issued client certificates option, you invalidate all existing certificates and you will need to add the new certificate to existing clients so that they can keep connecting to your CDF deployment.
5. Click Renew & Restart.
   The UI switches to the **KPIs and Alerts** pane where you can monitor as your deployment restarts and the new certificate or certificates become available.

What to do next

If you have renewed the NiFi Inbound SSL Context Service:

You have take no further action.

If you have renewed the Client SSL Context:

After your Cloudera Public Cloud deployment has restarted, you switch to the NiFi Configuration pane to download the Client Certificate and the Client Private Key. You can then add these to your client.

Related Information

Connecting applications to an endpoint

Cloudera DataFlow

Managing DataFlow in an Environment

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing in an environment.....4

Disabling for an environment.....4

Clearing the environment Event History.....6

Resetting your environment.....7

Managing Kubernetes API Server user access.....7

Downloading kubeconfig.....8

Renewing certificates.....9

Updating Kubernetes node images in a service.....9

Configuring access for NiFi metrics scraping.....10

Managing in an environment

You can use the DataFlow Manager page to manage and monitor your environment.

The Actions drop-down menu in the DataFlow Manager page allows you to choose between the following options to manage in an environment:

- Disable for the environment
- Reset for the environment
- Manage user access for the Kubernetes API Server
- Download the Kubeconfig file
- Renew certificates
- Manage the environment details in
- Configure NiFi metrics access

Apart from the information on your Environment, the DataFlow Manager page also displays the capacity, networking, Kubernetes API Server endpoint access and tags of your environment under DataFlow Settings. You can edit the capacity settings of the environment, update the IP address ranges that are allowed to access the Kubernetes API Server and Load Balancer, and review the tags associated with the environment under DataFlow Settings.

Click Manage DataFlow, from the Environment Details pane to perform some actions on your environment. The DataFlow Manager page appears.

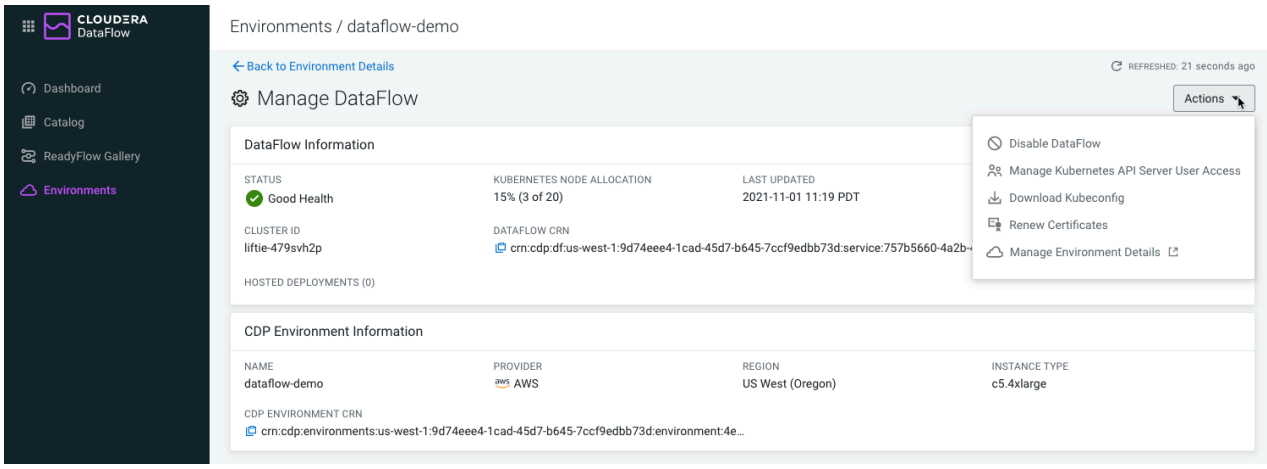

You can also go back to the environment details by clicking Back to Environment Details.

Disabling for an environment

Disabling for an environment terminates the cloud infrastructure that was created as part of the enablement process.

About this task

When you disable for an environment, you can specify whether to preserve your environment event history. Preserving the event history retains your environment event history and allows you to view past events even after the environment has been disabled. Regardless of whether you choose to preserve the event history, you can enable for an environment again after a successful disablement operation.

Steps

For UI

1. In , from the Environments page, select the environment you want to disable.

2. Click Manage DataFlow from the **Environment Details** pane.

   You are redirected to the Manage DataFlow page.
3. From the Actions menu, select Disable DataFlow.
4. Specify whether you want to Preserve event history.
5. Enter the environment name to confirm.
6. Select Disable to initiate the disablement process.

Example

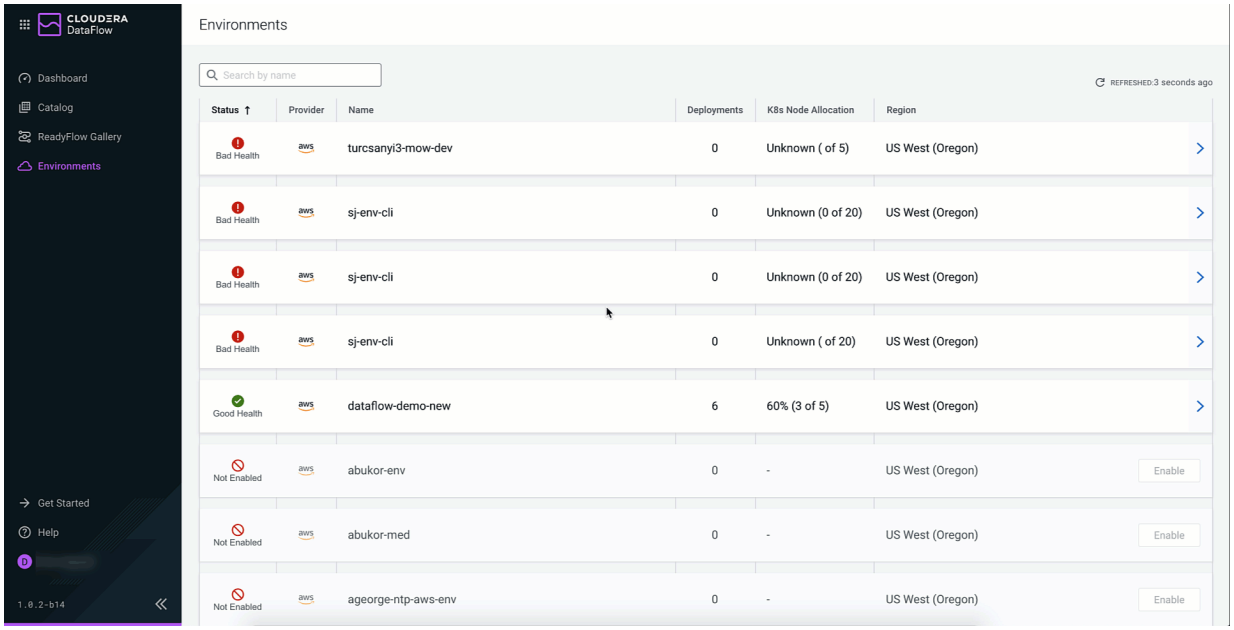

For CLI

Before you begin

- You have installed CDP CLI.
- Run cdp df list-services to get the service-crn.

1. To disable for an environment, enter:

```
cdp df disable-service
--service-crn [***SERVICE_CRN***]
[--persist] [--no-persist]
[--terminate-deployments] [--no-terminate-deployments]
                [help]
```

Where:

- service-crn – Provides the value you identified when you run cdp df list-services.
- [--persist] [--no-persist] – Select one to specify whether you want to preserve environment history.
- [--terminate-deployments] [--no-terminate-deployments] – Specifies whether you want to gracefully terminate deployments associated with this environment. Regardless of this setting all associated deployments will be terminated when you disable

Result

When you successfully disable for an environment, your result will be similar to:

```
{
```

```
  "status": {
    "state": "DISABLING",
    "message": "Disabling DataFlow",
    "detailedState": "TERMINATING_DEPLOYMENTS"
  }
}
```

Next steps

Disabling for an environment can take up to 30 minutes.

Related Information

Clearing the environment event history
Resetting your environment
Managing remote access
Downloading kubeconfig

Clearing the environment Event History

About this task

When disabling for an environment, you can choose to preserve the Event History for the specific environment. This allows you to review past events even after has been disabled for an environment. When the preserved events are no longer relevant, you can delete them by using the Clear Event History action.

Note:

The Clear event history action is only available for disabled environments with a preserved event history.

Before you begin

- You have the DFAdmin user role for the environment for which you want to clear the event history.

Procedure

1. Select the environment for which you want to clear the event history.
2. Click Manage DataFlow from the **Environment Details** pane.
   You are redirected to the **Manage DataFlow** page.
3. From the Actions menu, select Clear event history.
4. Select Clear Event History to confirm deleting all event-related information and past alert conditions.

Results

After successfully clearing the event history, you are no longer able to view the environment details by clicking on it. You can enable DF in the environment again by using the Enable button on the environment row.

Related Information

Disabling for an environment
Resetting your environment
Managing remote access
Downloading kubeconfig

Resetting your environment

When disabling for a specific environment fails, you can use the Reset Environment action to reset an environment state for .

Before you begin

- You have the DFAdmin user role for the environment you want to reset.

Procedure

1. In , from the Environment page, select the environment you want to reset.
2. Click Manage DataFlow from the **Environment Details** pane.
   You are redirected to the **Manage DataFlow** page.
3. From the Actions menu, select Reset Environment.
4. Click Reset in the confirmation dialog to proceed.

Results

Resetting an environment clears state without impacting the associated environment and any of its components including Data Hubs, Data Lakes, and FreeIPA. If the associated environment is still healthy, resetting allows you to enable it again for .

Note:

Resetting an environment does not delete associated cloud resources which were created during its enablement process. Manual steps may be necessary to address these orphaned resources in your cloud account.

Related Information

Disabling for an environment
Clearing the event history
Managing remote access
Downloading kubeconfig

Managing Kubernetes API Server user access

Giving users remote access to -enabled environments allows authorized users to use kubectl to manage and troubleshoot Kubernetes clusters using the Kubernetes API. To do this, use the Actions menu from the Environments page.

About this task

The API server of the Kubernetes cluster which is created when enabling a environment for is secured using authentication and role based access control. By default no one is allowed to connect to the Kubernetes API server. You can grant users access to the Kubernetes API server by adding their AWS ARN to the list of Authorized Users so they can communicate with the cluster using Kubernetes management tools such as `kubectl`.

Before you begin

- You have the DFAdmin user role.

- You have a cloud user ID. For AWS this is an ARN and looks similar to:

```
arn:aws:iam::{AWSaccountID}:role/{IAMRoleName}
```

See the *AWS documentation* for more information.

Procedure

1. In , from the Environments page, click the Environment for which you want to add or remove user access.
2. Click Manage DataFlow.
3. From the Actions menu, click Manage Kubernetes API Server User Access.
4. Provide the Cloud User ID you want to authorize.
   - To add more than one user, add Cloud User IDs one by one.
   - To remove a user, click the remove icon for the particular row.

What to do next

Download the kubeconfig file and share it with authorized users so they can connect to the cluster using their preferred Kubernetes management tools

Related Information

Amazon EKS IAM roles
Disabling for an environment
Clearing the environment event history
Resetting your environment
Downloading kubeconfig

Downloading kubeconfig

You can download the kubeconfig file so that you can use the `kubectl` management tool to manage and troubleshoot your Kubernetes cluster.

About this task

After granting users access to the Kubernetes API server, you can download the Kubeconfig for a Kubernetes cluster so they can communicate with it using Kubernetes management tools such as `kubectl`.

Before you begin

- You have the DFAdmin user role for the environment.

Procedure

1. In , from the Environments page, click the environment for which you want to download the kubeconfig file.
2. Click Manage DataFlow from the **Environment Details** pane.
   You are redirected to the **Manage DataFlow** page.
3. From the Actions menu, click Download Kubeconfig.
4. Share the kubeconfig file with authorized users.

Example

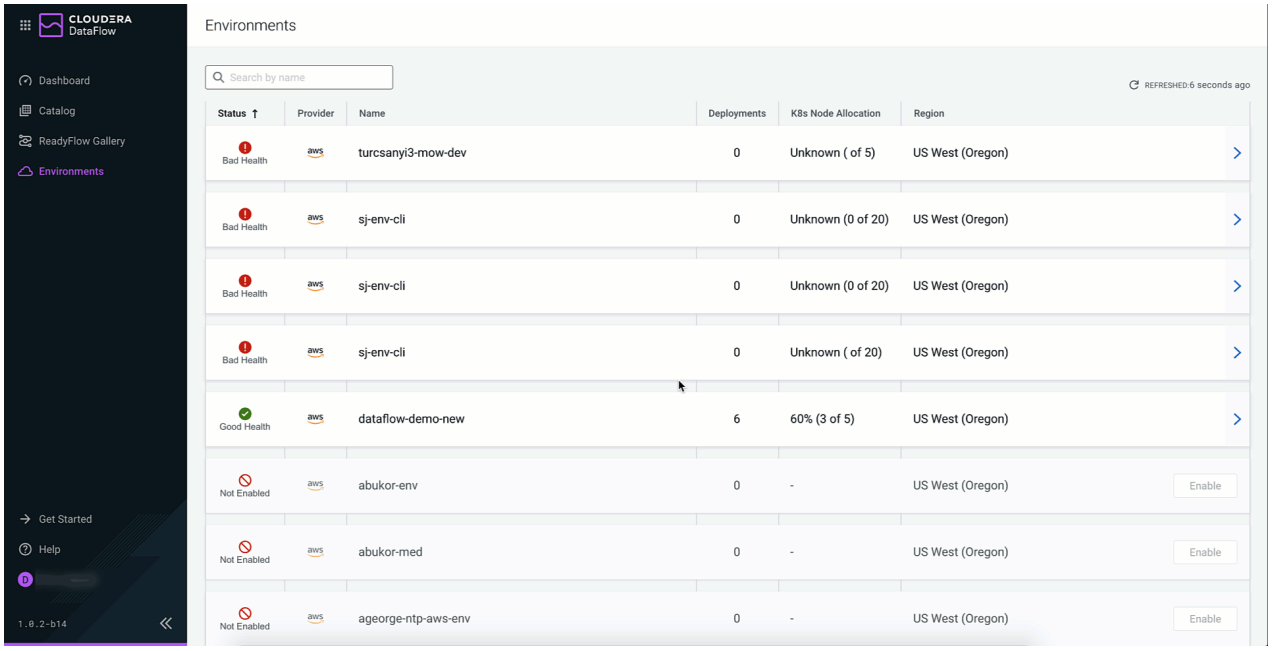

Related Information

Disabling for an environment
Clearing the environment event history
Resetting your environment
Managing remote access

Renewing certificates

Certificates for accessing have a 90 day lifespan. They are automatically renewed after 60 days. Should you need to manually renew your certificates you can use the Actions menu to do so.

Procedure

1. From the Environments page, click Actions.
2. Select Renew Certificates from the drop-down menu.
3. Confirm by clicking Renew Certificates.

Note:

Renewing your certificates assigns new certificates for accessing . The old certificates are not revoked.

Updating Kubernetes node images in a service

Learn about adopting a new Kubernetes node image for your service.

About this task

This action updates the images to the latest available version on the Kubernetes (K8s) nodes that form the underlying cluster in your service.

Note: This is primarily a troubleshooting option, you do not need to perform this as a routine maintenance task.

Important: During the update you are not able to create or manage deployments, drafts, or test sessions in the service.

Before you begin

- You have the DFAdmin user role for the environment where you want to update the node images.

Procedure

1. In , from the Environments page, select the Environment where you want to update the node images.
2. From the Actions menu select Update Node Images.
3. Click Update.
   - If there is a newer node image than the one already installed, the service status changes to Updating.

     Wait until the status returns to Good Health before initiating any other action.
   - If there is no newer version of the node image available, you get the message:

     'No new node image is available, nothing to update.'

Configuring access for NiFi metrics scraping

You can configure an external Prometheus service to scrape NiFi metrics for deployments. To do that, you need to generate a password and add a job for each deployment to your Prometheus configuration.

Before you begin

- You have the DFAdmin user role for the environment where you want to configure access for NiFi metrics scraping.

About this task

Tip: If you want to bulk add deployments to your Prometheus configuration, create a CLI script that collects deployment names and adds the required jobs with the generated password.

Note:

If you need the Endpoint URL of a deployment, go to Deployments [***DEPLOYMENT NAME***] Actions Manage Deployment NiFi Configuration and copy the **NIFI METRICS ENDPOINT URL**.

Procedure

1. In , from the Environments page, select the Environment where you want to configure NiFi metrics scraping.
2. From the Actions menu select Access NiFi Metrics.

3. Depending on whether you are setting up access for the first time or updating an existing one, select **Initial configuration** or **Manage existing**.

For Initial configuration

a. Click Generate Credentials and Enable Access.

   Copy the generated password. The username is nifi-metrics for all jobs, do not change it.

   Note: You will not be able to access the generated credentials after closing the dialog box.

b. Create a new job for each deployment where you want to perform metrics scraping and add it to your Prometheus configuration. Depending on your use case, either append it to an existing configuration or you can create a new one. You can use the provided **Sample Prometheus scrape configuration**.

   Figure 1: Sample metrics configuration code snippet

```
scrape_configs:
  - job_name: 'nifi-metrics-[***DEPLOYMENT-NAME***]'
    scrape_interval: 15s

    scheme: https
    honor_labels: true
    metrics_path: /dfx-[***DEPLOYMENT-NAME***]-ns/federate

    basic_auth:
      # Use ‘nifi-metrics’ as the username for all jobs.
      [ username: nifi-metrics ]
      [ password: [***GENERATED PASSWORD***] ]

    params:
      'match[]':
        # This parameter is mandatory, because Cloudera's Prometheus
 instance also scrapes cadvisor and Prometheus itself.
        - '{job="dfx-nifi-web"}'

    static_configs:
      - targets: ['https://dfx.qbllchii.xcu2-8y8x.dev.cldr.work']
```

   You need to replace

   - *[***DEPLOYMENT-NAME***]* with the encoded deployment name. (For example 'Some DataFlow Deployment' is encoded as 'some-dataflow-deployment')
   - *[***GENERATED PASSWORD***]* with the generated password.

   Depending on your Prometheus setup, you may need to make further additions to the job definition.

   Tip:

   To find a deployment-specific YAML snippet sample where you only need to substitute the generated password, go to Deployments [***DEPLOYMENT NAME***] Actions Manage Deployment NiFi Configuration and copy the **SAMPLE PROMETHEUS SCRAPE CONFIGURATION**.

c. Add the newly created or updated configuration file to your Prometheus service.

For Manage existing

- To turn on or off NiFi metrics scraping for all flow deployments in an environment, toggle the Access NiFi Metrics switch. Turning metrics scraping off stops exposing the Prometheus endpoint.

- If you need to reset the generated credentials, click Regenerate Credentials. Copy the generated password. The username is nifi-metrics for all jobs, do not change it. You will not be able to access the generated credentials after closing the dialog box.

  Note: Keep in mind that regenerating the credentials affects all flows in the given environment.

  Do not forget to update your Prometheus configuration with the regenerated credentials.
- To add new jobs or remove existing ones, modify the Prometheus configuration file.

  a. Create a new job for each deployment where you want to perform metrics scraping and add it to your Prometheus configuration. Add it to the existing configuration. You can use the provided **Sample Prometheus scrape configuration**.

  Figure 2: Sample metrics configuration code snippet

```
scrape_configs:
  - job_name: 'nifi-metrics-[***DEPLOYMENT-NAME***]'
    scrape_interval: 15s

    scheme: https
    honor_labels: true
    metrics_path: /dfx-[***DEPLOYMENT-NAME***]-ns/federate

    basic_auth:
      # Use 'nifi-metrics' as the username for all jobs.
      [ username: nifi-metrics ]
      [ password: [***GENERATED PASSWORD***] ]

    params:
      'match[]':
        # This parameter is mandatory, because Cloudera's Prometheu
s instance also scrapes cadvisor and Prometheus itself.
        - '{job="dfx-nifi-web"}'

    static_configs:
      - targets: ['https://dfx.qbllchii.xcu2-8y8x.dev.cldr.work']
```

  You need to replace

  - *[***DEPLOYMENT-NAME***]* with the encoded deployment name. (For example 'Some DataFlow Deployment' is encoded as 'some-dataflow-deployment')
  - *[***GENERATED PASSWORD***]* with the generated password.

  Depending on your Prometheus setup, you may need to make further additions to the job definition.

  Tip:

  To find a deployment-specific YAML snippet sample where you only need to substitute the generated password, go to Deployments [***DEPLOYMENT NAME***] Actions Manage Deployment NiFi Configuration and copy the **SAMPLE PROMETHEUS SCRAPE CONFIGURATION**.

  b. Add the updated configuration file to your Prometheus service.

Cloudera DataFlow

Managing Flow Definitions

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing flow definitions using the Catalog 4

Accessing flow definitions 4

Management of flow catalog 5

Management of flow definition details 6

Adding custom tags to flow definitions 8

Downloading the latest version of a flow definition 8

Deleting a flow definition 9

Managing flow definitions using the Catalog

In Cloudera DataFlow, your flow definitions are available in the Catalog. You can browse, discover, and manage flow definition versions in this view. You can also deploy and reuse your flow definitions across multiple environments.

Before you begin

- Make sure that you have DFCatalogAdmin permission. For information on account and resource roles, see *Cloudera DataFlow Authorization*.
- Import your Apache NiFi flow into Cloudera DataFlow, or publish your draft from Flow Designer to make it available as a flow definition in the Catalog.

Related Information

Cloudera DataFlow Authorization

Importing a flow definition from NiFi

Publishing a draft as a flow definition from Flow Designer

Accessing flow definitions

Learn how to access the Catalog page and flow definition details pane for managing flow definition details.

Procedure

1. Go to the Cloudera DataFlowUI.
2. Click Catalog from the left navigation pane.

   The Flow Catalog page appears showing the following information.

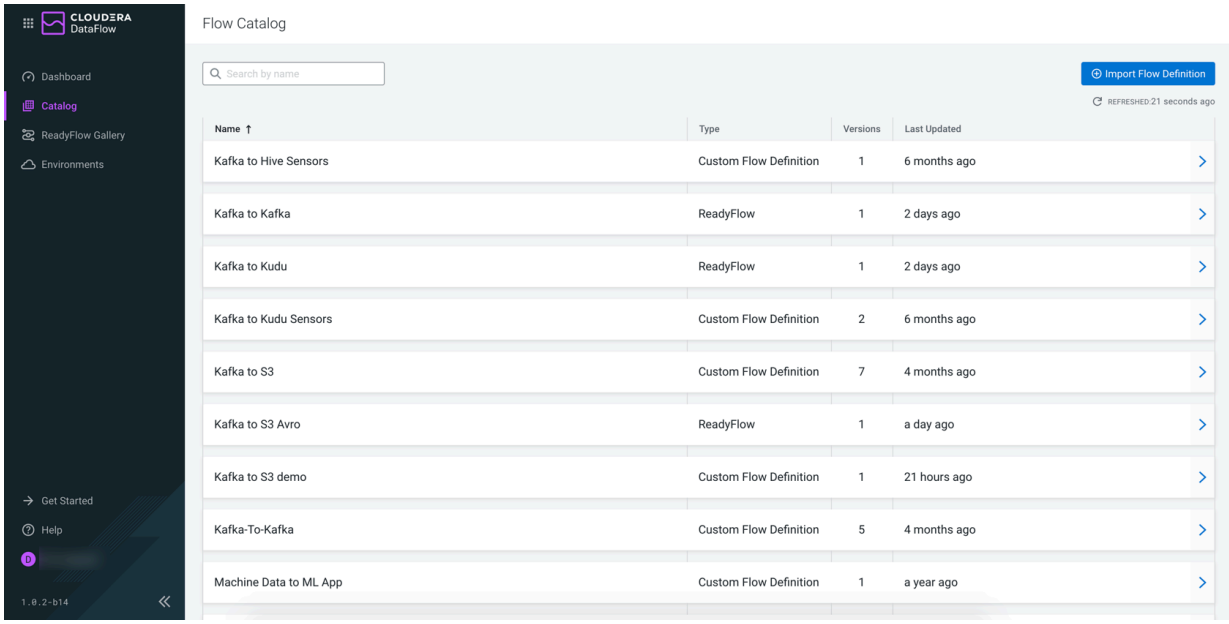

   For more information about what to track in the Flow Catalog page, see *Management of flow catalog*.

3. Click a flow definition row to expand and view additional information about it.

   The Flow Definition Details pane appears showing the following information.

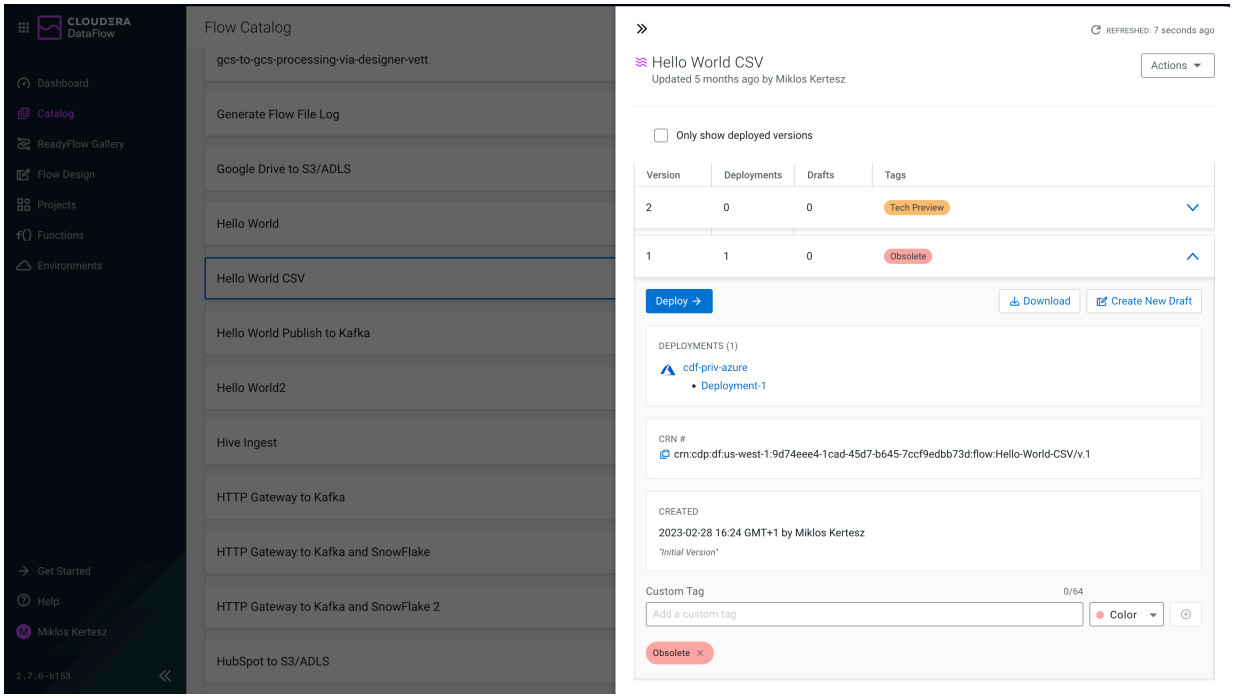

   For more information about what to track in the Flow Definition Details pane, see *Management of flow definition details*.

Related Information

Management of flow catalog

Management of flow definition details

Management of flow catalog

Learn the information that you can track in the Catalog page. Also, learn how to sort, search, and refresh your flow definitions.

You can track the following details of flow definitions in the Catalog page:

- Name – Name of the flow definition
- Type – Type of flow definition, Custom or ReadyFlow
- Versions – Number of available versions
- Last Updated – When the flow definition was last changed

You can sort flow definitions by Name, Type and Last Updated. Click the column name to sort by ascending or descending order.

You can also use the Search field at the top of the **Flow Catalog** page to directly search for the flow definition by its name.

You can use the Items per page option, at the bottom of the screen, to display the desired number of flow definitions in your page view.

You can click the Refresh button in the top right corner to refresh the data. By default, every 30 seconds all data for your flow definitions are updated.

Management of flow definition details

Find out about information that you can track in the Flow Definition Details pane. Also, learn how to import new version, download latest version, and delete your flow definitions.

You can track the following details of flow definitions in the Flow Definition Details pane:

- Flow description – This field gives you additional information about the flow definition.

  Note: You add this description when you import the flow from NiFi, or when you publish it to the Catalog from Flow Designer. You cannot edit it afterwards.

- CRN # – This field allows you to copy the CRN of the flow definition
- Only show deployed versions – Select this option if you want to view only the deployed versions of a flow definition.

- List of flow definition versions - This table provides you information about specific versions of the flow definition. It also allows you to perform various actions, such as deploy, download, or create a draft using a specific version of the flow definition.

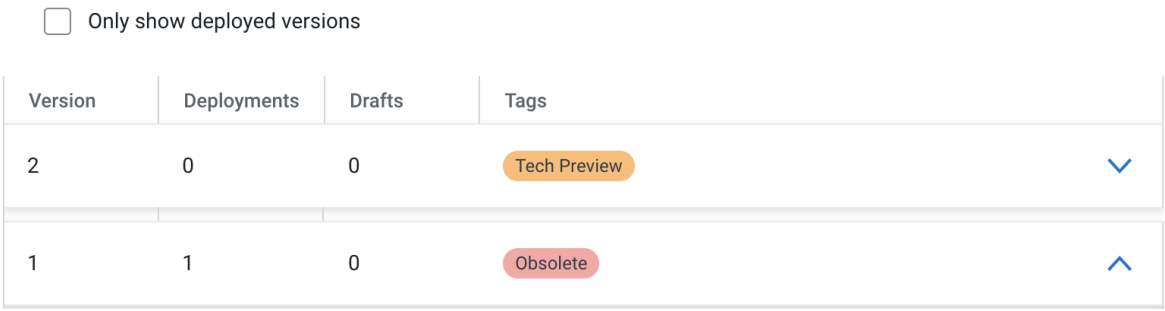

Version

This column shows the version number of the flow definition. You can store multiple versions of a flow definition in the Flow Catalog.

Deployments

This column shows the number of deployments that belong to a specific version of a flow definition.

Drafts

This column shows the number of drafts that are based on a specific version of a flow definition.

Tags

This column shows any tags added to a specific version of a flow definition.

Click a flow definition version row to perform actions using that version.

Deploy

Select to deploy the specific version of the flow definition.

Download

Select to download the JSON file of the specific version of a flow definition. This option is not available for the ReadyFlow type of flow definitions.

Create New Draft

Select to create a new draft in Flow Designer based on that specific version of the flow definition.

CRN #

This field enables you to copy the CRN number of the specific flow definition version.

Created

This field provides information about when the flow definition version was created and by which user. This section also includes the version comments entered by the user when the flow definition version was added.

Custom Tag

This field allows you to add free text tags to a specific version of a flow definition.

Deployments

This field shows you the number of deployments that belong to a flow definition. If you have the DFFlowAdmin role for the environments where the flow definition has been deployed, you can also see the list of the deployments.

- Actions – Use this drop-down menu to import a new version or delete a flow definition.

  Import New Version

  Select to upload new versions of an existing flow definition to the Flow Catalog.

  Download Latest Version

  Select to download the JSON file of the latest version of a flow definition from the flow catalog.

  Delete Flow Definition

  Select to remove a flow definition from the flow catalog along with all its versions.

Adding custom tags to flow definitions

You can add informative tags to individual versions of a flow definition in the Catalog.

About this task

Custom tags allow you to dynamically introduce additional information to flow versions on top of the static, non-modifiable version information. Once added, custom tags cannot be modified. You can delete an existing tag and replace it with a new one. If you change your mind, you can reintroduce a recently deleted tag using the Add Recently Used option.

Procedure

1. In the Cloudera DataFlow UI, from the left navigation pane, click Catalog.

   Flow definitions available to you are displayed, one definition per row.
2. Identify the flow definition where you want to add tags to a flow version, and click the row to display the flow definition details and versions.
3. Expand the flow version where you want to add tags by clicking Expand
4. Type the tag text under Custom Tag.
5. You can optionally change the default tag color, using the Color menu.
6. Click Add.

Downloading the latest version of a flow definition

You can directly download the latest version of a flow definition from the Cloudera DataFlow UI.

Before you begin

You have imported at least one flow definition.

Procedure

1. In the Cloudera DataFlow UI, from the left navigation pane, click Catalog.

   Flow definitions available for you are displayed, one definition per row.
2. Identify the flow definition you wish to download, and click the row to display the flow definition details and versions.
3. From the Actions drop-down, select Download Latest Version.
4. Click Download.

Results

The JSON file of the flow definition is downloaded. After you download the JSON file for a flow definition, you need to import it to add it to the Flow Catalog.

Example

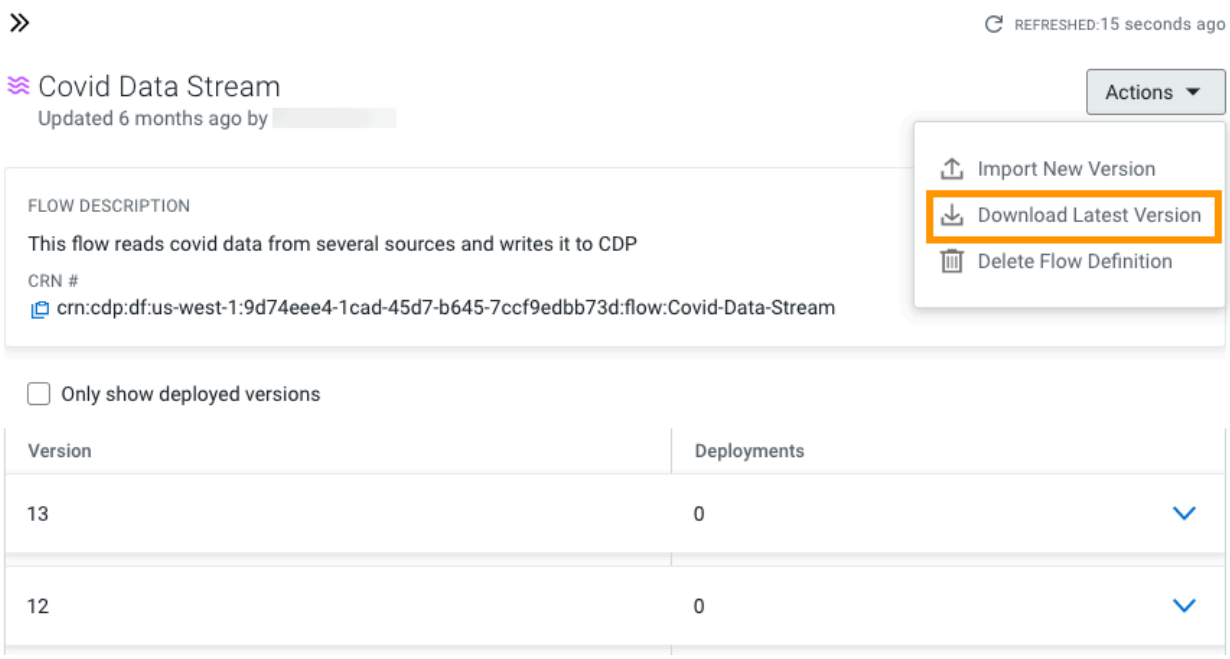

Deleting a flow definition

You can delete a flow definition along with all its versions from the Cloudera DataFlow UI.

Before you begin

You have imported at least one flow definition.

Procedure

1. In the Cloudera DataFlow UI, from the left navigation pane, click Catalog.

   Flow definitions available for you are displayed, one definition per row.
2. Identify the flow definition you wish to delete, and click the row to display the flow definition details and versions.

3. From the Actions drop-down, select Delete Flow Definition.

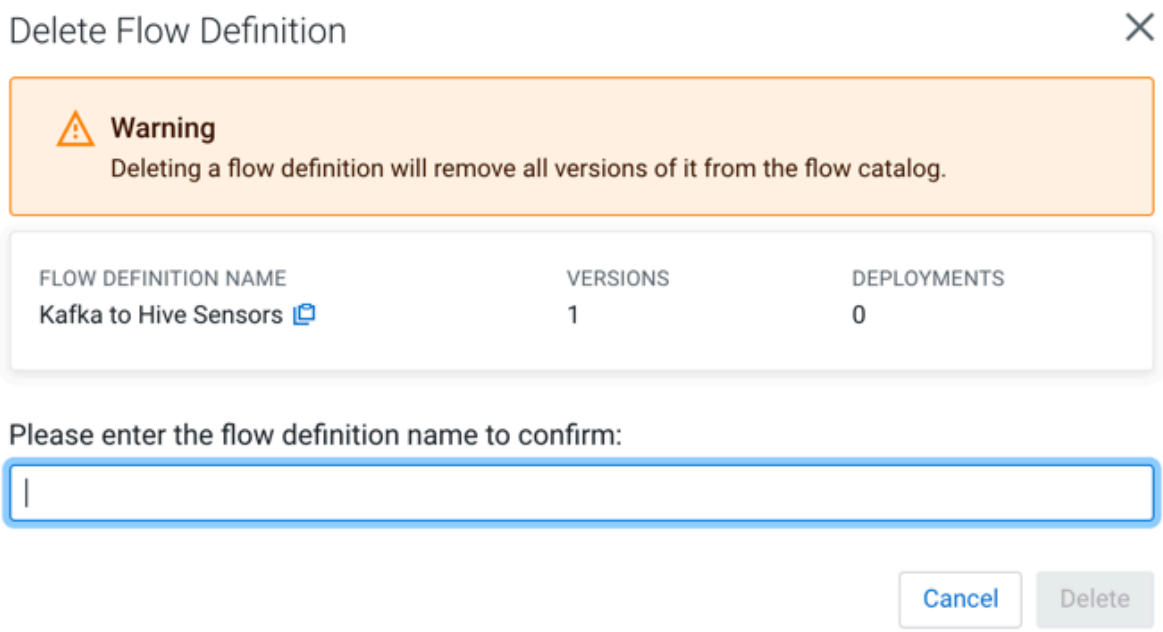

4. Enter the flow definition name, and click Delete.

Cloudera DataFlow

Managing Flow Deployments

Date published: 2021-04-06
Date modified: 2023-11-01

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing flow deployments through the Deployments page....4

Viewing data flow in NiFi....4

Starting a flow....5

Stopping a flow....6

Changing NiFi runtime version....7

Changing flow version....8

Change flow version steps....9

Change flow version troubleshooting....10

Downloading NiFi application log....11

Suspending a deployment....12

Resuming a deployment....13

Export deployment configuration....13

Terminating a deployment....14

Editing a deployment....15

Editing deployments on the user interface....15

Editing a deployment using the CLI....16

Managing flow deployments through the Deployments page

The Deployments page is the central place to monitor flow deployments across workspaces. It lists all flow deployments that you are allowed to view.

About this task

You can customize this page view by using the Status, Environments, Deployments, Projects, and Items per page.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.
5. In Deployment Details, click Actions to access various flow management options.
   - Select Actions Manage Deployment to access **Deployment Manager** where you can perform various management actions from the Actions drop-down. Under **Deployment Settings** you can also view and edit KPIs, alerts, parameters, sizing and scaling.
   - Select Actions View in NiFi to access the NiFi cluster where you can view and edit your flow.
   - Select Actions View Workspace to display all resources available to you within the present environment, providing a central place to view and manage them.

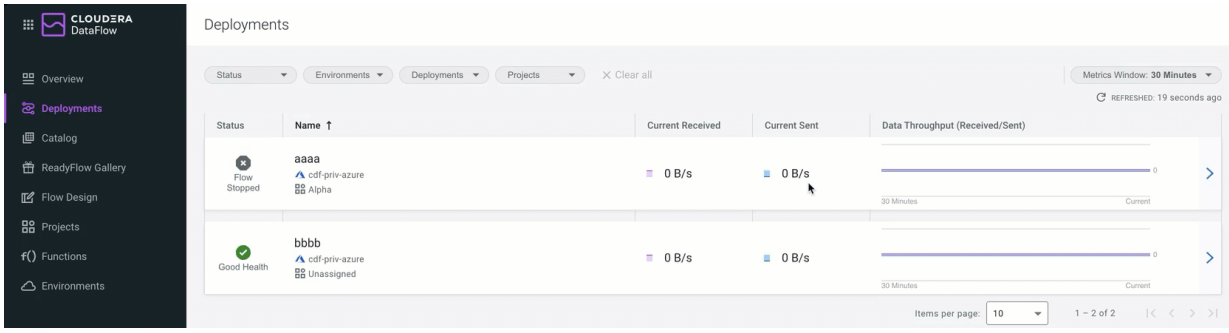

Viewing data flow in NiFi

You can go to the NiFi cluster where your flow is deployed and view or edit the data flow.

About this task

When you access the NiFi cluster, the ability to view or edit the flow is based on your Cloudera DataFlow authorizations. The DFFlowUser role has read-only privileges. The DFFlowAdmin role has full privileges.

Before you begin

You must have deployed a data flow in Cloudera DataFlow.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. Click Actions View in NiFi .

   The UI for the NiFi cluster where your flow is deployed opens.
5. View your data flow or edit it based on your NiFi privileges.

   If you edit the flow in NiFi and want the changes to exist in a new deployment, perform the following steps:

   a) Download the flow as a flow definition.

      For more information, see *Downloading a flow definition from NiFi*.

   b) Import the flow definition (as a new flow definition or as a new version of an existing flow definition).

      For more information, see *Importing a flow definition to Cloudera DataFlow*.

   c) Deploy the flow definition.

      For more information, see *Deploy a flow*.

Related Information

Downloading a flow definition from NiFi
Importing a flow definition to Cloudera DataFlow
Deploy a flow

Starting a flow

You can start a stopped flow a Cloudera DataFlow deployment.

About this task

Starting a flow deployment starts all processors of a Cloudera DataFlow deployment.

Before you begin

- You must have a stopped flow deployment in Cloudera DataFlow.
- You must have DFFlowAdmin permission.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.

5. Click Actions in the **Deployment Manager** page.

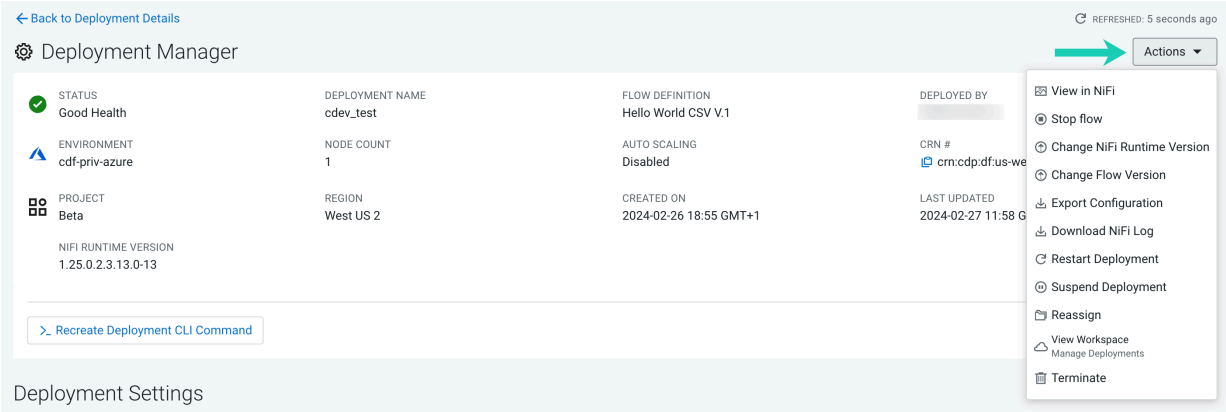

6. Select Start flow.

   The Start [Deployment Name] pop-up appears.

7. Confirm your choice by clicking Start Flow.

Stopping a flow

Stopping the flow of a Cloudera DataFlow deployment temporarily pauses the Apache NiFi flow.

About this task

Stopping a flow results in the following:

- All processors are stopped and no data processing happens within the NiFi flow.
- KPI alerts are stopped. Your KPI alerts are activated again when the flow is restarted.
- Any active KPI alerts are resolved.
- All underlying cloud resources remain allocated for the Cloudera DataFlow deployment.
- You can modify deployment configuration while the flow is stopped.
- Stopped flows are still billable however if auto-scaling is enabled for the flow, a certain amount of cost reduction may occur.

Before you begin

You must have deployed a flow definition in Cloudera DataFlow.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.

5. Click Actions in the **Deployment Manager** page.

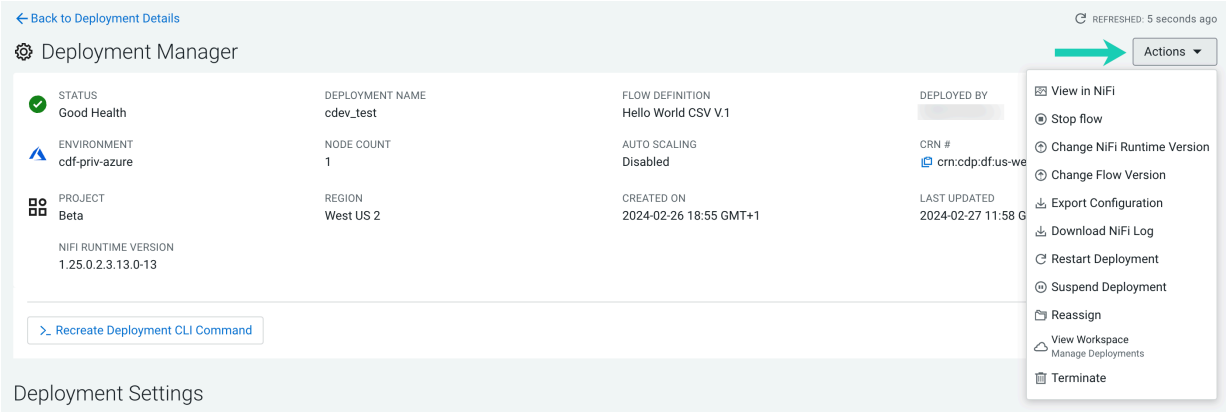

6. Select Stop flow.

   The Stop [Deployment Name] page appears.

7. Click Stop Flow to stop the flow deployment.

Changing NiFi runtime version

You can change the NiFi runtime version for your flow deployment. Generally, you change NiFi runtime version to pick up hotfixes and the latest NiFi version. However, when you create a flow deployment you can also pick the NiFi version to align with your flow certification needs. Cloudera recommends that you always use the latest NiFi version if possible.

About this task

If you opt to change the NiFi runtime version, the process is carried out node by node. If your flow deployment has multiple nodes, the flow deployment continues to run. If the flow deployment has only one node, it stops for a short period of time.

Before you begin

You must have deployed a flow definition in Cloudera DataFlow and you must have the DFFlowAdmin role.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.

5. Click Actions in the **Deployment Manager** page.

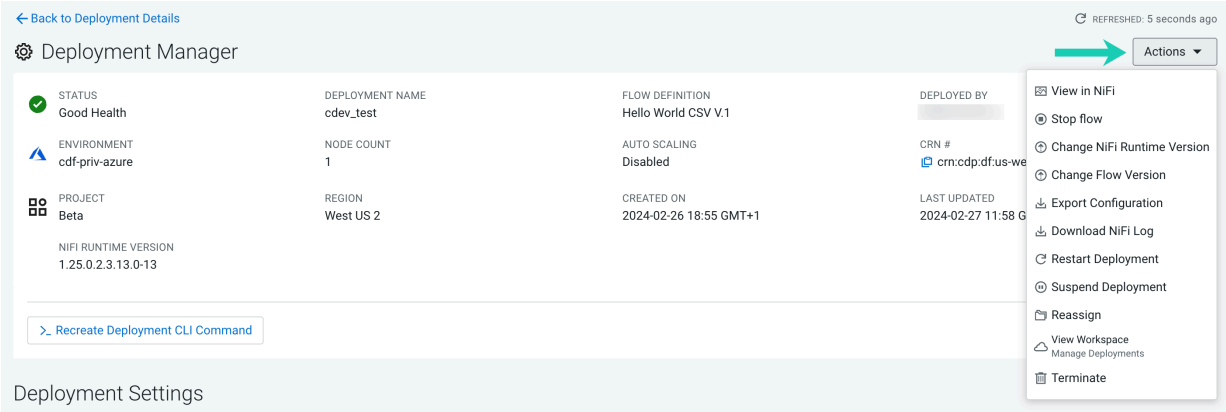

6. Select Change NiFi Runtime Version.

   The Change NiFi Runtime Version page appears. It shows the current NiFi runtime version and a list of available version numbers.

7. Select the new version number from the New Version dropdown, and click Update.

Changing flow version

The change flow version feature eliminates the need to tear down and rebuild your flow deployments when changing to another flow definition version. Consider restrictions and available version change strategies.

Important: As every flow is unique, you need to test version change before performing it in a production environment

Restrictions

The following restrictions apply to flow version changes:

- Changing inbound connections is not supported.
- Changing custom resource (custom NARs and custom Python resources) configurations is not supported.
- While you can add, change or remove assets when moving to a new version, you cannot introduce assets (text files, binaries, JARs, or similar) if the currently deployed version does not have any.
- Components where state or provenance and other repositories must be kept between flow versions must keep their flow JSON ids. The id changes if you move the component to a different process group or if you delete and then re-add the component to the same process group. NiFi Identifies components by these ids. If you move a component to a different process group between versions, its id changes and NiFi perceives it as a new component. This results in the original component being deleted during flow version change together with its state and a new, identical processor being created in a different process group. In an extreme case, you could change to an identical flow version with just the component ids changed and it would result in the deletion of the entire NiFi flow and the recreation of an identical one, with all history and data lost.
- Remapping Parameter Group and Parameter Context assignments is not supported as the original assignment is not removed. For example, you have Process Group 1 (PG1) with Parameter Context 1 (PC1) and Process Group 2 (PG2) with Parameter Context 2 (PC2) assigned. If you initiate a flow version change where parameter contexts are flipped, resulting in a PG1-PC2 and PG2-PC1 assignment, NiFi will not re-map the PG to PC assignments.

Version change strategies

Depending on the type of your flow, you may select the flow version change strategy most appropriate to you.

Stop & Process Data

This strategy prioritizes data consistency by stopping source processors and waiting until data is processed before stopping all other components. Once all components have stopped, the flow version is changed and components are started.

Use this strategy when your sources are durable and can handle your source processor being stopped. This generally works well when your source processors are pulling data from sources like Kafka or other messaging queues, databases or file systems.

Should the queued data not be processed within the set time, version change will fail and you can retry the operation with a bigger timeout or you can cancel

Only Restart Affected Components

This strategy prioritizes uptime by identifying and stopping only components that have changed while keeping all others running, replacing and then starting affected components.

Use this strategy when you want to prioritize uptime of unchanged components or you have made only minor processor configuration changes.

This works well for deployments with inbound connections and will keep your source processors running if they have not changed compared to the previous version.

Stop & Empty Queues

This strategy forces a version change by stopping all components, emptying all queues, changing flow version, and then starting all components.

Use this strategy only when you want to force a flow version change without keeping any processors running or attempting to process queued data.

All processors will be stopped and all queues will be emptied as part of this strategy.

Change flow version steps

Learn how to change the flow definition version of a running flow deployment. Using the ‘change flow version’ capability eliminates the need to terminate and re-create deployments when you want to deploy a new version of your flow definition.

Before you begin

- You must have DFFlowAdmin permission.
- There is at least one more version for the same flow definition present in the catalog
- The state of the flow deployment is either Good Health or Stopped.
- You have read the applicable restrictions and version change strategies.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.

5. Click Actions in the **Deployment Manager** page.

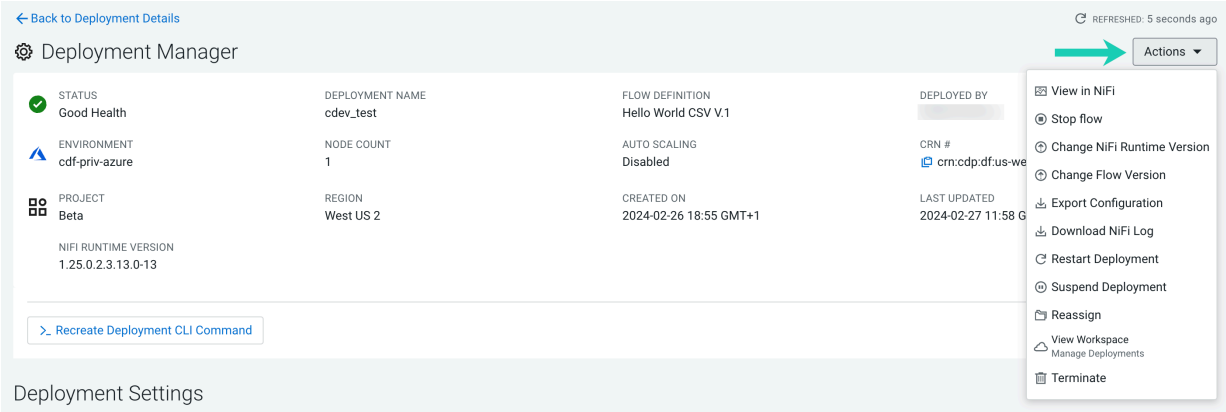

6. Select Change Flow Version.

   The **Change Flow Version** modal window opens. It shows the list of available flow versions. The current version is grayed out.

   Tip: You can filter versions by name or by tags.

7. Select the flow version you want to change to and click Continue .
8. Review a summary of the configuration changes caused by the version change and make any necessary edits from the left pane.
9. Select a flow version change strategy.

   The available options are:

   - Stop & Process Data - If you select this strategy, you can set the maximum wait time for data to be processed and queues to be emptied before the request timed out. The default value is 15 minutes.

     Note: If you initiate flow version change with this strategy on a stopped flow and there is no data in the queues, flow version change instantly happens regardless of the wait time you configured. However if there is data left in the queues of the stopped flow, the change operation instantly fails.
   - Only Restart Affected Components

     Note: If you initiate flow version change with this strategy and there are queued flow files on any connection that is going to be removed in the new flow version, the version change will fail.
   - Stop & Empty Queues - If you select this strategy, you must accept potential data loss by selecting I understand and choose to proceed with the configuration as is.
10. Click Change Flow Version.

Results

After you click Deploy, you are redirected to the **Alerts** tab in the **Flow Details** view where you can track how the version change progresses.

Change flow version troubleshooting

If changing the flow version fails, you can investigate the root cause in Cloudera DataFlow and in NiFi. In ceerain cases you can retry or cancel the operation.

Procedure

- Check the cause of the failure on the **Alerts** tab of the flow details view.
- If the information provided on the **Alerts** tab is insufficient, try checking the NiFi UI.

  Select Actions View In NiFi .
- If version change fails during the preparatory phase, before any actual changes were made, you have the option to cancel. This will roll back your flow to the state before initiating the version change. Cloudera DataFlow only offers this option when a rollback is possible.

  Select Actions Manage Deployment Cancel Version Change .
- Another option when version change fails during the preparatory phase is to retry the operation, possibly with a longer timeout or different parameter values in case incorrect parameter values caused the version change to fail.

  Select Actions Manage Deployment Retry Version Change .

Downloading NiFi application log

You can download the NiFi application log from the CDF Deployment Manager to use it for troubleshooting.

About this task

This feature allows you to download the NiFi application log that is currently being written. As the log file is rotated and the old file is archived once the file size reaches 10 MB, this is the theoretical maximum you can download using this method. For information on downloading archived log files, see *Diagnostic bundle collection*.

Before you begin

You need DFFlowAdmin permission to perform this action.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.
5. Click Actions in the **Deployment Manager** page.

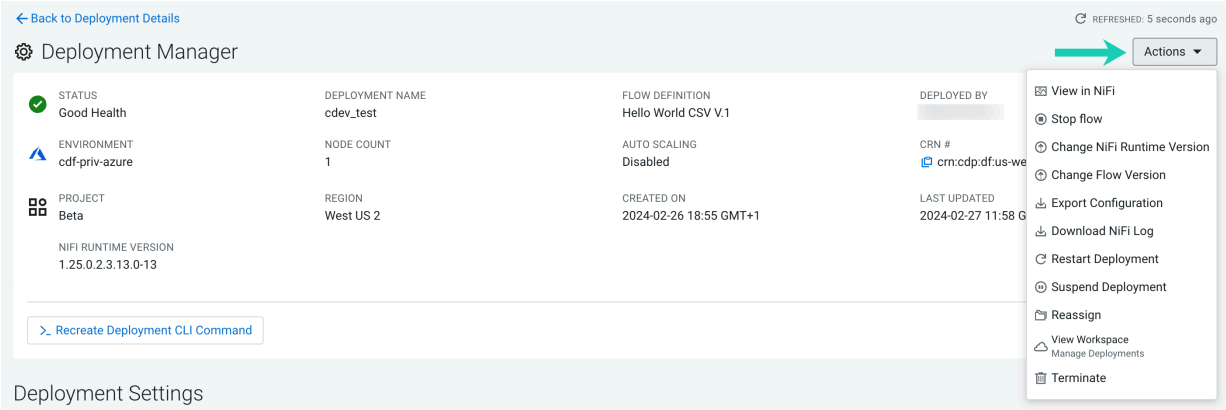

6. Select Download NiFi Log.

Results

The current NiFi application log is downloaded to your computer in tar.gz format.

Related Information

Diagnostic bundle collection

Suspending a deployment

Suspending a Cloudera DataFlow deployment terminates cloud resources belonging to an Apache NiFi flow, while maintaining flow persistence.

About this task

Suspending a Cloudera DataFlow deployment results in the following:

- The NiFi flow stops processing data and all underlying cloud resources are terminated. Any unprocessed data in the flow is stored in memory and its processing resumes when you resume the deployment.
- Flow persistence is maintained while a deployment is suspended.
- You cannot modify deployment configuration while the deployment is suspended.
- Suspended deployments are not billable, resulting in reduced costs.

Before you begin

You must have deployed a flow definition in Cloudera DataFlow.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.
5. Click Actions in the **Deployment Manager** page.

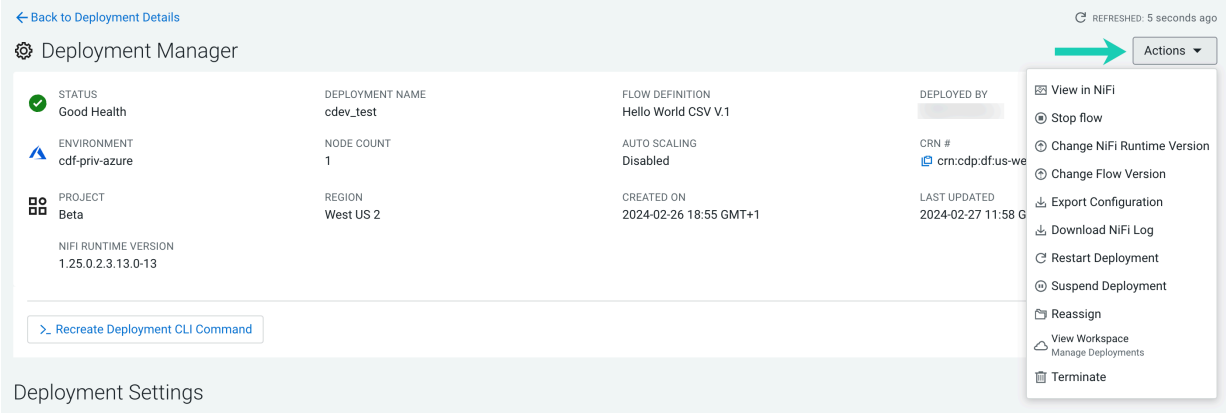

6. Select Suspend Deployment.

   The Suspend [Deployment Name] page appears.
7. Click Suspend Deployment to suspend the Cloudera DataFlow deployment.

Resuming a deployment

You can resume a suspended Cloudera DataFlow deployment.

About this task

Resuming a Cloudera DataFlow deployment reallocates the underlying cloud resources and returns a deployment to the state it was in before being suspended.

Before you begin

You must have a suspended flow deployment in Cloudera DataFlow.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.
5. Click Actions in the **Deployment Manager** page.

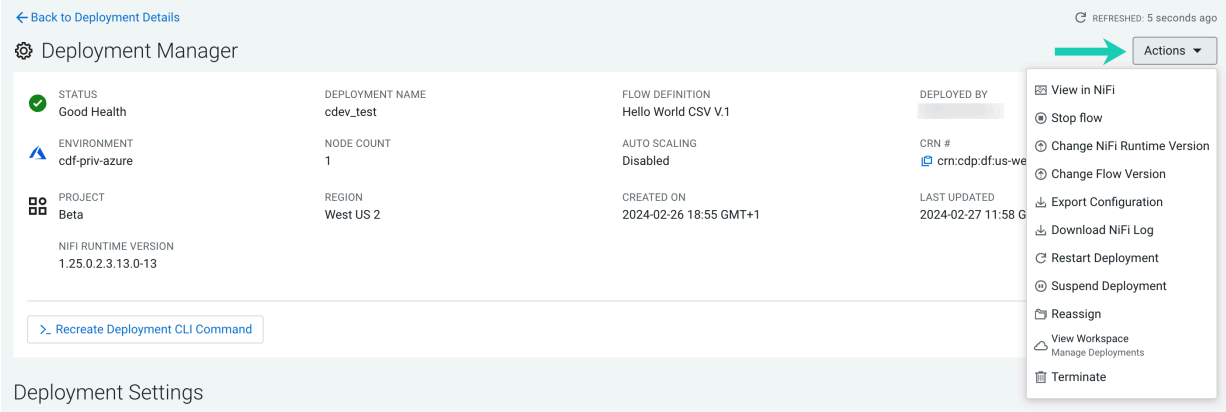

6. Select Resume Deployment.

   The Resume [Deployment Name] pop-up appears.
7. Click Resume Deployment to resume the flow deployment, reallocating cloud resources.

Export deployment configuration

You can export a deployment configuration to create additional deployments with a similar configuration in the same or a different environment.

About this task

- Exported configurations may be edited, and you can also modify them after the importing step during flow deployment.
- One deployment can have only one exported configuration. Performing a new export overwrites the existing one.

- Exported deployment configurations are available for every user who can start a new deployment in a given environment, even if the exported deployment was originally created under a specific Project.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.
5. Click Actions in the **Deployment Manager** page.

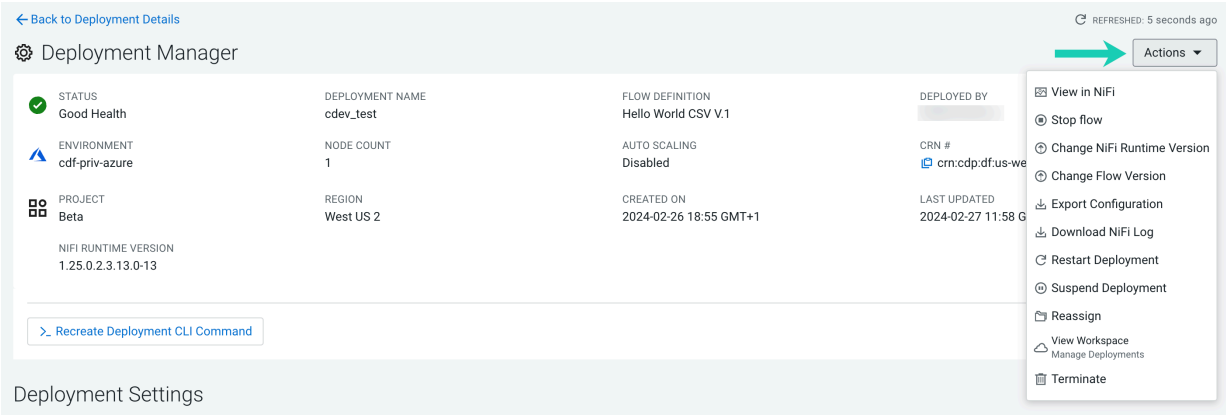

6. Select Export Configuration.
7. Confirm your choice by clicking Export in the pop-up.

Results

The configuration is exported to the {LOG location}/cdf-deployment-backup directory. {LOG location} is configured during the creation of the associated Cloudera Public Cloud Environment. If you want to reuse the exported configuration in a different environment, you can either configure that to use the same {LOG location}, or you can copy the exported .tar.gz and JSON files to the {LOG location}/cdf-deployment-backup directory of the target environment.

What to do next

You can reuse the exported configuration during deployment of the same flow definition to recreate a flow with similar configuration.

Terminating a deployment

You can terminate a deployment to remove it from Cloudera DataFlow.

About this task

If you terminate a deployment, you delete the associated NiFi resources and your flow no longer remains active. The associated flow definition remains in the catalog and is available to be deployed again in a new deployment.

Before you begin

You must have deployed a flow definition in Cloudera DataFlow.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.
5. Click Actions in the **Deployment Manager** page.

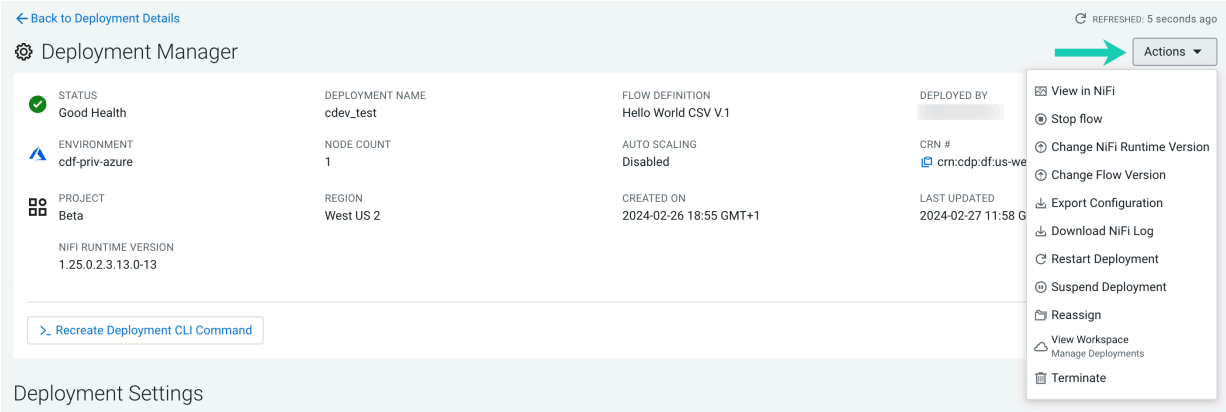

6. Select Terminate.

   The Terminate [Deployment Name] page appears.
7. Enter the name of the deployment to confirm and click Terminate.

Tip: You can also click the copy to clipboard icon to copy the deployment name, paste the name of the deployment, and then click Terminate.

Editing a deployment

You can edit the KPIs, alerts, parameters, size and scale details of your flow deployment under Deployment Settings or using the CDP CLI.

About this task

You can either edit your deployment on the Cloudera DataFlow UI or through the CLI.

Before you begin

- You must have the DFFlowAdmin role assigned to perform this task.
- You must have deployed a flow definition in Cloudera DataFlow.

Editing deployments on the user interface

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Deployments from the left navigation pane.

3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.
4. In Deployment Details, click Actions Manage Deployment .

   You are redirected to the **Deployment Manager** page.
5. Select KPIs and Alerts to edit KPIs or the alerts they trigger.
6. Select Sizing and Scaling
   - to change the size and number of NiFi nodes provisioned for your flow deployment
   - to toggle auto-scaling
   - to view storage type
7. Select Parameters to change flow deployment parameter values.

   Tip:

   Check the UI for CLI usage examples to help you learn to automate your common Cloudera DataFlow tasks.

   You can confirm by finding the Deployment Update success, in Alerts under **Deployment Details**.
8. Click Apply Changes before exiting each tab.

Editing a deployment using the CLI

Before you begin

- You have installed CDP CLI.
- You have run df list-deployments. The output includes the crn field containing the deployment-crn and the service.crn field containing the environment-crn.
- To obtain the configuration-version, you have run the following command, the output of which contains the configurationVersion field containing the configuration-version value:

```
dfworkload get-deployment-configuration
--environment-crn [***ENVIRONMENT_CRN***]
--deployment-crn [***DEPLOYMENT_CRN***]
```

Procedure

- To edit an existing flow deployment, use the `cdp dfworkload update-deployment` command.

```
cdp dfworkload update-deployment
          --environment-crn <value>
          --configuration-version <value>
          --deployment-crn <value>
          [--parameter-groups <value>]
          [--auto-scaling-enabled | --no-auto-scaling-enabled]
          [--auto-scale-min-nodes <value>]
          [--auto-scale-max-nodes <value>]
          [--static-node-count <value>]
          [--kpis <value>]
          [--asset-update-request-crn <value>]
```

  - [--static-node-count] – Specifies the number of NiFi nodes when autoscaling is not enabled. You can select between 1 and 32 nodes. The default value is 1.

- [--auto-scaling-enabled | --no-auto-scaling-enabled] – Specifies whether you want to enable autoscaling. The default is to disable autoscaling.
  - [--auto-scale-min-nodes] – Specifies the minimum nodes when you have autoscaling enabled. If you have autoscaling enabled, this parameter is required.
  - [--auto-scale-max-nodes] – Specifies the maximum nodes when autoscaling is enabled. If you have autoscaling enabled, this parameter is required.
- [--parameter-groups] – Specifies the location of the parameter group JSON file you, if you are using one for this flow deployment.
- [--kpis] – Specifies the location of the KPIs JSON file, if you are providing KPIs for this flow.

Example

Example of updating your static node count:

```
cdp dfworkload update-deployment \
    --environment-crn $ENVIRONMENT_CRN \
    --deployment-crn $DEPLOYMENT_CRN \
    --configuration-version $CONFIGURATION_VERSION \
    --static-node-count 2
```

Example

Example of how to use a .json file to update parameter values:

```
cdp dfworkload update-deployment \
    --environment-crn $ENVIRONMENT_CRN \
    --deployment-crn $DEPLOYMENT_CRN \
    --configuration-version $CONFIGURATION_VERSION \
    --parameter-groups file:///tmp/parameter-groups.json
```

Cloudera DataFlow

Monitoring Flow Deployments

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to monitoring flow deployments....4

Monitor health of deployments....4

Monitor performance of deployments....6
Key performance indicators for NiFi flow deployments....6
System metrics for NiFi flow deployments....7
Alerts for NiFi flow deployments....8

Monitor auto-scaling events....9

Key performance indicators....10

Working with KPIs....11
Use KPIs to monitor whether your data flow is processing data....11
Use KPIs to track the processing latency of streaming data....13
Use KPIs to monitor connection usage....15
Use KPIs to monitor individual processor metrics....17
Use KPIs to monitor auto-scaling operations....18

KPI scope types in Cloudera DataFlow....19
Connection KPIs....20
Processor KPIs....20
Process Group KPIs....22
Entire Flow KPIs....23
System KPIs....24

Reporting tasks....24
Reporting task JSON file reference....26

Introduction to monitoring flow deployments

The Cloudera DataFlow UI is the central monitoring console for all your deployments across environments. It is the space where you can monitor flow metrics and infrastructure usage, and manage deployments. , You can enable receiving event notifications on the Cloudera Management Console. You can also use reporting tasks to monitor flow deployments using external tools.

In the Cloudera DataFlow UI, you can monitor the status of your deployment and whether the deployment meets performance expectations. You can filter and sort your deployments based on your requirements. You can also fetch details for your deployments for a specified time period and refresh deployment data.

You can monitor the key performance indicators (KPIs), system metrics, and alerts set for your deployments in the Cloudera DataFlow UI. You get alerts for deployments breaching certain assigned KPIs and get recommendations about how to tune your deployments. You also receive alerts if there are issues in your deployment.

After you deploy a flow definition, you can monitor the deployment on the **Deployments** page.

You can configure to receive notifications on the Cloudera Management Console and also in emails, on events for all deployments in an environment.

You can also create reporting tasks to monitor health and performance of your flow deployments using external systems.

Monitor health of deployments

You can monitor the health of your deployments in the Deployments page of the Cloudera DataFlow UI. By monitoring health of deployments, you can track the status of the deployment, the amount of data received and sent by the deployment.

You can perform the following tasks in the **Deployments** page:

- Monitor health of deployments
- Filter and sort deployments
- Fetch metrics details for deployments
- Refresh deployments data

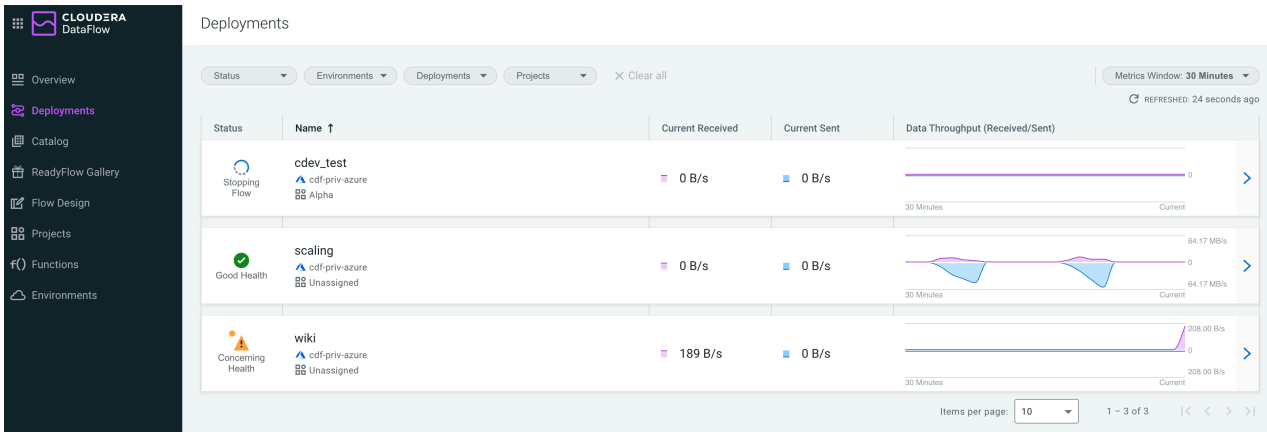

Monitor health of deployments

Monitor the status, amount of data sent and received details for your deployments. You can also view the details of the amount of data sent and received in a graphical format.

You can monitor the following details of your deployments in the Deployments page:

- Status

  Shows status of your deployments including Good Health, Terminating, Updating, Unknown, Deploying, Suspended, Concerning Health, and Bad Health. You can also click Status to sort the status of your deployments as required.
- Name

  Shows name of your deployments. You can also click Name to sort the names of your deployments alphabetically if required.
- Current Received

  Shows the amount of data that your deployments receive from the source. You can also click Current Received to sort your deployments in increasing or decreasing amounts of data received by the deployments.
- Current sent Sent

  Shows the amount of data that your deployments send to the targets. You can also click Current Sent to sort your deployments in increasing or decreasing amounts of data sent by the deployments.
- Data Throughput (Received/Sent)

  Shows graphical representation of data received and sent by your deployments.

Filter and sort deployments

Filter the deployments according to your requirement.

You can filter your deployments by:

- Status

  Status of deployments can be of many types including Bad Health, Concerning Health, Good Health, Deploying, Starting, Suspending, Suspended, Terminating, Unknown, and Updating. By default, all statuses are selected.
- Environments

  You can either select all environments or a specific environment which is enabled. By default, all environments are selected.
- Deployments

  You can use this filter to search your flow deployments by name. You can filter your deployments by either selecting all deployments or a specific deployment. By default, all deployments are selected.
- Projects

  You can filter your deployments by project affiliation.

Click Clear all to set the filter options to their default values.

You can sort deployments by:

- Status
- Name
- Current Received
- Current Sent

Click on the column name to sort by ascending or descending order.

Use the Items per page option, at the bottom of the screen, to display the desired number of deployments in a page.

Fetch metrics details for deployments

The Metrics Window option enables you to fetch details for your deployments for a specified time. For example, if you select 30 minutes in the Metrics Window, the Cloudera DataFlow UI displays details of your deployments for the last 30 minutes, and if you select 1 hour, the UI displays details of your deployments for the last one hour. You can also select 12 or 24 hours based on your requirement.

Refresh deployments data

By default, every 30 seconds all data for your deployments are updated. Click the circle-arrow icon to refresh the data.

Monitor performance of deployments

You can monitor performance of your deployments in the Deployments page of the Cloudera DataFlow UI.

To monitor performance, you can monitor the KPIs, system metrics, and alerts set for your deployments in the Cloudera DataFlow UI. Click on each deployment to get the details of KPIs, system metrics, and alerts. Alternately, you can click on the arrow placed at the end of each deployment row.

Key performance indicators for NiFi flow deployments

Apache NiFi has multiple metrics to monitor the system such as memory usage, CPU usage, data flow statistics, and so on. Key Performance Indicators (KPIs) are representations of those metrics for a NiFi component in Cloudera DataFlow. You can use KPIs to monitor critical parts of your NiFi deployments on the central monitoring dashboard of Cloudera DataFlow.

For more information on KPIs, see *KPI Overview* and *Using KPIs*.

Click the deployment to view the deployment details.

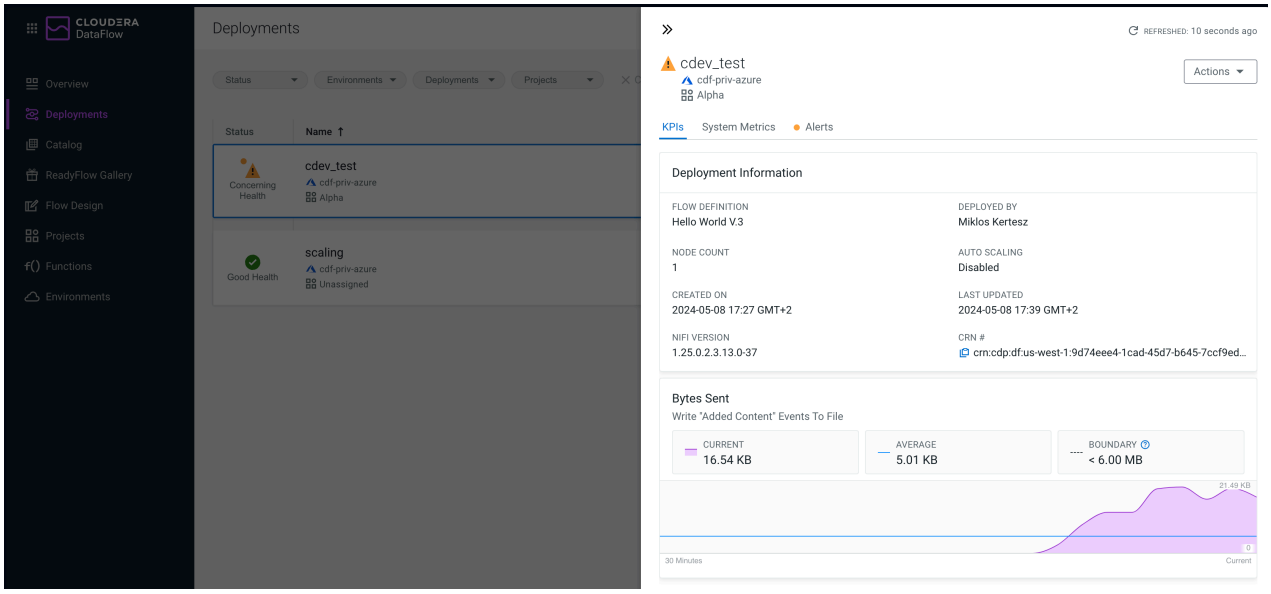

On the KPIs tab of the Deployment Details pane, you can view the deployment information and monitor the KPIs that are defined for your deployment.

You can view the following details in Deployment Information section:

- Flow Definition
- Node Count
- Auto Scaling
- Last Updated
- NiFi Version
- CRN

Depending on the KPIs defined, you can track the following values of the metrics in the deployment:

- Current

- Average
- Boundary

Related Information

Introduction to KPIs

Working with KPIs

System metrics for NiFi flow deployments

Cloudera DataFlow provides comprehensive NiFi cluster-specific metrics on core allocation, disk capacity utilisation, CPU utilisation, and disk usage. You can investigate these system metrics to identify and resolve performance issues related to your NiFi deployments.

Everytime you deploy and run a NiFi flow, you create a NiFi cluster. The system metrics are details about the NiFi cluster that provide an opportunity to understand the flow deployment status.

These metrics help you to understand the performance of a flow. For example, a flow running at 50% CPU utilization may be an expected (and fine) behavior compared to another flow running at 5% CPU utilization. You are the best judge for how your flow should perform and what thresholds for these metrics dictate that. One metric that you specifically monitor is CPU Utilization. If the value goes above the threshold of 80%, you must scale the cluster.

You can set the key performance indicators (KPIs), with knowledge of your flows, which allow alerts to be sent if certain metrics meet a particular threshold. You might detect situations that are not errors but notes on the state of the system that you need to be aware of. For example, a cluster size reaching the maximum value that is allowed.

The values you measure for metrics depend on the hardware chosen during the flow deployment. For example, the NiFi node size you select (Extra Small to Large) during the flow deployment. These selections enable a deployment to access CPU, memory, or storage which can cause numbers measured to change.

In the Cloudera DataFlow web interface, go to **Deployments** and click a deployment to view core allocation, disk capacity, CPU utilization, and disk usage metrics on the System Metrics tab of the Deployment Details pane.

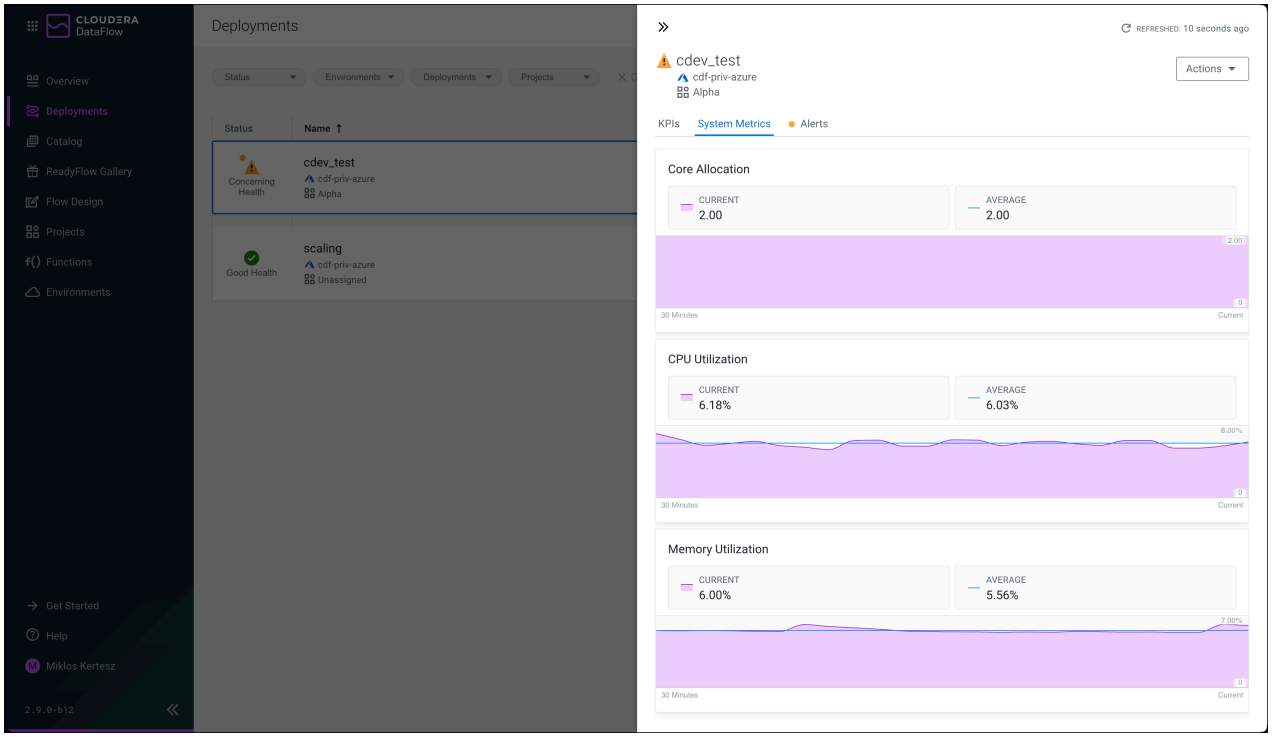

- Core Allocation

  Description: The number of cores that are currently allocated to NiFi containers for the deployment.

  Unit: vCores

  Functionality: Monitors current and average core allocation. You can move your mouse on the graph and get core allocation details for a specific point in time.

  Importance: Helps to assess if the number of cores a flow is using is within the expected range.

  Example: Together with the size selected when the NiFi flow is deployed, the metric reflects the real size of the cluster in terms of allocated CPU resources. For example, when Extra Small size is selected, each NiFi node is capped at 2 vCores. Therefore, a value of 6 means 3 NiFi nodes are currently running to execute the flow. The value fluctuates when the NiFi cluster is auto-scaled up or down.
- Disk Capacity

  Description: Percentage of total allocated storage capacity that is being used by the deployment.

  Unit: Percentage

  Functionality: Monitors current and average disk capacity. You can move your mouse on the graph and get disk capacity details of a specific point in time.

  Importance: Helps to identify if a disk is being under utilized or over utilized; however, again it depends on the expected behavior of a flow.

  Example: This value reflects the storage pressure of the NiFi flow, including Content, Provence, and Flow File repositories. A high percentage can indicate that the traffic is being set in the NiFi cluster, for example, due to failure in delivering the output to the destination system.
- CPU Utilization

  Description: The ratio of the total CPU time consumed by the deployment in the past 5 minutes over the total allocated CPU quota for the deployment.

  Unit: Percentage

  Functionality: Monitors current and average CPU utilization. You can move your mouse on the graph and get the CPU utilization details for a specific point in time.

  Importance: Helps to identify if a flow is using more or less CPU than expected.

  Example: The value indicates the computation pressure of the NiFi cluster running the deployed flow. In a typical case, when auto-scaling is enabled and the CPU utilization reaches the threshold internally defined by Cloudera DataFlow more NiFi instances are brought up and join the cluster to share the workload, before the value starts dropping.
- Disk Usage

  Description: The total amount of bytes saved on the storage volumes of the deployment.

  Unit: GB, MB, and so on depending on the scale of the value

  Functionality: Monitors current and average disk usage. You can move your mouse on the graph and get the disk usage details for a specific point in time.

  Importance: Provides visibility of how much disk is being used by the deployment or Kubernetes node. It can be used to assess what is utilizing a lot of space.

  Example: Enables you to know the current absolute amount of on-disk data of the whole deployment, which may indicate how long an offloading can take.

Alerts for NiFi flow deployments

You can set alerts while configuring KPIs in your deployment. If you set an alert for a KPI and metric combination, you get alerted if your flow definition meets the conditions set in that alert.

Click the deployment to view the deployment details.

On the Alerts tab of the Deployment Details pane, you can monitor the active alerts and event history details.

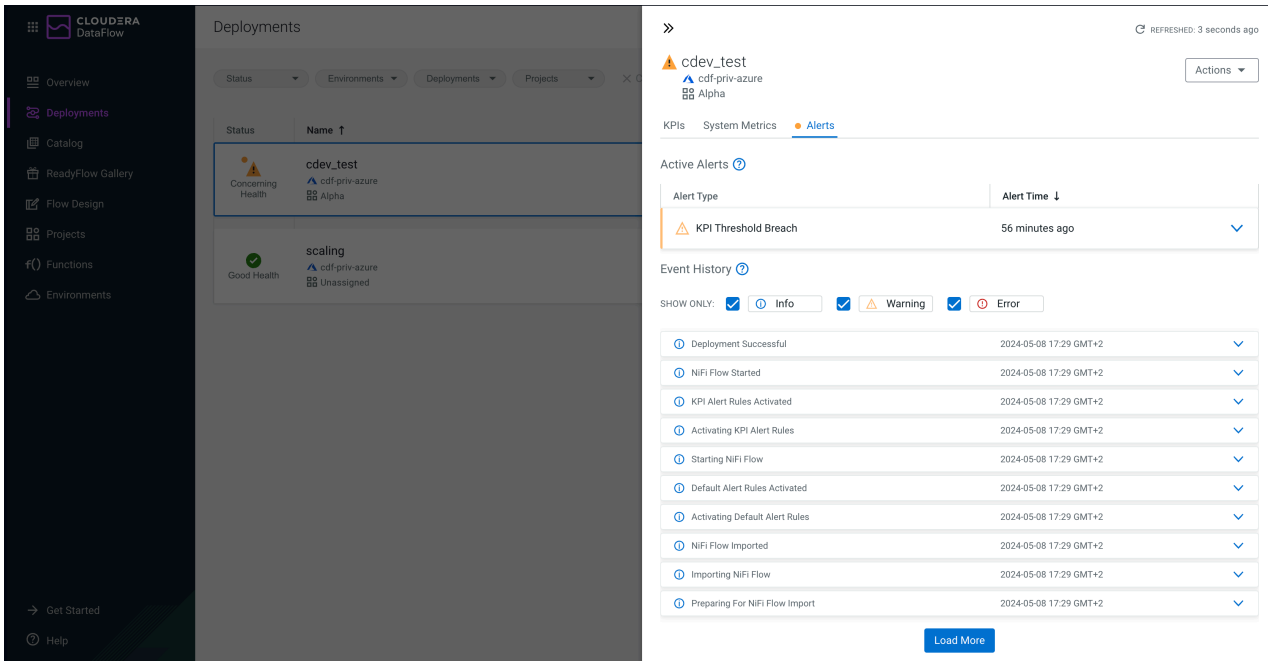

In the Active Alerts section, you can track the alert types and time of the alert. If you click on the alert you want to track, you can see the alert details and the exact time when the alert occurred. Active alerts notify you of a current condition that may require attention. If you set an alert for a deployment-related KPI, you get alerted here as well. Active alerts automatically become inactive and move to the Event History section if the condition is not met anymore.

Tip:

KPI Alerts are suspended when a flow deployment is suspended. They are reactivated when the flow deployment is restarted. Suspending a flow deployment also resolves any Active KPI alerts.

In the Event History section, you can track event-related information and past alert conditions such as when the deployment was initiated and whether the deployment was successful. You can filter the event history list to only show events of a specific type including information, warning, and error.

Monitor auto-scaling events

Every auto-scaling operation is logged as a flow deployment event. These events can be monitored through either the UI or the CDP CLI.

For UI

In the **Deployments**, click on the flow deployment that you want to monitor and perform the following steps:

1. In the side panel displaying Deployment Information, click on the Alerts tab.
2. Look for Deployment Pod Count Changed events.
3. Filter for informational events by only selecting the Info category.
4. Click Load More to see older events.

For CLI

To monitor auto-scaling events via the CDP CLI, execute the `cdp df list-deployment-events` command with the following configuration:

```
cdp df list-deployment-events --deployment-crn [***YOUR DEPLOYMENT CRN***]
```

Replace *[***YOUR DEPLOYMENT CRN***]* with the actual CRN of the deployment for which you want to monitor auto-scale events.

The command returns all deployment events in a JSON array that you can parse for Deployment Pod Count Cha nged events:


```
{
    "eventSummaries": [
        {
            "crn": "crn:cdp:df:us-west-1:558bc1d2-8867-4357-8524-311d512
59233:event:DEPLOYMENT/36761c1d-264b-46ee-8a20-34ccedf2bc90/042cc645-0d8
e-4d1e-aebd-2c90e665f845",
            "name": "Deployment Pod Count Changed",
            "severity": "INFO",
            "firstOccurrence": 1687314605747,
            "eventType": "STATUS_UPDATE"
        },
        {
            "crn": "crn:cdp:df:us-west-1:558bc1d2-8867-4357-8524-311d5125
9233:event:DEPLOYMENT/36761c1d-264b-46ee-8a20-34ccedf2bc90/20c9361f-d33e
-481a-98c2-0b62e3666798",
            "name": "Deployment Successful",
            "severity": "INFO",
            "firstOccurrence": 1687314062456,
            "eventType": "STATUS_UPDATE"
        }
     ]
}
```


Key performance indicators

You can use key performance indicators (KPIs) to monitor critical parts of your Apache NiFi deployments on the central monitoring dashboard. You do not need to drill deep into the NiFi data flow and find the metric to monitor it in NiFi. You can also choose to create alerts for your KPIs in Cloudera DataFlow.

Apache NiFi has multiple metrics to monitor for the system such as memory usage, CPU usage, data flow statistics and so on. KPIs are representations of those metrics for a NiFi component in Cloudera DataFlow.

You can set alerts while configuring KPIs in your deployment. If you set an alert for a KPI scope and metric combination, you get alerted if your flow definition meets the conditions set in that alert.

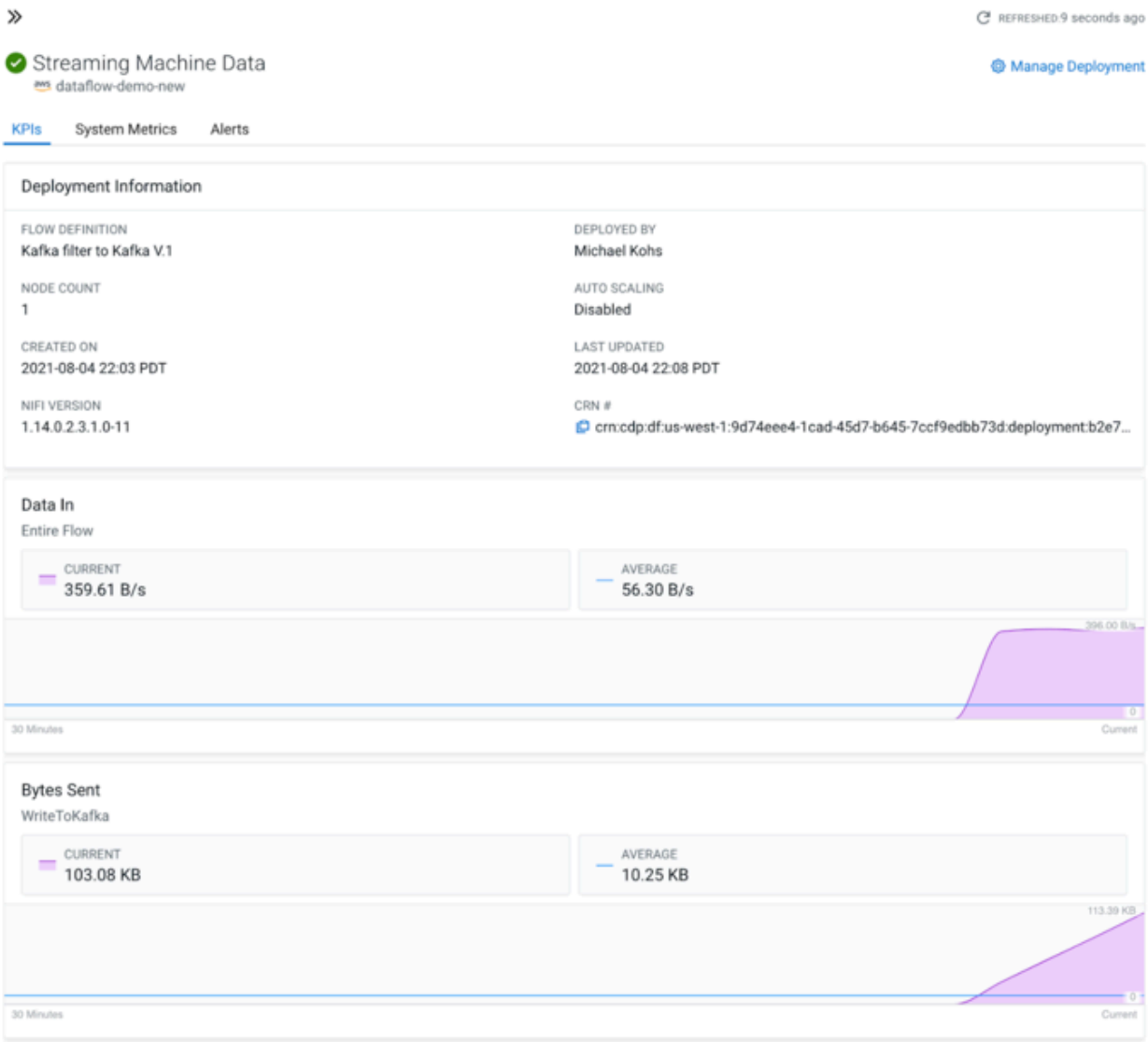

In the above image, a KPI is set for the Entire Flow KPI scope and Data In metric combination. When the data into the entire flow exceeds the 1MB boundary, an alert is created. No alert boundary is set for the Processor KPI scope and Average Linear Duration metric combination.

Working with KPIs

Learn multiple scenarios where you can view how to use KPIs to monitor critical parts of your NiFi deployments on the central monitoring dashboard.

Use KPIs to monitor whether your data flow is processing data

Learn how to use KPIs to monitor whether your data flow is processing data.

Scenario

You want to track how much data your flow is receiving from external systems or processing to an external system. It is a common use case where you might want to track whether your flow is reading or writing data as expected.

Solution

You can achieve your goal by selecting appropriate KPI scope and metric while deploying your flow in Cloudera DataFlow. Select Entire Flow KPI scope and Data In or Data Out metric to track. Select Data In metric to track how much data your flow is reading and Data Out metric to track how much data your flow is writing to the external system. In this example, Data In metric is selected. The Data In metric tracks rate of data received from an external source in bytes.

Before you begin

You must have reached the Add New KPI screen while deploying your flow in Cloudera DataFlow.

Procedure

1. Select Entire Flow as the KPI scope.
2. Select Data In as the metric to track.
3. Configure the alert settings as per your requirement or when you want to be alerted.
4. Click Apply.

   Similarly, you can set other metrics and alerts as per your requirement.

Example

In this example, you get an alert when your flow is receiving less than 2 MB data per second for 5 minutes.

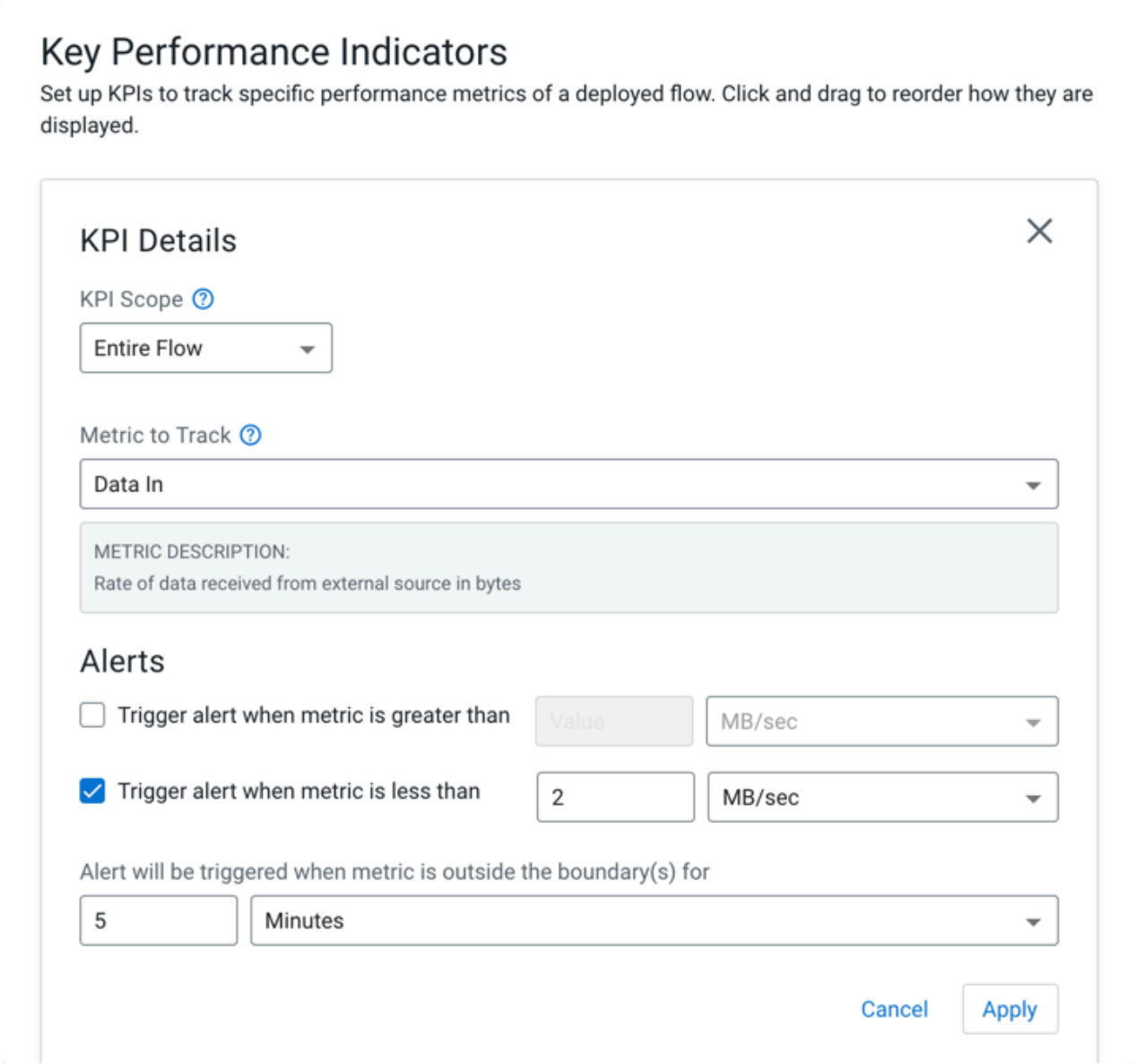

Use KPIs to track the processing latency of streaming data

Learn how to use KPI to track the processing latency of streaming data.

Scenario

Your flow starts reading data from an external system and ends writing the data to some other place. In between all processing is done. A common question or a common performance indicator that you want to track might be how long does it take for the data to be processed.

Solution

You can achieve your goal by selecting appropriate KPI scope and metric while deploying your flow in Cloudera DataFlow. Select Processor KPI scope and Average Lineage Duration metric to track. The Average Lineage Duration metric tracks the time elapsed between when the original source data was received and the time when the processing event occurred.

Note: Average Lineage Duration does not always measure processing latency of a flowfile from the very first processor to the last processor.

If you have a simple NiFi flow, where none of the processors creates a new flowfile in between, Average Lineage Duration measures the processing latency of the flowfile from the first to the last processor.

But, if a processor in your NiFi flow creates a new flowfile in between, Average Lineage Duration measures the processing latency from that very processor, where a new flow file is created, to the last processor.

Before you begin

You must have reached the Add New KPI screen while deploying your flow in Cloudera DataFlow.

Procedure

1. Select Processor as the KPI scope.
2. Select the processor, you want to monitor, in the Processor Name field.
3. Select Average Lineage Duration as the metric to track.
4. Configure the alert settings as per your requirement or when you want to be alerted.
5. Click Apply.

   Similarly, you can set other metrics and alerts as per your requirement.

Example

In this example, Processor KPI scope and Average Lineage Duration metric to track are selected. No alert is set here.

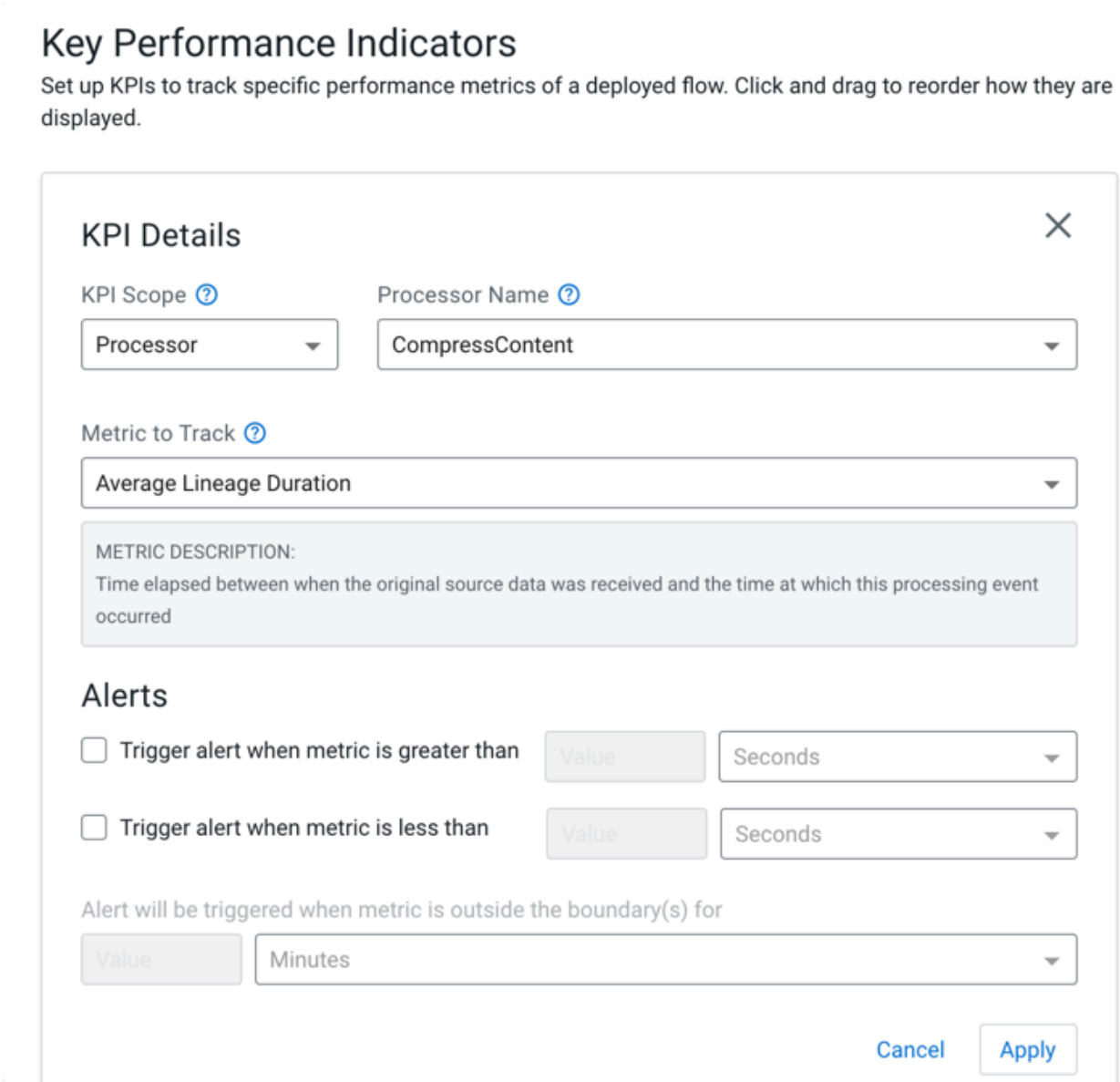

Use KPIs to monitor connection usage

Learn how to use KPIs to monitor connection usage.

Scenario

You want to track errors in a failure queue. You read data from a Kafka topic and apply a schema to it. The schema does not fit the message and the operation fails. You do not want to process the message and route it to a failure queue. Now, you want to track data quality in your deployment.

Solution

You can achieve your goal by selecting appropriate KPI scope and metric while deploying your flow in Cloudera DataFlow. You need to create a KPI on this queue to track failure. Select Connection KPI scope, connection name, and Percent Full metric to track. The Percent Full metric indicates how much of the queue's capacity is used for your data flow.

Before you begin

You must have reached the Add New KPI screen while deploying your flow in Cloudera DataFlow.

Procedure

1. Select Connection as the KPI scope.
2. Select Percent Full as the metric to track.
3. Select the failure queue, you created, in the Connection Name field.
4. Configure the alert settings as per your requirement or when you want to be alerted.
5. Click Apply.

   Similarly, you can set other metrics and alerts as per your requirement.

Example

In this example, you get an alert when 10 percent of the queue's capacity is used for storing failure messages for 5 minutes.

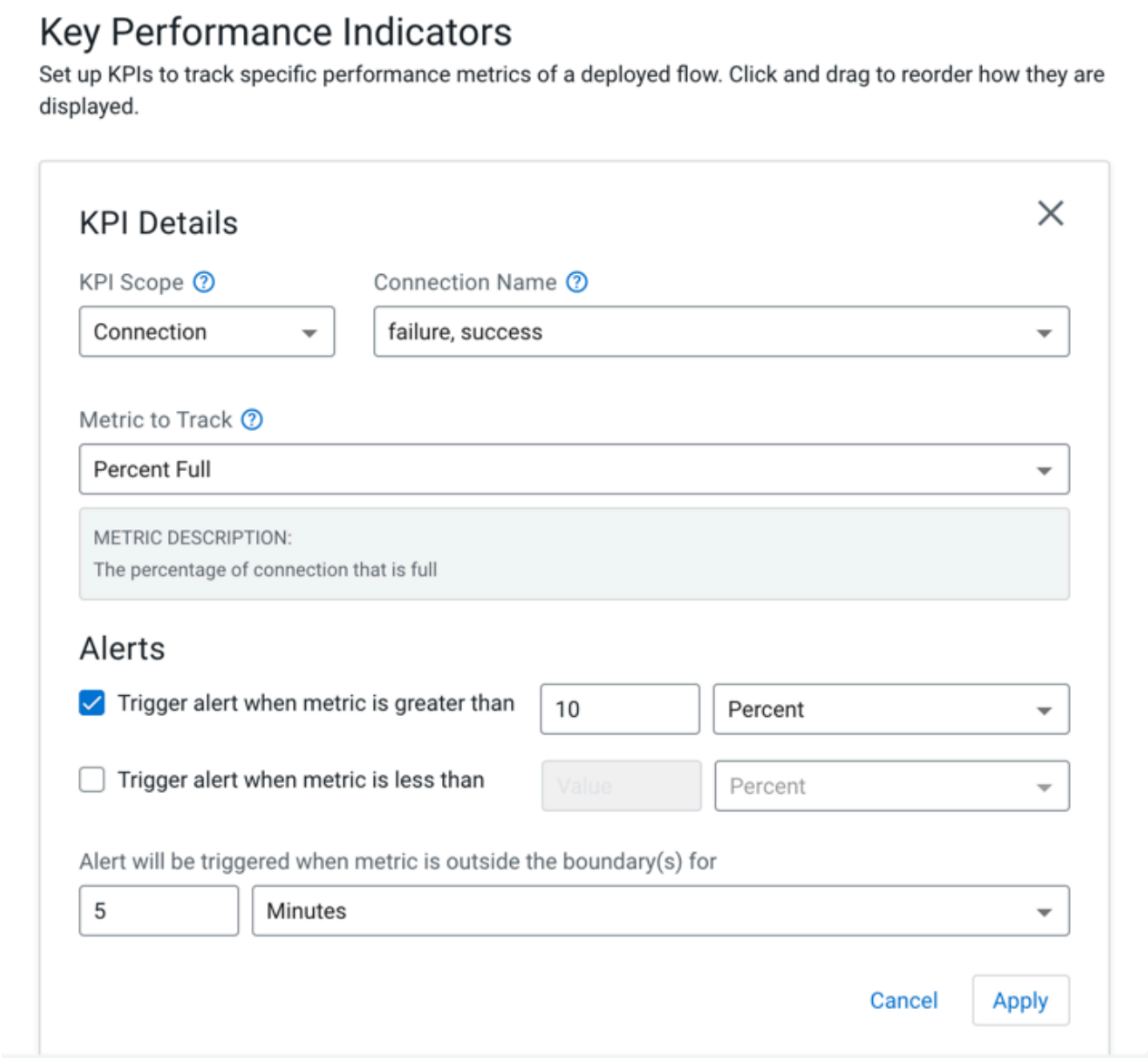

Use KPIs to monitor individual processor metrics

Learn how to use KPIs to monitor individual processor metrics.

Scenario

You want to track a specific metric for a processor in your data flow.

Solution

You can achieve your goal by selecting appropriate KPI scope and metric while deploying your flow in Cloudera DataFlow. Select Processor KPI scope, the processor you want to monitor in the Processor Name field, and the metric you want to track. In this example, Bytes Sent metric is selected. The Bytes Sent metric tracks number of bytes sent to an external recipient.

Before you begin

You must have reached the Add New KPI screen while deploying your flow in Cloudera DataFlow.

Procedure

1. Select Processor as the KPI scope.
2. Select Bytes Sent as the metric to track.
3. Configure the alert settings as per your requirement or when you want to be alerted.
4. Click Apply.

   Similarly, you can set other metrics and alerts as per your requirement.

Example

In this example, Processor KPI scope and Bytes Sent metric to track are selected. No alert is set here.

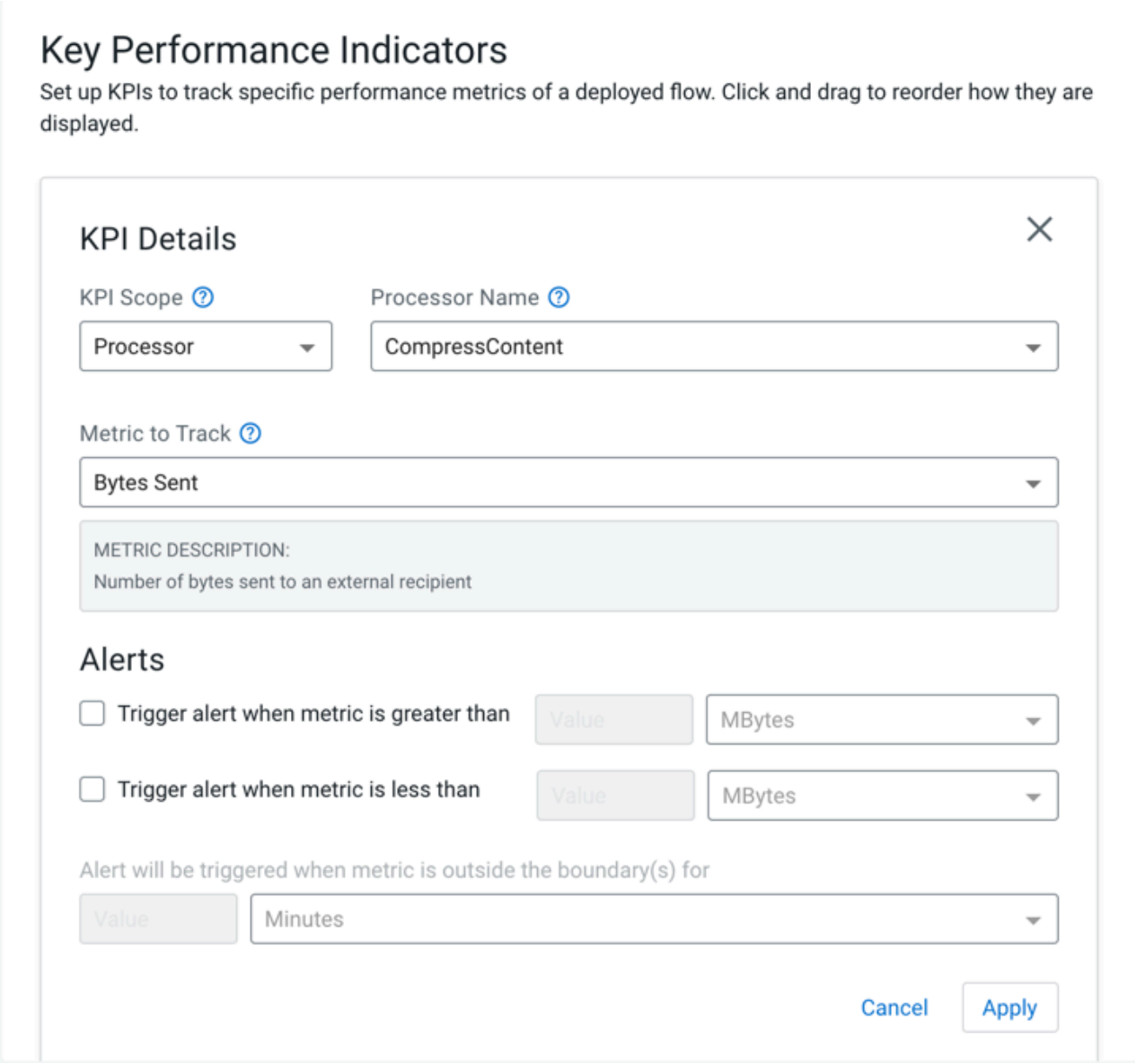

Use KPIs to monitor auto-scaling operations

Learn how to use KPIs to monitor auto-scaling operations.

Scenario

You want to use KPIs to track auto-scaling operations.

Solution

You can achieve your goal by selecting appropriate KPI scope and metric while deploying your flow in Cloudera DataFlow. Select System KPI scope and Core Allocation metric to track. Core Allocation metric tracks the numbers of cores allocated. After the flow deployment, you can view a graph that represents how many cores are used right now and you can also view when it scales up or scales down.

Before you begin

You must have reached the Add New KPI screen while deploying your flow in Cloudera DataFlow.

Procedure

1. Select System as the KPI scope.
2. Select Core Allocation as the metric to track.
3. Configure the alert settings as per your requirement or when you want to be alerted.
4. Click Apply.

   Similarly, you can set other metrics and alerts as per your requirement.

Example

In this example, you get an alert when the number of cores allocated for your flow is greater than seven.

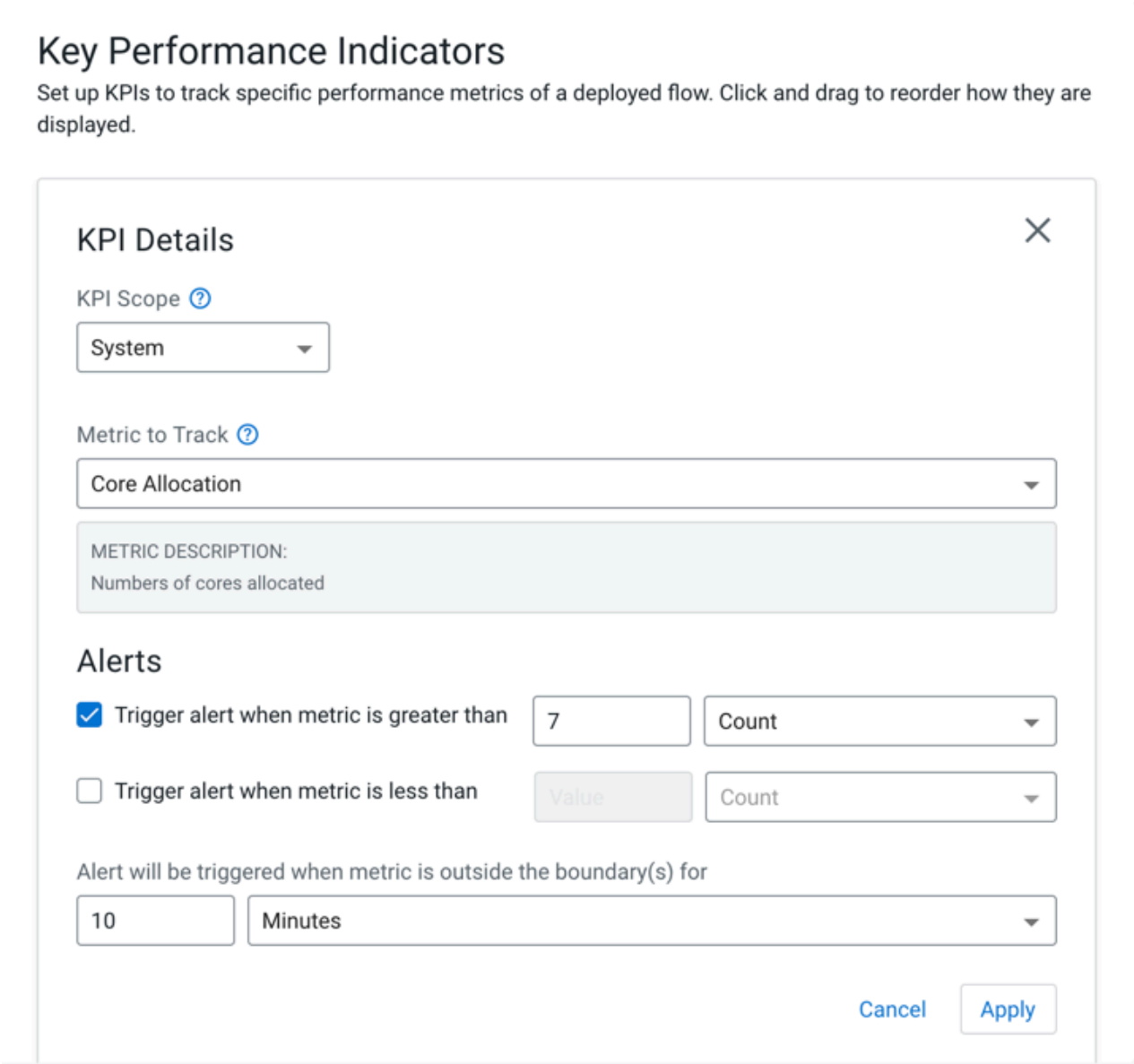

KPI scope types in Cloudera DataFlow

KPI scope specifies the focus area of the KPI for the flow deployment. You can select individual NiFi components such as a processor or connection, or larger components such as a process group or the entire flow.

You can track the different statistics of a NiFi data flow by using the following KPI scopes:

- Connection
- Processor
- Process Group
- Entire Flow
- System

Connection KPIs

Connection KPIs expose the metrics that NiFi provides for the connection used in the NiFi data flow. Choosing Connection allows you to set alerts based on any of the connection metrics that the data flow monitors.

Metric name: Bytes Queued

Metric unit: Bytes

Meaning: How much data is stored in a queue?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the data queued in the selected queue or connection is either more than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Metric name: Percent Full

Metric unit: Percentage

Meaning: How much of the queue's capacity is used?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the data queued in the selected queue or connection is either more than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Metric name: Flow Files Queued

Metric unit: Count

Meaning: How many flow files are queued in this connection?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles queued for the selected queue or connection is either higher than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Processor KPIs

Processor KPIs expose the metrics that NiFi provides for the processors. Choosing Processor allows you to set alerts based on any of the processor metrics that the data flow monitors.

Metric name: Average Lineage Duration

Metric unit: Seconds

Meaning: How much time elapsed between when the original source data was received and the time at which this processing event occurred?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the average lineage duration for all flowfiles within the specified time period is either above the upper boundary or below the lower boundary.

Metric name: Bytes In

Metric unit: Bytes

Meaning: How much data is received by the selected processor?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data received by the selected processor is either below the specified lower threshold or above the specified upper threshold for the given time period.

Metric name: Bytes Out

Metric unit: Bytes

Meaning: How much data is sent by the selected processor?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data sent by the selected processor is either below the specified lower threshold or above the specified upper threshold for the given time period.

Metric name: Bytes Queued

Metric unit: Bytes

Meaning: How much content is queued for the selected processor?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the data queued for the selected precessor is either more than what is specified in the upper boundary or less than specified in the lower boundary for the given time period.

Metric name: Bytes Received

Metric unit: Bytes

Meaning: How much data is received from a source that is external to the flows?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the data received or read by the selected precessor is either more than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Metric name: Bytes Sent

Metric unit: Bytes

Meaning: How much data is sent to an external recipient?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the data sent by the selected precessor is either more than what is specified in the upper boundary or less than specified in the lower boundary for the given time period.

Metric name: Flowfiles Input to Processor

Metric unit: Count

Meaning: How many flowfiles are received from a source within the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the number of flowfiles received from a source within the flow is either more than the specified in the upper boundary or less than the specified in the lower boundary for the given time period.

Metric name: Flowfiles Out from Processor

Metric unit: Count

Meaning: How many flowfiles are sent out of the processor?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the number of flowfiles sent out of the processor is either more than the specified in the upper boundary or less than the specified in the lower boundary for the given time period.

Metric name: Flowfiles Received

Metric unit: Count

Meaning: How many flowfiles are received by the selected processor from a source that is external to the flows?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles received by the selected processor is either below the specified lower threshold or above the upper threshold for the given time period.

Metric name: Flowfiles Sent

Metric unit: Count

Meaning: How many flowfiles are sent by the selected processor to an external recipient?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles sent by the selected processor is either below the specified lower threshold or above the upper threshold for the given time period.

Process Group KPIs

Process Group KPIs expose the metrics that NiFi provides for the process groups. Choosing Process Group enables you to set alerts based on any of the process group metrics that the data flow monitors.

Metric name: Bytes Queued

Metric unit: Bytes

Meaning: How much data is queued, in total, in queues of the selected process group?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data queued for the selected process group is either more than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Metric name: Bytes Received

Metric unit: Bytes

Meaning: How much data is received by the selected process group within the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data received by the selected process group is either below the specified lower threshold or above the upper threshold for the given time period.

Metric name: Bytes Sent

Metric unit: Bytes

Meaning: How much data is sent by the selected process group within the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data sent by the selected process group is either below the specified lower threshold or above the specified upper threshold for the given time period.

Metric name: Flowfiles Input

Metric unit: Count

Meaning: How many flowfiles are received by the selected process group within the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles received by the selected process group is either below the specified lower threshold or above the specified upper threshold for the given time period.

Metric name: Flowfiles Out

Metric unit: Count

Meaning: How many flowfiles are sent by the selected process group within the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles sent by the selected process group is either below the specified lower threshold or above the specified upper threshold for the given time period.

Metric name: Flowfiles Queued

Metric unit: Count

Meaning: How many flowfiles are queued, in total, in queues for the selected process group?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles queued for the selected process group is either higher than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Metric name: Flowfiles Received

Metric unit: Count

Meaning: How many flowfiles are received from a source that is external to the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles received from a source that is external to the flow is either above the specified in the upper boundary or below the specified in the lower boundary.

Metric name: Flowfiles Sent

Metric unit: Count

Meaning: How many flowfiles are sent to an external recipient?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles sent to an external recipient is either above the specified in the upper boundary or below the specified in the lower boundary.

Entire Flow KPIs

Entire Flow KPIs expose the metrics that NiFi provides for the entire data flow. Choosing Entire Flow enables you to set alerts based on any of the flow metrics that the data flow monitors.

Metric name: Data In

Metric unit: Bytes/sec

Meaning: How much data did the entire flow receive from external systems per second?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data received for the entire flow is either below the specified lower threshold or above the upper threshold for the given time period.

Metric name: Data Out

Metric unit: Bytes/sec

Meaning: How much data did the entire flow send to external systems per second?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data sent for the entire flow is either below the specified lower threshold or above the upper threshold for the given time period.

Metric name: Flow Files Queued

Metric unit: Count

Meaning: How many flow files are queued in all connections of the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles queued for all connections of the flow is either higher than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

System KPIs

System KPIs expose the metrics that NiFi provides for the system. Choosing System allows you to set alerts based on any of the system metrics that the data flow monitors.

Metric name: CPU Utilization

Metric unit: Percentage

Meaning: How much of the available CPU to the entire cluster is being used by deployment?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the CPU utilization is either above the specified upper threshold or below the specified lower threshold for the given time period. For example, if the threshold is 80% and the time period is 5 minutes, all data points that are being captured within five minutes must show a higher utilization than 80%, to trigger an alert.

Metric name: Core Allocation

Metric unit: Count

Meaning: How many cores are allocated?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the core allocation is either above the specified upper threshold or below the specified lower threshold for the given time period.

Reporting tasks

Reporting tasks allow you to monitor flow deployments using external applications.

Before you begin

- If you want to create a reporting task using a custom NAR, you have added a custom NAR to the flow during deployment.
- You have DFFlowAdmin role for the environment where you want to create the reporting task.

Create a JSON file

To create a reporting task, use the following template:

Replace elements marked *[***SOME VALUE***]* with actual values valid in your environment. You can specify additional properties that may be necessary to your specific use case.

```
{
  "name": "[***REPORTING TASK NAME***]",
  "type": "[***FULLY QUALIFIED CLASS NAME***]",
  "properties": {
    "[***PROPERTY NAME 1***]": "[***PROPERTY VALUE 1***]",
    "[***PROPERTY NAME 2***]": "[***PROPERTY VALUE 2***]"
  },
  "scheduledState": "[***RUNNING|DISABLED***]",
  "schedulingPeriod": "[***SCHEDULING PERIOD***]",
  "schedulingStrategy": "[***TIMER_DRIVEN|CRON_DRIVEN***]"
}
```

To define a schedulingPeriod that corresponds to the TIMER_DRIVEN schedulingStrategy, specify a frequency. For example, "5 min" or "45 sec" or "2 hours".

To define a schedulingPeriod that corresponds to the CRON_DRIVEN schedulingStrategy specify, for example, "* * * * * ?"

By setting scheduledState to RUNNING, the task automatically starts upon creation. Otherwise, it remains stopped.

For example:

```
{
  "name": "My Reporting Task",
  "type": "org.apache.nifi.controller.ControllerStatusReportingTask",
  "properties": {
    "Show Deltas": "true",
    "reporting-granularity": "five-minutes"
  },
  "scheduledState": "RUNNING",
  "schedulingPeriod": "5 min",
  "schedulingStrategy": "TIMER_DRIVEN"
}
```

If there is more than one reporting task with the same class name in the deployment, you can select the required one using a "bundle", as the following example shows:

```
{
  "name": "ControllerStatusReportingTask",
  "type": "org.apache.nifi.controller.ControllerStatusReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-standard-nar",
    "version": "1.23.2.2.3.12.0-7"
  },
  "properties": {
    "Show Deltas": "true",
    "reporting-granularity": "five-minutes"
  },
  "scheduledState": "RUNNING",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN"
}
```

Create a reporting task

Assuming you created a file named reporting-task.json with contents similar to one of the above examples, execute the command:

```
dfworkload create-reporting-task --environment-crn [***ENVIRONMENT CRN***]
 --deployment-crn [***DEPLOYMENT CRN***] --file-path reporting-task.json
```

Replace *[***ENVIRONMENT CRN***]* and *[***DEPLOYMENT CRN***]* with actual values valid in your environment.

List all reporting tasks

To list existing reporting tasks, use the following command:

```
dfworkload list-reporting-tasks --environment-crn [***ENVIRONMENT CRN***] --
deployment-crn [***DEPLOYMENT CRN***]
```

Replace *[***ENVIRONMENT CRN***]* and *[***DEPLOYMENT CRN***]* with actual values valid in your environment.

Delete a reporting task

Note:

The command output of dfworkload list-reporting-tasks includes a "crn" field, which you can use to replace *[***REPORTING TASK CRN***]* in the delete command.

To delete a reporting task, use the following command:

```
dfworkload delete-reporting-task --environment-crn [***ENVIRONMENT CRN***]
 --deployment-crn [***DEPLOYMENT CRN***] --reporting-task-crn [***REPORTING
 TASK CRN***]
```

Replace *[***ENVIRONMENT CRN***]*, *[***DEPLOYMENT CRN***]*, and *[***REPORTING TASK CRN***]* with actual values valid in your environment.

Reporting task JSON file reference

These file samples provide a starting point on how to configure reporting tasks for Cloudera DataFlow. You can use the Cloudera CLI to add the created reporting tasks to Cloudera DataFlow flow deployments.

AmbariReportingTask

```
{
    "name": "AmbariReportingTask",
    "type": "org.apache.nifi.reporting.ambari.AmbariReportingTask",
    "properties": {
      "Application ID": "nifi",
      "Hostname": "${hostname(true)}",
      "Metrics Collector URL": "http://localhost:6188/ws/v1/timeline/metrics
"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "1 min",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
    "name": "AmbariReportingTask",
    "type": "org.apache.nifi.reporting.ambari.AmbariReportingTask",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-ambari-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "Application ID": "nifi",
      "Hostname": "${hostname(true)}",
      "Metrics Collector URL": "http://localhost:6188/ws/v1/timeline/
metrics"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "1 min",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see AmbariReportingTask in the Cloudera DataFlow documentation.

AzureLogAnalyticsProvenanceReportingTask

```
{
    "name": "AzureLogAnalyticsProvenanceReportingTask",
    "type": "org.apache.nifi.reporting.azure.loganalytics.AzureLogAnalytic
sProvenanceReportingTask",
    "properties": {
      "Job Name": "nifi_reporting_job",
      "Platform": "nifi",
      "Application ID": "nifi",
      "Log Analytics URL Endpoint Format": "https://{0}.ods.opinsights.azur
e.com/api/logs?api-version=2016-04-01",
      "include-null-values": "false",
      "Batch Size": "1000",
      "Log Analytics Custom Log Name": "nifiprovenance",
      "Instance ID": "${hostname(true)}",
      "start-position": "beginning-of-stream",
      "Instance URL": "http://${hostname(true)}:8080/nifi"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "5 mins",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "AzureLogAnalyticsProvenanceReportingTask",
    "type": "org.apache.nifi.reporting.azure.loganalytics.AzureLogAnaly
ticsProvenanceReportingTask",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-azure-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "Job Name": "nifi_reporting_job",
      "Platform": "nifi",
      "Application ID": "nifi",
      "Log Analytics URL Endpoint Format": "https://{0}.ods.opinsight
s.azure.com/api/logs?api-version=2016-04-01",
      "include-null-values": "false",
      "Batch Size": "1000",
      "Log Analytics Custom Log Name": "nifiprovenance",
      "Instance ID": "${hostname(true)}",
      "start-position": "beginning-of-stream",
      "Instance URL": "http://${hostname(true)}:8080/nifi"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "5 mins",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see AzureLogAnalyticsProvenanceReportingTask in the Cloudera DataFlow documentation.

AzureLogAnalyticsReportingTask

```
  {
    "name": "AzureLogAnalyticsReportingTask",
    "type": "org.apache.nifi.reporting.azure.loganalytics.AzureLogAnalyti
csReportingTask",
    "properties": {
      "Send JVM Metrics": "false",
      "Job Name": "nifi_reporting_job",
      "Application ID": "nifi",
      "Log Analytics Custom Log Name": "nifimetrics",
      "Instance ID": "${hostname(true)}",
      "Log Analytics URL Endpoint Format": "https://{0}.ods.opinsights.azur
e.com/api/logs?api-version=2016-04-01"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "1 min",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "AzureLogAnalyticsReportingTask",
    "type": "org.apache.nifi.reporting.azure.loganalytics.AzureLogAna
lyticsReportingTask",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-azure-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "Send JVM Metrics": "false",
      "Job Name": "nifi_reporting_job",
      "Application ID": "nifi",
      "Log Analytics Custom Log Name": "nifimetrics",
      "Instance ID": "${hostname(true)}",
      "Log Analytics URL Endpoint Format": "https://{0}.ods.opinsigh
ts.azure.com/api/logs?api-version=2016-04-01"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "1 min",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see AzureLogAnalyticsReportingTask in the Cloudera DataFlow documentation.

ControllerStatusReportingTask

```
{
  "name": "ControllerStatusReportingTask",
  "type": "org.apache.nifi.controller.ControllerStatusReportingTask",
  "properties": {
    "Show Deltas": "true",
    "reporting-granularity": "five-minutes"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "ControllerStatusReportingTask",
  "type": "org.apache.nifi.controller.ControllerStatusReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-standard-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "Show Deltas": "true",
    "reporting-granularity": "five-minutes"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see ControllerStatusReportingTask in the Cloudera DataFlow documentation.

DataDogReportingTask

```
{
  "name": "DataDogReportingTask",
  "type": "org.apache.nifi.reporting.datadog.DataDogReportingTask",
  "properties": {
    "Metrics prefix": "nifi",
    "Datadog transport": "Datadog HTTP",
    "Environment": "dev"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "DataDogReportingTask",
  "type": "org.apache.nifi.reporting.datadog.DataDogReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-datadog-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "Metrics prefix": "nifi",
    "Datadog transport": "Datadog HTTP",
    "Environment": "dev"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see DataDogReportingTask in the Cloudera DataFlow documentation.

MetricsEventReportingTask

```
{
  "name": "MetricsEventReportingTask",
  "type": "org.apache.nifi.reporting.sql.MetricsEventReportingTask",
  "properties": {
    "dbf-default-precision": "10",
    "dbf-default-scale": "0"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "MetricsEventReportingTask",
  "type": "org.apache.nifi.reporting.sql.MetricsEventReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-sql-reporting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "dbf-default-precision": "10",
    "dbf-default-scale": "0"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see MetricsEventReportingTask in the Cloudera DataFlow documentation.

MetricsReportingTask

```
{
  "name": "MetricsReportingTask",
  "type": "org.apache.nifi.metrics.reporting.task.MetricsReportingTask",
  "properties": {},
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "MetricsReportingTask",
  "type": "org.apache.nifi.metrics.reporting.task.MetricsReportingTas
k",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-metrics-reporting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {},
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see MetricsReportingTask in the Cloudera DataFlow documentation.

MonitorDiskUsage

```
{
  "name": "MonitorDiskUsage",
  "type": "org.apache.nifi.controller.MonitorDiskUsage",
  "properties": {
    "Directory Display Name": "Un-Named",
    "Threshold": "80%"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "MonitorDiskUsage",
  "type": "org.apache.nifi.controller.MonitorDiskUsage",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-standard-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "Directory Display Name": "Un-Named",
    "Threshold": "80%"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see MonitorDiskUsage in the Cloudera DataFlow documentation.

MonitorMemory

```
{
  "name": "MonitorMemory",
  "type": "org.apache.nifi.controller.MonitorMemory",
  "properties": {
    "Memory Pool": "G1 Old Gen",
    "Usage Threshold": "65%"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "MonitorMemory",
  "type": "org.apache.nifi.controller.MonitorMemory",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-standard-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "Memory Pool": "G1 Old Gen",
    "Usage Threshold": "65%"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see MonitorMemory in the Cloudera DataFlow documentation.

PrometheusReportingTask

```
{
  "name": "PrometheusReportingTask",
  "type": "org.apache.nifi.reporting.prometheus.PrometheusReportingTask",
  "properties": {
    "prometheus-reporting-task-metrics-endpoint-port": "9092",
    "prometheus-reporting-task-metrics-strategy": "All Components",
    "prometheus-reporting-task-instance-id": "${hostname(true)}",
    "prometheus-reporting-task-client-auth": "No Authentication",
    "prometheus-reporting-task-metrics-send-jvm": "false"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "60 sec",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "PrometheusReportingTask",
    "type": "org.apache.nifi.reporting.prometheus.PrometheusReportin
gTask",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-prometheus-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "prometheus-reporting-task-metrics-endpoint-port": "9092",
      "prometheus-reporting-task-metrics-strategy": "All Components",
      "prometheus-reporting-task-instance-id": "${hostname(true)}",
      "prometheus-reporting-task-client-auth": "No Authentication",
      "prometheus-reporting-task-metrics-send-jvm": "false"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "60 sec",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see PrometheusReportingTask in the Cloudera DataFlow documentation.

QueryNiFiReportingTask

```
{
  "name": "QueryNiFiReportingTask",
  "type": "org.apache.nifi.reporting.sql.QueryNiFiReportingTask",
  "properties": {
    "dbf-default-precision": "10",
    "sql-reporting-include-zero-record-results": "false",
    "dbf-default-scale": "0"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "QueryNiFiReportingTask",
  "type": "org.apache.nifi.reporting.sql.QueryNiFiReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-sql-reporting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "dbf-default-precision": "10",
    "sql-reporting-include-zero-record-results": "false",
    "dbf-default-scale": "0"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see QueryNiFiReportingTask in the Cloudera DataFlow documentation.

ReportLineageToAtlas

```
{
  "name": "ReportLineageToAtlas",
  "type": "org.apache.nifi.atlas.reporting.ReportLineageToAtlas",
  "properties": {
    "atlas-authentication-method": "basic",
    "filesystem-paths-level": "FILE",
    "kafka-kerberos-service-name": "kafka",
    "atlas-connect-timeout": "60 sec",
    "atlas-conf-create": "false",
    "provenance-start-position": "beginning-of-stream",
    "atlas-read-timeout": "60 sec",
    "provenance-batch-size": "1000",
    "aws-s3-model-version": "v2",
    "kafka-security-protocol": "PLAINTEXT",
    "nifi-lineage-strategy": "SimplePath"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "ReportLineageToAtlas",
  "type": "org.apache.nifi.atlas.reporting.ReportLineageToAtlas",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-atlas-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "atlas-authentication-method": "basic",
    "filesystem-paths-level": "FILE",
    "kafka-kerberos-service-name": "kafka",
    "atlas-connect-timeout": "60 sec",
    "atlas-conf-create": "false",
    "provenance-start-position": "beginning-of-stream",
    "atlas-read-timeout": "60 sec",
    "provenance-batch-size": "1000",
    "aws-s3-model-version": "v2",
    "kafka-security-protocol": "PLAINTEXT",
    "nifi-lineage-strategy": "SimplePath"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see ReportLineageToAtlas in the Cloudera DataFlow documentation.

ScriptedReportingTask

```
{
  "name": "ScriptedReportingTask",
  "type": "org.apache.nifi.reporting.script.ScriptedReportingTask",
  "properties": {
    "Script Engine": "ECMAScript"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "ScriptedReportingTask",
  "type": "org.apache.nifi.reporting.script.ScriptedReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-scripting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "Script Engine": "ECMAScript"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see ScriptedReportingTask in the Cloudera DataFlow documentation.

SiteToSiteBulletinReportingTask

```
{
  "name": "SiteToSiteBulletinReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteBulletinReportingTask",
  "properties": {
    "s2s-transport-protocol": "RAW",
    "Platform": "nifi",
    "include-null-values": "false",
    "Compress Events": "true",
    "Communications Timeout": "30 secs",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "1 min",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "SiteToSiteBulletinReportingTask",
    "type": "org.apache.nifi.reporting.SiteToSiteBulletinReportingTa
sk",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-site-to-site-reporting-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "s2s-transport-protocol": "RAW",
      "Platform": "nifi",
      "include-null-values": "false",
      "Compress Events": "true",
      "Communications Timeout": "30 secs",
      "Instance URL": "http://${hostname(true)}:8080/nifi"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "1 min",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see SiteToSiteBulletinReportingTask in the Cloudera DataFlow documentation.

SiteToSiteMetricsReportingTask

```
{
  "name": "SiteToSiteMetricsReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteMetricsReportingTask",
  "properties": {
    "s2s-metrics-format": "ambari-format",
    "s2s-transport-protocol": "RAW",
    "s2s-metrics-application-id": "nifi",
    "s2s-metrics-hostname": "${hostname(true)}",
    "include-null-values": "false",
    "Compress Events": "true",
    "Communications Timeout": "30 secs",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "SiteToSiteMetricsReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteMetricsReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-site-to-site-reporting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "s2s-metrics-format": "ambari-format",
    "s2s-transport-protocol": "RAW",
    "s2s-metrics-application-id": "nifi",
    "s2s-metrics-hostname": "${hostname(true)}",
    "include-null-values": "false",
    "Compress Events": "true",
    "Communications Timeout": "30 secs",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see SiteToSiteMetricsReportingTask in the Cloudera DataFlow documentation.

SiteToSiteProvenanceReportingTask

```
{
  "name": "SiteToSiteProvenanceReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteProvenanceReportingTask",
  "properties": {
    "s2s-transport-protocol": "RAW",
    "Platform": "nifi",
    "include-null-values": "false",
    "Compress Events": "true",
    "Batch Size": "1000",
    "Communications Timeout": "30 secs",
    "start-position": "beginning-of-stream",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "SiteToSiteMetricsReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteMetricsReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-site-to-site-reporting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "s2s-metrics-format": "ambari-format",
    "s2s-transport-protocol": "RAW",
    "s2s-metrics-application-id": "nifi",
    "s2s-metrics-hostname": "${hostname(true)}",
    "include-null-values": "false",
    "Compress Events": "true",
    "Communications Timeout": "30 secs",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see SiteToSiteProvenanceReportingTask in the Cloudera DataFlow documentation.

SiteToSiteStatusReportingTask

```
{
  "name": "SiteToSiteStatusReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteStatusReportingTask",
  "properties": {
    "s2s-transport-protocol": "RAW",
    "Platform": "nifi",
    "Component Name Filter Regex": ".*",
    "include-null-values": "false",
    "Compress Events": "true",
    "Batch Size": "1000",
    "Component Type Filter Regex": "(Processor|ProcessGroup|RemoteProcessG
roup|RootProcessGroup|Connection|InputPort|OutputPort)",
    "Communications Timeout": "30 secs",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "SiteToSiteStatusReportingTask",
    "type": "org.apache.nifi.reporting.SiteToSiteStatusReportingTask",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-site-to-site-reporting-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "s2s-transport-protocol": "RAW",
      "Platform": "nifi",
      "Component Name Filter Regex": ".*",
      "include-null-values": "false",
      "Compress Events": "true",
      "Batch Size": "1000",
      "Component Type Filter Regex": "(Processor|ProcessGroup|Remote
ProcessGroup|RootProcessGroup|Connection|InputPort|OutputPort)",
      "Communications Timeout": "30 secs",
      "Instance URL": "http://${hostname(true)}:8080/nifi"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "5 mins",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see SiteToSiteStatusReportingTask in the Cloudera DataFlow documentation.

StandardGangliaReporter

```
{
  "name": "StandardGangliaReporter",
  "type": "org.apache.nifi.reporting.ganglia.StandardGangliaReporter",
  "properties": {
    "Send JVM Metrics": "false",
    "Port": "8649",
    "Hostname": "localhost"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "StandardGangliaReporter",
    "type": "org.apache.nifi.reporting.ganglia.StandardGangliaReport
er",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-standard-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "Send JVM Metrics": "false",
      "Port": "8649",
      "Hostname": "localhost"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "5 mins",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see StandardGangliaReporter in the Cloudera DataFlow documentation.

Cloudera DataFlow Overview

Date published: 2021-04-06
Date modified: 2024-06-03

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera DataFlow key features..... 4

Cloudera DataFlow key concepts.....5

Cloudera DataFlow Architecture.....7

Comparison of deployments and functions in Cloudera DataFlow..... 8

Cloudera DataFlow key features

Cloudera DataFlow is a cloud-native universal data distribution service powered by Apache NiFi that enables you to connect to any data source, process and deliver data to any destination. For more details on features and functionalities, see the below list.

Flow and resource isolation

Cloudera DataFlow allows you to easily isolate data flows from each other and guarantee a set of resources to each data flow without requiring administrators to create additional NiFi clusters. For each flow deployment, Cloudera DataFlow creates a dedicated, auto-scaling NiFi cluster on the shared Kubernetes resources in an environment. This way flow deployments can scale independently from each other, allowing you to isolate flow deployments and assign resources to deployments as needed.

Flow isolation can be useful when you want to guarantee a set of resources for a specific data flow or when you want to isolate failure domains.

Auto-scaling flow deployments

Cloudera DataFlow offers two types of auto-scaling capabilities for Apache NiFi data flows. Flow deployments may automatically scale up and down based on CPU utilization within the boundaries that are set in the deployment wizard. In this case Cloudera DataFlow scales flow deployments by adding or removing NiFi pods on the Kubernetes cluster as needed, as well as scaling the Kubernetes cluster up or down within boundaries specified during Cloudera DataFlow enablement.

On top of scaling based on CPU utilization, you may also enable Flow Metrics Scaling in the deployment wizard. This feature adds or removes NiFi pods to the Kubernetes cluster based on anticipated traffic on connection(s) where data first enters the flow. Scaling happens automatically, driven by a prediction algorithm, that uses a backpressure prediction metric to forecast how full a queue will be within a predefined period of time. The metric targets only connections which are attached to source processors. You do not need to configure anything during flow deployment. When Flow Metrics Scaling is enabled, both this metric and CPU utilization are considered. Whichever metric calls for higher scale will be obeyed.

Fault tolerant flow deployments

Flow deployments use persistent volumes to store NiFi repositories in a durable way. In case of an instance or pod failure, Cloudera DataFlow automatically spins up new pods and re-attaches the persistent volumes to ensure data processing continues from where it was interrupted.

Quick flow deployment with predefined ReadyFlows

You can quickly deploy a predefined set of data flows with minimal configuration called ReadyFlows. ReadyFlows provide you with an easy way to implement the most common data flow use cases.

Serverless NiFi Flows with Cloudera DataFlow Functions

Cloudera DataFlow Functions allows you to deploy NiFi flows not only as long running auto scaling Kubernetes clusters but also as functions on cloud providers' serverless compute services including AWS Lambda, Azure Functions, and Google Cloud Functions. Cloudera DataFlow Functions targets use cases that do not require always running NiFi flows, enables developers to focus more on business logic and less on operational management, and establishes a true pay for value model with a serverless architecture.

Central monitoring dashboard and KPIs

You can monitor your flow deployments across environments and cloud providers on a single dashboard. You can track important flow performance metrics by defining KPI alerts for your flow deployments.

Universal connectivity

You can connect to any data source or target using NiFi's rich processor library, including on-premise data sources, cloud data storage, cloud data warehouses, log data sources, cloud data analytics services, or cloud business process services.

Role-based access control

You can control which users are entitled to perform actions like enabling the data service, creating new flow deployments or new drafts by assigning predefined roles like Flow Administrator, Flow Developer or Flow User to individual Cloudera users or groups. By creating Projects and assigning resources to them, you can further restrict access to a subset of resources to Cloudera users or groups.

Secure inbound connections

You can easily provision secure, stable, and scalable endpoints, making it easy for any application to send data to flow deployments.

Parameter groups

You can create groups of parameters and share them between data flows. Parameter groups allow you to centrally manage, share and reuse common parameters that your data flows depend on. When developing new data flows, developers and administrators alike can re-use these common parameters for a simplified development and deployment experience.

Continuous integration (CI) / Continuous deployment (CD)

The Cloudera DataFlow service is built with automation in mind. Any action that is performed on the UI can be turned into a CLI statement for automation. Deploying a new NiFi flow is as easy as executing a single CLI command.

Cloudera DataFlow key concepts

Learn about the key concepts and terms used in Cloudera DataFlow.

Catalog

The **Catalog** is where your flow definitions are stored and where you manage the Cloudera DataFlow flow definition lifecycle from import through versioning to deletion. The **Catalog** is also the place from where you can initiate new deployments.

Deployments

The **Deployments** view is the central monitoring component within Cloudera DataFlow showing all flow deployments across environments at a glance. For each flow deployment, you can open the **Deployment Details** pane, which shows you the KPIs you have defined, system metrics, as well as system events and alerts.

Deployment Manager

The Deployment Manager allows you to review and modify flow deployment parameters, settings for size and scaling, and KPI and alert definitions. It also allows you to initiate NiFi version upgrades, access the NiFi canvas of your flow deployments as well as terminate them. Click the Actions Manage Deployment in the Deployment Details pane to access the Deployment Manager.

Environment

Cloudera DataFlow works in the context of Cloudera environments. You can enable the Cloudera DataFlow service for any supported environment you have registered with Cloudera. The enablement process creates the Kubernetes infrastructure required by Cloudera DataFlow and each environment maps to one Kubernetes cluster.

Once Cloudera DataFlow has been enabled for an environment, you can start deploying flow definitions to it.

Flow definition

A flow definition represents the data flow logic developed in Cloudera DataFlow's Flow Designer and published to the **Catalog**; or developed in Apache NiFi and exported by using the Download Flow Definition action on a NiFi process group or the root canvas. Flow definitions typically leverage parameterization to make the flows portable between for example development and production NiFi environments.

To run an existing NiFi data flow in Cloudera DataFlow, you have to export it as a flow definition and upload it to the Cloudera DataFlow **Catalog**.

Flow deployment

A flow deployment represents a NiFi cluster running on Kubernetes and executing a specific flow definition. When you initiate the flow deployment process from the **Catalog**, a deployment wizard helps you turn a flow definition into a flow deployment. When using the wizard, specify your environment, provide configuration parameters, auto-scaling settings and KPI definitions for your flow deployment.

Function

A function is a flow that is uploaded into the Cloudera DataFlow **Catalog** and that can be run in serverless mode by serverless cloud provider services.

KPI

Apache NiFi has multiple metrics to monitor the different statistics of the system such as memory usage, CPU usage, data flow statistics, and so on. Key Performance Indicators (KPIs) are representations of those metrics for a NiFi component in Cloudera DataFlow. They provide a critical monitoring tool for a real-time view into your data flow performance.

NiFi node

In a flow deployment, a NiFi node is a pod provisioned in the underlying Kubernetes (K8s) cluster. It does not directly relate to a Virtual Machine (VM) of the underlying K8s cluster.

You specify the allowed number of VMs in the K8s cluster when you configure the minimum and maximum number of K8s nodes, and autoscaling, while enabling Cloudera DataFlow for a Cloudera environment.

You specify the allowed number of NiFi nodes during flow deployment. These nodes are then provisioned in the K8s cluster on one or several VMs, depending on resource allocation. Auto-scaling of the NiFi nodes of a deployment may or may not trigger the auto-scaling of the underlying K8s cluster, if configured and depending on the current resources allocation.

Parameter group

A parameter group is a set of shared parameters that can be reused within the project it is currently assigned to. Using shared parameter groups facilitates flow development and flow deployment.

Project

A Project is a container for a set of Cloudera DataFlow resources that restricts visibility of resources associated with it.

ReadyFlow

A ReadyFlow is a predefined, out-of-the-box data flow which can be immediately deployed by providing a small set of required parameters.

ReadyFlow Gallery

The **ReadyFlow Gallery** is where you find all available ReadyFlows. To use a ReadyFlow, you need to add it from the **ReadyFlow Gallery** to the **Catalog** and then use it to create a flow deployment.

Resource

Flow deployments, flow drafts, parameter groups, inbound connections, custom NAR configs, and custom Python configs are collectively called resources in Cloudera DataFlow. You can view and manage them from the **Resources** view.

Workspace

The **Workspace** view displays all resources within an Environment, making it easier to switch between them and managing them.

Cloudera DataFlow Architecture

Cloudera DataFlow follows a two-tier architecture where product capabilities like the Dashboard, Catalog and Environment management are hosted on the Cloudera Control Plane while the flow deployments processing your data are provisioned in a Cloudera environment which represents infrastructure in your cloud provider account. Learn more about the service architecture, and how Cloudera DataFlow enables the various service users to achieve their goals.

When you enable Cloudera DataFlow for one of your registered Cloudera environments, Cloudera DataFlow creates and configures the required infrastructure including a Kubernetes cluster, Kubernetes Operators and the Cloudera DataFlow workload application in your cloud account. After Cloudera DataFlow has been successfully enabled for an environment, users can deploy Flow Definitions into this environment. Deploying a Flow Definition creates a dedicated NiFi cluster on Kubernetes allowing you to treat NiFi flows as isolated flow deployments.

Flow deployments run the NiFi flow logic and process data in your cloud account. Therefore data that is being processed by a flow deployment does not traverse the Cloudera Control Plane. Flow deployments send heartbeats containing health and performance information to the Control Plane where this data is visualized and presented in the Dashboard.

The Cloudera DataFlow Functions feature allows you to deploy NiFi flows stored in the Cloudera DataFlow Catalog as functions executed within AWS Lambda, Azure Functions and/or Google Cloud Functions. Leveraging Cloudera DataFlow Functions does not require to have Cloudera DataFlow enabled in a Cloudera environment. When a Cloudera DataFlow function is executed, the function will interact with the Control Plane to retrieve the flow definition and to send monitoring information.

Learn more about the specific details of Cloudera DataFlow architecture from the diagram below.

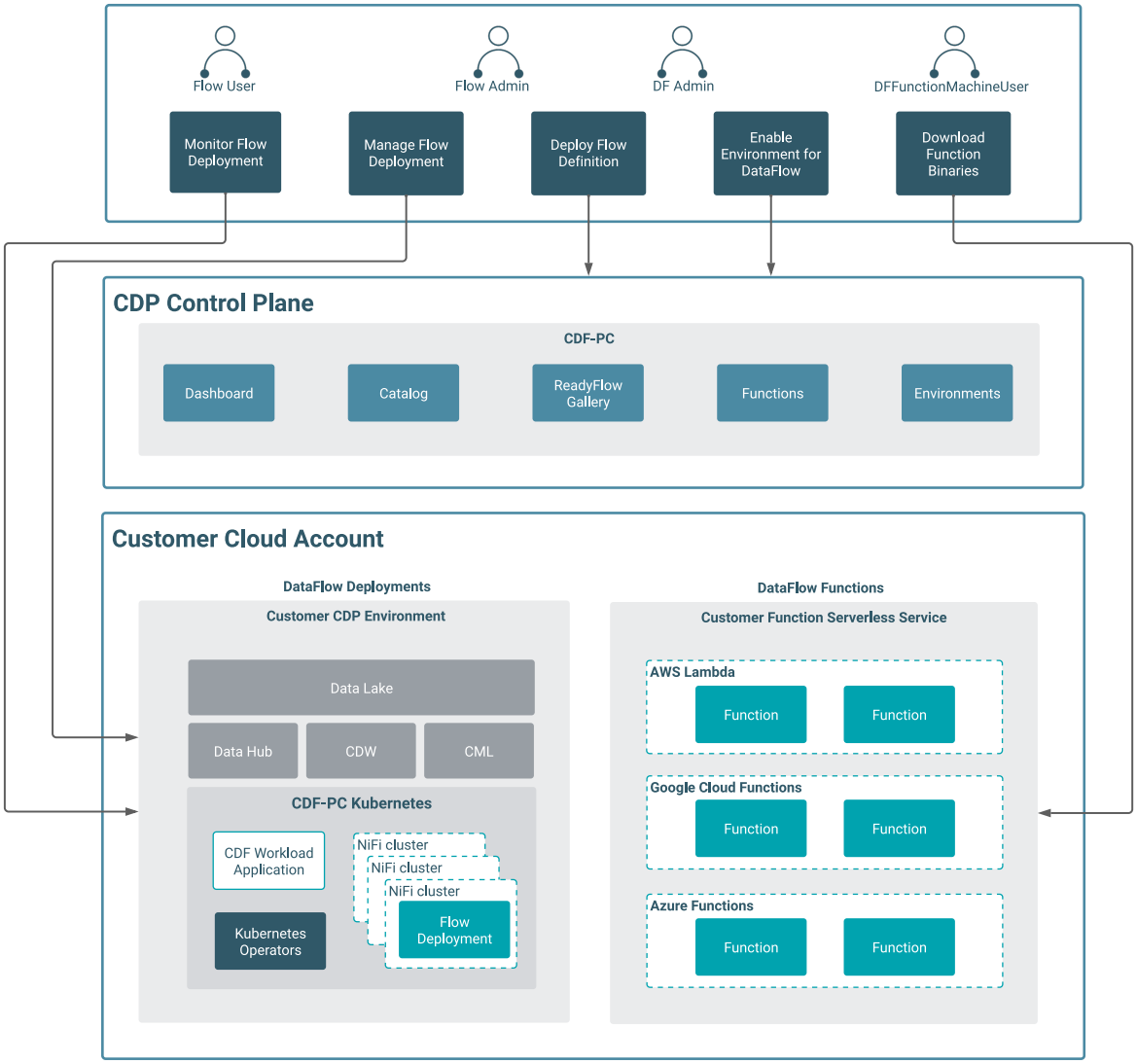

Comparison of deployments and functions in Cloudera DataFlow

This topic explains the differences between deployments and functions in the context of Cloudera DataFlow. It helps you to understand when to use one or the other from a feature as well as from a cost perspective.

Cloudera DataFlow allows you to run NiFi flows on Kubernetes clusters, providing you with an opportunity to run NiFi flows efficiently at scale. However, for a specific set of use cases, it is more beneficial to go one step further by using Cloudera DataFlow Functions. Cloudera DataFlow Functions is a Cloudera DataFlow extension to run NiFi flows as functions for event-driven use cases, where Apache NiFi provides a no-code UI for building and running functions efficiently.

Cloudera DataFlow Deployments provides a cloud-native runtime to run your Apache NiFi flows through auto-scaling Kubernetes clusters. It also provides a centralized monitoring and alerting capability that results in improved Software Development Life Cycle (SDLC) for developers. To run a Cloudera DataFlow deployment, a Kubernetes cluster (or Cloudera DataFlow environment) needs to be provisioned and at least one virtual machine needs to be

always up and running. Additional resources will be provisioned depending on your requirements and according to resource consumption by the running flows. The total cost of ownership (TCO) consists of the cloud provider's and Cloudera's costs. The cloud provider's costs include running the Kubernetes cluster through its native service and are based on the number of virtual machines being used. Cloudera's costs are based on a Concurrent User (CCU) pricing for the running flows which is based on the time period while the flows are running.

Cloudera DataFlow Functions provides a cloud-native runtime to run your Apache NiFi flows as functions on the serverless compute services of three cloud providers (AWS Lambda, Azure Functions, and Google Cloud Functions). It is particularly powerful when the flow does not require NiFi resources to be always up and running. The use cases are event-driven object store processing, microservices that power serverless web applications, IoT data processing, asynchronous API gateway request processing, batch file processing, job automation with cron/timer scheduling, and so on. For these use cases, the NiFi flows need to be treated like jobs with a distinct start and end. The start is based on a trigger event like a file landing in an object store, the start of a cron event, a gateway endpoint being invoked, and so on. Resources are only provisioned by the cloud provider for the duration required by the flow for processing the trigger event. The TCO consists of the cloud provider's costs depending on the amount and duration (in milliseconds) of resources provisioned with the function a and of Cloudera's costs, based on the number of function invocations and the execution time in case the processing of a single event takes more than one second.

	Cloudera DataFlow Deployments	Cloudera DataFlow Functions
Supported cloud providers	AWS, Azure	AWS, Azure, Google Cloud
Runtime	AWS EKS Azure AKS	AWS Lambda Azure Functions Google Cloud Functions
Cloudera pricing	CCU based – Per minute for resources used by the running flows	Per invocation
Supports NiFi clustering	Yes	No
Auto-scaling	Yes – NiFi cluster scales up and down based on CPU consumption. Minimum number of one pod for a given running flow.	Yes – Based on how many concurrent events need to be processed. Multiple instances of the function can be executed at the same time.
Supports all NiFi components	Yes	Yes
Resource limitations	Yes – A given NiFi pod can be provisioned with a maximum of 12vCore and 24 GB of memory	Yes – A function instance cannot be provisioned with more than a given amount of memory and CPU (depends on the cloud provider)
Duration limitations	No – A given flow can be running forever	Yes – A function execution cannot exceed about 15 minutes (depending on the cloud provider)
Access to the NiFi UI	Yes – The NiFi UI can be accessed for a running flow	No – Cloudera DataFlow Functions is powered by Stateless NiFi which does not provide a UI
Serverless	No – There is always a minimum amount of resources up and running (for the Kubernetes cluster and for a running flow)	Yes – No infrastructure to manage and no running resources when no events to be processed
In-memory processing	No – Data going through NiFi requires writes on attached volumes	Yes – Unless specified otherwise, processing is in-memory to improve performances
Multiple sources / multiple destinations	Yes – There is no limit in the flow definition complexity	No – Cloudera DataFlow Functions is designed for simple event processing where a given event is processed and sent to one destination

From a pure technical point of view, any flow that can run as a Cloudera DataFlow function can also run as a Cloudera DataFlow deployment. Choosing one or the other depends on various considerations:

Trigger

Using Cloudera DataFlow Functions assumes that the use case has a trigger mechanism that can be implemented at the cloud provider level. Most of the use cases are covered by the existing triggers but some are not. For example, a flow that listens for events over TCP or UDP do not have a corresponding trigger available.

Cost

There are many parameters to take into account to perform an exhaustive cost analysis between the two options (the flow design, the required resources, the event execution time, and so on), but Cloudera DataFlow Functions is more cost efficient up to one million events processed per month.

SLA

Depending on how the function is configured, a cold start may happen. In a serverless architecture, a cold start refers to the required time for provisioning the resources that will run the function. With Cloudera DataFlow Functions, it is the addition of the time required for provisioning the container and the time required for Stateless NiFi to start and load the components required for executing the NiFi flow. It may go from a few seconds to a minute depending on the function's configuration. A cold start only happens when the function has not been triggered for some time (it depends on the cloud provider). It is also possible to completely remove any cold start at additional cost on the cloud provider's side.

Serverless

Cloudera DataFlow Functions relies on the cloud provider for provisioning resources when the events need to be processed. There are no infrastructure considerations required for upgrades, patches, maintenance, monitoring, and so on. Cloudera DataFlow Functions delegates the scalability to the cloud provider and can virtually scale infinitely to handle very bursty use cases.

Use cases

A few example use cases for Cloudera DataFlow Functions

- Serverless data processing pipelines: Develop and run your data processing pipelines when files are created or updated in any of the cloud object stores (for example: when a photo is uploaded to object storage, a data flow is triggered which runs image resizing code and delivers resized image to different locations to be consumed by web, mobile, and tablets).
- Serverless workflows/orchestration: Chain different low-code functions to build complex workflows (for example: automate the handling of support tickets in a call center).
- Serverless scheduled tasks: Develop and run scheduled tasks without any code on pre-defined timed intervals (for example: offload an external database running on-premises into the cloud once a day every morning at 4:00 a.m.).
- Serverless IOT event processing: Collect, process, and move data from IOT devices with serverless IOT processing endpoints (for example: telemetry data from oil rig sensors that need to be filtered, enriched, and routed to different services are batched every few hours and sent to a cloud storage staging area).
- Serverless Microservices: Build and deploy serverless independent modules that power your applications microservices architecture (for example: event-driven functions for easy communication between thousands of decoupled services that power a ride-sharing application).
- Serverless Web APIs: Easily build endpoints for your web applications with HTTP APIs without any code using Cloudera DataFlow Functions and any of the cloud providers' function triggers (for example: build high performant, scalable web applications across multiple data centers).
- Serverless Customized Triggers: With the Cloudera DataFlow Functions State feature, build flows to create customized triggers allowing access to on-premises or external services (for example: near real time offloading of files from a remote SFTP server).
- Serverless Stream Processing: Easily process messages in real time as they arrive in your messaging queue service (Kinesis, Kafka, Pub/Sub, EventHub, and so on).

For further functions case studies, see AWS Lambda, Azure Functions, and Google Cloud Functions.

You should consider Cloudera DataFlow Deployments over Cloudera DataFlow Functions when:

- A single event (file, request, message, etc) needs to go to multiple destinations. Cloudera DataFlow Functions is better suited for use cases with a single source and a single destination.
- More than 1 million invocations / processing seconds per month are expected. In this case, Cloudera DataFlow Deployments would likely be more cost effective.
- Use cases are Listen based (other than HTTP) such as ListenTCP, ListenUDP, and so on.
- The data needs to be persisted across restarts (for example, the data source is not replayable).
- Buffering / merging of multiple events before sending to destination is required. Cloudera DataFlow Functions provides single event processing.
- Data to be processed is extremely large (multiple GB). While Cloudera DataFlow Functions can also be used, the addition of file store / ephemeral storage will incur additional costs.
- The use case cannot afford a cold start and should always offer very low latency. Cloudera DataFlow Functions can be configured with always running instances at additional cost.
- The workload would benefit from the NiFi clustering and auto-scaling capabilities.
- The user would like to use command and control features of the NiFi UI to stop and update parts of the running flow as well as to monitor the processing of the data in the NiFi UI.
- The user has a large set of use cases and running everything on a single Kubernetes cluster might provide a lower TCO.

Cloudera DataFlow

Using Python scripts

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Running Python scripts in Cloudera DataFlow flow deployments.....4

Upload and run Python scripts in flow deployments.....4

Install custom Python libraries in flow deployments.....5

Running Python scripts in Cloudera DataFlow flow deployments

Relying on Python scripts to perform data transformations within data flows is a common pattern for NiFi users. Cloudera DataFlow flow deployments come with Python 3 and the following custom pre-installed packages: requests, urllib3. You can design your data flows to use the pre-installed Python runtime as well as install additional custom packages which you might require.

Upload and run Python scripts in flow deployments

If running your data flow requires executing a Python script, you have to upload it when creating your data flow deployment through the Deployment Wizard or the CLI. Follow these steps to configure your NiFi processors correctly and upload your Python script.

Procedure

1. Create your Python script and save it as a file.
   For example:

   ```
   #!/usr/bin/python3
   print("Hello, World!")
   ```

2. Open the flow definition which requires a Python script in NiFi.
3. Add and configure an ExecuteStreamCommand processor to run your script.

   Make the following property settings:

   Command Arguments

   provide #{Script}

   Command Path

   provide python

   Leave all other properties with their default values.

   Note: If you need to upload additional supporting files for use by your script, add a dynamic property named Additional Resources referencing a parameter #{AdditionalResources}. The primary script may reference these files through the path /nifi-flow-assets/*[***PARAMETER CONTEXT NAME***]*/*[***PARAMETER NAME***]*/*[***FILE NAME***]*@f0.

4. If you have edited your data flow in NiFi, download it as a flow definition and import it to Cloudera DataFlow. If you have edited your data flow in the Flow Designer, publish the flow to the Catalog.
5. Initiate a flow deployment from the Catalog. In the Parameters step of the Deployment Wizard, upload your Python script to the Script parameter. Upload additional files to the AdditionalResources parameter if applicable. Complete the Wizard and submit your deployment request.

Results

Your Python script is uploaded to the flow deployment and executed as part of the data flow.

Related Information

Add a parameter in Flow Designer

Publish a draft to Catalog as a flow definition

Download a flow definition from NiFi

Import a flow definition to Cloudera DataFlow

Deploy a flow definition in Cloudera DataFlow

Install custom Python libraries in flow deployments

If your data flow requires custom Python packages you can modify your Python script to install these dependencies through the use of NiFi processors.

Procedure

1. Create a Python script, to install the package you want to add:

```
#!/usr/bin/python3
try: import [***PACKAGE NAME***] as [***IMPORT AS***]
except ImportError:
    from pip._internal import main as pip
    pip(['install', '--user', '[***PACKAGE NAME***]'])
    import [***PACKAGE NAME***] as [***IMPORT AS***]
import sys
file = [***IMPORT AS***].read_csv(sys.stdin)
```

   Replace *[***PACKAGE NAME***]* with the name of the package you want to import and *[***IMPORT AS***]* with a meaningful name you want the package to be called in your data flow.

```
#!/usr/bin/python3
try: import pandas as pd
except ImportError:
    from pip._internal import main as pip
    pip(['install', '--user', 'pandas'])
    import pandas as pd
import sys
file = pd.read_csv(sys.stdin)
```

2. Open the flow definition which requires custom packages in NiFi.
3. Add and configure an ExecuteStreamCommand processor to run your script.

   Make the following property settings:

   Command Arguments

   provide #{Script}

   Command Path

   provide python

   Leave all other properties with their default values.

   Note: If you need to upload additional supporting files for use by your script, add a dynamic property named Additional Resources referencing a parameter #{AdditionalResources}. The primary script may reference these files through the path /nifi-flow-assets/*[***PARAMETER CONTEXT NAME***]*/*[***PARAMETER NAME***]*/*[***FILE NAME***]*@f0.
4. If you have edited your data flow in NiFi, download it as a flow definition and import it to Cloudera DataFlow. If you have edited your data flow in the Flow Designer, publish the flow to the Catalog.
5. Initiate a flow deployment from the Catalog. In the Parameters step of the Deployment Wizard, upload your Python script to the Script parameter. Upload additional files to the AdditionalResources parameter if applicable. Complete the Wizard and submit your deployment request.

Results

Your Python script is uploaded to the flow deployment and the required custom libraries are installed when the script is executed as part of the data flow.

Cloudera DataFlow

Quickstart

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera DataFlow Quickstart 4

Verify cloud infrastructure prerequisites 6

AWS quickstart 6

Verify AWS prerequisites 6

Create a CDP credential 7

Register a CDP environment 8

Azure quickstart 15

Verify Azure prerequisites 16

Create an Azure AD app 17

Deploy the template 18

Assign roles 19

Create or locate an SSH Key 20

Create a CDP credential 20

Register a CDP environment 21

Give administrators access 25

Enable Cloudera DataFlow for your environment 26

Give users access 27

Add the Hello World ReadyFlow definition to the Catalog 28

Deploy the Hello World ReadyFlow using the deployment wizard 30

Cloudera DataFlow Quickstart

Get started with Cloudera DataFlow quickly by walking through a few simple steps. Ensure that administrators have access to Cloudera Public Cloud, enable Cloudera DataFlow for an environment, give users access to Cloudera DataFlow, and then add to the Catalog and deploy the Hello World ReadyFlow.

Cloudera DataFlow

Quickstart

1. Verify infrastructure prerequisites
2. Provide access for administrators
3. Enable CDF for your environment
4. Provide access for users
5. Add to Catalog
6. Deploy a flow definition

Verify cloud infrastructure prerequisites

As the administrator for your AWS environment, ensure that the environment meets the requirements for Cloudera Public Cloud and Cloudera DataFlow.

AWS quickstart

If you've reached the CDP landing page for the first time, you've come to the right place! In this quickstart, we'll show you step-by-step how to connect CDP to your AWS account, so that you can begin to provision clusters and workloads.

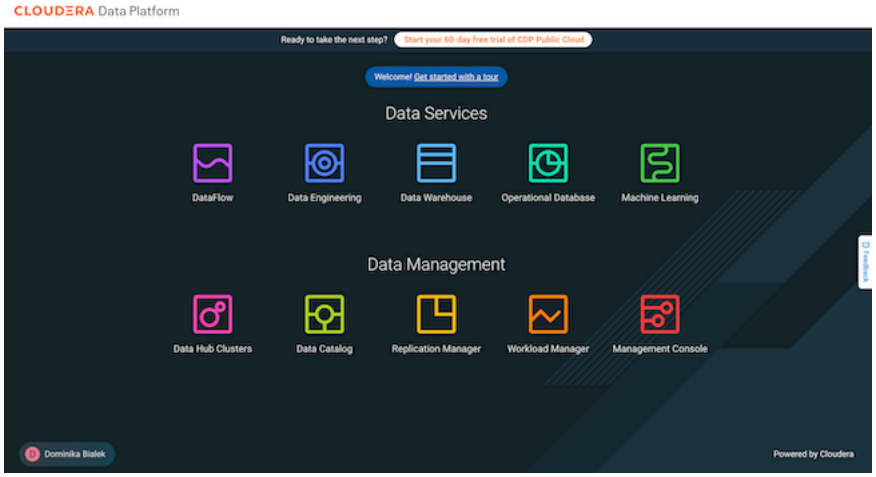

To complete this quickstart, you'll need access to two things:

- The CDP console pictured above
- The AWS console

Note: This AWS onboarding quickstart is intended for simple CDP evaluation deployments only. It may not work for scenarios where AWS resources such as VPC, security group, storage accounts, and so on, are pre-created or AWS accounts have restrictions in place.

The steps that we will perform are:

Step 0: Verify the AWS prerequisites

Step 1: Create a provisioning credential

Step 2: Register an AWS environment in CDP

Verify AWS cloud platform prerequisites

Before getting started with the AWS onboarding quickstart, review and acknowledge the following:

- This AWS onboarding quickstart is intended for simple CDP evaluation deployments only. It may not work for scenarios where AWS resources such as VPC, security group, storage accounts, and so on, are pre-created or AWS accounts have restrictions in place.
- Users running the AWS onboarding quickstart should have:
  - AWS Administrator permissions on the AWS account that you would like to use for CDP.
  - Rights to create AWS resources required by CDP. See list of AWS resources used by CDP.
  - CDP Admin role or Power User role in CDP subscription.
- This AWS onboarding quickstart uses a CloudFormation template that automatically creates the required resources such as buckets, IAM roles and policies, and so on.

- CDP Public Cloud relies on several AWS services that should be available and enabled in your region of choice. Verify if you have enough quota for each AWS service to set up CDP in your AWS account. See list of AWS resources used by CDP.

If you have more complex requirements than those listed here, contact Cloudera Sales Team to help you with the CDP onboarding.

Create a CDP credential

In the CDP console, the first step is to create a CDP credential. The CDP credential is the mechanism that allows CDP to create resources inside of your cloud account.

Procedure

1. Log in to the CDP web interface.
2. From the CDP home screen, click the Management Console icon.
3. In the Management Console, select Shared Resources > Credentials from the navigation pane.
4. Click Create Credential.
5. Click the Copy icon to the right of the **Create Cross-account Access Policy** text box.

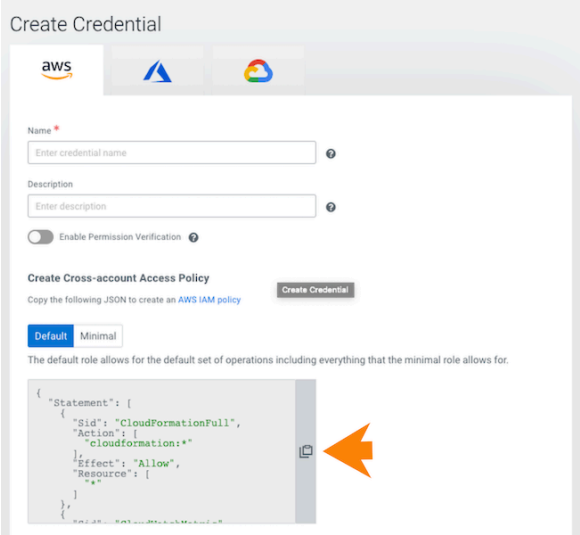

6. In a second browser tab, open the **AWS Console** and navigate to Identity and Access ManagementPolicies. Click Create Policy.

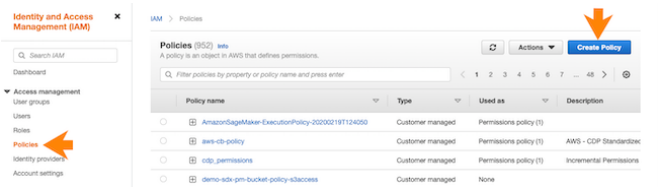

7. Click on the JSON tab and paste the access policy in the text box.

   You may get a warning related to using wildcards. You may ignore it and proceed to the next step.
8. Click Next: Tags.
9. Click Review Policy.
10. Give the policy a unique name and a description.
11. Click Create Policy.

    Next, you create the required cross-account role.
12. In the AWS console, navigate back to Identity and Access Management.
13. Click RolesCreate Role.

14. Under **Select type of trusted entity**, select Another AWS account.
15. Return to the CDP Management Console and copy the contents of the Service Manager Account ID field on the **Credentials** page.
16. In the AWS console, paste the Service Manager Account ID into the Account ID field.
17. Return to the CDP Management Console and copy the contents of the External ID field on the **Credentials** page.
18. In the AWS console, check the "Require external ID" options box, and then paste the External ID copied from CDP into the External ID field.
19. Click Permissions and select the checkbox next to the name of the policy that you created in Step 8.
20. Click Next: Tags.
21. Click Next: Review.
22. Give the role a unique name and description, then click Create Role.
23. Still in the role page of the AWS console, search for the role you just created, and click on it.
24. Copy the Role ARN at the top of the **Summary** page.

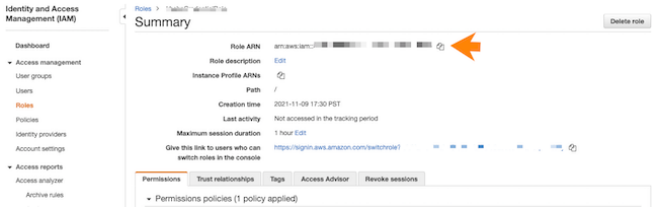

25. Return to the **Credentials** page in the CDP Management Console.
26. Give the CDP credential a name and description. The name can be any valid name.
27. Paste the Role ARN that you copied from the AWS console into the Cross-account Role ARN field, then click Create.

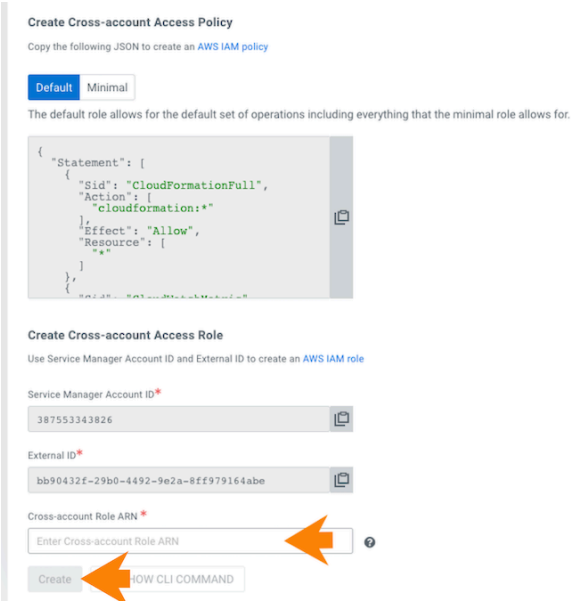

Now that you've created a cross-account role, proceed to creating a CDP environment.

Register a CDP environment

Before you register an environment, you'll want to create specific IAM roles and policies so that CDP can operate in a secure manner.

About this task

For background information, a description of what we're building and why can found here. For this quickstart, we'll use CloudFormation to set all of this up for you.

Procedure

1. Download the CloudFormation provided template here.

2. In the AWS console, deploy the CloudFormation template:
   a) In **AWS Services**, search for CloudFormation.
   b) Click Create Stack and select With new resources (standard).
   c) Select Template is ready and then Upload a template file.

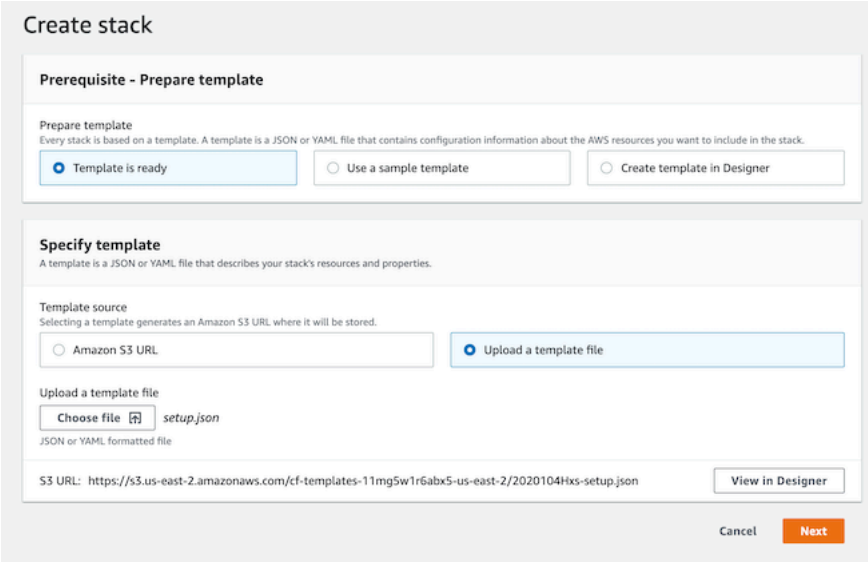

   d) Click Choose file and select the CloudFormation template that you downloaded.
   e) Click Next.
   f) Under Stack name, enter a stack name. The name can be any valid name.
   g) Under **Parameters**, complete the following fields:
      - BackupLocationBase: Choose an unused bucket name and path for the FreeIPA backups. CDP will be creating the bucket for you. The same bucket can be used for BackupLocationBase, LogsLocationBase, and StorageLocationBase. By default this is set to my-bucket/my-backups.
      - CrossAccountARN: Do not change the default value. This parameter is only required when encryption is enabled, but since in this quickstart we do not enable encryption, you should leave this value as is.
      - LogsLocationBase: Choose an unused bucket name and path for the logs. CDP will be creating the bucket for you. The same bucket can be used for BackupLocationBase, LogsLocationBase, and StorageLocationBase. By default this is set to my-bucket/my-logs.
      - StorageLocationBase: Choose an unused bucket name and path for the data. CDP will be creating the bucket for you. The same bucket can be used for BackupLocationBase, LogsLocationBase, and StorageLocationBase. By default this is set to my-bucket/my-data.
      - Prefix: A short prefix of your choosing, which will be added to the names of the IAM resources CDP will be creating. We chose "cloudera" as an example.
      - s3KmsEncryption: Encryption will be disabled for the created bucket. You don't need to change this value.

      For example:

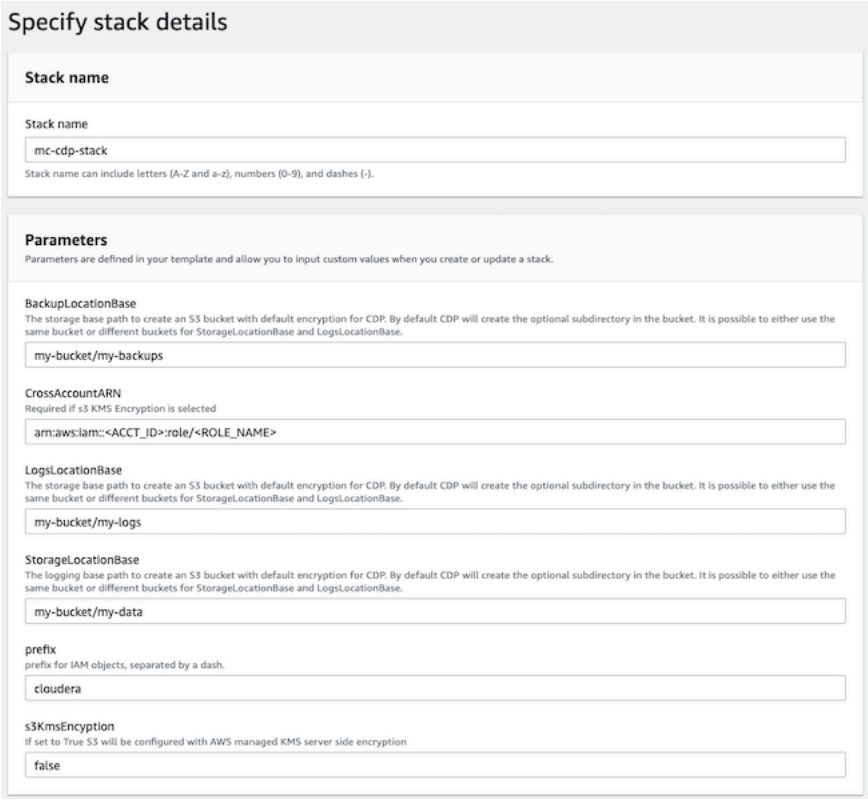

Make a note of the BackupLocationBase, LogsLocationBase, StorageLocationBase, and Prefix that you define. You will need them later.

h) Click Next.

i) At the **Configure Stack Options** page, click Next.

j) At the bottom of the **Review** page, under Capabilities, click the checkbox next to I acknowledge that AWS Cloudformation might create IAM resources with custom names, as that is exactly what we will be doing.

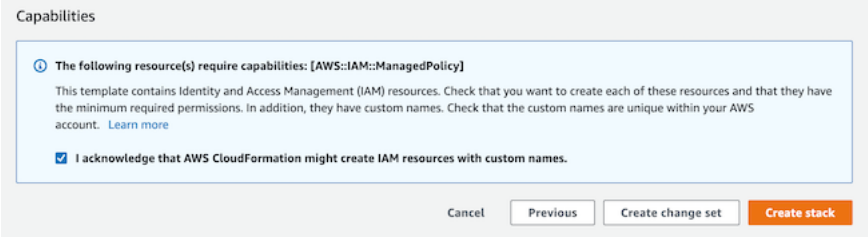

k) Click Create Stack.

3. Still in the AWS console, create an SSH key in the region of your choice. If there is already an SSH key in your preferred region that you'd like to use, you can skip these steps.

a) In **AWS Services**, search for EC2.

b) In the top right corner, verify that you are in your preferred region.

c) On the left hand navigation bar, choose Key Pairs.

d) On the top right of the screen, select Create Key Pair.

e) Provide a name. The name can be any valid name.

f) Choose RSA type, and then choose the pem format.

g) Click Create key pair.

4. Return to the CDP Management Console and navigate to EnvironmentsRegister Environments.

5. Provide an environment name and description. The name can be any valid name.

6. Choose Amazon as the cloud provider.

7. Under **Amazon Web Services Credential**, choose the credential that you created earlier.
8. Click Next.
9. Under **Data Lake Settings**, give your new data lake a name. The name can be any valid name. Choose the latest data lake version.
10. Under **Data Access and Audit**:
    - Choose prefix-data-access-instance-profile>
    - For Storage Location Base, choose the StorageLocationBase from the cloud formation template.
    - For Data Access Role, choose prefix-datalake-admin-role.
    - For Ranger Audit Role, choose prefix-ranger-audit-role, where "prefix" is the prefix you defined in the **Parameters** section of the stack details in AWS.

    For example:

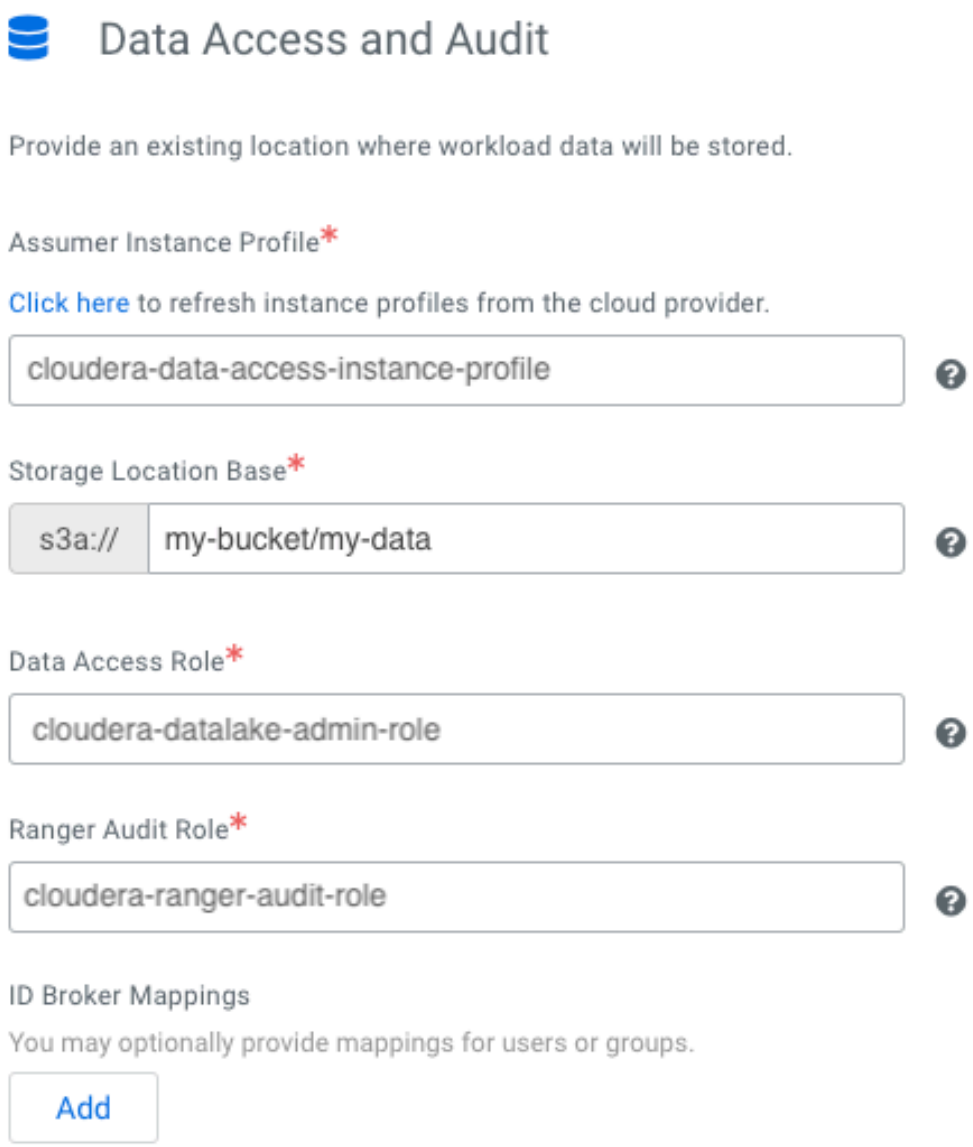

11. For Data Lake **Scale**, choose Light Duty.
12. Click Next.
13. Under Select Region, choose your desired region. This should be the same region you created an SSH key in previously.

14. Under **Select Network**, choose Create New Network.
15. Create private subnets should be enabled by default. If it isn't, enable it.

    Note:

    By enabling private subnets you will not have SSH access to cluster nodes unless you have access to the VPC.
16. Click the toggle button to enable Enable Public Endpoint Access Gateway.

    For example:

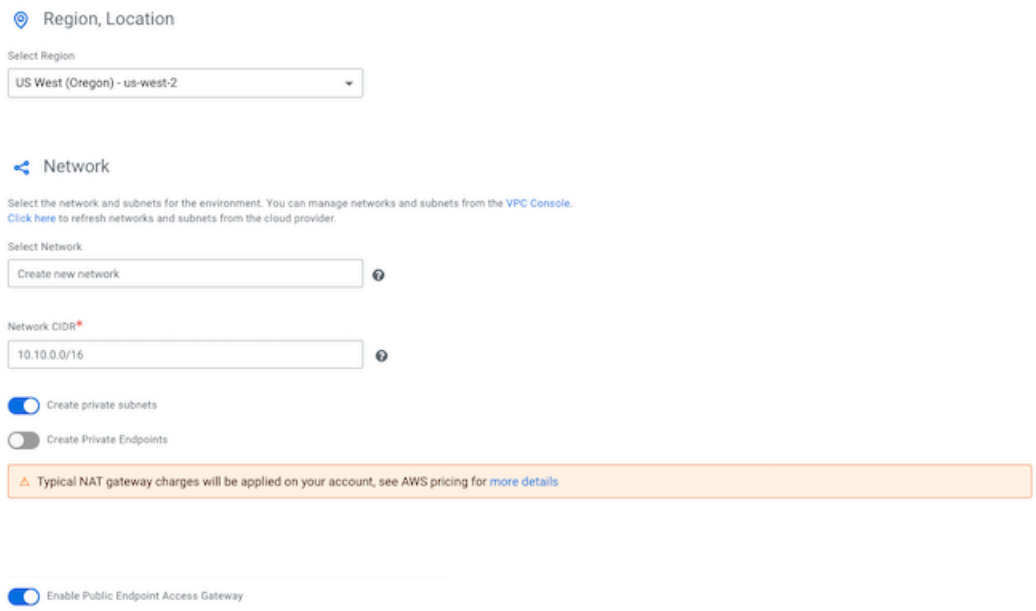

17. Under **Security Access Settings**, choose Create New Security Groups.

18. Under **SSH Settings**, choose the SSH key you created earlier.
    For example:

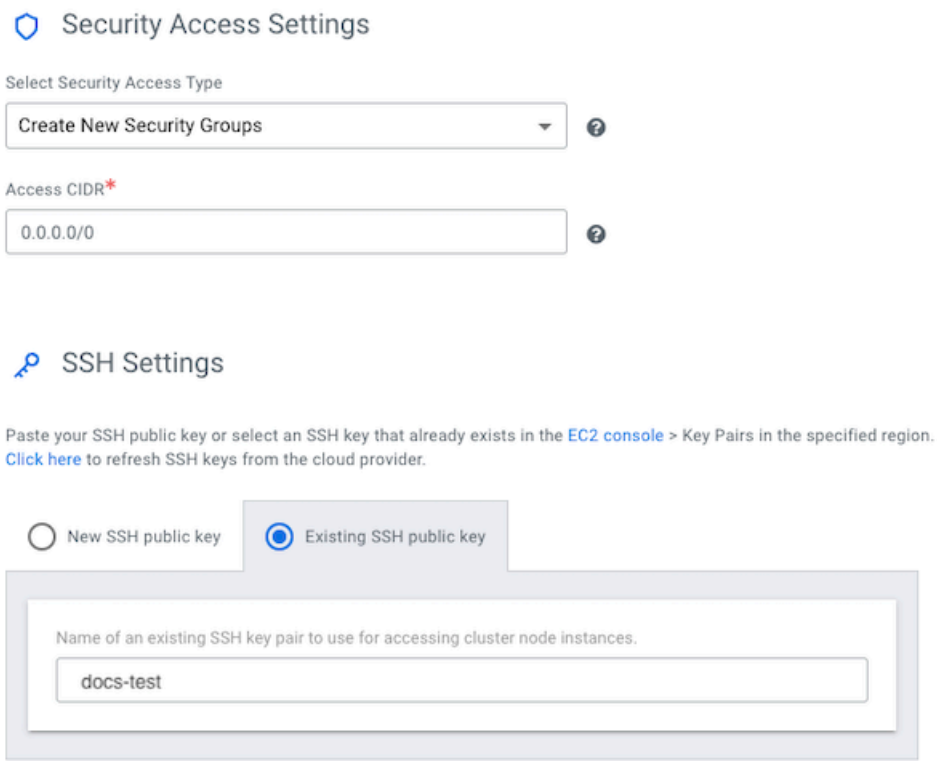

19. Optionally, under **Add Tags**, provide any tags that you'd like the resources to be tagged with in your AWS account.
20. Click Next.

21. Under **Logs**:

a) Choose the Instance Profile titled prefix-log-access-instance-profile, where "prefix" is the prefix you defined in the **Parameters** section of the stack details in AWS.

b) For Logs Location Base, choose the LogsLocationBase from the CloudFormation template.

c) For Backup Location Base, choose the BackupLocationBase from the CloudFormation template.

For example, using the parameters we defined earlier:

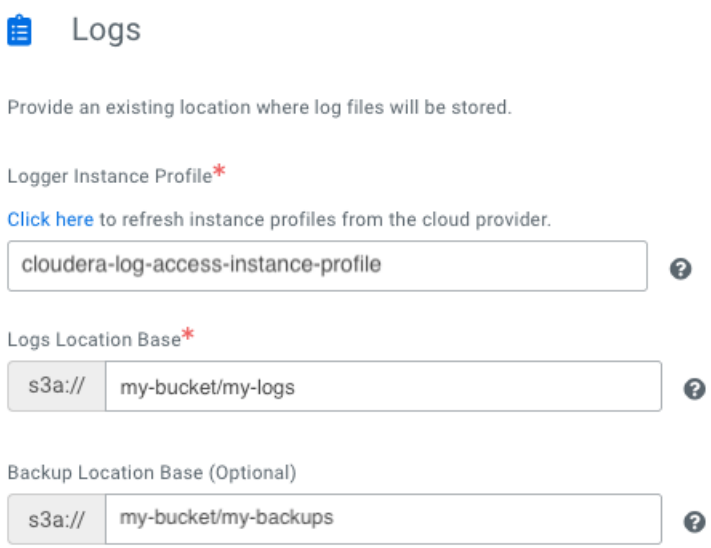

22. Click Register Environment.

Azure quickstart

If you've reached the CDP landing page for the first time, you've come to the right place! In this quickstart, we'll show you step-by-step how to connect CDP to your Azure subscription, so that you can begin to provision clusters and workloads.

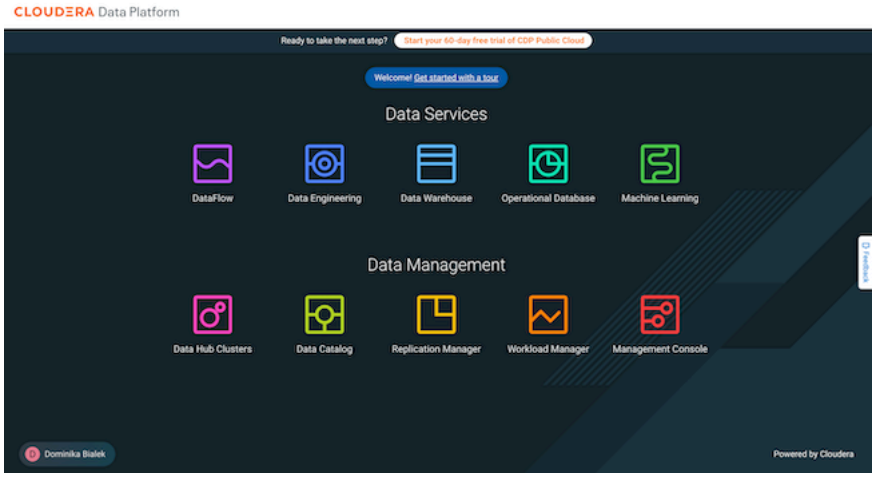

To complete this quickstart, you'll need access to three things:

- The CDP console pictured above
- The Azure console
- Azure Cloud shell

Note: This Azure onboarding quickstart is intended for simple CDP evaluation deployments only. It may not work for scenarios where Azure resources such as VNet, security group, storage accounts, and so on, are pre-created or Azure accounts have restrictions in place.

In addition to this documentation, you can refer to the following video:

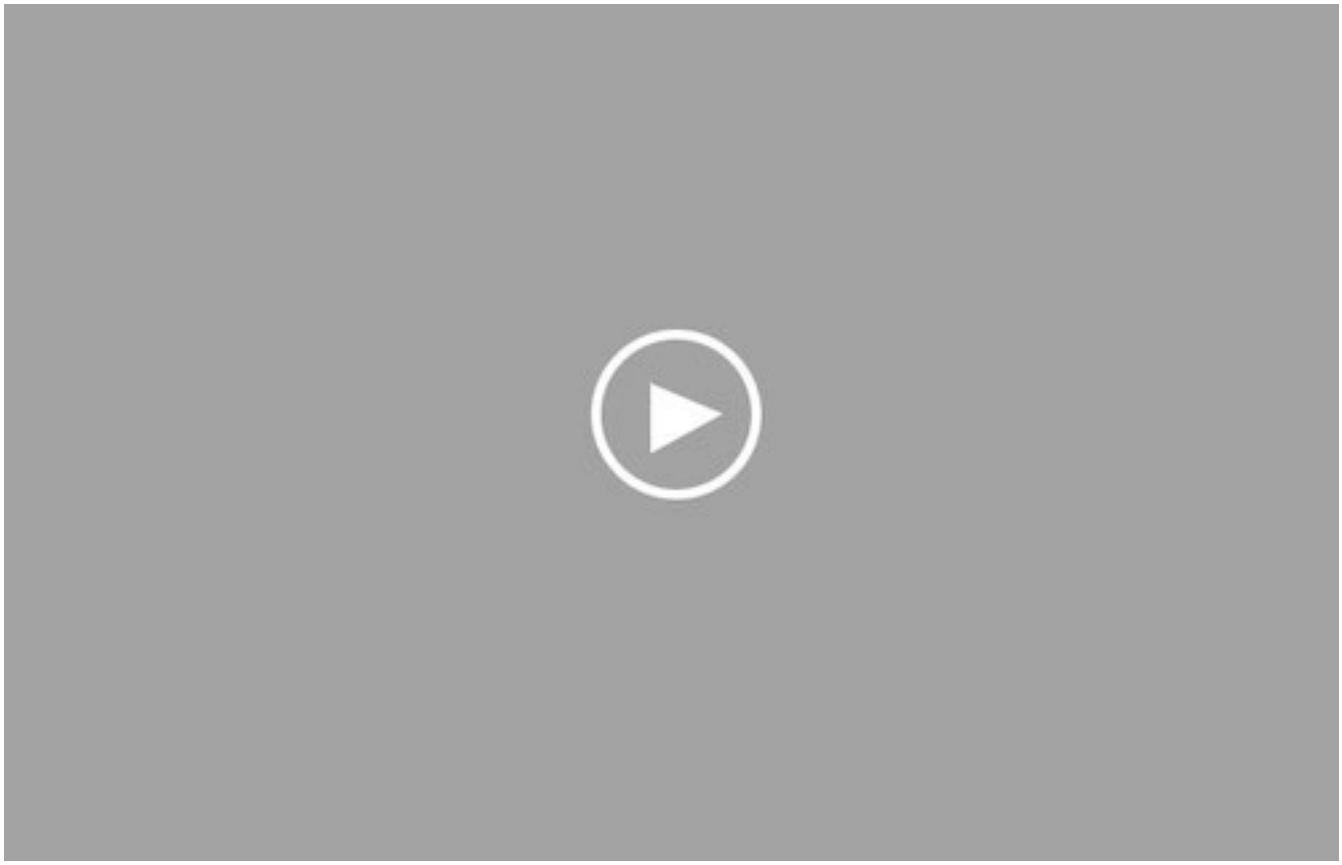

The steps that we will perform are:

Step 0: Verify the Azure prerequisites

Step 1: Create an Azure AD app

Step 2: Deploy the Azure quickstart template

Step 3: Assign roles

Step 4: Create or locate an SSH key

Step 5: Create a CDP credential

Step 6: Register a CDP environment

Verify Azure cloud platform prerequisites

Before getting started with the Azure onboarding quickstart, review and acknowledge the following:

- This Azure onboarding quickstart is intended for simple CDP evaluation deployments only. It may not work for scenarios where Azure resources such as VNet, security group, storage accounts, and so on, are pre-created or Azure accounts have restrictions in place.
- User running the Azure onboarding quickstart should have:
  - Owner permissions on the Azure subscription that you would like to use for CDP.
  - Rights to create Azure resources required by CDP. See list of Azure resources used by CDP.
  - Rights to create an Azure AD application (service principal) and assign Contributor role at subscription level.
  - CDP Admin role or Power User role in CDP subscription.

- This Azure onboarding quickstart uses an Azure ARM template that automatically creates the required resources such as storage accounts, containers, managed identities, resource groups, and so on.
- CDP Public Cloud relies on several Azure services that should be available and enabled in your region of choice. Verify if you have enough quota for each Azure service to set up CDP in your Azure account. See list of Azure resources used by CDP.

If you have more complex requirements than those listed here, contact Cloudera Sales Team to help you with CDP onboarding.

Create an Azure AD app

In the Azure portal, create an application in your Azure Active Directory tenant. This steps allows you to use the native Cloud Shell terminal and not have to set up Azure CLI.

Procedure

1. Log in to the Azure portal and launch Cloud Shell.

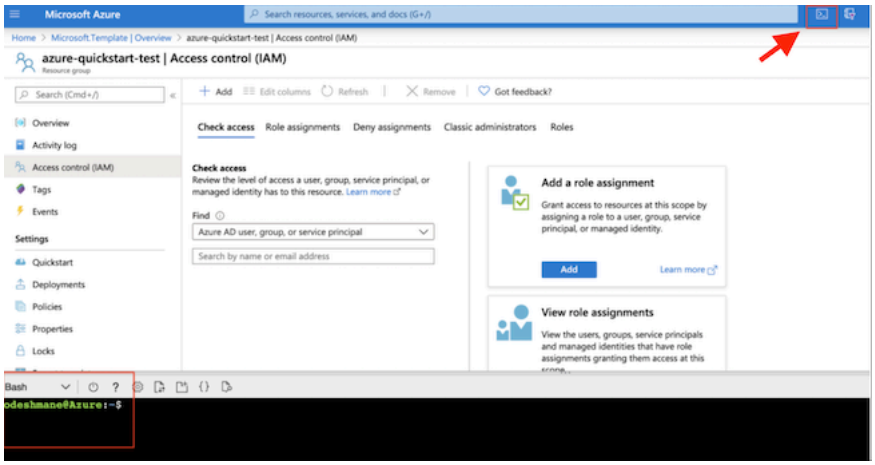

2. Run the following command to return the subscription ID and tenant ID:

```
az account list|jq '.[]|{"name": .name, "subscriptionId": .id, "tenantId
": .tenantId, "state": .state}'
```

The output of this command is shown below:

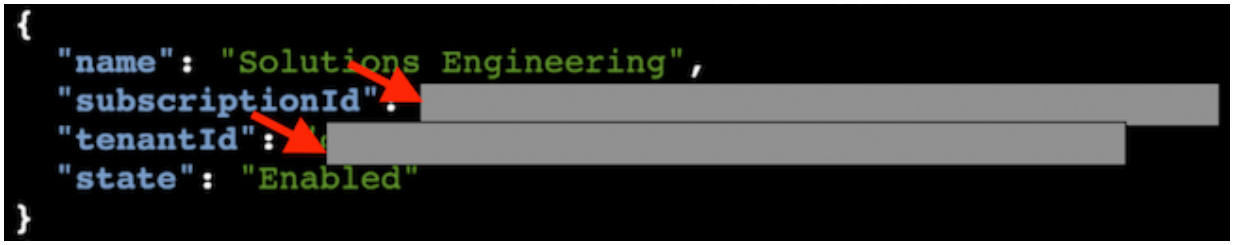

Make a note of the subscriptionId and tenantId values. You will need them later.

Note: In case you have more than one subscription, make sure to only make a note of the subscription that you would like to use for CDP.

3. Run the command below to create an app in Azure AD and assign the "Contributor" role at the subscription.

   Note: Replace {subscriptionId} in the command with the subscription ID value from the previous step.

```
az ad sp create-for-rbac --name http://cloudbreak-app --role Contributor
 --scopes /subscriptions/{subscriptionId}
```

   The output of this command is shown below:

Deploy the Azure quickstart template

The Azure quickstart template is a customized ARM template that deploys essential Azure resources for the CDP environment.

Procedure

1. Click Deploy to Azure to begin ARM template deployment of CDP prerequisites in your Azure subscription.
2. Log in to Azure to create the resources in your subscription that are required for CDP deployment. These resources include VNet, ADLS Gen2, and 4 user managed identities.
3. On the **Custom deployment** screen, click Create new under the Resource group field and then give the resource group a name (it should only contain letters, numbers, and hyphens).
4. Under **Settings**, provide an Environment Name in the corresponding field.

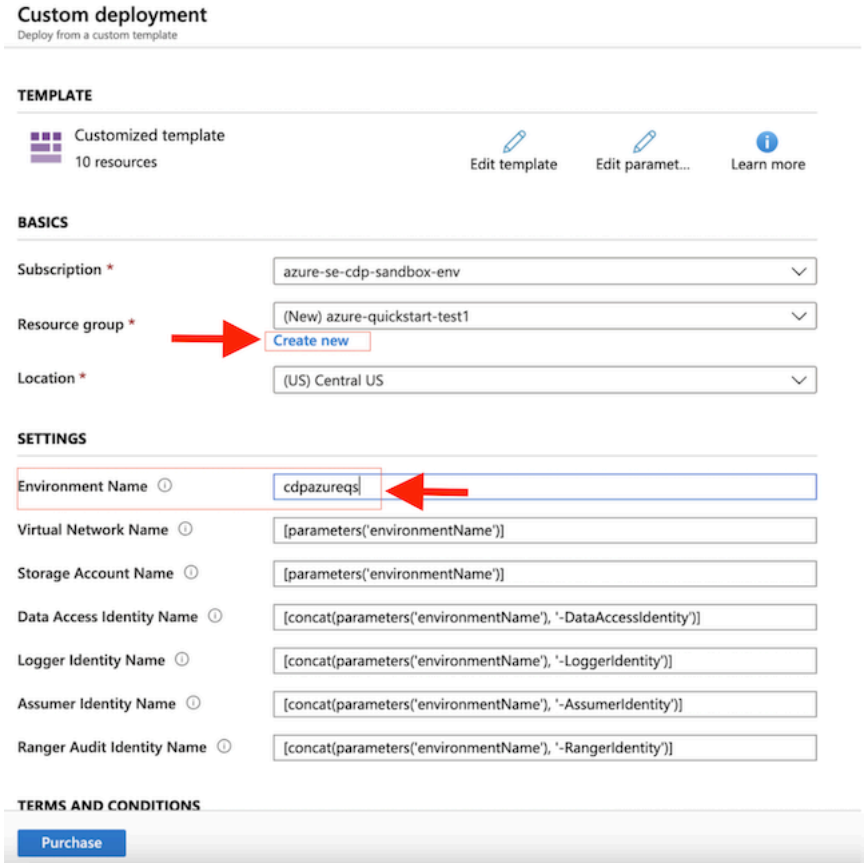

5. Accept the terms and conditions, and click Purchase.

   An ARM script begins to run in the background, creating the resources required for a CDP environment. This may take around 10 minutes.
6. When your resource group is up, navigate to the **Overview** page of the resource group.
7. Copy and paste the following values into a note, as you will need them in the next task:
   - Subscription ID: Your subscription ID is found at the top of the resource group **Overview** page.
   - Resource group: The name of the resource group that you created.

Assign roles

Azure Resource Manager templates do not support role assignments at a scope other than resource groups. Perform the following role assignments through UI or CLI.

Before you begin

Make sure that you have your note from the previous step, where you copied values for the Subscription ID and resource group name.

Procedure

1. Once you have values for the subscription ID, resource group name, storage account, environment name, and all four managed identities, click here to download a script.
2. Create a new file in Cloud Shell with the same name, and copy the content of the script there.
3. Replace the following values in the script with the values you have collected thus far:

```
#!/bin/sh

export SUBSCRIPTIONID="<REPLACE WITH YOUR AZURE SUBSCRIPTION ID>"
export RESOURCEGROUPNAME="<REPLACE WITH EXISTING RESOURCE GROUP NAME>"
export STORAGEACCOUNTNAME=$(az storage account list -g $RESOURCEGROUPNAME --subscription $SUBSCRIPTION
export ASSUMER_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subscription $SUBSCRIPTIONID|jq '.[
export DATAACCESS_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subscription $SUBSCRIPTIONID|jq
```

   For example, your script should look similar to this:

```
#!/bin/sh

export SUBSCRIPTIONID="jfs85ls8-sik8-8329-fq0m-jqo7v06dk6sy"
export RESOURCEGROUPNAME="myCDPresourcegroup"
export STORAGEACCOUNTNAME=$(az storage account list -g $RESOURCEGROUPNAME
--subscription $SUBSCRIPTIONID|jq '.[]|.name'| tr -d '"')
export ASSUMER_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subscri
ption $SUBSCRIPTIONID|jq '.[]|{"name":.name,"principalId":.principalId}|
select(.name | test("AssumerIdentity"))|.principalId'| tr -d '"')
export DATAACCESS_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subs
cription $SUBSCRIPTIONID|jq '.[]|{"name":.name,"principalId":.principalI
d}|select(.name | test("DataAccessIdentity"))|.principalId'| tr -d '"')
export LOGGER_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subscr
iption $SUBSCRIPTIONID|jq '.[]|{"name":.name,"principalId":.principalId}|
select(.name | test("LoggerIdentity"))|.principalId'| tr -d '"')
export RANGER_OBJECTID=$(az identity list -g $RESOURCEGROUPNAME --subscrip
tion $SUBSCRIPTIONID|jq '.[]|{"name":.name,"principalId":.principalId}|s
elect(.name | test("RangerIdentity"))|.principalId'| tr -d '"')
# Assign Managed Identity Operator role to the assumerIdentity principal
 at subscription scope
az role assignment create --assignee $ASSUMER_OBJECTID --role 'f1a07417-
d97a-45cb-824c-7a7467783830' --scope "/subscriptions/$SUBSCRIPTIONID"
# Assign Virtual Machine Contributor role to the assumerIdentity principal
 at subscription scope
az role assignment create --assignee $ASSUMER_OBJECTID --role '9980e02c-c
2be-4d73-94e8-173b1dc7cf3c' --scope "/subscriptions/$SUBSCRIPTIONID"
```

```
# Assign Storage Blob Data Contributor role to the assumerIdentity prin
cipal at logs filesystem scope
az role assignment create --assignee $ASSUMER_OBJECTID --role 'ba92f5b4-2
d11-453d-a403-e96b0029c9fe' --scope "/subscriptions/$SUBSCRIPTIONID/reso
urceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAccounts/
$STORAGEACCOUNTNAME/blobServices/default/containers/logs"
# Assign Storage Blob Data Contributor role to the loggerIdentity princ
ipal at logs/backup filesystem scope
az role assignment create --assignee $LOGGER_OBJECTID --role 'ba92f5b4-
2d11-453d-a403-e96b0029c9fe' --scope "/subscriptions/$SUBSCRIPTIONID/res
ourceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAccoun
ts/$STORAGEACCOUNTNAME/blobServices/default/containers/logs"
az role assignment create --assignee $LOGGER_OBJECTID --role 'ba92f5b4-
2d11-453d-a403-e96b0029c9fe' --scope "/subscriptions/$SUBSCRIPTIONID/res
ourceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAccoun
ts/$STORAGEACCOUNTNAME/blobServices/default/containers/backups"
# Assign Storage Blob Data Owner role to the dataAccessIdentity principal
 at logs/data/backup filesystem scope
az role assignment create --assignee $DATAACCESS_OBJECTID --role 'b7e6dc6
d-f1e8-4753-8033-0f276bb0955b' --scope "/subscriptions/$SUBSCRIPTIONID/r
esourceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAcco
unts/$STORAGEACCOUNTNAME/blobServices/default/containers/data"
az role assignment create --assignee $DATAACCESS_OBJECTID --role 'b7e6dc6
d-f1e8-4753-8033-0f276bb0955b' --scope "/subscriptions/$SUBSCRIPTIONID/r
esourceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAcco
unts/$STORAGEACCOUNTNAME/blobServices/default/containers/logs"
az role assignment create --assignee $DATAACCESS_OBJECTID --role 'b7e6dc6
d-f1e8-4753-8033-0f276bb0955b' --scope "/subscriptions/$SUBSCRIPTIONID/r
esourceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAcco
unts/$STORAGEACCOUNTNAME/blobServices/default/containers/backups"
# Assign Storage Blob Data Contributor role to the rangerIdentity principa
l at data/backup filesystem scope
az role assignment create --assignee $RANGER_OBJECTID --role 'ba92f5b4-2d1
1-453d-a403-e96b0029c9fe' --scope "/subscriptions/$SUBSCRIPTIONID/resour
ceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAccounts/
$STORAGEACCOUNTNAME/blobServices/default/containers/data"
az role assignment create --assignee $RANGER_OBJECTID --role 'ba92f5b4-2d1
1-453d-a403-e96b0029c9fe' --scope "/subscriptions/$SUBSCRIPTIONID/resour
ceGroups/$RESOURCEGROUPNAME/providers/Microsoft.Storage/storageAccounts/
$STORAGEACCOUNTNAME/blobServices/default/containers/backups"
```

4. Run the Cloud Shell script: sh azure_msi_role_assign.sh

Create or locate an SSH Key

CDP requires that you provide a public SSH key for admin access to VM instances.

You can find more information on SSH key requirement in the topic SSH key. If you need to create one, you can do so by running ssh-keygen -t rsa.

When you complete this step, you have created all of the Azure resources required for this quickstart.

Create a CDP credential

In the CDP Console, the first step is to create a CDP credential. The CDP credential is the mechanism that allows CDP to create resources inside of your cloud account.

Procedure

1. Log in to the CDP web interface.
2. From the CDP home screen, click the Management Console icon.
3. In the Management Console, select Shared Resources > Credentials from the navigation pane.

4. Select the Azure tab, name your credential, and enter the values you previously collected for subscription ID, app ID, and password.

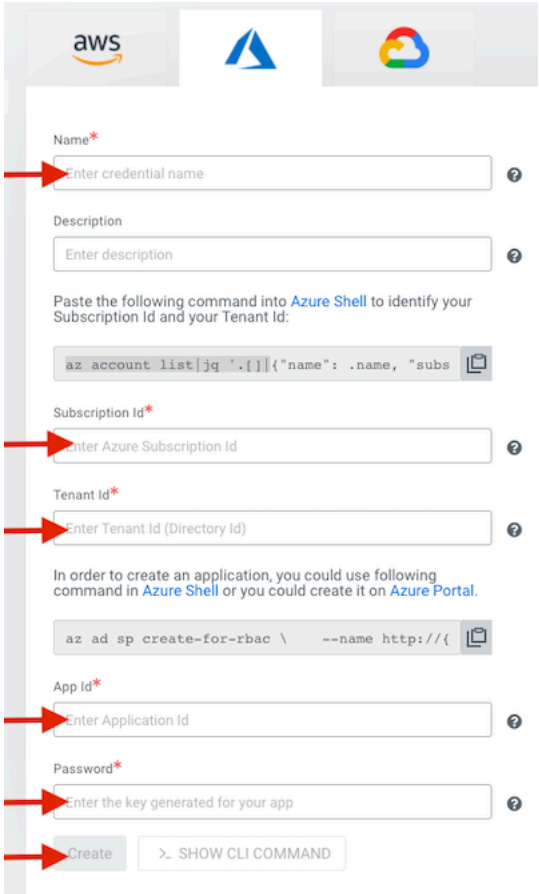

Register a CDP environment

When you register an environment, you set properties related to data lake scaling, networking, security, and storage. You will need your Azure environment name, resource group name, storage account name, and virtual network name from your resource group.

Procedure

1. In the CDP Management Console, navigate to Environments and click Register Environment.
2. Provide an Environment Name and description. The name can be any valid name.
3. Choose Azure as the cloud provider.

4. Under Microsoft Azure Credential, choose the credential you created in the previous task.

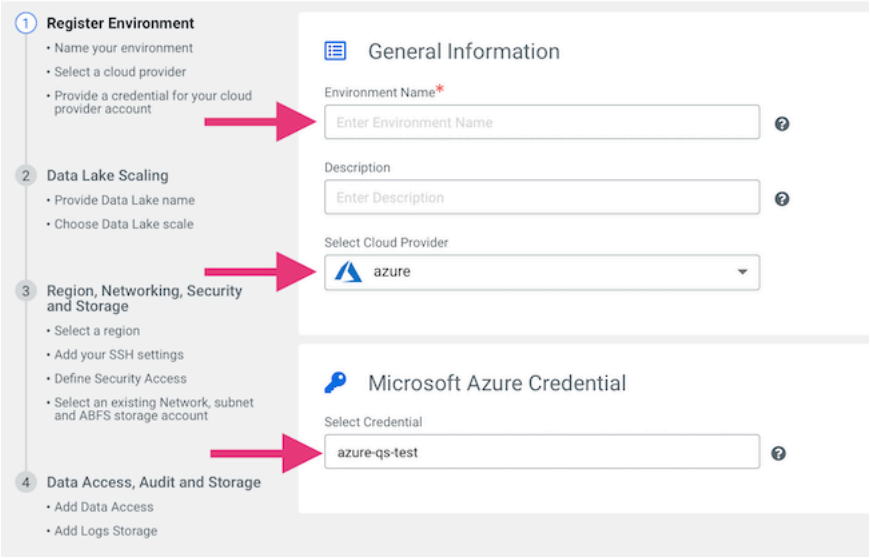

5. Click Next.
6. Under **Data Lake Settings**, give your new data lake a name. The name can be any valid name. Choose the latest data lake version.

7. Under **Data Access and Audit**, choose the following:
   - Assumer Identity: <resourcegroup-name>-<envName>-AssumerIdentity
   - Storage Location Base: data@<storageaccount-name>
   - Data Access Identity: <resourcegroup-name>-<envName>-DataAccessIdentity
   - Ranger Audit Role: <resourcegroup-name>-<envName>-RangerIdentity

   Warning: Ensure that you have entered the correct location base. If the name entered does not match the actual location base created by the quickstart script, environment registration will fail.

   For example:

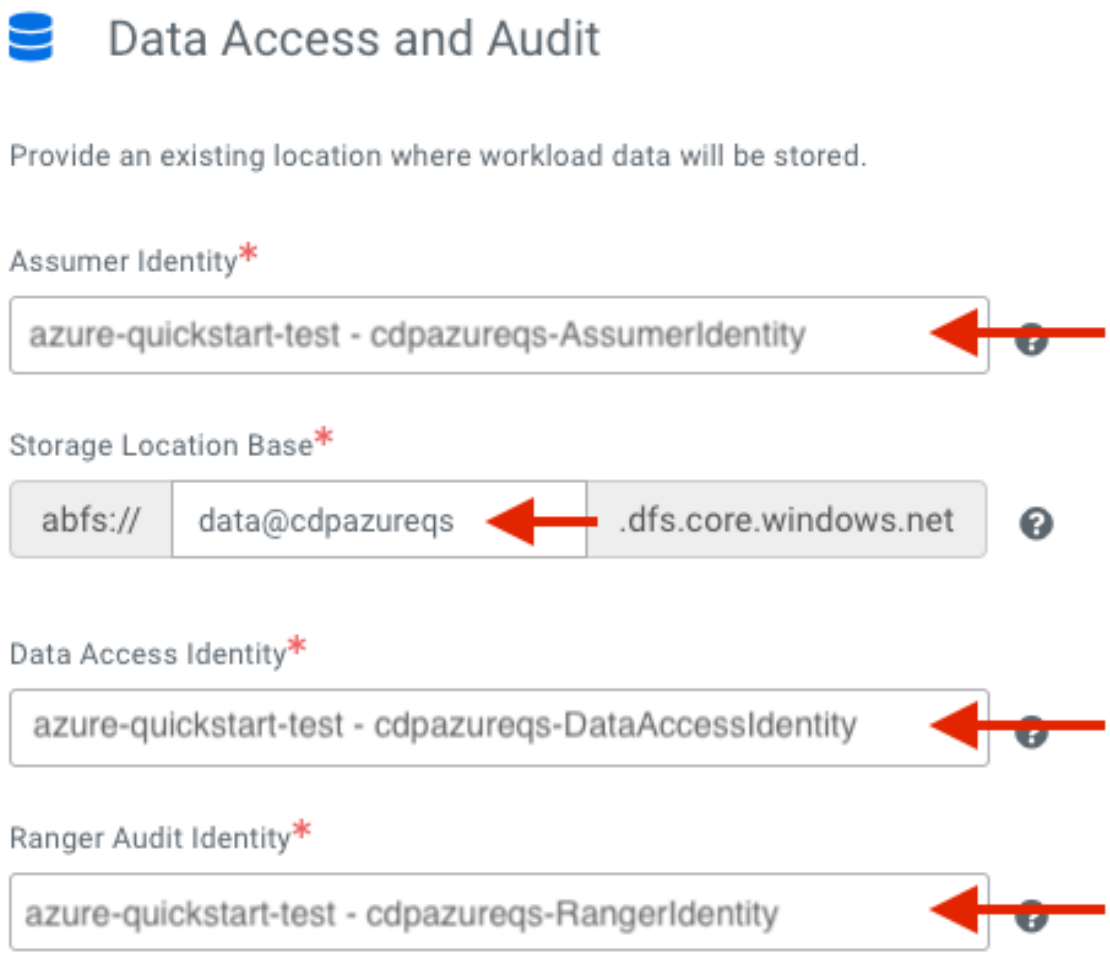

8. For Data Lake **Scale**, choose Light Duty.

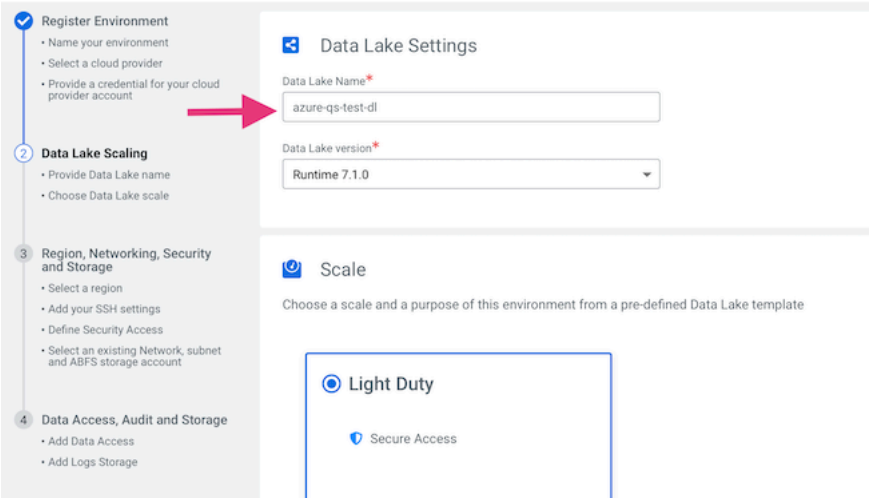

9. Click Next.
10. Under Select Region, choose your desired region. This should be the same region you created an SSH key in previously.
11. Under Select Resource Group, choose your resource group <resourcegroup-name>.
12. For the Select Network field, select the name of the "Virtual Network" resource that was created when you deployed the ARM template to create the resource group. The name of the Virtual Network should be the same as your environment name, but you can verify this in the Azure portal on the Overview page of your resource group.
13. Under **Security Access Settings**, select Create New Security Groups for the Security Access Type.

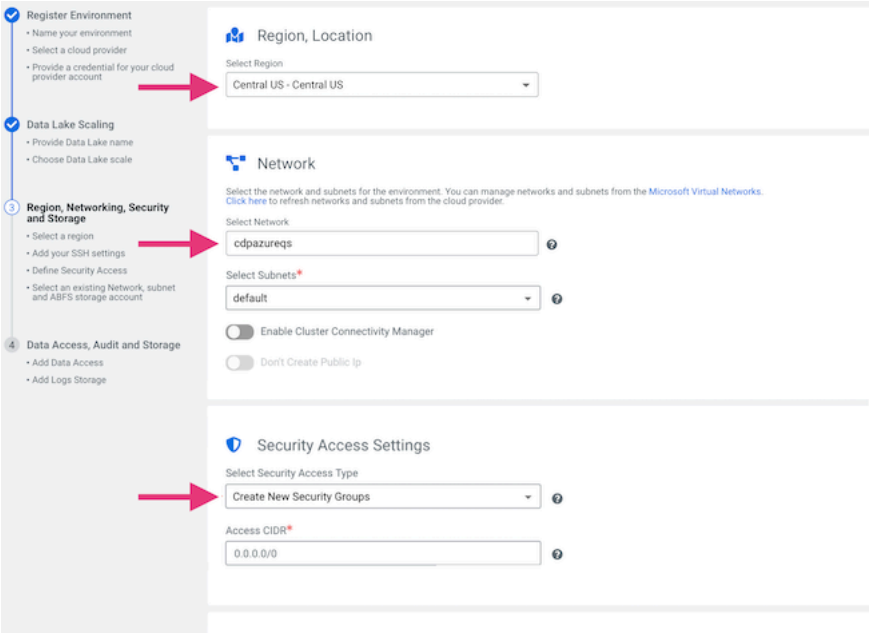

14. Under **SSH Settings**, paste the public SSH key that you created earlier.
15. Optionally, under **Add Tags**, provide any tags that you'd like the resources to be tagged with in your Azure account.
16. Click Next.

17. Under **Logs**, choose the following:

- Logger Identity: <resourcegroup-name>-<envName>-LoggerIdentity
- Logs Location Base: logs@<storageaccount-name>
- Backup Location Base: backups@<storageaccount-name>

Warning: Ensure that you have entered the correct location base. If the name entered does not match the actual location base created by the quickstart script, environment registration will fail.

For example:

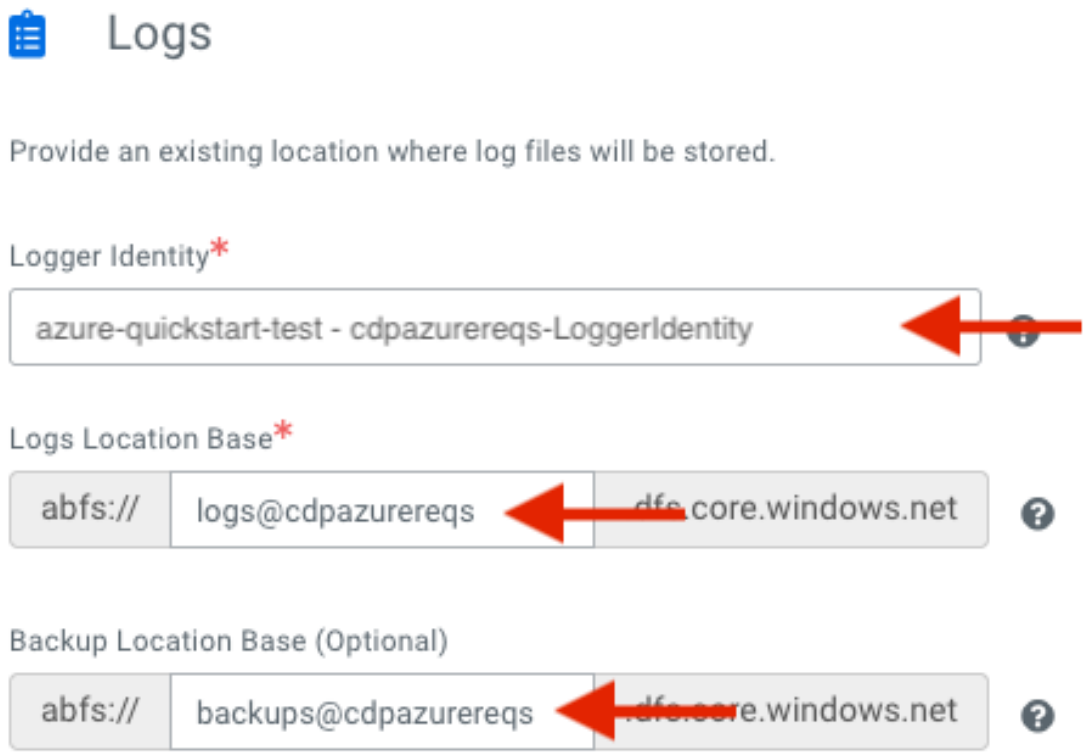

18. Click Register Environment.

Give administrators access

To enable Cloudera DataFlow for an environment, users must have the DFAdmin role. Grant the DFAdmin role to a user or group that should be allowed enable CDF for an environment.

Before you begin

- You have the Cloudera PowerUser role.

Procedure

1. From the Cloudera Management Console, click Environments.
2. Use the search field to find and select the Cloudera Public Cloud environment for which you want to grant DFAdmin rights.
3. Click Actions | Manage Access to display the Environment Access page.
4. Find the user to whom you want to grant the DFAdmin role, and click Update Roles.

5. From the Update Roles page, select DFAdmin and click Update.

Example

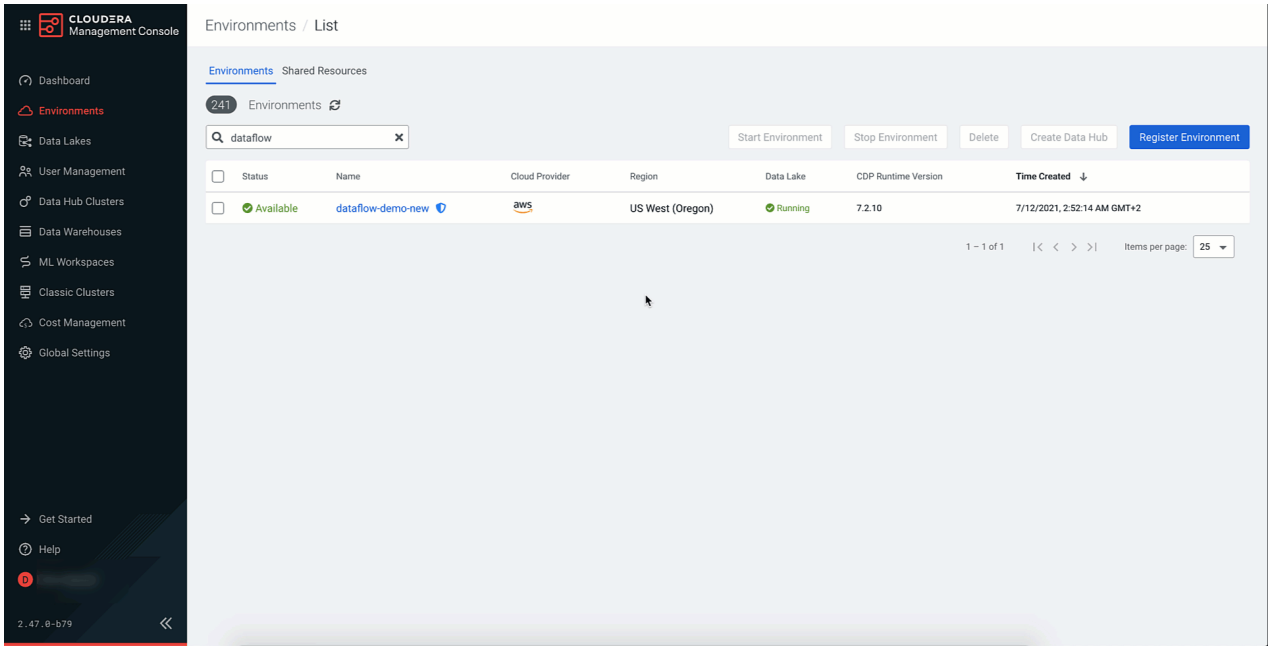

What to do next

When you have finished granting a user or group administrator permissions, they can now proceed by enabling Cloudera DataFlow for an environment.

Related Information

Cloudera DataFlow Security

Enable Cloudera DataFlow for your environment

Before you can deploy flow definitions, you must enable Cloudera DataFlow for a Cloudera Public Cloud environment. Enabling Cloudera DataFlow for an environment means that you are preparing an active and healthy Cloudera Public Cloud environment for use with Cloudera DataFlow.

Before you begin

- You have a cloud provider account and meet the infrastructure and network requirements.
- You have a healthy Cloudera Public Cloud environment, with FreeIPA and the data lake running and healthy.
- You have the DFAdmin role for the Cloudera Public Cloud environment for which you want to enable Cloudera DataFlow.

Procedure

1. Navigate to Cloudera DataFlow, by selecting DataFlow from the Cloudera Public Cloud Home Page.
2. In Cloudera DataFlow, navigate to Environments, and click Enable to launch the Enable Environment dialog for the environment you want to enable.

3. From Enable Environment, provide the following information:
   - DataFlow Capacity – Specifies Kubernetes cluster minimum and maximum size
   - Networking
   - Specify whether a public endpoint should be deployed to access CDF components via the internet.

     Note:

     CDF only makes this option selectable if it detects that at least one subnet in your environment is considered a public subnet.
   - A list of source IP address ranges which are allowed to connect to the Kubernetes API server.
4. Click Enable. Enabling CDF can take up to one hour.

Example

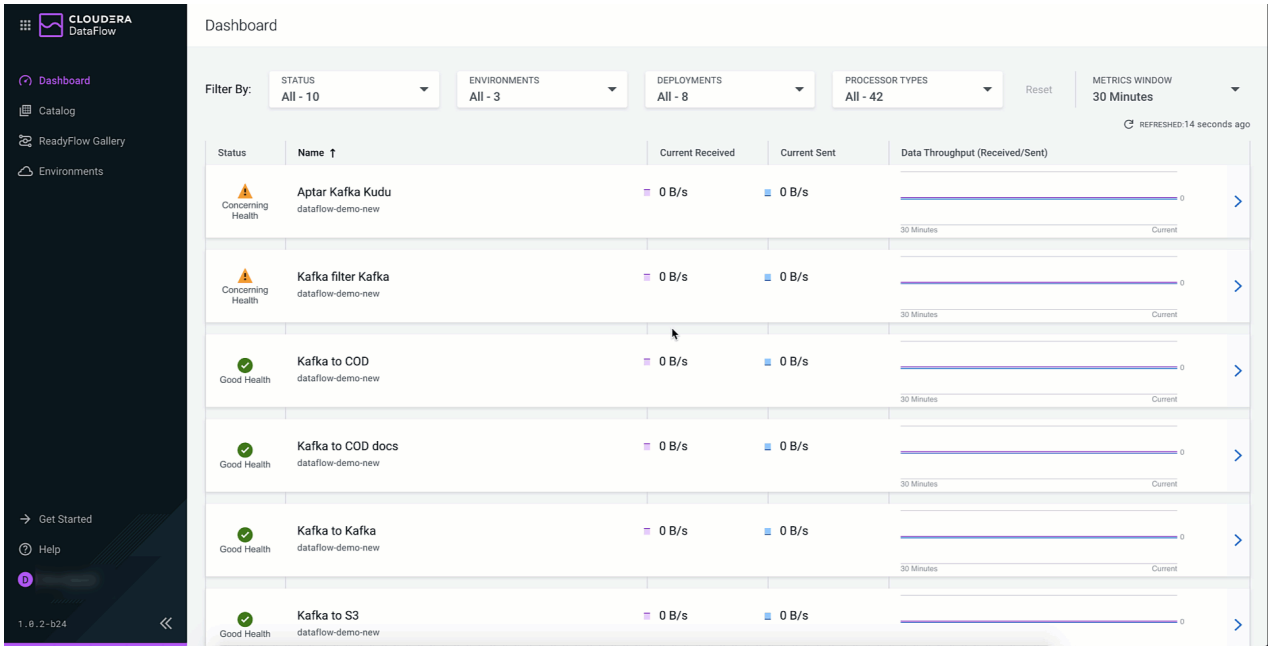

What to do next

When you have finished enabling Cloudera DataFlow for an environment, proceed by giving users permission to import and deploy flow definitions.

Related Information

Enabling Cloudera DataFlow for an environment

Managing Cloudera DataFlow in an environment

Give users access

Cloudera DataFlow restricts who can import flow definitions and deploy them. To get started with Cloudera DataFlow, you must grant users permissions to perform these tasks so that they can import and deploy flow definitions in Cloudera DataFlow.

About this task

To get started with importing and deploying flow definitions, a user needs to have the following Cloudera DataFlow roles, at a minimum:

- DFCatalogAdmin – To import flow definitions to the Cloudera DataFlow Catalog

- DFFlowAdmin – To deploy flow definitions in Cloudera DataFlow

Before you begin

- You have PowerUser role.
- You know the user or group name to which you want to grant Cloudera DataFlow user access roles.

Procedure

1. Give a user permission to import flow definitions.
   a) From Cloudera Management Console, click User Management.
   b) Enter the name of the user or group you wish to authorize in the Search field.
   c) Select the user or group from the list that displays.
   d) Click Roles, then Update Roles.
   e) From Update Roles, select DFCatalogAdmin and click Update.
2. Give a user or group permission to deploy flow definitions.
   a) From Cloudera Management Console, click Environments to display the Environment List page.
   b) Select the environment to which you want a user or group to deploy flow definitions.
   c) Click Actions Manage Access to display the Environment Access page.
   d) Find the user or group and click Update Roles.
   e) From Update Roles, select DFFlowAdmin.
   f) Click Update Roles.

What to do next

When you have finished giving user or groups permission to import and deploy flow definitions, proceed by importing a flow definition.

Related Information

Cloudera DataFlow Security

Add the Hello World ReadyFlow definition to the Catalog

Hello World is an out of box flow definition, designed to make getting started with Cloudera DataFlow quick and easy. To use it, just add the ReadyFlow to the Catalog.

Before you begin

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFCatalogAdmin role granting you access to the Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.

About this task

Note: In a new Cloudera DataFlow deployment with an empty Catalog, you can deploy Hello World directly from the Dashboard.

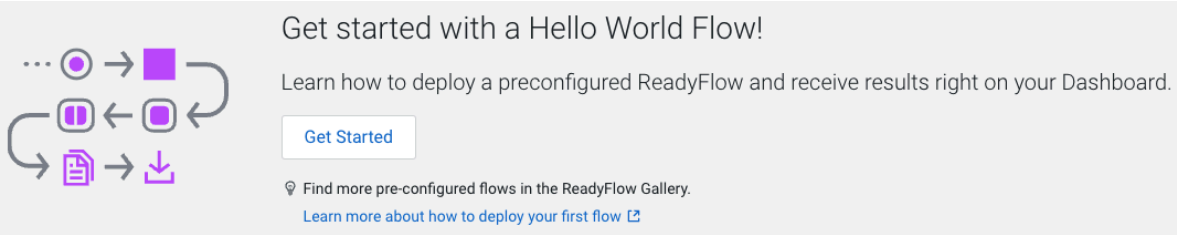

Just click Get Started on the main pane of the Dashboard. You will be forwarded to the Catalog, from where you can deploy the ReadyFlow.

If the Catalog in your ReadyFlow deployment already contains at least one flow definition, You can add Hello World to the Catalog from the ReadyFlow Gallery.

Procedure

1. On the left hand navigation pane, click ReadyFlow Gallery.

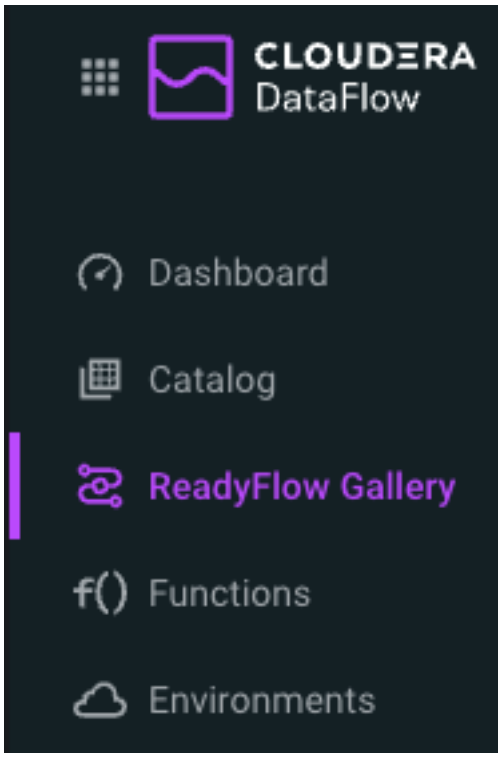

2. From the ReadyFlow Gallery page, select the Hello World ReadyFlow.

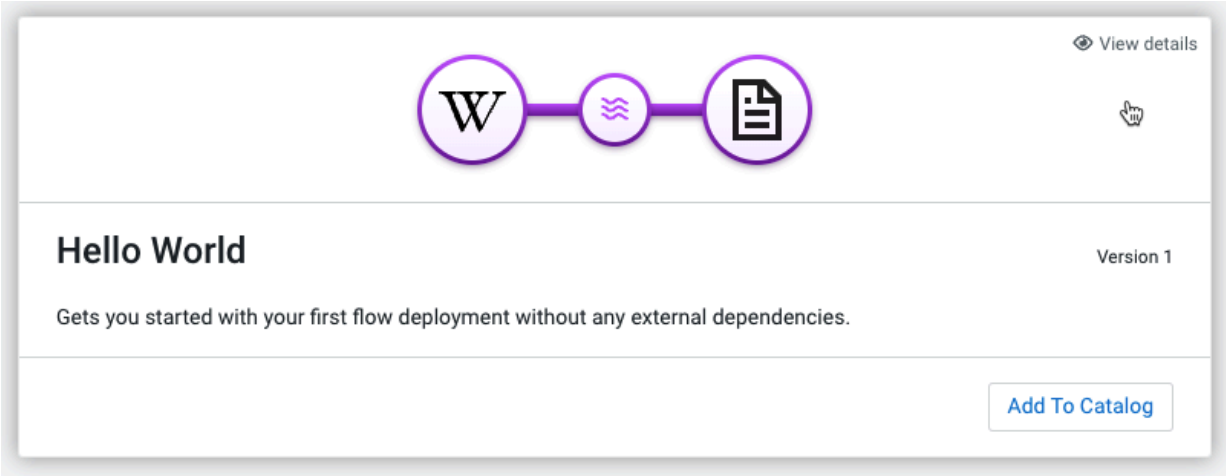

3. If you hover over the card representing the flow definition, the View Details hint appears. Click anywhere on the card to review the ReadyFlow details.
4. Click Add to Catalog to add the ReadyFlow into the CDF Catalog and make it ready for deployment.
5. Click Add to confirm that you want to add Hello World to the Catalog.

Results

The Hello World flow definition is added to the Catalog and is ready for deployment.

What to do next

You can now proceed with deploying the ReadyFlow.

Deploy the Hello World ReadyFlow using the deployment wizard

Learn about the steps to deploy the Hello World Flow to get started with Cloudera DataFlow.

About this task

Once you have added the Hello World ReadyFlow into the Catalog, stay in the Catalog and use the Deployment wizard to deploy that flow definition.

Before you begin

- You have an enabled and healthy Cloudera DataFlow environment.
- You have imported the Hello World Flow flow definition.
- You have been assigned the DFCatalogAdmin role granting you access to the Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.

Procedure

1. To launch the Deployment wizard, select the Hello World flow in the Catalog to display the flow definition details and versions.

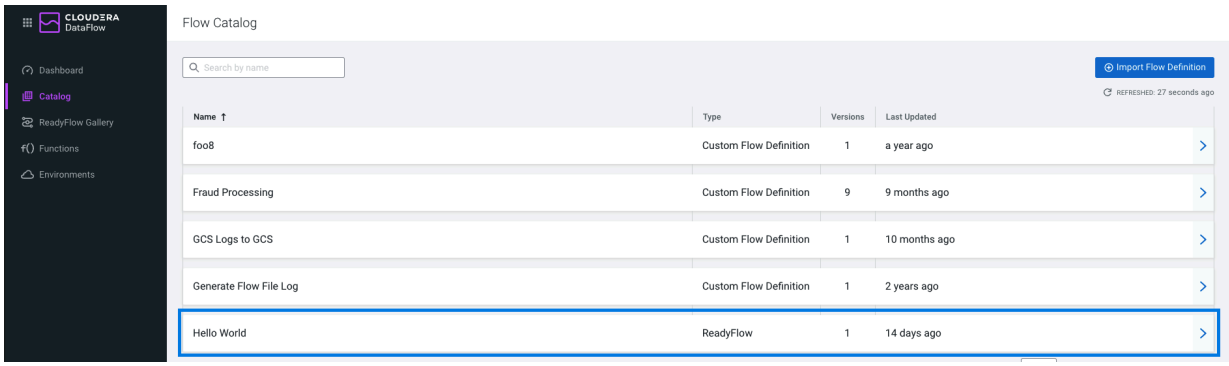

2. Select Version 2 and click Deploy to launch the Deployment wizard.

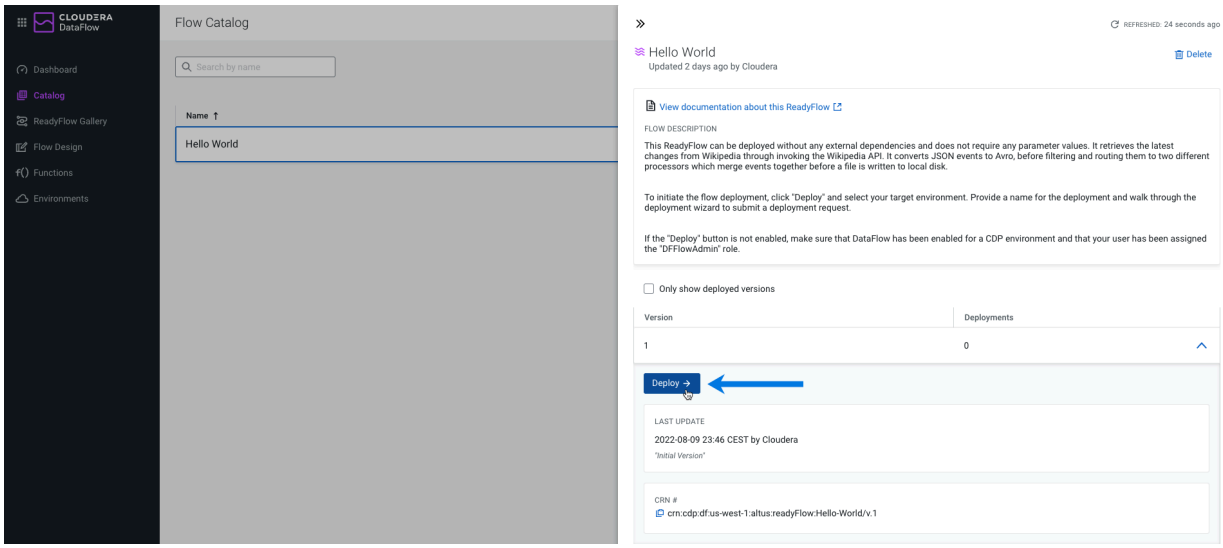

3. Select the Target Workspace where you want to deploy the ReadyFlow.

   The Deployment wizard only displays workspaces that meet the following conditions:

   - Cloudera DataFlow is enabled for the environment
   - The environment is in healthy state
   - You are allowed to access the environment

4. Click Continue.
5. In Overview, give your flow deployment a unique name.

   Note: Flow Deployment names need to be unique. The Deployment Wizard indicates whether a name is valid by displaying a green check below the Deployment Name text box.

   Note: As you have created no Projects for this quick start, you can leave your flow deployment Unassigned.

6. Click Next.
7. In NiFi Configuration, click Next.
8. In Parameters, click Next.
9. In Sizing & Scaling, click Next.

10. In Key Performance Indicators, click Add New KPI to add a Key Performance Indicator. It lets you monitor the performance of your Cloudera DataFlow deployment.

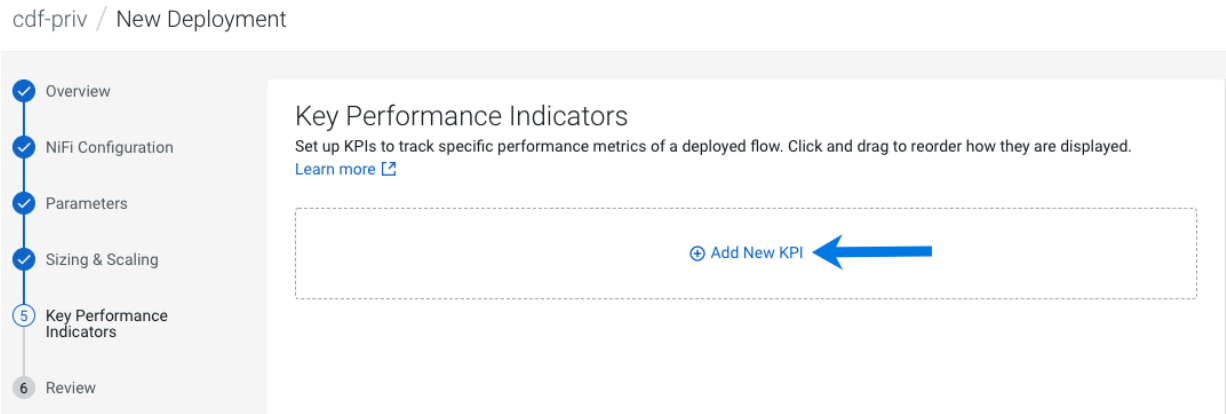

11. To learn about the use of Key Performance Indicaators (KPIs), add a KPI that will track how much data the processor is writing to the destination files. The KPI will raise an alert whenever this value goes below 1MB.

Set the following properties:

KPI Scope

select Processor.

Processor Name

select Write "Added Content" Events to File.

Metric to Track

select Bytes Sent

Alerts

select Trigger alert when metric is less than and set value to 1 MBytes.

Alert will be triggered when metric is outside the boundary(s) for

set to 2 Minutes.

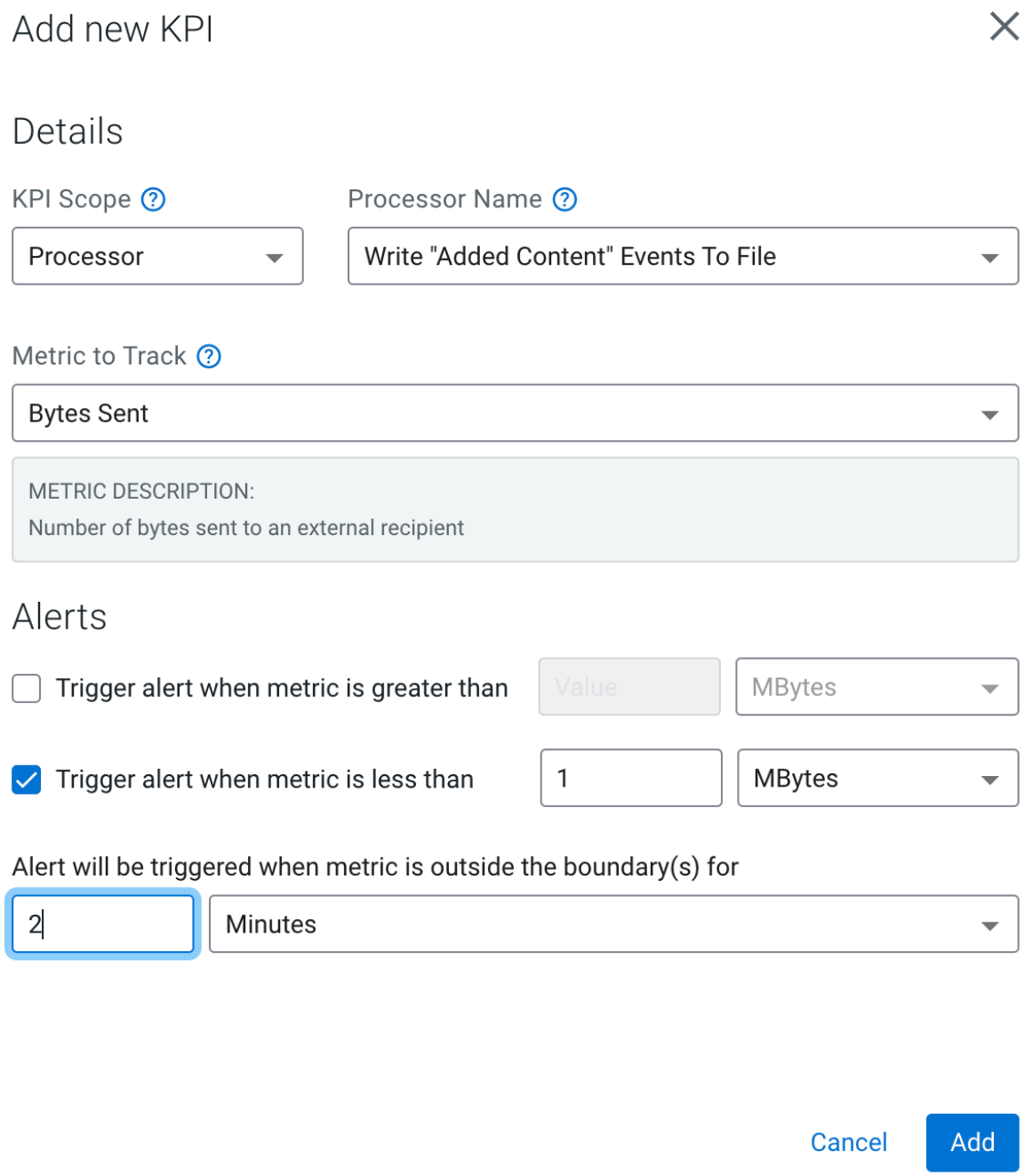

12. Click Add, to create the KPI.
13. Click Next.
14. Review a summary of the information provided. When you are finished, complete flow deployment by clicking Deploy.

Results

Once you click Deploy, you are redirected to the Alerts tab in the detail view for the deployment where you can track its progress.

Related Information

Deploying Flow Definitions
Monitoring and Managing Flow Deployments

KPI overview
Working with KPIs

Cloudera DataFlow

DataFlow Functions Quickstart

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Scope 4

Use case 4

Flow definition overview 4

Prerequisites for running the Quickstart 8

AWS Lambda 9

Azure Functions 13

Google Cloud Functions 15

Scope

This quickstart helps you to get a better understanding of Cloudera DataFlow Functions and how it works. It also walks you through deploying your first function on any of the three main cloud providers (AWS Lambda, Azure Functions, and Google Cloud Functions).

Use this guide if you want to try deploying your first serverless NiFi flow. Get started with Cloudera DataFlow Functions by walking through a few simple steps.

Use case

The quickstart is based on a 'Resize Image web service' use case.

This use case shows you how to deploy a serverless microservice to resize images using Apache NiFi on any of the three main cloud providers. You deploy a function that is triggered by an HTTP call. In this HTTP call, you send an image with optional HTTP headers to specify the new dimensions of the image. The Cloudera DataFlow function will return the resized image.

Flow definition overview

The NiFi flow definition used in this quickstart is based on Apache NiFi 1.18.0 and you can download this flow definition from here.

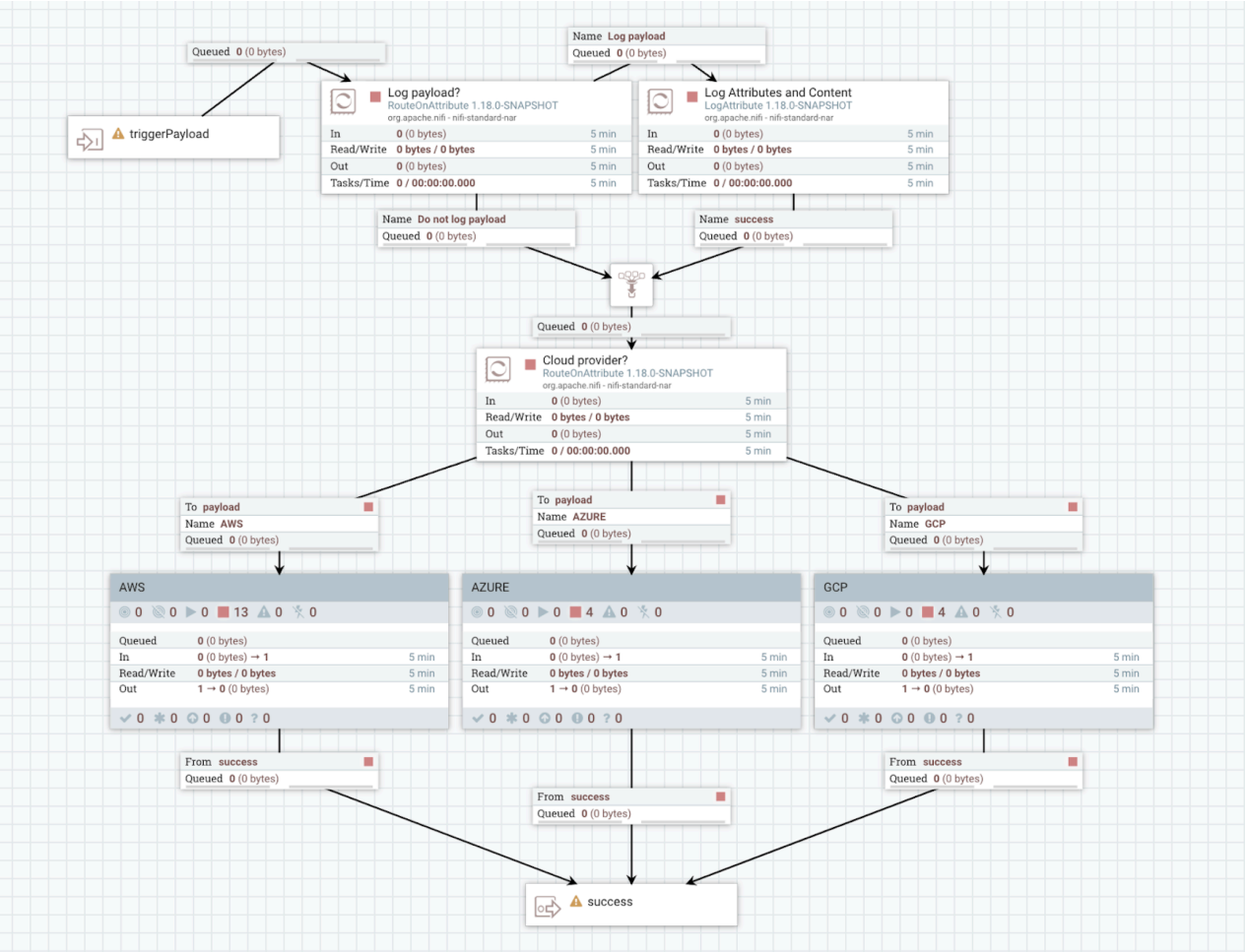

The triggerPayload input port is where the Cloudera DataFlow Functions framework is going to ingest the payload of the trigger that is generated by the cloud provider.

Then you decide whether or not you want to log the attributes and payload of the flowfile in the logs of the function. This is particularly useful when deploying the function for the first time as it helps you to understand what is going on and what is being generated. It would disabled for a function running in production. The logging is enabled by default but you can turn it off by setting the below variable in the function's configuration: logPayload = false

Then you have a RouteOnAttribute processor that redirects the flow file into a process group dedicated to each cloud provider depending on where the function is being deployed. The attribute containing the cloud provider name is automatically generated by the framework.

Note: This flow definition has been designed so that the same flow can be used on any cloud provider. If the function is deployed on a single cloud provider, the flow design could be greatly simplified by removing the parts related to the other cloud providers.

AWS Process Group

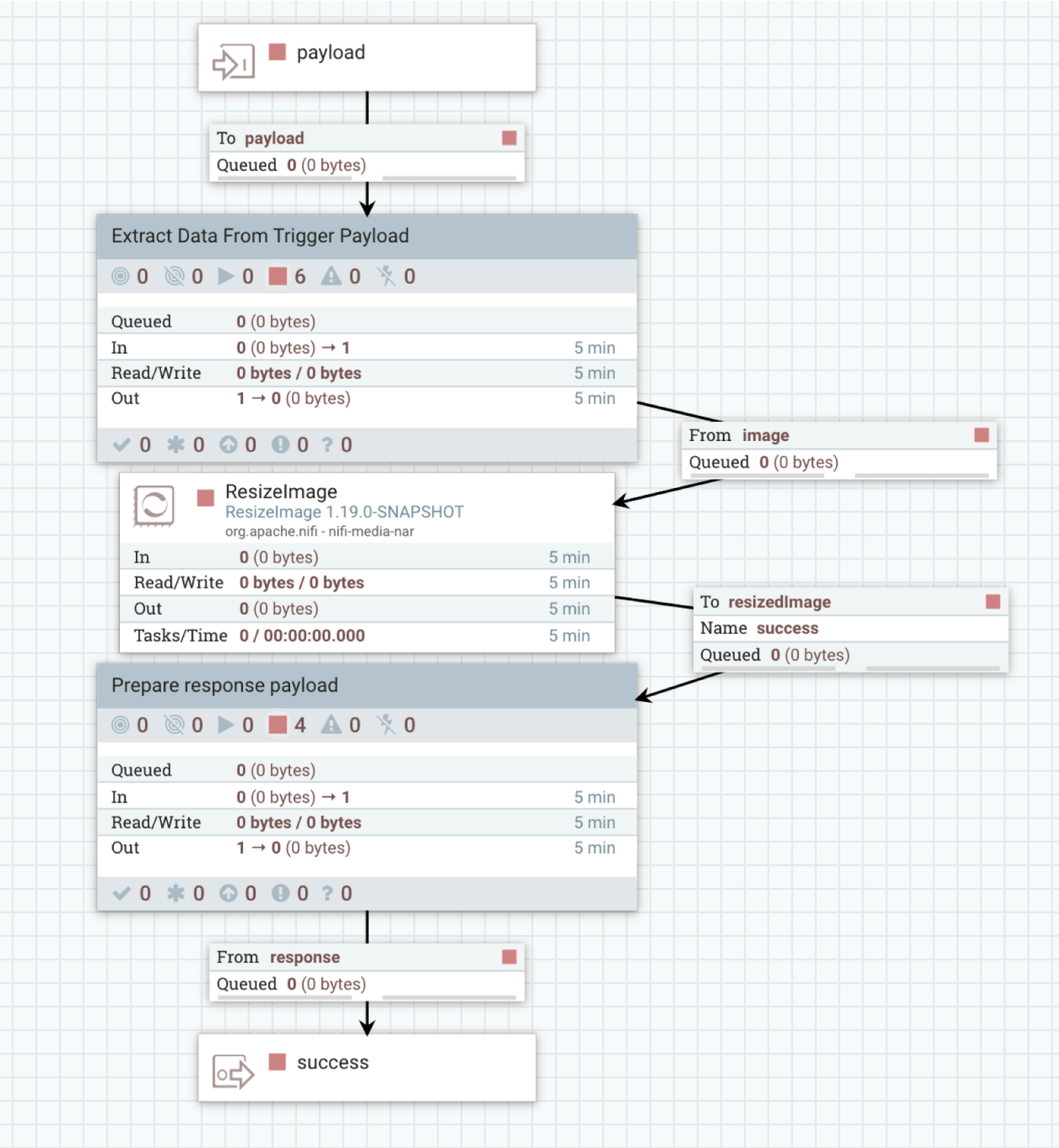

The payload of the flowfile generated by the API Gateway trigger is a JSON for which one of the fields will be the binary data of your image with base 64 encoding.

This is why you first need to extract the fields you are interested in, then do the base 64 decoding to have the binary data representing the image as the flow file content. After that you execute the ResizeImage processor, then re-encode the result with base 64 encoding and finally you generate the JSON payload that is expected by the AWS API Gateway to send back the response to the HTTP client.

Azure and GCP Process Groups

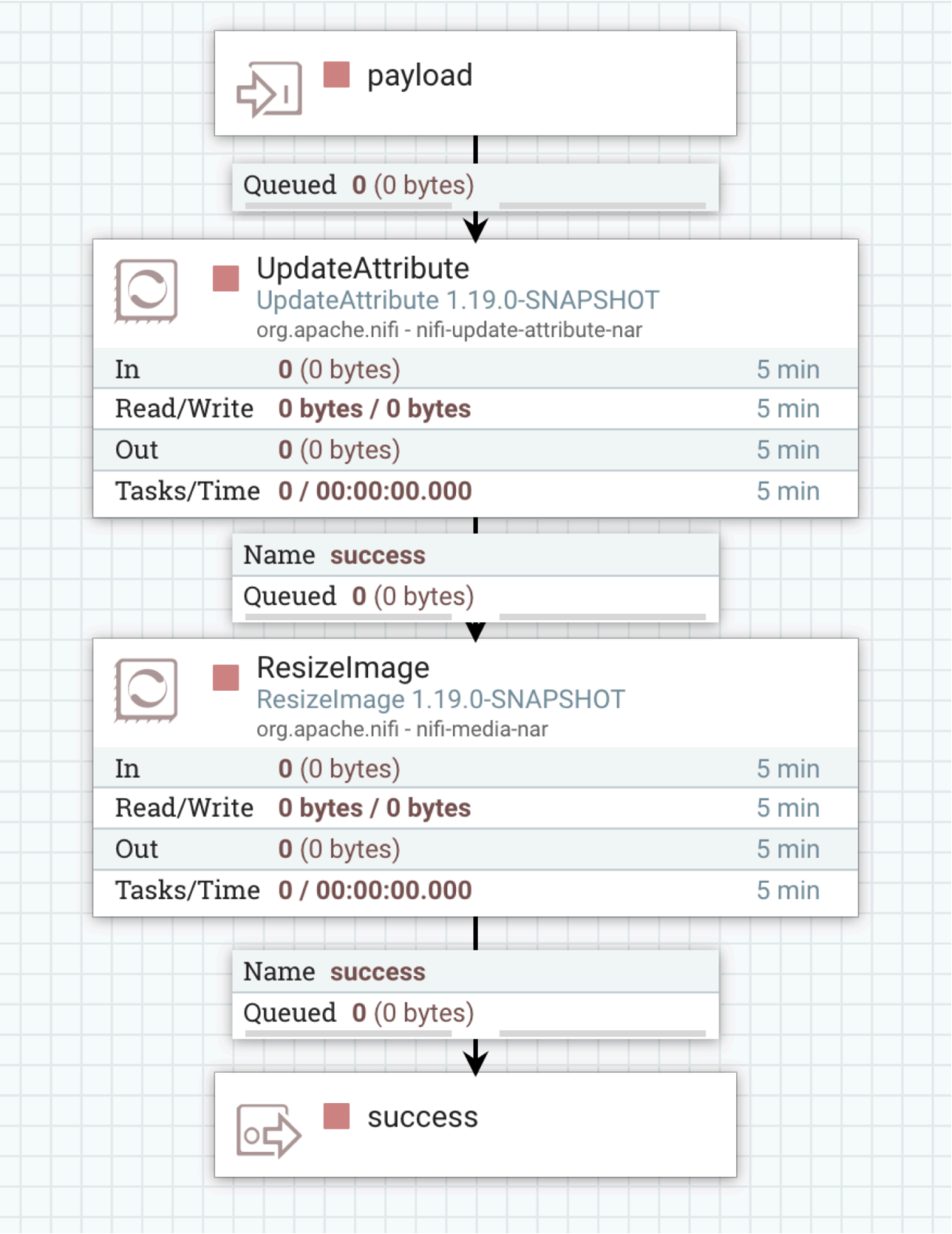

The binary data is the content of the generated flow file, so there is no need to do any conversion and you can directly use the ResizeImage processor based on the flow file attributes.

Note: For Azure Functions, the HTTP request has to contain the HTTP header Content-Type = applicat ion/octet-stream, so you have to override the mime.type attribute of the flow file before sending it to the ResizeImage to use an image friendly MIME type.

Prerequisites for running the Quickstart

This section helps you to examine the list of actions you must perfom before you start working on your function.

Procedure

1. Get the CRN (Customer Resource Number) of the flow definiton.
   a. Download the flow definition for this use case and upload it into your Cloudera DataFlow Catalog.
   b. Once you have the flow definition in the Catalog, copy its CRN with its version.

      For example:crn:cdp:df:us-west-1:558bc1d3-8867-4357-8524-311d51289233:flow:Resize-Image/v.1

      Note: Make sure you have the full CRN with the flow version as the end. For more information, see Retrieving data flow CRN.
2. Make sure that you have a Cloudera machine user created.

   This machine user is used when the function is triggered to retrieve the flow definition to be executed. For more information, see Creating CDP service account.
3. Give the Cloudera machine user the appropriate role and generate the necessary keys.

   For more information, see Provisioning Access Key ID and Private Key.
4. Download the Cloudera DataFlow Functions binary from the Functions page of Cloudera DataFlow for the cloud provider of your choice.

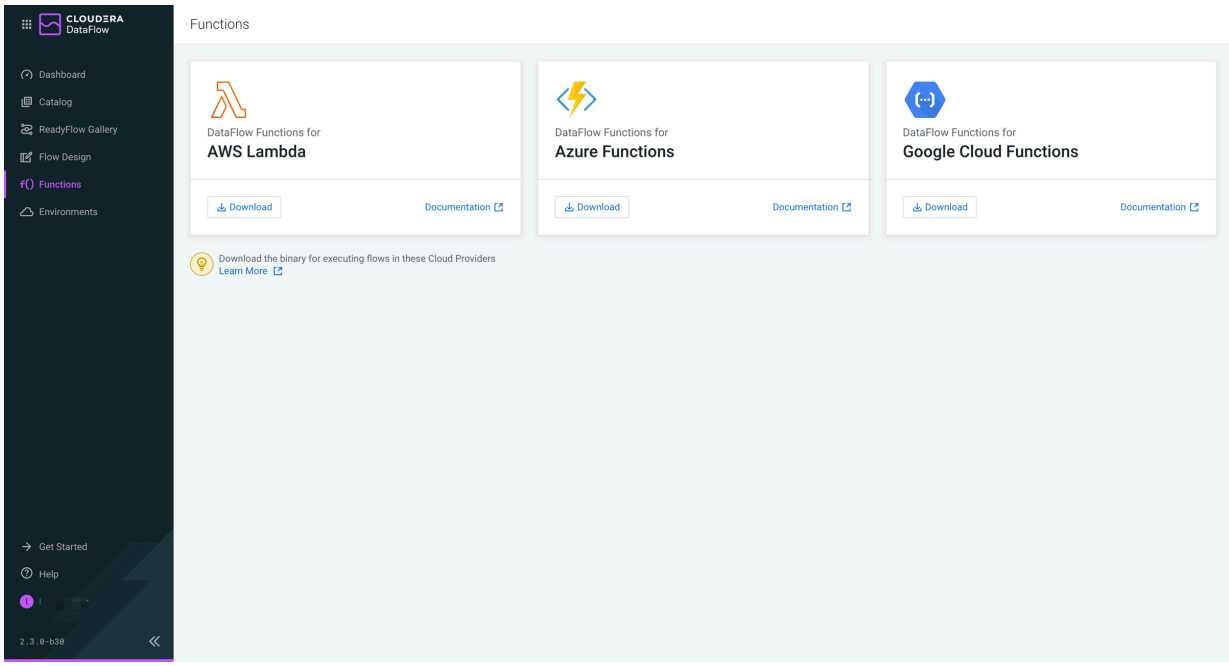

Note: This quickstart leverages the CLI of the cloud providers to make deploying the function straightforward. If you want to deploy your function using the control plane UI of the cloud provider, see the appropriate section of the Cloudera DataFlow Functions documentation. The instructions provided in this quickstart are based on using the CLI of the cloud provider.

AWS Lambda

This section helps you to check the list of actions you must perfom before you start working on your function.

Before you begin

- You have the latest version of the AWS Lambda CLI.
- You have jq installed to extract some information from the JSON responses when executing commands.

Procedure

1. Start by setting all the parameters required for deploying your function using the AWS CLI.

```
# Name of the binary file you downloaded from Cloudera DataFlow. Change it
 based on the version you downloaded.
FILEKEY="NAAF-AWS-LAMBDA-1.0.0.2.3.6.0-35-BIN.ZIP"

# Location where the function will be deployed. Change this value to ma
tch your needs/requirements.
LOCATION=EU-WEST-2

# Name of the bucket where the binary and NAR will be uploaded. Change thi
s value to match your needs/requirements.
BUCKET_NAME=MYDFFRESIZEIMAGEBUCKETPVILLARD

# Name of the function (only lower case letters). Change this value to mat
ch your needs/requirements.
FUNCTIONNAME=RESIZEIMAGEPVILLARD

# Flow CRN. Add the Customer Resource Number you obtained from the Clouder
a DataFlow Catalog.
FLOW_CRN=***YOUR VALUE HERE***

# Credentials of the machine user
DF_ACCESS_KEY=***YOUR VALUE HERE***
DF_PRIVATE_KEY=***YOUR VALUE HERE***
```

2. Download the function template from here and make it available locally.
3. Execute the below commands:
   a. Create your S3 bucket.

```
aws s3api create-bucket \
    --bucket $BUCKET_NAME \
    --region $LOCATION \
    --create-bucket-configuration LocationConstraint=$LOCATION
```

   b. Copy the Lambda Cloudera DataFlow Function binariy zip file to your bucket.

```
aws s3 cp $FILEKEY s3://$BUCKET_NAME/$FILEKEY
```

   c. Create the role for the Lambda.

```
ROLEARN=`aws iam create-role --role-name $FUNCTIONNAME\_lambda\_role --a
ssume-role-policy-document '{"Version": "2012-10-17","Statement": [{ "Ef
fect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Actio
n": "sts:AssumeRole"}]}' | jq -r ".Role.Arn" `
```

```
aws iam attach-role-policy --role-name $FUNCTIONNAME\_lambda\_role --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionR
ole
sleep 10
```

d. Create the function.

```
cp my-function-definition.json template.json

sed -i '.bak' -e 's|BUCKETNAME|'"$BUCKET_NAME"'|g' \
  -e 's|FUNCTIONNAME|'"$FUNCTIONNAME"'|g' \
  -e 's|FLOWCRN|'"$FLOW_CRN"'|g' \
  -e 's|DFACCESSKEY|'"$DF_ACCESS_KEY"'|g' \
  -e 's|DFPRIVATEKEY|'"$DF_PRIVATE_KEY"'|g' \
  -e 's|LAYERARN|'"$LAYERARNVERSION"'|g' \
  -e 's|FILEKEY|'"$FILEKEY"'|g' \
  -e 's|ROLEARN|'"$ROLEARN"'|g' template.json
aws lambda create-function --cli-input-json file://template.json

echo "please wait a bit more (30 sec)..."
sleep 30

aws lambda publish-version --function-name $FUNCTIONNAME
```

When the execution is complete (it takes a few minutes), the function will be created and a version will be published but without a trigger.

4. Open the AWS Console UI and go to your Lambda.

5. Find to the published version of the function and add the API Gateway trigger:

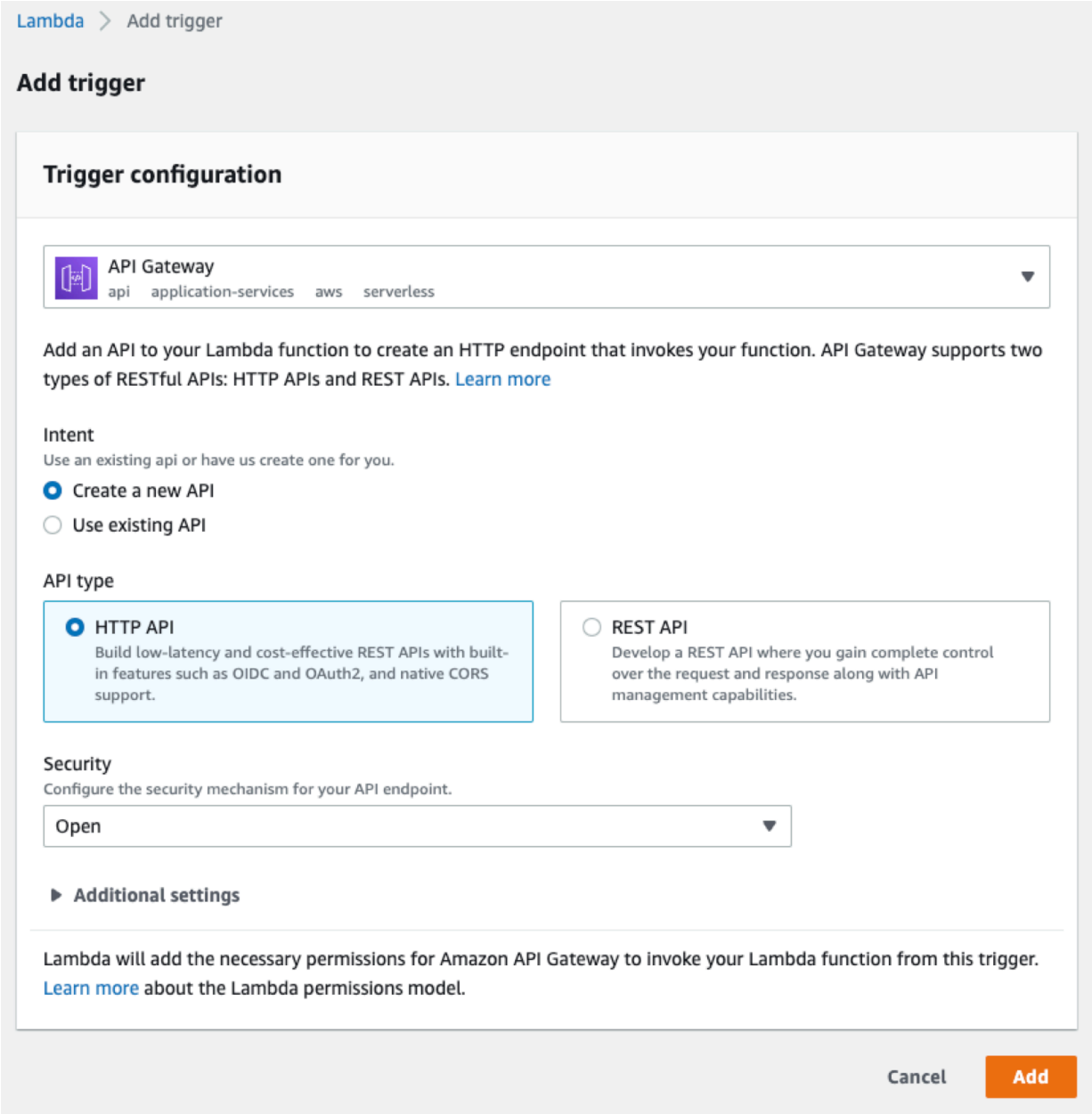

This gives you the endpoint on which the Lambda is listening.

For example: https://aa2q99qo3i.execute-api.eu-west-2.amazonaws.com/default/resizeimage

You can now use a tool like Postman to send your picture.

You can add two custom headers (resize-width and resize-height) to specify the resizing dimensions. If not specified, the default values of the NiFi flow parameters are used.

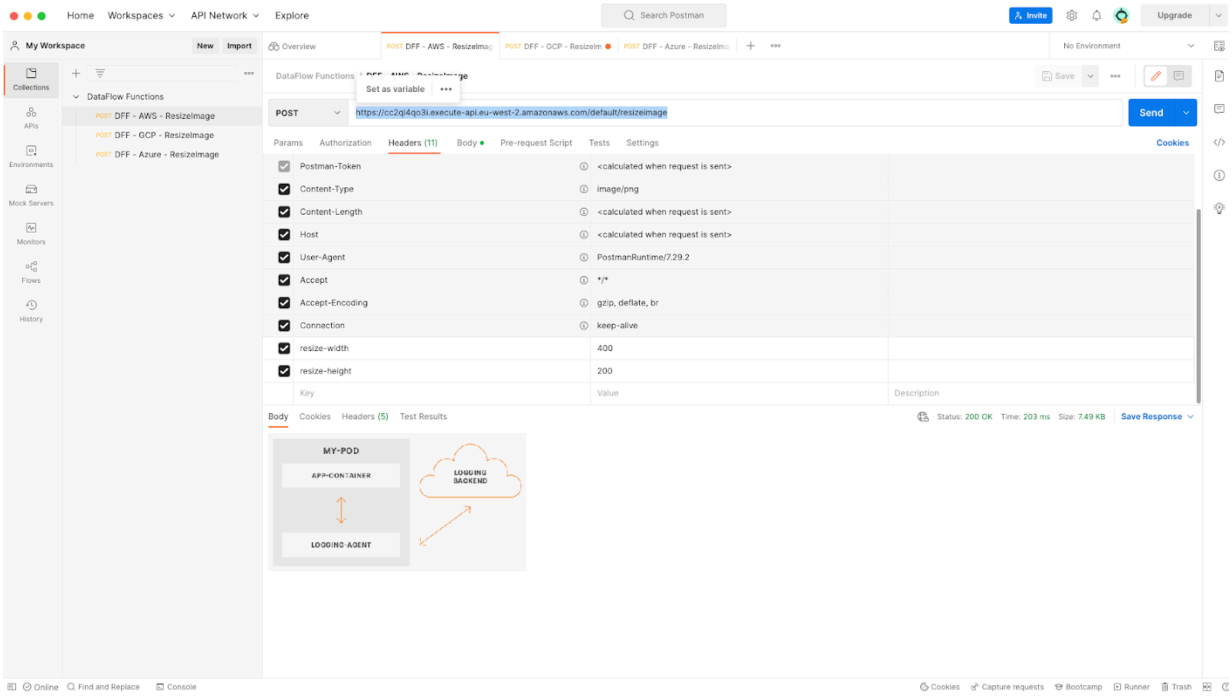

Note: The first attempt may fail due to a cold start and timeout (30 sec timeout on the API Gateway). A subsequent retry after a minute should work and should return the resized image much faster. For more information, see Cold start.

It is possible to send a request to your function using curl.

For example:

```
curl -X POST https://3bftvfa66m.execute-api.us-east-2.amazonaws.com/defa
ult/cpresizeimage \
-H "Content-Type: image/png" \
-H "resize-width: 400" \
-H "resize-height: 200" \
--data-binary "@/Users/pierre/Desktop/test.png" \
--output /tmp/my_resized_image.png
```

Note: The function will be open to the internet. Make sure to secure it if you want to keep it running (this is done when creating the API Gateway trigger) or delete everything once you are done with this quickstart.

You can delete the function using the following command:

```
aws lambda delete-function --function-name $FUNCTIONNAME --region $LOCAT
ION
aws iam detach-role-policy --role-name $FUNCTIONNAME\_lambda\_role --po
licy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
aws iam delete-role --role-name $FUNCTIONNAME\_lambda\_role

aws s3 rm s3://$BUCKET_NAME --recursive --region $LOCATION
aws s3api delete-bucket --bucket $BUCKET_NAME --region $LOCATION
rm template.json template.json.bak
```

You may also want to manually delete the API Gateway that you created earlier.

Azure Functions

This section helps you to check the list of actions you must perfom before you start working on your function.

Before you begin

- You have the latest version of the Azure Functions CLI.
- You have jq installed to extract some information from the JSON responses when executing commands.

Procedure

1. Start by setting all the parameters required for deploying your function using the Azure CLI.

```
# Name of the binary you download from the Cloudera Public Cloud control
 plane.
FILEKEY="NAAFAZUREFUNCTIONS.ZIP"

# Location where the function will be deployed. Change this value to match
 your needs/requirements.
LOCATION=FRANCECENTRAL

# Name of the file share where the binary will be uploaded. Change this va
lue to match your needs/requirements.
BUCKET_NAME=MYDFFRESIZEIMAGEBUCKET

# Name of the function (only lower case letters). Change this value to mat
ch your needs/requirements.
FUNCTIONNAME=RESIZEIMAGEABC

# Flow CRN. Add the Customer Resource Number you obtained from the Clouder
a DataFlow Catalog.
FLOW_CRN=***YOUR VALUE HERE***

# Credentials of the machine user
DF_ACCESS_KEY=***YOUR VALUE HERE***
DF_PRIVATE_KEY=***YOUR VALUE HERE***

# Name of the resource group that is going to be created for the function.
 Change this value to match your needs/requirements.
RESOURCEGROUP=MYDFFRESOURCEGROUP

# Subscription ID to use. Change this value to match your needs/requiremen
ts.
SUBSCRIPTIONID=***YOUR VALUE HERE***

# Name of the hosting plan that is going to be created for the function. C
hange this value to match your needs/requirements.
HOSTINGPLANNAME=MYHOSTINGPLANDFF

# Name of the storage account that is going to be created (only lower case
 letters). Change this value to match your needs/requirements.
STORAGEACCOUNTNAME=MYDFFSTORAGEACCOUNTNAME
```

2. Download the deployment template from here.
3. Execute the below commands:
   a. Create a resource group.

```
az group create -l $LOCATION -n $RESOURCEGROUP
```

```
echo "please
```

b. Deploy the Function App.

```
az deployment group create --resource-group $RESOURCEGROUP \
 --template-file template.json \
 --parameters \
     subscriptionId=$SUBSCRIPTIONID \
     name=$FUNCTIONNAME \
     location=$LOCATION \
     hostingPlanName=$HOSTINGPLANNAME \
     resourceGroup=$RESOURCEGROUP \
     storageAccountName=$STORAGEACCOUNTNAME \
     FLOW_CRN=$FLOW_CRN \
     DF_ACCESS_KEY=$DF_ACCESS_KEY \
     DF_PRIVATE_KEY=$DF_PRIVATE_KEY

echo "please wait a bit more (10 sec)..."
sleep 10
```

c. Retrieve the file share name created in the previous step.

```
FILESHARE=`az storage share list --account-name $STORAGEACCOUNTNAME | jq
 -r '.[0].name'`
```

d. Create the right directories for the binary.

```
az storage directory create --account-name $STORAGEACCOUNTNAME --share-n
ame $FILESHARE --name data
az storage directory create --account-name $STORAGEACCOUNTNAME --share-
name $FILESHARE --name data/SitePackages
```

e. Upload the binary and the metadata file.

```
az storage file upload -s $FILESHARE --source $FILEKEY --path data/SiteP
ackages/ --account-name $STORAGEACCOUNTNAME
echo "$FILEKEY" > /tmp/packagename.txt && az storage file upload -s $FI
LESHARE --source /tmp/packagename.txt --path data/SitePackages/ --accoun
t-name $STORAGEACCOUNTNAME && rm /tmp/packagename.txt
```

When the execution is complete (it takes a few minutes), the function will be publicly listening on:

```
https://<function
```

```
name>.azurewebsites.net/api/StatelessNiFiHttpBinaryTriggerFunction
```

You can now use a tool like Postman to send your picture.

Make sure to override the Content-Type header with application/octet-stream.

You can add two custom headers (resize-width and resize-height) to specify the resizing dimensions. If not specified, the default values of the NiFi flow parameters are used.

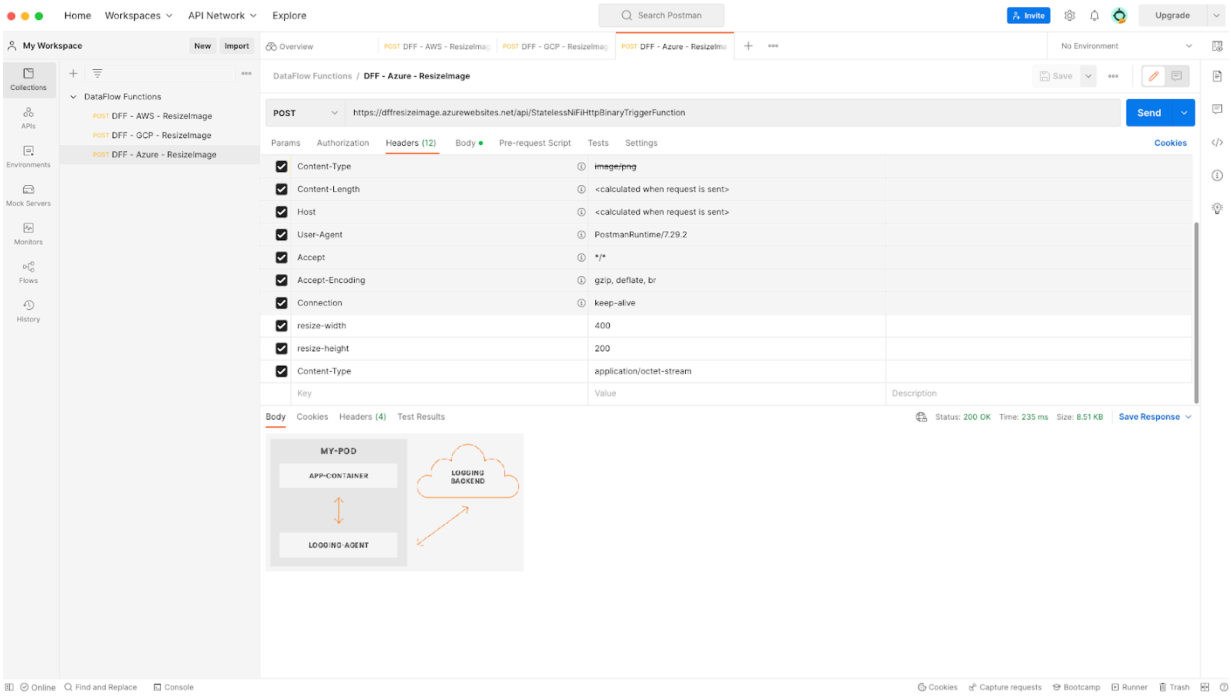

Note: The first attempt may fail due to a cold start and timeout. A subsequent retry after a minute should work and should return the resized image much faster. For more information, see Cold start.

It is possible to send a request to your function using curl.

For example:

```
curl -X POST https://myresizeimage.azurewebsites.net/api/StatelessNiFiHt
tpBinaryTriggerFunction \
-H "Content-Type: application/octet-stream" \
-H "resize-width: 400" \
-H "resize-height: 200" \
--data-binary "@/Users/pierre/Desktop/test.png" \
--output /tmp/my_resized_image.png
```

Note: The function will be open to the internet. Make sure to secure it if you want to keep it running or delete everything once you are done with this quickstart. For more information, see Securing your Azure Function App.

You can delete the function using the following command: az group delete --name $RESOURCEGROUP --yes

Google Cloud Functions

This section helps you to check the list of actions you must perfom before you start working on your function.

Before you begin

- You have the latest version of the Google Cloud Functions (GCP) CLI.
- The scripts assume that a default project has been associated to your local client:gcloud config set project PR OJECT_ID.

Procedure

1. Start by setting all the parameters required for deploying your function using the GCP CLI.

```
# Name of the binary file you downloaded from the Cloudera Public Cloud
control plane. Change it based on the version you downloaded.
FILEKEY="NAAF-GCP-CLOUD-FUNCTIONS-1.0.0.2.3.6.0-35-BIN.ZIP"


# Location where the function will be deployed. Change this value to match
 your needs/requirements.
LOCATION=EUROPE-WEST1


# Name of the bucket where the binary and NAR will be uploaded. Change thi
s value to match your needs/requirements.
BUCKET_NAME=MYDFFRESIZEIMAGEBUCKET

# Name of the function (only lower case letters). Cange this value to matc
h your needs/requirements.
FUNCTIONNAME=RESIZEIMAGE

# Flow CRN. Add the Customer Resource Number you obtained from the Clouder
a DataFlow Catalog.
FLOW_CRN=***YOUR VALUE HERE***

# Credentials of the machine user
DF_ACCESS_KEY=***YOUR VALUE HERE***
DF_PRIVATE_KEY=***YOUR VALUE HERE***
```

2. Create a bucket where you will upload the Cloudera DataFlow Functions binary.

```
gcloud alpha storage buckets create gs://$BUCKET_NAME --location $LOCATION
gsutil cp $FILEKEY gs://$BUCKET_NAME/$FILEKEY
```

3. Deploy the function.

```
gcloud functions deploy $FUNCTIONNAME \
 --gen2 \
 --region=$LOCATION \
 --runtime=java11 \
 --source=gs://$BUCKET_NAME/$FILEKEY \
 --memory=1024MB \
 --timeout=300 \
 --trigger-http \
 --allow-unauthenticated \
 --entry-point=com.cloudera.naaf.gcp.cloud.functions.StatelessNiFiHttpFu
nction \
```

```
--set-env-vars=FLOW_CRN=$FLOW_CRN,DF_PRIVATE_KEY=$DF_PRIVATE_KEY,DF_ACC
ESS_KEY=$DF_ACCESS_KEY,NEXUS_URL=https://maven-central.storage-download.
googleapis.com/maven2,STORAGE_BUCKET=$BUCKET_NAME,OUTPUT_PORT=success
```

After a couple of minutes, the function will be deployed and you will receive an HTTP endpoint exposed by Google Cloud to call your function.

For example: https://resizeimage-vqz99abcdea-ew.a.run.app

You can now use a tool like Postman to send your picture.

You can add two custom headers (resize-width and resize-height) to specify the resizing dimensions. If not specified, the default values of the NiFi flow parameters are used.

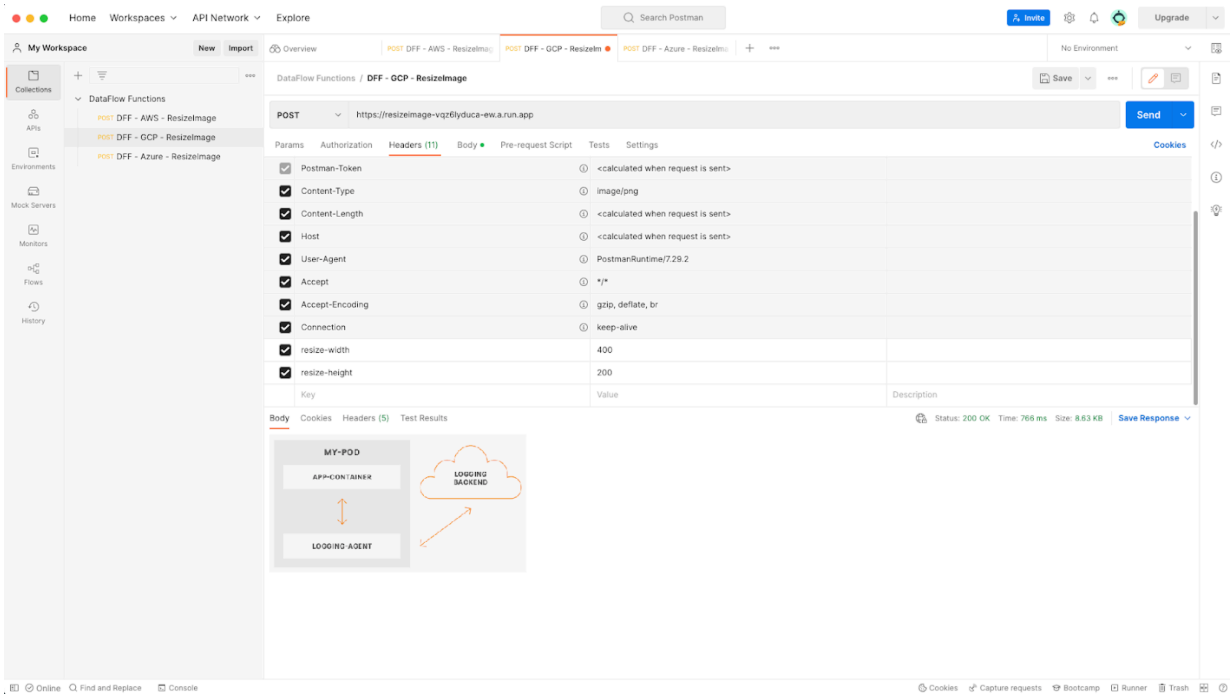

Note: The first attempt may fail due to cold start and timeout. A subsequent retry should work and should return the resized image much faster. For more information, see Cold start.

It is possible to send a request to your function using curl.

For example:

```
curl -X POST https://resizeimage-vqz99abcdea-ew.a.run.app \
-H "Content-Type: image/png" \
-H "resize-width: 400" \
-H "resize-height: 200" \
--data-binary "@/Users/pierre/Desktop/test.png" \
--output /tmp/my_resized_image.png
```

Note: The function will be open to the internet. Make sure to secure it if you want to keep it running or delete everything once you have completed this quickstart.

You can secure the function by deploying it without the --allow-unauthenticated parameter.

You can delete the function using the following command:

```
gcloud functions delete $FUNCTIONNAME --gen2 --region=$LOCATION --quiet
gcloud alpha storage rm -r gs://$BUCKET_NAME
```

Cloudera DataFlow

ADLS to ADLS Avro

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: ADLS to ADLS Avro..... 4

Prerequisites..... 4

List of required configuration parameters for the ADLS to ADLS Avro ReadyFlow..... 7

ReadyFlow overview: ADLS to ADLS Avro

You can use the ADLS to ADLS Avro ReadyFlow to move data between source and destination ADLS locations while converting the files into Avro format.

This ReadyFlow consumes JSON, CSV or Avro files from a source Azure Data Lake Service (ADLS) location, converts the files into Avro and writes them to the destination ADLS location. You can specify the source format, the source and target location as well as the schema to use for reading the source data. The ReadyFlow polls the source container for new files (it performs a listing periodically).

ADLS to ADLS Avro ReadyFlow details	
Source	ADLS
Source Format	JSON, CSV, Avro
Destination	ADLS
Destination Format	Avro

Moving data with an ADLS to ADLS Avro flow

You can use an ADLS to ADLS Avro data flow when you want to move data from a location in ADLS to another ADLS location, and at the same time convert the data to Avro format. You need to specify the source format, the source and target location as well as the schema to use for handling the data. Your flow can consume JSON, CSV or Avro files from the source ADLS location. It converts the files into Avro format and writes them to the destination ADLS location. You define and store the data processing schema in the Schema Registry on a Streams Messaging Data Hub cluster. The data flow parses the schema by looking up the schema name in the Schema Registry.

Prerequisites

Learn how to collect the information you need to deploy the ADLS to ADLS Avro ReadyFlow, and meet other prerequisites.

For your data source and target

- You have the two ADLS directories and their paths as source and destination for the data movement.

- You have performed one of the following to configure access to the ADLS folders:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.
- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
```

```
        },
        {
           "name":"sensor_1",
           "doc":"Reading #1.",
           "type":"int"
        },
        {
           "name":"sensor_2",
           "doc":"Reading #2.",
           "type":"int"
        },
        {
           "name":"sensor_3",
           "doc":"Reading #3.",
           "type":"int"
        }
    ]
}
```

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

Related Concepts

List of required configuration parameters for the ADLS to ADLS Avro ReadyFlow

List of required configuration parameters for the ADLS to ADLS Avro ReadyFlow

When deploying the ADLS to ADLS Avro ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: ADLS to ADLS Avro ReadyFlow configuration parameters

ADLS to ADLS Avro ReadyFlow configuration parameters	
Parameter Name	**Description**
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores and to the Schema Registry. Ensure this user has the appropriate access rights to the object store locations and to the Schema Registry in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker) and the Schema Registry.
CSV Delimiter	If your source data is CSV, specify the delimiter here.

ADLS to ADLS Avro ReadyFlow configuration parameters	
Parameter Name	**Description**
Data Input Format	Specify the format of your input data. • CSV • JSON • AVRO
Destination ADLS File System	Specify the file system name you want to write to. The full path will be constructed from: abfs://#{Destination ADLS File System}@#{Destination ADLS Storage Account}.dfs.core.windows.net/#{Destination ADLS Path}
Destination ADLS Path	Specify the path within the ADLS data container where you want to write to without any leading characters. The full path will be constructed from: abfs://#{Destination ADLS File System}@#{Destination ADLS Storage Account}.dfs.core.windows.net/#{Destination ADLS Path}
Destination ADLS Storage Account	Specify the storage account name you want to write to. The full path will be constructed from: abfs://#{Destination ADLS File System}@#{Destination ADLS Storage Account}.dfs.core.windows.net/#{Destination ADLS Path}
Schema Name	Specify the schema name to be looked up in the Schema Registry used to parse the source files.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.
Source ADLS File System	Specify the file system name you want to read from. The full path will be constructed from: abfs://#{Source ADLS File System}@#{Source ADLS Storage Account}.dfs.core.windows.net/#{Source ADLS Path}
Source ADLS Path	Specify the path within the ADLS data container where you want to read from without any leading characters. The full path will be constructed from: abfs://#{Source ADLS File System}@#{Source ADLS Storage Account}.dfs.core.windows.net/#{Source ADLS Path}
Source ADLS Storage Account	Specify the storage account name you want to read from. The full path will be constructed from: abfs://#{Source ADLS File System}@#{Source ADLS Storage Account}.dfs.core.windows.net/#{Source ADLS Path}

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

ADLS to Databricks

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: ADLS to Databricks **4**

Prerequisites **4**

List of required configuration parameters for the ADLS to Databricks ReadyFlow **7**

ReadyFlow: ADLS to Databricks

You can use the ADLS to Databricks ReadyFlow to retrieve CSV files from a source ADLS location and write them as Parquet files to a destination ADLS location and Databricks table.

This ReadyFlow consumes CSV data from a source ADLS location, parses the data using schema(s) provided by the Cloudera Schema Registry, converts it to Parquet format and writes the data to a destination ADLS location and Databricks table. The flow supports either a non-partitioned table or a partitioned table (single column only).

Note:

By default, the flow is configured for a partitioned Databricks table. This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up an IDBroker mapping or Ranger policies when using fine-grained object store access allowing your workload user access to the source and destination ADLS locations.

ADLS to Databricks ReadyFlow details	
Source	Cloudera managed ADLS
Source Format	CSV
Destination	Cloudera managed ADLS and Databricks
Destination Format	Parquet

Moving data to object stores

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost-effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Prerequisites

Learn how to collect the information you need to deploy the ADLS to Databricks ReadyFlow, and meet other prerequisites.

For your ADLS data source

- You have your ADLS container, storage account, and path from which you want to ingest data.

- You have performed one of the following to configure access to the ADLS folders:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.
- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
```

```
        },
        {
           "name":"sensor_1",
           "doc":"Reading #1.",
           "type":"int"
        },
        {
           "name":"sensor_2",
           "doc":"Reading #2.",
           "type":"int"
        },
        {
           "name":"sensor_3",
           "doc":"Reading #3.",
           "type":"int"
        }
    ]
}
```

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

Databricks target requirements

- You have created a Databricks table, non-partitioned or partitioned (single column only).
- You have the Storage Location of your Databricks Table, which consists of the ADLS Container, Storage Account, Path and Table Id.

Related Concepts

List of required configuration parameters for the ADLS to Databricks ReadyFlow

List of required configuration parameters for the ADLS to Databricks ReadyFlow

When deploying the ADLS to Databricks ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: ADLS to Databricks ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to the object stores and to the schema registry. Ensure this user has the appropriate access rights to the object store locations and to the schema registry in Ranger or IDBroker.

Parameter Name	Description
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores and the schema registry.
CDPEnvironment	The Cloudera Environment configuration resources.
Destination ADLS File System	Specify the name of the ADLS data container you want to write to. The full path is constructed from: abfs://#*[***DESTINATION ADLS FILE SYSTEM***]@#[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net/#*[***DESTINATION ADLS PATH***]*
Destination ADLS Path	Specify the path within the ADLS data container where you want to write to. Make sure that the path starts with "/". The path has to end with the destination Databricks Table Id. The full path is constructed from: abfs://#*[***DESTINATION ADLS FILE SYSTEM***]@#[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net/#*[***DESTINATION ADLS PATH***]*
Destination ADLS Storage Account	Specify the destination ADLS storage account name.
Partition Column	Specify the name of the column used to partition your destination Databricks table. This ReadyFlow only supports a single partition column.
Partition Column Exists	Specify whether the destination Databricks column is partitioned. The default value is YES.
Schema Name	Specify the schema name to be looked up in the Schema Registry used to parse the source files.
Schema Name 2	If your Databricks table is partitioned, specify the name of the modified schema to be looked up in the Schema Registry. This schema should not include the partition column field.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.
Source ADLS File System	Specify the name of the ADLS data container you want to read from. The full path is constructed from: abfs://#*[***SOURCE ADLS FILE SYSTEM***]@#[***SOURCE ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net/#*[***SOURCE ADLS PATH***]*
Source ADLS Path	The full path will be constructed from: abfs://#*[***SOURCE ADLS FILE SYSTEM***]@#[***SOURCE ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net/#*[***SOURCE ADLS PATH***]*
Source ADLS Storage Account	Specify the source ADLS storage account name.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

ADLS to Milvus

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: ADLS to Milvus [Technical Preview]..... 4

Prerequisites..... 4

List of required configuration parameters for the ADLS to Milvus [Technical Preview] ReadyFlow.....7

ReadyFlow overview: ADLS to Milvus [Technical Preview]

You can use the ADLS to Milvus [Technical Preview] ReadyFlow to consume PDF documents from ADLS, vectorize them using a HuggingFace model and write the results to Milvus.

This ReadyFlow consumes PDF documents from a source ADLS location, partitions the PDFs, chunks the data, vectorizes the data using a HuggingFace embedding model, and stores the results in Milvus vector DB. The default HuggingFace model is 'all-MiniLM-L12-v2'. A Milvus access token is required to run this flow. Define a KPI on the failure_WriteToMilvus connection to monitor failed write operations.

Note: This ReadyFlow is considered Technical Preview and is not designed for production use. The flow uses Python processors and must be run in NiFi 2.x.

To ensure correct functionality, make sure your Milvus collection has been created with the expected schema and the same dimension as the 'all-MiniLM-L12-v2' model (384). The images within PDFs will be ignored.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

Note:

This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the source ADLS location.

ADLS to Milvus [Technical Preview] ReadyFlow details	
Source	Cloudera managed ADLS
Source Format	PDF
Destination	Milvus
Destination Format	Vector DB

Prerequisites

Learn how to collect the information you need to deploy the ADLS to Milvus [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have your ADLS container, storage account, and path from which you want to ingest data.

- You have performed one of the following to configure access to the ADLS folders:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.

       a. To access IDBroker mappings in your environment, click Actions Manage Access .

       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- Your Milvus version is 2.4.4.
- You have the Milvus access token.
- You have the name of the destination Milvus collection.
- You have the URI of the destination Milvus instance.
- Your Milvus collection schema has the following field names and field types:
  - id *(INT64)* You must enable Auto ID on this field.
  - text_embedding *(FLOAT_VECTOR)*
  - source *(VARCHAR)*
  - section *(VARCHAR)*
  - text *(VARCHAR)*
- The ‘text_embedding’ field in your destination Milvus collection is configured with the same dimensions as the default HuggingFace 'all-MiniLM-L12-v2' model (384).
- The source, section and text VARCHAR fields have been configured with Max Length values large enough to accommodate their respective values in your PDFs.

Related Concepts

List of required configuration parameters for the ADLS to Milvus [Technical Preview] ReadyFlow

List of required configuration parameters for the ADLS to Milvus [Technical Preview] ReadyFlow

When deploying the ADLS to Milvus [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: ADLS to Milvus [Technical Preview] ReadyFlow configuration parameters

Parameter name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores.
CDPEnvironment	The Cloudera Environment configuration resources.
HuggingFace Embedding Model	Specify the HuggingFace model name to use for embedding the data. The default model is 'all-MiniLM-L12-v2'.
Milvus Access Token	Specify the access token for the destination Milvus instance.
Milvus Collection Name	Specify the name of the destination Milvus collection.
Milvus Instance URI	Specify the URI of the destination Milvus instance.
Source ADLS File System	Specify the name of the ADLS data container you want to read from. The full path will be constructed from: abfs://#{Source ADLS File System}@#{Source ADLS Storage Account}.dfs.core.windows.net/#{Source ADLS Path}
Source ADLS Path	Specify the path within the ADLS data container where you want to read from without any leading characters. The full path will be constructed from: abfs://#{Source ADLS File System}@#{Source ADLS Storage Account}.dfs.core.windows.net/#{Source ADLS Path}
Source ADLS Storage Account	Specify the source ADLS storage account name.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

ADLS to Pinecone

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: ADLS to Pinecone [Technical Preview]....4

Prerequisites.... 4

List of required configuration parameters for the ADLS to Pinecone [Technical Preview] ReadyFlow....6

ReadyFlow overview: ADLS to Pinecone [Technical Preview]

You can use the ADLS to Pinecone [Technical Preview] ReadyFlow to consume PDF documents from ADLS, vectorize them using an OpenAI model and write the results to Pinecone.

This ReadyFlow consumes PDF documents from a source ADLS location, partitions the PDFs, chunks the data, and transforms/splits the data into the required input format for the PutPinecone processor. The PutPinecone processor then vectorizes the data and writes the embeddings to a destination Pinecone index. The default OpenAI model is 'text-embedding-ada-002'. An OpenAI API key and a Pinecone API key are required to run this flow. Define a KPI on the failure_WriteToPinecone connection to monitor failed write operations.

Note: This ReadyFlow is considered Technical Preview and is not designed for production use.

The flow uses Python processors and must be run in NiFi 2.x.

To ensure correct functionality, make sure your Pinecone index has been created with the same dimension as the 'text-embedding-ada-002' model (1536). The images within PDFs will be ignored.

Note:

This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the source ADLS location.

ADLS to Pinecone [Technical Preview] ReadyFlow details	
Source	Cloudera managed ADLS
Source Format	PDF
Destination	Pinecone
Destination Format	Vector DB

Prerequisites

Learn how to collect the information you need to deploy the ADLS to Pinecone [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have your ADLS container, storage account, and path from which you want to ingest data.

- You have performed one of the following to configure access to the ADLS folders:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.

       a. To access IDBroker mappings in your environment, click Actions Manage Access .

       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have your OpenAI API key.
- You have your Pinecone environment and index into which you want to ingest data.
- Your Pinecone index was configured with the ’text-embedding-ada-002 model’.
- You have your Pinecone API key.

Related Concepts

List of required configuration parameters for the ADLS to Pinecone [Technical Preview] ReadyFlow

List of required configuration parameters for the ADLS to Pinecone [Technical Preview] ReadyFlow

When deploying the ADLS to Pinecone [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: ADLS to Pinecone [Technical Preview] ReadyFlow configuration parameters

Parameter name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores.
CDPEnvironment	The Cloudera Environment configuration resources.
OpenAI API Key	Specify the API key used to authenticate to OpenAI.
OpenAI Model	Specify the OpenAI model name to use for embedding the data. The default model is 'text-embedding-ada-002'.
Pinecone API Key	Specify the API key used to authenticate to Pinecone.
Pinecone Environment	Specify the name of the Pinecone environment.
Pinecone Index	Specify the name of the Pinecone index. To ensure correct functionality of this ReadyFlow, make sure your Pinecone index has been created with the same dimension as the 'text-embedding-ada-002' model (1536).
Source ADLS File System	Specify the name of the ADLS data container you want to read from. The full path will be constructed from: abfs://#{Source ADLS File System}@#{Source ADLS Storage Account}.dfs.core.windows.net/#{Source ADLS Path}
Source ADLS Path	Specify the path within the ADLS data container where you want to read from without any leading characters. The full path will be constructed from: abfs://#{Source ADLS File System}@#{Source ADLS Storage Account}.dfs.core.windows.net/#{Source ADLS Path}
Source ADLS Storage Account	Specify the source ADLS storage account name.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Airtable to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: Airtable to S3/ADLS..... 4

Prerequisites..... 4

List of required configuration parameters for the Airtable to S3/ADLS ReadyFlow..... 8

ReadyFlow overview: Airtable to S3/ADLS

You can use the Airtable to S3/ADLS ReadyFlow to consume objects from an Airtable table, filter them, and write them as JSON, CSV or Avro files to a destination in Amazon S3 or Azure Data Lake Service (ADLS).

This ReadyFlow consumes objects from an Airtable table, filtered by field name(s). You can pick whether to include metadata or to extract only the Airtable fields. The objects are then converted to the specified output data format and written to the target S3 or ADLS destination. The flow writes out a file every time its size has either reached 100 MB or five minutes have passed. Files can reach a maximum size of 1 GB. Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

Airtable to S3/ADLS ReadyFlow details	
Source	Airtable
Source Format	Airtable
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	Avro, CSV, or JSON

Prerequisites

Learn how to collect the information you need to deploy the Airtable to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have an Airtable account. If you do not have one, you can sign up for a free account here.
- You have data stored in an Airtable table that you want to move to an object store.
- You have your Airtable table ID or name.

- You have your Airtable credentials (API Key and Base ID).

  Getting your Airtable API Key

  1. Log in to Airtable.
  2. Click the user icon in the top right corner of the Airtable platform homepage.
  3. Click Account from the dropdown menu.
  4. Get your own personal API Key displayed on the Account Overview page.

     Note: If there is no personal API Key associated with your account, click Generate API key.

  Getting your Airtable Base ID

  1. Log in to Airtable and go to the Airtable Standard API page.
  2. Select which Base you want to connect to.
  3. Get the Airtable Base ID displayed on the Airtable API page for the Base you selected.

  Limitations of Airtable REST AP:
  API rate limit

  The Airtable REST API limits the number of requests that can be sent on a per-base basis to avoid bottlenecks. The limit is 5 requests per second per base. If this limit is exceeded you cannot make another request for 30 seconds. Yield Duration is set to 15 sec and Run Schedule is set to 1 min in the ReadyFlow to solve this problem.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Airtable to S3/ADLS ReadyFlow

List of required configuration parameters for the Airtable to S3/ADLS ReadyFlow

When deploying the Airtable to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Airtable to S3/ADLS ReadyFlow configuration parameters

Parameter name	Description
Airtable API Key	Specify your Airtable API key.
Airtable Base ID	Specify the ID of the Airtable base to be queried.
Airtable Field Names	Specify the field names requested from the Airtable table to be queried as a comma separated list.
Airtable Table ID or Name	Specify the ID or name of the Airtable table to be queried.
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).

Parameter name	Description
CSV Delimiter	If your desired output data is CSV, specify the delimiter here.
Data Output Format	Specify the desired format for your output data. You can use CSV, JSON or AVRO with this ReadyFlow.
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS container you want to write to. • For S3, enter a value in the form: s3a://*[***DESTINATION S3 BUCKET***]* • For ADLS, enter a value in the form: abfs://*[***DESTINATION ADLS FILE SYSTEM***]@[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net
Include Metadata	Specify whether to return Airtable fields only or include metadata fields in the result. Set to "true" to include metadata fields in the result. Airtable fields will be stored in the "fields" field. Set to "false" to include only Airtable fields in the result. The default value is "true".

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Box to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: Box to S3/ADLS.....4

Prerequisites..... 4

List of required configuration parameters for the Box to S3/ADLS ReadyFlow..... 7

ReadyFlow overview: Box to S3/ADLS

You can use the Box to S3/ADLS ReadyFlow to move data from a source Box location to a destination Amazon S3 bucket or Azure Data Lake Storage (ADLS) location.

This ReadyFlow consumes files from a Box folder and writes them to a Cloudera managed destination S3 or ADLS location. For the source, specify the Box App credentials (Account ID, config file) and the Box folder ID. You can choose whether to include subfolder files. For the destination, specify the S3 or ADLS storage location and path. The ReadyFlow polls the Box folder for new files (it performs a listing periodically). Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note:

This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up an IDBroker mapping or Ranger policies when using fine-grained object store access allowing your workload user access to the destination S3 or ADLS location.

Box to S3/ADLS ReadyFlow details	
Source	Box
Source Format	Any
Destination	Amazon S3 or ADLS
Destination Format	Same as source

Prerequisites

Learn how to collect the information you need to deploy the Box to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

A pre-configured Box App under the Box Account owning the resources being accessed (https://app.box.com/developers/console), with the following configuration:

- If you create a new App, select Server Authentication (with JWT) authentication method. If you want to use an existing App, choose one with OAuth 2.0 with JSON Web Tokens (Server Authentication) as the Authentication method.
- Make sure the App has a Client ID and a Client Secret.
- Set App Access Level to App + Enterprise Access.
- Enable Write all files and folders in Box for Application Scopes.
- Enable the Advanced Features: Generate user access tokens and Make API calls using the as-user header.
- Generate a Public/Private Keypair under Add and Manage Public Keys and download the configuration JSON file (under App Settings). Set the full path of this file in the Box Config File property.

  Note: You can only download the configuration JSON with the keypair details once, when you generate the keypair. Also this is the only time Box shows you the private key. If you want to download the configuration JSON file later (under App Settings) - or if you want to use your own keypair - after you download it, you need to edit the file and add the keypair details manually.
- After you have made all the above settings, you need to reauthorize the App on the admin page. (Reauthorize App at https://app.box.com/master/custom-apps.) If the app is configured for the first time you need to add it (Add App) before it can be authorized.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Box to S3/ADLS ReadyFlow

List of required configuration parameters for the Box to S3/ADLS ReadyFlow

When deploying the Box to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Box to S3/ADLS ReadyFlow configuration parameters

Parameter name	Description
Box Account ID	Specify your Box Account ID.
Box App Config File	Upload the Box App config file in JSON format.
Box Folder ID	Specify the ID of the Box folder you want to read from.
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".

Parameter name	Description
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS Container you want to write to. For S3, enter a value in the form: s3a://[Destination S3 Bucket] For ADLS, enter a value in the form: abfs://[Destination ADLS File System]@[Destination ADLS Storage Account].dfs.core.windows.net
Include Subfolder Files	Specify whether to include list of files from subfolders. Set to "true" to include files from subfolders. Set to "false" to only list files from the specified Box folder.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Confluent Cloud to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Confluent Cloud to S3/ADLS..... 4

Prerequisites..... 4

List of required configuration parameters for the Confluent Cloud to S3/ADLS ReadyFlow..... 7

ReadyFlow: Confluent Cloud to S3/ADLS

You can use the Confluent Cloud to S3/ADLS ReadyFlow to ingest JSON, CSV or Avro data from a source Kafka topic in Confluent Cloud to a destination S3 or ADLS location, while filtering events using a SQL query.

This ReadyFlow consumes JSON, CSV or Avro data from a source Kafka topic in Confluent Cloud and parses the schema by looking up the schema name in the Confluent Schema Registry. You can filter events by specifying a SQL query in the 'Filter Rule' parameter. The filtered records are then converted to the specified output data format. The flow then writes out a file every time its size has either reached 100MB or five minutes have passed. Files can reach a maximum size of 1GB. The filtered events are then written to the destination Amazon S3 or Azure Data Lake Service (ADLS) location. Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

ReadyFlow details	
Source	Confluent Cloud Kafka Topic
Source Format	JSON, CSV, Avro
Destination	Amazon S3 or ADLS
Destination Format	JSON, CSV, Avro

Prerequisites

Learn how to collect the information you need to deploy the Confluent Cloud to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have created a Confluent Cloud Kafka cluster.
- You have created at least one Kafka topic.
- You have created a schema for your data in the Confluent Cloud Schema Registry.
- You have the Confluent Cloud Kafka broker endpoint.
- You have the Confluent Cloud Kafka client API Key and Secret.
- You have the Kafka Consumer Group ID.
- You have the Confluent Cloud Schema Registry endpoint.
- You have the Confluent Cloud Schema Registry API Key and Secret.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Confluent Cloud to S3/ADLS ReadyFlow

List of required configuration parameters for the Confluent Cloud to S3/ADLS ReadyFlow

When deploying the Confluent Cloud to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Confluent Cloud to S3/ADLS ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
CSV Delimiter	If your source data is CSV, specify the delimiter here.

Parameter Name	Description
Data Input Format	Specify the format of your input data. You can use "CSV", "JSON" or "AVRO" with this ReadyFlow. • CSV • JSON • AVRO
Data Output Format	Specify the desired format for your output data. You can use "CSV", "JSON" or "AVRO" with this ReadyFlow.
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS Container you want to write to. For S3, enter a value in the form: s3a://[Destination S3 Bucket] For ADLS, enter a value in the form: abfs://[Destination ADLS File System]@[Destination ADLS Storage Account].dfs.core.windows.net
Filter Rule	Specify the filter rule expressed in SQL to filter streaming events for the destination database. Records matching the filter will be written to the destination database. The default value forwards all records.
Kafka Broker Endpoint	Specify the Kafka bootstrap server.
Kafka Client API Key	Specify the API Key to connect to the Kafka cluster.
Kafka Client API Secret	Specify the API Secret to connect to the Kafka cluster.
Kafka Consumer Group Id	The name of the consumer group used for the source topic you are consuming from.
Kafka Source Topic	Specify a topic name that you want to read from.
Kafka Schema Name	Specify the schema name to be looked up in the Confluent Schema Registry for the Source Kafka Topic.
Kafka Source Topic	Specify the topic name that you want to read from.
Schema Registry Client Key	Specify the API Key to connect to the Confluent Schema Registry.
Schema Registry Client Secret	Specify the API Secret to connect to the Confluent Schema Registry.
Schema Registry Endpoint	Specify the Schema Registry API endpoint.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Confluent Cloud to Snowflake

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Confluent Cloud to Snowflake.....4

Prerequisites..... 4

List of required configuration parameters for the Confluent Cloud to Snowflake ReadyFlow..... 6

ReadyFlow: Confluent Cloud to Snowflake

You can use the Confluent Cloud to Snowflake ReadyFlow to move your data from a Kafka topic hosted in a Confluent Cloud Kafka cluster to a Snowflake table.

This ReadyFlow consumes JSON, CSV or Avro data from a source Kafka topic in Confluent Cloud and parses the schema by looking up the schema name in the Confluent Schema Registry. You can filter events based on a user provided SQL query. The filtered events are then merged into CSV files, compressed into gzip format and written to the destination Snowflake DB table.

ReadyFlow details	
Source	Confluent Cloud Kafka Topic
Source Format	JSON, CSV, Avro
Destination	Snowflake DB
Destination Format	Snowflake DB

Prerequisites

Learn how to collect the information you need to deploy the Confluent Cloud to Snowflake ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have created a Confluent Cloud Kafka cluster.
- You have created at least one Kafka topic.
- You have created a schema for your data in the Confluent Cloud Schema Registry.
- You have the Confluent Cloud Kafka broker endpoint.
- You have the Confluent Cloud Kafka client API Key and Secret.
- You have the Kafka Consumer Group ID.
- You have the Confluent Cloud Schema Registry endpoint.
- You have the Confluent Cloud Schema Registry API Key and Secret.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.

- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have created a Snowflake account.
- Ensure that the Snowflake database and table you are ingesting data to exist. If not, create them.
- You have set up your destination Snowflake table with the right columns and columns type.
- Your Snowflake user account has write access to the Snowflake database table.
- You have the schema of the Snowflake database. The default value is PUBLIC.
- You have created Snowflake Private and Public Keys and assigned the Public Key to your Snowflake account user.

  For more information, see Key Pair Authentication in Snowflake documentation.
- You have created your Snowflake Pipe and configured the file format for CSV input.

  For more information, see CREATE FILE FORMAT in Snowflake documentation.

Related Concepts

List of required configuration parameters for the Confluent Cloud to Snowflake ReadyFlow

List of required configuration parameters for the Confluent Cloud to Snowflake ReadyFlow

When deploying the Confluent Cloud to Snowflake ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Confluent Cloud to Snowflake ReadyFlow configuration parameters

Parameter Name	Description
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Data Input Format	Specify the format of your input data. You can use "CSV", "JSON" or "AVRO" with this ReadyFlow. • CSV • JSON • AVRO
Filter Rule	Specify the filter rule expressed in SQL to filter streaming events for the destination database. Records matching the filter will be written to the destination database. The default value forwards all records.
Kafka Broker Endpoint	Specify the Kafka bootstrap server.
Kafka Client API Key	Specify the API Key to connect to the Kafka cluster.
Kafka Client API Secret	Specify the API Secret to connect to the Kafka cluster.
Kafka Consumer Group Id	The name of the consumer group used for the source topic you are consuming from.
Kafka Schema Name	Specify the schema name to be looked up in the Confluent Schema Registry for the Source Kafka Topic.
Kafka Source Topic	Specify a topic name that you want to read from.
Schema Registry Client Key	Specify the API Key to connect to the Confluent Schema Registry.
Schema Registry Client Secret	Specify the API Secret to connect to the Confluent Schema Registry.
Schema Registry Endpoint	Specify the Confluent Schema Registry API endpoint.
Snowflake Account Name	Specify the Snowflake Account Name.
Snowflake Organization	Specify the Snowflake organization name.
Snowflake Pipe	Specify the Snowflake pipe.
Snowflake Private Key File	Upload the Snowflake private key file.
Snowflake Private Key Password	Specify the password for the Snowflake private key.
Snowflake User Name	Specify your Snowflake user name.
Snowflake User Password	Specify the Snowflake user password.
Snowflake Warehouse	Specify your Snowflake warehouse. The default value is COMPUTE_WH
Table Name	Specify the Snowflake table name you want to write to.
Target Database Name	Specify the Snowflake database you want to write to.
Target Schema Name	Specify the schema of the Snowflake database. The default value is PUBLIC

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Db2 CDC to Iceberg

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Db2 CDC to Iceberg [Technical Preview]..... 4

Prerequisites..... 4

List of required configuration parameters for the Db2 CDC to Iceberg [Technical Preview] ReadyFlow.....6

ReadyFlow: Db2 CDC to Iceberg [Technical Preview]

You can use the Db2 CDC to Iceberg [Technical Preview] ReadyFlow to retrieve CDC events from a Db2 source table and stream them into Iceberg.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from a Db2 source table to stream the events to an Iceberg destination table. Failed Iceberg write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteIcebergTable connection to monitor failed write operations.

Note: This ReadyFlow is considered Technical Preview and is not designed for production use.

Note:

This ReadyFlow does not support schema changes or primary key field updates. The destination table must be Iceberg format v2. Make sure to assign the correct permissions for the Iceberg destination table to the specified CDP Workload User.

Db2 CDC to Iceberg [Technical Preview] ReadyFlow details	
Source	Db2 Table
Source Format	Db2 Table
Destination	Iceberg
Destination Format	Parquet

Prerequisites

Learn how to collect the information you need to deploy the Db2 CDC to Iceberg [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the Db2 database server hostname and port.
- You have obtained the Db2 database name, schema name, and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the Db2 table.
- You have obtained the Db2 database driver jar file.
- You have performed the Db2 setup tasks required to run Debezium.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- In Cloudera Data Warehouse, you have activated the same environment for which Cloudera DataFlow has been enabled. This will create a default database catalog. For more information, see Activating an AWS environment from Cloudera Data Warehouse or Activating Azure environments.
- You have created a Hive Virtual Warehouse referencing the default database catalog. For more information, see Creating your first Virtual Warehouse.
- You have created the Iceberg table that you want to ingest data into, running in your Hive Virtual Warehouse. For more information, see Iceberg table creation from Hive.

Related Concepts

List of required configuration parameters for the Db2 CDC to Iceberg [Technical Preview] ReadyFlow

List of required configuration parameters for the Db2 CDC to Iceberg [Technical Preview] ReadyFlow

When deploying the Db2 CDC to Iceberg [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Db2 CDC to Iceberg [Technical Preview] ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the CDP machine user or workload user name that you want to use to authenticate to Iceberg. Ensure this user has the appropriate access rights to the Iceberg table.
CDP Workload User Password	Specify the password of the CDP machine user or workload user you are using to authenticate to Iceberg.
CDPEnvironment	The CDP Environment configuration resources.
Hive Catalog Namespace	Specify the Hive catalog namespace.
Iceberg Table Name	Specify the Iceberg table name you want to write to. Table must be Iceberg format v2.
Source Database Driver	Upload the database driver jar file for your source database.
Source Database Name	Specify the source database name.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.
Source Database Server Port	Specify the source database server port. The default value is 50000.
Source Database Table Name	Specify the source database table name in the form: [schema_name].[table_name].
Source Database User	Specify the source database user.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Db2 CDC to Kudu

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Db2 CDC to Kudu **4**

Prerequisites **4**

List of required configuration parameters for the Db2 CDC to Kudu ReadyFlow **6**

ReadyFlow: Db2 CDC to Kudu

You can use the Db2 CDC to Kudu ReadyFlow to retrieve CDC events from a Db2 source table and stream them to a Kudu destination table.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from a Db2 source table to stream the events to a Kudu destination table. Failed Kudu write operations are retried automatically to handle transient issues. Define a KPI on the failure_ModifyKuduTable connection to monitor failed write operations.

Note:

This ReadyFlow does not support schema changes or primary key field updates. Make sure to assign the correct permissions for the Kudu destination table to the specified Cloudera Workload User.

Note: This ReadyFlow assumes that

- the DB2 table has column names in UPPERCASE (the default)
- the Kudu table is created through Impala, so the column names are lowercase (the default)

As a result, the PutKudu processor sets the property Lowercase Field Names to "true". Otherwise it would fail because of the different case.

Db2 CDC to Kudu ReadyFlow details	
Source	Db2 Table
Source Format	Db2 Table
Destination	Kudu
Destination Format	Kudu Table

Prerequisites

Learn how to collect the information you need to deploy the Db2 CDC to Kudu ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the Db2 database server hostname and port.
- You have obtained the Db2 database name, schema name, and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the Db2 table.
- You have obtained the Db2 database driver jar file.
- You have performed the Db2 setup tasks required to run Debezium.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.

- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have a Real-Time Data Mart cluster running Kudu, Impala, and Hue in the same environment for which Cloudera DataFlow has been enabled.
- You have the Kudu Master hostnames.

  1. From Management Console, click Data Hub Clusters.
  2. Select the Real-Time Data Mart cluster to which you want to ingest data into.
  3. Click the Hardware tab.
  4. Copy the FQDN for each Kudu Master.

- You have the Kudu database name.
  1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
  2. Click the Tables icon on the left pane.
  3. Select the default database.
- You have created the Kudu table that you want to ingest data into. Ensure that the field case sensitivity matches that of the source table.
  1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
  2. Click the Tables icon on the left pane.
  3. Select the default database, and click + New to create a new table.
  4. In the Type field, select Manually and click Next.
  5. Provide the table Name, Format, Primary keys, and any partitions.
  6. Click Submit. The newly created table displays in the default database Tables pane.
  7. Check the Kudu UI **Tables** tab for the name of the table you created. You will need this table name when you use the Cloudera DataFlow Deployment wizard to deploy the ReadyFlow.
- You have assigned permissions via IDBroker or in Ranger to enable the Cloudera Workload User to access the Kudu table that you want to ingest data into.
  1. From the base cluster on Cloudera Public Cloud, select Ranger.
  2. Select your Real Time Data Mart cluster from the **Kudu** folder.
  3. Click Add New Policy policy.
  4. On the **Create Policy** page, enter the Kudu table name in the topic field.
  5. Add the Cloudera Workload User in the Select User field.
  6. Add the Insert and Select permissions in the Permissions field.
  7. Click Save.

Related Concepts

List of required configuration parameters for the Db2 CDC to Kudu ReadyFlow

List of required configuration parameters for the Db2 CDC to Kudu ReadyFlow

When deploying the Db2 CDC to Kudu ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Db2 CDC to Kudu ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the CDP machine user or workload user name that you want to use to authenticate to Kudu. Ensure this user has the appropriate access rights to the Kudu table.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate to Kudu.
Destination Database Table Name	Specify the destination database table name in the form: [database_name].[table_name].
Kudu Master Hosts	Specify the Kudu Master hostnames in a comma separated list.
Source Database Driver	Upload the database driver jar file for your source database.
Source Database Name	Specify the source database name.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.

Parameter Name	Description
Source Database Server Port	Specify the source database server port. The default value is 50000.
Source Database Table Name	Specify the source database table name in the form: [schema_name].[table_name].
Source Database User	Specify the source database user.

Related Concepts

List of required configuration parameters for the Db2 CDC to Kudu ReadyFlow

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Dropbox to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: Dropbox to S3/ADLS..... 4

Prerequisites..... 4

List of required configuration parameters for the Dropbox to S3/ADLS ReadyFlow..... 8

ReadyFlow overview: Dropbox to S3/ADLS

You can use the Dropbox to S3/ADLS ReadyFlow to ingest data from Dropbox and write it to a destination in Amazon S3 or Azure Data Lake Service (ADLS).

This ReadyFlow consumes files from a Dropbox folder and writes them to a Cloudera managed destination S3 or ADLS location. For the source, specify the Dropbox App credentials (app key, app secret, access token, refresh token) and the ID or path of the Dropbox folder. You can choose whether to include subfolder files. For the destination, specify the S3 or ADLS storage location and path.

The ReadyFlow polls the Dropbox folder for new files (it performs a listing periodically). Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

Dropbox to S3/ADLS ReadyFlow details	
Source	Dropbox
Source Format	Any
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	Same as source

Prerequisites

Learn how to collect the information you need to deploy the Dropbox to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have a Dropbox account. If you do not have one, you can sign up for a free account here.
- You have data stored in Dropbox that you want to move to an object store.
- You have created a Dropbox App.

- You have generated the following Dropbox credentials: App Key, App Secret, Access Token, and Refresh Token

  Generating App Key and App Secret

  1. Log in with your Dropbox account.
  2. If you already have an app, go to Dropbox Developers.
  3. Click App Console and select your app.

     On the app's info page you can find the App key and App secret credentials.
  4. If you do not have any apps, go to Dropbox Developers and click Create apps.
  5. On the next page select Scoped access and Full Dropbox as access type.
  6. Provide a name for your app.
  7. Accept the Dropbox API Terms and Conditions and click Create app.

  Your app is created and you are taken to the app's info page. You can find the App key and App secret credentials on the Settings tab.

  Setting required permissions for your app

  1. Go to the Permissions tab of your app’s info page.
  2. Set the required permission for your app by enabling files.content.read.

     With this permission, your app will be able to read the files in Dropbox.
  3. Click Submit.

     Note: In case you already have an Access Token and Refresh Token, those tokens have to be regenerated after the permission change.

     For more information, see *Generating Access Token and Refresh Token*.

  Generating Access Token and Refresh Token

  1. Go to the following web page: https://www.dropbox.com/oauth2/authorize?token_access_type=offline&response_type=code&client_id=*YOUR_APP_KEY*
  2. Click Next and then click Allow on the next page.

     An access code is generated for you, and it is displayed on the next page.
  3. Execute the following command from the terminal to fetch the access and refresh tokens.curl https://api.dropbox.com/oauth2/token -d code=your_generated_access_code -d grant_type=authorization_code -u your_app_key:your_app_secret

     Important: Make sure you execute the curl command right after the access code generation, since the code expires very quickly. In case the curl command returns invalid grant error, generate a new access code.

     The curl command results in a json file that contains the access_token and the refresh_token.

     ```
     {
     "access_token": "sl.xxxxxxxxxxx"
     "expires_in": 14400,
     "refresh_token": "xxxxxx",
     "scope": "files.content.read files.metadata.read",
     "uid": "xxxxxx",
     "account_id": "dbid:xxxx"
     }
     ```

     For more information, see the Dropbox Getting Started.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Dropbox to S3/ADLS ReadyFlow

List of required configuration parameters for the Dropbox to S3/ADLS ReadyFlow

When deploying the Dropbox to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Dropbox to S3/ADLS ReadyFlow configuration parameters

Parameter name	Description
CDP Workload User	Specify the CDP machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the CDP machine user or workload user you are using to authenticate against the object stores (via IDBroker).
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".

Parameter name	Description
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS container you want to write to. • For S3, enter a value in the form: s3a:*//[***DESTINATION S3 BUCKET***]* • For ADLS, enter a value in the form: abfs:*//[***DESTINATION ADLS FILE SYSTEM***]@[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net
Dropbox App Access Token	Specify your Dropbox App Access Token.
Dropbox App Key	Specify your Dropbox App Key.
Dropbox App Refresh Token	Specify your Dropbox App Refresh Token.
Dropbox App Secret	Specify your Dropbox App Secret.
Dropbox Folder ID or Path	Specify the identifier or path of the Dropbox folder you want to read from. ‘Folder' should match the following regular expression pattern: /.|id:. Example folder identifier: id: odTlUvbpIEAAAAAAAAAGGQ Example folder path: /Folder1/Folder2
Include Subfolder Files	Specify whether to include a list of files from subfolders. Set to "true" to include files from subfolders. Set to "false" to only list files from the specified Dropbox folder. The default value is "true".

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Azure Event Hub to ADLS

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow Overview: Azure Event Hub to ADLS.....4

Prerequisites..... 4

List of required configuration parameters for the Azure Event Hub to ADLS ReadyFlow..... 8

ReadyFlow Overview: Azure Event Hub to ADLS

You can use the Azure Event Hub to ADLS ReadyFlow to move JSON, CSV or Avro files from an Azure Event Hub namespace, optionally parsing the schema using Cloudera Schema Registry or direct schema input. The flow then filters records based on a user-provided SQL query and writes them to a target Azure Data Lake Storage (ADLS) location in the specified output data format.

This ReadyFlow consumes JSON, CSV or Avro data from a source Azure Event Hub Namespace and merges the events into JSON, CSV or Avro files before writing the data to ADLS. The flow writes out a file every time its size has either reached 100MB or five minutes have passed. Files can reach a maximum size of 1GB. You can specify the Event Hub Namespace you want to read from as well as the target ADLS data container and path. Failed ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

Azure Event Hub to ADLS ReadyFlow details	
Source	Azure Event Hub Namespace
Source Format	JSON, CSV, Avro
Destination	ADLS
Destination Format	JSON, CSV, Avro

Moving data to object stores

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost-effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Prerequisites

Learn how to collect the information you need to deploy the Azure Event Hub to ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have obtained the Event Hub Namespace and Instance name
  - Navigate to Event Hubs and select your Event Hub instance (msignite, in this example):

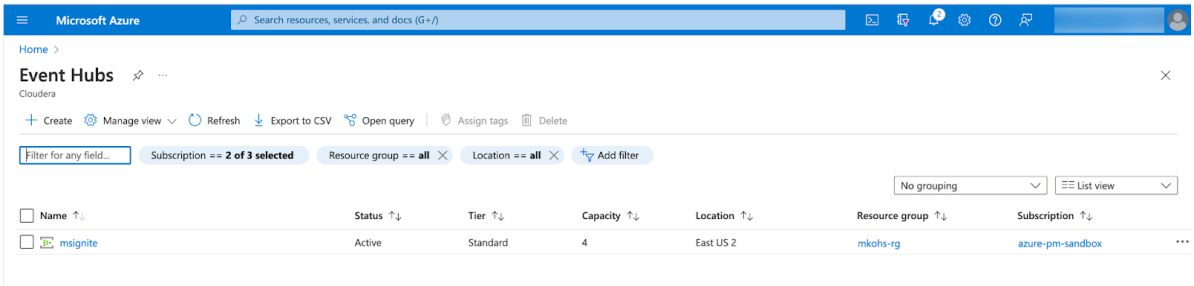

  - Copy the namespace and the Event Hub Instance name (msignite/msignite in this example).

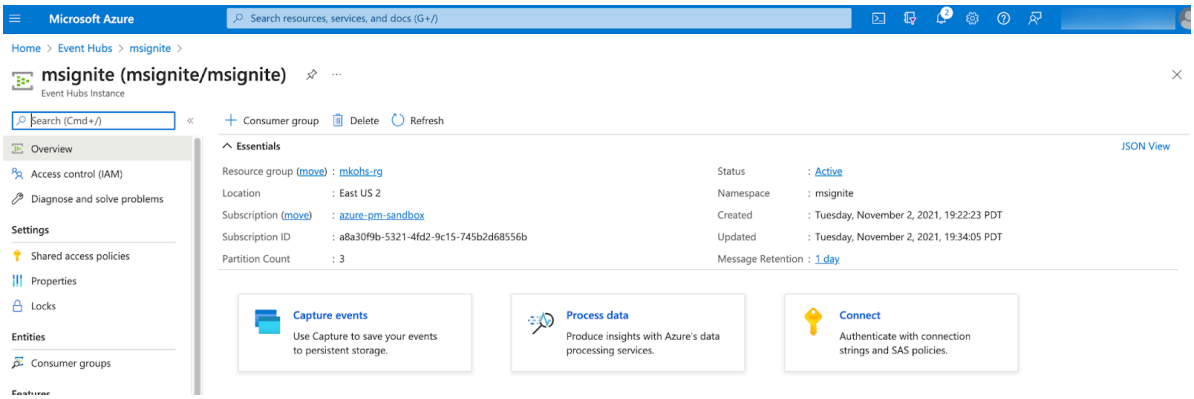

- You have created an Event Hub Shared Access Policy and obtained its name and Primary Access Key
  - Click Shared access policies.
  - Set up a shared access policy that allows “Listen”. Copy the Policy name.

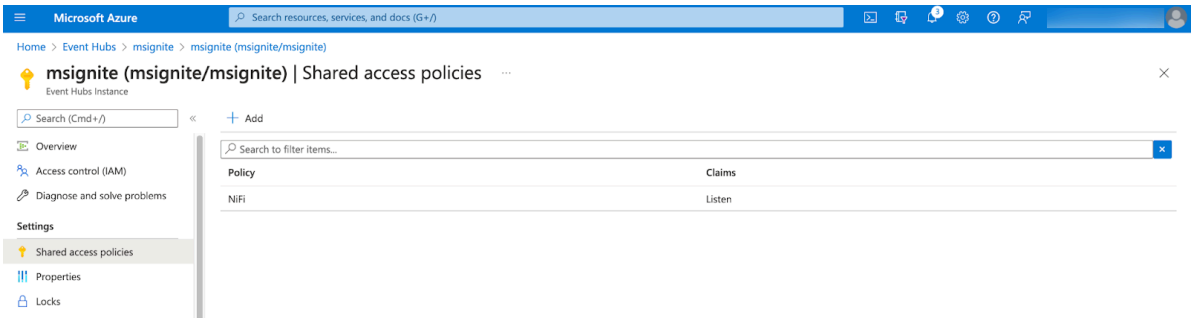

  - Select the policy and copy the primary key.

- You have created a consumer group.
  - Click Consumer groups.
  - Use the $Default consumer group or create your own group for the ReadyFlow:

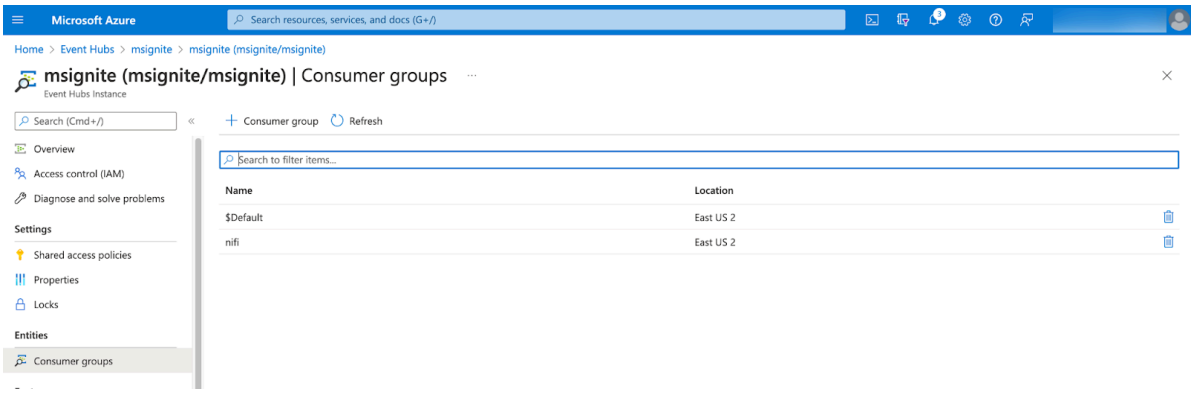

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Azure Event Hub to ADLS ReadyFlow

List of required configuration parameters for the Azure Event Hub to ADLS ReadyFlow

When deploying the Azure Event Hub to ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Azure Event Hub to ADLS ReadyFlow configuration parameters	
Parameter Name	**Description**
ADLS File System	Specify the name of the ADLS data container you want to write to. The full path will be constructed from: abfs://{ADLS File System}@#{ADLS Storage Account}.dfs.core.windows.net/#{ADLS Path}/${Kafka.topic}
ADLS Path	Specify the path within the ADLS data container where you want to write to without any leading characters. The full path will be constructed from: abfs://{ADLS File System}@#{ADLS Storage Account}.dfs.core.windows.net/#{ADLS Path}/${Kafka.topic}

Azure Event Hub to ADLS ReadyFlow configuration parameters	
Parameter Name	**Description**
ADLS Storage Account	Specify the storage account name you want to write to. The full ADLS data container path will be constructed from: abfs://{ADLS File System}@#{ADLS Storage Account}.dfs.core.windows.net/#{ADLS Path}/${Kafka.topic}
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to Kafka and the object store. Ensure this user has the appropriate access rights in Ranger for the Kafka topic and Ranger or IDBroker for object store access.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against Kafka and the object store.
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Data Output Format	Specify the desired format for your output data. You can select from • CSV • JSON • AVRO with this ReadyFlow.
Event Hub Access Policy Name	Specify the Access Policy Name that this flow should use. The full path for the event hub endpoint will be constructed from sb://#{Event Hub Namespace}.#{Event Hub Service Bus Endpoint}/;SharedAccessKeyName=#{Event Hub Access Policy Name};SharedAccessKey=#{Event Hub Access Primary Key}
Event Hub Access Primary Key	Specify the Primary Key that allows clients to use the Access Policy that you provided earlier. The full path for the event hub endpoint will be constructed from sb://#{Event Hub Namespace}.#{Event Hub Service Bus Endpoint}/;SharedAccessKeyName=#{Event Hub Access Policy Name};SharedAccessKey=#{Event Hub Access Primary Key}
Event Hub Consumer Group	Specify the Event Hub Consumer Group you want to use with this flow. Any consumer group other than $Default needs to be created in Event Hub first.
Event Hub Instance Name	Specify the Event Hub Instance Name inside the Event Hub Namespace you want to use.
Event Hub Namespace	Specify the Event Hub Namespace which contains the Event Hub instance you want to use. The full path for the event hub endpoint will be constructed from sb://#{Event Hub Namespace}.#{Event Hub Service Bus Endpoint}/;SharedAccessKeyName=#{Event Hub Access Policy Name};SharedAccessKey=#{Event Hub Access Primary Key}
Event Hub Partitions Count	Specify the number of partitions that the Event Hub has. Only this number of partitions will be used, so it is important to ensure that if the number of partitions changes that this value be updated. Otherwise, some messages may not be consumed.
Event Hub Service Bus Endpoint	Specify the Event Hub Service Bus Endpoint. The default value is .servicebus.windows.net The full path for the event hub endpoint will be constructed from sb://#{Event Hub Namespace}.#{Event Hub Service Bus Endpoint}/;SharedAccessKeyName=#{Event Hub Access Policy Name};SharedAccessKey=#{Event Hub Access Primary Key}
Filter Rule	Specify the filter rule expressed in SQL to filter streaming events for the destination object store. Records matching the filter will be written to the destination object store. The default value forwards all records.
Schema Text	Specify the Avro-formatted schema to be used for the source event hub data.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Google Drive to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: Google Drive to S3/ADLS **4**

Prerequisites **4**

List of required configuration parameters for the Google Drive to S3/ADLS ReadyFlow **7**

ReadyFlow overview: Google Drive to S3/ADLS

You can use the Google Drive to S3/ADLS ReadyFlow to ingest data from a Google Drive location to a destination in Amazon S3 or Azure Data Lake Service (ADLS).

This ReadyFlow consumes files from Google Drive and Cloudera managed S3 or ADLS location. For the source, specify the Google service account key file in JSON and the Google Drive folder ID. You can choose whether to include objects in subfolders or to only include objects from the specified Google Drive folder. For the destination, specify the S3 or ADLS storage location and path.

The ReadyFlow polls the folder on Google Drive for new files (it performs a listing periodically). Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Google Drive to S3/ADLS ReadyFlow details	
Source	Google Drive
Source Format	Any
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	Same as source

Prerequisites

Learn how to collect the information you need to deploy the Google Drive to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have a Google Cloud account.If you do not have one, you can sign up for a free account here.
- You have enabled Google Drive API in Google Cloud. For instructions, see Enable Google Workspace APIs.
- You have granted access to the Google Drive folder.

  1. In the Google Cloud Console, navigate to IAM & Admin Service Accounts .
  2. Copy the email address of the service account you are going to use.
  3. Navigate to the folder to be listed in Google Drive.
  4. Right-click the folder and select Share.
  5. Enter the service account email.
- You have your Google Drive folder ID.

  Navigate to the folder to be listed in Google Drive and enter it. The URL in your browser includes the ID at the end of the URL. For example, if the URL is https://drive.google.com/drive/folders/1trTraPVCnX5_TNwO8d9P_bz278xWOmGm, the folder ID is 1trTraPVCnX5_TNwO8d9P_bz278xWOmGm.
- You have data stored in the Google Drive folder that you want to move to an object store.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.

       a. To access IDBroker mappings in your environment, click Actions Manage Access .

       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Google Drive to S3/ADLS ReadyFlow

List of required configuration parameters for the Google Drive to S3/ADLS ReadyFlow

When deploying the Google Drive to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Google Drive to S3/ADLS ReadyFlow configuration parameters

Parameter name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".

Parameter name	Description
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS container you want to write to. • For S3, enter a value in the form: s3a://*[***DESTINATION S3 BUCKET***]* • For ADLS, enter a value in the form: abfs://*[***DESTINATION ADLS FILE SYSTEM***]@[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net
Google Drive Folder ID	Specify the ID of the Google Drive folder you want to read from.
Google Service Account Key File	Upload the Google Service Account key file in JSON format.
Include Subfolder Files	Specify whether to include a list of files from subfolders. Set to "true" to include files from subfolders Set to "false" to only list files from the specified Google Drive folder. The default value is *TRUE*.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Hello World

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Hello World **4**

Prerequisites **4**

List of required configuration parameters for the Hello World ReadyFlow **5**

ReadyFlow: Hello World

You can use the Hello World ReadyFlow to move change data from Wikipedia API to local disk. This ReadyFlow gets you started with your first flow deployment without any external dependencies.

This ReadyFlow can be deployed without any external dependencies and does not require any parameter values. It retrieves the latest changes from Wikipedia through invoking the Wikipedia API. It converts JSON events to Avro, before filtering and routing them to two different processors which merge events together before a file is written to local disk.

ReadyFlow details	
Source	Wikipedia
Source Format	JSON
Destination	Local disk
Destination Format	Avro

Prerequisites

Learn how to collect the information you need to deploy the Hello World ReadyFlow, and meet other prerequisites.

For your data ingest source

None

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

None

Related Concepts

List of required configuration parameters for the Hello World ReadyFlow

List of required configuration parameters for the Hello World ReadyFlow

When deploying the Hello World ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

When deploying this ReadyFlow, you do not need to provide any parameter values.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

HubSpot to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: HubSpot to S3/ADLS..... 4

Prerequisites..... 4

List of required configuration parameters for the HubSpot to S3/ADLS ReadyFlow..... 7

ReadyFlow overview: HubSpot to S3/ADLS

You can use the HubSpot to S3/ADLS ReadyFlow to consume objects from a HubSpot database table, filter them, and write the data as JSON, CSV or Avro files to a destination in Amazon S3 or Azure Data Lake Service (ADLS).

This ReadyFlow retrieves objects from a Private HubSpot App, filtered by field name(s). You can pick whether to include metadata or to extract only the HubSpot fields. The objects are then converted to the specified output data format and written to the target S3 or ADLS destination. The flow writes out a file every time its size has either reached 100MB or five minutes have passed. Files can reach a maximum size of 1GB. Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

HubSpot to S3/ADLS ReadyFlow details	
Source	HubSpot
Source Format	HubSpot
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	CSV, JSON or Avro

Moving data to object stores

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost-effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Prerequisites

Learn how to collect the information you need to deploy the HubSpot to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have a HubSpot account.
- You have created a HubSpot private app.
- You have the access token from the private app.
- You have data stored in HubSpot that you want to move to an object store.
- You have the HubSpot object type.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the HubSpot to S3/ADLS ReadyFlow

List of required configuration parameters for the HubSpot to S3/ADLS ReadyFlow

When deploying the HubSpot to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: HubSpot to S3/ADLS ReadyFlow configuration parameters

Parameter name	Description
CDP Workload User	Specify the CDP machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the CDP machine user or workload user you are using to authenticate against the object stores (via IDBroker).
CSV Delimiter	If your desired output data is CSV, specify the delimiter here.
Data Output Format	Specify the desired format for your output data. You can use CSV, JSON or AVRO with this ReadyFlow.
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".

Parameter name	Description
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS container you want to write to. For S3, enter a value in the form: s3a://[Destination S3 Bucket] For ADLS, enter a value in the form: abfs://[Destination ADLS File System]@[Destination ADLS Storage Account].dfs.core.windows.net
HubSpot Access Token	Specify the HubSpot access token to authenticate requests.
HubSpot Object Type	Specify the HubSpot object type to request. An Object Type is prefixed with /crm/v3/objects/ where the following suffixes are supported: • companies • contacts • deals • line_items • products • tickets • quotes • calls • emails • meetings • notes • tasks
HubSpot Result Limit	Specify the maximum number of results returned by the HubSpot processor (The maximum you can specify is 100). The default value is 100.
Include Metadata	Specify whether to return HubSpot fields only or include metadata fields in the result. Set to true to include metadata fields in the result. HubSpot fields will be stored in the properties field. Set to false to include only HubSpot fields in the result. The default value is true.
Incremental Delay	Specify a value for this HubSpot processor property to avoid missing objects when the clocks of your local machines and of HubSpot servers are not in sync. This property is only used if Incremental Loading is set to true. The default value is 30 seconds.
Incremental Loading	Specify a value for this HubSpot processor property to decide whether to load queried objects incrementally. Set to true to query each object exactly once. Set to false to query each object at least once. The default value is true.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

HuggingFace to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: HuggingFace to S3/ADLS..... 4

Prerequisites..... 4

List of required configuration parameters for the HuggingFace to S3/ADLS ReadyFlow..... 7

ReadyFlow overview: HuggingFace to S3/ADLS

You can use the HuggingFace to S3/ADLS ReadyFlow to retrieve a HuggingFace dataset and write the Parquet data to a target S3 or ADLS destination.

This ReadyFlow retrieves a dataset from the HuggingFace API and writes the Parquet data to a target S3 or ADLS destination. The dataset retrieved by default is "wikitext" (the default value for the Dataset Name parameter). Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

This flow is not meant to run continuously and should be run once per dataset retrieved.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

HuggingFace to S3/ADLS ReadyFlow details	
Source	HuggingFace Dataset
Source Format	Parquet
Destination	Cloudera Managed Amazon S3 or ADLS
Destination Format	Parquet

Prerequisites

Learn how to collect the information you need to deploy the HuggingFace to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

You have the name of the HuggingFace dataset to be retrieved. The dataset retrieved by default is "wikitext" (https://huggingface.co/datasets/wikitext).

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the HuggingFace to S3/ADLS ReadyFlow

List of required configuration parameters for the HuggingFace to S3/ADLS ReadyFlow

When deploying the HuggingFace to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: HuggingFace to S3/ADLS ReadyFlow configuration parameters

Parameter name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores.
CDPEnvironment	The Cloudera Environment configuration resources.
Dataset Name	Specify the Dataset name. The default is "wikitext".
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/" and that it does not end with "/".

Parameter name	Description
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS Container you want to write to. For S3, enter a value in the form: s3a://[Destination S3 Bucket] For ADLS, enter a value in the form: abfs://[Destination ADLS File System]@[Destination ADLS Storage Account].dfs.core.windows.net

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

JDBC to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: JDBC to JDBC.....4

Prerequisites..... 4

List of required configuration parameters for the JDBC to JDBC ReadyFlow..... 5

ReadyFlow overview: JDBC to JDBC

You can use the JDBC to JDBC ReadyFlow to move data between a source database table and destination database table.

This ReadyFlow consumes data from a source database table. You can filter records by specifying a SQL query in the Filter Rule parameter. The filtered records are then inserted into the target database table. Failed database operations are retried automatically to handle transient issues. Define a KPI on the failure_PutDatabaseRecord connection to monitor failed insert operations. Define a KPI on the retry_PutDatabaseRecord connection to monitor retryable insert operations.

JDBC to JDBC ReadyFlow details	
Source	Database Table
Source Format	Database Table
Destination	Database Table
Destination Format	Database Table

Prerequisites

Learn how to collect the information you need to deploy the JDBC to JDBC ReadyFlow, and meet other prerequisites.

For your data ingest source and target

- You have obtained the source and target table names
- You have identified the source table column which should be used to incrementally fetch records
- You have obtained the JDBC source and target database connection URLs
- You have obtained the appropriate JDBC driver and JDBC driver class names for your source and target database types
- You have obtained the usernames and passwords to access the source and target tables

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.

- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

1. Give a user permission to add the ReadyFlow to the Catalog.
   a. From the Management Console, click User Management.
   b. Enter the name of the user or group you wish to authorize in the Search field.
   c. Select the user or group from the list that displays.
   d. Click Roles Update Roles .
   e. From Update Roles, select DFCatalogAdmin and click Update.

   Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
2. Give your user or group permission to deploy flow definitions.
   a. From the Management Console, click Environments to display the Environment List page.
   b. Select the environment to which you want your user or group to deploy flow definitions.
   c. Click Actions Manage Access to display the Environment Access page.
   d. Enter the name of your user or group you wish to authorize in the Search field.
   e. Select your user or group and click Update Roles.
   f. Select DFFlowAdmin from the list of roles.
   g. Click Update Roles.
3. Give your user or group access to the Project where the ReadyFlow will be deployed.
   a. Go to DataFlow Projects .
   b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
4. Start typing the name of the user or group you want to add and select them from the list.
5. Select the Resource Roles you want to grant.
6. Click Update Roles.
7. Click Synchronize Users.

Related Concepts

List of required configuration parameters for the JDBC to JDBC ReadyFlow

List of required configuration parameters for the JDBC to JDBC ReadyFlow

When deploying the JDBC to JDBC ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: JDBC to JDBC ReadyFlow configuration parameters

Parameter name	Description
Destination Database Connection URL	Specify the JDBC database connection URL used to connect to your destination database. May contain database system name, host, port, database name and some parameters. The exact syntax of a database connection URL is specified by your DBMS, for example, jdbc:mysql://hostname:port/database for MySQL databases.
Destination Database Driver	Upload the database driver jar file for your destination database.

Parameter name	Description
Destination Database Driver Class Name	Specify the destination database driver class name. For example, com.mysql.jdbc.Driver for MySQL databases.
Destination Database Password	Specify the destination database password.
Destination Database Table Name	Specify the destination database table name.
Destination Database User	Specify the destination database user.
Filter Rule	Specify the filter rule expressed in SQL to filter database records. Records matching the filter are written to the destination. The default value forwards all records.
Source Database Connection URL	Specify the JDBC database connection URL used to connect to your source database. May contain database system name, host, port, database name and some parameters. The exact syntax of a database connection URL is specified by your DBMS, for example, jdbc:mysql://hostname:port/database for MySQL databases.
Source Database Driver	Upload the database driver jar file for your source database.
Source Database Driver Class Name	Specify the source database driver class name. For example, com.mysql.jdbc.Driver for MySQL databases.
Source Database Incremental Fetch Column	Specify the column that should be used to track state as new data lands in the source table.
Source Database Password	Specify the source database password.
Source Database Table Name	Specify the source database table name.
Source Database User	Specify the source database user.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

JDBC to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: JDBC to S3/ADLS..... 4

Prerequisites..... 4

List of required configuration parameters for the JDBC to S3/ADLS ReadyFlow..... 7

ReadyFlow overview: JDBC to S3/ADLS

You can use the JDBC to S3/ADLS ReadyFlow to move data from a database table to an Amazon S3 bucket or Azure Data Lake Storage location while converting the files to the specified format.

This ReadyFlow consumes data from a source database table. You can filter records by specifying a SQL query in the Filter Rule parameter. The filtered records are then converted to the specified output data format and written to S3 or ADLS. Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

JDBC to S3/ADLS ReadyFlow details	
Source	Database Table
Source Format	Database Table
Destination	Amazon S3 or ADLS
Destination Format	JSON, CSV, Avro

Prerequisites

Learn how to collect the information you need to deploy the JDBC to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have obtained the source table name
- You have identified the source table column which should be used to incrementally fetch records
- You have obtained the JDBC database connection URL
- You have obtained the appropriate JDBC driver and JDBC driver class name for your source database type
- You have obtained a username and password to access the source table

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.

- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the JDBC to S3/ADLS ReadyFlow

List of required configuration parameters for the JDBC to S3/ADLS ReadyFlow

When deploying the JDBC to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: JDBC to S3/ADLS ReadyFlow configuration parameters

Parameter name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
CSV Delimiter	If your output data is CSV, specify the delimiter here.
Data Output Format	Specify the desired format for your output data. You can use "CSV", "JSON" or "AVRO" with this ReadyFlow.

Parameter name	Description
Database Connection URL	Specify the JDBC database connection URL used to connect to your source database. May contain database system name, host, port, database name and some parameters. The exact syntax of a database connection URL is specified by your DBMS. For example, jdbc:mysql://hostname:port/database for MySQL databases.
Database Driver	Upload the database driver jar file for your source database.
Database Driver Class Name	Specify the source database driver class name. For example, com.mysql.jdbc.Driver for MySQL databases.
Database Incremental Fetch Column	Specify the column that is to be used to track state as new data lands in the source table.
Database Password	Specify the source database password.
Database Table Name	Specify the source database table name.
Database User	Specify the source database user.
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS Container you want to write to. For S3, enter a value in the form: s3a://[Destination S3 Bucket] For ADLS, enter a value in the form: abfs://[Destination ADLS File System]@[Destination ADLS Storage Account].dfs.core.windows.net
Filter Rule	Specify the filter rule expressed in SQL to filter database records. Records matching the filter are written to the destination. The default value forwards all records.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Kafka to ADLS Avro

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow Overview: Kafka to ADLS Avro..... 4

Prerequisites..... 4

List of required configuration parameters for the Kafka to ADLS Avro ReadyFlow..... 8

ReadyFlow Overview: Kafka to ADLS Avro

You can use the Kafka to ADLS Avro ReadyFlow to move your data from a Kafka topic to an ADLS directory.

This ReadyFlow consumes JSON, CSV, or Avro data from a source Kafka topic and merges the events into Avro files before writing the data to Azure Data Lake Storage (ADLS). The flow writes out a file every time its size has either reached 100 MB or five minutes have passed. Files can reach a maximum size of 1 GB. You can specify the topic you want to read from as well as the target ADLS data container and path. Failed ADLS write operations are retried automatically to handle transient issues. Define a KPI on the 'failure_WriteToADLS' connection to monitor failed write operations.

Kafka to ADLS Avro ReadyFlow details	
Source	Kafka topic
Source Format	JSON, CSV, Avro
Destination	ADLS
Destination Format	Avro

Moving data to object stores

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost-effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Prerequisites

Learn how to collect the information you need to deploy the Kafka to ADLS Avro ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created at least one Kafka topic.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:

     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.

- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
         "name":"sensor_1",
         "doc":"Reading #1.",
         "type":"int"
      },
      {
         "name":"sensor_2",
         "doc":"Reading #2.",
         "type":"int"
      },
      {
         "name":"sensor_3",
         "doc":"Reading #3.",
         "type":"int"
      }
   ]
}
```

- You have the Schema Registry Host Name.

  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.

- You have the Kafka broker end points.
  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.
- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.

- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Kafka to ADLS Avro ReadyFlow

List of required configuration parameters for the Kafka to ADLS Avro ReadyFlow

When deploying the Kafka to ADLS Avro ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Kafka to ADLS Avro ReadyFlow configuration parameters

Parameter Name	Description
ADLS File System	Specify the name of the ADLS data container you want to write to. The full path will be constructed from: abfs://{ADLS File System}@#{ADLS Storage Account}.dfs.core.windows.net/#{ADLS Path}/${Kafka.topic}
ADLS Path	Specify the path within the ADLS data container where you want to write to without any leading characters. The full path will be constructed from: abfs://{ADLS File System}@#{ADLS Storage Account}.dfs.core.windows.net/#{ADLS Path}/${Kafka.topic}

Parameter Name	Description
ADLS Storage Account	Specify the storage account name you want to write to. The full ADLS data container path will be constructed from: abfs://{ADLS File System}@#{ADLS Storage Account}.dfs.core.windows.net/#{ADLS Path}/${Kafka.topic}
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to Kafka and the object store. Ensure this user has the appropriate access rights in Ranger for the Kafka topic and Ranger or IDBroker for object store access.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against Kafka and the object store.
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Data Input Format	Specify the format of your input data. • CSV • JSON • AVRO
Kafka Broker Endpoint	Specify the Kafka bootstrap servers string as a comma separated list.
Kafka Consumer Group ID	The name of the consumer group used for the source topic you are consuming from.
Kafka Source Topic	Specify a topic name that you want to read from.
Schema Name	Specify the schema name to be looked up in the Schema Registry for the source Kafka topic.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Kafka to Apache Iceberg

Date published: 2021-04-06
Date modified: 2023-05-02

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2023. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow Overview: Kafka to Iceberg.....4

Prerequisites..... 4

List of required configuration parameters for the Kafka to Iceberg ReadyFlow..... 7

ReadyFlow Overview: Kafka to Iceberg

You can use the Kafka to Apache Iceberg ReadyFlow to move your data from a CDP-managed Kafka topic to a CDP-managed Apache Iceberg table in Cloudera Data Warehouse.

This ReadyFlow consumes JSON, CSV or Avro data from a Kafka topic, parses the schema by looking up the schema name in the CDP Schema Registry and ingests it into an Iceberg table. The flow writes out a file every time its size has either reached 100MB or five minutes have passed. Files can reach a maximum size of 1GB. You can specify the topic you want to read from as well as the target Iceberg table. Failed ingestion operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToIceberg connection to monitor failed write operations.

Kafka to Iceberg ReadyFlow details	
Source	Kafka topic
Source Format	JSON, CSV, Avro
Destination	Iceberg
Destination Format	Parquet

Prerequisites

Learn how to collect the information you need to deploy the Kafka to Iceberg ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have created a Streams Messaging cluster in CDP Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created at least one Kafka topic.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:
     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
```

```
    "name":"SensorReading",
    "namespace":"com.cloudera.example",
    "doc":"This is a sample sensor reading",
    "fields":[
       {
          "name":"sensor_id",
          "doc":"Sensor identification number.",
          "type":"int"
       },
       {
          "name":"sensor_ts",
          "doc":"Timestamp of the collected readings.",
          "type":"long"
       },
       {
          "name":"sensor_0",
          "doc":"Reading #0.",
          "type":"int"
       },
       {
          "name":"sensor_1",
          "doc":"Reading #1.",
          "type":"int"
       },
       {
          "name":"sensor_2",
          "doc":"Reading #2.",
          "type":"int"
       },
       {
          "name":"sensor_3",
          "doc":"Reading #3.",
          "type":"int"
       }
    ]
}
```

- You have the Schema Registry Host Name.

  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have the Kafka broker end points.

  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in DataFlow.

- You have assigned the CDP Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the CDP Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For DataFlow

- You have enabled DataFlow for an environment.

  For information on how to enable DataFlow for an environment, see Enabling DataFlow for an Environment.
- You have created a Machine User to use as the CDP Workload User.
- You have given the CDP Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The CDP Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the CDP Public Cloud environment that you enabled for DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your CDP user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.

For your data ingest target

- In Cloudera Data Warehouse, you have activated the same environment for which DataFlow has been enabled. This will create a default database catalog.
- You have created a Hive Virtual Warehouse referencing the default database catalog.
- You have created the Iceberg table that you want to ingest data into, running in your Hive Virtual Warehouse.

Related Concepts

List of required configuration parameters for the Kafka to Iceberg ReadyFlow

List of required configuration parameters for the Kafka to Iceberg ReadyFlow

When deploying the Kafka to Iceberg ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Kafka to Iceberg ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the CDP machine user or workload username that you want to use to authenticate to Kafka and the object store. Ensure this user has the appropriate access rights in Ranger for the Kafka topic and Ranger or IDBroker for Hive access.
CDP Workload User Password	Specify the CDP machine user or workload username that you want to use to authenticate to Kafka and Hive. Ensure this user has the appropriate access rights in Ranger for the Kafka topic and ID Broker for Hive access.
CDPEnvironment	The CDP Environment configuration resources.
CSV Delimiter	If your source data is CSV, specify the delimiter here.

Parameter Name	Description
Data Input Format	Specify the format of your input data. Possible values are: • CSV • JSON • AVRO
Hive Catalog Namespace	Specify the Hive Catalog Namespace. The default value is default, which references the database catalog created automatically when you activate an environment in Cloudera Data Warehouse.
Iceberg Table Name	Specify the Iceberg table that you want to write to, running in your Hive Virtual Warehouse.
Kafka Broker Endpoint	Specify the Kafka bootstrap servers string as a comma separated list.
Kafka Consumer Group ID	The name of the consumer group used for the source topic you are consuming from.
Kafka Source Topic	Specify a topic name that you want to read from.
Schema Name	Specify the schema name to be looked up in the Schema Registry for the source Kafka topic.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Kafka Filter to Kafka

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Kafka filter to Kafka **4**

Prerequisites **4**

List of required configuration parameters for the Kafka Filter to Kafka ReadyFlow **8**

ReadyFlow: Kafka filter to Kafka

You can use the Kafka filter to Kafka ReadyFlow to move your data between two Kafka topics, while applying a schema to the data in Cloudera DataFlow.

Kafka filter to Kafka ReadyFlow description

This ReadyFlow consumes JSON, CSV, or Avro data from a source Kafka topic and parses the schema by looking up the schema name in the Cloudera Schema Registry. You can filter events by specifying a SQL query in the `Filter Rule` parameter. The filtered events are then converted to the specified output data format and written to the destination Kafka topic. Failed Kafka write operations are retried automatically to handle transient issues. Define a KPI on the `failure_WriteToKafka` connection to monitor failed write operations.

Kafka filter to Kafka ReadyFlow details

ReadyFlow details	
Source	Kafka Topic
Source Format	JSON, CSV, or Avro
Destination	Kafka Topic
Destination Format	JSON, CSV, or Avro

Moving data with a Kafka filter to Kafka flow

You can use a Kafka filter to Kafka data flow when you want to filter specific events from a Kafka topic and write the filtered stream to another Kafka topic. For example, you could use this flow to filter out erroneous records that contain "null" for an important field. In order to filter the data using a SQL query, you need to provide a schema for the events that you are processing in the topic.

Your data flow can consume JSON, CSV, or Avro data from the source Kafka topic and write to the destination Kafka topic in any of these formats. The data flow parses the schema by looking up the schema name in the Cloudera Schema Registry. You can filter the events by specifying a SQL query. The filtered events are converted to the specified output data format, and they are written to the destination Kafka topic.

Prerequisites

Learn how to collect the information you need to deploy the Kafka filter to Kafka ReadyFlow, and meet other prerequisites.

For your data source and target

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.

- You have created at least two Kafka topics for your ingest data flow, according to your source and target requirements.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:

     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.
  7. Repeat the previous steps to add another topic.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
         "name":"sensor_1",
         "doc":"Reading #1.",
         "type":"int"
      },
      {
         "name":"sensor_2",
         "doc":"Reading #2.",
         "type":"int"
      },
      {
         "name":"sensor_3",
         "doc":"Reading #3.",
         "type":"int"
      }
```

```
    ]
}
```

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have the Kafka broker end points.
  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.
- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

Related Concepts

List of required configuration parameters for the Kafka Filter to Kafka ReadyFlow

Related Information

Deploying a ReadyFlow

List of required configuration parameters for the Kafka Filter to Kafka ReadyFlow

When deploying the Kafka Filter to Kafka ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Kafka Filter to Kafka ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kafka. Ensure this user has the appropriate access rights in Ranger for the source and target Kafka topics.
CDP Workload User Password	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kafka.
Data Input Format	Specify the format of your input data. If your data input is CSV, define a CSV delimiter for the data in the CSV Delimiter text box. If you use AVRO or JSON format, the delimiter is ignored. • CSV • JSON • AVRO
Data Output Format	Specify the format of your output data. If your data input is CSV, define a CSV delimiter for the data in the CSV Delimiter text box. If you use AVRO or JSON format, the delimiter is ignored. • CSV • JSON • AVRO
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Filter Rule	If you want to filter your data for the destination topic, enter a filter rule expressed in SQL. Records matching the filter are written to the destination topic in Kafka. If you do not provide a specific filter rule, the default rule forwards all records. Default rule:SELECT * FROM FLOWFILE
Kafka Broker Endpoint	Specify the Kafka bootstrap servers string as a comma separated list in the format <host>:<port>.
Kafka Consumer Group ID	Add the name of the consumer group used for the source topic you are consuming from. Make sure to use the Consumer Group ID that the selected Cloudera Workload User is allowed to use.
Kafka Source Topic	Specify a topic name that you want to read from.
Kafka Destination Topic	Specify a topic name that you want to write to.
Kafka Producer ID	Use it to identify your data flow in SMM.
Schema Name	Specify the schema that you want to use in your data flow. Cloudera DataFlow looks up this schema in the Schema Registry you define with the Schema Registry Hostname. See the *Appendix* for an example schema.
Schema Registry Hostname	Specify the hostname of the Schema Registry running on the master node in the Streams Messaging cluster that you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.

Related Concepts

Prerequisites

Cloudera DataFlow

Kafka to Apache Iceberg

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow Overview: Kafka to Iceberg.....4

Prerequisites..... 4

List of required configuration parameters for the Kafka to Iceberg ReadyFlow..... 7

ReadyFlow Overview: Kafka to Iceberg

You can use the Kafka to Apache Iceberg ReadyFlow to move your data from a Cloudera-managed Kafka topic to a Cloudera-managed Apache Iceberg table in Cloudera Data Warehouse.

This ReadyFlow consumes JSON, CSV or Avro data from a Kafka topic, parses the schema by looking up the schema name in the Cloudera Schema Registry and ingests it into an Iceberg table. The flow writes out a file every time its size has either reached 100MB or five minutes have passed. Files can reach a maximum size of 1GB. You can specify the topic you want to read from as well as the target Iceberg table. Failed ingestion operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToIceberg connection to monitor failed write operations.

Kafka to Iceberg ReadyFlow details	
Source	Kafka topic
Source Format	JSON, CSV, Avro
Destination	Iceberg
Destination Format	Parquet

Prerequisites

Learn how to collect the information you need to deploy the Kafka to Iceberg ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created at least one Kafka topic.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:

     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
```

```
    "type":"record",
    "name":"SensorReading",
    "namespace":"com.cloudera.example",
    "doc":"This is a sample sensor reading",
    "fields":[
       {
          "name":"sensor_id",
          "doc":"Sensor identification number.",
          "type":"int"
       },
       {
          "name":"sensor_ts",
          "doc":"Timestamp of the collected readings.",
          "type":"long"
       },
       {
          "name":"sensor_0",
          "doc":"Reading #0.",
          "type":"int"
       },
       {
          "name":"sensor_1",
          "doc":"Reading #1.",
          "type":"int"
       },
       {
          "name":"sensor_2",
          "doc":"Reading #2.",
          "type":"int"
       },
       {
          "name":"sensor_3",
          "doc":"Reading #3.",
          "type":"int"
       }
    ]
}
```

- You have the Schema Registry Host Name.

  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have the Kafka broker end points.

  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.

- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- In Cloudera Data Warehouse, you have activated the same environment for which Cloudera DataFlow has been enabled. This will create a default database catalog. For more information, see Activating an AWS environment from Cloudera Data Warehouse or Activating Azure environments.
- You have created a Hive Virtual Warehouse referencing the default database catalog. For more information, see Creating your first Virtual Warehouse.
- You have created the Iceberg table that you want to ingest data into, running in your Hive Virtual Warehouse. For more information, see Iceberg table creation from Hive.

Related Concepts

List of required configuration parameters for the Kafka to Iceberg ReadyFlow

List of required configuration parameters for the Kafka to Iceberg ReadyFlow

When deploying the Kafka to Iceberg ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Kafka to Iceberg ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to Kafka and the object store. Ensure this user has the appropriate access rights in Ranger for the Kafka topic and Ranger or IDBroker for Hive access.
CDP Workload User Password	Specify the Cloudera machine user or workload username that you want to use to authenticate to Kafka and Hive. Ensure this user has the appropriate access rights in Ranger for the Kafka topic and ID Broker for Hive access.
CDPEnvironment	The CDP Environment configuration resources.
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Data Input Format	Specify the format of your input data. Possible values are: • CSV • JSON • AVRO
Hive Catalog Namespace	Specify the Hive Catalog Namespace. The default value is default, which references the database catalog created automatically when you activate an environment in Cloudera Data Warehouse.
Iceberg Table Name	Specify the Iceberg table that you want to write to, running in your Hive Virtual Warehouse.
Kafka Broker Endpoint	Specify the Kafka bootstrap servers string as a comma separated list.
Kafka Consumer Group ID	The name of the consumer group used for the source topic you are consuming from.
Kafka Source Topic	Specify a topic name that you want to read from.
Schema Name	Specify the schema name to be looked up in the Schema Registry for the source Kafka topic.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Kafka to Kafka

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Kafka to Kafka **4**

Prerequisites **4**

List of required configuration parameters for the Kafka to Kafka ReadyFlow **7**

ReadyFlow: Kafka to Kafka

You can use the Kafka to Kafka ReadyFlow to move your data from a source Kafka topic to the specified destination Kafka topic.

This ReadyFlow consumes data from a source Kafka topic and writes it to the specified destination topic. The flow does not apply any schema to the data. The data is simply passed through to the specified destination topic. Failed Kafka write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToKafka connection to monitor failed write operations.

Kafka to Kafka ReadyFlow details	
Source	Kafka topic
Source Format	Any
Destination	Kafka topic
Destination Format	Any

Prerequisites

Learn how to collect the information you need to deploy the Kafka to S3 Avro ReadyFlow, and meet other prerequisites.

For your data source and target

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created at least two Kafka topics for your ingest data flow, according to your source and target requirements.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:
     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.
  7. Repeat the previous steps to add another topic.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
```

```
    "type":"record",
    "name":"SensorReading",
    "namespace":"com.cloudera.example",
    "doc":"This is a sample sensor reading",
    "fields":[
       {
          "name":"sensor_id",
          "doc":"Sensor identification number.",
          "type":"int"
       },
       {
          "name":"sensor_ts",
          "doc":"Timestamp of the collected readings.",
          "type":"long"
       },
       {
          "name":"sensor_0",
          "doc":"Reading #0.",
          "type":"int"
       },
       {
          "name":"sensor_1",
          "doc":"Reading #1.",
          "type":"int"
       },
       {
          "name":"sensor_2",
          "doc":"Reading #2.",
          "type":"int"
       },
       {
          "name":"sensor_3",
          "doc":"Reading #3.",
          "type":"int"
       }
    ]
}
```

- You have the Schema Registry Host Name.

  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have the Kafka broker end points.

  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.

- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

Related Concepts

List of required configuration parameters for the Kafka to Kafka ReadyFlow

List of required configuration parameters for the Kafka to Kafka ReadyFlow

When deploying the Kafka to Kafka ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Kafka to Kafka ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kafka. Ensure this user has the appropriate access rights to the source and destination Kafka topics.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against Kafka.
Kafka Broker Endpoint	Specify the Kafka bootstrap servers string as a comma separated list.
Kafka Consumer Group ID	Specify the ID for the consumer group used for the source topic you are consuming from.
Kafka Destination Topic	Specify the destination topic name that you want to write to.

Parameter Name	Description
Kafka Producer ID	Specify the id for the Kafka producer writing into the destination topic.
Kafka Source Topic	Specify a topic name that you want to read from.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Kafka to Kudu

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Kafka to Kudu **4**

Prerequisites **4**

List of required configuration parameters for the Kafka to Kudu ReadyFlow **8**

ReadyFlow: Kafka to Kudu

This use case shows you how you can move your data from a Kafka topic into Apache Kudu in your Cloudera Public Cloud Real-time Data Mart cluster. You can learn how to create such a data flow easily using the Kafka to Kudu ReadyFlow.

This ReadyFlow consumes JSON, CSV or Avro data from a source Kafka topic, parses the schema by looking up the schema name in the Cloudera Schema Registry and ingests it into a Kudu table. You can pick the Kudu operation (INSERT, INSERT_IGNORE, UPSERT, UPDATE, DELETE, UPDATE_IGNORE, DELETE_IGNORE) that fits best for your use case. Failed Kudu write operations are retried automatically to handle transient issues. Define a KPI on the `failure_WriteToKudu` connection to monitor failed write operations.

Kafka to Kudu ReadyFlow details	
Source	Kafka topic
Source Format	JSON, CSV, Avro
Destination	Kudu
Destination Format	Kudu Table

Time series use cases analyse data obtained during specified intervals, and enable you to improve performance based on available data. Examples include:

- Optimizing yield or yield quality in a manufacturing plant
- Dynamically optimizing network capacity during peak load of better telecommunications uptime and services

These use cases require that you store events at a high frequency, while providing ad-hoc query and record update abilities.

Prerequisites

Learn how to collect the information you need to deploy the Kafka to Kudu ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created at least one Kafka topic.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:

     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.

- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
         "name":"sensor_1",
         "doc":"Reading #1.",
         "type":"int"
      },
      {
         "name":"sensor_2",
         "doc":"Reading #2.",
         "type":"int"
      },
      {
         "name":"sensor_3",
         "doc":"Reading #3.",
         "type":"int"
      }
   ]
}
```

- You have the Schema Registry Host Name.

  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.

- You have the Kafka broker end points.
  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.
- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.

- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have a Real-Time Data Mart cluster running Kudu, Impala, and Hue in the same environment for which Cloudera DataFlow has been enabled.
- You have the Kudu Master hostnames.

  1. From Management Console, click Data Hub Clusters.
  2. Select the Real-Time Data Mart cluster to which you want to ingest data into.
  3. Click the Hardware tab.
  4. Copy the FQDN for each Kudu Master.
- You have created the Kudu table that you want to ingest data into.

  1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
  2. Click the Tables icon on the left pane.
  3. Select the default database, and click + New to create a new table.
  4. In the Type field, select Manually and click Next.
  5. Provide the table Name, Format, Primary keys, and any partitions.
  6. Click Submit. The newly created table displays in the default database Tables pane.
  7. Check the Kudu UI **Tables** tab for the name of the table you created. You will need this table name when you use the Cloudera DataFlow Deployment wizard to deploy the ReadyFlow.

- You have assigned permissions via IDBroker or in Ranger to enable the Cloudera Workload User to access the Kudu table that you want to ingest data into.

  1. From the base cluster on Cloudera Public Cloud, select Ranger.
  2. Select your Real Time Data Mart cluster from the **Kudu** folder.
  3. Click Add New Policy policy.
  4. On the **Create Policy** page, enter the Kudu table name in the topic field.
  5. Add the Cloudera Workload User in the Select User field.
  6. Add the Insert and Select permissions in the Permissions field.
  7. Click Save.

Related Concepts

List of required configuration parameters for the Kafka to Kudu ReadyFlow

List of required configuration parameters for the Kafka to Kudu ReadyFlow

When deploying the Kafka to Kudu ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Kafka to Kudu ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kafka and Kudu. Ensure this user has the appropriate access rights to the Kafka topics and Kudu table.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against Kafka and Kudu.
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Data Input Format	Specify the format of your input data. You can use "CSV", "JSON" or "AVRO" with this ReadyFlow.
Kafka Broker Endpoint	Specify the Kafka bootstrap servers string as a comma separated list.
Kafka Consumer Group ID	Specify the ID for the consumer group used for the source topic you are consuming from.
Kafka Source Topic	Specify a topic name that you want to read from.
Kudu Master Hosts	Specify the Kudu Master hostnames in a comma separated list.
Kudu Operation Type	Specify the operation that you want to use when writing data to Kudu. Valid values are: • INSERT • INSERT_IGNORE • UPSERT • UPDATE • DELETE
Kudu Table Name	Specify the Kudu table name you want to write to.
Schema Name	Specify the schema name to be looked up in the Schema Registry.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Kafka to Cloudera Operational Database

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Kafka to Cloudera Operational Database..... 4

Prerequisites..... 4

List of required configuration parameters for the Kafka to Cloudera Operational Database ReadyFlow..... 9

ReadyFlow: Kafka to Cloudera Operational Database

You can use an Apache NiFi data flow to ingest data into Cloudera Operational Database through Cloudera DataFlow. Learn how to use NiFi to move data from a range of locations to Cloudera Operational Database in Cloudera Public Cloud.

This ReadyFlow consumes JSON, CSV or Avro data from a source Kafka topic, parses the schema by looking up the schema name in the Cloudera Schema Registry and ingests it into an HBase table in Cloudera Operational Database. Failed HBase write operations are retried automatically to handle transient issues. Define a KPI on the `failure_WriteToCOD` connection to monitor failed write operations.

ReadyFlow details	
Source	Kafka topic
Source Format	JSON, CSV, Avro
Destination	Cloudera Operational Database
Destination Format	HBase Table

Today's scalable web applications for use cases like hotel or flight bookings as well as mobile banking applications are relying on an equally scalable database which can serve data at a very low latency. Cloudera Operational Database in Cloudera DataFlow is powered by Apache HBase and provides application developers with everything they need to build scalable applications on top of it.

You can use Apache NiFi data flows into Apache HBase in a Cloudera Operational Database cluster to make sure that the applications you build on top always have access to the latest data.

Prerequisites

Learn how to collect the information you need to deploy the Kafka to Cloudera Operational Database (COD) ReadyFlow, and meet other prerequisites.

Use the following checklist to ensure that you meet all the requirements before you start building your data flow:

For your data ingest source

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.

- You have created at least one Kafka topic.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:

     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
         "name":"sensor_1",
         "doc":"Reading #1.",
         "type":"int"
      },
      {
         "name":"sensor_2",
         "doc":"Reading #2.",
         "type":"int"
      },
      {
         "name":"sensor_3",
         "doc":"Reading #3.",
         "type":"int"
      }
   ]
```

```
}
```

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have the Kafka broker end points.
  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.
- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- Ensure that the HBase table you are ingesting data to exists. If not, create one.

  1. From Cloudera Shared Data Experience UI, click Operational Database from the left navigation pane.
  2. Click Create Database.
  3. Select the environment for which Cloudera DataFlow is enabled.
  4. Enter a name for your database, and click Create Database.
  5. Go to the newly created database from the Databases page.
  6. Go to Hue UI by clicking Hue SQL Editor.
  7. Click the HBase icon to go to HBase home.
  8. Click New Table.

     The Create New Table dialog appears.
  9. Enter table name and column family name, and click Submit.

     A blank table is created.
  10. Go to the newly created table and click New Row.

      The Insert New Row dialog appears.
  11. Click Add Field, and then specify row key, column name, and cell value.

      Note: The cloumn name should follow the format: family: column_name, where, family is the column family name.
  12. Click Submit.
- Obtain the table name, column family name, and row identifier of the HBase table in Cloudera Operational Database.

  1. From Cloudera Shared Data Experience UI, click Operational Database from the left navigation pane.
  2. Select the database where your HBase table exists.
  3. Go to Hue UI by clicking Hue SQL Editor.
  4. Click the HBase icon to go to HBase home.
  5. Click the HBase table in Cloudera Operational Database.
  6. After the table appears, obtain the table name, column family name, and row identifier.
- You have set Ranger policies for HBase table.

  1. From the Cloudera Management Console, click Environments.
  2. Use the search field to find and select the Cloudera environment for which Cloudera DataFlow is enabled.
  3. Go to the Ranger UI by clicking Ranger.
  4. Select your database from the HBase folder.
  5. Click Add New Policy.
  6. Enter policy name, HBase table name, HBase column-family name, and HBase column value.
  7. In Allow Conditions section, enter the name of the Machine User, you created in Cloudera, prefixed with srv_.
  8. Click Add Permissions, and assign Read and Write permissions to the user.
  9. Click Add.

- Obtain the hbase-site.xml file.

  To get the hbase-site.xml file from Cloudera Data Hub:

  1. From the Cloudera Management Console, click Environments.
  2. Use the search field to find and select the Cloudera Public Cloud environment for which Cloudera DataFlow is enabled.
  3. Go to Data Hubs.
  4. Select the Cloudera Operational Database cluster.
  5. Go to Cloudera Manager by clicking CM-UI.
  6. Click Clusters from the left-navigation pane, and click hbase.
  7. Click Actions Download Client Configuration to download the client configuration zip file.
  8. Unzip the zip file to obtain the hbase-site.xml file.

  To get the hbase-site.xml file from Cloudera Shared Data Experience:

  1. From Cloudera Shared Data Experience UI, click Operational Database from the left navigation pane.
  2. Select the database where your HBase table exists.
  3. Go to HBase Client Tarball tab.
  4. Copy the HBase Client Configuration URL.
  5. Use the URL as a command to download the client configuration zip file.
  6. Unzip the zip file to obtain the hbase-site.xml file.

Related Concepts

List of required configuration parameters for the Kafka to Cloudera Operational Database ReadyFlow

List of required configuration parameters for the Kafka to Cloudera Operational Database ReadyFlow

When deploying the Kafka to Cloudera Operational Database ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Kafka to Cloudera Operational Database ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kafka. Ensure this user has the appropriate access rights in Ranger for the source and target Kafka topics.
CDP Workload User Password	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kafka.
CDPEnvironment	Use this parameter to upload the hbase-site.xml file of your target Hbase cluster. Cloudera DataFlow will also use this parameter to auto-populate the Flow Deployment with additional Hadoop configuration files required to interact with HBase.
COD Column Family Name	Specify the column family to use when inserting data into Cloudera Operational Database.
COD Row Identifier Field Name	Specify the name of a record field whose name should be used as the row ID for the given record.
COD Table Name	Specify the target table name in Cloudera Operational Database.
CSV Delimiter	If your source data is CSV, specify the delimiter here,
Data Input Format	Specify the desired format for your output data. You can use "CSV", "JSON" or "AVRO" with this ReadyFlow.

Parameter Name	Description
Kafka Broker Endpoint	Specify the Kafka bottstrap servers sting as a comma separated list.
Kafka Consumer Group ID	Specify the id for the consumer group used for the source topic you are consuming from.
Kafka Source Topic	Specify a topic name that you want to read from.
Schema Name	Specify the schema name to be looked up in the Schema Registry.
Schema Registry Hostname	Specify the host name of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox endpoint.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Kafka to S3 Avro

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Kafka to S3 Avro **4**

Prerequisites **4**

List of required configuration parameters for the Kafka to S3 Avro ReadyFlow **8**

ReadyFlow: Kafka to S3 Avro

You can use the Kafka to S3 Avro ReadyFlow to move your data from a Kafka topic to an Amazon S3 bucket.

This ReadyFlow consumes JSON, CSV or Avro data from a source Kafka topic and merges the events into Avro files before writing the data to S3. The flow writes out a file every time its size has either reached 100MB or five minutes have passed. Files can reach a maximum size of 1GB. You can specify the topic you want to read from as well as the target S3 bucket and path. Failed S3 write operations are retried automatically to handle transient issues. Define a KPI on the `failure_WriteToS3` connection to monitor failed write operations.

ReadyFlow details	
Source	Kafka topic
Source Format	JSON, CSV, Avro
Destination	Amazon S3
Destination Format	Avro

Moving data to object stores

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost-effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Prerequisites

Learn how to collect the information you need to deploy the Kafka to S3 Avro ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created at least one Kafka topic.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:

     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.

- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
         "name":"sensor_1",
         "doc":"Reading #1.",
         "type":"int"
      },
      {
         "name":"sensor_2",
         "doc":"Reading #2.",
         "type":"int"
      },
      {
         "name":"sensor_3",
         "doc":"Reading #3.",
         "type":"int"
      }
   ]
}
```

- You have the Schema Registry Host Name.

  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.

- You have the Kafka broker end points.
  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.
- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.

- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

List of required configuration parameters for the Kafka to S3 Avro ReadyFlow

When deploying the Kafka to S3 Avro ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Kafka to S3 Avro ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kafka and the object store. Ensure this user has the appropriate access rights in Ranger for the Kafka topic and ID Broker for object store access.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you're using to authenticate against Kafka and the object store.
CDPEnvironment	Cloudera DataFlow will use this parameter to auto-populate the Flow Deployment with Hadoop configuration files required to interact with S3. Cloudera DataFlow automatically adds all required configuration files to interact with Data Lake services. Unnecessary files that are added won't impact the deployment process.
CSV Delimiter	If your source data is CSV, specify the delimiter here.

Parameter Name	Description
Data Input Format	Specify the format of your input data. Supported values are: • CSV • JSON • AVRO
Kafka Broker Endpoint	Specify the Kafka bootstrap servers string as a comma separated list.
Kafka Consumer Group ID	The name of the consumer group used for the the source topic you are consuming from.
Kafka Source Topic	Specify a topic name that you want to read from.
S3 Bucket	Specify the name of the S3 bucket you want to write to. The full path will be constructed from: s3a://#{S3 Bucket}/#{S3 Path}/${Kafka.topic}
S3 Bucket Region	Specify the AWS region in which your bucket was created. Supported values are: • us-gov-west-1 • us-gov-east-1 • us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-north-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-1 • ap-southeast-2 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • sa-east-1 • cn-north-1 • cn-northwest-1 • ca-central-1 • me-south-1 • af-south-1 • us-iso-east-1 • us-isob-east-1 • us-iso-west-1
S3 Path	Specify the path within the bucket where you want to write to without any leading characters. The full path will be constructed from: s3a://#{S3 Bucket}/#{S3 Path}/${Kafka.topic}
Schema Name	Specify the schema name to be looked up in the Schema Registry for the source Kafka topic.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.

Cloudera DataFlow

Kafka to Snowflake

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow Overview: Kafka to Snowflake..... 4

Prerequisites..... 4

List of required configuration parameters for the Kafka to Snowflake ReadyFlow..... 8

ReadyFlow Overview: Kafka to Snowflake

ou can use the Kafka to Snowflake ReadyFlow to move your data from a Kafka topic to a Snowflake table.

This ReadyFlow consumes JSON, CSV or Avro data from a source Kafka topic and parses the schema by looking up the schema name in the CDP Schema Registry. You can filter events based on a user provided SQL query. The filtered events are then merged into CSV files, compressed into gzip format and written to the destination Snowflake DB table.

Kafka to Snowflake ReadyFlow details	
Source	Kafka Topic
Source Format	JSON, CSV, Avro
Destination	Snowflake DB
Destination Format	Snowflake DB

Prerequisites

Learn how to collect the information you need to deploy the Kafka to Snowflake ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created at least one Kafka topic.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:

     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
```

```
    "doc":"This is a sample sensor reading",
    "fields":[
       {
          "name":"sensor_id",
          "doc":"Sensor identification number.",
          "type":"int"
       },
       {
          "name":"sensor_ts",
          "doc":"Timestamp of the collected readings.",
          "type":"long"
       },
       {
          "name":"sensor_0",
          "doc":"Reading #0.",
          "type":"int"
       },
       {
          "name":"sensor_1",
          "doc":"Reading #1.",
          "type":"int"
       },
       {
          "name":"sensor_2",
          "doc":"Reading #2.",
          "type":"int"
       },
       {
          "name":"sensor_3",
          "doc":"Reading #3.",
          "type":"int"
       }
    ]
}
```

- You have the Schema Registry Host Name.

  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have the Kafka broker end points.

  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.

- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have created a Snowflake account.
- Ensure that the Snowflake database and table you are ingesting data to exist. If not, create them.
- You have set up your destination Snowflake table with the right columns and columns type.
- Your Snowflake user account has write access to the Snowflake database table.
- You have the schema of the Snowflake database. The default value is PUBLIC.
- You have created Snowflake Private and Public Keys and assigned the Public Key to your Snowflake account user.

  For more information, see Key Pair Authentication in Snowflake documentation.
- You have created your Snowflake Pipe and configured the file format for CSV input.

  For more information, see CREATE FILE FORMAT in Snowflake documentation.

Related Concepts

List of required configuration parameters for the Kafka to Snowflake ReadyFlow

List of required configuration parameters for the Kafka to Snowflake ReadyFlow

When deploying the Kafka to Snowflake ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Kafka to Snowflake ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the CDP machine user or workload user name that you want to use to authenticate to Kafka. Ensure this user has the appropriate access rights to the source Kafka topic.
CDP Workload User Password	Specify the password of the CDP machine user or workload user you are using to authenticate against Kafka.
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Data Input Format	Specify the format of your input data. You can use "CSV", "JSON" or "AVRO" with this ReadyFlow. • CSV • JSON • AVRO
Filter Rule	Specify the filter rule expressed in SQL to filter streaming events for the destination database. Records matching the filter will be written to the destination database. The default value forwards all records.
Kafka Broker Endpoint	Specify the Kafka bootstrap server.
Kafka Consumer Group ID	The name of the consumer group used for the source topic you are consuming from.
Kafka Source Topic	Specify a topic name that you want to read from.
Kafka Schema Name	Specify the schema name to be looked up in the CDP Schema Registry for the source Kafka topic.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.
Snowflake Account Name	Specify the Snowflake account name.
Snowflake Organization	Specify the Snowflake organization name.
Snowflake Pipe	Specify the Snowflake pipe.
Snowflake Private Key File	Upload the Snowflake private key file.
Snowflake Private Key Password	Specify the password for the Snowflake private key.
Snowflake User Name	Specify your Snowflake user name.
Snowflake User Password	Specify your Snowflake user password.
Snowflake Warehouse	Specify your Snowflake warehouse. The default value is COMPUTE_WH.
Table Name	Specify the Snowflake table name you want to write to.
Target Database Name	Specify the Snowflake database you want to write to.
Target Schema Name	Specify the schema of the Snowflake database. The default value is PUBLIC.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Kafka to Snowflake

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow Overview: Kafka to Snowflake Streaming..... 4

Prerequisites..... 4

List of required configuration parameters for the Kafka to Snowflake Streaming ReadyFlow..... 8

ReadyFlow Overview: Kafka to Snowflake Streaming

ou can use the Kafka to Snowflake Streaming ReadyFlow to move your data from a Kafka topic to a Snowflake table.

This ReadyFlow consumes JSON, CSV or Avro data from a source Kafka topic and parses the schema by looking up the schema name in the Cloudera Schema Registry. You can filter events based on a user provided SQL query. The filtered events are then merged into CSV files, compressed into gzip format and written to the destination Snowflake DB table.

Kafka to Snowflake Streaming ReadyFlow details	
Source	Kafka Topic
Source Format	JSON, CSV, Avro
Destination	Snowflake DB
Destination Format	Snowflake DB

Prerequisites

Learn how to collect the information you need to deploy the Kafka to Snowflake Streaming ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created at least one Kafka topic.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:

     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
```

```
    "doc":"This is a sample sensor reading",
    "fields":[
       {
          "name":"sensor_id",
          "doc":"Sensor identification number.",
          "type":"int"
       },
       {
          "name":"sensor_ts",
          "doc":"Timestamp of the collected readings.",
          "type":"long"
       },
       {
          "name":"sensor_0",
          "doc":"Reading #0.",
          "type":"int"
       },
       {
          "name":"sensor_1",
          "doc":"Reading #1.",
          "type":"int"
       },
       {
          "name":"sensor_2",
          "doc":"Reading #2.",
          "type":"int"
       },
       {
          "name":"sensor_3",
          "doc":"Reading #3.",
          "type":"int"
       }
    ]
}
```

- You have the Schema Registry Host Name.

  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have the Kafka broker end points.

  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.

- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have created a Snowflake account.
- Ensure that the Snowflake database and table you are ingesting data to exist. If not, create them.
- You have set up your destination Snowflake table with the right columns and columns type.
- Your Snowflake user account has write access to the Snowflake database table.
- You have the schema of the Snowflake database. The default value is PUBLIC.
- You have created Snowflake Private and Public Keys and assigned the Public Key to your Snowflake account user.

  For more information, see Key Pair Authentication in Snowflake documentation.
- You have created your Snowflake Pipe and configured the file format for CSV input.

  For more information, see CREATE FILE FORMAT in Snowflake documentation.

Related Concepts

List of required configuration parameters for the Kafka to Snowflake Streaming ReadyFlow

List of required configuration parameters for the Kafka to Snowflake Streaming ReadyFlow

When deploying the Kafka to Snowflake Streaming ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Kafka to Snowflake Streaming ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kafka. Ensure this user has the appropriate access rights to the source Kafka topic.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against Kafka.
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Data Input Format	Specify the format of your input data. You can use "CSV", "JSON" or "AVRO" with this ReadyFlow. • CSV • JSON • AVRO
Filter Rule	Specify the filter rule expressed in SQL to filter streaming events for the destination database. Records matching the filter will be written to the destination database. The default value forwards all records.
Kafka Broker Endpoint	Specify the Kafka bootstrap server.
Kafka Consumer Group ID	The name of the consumer group used for the source topic you are consuming from.
Kafka Source Topic	Specify a topic name that you want to read from.
Kafka Schema Name	Specify the schema name to be looked up in the Cloudera Schema Registry for the source Kafka topic.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.
Snowflake Account Name	Specify the Snowflake account name.
Snowflake Organization	Specify the Snowflake organization name.
Snowflake Pipe	Specify the Snowflake pipe.
Snowflake Private Key File	Upload the Snowflake private key file.
Snowflake Private Key Password	Specify the password for the Snowflake private key.
Snowflake User Name	Specify your Snowflake user name.
Snowflake User Password	Specify your Snowflake user password.
Snowflake Warehouse	Specify your Snowflake warehouse. The default value is COMPUTE_WH.
Table Name	Specify the Snowflake table name you want to write to.
Target Database Name	Specify the Snowflake database you want to write to.
Target Schema Name	Specify the schema of the Snowflake database. The default value is PUBLIC.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

ListenHTTP filter to Kafka

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: ListenHTTP filter to Kafka.....4

Prerequisites..... 4

List of required configuration parameters for the ListenHTTP filter to Kafka ReadyFlow..... 7

ReadyFlow overview: ListenHTTP filter to Kafka

You can use the ListenHTTP filter to Kafka ReadyFlow to listen to JSON, CSV or Avro events on a specified port and write them to Cloudera Kafka.

This ReadyFlow listens to a JSON, CSV or Avro data stream on a specified port and parses the schema by looking up the schema name in the Cloudera Schema Registry. You can filter events by specifying a SQL query in the 'Filter Rule' parameter. The filtered events are then converted to the specified output data format and written to the destination Cloudera Kafka topic. Failed Kafka write operations are retried automatically to handle transient issues. Define a KPI on the 'failure_WriteToKafka' connection to monitor failed write operations.

Note:

The default filter criteria allows all records to pass through. The ListenHTTP processor is configured to use mutual TLS authentication.

ListenHTTP filter to Kafka ReadyFlow details	
Source	ListenHTTP Processor
Source Format	JSON, CSV, Avro
Destination	CDP Kafka Topic
Destination Format	JSON, CSV, Avro

Moving data with a ListenHTTP filter to Kafka flow

You can use a ListenHTTP filter to Kafka data flow when you want to stream records from a non-Cloudera source location to a Cloudera managed Kafka Topic.

Prerequisites

Learn how to collect the information you need to deploy the ListenHTTP filter to Kafka ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the port to listen on for incoming HTTP events.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.

- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.

- You have created at least one Kafka topic.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:

     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
         "name":"sensor_1",
         "doc":"Reading #1.",
         "type":"int"
      },
      {
         "name":"sensor_2",
         "doc":"Reading #2.",
         "type":"int"
      },
      {
         "name":"sensor_3",
         "doc":"Reading #3.",
         "type":"int"
      }
   ]
```

```
}
```

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have the Kafka broker end points.
  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.
- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

Related Concepts

List of required configuration parameters for the ListenHTTP filter to Kafka ReadyFlow

List of required configuration parameters for the ListenHTTP filter to Kafka ReadyFlow

When deploying the ListenHTTP filter to Kafka ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

ListenHTTP filter to Kafka ReadyFlow configuration parameters	
Parameter Name	**Description**
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kafka. Ensure this user has the appropriate access rights to the destination Kafka topic.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against Kafka.
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Data Input Format	Specify the format of your input data. You can use "CSV", "JSON", or "AVRO" with this ReadyFlow.
Data Output Format	Specify the desired format for your output data. You can use "CSV", "JSON" or "AVRO" with this ReadyFlow.
Filter Rule	Specify the filter rule expressed in SQL to filter streaming events for the destination topic. Records matching the filter will be written to the destination topic. The default value forwards all records.
Kafka Broker Endpoint	Specify the Kafka bootstrap servers string as a comma separated list.
Kafka Destination Topic	Specify the destination topic name.
Kafka Producer ID	Specify the id for the Kafka producer writing into the destination topic.
Listening Port	Specify the port to listen on for incoming connections.
Schema Name	Specify the schema name to be looked up in the Schema Registry.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

ListenSyslog filter to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: ListenSyslog filter to S3/ADLS.....4

Prerequisites.....4

List of required configuration parameters for the ListenSyslog filter to S3/ADLS ReadyFlow.....7

ReadyFlow: ListenSyslog filter to S3/ADLS

You can use the ListenSyslog filter to S3/ADLS ReadyFlow to listen to Syslog events on a specified port, filter them, and write them as JSON, CSV or Avro files to S3 or ADLS.

This ReadyFlow listens to a Syslog data stream on a specified port. You can filter events by specifying a SQL query in the Filter Rule parameter. The default filter criteria allows all records to pass through. The filtered events are then converted to the specified output data format and written to the target S3 or ADLS destination. The flow writes out a file every time its size has either reached 100MB or five minutes have passed. Files can reach a maximum size of 1GB. Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

The ListenSyslog processor is configured to use mutual TLS authentication.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

ReadyFlow details	
Source	ListenSyslog Processor
Source Format	Syslog
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	JSON, CSV, Avro

Moving data to object stores

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost-effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Prerequisites

Learn how to collect the information you need to deploy the ListenSyslog to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the port to listen on for incoming Syslog events.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the ListenSyslog filter to S3/ADLS ReadyFlow

List of required configuration parameters for the ListenSyslog filter to S3/ADLS ReadyFlow

When deploying the ListenSyslog filter to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: ListenSyslog filter to S3/ADLS ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
CSV Delimiter	If your desired output data is CSV, specify the delimiter here.

Parameter Name	Description
Data Output Format	Specify the format of your output data. You can use • CSV • JSON • AVRO with this ReadyFlow.
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS container you want to write to. • For S3, enter a value in the form: s3a://*[***DESTINATION S3 BUCKET***]* • For ADLS, enter a value in the form: abfs://*[***DESTINATION ADLS FILE SYSTEM***]@[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net
Filter Rule	Specify the filter rule expressed in SQL to filter events for the destination object store. Records matching the filter will be written to the destination object store. The default value forwards all records.
Listening Port	Specify the port to listen on for incoming connections. The default value is 7002.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

ListenTCP filter to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: ListenTCP filter to S3/ADLS.....4

Prerequisites..... 4

List of required configuration parameters for the ListenTCP filter to S3/ADLSReadyFlow.....7

ReadyFlow: ListenTCP filter to S3/ADLS

You can use the ListenTCP filter to S3/ADLS ReadyFlow to listen to TCP events on a specified port, filter them, and write them as JSON, CSV or Avro files to S3 or ADLS.

This ReadyFlow listens to a JSON, CSV or Avro data stream on a specified port and parses the data based on a specified Avro-formatted schema. You can filter events by specifying a SQL query in the 'Filter Rule' parameter. The filtered events are then converted to the specified output data format and written to the target S3 or ADLS destination. The flow writes out a file every time its size has either reached 100MB or five minutes have passed. Files can reach a maximum size of 1GB. Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the 'failure_WriteToS3/ADLS' connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

ReadyFlow details	
Source	ListenTCP Processor
Source Format	Avro, JSON, or CSV
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	Avro, JSON, or CSV

Prerequisites

Learn how to collect the information you need to deploy the ListenTCP filter to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the port to listen on for incoming TCP events.
- You have the Avro-formatted schema for your source event data.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.

- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the ListenTCP filter to S3/ADLSReadyFlow

List of required configuration parameters for the ListenTCP filter to S3/ADLSReadyFlow

When deploying the ListenTCP filter to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: ListenTCP filter to S3/ADLS ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
CSV Delimiter	If your desired output data is CSV, specify the delimiter here.

Parameter Name	Description
Data Input Format	Specify the format of your input data. You can use • CSV • JSON • AVRO with this ReadyFlow.
Data Output Format	Specify the format of your output data. You can use • CSV • JSON • AVRO with this ReadyFlow.
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS container you want to write to. • For S3, enter a value in the form: s3a://*[***DESTINATION S3 BUCKET***]* • For ADLS, enter a value in the form: abfs://*[***DESTINATION ADLS FILE SYSTEM***]@[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net
Filter Rule	Specify the filter rule expressed in SQL to filter events for the destination object store. Records matching the filter will be written to the destination object store. The default value forwards all records.
Listening Port	Specify the port to listen on for incoming connections. The default value is 7003.
Schema Text	Specify the Avro-formatted schema to be used for the source event data.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

MongoDB CDC to Kudu

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: MongoDB CDC to Kudu [Technical Preview]..... 4

Prerequisites..... 4

List of required configuration parameters for the MongoDB CDC to Kudu [Technical Preview] ReadyFlow..... 6

ReadyFlow: MongoDB CDC to Kudu [Technical Preview]

You can use the MongoDB CDC to Kudu [Technical Preview] ReadyFlow to retrieve CDC events from a MongoDB source table and stream them to a Kudu destination table.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from a MySQL source table to stream the events to a Kudu destination table. Failed Kudu write operations are retried automatically to handle transient issues. Define a KPI on the failure_ModifyKuduTable connection to monitor failed write operations.

Note:

This ReadyFlow does not support schema changes or primary key field updates. Make sure to assign the correct permissions for the Kudu destination table to the specified Cloudera Workload User.

MongoDB CDC to Kudu [Technical Preview] ReadyFlow details	
Source	MongoDB Table
Source Format	MongoDB Table
Destination	Kudu
Destination Format	Kudu Table

Prerequisites

Learn how to collect the information you need to deploy the MongoDB CDC to Kudu [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the MySQL database server hostname and port.
- You have obtained the MySQL schema name and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the MySQL table.
- You have performed the MySQL setup tasks required to run Debezium.

For DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have a Real-Time Data Mart cluster running Kudu, Impala, and Hue in the same environment for which Cloudera DataFlow has been enabled.
- You have the Kudu Master hostnames.
  1. From Management Console, click Data Hub Clusters.
  2. Select the Real-Time Data Mart cluster to which you want to ingest data into.
  3. Click the Hardware tab.
  4. Copy the FQDN for each Kudu Master.

- You have the Kudu database name.
  1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
  2. Click the Tables icon on the left pane.
  3. Select the default database.
- You have created the Kudu table that you want to ingest data into. Ensure that the field case sensitivity matches that of the source table.
  1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
  2. Click the Tables icon on the left pane.
  3. Select the default database, and click + New to create a new table.
  4. In the Type field, select Manually and click Next.
  5. Provide the table Name, Format, Primary keys, and any partitions.
  6. Click Submit. The newly created table displays in the default database Tables pane.
  7. Check the Kudu UI **Tables** tab for the name of the table you created. You will need this table name when you use the Cloudera DataFlow Deployment wizard to deploy the ReadyFlow.
- You have assigned permissions via IDBroker or in Ranger to enable the Cloudera Workload User to access the Kudu table that you want to ingest data into.
  1. From the base cluster on Cloudera Public Cloud, select Ranger.
  2. Select your Real Time Data Mart cluster from the **Kudu** folder.
  3. Click Add New Policy policy.
  4. On the **Create Policy** page, enter the Kudu table name in the topic field.
  5. Add the Cloudera Workload User in the Select User field.
  6. Add the Insert and Select permissions in the Permissions field.
  7. Click Save.

Related Concepts

List of required configuration parameters for the MongoDB CDC to Kudu [Technical Preview] ReadyFlow

List of required configuration parameters for the MongoDB CDC to Kudu [Technical Preview] ReadyFlow

When deploying the MongoDB CDC to Kudu [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: MongoDB CDC to Kudu [Technical Preview] ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the CDP machine user or workload user name that you want to use to authenticate to Kafka and Kudu. Ensure this user has the appropriate access rights to the Kudu table.
CDP Workload User Password	Specify the password of the CDP machine user or workload user you are using to authenticate to Kudu.
Destination Database Table Name	Specify the destination database table name in the form: [database_name].[table_name].
Kudu Master Hosts	Specify the Kudu Master hostnames in a comma separated list.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.
Source Database Server Port	Specify the source database server port. The default value is 3306.

Parameter Name	Description
Source Database Table Name	Specify the source database table name in the form: [schema_name].[table_name].
Source Database User	Specify the source database user.

Related Concepts

List of required configuration parameters for the MongoDB CDC to Kudu [Technical Preview] ReadyFlow

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

MQTT filter to Kafka

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: MQTT filter to Kafka **4**

Prerequisites **4**

List of required configuration parameters for the MQTT filter to Kafka ReadyFlow **7**

ReadyFlow: MQTT filter to Kafka

You can use the MQTT filter to Kafka ReadyFlow to move your data from an MQTT source to a Kafka topic.

This ReadyFlow consumes JSON, CSV or Avro data from a source MQTT topic and parses the schema by looking up the schema name in the Cloudera Schema Registry. You can filter events by specifying a SQL query in the 'Filter Rule' parameter. The filtered events are then converted to the specified output data format and written to the destination Kafka topic. Failed Kafka write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToKafka connection to monitor failed write operations.

Note: The default filter criteria allows all records to pass through.

ReadyFlow details	
Source	MQTT Processor
Source Format	JSON, CSV, Avro
Destination	Kafka Topic
Destination Format	JSON, CSV, Avro

Moving data with MQTT filter to Kafka flow

MQTT is a messaging protocol for the Internet of Things (IoT) that is used in a wide variety of industries. You can use the MQTT filter to Kafka ReadyFlow to read data from a MQTT topic, parse the data using a schema provided either through the Cloudera Schema Registry or through direct schema input. You can then filter or transform the data using a SQL query before it is written to a destination topic in a Cloudera Kafka cluster.

Prerequisites

Learn how to collect the information you need to deploy the MQTT filter to Kafka ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the MQTT Broker URI.
- You have the MQTT Broker Username and Password to authenticate to the MQTT Broker.
- You have the MQTT Topic Filter to designate which source topics to subscribe to.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.

- You have created at least one Kafka topic.

  1. Navigate to Management Console > Environments and select your environment.
  2. Select your Streams Messaging cluster.
  3. Click on the Streams Messaging Manager icon.
  4. Navigate to the Topics page.
  5. Click Add New and provide the following information:

     - Topic name
     - Number of partitions
     - Level of availability
     - Cleanup policy

     Tip:

     SMM has automatically set Kafka topic configuration parameters. To manually adjust them, click Advanced.
  6. Click Save.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
         "name":"sensor_1",
         "doc":"Reading #1.",
         "type":"int"
      },
      {
         "name":"sensor_2",
         "doc":"Reading #2.",
         "type":"int"
      },
      {
         "name":"sensor_3",
         "doc":"Reading #3.",
         "type":"int"
      }
   ]
```

```
}
```

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have the Kafka broker end points.
  1. From the Management Console, click Data Hub Clusters.
  2. Select the Streams Messaging cluster from which you want to ingest data.
  3. Click the Hardware tab.
  4. Note the Kafka Broker FQDNs for each node in your cluster.
  5. Construct your Kafka Broker Endpoints by using the FQDN and Port number 9093 separated by a colon. Separate endpoints by a comma. For example:

     ```
     broker1.fqdn:9093,broker2.fqdn:9093,broker3.fqdn:9093
     ```

     Kafka broker FQDNs are listed under the **Core_broker** section.
- You have the Kafka Consumer Group ID.

  This ID is defined by the user. Pick an ID and then create a Ranger policy for it. Use the ID when deploying the flow in Cloudera DataFlow.
- You have assigned the Cloudera Workload User policies to access the consumer group ID and topic.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Kafka** folder.
  4. Create a policy to enable your Workload User to access the Kafka source topic.
  5. On the **Create Policy** page, give the policy a name, select topic from the drop-down list, add the user, and assign the Consume permission.
  6. Create another policy to give your Workload User access to the consumer group ID.
  7. On the **Create Policy** page, give the policy a name, select consumergroup from the drop-down list, add the user, and assign the Consume permission.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

Related Concepts

List of required configuration parameters for the MQTT filter to Kafka ReadyFlow

List of required configuration parameters for the MQTT filter to Kafka ReadyFlow

When deploying the MQTT filter to Kafka ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: MQTT filter to Kafka ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kafka. Ensure this user has the appropriate access rights to the destination Kafka topic.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against Kafka.
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Data Input Format	Specify the format of your input data. Supported values are: • CSV • JSON • AVRO
Data Output Format	Specify the desired format for your output data. Supported values are: • CSV • JSON • AVRO
Filter Rule	Specify the filter rule expressed in SQL to filter streaming events for the destination topic. Records matching the filter will be written to the destination topic. The default value forwards all records.
Kafka Broker Endpoint	Specify the Kafka bootstrap servers string as a comma separated list.
Kafka Destination Topic	Specify the destination topic name.
Kafka Producer ID	Specify the id for the Kafka producer writing into the destination topic.
MQTT Broker Password	Specify the password to use when connecting to the MQTT broker.
MQTT Broker URI	Specify the URI to use to connect to the MQTT broker.
MQTT Broker Username	Specify the username to use when connecting to the MQTT broker.
MQTT Topic Filter	Specify the MQTT topic filter to designate the topics to subscribe to.
Schema Name	Specify the schema name to be looked up in the Schema Registry.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

MySQL CDC to Iceberg

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: MySQL CDC to Iceberg [Technical Preview]....4

Prerequisites.... 4

List of required configuration parameters for the MySQL CDC to Iceberg [Technical Preview] ReadyFlow....6

ReadyFlow: MySQL CDC to Iceberg [Technical Preview]

You can use the MySQL CDC to Iceberg [Technical Preview] ReadyFlow to retrieve CDC events from a MySQL source table and stream them to an Iceberg destination table.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from a MySQL source table to stream the events to an Iceberg destination table. Failed Iceberg write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteIcebergTable connection to monitor failed write operations.

Note: This ReadyFlow is considered Technical Preview and is not designed for production use.

Note:

This ReadyFlow does not support schema changes or primary key field updates. The destination table must be Iceberg format v2. Make sure to assign the correct permissions for the Iceberg destination table to the specified CDP Workload User.

MySQL CDC to Iceberg [Technical Preview] ReadyFlow details	
Source	MySQL Table
Source Format	MySQL Table
Destination	Iceberg
Destination Format	Parquet

Prerequisites

Learn how to collect the information you need to deploy the MySQL CDC to Iceberg [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the MySQL database server hostname and port.
- You have obtained the MySQL schema name and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the MySQL table.
- You have performed the MySQL setup tasks required to run Debezium.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- In Cloudera Data Warehouse, you have activated the same environment for which Cloudera DataFlow has been enabled. This will create a default database catalog. For more information, see Activating an AWS environment from Cloudera Data Warehouse or Activating Azure environments.
- You have created a Hive Virtual Warehouse referencing the default database catalog. For more information, see Creating your first Virtual Warehouse.
- You have created the Iceberg table that you want to ingest data into, running in your Hive Virtual Warehouse. For more information, see Iceberg table creation from Hive.

Related Concepts

List of required configuration parameters for the MySQL CDC to Iceberg [Technical Preview] ReadyFlow

List of required configuration parameters for the MySQL CDC to Iceberg [Technical Preview] ReadyFlow

When deploying the MySQL CDC to Iceberg [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: MySQL CDC to Iceberg [Technical Preview] ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Hive Ensure this user has the appropriate access rights to the Iceberg table.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate to Hive.
CDPEnvironment	The Cloudera Environment configuration resources.
Hive Catalog Namespace	Specify the Hive catalog namespace.
Iceberg Table Name	Specify the Iceberg table name you want to write to. Table must be Iceberg format v2.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.
Source Database Server Port	Specify the source database server port. The default value is 3306.
Source Database Table Name	Specify the source database table name in the form: [schema_name].[table_name].
Source Database User	Specify the source database user.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

MySQL CDC to Kudu

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: MySQL CDC to Kudu **4**

Prerequisites **4**

List of required configuration parameters for the MySQL CDC to Kudu ReadyFlow **6**

ReadyFlow: MySQL CDC to Kudu

You can use the MySQL CDC to Kudu ReadyFlow to retrieve CDC events from a MySQL source table and stream them to a Kudu destination table.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from a MySQL source table to stream the events to a Kudu destination table. Failed Kudu write operations are retried automatically to handle transient issues. Define a KPI on the failure_ModifyKuduTable connection to monitor failed write operations.

Note:

This ReadyFlow does not support schema changes or primary key field updates. Make sure to assign the correct permissions for the Kudu destination table to the specified Cloudera Workload User.

MySQL CDC to Kudu ReadyFlow details	
Source	MySQL Table
Source Format	MySQL Table
Destination	Kudu
Destination Format	Kudu Table

Prerequisites

Learn how to collect the information you need to deploy the MySQL CDC to Kudu ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the MySQL database server hostname and port.
- You have obtained the MySQL schema name and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the MySQL table.
- You have performed the MySQL setup tasks required to run Debezium.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have a Real-Time Data Mart cluster running Kudu, Impala, and Hue in the same environment for which Cloudera DataFlow has been enabled.
- You have the Kudu Master hostnames.
  1. From Management Console, click Data Hub Clusters.
  2. Select the Real-Time Data Mart cluster to which you want to ingest data into.
  3. Click the Hardware tab.
  4. Copy the FQDN for each Kudu Master.

- You have the Kudu database name.
  1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
  2. Click the Tables icon on the left pane.
  3. Select the default database.
- You have created the Kudu table that you want to ingest data into. Ensure that the field case sensitivity matches that of the source table.
  1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
  2. Click the Tables icon on the left pane.
  3. Select the default database, and click + New to create a new table.
  4. In the Type field, select Manually and click Next.
  5. Provide the table Name, Format, Primary keys, and any partitions.
  6. Click Submit. The newly created table displays in the default database Tables pane.
  7. Check the Kudu UI **Tables** tab for the name of the table you created. You will need this table name when you use the Cloudera DataFlow Deployment wizard to deploy the ReadyFlow.
- You have assigned permissions via IDBroker or in Ranger to enable the Cloudera Workload User to access the Kudu table that you want to ingest data into.
  1. From the base cluster on Cloudera Public Cloud, select Ranger.
  2. Select your Real Time Data Mart cluster from the **Kudu** folder.
  3. Click Add New Policy policy.
  4. On the **Create Policy** page, enter the Kudu table name in the topic field.
  5. Add the Cloudera Workload User in the Select User field.
  6. Add the Insert and Select permissions in the Permissions field.
  7. Click Save.

Related Concepts

List of required configuration parameters for the MySQL CDC to Kudu ReadyFlow

List of required configuration parameters for the MySQL CDC to Kudu ReadyFlow

When deploying the MySQL CDC to Kudu ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: MySQL CDC to Kudu ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kudu. Ensure this user has the appropriate access rights to the Kudu table.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate to Kudu.
Destination Database Table Name	Specify the destination database table name in the form: [database_name].[table_name].
Kudu Master Hosts	Specify the Kudu Master hostnames in a comma separated list.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.
Source Database Server Port	Specify the source database server port. The default value is 3306.

Parameter Name	Description
Source Database Table Name	Specify the source database table name in the form: [schema_name].[table_name].
Source Database User	Specify the source database user.

Related Concepts

List of required configuration parameters for the MySQL CDC to Kudu ReadyFlow

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Non-CDP ADLS to CDP ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: Non-CDP ADLS to CDP ADLS..... 4

Prerequisites..... 4

List of required configuration parameters for the Non-CDP ADLS to CDP ADLS ReadyFlow..... 6

ReadyFlow overview: Non-CDP ADLS to CDP ADLS

You can use the Non-CDP ADLS to ADLS ReadyFlow to move data between a non-Cloudera managed source ADLS location and a Cloudera managed destination ADLS location.

This ReadyFlow consumes files from an external source Azure Data Lake Service (ADLS) location and writes them to the destination Cloudera managed ADLS location. You can specify the source and target locations. The ReadyFlow polls the source container for new files by periodically listing its content.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

Non-CDP ADLS to CDP ADLS ReadyFlow details	
Source	External ADLS
Source Format	Any
Destination	Cloudera managed ADLS
Destination Format	Same as source

Prerequisites

Learn how to collect the information you need to deploy the Non-CDP ADLS to CDP ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the source ADLS file system/container name, path, and storage account name and key.
- You have access to the source ADLS directory.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have the destination ADLS file system/containeer name, path, and storage account name.

- You have performed one of the following to configure access to the destination ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.

       a. To access IDBroker mappings in your environment, click Actions Manage Access .

       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Non-CDP ADLS to CDP ADLS ReadyFlow

List of required configuration parameters for the Non-CDP ADLS to CDP ADLS ReadyFlow

When deploying the Non-CDP ADLS to CDP ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

You have to provide the following parameters during Non-CDP ADLS to CDP ADLS ReadyFlow deployment. For more information, see *Deploying a ReadyFlow*.

Table 1: Non-CDP ADLS to CDP ADLS ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the destination object store. Ensure this user has the appropriate access rights to the destination object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the destination object stores (via IDBroker).

Parameter Name	Description
Destination ADLS File System	Specify the file system name you want to write to. The full path will be constructed from: abfs://#{Destination ADLS File System}@#{Destination ADLS Storage Account}.dfs.core.windows.net/#{Destination ADLS Path}
Destination ADLS Path	Specify the path within the ADLS data container where you want to write to. Make sure that the path starts with "/". The full path will be constructed from: abfs://#{Destination ADLS File System}@#{Destination ADLS Storage Account}.dfs.core.windows.net/#{Destination ADLS Path}
Destination ADLS Storage Account	Specify the storage account name you want to write to. The full path will be constructed from: abfs://#{Destination ADLS File System}@#{Destination ADLS Storage Account}.dfs.core.windows.net/#{Destination ADLS Path}
Source ADLS File System	Specify the name of the source ADLS file system also known as container.
Source ADLS Path	Specify the name of the ADLS path you want to read from.
Source ADLS Storage Account Key	Specify the source ADLS storage account key.
Source ADLS Storage Account Name	Specify the source ADLS storage account name.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Non-CDP ADLS to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Non-CDP ADLS to S3/ADLS.....4

Prerequisites..... 4

List of required configuration parameters for the Non-CDP ADLS to S3/ADLS ReadyFlow..... 7

ReadyFlow: Non-CDP ADLS to S3/ADLS

You can use the Non-CDP ADLS to S3/ADLS ReadyFlow to move your data from a non-Cloudera ADLS to a destination S3 or ADLS.

This ReadyFlow consumes files from an external source non-Cloudera ADLS location and writes them to a Cloudera managed destination S3 or ADLS location. For the source, specify the ADLS file system, path, and storage account name and key. For the destination, specify the S3 or ADLS storage location and path. The ReadyFlow polls the source container for new files (it performs a listing periodically). Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

ReadyFlow details	
Source	External ADLS
Source Format	Any
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	Same as source

Prerequisites

Learn how to collect the information you need to deploy the Non-CDP ADLS to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the source ADLS file system/container name, path, and storage account name and key.
- You have access to the source ADLS directory.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.

- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have the destination ADLS file system/container name, path, and storage account name.

- You have performed one of the following to configure access to the destination ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- You have performed one of the following to configure access to the destination S3 bucket:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Non-CDP ADLS to S3/ADLS ReadyFlow

List of required configuration parameters for the Non-CDP ADLS to S3/ADLS ReadyFlow

When deploying the Non-CDP ADLS to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Non-CDP ADLS to S3/ADLS ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".

Parameter Name	Description
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS container you want to write to. • For S3, enter a value in the form: ns3a:*//[***DESTINATION S3 BUCKET***]* • For ADLS, enter a value in the form: nabfs:*//[***DESTINATION ADLS FILE SYSTEM***]@[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net
Source ADLS File System	Specify the name of the source ADLS file system also known as container.
Source ADLS Path	Specify the name of the ADLS path you want to read from.
Source ADLS Storage Account Key	Specify the source ADLS storage account key.
Source ADLS Storage Account Name	Specify the source ADLS storage account name.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Non-CDP S3 to CDP S3

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: Non-CDP S3 to CDP S3 4

Prerequisites 4

List of required configuration parameters for the Non-CDP S3 to CDP S3 ReadyFlow 6

ReadyFlow overview: Non-CDP S3 to CDP S3

You can use the Non-CDP S3 to CDP S3 ReadyFlow to move data between a non-Cloudera managed source S3 location and a Cloudera managed destination S3 location.

This ReadyFlow consumes files from an external source S3 location and writes them to the destination Cloudera managed S3 location. You can specify the source and target locations. You can pick whether to preserve or ignore the folder hierarchy of the source files when writing to the destination path. The ReadyFlow polls the source bucket for new files by periodically listing its content.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

Non-CDP S3 to CDP S3 ReadyFlow details	
Source	External Amazon S3
Source Format	Any
Destination	Cloudera managed Amazon S3
Destination Format	Same as source

Prerequisites

Learn how to collect the information you need to deploy the Non-CDP S3 to CDP S3 ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the source S3 region, bucket, path, access key ID, and secret access key.
- You have access to the source S3 bucket.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have the destination S3 path and bucket into which you want to ingest data.

- You have performed one of the following to configure access to the destination S3 bucket:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Non-CDP S3 to CDP S3 ReadyFlow

List of required configuration parameters for the Non-CDP S3 to CDP S3 ReadyFlow

When deploying the Non-CDP S3 to CDP S3 ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Non-CDP S3 to S3 ReadyFlow configuration parameters	
Parameter Name	**Description**
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
Destination S3 Bucket	Specify the name of the destination (Cloudera managed) S3 bucket you want to write to. The full path is constructed from: s3a://#{Destination S3 Bucket}/#{Destination S3 Path}

Non-CDP S3 to S3 ReadyFlow configuration parameters	
Parameter Name	**Description**
Destination S3 Path	Specify the path within the destination bucket where you want to write to. Make sure that the path starts with "/". The full path is constructed from: s3a://#{Destination S3 Bucket}/#{Destination S3 Path}
Preserve S3 Prefix	Specify whether to preserve or ignore the folder hierarchy of the source files. Set to "TRUE" to preserve folder hierarchy in the destination path. Set to "FALSE" to flatten the hierarchy and only keep the file in the destination path.
Source S3 Access Key ID	Specify the source (external) S3 access key ID.
Source S3 Bucket	Specify the name of the source (external) S3 bucket you want to read from.
Source S3 Path	Specify the name of the S3 path you want to read from.
Source S3 Region	Specify the AWS region in which your source (external) bucket was created.
Source S3 Secret Access Key	Specify the source (external) S3 secret access key.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Non-CDP S3 to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Non-CDP S3 to S3/ADLS.....4

Prerequisites..... 4

List of required configuration parameters for the Non-CDP S3 to S3/ADLS ReadyFlow..... 7

ReadyFlow: Non-CDP S3 to S3/ADLS

You can use the Non-CDP S3 to S3/ADLS ReadyFlow to move your data from a non-Cloudera Amazon S3 to a destination S3 or ADLS.

This ReadyFlow consumes files from an external source non-Cloudera S3 location and writes them to a Cloudera managed destination S3 or ADLS location. For the source, specify the S3 bucket region, name, path and access keys. For the destination, specify the S3 or ADLS storage location and path. The ReadyFlow polls the source bucket for new files (it performs a listing periodically). You can pick whether to preserve or ignore the folder hierarchy of the source files when writing to the destination path. Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

ReadyFlow details	
Source	External Amazon S3
Source Format	Any
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	Same as source

Prerequisites

Learn how to collect the information you need to deploy the Non-CDP S3 to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the source S3 region, bucket, path, access key ID, and secret access key.
- You have access to the source S3 bucket.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.

- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have the destination ADLS file system/containeer name, path, and storage account name.

- You have performed one of the following to configure access to the destination ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- You have performed one of the following to configure access to the destination S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Non-CDP S3 to S3/ADLS ReadyFlow

List of required configuration parameters for the Non-CDP S3 to S3/ADLS ReadyFlow

When deploying the Non-CDP S3 to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Non-CDP S3 to S3/ADLS ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".

Parameter Name	Description
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS container you want to write to. • For S3, enter a value in the form: ns3a:*//[***DESTINATION S3 BUCKET***]* • For ADLS, enter a value in the form: nabfs:*//[***DESTINATION ADLS FILE SYSTEM***]@[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net
Preserve S3 Prefix	Specify whether to preserve or ignore the folder hierarchy of the source files. • Set to "TRUE" to preserve folder hierarchy in the destination path. • Set to "FALSE" to flatten the hierarchy and only keep the file in the destination path.
Source S3 Access Key ID	Specify the source (external) S3 access key ID.
Source S3 Bucket	Specify the name of the source (external) S3 bucket you want to read from.
Source S3 Path	Specify the name of the S3 path you want to read from. Make sure that the path does not start with "/".
Source S3 Region	Specify the AWS region in which your source (external) bucket was created.
Source S3 Secret Access Key	Specify the source (external) S3 secret access key.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Oracle CDC to Iceberg

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Oracle CDC to Iceberg [Technical Preview]....4

Prerequisites.... 4

List of required configuration parameters for the Oracle CDC to Iceberg [Technical Preview] ReadyFlow....6

ReadyFlow: Oracle CDC to Iceberg [Technical Preview]

You can use the Oracle CDC to Iceberg ReadyFlow to retrieve CDC events from an Oracle source table and stream them to an Iceberg table.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from an Oracle source table to stream the events to an Iceberg destination table. Failed Iceberg write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteIcebergTable connection to monitor failed write operations.

Note: This ReadyFlow is considered Technical Preview and is not designed for production use.

Note:

This ReadyFlow does not support schema changes or primary key field updates. The destination table must be Iceberg format v2. Make sure to assign the correct permissions for the Iceberg destination table to the specified CDP Workload User.

Oracle CDC to Iceberg [Technical Preview] ReadyFlow details	
Source	Oracle Table
Source Format	Oracle Table
Destination	Iceberg Table
Destination Format	Parquet

Prerequisites

Learn how to collect the information you need to deploy the Oracle CDC to Iceberg [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the Oracle database server hostname and port.
- You have obtained the Oracle database name , schema name and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the Oracle table.
- You have obtained the Oracle database driver jar file.
- You have performed the Oracle Server setup tasks required to run Debezium.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- In Cloudera Data Warehouse, you have activated the same environment for which Cloudera DataFlow has been enabled. This will create a default database catalog. For more information, see Activating an AWS environment from Cloudera Data Warehouse or Activating Azure environments.
- You have created a Hive Virtual Warehouse referencing the default database catalog. For more information, see Creating your first Virtual Warehouse.
- You have created the Iceberg table that you want to ingest data into, running in your Hive Virtual Warehouse. For more information, see Iceberg table creation from Hive.

Related Concepts

List of required configuration parameters for the Oracle CDC to Iceberg [Technical Preview] ReadyFlow

List of required configuration parameters for the Oracle CDC to Iceberg [Technical Preview] ReadyFlow

When deploying the Oracle CDC to Iceberg [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Oracle CDC to Iceberg [Technical Preview] ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Hive. Ensure this user has the appropriate access rights to the Iceberg table.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate to Hive.
CDPEnvironment	The Cloudera Environment configuration resources.
Hive Catalog Namespace	Specify the Hive catalog namespace.
Iceberg Table Name	Specify the Iceberg table name you want to write to. Table must be Iceberg format v2.
Source Database Driver	Upload the database driver jar file for your source database.
Source Database Name	Specify the source database name.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.
Source Database Server Port	Specify the source database server port. The default value is 1521.
Source Database Table Name	Specify the source database table name in the form: [schema_name].[table_name]
Source Database User	Specify the source database user.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Oracle CDC to Kudu

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Oracle CDC to Kudu.....4

Prerequisites..... 4

List of required configuration parameters for the Oracle CDC to Kudu ReadyFlow..... 6

ReadyFlow: Oracle CDC to Kudu

You can use the Oracle CDC to Kudu ReadyFlow to retrieve CDC events from an Oracle source table and stream them to a Kudu destination table.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from an Oracle source table to stream the events to a Kudu destination table. Failed Kudu write operations are retried automatically to handle transient issues. Define a KPI on the failure_ModifyKuduTable connection to monitor failed write operations.

Note:

This ReadyFlow does not support schema changes or primary key field updates. Make sure to assign the correct permissions for the Kudu destination table to the specified Cloudera Workload User.

Oracle CDC to Kudu ReadyFlow details	
Source	Oracle Table
Source Format	Oracle Table
Destination	Kudu
Destination Format	Kudu Table

Prerequisites

Learn how to collect the information you need to deploy the Oracle CDC to Kudu ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the Oracle database server hostname and port.
- You have obtained the Oracle database name , schema name and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the Oracle table.
- You have obtained the Oracle database driver jar file.
- You have performed the Oracle Server setup tasks required to run Debezium.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have a Real-Time Data Mart cluster running Kudu, Impala, and Hue in the same environment for which Cloudera DataFlow has been enabled.
- You have the Kudu Master hostnames.
  1. From Management Console, click Data Hub Clusters.
  2. Select the Real-Time Data Mart cluster to which you want to ingest data into.
  3. Click the Hardware tab.
  4. Copy the FQDN for each Kudu Master.
- You have the Kudu database name.

- You have created the Kudu table that you want to ingest data into. Ensure that the field case sensitivity matches that of the source table.
  1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
  2. Click the Tables icon on the left pane.
  3. Select the default database, and click + New to create a new table.
  4. In the Type field, select Manually and click Next.
  5. Provide the table Name, Format, Primary keys, and any partitions.
  6. Click Submit. The newly created table displays in the default database Tables pane.
  7. Check the Kudu UI **Tables** tab for the name of the table you created. You will need this table name when you use the Cloudera DataFlow Deployment wizard to deploy the ReadyFlow.
- You have assigned permissions via IDBroker or in Ranger to enable the Cloudera Workload User to access the Kudu table that you want to ingest data into.
  1. From the base cluster on Cloudera Public Cloud, select Ranger.
  2. Select your Real Time Data Mart cluster from the **Kudu** folder.
  3. Click Add New Policy policy.
  4. On the **Create Policy** page, enter the Kudu table name in the topic field.
  5. Add the Cloudera Workload User in the Select User field.
  6. Add the Insert and Select permissions in the Permissions field.
  7. Click Save.

Related Concepts

List of required configuration parameters for the Oracle CDC to Kudu ReadyFlow

List of required configuration parameters for the Oracle CDC to Kudu ReadyFlow

When deploying the Oracle CDC to Kudu ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Oracle CDC to Kudu ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kudu. Ensure this user has the appropriate access rights to the Kudu table.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate to Kudu.
Destination Database Table Name	Specify the destination database table name in the form: [database_name].[table_name]
Kudu Master Hosts	Specify the Kudu Master hostnames in a comma separated list.
Source Database Driver	Upload the database driver jar file for your source database.
Source Database Name	Specify the source database name.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.
Source Database Server Port	Specify the source database server port. The default value is 1521.
Source Database Table Name	Specify the source database table name in the form: [schema_name].[table_name]
Source Database User	Specify the source database user.

Related Concepts

List of required configuration parameters for the Oracle CDC to Kudu ReadyFlow

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

PostgreSQL CDC to Iceberg

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: PostgreSQL CDC to Iceberg [Technical Preview]....4

Prerequisites.... 4

List of required configuration parameters for the PostgreSQL CDC to Iceberg [Technical Preview] ReadyFlow....6

ReadyFlow: PostgreSQL CDC to Iceberg [Technical Preview]

You can use the PostgreSQL CDC to Iceberg [Technical Preview] ReadyFlow to retrieve CDC events from a PostgreSQL source table and stream them into Iceberg.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from a PostgreSQL source table to stream the events to an Iceberg destination table. Failed Iceberg write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteIcebergTable connection to monitor failed write operations.

Note: This ReadyFlow is considered Technical Preview and is not designed for production use.

Note:

This ReadyFlow does not support schema changes or primary key field updates. The destination table must be Iceberg format v2. Make sure to assign the correct permissions for the Iceberg destination table to the specified CDP Workload User.

PostgreSQL CDC to Iceberg [Technical Preview] ReadyFlow details	
Source	PostgreSQL Table
Source Format	PostgreSQL Table
Destination	Iceberg
Destination Format	Parquet

Prerequisites

Learn how to collect the information you need to deploy the PostgreSQL CDC to Iceberg [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the PostgreSQL database server hostname and port.
- You have obtained the PostgreSQL schema name and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the PostgreSQL table.
- You have performed the PostgreSQL setup tasks required to run Debezium.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- In Cloudera Data Warehouse, you have activated the same environment for which Cloudera DataFlow has been enabled. This will create a default database catalog. For more information, see Activating an AWS environment from Cloudera Data Warehouse or Activating Azure environments.
- You have created a Hive Virtual Warehouse referencing the default database catalog. For more information, see Creating your first Virtual Warehouse.
- You have created the Iceberg table that you want to ingest data into, running in your Hive Virtual Warehouse. For more information, see Iceberg table creation from Hive.

Related Concepts

List of required configuration parameters for the PostgreSQL CDC to Iceberg [Technical Preview] ReadyFlow

List of required configuration parameters for the PostgreSQL CDC to Iceberg [Technical Preview] ReadyFlow

When deploying the PostgreSQL CDC to Iceberg [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: PostgreSQL CDC to Iceberg [Technical Preview] ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Hive. Ensure this user has the appropriate access rights to the Iceberg table.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate to Hive.
CDPEnvironment	The Cloudera Environment configuration resources.
Hive Catalog Namespace	Specify the Hive catalog namespace.
Iceberg Table Name	Specify the Iceberg table name you want to write to. Table must be Iceberg format v2.
Source Database Name	Specify the source database name.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.
Source Database Server Port	Specify the source database server port. The default value is 5432.
Source Database Table Name	Specify the source database table name in the form:[schema_name].[table_name].
Source Database User	Specify the source database user.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

PostgreSQL CDC to Kudu

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: PostgreSQL CDC to Kudu **4**

Prerequisites **4**

List of required configuration parameters for the PostgreSQL CDC to Kudu ReadyFlow **6**

ReadyFlow: PostgreSQL CDC to Kudu

You can use the PostgreSQL CDC to Kudu ReadyFlow to retrieve CDC events from a PostgreSQL source table and stream them to a Kudu destination table.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from a PostgreSQL source table to stream the events to a Kudu destination table. Failed Kudu write operations are retried automatically to handle transient issues. Define a KPI on the failure_ModifyKuduTable connection to monitor failed write operations.

Note:

This ReadyFlow does not support schema changes or primary key field updates. Make sure to assign the correct permissions for the Kudu destination table to the specified Cloudera Workload User.

PostgreSQL CDC to Kudu ReadyFlow details	
Source	PostgreSQL Table
Source Format	PostgreSQL Table
Destination	Kudu
Destination Format	Kudu Table

Prerequisites

Learn how to collect the information you need to deploy the PostgreSQL CDC to Kudu ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the PostgreSQL database server hostname and port.
- You have obtained the PostgreSQL schema name and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the PostgreSQL table.
- You have performed the PostgreSQL setup tasks required to run Debezium.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have a Real-Time Data Mart cluster running Kudu, Impala, and Hue in the same environment for which Cloudera DataFlow has been enabled.
- You have the Kudu Master hostnames.
  1. From Management Console, click Data Hub Clusters.
  2. Select the Real-Time Data Mart cluster to which you want to ingest data into.
  3. Click the Hardware tab.
  4. Copy the FQDN for each Kudu Master.

- You have created the Kudu table that you want to ingest data into.
  1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
  2. Click the Tables icon on the left pane.
  3. Select the default database, and click + New to create a new table.
  4. In the Type field, select Manually and click Next.
  5. Provide the table Name, Format, Primary keys, and any partitions.
  6. Click Submit. The newly created table displays in the default database Tables pane.
  7. Check the Kudu UI **Tables** tab for the name of the table you created. You will need this table name when you use the Cloudera DataFlow Deployment wizard to deploy the ReadyFlow.
- You have assigned permissions via IDBroker or in Ranger to enable the Cloudera Workload User to access the Kudu table that you want to ingest data into.
  1. From the base cluster on Cloudera Public Cloud, select Ranger.
  2. Select your Real Time Data Mart cluster from the **Kudu** folder.
  3. Click Add New Policy policy.
  4. On the **Create Policy** page, enter the Kudu table name in the topic field.
  5. Add the Cloudera Workload User in the Select User field.
  6. Add the Insert and Select permissions in the Permissions field.
  7. Click Save.

Related Concepts

List of required configuration parameters for the PostgreSQL CDC to Kudu ReadyFlow

List of required configuration parameters for the PostgreSQL CDC to Kudu ReadyFlow

When deploying the PostgreSQL CDC to Kudu ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: PostgreSQL CDC to Kudu ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to Kudu. Ensure this user has the appropriate access rights to the Kudu table.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate to Kudu.
Destination Database Table Name	Specify the destination database table name in the form: [database_name].[table_name]
Kudu Master Hosts	Specify the Kudu Master hostnames in a comma separated list.
Source Database Name	Specify the source database name.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.
Source Database Server Port	Specify the source database server port. The default value is 5432.
Source Database Table Name	Specify the source database table name.
Source Database User	Specify the source database user in the form: [schema_name].[table_name]

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

RAG Query Milvus

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: RAG Query Milvus [Technical Preview]....4

Prerequisites.... 4

List of required configuration parameters for the RAG Query Milvus [Technical Preview] ReadyFlow....6

ReadyFlow overview: RAG Query Milvus [Technical Preview]

You can use the RAG Query Milvus [Technical Preview] ReadyFlow to query Milvus VectorDB with an embedded prompt.

This ReadyFlow generates and vectorizes a query from a user-specified question. It then performs a similarity search using one of five search metrics (L2, IP, COSINE, JACCARD, HAMMING) on the specified Milvus collection and returns the result with the shortest distance, extracting the specified fields. The flow then performs a lexical search based on the specified fields and returns all rows with the returned 'section', storing them as a text attribute. A prompt is created from the original user query and generated text document. The prompt is then passed to ChatGPT for an answer to the user query. A Milvus access token and an OpenAI API key are required to run this flow. Define a KPI on the failure_AskChatGPT connection to monitor failed responses.

Note: This ReadyFlow is considered Technical Preview and is not designed for production use. The flow uses Python processors and must be run in NiFi 2.x.

To ensure correct functionality, make sure the HuggingFace embedding model specified is the same model used to insert data in the queried Milvus collection and that the Milvus collection has been created with the expected schema.

RAG Query Milvus [Technical Preview] ReadyFlow details	
Source	Custom prompt-query
Source Format	Natural language
Destination	Milvus
Destination Format	Vector DB

Prerequisites

Learn how to collect the information you need to deploy the RAG Query Milvus [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have a query in the form of a question that is relevant to the data in your Milvus collection.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have the OpenAI API key.
- Your Milvus version is 2.4.4.
- You have the Milvus access token.
- You have the name of the destination Milvus collection.
- You have the URI of the destination Milvus instance.

- Your Milvus collection schema has the following field names and field types:
  - id *(INT64)* You must enable Auto ID on this field.
  - text_embedding *(FLOAT_VECTOR)*
  - source *(VARCHAR)*
  - section *(VARCHAR)*
  - text *(VARCHAR)*
- The ‘text_embedding’ field in your destination Milvus collection is configured with the same dimensions as the default HuggingFace 'all-MiniLM-L12-v2' model (384).
- The source, section and text VARCHAR fields have been configured with Max Length values large enough to accommodate their respective values in your PDFs.
- Your Milvus collection has data loaded that is relevant to your prompt query. For example, PDFs related to the query have been loaded to Milvus using the “S3 to Milvus” or “ADLS to Milvus” ReadyFlows.

  Note: The flow ends with a response from ChatGPT received by a placeholder processor (LogAttribute). Replace it with the processor(s) required to support your use case for the response data. This may be an integration with productivity applications like Slack or Discord, a web UI, or any other customer facing user interface.

-

Related Concepts

List of required configuration parameters for the RAG Query Milvus [Technical Preview] ReadyFlow

List of required configuration parameters for the RAG Query Milvus [Technical Preview] ReadyFlow

When deploying the RAG Query Milvus [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: RAG Query Milvus [Technical Preview] ReadyFlow configuration parameters

Parameter name	Description
HuggingFace Embedding Model	Specify the HuggingFace model name to use for embedding the data. The default model is 'all-MiniLM-L12-v2'.
Milvus Access Token	Specify the access token for the destination Milvus instance.
Milvus Collection Name	Specify the name of the destination Milvus collection.
Milvus Instance URI	Specify the URI of the destination Milvus instance.
OpenAI API Key	Specify the API key used to authenticate to OpenAI.
OpenAI Model Name	Specify the OpenAI model name. The default model is 'gpt-4o-mini'.
User Query	Specify the question to include in the ChatGPT prompt.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

S3 to CDW

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: S3 to Cloudera Data Warehouse.....4

Prerequisites.....4

List of required configuration parameters for the S3 to Cloudera Data Warehouse ReadyFlow.....7

ReadyFlow: S3 to Cloudera Data Warehouse

You can use the S3 to Cloudera Data Warehouse Readyflow to consume CSV files from a source S3 location and write them as Parquet files to a destination S3 location and Cloudera Data Warehouse Impala table.

This ReadyFlow consumes CSV files from a source S3 location, parses the schema by looking up the schema name in the Cloudera Schema Registry, converts the files into Parquet and writes them to a destination S3 location and Cloudera Data Warehouse Impala table. You can specify the source S3 location, the target S3 location and the destination Impala table name. The ReadyFlow polls the source bucket for new files (it performs a listing periodically). Define KPIs on the failure_WriteS3 and failure_CreateCDWImpalaTable connections to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

S3 to Cloudera Data Warehouse ReadyFlow details	
Source	Cloudera Public Cloud Managed Amazon S3
Source Format	CSV
Destination	Cloudera Public Cloud Managed Amazon S3 and Cloudera Data Warehouse Impala
Destination Format	Parquet

Prerequisites

Learn how to collect the information you need to deploy the S3 to Cloudera Data Warehouse ReadyFlow, and meet other prerequisites.

For your S3 data ingest source and target

- You have the two S3 buckets and their paths as source and destination for the data movement.

- You have performed one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.
- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
```

```
            "name":"sensor_1",
            "doc":"Reading #1.",
            "type":"int"
        },
        {
            "name":"sensor_2",
            "doc":"Reading #2.",
            "type":"int"
        },
        {
            "name":"sensor_3",
            "doc":"Reading #3.",
            "type":"int"
        }
    ]
}
```

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your Impala data ingest target

- You have activated your environment in Cloudera Data Warehouse. This automatically creates a default Database Catalog.
- You have created an Impala Virtual Warehouse referencing the default Database Catalog. Uncheck the Enable SSO setting.
- Select Copy the JDBC URL from the Virtual Warehouse UI. Use this connection string as the basis for the Impala JDBC URL parameter.
- For a RAZ enabled environment, you have assigned the CDP Workload User read and write access to the URL path where the Parquet files will be written via the Hadoop_SQL_URL policy.

Related Concepts

List of required configuration parameters for the S3 to Cloudera Data Warehouse ReadyFlow

List of required configuration parameters for the S3 to Cloudera Data Warehouse ReadyFlow

When deploying the S3 to Cloudera Data Warehouse ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: S3 to Cloudera Data Warehouse ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to the object stores (via IDBroker) and to the schema registry. Ensure this user has the appropriate access rights to the object store locations and to the schema registry.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker) and the schema registry.
Destination Database Driver	Upload the database driver jar file for your destination database.
Destination Database Driver Class Name	Specify the destination database driver class name, for example, com.cloudera.impala.jdbc.Driver for Impala databases.
Destination Database Table Name	Specify the destination database table name.
Destination S3 Bucket	Specify the name of the destination S3 bucket you want to write to. The full path will be constructed out of s3a://#{Destination S3 Bucket}/#{Destination S3 Path}
Destination S3 Path	Specify the path within the destination bucket where you want to write to. The full path will be constructed out of s3a://#{Destination S3 Bucket}/#{Destination S3 Path}
Impala JDBC URL	Specify the Impala JDBC URL from the CDW Virtual Warehouse. Do not include the UID and PASSWORD connection parameters. The full JDBC URL will be constructed from: #{Impala JDBC URL};UID=#{CDP Workload User};PWD=#{CDP Workload User Password} where PWD value is provided by the dynamic property SENSITIVE.PWD in the Impala connection pool service configuration.
Schema Name	Specify the schema name to be looked up in the Schema Registry used to parse the source files.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.
Source S3 Bucket	Specify the name of the source S3 bucket you want to read from.
Source S3 Path	Specify the path within the source bucket where you want to read files from.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

S3 to Databricks

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: S3 to Databricks.....4

Prerequisites..... 4

List of required configuration parameters for the S3 to Databricks ReadyFlow..... 8

ReadyFlow: S3 to Databricks

You can use the S3 to Databricks ReadyFlow to retrieve CSV files from a source S3 location and write them as Parquet files to a destination S3 location and Databricks table.

This ReadyFlow consumes CSV data from a source S3 location, parses the data using schema(s) provided by the Cloudera Schema Registry, converts it to Parquet format and writes the data to a destination S3 bucket and Databricks table. The flow supports either a non-partitioned table or a partitioned table (single column only).

Note:

By default, the flow is configured for a partitioned Databricks table. This ReadyFlow leverages Cloudera's centralized access control for cloud storage access. Make sure to either set up an IDBroker mapping or Ranger policies when using fine-grained object store access allowing your workload user access to the source and destination S3 locations.

S3 to Databricks ReadyFlow details	
Source	Cloudera managed Amazon S3
Source Format	CSV
Destination	Cloudera managed Amazon S3 and Databricks
Destination Format	Parquet

Prerequisites

Learn how to collect the information you need to deploy the S3 to Databricks ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the source S3 bucket and path.

- You have performed one of the following to configure access to the source S3 bucket:
  - You have configured access to the S3 bucket with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera Public Cloud credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to the S3 bucket using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to the S3 bucket using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.
- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
```

```
            "name":"sensor_1",
            "doc":"Reading #1.",
            "type":"int"
        },
        {
            "name":"sensor_2",
            "doc":"Reading #2.",
            "type":"int"
        },
        {
            "name":"sensor_3",
            "doc":"Reading #3.",
            "type":"int"
        }
    ]
}
```

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have created a Databricks table, non-partitioned or partitioned (single column only).
- You have the Storage Location of your Databricks Table, which consists of the S3 Bucket, Path and Table Id.

- You have performed one of the following to configure access to the target S3 bucket:
  - You have configured access to the S3 bucket with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera Public Cloud credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to the S3 bucket using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to the S3 bucket using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the S3 to Databricks ReadyFlow

List of required configuration parameters for the S3 to Databricks ReadyFlow

When deploying the S3 to Databricks ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: S3 to Databricks ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to the object stores and to the schema registry. Ensure this user has the appropriate access rights to the object store locations and to the schema registry in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores and the schema registry.
CDPEnvironment	The Cloudera Environment configuration resources.
Destination S3 Bucket	Specify the name of the destination S3 bucket you want to write to. The full path will be constructed out of s3a://#{Destination S3 Bucket}/#{Destination S3 Path}

Parameter Name	Description
Destination S3 Path	Specify the path within the destination bucket where you want to write to. Make sure that the path starts with "/". The path must end with the destination Databricks Table Id. The full path is constructed out of s3a://#{Destination S3 Bucket}/#{Destination S3 Path}
Partition Column	Specify the name of the column used to partition your destination Databricks table. This ReadyFlow only supports a single partition column.
Partition Column Exists	Specify whether the destination Databricks column is partitioned. The default value is YES.
Schema Name	Specify the schema name to be looked up in the Schema Registry used to parse the source files.
Schema Name 2	If your Databricks table is partitioned, specify the name of the modified schema to be looked up in the Schema Registry. This schema should not include the partition column field.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.
Source S3 Bucket	Specify the name of the source S3 bucket you want to read from.
Source S3 Path	Specify the path within the source bucket where you want to read files from.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

S3 to Milvus

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: S3 to Milvus [Technical Preview]....4

Prerequisites.... 4

List of required configuration parameters for the S3 to Milvus [Technical Preview] ReadyFlow....7

ReadyFlow overview: S3 to Milvus [Technical Preview]

You can use the S3 to Milvus [Technical Preview] ReadyFlow to consume PDF documents from S3, vectorize them using a HuggingFace model and write the results to Milvus.

This ReadyFlow consumes PDF documents from a source S3 location, partitions the PDFs, chunks the data, vectorizes the data using a HuggingFace embedding model, and stores the results in Milvus vector DB. The default HuggingFace model is 'all-MiniLM-L12-v2'. A Milvus access token is required to run this flow. Define a KPI on the failure_WriteToMilvus connection to monitor failed write operations.

Note: This ReadyFlow is considered Technical Preview and is not designed for production use.

The flow uses Python processors and must be run in NiFi 2.x.

To ensure correct functionality, make sure your Milvus collection has been created with the expected schema and the same dimension as the 'all-MiniLM-L12-v2' model (384). The images within PDFs will be ignored.

Note:

This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the source S3 location.

S3 to Milvus [Technical Preview] ReadyFlow details	
Source	Cloudera Public Cloud managed Amazon S3
Source Format	PDF
Destination	Milvus
Destination Format	Vector DB

Prerequisites

Learn how to collect the information you need to deploy the S3 to Milvus [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the source S3 bucket and path.

- You have performed one of the following to configure access to the source S3 bucket:
  - You have configured access to the S3 bucket with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera Public Cloud credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to the S3 bucket using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to the S3 bucket using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- Your Milvus version is 2.4.4.
- You have the Milvus access token.
- You have the name of the destination Milvus collection.
- You have the URI of the destination Milvus instance.
- Your Milvus collection schema has the following field names and field types:
  - id *(INT64)* You must enable Auto ID on this field.
  - text_embedding *(FLOAT_VECTOR)*
  - source *(VARCHAR)*
  - section *(VARCHAR)*
  - text *(VARCHAR)*
- The ‘text_embedding’ field in your destination Milvus collection is configured with the same dimensions as the default HuggingFace 'all-MiniLM-L12-v2' model (384).
- The source, section and text VARCHAR fields have been configured with Max Length values large enough to accommodate their respective values in your PDFs.

Related Concepts

List of required configuration parameters for the S3 to Milvus [Technical Preview] ReadyFlow

List of required configuration parameters for the S3 to Milvus [Technical Preview] ReadyFlow

When deploying the S3 to Milvus [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: S3 to Milvus [Technical Preview] ReadyFlow configuration parameters

Parameter name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores.
CDPEnvironment	The Cloudera Environment configuration resources.
HuggingFace Embedding Model	Specify the HuggingFace model name to use for embedding the data. The default model is 'all-MiniLM-L12-v2'.
Milvus Access Token	Specify the access token for the destination Milvus instance.
Milvus Collection Name	Specify the name of the destination Milvus collection.
Milvus Instance URI	Specify the URI of the destination Milvus instance.
Source S3 Bucket	Specify the name of the source S3 bucket you want to read from.
Source S3 Path	Specify the path within the source bucket where you want to read files from.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

S3 to Pinecone

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: S3 to Pinecone [Technical Preview]..... 4

Prerequisites..... 4

List of required configuration parameters for the S3 to Pinecone [Technical Preview] ReadyFlow..... 6

ReadyFlow overview: S3 to Pinecone [Technical Preview]

You can use the S3 to Pinecone [Technical Preview] ReadyFlow to consume PDF documents from S3, vectorize them using an OpenAI model and write the results to Pinecone.

This ReadyFlow consumes PDF documents from a source S3 location, partitions the PDFs, chunks the data, and transforms/splits the data into the required input format for the PutPinecone processor. The PutPinecone processor then vectorizes the data and writes the embeddings to a destination Pinecone index. The default OpenAI model is 'text-embedding-ada-002'. An OpenAI API key and a Pinecone API key are required to run this flow. Define a KPI on the failure_WriteToPinecone connection to monitor failed write operations.

Note: This ReadyFlow is considered Technical Preview and is not designed for production use.

The flow uses Python processors and must be run in NiFi 2.x.

To ensure correct functionality, make sure your Pinecone index has been created with the same dimension as the 'text-embedding-ada-002' model (1536). The images within PDFs will be ignored.

Note:

This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the source S3 location.

S3 to Pinecone [Technical Preview] ReadyFlow details	
Source	Cloudera managed Amazon S3
Source Format	PDF
Destination	Pinecone
Destination Format	Vector DB

Prerequisites

Learn how to collect the information you need to deploy the S3 to Pinecone [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the source S3 bucket and path.

- You have performed one of the following to configure access to the source S3 bucket:
  - You have configured access to the S3 bucket with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera Public Cloud credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to the S3 bucket using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to the S3 bucket using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have your OpenAI API key.
- You have your Pinecone environment and index into which you want to ingest data.
- Your Pinecone index was configured with the ’text-embedding-ada-002 model’.
- You have your Pinecone API key.

Related Concepts

List of required configuration parameters for the S3 to Pinecone [Technical Preview] ReadyFlow

List of required configuration parameters for the S3 to Pinecone [Technical Preview] ReadyFlow

When deploying the S3 to Pinecone [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: S3 to Pinecone [Technical Preview] ReadyFlow configuration parameters

Parameter name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores.
CDPEnvironment	The Cloudera Environment configuration resources.
OpenAI API Key	Specify the API key used to authenticate to OpenAI.
OpenAI Model	Specify the OpenAI model name to use for embedding the data. The default model is 'text-embedding-ada-002'.
Pinecone API Key	Specify the API key used to authenticate to Pinecone.
Pinecone Environment	Specify the name of the Pinecone environment.
Pinecone Index	Specify the name of the Pinecone index. To ensure correct functionality of this ReadyFlow, make sure your Pinecone index has been created with the same dimension as the 'text-embedding-ada-002' model (1536).
Source S3 Bucket	Specify the name of the source S3 bucket you want to read from.
Source S3 Path	Specify the path within the source bucket where you want to read files from.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

S3 to S3 Avro

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: S3 to S3 Avro **4**

Prerequisites **4**

List of required configuration parameters for the S3 to S3 Avro ReadyFlow **7**

ReadyFlow: S3 to S3 Avro

You can use the S3 to S3 Avro ReadyFlow to move data between source and destination S3 locations while converting the files into Avro format.

This ReadyFlow consumes JSON, CSV or Avro files from a source S3 location, converts the files into Avro and writes them to the destination S3 location. You can specify the source format, the source and target location as well as the schema to use for reading the source data.

S3 to S3 Avro ReadyFlow details	
Source	Amazon S3
Source Format	JSON, CSV, Avro
Destination	Amazon S3
Destination Format	Avro

Moving data with an S3 to S3 Avro flow

You can use an S3 to S3 Avro data flow when you want to move data from a location in AWS S3 to another S3 location, and at the same time convert the data to Avro format. You need to specify the source format, the source and target location as well as the schema to use for handling the data. Your flow can consume JSON, CSV or Avro files from the source S3 location. It converts the files into Avro format and writes them to the destination S3 location. You define and store the data processing schema in the Schema Registry on a Cloudera Streams Messaging Data Hub cluster. The data flow parses the schema by looking up the schema name in the Schema Registry.

Prerequisites

Learn how to collect the information you need to deploy the S3 to S3 Avro ReadyFlow, and meet other prerequisites.

For your data source and target

- You have the two S3 buckets and their paths as source and destination for the data movement.

- You have performed one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.
- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
```

```
            "name":"sensor_1",
            "doc":"Reading #1.",
            "type":"int"
        },
        {
            "name":"sensor_2",
            "doc":"Reading #2.",
            "type":"int"
        },
        {
            "name":"sensor_3",
            "doc":"Reading #3.",
            "type":"int"
        }
    ]
}
```

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

Related Concepts

List of required configuration parameters for the S3 to S3 Avro ReadyFlow

List of required configuration parameters for the S3 to S3 Avro ReadyFlow

When deploying the S3 to S3 Avro ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: S3 to S3 Avro ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate Schema Registry. Ensure this user has the appropriate access rights in Ranger.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against Schema Registry.

Parameter Name	Description
CDP Environment	Cloudera DataFlow uses this parameter to auto-populate the Flow Deployment with Hadoop configuration files required to interact with S3. Cloudera DataFlow automatically adds all required configuration files to interact with Data Lake services. Unnecessary files that are added will not impact the deployment process.
Data Input Format	Specify the format of your input data. If your data input format is CSV, define a CSV delimiter for the data in the CSV Delimiter text box. If you use AVRO or JSON format, the delimiter is ignored. • CSV • JSON • AVRO
Data Output Format	Specify the format of your output data. As you are using AVRO format, the delimiter is ignored.
CSV Delimiter	If your source data is CSV, specify the delimiter here.
S3 Bucket	Specify the name of the S3 bucket that you want to read from. The full path is constructed from: s3a://#{S3 Bucket}/#{S3 Path}/${S3 Bucket}
S3 Path	Specify the path within the bucket where you want to read from without any leading characters. The full path is constructed from: s3a://#{S3 Bucket}/#{S3 Path}/${S3 Bucket}
S3 Bucket	Specify the name of the S3 bucket that you want to write to. The full path is constructed from: s3a://#{S3 Bucket}/#{S3 Path}/${S3 Bucket}
S3 Path	Specify the path within the bucket where you want to write to without any leading characters. The full path is constructed from: s3a://#{S3 Bucket}/#{S3 Path}/${S3 Bucket}

Parameter Name	Description
S3 Bucket Region	Specify the AWS region in which your bucket was created. Supported values are: • us-gov-west-1 • us-gov-east-1 • us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-west-1 • eu-west-2 • eu-west-3 • eu-central-1 • eu-north-1 • eu-south-1 • ap-east-1 • ap-south-1 • ap-southeast-1 • ap-southeast-2 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • sa-east-1 • cn-north-1 • cn-northwest-1 • ca-central-1 • me-south-1 • af-south-1 • us-iso-east-1 • us-isob-east-1 • us-iso-west-1
S3 Path	Specify the path within the bucket where you want to write to without any leading characters. The full path will be constructed from: s3a://#{S3 Bucket}/#{S3 Path}/${Kafka.topic}
Schema Name	Identify the schema that you want to use in your data flow. Cloudera DataFlow looks up this schema in the Schema Registry you define with the Schema Registry Hostname. See the Appendix for an example schema.
Schema Registry Hostname	Specify the hostname of the Schema Registry running on the master node in the Streams Messaging cluster that you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

S3 to S3 Avro with S3 Notifications

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: S3 to S3 Avro with S3 Notifications.....4

Prerequisites..... 4

List of required configuration parameters for the S3 to S3 Avro with S3 Notifications ReadyFlow..... 8

ReadyFlow overview: S3 to S3 Avro with S3 Notifications

You can use the S3 to S3 Avro with S3 Notifications ReadyFlow to move your data between Amazon S3 buckets and to configure notifications.

This ReadyFlow consumes JSON, CSV, or Avro files from a source S3 location, converts the files into Avro and then writes them to a destination S3 location. You can specify the source format, the source and target location and the schema to use for reading the source data. The ReadyFlow is notified about the new files in the source bucket by configuring S3 notifications in AWS.

ReadyFlow details	
Source	Amazon S3
Source Format	JSON, CSV, Avro
Destination	Amazon S3
Destination Format	Avro

Moving data between S3 locations

You can use this ReadyFlow when you want to move data from a location in AWS S3 to another S3 location, and at the same time convert the data to Avro format. You need to specify the source format, the source and target location as well as the schema to use for handling the data. Your flow can consume JSON, CSV or Avro files from the source S3 location. It converts the files into Avro format and writes them to the destination S3 location. You define and store the data processing schema in the Schema Registry on a Cloudera Streams Messaging Data Hub cluster. The data flow parses the schema by looking up the schema name in the Schema Registry.

Setting up S3 notifications

S3 buckets can be configured to generate notifications when new data arrives. You can configure this ReadyFlow to receive S3 notifications from an SQS queue so that new files are automatically moved from source to destination directory

Prerequisites

Learn how to collect the information you need to deploy the S3 to S3 Avro ReadyFlow, and meet other prerequisites.

For your data source and target

- You have the two S3 buckets and their paths as source and destination for the data movement.

- You have performed one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.
- Create an ID Broker Mapping for the Cloudera Workload User and the IAM role that you want to use in the SQS access policy.
  1. Access IDBroker mappings.
     a. To access IDBroker mappings in your environment, click Actions Manage Access .
     b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
  2. Add your Cloudera Workload User and the corresponding AWS role that provides access to the SQS Queue.

     Note:

     You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
  3. Click Save and Sync.

- You have configured your Simple Queue Service URL for S3 bucket notifications.

  For detailed instructions, see Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue)

  Note:

  Update the access policy in *Step 1: Create an Amazon SQS Queue* with information about the ID Broker Mapping for the Cloudera Workload User and the IAM role that you want to use in the SQS access policy. The updated policy will look similar to:

```
{
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::[***bucket-owner-account-id***]:rol
e/[***IAM-ROLE***]"
      },
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
      ],
      "Resource": "arn:aws:sqs:[***region***]:[***queue-owner-
account-id***]:[***queue-name***]"
    }
```

- You have created a Streams Messaging cluster in Cloudera Public Cloud to host your Schema Registry.

  For information on how to create a Streams Messaging cluster, see Setting up your Streams Messaging Cluster.
- You have created a schema for your data and have uploaded it to the Schema Registry in the Streams Messaging cluster.

  For information on how to create a new schema, see Creating a new schema. For example:

```
{
   "type":"record",
   "name":"SensorReading",
   "namespace":"com.cloudera.example",
   "doc":"This is a sample sensor reading",
   "fields":[
      {
         "name":"sensor_id",
         "doc":"Sensor identification number.",
         "type":"int"
      },
      {
         "name":"sensor_ts",
         "doc":"Timestamp of the collected readings.",
         "type":"long"
      },
      {
         "name":"sensor_0",
         "doc":"Reading #0.",
         "type":"int"
      },
      {
         "name":"sensor_1",
         "doc":"Reading #1.",
         "type":"int"
      },
      {
         "name":"sensor_2",
         "doc":"Reading #2.",
         "type":"int"
      },
```

```
        {
            "name":"sensor_3",
            "doc":"Reading #3.",
            "type":"int"
        }
    ]
}
```

- You have the Schema Registry Host Name.
  1. From the Management Console, go to Data Hub Clusters and select the Streams Messaging cluster you are using.
  2. Navigate to the **Hardware** tab to locate the Master Node FQDN. Schema Registry is always running on the Master node, so copy the Master node FQDN.
- You have assigned the Cloudera Workload User read-access to the schema.
  1. Navigate to Management Console > Environments, and select the environment where you have created your cluster.
  2. Select Ranger. You are redirected to the Ranger **Service Manager** page.
  3. Select your Streams Messaging cluster under the **Schema Registry** folder.
  4. Click Add New Policy.
  5. On the **Create Policy** page, give the policy a name, specify the schema details, add the user, and assign the Read permission.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

Related Concepts

List of required configuration parameters for the S3 to S3 Avro with S3 Notifications ReadyFlow

List of required configuration parameters for the S3 to S3 Avro with S3 Notifications ReadyFlow

When deploying the S3 to S3 Avro with S3 Notifications ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: S3 to S3 Avro with S3 Notifications ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to the object stores (via IDBroker) and to the schema registry. Ensure this user has the appropriate access rights to the object store locations and to the schema registry.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you're using to authenticate against Kafka and the object store.

Parameter Name	Description
CDPEnvironment	Cloudera DataFlow will use this parameter to auto-populate the Flow Deployment with Hadoop configuration files required to interact with S3. Cloudera DataFlow automatically adds all required configuration files to interact with Data Lake services. Unnecessary files that are added won't impact the deployment process.
CSV Delimiter	If your source data is CSV, specify the delimiter here.
Data Input Format	Specify the format of your input data. Possible values for this ReadyFlow are: • Avro • CSV • JSON
Destination S3 Bucket	Specify the name of the destination S3 bucket you want to write to. The full path will be constructed out of s3a://#{Destination S3 Bucket}/#{Destination S3 Path}/[subdirectory]/${filename} ([subdirectory] is the original directory of the file if the #{Source S3 Path} prefix is not specified)
Destination S3 Path	Specify the path within the destination bucket where you want to write to. The full path will be constructed out of s3a://#{Destination S3 Bucket}/#{Destination S3 Path}/[subdirectory]/${filename} ([subdirectory] is the original directory of the file if the #{Source S3 Path} prefix is not specified)
Notification SQS Queue URL	URL of the SQS Queue set up for delivering notifications about S3 object creation events. The SQS Queue must be in the same region as the source S3 Bucket and the bucket needs to be configured to send notifications to the queue.
Schema Name	Specify the schema name to be looked up in the Schema Registry used to parse the source files.
Schema Registry Hostname	Specify the hostname of the Schema Registry you want to connect to. This must be the direct hostname of the Schema Registry itself, not the Knox Endpoint.
Source S3 Bucket	Specify the name of the source S3 bucket you want to read from.
Source S3 Path	Specify the path within the source bucket where you want to read files from. If notifications are received with a source object key not matching the Source S3 Path (as a prefix), they will be filtered out and will not be loaded. The Source S3 Path will be stripped off from the source object key and will be replaced with the Destination S3 Path in the destination object key.
Source S3 Region	Specify the AWS region of your source S3 Bucket and S3 Event Notification SQS Queue. Supported values are: us-gov-west-1, us-gov-east-1, us-east-1, us-east-2, us-west-1, us-west-2, eu-west-1, eu-west-2, eu-west-3, eu-central-1, eu-north-1, eu-south-1, ap-east-1, ap-south-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, ap-northeast-2, ap-northeast-3, sa-east-1, cn-north-1, cn-northwest-1, ca-central-1, me-south-1, af-south-1, us-iso-east-1, us-isob-east-1, us-iso-west-1

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

S3 to IBM watsonx

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: S3 to IBM watsonx **4**

Prerequisites **4**

List of required configuration parameters for the S3 to IBM watsonx ReadyFlow **7**

ReadyFlow: S3 to IBM watsonx

You can use the S3 to IBM watsonx ReadyFlow to retrieve text files from a source S3 location and use IBM watsonx to summarize the text, then write the LLM response to a destination S3 location.

This ReadyFlow consumes text data from a source S3 location and invokes a large language model in IBM watsonx.ai to summarize the input data. The response from the model is written to a destination S3 location. The default LLM is ibm/granite-13b-chat-v1, but the LLM and its attributes can be changed via the IBM watsonx Model Input parameter. An IBM Project ID and IBM Cloud API Key are required to run this flow. Define a KPI on the failure_WriteS3 connection to monitor failed write operations.

Note:

This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up an IDBroker mapping or Ranger policies when using fine-grained object store access allowing your workload user access to the source and destination S3 locations.

S3 to IBM watsonx ReadyFlow details	
Source	Cloudera managed Amazon S3
Source Format	Text
Destination	watsonx + Cloudera managed Amazon S3
Destination Format	Text

Prerequisites

Learn how to collect the information you need to deploy the S3 to IBM watsonx ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have the source S3 bucket and path.

- You have performed one of the following to configure access to the source S3 bucket:
  - You have configured access to the S3 bucket with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera Public Cloud credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to the S3 bucket using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to the S3 bucket using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.
- You have given the Cloudera Workload User the EnvironmentUser role.

  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.

- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.

  1. Give a user permission to add the ReadyFlow to the Catalog.

     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.

  2. Give your user or group permission to deploy flow definitions.

     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.

     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest targets

- You have the IBM Project ID and IBM Cloud API Key.

- You have the destination S3 bucket and path. You have performed one of the following to configure access to the destination S3 bucket:
  - You have configured access to the S3 bucket with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera Public Cloud credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to the S3 bucket using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to the S3 bucket using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the S3 to IBM watsonx ReadyFlow

List of required configuration parameters for the S3 to IBM watsonx ReadyFlow

When deploying the S3 to IBM watsonx ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: S3 to IBM watsonx ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload user name that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores.
CDPEnvironment	The Cloudera Environment configuration resources.
Destination S3 Bucket	Specify the name of the destination S3 bucket you want to write to. The full path will be constructed out of s3a://#{Destination S3 Bucket}/#{Destination S3 Path}

Parameter Name	Description
Destination S3 Path	Specify the path within the destination bucket where you want to write to. The full path is constructed out of s3a://#{Destination S3 Bucket}/#{Destination S3 Path}
IBM Cloud API Key	Specify the IBM Cloud API Key needed for the IAM token request.
IBM watsonx Model Input	Specify the model input body in JSON format. You must enter your IBM project ID. The default LLM model used is ibm/granite-13b-chat-v1. Do not modify the input attribute. Update other attributes as necessary.
IBM watsonx Model URL	Specify the IBM watsonx model URL. The default value is https://us-south.ml.cloud.ibm.com/ml/v1-beta/generation/text?version=2023-05-29.
Source S3 Bucket	Specify the name of the source S3 bucket you want to read from.
Source S3 Path	Specify the path within the source bucket where you want to read files from.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Salesforce filter to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: Salesforce filter to S3/ADLS **4**

Prerequisites **4**

List of required configuration parameters for the Salesforce filter to S3/ADLS ReadyFlow **7**

ReadyFlow: Salesforce filter to S3/ADLS

You can use the Salesforce filter to S3/ADLS ReadyFlow to consume objects from a Salesforce database table, filter them, and write the data as JSON, CSV or Avro files to a destination in Amazon S3 or Azure Data Lake Service (ADLS).

This ReadyFlow consumes objects from a Salesforce database table, filtered by field name(s). You can filter events by specifying a SQL query in the Filter Rule parameter. The filtered events are then converted to the specified output data format and written to the target S3 or ADLS destination. The default filter criteria allows all records to pass through.

The flow writes out a file every time its size has either reached 100 MB or five minutes have passed. Files can reach a maximum size of 1 GB. Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

ReadyFlow details	
Source	Salesforce
Source Format	Salesforce
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	JSON, CSV, Avro

Prerequisites

Learn how to collect the information you need to deploy the Salesforce filter to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have a Salesforce account. If you do not have one, you can sign up for a free developer account here.
- You have your Salesforce username and password.
- You have your Salesforce domain name.
- You have data stored in Salesforce that you want to move to an object store.
- Your Salesforce user has the “API Enabled” permission.
- You have created a Salesforce connected app and enabled OAuth Settings for API Integration.
- You have the Consumer Key and Secret from the connected app.
- You have reset your Salesforce security token.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- You have configured access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Salesforce filter to S3/ADLS ReadyFlow

List of required configuration parameters for the Salesforce filter to S3/ADLS ReadyFlow

When deploying the Salesforce filter to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Salesforce filter to S3/ADLS ReadyFlow configuration parameters

Parameter Name	Description
Age Field	Specify the name of a TIMESTAMP field that will be used to filter records using a bounded time window. The QuerySalesforceObject processor will return only those records with a timestamp value newer than the timestamp recorded after the last processor run.
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
CSV Delimiter	If your source data is CSV, specify the delimiter here.

Parameter Name	Description
Data Output Format	Specify the desired format for your output data. You can use CSV, JSON or AVRO with this ReadyFlow.
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS container you want to write to. • For S3, enter a value in the form: s3a:*//[***DESTINATION S3 BUCKET***]* • For ADLS, enter a value in the form: abfs:*//[***DESTINATION ADLS FILE SYSTEM***]@[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net
Filter Rule	Specify the filter rule expressed in SQL to filter events for the destination object store. Records matching the filter will be written to the destination object store. The default value forwards all records.
Salesforce Consumer Key	Specify your Salesforce consumer key.
Salesforce Consumer Secret	Specify your Salesforce consumer secret.
Salesforce Domain Name	Specify your Salesforce domain name.
Salesforce Field Names	Specify the field names requested from the sObject to be queried as a comma separated list.
Salesforce Password and Security Token	Specify your Salesforce password concatenated with your Salesforce security token.
Salesforce Username	Specify your Salesforce username.
Salesforce sObject Name	Specify the Salesforce sObject to be queried.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Shopify to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: Shopify to S3/ADLS..... 4

Prerequisites..... 4

List of required configuration parameters for the Shopify to S3/ADLS ReadyFlow..... 7

ReadyFlow overview: Shopify to S3/ADLS

You can use the Shopify to S3/ADLS ReadyFlow to ingest Shopify objects, filter them, and write them as JSON, CSV or Avro files to S3 or ADLS.

This ReadyFlow consumes objects from a Custom Shopify App, converts them to the specified output data format, and writes them to a Cloudera managed destination S3 or ADLS location. For the source, specify the Shopify App's Store Domain name, Access Token, the Object Category and the Object Type. For the destination, specify the S3 or ADLS storage location and path. The flow writes out a file every time its size has either reached 100MB or five minutes have passed. Files can reach a maximum size of 1GB. Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

Shopify to S3/ADLS ReadyFlow details	
Source	Shopify
Source Format	Shopify
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	JSON, CSV, Avro

Moving data to object stores

Cloud environments offer numerous deployment options and services. There are many ways to store data in the cloud, but the easiest option is to use object stores. Object stores are extremely robust and cost-effective storage solutions with multiple levels of durability and availability. You can include them in your data pipeline, both as an intermediate step and as an end state. Object stores are accessible to many tools and connecting systems, and you have a variety of options to control access.

Prerequisites

Learn how to collect the information you need to deploy the Shopify to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have a Shopify account.
- You have your Shopify store domain name.
- You have data stored in Shopify that you want to move to an object store.
- You have the Shopify object category and object type.
- You have created a custom Shopify app.
- You have the access token from the custom app.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Shopify to S3/ADLS ReadyFlow

List of required configuration parameters for the Shopify to S3/ADLS ReadyFlow

When deploying the Shopify to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Shopify to S3/ADLS ReadyFlow configuration parameters

Parameter name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
CSV Delimiter	If your desired output data is CSV, specify the delimiter here.
Data Output Format	Specify the desired format for your output data. You can use CSV, JSON or AVRO with this ReadyFlow.
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".

Parameter name	Description
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS Container you want to write to. For S3, enter a value in the form: s3a://[Destination S3 Bucket] For ADLS, enter a value in the form: abfs://[Destination ADLS File System]@[Destination ADLS Storage Account].dfs.core.windows.net
Incremental Delay	Specify a value for this Shopify processor property to avoid missing objects when the clocks of your local machines and of Shopify servers are not in sync. This property is only used if Incremental Loading is set to true. The default value is 3 seconds.
Incremental Loading	Specify a value for this Shopify processor property to decide whether to load queried objects incrementally. Set to true to query each object exactly once. Set to false to query each object at least once. The default value is true.
Shopify Access Token	Specify the Shopify access token to authenticate requests.
Shopify Object Category	Specify the Shopify object category to request. The following values are supported: • CUSTOMERS • DISCOUNTS • INVENTORY • ONLINE_STORE • ORDERS • PRODUCTS • SALES_CHANNEL • STORE_PROPERTIES
Shopify Object Type	Specify the Shopify object type (subcategory) to request. The Object Type depends on the Object Category and the following (category: types) groups are supported: • CUSTOMERS: customers; • DISCOUNTS: price_rules; • INVENTORY: locations; • ONLINE_STORE: blogs, comments, pages, redirects, script_tags, themes; • ORDERS: checkouts, draft_orders, orders; • PRODUCTS: collects, custom_collections, products, custom_collections; • SALES_CHANNEL: collection_listings, product_listings; • STORE_PROPERTIES: countries, currencies, policies, shipping_zones;
Shopify Result Limit	Specify the maximum number of results returned by Shopify. The default value is 100.
Shopify Store Domain	Specify the domain of the Shopify store. For example, nifistore.myshopify.com

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Slack to Pinecone

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: Slack to Pinecone [Technical Preview]..... 4

Prerequisites..... 4

List of required configuration parameters for the Slack to Pinecone [Technical Preview] ReadyFlow.....8

ReadyFlow overview: Slack to Pinecone [Technical Preview]

You can use the Slack to Pinecone [Technical Preview] ReadyFlow to consume messages from a Slack channel, vectorize them using an OpenAI model and write the results to Pinecone.

This ReadyFlow reads messages from a Slack channel, vectorizes the data using an OpenAI embedding model, and stores the results in Pinecone vector DB. The default OpenAI model is 'text-embedding-ada-002'. A Slack bot access token, an OpenAI API key and a Pinecone API key are required to run this flow. Define a KPI on the failure_WriteToPinecone connection to monitor failed write operations.

Note:

This ReadyFlow is considered Technical Preview and is not designed for production use.

This ReadyFlow uses Python processors and must be run in NiFi 2.x.

To ensure correct functionality, make sure your Pinecone index has been created with the same dimension as the 'text-embedding-ada-002' model (1536). The ReadyFlow handles the Slack text messages only, it does not handle attachments in the message.

Slack to Pinecone [Technical Preview] ReadyFlow details	
Source	Slack
Source Format	Slack
Destination	Pinecone
Destination Format	Vector DB

Prerequisites

Learn how to collect the information you need to deploy the Slack to Pinecone [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have created a Slack workspace and channel.
  1. In Slack Workspace Directory click Manage Organization.

  2. Click Create Workspace.

     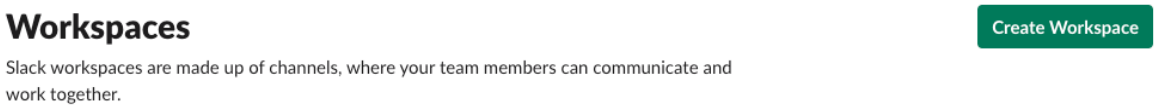

  3. Provide Workspace Name and Workspace Domain. You may leave Workspace Description empty.

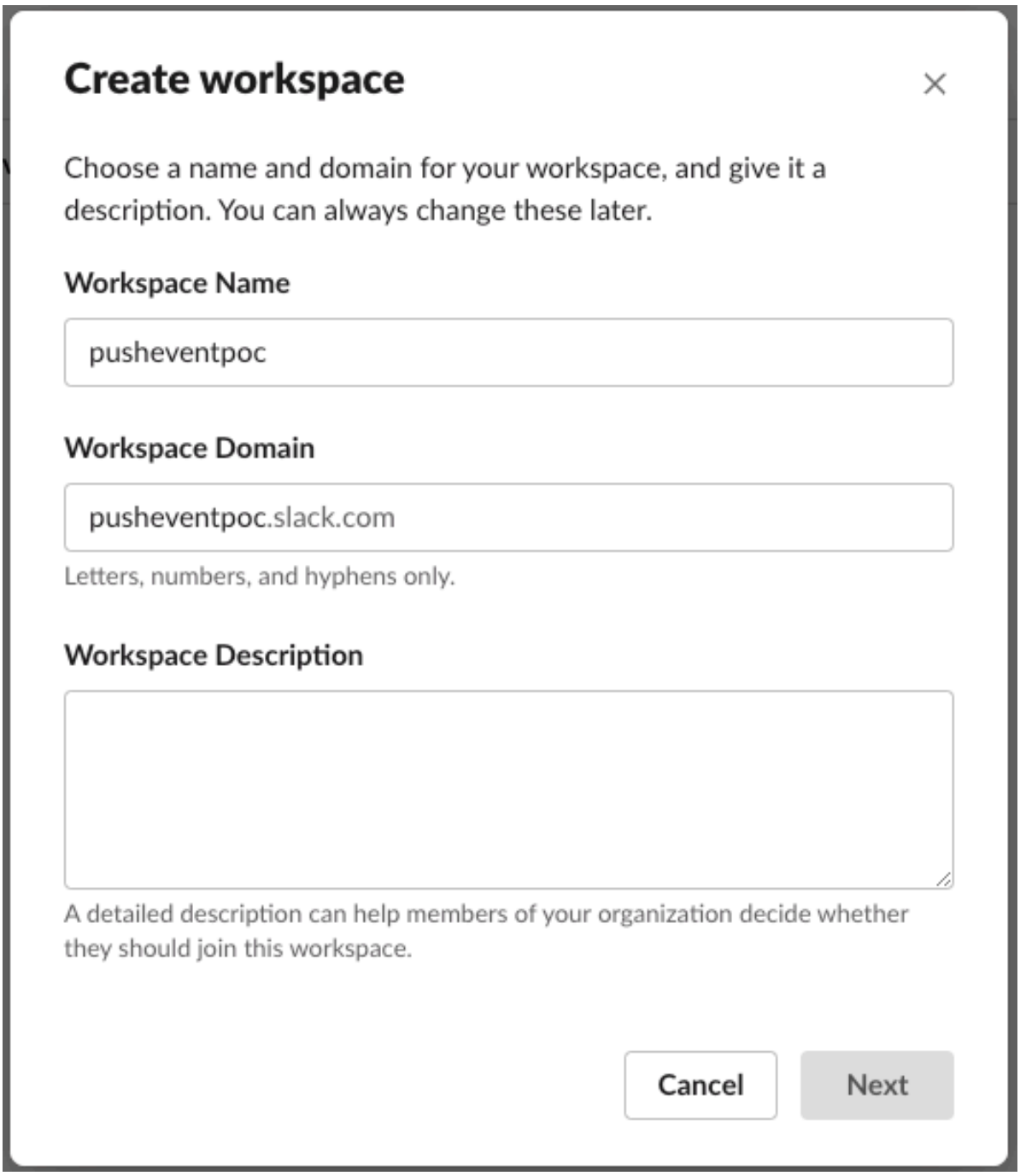

- You have your Slack User or Bot Access Token.
- You have configured the necessary scopes for your token access.

For more information on getting started with Slack Enterprise Grid sandboxes, see the Slack documentation.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have your OpenAI API key.
- You have your Pinecone environment and index into which you want to ingest data.
- Your Pinecone index was configured with the ’text-embedding-ada-002 model’.
- You have your Pinecone API key.

Related Concepts

List of required configuration parameters for the Slack to Pinecone [Technical Preview] ReadyFlow

List of required configuration parameters for the Slack to Pinecone [Technical Preview] ReadyFlow

When deploying the Slack to Pinecone [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Slack to Pinecone [Technical Preview] ReadyFlow configuration parameters

Parameter name	Description
OpenAI API Key	Specify the API key used to authenticate to OpenAI.
Pinecone API Key	Specify the API key used to authenticate to Pinecone.
Pinecone Environment	Specify the name of the Pinecone environment.
Pinecone Index	Specify the name of the Pinecone index. To ensure correct functionality of this ReadyFlow, make sure your Pinecone index has been created with the same dimension as the 'text-embedding-ada-002' model (1536).
Slack Channel ID	Specify the Slack channel ID.
Slack User or Bot Access Token	Specify the Slack user or bot access token.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Slack to S3/ADLS

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow overview: Slack to S3/ADLS..... 4

Prerequisites..... 4

List of required configuration parameters for the Slack to S3/ADLS ReadyFlow..... 11

ReadyFlow overview: Slack to S3/ADLS

You can use the Slack to S3/ADLS ReadyFlow to consume events from Slack, convert them to Avro, CSV, or JSON format and write them to a Cloudera managed destination in Amazon S3 or Azure Data Lake Service (ADLS).

This ReadyFlow consumes events from a Slack App, converts them to the specified output data format, and writes them to a Cloudera managed destination S3 or ADLS location. For the source, subscribe to the events to be notified of in Slack. For the destination, specify the S3 or ADLS storage location and path. The flow writes out a file every time its size has either reached 100MB or five minutes have passed. Files can reach a maximum size of 1GB. Failed S3 or ADLS write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteToS3/ADLS connection to monitor failed write operations.

Note: This ReadyFlow leverages Cloudera Public Cloud's centralized access control for cloud storage access. Make sure to either set up Ranger policies or an IDBroker mapping allowing your workload user access to the target S3 or ADLS location.

Slack to S3/ADLS ReadyFlow details	
Source	Slack
Source Format	Slack
Destination	Cloudera managed Amazon S3 or ADLS
Destination Format	Avro, CSV, or JSON

Prerequisites

Learn how to collect the information you need to deploy the Slack to S3/ADLS ReadyFlow, and meet other prerequisites.

For your data ingest source

- You have a Slack sandbox account.

  If you already have a Slack environment, you may skip this step. However you are highly recommended to make changes in dev/test environments before moving them to production. Slack offers a sandbox environment where such experiments can be carried out.

- You have created a Slack workspace.
  1. In Slack Workspace Directory click Manage Organization.

  2. Click Create Workspace.

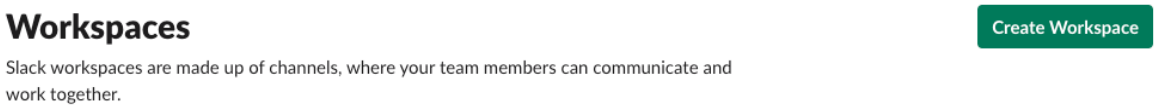

  3. Provide Workspace Name and Workspace Domain. You may leave Workspace Description empty.

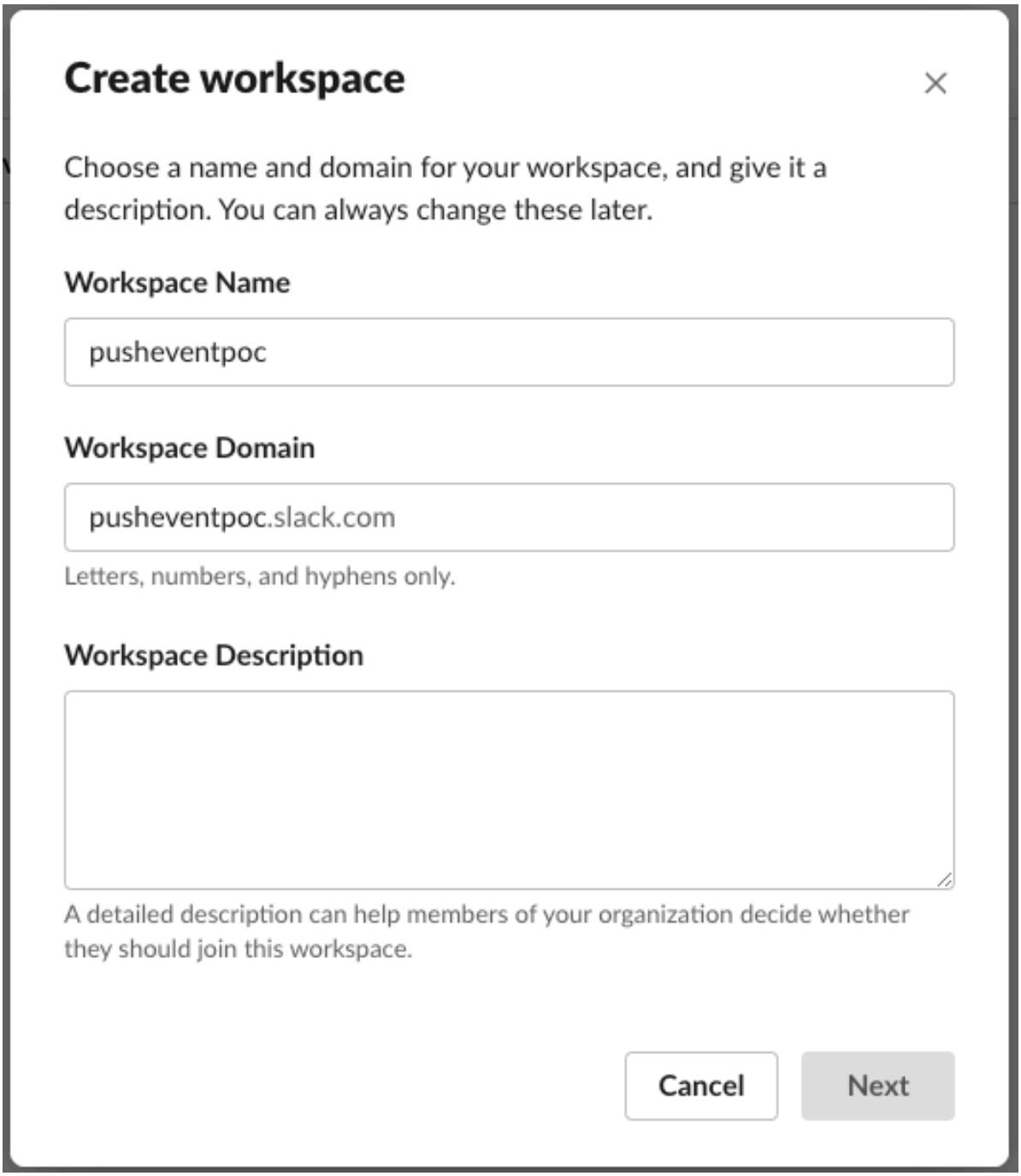

- You have created, configured, and installed a Slack App.
  1. Go to https://api.slack.com/apps, and click on Create New App.

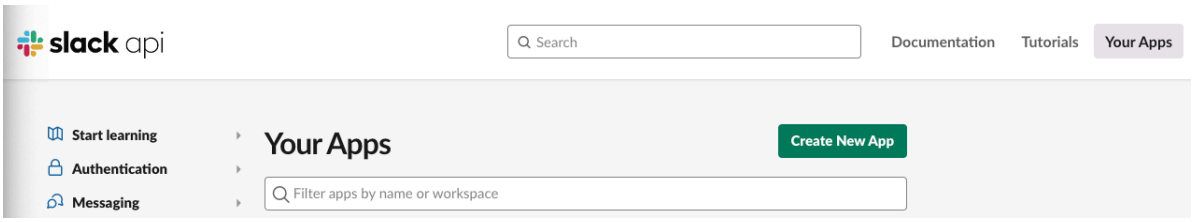

  2. Select From Scratch.
  3. Provide an App Name.
  4. Under Pick a workspace to develop your app in: select the workspace you created in the previous step.

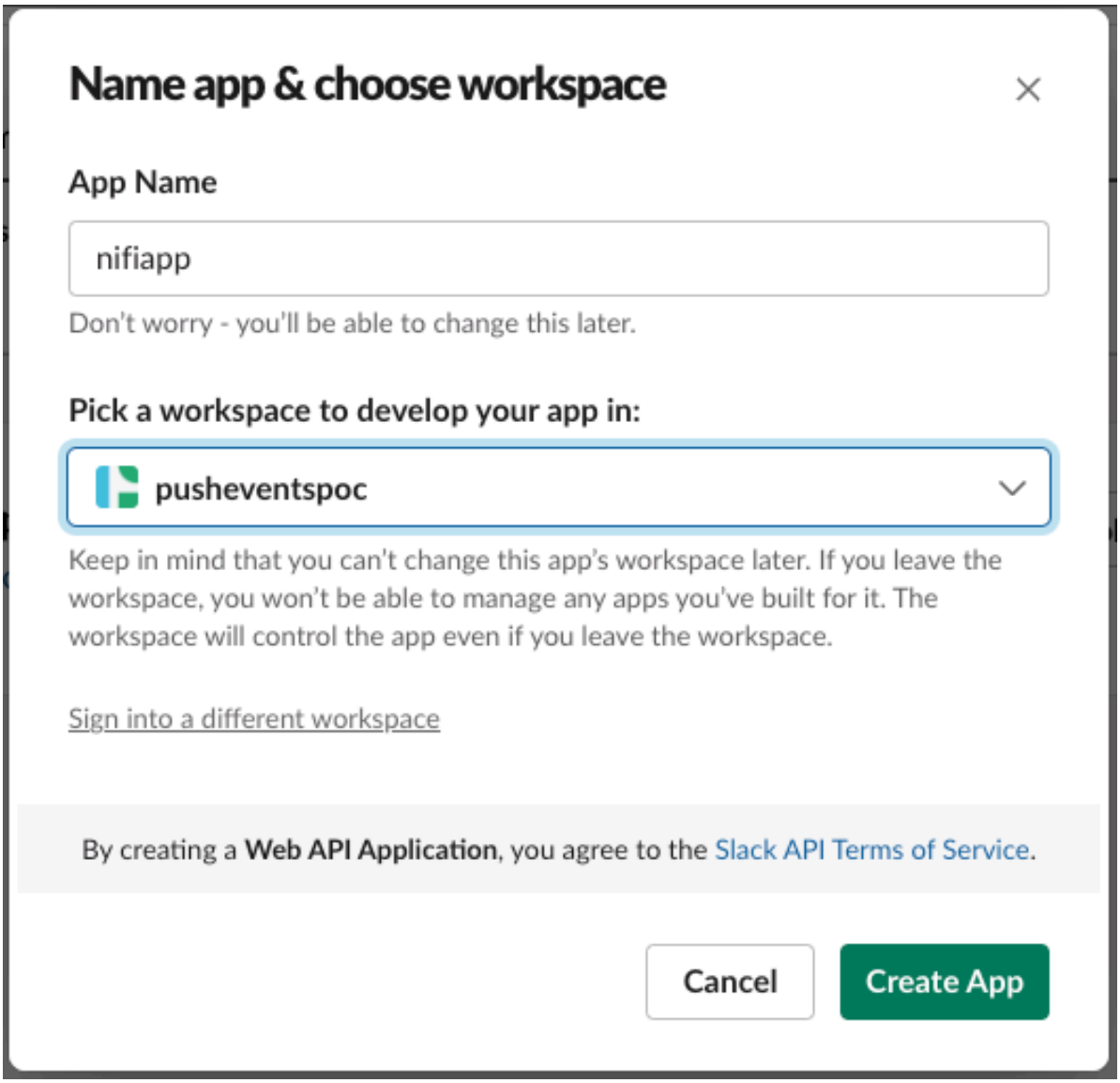

  5. Go to https://api.slack.com/apps/ and select the application you have created.
  6. Select Event Subscription and click on the toggle to enable events.

     After enabling event subscription an input box appears.
  7. Provide a request URL.

     The request URL consists of the Cloudera DataFlow Inbound Connection Endpoint Hostname, the value of the Allowed Paths parameter (if specified), and the Listening Port in the following format https://*[***ENDPOINT*

*HOSTNAME***]:[***LISTENING PORT***][***/ALLOWED PATH***]*. For example, https://cdev-test.inbound.dfx.nnfjwxkq.xcu2-8y8x.dev.cldr.work:9876/events

8. Select the events you want to subscribe to. Only the events configured here will be pushed to Cloudera DataFlow.

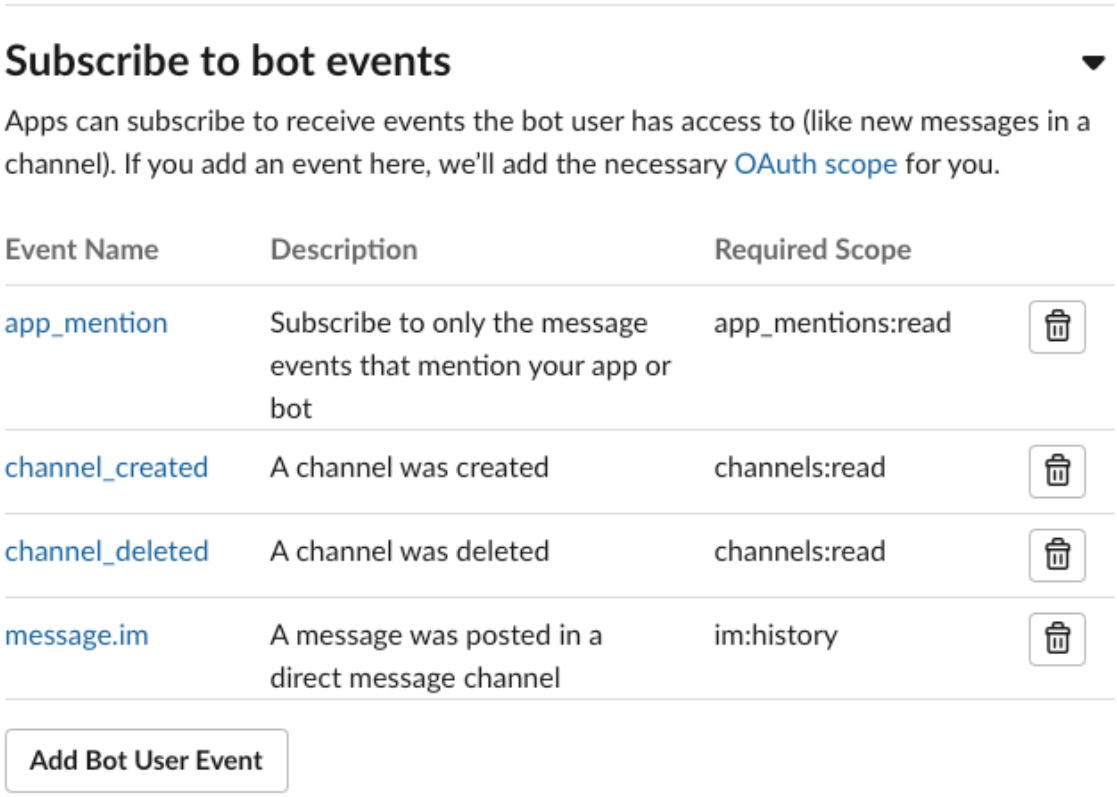

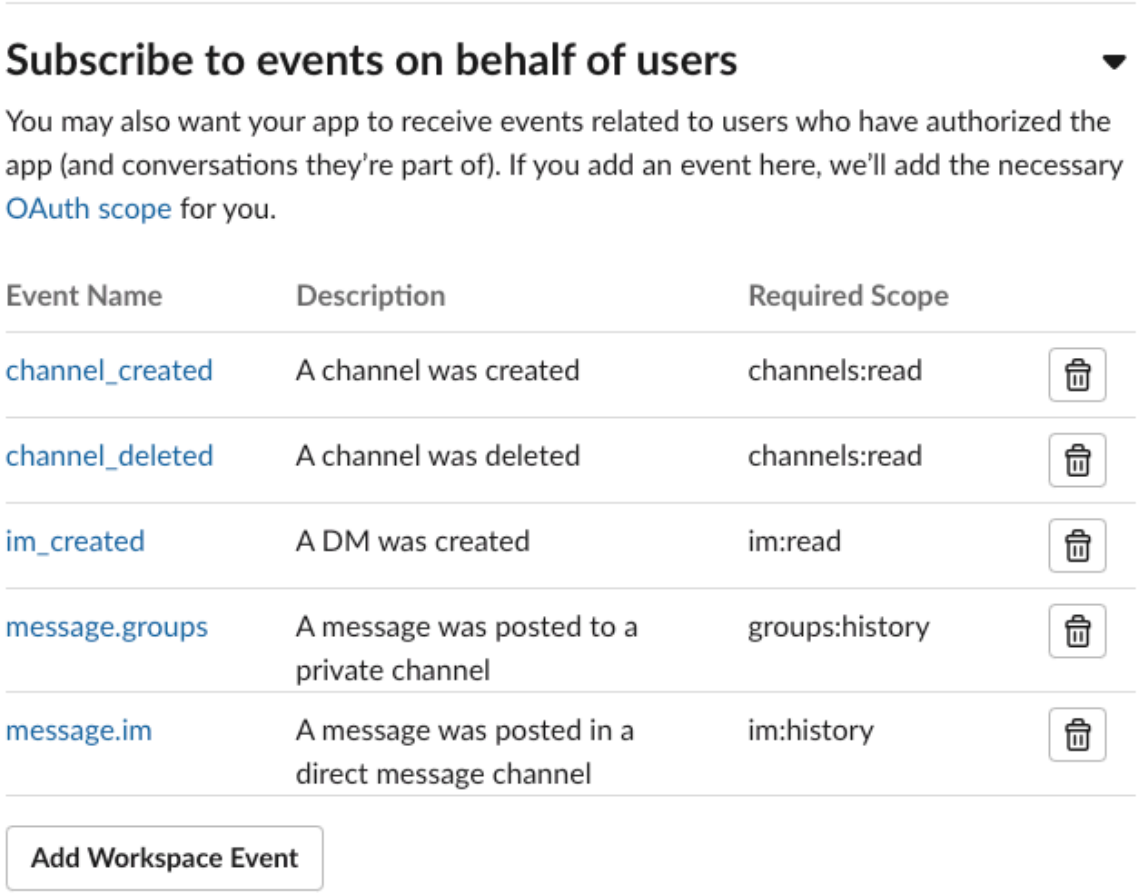

9. Install the Slack App by clicking Install App in the left pane then Install to Workspace in the main pane.

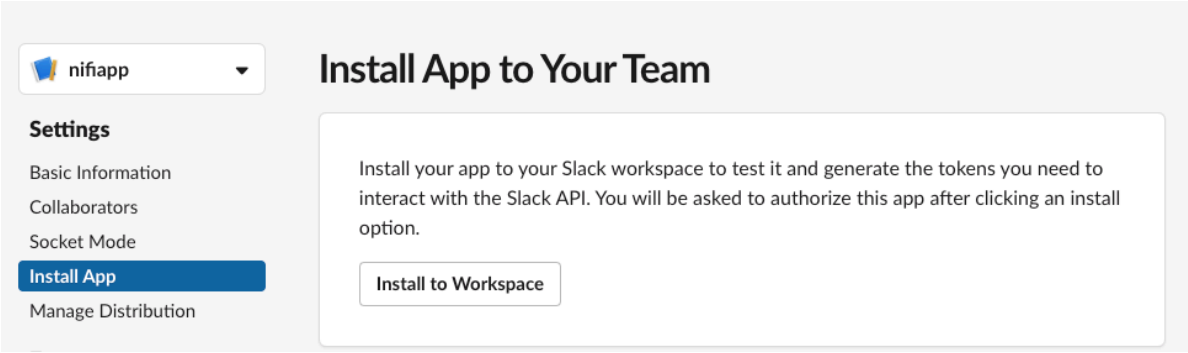

For more information on getting started with Slack Enterprise Grid sandboxes, see the Slack documentation.

For Cloudera DataFlow

- Your environment has public subnets and public connectivity enabled.
- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.
- You have enabled Inbound Connection Support for your Cloudera DataFlow during flow deployment.

  For more information, see Create an Inbound Connection Endpoint during flow deployment .

For your ADLS data ingest target

- You have your ADLS container and path into which you want to ingest data.

- You have performed one of the following to configure access to your ADLS folder:
  - You have configured access to the ADLS folders with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store folders. This allows you to use your Cloudera Public Cloud credentials to access ADLS folders, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the ADLS repository.
    3. Create a policy to govern access to the ADLS container and path used in your ingest workflow. For example: adls-to-adls-avro-ingest

       Tip: The Path field must begin with a forward slash (/).

    4. Add the machine user that you have created for your ingest workflow to ingest the policy you just created.

    For more information, see *Ranger policies for RAZ-enabled Azure environment*.
  - You have configured access to ADLS folders using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to ADLS folders using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding Azure role that provides write access to your folder in ADLS to the Current Mappings section by clicking the blue + sign.

       Note: You can get the Azure Managed Identity Resource ID from the Azure Portal by navigating to Managed Identities Your Managed Identity Properties Resource ID . The selected Azure MSI role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

For your S3 data ingest target

- You have your source S3 path and bucket.

- Perform one of the following to configure access to S3 buckets:
  - You have configured access to S3 buckets with a RAZ enabled environment.

    It is a best practice to enable RAZ to control access to your object store buckets. This allows you to use your Cloudera credentials to access S3 buckets, increases auditability, and makes object store data ingest workflows portable across cloud providers.

    1. Ensure that Fine-grained access control is enabled for your Cloudera DataFlow environment.
    2. From the Ranger UI, navigate to the S3 repository.
    3. Create a policy to govern access to the S3 bucket and path used in your ingest workflow.

       Tip:

       The Path field must begin with a forward slash (/).
    4. Add the machine user that you have created for your ingest workflow to the policy you just created.

    For more information, see *Creating Ranger policy to use in RAZ-enabled AWS environment*.
  - You have configured access to S3 buckets using ID Broker mapping.

    If your environment is not RAZ-enabled, you can configure access to S3 buckets using ID Broker mapping.

    1. Access IDBroker mappings.
       a. To access IDBroker mappings in your environment, click Actions Manage Access .
       b. Choose the IDBroker Mappings tab where you can provide mappings for users or groups and click Edit.
    2. Add your Cloudera Workload User and the corresponding AWS role that provides write access to your folder in your S3 bucket to the Current Mappings section by clicking the blue + sign.

       Note: You can get the AWS IAM role ARN from the Roles Summary page in AWS and can copy it into the IDBroker role field. The selected AWS IAM role must have a trust policy allowing IDBroker to assume this role.
    3. Click Save and Sync.

Related Concepts

List of required configuration parameters for the Slack to S3/ADLS ReadyFlow

List of required configuration parameters for the Slack to S3/ADLS ReadyFlow

When deploying the Slack to S3/ADLS ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: Slack to S3/ADLS ReadyFlow configuration parameters

Parameter name	Description
Allowed Paths	Specify the allowed HTTP paths configured in your Slack App. The default value (/events) allows Slack events.
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to the object stores. Ensure this user has the appropriate access rights to the object store locations in Ranger or IDBroker.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate against the object stores (via IDBroker).
CSV Delimiter	If your desired output data is CSV, specify the delimiter here.
Data Output Format	Specify the desired format for your output data. You can use CSV, JSON or AVRO with this ReadyFlow.

Parameter name	Description
Destination S3 or ADLS Path	Specify the name of the destination S3 or ADLS path you want to write to. Make sure that the path starts with "/".
Destination S3 or ADLS Storage Location	Specify the name of the destination S3 bucket or ADLS container you want to write to. • For S3, enter a value in the form: s3a://*[***DESTINATION S3 BUCKET***]* • For ADLS, enter a value in the form: abfs://*[***DESTINATION ADLS FILE SYSTEM***]@[***DESTINATION ADLS STORAGE ACCOUNT***]*.dfs.core.windows.net
Listening Port	Specify the port to listen on for incoming connections.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

SQL Server CDC to Iceberg

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: SQL Server CDC to Iceberg [Technical Preview]..... 4

Prerequisites..... 4

List of required configuration parameters for the SQL Server CDC to Iceberg [Technical Preview] ReadyFlow.....6

ReadyFlow: SQL Server CDC to Iceberg [Technical Preview]

You can use the SQL Server CDC to Iceberg [Technical Preview] ReadyFlow to retrieve CDC events from a SQL Server source table and stream them to an Iceberg destination table.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from a SQL Server source table to stream the events to an Iceberg destination table. Failed Iceberg write operations are retried automatically to handle transient issues. Define a KPI on the failure_WriteIcebergTable connection to monitor failed write operations.

Note: This ReadyFlow is considered Technical Preview and is not designed for production use.

Note:

This ReadyFlow does not support schema changes or primary key field updates. The destination table must be Iceberg format v2. Make sure to assign the correct permissions for the Iceberg destination table to the specified CDP Workload User.

SQL Server CDC to Iceberg [Technical Preview] ReadyFlow details	
Source	SQL Server Table
Source Format	SQL Server Table
Destination	Iceberg Table
Destination Format	Parquet

Prerequisites

Learn how to collect the information you need to deploy the SQL Server CDC to Iceberg [Technical Preview] ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the SQL Server database server hostname and port.
- You have obtained the SQL Server schema name and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the SQL Server table.
- You have performed the SQL Server setup tasks required to run Debezium.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- In Cloudera Data Warehouse, you have activated the same environment for which Cloudera DataFlow has been enabled. This will create a default database catalog. For more information, see Activating an AWS environment from Cloudera Data Warehouse or Activating Azure environments.
- You have created a Hive Virtual Warehouse referencing the default database catalog. For more information, see Creating your first Virtual Warehouse.
- You have created the Iceberg table that you want to ingest data into, running in your Hive Virtual Warehouse. For more information, see Iceberg table creation from Hive.

Related Concepts

List of required configuration parameters for the SQL Server CDC to Iceberg [Technical Preview] ReadyFlow

List of required configuration parameters for the SQL Server CDC to Iceberg [Technical Preview] ReadyFlow

When deploying the SQL Server CDC to Iceberg [Technical Preview] ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: SQL Server CDC to Iceberg [Technical Preview] ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to Hive.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate to Hive.
CDPEnvironment	The Cloudera Environment configuration resources.
Hive Catalog Namespace	Specify the Hive catalog namespace.
Iceberg Table Name	Specify the Iceberg table name you want to write to. Table must be Iceberg format v2.
Source Database Name	Specify the source database name.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.
Source Database Server Port	Specify the source database server port. The default value is 1433.
Source Database Table Name	Specify the source database table name in the format [schema_name].[table_name]
Source Database User	Specify the source database user.

Related Concepts

Prerequisites

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

SQL Server CDC to Kudu

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ReadyFlow: SQL Server CDC to Kudu.....4

Prerequisites..... 4

List of required configuration parameters for the SQL Server CDC to Kudu ReadyFlow..... 6

ReadyFlow: SQL Server CDC to Kudu

You can use the SQL Server CDC to Kudu ReadyFlow to retrieve CDC events from a SQL Server source table and stream them to a Kudu destination table.

This ReadyFlow uses Debezium to retrieve CDC events (INSERT, UPDATE, DELETE) from a SQL Server source table to stream the events to a Kudu destination table. Failed Kudu write operations are retried automatically to handle transient issues. Define a KPI on the failure_ModifyKuduTable connection to monitor failed write operations.

Note:

This ReadyFlow does not support schema changes or primary key field updates. Make sure to assign the correct permissions for the Kudu destination table to the specified Cloudera Workload User.

SQL Server CDC to Kudu ReadyFlow details	
Source	SQL Server Table
Source Format	SQL Server Table
Destination	Kudu
Destination Format	Kudu Table

Prerequisites

Learn how to collect the information you need to deploy the SQL Server CDC to Kudu ReadyFlow, and meet other prerequisites.

For your data ingest source

Note: Do not change primary key field values in your source table after configuring the ReadyFlow. Doing so will cause the ReadyFlow to reject updates.

Note: You need to take care of field case sensitivity when defining source and destination table structure.

- You have obtained the SQL Server database server hostname and port.
- You have obtained the SQL Server schema name and table name. Take note of the table structure, specifically field case sensitivity.
- You have obtained a username and password to access the SQL Server table.
- You have performed the SQL Server setup tasks required to run Debezium.

For Cloudera DataFlow

- You have enabled Cloudera DataFlow for an environment.

  For information on how to enable Cloudera DataFlow for an environment, see Enabling Cloudera DataFlow for an Environment.
- You have created a Machine User to use as the Cloudera Workload User.

- You have given the Cloudera Workload User the EnvironmentUser role.
  1. From the Management Console, go to the environment for which Cloudera DataFlow is enabled.
  2. From the Actions drop down, click Manage Access.
  3. Identify the user you want to use as a Workload User.

     Note:

     The Cloudera Workload User can be a machine user or your own user name. It is best practice to create a dedicated Machine user for this.
  4. Give that user EnvironmentUser role.
- You have synchronized your user to the Cloudera Public Cloud environment that you enabled for Cloudera DataFlow.

  For information on how to synchronize your user to FreeIPA, see Performing User Sync.
- You have granted your Cloudera user the DFCatalogAdmin and DFFlowAdmin roles to enable your user to add the ReadyFlow to the Catalog and deploy the flow definition.
  1. Give a user permission to add the ReadyFlow to the Catalog.
     a. From the Management Console, click User Management.
     b. Enter the name of the user or group you wish to authorize in the Search field.
     c. Select the user or group from the list that displays.
     d. Click Roles Update Roles .
     e. From Update Roles, select DFCatalogAdmin and click Update.

     Note: If the ReadyFlow is already in the Catalog, then you can give your user just the DFCatalogViewer role.
  2. Give your user or group permission to deploy flow definitions.
     a. From the Management Console, click Environments to display the Environment List page.
     b. Select the environment to which you want your user or group to deploy flow definitions.
     c. Click Actions Manage Access to display the Environment Access page.
     d. Enter the name of your user or group you wish to authorize in the Search field.
     e. Select your user or group and click Update Roles.
     f. Select DFFlowAdmin from the list of roles.
     g. Click Update Roles.
  3. Give your user or group access to the Project where the ReadyFlow will be deployed.
     a. Go to DataFlow Projects .
     b. Select the project where you want to manage access rights and click ⋮ More Manage Access .
  4. Start typing the name of the user or group you want to add and select them from the list.
  5. Select the Resource Roles you want to grant.
  6. Click Update Roles.
  7. Click Synchronize Users.

For your data ingest target

- You have a Real-Time Data Mart cluster running Kudu, Impala, and Hue in the same environment for which Cloudera DataFlow has been enabled.
- You have the Kudu Master hostnames.
  1. From Management Console, click Data Hub Clusters.
  2. Select the Real-Time Data Mart cluster to which you want to ingest data into.
  3. Click the Hardware tab.
  4. Copy the FQDN for each Kudu Master.
- You have the Kudu database name.

- You have created the Kudu table that you want to ingest data into.
  1. Navigate to your Real Time Data Mart cluster and click Hue from the Services pane.
  2. Click the Tables icon on the left pane.
  3. Select the default database, and click + New to create a new table.
  4. In the Type field, select Manually and click Next.
  5. Provide the table Name, Format, Primary keys, and any partitions.
  6. Click Submit. The newly created table displays in the default database Tables pane.
  7. Check the Kudu UI **Tables** tab for the name of the table you created. You will need this table name when you use the Cloudera DataFlow Deployment wizard to deploy the ReadyFlow.
- You have assigned permissions via IDBroker or in Ranger to enable the Cloudera Workload User to access the Kudu table that you want to ingest data into.
  1. From the base cluster on Cloudera Public Cloud, select Ranger.
  2. Select your Real Time Data Mart cluster from the **Kudu** folder.
  3. Click Add New Policy policy.
  4. On the **Create Policy** page, enter the Kudu table name in the topic field.
  5. Add the Cloudera Workload User in the Select User field.
  6. Add the Insert and Select permissions in the Permissions field.
  7. Click Save.

Related Concepts

List of required configuration parameters for the SQL Server CDC to Kudu ReadyFlow

List of required configuration parameters for the SQL Server CDC to Kudu ReadyFlow

When deploying the SQL Server CDC to Kudu ReadyFlow, you have to provide the following parameters. Use the information you collected in *Prerequisites*.

Table 1: SQL Server CDC to Kudu ReadyFlow configuration parameters

Parameter Name	Description
CDP Workload User	Specify the Cloudera machine user or workload username that you want to use to authenticate to Kudu. Ensure this user has the appropriate access rights to the Kudu table.
CDP Workload User Password	Specify the password of the Cloudera machine user or workload user you are using to authenticate to Kudu.
Destination Database Table Name	Specify the destination database table name in the form: [database_name].[table_name]
Kudu Master Hosts	Specify the Kudu Master hostnames in a comma separated list.
Source Database Name	Specify the source database name.
Source Database Password	Specify the source database password.
Source Database Server Host Name	Specify the source database server host name.
Source Database Server Port	Specify the source database server port. The default value is 1433.
Source Database Table Name	Specify the source database table name in the format [schema_name].[table_name]
Source Database User	Specify the source database user.

Related Concepts

List of required configuration parameters for the SQL Server CDC to Kudu ReadyFlow

Related Information

Deploying a ReadyFlow

Cloudera DataFlow

Release Notes

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What's new 5
- December 03, 2024 5
- November 20, 2024 5
- November 15, 2024 5
- Older releases 5
  - November 11, 2024 5
  - October 31, 2024 6
  - June 25, 2024 7
  - June 03, 2024 7
  - May 20, 2024 7
  - January 9, 2024 9
  - December 5, 2023 10
  - November 1, 2023 10
  - October 5, 2023 11
  - August 1, 2023 12
  - July 18, 2023 12
  - June 21, 2023 12
  - May 02, 2023 13
  - April 27, 2023 13
  - April 05, 2023 14
  - March 14, 2023 14
  - February 13, 2023 15
  - January 25, 2023 16
  - January 20, 2023 16
  - January 03, 2023 17
  - December 08, 2022 17
  - October 21, 2022 18
  - October 10, 2022 18
  - September 29, 2022 19
  - September 28, 2022 19
  - September 08, 2022 19
  - August 16, 2022 20
  - June 28, 2022 21
  - May 17, 2022 21
  - April 06, 2022 22
  - March 23, 2022 23
  - March 8, 2022 23
  - February 15, 2022 23
  - January 27, 2022 24
  - December 21, 2021 24
  - December 17, 2021 25
  - December 1, 2021 25
  - October 13, 2021 26
  - September 29, 2021 27
  - August 24, 2021 27
  - August 16, 2021 28

Supported component versions 28

Known issues and limitations.....30

What's new

Learn about major features, updates, and fixes for Cloudera DataFlow service.

December 03, 2024

This hotfix release (2.9.0-h4-b3) of Cloudera DataFlow resolves an issue where environments with a high number of deployments could unpredictably fail to upgrade during a standard upgrade process.

Fixed issues

Fixed an issue, identified during testing, where environments with 50+ flow deployments sometimes encountered incomplete upgrade processes. This release ensures that upgrades can complete successfully, even for environments with a large number of deployments.

November 20, 2024

This release (2.9.0-h3-b1) of Cloudera DataFlow fixes a control plane issue that is not visible to Cloudera customers.

- Environments running Cloudera DataFlow version 2.9.0-h2-b1 do not require this upgrade and will not have the upgrade banner.
- Environments running Cloudera DataFlow version 2.9.0-h1-b1 or lower, will need to upgrade to 2.9.0-h3-b1.

Fixed issues

See 2.9.0 hotfix 2.

November 15, 2024

This release (2.9.0-h2-b1) of Cloudera DataFlow fixes an issue preventing the running of test sessions, and an upgrade issue affecting single to Flexible Server migration in Azure environments.

Fixed issues

- Fixed an issue that was causing test sessions in Flow Designer to either take a long time to start, or not start at all.
- Fixed an issue that was causing the 2.9.0 upgrade to fail for some customers running in Azure Environments.

Older releases

Overview of new features, enhancements, and changed behavior introduced in earlier releases of Cloudera DataFlow.

November 11, 2024

This release (2.9.0-h1-b1) of Cloudera DataFlow fixes issues where the version 2.9.0 control plane had problems interacting with older workloads.

Fixed issues

- Fixed an issue where the Resources view handled workloads with version lower than 2.9.0 incorrectly.
- Fixed an issue where Flow Designer became inaccessible in workloads with version lower than 2.9.0.
- Fixed an issue where publishing a flow to the Catalog in a workload with version lower than 2.9.0 failed.

October 31, 2024

This release (2.9.0-b383) of Cloudera DataFlow increases developer productivity through the introduction of Parameter Groups which can be shared between flow drafts. Developers can now also create NiFi 2.0 flows in the Designer leveraging new Cloudera exclusive processors for building RAG data pipelines. Deployments can now be configured with a Prometheus endpoint that allows scraping Apache NiFi metrics. Cloudera DataFlow's service and deployment events and alerts now support in-app and email notifications.

What's new

Latest NiFi version

Flow Deployments and Test Sessions now support the latest Apache NiFi 1.27 release.

Build NiFi 2.0 flows in Flow Designer [Technical Preview]

You can now select NiFi 2.0 when creating drafts and start test sessions including the ability to configure your test session to use your custom Python based processors.

New Cloudera-exclusive AI processors for NiFi 2 [Technical Preview]

You can now implement RAG pipelines by using new processors to parse, chunk and vectorize data, bringing context to their LLMs. The following processors are now available with NiFi 2:

- PartitionPdf, PartitionHtml, PartitionText, PartitionDocx, PartitionCsv
- ChunkData
- EmbedData
- InsertToMilvus, LexicalQueryMilvus, VectorQueryMilvus
- PutChroma, QueryChroma
- PutOpenSearchVector, QueryOpenSearchVector

Bedrock Parameter Groups

You can now centrally define and manage parameter groups in a workspace and re-use them for multiple drafts, eliminating tedious copy-and-pasting of parameters and their values.

New Resources page

Users can now easily view and manage all their workspace resources like deployments, drafts, parameter groups, inbound connections, custom NAR/Python configurations in a single place.

Notifications via App and Email for Cloudera DataFlow service and deployment events

You can now receive real-time notifications for all events related to a Cloudera DataFlow Service and its deployments through the Cloudera Management Console, under the **Notifications** tab, and through email.

For more information, see Setting up service and deployment notifications

NiFi metrics can now be exposed via a Prometheus endpoint

You can now configure deployments to expose NiFi metrics through a Prometheus endpoint. Once set up, you can configure your Prometheus instances to scrape these endpoints, consume relevant metrics and build custom dashboards.

For more information, see Configuring access for NiFi metrics scraping.

New ReadyFlows

- ADLS to Pinecone
- S3 to Pinecone
- ADLS to Milvus
- S3 to Milvus
- RAG Query Milvus

New Kubernetes version support

Cloudera DataFlow now supports EKS/AKS 1.29

Changes and improvements

- As part of the upgrade process to Cloudera DataFlow 2.9.0, the Azure Postgres database is migrated from a single server to a flexible server deployment.
- Improved asset handling for deployments makes deployment creation more robust in cases where many deployments are created at the same time.
- Kubernetes scale up events could result in the Cloudera DataFlow application container being rescheduled causing Cloudera DataFlow to become unavailable. Additional restrictions for rescheduling the Cloudera DataFlow application were added to avoid downtime.
- Dependencies have been updated to Java 21, Spring 6 and Spring Boot 3.

Fixed issues

- NiFi cluster failed to auto scale with a UDP inbound connection configured
- NiFi node failed to start up due to custom Kubernetes cluster domain name
- MiniFi logging failed to clean up a full content volume
- Vault failed to start up due to insufficient wait in its postStart script
- Auto scaling driven by flow metrics did not kick in

June 25, 2024

This release (2.8.0-h2-b2) of Cloudera DataFlow introduces bug fixes for RAZ-enabled CDP environments running Apache NiFi 1.25.

Fixed issues

- CDPDFX-8844 - RAZ is broken on AWS with NiFi 1.25 in Cloudera DataFlow

June 03, 2024

This release (2.8.0-h1-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces bug fixes which fix issues with upgrades and changing flow versions.

Fixed issues

- CDPDFX-8851 - fixes an issue where a CDF upgrade on Azure could fail due to MiNiFi pods getting stuck.
- CDPDFX-8826 - fixes an issue where a CDF upgrade could fail due to an existing deployment being in ‘FAILED_TO_IMPORT’ state.
- CDPDFX-8803 - fixes an issue where deployments failed due to too long HTTP timeouts during asset loader API calls.
- CDPDFX-8828 - fixes an issue where certain configuration files represented by the ‘CDP Environment’ parameter were uploaded multiple times when change flow version was used.
- CDPDFX-8760 - fixes an issue where CDF was sending pod deployment status even when the pod count was zero.

May 20, 2024

This release (2.8.0-b274) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces the ability to change the flow definition version of running deployments, a new Overview page with tutorials and shortcuts, the ability to create NiFi 2.0 deployments (Technical Preview), filtering ReadyFlows by different categories, new GenAI ReadyFlows, and supports new Kubernetes versions.

What's new

Latest NiFi version

Flow Deployments and Test Sessions now support the latest Apache NiFi 1.25 release.

NiFi 2.0 Tech Preview

You can now select NiFi 2.0 (based on upstream Apache NiFi M2 release with critical fixes from M3) when creating deployments including the ability to configure your deployment to use your custom Python based processors.

Change flow definition version of existing deployments

You can now change the version of your flow definition for existing deployments. This eliminates the need to recreate deployments whenever a new version of your flow is available. Depending on your needs, you can choose from three different strategies when changing the flow definition version. Learn more about changing flow definition versions.

New Overview Page

When navigating to CDF, users now start on the new **Overview** page. The **Overview** page helps new users getting started with guides and documentation, informs administrators about recent releases and offers shortcuts to power users.

Custom NiFi Node sizing (requires entitlement)

When creating a deployment, CDF now supports specifying custom core/memory settings. This feature requires an entitlement. Reach out to your Cloudera team to request access.

ReadyFlow Gallery filtering

The ReadyFlow Gallery now supports filtering available ReadyFlows by four different categories: Use case category, Source, Destination and compatible NiFi version.

New ReadyFlows

- DB2 CDC to Iceberg (Technical Preview)
- DB2 CDC to Kudu
- MySQL CDC to Iceberg (Technical Preview)
- Oracle CDC to Iceberg (Technical Preview)
- PostgreSQL to Iceberg (Technical Preview)
- Slack to Pinecone (NiFi 2.0)
- SQL Server CDC to Iceberg (Technical Preview)

Support for new Kubernetes versions

CDF now supports Kubernetes 1.28 on EKS and AKS.

Include JVM Heap and Thread Dump in UDX

Users can trigger a Diagnostic Bundle collection to include heap and thread dump of all the NiFi nodes of a flow deployment.

Changes and improvements

- The **Dashboard** page has been renamed to **Deployments** and continues to be the single pane of glass to monitor all existing deployments.
- Diagnostic Bundle collection now includes additional information and allows collection of heap dumps for faster troubleshooting.
- **Flow Details** of the **Catalog** page now support text search and filtering by tags.
- When enabling CDF on Azure, CDF now provisions an Azure Database for PostgreSQL instead of a single server. This ensures supportability as single servers get phased out by Microsoft.
- CDF on AWS now supports RDS certificate rotation ensuring compatibility with planned certification rotation changes from AWS.

Removed NiFi components

In this release of CDF-PC, a number of deprecated NiFi processors and controller services have been removed from the porduct. For a list of removed components and suggested replacements, see Removed processors and Removed controller services, respectively.

Fixed issues

- CDPDFX-8257 - Updated sensitive parameter values are now propagated to NiFi
- CDPDFX-8513 - Hello World ReadyFlow corrected
- CDPDFX-8605 - Alert mapping for PROMETHEUS_MEMORY_WARNING_ALERT added so deployment termination won't get stuck
- CDPDFX-8624 - Reassign Workload Resource now updates project id in global deployment table for test sessions so after reassign projects can be deleted
- CDPDFX-8268 - False alerts during vertical auto scaling of Prometheus instances

Technical Service Bulletins

- • TSB 2024-782: Cloudera DataFlow for Public Cloud on AWS requires upgrade before the Root Certificate expires

January 9, 2024

This release (2.7.0-b190) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces new Change Data Capture processors, flow version tagging, deployment configuration reuse, NiFi bulletin monitoring and supports new Kubernetes versions.

What's new

Latest NiFi version

Flow Deployments and Test Sessions now support the latest Apache NiFi 1.24 release.

New Debezium CDC processors

You can now build and deploy flows with Debezium based CDC processors for MySQL, Postgres, Oracle, SQLServer and DB2 databases.

Deployment alerts for NiFi bulletins

CDF now detects bulletin error messages in NiFi flow deployments and displays them as a warning in the dashboard.

Flow Definition versions can now be tagged

You can now use tags to identify versions easier. Common use cases for tags are applying a custom versioning scheme, or labeling flow versions as 'ready for deployment'. The CLI introduces new commands to search flow definitions for tags and streamlines the CI/CD process.

Deployment configurations can now be exported and reused

Customers can now export a configuration archive of existing deployments and reuse them when creating new deployments. This speeds up redeploying new versions of the same flow through the Deployment Wizard by filling in parameter values, KPI configurations etc. from the deployment configuration.

Support for new Kubernetes versions

AKS 1.27 / EKS 1.27

New ReadyFlows

- MySQL CDC to Kudu (Tech Preview)
- Postgres CDC to Kudu (Tech Preview)
- Oracle CDC to Kudu (Tech Preview)
- SQL Server CDC to Kudu (Tech Preview)
- ADLS to Databricks
- S3 to Databricks
- HuggingFace dataset to S3/ADLS
- S3 to IBM watsonx

Changes and improvements

- The HelloWorld ReadyFlow has been updated. You need to deploy Version 2 to run it with NiFi 1.24.0 or higher.
- Underlying AKS creation error during a CDF enablement, if available, is now reported together with the enable failed status event instead of showing up a few minutes later.
- You can now download Client Certificate and Private Key for an Inbound Connection in a single action.

Fixed issues

- CDPDFX-8057: Fixed an issue where TLS certificate renewal was not correctly handled by VPA deployment.
- CDPDFX-7969: Fixed an issue where upgrade fails if there is a deployment using an unsupported NiFi version.
- CDPDFX-8162: Fixed an issue where a deployment creation can be initiated with an existing inbound connection endpoint.
- CDPDFX-8102: Fixed an issue where suspended deployments are reported as not healthy after upgrade.
- CDPDFX-8202 and CDPDFX-8203: CLI improvements when listing projects and deployment archives.
- CDPDFX-8057: Introduced a common component to address TLS certificate reload after periodic renewal.
- CDPDFX-6745: Re-instated bounded automatic helm operation retry during upgrade.
- CDPDFX-8096: Rolling upgrade for NiFi cluster when there is no core NiFi version change.

December 5, 2023

This release (2.6.1-h1-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces a fix that prevents potential NullPointerExceptions (NPEs).

Fixed issues

- CDPDFX-8136 Fixed an issue where the Kubernetes PATCH API response potentially caused an NPE during upgrades and Environment configuration updates in private clusters.

November 1, 2023

This release (2.6.1-b92) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces fine-grained access controls over resources within an environment, support for creating and managing reporting tasks through the CLI, besides other improvements and fixes.

Note: This release of CDF supports deployments running NiFi 1.18.0.2.3.7.0-100 or newer. If your DataFlow service has older NiFi versions, you can perform a Change NiFi Version to bring each into compliance or select to update to the latest as part of DataFlow Upgrade.

What's new

- Support for granular access controls over resources have been introduced to the DF Service. On the new Projects page users can define Projects that limit the visibility of Flow Drafts, Deployments, Inbound Connections, and Custom NAR Configurations within an Environment. Dashboard, Flow Designs and Workspace Resources pages now have new filters and controls to organize resources into Projects.
- Added support for creating and deleting reporting tasks in a CDF deployment through the CLI. Listing reporting tasks is also available in the UI under the NiFi Configuration tab in deployment manager.

Changes and improvements

- Third party and base images updated to address CVEs.
- Improvements for larger scale CDF clusters of up to 50 nodes, including how Prometheus instances are monitored and vertical scaling cluster resources in terms of CPU.
- View NiFi UI is a valid action when the deployment is in a failed to upgrade state.
- Port 80 is removed from the security group for the load balancer that is created by DFX on AWS. It redirected requests to port 443, but port 80 was not necessary to be open.

October 5, 2023

This release (2.6.0-b311) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces NiFi 1.23, vertical scaling, several cost optimization features and the ability to collect Diagnostic Bundles through the UI.

Note: This release of CDF supports deployments running NiFi 1.18.0.2.3.7.0-100 or newer. If your DataFlow service has older NiFi versions, you can perform a Change NiFi Version to bring each into compliance or select to update to the latest as part of DataFlow Upgrade.

What's new

- Added support for downloading NiFi application logs directly from CDF for deployments and test sessions.
- Added support for suspend and resume CDF Deployments. Suspending a Deployment temporarily terminates the NiFi cluster resources and billing activities while maintaining flow persistence.
- Added support for resizing an existing CDF Deployment, which allows increasing and decreasing the memory and CPU allotted to an existing NiFi cluster
- Added support for selecting the Node Storage Profile when creating a CDF Deployment. This allows the use of economical or high performance persistent storage for the NiFi cluster.
- New flow deployments on AWS will now use GP3 volumes instead of GP2 providing a better cost/performance ratio.
- Added active monitoring of NiFi cluster health for CDF Deployments which raises and resolves an alert automatically when a NiFi cluster encounters and recovers from issues impacting its running nodes.
- Added ability to create sensitive dynamic properties in Flow Designer.
- Diagnostic Bundles can now be requested via the Unified Diagnostics UI in the Cloudera Manager. Bundles will be uploaded directly to the selected support case.
- Added Workspace Resources page to workload instances. This provides authorized users with a view of deployments, draft flows, inbound connections, and custom NAR configurations in the workspace.
- Introduced new deployment alerts for Kubernetes component failures like NiFi pods or supporting statefulsets.

Changes and improvements

- Support for updated Kubernetes server versions: AKS 1.26 / EKS 1.25.
- Improved role assignment processing in the DataFlow workload application so role updates are automatically updated within 5 minutes by regular queries to FreeIPA. The user no longer has to log out and log back in to refresh their roles.
- Overhauled NiFi node offloading behavior to be more robust and fault resistant in a variety of cases, including when offloading is disrupted by Kubernetes events.
- Completed significant reliability improvements for larger scale CDF clusters up to 50 nodes, including a variety of efficiency refinements and support for vertical scaling for essential cluster resources.
- Renamed existing Suspend Flow action to Stop Flow. Renamed the resulting deployment state Suspended to Flow Stopped. A CDF Deployment in Flow Stopped state does not remove NiFi cluster resources and the deployment remains billable.
- Released the latest version of NiFi 1.23 which includes the following improvements and CDF integrations:
  - Kubernetes Leader Election and State Management - eliminates the need for a ZooKeeper pod in each CDF Deployment cluster.
- Automatic restart the deployment under the hood if NiFi properties are changed.
- Vertical auto scaling for Prometheus to automatically adjust memory to support a larger number of deployments and metrics.
- Improved stability of helm upgrade during CDF upgrade process.
- Retaining more NiFi provenance data given the current storage capacity allocated for that purpose.

Fixed issues

- CDPDFX-7455: Fixed issue where underlying failure to create AKS cluster on CDF is masked by a 404 error. The actual Azure activity log failure is now raised in a status update event soon after the 404 failure occurs.

- CDPDFX-7540: Fixed issue preventing Kafka to Kudu ReadyFlow from being opened in Flow Designer.
- CDPDFX-7828: Improved sensitivity of CPU based autoscaling behavior when flow metrics autoscaling feature is enabled.
- CDPDFX-7400: Fixed issue rendering an apostrophe in Flow Designer Data Viewer.
- CDPDFX-7399: Fixed issue with MiNiFi pod failing to recover after a CDF upgrade encounters an issue and triggers rollback.
- CDPDFX-7815: Fixed issue with validation of Custom NAR Configuration in RAZ enabled environments.
- CDPDFX-7399: Fixed MiNiFi logging data migration.
- CDPDFX-7704: Fixed racing condition among containers in DFX-Local pod.
- CDPDFX-7737: Fixed issue where UDX collection is not associated with case properly.

August 1, 2023

This release (2.5.0-h2-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud resolves an issue with NiFi cluster autoscaling behavior that can cause unnecessary flow disruptions.

Fixed issues

- CDPDFX-7567 - Fixed unnecessary NiFi cluster pod restarts triggered by autoscaling operations.
  - This issue impacted only those running DataFlow 2.5.0-h2-b1 and deployments configured with autoscaling enabled.
  - Any scale-up or scale-down operation would trigger unnecessary restarts of existing pods resulting in temporary flow disruptions.
  - Upgrading to this new DataFlow version automatically resolves the issue.

Technical Service Bulletins

- TSB 2023-688: Incorrect Java heap memory allocation when creating Small, Medium or Large NiFi flow deployments
- TSB 2023-690: Auto-Scaling operations triggering NiFi pod restarts causing temporary flow disruptions

July 18, 2023

This release (2.5.0-h1-b6) of Cloudera DataFlow (CDF) on CDP Public Cloud resolves a critical bug in JVM heap allocation for newly created NiFi deployments besides other minor bug fixes.

Fixed issues

- CDPDFX-7507 - NiFi deployments assigned incorrect JVM heap settings:
  - This issue only impacts deployments created after upgrading to CDF release 2.5.0-b210.
  - SMALL, MEDIUM and LARGE deployments appear to be utilizing the appropriate amount of heap memory for their allocated cluster resources, but they are only allocated 2GB of heap memory. This results in a possible limitation of flow functionality.
  - Upgrading to CDF release 2.5.0-h1-b6 automatically resolves this issue.
- CDPDFX-7461 - introduces a minor fix to show the proper axis label on metric charts.

Technical Service Bulletins

- TSB 2023-688: Incorrect Java heap memory allocation when creating Small, Medium or Large NiFi flow deployments

June 21, 2023

This release (2.5.0-b210) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces NiFi 1.21, flow metrics driven auto-scaling, advanced UIs for JoltTransformJSON and UpdateAttribute processors, bulk actions in the Flow Designer, UDR support on Azure and makes in-place upgrades generally available to all customers.

New features and changes

- Flow Deployments and Test Sessions now support the latest Apache NiFi 1.21 release.
- In-place upgrades are now generally available to all customers. For more information, see Service upgrade.
- Flow Deployments now support Flow Metrics Scaling in addition to CPU utilization based auto-scaling. For more information, see Auto-scaling flow deployments.
- When selecting Private Cluster during enablement on Azure, CDF no longer provisions a public load balancer and now supports User Defined Routes (UDR).
- Users can now provide their own values for the Kubernetes Pod CIDR Range and Kubernetes Service CIDR Range during enablement.
- CDF now inherits load balancer configuration including available subnets from its associated CDP environment during the enablement process.

Flow Designer

- The Flow Designer now supports multi-selection on the canvas and bulk actions for Start, Stop, Enable, Disable, Move, Change parent group, Copy/Paste, and Delete.
- The Flow Designer now supports the advanced configuration UI for UpdateAttribute.
- The Flow Designer now supports the advanced configuration UI for JoltTransformJson.
- The Flow Designer now supports Birdseye and Zoom controls.
- The Flow Designer now supports Processor Diagnostics with an active Test Session.

ReadyFlows

The following new ReadyFlows have been added to the ReadyFlow Gallery:

- CDW Ingest
- CDP Kafka to Snowflake
- Slack to S3
- Updated Confluent Cloud to Snowflake using new Snowpipe processors

Fixed issues

- CDPDFX-7231 - Fixed an issue where asset uploads were not supported in the Deployment Wizard following publishing from the Flow Designer.
- CDPDFX-7355 - Fixed issue where CDF Upgrade can lead to a stuck active alert indicating metrics are unavailable if it was present prior to the upgrade attempt.
- CDPDFX-7042 - Fixed lack of –no-paginate support on some df list calls in CDP CLI.
- CDPDFX-7060 - Fixed CDP CLI handling of large asset uploads as part of create deployment.

May 02, 2023

This release (2.4.1-h1-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces various improvements and bug fixes for Flow Designer and DataFlow enablement.

Fixed issues

- CDPDFX-7144 and CDPDFX-7116 - Addressed performance issues when determining referencing components of Parameters and Controller Services.
- CDPDFX-7150, CDPDFX-7148, CDPDFX-7151, and CDPDFX-7153 - Made the loading of the Flow in the Flow Designer more efficient by eliminating unnecessary details in the response.
- CDPDFX-7147 and CDPDFX-7167 - Fixed skeleton loader in the Flow Listing.

April 27, 2023

This release (2.4.1-b22) of Cloudera DataFlow (CDF) on CDP Public Cloud supports new NiFi and Kubernetes versions, includes usability improvements for Flow Designer, resolves an issue with deployment ZooKeeper persistence and introduces various bug fixes across the platform.

New features and changes

- CDF now creates Kubernetes clusters with version 1.24 in AWS EKS and 1.25 in Azure AKS.
- The PutIceberg processor is now considered GA (requires selecting NiFi version 1.20.0.2.3.8.2-2).
- Improves event age off logic for environments and deployments that produce a large number of events.
- In the Flow Designer, when deleting a Parameter referencing components are populated in the affected components dialog.
- In the Flow Designer, the Inbound Connection Details button visibility was fixed when a Test Session successfully starts.
- After upgrading CDF, aligned the CFM version of any existing Test Session configurations with the minimum CFM version supported by the Flow Designer.

Fixed issues

- CDPDFX-7081 - Fixed DataFlow Update sometimes timing out when a new AMI is available and AWS performs rolling upgrade of cluster nodes.
- CDPDFX-7018 - Fixed missing persistent volume support for ZooKeeper in deployments.
- CDPDFX-7095 - Fixed missing vault missing Flow Designer policies after CDF upgrade (Relevant only for environments that previously ran with CDF 2.1.x).

April 05, 2023

This release (2.4.0-h1-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces various improvements and bug fixes for Flow Designer and DataFlow enablement.

Fixed issues

- CDPDFX-6970 - Fixed an issue where using the Convert to Parameter feature in Flow Designer cleared all sensitive parameter values.
- CDPDFX-7068 - Added missing support for skipPreflightChecks to Enable and Update DataFlow CLI calls
- CDPDFX-6991 - Fixed an issue with Enable DataFlow for Azure where the cloud provider sometimes reflects clusters in an unexpected UPDATING state.
- CDPDFX-7057 - Fixed Enable DataFlow request for AWS sometimes timing out during initial validation phase due to cloud provider API latency with resources in some regions.
- CDPDFX-6534 - Fixed concurrency issues with uploads of large assets via CLI. Assets of up to 150 MB are now supported.
- CDPDFX-7052 - Fixed an issue with validation of Flow Designer parameters where validation was too strict and could prevent certain values from being entered. The validation is now consistent with parameter values from the Deployment Wizard.

March 14, 2023

This release (2.4.0--b317) of Cloudera DataFlow (CDF) on CDP Public Cloud makes the Flow Designer generally available to all CDP Public Cloud customers, introduces a streamlined way to collect log files, supports new NiFi and Kubernetes versions and introduces several usability improvements when creating and managing flow deployments through the UI.

New features and changes

- The ability to design flows with the new Flow Designer is now generally available to all CDP Public Cloud customers.
- The CDP cli now supports commands to generate diagnostic bundles for CDF for troubleshooting.
- Flow Deployments and Test Sessions now support the latest Apache NiFi 1.20 release.
- When creating a deployment through the deployment wizard, users can now filter for sensitive or empty parameters.
- CDF now creates Kubernetes clusters with version 1.23 in AWS EKS and 1.24 in Azure AKS.

- Upgrading and restarting deployments now triggers a NiFi Health Alert if the NiFi cluster does not return to a healthy state. This can be cleared by performing a Restart Deployment once the underlying issue has been fixed.
- Restart Deployment now supports force suspending the NiFi Flow in case the running flow itself is impairing the health of the NiFi cluster.
- Enable DataFlow action now performs validation steps immediately and fails fast if there are issues with the request.
- Some enablement failure messages are improved when the failure is due to common issues surrounding volume provisioning.
- The following ReadyFlows have been added to the ReadyFlow Gallery:
  - HubSpot to S3/ADLS
  - JDBC to JDBC
  - Kafka to Apache Iceberg
  - Shopify to S3/ADLS

Fixed issues

- CDPDFX-6416 - Reduced default tags applied to cloud resources to avoid tag limit concerns
- CDPDFX-6740 - Fixed private cluster support for AKS
- CDPDFX-6542 - Fixed tags not being applied to Load Balancer created for DataFlow ingress
- CDPDFX-6872 - Fixed issue with control plane missing alert resolutions
- CDPDFX-6591 - DFX Local stability improvements
- Various upgrade reliability improvements

February 13, 2023

This release (2.3.0-h3-b4) of Cloudera DataFlow (CDF) on CDP Public Cloud addresses various bug fixes and improvements in Flow Designer, private cluster support, upgrades and general stability.

Fixed issues

Stability Improvements

- CDPDFX-6591 - Added liveness probe to dfx-local to influence a restart if the container becomes unhealthy.
- CDPDFX-6701 - Fixed a resource leak that would eventually lead to degraded behavior of the application like interruption of workload communication (through heartbeats) to the control plane or the Flow Designer not displaying feedback for running test sessions and preventing processors from being started.

Flow Designer

- CDPDFX-6641 - Updated the redis client in the workload to use a connection pool that was previously aggressively allocating new threads.
- CDPDFX-6635 - Fixed an issue with Flow Designer UI that would continually attempt to reconnect/resubscribe even when their authentication expires.
- CDPDFX-6636 - Fixed an issue with Flow Designer test session event blocking due to infinite event retry.
- CDPDFX-6632 - Ensure additional access rights are enforced when using Flow Designer.
- CDPDFX-6695 - Fixed issues in the Flow Designer that prevented applying Parameter changes when an asset was removed.
- CDPDFX-6695 - Fixed an issue in the Flow Designer where editing a parameter with multiple assets does not trigger a message to apply changes.

Private Cluster Support

- CDPDFX-6740 - Fixed an issue causing AKS private clusters to fail when waiting for load balancer IP via cluster proxy service.

Other Issues

- CDPDFX-6731 - Fixed an issue with CDF 2.1.0-b123 or older not being able to perform workload access certificate renewal.
- CDPDFX-6634 - Fixed issue that would result in the user being redirected to the wrong URL following successful authentication on the workload.
- CDPDFX-6655 - Fixed issue that would load the app switcher in the sidebar in the Deployment Manager and Flow Designer with the incorrect options.
- CDPDFX-6567 - Fixed an issue with logging in AWS AP region 3 (Jakarta).
- CDPDFX-6693 - Fixed navigation when using the Home button from the Manage Deployment page.
- CDPDFX-6713 - Fixed DataFlow configuration update failures when attempting to add new authorized IP ranges without changing minimum or maximum cluster capacity.

Technical Service Bulletins

- TSB 2023-648: ApacheNiFi pods not properly configured with fully qualified domain name causing communication errors
- TSB 2023-649: File watcher container in MiNifi logging pod causing process ID leaks

Behavioral changes

If you defined a custom NAR storage location during flow deployment in earlier releases, the setting got stored as the default value for that environment. CDF 2.3.0-h3 changes this behavior: custom NAR configuration does not get stored for the environment any more. This means every time you deploy a flow that requires custom NARs, you need to configure the access credentials and the storage location.

January 25, 2023

This release (2.3.0-h2-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces fixes and improvements for etrics collection, enablements, upgrades and general stability.

Fixed issues

- CDPDFX-6458 - Fixed issue where NiFi flow deployents failed at host name verification due to a startup script returning the wrong host name.
- CDPDFX-6671 - Reduced metric collection sample sizes in Cluster-level Prometheus, resulting in lower memory footprint and faster query times without regression or reduction of performance for existing environment metrics.
- CDPDFX-6677 - Fixed issue where the import of very large flows failed with timeout.
- CDPDFX-6683 - Fixed issue where a TimeoutException was thrown without its cause being logged.

January 20, 2023

This release (2.3.0-h1-b8) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces fixes and improvements for Flow Designer, enablements, upgrades and general stability.

Fixed issues

- CDPDFX-6593 - Fixed issue where test session reports the wrong state after being started and stopped.
- CDPDFX-6582 - Fixed issue where processor property configuration does not support special characters.
- CDPDFX-6596 - Fixed errors which occurred in certain interdependent controller services configurations.
- CDPDFX-6594 - Fixed issue where suspend updates flag is not cleared after an enable attempt fails, but retry enable succeeds. This results in newly created deployments not displaying properly in the deployment dashboard because heartbeats are being ignored.
- CDPDFX-6578 - Fixed upgrade issue resulting in larger deployment sizes having insufficient java heap memory allocated to NiFi pods, resulting in Out of Memory issues.
- CDPDFX-6561 - Fixed issue preventing retry upgrade when previous upgrade attempt failed after kubernetes upgrade to 1.23 but before liftie service finished upgrading deployments..
- CDPDFX-6479 - Fixed redis pods being unable to recover after non-graceful shutdown

- CDPDFX-6512 - Improved minifi logging stability by defining memory needs for toybox container

January 03, 2023

This release (2.3.0-h1-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces a fix for loading custom NARs.

Fixed issues

- NIFI-10981 - Fixed an issue where NarAutoLoader started before the provider retrieved all NARs.

December 08, 2022

This release (2.3.0-b347) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces the technical preview of Flow Designer, adds several new ReadyFlows, improves upgrade reliability and fixes issues with stability for clusters with high utilization.

New features and changes

- Flow Designer [Technical Preview]

  Developers can now build new data flows from scratch using the integrated Designer. Flow Drafts can be tested using Test Sessions before they are published to the Catalog. Once they are in the catalog, users can create Flow Deployments using the Deployment Wizard or CLI.
- Added support for certificate renewal of inbound connection endpoints
- Six new ReadyFlows have been added to the ReadyFlow Gallery:
  - Airtable to S3/ADLS
  - Box to S3/ADLS
  - Dropbox to S3/ADLS
  - Google Drive to S3/ADLS
  - MQTT to Kafka
  - Salesforce to S3/ADLS
- Apache Iceberg Integration [Technical Preview]

  Developers can now use the PutIceberg processor to write data to Apache Iceberg in CDP.
- DataFlow is now also available in the EU and APAC regional CDP control planes.

  Note: DataFlow Functions is an exception, as it is only available in the US control plane.
- Updated to support using Kubernetes 1.23 EKS and AKS clusters.

  If you use restricted IAM roles in your EKS clusters, this impacts the upgrade procedure. For more information, see the CDF Service Upgrade documentation.
- General improvements to DataFlow upgrade reliability to allow for improved success rates.
- Includes new NiFi version 1.18.0.2.3.7.0 for flow development and flow deployment.
- Updated DataFlow Function binaries to use NiFi version 1.18.0.

Fixed issues

- CDPDFX-5884 - Internal Logging Improvements - fixed logging issues sent to S3 (AWS) and Azure Blob Storage (Azure)
- CDPDFX-6308 / CDPDFX-6323 - updated legacy inbound connection endpoints to make them compatible with new certificate renewal capability
- CDPDFX-6305 - Fixed issue with inbound connection endpoints created prior to DFX 2.2.0 preventing upgrade
- CDPDFX-5689 - Expanded character limit on deployment parameters to allow for parameters of up to 100000 characters.

- CDPDFX-5942 - Fixed a rare issue where a cluster update request can fail due to cloud API latency leading to timeouts.
- CDPDFX-6334 - Fixed an issue where a large deployment count with active metrics unavailable alerts can lead to a deadlock which prevents upgrade
- CDPDFX-6345 - Fixed issue which causes existing deployments using assets to no longer allow changes to those assets after upgrade due to a certificate mismatch
- CDPDFX-6364 - Fixed CLI issue where some listen components are ignored when creating deployments with multiple listen components
- CDPDFX-6378 - Fixed issue where a post-upgrade cleanup effort can fail with a warning if a customer's outbound network restrictions do not allow retrieval of a liquibase schema file.
- CDPDFX-5731 - Cert-manager OOM - increased memory limits for cert-manager to ensure certificates can still be issued as deployment numbers increase
- CDPDFX-6371 - Correct nifi.provenance.repository.debug.frequency to a parseable default value (default 1000000)
- CDPDFX-6309 - ZK-Operator OOM - prevent operator from running OOM due to it scan/filtering of multiple secrets
- CDPDFX-6118 - Suppress deprecation logging in NiFi
- CDPDFX-6387 - Increased cpu limit for cer-manager-webhook container
- CDPDFX-6235 - Fix errors seen during NiFi autoscaling due to inconsistent nifi identity configmap updates
- CDPDFX-6230 - Fix to prevent metering alert from firing when DataFlow service not fully enabled

October 21, 2022

This release (2.2.0-h4-b2) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces compatibility fixes around upgrades and public ingress certificate renewals.

Fixed issues

- CDPDFX-6119 - Fixed an issue where a failed attempt to renew public ingress certificates incorrectly fires a certificate expiring soon warning.
- CDPDFX-6189 - Fixed an issue where public ingress certificate renewal fails for workloads that are not running the latest DataFlow version.
- CDPDFX-6191 - Fixed an issue where DataFlow upgrade fails if a public ingress certificate renewal has ever been attempted on the DataFlow service.
- CDPDFX-4177 - Improved reuse of recently generated public ingress certificate upon repetitive retries to avoid a Let's Encrypt certificate rate limit of 5 per 7 days.

October 10, 2022

This release (2.2.0-h3-b2) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces compatibility fixes around upgrades.

Fixed issues

- CDPDFX-6101 - Fixed an issue with DataFlow upgrade leaving deployments using custom NARs not able to retrieve the NAR components from S3. This results in custom processors becoming non-functional.
- CDPDFX-6120 - Fixed an issue with DataFlow Upgrade where the internal self-signed CA certificate is being unnecessarily renewed in upgrade scenarios where the certificates are not also renewed because they were already renewed in prior upgrades. This issue would have arisen for customers upgrading from DFX 2.2.0-h1 or 2.2.0-h2 to this version.
- CDPDFX-6025 - Fixed a documented known issue where a CDP environment configured with a storage base location with an S3 bucket but no root directory results in workload logs not being uploaded.
- CDPDFX-6055 - Fixed an issue where Update DataFlow fails on DataFlow services running version 2.0.0, 2.1.0 and 2.1.0-h1.
- CDPDFX-5944 - Added support to allow certificate renewal action on a DataFlow service in an 'Update Failed' state.

September 29, 2022

This release (2.2.0-h2-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces fixes improving performance of control plane database queries.

Fixed issues

Control Plane bug fixes

September 28, 2022

This release (2.2.0-h1-b11) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces fixes for certificate renewal issues that could cause the deployment manager and NiFi canvas to become inaccessible.

Fixed issues

- CDPDFX-5941 - Fixed an issue where CDF did not restart automatically to pick up renewed certificates resulting in the deployment manager and NiFi canvas becoming inaccessible.
- CDPDFX-5943 - Fixed an issue where the trusted certificate CA was not updated for vault after renewing certificates resulting in the deployment manager and NiFi canvas becoming inaccessible.
- CDPDFX-5953 - Fixed an issue where CDF provisioned certificates with an incorrect CA lifetime resulting in the deployment manager and NiFi canvas becoming inaccessible after 90 days .
- CDPDFX-5954 - Fixed an issue where prometheus and alertmanager pods did not correctly reload certificates upon renewal.
- CDPDFX-5961 - Added renewal of all certificates during CDF upgrades to prevent any further certificate related issues.
- CDPDFX-6025 - Fixed an issue where no CDF logs were written to S3 when CDP's storage base location contains only a bucket name.
- CDPDFX - 5990 - Fixed an issue where a 'metrics unavailable' alert was incorrectly fired during CDF upgrades.
- CDPDFX-5965 - Fixed an issue where flow definitions failed to deploy successfully due to their size.

Technical Service Bulletins

- TSB-2022-621: Certificate Authority (CA) certificate lifespan being too short causing authentication problems with the Cloudera DataFlow Service

September 08, 2022

This release (2.2.0-b194) of Cloudera DataFlow (CDF) on CDP Public Cloud improves the first time user experience, introduces more ReadyFlows, and supports renewing certificates for Inbound Connection.

New features and changes

- A streamlined first time user experience

  You can use the new Hello World ReadyFlow to create your first flow deployment without dependencies on source or target systems. New landing pages for the Dashboard and Catalog allow you to get started with just one click.

- Four additional new ReadyFlows are available in the ReadyFlow Gallery
  - Non-CDP ADLS to S3/ADLS
  - Non-CDP S3 to S3/ADLS
  - ListenTCP filter to S3/ADLS
  - ListenSyslog filter to S3/ADLS
- Endpoint Hostnames certificates can now be renewed when using Inbound Connections

Fixed issues

- CDPDFX-3348 - Support custom KMS encryption key defined at CDP environment level
- CDPDFX-5126 - Improve tag validation for DataFlow service enablement
- CDPDFX-5449 - Fixed an issue where Data offloading during scale down operations is blocking reconciliation in CDF causing deployment requests to time out
- CDPDFX-5494 - Fixed an issue where Kubernetes node scaling events are not surfaced in the event history for DataFlow services
- CDPDFX-5555 and CDPDFX-5556 - Improved reliability of Kubernetes infrastructure reuse when DataFlow enablement fails
- CDPDFX-5577 - Fixed an issue where UDP Inbound Connections fail in AWS due to incorrect health check configuration for inbound data load balancer
- CDPDFX-5607 - Improve visibility of helm release details when enable or upgrade DataFlow action fails to complete

August 16, 2022

This release (2.1.0-h1-b10) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces improvements for Prometheus performance, reliability and performance improvements for upgrades, as well as a number of additional bug fixes.

Improvements and fixed issues

- Fixed regression in DFX 2.1.0 where new deployments contain NiFi pods with misconfigured cpu/memory limits. These limits are lower than necessary, leading to the possibility of NiFi pod crashing due to off-heap storage growing large enough to cause an Out of Memory error.
- CDPDFX-5483 - Fixed a bug where disabling DataFlow from certain failure states can potentially leave behind some cloud resources deployed by DataFlow.
- CDPDFX-5486 - Several settings for Prometheus instances running on the workload cluster have been adjusted to improve performance. This includes a reduction in the amount of metrics collected by Prometheus instances and enabling write ahead log compression.
- CDPDFX-5487 - Improved Enable, Upgrade/rollback and Create deployment performance to more reliably interpret status of helm releases and NiFi resources for improved success rates.
- CDPDFX-5494 - Fixed an issue where cluster node scaling events are not being properly surfaced to the user.
- CDPDFX-5495 - Fixed a bug where upgrades failed if a deployment was created with autoscaling disabled, but since was updated in any way (node count, params, and so on).
- CDPDFX-5498 - Improved upgrade timeout configurations to account for delays upgrading clusters with high node counts.
- CDPDFX-5568 - Fixed a bug causing some environments not to show in the filter options on the Deployment Dashboard.
- CDPDFX-5571 - Fixed potential for up to 10-minute delay during upgrades where helm repository sync is unnecessarily delayed. This could sometimes lead to timeout failures during upgrade.
- CDPDFX-5572 - Fixed a bug in the user interface on the Deployment Dashboard when an environment filter option is selected, but becomes no longer valid.
- CDPDFX-5606 - Fixed a bug in retrying a failed DataFlow enablement where a new enable attempt fails fast and old infrastructure is eventually deleted. Old infrastructure details are not cleared from the database, so subsequent retries attempt and fail to clean up infrastructure that is already deleted.
- CDPDFX-5676 - Fixed an issue where in-cluster certificates are renewed, but certain cluster resources are not automatically restarted to start using the new certificates.
- CDPDFX-5736 - Fixed an issue limiting upload of flow definition files to the catalog to 1MB in size. Flow uploads are now supported up to 25MB in size.
- CDPDFX-5780 - Fixed an issue preventing creation of deployments with large flow definitions.

June 28, 2022

This release (2.1.0-b123) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces support for DataFlow service upgrades as a technical preview feature, Inbound Connection configuration through the create-deployment CLI command, Kubernetes 1.22 as well as additional improvements and fixes.

New features and changes

- [Technical Preview] CDF environment and deployment upgrades are now supported.

  CDF service upgrades require a specific entitlement that is granted by request only. Reach out to your Cloudera Account Representative or open a support ticket to request CDF service upgrades.
- The `create-deployment` CLI command now supports Inbound Connections.
- CDF now correctly configures the deployment when it detects any of the two reserved Controller Services Default NiFi SSL Context Service and Inbound SSL Context Service regardless of whether they are external references or not.
- Kubernetes Version 1.22 is now the default Kubernetes version for CDF. When CDF is enabled, it creates AWS EKS and Azure AKS clusters based on version 1.22.
- CDF now supports Data Lake scaling. After the Data Lake scale operation has completed, affected deployments will show an active alert and need to be manually restarted to put the new configuration in effect.
- The Event Hub to ADLS ReadyFlow now allows you to specify an Event Hub Service Bus Endpoint.

Fixed issues

- CDPDFX-5317 - Fixed an issue that could cause timeouts when uploading assets as part of the Deployment Wizard
- CDPDFX-5400 - Fixed several issues that could cause cluster and deployment Prometheus instances to run out of memory
- CDPDFX-5322 - Fixed an issue with caching of UI content that sometimes required user to perform a hard refresh
- CDPDFX-5261 - Fixed an issue that causes autoscaling CPU utilization target to change from 75% to 80% if user performs a deployment update

  CDPDFX-5325 - Use availability sets instead of zones for Azure to reduce the chance of a customer failing to enable DataFlow due to instance type resource constraints in certain regions

  CDPDFX-5319/CDPDFX-5385 - Improved resiliency when recovering from network disruptions and database failovers to improve the chances of deployment related requests succeeding

  CDPDFX-5260 - Expanded the types of cluster resource logs that are exported to cloud storage

  CDPDFX-5483 - Fixed an issue where some cloud resources are not deleted properly when disabling DataFlow after it has encountered an update failure or upgrade failure

May 17, 2022

This release (2.0.0-b302) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces support for Inbound Connection Endpoints, allowing you to stream data from external sources to Flow Deployments. It also adds support for Apache NiFi 1.16, adds seven new ReadyFlows and improves stability of the service.

New features and changes

- Inbound Connection Endpoints for Flow Deployments

  You can now deploy NiFi flows containing processors like ListenHTTP, ListenTCP, ListenUDP, ListenSyslog etc. and let CDF-PC take care of exposing a stable and secure endpoint to connect your clients to.
- Flow Deployments now support the latest Apache NiFi 1.16 release

- Seven new ReadyFlows are available in the ReadyFlow Gallery
  - Azure Event Hub to ADLS ReadyFlow
  - Confluent Cloud to S3/ADLS ReadyFlow
  - Confluent Cloud to Snowflake ReadyFlow
  - JDBC to S3/ADLS ReadyFlow
  - ListenHTTP to CDP Kafka ReadyFlow
  - Non-CDP ADLS to CDP ADLS ReadyFlow
  - Non-CDP S3 to CDP S3 ReadyFlow
- Single AZ support for DataFlow clusters

  CDF Kubernetes nodes are now always deployed in a single Availability Zone (AZ). When enabling CDF, you still have to provide subnets in at least two Availability Zones to satisfy Kubernetes control plane requirements.
- Value of javax.net.ssl.trustStore property no longer set to existing trust store on NiFi node

  The javax.net.ssl.trustStore system property is no longer set by default to the existing truststore on the NiFi node. Flows which rely on a truststore being available by default via the JVM should be updated to leverage the SSLContextService.

  This change may cause errors affecting older existing processors or custom processors that do not have a property to reference an SSL Context Service.
- Flow Deployments on Azure now use Premium SSD disks to store NiFi's various repositories
- CDF-PC integration into the CDP CLI is now generally available. Make sure you run the latest CDP CLI version to use cdp df commands.
- Python 3.9 is now installed on NiFi containers allowing users to run python scripts through NiFi processors. Requests and urllib3 libraries are also installed by default.
- Information level alerts are now displayed in the Dashboard to inform users about an action they should perform for a Flow Deployment without changing the Flow Deployment status to Concerning.
- Deployments can now be restarted from the Deployment Manager to perform actions that require a NiFi process restart.
- Filters that have been set in the Dashboard are now persisted after a user has navigated to the Deployment Manager and returned back to the Dashboard.
- When enabling DataFlow, you can now specify CIDRs as a comma separated list to configure Load Balancer and Kubernetes API Server Endpoint Access.

Fixed issues

This release includes all cumulative fixes released throughout 1.1.0-h1 to 1.1.0-h5 as well as additional stability and performance improvements.

April 06, 2022

This release (1.1.0-h5-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud fixes a deployment issue on Azure and introduces a new NiFi version (1.15.0.2.3.3.1-2) which fixes several issues that prevented deployments from downscaling.

Fixed issues

- CDPDFX-4440 – Fixed an issue that caused deployments to fail on Azure when a persistent storage volume was provisioned in a different Availability Zone than the NiFi pod.
- NIFI-9433 -- Fixed an issue that resulted in negative queue counts when using load balanced connections.
- NIFI-9835 -- Fixed an issue that prevented downscaling when queue count was negative.
- NIFI-9847 -- Fixed an issue where use of certain processors could prevent a NiFi node from offloading its data.

March 23, 2022

This release (1.1.0-h4-b2) of Cloudera DataFlow (CDF) on CDP Public Cloud fixes several issues around certificate renewal of long running DataFlow services and improves stability and performance for deployments with a large event history.

Fixed issues

- CDPDFX-4251 – Fixed an issue that could cause DataFlow service disable to fail when disabling was initiated through deleting the associated CDP environment.
- CDPDFX-4204 – Improved how CDF ages off historical heartbeats for increased stability and performance.
- CDPDFX-4203 – Reduced the retention time of events and metrics gathered in the DataFlow Prometheus instance for increased stability and performance.
- CDPDFX-4199 – Fixed an issue where DataFlow could become unresponsive due to the high number of events for a deployment.
- CDPDFX-4198 – Fixed an issue where DataFlow attempted to revoke already expired certificates causing the certificate renewal to fail.
- CDPDFX-4201 – Fixed an issue that prevented users from initiating certificate renewal for healthy DataFlow services when the associated CDP environment was in an unhealthy state.

March 8, 2022

This release (1.1.0-h3-b2) of Cloudera DataFlow (CDF) on CDP Public Cloud fixes an issue that caused DataFlow enablements to fail on AWS.

Fixed issues

CDPDFX-4010 – Fixed an issue that caused DataFlow enablements to fail on AWS due to requesting a deprecated Postgres version for RDS.

February 15, 2022

This release (1.1.0-h2-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces support for Kubernetes v1.21, removes disk-related metrics, and fixes a NiFi deployment issue.

New features and changes

Added support for Kubernetes v1.21

Version 1.21 is now the default Kubernetes version for CDF. When CDF is enabled, it creates AKS/EKS clusters based on version 1.21.

Removed disk related metrics from deployments and DataFlow service details

In previous versions of CDF, deployments and enabled DataFlow services showed disk capacity and disk usage metrics as part of their system metrics. You were also able to define KPIs and alerts on these metrics. Due to issues with the underlying metrics collection framework, the following metrics have been removed starting with CDF 1.1.0-h2-b1:

- Disk Capacity (DF Service Metric)
- Disk Capacity (Deployment System Metric)
- Disk Usage (Deployment System Metric)

Fixed issues

CDPDFX-4030 – Fixed an issue where NiFi pods were not correctly applying the heap size which the user selected during deployment.

Documentation updates

Removed disk related metrics documentation.

January 27, 2022

The 1.1.0-h1-b2 and 1.1.0-b127 releases of Cloudera DataFlow (CDF) on CDP Public Cloud introduce the following changes.

January 27, 1.1.0-h1-b2

Fixed issues

- Fixed an issue that prevented DataFlow from being enabled on Azure environments.

January 27, 1.1.0-b127

New features and enhancements

- The ability to deploy CDF on Azure is now generally available to all CDP Public Cloud customers.
- Flow Deployments now support the latest Apache NiFi 1.15 release.
- CDF now creates Kubernetes clusters with version 1.20 in AWS EKS and Azure AKS.
- CDF now supports username/password authentication for AWS environments with non-transparent proxy setups.
- When using the Default NiFi SSL Context Service in a flow deployment, the automatically generated truststore now contains the default cacerts from the local JDK. This ensures that the SSL Context Service can be used with third-party applications using certificates from common public certificate authorities.
- Users can now perform the Download Kubeconfig and Manage User Access actions when an enablement request fails. This allows in-depth troubleshooting of a failed enablement attempt.
- The following ReadyFlows have been added to the ReadyFlow Gallery:
  - ADLS to ADLS Avro ReadyFlow
  - Kafka to ADLS Avro ReadyFlow

Fixed issues

- CDPDFX-3888 - Fixed an issue that prevented CDF from being enabled in Azure regions where the multi-AZ feature is not available.
- CDPDFX-3873 - Fixed an issue where large file uploads could cause flow deployment requests to fail on Azure.
- CDPDFX-3909 - Fixed an issue that prevented using special characters like “@” as tag values when enabling CDF.

Documentation updates

- Documentation has been created for the new ADLS to ADLS Avro ReadyFlow and Kafka to ADLS Avro ReadyFlow.
- NiFi Endpoints Support Matrix has been updated.
- Troubleshooting information has been updated for Azure.
- Azure technical preview references have been removed.

December 21, 2021

This release (1.0.3-h2-b1) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces these changes.

Fixed issues

- Fixed bug causing DataFlow to incorrectly appear to have been upgraded to the latest version.

  When you use Manage DataFlow to modify your DataFlow service, updates to the following values cause the Workload Version field for your service to be set to the current DataFlow application version.

  - Capacity
  - Networking
  - Kubernetes API Server Endpoint Access

  Once the update is started, it will incorrectly appear as if your DataFlow service is upgraded to the latest version.

  CDPDFX-3897

NOTE: There is no need to tear down and recreate your enabled DataFlow services to benefit from these updates.

December 17, 2021

This release (1.0.3-h1-b6) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces these changes.

Fixed issues

- Upgraded all log4j and logback libraries present in DataFlow application to the latest version.
- Upgraded all log4j and logback libraries present in the DataFlow NiFi Runtime to the latest version.
- Removed requirement for setting unnecessary ingress rules on Azure Network Security Group during AKS cluster creation, which could trip policy rules preventing such NSG updates.
- Fixed a bug that prevented selecting specific Load Balancer subnets when enabling Azure DataFlow service.

December 1, 2021

This release (1.0.3-222) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces these changes.

New features and enhancements

[Technical Preview] CDF can now be deployed on Azure

Deploying CDF on Azure requires a specific entitlement that is granted by request only. Reach out to your Cloudera Account Representative or open a support ticket to request CDF on Azure.

Custom Processors and Controller Services can be used in NiFi flow deployments.

CDF supports deploying NiFi flows that require custom components such as processors or controller services. Users can store their custom NiFi Archive (NAR) files in an object store location in S3 or ADLS and use the Deployment Wizard or the CLI to create their flow deployments accordingly. Once configured, CDF ensures that the custom NARs are available to all NiFi pods as the deployment scales up and down.

Non-transparent proxy support in AWS environments

If you have a non-transparent proxy registered for your CDP environment, CDF reuses the existing configuration and routes its outbound traffic through the proxy.

Note: Depending on the NiFi flows you are using, you might have to configure additional proxy settings if you require the NiFi data to go through the proxy as well.

Private EKS/AKS Kubernetes API Endpoints

CDF now supports private EKS/AKS Kubernetes API Endpoints for CDP environments which have been setup using Cloudera Connectivity Manager (CCM v2). When enabling CDF, you need to select a checkbox to create a private endpoint.

Enabling CDF for an environment with a minimum of one EKS/AKS instance

CDF can now be enabled with a minimum of one EKS/AKS instance and will scale up as required until the specified maximum number of Kubernetes nodes is reached. Previously the minimum number of nodes was set to three.

CDF instance sizing changes

CDF now uses c5.4xlarge instances on AWS and Standard_D16s_v4 on Azure to provide more efficient infrastructure usage.

NiFi node sizing changes

NiFi Node Sizes in the Deployment Wizard have also been updated to use available resources more efficiently. The new options are:

- Extra Small – 2 vCores, 4GB memory
- Small – 3 vCores, 6GB memory
- Medium – 6 vCores, 12GB memory
- Large – 12 vCores, 24GB memory

Flow deployment security certificates

Security certificates for deployments are now automatically renewed one month before their expiration. A manual restart is required to pick up the new certificates.

CDF UI guidance on using CDP CLI

The CDF UI now provides guidance on how to use the CDP CLI to enable and disable CDF, upload and delete flow definitions as well as editing existing flow deployments.

CDP Beta CLI support for editing flow deployments

The CDP Beta CLI now includes functionality to edit existing deployments allowing configuration changes such as parameter and KPI updates, scaling configuration, or NiFi version upgrades.

New events and alerts for pod scaling

Introduced deployment pod scaling alerts and events to reflect when a deployment auto scales and when scaling limits are approached.

Fixed issues

- CDPDFX-3568 – Fixed an issue where no tooltip data was shown for the Data Throughput charts in the Dashboard.
- CDPDFX-3588 – Fixed an issue where CDF did not write logging data to non us-west-1 S3 buckets.
- NIFI-9217 – Fixed an issue that prevented scale down operations due to NiFi nodes getting stuck in “offloading” mode.
- CDPDFX-3624 – Fixed an issue where deployment auto-scaling value changes were not saved.

October 13, 2021

This release (1.0.2-244) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces these changes.

Fixed issues

- Fixed an issue that caused some DataFlow CLI commands to be unavailable in the Beta CDP CLI.

Documentation updates

The following documentation has been updated with CLI instructions.

- Enable DataFlow for an Environment
- Disable DataFlow for an Environment
- Deploy a flow
- Importing a flow definition to Cloudera DataFlow

- Importing a new version of a flow definition
- Adding a ReadyFlow to the Catalog

September 29, 2021

This release (1.0.2-b240) of Cloudera DataFlow (CDF) on CDP Public Cloud introduces these changes.

New features and enhancements

- CLI support has been added for creating flow deployments and flow definition management.

  DataFlow CLI is available in the Beta CDP CLI.
- Added the S3 to S3 ReadyFlow leveraging S3 bucket notifications.
- DataFlow now supports Cloudera RAZ (fine-grained access control) for object store access.
- The Dashboard, Catalog, and Environments pages now persist search queries and applied filters for a better user experience.
- When deploying a flow definition, you can now select whether the NiFi flow should be started automatically.

Fixed issues

- Fixed an issue where Alerts would not correctly become inactive after the alert condition is no longer met.

  CDPDFX-3396
- Fixed an issue where Alerts were still generated when a deployment was suspended.

  CDPDFX-3419
- Fixed an issue that prevented users from using carriage return in the Flow Definition description.

  CDPDFX-3294
- Fixed an issue that resulted in environment system metrics being rendered multiple times.

Documentation updates

- ReadyFlow documentation
  - New ReadyFlow documentation has been developed for all ReadyFlows

    Using ReadyFlows
  - Documentation for the new S3 to S3 Avro with S3 Notifications ReadyFlow

    S3 to S3 Avro with S3 Notifications ReadyFlow Overview
- CLI documentation
  - Enable DataFlow for an Environment
  - Disable DataFlow for an Environment
  - Deploy a flow
- Updated ReadyFlows for RAZ integration

  The three AWS ReadyFlows include instructions on configuring access to AWS buckets when your environment is RAZ enabled
  - Kafka to S3 Avro
  - S3 to S3 Avro
  - S3 to S3 Avro with S3 Notifications
- NiFi Endpoints Support Matrix

August 24, 2021

This release of Cloudera DataFlow (CDF) on CDP Public Cloud introduces these changes.

Fixed Issues

- Fixed a bug that prevented DataFlow and associated deployments from being terminated when the associated CDP Environment had been deleted.

  CDPDFX-3327
- Fixed a bug that could result in duplicate system metrics being displayed in the Environment details view.

  CDPDFX-3318
- DataFlow now uses the single api.us-west-1.cdp.cloudera.com endpoint for Control Plane API access.

  CDPDFX-3288

August 16, 2021

This is the GA release of the Cloudera DataFlow service.

Cloudera DataFlow (CDF) is a CDP Public Cloud service that enables self-serve deployments of Apache NiFi data flows from a central catalog to auto-scaling Kubernetes clusters managed by CDP. Flow deployments can be monitored from a central dashboard with the ability to define KPIs to keep track of critical data flow metrics.

CDF eliminates the operational overhead that is typically associated with running Apache NiFi clusters and allows users to fully focus on developing data flows and ensuring they meet business SLAs.

Related Information

Cloudera DataFlow Overview

Cloudera DataFlow Quick Start

Supported component versions

Cloudera DataFlow includes support for the following component versions.

Cloudera DataFlow version	Release date	Cloudera DataFlow component	Component version
Cloudera DataFlow 2.9.0 hotfix 4, build 3	December 03, 2024	Apache NiFi	2.0.0-M2 [Technical Preview] 1.27.0 1.25.0 1.24.0 1.23.0 1.21.0 1.20.0 1.18.0
Cloudera DataFlow 2.9.0 hotfix 3, build 1	November 20, 2024	Apache NiFi	2.0.0-M2 [Technical Preview] 1.27.0 1.25.0 1.24.0 1.23.0 1.21.0 1.20.0 1.18.0

Cloudera DataFlow version	Release date	Cloudera DataFlow component	Component version
Cloudera DataFlow 2.9.0 hotfix 2, build 1	November 15, 2024	Apache NiFi	2.0.0-M2 [Technical Preview] 1.27.0 1.25.0 1.24.0 1.23.0 1.21.0 1.20.0 1.18.0
Cloudera DataFlow 2.9.0 hotfix 1, build 1	November 11, 2024	Apache NiFi	2.0.0-M2 [Technical Preview] 1.27.0 1.25.0 1.24.0 1.23.0 1.21.0 1.20.0 1.18.0
Cloudera DataFlow 2.9.0, build 383	October 31, 2024	Apache NiFi	2.0.0-M2 [Technical Preview] 1.27.0 1.25.0 1.24.0 1.23.0 1.21.0 1.20.0 1.18.0
Cloudera DataFlow 2.8.0 hotfix 2, build 2	June 25, 2024	Apache NiFi	2.0.0-M2 [Technical Preview] 1.25.0 1.24.0 1.23.0 1.21.0 1.20.0 1.18.0
Cloudera DataFlow 2.8.0 hotfix 1, build 1	June 03, 2024	Apache NiFi	2.0.0-M2 [Technical Preview] 1.25.0 1.24.0 1.23.0 1.21.0 1.20.0 1.18.0

Cloudera DataFlow version	Release date	Cloudera DataFlow component	Component version
Cloudera DataFlow 2.8.0, build 274	May 20, 2024	Apache NiFi	2.0.0-M2 [Technical Preview] 1.25.0 1.24.0 1.23.0 1.21.0 1.20.0 1.18.0
Cloudera DataFlow 2.7.0, build 190	January 9, 2024	Apache NiFi	1.24.0 1.23.0 1.21.0 1.20.0 1.18.0

Related Information

Cloudera Support lifecycle policy

Known issues and limitations

You must be aware of the known issues and limitations, the areas of impact, and workarounds in Cloudera DataFlow.

Known issues

Cloudera DataFlow Functions: DEFAULT_PARAM_CONTEXT variable no longer works alone

This issue only occurs in AWS environments and affects configurations where there are no PARAM_CONTEXT_ variables defined, only a DEFAULT_PARAM_CONTEXT.

The DEFAULT_PARAM_CONTEXT configuration variable instructs Cloudera DataFlow Functions which default secret to use when there is no secret that matches the parameter contexts in the flow. This variable is now ignored.

Create an environment variable in Configuration called PARAM_CONTEXT_*[***NAME***]* where *[***NAME***]* is the user-defined name of the parameter context. Specify the name of the AWS Secret you want to use as the value of this variable.

NiFi 2.0 [Technical Preview] deployments fail to obtain authentication token in RAZ enabled AWS environments

NiFi 2.0 [Technical Preview] deployments in RAZ enabled AWS environments are not able to obtain an authentication token and therefore fail to read/write to RAZ protected sources/destinations. NiFi 2.0 [Technical Preview] deployments without any Ranger RAZ dependencies work as expected.

There is no workaround for this issue.

IAM Policy Simulator preflight check fails with resource policy validation

With all cross account policies in place, IAM Policy Simulator preflight check still fails with the following error message:

```
IAM Resource Policy validation failed on AWS. CrossAccount role
does not have permissions for these operations : : ssm:GetParame
ter, ssm:GetParameters, ssm:GetParameterHistory, ssm:GetParamete
rsByPath
```

This happens because even if a given cross account role is allowed to perform a certain action (granted through IAM Policies), an attached Service Control Policy (SCP) may override that capability if it enforces a Deny on that action. SCP takes precedence over IAM Policies. SCPs are either applied at the root of an organization, or can be applied to individual accounts. A permission can be blocked at any level above the account, either implicitly or explicitly (by including it in a Deny policy statement).

As the IAM Simulator SDK does not have an option to include or exclude an organization's SCP policy, the preflight check will fail if an SCP policy is denying an action, even though the IAM role has the necessary permissions.

This is a known issue in AWS.

Do not select the Skip Validations option when enabling Cloudera DataFlow to bypass this issue. This bypasses all preflight validation checks. Instead, submit a request to add the LIFTIE_DISABLE_IAM_PREFLIGHT_CHECK entitlement to your account which ensures only the IAM Policy preflight validation check is skipped.

Limitations

Parameter groups that have referencing flow drafts, and flow drafts referencing a parameter group cannot be reassigned to another project

A flow draft referencing any parameter group cannot be reassigned to another project.

Project reassignment does not move assets. When reassigning a parameter group that includes a FILE type parameter (asset) reference to another project, that asset needs to be re-uploaded to the new project.

There is no workaround for this issue.

Duplication of parameter groups referencing assets does not result in asset duplication

After duplicating a parameter group, assets have to be re-uploaded manually. This means that if you reassign a parameter group to another project, duplicate a parameter group, or export a parameter group, the asset will not be moved or duplicated.

Duplication can only happen inside a given project. If the newly created group is to be assigned to another project, that step can happen only after the duplication concludes. (Another project cannot be targeted for duplication.)

There is no workaround for this issue.

Diagnostic Bundle collection through the Management Console is available on the US Control Plane only

There is no workaround for this issue.

Data Lineage information is not automatically reported to Atlas in the Data Catalog

Flow deployments created by Cloudera DataFlow do not come with a pre-configured `ReportLineageToAtlas` Reporting Task.

If you have been assigned the DFFlowAdmin role, you can manually create and configure the `ReportLineageToAtlas` Reporting Task in the NiFi canvas after a deployment is completed.

PowerUsers are not able to create flow deployments without additional Cloudera DataFlow roles

While the PowerUser role gives you the ability to view flow deployments in the Dashboard, view flow definitions in the Catalog, and allows the user to initiate flow deployments, the Deployment Wizard fails after selecting an environment for which the user does not have the DFFlowAdmin resource role assigned.

Assign the DFFlowAdmin role to the user of the environment to which they want to deploy flow definitions.

Cloudera DataFlow reports "Bad Health" during Data Lake upgrade

Cloudera DataFlow monitors the state of the associated Cloudera Public Cloud environment to decide which actions Cloudera DataFlow users can take. Cloudera DataFlow detects Data Lake upgrades of the associated Cloudera Public Cloud environment and puts the Cloudera DataFlow service into Bad Health for the duration of the upgrade blocking new deployments.

To work around this issue, wait for the Data Lake upgrade to complete before creating new flow deployments.

Deployments and Cloudera DataFlow Services are no longer visible in the Cloudera DataFlow Dashboard or Environments page when the associated Cloudera Public Cloud Environment has been deleted

If the associated Cloudera Public Cloud Environment is deleted while a Cloudera DataFlow Service is enabled, it will become orphaned. Orphaned resources are no longer visible to users without the `PowerUser` role.

To work around this issue, open the Environments or Dashboard page with a user who has been assigned the `PowerUser` role. `PowerUsers` are able to view orphaned deployments and Cloudera DataFlow services.

Non-transparent proxies are not supported on Azure

There is no workaround for this issue.

Cloudera DataFlow

Resources

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

The Resources view 4

Managing workspace resources 4

Deployments 5
- Viewing data flow in NiFi 5
- Starting a flow 5
- Stopping a flow 6
- Changing flow version 6
- Downloading NiFi application log 8
- Suspending a deployment 9
- Resuming a deployment 9
- Export deployment configuration 10
- Terminating a deployment 10

Flow drafts 11
- Opening a draft for editing 11
- Starting, stopping, and restarting a test session 11
- Publishing a flow draft 11
- Reassigning a flow draft to a different project 12
- Deleting a flow draft 12

Parameter groups 12
- Modifying parameters in a parameter group 13
- Deleting a parameter group 13
- Importing a parameter group to a flow draft 14
- Duplicating a parameter group 14
- Reassigning a parameter group to a different project 14

Inbound connections 15
- Renewing the certificate for an inbound connection endpoint 15
- Reassigning an inbound connection endpoint to a different project 15

Custom NAR configs 16
- Validating a custom NAR config 16
- Reassigning a custom NAR config to a different project 16

Custom Python configs 17
- Validating a custom Python config 17
- Reassigning a custom Python config to a different project 17

The Resources view

The Resources page is the central place for accessing Cloudera DataFlow resources across environments.

About this task

On the **Resources** you see all workspaces where you have DFFlowUser role. The number of resources is listed by type on each workspace card. To view or manage a resource type in a given workspace, click on the number displayed under the selected resource type.

The following resource types are listed:

- Deployments
- Flow drafts
- Parameter groups
- Custom NAR configurations
- Custom Python configurations

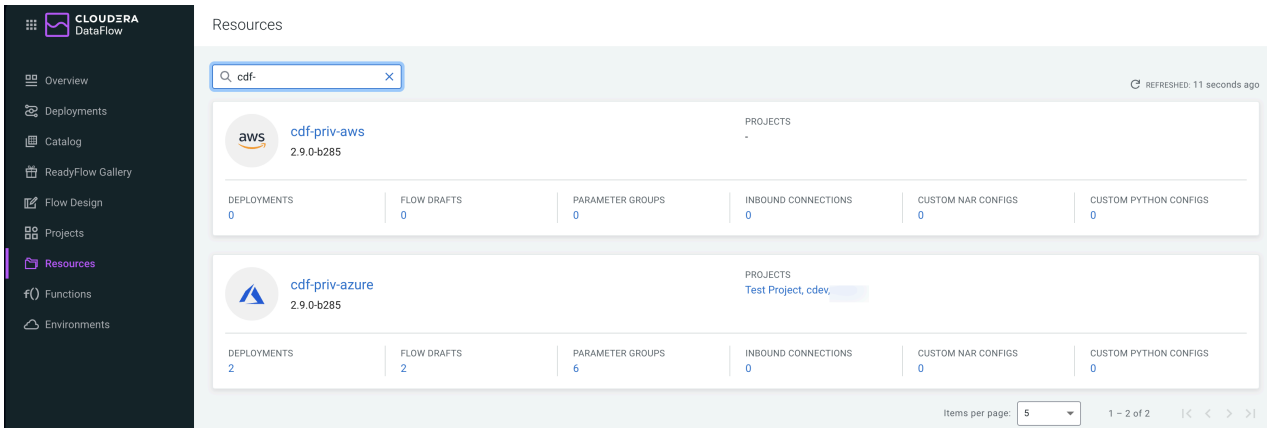

You can filter this view by starting to type the name of a workspace in the search box.

Note: If you do not see the workspaces you expect, check that you have been granted the proper roles for environments.

To manage workspace resources, select a workspace from the list. By clicking the blue number under the resource type, you will be redirected directly to the tab of that resource type on the **Workspace Resoureces** page. By default, the page opens on the **Deployments** tab.

Managing workspace resources

You can access all workspace resources accessible to you through the Resources view. Select a workspace to view, manage, and reassign resources.

Before you begin

You must have the DFAdmin role for the environments where you want to manage resources.

About this task

You can manage the following types of resources:

- Flow deployments
- Flow drafts

- Parameter groups
- Inbound connections
- Custom NAR configurations
- Custom Python configurations

The actions available to you depend on the resource type you select.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Select Resources.
3. Select the workspace where you want to manage resources.

   Tip: Start typing the resource name in the search bar to filter for a workspace.

4. On the **Workspace Resources** page select the tab with the type of resource you want to manage.

Deployments

On the Deployments tab you can view and manage all deployments within a workspace.

Viewing data flow in NiFi

You can go to the NiFi cluster where your flow is deployed and view or edit the data flow.

About this task

When you access the NiFi cluster, the ability to view or edit the flow is based on your Cloudera DataFlow authorizations. The DFFlowUser role has read-only privileges. The DFFlowAdmin role has full privileges.

Before you begin

You must have deployed a data flow in Cloudera DataFlow.

Procedure

1. Select the **Deployment** that you want to manage.
2. Click ⋮ Options View in NiFi .

   The UI for the NiFi cluster where your flow is deployed opens.
3. View your data flow or edit it based on your NiFi privileges.

   If you edit the flow in NiFi and want the changes to exist in a new deployment, perform the following steps:

   a) Download the flow as a flow definition.

      For more information, see *Downloading a flow definition from NiFi*.

   b) Import the flow definition (as a new flow definition or as a new version of an existing flow definition).

      For more information, see *Importing a flow definition to Cloudera DataFlow*.

   c) Deploy the flow definition.

      For more information, see *Deploy a flow*.

Starting a flow

You can start a stopped flow a Cloudera DataFlow deployment.

About this task

Starting a flow deployment starts all processors of a Cloudera DataFlow deployment.

Before you begin

- You must have a stopped flow deployment in Cloudera DataFlow.
- You must have DFFlowAdmin permission.

Procedure

1. Select the **Deployment** that you want to manage.
2. Click ⋮ Options Start flow .

   The Start [Deployment Name] pop-up appears.
3. Confirm your choice by clicking Start Flow.

Stopping a flow

Stopping the flow of a Cloudera DataFlow deployment temporarily pauses the NiFi flow.

About this task

Stopping a flow results in the following:

- All processors are stopped and no data processing happens within the NiFi flow.
- KPI alerts are stopped. Your KPI alerts are activated again when the flow is restarted.
- Any active KPI alerts are resolved.
- All underlying cloud resources remain allocated for the Cloudera DataFlow deployment.
- You can modify deployment configuration while the flow is stopped.
- Stopped flows are still billable however if auto-scaling is enabled for the flow, a certain amount of cost reduction may occur.

Before you begin

You must have deployed a flow definition in Cloudera DataFlow.

Procedure

1. Select the **Deployment** that you want to manage.
2. Click ⋮ Options Stop flow .

   The Stop [Deployment Name] pop-up appears.
3. Click Stop Flow to stop the flow deployment.

Changing flow version

Learn how to change the flow definition version of a running flow deployment. Using the ‘change flow version’ capability eliminates the need to terminate and re-create deployments when you want to deploy a new version of your flow definition.

Before you begin

- You must have DFFlowAdmin permission.
- There is at least one more version for the same flow definition present in the catalog.
- The state of the flow deployment is either Good Health or Stopped.
- You have read the applicable restrictions and version change strategies.

About this task

Important: As every flow is unique, you need to test version change before performing it in a production environment

Restrictions

The following restrictions apply to flow version changes:

- Changing inbound connections is not supported.
- Changing custom resource (custom NARs and custom Python resources) configurations is not supported.
- While you can add, change or remove assets when moving to a new version, you cannot introduce assets (text files, binaries, JARs, or similar) if the currently deployed version does not have any.
- Components where state or provenance and other repositories must be kept between flow versions must keep their flow JSON ids. The id changes if you move the component to a different process group or if you delete and then re-add the component to the same process group. NiFi Identifies components by these ids. If you move a component to a different process group between versions, its id changes and NiFi perceives it as a new component. This results in the original component being deleted during flow version change together with its state and a new, identical processor being created in a different process group. In an extreme case, you could change to an identical flow version with just the component ids changed and it would result in the deletion of the entire NiFi flow and the recreation of an identical one, with all history and data lost.
- Remapping Parameter Group and Parameter Context assignments is not supported as the original assignment is not removed. For example, you have Process Group 1 (PG1) with Parameter Context 1 (PC1) and Process Group 2 (PG2) with Parameter Context 2 (PC2) assigned. If you initiate a flow version change where parameter contexts are flipped, resulting in a PG1-PC2 and PG2-PC1 assignment, NiFi will not re-map the PG to PC assignments.

Version change strategies

Depending on the type of your flow, you may select the flow version change strategy most appropriate to you.

Stop & Process Data

This strategy prioritizes data consistency by stopping source processors and waiting until data is processed before stopping all other components. Once all components have stopped, the flow version is changed and components are started.

Use this strategy when your sources are durable and can handle your source processor being stopped. This generally works well when your source processors are pulling data from sources like Kafka or other messaging queues, databases or file systems.

Should the queued data not be processed within the set time, version change will fail and you can retry the operation with a bigger timeout or you can cancel

Only Restart Affected Components

This strategy prioritizes uptime by identifying and stopping only components that have changed while keeping all others running, replacing and then starting affected components.

Use this strategy when you want to prioritize uptime of unchanged components or you have made only minor processor configuration changes.

This works well for deployments with inbound connections and will keep your source processors running if they have not changed compared to the previous version.

Stop & Empty Queues

This strategy forces a version change by stopping all components, emptying all queues, changing flow version, and then starting all components.

Use this strategy only when you want to force a flow version change without keeping any processors running or attempting to process queued data.

All processors will be stopped and all queues will be emptied as part of this strategy.

Procedure

1. Select the **Deployment** that you want to manage.
2. Click Options Change Flow Version .

   The **Change Flow Version** modal window opens. It shows the list of available flow versions. The current version is grayed out.

   Tip: You can filter versions by name or by tags.
3. Select the flow version you want to change to and click Continue .
4. Review a summary of the configuration changes caused by the version change and make any necessary edits from the left pane.
5. Select a flow version change strategy.

   The available options are:

   - Stop & Process Data - If you select this strategy, you can set the maximum wait time for data to be processed and queues to be emptied before the request timed out. The default value is 15 minutes.

     Note: If you initiate flow version change with this strategy on a stopped flow and there is no data in the queues, flow version change instantly happens regardless of the wait time you configured. However if there is data left in the queues of the stopped flow, the change operation instantly fails.
   - Only Restart Affected Components

     Note: If you initiate flow version change with this strategy and there are queued flow files on any connection that is going to be removed in the new flow version, the version change will fail.
   - Stop & Empty Queues - If you select this strategy, you must accept potential data loss by selecting I understand and choose to proceed with the configuration as is.
6. Click Change Flow Version.

Results

After you click Deploy, you are redirected to the **Alerts** tab in the **Flow Details** view where you can track how the version change progresses.

Downloading NiFi application log

You can download the NiFi application log from the CDF Deployment Manager to use it for troubleshooting.

About this task

This feature allows you to download the NiFi application log that is currently being written. As the log file is rotated and the old file is archived once the file size reaches 10 MB, this is the theoretical maximum you can download using this method. For information on downloading archived log files, see *Diagnostic bundle collection*.

Before you begin

You need DFFlowAdmin permission to perform this action.

Procedure

1. Select the **Deployment** that you want to manage.
2. Click ⋮ Options Download NiFi Log .

   The current NiFi application log is downloaded to your computer in tar.gz format.

Suspending a deployment

Suspending a Cloudera DataFlow deployment terminates cloud resources belonging to a NiFi flow, while maintaining flow persistence.

About this task

Suspending a Cloudera DataFlow deployment results in the following:

- The NiFi flow stops processing data and all underlying cloud resources are terminated. Any unprocessed data in the flow is stored in memory and its processing resumes when you resume the deployment.
- Flow persistence is maintained while a deployment is suspended.
- You cannot modify deployment configuration while the deployment is suspended.
- Suspended deployments are not billable, resulting in reduced costs.

Before you begin

You must have deployed a flow definition in Cloudera DataFlow.

Procedure

1. Select the **Deployment** that you want to manage.
2. Click ⋮ Options Suspend Deployment .

   The Suspend [Deployment Name] modal opens.
3. Select the Finish data processing option and set a maximum wait time in minutes for data to be processed and queues to be emptied before the request times out.

   This option stops source processors first and waits for queued data to be processed before the flow is suspended. Set a wait time between 5 and 60 minutes using the slider.
4. Click Suspend to suspend the Cloudera DataFlow deployment.

Resuming a deployment

You can resume a suspended Cloudera DataFlow deployment.

About this task

Resuming a Cloudera DataFlow deployment reallocates the underlying cloud resources and returns a deployment to the state it was in before being suspended.

Before you begin

You must have a suspended flow deployment in Cloudera DataFlow.

Procedure

1. Select the **Deployment** that you want to manage.
2. Click ⋮ Options Resume Deployment .

   The Resume [Deployment Name] modal opens.
3. Click Resume Deployment to resume the flow deployment, reallocating cloud resources.

Export deployment configuration

You can export a deployment configuration to create additional deployments with a similar configuration in the same or a different environment.

About this task

- Exported configurations may be edited, and you can also modify them after the importing step during flow deployment.
- One deployment can have only one exported configuration. Performing a new export overwrites the existing one.
- Exported deployment configurations are available for every user who can start a new deployment in a given environment, even if the exported deployment was originally created under a specific Project.

Procedure

1. Select the **Deployment** that you want to manage.
2. Click ⋮ Options Export Configuration .

   The Export Deployment Configuration modal opens.
3. You can optionally add comments to the exported cofiguration.
4. Confirm your choice by clicking Export in the modal.

Results

The configuration is exported to the {LOG location}/cdf-deployment-backup directory. {LOG location} is configured during the creation of the associated Cloudera Environment. If you want to reuse the exported configuration in a different environment, you can either configure that to use the same {LOG location}, or you can copy the exported .tar.gz and JSON files to the {LOG location}/cdf-deployment-backup directory of the target environment.

What to do next

You can reuse the exported configuration during deployment of the same flow definition to recreate a flow with similar configuration.

Terminating a deployment

You can terminate a deployment to remove it from Cloudera DataFlow.

About this task

If you terminate a deployment, you delete the associated NiFi resources and your flow no longer remains active. The associated flow definition remains in the catalog and is available to be deployed again in a new deployment.

Before you begin

You must have deployed a flow definition in Cloudera DataFlow.

Procedure

1. Select the **Deployment** that you want to manage.
2. Click ⋮ Options Terminate Deployment .

   The Terminate [Deployment Name] modal opens.
3. If you select the Delete assigned endpoint hostname option. If you do not select this option, you can reassign existing, unassigned endpoints during flow deployment.

4. Enter the name of the deployment to confirm and click Terminate.

   Tip: You can also click the copy to clipboard icon to copy the deployment name, paste the name of the deployment, and then click Terminate.

Flow drafts

On the Flow drafts tab you can manage the life cycle of flow drafts and associated test sessions.

Opening a draft for editing

You can open a selected flow draft for editing in the Flow Designer.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Select the **Flow draft** that you want to manage.
2. Click Options Open .

   The flow draft opens in the Flow Designer.

Starting, stopping, and restarting a test session

Learn how you can start, stop, or restart a test session for a flow draft.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Select the **Flow draft** that you want to manage.
2. Select one of the following options
   - To start a test session, click Options Start Test Session . For more information on configuring a test session, see Test sessions.

     Note: You cannot start a test session if there is already an **Active** or **Failed** test session.
   - To stop a test session, click Stop Test Session, then click End to confirm your choice.
   - To restart an **Inactive** test session, click Options Restart Test Session

Publishing a flow draft

Learn about publishing a draft to the Catalog as a flow definition.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Select the **Flow draft** that you want to manage.
2. Click ⋮ Options Publish .
3. Provide a Flow Name and optional Flow Description, Custom Tag, and Version Description.
4. Click Publish.

Reassigning a flow draft to a different project

Learn how to reassign a flow draft to another project.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Select the **Flow draft** that you want to reassign.
2. Click Reassign.

   The **Reassign Resource** modal opens.
3. Select a Project and click Reassign.

Deleting a flow draft

Learn about deleting flow drafts.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Select the **Flow draft** that you want to delete.
2. Click ⋮ Options 🗑 .
3. Click Delete to confirm your choice.

Parameter groups

You can manage shared parameter groups in the Resources view of Cloudera DataFlow.

Procedure

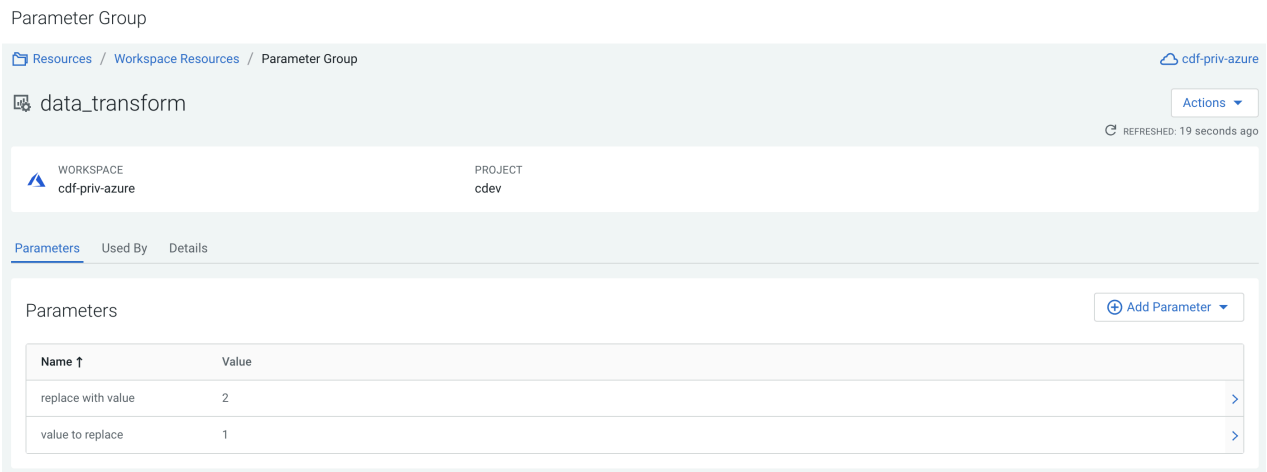

- To view the list of flow deployments and flow drafts using this parameter group, select the **Used By** tab.

  It is important to be aware of the flow deployments and flow drafts that are going to be affected by any change you make to a parameter group.
- To view or modify the Parameter Group Name and Description, select the **Details** tab.

Modifying parameters in a parameter group

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Select the **Parameter Group** where you want to modify parameters.
2. Click ⋮ Options Open .
3. Click on the parameter card of the parameter you want to modify.

   In the parameter details pane modify the parameter Value and Description as necessary.
4. Click ⬆ Apply Changes.

Deleting a parameter group

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Select the **Parameter Group** that you want to delete.
2. Click ⋮ Options Open .

3. Click on the parameter card that you want to delete.

   In the parameter details pane click to delete the parameter from the group.

   If the parameter group is not used by any flow draft, the **Delete [parameter group name]** modal opens. If the parameter group is in use by a flow draft, it cannot be deleted.
4. Click Delete to confirm your choice.
5. Click Apply Changes.

Importing a parameter group to a flow draft

Procedure

1. Select the **Parameter Group** that you want to import to a flow draft.
2. Click Options Import to Flow Draft .
3. In the **Import to Flow Draft** modal select the Flow Draft where you want to import the parameter group.
4. Click Import.

Duplicating a parameter group

Learn how to duplicate parameter groups. Duplicating a parameter group allows you to reassign the newly created group to another project, or to customize it according to your needs without touching the original group.

About this task

Note: Duplicating a parameter group does not duplicate any assets (file-type parameters) associated with it. You need to reupload those to the newly created group.

Procedure

1. Select the **Parameter Group** that you want to import to duplicate.
2. Click Options Duplicate .
3. In the **Duplicate Parameter Group** modal provide a Parameter Group Name and an optional Description.
4. Click Duplicate.

Reassigning a parameter group to a different project

Learn how to reassign a parameter group to another project.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

About this task

You cannot reassign a parameter group that is used by flow drafts. You have to remove the group from all flow drafts before you can reassign it to a different project.

Note: When you reassign a parameter group to another project, assets associated with it are not moved with it. You need to reupload those to the new project.

Procedure

1. Select the **Parameter Group** that you want to reassign.

   Note: You can reassign multiple items simultaneously by selecting the checkboxes in front of them.

2. Click Reassign.

   If the parameter group is not used by any flow draft, the **Reassign Resource** modal opens.

3. Select a Project and click Reassign.
4. Click Apply Changes.

Inbound connections

On the Inbound Connections tab you can reassign and renew the certifacates of inbound connections.

Renewing the certificate for an inbound connection endpoint

If you need to replace an X.509 certificate for an inbound connection endpoint before it expires, you can do so manually.

Before you begin

You need DFFlowAdmin privilege to perform this action.

Procedure

1. Select the **Inbound Connection** that you want to manage.
2. Click Renew.
   - To renew the server certificate, select NiFi Inbound SSL Context Service.

     Note: Each server certificate is limited to five renewals in a 7 day sliding window.

   - To renew the client certificate, select Client SSL Context.
   - If you leave Revoke previously issued client certificates unchecked, existing client certificates remain valid and existing clients can continue to connect to your deployment using it. By selecting the Revoke previously issued client certificates option, you invalidate all existing certificates and you will need to add the new certificate to existing clients so that they can keep connecting to your Cloudera DataFlow deployment.
3. Click Renew & Restart.

   The UI switches to the **KPIs and Alerts** pane where you can monitor as your deployment restarts and the new certificate or certificates become available.

What to do next

If you have renewed the NiFi Inbound SSL Context Service:

You have to take no further action.

If you have renewed the Client SSL Context:

After your Cloudera DataFlow deployment has restarted, you switch to the NiFi Configuration pane to download the Client Certificate and the Client Private Key. You can then add these to your client.

Reassigning an inbound connection endpoint to a different project

Learn how to reassign an inbound connection to another project.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

About this task

You cannot reassign an inbound connection that is currently used by a deployment. You have to terminate the deployment using it making sure that the Delete assigned endpoint hostname option is not selected before you can reassign it to a different project.

Procedure

1. Select the **Inbound Connection** that you want to reassign.

   Note: You can reassign multiple items simultaneously by selecting the checkboxes in front of them.

2. Click Reassign.

   If the inbound connection is not used by any deployment, the **Reassign Resource** modal opens.

3. Select a Project and click Reassign.
4. Click Apply Changes.

Custom NAR configs

On the Custom NAR Configs tab you can reassign and validate Custom NAR configs.

Validating a custom NAR config

You can validate custom NAR files and their locations in the Resources view.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Select the **Custom NAR** that you want to validate.
2. Click Options Validate .

   Cloudera DataFlow displays a message about the validity of the NAR config and its storage location.

Reassigning a custom NAR config to a different project

Learn how to reassign a custom NAR config to another project.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

About this task

You cannot reassign an inbound connection that is currently used by a deployment. You have to teminate the deployment using it making sure that the Delete assigned endpoint hostname option is not selected before you can reassign it to a different project.

Procedure

1. Select the **Custom NAR** that you want to reassign.

   Note: You can reassign multiple items simultaneously by selecting the checkboxes in front of them.

2. Click Reassign.

   If the custom NAR config is not used by any deployment, the **Reassign Resource** modal opens.

3. Select a Project and click Reassign.
4. Click Apply Changes.

Custom Python configs

On the Custom Python Configs tab you can reassign and validate Custom Python configs.

Validating a custom Python config

You can validate custom Python configs and their storage locations in the Resources view.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Select the **Custom Python Config** that you want to validate.
2. Click Options Validate .

   Cloudera DataFlow displays a message about the validity of the Python config and its storage location.

Reassigning a custom Python config to a different project

Learn how to reassign a custom Python config to another project.

Before you begin

- Make sure that you have DFDeveloper permission to perform this task. For information on account and resource roles, see Cloudera DataFlow Authorization.

Procedure

1. Select the **Custom Python Config** that you want to reassign.

   Note: You can reassign multiple items simultaneously by selecting the checkboxes in front of them.

2. Click Reassign.

   If the custom Python config is not used by any deployment, the **Reassign Resource** modal opens.

3. Select a Project and click Reassign.
4. Click Apply Changes.

Cloudera DataFlow

CDF Security

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera DataFlow security....4

Cloudera DataFlow authorization....5

Cloudera DataFlow authorization process....7

Authorize user to manage flow definitions in Cloudera DataFlow Catalog....9
Authorize user to manage lifecycle of a Cloudera DataFlow environment....9
Authorize user to develop flows in a Cloudera DataFlow environment....9
Authorize user to manage deployments in a Cloudera DataFlow environment....10
Authorize user to manage deployments and deploy flows in a Cloudera DataFlow environment....11
Authorize user to view and monitor deployments in a Cloudera DataFlow environment....11
Authorize user to create and manage Projects for a tenant....12
Authorize user to create and manage their Projects in a Cloudera DataFlow environment....12

Projects....13

Creating a Project....14
Editing a Project....14
Managing access to a Project....15
Adding resources to a Project....15
Managing and reassigning resources to a Project....16

Cloudera DataFlow security

As a security administrator, understand the out-of-the box security features as well as the tasks that you need to perform to authorize Cloudera Public Cloud users.

Cloudera Public Cloud provides the following security features:

- User authentication
- Role-based user authorization
- Data encryption

You must assign roles to Cloudera Public Cloud users based on their Cloudera DataFlow job responsibilities.

User authentication

Users are automatically authenticated through the Cloudera Public Cloud identity provider. For more information, see *Cloudera Public Cloud Security Overview*.

Role-based user authorization

Cloudera DataFlow roles allow you to set user permissions. Through these permissions you can define actions a user or group is allowed to perform and you can also scope the resources a they can perform those actions on.

For more information on the Cloudera DataFlow roles, see *Cloudera DataFlow Authorization*.

Data encryption

Cloudera Public Cloud encrypts data at rest and in motion.

Data at rest

When you import a flow definition, the flow definition is encrypted and stored in the Cloudera DataFlow Catalog. Cloudera DataFlow Catalog is a service that enables you to manage flow definitions centrally.

Note:

- Logs and the NiFi FlowFile, Content, and Provenance repositories are stored in your cloud account. Access to the logs and repositories is based on user permissions.
- Before you import a flow definition:
  - Verify that the flow definition will not write secure information to the logs
  - Parameterize sensitive configuration

Data in transit

When you deploy a flow, the encrypted flow definition is transferred to your cloud account where it is decrypted and deployed. During a flow deployment, data can be transferred between servers, systems, applications, and users. Every transfer is a secure and trusted exchange through TLS. Through cryptographic protocols, TLS encrypts and authenticates:

- The flow definition from the Cloudera DataFlow Catalog to a Cloudera DataFlow environment.
- The connections from the user's browser to the Cloudera DataFlow service.
- The communication between the Cloudera DataFlow service and other services in the Cloudera Public Cloud Control Plane.
- The communication between the Cloudera Public Cloud Control Plane services and environment services.
- The communication between the Cloudera DataFlow service, flows, and shared services within an environment.

Related Concepts

Cloudera DataFlow authorization

Related Information

Cloudera

Cloudera DataFlow authorization

As a security administrator with the PowerUser role, you must assign one or more Cloudera DataFlow roles to a user to allow the user to perform tasks in the Cloudera DataFlow Catalog, the Cloudera DataFlow environment, and the Apache NiFi cluster.

Review each of the following Cloudera DataFlow roles to determine which roles a user requires:

DFCatalogViewer

This role allows a user to perform the following tasks:

- View and search flow definitions in the Cloudera DataFlow Catalog
- View and search ReadyFlows in the ReadyFlow Gallery

DFCatalogAdmin

This role allows a user to centrally manage flow definitions and ReadyFlows.

A user with the DFCatalogAdmin role can perform the following tasks in the Cloudera DataFlow Catalog:

- View and search flow definitions
- Import flow definitions
- Import new versions of existing flow definitions
- View flow definition details
- Delete flow definitions

A user with the DFCatalogAdmin role can perform the following tasks in the ReadyFlow Gallery:

- View and search ReadyFlows
- Add ReadyFlows to the Catalog

DFAdmin

This role allows a user to manage the lifecycle of a Cloudera DataFlow environment.

Note: The DFAdmin role does not act as a superset of the DFFlowAdmin and DFFlowUser roles. Users that have only the DFAdmin role cannot view or create flow deployments.

A user with the DFAdmin role can perform the following tasks:

- Enable a Cloudera environment for Cloudera DataFlow
- View an environment
- View environment details and Cloudera DataFlow Settings
- Disable an environment
- Update resource roles
- Grant remote access to an environment
- Revoke remote access to an environment
- Terminate all running deployments when disabling the Cloudera DataFlow service for an environment.

A user with the DFAdmin and PowerUser roles can perform the following additional tasks:

- Assign the DFAdmin role to other users

- Revoke the DFAdmin role from other users

DFFlowAdmin

This role allows a user to manage flow deployments in a Cloudera DataFlow environment. In addition, this role together with either the DFCatalogViewer or DFCatalogAdmin role allows the user to deploy flows.

A user with the DFFlowAdmin role alone can perform the following tasks:

- View a deployment on the dashboard
- View deployment details and settings
- View a deployment in NiFi
- Stop a deployment in NiFi
- Suspend a deployment in NiFi
- Terminate a deployment in NiFi
- Change NiFi version

A user with the DFFlowAdmin and PowerUser roles can perform the following additional tasks:

- Assign the DFFlowAdmin and DFFlowUser roles to other users
- Revoke the DFFlowAdmin and DFFlowUser roles from other users

A user with the DFFlowAdmin role and either the DFCatalogViewer or DFCatalogAdmin role can perform the following additional task:

- Deploy flow definitions

DFFlowUser

This role allows a user to view, search, and monitor flow deployments in a Cloudera DataFlow environment.

A user with the DFFlowUser role can perform the following tasks:

- View a deployment on the dashboard
- View deployment details and settings
- View a deployment in NiFi

DFDeveloper

This role allows a user to view, search, create, and manage drafts in Flow Designer on environment level.

A user with the DFDeveloper role can perform the following tasks:

- Create a new draft
- View all drafts in an environment
- View all draft details and settings in an environment
- Update all drafts in an environment
- Start and end Test Sessions for any draft in an environment
- Delete any draft in an environment

A user with the DFDeveloper and the DFCatalogViewer or DFCatalogAdmin roles can perform the following additional tasks:

- Create drafts from flow definitions in the Cloudera DataFlow Catalog
- Create drafts from ReadyFlows in the ReadyFlow Gallery
- Publish drafts as flow definitions to the Catalog (DFCatalogAdmin only)

DFProjectsAdmin

This role allows a user to centrally manage Projects.

A user with the DFProjectsAdmin role can perform the following tasks in a tenant:

- View and create Projects.
- Edit users and groups assigned to Projects.
- Delete Projects.

DFProjectCreator

This role allows a user to create and manage Projects on environment level.

A user with the DFProjectCreator role can perform the following tasks:

- Create a Project.
- Edit a Project.
- Manage users and groups assigned to a Project.

For information on configuring Cloudera user management, see *User Management*.

Related Information

User Management

Cloudera DataFlow authorization process

As a security administrator with the Cloudera PowerUser role, understand the authorization process to allow Cloudera users to perform tasks.

The following diagram shows the Cloudera DataFlow authorization process:

Authorize user to manage flow definitions in Cloudera DataFlow Catalog

To authorize a user to import and manage flow definitions in the Catalog and add ReadyFlows to the Catalog, you must assign the DFCatalogAdmin role to the user.

Before you begin

- You must have PowerUser permission for the account where you want to perform this task.

Procedure

1. From the Cloudera Management Console, click User Management.
2. Enter the user's name in the Search field.
3. Select the user from the list.

   The **User** page appears.
4. Go to the Roles tab.
5. Select Update Roles.
6. On the **Update Roles** dialog, select the DFCatalogAdmin role along with any other role the user requires.
7. Click Update.

Authorize user to manage lifecycle of a Cloudera DataFlow environment

To authorize a user to enable, disable, and administer a given Cloudera environment for Cloudera DataFlow, assign the DFAdmin role to the user. This role also permits the user to grant and revoke the ability to access the Cloudera DataFlow Kubernetes API server.

Before you begin

- You must have PowerUser permission for the account where you want to perform this task.
- You must have Owner permission for the environment where you want to perform this task.
- You must have IAMViewer or IAMUser permission for the account where you want to perform this task.

Procedure

1. From the Cloudera Management Console, click Environments.

   The **Environment List** page appears.
2. Search for and select the environment you want the user to have rights to.

   The **Environment** page appears.
3. Click Actions Manage Access .

   The **Environment Access** page appears.
4. Find the user and click Update Roles.
5. From the **Update Roles** dialog, select the DFAdmin role and any other role the user requires.
6. Click Update Roles.

Authorize user to develop flows in a Cloudera DataFlow environment

To authorize a user to create and manage drafts in a Cloudera DataFlow environment, assign the DFFlowDeveloper role to the user.

Before you begin

- You must have PowerUser permission for the account where you want to perform this task.
- You must have Owner permission for the environment where you want to perform this task.
- You must have IAMViewer or IAMUser permission for the account where you want to perform this task.

About this task

Note: To create drafts from flow definitions in the Catalog and ReadyFlows in the ReadyFlow Gallery, users also need the DFFlowViewer or the DFFlowAdmin role. To publish drafts to the Catalog as new flow definitions or versions of existing flow definitions, users need the DFFlowAdmin role.

Procedure

1. From the Cloudera Management Console, click Environments.

   The **Environment List** page appears.
2. Search for and select the environment you want the user to have rights to.

   The **Environment** page appears.
3. Click Actions Manage Access .

   The **Environment Access** page appears.
4. Find the user and click Update Roles.
5. From the **Update Roles** dialog, select the DFFlowDeveloper role.
6. Click Update Roles.
7. Click Synchronize Users to propagate the user role changes you made to the environment.

   Note: It may take up to five minutes for the user role changes to propagate to the Environment.

Authorize user to manage deployments in a Cloudera DataFlow environment

To authorize a user to manage flow deployments in a Cloudera DataFlow environment, assign the DFFlowAdmin role to the user.

Before you begin

- You must have PowerUser permission for the account where you want to perform this task.
- You must have Owner permission for the environment where you want to perform this task.
- You must have IAMViewer or IAMUser permission for the account where you want to perform this task.

Procedure

1. From the Cloudera Management Console, click Environments.

   The **Environment List** page appears.
2. Search for and select the environment you want the user to have rights to.

   The **Environment** page appears.
3. Click Actions Manage Access .

   The **Environment Access** page appears.
4. Find the user and click Update Roles.
5. From the **Update Roles** dialog, select the DFFlowAdmin role.
6. Click Update Roles.

7. Click Synchronize Users to propagate the user role changes you made to the environment.

   Note: It may take up to five minutes for the user role changes to propagate to the Environment.

Authorize user to manage deployments and deploy flows in a Cloudera DataFlow environment

To authorize a user to manage flow deployments as well as deploy flows in a Cloudera DataFlow environment, you must assign the DFFlowAdmin role and either the DFCatalogAdmin or DFCatalogViewer role to the user.

Before you begin

- You must have PowerUser permission for the account where you want to perform this task.
- You must have IAMViewer or IAMUser permission for the account where you want to perform this task.

Procedure

1. From the Cloudera Management Console, click Environments.

   The **Environment List** page appears.
2. Search for and select the environment you want the user to have rights to.

   The **Environment** page appears.
3. Click Actions Manage Access .

   The **Environment Access** page appears.
4. Find the user and click Update Roles.
5. From the **Update Roles** dialog, select the DFFlowAdmin role.
6. Click Update Roles.
7. Click Synchronize Users to propagate the user role changes you made to the environment.

   Note: It may take up to five minutes for the user role changes to propagate to the Environment.
8. From the Cloudera Management Console, click User Management.
9. Enter the user's name in the Search field.
10. Select the user from the list.

    The **User** page appears.
11. Go to the Roles tab.
12. Select Update Roles.
13. On the **Update Roles** dialog, assign one of the following DFCatalog roles to the user:
    - If the user needs to deploy flow definitions in the environment, assign the DFCatalogViewer role.
    - If the user needs to manage flow definitions in the Cloudera DataFlow Catalog and deploy flow definitions in the environment, assign the DFCatalogAdmin role.
14. Click Update.

Authorize user to view and monitor deployments in a Cloudera DataFlow environment

To authorize a user to view, search, and monitor deployments in a Cloudera DataFlow environment, assign the DFFlowUser role to the user.

Before you begin

- You must have PowerUser permission for the account where you want to perform this task.
- You must have Owner permission for the environment where you want to perform this task.
- You must have IAMViewer or IAMUser permission for the account where you want to perform this task.

Procedure

1. From the Cloudera Management Console, click Environments.
   The **Environment List** page appears.
2. Search for and select the environment you want the user to have rights to.
   The **Environment** page appears.
3. Click Actions Manage Access .
   The **Environment Access** page appears.
4. Find the user and click Update Roles.
5. From the **Update Roles** dialog, select the DFFlowUser role and any other role the user requires.
6. Click Update Roles.
7. Click Synchronize Users to propagate the user role changes you made to the environment.

   Note: It may take up to five minutes for the user role changes to propagate to the Environment.

Authorize user to create and manage Projects for a tenant

To Authorize a user to create and delete Projects, edit Users and Groups assigned to Projects within a tenant, you must assign the DFProjectsAdmin role to the user.

Before you begin

You must have the Cloudera PowerUser role.

Procedure

1. From the Cloudera Management Console, click User Management.
2. Enter the user's name in the Search field.
3. Select the user from the list.
   The **User** page appears.
4. Go to the Roles tab.
5. Select Update Roles.
6. On the **Update Roles** dialog, select the DFProjectsAdmin role along with any other role the user requires.
7. Click Update.

Authorize user to create and manage their Projects in a Cloudera DataFlow environment

To authorize a user to create and manage their own Projects, edit Users and Groups assigned to Projects within an environment, you must assign the DFProjectCreator role to the user.

Before you begin

- You must have PowerUser permission for the account where you want to perform this task.

- You must have Owner permission for the environment where you want to perform this task.
- You must have IAMViewer or IAMUser permission for the account where you want to perform this task.

Procedure

1. From the Cloudera Management Console, click Environments.

   The **Environment List** page appears.
2. Search for and select the environment you want the user to have rights to.

   The **Environment** page appears.
3. Click Actions Manage Access .

   The **Environment Access** page appears.
4. Find the user and click Update Roles.
5. From the **Update Roles** dialog, select the DFProjectCreator role.
6. Click Update Roles.
7. Click Synchronize Users to propagate the user role changes you made to the environment.

   Note: It may take up to five minutes for the user role changes to propagate to the Environment.

Projects

Projects provide fine grained role-based access control over resources within an environment.

Projects are a layer of role-based access control in CDF. You may consider them as logical containers that you can use to govern access to resources within an environment.

Resources (flow deployments, flow drafts, Inbound Connections, and custom NARs) can have one of two states within an environment:

- Unassigned - the resource is not assigned to a particular Project, it is freely accessible for every user or group with the appropriate user role for that environment.
- Assigned to a Project - the resource is only accessible to users or groups with access to that particular Project. This assignment is exclusive, you cannot assign the same resource to more than one Project at a time.

The roles associated with projects are additive, they work in conjunction with user permissions controlling the types of actions a user or group is allowed to perform in a Cloudera Public Cloud environment.

Projects introduces the following user roles:

DFProjectsAdmin

This role enables users to manage all Projects in all environments within an tenant.

DFProjectCreator

This role enables users to create Projects in an Environment. When creating a Project, they automatically become the DFProjectAdmin in that Project.

DFProjectAdmin

This role is automatically assigned to users with DFProjectCreator role upon creating a Project. DFProjectAdmins can add/remove users and groups, changing user and group roles (DFProjectAdmin or DFProjectMember), modify the Project name and description, and delete the Project.

DFProjectMember

This role enables users to access resources assigned to a Project and perform actions on them that are allowed by their other user roles.

For example, users with DFFlowUser role in ‘Environment_A’, and DFProjectMember role in ‘Project_Alpha’ are able to view flow deployments ‘Deployment-1’ that is assigned to ‘Project_Alpha’ and ‘Deployment-2’ that is currently unassigned. They cannot view ‘Deployment-3’ that is assigned to Project_Beta, just as they do not see other resource types (in this example draft flows ‘Draft-1’, ‘Draft-2’, and ‘Draft-3’) regardless of their assignment status. That is, the DFFlowUser role defines the types of actions they can perform in the environment, and the ‘DFProjectMember’ role defines the resources within that environment they can perform those actions on.

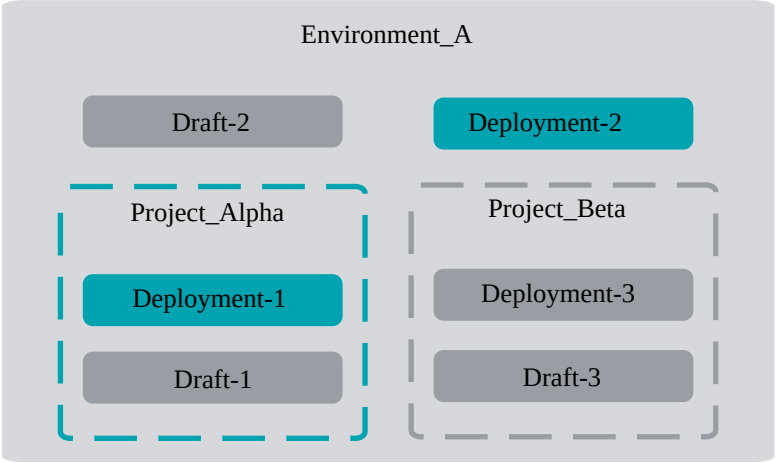

Creating a Project

You can create a Project to limit the visibility of drafts, deployments, Inbound Connections, and custom NAR Configurations within an Environment.

Before you begin

- Make sure that you have DFProjectCreator permission to perform this task. For information on account and resource roles, see Cloudera DataFlow authorization.
- Cloudera DataFlow must be enabled for the Environment.

Procedure

1. Go to DataFlow Projects .
2. Click Create Project.
3. Select Target Workspace.
4. Provide Project Name.

   Note: Project names need to be unique within a Workspace. The Deployment wizard indicates whether a name is valid by displaying a green check below the Project Name text box.

5. Provide a Project Description.
6. Click Create.

   Note: After creating a Project, it needs to be synchronized before resources can be assigned. This may take a few minutes and during the synchronization, the new Project is not available in the Workspace Resources view yet.

Editing a Project

You can change the name and the description of Projects where you have DFProjectAdmin permission.

Before you begin

- Make sure that you have DFProjectAdmin permission for the project you want to edit to perform this task.

Procedure

1. Go to DataFlow Projects .
2. Select the Project you want to edit and click ⋮ More Edit Project .
3. Modify the Project name and/or Project description.
4. Click Apply Changes.

Managing access to a Project

You can grant and revoke access to Projects where you have DFProjectAdmin permission.

Before you begin

- Make sure that you have DFProjectAdmin permission for the project you want to manage to perform this task.
- Make sure that you have IAMViewer or IAMUser permission for the tenant where you want to perform this task. Without one of those, users and groups will not be visible to you.

Procedure

1. Go to DataFlow Projects .
2. Select the project where you want to manage access rights and click ⋮ More Manage Access .

To grant roles:

3. Start typing the name of the user or group you want to add and select them from the list.
4. Select the Resource Roles you want to grant.
5. Click Update Roles.
6. Click Synchronize Users.

To revoke roles:

7. Click Update Roles next to the name of the user or group for which you want to revoke roles.
8. Unselect the Resource Roles you want to revoke.
9. Click Update Roles.
10. Click Synchronize Users.

Adding resources to a Project

You can add resources to Projects where you have DFProjectMember permission.

Before you begin

- Make sure that you have DFProjectMember permission for the project where you want to add resources to perform this task.

Procedure

1. Go to DataFlow Projects .

2. Select the project where you want to add resources and click ⋮ More View Workspace Manage Project Resources .

   The **Workspace Resources** view shows you all resources across Projects that you have access to.
3. Select the type of resource you want to reassign using the resource type tabs.
4. Select the resource you want to reassign and click ⋮ More Reassign .

   Tip: Select several resources using the checkboxes and click the Reassign button for bulk resource reassignment.
5. In the **Reassign Resource** pop-up select an available Project or select Unassigned.
6. Click Reassign.

Managing and reassigning resources to a Project

You can manage and reassign resources to Projects where you have DFProjectMember permission.

Before you begin

- You must have DFProjectMember permission for the Project where you want to perform this task.

Procedure

1. Go to DataFlow Projects .
2. Select the project where you want to manage resources and click ⋮ Options View Project Resources - Manage & Assign .

   The **Workspace Resources** view shows you resources filtered for the Project you selected.

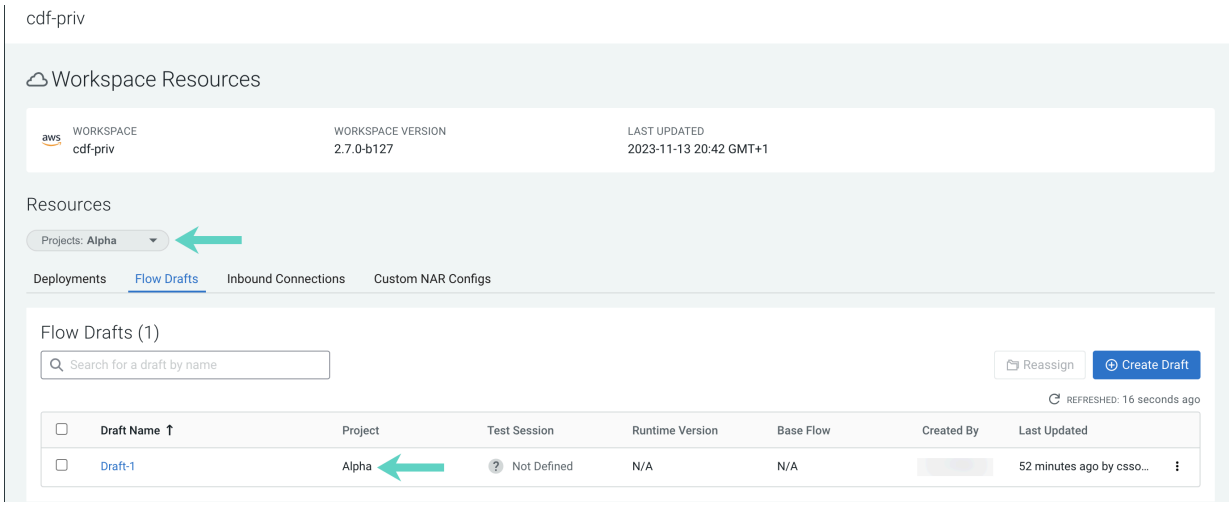

   Note: After creating a Project, it needs to be synchronized before resources can be assigned. This may take a few minutes and during the synchronization, the new Project is not available in the Workspace Resources view yet.
3. Select the type of resource you want to reassign using the resource type tabs.

4. Select the resource you want to reassign and click ⋮ More Reassign .

   Tip: Select several resources using the checkboxes and click the Reassign button for bulk resource reassignment.

5. In the **Reassign Resource** pop-up select an available Project or select Unassigned.
6. Click Reassign.

Cloudera DataFlow

Service upgrade

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Pre-upgrade information 4

Update cdp-liftie-instance-profile 4

Upgrade a Cloudera DataFlow service 7

Troubleshooting Cloudera DataFlow upgrade errors 9

Pre-upgrade information

Learn about tasks you need to perform and information you need to consider before starting a Cloudera DataFlow service upgrade.

Important: If you are upgrading on AWS to Cloudera DataFlow 2.3.0 or higher from release 2.2.0 or lower and you use a pre-created IAM policy, you need to update the cdp-liftie-instance-profile before the upgrade. For more information, see Update cdp-liftie-instance-profile.

Note: In Cloudera DataFlow versions 2.4.0 and lower, ZooKeeper may lose component state information under certain conditions due to its previous configuration to use ephemeral storage. If a ZooKeeper pod fails, it retains component state information as long as it is rescheduled to the same Kubernetes (K8s) node, as it can reattach to storage already present there. If the ZooKeeper pod gets rescheduled to a different K8s node, existing storage is not moved with it, therefore state information is lost. If your use case requires that component state information is retained even during an upgrade, contact Cloudera Support to provide you with a patch that you need to apply to every deployment before starting the upgrade.

- You trigger an upgrade for a selected Cloudera DataFlow environment. Upgrades move from an older version to the latest version available.
- Once you trigger the upgrade, the service enters UPGRADING state and is upgraded to the latest supported Cloudera DataFlow and Kubernetes versions. Kubernetes upgrade can take up to an hour to complete. During the upgrade service actions are restricted, that is, you are not able to create or manage deployments in the given Cloudera DataFlow service.
- As part of a Cloudera DataFlow Service (Environment) upgrade, all existing deployments are also upgraded.
- To keep existing Cloudera DataFlow deployments working, you may choose not to upgrade the NiFi version of existing deployments during the upgrade. This selection applies to all existing deployments.

  Note: The option to skip NiFi version upgrade is not always available: NiFi version upgrade is mandatory if the upgrade wizard determines that the new workload version is not compatible with the current NiFi version running in any deployment. The same automatism prevents you from accidentally downgrading NiFi below the minimum supported version later.

  You can manually upgrade NiFi for individual deployments later.
- If upgrading a deployment fails, all components are rolled back to their original state and it causes the overall Cloudera DataFlow service upgrade to be rolled back as well.

Update cdp-liftie-instance-profile

When upgrading to Cloudera DataFlow 2.3.0 or higher from release 2.2.0-h4 or lower on AWS and you use a pre-created IAM policy, you need to update the cdp-liftie-instance-profile before the upgrade. Without these policy changes, volumes provisioned by a pre-2.3.0 Cloudera DataFlow version will be left behind if the corresponding deployment is terminated or the data flow is disabled after the upgrade.

About this task

These updates are necessary because Cloudera DataFlow has migrated to use EKS k8s 1.23 and CSI EBS provisioner. This storage provisioner requires updated policies to allow the creation and deletion of volumes from within the EKS cluster.

Update the cdp-liftie-instance-profile using either the AWS UI or CLI.

Procedure

1. Add the ebs-csi policy to the cdp-liftie-instance-profile IAM profile.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:CreateSnapshot",
                "ec2:AttachVolume",
                "ec2:DetachVolume",
                "ec2:ModifyVolume",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInstances",
                "ec2:DescribeSnapshots",
                "ec2:DescribeTags",
                "ec2:DescribeVolumes",
                "ec2:DescribeVolumesModifications"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": [
                        "CreateVolume",
                        "CreateSnapshot"
                    ]
                }
            },
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:snapshot/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "ec2:DeleteTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:snapshot/*"
            ],
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/ebs.csi.aws.com/cluster": "true"
                }
            },
            "Action": [
                "ec2:CreateVolume"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
```

```
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/CSIVolumeName": "*"
                }
            },
            "Action": [
                "ec2:CreateVolume"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
                }
            },
            "Action": [
                "ec2:DeleteVolume"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/CSIVolumeName": "*"
                }
            },
            "Action": [
                "ec2:DeleteVolume"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/kubernetes.io/created-for/pvc/name":
"*"
                }
            },
            "Action": [
                "ec2:DeleteVolume"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/CSIVolumeSnapshotName": "*"
                }
            },
            "Action": [
                "ec2:DeleteSnapshot"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
```

```
                }
            },
            "Action": [
                "ec2:DeleteSnapshot"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

2. Add the efs-csi policy to the cdp-liftie-instance-profile IAM profile.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticfilesystem:DescribeAccessPoints",
                "elasticfilesystem:DescribeFileSystems",
                "elasticfilesystem:DescribeMountTargets"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/efs.csi.aws.com/cluster": "true"
                }
            },
            "Action": [
                "elasticfilesystem:CreateAccessPoint"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/efs.csi.aws.com/cluster": "true"
                }
            },
            "Action": [
                "elasticfilesystem:DeleteAccessPoint"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Upgrade a Cloudera DataFlow service

You can upgrade your Cloudera DataFlow service together with all DataFlow deployments in that environment with the push of a button.

Procedure

1. In Cloudera DataFlow, select Environments from the left navigation pane.

2. Select the environment you want to upgrade and click Manage DataFlow.
3. Click Actions Upgrade DataFlow .

Note: If the upgrade wizard determines that the NiFi version of your Cloudera DataFlow deployments is compatible with the Cloudera DataFlow service after the upgrade, you have the option to skip upgrading NiFi in current deployments. In this case you can upgrade NiFi manually later.

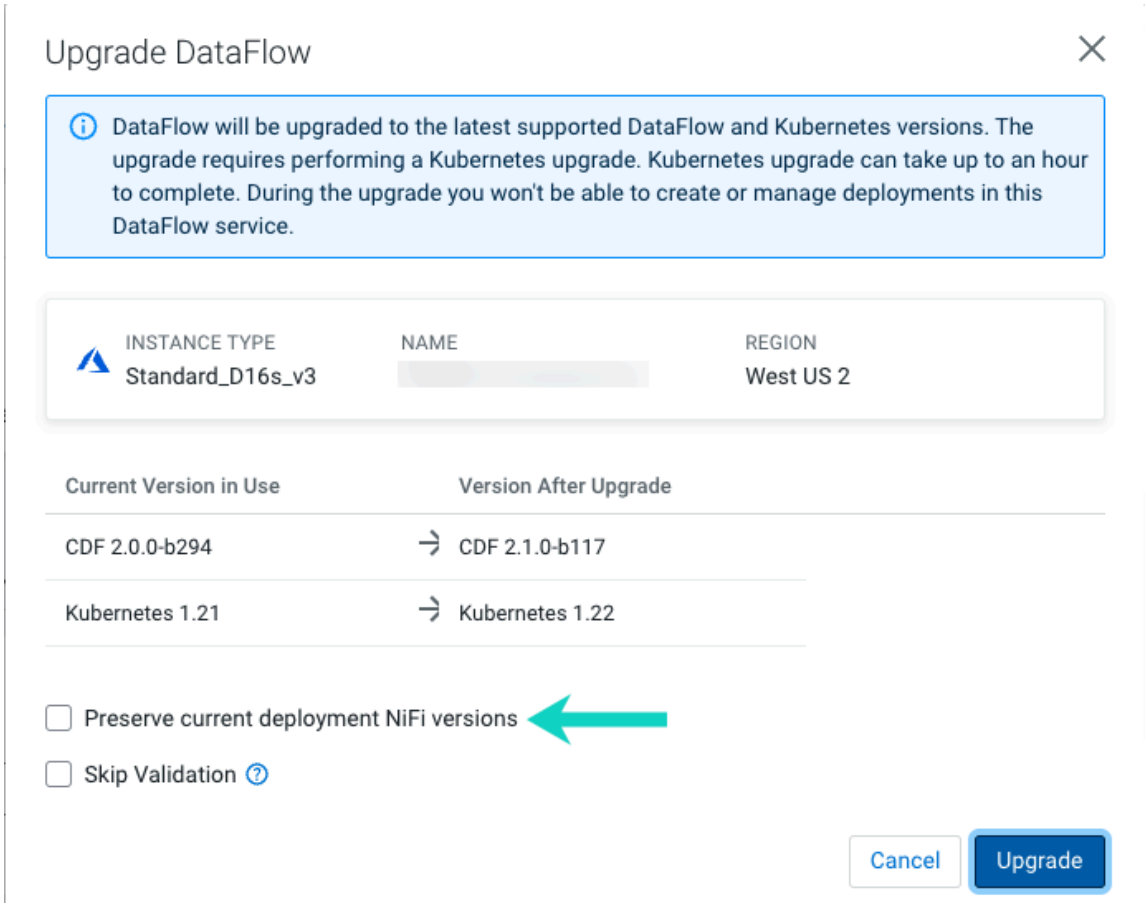

4. Click Upgrade.
   The environment status changes to Upgrading and you see a similar message under Event History:

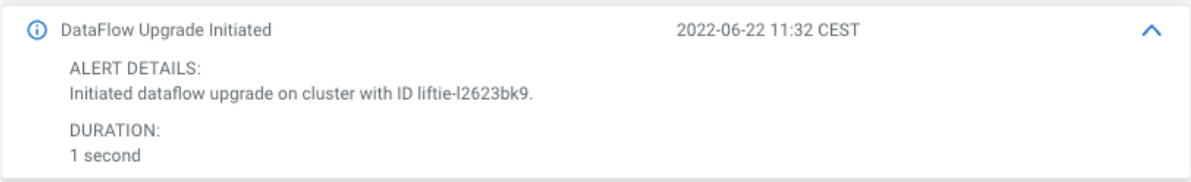

5. You can monitor the upgrade procedure in Event History. Click Manage DataFlow then select Alerts.

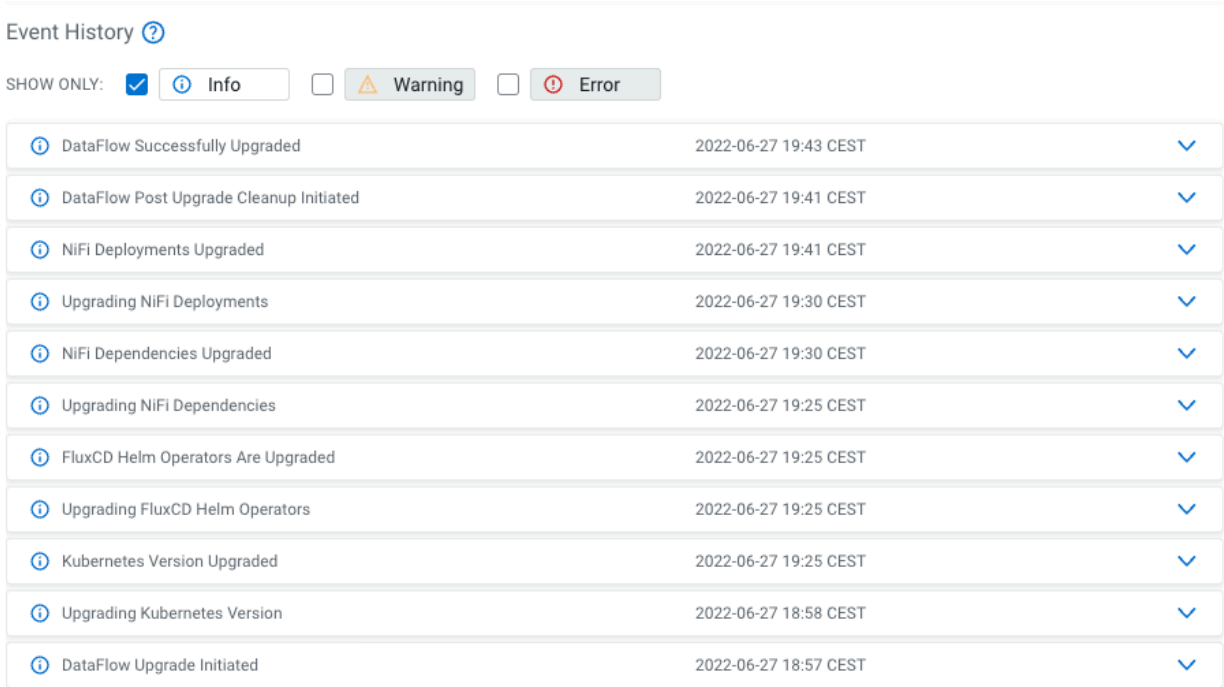

Results

The message DataFlow Successfully Upgraded in **Event History** and the Status of the service changing from Upgrading to Good Health on the **Environments** page signal that the upgrade has ended. Depending on the number of running nodes in the cluster, the upgrade can take more than an hour to complete.

What to do next

If you had restricted policies in place before the upgrade, run the

```
aws ec2 describe-volumes --profile <aws-profile> --query
                 'Volumes[?Tags[?contains(Key,`kubernetes.io/cluster/liftie-
`)] &&
                 State==`available`].[VolumeId,State,Tags[?contains(Key,`k
ubernetes.io/cluster/liftie-`)].Key|[0]]'
```

command to confirm that no unused volumes were left behind.

Troubleshooting Cloudera DataFlow upgrade errors

Learn how to recognize and address common errors with your Cloudera DataFlow service and flow deployment upgrades.

Setting up `kubectl` to connect to the Kubernetes cluster

It is helpful to have access to the Kubernetes cluster using command line tools such as `kubectl` when you are troubleshooting deployment or upgrade failures. To set up `kubectl` access, follow these steps:

1. In , from the Environments page, select the service for which you want to add or remove user access.
2. Click Manage DataFlow.
3. From the Actions menu, click Manage Kubernetes API Server User Access.
4. Add the AWS IAM role that you will authenticate as to the list of authorized users for DataFlow by entering the ARN in the Add User dialog.

5. Use the Download Kubeconfig action to retrieve the kubeconfig file for connecting to your cluster.
6. Set up your `kubectl` to use the downloaded kubeconfig file:

```
export KUBECONFIG=[***PATH/TO/DOWNLOADED/KUBECONFIG/FILE***]
```

7. Run kubectl get ns and validate that your output looks similar to:

```
NAME                        STATUS   AGE
cadence                     Active   37h
cert-manager                Active   37h
cfm-operator-system         Active   37h
default                     Active   38h
dfx-dev-environment-ns      Active   36h
dfx-idbrokers3-ns           Active   36h
dfx-kafkatos3-ns            Active   22h
dfx-local                   Active   37h
dfx-ops                     Active   37h
kube-node-lease             Active   38h
kube-public                 Active   38h
kube-system                 Active   38h
liftie                      Active   37h
logging                     Active   37h
monitoring                  Active   37h
nfs-provisioner-system      Active   37h
nginx-ingress               Active   37h
prometheus-operator-system  Active   37h
```

With `kubectl` being set up correctly, you are able to access NiFi and other logs directly through the CLI.

Understanding upgrade failures

There are various reasons that may prevent an upgrade from happening. This document covers the following scenarios:

- The upgrade fails upon initiation.
- The upgrade starts but fails, and rollback is possible.
- The upgrade starts but fails and rollback is not possible.

Upgrade fails on initiation

Symptom: When the upgrade fails to start, the status of the environment does not change to Upgrading and you receive an error message stating the cause of the failure.

Possible causes: The service is in one of the following states:

- The service is in one of the following states:
  - Failed to Update
  - Failed to Update NiFi version
  - Any transitional / in-progress state
- Any of the deployments is in one of the following states:
  - New
  - Failed to Update
  - Failed to Update NiFi version
  - Any transitional / in progress state

How to fix it: Fix the error indicated in the error message, then retry the upgrade. If the upgrade still fails, contact Cloudera Support.

Upgrade fails and rollback is possible

Symptom: Upgrade rollback process starts. If the rollback is successful, the **Status** of the service returns to **Good Health**, Cloudera DataFlow and NiFi versions return to their pre-upgrade values. If the rollback fails, the **Status** of both the Cloudera DataFlow service and running deployments becomes **Failed to Upgrade**. The order in which this transition happens depends on where the rollback actually failed.

How to fix it: In the event of an upgrade failure, Cloudera DataFlow determines if rollback is possible and automatically initiates it. No user intervention is required.

If the rollback is successful:

1. After the rollback ends, from the Environments page, select the Cloudera DataFlow service you were trying to upgrade and click the Alerts tab. Check Event History to see if there is anything obvious in there or something that narrows down the possible cause of the failure.
2. From the Deployments page select running deployments that failed to upgrade and check Alerts>Event History to see if there is anything obvious in there or something that narrows down the possible cause of the failure.
3. Check the K8s cluster using CLI and try to fix errors based on the information you obtained from Event History in the previous steps.
4. Retry the upgrade.
5. If the upgrade still fails contact Cloudera Support.

If the rollback fails:

The status of both the service and the running deployments changes to Failed to Update. The order in which they transition depends on where the rollback actually failed.

Note: A Cloudera DataFlow upgrade can only be rolled back if Kubernetes (K8s) was not upgraded.

Upgrade fails and rollback is not possible

Symptom: The status of the service, and shortly after the deployments, changes to Failed to Upgrade. The status icon on the service changes to Bad Health.

How to fix it:

1. If K8s upgrade failed, check the Event History if there is anything obvious in there.
2. Check on the cloud provider side whether there is anything obvious as a reason for the failure. For example, check activity logs and node pools on Azure, or EKS cluster on AWS.
3. If the K8s upgrade succeeded, but the upgrade failed somewhere later, then the previous point in the "Upgrade fails and rollback is possible" can follow here (check the events and then the K8s cluster).
4. If you managed to pinpoint and fix the possible cause of the failure, retry the upgrade.
5. If you did not find the cause of the failure, or the upgrade still fails upon retry, contact Cloudera Support.

Retrying an upgrade

1. On the Cloudera DataFlow Overview select Environments from the left navigation pane.
2. Select the service where you want to retry the upgrade and click Manage DataFlow.
3. Click Actions Retry Upgrade DataFlow .

Cloudera DataFlow

Support Matrix

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache NiFi endpoints **4**
Supported NiFi processors 4
Removed NiFi processors 12
Supported NiFi controller services 13
Removed NiFi controller services 16
Supported NiFi reporting tasks 17

Feature support matrix **18**

Cloudera DataFlow Functions **18**

Apache NiFi endpoints

Learn about the Apache NiFi processors, controller services, and reporting tasks supported by Cloudera that you can use with Cloudera DataFlow.

Supported NiFi processors

Learn about the NiFi processors supported by Cloudera that you can use with Cloudera DataFlow.

Additional processors are developed and tested by the Cloudera community but are not officially supported by Cloudera. Processors are excluded for a variety of reasons, including insufficient reliability or incomplete test case coverage, declaration of non-production readiness by the community at large, and feature deviation from Cloudera best practices. Do not use these features in your production environments.

NiFi 1.27

AttributesToJSON	ListenSMTP
Base64EncodeContent	ListenSyslog
CalculateParquetOffsets	ListenTCP
CalculateParquetRowGroupOffsets	ListenTCPRecord
CalculateRecordStats	ListenTrapSNMP
CaptureChangeDebeziumDB21	ListenUDP
CaptureChangeDebeziumMySQL1	ListenUDPRecord
CaptureChangeDebeziumOracle1	ListenWebSocket
CaptureChangeDebeziumPostgreSQL1	ListFile
CaptureChangeDebeziumSQLServer1	ListFTP
CaptureChangeMySQL	ListGCSBucket
CompareFuzzyHash	ListGoogleDrive
CompressContent	ListHDFS
ConnectWebSocket	ListS3
ConsumeAMQP	ListSFTP
ConsumeAzureEventHub	ListSmb
ConsumeElasticsearch	LogAttribute
ConsumeEWS	LogMessage
ConsumeGCPubSub	LookupAttribute
ConsumeGCPubSubLite	LookupRecord
ConsumeIMAP	MergeContent
ConsumeJMS	MergeRecord
ConsumeKafka_2_6	ModifyBytes
ConsumeKafka2CDP	ModifyCompression
ConsumeKafka2RecordCDP	ModifyHTMLElement
ConsumeKafkaRecord_2_6	MonitorActivity
ConsumeKinesisStream	MoveAzureDataLakeStorage
ConsumeMQTT	MoveHDFS
ConsumePOP3	Notify
ConsumeTwitter	PackageFlowFile
ConsumeWindowsEventLog	PaginatedJsonQueryElasticsearch
ControlRate	ParseCEF

ConvertAvroToJSON
ConvertAvroToParquet
ConvertCharacterSet
ConvertExcelToCSVProcessor
ConvertJSONToSQL
ConvertProtobuf
ConvertRecord
CountText
CreateHadoopSequenceFile
CryptographicHashAttribute
CryptographicHashContent
DebugFlow
DecryptContent
DecryptContentAge
DecryptContentCompatibility
DecryptContentPGP
DeduplicateRecord
DeleteAzureBlobStorage
DeleteAzureBlobStorage_v12
DeleteAzureDataLakeStorage
DeleteByQueryElasticsearch
DeleteCDPObjectStore
DeleteDynamoDB
DeleteGCSObject
DeleteGridFS
DeleteHBaseCells
DeleteHBaseRow
DeleteHDFS
DeleteMongo
DeleteRethinkDB
DeleteS3Object
DeleteSQS
DetectDuplicate
DistributeLoad
DuplicateFlowFile
EncodeContent
EncryptContent
EncryptContentAge
EncryptContentPGP
EnforceOrder
EvaluateJsonPath
EvaluateXPath
EvaluateXQuery
ExecuteGraphQuery
ExecuteGraphQueryRecord
ExecuteGroovyScript
ExecuteInfluxDBQuery
ExecuteProcess
ExecuteScript
ExecuteSparkInteractive
ParseEvtx
ParseNetflowv5
ParseSyslog
ParseSyslog5424
PartitionRecord
PostHTTP
PostSlack
PublishAMQP
PublishGCPubSub
PublishGCPubSubLite
PublishJMS
PublishKafka_2_6
PublishKafka2CDP
PublishKafka2RecordCDP
PublishKafkaRecord_2_6
PublishMQTT
PublishSlack
PutAccumuloRecord
PutAzureBlobStorage
PutAzureBlobStorage_v12
PutAzureCosmosDBRecord
PutAzureDataLakeStorage
PutAzureEventHub
PutAzureQueueStorage
PutAzureQueueStorage_v12
PutBigQuery
PutBigQueryBatch
PutBigQueryStreaming
PutBoxFile
PutCassandraQL
PutCassandraRecord
PutCDPObjectStore
PutClouderaHiveQL
PutClouderaHiveStreaming
PutClouderaORC
PutCloudWatchMetric
PutCouchbaseKey
PutDatabaseRecord
PutDistributedMapCache
PutDropbox
PutDynamoDB
PutDynamoDBRecord
PutElasticsearchHttp
PutElasticsearchHttpRecord
PutElasticsearchJson
PutElasticsearchRecord
PutEmail
PutFile
PutFTP
PutGCSObject

ExecuteSQL
ExecuteSQLRecord
ExecuteStateless
ExecuteStreamCommand
ExtractAvroMetadata
ExtractCCDAAttributes
ExtractDocumentText
ExtractEmailAttachments
ExtractEmailHeaders
ExtractGrok
ExtractHL7Attributes
ExtractImageMetadata
ExtractMediaMetadata
ExtractRecordSchema
ExtractText
ExtractTNEFAttachments
FetchAzureBlobStorage
FetchAzureBlobStorage_v12
FetchAzureDataLakeStorage
FetchBoxFile
FetchCDPObjectStore
FetchDistributedMapCache
FetchDropbox
FetchElasticsearchHttp
FetchFile
FetchFTP
FetchGCSObject
FetchGoogleDrive
FetchGridFS
FetchHBaseRow
FetchHDFS
FetchParquet
FetchS3Object
FetchSFTP
FetchSmb
FilterAttribute
FlattenJson
ForkEnrichment
ForkRecord
FuzzyHashContent
GenerateFlowFile
GenerateRecord
GenerateTableFetch
GeoEnrichIP
GeoEnrichIPRecord
GeohashRecord
GetAsanaObject
GetAwsPollyJobStatus
GetAwsTextractJobStatus
GetAwsTranscribeJobStatus
PutGoogleDrive
PutGridFS
PutHBaseCell
PutHBaseJSON
PutHBaseRecord
PutHDFS
PutHive3QL
PutHive3Streaming
PutHTMLElement
PutIceberg
PutIcebergCDC1
PutInfluxDB
PutIoTDBRecord
PutJiraIssue
PutJMS
PutKinesisFirehose
PutKinesisStream
PutKudu
PutLambda
PutMongo
PutMongoRecord
PutORC
PutParquet
PutRecord
PutRethinkDB
PutS3Object
PutSalesforceObject
PutSFTP
PutSlack
PutSmbFile
PutSnowflakeInternalStage
PutSNS
PutSolrContentStream
PutSolrRecord
PutSplunk
PutSplunkHTTP
PutSQL
PutSQS
PutSyslog
PutTCP
PutUDP
PutWebSocket
PutZendeskTicket
QueryAirtableTable
QueryCassandra
QueryDatabaseTable
QueryDatabaseTableRecord
QueryDNS
QueryElasticsearchHttp
QueryIoTDBRecord

GetAwsTranslateJobStatus
GetAzureEventHub
GetAzureQueueStorage
GetAzureQueueStorage_v12
GetCouchbaseKey
GetDynamoDB
GetElasticsearch
GetFile
GetFTP
GetGcpVisionAnnotateFilesOperationStatus
GetGcpVisionAnnotateImagesOperationStatus
GetHBase
GetHDFS
GetHDFSEvents
GetHDFSFileInfo
GetHDFSSequenceFile
GetHTMLElement
GetHTTP
GetHubSpot
GetJiraIssue
GetJMSQueue
GetJMSTopic
GetMongo
GetMongoRecord
GetRethinkDB
GetSFTP
GetShopify
GetSmbFile
GetSNMP
GetSnowflakeIngestStatus
GetSolr
GetSplunk
GetSQS
GetTCP
GetTwitter
GetWorkdayReport
GetZendesk
HandleHttpRequest
HandleHttpResponse
HashAttribute
HashContent
IdentifyMimeType
InvokeAWSGatewayApi
InvokeGRPC
InvokeGRPC
InvokeHTTP
InvokeScriptedProcessor
ISPEnrichIP
JoinEnrichment
JoltTransformJSON
QueryRecord
QuerySalesforceObject
QuerySolr
QuerySplunkIndexingStatus
QueryWhois
RemoveRecordField
ReplaceText
ReplaceTextWithMapping
ResizeImage
RetryFlowFile
RouteHL7
RouteOnAttribute
RouteOnContent
RouteText
RunMongoAggregation
SampleRecord
ScanAccumulo
ScanAttribute
ScanContent
ScanHBase
ScriptedFilterRecord
ScriptedPartitionRecord
ScriptedTransformRecord
ScriptedValidateRecord
ScrollElasticsearchHttp
SearchElasticsearch
SegmentContent
SelectClouderaHiveQL
SelectHive3QL
SendTrapSNMP
SetSNMP
SignContentPGP
SplitAvro
SplitContent
SplitJson
SplitRecord
SplitText
SplitXml
SpringContextProcessor
StartAwsPollyJob
StartAwsTextractJob
StartAwsTranscribeJob
StartAwsTranslateJob
StartGcpVisionAnnotateFilesOperation
StartGcpVisionAnnotateImagesOperation
StartSnowflakeIngest
TagS3Object
TailFile
TransformXml
TriggerClouderaHiveMetaStoreEvent

JoltTransformRecord	TriggerHiveMetaStoreEvent
JSLTTransformJSON	UnpackContent
JsonQueryElasticsearch	UpdateAttribute
ListAzureBlobStorage	UpdateByQueryElasticsearch
ListAzureBlobStorage_v12	UpdateClouderaHiveTable
ListAzureDataLakeStorage	UpdateCounter
ListBoxFile	UpdateDatabaseTable
ListCDPObjectStore	UpdateDeltaLakeTable1
ListDatabaseTables	UpdateHive3Table
ListDropbox	UpdateJiraIssue
ListenBeats	UpdateRecord
ListenFTP	ValidateCsv
ListenGRPC	ValidateJson
ListenGRPC	ValidateRecord
ListenHTTP	ValidateXml
ListenNetFlow	VerifyContentMAC
ListenOTLP	VerifyContentPGP
ListenRELP	Wait
	YandexTranslate

Legend

1 - Technical Preview

NiFi 2.0 [Technical Preview]

AttributeRollingWindow	ListGCSBucket
AttributesToCSV	ListGoogleDrive
AttributesToJSON	ListHDFS
Bedrock	ListS3
CalculateParquetOffsets	ListSFTP
CalculateParquetRowGroupOffsets	ListSmb
CalculateRecordStats	LogAttribute
CaptureChangeDebeziumDB2	LogMessage
CaptureChangeDebeziumMySQL	LookupAttribute
CaptureChangeDebeziumOracle	LookupRecord
CaptureChangeDebeziumPostgreSQL	MergeContent
CaptureChangeDebeziumSQLServer	MergeRecord
CaptureChangeMySQL	ModifyBytes
ChunkData	ModifyCompression
ChunkDocument	MonitorActivity
CompressContent	MoveAzureDataLakeStorage
ConnectWebSocket	MoveHDFS
ConsumeAMQP	Notify
ConsumeAzureEventHub	PackageFlowFile
ConsumeElasticsearch	PaginatedJsonQueryElasticsearch
ConsumeGCPubSub	ParseCEF
ConsumeGCPubSubLite	ParseDocument
ConsumeIMAP	ParseEvtx
ConsumeJMS	ParseNetflowv5
ConsumeKafka_2_6	ParseSyslog

ConsumeKafka2CDP
ConsumeKafka2RecordCDP
ConsumeKafkaRecord_2_6
ConsumeKinesisStream
ConsumeMQTT
ConsumePOP3
ConsumeSlack
ConsumeTwitter
ConsumeWindowsEventLog
ControlRate
ConvertAvroToJSON
ConvertAvroToParquet
ConvertCharacterSet
ConvertJSONToSQL
ConvertProtobuf
ConvertRecord
CopyAzureBlobStorage_v12
CountText
CreateHadoopSequenceFile
CryptographicHashContent
DebugFlow
DecryptContent
DecryptContentAge
DecryptContentCompatibility
DecryptContentPGP
DeduplicateRecord
DeleteAzureBlobStorage_v12
DeleteAzureDataLakeStorage
DeleteByQueryElasticsearch
DeleteCDPObjectStore
DeleteDynamoDB
DeleteGCSObject
DeleteGridFS
DeleteHBaseCells
DeleteHBaseRow
DeleteHDFS
DeleteMongo
DeleteS3Object
DeleteSQS
DetectDuplicate
DistributeLoad
DuplicateFlowFile
EmbedData
EncodeContent
EncryptContentAge
EncryptContentPGP
EnforceOrder
EvaluateJsonPath
EvaluateXPath
EvaluateXQuery
ParseSyslog5424
PartitionCsv
PartitionDocx
PartitionHtml
PartitionPdf
PartitionRecord
PartitionText
PromptChatGPT
PublishAMQP
PublishGCPubSub
PublishGCPubSubLite
PublishJMS
PublishKafka_2_6
PublishKafka2CDP
PublishKafka2RecordCDP
PublishKafkaRecord_2_6
PublishMQTT
PublishSlack
PutAccumuloRecord
PutAzureBlobStorage_v12
PutAzureCosmosDBRecord
PutAzureDataExplorer
PutAzureDataLakeStorage
PutAzureEventHub
PutAzureQueueStorage_v12
PutBigQuery
PutBoxFile
PutCassandraQL
PutCassandraRecord
PutCDPObjectStore
PutChroma
PutClouderaHiveQL
PutClouderaHiveStreaming
PutClouderaORC
PutCloudWatchMetric
PutCouchbaseKey
PutDatabaseRecord
PutDistributedMapCache
PutDropbox
PutDynamoDB
PutDynamoDBRecord
PutElasticsearchJson
PutElasticsearchRecord
PutEmail
PutFile
PutFTP
PutGCSObject
PutGoogleDrive
PutGridFS
PutHBaseCell

ExecuteGraphQuery
ExecuteGraphQueryRecord
ExecuteGroovyScript
ExecuteProcess
ExecuteScript
ExecuteSparkInteractive
ExecuteSQL
ExecuteSQLRecord
ExecuteStateless
ExecuteStreamCommand
ExtractAvroMetadata
ExtractDocumentText
ExtractEmailAttachments
ExtractEmailHeaders
ExtractGrok
ExtractHL7Attributes
ExtractImageMetadata
ExtractMediaMetadata
ExtractRecordSchema
ExtractText
ExtractTNEFAttachments
FetchAzureBlobStorage_v12
FetchAzureDataLakeStorage
FetchBoxFile
FetchCDPObjectStore
FetchDistributedMapCache
FetchDropbox
FetchFile
FetchFTP
FetchGCSObject
FetchGoogleDrive
FetchGridFS
FetchHBaseRow
FetchHDFS
FetchParquet
FetchS3Object
FetchSFTP
FetchSmb
FilterAttribute
FlattenJson
ForkEnrichment
ForkRecord
GenerateFlowFile
GenerateRecord
GenerateTableFetch
GeoEnrichIP
GeoEnrichIPRecord
GeohashRecord
GetAsanaObject
GetAwsPollyJobStatus
PutHBaseJSON
PutHBaseRecord
PutHDFS
PutHive3QL
PutHive3Streaming
PutIceberg
PutIcebergCDC
PutIoTDBRecord
PutJiraIssue
PutKinesisFirehose
PutKinesisStream
PutKudu
PutLambda
PutMongo
PutMongoBulkOperations
PutMongoRecord
PutOpenSearchVector
PutORC
PutParquet
PutPinecone
PutQdrant
PutRecord
PutRedisHashRecord
PutS3Object
PutSalesforceObject
PutSFTP
PutSmbFile
PutSnowflakeInternalStage
PutSNS
PutSolrContentStream
PutSolrRecord
PutSplunk
PutSplunkHTTP
PutSQL
PutSQS
PutSyslog
PutTCP
PutUDP
PutWebSocket
PutZendeskTicket
QueryAirtableTable
QueryAzureDataExplorer
QueryCassandra
QueryChroma
QueryDatabaseTable
QueryDatabaseTableRecord
QueryDNS
QueryIoTDBRecord
QueryOpenSearchVector
QueryPinecone

GetAwsTextractJobStatus
GetAwsTranscribeJobStatus
GetAwsTranslateJobStatus
GetAzureEventHub
GetAzureQueueStorage_v12
GetCouchbaseKey
GetDynamoDB
GetElasticsearch
GetFile
GetFTP
GetGcpVisionAnnotateFilesOperationStatus
GetGcpVisionAnnotateImagesOperationStatus
GetHBase
GetHDFS
GetHDFSEvents
GetHDFSFileInfo
GetHDFSSequenceFile
GetHubSpot
GetJiraIssue
GetMongo
GetMongoRecord
GetSFTP
GetShopify
GetSmbFile
GetSNMP
GetSnowflakeIngestStatus
GetSolr
GetSplunk
GetSQS
GetWorkdayReport
GetZendesk
HandleHttpRequest
HandleHttpResponse
IdentifyMimeType
InsertToMilvus
InvokeAWSGatewayApi
InvokeGRPC
InvokeHTTP
InvokeScriptedProcessor
ISPEnrichIP
JoinEnrichment
JoltTransformJSON
JoltTransformRecord
JSLTTransformJSON
JsonQueryElasticsearch
LexicalQueryMilvus
ListAzureBlobStorage_v12
ListAzureDataLakeStorage
ListBoxFile
ListCDPObjectStore
QueryQdrant
QueryRecord
QuerySalesforceObject
QuerySolr
QuerySplunkIndexingStatus
QueryWhois
RemoveRecordField
RenameRecordField
ReplaceText
ReplaceTextWithMapping
ResizeImage
RetryFlowFile
RouteHL7
RouteOnAttribute
RouteOnContent
RouteText
RunMongoAggregation
SampleRecord
ScanAccumulo
ScanAttribute
ScanContent
ScanHBase
ScriptedFilterRecord
ScriptedPartitionRecord
ScriptedTransformRecord
ScriptedValidateRecord
SearchElasticsearch
SegmentContent
SelectClouderaHiveQL
SelectHive3QL
SendTrapSNMP
SetSNMP
SignContentPGP
SplitAvro
SplitContent
SplitJson
SplitRecord
SplitText
SplitXml
StartAwsPollyJob
StartAwsTextractJob
StartAwsTranscribeJob
StartAwsTranslateJob
StartGcpVisionAnnotateFilesOperation
StartGcpVisionAnnotateImagesOperation
StartSnowflakeIngest
TagS3Object
TailFile
TransformXml
TriggerClouderaHiveMetaStoreEvent

ListDatabaseTables ListDropbox ListenBeats ListenFTP ListenGRPC ListenHTTP ListenNetFlow ListenOTLP ListenRELP ListenSlack ListenSMTP ListenSyslog ListenTCP ListenTCPRecord ListenTrapSNMP ListenUDP ListenUDPRecord ListenWebSocket ListFile ListFTP	TriggerHiveMetaStoreEvent UnpackContent UpdateAttribute UpdateByQueryElasticsearch UpdateClouderaHiveTable UpdateCounter UpdateDatabaseTable UpdateDeltaLakeTable UpdateHive3Table UpdateRecord ValidateCsv ValidateJson ValidateRecord ValidateXml VectorQueryMilvus VerifyContentMAC VerifyContentPGP Wait YandexTranslate

Removed NiFi processors

Learn about the NiFi processors removed in this release of Cloudera DataFlow and the alternatives you can replace them with.

Note: These components are removed in NiFi Runtime version 1.25.0.2.3.13.0-36. If updating your flow definition with supported alternatives is not possible, select a lower NiFi Runtime Version (specify a lower --cfm-nifi-version if using the CLI) during flow deployment.

Removed processor	Replace it with
ConsumeKafka_1_0	ConsumeKafka2CDP, ConsumeKafka_2_6
ConsumeKafka_2_0	ConsumeKafka2CDP, ConsumeKafka_2_6
ConsumeKafkaRecord_1_0	ConsumeKafka2RecordCDP, ConsumeKafkaRecord_2_6
ConsumeKafkaRecord_2_0	ConsumeKafka2RecordCDP, ConsumeKafkaRecord_2_6
ConvertAvroSchema	ConvertRecord
ConvertAvroToORC	ConvertRecord
ConvertCSVToAvro	ConvertRecord
ConvertJSONToAvro	ConvertRecord
ExecuteFlumeSink	None If this processor is required, you must use a NiFi Runtime version lower than 1.25.0.2.3.13.0-36.
ExecuteFlumeSource	None If this processor is required, you must use a NiFi Runtime version lower than 1.25.0.2.3.13.0-36.
GetIgniteCache	None If this processor is required, you must use a NiFi Runtime version lower than 1.25.0.2.3.13.0-36.

Removed processor	Replace it with
InferAvroSchema	ExtractSchemaRecord
PublishKafka_1_0	PublishKafka_2_6
PublishKafka_2_0	PublishKafka2CDP, PublishKafka_2_6
PublishKafkaRecord_1_0	PublishKafka2RecordCDP, PublishKafkaRecord_2_6
PublishKafkaRecord_2_0	PublishKafka2RecordCDP, PublishKafkaRecord_2_6
PutHive_1_1QL	PutClouderaHiveQL
PutHiveQL	PutClouderaHiveQL
PutHiveStreaming	PutClouderaHiveStreaming
PutIgniteCache	None If this processor is required, you must use a NiFi Runtime version lower than 1.25.0.2.3.13.0-36.
SelectHive_1_1QL	SelectClouderaHiveQL
SelectHiveQL	SelectClouderaHiveQL
StoreInKiteDataset	None If this processor is required, you must use a NiFi Runtime version lower than 1.25.0.2.3.13.0-36.
UpdateHive_1_1Table	UpdateClouderaHiveTable
UpdateHiveTable	UpdateClouderaHiveTable

Supported NiFi controller services

Learn about the Apache NiFi controller services supported by Cloudera that you can use with Cloudera DataFlow.

Additional controller services are developed and tested by the Cloudera community but are not officially supported by Cloudera. Controller services are excluded for a variety of reasons, including insufficient reliability or incomplete test case coverage, declaration of non-production readiness by the community at large, and feature deviation from Cloudera best practices. Do not use these features in your production environments.

NiFi 1.27

AccumuloService	JMSConnectionFactoryProvider
ActionHandlerLookup	JndiJmsConnectionFactoryProvider
ActiveMQJMSConnectionFactoryProvider	JsonConfigBasedBoxClientService
ADLSCredentialsControllerService	JsonPathReader
ADLSCredentialsControllerServiceLookup	JsonRecordSetWriter
ADLSIDBrokerCloudCredentialsProviderControllerService	JsonTreeReader
AlertHandler	KafkaRecordSink_2_6
AmazonGlueSchemaRegistry	KerberosKeytabUserService
AvroReader	KerberosPasswordUserService
AvroRecordSetWriter	KerberosTicketCacheUserService
AvroSchemaRegistry	KeytabCredentialsService
AWSCredentialsProviderControllerService	KuduLookupService
AWSIDBrokerCloudCredentialsProviderControllerService	LivySessionController
AzureBlobIDBrokerCloudCredentialsProviderControllerService	LoggingRecordSink
AzureCosmosDBClientService	LogHandler
AzureEventHubRecordSink	MongoDBControllerService

AzureServiceBusJMSConnectionFactoryProvider
AzureStorageCredentialsControllerService
AzureStorageCredentialsControllerService_v12
AzureStorageCredentialsControllerServiceLookup
AzureStorageCredentialsControllerServiceLookup_v12
AzureStorageEmulatorCredentialsControllerService
CassandraDistributedMapCache
CassandraSessionProvider
CdpCredentialsProviderControllerService
CdpOauth2AccessTokenProviderControllerService
CEFReader
CiscoEmblemSyslogMessageReader
ClouderaHiveConnectionPool
ClouderaSchemaRegistry
CMLLookupService
ConfluentSchemaRegistry
CouchbaseClusterService
CouchbaseKeyValueLookupService
CouchbaseMapCacheClient
CouchbaseRecordLookupService
CSVReader
CSVRecordLookupService
CSVRecordSetWriter
DatabaseRecordLookupService
DatabaseRecordSink
DatabaseTableSchemaRegistry
DBCPConnectionPool
DBCPConnectionPoolLookup
DistributedMapCacheClientService
DistributedMapCacheLookupService
DistributedMapCacheServer
DistributedSetCacheClientService
DistributedSetCacheServer
EBCDICRecordReader1
ElasticSearchClientServiceImpl
ElasticSearchLookupService
ElasticSearchStringLookupService
EmailRecordSink
EmbeddedHazelcastCacheManager
ExcelReader
ExpressionHandler
ExternalHazelcastCacheManager
FreeFormTextRecordSetWriter
GCPCredentialsControllerService
GraphiteMetricReporterService
GremlinClientService
GrokReader
HadoopCatalogService
HadoopDBCPConnectionPool
HazelcastMapCacheClient
MongoDBLookupService
Neo4JCypherClientService
OAuth2TokenProviderImpl
OpenCypherClientService
ParquetReader
ParquetRecordSetWriter
PhoenixThickConnectionPool
PhoenixThinConnectionPool
PostgreSQLConnectionPool
PrometheusRecordSink
PropertiesFileLookupService
ProtobufReader
RabbitMQJMSConnectionFactoryProvider
ReaderLookup
RecordSetWriterLookup
RecordSinkHandler
RecordSinkServiceLookup
RedisConnectionPoolService
RedisDistributedMapCacheClientService
RedshiftConnectionPool
RestLookupService
ScriptedActionHandler
ScriptedLookupService
ScriptedReader
ScriptedRecordSetWriter
ScriptedRecordSink
ScriptedRulesEngine
SimpleCsvFileLookupService
SimpleDatabaseLookupService
SimpleKeyValueLookupService
SimpleRedisDistributedMapCacheClientService
SimpleScriptedLookupService
SiteToSiteReportingRecordSink
SmbjClientProviderService
SnowflakeComputingConnectionPool
StandardAsanaClientProviderService
StandardAzureCredentialsControllerService
StandardDropboxCredentialService
StandardFileResourceService
StandardHashiCorpVaultClientService
StandardHttpContextMap
StandardJiraCredentialService
StandardJsonSchemaRegistry1
StandardOauth2AccessTokenProvider
StandardPGPPrivateKeyService
StandardPGPPublicKeyService
StandardPrivateKeyService
StandardProxyConfigurationService
StandardRestrictedSSLContextService
StandardS3EncryptionService

HBase_2_ClientMapCacheService	StandardSnowflakeIngestManagerProviderService
HBase_2_ClientService	StandardSSLContextService
HBase_2_RecordLookupService	StandardWebClientServiceProvider
HikariCPConnectionPool	Syslog5424Reader
Hive3ConnectionPool	SyslogReader
HiveCatalogService	UDPEventRecordSink
HortonworksSchemaRegistry	VolatileSchemaCache
ImpalaConnectionPool	WindowsEventLogReader
IPFIXReader	XMLFileLookupService
IPLookupService	XMLReader
JASN1Reader	XMLRecordSetWriter
JettyWebSocketClient	YamlTreeReader
JettyWebSocketServer	ZendeskRecordSink
JiraRecordSink	

Legend

1 - Technical Preview

NiFi 2.0 [Technical Preview]

AccumuloService	JMSConnectionFactoryProvider
ActiveMQJMSConnectionFactoryProvider	JndiJmsConnectionFactoryProvider
ADLSCredentialsControllerService	JsonConfigBasedBoxClientService
ADLSCredentialsControllerServiceLookup	JsonPathReader
ADLSIDBrokerCloudCredentialsProviderControllerService	JsonRecordSetWriter
AmazonGlueSchemaRegistry	JsonTreeReader
ApicurioSchemaRegistry	KafkaRecordSink_2_6
AvroReader	KerberosKeytabUserService
AvroRecordSetWriter	KerberosPasswordUserService
AvroSchemaRegistry	KerberosTicketCacheUserService
AWSCredentialsProviderControllerService	KeytabCredentialsService
AWSIDBrokerCloudCredentialsProviderControllerService	KuduLookupService
AzureBlobIDBrokerCloudCredentialsProviderControllerService	LivySessionController
AzureCosmosDBClientService	LoggingRecordSink
AzureEventHubRecordSink	MongoDBControllerService
AzureServiceBusJMSConnectionFactoryProvider	MongoDBLookupService
AzureStorageCredentialsControllerService_v12	Neo4JCypherClientService
AzureStorageCredentialsControllerServiceLookup_v12	ParquetReader
CassandraDistributedMapCache	ParquetRecordSetWriter
CassandraSessionProvider	PostgreSQLConnectionPool
CdpCredentialsProviderControllerService	PrometheusRecordSink
CdpOauth2AccessTokenProviderControllerService	PropertiesFileLookupService
CEFReader	RabbitMQJMSConnectionFactoryProvider
CiscoEmblemSyslogMessageReader	ReaderLookup
ClouderaHiveConnectionPool	RecordSetWriterLookup
ClouderaSchemaRegistry	RecordSinkServiceLookup
CMLLookupService	RedisConnectionPoolService
ConfluentEncodedSchemaReferenceReader	RedisDistributedMapCacheClientService
ConfluentEncodedSchemaReferenceWriter	RedshiftConnectionPool
ConfluentSchemaRegistry	RestLookupService

CouchbaseClusterService	ScriptedLookupService
CouchbaseKeyValueLookupService	ScriptedReader
CouchbaseMapCacheClient	ScriptedRecordSetWriter
CouchbaseRecordLookupService	ScriptedRecordSink
CSVReader	SimpleCsvFileLookupService
CSVRecordLookupService	SimpleDatabaseLookupService
CSVRecordSetWriter	SimpleKeyValueLookupService
DatabaseRecordLookupService	SimpleRedisDistributedMapCacheClientService
DatabaseRecordSink	SimpleScriptedLookupService
DatabaseTableSchemaRegistry	SiteToSiteReportingRecordSink
DBCPConnectionPool	SlackRecordSink
DBCPConnectionPoolLookup	SmbjClientProviderService
DistributedMapCacheClientService	SnowflakeComputingConnectionPool
DistributedMapCacheLookupService	StandardAsanaClientProviderService
DistributedMapCacheServer	StandardAzureCredentialsControllerService
DistributedSetCacheClientService	StandardDropboxCredentialService
DistributedSetCacheServer	StandardFileResourceService
EBCDICRecordReader	StandardHashiCorpVaultClientService
ElasticSearchClientServiceImpl	StandardHttpContextMap
ElasticSearchLookupService	StandardJsonSchemaRegistry
ElasticSearchStringLookupService	StandardKustoIngestService
EmailRecordSink	StandardKustoQueryService
EmbeddedHazelcastCacheManager	StandardOauth2AccessTokenProvider
ExcelReader	StandardPGPPrivateKeyService
ExternalHazelcastCacheManager	StandardPGPPublicKeyService
FreeFormTextRecordSetWriter	StandardPrivateKeyService
GCPCredentialsControllerService	StandardProxyConfigurationService
GCSFileResourceService	StandardRestrictedSSLContextService
GrokReader	StandardS3EncryptionService
HadoopCatalogService	StandardSnowflakeIngestManagerProviderService
HadoopDBCPConnectionPool	StandardSSLContextService
HazelcastMapCacheClient	StandardWebClientServiceProvider
HBase_2_ClientMapCacheService	Syslog5424Reader
HBase_2_ClientService	SyslogReader
HBase_2_RecordLookupService	TinkerpopClientService
HikariCPConnectionPool	UDPEventRecordSink
Hive3ConnectionPool	VolatileSchemaCache
HiveCatalogService	WindowsEventLogReader
IPFIXReader	XMLFileLookupService
IPLookupService	XMLReader
JASN1Reader	XMLRecordSetWriter
JettyWebSocketClient	YamlTreeReader
JettyWebSocketServer	ZendeskRecordSink
JiraRecordSink	

Removed NiFi controller services

Learn about the Apache NiFi controller services removed in this release of Cloudera DataFlow and the alternatives you can replace them with.

Note: These components are removed in NiFi Runtime version 1.25.0.2.3.13.0-36. If updating your flow definition with supported alternatives is not possible, select a lower NiFi Runtime Version (specify a lower --cfm-nifi-version if using the CLI) during flow deployment.

Removed controller service	Replace it with
EasyRulesEngineProvider	None If this processor is required, you must use a NiFi Runtime version lower than 1.25.0.2.3.13.0-36.
EasyRulesEngineService	None If this processor is required, you must use a NiFi Runtime version lower than 1.25.0.2.3.13.0-36.
Hive_1_1ConnectionPool	Hive3ConnectionPool
HiveConnectionPool	Hive3ConnectionPool
KafkaRecordSink_1_0	KafkaRecordSink_2_6
KafkaRecordSink_2_0	KafkaRecordSink_2_6

Supported NiFi reporting tasks

Learn about the Apache NiFi reporting tasks supported by Cloudera that you can use with Cloudera DataFlow.

The following NiFi reporting tasks are supported in Cloudera DataFlow:

NiFi 1.27

- AmbariReportingTask
- ControllerStatusReportingTask
- MetricsEventReportingTask
- MonitorDiskUsage
- MonitorMemory
- PrometheusReportingTask
- QueryNiFiReportingTask
- ReportLineageToAtlas
- ScriptedReportingTask
- SiteToSiteBulletinReportingTask
- SiteToSiteMetricsReportingTask
- SiteToSiteProvenanceReportingTask
- SiteToSiteStatusReportingTask

NiFi 2.0 [Technical Preview]

- ControllerStatusReportingTask
- MonitorDiskUsage
- MonitorMemory
- PrometheusReportingTask
- QueryNiFiReportingTask
- ReportLineageToAtlas
- ScriptedReportingTask
- SiteToSiteBulletinReportingTask
- SiteToSiteMetricsReportingTask
- SiteToSiteProvenanceReportingTask
- SiteToSiteStatusReportingTask

Feature support matrix

Certain features may not work on every available combination of Apache NiFi and Cloudera Runtime versions. Learn about these restrictions.

Supported feature	NiFi Runtime version	Cloudera Runtime version
Ranger Authorization Service (RAZ)	1.16.0.2.3.6.1-1 or lower	7.2.12-7.2.14
Ranger Authorization Service (RAZ)	1.18.0.2.3.6.2-2 or higher	7.2.15 or higher

Cloudera DataFlow Functions

AWS Lambda, Google Cloud Functions, and Azure Functions are supported for Cloudera DataFlow Functions. You can use any trigger available in these services when working with Cloudera DataFlow Functions.

All processors and controller services supported in the latest version of the Cloudera DataFlow and Cloudera Flow Management are supported with Cloudera DataFlow Functions.

Cloudera DataFlow

Top Tasks

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Deploying a flow definition using the wizard..... 4
Select the flow definition version you want to deploy from the catalog.....4
Launch the deployment wizard..... 4
Name your flow deployment and assign it to a project.....5
Configure NiFi.....6
Provide parameter values..... 7
Configure sizing and scaling.....7
Set Key performance indicators..... 7
Verify your settings and initiate deployment.....8

Creating a Cloudera DataFlow function in AWS..... 8
Import a flow definition..... 8
Retrieving data flow CRN.....9
Creating Cloudera service account.....10
Generating Access Key ID and Private Key..... 10
Creating a Lambda function.....11
Configuring your Lambda function.....13

Tutorial: Building a new flow from scratch..... 18
Create a new flow..... 19
Create controller services..... 23
Build your draft flow.....28
Start a test session..... 40
Publish your flow definition to the Catalog..... 42

Deploying a flow definition using the wizard

Deploy a flow definition to run Apache NiFi flows as flow deployments in Cloudera DataFlow. To do this, launch the Deployment wizard and specify your environment, parameters, sizing, and KPIs.

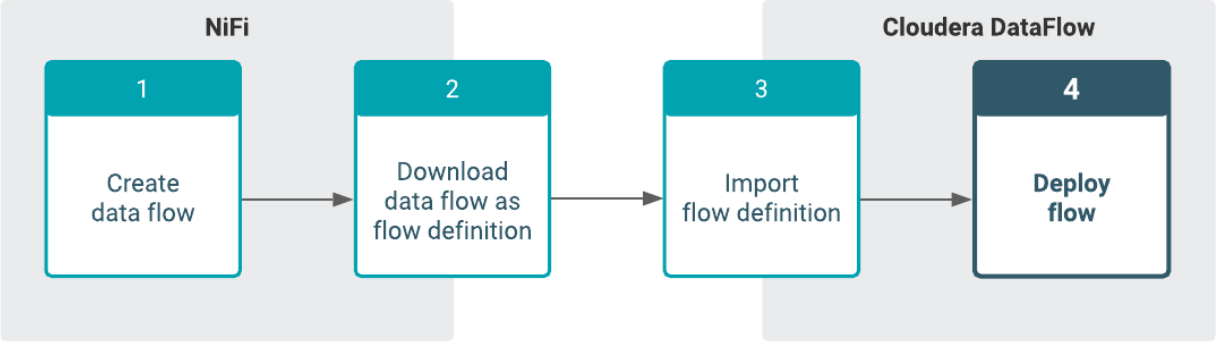

Before you begin

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog.
- The flow definition you want to deploy has been added to the Catalog by someone with DFCatalogAdmin role.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- You have been assigned DFProjectMember role for the Project where you want to deploy the flow definition.
- If you are deploying custom processors or controller services, you may need to meet additional prerequisites.

Select the flow definition version you want to deploy from the catalog

The Catalog is where you manage the flow definition lifecycle, from initial import, to versioning, to deploying a flow definition.

Procedure

1. In Cloudera DataFlow, select Catalog from the left navigation pane.

   Flow definitions available for you to deploy are displayed, one definition per row.
2. Select a row to display the flow definition details and available versions.

   The flow details pane opens on the right.

Launch the deployment wizard

After selecting a flow definition version from the catalog, you need to select an environment, provide a deployment name and assign it to a project using the deployment wizard.

Procedure

1. Click Deploy to launch the Deployment wizard.

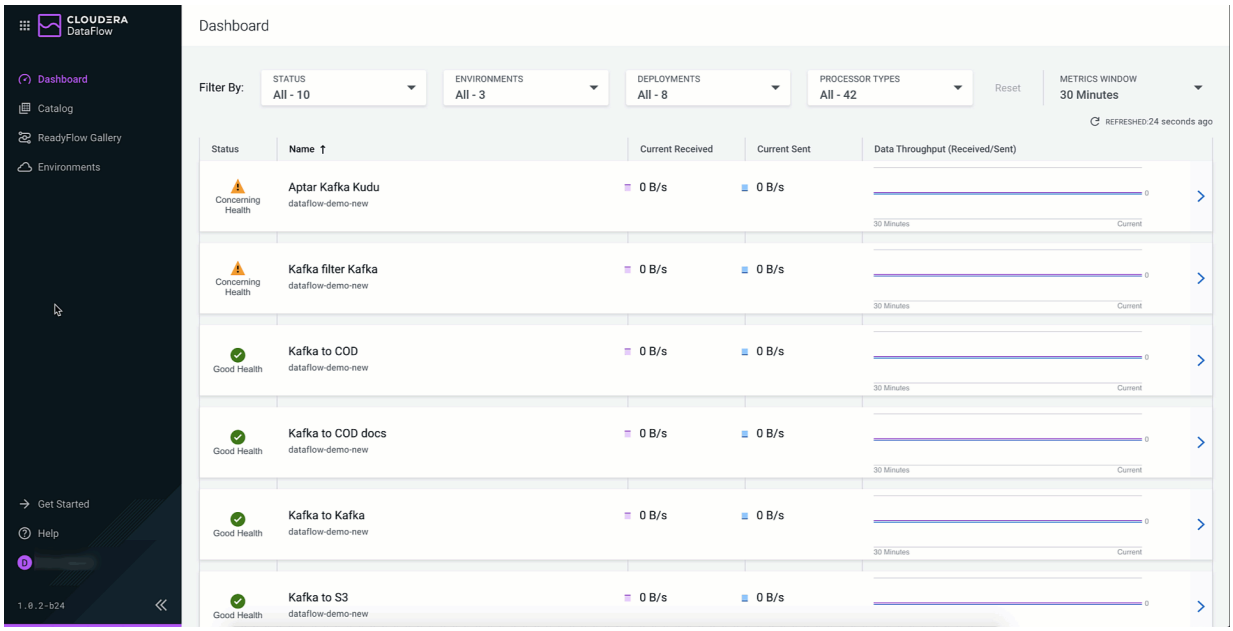

2. Select the environment where you want to deploy the flow.

   Note:

   Only environments which have been enabled for Cloudera DataFlow, are in a healthy state, and to which you have access show up in the dropdown. Once you have selected the target environment, you cannot change it later in the flow deployment process without starting over.

3. Click Deploy.

Name your flow deployment and assign it to a project

After selecting the flow version and an environment, the deployment wizard takes you to the Overview page. Here you need to provide a name for your flow deployment and assign it to a project. At this point you can also import a previously exported deployment configuration, auto-filling configuration values and thus speeding up deployment.

Procedure

1. Give your flow a unique Deployment Name.

   You can use this name to distinguish between different versions of a flow definition, flow definitions deployed to different environments, and similar.

   Note:

   Flow Deployment names need to be unique within an Environment. The Deployment wizard indicates whether a name is valid by displaying a green check below the Deployment Name text box.

2. Select a Target Project for your flow deployment from the list of Projects available to you.

   - If you do not want to assign the deployment to any of the available Projects, select Unassigned. Unassigned deployments are accessible to every user with DFFlowUser role in the environment.
   - This field is automatically populated if you import a configuration and the Project referenced there exists in your environment, and you have access to it.

3. If you have previously exported a deployment configuration that closely aligns with the one you are about to deploy, you can import it under Import Configuration to auto-fill as much of the wizard as possible.

   You can later manually modify auto-filled configuration values during deployment.
4. Click Next.

Configure NiFi

After selecting the target environment, project, and naming your flow, you need to set Apache NiFi version, possible inbound connections, and custom processors. Depending on the flow definition, you may also need to provide values for a number of configuration parameters. Finally, you need to set the capacity of the NiFi cluster servicing your deployment.

Procedure

1. Pick a NiFi Runtime Version for your flow deployment.

   Select if you want to use Apache NiFi 1.x or 2.x with your deployment.

   Important: NiFi 2.x is currently provided as a technical preview feature, do not use it for deployments in production environments.

   Cloudera recommends that you always use the latest available version within the 1.x and 2.x lines, if possible.
2. Specify whether you want the flow deployment to auto-start once deployed.
3. Specify whether you want to use Inbound Connections that allow your flow deployment receiving data from an external data source.

   If yes, specify the endpoint host name and listening port(s) where your flow deployment listens to incoming data.

   See *Creating an inbound connection endpoint* for complete information on endpoint configuration options.
4. Specify whether you want to use NiFi Archives (NARs) to deploy custom NiFi processors or controller services.

   If yes, specify the CDP Workload Username, password, and cloud storage location you used when preparing to deploy custom processors.

   Tip: If you want to provide a machine user as CDP Workload Username during flow deployment, make sure to note the full workload user name including the srv_ prefix.

   Make sure that you click the Apply button specific to Custom NAR Configuration before proceeding.
5. If you selected to run your flow with NiFi 2.x [Technical Preview], specify whether you want to use custom Python processors with your flow deployment.

   If yes, specify the CDP Workload Username, password, and cloud storage location where the processors are stored.

   Tip: Create a dedicated directory in your object store where you keep all your Python processors. Create one Python script per processor and store it in this directory.

   Tip: If you want to provide a machine user as CDP Workload Username during flow deployment, make sure to note the full workload user name including the srv_ prefix.

   Make sure that you click the Apply button specific to Custom Python Processors before proceeding.
6. Click Next.

Related Information

Inbound connections

Provide parameter values

Depending on the flow you deploy, you may need to specify parameter values like connection strings, usernames and similar, and upload files like truststores, JARs, and similar.

Procedure

1. Provide values to parameters required for your flow deployment.

   You have to provide values for all parameters. You can filter for the still empty fields by selecting the No value checkbox.

   Tip: If you are deploying a ReadyFlow, you can learn about required parameters and instructions on how to obtain parameter values by checking *Prerequisites* and *Required parameters* in the documentation of the respective ReadyFlow.
2. When you finished setting configuration parameters, click Next.

Configure sizing and scaling

Set the size and number of Apache NiFi nodes, auto-scaling, and the type of storage to be used.

Procedure

1. Specify NiFi node size.

   Select one of the following options:

   - Extra Small: 2 vCores per Node, 4 GB per Node
   - Small: 3 vCores per Node, 6 GB per Node
   - Medium: 6 vCores per Node, 12 GB per Node
   - Large: 12 vCores per Node, 24 GB per Node
2. Set the number of NiFi nodes and auto-scaling.

   - You can set whether you want to automatically scale your cluster according to flow deployment capacity requirements. When you enable auto-scaling, the minimum number of NiFi nodes are used for initial size and the workload scales up or down depending on resource demands.
   - You can set the number of nodes between 1 and 32.
   - You can set whether you want to enable Flow Metrics Scaling.
3. Select storage type.

   Select whether you want your deployment to use storage optimized for cost or for performance.

   - Standard: 512 GB Content Repo Size, 512 GB Provenance Repo Size, 256 GB Flow File Repo Size, 2300 IOPS, 150 MB/s Max Throughput
   - Performance: 1024 GB Content Repo Size, 1024 GB Provenance Repo Size, 256 GB Flow File Repo Size, 5000 IOPS, 200 MB/s Max Throughput
4. Click Next.

Related Information

Auto-scaling

Set Key performance indicators

Optionally add key performance indicators to help you track the performance of your flow deployment then review your settings and launch the deployment process.

Procedure

1. From KPIs, you may choose to identify key performance indicators (KPIs), the metrics to track those KPIs, and when and how to receive alerts about the KPI metrics tracking.

   Tip:

   You can reorder the KPIs by dragging them up or down the list. The order you configure here will be reflected in the detailed monitoring view of the resulting deployment.

   See *Working with KPIs* for complete information about the KPIs available to you and how to monitor them.
2. Click Next.

Related Information

Working with KPIs

Verify your settings and initiate deployment

Review deployment settings, make any necessary changes, and start deployment.

Procedure

1. Review a summary of the information provided and make any necessary edits by clicking Previous.
2. When you are finished, complete your flow deployment by clicking Deploy.

   Tip:

   Click View CLI Command to see the equivalent Cloudera CLI syntax in a help pane.

Results

After you click Deploy, you are redirected to the **Alerts** tab in the **Flow Details** where you can track how the deployment progresses.

Creating a Cloudera DataFlow function in AWS

You are all set up for creating your first Cloudera DataFlow function: you have a flow definition created in and downloaded from NiFi, a Cloudera DataFlow environment, and the appropriate rights to that environment. All you need now is to follow these steps to get your function up and running.

Prerequisites

- You have a data flow created in Apache NiFi and downloaded as a flow definition JSON file.
- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFCatalogAdmin role granting you access to the Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- An AWS user account is required with access policies that has permissions to list and create buckets, roles, and Lambda functions.

Import a flow definition

If you want to use an Apache NiFi flow in Cloudera DataFlow, you must import it as a flow definition. When imported, the flow definition is added to the Flow Catalog.

Steps

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Catalog from the left navigation pane.

   The Flow Catalog page is displayed. Previously imported flow definitions are displayed, one definition per row.
3. Click Import Flow Definition.

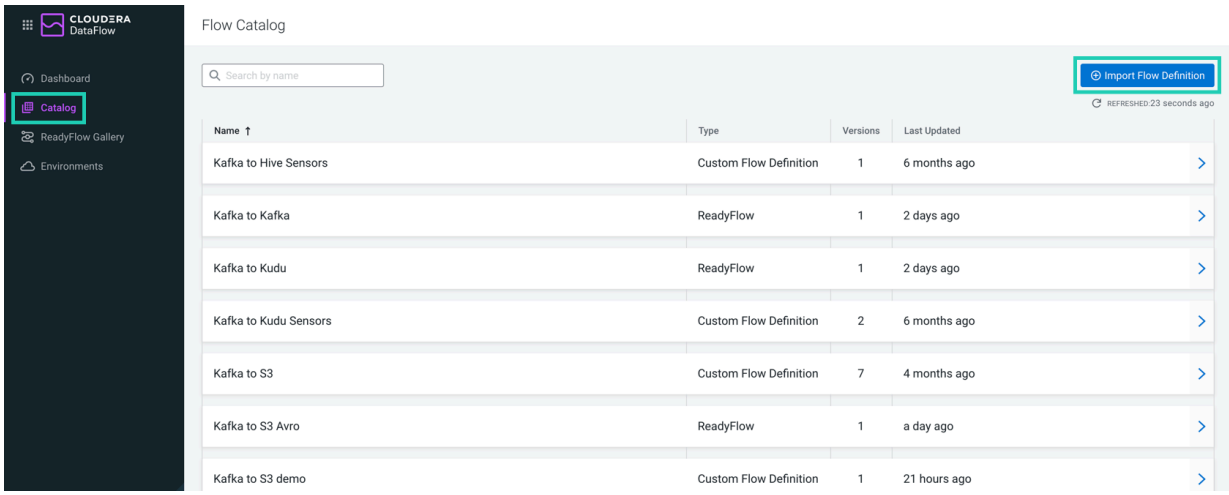

4. Provide a name (Flow Name) and a description (Flow Description) for the flow definition.
5. In the NiFi Flow Configuration field, select the JSON file you downloaded from NiFi.

   Alternatively, you can drop the JSON file into this field to select it.
6. Optional:You can add comments to the flow definition version you are importing (Version Comments).

   Note: When you are importing the first version of a flow definition, the Version Comments field contains Initial Version by default. You can change this comment if needed.
7. Click Import.

Result

You have successfully imported your NiFi flow to Cloudera DataFlow. It is available in the Flow Catalog as the first version of your flow definition.

Retrieving data flow CRN

When configuring your function on the cloud provider service page, you need to provide the Customer Resource Number (CRN) of the flow to be executed.

You can retrieve the CRN by checking the flow in the Cloudera DataFlow Catalog.

Note: The CRN must include the specific version of the flow that should be executed, so it should end with some version suffix such as /v.1.

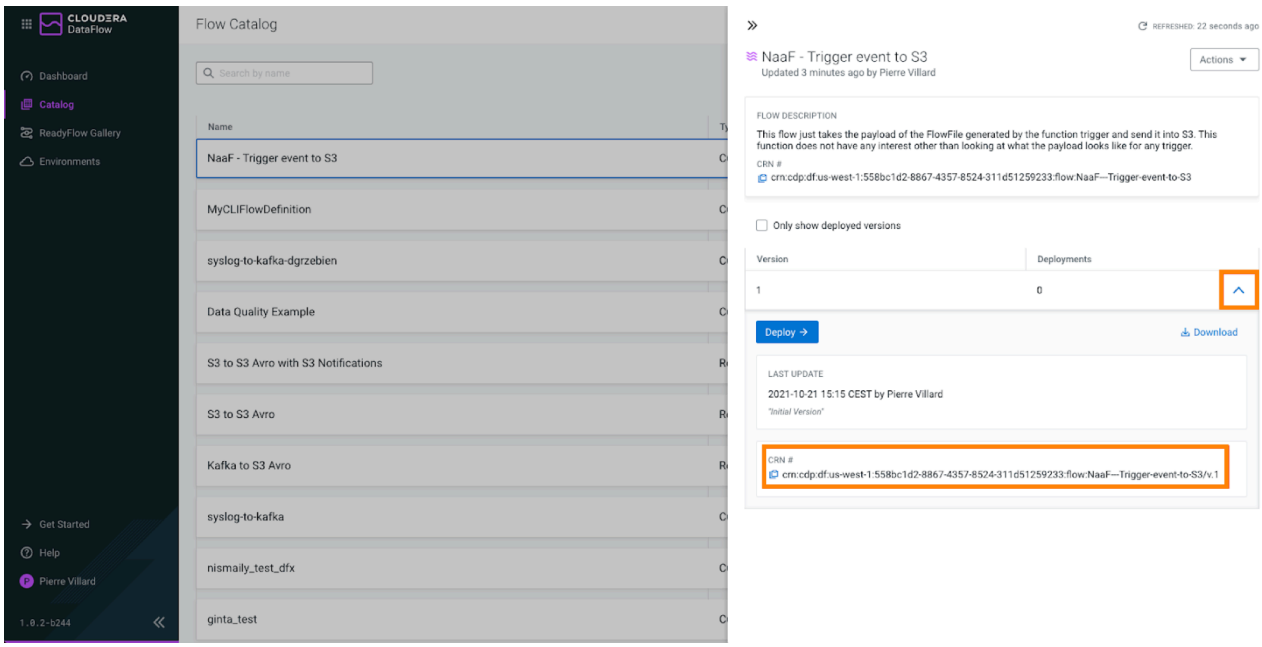

Creating Cloudera service account

The first step of executing Cloudera DataFlow Functions code in your cloud environment is fetching the flow definition to be executed from the Cloudera DataFlow Catalog. For this, you need to create a service account and provision an access key.

Procedure

1. Navigate to the Management Console User Management Users .
2. From the Actions menu, select Create Machine User.
3. Provide a name and click Create.

Note: Machine user names cannot start with a double underscore ("__").

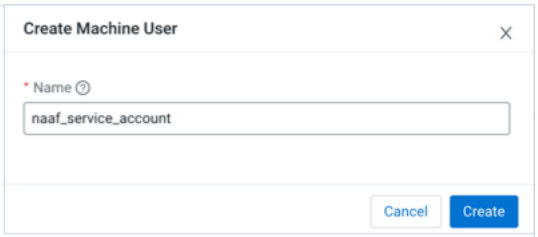

The user details page is displayed showing information about the user.

Generating Access Key ID and Private Key

A Cloudera machine user must have API access credentials to access Cloudera services through the Cloudera CLI or API.

Procedure

1. Click the Roles tab on the user account details page.

2. Click Update Roles.

   The Update Roles pane is displayed.
3. Select the DFFunctionMachineUser role to assign it to your user.
4. Click Update.

   When the role is added, you should see:

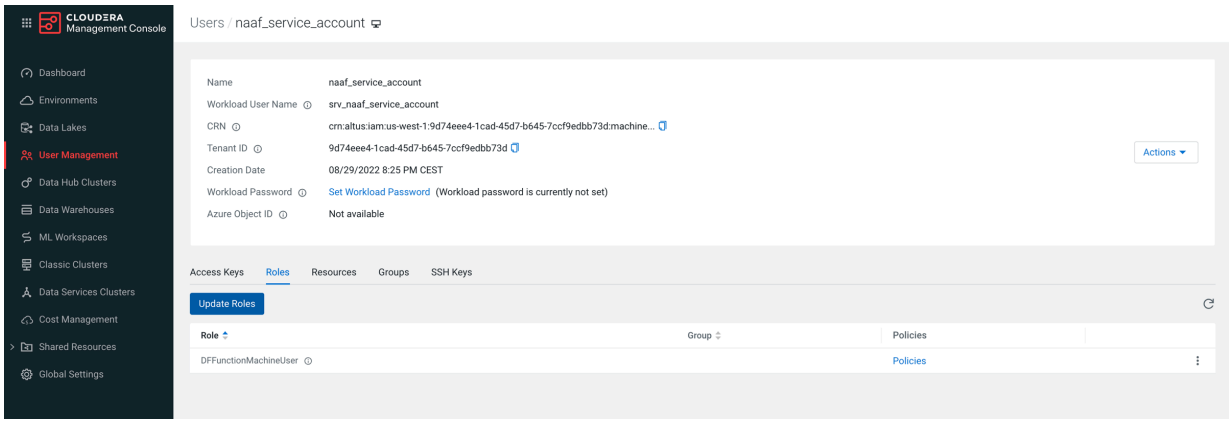

5. Select the Access Keys tab on the user account details page.
6. Click Generate Access Key.

   The Generate Access Key modal window is displayed. it gives you an Access Key ID and a Private Key, which will be required when configuring your function.
7. Click Generate Access Key.

   A message is displayed that your access key has been successfully created.

   You can copy your Access Key ID and your Private Key. You will need these when configuring your function.
8. You can also download the credentials file into the .cdp directory in your user home directory. Or run the command cdp configure and enter the access key ID and private key to create a Cloudera credentials file in the same directory.

Results

When ready, you should see something like this:

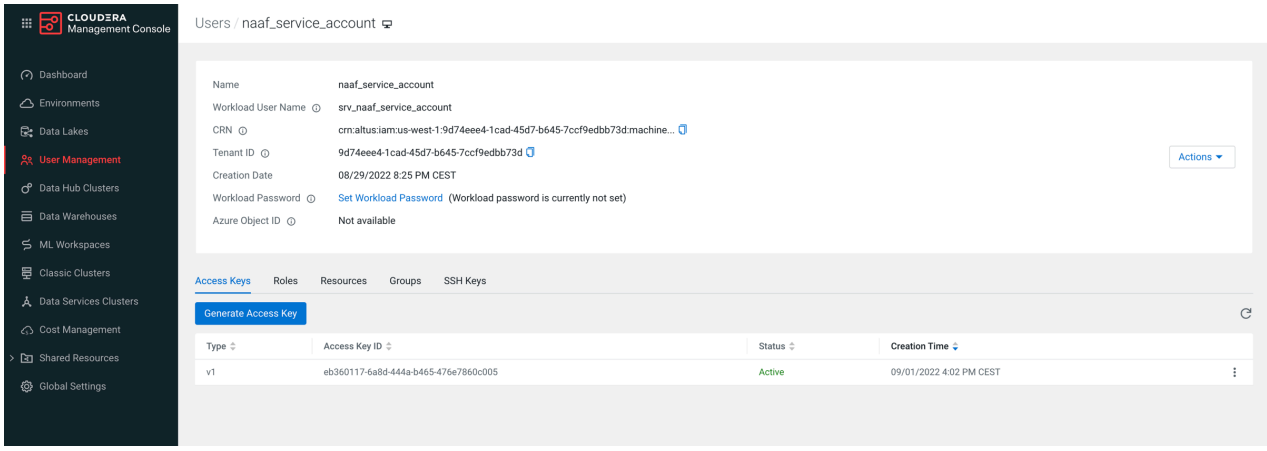

Creating a Lambda function

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers.

About this task

Follow these steps to create an AWS Lambda function that is able to run Cloudera DataFlow Functions:

Procedure

1. Navigate to the AWS Lambda page in your cloud account in the region of your choice.

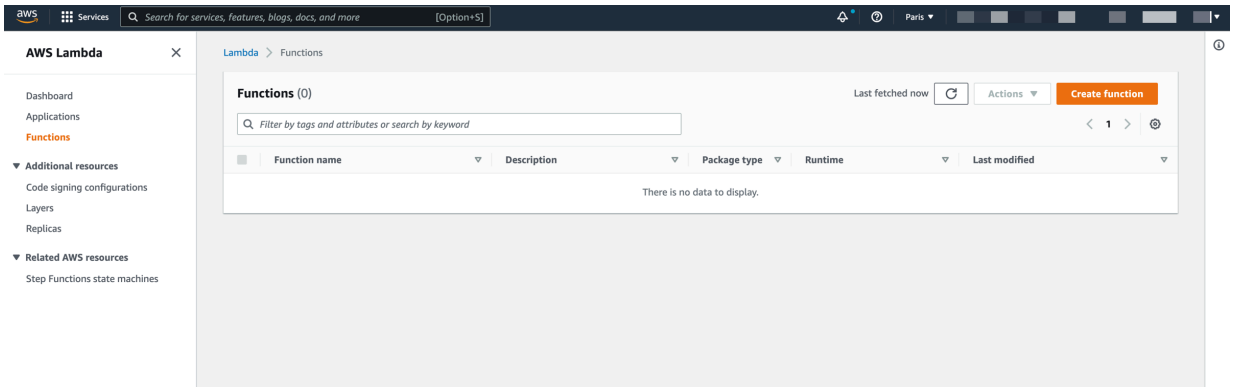

2. Click Create function.

   The Create function page opens.

3. Fill in the Basic information section:
   a) Keep the default Author from scratch option selected.
   b) Give your function a meaningful name for your use case.
   c) Select Java 8 on Amazon Linux 2 for Runtime.
   d) Choose x86_64 for Architecture.
   e) Click Create function.

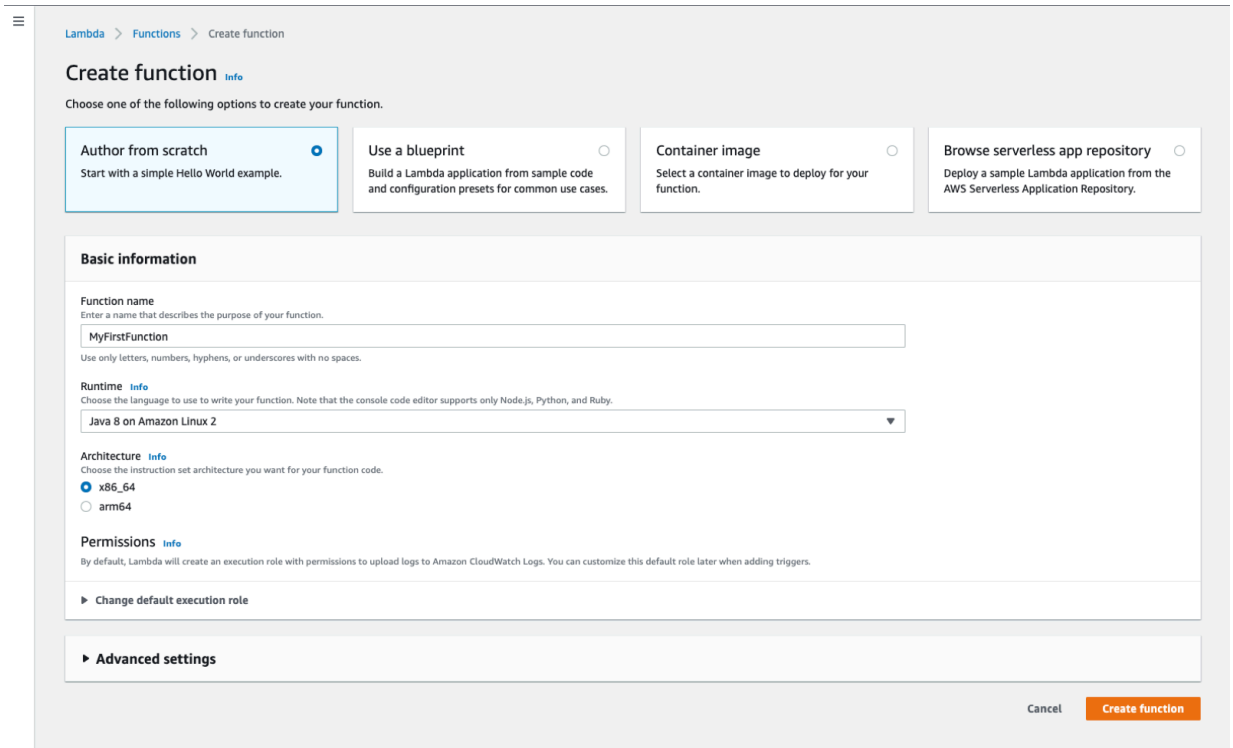

   Your function is created and its details are displayed.

4. In the Code source section of the Code tab, click Upload from Amazon S3 location .

   The Upload a file from Amazon S3 modal window appears.

5. Provide the S3 URL of Cloudera DataFlow Function binaries in the textbox.

   Note: You have downloaded the Cloudera DataFlow Function handler libraries as a zip of binaries from Cloudera DataFlow and uploaded them to an S3 bucket when getting ready to run functions. AWS Lambda will use these binaries to run the NiFi flow. For instructions, see *Downloading Lambda Cloudera DataFlow Function binaries and uploading to S3*.

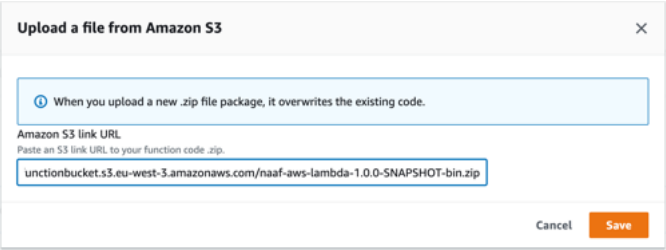

6. Click Save.
7. Navigate to the Runtime Settings on the Code tab and click Edit to change the Handler property to: com.cloudera .naaf.aws.lambda.StatelessNiFiFunctionHandler::handleRequest

   The details of your first function:

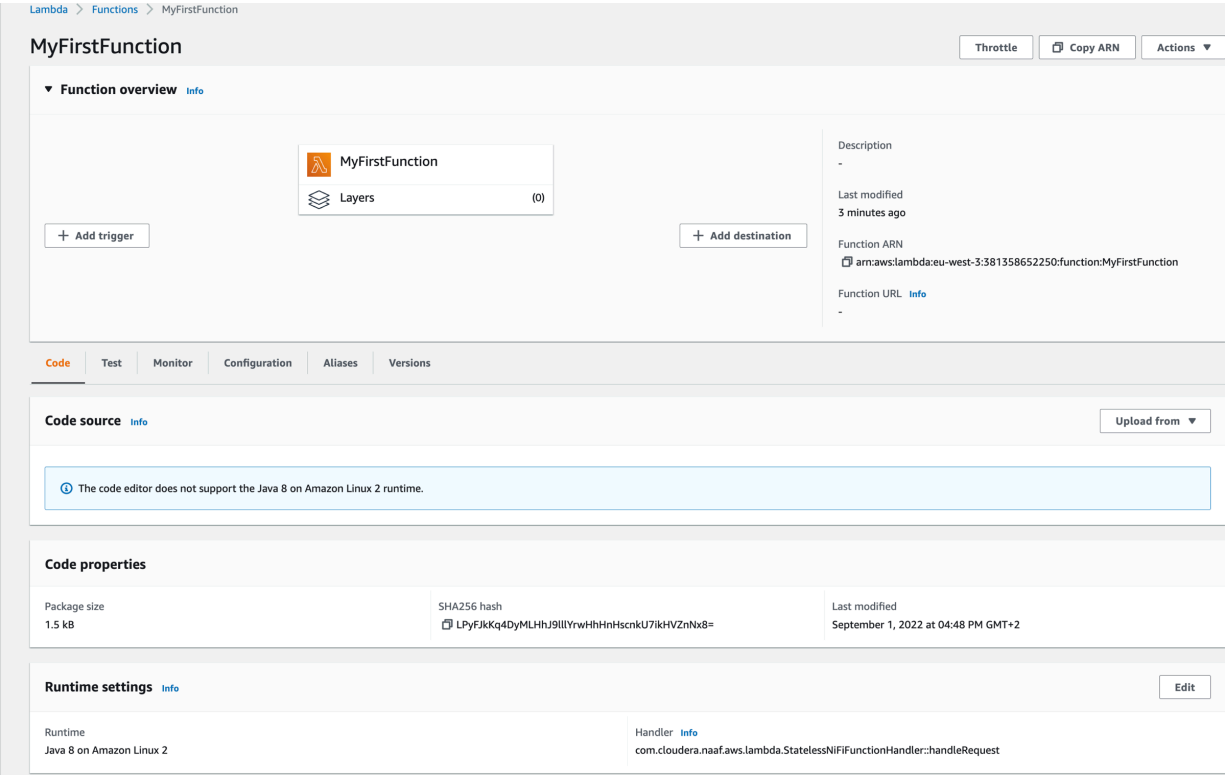

What to do next

You can now start configuring your Lambda specifically for your flow and for your use case.

Configuring your Lambda function

After you create a function, you can use the built-in configuration options to control its behavior. You can configure additional capabilities, adjust resources associated with your function, such as memory and timeout, or you can also create and edit test events to test your function using the console.

Memory and timeout in runtime configuration

You can configure two very important elements, memory and timeout on the Configuration General configuration tab of the Lambda function details screen.

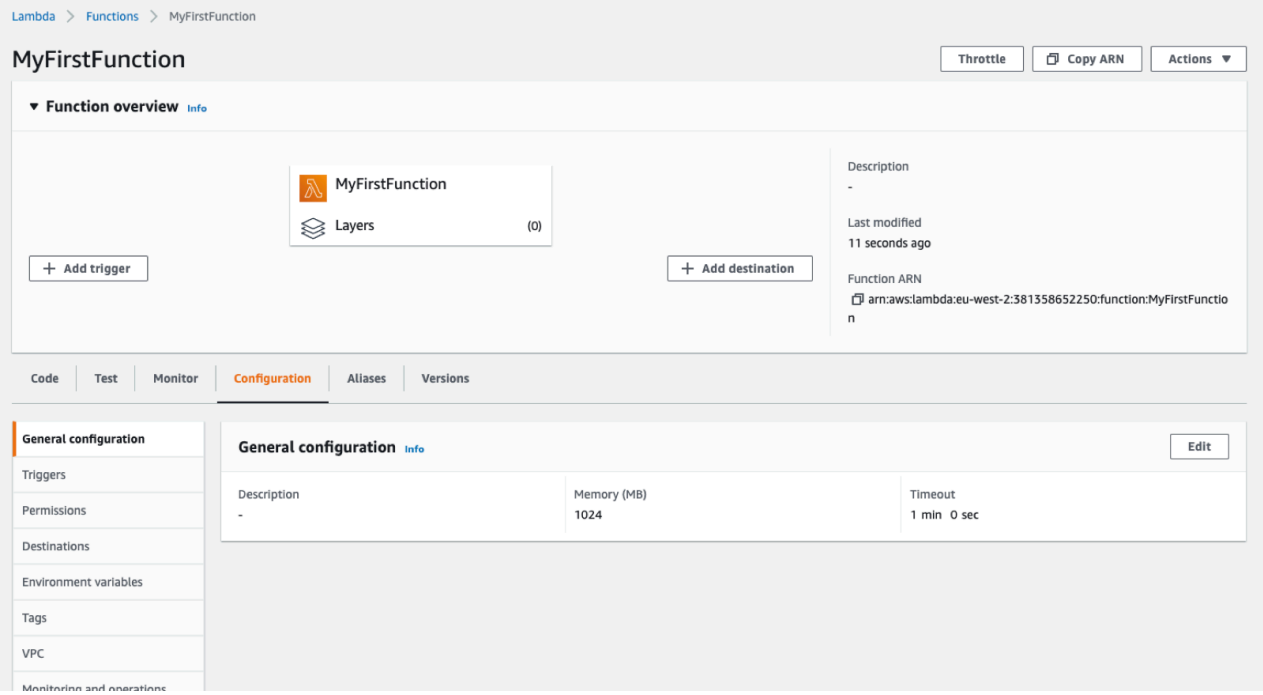

The appropriate values for these two settings depend on the data flow to be deployed. Typically, it is recommended to start with 1536-2048 MB (1.5 - 2 GB) memory allocated to Lambda. This amount of memory may be more than necessary or less than required. The default value of 512 MB may be perfectly fine for many use cases. You can adjust this value later as you see how much memory your function needs.

While Cloudera DataFlow Functions tend to perform very well during a warm start, a cold start that must source extensions (processors, controller services, and so on) may take several seconds to initialize. So it is recommended to set the Timeout to at least 30 seconds. If the data flow to run reaches out to many other services or performs complex computation, it may be necessary to use a larger value, even several minutes.

Click Edit if you want to change the default values.

See the corresponding AWS documentation to learn more about timeout duration and memory sizing, which determines the amount of resources provisioned for your function.

Runtime environment variables

You must configure the function to specify which data flow to run and add any necessary runtime configuration using environment variables.

Procedure

1. Navigate to the Configuration tab and select Environment variables from the menu on the left.
2. Click Edit on the Environment variables pane.

3. Define your environment variables as key-value pairs.

   You can add one or more environment variables to configure the function. The following environment variables are supported:

Variable Name	Description	Required	Default Value
FLOW_CRN	The Cloudera Resource Name (CRN) for the data flow that is to be run. The data flow must be stored in the Cloudera DataFlow Catalog. This CRN should indicate the specific version of the data flow and as such will end with a suffix like /v.1. For more information, see *Retrieving data flow CRN*.	true	--
DF_PRIVATE_KEY	The Private Key for accessing the Cloudera DataFlow service. The Private Key and Access Key are used to authenticate with the Cloudera DataFlow Service and they must provide the necessary authorizations to access the specified data flow. For more information, see *Provisioning Access Key ID and Private Key*.	true	--
DF_ACCESS_KEY	The Access Key for accessing the Cloudera DataFlow service. The Private Key and Access Key are used to authenticate with the Cloudera DataFlow Service and they must provide the necessary authorizations to access the specified data flow. For more information, see *Provisioning Access Key ID and Private Key*.	true	--
INPUT_PORT	The name of the Input Port to use. If the specified data flow has more than one Input Port at the root group level, this environment variable must be specified, indicating the name of the Input Port to queue up the AWS Lambda notification. If there is only one Input Port, this variable is not required. If it is specified, it must properly match the name of the Input Port.	false	--

Variable Name	Description	Required	Default Value
OUTPUT_PORT	The name of the Output Port to retrieve the results from. If no Output Port exists, the variable does not need to be specified and no data will be returned. If at least one Output Port exists in the data flow, this variable can be used to determine the name of the Output Port whose data will be sent along as the output of the function. For more information on how the appropriate Output Port is determined, see *Output ports*.	false	--
FAILURE_PORTS	A comma-separated list of Output Ports that exist at the root group level of the data flow. If any FlowFile is sent to one of these Output Ports, the function invocation is considered a failure. For more information, see *Output ports*.	false	--
DEFAULT_PARAM_CONTEXT	If the data flow uses a Parameter Context, the Lambda function will look for a Secret in the AWS Secrets Manager with the same name. If no Secret can be found with that name, this variable specifies the name of the Secret to default to. For more information, see *Parameters*.	false	--
PARAM_CONTEXT_*	If the data flow makes use of Parameter Contexts, this environment variable provides a mechanism for mapping a Parameter Context to one or more alternate Secrets in the AWS Secrets Manager, in a comma-separated list. For more information, see *Parameters*.	false	--
CONTENT_REPO	The contents of the FlowFiles can be stored either in memory, on the JVM heap, or on disk. If this environment variable is set, it specifies the path to a directory where the content should be stored. If it is not specified, the content is held in memory. For more information, see *File system for content repository*.	false	--

Variable Name	Description	Required	Default Value
EXTENSIONS_*	The directory to look for custom extensions / NiFi Archives (NARs). For more information, see *Providing custom extensions / NARs*.	false	--
DF_SERVICE_URL	The Base URL for the Cloudera DataFlow Service.	false	https://api.us-west-1.cdp.cloudera.com/
NEXUS_URL	The Base URL for a Nexus Repository for downloading any NiFi Archives (NARs) needed for running the data flow.	false	https://repository.cloudera.com/artifactory/cloudera-repos/
STORAGE_BUCKET	An S3 bucket in which to look for custom extensions / NiFi Archives (NARs) and resources. For more information, see *S3 Bucket storage*.	false	--
STORAGE_EXTENSIONS_DIRECTORY	The directory in the S3 bucket to look for custom extensions / NiFi Archives (NARs). For more information, see *S3 Bucket storage*.	false	extensions
STORAGE_RESOURCES_DIRECTORY	The directory in the S3 bucket to look for custom resources. For more information, see *Providing additional resources*.	false	resources
WORKING_DIR	The working directory, where NAR files will be expanded.	false	/tmp/working
EXTENSIONS_DOWNLOAD_DIR	The directory to which missing extensions / NiFi Archives (NARs) will be downloaded. For more information, see *Providing custom extensions / NARs*.	false	/tmp/extensions
EXTENSIONS_DIR_*	It specifies read-only directories that may contain custom extensions / NiFi Archives (NARs). For more information, see *Providing custom extensions / NARs*.	false	--
DISABLE_STATE_PROVIDER	If true, it disables the DynamoDB data flow state provider, even if the data flow has stateful processors.	false	--
DYNAMODB_STATE_TABLE	The DynamoDB table name where the state will be stored	false	nifi_state
DYNAMODB_BILLING_MODE	It sets the DynamoDB Billing Mode (PROVISIONED or PAY_PER_REQUEST).	false	PAY_PER_REQUEST

Variable Name	Description	Required	Default Value
DYNAMODB_WRITE_CAPACITY_UNITS	It sets the DynamoDB Write Capacity Units, when using PROVISIONED Billing Mode.	true if using PROVISIONED Billing Mode	--
DYNAMODB_READ_CAPACITY_UNITS	It sets the DynamoDB Read Capacity Units, when using PROVISIONED Billing Mode.	true if using PROVISIONED Billing Mode	--
KRB5_FILE	It specifies the filename of the krb5.conf file. This is necessary only if connecting to a Kerberos-protected endpoint. For more information, see *Configuring Kerberos*.	false	/etc/krb5.conf

4. When ready, click Save.

Tutorial: Building a new flow from scratch

If you are new to flow design and have never used NiFi before, this tutorial is for you. Learn how to build a draft adding and configuring components, connecting them, creating Controller Services, and testing your flow while creating it.

About this task

This tutorial walks you through the creation of a simple flow design that retrieves the latest changes from Wikipedia through invoking the Wikipedia API. It converts JSON events to Avro, before filtering and routing them to two different processors which merge events together before a file is written to local disk.

You will learn about:

- Creating a draft
- Creating a Controller Service
- Adding processors to your draft
- Configuring processors
- Adding a user-defined property to a processor configuration
- Connecting processors to create relationships between them
- Running a Test Session
- Publishing a draft to the Catalog as a flow definition.

Before you begin

The flow you are about to build can be deployed without any external dependencies and does not require any parameter values during deployment. Still, there are prerequisites you have to meet before you can start building your first draft.

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFDeveloper role granting you access to the Flow Designer.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog. You will need this authorization to publish your draft as a flow definition to the Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.

Create a new flow

Create a new flow in a Flow Designer Workspace and give it a name.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.

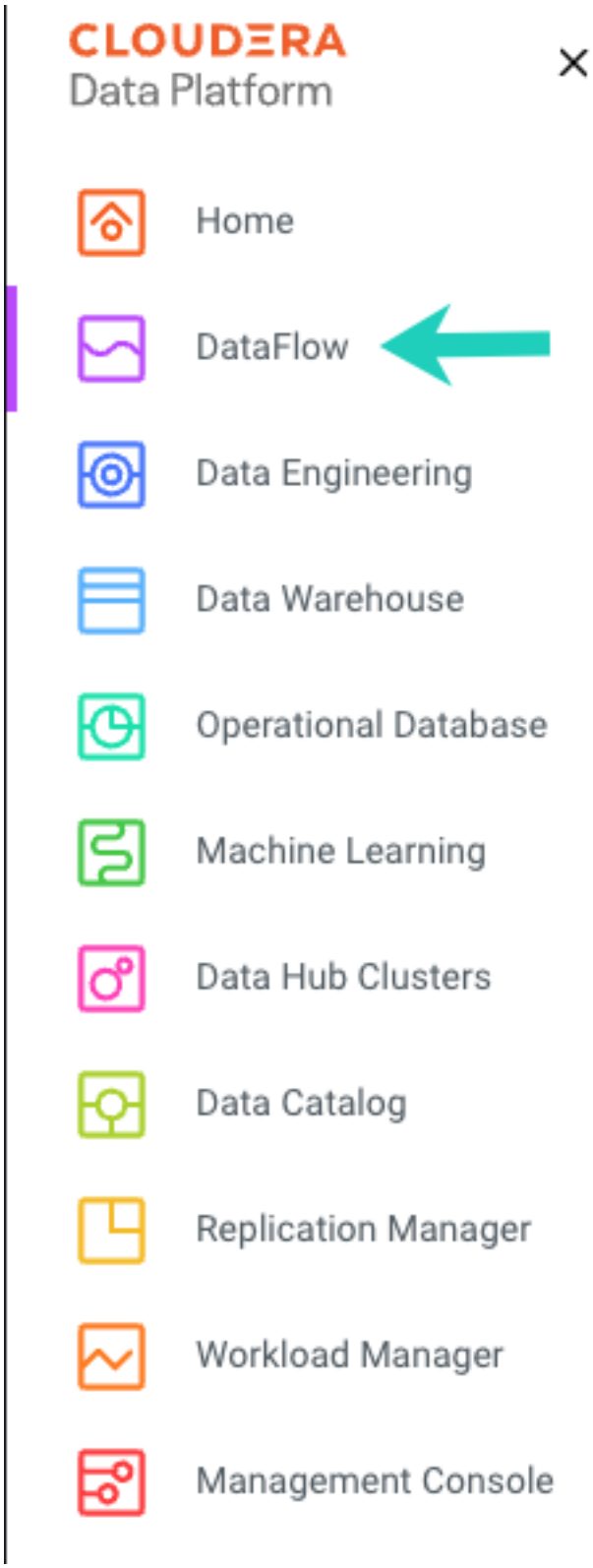

2. Click Flow Design in the left navigation pane.

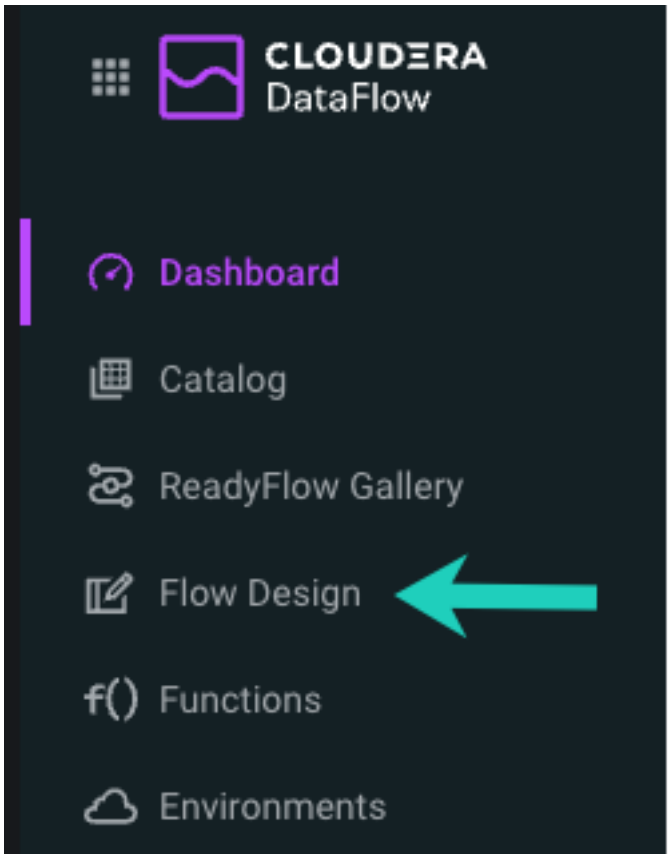

You are redirected to the **Flow Design** page, where previously created draft flows are displayed, one flow per row.

3. Click Create New Flow.

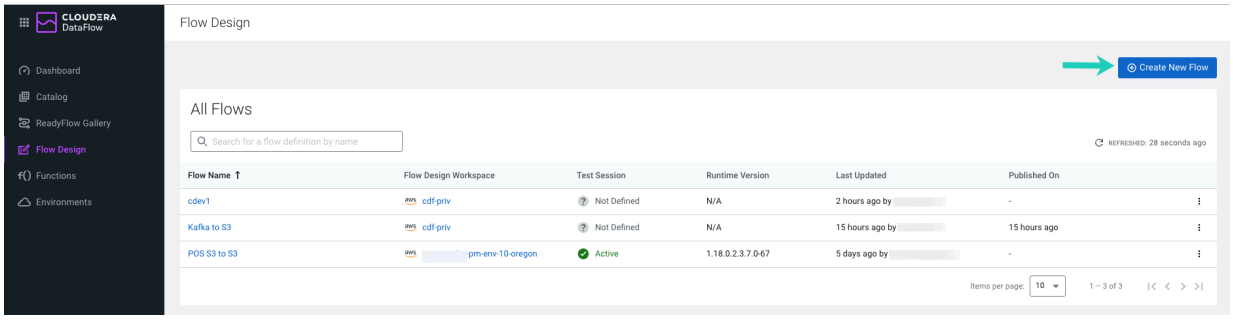

4. Select a Target Workspace where you want to create the draft.

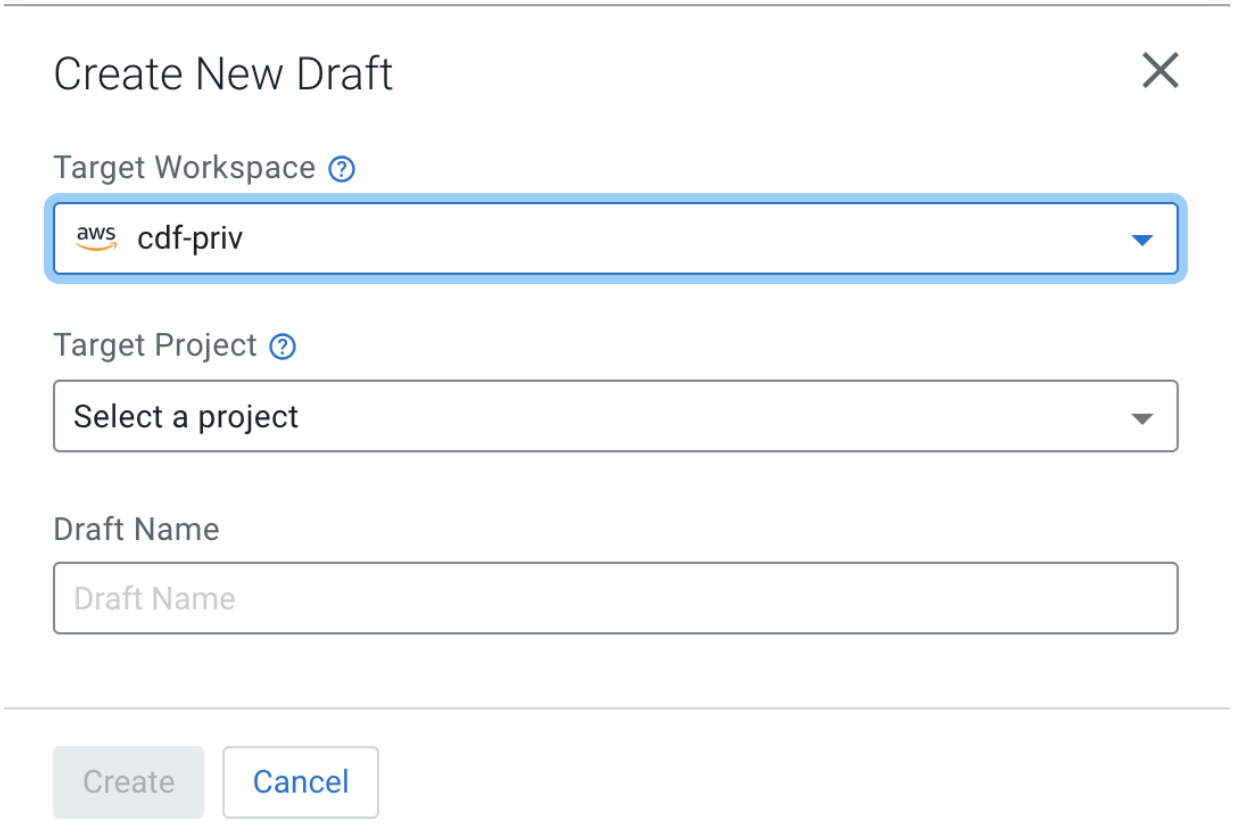

5. Under Target Project select Unassigned.
6. Provide a Draft Name.
   Provide Hello World.

7. Click Create.
   Flow Designer creates a default Process Group with the Draft Name you provided, 'Hello World' in this case, and you are redirected to the **Flow Design** canvas. The **Configuration** pane on the right displays configuration options for the default Process Group.

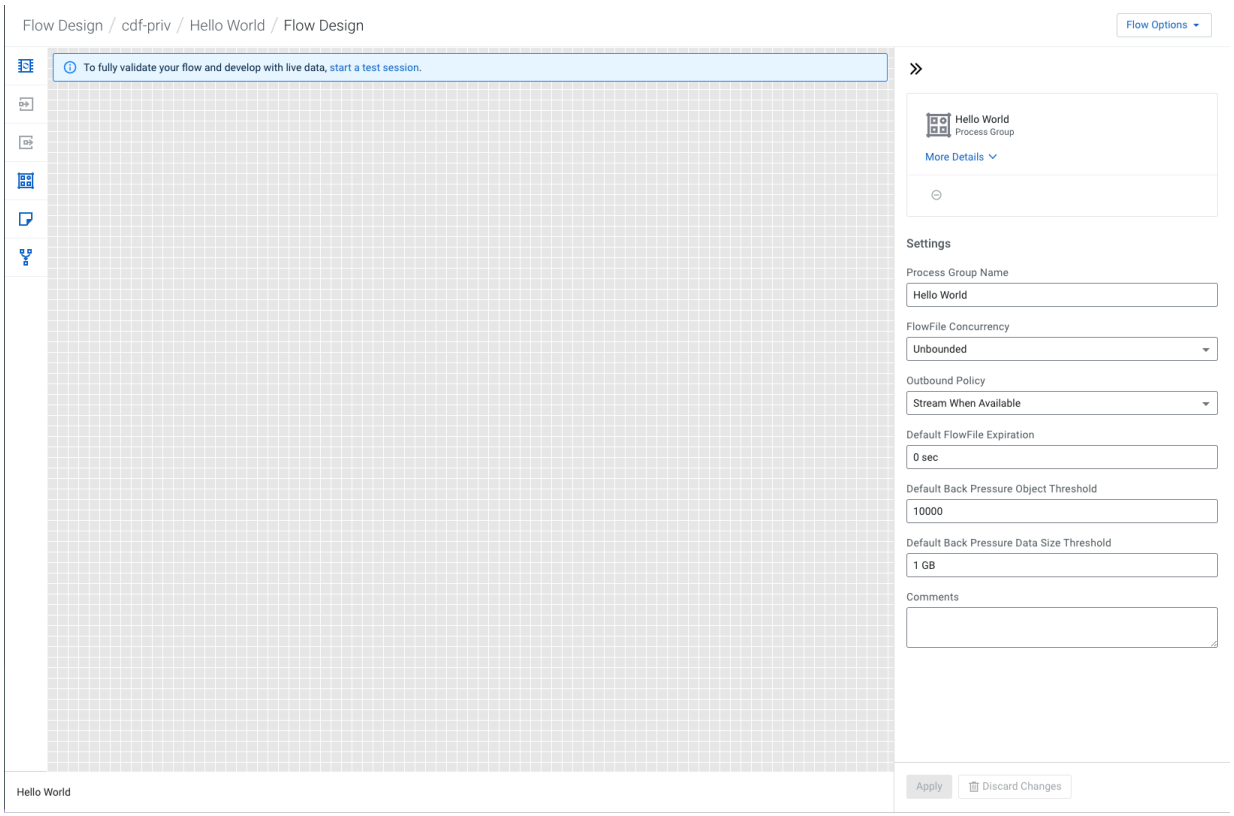

What to do next

Proceed to creating controller services.

Create controller services

Learn about creating Controller Services in Cloudera DataFlow Flow Designer.

About this task

Controller Services are extension points that provide information for use by other components (such as processors or other controller services). The idea is that, rather than configure this information in every processor that might need it, the controller service provides it for any processor to use as needed.

You will use the controller services you create now to configure the behavior of several processors you will add to your flow as you are building it.

Procedure

1. Go to Flow Options Services .

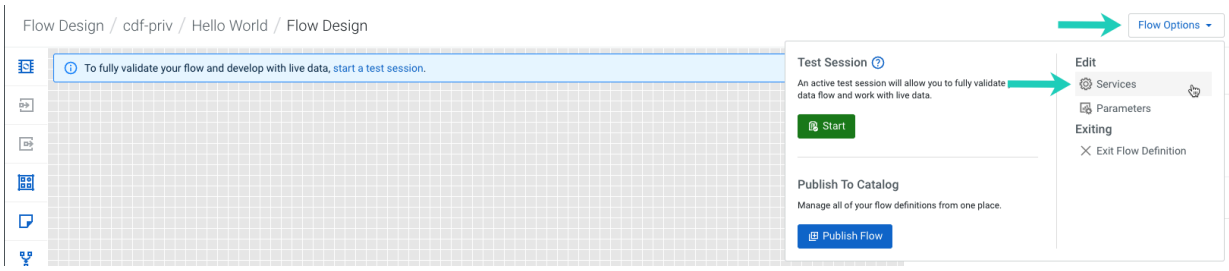

2. Click Add Service.
   The Add Service page opens.

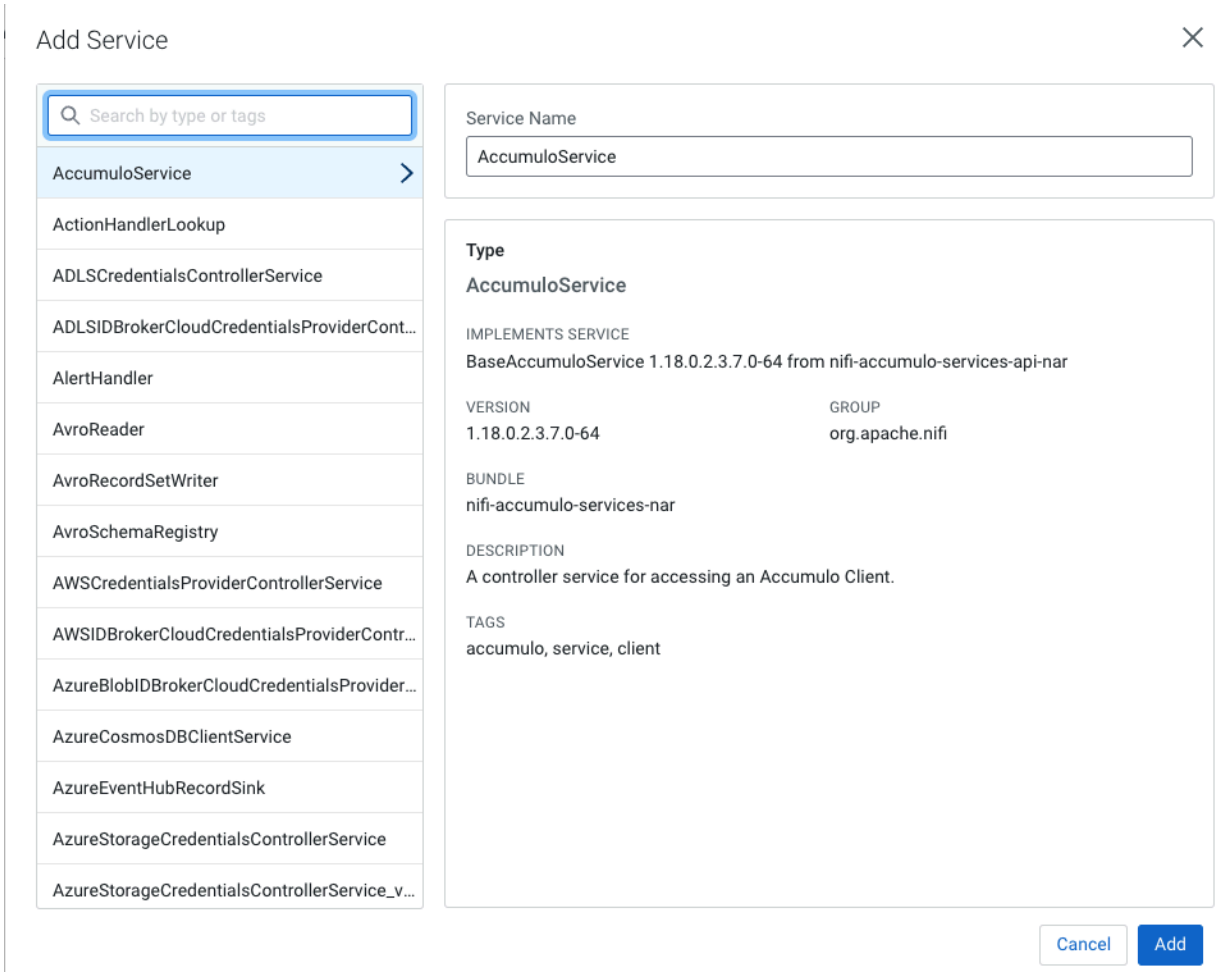

3. In the text box, filter for JsonTreeReader.

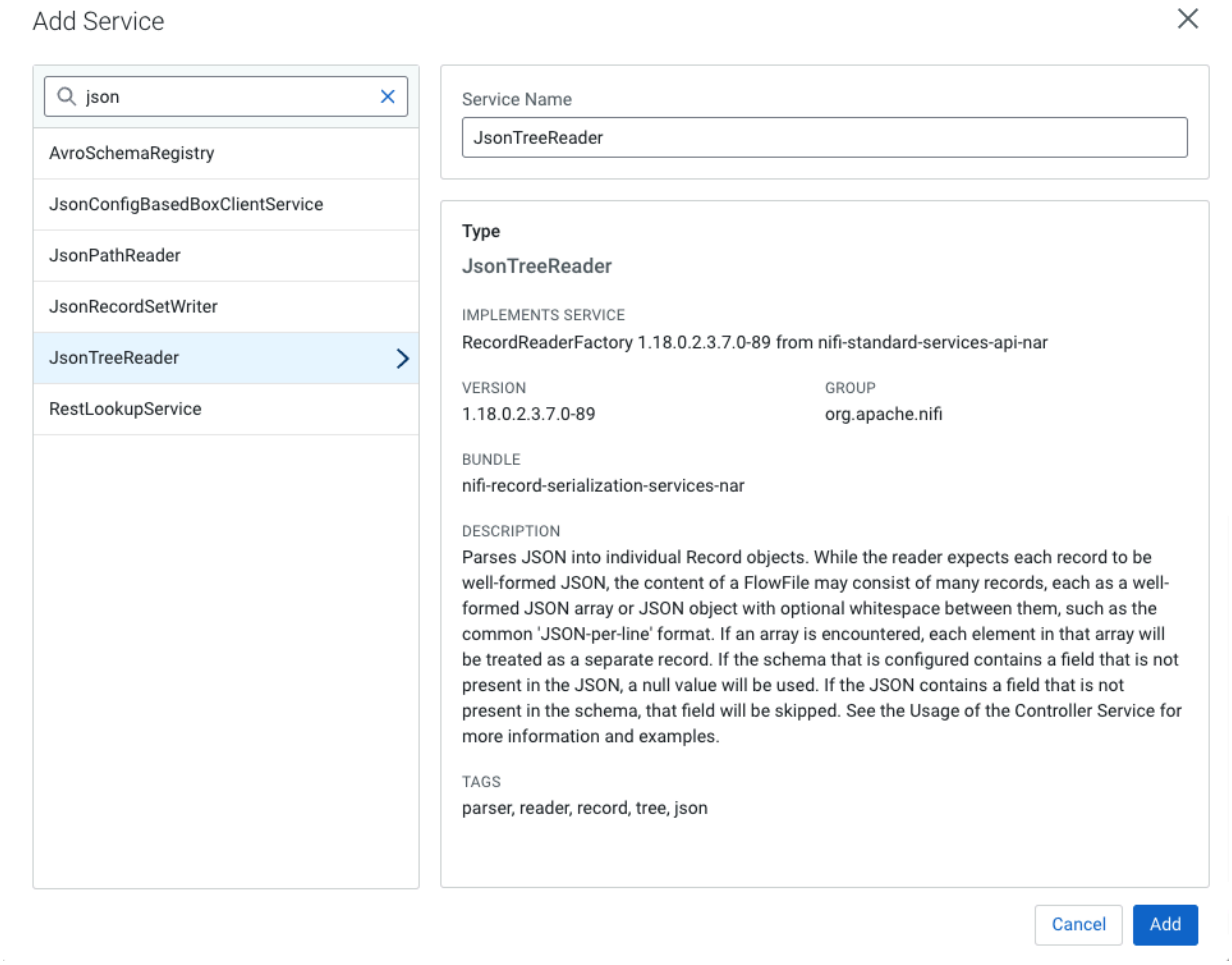

4. Provide Service Name: JSON_Reader_Recent_Changes.
5. Click Add.

6. Configure the JSON_Reader_Recent_Changes service.

   Set the following Properties:

   Starting Field Strategy

   Nested Field

   Starting Field Name

   recentchanges

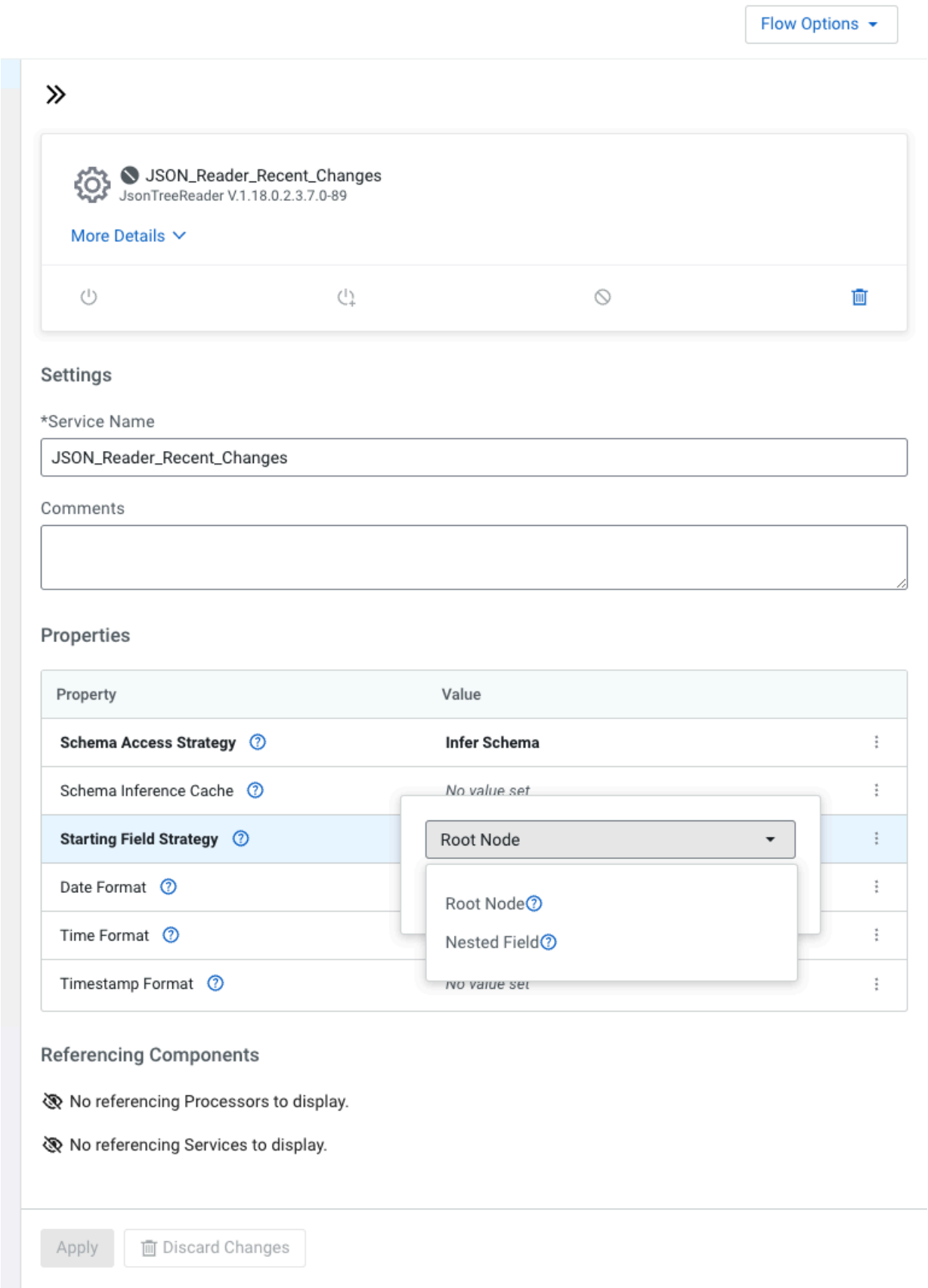

7. Click Apply.

8. Click Add Service to create another service.
9. In the text box, filter for AvroRecordSetWriter.
10. Provide Service Name: AvroWriter_Recent_Changes.
11. Click Add.

    You do not need to configure the AvroWriter_Recent_Changes service. You can leave all properties with their default values.
12. Click Add Service to create a third service.
13. In the text box, filter for AvroReader.
14. Provide Service Name: AvroReader_Recent_Changes.
15. Click Add.

    You do not need to configure the AvroReader_Recent_Changes service. You can leave all properties with their default values.
16. Click Back To Flow Designer to return to the flow design Canvas.

What to do next

After creating the necessary Controller Services, you can start building and configuring your flow.

Build your draft flow

Start building your draft flow by adding components to the Canvas and configuring them.

Procedure

1. Add an InvokeHTTP processor to the canvas.
   a) Drag a Processor from the Components sidebar to the canvas.

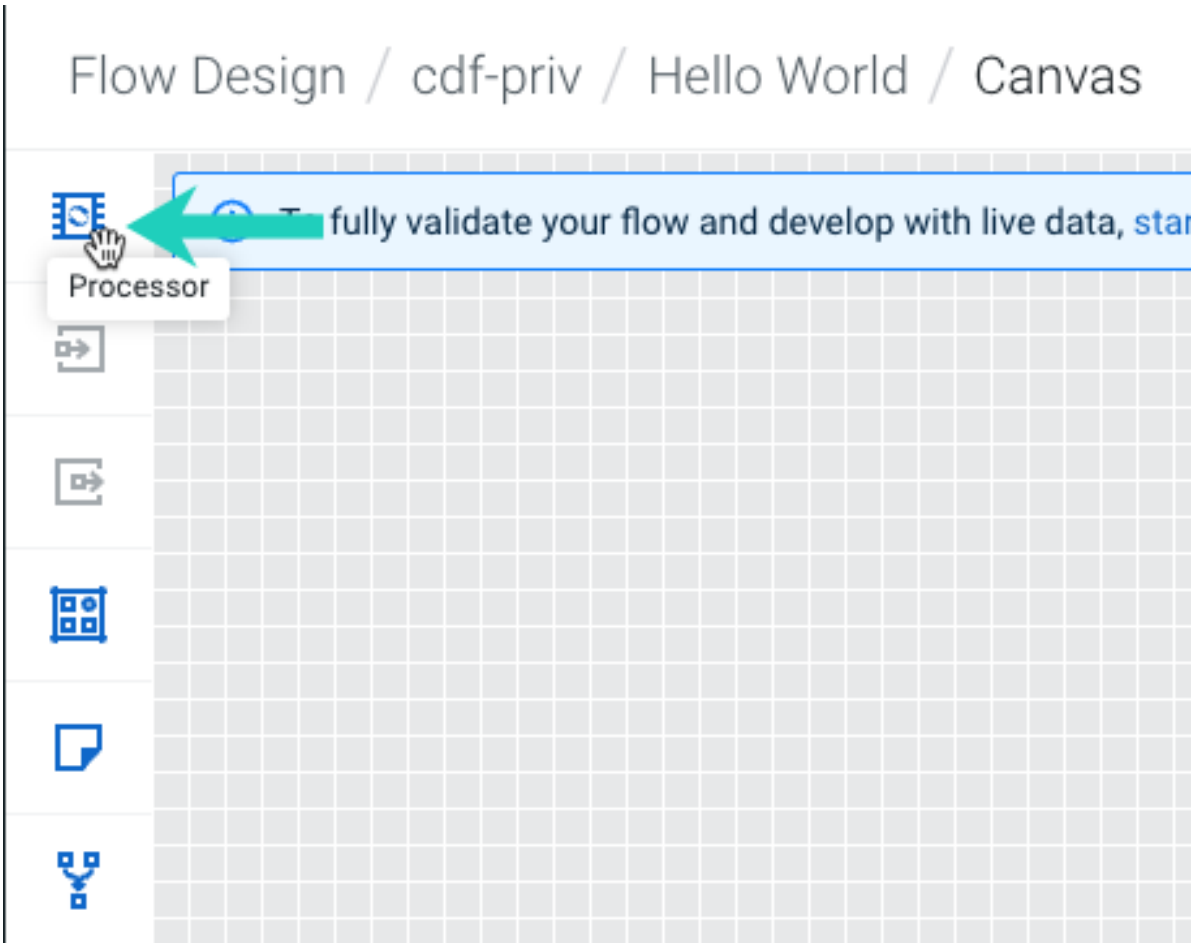

   b) In the text box, filter for InvokeHTTP.

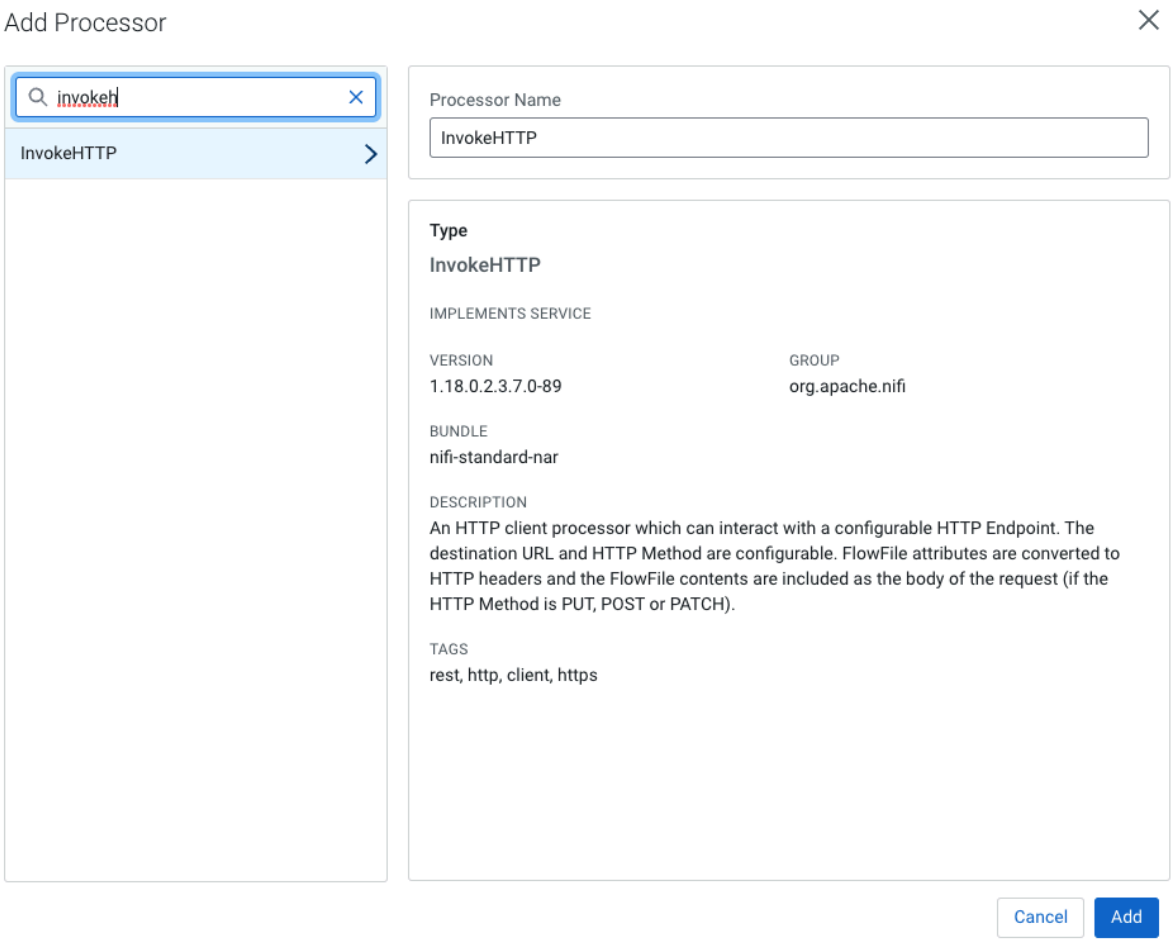

c) Change the Processor Name to Get Recent Wikipedia Changes.
d) Click Add.

After configuration, this processor calls the Wikipedia API to retrieve the latest changes.

2. Configure the Get Recent Wikipedia Changes processor.

   Properties

   HTTP URL

   provide https://en.wikipedia.org/w/api.php?action=query&list=recentchanges&format=json&rcprop=user%7Ccomment%7Cparsedcomment%7Ctimestamp%7Ctitle%7Csizes%7Ctags

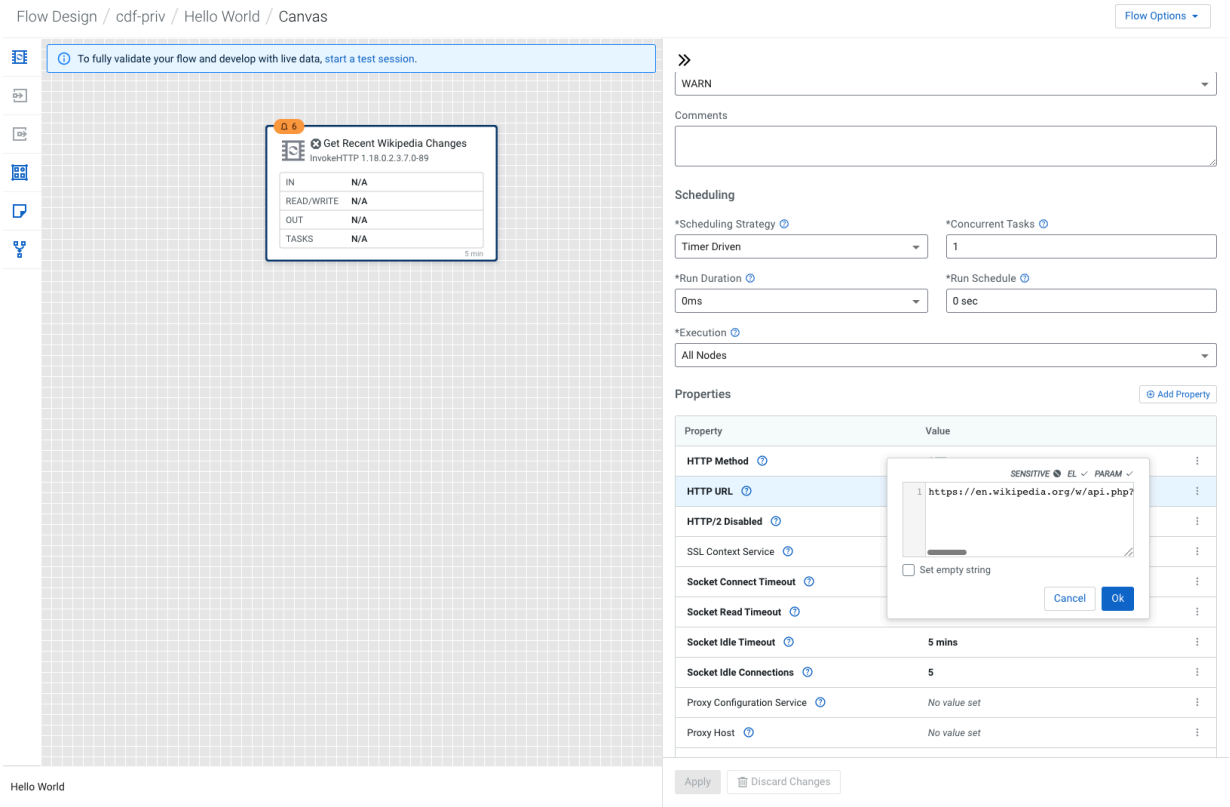

   Relationships

   Select the following relationships:

   - Original - Terminate
   - Failure - Terminate, Retry
   - Retry - Terminate
   - No Retry - Terminate

3. Click Apply.

Notice: Note the orange Notification Pill in the upper left corner of your processor. It is there to warn you about possible issues with a component.

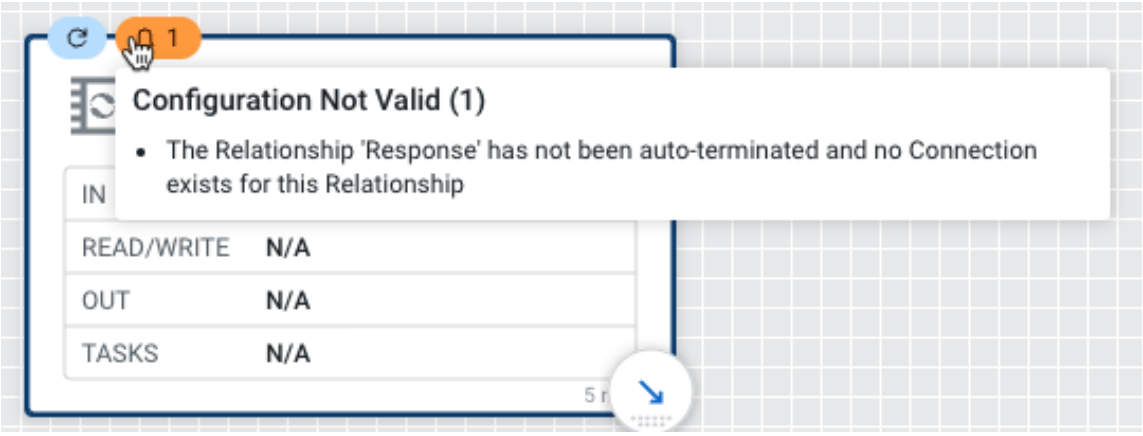

In this particular case, it warns you about one of the Relationships not being addressed when configuring your processor. Do not worry, it will disappear when you create a connection to another processor for the 'Response' relationship.

4. Add a ConvertRecord processor to the canvas.
   a) Drag a Processor from the Components sidebar to the canvas.
   b) In the text box, filter for ConvertRecord.
   c) Change the Processor Name to Convert JSON to AVRO.
   d) Click Add.

   This processor converts the JSON response to AVRO format. It uses RecordReaders and RecordWriters to accomplish this. It infers the JSON schema starting from the recent changes field.
5. Configure the Convert JSON to AVRO processor.

   Properties

   Record Reader

   Select the JSON_Reader_Recent_Changes controller service you have created from the drop-down list.

   Record Writer

   Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

   Relationships

   Select the following relationships:

   failure - Terminate, Retry
6. Click Apply.

7. Connect the Get Recent Wikipedia Changes and Convert JSON to AVRO processors by hovering over the lower-right corner of the Get Recent Wikipedia Changes processor, clicking the arrow that appears and dragging it to the Convert JSON to AVRO processor.

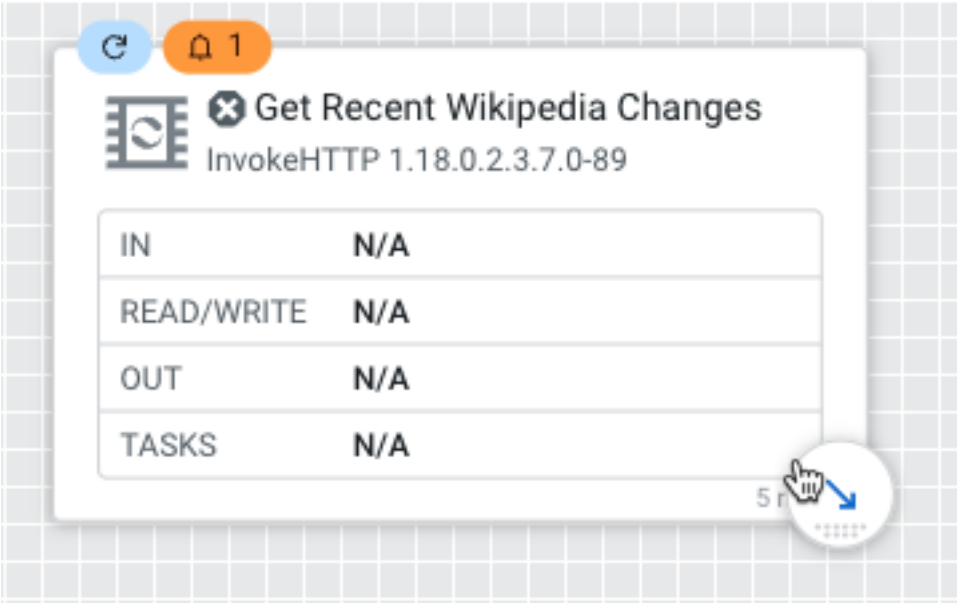

8. In the configuration popup, select the Response relationship and click Add.

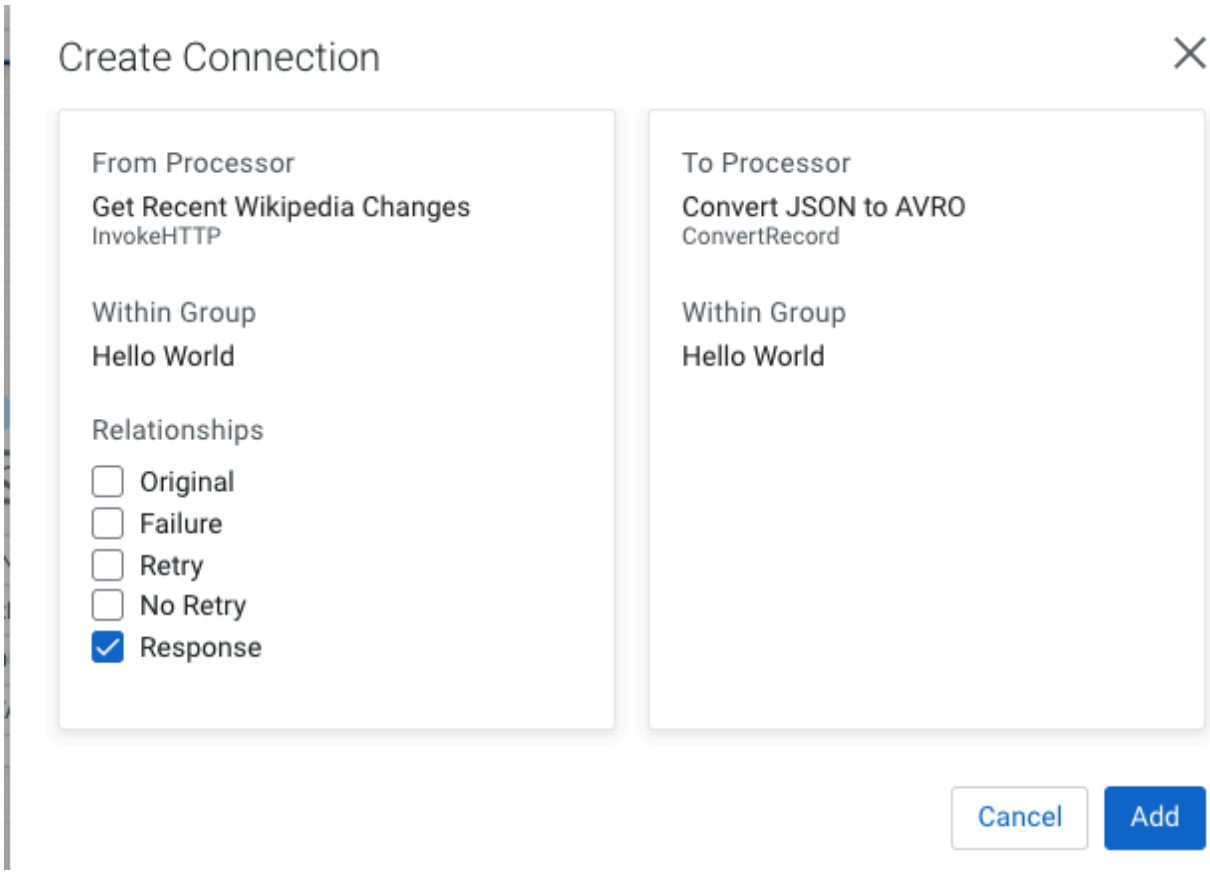

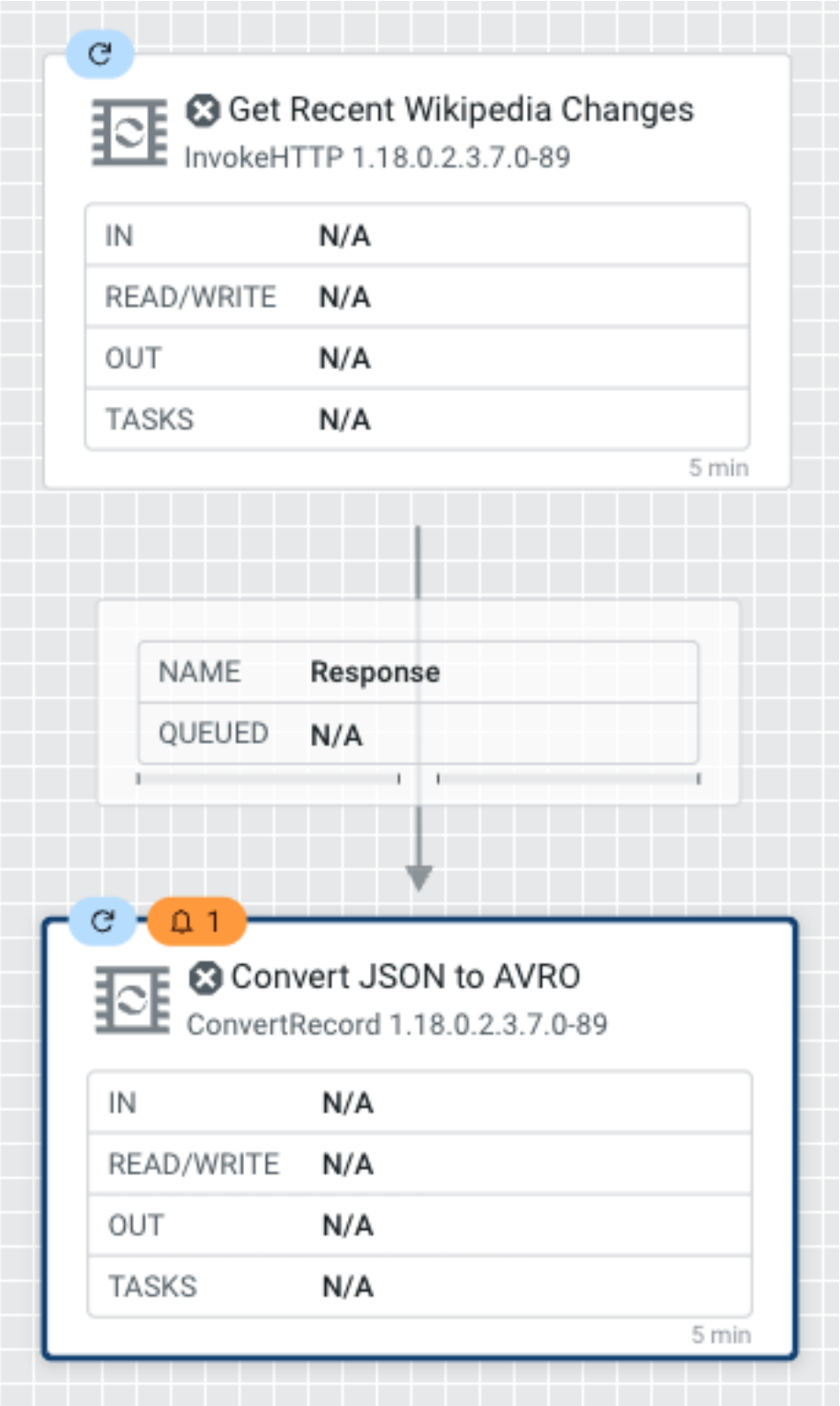

Notice: Note, that the Notification Pill warning about the unconfigured 'Response' relationship disappeared from your Get Recent Wikipedia Changes processor.

9. Add a QueryRecord processor.
   a) Drag a Processor from the Components sidebar to the canvas.
   b) In the text box, filter for QueryRecord.
   c) Name it Filter Edits.
   d) Click Add.

   This processor filters out anything but actual page edits. To achieve this, it's running a query that selects all FlowFiles (events) of the type edit.
10. Configure the Filter Edits processor.

    Properties

    Record Reader

    Select the AvroReader_Recent_Changes controller service you have created from the drop-down list.

    Record Writer

    Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

    Relationships

    Select the following relationship:

    - failure - Terminate
    - original - Terminate
11. For the Filter Edits processor you also need to add a user-defined property. Click Add Property.
    a) Provide Filtered edits as Name
    b) Provide Select * from FLOWFILE where type='edit' as Value.
    c) Click Apply.
12. Connect the Convert JSON to AVRO and Filter Edits processors by hovering over the lower-right corner of the Convert JSON to AVRO processor, clicking the arrow that appears and dragging it to the Filter Edits processor.
13. In the configuration pane, select the success and failure relationships and click Add.
14. Add a second QueryRecord processor.
    a) Drag a Processor from the Components sidebar to the canvas.
    b) In the text box, filter for QueryRecord.
    c) Name it Route on Content Size.
    d) Click Add.

    This processor uses two SQL statements to separate edit events that resulted in a longer article from edit events that resulted in a shorter article.
15. Configure the Route on Content Size processor.

    Properties

    Record Reader

    Select the AvroReader_Recent_Changes controller service you have created from the drop-down list.

    Record Writer

    Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

    Relationships

    Select the following relationships:

    - failure - Terminate, Retry
    - original - Terminate

16. For the Route on Content Size processor you also need to add two user-defined properties.
    a) Click Add Property.
    b) Provide Added content as Name
    c) Provide Select * from FLOWFILE where newlen>=oldlen as Value.
    d) Click Add.
    e) Click Add Property, to create the second property.
    f) Provide Removed content as Name.
    g) Provide Select * from FLOWFILE where newlen<oldlen as Value.
    h) Click Add.
    i) Click Apply.
17. Connect the Filter Edits and Route on Content Size processors by hovering over the lower-right corner of the Filter Edits processor, clicking the arrow that appears and drawing it to Route on Content Size.

    In the Create Connection pop up select the Filtered edits relation and click Add.
18. Add two MergeRecord processors.
    a) Drag a Processor from the Components sidebar to the canvas.
    b) In the text box, filter for MergeRecord.
    c) Name it Merge Edit Events.
    d) Click Add.
    e) Repeat the above steps to add another identical processor.

    These processors are configured to merge at least 100 records into one flowfile to avoid writing lots of small files. The MaxBinAge property is set to 2 minutes which makes the processors merge records after two minutes even if less than 100 records have arrived.
19. Configure the two Merge Edit Events processors.

    Properties

    Record Reader

    Select the AvroReader_Recent_Changes controller service you have created from the drop-down list.

    Record Writer

    Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

    Max Bin Age

    Set to two minutes by providing a value of 2 min.

    Relationships

    Select the following relationships:

    - failure - Terminate
    - original - Terminate

    Note: Do not forget to perform configuration for both Merge Edit Events processors.

20. Connect the Route on Content Size processor to both of the Merge Edit Events processors.

a) For the first Merge Edit Events processor, select Added content from Relationships and click Add.

b) For the second Merge Edit Events processor, select Removed content from Relationships and click Add.

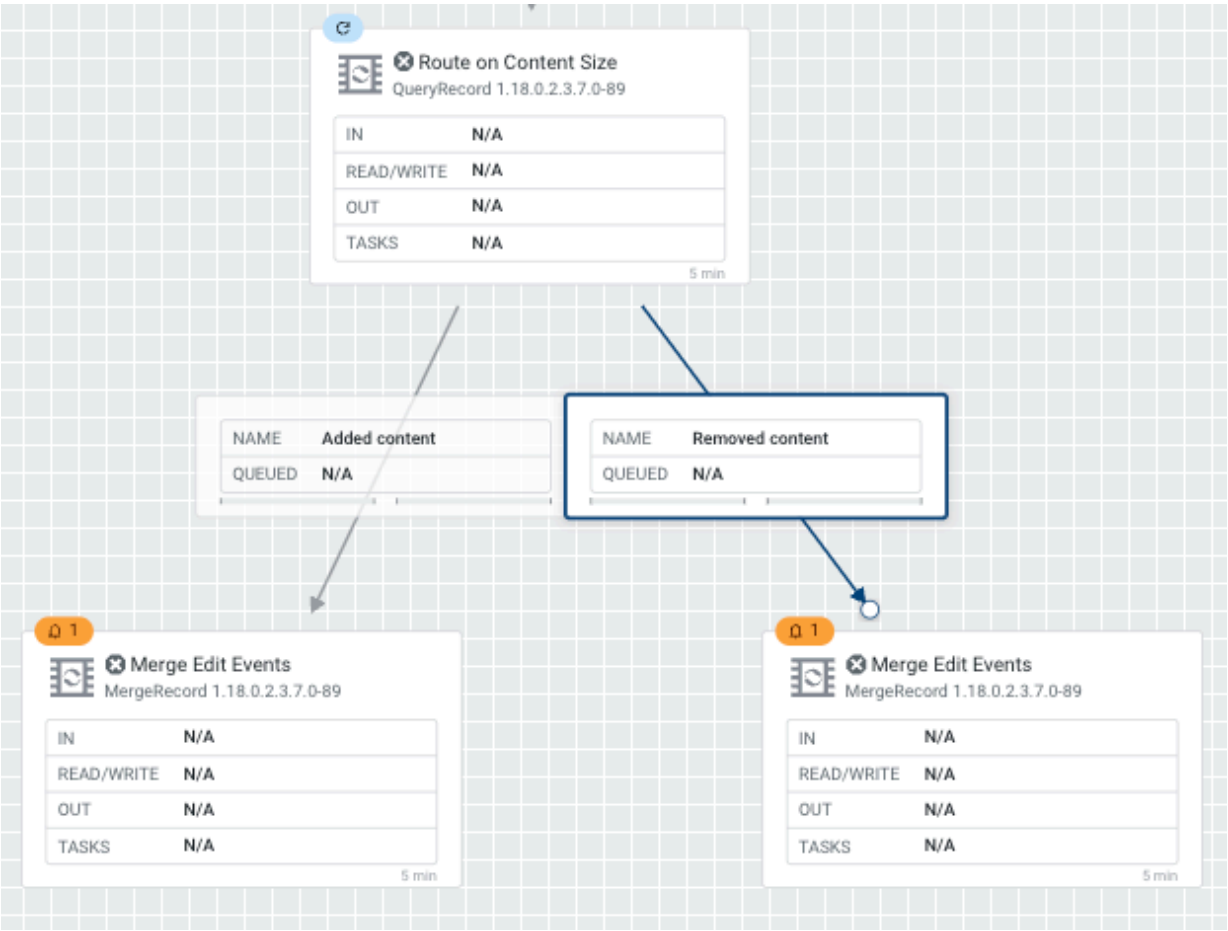

21. Add two PutFile processors to the canvas.

a) Drag a Processor from the Components sidebar to the canvas.

b) In the text box, filter for PutFile.

c) Name it Write "Added Content" Events To File.

d) Click Add.

e) Repeat the above steps to add another identical processor, naming this second PutFile processor Write "Removed Content" Events To File.

These processors write the filtered, routed edit events to two different locations on the local disk. In Cloudera DataFlow, you would typically not write to local disk but replace these processors with processors that resemble your destination system (Kafka, Database, Object Store etc.)

22. Configure the Write "Added Content" Events To File processor.

    Properties:

    Directory

    Provide /tmp/larger_edits

    Maximum File Count

    Provide 500

    Relationships

    Select the following relationships:

    - SUCCESS - Terminate
    - failure - Terminate
23. Click Apply.
24. Configure the Write "Removed Content" Events To File processor.

    Properties:

    Directory

    Provide /tmp/smaller_edits

    Maximum File Count

    Provide 500

    Relationships

    Select the following relationships:

    - SUCCESS - Terminate
    - failure - Terminate
25. Click Apply.
26. Connect the Merge Edit Events processor with the Added content connection to the Write "Added Content" Events To File processor.

    In the Create Connection pop up select merged and click Add.
27. Connect the Merge Edit Events processor with the Removed content connection to the Write "Removed Content" Events To File processor.

    In the Create Connection pop up select merged and click Add.

Results

Congratulations, you have created your first draft flow. Now proceed to testing it by launching a Test Session.

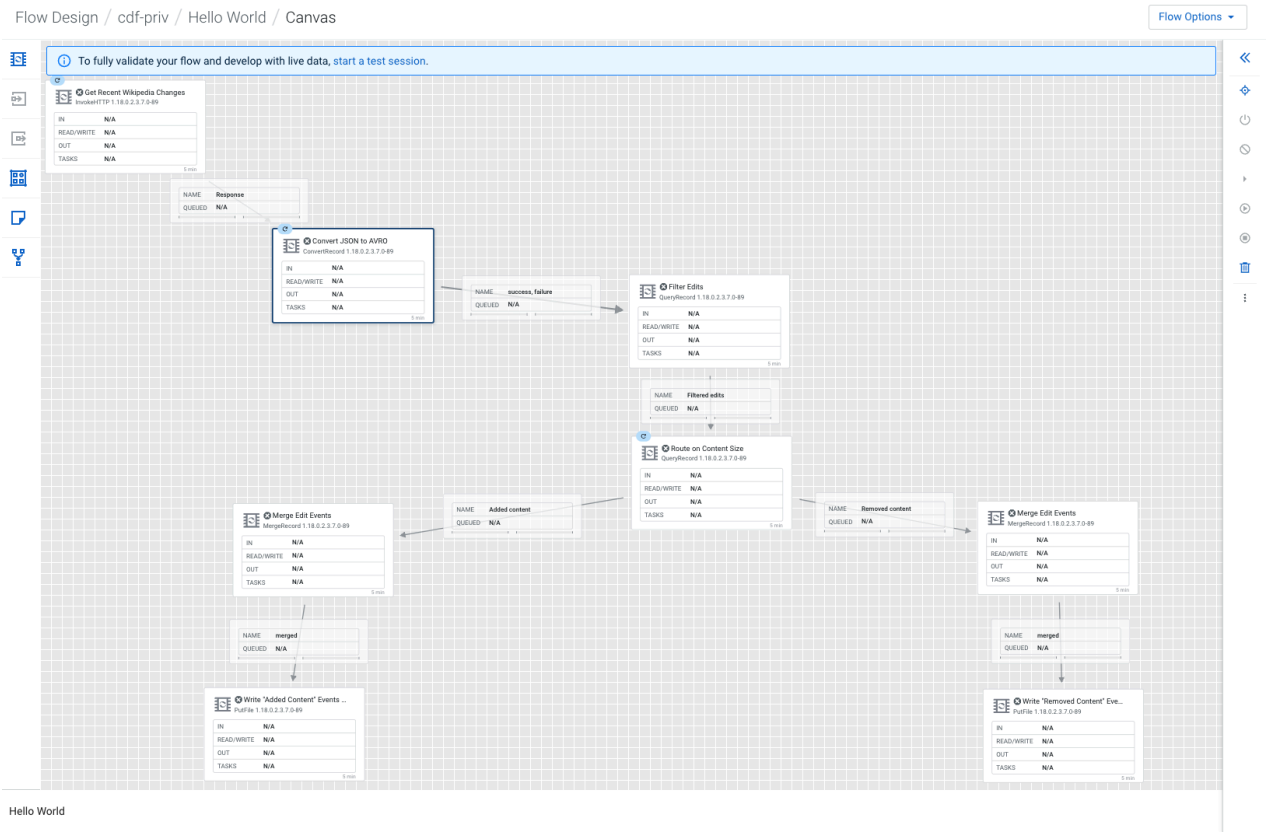

Start a test session

To validate your draft, start a test session. This provisions an Apache NiFi cluster where you can test your draft.

About this task

Starting a Test Session provisions NiFi resources, acting like a development sandbox for a particular draft. It allows you to work with live data to validate your data flow logic while updating your draft. You can suspend a test session any time and change the configuration of the NiFi cluster then resume testing with the updated configuration.

Procedure

1. Click start a test session in the banner on top of the Canvas.

2. Click Start Test Session.

   Test Session status Initializing Test Session... appears on top of the page.

3. Wait for the status to change to

   Active Test Session
   This may take several minutes.

4. Click Flow Options Services to enable Controller Services.

5. Select a service you want to enable, then click Enable Service and Referencing Components.

   This option does not only enable the controller service, but also any component that references it. This way, you do not need to enable the component separately to run the test session. In the context of this tutorial, enabling the 'AvroReader_Recent_Changes' controller service will also enable 'Filter Edits', 'Route on Content Size', and 'Merge Edit Events' processors as well.

   Repeat this step for all Controller Services.
6. Click Back To Flow Designer to return to the flow design Canvas.
7. Start the Get Recent Wikipedia Changes, Write "Added Content" Events To File, and Write "Removed Content" Events To File components by selecting them on the Canvas then clicking Start.

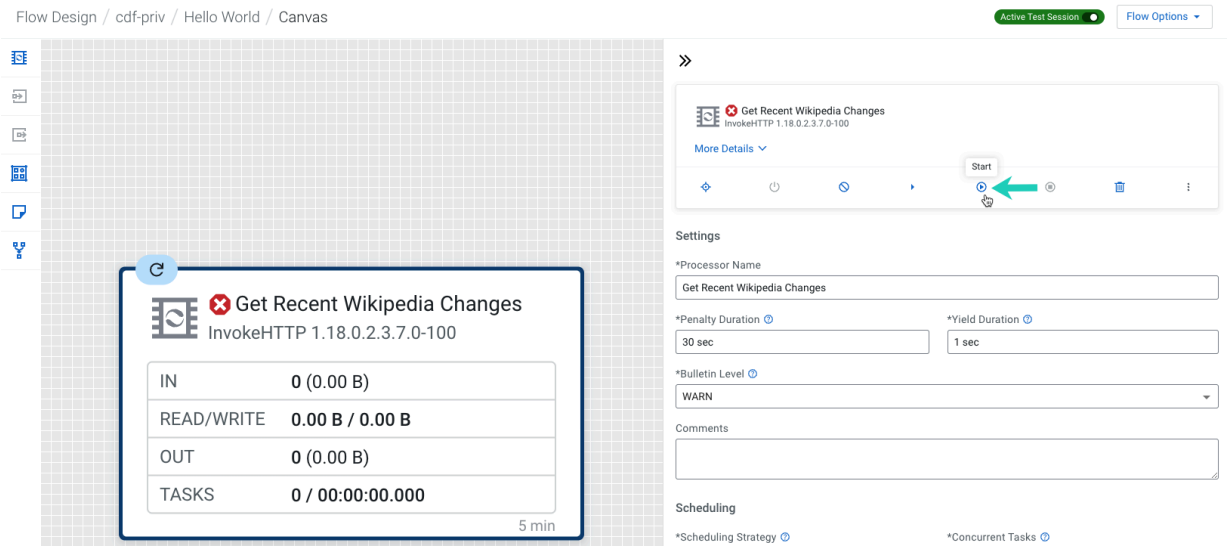

All other components were auto-started when you selected the Enable Service and Referencing Components option.

8. Observe your first draft flow processing data.

   On the Flow Design Canvas you can observe statistics on your processors change as they consume and process data from Wikipedia. You can also observe one or more blue Notification Pills, providing information about the current task.

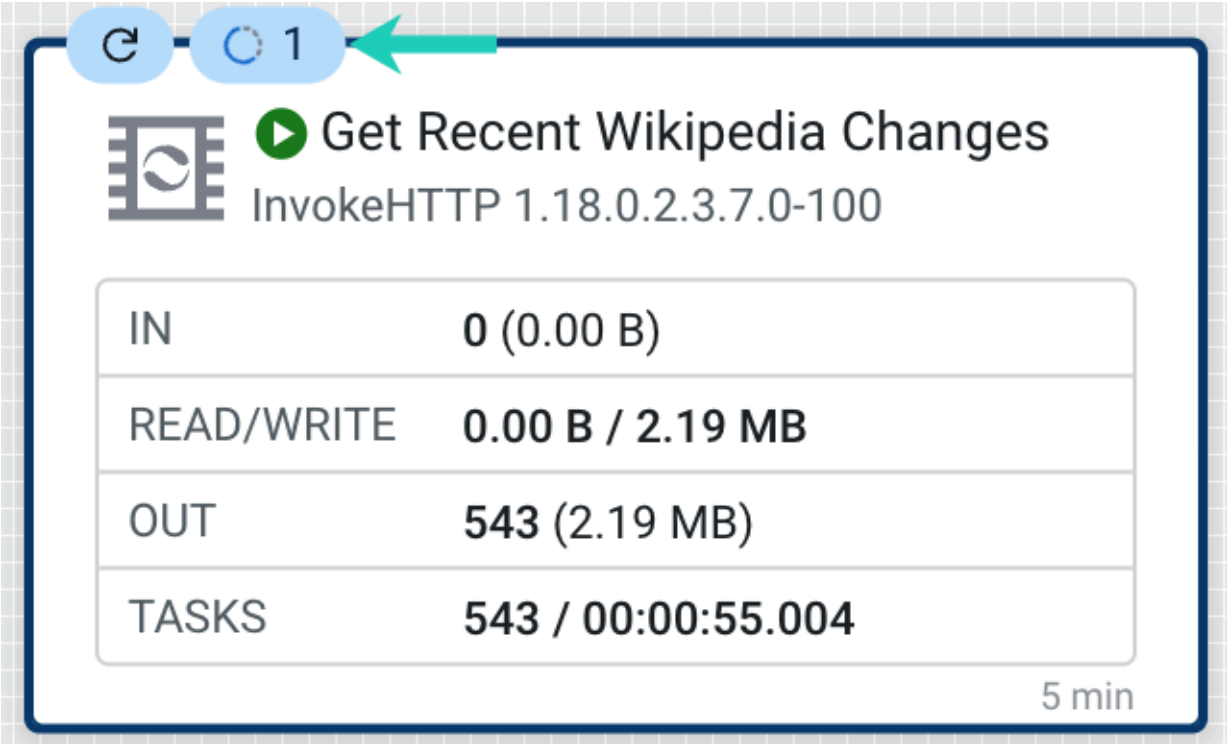

Publish your flow definition to the Catalog

Now that you have tested your draft and it works fine, you can go on and publish it to the Catalog as a flow definition so that you can create a Cloudera DataFlow deployment.

Procedure

1. On the **Flow Designer** canvas, click Flow Options Publish To Catalog Publish .
2. Fill in the fields in the **Publish A New Flow** box.
   - Provide a Flow Name for your flow definition.

     You can only provide a name when you publish your flow for the first time.
   - Optionally provide a Flow Description.

     You can only provide a description when you publish your flow for the first time.
   - Optionally provide Custom Tags.

     You can filter flow definition versions by tags in the Catalog.
   - Optionally provide Version Comments.
3. Click Publish.

Results

Your draft is published to the Catalog as a flow definition.

Cloudera DataFlow

Troubleshooting

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Overview 4

Troubleshooting errors that occur when enabling for an environment fails 4

Troubleshooting errors that occur when disabling for an environment fails 8

Troubleshooting errors that occur after successful enablement 8

Troubleshooting flow deployment errors 10

Collecting and sending a diagnostic bundle using the CLI 13

Collecting and sending a diagnostic bundle using Unified Diagnostics 16

Downloading NiFi application log 19

Overview

You may encounter some common errors while using , and it is useful to understand how to recognize and correct them.

Troubleshooting errors that occur when enabling for an environment

Learn how to recognize and correct common errors that occur when you are enabling for an environment.

Troubleshooting errors that occur when is disabled for an environment

Learn how to recognize and correct common errors that occur when you are disabling for an environment.

Troubleshooting errors that occur when is enabled for an environment

Learn how to recognize and correct common errors with environments for which has been enabled.

Troubleshooting errors that occur during flow deployment

Learn how to recognize and address common errors with your flow deployments.

Troubleshooting errors that occur when enabling for an environment fails

Learn how to recognize and correct common errors that occur when you are enabling for an environment.

Your ability to deploy flow definitions depends on being enabled and in good health in your target environment. If a deployment is unhealthy in an environment, it is typically because enabling or disabling for an environment has failed. Review the environment troubleshooting information to understand common errors and their solutions.

When you enable for an existing environment, creates the required infrastructure in your cloud environment and installs core services as well as the software in the Kubernetes cluster. The enablement process can be divided into three parts:

1. Provisioning Cloud Infrastructure
2. Installing core software and services
3. Installing software and services

To understand why the enablement process is failing, the first step is to identify at which stage the enablement process failed.

Identifying where the enablement process failed

Use the information provided in the hover state of an environment and the status messages that have been logged in the environment's Event History to identify where the enablement process has failed.

1. In DataFlow, navigate to the Environments page and find the environment where the enablement process failed.
2. Hover over the status icon and take note of the error message.
3. Click the environment, select the Alerts tab, and review the error and info events that have been logged during enablement.

Enablement fails during infrastructure provisioning or core software installation

If you only see a status message in the Event History indicating that the infrastructure provisioning has started but you cannot see a corresponding status message confirming that the infrastructure has been provisioned successfully, was either not able to create the required infrastructure or install the core software and services afterwards.

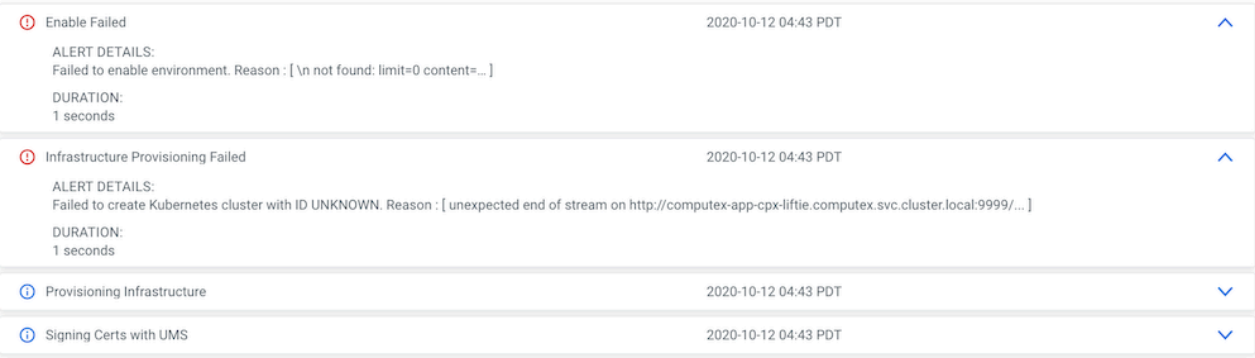

The infrastructure provisioning failed error message indicates that there was either an issue with creating AWS infrastructure or setting up core services.

Validating infrastructure creation – for AWS users

uses CloudFormation scripts to create the required infrastructure in your AWS account. To validate whether the requested resources have been created successfully, log in to your AWS account, navigate to CloudFormation and search for the Kubernetes cluster ID that you have extracted from the environment events and looks similar to liftie-q4nlzm5p. Verify that the CloudFormation scripts completed successfully.

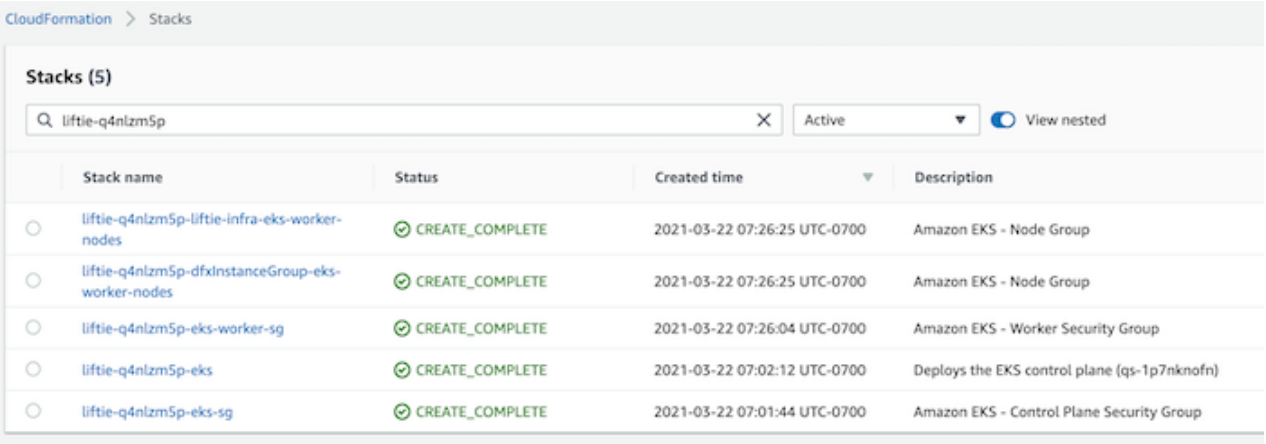

If the CloudFormation script did not complete successfully, make sure that the cross account role for your environment has been assigned appropriate permissions.

If the CloudFormation script completed successfully but enabling failed before completing the infrastructure setup, this might be an indication that the Kubernetes cluster cannot communicate with the control plane or other public endpoints like container image repositories. Make sure that the VPC and subnets you are using for meet the and prerequisites.

Validating infrastructure creation – for Azure users

uses Azure Resource Manager to orchestrate infrastructure creation in Azure. When you enable for a environment, a Cluster ID is generated that can be used to track all associated resources in Azure. Navigate to Environments in , select your Azure environment and copy the Cluster ID, which will look similar to liftie-fq6hq9sc.

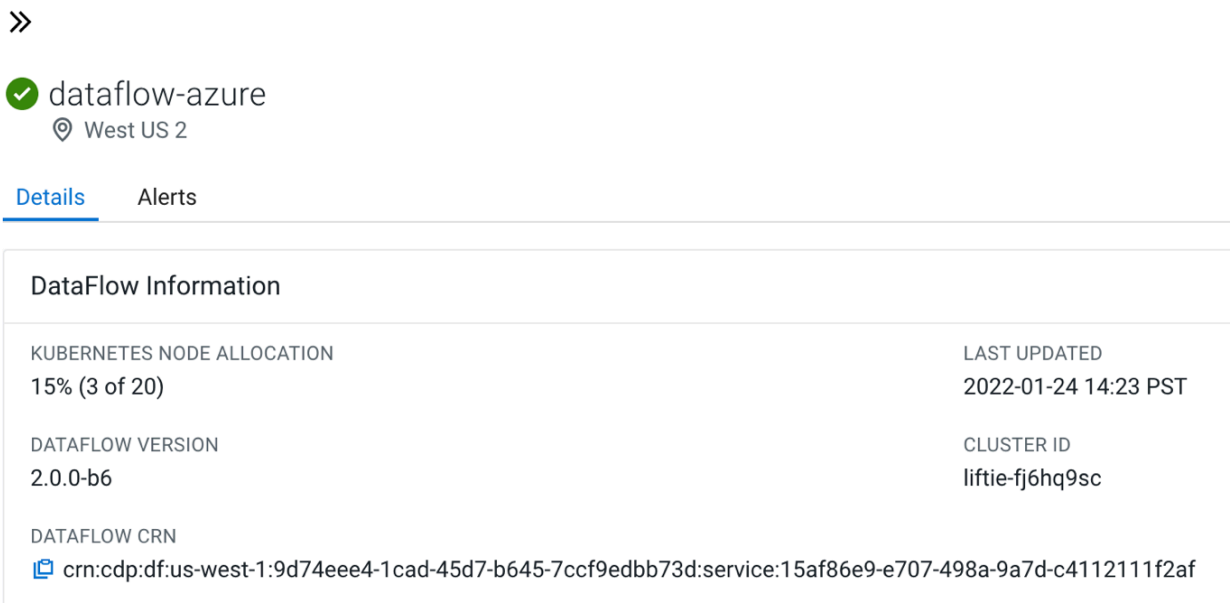

First, make sure that ’s Azure PostgreSQL database has been created successfully. In the Azure Portal, navigate to your resource group, explore Deployments under Settings, and find the database deployment associated with the previously obtained Cluster ID.

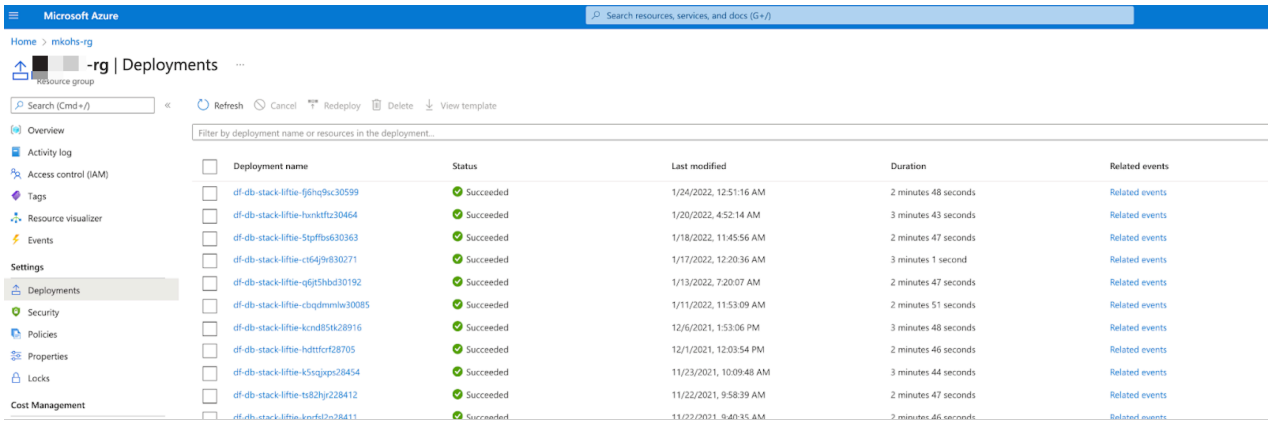

Next, validate that the AKS Kubernetes cluster and associated infrastructure has been created successfully. In the Azure Portal, use the Cluster ID to search for associated resources in your Azure subscription.

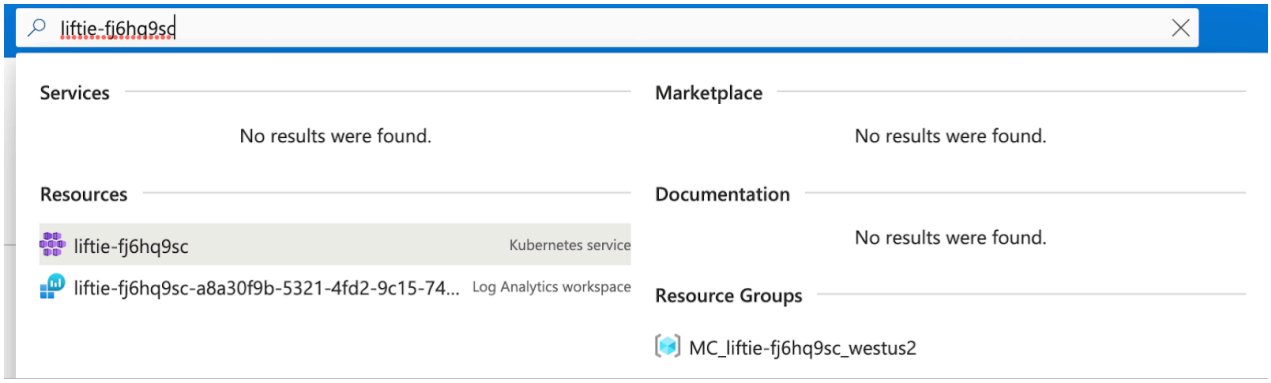

You should find a Kubernetes service as well as a new resource group called MC_<Cluster ID>_<AZURE Region>, which contains the virtual machine scale sets, load balancers, IP addresses, and storage disks that have been created for the Kubernetes cluster.

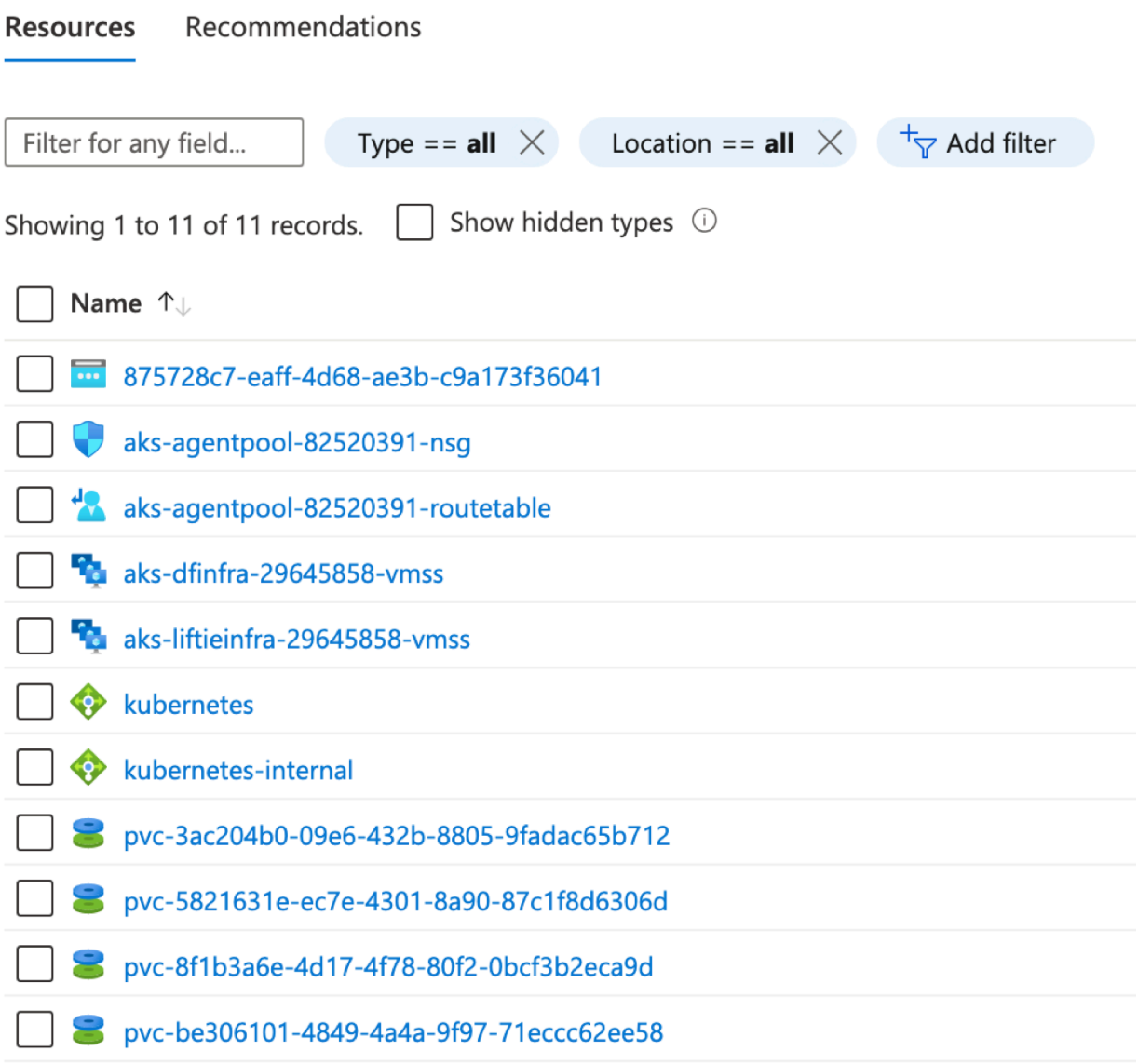

If the resources were not created successfully, make sure that the app-based credential for your environment has been assigned the appropriate permissions. For more information, see *Prerequisites for the provisioning credential*.

If the infrastructure resources were created successfully, but enabling failed before completing the infrastructure setup, it might be an indication that the Kubernetes cluster cannot communicate with the control plane or other public endpoints like container image repositories. Make sure that the virtual network and subnets you are using for meet the and prerequisites. For more information, see *VNet and subnets*.

Enablement fails during software and service installation

If you see a status message that indicates that the required Infrastructure has been provisioned successfully but the enablement process still failed, this is an indication that installing and setting up the software and services has failed.

The Infrastructure Provisioned status event indicates that the Kubernetes cluster has been created and core services have been setup successfully.

To ensure that this is not a transient issue, use the Retry Enablement action to start the enablement process again. Retry Enablement terminates all existing resources and provisions new infrastructure.

If retrying does not help and enabling still fails after successfully provisioning the infrastructure, copy the error message from the Event History and open a support case with .

Related Concepts

Overview

Related Information

Prerequisites for the provisioning credential

VNet and subnets

Troubleshooting errors that occur when disabling for an environment fails

Learn how to recognize and correct common errors that occur when you are disabling for an environment.

can be in Bad Health due to a failed disablement process. If you see error messages in the Event History indicating that disabling has failed, use the Retry Disable Process to try again to ensure that the issue is not transient.

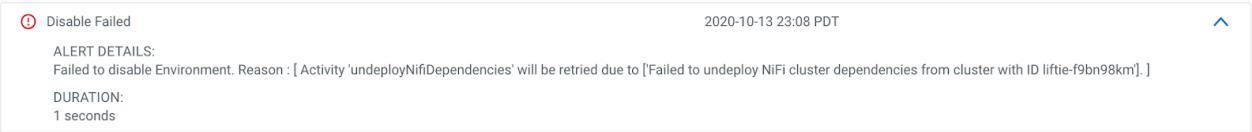

If the issue persists, select Environments Action Reset Environment to clear ’s state for the environment. Resetting the Environment allows you to enable for the same environment again at a later stage.

Note:

Resetting does not terminate the associated cloud infrastructure. After resetting for an environment, you must manually delete the cloud resources that were created during enablement.

Related Concepts

Overview

Troubleshooting errors that occur after successful enablement

Learn how to recognize and correct common errors with environments for which has been enabled.

After you have successfully enabled for an environment there are several reasons why 's health can become Concerning or Bad.

Concerning health due to “Workload Failed to Heartbeat”

If health is concerning for one of your environments, hover over the status icon and check the Alerts tab in the environment details to see details about the issue. Workload Failed to Heartbeat means that the Control Plane has not received a recent heartbeat from the workload application running in your cloud account.

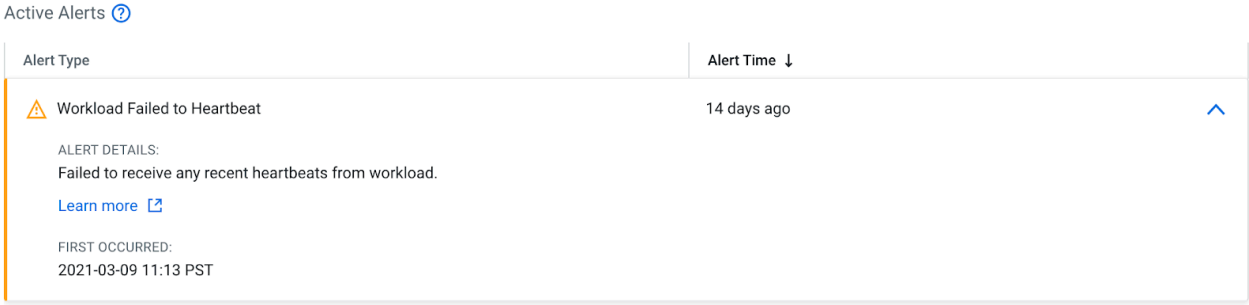

fails to receive heartbeats from a particular environment

Heartbeat failures can have several reasons. Make sure that:

- The associated environment has been started and is running.
- There was no networking related change in your VPC/subnet configuration and that your networking setup still meets the requirements outlined in the *Networking*.

If the issue persists, open a support case with .

Concerning Health due to Nearing Maximum Kubernetes Limit

When your Kubernetes cluster is close to its maximum node count, will show Concerning Health for the particular environment and display an Active Alert with more details about the boundaries. To return to Good Health you can adjust the maximum node number through the Edit Configuration option in the Environment actions menu.

Bad Health due to issues with the associated environment

Certain issues with the associated environment will result in reporting Bad Health for an environment. Once is in Bad Health you can no longer create or termnate flow deployments in the environment.

Bad Health due to environment state

If the associated environment is either unhealthy or in a starting/stopping state, will report Bad Health. To return to Good Health make sure that:

- The associated environment has been started and is running.
- FreeIPA and DataLake are both running.

Bad Health due to environment having been deleted

If the associated environment has been deleted without disabling first, will report Bad Health indicating that it has been orphaned and required services such as FreeIPA are no longer available in the environment.

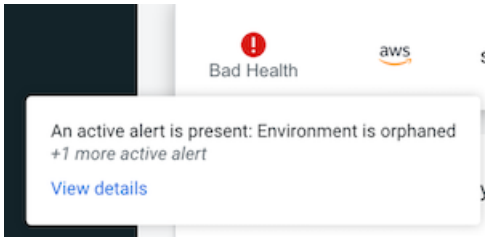

You are not able to create new Flow Deployments for . You cannot recover health in this situation. Terminate your Flow Deployments and use the Disable Environment action to terminate and associated cloud infrastructure and enable again for a different environment.

Related Concepts

Overview

Troubleshooting flow deployment errors

Learn how to recognize and address common errors with your flow deployments.

Setting up `kubectl` to connect to the Kubernetes cluster

It is helpful to have access to the Kubernetes cluster using command line tools such as `kubectl` when you are troubleshooting deployment or upgrade failures. To set up `kubectl` access, follow these steps:

1. In , from the Environments page, select the service for which you want to add or remove user access.
2. Click Manage DataFlow.
3. From the Actions menu, click Manage Kubernetes API Server User Access.
4. Add the AWS IAM role that you will authenticate as to the list of authorized users for DataFlow by entering the ARN in the Add User dialog.
5. Use the Download Kubeconfig action to retrieve the kubeconfig file for connecting to your cluster.
6. Set up your `kubectl` to use the downloaded kubeconfig file:

```
export KUBECONFIG=[***PATH/TO/DOWNLOADED/KUBECONFIG/FILE***]
```

7. Run kubectl get ns and validate that your output looks similar to:

```
NAME                        STATUS   AGE
cadence                     Active   37h
cert-manager                Active   37h
cfm-operator-system         Active   37h
default                     Active   38h
dfx-dev-environment-ns      Active   36h
dfx-idbrokers3-ns           Active   36h
dfx-kafkatos3-ns            Active   22h
dfx-local                   Active   37h
dfx-ops                     Active   37h
kube-node-lease             Active   38h
kube-public                 Active   38h
kube-system                 Active   38h
liftie                      Active   37h
logging                     Active   37h
monitoring                  Active   37h
nfs-provisioner-system      Active   37h
nginx-ingress               Active   37h
prometheus-operator-system  Active   37h
```

With `kubectl` being set up correctly, you are able to access NiFi and other logs directly through the CLI.

Understanding flow deployment failures

The Flow Deployment process consists of two phases:

1. Scheduling resources on Kubernetes and creating a new NiFi cluster:

NiFi Cluster Provisioned	2021-03-22 22:59 PDT
Provisioning NiFi Cluster	2021-03-22 22:58 PDT
Deployment Initiated	2021-03-22 22:58 PDT

2. Importing, configuring, and starting the NiFi flow definition:

Deployment Successful	2021-03-22 22:59 PDT
Alert Rules Deployed	2021-03-22 22:59 PDT
Deploying Alert Rules	2021-03-22 22:59 PDT
NiFi Flow Started	2021-03-22 22:59 PDT
Starting NiFi Flow	2021-03-22 22:59 PDT
NiFi Flow Imported	2021-03-22 22:59 PDT
Importing NiFi Flow	2021-03-22 22:59 PDT

If your flow deployment fails, check the Event History to see where exactly the issue occurred.

Deployment fails during Phase 1

If the issue occurs during Phase 1 while scheduling resources on Kubernetes and creating the NiFi cluster, you can get more details on why the deployment failed by looking at the application logs.

Identify the application pod by running:

```
kubectl get pods
--namespace dfx-local
```

The result should look similar to

```
NAME                                    READY   STATUS      RESTARTS   AGE
create-db-zxb2q                         0/1     Completed   0          38h
dfx-local-deployment-7f8b466c68-xwrbf   3/3     Running     0          38h
```

Copy the `dfx-local-deployment` pod ID and run to view the application logs.

```
kubectl logs \
  -f dfx-local-deployment-7f8b466c68-xwrbf \
  -c dfx-local \
  --namespace dfx-local
```

Note:

`-f` tails the log file as is writing to it.

A common reason for flow deployment issues is that the Kubernetes cluster does not have enough resources available for the deployment pods to be scheduled. Pods that are stuck in pending are an indicator for not enough resources being available.

You can explore flow deployments and their resources by running:

```
get pods --namespace dfx-deployment_name-ns
```

A healthy deployment should look similar to this with one or more NiFi pods (depending on the Sizing & Scaling settings), a ZooKeeper and a Prometheus pod.

```
NAME                      READY   STATUS    RESTARTS   AGE
dfx-nifi-0                5/5     Running   0          22h
dfx-zookeeper-0           1/1     Running   0          22h
prometheus-prometheus-0   4/4     Running   0          22h
```

If one of the pods is stuck in Pending you can explore the pod further and identify any potential issues by looking at its events.

If in the above screenshot dfx-nifi-0 was Pending and you wanted to find out why, you would run the following command to get detailed information about the containers in the pod.

```
kubectl describe pod dfx-nifi-0 -n dfx-deployment_name-ns
```

Find the Events section and check if there are any messages about why a container could not be scheduled.

If the flow deployment failed because of insufficient resources in the Kubernetes cluster, you can increase the Kubernetes cluster size by using the Edit Configuration action of the affected environment.

Deployment fails during Phase 2

If the issue occurs during Phase 2, check the NiFi canvas of the deployment for any error messages. To get there, open the deployment details, click Manage Deployment and in the following Deployment Manager page select the view in NiFi action.

If a processor or controller service failed to start, make sure that you have provided the correct values for the deployment parameters. You can adjust parameter values in the NiFi canvas and restart processors or controller services as needed. Once you have identified the issue, note down the correct parameter values and start a new deployment.

To view the NiFi log for a particular deployment run the following `kubectl` command.

```
kubectl logs -f dfx-nifi-0 -c app-log --namespace dfx-deployment_name-ns
```

Note:

`-f` tails the log file as NiFi is writing to it.

Troubleshooting using a diagnostic bundle

You can collect diagnostic logs in a bundle and optionally send them to support either using the `start-get-diagnostics-collection` CLI command, or using the Unified Diagnostics feature of the . For more information, see *Collecting diagnostic bundle using CLI* and *Collecting diagnostic bundle using Unified Diagnostics*.

Related Concepts

Overview

Related Tasks

Collecting and sending a diagnostic bundle using Unified Diagnostics

Related reference

Collecting and sending a diagnostic bundle using the CLI

Collecting and sending a diagnostic bundle using the CLI

Learn about collecting diagnostic logs and uploading them using the CLI to facilitate troubleshooting both environment and deployment issues.

```
cdp df start-get-diagnostics-collection \
--df-service-crn '[***DATAFLOW SERVICE CRN***]' \
--description 'some description' \
--destination 'CLOUD_STORAGE' \
--environment-components ‘[“CFM_OPERATOR”, “CERT_MANAGER”]’ \
--start-time [***YYYY-MM-DDTHH:MM***] \
--end-time [***YYYY-MM-DDTHH:MM***] \
--deployments ‘[“[***CRN1***]”, “[***CRN2***]”]’ \
--collection-scope [***ALL|ENVIRONMENT|DEPLOYMENT***] \
--include-nifi-diagnostics [***yes|no***]
```

--df-service-crn

Description

Provide the Cloud Resource Name (CRN) of the service for which you want to collect diagnostic data.

Possible values

Valid CRN

Importance

Required

--description

Description

Provide a free text description of the case for which diagnostic data is being collected. The provided description is persisted in the database and returned in the listing for future reference.

Possible values

Free text

Importance

Required

--destination

Description

Specify the upload destination for the generated bundle.

Possible values

- CLOUD_STORAGE - uploads the bundle to cloud storage associated with the Service environment. A bundle path is included in the listing output with the format <bucket/container>/<datalake-name>/cdf/….
- SUPPORT - uploads bundle to the UDX backend where the bundle is associated with the provided case number and is automatically attached to the case.

Importance

Required

--environment-components

Description

Specify a list of Environment Components for which you want to collect logs. This list operates as an allowed list when present, only collecting logs for specified components. If this list is not present, it implies log collection from all components.

Possible values

Environment component	Namespace
CFM_OPERATOR	cfm-operator-system
CADENCE	cadence
CERT_MANAGER	cert-manager
DFX_LOCAL	dfx-local
DFX_LOGGING	dfx-logging
FLUXCD	flux-system
NFS_PROVISIONER	nfs-provisioner-system
NGINX	nginx-ingress
REDIS	redis
VAULT	vault
VPA	vpa
ZOOKEEPER_OPERATOR	zookeeper-operator-system

Importance

Optional

--start-time

Description

Specify the time from which you want logs to be collected. No logs generated prior to the start time are collected.

If no end or start time is specified, log collection defaults to logs from the past 24 hours. Time zone is assumed to be UTC. Collection throws an error and does not begin if the provided end time is earlier than the provided start time.

Possible values

Date and time in *YYYY-MM-DD*T*hh:mm* format. For example, 2023-02-03T11:00

Importance

Optional

--end-time

Description

Specify the time to which you want logs to be collected. No logs generated after the end time are collected.

If no end or start time is specified, log collection defaults to logs from the past 24 hours. Time zone is assumed to be UTC. Collection throws an error and does not begin if the provided end time is earlier than the provided start time.

Possible values

Date and time in *YYYY-MM-DD*T*hh:mm* format. For example, 2023-02-03T11:01

Importance

Optional

--deployments

Description

Specify a list of Deployment CRNs for which logs should be collected. This list operates as an allowed list when present, only collecting logs for specified deployments. If this list is not present, it allows log collection from all Flow Deployments in the environment.

Possible values

List of valid CRNs

Importance

Optional

--collection-scope

Description

Specify the scope of data collection.

Possible values

ALL	collects logs from environment components and deployments, honoring the provided allowed lists.
ENVIRONMENT	collects logs from environment components, honoring the environment-components allowed list, and no logs from deployments except for those specified in the deployments allowed list.
DEPLOYMENT	collects logs from deployments, honoring the deployments allowed list, and no logs from environment components except for those specified in the environment-components allowed list.

Importance

Optional

--include-nifi-diagnostics

Description

Set this value to 'true' if you want to get a heap and thread dump for a diagnostic bundle. The --de ployments field defines the deployments for which the heap and thread dump are collected.

The following information is collected:

Resource	What is collected
HelmReleases	yaml, events
HelmCharts	yaml
HelmRepositories	yaml
Certificates	yaml
Values Secrets	yaml
PrometheusRules	yaml
Nifis	yaml
StatefulSets	yaml, events
Deployments	yaml, events
Services	yaml
ReplicaSets	yaml, events

Resource	What is collected
DaemonSets	yaml, events

These resources are organized in a namespaced fashion under the AdditionalClusterDetails directory. Note that the logs are stored in a different directory.

Possible values

true or false

Importance

Optional

Default collection:

```
cdp df start-get-diagnostics-collection \
--df-service-crn '[***DATAFLOW SERVICE CRN***]' \
--description 'some description' \
--destination 'CLOUD_STORAGE'
```

Support upload:

```
cdp df start-get-diagnostics-collection \
 --df-service-crn ‘[***DATAFLOW SERVICE CRN***]’ \
 --description ‘some description’ \
 --destination ‘SUPPORT’ \
 --case-number ‘123456’
```

Note that case-number is required when destination is SUPPORT

List collection attempts:

```
cdp df list-diagnostics \
--df-service-crn ‘[***DATAFLOW SERVICE CRN***]'
```

Related Tasks

Collecting and sending a diagnostic bundle using Unified Diagnostics

Related Information

Send a diagnostic bundle using the

Collecting and sending a diagnostic bundle using Unified Diagnostics

You can trigger diagnostic bundle collection for your environment in order to send Data Lake, FreeIPA, and service and environment logs to Support for support case resolution.

Before you begin

Required role: You need the EnvironmentAdmin or Owner role for the environment for which you would like to create a bundle.

- FreeIPA deployment time must be September 2020 or later.
- The VMs from which diagnostic data is to be collected must be running.
- Salt minions must be started properly on the nodes.

Procedure

1. Log in to the web interface.

2. Navigate to the Management Console.
3. Click Help in the bottom left corner and select Collect Diagnostic Data.

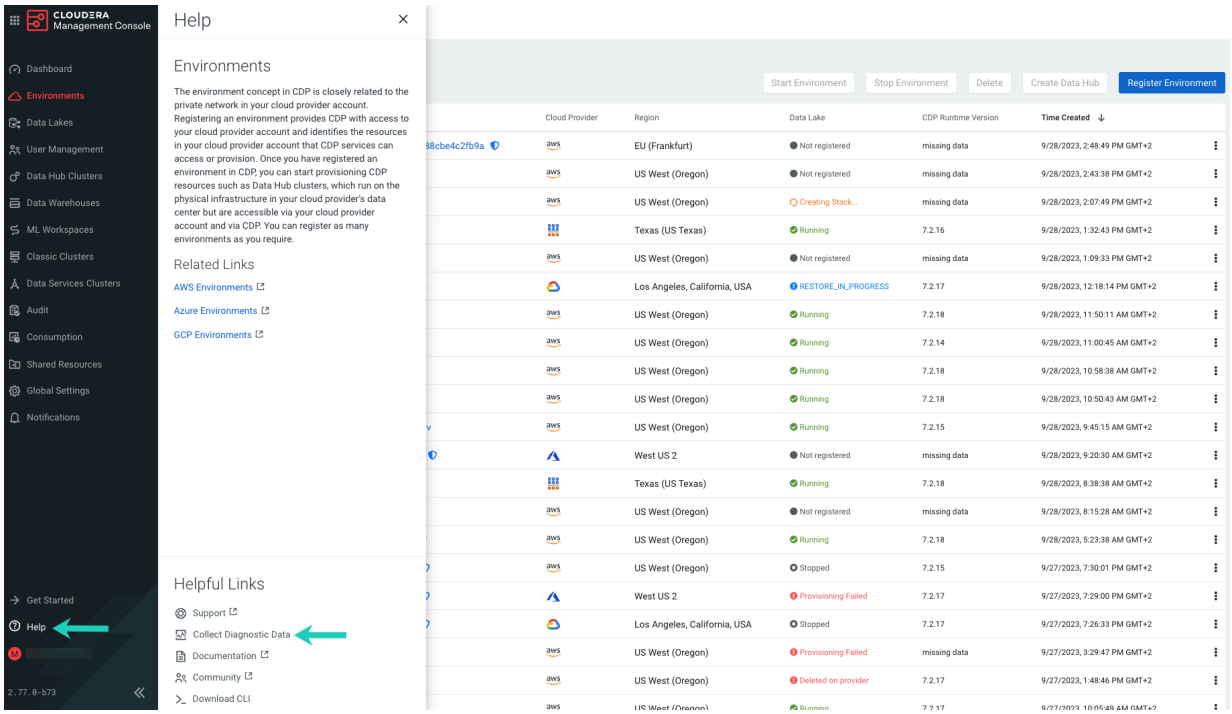

4. Click the Collect New Diagnostic Data.
5. On the **General Information** page, provide the following:

   Envionment

   Select the environment for which you would like to generate a diagnostic bundle. This is a required field.

   Case Number

   Enter the related support case number. You can obtain the case number from the MyCloudera portal or from your email by checking support case notification emails. This is a required field.

   Description

   Provide a free text description of the case for which you would like to generate diagnostic data. The provided description is persisted in the database and returned in the listing for future reference. This is a required field.

   Time Range

   You can specify specific points in time (date and time) from which to start and end the collection. This is an optional field.
6. Click Next.

7. On the **Selected Services** page under DataFlow, you can configure the diagnostic bundles you want to generate for environment components and deployments:

   Environment Components

   Select a list of Environment Components from which you want to collect logs. This list operates as an allowed list when present, only collecting logs for specified components.

   If you leave this list empty, it implies you want to collect logs from all components.

   Select Components

   Specify a list of Deployment CRNs for which you want to collect logs. This list operates as an allowed list when present, only collecting logs for specified deployments.

   If you leave this list empty, it implies log collection from all Flow Deployments in the environment.

   Collection Scope

   Specify whether you want to collect environment logs, deployment logs, or both.

   Regardless of the selection you make here, logs for environment components and deployments you specified in their respective allowed lists will be collected.

   To generate diagnostic data for a specific Deployment, enter the CRN of the Deployment for which you want to collect logs in the Select Components field and select DEPLOYMENT as Collection Scope.

   Note: To obtain the CRN of a Deployment:

   a. Navigate to DataFlow on the web interface.
   b. On the **Dashboard** select the deployment you want to collect diagnostic data for.
   c. Click Copy under CRN #.

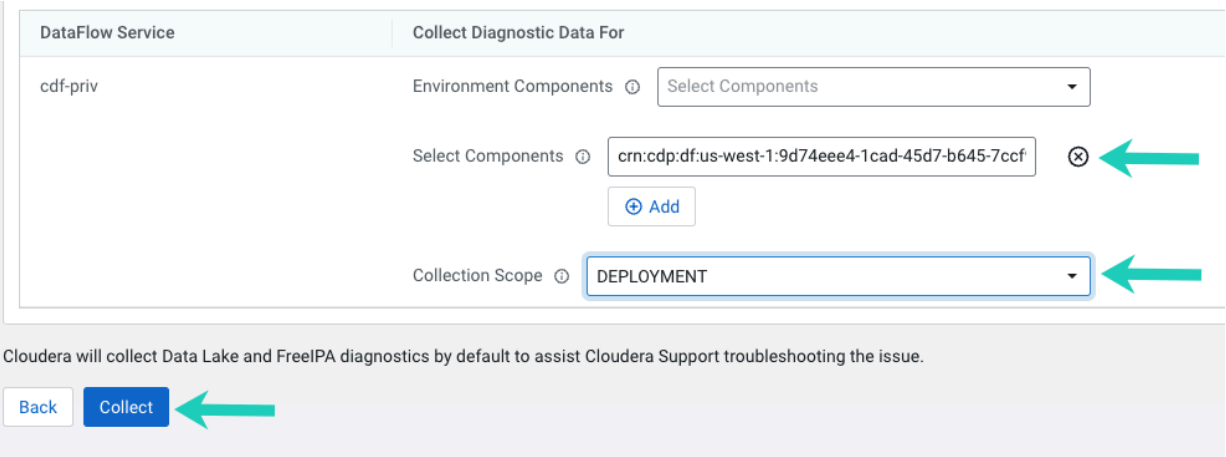

   To obtain all diagnostic data for a Service, set Collection Scope to ALL.

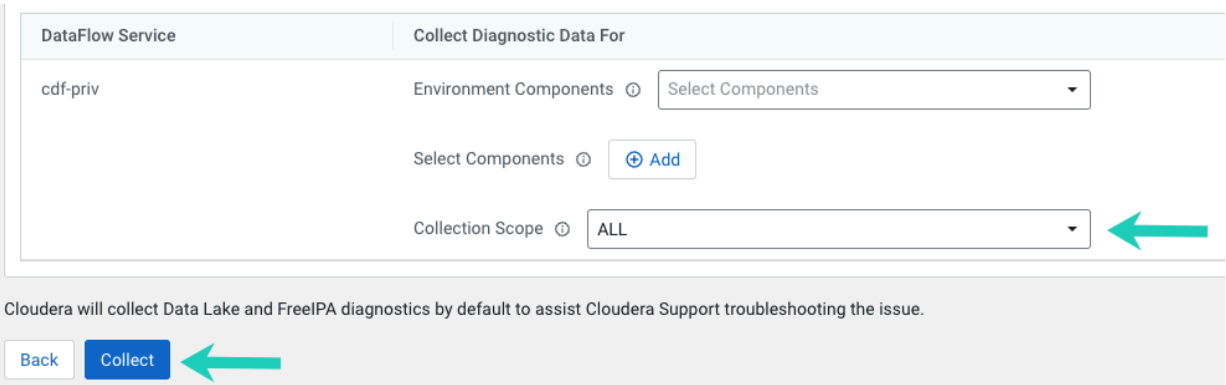

8. Click Collect.
   New entries appear on the page, allowing you to track the status of the log collection. Since a separate bundle is created for each service, there is one entry for each of the following: Data Lake, FreeIPA, and one entry for each service. While the collection is in progress, the collection status is In progress:

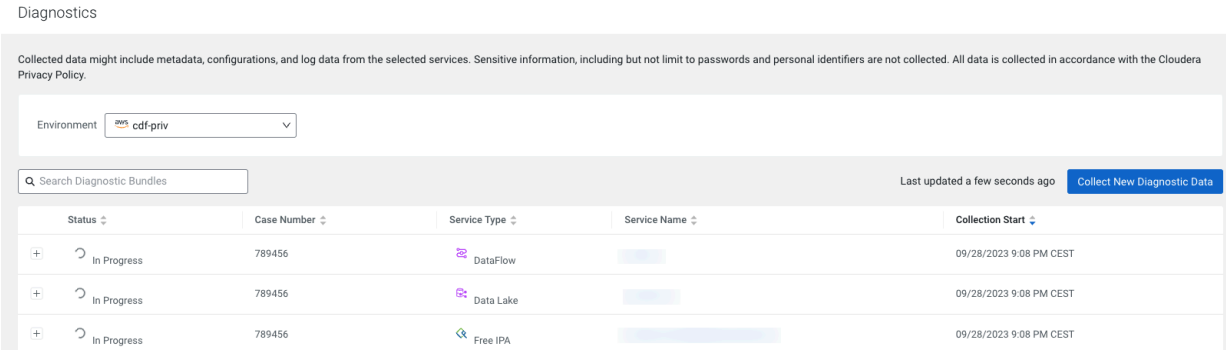

Results

Once the collection is complete, the status changes to Completed. This means that Support can now access the diagnostic bundles:

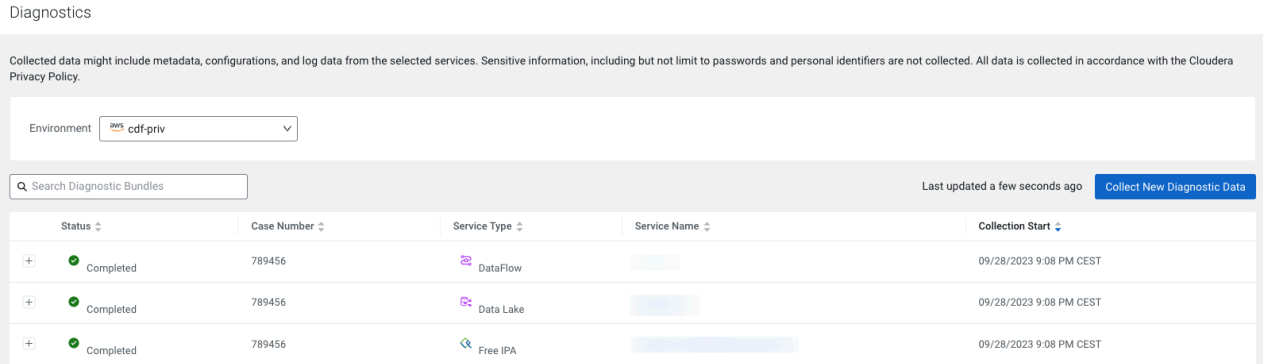

Related reference

Collecting and sending a diagnostic bundle using the CLI

Downloading NiFi application log

You can download the NiFi application log from the CDF Deployment Manager to use it for troubleshooting.

About this task

This feature allows you to download the NiFi application log that is currently being written. As the log file is rotated and the old file is archived once the file size reaches 10 MB, this is the theoretical maximum you can download using this method. For information on downloading archived log files, see *Diagnostic bundle collection*.

Before you begin

You need DFFlowAdmin permission to perform this action.

Procedure

1. Open by clicking the DataFlow tile in the sidebar.
2. Click Deployments from the left navigation pane.
3. Select a flow deployment to expand the Deployment Details pane and get deployment, KPI, system metrics, alerts, and event history information.

4. In Deployment Details, click Actions .
   You are redirected to the **Deployment Manager** page.
5. Click Actions in the **Deployment Manager** page.

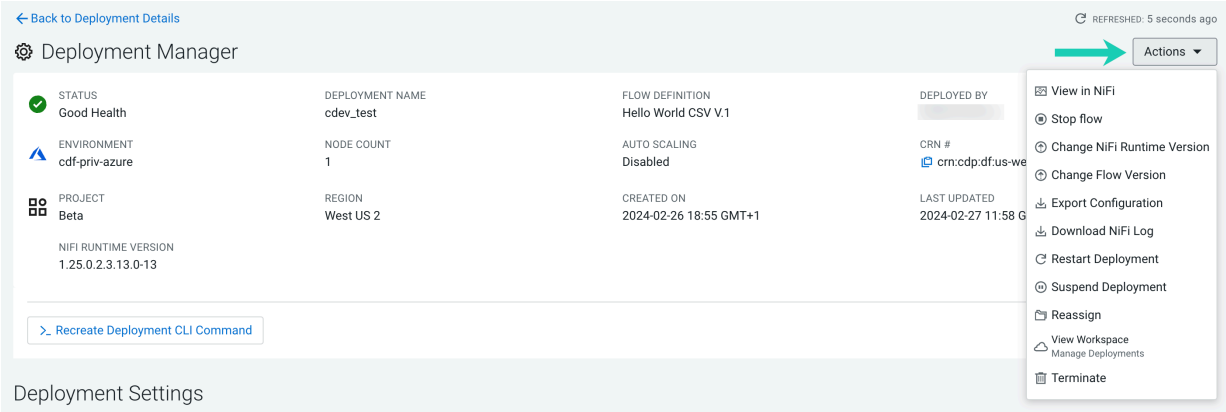

6. Select Download NiFi Log.

Results

The current NiFi application log is downloaded to your computer in tar.gz format.

Cloudera DataFlow

Troubleshooting

Date published: 2021-04-06
Date modified: 2024-12-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Downloading NARs failed.....4

Fetching flow from Cloudera DataFlow failed with FileNotFoundException.....4

Downloading NARs failed

Learn how to recognize and fix common errors that might occur when you are downloading NAR (NiFi Archive) files.

By default, the NARs required for the execution of the function are downloaded from the Cloudera repository. Depending on where you designed your data flow, specific versions of the NAR files are expected, which may not be available in the Cloudera repository.

It is important to make sure that the referenced components have a version that is available in the repositories. If the component version is not available, you can:

- override the NEXUS_URL property to use another repository
- make the NARs available to your function by following the instructions provided in the *Custom NARs* topic of the relevant cloud provider section in the Cloudera DataFlow Functions documentation
- manually edit your flow definition JSON file to update the version of the components to use a version provided by Cloudera before you upload the file in the Cloudera DataFlow Catalog.

Related Information

Custom NARs in AWS

Custom NARs in Google Cloud

Custom NARs in Azure

Fetching flow from Cloudera DataFlow failed with FileNotFoundException

Learn how to recognize and fix common errors that might occur when you are fetching a flow from Cloudera DataFlow.

This error can happen when the flow definition CRN that you provided to check out the flow from the Cloudera DataFlow Catalog does not contain the version information. The CRN of the flow definition should always end with /v.<*VERSION OF THE FLOW*>.

```
java.io.IOException: Failed to fetch flow from Dataflow Service with provide
d information
Caused by java.io.FileNotFoundException: https://api.us-west-1.cdp.clouder
a.com/dfx/api/v1/catalog/flows/versio
ns/crn%3Acdp%3Adf%3Aus-west-1%3A558bc1d2-8867-4357-8524-311d51259233%3
Aflow%3Ajvp-sfet-naaf
```

Cloudera DataFlow

DataFlow Functions Telemetry Tutorial

Date published: 2021-04-06
Date modified: 2024-06-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Scope and goals 4

Terminology 4

Assets 4

Telemetry tutorial steps 5

Review the prerequisites 5
Register for Cloudera Public Cloud 60-day trial 5
Develop and test a flow in Apache NiFi 9
1. Install Apache NiFi 9
2. Import the Telemetry Tutorial NiFi Flow 10
3. Configure and start the Telemetry Tutorial NiFi Flow 14
4. Test the Telemetry Tutorial NiFi Flow 15
5. Download the Telemetry Tutorial NiFi Flow 18
Add the Telemetry Tutorial NiFi Flow as a function in Cloudera DataFlow 20
1. Upload the Telemetry Tutorial NiFi Flow to the Cloudera DataFlow Catalog 20
2. Download the Lambda Cloudera DataFlow Function binaries zip and upload to S3 21
3. Create a Cloudera Public Cloud Service Account 22
Run the Cloudera DataFlow function in serverless mode in AWS Lambda 24
1. Create the Cloudera DataFlow function 24
2. Test the Cloudera DataFlow function 28
3. Create S3 trigger for the Cloudera DataFlow function 31
4. Monitor the Cloudera DataFlow function 33

Scope and goals

This tutorial walks you through the process of going from zero – no Cloudera Public Cloud tenant and no data flow – to a serverless Apache NiFi flow on AWS Lambda in under 30 minutes without any Cloudera Sales or technical assistance.

You can deploy a serverless NiFi flow using Cloudera DataFlow Functions without the need to create a Data Lake and to perfom the corresponding cloud prerequisite steps. Following this end-to-end workflow, you will:

1. Register for a 60-day trial for Cloudera Public Cloud Trial which will provision a tenant in Cloudera Public Cloud.
2. Implement a data distribution use case on a local development workstation using Apache NiFi.
3. Register the NiFi flow in Cloudera DataFlow Service as a Cloudera DataFlow function.
4. Deploy the Cloudera DataFlow function in serverless mode using AWS Lambda.

Terminology

This tutorial uses the following terms and concepts that you should be familiar with.

Term	Definition
Apache NiFi	Low-code data ingestion tool built to automate the flow of data between systems
Flow	Represents data flow logic that was developed using Apache NiFi
Processor	Component in the data flow that perform work combining data routing, transformation, and mediation between systems
Cloudera Public Cloud	Cloudera's data management platform in the cloud
Cloudera DataFlow Service	Cloudera DataFlow data service that enables self-serve deployments of Apache NiFi
Cloudera DataFlow function	Flow that is uploaded into the Cloudera DataFlow Catalog that can be run in serverless mode by serverless cloud provider services
Cloudera DataFlow Catalog	Inventory of flow definitions from which you can initiate new deployments
AWS Lambda	Serverless, event-driven compute service that lets you run a Cloudera DataFlow function without provisioning or managing servers

Assets

This tutorial relies on the following assets:

Asset	Description
Telemetry Tutorial NiFi Flow	The NiFi flow that implements the requirements of the use case described in this guide and that is run in serverless mode on AWS Lambda
Cloudera DataFlow Function Definition	The function definition be used with the AWS CLI command to create the function on AWS Lambda
trust-policy.json	Trust policy used with the AWS CLI command to create the IAM role required to build the function in AWS Lambda
Sample Trigger Event	Sample trigger event used to test the function in the AWS Lambda
Sample Telemetry File	Raw sample telemetry file to be uploaded into S3 to test the NiFi Flow and Lambda function
Second Sample Telemetry File	Another raw sample telemetry file to be uploaded into S3 to test the function in AWS Lambda with the S3 trigger enabled

Telemetry tutorial steps

Review the prerequisites

This section helps you to examine the list of actions you must perfom before you start working on your function.

- AWS user account is required with access policies that has permissions to list and create buckets, roles, and Lambda functions.
- Access to AWS console is required.
- Access key is required for the AWS user account to use AWS CLI.
- AWS CLI client needs to be installed and configured to use the access key.
- AWS bucket is needed for the telemetry events files:

  1. Create two folders in the bucket: truck-telemetry-raw and truck-telemetry-processed.
  2. Download the sample telemetry file and upload it to the truck-telemetry-raw folder.This file will be used during the tutorial to run tests.

Register for Cloudera Public Cloud 60-day trial

This section provides information on the free Cloudera Public Cloud trial.

Procedure

1. Log into your Cloudera account.

   If you do not have an account on cloudera.com, register for a new account here.

2. Once you have a cloudera.com account created and you logged into the account, register for the 60-day Cloudera Public Cloud trial.
   - If your company has been registered for a trial account, by the end of the registration you will have a newly provisioned tenant on Cloudera Public Cloud.

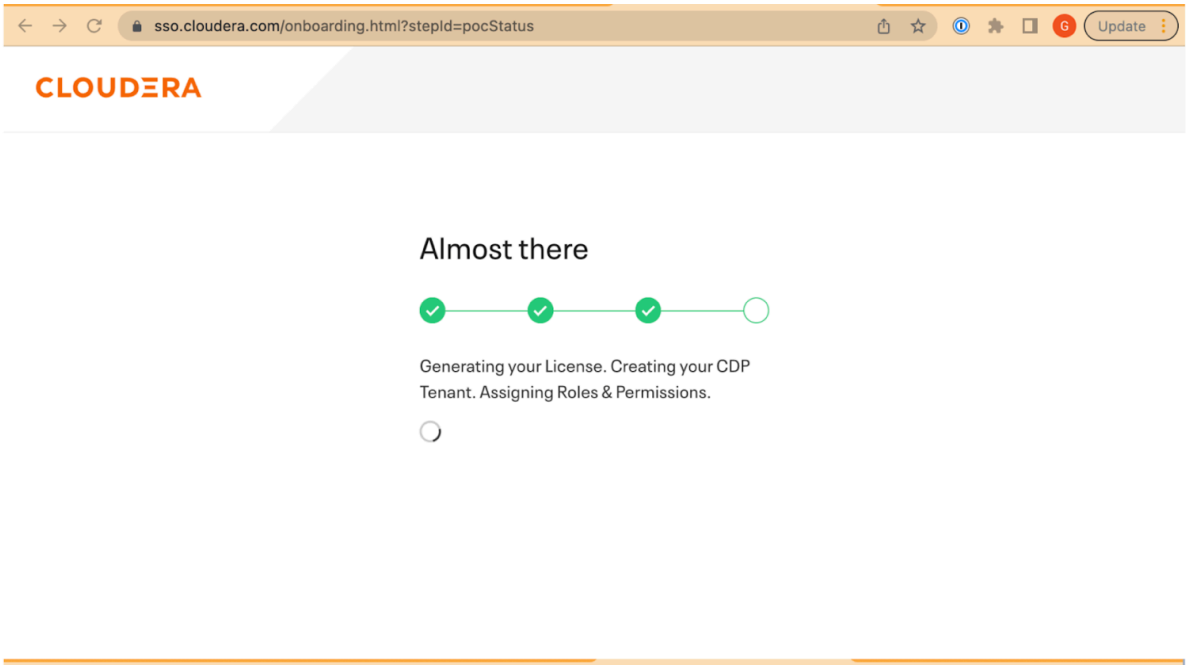

   - If your company has not been registered for a trial account, shortly after completing the registration, the Cloudera Sales Team will reach out to you to complete your trial registration.

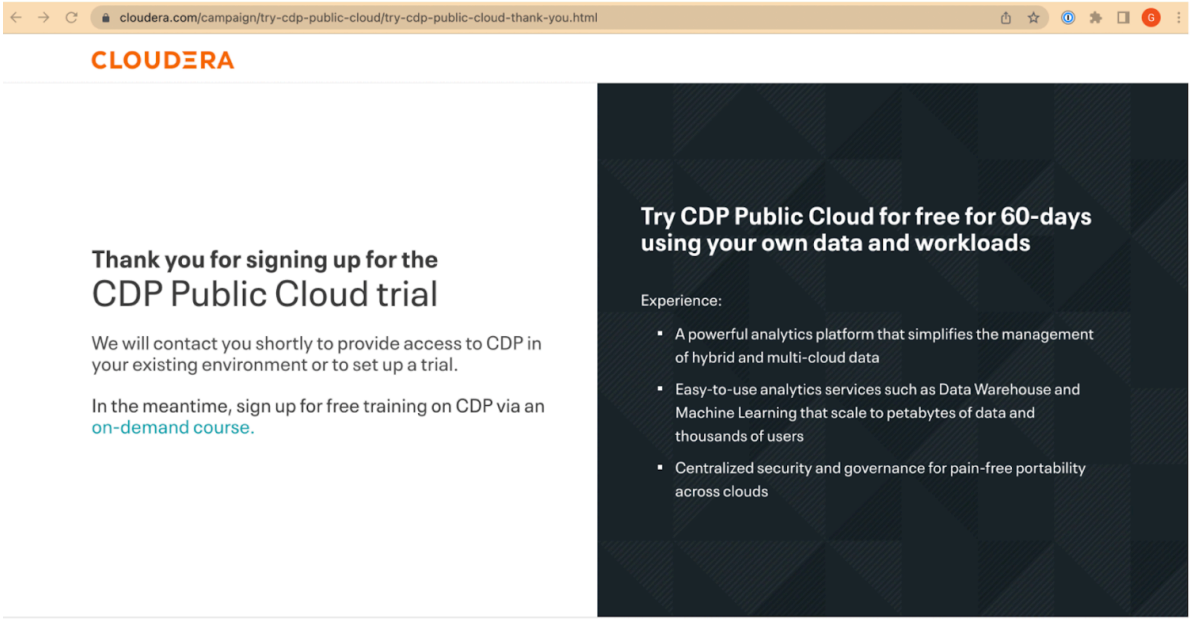

3. After completing the registration, you will receive an email with instructions on how to log into your newly provisioned tenant on Cloudera Public Cloud. Disregard steps 2 and 3 as Cloudera DataFlow Function does not require any of the prerequisites to create an environment or a DataLake.

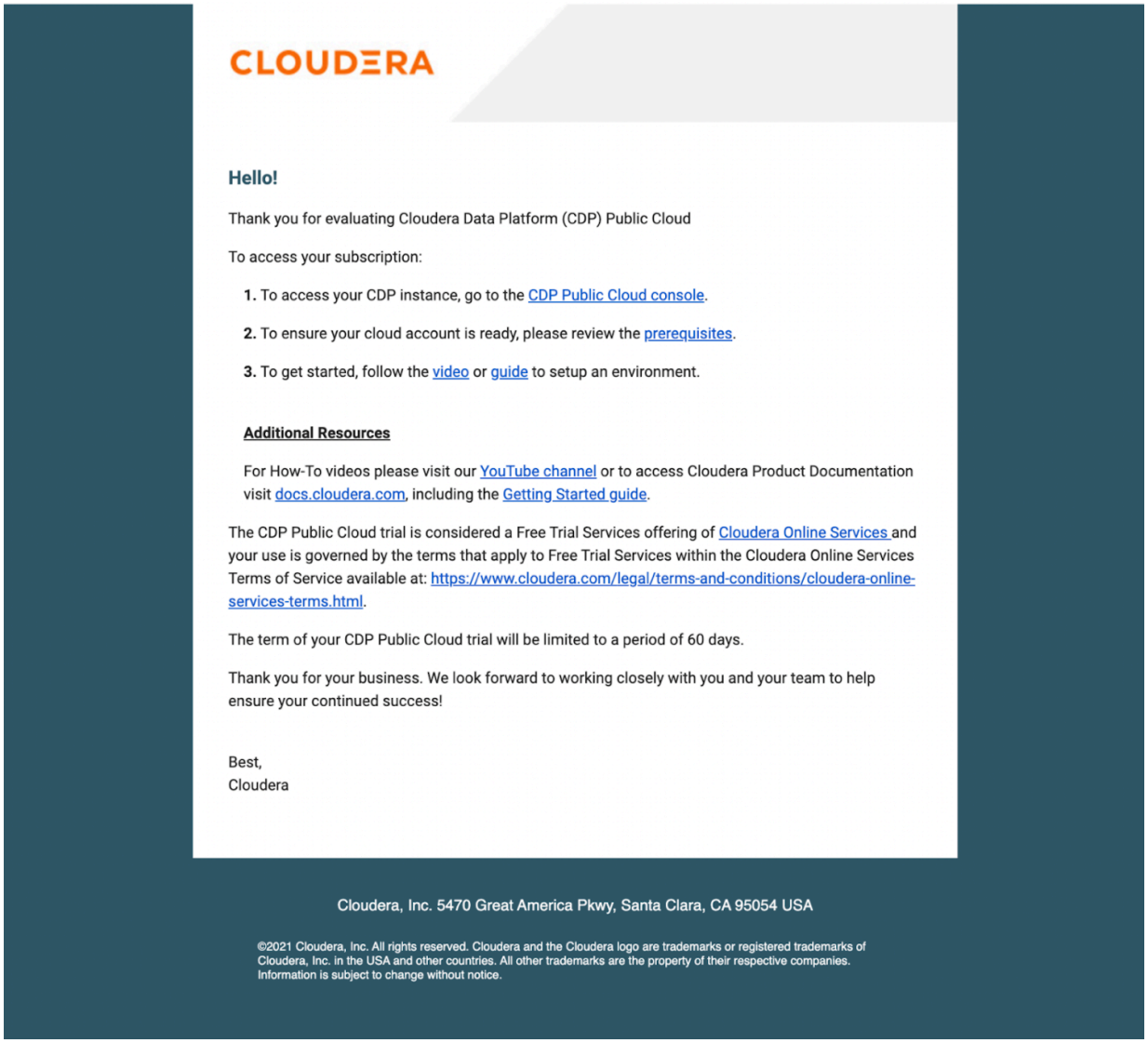

CLOUDERA

Hello!

Thank you for evaluating Cloudera Data Platform (CDP) Public Cloud

To access your subscription:

1. To access your CDP instance, go to the CDP Public Cloud console.
2. To ensure your cloud account is ready, please review the prerequisites.
3. To get started, follow the video or guide to setup an environment.

Additional Resources

For How-To videos please visit our YouTube channel or to access Cloudera Product Documentation visit docs.cloudera.com, including the Getting Started guide.

The CDP Public Cloud trial is considered a Free Trial Services offering of Cloudera Online Services and your use is governed by the terms that apply to Free Trial Services within the Cloudera Online Services Terms of Service available at: https://www.cloudera.com/legal/terms-and-conditions/cloudera-online-services-terms.html.

The term of your CDP Public Cloud trial will be limited to a period of 60 days.

Thank you for your business. We look forward to working closely with you and your team to help ensure your continued success!

Best,
Cloudera

Cloudera, Inc. 5470 Great America Pkwy, Santa Clara, CA 95054 USA

©2021 Cloudera, Inc. All rights reserved. Cloudera and the Cloudera logo are trademarks or registered trademarks of Cloudera, Inc. in the USA and other countries. All other trademarks are the property of their respective companies. Information is subject to change without notice.

4. After logging into the Cloudera Public Cloud console with the credentials that you used to register for the trial, select the Cloudera DataFlow service from the homepage.

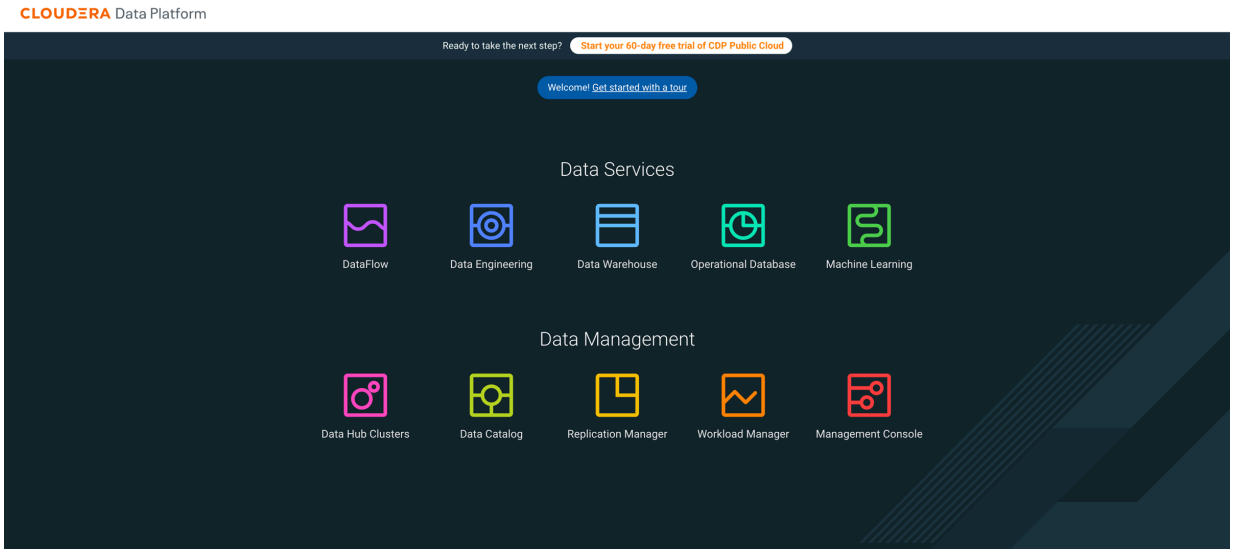

5. If you click Functions in the left navigation pane, you can see the options for cloud provider functions.

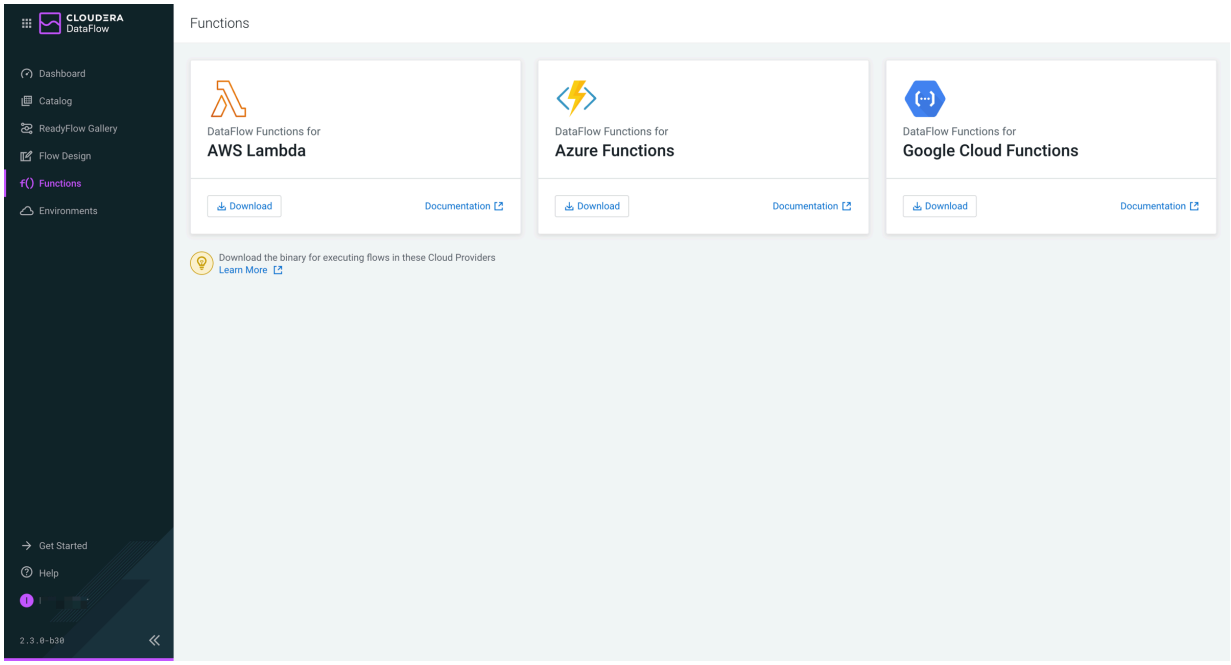

This is from where you can Download on the AWS Lambda tile to download the Cloudera DataFlow function binaries for AWS Lambda.

6. If you click Catalog in the left navigation pane, you can see that it is empty. You can develop an Apache NiFi flow locally, test it, and then come back to the Catalog to upload it as a Cloudera DataFlow function.

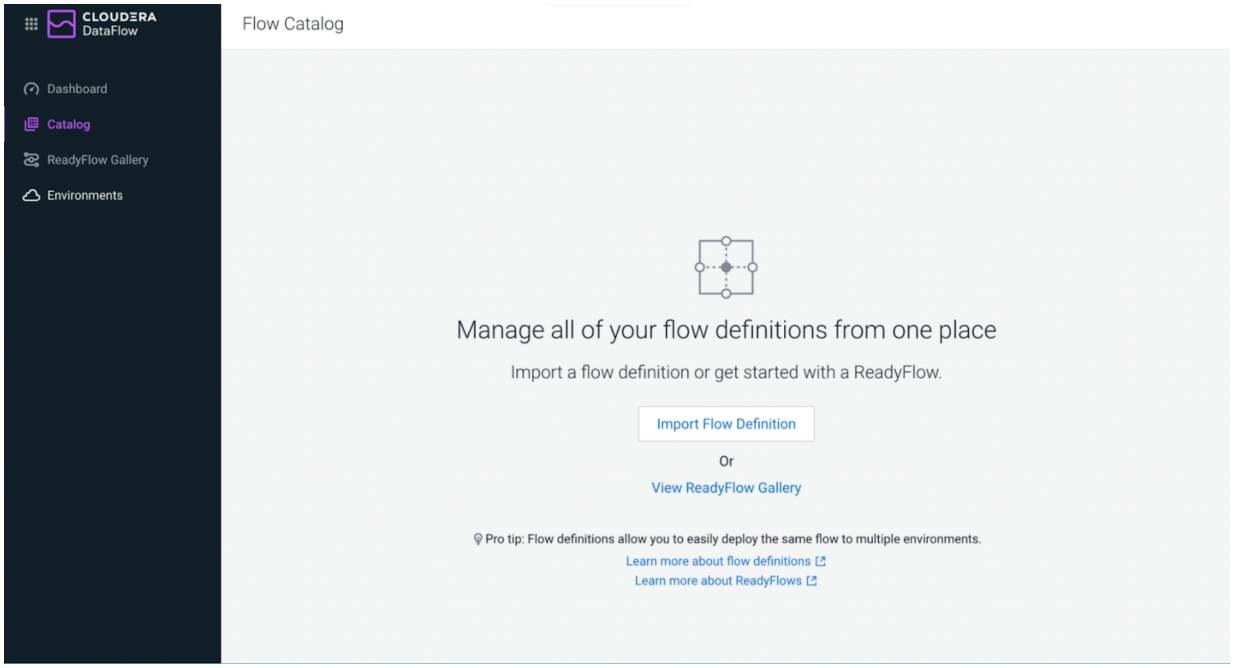

Develop and test a flow in Apache NiFi

This section walks you through designing your function by developing a data flow using NiFi on your local development workstation. Once the NiFi flow has been developed and tested, you will deploy it as a Cloudera DataFlow function in serverless mode using AWS Lambda.

1. Install Apache NiFi

Download and install Apache NiFi 1.16.X on your local development workstation.

Procedure

1. Follow the instructions to download and install the latest version of Apache NiFi.

2. Log into NiFi with the generated credentials.

   You can see the NiFi flow designer canvas.

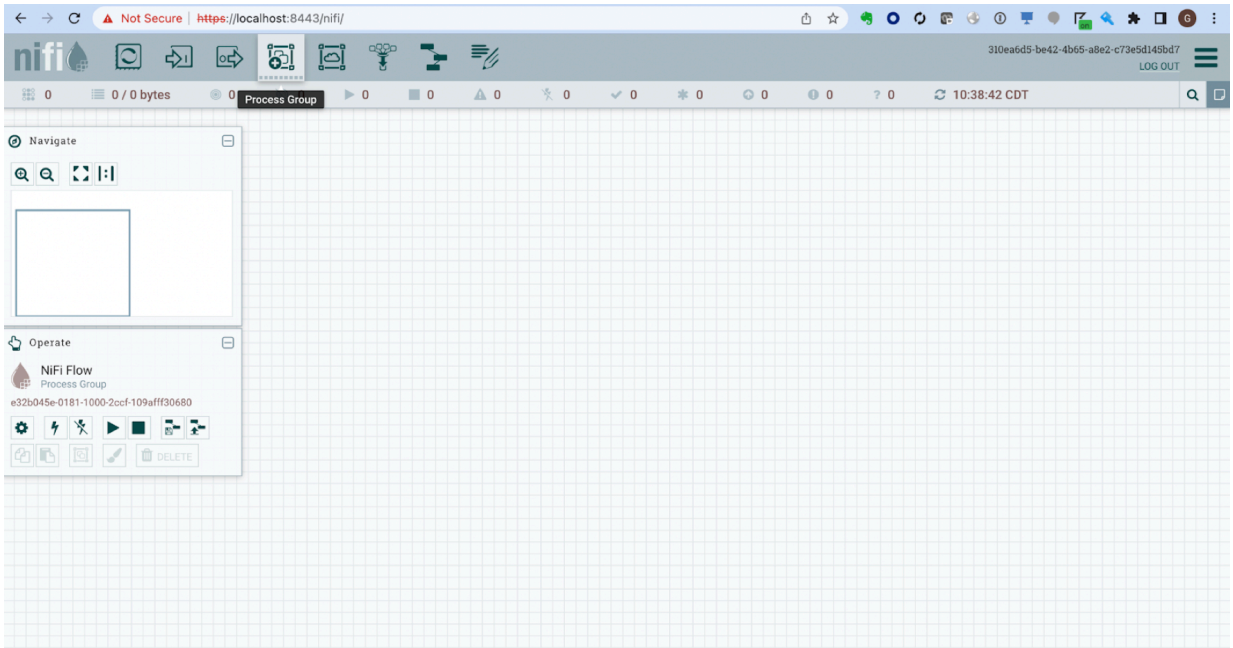

2. Import the Telemetry Tutorial NiFi Flow

Download and add the example data flow to Apache NiFi. This example flow implements the requirements of the use case described in this quickstart.

Procedure

1. Download the Telemetry Tutorial NiFi Flow to your local machine.
2. Drag a Process Group (fourth icon from the left) onto the canvas.

3. Click the browse link, select the Telemetry Tutorial NiFi Flow you have downloaded, and click Add.

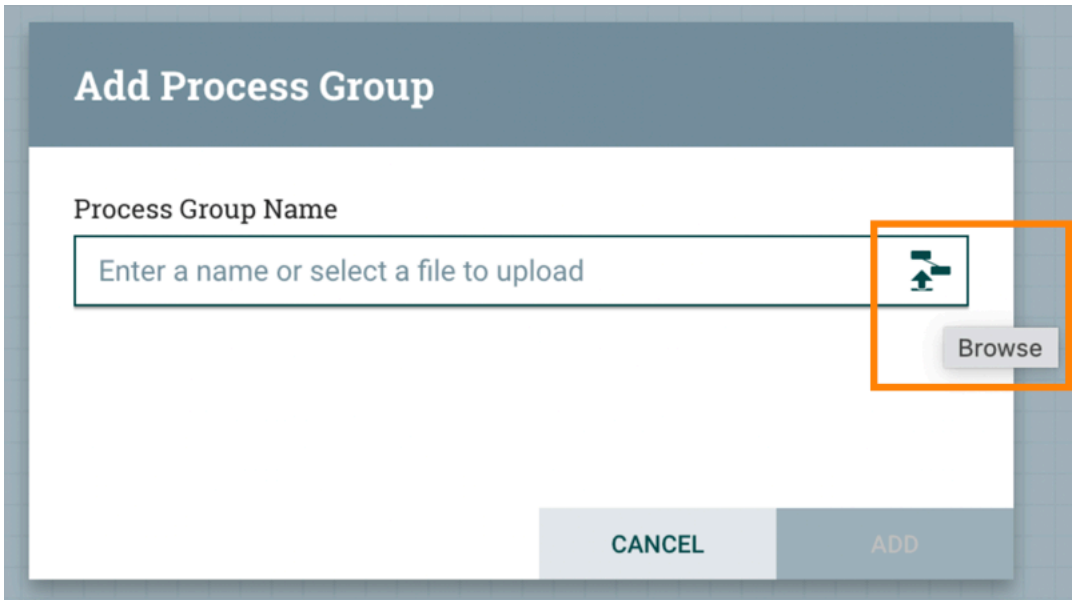

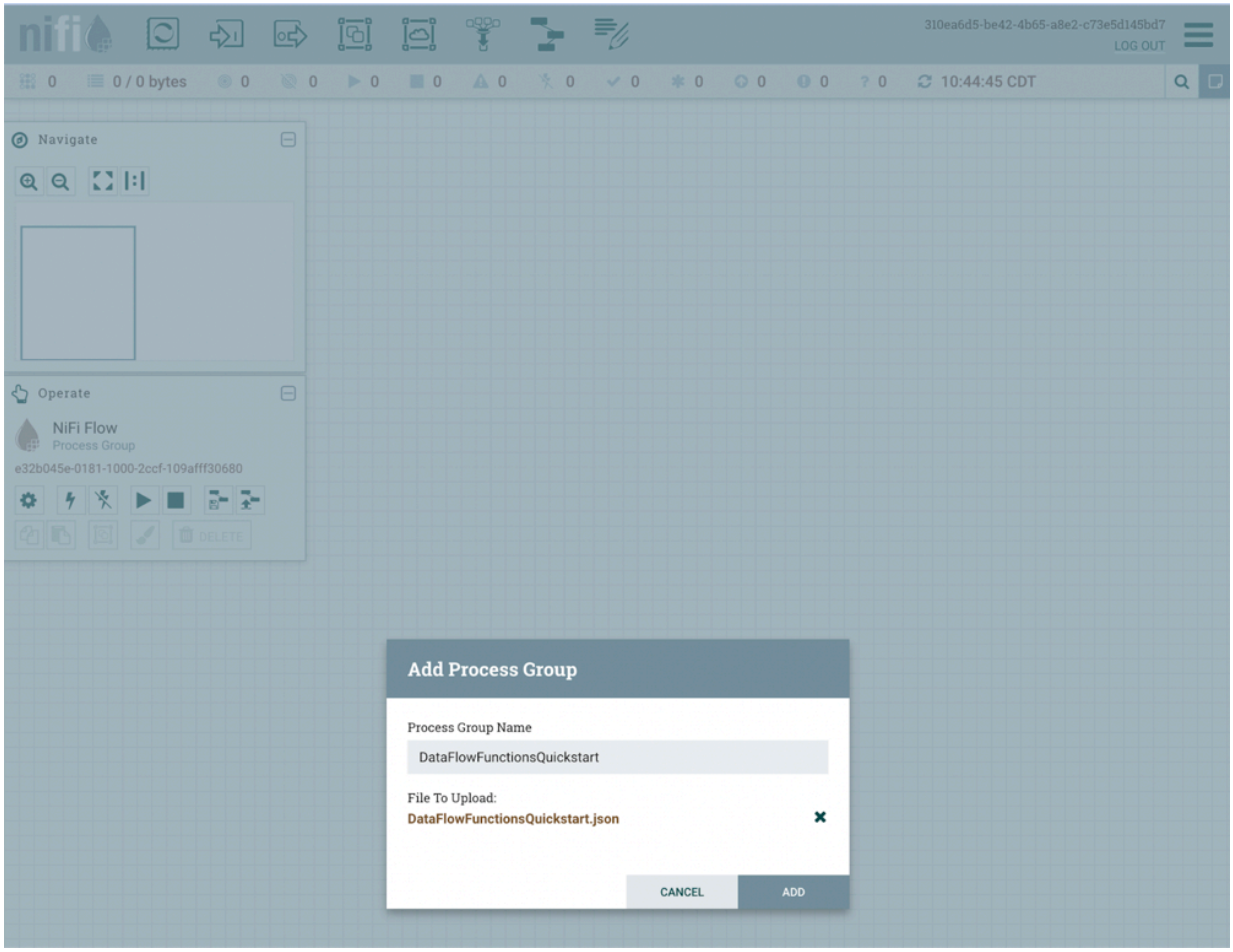

4. Explore the flow that you have uploaded to the canvas.

a) Double-click the DataFlowFunctionsQuickstart process group.

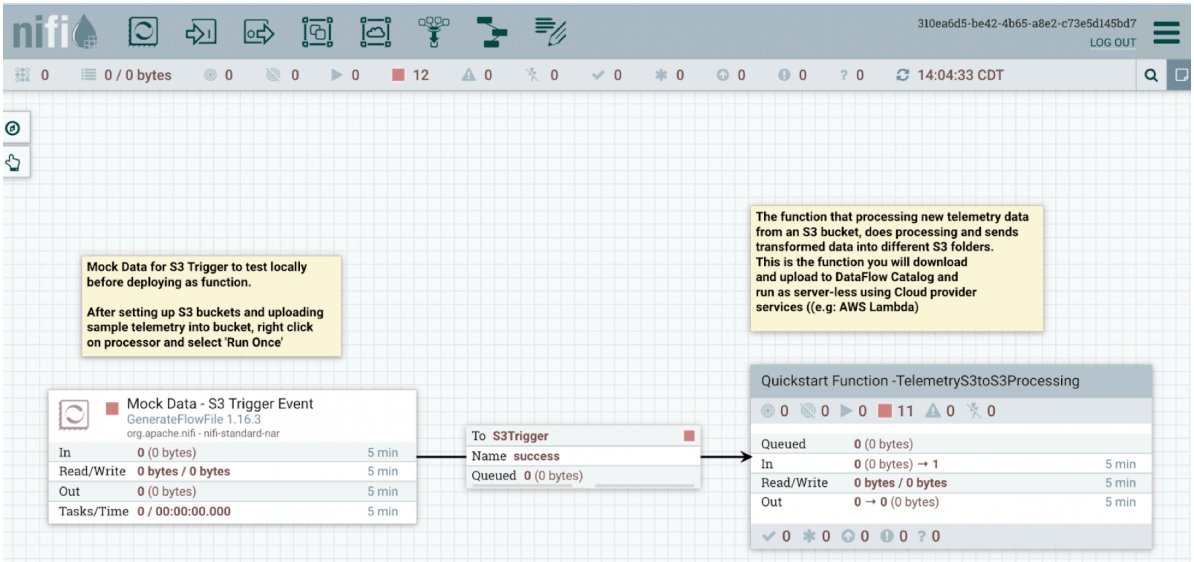

The Mock Data - S3 Trigger Event processor will generate a test Lambda S3 trigger event. You can use this processor to test the function locally before deploying it on AWS Lambda.

b) Double-click the Quickstart Function -TelemetryS3toS3Processing process group that contains the flow for processing the telemetry data.

Note the details of this flow:

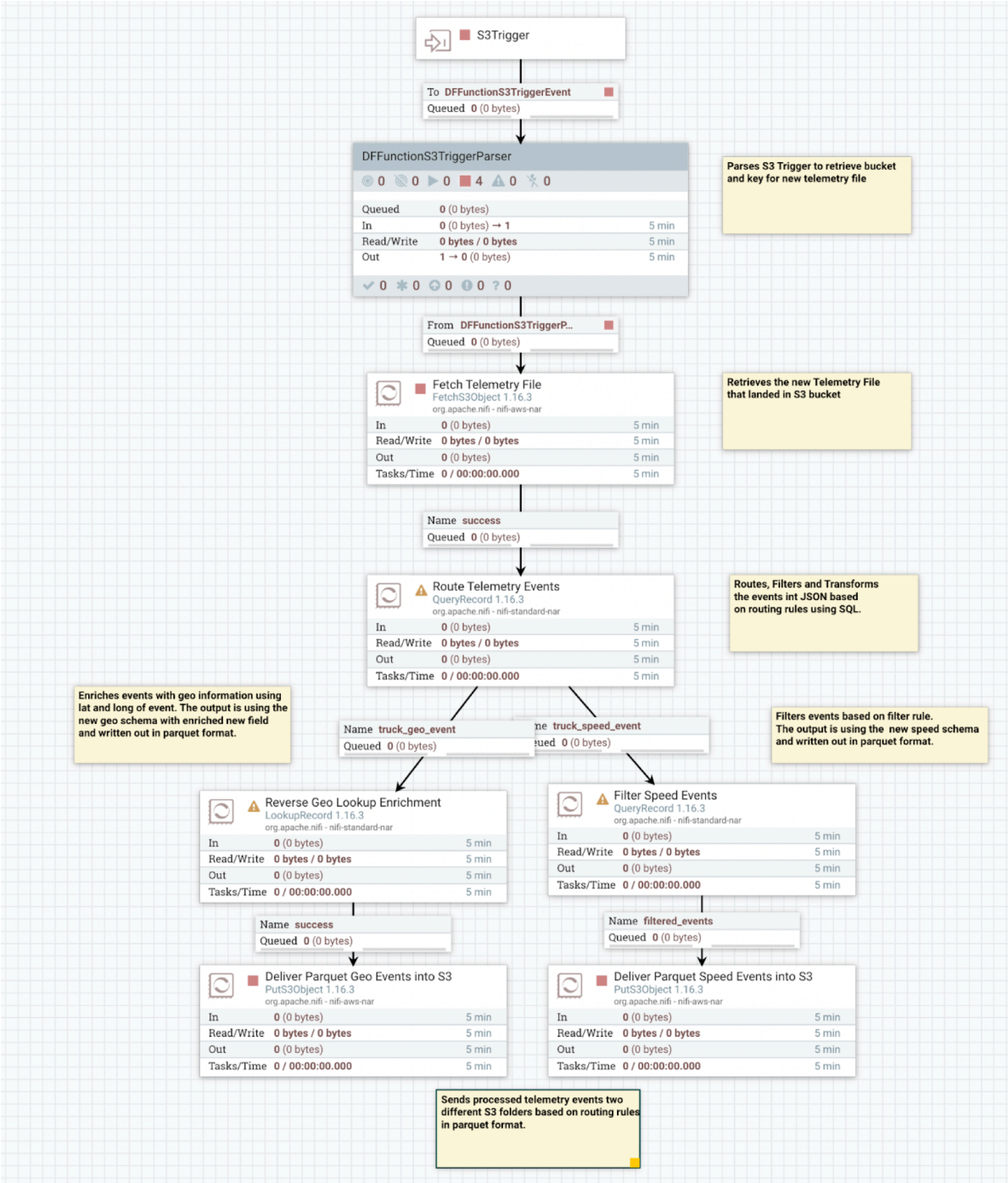

- S3Trigger is an input port that the AWS Lambda handler sends the S3 trigger event to. All functions need to start with the input port for the corresponding trigger that will be used.
- DFFunctionS3TriggerParser parses the json trigger event to extract the bucket, key and region and store in flow attributes so it can be easily used by the downstream processors.
- Fetch Telemetry File fetches the new telemetry file that landed in S3 and stores it as a flowfile.
- Route Telemetry Events routes sensor events in the file to different paths based on the eventSource value in the event. Events with eventSource of ‘truck_geo_event” will be routed to one path and events with value ‘truck_speed_event’ be routed to another path.
- For the events sent to the truck_geo_event path, the event will be enriched using custom groovy code that looks up the geo address based on the lat and long values in the event. The enriched events are then converted to parquet format using the supplied schema and the file is stored in a new s3 folder.

- For the events sent to the truck_speed_event path, the events are filtered for any events with speed > 40 and then converted into parquet format and stored in a different s3 folder.

3. Configure and start the Telemetry Tutorial NiFi Flow

Apache NiFi uses the concept of parameter context to store properties of the flow that need to change when deploying to different environments. The parameter context for the function in this tutorial is called DataFlowFunctionsQuickstart.

Procedure

1. Configure the parameter context.
   a) Right-click the canvas and select Parameters.
   b) Update the following four parameter values based on your environment.

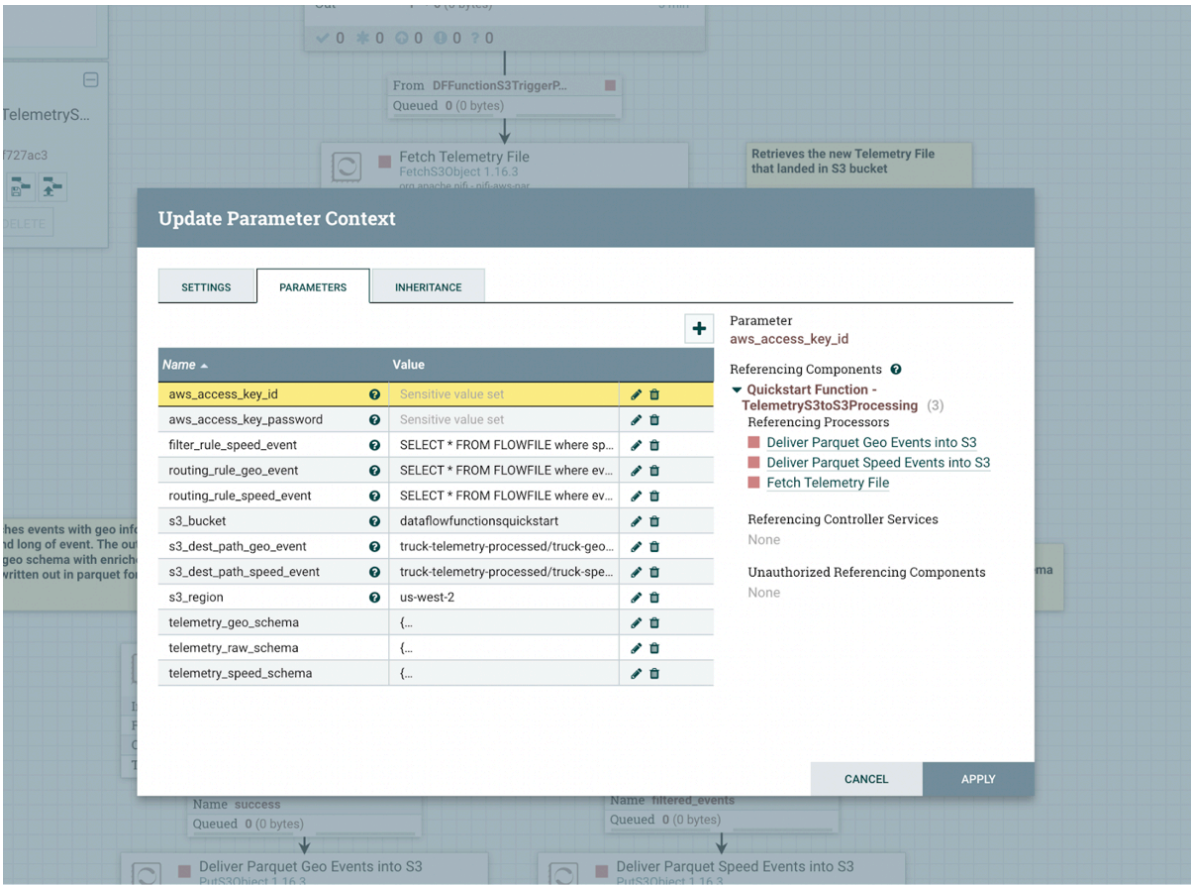

- aws_access_key_id - AWS access key id to fetch and write objects to S3 bucket
- aws_access_key_password - AWS secret access key secret to fetch and write objects to S3
- s3_bucket - S3 bucket you created where telemetry data will be processed
- s3_region - the region where you created the bucket

2. Start the NiFi flow.
   a) Start Controller Services.

      Controller Services are shared services that can be used by processors and other services to utilize for configuration or task execution
   b) Within the process group called Quickstart Function -TelemetryS3toS3Processing, select the canvas to bring mouse focus to it, right-click the canvas, select Configure Controller Services .

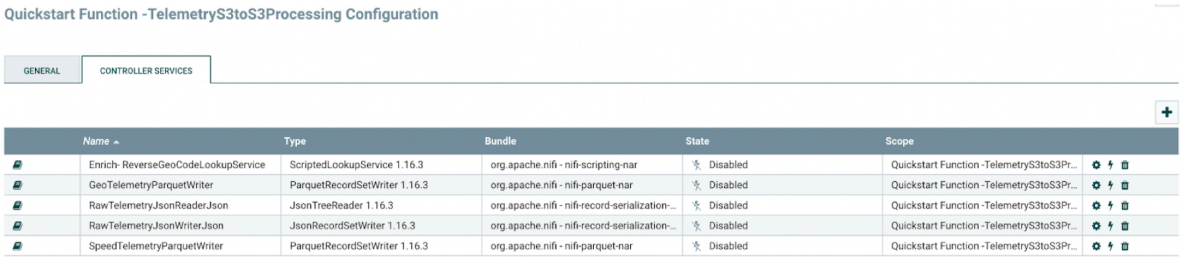

      There are five controller services defined for this function that are responsible for parsing the incoming JSON telemetry data, writing the data in Parquet format and doing the geo address lookup.
   c) Click the bolt icon next to each service to enable it.
   d) Close the configuration dialog.
   e) Right-click the canvas and select Start to start the process group called Quickstart Function -TelemetryS3toS3Processing.

      You should see that each processor have a green play button which indicates that all the processors are started and ready to receive data.

4. Test the Telemetry Tutorial NiFi Flow

With the flow configured and started, you can test it locally with a sample trigger event before deploying it as a serverless function on AWS Lambda.

Procedure

1. Configure the test S3 trigger event.
   a) From within the process group Quickstart Function -TelemetryS3toS3Processing, right-click the canvas and select Leave Group to go to the parent processor group.
   b) Right-click the Mock Data - S3 Trigger Event processor group and select Configure.
   c) Click the Properties tab and edit the Custom Text property value.

      This property value represents a mock trigger event that Lambda would create when a new file called sampleTelemetryRawData.txt is added to the bucket folder.

      Note: This is the file is that you uploaded to <<your_bucket>>/truck-telemetry-raw in the prerequisite step.
   d) Update the bucket.name JSON field with the name of the bucket you created earlier.
   e) Update the awsRegion JSON field to match the region in which you earlier created the bucket.
   f) Click OK and then Apply to save the change.

2. Run the NiFi flow with the test trigger event.

a) Right-click the ‘Mock Data - S3 Trigger Event’ processor group and select ‘Run Once’.

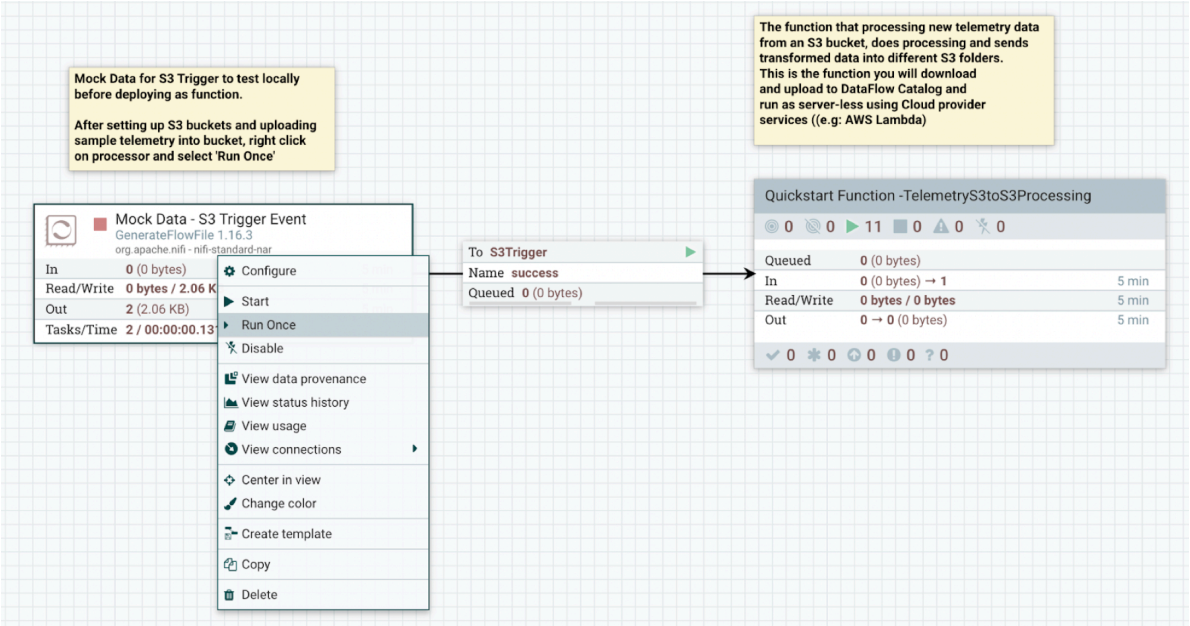

This will create a mock trigger event flow file.

b) If you double-click the process group Quickstart Function -TelemetryS3toS3Processing, you can see the flow file processed by the processors.

The In metric should show '1' across all processors.

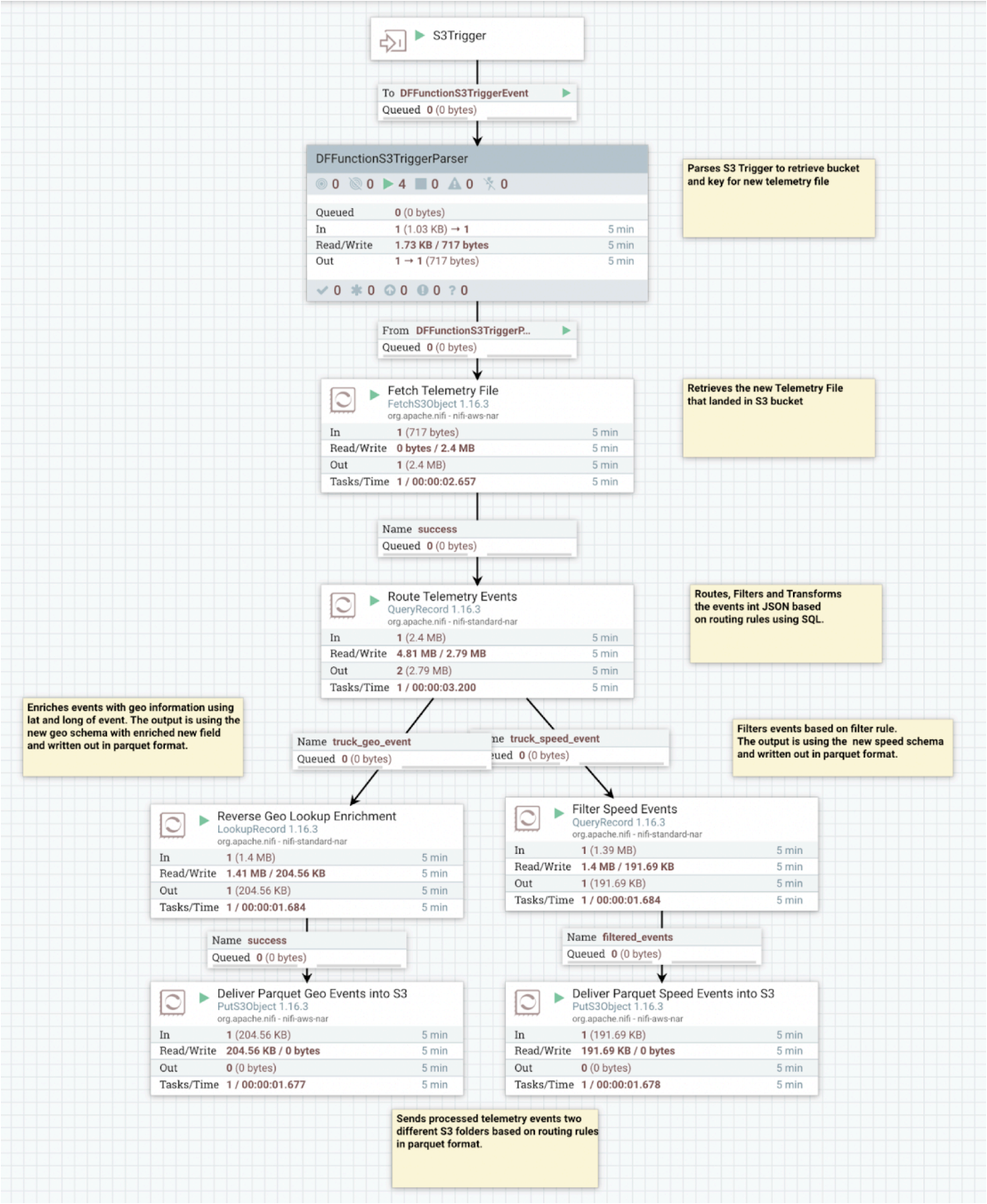
S3Trigger
To DFFunctionS3TriggerEvent
Queued 0 (0 bytes)
DFFunctionS3TriggerParser
Queued 0 (0 bytes)
In 1 (1.03 KB) → 1 5 min
Read/Write 1.73 KB / 717 bytes 5 min
Out 1 → 1 (717 bytes) 5 min
Parses S3 Trigger to retrieve bucket and key for new telemetry file
From DFFunctionS3TriggerP...
Queued 0 (0 bytes)
Fetch Telemetry File
FetchS3Object 1.16.3
org.apache.nifi - nifi-aws-nar
In 1 (717 bytes) 5 min
Read/Write 0 bytes / 2.4 MB 5 min
Out 1 (2.4 MB) 5 min
Tasks/Time 1 / 00:00:02.657 5 min
Retrieves the new Telemetry File that landed in S3 bucket
Name success
Queued 0 (0 bytes)
Route Telemetry Events
QueryRecord 1.16.3
org.apache.nifi - nifi-standard-nar
In 1 (2.4 MB) 5 min
Read/Write 4.81 MB / 2.79 MB 5 min
Out 2 (2.79 MB) 5 min
Tasks/Time 1 / 00:00:03.200 5 min
Routes, Filters and Transforms the events int JSON based on routing rules using SQL.
Enriches events with geo information using lat and long of event. The output is using the new geo schema with enriched new field and written out in parquet format.
Name truck_geo_event
Queued 0 (0 bytes)
truck_speed_event
0 (0 bytes)
Filters events based on filter rule. The output is using the new speed schema and written out in parquet format.
Reverse Geo Lookup Enrichment
LookupRecord 1.16.3
org.apache.nifi - nifi-standard-nar
In 1 (1.4 MB) 5 min
Read/Write 1.41 MB / 204.56 KB 5 min
Out 1 (204.56 KB) 5 min
Tasks/Time 1 / 00:00:01.684 5 min
Filter Speed Events
QueryRecord 1.16.3
org.apache.nifi - nifi-standard-nar
In 1 (1.39 MB) 5 min
Read/Write 1.4 MB / 191.69 KB 5 min
Out 1 (191.69 KB) 5 min
Tasks/Time 1 / 00:00:01.684 5 min
Name success
Queued 0 (0 bytes)
Name filtered_events
Queued 0 (0 bytes)
Deliver Parquet Geo Events into S3
PutS3Object 1.16.3
org.apache.nifi - nifi-aws-nar
In 1 (204.56 KB) 5 min
Read/Write 204.56 KB / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 1 / 00:00:01.677 5 min
Deliver Parquet Speed Events into S3
PutS3Object 1.16.3
org.apache.nifi - nifi-aws-nar
In 1 (191.69 KB) 5 min
Read/Write 191.69 KB / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 1 / 00:00:01.678 5 min
Sends processed telemetry events two different S3 folders based on routing rules in parquet format.

3. Validate the output of the test.
   - The output of the test should be a parquet file that contains telemetry data that is filtered and enriched based on the above requirements stored in the following s3 folders: <<your_bucket>>/processed/truck-geo-events and <<your_bucket>>/processed/truck-speed-events.

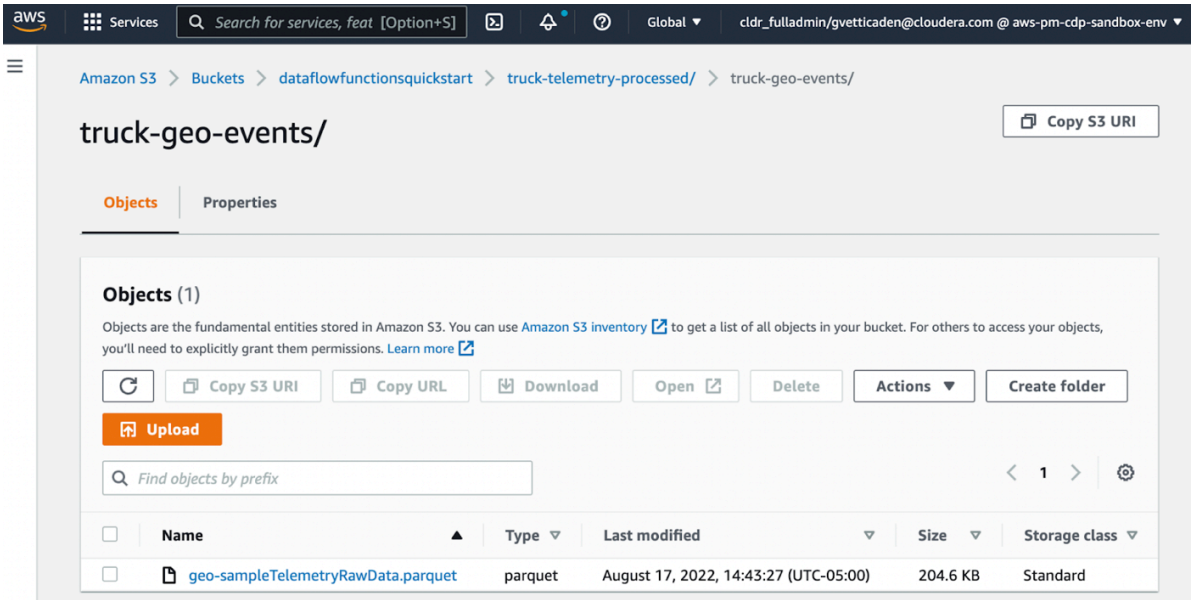

   - If the Parquet file is in each of these folders, the local test has completed successfully and the function works as expected and now can run on AWS Lambda.

5. Download the Telemetry Tutorial NiFi Flow

Download the Apache NiFi flow so that you can upload it into the Cloudera DataFlow Catalog and run it in serverless mode.

Procedure

1. Right-click the ‘Quickstart Function -TelemetryS3toS3Processing’ process group.

2. Select Download Flow Definition and Without external services.

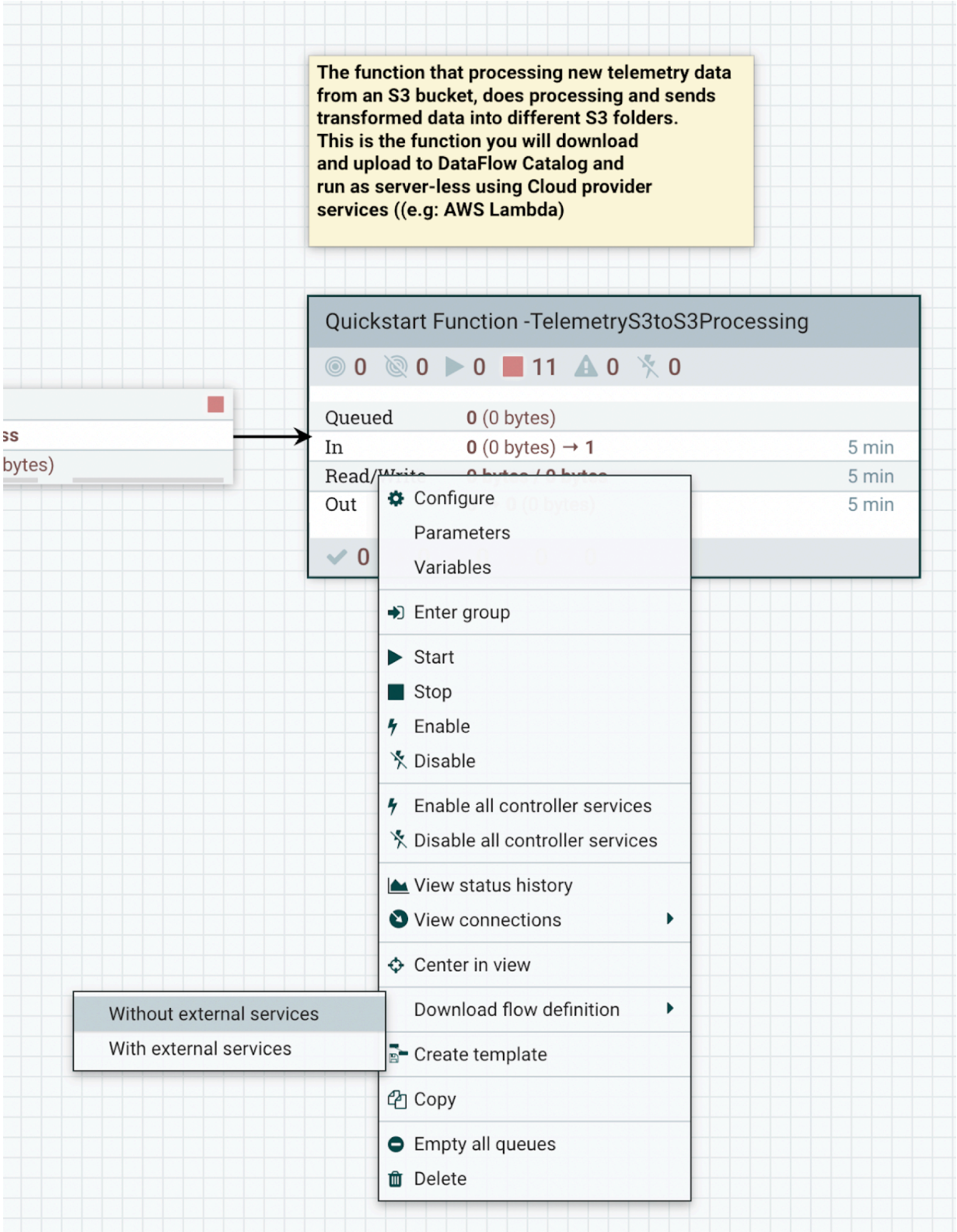

Add the Telemetry Tutorial NiFi Flow as a function in Cloudera DataFlow

Before you can run the Apache NiFi flow in serverless mode on AWS Lambda, you must register it in Cloudera DataFlow service and download the Lambda DF function handler libraries.

1. Upload the Telemetry Tutorial NiFi Flow to the Cloudera DataFlow Catalog

Procedure

1. After logging into the Cloudera Public Cloud console with the credentials that you used to register for the trial, select the Cloudera DataFlow service on the Cloudera Public Cloud homepage.
2. Click Catalog in the left navigation pane to display the Flow Catalog.
3. Click Import Flow Definition.
4. Provide a name and a description and upload the data flow you downloaded from your local NiFi instance.

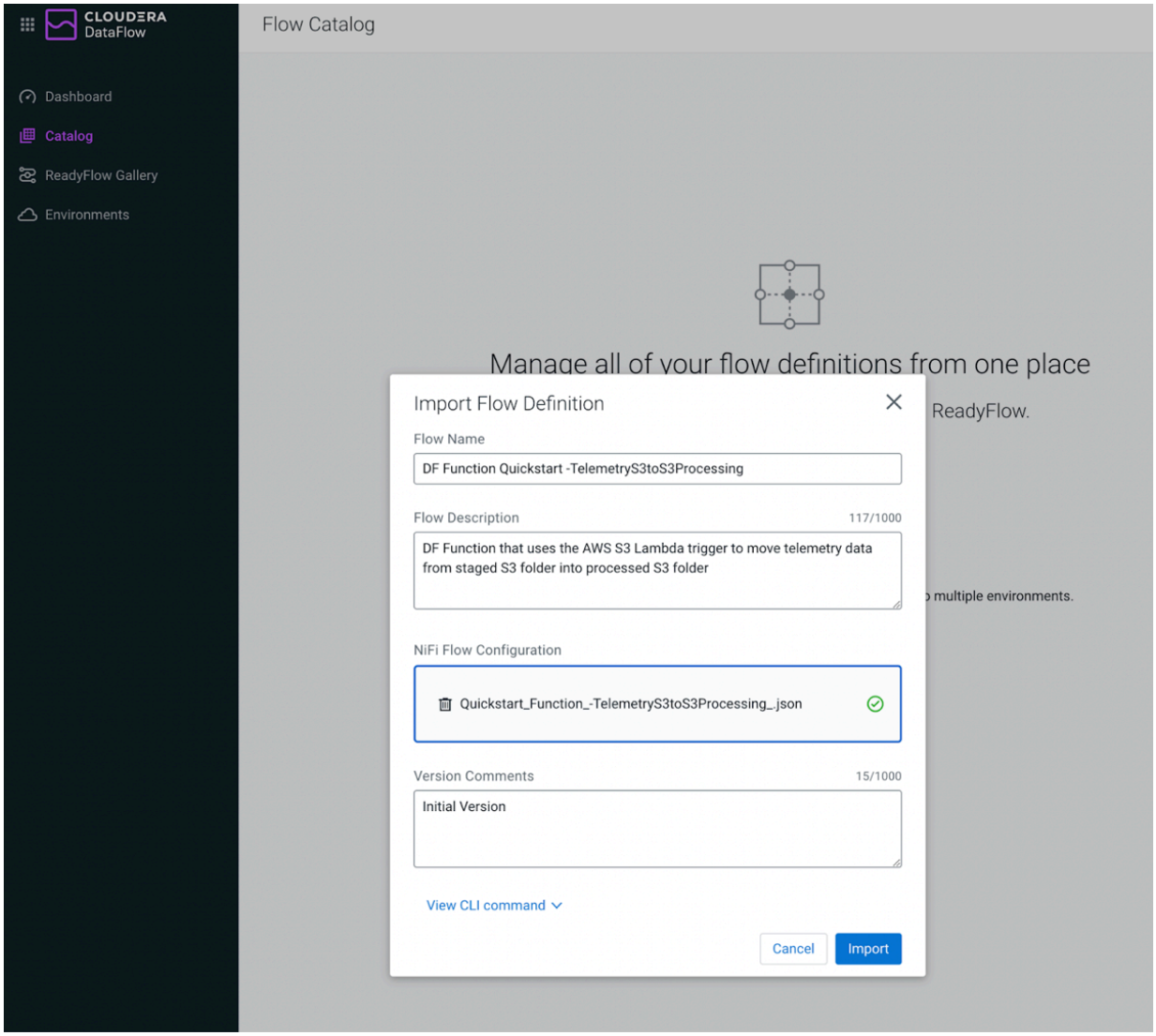

5. Click Import.

6. Copy the CRN # for version 1 of the flow you uploaded.

   Make sure to select the version's CRN in the orange box below. You will need it when configuring the function in AWS Lambda.

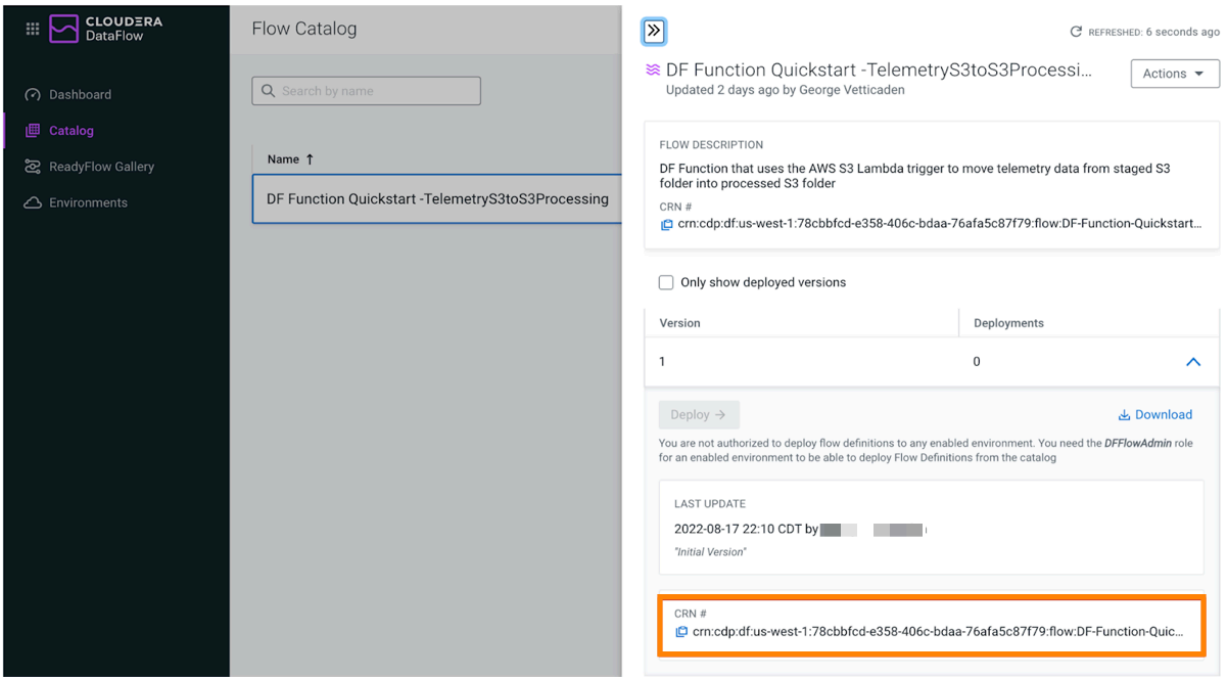

2. Download the Lambda Cloudera DataFlow Function binaries zip and upload to S3

To be able to run the NiFi flow in AWS Lambda, you need the Cloudera DataFlow function handler libraries.

Procedure

1. Click Functions in the left navigation pane and download the Cloudera DataFlow Function binaries for AWS Lambda.

   AWS Lambda will use these binaries to run the NiFi flow.

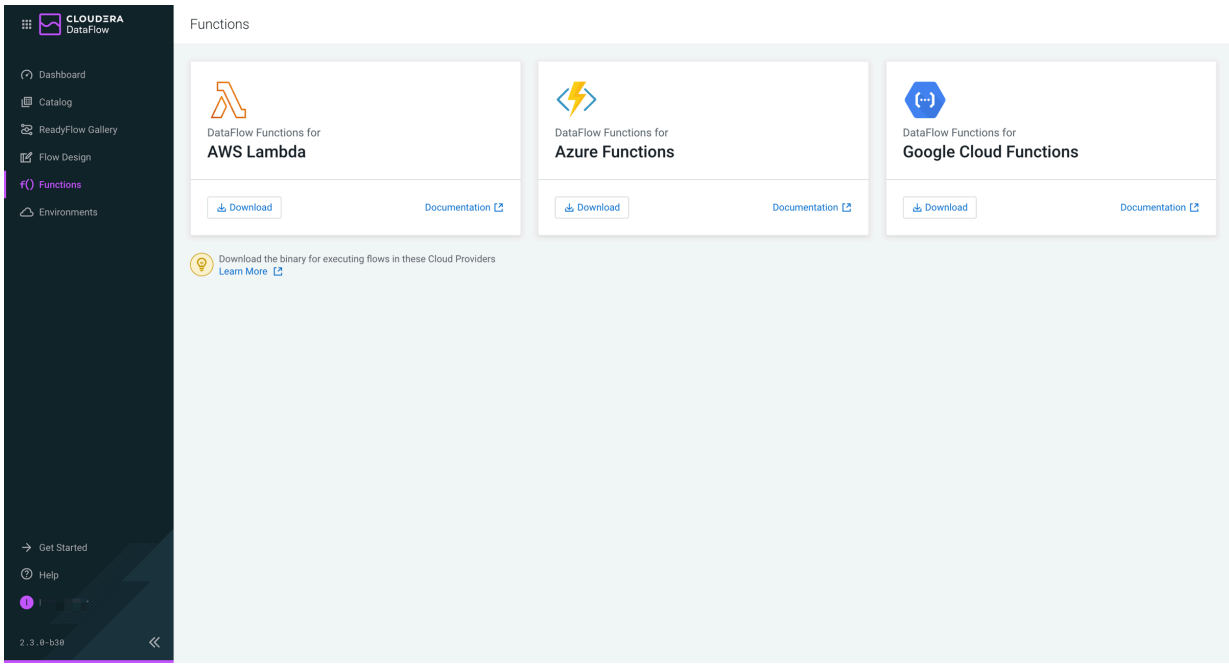

2. Upload this binary to an S3 bucket that you will later reference when creating the function in AWS Lambda. The S3 bucket needs to be in the same region as where the Lambda is being created/deployed.
3. Copy the S3 URI for later use.

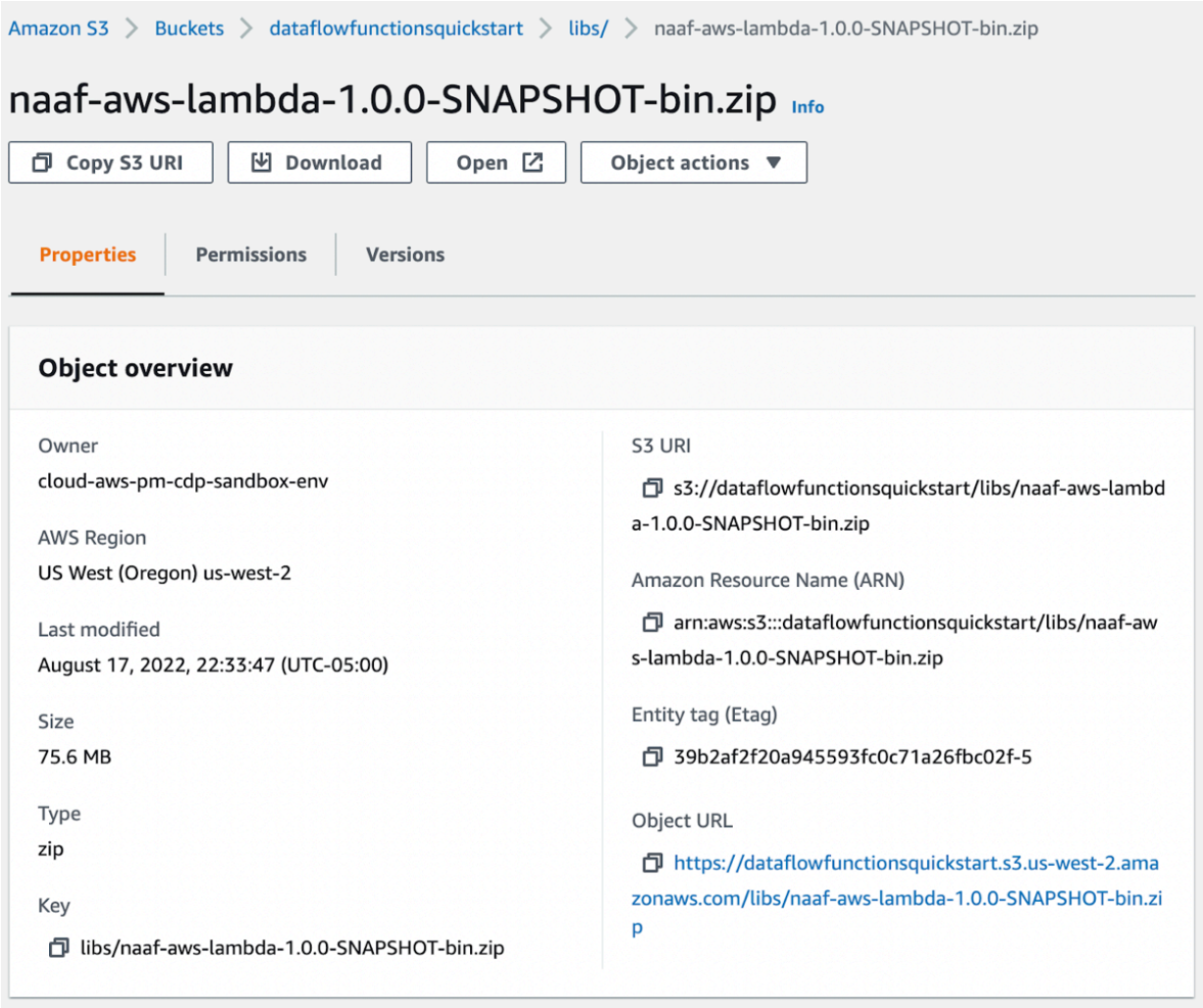

3. Create a Cloudera Public Cloud Service Account

You need to create a Cloudera Public Cloud service account for the AWS Lambda function to be able to retrieve the Cloudera DataFlow function from the Catalog where you uploaded it earlier.

Procedure

1. Go to the Cloudera Public Cloud homepage, select Management Console User Management .

2. Under Actions, select Create Machine User and provide a name for the service account.

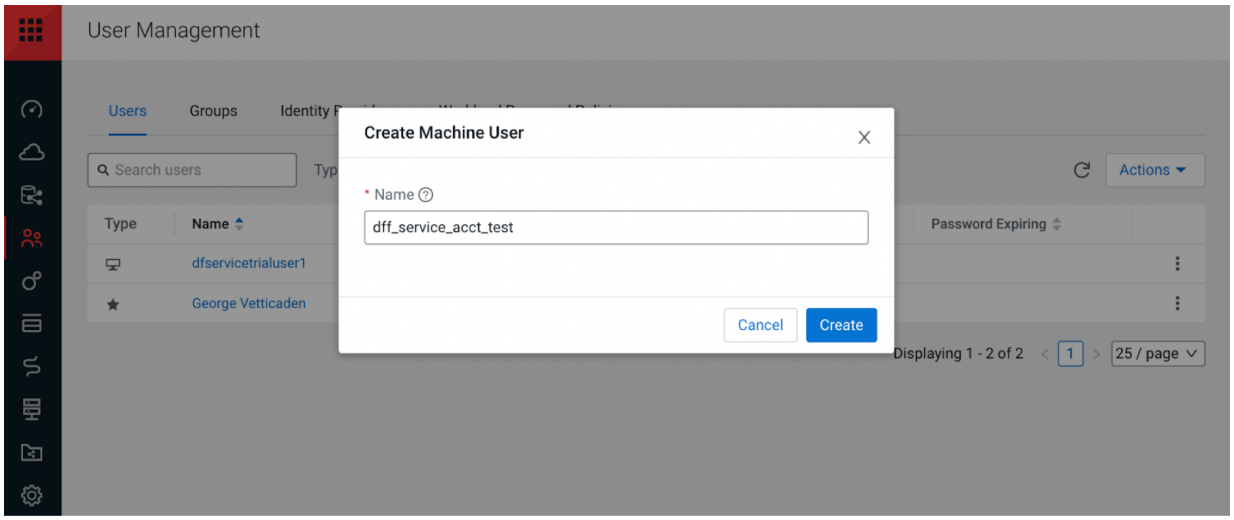

3. Click Roles Update Roles and select the DFFunctionMachineUser.

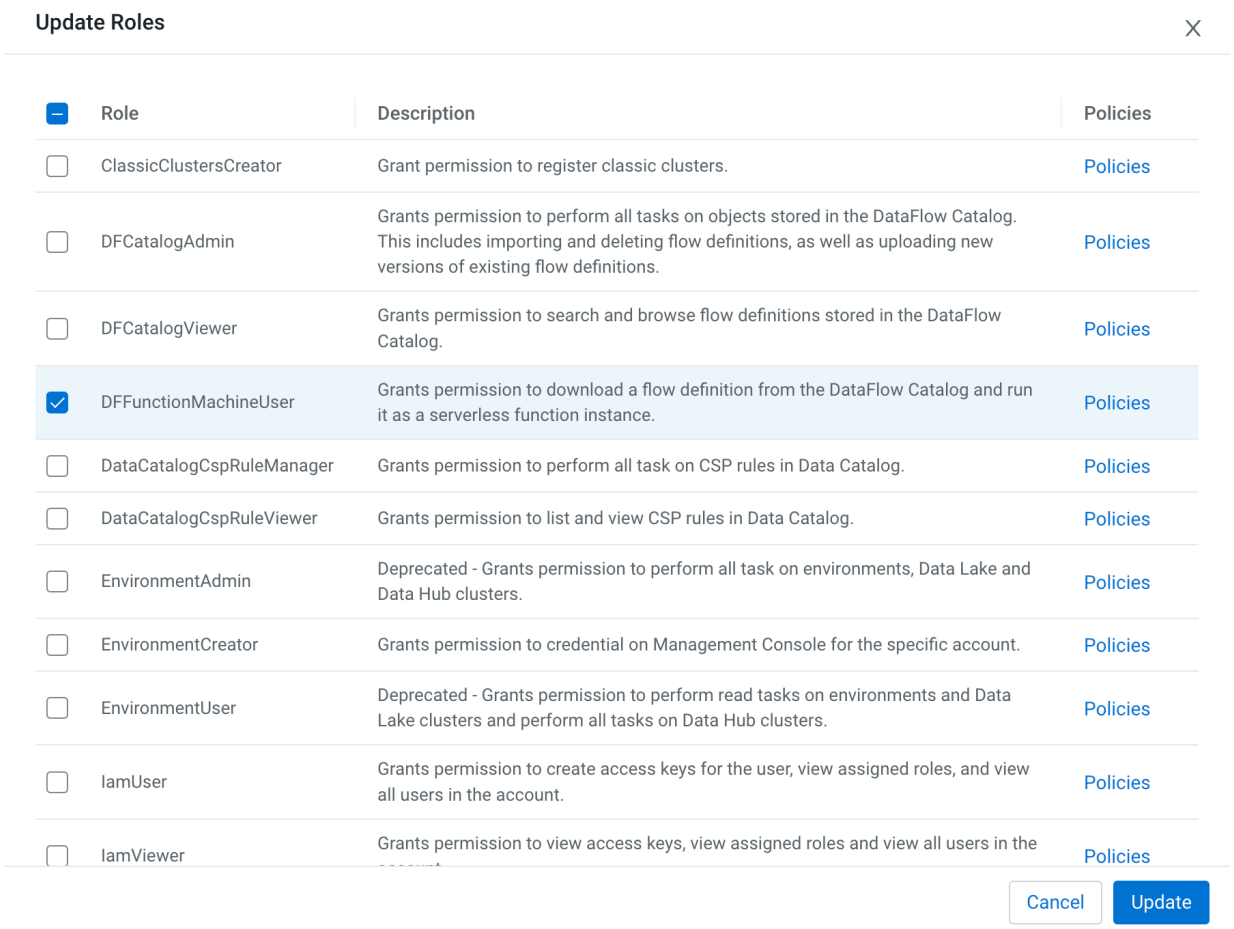

4. Click Update.
5. Select Access Keys Generate Access Key .

6. Save the Access Key ID and Private Key, which will be used to configure the AWS Lambda function.

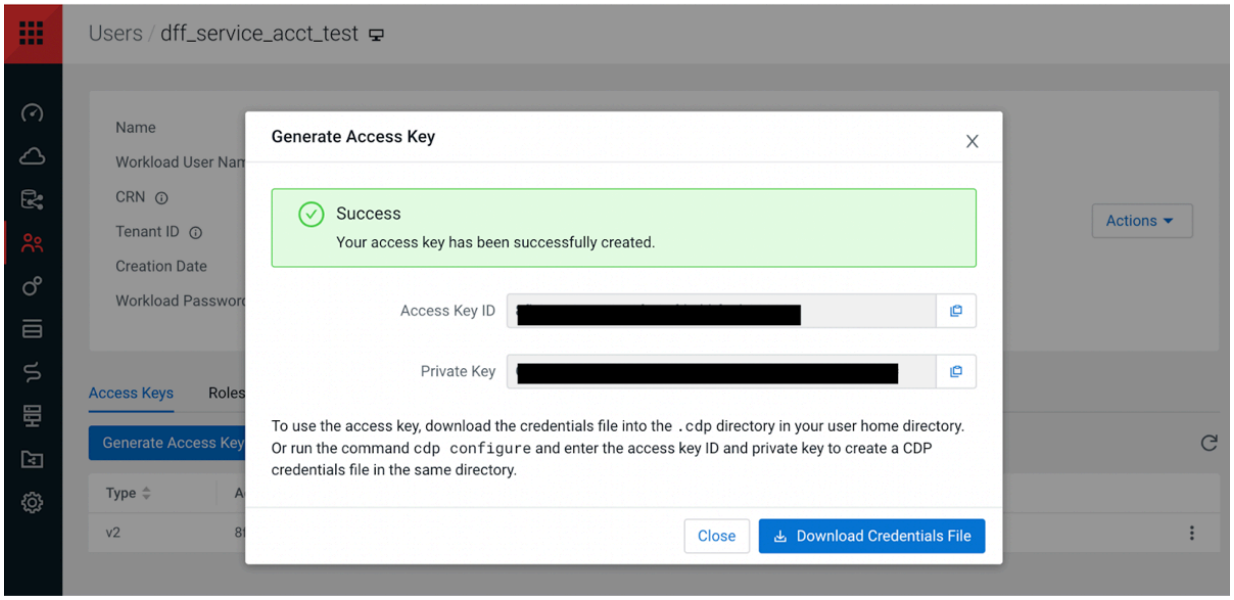

Run the Cloudera DataFlow function in serverless mode in AWS Lambda

Now that you have developed the NiFi flow and tested locally, registered it as Cloudera DataFlow function in Cloudera DataFlow service, you are ready to run the function in serverless mode using AWS Lambda. For this, you will need to create, configure, test and deploy the function in AWS Lambda.

1. Create the Cloudera DataFlow function

You can use the AWS CLI to create and configure the Cloudera DataFlow function in AWS Lambda.

Procedure

1. Create the AWS IAM Role required to create the lambda function.
   a) When Lambda executes a function, it requires an execution role that grants the function permission to access AWS services and resources. Lambda assumes the role when the Cloudera DataFlow function is invoked. Assign the most limited permissions/policies for the function to execute.
   b) Download the trust-policy.json file.
   c) Using the below AWS CLI command, create a role called NiFi_Function_Quickstart_Lambda_Role that the Lambda service will assume.

      The role will be attached to an AWS managed role that provides the limited permissions for the function to execute.

```
aws iam create-role --role-name NiFi_Function_Quickstart_Lambda_Role --a
ssume-role-policy-document file://trust-policy.json
```

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/service
-role/AWSLambdaBasicExecutionRole --role-name NiFi_Function_Quickstart_L
ambda_Role
```

d) These two commands will create the following IAM role. Copy and save the Role ARN which you will need to create the function.

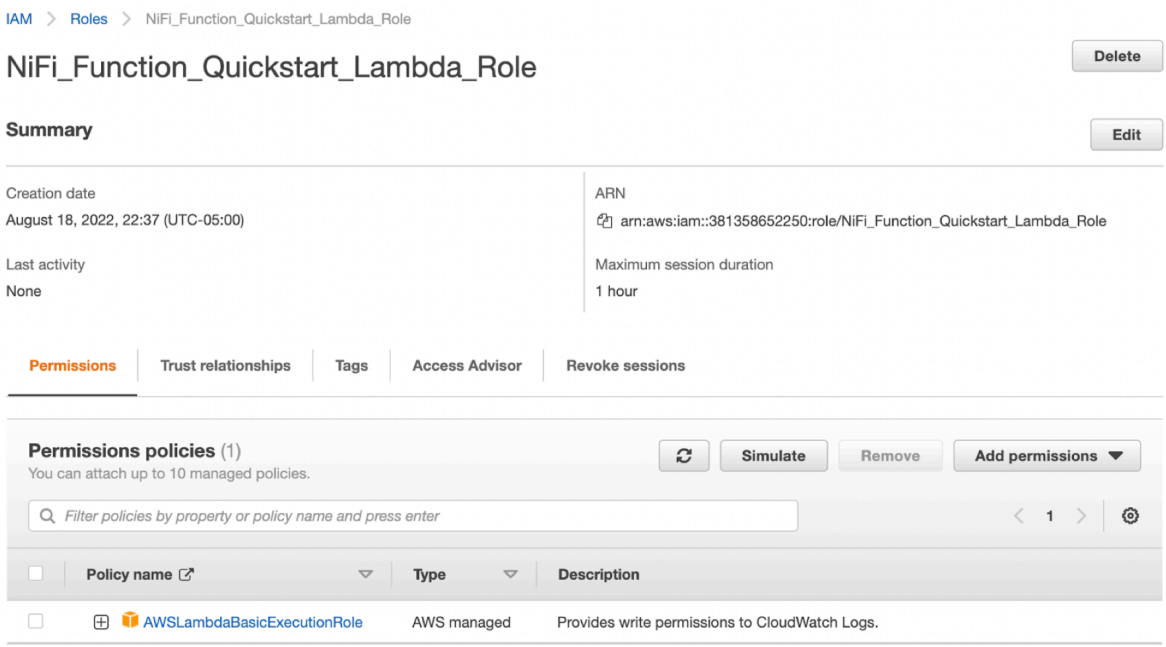

2. Create the Cloudera DataFlow function in Lambda.
   a) Download the Cloudera DataFlow Function Definition JSON file.
      - This file has the full definition required to create the Cloudera DataFlow function including function, code, environment variable and security configuration.
      - The environment variables in this file contain the info for Lambda to fetch the function definition from the Cloudera DataFlow Catalog as well as the function's application parameters.
   b) Update the following properties in the definition file:
      - DF_ACCESS_KEY – The access key created for the Cloudera Public Cloud service account
      - DF_PRIVATE_KEY – The private key created for the Cloudera Public Cloud service account
      - FLOW_CRN – The CRN value you copied from the Cloudera DataFlow Catalog page after uploading the function
      - aws_access_key_id – The AWS access key that has permissions to access (read/write) the S3 bucket you created in the prerequisite section
      - aws_access_key_password – The AWS access key password that has permissions to access (read/write) the S3 bucket you created in the prerequisite section
      - s3_bucket – The name of the bucket you created
      - s3_region – The bucket's region
      - S3Bucket – The name of the bucket that you uploaded the binaries ZIP file that you downloaded from the Cloudera DataFlow Functions page
      - S3Key – The key to the binaries ZIP file in S3.

        For example, if you uploaded the ZIP to S3 with this URI s3://dataflowfunctionsquickstart/libs/naaf-aws-lambda-1.0.0-SNAPSHOT-bin.zip, the key would be libs/naaf-aws-lambda-1.0.0-SNAPSHOT-bin.zip
      - Role – The ARN of the role created in the previous step

        Note: In the definition file, there is a parameter named NEXUS_URL. This parameter defines the location from where NARs will be downloaded when your function is instantiated.
        - If you have designed your NiFi flow in a Cloudera Flow Management environment, you need to remove the parameter from the definition file.
        - If you have designed your NiFi flow using an Apache NiFi environment as instructed before, you should keep this parameter.
   c) Run the following command to create a function called NiFi_Function_Quickstart (if the FunctionName property was not modified):

```
aws lambda create-function --cli-input-json file://NiFi_Function_Quickst
art-definition.json
```

3. View the Lambda function in AWS Console.
   a) On the AWS console, navigate to the Lambda service and click the function called NiFi_Function_Quickstart (if the FunctionName property was not modified).

      You can see the following under the Code tab of the function:

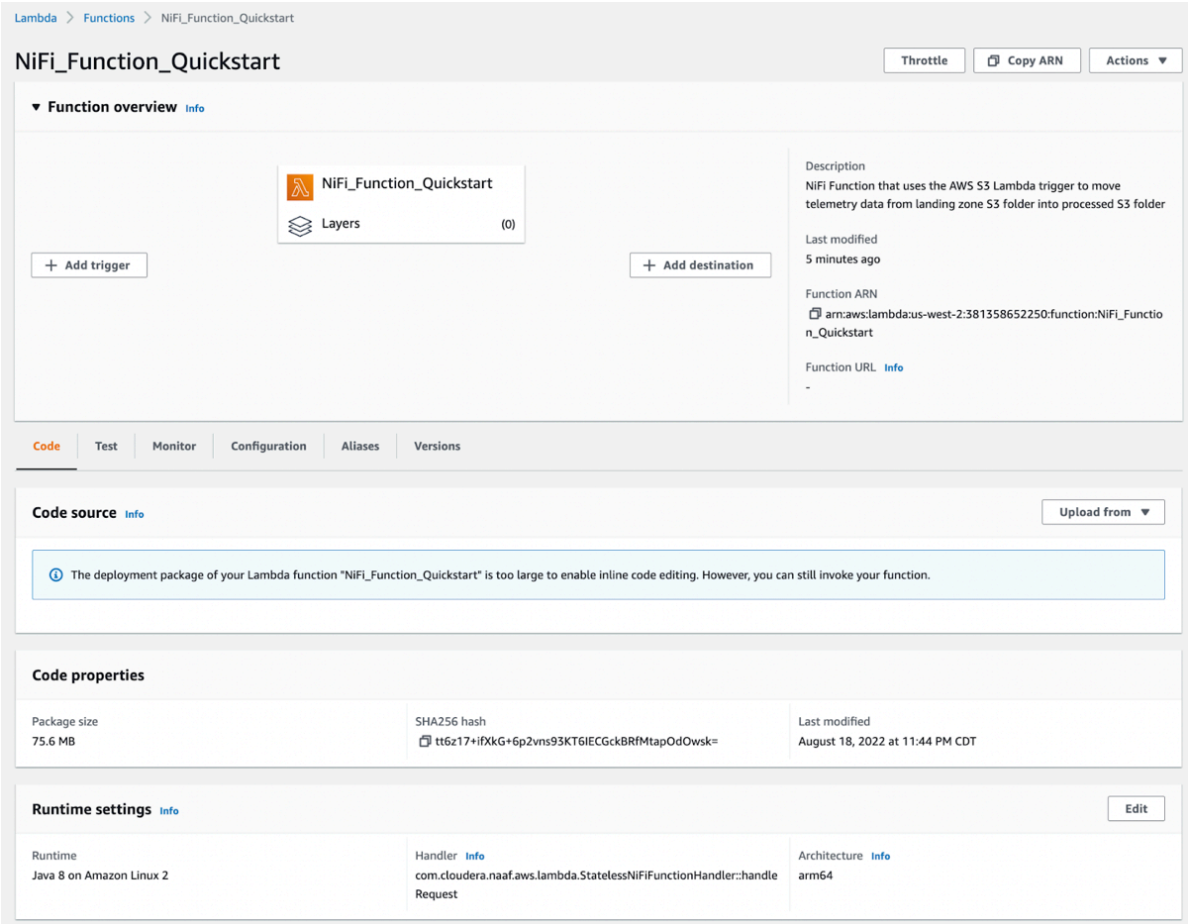

   b) If you click the Configuration tab, you can see all the configured parameters required to run the function under Environment variables.

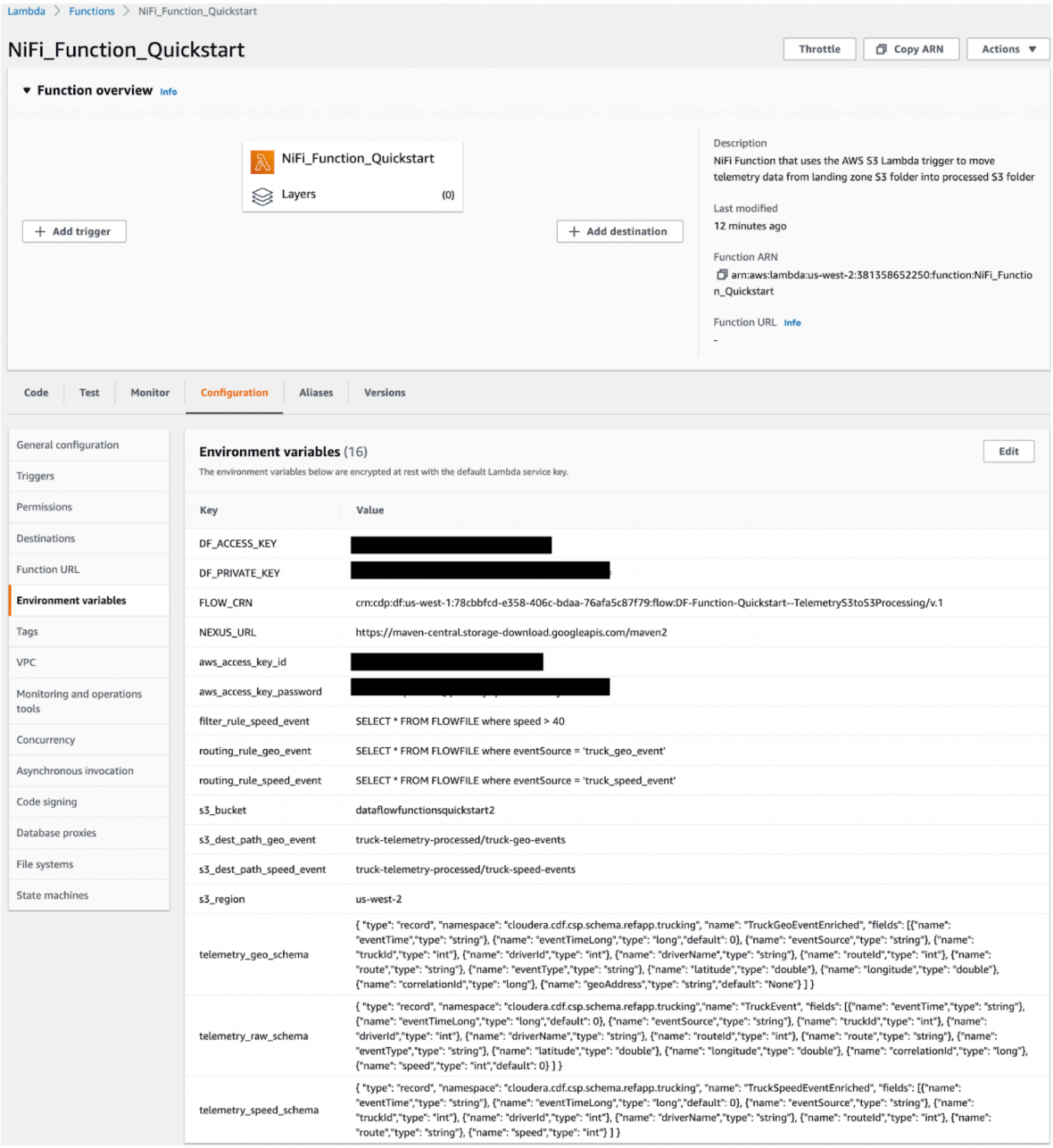

2. Test the Cloudera DataFlow function

With the function created and fully configured, you can test the function with a test trigger event before configuring the real trigger in the Lambda service.

Procedure

1. Create a Test Event.

   Click the Test tab, select Create New Event, and configure the following:

   a. Provide a name for the test: NiFi_Function_Quickstart.
   b. Copy the contents from the sampleTriggerEvent and paste it into the Event JSON field.
   c. Replace the bucket name with the one you created.
   d. Click Save.

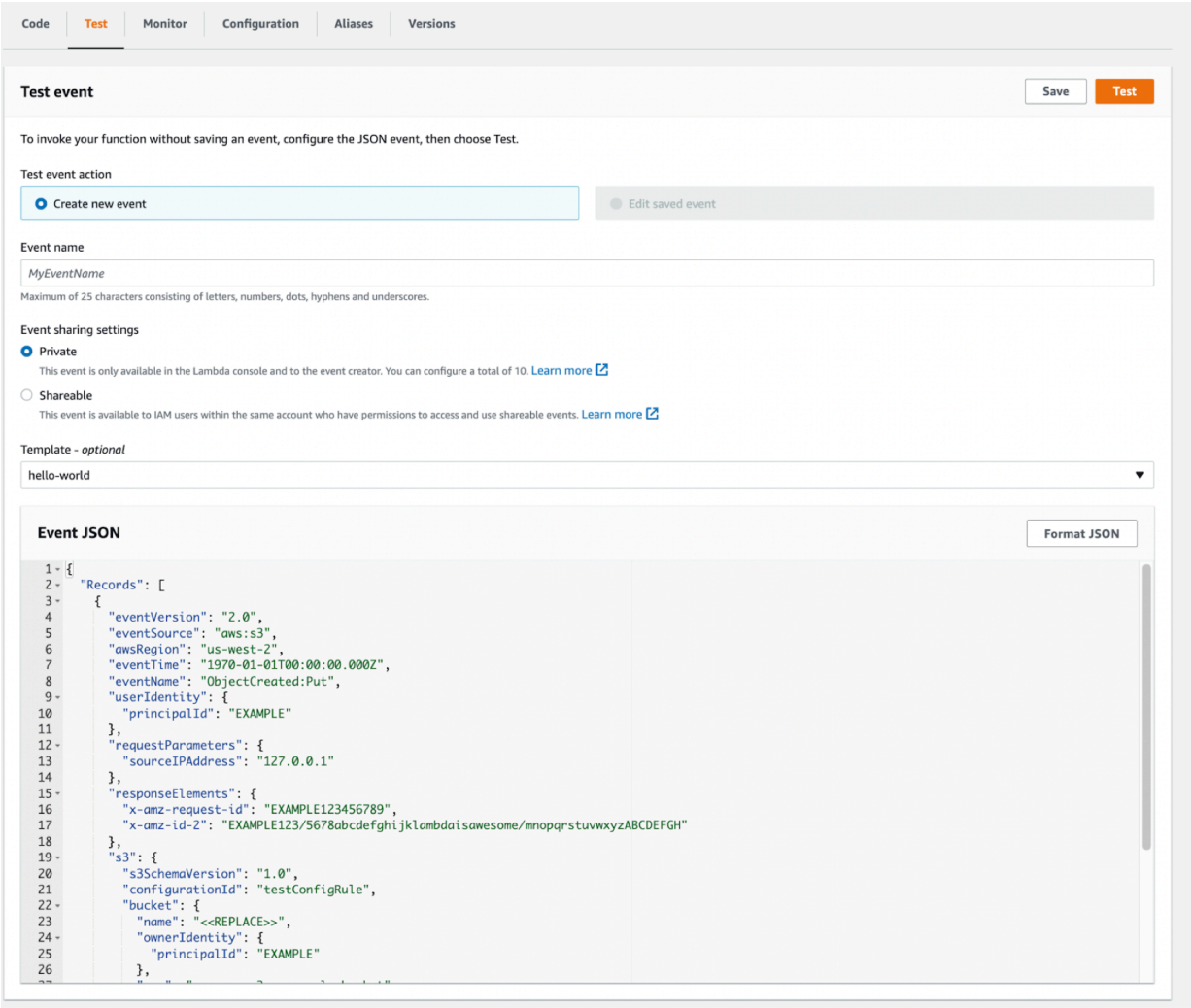

2. Execute the Test Event.
   a) Reset data for the test to run by deleting the folders under <<your_bucket>>/processed which was created after the test run on your local NiFi instance.
   b) Click Test.

      The initial run of the test is a cold start which will take a few minutes because it requires additional binaries to be downloaded from Nexus. Subsequent runs should be faster.

      If the run is successful, the logs should look like this:

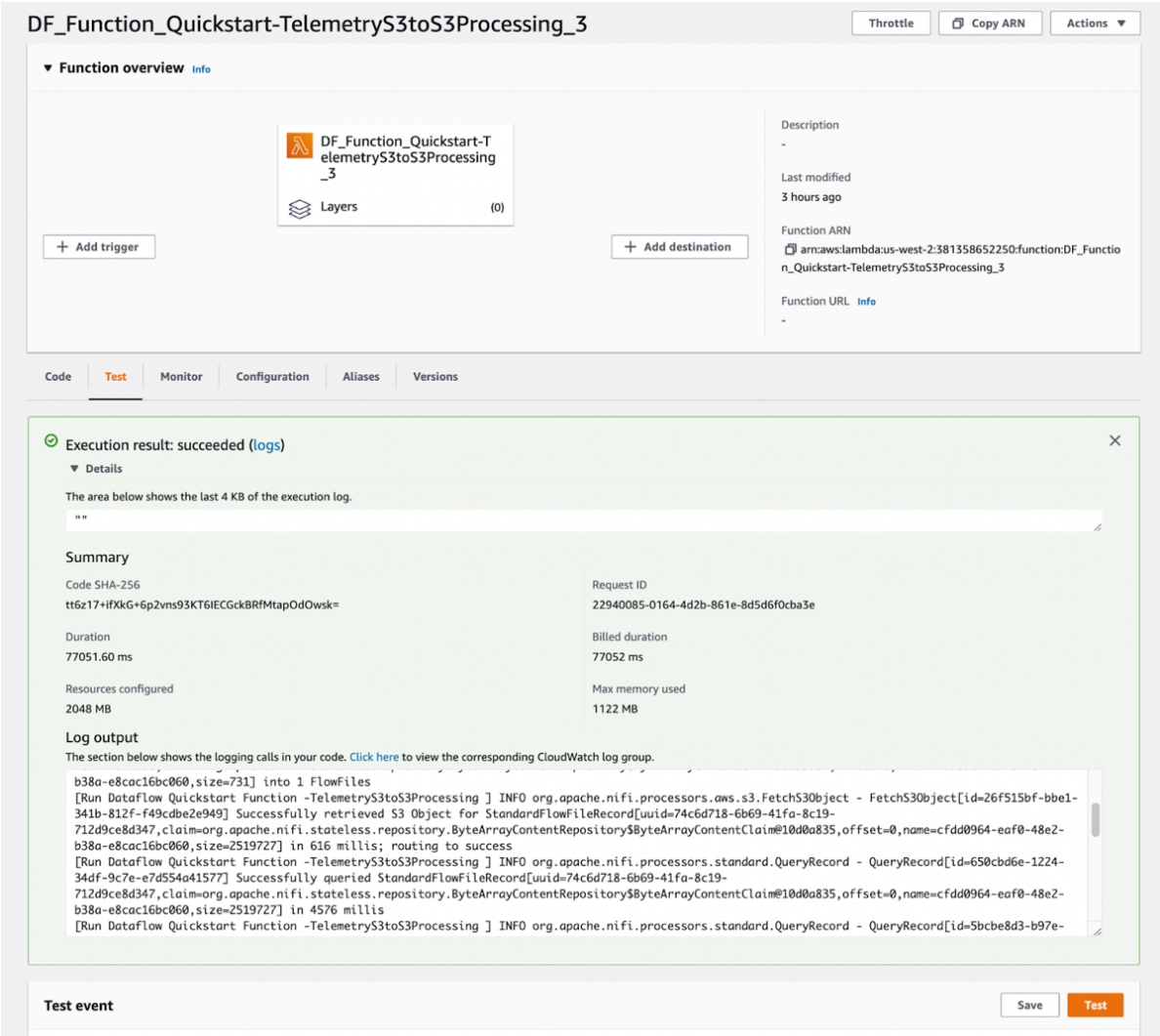

   c) Validate that the processed Parquet files are under the following directories:

      - <<your_bucket>>/processed/truck-geo-events
      - <<your_bucket>>/processed/truck-speed-events

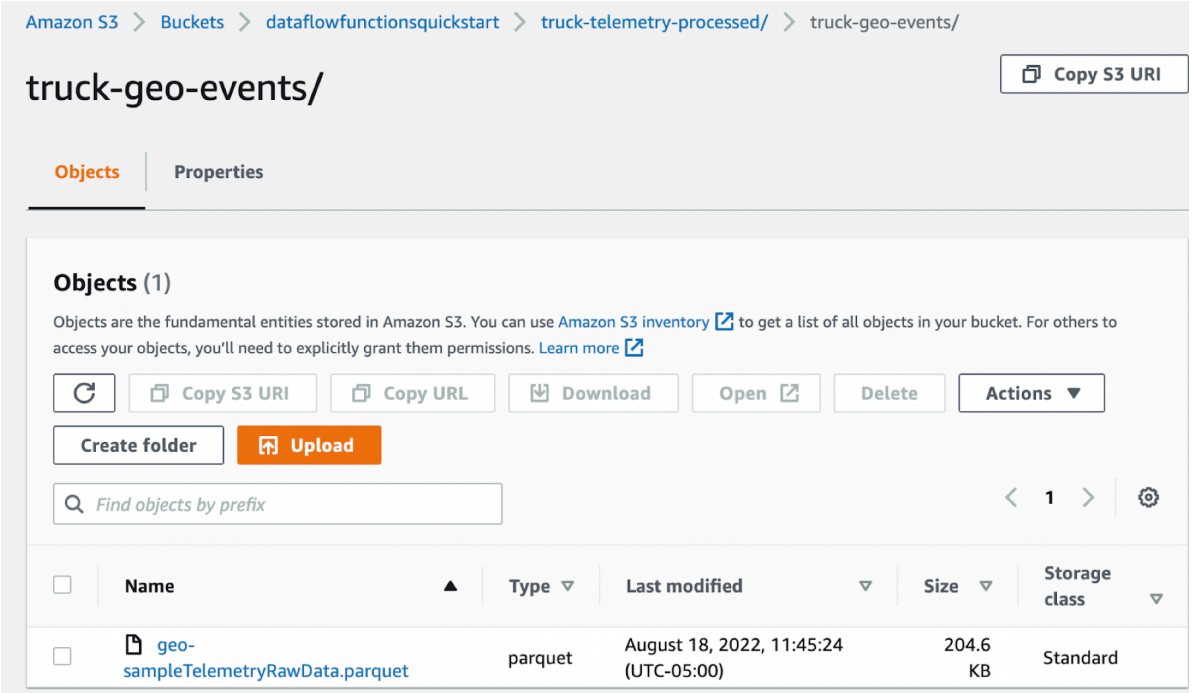

3. Create S3 trigger for the Cloudera DataFlow function

With the Cloudera DataFlow function fully configured in Lambda and tested using a sample trigger event, you can now create a S3 trigger for the function.

Procedure

1. Select the function that you created and click Add Trigger in the Function overview section.

2. Select S3 as the trigger source and configure the following:
   - Bucket – Select the bucket you created
   - Event type – All object create events
   - Prefix – truck-telemetry-raw/
   - Click the checkbox to acknowledge recursive invocation

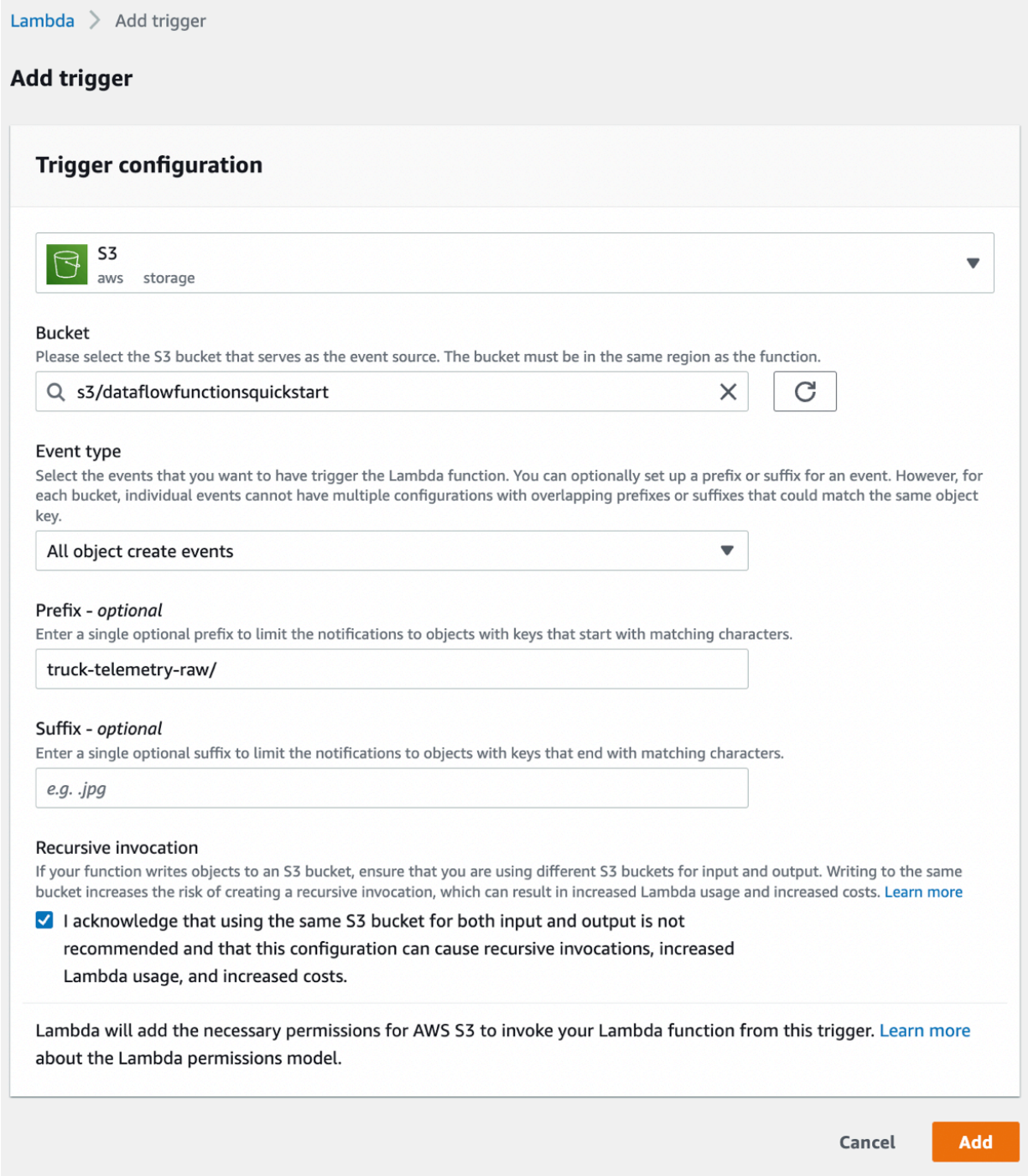

3. Click Add.
4. With the trigger created, when any new telemetry file lands in <<your_bucket>>/truck-telemetry-raw, AWS Lambda will execute the Cloudera DataFlow function.

   With the trigger created, when any new telemetry file lands in <<your_bucket>>/truck-telemetry-raw, AWS Lambda will execute the Cloudera DataFlow function.

5. You can test this by uploading the sample telemetry file into <<your_bucket>>/truck-telemetry-raw.

   It will generate a trigger event that spins up the Cloudera DataFlow function which results in the processed files landing in <<your_bucket>>/processed/truck-geo-events and <<your_bucket>>/processed/truck-speed-events.

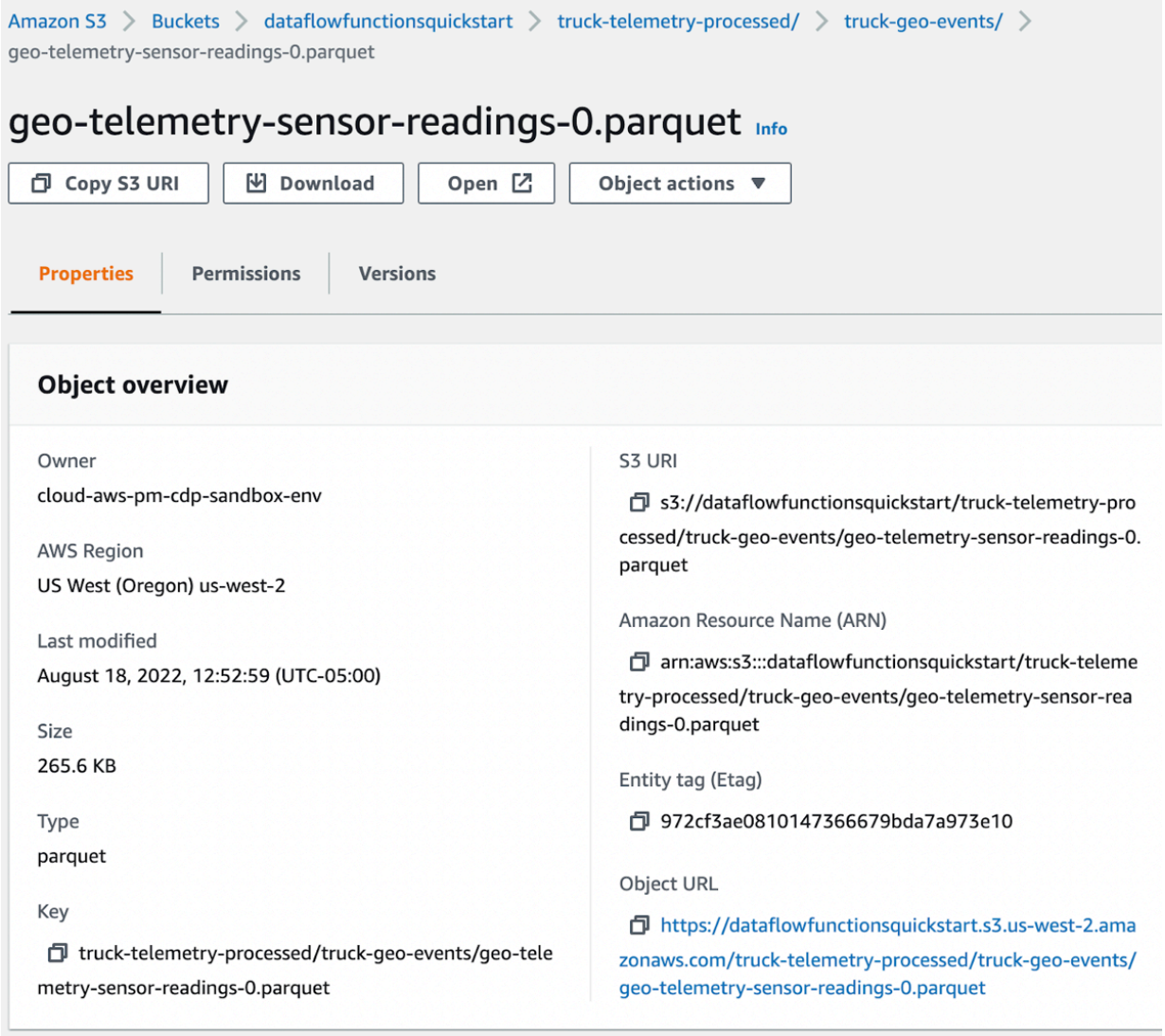

4. Monitor the Cloudera DataFlow function

As more telemetry files are added to the landing S3 folder, you can view the metrics and logs of the serverless Cloudera DataFlow Functions in the AWS Lambda monitoring view.

Procedure

- If you want to view metrics on all the function invocations, go to the Monitor tab and select the Metrics sub-tab.

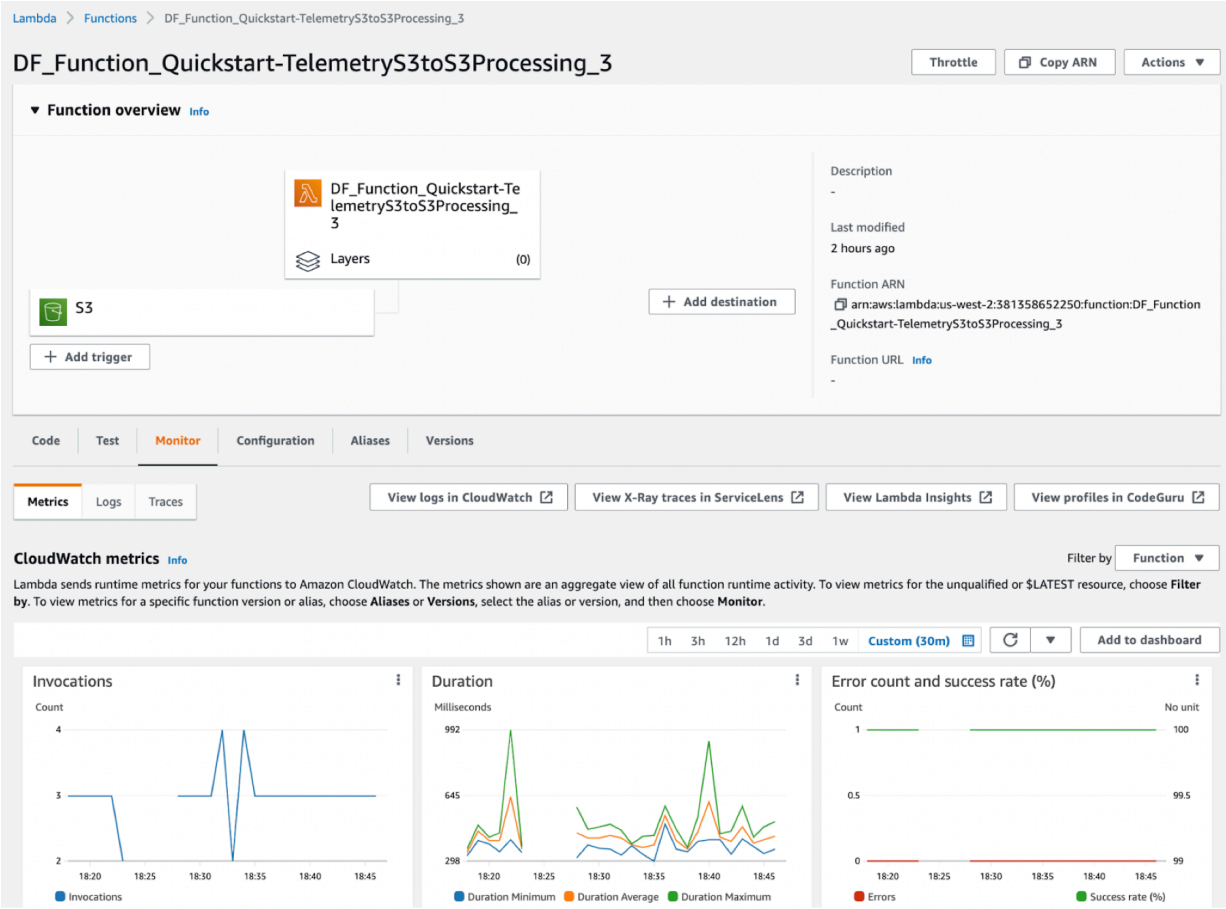

- If you want to view all the Cloudera DataFlow function logs for each function invocation, check out the Logs sub-tab under the Monitor tab.

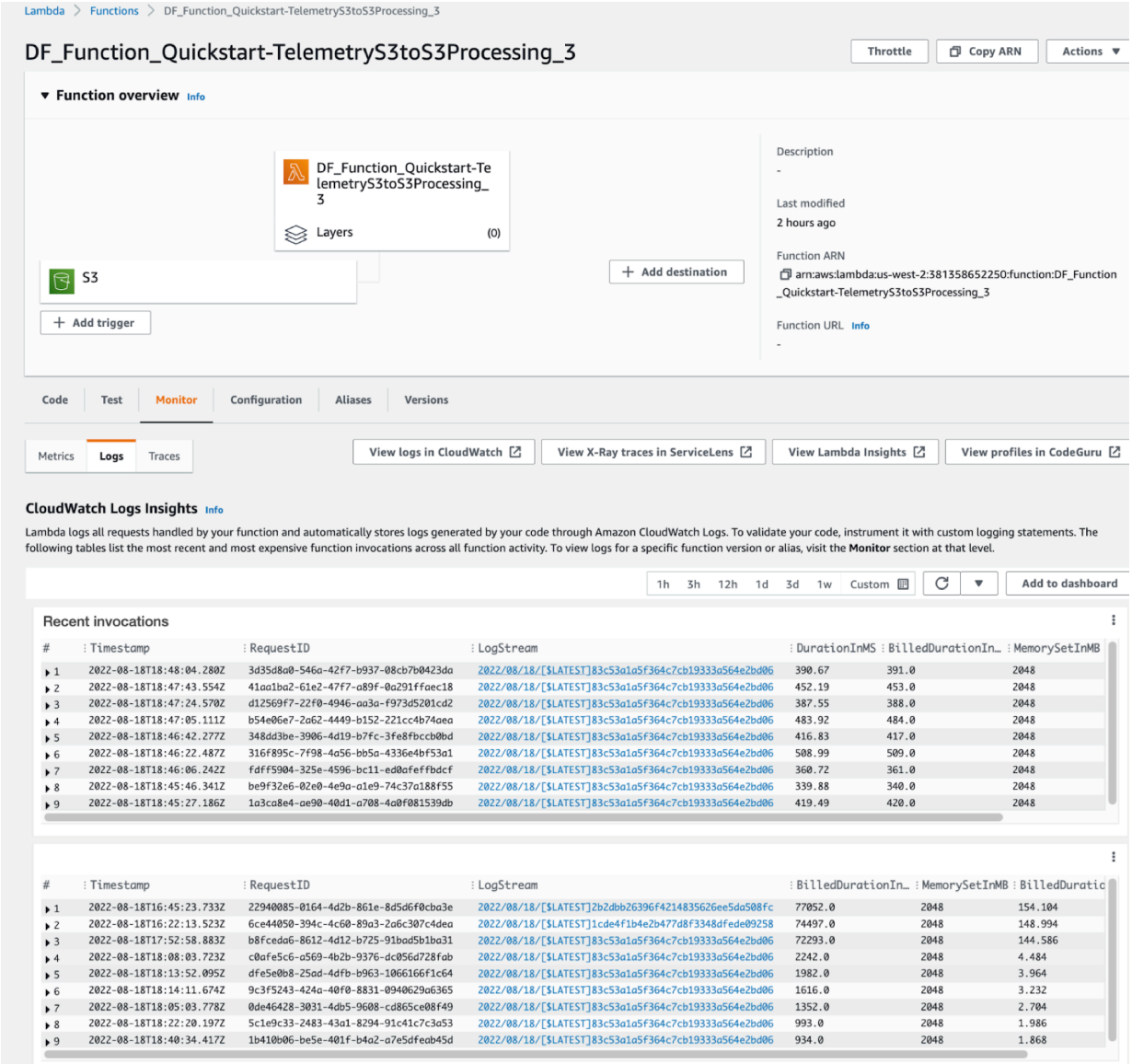

..

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

.....4

Cloudera Data Hub

Accessing Clusters

Date published: 2019-12-17
Date modified: 2023-06-27

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enabling admin and user access to Cloudera Data Hub clusters..... 4

Understanding Cloudera Data Hub cluster details..... 4

Accessing Cloudera Manager, cluster UIs, and endpoints..... 7

Accessing a Cloudera Data Hub cluster via SSH..... 8

Set workload password..... 9

Finding your workload user name..... 10

Retrieving keytabs for workload users..... 11

Running workloads..... 12

Enabling admin and user access to Cloudera Data Hub clusters

Cloudera Data Hub resource roles can be assigned on the scope of a specific Cloudera Data Hub cluster.

When you grant access to admins and users of a Cloudera Data Hub, consider the following guidelines:

- Any user or group that needs to access a specific Cloudera Data Hub needs the EnvironmentUser role at the scope of the environment where that Cloudera Data Hub cluster is running.
- A user with the DataHubCreator (or higher) account role can create Cloudera Data Hub clusters.
- The user who creates a Cloudera Data Hub cluster gets the Owner role for that Cloudera Data Hub cluster.
- The Owner of the Cloudera Data Hub cluster can grant others access to the cluster. The following roles can be assigned:
  - Owner - Grants the permission to mange the Cloudera Data Hub cluster in Cloudera and delete it. It does not grant any cluster-level access (such as access to Cloudera Manager).
  - DataHubAdmin (Technical Preview) - Grants administrative rights over the Cloudera Data Hub cluster.

The roles are described in detail in *Resource roles*. The steps for assigning the roles are described in *Assigning resource roles to users* and *Assigning resource roles to groups*.

Related Information

Resource roles

Assigning resource roles to users

Assigning resource roles to groups

Understanding Cloudera Data Hub cluster details

To access information about Data Hub clusters, navigate to the Data Hub Clusters service or to the Cloudera Management Console service > Data Hub Clusters.

Each cluster is represented by a tile that includes basic information about the cluster:

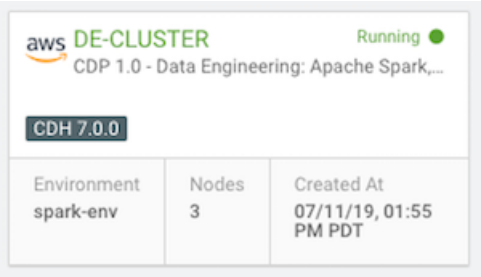

Click the cluster tile to display more information:

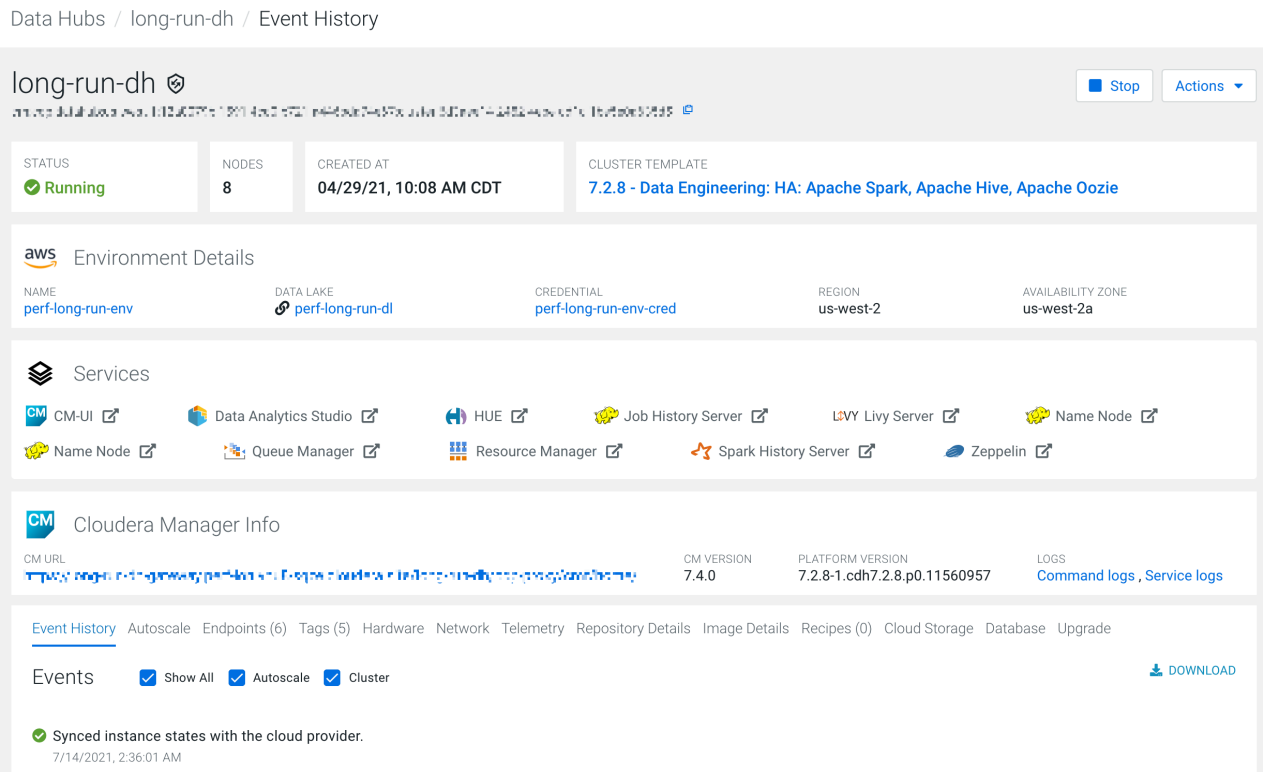

The summary bar includes the following information:

Item	Description
Status	Current cluster status. When a cluster is healthy, the status is Running.
Nodes	The current number of cluster nodes.
Created at	The date and time when the cluster was created. The format is MM/DD/YY, Time AM/PM Timezone. For example: 06/20/19, 7:56 AM PDT.
Cluster Template	The name of the cluster template used to create the cluster.

Environment Details

You can find information about the cluster cloud provider environment under Environment Details:

Item	Description
Cloud Provider	The logo of the cloud provider on which the cluster is running.
Name	The name of the environment used to create the cluster.
Data Lake	The name of a Data Lake to which the cluster is attached.
Credential	The name of the credential used to create the cluster.
Region	The region in which the cluster is running in the cloud provider infrastructure.
Availability Zone	The availability zone within the region in which the cluster is running.

Services

In the Services section, you will find links to cluster UIs. The exact content depends on what components are running on your cluster.

Cloudera Manager Info

The Cloudera Manager Info section provides the following information:

Item	Description
Cloudera Manager URL	Link to the Cloudera Managerr web UI.
Cloudera Manager Version	The Cloudera Manager version which the cluster is currently running.
Platform Version	The Cloudera Runtime version which the cluster is currently running.

Event History and other tabs

Under Cloudera Manager, the Event History tab is displayed by default. You can also click the other tabs to view additional cluster information.

Item	Description
Event History	Shows events logged for the cluster, with the most recent event at the top. The Download option allows you to download the event history.
Hardware	This section includes information about your cluster instances: instance names, instance IDs, instance types, their status, fully qualified domain names (FQDNs), and private and public IPs. If you click on the » , you can access more information about the instance, storage, image, and packages installed on the image.
Tags	This section lists keys and values of the user-defined tags, in the same order as you added them.
Endpoints	This section includes the endpoints for various cluster servces.
Recipes	This section includes recipe-related information. For each recipe, you can see the host group on which a recipe was executed, recipe name, and recipe type.
Repository Details	This section includes Cloudera Manager and Cloudera Runtime repository information, as you provided when creating a cluster.
Image Details	This section includes information about the prewarmed or base image that was used for the cluster nodes.
Network	This section includes information about the names of the network and subnet in which the cluster is running and the links to related cloud provider console.
Cloud Storage	This section provides information about the base storage locations used for YARN and Zeppelin.
Database	This section provides information about any external managed database you might have created for the cluster.
Telemetry	This section provides information about logging, metering, cluster log collection, and other analytics.

Actions Menu

Click Show Cluster Template on the Actions menu to review the template used in cluster creation. Click Show CLI Command to review the CDP CLI command used to create the cluster (which you can copy to create similar clusters via the CDP CLI). Select Manage Access to manage access to the cluster.

You can also perform some basic Cloudera Manager management functions from the Actions menu, such as resizing, retrying, and repairing the cluster, as well renewing the host certificate.

Accessing Cloudera Manager, cluster UIs, and endpoints

Cluster UIs and endpoints can be accessed from cluster details.

Required role: EnvironmentUser at the scope of the environment where the Data Hub is running, but Cloudera Manager access is read-only. EnvironmentAdmin grants a limited administrator role in Cloudera Manager. DatahubAdmin or the Owner of the Cloudera Data Hub can access cluster details, but access to Cloudera Manager is read-only.

To access cluster UIs and endpoints navigate to the to the Data Hub Clusters service and click the tile for your cluster. This opens the cluster details page, which lists the URLs for the cluster UIs and endpoints:

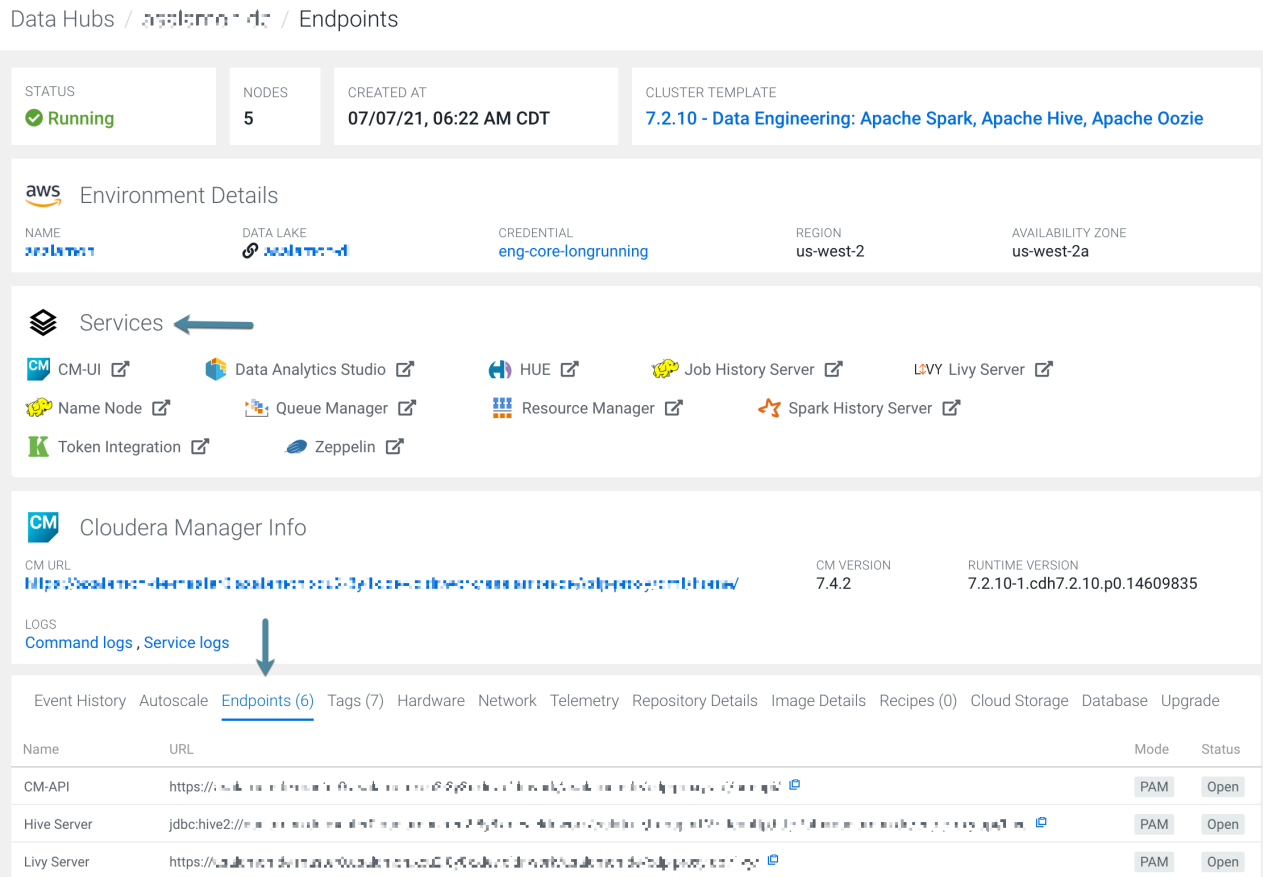

Click the URL for the service that you would like to access and you will be logged in automatically with your Cloudera credentials. All of the UIs and endpoints are accessible via the Knox gateway. The URLs listed connect you to a chosen service via Knox, and Knox securely passes your Cloudera credentials.

Credentials to use for logging in

The following table lists the credentials to use to access clusters:

Method	URL	Credentials
Cloudera Manager web UI	Access from the URL listed in cluster details > Services section.	You do not need to provide any credentials. You are automatically logged in with your Cloudera credentials. When accessing a Cloudera Data Hub cluster via Cloudera Manager, you assume the Configurator role.

Method	URL	Credentials
All cluster web UIs	Access from the URLs listed in cluster details.	You do not need to provide any credentials. You are automatically logged in with your Cloudera credentials.
Data Analytics Studio (DAS)	Access from the URLs listed in cluster details.	Access DAS with your workload user name and workload password. When accessing Cloudera for the first time, you must set a workload password. For more instructions on how to set your workload password, refer to Set or reset workload password. For instructions on how to find your workload user name, refer to Finding your workload user name.
All cluster endpoints	Access by using the API endpoint listed in cluster details > Endpoints tab.	Access all cluster API endpoints (such as JDBC and ODBC) with your workload user name and workload password. When accessing Cloudera for the first time, you must set a workload password. For more instructions on how to set your workload password, refer to Set or reset workload password. For instructions on how to find your workload user name, refer to Finding your workload user name. For information on how to set up a connection from a business intelligence tool such as Tableau, refer to Configuring JDBC for ImpalaConfiguring JDBC for Impala and Configuring ODBC for ImpalaConfiguring ODBC for Impala.

Security exception

The first time you access the UIs, your browser will attempt to confirm that the SSL Certificate is valid. Since CDP automatically generates a certificate with self-signed CA, your browser will warn you about an untrusted connection and ask you to confirm a security exception. Depending on your browser, perform the steps below to proceed:

Browser	Steps
Firefox	Click Advanced > Click Add Exception… > Click Confirm Security Exception
Safari	Click Continue
Chrome	Click Advanced > Click Proceed…

Accessing a Cloudera Data Hub cluster via SSH

You can use SSH to access cluster nodes via a command line client.

Non-root SSH access to cluster VMs

Required role: Any user who has access to the environment (EnvironmentUser, DataSteward, and EnvironmentAdmin) can access Cloudera Data Hub clusters via SSH.

To execute privileged operating system actions on a Cloudera Data Hub, the EnvironmentPrivilegedUser role is required.

All authorized users can access cluster nodes via SSH using either a private key that matches the user's public key previously provided in Cloudera, or their workload user name and workload password.

For SSH access through a workload user name and password:

When accessing Cloudera for the first time, you must set a workload password. The password also needs to be reset each time you are added to a new environment.

For more information about workload passwords and instructions for setting/resetting it, refer to Set or Reset Workload Password.

For instructions on how to find your workload user name, refer to Finding Your Workload User Name.

On Mac OS, you can use the following syntax to SSH to the VM:

```
$ ssh <workloaduser>@<nodeIPaddress>
```

For example:

```
$ ssh jsmith@190.101.0.132
```

To SSH to a cluster using the private key file that pairs with the public key associated with a user, use the ssh utility:

```
$ ssh -i <path-to-private-key-file> <cdpusername>@<nodeIPaddress>
```

For example:

```
$ ssh -i ~/.ssh/my-private-key jsmith@192.12.141.12
```

On Windows, you can access your cluster via SSH by using an SSH client such as PuTTY. For more information, refer to How to use PuTTY on Windows.

Root SSH access to cluster VMs

Required role: None

Cloudera administrators can access cluster nodes as cloudbreak user with the SSH key provided during environment creation.

On Mac OS, you can use the following syntax to SSH to the VM:

```
ssh -i <path-to-cloudbreak-private-key-file> cloudbreak@<nodeIPaddress>
```

For example:

```
ssh -i ~/.ssh/cloudbreak-private-key cloudbreak@p90.101.0.132
```

On Windows, you can access your cluster via SSH by using an SSH client such as PuTTY. For more information, refer to How to use PuTTY on Windows.

Set workload password

A workload password is used to access Cloudera Data Hub clusters via SSH, endpoints such as JDBC/ODBC, and some UIs such as DAS.

Required role: All users can manage their workload passwords from the account management page. All users can manage their workload password from CDP CLI, but this action requires an API access key, which can only be generated by users with the IAMUser role. As a Cloudera administrator or PowerUser, you can manage the workload password for all user accounts.

The workload password is independent from your SSO password.

To set or reset the password:

1. Sign in to the Cloudera web interface.
2. Navigate to Cloudera Management Console > User Management.
3. Search for your user name, and then click your user name:

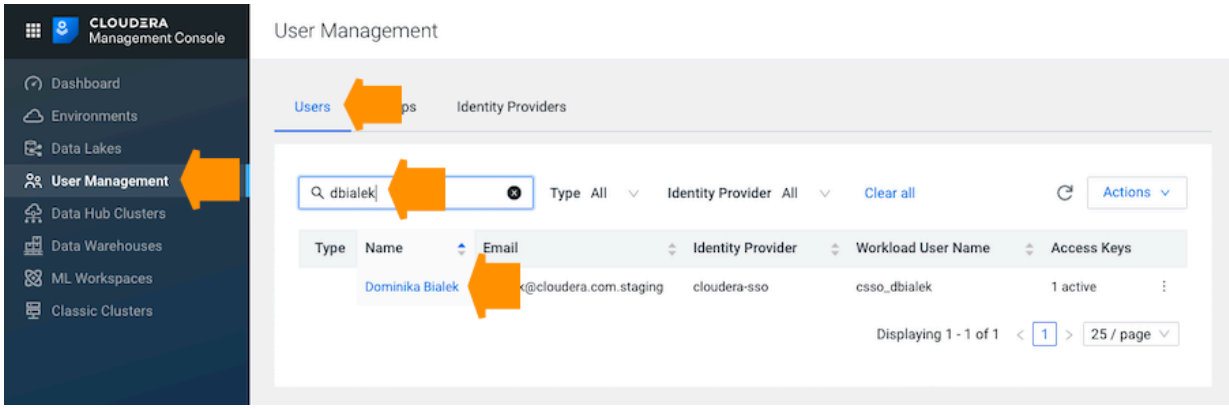

4. Click Set Password for User:

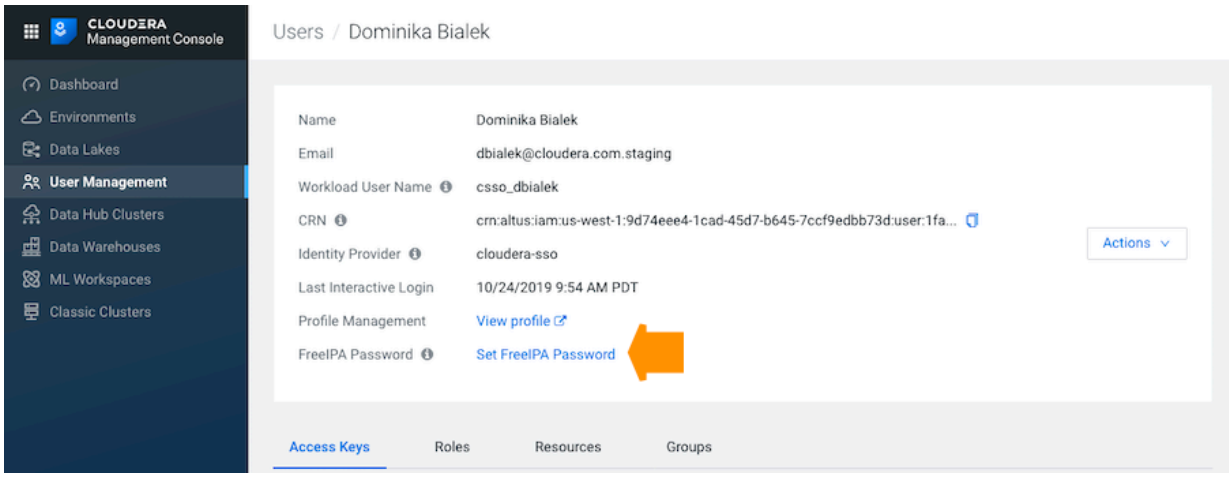

5. In the dialog box that appears, enter the workload password twice:

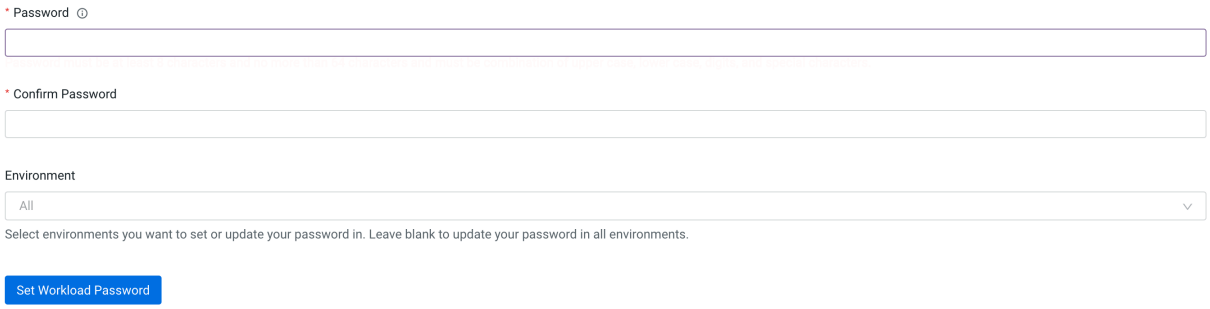

6. Click Set Workload Password.

Finding your workload user name

Once you have reset your workload password, locate your workload user name.

To check your workload user name, select Cloudera Management Console > User Management > Users, find your user name, and then find your Workload User Name:

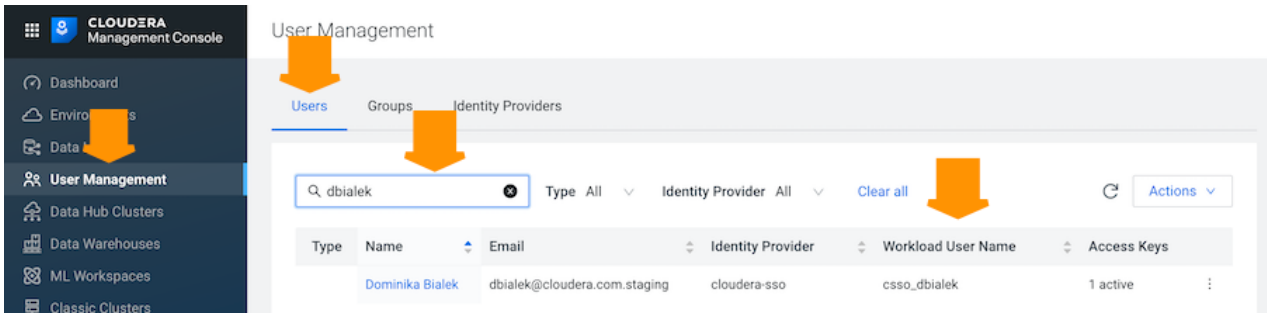

Retrieving keytabs for workload users

A keytab file stores long-term keys for a principal in Kerberos. You can generate a keytab either through the Cloudera Management Console user interface or the CDP CLI.

About this task

Required roles: All users can retrieve their keytabs from the account management page. All users can retrieve their keytabs from CDP CLI, but this action requires an API access key, which can only be generated by users with the IAMUser role. As a Cloudera administrator or PowerUser, you can retrieve the keytab for all user accounts.

You may need to generate a keytab for a workload user in certain Cloudera Data Hub use cases, for example long-running Spark streaming jobs, which require a keytab as a long-lived credential; or NiFi flows requiring a keytab to write data into HBase.

Note: Keytabs are scoped to an environment, whereas workload passwords are the same for every environment. A keytab is, however, tied to the workload password. If you change the workload password, you must retrieve a new keytab. When you change a workload password, retrieve the keytab only after the user sync operation is complete. For more information on user sync, see *Assigning resources to users*.

Procedure

You can retrieve a keytab either in the Cloudera Management Console or in the CDP CLI:

- Cloudera Management Console:
  - **a.** Click User Management Users and then search for and select the Name of the user that you want to get a keytab for.
  - **b.** Click Actions Get Keytab .
  - **c.** Select the environment in which the Cloudera Data Hub cluster is running and then click Download.
  - **d.** Save the keytab file in a location of your choice.

    Once you have downloaded the keytab file, you can copy it to the machine on which the cluster runs and use the keytab to authenticate as the workload user principal, or point to the keytab file when running a Spark job or other job that requires a keytab.
- CDP CLI:
  - **a.** Keytab retrieval (get-keytab) is within the environments module. Run cdp environments get-keytab help for more information. You will need to pass the environment name and an actor CRN:

```
cdp environments get-keytab \
--environment-name=EnvironmentName \
--actor-crn=ActorCrn
```

b. The output of the command is a base64-encoded representation of a keytab. The contents of the output must be base64 decoded and saved to a file for it to work as a keytab.

Note: There are ways to generate keytabs with utilities outside of Cloudera, such as ipa-getkeytab or ktutil. Cloudera recommends against using these methods as they may not work as expected. For example, ipa-getk eytab creates a keytab that may work but only temporarily.

Related Information

Assigning resources to users
CLI client setup

Running workloads

Once your cluster is running, refer to the Cloudera Runtime and Cloudera Data Hub documentation for information on how to run workloads using services such as Hive, Spark, Impala, Hue, or Kafka.

Related Information

Data Access
Data Science
Streaming
Flow Management
Operational Database

Cloudera Data Hub

Cluster Definitions

Date published: 2019-12-17
Date modified: 2023-06-27

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cluster definitions 4

Default cluster configurations 4

Data Engineering clusters 5
Data Mart clusters 13
Operational Database clusters 14
Streams Messaging clusters 15
Flow Management clusters 16
Streaming Analytics clusters 18
Data Discovery and Exploration clusters 19

Save a custom cluster definition 20

Delete a cluster definition 20

Manage cluster definitions from CLI 21

Cluster definitions

A cluster definition is a reusable cluster template in JSON format that can be used for creating multiple Cloudera Data Hub clusters with identical cloud provider settings.

Data Hub includes a set of default cluster definitions for common data analytics and data engineering use cases, and allows you to save your own custom cluster definitions. While default cluster definitions can be used across all environments, custom cluster definitions are associated with one or more specific environments. Cluster definitions facilitate repeated cluster reuse and are therefore useful for creating automation around cluster creation.

The easiest way to create a custom cluster definition is to open the create cluster wizard, provide all the parameters, and then save the settings as a cluster definition. The option to save is available on the last page of the create cluster wizard.

Note:

A cluster definition is not synonymous with a cluster template. While a cluster template primarily defines Cloudera Runtime services, a cluster definition primarily includes cloud provider settings. Furthermore, a cluster definition always references one specific cluster template.

Prior to creating your own cluster definitions, we recommend that you review the default cluster definitions to check if they meet your requirements. You can access the default cluster definitions by clicking Environments, then selecting an environment and clicking the Cluster Definitions tab. To view details of a cluster definition, click on its name. For each cluster template, you can access a graphical representation ("list view") and a raw JSON file ("raw view") of all cluster host groups and their components.

Default cluster configurations

Cloudera Data Hub includes a set of prescriptive cluster configurations. Each of these default cluster configurations include a cloud-provider specific cluster definition, which primarily defines cloud provider settings. The cluster definition references a cluster template, which defines a number of Cloudera Runtime or Cloudera DataFlow components used for common data analytics and data engineering use cases.

Refer to the topic for each default cluster configuration to view the included services and compatible Runtime versions. These topics include links to documentation that will help you to understand the included components and use the workload cluster.

Many of the cluster components are included in the Cloudera Runtime software distribution. The Streams Messaging, Cloudera Flow Management, and Cloudera Streaming Analytics cluster configurations are part of Cloudera DataFlow for Data Hub and have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for more details.

You can access the default cluster definitions by clicking Environments, then selecting an environment and clicking the Cluster Definitions tab.

You can access the default cluster templates from Shared ResourcesCluster Templates.

To view details of a cluster definition or cluster template, click on its name. For each cluster definition, you can access a raw JSON file. For each cluster template, you can access a graphical representation ("list view") and a raw JSON file ("raw view") of all cluster host groups and their components.

Related Information

Cloudera DataFlow for Data Hub

Cloudera Runtime

Data Engineering clusters

Learn about the default Data Engineering clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Engineering provides a complete data processing solution, powered by Apache Spark and Apache Hive. Spark and Hive enable fast, scalable, fault-tolerant data engineering and analytics over petabytes of data.

Data Engineering cluster definition

This Data Engineering template includes a standalone deployment of Spark and Hive, as well as Apache Oozie for job scheduling and orchestration, Apache Livy for remote job submission, and Hue and Apache Zeppelin for job authoring and interactive analysis.

Cluster definition names

- Data Engineering for AWS
- Data Engineering for Azure
- Data Engineering for Google Cloud
- Data Engineering HA - Spark3 for AWS

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering HA - Spark3 for Azure

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering HA - Spark3 for Google Cloud
- Data Engineering Spark3 for AWS
- Data Engineering Spark3 for Azure
- Data Engineering Spark3 for Google Cloud

Cluster template name

- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: Apache Spark, Apache Hive, Apache Oozie."
- Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: HA: Apache Spark, Apache Hive, Apache Oozie."

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: The "Data Engineering: Apache Spark3" cluster template is deleted. Therefore, the "Data Engineering: Apache Spark3, Apache Hive, Apache Oozie" cluster template can be used instead.

Included services

- HDFS
- Hive
- Hue
- Livy
- Spark 3
- Yarn
- Zeppelin
- ZooKeeper
- Oozie is supported for Spark 3 as of Cloudera Runtime version 7.2.18

- Hive Warehouse Connector is supported as of Cloudera Runtime version 7.2.16.

Compatible Cloudera Runtime version

7.2.16, 7.2.17, 7.2.18

Topology of the Data Engineering cluster

Topology is a set of host groups that are defined in the cluster template and cluster definition used by Data Engineering. Data Engineering uses the following topology:

Host group	Description	Node configuration
Master	The master host group runs the components for managing the cluster resources including Cloudera Manager, Name Node, Resource Manager, as well as other master components such HiveServer2, HMS, Hue etc.	1 For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.4xlarge; gp2 - 100 GB Azure : Standard_D16_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-16; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE, DE Spark3, and DE HA: AWS : m5.4xlarge; gp2 - 100 GB Azure: Standard_D16_v3 GCP : e2-standard-16; pd-ssd - 100 GB
Worker	The worker host group runs the components that are used for executing processing tasks (such as NodeManager) and handling storing data in HDFS such as DataNode.	3 For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.2xlarge; gp2 - 100 GB Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-8; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: AWS: r5d.2xlarge - (gp2/EBS volumes) Azure: Standard_D5_v2 GCP : e2-standard-8; pd-ssd - 100 GB DE HA: AWS: r5d.4xlarge - (gp2/EBS volumes)

Host group	Description	Node configuration
		Azure: Standard_D5_v2 GCP : e2-standard-8; pd-ssd - 100 GB
Compute	The compute host group can optionally be used for running data processing tasks (such as NodeManager). By default the number of compute nodes is set to 1 for proper configurations of YARN containers. This node group can be scaled down to 0 when there are no compute needs. Additionally, if load-based auto-scaling is enabled with minimum count set to 0, the compute nodegroup will be resized to 0 automatically.	0+ For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.2xlarge; gp2 - 100 GB Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-8; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: AWS: r5d.2xlarge - (ephemeral volumes) Azure: Standard_D5_v2 For Azure, the attached volume count for the compute host group is changed to 0. Only ephemeral/local volumes are used by default. GCP : e2-standard-8; pd-ssd - 100 GB DE HA: AWS: r5d.4xlarge - (ephemeral volumes) Azure: Standard_D5_v2 For Azure, the attached volume count for the compute host group is changed to 0. Only ephemeral/local volumes are used by default. GCP : e2-standard-8; pd-ssd - 100 GB **Note:** Compute nodes run YARN and require storage only for temporary data - this requirement is fulfilled by instance storage, so making the attached volumes count to 0 by default is more cost-efficient.
Gateway	The gateway host group can optionally be used for connecting	0+ AWS : m5.2xlarge; gp2 - 100 GB

Host group	Description	Node configuration
	to the cluster endpoints like Oozie, Beeline etc. This nodegroup does not run any critical services. This nodegroup resides in the same subnet as the rest of the nodegroups. If additional software binaries are required they could be installed using recipes.	Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-8; pd-ssd - 100 GB

Service configurations			
Master host group Cloudera Manager, HDFS, Hive (on Tez), HMS, Yarn RM, Oozie, Hue, DAS, Zookeeper, Livy, Zeppelin and Sqoop	Gateway host group Configurations for the services on the master node	Worker host group Data Node and YARN NodeManager	Compute group YARN NodeManager

Configurations

Note the following:

- There is a Hive Metastore Service (HMS) running in the cluster that talks to the same database instance as the Data Lake in the environment.
- If you use CLI to create the cluster, you can optionally pass an argument to create an external database for the cluster use such as Cloudera Manager, Oozie, Hue, and DAS. This database is by default embedded in the master node external volume. If you specify the external database to be of type HA or NON_HA, the database will be provisioned in the cloud provider. For all these types of databases the lifecycle is still associated with the cluster, so upon deletion of the cluster, the database will also be deleted.
- The HDFS in this cluster is for storing the intermediary processing data. For resiliency, store the data in the cloud object stores.
- For high availability requirements choose the Data Engineering High Availability cluster shape.

Architecture of the Data Engineering HA for AWS cluster

The Data Engineering HA for AWS and Azure cluster shape provides failure resilience for several of the Data Engineering HA services, including Knox, Oozie, HDFS, HS2, Hue, Livy, YARN, and HMS.

Services that do not yet run in HA mode include Cloudera Manager, DAS, and Zeppelin.

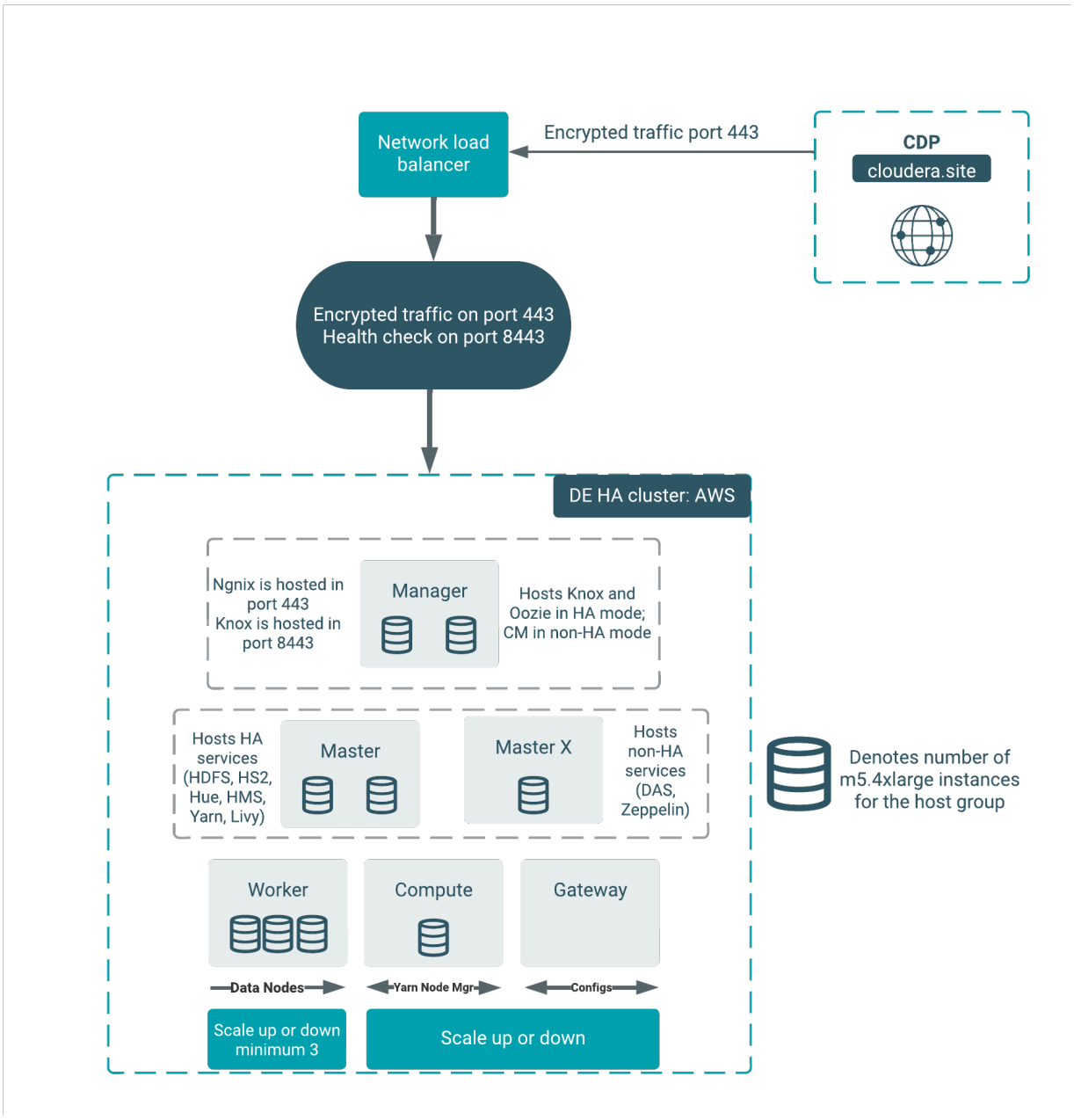

The architecture outlined in the diagram above handles the failure of one node in all of the host groups except for the “masterx” group. See the table below for additional details about the component interactions in failure mode:

Component	Failure	User experience
Knox	One of the Knox services is down	External users will still be able to access all of the UIs, APIs, and JDBC.
Cloudera Manager	The first node in manager host group is down	The cluster operations (such as repair, scaling, and upgrade) will not work.
Cloudera Manager	The second node in the manager host group is down	No impact.
HMS	One of the HMS services is down	No impact.
Hue	One of the Hue services is down in master host group	No impact.

HS2	One of the HS2 services is down in the master host group	External users will still be able to access the Hive service via JDBC. But if Hue was accessing that particular service it will not failover to the other host. The quick fix for Hue is to restart Hue to be able to use Hive functionality.
YARN	One of the YARN services is down	No impact.
HDFS	One of the HDFS services is down	No impact.
Nginx	Nginx in one of the manager hosts is down	Fifty percent of the UI, API, and JDBC calls will be affected. If the entire manager node is down, there is no impact. This is caused by the process of forwarding and health checking that is done by the network load-balancer.
Oozie	One of the Oozie servers is down in the manager host group.	No impact for AWS and Azure as of Cloudera Runtime version 7.2.11. If you create a custom template for DE HA, follow these two rules: **1.** Oozie must be in single hostgroup. **2.** Oozie and Hue must not be in the same hostgroup.

Important: If you are creating a DE HA cluster through the CDP CLI using the create-aws-cluster command, note that there is a CLI parameter to provision the network load-balancer in HA cluster shapes. Make sure to use the [--enable-load-balancer | --no-enable-load-balancer] parameter when provisioning a DE HA cluster via the CLI. For more information see the CDP CLI reference.

Architecture of the Data Engineering HA for Azure cluster

The Data Engineering HA for Azure cluster shape provides failure resilience for several of the Data Engineering HA services, including Knox, Oozie, HDFS, HS2, Hue, Livy, YARN, and HMS.

Services that do not yet run in HA mode include Cloudera Manager, DAS, and Zeppelin.

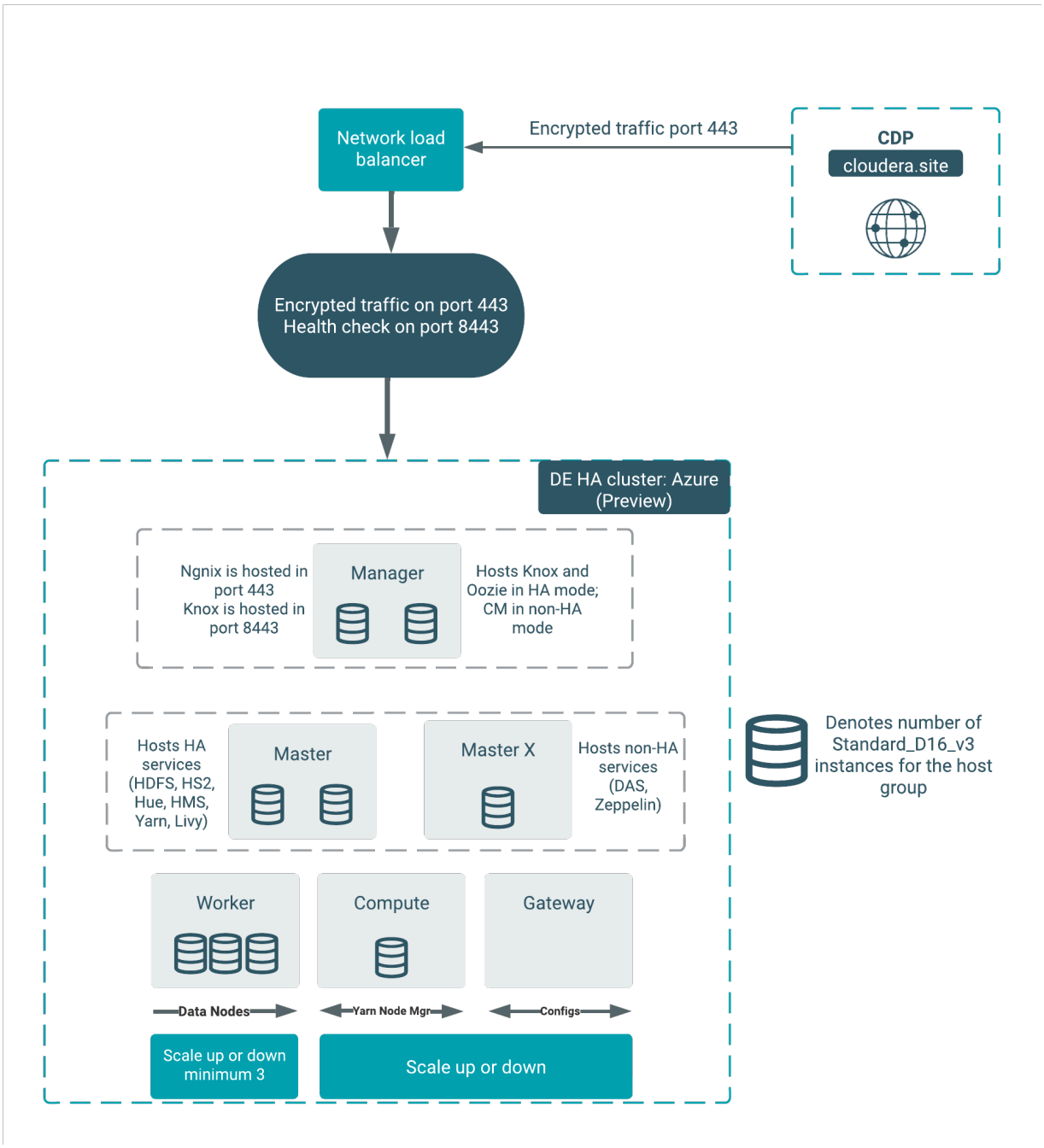

Component	Failure	User experience
Knox	One of the Knox services is down	External users will still be able to access all of the UIs, APIs, and JDBC.
Cloudera Manager	The first node in manager host group is down	The cluster operations (such as repair, scaling, and upgrade) will not work.
Cloudera Manager	The second node in the manager host group is down	No impact.
HMS	One of the HMS services is down	No impact.
Hue	One of the Hue services is down in master host group	No impact.

HS2	One of the HS2 services is down in the master host group	External users will still be able to access the Hive service via JDBC. But if Hue was accessing that particular service it will not failover to the other host. The quick fix for Hue is to restart Hue to be able to use Hive functionality.
YARN	One of the YARN services is down	No impact.
HDFS	One of the HDFS services is down	No impact.
Nginx	Nginx in one of the manager hosts is down	Fifty percent of the UI, API, and JDBC calls will be affected. If the entire manager node is down, there is no impact. This is caused by the process of forwarding and health checking that is done by the network load-balancer.
Oozie	One of the Oozie servers is down in the manager host group.	No impact for AWS and Azure as of Cloudera Runtime version 7.2.11. If you create a custom template for DE HA, follow these two rules: **1.** Oozie must be in single hostgroup. **2.** Oozie and Hue must not be in the same hostgroup.

Important: If you are creating a DE HA cluster through the CDP CLI using the create-azure-cluster command, note that there is a CLI parameter to provision the network load-balancer in HA cluster shapes. Make sure to use the [--enable-load-balancer | --no-enable-load-balancer] parameter when provisioning a DE HA cluster via the CLI. For more information see the CDP CLI reference.

GCP HA (Preview)

Note: HA for Oozie is not yet available in the GCP template.

Custom templates

Any custom DE HA template that you create must be forked from the default templates of the corresponding version. You must create a custom cluster definition for this with the JSON parameter “enableLoadBalancers”: true , using the create-aws/azure/gcp-cluster CLI command parameter --request-template. Support for pre-existing custom cluster definitions will be added in a future release. As with the template, the custom cluster definition must be forked from the default cluster definition. You are allowed to modify the instance types and disks in the custom cluster definition. You must not change the placement of the services like Cloudera Manager, Oozie, and Hue. Currently the custom template is fully supported only via CLI.

The simplest way to change the DE HA definition is to create a custom cluster definition. In the Create Data Hub UI when you click Advanced Options, the default definition is not used fully, which will cause issues in the HA setup.

Related Information

HDFS
Hive
Hue
Oozie
Spark
YARN
Zookeeper

Data Mart clusters

Learn about the default Data Mart and Real Time Data Mart clusters, including cluster definition and template names, included services, and compatible Runtime version.

Data Mart is an MPP SQL database powered by Apache Impala designed to support custom Data Mart applications at big data scale. Impala easily scales to petabytes of data, processes tables with trillions of rows, and allows users to store, browse, query, and explore their data in an interactive way.

Data Mart clusters

The Data Mart template provides a ready to use, fully capable, standalone deployment of Impala. Upon deployment, it can be used as a standalone Data Mart to which users point their BI dashboards using JDBC/ODBC end points. Users can also choose to author SQL queries in Cloudera's web-based SQL query editor, Hue, and run them with Impala providing a delightful end-user focused and interactive SQL/BI experience.

Cluster definition names

- Data Mart for AWS
- Data Mart for Azure
- Data Mart for Google Cloud

Cluster template name

Cloudera - Data Mart: Apache Impala, Hue

Included services

- HDFS
- Hue
- Impala

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17, 7.2.18

Real Time Data Mart clusters

The Real-Time Data Mart template provides a ready-to-use, fully capable, standalone deployment of Impala and Kudu. You can use a Real Time Data Mart cluster as a standalone Data Mart which allows high throughput streaming ingest, supporting updates and deletes as well as inserts. You can immediately query data through BI dashboards using JDBC/ODBC end points. You can choose to author SQL queries in Cloudera's web-based SQL query editor, Hue. Executing queries with Impala, you will enjoy an end-user focused and interactive SQL/BI experience. This template is commonly used for Operational Reporting, Time Series, and other real time analytics use cases.

Cluster definition names

- Real-time Data Mart for AWS
- Real-time Data Mart for Azure
- Real-time Data Mart for Google Cloud

Cluster template name

Cloudera - Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 2

- Yarn

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster definition names

- Real-time Data Mart - Spark3 for AWS
- Real-time Data Mart - Spark3 for Azure
- Real-time Data Mart - Spark3 for Google Cloud

Cluster template name

Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark3

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 3
- Yarn

Compatible Cloudera Runtime versions

7.2.16, 7.2.17, 7.2.18

High availability

Cloudera recommends that you use high availability (HA), and track any services that are not capable of restarting or performing failover in some way.

Impala HA

The Impala nodes offer high availability. The following Impala services are not HA.

- Catalog service
- Statestore service

Kudu HA

Both Kudu Masters and TabletServers offer high availability.

Related Information

HDFS
Hue
Impala
Kudu
Spark
YARN

Operational Database clusters

The Operational Database (OpDB) template is removed from the Cloudera Data Hub. You can access the Cloudera Operational Database instead as a superior product.

The Cloudera Operational Database is a NoSQL database powered by Apache HBase designed to support custom OLTP applications that want to leverage the power of BigData. Apache HBase is a NoSQL, scale-out database that can easily scale to petabytes and stores tables with millions of columns and billions of rows.

Cloudera Operational Database also contains Apache Phoenix which provides a way to use HBase through an SQL interface.

Cloudera recommends you to use the Cloudera Operational Database to create Operational Database clusters.

Related Information

Getting started with Operational Database
Before you create an Operational Database cluster
Creating an Operational Database cluster
Cloudera Operational Database
HDFS
HBase
Knox
Zookeeper
Phoenix

Streams Messaging clusters

Learn about the default Streams Messaging clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streams Messaging provides the following features:

- Advanced messaging and real-time processing on streaming data using Apache Kafka
- Centralized schema management using Schema Registry
- Management and monitoring capabilities powered by Streams Messaging Manager
- Cross-cluster Kafka topic replication using Streams Replication Manger
- Kafka partition rebalancing with Cruise Control

This template sets up a fault-tolerant standalone deployment of Apache Kafka and supporting Cloudera components (Schema Registry, Streams Messaging Manager, Streams Replication Manager and Cruise Control), which can be used for production Kafka workloads in the cloud or as a disaster recovery instance for on-premises. Kafka clusters.

Note:

Streams Messaging clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streams Messaging cluster deployment
- Creating your first Streams Messaging cluster
- Connecting Kafka clients to Cloudera Public Cloud clusters

Cluster definition names

- Streams Messaging Heavy Duty for AWS
- Streams Messaging Light Duty for AWS
- Streams Messaging HA for AWS
- Streams Messaging Heavy Duty for Azure
- Streams Messaging Light Duty for Azure
- Streams Messaging HA for Azure (Technical Preview)
- Streams Messaging Heavy Duty for GCP
- Streams Messaging Light Duty for GCP
- Streams Messaging HA for GCP (Technical Preview)

Cluster template name

- Cloudera - Streams Messaging Heavy Duty
- Cloudera - Streams Messaging Light Duty

- Cloudera - Streams Messaging High Availability

Included services

- Kafka
- Schema Registry
- Streams Messaging Manager
- Streams Replication Manager
- Cruise Control
- Kafka Connect

Compatible Cloudera Runtime version

- 7.1.0 (Preview)
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streams Messaging cluster
Authorizing Access to Flow Management Clusters in ClouderaPublic Cloud
Ingesting Data into ClouderaPublic Cloud
Ingesting Data into Cloudera Public Cloud
Kafka
Schema Registry
Streams Messaging Manager
Streams Replication Manager

Flow Management clusters

Learn about the default Flow Management clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime versions.

Flow Management delivers high-scale data ingestion, transformation, and management to enterprises from any-to-any environment. It addresses key enterprise use cases such as data movement, continuous data ingestion, log data ingestion, and acquisition of all types of streaming data including social, mobile, clickstream, and IoT data.

The Flow Management template includes a no-code data ingestion and management solution powered by Apache NiFi. With NiFi's intuitive graphical interface and 300+ processors, Flow Management enables easy data ingestion and movement between CDP services as well as 3rd party cloud services. NiFi Registry is automatically set up and provides a central place to manage versioned Data Flows.

Note:

Flow Management clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Flow Management cluster deployment
- Creating your first Flow Management cluster
- Security considerations for Flow Management clusters
- Using Apache NiFi to ingest data into Cloudera Public Cloud
- Using NiFi and NiFi Registry

Cluster definition names

- Flow Management Light Duty for AWS
- Flow Management Light Duty for Azure
- Flow Management Light Duty for GCP
- Flow Management Heavy Duty for AWS
- Flow Management Heavy Duty for Azure
- Flow Management Heavy Duty for GCP

Cluster template name

- CDP - Flow Management: Light Duty
- CDP - Flow Management: Heavy Duty

Included services

- NiFi
- NiFI Registry

Compatible Cloudera Runtime versions

- 7.1.0
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Flow Management cluster
Apache NiFi documentation
Apache NiFi Registry documentation

Streaming Analytics clusters

Learn about the default Streaming Analytics clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streaming Analytics offers real-time stream processing and stream analytics with low-latency and high scaling capabilities powered by Apache Flink.

Streaming Analytics templates include Apache Flink that works out of the box in stateless or heavy state environments. Beside Flink, the template includes its supporting services namely YARN, Zookeeper and HDFS. The Heavy Duty template comes preconfigured with RocksDB as state backend, while Light Duty clusters use the default Heap state backend. You can create your streaming application by choosing between Kafka, Kudu, and HBase as datastream connectors.

You can also use SQL to query real-time data with SQL Stream Builder (SSB) in the Streaming Analytics template. By supporting the SSB service in Cloudera Public Cloud, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. SSB is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

Note:

Streaming Analytics clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streaming Analytics cluster deployment
- Creating your first Streaming Analytics cluster
- Analyzing data using Apache Flink
- Querying data using SQL Stream Builder

Cluster definition names

- Streaming Analytics Light Duty for AWS
- Streaming Analytics Light Duty for Azure
- Streaming Analytics Light Duty for GCP
- Streaming Analytics Heavy Duty for AWS
- Streaming Analytics Heavy Duty for Azure
- Streaming Analytics Heavy Duty for GCP

Cluster template name

- 7.2.17 - Streaming Analytics Light Duty
- 7.2.17 - Streaming Analytics Heavy Duty

Included services

- Flink
- SQL Stream Builder
- YARN
- Zookeeper
- HDFS
- Kafka

Important: In the Streaming Analytics cluster templates, Kafka service is included by default to serve as a background service only for the websocket ouput and sampling feature of SQL Stream Builder. The Kafka service in the Streaming Analytics cluster template cannot be used for production, you need to use the Streams Messaging cluster template when Kafka is needed for your deployment.

Compatible Cloudera Runtime version

- 7.2.2

- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streaming Analytics cluster
Flink
YARN
Zookeeper
HDFS

Data Discovery and Exploration clusters

Learn about the default Data Discovery and Exploration clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Discovery and Exploration

Explore and discover data sets ad-hoc. Do relevance-based analytics over unstructured data (logs, images, text, PDFs, etc). Get started with search or log analytics. Make data more accessible to everyone with Data Discovery and Exploration.

Cluster Definition Names

- Data Discovery and Exploration for AWS
- Data Discovery and Exploration for Azure

Cluster Template Name

- Data Discovery and Exploration

Included Services

- Solr
- Spark 2
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Versions

7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster Definition Names

- Data Discovery and Exploration - Spark3 for AWS
- Data Discovery and Exploration - Spark3 for Azure
- Data Discovery and Exploration - Spark3 for Google Cloud

Cluster Template Name

- Data Discovery and Exploration for Spark3

Included Services

- Solr
- Spark 3
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Version

7.2.18

Related Information

Solr
Spark
HDFS
Hue
YARN
Zookeeper

Save a custom cluster definition

The easiest way to obtain and save a custom cluster definition is by providing parameters in the create cluster wizard and then generating a cluster definition on the last page of the wizard.

About this task

Required role: Any user who can create a Cloudera Data Hub cluster

Procedure

1. Log in to the Cloudera web interface.
2. In the Cloudera Management Console, navigate to Environments and then navigate to a specific environment.
3. Click on Create Data Hub.
4. In the cluster wizard, provide all parameters that you need.
5. Once done, click on Save as new definition.
6. Provide a name for the template.
7. Click on Save as cluster template.

   Once registered, your custom cluster definition can be viewed by clicking Environments, then selecting an environment and clicking the Cluster Definitions tab. Note that the cluster definition is only available for use within the environment for which it was created.

What to do next

Now you can use the registered custom cluster definition to create Cloudera Data Hub clusters.

Delete a cluster definition

You can delete a custom cluster definition. Default cluster definitions cannot be deleted.

About this task

Required role: Owner of the cluster definition

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the environment in which you created the definition, then click the Cluster Definitions tab.
3. Select one or more custom cluster definitions.
4. Click .
5. Click Yes to confirm delete.

   After performing these steps, the selected cluster definition(s) will be deleted.

Manage cluster definitions from CLI

You can manage cluster definitions from CLI using cdp datahub commands.

- Register a cluster definition: cdp datahub create-cluster-definition --cluster-definition-name <value> --workload-template <value>
- List all available cluster definitions: cdp datahub list-cluster-definitions
- Describe a specific cluster definition: cdp datahub describe-cluster-definition --cluster-definition-name <value>
- Delete one or more existing cluster definitions: cdp datahub delete-cluster-definitions --cluster-definition-names <value>

Cloudera Data Hub

Cluster Planning

Date published: 2019-12-17
Date modified: 2023-06-27

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cluster planning checklist..... 4

Cluster planning checklist

When creating a production cluster, refer to this checklist in order to ensure that you've performed all prerequisite steps required for cluster deployment.

The following table includes a list of available cluster options and prerequisites required for using each option. Depending on where you are at, you can use this checklist as a starting point for planning your cluster and understanding cloud provider requirements related to specific options.

Option	Description
General Settings	
Tags	You can define tags that will be applied to your cluster-related resources (such as VMs) on your cloud provider account. If you would like to use tags, make sure to prepare a list of tags that meet your organization's requirements as well as the cloud provider's requirements. For more information, refer to Tags.
Image catalog	
Image catalog	When creating a Cloudera Data Hub cluster, you can use default prewarmed images, or you can customize a default image. If you would like to use custom images, you should prepare the images and an image catalog, and then register the image catalog prior to cluster creation.
Network and availability	
Subnet	Your cluster is automatically provisioned within the network selected during environment creation. There are two possibilities as far as subnets are concerned: • If your network includes a single subnet, your cluster is automatically provisioned into that subnet. • During cluster creation you can select the subnet(s) in which your cluster should be provisioned. • If you would like to deploy your Cloudera Data Hub in multiple availability zones, you should select multiple subnets.
Hardware and compute	
EC2 instances	Prior to creating a cluster, determine which instance type you would like to use for each host group. Cloudera Data Hub supports all instance types that have more than 16 GB RAM. If using cluster definitions, default instance types are suggested. Similarly, for custom deployment, there are defaults provided.
Storage options	Prior to creating a cluster, determine the storage type, a number of storage volumes attached per instance, and the volume size that you would like to use for each host group. Supported storage types vary by region, including: • Ephemeral • Magnetic • General purpose • SSD For more information about these options refer to Amazon EC2 Instance Store in AWS documentation. Note that attaching custom disks is not supported.

Option	Description
Encryption	AWS By default, Cloudera Data Hub clusters use the same default key from Amazon's KMS or CMK as the parent environment but you have an option to pass a different CMK during Cloudera Data Hub cluster creation. For more information about configuring encryption on an environment level, refer to Adding a customer managed encryption key to a CDP environment running on AWS. For more information about configuring a CMK for a specific Cloudera Data Hub cluster. refer to EBS Encryption on AWS. Azure By default, local Cloudera Data Hub disks attached to Azure VMs and the PostgreSQL server instance used by the Data Lake and Cloudera Data Hub clusters are encrypted with server-side encryption (SSE) using Platform Managed Keys (PMK), but you can optionally configure SSE with Customer Managed Keys (CMK). This can only be configured on an environment level. For more information, refer to Adding a customer managed encryption key for Azure.
Cluster extensions	
Recipes	You can optionally create and run scripts (called "recipes") that perform specific tasks on all nodes of a given host group. If you would like to use recipes, you should prepare them prior to creating a cluster. For more information, refer to Recipes.
Custom properties	When creating a cluster, it is possible to include a list of custom properties and set them to specific values. If you would like to set custom properties, prior to cluster creation you should prepare a JSON file listing these properties. For more information, refer to Custom Properties.

Cloudera Data Hub

Cluster Templates

Date published: 2019-12-17
Date modified: 2023-06-27

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cluster templates overview....4

Default cluster configurations.... 4

Data Engineering clusters....5
Data Mart clusters....13
Operational Database clusters.... 14
Streams Messaging clusters....15
Flow Management clusters....16
Streaming Analytics clusters.... 18
Data Discovery and Exploration clusters....19

Building a custom cluster template.... 20

Cluster template structure....26

Dynamic cluster template parameters....35

Recipe and cluster template parameters....35
Example: A cluster template with parameters.... 38

Upload a cluster template....41

Manage cluster templates from CLI.... 41

Cluster templates overview

A cluster template is a declarative definition of a cluster that defines cluster topology (cluster host groups and all cluster services and their components running on them).

A Cloudera template is a reusable cluster template in JSON format that can be used for creating multiple Cloudera Data Hub clusters with identical Cloudera Runtime settings. It primarily defines the list of host groups and how components of various Cloudera Runtime services are distributed on these host groups. A cluster template allows you to specify stack, component layout, and configurations to materialize a cluster instance via Cloudera Manager REST API, without having to use the Cloudera Manager install wizard. After you provide the cluster template to Cloudera Data Hub, the host groups in the JSON are mapped to a set of instances when starting the cluster, and the specified services and components are installed on the corresponding nodes.

Note:

A cluster template is not synonymous with a cluster definition, which primarily defines cloud provider settings. Each cluster definition must reference a specific cluster template.

Cloudera Data Hub includes a few default cluster templates and allows you to upload your own cluster templates.

Prior to creating your own cluster templates, we recommend that you review the default cluster templates to check if they meet your requirements. These default cluster templates can be accessed from Shared Resources Templates . To view details of a cluster template, click on its name. For each cluster template, you can access a graphical representation ("list view") and a raw JSON file ("raw view") of all cluster host groups and their components. If you want create a custom template to modify a service's configuration, for example to tune Yarn or Hive, refer to the Cloudera Manager configuration properties for the desired service. You can search these properties by their API Name, which is how they appear in a Cloudera Data Hub template.

If you require a custom cluster template, consider modifying a default template and registering it as a new cluster template.

Default cluster configurations

Cloudera Data Hub includes a set of prescriptive cluster configurations. Each of these default cluster configurations include a cloud-provider specific cluster definition, which primarily defines cloud provider settings. The cluster definition references a cluster template, which defines a number of Cloudera Runtime or Cloudera DataFlow components used for common data analytics and data engineering use cases.

Refer to the topic for each default cluster configuration to view the included services and compatible Runtime versions. These topics include links to documentation that will help you to understand the included components and use the workload cluster.

Many of the cluster components are included in the Cloudera Runtime software distribution. The Streams Messaging, Cloudera Flow Management, and Cloudera Streaming Analytics cluster configurations are part of Cloudera DataFlow for Data Hub and have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for more details.

You can access the default cluster definitions by clicking Environments, then selecting an environment and clicking the Cluster Definitions tab.

You can access the default cluster templates from Shared ResourcesCluster Templates.

To view details of a cluster definition or cluster template, click on its name. For each cluster definition, you can access a raw JSON file. For each cluster template, you can access a graphical representation ("list view") and a raw JSON file ("raw view") of all cluster host groups and their components.

Related Information

Cloudera DataFlow for Data Hub

Cloudera Runtime

Data Engineering clusters

Learn about the default Data Engineering clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Engineering provides a complete data processing solution, powered by Apache Spark and Apache Hive. Spark and Hive enable fast, scalable, fault-tolerant data engineering and analytics over petabytes of data.

Data Engineering cluster definition

This Data Engineering template includes a standalone deployment of Spark and Hive, as well as Apache Oozie for job scheduling and orchestration, Apache Livy for remote job submission, and Hue and Apache Zeppelin for job authoring and interactive analysis.

Cluster definition names

- Data Engineering for AWS
- Data Engineering for Azure
- Data Engineering for Google Cloud
- Data Engineering HA - Spark3 for AWS

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering HA - Spark3 for Azure

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering HA - Spark3 for Google Cloud
- Data Engineering Spark3 for AWS
- Data Engineering Spark3 for Azure
- Data Engineering Spark3 for Google Cloud

Cluster template name

- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: Apache Spark, Apache Hive, Apache Oozie."
- Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: HA: Apache Spark, Apache Hive, Apache Oozie."

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: The "Data Engineering: Apache Spark3" cluster template is deleted. Therefore, the "Data Engineering: Apache Spark3, Apache Hive, Apache Oozie" cluster template can be used instead.

Included services

- HDFS
- Hive
- Hue
- Livy
- Spark 3
- Yarn
- Zeppelin

- ZooKeeper
- Oozie is supported for Spark 3 as of Cloudera Runtime version 7.2.18
- Hive Warehouse Connector is supported as of Cloudera Runtime version 7.2.16.

Compatible Cloudera Runtime version

7.2.16, 7.2.17, 7.2.18

Topology of the Data Engineering cluster

Topology is a set of host groups that are defined in the cluster template and cluster definition used by Data Engineering. Data Engineering uses the following topology:

Host group	Description	Node configuration
Master	The master host group runs the components for managing the cluster resources including Cloudera Manager, Name Node, Resource Manager, as well as other master components such HiveServer2, HMS, Hue etc.	1 For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.4xlarge; gp2 - 100 GB Azure : Standard_D16_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-16; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE, DE Spark3, and DE HA: AWS : m5.4xlarge; gp2 - 100 GB Azure: Standard_D16_v3 GCP : e2-standard-16; pd-ssd - 100 GB
Worker	The worker host group runs the components that are used for executing processing tasks (such as NodeManager) and handling storing data in HDFS such as DataNode.	3 For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.2xlarge; gp2 - 100 GB Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-8; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: AWS: r5d.2xlarge - (gp2/EBS volumes) Azure: Standard_D5_v2 GCP : e2-standard-8; pd-ssd - 100 GB DE HA:

Host group	Description	Node configuration
		AWS: r5d.4xlarge - (gp2/EBS volumes) Azure: Standard_D5_v2 GCP : e2-standard-8; pd-ssd - 100 GB
Compute	The compute host group can optionally be used for running data processing tasks (such as NodeManager). By default the number of compute nodes is set to 1 for proper configurations of YARN containers. This node group can be scaled down to 0 when there are no compute needs. Additionally, if load-based auto-scaling is enabled with minimum count set to 0, the compute nodegroup will be resized to 0 automatically.	0+ For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.2xlarge; gp2 - 100 GB Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-8; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: AWS: r5d.2xlarge - (ephemeral volumes) Azure: Standard_D5_v2 For Azure, the attached volume count for the compute host group is changed to 0. Only ephemeral/local volumes are used by default. GCP : e2-standard-8; pd-ssd - 100 GB DE HA: AWS: r5d.4xlarge - (ephemeral volumes) Azure: Standard_D5_v2 For Azure, the attached volume count for the compute host group is changed to 0. Only ephemeral/local volumes are used by default. GCP : e2-standard-8; pd-ssd - 100 GB **Note:** Compute nodes run YARN and require storage only for temporary data - this requirement is fulfilled by instance storage, so making the attached volumes count to 0 by default is more cost-efficient.

Host group	Description	Node configuration
Gateway	The gateway host group can optionally be used for connecting to the cluster endpoints like Oozie, Beeline etc. This nodegroup does not run any critical services. This nodegroup resides in the same subnet as the rest of the nodegroups. If additional software binaries are required they could be installed using recipes.	0+ AWS : m5.2xlarge; gp2 - 100 GB Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-8; pd-ssd - 100 GB

Service configurations			
Master host group Cloudera Manager, HDFS, Hive (on Tez), HMS, Yarn RM, Oozie, Hue, DAS, Zookeeper, Livy, Zeppelin and Sqoop	Gateway host group Configurations for the services on the master node	Worker host group Data Node and YARN NodeManager	Compute group YARN NodeManager

Configurations

Note the following:

- There is a Hive Metastore Service (HMS) running in the cluster that talks to the same database instance as the Data Lake in the environment.
- If you use CLI to create the cluster, you can optionally pass an argument to create an external database for the cluster use such as Cloudera Manager, Oozie, Hue, and DAS. This database is by default embedded in the master node external volume. If you specify the external database to be of type HA or NON_HA, the database will be provisioned in the cloud provider. For all these types of databases the lifecycle is still associated with the cluster, so upon deletion of the cluster, the database will also be deleted.
- The HDFS in this cluster is for storing the intermediary processing data. For resiliency, store the data in the cloud object stores.
- For high availability requirements choose the Data Engineering High Availability cluster shape.

Architecture of the Data Engineering HA for AWS cluster

The Data Engineering HA for AWS and Azure cluster shape provides failure resilience for several of the Data Engineering HA services, including Knox, Oozie, HDFS, HS2, Hue, Livy, YARN, and HMS.

Services that do not yet run in HA mode include Cloudera Manager, DAS, and Zeppelin.

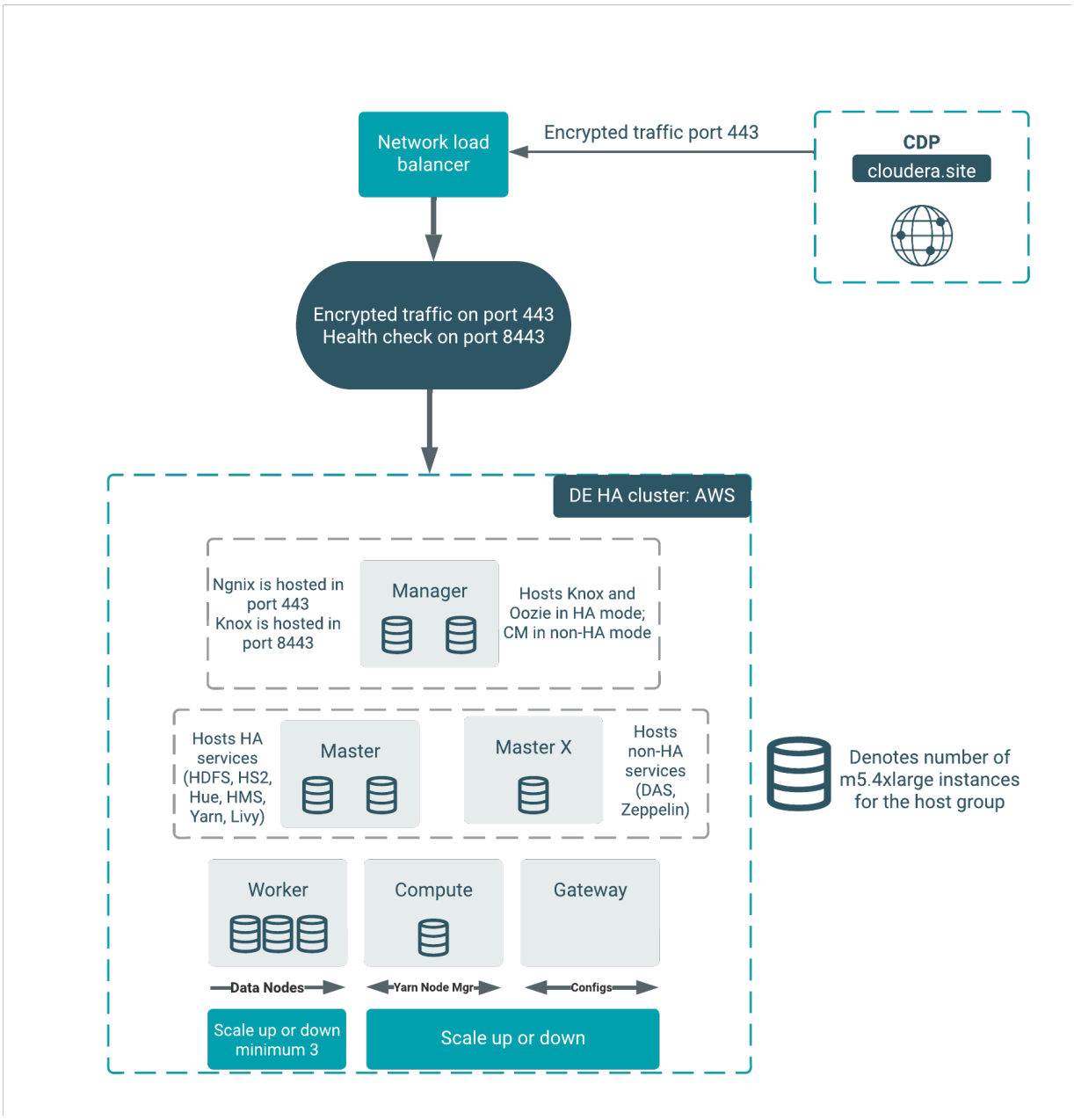

The architecture outlined in the diagram above handles the failure of one node in all of the host groups except for the “masterx” group. See the table below for additional details about the component interactions in failure mode:

Component	Failure	User experience
Knox	One of the Knox services is down	External users will still be able to access all of the UIs, APIs, and JDBC.
Cloudera Manager	The first node in manager host group is down	The cluster operations (such as repair, scaling, and upgrade) will not work.
Cloudera Manager	The second node in the manager host group is down	No impact.
HMS	One of the HMS services is down	No impact.
Hue	One of the Hue services is down in master host group	No impact.

HS2	One of the HS2 services is down in the master host group	External users will still be able to access the Hive service via JDBC. But if Hue was accessing that particular service it will not failover to the other host. The quick fix for Hue is to restart Hue to be able to use Hive functionality.
YARN	One of the YARN services is down	No impact.
HDFS	One of the HDFS services is down	No impact.
Nginx	Nginx in one of the manager hosts is down	Fifty percent of the UI, API, and JDBC calls will be affected. If the entire manager node is down, there is no impact. This is caused by the process of forwarding and health checking that is done by the network load-balancer.
Oozie	One of the Oozie servers is down in the manager host group.	No impact for AWS and Azure as of Cloudera Runtime version 7.2.11. If you create a custom template for DE HA, follow these two rules: **1.** Oozie must be in single hostgroup. **2.** Oozie and Hue must not be in the same hostgroup.

Important: If you are creating a DE HA cluster through the CDP CLI using the create-aws-cluster command, note that there is a CLI parameter to provision the network load-balancer in HA cluster shapes. Make sure to use the [--enable-load-balancer | --no-enable-load-balancer] parameter when provisioning a DE HA cluster via the CLI. For more information see the CDP CLI reference.

Architecture of the Data Engineering HA for Azure cluster

The Data Engineering HA for Azure cluster shape provides failure resilience for several of the Data Engineering HA services, including Knox, Oozie, HDFS, HS2, Hue, Livy, YARN, and HMS.

Services that do not yet run in HA mode include Cloudera Manager, DAS, and Zeppelin.

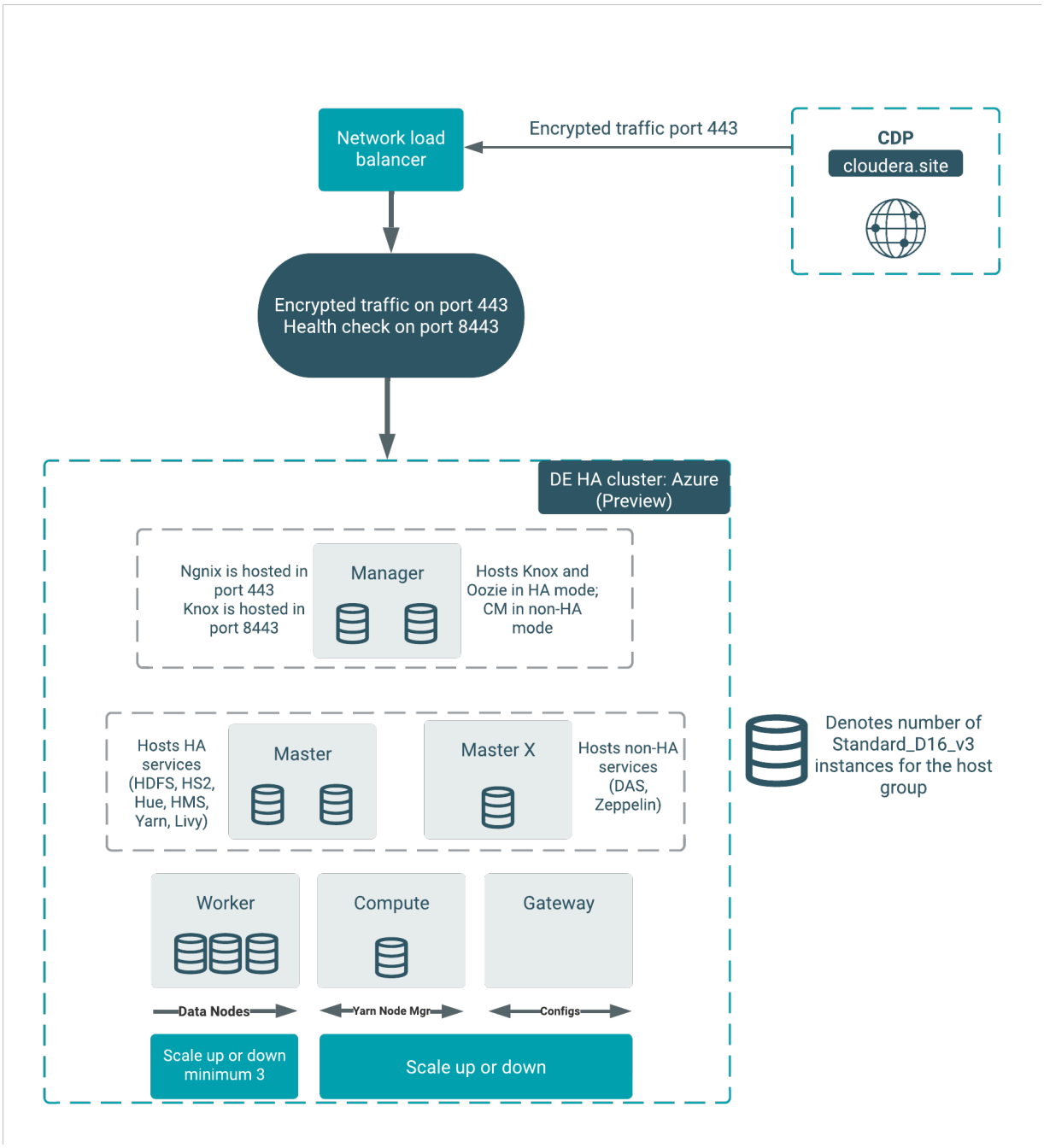

Component	Failure	User experience
Knox	One of the Knox services is down	External users will still be able to access all of the UIs, APIs, and JDBC.
Cloudera Manager	The first node in manager host group is down	The cluster operations (such as repair, scaling, and upgrade) will not work.
Cloudera Manager	The second node in the manager host group is down	No impact.
HMS	One of the HMS services is down	No impact.
Hue	One of the Hue services is down in master host group	No impact.

HS2	One of the HS2 services is down in the master host group	External users will still be able to access the Hive service via JDBC. But if Hue was accessing that particular service it will not failover to the other host. The quick fix for Hue is to restart Hue to be able to use Hive functionality.
YARN	One of the YARN services is down	No impact.
HDFS	One of the HDFS services is down	No impact.
Nginx	Nginx in one of the manager hosts is down	Fifty percent of the UI, API, and JDBC calls will be affected. If the entire manager node is down, there is no impact. This is caused by the process of forwarding and health checking that is done by the network load-balancer.
Oozie	One of the Oozie servers is down in the manager host group.	No impact for AWS and Azure as of Cloudera Runtime version 7.2.11. If you create a custom template for DE HA, follow these two rules: **1.** Oozie must be in single hostgroup. **2.** Oozie and Hue must not be in the same hostgroup.

Important: If you are creating a DE HA cluster through the CDP CLI using the create-azure-cluster command, note that there is a CLI parameter to provision the network load-balancer in HA cluster shapes. Make sure to use the [--enable-load-balancer | --no-enable-load-balancer] parameter when provisioning a DE HA cluster via the CLI. For more information see the CDP CLI reference.

GCP HA (Preview)

Note: HA for Oozie is not yet available in the GCP template.

Custom templates

Any custom DE HA template that you create must be forked from the default templates of the corresponding version. You must create a custom cluster definition for this with the JSON parameter “enableLoadBalancers”: true , using the create-aws/azure/gcp-cluster CLI command parameter --request-template. Support for pre-existing custom cluster definitions will be added in a future release. As with the template, the custom cluster definition must be forked from the default cluster definition. You are allowed to modify the instance types and disks in the custom cluster definition. You must not change the placement of the services like Cloudera Manager, Oozie, and Hue. Currently the custom template is fully supported only via CLI.

The simplest way to change the DE HA definition is to create a custom cluster definition. In the Create Data Hub UI when you click Advanced Options, the default definition is not used fully, which will cause issues in the HA setup.

Related Information

HDFS
Hive
Hue
Oozie
Spark
YARN
Zookeeper

Data Mart clusters

Learn about the default Data Mart and Real Time Data Mart clusters, including cluster definition and template names, included services, and compatible Runtime version.

Data Mart is an MPP SQL database powered by Apache Impala designed to support custom Data Mart applications at big data scale. Impala easily scales to petabytes of data, processes tables with trillions of rows, and allows users to store, browse, query, and explore their data in an interactive way.

Data Mart clusters

The Data Mart template provides a ready to use, fully capable, standalone deployment of Impala. Upon deployment, it can be used as a standalone Data Mart to which users point their BI dashboards using JDBC/ODBC end points. Users can also choose to author SQL queries in Cloudera's web-based SQL query editor, Hue, and run them with Impala providing a delightful end-user focused and interactive SQL/BI experience.

Cluster definition names

- Data Mart for AWS
- Data Mart for Azure
- Data Mart for Google Cloud

Cluster template name

Cloudera - Data Mart: Apache Impala, Hue

Included services

- HDFS
- Hue
- Impala

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17, 7.2.18

Real Time Data Mart clusters

The Real-Time Data Mart template provides a ready-to-use, fully capable, standalone deployment of Impala and Kudu. You can use a Real Time Data Mart cluster as a standalone Data Mart which allows high throughput streaming ingest, supporting updates and deletes as well as inserts. You can immediately query data through BI dashboards using JDBC/ODBC end points. You can choose to author SQL queries in Cloudera's web-based SQL query editor, Hue. Executing queries with Impala, you will enjoy an end-user focused and interactive SQL/BI experience. This template is commonly used for Operational Reporting, Time Series, and other real time analytics use cases.

Cluster definition names

- Real-time Data Mart for AWS
- Real-time Data Mart for Azure
- Real-time Data Mart for Google Cloud

Cluster template name

Cloudera - Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 2

- Yarn

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster definition names

- Real-time Data Mart - Spark3 for AWS
- Real-time Data Mart - Spark3 for Azure
- Real-time Data Mart - Spark3 for Google Cloud

Cluster template name

Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark3

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 3
- Yarn

Compatible Cloudera Runtime versions

7.2.16, 7.2.17, 7.2.18

High availability

Cloudera recommends that you use high availability (HA), and track any services that are not capable of restarting or performing failover in some way.

Impala HA

The Impala nodes offer high availability. The following Impala services are not HA.

- Catalog service
- Statestore service

Kudu HA

Both Kudu Masters and TabletServers offer high availability.

Related Information

HDFS
Hue
Impala
Kudu
Spark
YARN

Operational Database clusters

The Operational Database (OpDB) template is removed from the Cloudera Data Hub. You can access the Cloudera Operational Database instead as a superior product.

The Cloudera Operational Database is a NoSQL database powered by Apache HBase designed to support custom OLTP applications that want to leverage the power of BigData. Apache HBase is a NoSQL, scale-out database that can easily scale to petabytes and stores tables with millions of columns and billions of rows.

Cloudera Operational Database also contains Apache Phoenix which provides a way to use HBase through an SQL interface.

Cloudera recommends you to use the Cloudera Operational Database to create Operational Database clusters.

Related Information

Getting started with Operational Database
Before you create an Operational Database cluster
Creating an Operational Database cluster
Cloudera Operational Database
HDFS
HBase
Knox
Zookeeper
Phoenix

Streams Messaging clusters

Learn about the default Streams Messaging clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streams Messaging provides the following features:

- Advanced messaging and real-time processing on streaming data using Apache Kafka
- Centralized schema management using Schema Registry
- Management and monitoring capabilities powered by Streams Messaging Manager
- Cross-cluster Kafka topic replication using Streams Replication Manger
- Kafka partition rebalancing with Cruise Control

This template sets up a fault-tolerant standalone deployment of Apache Kafka and supporting Cloudera components (Schema Registry, Streams Messaging Manager, Streams Replication Manager and Cruise Control), which can be used for production Kafka workloads in the cloud or as a disaster recovery instance for on-premises. Kafka clusters.

Note:

Streams Messaging clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streams Messaging cluster deployment
- Creating your first Streams Messaging cluster
- Connecting Kafka clients to Cloudera Public Cloud clusters

Cluster definition names

- Streams Messaging Heavy Duty for AWS
- Streams Messaging Light Duty for AWS
- Streams Messaging HA for AWS
- Streams Messaging Heavy Duty for Azure
- Streams Messaging Light Duty for Azure
- Streams Messaging HA for Azure (Technical Preview)
- Streams Messaging Heavy Duty for GCP
- Streams Messaging Light Duty for GCP
- Streams Messaging HA for GCP (Technical Preview)

Cluster template name

- Cloudera - Streams Messaging Heavy Duty
- Cloudera - Streams Messaging Light Duty

- Cloudera - Streams Messaging High Availability

Included services

- Kafka
- Schema Registry
- Streams Messaging Manager
- Streams Replication Manager
- Cruise Control
- Kafka Connect

Compatible Cloudera Runtime version

- 7.1.0 (Preview)
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streams Messaging cluster
Ingesting Data into Cloudera Public Cloud
Kafka
Schema Registry
Streams Messaging Manager
Streams Replication Manager

Flow Management clusters

Learn about the default Flow Management clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime versions.

Flow Management delivers high-scale data ingestion, transformation, and management to enterprises from any-to-any environment. It addresses key enterprise use cases such as data movement, continuous data ingestion, log data ingestion, and acquisition of all types of streaming data including social, mobile, clickstream, and IoT data.

The Flow Management template includes a no-code data ingestion and management solution powered by Apache NiFi. With NiFi's intuitive graphical interface and 300+ processors, Flow Management enables easy data ingestion and movement between CDP services as well as 3rd party cloud services. NiFi Registry is automatically set up and provides a central place to manage versioned Data Flows.

Note:

Flow Management clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Flow Management cluster deployment
- Creating your first Flow Management cluster
- Security considerations for Flow Management clusters
- Using Apache NiFi to ingest data into Cloudera Public Cloud
- Using NiFi and NiFi Registry

Cluster definition names

- Flow Management Light Duty for AWS
- Flow Management Light Duty for Azure
- Flow Management Light Duty for GCP
- Flow Management Heavy Duty for AWS
- Flow Management Heavy Duty for Azure
- Flow Management Heavy Duty for GCP

Cluster template name

- CDP - Flow Management: Light Duty
- CDP - Flow Management: Heavy Duty

Included services

- NiFi
- NiFI Registry

Compatible Cloudera Runtime versions

- 7.1.0
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Flow Management cluster
Apache NiFi documentation
Apache NiFi Registry documentation

Streaming Analytics clusters

Learn about the default Streaming Analytics clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streaming Analytics offers real-time stream processing and stream analytics with low-latency and high scaling capabilities powered by Apache Flink.

Streaming Analytics templates include Apache Flink that works out of the box in stateless or heavy state environments. Beside Flink, the template includes its supporting services namely YARN, Zookeeper and HDFS. The Heavy Duty template comes preconfigured with RocksDB as state backend, while Light Duty clusters use the default Heap state backend. You can create your streaming application by choosing between Kafka, Kudu, and HBase as datastream connectors.

You can also use SQL to query real-time data with SQL Stream Builder (SSB) in the Streaming Analytics template. By supporting the SSB service in Cloudera Public Cloud, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. SSB is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

Note:

Streaming Analytics clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streaming Analytics cluster deployment
- Creating your first Streaming Analytics cluster
- Analyzing data using Apache Flink
- Querying data using SQL Stream Builder

Cluster definition names

- Streaming Analytics Light Duty for AWS
- Streaming Analytics Light Duty for Azure
- Streaming Analytics Light Duty for GCP
- Streaming Analytics Heavy Duty for AWS
- Streaming Analytics Heavy Duty for Azure
- Streaming Analytics Heavy Duty for GCP

Cluster template name

- 7.2.17 - Streaming Analytics Light Duty
- 7.2.17 - Streaming Analytics Heavy Duty

Included services

- Flink
- SQL Stream Builder
- YARN
- Zookeeper
- HDFS
- Kafka

Important: In the Streaming Analytics cluster templates, Kafka service is included by default to serve as a background service only for the websocket ouput and sampling feature of SQL Stream Builder. The Kafka service in the Streaming Analytics cluster template cannot be used for production, you need to use the Streams Messaging cluster template when Kafka is needed for your deployment.

Compatible Cloudera Runtime version

- 7.2.2

- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streaming Analytics cluster
Flink
YARN
Zookeeper
HDFS

Data Discovery and Exploration clusters

Learn about the default Data Discovery and Exploration clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Discovery and Exploration

Explore and discover data sets ad-hoc. Do relevance-based analytics over unstructured data (logs, images, text, PDFs, etc). Get started with search or log analytics. Make data more accessible to everyone with Data Discovery and Exploration.

Cluster Definition Names

- Data Discovery and Exploration for AWS
- Data Discovery and Exploration for Azure

Cluster Template Name

- Data Discovery and Exploration

Included Services

- Solr
- Spark 2
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Versions

7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster Definition Names

- Data Discovery and Exploration - Spark3 for AWS
- Data Discovery and Exploration - Spark3 for Azure
- Data Discovery and Exploration - Spark3 for Google Cloud

Cluster Template Name

- Data Discovery and Exploration for Spark3

Included Services

- Solr
- Spark 3
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Version

7.2.18

Related Information

Solr
Spark
HDFS
Hue
YARN
Zookeeper

Building a custom cluster template

You can build a custom cluster template to modify the cluster Cloudera Runtime services, including the Cloudera Runtime configuration properties and the distribution of Runtime services across host groups. To create a custom template, modify an existing default cluster template and then upload and register the custom template.

About this task

If the default cluster templates are insufficient for the cluster that you want to create, you can build a custom cluster template. The recommended method for building a cluster template is to modify an existing default template, so that the structure of the template is mostly preserved.

Required role: EnvironmentCreator can create a shared resource and then assign users to it. SharedResourceUser or Owner of the shared resource can use the resource. The Owner of the shared resource can delete it.

Procedure

1. Review information on the default cluster configurations to find one that includes services suitable for the type of cluster that you want to create. In general, it is best to use the templates for the current release.
2. To access the existing default cluster templates, click Shared ResourcesCluster Templates.
3. To find the newest template versions, click Platform at the top of the Platform column to sort the templates in descending order.

4. Under the Name column, click the desired default ("Built In") template.

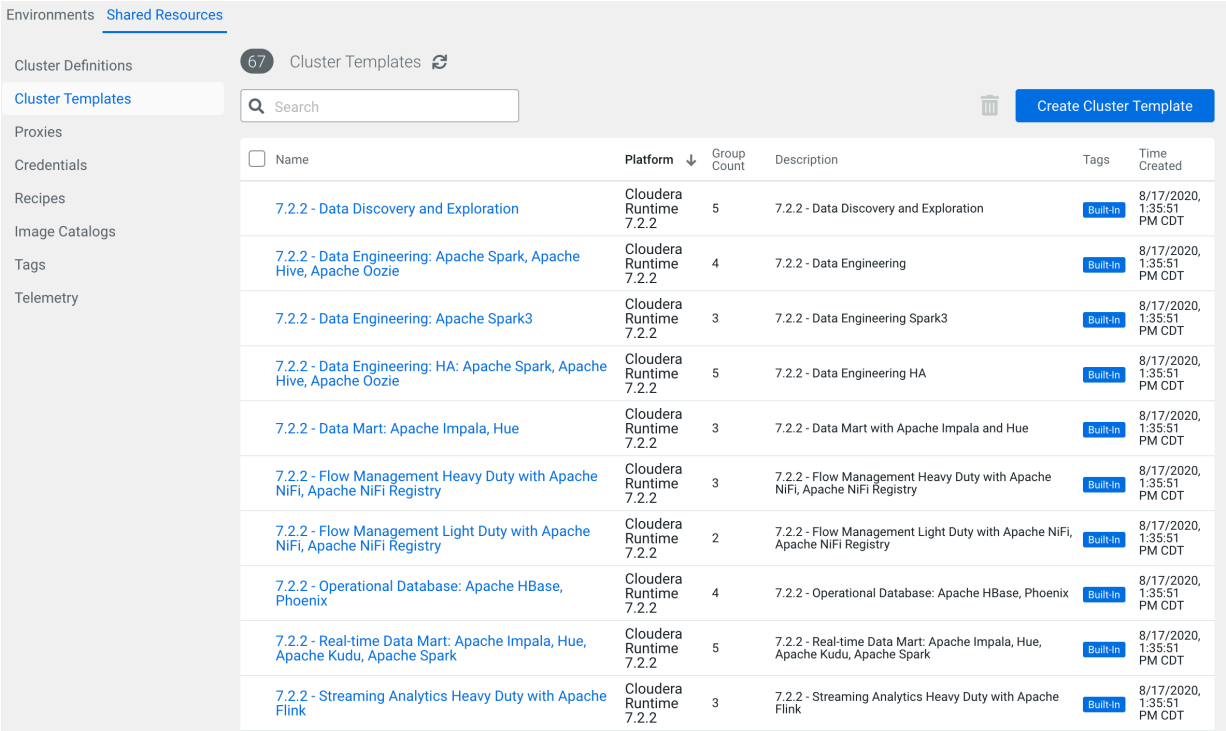

The template opens in LIST view, which shows how the template is structured across host groups.

5. Click RAW VIEW to view the JSON structure.

Environments / List / Cluster Templates / 7.2.2 - Data Engineering: A...

Environments Shared Resources

Cluster Templates
Proxies
Credentials
Recipes
Image Catalogs

7.2.2 - Data Engineering: Apache Spark, Apache Hive, Apache Oozie
7.2.2 - Data Engineering

Delete

LIST VIEW RAW VIEW


```
{
  "cdhVersion": "7.2.2",
  "displayName": "dataengineering",
  "services": [
    {
      "refName": "zookeeper",
      "serviceType": "ZOOKEEPER",
      "serviceConfigs": [
        {
          "name": "service_config_suppression_server_count_validator",
          "value": "true"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "zookeeper-SERVER-BASE",
          "roleType": "SERVER",
          "base": true
        }
      ]
    },
    {
      "refName": "hdfs",
      "serviceType": "HDFS",
      "serviceConfigs": [
        {
          "name": "hdfs_verify_ec_with_topology_enabled",
          "value": false
        },
        {
          "name": "core_site_safety_valve",
          "value": "<property><name>fs.s3a.buffer.dir</name><value>${env.LOCAL_DIRS:-${hadoop.tmp.dir}}/s3a</value></property><property>
<name>fs.s3a.committer.name</name><value>directory</value></property>"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "hdfs-NAMENODE-BASE",
          "roleType": "NAMENODE",
          "base": true,
          "configs": [
            {
              "name": "role_config_suppression_fs_trash_interval_minimum_validator",
              "value": "true"
            },
            {
              "name": "role_config_suppression_namenode_java_heapsize_minimum_validator",
              "value": "true"
            },
            {
              "name": "fs_trash_interval",
              "value": "0"
            },
            {
              "name": "fs_trash_checkpoint_interval",
              "value": "0"
            },
            {
              "name": "erasure_coding_default_policy",
              "value": " "
            }
          ]
        },
        {
          "refName": "hdfs-SECONDARYNAMENODE-BASE",
          "roleType": "SECONDARYNAMENODE",
          "base": true
        },
        {
          "refName": "hdfs-DATANODE-BASE",
          "roleType": "DATANODE",
          "base": true
        },
        {
          "refName": "hdfs-BALANCER-BASE",
          "roleType": "BALANCER",
          "base": true
        },
        {
          "refName": "hdfs-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true,
          "configs": [
            {
              "name": "dfs_client_use_trash",
              "value": false
            },
            {
              "name": "role_config_suppression_hdfs_trash_disabled_validator",
              "value": "true"
            },
            {
              "name": "hdfs_client_env_safety_valve",
              "value": "HADOOP_OPTS=\"-Dorg.wildfly.openssl.path=/usr/lib64 ${HADOOP_OPTS}\""
            }
          ]
        }
      ]
    },
    {
      "refName": "yarn",
      "serviceType": "YARN",
      "serviceConfigs": [
        {
          "name": "yarn_admin_acl",
          "value": "yarn,hive,hdfs,mapred"
        },
        {
          "name": "yarn_service_mapred_safety_valve",
          "value": "<property><name>mapreduce.fileoutputcommitter.algorithm.version</name><value>1</value></property><property>
<name>mapreduce.input.fileinputformat.list-status.num-threads</name><value>100</value></property>"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "yarn-RESOURCEMANAGER-BASE",
          "roleType": "RESOURCEMANAGER",
          "base": true,
          "configs": [
            {
              "name": "resourcemanager_config_safety_valve",
              "value": "<property><name>yarn.scheduler.configuration.store.class</name><value>zk</value></property>"
            },
            {
              "name": "yarn_resourcemanager_scheduler_class",
              "value": "org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler"
            },
            {
              "name": "yarn_scheduler_capacity_resource_calculator",
              "value": "org.apache.hadoop.yarn.util.resource.DefaultResourceCalculator"
            },
            {
              "name": "resourcemanager_capacity_scheduler_configuration",
              "value": "<configuration><property><name>yarn.scheduler.capacity.root.queues</name><value>default</value></property><property>
<name>yarn.scheduler.capacity.root.capacity</name><value>100</value></property><property><name>yarn.scheduler.capacity.root.default.capacity</name>
<value>100</value></property><property><name>yarn.scheduler.capacity.root.acl_submit_applications</name><value> </value></property><property>
<name>yarn.scheduler.capacity.root.acl_administer_queue</name><value> </value></property><property>
<name>yarn.scheduler.capacity.root.default.acl_submit_applications</name><value>*</value></property><property>
<name>yarn.scheduler.capacity.root.default.minimum-user-limit-percent</name><value>100</value></property><property>
<name>yarn.scheduler.capacity.maximum-am-resource-percent</name><value>0.33</value></property><property><name>yarn.scheduler.capacity.node-
locality-delay</name><value>0</value></property><property><name>yarn.scheduler.capacity.schedule-asynchronously.maximum-threads</name>
<value>1</value></property><property><name>yarn.scheduler.capacity.schedule-asynchronously.scheduling-interval-ms</name><value>10</value>
</property></configuration>"
            }
          ]
        },
        {
          "refName": "yarn-NODEMANAGER-WORKER",
          "roleType": "NODEMANAGER",
          "base": false
        },
        {
          "refName": "yarn-NODEMANAGER-COMPUTE",
          "roleType": "NODEMANAGER",
          "base": false
        },
        {
          "refName": "yarn-JOBHISTORY-BASE",
          "roleType": "JOBHISTORY",
          "base": true
        },
        {
          "refName": "yarn-GATEWAY-BASE",
```

6. Select all of the JSON code, copy it, and paste it to a suitable code editor. Standard text editors are not recommended.
7. If you are using a code editor such as Microsoft Visual Studio Code, you can use built-in tools to validate the JSON before you proceed. Optionally, you can save the file as-is (without having made any changes), and upload it using the instructions in the topic *Upload a cluster template*. To verify that the template JSON is functional, you can create a test cluster by selecting the template that you just registered, and see if the cluster successfully deploys.
8. When you are satisfied that you are working with a clean template, you can begin to modify the template. Return to the JSON template file in your code editor.

   Each default template consists of two main sections: the services section and the hostTemplates section. The serv ices section includes the components that make up the cluster. This is where you can add or remove services, as well as modify service configuration properties. If you want to modify a service's configuration, for example to tune Yarn or Hive, refer to the Cloudera Manager configuration properties for the desired service. You can search these properties by their API Name, which is how they appear in a Cloudera Data Hub template.

   For example, in a Data Engineering template you might want to adjust the amount of physical memory allocated for containers by configuring the yarn.nodemanager.resource.memory-mb property. If you want to configure this property to 80% of the total system RAM, for a 256 GB machine this would look like:

```
{
      "refName": "yarn",
      "serviceType": "YARN",
      "serviceConfigs": [
        {
          "name": "yarn_admin_acl",
          "value": "yarn,hive,hdfs,mapred"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "yarn-RESOURCEMANAGER-BASE",
          "roleType": "RESOURCEMANAGER",
          "base": true,
          "configs": [
            {
              "name": "yarn.nodemanager.resource.memory-mb",
              "value": 2052096
            },
```

   If you want to add a service in a template, the simplest method is to find the service in the RAW VIEW of another default template and copy it into your JSON.

   For example, if you want to add Sqoop to the services in a template, copy it from the Data Engineering or Data Engineering HA template into the services section of another template:

```
{
      "refName": "sqoop",
      "serviceType": "SQOOP_CLIENT",
      "roleConfigGroups": [
        {
          "refName": "sqoop-SQOOP_CLIENT-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "configs": [],
          "base": true
        }
      ]
```

```
    },
```

The hostTemplates section of the JSON file describes the nodes by their type and the services on the node. This section also includes a cardinality parameter, which you can set to increase or decrease the quantity of that specific node type.

For example, say that you want to create a new node type called "ZKserver" that runs a single service, Zookeeper. Assuming that Zookeeper is already a service defined in the services section of the template, you can move down to the hostTemplate section. In the master node section of this Data Engineering template, you can see that Zookeeper is already defined in the "master" node section with the string "zookeeper-SERVER-BASE":

```
"hostTemplates": [
    {
      "refName": "master",
      "cardinality": 1,
      "roleConfigGroupsRefNames": [
        "hdfs-BALANCER-BASE",
        "hdfs-NAMENODE-BASE",
        "hdfs-SECONDARYNAMENODE-BASE",
        "hdfs-GATEWAY-BASE",
        "hms-GATEWAY-BASE",
        "hms-HIVEMETASTORE-BASE",
        "hive_on_tez-HIVESERVER2-BASE",
        "hive_on_tez-GATEWAY-BASE",
        "hue-HUE_LOAD_BALANCER-BASE",
        "hue-HUE_SERVER-BASE",
        "tez-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "spark_on_yarn-SPARK_YARN_HISTORY_SERVER-BASE",
        "livy-LIVY_SERVER-BASE",
        "zeppelin-ZEPPELIN_SERVER-BASE",
        "oozie-OOZIE_SERVER-BASE",
        "sqoop-SQOOP_CLIENT-GATEWAY-BASE",
        "yarn-JOBHISTORY-BASE",
        "yarn-RESOURCEMANAGER-BASE",
        "zookeeper-SERVER-BASE",
        "das-DAS_WEBAPP",
        "das-DAS_EVENT_PROCESSOR",
        "yarn-QUEUEMANAGER_WEBAPP-BASE",
        "yarn-QUEUEMANAGER_STORE-BASE",
        "yarn-GATEWAY-BASE"
      ]
```

To create our new ZKserver node, you can copy the standard node format and modify it for your purpose:

```
{
      "refName": "ZKserver",
      "cardinality": 1,
      "roleConfigGroupsRefNames": [
        "zookeeper-SERVER-BASE"
      ]
    }
```

If you also want to include dynamic parameters in your custom template, see the documentation for Dynamic cluster templates. During the cluster creation phase, dynamic parameters pick up the parameter values that you provided in the Cloudera Data Hub cluster wizard. See the *Custom Properties* documentation for a list of properties that can be dynamically replaced. You might want to use dynamic parameters when you regularly provision clusters using a specific template, but want to change a few of the property values each time you provision a new cluster.

9. When you have finished modifying the template, validate the JSON in your code editor and save the template.

What to do next

Upload the JSON file and register the template following the instructions in the *Upload a cluster template* topic. Then, when you navigate to the Data Hubs page and select Create Data Hub, be sure to select the Custom radio button underneath environment selection. Here you can provision a Cloudera Data Hub cluster using the custom template that you registered. Select your custom template from the drop-down menu before you configure any advanced options.

Related Information

Upload a cluster template

Cluster template structure

The template below is an example of a 7.2.2 Data Engineering HA cluster template in JSON view.

Many of the default templates contain properties for services that aren't displayed when you select a default template in the user interface, for example Hive on Tez or Sqoop in the default Data Engineering HA template below. If you are creating a custom template and you need to know more about the properties and services in a template, refer to the Cloudera Manager configuration properties for each Cloudera Runtime service. These properties can be helpful if you are trying to tune Yarn or Hive, for example. You can also copy a service like Sqoop from the JSON of one template to another.

```
{
  "cdhVersion": "7.2.2",
  "displayName": "dataengineering ha",
  "services": [
    {
      "refName": "zookeeper",
      "serviceType": "ZOOKEEPER",
      "roleConfigGroups": [
        {
          "refName": "zookeeper-SERVER-BASE",
          "roleType": "SERVER",
          "base": true
        }
      ]
    },
    {
      "refName": "hdfs",
      "serviceType": "HDFS",
      "serviceConfigs": [
        {
          "name": "zookeeper_service",
          "ref": "zookeeper"
        },
        {
          "name": "hdfs_verify_ec_with_topology_enabled",
          "value": false
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "hdfs-NAMENODE-BASE",
          "roleType": "NAMENODE",
          "base": true,
          "configs": [
            {
              "name": "fs_trash_interval",
              "value": "0"
            },
            {
```

```
              "name": "fs_trash_checkpoint_interval",
              "value": "0"
            },
            {
              "name": "erasure_coding_default_policy",
              "value": " "
            }
          ]
        },
        {
          "refName": "hdfs-FAILOVERCONTROLLER-BASE",
          "roleType": "FAILOVERCONTROLLER",
          "base": true
        },
        {
          "refName": "hdfs-JOURNALNODE-BASE",
          "roleType": "JOURNALNODE",
          "base": true
        },
        {
          "refName": "hdfs-DATANODE-BASE",
          "roleType": "DATANODE",
          "base": true
        },
        {
          "refName": "hdfs-SECONDARYNAMENODE-BASE",
          "roleType": "SECONDARYNAMENODE",
          "configs": [
            {
              "name": "fs_checkpoint_dir_list",
              "value": "/should_not_be_required_in_HA_setup"
            }
          ],
          "base": true
        },
        {
          "refName": "hdfs-BALANCER-BASE",
          "roleType": "BALANCER",
          "base": true
        },
        {
          "refName": "hdfs-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true,
          "configs": [
            {
              "name": "dfs_client_use_trash",
              "value": false
            },
            {
              "name": "hdfs_client_env_safety_valve",
              "value": "HADOOP_OPTS=\"-Dorg.wildfly.openssl.path=/usr/lib64
${HADOOP_OPTS}\""
            }
          ]
        }
      ]
    },
    {
      "refName": "hms",
      "serviceType": "HIVE",
      "displayName": "Hive Metastore",
      "roleConfigGroups": [
        {
```

```
          "refName": "hms-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true
        },
        {
          "refName": "hms-HIVEMETASTORE-BASE",
          "roleType": "HIVEMETASTORE",
          "base": true
        }
      ]
    },
    {
      "refName": "hive",
      "serviceType": "HIVE_ON_TEZ",
      "displayName": "Hive",
      "serviceConfigs": [
        {
          "name": "tez_container_size",
          "value": "4096"
        },
        {
          "name": "tez_auto_reducer_parallelism",
          "value": "false"
        },
        {
          "name": "hive_service_config_safety_valve",
          "value": "<property><name>fs.s3a.ssl.channel.mode</name><value>op
enssl</value></property><property><name>hive.txn.acid.dir.cache.duration</na
me><value>0</value></property>"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "hive-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true
        },
        {
          "refName": "hive-HIVESERVER2-BASE",
          "roleType": "HIVESERVER2",
          "base": true,
          "configs": [
            {
              "name": "hive_server2_transport_mode",
              "value": "http"
            },
            {
              "name": "hiveserver2_mv_files_thread",
              "value": 20
            },
            {
              "name": "hiveserver2_load_dynamic_partitions_thread_count",
              "value": 20
            }
          ]
        }
      ]
    },
    {
      "refName": "hue",
      "serviceType": "HUE",
      "serviceConfigs": [
        {
          "name": "hue_service_safety_valve",
```

```
          "value": "[desktop]\napp_blacklist=spark,zookeeper,hbase,impala
,search,sqoop,security,pig"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "hue-HUE_SERVER-BASE",
          "roleType": "HUE_SERVER",
          "base": true
        },
        {
          "refName": "hue-HUE_LOAD_BALANCER-BASE",
          "roleType": "HUE_LOAD_BALANCER",
          "base": true
        }
      ]
    },
    {
      "refName": "livy",
      "serviceType": "LIVY",
      "roleConfigGroups": [
        {
          "refName": "livy-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true
        },
        {
          "refName": "livy-LIVY_SERVER-BASE",
          "roleType": "LIVY_SERVER",
          "base": true
        }
      ]
    },
    {
      "refName": "oozie",
      "serviceType": "OOZIE",
      "roleConfigGroups": [
        {
          "refName": "oozie-OOZIE_SERVER-BASE",
          "roleType": "OOZIE_SERVER",
          "configs": [
            {
              "name": "oozie_config_safety_valve",
              "value": "<property><name>oozie.service.HadoopAccessorService
.nameNode.whitelist</name><value></value></property>"
            }
          ],
          "base": true
        }
      ]
    },
    {
      "refName": "sqoop",
      "serviceType": "SQOOP_CLIENT",
      "roleConfigGroups": [
        {
          "refName": "sqoop-SQOOP_CLIENT-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "configs": [],
          "base": true
        }
      ]
    },
    {
```

```
      "refName": "yarn",
      "serviceType": "YARN",
      "serviceConfigs": [
        {
          "name": "yarn_admin_acl",
          "value": "yarn,hive,hdfs,mapred"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "yarn-RESOURCEMANAGER-BASE",
          "roleType": "RESOURCEMANAGER",
          "base": true,
          "configs": [
            {
              "name": "resourcemanager_config_safety_valve",
              "value": "<property><name>yarn.scheduler.configuration.stor
e.class</name><value>zk</value></property>"
            },
            {
              "name": "yarn_resourcemanager_scheduler_class",
              "value": "org.apache.hadoop.yarn.server.resourcemanager.sch
eduler.capacity.CapacityScheduler"
            },
            {
              "name": "yarn_scheduler_capacity_resource_calculator",
              "value": "org.apache.hadoop.yarn.util.resource.DefaultResourc
eCalculator"
            },
            {
              "name": "resourcemanager_capacity_scheduler_configuration",
              "value": "<configuration><property><name>yarn.scheduler.capaci
ty.root.queues</name><value>default</value></property><property><name>yarn.s
cheduler.capacity.root.capacity</name><value>100</value></property><property
><name>yarn.scheduler.capacity.root.default.capacity</name><value>100</value
></property><property><name>yarn.scheduler.capacity.root.acl_submit_applicat
ions</name><value> </value></property><property><name>yarn.scheduler.capacit
y.root.acl_administer_queue</name><value> </value></property><property><name
>yarn.scheduler.capacity.root.default.acl_submit_applications</name><value>*
</value></property><property><name>yarn.scheduler.capacity.root.default.mini
mum-user-limit-percent</name><value>100</value></property><property><name>ya
rn.scheduler.capacity.maximum-am-resource-percent</name><value>0.33</value><
/property><property><name>yarn.scheduler.capacity.node-locality-delay</name>
<value>0</value></property><property><name>yarn.scheduler.capacity.schedule-
asynchronously.maximum-threads</name><value>1</value></property><property><n
ame>yarn.scheduler.capacity.schedule-asynchronously.scheduling-interval-ms</
name><value>10</value></property></configuration>"
            }
          ]
        },
        {
          "refName": "yarn-NODEMANAGER-WORKER",
          "roleType": "NODEMANAGER",
          "base": false
        },
        {
          "refName": "yarn-NODEMANAGER-COMPUTE",
          "roleType": "NODEMANAGER",
          "base": false
        },
        {
          "refName": "yarn-JOBHISTORY-BASE",
          "roleType": "JOBHISTORY",
          "base": true
```

```
        },
        {
          "refName": "yarn-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true,
          "configs": [
            {
              "name": "mapreduce_map_memory_mb",
              "value": 4096
            },
            {
              "name": "mapreduce_reduce_memory_mb",
              "value": 4096
            },
            {
              "name": "mapreduce_client_env_safety_valve",
              "value": "HADOOP_OPTS=\"-Dorg.wildfly.openssl.path=/usr/lib64
 ${HADOOP_OPTS}\""
            }
          ]
        }
      ]
    },
    {
      "refName": "spark_on_yarn",
      "serviceType": "SPARK_ON_YARN",
      "roleConfigGroups": [
        {
          "refName": "spark_on_yarn-SPARK_YARN_HISTORY_SERVER-BASE",
          "roleType": "SPARK_YARN_HISTORY_SERVER",
          "base": true
        },
        {
          "refName": "spark_on_yarn-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true,
          "configs": [
            {
              "name": "spark-conf/spark-defaults.conf_client_config_safety_v
alve",
              "value": "spark.hadoop.fs.s3a.ssl.channel.mode=openssl"
            }
          ]
        }
      ]
    },
    {
      "refName": "tez",
      "serviceType": "TEZ",
      "serviceConfigs": [
        {
          "name": "yarn_service",
          "ref": "yarn"
        },
        {
          "name": "tez.am.container.reuse.non-local-fallback.enabled",
          "value": "true"
        },
        {
          "name": "tez.am.container.reuse.locality.delay-allocation-millis",
          "value": "0"
        },
        {
          "name": "tez.am.launch.cmd-opts",
```

```
          "value": "-XX:+PrintGCDetails -verbose:gc -XX:+UseNUMA -XX:+UseG1G
C -XX:+ResizeTLAB"
        },
        {
          "name": "tez.task.launch.cmd-opts",
          "value": "-XX:+PrintGCDetails -verbose:gc -XX:+UseNUMA -XX:+UseG
1GC -XX:+ResizeTLAB"
        },
        {
          "name": "tez.grouping.split-waves",
          "value": 1.4
        },
        {
          "name": "tez.grouping.min-size",
          "value": 268435456
        },
        {
          "name": "tez.grouping.max-size",
          "value": 268435456
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "tez-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true
        }
      ]
    },
    {
      "refName": "das",
      "serviceType": "DAS",
      "roleConfigGroups": [
        {
          "refName": "das-DAS_WEBAPP",
          "roleType": "DAS_WEBAPP",
          "base": true,
          "configs": [
            {
              "name": "data_analytics_studio_admin_users",
              "value": "*"
            }
          ]
        },
        {
          "refName": "das-DAS_EVENT_PROCESSOR",
          "roleType": "DAS_EVENT_PROCESSOR",
          "base": true
        }
      ]
    },
    {
      "refName": "knox",
      "roleConfigGroups": [
        {
          "base": true,
          "refName": "knox-KNOX_GATEWAY-BASE",
          "roleType": "KNOX_GATEWAY"
        }
      ],
      "serviceType": "KNOX"
    },
    {
      "refName": "zeppelin",
```

```
      "serviceType": "ZEPPELIN",
      "serviceConfigs": [
        {
          "name": "yarn_service",
          "ref": "yarn"
        },
        {
          "name": "hdfs_service",
          "ref": "hdfs"
        },
        {
          "name": "spark_on_yarn_service",
          "ref": "spark_on_yarn"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "zeppelin-ZEPPELIN_SERVER-BASE",
          "roleType": "ZEPPELIN_SERVER",
          "base": true
        }
      ]
    },
    {
      "refName": "queuemanager",
      "serviceType": "QUEUEMANAGER",
      "serviceConfigs": [
        {
          "name": "kerberos.auth.enabled",
          "value": "true"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "yarn-QUEUEMANAGER_WEBAPP-BASE",
          "roleType": "QUEUEMANAGER_WEBAPP",
          "base": true
        },
        {
          "refName": "yarn-QUEUEMANAGER_STORE-BASE",
          "roleType": "QUEUEMANAGER_STORE",
          "base": true
        }
      ]
    }
  ],
  "hostTemplates": [
    {
      "refName": "gateway",
      "cardinality": 0,
      "roleConfigGroupsRefNames": [
        "hdfs-GATEWAY-BASE",
        "hive-GATEWAY-BASE",
        "hms-GATEWAY-BASE",
        "livy-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "tez-GATEWAY-BASE",
        "yarn-GATEWAY-BASE"
      ]
    },
    {
      "refName": "master",
      "cardinality": 2,
      "roleConfigGroupsRefNames": [
```

```
        "hdfs-FAILOVERCONTROLLER-BASE",
        "hdfs-NAMENODE-BASE",
        "hdfs-GATEWAY-BASE",
        "hive-GATEWAY-BASE",
        "hive-HIVESERVER2-BASE",
        "hms-GATEWAY-BASE",
        "hms-HIVEMETASTORE-BASE",
        "hue-HUE_LOAD_BALANCER-BASE",
        "hue-HUE_SERVER-BASE",
        "knox-KNOX_GATEWAY-BASE",
        "livy-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "tez-GATEWAY-BASE",
        "yarn-RESOURCEMANAGER-BASE",
        "hdfs-JOURNALNODE-BASE",
        "zookeeper-SERVER-BASE",
        "yarn-GATEWAY-BASE"
      ]
    },
    {
      "refName": "manager",
      "cardinality": 1,
      "roleConfigGroupsRefNames": [
        "hdfs-BALANCER-BASE",
        "hdfs-GATEWAY-BASE",
        "hive-GATEWAY-BASE",
        "hms-GATEWAY-BASE",
        "livy-GATEWAY-BASE",
        "livy-LIVY_SERVER-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "spark_on_yarn-SPARK_YARN_HISTORY_SERVER-BASE",
        "tez-GATEWAY-BASE",
        "yarn-JOBHISTORY-BASE",
        "oozie-OOZIE_SERVER-BASE",
        "sqoop-SQOOP_CLIENT-GATEWAY-BASE",
        "zeppelin-ZEPPELIN_SERVER-BASE",
        "das-DAS_EVENT_PROCESSOR",
        "das-DAS_WEBAPP",
        "knox-KNOX_GATEWAY-BASE",
        "hdfs-JOURNALNODE-BASE",
        "zookeeper-SERVER-BASE",
        "yarn-QUEUEMANAGER_WEBAPP-BASE",
        "yarn-QUEUEMANAGER_STORE-BASE",
        "yarn-GATEWAY-BASE"
      ]
    },
    {
      "refName": "worker",
      "cardinality": 3,
      "roleConfigGroupsRefNames": [
        "hdfs-DATANODE-BASE",
        "hdfs-GATEWAY-BASE",
        "hive-GATEWAY-BASE",
        "hms-GATEWAY-BASE",
        "livy-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "tez-GATEWAY-BASE",
        "yarn-NODEMANAGER-WORKER",
        "yarn-GATEWAY-BASE"
      ]
    },
    {
      "refName": "compute",
      "cardinality": 0,
```

```
      "roleConfigGroupsRefNames": [
        "hdfs-GATEWAY-BASE",
        "hive-GATEWAY-BASE",
        "hms-GATEWAY-BASE",
        "livy-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "tez-GATEWAY-BASE",
        "yarn-NODEMANAGER-COMPUTE",
        "yarn-GATEWAY-BASE"
      ]
    }
  ]
}
```

Dynamic cluster template parameters

You can add dynamic parameters to a cluster template. The values of the variables specified in the cluster template are dynamically replaced during the cluster creation phase, picking up the parameter values that you provided in the Cloudera Data Hub cluster wizard. Cloudera Data Hub supports the "mustache" formatting syntax for dynamic properties.

Production cluster configurations typically include certain configuration parameters, forcing you to create multiple versions of the same cluster template to handle different component configurations for these external systems. Dynamic parameters in a cluster template solve this problem by offering the ability to manage external sources outside of your cluster template. They use the cluster template as a blueprint, and Cloudera Data Hub injects the actual configurations into your cluster template. This simplifies the reuse of cluster configurations for external sources and simplifies the cluster templates themselves.

Note:

You cannot use functions in the cluster template file; only variable injection is supported.

The following provides a list of properties that can be used in a dynamic cluster template and an example:

Recipe and cluster template parameters

The following supported parameters can be specified as variables/dynamic parameters in recipes or cluster templates by using mustache formatting with "{{{ }}}" syntax.

Note: Using variable parameters is not supported for FreeIPA recipes.

The parameter keys listed below follow the following general conventions:

- { } indicates that the parameter key has multiple supported values, which are provided in this documentation. For example {fileSystemType} can be one of the following: s3, adls_gen_2, or wasb.
- [index] indicates that the parameter includes an index value for example sharedService.datalakeComponents.[index] can be "sharedService.datalakeComponents.[0]", "sharedService.datalakeComponents.[1]", and so on. There is no easy way to find out what the index will be, but you may still be able to use these parameters (for example by creating a condition to filter them).

For information on how to set these parameters dynamically in a cluster template, refer to Setting custom properties.

Custom properties

Any custom property specified in the cluster template can be used as a recipe parameter. Refer to Custom properties documentation.

General

The general parameter group includes parameters related to general cluster configuration.

Description	Example key	Example value
Name of stack	general.stackName	teststack
UUID of cluster	general.uuid	9aab7fdb-8940-454b-bc0a-62f04bce6519
Cloudera Manager user name	general.cmUserName	
Cloudera Manager password	general.cmPassword	
Cloudera Manager IP	general.clusterManagerIp	127.0.0.1
Number of nodes	general.nodeCount	5
FQDN of primary gateway instance	general.primaryGatewayInstanceDiscoveryFQDN	ip-10-0-88-28.example.com
Number indicating the Kafka replication factor (3 or 1)	general.kafkaReplicationFactor	1

Blueprint

The blueprint parameter group includes parameters related to cluster template configuration.

Parameter key	Description	Example key	Example value
blueprint.blueprintText	Blueprint text in JSON format	blueprint.blueprintText	
blueprint.version	Version of blueprint	blueprint.version	7.2.8

Cloud storage

The fileSystemConfigs parameter group includes parameters related to cloud storage configuration.

When forming the parameter keys, the {fileSystemType} should be replaced with an actual cloud storage type such as "s3", "adls_gen_2", or "wasb".

Parameter key	Description	Example key	Example value
File system common configurations			
fileSystemConfigs.{fileSystemType}.storageContainer	Name of container in Azure storage account (Cloudbreak + stackId)	fileSystemConfigs.s3.storageContainer	cloudbreak123
fileSystemConfigs.{fileSystemType}.type	Type of filesystem	fileSystemConfigs.s3.type	S3
fileSystemConfigs.{fileSystemType}.locations.[index].configFile	Configuration file used to configure the filesystem	fileSystemConfigs.s3.locations.[0].configFile	hbase-site
fileSystemConfigs.{fileSystemType}.locations.[index].property	Property key of filesystem path in defined config	fileSystemConfigs.s3.locations.[0].property	hbase.rootdir
fileSystemConfigs.{fileSystemType}.locations.[index].value	Value of filesystem path in defined config	fileSystemConfigs.s3.locations.[0].value	s3a://testranger/testrecipe2/apps/hbase/data
Amazon S3 configurations			
fileSystemConfigs.s3.storageContainer	Generated name (cloudbreak + stack id number)	fileSystemConfigs.s3.storageContainer	cloudbreak7941
fileSystemConfigs.s3.locations.[index].configFile	Hadoop component configuration file	fileSystemConfigs.s3.locations.[0].configFile	zeppelin-site
fileSystemConfigs.s3.locations.[index].property	Component property name	fileSystemConfigs.s3.locations.[0].property	zeppelin.notebook.dir

Parameter key	Description	Example key	Example value
fileSystemConfigs.s3.locations.[index].value	Component property value	fileSystemConfigs.s3.locations.[0].value	s3a://storagename/clustername/zeppelin/notebook
ADLS Gen2 configurations			
fileSystemConfigs.adls_gen_2.accountName	Name of the corresponding Azure storage account	fileSystemConfigs.adls_gen_2.accountName	teststorageaccount
fileSystemConfigs.adls_gen_2.storageContainerName	Container name in Azure storage account	fileSystemConfigs.adls_gen_2.storageContainerName	testcontainer

External database

The rds parameter group includes parameters related to external database configuration.

When forming the parameter keys, the {rdsType} should be replaced with the actual database type such as "cloudera_manager", "beacon", "druid", "hive", "oozie", "ranger", "superset", or some other user-defined type.

Parameter key	Description	Example key	Example value
rds.{rdsType}.connectionURL	JDBC connection URL	rds.hive.connectionURL	Value is specified in the following format: jdbc:postgresql://host:port/database
rds.{rdsType}.connectionDriver	JDBC driver used for connection	rds.hive.connectionDriver	org.postgresql.Driver
rds.{rdsType}.connectionUserName	Username used for the JDBC connection	rds.hive.connectionUserName	testuser
rds.{rdsType}.connectionPassword	Password used for the JDBC connection	rds.hive.connectionPassword	TestPssword123
rds.{rdsType}.databaseName	Target database of the JDBC connection	rds.hive.databaseName	myhivedb
rds.{rdsType}.host	Host of the JDBC connection	rds.hive.host	mydbhost
rds.{rdsType}.hostWithPortWithJdbc	Host of JDBC connection with port and JDBC prefix	rds.hive.hostWithPortWithJdbc	Value is specified in the following format: jdbc:postgresql://host:port
rds.{rdsType}.subprotocol	Sub-protocol from the JDBC URL	rds.hive.subprotocol	postgresql
rds.{rdsType}.connectionString	URL of JDBC the connection. In case of Ranger, this does not contain the port	rds.hive.connectionString	Value is specified in the following format: jdbc:postgresql://host:port/database
rds.{rdsType}.databaseVendor	Database vendor	rds.hive.databaseVendor	POSTGRES
rds.{rdsType}.withoutJDBCPrefix	URL of the JDBC connection without JDBC prefix	rds.hive.withoutJDBCPrefix	Value is specified in the following format: host:port/database

Gateway

The gateway parameter group includes parameters related to Knox gateway configuration.

Parameter key	Description	Example key	Example value
gateway.ssoProvider	Path to the SSO provider	gateway.ssoProvider	/test/sso/api/v1/websso
gateway.signKey	Base64 encoded signing key	gateway.signKey	
gateway.signPub	Signing certificate (x509 format)	gateway.signPub	
gateway.signCert	Public SSH key used for signing (standard public key format)	gateway.signCert	

Shared services

The sharedService parameter group includes parameters related to Data Lake configuration.

Parameter key	Description	Example key	Example value
sharedService.rangerAdminPassword	Admin password of the Ranger component	sharedService.rangerAdminPassword	Admin1234!
sharedService.datalakeCluster	Flag indicating that the cluster is a data lake cluster	sharedService.datalakeCluster	true
sharedService.datalakeClusterManagerIp	Cloudera Manager IP of data lake cluster	sharedService.datalakeClusterManagerIp	127.0.0.1
sharedService.datalakeClusterManagerFqdn	Cloudera Manager FQDN of data lake cluster (or the IP if FQDN cannot be found)	sharedService.datalakeClusterManagerFqdn	ip-10-88-88-28.example.com

Example: A cluster template with parameters

The template below is an example of a cluster template that includes placeholder values (the {{{general.clusterName}}} property) that will be dynamically fetched and replaced during cluster creation.

```
{
  "cdhVersion": "7.0.0",
  "products": [
    {
      "version": "7.0.0-1.cdh7.0.0.p0.1309226",
      "product": "CDH"
    }
  ],
  "services": [
    {
      "refName": "hdfs",
      "serviceType": "HDFS",
      "serviceConfigs": [],
      "roleConfigGroups": [
        {
          "refName": "hdfs-NAMENODE-BASE",
          "roleType": "NAMENODE",
          "base": true,
          "displayName": null,
          "configs": [
            {
              "name": "dfs_name_dir_list",
              "value": "/hadoopfs/{{{general.clusterName}}}/namenode"
            }
          ]
        }
      ],
      "roles": null,
      "displayName": null
    }
  ],
  "hostTemplates": [
    {
      "refName": "master",
      "roleConfigGroupsRefNames": [
        "hdfs-BALANCER-BASE",
        "hdfs-NAMENODE-BASE",
        "hdfs-SECONDARYNAMENODE-BASE",
        "hms-GATEWAY-BASE",
        "hms-HIVEMETASTORE-BASE",
        "hive_on_tez-HIVESERVER2-BASE",
        "hive_on_tez-GATEWAY-BASE",
        "hue-HUE_SERVER-BASE",
```

```
        "tez-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "spark_on_yarn-SPARK_YARN_HISTORY_SERVER-BASE",
        "livy-LIVY_SERVER-BASE",
        "zeppelin-ZEPPELIN_SERVER-BASE",
        "oozie-OOZIE_SERVER-BASE",
        "yarn-JOBHISTORY-BASE",
        "yarn-RESOURCEMANAGER-BASE",
        "zookeeper-SERVER-BASE",
        "knox-KNOX_GATEWAY-BASE"
      ],
      "cardinality": 1
    },
    {
      "refName": "worker",
      "roleConfigGroupsRefNames": [
        "hdfs-DATANODE-BASE",
        "hms-GATEWAY-BASE",
        "hive_on_tez-GATEWAY-BASE",
        "tez-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "livy-GATEWAY-BASE",
        "yarn-NODEMANAGER-WORKER"
      ],
      "cardinality": 1
    },
    {
      "refName": "compute",
      "roleConfigGroupsRefNames": [
        "hms-GATEWAY-BASE",
        "hive_on_tez-GATEWAY-BASE",
        "tez-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "yarn-NODEMANAGER-COMPUTE"
      ],
      "cardinality": 0
    }
  ],
  "displayName": "dataengineering",
  "cmVersion": "7.x.0",
  "repositories": [
    "http://cloudera-build-us-west-1.vpc.cloudera.com/s3/build/1309226/cdh
/7.x/parcels/"
  ],
  "clusterSpec": null
}
```

Example cluster template after {{{general.clusterName}}} is set to my-super-cluster based on the actual cluster name:

```
{
  "cdhVersion": "7.0.0",
  "products": [
    {
      "version": "7.0.0-1.cdh7.0.0.p0.1309226",
      "product": "CDH"
    }
  ],
  "services": [
    {
      "refName": "hdfs",
      "serviceType": "HDFS",
      "serviceConfigs": [],
```

```
      "roleConfigGroups": [
        {
          "refName": "hdfs-NAMENODE-BASE",
          "roleType": "NAMENODE",
          "base": true,
          "displayName": null,
          "configs": [
            {
              "name": "dfs_name_dir_list",
              "value": "/hadoopfs/my-super-cluster/namenode"
            }
          ]
        }
      ],
      "roles": null,
      "displayName": null
    }
  ],
  "hostTemplates": [
    {
      "refName": "master",
      "roleConfigGroupsRefNames": [
        "hdfs-BALANCER-BASE",
        "hdfs-NAMENODE-BASE",
        "hdfs-SECONDARYNAMENODE-BASE",
        "hms-GATEWAY-BASE",
        "hms-HIVEMETASTORE-BASE",
        "hive_on_tez-HIVESERVER2-BASE",
        "hive_on_tez-GATEWAY-BASE",
        "hue-HUE_SERVER-BASE",
        "tez-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "spark_on_yarn-SPARK_YARN_HISTORY_SERVER-BASE",
        "livy-LIVY_SERVER-BASE",
        "zeppelin-ZEPPELIN_SERVER-BASE",
        "oozie-OOZIE_SERVER-BASE",
        "yarn-JOBHISTORY-BASE",
        "yarn-RESOURCEMANAGER-BASE",
        "zookeeper-SERVER-BASE",
        "knox-KNOX_GATEWAY-BASE"
      ],
      "cardinality": 1
    },
    {
      "refName": "worker",
      "roleConfigGroupsRefNames": [
        "hdfs-DATANODE-BASE",
        "hms-GATEWAY-BASE",
        "hive_on_tez-GATEWAY-BASE",
        "tez-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "livy-GATEWAY-BASE",
        "yarn-NODEMANAGER-WORKER"
      ],
      "cardinality": 1
    },
    {
      "refName": "compute",
      "roleConfigGroupsRefNames": [
        "hms-GATEWAY-BASE",
        "hive_on_tez-GATEWAY-BASE",
        "tez-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "yarn-NODEMANAGER-COMPUTE"
```

```
      ],
      "cardinality": 0
    }
  ],
  "displayName": "dataengineering",
  "cmVersion": "7.x.0",
  "repositories": [
    "http://cloudera-build-us-west-1.vpc.cloudera.com/s3/build/1309226/cdh/
7.x/parcels/"
  ],
  "clusterSpec": null
}
```

For more information on setting these properties, see *Setting custom properties*.

Related Information

Setting custom properties

Upload a cluster template

Once you have your custom template ready, upload it to the Cloudera Management Console. Once the template is uploaded, you can select it during cluster creation.

About this task

Required role: EnvironmentCreator can create a shared resource and then assign users to it. SharedResourceUser or Owner of the shared resource can use the resource.

Procedure

1. Log in to the CDP web interface.
2. Navigate to Shared ResourcesCluster TemplatesCreate Cluster Template.
3. Provide the following information:

Parameter	Value
Name	Enter a name for your cluster template.
Description	(Optional) Enter a description for your cluster template.
Cluster Template Source	Select one of: • Text: Paste cluster template in JSON format. • File: Upload a file that contains the cluster template. • URL: Specify the URL for your cluster template. If you use this option, the URL must be accessible from the Management Console.

4. Click Register.

What to do next

To use the uploaded cluster templates, select it when creating a cluster. The option is available on the General Configuration page of the create cluster wizard. First select the Platform Version and then select your chosen cluster template under Cluster Template.

Manage cluster templates from CLI

You can manage cluster templates from CLI using cdp datahub commands.

Required role: Required role: EnvironmentCreator can create a shared resource and then assign users to it. The Owner of the shared resource can delete it.

- Register a new cluster templates by pasting its content: cdp datahub create-cluster-template --cluster-template-name <value> --cluster-template-content <value>
- List all available cluster templates: cdp datahub list-cluster-templates
- Describe a specific cluster templates: cdp datahub describe-cluster-template --cluster-template-name <value>
- Delete one or more existing cluster templates: cdp datahub delete-cluster-templates --cluster-template-name <value>

Cloudera Data Hub

Creating a Cluster on AWS

Date published: 2019-12-17
Date modified: 2023-06-27

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cluster deployment options 4

Default cluster configurations 4
- Data Engineering clusters 5
- Data Mart clusters 9
- Operational Database clusters 11
- Streams Messaging clusters 12
- Flow Management clusters 13
- Streaming Analytics clusters 14
- Data Discovery and Exploration clusters 15

Create a cluster from a definition on AWS 16

Create a custom cluster on AWS 18

Create a cluster with an external database 20

Create a cluster from an existing cluster 21

Creating a cluster with a specific Java version 21

Advanced cluster options 22
- Tags 22
- Image catalog 23
- Custom images and image catalogs 24
  - Creating a custom image and image catalog with the CDP CLI 25
  - Switching image catalogs 26
- Network and availability 27
- Hardware and storage 28
- Cloud storage 28
- Recipes 29
- Cluster template overrides 29
  - Creating a cluster template override 31
  - Creating a Cloudera Data Hub cluster with a cluster template override 35
  - Cluster template override CLI commands 35
- Custom properties 36
- Cluster deployment logs 37
  - Enabling/disabling cluster deployment log collection 37

Cluster deployment options

You have two basic options when creating a cluster: (1) create a cluster from one of the default or custom cluster definition, or (2) create a custom cluster.

Cluster definitions

Cloudera Data Hub includes a set of prescriptive cluster definitions that can be used to quickly provision clusters for common use cases. These default cluster definitions include prescriptive options related to compute instance types and storage options.

We recommend that you start by reviewing these definitions. If you conclude that they do not match your requirements, you can customize them and then save them as custom cluster definitions.

Custom deployment

Use the custom cluster deployment option if you would like to select specific Cloudera Runtime components for your cluster.

Note:

Currently, this feature is limited and only allows you to select a cluster template that determines which components will be used for your cluster.

Advanced options

The cluster wizard includes a set of advanced options that can be optionally configured. Some of these options require additional configuration prior to cluster creation, so if you would like to use these options, read the cluster planning documentation first.

Related Information

Default cluster configurations

Cluster Planning

Default cluster configurations

Cloudera Data Hub includes a set of prescriptive cluster configurations. Each of these default cluster configurations include a cloud-provider specific cluster definition, which primarily defines cloud provider settings. The cluster definition references a cluster template, which defines a number of Cloudera Runtime or Cloudera DataFlow components used for common data analytics and data engineering use cases.

Refer to the topic for each default cluster configuration to view the included services and compatible Runtime versions. These topics include links to documentation that will help you to understand the included components and use the workload cluster.

Many of the cluster components are included in the Cloudera Runtime software distribution. The Streams Messaging, Cloudera Flow Management, and Cloudera Streaming Analytics cluster configurations are part of Cloudera DataFlow for Data Hub and have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for more details.

You can access the default cluster definitions by clicking Environments, then selecting an environment and clicking the Cluster Definitions tab.

You can access the default cluster templates from Shared ResourcesCluster Templates.

To view details of a cluster definition or cluster template, click on its name. For each cluster definition, you can access a raw JSON file. For each cluster template, you can access a graphical representation ("list view") and a raw JSON file ("raw view") of all cluster host groups and their components.

Related Information

Cloudera DataFlow for Data Hub

Cloudera Runtime

Data Engineering clusters

Learn about the default Data Engineering clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Engineering provides a complete data processing solution, powered by Apache Spark and Apache Hive. Spark and Hive enable fast, scalable, fault-tolerant data engineering and analytics over petabytes of data.

Data Engineering cluster definition

This Data Engineering template includes a standalone deployment of Spark and Hive, as well as Apache Oozie for job scheduling and orchestration, Apache Livy for remote job submission, and Hue and Apache Zeppelin for job authoring and interactive analysis.

Cluster definition names

- Data Engineering for AWS
- Data Engineering HA - Spark3 for AWS

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering Spark3 for AWS

Cluster template name

- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: Apache Spark, Apache Hive, Apache Oozie."
- Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: HA: Apache Spark, Apache Hive, Apache Oozie."

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: The "Data Engineering: Apache Spark3" cluster template is deleted. Therefore, the "Data Engineering: Apache Spark3, Apache Hive, Apache Oozie" cluster template can be used instead.

Included services

- HDFS
- Hive
- Hue
- Livy
- Spark 3
- Yarn
- Zeppelin
- ZooKeeper
- Oozie is supported for Spark 3 as of Cloudera Runtime version 7.2.18

- Hive Warehouse Connector is supported as of Cloudera Runtime version 7.2.16.

Compatible Cloudera Runtime version

7.2.16, 7.2.17, 7.2.18

Topology of the Data Engineering cluster

Topology is a set of host groups that are defined in the cluster template and cluster definition used by Data Engineering. Data Engineering uses the following topology:

Host group	Description	Node configuration
Master	The master host group runs the components for managing the cluster resources including Cloudera Manager, Name Node, Resource Manager, as well as other master components such HiveServer2, HMS, Hue etc.	1 For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.4xlarge; gp2 - 100 GB For Cloudera Runtime versions 7.2.14+ DE, DE Spark3, and DE HA: AWS : m5.4xlarge; gp2 - 100 GB
Worker	The worker host group runs the components that are used for executing processing tasks (such as NodeManager) and handling storing data in HDFS such as DataNode.	3 For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.2xlarge; gp2 - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: AWS: r5d.2xlarge - (gp2/EBS volumes) DE HA: AWS: r5d.4xlarge - (gp2/EBS volumes)
Compute	The compute host group can optionally be used for running data processing tasks (such as NodeManager). By default the number of compute nodes is set to 1 for proper configurations of YARN containers. This node group can be scaled down to 0 when there are no compute needs. Additionally, if load-based auto-scaling is enabled with minimum count set to 0, the compute nodegroup will be resized to 0 automatically.	0+ For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.2xlarge; gp2 - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: AWS: r5d.2xlarge - (ephemeral volumes) DE HA: AWS: r5d.4xlarge - (ephemeral volumes)

Host group	Description	Node configuration
		Note: Compute nodes run YARN and require storage only for temporary data - this requirement is fulfilled by instance storage, so making the attached volumes count to 0 by default is more cost-efficient.
Gateway	The gateway host group can optionally be used for connecting to the cluster endpoints like Oozie, Beeline etc. This nodegroup does not run any critical services. This nodegroup resides in the same subnet as the rest of the nodegroups. If additional software binaries are required they could be installed using recipes.	0+ AWS : m5.2xlarge; gp2 - 100 GB

Service configurations			
Master host group Cloudera Manager, HDFS, Hive (on Tez), HMS, Yarn RM, Oozie, Hue, DAS, Zookeeper, Livy, Zeppelin and Sqoop	Gateway host group Configurations for the services on the master node	Worker host group Data Node and YARN NodeManager	Compute group YARN NodeManager

Configurations

Note the following:

- There is a Hive Metastore Service (HMS) running in the cluster that talks to the same database instance as the Data Lake in the environment.
- If you use CLI to create the cluster, you can optionally pass an argument to create an external database for the cluster use such as Cloudera Manager, Oozie, Hue, and DAS. This database is by default embedded in the master node external volume. If you specify the external database to be of type HA or NON_HA, the database will be provisioned in the cloud provider. For all these types of databases the lifecycle is still associated with the cluster, so upon deletion of the cluster, the database will also be deleted.
- The HDFS in this cluster is for storing the intermediary processing data. For resiliency, store the data in the cloud object stores.
- For high availability requirements choose the Data Engineering High Availability cluster shape.

Architecture of the Data Engineering HA for AWS cluster

The Data Engineering HA for AWS and Azure cluster shape provides failure resilience for several of the Data Engineering HA services, including Knox, Oozie, HDFS, HS2, Hue, Livy, YARN, and HMS.

Services that do not yet run in HA mode include Cloudera Manager, DAS, and Zeppelin.

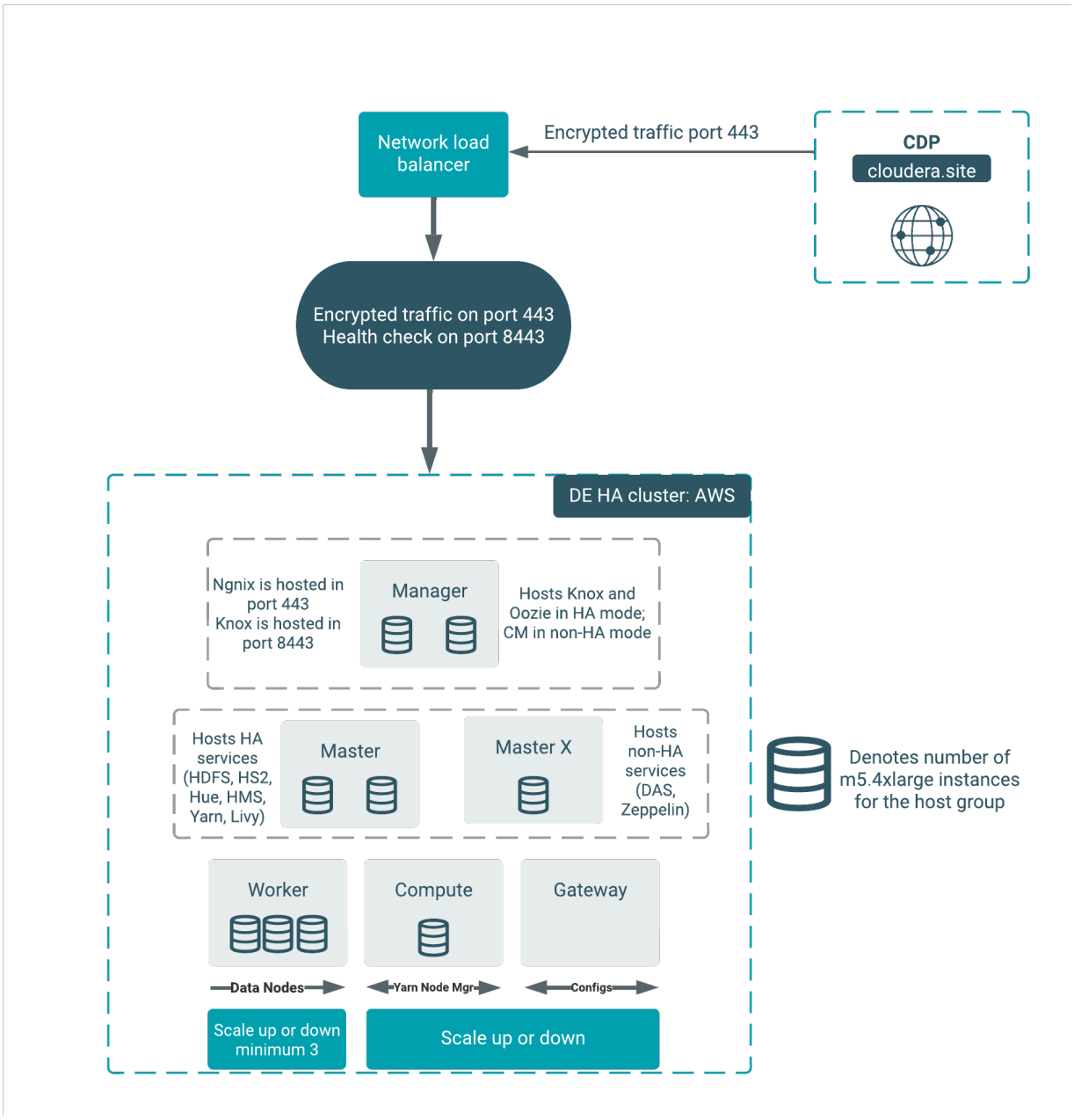

The architecture outlined in the diagram above handles the failure of one node in all of the host groups except for the “masterx” group. See the table below for additional details about the component interactions in failure mode:

Component	Failure	User experience
Knox	One of the Knox services is down	External users will still be able to access all of the UIs, APIs, and JDBC.
Cloudera Manager	The first node in manager host group is down	The cluster operations (such as repair, scaling, and upgrade) will not work.
Cloudera Manager	The second node in the manager host group is down	No impact.
HMS	One of the HMS services is down	No impact.
Hue	One of the Hue services is down in master host group	No impact.

HS2	One of the HS2 services is down in the master host group	External users will still be able to access the Hive service via JDBC. But if Hue was accessing that particular service it will not failover to the other host. The quick fix for Hue is to restart Hue to be able to use Hive functionality.
YARN	One of the YARN services is down	No impact.
HDFS	One of the HDFS services is down	No impact.
Nginx	Nginx in one of the manager hosts is down	Fifty percent of the UI, API, and JDBC calls will be affected. If the entire manager node is down, there is no impact. This is caused by the process of forwarding and health checking that is done by the network load-balancer.
Oozie	One of the Oozie servers is down in the manager host group.	No impact for AWS and Azure as of Cloudera Runtime version 7.2.11. If you create a custom template for DE HA, follow these two rules: **1.** Oozie must be in single hostgroup. **2.** Oozie and Hue must not be in the same hostgroup.

Important: If you are creating a DE HA cluster through the CDP CLI using the create-aws-cluster command, note that there is a CLI parameter to provision the network load-balancer in HA cluster shapes. Make sure to use the [--enable-load-balancer | --no-enable-load-balancer] parameter when provisioning a DE HA cluster via the CLI. For more information see the CDP CLI reference.

Custom templates

Any custom DE HA template that you create must be forked from the default templates of the corresponding version. You must create a custom cluster definition for this with the JSON parameter “enableLoadBalancers”: true , using the create-aws/azure/gcp-cluster CLI command parameter --request-template. Support for pre-existing custom cluster definitions will be added in a future release. As with the template, the custom cluster definition must be forked from the default cluster definition. You are allowed to modify the instance types and disks in the custom cluster definition. You must not change the placement of the services like Cloudera Manager, Oozie, and Hue. Currently the custom template is fully supported only via CLI.

The simplest way to change the DE HA definition is to create a custom cluster definition. In the Create Data Hub UI when you click Advanced Options, the default definition is not used fully, which will cause issues in the HA setup.

Related Information

HDFS
Hive
Hue
Oozie
Spark
YARN
Zookeeper

Data Mart clusters

Learn about the default Data Mart and Real Time Data Mart clusters, including cluster definition and template names, included services, and compatible Runtime version.

Data Mart is an MPP SQL database powered by Apache Impala designed to support custom Data Mart applications at big data scale. Impala easily scales to petabytes of data, processes tables with trillions of rows, and allows users to store, browse, query, and explore their data in an interactive way.

Data Mart clusters

The Data Mart template provides a ready to use, fully capable, standalone deployment of Impala. Upon deployment, it can be used as a standalone Data Mart to which users point their BI dashboards using JDBC/ODBC end points. Users can also choose to author SQL queries in Cloudera's web-based SQL query editor, Hue, and run them with Impala providing a delightful end-user focused and interactive SQL/BI experience.

Cluster definition names

- Data Mart for AWS

Cluster template name

Cloudera - Data Mart: Apache Impala, Hue

Included services

- HDFS
- Hue
- Impala

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17, 7.2.18

Real Time Data Mart clusters

The Real-Time Data Mart template provides a ready-to-use, fully capable, standalone deployment of Impala and Kudu. You can use a Real Time Data Mart cluster as a standalone Data Mart which allows high throughput streaming ingest, supporting updates and deletes as well as inserts. You can immediately query data through BI dashboards using JDBC/ODBC end points. You can choose to author SQL queries in Cloudera's web-based SQL query editor, Hue. Executing queries with Impala, you will enjoy an end-user focused and interactive SQL/BI experience. This template is commonly used for Operational Reporting, Time Series, and other real time analytics use cases.

Cluster definition names

- Real-time Data Mart for AWS

Cluster template name

Cloudera - Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 2
- Yarn

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster definition names

- Real-time Data Mart - Spark3 for AWS

Cluster template name

Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark3

Included services

- HDFS

- Hue
- Impala
- Kudu
- Spark 3
- Yarn

Compatible Cloudera Runtime versions

7.2.16, 7.2.17, 7.2.18

High availability

Cloudera recommends that you use high availability (HA), and track any services that are not capable of restarting or performing failover in some way.

Impala HA

The Impala nodes offer high availability. The following Impala services are not HA.

- Catalog service
- Statestore service

Kudu HA

Both Kudu Masters and TabletServers offer high availability.

Related Information

HDFS
Hue
Impala
Kudu
Spark
YARN

Operational Database clusters

The Operational Database (OpDB) template is removed from the Cloudera Data Hub. You can access the Cloudera Operational Database instead as a superior product.

The Cloudera Operational Database is a NoSQL database powered by Apache HBase designed to support custom OLTP applications that want to leverage the power of BigData. Apache HBase is a NoSQL, scale-out database that can easily scale to petabytes and stores tables with millions of columns and billions of rows.

Cloudera Operational Database also contains Apache Phoenix which provides a way to use HBase through an SQL interface.

Cloudera recommends you to use the Cloudera Operational Database to create Operational Database clusters.

Related Information

Cloudera Operational Database
Getting started with Operational Database
Before you create an Operational Database cluster
Creating an Operational Database cluster
HDFS
HBase
Knox
Zookeeper
Phoenix

Streams Messaging clusters

Learn about the default Streams Messaging clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streams Messaging provides the following features:

- Advanced messaging and real-time processing on streaming data using Apache Kafka
- Centralized schema management using Schema Registry
- Management and monitoring capabilities powered by Streams Messaging Manager
- Cross-cluster Kafka topic replication using Streams Replication Manger
- Kafka partition rebalancing with Cruise Control

This template sets up a fault-tolerant standalone deployment of Apache Kafka and supporting Cloudera components (Schema Registry, Streams Messaging Manager, Streams Replication Manager and Cruise Control), which can be used for production Kafka workloads in the cloud or as a disaster recovery instance for on-premises. Kafka clusters.

Note:

Streams Messaging clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streams Messaging cluster deployment
- Creating your first Streams Messaging cluster
- Connecting Kafka clients to Cloudera Public Cloud clusters

Cluster definition names

- Streams Messaging Heavy Duty for AWS
- Streams Messaging Light Duty for AWS
- Streams Messaging HA for AWS

Cluster template name

- Cloudera - Streams Messaging Heavy Duty
- Cloudera - Streams Messaging Light Duty
- Cloudera - Streams Messaging High Availability

Included services

- Kafka
- Schema Registry
- Streams Messaging Manager
- Streams Replication Manager
- Cruise Control
- Kafka Connect

Compatible Cloudera Runtime version

- 7.1.0 (Preview)
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11

- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streams Messaging cluster
Ingesting Data into Cloudera Public Cloud
Kafka
Schema Registry
Streams Messaging Manager
Streams Replication Manager

Flow Management clusters

Learn about the default Flow Management clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime versions.

Flow Management delivers high-scale data ingestion, transformation, and management to enterprises from any-to-any environment. It addresses key enterprise use cases such as data movement, continuous data ingestion, log data ingestion, and acquisition of all types of streaming data including social, mobile, clickstream, and IoT data.

The Flow Management template includes a no-code data ingestion and management solution powered by Apache NiFi. With NiFi's intuitive graphical interface and 300+ processors, Flow Management enables easy data ingestion and movement between CDP services as well as 3rd party cloud services. NiFi Registry is automatically set up and provides a central place to manage versioned Data Flows.

Note:

Flow Management clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Flow Management cluster deployment
- Creating your first Flow Management cluster
- Security considerations for Flow Management clusters
- Using Apache NiFi to ingest data into Cloudera Public Cloud
- Using NiFi and NiFi Registry

Cluster definition names

- Flow Management Light Duty for AWS
- Flow Management Heavy Duty for AWS

Cluster template name

- CDP - Flow Management: Light Duty
- CDP - Flow Management: Heavy Duty

Included services

- NiFi
- NiFI Registry

Compatible Cloudera Runtime versions

- 7.1.0
- 7.2.0

- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Flow Management cluster
Apache NiFi documentation
Apache NiFi Registry documentation

Streaming Analytics clusters

Learn about the default Streaming Analytics clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streaming Analytics offers real-time stream processing and stream analytics with low-latency and high scaling capabilities powered by Apache Flink.

Streaming Analytics templates include Apache Flink that works out of the box in stateless or heavy state environments. Beside Flink, the template includes its supporting services namely YARN, Zookeeper and HDFS. The Heavy Duty template comes preconfigured with RocksDB as state backend, while Light Duty clusters use the default Heap state backend. You can create your streaming application by choosing between Kafka, Kudu, and HBase as datastream connectors.

You can also use SQL to query real-time data with SQL Stream Builder (SSB) in the Streaming Analytics template. By supporting the SSB service in Cloudera Public Cloud, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. SSB is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

Note:

Streaming Analytics clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streaming Analytics cluster deployment
- Creating your first Streaming Analytics cluster
- Analyzing data using Apache Flink
- Querying data using SQL Stream Builder

Cluster definition names

- Streaming Analytics Light Duty for AWS
- Streaming Analytics Heavy Duty for AWS

Cluster template name

- 7.2.17 - Streaming Analytics Light Duty
- 7.2.17 - Streaming Analytics Heavy Duty

Included services

- Flink
- SQL Stream Builder
- YARN
- Zookeeper
- HDFS
- Kafka

Important: In the Streaming Analytics cluster templates, Kafka service is included by default to serve as a background service only for the websocket ouput and sampling feature of SQL Stream Builder. The Kafka service in the Streaming Analytics cluster template cannot be used for production, you need to use the Streams Messaging cluster template when Kafka is needed for your deployment.

Compatible Cloudera Runtime version

- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streaming Analytics cluster
Flink
YARN
Zookeeper
HDFS

Data Discovery and Exploration clusters

Learn about the default Data Discovery and Exploration clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Discovery and Exploration

Explore and discover data sets ad-hoc. Do relevance-based analytics over unstructured data (logs, images, text, PDFs, etc). Get started with search or log analytics. Make data more accessible to everyone with Data Discovery and Exploration.

Cluster Definition Names

- Data Discovery and Exploration for AWS

Cluster Template Name

- Data Discovery and Exploration

Included Services

- Solr
- Spark 2
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Versions

7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster Definition Names

- Data Discovery and Exploration - Spark3 for AWS
- Data Discovery and Exploration - Spark3 for Azure
- Data Discovery and Exploration - Spark3 for Google Cloud

Cluster Template Name

- Data Discovery and Exploration for Spark3

Included Services

- Solr
- Spark 3
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Version

7.2.18

Related Information

Solr
Spark
HDFS
Hue
YARN
Zookeeper

Create a cluster from a definition on AWS

You can quickly create clusters from default or custom cluster definitions within an existing AWS environment.

Before you begin

To create a Cloudera Data Hub cluster on AWS, you must have an existing AWS environment. Also, you should make sure that the Runtime version of the Data Lake cluster matches the Cloudera Runtime version of the Cloudera Data Hub cluster that you are about to create; If these versions don't match, you may encounter warnings and/or errors.

About this task

Required role: DataHubCreator, EnvironmentAdmin at the scope of the environment where the Cloudera Data Hub cluster is running, or Owner of the environment

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Environments > click on an environment where you would like to create a cluster > click Create Data Hub. The following page is displayed:

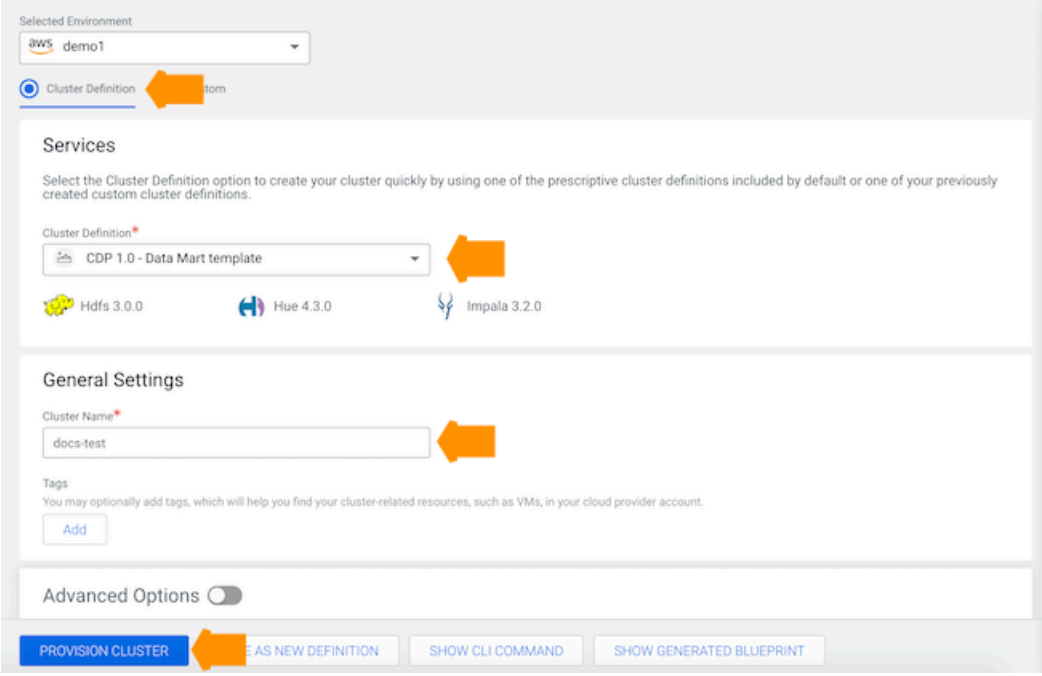

3. Select Cluster Definition.
4. From the Cluster Definition dropdown, select the cluster definition that you would like to use for your cluster.

   The cluster template referenced in the selected cluster definition determines which services are included in the cluster. The list of services is automatically shown below the selected cluster definition name:
5. Specify General Settings for your cluster:

Parameter	Description
Cluster Name	Enter a name for your cluster. The name must be between 5 and 40 characters, must start with a letter, and must only include lowercase letters, numbers, and hyphens.
Tags	(Optional) Add tags that Data Hub should use to tag your AWS resources. Click Add to add a tag, and then enter a key and value for each tag. Repeat the steps if you would like to add more tags. For more information about tags, refer to Tags.

6. Optionally, click on Advanced Options to modify advanced cluster settings. For more information on these options, refer to *Advanced cluster options*.
7. On AWS and Azure only: Optionally, when you have finished providing the cluster settings, you can click the Show CLI Command button at the bottom of the page to review or copy the CDP CLI command used to create the cluster. You can copy the command from the pop-up window that appears, either to provision the cluster later or for use in scripts.
8. You also have the option to review or copy the cluster template that is generated and will be used in cluster creation. Click the Show Generated Cluster Template button at the bottom of the page.
9. To proceed with cluster provisioning immediately, click on Provision Cluster.
10. You will be redirected to the Cloudera Data Hub cluster dashboard, and a new tile representing your cluster will appear at the top of the page.

    The following messages are written to the event history as your cluster is being created:

```
Cluster built; Cluster manager ip:10.97.82.237
8/2/2019, 6:10:44 PM
```

```
Updating Cluster Proxy service with gateway configuration
8/2/2019, 6:10:43 PM
Building cluster; Cluster manager ip:10.97.82.237
8/2/2019, 5:45:12 PM
Starting cluster services
8/2/2019, 5:42:33 PM
Mounting attached disks
8/2/2019, 5:42:18 PM
Setting up infrastructure metadata
8/2/2019, 5:42:14 PM
Bootstrapping infrastructure cluster
8/2/2019, 5:41:59 PM
Registering cluster with Cluster Proxy service
8/2/2019, 5:41:57 PM
Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Billing started, Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Infrastructure metadata collection finished
8/2/2019, 5:41:55 PM
Infrastructure creation took 96 seconds
8/2/2019, 5:41:52 PM
Creating infrastructure
8/2/2019, 5:40:16 PM
Setting up CDP image
8/2/2019, 5:40:15 PM
```

11. When your cluster is ready, its status will change to Running.

What to do next

You can access links to Cloudera Manager, and other cluster UIs and endpoints from cluster details.

Related Information

Advanced cluster options

Accessing a Cluster

Create a custom cluster on AWS

Create a custom Cloudera Data Hub cluster within an existing AWS environment.

Before you begin

To create a Cloudera Data Hub cluster on AWS, you must have an existing AWS environment. Also, you should make sure that the Cloudera Runtime version of the Data Lake cluster matches the Runtime version of the Cloudera Data Hub cluster that you are about to create; If these versions don't match, you may encounter warnings and/or errors.

https://www.youtube.com/embed/pGHHW9nOrPU

About this task

Required role: DataHubCreator, EnvironmentAdmin at the scope of the environment where the Data Hub is running, or Owner of the environment

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Environments > click on an environment > click Create Data Hub.

3. Under Selected Environment, confirm that the selected environment is the one where you would like to create your cluster.
4. Select Custom.
5. Under Platform Version, current Cloudera Runtime version is pre-selected.
6. Under Cluster template, select the cluster template that you would like to use.

   The selected cluster template determines which services are included in the cluster. Select the cluster template and the list of services is automatically shown below it:

   For more information about cluster templates, refer to Cluster templates.
7. Specify General Settings for your cluster:

Parameter	Description
Cluster Name	Enter a name for your cluster. The name must be between 5 and 40 characters, must start with a letter, and must only include lowercase letters, numbers, and hyphens.
Tags	(Optional) Add tags that Data Hub should use to tag your AWS resources. Click Add to add a tag, and then enter a key and value for each tag. Repeat the steps if you would like to add more tags. For more information about tags, refer to Tags.

8. Optionally, click on Advanced Options to modify advanced cluster settings. For more information on these options, refer to LINK.
9. Once done, click on Provision Cluster.
10. You will be redirected to the Cloudera Data Hub cluster dashboard, and a new tile representing your cluster will appear at the top of the page.

    The following messages are written to the event history as your cluster is being created:

```
Cluster built; Cluster manager ip:10.97.82.237
8/2/2019, 6:10:44 PM
Updating Cluster Proxy service with gateway configuration
8/2/2019, 6:10:43 PM
Building cluster; Cluster manager ip:10.97.82.237
8/2/2019, 5:45:12 PM
Starting cluster services
8/2/2019, 5:42:33 PM
Mounting attached disks
8/2/2019, 5:42:18 PM
Setting up infrastructure metadata
8/2/2019, 5:42:14 PM
Bootstrapping infrastructure cluster
8/2/2019, 5:41:59 PM
Registering cluster with Cluster Proxy service
8/2/2019, 5:41:57 PM
Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Billing started, Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Infrastructure metadata collection finished
8/2/2019, 5:41:55 PM
```

```
Infrastructure creation took 96 seconds
8/2/2019, 5:41:52 PM
Creating infrastructure
8/2/2019, 5:40:16 PM
Setting up CDP image
8/2/2019, 5:40:15 PM
```

11. When your cluster is ready, its status will change to Running.

What to do next

You can access links to Cloudera Manager, and other cluster UIs and endpoints from cluster details.

Related Information

Advanced cluster options

Accessing a Cluster

Create a cluster with an external database

Through the CDP CLI, you can configure an external, durable database when you create a Data Hub cluster. An external database keeps a persistent state for cluster services like Hue, DAS and Zeppelin in case of instance failures. You also have the option to configure an external database as a custom property within a cluster definition.

A top-level custom property for the external managed database is available to be set within a cluster definition:

```
"externalDatabase": {
  "availabilityType": "NONE" | "NON_HA" | "HA"
```

For more information on using custom cluster definitions to create clusters, see *Cluster definitions*.

A new flag, called --datahub-database, is available in the CDP CLI create-aws-cluster command. The supported values are NONE, NON_HA, and HA. To create a Data Hub cluster with an external managed database, use the following CLI options:

```
create-aws-cluster
          [--cluster-name <value>]
          [--cluster-definition-name <value>]
          [--environment-name <value>]
          [--cluster-template-name <value>]
          [--instance-groups <value>]
          [--subnet-id <value>]
          [--image <value>]
          [--tags <value>]
          [--request-template <value>]
          [--datahub-database <value>]
          [--overall-spot-percentage <value>]
          [--cli-input-json <value>]
          [--generate-cli-skeleton]
```

- --cluster-name: String. The name of the cluster. This name must be unique, must have between 5 and 40 characters, and must contain only lowercase letters, numbers and hyphens. Names are case-sensitive.
- --cluster-definition-name: String. The name or CRN of the cluster definition to use for cluster creation.
- --environment-name: Name or CRN of the environment to use when creating the cluster. The environment must be an AWS environment.
- --cluster-template-name: String. Name or CRN of the cluster template to use for cluster creation.
- --instance-groups: Array. Instance group details.
- --subnet-id: String. The subnet ID.
- --image: Object. The details of the image used for cluster instances.

- --tags: Array. Tags to be added to Data Hub-related resources.
- --request-template: String. JSON template to use for cluster creation. This is different from the cluster template and would be removed in the future.
- --datahub-database: String. Database type for the Data Hub cluster. Currently supported values: NONE, NON_HA, HA
- --overall-spot-percentage: Integer. Percentage of spot instances in all the instance groups. This may override the already set spotPercentage values under the instanceGroup(s).
- --cli-input-json: String. Performs service operation based on the JSON string provided. The JSON string follows the format provided by --generate-cli-skeleton. If other arguments are provided on the command line, the CLI values will override the JSON-provided values.
- --generate-cli-skeleton: Prints a sample input JSON to standard output. Note the specified operation is not run if this argument is specified. The sample input can be used as an argument for --cli-input-json.

For information on shorthand and JSON syntax of these options, and the output of the command, invoke the CDP CLI help.

Note: The external managed database is destroyed along with the Data Hub cluster.

Related Information

Cluster definitions

Create a cluster from an existing cluster

You can copy the CDP CLI command used to create a Cloudera Data Hub cluster on AWS or Azure and then use the command to create a new cluster with the same or similar settings.

Procedure

1. Click Data Hub Clusters in the left navigation menu and then click the tile of the Cloudera Data Hub cluster that you want to use to create the new cluster.
2. From the Data Hub details page, click the Actions menu in the top right of the page and then Show CLI Command.
3. Copy the CDP CLI command used for the cluster creation.
4. Before you use the command in the CDP CLI to create a new cluster, ensure that you change the --cluster-name parameter as this must be a unique name.

Results

You can now use the CDP CLI to run the command to create a new cluster.

Related Information

CDP CLI client setup

Creating a cluster with a specific Java version

When you create a new Cloudera Data Hub or Data Lake cluster, you can specify Java 8 or Java 11 as the default JDK.

Java 11

When you launch a new Cloudera Data Hub or Data Lake cluster, the default JDK version is Java 8 unless you specify Java 11. If you specify Java 11 as the default JDK version, Cloudera Manager and all Cloudera Runtime

services (as well as all workloads launched) will use pre-installed Java 11 dependencies rather than Java 8. This may be useful, for instance, when you have workloads such as Spark 3 jobs that are written in Java 11, when your workloads have Java 11 dependencies, or if you are running third-party components on a cluster that require Java 11.

The ability to launch new Cloudera Data Hub and Data Lake clusters with Java 11 is currently available only through the CDP CLI. Include the parameter --java-version in the create-<cloud_provider>-datalake and create-<cloud_provider>-datahub CLI commands. Valid inputs for the --java-version parameter are 8 and 11.

Launching clusters with Java 11 as the default is supported from Cloudera Runtime versions 7.2.15.11 and 7.2.16.200 and above. Creating clusters with the Java version property set to 11 on older Cloudera Runtime versions will be blocked by an internal validation. Currently, the Java version selection is only available when creating a new cluster.

Java 8 and 11 compatibility

Java 11 introduces breaking changes, which could lead to incompatibility with your current workloads or custom/third-party Java-based applications deployed on Cloudera Data Hub clusters.

Cloudera recommends that all existing production workloads are tested before moving them over to a Cloudera Data Hub cluster that is set to use Java 11 as the default JDK.

Note that Data Lake and Cloudera Data Hub hosts will continue to have both Java 8 and 11 pre-installed. The --java-version switch controls which Java versionn should be set as the OS default during bootstrapping of the hosts. Therefore, it is possible to continue to run jobs written for Java 8 even if the OS default is Java 11. Some services (for example, Spark) make this possible by setting the JAVA_PATH environment variable to point to the Java 8 path /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.[...].x86_64/jre/bin/java.

Advanced cluster options

In the create cluster wizard, click on Advanced Options to view the advanced cluster configuration options.

https://www.youtube.com/embed/Q8Xjb3-8KFk

While some of these advanced options can be configure in the wizard, others require prior setup.

The following options are available:

Tags

You can define tags that will be applied to your cluster-related resources (such as VMs) on your cloud provider account.

The tags added during cluster creation are displayed in your cloud account on the resources that Cloudera Data Hub provisioned for your clusters. You can use tags to categorize your cloud resources by purpose, owner, and so on. Tags come in especially handy when you are using a corporate cloud provider account and you want to quickly identify which resources belong to your cluster(s). In fact, your corporate cloud account admin may require you to tag all the resources that you create, in particular resources, such as VMs, which incur charges.

By default, the following tags are created:

Tag	Description
Cloudera-Resource-Name	The workload-appropriate Cloudera resource name. This CRN serves as a unique identifier for the resource over time.
Cloudera-Creator-Resource-Name	Cloudera resource name of the Cloudera user that created the resource.
Cloudera-Environment-Resource-Name	The name of the environment with which the resource is associated.

You can optionally add additional tags. To add custom tags:

1. In the create cluster wizard, navigate to the General Configuration page.

2. Specify your tags in the Tags section by providing a key and value for each tag.

Note:

It is not possible to add tags via Cloudera Data Hub after your cluster has been created. In this case, you can only add the tags manually via your cloud provider's interface.

To learn more about tags and their restrictions, refer to AWS documentation.

Related Information

Tagging your Amazon EC2 resources (AWS)

Image catalog

The options on the "Image Settings" page of the advanced create cluster wizard allow you to select custom image settings.

By default, Cloudera Data Hub uses the prewarmed image from the image catalog provided in Cloudera Data Hub. If necessary, you can also customize a default image.

Choose image catalog

Cloudera Data Hub uses the image catalog provided by default. If you would like to use a custom image catalog instead of the default image catalog, you must first prepare your custom images, then create and register an image catalog.

Choose image type

By default, Cloudera Data Hub uses the included prewarmed images with the default Cloudera Manager and Cloudera Runtime version, but you can select a different prewarmed image or customized prewarmed image to use for your cluster. Cloudera Data Hub currently supports the following types of images for launching clusters:

Image type	Description	Default images provided
Prewarmed Image	By default, Cloudera Data Hub launches clusters from prewarmed images. Prewarmed images include the operating system as well as Cloudera Manager and Cloudera Runtime. The Cloudera Manager and Cloudera Runtime version used on prewarmed images cannot be customized.	Yes
Custom Image	You can customize a default image for compliance or security reasons. You can then use the CDP CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create a Cloudera Data Hub cluster.	

Choose image

This option allows you to select a different image.

Related Information

Image catalog updates

Custom images and image catalogs

If necessary, you can use a custom Cloudera Runtime or FreeIPA image for compliance or security reasons. You can then use the Cloudera CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create a Data Lake, Cloudera Data Hub cluster or environment with a custom FreeIPA image.

Overview

A custom image should inherit most of its attributes from its source image, which is a default image that you select from the cdp-default image catalog.

The typical method of creating a Data Lake or Cloudera Data Hub picks up the latest pre-warmed image from the cdp-default image catalog for the specified version of Cloudera Runtime. These default images are pre-warmed VM images that contain a base URL to the default parcels in the Cloudera archive, amongst other configurations. If the default pre-warmed images do not suit your business needs, you can specify that the Data Lake, Cloudera Data Hub cluster or the environment (in the case of FreeIPA) uses a custom image instead.

What is a custom image?

A custom image is an entry in a custom image catalog that inherits most of its attributes from a source (default) image.

Custom image entries have:

- An image type: Cloudera Runtime [which includes Cloudera Data Hub and Data Lake images] or FreeIPA
- A source image ID that points to an image in the cdp-default image catalog
- A timestamp of creation
- An option to specify a VM region and image reference (such as an AMI ID) if you are overriding the source image with a custom VM image
- An option to override the parcel base URL

Why use a custom image?

You might require a custom image for compliance or security reasons (a “hardened” image), or to have your own packages pre-installed on the image, for example monitoring tools or software. You might also want to specify a custom image if you need to use a default image with a specific Runtime maintenance version applied, rather than simply specifying the latest major Runtime version.

Note: When customizing VM images, note that certain customizations (for example, CIS hardening rules) may not be compatible with Cloudera.

What can you customize?

In a custom image entry, you can override the VM images themselves with your own custom images that are sufficiently hardened. Importantly, you should only customize a default image from the cdp-default catalog as opposed to creating one from scratch. You can also override the default parcel base URL (at archive.cloudera.com) with your own host site.

What is a custom image catalog?

A custom image catalog is simply a catalog that holds custom images. A custom image catalog can contain a single or multiple custom image entries.

Custom image catalogs have:

- A name. The name is a unique identifier and is used to refer to the catalog during environment, Data Lake, and Cloudera Data Hub cluster creation; as well as during catalog operations like creating an image.

- A description.
- An owner. The owner is the user who runs the command to create the catalog.

What is the process for creating a custom image and catalog?

- If you are replacing the VM images in a custom image entry with a customized version, you should first prepare the image by modifying an official Cloudera default image, which you can find under Shared Resources > Image Catalogs > cdp-default.
- Select a source image from the cdp-default image catalog to be the source of customization. When you run the CLI command to find a default image, you specify the Runtime version, provider, image type, or a combination of the three.
- Create a custom image catalog, or identify an existing catalog where you want to save the custom image entry.
- Apply the necessary changes to the custom image entry, like the override AMI IDs with the new, customized AMIs; or add a new parcel base URL using the `--base-parcel-url` command when you set the custom image.
- You can then create an environment, Data Lake, or Cloudera Data Hub cluster, based on custom catalogs via the CDP CLI.

Creating a custom image and image catalog with the CDP CLI

You can create a custom Cloudera Runtime or FreeIPA image and image catalog through the CDP CLI.

Before you begin

If you are replacing the VM images in a custom image entry with a customized version, you must first prepare the image by modifying an official Cloudera default image, which you can find under Shared Resources Image Catalogs cdp-default . Take note of the image reference, such as the AMI ID.

Procedure

1. To find a source image from the cdp-default catalog that you want to use as the source of your custom image entry, run the following command:

   ```
   cdp imagecatalog find-default-image --provider <cloud provider> --image-
   type <image type> --runtime-version <Cloudera Runtime version>
   ```

   For example:

   ```
   cdp imagecatalog find-default-image --provider AWS --image-type runtime --
   runtime-version 7.2.12
   ```

2. A custom image requires a custom catalog. If you haven't yet created a custom catalog, or if you want to create a new one for a new custom image, run the following command:

   ```
   cdp imagecatalog create-custom-catalog --catalog-name <unique catalog na
   me> --description <catalog description>
   ```

   For example:

   ```
   cdp imagecatalog create-custom-catalog --catalog-name my custom catalog --
   description test catalog
   ```

3. Within the custom image catalog that you created (or an older custom catalog that you want to use), create a custom image entry with the selected source image marked as its source. Providing your own VM images is optional. Run the following command:

   ```
   cdp imagecatalog set-<image-type>-image --catalog-name <name of the custom
    catalog> --vm-images region=<region of customized image>,imageReference=<
   ```

```
cloud provider specific ID of a customized image> --source-image-id <cdp
 image ID of source image>
```

For example:

```
cdp imagecatalog set-runtime-image --catalog-name my custom catalog --vm-
images region=eu-central-1,imageReference=ami-7torotmhqi6q7438y --source-
image-id 8t4y9853-12b6-3n6z-75dh-tx775k4c793w
```

4. You can then apply the necessary changes to the custom image entry, like overriding AMI IDs with new, customized ones, or adding a new parcel base URL.

Command	Description
cdp imagecatalog find-default-image	Finds the default images in the cdp-default image catalog for a specified version of Cloudera Runtime. Parameters: • --provider • --image-type • --runtime-version You can provide any combination of these parameters.
cdp imagecatalog create-custom-catalog	Creates a custom catalog. Parameters: • --catalog-name • --description –catalog-name is required.
cdp imagecatalog set-runtime-image or cdp imagecatalog set-freeipa-image	Creates a custom image entry (either Cloudera Runtime [Cloudera Data Hub/Data Lake] or FreeIPA) within the specified catalog. Parameters: • --catalog-name • --vm-images [region,imageReference] • --source-image-id • --image-id • --base-parcel-url --catalog-name and --source-image-id are required.

Switching image catalogs

You can switch the image catalog of an already existing Cloudera Data Hub, Data Lake, or FreeIPA cluster. You may want to switch the image catalog for a cluster in order to restrict which Cloudera Runtime version can be upgraded to, or in order to move to custom images for an existing cluster.

Use the following CDP CLI commands to switch the image catalog for an existing cluster:

- FreeIPA:

```
cdp environments set-catalog --environment $ENVIRONMENT_NAME --catalog $
CATALOG
```

Parameter	Description
--environment	Name or CRN of the environment that holds the FreeIPA installation.
--catalog	URL of the FreeIPA catalog to be used.

- Cloudera Data Hub:

```
cdp datahub set-catalog --cluster $DATAHUB_NAME --catalog-name $CATALOG
```

Parameter	Description
--cluster	Name or CRN of the Cloudera Data Hub cluster for which you want to use the new image catalog.
--catalog-name	Name of the image catalog to be used. The image catalog must be a JSON based catalog, and switching is only possible from catalogs that are JSON based.

- Data Lake:

```
cdp datalake set-catalog --datalake $DATALAKE_NAME --catalog-name $CATALOG
```

Parameter	Description
--datalake	Name of the Data Lake for which you want to use the new image catalog.
--catalog-name	Name of the image catalog to be used. The image catalog must be a JSON based catalog, and switching is only possible from catalogs that are JSON based.

Network and availability

The "Network and availability" options allow you to customize the networking settings of your cluster.

On the Network and Availability page, provide the following to specify the networking resources that will be used for your cluster:

Parameter	Description
Select Subnet	If your environment includes a single subnet, that subnet is preselected and it cannot be changed. If your environment includes more than one subnet, you can select the subnet(s) in which your cluster will be provisioned. If you would like to deploy your Cloudera Data Hub cluster in multiple availability zones, you should select multiple subnets.
Select Azure Database Server	A Cloudera Data Hub cluster uses the same Flexible Server or Single Server settings as the environment in which it runs but you can choose to enable a Flexible server on Cloudera Data Hub clusters running in an environment that uses a Single server. For more information, see Using Azure Database for PostgreSQL Flexible Server.

Hardware and storage

The "Hardware and storage" options allow you to customize the cloud provider specific cluster hardware and storage options.

The Hardware and Storage options can be selected for each host group. To edit this section for a specific host group, click on the . When done editing, click on the to save the changes. Repeat for these steps for all host groups that you would like to edit.

The following hardware and storage settings are available:

Parameter	Description
Cloudera Manager Server	You must select one node for Cloudera Manager Server by clicking the button. The “Instance Count” for that host group must be set to “1”. If you are using one of the default cluster templates, this is set by default.
Instance Type	Select an instance type. For information about instance types on AWS refer to Amazon EC2 Instance Types in AWS documentation.
Instance Count	Enter the number of instances of a given type. Default is 1.
Storage Type	Select the volume type. The options vary by instance type and include: (1) Ephemeral (2) Magnetic (3) General Purpose SSD, (4) Throughput Optimized HDD. For more information about these options refer to Amazon EC2 Instance Store in AWS documentation. **Note:** Stopping and restarting Cloudera Data Hub clusters using ephemeral storage is not supported.
Encryption	Under Encryption Key, you can select an existing encryption key. For more information, refer to EBS Encryption on AWS.
Attached Volumes Per Instance	Enter the number of volumes attached per instance. Default is 1.
Volume Size	Enter the size in GB for each volume. Default is 100.
Root Volume Size	This option allows you to increase or decrease the root volume size. Default is 200 GB. This option is useful if your custom image requires more space than the default 200 GB. If you use a custom Cloudera Data Hub template specifying a root volume size smaller than 200GB, you may encounter an error.

Cloud storage

The options on the "Cloud Storage" page allow you to optionally specify the base storage location used for YARN and Zeppelin.

During environment creation under Data Access > Storage Location Base you configure a default S3 base storage location for the environment, and all Cloudera Data Hub clusters created within that environment use this location. The Cloud Storage options in the Cloudera Data Hub cluster wizard allow you to additionally specify a different location for YARN application logs and Zeppelin Notebook's root directory:

- Existing Base Storage Location - By default, this is set to the Storage Location Base configured on environment level. If you do not want to make any changes, simply leave this blank. If you would like to use a different location for YARN application logs and Zeppelin Notebook's root directory, you can specify a different S3 location. Note that the specified S3 location must exist prior to Cloudera Data Hub cluster creation and that you must adjust the IAM policies created during environment's cloud storage setup to make sure that IDBroker has write access to this location.
- Path for YARN Application Logs property - This directory structure gets created automatically during cluster creation. You can customize it if you would like it be different than what is suggested by default.
- Path for Zeppelin Notebooks Root Directory property - This directory structure gets created automatically during cluster creation. You can customize it if you would like it to be different than what is suggested by default.

Any S3 bucket that you designate for Cloudera Data Hub cloud storage on AWS must be in the same region as the environment.

Related Information

Minimal setup for cloud storage

Recipes

The "Recipes" option allows you to select previously uploaded recipes (scripts that can be run pre or post cluster deployment) for each host group.

This option is available from the Cluster Extensions page of the advanced create cluster wizard. For more information, refer to the documentation linked below.

Related Information

Recipes

Cluster template overrides

You can specify custom configurations that override or append the properties in a built-in Cloudera Data Hub template or a custom template.

Overview

You can launch Cloudera Data Hub clusters from a set of pre-defined cluster templates created for prescriptive use cases. These cluster templates are a “shared resource” that define the list of services that will be installed on the Cloudera Data Hub, including their configurations.

For example, this is a portion of a default template:

```
{
  "services": [
    {
      "refName": "zookeeper",
      "serviceType": "ZOOKEEPER",
      "serviceConfigs": [
        {
          "name": "service_config_suppression_server_count_validator",
          "value": "true"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "zookeeper-SERVER-BASE",
          "roleType": "SERVER",
          "base": true
        }
      ]
    },
    {
      "refName": "hdfs",
      "serviceType": "HDFS",
      "serviceConfigs": [
        {
          "name": "hdfs_verify_ec_with_topology_enabled",
          "value": false
        },
        {
          "name": "core_site_safety_valve",
```

```
          "value": "<property><name>fs.s3a.buffer.dir</name><value>${env.LOC
AL_DIRS:-${hadoop.tmp.dir}}/s3a</value></property><property><name>fs.s3a.com
mitter.name</name><value>directory</value></property>"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "hdfs-NAMENODE-BASE",
          "roleType": "NAMENODE",
          "base": true,
          "configs": [
            {
              "name": "role_config_suppression_namenode_java_heapsize_mini
mum_validator",
              "value": "true"
            },
            {
              "name": "role_config_suppression_fs_trash_interval_minimum_
validator",
              "value": "true"
            },
            {
              "name": "fs_trash_interval",
              "value": "0"
            },
            {
              "name": "fs_trash_checkpoint_interval",
              "value": "0"
            },
            {
              "name": "erasure_coding_default_policy",
              "value": " "
            }
          ]
        },
        {
          "refName": "hdfs-SECONDARYNAMENODE-BASE",
          "roleType": "SECONDARYNAMENODE",
          "base": true
        },
        {
          "refName": "hdfs-DATANODE-BASE",
          "roleType": "DATANODE",
          "base": true
        },
        {
          "refName": "hdfs-BALANCER-BASE",
          "roleType": "BALANCER",
          "base": true
        },
        {
          "refName": "hdfs-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true,
          "configs": [
            {
              "name": "dfs_client_use_trash",
              "value": false
            },
            {
              "name": "role_config_suppression_hdfs_trash_disabled_validat
or",
              "value": "true"
            },
```

```
            {
              "name": "hdfs_client_env_safety_valve",
              "value": "HADOOP_OPTS=\"-Dorg.wildfly.openssl.path=/usr/lib64
${HADOOP_OPTS}\""
            }
          ]
        }
      ]
```

In this section of an example cluster template, two primary types of service configurations are visible: serviceConfigs and configurations for various roleConfigGroups, in addition to a special type of service configuration called safety_valves. You can read about the details of these configs in the *Cloudera Manager Configuration Properties* reference.

Oftentimes you may want to modify the serviceConfigs, roleConfigGroups, and safety_valves present in a default Cloudera Data Hub cluster template. Although you can create a custom cluster template by modifying the JSON of a default template, this process can be cumbersome and error-prone.

As an alternative to creating a custom template, you can specify custom configurations that override or append the properties in a default template. These custom configurations are saved as a shared resource called “cluster template overrides,” and can be used and re-used across Cloudera Data Hub clusters in different environments. As a shared resource, they do not need to be attached to a specific Cloudera Runtime version.

By using a default cluster template along with a cluster template override, you can create a customized Cloudera Data Hub cluster, along with receiving improvements to the default templates that are present in newer Cloudera Runtime releases. Cluster template overrides can be used to override or append properties present in both the identified types of service configurations, as well as safety valves.

Important: If you customize safety valves, the provided value will be appended to the already existing value for that safety valve. Custom configs will not override the existing value.

You are not limited to customizing the configs present in a default cluster template; you can add any valid configuration recognized by Cloudera Manager for an included service. New configurations are appended to the cluster template when the cluster is launched.

Cluster template overrides can be applied to custom templates as well. If you want to apply a cluster template override to a custom template that contains properties that are dynamically replaced during cluster creation, the cluster template override will override any dynamically-replaced properties when the two conflict.

Limitations

At present, there is no way to validate any individual property to ensure that it is valid and recognised by Cloudera Manager, so that it can be overridden to the desired value. Adding an incorrect config name can lead to errors while installing the Cloudera Manager template on the Cloudera Data Hub cluster, or the property could be ignored by Cloudera Manager entirely.

An invalid value for a particular property can also cause errors during the cluster creation process. Carefully review the *Cloudera Manager Configuration Properties* reference in regards to configurations that you want to customize.

Cluster template overrides are only for overriding/appending the serviceConfigs, roleConfigGroups, and safety_valves in a Cloudera Data Hub template. Unlike creating a custom template, you can not add a service to the template.

Related Information

Cloudera Manager Configuration Properties

Building a custom Cloudera Data Hub cluster template

Custom Properties

Creating a cluster template override

From the Shared Resources menu, you can create a cluster template override to overwrite the default values of service and role configs in a Cloudera Data Hub template.

Procedure

1. In the Cloudera Management Console UI, Shared Resources Cluster Template Overrides .

2. Click Create Custom Override.

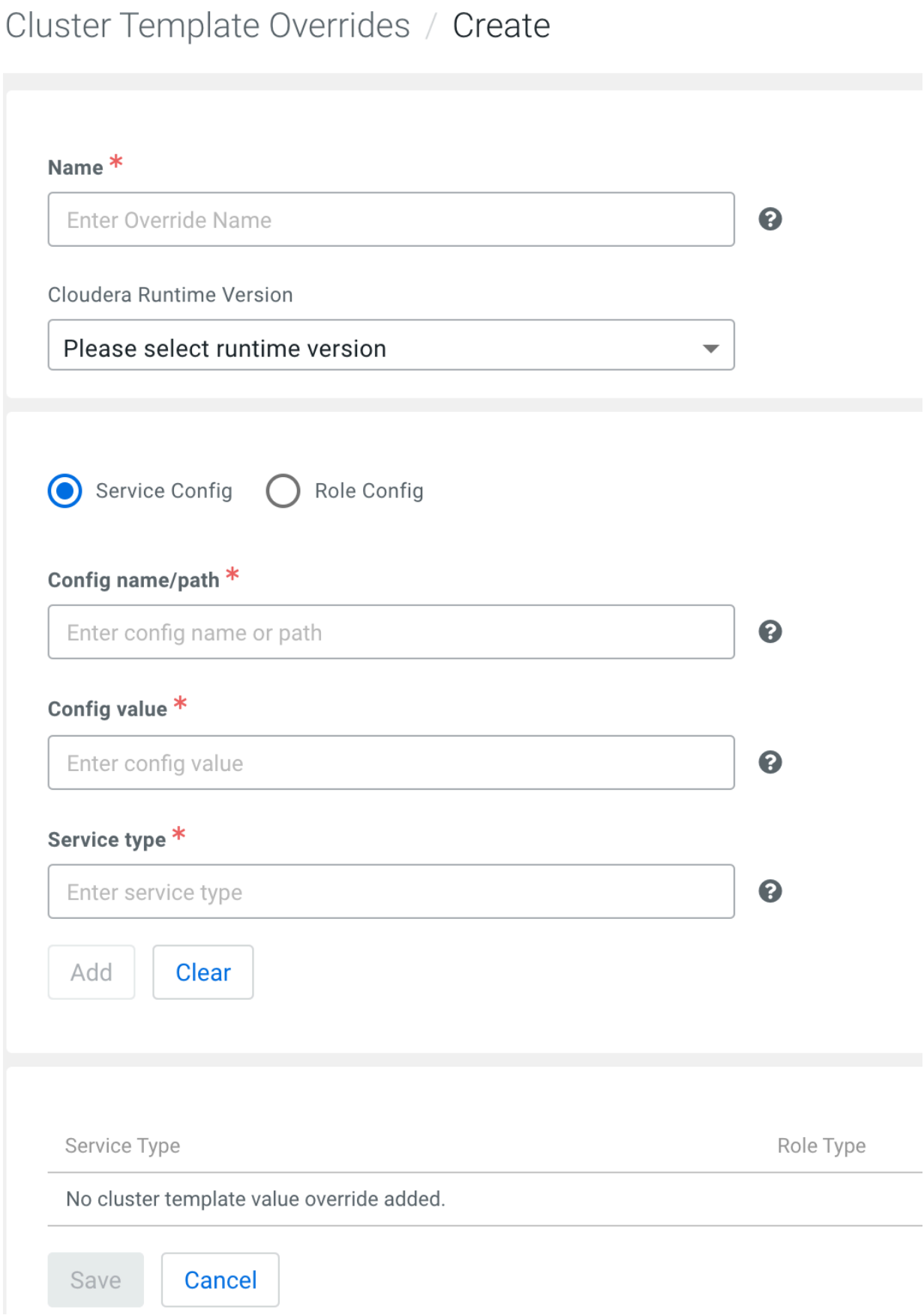

3. Provide the following information:

Property	Description
Name	Provide a unique name for the override.
Cloudera Runtime Version	Select a specific Cloudera Runtime version to which this override applies, or select “Not Runtime specific.”
Service Config / Role Config	Use the radio buttons to select whether you are customizing a service config or a role config.
Config Name/Path	Exact name/path of the config are you customizing.
Config Value	New value for the config. Ensure the value is valid.
Service Type	From the drop-down menu, select the service type for the service or role config that you are customizing.
Role Type	For role configurations: the role type as it appears in the cluster template that you will apply the override to (or any valid role configuration that is recognized by Cloudera Manager).

For example:

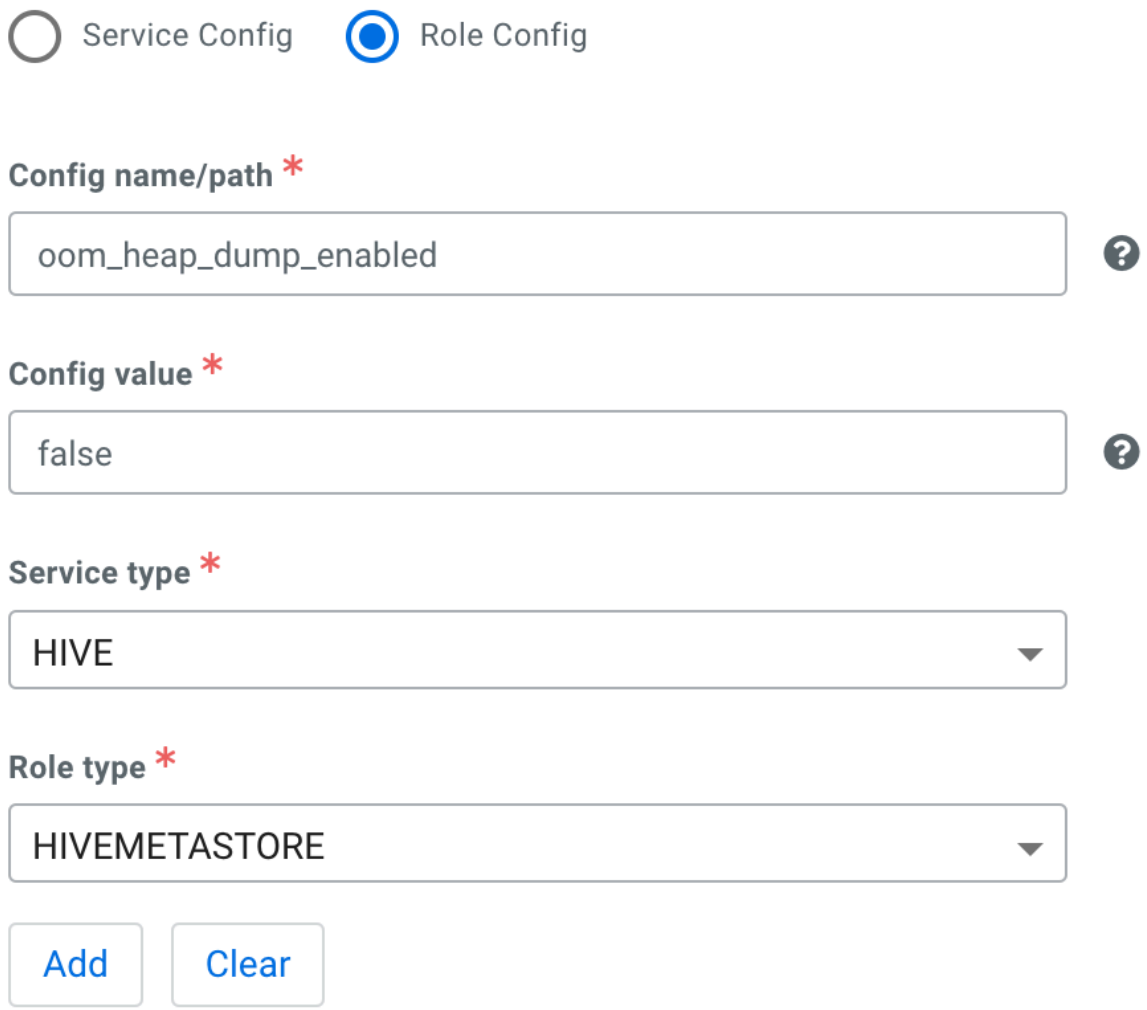

4. Click Add and then repeat the previous step (excluding the Name and Cloudera Runtime Version) for each service or role config that you want to customize in this template override.

5. When you are finished adding service and role configs to the template override, click Save. The override appears in the list of saved overrides, accessible from the Shared Resources menu.

Creating a Cloudera Data Hub cluster with a cluster template override

You can use a cluster template override that you previously created to provision a Cloudera Data Hub cluster.

Procedure

1. Click Data Hub Clusters Create Data Hub Cluster .
2. Enter the information required to create your cluster, then at the bottom of the cluster creation wizard, select Advanced OptionsCluster Extensions.
3. From the Cluster Templates Overrides drop-down box, select the template override that you previously created and want to apply when the cluster is provisioned. The only visible overrides are those with a Cloudera Runtime version matching the selected cluster definition version, or those that are Runtime-agnostic.

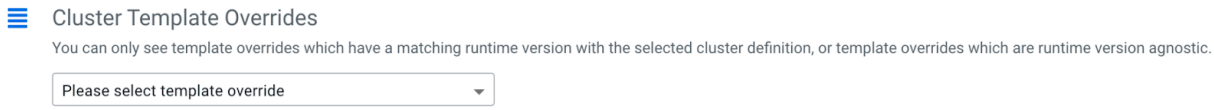

4. Click Provision Cluster.

Results

The Cloudera Data Hub cluster is provisioned using the service and role cofigs that you specified in the template override.

Cluster template override CLI commands

You can use the CDP CLI to create cluster template overrides, manage the cluster template overrides, and provision a Data Hub cluster using a cluster template override.

Table 1: Cluster Template Override CLI

Command	Description	Example
cdp datahub create-custom-configurations	Creates a new cluster template override shared resource with an indefinite amount of properties.	`cdp datahub create-custom-configurations --custom-configurations-name "test-custom-configs" --configurations configName="name",configValue="value",serviceType="hdfs" configName="name2",configValue="value2",roleType="role2",serviceType="hive_on_tez" .... --runtime-version "7.2.8"`
cdp datahub list-custom-configurations	Lists all the cluster template overrides for the account.	
cdp datahub describe-custom-configurations	Returns the name, CRN, the properties, and the platform version associated with the cluster template override resource.	`cdp datahub describe-custom-configurations --custom-configurations "test-overrides"`
cdp datahub delete-custom-configurations	Deletes multiple custom configs given their names or CRNs.	`cdp datahub delete-custom-configurations –custom-configurations "test-override" "test-override-2" ...`
cdp datahub create-aws-clustercdp datahub create-azure-clustercdp datahub create-gcp-cluster	Launches a Cloudera Data Hub cluster with custom configs from a cluster template override. The parameter for specifying a cluster template override is:--custom-configurations-name	`cdp datahub create-aws-cluster --cluster-name "de-dh1" --environment-name "local-aws-env" --cluster-template-name "7.2.8 - Data Engineering: Apache Spark, Apache Hive, Apache Ozie" --cluster-definition-name "7.2.8 - Data Engineering for AWS" --custom-configurations-name "test-override"`

Custom properties

The "Custom Properties "option allows you to set properties on a per-cluster basis.

This option is available from the Cluster Extensions page of the advanced create cluster wizard. For more information, refer to the documentation linked below.

Related Information

Custom Properties

Cluster deployment logs

Cluster deployment logs gather the same diagnostic (system and service) logs that are collected into your cloud storage location, but the Cloudera platform sends the logs to Cloudera engineering and support for troubleshooting purposes. You can apply configurable redaction rules on any sensitive data.

Cluster deployment logs are disabled by default. You can change this setting during environment creation or after environment creation, though the Cloudera platform will only collect logs for new deployments. You can also set a default behavior at the Cloudera account level, so that you won't have to enable the setting every time in the wizard. The logs are only gathered during deployment, which is approximately the first 20 minutes of cluster creation.

You can configure anonymization rules for log collection at the Cloudera account level. The rules are a list of rule objects with two fields: a regex pattern (PCRE) as the value, and a replacement string. Configure these rules to hide any sensitive data. By default, the rules hide email addresses and card numbers. This default behavior is the same as Cloudera Manager diagnostics collection.

The anonymization options are available under Global ResourcesTelemetry:

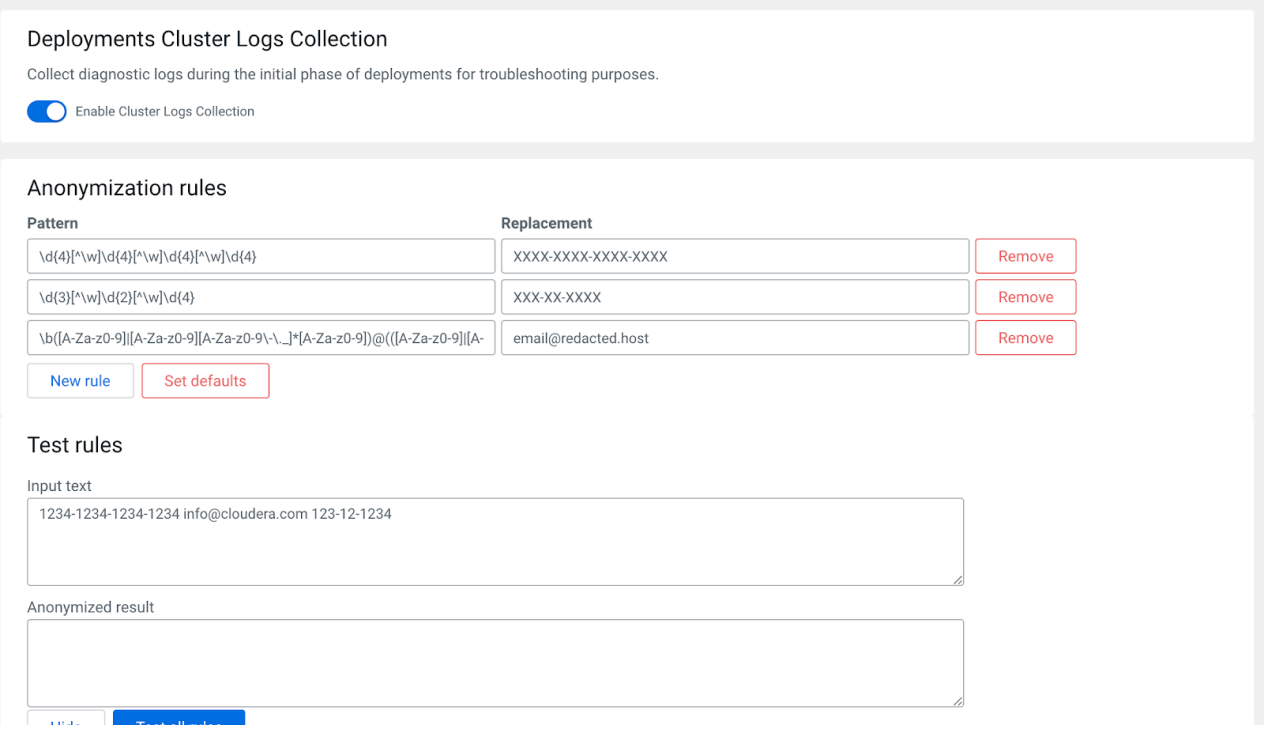

Currently, cluster deployment logs are collected only for Data Lake and Cloudera Data Hub cluster shapes.

Enabling/disabling cluster deployment log collection

You can enable cluster deployment log collection during environment creation. You can also enable or disable it after environment creation.

To enable cluster deployment log collection during environment creation:

1. Click the slider button under **Telemetry** in the **Region, Networking, Security, and Storage** page of environment registration.

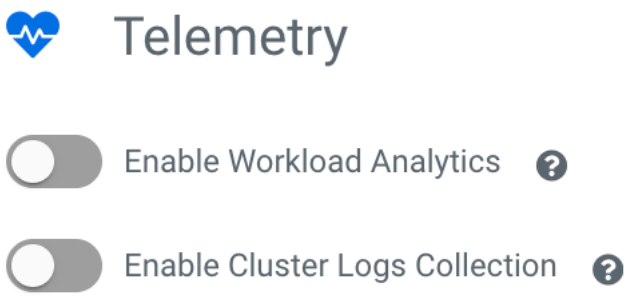

2. Select the checkbox to consent to data collection and click Enable.

To enable or disable cluster log deployment collection after environment creation:

1. Navigate to the **Environments** interface.
2. Select the desired environment, then click the Summary tab.
3. Scroll down to **Telemetry** and click the Edit button.
4. Click the slider button under Enable Deployment Cluster Logs Collection.
5. Select the checkbox to consent to data collection and click Enable.

To enable or disable cluster log deployment collection at the Cloudera account level (globally):

1. From the left navigation menu, click Global Settings and then Telemetry.
2. Click the slider button under Deployment Cluster Logs Collection.
3. Select the checkbox to consent to data collection and click Enable.

Cloudera Data Hub

Creating a Cluster on Azure

Date published: 2019-12-17
Date modified: 2023-06-27

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cluster deployment options 4

Default cluster configurations 4
- Data Engineering clusters 5
- Data Mart clusters 10
- Operational Database clusters 11
- Streams Messaging clusters 12
- Flow Management clusters 13
- Streaming Analytics clusters 14
- Data Discovery and Exploration clusters 16

Create a cluster from a definition on Azure 16

Create a custom cluster on Azure 18

Create a cluster with an external database 20

Create a cluster from an existing cluster 21

Creating a cluster with a specific Java version 21

Advanced cluster options 22
- Tags 22
- Image catalog 23
- Custom images and image catalogs 24
  - Creating a custom image and image catalog with the CDP CLI 25
  - Switching image catalogs 26
- Network and availability 27
- Hardware and storage 28
- Cloud storage 29
- Recipes 29
- Cluster template overrides 29
  - Creating a cluster template override 32
  - Creating a Cloudera Data Hub cluster with a cluster template override 35
  - Cluster template override CLI commands 35
- Custom properties 36
- Cluster deployment logs 37
  - Enabling/disabling cluster deployment log collection 37

Cluster deployment options

You have two basic options when creating a cluster: (1) create a cluster from one of the default or custom cluster definition, or (2) create a custom cluster.

Cluster definitions

Cloudera Data Hub includes a set of prescriptive cluster definitions that can be used to quickly provision clusters for common use cases. These default cluster definitions include prescriptive options related to compute instance types and storage options.

We recommend that you start by reviewing these definitions. If you conclude that they do not match your requirements, you can customize them and then save them as custom cluster definitions.

Custom deployment

Use the custom cluster deployment option if you would like to select specific Cloudera Runtime components for your cluster.

Note:

Currently, this feature is limited and only allows you to select a cluster template that determines which components will be used for your cluster.

Advanced options

The cluster wizard includes a set of advanced options that can be optionally configured. Some of these options require additional configuration prior to cluster creation, so if you would like to use these options, read the cluster planning documentation first.

Related Information

Default cluster configurations

Cluster Planning

Default cluster configurations

Cloudera Data Hub includes a set of prescriptive cluster configurations. Each of these default cluster configurations include a cloud-provider specific cluster definition, which primarily defines cloud provider settings. The cluster definition references a cluster template, which defines a number of Cloudera Runtime or Cloudera DataFlow components used for common data analytics and data engineering use cases.

Refer to the topic for each default cluster configuration to view the included services and compatible Runtime versions. These topics include links to documentation that will help you to understand the included components and use the workload cluster.

Many of the cluster components are included in the Cloudera Runtime software distribution. The Streams Messaging, Cloudera Flow Management, and Cloudera Streaming Analytics cluster configurations are part of Cloudera DataFlow for Data Hub and have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for more details.

You can access the default cluster definitions by clicking Environments, then selecting an environment and clicking the Cluster Definitions tab.

You can access the default cluster templates from Shared ResourcesCluster Templates.

To view details of a cluster definition or cluster template, click on its name. For each cluster definition, you can access a raw JSON file. For each cluster template, you can access a graphical representation ("list view") and a raw JSON file ("raw view") of all cluster host groups and their components.

Related Information

Cloudera DataFlow for Data Hub

Cloudera Runtime

Data Engineering clusters

Learn about the default Data Engineering clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Engineering provides a complete data processing solution, powered by Apache Spark and Apache Hive. Spark and Hive enable fast, scalable, fault-tolerant data engineering and analytics over petabytes of data.

Data Engineering cluster definition

This Data Engineering template includes a standalone deployment of Spark and Hive, as well as Apache Oozie for job scheduling and orchestration, Apache Livy for remote job submission, and Hue and Apache Zeppelin for job authoring and interactive analysis.

Cluster definition names

- Data Engineering for Azure
- Data Engineering HA - Spark3 for Azure

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering Spark3 for Azure

Cluster template name

- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: Apache Spark, Apache Hive, Apache Oozie."
- Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: HA: Apache Spark, Apache Hive, Apache Oozie."

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: The "Data Engineering: Apache Spark3" cluster template is deleted. Therefore, the "Data Engineering: Apache Spark3, Apache Hive, Apache Oozie" cluster template can be used instead.

Included services

- HDFS
- Hive
- Hue
- Livy
- Spark 3
- Yarn
- Zeppelin
- ZooKeeper
- Oozie is supported for Spark 3 as of Cloudera Runtime version 7.2.18

- Hive Warehouse Connector is supported as of Cloudera Runtime version 7.2.16.

Compatible Cloudera Runtime version

7.2.16, 7.2.17, 7.2.18

Topology of the Data Engineering cluster

Topology is a set of host groups that are defined in the cluster template and cluster definition used by Data Engineering. Data Engineering uses the following topology:

Host group	Description	Node configuration
Master	The master host group runs the components for managing the cluster resources including Cloudera Manager, Name Node, Resource Manager, as well as other master components such HiveServer2, HMS, Hue etc.	1 For Cloudera Runtime versions earlier than 7.2.14: Azure : Standard_D16_v3; StandardSSD_LRS - 100 GB For Cloudera Runtime versions 7.2.14+ DE, DE Spark3, and DE HA: Azure: Standard_D16_v3
Worker	The worker host group runs the components that are used for executing processing tasks (such as NodeManager) and handling storing data in HDFS such as DataNode.	3 For Cloudera Runtime versions earlier than 7.2.14: Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: Azure: Standard_D5_v2 DE HA: Azure: Standard_D5_v2
Compute	The compute host group can optionally be used for running data processing tasks (such as NodeManager). By default the number of compute nodes is set to 1 for proper configurations of YARN containers. This node group can be scaled down to 0 when there are no compute needs. Additionally, if load-based auto-scaling is enabled with minimum count set to 0, the compute nodegroup will be resized to 0 automatically.	0+ For Cloudera Runtime versions earlier than 7.2.14: Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: Azure: Standard_D5_v2 For Azure, the attached volume count for the compute host group is changed to 0. Only ephemeral/local volumes are used by default. DE HA:

Host group	Description	Node configuration
		Azure: Standard_D5_v2 For Azure, the attached volume count for the compute host group is changed to 0. Only ephemeral/local volumes are used by default. **Note:** Compute nodes run YARN and require storage only for temporary data - this requirement is fulfilled by instance storage, so making the attached volumes count to 0 by default is more cost-efficient.
Gateway	The gateway host group can optionally be used for connecting to the cluster endpoints like Oozie, Beeline etc. This nodegroup does not run any critical services. This nodegroup resides in the same subnet as the rest of the nodegroups. If additional software binaries are required they could be installed using recipes.	0+ Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB

Service configurations			
Master host group Cloudera Manager, HDFS, Hive (on Tez), HMS, Yarn RM, Oozie, Hue, DAS, Zookeeper, Livy, Zeppelin and Sqoop	Gateway host group Configurations for the services on the master node	Worker host group Data Node and YARN NodeManager	Compute group YARN NodeManager

Configurations

Note the following:

- There is a Hive Metastore Service (HMS) running in the cluster that talks to the same database instance as the Data Lake in the environment.
- If you use CLI to create the cluster, you can optionally pass an argument to create an external database for the cluster use such as Cloudera Manager, Oozie, Hue, and DAS. This database is by default embedded in the master node external volume. If you specify the external database to be of type HA or NON_HA, the database will be provisioned in the cloud provider. For all these types of databases the lifecycle is still associated with the cluster, so upon deletion of the cluster, the database will also be deleted.
- The HDFS in this cluster is for storing the intermediary processing data. For resiliency, store the data in the cloud object stores.
- For high availability requirements choose the Data Engineering High Availability cluster shape.

Architecture of the Data Engineering HA for Azure cluster

The Data Engineering HA for Azure cluster shape provides failure resilience for several of the Data Engineering HA services, including Knox, Oozie, HDFS, HS2, Hue, Livy, YARN, and HMS.

Services that do not yet run in HA mode include Cloudera Manager, DAS, and Zeppelin.

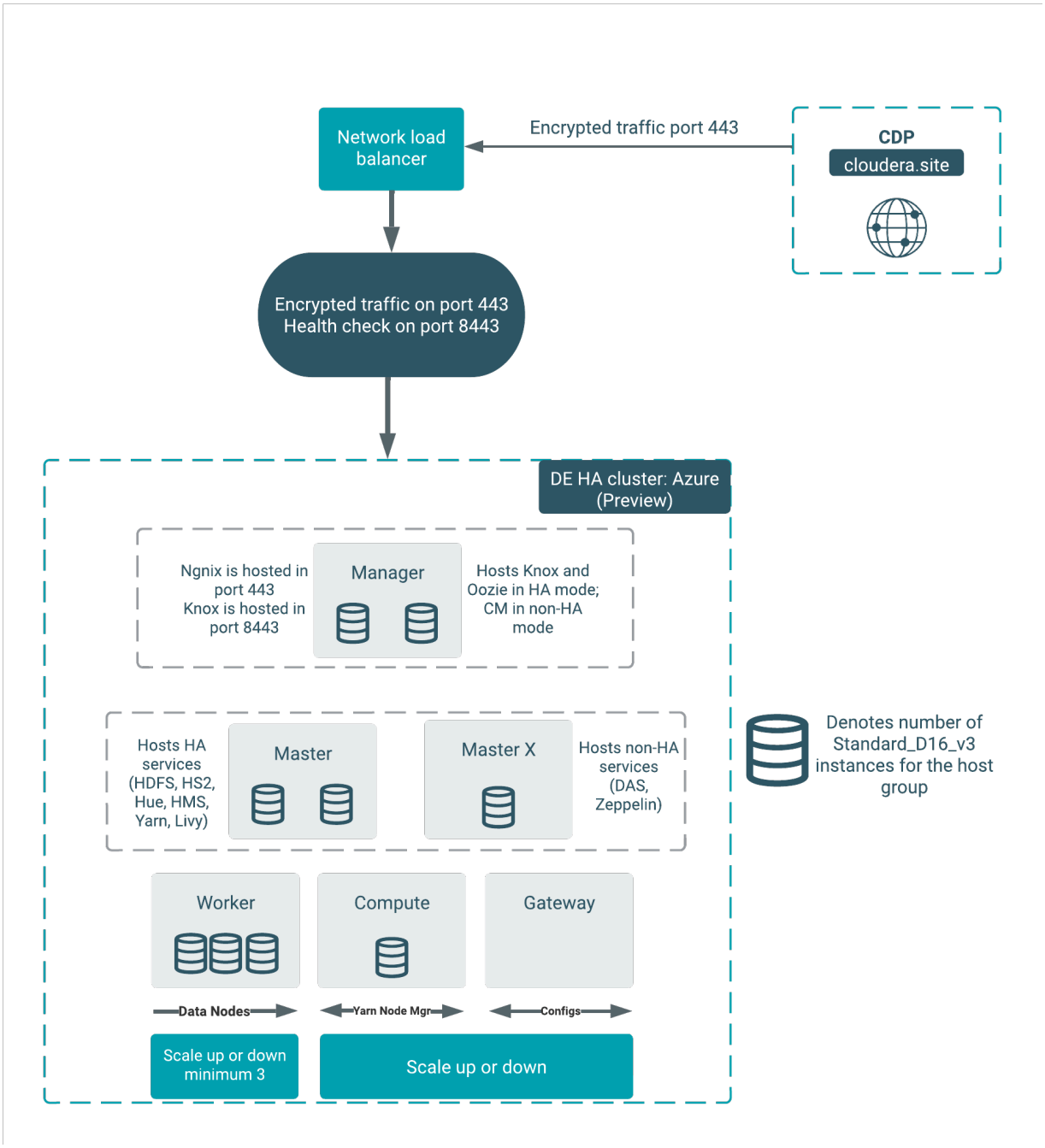

Component	Failure	User experience
Knox	One of the Knox services is down	External users will still be able to access all of the UIs, APIs, and JDBC.
Cloudera Manager	The first node in manager host group is down	The cluster operations (such as repair, scaling, and upgrade) will not work.

Cloudera Manager	The second node in the manager host group is down	No impact.
HMS	One of the HMS services is down	No impact.
Hue	One of the Hue services is down in master host group	No impact.
HS2	One of the HS2 services is down in the master host group	External users will still be able to access the Hive service via JDBC. But if Hue was accessing that particular service it will not failover to the other host. The quick fix for Hue is to restart Hue to be able to use Hive functionality.
YARN	One of the YARN services is down	No impact.
HDFS	One of the HDFS services is down	No impact.
Nginx	Nginx in one of the manager hosts is down	Fifty percent of the UI, API, and JDBC calls will be affected. If the entire manager node is down, there is no impact. This is caused by the process of forwarding and health checking that is done by the network load-balancer.
Oozie	One of the Oozie servers is down in the manager host group.	No impact for AWS and Azure as of Cloudera Runtime version 7.2.11. If you create a custom template for DE HA, follow these two rules: **1.** Oozie must be in single hostgroup. **2.** Oozie and Hue must not be in the same hostgroup.

Important: If you are creating a DE HA cluster through the CDP CLI using the create-azure-cluster command, note that there is a CLI parameter to provision the network load-balancer in HA cluster shapes. Make sure to use the [--enable-load-balancer | --no-enable-load-balancer] parameter when provisioning a DE HA cluster via the CLI. For more information see the CDP CLI reference.

Custom templates

Any custom DE HA template that you create must be forked from the default templates of the corresponding version. You must create a custom cluster definition for this with the JSON parameter “enableLoadBalancers”: true , using the create-aws/azure/gcp-cluster CLI command parameter --request-template. Support for pre-existing custom cluster definitions will be added in a future release. As with the template, the custom cluster definition must be forked from the default cluster definition. You are allowed to modify the instance types and disks in the custom cluster definition. You must not change the placement of the services like Cloudera Manager, Oozie, and Hue. Currently the custom template is fully supported only via CLI.

The simplest way to change the DE HA definition is to create a custom cluster definition. In the Create Data Hub UI when you click Advanced Options, the default definition is not used fully, which will cause issues in the HA setup.

Related Information

HDFS
Hive
Hue
Oozie
Spark
YARN
Zookeeper

Data Mart clusters

Learn about the default Data Mart and Real Time Data Mart clusters, including cluster definition and template names, included services, and compatible Runtime version.

Data Mart is an MPP SQL database powered by Apache Impala designed to support custom Data Mart applications at big data scale. Impala easily scales to petabytes of data, processes tables with trillions of rows, and allows users to store, browse, query, and explore their data in an interactive way.

Data Mart clusters

The Data Mart template provides a ready to use, fully capable, standalone deployment of Impala. Upon deployment, it can be used as a standalone Data Mart to which users point their BI dashboards using JDBC/ODBC end points. Users can also choose to author SQL queries in Cloudera's web-based SQL query editor, Hue, and run them with Impala providing a delightful end-user focused and interactive SQL/BI experience.

Cluster definition names

- Data Mart for Azure

Cluster template name

Cloudera - Data Mart: Apache Impala, Hue

Included services

- HDFS
- Hue
- Impala

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17, 7.2.18

Real Time Data Mart clusters

The Real-Time Data Mart template provides a ready-to-use, fully capable, standalone deployment of Impala and Kudu. You can use a Real Time Data Mart cluster as a standalone Data Mart which allows high throughput streaming ingest, supporting updates and deletes as well as inserts. You can immediately query data through BI dashboards using JDBC/ODBC end points. You can choose to author SQL queries in Cloudera's web-based SQL query editor, Hue. Executing queries with Impala, you will enjoy an end-user focused and interactive SQL/BI experience. This template is commonly used for Operational Reporting, Time Series, and other real time analytics use cases.

Cluster definition names

- Real-time Data Mart for Azure

Cluster template name

Cloudera - Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 2
- Yarn

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster definition names

- Real-time Data Mart - Spark3 for Azure

Cluster template name

Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark3

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 3
- Yarn

Compatible Cloudera Runtime versions

7.2.16, 7.2.17, 7.2.18

High availability

Cloudera recommends that you use high availability (HA), and track any services that are not capable of restarting or performing failover in some way.

Impala HA

The Impala nodes offer high availability. The following Impala services are not HA.

- Catalog service
- Statestore service

Kudu HA

Both Kudu Masters and TabletServers offer high availability.

Related Information

HDFS
Hue
Impala
Kudu
Spark
YARN

Operational Database clusters

The Operational Database (OpDB) template is removed from the Cloudera Data Hub. You can access the Cloudera Operational Database instead as a superior product.

The Cloudera Operational Database is a NoSQL database powered by Apache HBase designed to support custom OLTP applications that want to leverage the power of BigData. Apache HBase is a NoSQL, scale-out database that can easily scale to petabytes and stores tables with millions of columns and billions of rows.

Cloudera Operational Database also contains Apache Phoenix which provides a way to use HBase through an SQL interface.

Cloudera recommends you to use the Cloudera Operational Database to create Operational Database clusters.

Related Information

Cloudera Operational Database
Getting started with Operational Database
Before you create an Operational Database cluster
Creating an Operational Database cluster
HDFS
HBase
Knox
Zookeeper
Phoenix

Streams Messaging clusters

Learn about the default Streams Messaging clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streams Messaging provides the following features:

- Advanced messaging and real-time processing on streaming data using Apache Kafka
- Centralized schema management using Schema Registry
- Management and monitoring capabilities powered by Streams Messaging Manager
- Cross-cluster Kafka topic replication using Streams Replication Manger
- Kafka partition rebalancing with Cruise Control

This template sets up a fault-tolerant standalone deployment of Apache Kafka and supporting Cloudera components (Schema Registry, Streams Messaging Manager, Streams Replication Manager and Cruise Control), which can be used for production Kafka workloads in the cloud or as a disaster recovery instance for on-premises. Kafka clusters.

Note:

Streams Messaging clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streams Messaging cluster deployment
- Creating your first Streams Messaging cluster
- Connecting Kafka clients to Cloudera Public Cloud clusters

Cluster definition names

- Streams Messaging Heavy Duty for Azure
- Streams Messaging Light Duty for Azure
- Streams Messaging HA for Azure (Technical Preview)

Cluster template name

- Cloudera - Streams Messaging Heavy Duty
- Cloudera - Streams Messaging Light Duty
- Cloudera - Streams Messaging High Availability

Included services

- Kafka
- Schema Registry
- Streams Messaging Manager
- Streams Replication Manager
- Cruise Control
- Kafka Connect

Compatible Cloudera Runtime version

- 7.1.0 (Preview)
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streams Messaging cluster
Ingesting Data into Cloudera Public Cloud
Kafka
Schema Registry
Streams Messaging Manager
Streams Replication Manager

Flow Management clusters

Learn about the default Flow Management clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime versions.

Flow Management delivers high-scale data ingestion, transformation, and management to enterprises from any-to-any environment. It addresses key enterprise use cases such as data movement, continuous data ingestion, log data ingestion, and acquisition of all types of streaming data including social, mobile, clickstream, and IoT data.

The Flow Management template includes a no-code data ingestion and management solution powered by Apache NiFi. With NiFi's intuitive graphical interface and 300+ processors, Flow Management enables easy data ingestion and movement between CDP services as well as 3rd party cloud services. NiFi Registry is automatically set up and provides a central place to manage versioned Data Flows.

Note:

Flow Management clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Flow Management cluster deployment
- Creating your first Flow Management cluster
- Security considerations for Flow Management clusters
- Using Apache NiFi to ingest data into Cloudera Public Cloud
- Using NiFi and NiFi Registry

Cluster definition names

- Flow Management Light Duty for Azure
- Flow Management Heavy Duty for Azure

Cluster template name

- CDP - Flow Management: Light Duty
- CDP - Flow Management: Heavy Duty

Included services

- NiFi
- NiFI Registry

Compatible Cloudera Runtime versions

- 7.1.0
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Flow Management cluster

Apache NiFi documentation

Apache NiFi Registry documentation

Streaming Analytics clusters

Learn about the default Streaming Analytics clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streaming Analytics offers real-time stream processing and stream analytics with low-latency and high scaling capabilities powered by Apache Flink.

Streaming Analytics templates include Apache Flink that works out of the box in stateless or heavy state environments. Beside Flink, the template includes its supporting services namely YARN, Zookeeper and HDFS. The Heavy Duty template comes preconfigured with RocksDB as state backend, while Light Duty clusters use the default Heap state backend. You can create your streaming application by choosing between Kafka, Kudu, and HBase as datastream connectors.

You can also use SQL to query real-time data with SQL Stream Builder (SSB) in the Streaming Analytics template. By supporting the SSB service in Cloudera Public Cloud, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. SSB is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

Note:

Streaming Analytics clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streaming Analytics cluster deployment
- Creating your first Streaming Analytics cluster
- Analyzing data using Apache Flink
- Querying data using SQL Stream Builder

Cluster definition names

- Streaming Analytics Light Duty for Azure
- Streaming Analytics Heavy Duty for Azure

Cluster template name

- 7.2.17 - Streaming Analytics Light Duty
- 7.2.17 - Streaming Analytics Heavy Duty

Included services

- Flink
- SQL Stream Builder
- YARN
- Zookeeper
- HDFS
- Kafka

Important: In the Streaming Analytics cluster templates, Kafka service is included by default to serve as a background service only for the websocket ouput and sampling feature of SQL Stream Builder. The Kafka service in the Streaming Analytics cluster template cannot be used for production, you need to use the Streams Messaging cluster template when Kafka is needed for your deployment.

Compatible Cloudera Runtime version

- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streaming Analytics cluster
Flink
YARN
Zookeeper
HDFS

Data Discovery and Exploration clusters

Learn about the default Data Discovery and Exploration clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Discovery and Exploration

Explore and discover data sets ad-hoc. Do relevance-based analytics over unstructured data (logs, images, text, PDFs, etc). Get started with search or log analytics. Make data more accessible to everyone with Data Discovery and Exploration.

Cluster Definition Names

- Data Discovery and Exploration for Azure

Cluster Template Name

- Data Discovery and Exploration

Included Services

- Solr
- Spark 2
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Versions

7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Related Information

Solr
Spark
HDFS
Hue
YARN
Zookeeper

Create a cluster from a definition on Azure

You can quickly create clusters from default or custom cluster definitions within an existing Azure environment.

Before you begin

To create a Cloudera Data Hub cluster on Azure, you must have an existing Azure environment. Also, you should make sure that the Runtime version of the Data Lake cluster matches the Cloudera Runtime version of the Cloudera Data Hub cluster that you are about to create; If these versions don't match, you may encounter warnings and/or errors.

About this task

Required role: DataHubCreator, EnvironmentAdmin at the scope of the environment where the Cloudera Data Hub cluster is running, or Owner of the environment

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Environments > click on an environment where you would like to create a cluster > click Create Data Hub. The following page is displayed:

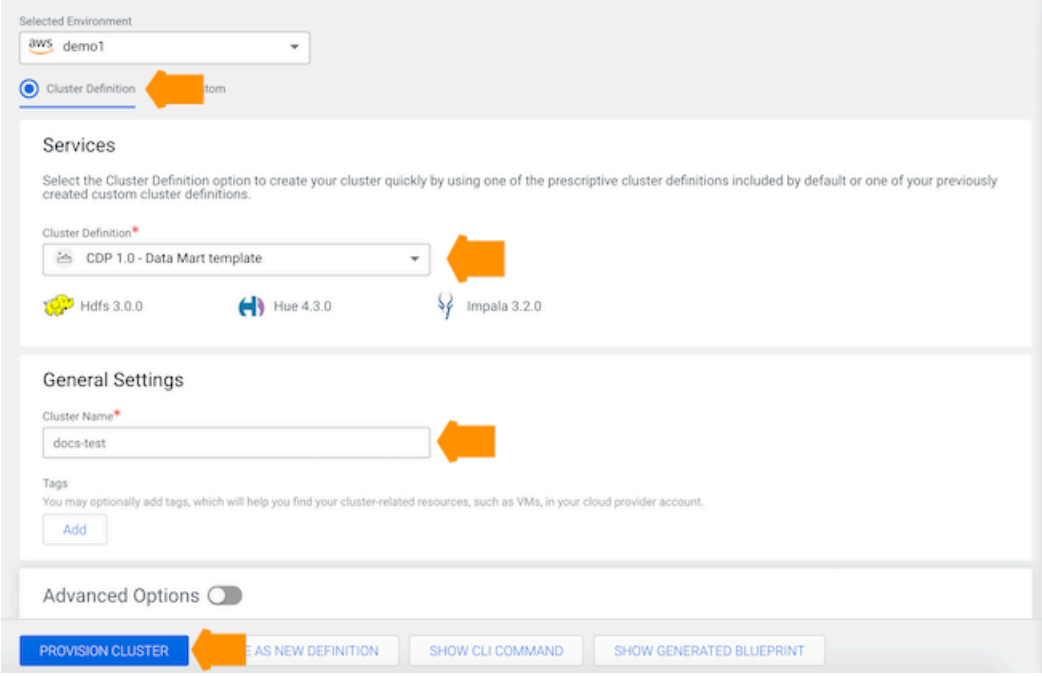

3. Select Cluster Definition.
4. From the Cluster Definition dropdown, select the cluster definition that you would like to use for your cluster.

   The cluster template referenced in the selected cluster definition determines which services are included in the cluster. The list of services is automatically shown below the selected cluster definition name:
5. Specify General Settings for your cluster:

Parameter	Description
Cluster Name	Enter a name for your cluster. The name must be between 5 and 40 characters, must start with a letter, and must only include lowercase letters, numbers, and hyphens.
Tags	(Optional) Add tags that Data Hub should use to tag your Azure resources. Click Add to add a tag, and then enter a key and value for each tag. Repeat the steps if you would like to add more tags. For more information about tags, refer to Tags.

6. Optionally, click on Advanced Options to modify advanced cluster settings. For more information on these options, refer to *Advanced cluster options*.
7. On AWS and Azure only: Optionally, when you have finished providing the cluster settings, you can click the Show CLI Command button at the bottom of the page to review or copy the CDP CLI command used to create the cluster. You can copy the command from the pop-up window that appears, either to provision the cluster later or for use in scripts.
8. You also have the option to review or copy the cluster template that is generated and will be used in cluster creation. Click the Show Generated Cluster Template button at the bottom of the page.
9. To proceed with cluster provisioning immediately, click on Provision Cluster.
10. You will be redirected to the Cloudera Data Hub cluster dashboard, and a new tile representing your cluster will appear at the top of the page.

    The following messages are written to the event history as your cluster is being created:

```
Cluster built; Cluster manager ip:10.97.82.237
8/2/2019, 6:10:44 PM
```

```
Updating Cluster Proxy service with gateway configuration
8/2/2019, 6:10:43 PM
Building cluster; Cluster manager ip:10.97.82.237
8/2/2019, 5:45:12 PM
Starting cluster services
8/2/2019, 5:42:33 PM
Mounting attached disks
8/2/2019, 5:42:18 PM
Setting up infrastructure metadata
8/2/2019, 5:42:14 PM
Bootstrapping infrastructure cluster
8/2/2019, 5:41:59 PM
Registering cluster with Cluster Proxy service
8/2/2019, 5:41:57 PM
Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Billing started, Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Infrastructure metadata collection finished
8/2/2019, 5:41:55 PM
Infrastructure creation took 96 seconds
8/2/2019, 5:41:52 PM
Creating infrastructure
8/2/2019, 5:40:16 PM
Setting up CDP image
8/2/2019, 5:40:15 PM
```

11. When your cluster is ready, its status will change to Running.

What to do next

You can access links to Cloudera Manager, and other cluster UIs and endpoints from cluster details.

Related Information

Advanced cluster options

Accessing a Cluster

Create a custom cluster on Azure

Create a custom Cloudera Data Hub cluster within an existing Azure environment.

Before you begin

To create a Cloudera Data Hub cluster on Azure, you must have an existing Azure environment. Also, you should make sure that the Cloudera Runtime version of the Data Lake cluster matches the Runtime version of the Cloudera Data Hub cluster that you are about to create; If these versions don't match, you may encounter warnings and/or errors.

https://www.youtube.com/embed/pGHHW9nOrPU

About this task

Required role: DataHubCreator, EnvironmentAdmin at the scope of the environment where the Data Hub is running, or Owner of the environment

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Environments > click on an environment > click Create Data Hub.

3. Under Selected Environment, confirm that the selected environment is the one where you would like to create your cluster.
4. Select Custom.
5. Under Platform Version, current Cloudera Runtime version is pre-selected.
6. Under Cluster template, select the cluster template that you would like to use.

   The selected cluster template determines which services are included in the cluster. Select the cluster template and the list of services is automatically shown below it:

   For more information about cluster templates, refer to Cluster templates.
7. Specify General Settings for your cluster:

Parameter	Description
Cluster Name	Enter a name for your cluster. The name must be between 5 and 40 characters, must start with a letter, and must only include lowercase letters, numbers, and hyphens.
Tags	(Optional) Add tags that Data Hub should use to tag your Azure resources. Click Add to add a tag, and then enter a key and value for each tag. Repeat the steps if you would like to add more tags. For more information about tags, refer to Tags.

8. Optionally, click on Advanced Options to modify advanced cluster settings. For more information on these options, refer to LINK.
9. Once done, click on Provision Cluster.
10. You will be redirected to the Cloudera Data Hub cluster dashboard, and a new tile representing your cluster will appear at the top of the page.

    The following messages are written to the event history as your cluster is being created:

```
Cluster built; Cluster manager ip:10.97.82.237
8/2/2019, 6:10:44 PM
Updating Cluster Proxy service with gateway configuration
8/2/2019, 6:10:43 PM
Building cluster; Cluster manager ip:10.97.82.237
8/2/2019, 5:45:12 PM
Starting cluster services
8/2/2019, 5:42:33 PM
Mounting attached disks
8/2/2019, 5:42:18 PM
Setting up infrastructure metadata
8/2/2019, 5:42:14 PM
Bootstrapping infrastructure cluster
8/2/2019, 5:41:59 PM
Registering cluster with Cluster Proxy service
8/2/2019, 5:41:57 PM
Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Billing started, Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Infrastructure metadata collection finished
8/2/2019, 5:41:55 PM
```

```
Infrastructure creation took 96 seconds
8/2/2019, 5:41:52 PM
Creating infrastructure
8/2/2019, 5:40:16 PM
Setting up CDP image
8/2/2019, 5:40:15 PM
```

11. When your cluster is ready, its status will change to Running.

What to do next

You can access links to Cloudera Manager, and other cluster UIs and endpoints from cluster details.

Related Information

Advanced cluster options

Accessing a Cluster

Create a cluster with an external database

Through the CDP CLI, you can configure an external, durable database when you create a Data Hub cluster. An external database keeps a persistent state for cluster services like Hue, DAS and Zeppelin in case of instance failures. You also have the option to configure an external database as a custom property within a cluster definition.

A top-level custom property for the external managed database is available to be set within a cluster definition:

```
"externalDatabase": {
  "availabilityType": "NONE" | "NON_HA" | "HA"
```

For more information on using custom cluster definitions to create clusters, see *Cluster definitions*.

A new flag, called --datahub-database, is available in the CDP CLI create-azure-cluster command. The supported values are NONE, NON_HA, and HA. To create a Data Hub cluster with an external managed database, use the following CLI options:

```
create-azure-cluster
          [--cluster-name <value>]
          [--cluster-definition-name <value>]
          [--environment-name <value>]
          [--cluster-template-name <value>]
          [--instance-groups <value>]
          [--subnet-id <value>]
          [--image <value>]
          [--tags <value>]
          [--request-template <value>]
          [--datahub-database <value>]
          [--cli-input-json <value>]
          [--generate-cli-skeleton]
```

- --cluster-name: String. The name of the cluster. This name must be unique, must have between 5 and 40 characters, and must contain only lowercase letters, numbers and hyphens. Names are case-sensitive.
- --cluster-definition-name: String. The name or CRN of the cluster definition to use for cluster creation.
- --environment-name: Name or CRN of the environment to use when creating the cluster. The environment must be an Azure environment.
- --cluster-template-name: String. Name or CRN of the cluster template to use for cluster creation.
- --instance-groups: Array. Instance group details.
- --subnet-id: String. The subnet ID.
- --image: Object. The details of the image used for cluster instances.
- --tags: Array. Tags to be added to Data Hub-related resources.

- --request-template: String. JSON template to use for cluster creation. This is different from the cluster template and would be removed in the future.
- --datahub-database: String. Database type for the Data Hub cluster. Currently supported values: NONE, NON_HA, HA
- --cli-input-json: String. Performs service operation based on the JSON string provided. The JSON string follows the format provided by --generate-cli-skeleton. If other arguments are provided on the command line, the CLI values will override the JSON-provided values.
- --generate-cli-skeleton: Prints a sample input JSON to standard output. Note the specified operation is not run if this argument is specified. The sample input can be used as an argument for --cli-input-json.

For information on shorthand and JSON syntax of these options, and the output of the command, invoke the CDP CLI help.

Note: The external managed database is destroyed along with the Data Hub cluster.

Related Information

Cluster definitions

Default cluster configurations

Cloudera DataFlow for Data Hub

Cloudera Runtime

Create a cluster from an existing cluster

You can copy the CDP CLI command used to create a Cloudera Data Hub cluster on AWS or Azure and then use the command to create a new cluster with the same or similar settings.

Procedure

1. Click Data Hub Clusters in the left navigation menu and then click the tile of the Cloudera Data Hub cluster that you want to use to create the new cluster.
2. From the Data Hub details page, click the Actions menu in the top right of the page and then Show CLI Command.
3. Copy the CDP CLI command used for the cluster creation.
4. Before you use the command in the CDP CLI to create a new cluster, ensure that you change the --cluster-name parameter as this must be a unique name.

Results

You can now use the CDP CLI to run the command to create a new cluster.

Related Information

CDP CLI client setup

Creating a cluster with a specific Java version

When you create a new Cloudera Data Hub or Data Lake cluster, you can specify Java 8 or Java 11 as the default JDK.

Java 11

When you launch a new Cloudera Data Hub or Data Lake cluster, the default JDK version is Java 8 unless you specify Java 11. If you specify Java 11 as the default JDK version, Cloudera Manager and all Cloudera Runtime

services (as well as all workloads launched) will use pre-installed Java 11 dependencies rather than Java 8. This may be useful, for instance, when you have workloads such as Spark 3 jobs that are written in Java 11, when your workloads have Java 11 dependencies, or if you are running third-party components on a cluster that require Java 11.

The ability to launch new Cloudera Data Hub and Data Lake clusters with Java 11 is currently available only through the CDP CLI. Include the parameter --java-version in the create-<cloud_provider>-datalake and create-<cloud_provider>-datahub CLI commands. Valid inputs for the --java-version parameter are 8 and 11.

Launching clusters with Java 11 as the default is supported from Cloudera Runtime versions 7.2.15.11 and 7.2.16.200 and above. Creating clusters with the Java version property set to 11 on older Cloudera Runtime versions will be blocked by an internal validation. Currently, the Java version selection is only available when creating a new cluster.

Java 8 and 11 compatibility

Java 11 introduces breaking changes, which could lead to incompatibility with your current workloads or custom/third-party Java-based applications deployed on Cloudera Data Hub clusters.

Cloudera recommends that all existing production workloads are tested before moving them over to a Cloudera Data Hub cluster that is set to use Java 11 as the default JDK.

Note that Data Lake and Cloudera Data Hub hosts will continue to have both Java 8 and 11 pre-installed. The --java-version switch controls which Java versionn should be set as the OS default during bootstrapping of the hosts. Therefore, it is possible to continue to run jobs written for Java 8 even if the OS default is Java 11. Some services (for example, Spark) make this possible by setting the JAVA_PATH environment variable to point to the Java 8 path /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.[...].x86_64/jre/bin/java.

Advanced cluster options

In the create cluster wizard, click on Advanced Options to view the advanced cluster configuration options.

https://www.youtube.com/embed/Q8Xjb3-8KFk

While some of these advanced options can be configure in the wizard, others require prior setup.

The following options are available:

Tags

You can define tags that will be applied to your cluster-related resources (such as VMs) on your cloud provider account.

The tags added during cluster creation are displayed in your cloud account on the resources that Cloudera Data Hub provisioned for your clusters. You can use tags to categorize your cloud resources by purpose, owner, and so on. Tags come in especially handy when you are using a corporate cloud provider account and you want to quickly identify which resources belong to your cluster(s). In fact, your corporate cloud account admin may require you to tag all the resources that you create, in particular resources, such as VMs, which incur charges.

By default, the following tags are created:

Tag	Description
Cloudera-Resource-Name	The workload-appropriate Cloudera resource name. This CRN serves as a unique identifier for the resource over time.
Cloudera-Creator-Resource-Name	Cloudera resource name of the Cloudera user that created the resource.
Cloudera-Environment-Resource-Name	The name of the environment with which the resource is associated.

You can optionally add additional tags. To add custom tags:

1. In the create cluster wizard, navigate to the General Configuration page.

2. Specify your tags in the Tags section by providing a key and value for each tag.

Note:

It is not possible to add tags via Cloudera Data Hub after your cluster has been created. In this case, you can only add the tags manually via your cloud provider's interface.

To learn more about tags and their restrictions, refer to Azure documentation.

Related Information

Use tags to organize your Azure resources (Azure)

Image catalog

The options on the "Image Settings" page of the advanced create cluster wizard allow you to select custom image settings.

By default, Cloudera Data Hub uses the prewarmed image from the image catalog provided in Cloudera Data Hub. If necessary, you can also customize a default image.

Choose image catalog

Cloudera Data Hub uses the image catalog provided by default. If you would like to use a custom image catalog instead of the default image catalog, you must first prepare your custom images, then create and register an image catalog.

Choose image type

By default, Cloudera Data Hub uses the included prewarmed images with the default Cloudera Manager and Cloudera Runtime version, but you can select a different prewarmed image or customized prewarmed image to use for your cluster. Cloudera Data Hub currently supports the following types of images for launching clusters:

Image type	Description	Default images provided
Prewarmed Image	By default, Cloudera Data Hub launches clusters from prewarmed images. Prewarmed images include the operating system as well as Cloudera Manager and Cloudera Runtime. The Cloudera Manager and Cloudera Runtime version used on prewarmed images cannot be customized.	Yes
Custom Image	You can customize a default image for compliance or security reasons. You can then use the CDP CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create a Cloudera Data Hub cluster.	

Choose image

This option allows you to select a different image.

Related Information

Image catalog updates

Custom images and image catalogs

If necessary, you can use a custom Cloudera Runtime or FreeIPA image for compliance or security reasons. You can then use the Cloudera CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create a Data Lake, Cloudera Data Hub cluster or environment with a custom FreeIPA image.

Overview

A custom image should inherit most of its attributes from its source image, which is a default image that you select from the cdp-default image catalog.

The typical method of creating a Data Lake or Cloudera Data Hub picks up the latest pre-warmed image from the cdp-default image catalog for the specified version of Cloudera Runtime. These default images are pre-warmed VM images that contain a base URL to the default parcels in the Cloudera archive, amongst other configurations. If the default pre-warmed images do not suit your business needs, you can specify that the Data Lake, Cloudera Data Hub cluster or the environment (in the case of FreeIPA) uses a custom image instead.

What is a custom image?

A custom image is an entry in a custom image catalog that inherits most of its attributes from a source (default) image.

Custom image entries have:

- An image type: Cloudera Runtime [which includes Cloudera Data Hub and Data Lake images] or FreeIPA
- A source image ID that points to an image in the cdp-default image catalog
- A timestamp of creation
- An option to specify a VM region and image reference (such as an AMI ID) if you are overriding the source image with a custom VM image
- An option to override the parcel base URL

Why use a custom image?

You might require a custom image for compliance or security reasons (a “hardened” image), or to have your own packages pre-installed on the image, for example monitoring tools or software. You might also want to specify a custom image if you need to use a default image with a specific Runtime maintenance version applied, rather than simply specifying the latest major Runtime version.

Note: When customizing VM images, note that certain customizations (for example, CIS hardening rules) may not be compatible with Cloudera.

What can you customize?

In a custom image entry, you can override the VM images themselves with your own custom images that are sufficiently hardened. Importantly, you should only customize a default image from the cdp-default catalog as opposed to creating one from scratch. You can also override the default parcel base URL (at archive.cloudera.com) with your own host site.

What is a custom image catalog?

A custom image catalog is simply a catalog that holds custom images. A custom image catalog can contain a single or multiple custom image entries.

Custom image catalogs have:

- A name. The name is a unique identifier and is used to refer to the catalog during environment, Data Lake, and Cloudera Data Hub cluster creation; as well as during catalog operations like creating an image.

- A description.
- An owner. The owner is the user who runs the command to create the catalog.

What is the process for creating a custom image and catalog?

- If you are replacing the VM images in a custom image entry with a customized version, you should first prepare the image by modifying an official Cloudera default image, which you can find under Shared Resources > Image Catalogs > cdp-default.
- Select a source image from the cdp-default image catalog to be the source of customization. When you run the CLI command to find a default image, you specify the Runtime version, provider, image type, or a combination of the three.
- Create a custom image catalog, or identify an existing catalog where you want to save the custom image entry.
- Apply the necessary changes to the custom image entry, like the override AMI IDs with the new, customized AMIs; or add a new parcel base URL using the `--base-parcel-url` command when you set the custom image.
- You can then create an environment, Data Lake, or Cloudera Data Hub cluster, based on custom catalogs via the CDP CLI.

Creating a custom image and image catalog with the CDP CLI

You can create a custom Cloudera Runtime or FreeIPA image and image catalog through the CDP CLI.

Before you begin

If you are replacing the VM images in a custom image entry with a customized version, you must first prepare the image by modifying an official Cloudera default image, which you can find under Shared Resources Image Catalogs cdp-default . Take note of the image reference, such as the AMI ID.

Procedure

1. To find a source image from the cdp-default catalog that you want to use as the source of your custom image entry, run the following command:

   ```
   cdp imagecatalog find-default-image --provider <cloud provider> --image-
   type <image type> --runtime-version <Cloudera Runtime version>
   ```

   For example:

   ```
   cdp imagecatalog find-default-image --provider AWS --image-type runtime --
   runtime-version 7.2.12
   ```

2. A custom image requires a custom catalog. If you haven't yet created a custom catalog, or if you want to create a new one for a new custom image, run the following command:

   ```
   cdp imagecatalog create-custom-catalog --catalog-name <unique catalog na
   me> --description <catalog description>
   ```

   For example:

   ```
   cdp imagecatalog create-custom-catalog --catalog-name my custom catalog --
   description test catalog
   ```

3. Within the custom image catalog that you created (or an older custom catalog that you want to use), create a custom image entry with the selected source image marked as its source. Providing your own VM images is optional. Run the following command:

   ```
   cdp imagecatalog set-<image-type>-image --catalog-name <name of the custom
    catalog> --vm-images region=<region of customized image>,imageReference=<
   ```

```
cloud provider specific ID of a customized image> --source-image-id <cdp
 image ID of source image>
```

For example:

```
cdp imagecatalog set-runtime-image --catalog-name my custom catalog --vm-
images region=eu-central-1,imageReference=ami-7torotmhqi6q7438y --source-
image-id 8t4y9853-12b6-3n6z-75dh-tx775k4c793w
```

4. You can then apply the necessary changes to the custom image entry, like overriding AMI IDs with new, customized ones, or adding a new parcel base URL.

Command	Description
cdp imagecatalog find-default-image	Finds the default images in the cdp-default image catalog for a specified version of Cloudera Runtime. Parameters: • --provider • --image-type • --runtime-version You can provide any combination of these parameters.
cdp imagecatalog create-custom-catalog	Creates a custom catalog. Parameters: • --catalog-name • --description –catalog-name is required.
cdp imagecatalog set-runtime-image or cdp imagecatalog set-freeipa-image	Creates a custom image entry (either Cloudera Runtime [Cloudera Data Hub/Data Lake] or FreeIPA) within the specified catalog. Parameters: • --catalog-name • --vm-images [region,imageReference] • --source-image-id • --image-id • --base-parcel-url --catalog-name and --source-image-id are required.

Switching image catalogs

You can switch the image catalog of an already existing Cloudera Data Hub, Data Lake, or FreeIPA cluster. You may want to switch the image catalog for a cluster in order to restrict which Cloudera Runtime version can be upgraded to, or in order to move to custom images for an existing cluster.

Use the following CDP CLI commands to switch the image catalog for an existing cluster:

- FreeIPA:

```
cdp environments set-catalog --environment $ENVIRONMENT_NAME --catalog $
CATALOG
```

Parameter	Description
--environment	Name or CRN of the environment that holds the FreeIPA installation.
--catalog	URL of the FreeIPA catalog to be used.

- Cloudera Data Hub:

```
cdp datahub set-catalog --cluster $DATAHUB_NAME --catalog-name $CATALOG
```

Parameter	Description
--cluster	Name or CRN of the Cloudera Data Hub cluster for which you want to use the new image catalog.
--catalog-name	Name of the image catalog to be used. The image catalog must be a JSON based catalog, and switching is only possible from catalogs that are JSON based.

- Data Lake:

```
cdp datalake set-catalog --datalake $DATALAKE_NAME --catalog-name $CATALOG
```

Parameter	Description
--datalake	Name of the Data Lake for which you want to use the new image catalog.
--catalog-name	Name of the image catalog to be used. The image catalog must be a JSON based catalog, and switching is only possible from catalogs that are JSON based.

Network and availability

The "Network and availability" options allow you to customize the networking settings of your cluster.

On the Network and Availability page, provide the following to specify the networking resources that will be used for your cluster:

Parameter	Description
Select Subnet	If your environment includes a single subnet, that subnet is preselected and it cannot be changed. If your environment includes more than one subnet, you can select the subnet(s) in which your cluster will be provisioned.
Select Azure Database Server	A Cloudera Data Hub cluster uses the same Flexible Server or Single Server settings as the environment in which it runs but you can choose to enable a Flexible server on Cloudera Data Hub clusters running in an environment that uses a Single server. For more information, see Using Azure Database for PostgreSQL Flexible Server.

Hardware and storage

The "Hardware and storage" options allow you to customize the cloud provider specific cluster hardware and storage options.

The Hardware and Storage options can be selected for each host group. To edit this section for a specific host group, click on the . When done editing, click on the to save the changes. Repeat for these steps for all host groups that you would like to edit.

The following hardware and storage settings are available:

Parameter	Description
Cloudera Manager Server	You must select one node for Cloudera Manager Server by clicking the button. The “Instance Count” for that host group must be set to “1”. If you are using one of the default cluster templates, this is set by default.
Instance Type	Select an instance type. For information about instance types on Azure refer to General Purpose Virtual Machine Sizes in Azure documentation.
Instance Count	Enter the number of instances of a given type. Default is 1.
Storage Type	Select the volume type. The options are: (1) Locally-redundant storage, (2) Geo-redundant storage, (3) Premium locally-redundant storage. For more information about these options refer to Introduction to Azure Storage in Azure documentation.
Attached Volumes Per Instance	Enter the number of volumes attached per instance. Default is 1.
Volume Size	Enter the size in GB for each volume. Default is 100.
Root Volume Size	This option allows you to increase or decrease the root volume size. Default is 200 GB. This option is useful if your custom image requires more space than the default 200 GB. If you change the value of the root volume size, an osDisk with the given rootVolumeSize will be created for the instance automatically; however, you will have to manually resize the osDisk partition by using the steps provided in How to Resize Linux osDisk Partition on Azure in Azure documentation. If you use a custom Cloudera Data Hub template specifying a root volume size smaller than 200GB, you may encounter an error.
Availability Sets	To support fault tolerance for VMs, Azure uses availability sets. This allows two or more VMs to be mapped to multiple fault domains, each of which defines a group of virtual machines that share a common power source and a network switch. When adding VMs to an availability set, Azure automatically assigns each VM a fault domain. The SLA includes guarantees that during OS Patching in Azure or during maintenance operations, at least one VM belonging to a given fault domain will be available. In Cloudera, no availability sets are configured by default. To have availability sets configured, make sure to navigate to the "Hardware and Storage" tab in the Advanced Options during Cloudera Data Hub cluster creation. In this case, one availability set is configured during cluster creation for each host group with “Instance Count” that is set to 1 or larger. The assignment of fault domains is automated by Azure, so there is no option for this in Cloudera web UI. **Note:** Although there is no specific option in the "Hardware and Storage" tab to enable availability sets, just navigating to this page will cause the availability sets to be configured. On that page, one can observe the availability set names present for each group. This and other availability set parameters for the availability sets cannot be configured. After the deployment is finished, you can check the layout of the VMs inside an availability set on Azure Portal. You will find the “Availability set” resources corresponding to the host groups inside the deployment’s resource group.
Availability Set Name	Choose a name for the availability set that will be created for the selected host group. The following convention is used: “clustername-hostgroupname-as”.
Fault Domain Count	The number of fault domains. Default is 2 or 3, depending on the setting supported by Azure.
Update Domain Count	This number of update domains. This can be set to a number in range of 2-20. Default is 20.

Cloud storage

The options on the "Cloud Storage" page allow you to optionally specify the base storage location used for YARN and Zeppelin.

During environment creation under Data Access > Storage Location Base you configure a default ADLS Gen2 base storage location for the environment, and all Cloudera Data Hub clusters created within that environment use this location. The Cloud Storage options in the Cloudera Data Hub cluster wizard allow you to additionally specify a different location for YARN application logs and Zeppelin Notebook's root directory:

- Existing Base Storage Location - By default, this is set to the Storage Location Base configured on environment level. If you do not want to make any changes, simply leave this blank. If you would like to use a different location for YARN application logs and Zeppelin Notebook's root directory, you can specify a different ADLS Gen2 location. Note that the specified ADLS Gen2 location must exist prior to Cloudera Data Hub cluster creation and that you must adjust the IAM policies created during environment's cloud storage setup to make sure that IDBroker has write access to this location.
- Path for YARN Application Logs property - This directory structure gets created automatically during cluster creation. You can customize it if you would like it be different than what is suggested by default.
- Path for Zeppelin Notebooks Root Directory property - This directory structure gets created automatically during cluster creation. You can customize it if you would like it to be different than what is suggested by default.

Related Information

Minimal setup for cloud storage

Recipes

The "Recipes" option allows you to select previously uploaded recipes (scripts that can be run pre or post cluster deployment) for each host group.

This option is available from the Cluster Extensions page of the advanced create cluster wizard. For more information, refer to the documentation linked below.

Related Information

Recipes

Cluster template overrides

You can specify custom configurations that override or append the properties in a built-in Cloudera Data Hub template or a custom template.

Overview

You can launch Cloudera Data Hub clusters from a set of pre-defined cluster templates created for prescriptive use cases. These cluster templates are a “shared resource” that define the list of services that will be installed on the Cloudera Data Hub, including their configurations.

For example, this is a portion of a default template:

```
{
  "services": [
    {
      "refName": "zookeeper",
      "serviceType": "ZOOKEEPER",
      "serviceConfigs": [
        {
          "name": "service_config_suppression_server_count_validator",
          "value": "true"
```

```
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "zookeeper-SERVER-BASE",
          "roleType": "SERVER",
          "base": true
        }
      ]
    },
    {
      "refName": "hdfs",
      "serviceType": "HDFS",
      "serviceConfigs": [
        {
          "name": "hdfs_verify_ec_with_topology_enabled",
          "value": false
        },
        {
          "name": "core_site_safety_valve",
          "value": "<property><name>fs.s3a.buffer.dir</name><value>${env.LOC
AL_DIRS:-${hadoop.tmp.dir}}/s3a</value></property><property><name>fs.s3a.com
mitter.name</name><value>directory</value></property>"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "hdfs-NAMENODE-BASE",
          "roleType": "NAMENODE",
          "base": true,
          "configs": [
            {
              "name": "role_config_suppression_namenode_java_heapsize_mini
mum_validator",
              "value": "true"
            },
            {
              "name": "role_config_suppression_fs_trash_interval_minimum_
validator",
              "value": "true"
            },
            {
              "name": "fs_trash_interval",
              "value": "0"
            },
            {
              "name": "fs_trash_checkpoint_interval",
              "value": "0"
            },
            {
              "name": "erasure_coding_default_policy",
              "value": " "
            }
          ]
        },
        {
          "refName": "hdfs-SECONDARYNAMENODE-BASE",
          "roleType": "SECONDARYNAMENODE",
          "base": true
        },
        {
          "refName": "hdfs-DATANODE-BASE",
          "roleType": "DATANODE",
          "base": true
```

```
        },
        {
          "refName": "hdfs-BALANCER-BASE",
          "roleType": "BALANCER",
          "base": true
        },
        {
          "refName": "hdfs-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true,
          "configs": [
            {
              "name": "dfs_client_use_trash",
              "value": false
            },
            {
              "name": "role_config_suppression_hdfs_trash_disabled_validat
or",
              "value": "true"
            },
            {
              "name": "hdfs_client_env_safety_valve",
              "value": "HADOOP_OPTS=\"-Dorg.wildfly.openssl.path=/usr/lib64
 ${HADOOP_OPTS}\""
            }
          ]
        }
      ]
```

In this section of an example cluster template, two primary types of service configurations are visible: serviceConfigs and configurations for various roleConfigGroups, in addition to a special type of service configuration called safety_valves. You can read about the details of these configs in the *Cloudera Manager Configuration Properties* reference.

Oftentimes you may want to modify the serviceConfigs, roleConfigGroups, and safety_valves present in a default Cloudera Data Hub cluster template. Although you can create a custom cluster template by modifying the JSON of a default template, this process can be cumbersome and error-prone.

As an alternative to creating a custom template, you can specify custom configurations that override or append the properties in a default template. These custom configurations are saved as a shared resource called “cluster template overrides,” and can be used and re-used across Cloudera Data Hub clusters in different environments. As a shared resource, they do not need to be attached to a specific Cloudera Runtime version.

By using a default cluster template along with a cluster template override, you can create a customized Cloudera Data Hub cluster, along with receiving improvements to the default templates that are present in newer Cloudera Runtime releases. Cluster template overrides can be used to override or append properties present in both the identified types of service configurations, as well as safety valves.

Important: If you customize safety valves, the provided value will be appended to the already existing value for that safety valve. Custom configs will not override the existing value.

You are not limited to customizing the configs present in a default cluster template; you can add any valid configuration recognized by Cloudera Manager for an included service. New configurations are appended to the cluster template when the cluster is launched.

Cluster template overrides can be applied to custom templates as well. If you want to apply a cluster template override to a custom template that contains properties that are dynamically replaced during cluster creation, the cluster template override will override any dynamically-replaced properties when the two conflict.

Limitations

At present, there is no way to validate any individual property to ensure that it is valid and recognised by Cloudera Manager, so that it can be overridden to the desired value. Adding an incorrect config name can lead to errors while

installing the Cloudera Manager template on the Cloudera Data Hub cluster, or the property could be ignored by Cloudera Manager entirely.

An invalid value for a particular property can also cause errors during the cluster creation process. Carefully review the *Cloudera Manager Configuration Properties* reference in regards to configurations that you want to customize.

Cluster template overrides are only for overriding/appending the serviceConfigs, roleConfigGroups, and safety_v alves in a Cloudera Data Hub template. Unlike creating a custom template, you can not add a service to the template.

Related Information

Cloudera Manager Configuration Properties

Building a custom Cloudera Data Hub cluster template

Custom Properties

Creating a cluster template override

From the Shared Resources menu, you can create a cluster template override to overwrite the default values of service and role configs in a Cloudera Data Hub template.

Procedure

1. In the Cloudera Management Console UI, Shared Resources Cluster Template Overrides .

2. Click Create Custom Override.

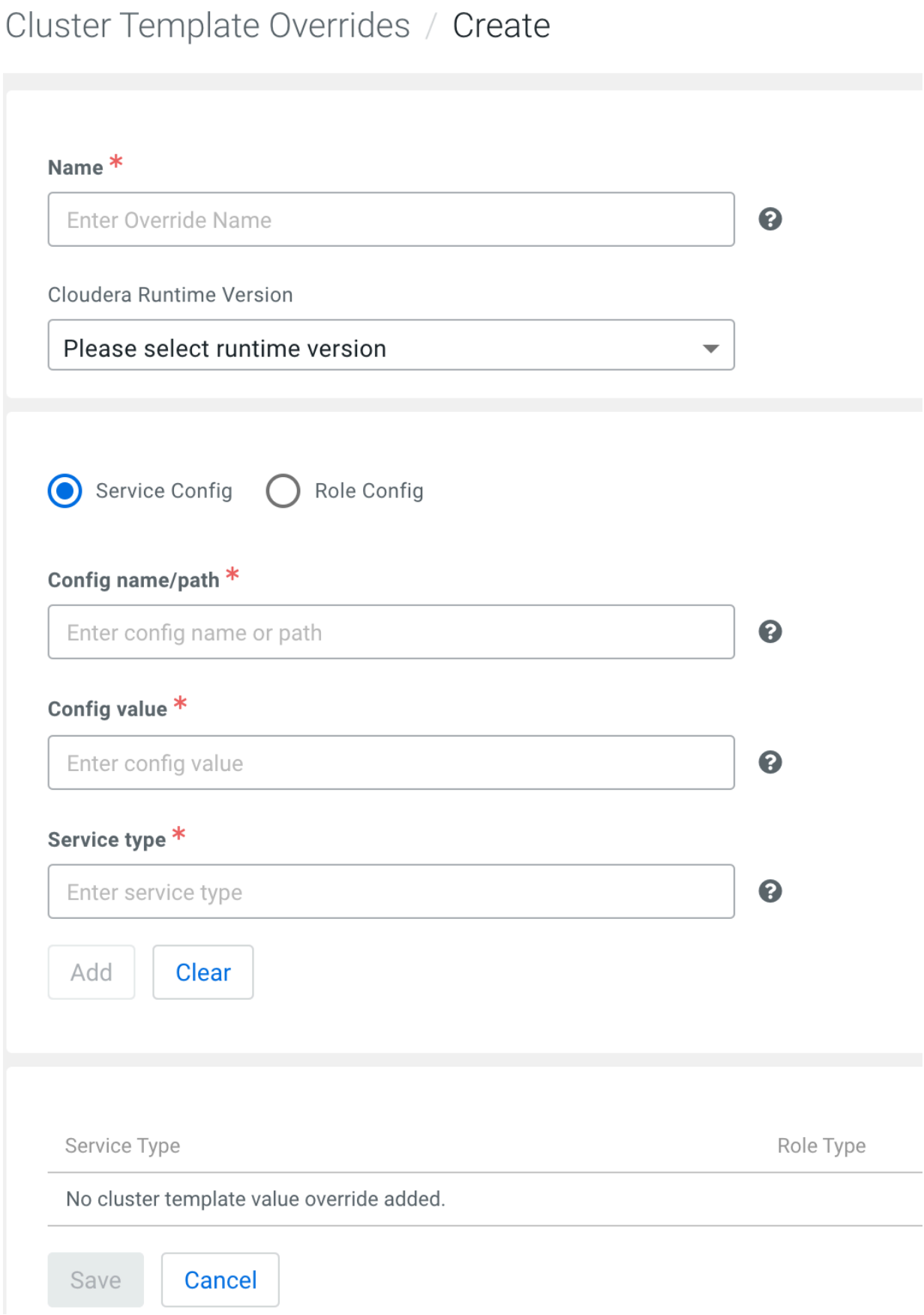

3. Provide the following information:

Property	Description
Name	Provide a unique name for the override.
Cloudera Runtime Version	Select a specific Cloudera Runtime version to which this override applies, or select “Not Runtime specific.”
Service Config / Role Config	Use the radio buttons to select whether you are customizing a service config or a role config.
Config Name/Path	Exact name/path of the config are you customizing.
Config Value	New value for the config. Ensure the value is valid.
Service Type	From the drop-down menu, select the service type for the service or role config that you are customizing.
Role Type	For role configurations: the role type as it appears in the cluster template that you will apply the override to (or any valid role configuration that is recognized by Cloudera Manager).

For example:

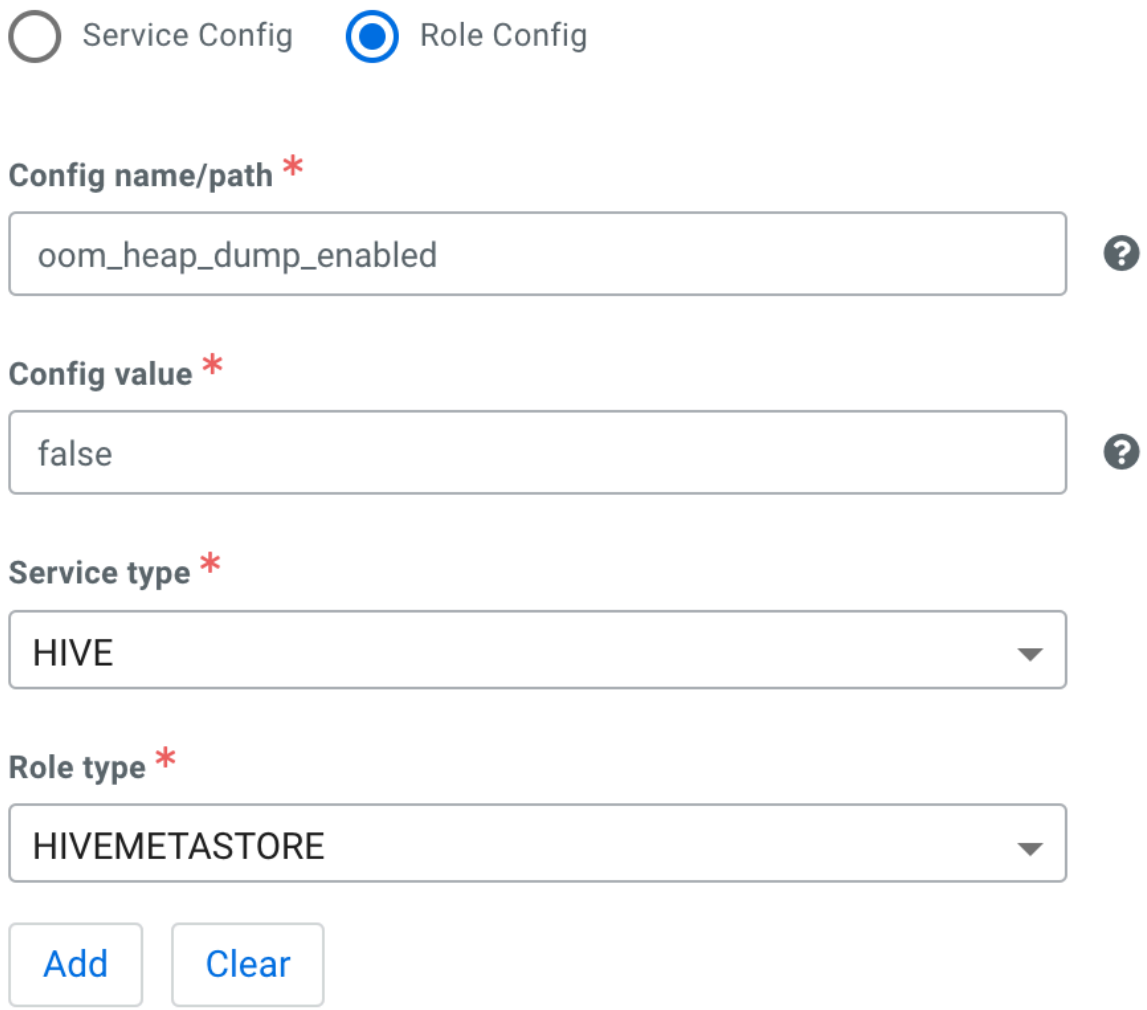

4. Click Add and then repeat the previous step (excluding the Name and Cloudera Runtime Version) for each service or role config that you want to customize in this template override.

5. When you are finished adding service and role configs to the template override, click Save. The override appears in the list of saved overrides, accessible from the Shared Resources menu.

Creating a Cloudera Data Hub cluster with a cluster template override

You can use a cluster template override that you previously created to provision a Cloudera Data Hub cluster.

Procedure

1. Click Data Hub Clusters Create Data Hub Cluster .
2. Enter the information required to create your cluster, then at the bottom of the cluster creation wizard, select Advanced OptionsCluster Extensions.
3. From the Cluster Templates Overrides drop-down box, select the template override that you previously created and want to apply when the cluster is provisioned. The only visible overrides are those with a Cloudera Runtime version matching the selected cluster definition version, or those that are Runtime-agnostic.

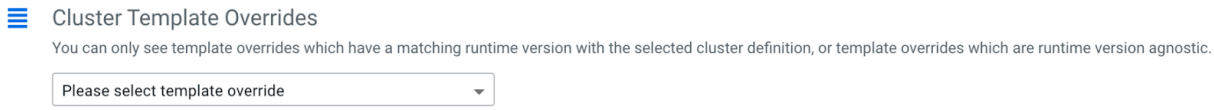

4. Click Provision Cluster.

Results

The Cloudera Data Hub cluster is provisioned using the service and role cofigs that you specified in the template override.

Cluster template override CLI commands

You can use the CDP CLI to create cluster template overrides, manage the cluster template overrides, and provision a Data Hub cluster using a cluster template override.

Table 1: Cluster Template Override CLI

Command	Description	Example
cdp datahub create-custom-configurations	Creates a new cluster template override shared resource with an indefinite amount of properties.	`cdp datahub create-custom-configurations --custom-configurations-name "test-custom-configs" --configurations configName="name",configValue="value",serviceType="hdfs" configName="name2",configValue="value2",roleType="role2",serviceType="hive_on_tez" .... --runtime-version "7.2.8"`
cdp datahub list-custom-configurations	Lists all the cluster template overrides for the account.	
cdp datahub describe-custom-configurations	Returns the name, CRN, the properties, and the platform version associated with the cluster template override resource.	`cdp datahub describe-custom-configurations --custom-configurations "test-overrides"`
cdp datahub delete-custom-configurations	Deletes multiple custom configs given their names or CRNs.	`cdp datahub delete-custom-configurations –custom-configurations "test-override" "test-override-2" ...`
cdp datahub create-aws-clustercdp datahub create-azure-clustercdp datahub create-gcp-cluster	Launches a Cloudera Data Hub cluster with custom configs from a cluster template override. The parameter for specifying a cluster template override is:--custom-configurations-name	`cdp datahub create-aws-cluster --cluster-name "de-dh1" --environment-name "local-aws-env" --cluster-template-name "7.2.8 - Data Engineering: Apache Spark, Apache Hive, Apache Ozie" --cluster-definition-name "7.2.8 - Data Engineering for AWS" --custom-configurations-name "test-override"`

Custom properties

The "Custom Properties "option allows you to set properties on a per-cluster basis.

This option is available from the Cluster Extensions page of the advanced create cluster wizard. For more information, refer to the documentation linked below.

Related Information

Custom Properties

Cluster deployment logs

Cluster deployment logs gather the same diagnostic (system and service) logs that are collected into your cloud storage location, but the Cloudera platform sends the logs to Cloudera engineering and support for troubleshooting purposes. You can apply configurable redaction rules on any sensitive data.

Cluster deployment logs are disabled by default. You can change this setting during environment creation or after environment creation, though the Cloudera platform will only collect logs for new deployments. You can also set a default behavior at the Cloudera account level, so that you won't have to enable the setting every time in the wizard. The logs are only gathered during deployment, which is approximately the first 20 minutes of cluster creation.

You can configure anonymization rules for log collection at the Cloudera account level. The rules are a list of rule objects with two fields: a regex pattern (PCRE) as the value, and a replacement string. Configure these rules to hide any sensitive data. By default, the rules hide email addresses and card numbers. This default behavior is the same as Cloudera Manager diagnostics collection.

The anonymization options are available under Global ResourcesTelemetry:

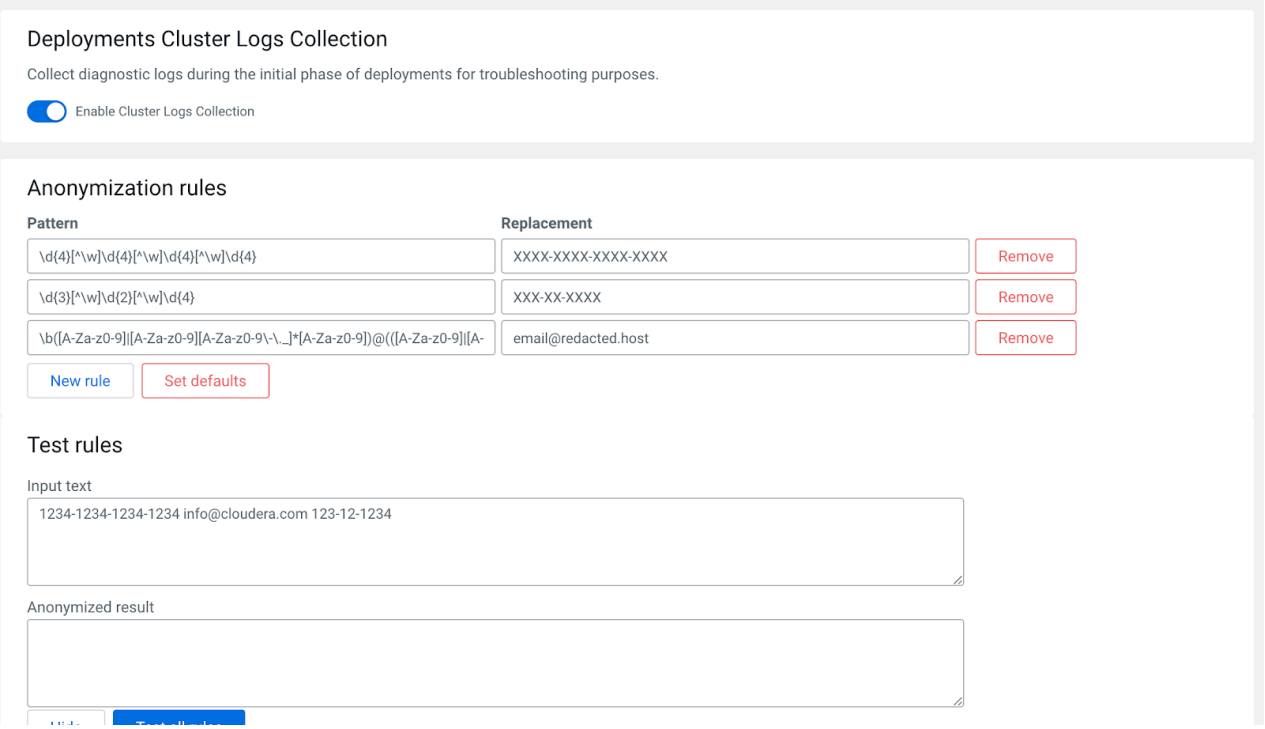

Currently, cluster deployment logs are collected only for Data Lake and Cloudera Data Hub cluster shapes.

Enabling/disabling cluster deployment log collection

You can enable cluster deployment log collection during environment creation. You can also enable or disable it after environment creation.

To enable cluster deployment log collection during environment creation:

1. Click the slider button under **Telemetry** in the **Region, Networking, Security, and Storage** page of environment registration.

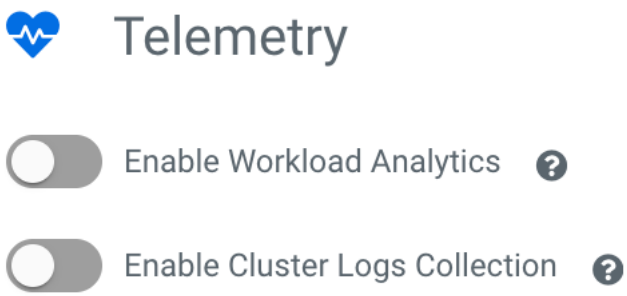

2. Select the checkbox to consent to data collection and click Enable.

To enable or disable cluster log deployment collection after environment creation:

1. Navigate to the **Environments** interface.
2. Select the desired environment, then click the Summary tab.
3. Scroll down to **Telemetry** and click the Edit button.
4. Click the slider button under Enable Deployment Cluster Logs Collection.
5. Select the checkbox to consent to data collection and click Enable.

To enable or disable cluster log deployment collection at the Cloudera account level (globally):

1. From the left navigation menu, click Global Settings and then Telemetry.
2. Click the slider button under Deployment Cluster Logs Collection.
3. Select the checkbox to consent to data collection and click Enable.

Cloudera Data Hub

Creating a Cluster on GCP

Date published: 2019-12-17
Date modified: 2023-06-27

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cluster deployment options 4

Default cluster configurations 4
- Data Engineering clusters 5
- Data Mart clusters 8
- Operational Database clusters 10
- Streams Messaging clusters 10
- Flow Management clusters 11
- Streaming Analytics clusters 13

Create a cluster from a definition on GCP 14

Create a custom cluster on GCP 16

Creating a cluster with a specific Java version 18

Advanced cluster options 19
- Tags 19
- Image catalog 19
- Custom images and image catalogs 20
  - Creating a custom image and image catalog with the CDP CLI 21
  - Switching image catalogs 23
- Network and availability 24
- Hardware and storage 24
- Cloud storage 25
- Recipes 25
- Cluster template overrides 25
  - Creating a cluster template override 28
  - Creating a Cloudera Data Hub cluster with a cluster template override 31
  - Cluster template override CLI commands 31
- Custom properties 32
- Cluster deployment logs 33
  - Enabling/disabling cluster deployment log collection 33

Cluster deployment options

You have two basic options when creating a cluster: (1) create a cluster from one of the default or custom cluster definition, or (2) create a custom cluster.

Cluster definitions

Cloudera Data Hub includes a set of prescriptive cluster definitions that can be used to quickly provision clusters for common use cases. These default cluster definitions include prescriptive options related to compute instance types and storage options.

We recommend that you start by reviewing these definitions. If you conclude that they do not match your requirements, you can customize them and then save them as custom cluster definitions.

Custom deployment

Use the custom cluster deployment option if you would like to select specific Cloudera Runtime components for your cluster.

Note:

Currently, this feature is limited and only allows you to select a cluster template that determines which components will be used for your cluster.

Advanced options

The cluster wizard includes a set of advanced options that can be optionally configured. Some of these options require additional configuration prior to cluster creation, so if you would like to use these options, read the cluster planning documentation first.

Related Information

Default cluster configurations

Cluster Planning

Default cluster configurations

Cloudera Data Hub includes a set of prescriptive cluster configurations. Each of these default cluster configurations include a cloud-provider specific cluster definition, which primarily defines cloud provider settings. The cluster definition references a cluster template, which defines a number of Cloudera Runtime or Cloudera DataFlow components used for common data analytics and data engineering use cases.

Refer to the topic for each default cluster configuration to view the included services and compatible Runtime versions. These topics include links to documentation that will help you to understand the included components and use the workload cluster.

Many of the cluster components are included in the Cloudera Runtime software distribution. The Streams Messaging, Cloudera Flow Management, and Cloudera Streaming Analytics cluster configurations are part of Cloudera DataFlow for Data Hub and have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for more details.

You can access the default cluster definitions by clicking Environments, then selecting an environment and clicking the Cluster Definitions tab.

You can access the default cluster templates from Shared ResourcesCluster Templates.

To view details of a cluster definition or cluster template, click on its name. For each cluster definition, you can access a raw JSON file. For each cluster template, you can access a graphical representation ("list view") and a raw JSON file ("raw view") of all cluster host groups and their components.

Related Information

Cloudera DataFlow for Data Hub

Cloudera Runtime

Data Engineering clusters

Learn about the default Data Engineering clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Engineering provides a complete data processing solution, powered by Apache Spark and Apache Hive. Spark and Hive enable fast, scalable, fault-tolerant data engineering and analytics over petabytes of data.

Data Engineering cluster definition

This Data Engineering template includes a standalone deployment of Spark and Hive, as well as Apache Oozie for job scheduling and orchestration, Apache Livy for remote job submission, and Hue and Apache Zeppelin for job authoring and interactive analysis.

Cluster definition names

- Data Engineering for Google Cloud
- Data Engineering HA - Spark3 for Google Cloud
- Data Engineering Spark3 for Google Cloud

Cluster template name

- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: Apache Spark, Apache Hive, Apache Oozie."

- Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: HA: Apache Spark, Apache Hive, Apache Oozie."

  See the architectural information below for the Data Engineering HA clusters

- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: The "Data Engineering: Apache Spark3" cluster template is deleted. Therefore, the "Data Engineering: Apache Spark3, Apache Hive, Apache Oozie" cluster template can be used instead.

Included services

- HDFS
- Hive
- Hue
- Livy
- Spark 3
- Yarn
- Zeppelin
- ZooKeeper
- Oozie is supported for Spark 3 as of Cloudera Runtime version 7.2.18
- Hive Warehouse Connector is supported as of Cloudera Runtime version 7.2.16.

Compatible Cloudera Runtime version

7.2.16, 7.2.17, 7.2.18

Topology of the Data Engineering cluster

Topology is a set of host groups that are defined in the cluster template and cluster definition used by Data Engineering. Data Engineering uses the following topology:

Host group	Description	Node configuration
Master	The master host group runs the components for managing the cluster resources including Cloudera Manager, Name Node, Resource Manager, as well as other master components such HiveServer2, HMS, Hue etc.	1 For Cloudera Runtime versions earlier than 7.2.14: GCP : e2-standard-16; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE, DE Spark3, and DE HA: GCP : e2-standard-16; pd-ssd - 100 GB
Worker	The worker host group runs the components that are used for executing processing tasks (such as NodeManager) and handling storing data in HDFS such as DataNode.	3 For Cloudera Runtime versions earlier than 7.2.14: GCP : e2-standard-8; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: GCP : e2-standard-8; pd-ssd - 100 GB DE HA: GCP : e2-standard-8; pd-ssd - 100 GB
Compute	The compute host group can optionally be used for running data processing tasks (such as NodeManager). By default the number of compute nodes is set to 1 for proper configurations of YARN containers. This node group can be scaled down to 0 when there are no compute needs. Additionally, if load-based auto-scaling is enabled with minimum count set to 0, the compute nodegroup will be resized to 0 automatically.	0+ For Cloudera Runtime versions earlier than 7.2.14: GCP : e2-standard-8; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: GCP : e2-standard-8; pd-ssd - 100 GB DE HA:

Host group	Description	Node configuration
		GCP : e2-standard-8; pd-ssd - 100 GB **Note:** Compute nodes run YARN and require storage only for temporary data - this requirement is fulfilled by instance storage, so making the attached volumes count to 0 by default is more cost-efficient.
Gateway	The gateway host group can optionally be used for connecting to the cluster endpoints like Oozie, Beeline etc. This nodegroup does not run any critical services. This nodegroup resides in the same subnet as the rest of the nodegroups. If additional software binaries are required they could be installed using recipes.	0+ GCP : e2-standard-8; pd-ssd - 100 GB

Service configurations			
Master host group Cloudera Manager, HDFS, Hive (on Tez), HMS, Yarn RM, Oozie, Hue, DAS, Zookeeper, Livy, Zeppelin and Sqoop	Gateway host group Configurations for the services on the master node	Worker host group Data Node and YARN NodeManager	Compute group YARN NodeManager

Configurations

Note the following:

- There is a Hive Metastore Service (HMS) running in the cluster that talks to the same database instance as the Data Lake in the environment.
- If you use CLI to create the cluster, you can optionally pass an argument to create an external database for the cluster use such as Cloudera Manager, Oozie, Hue, and DAS. This database is by default embedded in the master node external volume. If you specify the external database to be of type HA or NON_HA, the database will be provisioned in the cloud provider. For all these types of databases the lifecycle is still associated with the cluster, so upon deletion of the cluster, the database will also be deleted.
- The HDFS in this cluster is for storing the intermediary processing data. For resiliency, store the data in the cloud object stores.
- For high availability requirements choose the Data Engineering High Availability cluster shape.

GCP HA (Preview)

Note: HA for Oozie is not yet available in the GCP template.

Custom templates

Any custom DE HA template that you create must be forked from the default templates of the corresponding version. You must create a custom cluster definition for this with the JSON parameter “enableLoadBalancers”: true , using the create-aws/azure/gcp-cluster CLI command parameter --request-template. Support for pre-existing custom cluster definitions will be added in a future release. As with the template, the custom cluster definition must be forked from the default cluster definition. You are allowed to modify the instance types and disks in the custom cluster definition. You must not change the placement of the services like Cloudera Manager, Oozie, and Hue. Currently the custom template is fully supported only via CLI.

The simplest way to change the DE HA definition is to create a custom cluster definition. In the Create Data Hub UI when you click Advanced Options, the default definition is not used fully, which will cause issues in the HA setup.

Related Information

HDFS
Hive
Hue
Oozie
Spark
YARN
Zookeeper

Data Mart clusters

Learn about the default Data Mart and Real Time Data Mart clusters, including cluster definition and template names, included services, and compatible Runtime version.

Data Mart is an MPP SQL database powered by Apache Impala designed to support custom Data Mart applications at big data scale. Impala easily scales to petabytes of data, processes tables with trillions of rows, and allows users to store, browse, query, and explore their data in an interactive way.

Data Mart clusters

The Data Mart template provides a ready to use, fully capable, standalone deployment of Impala. Upon deployment, it can be used as a standalone Data Mart to which users point their BI dashboards using JDBC/ODBC end points. Users can also choose to author SQL queries in Cloudera’s web-based SQL query editor, Hue, and run them with Impala providing a delightful end-user focused and interactive SQL/BI experience.

Cluster definition names

- Data Mart for Google Cloud

Cluster template name

Cloudera - Data Mart: Apache Impala, Hue

Included services

- HDFS
- Hue
- Impala

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17, 7.2.18

Real Time Data Mart clusters

The Real-Time Data Mart template provides a ready-to-use, fully capable, standalone deployment of Impala and Kudu. You can use a Real Time Data Mart cluster as a standalone Data Mart which allows high throughput streaming

ingest, supporting updates and deletes as well as inserts. You can immediately query data through BI dashboards using JDBC/ODBC end points. You can choose to author SQL queries in Cloudera's web-based SQL query editor, Hue. Executing queries with Impala, you will enjoy an end-user focused and interactive SQL/BI experience. This template is commonly used for Operational Reporting, Time Series, and other real time analytics use cases.

Cluster definition names

- Real-time Data Mart for Google Cloud

Cluster template name

Cloudera - Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 2
- Yarn

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster definition names

- Real-time Data Mart - Spark3 for Google Cloud

Cluster template name

Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark3

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 3
- Yarn

Compatible Cloudera Runtime versions

7.2.16, 7.2.17, 7.2.18

High availability

Cloudera recommends that you use high availability (HA), and track any services that are not capable of restarting or performing failover in some way.

Impala HA

The Impala nodes offer high availability. The following Impala services are not HA.

- Catalog service
- Statestore service

Kudu HA

Both Kudu Masters and TabletServers offer high availability.

Related Information

HDFS

Hue
Impala
Kudu
Spark
YARN

Operational Database clusters

The Operational Database (OpDB) template is removed from the Cloudera Data Hub. You can access the Cloudera Operational Database instead as a superior product.

The Cloudera Operational Database is a NoSQL database powered by Apache HBase designed to support custom OLTP applications that want to leverage the power of BigData. Apache HBase is a NoSQL, scale-out database that can easily scale to petabytes and stores tables with millions of columns and billions of rows.

Cloudera Operational Database also contains Apache Phoenix which provides a way to use HBase through an SQL interface.

Cloudera recommends you to use the Cloudera Operational Database to create Operational Database clusters.

Related Information

Cloudera Operational Database
Getting started with Operational Database
Before you create an Operational Database cluster
Creating an Operational Database cluster
HDFS
HBase
Knox
Zookeeper
Phoenix

Streams Messaging clusters

Learn about the default Streams Messaging clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streams Messaging provides the following features:

- Advanced messaging and real-time processing on streaming data using Apache Kafka
- Centralized schema management using Schema Registry
- Management and monitoring capabilities powered by Streams Messaging Manager
- Cross-cluster Kafka topic replication using Streams Replication Manger
- Kafka partition rebalancing with Cruise Control

This template sets up a fault-tolerant standalone deployment of Apache Kafka and supporting Cloudera components (Schema Registry, Streams Messaging Manager, Streams Replication Manager and Cruise Control), which can be used for production Kafka workloads in the cloud or as a disaster recovery instance for on-premises. Kafka clusters.

Note:

Streams Messaging clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streams Messaging cluster deployment
- Creating your first Streams Messaging cluster
- Connecting Kafka clients to Cloudera Public Cloud clusters

Cluster definition names

- Streams Messaging Heavy Duty for GCP
- Streams Messaging Light Duty for GCP
- Streams Messaging HA for GCP (Technical Preview)

Cluster template name

- Cloudera - Streams Messaging Heavy Duty
- Cloudera - Streams Messaging Light Duty
- Cloudera - Streams Messaging High Availability

Included services

- Kafka
- Schema Registry
- Streams Messaging Manager
- Streams Replication Manager
- Cruise Control
- Kafka Connect

Compatible Cloudera Runtime version

- 7.1.0 (Preview)
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streams Messaging cluster
Ingesting Data into Cloudera Public Cloud
Kafka
Schema Registry
Streams Messaging Manager
Streams Replication Manager

Flow Management clusters

Learn about the default Flow Management clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime versions.

Flow Management delivers high-scale data ingestion, transformation, and management to enterprises from any-to-any environment. It addresses key enterprise use cases such as data movement, continuous data ingestion, log data ingestion, and acquisition of all types of streaming data including social, mobile, clickstream, and IoT data.

The Flow Management template includes a no-code data ingestion and management solution powered by Apache NiFi. With NiFi's intuitive graphical interface and 300+ processors, Flow Management enables easy data ingestion and movement between CDP services as well as 3rd party cloud services. NiFi Registry is automatically set up and provides a central place to manage versioned Data Flows.

Note:

Flow Management clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Flow Management cluster deployment
- Creating your first Flow Management cluster
- Security considerations for Flow Management clusters
- Using Apache NiFi to ingest data into Cloudera Public Cloud
- Using NiFi and NiFi Registry

Cluster definition names

- Flow Management Light Duty for GCP
- Flow Management Heavy Duty for GCP

Cluster template name

- CDP - Flow Management: Light Duty
- CDP - Flow Management: Heavy Duty

Included services

- NiFi
- NiFI Registry

Compatible Cloudera Runtime versions

- 7.1.0
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Flow Management cluster
Apache NiFi documentation
Apache NiFi Registry documentation

Streaming Analytics clusters

Learn about the default Streaming Analytics clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streaming Analytics offers real-time stream processing and stream analytics with low-latency and high scaling capabilities powered by Apache Flink.

Streaming Analytics templates include Apache Flink that works out of the box in stateless or heavy state environments. Beside Flink, the template includes its supporting services namely YARN, Zookeeper and HDFS. The Heavy Duty template comes preconfigured with RocksDB as state backend, while Light Duty clusters use the default Heap state backend. You can create your streaming application by choosing between Kafka, Kudu, and HBase as datastream connectors.

You can also use SQL to query real-time data with SQL Stream Builder (SSB) in the Streaming Analytics template. By supporting the SSB service in Cloudera Public Cloud, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. SSB is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

Note:

Streaming Analytics clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streaming Analytics cluster deployment
- Creating your first Streaming Analytics cluster
- Analyzing data using Apache Flink
- Querying data using SQL Stream Builder

Cluster definition names

- Streaming Analytics Light Duty for GCP
- Streaming Analytics Heavy Duty for GCP

Cluster template name

- 7.2.17 - Streaming Analytics Light Duty
- 7.2.17 - Streaming Analytics Heavy Duty

Included services

- Flink
- SQL Stream Builder
- YARN
- Zookeeper
- HDFS
- Kafka

Important: In the Streaming Analytics cluster templates, Kafka service is included by default to serve as a background service only for the websocket ouput and sampling feature of SQL Stream Builder. The Kafka service in the Streaming Analytics cluster template cannot be used for production, you need to use the Streams Messaging cluster template when Kafka is needed for your deployment.

Compatible Cloudera Runtime version

- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9

- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streaming Analytics cluster
Flink
YARN
Zookeeper
HDFS

Create a cluster from a definition on GCP

You can quickly create clusters from default or custom cluster definitions within an existing GCP environment.

Before you begin

To create a Cloudera Data Hub cluster on GCP, you must have an existing GCP environment. Also, you should make sure that the Runtime version of the Data Lake cluster matches the Cloudera Runtime version of the Cloudera Data Hub cluster that you are about to create; If these versions don't match, you may encounter warnings and/or errors.

About this task

Required role: DataHubCreator, EnvironmentAdmin at the scope of the environment where the Cloudera Data Hub cluster is running, or Owner of the environment

Procedure

1. Log in to the Cloudera web interface.

2. Navigate to the Cloudera Management Console > Environments > click on an environment where you would like to create a cluster > click Create Data Hub. The following page is displayed:

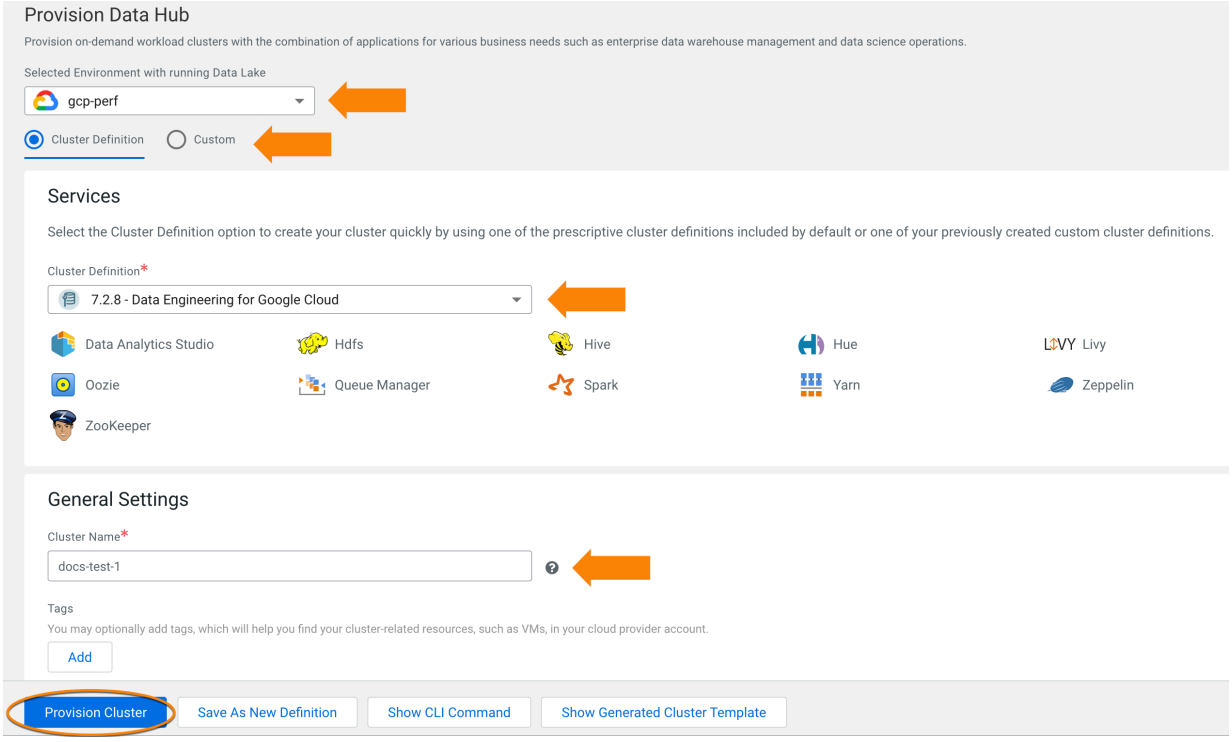

3. Select Cluster Definition.
4. From the Cluster Definition dropdown, select the cluster definition that you would like to use for your cluster.

   The cluster template referenced in the selected cluster definition determines which services are included in the cluster. The list of services is automatically shown below the selected cluster definition name:
5. Specify General Settings for your cluster:

Parameter	Description
Cluster Name	Enter a name for your cluster. The name must be between 5 and 40 characters, must start with a letter, and must only include lowercase letters, numbers, and hyphens.
Tags	(Optional) Add tags that Data Hub should use to tag your GCP resources. Click Add to add a tag, and then enter a key and value for each tag. Repeat the steps if you would like to add more tags. For more information about tags, refer to Tags.

6. Optionally, click on Advanced Options to modify advanced cluster settings. For more information on these options, refer to *Advanced cluster options*.
7. On AWS and Azure only: Optionally, when you have finished providing the cluster settings, you can click the Show CLI Command button at the bottom of the page to review or copy the CDP CLI command used to create the cluster. You can copy the command from the pop-up window that appears, either to provision the cluster later or for use in scripts.
8. You also have the option to review or copy the cluster template that is generated and will be used in cluster creation. Click the Show Generated Cluster Template button at the bottom of the page.
9. To proceed with cluster provisioning immediately, click on Provision Cluster.
10. You will be redirected to the Cloudera Data Hub cluster dashboard, and a new tile representing your cluster will appear at the top of the page.

    The following messages are written to the event history as your cluster is being created:

```
Cluster built; Cluster manager ip:10.97.82.237
8/2/2019, 6:10:44 PM
```

```
Updating Cluster Proxy service with gateway configuration
8/2/2019, 6:10:43 PM
Building cluster; Cluster manager ip:10.97.82.237
8/2/2019, 5:45:12 PM
Starting cluster services
8/2/2019, 5:42:33 PM
Mounting attached disks
8/2/2019, 5:42:18 PM
Setting up infrastructure metadata
8/2/2019, 5:42:14 PM
Bootstrapping infrastructure cluster
8/2/2019, 5:41:59 PM
Registering cluster with Cluster Proxy service
8/2/2019, 5:41:57 PM
Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Billing started, Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Infrastructure metadata collection finished
8/2/2019, 5:41:55 PM
Infrastructure creation took 96 seconds
8/2/2019, 5:41:52 PM
Creating infrastructure
8/2/2019, 5:40:16 PM
Setting up CDP image
8/2/2019, 5:40:15 PM
```

11. When your cluster is ready, its status will change to Running.

What to do next

You can access links to Cloudera Manager, and other cluster UIs and endpoints from cluster details.

Related Information

Advanced cluster options

Accessing a Cluster

Create a custom cluster on GCP

Create a custom Cloudera Data Hub cluster within an existing GCP environment.

Before you begin

To create a Cloudera Data Hub cluster on GCP, you must have an existing environment. Also, you should make sure that the Cloudera Runtime version of the Data Lake cluster matches the Runtime version of the Cloudera Data Hub cluster that you are about to create; If these versions don't match, you may encounter warnings and/or errors.

https://www.youtube.com/embed/pGHHW9nOrPU

About this task

Required role: DataHubCreator, EnvironmentAdmin at the scope of the environment where the Data Hub is running, or Owner of the environment

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Environments > click on an environment > click Create Data Hub.

3. Under Selected Environment, confirm that the selected environment is the one where you would like to create your cluster.
4. Select Custom.
5. Under Platform Version, current Cloudera Runtime version is pre-selected.
6. Under Cluster template, select the cluster template that you would like to use.

   The selected cluster template determines which services are included in the cluster. Select the cluster template and the list of services is automatically shown below it:

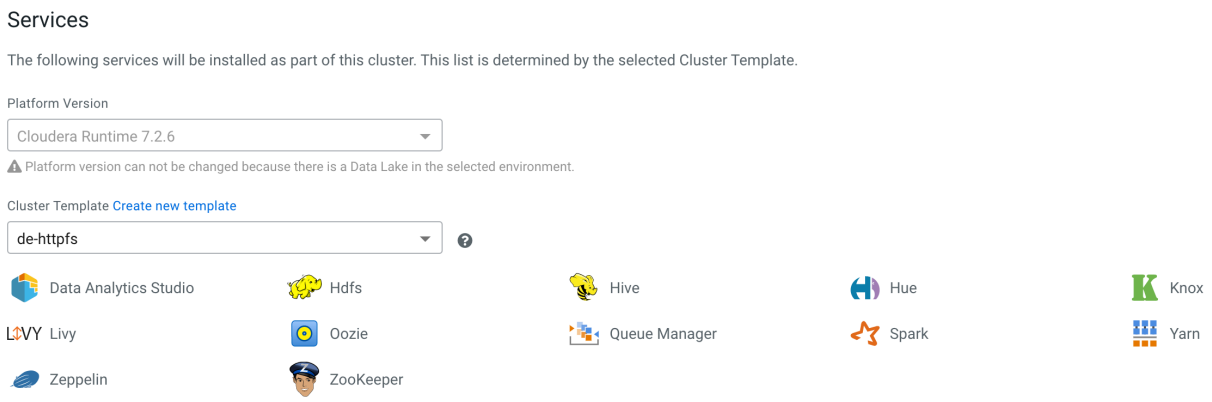

   For more information about cluster templates, refer to Cluster templates.
7. Specify General Settings for your cluster:

Parameter	Description
Cluster Name	Enter a name for your cluster. The name must be between 5 and 40 characters, must start with a letter, and must only include lowercase letters, numbers, and hyphens.
Tags	(Optional) Add tags that Data Hub should use to tag your GCP resources. Click Add to add a tag, and then enter a key and value for each tag. Repeat the steps if you would like to add more tags. For more information about tags, refer to Tags.

8. Optionally, click on Advanced Options to modify advanced cluster settings. For more information on these options, refer to LINK.
9. Once done, click on Provision Cluster.
10. You will be redirected to the Cloudera Data Hub cluster dashboard, and a new tile representing your cluster will appear at the top of the page.

    The following messages are written to the event history as your cluster is being created:

```
Cluster built; Cluster manager ip:10.97.82.237
8/2/2019, 6:10:44 PM
Updating Cluster Proxy service with gateway configuration
8/2/2019, 6:10:43 PM
Building cluster; Cluster manager ip:10.97.82.237
8/2/2019, 5:45:12 PM
Starting cluster services
8/2/2019, 5:42:33 PM
Mounting attached disks
8/2/2019, 5:42:18 PM
Setting up infrastructure metadata
8/2/2019, 5:42:14 PM
Bootstrapping infrastructure cluster
8/2/2019, 5:41:59 PM
Registering cluster with Cluster Proxy service
8/2/2019, 5:41:57 PM
Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
```

```
Billing started, Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Infrastructure metadata collection finished
8/2/2019, 5:41:55 PM
Infrastructure creation took 96 seconds
8/2/2019, 5:41:52 PM
Creating infrastructure
8/2/2019, 5:40:16 PM
Setting up CDP image
8/2/2019, 5:40:15 PM
```

11. When your cluster is ready, its status will change to Running.

What to do next

You can access links to Cloudera Manager, and other cluster UIs and endpoints from cluster details.

Related Information

Advanced cluster options

Accessing a Cluster

Creating a cluster with a specific Java version

When you create a new Cloudera Data Hub or Data Lake cluster, you can specify Java 8 or Java 11 as the default JDK.

Java 11

When you launch a new Cloudera Data Hub or Data Lake cluster, the default JDK version is Java 8 unless you specify Java 11. If you specify Java 11 as the default JDK version, Cloudera Manager and all Cloudera Runtime services (as well as all workloads launched) will use pre-installed Java 11 dependencies rather than Java 8. This may be useful, for instance, when you have workloads such as Spark 3 jobs that are written in Java 11, when your workloads have Java 11 dependencies, or if you are running third-party components on a cluster that require Java 11.

The ability to launch new Cloudera Data Hub and Data Lake clusters with Java 11 is currently available only through the CDP CLI. Include the parameter --java-version in the create-<cloud_provider>-datalake and create-<cloud_provider>-datahub CLI commands. Valid inputs for the --java-version parameter are 8 and 11.

Launching clusters with Java 11 as the default is supported from Cloudera Runtime versions 7.2.15.11 and 7.2.16.200 and above. Creating clusters with the Java version property set to 11 on older Cloudera Runtime versions will be blocked by an internal validation. Currently, the Java version selection is only available when creating a new cluster.

Java 8 and 11 compatibility

Java 11 introduces breaking changes, which could lead to incompatibility with your current workloads or custom/third-party Java-based applications deployed on Cloudera Data Hub clusters.

Cloudera recommends that all existing production workloads are tested before moving them over to a Cloudera Data Hub cluster that is set to use Java 11 as the default JDK.

Note that Data Lake and Cloudera Data Hub hosts will continue to have both Java 8 and 11 pre-installed. The --java-version switch controls which Java versionn should be set as the OS default during bootstrapping of the hosts. Therefore, it is possible to continue to run jobs written for Java 8 even if the OS default is Java 11. Some services (for example, Spark) make this possible by setting the JAVA_PATH environment variable to point to the Java 8 path /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.[...].x86_64/jre/bin/java.

Advanced cluster options

In the create cluster wizard, click on Advanced Options to view the advanced cluster configuration options.

https://www.youtube.com/embed/Q8Xjb3-8KFk

While some of these advanced options can be configure in the wizard, others require prior setup.

The following options are available:

Tags

You can define tags that will be applied to your cluster-related resources (such as VMs) on your cloud provider account.

The tags added during cluster creation are displayed in your cloud account on the resources that Cloudera Data Hub provisioned for your clusters. You can use tags to categorize your cloud resources by purpose, owner, and so on. Tags come in especially handy when you are using a corporate cloud provider account and you want to quickly identify which resources belong to your cluster(s). In fact, your corporate cloud account admin may require you to tag all the resources that you create, in particular resources, such as VMs, which incur charges.

By default, the following tags are created:

Tag	Description
Cloudera-Resource-Name	The workload-appropriate Cloudera resource name. This CRN serves as a unique identifier for the resource over time.
Cloudera-Creator-Resource-Name	Cloudera resource name of the Cloudera user that created the resource.
Cloudera-Environment-Resource-Name	The name of the environment with which the resource is associated.

You can optionally add additional tags. To add custom tags:

1. In the create cluster wizard, navigate to the General Configuration page.
2. Specify your tags in the Tags section by providing a key and value for each tag.

Note:

It is not possible to add tags via Cloudera Data Hub after your cluster has been created. In this case, you can only add the tags manually via your cloud provider's interface.

To learn more about tags and their restrictions, refer to GCP documentation.

Image catalog

The options on the "Image Settings" page of the advanced create cluster wizard allow you to select custom image settings.

By default, Cloudera Data Hub uses the prewarmed image from the image catalog provided in Cloudera Data Hub. If necessary, you can also customize a default image.

Choose image catalog

Cloudera Data Hub uses the image catalog provided by default. If you would like to use a custom image catalog instead of the default image catalog, you must first prepare your custom images, then create and register an image catalog.

Choose image type

By default, Cloudera Data Hub uses the included prewarmed images with the default Cloudera Manager and Cloudera Runtime version, but you can select a different prewarmed image or customized prewarmed image to use for your cluster. Cloudera Data Hub currently supports the following types of images for launching clusters:

Image type	Description	Default images provided
Prewarmed Image	By default, Cloudera Data Hub launches clusters from prewarmed images. Prewarmed images include the operating system as well as Cloudera Manager and Cloudera Runtime. The Cloudera Manager and Cloudera Runtime version used on prewarmed images cannot be customized.	Yes
Custom Image	You can customize a default image for compliance or security reasons. You can then use the CDP CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create a Cloudera Data Hub cluster.	

Choose image

This option allows you to select a different image.

Related Information

Image catalog updates

Custom images and image catalogs

If necessary, you can use a custom Cloudera Runtime or FreeIPA image for compliance or security reasons. You can then use the Cloudera CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create a Data Lake, Cloudera Data Hub cluster or environment with a custom FreeIPA image.

Overview

A custom image should inherit most of its attributes from its source image, which is a default image that you select from the cdp-default image catalog.

The typical method of creating a Data Lake or Cloudera Data Hub picks up the latest pre-warmed image from the cdp-default image catalog for the specified version of Cloudera Runtime. These default images are pre-warmed VM images that contain a base URL to the default parcels in the Cloudera archive, amongst other configurations. If the default pre-warmed images do not suit your business needs, you can specify that the Data Lake, Cloudera Data Hub cluster or the environment (in the case of FreeIPA) uses a custom image instead.

What is a custom image?

A custom image is an entry in a custom image catalog that inherits most of its attributes from a source (default) image.

Custom image entries have:

- An image type: Cloudera Runtime [which includes Cloudera Data Hub and Data Lake images] or FreeIPA
- A source image ID that points to an image in the cdp-default image catalog
- A timestamp of creation

- An option to specify a VM region and image reference (such as an AMI ID) if you are overriding the source image with a custom VM image
- An option to override the parcel base URL

Why use a custom image?

You might require a custom image for compliance or security reasons (a “hardened” image), or to have your own packages pre-installed on the image, for example monitoring tools or software. You might also want to specify a custom image if you need to use a default image with a specific Runtime maintenance version applied, rather than simply specifying the latest major Runtime version.

Note: When customizing VM images, note that certain customizations (for example, CIS hardening rules) may not be compatible with Cloudera.

What can you customize?

In a custom image entry, you can override the VM images themselves with your own custom images that are sufficiently hardened. Importantly, you should only customize a default image from the cdp-default catalog as opposed to creating one from scratch. You can also override the default parcel base URL (at archive.cloudera.com) with your own host site.

What is a custom image catalog?

A custom image catalog is simply a catalog that holds custom images. A custom image catalog can contain a single or multiple custom image entries.

Custom image catalogs have:

- A name. The name is a unique identifier and is used to refer to the catalog during environment, Data Lake, and Cloudera Data Hub cluster creation; as well as during catalog operations like creating an image.
- A description.
- An owner. The owner is the user who runs the command to create the catalog.

What is the process for creating a custom image and catalog?

- If you are replacing the VM images in a custom image entry with a customized version, you should first prepare the image by modifying an official Cloudera default image, which you can find under Shared Resources > Image Catalogs > cdp-default.
- Select a source image from the cdp-default image catalog to be the source of customization. When you run the CLI command to find a default image, you specify the Runtime version, provider, image type, or a combination of the three.
- Create a custom image catalog, or identify an existing catalog where you want to save the custom image entry.
- Apply the necessary changes to the custom image entry, like the override AMI IDs with the new, customized AMIs; or add a new parcel base URL using the `--base-parcel-url` command when you set the custom image.
- You can then create an environment, Data Lake, or Cloudera Data Hub cluster, based on custom catalogs via the CDP CLI.

Creating a custom image and image catalog with the CDP CLI

You can create a custom Cloudera Runtime or FreeIPA image and image catalog through the CDP CLI.

Before you begin

If you are replacing the VM images in a custom image entry with a customized version, you must first prepare the image by modifying an official Cloudera default image, which you can find under Shared Resources Image Catalogs cdp-default . Take note of the image reference, such as the AMI ID.

Procedure

1. To find a source image from the cdp-default catalog that you want to use as the source of your custom image entry, run the following command:

```
cdp imagecatalog find-default-image --provider <cloud provider> --image-
type <image type> --runtime-version <Cloudera Runtime version>
```

For example:

```
cdp imagecatalog find-default-image --provider AWS --image-type runtime --
runtime-version 7.2.12
```

2. A custom image requires a custom catalog. If you haven't yet created a custom catalog, or if you want to create a new one for a new custom image, run the following command:

```
cdp imagecatalog create-custom-catalog --catalog-name <unique catalog na
me> --description <catalog description>
```

For example:

```
cdp imagecatalog create-custom-catalog --catalog-name my custom catalog --
description test catalog
```

3. Within the custom image catalog that you created (or an older custom catalog that you want to use), create a custom image entry with the selected source image marked as its source. Providing your own VM images is optional. Run the following command:

```
cdp imagecatalog set-<image-type>-image --catalog-name <name of the custom
 catalog> --vm-images region=<region of customized image>,imageReference=<
cloud provider specific ID of a customized image> --source-image-id <cdp
 image ID of source image>
```

For example:

```
cdp imagecatalog set-runtime-image --catalog-name my custom catalog --vm-
images region=eu-central-1,imageReference=ami-7torotmhqi6q7438y --source-
image-id 8t4y9853-12b6-3n6z-75dh-tx775k4c793w
```

4. You can then apply the necessary changes to the custom image entry, like overriding AMI IDs with new, customized ones, or adding a new parcel base URL.

Command	Description
cdp imagecatalog find-default-image	Finds the default images in the cdp-default image catalog for a specified version of Cloudera Runtime. Parameters: • --provider • --image-type • --runtime-version You can provide any combination of these parameters.
cdp imagecatalog create-custom-catalog	Creates a custom catalog. Parameters: • --catalog-name • --description

Command	Description
	–catalog-name is required.
cdp imagecatalog set-runtime-image or cdp imagecatalog set-freeipa-image	Creates a custom image entry (either Cloudera Runtime [Cloudera Data Hub/Data Lake] or FreeIPA) within the specified catalog. Parameters: • --catalog-name • --vm-images [region,imageReference] • --source-image-id • --image-id • --base-parcel-url --catalog-name and --source-image-id are required.

Switching image catalogs

You can switch the image catalog of an already existing Cloudera Data Hub, Data Lake, or FreeIPA cluster. You may want to switch the image catalog for a cluster in order to restrict which Cloudera Runtime version can be upgraded to, or in order to move to custom images for an existing cluster.

Use the following CDP CLI commands to switch the image catalog for an existing cluster:

- FreeIPA:

```
cdp environments set-catalog --environment $ENVIRONMENT_NAME --catalog $
CATALOG
```

Parameter	Description
--environment	Name or CRN of the environment that holds the FreeIPA installation.
--catalog	URL of the FreeIPA catalog to be used.

- Cloudera Data Hub:

```
cdp datahub set-catalog --cluster $DATAHUB_NAME --catalog-name $CATALOG
```

Parameter	Description
--cluster	Name or CRN of the Cloudera Data Hub cluster for which you want to use the new image catalog.
--catalog-name	Name of the image catalog to be used. The image catalog must be a JSON based catalog, and switching is only possible from catalogs that are JSON based.

- Data Lake:

```
cdp datalake set-catalog --datalake $DATALAKE_NAME --catalog-name $CATALOG
```

Parameter	Description
--datalake	Name of the Data Lake for which you want to use the new image catalog.

Parameter	Description
--catalog-name	Name of the image catalog to be used. The image catalog must be a JSON based catalog, and switching is only possible from catalogs that are JSON based.

Network and availability

The "Network and availability" options allow you to customize the networking settings of your cluster.

On the Network and Availability page, provide the following to specify the networking resources that will be used for your cluster:

Parameter	Description
Select Subnet	If your environment includes a single subnet, that subnet is preselected and it cannot be changed. If your environment includes more than one subnet, you can select the subnet(s) in which your cluster will be provisioned.
Select Azure Database Server	A Cloudera Data Hub cluster uses the same Flexible Server or Single Server settings as the environment in which it runs but you can choose to enable a Flexible server on Cloudera Data Hub clusters running in an environment that uses a Single server. For more information, see Using Azure Database for PostgreSQL Flexible Server.

Hardware and storage

The "Hardware and storage" options allow you to customize the cloud provider specific cluster hardware and storage options.

The Hardware and Storage options can be selected for each host group. To edit this section for a specific host group, click on the . When done editing, click on the to save the changes. Repeat for these steps for all host groups that you would like to edit.

The following hardware and storage settings are available:

Parameter	Description
Cloudera Manager Server	You must select one node for Cloudera Manager Server by clicking the button. The “Instance Count” for that host group must be set to “1”. If you are using one of the default cluster templates, this is set by default.
Instance Type	Select an instance type. For information about instance types on GCP refer to Virtual Machine instances in GCP documentation.
Instance Count	Enter the number of instances of a given type. Default is 1.
Storage Type	Select the volume type. The options are: Standard persistent disks (HDD), solid-state persistent disks (SSD), or local scratch disk (SSD). For more information about these options refer to Storage options in GCP documentation. **Note:** Stopping and restarting Cloudera Data Hub clusters using ephemeral storage is not supported.
Attached Volumes Per Instance	Enter the number of volumes attached per instance. Default is 1.
Volume Size	Enter the size in GB for each volume. Default is 100.
Root Volume Size	This option allows you to increase or decrease the root volume size. Default is 200 GB. This option is useful if your custom image requires more space than the default 200 GB. If you use a custom Cloudera Data Hub template specifying a root volume size smaller than 200GB, you may encounter an error.

Parameter	Description
Encryption	By default, Compute Engine encrypts customer content at rest but you can use Customer-Supplied Encryption Key or Cloud KMS Key to encrypt your data. Read more about Customer-Supplied Encryption Key encryption here or Cloud KMS Key encryption here.

Cloud storage

The options on the "Cloud Storage" page allow you to optionally specify the base storage location used for YARN and Zeppelin.

During environment creation under Data Access > Storage Location Base you configure a default Google Cloud Storage base storage location for the environment, and all Cloudera Data Hub clusters created within that environment use this location. The Cloud Storage options in the Cloudera Data Hub cluster wizard allow you to additionally specify a different location for YARN application logs and Zeppelin Notebook's root directory:

- Existing Base Storage Location - By default, this is set to the Storage Location Base configured on environment level. If you do not want to make any changes, simply leave this blank. If you would like to use a different location for YARN application logs and Zeppelin Notebook's root directory, you can specify a different Google Cloud Storage location. Note that the specified Google Cloud Storage location must exist prior to Cloudera Data Hub cluster creation and that you must adjust the IAM policies created during environment's cloud storage setup to make sure that IDBroker has write access to this location.
- Path for YARN Application Logs property - This directory structure gets created automatically during cluster creation. You can customize it if you would like it be different than what is suggested by default.
- Path for Zeppelin Notebooks Root Directory property - This directory structure gets created automatically during cluster creation. You can customize it if you would like it to be different than what is suggested by default.

Related Information

Minimal setup for cloud storage

Recipes

The "Recipes" option allows you to select previously uploaded recipes (scripts that can be run pre or post cluster deployment) for each host group.

This option is available from the Cluster Extensions page of the advanced create cluster wizard. For more information, refer to the documentation linked below.

Related Information

Recipes

Cluster template overrides

You can specify custom configurations that override or append the properties in a built-in Cloudera Data Hub template or a custom template.

Overview

You can launch Cloudera Data Hub clusters from a set of pre-defined cluster templates created for prescriptive use cases. These cluster templates are a “shared resource” that define the list of services that will be installed on the Cloudera Data Hub, including their configurations.

For example, this is a portion of a default template:

```
{
  "services": [
    {
```

```
      "refName": "zookeeper",
      "serviceType": "ZOOKEEPER",
      "serviceConfigs": [
        {
          "name": "service_config_suppression_server_count_validator",
          "value": "true"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "zookeeper-SERVER-BASE",
          "roleType": "SERVER",
          "base": true
        }
      ]
    },
    {
      "refName": "hdfs",
      "serviceType": "HDFS",
      "serviceConfigs": [
        {
          "name": "hdfs_verify_ec_with_topology_enabled",
          "value": false
        },
        {
          "name": "core_site_safety_valve",
          "value": "<property><name>fs.s3a.buffer.dir</name><value>${env.LOC
AL_DIRS:-${hadoop.tmp.dir}}/s3a</value></property><property><name>fs.s3a.com
mitter.name</name><value>directory</value></property>"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "hdfs-NAMENODE-BASE",
          "roleType": "NAMENODE",
          "base": true,
          "configs": [
            {
              "name": "role_config_suppression_namenode_java_heapsize_mini
mum_validator",
              "value": "true"
            },
            {
              "name": "role_config_suppression_fs_trash_interval_minimum_
validator",
              "value": "true"
            },
            {
              "name": "fs_trash_interval",
              "value": "0"
            },
            {
              "name": "fs_trash_checkpoint_interval",
              "value": "0"
            },
            {
              "name": "erasure_coding_default_policy",
              "value": " "
            }
          ]
        },
        {
          "refName": "hdfs-SECONDARYNAMENODE-BASE",
          "roleType": "SECONDARYNAMENODE",
```

```
          "base": true
        },
        {
          "refName": "hdfs-DATANODE-BASE",
          "roleType": "DATANODE",
          "base": true
        },
        {
          "refName": "hdfs-BALANCER-BASE",
          "roleType": "BALANCER",
          "base": true
        },
        {
          "refName": "hdfs-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true,
          "configs": [
            {
              "name": "dfs_client_use_trash",
              "value": false
            },
            {
              "name": "role_config_suppression_hdfs_trash_disabled_validat
or",
              "value": "true"
            },
            {
              "name": "hdfs_client_env_safety_valve",
              "value": "HADOOP_OPTS=\"-Dorg.wildfly.openssl.path=/usr/lib64
 ${HADOOP_OPTS}\""
            }
          ]
        }
      ]
```

In this section of an example cluster template, two primary types of service configurations are visible: serviceConfigs and configurations for various roleConfigGroups, in addition to a special type of service configuration called safety_valves. You can read about the details of these configs in the *Cloudera Manager Configuration Properties* reference.

Oftentimes you may want to modify the serviceConfigs, roleConfigGroups, and safety_valves present in a default Cloudera Data Hub cluster template. Although you can create a custom cluster template by modifying the JSON of a default template, this process can be cumbersome and error-prone.

As an alternative to creating a custom template, you can specify custom configurations that override or append the properties in a default template. These custom configurations are saved as a shared resource called “cluster template overrides,” and can be used and re-used across Cloudera Data Hub clusters in different environments. As a shared resource, they do not need to be attached to a specific Cloudera Runtime version.

By using a default cluster template along with a cluster template override, you can create a customized Cloudera Data Hub cluster, along with receiving improvements to the default templates that are present in newer Cloudera Runtime releases. Cluster template overrides can be used to override or append properties present in both the identified types of service configurations, as well as safety valves.

Important: If you customize safety valves, the provided value will be appended to the already existing value for that safety valve. Custom configs will not override the existing value.

You are not limited to customizing the configs present in a default cluster template; you can add any valid configuration recognized by Cloudera Manager for an included service. New configurations are appended to the cluster template when the cluster is launched.

Cluster template overrides can be applied to custom templates as well. If you want to apply a cluster template override to a custom template that contains properties that are dynamically replaced during cluster creation, the cluster template override will override any dynamically-replaced properties when the two conflict.

Limitations

At present, there is no way to validate any individual property to ensure that it is valid and recognised by Cloudera Manager, so that it can be overridden to the desired value. Adding an incorrect config name can lead to errors while installing the Cloudera Manager template on the Cloudera Data Hub cluster, or the property could be ignored by Cloudera Manager entirely.

An invalid value for a particular property can also cause errors during the cluster creation process. Carefully review the *Cloudera Manager Configuration Properties* reference in regards to configurations that you want to customize.

Cluster template overrides are only for overriding/appending the serviceConfigs, roleConfigGroups, and safety_v alves in a Cloudera Data Hub template. Unlike creating a custom template, you can not add a service to the template.

Related Information

Cloudera Manager Configuration Properties

Building a custom Cloudera Data Hub cluster template

Custom Properties

Creating a cluster template override

From the Shared Resources menu, you can create a cluster template override to overwrite the default values of service and role configs in a Cloudera Data Hub template.

Procedure

1. In the Cloudera Management Console UI, Shared Resources Cluster Template Overrides .

2. Click Create Custom Override.

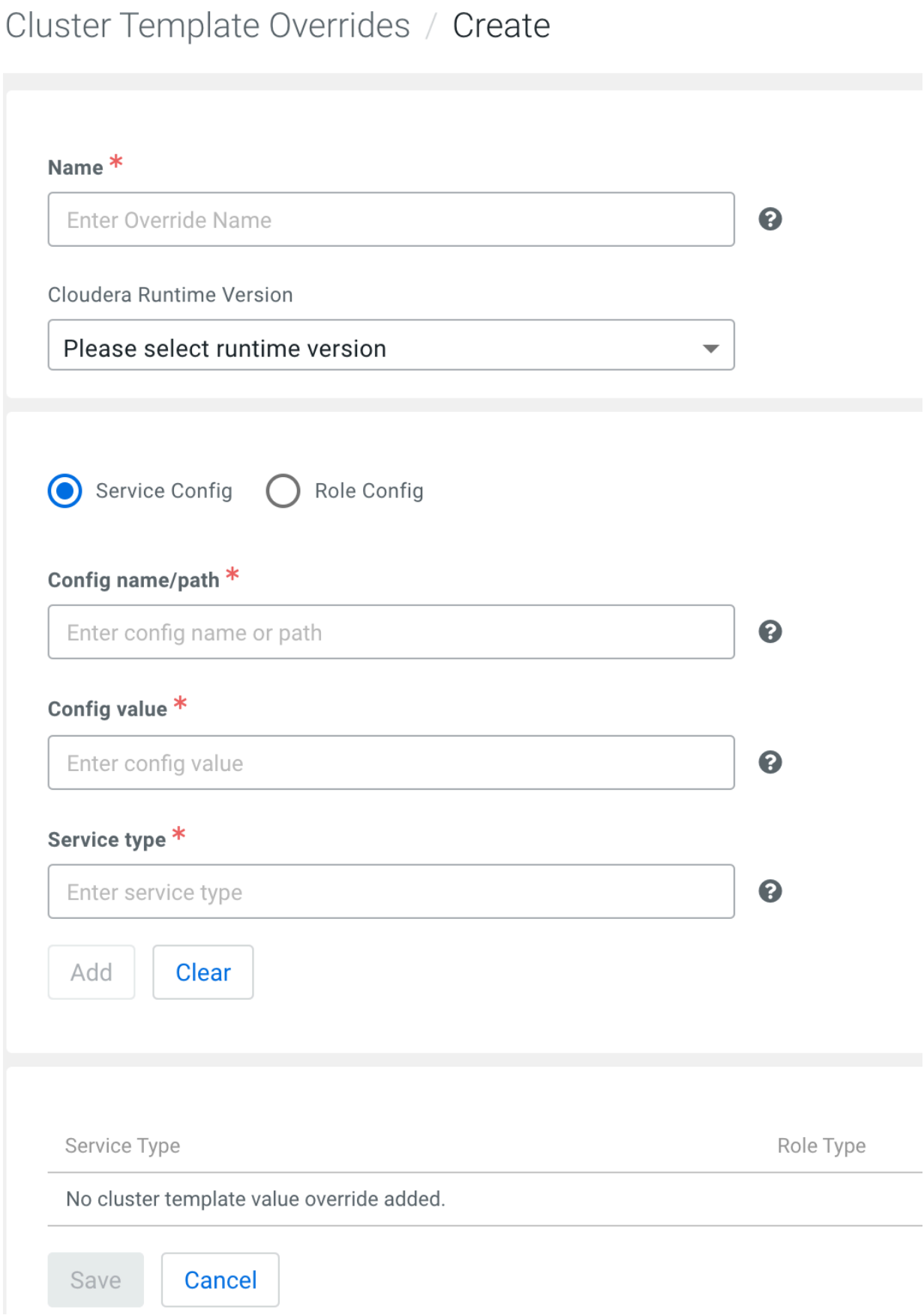

3. Provide the following information:

Property	Description
Name	Provide a unique name for the override.
Cloudera Runtime Version	Select a specific Cloudera Runtime version to which this override applies, or select “Not Runtime specific.”
Service Config / Role Config	Use the radio buttons to select whether you are customizing a service config or a role config.
Config Name/Path	Exact name/path of the config are you customizing.
Config Value	New value for the config. Ensure the value is valid.
Service Type	From the drop-down menu, select the service type for the service or role config that you are customizing.
Role Type	For role configurations: the role type as it appears in the cluster template that you will apply the override to (or any valid role configuration that is recognized by Cloudera Manager).

For example:

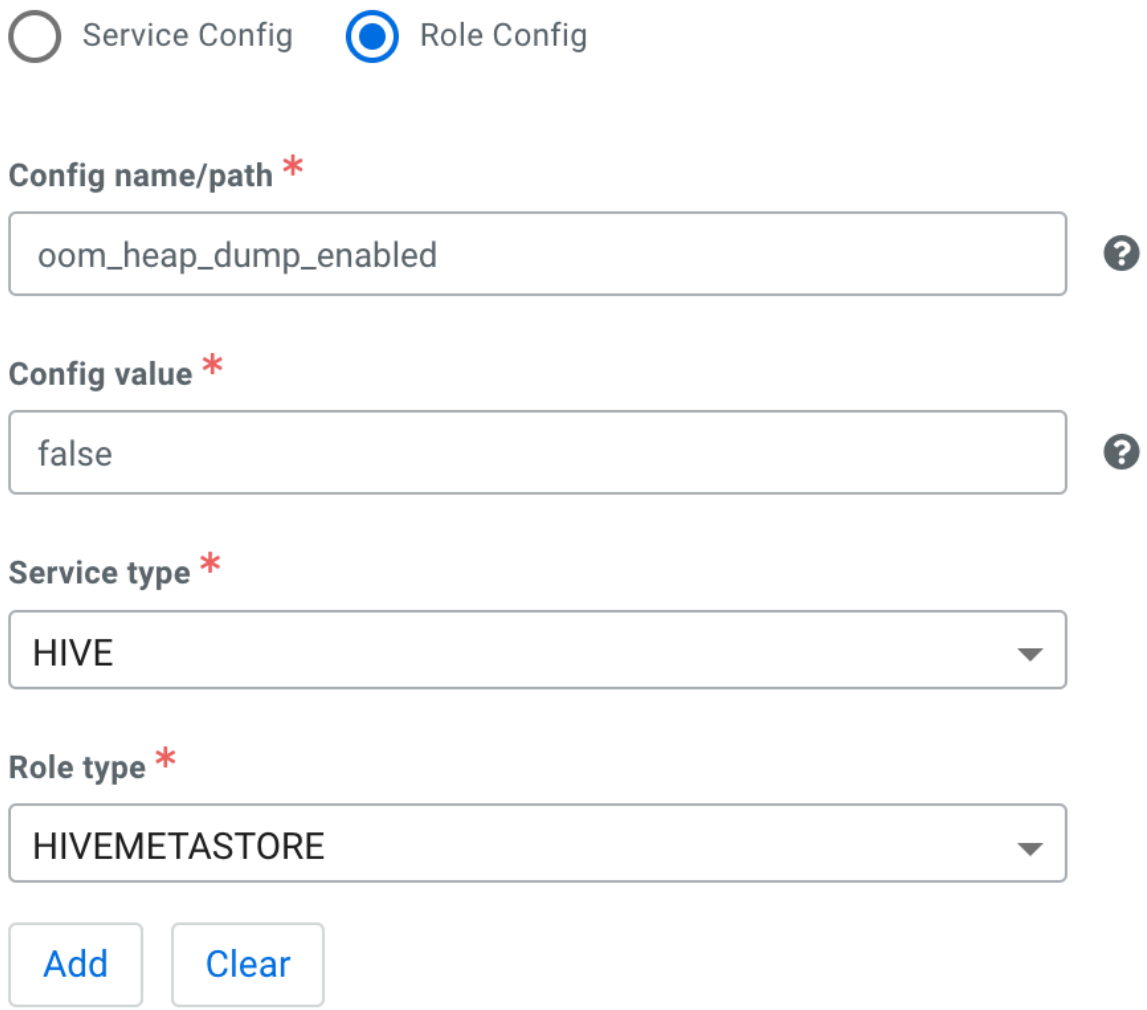

4. Click Add and then repeat the previous step (excluding the Name and Cloudera Runtime Version) for each service or role config that you want to customize in this template override.

5. When you are finished adding service and role configs to the template override, click Save. The override appears in the list of saved overrides, accessible from the Shared Resources menu.

Creating a Cloudera Data Hub cluster with a cluster template override

You can use a cluster template override that you previously created to provision a Cloudera Data Hub cluster.

Procedure

1. Click Data Hub Clusters Create Data Hub Cluster .
2. Enter the information required to create your cluster, then at the bottom of the cluster creation wizard, select Advanced OptionsCluster Extensions.
3. From the Cluster Templates Overrides drop-down box, select the template override that you previously created and want to apply when the cluster is provisioned. The only visible overrides are those with a Cloudera Runtime version matching the selected cluster definition version, or those that are Runtime-agnostic.

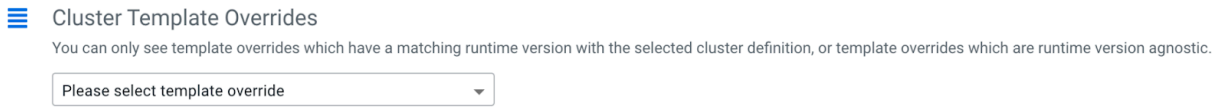

4. Click Provision Cluster.

Results

The Cloudera Data Hub cluster is provisioned using the service and role cofigs that you specified in the template override.

Cluster template override CLI commands

You can use the CDP CLI to create cluster template overrides, manage the cluster template overrides, and provision a Data Hub cluster using a cluster template override.

Table 1: Cluster Template Override CLI

Command	Description	Example
cdp datahub create-custom-configurations	Creates a new cluster template override shared resource with an indefinite amount of properties.	`cdp datahub create-custom-configurations --custom-configurations-name "test-custom-configs" --configurations configName="name",configValue="value",serviceType="hdfs" configName="name2",configValue="value2",roleType="role2",serviceType="hive_on_tez" .... --runtime-version "7.2.8"`
cdp datahub list-custom-configurations	Lists all the cluster template overrides for the account.	
cdp datahub describe-custom-configurations	Returns the name, CRN, the properties, and the platform version associated with the cluster template override resource.	`cdp datahub describe-custom-configurations --custom-configurations "test-overrides"`
cdp datahub delete-custom-configurations	Deletes multiple custom configs given their names or CRNs.	`cdp datahub delete-custom-configurations –custom-configurations "test-override" "test-override-2" ...`
cdp datahub create-aws-clustercdp datahub create-azure-clustercdp datahub create-gcp-cluster	Launches a Cloudera Data Hub cluster with custom configs from a cluster template override. The parameter for specifying a cluster template override is:--custom-configurations-name	`cdp datahub create-aws-cluster --cluster-name "de-dh1" --environment-name "local-aws-env" --cluster-template-name "7.2.8 - Data Engineering: Apache Spark, Apache Hive, Apache Ozie" --cluster-definition-name "7.2.8 - Data Engineering for AWS" --custom-configurations-name "test-override"`

Custom properties

The "Custom Properties "option allows you to set properties on a per-cluster basis.

This option is available from the Cluster Extensions page of the advanced create cluster wizard. For more information, refer to the documentation linked below.

Related Information

Custom Properties

Cluster deployment logs

Cluster deployment logs gather the same diagnostic (system and service) logs that are collected into your cloud storage location, but the Cloudera platform sends the logs to Cloudera engineering and support for troubleshooting purposes. You can apply configurable redaction rules on any sensitive data.

Cluster deployment logs are disabled by default. You can change this setting during environment creation or after environment creation, though the Cloudera platform will only collect logs for new deployments. You can also set a default behavior at the Cloudera account level, so that you won't have to enable the setting every time in the wizard. The logs are only gathered during deployment, which is approximately the first 20 minutes of cluster creation.

You can configure anonymization rules for log collection at the Cloudera account level. The rules are a list of rule objects with two fields: a regex pattern (PCRE) as the value, and a replacement string. Configure these rules to hide any sensitive data. By default, the rules hide email addresses and card numbers. This default behavior is the same as Cloudera Manager diagnostics collection.

The anonymization options are available under Global ResourcesTelemetry:

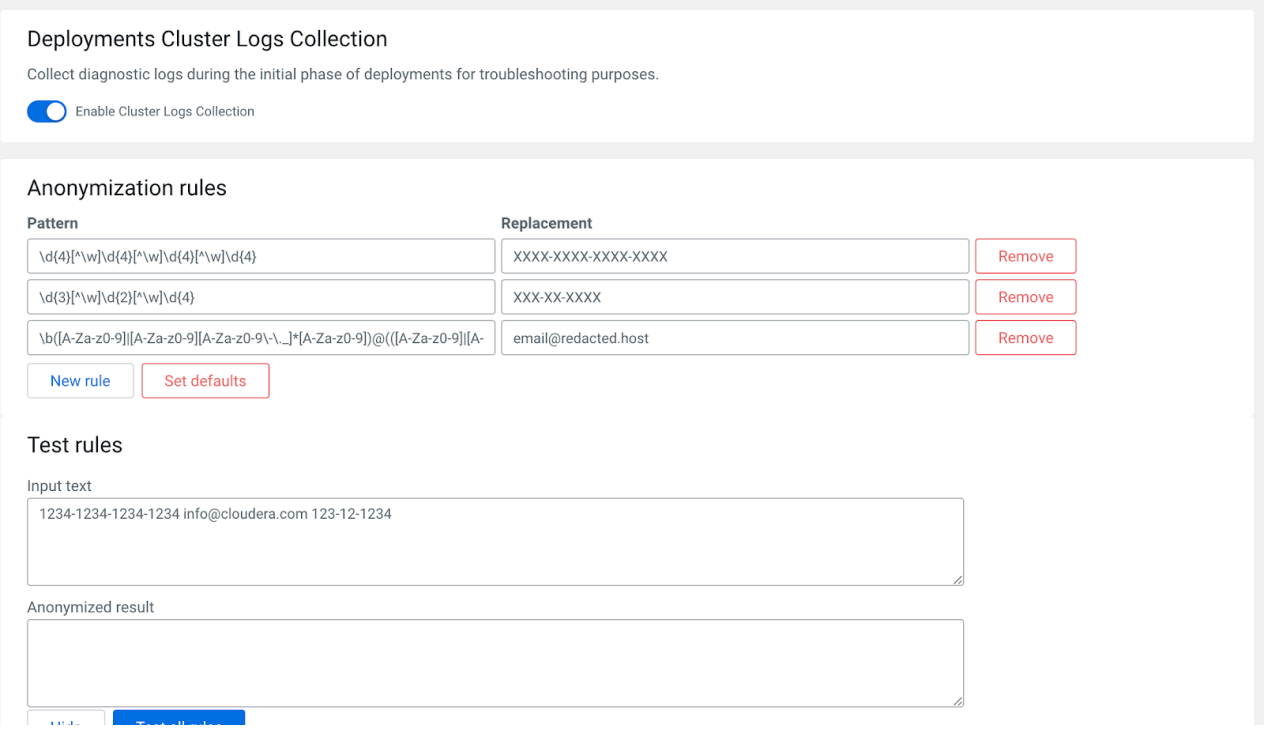

Currently, cluster deployment logs are collected only for Data Lake and Cloudera Data Hub cluster shapes.

Enabling/disabling cluster deployment log collection

You can enable cluster deployment log collection during environment creation. You can also enable or disable it after environment creation.

To enable cluster deployment log collection during environment creation:

1. Click the slider button under **Telemetry** in the **Region, Networking, Security, and Storage** page of environment registration.

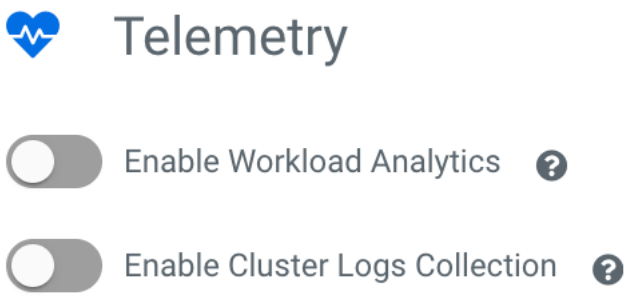

2. Select the checkbox to consent to data collection and click Enable.

To enable or disable cluster log deployment collection after environment creation:

1. Navigate to the **Environments** interface.
2. Select the desired environment, then click the Summary tab.
3. Scroll down to **Telemetry** and click the Edit button.
4. Click the slider button under Enable Deployment Cluster Logs Collection.
5. Select the checkbox to consent to data collection and click Enable.

To enable or disable cluster log deployment collection at the Cloudera account level (globally):

1. From the left navigation menu, click Global Settings and then Telemetry.
2. Click the slider button under Deployment Cluster Logs Collection.
3. Select the checkbox to consent to data collection and click Enable.

Cloudera Data Hub

Custom Properties

Date published: 2019-12-17
Date modified: 2023-06-27

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Custom properties.....4

Setting custom properties..... 4

Custom properties

Custom properties are configuration properties that can be set on a Cloudera Runtime cluster, but Cloudera Data Hub allows you to conveniently set these during cluster creation.

When working with on-premise clusters, you would typically set configuration properties once your cluster is already running by using Cloudera Manager or manually in a specific configuration file.

When you create a Cloudera Data Hub cluster, you can specify a list of configuration properties that should be set during cluster creation and Cloudera Data Hub sets these for you.

Note:

Alternatively, it is also possible to set configuration properties by using Cloudera Manager once the cluster is already running.

Setting custom properties

Cloudera Data Hub allows you to set custom properties dynamically for each cluster during cluster creation.

To set custom properties, do the following:

1. Prepare a cluster template that includes the appropriate mustache templating for setting custom properties. Use the "{{{ }}}" syntax. For example, this is a cluster template where the values for the Ranger plugin service name/repository and the HDFS directory used for the WAL will be dynamically replaced and set during cluster creation:

```
{
  "cdhVersion": "7.2.8",
  "displayName": "test-cluster",
  "services": [
    {
      "refName": "zookeeper",
      "serviceType": "ZOOKEEPER"
    },
    {
      "refName": "hdfs",
      "serviceType": "HDFS",
      "roleConfigGroups": [
        {
          "refName": "hdfs-NAMENODE-BASE",
          "roleType": "NAMENODE",
          "base": true
        },
        {
          "refName": "hdfs-SECONDARYNAMENODE-BASE",
          "roleType": "SECONDARYNAMENODE",
          "base": true
        },
        {
          "refName": "hdfs-DATANODE-BASE",
          "roleType": "DATANODE",
          "base": true
        },
        {
          "refName": "hdfs-FAILOVERCONTROLLER-BASE",
          "roleType": "FAILOVERCONTROLLER",
          "base": true
        },
        {
```

```
          "refName": "hdfs-JOURNALNODE-BASE",
          "roleType": "JOURNALNODE",
          "base": true
        },
        {
          "refName": "hdfs-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true
        }
      ]
    },
    {
      "refName": "hbase",
      "serviceType": "HBASE",
      "serviceConfigs": [
        {
          "name": "hbase_wal_dir",
          "value": "{{{wal.dir}}}",
          "autoConfig": false
        },
        {
          "name": "ranger_plugin_service_name",
          "value": "{{{test.cluster.rangersvc_name}}}",
          "autoConfig": false
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "hbase-MASTER-BASE",
          "roleType": "MASTER",
          "base": true
        },
        {
          "refName": "hbase-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true
        },
        {
          "refName": "hbase-REGIONSERVER-BASE",
          "roleType": "REGIONSERVER",
          "base": true
        },
        {
          "refName": "hbase-HBASETHRIFTSERVER-BASE",
          "roleType": "HBASETHRIFTSERVER",
          "base": true
        },
        {
          "refName": "hbase-HBASERESTSERVER-BASE",
          "roleType": "HBASERESTSERVER",
          "base": true
        }
      ]
    },
    {
      "refName": "phoenix",
      "serviceType": "PHOENIX",
      "roleConfigGroups": [
        {
          "refName": "phoenix-PHOENIX_QUERY_SERVER-BASE",
          "roleType": "PHOENIX_QUERY_SERVER",
          "base": true
        }
      ]
```

```
    },
    {
      "refName": "yarn",
      "serviceType": "YARN",
      "serviceConfigs": [],
      "roleConfigGroups": [
        {
          "refName": "yarn-RESOURCEMANAGER-BASE",
          "roleType": "RESOURCEMANAGER",
          "base": true
        },
        {
          "refName": "yarn-NODEMANAGER-BASE",
          "roleType": "NODEMANAGER",
          "base": true
        },
        {
          "refName": "yarn-JOBHISTORY-BASE",
          "roleType": "JOBHISTORY",
          "base": true
        },
        {
          "refName": "yarn-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true
        }
      ]
    },
    {
      "refName": "hue",
      "serviceType": "HUE",
      "roleConfigGroups": [
        {
          "refName": "hue-HUE_SERVER-BASE",
          "roleType": "HUE_SERVER",
          "base": true
        },
        {
          "refName": "hue-HUE_LOAD_BALANCER-BASE",
          "roleType": "HUE_LOAD_BALANCER",
          "base": true
        }
      ]
    }
  ],
  "hostTemplates": [
    {
      "refName": "gateway",
      "roleConfigGroupsRefNames": [
        "hbase-GATEWAY-BASE",
        "hdfs-GATEWAY-BASE",
        "yarn-GATEWAY-BASE",
        "hue-HUE_SERVER-BASE",
        "hue-HUE_LOAD_BALANCER-BASE"
      ]
    },
    {
      "refName": "master",
      "roleConfigGroupsRefNames": [
        "hbase-GATEWAY-BASE",
        "hbase-MASTER-BASE",
        "hdfs-FAILOVERCONTROLLER-BASE",
        "hdfs-GATEWAY-BASE",
        "hdfs-JOURNALNODE-BASE",
```

```
        "hdfs-NAMENODE-BASE",
        "yarn-GATEWAY-BASE",
        "zookeeper-SERVER-BASE"
      ]
    },
    {
      "refName": "leader",
      "roleConfigGroupsRefNames": [
        "hbase-GATEWAY-BASE",
        "hbase-HBASETHRIFTSERVER-BASE",
        "hbase-HBASERESTSERVER-BASE",
        "hdfs-GATEWAY-BASE",
        "hdfs-JOURNALNODE-BASE",
        "yarn-JOBHISTORY-BASE",
        "yarn-RESOURCEMANAGER-BASE",
        "yarn-GATEWAY-BASE",
        "zookeeper-SERVER-BASE"
      ]
    },
    {
      "refName": "worker",
      "roleConfigGroupsRefNames": [
        "hbase-GATEWAY-BASE",
        "hbase-REGIONSERVER-BASE",
        "hdfs-DATANODE-BASE",
        "hdfs-GATEWAY-BASE",
        "phoenix-PHOENIX_QUERY_SERVER-BASE",
        "yarn-GATEWAY-BASE",
        "yarn-NODEMANAGER-BASE"
      ]
    }
  ]
}
```

If you are customizing Cloudera Manager properties, refer to the Cloudera Manager Configuration Properties Reference. Cloudera-specific custom properties can be found in Recipe and cluster template parameters.

2. Prepare a JSON file that includes the custom property names and values that you want to set during cluster creation. In the template above, the custom property names that you provided ("wal.dir" and "test.cluster.rangers vc_name") will be replaced with the values that you give in this file. You can format the custom properties JSON in two different ways:

```
{
  "wal.dir": "/hadoopfs/fs1",
  "test.cluster.rangersvc_name": "test-cluster-ranger"
}
```

or

```
{
  "wal": {
    "dir": "/hadoopfs/fs1"
  },
  "test": {
    "cluster": {
      "rangersvc_name": "test-cluster-ranger"
    }
  }
}
```

Note: The JSON field should contain all of the required custom properties from the cluster template.

3. Register the cluster template that you created in the Cloudera Management Console. Click Shared Resources > Cluster Templates > Create Cluster Template. For more information on registering custom templates, see *Upload a cluster template*.
4. Once you have the template and the custom properties JSON ready, you can set the custom properties on a per-cluster basis by using the options available on the Cluster Extensions page in the create cluster wizard:
   a. From the left-side navigation menu, click Data Hub Clusters Create Data Hub and then the Custom radio button.
   b. Under Cluster Template, select the cluster template that you prepared and registered.
   c. Enter a name for the cluster and then click Advanced OptionsCluster Extensions.
   d. In the Custom Properties box, paste the custom properties JSON file that you prepared. During the cluster creation phase, the properties will be set to the values that you provided.

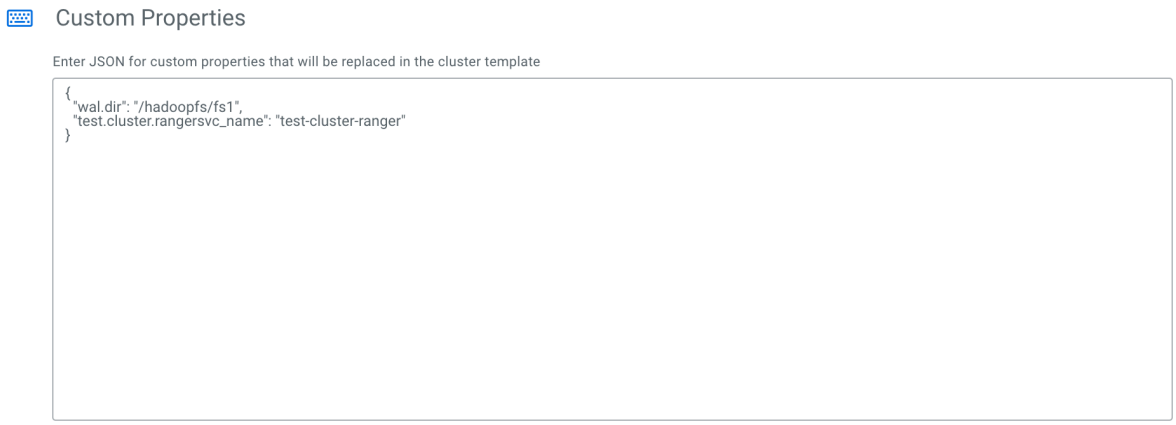

   e. Click Provision Cluster.

Note:

Instead of a cluster template, it is also possible to use a recipe for setting custom properties.

Related Information

Building a custom cluster template

Upload a cluster template

Cloudera Data Hub

EBS Volume Encryption on AWS

Date published: 2019-12-17
Date modified: 2023-06-27

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring encryption for EBS volumes of Cloudera Data Hub clusters on AWS.....4

Create a Cloudera Data Hub cluster with encrypted EBS volumes..... 4

Encrypting EBS volumes with a CMK generated from AWS CloudHSM.....5

Cross-account role permissions..... 5
Create a CloudHSM cluster..... 5
Initialize the CloudHSM cluster.....6
Activate the CloudHSM cluster..... 8
Create a CMK from CloudHSM..... 8
Encrypting data with a CMK on Cloudera Data Hub clusters..... 9

Configuring encryption for EBS volumes of Cloudera Data Hub clusters on AWS

You can configure encryption for Amazon Elastic Block Store (EBS) volumes used by the Cloudera Data Hub cluster's VM instances to store data.

Amazon offers the option to encrypt EBS volumes and RDS instances using the default key from Amazon's Key Management System (KMS) or using an external customer-managed KMS (CMK). By default, Cloudera Data Hub clusters use the same default key from Amazon's KMS or CMK as the parent environment but you have an option to pass a different CMK during Cloudera Data Hub cluster creation.

Encryption is configured for block devices and root devices. When encryption is configured for a given cluster, it is automatically applied to all the disk devices of any new VM instances added as a result of cluster scaling or repair.

Environment and Cloudera Data Hub encryption options

To learn about encryption options that Cloudera offers for Data Lake, FreeIPA, and Cloudera Data Hub clusters, refer to Environment and Cloudera Data Hub encryption options.

AWS prerequisites for using a CMK

You can use your existing AWS CMK or create a new AWS CMK. For detailed requirements, refer to AWS Requirements: Customer managed encryption keys.

Create a Cloudera Data Hub cluster with encrypted EBS volumes

EBS encryption can be configured for a Cloudera Data Hub cluster on the Hardware and Storage page of the advanced create cluster wizard in the Cloudera web interface.

Before you begin

Note the following:

- If you previously configured encryption on environment level, the CMK that was provided will be used by default for all Cloudera Data Hub clusters running in that environment. You can overwrite it for a specific Cloudera Data Hub by providing another CMK during Cloudera Data Hub cluster creation. If no CMK was provided during environment registration, a default key from Amazon's KMS is used by default to encrypt Cloudera Data Hub clusters running in this environment. See Adding a customer managed encryption key to a Cloudera environment running on AWS.
- During Cloudera Data Hub cluster creation, you can specify an encryption key Customer Managed Key (CMK).
- The Encryption configuration option is available per Cloudera Data Hub host group. The default setting is Encryption: Not encrypted. To enable encryption, perform the following steps in the create cluster wizard for all host groups for which you would like to use encryption.

Procedure

1. Under Instance Type you can see Encryption Supported next to all instance types for which encryption is supported. Ensure that encryption is supported for the instance type that you would like to use.
2. Click on the icon next to the chosen host group.
3. Enable the Enable Customer-Managed Keys toggle button.
4. Under Select Encryption key, select the CMK that you would like to use.

What to do next

Once the cluster is running, you can confirm that encryption is enabled by navigating to the details of the block devices or root devices in the EC2 console on AWS. The device should be marked as “Encrypted” and the “KMS Key ARN” is listed.

Encrypting EBS volumes with a CMK generated from AWS CloudHSM

This section provides steps for encrypting a Cloudera Data Hub cluster with a CMK generated from a custom key store on AWS CloudHSM.

AWS CloudHSM is a cloud-based hardware security module (HSM) that enables you to easily generate and use their own encryption keys in AWS. CloudHSM runs in your VPC and provides you with the flexibility to integrate with many applications.

When using a custom key store, the default KMS key store is replaced by a dedicated key store, which is hosted on a single tenant CloudHSM cluster on AWS. When a user generates a CMK (Customer Managed Key), the key material is generated and stored in the CloudHSM and KMS forwards all the data encryption requests to the CloudHSM.

For more general information about CloudHSM, see AWS Security Blog.

To set this up, you need to create, initialize, and activate the HSM cluster, and then create a CMK from HSM. Once done, you can use the CMK for encrypting Cloudera Data Hub clusters. Detailed steps are provided below.

Cross-account role permissions

If planning to use encryption, ensure that the cross-account IAM role used for the provisioning credential includes the permissions mentioned in Permissions for using encryption.

Create a CloudHSM cluster

First, create a CloudHSM cluster.

Procedure

1. Log in to your AWS account and navigate to CloudHSM.
2. In the top-right corner, verify that you are in the region where you would like to create the CloudHSM cluster.
3. Click on Create Cluster in the top corner.
4. On the “Cluster configuration” page:
   a. Select the VPC in which you wish to create the cluster. This should be the VPC where your Cloudera environment should be running.
   b. Under AZ(s), select the private subnets in the selected VPC (or create new private subnets If subnets are not present). Make sure that each subnet exists in a different availability zone.
5. Click Next.
6. On the “Backup retention” page, select a desired backup retention period (in days).
7. Click Next.
8. On the “Tags” page, add tags if required by your organization.
9. Review the cluster configuration and click on Create Cluster.

Results

This creates a cluster in an uninitialized state. You need to initialize the cluster before you can use it.

Note: When you create a cluster, AWS CloudHSM creates a security group with the name cloudhsm-cluster-clusterID-sg. This security group contains a preconfigured TCP rule that allows inbound and outbound communication within the cluster security group over ports 2223-2225. This rule allows HSMs in your cluster to communicate with each other.

Initialize the CloudHSM cluster

Now that you have created the CloudHSM cluster, you should initialize it.

Procedure

1. In the cluster created above, click on the Initialize button.
2. Select an Availability Zone (AZ) for the HSM and then choose Create.
3. Once the CSR is ready, you get a link to download the CSR. Click on Cluster CSR to download and save it.

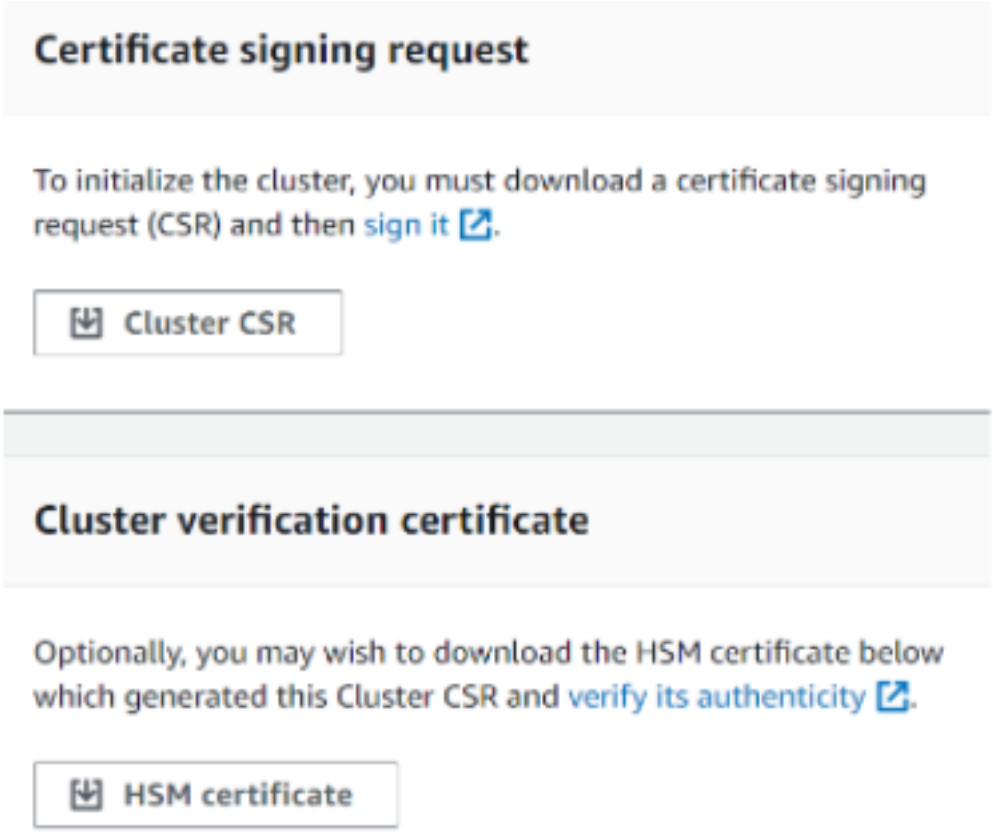

Leave the browser tab open, as you will need to continue the steps later.

4. At this point, you need to sign the CSR to initialize the cluster. To sign the CSR the below steps are required:
   a. Create a private key.

      You can create a private key using OpenSSL for development and testing:

      ```
      openssl genrsa -aes256 -out customerCA.key 2048
      ```

      For example:

      ```
      $ openssl genrsa -aes256 -out customerCA.key 2048
      Generating RSA private key, 2048 bit long modulus
      ........+++
      ............+++
      ```

```
e is 65537 (0x10001)
Enter pass phrase for customerCA.key:
Verifying - Enter pass phrase for customerCA.key:
```

For a production scenario AWS recommends creating the private key in a more secure fashion using an offline HSM or equivalent.

b. Use the private key to create a signing certificate.

You can use OpenSSL to sign the certificate. After running the command, you will be prompted to answer a few questions. The certificate will be valid for 10 years:

```
openssl req -new -x509 -days 3652 -key customerCA.key -out customerC
A.crt
```

This command creates a certificate file named customerCA.crt.

For example:

```
$ openssl req -new -x509 -days 3652 -key customerCA.key -out customerCA.
crt
Enter pass phrase for customerCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
 DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

c. Put the certificate generated in the previous step on every host from which you are planning to connect to your AWS CloudHSM cluster.

If you rename the file or store it in a path other than the root of your host, you should edit your client configuration file accordingly.

d. Sign your cluster CSR using the CSR downloaded earlier and download your HSM certificate.

The following example uses OpenSSL to sign the cluster's CSR. The example uses your private key and the self-signed certificate that you created in the previous step:

```
openssl x509 -req -days 3652 -in <cluster ID>_ClusterCsr.csr \
   -CA customerCA.crt \
   -CAkey customerCA.key \
   -CAcreateserial \
   -out <cluster ID>_CustomerHsmCertificate.crt
```

Example command and output:

```
$ openssl x509 -req -days 3652 -in <cluster ID>_ClusterCsr.csr \
   -CA customerCA.crt \
   -CAkey customerCA.key \
   -CAcreateserial \
   -out <cluster ID>_CustomerHsmCertificate.crt
Signature ok
```

```
subject=/C=US/ST=CA/O=Cavium/OU=N3FIPS/L=SanJose/CN=HSM:<HSM identifer
>:PARTN:<partition number>, for FIPS mode
Getting CA Private Key
Enter pass phrase for customerCA.key:
```

This command creates a file named <cluster ID>_CustomerHsmCertificate.crt. You will use this file as the signed certificate when you initialize the cluster.

5. Navigate back to your browser window and initialize the cluster. You will need to use the signed HSM certificate and also the actual certificate to initialize the cluster. Click Next on the “Sign certificate Signing request” page.
6. On the upload certificates page, do the following:
   a. Next to “Cluster certificate”, click on Upload file. Next, locate and select the HSM certificate. The file should be called <cluster ID>_CustomerHsmCertificate.crt.
   b. Next to “Issuing certificate”, choose Upload file. Then select your signing certificate. If you completed the steps in the previous section, the file should be called customerCA.crt.
   c. Choose Upload and initialize.

Activate the CloudHSM cluster

Once you have initialized the cluster, you need to activate it. Activating the cluster can be done through the command line using an Amazon EC2 instance with a CloudHSM client installed on it.

Before you begin

Install the CloudHSM client on an Amazon EC2 instance. The instructions to install the CloudHSM client on an Amazon EC2 instance are documented well in Install and Configure the AWS CloudHSM Client (Linux). Make sure that the EC2 instance is on the same VPC that our cluster is created on.

Procedure

1. Attach the EC2 instance to the security group that was created as a result of creating the HSM cluster.
2. To activate the CloudHSM cluster, you need to start the cloudhsm_mgmt_util through a command line utility, which will connect to the new HSM node and automatically enter the cloudHSM CLI. Follow the steps in Activate the Cluster (AWS docs).

What to do next

Once the Cloud HSM is active, you can start creating CMK’s from the custom key store.

Create a CMK from CloudHSM

Now that you set up CloudHSM cluster, you can create a CMK from CloudHSM.

Procedure

1. From the AWS console, navigate to the Key Management Service (KMS).
2. In the top-right corner, verify that you are in the region where you would like to create the CMK.
3. Select Customer Managed Keys (CMK) from the KMS page.
4. Click on Create Key in the top left corner.

5. In the “Configure key” wizard, select Symmetric and then in the “Advanced options” select the Custom key store (CloudHSM):

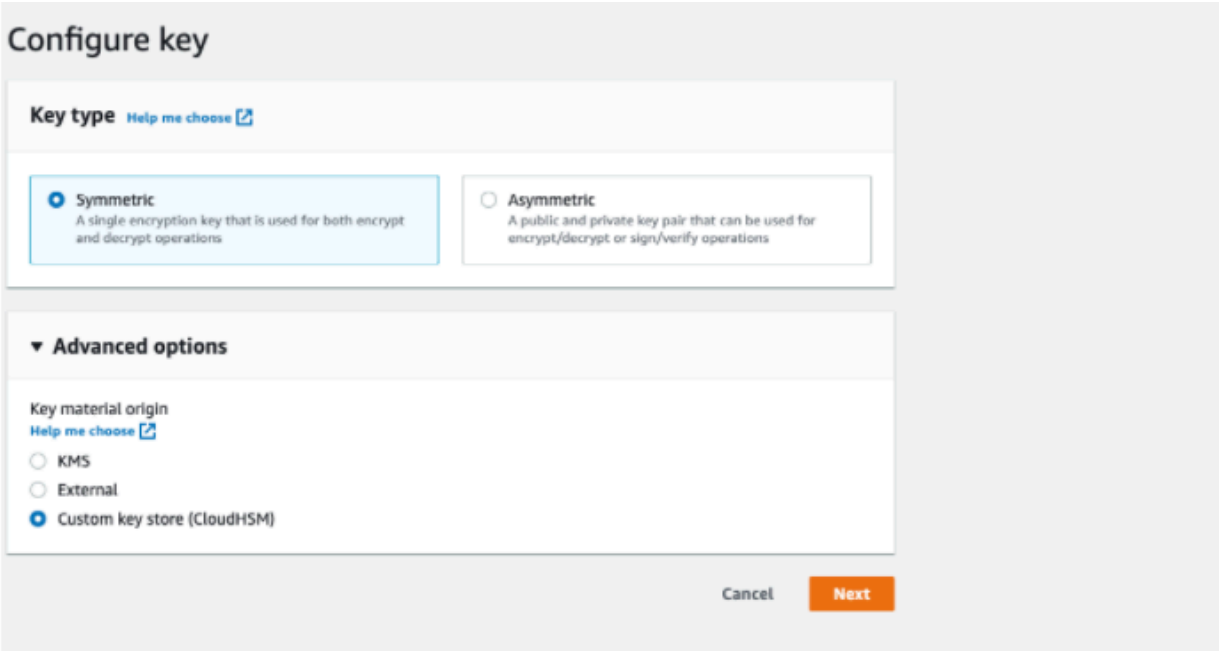

6. Click Next.
7. Add an alias to the key.
8. Click Next.
9. Provide required permissions to the key to encrypt and decrypt the data.
10. Click Next and then finish creating the key.

Results

Once you've performed these steps, you are ready to create Cloudera Data Hub clusters with the CMK that you just created.

Encrypting data with a CMK on Cloudera Data Hub clusters

Use these steps to set up encryption for your Cloudera Data Hub cluster data with the CMK created earlier.

Procedure

1. Create an environment in the same VPC where the CloudHSM is hosted. Steps to register a new environment in AWS cloud are documented in Register an AWS environment.
2. Once the environment is running, you can create Cloudera Data Hub clusters within the environment as described in Create a Cloudera Data Hub Cluster on AWS. Make sure to select the following in the Advanced options of the cluster creation wizard:
   a. Under Hardware and Storage, make sure to select the instance types that support encryption.
   b. Under Encryption Key, select the CMK that you have created earlier. This CMK will be used to encrypt your data.
   c. Repeat the process for all host groups where you would like to encrypt data.
3. Finish creating the Cloudera Data Hub cluster.

Results

Once the Cloudera Data Hub cluster is running, the specified CMK is used to encrypt data stored in the cluster.

Cloudera Data Hub

Managing Clusters

Date published: 2019-12-17
Date modified: 2023-06-27

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing clusters 5

Retry a cluster 5

Resize a cluster 5

Stop a cluster 7

Restart a cluster 8

Stopping and restarting cluster services 9

Performing manual repair 9

Terminate cluster 12

Terminate cluster nodes 12

Force terminate cluster 13

Deleting clusters when termination fails 13

Creating a multi-AZ Cloudera Data Hub cluster on AWS 14

Placement group support 15

Vertically scaling instances and disks 16

Vertically scaling instance types 16

Vertically scaling disks 18

Modifying disks 19

Adding disks 21

Deleting disks 21

Upgrading Cloudera Data Hub clusters 22

Support matrix for Cloudera Data Hub upgrades....26
Backing up before an upgrade....27
Performing a Cloudera Data Hub major/minor version cluster upgrade....31
Post-upgrade tasks....36
Adding configs manually....37
Upgrading Cloudera Operational Database clusters....39
Performing a Cloudera Data Hub service pack upgrade....40
Performing a Cloudera Data Hub OS upgrade....40
Upgrading a Cloudera Data Hub cluster with the CDP CLI....43
Cloudera Data Hub rolling upgrades....45
Cloudera Data Hub rolling upgrade limitations and issues....46
Troubleshooting upgrade....47

Autoscaling clusters....48
Autoscaling behavior....49
Configuring autoscaling....51
Using YARN queues with autoscaling....54
Configuring multiple YARN queues for autoscaling....54
Managing autoscaling....59
Manually recovering from load-based scaling failures....61
Autoscaling FAQ....62

Upgrading Data Lake/Cloudera Data Hub database....62
Database upgrade known limitations and troubleshooting....71
Installing Postgres 14 packages manually....72
Installing Postgres 11 packages manually....73

Rotating database certificates....74
SSL enforcement enabled....75
SSL enforcement disabled....78

Managing public and private certificates....79
Renewing private/host certificates on Data Lake and Cloudera Data Hub clusters....80
Manually renewing public certificates for Data Lake and Cloudera Data Hub clusters....82

More cluster management options....83

Managing clusters from CLI....83

Managing clusters

You can manage your Cloudera Data Hub clusters from the Cloudera Data Hub cluster dashboard in the Cloudera Management Console > Data Hub Clusters.

You can access cluster management options by clicking on the tile representing the cluster that you want to access. The actions available for your cluster are listed in the top right corner:

Retry a cluster

When stack provisioning or cluster creation fails, use the retry option to resume the process from the last failed step.

About this task

Required role: The DataHubAdmin or Owner roles at the scope of Cloudera Data Hub allow you to manage the Cloudera Data Hub cluster. Note that EnvironmentAdmin and Owner of the environment can also manage Cloudera Data Hub clusters.

Only failed stack or cluster creations can be retried. A retry can be initiated any number of times on a failed creation process.

In some cases the cause of a failed stack provisioning or cluster creation can be eliminated by simply retrying the process. For example, in case of a temporary network outage, a retry should be successful. In other cases, a manual modification is required before a retry can succeed. For example, if you are using a custom image but some configuration is missing, causing the process to fail, you must log in to the machine and fix the issue; Only after that you can retry the process.

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Data Hub Clusters.
3. Browse to cluster details.
4. Click Actions and select Retry.

   Note:

   Only failed stack or cluster creations can be retried, so the option is only available in these two cases.
5. Click Yes to confirm. The operation continues from the last failed step.

Resize a cluster

The resize option allows you to add or remove cluster nodes.

Before you begin

Ensure that you are using a cluster that can be resized.

Note: You can scale up and down Flow Management clusters since 7.2.11 and you can scale up and down Streams Messaging clusters since 7.2.12. For more information, see *Scaling up or down a NiFi cluster* and *Scaling Kafka broker nodes*.

About this task

Required role: The DataHubAdmin or Owner roles at the scope of Cloudera Data Hub allow you to manage the Cloudera Data Hub cluster. Note that EnvironmentAdmin and Owner of the environment can also manage Cloudera Data Hub clusters.

Unhealthy clusters can be resized, provided that the failed node is not a gateway node. If a new node fails to come up after scaling, the upscale operation will still complete successfully, and the failed nodes will be marked as "zombie" nodes. The stack is usable without these nodes, and you can clean them up manually. The "zombie" status indicates that the node provision has failed.

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Data Hub Clusters
3. Browse to cluster details.
4. Click Actions and select Resize. The cluster resize dialog is displayed.
5. Using the +/- controls, adjust the number of nodes for a chosen host group.

   Note:

   You can only modify one host group at a time. It is not possible to resize the Cloudera Manager server host group (by default, this is the master host group).

   Note:

   Downscaling the worker host group below 3 nodes is not possible on clusters that have at least 3 worker nodes by design.
6. Click Yes to confirm the scale-up/scale-down.

   While nodes are being added or removed, cluster status changes to “Update In Progress”. Once the operation has completed, cluster status changes back to “Running”. Messages similar to the following are written to even history:

```
Cluster scaled up
8/2/2019, 6:23:24 PM
Scaling up the cluster
8/2/2019, 6:14:05 PM
Stack successfully upscaled
8/2/2019, 6:14:04 PM
Mounting disks on new nodes
8/2/2019, 6:14:04 PM
Bootstrapping new nodes
8/2/2019, 6:13:48 PM
Infrastructure metadata extension finished
8/2/2019, 6:13:48 PM
Billing changed due to upscaling of cluster infrastructure
8/2/2019, 6:13:47 PM
Adding 1 new instances to the infrastructure
8/2/2019, 6:12:34 PM
```

Related Information

Scaling up or down a NiFi cluster

Scaling Kafka broker nodes

Stop a cluster

Cloudera Data Hub supports stopping and restarting clusters. Once a cluster is in the "Running" state, it can be stopped.

About this task

Required role: The DataHubAdmin or Owner roles at the scope of Cloudera Data Hub allow you to manage the Cloudera Data Hub cluster. Note that EnvironmentAdmin and Owner of the environment can also manage Cloudera Data Hub clusters.

Note: Most clusters that use ephemeral volumes cannot be stopped via Cloudera Data Hub. If a cluster has a host group that contains only yarn-NODEMANAGER and/or any "-GATEWAY" service roles, using ephemeral volumes as temporary storage for these host groups is supported, which allows for stop-start of the cluster.

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Data Hub Clusters
3. Browse to cluster details.
4. Click Stop to stop a currently running cluster.
5. Click Yes to confirm.
6. Your cluster status changes to “Stopping in progress” and then to “Stopped”.

   Messages similar to the following are written to the even history:

```
Synced instance states with the cloud provider.
8/2/2019, 6:29:39 PM
Updated metadata of instance i-053eb3a4e560d99e5 to stopped as the cloud
 provider reported it as stopped.
8/2/2019, 6:29:39 PM
Updated metadata of instance i-081925ba5152e36d2 to stopped as the cloud
 provider reported it as stopped.
8/2/2019, 6:29:38 PM
Updated metadata of instance i-00dc4cb6a7b9429ec to stopped as the cloud
provider reported it as stopped.
8/2/2019, 6:29:37 PM
Updated metadata of instance i-08649b04abc3b54a2 to stopped as the cloud p
rovider reported it as stopped.
8/2/2019, 6:29:36 PM
Billing stopped, the cluster and its infrastructure have been stopped
8/2/2019, 6:29:34 PM
Infrastructure successfully stopped
8/2/2019, 6:29:34 PM
Infrastructure is now stopping
8/2/2019, 6:28:51 PM
Cluster stopped
8/2/2019, 6:28:50 PM
Cloudera Manager services have been stopped.
8/2/2019, 6:28:49 PM
Stopping Cloudera Manager services.
8/2/2019, 6:27:28 PM
Stopping cluster
8/2/2019, 6:27:28 PM
Cluster infrastructure stop requested
```

```
8/2/2019, 6:27:26 PM
```

Once stopping the infrastructure has completed, you will see a Start option to restart your cluster. When a cluster is put in the “stopped” state, cluster VMs are given back to the cloud provider. To provision new VMs, restart the cluster.

Related Information

Restart a cluster

Restart a cluster

If your cluster is in the “Stopped” state, you can restart it by using the "Start" option.

About this task

Required role: The DataHubAdmin or Owner roles at the scope of Cloudera Data Hub allow you to manage the Cloudera Data Hub cluster. Note that EnvironmentAdmin and Owner of the environment can also manage Cloudera Data Hub clusters.

Note: Most clusters that use ephemeral volumes cannot be stopped/restarted via Cloudera Data Hub. If a cluster has a host group that contains only yarn-NODEMANAGER and/or any "-GATEWAY" service roles, using ephemeral volumes as temporary storage for these host groups is supported, which allows for stop-start of the cluster.

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Data Hub Clusters
3. Browse to cluster details.
4. Click Start. This option is only available when the cluster has been stopped.
5. Click Yes to confirm.

   Your cluster status changes to “Start in progress” and then once new VMs have been provisioned and registered, it changes to “Running”. Messages similar to the following are written to event history:

```
Synced instance states with the cloud provider.
8/2/2019, 6:33:37 PM
Updated metadata of instance ip-10-97-82-237.cloudera.site to running as
 the cloud provider reported it as running.
8/2/2019, 6:33:37 PM
Updated metadata of instance ip-10-97-83-214.cloudera.site to running as t
he cloud provider reported it as running.
8/2/2019, 6:33:36 PM
Updated metadata of instance ip-10-97-82-184.cloudera.site to running as
the cloud provider reported it as running.
8/2/2019, 6:33:36 PM
Updated metadata of instance ip-10-97-82-151.cloudera.site to running as
 the cloud provider reported it as running.
8/2/2019, 6:33:35 PM
Cluster started; Cluster manager ip:10.97.82.237
8/2/2019, 6:33:34 PM
Cloudera Manager services have been started.
8/2/2019, 6:33:33 PM
Starting Cloudera Manager services.
8/2/2019, 6:31:15 PM
Starting cluster
8/2/2019, 6:30:07 PM
```

```
Billing started, the cluster and its infrastructure have successfully been
 started
8/2/2019, 6:30:06 PM
Infrastructure successfully started
8/2/2019, 6:30:06 PM
Infrastructure is now starting
8/2/2019, 6:29:51 PM
Cluster start requested
8/2/2019, 6:29:51 PM
```

What to do next

Once your cluster status is “Running”, the cluster can be used.

Related Information

Stop a cluster

Stopping and restarting cluster services

If you would like to stop and restart services, you should use Cloudera Manager.

Performing manual repair

Manual repair should be performed on a cluster that has nodes marked as unhealthy.

Required role: The DataHubAdmin or Owner roles at the scope of Cloudera Data Hub allow you to manage the Cloudera Data Hub cluster. Note that EnvironmentAdmin and Owner of the environment can also manage Cloudera Data Hub clusters.

Note:

The manual repair option is available for all clusters and covers all nodes, including the Cloudera Manager server node (by default this is the master node). There is an exception: for clusters created before March 26, 2021, repair of the master node is enabled only if the Cloudera Data Hub cluster uses an external database. Ephemeral disks cannot be reattached; they are deleted and new disks are provisioned.

Important:

Due to their inherent ephemeral nature, Cloudera cannot preserve data that resides on the root disk. A manual repair operation on an unhealthy node will replace the disk mounted as root, which means all data on the root disk is lost.

To avoid the loss of important data, do not store it in the root disk. Instead, do one of the following:

- Store data that needs to persist beyond the lifetime of the cluster in S3 or ADLS Gen 2, depending on the platform in use.
- Store data that needs to survive a repair operation in /hadoopfs/fsN/ (where N is an integer), as long as it is not so large that it could crowd out components that use that location.

  For example, storing 1 GB of data in /hadoopfs/fs1/save_me would be an option to ensure that the data is available in a replacement node after a manual repair operation.

When a cluster has unhealthy nodes, a warning is displayed:

- Cluster tile on the cluster dashboard shows unhealthy nodes
- Nodes are marked as "UNHEALTHY" in the Hardware section
- Cluster's event history shows "Manual recovery is needed for the following failed nodes"

There are two ways to repair the failed nodes:

- Repair the failed nodes: (1) All non-ephemeral disks are detached from the failed nodes. (2) Failed nodes are removed (3) New nodes of the same type are provisioned. (4) The disks are attached to the new volumes, preserving the data.
- Delete the failed nodes: Failed nodes are deleted with their attached volumes.

If a node is marked as deleted from the cloud provider side, you must perform manual repair before restarting the cluster. If you do not perform the manual repair, the cluster will not restart.

You can perform manual repair from the Cloudera web interface or CLI.

To perform manual repair from Cloudera web interface:

1. Log in to the Cloudera web interface.
2. Navigate to Cloudera Management Console > Data Hub Clusters.
3. Browse to the cluster details.

4. Choose one of the following options:
   - To repair all of the failed nodes in a particular host group, select Actions > Repair and then select the host group to repair. Only one host group can be selected at a time:

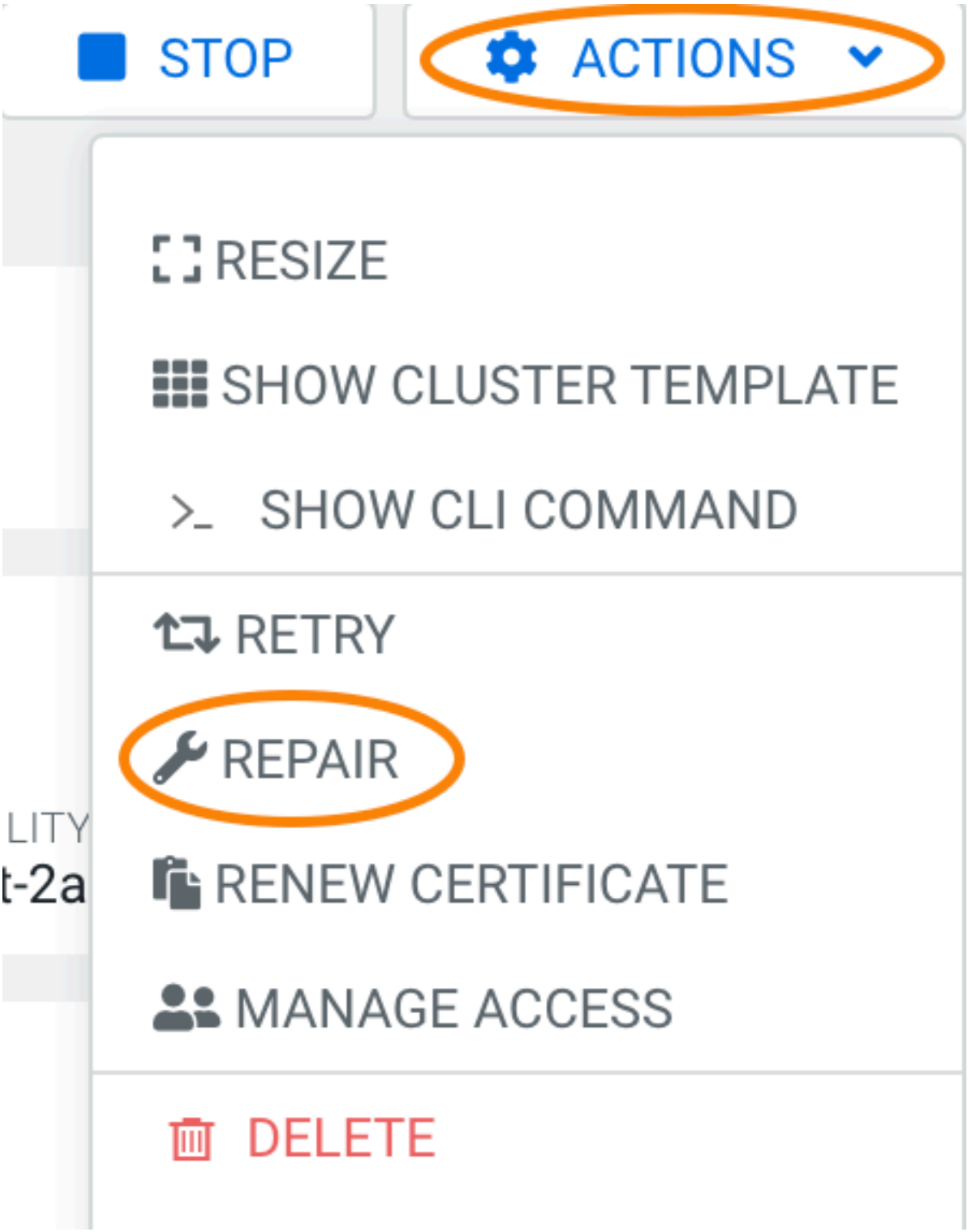

   - To repair a single node failure or select certain nodes within a host group to repair, select the Hardware tab and then the repair icon next to the host group that contains the failed node(s). Alternatively you can use the CDP CLI: cdp datahub repair-cluster help

- When you initiate a repair from the Hardware tab, you also have the option to delete any volumes attached to the instance. This can be useful if a volume is lost on the cloud provider side. To delete the attached volumes, select the Delete Volumes checkbox.

5. By default, unhealthy nodes are removed and then replaced. If you would like to just remove the nodes without replacing them, select Remove only.
6. Click Repair.
7. Once the recovery flow is completed, the cluster status changes to 'RUNNING'.

Related Information

Node repair

Terminate cluster

You can terminate any cluster managed by Cloudera Data Hub.

About this task

Required role: Owner at the scope of the Cloudera Data Hub (which provides delete access).

If you want to delete individual cluster nodes instead of terminating an entire cluster, see *Terminate cluster nodes*.

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Data Hub Clusters
3. Browse to cluster details.
4. Click Terminate.
5. Click Yes to confirm.

   All cluster-related resources will be deleted, unless the resources (such as networks and subnets) existed prior to cluster creation or are used by other VMs in which case they will be preserved.

Related Information

Force terminate cluster

Terminate cluster nodes

Terminate cluster nodes

Rather than terminating an entire Cloudera Data Hub cluster, you have the option to terminate one or more nodes of the cluster.

About this task

Required role: Owner at the scope of the Cloudera Data Hub (which provides delete access).

Procedure

1. In the Cloudera UI, navigate to Data Hub Clusters and click on the cluster details tile.
2. Click the Hardware tab.
3. Select the checkbox of the node(s) that you want to delete from the desired instance group.
4. Click the Delete (garbage can) icon.

5. Click Yes to confirm.

   Node deletion may fail if the Cloudera Management Console is unable to delete one or more of the cloud resources that were part of the node's infrastructure. Use the Force terminate option to remove the node entry from the Cloudera Management Console web UI. If you use this option, you must also check your cloud provider account to see if there are any resources that must be deleted manually.

Related Information

Performing manual repair

Force terminate cluster

The force terminate option can be used when cluster deletion fails.

Before you begin

Required role: Owner at the scope of the Cloudera Data Hub (which provides delete access).

Cluster deletion may fail if Cloudera Data Hub is unable to delete one or more of the cloud resources that were part of your cluster infrastructure. Use the Terminate > Force terminate option to remove the cluster entry from the Cloudera Data Hub web UI. If you use this option, you must also check your cloud provider account to see if there are any resources that must be deleted manually.

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Data Hub Clusters
3. Browse to cluster details.
4. Click Terminate.
5. Check Force terminate.
6. Click Yes to confirm.

   When terminating a cluster with Kerberos enabled, you have an option to disable Kerberos prior to cluster termination. This option removes any cluster-related principals from the KDC.
7. This deletes the cluster tile from the UI.
8. Log in to your cloud provider account and manually delete any resources that failed to be deleted.

What to do next

If force-termination fails, you may have to delete the cluster manually.

Related Information

Deleting clusters when termination fails

Deleting clusters when termination fails

Upon cluster termination, Cloudera Data Hub only terminates the resources that it created. It does not terminate any resources (such as networks, subnets, roles, and so on) which existed prior to cluster creation.

The proper way to delete clusters is by using the Terminate option available from the cluster details in the Cloudera Data Hub service console. If the terminate process fails, try the Terminate > Force terminate option. If the force termination does not delete all cluster resources, delete the resources manually:

- To find the VMs, click on the links available in the Cloudera Management Console > Data Hub Clusters cluster details > Hardware section.

  If you terminate instances on the cloud provider without using Cloudera Data Hub, during the sync operation, Cloudera Data Hub realizes that instances have been removed and marks the instances as DELETED_ON_PROVIDER_SIDE.

  On AWS and GCP, Cloudera Data Hub automatically removes all VM dependencies such as attached disks, network interfaces, public IPs and storage accounts. On Azure, you must manually delete the disks created for that instance and release the public IP assigned to that instance.
- To find the network and subnets, see the Cluster Information pane in the cluster details.

Creating a multi-AZ Cloudera Data Hub cluster on AWS

By default, Cloudera provisions Cloudera Data Hub clusters in a single AWS availability zone (AZ), but you can optionally choose to deploy them across multiple availability zones (multi-AZ).

For general information about multi-AZ in Cloudera, refer to Deploying Cloudera in multiple AWS availability zones.

Create a multi-AZ Cloudera Data Hub cluster

You can create a multi-AZ Data Hub via Cloudera UI or CDP CLI within an environment. Note that the CLI allows you to manually specify subnets, which is not possible via the UI.

Steps

For CDP UI

Create your Cloudera Data Hub cluster as usual. In the Advanced Options > Network and Availability, you select the multiple subnets across which the Cloudera Data Hub cluster is to be provisioned. If multiple subnets are selected, the Cloudera Manager node group will only have one subnet for each AZ; All other nodes will have all the selected subnets.

For CDP CLI

When creating a multi-AZ Cloudera Data Hub cluster, the creation request should contain the --multi-az option. For example:

```
cdp datahub create-aws-cluster \
--cluster-name tb-datamart-multiaz \
--environment-name tb-multiaz-env \
--cluster-template-name "7.2.15 - COD Edge Node" \
--instance-groups nodeCount=3,instanceGroupName=coordinator,instanceGro
upType=CORE,instanceType=r5d.4xlarge,rootVolumeSize=100,attachedVolumeCo
nfiguration=\[\{volumeSize=300,volumeCount=2,volumeType=ephemeral\}\],re
coveryMode=MANUAL,volumeEncryption=\{enableEncryption=false\} nodeCount=
1,instanceGroupName=master,instanceGroupType=GATEWAY,instanceType=r5.4xl
arge,rootVolumeSize=100,attachedVolumeConfiguration=\[\{volumeSize=100,v
olumeCount=1,volumeType=standard\}\],recoveryMode=MANUAL,volumeEncryption=
\{enableEncryption=false\} nodeCount=2,instanceGroupName=executor,instance
GroupType=CORE,instanceType=r5d.4xlarge,rootVolumeSize=100,attachedVolum
eConfiguration=\[\{volumeSize=300,volumeCount=2,volumeType=ephemeral\}\]
,recoveryMode=MANUAL,volumeEncryption=\{enableEncryption=false\} \
--image id=23e20852-7865-4980-a045-539296340b55,catalogName=cdp-default \
--profile mowdev \
--multi-az
```

By default, Cloudera Data Hub instances are distributed across the subnets provided during environment registration. If you prefer to specify the subnets manually, you can add the --subnet-ids option and specify subnets where you would like to deploy the hosts, overwriting the default node distribution. For example:

```
--subnet-ids "subnet-013855b2fc32c2cd8" "subnet-02b9054ec829374fe" "subn
et-085c9ff36b38c0b35"
```

If multiple subnets are provided, the Cloudera Manager node group will only have one subnet for each AZ; All other nodes will have all the selected subnets.

The subnets passed via the --subnet-ids option will be applied to all cluster instance groups. If you would like to specify custom subnet ID lists for any given instance group, you can pass them in the subnetIds inside the --instan ce-groups option.

Scaling a multi-AZ Cloudera Data Hub cluster

If there is an availability zone that is offline, Cloudera may not detect the outage. In such a case, if you know that a certain availability zone is offline, you can scale your cluster and manually specify where the new nodes should be provisioned.

When scaling a multi-AZ cluster, Cloudera automatically distributes the new nodes in a round-robin fashion across all available availability zones, prioritizing the least used availability zones. If you prefer to manually control the distribution of nodes across subnets during Cloudera Data Hub scaling, the desired availability zones can be controlled via the related subnets during upscales with the optional --preferred-subnet-ids field.

For example:

```
cdp datahub scale-cluster --cluster-name tb-datamart-multiaz \
  --instance-group-name "coordinator" \
  --instance-group-desired-count 5 \
  --preferred-subnet-ids "subnet-013855b2fc32c2cd8"
 "subnet-02b9054ec829374fe" "subnet-085c9ff36b38c0b35"
```

If you manually specify the subnets in this manner, this overwrites the default behavior.

Placement group support

You can configure a Cloudera Data Hub cluster on AWS for placement group support. Placement groups support placing VM instances across different physical hardware within a single availability zone. Placement groups can help ensure cluster availability in the event of a physical hardware failure within an availability zone.

In a Cloudera Data Hub cluster, each host group can be associated with a placement group, of which there are three supported types: partition, spread, or cluster. Cloudera recommends using the partition type as the default for all host groups. For a partition placement group, the partition count will always be 2 and is not configurable.

Note: Only specific host groups, such as the master host group of the Data Engineering HA template, can be configured with the spread placement group. The spread placement group has a maximum instance limit of seven instances per placement group, per availability zone. Configuring multiple spread placement groups does not guarantee hardware isolation across the placement group.

Configuring placement group support requires adding custom properties to a cluster definition at the host group level. You can define the strategy as “NONE,” “PARTITION,” “SPREAD,” or “CLUSTER.”

For example:

```
{
  "environmentName": "sample-env",
  "instanceGroups": [
```

```
    {
      "nodeCount": 1,
      "name": "master",
      "type": "GATEWAY",
      "recoveryMode": "MANUAL",
      "template": {
        "aws": {
          "encryption": {
            "type": "NONE",
            "key": null
          },
          "placementGroup": {
            "strategy": "PARTITION"
          }
        },
        "instanceType": "m5.2xlarge",
        "rootVolume": {
          "size": 50
        },
        "attachedVolumes": [
          {
            "size": 100,
            "count": 1,
            "type": "standard"
          }
        ],
        "cloudPlatform": "AWS"
      },
```

Placement groups have a number of rules and limitations. Importantly, it is possible to run out of placement groups within an availability zone. Refer to the AWS documentation for detailed rules and limitations.

Vertically scaling instances and disks

If necessary, you can select a larger or smaller instance type for a Cloudera Data Hub or Data Lake cluster after it has been deployed, or add, delete, or modify attached disks.

Vertically scaling instance types

If necessary, you can select a larger or smaller instance type for a Cloudera Data Hub or Data Lake cluster after it has been deployed in AWS and Azure.

Before you begin

You must stop the Data Lake or Cloudera Data Hub cluster before you vertically scale any of the instances.

About this task

Selecting a larger instance type adds more vCPU and/or RAM to your instances. Instances can be scaled both up and down, but scaling down to a smaller size requires 4 CPU and a minimum of 4 GB memory.

If you are using an instance without ephemeral disks, you can scale up or down to a new instance with ephemeral disks; however, the reverse is not supported. You cannot start with an instance with ephemeral disks and move to an instance without ephemeral disks.

Note:

You cannot scale down to a target instance type that has fewer ephemeral volumes than the current instance type.

Vertical scaling is supported on AWS and Azure.

Note: Vertical scaling to Azure v5 instances is not supported and results in the following error:

```
Unable to resize since changing from resource disk to non-resource d
isk VM size and vice-versa is not allowed.
```

For more information, see the Azure VM sizes with no local temporary disk documentation.

Data Lake and Cloudera Data Hub instances must be stopped before scaling. See Change the instance type in AWS documentation for more information.

For information on vertically scaling FreeIPA, see Vertically scale FreeIPA instances.

Procedure

1. In the Cloudera main navigation menu, click Data Hub or Data Lakes and select the cluster that requires a larger instance type.
2. Scroll to the bottom of the page and click the **Nodes** tab.
3. Click the Vertical Scaling icon on the top right of the host group that you want to scale.

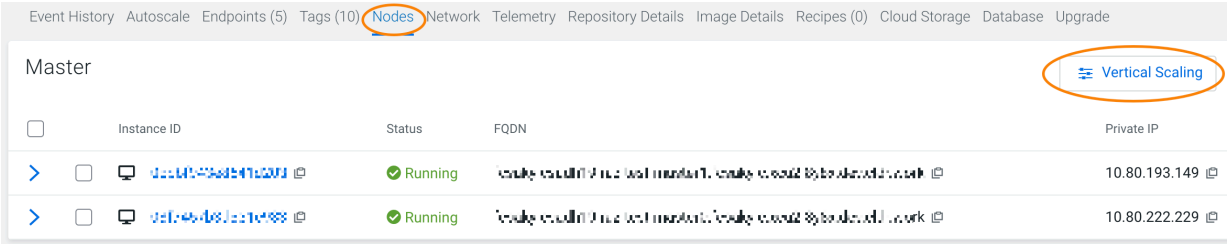

4. Select a larger instance type from the drop-down menu of suggested instance types.

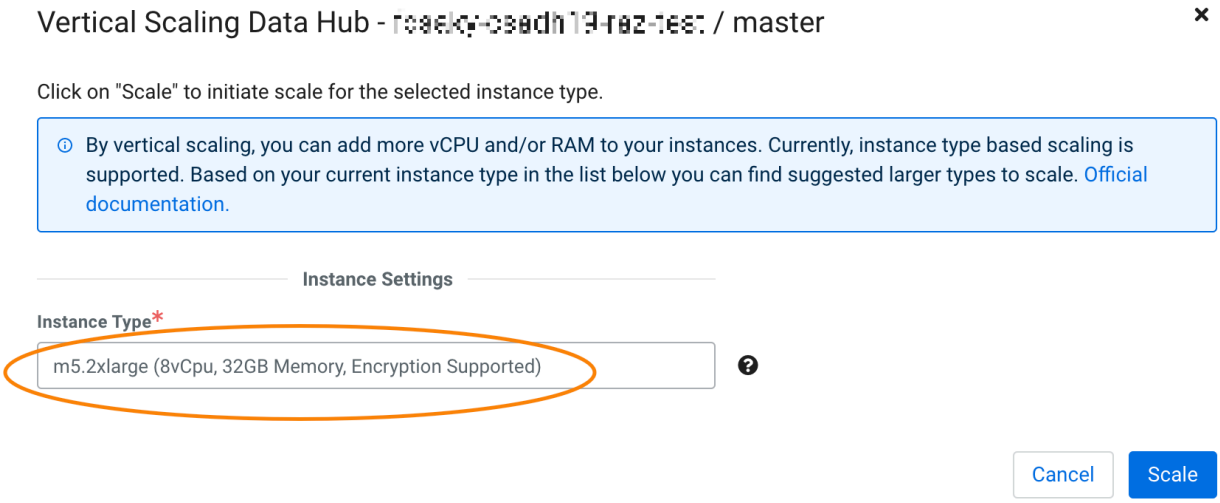

5. Click Scale. You can monitor the action from the **Event History** tab.

   Alternatively, you can use the CDP CLI to select a new instance for the Data Lake or Cloudera Data Hub cluster:

   Data Lake cluster:

```
cdp datalake start-datalake-vertical-scaling
--datalake <your-data-lake-name-or-its-crn>
--group <master>
```

```
--instance-template instanceType="<m5.4xlarge>"
```

Cloudera Data Hub cluster:

```
cdp datahub start-cluster-vertical-scaling
--datahub <your-data-hub-name-or-its-crn>
--group <master>
--instance-template instanceType="<m5.4xlarge>"
```

What to do next

After you have vertically scaled the cluster, configure the services on the cluster to use the additional or reduced resources/memory.

Vertically scaling disks

With the growing amount of data, it might be necessary to add, delete, or modify disks attached to Data Lake or Cloudera Data Hub clusters in AWS.

There are many clusters that are deployed with standard magnetic storage. With the growing lineage data, these disks are running out of space on core nodes. These need to be moved to General Purpose SSDs (gp2/gp3 on AWS) and or resized to a bigger disk.

The disks attached to the Data Lake and Cloudera Data Hub clusters can be changed or resized in AWS without downtime.

Limitations

When using this preview feature, be aware of the following limitations:

- This feature is only available for AWS.
- The disks can only be resized up, meaning you cannot reduce the size of an attached block storage. If there are multiple disks of different sizes, the size of all the disks attached to the instances in the group that are smaller or lesser than the requested size will be increased to the requested size.
- This feature will only resize additional block storages in an instance and not the root volume.
- Clusters and cluster services must be in running state before disk vertical scaling is performed.
- Current implementation does not support this feature through Cloudera UI; It is available only through Beta CDP CLI. To install Beta Cloudera, refer to Installing Beta CDP CLI.
- The disk modification feature on AWS can only be used once in 6 hours. This is a limitation on the AWS side.

Permissions

This feature requires the following permissions to be added to the cross-account policy described in Cross-account access IAM role.

- ec2:ModifyVolume
- ec2:DescribeVolumesModifications
- ec2:DescribeVolumeStatus

The following table explains why Cloudera needs these permissions:

Permission	Description
ec2:ModifyVolume	It is required to modify the volume attributes such as type, size and IOPS capacity. Without this, volume modifications cannot be performed by Cloudera.
ec2:DescribeVolumesModifications	It is required to verify whether the volume modifications performed by Cloudera were successful. Only upon successful modification, other steps like resizing will be done.

Permission	Description
ec2:DescribeVolumeStatus	This is required to make sure that the volume being modified is attached to an instance and not an orphaned volume.

Modifying disks

The disk volumes attached to the instances in a host group can be modified using CLI commands.

Modifying disks using CDP CLI

Use the following Beta CDP CLI command to modify the volumes attached to the instances in a host group. Replace the placeholders with actual values. For example *[***DATALAKE-NAME***]* should be replaced with an actual name. As part of this update, the instance-template parameter in the vertical scaling command has been made optional. But one of the instance-template or disk-options have to be provided.

Additional parameters for vertical scaling have been added to both the datalake and datahub commands. The modification request is sent as part of the --disk-options parameter.

The *[***VOLUME-TYPE***]* placeholder is for the type of volume the disks are being modified to. It is an optional field and should be added if the volume type has to be modified, in which case the type will not be modified.

The *[***SIZE***]* placeholder is optional as well and is for the size the disks are being increased to in GB. Specify only the integer value.

```
//DATA LAKE
cdp datalake start-datalake-vertical-scaling
          --datalake [***DATALAKE-NAME***]
          --group [***INSTANCE-GROUP-NAME***]
          --disk-options modifyDisks="{volumeType=\"[***VOLUME-TYPE***]\"
,size=[***SIZE***]}"



//DATA HUB

cdp datahub start-cluster-vertical-scaling
          --datahub [***DATAHUB-NAME***]
          --group [***INSTANCE-GROUP-NAME***]
          --disk-options modifyDisks="{volumeType=\"[***VOLUME-TYPE***]\",
size=[***SIZE***]}"
```

Verifying that modification is complete

The change to the disk can be verified through Cloudera UI, AWS console, or by logging into the instances directly, after the flow is completed in the Event History.

The following is a screenshot from Event History showing the completion of the disk update:

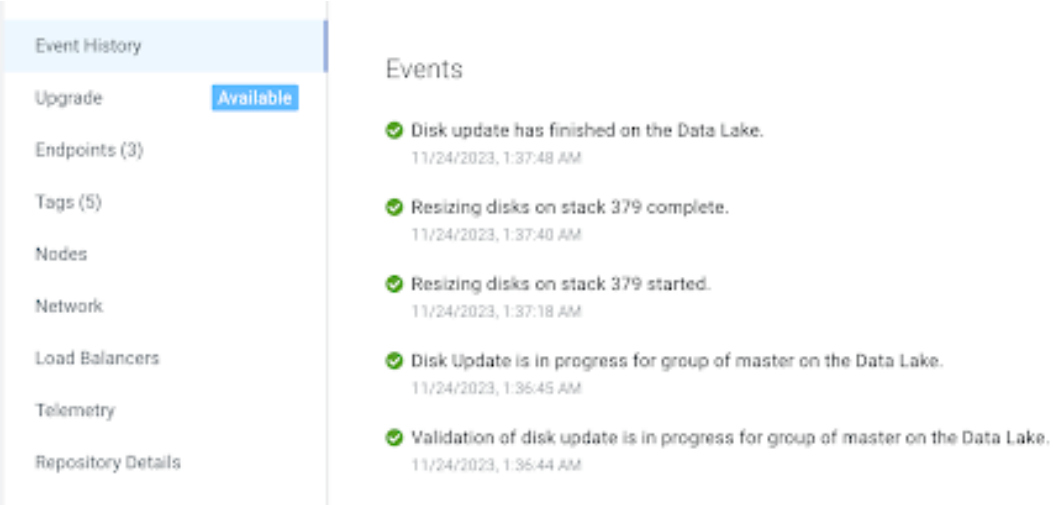

Verifying disk size/type in Cloudera UI

1. Navigate to the Cloudera Management Console Data Lake or Cloudera Management Console Cloudera Management Console
2. Click into the Data Lake or Cloudera Data Hub that was modified.
3. Click Nodes in the left hand tree.
4. Open the accordion of the instance group that was modified.
5. Open the Storage Settings accordion in any of the instances in the group.

The Storage Type and Volume Size are updated based on the request.

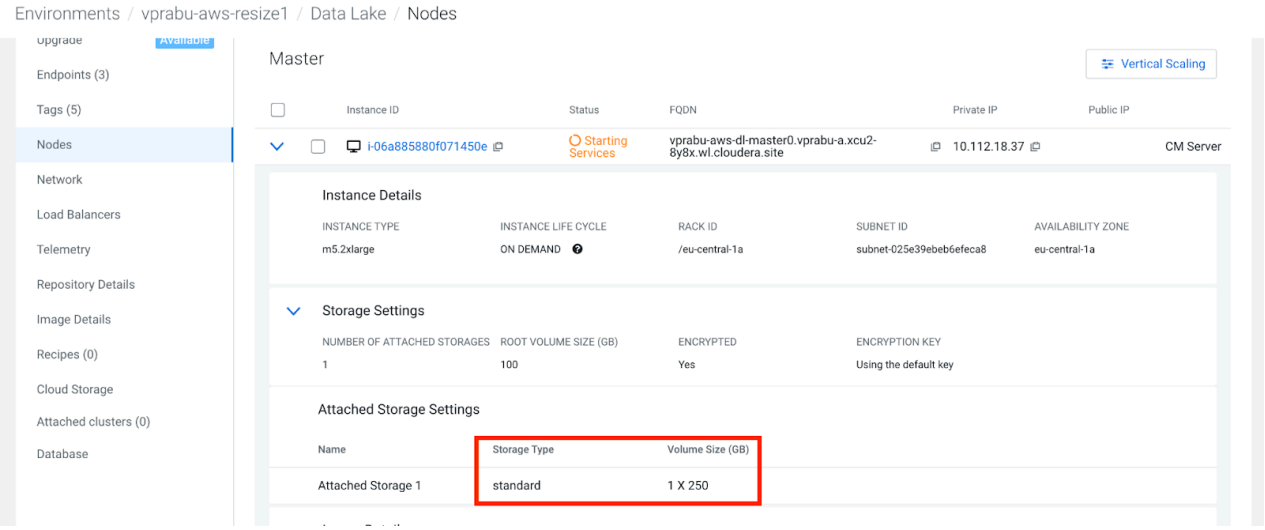

Verifying disk size/type in AWS Console

1. Navigate to the Cloudera Management Console Data Lake or Cloudera Management Console Cloudera Management Console
2. Click into the Data Lake or Cloudera Data Hub that was modified.
3. Click Nodes in the left hand tree.
4. Click on the AWS link for the instance and log into AWS Console.
5. Select the Storage tab for the instance.

The volume size is updated as per the request.

You can also click into an individual volume to make sure the type and size are modified correctly.

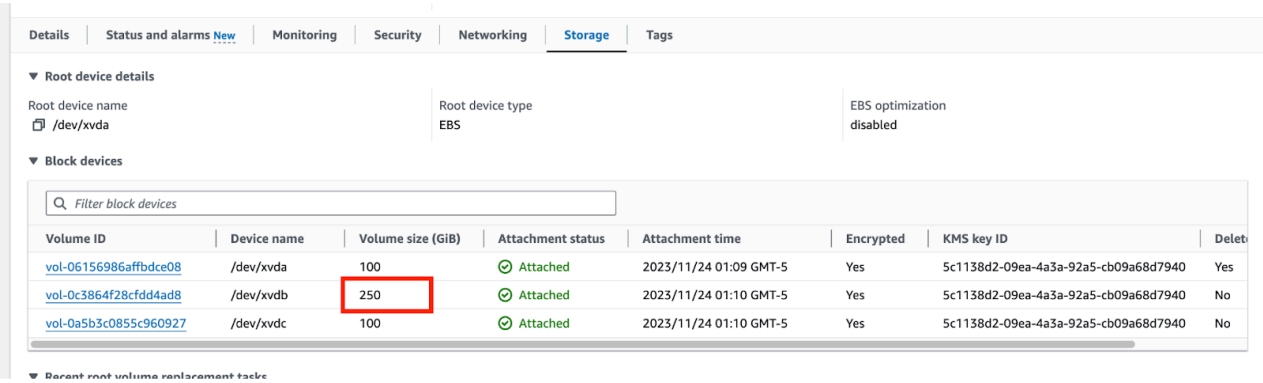

Adding disks

Additional volumes can be added to the instances in a host group using CLI commands.

In some cases when customers run out of memory, it is cheaper to add an additional volume instead of resizing to a bigger volume. This feature allows for adding additional block storages to an instance group with minimum interruption. Only the services running on the instance group are stopped, as the additional volume has to be mounted and configured for use on the instances.

Based on the request, ‘N’ number of disks of the same type and size can be added to the instance group.

Adding disks using CDP CLI

Use the following CDP CLI Beta command to add additional volumes attached to the instances in a host group. Replace the placeholders with actual values. For example, *[***DATALAKE-NAME***]* should be replaced with an actual name. As part of this update, the instance-template parameter in the vertical scaling command has been made optional. But one of the -- instance-template or -- disk-options have to be provided.

The addDisks request is sent as part of the --disk-options parameter. All parameters for the add disk input are required.

The *[***NUMBER_OF_DISKS***]* placeholder is for the number of volumes being added.

The *[***VOLUME-TYPE***]* placeholder is for the type of volume being added.

The *[***SIZE***]* placeholder is for the size the disks being added in GB. Specify only the integer value.

The cloudVolumeUsageType is the purpose for which the disk is being added; it is an enum field with either “GEN ERAL” or “DATABASE” value.

```
//DATALAKE

cdp datalake start-datalake-vertical-scaling --datalake=[***DATALAKE_NAME***
] --group=[***DATALAKE_INSTANCE_GROUP_NAME***] --disk-options addDisks="{num
berOfDisks=[***NUMBER_OF_DISKS_TO_ADD***], volumeType=\"[***TYPE_OF_COLUME_T
O_ADD***]\",size=[***SIZE_OF_VOLUME***],cloudVolumeUsageType=\"GENERAL\"|\"D
ATABASE\"}"

//DATAHUB
cdp datahub start-cluster-vertical-scaling --datahub=[***DATAHUB_NAME***] --
group=[***DATAHUB_INSTANCE_GROUP_NAME***] --disk-options addDisks="{numberO
fDisks=[***NUMBER_OF_DISKS_TO_ADD***], volumeType=\"[***TYPE_OF_COLUME_TO_AD
D***]\",size=[***SIZE_OF_VOLUME***],cloudVolumeUsageType=\"GENERAL\"|\"DATAB
ASE\"}"
```

Deleting disks

To save costs, you can delete volumes attached to instances in an instance group using CLI commands.

In cases where only compute services are being run on an instance group, any block storage attached to the instance is going to cost the customer even if the cluster is stopped. As compute services do not store any persistent data on attached volumes, having additional volumes to store temporary data is unnecessary. In such cases, customers can use this command to delete all attached volumes on instances in an instance group.

This can be done only for compute instance groups on only Data Hubs, as Data Lakes need persistent volumes. This command deletes all additional volumes for instances in an instance group.

Deleting disks using CDP CLI

Use the following CDP CLI Beta command to delete all attached additional volumes to the instances in a host group. Replace the placeholders with actual values. For example, *[***DATALAKE-NAME***]* should be replaced with an actual name. As part of this update, the instance-template parameter in the vertical scaling command has been made optional. But one of the -- instance-template or -- disk-options have to be provided.

The delete disks request is sent as part of the --disk-options parameter. All parameters for the add disk input are required.

The deleteDisks parameter accepts a boolean value, either true or false.

```
//DATAHUB

cdp datahub start-cluster-vertical-scaling --datahub=[***DATAHUB_NAME***] --
group=[***DATAHUB_INSTANCE_GROUP_NAME***] --disk-options deleteDisks=true|
false
```

Upgrading Cloudera Data Hub clusters

You can upgrade a Cloudera Data Hub cluster in one of three ways: Cloudera Runtime and Cloudera Manager major/minor version upgrades, service pack upgrades, and OS upgrades.

Cloudera Runtime and Cloudera Manager Major/Minor Version Upgrades

A Cloudera environment consists of both Data Lake and Cloudera Data Hub clusters, and currently these clusters should run the same major/minor versions of Cloudera Runtime and Cloudera Manager. This major/minor version is the first three digits of the Platform Version displayed along with the Cloudera Manager information:

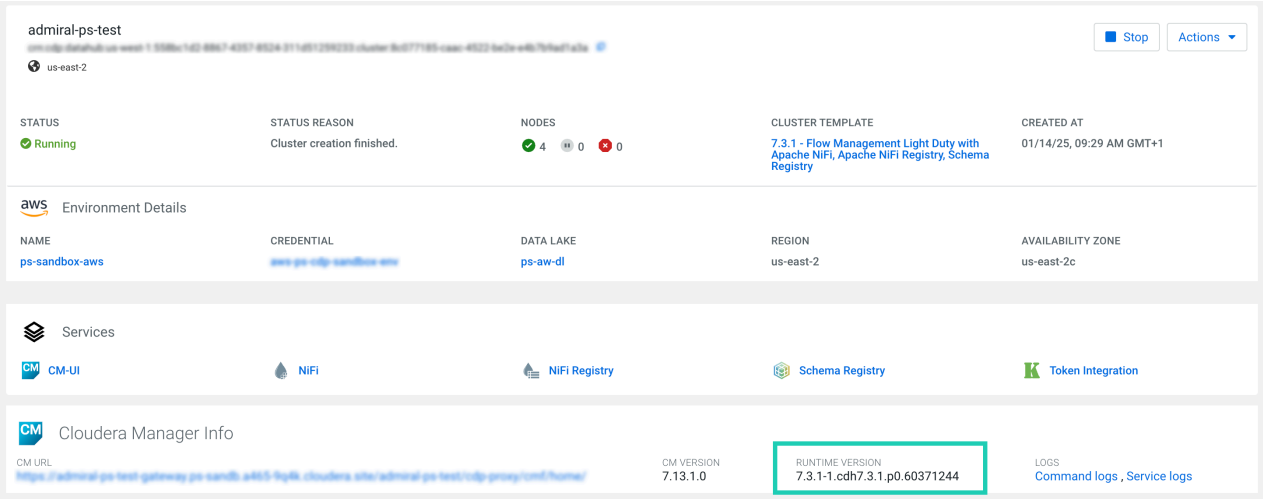

In the image above, the Cloudera Runtime major/minor version is 7.3.1.

A Cloudera Data Hub major/minor version upgrade initiates an upgrade of the major/minor Cloudera Manager and Cloudera Runtime versions, as well as the required additional parcels (Spark3, Flink, Profiler, and Cloudera Flow

Management). The root volumes, additional volumes, and the cluster state is retained. A major/minor version upgrade is an express upgrade where the services will not be available during the process.

A major/minor version upgrade is not an OS upgrade, as the OS/VM packages are not updated and VMs are not replaced. You can initiate an OS upgrade in a separate step. Major/minor upgrades do not update other Cloudera packages such as Salt.

A major/minor version upgrade does not redistribute the services according to the cluster template of the newer version. Nor does it add new services according to the new cluster template. Major/minor version upgrades do not add new service configurations in the new cluster template. These additional configurations have to be applied manually.

This is not a zero-downtime upgrade, and there will be service outages during the upgrade.

When a major/minor version upgrade is available, you will be able to select the target version for upgrade from the Upgrade tab at the bottom of the Data Hub details page:

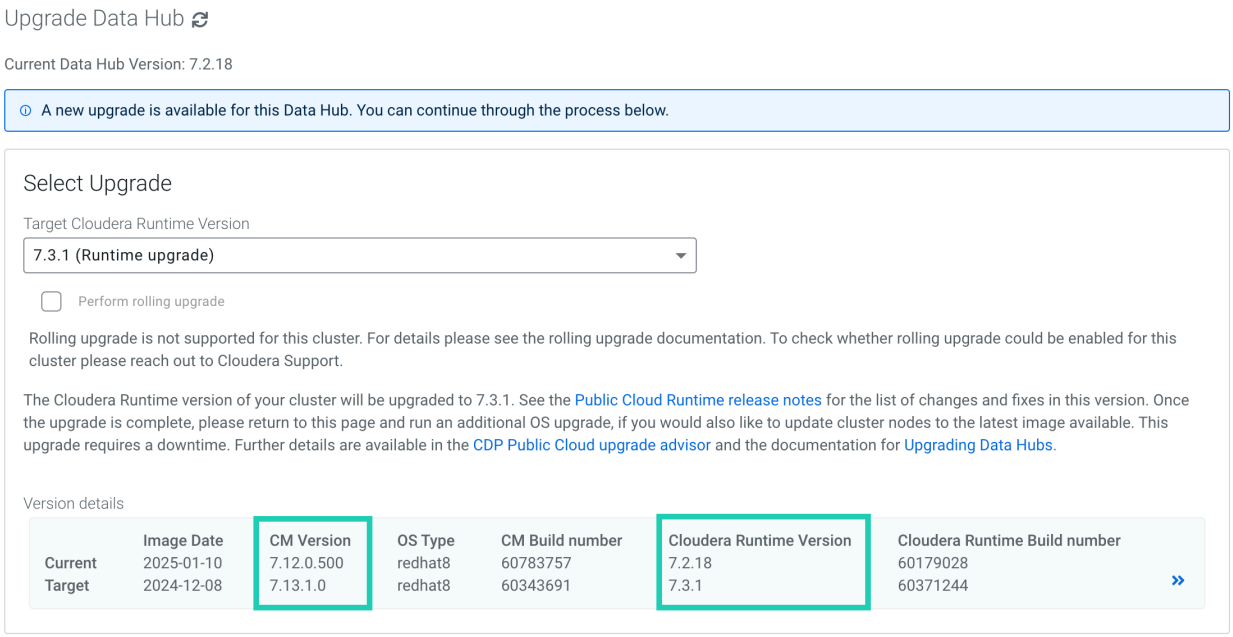

For supported versions and templates, see *Support matrix for major/minor upgrades*.

Note: Upgrading a custom template is supported, if the template contains only services that are included in any supported built-in templates.

If your cluster contains a custom service, such as a third-party Cloudera Manager service, contact Cloudera support for an entitlement that will allow you to upgrade the cluster. Cloudera cannot support failed upgrades for clusters that run unsupported services.

Always test any cluster upgrade in a development environment first.

Service pack upgrades

The service pack upgrade process checks to see if a new Cloudera Runtime or Cloudera Manager service pack is available, and then upgrades the Cloudera Data Hub cluster to the newest builds. Service pack upgrades do not upgrade to a new major/minor version of Cloudera Runtime and Cloudera Manager; they only upgrade to the latest build of a service pack version.

A service pack upgrade can be conducted on a single Cloudera Data Hub cluster in an environment, or multiple Cloudera Data Hub clusters. You can perform a service pack upgrade independent of a Data Lake upgrade.

When a service pack upgrade is available, you will be able to select the target version (which is the same as the current version) for upgrade from the Upgrade tab at the bottom of the Data Hub details page. When you select the target version, note that the Cloudera Manager/Cloudera Runtime versions are the same, but the build numbers differ:

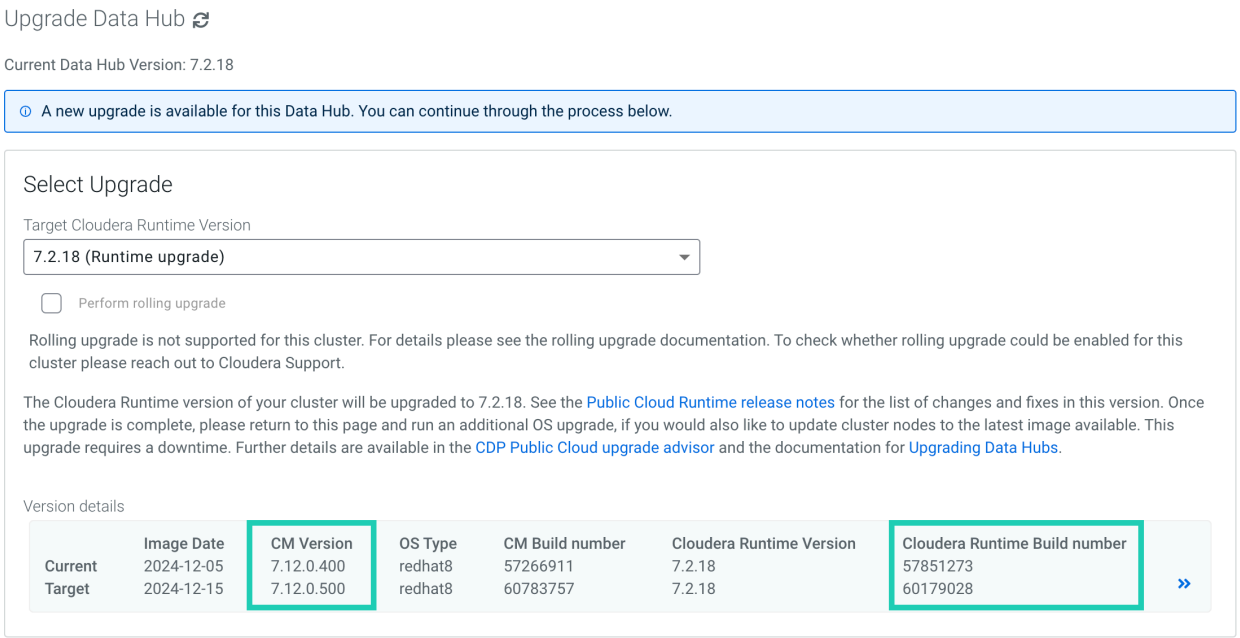

Service pack upgrades are available from Cloudera Runtime 7.2.7 onward for RAZ and non-RAZ Cloudera Data Hub clusters.

Important: The Cloudera Runtime 7.2.16.3 service pack has new dependencies that were not present in most of the previous Cloudera Runtime versions. Because of this, you may be unable to upgrade directly to this service pack version and future service packs without first upgrading to a more recent Cloudera Runtime version.

If you plan to upgrade your existing Cloudera Data Hub clusters from a previous release to 7.2.16.3 or later, you can verify whether or not you will first be required to perform an additional upgrade step:

1. Select the Cloudera Data Hub that you want to upgrade and click on the Upgrade tab.
2. If you see a warning message about missing prerequisites, follow the given steps to perform the additional upgrade before you upgrade to the latest 7.2.16 service pack.

If your current Cloudera Runtime version is 7.2.16 or any 7.2.16 service packs, these additional steps will include first performing an OS upgrade before upgrading to the 7.2.16.3 service pack.

If your current Cloudera Runtime version is 7.2.12, 7.2.14, or 7.2.15, these steps may include upgrading to the most recent service pack of your current Cloudera Runtime version, as well as performing a separate OS upgrade for your current Cloudera Runtime version, before you can perform a major/minor version upgrade to 7.2.16.3.

OS Upgrades

An OS upgrade updates the OS and VM packages to those available in the latest pre-warmed image. This is done by replacing the VM, re-attaching the attached volumes, and restarting the services. In the process, the data on root volume (for example, parcels and service logs) is lost. On larger clusters, test the OS upgrade in a development environment as the upgrade may exceed some of the cloud resource limits.

An OS upgrade does not upgrade the platform version (Cloudera Manager, Cloudera Runtime, and additional parcels). You can perform an OS upgrade independent of a Cloudera Data Hub cluster upgrade. The OS upgrade triggers the execution of any pre-service-deployment, post-cluster-manager-start, or post-service-deployment recipes.

If an OS upgrade is available, it will appear in the Upgrade Data Hub menu with “(OS Upgrade, OS: <target-OS>)” when you select a Cloudera Runtime version:

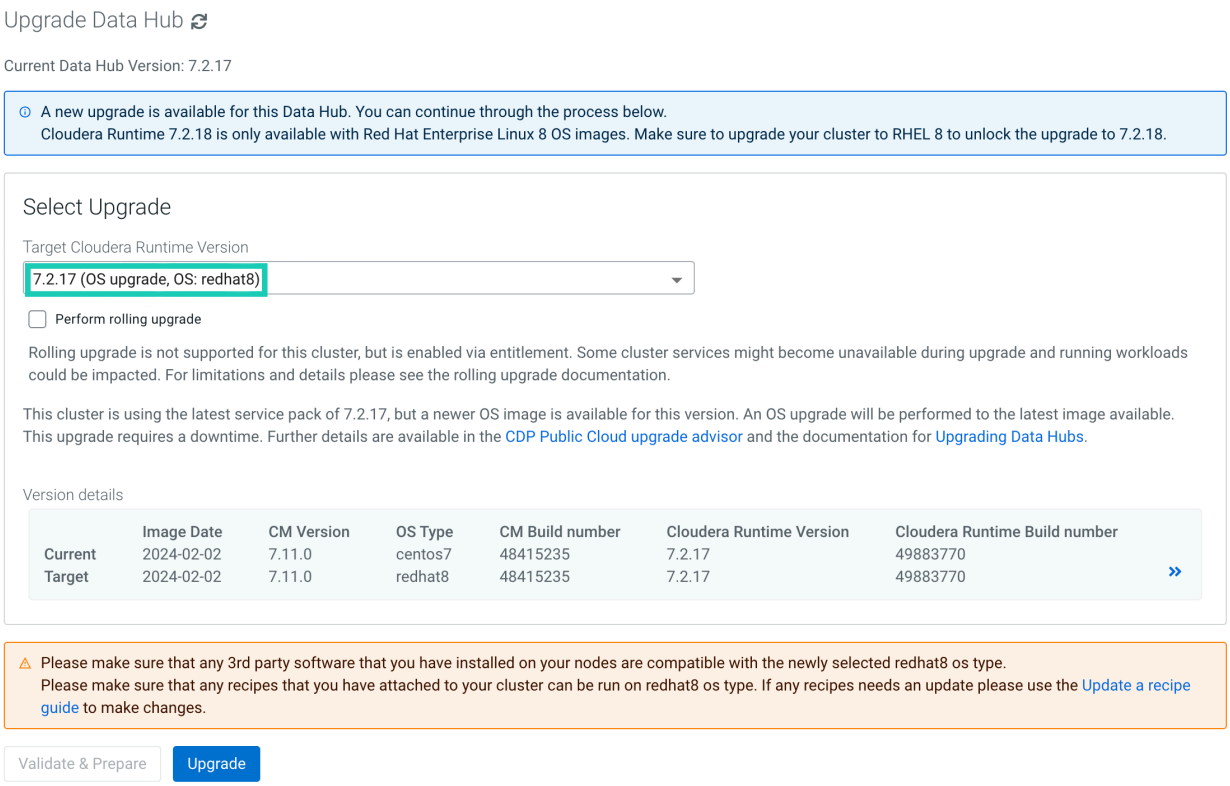

Rolling upgrades

Certain Cloudera Data Hub upgrades can be performed in a rolling fashion, depending on the Cloudera Data Hub template, Cloudera Data Hub OS, and the Cloudera Runtime version you are upgrading to and from. For more information, see Cloudera Data Hub rolling upgrades.

Limitations

- With the exception of rolling upgrades, Cloudera Data Hub upgrades are not a zero-downtime upgrade and service outages will occur.
- Ranger Authorization: Cloudera Data Hub service pack upgrades with RAZ are supported only for Cloudera Runtime versions 7.2.7+. Major/minor version upgrades with RAZ are supported only for Cloudera Runtime versions 7.2.10-7.2.12 to versions 7.2.14+.
- During an OS upgrade, any data on the root volume (parcels, service logs, custom software) will be lost. For older clusters (created before March 26, 2021), OS upgrade is not available when the embedded Cloudera Manager database is on the root volume.
- Cloudera Operational Database cannot be upgraded through the Cloudera Data Hub user interface and must be upgraded through the Cloudera beta CLI. For more information see *Upgrading Cloudera Operational Database*.
- When upgrading the Data Lake from Cloudera Runtime 7.2.17 to 7.2.18 or 7.3.1, if Iceberg metadata is required to be captured in Atlas, then do not use mixed versions (7.2.18 or 7.3.1 Data Lake with 7.2.17 Cloudera Data Hub), as the mismatch in the Iceberg models between 7.2.17 and 7.2.18 or 7.3.1 can cause problems.
- A Cloudera Data Hub cluster must be using Cloudera Runtime 7.2.17 in order to be eligible for CentOS to RHEL upgrade. If you are not seeing the option to upgrade to RHEL, ensure that your cluster is running Cloudera Runtime 7.2.17.

Prerequisites

- There is required downtime of the environment during upgrades, so plan the upgrade accordingly.
- Verify using the *Support matrix for Cloudera Data Hub upgrades* that all Cloudera Data Hub clusters in the environment are a type and version supported for major/minor upgrades.

- Test your applications against the new platform version in a separate environment before the Cloudera Runtime/Cloudera Manager upgrade to ensure application compatibility with the new platform version.
- The Data Lake and all the Cloudera Data Hub clusters in an environment must be upgraded to the same major/minor version, with the exception of Cloudera Data Hub clusters on Cloudera Runtime version 7.2.16+, which are compatible with newer versions of the Data Lake (7.2.17+, because the Data Lake must always run a higher Cloudera Runtime version). If you plan to upgrade your Cloudera Data Hub clusters to a later major/minor version, you must first backup and then upgrade the Data Lake.
- Verify that any Experiences you use are running the latest version available.
- Before being able to upgrade to Cloudera Runtime 7.3.1, you need to upgrade Data Lakes and all Cloudera Data Hub databases to a to PostgreSQL 14.
- If the upgrade involves upgrading from CentOS to RHEL, review the Prerequisites for upgrading from CentOS to RHEL.

Related Information

Data Lake upgrade
Support matrix for Cloudera Data Hub upgrades
Upgrading Cloudera Operational Database clusters

Support matrix for Cloudera Data Hub upgrades

Cloudera Data Hub upgrades are supported only for the templates and Cloudera Runtime versions listed below.

Template	Platform versions that you can upgrade from	Platform versions that you can upgrade to
DE, DE HA, DE Spark3	7.1.0 and above	7.2.11 and above
Operational Database	7.2.0 and above	7.2.11 and 7.2.12
Streams Messaging	7.2.0 and above	7.2.11 and above
Flow Management	7.2.0 and above	7.2.11 and above
Data Discovery	7.2.0 and above	7.2.11 and above
Datamart	7.2.2 and above	7.2.11 and above
Real-Time Datamart	7.2.2 and above	7.2.11 and above
Streaming Analytics (*See warning below)	7.2.10 until 7.2.14	7.2.11 until 7.2.15
Cloudera Data Catalog	Not supported. Use backup and restore	
Cloudera Operational Database	7.2.7 and above	7.2.11 and above

Warning:

The following upgrade paths are not supported for the Streaming Analytics cluster:

- from 7.2.11 to 7.2.12
- from any version to 7.2.16

For information about how to migrate your Flink jobs to a cluster with a new version, see the Migrating Flink jobs documentation.

Ranger Authorization (RAZ) Upgrades

Major/minor version upgrades with RAZ are supported only for Cloudera Runtime versions 7.2.10-7.2.12 to 7.2.14+.

Service Pack Upgrades

Cloudera Data Hub service pack upgrades, both RAZ and non-RAZ, are supported only for Cloudera Runtime versions 7.2.7+.

Note: Upgrading a custom template is supported, if the template contains only services that are included in any supported built-in templates.

If your cluster contains a custom service, such as a third-party Cloudera Manager service, contact Cloudera support for an entitlement that will allow you to upgrade the cluster. Cloudera cannot support failed upgrades for clusters that run unsupported services.

Always test any cluster upgrade in a development environment first.

Backing up before an upgrade

The Cloudera Data Hub cluster upgrade is an in-place upgrade and the data on attached volumes will be retained. The following section outlines data stored in the Cloudera Data Hub clusters by different services. To prepare for a worst-case scenario where the cluster can't be restored, retain the data in these clusters. Also take note of any cluster configurations updated through Cloudera Manager.

Service configurations in Cloudera Manager

- Any service configurations that you have updated since the cluster creation. Typically these can be found by selecting the “Non-default values” filter on the **Configuration** menu in Cloudera Manager.

Custom drivers

Take a backup of any custom drivers for services running on Cloudera Data Hub. An OS upgrade will delete the root volume and you will not be able to recover the driver if a backup is not taken.

Data Engineering clusters

The underlying data is in cloud storage. However, the following data is specific to each Cloudera Data Hub cluster:

- Oozie - workflow and coordinator definitions
- Hue - saved queries
- Hive UDFs and Spark applications
- Oozie custom sharelibs

A script is available for backing up and restoring the Oozie and Hue databases in clusters created with the Data Engineering and Data Engineering HA templates. Execute the commands from the Cloudera Manager node. Run the following:

```
sudo su
chmod +x /srv/salt/postgresql/disaster_recovery/scripts/backup_and_restore_d
h.sh
```

Then, for backup run:

```
/srv/salt/postgresql/disaster_recovery/scripts/backup_and_resto
re_dh.sh -b <s3 or abfs path> <database names>
```

For example:

```
/srv/salt/postgresql/disaster_recovery/scripts/backup_and_resto
re_dh.sh -b s3a://my/test/path oozie hue
```

For restoring after an upgrade, run:

```
/srv/salt/postgresql/disaster_recovery/scripts/backup_and_resto
re_dh.sh -r <s3 or abfs path> <database names>
```

For example:

```
/srv/salt/postgresql/disaster_recovery/scripts/backup_and_resto
re_dh.sh -r s3a://my/test/path oozie hue
```

Streams Messaging clusters

Backing up and restoring data

Data stored on volumes:

- Zookeeper: /hadoopfs/fs1/zookeeper (only on the first attached volume)
- Kafka: /hadoopfs/fsN/kafka (for on all attached volumes, N stands for volume index)

Data stored in database:

- Schema Registry schemas

Data stored in S3/ADLS:

- Schema Registry serde jars

A script is available for backing up and restoring the Schema_registry and SMM databases in clusters created with the Streams Messaging Light Duty and Streams Messaging Heavy Duty templates. Run the following:

```
sudo su
chmod +x /srv/salt/postgresql/disaster_recovery/scripts/backup_and_restore_d
h.sh
```

Then, for backup run:

```
/srv/salt/postgresql/disaster_recovery/scripts/backup_and_resto
re_dh.sh -b <s3 or abfs path> <database names>
```

For example:

```
/srv/salt/postgresql/disaster_recovery/scripts/backup_and_resto
re_dh.sh -b s3a://my/test/path smm schema_registry
```

For restoring after an upgrade, run:

```
/srv/salt/postgresql/disaster_recovery/scripts/backup_and_resto
re_dh.sh -r <s3 or abfs path> <database names>
```

For example:

```
/srv/salt/postgresql/disaster_recovery/scripts/backup_and_resto
re_dh.sh -r s3a://my/test/path smm schema_registry
```

Backing up and restoring Kafka Connect secret password and salt

On a Streams Messaging Cloudera Data Hub cluster, you need to manually reset the Kafka Connect secret password and salt in Cloudera Manager and save them for later so that you can manually reset them again when you need to recreate the cluster from scratch. Otherwise, Kafka Connect can go into a bad health state. The only way for you to know this password and salt is to reset them and manually record them somewhere.

Set new password and salt values in Cloudera Manager for the below values and Restart Kafka Connect to ensure the new password and salt is used for encryption:

- kafka.connect.secret.global.password
- kafka.connect.secret.pbe.salt

Save the password and salt for later use because you will need them if you need to recreate the cluster.

Resetting Kafka Connect secret password when recreating a cluster from backup

After recreating a Cloudera Data Hub cluster, you need to manually reset the Kafka Connect password and salt values to the values used on the original cluster.

1. Restart Kafka Connect after the Cloudera Data Hub cluster is recreated.
2. Set the following password and salt values to the ones that you have saved earlier for the original cluster:
   - kafka.connect.secret.global.password
   - kafka.connect.secret.pbe.salt
3. Restart Kafka Connect again.

Flow Management clusters

Verify that you have committed all of your flows, then backup the following files/directories into your S3 backup location:

- nifi.properties - on any NiFi node, run: ps -ef | grep nifi.properties

  This will indicate the path to the file. For example:

  -Dnifi.properties.file.path=/var/run/cloudera-scm-agent/process/1546335400-nifi-NIFI_NODE/nifi.properties
- bootstrap.conf - on any NiFi node, run: ps -ef | grep bootstrap.conf

  This will indicate the path to the file. For example:

  -Dorg.apache.nifi.bootstrap.config.file=/var/run/cloudera-scm-agent/process/1546335400-nifi-NIFI_NODE/bootstrap.conf
- Zookeeper data for /nifi znode. From a NiFi node:

  ```
  /opt/cloudera/parcels/CFM-<CFMversion>/TOOLKIT/bin/zk-migrator.sh -r -z
  <zk node>:2181/nifi
            -f /tmp/zk-data.json
  ```

  If a Zookeeper node is co-located with NiFi (light duty template, it's possible to use localhost:2181), then back up the created zk-data.json file.
- NiFi Registry data. The versioned flows are stored in an external database provisioned in your cloud provider. Perform a backup of this database following the instructions of your cloud provider and restore the backup in the new database.
- /hadoopfs/fs4/working-dir/ (for the NiFi nodes) - without the 'work' directory inside it. This directory contains, for example, the local state directory as well as the flow.xml.gz which represents the flow definitions.

  Note: On Cloudera Runtime versions lower than 7.2.10, this directory is located in /var/lib/nifi/working-dir.
- /hadoop/fs1/working-dir/ (for the Management node).

  Note: On Cloudera Runtime versions lower than 7.2.10, this directory is located in /var/lib/nifiregistry/working-dir.
- Any other "custom" directory, for example directories where client configs and JDBC driver are located.

Data Discovery & Exploration clusters

Complete the following steps to backup DDE clusters:

- Assign the datalake-admin-bucket policy to RANGER_AUDIT_ROLE.
- Create a subfolder for this Cloudera Data Hub cluster inside your S3 backup location.
- Log in to the gateway node of the Cloudera Data Hub cluster.
- kinit with the proper user keytab if needed.

- Follow this Backing Up and Restoring Solr Collections with these exceptions:
  - Use solrctl --get-solrxmlto get the solr.xml file.

    Edit it by changing your backup repository solr.hdfs.home parameter (see image) to your backup location (for example: s3a://datalake-bucket/backup/solr).

    User solrctl --put-solrxml to publish your edited solr.xml.

    Restart Solr.

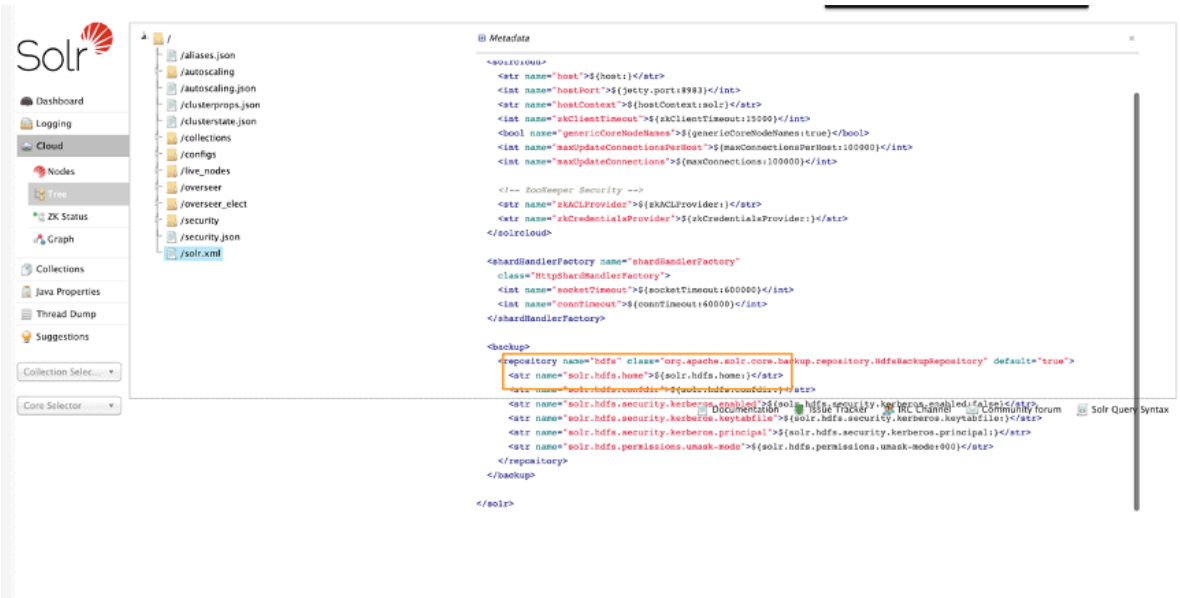

  - You do not need to launch the prepare-backup command; instead launch create-snapshot and export-snapshot for each collection.
  - For export-snapshot, specify your destination as s3a://[your backup location]/[your data hub subfolder] (for example: s3a://datalake-bucket/backup/solr).
  - Verify on the YARN history server that all of the executions succeeded. If not, troubleshoot and relaunch any export-snapshot commands that failed.

Operational Database clusters

Complete the following steps to backup Operational Database clusters:

- Create a subfolder for this Cloudera Data Hub cluster inside of your S3 backup location.
- Ensure snapshots are enabled on Cloudera Manager. They are enabled by default.
- Log in to the gateway node of the Cloudera Data Hub cluster.
- kinit with the proper user keytab if needed.
- Launch hbase shell.
- Issue the snapshot command for each of the tables that you want to back up (for example: snapshot ‘mytable’, ‘mytable-snapshot’)
- Exit hbase shell.
- Launch ExportSnapshot command for each of the snapshots you created, for example:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot mytable-
snapshot -copy-to
        s3a://datalake-bucket/backup/hbase -mappers 10
```

- Verify on the YARN history server that all of the executions succeeded. If not, troubleshoot and relaunch any ExportSnapshot commands that failed.

Performing a Cloudera Data Hub major/minor version cluster upgrade

A major/minor version upgrade initiates an upgrade of the major/minor Cloudera Manager and Cloudera Runtime versions, as well as the required additional parcels (Spark3, Flink, Profiler, and Cloudera Flow Management). After you perform a major/minor upgrade on a Data Lake, you should in most cases perform a major/minor version upgrade on each Cloudera Data Hub cluster attached to the Data Lake. The Cloudera Data Hub clusters must run the same major/minor Cloudera Runtime version as the Data Lake--with the exception of Cloudera Data Hub clusters on Cloudera Runtime 7.2.16+, which are compatible with Data Lake versions 7.2.17+.

About this task

Note that for major/minor version upgrades, the cluster can only be upgraded to the same major/minor version as the Data Lake, so you must first upgrade the Data Lake. For instructions, see *Data Lake upgrades*.

Complete the steps for each Cloudera Data Hub cluster that you are upgrading.

Required role: DatahubAdmin or Owner over the Cloudera Data Hub cluster

Before you begin

Procedure

1. Start the cluster.
2. Before you begin the Cloudera Data Hub cluster upgrade, check if the current version of Cloudera Runtime is <= 7.1.0.0. If yes, then verify the memory settings for Cloudera Manager in the /etc/default/cloudera-scm-server file on the Cloudera Manager server host. If you find the value “-Xmx2G” in CMF_JAVA_OPTS, update it to “-Xmx4G” and restart the Cloudera Manager server after the change.
3. For clusters that contain the Hive service in Cloudera Runtime versions prior to version 7.2.2, it is required to terminate all running YARN applications before starting the upgrade. So, if the current Cloudera Runtime version is < 7.2.2:
   a) SSH to any Hive node (master or worker).
   b) Get an initial ticket-granting ticket for YARN principal. This passes the Kerberos authentication so that you can run the YARN application in shell. YARN principals are installed in /run/cloudera-scm-agent/process/xxxxx-yarn-RESOURCEMANAGER/yarn.keytab

      Enter the directory and run klist command to display the Kerberos principals in YARN keytab (sample command):

      klist -kt yarn.keytab

      Sample Kerberos principal output: <format: userid/host@domain>

      ```
      yarn/nightly-7x-1-1.nightly-7x-1.root.hwx.site@ROOT.HWX.SITE
      ```

      From the above directory, run kinit command in this format (sample command): kinit -kt yarn.keytab <kerberos-principal format: userid/host@domain>

      ```
      kinit -kt yarn.keytab yarn/nightly-7x-1-1.nightly-7x-1.root.hwx.site@ROOT.HWX.SITE
      ```
   c) Run the command: yarn application -list
   d) For each running YARN application, run the command: yarn application -kill <appId>
   e) Run the command (to verify that no apps are running): yarn application -list

4. For Streams Messaging clusters, if you are upgrading from Cloudera Runtime version 7.2.12 to 7.2.14, complete the following steps:
   a) Open the Cloudera Manager UI for the Streams Messaging cluster.
   b) Click on the Cruise Control service, then click on Configurations.
   c) Search for 'RackAwareGoal' in the search bar and remove the entry for 'com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareGoal' for 'Default goals,' 'Hard Goals,' 'Support Goals,' ‘Self-Healing Goals, and 'Anomaly Goals.'
   d) Save the changes and restart the Cruise Control service.
   e) Proceed with the upgrade, but note that once the upgrade is complete, add the entries back to 'Default goals', 'Hard Goals', 'Support Goals,' ‘Self-Healing Goals,’ and 'Anomaly Goals.' This time rename the value to 'com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareDistributionGoal'
5. If your cluster uses Streams Replication Manager, export or migrate aggregated metrics.

   In Cloudera Runtime 7.2.18, major changes are made to the internal Kafka Streams application of SRM. As a result, SRM by default loses all aggregated metrics that were collected before the upgrade. This means that you will not be able to query metrics with the SRM Service REST API that describe the pre-upgrade state of replications. If you want to retain the metrics, you can either export them, for archival purposes, or migrate them to the new format used by SRM. If you do not need to retain metrics, you can skip this step and continue with the upgrade.

   Exporting metrics creates a backup of the metric data, however, exported metrics cannot be imported into the SRM Service for consumption. As a result, exporting metrics is only useful for data archival purposes.

   Migrating metrics can be done in two different ways depending on whether you are doing a rolling upgrade or a non-rolling upgrade.

   - In case of a non-rolling upgrade, migration happens following the upgrade. In this case, the new version of the internal Kafka Streams application running in the upgraded cluster starts to process historical metrics as soon as it is online. However, until the metrics are processed, the SRM Service cannot serve requests regarding latest metrics and returns empty or missing responses on its REST API. The duration of this downtime depends on the number SRM Service instances and the amount of metrics in the cluster.
   - In case of a rolling upgrade, a migration process called SRM Service Migrator is initiated during the upgrade. The Migrator processes existing metrics so that they become compatible with your upgraded cluster. Depending on the size of your cluster and the amount of metrics you have, this process may take up to multiple hours to finish.

   For Export metrics

   Use the following endpoints of the SRM Service REST API to export metrics.

   If upgrading from Cloudera Runtime 7.2.12 or higher:

   - `/v2/topic-metrics/{source}/{target}/{upstreamTopic}/{metric}`
   - `/v2/cluster-metrics/{source}/{target}/{metric}`

   If upgrading from Cloudera Runtime 7.2.11 or lower:

   - `/topic-metrics/{topic}/{metric}`
   - `/cluster-metrics/{cluster}/{metric}`

   For more information regarding the SRM Service REST API, see Streams Replication Manager Service REST API or Streams Replication Manager REST API Reference.

   For Non-rolling upgrade migration

   a. In Cloudera Manager, select the SRM service.
   b. Go to Configuration.
   c. Add the following to the SRM Service Environment Advanced Configuration Snippet (Safety Valve) property:

   ```
   Key: SRM_SERVICE_SKIP_MIGRATION
   ```

```
Value: false
```

For Rolling upgrade migration

Note: All SRM Service role hosts experience increased resource utilization during a rolling upgrade if you enable migration. This is because the migration process uses the same system configurations (for example, heap size) as the SRM Service role.

a. Ensure that the target clusters of the SRM Service are available and healthy.

If a target cluster is unavailable, the upgrade will fail. As a result, if a target cluster is unavailable, or you expect a target cluster to become unavailable during the upgrade, remove it from SRM's configuration for the duration of the upgrade. Metrics in the target clusters that you remove are not migrated. Target clusters are specified in Streams Replication Manager Service Target Cluster.

b. In Cloudera Manager, select the SRM service and go to Configuration.

c. Add the following to the SRM Service Environment Advanced Configuration Snippet (Safety Valve) property:

```
Key: SRM_SERVICE_SKIP_MIGRATION
Value: false
```

d. Fine-tune the behavior of the migration process.

The SRM Service Metrics Migrator (the migration process) has a number of user configurable properties. Fine tuning the configuration can help in reducing the time it takes to migrate the metrics.

These properties do not have dedicated entries in Cloudera Manager, instead you must use SRM Service Advanced Configuration Snippet (Safety Valve) for srm-service.yaml to configure them. If you are unsure about configuration, skip configuration and continue with the next step.

Table 1: SRM Service Migrator properties and recommendations

Property	Default Value	Description	Cloudera Recommendations
streams.replication.manager.migrator.monitor.timeout.ms	3,600,000	The time in milliseconds after the Streams Replication Manager (SRM) Service Metrics Migrator times out.	Set this timeout to a value that is higher than the expected migration time. Cloudera recommends a value that is at least three times the expected migration time. The migration time depends on the amount of metrics in your deployment. The higher the number of metrics, the longer the migration process, and the higher this property must be set. For example, a deployment with 10,000 partitions (100 topics with a 100 partitions each) and a 2 hour retention period produces, at minimum, 30,000 metrics per metric emission cycle. In a case like this, migration takes around 10 minutes to finish.

Property	Default Value	Description	Cloudera Recommendations
streams.replication.manager.migrator.monitor.backoff.ms	120,000	The frequency at which the progress of the Streams Replication Manager (SRM) Service Metrics Migrator is checked.	The recommended values for this property differ depending on the version you are upgrading from. If upgrading from 7.2.12 or higher: Set this property to a value that is identical with or similar to the interval set in SRM Service Streams Commit Interval The default value of this property is identical with the default value of SRM Service Streams Commit Interval. If upgrading from 7.2.11 or lower: Set this property to 30,000 (30 seconds).
streams.replication.manager.migrator.monitor.stop.delay.ms	60,000	The amount of time in milliseconds that the Streams Replication Manager (SRM) Service Metrics Migrator processes metrics after the streams application is considered caught up	
streams.replication.manager.migrator.monitor.min.consecutive.successful.checks	3	The number of consecutive checks where the lag must be within the configured threshold to consider the Streams Replication Manager (SRM) Service Metrics Migrator successful. All target clusters of the SRM Service must be caught up for a check to be successful.	

Property	Default Value	Description	Cloudera Recommendations
streams.replication.manager.migrator.monitor.max.offset.lag	Calculated automatically if left empty.	The amount of offsets that the streams application in the Streams Replication Manager (SRM) Service Metrics Migrator is allowed to lag behind and still be considered up to date. When left empty, the migration logic will automatically calculate the maximum offset lag based on the Kafka Streams application configuration and the amount of metrics messages. Low offset lag values result in more up-to-date metrics processing following the upgrade, but also increase the time required for the upgrade to finish.	An appropriate value for this property is calculated automatically if the property is left empty. As a result, Cloudera recommends that you leave this property empty and use the automatically calculated value.

e. Click Save Change.

f. Restart the SRM service.

6. If you use autoscaling, disable autoscaling on the cluster.

7. Upgrade the cluster. To upgrade the cluster with the UI:

a) In the left-hand menu, click Data Hub Clusters and then select the Cloudera Data Hub cluster to upgrade.

b) Scroll to the bottom of the Data Hub details page and select the Upgrade tab.

c) From the drop-down menu, select the Target Runtime Version.

d) If a rolling upgrade is available, select the Perform rolling upgrade checkbox if you would like to perform this type of upgrade. The availability of a rolling upgrade depends on the current and target Cloudera Runtime versions, the Cloudera Data Hub template, and the Cloudera Data Hub OS. See Cloudera Data Hub rolling upgrades for more information.

e) Click Validate and Prepare to check for any configuration issues and begin the Cloudera Runtime parcel download and distribution. Using the validate and prepare option does not require downtime and makes the maintenance window for an upgrade shorter. Validate and prepare also does not make any changes to your cluster and can be run independently of the upgrade itself. Although you can begin the upgrade without first running the validate and prepare option, using it will make the process smoother and the downtime shorter.

f) When the validate and prepare process is complete, click Upgrade.

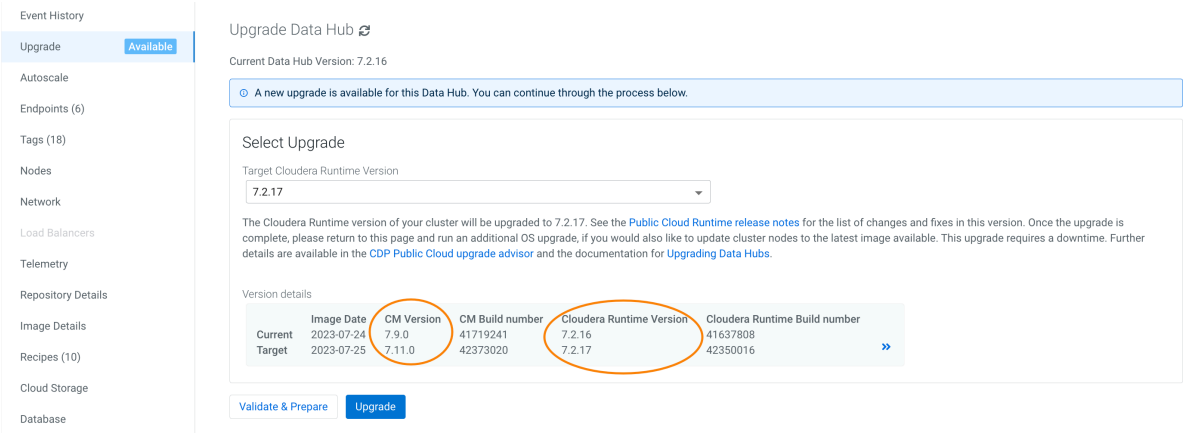

8. Monitor the upgrade progress using the Cloudera Data Hub **Event History** tab.

9. When the upgrade is complete, verify the new version. Note that the new version is reflected in the Platform Version, and not the version displayed in the cluster template name (which will remain the same):

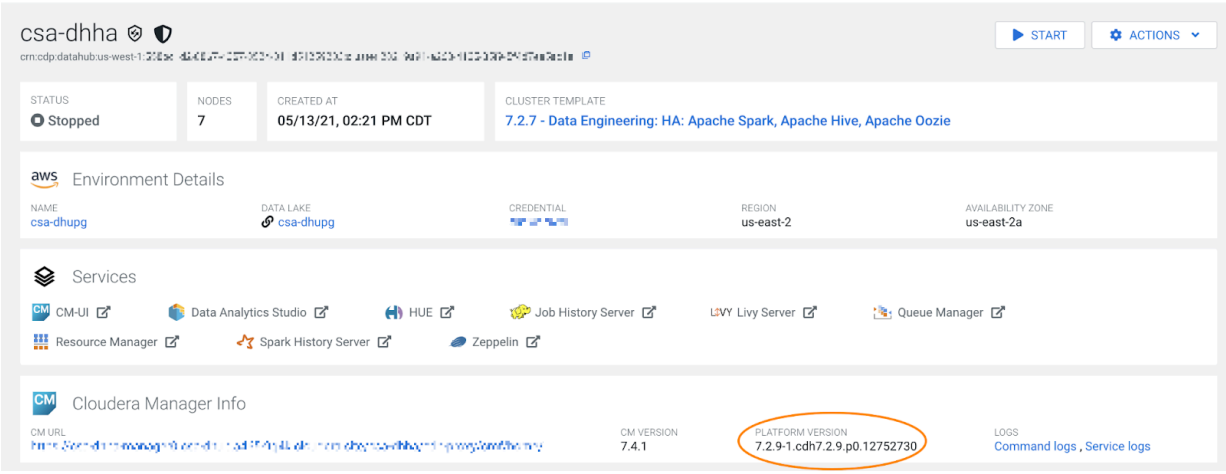

10. If you disabled autoscaling on the cluster, you can re-enable it after upgrade.

What to do next

If the upgrade is successful, proceed to the topic *Post-upgrade tasks*. If the upgrade fails, check the *Troubleshooting* section and re-try the upgrade.

Important: Once you have upgraded the Data Lake and started the Cloudera Data Hub upgrade process, you must repeat the upgrade procedure for every Cloudera Data Hub cluster in the environment. Do not conduct a major/minor version update asynchronously.

Related Information

Data Lake upgrade

Troubleshooting upgrade

Post-upgrade tasks

After a successful major/minor version upgrade, complete the following tasks if they apply to the cluster you upgraded.

Flow Management clusters

If doing a software-only upgrade for your Flow Management Cloudera Data Hub clusters and later repairing one of the NiFi nodes after the upgrade, you may be in a situation where the JDK used by NiFi is not the same across the nodes. This may cause issues in the NiFi UI and you may get an "Unexpected error" message.

After you upgrade a Flow Management cluster and repair a node, ensure that the same JDK is used across the NiFi nodes and if there is a JDK version mismatch, manually upgrade the JDK to match the JDK version being installed on the node that has been repaired.

Data Engineering upgrades to 7.2.11

Upgrading a cluster with the Spark service from Cloudera Runtime version 7.2.6 or 7.2.7 to version 7.2.11 may cause Spark cluster mode to fail. To workaround this issue, add the following configuration to the /etc/spark/conf/atlas-application.properties file: atlas.kafka.sasl.kerberos.service.name=kafka

Data Engineering HA/Hue HA

If you upgrade a template (such as Data Engineering HA) that contains Hue in an HA configuration, add the Hue load balancer hostnames to Cloudera Manager configurations: Cloudera Manager > Hue > Configuration > knox_proxyhosts. If you do not, you will experience errors preventing users from logging in.

Streams Messaging clusters

Clusters that contain the Streams Replication Manager service require a configuration change following a successful upgrade from Cloudera Runtime 7.2.11 or lower to 7.2.12 or higher. You must configure SRM to use its latest internal changelog data format and intra cluster hostname format. If this configuration is not completed, the SRM Service will not be able to target multiple clusters.

This post upgrade step is required because during an upgrade, SRM is configured to use a legacy version of its changelog data format and intra cluster hostname format. This is done to ensure backward compatibility so that if necessary, a rollback is possible.

Complete the following steps:

1. Verify that the SRM Service is up and running. This can be done by testing the REST API endpoints using the Swagger UI. If there are any issues, and a rollback is necessary, complete the rollback. No backward incompatible changes occurred up until this point.
2. In Cloudera Manager, select the Streams Replication Manager service.
3. Go to Configuration.
4. Find and disable the following properties:
   a. Use Legacy Internal Changelog Data Format
   b. Use Legacy Intra Cluster Host Name Format
5. Restart Streams Replication Manager.

Reset Kafka Connect secret password

After a Data Hub upgarde, you need to manually reset the Kafka Connect password and salt values to the values used before the upgrade was started. This step is required because these values are changed during the upgrade process but unless they are again set to the same values as before, Kafka Connect will go into bad health.

1. Restart Kafka Connect after the Data Hub has been recreated.
2. Set the following password and salt values to the ones that you have saved before starting the Data Hub upgrade process:
   - kafka.connect.secret.global.password
   - kafka.connect.secret.pbe.salt
3. Restart Kafka Connect again.

Configure Schema Registry to use V2 of its fingerprinting mechanism after upgrade to 7.2.18:

Following an upgrade to Cloudera Runtime 7.2.18 or later, Cloudera recommends that you configure Schema Registry to use fingerprinting V2. Fingerprinting V2 resolves an issue in V1 where schemas were not created under certain circumstances. For more information on the original issues as well as Schema Registry fingerprinting, see TSB-713. Note that even if you switch to V2, some issues might still persist, see TSB-718 for more information.

1. Access the Cloudera Manager instance managing your Cloudera Data Hub cluster.
2. Select the Schema Registry service and go to Configuration.
3. Set the Fingerprint Version property to VERSION_2.
4. Select Actions>Regenerate Fingerprints.
5. Click Regenerate Fingerprints to start the action.
6. Restart Schema Registry.

Adding configs manually

The major/minor Runtime upgrade does not install additional configs available in the newer versions of the Data Engineering cluster templates. This topic contains a list of configs added over different template versions. Use the Cloudera Manager UI to add them manually, if the config doesn't already exist.

- Clusters -> HDFS -> Configuration:

```
hdfs_verify_ec_with_topology_enabled=false
```

```
erasure_coding_default_policy=” “
```

- Clusters -> HDFS -> Configuration -> Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml:

```
fs.s3a.buffer.dir=${env.LOCAL_DIRS:-${hadoop.tmp.dir}}/s3a
HADOOP_OPTS="-Dorg.wildfly.openssl.path=/usr/lib64 ${HADOOP_OPTS}"
```

- Clusters -> Yarn -> Configuration:

```
yarn_admin_acl=yarn,hive,hdfs,mapred
```

- Clusters -> Yarn -> Configuration -> YARN Service MapReduce Advanced Configuration Snippet (Safety Valve):

```
mapreduce.fileoutputcommitter.algorithm.version=1
mapreduce.input.fileinputformat.list-status.num-threads=100
```

- Clusters -> Tez -> Configuration:

```
tez.grouping.split-waves=1.4
tez.grouping.min-size=268435456
tez.grouping.max-size=268435456
```

- Clusters -> Tez -> Configuration -> Tez Client Advanced Configuration Snippet (Safety Valve) for tez-conf/tez-site.xml:

```
tez.runtime.pipelined.sorter.lazy-allocate.memory=true
```

- Clusters -> Hive -> Configuration -> Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml:

```
fs.s3a.ssl.channel.mode=openssl
hive.txn.acid.dir.cache.duration=0
hive.server2.tez.session.lifetime=30m
hive.blobstore.supported.schemes=s3,s3a,s3n,abfs,gs
hive.orc.splits.include.fileid=false
hive.hook.proto.events.clean.freq=1h
hive.metastore.try.direct.sql.ddl=true
hive.privilege.synchronizer=false
```

- Clusters -> Hive -> Configuration:

```
hiveserver2_idle_session_timeout=14400000
```

- spark_on_yarn -> Configuration -> Spark Client Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-defaults.conf:

```
spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version=1
spark.hadoop.fs.s3a.ssl.channel.mode=openssl
```

- Clusters -> Hive Metastore -> Configuration -> Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml

```
hive.metastore.try.direct.sql.ddl=true
```

- Clusters -> <your cluster name> -> Configuration -> Suppressed Health and Configuration Issues:

```
role_config_suppression_namenode_java_heapsize_minimum_validator=true
```

Upgrading Cloudera Operational Database clusters

Cloudera Operational Database clusters cannot be upgraded through the Data Hub user interface. Instead, use the Cloudera Beta CLI to run the upgrade-database command.

For more information on CDP CLI beta commands, see the *CDP CLI Beta command reference*. For more information on rolling and non-rolling Cloudera Runtime upgrades for Cloudera Operational Database, see *Performing a Cloudera Runtime upgrade* in the Cloudera Operational Database documentation. For more information about rolling and non-rolling OS system upgrades for Cloudera Operational Database, see *Performing a Cloudera operating system upgrade* in the Cloudera Operational Database documentation.

Before you begin:

- Download and install the latest CDP Beta CLI.
- Required role: You must be logged into the Cloudera Operational Database as an ODAdmin.
- To use Cloudera Operational Database on a GCP environment, you must do it through CDP CLI with --use-hdfs flag.
- Understand Cloudera environment and user management. For more information, see the Cloudera Operational Database documentation topics *User management in Cloudera Operational Database* and *Cloudera Environments* topics.
- In the Cloudera Manager properties, increase the omid_max_heap_size property for the Omid service to at least 3GB before starting the upgrade from 7.2.9/7.2.10 to 7.2.11:

Important: Before upgrading the Cloudera Runtime version in your existing Cloudera Operational Database clusters to 7.2.16 or higher versions, you might need to perform some additional steps before upgrading. For more information, see *Data Lake upgrade validations for Python dependency* under the Cloudera Management Console what's new topic and *Upgrading Data Hubs*.

This Beta CDP CLI command upgrades an operational database in an environment to a given Runtime:

```
cdp opdb upgrade-database --environment <ENVIRONMENT-NAME> --
database <DATABASE-NAME> --runtime <RUNTIME-VERSION> [--os-upgrade-only | --
no-os-upgrade only]
```

Option	Description
--environment (string)	The name or CRN of the environment.
--database <value>	The name or CRN of the database.
--runtime <value>	The runtime version to upgrade to.
[--os-upgrade-only | --no-os-upgrade-only]	Controls whether to perform only an Operating System upgrade.

Related Information

CDP CLI Beta command reference
Performing a Cloudera operating system upgrade
Performing a Cloudera Runtime upgrade
User management in Cloudera Operational Database
Cloudera environments

Performing a Cloudera Data Hub service pack upgrade

A service pack upgrade can be conducted on a single Cloudera Data Hub cluster in an environment, or on multiple Data Hub clusters.

About this task

Complete the steps for each Cloudera Data Hub cluster that you are upgrading.

Required role: DatahubAdmin or Owner over the Cloudera Data Hub cluster

Procedure

1. In the left-hand menu, click Data Hub Clusters and then select the Cloudera Data Hub cluster to upgrade.
2. Scroll to the bottom of the Data Hub details page and select the Upgrade tab.
3. From the drop-down menu, select the Target Runtime Version.

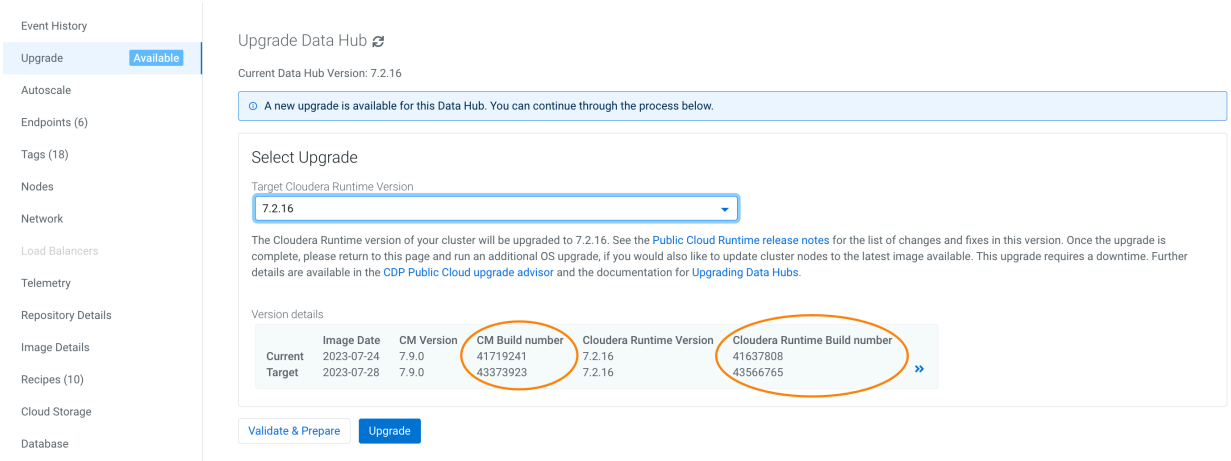

4. Click Validate and Prepare to check for any configuration issues and begin the Cloudera Runtime parcel download and distribution. Using the validate and prepare option does not require downtime and makes the maintenance window for an upgrade shorter. Validate and prepare also does not make any changes to your cluster and can be run independently of the upgrade itself. Although you can begin the upgrade without first running the validate and prepare option, using it will make the process smoother and the downtime shorter.
5. Click Upgrade.
6. Monitor the upgrade progress using the Cloudera Data Hub Event History tab.
7. If the upgrade fails, check the *Troubleshooting upgrade* section and re-try the upgrade.
8. If the upgrade is successful, you can re-enable autoscaling on the cluster if desired.
9. If your cluster contains the Oozie service, after the service pack upgrade is complete, re-install the Oozie shared libraries and YARN MapReduce Framework JARs.
   a) Navigate to the cluster's Oozie service in Cloudera Manager and click the Status tab.
   b) Click ActionsInstall Oozie ShareLib.
   c) Navigate to the cluster's YARN service in Cloudera Manager and click the Status tab.
   d) Click ActionsInstall YARN MapReduce Framework JARs.

Related Information

Troubleshooting upgrade

Performing a Cloudera Data Hub OS upgrade

Perform a Cloudera Data Hub OS upgrade to update the OS and VM packages to those available in the latest pre-warmed image.

About this task

Required role: DatahubAdmin or Owner over the Cloudera Data Hub cluster

Before you begin

It your cluster uses Streams Replication Manager (SRM), it is likely that you are storing security files, like truststores, in /opt/cloudera/security/ on your cluster hosts. These files are lost during the upgrade. As a result, you must back them up before the upgrade and restore them following the upgrade.

Before you begin

Before performing any OS upgrade, make sure there is no data belonging to NiFi or NiFi Registry on the root disk of the VM (this is the case for any version before Cloudera Runtime 7.2.10).

If you have NiFi or NiFi Registry data on the root disk, run the commands below to move the data to the right location prior to performing the upgrade. Before executing these scripts, stop the NiFi and NiFi Registry services.

On the NiFi nodes:

When upgrading from Cloudera Runtime 7.2.9 or lower versions

```
hadoopDirectory=/hadoopfs/$(ls /hadoopfs/ | sort | tail -n 1)
mkdir $hadoopDirectory/working-dir
cp -R /var/lib/nifi/* $hadoopDirectory/working-dir
echo "nifi.working.directory should be set to $hadoopDirectory/working-dir"l
atest_nifi_conf_directory=$(find /run/cloudera-scm-agent/process -name nifi\
.properties | sort | tail -n 1)
latest_nifi_conf_directory=${latest_nifi_conf_directory%nifi.properties}
echo "Latest nifi conf directory used to copy files before migration: $late
st_nifi_conf_directory"mkdir -p $hadoopDirectory/working-dir/config_backup
chmod 755 $hadoopDirectory/working-dir/config_backup
cp $latest_nifi_conf_directory/nifi.properties $hadoopDirectory/working-dir/
config_backup
cp $latest_nifi_conf_directory/bootstrap.conf $hadoopDirectory/working-dir/
config_backup
chown nifi:nifi -R $hadoopDirectory/working-dir
```

After the bash script executes, update the nifi.working.directory configuration value with what the script returns.

To set nifi.working.directory, perform the following steps:

1. Open the Cloudera Manager UI.
2. Go to the NiFi service.
3. Select the Configuration tab.
4. Search for nifi.working.directory.
5. Set the new value and click Save.

Note: When upgrading from Cloudera Runtime 7.2.10 or higher, no manual steps are required.

On the management node (where NiFi Registry is):

```
hadoopDirectory=/hadoopfs/$(ls /hadoopfs/ | sort | tail -n 1)
mkdir $hadoopDirectory/working-dir
cp -R /var/lib/nifiregistry/* $hadoopDirectory/working-dir
chown nifiregistry:nifiregistry -R $hadoopDirectory/working-dir
echo "nifi.registry.working.directory should be set to
$hadoopDirectory/working-dir"
```

After the bash script executes, update the nifi.registry.working.directory configuration value with what the script returns.

To set nifi.registry.working.directory, perform the following steps:

1. Open the Cloudera Manager UI.
2. Go to the NiFi service.
3. Select the Configuration tab.
4. Search for nifi.registry.working.directory.
5. Set the new value and click Save.

After the new configuration values are set on the NiFi and the management nodes, start the services.

Important: During an OS upgrade, any data on the root volume (parcels, service logs, custom software) will be lost.

Procedure

1. In the left-hand menu of Cloudera, click Data Hub Clusters and then select the Cloudera Data Hub cluster to upgrade.
2. Scroll to the bottom of the Data Hub details page and select the **Upgrade** tab.
3. From the drop-down menu, select the Target Runtime Version. An OS upgrade is indicated by the text “OS Upgrade (OS: <TARGET-OS>)” (where <TARGET-OS> is centos7 or redhat 8) next to the target Cloudera Runtime version.

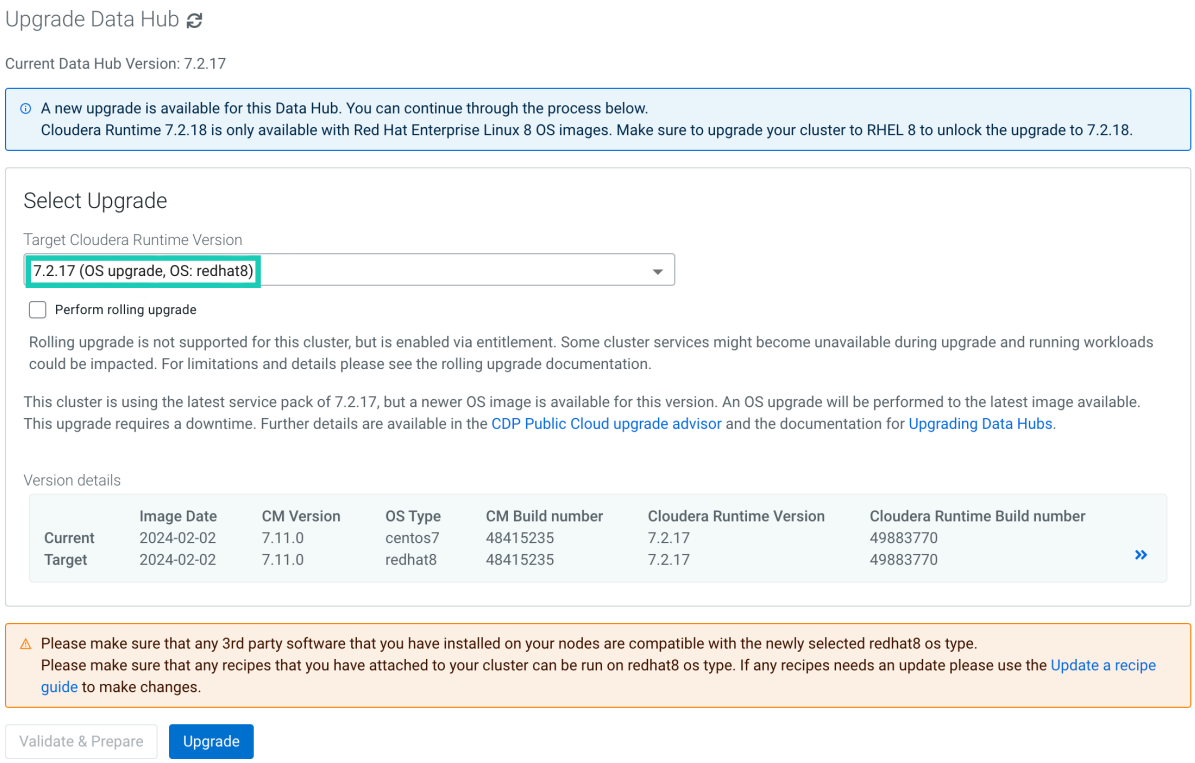

4. Click Upgrade.
5. Monitor the upgrade progress using the Cloudera Data Hub **Event History** tab.

What to do next

If you upgraded from Cloudera Runtime 7.2.9 or a lower version, perform the following tasks on the NiFi UI as post-upgrade steps when NiFi is available and running.

1. Open the NiFi UI.
2. Select Controller Settings from the Global Menu available in the top-right corner of the NiFi UI.

3. Go to the Management Controller Services tab of the **NiFi Settings** dialog.
4. Edit the Default Reporting Task SSL Context Service.
   a. Disable the running controller service by using the button.
   b. Modify the Keystore Filename value to point to the new nifi working directory. Use /hadoopfs/fs4/working-dir/cm-auto-host_keystore.jks instead of /var/lib/nifi/cm-auto-host_keystore.jks.
   c. Click APPLY.
5. Enable the Default Reporting Task SSL Context Service using the button.
6. Go to the Reporting Tasks tab.
7. Edit the Default Atlas Reporting Task by clicking the button.
   a. Modify the Atlas Configuration Directory to point to the new nifi working directory. Use /hadoopfs/fs4/working-dir/ instead of /var/lib/nifi/.
   b. Modify the Kerberos Keytab to point to the new nifi working directory. Use /hadoopfs/fs4/working-dir/nifi.keytab instead of /var/lib/nifi/nifi.keytab.
   c. Click APPLY.

What to do next

If your cluster uses SRM, restore the security files you backed up before the upgrade.

Upgrading a Cloudera Data Hub cluster with the CDP CLI

You can initiate a Cloudera Data Hub cluster upgrade (either OS, Runtime, or both) with the CDP CLI. Using the same CLI command, you can also search for and validate available images to upgrade to, and generate JSON templates for specific upgrade scenarios.

Obtain image ID

If your Data Lake upgrade includes upgrading from CentOS to RHEL 8, prior to attempting an upgrade you need to obtain an ID of a target RHEL 8 image. You can obtain it from the image catalog by finding an image with your target Runtime version which has an OS Type of RHEL8.

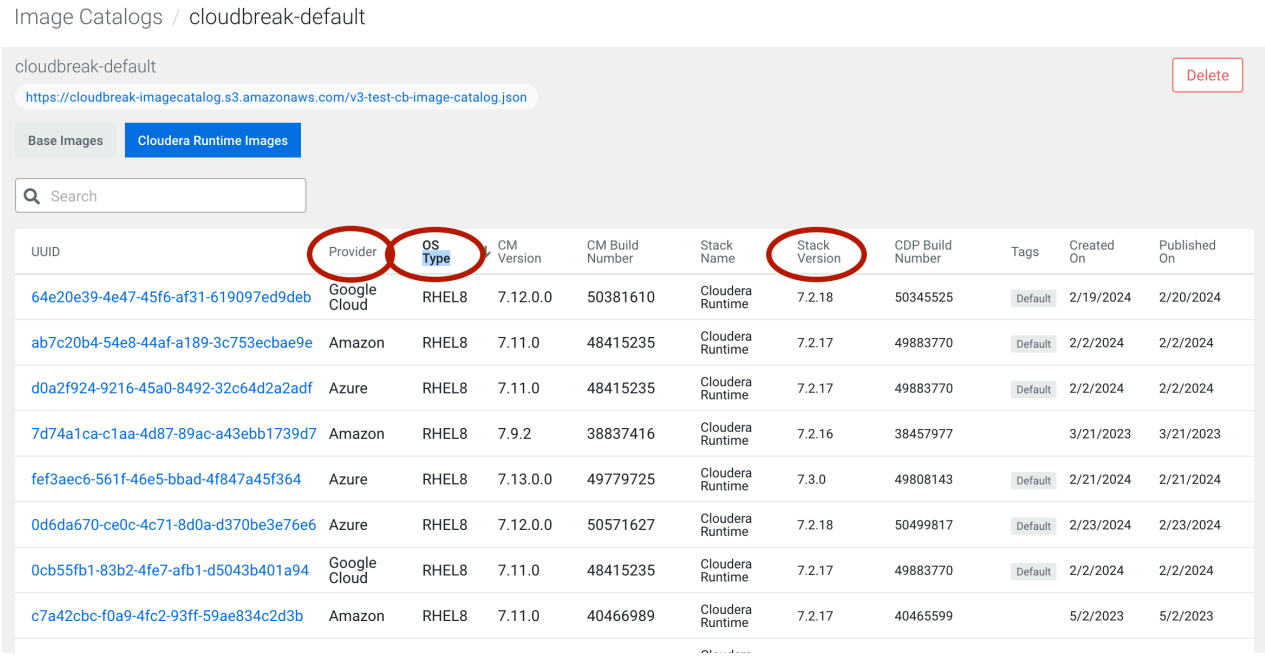

UUID	Provider	OS Type	CM Version	CM Build Number	Stack Name	Stack Version	CDP Build Number	Tags	Created On	Published On
64e20e39-4e47-45f6-af31-619097ed9deb	Google Cloud	RHEL8	7.12.0.0	50381610	Cloudera Runtime	7.2.18	50345525	Default	2/19/2024	2/20/2024
ab7c20b4-54e8-44af-a189-3c753ecbae9e	Amazon	RHEL8	7.11.0	48415235	Cloudera Runtime	7.2.17	49883770	Default	2/2/2024	2/2/2024
d0a2f924-9216-45a0-8492-32c64d2a2adf	Azure	RHEL8	7.11.0	48415235	Cloudera Runtime	7.2.17	49883770	Default	2/2/2024	2/2/2024
7d74a1ca-c1aa-4d87-89ac-a43ebb1739d7	Amazon	RHEL8	7.9.2	38837416	Cloudera Runtime	7.2.16	38457977		3/21/2023	3/21/2023
fef3aec6-561f-46e5-bbad-4f847a45f364	Azure	RHEL8	7.13.0.0	49779725	Cloudera Runtime	7.3.0	49808143	Default	2/21/2024	2/21/2024
0d6da670-ce0c-4c71-8d0a-d370be3e76e6	Azure	RHEL8	7.12.0.0	50571627	Cloudera Runtime	7.2.18	50499817	Default	2/23/2024	2/23/2024
0cb55fb1-83b2-4fe7-afb1-d5043b401a94	Google Cloud	RHEL8	7.11.0	48415235	Cloudera Runtime	7.2.17	49883770	Default	2/2/2024	2/2/2024
c7a42cbc-f0a9-4fc2-93ff-59ae834c2d3b	Amazon	RHEL8	7.11.0	40466989	Cloudera Runtime	7.2.17	40465599		5/2/2023	5/2/2023

Once you have identified the ID, you can provide it in the upgrade CLI command by using the --image-id flag.

Upgrade steps

The cdp datahub upgrade-cluster command includes the following options:

```
cdp datahub upgrade-cluster

          --cluster-name <value>
          [--image-id <value>]
          [--runtime <value>]
          [--lock-components | --no-lock-components]
          [--dry-run | --no-dry-run]
          [--show-available-images | --no-show-available-images]
          [--show-available-image-per-runtime | --no-show-available-image-
per-runtime]
          [--cli-input-json <value>]
          [--generate-cli-skeleton]
```

In order to use this command for upgrading from CentOS to RHEL, ensure to provide an image ID of a RHEL 8 image.

Important: The --runtime option does not upgrade the OS. Upgrading the OS is a separate process that requires specifying the --lock-components option.

Table 2:

Option	Description
--cluster-name (string)	Required. The name or CRN of the Cloudera Data Hub cluster to upgrade.
--image-id (string)	The ID of an image to upgrade to. If upgrading from CentOS to RHEL, ensure to provide a RHEL 8 image ID.
–runtime (string)	The Runtime version to upgrade to. When you specify the Cloudera Runtime version, the upgrade uses the latest image ID of the given Cloudera Runtime version from the same image catalog used for Cloudera Data Hub cluster creation.
--lock-components | --no-lock-components (boolean)	Use --lock components to perform an OS upgrade only.
--dry-run | --no-dry-run (boolean)	Checks the eligibility of an image to upgrade. Can be used in conjunction with any other parameter, returning the available image (with respect to image Id, Runtime or lock-components set) without performing any actions.
--show-available-images | --no-show-available-images (boolean)	Returns the list of images that are eligible to upgrade to.
--show-latest-available-image-per-runtime | --no-show-latest-available-image-per-runtime (boolean)	Returns the latest image that is eligible to upgrade to, for each Runtime version with at least one available upgrade candidate.

When you run the cdp datahub upgrade-cluster command to initiate an upgrade, you have one of three options:

1. Specify one of either --image-id, --runtime, or --lockComponents, which makes an explicit choice of the exact image, Cloudera Runtime (latest OS), or latest OS (same Runtime) for upgrade.
2. Specify both --image-id and --lockComponents,which specifies an image and ensures the image represents an OS only upgrade.
3. Specify none of the --image-id, --runtime, or --lockComponents parameters, which initiates a Cloudera Runtime/Cloudera Manager upgrade to the latest compatible version and OS image.

Outside of upgrade, you can use the following options:

```
--show-available-images/--no-show-available-images
--show-available-images-per-runtime/--no-show-available-images-per-runtime
```

```
--dry-run
```

Cloudera Data Hub rolling upgrades

The Cloudera Data Hub rolling upgrade allows you to upgrade the cluster without stopping the Cloudera Data Hub cluster itself and incurring workload downtime.

Cloudera Data Hub rolling upgrades are only available for the following cluster types:

- Cloudera Operational Database
- Cloudera Streams Messaging

Cloudera Operational Database clusters

For a Cloudera Operational Database cluster to be eligible for rolling upgrade, the following requirements must be met:

- Currently, rolling OS upgrade is only supported on Cloudera Operational Database clusters with storage selected as HDFS or cloud without ephemeral storage.
- Rolling OS upgrade is not supported on Micro Cloudera Operational Database clusters.

Table 3:

Current Cloudera Runtime version & template	Rolling upgrade support for Cloudera Runtime?	Rolling OS upgrade support?
7.2.18	Yes, to 7.2.18+ (including service pack upgrades to 7.2.18.100 and higher versions)	Yes
7.2.17.x	Yes, to 7.2.18+	Yes
7.2.17.200 - 7.2.17.500	Yes, to 7.3.1	Yes
7.2.18.0 - 7.2.18.200	Yes, to 7.3.1	Yes
7.3.1.0 **Note:** Rolling upgrade for Cloudera Operational Database clusters from 7.3.1.0 to 7.3.1 higher versions is in Technical Preview and only available for clusters with Ephemeral cache.	Yes, to 7.3.1+	Yes

For instructions on performing Cloudera Operational Database rolling upgrades, see Performing a Cloudera Runtime upgrade (Cloudera Operational Database) and Performing a Cloudera operating system upgrade (Cloudera Operational Database).

Cloudera Streams Messaging clusters

For a Streams Messaging cluster to be eligible for rolling upgrade, the following requirements must be met:

- The cluster OS must be RHEL 8. Streams Messaging clusters on CentOS are not supported.
- The cluster must be Runtime version 7.2.18 or higher version.
- The cluster template must be Streams Messaging High Availability.

Current Cloudera Runtime version & template	Rolling upgrade support for Cloudera Runtime?	Rolling OS upgrade support?
7.2.18 Streams Messaging High Availability template (Clusters **created** on 7.2.18 only)	Yes, to 7.2.18+ (including service pack upgrades to 7.2.18.100 and higher versions) and 7.3.1	Yes, for up to 20 nodes. **Important:** Rolling OS upgrades replace nodes one-by-one and can take a significant amount of time, several hours or more, depending on the number of nodes.
7.2.18.0 - 7.2.18.200	Yes, to 7.3.1	Yes

In some circumstances, a rolling upgrade may not be supported for a Streams Messaging cluster, but can be enabled through entitlement. The Cloudera Data Hub upgrade UI displays information about whether a rolling upgrade is available, unavailable, or may be available under entitlement. For instructions on performing a Cloudera Data Hub cluster upgrade, including rolling upgrades, see Upgrading Cloudera Data Hub clusters. For information about obtaining an entitlement for rolling upgrade, contact Cloudera Customer Support.

Cloudera Data Hub rolling upgrade limitations and issues

Cloudera Data Hub rolling upgrades have the following limitations:

Cloudera Operational Database clusters

- HBase commands may fail during the rolling upgrade with the error "ServerNotRunningYetException: Server is not running yet." HBase retries DDL operations submitted while the master is initializing until the master is fully initialized to serve the request. However, a situation might arise where the default number of retries or intervals proves to be insufficient for an operation submitted by the client to complete.

  Implementing the following configuration adjustments in your client application can support the master getting initialized up to 10 minutes:

  ```
  <property>
      <name>hbase.client.pause</name>
      <value>300</value>
   </property>
   <property>
      <name>hbase.client.retries.number</name>
      <value>20</value>
   </property>
  ```

  If you have seen a longer or shorter master initialization period, you can modify these values accordingly. These retry settings apply to all types of calls to HBase service, encompassing GET, SCAN, MUTATE, and DDLs.
- In a rolling restart, if a COD cluster has less than 10 datanodes, existing writes can fail with an error indicating a new block cannot be allocated and all nodes are excluded. This is because the client has attempted to use all the datanodes in the cluster, and failed to write to each of them as they were restarted. This will only happen on small clusters of less than 10 datanodes, as larger clusters have more spare nodes to allow the write to continue.
- When performing a maintenance upgrade or other cluster upgrade, in some occasions there can been an error when the upgrade is nearly complete, but trying to re-start services/roles. The error is similar to: "Failed to start role hue-HUE_SERVER-8cc9321b2213cc5c6846c64e1fc6b1cb of service hue in cluster cod--xoaitnb0wnl1. This role requires the following additional parcels to be activated before it can start: [cdh]."

  This is due to an agent operation that sometimes is delayed and can interfere with the role start. When this happens, resume the failed upgrade from Cloudera Manager as a 'Full Admin' user.
- During the VM replacement as part of OS upgrade, every new node gets a new IP Address, and if the old IP address is cached somewhere, HDFS requests fail with UnknownHostException. It recovers after some time (10 mins max).
- If Knox is HA and one of the Knox servers is down, then accessing the service through a Cloudera Control Plane endpoint URL (i.e., through cloud load balancer) will take approximately 30 seconds to failover the request to the

available Knox instance. This also means that the services that are reached through Knox will not be available behind Knox during this period time.
- During OS upgrades, attempts to access Knox on the host being upgraded may produce occasional 403 HTTP responses. Wait and retry the failed requests.
- When upgrading Cloudera Data Hub clusters to Cloudera Runtime 7.2.18.100, you might encounter staleness in knox.jwt.client.gateway.address configuration in case its value points to the address of the Data Lake node. If staleness occurs after the upgrade, you need to run Deploy Client Configuration in Cloudera Manager.

For more limitations of Cloudera Operational Database, see Rolling upgrade limitations (Cloudera Operational Database).

Cloudera Streams Messaging clusters

- Rolling upgrades are not supported for Cruise Control or Streams Messaging Manager (SMM). When upgrading a Streams Messaging cluster, expect that both of these services will be temporarily unavailable during the upgrade. This, however, does not impact Kafka's ability to perform a rolling upgrade.
- Rolling upgrades for Schema Registry are only supported from Cloudera Runtime 7.2.18 or higher to higher versions. When upgrading a Streams Messaging cluster from a lower version, expect that clients connecting to the Schema Registry service might experience downtime. This, however, does not impact Kafka's ability to perform a rolling upgrade.
- If Knox is HA and one of the Knox servers is down, then accessing the service through a Cloudera Control Plane endpoint URL (i.e., through cloud load balancer) will take approximately 30 seconds to failover the request to the available Knox instance. This also means that the services that are reached through Knox will not be available behind Knox during this period time.
- During OS upgrades, attempts to access Knox on the host being upgraded may produce occasional 403 HTTP responses. Wait and retry the failed requests.
- When upgrading Cloudera Data Hub clusters to Cloudera Runtime 7.2.18.100, you might encounter staleness in knox.jwt.client.gateway.address configuration in case its value points to the address of the Data Lake node. If staleness occurs after the upgrade, you need to run Deploy Client Configuration in Cloudera Manager.

Troubleshooting upgrade

Use the information in this section to troubleshoot problems with Cloudera Data Hub cluster upgrades.

Root disk space

Service pack upgrades require a certain amount of root disk space. Cloudera Runtime upgrades require downloading additional parcels, and hence more storage. When you begin an upgrade, Cloudera checks the available root disk space and issues a warning if there is insufficient root disk space for the upgrade:

If your instances do not have the free space required for an upgrade (Cloudera Manager: 27 GB, other instances: 20 GB), run the scripts below to increase the root volume of the cluster nodes.

- Script to increase the root disk volume for AWS
- Script to increase the root disk volume for Azure

Oozie Shared Library mismatch

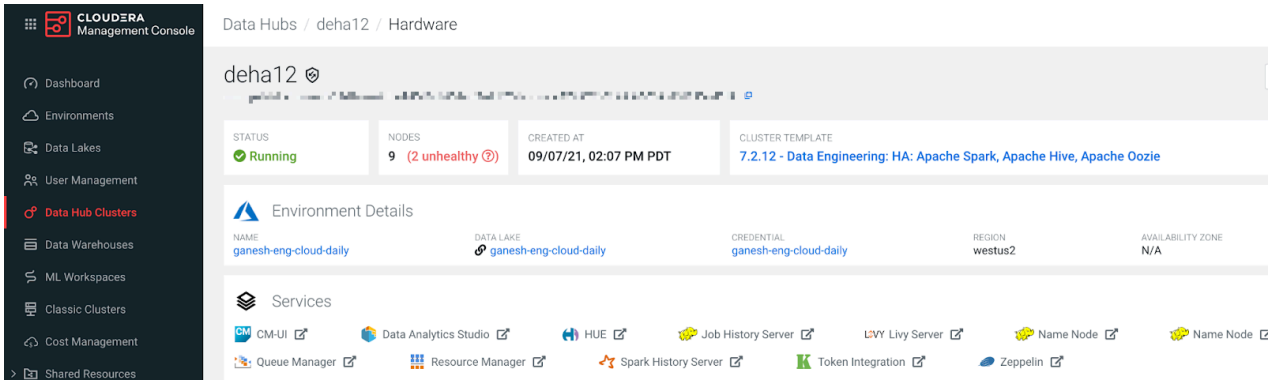

If the upgraded cluster contains the Oozie service, it may appear as being in bad health after the upgrade due to an known issue with the Oozie server shared library. On the Oozie Shared Library Check page in Cloudera Manager, you will see an error similar to: “The Oozie Server build version and the Oozie Server shared library version do not match.”

To workaround this issue, follow the steps at the end of the Performing a service pack upgrade section to re-install the Oozie shared libraries and YARN MapReduce Framework JARs.

Upgrade fails due to active Cloudera Manager commands

Upgrade may fail if there are active Cloudera Manager commands running when an upgrade is triggered. If you receive the error message “There are active commands running on Cloudera Manager, upgrade is not possible. Active commands: ApiCommand[..., name: <cm command name>,]”, then kill the active commands and retry the upgrade.

Cloudera Manager, Cloudera Runtime, or other components are out-of-sync with Cloudera

When an upgrade fails, the versions of Cloudera Manager, Cloudera Runtime, and other components may become out-of-sync with the Cloudera Management Console. Similarly, if you try to fix errors by installing parcels manually, it may not be reflected in the Cloudera Management Console.

To overcome the mismatch between versions reflected in the Cloudera Management Console, run the cdp datahub sync-component-versions-from-cm CDP CLI command. This command reads the Cloudera Manager, Cloudera Runtime, and other parcel versions (if applicable) from Cloudera Manager and updates the versions in the Cloudera Management Console. Using this command forces the Cloudera Management Console back in sync so that it shows the actual versions installed in Cloudera Manager.

Run the command as follows: cdp datahub sync-component-versions-from-cm --datahub-name <datahub name or CRN>

Related Information

Performing a Cloudera Data Hub service pack upgrade

Autoscaling clusters

Autoscaling is a feature that adjusts the capacity of cluster nodes running YARN by automatically increasing or decreasing, or suspending and resuming, the nodes in a host group. You can enable autoscaling based either on a schedule that you define, or the real-time demands of your workloads.

Load-based autoscaling

Attention:

Load-based autoscaling is available for clusters with the following Cloudera Runtime versions:

- 7.2.15 and higher

Load-based auto-scaling is supported in AWS and Azure. In GCP, load-based autoscaling is available as technical preview and is under entitlement. To obtain the required entitlement, contact your Cloudera Account Representative.

Load-based autoscaling suspends and resumes (stops and starts) instances on the cloud provider to increase or decrease capacity for nodes running NodeManagers (for example, the compute host group), based upon YARN's assessment of pending demand and available capacity. Load-based autoscaling can help control costs while providing quick, on-demand cluster capacity when you need it (within a few minutes).

When you configure a load-based autoscaling policy, you choose a minimum and maximum number of nodes for the host group. The maximum number of nodes determines how many instances are provisioned, but these instances are suspended and resumed as the workload demand requires. The policy will not provision instances beyond the maximum range of nodes that you define, regardless of the demand on the cluster. You also define a cooldown period, which is the amount of time in minutes to wait before another autoscaling operation is performed.

Schedule-based autoscaling

Attention:

Schedule-based autoscaling is available for clusters with the following Cloudera Runtime versions:

- 7.2.15 and higher

Schedule-based autoscaling scales the nodes in a host group up or down based upon a schedule that you define. Schedule-based autoscaling is useful if workload demands tend to be high or low on a fairly regular, consistent basis.

When you configure a schedule-based autoscaling policy, you define a target node count, which is the number of nodes that you want to scale up or down to at a particular time. You also select whether or not to repeat the schedule, and finalize the schedule by entering a CRON expression and selecting the desired timezone. When the particular time/date that you define in the CRON expression occurs, the cluster is upscaled or downscaled to your target node count. The time taken to add nodes to a cluster varies by cloud provider and the configuration of the nodes (for example, recipes can have an impact on the time it takes to add a node).

Note: Typically, schedules (especially those configured to scale-up) should be defined in a manner to factor in the time it takes to add new nodes. For example, workloads starting at 9pm will typically set a scale-up schedule for 8:45 pm, assuming it takes around 15 minutes to add a node.

General considerations

Currently, YARN NodeManager is the only service compatible with autoscaling. Because of this, autoscaling is only available by default in the Data Engineering and Data Engineering HA cluster definitions. Autoscaling can also be configured for clusters with custom templates that include YARN NodeManager and the necessary gateway components.

Known Limitations

- Load-based autoscaling does not work with YARN dynamic queues.

Autoscaling behavior

Before you define an autoscaling policy, note the following autoscaling behaviors.

General behaviors

- Clusters can perform one upscale or downscale operation at a time.
- A cluster will continue to accept jobs while it is running, regardless of any in-progress upscale or downscale operations.
- Only one autoscale policy type (either load-based or schedule-based) can be configured for a single host group, in a single cluster, at a time.
- Autoscaling is available for host groups with nodes running YARN NodeManager only (and optionally client/GATEWAY components, but not any other service components).
- If there are not enough nodes available to match the requested scale operation, the operation will proceed on however many nodes are available (for example, during a request for a 10 node scale-up, if the cluster loses 1 node, the operation will proceed with scaling-up 9 nodes instead of 10).
- Autoscaling will be disabled if the cluster has any node failures on instances running YARN ResourceManager, or the ClouderaManager node.
- After scaling down, nodes will show up as UNHEALTHY in Cloudera Manager. This is expected. While scaling down, stopped nodes are put into maintenance mode, to suppress alerts.

Attention: Autoscaling is not available for Cloudera Data Hub clusters created with the default Streaming Analytics Light Duty cluster template.

Note: Make sure to create clusters with at least one node in each host group running YARN NodeManagers, for automatic resource configuration to take effect (unless explicitly configuring YARN node-level resources in the cluster template). Once a cluster has been created, the node count for the same can go to 0 (or the minimum can be set to 0 via autoscaling configuration).

Warning: Before performing the FreeIPA upgrade for Cloudera Data Hub clusters where autoscaling is enabled, you must disable autoscaling and start all compute nodes to ensure the healthy state of Cloudera Data Hub clusters. After upgrading FreeIPA, autoscaling can be enabled again. For more information, see the Autoscaling must be stopped before performing FreeIPA upgrade description in the Known issue.

Cluster management operations with load-based autoscaling enabled

Specific operations are affected as follows:

- Node repair: If the node on which Cloudera Manager is running is selected for repair, then all the stopped nodes need to be selected for repair.
- Cluster stop/start: Cluster stop is supported. During a cluster start, all nodes in the host group will be started and recommissioned.
- Node delete: Node delete (non-forced) is supported, including for STOPPED instances. Forced node delete may require you to End Maintenance (Enable Alerts/Recommission) in Cloudera Manager after performing another upscale operation.
- Retry: Do not attempt retry on an autoscale operation. Instead, see *Manually recovering from load-based scaling failures*. The nature of load-based autoscaling is to automatically try again after some time, based on the load on the cluster. There are a few scenarios where this retry will be adequate to continue cluster operations.

Only a single management operation (including scaling) can be performed on a cluster at a time. For example, if an upgrade is in progress, scaling cannot be performed until the upgrade completes.

EBS versus ephemeral storage

Caution: When you configure a load-based policy, you may receive a warning about using attached EBS volumes over a certain size. While instances are in a STOPPED state (scaled-down), EBS volumes continue to be charged. In order to reduce costs, Cloudera recommends using instances with ephemeral storage (for example, the r5d or m5d instance families) for compute nodes. Ephemeral volumes will also result in better performance of the workloads–they are typically faster, and don’t utilize network throughput when writing data.

YARN Decommission Timeout

- Schedule-based downscaling will attempt to remove nodes to reach the target count, regardless of whether the node has running tasks, or data for a running job. You can use YARN DecommissionTimeout to control the leeway available to jobs beyond the scheduled decommission time. For example, with YARN decommission timeout set to one hour, a scheduled downscale will result in the following behavior:
  - Vacant nodes (no running tasks, no tasks previously run for running jobs) will be removed immediately.
  - Other nodes will be marked as decommissioning, and will not accept new work.
  - The remaining nodes will be terminated once any jobs complete, or at the one hour (YARN decommission timeout) mark.

For load-based autoscaling, the YARN DecommissionTimeout should be set to 30 seconds. See Configuring autoscaling for more details.

Related Information

Manually recovering from load-based scaling failures

Configuring autoscaling

Configuring autoscaling

To configure autoscaling, add a load-based or schedule-based policy to a cluster and define the policy parameters.

Before you begin

If you are configuring a load-based autoscaling policy, you must set the YARN Node Decommission Timeout property to 30 seconds. Configure the following property in Cloudera Manager: yarn_resourcemanager_nodemanager _graceful_decommission_timeout_secs

Procedure

1. From the Cloudera Management Console, click Data Hub Clusters and then select the cluster that you want to add an autoscaling policy to. You can add autoscaling policies to clusters after they have been created, not during the cluster creation process.
2. From the cluster details page, select the Autoscale tab and then click the slider button to enable autoscaling. Autoscaling is disabled by default.
3. Click Add Autoscale Policy and select Load-Based or Schedule-Based.

4. Define the policy parameters.
   - For load-based scaling:

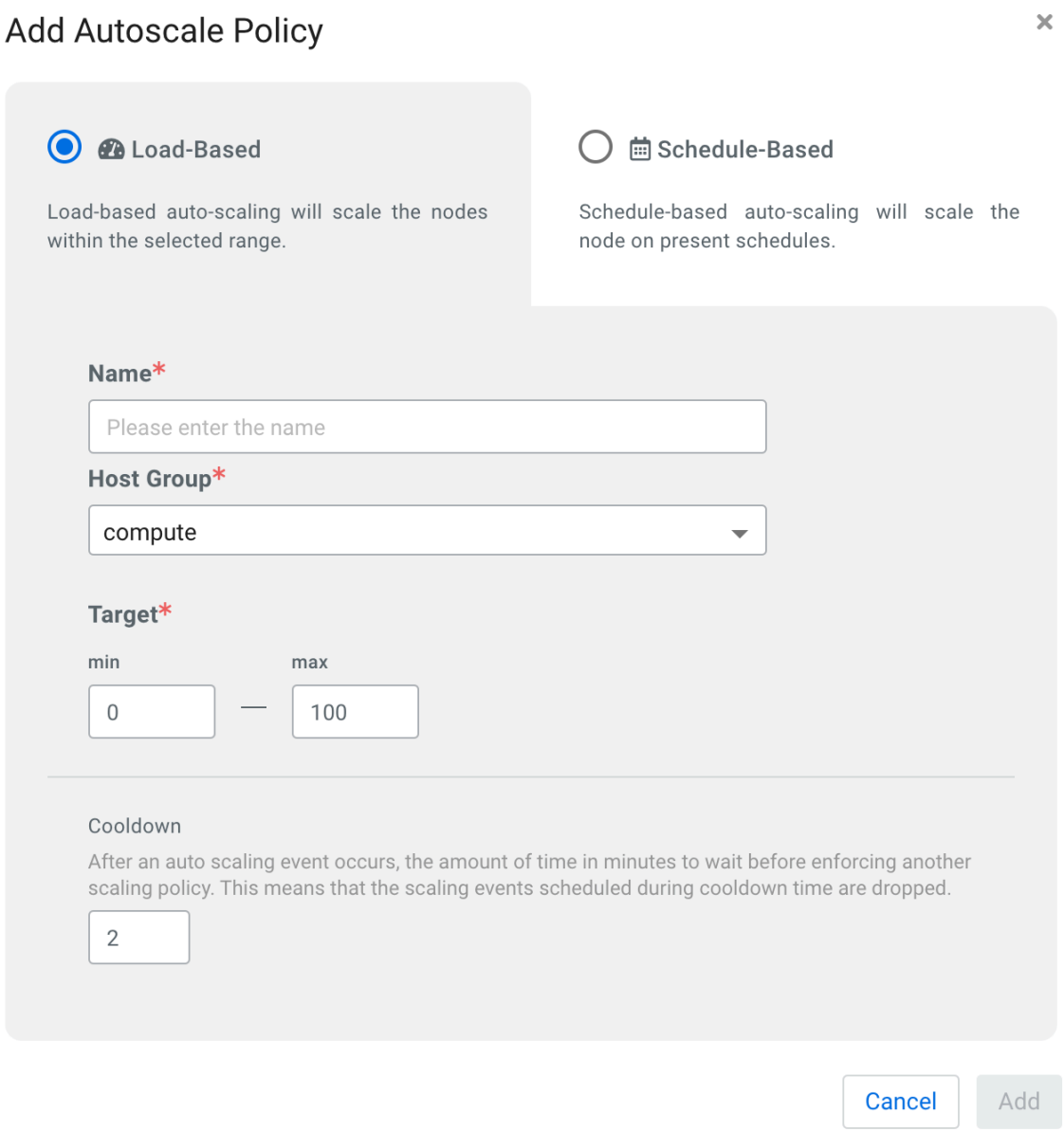

Parameter	Description
Name	Enter a unique name for the policy.
Hostgroup	Select the host group that you want to scale. The list of available host groups is determined by which host groups include services that can be scaled.
Target	Enter a minimum and maximum number of nodes for the policy. The maximum number of nodes determines how many instances are provisioned, but these instances are suspended and resumed as the workload demand requires. The policy will not provision instances beyond the maximum range of

Parameter	Description
	nodes that you define, regardless of the demand on the cluster.

- For schedule-based scaling:

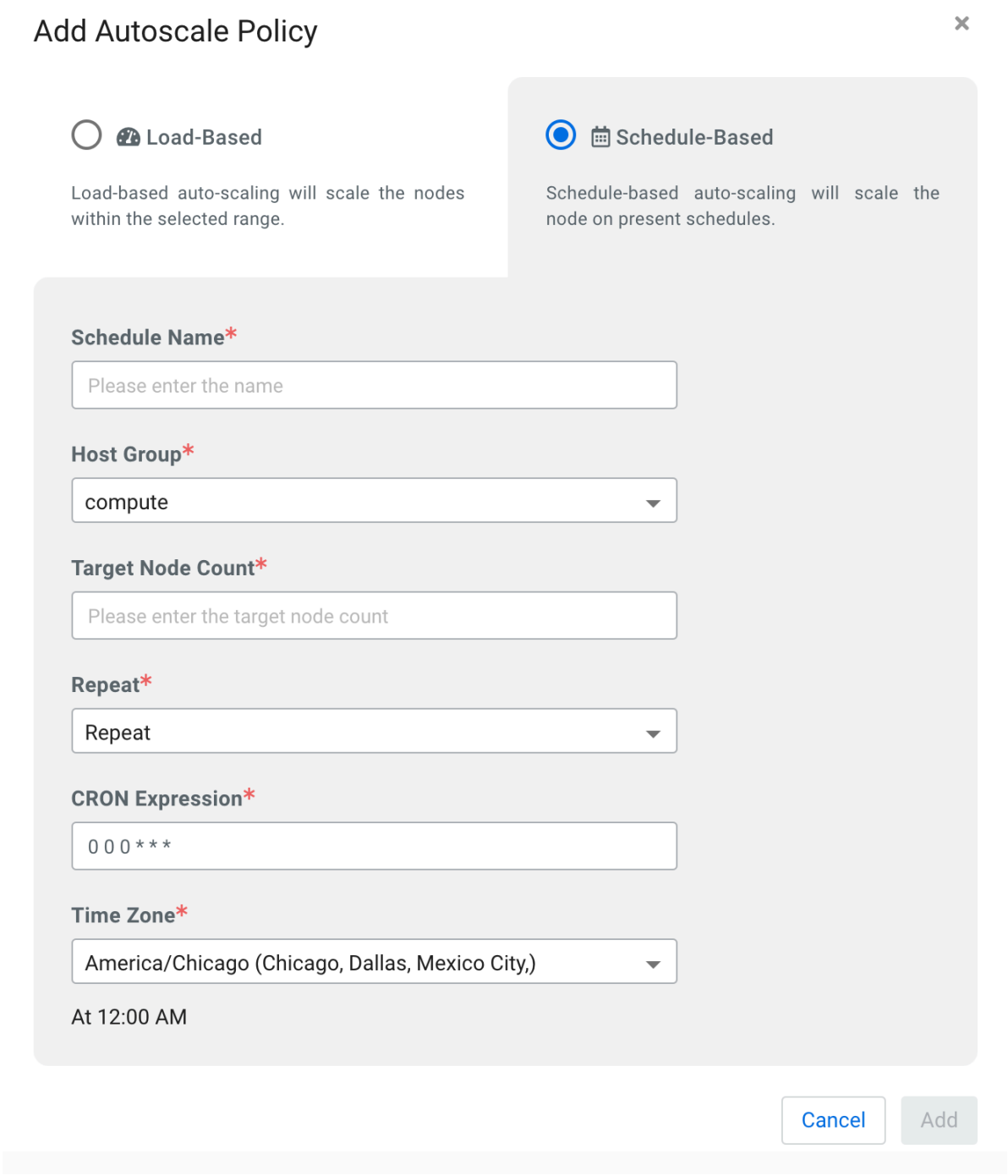

Parameter	Description
Schedule Name	Enter a unique name for the schedule.

Parameter	Description
Hostgroup	Select the host group that you want to scale. The list of available host groups is determined by which host groups include services that can be scaled.
Target Node Count	Enter the number of nodes that you want to upscale or downscale the host group to.
Repeat	Currently, only repeating schedules are supported.
CRON Expression	Enter a CRON expression string to define the details of the schedule that you want to create.
Time Zone	Select the appropriate timezone on which to base the CRON expression.

5. Click Add. The policy appears under Auto Scaling on the **Provision Data Hub** page.

Using YARN queues with autoscaling

Beyond the regular considerations to keep in mind while configuring YARN queues, a few additional aspects need to be considered, and some additional YARN configuration parameters are required.

Using a single queue for the entire YARN cluster

Nothing specific is required when YARN is configured with a single queue only.

Using multiple queues

Workloads executing in a queue will result in a scaling operation within the bounds of the resources allocated to the specific queue.

There are manual steps that are required on the YARN side to configure multiple queues for autoscaling. The same set of steps are also required before queue configuration is changed:

- Configure Resource Manager for queue-based autoscaling. Configuring Resource Manager requires setting the yarn.resourcemanager.autoscaling.plugin-type property to the value fine in Cloudera Manager.
- User-limit-factor and other queue configuration should be tuned as usual.

Note: If all of the queues in the system are allowed to use the maximum available resources in the cluster, 'percentage' mode can be used, by setting the max-capacity for each queue to 100%. The following steps are NOT required if using percentage mode, but are required otherwise:

- Queue configuration should use `Absolute` values instead of percentages.
- The `max-capacity` for `memory`, and optionally `cores` for a cluster should correspond to the total value available to YARN when scaled up to the configured `target maximum`.
- The `maximum capacity` for a queue should be configured to be less than or equal to the `maximum-capacity` for the entire cluster.

Instructions for these manual steps are in the following topic, *Configuring multiple YARN queues for autoscaling*.

Configuring multiple YARN queues for autoscaling

Manual steps are required to configure multiple YARN queues for load-based autoscaling.

Before you begin

See the previous topic, *Using YARN queues with autoscaling*.

Procedure

1. Set the Resource Manager plugin property to *FINE* using Cloudera Manager:
   a) In Cloudera Manager, select the YARN service.
   b) Go to the **Configuration** tab.
   c) Search for *PLUGIN-TYPE* and set the below property in the ResourceManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml field.

```
Name:  yarn.resourcemanager.autoscaling.plugin-type
Value: fine
```

   d) Save the changes.
   e) Start or restart the YARN ResourceManager service for the changes to apply.

   Important: If all of the queues in the system are allowed to use the maximum available resources in the cluster, 'percentage' mode can be used, by setting the max-capacity for each queue to 100%. If you are using percentage mode in this manner, you can skip the remainder of this task.

2. Ensure that the queue configuration uses absolute values instead of percentages:
   a) In the Cloudera Management Console, navigate to the specific Cloudera Data Hub cluster, and scroll to the **Services** section. Click Queue Manager.
   b) Click on the three vertical dots on the root and select Edit Queue Properties option.
   c) In the **Queue Properties** dialog box, select Absolute under Resource Allocation mode and click Save.

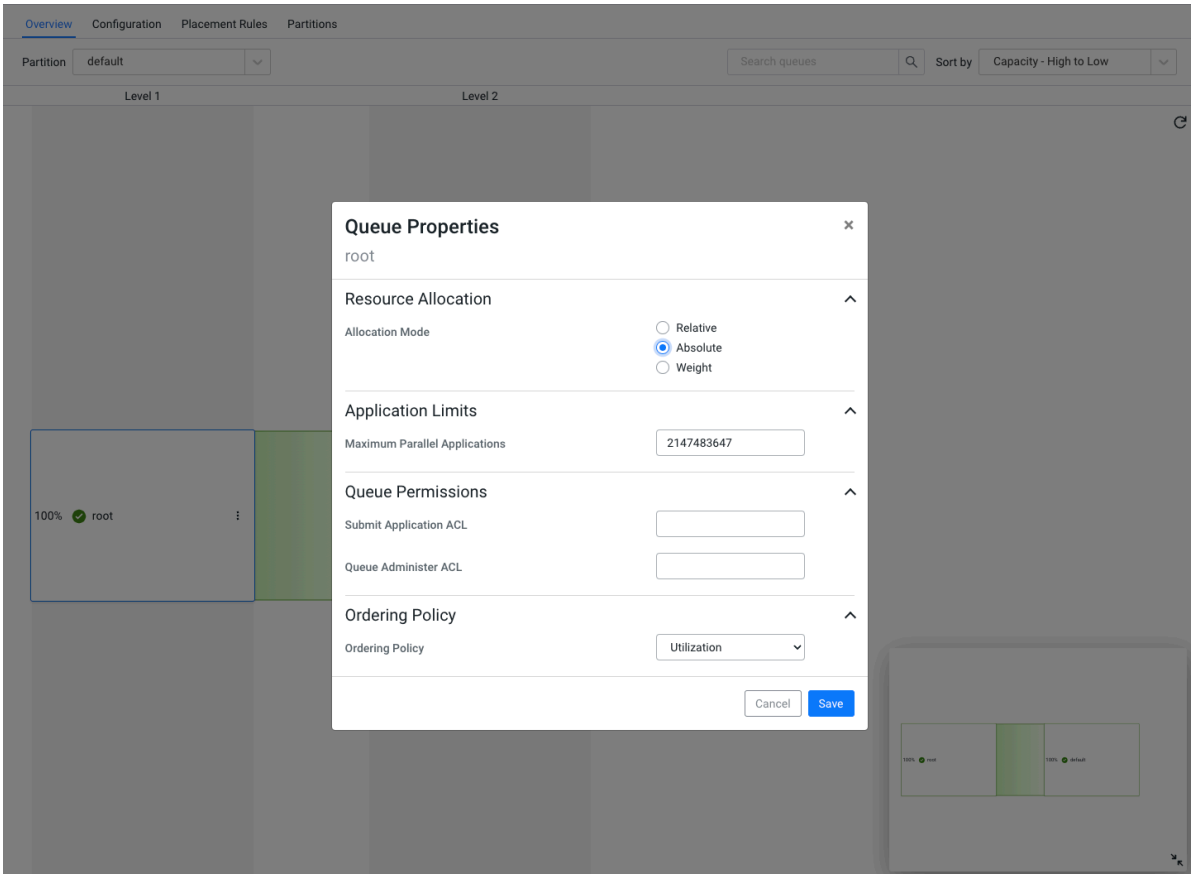

3. Disable autoscaling on the cluster. You can skip this step if you are on Cloudera Runtime version 7.2.17+.
4. Scale-up the autoscaling hostGroup manually to the new ‘Target max’ number of nodes. You can skip this step if you are on Cloudera Runtime version 7.2.17+.

5. Configure the queues:
   a) In the Queue Manager UI, click on the three vertical dots on the root, select Edit Root Queue and set the 'Configured Memory' and 'Maximum Memory' to the sum total memory of the maximum number of nodes that was set in the target field of Autoscale Policy parameters. Optionally, do the same for the CPUs.

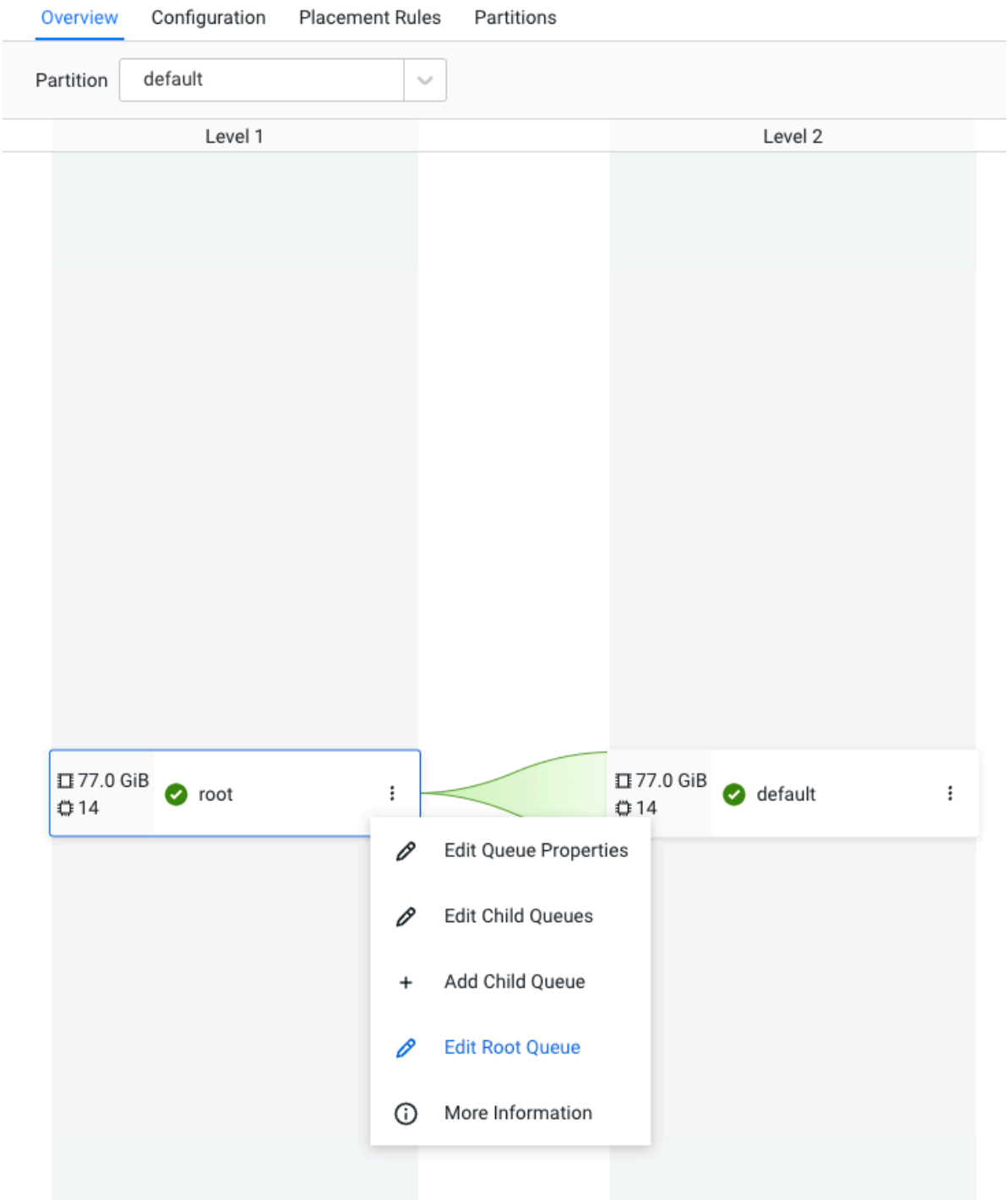

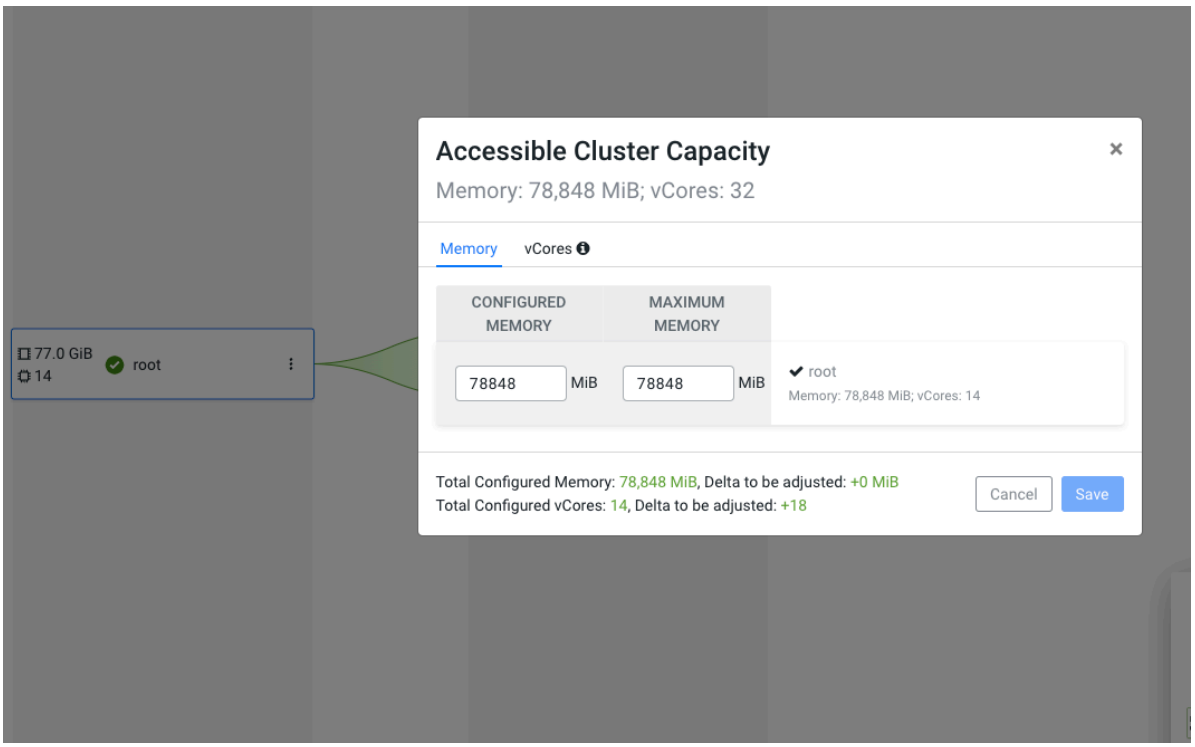

Note: With Cloudera Runtime 7.2.17+, you may exceed the Maximum Memory allowed for scenarios where the nodes will be added to the cluster manually or via autoscaling. If you exceed the Maximum Memory, select Check here to accept this change.

b) Edit the Child Queues as needed, and configure the resource allocation. In the example below, a new queue called `cq1` is added, which has a configured memory allocation, and maximum memory allocation of 35GB. The `default` queue has a configured memory allocation of 42GB, and a maximum memory allocation of 77GB.

- Jobs submitted to cq1 queue can cause the cluster to autoscale so that 35GB is available to the cq1 queue (within the cluster max of 77GB).

- Jobs submitted to the default queue can cause the cluster to autoscale so that 77GB is available to the default queue (which is the cluster max in this case).

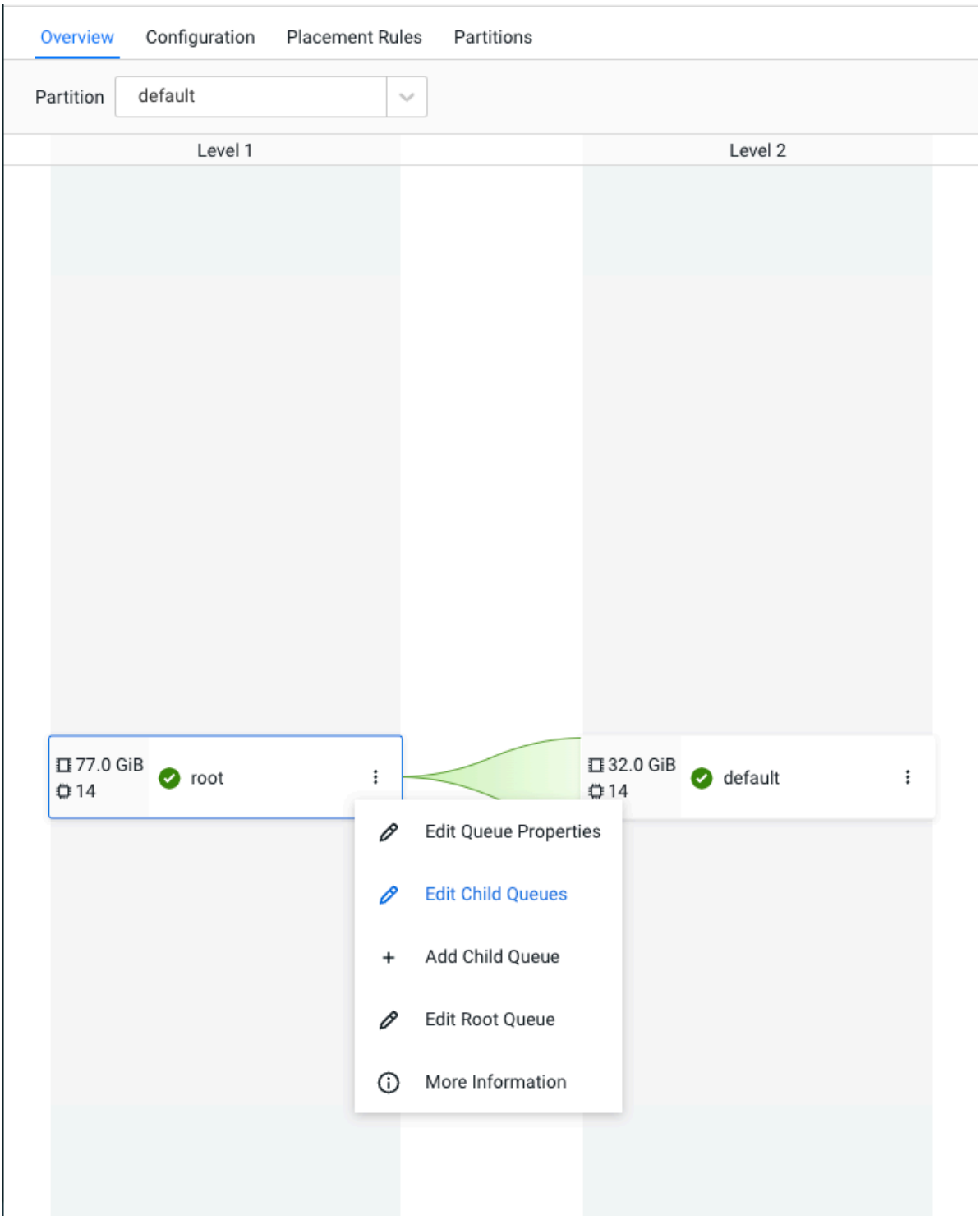

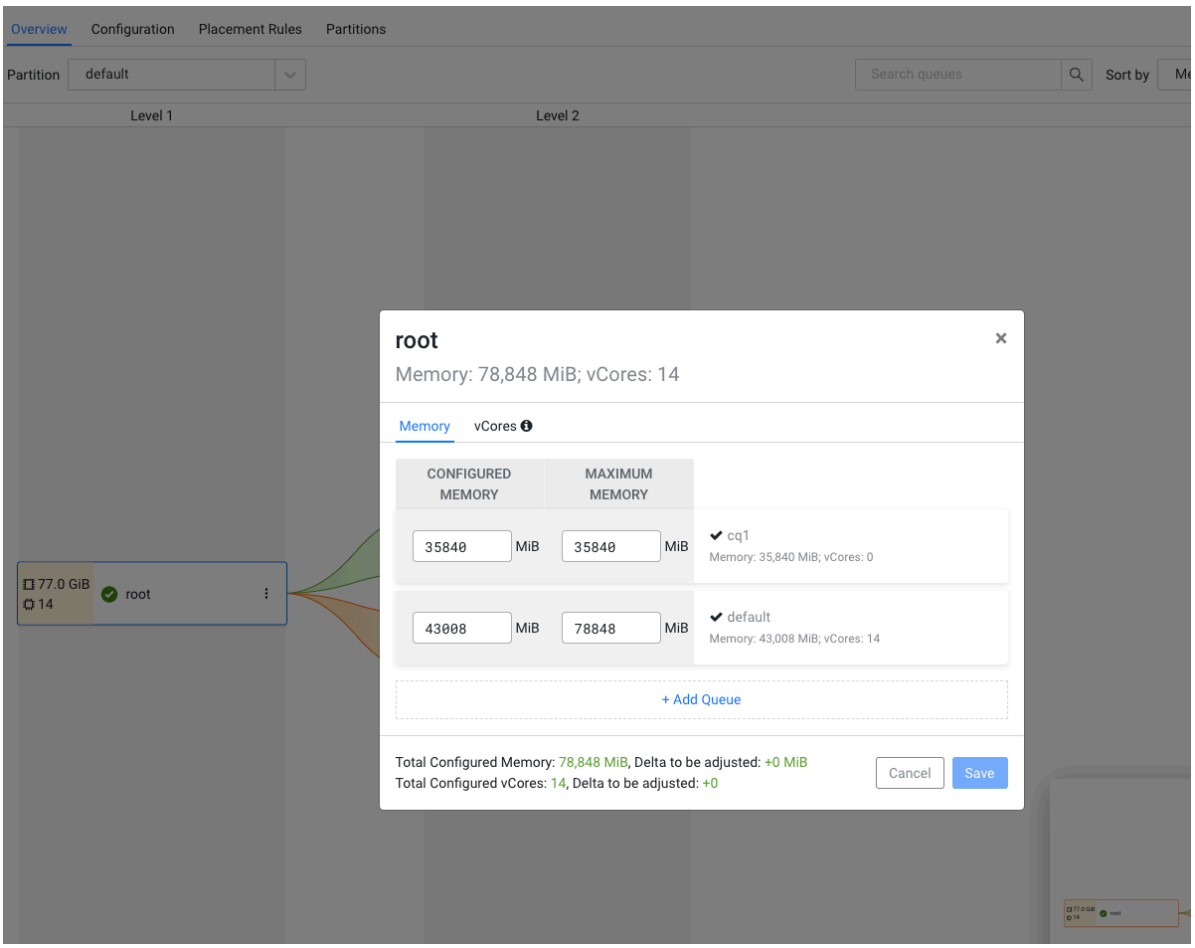

6. Re-enable autoscaling on the cluster.

Note: In case a queue has only one type of user, for example Hive user, increase the value of the user limits within a queue to a larger number like 100.

Note: When the 'Allocation Mode' is set to 'Absolute' - if re-configuring the min or max node count for AutoScaling, make sure to update the various queue configuration values via the Queue Manager UI appropriately.

Managing autoscaling

The CDP CLI has two options for reviewing autoscaling activity.

cdp datahub list-scaling-activities

Returns all of the scaling activities (or only failed scaling activities) in the given duration, or activities that have a start time in the given time interval between start-time and end-time for a particular cluster. YARN recommendation and YARN recommendation time for all the activities are also included in the output.

```
list-scaling-activities
--cluster <value>
[--duration <value>]
[--start-time <value>]
[--end-time <value>]
[--only-failed-scaling-activities | --no-only-failed-scaling-acti
vities]
[--max-items <value>]
```

```
[--starting-token <value>]
[--page-size <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

Table 4:

Parameter	Description
--cluster	Required. The cluster Name or CRN of the cluster for which you want to list all of the scaling activities.
--duration	Value of the duration in minutes from the current point to current point minus the duration. The default value is 60 minutes.
--start-time	Time in epoch milliseconds after which you want the activities.
--end-time	Time in epoch milliseconds until which you want the activities.
--only-failed-scaling-activities | --no-only-failed-scaling-activities	Determines if only the failed scaling activities for that particular cluster are returned.
--max-items	Indicates the maximum number of scaling activities that can be present in a page. Also indicates next-token in the output, which can be used as starting-token to get the next set of scaling-activities.
--starting-token	Number of the page from which you want all the pages. Default value is 0.
--page-size	Number of entries you want to get in each API call for listing scaling activities. The default value is 100.

Sample output:

```
{
"operationId":"61ed5711-6a80-45b7-850c-b428b838145c",
"startTime":"2025-01-24T10:25:00.115000+00:00",
"yarnRecommendationTime":"2025-01-24T10:25:00.115000+00:00",
"yarnRecommendation":"Successfully polled YARN to fetch cluster
 metrics. YarnRecommendedScaleUpCount: '0', YarnRecommendedDecom
mission: [], DeterminedScaleUpCount (based on step-size and maxA
llowedUpscale): '0', DeterminedDecommissionHosts (based on step-
size and maxAllowedDownscale): []. ",
"scalingActivityReason":" Cannot trigger any scaling operation
 as there was no recommendation for scaling from yarn",
"activityStatus":"METRICS_COLLECTION_SUCCESS"
}
```

cdp datahub describe-scaling-activity

Returns the scaling activity corresponding to the cluster and operationId provided by the user.

```
describe-scaling-activity
--cluster <value>
--operation-id <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

Table 5:

Parameter	Description
--cluster	Required. The cluster name or CRN of the cluster for which you want to fetch the particular scaling activity.
--operation-id	Required. The operation ID of the scaling activity that you want to fetch.

Manually recovering from load-based scaling failures

The mechanics for scaling are straightforward. There are four operations performed, and instances are not deleted during a load-based scale-down. Scale-up and scale-down operations are also meant to fail relatively fast, so that manual repair operations, if necessary, can be performed faster. Additionally, these operations can also be performed if there is another Management Operation (other than scaling) already in progress, which would otherwise block scaling.

The operations performed during a load-based scale-up and scale-down are outlined below.

- Scaleup: Start Instances on Cloud Provider: Specific instances for scale-up are started via the cloud provider. This can be replicated manually by changing the 'Instance State' on the cloud provider console.
- Scaleup: Commission Services via Cloudera Manager: The next step is to commission the services on these nodes. To replicate this manually, find these hosts using the filter: Status: Good Health, CommissionState: Decommissioned. Select them, then click Actions for Selected End Maintenance (Enable Alerts/Recommission) .

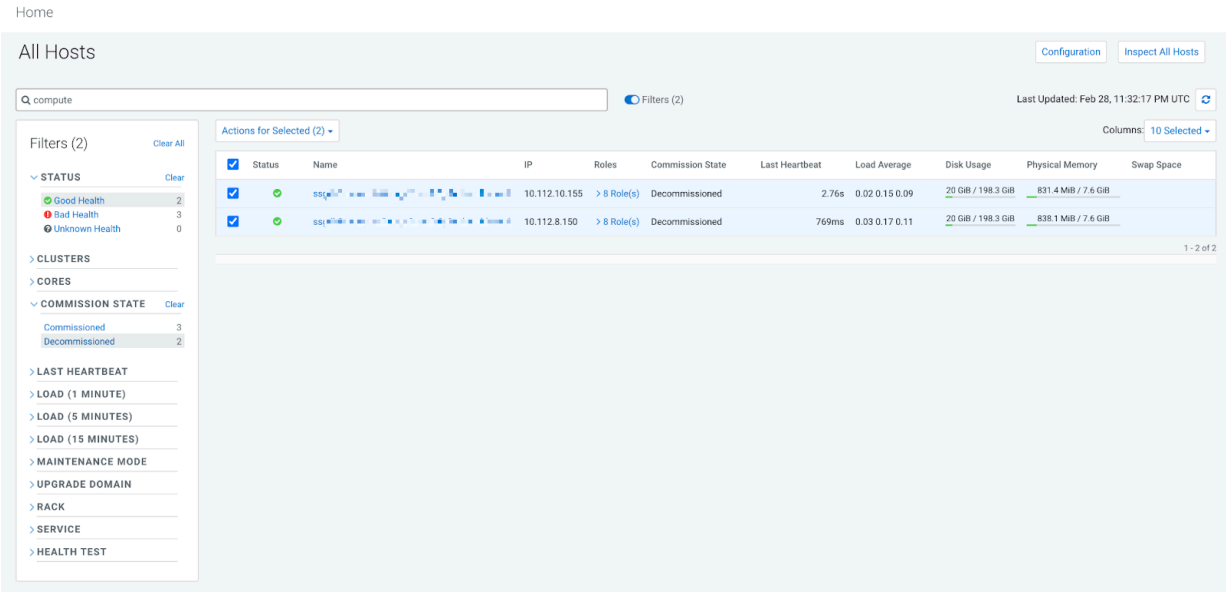

- Scale-down: Decommission Services via Cloudera Manager: This is the first step of a downscale operation. This can be replicated manually by selecting the relevant hosts in ClouderaManager as above, then click Actions for Selected Begin Maintenance (Suppress Alerts/Decommission) .
- Scale-down: Stop Instances on Cloud Provider: This is the second step of a scale-down operation, and specific instances are stopped via AWS or Azure. This can be replicated manually by changing the 'Instance State' on the cloud provider console.

Given the simplicity of the operations, and the fact that all the operations can be easily replicated by a cluster administrator, most failures can be handled quickly by following one or more of the procedures above. Some specific scenarios are outlined below.

Scale-up failures, where capacity is required

1. On the Autoscale tab for the cluster, use the toggle to disable autoscaling on the cluster.
2. Find stopped instances on the cloud provider console for the specific host group, and start them.

3. Wait for a few minutes for the instances to show up marked as "healthy" in Cloudera Manager (use the filter: Status: Good Health, Commission State: Decommissioned). Select instances from the appropriate host group and End Maintenance.

Cloudera Management Console has the following message: "Cloudera Manager reported health … … hostname1: [This host is in maintenance mode.], hostname2: [This host is in maintenance mode.]"

The cluster moves to ‘Node Failure’ state, and autoscaling is disabled. This typically indicates that some nodes are in the ‘STARTED’ state on the cloud-provider, but they are in ‘Maintenance Mode’ in Cloudera Manager. To remediate this, find the affected nodes in Cloudera Manager, and perform the steps in Scaleup: Commission Services via Cloudera Manager above. Wait for a few minutes for the Cloudera Management Console to sync state. After this, the cluster should move to the running state, and autoscale is re-enabled. Alternately, you can stop the specific nodes on the cloud provider.

Not enough nodes available - i.e. Autoscale configured with maxNodes, but the instance group has fewer than maxNodes available

Autoscaling will commission as many nodes as are available, and log an error in the Cloudera Management Console indicating that not enough nodes are available. Once all nodes are started, it will continue to try, causing a loop in which 0 nodes are added in each upscale attempt. To remedy this, when the cluster has all nodes in the RUNNING state, disable autoscaling and perform a scale-up of the host group to reach the ‘maxNodes’ configured for autoscaling (Actions Resize). Re-enable autoscaling.

Autoscaling FAQ

Review frequently asked questions about Cloudera Data Hub autoscaling.

A node is not running any tasks, why is it not being scaled down?

There are multiple possibilities:

- The cluster may already be at min-node-count.
- Check if running on a Cloudera Runtime version prior to 7.2.15, with max-capacity of queues not set to 100%.
- Check user-limit-factor. Is this preventing YARN from executing additional tasks for a user?
- Nodes that are not running containers, but have run containers for a running job, will not be considered for load-based downscaling. This is because the data generated on these nodes could be required while the job is running.

Will load-based autoscaling remove nodes that have running tasks?

Not unless the maximum node count limit is reduced. This forces the removal of certain nodes to reach the new limit.

I've encountered the following error: "Autoscaling Trigger Failed. Autoscaling Collection metrics via user <CRN> failed." How should I proceed?

For Cloudera accounts with a large number of users, autoscaling might encounter failures when capturing cluster metrics to scale up/down. If you encounter this issue please contact Cloudera Support.

Upgrading Data Lake/Cloudera Data Hub database

This document describes the process to upgrade the database to the latest version supported by Cloudera Public Cloud services. You may use Cloudera UI or CDP CLI to perform this upgrade.

About this task

Note: If you are upgrading an Azure Single Server database to Azure Flexible Server, you can read a specific process description in Upgrading Azure Single Server to Flexible Server.

Several Cloudera Public Cloud services, including the Data Lake cluster and the Cloudera Data Hub cluster templates and Data Services, require a relational database. Most of these databases are external and are provisioned during the initial deployment of the respective service.

The databases used by the Data Lake and some of the Cloudera Data Hub templates are hosted on external instances that are provisioned during the initial deployment of the respective service. For these external databases Cloudera Public Cloud leverages cloud-native service offerings of the three supported Cloud Service Providers (AWS RDS for PostgreSQL, Azure Database for PostgreSQL, and Cloud SQL for PostgreSQL).

Databases used by other Cloudera Data Hub templates are hosted on an embedded database instance, typically co-located on the Cloudera Manager host, in order to reduce the resource footprint.

Cloudera provides a database upgrade capability in Public Cloud that allows moving both external and embedded databases to a higher major version.

The database upgrade is a fully automated operation. The upgrade process itself completes all of the required steps, including creating a backup, stopping and upgrading the database, restarting the database, and running post-upgrade maintenance tasks. You are not required to manually stop the Postgres instances before the upgrade.

Attention: In accordance with the PostgreSQL Versioning Policy, the cloud database services mentioned above may end support for PostgreSQL major version 11 on November 9, 2023 or shortly thereafter. Different cloud providers may have extended support for PostgreSQL 11. Despite this, it is recommended to upgrade to PostgreSQL 14 when the upgrade is available to you in Cloudera.

Important: In order to avoid disruption to the deployed Data Lake and Cloudera Data Hub services, caused by configuration changes to the underlying database service by the Cloud Service Providers, it is recommended that the database upgrade in Cloudera Public Cloud is performed before the End of Life date.

If you wish to disregard this recommendation, you may do so considering the risks involved as per the Cloud Service Provider policies.

The database upgrade is a separate operation, complementary to the existing maintenance, minor/major version and OS upgrades, as described in the Cloudera Public Cloud Upgrade Advisor.

This is a one-time operation. Once the database of a Data Lake or Cloudera Data Hub has been successfully upgraded to the newer major version, no further action is needed for the respective cluster.

Note: Cloudera recommends that the database upgrade is performed separately from other upgrade actions.

If a cluster uses a database that requires an upgrade, you will receive a notification, as shown below, on the Cloudera Management Console UI.

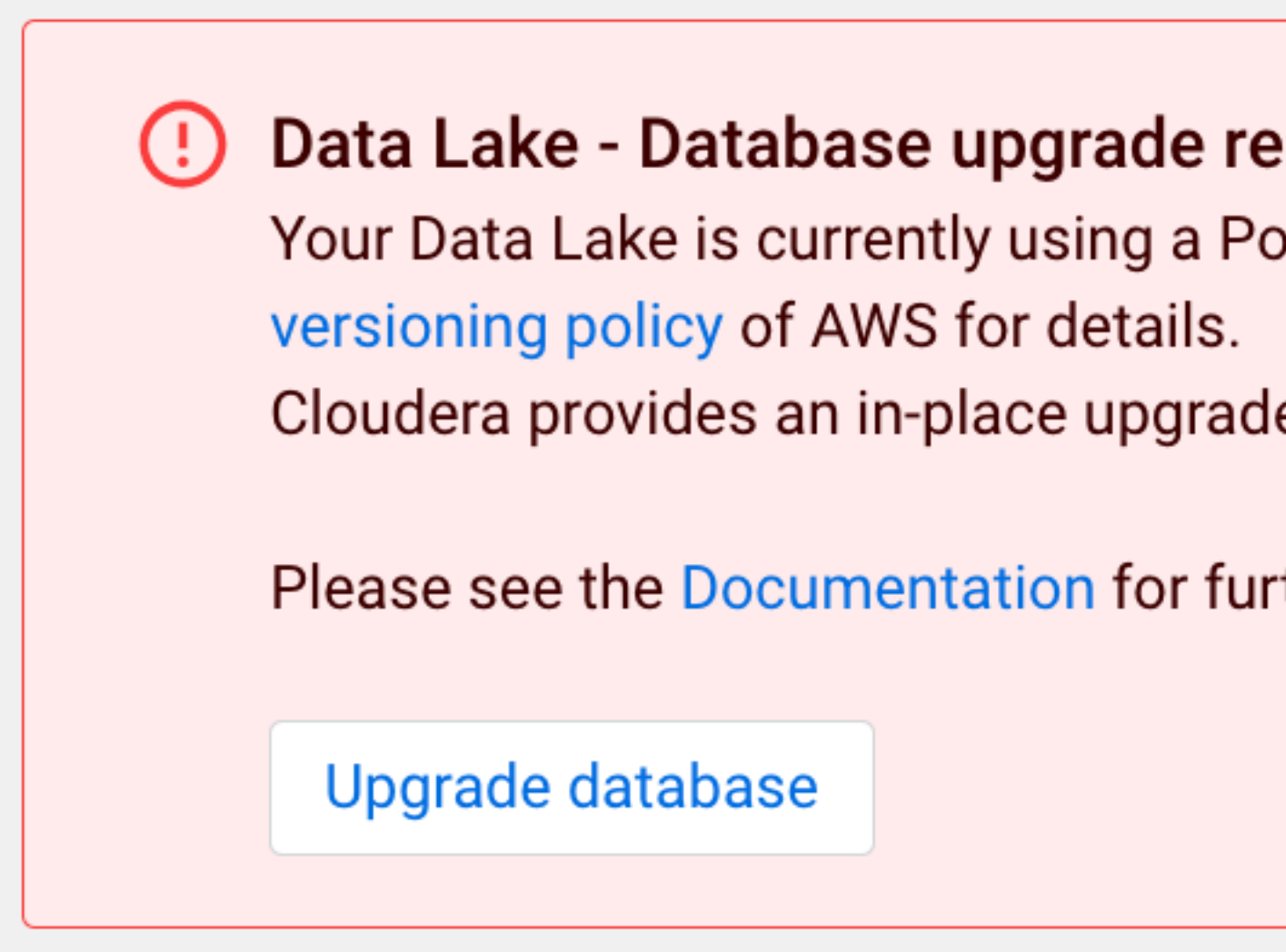

Running the database upgrade operation on the Cloudera Data Hub cluster will mean that all cluster services (Cloudera Manager and Cloudera Runtime services) are stopped on the cluster automatically without having to stop them manually. For the Data Lake database upgrade, it is recommended that attached Cloudera Data Hub clusters and Data services are in stopped state.

Note: Cloudera strongly recommends stopping all workloads in Data Services that interact with the Data Lake.

If you are concerned about stopping the workloads in your deployment, contact Cloudera support for a custom upgrade path.

For AWS and GCP environments, the Database Upgrade operation will trigger a backup and a major version upgrade for the attached external database. But for Azure environments, the mechanism is different; as in the background, it will create a new database instance with a higher major version and transfer the data from the older database instance.

Note: During Postgres database upgrade for Data Lakes and Cloudera Data Hub clusters on AWS and Azure, there is a possibility that manually changed configs of the database server will be reverted to the original configs. For more information, see Database upgrade known limitations.

Instructions

Here are the UI and CLI instructions to perform Database Upgrade on Data Lake and Cloudera Data Hub:

For CDP UI

Steps

1. In Cloudera Management Console, go to Environments. Select the cluster to perform the upgrade from the list of available clusters. The clusters are eligible for this upgrade are indicated in the right most column:

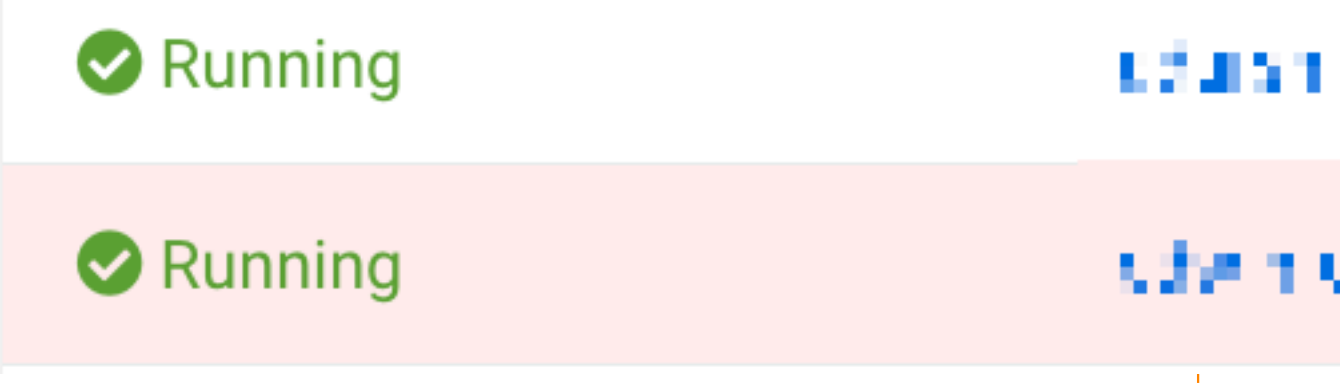

2. Once you select the cluster, you will see a message asking to update the Postgres version. Click the Upgrade database.

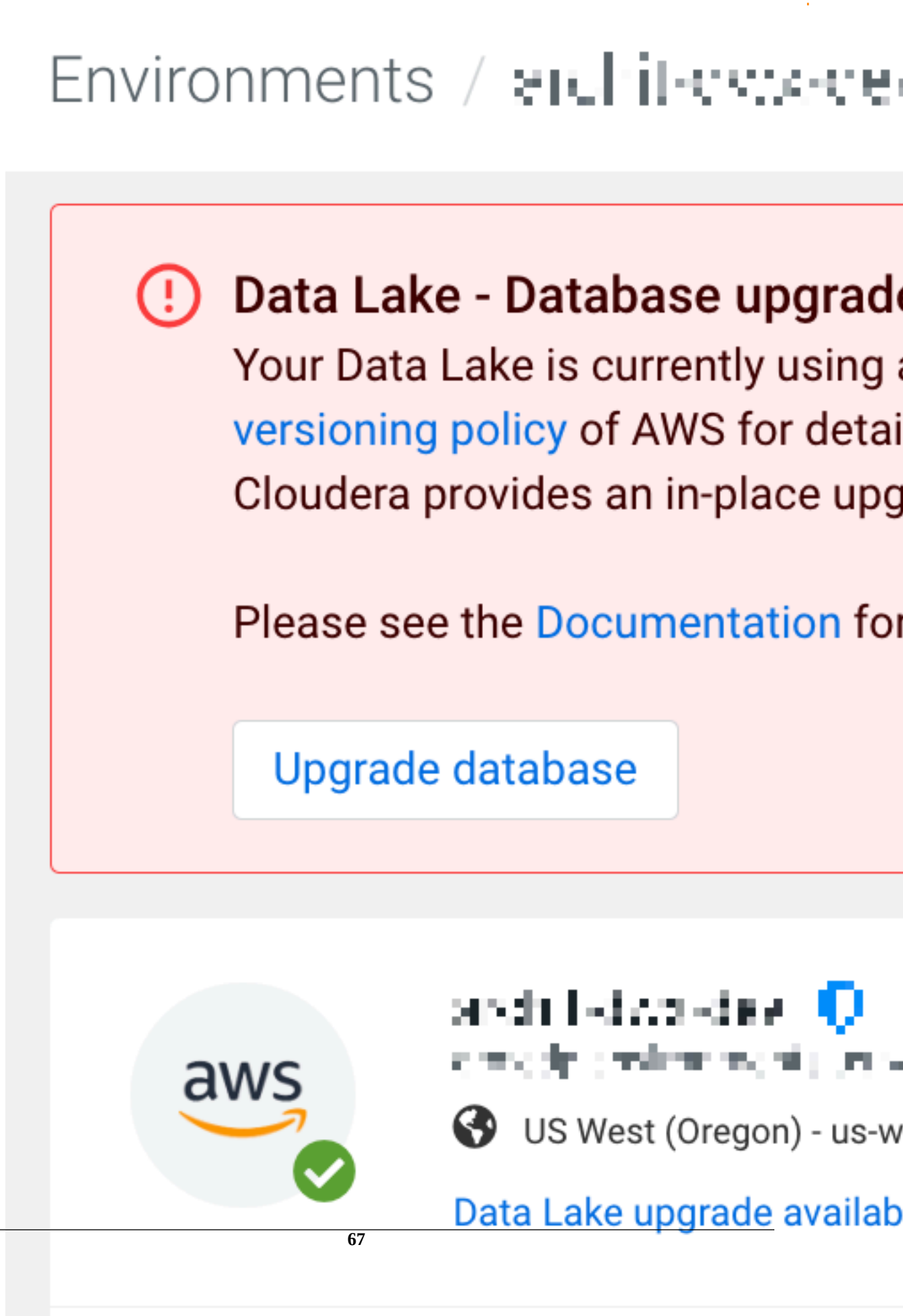
Environments /
Data Lake - Database upgrad
Your Data Lake is currently using
versioning policy of AWS for detai
Cloudera provides an in-place upg
Please see the Documentation fo
Upgrade database
aws
US West (Oregon) - us-w
Data Lake upgrade availab

3. Click Upgrade in the confirmation box.

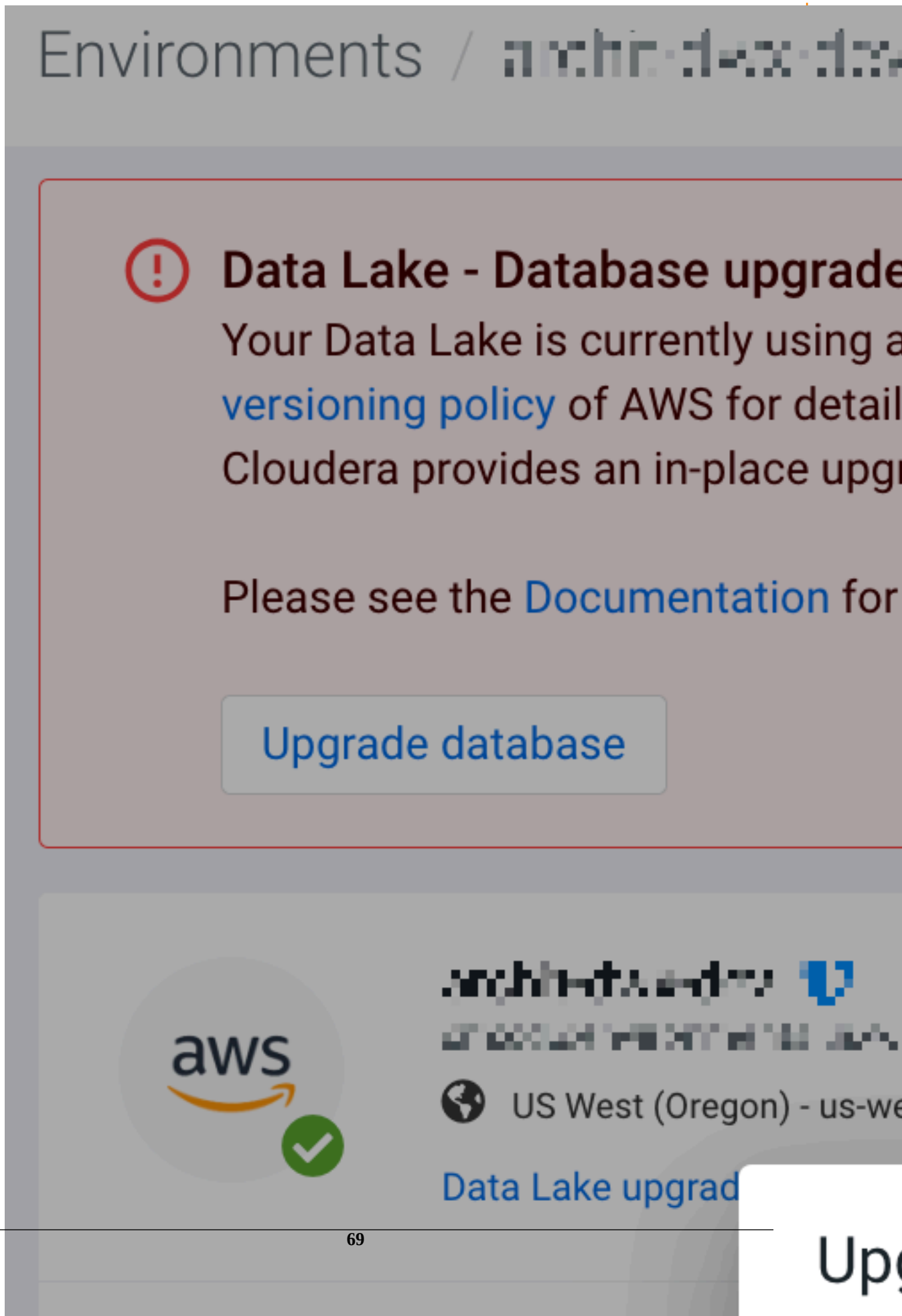
Environments /
Data Lake - Database upgrade
Your Data Lake is currently using a
versioning policy of AWS for detail
Cloudera provides an in-place upg
Please see the Documentation for
Upgrade database
aws
US West (Oregon) - us-we
Data Lake upgrad

4. Once the Data Lake database is updated, check for the Cloudera Data Hub clusters for that Data Lake, if there is any database upgrade notification and perform the database upgrade as described above.

   Note: The Database upgrade needs to be performed in every Data Lake and Cloudera Data Hub cluster separately, one by one.

For CDP CLI

Data Lake Database upgrade:

You can perform Data Lake database upgrade using cdp datalake start-database-upgrade CLI command.

The --target-version parameter is optional. If you do not provide it, the database will be upgraded to PostgreSQL 14.

```
cdp datalake start-database-upgrade --help --form-factor public
NAME
       start-database-upgrade  -  Upgrades the database of the Data Lake
 clus-
       ter.
DESCRIPTION
       This command initiates the upgrade of the database  of  the  Data
 Lake
       cluster.
SYNOPSIS
            start-database-upgrade
          --datalake <value>
          --target-version <value>
          [--cli-input-json <value>]
          [--generate-cli-skeleton]

OPTIONS
       --datalake (string)
          The name or CRN of the Data Lake.

       --target-version (string)
          The database engine major version to upgrade to.

          Possible values:

          o VERSION_14
```

Cloudera Data Hub Database upgrade:

You can perform Cloudera Data Hub database upgrade using cdp datahub start-database-upgrade CLI command.

The --target-version parameter is optional. If you do not provide it, the database will be upgraded to PostgreSQL 14.

```
cdp datahub start-database-upgrade --help --form-factor public
NAME
       start-datahub-upgrade  -  Upgrades the database of the Data Hub c
lus-
       ter.

DESCRIPTION
       This command initiates the upgrade of the database  of  the  Data
Hub
       cluster.
SYNOPSIS
            start-database-upgrade
          --datahub <value>
```

```
          --target-version <value>
          [--cli-input-json <value>]
          [--generate-cli-skeleton]

OPTIONS
       --datahub (string)
          The name or CRN of the Data Hub.

       --target-version (string)
          The database engine major version to upgrade to.
          Possible values:
          o VERSION_14
```

The progress of the upgrade can be tracked on the respective service's Event History page. You can verify a successful database upgrade in the Event History or in the Database tab of the cluster. Once the upgrade is complete, Cloudera recommends verifying your workloads before attempting an additional Cloudera Runtime or OS upgrade.

Note: As part of the database upgrade operation, PostgreSQL 14 client binaries will be installed on the cluster hosts, replacing earlier client versions. This may impact third-party components or custom services running on the cluster hosts.

Database upgrade known limitations and troubleshooting

Below are the known limitations associated with the database upgrade of Data Lake and Cloudera Data Hub clusters and ways to troubleshoot them.

Note: For troubleshooting information related to upgrading to Azure Flexible Server, see Troubleshooting Flexible Server. When you are upgrading an Azure Single Server database to PostgreSQL version 14, the Azure Single Server is automatically updated to Azure Flexible Server.

Known limitations and troubleshooting:

- Performing the Database Upgrade on Cloudera Runtime versions 7.2.6 or below

  Cloudera has verified PostgreSQL version 11 compatibility for Cloudera Runtime version 7.2.7 and above. There is no known reason why older Cloudera Runtime versions should not be compatible with PostgreSQL version 10.

  Workaround: You can request an entitlement that allows the Database Upgrade to be performed on older Cloudera Runtime versions on an exceptional basis.
- Performing the Database Upgrade on Data Lakes with attached Cloudera Data Hub clusters that cannot be stopped

  Technically, the Database Upgrade can be performed on a Data Lake without stopping the attached Cloudera Data Hub clusters. However, please be aware that during the upgrade, the Hive Metastore database will likely become temporarily unavailable and this can cause serious disruption or in the worst case can result in an inconsistent state for workloads running in Cloudera Data Hub clusters or Data Services.

  Workaround: If you acknowledge the risk and confirm that all cluster services and third party components relying on the Hive Metastore will be stopped for the time of the Database Upgrade, Cloudera can grant an entitlement that allows performing the upgrade with a running Cloudera Data Hub cluster on an exceptional basis.
- PostgreSQL client binaries will be upgraded to version 11 on all clusters hosts

  As part of the upgrade process we will try to install the PostgreSQL 11 libraries, pulling them from archive.cloudera.com. If the installation of these libraries does not succeed, a notification message will be sent that installation was attempted, but failed for some reason (network connectivity issues, etc).

  Workaround: Follow the process to install the libraries manually, see Installing PostergeSQL 11 packages manually.

  Note: Failing to install the PostgreSQL 11 client libraries as part of the Database Upgrade process will cause the Data Lake backup and restore operations to stop working correctly.

- Upgrading embedded databases

  Cloudera Data Hub clusters using an embedded database will not require the Database Upgrade operation to be performed. The embedded database, including client libraries will be automatically upgraded during an OS upgrade.

  Note: This capability is currently disabled and will be activated later.

  Workaround: If you need to upgrade the embedded databases of your Cloudera Data Hub clusters, contact Cloudera to enable this capability on an exceptional basis. Once this entitlement has been granted, your embedded databases can be upgraded by performing an OS upgrade.
- Exceeding the End of Life deadline

  Data Lake and Cloudera Data Hub clusters that are not upgraded until November 10, 2022 will continue to run on a PostgreSQL version 10 instance of the underlying AWS, Azure or GCP database service. As this instance will be considered End-of-Life (EoL) by the respective Cloud Service provider, they may reserve the right to schedule an automated major version upgrade, resulting in a temporary downtime. In the case of extreme events the Cloud Service Provider may also stop the instance, see Versioning policy- Azure Database for PostgreSQL. In either case, your Cloudera Public Cloud workloads may be seriously impacted.

  Workaround: Cloudera recommends performing the Database Upgrade via the Cloudera UI, or CLI as soon as possible.
- Possibility of custom config reset after Database upgrade on AWS and Azure

  During Postgres database upgrade for Data Lakes and Cloudera Data Hub clusters there is a possibility that manually changed configs of the database server, that the control plane does not know about, will be reverted to the original configs.

  Reason: On Azure the custom config can possibly reset during the database upgrade because Cloudbreak deletes and recreates the database server with the configs that the control plane knows about, so custom configs will be reverted.

  On AWS if SSL enforcement is enabled then the database server uses a custom parameter group with the SSL enforcement settings (created by control plane) and if the customer made any custom changes to this custom parameter group then those changed will be reverted, because the database upgrade requires the recreation of the custom parameter group.

Installing Postgres 14 packages manually

Steps for manual installation of PostgreSQL 14 packages.

About this task

The last step of the Database upgrade flow is the installation of PostgreSQL 14 packages on the cluster hosts. This is relevant in the case of an operating system image that does not yet contain the PostgreSQL 14 packages.

The required repositories are being hosted in the same location that is used for Cloudera Runtime upgrades: https://archive.cloudera.com/p/postgresql/postgresql14/redhat7/

If for some reason the package installation fails, it is required for the customers to manually install the aforementioned packages because otherwise the pg_dump utility driving the backup functionality will stop working.

- Method 1 : Installation using Cloudera hosted

  This method works only if you have proper network access and paywall credentials to the archive.cloudera.com repository as the required metadata is already pushed onto the nodes during the RDS upgrade process.

  SSH into the master node and run the following with superuser privileges:

  ```
  {code}
  ```

```
source activate_salt_env
salt '*' state.apply postgresql/pg14-install
{code}
```

- Method 2: Manual installation

  Using this method you will install PostgreSQL packages using the official repo file

  1. SSH into the master node and run the following with superuser privileges (install PostgreSQL packages using the official repo file)

     ```
     yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL
     -7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
     ```

  2. Install required packages

     ```
     yum install -y postgresql14-server postgresql14 postgresql14-contrib pos
     tgresql14-docs
     ```

Installing Postgres 11 packages manually

Steps for manual installation of PostgreSQL 11 packages.

About this task

The last step of the Database upgrade flow is the installation of PostgreSQL 11 packages on the cluster hosts. This is relevant in the case of an operating system image that does not yet contain the PostgreSQL 11 packages.

The required repositories are being hosted in the same location that is used for Cloudera Runtime upgrades: https://archive.cloudera.com/p/postgresql/11/redhat7/

If for some reason the package installation fails, it is required for the customers to manually install the aforementioned packages because otherwise the pg_dump utility driving the backup functionality will stop working.

- Method 1 : Installation using Cloudera hosted

  This method works only if you have proper network access and paywall credentials to the archive.cloudera.com repository as the required metadata is already push onto the nodes during the RDS upgrade process.

  SSH into the master node and run the following with superuser privileges.

  ```
  source activate_salt_env
  salt '*' state.apply postgresql/pg11-install
  ```

- Method 2: Manual installation

  Using this method you will install PostgreSQL packages using the official repo file

  1. SSH into the master node and run the following with superuser privileges (install PostgreSQL packages using the official repo file)

     ```
     yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL
     -7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
     ```

  2. Install required packages

     ```
     yum install -y postgresql11-server postgresql11 postgresql11-contrib pos
     tgresql11-docs
     ```

Rotating database certificates

AWS requires the rotation of the SSL/TLS certificates used for secure communication between Cloudera Public Cloud Data Lakes and certain Data Hubs and the external AWS RDS database instances that they rely on. Cloudera Public Cloud provides multiple options to perform the required RDS certificate rotation.

Cloudera Public Cloud Data Lakes and certain Cloudera Data Hub clusters rely on Amazon Web Services (AWS) Relational Database Service (RDS) database instances that are provisioned by Cloudera during the creation of the respective resources. Similarly to other compute resources, these database instances are created in the cloud accounts of customers. The deployed Cloudera resources use a secure connection to communicate with the database using the SSL/TLS certificates issued by AWS.

SSL/TLS certificates used by Amazon RDS instances created before December 2022 are potentially using certificates that will expire in 2024. Amazon RDS instances with an expiring certificate require that a certification rotation is performed. This will replace the soon expiring certificate with a newer one. If the RDS certification rotation is not performed by 22nd August, 2024, Amazon will perform the rotation for each RDS instance and as a result, the affected Cloudera clusters will not be able to connect to the external database instances.

The actions you need to take depend on whether you have SSL enforcement enabled or disabled on the Data Lake or Cloudera Data Hub cluster as shown in the **Database** tab of the Data Lake or Data Hub details page. This setting controls whether the database server mandates the use of an encrypted connection. It is recommended to check SSL enforcement setting on the Data Lake of your environment and separately for each Cloudera Data Hub cluster.

It is strongly recommended that, when SSL enforcement is enabled, you do not perform the RDS certificate rotation directly using AWS tools, as this can lead to a service degradation or an outage of Cloudera services using the respective database. Instead, follow the instructions below so that the new certificates can be properly installed by Cloudera before changing them on the database instances. Once you have performed the required steps in Cloudera, there are no additional actions needed in the AWS console. During the database certificate rotation, Cloudera will automatically make the changes required and rotate the certificate of the attached AWS RDS database.

To determine whether you have SSL enforcement enabled on your Data Lake, perform the following steps:

1. Sign in to the Cloudera web interface.
2. Navigate to Cloudera Management Console Environments .
3. Select the environment by clicking the environment name.
4. Click the Data Lake tab.
5. Scroll to the bottom of the page and select Database.

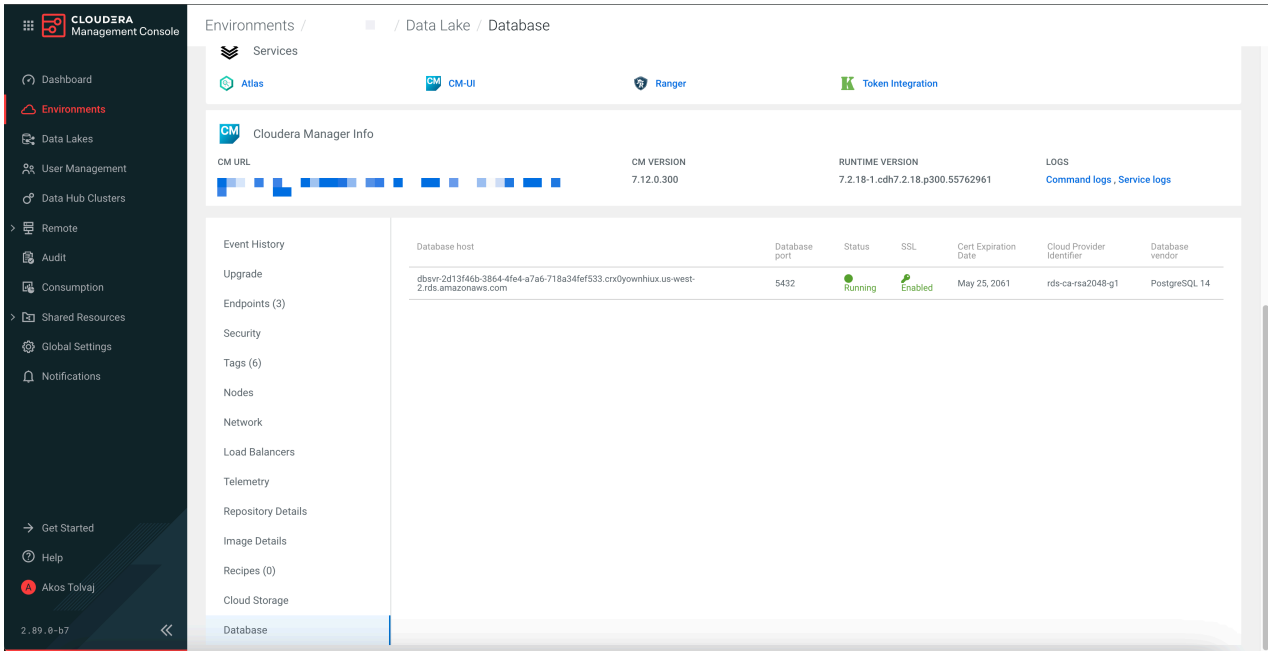

Similarly, perform the following steps on each Cloudera Data Hub cluster to determine whether you have SSL enforcement enabled:

1. Sign in to the Cloudera web interface.
2. Navigate to Cloudera Management Console Environments .
3. Select the environment by clicking the environment name.
4. Click the Data Hub tab.
5. Select a Cloudera Data Hub cluster.
6. Scroll to the bottom of the Data Hub details page and select Database.

If the value for **SSL** is **Enabled**, it means SSL enforcement is enabled on your Data Lake or Cloudera Data Hub cluster, and you must perform certificate rotation as described in *Rotating database certificates when SSL enforcement is enabled*. If the value is **Disabled**, no action is required on that particular Data Lake or Cloudera Data Hub cluster. For further information, see *Rotating database certificates when SSL enforcement is disabled*.

Rotating database certificates when SSL enforcement is enabled

When SSL enforcement is enabled on your database server, do not perform the RDS certificate rotation directly using AWS tools, as this can lead to a service degradation or an outage of the respective resource. Use the options provided by Cloudera Public Cloud to perform the required RDS certificate rotation instead.

Required role (Data Lakes): EnvironmentAdmin

Required role (Data Hub): DataHubAdmin or EnvironmentAdmin

If the following warning message is displayed on top of the Cloudera Management Console window, you must perform the TLS/SSL certificate rotation:

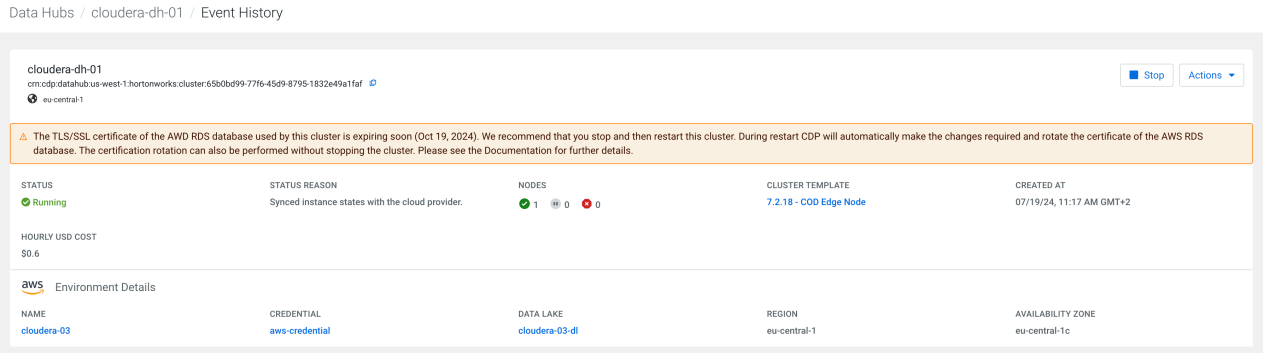

If you do not see the warning message, your clusters are not affected or the rotation has already been performed when you recently stopped and started your cluster or the environment.

Cloudera provides two options for rotating the expiring database certificates, with and without stopping your cluster:

Rotating the database certificate by stopping and starting your cluster

This is the default and most simple way to let Cloudera perform the certificate rotation and all changes required.

Instructions

1. Stop all Cloudera Data Hub clusters where the warning about expiring database certificates is shown.
2. Stop and start your Data Lakes where the warning message is displayed.

   CDP will automatically perform all Data Lake and AWS RDS changes required.
3. Start all Data Hubs that you have previously stopped.

   Cloudera will automatically perform all Cloudera Data Hub and AWS RDS changes required where the warning message is displayed.

You can perform the stop and start either via UI or CLI, see Stop a cluster.

Note: It is recommended that you also suspend workloads running on Data Services attached to the Data Lake during this operation.

Rotating the database certificate without stopping the cluster

Stopping the Data Lake or Cloudera Data Hub clusters is not feasible in some scenarios, as workloads need to be stopped and data stored on ephemeral disks or ephemeral cache is lost during cluster restart.

In certain cases, Cloudera allows the database certificate rotation to be performed without stopping your cluster. If your cluster supports this, you will see a Rotate Database Certificate button in the warning message.

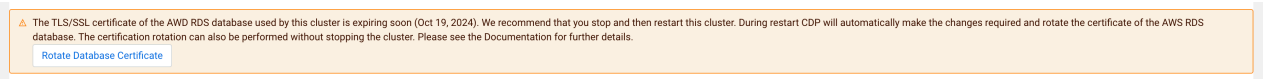

The Rotate Database Certificate button is only available if a rolling restart is supported on your cluster. The list of supported Data Lakes and Cloudera Data Hub clusters is available in the Rolling upgrades overview.

Important:

The certificate rotation can also be performed on Cloudera Operational Database databases by opening the cluster backing the Cloudera Operational Database database in the Data Hub screen and clicking the Rotate Database Certificate button there. However, this is only supported for Cloudera Operational Database clusters whose storage is selected as HDFS or cloud without ephemeral storage, and not supported on using Cloud Storage with Caching storage type.

Under some circumstances, a rolling certificate rotation may not be supported for a Data Lake or Cloudera Data Hub cluster, but can be enabled through entitlement. For information about obtaining this entitlement, contact Cloudera Customer Support.

Instructions

To perform a rolling restart, click the Rotate Database Certificate button on top of the screen, inside the warning message.

Note: Your cluster must be in a healthy status, otherwise the certificate rotation is not possible.

Important: You must rotate the certificates on each Data Hub cluster first, and then proceed with rotating the certificates on the Data Lake. Otherwise, the Data Lake rotate database certificate operation will fail with a warning message.

You can also use the following CLI commands to perform the certificate rotation with minimal downtime and without stopping the cluster. The same roles are required as for the Rotate Database Certificate action through the UI.

1. Use the following command to rotate the certificate for each Data Hub cluster.

```
cdp datahub rotate-db-certificate --datahubName <VALUE CAN BE cluster CRN
 or name>
```

2. Use the following command to rotate the certificate for the Data Lake.

```
cdp datalake rotate-db-certificate --datalakeName <VALUE CAN BE cluster
CRN or name>
```

Note: If you are performing the certificate rotation using the Rotate Database Certificate button on the UI that was made available to you through entitlement or if you are using the CLI commands, it is possible that a rolling restart will not happen because it is not supported on your cluster. In such cases, a regular restart action will be evoked. In this case, minimal downtime can be expected, however, your cluster will not be stopped.

Overriding default root certificate

CDP automatically chooses the target root certificate for the RDS instance. Currently, the following certificate authority (CA) certificates are available for the RDS database:

- rds-ca-rsa2048-g1, which is valid for 40 years
- rds-ca-rsa4096-g1, which is valid for 100 years
- rds-ca-ecc384-g1, which is valid for 100 years

By default, rds-ca-rsa2048-g1 is the root certificate in all supported regions that is picked as a target certificate during the certificate rotation. You have the option to override the default certificate, and set one of the certificates available in your region as a target root certificate that will be picked during the certificate rotation. Overriding the default root certificate is not tied to any particular RDS instance, but applicable to the whole AWS region in the target AWS account.

The list of available root certificates in your region can be retrieved with the following AWS CLI command:

```
aws --region [*** REGION ***] rds describe-certificates
```

The command returns the output JSON containing the properties of the RDS root certificates that are currently available in the given region. The properties also detail if an override is already set for the root certificate.

If there are more than one entries, you can use the following command to set one of the root certificates as a target, and override what is configured as default:

```
aws --region [*** REGION ***] rds modify-certificates --certificate-ident
ifier [*** ROOT CERTIFICATE ***]
```

The following command is an example to override the default rds-ca-rsa2048-g1 certificaate in the us-west-1 to rds-ca-ecc384-g1:

```
aws --region us-west-1 rds modify-certificates --certificate-identifier rds-
ca-ecc384-g1
```

Note: You can also set up the override for the default root certificate. This ensures that the root certificate will remain the same in case the default certificate is changed by AWS. You can use the following command to remove the override:

```
aws --region [*** REGION ***] rds modify-certificates --remove-custom
er-override
```

The command returns the root certificate that will be used by default.

Related Information

Describing certificates | AWS CLI

Modifying certificates | AWS CLI

Rotating database certificates when SSL enforcement is disabled

If the Data Lake for your environment or your Cloudera Data Hub cluster is using an RDS where SSL enforcement is disabled, no action is required on your side. You can simply let the root certificate expire and be replaced by AWS upon expiry.

It is recommended to check SSL enforcement setting on the Data Lake of your environment and separately for each Cloudera Data Hub cluster.

A Data Lake or a Cloudera Data Hub using an RDS that is shown as SSL Disabled are essentially immune to the validity of the RDS root certificate. This is because DB connections made from Cloudera cluster services do not explicitly validate the certificate chain received from the RDS instance in such cases. The AWS RDS instance may, therefore, be left as is safely, letting its root certificate expire and be replaced automatically by AWS upon the expiry date.

Alternatively, you can instead opt to change the RDS root certificate manually using standard AWS tools like the AWS RDS Console or the AWS CLI, as described in Updating your CA certificate by modifying your DB instance or cluster in the AWS documentation.

When manually rotating the root certificate, you have the option to choose from the following available RDS certificates based on your region:

- rds-ca-rsa2048-g1, which is valid for 40 years
- rds-ca-rsa4096-g1, which is valid for 100 years
- rds-ca-ecc384-g1, which is valid for 100 years

Cloudera does not provide automation for the rotation of the RDS root certificate for databases where SSL enforcement is disabled.

Note: The Cloudera Management Console does not reflect the change of the RDS root certificate after any manual rotation.

Managing public and private certificates

There are two types of certificates within Cloudera that you must manage: public and private, also called host certificates.

- Public certificates are Let's Encrypt-issued certificates for Cloudera Data Hub and Data Lake clusters. These certificates are available on port 443 (HTTPS) of the cluster and are responsible for enabling TLS in front of Knox and other available services on that port. They are valid for 90 days, and in most circumstances Cloudera will renew these certificates automatically before they expire.

  Note the following limitations in regards to automatic renewal of public certificates:

  - Cloudera Data Hub or Data Lake clusters created on or after March 7, 2022 are eligible for automatic renewal of public certificates. Clusters created before March 7, 2022, must be renewed manually once following the instructions in *Manually renewing public certificates for Data Lake and Cloudera Data Hub clusters*. After the public certificate for a cluster has been manually renewed once from the Cloudera UI or CLI, it is eligible for automatic certificate renewal in the future.
  - If an automatic renewal fails, the renewal service will retry the renewal for three consecutive days or three attempts. Any cluster that cannot be renewed by these retry attempts must be renewed manually through the Cloudera UI or CLI.
  - The automatic renewal is tried three times: on the 69th, 72nd and 78th day after the certificate creation date. For example, if a certificate is getting expired on September 24th, 2022, the renewal will be tried in the following sequence:

    - First renewal: September 3rd, 2022 2:00 A.M.
    - Second renewal: September 6th, 2022 2:00 A.M.
    - Third renewal: September 12th, 2022 2:00 A.M.

    In case the renewal is successful on the first attempt, the renewal will not be tried again.
  - Renewal of the certificates happens at 2 A.M. of the Cloudera Control Plane time. If the Cloudera Control Plane is in the United States region, the renewal starts at 2 A.M. Pacific Daylight Time (PDT). If the Cloudera Control Plane is in the European region, the renewal starts at 2 A.M. Central European Summer Time (CEST). If the Cloudera Control Plane is in the East-Asian and Pacific region, the renewal starts at 2 A.M. Australian Eastern Standard Time (AEST).
  - The auto renewal service does not know the status of the cluster. If the cluster is down or performing another operation, the automatic renewal may fail and you should initiate the renewal from the UI or CLI manually. Certificate renewal will not happen if the Cloudera Data Hub and Data Lake clusters or the Public Cloud environment has a Stopped state.
  - If the cluster is down during the renewal attempts and comes back up after the renewal retries are exhausted, automatic renewal will not happen for that cluster. The certificate has to be renewed manually from the UI or CLI.
  - If a public certificate expires, you'll receive a warning that your connection is not secure when you attempt to access a Data Lake or Cloudera Data Hub cluster through the Cloudera UI.

  See *Manually renewing public certificates for Data Lake and Cloudera Data Hub clusters* for instructions on renewing the public certificates manually.

- Private certificates, or host certificates, are certificates created during cluster provisioning for every host with Auto-TLS. Private/host certificates have a default expiration date of one year. As private certificates get closer to expiration, the Cloudera UI displays a warning that the certificate is about to expire.

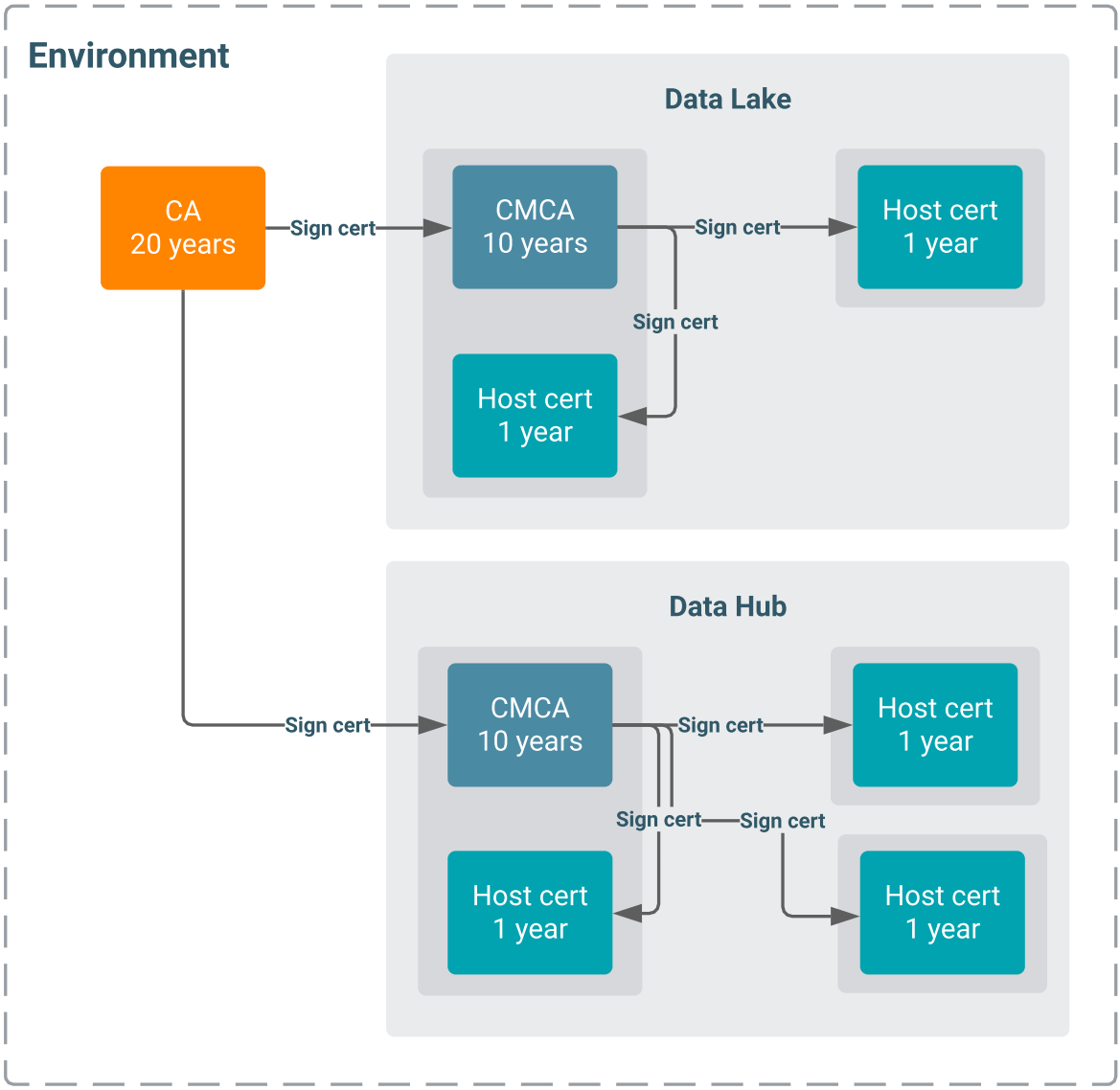

Though the Cloudera UI displays a warning about the expiration of private/host certificates, you are still responsible for renewing them through the UI or CDP CLI. After the certificates expire, the cluster is not functional, so you must renew them before expiration.

Renewing private/host certificates on Data Lake and Cloudera Data Hub clusters

Private (host) certificates have a default expiration date of one year; to keep the Data Lake and Cloudera Data Hub clusters running, you must renew the host certificates before they expire.

About this task

Required role (Data Lakes): EnvironmentAdmin or Owner of the environment

Required role (Cloudera Data Hub): DatahubAdmin, Owner of the Data Hub, EnvironmentAdmin, or Owner of the environment

During cluster provisioning, Cloudera Manager creates an intermediate certificate (CMCA) signed by FreeIPA CA. The CMCA is used to create certificates for every host with Auto-TLS.

There are two ways to renew a private/host certificate. To renew the private/host certificates at any time, use the following CLI commands:

Data Lake certificate renewal:

```
cdp datalake rotate-private-certificates --datalake <Data Lake name or CRN>
```

Cloudera Data Hub certificate renewal:

```
cdp datahub rotate-private-certificates --datahub <Data Hub name or CRN>
```

Alternatively, you can wait until the host certificate is close to expiration. During periodic cluster state synchronization, Cloudera uses the Cloudera Manager API to check that the HOST_AGENT_CERTIFICATE_EXPIRY apiHealthCheck alert is in a GOOD state. If the apiHealthCheck is not in a GOOD state, Cloudera displays a warning in the UI.

These UI warnings will display on the associated Environments, Data Lakes, or Cloudera Data Hub clusters list and details pages. For example:

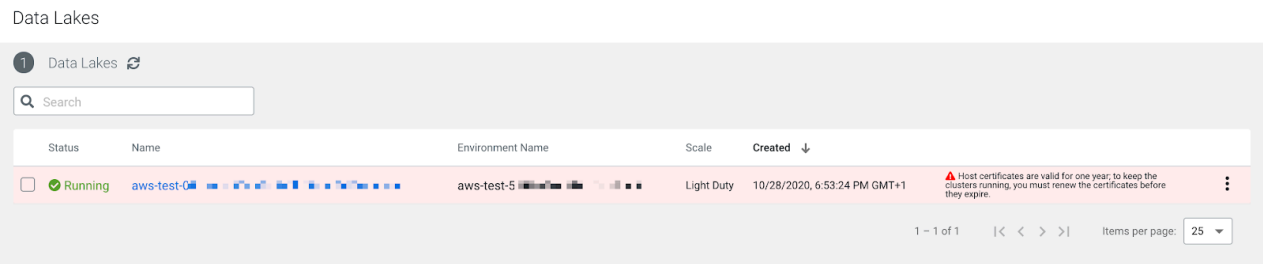

To renew the host certificate once you receive an expiration warning, follow the steps below.

Note: Renewing host certificates causes some downtime because all services are restarted. The Cloudera Manager UI will be inaccessible for some time because the Cloudera Manager server is also restarted.

Procedure

1. On the Environments, Data Lakes, or Cloudera Data Hub clusters list pages, click the three vertical dots next to the expiration message.
2. Click Renew Host Certificates or Renew Data Lake Host Certificates.

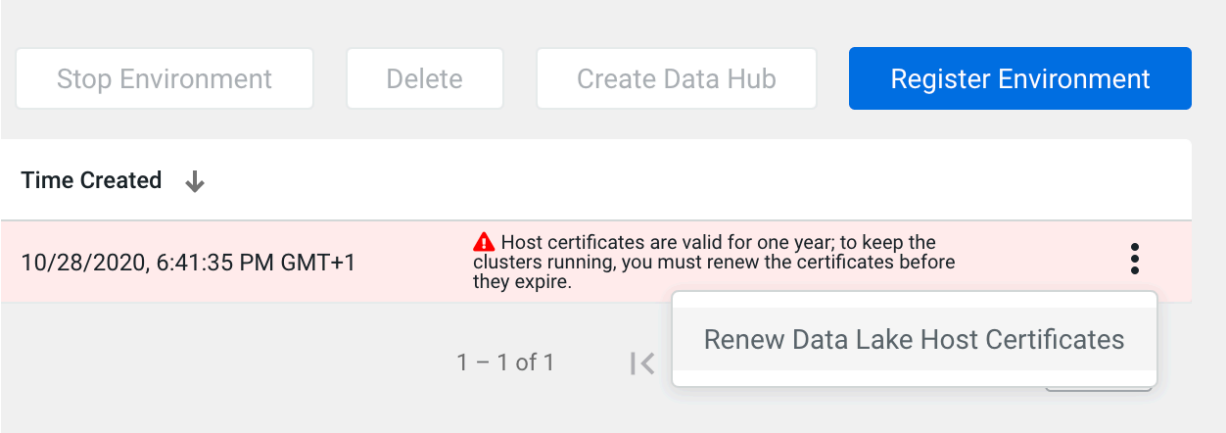

3. Click Yes when you are asked if you want to renew the certificates.

Manually renewing public certificates for Data Lake and Cloudera Data Hub clusters

Public certificates are responsible for enabling TLS in front of Knox and other available services on port 443 of Data Lake and Cloudera Data Hub clusters. Public certificates expire every 90 days and are often automatically renewed by Cloudera. If automatic renewal fails, you can renew these certificates manually.

Required role (Data Lakes): EnvironmentAdmin or Owner of the environment

Required role (Cloudera Data Hub): DatahubAdmin, Owner of the Data Hub, EnvironmentAdmin, or Owner of the environment

To renew a public certificate, click the Renew Public Certificate button on the details page of a chosen Data Lake:

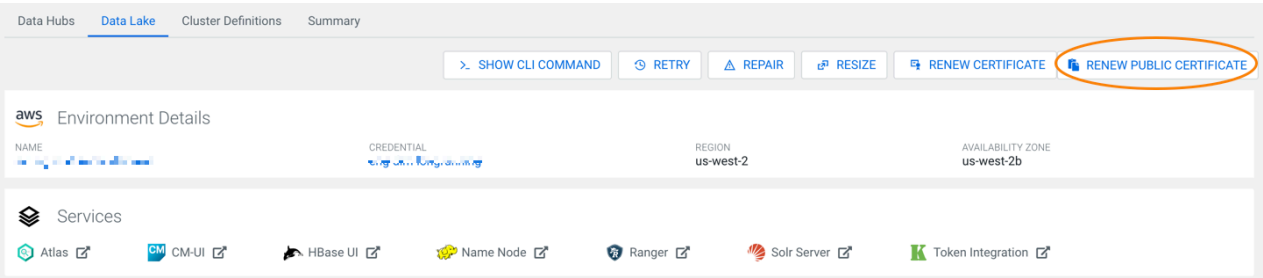

Or from the Actions menu of a Cloudera Data Hub cluster details page:

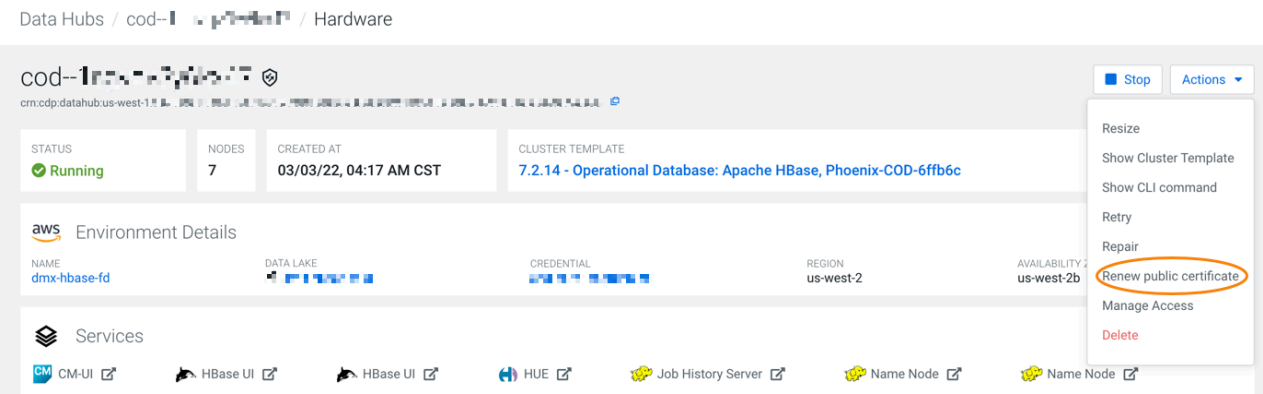

Triggering the certificate renewal may cause a minor cluster downtime of a few seconds. The entire renewal process takes a few minutes.

If you prefer to renew the certificates using the CLI, use the following commands:

Data Lake public certificate renewal:

```
cdp datalake renew-public-certificate --datalake <Data Lake name or CRN>
```

Cloudera Data Hub public certificate renewal:

```
cdp datahub renew-public-certificate --datahub <Data Hub name or CRN>
```

More cluster management options

You can access more cluster management options by logging in to Cloudera Manager.

For Cloudera Manager documentation, refer to the link below.

Related Information

Managing Cloudera Data Hub Clusters

Managing clusters from CLI

You can manage Cloudera Data Hub clusters from the CLI using the cdp datahub commands.

Required role: The DataHubAdmin or Owner roles at the scope of Cloudera Data Hub allow you to manage the Cloudera Data Hub cluster. Note that EnvironmentAdmin and Owner of the environment can also manage Cloudera Data Hub clusters.

Monitoring clusters

- List all available clusters: cdp datahub list-clusters
- Describe a specific cluster: cdp datahub describe-cluster --cluster-name <value>
- Get status of cluster hosts: cdp datahub get-cluster-host-status --cluster-name <value>
- Get status of cluster services: cdp datahub get-cluster-service-status --cluster-name <value>

Managing clusters

- When stack provisioning or cluster creation fails, retry the cluster to resume the process from the last failed step: cdp datahub retry-cluster --cluster-name <value>
- Resize cluster: cdp datahub scale-cluster --cluster-name <value> --instance-group-name <value> --instance-group-desired-count <value>
- Restart a cluster that is in stopped state: cdp datahub start-cluster --cluster-name <value>
- Stop cluster that is in running state: cdp datahub stop-cluster --cluster-name <value>
- Sync your cluster with cloud provider and Cloudera Manager: cdp datahub sync-cluster --cluster-name <value>
- Repair a cluster when cluster has failed nodes: cdp datahub repair-cluster --cluster-name <value> --instance-group-names <value> --instances <value>

Deleting clusters

- Delete one or more existing clusters: cdp datahub delete-clusters --cluster-name <value>

Cloudera Data Hub Overview

Date published: 2019-12-17
Date modified: 2023-06-27

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Data Hub overview....4

Core concepts.... 5
Workload clusters.... 5
Cluster definitions....5
Cluster templates....6
Recipes.... 6
Custom properties.... 7
Image catalogs.... 7

Default cluster configurations.... 7
Data Engineering clusters....8
Data Mart clusters....16
Operational Database clusters.... 17
Streams Messaging clusters....18
Flow Management clusters....19
Streaming Analytics clusters.... 21
Data Discovery and Exploration clusters....22

Default cluster topology.... 23

Node repair....23

Cloud storage....24

Databases for cluster components....25

Cloudera Data Hub overview

Cloudera Data Hub is a service for launching and managing workload clusters powered by Cloudera Runtime (Cloudera's unified open source distribution including the best of CDH and HDP). Cloudera Data Hub clusters can be created on AWS, Microsoft Azure, and Google Cloud Platform.

https://www.youtube.com/embed/cBGxwVIoXk4

Cloudera Data Hub includes a set of cloud optimized built-in templates for common workload types, as well as a set of options allowing for extensive customization based on your enterprise's needs. Furthermore, it offers a set of convenient cluster management options such as cluster scaling, stop, restart, terminate, and more. All clusters are secured via wire encryption and strong authentication out of the box, and users can access cluster UIs and endpoints through a secure gateway powered by Apache Knox. Access to S3 cloud storage from Cloudera Data Hub clusters is enabled by default (S3Guard is enabled and required in Cloudera Runtime versions older than 7.2.2).

Cloudera Data Hub provides complete workload isolation and full elasticity so that every workload, every application, or every department can have their own cluster with a different version of the software, different configuration, and running on different infrastructure. This enables a more agile development process.

Since Cloudera Data Hub clusters are easy to launch and their lifecycle can be automated, you can create them on demand and when you don't need them, you can return the resources to the cloud.

The following diagram describes a simplified Cloudera Data Hub architecture:

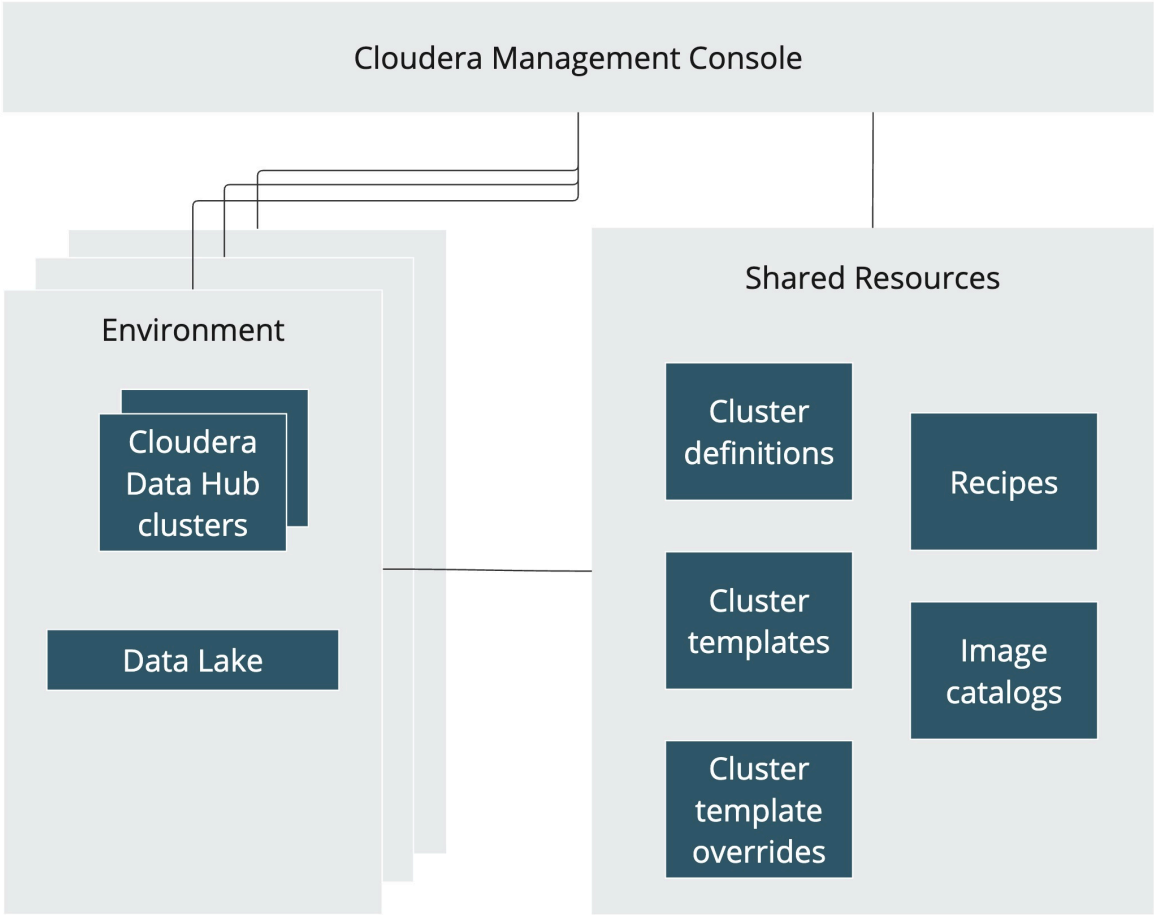

Data Hub clusters can be launched, managed, and accessed from the Cloudera Management Console. All Cloudera Data Hub clusters are attached to a Data Lake that runs within an environment and provides security and governance for the environment's clusters.

Cloudera Data Hub provides a set of shared resources and allows you to register your own resources that can be reused between multiple Cloudera Data Hub clusters. As illustrated in the following diagram, these resources (cluster definitions, cluster templates, cluster template overrides, recipes, and image catalogs) can be managed in the Cloudera Management Console and shared between multiple Cloudera Data Hub clusters:

- Default Cluster definitions (with cloud provider specific settings) and cluster templates (with Cloudera Runtime service configurations) allow you to quickly provision workload clusters for prescriptive use cases. You can also save your own cluster definitions and templates for future reuse.
- With cluster template overrides, you can easily specify custom configurations that override or append the properties in a built-in Cloudera Data Hub template or a custom template.
- You can create and run your own pre- and post-deployment and startup scripts (called recipes) to simplify installation of third party components required by your enterprise security guidelines.
- Cloudera Data Hub comes with a default image catalog that includes a set of prewarmed images (including Cloudera Manager and Cloudera Runtime). You can also customize default images and create custom image catalogs.

All of this functionality is available via the Cloudera web interface (as part of the Cloudera Management Console service) and CDP CLI. While the Cloudera web interface allows you to get started quickly, the CLI allows you to create reusable scripts to automate cluster creation and cluster lifecycle management.

Related Information

Azure Load Balancers in Data Lakes and Cloudera Data Hub clusters

Cloudera Management Console Overview

Core concepts

The following concepts are key to understanding the Cloudera Data Hub:

Workload clusters

All Cloudera Data Hub clusters are workload clusters. These clusters are created for running specific workloads such as data engineering or data analytics.

Cloudera Data Hub clusters are powered by Cloudera Runtime. They can be ephemeral or long-running. Once a Cloudera Data Hub cluster is created it can be managed by using the Cloudera Management Console and Cloudera Manager.

Each Cloudera Data Hub cluster is created within an existing environment and is attached to a data lake, which provides the security and governance layer.

Cluster definitions

Cloudera Data Hub provides a set of default cluster definitions for prescriptive use cases and allows you to create your own custom cluster definitions.

A cluster definition is a reusable cluster template in JSON format that can be used for creating multiple Cloudera Data Hub clusters with identical cloud provider settings.

Cloudera Data Hub includes a set of default cluster definitions for common data analytics and data engineering use cases, and allows you to save your own custom cluster definitions. While default cluster definitions can be used across all environments, custom cluster definitions are associated with one or more specific environments. Cluster definitions facilitate repeated cluster reuse and are therefore useful for creating automation around cluster creation.

The easiest way to create a custom cluster definition is to start with an existing cluster definition, modify it in the create cluster wizard, and then save it. Or you can open the custom create cluster wizard, provide all the parameters, and then save the cluster definition.

Note:

A cluster definition is not synonymous with a cluster template. While a cluster template primarily defines Cloudera Runtime services, a cluster definition primarily includes cloud provider settings. Furthermore, a cluster definition always references one specific cluster template.

Related Information

Cluster Definitions

Cluster templates

Cloudera Data Hub uses cluster templates for defining cluster topology: defining host groups and components installed on each host group.

A cluster template is a reusable template in JSON format that can be used for creating multiple Cloudera Data Hub clusters with identical Cloudera Runtime settings. It primarily defines the list of host groups and how components of various Cloudera Runtime services are distributed on these host groups. A cluster template allows you to specify stack, component layout, and configurations to materialize a cluster instance via Cloudera Manager REST API, without having to use the Cloudera Manager install wizard. After you provide the cluster template to Cloudera Data Hub, the host groups in the JSON are mapped to a set of instances when starting the cluster, and the specified services and components are installed on the corresponding nodes.

Note:

A cluster template is not synonymous with a cluster definition, which primarily defines cloud provider settings. Each cluster definition must reference a specific cluster template.

Cloudera Data Hub includes a few default cluster templates and allows you to upload your own cluster templates. Custom cluster templates can be uploaded and managed via the Cloudera web interface or CLI and then selected, when needed, for a specific cluster.

Related Information

Cluster Templates

Recipes

A recipe is a script that runs on all nodes of a selected host group at a specific time. You can use recipes to create and run scripts that perform specific tasks on your Cloudera Data Hub cluster nodes.

You can use recipes for tasks such as installing additional software or performing advanced cluster configuration. For example, you can use a recipe to put a JAR file on the Hadoop classpath.

Recipes can be uploaded and managed via the Cloudera web interface or CLI and then selected, when needed, for a specific cluster and for a specific host group. If selected, they will be executed on a specific host group at a specified time.

Depending on the need, a recipe can be executed at various times. Available recipe execution times are:

- Before Cloudera Manager server start
- After Cloudera Manager server start
- After cluster installation
- Before cluster termination

Related Information

Recipes

Custom properties

Custom properties are configuration properties that can be set on a Cloudera Runtime cluster, but Cloudera Data Hub allows you to conveniently set these during cluster creation.

When working with on premise clusters, you would typically set configuration properties once your cluster is already running by using Cloudera Manager or manually in a specific configuration file.

When creating a Cloudera Data Hub cluster in Cloudera, you can specify a list of configuration properties that should be set during cluster creation and Cloudera Data Hub sets these for you.

Note:

Alternatively, it is also possible to set configuration properties by using Cloudera Manager once the cluster is already running.

Related Information

Custom Properties

Image catalogs

Cloudera Data Hub provides a set of default, prewarmed images. You can also customize the default images.

Cloudera Data Hub provides a set of default prewarmed images. The prewarmed images include Cloudera Manager and Cloudera Runtime packages. Default images are available for each supported cloud provider and region as part of the default image catalog.

The following table lists the default images available:

Cloud provider	Operating system	Image type	Description
AWS	CentOS 7	Prewarmed	Includes Cloudera Manager and Cloudera Runtime packages.
Azure	CentOS 7	Prewarmed	Includes Cloudera Manager and Cloudera Runtime packages.
Google Cloud	CentOS 7	Prewarmed	Includes Cloudera Managerand Cloudera Runtime packages.

Cloudera does not support base images, but does support customizing the default images. You might need to customize an image for compliance or security reasons.

Default cluster configurations

Cloudera Data Hub includes a set of prescriptive cluster configurations. Each of these default cluster configurations include a cloud-provider specific cluster definition, which primarily defines cloud provider settings. The cluster definition references a cluster template, which defines a number of Cloudera Runtime or Cloudera DataFlow components used for common data analytics and data engineering use cases.

Refer to the topic for each default cluster configuration to view the included services and compatible Runtime versions. These topics include links to documentation that will help you to understand the included components and use the workload cluster.

Many of the cluster components are included in the Cloudera Runtime software distribution. The Streams Messaging, Cloudera Flow Management, and Cloudera Streaming Analytics cluster configurations are part of Cloudera DataFlow for Data Hub and have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for more details.

You can access the default cluster definitions by clicking Environments, then selecting an environment and clicking the Cluster Definitions tab.

You can access the default cluster templates from Shared ResourcesCluster Templates.

To view details of a cluster definition or cluster template, click on its name. For each cluster definition, you can access a raw JSON file. For each cluster template, you can access a graphical representation ("list view") and a raw JSON file ("raw view") of all cluster host groups and their components.

Related Information

Cloudera DataFlow for Data Hub

Cloudera Runtime

Data Engineering clusters

Learn about the default Data Engineering clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Engineering provides a complete data processing solution, powered by Apache Spark and Apache Hive. Spark and Hive enable fast, scalable, fault-tolerant data engineering and analytics over petabytes of data.

Data Engineering cluster definition

This Data Engineering template includes a standalone deployment of Spark and Hive, as well as Apache Oozie for job scheduling and orchestration, Apache Livy for remote job submission, and Hue and Apache Zeppelin for job authoring and interactive analysis.

Cluster definition names

- Data Engineering for AWS
- Data Engineering for Azure
- Data Engineering for Google Cloud
- Data Engineering HA - Spark3 for AWS

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering HA - Spark3 for Azure

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering HA - Spark3 for Google Cloud
- Data Engineering Spark3 for AWS
- Data Engineering Spark3 for Azure
- Data Engineering Spark3 for Google Cloud

Cluster template name

- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: Apache Spark, Apache Hive, Apache Oozie."
- Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: HA: Apache Spark, Apache Hive, Apache Oozie."

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: The "Data Engineering: Apache Spark3" cluster template is deleted. Therefore, the "Data Engineering: Apache Spark3, Apache Hive, Apache Oozie" cluster template can be used instead.

Included services

- HDFS
- Hive
- Hue
- Livy
- Spark 3
- Yarn
- Zeppelin
- ZooKeeper
- Oozie is supported for Spark 3 as of Cloudera Runtime version 7.2.18
- Hive Warehouse Connector is supported as of Cloudera Runtime version 7.2.16.

Compatible Cloudera Runtime version

7.2.16, 7.2.17, 7.2.18

Topology of the Data Engineering cluster

Topology is a set of host groups that are defined in the cluster template and cluster definition used by Data Engineering. Data Engineering uses the following topology:

Host group	Description	Node configuration
Master	The master host group runs the components for managing the cluster resources including Cloudera Manager, Name Node, Resource Manager, as well as other master components such HiveServer2, HMS, Hue etc.	1 For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.4xlarge; gp2 - 100 GB Azure : Standard_D16_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-16; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE, DE Spark3, and DE HA: AWS : m5.4xlarge; gp2 - 100 GB Azure: Standard_D16_v3 GCP : e2-standard-16; pd-ssd - 100 GB
Worker	The worker host group runs the components that are used for executing processing tasks (such as NodeManager) and handling storing data in HDFS such as DataNode.	3 For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.2xlarge; gp2 - 100 GB Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-8; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+

Host group	Description	Node configuration
		DE and DE Spark3: AWS: r5d.2xlarge - (gp2/EBS volumes) Azure: Standard_D5_v2 GCP : e2-standard-8; pd-ssd - 100 GB DE HA: AWS: r5d.4xlarge - (gp2/EBS volumes) Azure: Standard_D5_v2 GCP : e2-standard-8; pd-ssd - 100 GB
Compute	The compute host group can optionally be used for running data processing tasks (such as NodeManager). By default the number of compute nodes is set to 1 for proper configurations of YARN containers. This node group can be scaled down to 0 when there are no compute needs. Additionally, if load-based auto-scaling is enabled with minimum count set to 0, the compute nodegroup will be resized to 0 automatically.	0+ For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.2xlarge; gp2 - 100 GB Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-8; pd-ssd - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: AWS: r5d.2xlarge - (ephemeral volumes) Azure: Standard_D5_v2 For Azure, the attached volume count for the compute host group is changed to 0. Only ephemeral/local volumes are used by default. GCP : e2-standard-8; pd-ssd - 100 GB DE HA: AWS: r5d.4xlarge - (ephemeral volumes) Azure: Standard_D5_v2 For Azure, the attached volume count for the compute host group is changed to 0. Only ephemeral/local volumes are used by default. GCP : e2-standard-8; pd-ssd - 100 GB

Host group	Description	Node configuration
		Note: Compute nodes run YARN and require storage only for temporary data - this requirement is fulfilled by instance storage, so making the attached volumes count to 0 by default is more cost-efficient.
Gateway	The gateway host group can optionally be used for connecting to the cluster endpoints like Oozie, Beeline etc. This nodegroup does not run any critical services. This nodegroup resides in the same subnet as the rest of the nodegroups. If additional software binaries are required they could be installed using recipes.	0+ AWS : m5.2xlarge; gp2 - 100 GB Azure : Standard_D8_v3; StandardSSD_LRS - 100 GB GCP : e2-standard-8; pd-ssd - 100 GB

Service configurations			
Master host group Cloudera Manager, HDFS, Hive (on Tez), HMS, Yarn RM, Oozie, Hue, DAS, Zookeeper, Livy, Zeppelin and Sqoop	Gateway host group Configurations for the services on the master node	Worker host group Data Node and YARN NodeManager	Compute group YARN NodeManager

Configurations

Note the following:

- There is a Hive Metastore Service (HMS) running in the cluster that talks to the same database instance as the Data Lake in the environment.
- If you use CLI to create the cluster, you can optionally pass an argument to create an external database for the cluster use such as Cloudera Manager, Oozie, Hue, and DAS. This database is by default embedded in the master node external volume. If you specify the external database to be of type HA or NON_HA, the database will be provisioned in the cloud provider. For all these types of databases the lifecycle is still associated with the cluster, so upon deletion of the cluster, the database will also be deleted.
- The HDFS in this cluster is for storing the intermediary processing data. For resiliency, store the data in the cloud object stores.
- For high availability requirements choose the Data Engineering High Availability cluster shape.

Architecture of the Data Engineering HA for AWS cluster

The Data Engineering HA for AWS and Azure cluster shape provides failure resilience for several of the Data Engineering HA services, including Knox, Oozie, HDFS, HS2, Hue, Livy, YARN, and HMS.

Services that do not yet run in HA mode include Cloudera Manager, DAS, and Zeppelin.

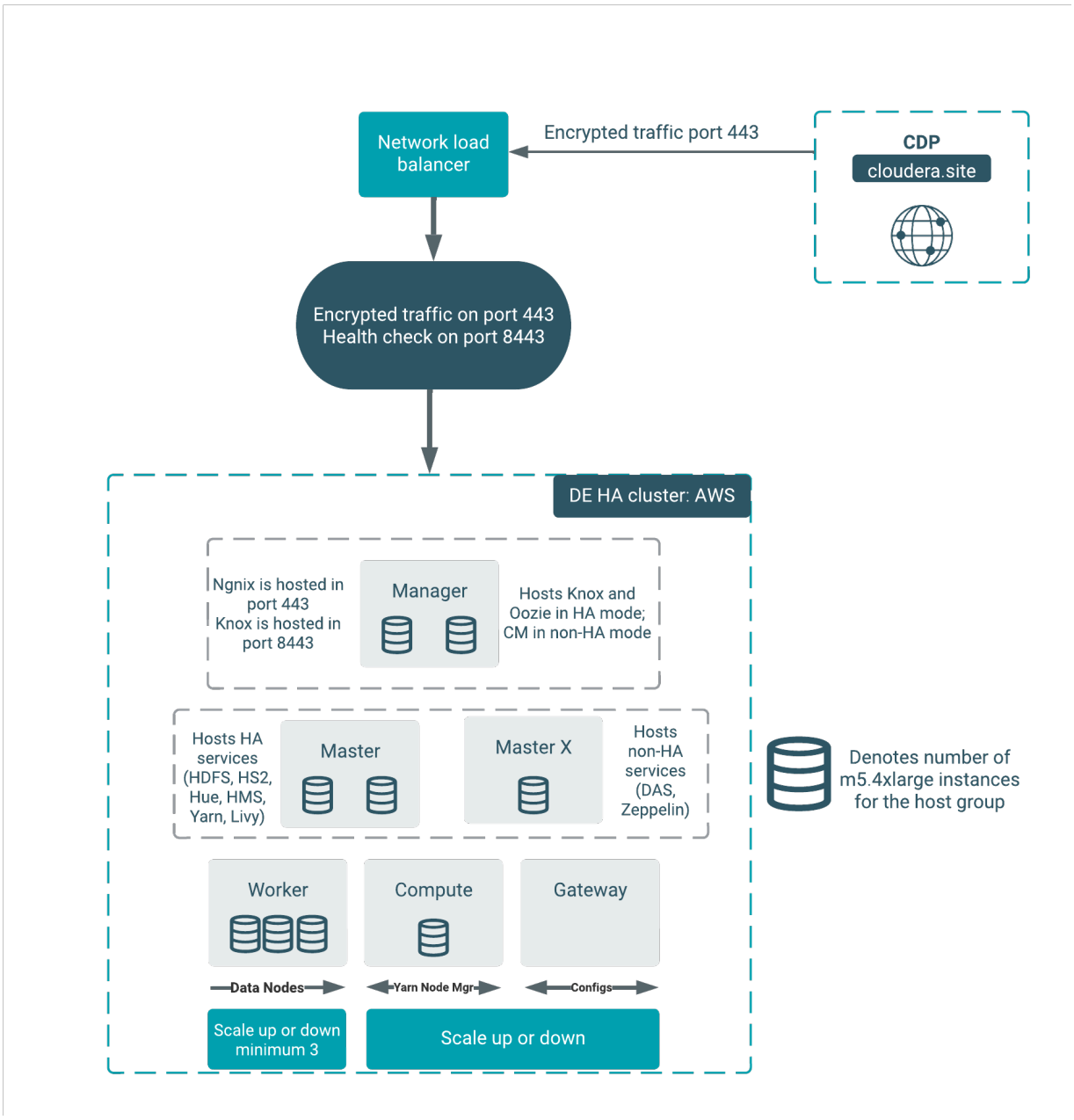

The architecture outlined in the diagram above handles the failure of one node in all of the host groups except for the “masterx” group. See the table below for additional details about the component interactions in failure mode:

Component	Failure	User experience
Knox	One of the Knox services is down	External users will still be able to access all of the UIs, APIs, and JDBC.
Cloudera Manager	The first node in manager host group is down	The cluster operations (such as repair, scaling, and upgrade) will not work.
Cloudera Manager	The second node in the manager host group is down	No impact.
HMS	One of the HMS services is down	No impact.
Hue	One of the Hue services is down in master host group	No impact.

HS2	One of the HS2 services is down in the master host group	External users will still be able to access the Hive service via JDBC. But if Hue was accessing that particular service it will not failover to the other host. The quick fix for Hue is to restart Hue to be able to use Hive functionality.
YARN	One of the YARN services is down	No impact.
HDFS	One of the HDFS services is down	No impact.
Nginx	Nginx in one of the manager hosts is down	Fifty percent of the UI, API, and JDBC calls will be affected. If the entire manager node is down, there is no impact. This is caused by the process of forwarding and health checking that is done by the network load-balancer.
Oozie	One of the Oozie servers is down in the manager host group.	No impact for AWS and Azure as of Cloudera Runtime version 7.2.11. If you create a custom template for DE HA, follow these two rules: **1.** Oozie must be in single hostgroup. **2.** Oozie and Hue must not be in the same hostgroup.

Important: If you are creating a DE HA cluster through the CDP CLI using the create-aws-cluster command, note that there is a CLI parameter to provision the network load-balancer in HA cluster shapes. Make sure to use the [--enable-load-balancer | --no-enable-load-balancer] parameter when provisioning a DE HA cluster via the CLI. For more information see the CDP CLI reference.

Architecture of the Data Engineering HA for Azure cluster

The Data Engineering HA for Azure cluster shape provides failure resilience for several of the Data Engineering HA services, including Knox, Oozie, HDFS, HS2, Hue, Livy, YARN, and HMS.

Services that do not yet run in HA mode include Cloudera Manager, DAS, and Zeppelin.

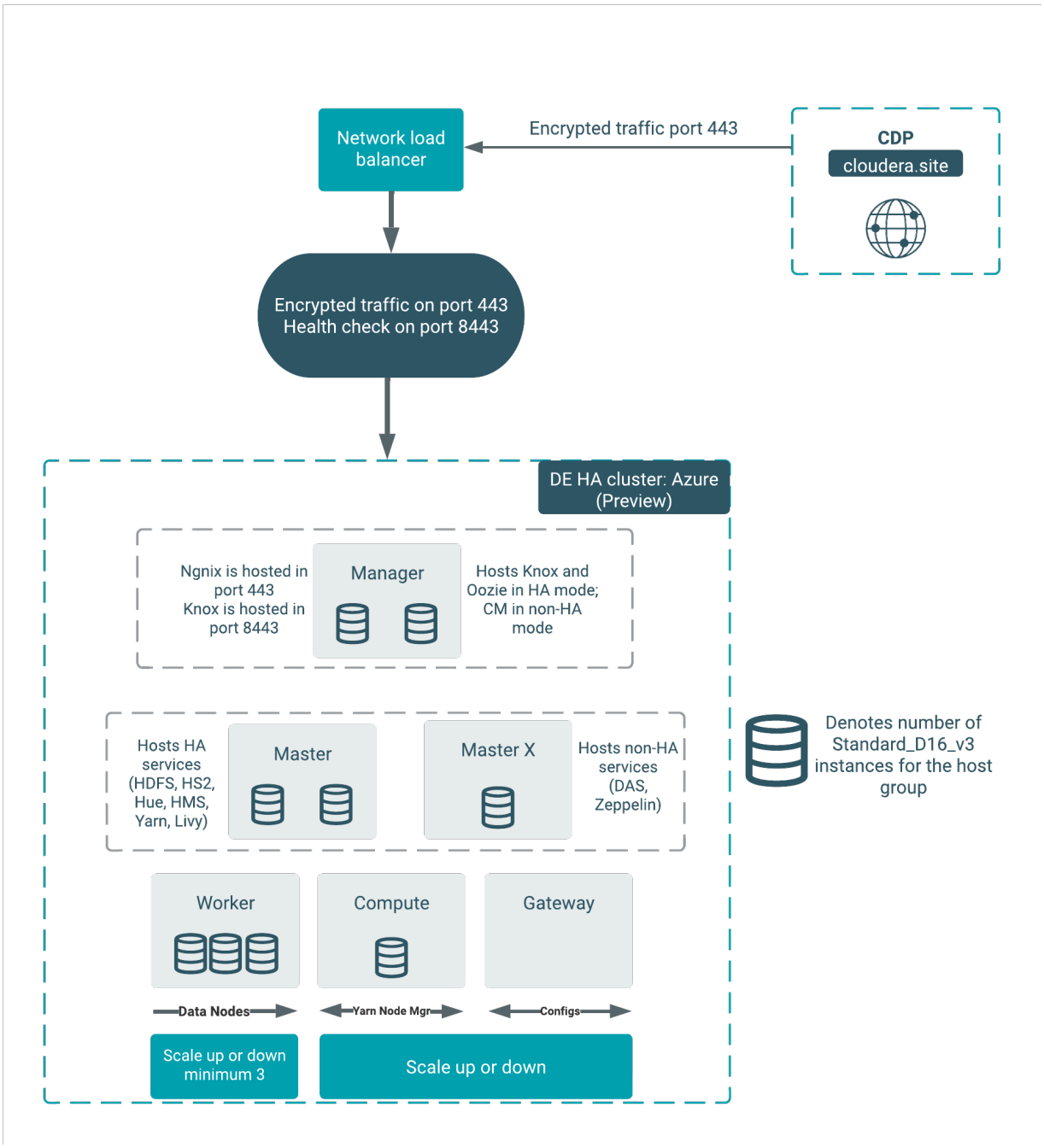

Component	Failure	User experience
Knox	One of the Knox services is down	External users will still be able to access all of the UIs, APIs, and JDBC.
Cloudera Manager	The first node in manager host group is down	The cluster operations (such as repair, scaling, and upgrade) will not work.
Cloudera Manager	The second node in the manager host group is down	No impact.
HMS	One of the HMS services is down	No impact.
Hue	One of the Hue services is down in master host group	No impact.

HS2	One of the HS2 services is down in the master host group	External users will still be able to access the Hive service via JDBC. But if Hue was accessing that particular service it will not failover to the other host. The quick fix for Hue is to restart Hue to be able to use Hive functionality.
YARN	One of the YARN services is down	No impact.
HDFS	One of the HDFS services is down	No impact.
Nginx	Nginx in one of the manager hosts is down	Fifty percent of the UI, API, and JDBC calls will be affected. If the entire manager node is down, there is no impact. This is caused by the process of forwarding and health checking that is done by the network load-balancer.
Oozie	One of the Oozie servers is down in the manager host group.	No impact for AWS and Azure as of Cloudera Runtime version 7.2.11. If you create a custom template for DE HA, follow these two rules: **1.** Oozie must be in single hostgroup. **2.** Oozie and Hue must not be in the same hostgroup.

Important: If you are creating a DE HA cluster through the CDP CLI using the create-azure-cluster command, note that there is a CLI parameter to provision the network load-balancer in HA cluster shapes. Make sure to use the [--enable-load-balancer | --no-enable-load-balancer] parameter when provisioning a DE HA cluster via the CLI. For more information see the CDP CLI reference.

GCP HA (Preview)

Note: HA for Oozie is not yet available in the GCP template.

Custom templates

Any custom DE HA template that you create must be forked from the default templates of the corresponding version. You must create a custom cluster definition for this with the JSON parameter “enableLoadBalancers”: true , using the create-aws/azure/gcp-cluster CLI command parameter --request-template. Support for pre-existing custom cluster definitions will be added in a future release. As with the template, the custom cluster definition must be forked from the default cluster definition. You are allowed to modify the instance types and disks in the custom cluster definition. You must not change the placement of the services like Cloudera Manager, Oozie, and Hue. Currently the custom template is fully supported only via CLI.

The simplest way to change the DE HA definition is to create a custom cluster definition. In the Create Data Hub UI when you click Advanced Options, the default definition is not used fully, which will cause issues in the HA setup.

Related Information

HDFS
Hive
Hue
Oozie
Spark
YARN
Zookeeper

Data Mart clusters

Learn about the default Data Mart and Real Time Data Mart clusters, including cluster definition and template names, included services, and compatible Runtime version.

Data Mart is an MPP SQL database powered by Apache Impala designed to support custom Data Mart applications at big data scale. Impala easily scales to petabytes of data, processes tables with trillions of rows, and allows users to store, browse, query, and explore their data in an interactive way.

Data Mart clusters

The Data Mart template provides a ready to use, fully capable, standalone deployment of Impala. Upon deployment, it can be used as a standalone Data Mart to which users point their BI dashboards using JDBC/ODBC end points. Users can also choose to author SQL queries in Cloudera's web-based SQL query editor, Hue, and run them with Impala providing a delightful end-user focused and interactive SQL/BI experience.

Cluster definition names

- Data Mart for AWS
- Data Mart for Azure
- Data Mart for Google Cloud

Cluster template name

Cloudera - Data Mart: Apache Impala, Hue

Included services

- HDFS
- Hue
- Impala

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17, 7.2.18

Real Time Data Mart clusters

The Real-Time Data Mart template provides a ready-to-use, fully capable, standalone deployment of Impala and Kudu. You can use a Real Time Data Mart cluster as a standalone Data Mart which allows high throughput streaming ingest, supporting updates and deletes as well as inserts. You can immediately query data through BI dashboards using JDBC/ODBC end points. You can choose to author SQL queries in Cloudera's web-based SQL query editor, Hue. Executing queries with Impala, you will enjoy an end-user focused and interactive SQL/BI experience. This template is commonly used for Operational Reporting, Time Series, and other real time analytics use cases.

Cluster definition names

- Real-time Data Mart for AWS
- Real-time Data Mart for Azure
- Real-time Data Mart for Google Cloud

Cluster template name

Cloudera - Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 2

- Yarn

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster definition names

- Real-time Data Mart - Spark3 for AWS
- Real-time Data Mart - Spark3 for Azure
- Real-time Data Mart - Spark3 for Google Cloud

Cluster template name

Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark3

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 3
- Yarn

Compatible Cloudera Runtime versions

7.2.16, 7.2.17, 7.2.18

High availability

Cloudera recommends that you use high availability (HA), and track any services that are not capable of restarting or performing failover in some way.

Impala HA

The Impala nodes offer high availability. The following Impala services are not HA.

- Catalog service
- Statestore service

Kudu HA

Both Kudu Masters and TabletServers offer high availability.

Related Information

HDFS
Hue
Impala
Kudu
Spark
YARN

Operational Database clusters

The Operational Database (OpDB) template is removed from the Cloudera Data Hub. You can access the Cloudera Operational Database instead as a superior product.

The Cloudera Operational Database is a NoSQL database powered by Apache HBase designed to support custom OLTP applications that want to leverage the power of BigData. Apache HBase is a NoSQL, scale-out database that can easily scale to petabytes and stores tables with millions of columns and billions of rows.

Cloudera Operational Database also contains Apache Phoenix which provides a way to use HBase through an SQL interface.

Cloudera recommends you to use the Cloudera Operational Database to create Operational Database clusters.

Related Information

Cloudera Operational Database
Getting started with Operational Database
Before you create an Operational Database cluster
Creating an Operational Database cluster
HDFS
HBase
Knox
Zookeeper
Phoenix

Streams Messaging clusters

Learn about the default Streams Messaging clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streams Messaging provides the following features:

- Advanced messaging and real-time processing on streaming data using Apache Kafka
- Centralized schema management using Schema Registry
- Management and monitoring capabilities powered by Streams Messaging Manager
- Cross-cluster Kafka topic replication using Streams Replication Manger
- Kafka partition rebalancing with Cruise Control

This template sets up a fault-tolerant standalone deployment of Apache Kafka and supporting Cloudera components (Schema Registry, Streams Messaging Manager, Streams Replication Manager and Cruise Control), which can be used for production Kafka workloads in the cloud or as a disaster recovery instance for on-premises. Kafka clusters.

Note:

Streams Messaging clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streams Messaging cluster deployment
- Creating your first Streams Messaging cluster
- Connecting Kafka clients to Cloudera Public Cloud clusters

Cluster definition names

- Streams Messaging Heavy Duty for AWS
- Streams Messaging Light Duty for AWS
- Streams Messaging HA for AWS
- Streams Messaging Heavy Duty for Azure
- Streams Messaging Light Duty for Azure
- Streams Messaging HA for Azure (Technical Preview)
- Streams Messaging Heavy Duty for GCP
- Streams Messaging Light Duty for GCP
- Streams Messaging HA for GCP (Technical Preview)

Cluster template name

- Cloudera - Streams Messaging Heavy Duty
- Cloudera - Streams Messaging Light Duty

- Cloudera - Streams Messaging High Availability

Included services

- Kafka
- Schema Registry
- Streams Messaging Manager
- Streams Replication Manager
- Cruise Control
- Kafka Connect

Compatible Cloudera Runtime version

- 7.1.0 (Preview)
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streams Messaging cluster
Ingesting Data into Cloudera Public Cloud
Kafka
Schema Registry
Streams Messaging Manager
Streams Replication Manager

Flow Management clusters

Learn about the default Flow Management clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime versions.

Flow Management delivers high-scale data ingestion, transformation, and management to enterprises from any-to-any environment. It addresses key enterprise use cases such as data movement, continuous data ingestion, log data ingestion, and acquisition of all types of streaming data including social, mobile, clickstream, and IoT data.

The Flow Management template includes a no-code data ingestion and management solution powered by Apache NiFi. With NiFi's intuitive graphical interface and 300+ processors, Flow Management enables easy data ingestion and movement between CDP services as well as 3rd party cloud services. NiFi Registry is automatically set up and provides a central place to manage versioned Data Flows.

Note:

Flow Management clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Flow Management cluster deployment
- Creating your first Flow Management cluster
- Security considerations for Flow Management clusters
- Using Apache NiFi to ingest data into Cloudera Public Cloud
- Using NiFi and NiFi Registry

Cluster definition names

- Flow Management Light Duty for AWS
- Flow Management Light Duty for Azure
- Flow Management Light Duty for GCP
- Flow Management Heavy Duty for AWS
- Flow Management Heavy Duty for Azure
- Flow Management Heavy Duty for GCP

Cluster template name

- CDP - Flow Management: Light Duty
- CDP - Flow Management: Heavy Duty

Included services

- NiFi
- NiFI Registry

Compatible Cloudera Runtime versions

- 7.1.0
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Flow Management cluster
Apache NiFi documentation
Apache NiFi Registry documentation

Streaming Analytics clusters

Learn about the default Streaming Analytics clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streaming Analytics offers real-time stream processing and stream analytics with low-latency and high scaling capabilities powered by Apache Flink.

Streaming Analytics templates include Apache Flink that works out of the box in stateless or heavy state environments. Beside Flink, the template includes its supporting services namely YARN, Zookeeper and HDFS. The Heavy Duty template comes preconfigured with RocksDB as state backend, while Light Duty clusters use the default Heap state backend. You can create your streaming application by choosing between Kafka, Kudu, and HBase as datastream connectors.

You can also use SQL to query real-time data with SQL Stream Builder (SSB) in the Streaming Analytics template. By supporting the SSB service in Cloudera Public Cloud, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. SSB is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

Note:

Streaming Analytics clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streaming Analytics cluster deployment
- Creating your first Streaming Analytics cluster
- Analyzing data using Apache Flink
- Querying data using SQL Stream Builder

Cluster definition names

- Streaming Analytics Light Duty for AWS
- Streaming Analytics Light Duty for Azure
- Streaming Analytics Light Duty for GCP
- Streaming Analytics Heavy Duty for AWS
- Streaming Analytics Heavy Duty for Azure
- Streaming Analytics Heavy Duty for GCP

Cluster template name

- 7.2.17 - Streaming Analytics Light Duty
- 7.2.17 - Streaming Analytics Heavy Duty

Included services

- Flink
- SQL Stream Builder
- YARN
- Zookeeper
- HDFS
- Kafka

Important: In the Streaming Analytics cluster templates, Kafka service is included by default to serve as a background service only for the websocket ouput and sampling feature of SQL Stream Builder. The Kafka service in the Streaming Analytics cluster template cannot be used for production, you need to use the Streams Messaging cluster template when Kafka is needed for your deployment.

Compatible Cloudera Runtime version

- 7.2.2

- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streaming Analytics cluster
Flink
YARN
Zookeeper
HDFS

Data Discovery and Exploration clusters

Learn about the default Data Discovery and Exploration clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Discovery and Exploration

Explore and discover data sets ad-hoc. Do relevance-based analytics over unstructured data (logs, images, text, PDFs, etc). Get started with search or log analytics. Make data more accessible to everyone with Data Discovery and Exploration.

Cluster Definition Names

- Data Discovery and Exploration for AWS
- Data Discovery and Exploration for Azure

Cluster Template Name

- Data Discovery and Exploration

Included Services

- Solr
- Spark 2
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Versions

7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster Definition Names

- Data Discovery and Exploration - Spark3 for AWS
- Data Discovery and Exploration - Spark3 for Azure
- Data Discovery and Exploration - Spark3 for Google Cloud

Cluster Template Name

- Data Discovery and Exploration for Spark3

Included Services

- Solr
- Spark 3
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Version

7.2.18

Related Information

Solr
Spark
HDFS
Hue
YARN
Zookeeper

Default cluster topology

Cloudera Data Hub uses a specific cluster topology including the following host groups: master, worker, and compute.

Cloudera Data Hub uses the following host groups. These host groups are defined in cluster templates and cluster definitions used by Cloudera Data Hub:

Host group	Description	Number of nodes
Master	The master host group runs the components for managing the cluster resources (including Cloudera Manager), storing intermediate data (e.g. HDFS), processing tasks, as well as other master components.	1
Worker	The worker host group runs the components that are used for executing processing tasks (such as NodeManager) and handling storing data in HDFS such as DataNode).	1+
Compute	The compute host group can optionally be used for running data processing tasks (such as NodeManager).	0+

Node repair

Cloudera Data Hub monitors clusters, ensuring that when host-level failures occur, they are reported right away and can be quickly resolved by performing manual repair which deletes and replaces failed nodes and reattaches the disks.

For each Cloudera Data Hub cluster, Cloudera checks for Cloudera Manager agent heartbeat on all cluster nodes. If the Cloudera Manager agent heartbeat is lost on a node, a failure is reported for that node. This may happen for the following reasons:

- The Cloudera Manager agent process exited on a node
- An instance crashed
- An instance was terminated

Once a failure is reported, options are available for you to repair the failure manually.

Important: A manual repair operation on an unhealthy node will replace the disk mounted as root, which means all data on the root disk is lost.

To avoid the loss of important data, do not store it in the root disk. Instead, do one of the following:

- Store data that needs to persist beyond the lifetime of the cluster in S3 or ADLS Gen 2, depending on the platform in use.
- Store data that needs to survive a repair operation in /hadoopfs/fsN/ (where N is an integer), as long as it is not so large that it could crowd out components that use that location.

  For example, storing 1 GB of data in /hadoopfs/fs1/save_me would be an option to ensure that the data is available in a replacement node after a manual repair operation.

Manual repair is enabled for all clusters by default and covers all nodes, including the Cloudera Manager server node (by default, Cloudera Manager is installed on the master node). When a node fails, a notification about node failure is printed in the Event History, the affected node is marked as unhealthy, and an option to repair the cluster is available from the Actions menu. From the Hardware tab you can choose to repair a single node or specific nodes within a host group. There are two ways to repair the cluster:

- Repair the failed nodes: (1) All non-ephemeral disks are detached from the failed nodes. (2) Failed nodes are removed (3) New nodes of the same type are provisioned. (4) The disks are attached to the new volumes, preserving the data.
- Delete the failed nodes: Failed nodes are deleted with their attached volumes.

Note:

As of March 26, 2021, the manual repair option is available for all clusters and typically covers all nodes, including the master node, regardless of whether or not the cluster uses an external database.

For clusters created before March 26, 2021, manual repair covers all nodes except the master node (Cloudera Manager server node). There is an exception: if a cluster (created before March 26, 2021) uses an external database, then manual repair is enabled for the master node. Ephemeral disks cannot be reattached (they are deleted and new disks are provisioned).

Cloud storage

The options on the "Cloud Storage" page allow you to optionally specify the base storage location used for YARN and Zeppelin.

During environment creation under Data Access > Storage Location Base you configure a default S3, ADLS Gen2, or Google Cloud Storage base storage location for the environment, and all Data Hub clusters created within that environment use this location. The Cloud Storage options in the Cloudera Data Hub cluster wizard allow you to additionally specify a different location for YARN application logs and Zeppelin Notebook's root directory:

- Existing Base Storage Location - By default, this is set to the Storage Location Base configured on environment level. If you do not want to make any changes, simply leave this blank. If you would like to use a different location for YARN application logs and Zeppelin Notebook's root directory, you can specify a different S3 or ADLS Gen2 location. Note that the specified S3, ADLS Gen2, or Google Cloud Storage location must exist prior to Data Hub cluster creation and that you must adjust the IAM policies created during environment's cloud storage setup to make sure that IDBroker has write access to this location.
- Path for YARN Application Logs property - This directory structure gets created automatically during cluster creation. You can customize it if you would like it be different than what is suggested by default.
- Path for Zeppelin Notebooks Root Directory property - This directory structure gets created automatically during cluster creation. You can customize it if you would like it to be different than what is suggested by default.

Note: Any S3 bucket that you designate for Cloudera Data Hub cloud storage on AWS must be in the same region as the environment.

Databases for cluster components

By default, Cloudera Data Hub provisions embedded databases for all cluster services that require it. These databases run on the Cloudera Manager server node (By default Cloudera Manager is installed on the master node).

Note:

One exception is Hive Metastore, which always uses the external database attached to the data lake. Therefore, no embedded database is needed for Hive Metastore.

Cloudera Data Hub

Recipes

Date published: 2019-12-17
Date modified: 2023-06-27

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Recipes 4

Writing recipes 4

Recipe and cluster template parameters 6

Example: Recipe with parameters 9

Register a recipe 9

Update a recipe 11

Managing recipes from CLI 12

Recipes

A recipe is a script that runs on all nodes of a selected host group at a specific time. You can use recipes to create and run scripts that perform specific tasks on your Cloudera Data Hub, Data Lake, or FreeIPA cluster nodes.

You can use recipes for tasks such as installing additional software or performing advanced cluster configuration. For example, you can use a recipe to put a JAR file on the Hadoop classpath.

Recipes can be uploaded and managed via the Cloudera web interface or CLI and then selected, when needed, for a specific cluster and for a specific host group. If selected, they will be executed on a specific host group at a specified time.

Recipes are stored on the Cloudera Manager server for the lifetime of the master node, and are executed at specific times of your choosing:

- pre-service-deployment (formerly pre-cloudera-manager-start and pre-cluster-manager-start): During a Data Hub, Data Lake, or environment deployment, the script will be executed on every node before the CM server starts, and after node repair or OS upgrade of a cluster.
- post-cloudera-manager-start: During a Cloudera Data Hub or Data Lake deployment, the script will be executed on every node after the Cloudera Manager server starts, but before cluster installation. post-cluster-manager start recipes are also executed after node repair or OS upgrade of a cluster. This option is not available for FreeIPA recipes.
- post-service-deployment (formerly post-cluster-install): The script will be executed on every node after cluster installation on the CM server is finished.
- pre-termination: The script will be executed on every node before cluster termination.

Note: On the master node, recipes are triggered when the Cloudera Manager server starts; on other nodes, recipes are triggered when the Cloudera Manager agent starts.

Writing recipes

Refer to these guidelines when creating your recipes.

When using recipes, consider the following guidelines:

- Running bash and python scripts as recipes is supported. We recommend using scripts with Shebang character sequence, for example:

```
#!/bin/sh
#!/bin/bash
#!/usr/bin/sh
#!/usr/bin/bash
#!/usr/bin/env sh
#!/usr/bin/env bash
#!/bin/sh -x
#!/usr/bin/python
#!/usr/bin/env python
```

- The scripts are executed as root. The recipe output is written to /var/log/recipes on each node on which it was executed.
- Supported parameters can be specified as variables by using mustache kind of templating with "{{{ }}}" syntax. Once specified in a recipe, these variables are dynamically replaced when the recipe is executed, generating the

actual values that you provided as part of cluster creation process. For the list of parameters, refer to Recipe and cluster template parameters. For an example, see Example: Recipe using parameters.

Note: Using variable parameters is not supported for FreeIPA recipes.

For example, if your cluster includes an external LDAP and your recipe includes {{{ldap.connectionURL}}}, as demonstrated in the following example

```
#!/bin/bash -e

main() {
 ping {{{ ldap.connectionURL }}}
}
[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

then, when this recipe runs, the {{{ldap.connectionURL}}} is replaced with the actual connection URL specified as part of cluster creation process, as demonstrated in the following example:

```
#!/bin/bash -e

main() {
 ping 192.168.59.103
}
[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

- Recipe logs can be found at /var/log/recipes/${RECIPE_TYPE}/${RECIPE_NAME}.log
- The scripts are executed on all nodes of the host groups that you select (such as “master”, “worker”, “compute”).
- In order to be executed, your script must be in a network location which is accessible from the Cloudera Management Console and the virtual network in which your cluster is located.
- Make sure to follow Linux best practices when creating your scripts. For example, don’t forget to script “Yes” auto-answers where needed.
- Do not execute yum update –y as it may update other components on the instances (such as salt) – which can create unintended or unstable behavior.

Example Python script

```
#!/usr/bin/python
print("An example of a python script")
import sys
print(sys.version_info)
```

Example bash script for yum proxy settings

```
#!/bin/bash
cat >> /etc/yum.conf
<<ENDOF
proxy=http://10.0.0.133:3128
ENDOF
```

Example recipe including variables

Original recipe:

```
#!/bin/bash -e

function setupAtlasServer() {
```

```
    curl -iv -u {{{ general.userName }}}:{{{ general.password }}} -H "X-Re
quested-By: ambari" -X POST -d '{"RequestInfo":{"command":"RESTART","context
":"Restart all components required ATLAS","operation_level":{"level":"SERVIC
E","cluster_name":"{{{ general.clusterName }}}","service_name":"ATLAS"}},"Re
quests/resource_filters":[{"hosts_predicate":"HostRoles/stale_configs=false&
HostRoles/cluster_name={{{ general.clusterName }}}"}]}' http://$(hostname -f
):8080/api/v1/clusters/{{{ general.clusterName }}}/requests
}

main() {
    setupAtlasServer
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Generated recipe (to illustrate how the variables from the original recipe were replaced during cluster creation):

```
#!/bin/bash -e

function setupAtlasServer() {
    curl -iv -u admin:admin123 -H "X-Requested-By: ambari" -X POST -d '{"R
equestInfo":{"command":"RESTART","context":"Restart all components required
ATLAS","operation_level":{"level":"SERVICE","cluster_name":"super-cluster","
service_name":"ATLAS"}},"Requests/resource_filters":[{"hosts_predicate":"Hos
tRoles/stale_configs=false&HostRoles/cluster_name=super-cluster"}]}' http://
$(hostname -f):8080/api/v1/clusters/super-cluster/requests
}

main() {
    setupAtlasServer
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Recipe and cluster template parameters

The following supported parameters can be specified as variables/dynamic parameters in recipes or cluster templates by using mustache formatting with "{{{ }}}" syntax.

Note: Using variable parameters is not supported for FreeIPA recipes.

The parameter keys listed below follow the following general conventions:

- { } indicates that the parameter key has multiple supported values, which are provided in this documentation. For example {fileSystemType} can be one of the following: s3, adls_gen_2, or wasb.
- [index] indicates that the parameter includes an index value for example sharedService.datalakeComponents.[index] can be "sharedService.datalakeComponents.[0]", "sharedService.datalakeComponents.[1]", and so on. There is no easy way to find out what the index will be, but you may still be able to use these parameters (for example by creating a condition to filter them).

For information on how to set these parameters dynamically in a cluster template, refer to Setting custom properties.

Custom properties

Any custom property specified in the cluster template can be used as a recipe parameter. Refer to Custom properties documentation.

General

The general parameter group includes parameters related to general cluster configuration.

Description	Example key	Example value
Name of stack	general.stackName	teststack
UUID of cluster	general.uuid	9aab7fdb-8940-454b-bc0a-62f04bce6519
Cloudera Manager user name	general.cmUserName	
Cloudera Manager password	general.cmPassword	
Cloudera Manager IP	general.clusterManagerIp	127.0.0.1
Number of nodes	general.nodeCount	5
FQDN of primary gateway instance	general.primaryGatewayInstanceDiscoveryFQDN	ip-10-0-88-28.example.com
Number indicating the Kafka replication factor (3 or 1)	general.kafkaReplicationFactor	1

Blueprint

The blueprint parameter group includes parameters related to cluster template configuration.

Parameter key	Description	Example key	Example value
blueprint.blueprintText	Blueprint text in JSON format	blueprint.blueprintText	
blueprint.version	Version of blueprint	blueprint.version	7.2.8

Cloud storage

The fileSystemConfigs parameter group includes parameters related to cloud storage configuration.

When forming the parameter keys, the {fileSystemType} should be replaced with an actual cloud storage type such as "s3", "adls_gen_2", or "wasb".

Parameter key	Description	Example key	Example value
File system common configurations			
fileSystemConfigs.{fileSystemType}.storageContainer	Name of container in Azure storage account (Cloudbreak + stackId)	fileSystemConfigs.s3.storageContainer	cloudbreak123
fileSystemConfigs.{fileSystemType}.type	Type of filesystem	fileSystemConfigs.s3.type	S3
fileSystemConfigs.{fileSystemType}.locations.[index].configFile	Configuration file used to configure the filesystem	fileSystemConfigs.s3.locations.[0].configFile	hbase-site
fileSystemConfigs.{fileSystemType}.locations.[index].property	Property key of filesystem path in defined config	fileSystemConfigs.s3.locations.[0].property	hbase.rootdir
fileSystemConfigs.{fileSystemType}.locations.[index].value	Value of filesystem path in defined config	fileSystemConfigs.s3.locations.[0].value	s3a://testranger/testrecipe2/apps/hbase/data
Amazon S3 configurations			
fileSystemConfigs.s3.storageContainer	Generated name (cloudbreak + stack id number)	fileSystemConfigs.s3.storageContainer	cloudbreak7941
fileSystemConfigs.s3.locations.[index].configFile	Hadoop component configuration file	fileSystemConfigs.s3.locations.[0].configFile	zeppelin-site
fileSystemConfigs.s3.locations.[index].property	Component property name	fileSystemConfigs.s3.locations.[0].property	zeppelin.notebook.dir

Parameter key	Description	Example key	Example value
fileSystemConfigs.s3.locations.[index].value	Component property value	fileSystemConfigs.s3.locations.[0].value	s3a://storagename/clustername/zeppelin/notebook
ADLS Gen2 configurations			
fileSystemConfigs.adls_gen_2.accountName	Name of the corresponding Azure storage account	fileSystemConfigs.adls_gen_2.accountName	teststorageaccount
fileSystemConfigs.adls_gen_2.storageContainerName	Container name in Azure storage account	fileSystemConfigs.adls_gen_2.storageContainerName	testcontainer

External database

The rds parameter group includes parameters related to external database configuration.

When forming the parameter keys, the {rdsType} should be replaced with the actual database type such as "cloudera_manager", "beacon", "druid", "hive", "oozie", "ranger", "superset", or some other user-defined type.

Parameter key	Description	Example key	Example value
rds.{rdsType}.connectionURL	JDBC connection URL	rds.hive.connectionURL	Value is specified in the following format: jdbc:postgresql://host:port/database
rds.{rdsType}.connectionDriver	JDBC driver used for connection	rds.hive.connectionDriver	org.postgresql.Driver
rds.{rdsType}.connectionUserName	Username used for the JDBC connection	rds.hive.connectionUserName	testuser
rds.{rdsType}.connectionPassword	Password used for the JDBC connection	rds.hive.connectionPassword	TestPssword123
rds.{rdsType}.databaseName	Target database of the JDBC connection	rds.hive.databaseName	myhivedb
rds.{rdsType}.host	Host of the JDBC connection	rds.hive.host	mydbhost
rds.{rdsType}.hostWithPortWithJdbc	Host of JDBC connection with port and JDBC prefix	rds.hive.hostWithPortWithJdbc	Value is specified in the following format: jdbc:postgresql://host:port
rds.{rdsType}.subprotocol	Sub-protocol from the JDBC URL	rds.hive.subprotocol	postgresql
rds.{rdsType}.connectionString	URL of JDBC the connection. In case of Ranger, this does not contain the port	rds.hive.connectionString	Value is specified in the following format: jdbc:postgresql://host:port/database
rds.{rdsType}.databaseVendor	Database vendor	rds.hive.databaseVendor	POSTGRES
rds.{rdsType}.withoutJDBCPrefix	URL of the JDBC connection without JDBC prefix	rds.hive.withoutJDBCPrefix	Value is specified in the following format: host:port/database

Gateway

The gateway parameter group includes parameters related to Knox gateway configuration.

Parameter key	Description	Example key	Example value
gateway.ssoProvider	Path to the SSO provider	gateway.ssoProvider	/test/sso/api/v1/websso
gateway.signKey	Base64 encoded signing key	gateway.signKey	
gateway.signPub	Signing certificate (x509 format)	gateway.signPub	
gateway.signCert	Public SSH key used for signing (standard public key format)	gateway.signCert	

Shared services

The sharedService parameter group includes parameters related to Data Lake configuration.

Parameter key	Description	Example key	Example value
sharedService.rangerAdminPassword	Admin password of the Ranger component	sharedService.rangerAdminPassword	Admin1234!
sharedService.datalakeCluster	Flag indicating that the cluster is a data lake cluster	sharedService.datalakeCluster	true
sharedService.datalakeClusterManagerIp	Cloudera Manager IP of data lake cluster	sharedService.datalakeClusterManagerIp	127.0.0.1
sharedService.datalakeClusterManagerFqdn	Cloudera Manager FQDN of data lake cluster (or the IP if FQDN cannot be found)	sharedService.datalakeClusterManagerFqdn	ip-10-88-28.example.com

Related Information

Example: Recipe with parameters

Example: Recipe with parameters

If you pass the supported parameters in a recipe, their values are dynamically fetched and replaced.

Note: Using variable parameters is not supported for FreeIPA recipes.

Example recipe template (the {{{general.clusterName}}} is included as a template):

```
#!/bin/bash -e

function setupDefaultClusterFolder() {
  mkdir -p /var/log/{{{general.clusterName}}}
}

main() {
    setupDefaultClusterFolder
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Example recipe after {{{general.clusterName}}} is set to my-super-cluster based on the actual cluster name:

```
#!/bin/bash -e

function setupDefaultClusterFolder() {
  mkdir -p /var/log/my-super-cluster
}

main() {
    setupDefaultClusterFolder
}
[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Related Information

Recipe and cluster template parameters

Register a recipe

In order to use your recipe for clusters, you must first register it with the Cloudera Management Console.

About this task

Required role: EnvironmentCreator can create a shared resource and then assign users to it.

SharedResourceUser or Owner of the shared resource can use the resource.

Before you begin

If you are using Cloudera with a proxy, note that the Cloudera proxy settings do not apply to cluster recipes. If you planning to use the recipes, then you can set the proxy settings manually. You can find the proxy settings in the /etc/cdp/proxy.env file.

Procedure

1. Place your script in a network location accessible from Cloudera Management Console and from the virtual network in which your clusters are located.
2. Log in to the Cloudera web interface.
3. Navigate to Shared ResourcesRecipes and click Register Recipe.
4. Provide the following:

Parameter	Value
Name	Enter a name for your recipe.
Description	(Optional) Enter a description for your recipe.
Execution Type	Select one of the following options: • pre-service-deployment (formerly pre-cluster-manager-start): During a Cloudera Data Hub, Data Lake, or environment deployment, the script will be executed on every node (in the host group where you assigned the recipe) before the Cloudera Manager server starts. • post-cluster-manager-start: During a Cloudera Data Hub or Data Lake deployment, the script will be executed on every node (in the host group where you assigned the recipe) after the Cloudera Manager server starts, but before cluster installation. This option is not available for FreeIPA recipes. • post-service-deployment (formerly post-cluster-install):: The script will be executed on every node (in the host group where you assigned the recipe) after cluster installation on the Cloudera Manager server is finished. • pre-termination: The script will be executed on every node (in the host group where you assigned the recipe) before cluster termination.
Script	Select one of: • File: Point to a file on your machine that contains the recipe. • Text: Paste the script.

5. Click Register.

What to do next

- When you create a Cloudera Data Hub cluster, you can select a previously added recipe on the advanced Cluster Extensions page of the create cluster wizard.
- When you create an environment, you can select a previously added recipe on the Data Access and Data Lake Scaling page of the environment creation wizard, under Advanced Options > Cluster Extensions > Recipes.
- When you create an environment, you can select a previously added FreeIPA recipe on the **Region, Networking, and Security** page of the environment creation wizard, under Advanced OptionsCluster ExtensionsRecipes.
- You can also attach recipes to Cloudera Data Hub or Data Lake clusters when you create an environment/Data Lake or Cloudera Data Hub cluster through the CDP CLI.

Update a recipe

You can attach or detach recipes to/from existing Cloudera Data Hub clusters. Using this capability, you can update a recipe by removing it from the cluster, replacing the old recipe with a modified recipe of the same type, and attaching the new modified recipe to the cluster.

About this task

Attaching or detaching a recipe will not execute the recipe. The next execution of the recipe will take place based on the type of the recipe. After an upscale, a newly attached recipe runs only on the new hosts.

Required role (one of the following):

- PowerUser on Cloudera tenant
- Owner of the environment
- EnvironmentAdmin
- Owner of the Cloudera Data Hub cluster
- DataHubAdmin

Procedure

1. Create a new recipe (with updated/modified content) of the same type as the old recipe that you want to replace.
2. Click Data Hub Clusters <Cluster Name> Recipes and find the recipe that you want to remove in the list of recipes for the cluster.
3. Click Remove Recipe next to the name of the recipe that you want to remove, then click Yes in the confirmation window.

4. Once you have removed the old recipe, click on the Add Recipe button for the cluster and select the same host group that you previously used for the old recipe. Then select the name of the new recipe that contains the modified content and click Add.

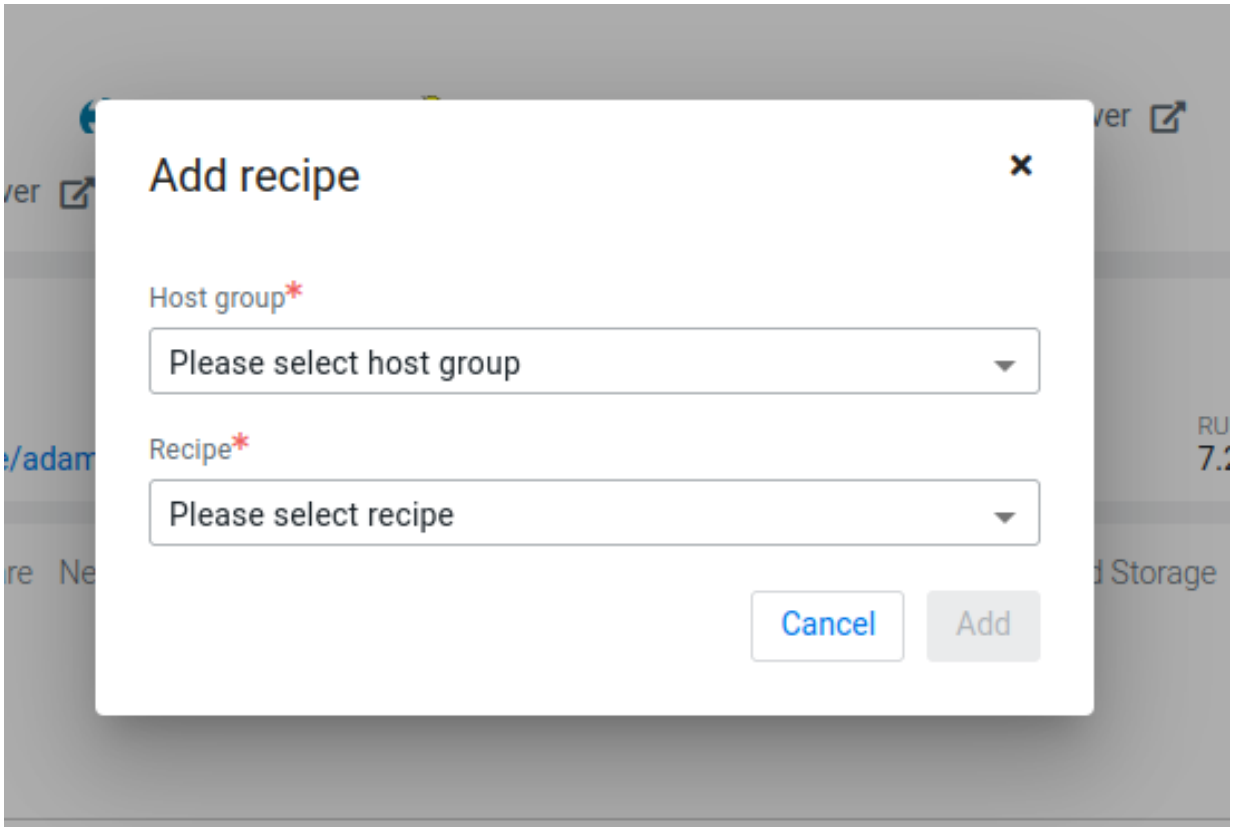

Results

You should see the new recipe appear for the same host group. After this change, the next recipe execution will execute the new script.

Managing recipes from CLI

You can manage recipes from CLI using cdp datahub commands.

Required role: EnvironmentCreator can create a shared resource and then assign users to it.

SharedResourceUser or Owner of the shared resource can use the resource. The Owner of the shared resource can delete it.

Note: Currently, recipes use cdp datahub commands regardless of whether the recipe is intended to run on Cloudera Data Hub, Data Lake, or FreeIPA cluster nodes.

- Register a new recipe: cdp datahub create-recipe --recipe-name <value> --recipe-content <value> --type <value>

  Supported types:

  - PRE_SERVICE_DEPLOYMENT (formerly PRE_CLOUDERA_MANAGER_START)
  - POST_CLOUDERA_MANAGER_START (this option is not available for FreeIPA recipes)
  - POST_SERVICE_DEPLOYMENT (formerly POST_CLUSTER_INSTALL)
  - PRE_TERMINATION
- List all available recipes: cdp datahub list-recipes
- Describe a specific recipe: cdp datahub describe-recipe --recipe-name <value>

- Delete one or more existing recipes: cdp datahub delete-recipes --recipe-name <value>

Cloudera Data Hub Release Notes

Date published: 2019-12-17
Date modified: 2023-06-27

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What's new **4**
January 30, 2025 4
Older releases 4
2024 4
2023 8
2022 9
2021 14
2020 16
2019 17

Log4j vulnerabilities **18**

Known issues **18**

Image catalog updates **19**

What's new

This section lists major features and updates for the Cloudera Data Hub service.

January 30, 2025

This release of the Cloudera Data Hub service introduces the following changes:

Added YARN recommendations to Cloudera Data Hub scaling activities CLI output

The list of Cloudera Data Hub scaling activities in the output of the cdp datahub list-scaling-activities command have been extended with YARN recommendations. For more information, see Managing autoscaling.

Older releases

Overview of new features, enhancements, and changed behavior introduced in earlier releases of the Cloudera Data Hub service.

2024

December 2024

This release of the Cloudera Data Hub service introduces the following changes:

Cloudera Runtime 7.3.1

Cloudera Runtime 7.3.1 is now available and can be used for registering an environment with a 7.3.1 Data Lake and creating Cloudera Data Hub clusters. For more information about the new Cloudera Runtime version, see Cloudera Runtime. If you need to upgrade your existing Cloudera environment, your upgrade path may be complex. To determine your upgrade path, refer to Upgrading to Runtime 7.3.1 documentation.

Added pd-extreme and pd-balanced as supported GCP disktypes

The following disk types have been added to supported GCP block storage types:

- pd-extreme
- pd-balanced

You can review all supported disk types for GCP at Supported GCP block storage.

Added EnvironmentPrivilegedUser to environment resource roles

The new EnvironmentPrivilegedUser resource role grants permission to execute privileged operating system actions on Data Lake, FreeIPA, and Cloudera Data Hub virtual machines.

For more information, see Resource roles.

October 2024

This release of the Cloudera Data Hub service introduces the following changes:

Azure Single Server to Azure Flexible Server upgrade

Single Server on Microsoft Azure databases used by Data Lakes and Cloudera Data Hub clusters can now be upgraded to Azure Flexible Server. During the upgrade process from PosgtreSQL version 11 to PostgreSQL 14,

Azure Single Server will be upgraded to Azure Flexible Server. For more information, see Upgrading Azure Single Server to Flexible Server.

July 2024

This release of the Cloudera Data Hub service introduces the following changes:

AWS RDS certificate rotation

AWS requires the rotation of the SSL/TLS certificates used for secure communication between Data Lakes and certain Cloudera Data Hub clusters and the external AWS RDS database instances that they rely on. Cloudera Public Cloud now provides multiple options to perform the required RDS certificate rotation.

For more information, see Rotating database certificates.

Rolling upgrade support for Streams Messaging clusters

Rolling upgrades for Streams Messaging Cloudera Data Hub clusters are now available. Rolling upgrades for Cloudera Data Hub clusters are limited to certain Data Hub Runtime versions. For more information, see Rolling upgrades.

May 2024

This release of the Cloudera Data Hub service introduces the following changes:

New default Azure VM instance types in cluster templates

In the built-in Cloudera Data Hub templates, the following instance types are being replaced:

- Standard_D5_v2 is being replaced with Standard_D16s_v3
- Standard_D8_v3 is being replaced with Standard_D8s_v3
- Standard_D16_v3 is being replaced with Standard_D16s_v3
- Standard_E16_v3 is being replaced with Standard_E16s_v3
- Standard_E8_v3 is being replaced with Standard_E8s_v3

The following table provides more detail:

Template name	Group	Previous instance type	New instance type
7.x - Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie for Azure	manager	Standard_D16_v3	Standard_D16s_v3
7.x - Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie for Azure	compute	Standard_D5_v2	Standard_D16s_v3
7.x - Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie for Azure	worker	Standard_D16_v3	Standard_D16s_v3
7.x - Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie for Azure	gateway	Standard_D16_v3	Standard_D16s_v3
7.x - Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie for Azure	master	Standard_D16_v3	Standard_D16s_v3
7.x - Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie for Azure	masterx	Standard_D16_v3	Standard_D16s_v3
7.x - Data Engineering Spark3 for Azure	master	Standard_D16_v3	Standard_D16s_v3
7.x - Data Engineering Spark3 for Azure	compute	Standard_D5_v2	Standard_D16s_v3

7.x - Data Engineering Spark3 for Azure	worker	Standard_D5_v2	Standard_D16s_v3
7.x - Data Engineering Spark3 for Azure	gateway	Standard_D8_v3	Standard_D16s_v3
7.x - Data Discovery and Exploration for Spark3 for Azure	master	Standard_D8_v3	Standard_D8s_v3
7.x - Data Discovery and Exploration for Spark3 for Azure	gateway	Standard_D8_v3	Standard_D8s_v3
7.x - Data Discovery and Exploration for Spark3 for Azure	leader	Standard_D8_v3	Standard_D8s_v3
7.x - Data Discovery and Exploration for Spark3 for Azure	worker	Standard_D8_v3	Standard_D8s_v3
7.x - Data Discovery and Exploration for Spark3 for Azure	yarnworker	Standard_D8_v3	Standard_D8s_v3
7.x - Data Mart: Apache Impala, Hue for Azure	master	Standard_E8_v3	Standard_E8s_v3
7.x - Edge Flow Management Light Duty for Azure	management	Standard_D8_v3	Standard_D8s_v3
7.x - COD Edge Node for for Azure	leader	Standard_D8_v3	Standard_D8s_v3
7.x - Streaming Analytics Heavy Duty for Azure	manager	Standard_D8_v3	Standard_D8s_v3
7.x - Streaming Analytics Heavy Duty for Azure	master	Standard_D8_v3	Standard_D8s_v3
7.x - Streaming Analytics Heavy Duty for Azure	worker	Standard_D8_v3	Standard_D8s_v3
7.x - Streaming Analytics Light Duty for Azure	manager	Standard_D8_v3	Standard_D8s_v3
7.x - Streaming Analytics Light Duty for Azure	master	Standard_D8_v3	Standard_D8s_v3
7.x - Streaming Analytics Light Duty for Azure	worker	Standard_D8_v3	Standard_D8s_v3
7.x - Flow Management Light Duty for Azure	management	Standard_D8_v3	Standard_D8s_v3
7.x - Flow Management Light Duty for Azure	nifi_scaling	Standard_D8_v3	Standard_D8s_v3
7.x - Flow Management Light Duty for Azure	nifi	Standard_D8_v3	Standard_D8s_v3
7.x - Flow Management Heavy Duty for Azure	management	Standard_D8_v3	Standard_D8s_v3
7.x - Real-time Data Mart - Spark3 for Azure	master1	Standard_D8_v3	Standard_D8s_v3
7.x - Real-time Data Mart - Spark3 for Azure	master2	Standard_D8_v3	Standard_D8s_v3
7.x - Real-time Data Mart - Spark3 for Azure	master3	Standard_D8_v3	Standard_D8s_v3
7.x - Streams Messaging High Availability for Azure	manager	Standard_D16_v3	Standard_D16s_v3
7.x - Streams Messaging High Availability for Azure	core_zookeeper	Standard_D8_v3	Standard_D8s_v3

7.x - Streams Messaging High Availability for Azure	core_broker	Standard_D8_v3	Standard_D8s_v3
7.x - Streams Messaging High Availability for Azure	srm	Standard_D8_v3	Standard_D8s_v3
7.x - Streams Messaging Light Duty for Azure	broker	Standard_D8_v3	Standard_D8s_v3
7.x - Streams Messaging Light Duty for Azure	kraft	Standard_D8_v3	Standard_D8s_v3
7.x - Streams Messaging Heavy Duty for Azure	registry	Standard_D8_v3	Standard_D8s_v3
7.x - Streams Messaging Heavy Duty for Azure	smm	Standard_D8_v3	Standard_D8s_v3
7.x - Streams Messaging Heavy Duty for Azure	srm	Standard_D8_v3	Standard_D8s_v3
7.x - Streams Messaging Heavy Duty for Azure	connect	Standard_D8_v3	Standard_D8s_v3
7.x - Streams Messaging Heavy Duty for Azure	kraft	Standard_D8_v3	Standard_D8s_v3

If you would like to create Cloudera Data Hub clusters with the previously used instance types, you can achieve this by configuring a custom instance type.

April 2024

This release of the Cloudera Data Hub service introduces the following changes:

Cloudera Runtime 7.2.18

Cloudera Runtime 7.2.18 is now available and can be used for registering an environment with a 7.2.18 Data Lake and creating Cloudera Data Hub clusters. For more information about the new Runtime version, see Cloudera Runtime. If you need to upgrade your existing Cloudera environment, your upgrade path may be complex. To determine your upgrade path, refer to Upgrading to Cloudera Runtime 7.2.18 documentation.

RHEL replaces CentOS as default OS

As of June 30, 2024, CentOS reaches End of Life (EOL), and consequently, Cloudera Runtime 7.2.18 supports RHEL 8 only. New deployments of Data Lakes and Cloudera Data Hub clusters with Cloudera Runtime 7.2.18 and upgrades to 7.2.18 are only possible with RHEL 8. Data Lake and Cloudera Data Hub clusters running Cloudera Runtime 7.2.17 support both CentOS 7 and RHEL 8. Earlier Runtime versions support CentOS 7 only. Cloudera will not publish any updates or fixes for CentOS-based images after June 2024.

As part of FreeIPA, Data Lake, and Cloudera Data Hub upgrade, you have the option to upgrade the operating system (OS) on the virtual machines (VMs) from CentOS 7 to Red Hat Enterprise Linux 8 (RHEL 8). For more information, see Upgrading from CentOS to RHEL.

Rolling upgrade support for Cloudera Operational Database Data Hub clusters

With the release of Runtime 7.2.18, rolling upgrades for certain Cloudera Operational Database Data Hub clusters are now available. For more information, see Rolling upgrades.

Disk vertical scaling in Azure (Preview)

Disk vertical scaling (that is, disk type change and resizing) is now supported by Cloudera for Data Lakes and Cloudera Data Hub clusters running in Azure. Previously, only AWS support was available for this feature in Cloudera. For more information, see Disk Vertical Scaling - Disk Type Change and Resizing in AWS and Azure.

Note: You need to contact Cloudera to have this feature enabled.

March 2024

This release of the Cloudera Data Hub service introduces the following changes:

Cloudera Data Hub root volume size was increased to 200 GB

The root volume size of Cloudera Data Hub clusters was increased from 100 GB to 200 GB.

2023

October 2023

This release of the Data Hub service introduces the following changes:

Streams Messaging High Availability clusters now include highly available Knox (October 30, 2023)

The Streams Messaging High Availability cluster configuration is updated. The configuration now includes a highly available Knox service. Specifically, a new host group called manager is introduced. This host group is not scalable and includes two nodes. Both nodes host a Knox service instance. In addition, Cloudera Manager is moved from the master host group and is now deployed on one of the manager nodes. The master host group now only includes Cruise Control and Streams Messaging Manager.

The previous cluster configuration is no longer available. Streams Messaging High Availability clusters provisioned at a previous date will continue to run using the old configuration. Updating an existing cluster to use the new configuration is not possible. For more information, see Streams Messaging cluster layout.

August 2023

This release of the Data Hub service introduces the following changes:

Support for Local SSDs in GCP (August 21, 2023)

Data Hub now supports using Local SSDs as storage in GCP. During Data Hub cluster creation in CDP, you can navigate to advanced Hardware and Storage options and select to use "Local scratch disk (SSD)" with certain instance types. Prior to using Local SSDs with CDP, make sure to review Google Cloud Platform documentation related to Local SSDs and familiarize yourself with the applicable restrictions and limitations (such as 12 TB maximum capacity, limited configurations).

Note: Stopping and restarting Data Hub clusters using local SSDs is not supported; When you stop or suspend a VM, all data on the attached Local SSD is discarded.

Data Hub database upgrade and default major version change (August 30, 2023)

Newly-deployed Data Hub clusters on AWS or GCP with Cloudera Runtime 7.2.7 or above are now configured to use a PostgreSQL version 14 database by default.

Newly-deployed Data Hub clusters on Azure with Cloudera Runtime 7.2.7 or above will continue to use a PostgreSQL version 11 database by default.

The database for Data Hub clusters on AWS and GCP can now be upgraded to PostgreSQL version 14. If your AWS or GCP Data Hub cluster requires an upgrade to PostgreSQL 14, you will receive a notification in the Management Console UI.

Cloudera strongly recommends that the database upgrade to PostgreSQL 14 for AWS and GCP clusters is performed on all clusters running PostgreSQL version 11 by November 9, 2023.

A database upgrade to PostgreSQL 14 for Azure Data Hubs will be available in the future. Any Data Hub clusters on Azure that require a database upgrade will be upgraded from PostgreSQL 10 to PostgreSQL 11.

For more information, see Upgrading Data Lake/Data Hub database

Support for autoscaling Data Hub clusters on Azure (August 30, 2023)

Data Hub now supports autoscaling for clusters provisioned in Azure. For more information see Autoscaling clusters.

July 2023

This release of the Data Hub service introduces the following changes:

Java 11 support for Data Hub and Data Lake clusters (July 12, 2023)

You can launch a new Data Hub or Data Lake cluster with Java 11 as the default JDK. This capability is currently only available through the CDP CLI. See Java 11 for more information.

April 2023

This release of the Data Hub service introduces new features and behaviors.

GP3 support for attached storage disks (April 19)

Data Hubs now support GP3 (SSD) volume types for attached storage. GP3 volumes allow you to increase performance (independently provisioning IOPS and throughput) without increasing storage size. GP3 volumes will deliver similar performance as GP2 volumes at a lower cost. GP3 is now the default attached storage type for Data Hub instances that previously used GP2 storage. You can configure storage options when you provision a Data Hub cluster under the Hardware and Storage section of the Advanced Options.

February 2023

This release of the Data Hub service introduces new features and behaviors.

Data Hub Autoscaling (February 27, 2023)

Autoscaling is a feature that adjusts the capacity of cluster nodes running YARN by automatically increasing or decreasing, or suspending and resuming, the nodes in a host group. You can enable autoscaling based either on a schedule that you define, or the real-time demands of your workloads. For more information see Autoscaling clusters.

January 2023

This release of the Data Hub service introduces new features and behaviors.

Vertical scaling of Data Hub instances on AWS

If necessary, you can now select a larger or smaller instance type for a Data Hub cluster after it has been deployed. This is only supported on AWS. For more information see Vertically scaling instances types.

2022

December 2022

This release of the Data Hub service introduces new features and behaviors.

Instance support (December 19, 2022)

The following new instance types are supported in Data Hub:

AWS:

- m6i.xlarge
- m6i.2xlarge
- m6i.4xlarge
- m6i.8xlarge
- m6i.12xlarge

- m6i.16xlarge
- m6i.24xlarge
- m6i.32xlarge
- m6id.xlarge
- m6id.2xlarge
- m6id.4xlarge
- m6id.8xlarge
- m6id.12xlarge
- m6id.16xlarge
- m6id.24xlarge
- m6id.32xlarge
- m6a.xlarge
- m6a.2xlarge
- m6a.4xlarge
- m6a.8xlarge
- m6a.12xlarge
- m6a.16xlarge
- m6a.24xlarge
- m6a.32xlarge
- m6a.48xlarge

Azure:

- Standard_D8_v5
- Standard_D16_v5
- Standard_D32_v5
- Standard_D48_v5
- Standard_D64_v5
- Standard_D8s_v5
- Standard_D16s_v5
- Standard_D32s_v5
- Standard_D48s_v5
- Standard_D64s_v5
- Standard_D8as_v5
- Standard_D16as_v5
- Standard_D32as_v5
- Standard_D48as_v5
- Standard_D64as_v5

October 2022

This release of the Data Hub service introduces new features and behaviors.

Changes to Data Hub cluster deployments on AWS (October 28, 2022)

The CloudFormation template previously in use for Data Hub deployments on AWS is no longer required for new Data Hub deployments. Clusters provisioned without the CloudFormation template will have the following changes:

- Hostnames in the AWS Management Console appear differently between native and CloudFormation clusters. For example:
  - dev-pdp-de-fix-01-master0 is an example of a hostname in a cluster deployed with the CloudFormation template
  - devpdpdefix01-164597-master-0 is an example of a hostname in a cluster deployed without the CloudFormation template

- Some of the default tags were removed during the implementation of this change from CloudFormation to native deployed clusters:
  - aws:autoscaling:groupName
  - aws:cloudformation:stack-id
  - aws:ec2launchtemplate:id
  - aws:ec2launchtemplate:version
  - aws:cloudformation:logical-id
  - aws:ec2:fleet-id
  - aws:cloudformation:stack-name

September 2022

This release of the Data Hub service introduces new features.

Database Upgrade and default major version change (September 27, 2022)

Newly deployed Data Lake and Data Hub clusters with Cloudera Runtime 7.2.7 or above are now configured to use a PostgreSQL version 11 database by default.

A new Database Upgrade capability is now available for existing Data Lake and Data Hub clusters. If you are running clusters on Cloudera Runtime version 7.2.6 or below, please upgrade to a more recent version before performing the database upgrade.

The major version of the database used by Data Lake or Data Hub clusters is now also displayed on the Database page of the respective service.

Cloudera strongly recommends that the Database Upgrade is performed on all clusters running PostgreSQL version 10 before November 10, 2022.

For more information, see Upgrading database.

Recipe type changes

The following recipe types have been renamed for Data Hub, Data Lake, and FreeIPA recipes:

- pre-service-deployment (formerly pre-cluster-manager-start)
- post-service-deployment (formerly post-cluster-install)

These changes will not affect existing recipe automation. For more information see Recipes.

Instance support (September 1, 2022)

The following new AWS instance types are supported in Data Hub:

- m5dn.xlarge
- m5dn.2xlarge
- m5dn.4xlarge
- m5dn.8xlarge
- m5dn.12xlarge
- m5dn.16xlarge
- m5dn.24xlarge

Validate and prepare for upgrade (September 1, 2022)

Before you perform a Data Hub upgrade, you can run the new Validate and Prepare option to check for any configuration issues and begin the Cloudera Runtime parcel download and distribution. Using the validate and prepare option does not require downtime and makes the maintenance window for an upgrade shorter. For more information see Preparing for an upgrade.

July 2022

This release of the Data Hub service introduces new features.

The default authentication method for Kafka brokers is changed to LDAP (June 28, 2022)

To improve performance, the default SASL_SSL/PLAIN authentication method for Kafka brokers is changed to LDAP. Previously, brokers were configured to use PAM. This change is Runtime version independent and applies to all Data Hub clusters provisioned after July 28, 2022. For existing clusters, Cloudera recommends that you transition from PAM to LDAP. For more information on how to enable and configure LDAP on existing clusters, see Kafka LDAP Authentication.

June 2022

This release of the Data Hub service introduces new features.

Customer managed encryption keys on AWS (June 7, 2022)

Previously, Data Hub EBS encryption could be configured only as part of Data Hub creation. Now you can also configure encryption during environment registration and all EBS volumes and RDS database and used by the Data Hubs running in that environment will be encrypted. It is still possible to add a CMK during Data Hub creation, overwriting the encryption key provided during environment registration for that Data Hub. For more information, refer to Adding a customer managed encryption key to a CDP environment running on AWS and Configuring encryption for Data Hub's EBS volumes on AWS.

Cluster template overrides (June 7, 2022)

As an alternative to creating a custom template, you can specify custom configurations that override or append the properties (service and role configs) in a built-in or custom Data Hub template. These configurations are saved as a shared resource called “cluster template overrides,” and can be used and re-used across Data Hub clusters in different environments.

For more information, see Cluster template overrides.

Creating multi-AZ Data Hubs on AWS (June 9, 2022)

By default, CDP provisions Data Hubs in a single AWS availability zone (AZ), but you can optionally choose to deploy them across multiple availability zones (multi-AZ). For more information, see Creating a multi-AZ Data Hub on AWS.

Multi-AZ deployments do not use a CloudFormation template for VM creation. Neither autoscaling groups or launch templates are created. The cluster resources are managed individually using AWS native components (for example, EC2 instances).

Customer managed encryption keys on GCP (June 16, 2022)

By default, a Google-managed encryption key is used to encrypt disks and Cloud SQL instances used by Data Hub clusters, but you can optionally configure CDP to use a customer-managed encryption key (CMEK) instead. This can only be configured during CDP environment registration using CDP CLI. For more information, refer to Adding a customer managed encryption key for GCP.

Data Engineering HA template for Azure (June 22, 2022)

The Data Engineering HA template for Azure is now GA. For more information see Data Engineering clusters (Azure).

New option to delete attached volumes during Data Hub repair (June 29, 2022)

When you initiate a repair from the Data Hub Hardware tab, you have the option to delete any volumes attached to the instance. For more information see Performing manual repair.

May 2022

This release of the Data Hub service introduces new features.

Resilient scaling improvements (May 10, 2022)

- It is possible to upscale a Data Hub cluster even if there are unhealthy nodes in the stack.
- An upscale operation will finish successfully even if one of the new nodes fails to come up. Nodes that fail during an upscale will be marked as "zombie" nodes.

For more information, see Resize a cluster.

Extended upgrade version support for RAZ-enabled environments (May 10, 2022)

Data Hub major/minor version upgrades for RAZ-enabled clusters are now available for Runtime versions 7.2.10-7.2.12 to 7.2.14+.

Instance support (May 25, 2022)

The following new AWS instance types are supported in Data Hub:

- m5ad.xlarge
- m5ad.2xlarge
- m5ad.4xlarge
- m5ad.8xlarge
- m5ad.12xlarge
- m5ad.16xlarge
- m5ad.24xlarge
- m5d.xlarge
- m5d.2xlarge
- m5d.4xlarge
- m5d.8xlarge
- m5d.12xlarge
- m5d.16xlarge
- m5d.24xlarge
- r5ad.12xlarge
- r5ad.24xlarge
- r5d.24xlarge
- r5d.12xlarge

April 2022

This release of the Data Hub service introduces new features.

Data Hub OS upgrades

OS upgrades of Cloudera Runtime are generally available. See Performing a Data Hub OS upgrade.

March 2022

This release of the Data Hub service introduces new features.

Instance support (March 4, 2022)

The following new GCP instances are supported in Data Hub:

- n2d-highmem-32
- n2d-highmem-64

Public certificate auto-renewal (March 7, 2022)

Most public (Let's Encrypt-issued) certificates for Data Lake and Data Hub clusters will now auto-renew without intervention from a user. For more information, refer to Managing Certificates.

Data Hub upgrades

Major/minor version upgrades of Cloudera Runtime and Cloudera Manager are generally available for most Data Hub templates. Data Hub maintenance upgrades for RAZ-enabled Data Hubs are generally available for versions 7.2.7+. For more information see Upgrading Data Hubs.

February 2022

This release of the Data Hub service introduces new features.

Customer managed encryption keys for Azure disks and databases

By default, local Data Hub disks attached to Azure VMs and the PostgreSQL server instance used by the Data Lake and Data Hubs are encrypted with server-side encryption (SSE) using Platform Managed Keys (PMK), but during environment registration you can optionally configure SSE with Customer Managed Keys (CMK). For more information, refer to Adding a customer managed encryption key for Azure. For more information, refer to Adding a customer managed encryption key for Azure.

Instance support

The following new AWS and Azure GPU-based instances are supported in Data Hub:

AWS:

- c5.12xlarge
- c5a.12xlarge

Azure:

- NC6sv3
- NC24sv3

January 2022

This release of the Data Hub service introduces new features.

Updating recipes post-deployment

You can attach or detach recipes to/from existing Data Hub clusters. Using this capability, you can update a recipe by removing it from the cluster, replacing the old recipe with a modified recipe of the same type, and attaching the new modified recipe to the cluster. For more information refer to Update a recipe.

Custom images and catalogs

If necessary, you can use a custom Runtime image for compliance or security reasons. You can then use the CDP CLI to register a custom image catalog and set the custom image within the custom image catalog. For more information refer to Custom images and catalogs.

2021

October 2021

This release of the Data Hub service introduces new features.

Encrypting EBS volumes with a CMK generated from AWS CloudHSM (October 4)

New documentation with steps for encrypting Data Hub's EBS volumes with a CMK generated from AWS CloudHSM is available. See Encrypting EBS volumes with a CMK generated from AWS CloudHSM.

Data Hub maintenance upgrades (October 26)

You can perform a maintenance (or "hotfix") upgrade on certain Data Hub clusters. For more information see Upgrading Data Hubs.

August 2021

This release of the Data Hub service introduces new features.

Data Engineering HA template

The Oozie service is now highly available in the Data Engineering HA templates for AWS and Azure.

May 31, 2021

This release of the Data Hub service introduces new features.

Data Engineering HA template on AWS

The Data Engineering HA template is now GA. It now supports fault-tolerant access to all cluster web UIs and end-to-end reliability for more workload types.

March 30, 2021

This release of the Data Hub service introduces new features.

Google Cloud Platform support for Data Hub

CDP now supports the creation of Data Hub clusters on the GCP platform. For more information see Create a Data Hub cluster on Google Cloud.

Show CLI Command

You can obtain the CDP CLI command for creating a Data Hub cluster from the details of an existing Data Hub or from the create Data Hub wizard. See Create a Data Hub cluster on AWS from an existing cluster and Create a Data Hub cluster on Azure from an existing cluster.

Default Root Volume Size

The default root volume size has been increased from 30-50 GB (depending on cloud provider) to 100 GB for all cloud providers.

Data Engineering and Data Engineering HA Templates

- Instance types in the Data Engineering and Data Engineering HA templates have been updated to 64 GB memory machines from 32 GB memory machines:
  - The master instance in the DE template has changed from m5.2xlarge to m5.4xlarge for AWS and from Standard_D8_v3 to Standard_D16_v3 for Azure.
  - All instances in the DE HA template have changed from m5.2xlarge to m5.4xlarge for AWS and from Standard_D8_v3 to Standard_D16_v3 for Azure
- From Cloudera Runtime 7.2.8 onwards, the DE HA template includes a new load-balancer added for the Knox instances for UI access. The external endpoint has changed from "-manager0" to "-gateway" because of the presence of the load-balancer.

February 28, 2021

This release of the Data Hub service introduces new features.

Support for Cloud Hardware Redundancy

Added support for AWS placement groups guarantees physical hardware redundancy for Data Hub clusters. See Placement group support for more details.

2020

September 30, 2020

This release of the Data Hub service introduces new features.

Data Discovery and Exploration Template (Tech Preview) now includes a Spark connector

- Provides support for indexing of RDDs and for Spark Streaming.
- Facilitates speedier data ingestion pipelines to Solr, while allowing all the power and flexibility of Spark to transform and process data on its way to be served.

The “7.2.0 - Operational Database” Data Hub template includes HUE for HBase DML/DDL

This update greatly simplifies the process of data exploration and query.

July 31, 2020

This release of the Data Hub service introduces new features.

System auditing for Data Hub events (as well as Data Lake and user management) is now available via CLI. For more information see the *Auditing* section of the Management Console documentation.

June 30, 2020

This release of the Data Hub service introduces new features.

YARN Queue Manager in Data Engineering Cluster Templates

Data Engineering cluster administrators can now leverage the visual YARN Queue Manager via Cloudera Manager to isolate and prioritize workloads within a cluster. This simplifies the administration and management of workloads.

Data Discovery and Exploration (Technical Preview)

A new Data Discovery and Exploration cluster definition is available in Data Hub. The cluster definition deploys HDFS, Hue, Solr, Spark, Yarn, and ZooKeeper services. The cluster definition is available for AWS.

May 1, 2020

This release of the Data Hub service introduces the following new features:

Flow Management for Data Hub Cluster Definitions

New Flow Management cluster definitions are available in Data Hub. The Flow Management cluster definitions (heavy duty and light duty) include the Nifi, Nifi Registry, and Zookeeper services. The cluster definitions are available for both AWS and Azure.

Data Engineering HA Cluster Definition (Technical Preview)

A new Data Engineering HA cluster definition is available in Data Hub for technical preview. The cluster definition deploys YARN, HDFS, and Hive in a highly available fashion.

External Databases for Data Hub Clusters

Through the CDP CLI, you can now configure an external, durable database for Data Hub clusters. An external database can keep a persistent state for services like Hue, DAS, and Zeppelin.

March 2, 2020

This release of the Data Hub service introduces the following new features:

Data Hub Clusters on Microsoft Azure

You can create a Data Hub cluster on Azure. You have two basic options when creating a cluster: (1) create a cluster from one of the default or custom cluster definitions, or (2) create a custom cluster.

Cloudera Runtime 7.1.0

By default, Data Hub creates clusters running Cloudera Runtime 7.1.0 and Cloudera Manager 7.1.0. Previously, 7.0.2.2 was used.

2019

December 17, 2019

This release of the Data Hub service introduces the following new features:

Cloudera Runtime 7.0.2.2

By default, Data Hub creates clusters running Cloudera Runtime 7.0.2.2 and Cloudera Manager 7.0.2.2. Previously, 7.0.2.0 was used. Note that there is no separate 7.0.2.2 documentation available, but Cloudera Runtime 7.0.2 and Cloudera Manager 7.0.2 documentation has been updated to reflect the 7.0.2.2 changes;

New Streams Messaging cluster templates and definitions

The Streams Messaging Heavy Duty and Streams Messaging Light Duty cluster templates and definitions are now available in Data Hub with advanced messaging and real-time processing on streaming data using Apache Kafka, centralized schema management using Schema Registry, and management and monitoring capabilities powered by Streams Messaging Manager.

Note:

The Streams Messaging cluster templates and definitions are technical preview: They are not suitable for production.

November 14, 2019

This release of the Data Hub service introduces the following new features:

Cloudera Runtime 7.0.2

By default, Data Hub creates clusters running Cloudera Runtime 7.0.2 and Cloudera Manager 7.0.2. Previously, 7.0.1 was used.

New Real-Time Data Mart cluster definition

The Real-Time Data Mart cluster definition is now available in Data Hub with Apache Kudu, Apache Impala, and Apache Spark for real-time data management and operational analytics use cases. This cluster definition is suitable for real-time analytics on fast changing data, and can be leveraged together with the Data Warehouse service for a holistic analytics experience.

Enhanced SQL access for Data Engineering clusters

Data Hub clusters built with Data Engineering cluster definition now offer an improved SQL experience for Hive SQL users with smart query authorizing, build-in performance analysis and recommendations tools available in Data Analytics Studio (DAS). This capability provides an enhanced self-service experience for data analysts and data engineers. Note that DAS web UI can be accessed by using workload user name and FreeIPA password.

Operational Database cluster enhancements

The Operational Database cluster definition enables you to build real-time, high-concurrency, mission-critical business applications in the cloud environment. Key enhancements in this release include real-time monitoring, out-of-the-box resilience, and replication capabilities (Tech Preview) for Disaster Recovery (DR) and for enabling seamless migration of operational applications from on-premises to cloud.

September 23, 2019

This release of the Data Hub service introduces the following new features:

Support for Cloudera Runtime 7.0.1

By default, Data Hub creates clusters running Cloudera Runtime 7.0.1. Previously, 7.0.0 was used.

August 22, 2019

This is the first release of the Data Hub service. For an overview of Data Hub functionality, refer to:

Related Information

Cloudera Data Hub Overview

Log4j vulnerabilities

Cloudera has released hotfixes for Cloudera Runtime versions 7.2.7 and newer on Public Cloud that address Log4j2 vulnerabilities.

The following vulnerabilities have been addressed for Cloudera Runtime versions 7.2.7 through 7.2.12:

CVE-2021-44228

CVE-2021-45046

LOGBACK-1591

CVE-2021-45105

CVE-2021-44832

You should upgrade your Cloudera services running Cloudera Runtime versions 7.2.7+ so that they include the latest hotfixes. You can update your existing Data Lake and Cloudera Data Hub clusters by doing a maintenance upgrade. You should first upgrade the Data Lake and then upgrade all the Cloudera Data Hub clusters that are using the Data Lake. Refer to Data Lake upgrade and Cloudera Data Hub upgrade documentation. Data Lake and Cloudera Data Hub maintenance upgrade is supported only in technical preview for maintenance upgrades from Cloudera Runtime versions 7.2.7 and higher.

If you are running a version of Cloudera Runtime prior to 7.2.7, contact Cloudera Support for details on how to upgrade Cloudera Runtime.

For more information about these hotfixes, refer to the Cloudera Runtime Release Notes for the version of Runtime that you use.

Known issues

This section lists known issues that you might run into while using the Cloudera Data Hub service.

CB-11925: Knox Load Balancer API requests initiated from Knox Gateway hosts can fail with Connection timeout error

Problem: When logged into a Data Lake or Cloudera Data Hub node that has a Knox Gateway service instance configured on it, making Knox API calls through the Knox load balancer can result in a connection timeout error.

This is because for security reasons, the IP address of the request is preserved in the traffic passed through the load balancer. Preserving the IP address means that the load balancer will reject "loopback" traffic, meaning traffic that originates and is directed back to the same node.

Workaround: If Knox API calls need to be made while logged into a Knox gateway node, use the hostname of the node instead of the load balancer hostname in the API call.

The Knox load balancer hostname can be identified by the "-gateway" suffix in the first clause of the hostname with no numeric identifier. For example: <cluster-name>-gateway.<env-shortname>.<hash>.cloudera.site is the load balancer hostname, and <cluster-name>-gateway0.<env-shortname>.<hash>.cloudera.site is a direct node hostname.

CB-23926: Autoscaling must be stopped before performing FreeIPA upgrade

Problem: During autoscaling, compute nodes are stopped and cannot receive the latest FreeIPA IP address update in a stopped state. This causes the Cloudera Data Hub clusters to become unhealthy after the FreeIPA upgrade when autoscaling is enabled.

Workaround:

1. Disable autoscaling before perfoming FreeIPA upgrade.
2. Start all compute nodes, which are in a STOPPED state.
3. Perform the FreeIPA upgrade.
4. Enable autoscaling after FreeIPA upgrade is successful.

CB-23940: Ranger YARN repository is not cleaned up after deleting stopped cluster

Problem: When a stopped Cloudera Data Hub cluster is deleted, the Ranger YARN repository that belongs to the deleted Cloudera Data Hub cluster is not cleaned up. This causes errors when creating a cluster with the same cluster name.

Workaround: Start the Cloudera Data Hub cluster before deleting. If the cluster was deleted in stopped state, delete the YARN repository using the Ranger Admin UI.

Technical Service Bulletins

TSB 2022-568: HBase normalizer must be disabled for Salted Phoenix tables

When Apache Phoenix (“Phoenix”) creates a salted table, it pre-splits the table according to the number of salt regions. These regions must always be kept separate, otherwise Phoenix does not work correctly.

The HBase normalizer is not aware of this requirement, and in some cases the pre-split regions are merged automatically. This causes failure in Phoenix.

The same requirement applies when merging regions of salted tables manually: regions containing different salt keys (the first byte of the rowkey) must never be merged.

Note that either automatic or manual splitting of the regions for a salted table does not cause a problem. The problem only occurs when adjacent regions containing different salt keys are merged.

Upstream JIRA

PHOENIX-4906

Knowledge article

For the latest update on this issue, see the corresponding Knowledge article: TSB 2022-568: Hbase normalizer must be disabled for Salted Phoenix tables

Image catalog updates

This section lists prewarmed image updates in Cloudera Data Hub.

The following prewarmed images were added on the following dates:

First released	Platform	Runtime component versions	Cloudera Manager version	Available OSs
April 3, 2024	Cloudera Runtime 7.2.18	7.2.18 Component Versions	Cloudera Manager 7.12.0	RHEL 8.8 EUS
June 27, 2023	Cloudera Runtime 7.2.17	7.2.17 Component Versions	Cloudera Manager 7.11.0	RHEL 8.8 EUS, CentOS 7
January 11, 2023	Cloudera Runtime 7.2.16	7.2.16 Component Versions	Cloudera Manager 7.9.0	CentOS 7
May 12, 2022	Cloudera Runtime 7.2.15	7.2.15 Component Versions	Cloudera Manager 7.6.2	CentOS 7
February 25, 2022	Cloudera Runtime 7.2.14	7.2.14 Component Versions	Cloudera Manager 7.6.0	CentOS 7
October 25, 2021	Cloudera Runtime 7.2.12	7.2.12 Component Versions	Cloudera Manager 7.5.2	CentOS 7
September 9, 2021	Cloudera Runtime 7.2.11	7.2.11 Component Versions	Cloudera Manager 7.4.3	CentOS 7
June 14, 2021	Cloudera Runtime 7.2.10	7.2.10 Component Versions	Cloudera Manager 7.4.2	CentOS 7
April 30, 2021	Cloudera Runtime 7.2.9	7.2.9 Component Versions	Cloudera Manager 7.4.1	CentOS 7
March 25, 2021	Cloudera Runtime 7.2.8	7.2.8 Component Versions	Cloudera Manager 7.4.0	CentOS 7
February 9, 2021	Cloudera Runtime 7.2.7	7.2.7 Component Versions	Cloudera Manager 7.3.0	CentOS 7
December 11, 2020	Cloudera Runtime 7.2.6	7.2.6 Component Versions	Cloudera Manager 7.2.6	CentOS 7
October 8, 2020	Cloudera Runtime 7.2.2	7.2.2 Component Versions	Cloudera Manager 7.2.2	CentOS 7
July 31, 2020	Cloudera Runtime 7.2.1	7.2.1 Component Versions	Cloudera Manager 7.2.1	CentOS 7
June 16, 2020	Cloudera Runtime 7.2.0	7.2.0 Component Versions	Cloudera Manager 7.2.0	CentOS 7
February 20, 2020	Cloudera Runtime 7.1.0	7.1.0 Component Versions	CCloudera Manager 7.1.0	CentOS 7

For the latest image catalog updates, refer to Cloudera web interface > Cloudera Management Console > Shared Resources > Image Catalogs.

Cloudera Data Hub

Date published: 2019-12-17
Date modified: 2023-06-27

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cluster deployment options 4

Default cluster configurations 4
- Data Engineering clusters 5
- Data Mart clusters 9
- Operational Database clusters 11
- Streams Messaging clusters 12
- Flow Management clusters 13
- Streaming Analytics clusters 14
- Data Discovery and Exploration clusters 15

Create a cluster from a definition on AWS 16

Create a custom cluster on AWS 18

Advanced cluster options 20
- Tags 20
- Image catalog 20
- Custom images and image catalogs 21
  - Creating a custom image and image catalog with the CDP CLI 22
  - Switching image catalogs 24
- Network and availability 25
- Hardware and storage 25
- Cloud storage 26

Building a custom cluster template 26

CDP Public Cloud upgrade advisor 32
- Upgrade Checklist FAQ 32
- Preparing for an upgrade 33

Cluster deployment options

You have two basic options when creating a cluster: (1) create a cluster from one of the default or custom cluster definition, or (2) create a custom cluster.

Cluster definitions

Cloudera Data Hub includes a set of prescriptive cluster definitions that can be used to quickly provision clusters for common use cases. These default cluster definitions include prescriptive options related to compute instance types and storage options.

We recommend that you start by reviewing these definitions. If you conclude that they do not match your requirements, you can customize them and then save them as custom cluster definitions.

Custom deployment

Use the custom cluster deployment option if you would like to select specific Cloudera Runtime components for your cluster.

Note:

Currently, this feature is limited and only allows you to select a cluster template that determines which components will be used for your cluster.

Advanced options

The cluster wizard includes a set of advanced options that can be optionally configured. Some of these options require additional configuration prior to cluster creation, so if you would like to use these options, read the cluster planning documentation first.

Default cluster configurations

Cloudera Data Hub includes a set of prescriptive cluster configurations. Each of these default cluster configurations include a cloud-provider specific cluster definition, which primarily defines cloud provider settings. The cluster definition references a cluster template, which defines a number of Cloudera Runtime or Cloudera DataFlow components used for common data analytics and data engineering use cases.

Refer to the topic for each default cluster configuration to view the included services and compatible Runtime versions. These topics include links to documentation that will help you to understand the included components and use the workload cluster.

Many of the cluster components are included in the Cloudera Runtime software distribution. The Streams Messaging, Cloudera Flow Management, and Cloudera Streaming Analytics cluster configurations are part of Cloudera DataFlow for Data Hub and have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for more details.

You can access the default cluster definitions by clicking Environments, then selecting an environment and clicking the Cluster Definitions tab.

You can access the default cluster templates from Shared ResourcesCluster Templates.

To view details of a cluster definition or cluster template, click on its name. For each cluster definition, you can access a raw JSON file. For each cluster template, you can access a graphical representation ("list view") and a raw JSON file ("raw view") of all cluster host groups and their components.

Related Information

Cloudera DataFlow for Data Hub

Cloudera Runtime

Data Engineering clusters

Learn about the default Data Engineering clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Engineering provides a complete data processing solution, powered by Apache Spark and Apache Hive. Spark and Hive enable fast, scalable, fault-tolerant data engineering and analytics over petabytes of data.

Data Engineering cluster definition

This Data Engineering template includes a standalone deployment of Spark and Hive, as well as Apache Oozie for job scheduling and orchestration, Apache Livy for remote job submission, and Hue and Apache Zeppelin for job authoring and interactive analysis.

Cluster definition names

- Data Engineering for AWS
- Data Engineering HA - Spark3 for AWS

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering Spark3 for AWS

Cluster template name

- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: Apache Spark, Apache Hive, Apache Oozie."
- Data Engineering: HA: Apache Spark3, Apache Hive, Apache Oozie

  Note: This cluster template was formerly named "Data Engineering: HA: Apache Spark, Apache Hive, Apache Oozie."

  See the architectural information below for the Data Engineering HA clusters
- Data Engineering: Apache Spark3, Apache Hive, Apache Oozie

  Note: The "Data Engineering: Apache Spark3" cluster template is deleted. Therefore, the "Data Engineering: Apache Spark3, Apache Hive, Apache Oozie" cluster template can be used instead.

Included services

- HDFS
- Hive
- Hue
- Livy
- Spark 3
- Yarn
- Zeppelin
- ZooKeeper
- Oozie is supported for Spark 3 as of Cloudera Runtime version 7.2.18
- Hive Warehouse Connector is supported as of Cloudera Runtime version 7.2.16.

Compatible Cloudera Runtime version

7.2.16, 7.2.17, 7.2.18

Topology of the Data Engineering cluster

Topology is a set of host groups that are defined in the cluster template and cluster definition used by Data Engineering. Data Engineering uses the following topology:

Host group	Description	Node configuration
Master	The master host group runs the components for managing the cluster resources including Cloudera Manager, Name Node, Resource Manager, as well as other master components such HiveServer2, HMS, Hue etc.	1 For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.4xlarge; gp2 - 100 GB For Cloudera Runtime versions 7.2.14+ DE, DE Spark3, and DE HA: AWS : m5.4xlarge; gp2 - 100 GB
Worker	The worker host group runs the components that are used for executing processing tasks (such as NodeManager) and handling storing data in HDFS such as DataNode.	3 For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.2xlarge; gp2 - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: AWS: r5d.2xlarge - (gp2/EBS volumes) DE HA: AWS: r5d.4xlarge - (gp2/EBS volumes)
Compute	The compute host group can optionally be used for running data processing tasks (such as NodeManager). By default the number of compute nodes is set to 1 for proper configurations of YARN containers. This node group can be scaled down to 0 when there are no compute needs. Additionally, if load-based auto-scaling is enabled with minimum count set to 0, the compute nodegroup will be resized to 0 automatically.	0+ For Cloudera Runtime versions earlier than 7.2.14: AWS : m5.2xlarge; gp2 - 100 GB For Cloudera Runtime versions 7.2.14+ DE and DE Spark3: AWS: r5d.2xlarge - (ephemeral volumes) DE HA: AWS: r5d.4xlarge - (ephemeral volumes)

Host group	Description	Node configuration
		Note: Compute nodes run YARN and require storage only for temporary data - this requirement is fulfilled by instance storage, so making the attached volumes count to 0 by default is more cost-efficient.
Gateway	The gateway host group can optionally be used for connecting to the cluster endpoints like Oozie, Beeline etc. This nodegroup does not run any critical services. This nodegroup resides in the same subnet as the rest of the nodegroups. If additional software binaries are required they could be installed using recipes.	0+ AWS : m5.2xlarge; gp2 - 100 GB

Service configurations			
Master host group Cloudera Manager, HDFS, Hive (on Tez), HMS, Yarn RM, Oozie, Hue, DAS, Zookeeper, Livy, Zeppelin and Sqoop	Gateway host group Configurations for the services on the master node	Worker host group Data Node and YARN NodeManager	Compute group YARN NodeManager

Configurations

Note the following:

- There is a Hive Metastore Service (HMS) running in the cluster that talks to the same database instance as the Data Lake in the environment.
- If you use CLI to create the cluster, you can optionally pass an argument to create an external database for the cluster use such as Cloudera Manager, Oozie, Hue, and DAS. This database is by default embedded in the master node external volume. If you specify the external database to be of type HA or NON_HA, the database will be provisioned in the cloud provider. For all these types of databases the lifecycle is still associated with the cluster, so upon deletion of the cluster, the database will also be deleted.
- The HDFS in this cluster is for storing the intermediary processing data. For resiliency, store the data in the cloud object stores.
- For high availability requirements choose the Data Engineering High Availability cluster shape.

Architecture of the Data Engineering HA for AWS cluster

The Data Engineering HA for AWS and Azure cluster shape provides failure resilience for several of the Data Engineering HA services, including Knox, Oozie, HDFS, HS2, Hue, Livy, YARN, and HMS.

Services that do not yet run in HA mode include Cloudera Manager, DAS, and Zeppelin.

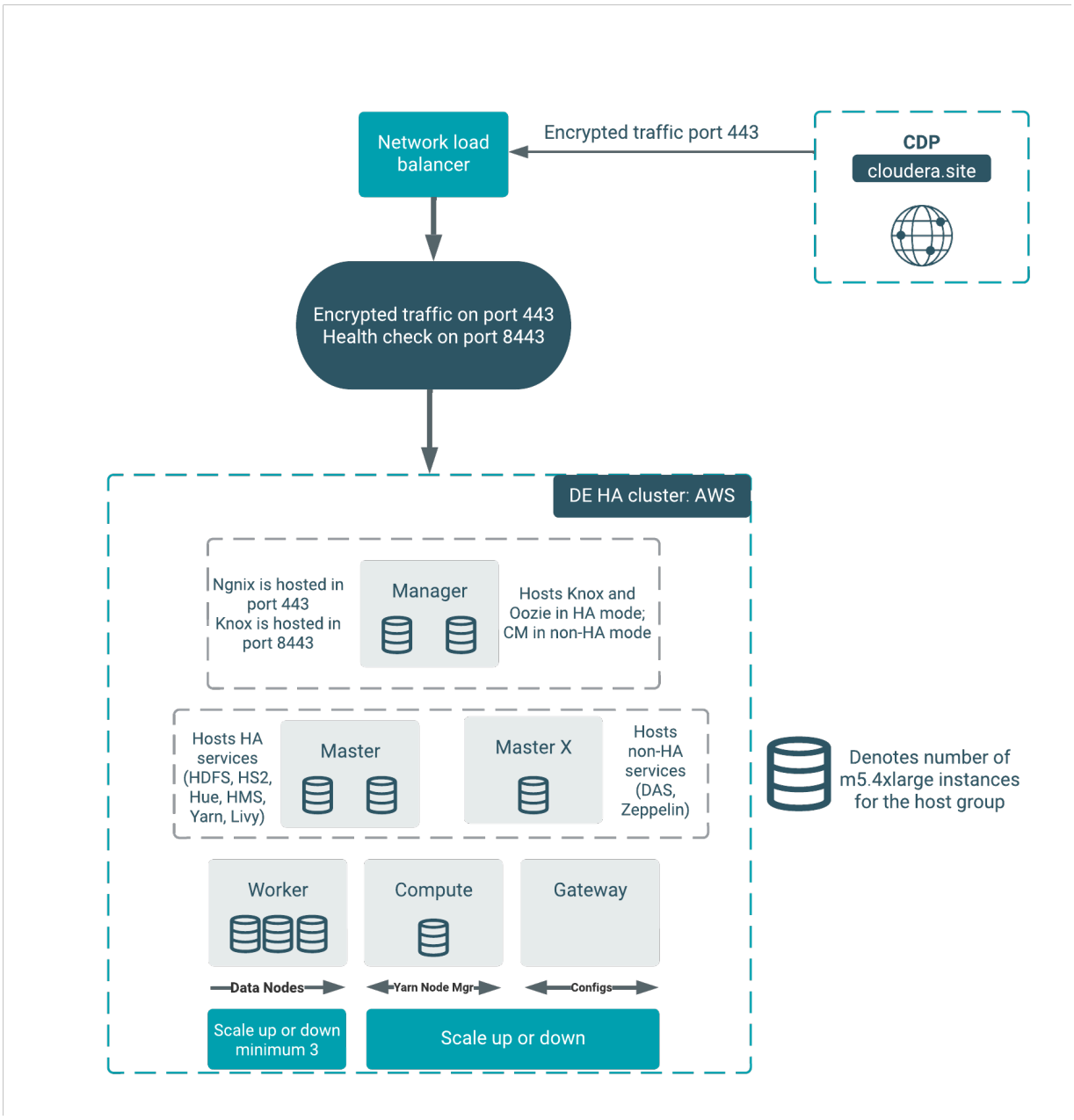

The architecture outlined in the diagram above handles the failure of one node in all of the host groups except for the “masterx” group. See the table below for additional details about the component interactions in failure mode:

Component	Failure	User experience
Knox	One of the Knox services is down	External users will still be able to access all of the UIs, APIs, and JDBC.
Cloudera Manager	The first node in manager host group is down	The cluster operations (such as repair, scaling, and upgrade) will not work.
Cloudera Manager	The second node in the manager host group is down	No impact.
HMS	One of the HMS services is down	No impact.
Hue	One of the Hue services is down in master host group	No impact.

HS2	One of the HS2 services is down in the master host group	External users will still be able to access the Hive service via JDBC. But if Hue was accessing that particular service it will not failover to the other host. The quick fix for Hue is to restart Hue to be able to use Hive functionality.
YARN	One of the YARN services is down	No impact.
HDFS	One of the HDFS services is down	No impact.
Nginx	Nginx in one of the manager hosts is down	Fifty percent of the UI, API, and JDBC calls will be affected. If the entire manager node is down, there is no impact. This is caused by the process of forwarding and health checking that is done by the network load-balancer.
Oozie	One of the Oozie servers is down in the manager host group.	No impact for AWS and Azure as of Cloudera Runtime version 7.2.11. If you create a custom template for DE HA, follow these two rules: **1.** Oozie must be in single hostgroup. **2.** Oozie and Hue must not be in the same hostgroup.

Important: If you are creating a DE HA cluster through the CDP CLI using the create-aws-cluster command, note that there is a CLI parameter to provision the network load-balancer in HA cluster shapes. Make sure to use the [--enable-load-balancer | --no-enable-load-balancer] parameter when provisioning a DE HA cluster via the CLI. For more information see the CDP CLI reference.

Custom templates

Any custom DE HA template that you create must be forked from the default templates of the corresponding version. You must create a custom cluster definition for this with the JSON parameter “enableLoadBalancers”: true , using the create-aws/azure/gcp-cluster CLI command parameter --request-template. Support for pre-existing custom cluster definitions will be added in a future release. As with the template, the custom cluster definition must be forked from the default cluster definition. You are allowed to modify the instance types and disks in the custom cluster definition. You must not change the placement of the services like Cloudera Manager, Oozie, and Hue. Currently the custom template is fully supported only via CLI.

The simplest way to change the DE HA definition is to create a custom cluster definition. In the Create Data Hub UI when you click Advanced Options, the default definition is not used fully, which will cause issues in the HA setup.

Related Information

HDFS
Hive
Hue
Oozie
Spark
YARN
Zookeeper

Data Mart clusters

Learn about the default Data Mart and Real Time Data Mart clusters, including cluster definition and template names, included services, and compatible Runtime version.

Data Mart is an MPP SQL database powered by Apache Impala designed to support custom Data Mart applications at big data scale. Impala easily scales to petabytes of data, processes tables with trillions of rows, and allows users to store, browse, query, and explore their data in an interactive way.

Data Mart clusters

The Data Mart template provides a ready to use, fully capable, standalone deployment of Impala. Upon deployment, it can be used as a standalone Data Mart to which users point their BI dashboards using JDBC/ODBC end points. Users can also choose to author SQL queries in Cloudera's web-based SQL query editor, Hue, and run them with Impala providing a delightful end-user focused and interactive SQL/BI experience.

Cluster definition names

- Data Mart for AWS

Cluster template name

Cloudera - Data Mart: Apache Impala, Hue

Included services

- HDFS
- Hue
- Impala

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17, 7.2.18

Real Time Data Mart clusters

The Real-Time Data Mart template provides a ready-to-use, fully capable, standalone deployment of Impala and Kudu. You can use a Real Time Data Mart cluster as a standalone Data Mart which allows high throughput streaming ingest, supporting updates and deletes as well as inserts. You can immediately query data through BI dashboards using JDBC/ODBC end points. You can choose to author SQL queries in Cloudera's web-based SQL query editor, Hue. Executing queries with Impala, you will enjoy an end-user focused and interactive SQL/BI experience. This template is commonly used for Operational Reporting, Time Series, and other real time analytics use cases.

Cluster definition names

- Real-time Data Mart for AWS

Cluster template name

Cloudera - Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark

Included services

- HDFS
- Hue
- Impala
- Kudu
- Spark 2
- Yarn

Compatible Cloudera Runtime versions

7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster definition names

- Real-time Data Mart - Spark3 for AWS

Cluster template name

Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark3

Included services

- HDFS

- Hue
- Impala
- Kudu
- Spark 3
- Yarn

Compatible Cloudera Runtime versions

7.2.16, 7.2.17, 7.2.18

High availability

Cloudera recommends that you use high availability (HA), and track any services that are not capable of restarting or performing failover in some way.

Impala HA

The Impala nodes offer high availability. The following Impala services are not HA.

- Catalog service
- Statestore service

Kudu HA

Both Kudu Masters and TabletServers offer high availability.

Related Information

HDFS
Hue
Impala
Kudu
Spark
YARN

Operational Database clusters

The Operational Database (OpDB) template is removed from the Cloudera Data Hub. You can access the Cloudera Operational Database instead as a superior product.

The Cloudera Operational Database is a NoSQL database powered by Apache HBase designed to support custom OLTP applications that want to leverage the power of BigData. Apache HBase is a NoSQL, scale-out database that can easily scale to petabytes and stores tables with millions of columns and billions of rows.

Cloudera Operational Database also contains Apache Phoenix which provides a way to use HBase through an SQL interface.

Cloudera recommends you to use the Cloudera Operational Database to create Operational Database clusters.

Related Information

Cloudera Operational Database
Getting started with Operational Database
Before you create an Operational Database cluster
Creating an Operational Database cluster
HDFS
HBase
Knox
Zookeeper
Phoenix

Streams Messaging clusters

Learn about the default Streams Messaging clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streams Messaging provides the following features:

- Advanced messaging and real-time processing on streaming data using Apache Kafka
- Centralized schema management using Schema Registry
- Management and monitoring capabilities powered by Streams Messaging Manager
- Cross-cluster Kafka topic replication using Streams Replication Manger
- Kafka partition rebalancing with Cruise Control

This template sets up a fault-tolerant standalone deployment of Apache Kafka and supporting Cloudera components (Schema Registry, Streams Messaging Manager, Streams Replication Manager and Cruise Control), which can be used for production Kafka workloads in the cloud or as a disaster recovery instance for on-premises. Kafka clusters.

Note:

Streams Messaging clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streams Messaging cluster deployment
- Creating your first Streams Messaging cluster
- Connecting Kafka clients to Cloudera Public Cloud clusters

Cluster definition names

- Streams Messaging Heavy Duty for AWS
- Streams Messaging Light Duty for AWS
- Streams Messaging HA for AWS

Cluster template name

- Cloudera - Streams Messaging Heavy Duty
- Cloudera - Streams Messaging Light Duty
- Cloudera - Streams Messaging High Availability

Included services

- Kafka
- Schema Registry
- Streams Messaging Manager
- Streams Replication Manager
- Cruise Control
- Kafka Connect

Compatible Cloudera Runtime version

- 7.1.0 (Preview)
- 7.2.0
- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11

- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streams Messaging cluster
Ingesting Data into Cloudera Public Cloud
Kafka
Schema Registry
Streams Messaging Manager
Streams Replication Manager

Flow Management clusters

Learn about the default Flow Management clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime versions.

Flow Management delivers high-scale data ingestion, transformation, and management to enterprises from any-to-any environment. It addresses key enterprise use cases such as data movement, continuous data ingestion, log data ingestion, and acquisition of all types of streaming data including social, mobile, clickstream, and IoT data.

The Flow Management template includes a no-code data ingestion and management solution powered by Apache NiFi. With NiFi's intuitive graphical interface and 300+ processors, Flow Management enables easy data ingestion and movement between CDP services as well as 3rd party cloud services. NiFi Registry is automatically set up and provides a central place to manage versioned Data Flows.

Note:

Flow Management clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Flow Management cluster deployment
- Creating your first Flow Management cluster
- Security considerations for Flow Management clusters
- Using Apache NiFi to ingest data into Cloudera Public Cloud
- Using NiFi and NiFi Registry

Cluster definition names

- Flow Management Light Duty for AWS
- Flow Management Heavy Duty for AWS

Cluster template name

- CDP - Flow Management: Light Duty
- CDP - Flow Management: Heavy Duty

Included services

- NiFi
- NiFI Registry

Compatible Cloudera Runtime versions

- 7.1.0
- 7.2.0

- 7.2.1
- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Flow Management cluster
Apache NiFi documentation
Apache NiFi Registry documentation

Streaming Analytics clusters

Learn about the default Streaming Analytics clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Streaming Analytics offers real-time stream processing and stream analytics with low-latency and high scaling capabilities powered by Apache Flink.

Streaming Analytics templates include Apache Flink that works out of the box in stateless or heavy state environments. Beside Flink, the template includes its supporting services namely YARN, Zookeeper and HDFS. The Heavy Duty template comes preconfigured with RocksDB as state backend, while Light Duty clusters use the default Heap state backend. You can create your streaming application by choosing between Kafka, Kudu, and HBase as datastream connectors.

You can also use SQL to query real-time data with SQL Stream Builder (SSB) in the Streaming Analytics template. By supporting the SSB service in Cloudera Public Cloud, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. SSB is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

Note:

Streaming Analytics clusters have distinct planning considerations and how-to information. See the Cloudera DataFlow for Data Hub documentation for information about:

- Planning your Streaming Analytics cluster deployment
- Creating your first Streaming Analytics cluster
- Analyzing data using Apache Flink
- Querying data using SQL Stream Builder

Cluster definition names

- Streaming Analytics Light Duty for AWS
- Streaming Analytics Heavy Duty for AWS

Cluster template name

- 7.2.17 - Streaming Analytics Light Duty
- 7.2.17 - Streaming Analytics Heavy Duty

Included services

- Flink
- SQL Stream Builder
- YARN
- Zookeeper
- HDFS
- Kafka

Important: In the Streaming Analytics cluster templates, Kafka service is included by default to serve as a background service only for the websocket ouput and sampling feature of SQL Stream Builder. The Kafka service in the Streaming Analytics cluster template cannot be used for production, you need to use the Streams Messaging cluster template when Kafka is needed for your deployment.

Compatible Cloudera Runtime version

- 7.2.2
- 7.2.6
- 7.2.7
- 7.2.8
- 7.2.9
- 7.2.10
- 7.2.11
- 7.2.12
- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Related Information

Setting up your Streaming Analytics cluster
Flink
YARN
Zookeeper
HDFS

Data Discovery and Exploration clusters

Learn about the default Data Discovery and Exploration clusters, including cluster definition and template names, included services, and compatible Cloudera Runtime version.

Data Discovery and Exploration

Explore and discover data sets ad-hoc. Do relevance-based analytics over unstructured data (logs, images, text, PDFs, etc). Get started with search or log analytics. Make data more accessible to everyone with Data Discovery and Exploration.

Cluster Definition Names

- Data Discovery and Exploration for AWS

Cluster Template Name

- Data Discovery and Exploration

Included Services

- Solr
- Spark 2
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Versions

7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, 7.2.14, 7.2.15, 7.2.16, 7.2.17

Cluster Definition Names

- Data Discovery and Exploration - Spark3 for AWS
- Data Discovery and Exploration - Spark3 for Azure
- Data Discovery and Exploration - Spark3 for Google Cloud

Cluster Template Name

- Data Discovery and Exploration for Spark3

Included Services

- Solr
- Spark 3
- HDFS
- Hue
- YARN
- ZooKeeper

Compatible Cloudera Runtime Version

7.2.18

Related Information

Solr
Spark
HDFS
Hue
YARN
Zookeeper

Create a cluster from a definition on AWS

You can quickly create clusters from default or custom cluster definitions within an existing AWS environment.

Before you begin

To create a Cloudera Data Hub cluster on AWS, you must have an existing AWS environment. Also, you should make sure that the Runtime version of the Data Lake cluster matches the Cloudera Runtime version of the Cloudera Data Hub cluster that you are about to create; If these versions don't match, you may encounter warnings and/or errors.

About this task

Required role: DataHubCreator, EnvironmentAdmin at the scope of the environment where the Cloudera Data Hub cluster is running, or Owner of the environment

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Environments > click on an environment where you would like to create a cluster > click Create Data Hub. The following page is displayed:

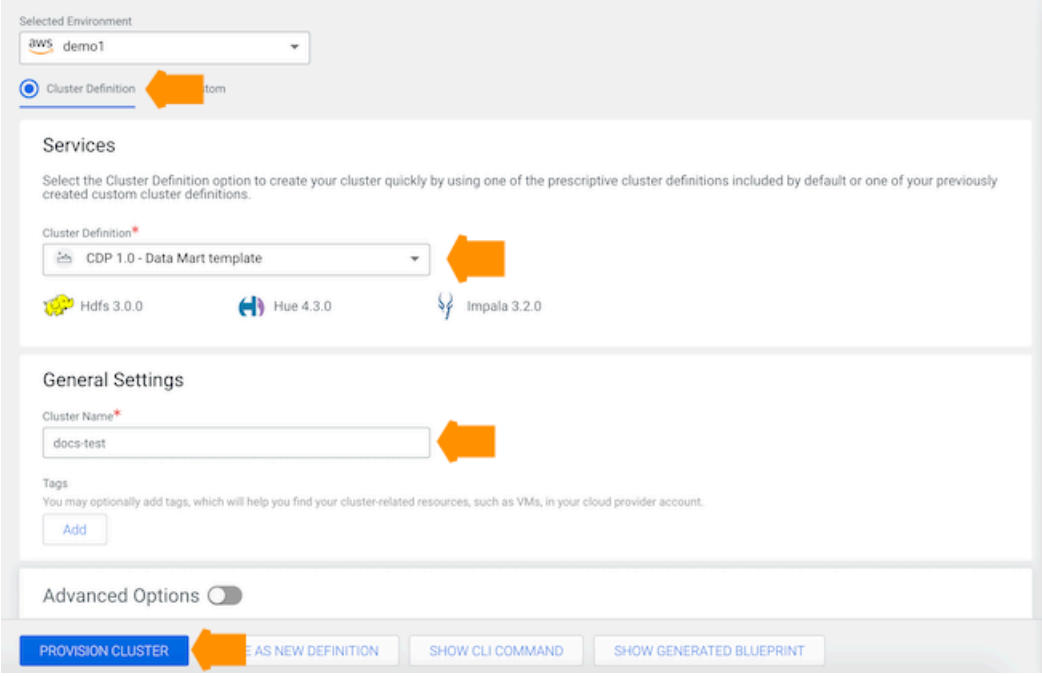

3. Select Cluster Definition.
4. From the Cluster Definition dropdown, select the cluster definition that you would like to use for your cluster.

   The cluster template referenced in the selected cluster definition determines which services are included in the cluster. The list of services is automatically shown below the selected cluster definition name:
5. Specify General Settings for your cluster:

Parameter	Description
Cluster Name	Enter a name for your cluster. The name must be between 5 and 40 characters, must start with a letter, and must only include lowercase letters, numbers, and hyphens.
Tags	(Optional) Add tags that Data Hub should use to tag your AWS resources. Click Add to add a tag, and then enter a key and value for each tag. Repeat the steps if you would like to add more tags. For more information about tags, refer to Tags.

6. Optionally, click on Advanced Options to modify advanced cluster settings. For more information on these options, refer to *Advanced cluster options*.
7. On AWS and Azure only: Optionally, when you have finished providing the cluster settings, you can click the Show CLI Command button at the bottom of the page to review or copy the CDP CLI command used to create the cluster. You can copy the command from the pop-up window that appears, either to provision the cluster later or for use in scripts.
8. You also have the option to review or copy the cluster template that is generated and will be used in cluster creation. Click the Show Generated Cluster Template button at the bottom of the page.
9. To proceed with cluster provisioning immediately, click on Provision Cluster.
10. You will be redirected to the Cloudera Data Hub cluster dashboard, and a new tile representing your cluster will appear at the top of the page.

    The following messages are written to the event history as your cluster is being created:

```
Cluster built; Cluster manager ip:10.97.82.237
8/2/2019, 6:10:44 PM
```

```
Updating Cluster Proxy service with gateway configuration
8/2/2019, 6:10:43 PM
Building cluster; Cluster manager ip:10.97.82.237
8/2/2019, 5:45:12 PM
Starting cluster services
8/2/2019, 5:42:33 PM
Mounting attached disks
8/2/2019, 5:42:18 PM
Setting up infrastructure metadata
8/2/2019, 5:42:14 PM
Bootstrapping infrastructure cluster
8/2/2019, 5:41:59 PM
Registering cluster with Cluster Proxy service
8/2/2019, 5:41:57 PM
Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Billing started, Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Infrastructure metadata collection finished
8/2/2019, 5:41:55 PM
Infrastructure creation took 96 seconds
8/2/2019, 5:41:52 PM
Creating infrastructure
8/2/2019, 5:40:16 PM
Setting up CDP image
8/2/2019, 5:40:15 PM
```

11. When your cluster is ready, its status will change to Running.

What to do next

You can access links to Cloudera Manager, and other cluster UIs and endpoints from cluster details.

Create a custom cluster on AWS

Create a custom Cloudera Data Hub cluster within an existing AWS environment.

Before you begin

To create a Cloudera Data Hub cluster on AWS, you must have an existing AWS environment. Also, you should make sure that the Cloudera Runtime version of the Data Lake cluster matches the Runtime version of the Cloudera Data Hub cluster that you are about to create; If these versions don't match, you may encounter warnings and/or errors.

https://www.youtube.com/embed/pGHHW9nOrPU

About this task

Required role: DataHubCreator, EnvironmentAdmin at the scope of the environment where the Data Hub is running, or Owner of the environment

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console > Environments > click on an environment > click Create Data Hub.
3. Under Selected Environment, confirm that the selected environment is the one where you would like to create your cluster.
4. Select Custom.
5. Under Platform Version, current Cloudera Runtime version is pre-selected.

6. Under Cluster template, select the cluster template that you would like to use.

   The selected cluster template determines which services are included in the cluster. Select the cluster template and the list of services is automatically shown below it:

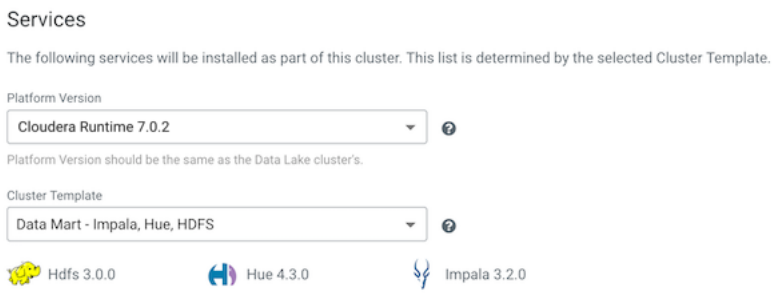

   For more information about cluster templates, refer to Cluster templates.

7. Specify General Settings for your cluster:

Parameter	Description
Cluster Name	Enter a name for your cluster. The name must be between 5 and 40 characters, must start with a letter, and must only include lowercase letters, numbers, and hyphens.
Tags	(Optional) Add tags that Data Hub should use to tag your AWS resources. Click Add to add a tag, and then enter a key and value for each tag. Repeat the steps if you would like to add more tags. For more information about tags, refer to Tags.

8. Optionally, click on Advanced Options to modify advanced cluster settings. For more information on these options, refer to LINK.
9. Once done, click on Provision Cluster.
10. You will be redirected to the Cloudera Data Hub cluster dashboard, and a new tile representing your cluster will appear at the top of the page.

    The following messages are written to the event history as your cluster is being created:

```
Cluster built; Cluster manager ip:10.97.82.237
8/2/2019, 6:10:44 PM
Updating Cluster Proxy service with gateway configuration
8/2/2019, 6:10:43 PM
Building cluster; Cluster manager ip:10.97.82.237
8/2/2019, 5:45:12 PM
Starting cluster services
8/2/2019, 5:42:33 PM
Mounting attached disks
8/2/2019, 5:42:18 PM
Setting up infrastructure metadata
8/2/2019, 5:42:14 PM
Bootstrapping infrastructure cluster
8/2/2019, 5:41:59 PM
Registering cluster with Cluster Proxy service
8/2/2019, 5:41:57 PM
Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Billing started, Infrastructure successfully provisioned
8/2/2019, 5:41:56 PM
Infrastructure metadata collection finished
8/2/2019, 5:41:55 PM
Infrastructure creation took 96 seconds
8/2/2019, 5:41:52 PM
Creating infrastructure
8/2/2019, 5:40:16 PM
Setting up CDP image
```

```
8/2/2019, 5:40:15 PM
```

11. When your cluster is ready, its status will change to Running.

What to do next

You can access links to Cloudera Manager, and other cluster UIs and endpoints from cluster details.

Advanced cluster options

In the create cluster wizard, click on Advanced Options to view the advanced cluster configuration options.

https://www.youtube.com/embed/Q8Xjb3-8KFk

While some of these advanced options can be configure in the wizard, others require prior setup.

The following options are available:

Tags

You can define tags that will be applied to your cluster-related resources (such as VMs) on your cloud provider account.

The tags added during cluster creation are displayed in your cloud account on the resources that Cloudera Data Hub provisioned for your clusters. You can use tags to categorize your cloud resources by purpose, owner, and so on. Tags come in especially handy when you are using a corporate cloud provider account and you want to quickly identify which resources belong to your cluster(s). In fact, your corporate cloud account admin may require you to tag all the resources that you create, in particular resources, such as VMs, which incur charges.

By default, the following tags are created:

Tag	Description
Cloudera-Resource-Name	The workload-appropriate Cloudera resource name. This CRN serves as a unique identifier for the resource over time.
Cloudera-Creator-Resource-Name	Cloudera resource name of the Cloudera user that created the resource.
Cloudera-Environment-Resource-Name	The name of the environment with which the resource is associated.

You can optionally add additional tags. To add custom tags:

1. In the create cluster wizard, navigate to the General Configuration page.
2. Specify your tags in the Tags section by providing a key and value for each tag.

Note:

It is not possible to add tags via Cloudera Data Hub after your cluster has been created. In this case, you can only add the tags manually via your cloud provider's interface.

To learn more about tags and their restrictions, refer to AWS documentation.

Image catalog

The options on the "Image Settings" page of the advanced create cluster wizard allow you to select custom image settings.

By default, Cloudera Data Hub uses the prewarmed image from the image catalog provided in Cloudera Data Hub. If necessary, you can also customize a default image.

Choose image catalog

Cloudera Data Hub uses the image catalog provided by default. If you would like to use a custom image catalog instead of the default image catalog, you must first prepare your custom images, then create and register an image catalog.

Choose image type

By default, Cloudera Data Hub uses the included prewarmed images with the default Cloudera Manager and Cloudera Runtime version, but you can select a different prewarmed image or customized prewarmed image to use for your cluster. Cloudera Data Hub currently supports the following types of images for launching clusters:

Image type	Description	Default images provided
Prewarmed Image	By default, Cloudera Data Hub launches clusters from prewarmed images. Prewarmed images include the operating system as well as Cloudera Manager and Cloudera Runtime. The Cloudera Manager and Cloudera Runtime version used on prewarmed images cannot be customized.	Yes
Custom Image	You can customize a default image for compliance or security reasons. You can then use the CDP CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create a Cloudera Data Hub cluster.	

Choose image

This option allows you to select a different image.

Custom images and image catalogs

If necessary, you can use a custom Cloudera Runtime or FreeIPA image for compliance or security reasons. You can then use the Cloudera CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create a Data Lake, Cloudera Data Hub cluster or environment with a custom FreeIPA image.

Overview

A custom image should inherit most of its attributes from its source image, which is a default image that you select from the cdp-default image catalog.

The typical method of creating a Data Lake or Cloudera Data Hub picks up the latest pre-warmed image from the cdp-default image catalog for the specified version of Cloudera Runtime. These default images are pre-warmed VM images that contain a base URL to the default parcels in the Cloudera archive, amongst other configurations. If the default pre-warmed images do not suit your business needs, you can specify that the Data Lake, Cloudera Data Hub cluster or the environment (in the case of FreeIPA) uses a custom image instead.

What is a custom image?

A custom image is an entry in a custom image catalog that inherits most of its attributes from a source (default) image.

Custom image entries have:

- An image type: Cloudera Runtime [which includes Cloudera Data Hub and Data Lake images] or FreeIPA
- A source image ID that points to an image in the cdp-default image catalog

- A timestamp of creation
- An option to specify a VM region and image reference (such as an AMI ID) if you are overriding the source image with a custom VM image
- An option to override the parcel base URL

Why use a custom image?

You might require a custom image for compliance or security reasons (a “hardened” image), or to have your own packages pre-installed on the image, for example monitoring tools or software. You might also want to specify a custom image if you need to use a default image with a specific Runtime maintenance version applied, rather than simply specifying the latest major Runtime version.

Note: When customizing VM images, note that certain customizations (for example, CIS hardening rules) may not be compatible with Cloudera.

What can you customize?

In a custom image entry, you can override the VM images themselves with your own custom images that are sufficiently hardened. Importantly, you should only customize a default image from the cdp-default catalog as opposed to creating one from scratch. You can also override the default parcel base URL (at archive.cloudera.com) with your own host site.

What is a custom image catalog?

A custom image catalog is simply a catalog that holds custom images. A custom image catalog can contain a single or multiple custom image entries.

Custom image catalogs have:

- A name. The name is a unique identifier and is used to refer to the catalog during environment, Data Lake, and Cloudera Data Hub cluster creation; as well as during catalog operations like creating an image.
- A description.
- An owner. The owner is the user who runs the command to create the catalog.

What is the process for creating a custom image and catalog?

- If you are replacing the VM images in a custom image entry with a customized version, you should first prepare the image by modifying an official Cloudera default image, which you can find under Shared Resources > Image Catalogs > cdp-default.
- Select a source image from the cdp-default image catalog to be the source of customization. When you run the CLI command to find a default image, you specify the Runtime version, provider, image type, or a combination of the three.
- Create a custom image catalog, or identify an existing catalog where you want to save the custom image entry.
- Apply the necessary changes to the custom image entry, like the override AMI IDs with the new, customized AMIs; or add a new parcel base URL using the `--base-parcel-url` command when you set the custom image.
- You can then create an environment, Data Lake, or Cloudera Data Hub cluster, based on custom catalogs via the CDP CLI.

Creating a custom image and image catalog with the CDP CLI

You can create a custom Cloudera Runtime or FreeIPA image and image catalog through the CDP CLI.

Before you begin

If you are replacing the VM images in a custom image entry with a customized version, you must first prepare the image by modifying an official Cloudera default image, which you can find under Shared Resources Image Catalogs cdp-default . Take note of the image reference, such as the AMI ID.

Procedure

1. To find a source image from the cdp-default catalog that you want to use as the source of your custom image entry, run the following command:

```
cdp imagecatalog find-default-image --provider <cloud provider> --image-
type <image type> --runtime-version <Cloudera Runtime version>
```

For example:

```
cdp imagecatalog find-default-image --provider AWS --image-type runtime --
runtime-version 7.2.12
```

2. A custom image requires a custom catalog. If you haven't yet created a custom catalog, or if you want to create a new one for a new custom image, run the following command:

```
cdp imagecatalog create-custom-catalog --catalog-name <unique catalog na
me> --description <catalog description>
```

For example:

```
cdp imagecatalog create-custom-catalog --catalog-name my custom catalog --
description test catalog
```

3. Within the custom image catalog that you created (or an older custom catalog that you want to use), create a custom image entry with the selected source image marked as its source. Providing your own VM images is optional. Run the following command:

```
cdp imagecatalog set-<image-type>-image --catalog-name <name of the custom
 catalog> --vm-images region=<region of customized image>,imageReference=<
cloud provider specific ID of a customized image> --source-image-id <cdp
 image ID of source image>
```

For example:

```
cdp imagecatalog set-runtime-image --catalog-name my custom catalog --vm-
images region=eu-central-1,imageReference=ami-7torotmhqi6q7438y --source-
image-id 8t4y9853-12b6-3n6z-75dh-tx775k4c793w
```

4. You can then apply the necessary changes to the custom image entry, like overriding AMI IDs with new, customized ones, or adding a new parcel base URL.

Command	Description
cdp imagecatalog find-default-image	Finds the default images in the cdp-default image catalog for a specified version of Cloudera Runtime. Parameters: • --provider • --image-type • --runtime-version You can provide any combination of these parameters.
cdp imagecatalog create-custom-catalog	Creates a custom catalog. Parameters: • --catalog-name • --description

Command	Description
	–catalog-name is required.
cdp imagecatalog set-runtime-image or cdp imagecatalog set-freeipa-image	Creates a custom image entry (either Cloudera Runtime [Cloudera Data Hub/Data Lake] or FreeIPA) within the specified catalog. Parameters: • --catalog-name • --vm-images [region,imageReference] • --source-image-id • --image-id • --base-parcel-url --catalog-name and --source-image-id are required.

Switching image catalogs

You can switch the image catalog of an already existing Cloudera Data Hub, Data Lake, or FreeIPA cluster. You may want to switch the image catalog for a cluster in order to restrict which Cloudera Runtime version can be upgraded to, or in order to move to custom images for an existing cluster.

Use the following CDP CLI commands to switch the image catalog for an existing cluster:

- FreeIPA:

```
cdp environments set-catalog --environment $ENVIRONMENT_NAME --catalog $
CATALOG
```

Parameter	Description
--environment	Name or CRN of the environment that holds the FreeIPA installation.
--catalog	URL of the FreeIPA catalog to be used.

- Cloudera Data Hub:

```
cdp datahub set-catalog --cluster $DATAHUB_NAME --catalog-name $CATALOG
```

Parameter	Description
--cluster	Name or CRN of the Cloudera Data Hub cluster for which you want to use the new image catalog.
--catalog-name	Name of the image catalog to be used. The image catalog must be a JSON based catalog, and switching is only possible from catalogs that are JSON based.

- Data Lake:

```
cdp datalake set-catalog --datalake $DATALAKE_NAME --catalog-name $CATALOG
```

Parameter	Description
--datalake	Name of the Data Lake for which you want to use the new image catalog.

Parameter	Description
--catalog-name	Name of the image catalog to be used. The image catalog must be a JSON based catalog, and switching is only possible from catalogs that are JSON based.

Network and availability

The "Network and availability" options allow you to customize the networking settings of your cluster.

On the Network and Availability page, provide the following to specify the networking resources that will be used for your cluster:

Parameter	Description
Select Subnet	If your environment includes a single subnet, that subnet is preselected and it cannot be changed. If your environment includes more than one subnet, you can select the subnet(s) in which your cluster will be provisioned. If you would like to deploy your Cloudera Data Hub cluster in multiple availability zones, you should select multiple subnets.
Select Azure Database Server	A Cloudera Data Hub cluster uses the same Flexible Server or Single Server settings as the environment in which it runs but you can choose to enable a Flexible server on Cloudera Data Hub clusters running in an environment that uses a Single server. For more information, see Using Azure Database for PostgreSQL Flexible Server.

Hardware and storage

The "Hardware and storage" options allow you to customize the cloud provider specific cluster hardware and storage options.

The Hardware and Storage options can be selected for each host group. To edit this section for a specific host group, click on the . When done editing, click on the to save the changes. Repeat for these steps for all host groups that you would like to edit.

The following hardware and storage settings are available:

Parameter	Description
Cloudera Manager Server	You must select one node for Cloudera Manager Server by clicking the button. The “Instance Count” for that host group must be set to “1”. If you are using one of the default cluster templates, this is set by default.
Instance Type	Select an instance type. For information about instance types on AWS refer to Amazon EC2 Instance Types in AWS documentation.
Instance Count	Enter the number of instances of a given type. Default is 1.
Storage Type	Select the volume type. The options vary by instance type and include: (1) Ephemeral (2) Magnetic (3) General Purpose SSD, (4) Throughput Optimized HDD. For more information about these options refer to Amazon EC2 Instance Store in AWS documentation. **Note:** Stopping and restarting Cloudera Data Hub clusters using ephemeral storage is not supported.
Encryption	Under Encryption Key, you can select an existing encryption key. For more information, refer to EBS Encryption on AWS.
Attached Volumes Per Instance	Enter the number of volumes attached per instance. Default is 1.

Parameter	Description
Volume Size	Enter the size in GB for each volume. Default is 100.
Root Volume Size	This option allows you to increase or decrease the root volume size. Default is 200 GB. This option is useful if your custom image requires more space than the default 200 GB. If you use a custom Cloudera Data Hub template specifying a root volume size smaller than 200GB, you may encounter an error.

Cloud storage

The options on the "Cloud Storage" page allow you to optionally specify the base storage location used for YARN and Zeppelin.

During environment creation under Data Access > Storage Location Base you configure a default S3 base storage location for the environment, and all Cloudera Data Hub clusters created within that environment use this location. The Cloud Storage options in the Cloudera Data Hub cluster wizard allow you to additionally specify a different location for YARN application logs and Zeppelin Notebook's root directory:

- Existing Base Storage Location - By default, this is set to the Storage Location Base configured on environment level. If you do not want to make any changes, simply leave this blank. If you would like to use a different location for YARN application logs and Zeppelin Notebook's root directory, you can specify a different S3 location. Note that the specified S3 location must exist prior to Cloudera Data Hub cluster creation and that you must adjust the IAM policies created during environment's cloud storage setup to make sure that IDBroker has write access to this location.
- Path for YARN Application Logs property - This directory structure gets created automatically during cluster creation. You can customize it if you would like it be different than what is suggested by default.
- Path for Zeppelin Notebooks Root Directory property - This directory structure gets created automatically during cluster creation. You can customize it if you would like it to be different than what is suggested by default.

Any S3 bucket that you designate for Cloudera Data Hub cloud storage on AWS must be in the same region as the environment.

Building a custom cluster template

You can build a custom cluster template to modify the cluster Cloudera Runtime services, including the Cloudera Runtime configuration properties and the distribution of Runtime services across host groups. To create a custom template, modify an existing default cluster template and then upload and register the custom template.

About this task

If the default cluster templates are insufficient for the cluster that you want to create, you can build a custom cluster template. The recommended method for building a cluster template is to modify an existing default template, so that the structure of the template is mostly preserved.

Required role: EnvironmentCreator can create a shared resource and then assign users to it. SharedResourceUser or Owner of the shared resource can use the resource. The Owner of the shared resource can delete it.

Procedure

1. Review information on the default cluster configurations to find one that includes services suitable for the type of cluster that you want to create. In general, it is best to use the templates for the current release.
2. To access the existing default cluster templates, click Shared ResourcesCluster Templates.
3. To find the newest template versions, click Platform at the top of the Platform column to sort the templates in descending order.

4. Under the Name column, click the desired default ("Built In") template.

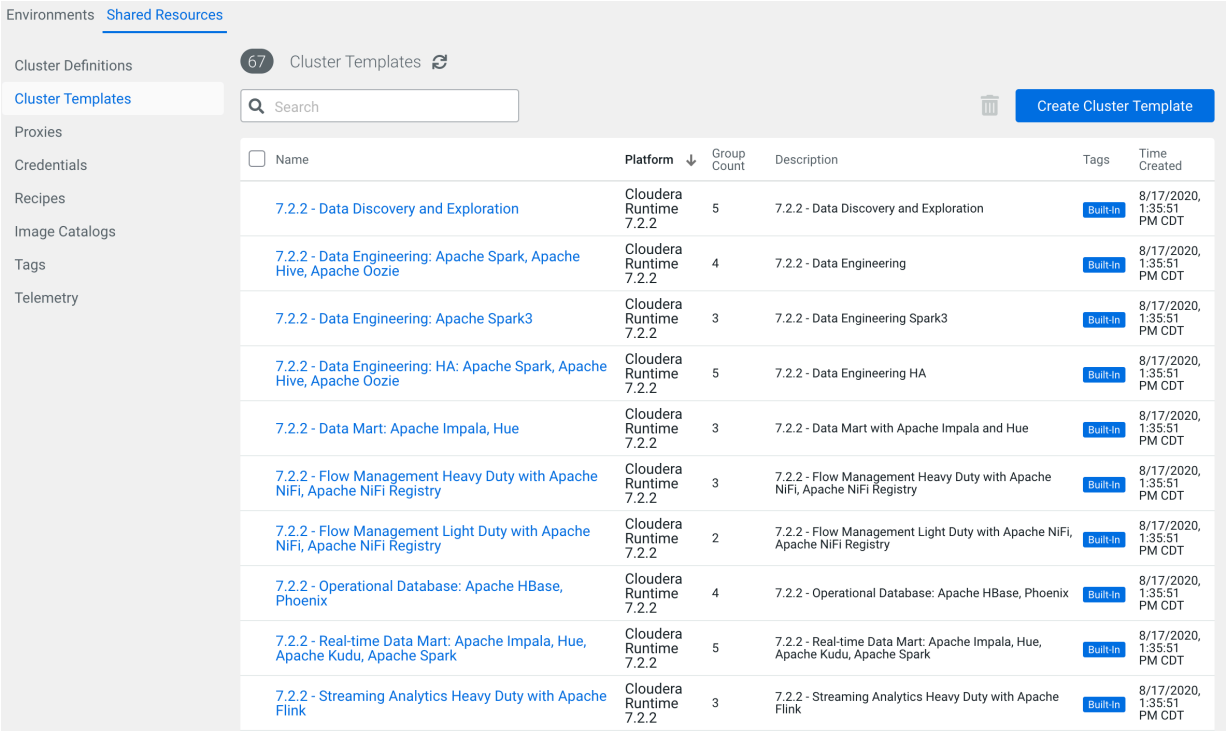

The template opens in LIST view, which shows how the template is structured across host groups.

5. Click RAW VIEW to view the JSON structure.

Environments / List / Cluster Templates / 7.2.2 - Data Engineering: A...

Environments Shared Resources

Cluster Templates
Proxies
Credentials
Recipes
Image Catalogs

7.2.2 - Data Engineering: Apache Spark, Apache Hive, Apache Oozie
7.2.2 - Data Engineering

Delete

LIST VIEW RAW VIEW


```
{
  "cdhVersion": "7.2.2",
  "displayName": "dataengineering",
  "services": [
    {
      "refName": "zookeeper",
      "serviceType": "ZOOKEEPER",
      "serviceConfigs": [
        {
          "name": "service_config_suppression_server_count_validator",
          "value": "true"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "zookeeper-SERVER-BASE",
          "roleType": "SERVER",
          "base": true
        }
      ]
    },
    {
      "refName": "hdfs",
      "serviceType": "HDFS",
      "serviceConfigs": [
        {
          "name": "hdfs_verify_ec_with_topology_enabled",
          "value": false
        },
        {
          "name": "core_site_safety_valve",
          "value": "<property><name>fs.s3a.buffer.dir</name><value>${env.LOCAL_DIRS:-${hadoop.tmp.dir}}/s3a</value></property><property>
<name>fs.s3a.committer.name</name><value>directory</value></property>"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "hdfs-NAMENODE-BASE",
          "roleType": "NAMENODE",
          "base": true,
          "configs": [
            {
              "name": "role_config_suppression_fs_trash_interval_minimum_validator",
              "value": "true"
            },
            {
              "name": "role_config_suppression_namenode_java_heapsize_minimum_validator",
              "value": "true"
            },
            {
              "name": "fs_trash_interval",
              "value": "0"
            },
            {
              "name": "fs_trash_checkpoint_interval",
              "value": "0"
            },
            {
              "name": "erasure_coding_default_policy",
              "value": " "
            }
          ]
        },
        {
          "refName": "hdfs-SECONDARYNAMENODE-BASE",
          "roleType": "SECONDARYNAMENODE",
          "base": true
        },
        {
          "refName": "hdfs-DATANODE-BASE",
          "roleType": "DATANODE",
          "base": true
        },
        {
          "refName": "hdfs-BALANCER-BASE",
          "roleType": "BALANCER",
          "base": true
        },
        {
          "refName": "hdfs-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "base": true,
          "configs": [
            {
              "name": "dfs_client_use_trash",
              "value": false
            },
            {
              "name": "role_config_suppression_hdfs_trash_disabled_validator",
              "value": "true"
            },
            {
              "name": "hdfs_client_env_safety_valve",
              "value": "HADOOP_OPTS=\"-Dorg.wildfly.openssl.path=/usr/lib64 ${HADOOP_OPTS}\""
            }
          ]
        }
      ]
    },
    {
      "refName": "yarn",
      "serviceType": "YARN",
      "serviceConfigs": [
        {
          "name": "yarn_admin_acl",
          "value": "yarn,hive,hdfs,mapred"
        },
        {
          "name": "yarn_service_mapred_safety_valve",
          "value": "<property><name>mapreduce.fileoutputcommitter.algorithm.version</name><value>1</value></property><property>
<name>mapreduce.input.fileinputformat.list-status.num-threads</name><value>100</value></property>"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "yarn-RESOURCEMANAGER-BASE",
          "roleType": "RESOURCEMANAGER",
          "base": true,
          "configs": [
            {
              "name": "resourcemanager_config_safety_valve",
              "value": "<property><name>yarn.scheduler.configuration.store.class</name><value>zk</value></property>"
            },
            {
              "name": "yarn_resourcemanager_scheduler_class",
              "value": "org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler"
            },
            {
              "name": "yarn_scheduler_capacity_resource_calculator",
              "value": "org.apache.hadoop.yarn.util.resource.DefaultResourceCalculator"
            },
            {
              "name": "resourcemanager_capacity_scheduler_configuration",
              "value": "<configuration><property><name>yarn.scheduler.capacity.root.queues</name><value>default</value></property><property>
<name>yarn.scheduler.capacity.root.capacity</name><value>100</value></property><property><name>yarn.scheduler.capacity.root.default.capacity</name>
<value>100</value></property><property><name>yarn.scheduler.capacity.root.acl_submit_applications</name><value> </value></property><property>
<name>yarn.scheduler.capacity.root.acl_administer_queue</name><value> </value></property><property>
<name>yarn.scheduler.capacity.root.default.acl_submit_applications</name><value>*</value></property><property>
<name>yarn.scheduler.capacity.root.default.minimum-user-limit-percent</name><value>100</value></property><property>
<name>yarn.scheduler.capacity.maximum-am-resource-percent</name><value>0.33</value></property><property><name>yarn.scheduler.capacity.node-
locality-delay</name><value>0</value></property><property><name>yarn.scheduler.capacity.schedule-asynchronously.maximum-threads</name>
<value>1</value></property><property><name>yarn.scheduler.capacity.schedule-asynchronously.scheduling-interval-ms</name><value>10</value>
</property></configuration>"
            }
          ]
        },
        {
          "refName": "yarn-NODEMANAGER-WORKER",
          "roleType": "NODEMANAGER",
          "base": false
        },
        {
          "refName": "yarn-NODEMANAGER-COMPUTE",
          "roleType": "NODEMANAGER",
          "base": false
        },
        {
          "refName": "yarn-JOBHISTORY-BASE",
          "roleType": "JOBHISTORY",
          "base": true
        },
        {
          "refName": "yarn-GATEWAY-BASE",
```

6. Select all of the JSON code, copy it, and paste it to a suitable code editor. Standard text editors are not recommended.
7. If you are using a code editor such as Microsoft Visual Studio Code, you can use built-in tools to validate the JSON before you proceed. Optionally, you can save the file as-is (without having made any changes), and upload it using the instructions in the topic *Upload a cluster template*. To verify that the template JSON is functional, you can create a test cluster by selecting the template that you just registered, and see if the cluster successfully deploys.
8. When you are satisfied that you are working with a clean template, you can begin to modify the template. Return to the JSON template file in your code editor.

   Each default template consists of two main sections: the services section and the hostTemplates section. The serv ices section includes the components that make up the cluster. This is where you can add or remove services, as well as modify service configuration properties. If you want to modify a service's configuration, for example to tune Yarn or Hive, refer to the Cloudera Manager configuration properties for the desired service. You can search these properties by their API Name, which is how they appear in a Cloudera Data Hub template.

   For example, in a Data Engineering template you might want to adjust the amount of physical memory allocated for containers by configuring the yarn.nodemanager.resource.memory-mb property. If you want to configure this property to 80% of the total system RAM, for a 256 GB machine this would look like:

```
{
      "refName": "yarn",
      "serviceType": "YARN",
      "serviceConfigs": [
        {
          "name": "yarn_admin_acl",
          "value": "yarn,hive,hdfs,mapred"
        }
      ],
      "roleConfigGroups": [
        {
          "refName": "yarn-RESOURCEMANAGER-BASE",
          "roleType": "RESOURCEMANAGER",
          "base": true,
          "configs": [
            {
              "name": "yarn.nodemanager.resource.memory-mb",
              "value": 2052096
            },
```

   If you want to add a service in a template, the simplest method is to find the service in the RAW VIEW of another default template and copy it into your JSON.

   For example, if you want to add Sqoop to the services in a template, copy it from the Data Engineering or Data Engineering HA template into the services section of another template:

```
{
      "refName": "sqoop",
      "serviceType": "SQOOP_CLIENT",
      "roleConfigGroups": [
        {
          "refName": "sqoop-SQOOP_CLIENT-GATEWAY-BASE",
          "roleType": "GATEWAY",
          "configs": [],
          "base": true
        }
      ]
```

```
    },
```

The hostTemplates section of the JSON file describes the nodes by their type and the services on the node. This section also includes a cardinality parameter, which you can set to increase or decrease the quantity of that specific node type.

For example, say that you want to create a new node type called "ZKserver" that runs a single service, Zookeeper. Assuming that Zookeeper is already a service defined in the services section of the template, you can move down to the hostTemplate section. In the master node section of this Data Engineering template, you can see that Zookeeper is already defined in the "master" node section with the string "zookeeper-SERVER-BASE":

```
"hostTemplates": [
    {
      "refName": "master",
      "cardinality": 1,
      "roleConfigGroupsRefNames": [
        "hdfs-BALANCER-BASE",
        "hdfs-NAMENODE-BASE",
        "hdfs-SECONDARYNAMENODE-BASE",
        "hdfs-GATEWAY-BASE",
        "hms-GATEWAY-BASE",
        "hms-HIVEMETASTORE-BASE",
        "hive_on_tez-HIVESERVER2-BASE",
        "hive_on_tez-GATEWAY-BASE",
        "hue-HUE_LOAD_BALANCER-BASE",
        "hue-HUE_SERVER-BASE",
        "tez-GATEWAY-BASE",
        "spark_on_yarn-GATEWAY-BASE",
        "spark_on_yarn-SPARK_YARN_HISTORY_SERVER-BASE",
        "livy-LIVY_SERVER-BASE",
        "zeppelin-ZEPPELIN_SERVER-BASE",
        "oozie-OOZIE_SERVER-BASE",
        "sqoop-SQOOP_CLIENT-GATEWAY-BASE",
        "yarn-JOBHISTORY-BASE",
        "yarn-RESOURCEMANAGER-BASE",
        "zookeeper-SERVER-BASE",
        "das-DAS_WEBAPP",
        "das-DAS_EVENT_PROCESSOR",
        "yarn-QUEUEMANAGER_WEBAPP-BASE",
        "yarn-QUEUEMANAGER_STORE-BASE",
        "yarn-GATEWAY-BASE"
      ]
```

To create our new ZKserver node, you can copy the standard node format and modify it for your purpose:

```
{
      "refName": "ZKserver",
      "cardinality": 1,
      "roleConfigGroupsRefNames": [
        "zookeeper-SERVER-BASE"
      ]
    }
```

If you also want to include dynamic parameters in your custom template, see the documentation for Dynamic cluster templates. During the cluster creation phase, dynamic parameters pick up the parameter values that you provided in the Cloudera Data Hub cluster wizard. See the *Custom Properties* documentation for a list of properties that can be dynamically replaced. You might want to use dynamic parameters when you regularly provision clusters using a specific template, but want to change a few of the property values each time you provision a new cluster.

9. When you have finished modifying the template, validate the JSON in your code editor and save the template.

What to do next

Upload the JSON file and register the template following the instructions in the *Upload a cluster template* topic. Then, when you navigate to the Data Hubs page and select Create Data Hub, be sure to select the Custom radio button underneath environment selection. Here you can provision a Cloudera Data Hub cluster using the custom template that you registered. Select your custom template from the drop-down menu before you configure any advanced options.

Related Information

Upload a cluster template

CDP Public Cloud upgrade advisor

Compute resources deployed via cloud services are generally considered transient in nature. With the separation of compute and storage, compute clusters in CDP can be stopped or decommissioned, while the data used or created by workloads running on these clusters generally remains accessible on persistent cloud storage.

There are some exceptions to the above, most importantly SDX metadata and cloud service-specific metadata. SDX metadata is stored in the Data Lake, while metadata specific to a cloud service may be stored in databases, local configurations, or even locally attached disks. This local storage can also be persistent (block storage volumes) or transient (ephemeral disks).

In cloud services where compute is elastic and state is transient, we need safeguards to protect all data that is not persistent, especially when changes are performed on the services themselves. In CDP Public Cloud, there can be several changes, most notably Runtime maintenance upgrades, Runtime minor/major software version upgrades, and OS upgrades.

In general, there are two main approaches to upgrading cloud services:

1. Backup, re-create and restore
2. In-place upgrade

These two approaches have similarities: a prior backup should be performed, service endpoints should remain stable after the operation, and they should result in the same outcome. CDP Public Cloud supports both approaches. While the first approach may be convenient for simple data workloads, complex data applications and their custom data pipelines spanning multiple clusters may require an in-place upgrade path.

In this guide we will describe the high-level steps of performing in-place upgrade of Data Lake and Data Hub clusters. For steps required for the backup and restore approach, refer to the respective documentation on backing up Data Lakes and Data Hubs and performing metadata restore (automated for Data Lake clusters only).

Upgrade Checklist FAQ

During the preparation for an upgrade, Cloudera recommends carefully reviewing the questions and answers below.

What is the length of the available maintenance window?

Currently, Data Lake backup and restore requires a maintenance window, where no metadata changes occur. Furthermore, Data Hubs need to be stopped during an upgrade.

The CDP Public Cloud environment does not need to be upgraded in one go: you may opt to upgrade the Data Lake and all attached Data Hubs together, or start with the Data Lake upgrade only and perform individual Data Hub upgrades in consecutive steps. However, after you perform a major/minor Data Lake upgrade, you must upgrade all attached Data Hub clusters, as the Data Hubs must run the same major/minor version of Runtime as the Data Lake.

What type of upgrade is required?

Currently, there are three types of upgrades available to Data Lake and Data Hub clusters: maintenance upgrades; minor/major version upgrades; and OS upgrades. Maintenance and minor/

major version upgrades install a newer version of Cloudera Manager and/or Cloudera Runtime. OS upgrades for Data Lakes and Data Hubs are complementary and will bring the image of the cluster hosts to a newer version. If you plan to also perform an OS upgrade, plan the maintenance window accordingly.

Are ephemeral disks used for user or workload-related persistent data?

Major/minor version upgrades as well as maintenance upgrades will bring Cloudera Manager and Cloudera Runtime to the selected version without impacting the underlying VM. However, OS upgrades will recreate the underlying VM with a fresh image, which results in the loss of any data stored on ephemeral disks.

If you are currently storing user or workload-related data on volumes using ephemeral disks, please reach out to Cloudera support while planning for the upgrade.

What Data Hub cluster templates are in use? Are you using custom templates?

Check whether in-place upgrade is supported for your built-in or custom data hub template. Depending on the type and version of the Data Hub, additional backup steps, manual configuration changes or post-upgrade steps may be required. Check specific steps for upgrading the OS if you use Flow Management. Operational Database clusters have a different upgrade process.

What is the size of the SDX / Data Lake metadata?

SDX metadata includes the Hive Metastore database, Ranger audit log index, as well as Atlas metadata. If you are planning to perform a Data Lake backup before an upgrade (which is recommended), prepare your maintenance window accordingly. CDP supports skipping the backup of certain metadata to reduce the time required for backup and restore operations.

Are you using Data Services?

If you have deployed Cloudera Data Engineering, Data Warehouse, Data Flow, or Machine Learning in your environment, it is recommended that you upgrade them to the latest version before upgrading your Data Lake to a more recent minor/major version.

Preparing for an upgrade

Upgrading CDP Public Cloud consists of two major steps: upgrading the Data Lake within an environment and then upgrading the attached Data Hubs. Currently the Data Hubs must run the same version of Cloudera Runtime as the Data Lake.

You should also periodically upgrade the environment (FreeIPA cluster) to ensure that you are running the latest security patches, but this is not required at the same time as upgrading the Data Lake and Data Hubs.

Data Lake upgrade workflow

Pre-upgrade tasks

1. Review the Data Lake upgrade requirements and limitations.
2. Carefully review the differences between the types of upgrades and what they entail.
3. Check the Data Lake support matrix for upgrade to verify which Runtime versions you can upgrade to and from.
4. If you have not configured your Data Lake for backup and restore, you will need to do so. The backup and restore process is integrated into the upgrade flow automatically, but successful upgrade requires that the correct IAM policies exist on the DATALAKE_ADMIN_ROLE and RANGER_AUDIT_ROLE (for backup), and the DATALAKE_ADMIN_ROLE, RANGER_AUDIT_ROLE, and LOG_ROLE (for restore).

   If your roles are not configured correctly, CDP will not be able to write the backup to the BACKUP_LOCATION_BASE path of your cloud storage.
5. If you are performing the backup manually (as opposed to the integrated backup available during the upgrade process), you can launch the Data Lake backup from the UI or CLI. When using the CLI, you can specify to skip

certain backup actions (skip HMS, Atlas metadata or Ranger audit log index backup). You can monitor the backup process using the CLI.

6. From the Data Lake UI, run the Validate and Prepare option to check for any configuration issues and begin the Cloudera Runtime parcel download and distribution. Using the validate and prepare option does not require downtime and makes the maintenance window for an upgrade shorter. Validate and prepare also does not make any changes to your cluster and can be run independently of the upgrade itself. Although you can begin the upgrade without first running the validate and prepare option, using it will make the process smoother and the downtime shorter. (The parcels that are downloaded and distributed by the validate and prepare option are specific to the Runtime version that you have selected, so if you use validate and prepare and then decide to upgrade to a different Runtime version instead, you will need to re-run validate and prepare. Be aware that if you use validate and prepare for multiple major/minor Runtime versions, the parcels for different versions are not cleaned up and may saturate the disk. These parcels are cleaned up only once the upgrade is complete.)

Data Lake upgrade tasks

1. Perform the upgrade, either through the UI or CLI. The type of upgrade that you perform will depend on whether a new version of Runtime or only a hotfix is available. A new OS image may also be available to upgrade to.
2. If the upgrade fails, check the logs for manual troubleshooting info. You can also recover from failed upgrades.
3. When the upgrade succeeds, proceed to upgrading the attached Data Hubs.

Data Hub upgrade workflow

Pre-upgrade tasks

1. Carefully review the differences between the types of upgrades and what they entail.
2. Check that the cluster that you want to upgrade is supported.
3. From the Data Hub UI, run the Validate and Prepare option to check for any configuration issues and begin the Cloudera Runtime parcel download and distribution. Using the validate and prepare option does not require downtime and makes the maintenance window for an upgrade shorter. Validate and prepare also does not make any changes to your cluster and can be run independently of the upgrade itself. Although you can begin the upgrade without first running the validate and prepare option, using it will make the process smoother and the downtime shorter. (The parcels that are downloaded and distributed by the validate and prepare option are specific to the Runtime version that you have selected, so if you use validate and prepare and then decide to upgrade to a different Runtime version instead, you will need to re-run validate and prepare. Be aware that if you use validate and prepare for multiple major/minor Runtime versions, the parcels for different versions are not cleaned up and may saturate the disk. These parcels are cleaned up only once the upgrade is complete.)

Major/minor Runtime version upgrade tasks

1. Backup the cluster data and CM configurations.
2. Perform the upgrade, either through the UI or CLI. (Operational database clusters have a different process.)
3. Complete any post-upgrade tasks for the type of cluster that you upgraded.
4. For DE clusters, use the CM UI to add any configs that were not added during the upgrade.
5. If the upgrade fails, check the Event log and the troubleshooting section.
6. Complete any post-upgrade tasks for the type of cluster that you upgraded.
7. For DE clusters, use the CM UI to add any configs that were not added during the upgrade.
8. If the upgrade fails, check the Event log and the troubleshooting section.

Maintenance (hotfix) upgrade tasks

1. Backup the cluster data and CM configurations.
2. Perform the upgrade, either through the UI or CLI.
3. If the upgrade fails, check the Event log and the troubleshooting section.

OS upgrade tasks

1. Review the Before you begin section to verify that there is no data belonging to NiFi or NiFi Registry on the root disk of the VM. Note that during an OS upgrade, any data on the root volume (parcels, service logs, custom software) will be lost.
2. Unlike the Data Lake upgrade, OS upgrades are not integrated in the larger upgrade flow and must be performed separately, either through the UI or CLI.

FreeIPA upgrades

When a FreeIPA upgrade is available, upgrade the FreeIPA cluster to ensure that the nodes are running the latest OS-level security patches.

Cloudera Data Hub

Troubleshooting Cloudera Data Hub

Date published: 2019-12-17
Date modified: 2023-06-27

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Checking logs **4**
Cloudera Manager logs 4
Recipe logs 4

Troubleshooting cluster creation **4**
Failed to retrieve the server's certificate 4
Quota limitations 4

Troubleshooting upgrade issues **5**

Checking logs

When troubleshooting Cloudera Data Hub clusters, you can access the following logs:

Cloudera Manager logs

Cloudera Data Hub uses Cloudera Manager to orchestrate the installation of Cloudera Runtime components. Each instance in the cluster runs an Cloudera Manager agent which connects to the Cloudera Manager server. Cloudera Manager server is declared by the user during the cluster installation wizard.

Refer to the following documentation for information on how to access Cloudera Manager logs:

Related Information
Cloudera Manager Logs

Recipe logs

Cloudera Data Hub supports “recipes” - user-provided customization scripts that can be run prior to or after cluster installation. It is the user’s responsibility to provide an idempotent well tested script. If the execution fails, the recipe logs can be found at /var/log/recipes on the nodes on which the recipes were executed.

It is advised, but not required to have an advanced logging mechanism in the script, as Cloudera Data Hub always logs every script that are run. Recipes are often the sources of installation failures as users might try to remove necessary packages or reconfigure services.

Troubleshooting cluster creation

The following section lists common issues related to cluster creation.

Failed to retrieve the server's certificate

Cloudera Data Hub cluster fails with "Infrastructure creation failed. Reason: Failed to retrieve the server's certificate".

When a cluster fails with "Infrastructure creation failed. Reason: Failed to retrieve the server's certificate", the most common reasons for this error are related to using your own custom image. If you are using your own custom image and if using the CLI, you must send the imageId explicitly in the CLI in the cluster template request.

Quota limitations

Each cloud provider has quota limitations on various cloud resources, and these quotas can usually be increased on request. If there is an error message in Cloudera Data Hub stating that there are no more available EIPs (Elastic IP Address) or VPCs, you need to request more of these resources.

To see the limitations visit the cloud provider’s site:

- AWS service limits
- Azure subscription and service limits, quotas, and constraints

Troubleshooting upgrade issues

The following section lists common issues related to upgrading Cloudera Data Hub clusters.

Failed to start role of service profiler_scheduler in cluster

While upgrading from Cloudera Runtime version 7.2.17 to 7.2.18, and specifically during the Cloudera Data Hub OS upgrade step, the cluster goes into the failure state. The following message is seen:

```
NODE_FAILURE:
New node(s) could not be added to the cluster. Reason Please find more det
ails on Cloudera Manager UI. Failed command(s): Start(id=1546339088): Failed
 to start role profc6cf3856-PROFILER_SCHEDULER_AGENT-484032cb8f17cacf9e684ef
e50 of service profiler_scheduler in cluster cdp-dc-profilers-258395ef.
```

Impact on Cloudera Data Catalog profilers:

If the Cloudera Data Hub cluster is not created, the profilers will not be created in Cloudera Runtime 7.2.18 version.

Workaround:

You must use the following process to bring up the Cloudera Data Catalog profilers in the 7.2.18 version.

1. First you must delete your existing 7.2.17 clusters. For more information, see Deleting profiler cluster.
2. Next, after you upgrade to the 7.2.18 Data Lake, then you can launch the Cloudera Data Catalog profilers. For more information, see Launch profiler cluster.

Note: There is no data loss expected on the users' side or the Profiler analysis. However, the only loss that could be expected is related to the last runtime value of the profiler and the profiler run history. The Profiler Last Runtime history refers to the records of how many runs of the profiler are displayed on the history page. It includes information on whether the runs were completed successfully or resulted in failures.

CDV 7.0.0 CUSTOM STYLES MIGRATION GUIDE

CDV 7.0.0, May 16, 2022

Table of Contents

Legal Notice 3

What is the purpose of this guide? 4

Does this apply to me? 4

How can I confirm we are impacted? 4

Impact and general guidance on updating Custom Styles CSS and/or JS 6

Example workflow for updating a Custom Style broken by the changes in CDV 7.0.0 6

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses.

Please review the license and notice files accompanying the software for additional licensing information. Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

What is the purpose of this guide?

As part of the CDV 7.0.0 release, Cloudera has made some major updates to the Bootstrap and jQuery libraries. Bootstrap was upgraded from 3.3.7 to 4.6.0 and Jquery was upgraded from 2.2.4 to 3.6.0. These changes have potential impacts to customers who have created Custom Styles since these would potentially contain code that references legacy Bootstrap classes or Jquery methods in their CSS and/or JS respectively.

Does this apply to me?

If you have created any Custom Styles prior to CDV 7.0.0 or using legacy Arcadia Enterprise, the information in this guide might be relevant for you.

How can I confirm we are impacted?

1. Create a new CDV Project.
2. Take a copy of your existing CDV metadata (from the project files this would be the SQLite database located in /home/cdsw/.arc/arcviz.db)
3. Copy this SQLite database to /home/cdsw/.arc/ in your new project.
4. Select the new CDV 7.0.0 runtime for the project, and start the CDV Application.

   NOTE: If you are using an external database like MySQL, MariaDB, Postgres, or Oracle, you will need to make a backup of the database and restore it as a new database that the new instance of CDV can connect to.

5. After CDV starts, go through each Dashboard and Application using Custom Styles in the new CDV instance and compare with the original CDV instance running an older runtime version (CDV 6.x) to see if the styling or functionality is broken in comparison.

To determine where your Custom Styles are being used on certain Dashboards and Visuals, check the Custom Styles page and look at the column that indicates their links to the Dashboards and Visuals.

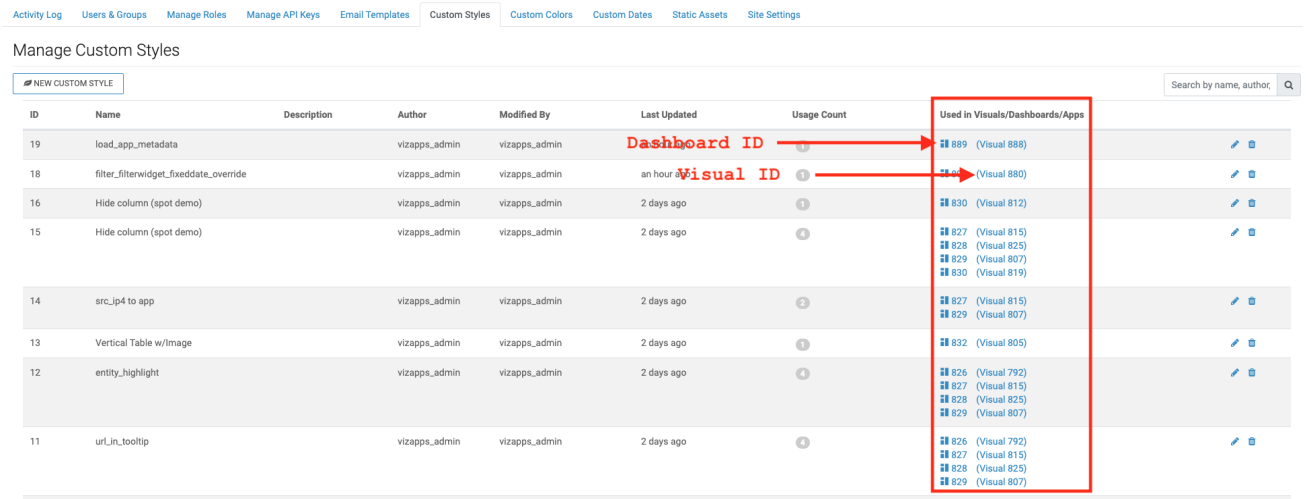

Also check your Dashboards, Visuals, and Applications to see if they have any inline CSS applied to them as they could be impacted by the Bootstrap upgrade.

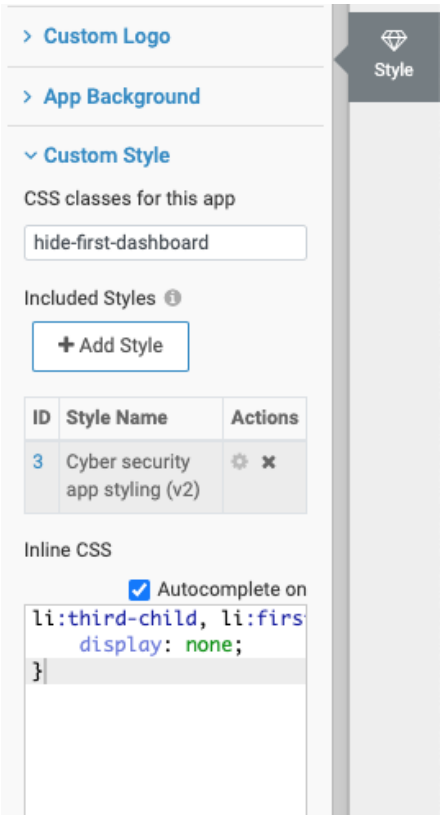

Impact and general guidance on updating Custom Styles CSS and/or JS

- If you added custom CSS Styling directly to your Arcadia or CDV Applications (Appgroups) you may have issues following this upgrade with certain class changes related to the Application navbar.
- You may also have nested CSS classes you're using within your Custom Style JS selectors to make modifications to visuals. Check your JS code to see if you need to change your nested class selections if any class has changed as part of the CDV 7.0.0 upgrade.
- If you were using Glyphicons as part of your Custom Style you need to replace the Glyphicon references to icons that match the Font Awesome icon set:
    - Font Awesome Icon Reference: https://fontawesome.com/icons
- As part of the Bootstrap upgrade, there were some structural changes for certain types of classes. It is best to reference the Bootstrap 3.x to 4.x migration guide here if you need to make any fixes if there were any problems during functional testing.

  Bootstrap 3.x - 4.x changes:
  https://getbootstrap.com/docs/4.0/migration/
- You may also be using JQuery element selectors or functions in your Custom Style JS. If your selectors or functions are not working properly you might find some helpful guidance on updating your code here.

  Jquery upgrade guide:
  https://jquery.com/upgrade-guide/3.0/

Example workflow for updating a Custom Style broken by the changes in CDV 7.0.0

Example: Broken Application Navbar Styling

After upgrading to CDV 7.0.0 our Application Navbar appears to be broken. The Font types, colors, navbar lower bar color, and logo positioning look very different from the original version if you compare both the screenshots below.

Before (Arcadia Enterprise or CDV 6.x):

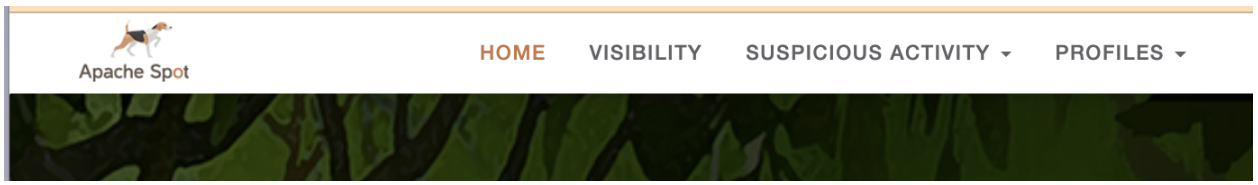

After 7.0.0 upgrade:

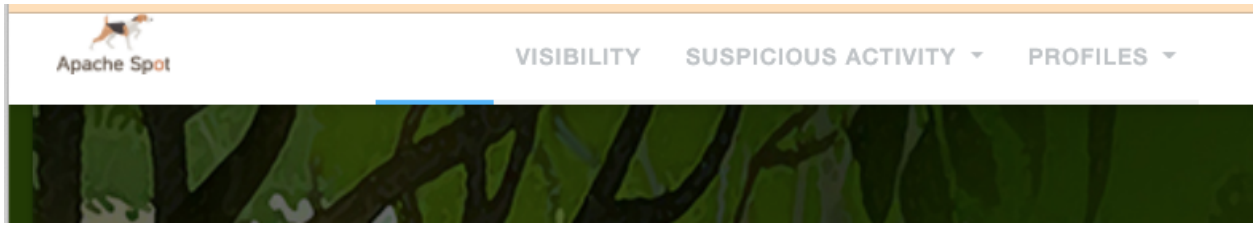

There are two options we have for repairing the styling of our Application:

1. Fix the CSS - Conform your styling code to the new HTML and CSS changes as part of the CDV 7.0.0 upgrade.
2. Use Application Designer styling options to match previous styling and remove custom CSS.

Option 1: Fix CSS

Based on the screenshots above we need to make the following fixes to our CSS:

- Fix Navbar menu tab text colors when selected, when not selected, and the font weight.
- Fix Navbar menu tab lower border bar color when selected, and when not selected
- Fix Logo position and top margin spacing

Step 1: Open your original CSS Custom Style

Click the ID to open your Custom Style in a new browser tab.

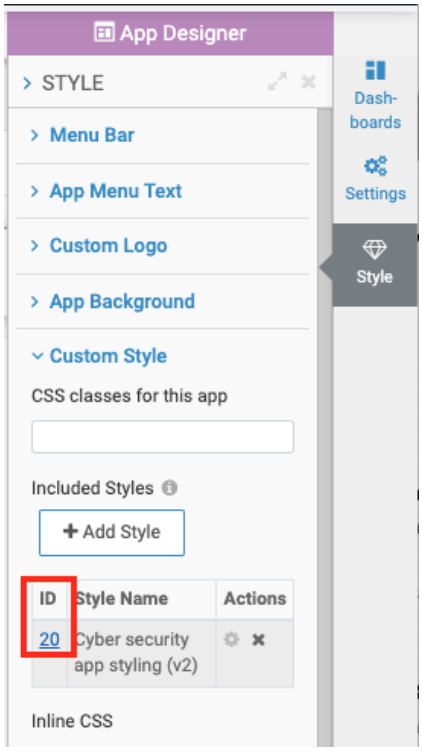

Step 2: Clone the Custom Style

Cloning is a safer approach so you can compare the old CSS to the new as you make changes.

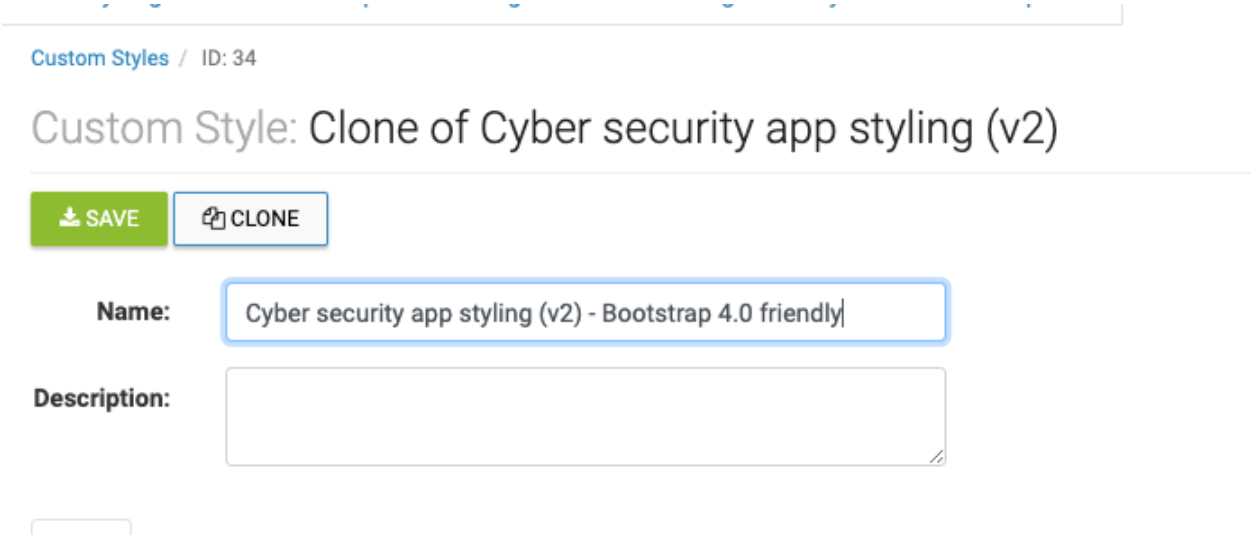

Step 3: Compare the HTML and CSS structure between Arcadia / CDV 6.x and CDV 7.0.0

Arcadia / CDV 6.x HTML / CSS

Use the Elements selector under Chrome Developer Tools to select the navbar so you can focus on those HTML / CSS elements.

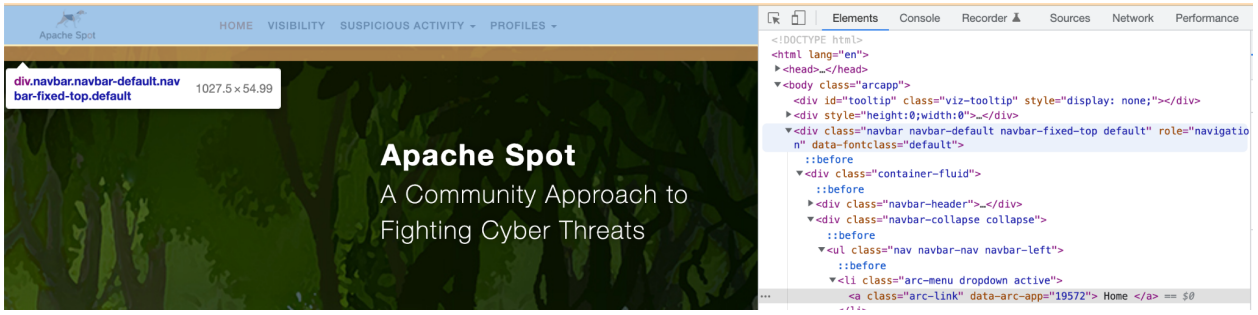

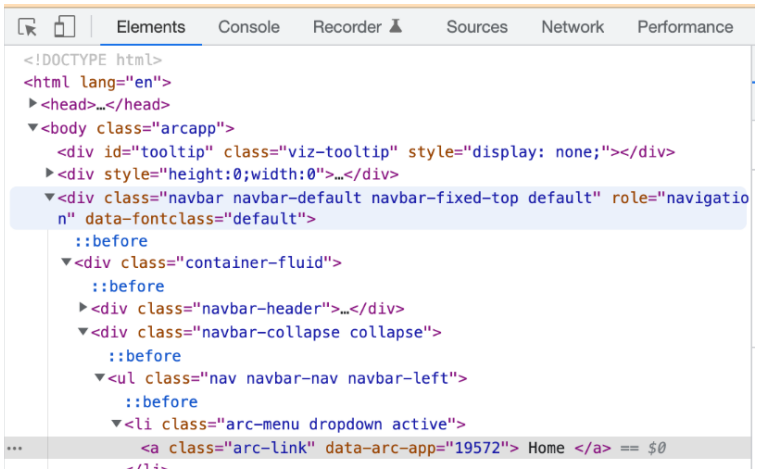

```
▼<ul class="nav navbar-nav navbar-left">
  ::before
  ▼<li class="arc-menu dropdown">
...  <a class="arc-link" data-arc-app="19572"> Home </a> == $0
  </li>
  ▶<li class="arc-menu dropdown active">…</li>
  ▼<li data-appgroupid="838" class="arc-menu dropdown open">
    ▼<a class="dropdown-toggle" data-toggle="dropdown" role="button"
     aria-expanded="true">
       " Suspicious Activity "
       <span class="caret"></span>
     </a>
```

CDV HTML / CSS

Use the Elements selector under Chrome Developer Tools to select the navbar so you can focus on those HTML / CSS elements.

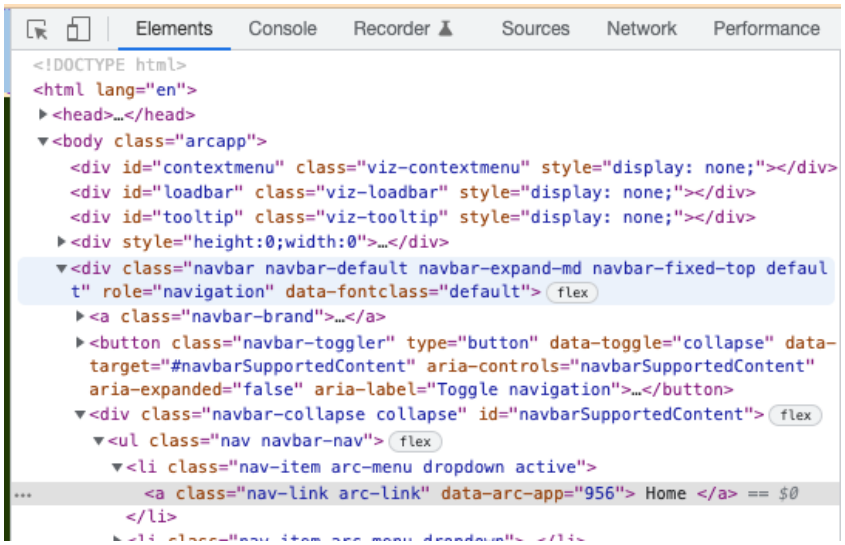

```
▼<ul class="nav navbar-nav">  flex
  ▼<li class="nav-item arc-menu dropdown">
...     <a class="nav-link arc-link" data-arc-app="956">
        Home </a> == $0
    </li>
  ▼<li class="nav-item arc-menu dropdown active">
      <a class="nav-link arc-link" data-arc-app="955">
      Visibility </a>
    </li>
  ▼<li data-appgroupid="58" class="nav-item arc-menu drop
    down show">
    ▶<a class="nav-link dropdown-toggle" data-toggle="dro
      pdown" role="button" aria-expanded="true">…</a>
    ▶<ul class="dropdown-menu show" role="menu">…</ul>
    </li>
  ▶<li data-appgroupid="61" class="nav-item arc-menu drop
    down">…</li>
```

Step 4: Update CSS to work with new CDV HTML and CSS class structure

Original CSS:

```
#dashboard-viz-title {
  display: none;
}
.arc-dashboard-title-container {
  color: #c97644;
}
/*.viz-params-control {*/
/*  display:none;*/
/*}*/
.arc-appgroup-logo {
  background-image: url(/arc/reports/staticasset/raw/6.png);
  max-height: 46px;
  margin-top:-1px;
}
.navbar > .container-fluid .navbar-brand {
  margin-left: 0;
  padding: 5px 20px 15px 23px;
}
.navbar {
  background-color: #fff;
  border-bottom: solid 3px #fff;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.25);
  -webkit-box-shadow: 0 0 10px rgba(0, 0, 0, 0.25);
  -moz-box-shadow: 0 0 10px rgba(0, 0, 0, 0.25);
}
.navbar-default .navbar-nav > li > a {
  font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
  color: #626366;
  font-weight: 500;
  font-size:14px;
  border-bottom: solid 3px #fff;
}
.navbar-default .navbar-nav > li > a:hover, .navbar-default .navbar-nav > li > a:focus {
  background-color: transparent;
  border-bottom: solid 3px #fff;
  margin-bottom: -3px;
  color: #fff;
}
.navbar-default .navbar-nav > li > a:hover {
  border-bottom: solid 3px #fff;
  color: #c97644;
}
```

```
.navbar-default .navbar-nav > .active > a, .navbar-default .navbar-nav > .active > a:hover, .navbar-default .navbar-nav > .active > a:focus
{
  background-color: transparent;
  border-bottom: solid 3px #fff;
  color: #c97644;
}
.nav .open {
  background-color: #fff;
}
.navbar-default .navbar-nav>.open>a,
.navbar-default .navbar-nav>.open>a:hover{
  background-color: #fff;
  color: #c97644;
}
```

Fixed CSS (see bolded/highlighted areas):

```
#dashboard-viz-title {
  display: none;
}
.arc-dashboard-title-container {
  color: #c97644;
}
/*.viz-params-control {*/
/*  display:none;*/
/*}*/
.arc-appgroup-logo {
  background-image: url(/arc/reports/staticasset/raw/6.png);
  max-height: 46px;
  margin-top:5px;
}
.navbar > .navbar-brand {
  margin-left: 0;
  padding: 5px 20px 15px 23px;
}
.navbar {
  background-color: #fff;
  border-bottom: solid 3px #fff;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.25);
  -webkit-box-shadow: 0 0 10px rgba(0, 0, 0, 0.25);
  -moz-box-shadow: 0 0 10px rgba(0, 0, 0, 0.25);
}
.navbar .navbar-nav > .nav-item > a {
  font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
  color: #626366;
  font-weight: 500;
  font-size:14px;
  border-bottom: solid 3px #fff;
}
.navbar .navbar-nav > .nav-item > a:hover,
.navbar .navbar-nav > .nav-item > a:focus {
  background-color: transparent;
  border-bottom: solid 3px #fff;
  margin-bottom: -3px;
  color: #fff;
}
.navbar .navbar-nav > .nav-item > a:hover {
  border-bottom: solid 3px #fff;
  color: #c97644;
}
.navbar .navbar-nav > .nav-item.arc-menu.active > a,
.navbar .navbar-nav > .nav-item.arc-menu.active > a:hover,
.navbar .navbar-nav > .nav-item.arc-menu.active > a:focus {
  background-color: transparent;
```

```
  border-bottom: solid 3px #fff;
  color: #c97644;
}
.nav .show {
  background-color: #fff;
}
.navbar .navbar-nav > .nav-item .show > a,
.navbar .navbar-nav > .nav-item .show > a:hover{
  background-color: #fff;
  color: #c97644;
}
```

Notes on fixing the Logo top margin

Issue: Logo is not centered well between top and bottom of Navbar

Original CSS

```
.arc-appgroup-logo {
  background-image: url(/arc/reports/staticasset/raw/6.png);
  max-height: 46px;
  margin-top:-1px;
}
```

Fix: Adjust the top margin pixels to 5px to push the logo down

```
.arc-appgroup-logo {
  background-image: url(/arc/reports/staticasset/raw/6.png);
  max-height: 46px;
  margin-top:5px;
}
```

Before

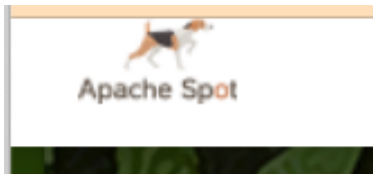

After

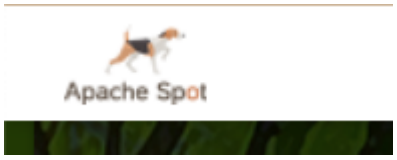

Step 5: Apply your newly updated CSS Custom Style

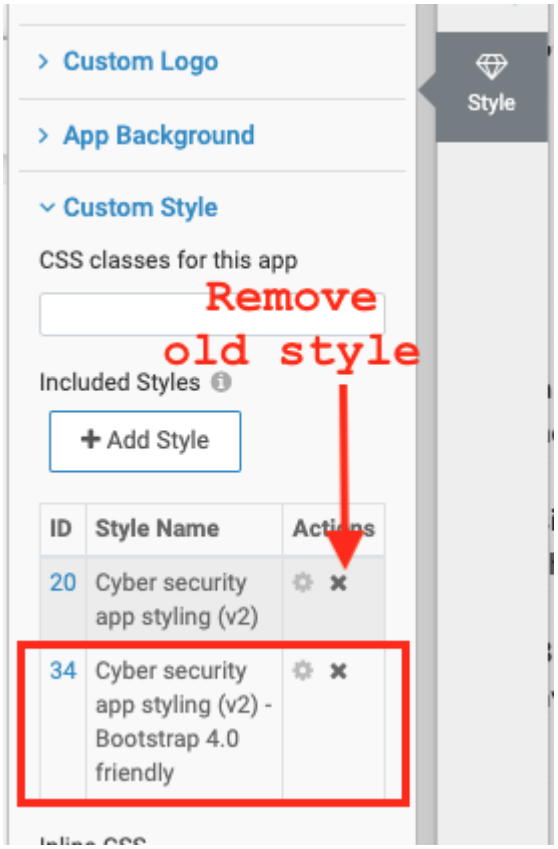

Step 6: Compare before and after changes

Before (with broken Custom CSS):

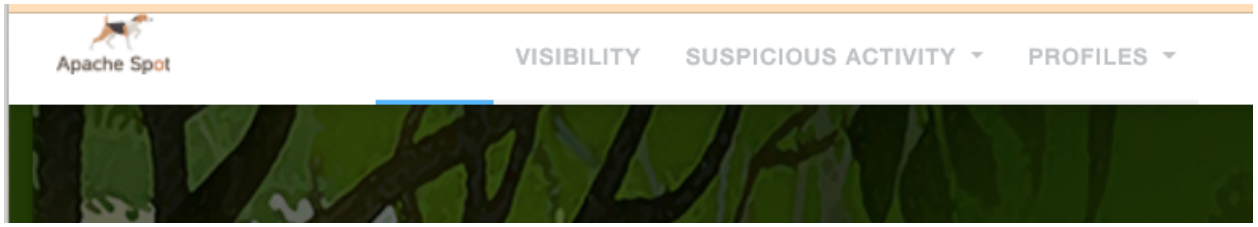

After (with fixed Custom CSS):

Option 2: Use Application Designer styling options

You can use Application Designer styling options to match previous styling and remove your custom CSS.

Step 1: Match your Application styling to the out of the box Style options in the App Designer menu

In this example, the Background color of our application is white, the bottom bar color on the navbar is white (both by default and when selected), Text color is dark gray and it becomes burnt orange when selected.

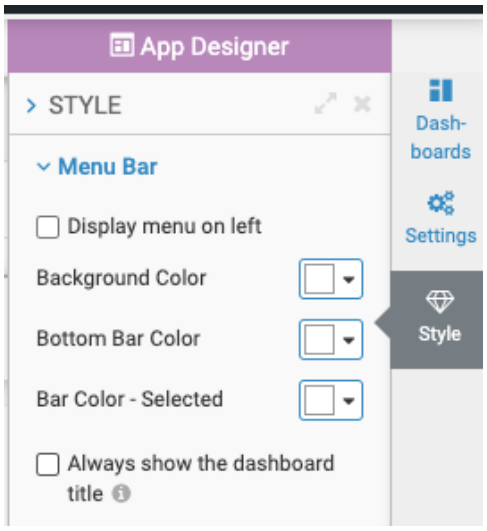

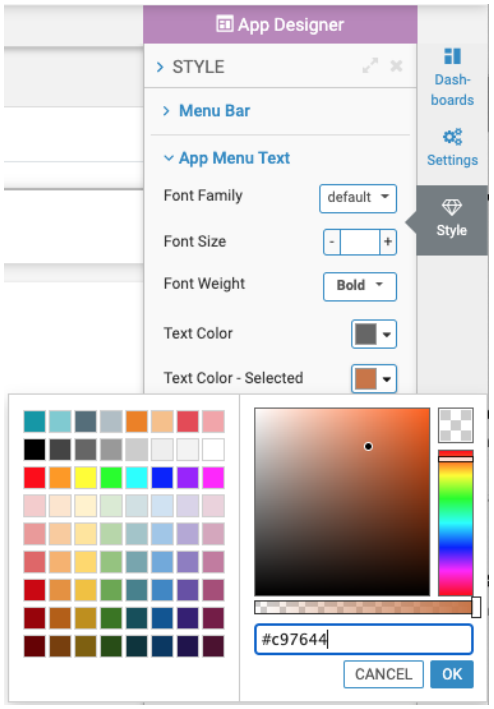

You can paste the relative URL path to the static asset of our logo here (this path can be copied from the CSS in Option 1).

```
.arc-appgroup-logo {
  background-image: url(/arc/reports/staticasset/raw/6.png);
  max-height: 46px;
  margin-top:-1px;
}
```

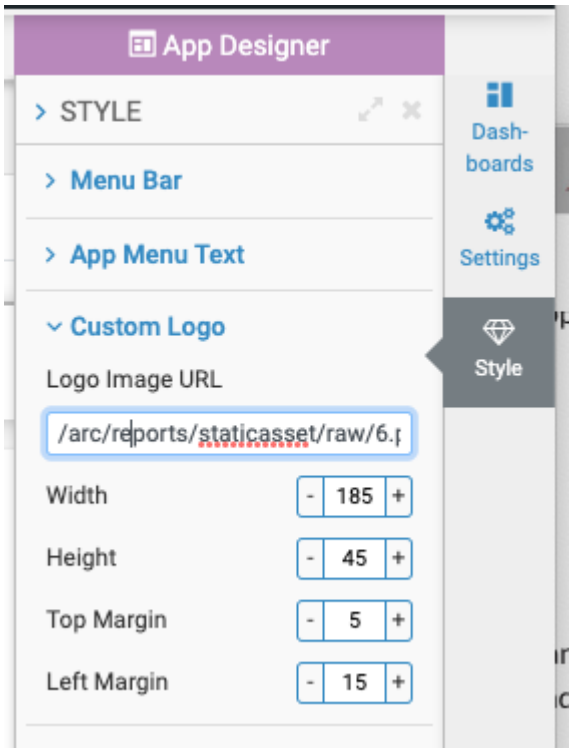

The Background behind the Dashboards is always the default, so no color needs to be selected here.

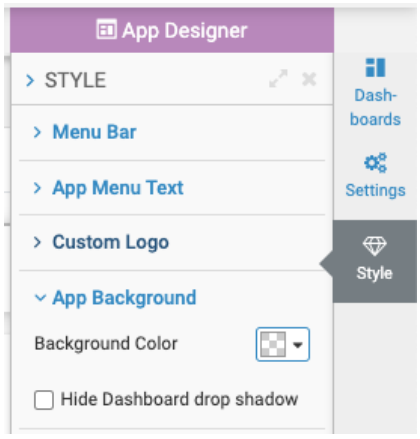

Step 2: Remove CSS Custom Style from Style settings

Since you have matched our CSS styles to the App Designer styles, you can now remove the CSS Custom Style from the Application.

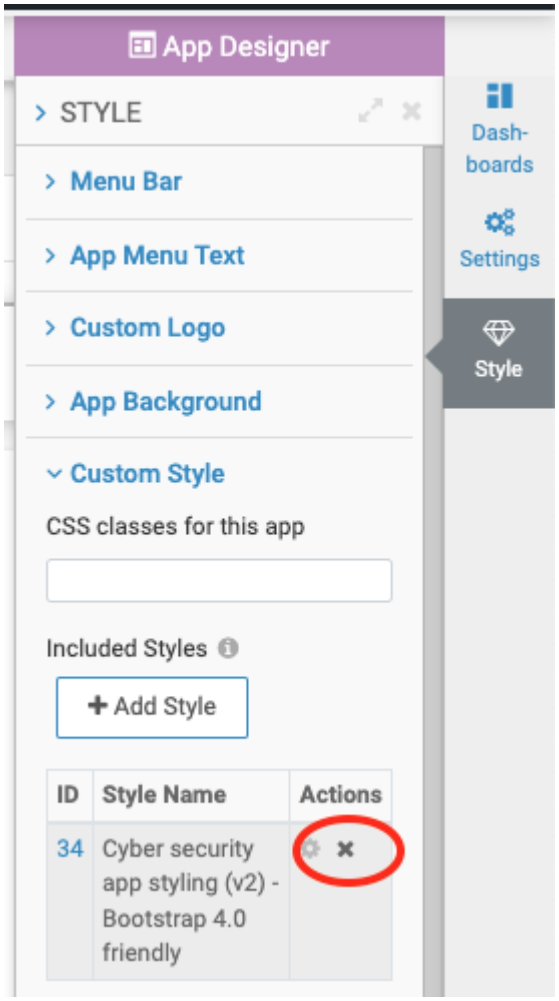

Step 3: Compare before and after changes

Before (with Custom CSS):

After (with App Designer Options):

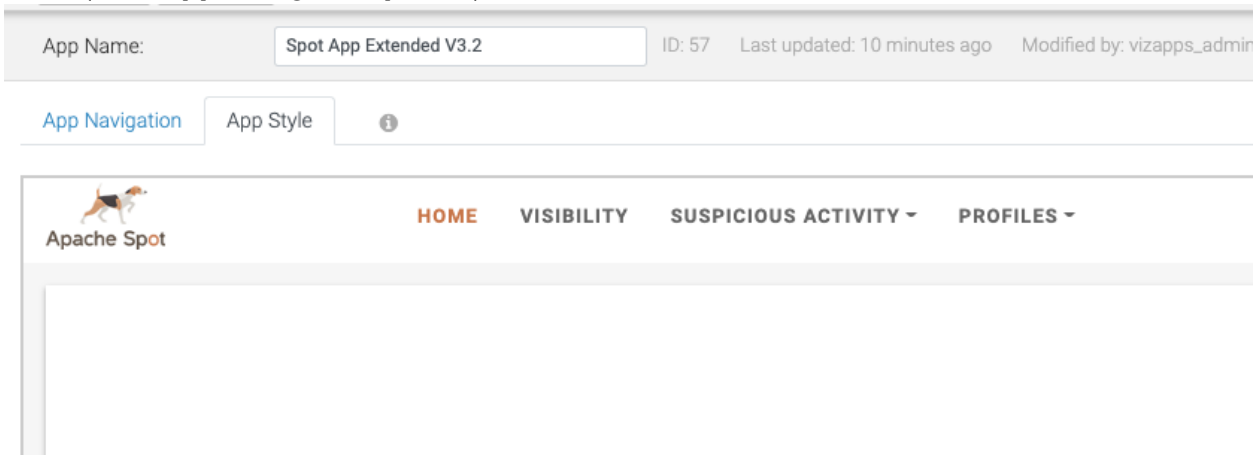

Cloudera Data Visualization 7.2.9

Configuring Account Lockout

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring account lockout..... 4

Unlocking account.....5

Configuring account lockout

Cloudera Data Visualization allows an administrator to change the default account lockout configurations.

About this task

Note: This feature is only available to users with administrative privileges.

Procedure

- To change the account lockout settings for a user, edit the following parameters in the settings_local.py file.

  Note: Updated values can get overwritten if the application is restarted.

  AXES_FAILURE_LIMIT

  Specifies the number of failed login attempts after which the user is locked out.

  The limit definition uses the following formula:

  ```
  AXES_FAILURE_LIMIT = Number of failed attempts * 2
  ```

  Note: The default setting is '10', which means that users will be locked out after five failed attempts.

  AXES_LOCK_OUT_BY_COMBINATION_USER_AND_IP

  Specifies the account lockout settings based on a combination of failed login attempts and IP address. The default ruevalue is 'True'.

  If the flag is set to True, the user cannot login using the same IP address after exceeding login attempts specified in the AXES_FAILURE_LIMIT . However, other users can login using the same IP address.

  If the flag is set to False, the user cannot login using the same IP address after exceeding login attempts specified in the AXES_FAILURE_LIMIT . In this case, other users are also not able to login using the same IP address.

  Note: In both settings the locked out user and other users can login using a different IP address.

  To lock the user based on a combination of failed login attempts and IP address, enter the the following command syntax:

  ```
  AXES_LOCK_OUT_BY_COMBINATION_USER_AND_IP = <True>
  ```

  AXES_COOLOFF_TIME

  An integer that specifies the number of hours after lockout that the user must wait before being able to login again. The default value is '0.33 hours' (20 mins.)

  To specify a wait time before user can login again after failed attempts, enter the the following command syntax:

  ```
  AXES_COOLOFF_TIME = <Number of Hours>
  ```

Unlocking account

Cloudera Data Visualization allows an administrator to unlock a user's account.

About this task

Note: This feature is only available to users with administrative privileges.

To unlock a user account, enter the following command syntax: $ ./arcviz axes_reset_user <username>.

Cloudera Data Visualization 7.2.9

Logs and Usage

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Activity Log **4**

Usage Analytics [Technical Preview] **8**

Daily Usage **12**

WebServer Stats **13**

Current Load **19**

WebServer Log **20**

Activity Log

Logs and Usage is a powerful monitoring interface that can help you better understand how users are engaging with Cloudera Data Visualization in your organization. The different tabs provide you detailed information on various aspects, including user activities, system load, and runtime performance. The Activity Log tab of the interface provides access to the audit logs stored in query log files.

You can access the Activity Log through Site Administration Logs and Usage .

Note:

The Logs and Usage interface is only accessible to system administrators (or to users who have a role with View activity logs or Admin user permissions).

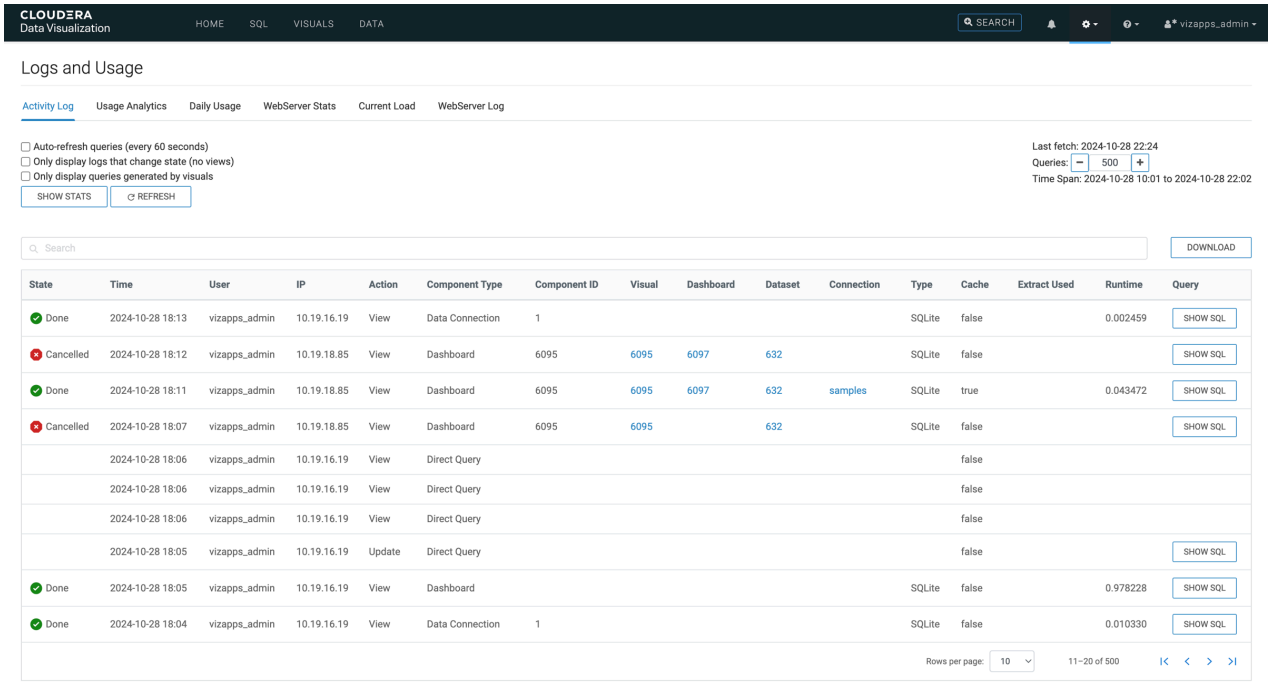

When you open the Activity Log interface, you can see a list view of the activity logs by default. This view provides comprehensive details about each activity.

Note:

As the data shown is persisted in files, restarting the Cloudera Data Visualization instance does not impact the information presented.

Key functions

Within Activity Log, you have access to several essential functions and information:

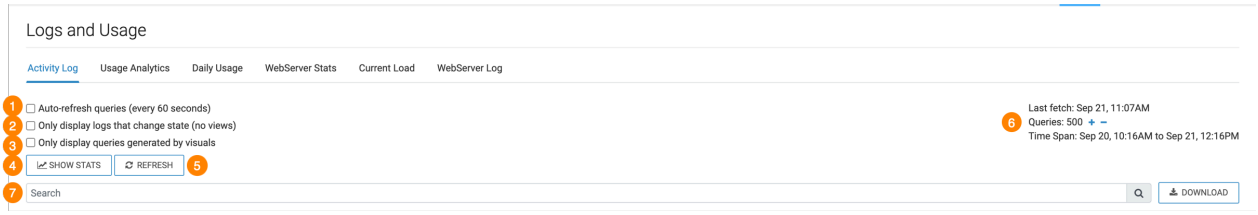

1. Auto-refresh queries

   Enabling this option automatically updates the information displayed in the interface every 60 seconds.
2. Only display logs that change state

   You can use this option to filter the logs, showing only actions that alter the state of artifacts.
3. Only display queries generated by visuals

   You can use this option to narrow the reporting to actions related to visuals. It excludes actions on datasets, connections, and analytical views from the logs.

4. Show Stats/Hide Stats

You can use this option to display or hide statistical visualizations of the activity logs based on object type.

Note: When you choose to display the visualizations, the traditional list view remains accessible below the visuals. Simply scroll down in your browser window to view it.

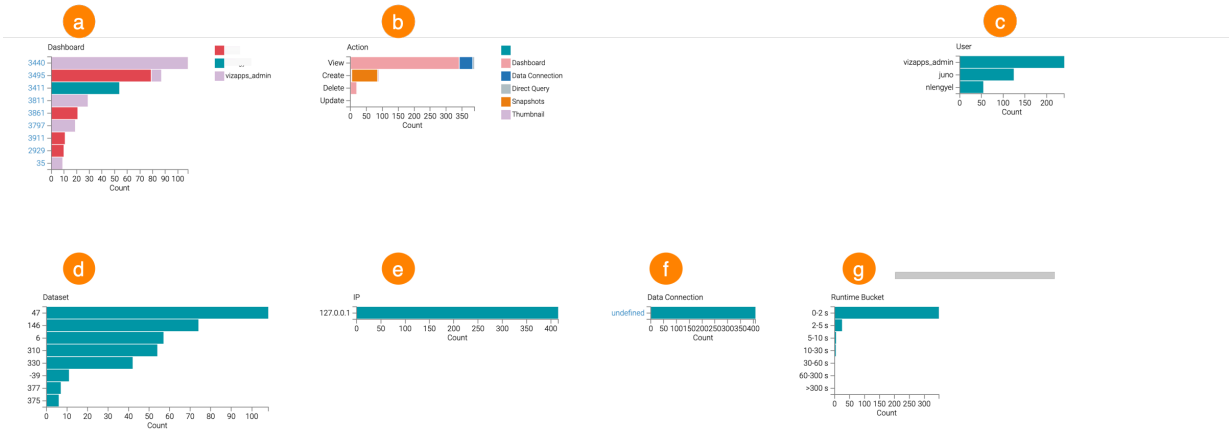

a. Dashboard

The horizontal bar chart shows the most frequently accessed dashboards, listing them by their unique IDs.

You can click the IDs to navigate to the respective dashboard, which opens in another browser tab. You can hover over a bar to display detailed information including dashboard ID, activity count, dataset ID, dataset name.

b. Action

The stacked bar chart provides a breakdown of requests by type, arranged along the vertical axis. Individual components within each bar and the legend show the breakdown by component type, such as Dashboard, Data Connection, Dataset, Snapshots, Thumbnail, or Workspaces.

You can hover over a bar to display in-depth insights into each action, including its type, activity count, and associated component.

c. User

The bar chart compiles data on all users who initiated actions. You can hover over a bar to display comprehensive information about each user, including their user name, activity count.

d. Dataset

The horizontal bar chart highlights the most frequently accessed datasets. You can hover over a bar to display detailed dataset information, including its ID, activity count, and name.

e. IP

The bar chart reports server addresses hosting various actions. You can hover over a bar to display detailed information about each connection, including its IPnumber and activity count.

f. Data Connection

The bar chart tracks activity on each live data connection. You can hover over a bar to display detailed information about each connection, including its name and activity count.

g. Runtime Bucket

The bar chart functions as a histogram, categorizing possible runtimes into distinct buckets. Each bar represents a bucket of duration times.You can hover over a bar to provide a breakdown of runtime bucket and activity count for further analysis.

5. Refresh

   This option triggers an immediate refresh operation, bypassing the waiting period for auto-refresh.
6. Information panel on the right side of the page:
   - Last fetch

     This timestamp captures the moment when the query was generated.
   - Queries

     This indicates the number of queries currently reported.

     The default value is 500. You can increase the number of queries by clicking the + icon incrementing them in multiples of 500. Conversely, clicking the - icon decreases the number of queries in increments of 500.
   - Time Span

     This time period reports the interval covered by the log data.
7. Search

   You can look up multiple terms, separated by spaces. The search results will show records that contain all the specified terms.

List view

You can check the list view of the Activity Log to get more detailed information about past activities.

By default, the log displays the last 500 activities in reverse chronological order, with the most recent action appearing at the top.

State	Time	User	IP	Action	Component Type	Component ID	Visual	Dashboard	Dataset	Connection	Type	Cache	Extract Used	Runtime	Query
Done	2024-10-28 18:13	vizapps_admin	10.19.16.19	View	Data Connection	1					SQLite	false		0.002459	SHOW SQL
Cancelled	2024-10-28 18:12	vizapps_admin	10.19.18.85	View	Dashboard	6095	6095	6097	632		SQLite	false			SHOW SQL
Done	2024-10-28 18:11	vizapps_admin	10.19.18.85	View	Dashboard	6095	6095	6097	632	samples	SQLite	true		0.043472	SHOW SQL
Cancelled	2024-10-28 18:07	vizapps_admin	10.19.18.85	View	Dashboard	6095	6095		632		SQLite	false			SHOW SQL
	2024-10-28 18:06	vizapps_admin	10.19.16.19	View	Direct Query							false			
	2024-10-28 18:06	vizapps_admin	10.19.16.19	View	Direct Query							false			
	2024-10-28 18:06	vizapps_admin	10.19.16.19	View	Direct Query							false			
	2024-10-28 18:05	vizapps_admin	10.19.16.19	Update	Direct Query							false			SHOW SQL
Done	2024-10-28 18:05	vizapps_admin	10.19.16.19	View	Dashboard						SQLite	false		0.978228	SHOW SQL
Done	2024-10-28 18:04	vizapps_admin	10.19.16.19	View	Data Connection	1					SQLite	false		0.010330	SHOW SQL

Rows per page: 10 11–20 of 500

- State indicates the completion status of the SQL query. Possible values are: Started, Done, Queued, Cancelled, Abandoned, Error.
- Time displays the date and time of the audited action.
- User shows the username of the person who initiated the activity (if available).
- IP provides the IP address of the computer used by the user carrying out the action (if available).
- Action indicates the type of event that triggered the logged activity (View, Create, Update, Delete).
- Component Type specifies the category of the activity (Dashboard, Data Connection, Dataset, User, App, Custom Style, Custom Color, Snapshots, Thumbnail, Static Asset, Role, Custom Email, Custom Date Range, Workspace, Search, Direct Query, Segment, Filter Association, Job, Extract).
- Component ID reflects the ID of the component involved in the action.
- Visual displays the ID of the visual that generated the action or is the target of the action. You can click this number to navigate to the visual.
- Dashboard displays the ID of the dashboard that generated the action or is the target of the action. You can click this number to navigate to the dashboard.
- Dataset displays the ID of the dataset that generated the action or is the target of the action. You can click this number to access the Dataset Detail interface for that dataset.

- Connection displays the ID of the data connection associated with the activity. You can click this name to navigate to the Datasets interface for that connection.
- Type specifies the type of data connection (for example: SQLite, Hive, Impala).
- Cache indicates whether the query was served from the cache instead of executing the SQL query. Possible values are true or false.
- Extract Used shows when SQL query rewriting is enabled. A link to a data extract indicates that the query was redirected to a different data table.
- Runtime shows the execution time of the SQL query in seconds (if SQL execution is involved).
- Query shows the SQL query statement (when available). By default, Cloudera Data Visualization hides the SQL statements. To view a specific query, click Show SQL. To hide statements, click Hide SQL.

Usage Analytics [Technical Preview]

Logs and Usage is a powerful monitoring interface that can help you better understand how users are engaging with Cloudera Data Visualization in your organization. The different tabs provide you detailed information on various aspects, including user activities, system load, and runtime performance. The Usage Analytics tab is a dynamic dashboard featuring visualizations generated from the audit logs presented on the Activity Log tab.

You can use Usage Analytics to get valuable insights into how users engage with Cloudera Data Visualization. You can get information on the most active users, frequently used datasets and visual types, peak activity hours, and other critical aspects of Cloudera Data Visualization operations, all through pre-built visualizations.

You can access the Usage Analytics through Site Administration Logs and Usage .

Note:

The Logs and Usage interface is only accessible to system administrators (or to users who have a role with View activity logs or Admin user permissions).

You have to enable Usage Analytics in Site Settings to add it to the Logs and Usage interface. For more information, see *Site Configuration*.

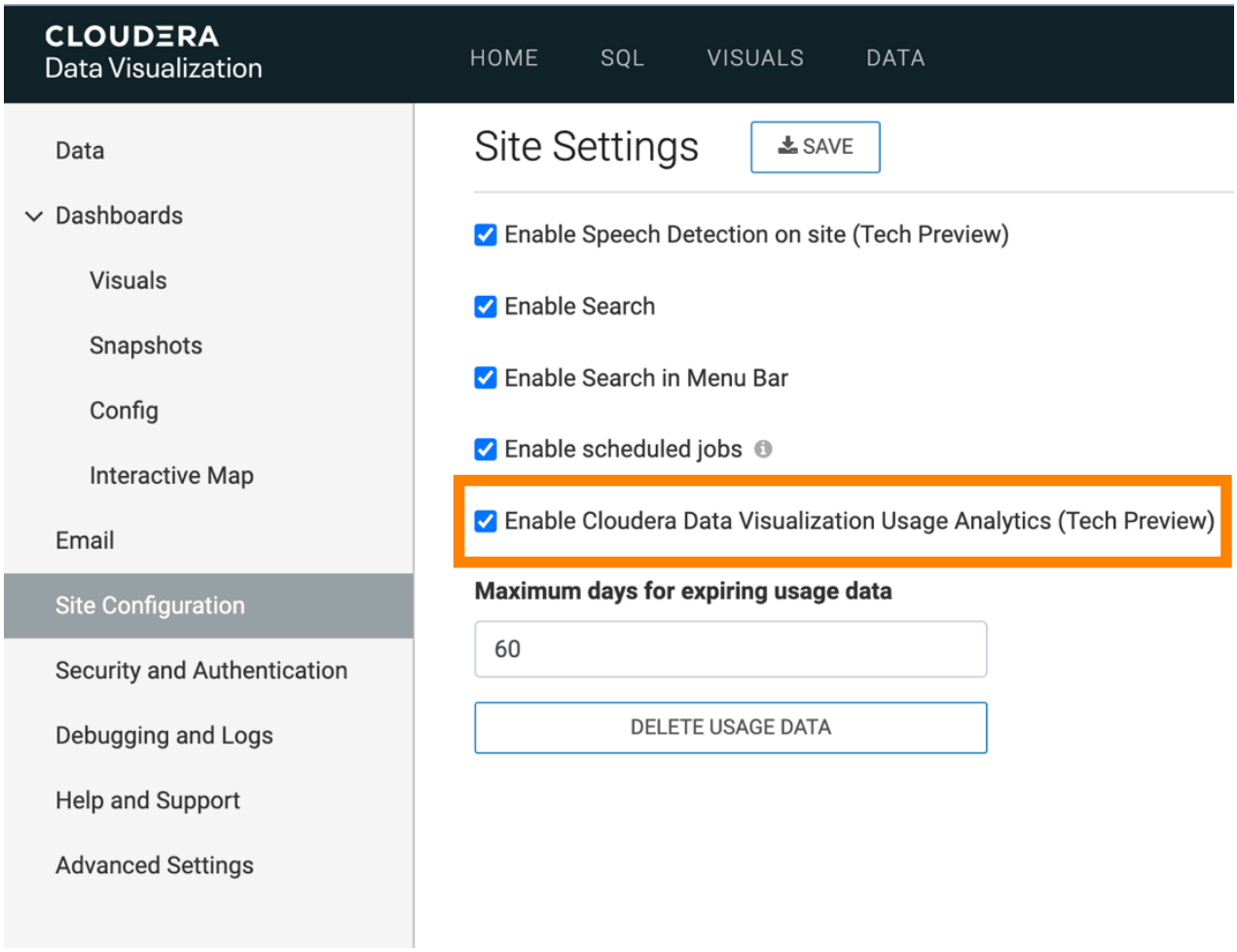

The dashboard offers a time range selector that enables you to specify the data used for the visualization. Filters are available for action type, visual type, and username to refine the data displayed.

The Usage Analytics view is further customizable. You can click IMPORT DASHBOARD to add an editable version labeled Cloudera Data Visualization Usage Analytics to your collection of dashboards.

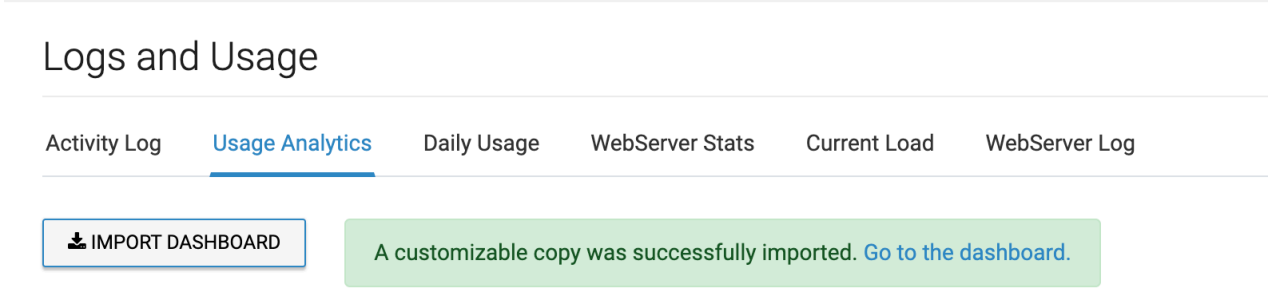

By default, the Usage Analytics dashboard includes the following visualizations:

- Activity count by user: Tracks user activity and provides insights into user engagement.
- Action types: Displays a breakdown of various action types, helping you understand user interactions.
- Interaction count by widget type: Provides insights into widget interactions and usage.
- Most used datasets: Highlights frequently accessed datasets.
- Number of dashboards: Shows the count of dashboards in use.
- Visuals usage: Offers insights into the usage of visual components.
- Number of other visuals: Indicates the usage count of other types of visuals.

- Users and their dataset usage: Provides information about users and their dataset usage patterns.
- Dataset usage over day's hours: Shows dataset usage trends throughout the day.

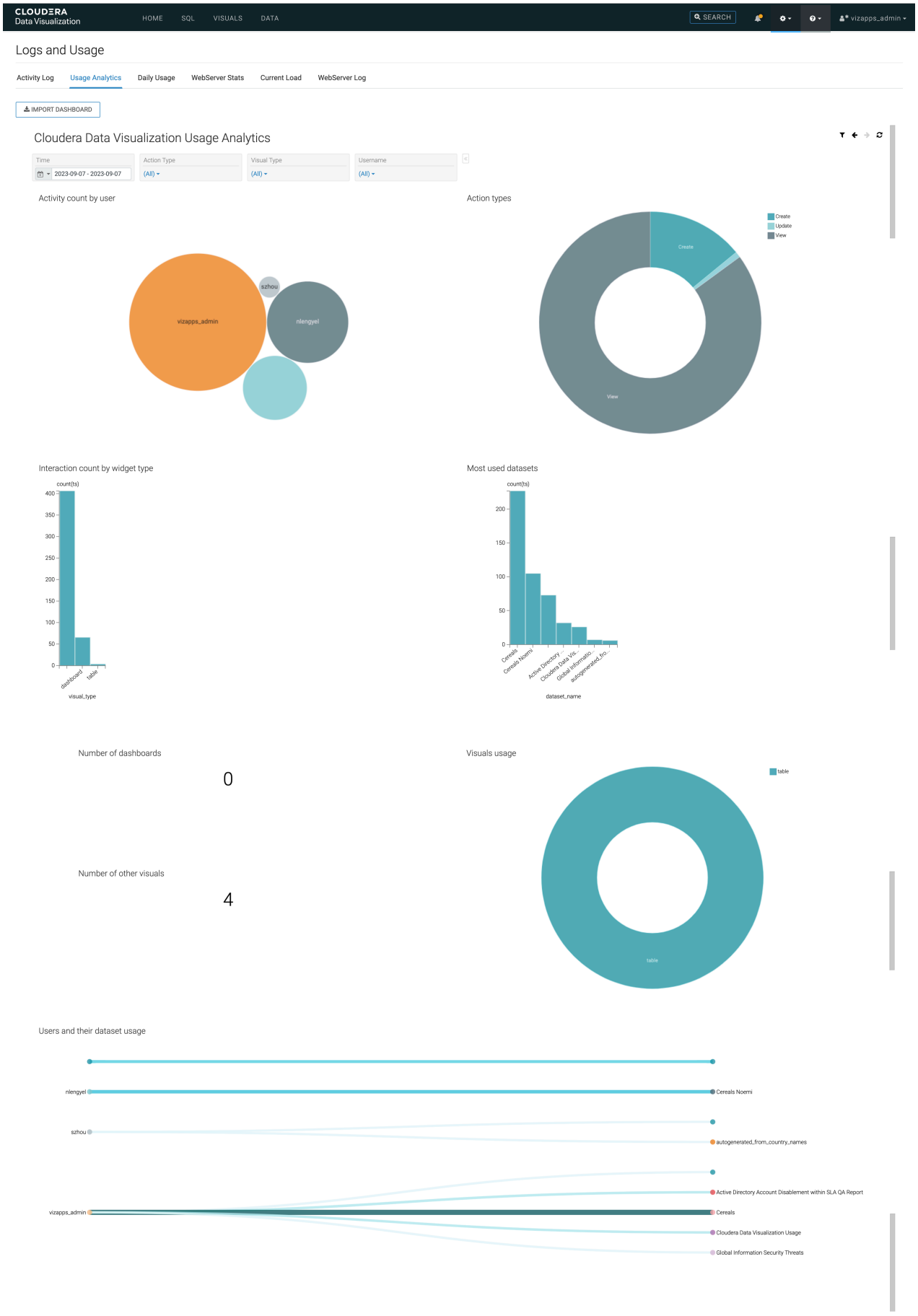
CLOUDERA
Data Visualization
HOME
SQL
VISUALS
DATA
SEARCH
vizapps_admin
Logs and Usage
Activity Log
Usage Analytics
Daily Usage
WebServer Stats
Current Load
WebServer Log
IMPORT DASHBOARD
Cloudera Data Visualization Usage Analytics
Time
2023-09-07 - 2023-09-07
Action Type
(All)
Visual Type
(All)
Username
(All)
Activity count by user
vizapps_admin
szhou
nlengyel
Action types
Create
Update
View
Interaction count by widget type
count(ts)
dashboard
table
visual_type
Most used datasets
dataset_name
Number of dashboards
0
Number of other visuals
4
Visuals usage
table
Users and their dataset usage
nlengyel
szhou
vizapps_admin
Cereals Noemi
autogenerated_from_country_names
Active Directory Account Disablement within SLA QA Report
Cereals
Cloudera Data Visualization Usage
Global Information Security Threats
Dataset usage over day's hours
Record Count

Related Information

Site Configurations

Daily Usage

Logs and Usage is a powerful monitoring interface that can help you better understand how users are engaging with Cloudera Data Visualization in your organization. The different tabs provide you detailed information on various aspects, including user activities, system load, and runtime performance. The Daily Usage tab of the interface provides you information on activities, users, and dashboards from the audit logs grouped by day.

Note:

The Logs and Usage interface is only accessible to system administrators (or to users who have a role with View activity logs or Admin user permissions).

You can access Daily Usage through Site Administration Logs and Usage .

The following information is available on the Daily Usage tab:

- Activities – This table provides a comprehensive view of all audit events that occurred on a specific day, encompassing actions like creates, reads, writes, and deletes
- Users – This table presents audit events categorized by day and user, showing the number of audited actions associated with each user.
- Dashboards – This table shows the number of activities impacting or related to visual components, grouped by date and dashboard/visual ID.

By default, Daily Usage shows data for the last 30 days, but you can customize the time period to suit your analytical needs.

You can download the collected data in CSV format, enabling further analysis and integration with other tools and systems.

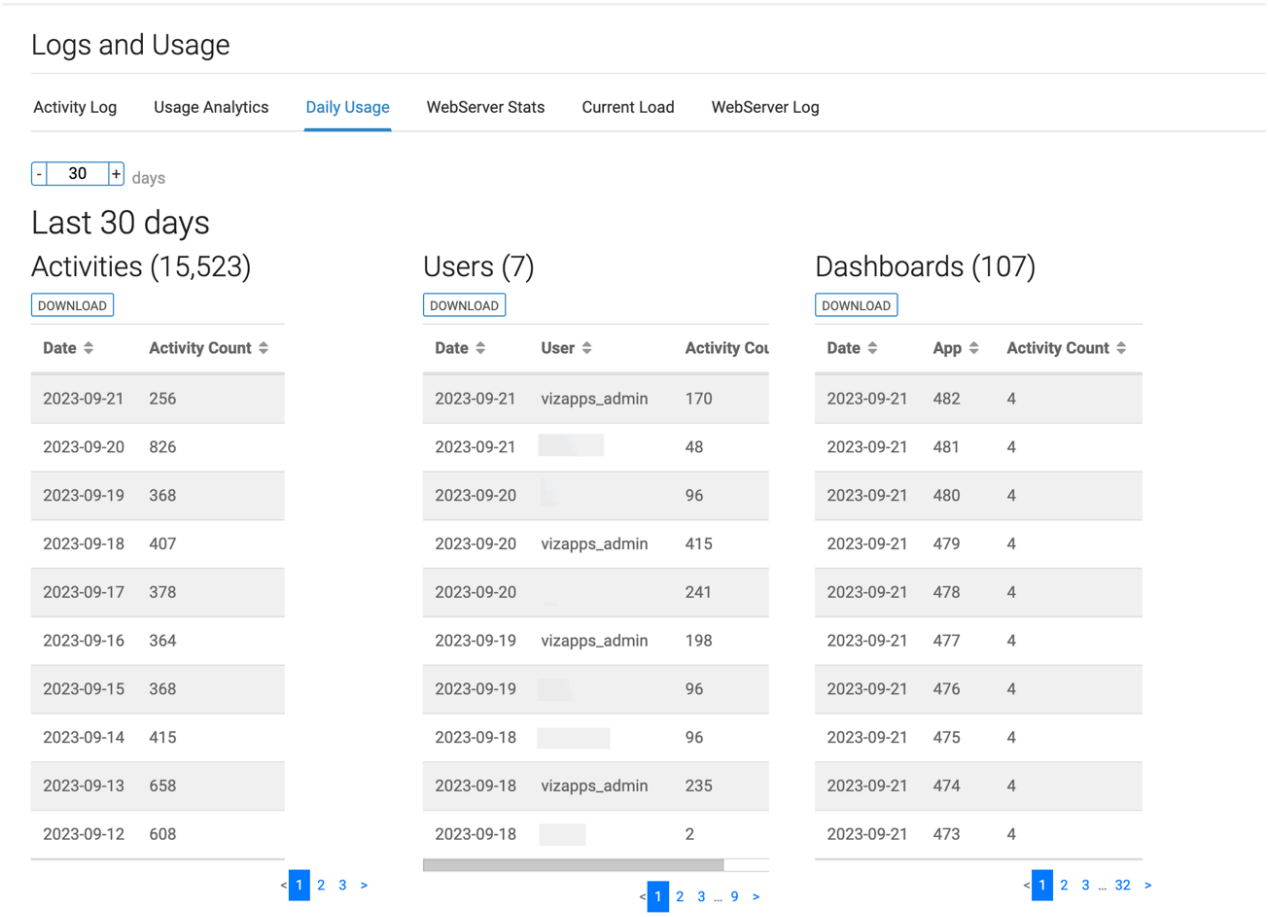

WebServer Stats

Logs and Usage is a powerful monitoring interface that can help you better understand how users are engaging with Cloudera Data Visualization in your organization. The different tabs provide you detailed information on various aspects, including user activities, system load, and runtime performance. The WebServer Stats tab of the interface provides runtime information. You can use it for performance tuning and troubleshooting. Monitoring runtime can help you identify performance bottlenecks, and plan how to optimize system performance.

Note:

The Logs and Usage interface is only accessible to system administrators (or to users who have a role with View activity logs or Admin user permissions).

You can access WebServer Stats through Site Administration Logs and Usage .

Most of the information displayed as webserver statistics is not permanently stored. Restarting the Cloudera Data Visualization instance significantly impacts the data presented.

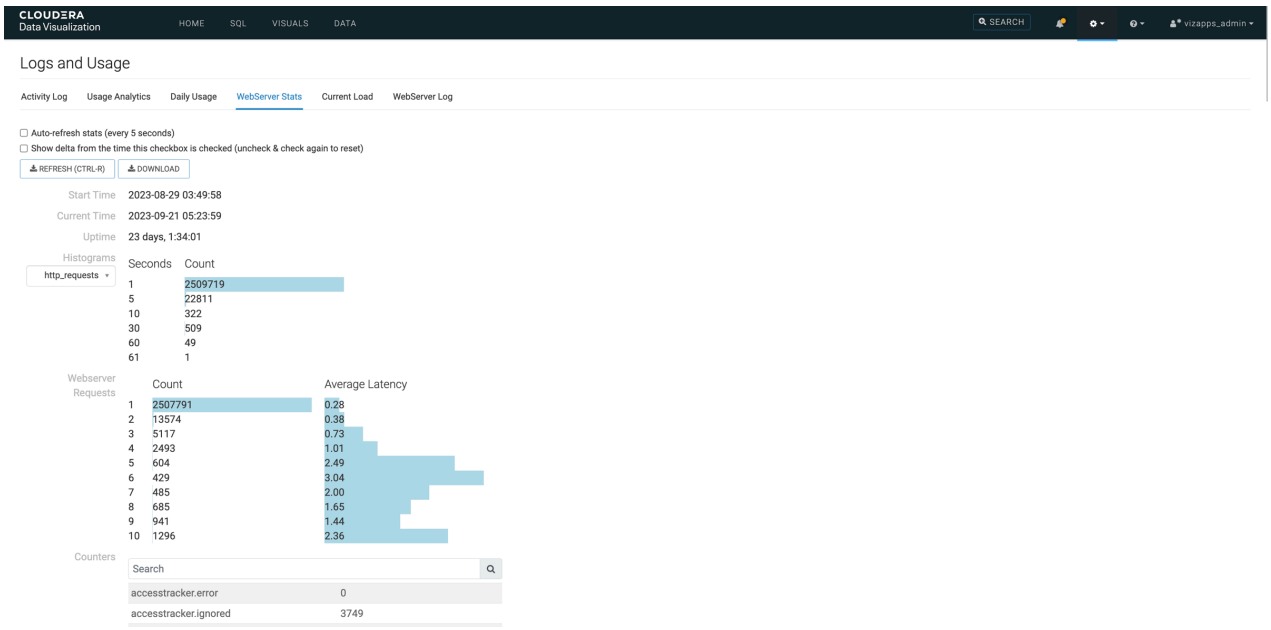

Key features and data:

- Auto-Refresh

  You can set the interface to auto-refresh at 5-second intervals for real-time data updates.
- Delta

  You have the option to view delta information from a specific point in time.
- Start Time

  This timestamp indicates when Cloudera Data Visualization was initially started.

  The default time zone for all time data received from the server is Pacific Standard Time (PST). You can configure the time zone in Advanced Site Settings. For instructions, see *Time zone* in the Advanced Settings documentation.
- Current Time

  It shows the time when the current page was retrieved.
- Uptime

  It shows the time elapsed since Cloudera Data Visualization was started, presented in the format of days, hours:minutes:seconds.
- Histograms

  This section provides insights into execution times within worker pools collected in several buckets (6 or 10).

  Worker pools (thread pools) are used for executing tasks asynchronously, such as database queries. This view provides information on the performance characteristics of tasks executed by worker pools. It helps assess the

access speed of specific data sources. The drop-down menu allows you to select different worker pools to gauge resource access speed.

Note: The histogram is constructed based on task (query) execution time, excluding the time spent in the queue.

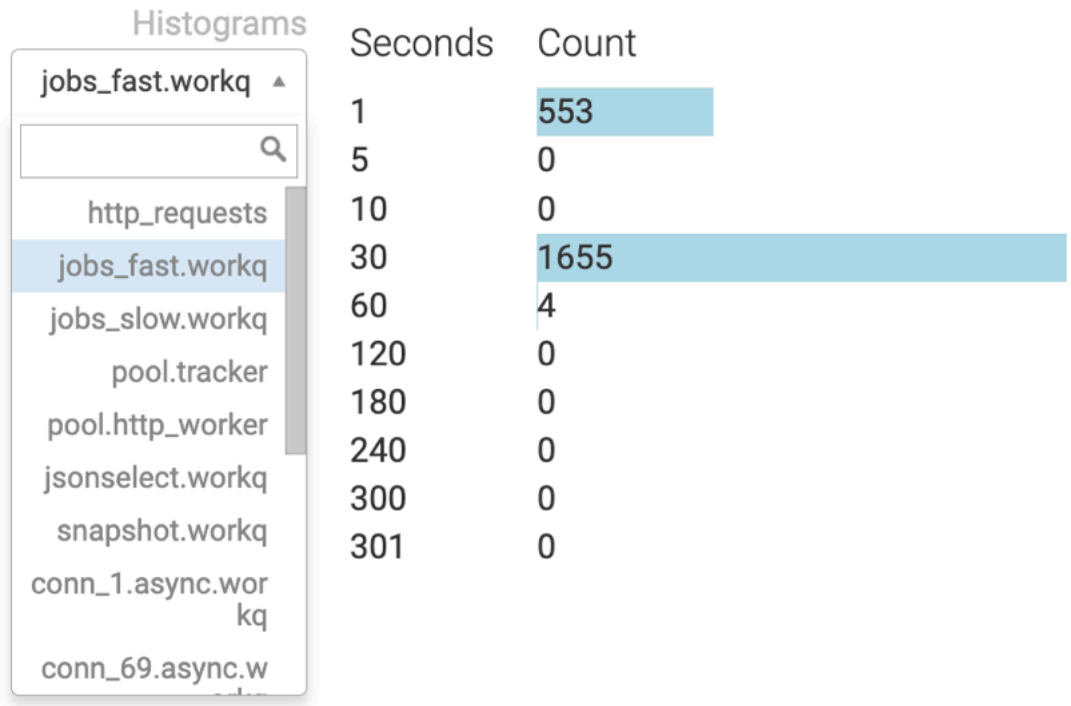

- Webserver Requests

  This section provides information regarding concurrent HTTP request handling. Since there are 10 HTTP worker threads, a maximum of 10 rows are displayed. The Count column shows the number of cases where the server had to manage a specific number of parallel requests (ranging from 1 to 10). The Average Latency column shows the average response time for requests within a given category.

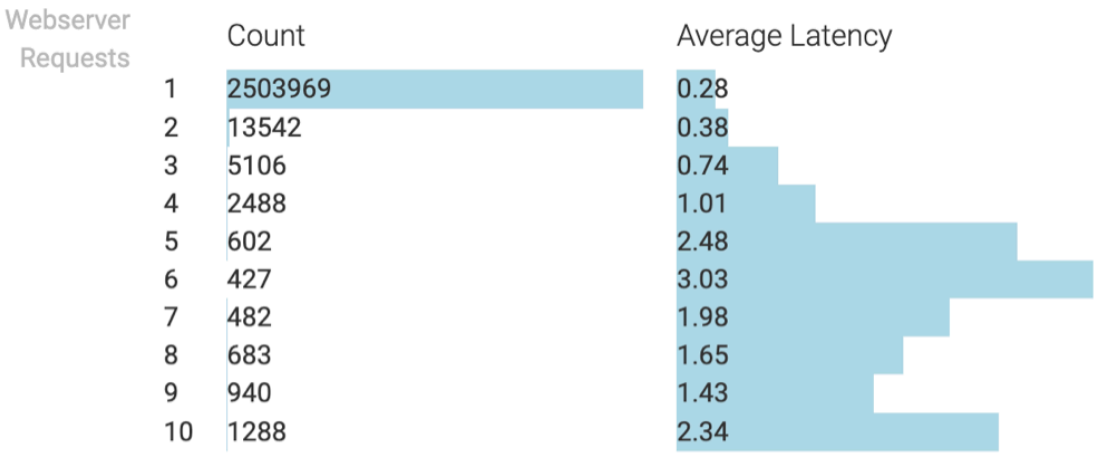

- Counters

  This section provides data for debugging purposes. It lists key-value pairs, where keys are dot-separated strings.

  - Keys starting with cpu offer CPU-related information:

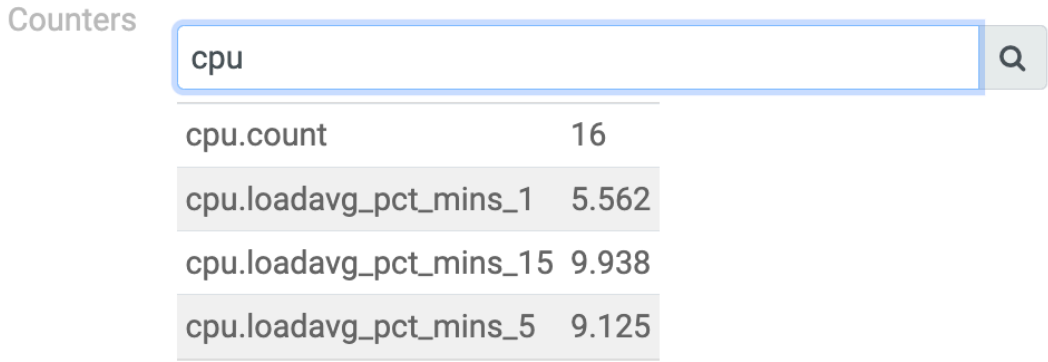

  - Keys starting with conn_ are related to a particular data connection:

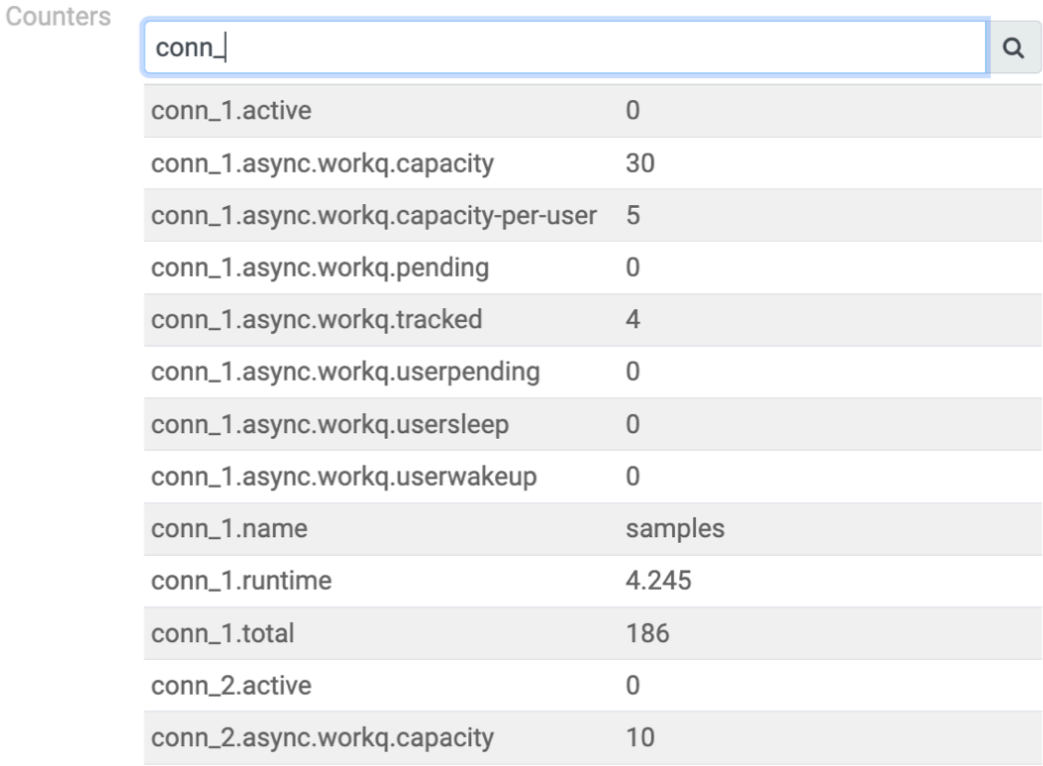

  - Explanation of the connection related counters:
    - active: It represents the number of queries currently being executed. A value of 0 indicates an idle connection, with no queries being executed.
    - async.workq counters:
      - capacity, capacity-per-user display the values of concurrency and concurrency per user settings for the data connection. These values remain constant unless you edit the data connection.
      - pending shows the number of queries waiting in the queue to be processed. For an idle connection, this value is 0.
      - Tracked shows the number of queries that were queued and are still awaiting to be delivered to the UI. For an idle connection, this value should be 0.

        - Userpending displays the number of queries waiting for execution due to exceeding the per-user concurrent execution limit.
        - Usersleep is the cumulative count of events when a query could have been executed because there were free workers in the pool, but was postponed because of the user-specific limit.
        - Userwakeup is the cumulative count of events when a query was rescheduled after the user-specific concurrency dropped below the configured limit.
    - name: It denotes the name of the data connection.
    - runtime: It indicates the cumulative runtime of queries, excluding queue time.
    - total: It represents the number of queries sent to the database backend. You can calculate the average execution time as runtime/total.
- Explanation of other counters:
    - accesstracker.*: It pertains to the Access tracking feature, which is no longer in use. It always displays only ignored items.
    - arcengine.create: It indicates the total number of legacy Arcengine, legacy Impala, or legacy Hive connections defined among the data connections.
    - arcengine_v2.create: It indicates the total number of non-legacy Arcengine connections. Non-legacy Impala and Hive connections have their respective counters: 'impyla' and 'hyve'.
    - arcengine_v2.connect: It tracks the number of times a non-legacy Arcengine connection was initiated to some of the defined data connection targets.
    - async.*: It provides metrics on specific HTTP endpoints marked as 'asynchronous'. These endpoints use a worker thread ('http_worker') to run incoming requests, returning a task handle for client-side result polling.
        - Async.jsonselect_parallel_async measures the cumulative number of scheduled tasks, typically data requests sent by dashboard visuals.
        - async.asyncstatus_async captures the cumulative number of client queries checking the status of previous requests. A considerably higher count compared to 'jsonselect_parallel_async' may indicate slow data retrieval (multiple checks are needed before getting the results to the client).
        - async.asyncresults_async records the number of times results were fetched by the client, which is typically low or 0, as status checks often yield data as a sign of completed processing.
        - async.asynccancel_async indicates the cumulative number of query cancellations not initiated explicitly by the user, often occurring when closing the browser page before data retrieval is complete.
        - Async.asynccancel_qt tracks the total number of tasks canceled explicitly by the user by clicking CANCEL ALL MY QUERIES on the current load page.
        - pool.http_worker.concurrency represents the number of concurrent task executions at the time of data collection, typically 0 in an idle system.
    - hive.create: It reflects the number of legacy Hive connections.
    - hs2.*: (hs2.conncreate, hs2.opensession, hs2.poolcreate, hs2.queries_retry, hs2.queries_submit): It contains legacy Hive Server 2 connection-related information (no longer maintained).
    - http_async.*: It offers information on the asynchronous HTTP-handling infrastructure.
        - Http_async.calls shows all HTTP client requests handled by the async framework.
        - Http_async.tasks shows the number of requests that resulted in a task being scheduled.
        - Http_async.retries shows the number of requests responded immediately (without scheduling a task).
    - hyve.create: It tracks the number of non-legacy Hive data connections defined by users.
    - hyve.connect: It monitors the number of connections established to non-legacy Hive data connections.
    - Impyla.create: It tracks the number of non-legacy Impala data connections defined by users.
    - impyla.connect: It monitors the number of connections established to non-legacy Impala connections.
    - jobs_fast.workq.*: It contains information related to email jobs.
        - Jobs_fast.workq.capacity shows the number of worker threads assigned for email jobs.
        - Jobs_fast.workq.pending shows the number of email jobs waiting to be processed. In an idle system this should be 0.

    - tracked, userpending, usersleep, userwakeup all have values that are constant zeros. They are inherited from the base worker queue implementation.
- jobs_slow.workq.*: It offers data extracts related information, similar to the information jobs_fast.workq provides for email jobs.
- jsonselect.workq.*: It provides details on an additional worker pool managing data queries. Entries have the same meaning as for jobs_fast and jobs_slow counters above.
- memory.rss_mb: It displays the memory resident set size (in megabytes).
- pool.tracker.concurrency: It indicates the concurrency level, related to access tracking. This counter is currently unused.
- query_stat.*: It encompasses cumulative data for existing data connections. In contrast to other counters, these values are persisted in the metadata store and are not reset by a server restart.
    - Query_stat.total_number_of_queries_returned shows the sum of all queries issued for all currently defined data connections since the time they were created.
    - Query_stat.returned_from_backend shows the number of queries that were actually sent to the backing data source.
    - Query_stat.returned_from_cache shows the number of queries that were served from the local cache.
- requests.concurrency_overflow: It reflects the number of times an HTTP request was ignored when calculating webserver requests statistics. Typically 0 due to the current concurrent limit of 10. This counter is currently unused.
- response.totalsize: It represents the total number of bytes returned by the HTTP requests, excluding streaming responses.
- rowdata.count: It captures the total number of data rows returned by non-streaming data queries.
- snapshot.queue: It shows the number of times a snapshot was created either when creating/updating a visual or when saving a dashboard.
- snapshot.workstarted: It tracks the total number of snapshoting tasks started, including snapshots created by email jobs and other actions.
- snapshot.workcompleted: It records the total number of completed snapshot tasks, typically matching snapshot.workstarted in an idle system.
- snapshot.workq.*: It contains snapshot worker queue counters, with meanings similar to jobs_fast.workq.*.
- workq.error: It shows the number of unhandled exceptions that occurred when executing tasks in worker pools. A value of 0 indicates stability.
- workq.markInterrupted: It captures the number of tasks not executed because they spent more time waiting in the queue than the specified execution timeout.
- workq.querycleanup.create: It indicates the number of times a cleanup thread was created for worker queues associated with data connections with a query timeout set. If the query timeout is modified, the cleanup thread is recreated.
- workq.querycleanup.cancels: It reflects the number of times a worker queue cleanup thread was canceled due to user changes in connection timeout settings.

- Active Queries Summary

  It provides a summary of queries/tasks executed or queued by various worker pools.

  Active Queries Summary

Queue	Capacity (Yours)	Running (Yours)	Queued (Yours)	Capacity (All)	Running (All)	Queued (All)
http	-	4	0	-	4	0

  - Queue is the name of the worker pool queue.
  - Capacity (Yours) shows the user-specific concurrency limit, if the queue supports it (data connection queues support it, others do not (like the http worker pool)).
  - Running (Yours) shows the number of tasks/queries running on behalf of the logged-in user.
  - Queued (Yours) shows the number of tasks queued for the current user.
  - Capacity (All) shows the overall number of tasks that can be executed in parallel by the worker pool.
  - Running (All) shows the number of all tasks executed by the pool across all users.
  - Queued (All) shows all tasks waiting in the pool to be processed for all users.
- Active Queries Detail

  It provides information about tasks being executed or queued.

  Active Queries Detail

Queue	Query Tracker	Start Time	Runtime	Username	Status	Query
http	00-1695299037.4306338	2023-09-21T05:23:57	0:00:01		Running	GET /arc/apps/
http	00-1695299037.4673054	2023-09-21T05:23:57	0:00:01		Running	GET /arc/datasets/dataconnection
http	00-1695299037.4891708	2023-09-21T05:23:57	0:00:01		Running	GET /arc/datasets/dataset
http	00-1695299037.6130688	2023-09-21T05:23:57	0:00:01		Running	GET /arc/stats_data/all

  - Queue is the name of the worker pool queue.
  - Query tracker is a task identifier.
  - Start Time shows the time when the task was submitted to the queue.
  - Runtime shows the time elapsed since the task was submitted.
  - Username displays the user to whom the task belongs.
  - Status indicates whether the task is Pending or Running.
  - Query displays the path of the SQL query or HTTP request, if available.

Current Load

Logs and Usage is a powerful monitoring interface that can help you better understand how users are engaging with Cloudera Data Visualization within your organization. The different tabs provide you detailed information on various aspects, including user activities, system load, and runtime performance. The Current Load tab of the interface offers the convenience of viewing essential information in real-time and provides a quick way to manage ongoing queries by canceling them if needed.

Note:

The Logs and Usage interface is only accessible to system administrators (or who have a role with View activity logs or Admin user permissions).

You can access the Current Load view through Site Administration Logs and Usage .

Current Load provides the following information:

- Data duplicated from WebServer Stats: Start Time, Current Time, Uptime, and Active Queries Detail.
- Active Queries Summary presenting aggregated data for all users. It is a consolidated view of the system's load but does not provide user-specific details.
- Option to cancel all queries currently in progress.

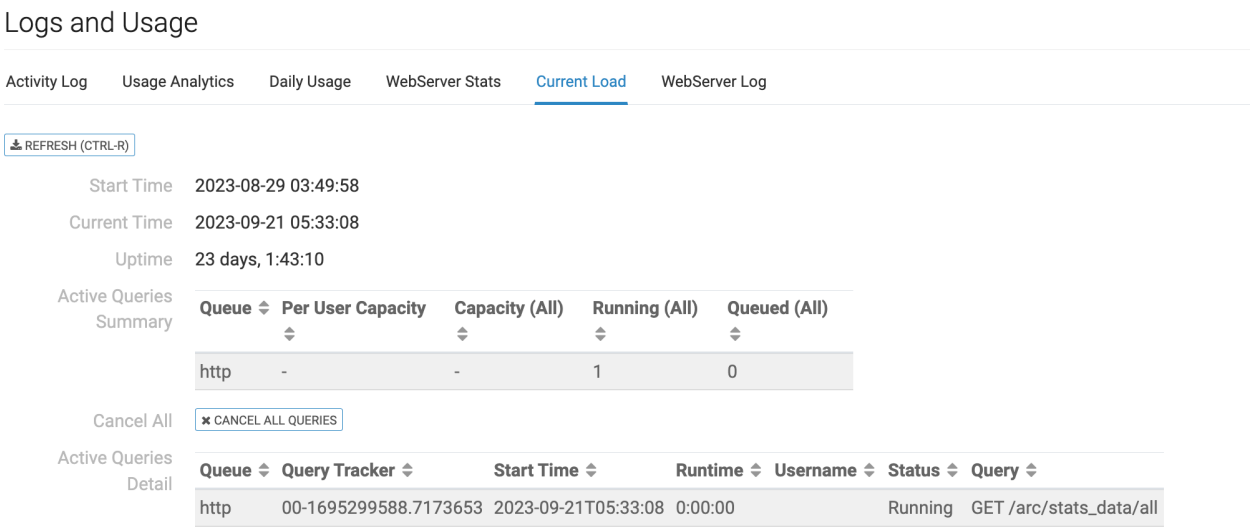

WebServer Log

Logs and Usage is a powerful monitoring interface that can help you better understand how users are engaging with Cloudera Data Visualization in your organization. The different tabs provide you detailed information on various aspects, including user activities, system load, and runtime performance. The WebServer Log tab provides access to the log messages generated by the Cloudera Data Visualization application.

You can use the technical details as a resource for debugging Cloudera Data Visualization issues or download the log files and share them with Cloudera Support.

Note:

The Logs and Usage interface is only accessible to system administrators (or to users who have a role with View activity logs or Admin user permissions).

You can access the WebServer Log view through Site Administration Logs and Usage .

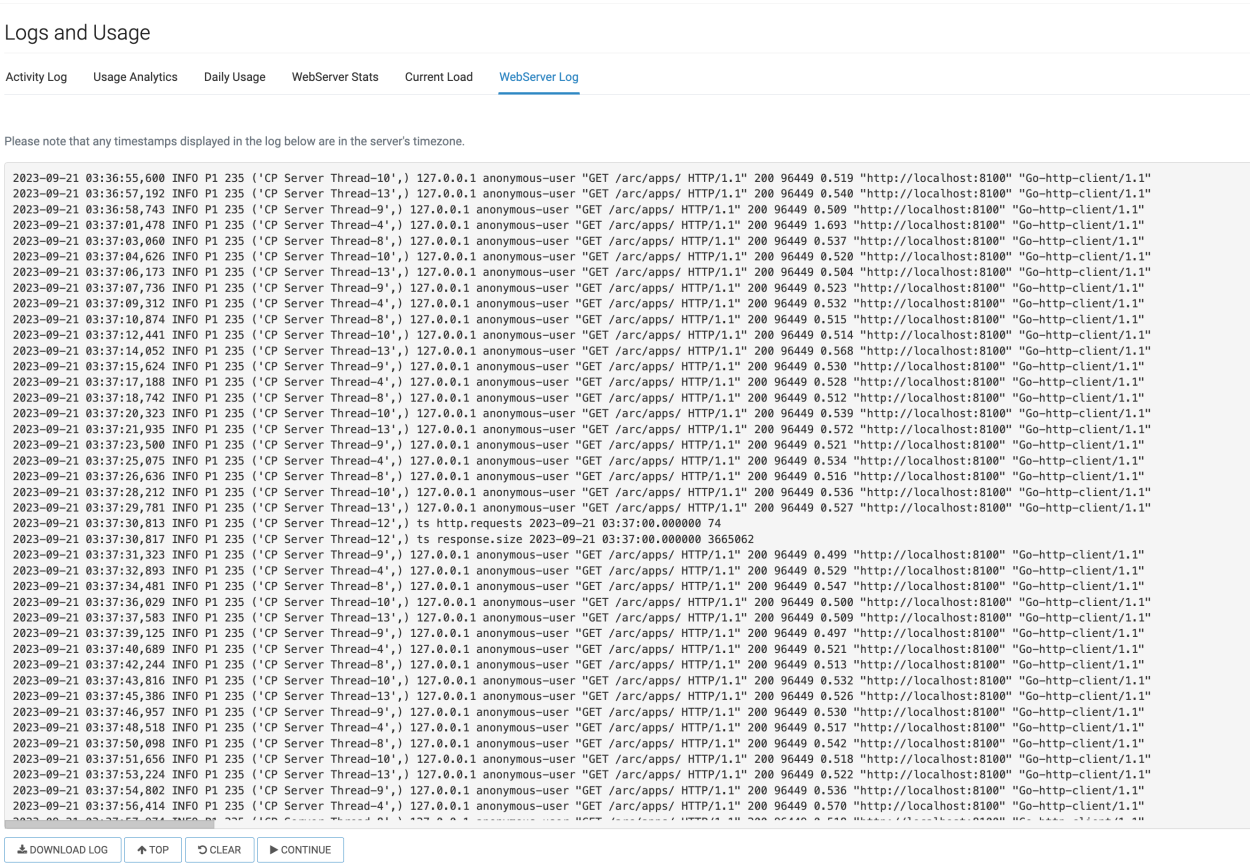

1. WebServer Activity log window

   The activity log is presented as a flat file, featuring timestamps for each event, followed by technical details.
2. DOWNLOAD LOG

   You can save log files to your local system. The downloaded files follow the naming convention arcviz_yyyy_mm_dd_hh_mm_ss.log. For example, a file saved on April 17, 2018 at 1:46:42 P.M. has the name arcviz_2018_04_17_13_46_42.log.
3. TOP

   You can use this option to swiftly return to the top of the activity list after scrolling down.
4. CLEAR

   You can use this option to empty the log window ensuring a clean slate for new log data.
5. PAUSE/CONTINUE

   You can use this option to temporarily suspend/resume log reporting.

Cloudera Data Visualization 7.2.9

Advanced Analytics

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Advanced Analytics overview 4

Complex data types 5
- STRUCT data type 6
- ARRAY data type 8
- MAP data type 12

Dimension hierarchies 13

Segments 14

Events 14

Visuals for analytics 15
- Combined bar/line visuals 15

Advanced visualization techniques 15
- Trellised visuals 16

Derived data 16

Advanced Analytics overview

Cloudera Data Visualization provides an extensive array of tools to support advanced analysis.

Complex data types

Cloudera Data Visualization supports native complex data type configurations, bypassing ETL workflows and avoiding unnecessary flattening at query time. Explore complex data types to leverage the powerful native SQL already optimized for complex types.

Dimension hierarchies

Cloudera Data Visualization provides full support for dimensional hierarchy modelling at dataset level, enabling smooth natural transition between granularity levels of data. For more information, see Dimension hierarchies.

Segments

Segments can be very powerful in data visualization, and form the foundation of cohort analysis and row-level access restrictions. For more information, see Segments.

Events

Identifying and managing events within your time-stamped data enables you to measure outcomes of complex workflow scenarios. For more information, see Events.

Visuals for analytics

For advanced analysis, you can create advanced visual types, such as funnels, flows, correlation flows, and histograms. For more information, see Visuals for Analytics.

Advanced visualization techniques

Advanced visualization techniques, such as trellises, dual-axis line visuals, and dimension hierarchies free you to explore the visualized data from diverse perspectives. For more information, see Advanced Visualization Techniques.

Analytic functions

On select connection types, Cloudera Data Visualization supports several analytic functions that examine overlapping groupings of data. For more information, see Supported connections.

Derived data

You can derive data from an existing query, build computed columns for re-use, and reference results in new queries. Derived data enables you to stack sub-query analytical results.

Related Information

Complex data types
Dimension hierarchies
Segments
Events
Visuals for analytics
Advanced visualization techniques
Derived data
Supported Connections

Complex data types

Cloudera Data Visualization understands native complex data type configurations, and automatically translates complex structures into intuitive drag and drop dataset field configurations.

Native support for complex datatypes allows us to bypass ETL workflows and avoid unnecessary flattening at query time. It also leverages the powerful native SQL already optimized for complex types.

Note:

- Impala connections support the STRUCT, ARRAY, and MAP complex data types.
- For Hive connections, support is limited to the STRUCT data type.

Note:

When describing how Cloudera Data Visualization supports complex data types, data and structures are adapted from the TPC-H benchmark as a Sample schema and data for experimenting with Impala complex types.

Overview of complex data types

- *STRUCT*

  A group of named fields, where each field can have a different data type, both primitive and complex.

  See STRUCT data type.
- *ARRAY*

  A list of values that share the same data type.

  See ARRAY data type.
- *MAP*

  A set of (key,value) pairs.

  See MAP data type.

The elements of an ARRAY or a MAP, or the fields of a STRUCT may be primitive, or they may be other complex types. You can construct elaborate data structures with many levels, such as a ARRAY with STRUCT elements.

Appearance of complex data types

Cloudera Data Visualization presents complex types very simply. Users can manipulate the individual component fields of complex types in the same manner as the primitive data type fields: place them on the shelves of the visual, as a component of an expressions, as a filter, and so on. Each level of a complex data type may be expanded to show component details, or collapsed.

Cloudera Data Visualization processes complex types as Dimensions, grouping individual elements under the complex type. The components of a complex type cannot be re-assigned to the Measures group even if they logically represent a measurement; however, you can easily use them as measurements on visual shelves, in filters, and so on.

Because native SQL processing is supported, you do not have to worry about the complexities of query generation to access these elements.

Restrictions

Complex data types have the following implementation restrictions:

- MAP, ARRAY, and STRUCT fields require base tables or partitions that use the Parquet file format.
- You cannot use the fields that contain complex data types as partition keys in a partitioned table.
- The Compute Statistics operation does not work for complex data types.

- The column definition for any complex type, including nested types, can have a maximum of 4000 characters.
- The Import into Dataset feature does not support files that contain complex datatypes.
- Analytical Views do not support complex type queries, because complex types require JOIN operations. As a result, visuals that use complex data types do not have valid recommendations. However, if you associate a logical view with the dataset, then the recommendation engine proceeds to generate suggestions, and presents the relevant dataset columns as complex data types.

Related Information

TPC-H
Sample Schema and Data for Experimenting with Impala Complex Types
STRUCT data type
ARRAY data type
MAP data type

STRUCT data type

This article describes the specifics of the STRUCT complex data type.

Syntax for STRUCT

```
column_name STRUCT < name : type [COMMENT 'comment_string'], ... >
type ::= primitive_type | complex_type
```

A STRUCT has a fixed number of named fields, and each field can be a different type. A field within a STRUCT can also be another STRUCT, an ARRAY, or a MAP

STRUCTs in the Dataset Field interface

In the Dataset Fields interface, a basic STRUCT data type may look something like the following example. You can see that each level of a complex data type can be expanded to show component details, or collapsed for simplicity.

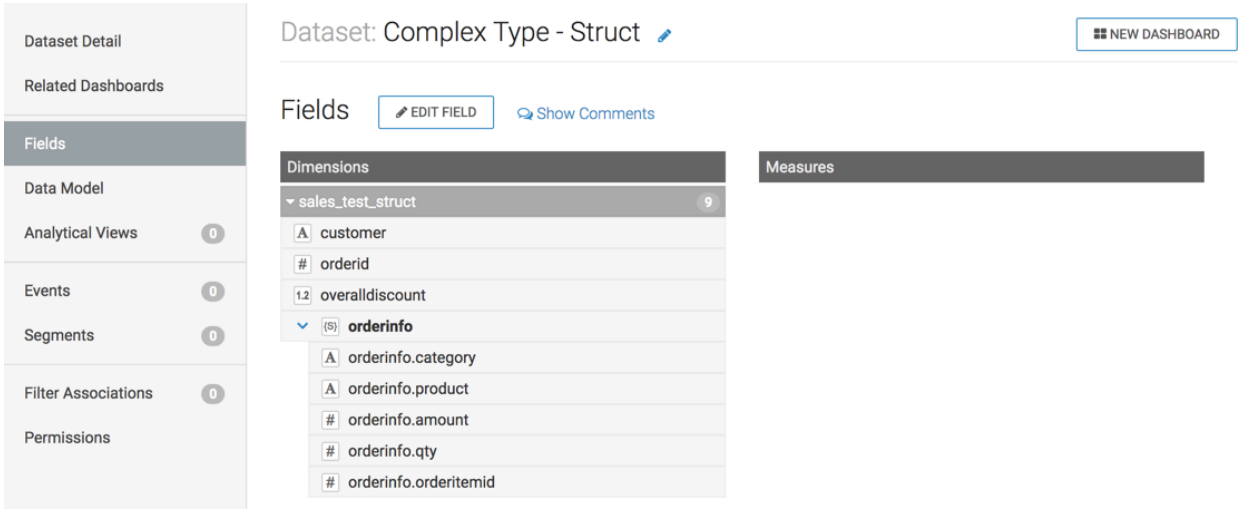

In the example of the dataset Complex Type - Struct, you can see that the Dimensions Customer (String, marked with the symbol A), Orderid (Integer, marked with the symbol #), and overalldiscount (Real, marked with the symbol 1.2) are primitive types. However, orderinfo is a Struct data type, marked with the symbol [S].

When you click Edit Fields, you can see that while primitive types can be cast as alternate data types (such as Integer into Real), the complex data type STRUCT cannot be changed to another type. However, the primitive components of the array may be cast as other primitive data types. Additionally, unlike other data types, Cloudera Data Visualization

uses complex data types only as Dimensions. They or their components cannot be redefined as Measurements of the dataset.

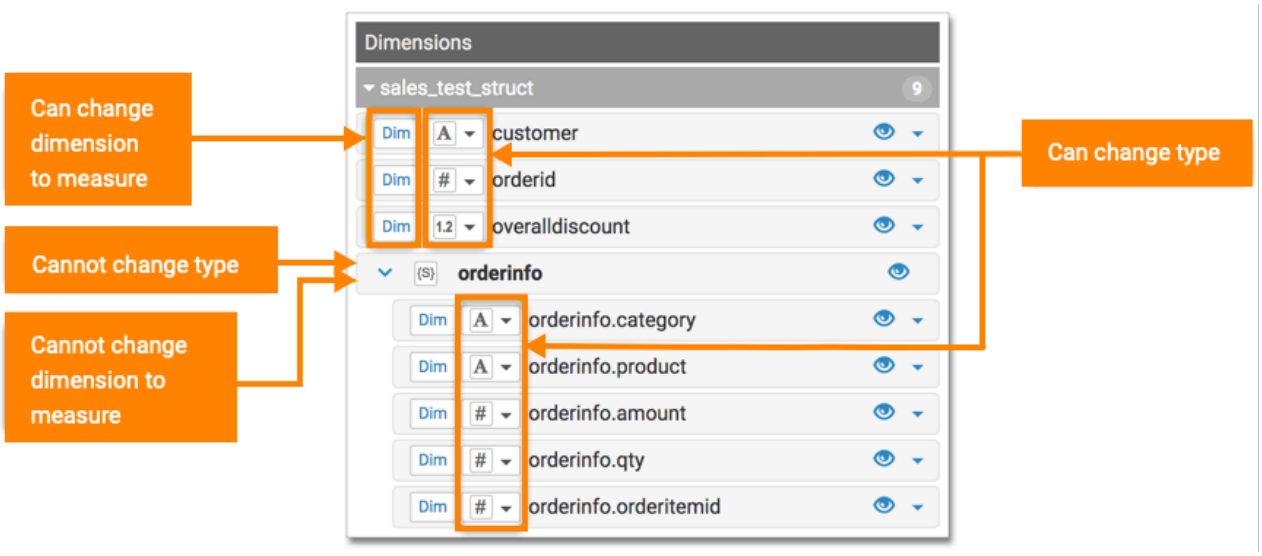

STRUCTs in visuals

When building a visual with complex data, you cannot use the complex type directly as a whole. However, you can add the primitive components of the complex type to the shelves of the visual.

For example, in a Pie visual, you might place Customer Name on the X Trellis shelf, the orderinfo.category component on the Dimensions shelf, and the orderinfo.amount component on the Measures shelf.

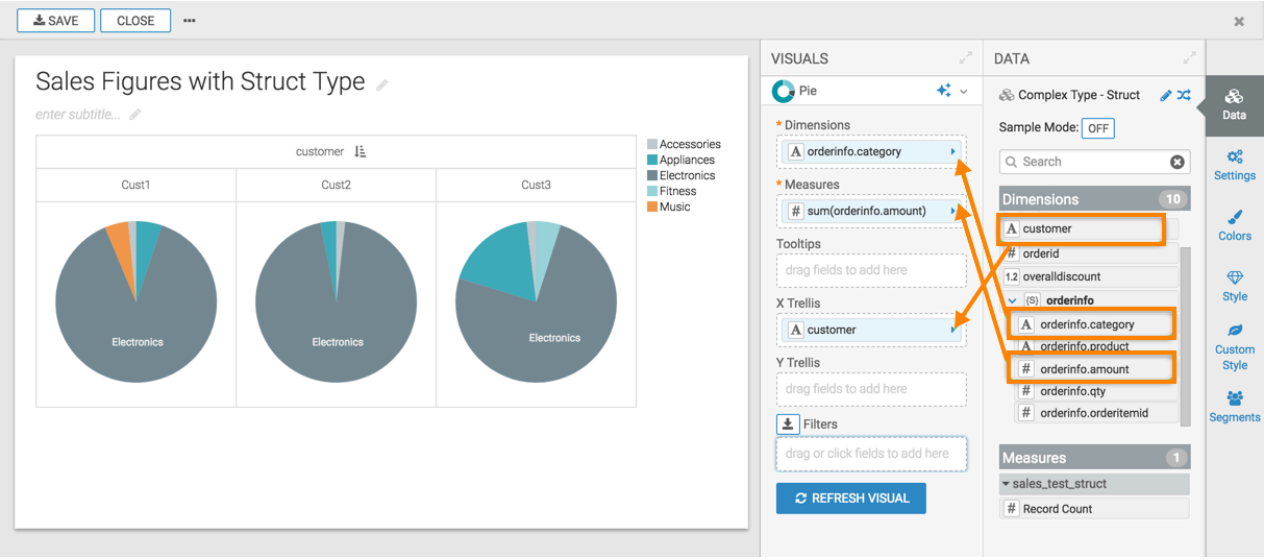

Changing field properties

It is very simple to change field properties for a component of a complex data type.

For example, you can change the orderinfor.qty component on the Tooltips shelf from the default max() aggregation to the count() function.

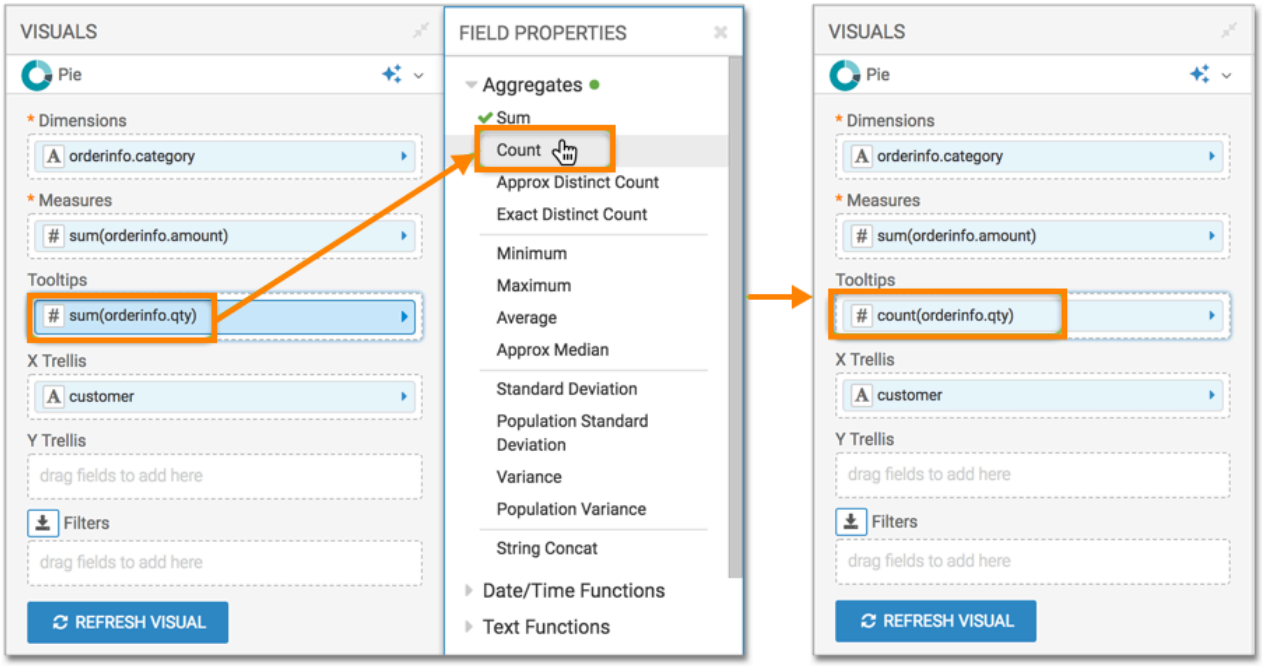

STRUCT in Expression Editor

The expression editor fully supports the use of STRUCTs, both in the Dataset and Visual interfaces. This enables advanced data manipulation and customization, leveraging the capabilities of STRUCT data type in your data visualization tasks.

ARRAY data type

This article describes the specifics of the ARRAY complex data type.

Syntax for ARRAY

```
column_name ARRAY < type >
type ::= primitive_type | complex_type
```

ARRAY data types represent collections with arbitrary numbers of elements, where each element is the same type. An ARRAY type is like a miniature table, with two columns:

POS

Position of element in the array

Access as array_name.pos

ITEM

Value of array element that may be a scalar, or another complex type (another ARRAY, a STRUCT, or a MAP)

Access as array_name.item

If an array contains a STRUCT, access as array_name.item.field_name, or as array_name.field_name

Arrays in the Dataset Field interface

In the Dataset Fields interface, an example of a basic ARRAY data type may look like the following example. You can see that each level of a complex data type may be expanded to show component details, or collapsed for simplicity.

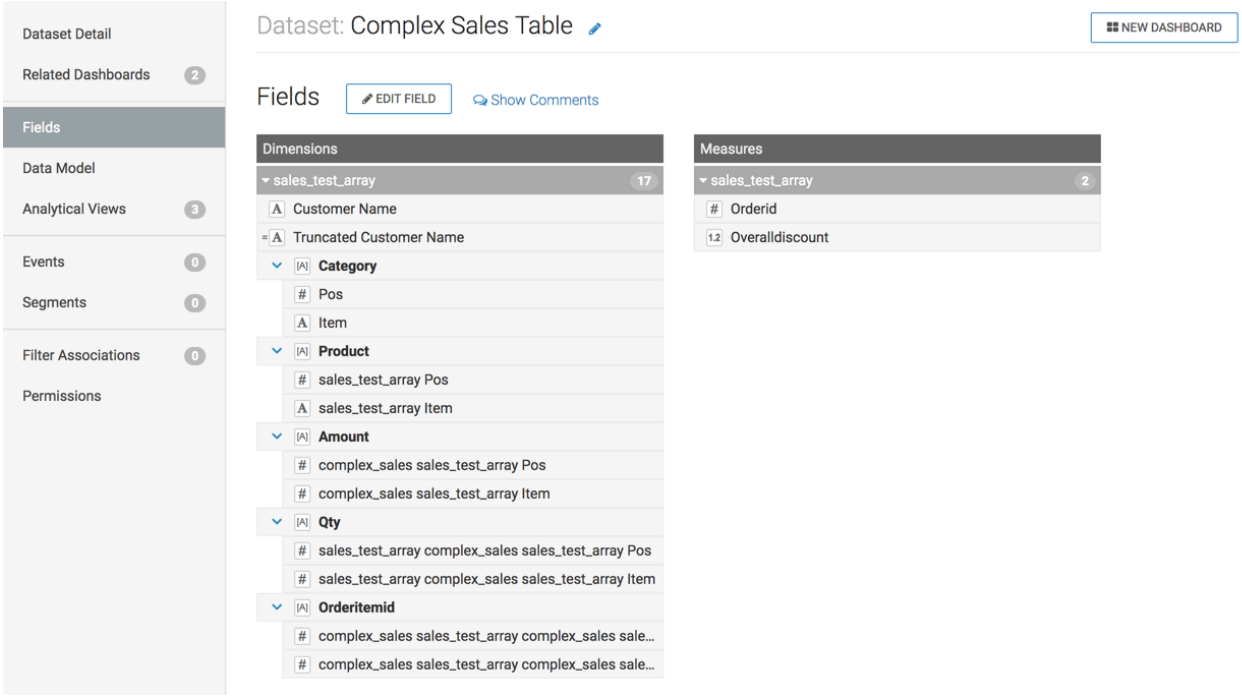

In the example of the dataset Complex Sales Table, you can see that the Dimensions Customer Name and the derived Truncated Customer Name are primitive types (both are Strings, marked with the symbol A), along with the Measures Orderid (Integer, marked with the symbol #) and Overalldiscount (Real, marked with the symbol 1.2). However, the Dimensions Category, Product, Amount, Qty and Orderitemid are all Array data types, symbolized by [A].

When you click Edit Fields, you can see that while primitive types can be cast as alternate data types (such as Integer into Real), the complex data type Array cannot be changed to another type. However, the primitive components of the array may be cast as other primitive data types. Additionally, unlike other data types, Cloudera Data Visualization uses complex datatypes only as Dimensions. They or their components cannot be redefined as Measurements of the dataset.

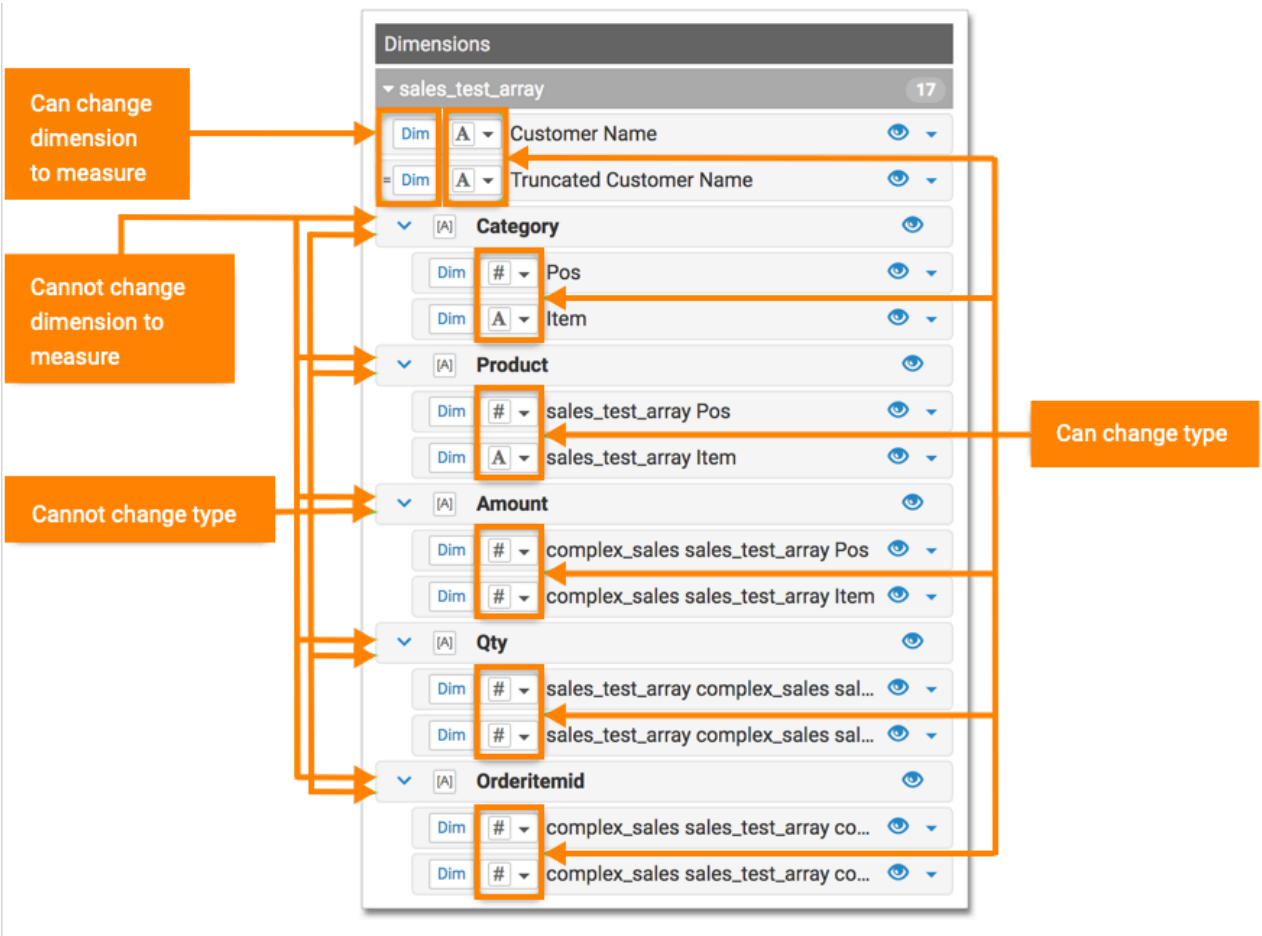

Arrays in visuals

When building a visual with complex data, you cannot use the complex type directly, as a whole. However, you can add the primitive components of the complex type to the shelves of the visual.

For example, in a Bars visual you might place Customer Name on the X Axis shelf, the Amount:Item component on the Y Axis shelf, and grouped on the Colors shelf by Product:Item component.

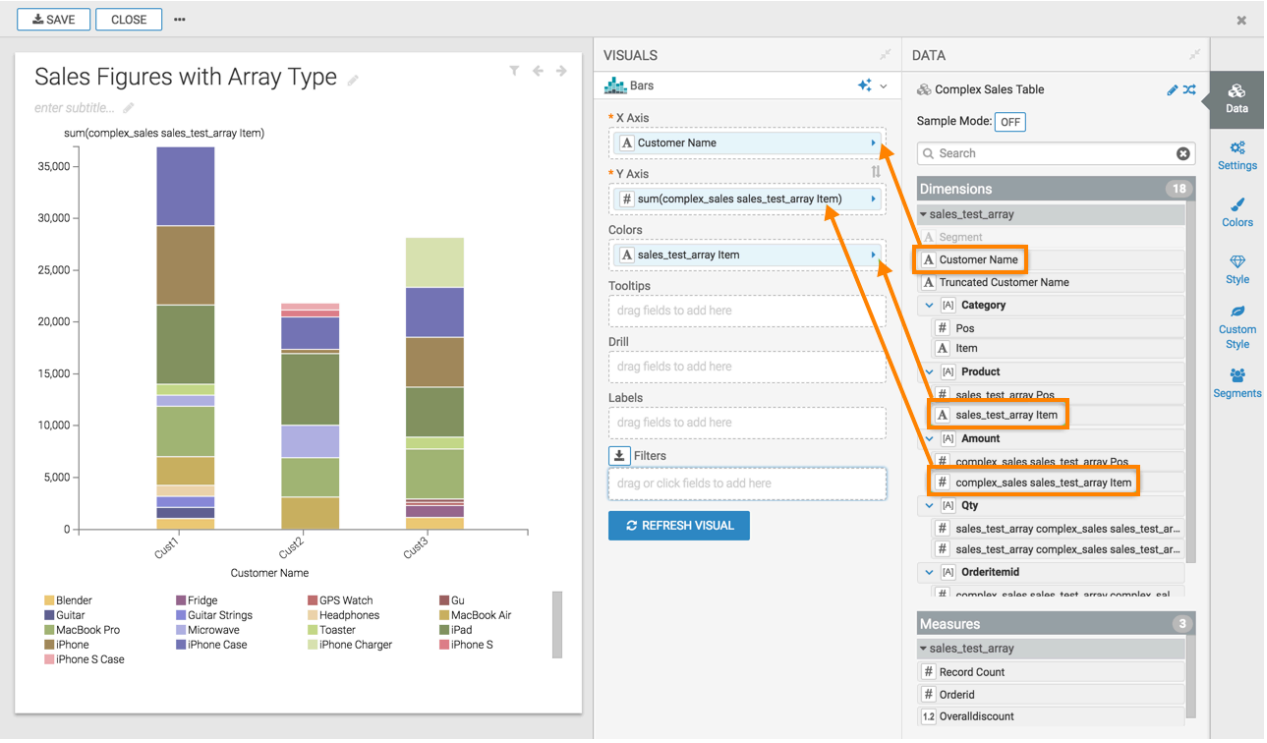

Changing field properties

It is very simple to change field properties for a component of a complex data type.

For example, you can change the Product:Item component on the Tooltips shelf from the default max() aggregation to the count() function.

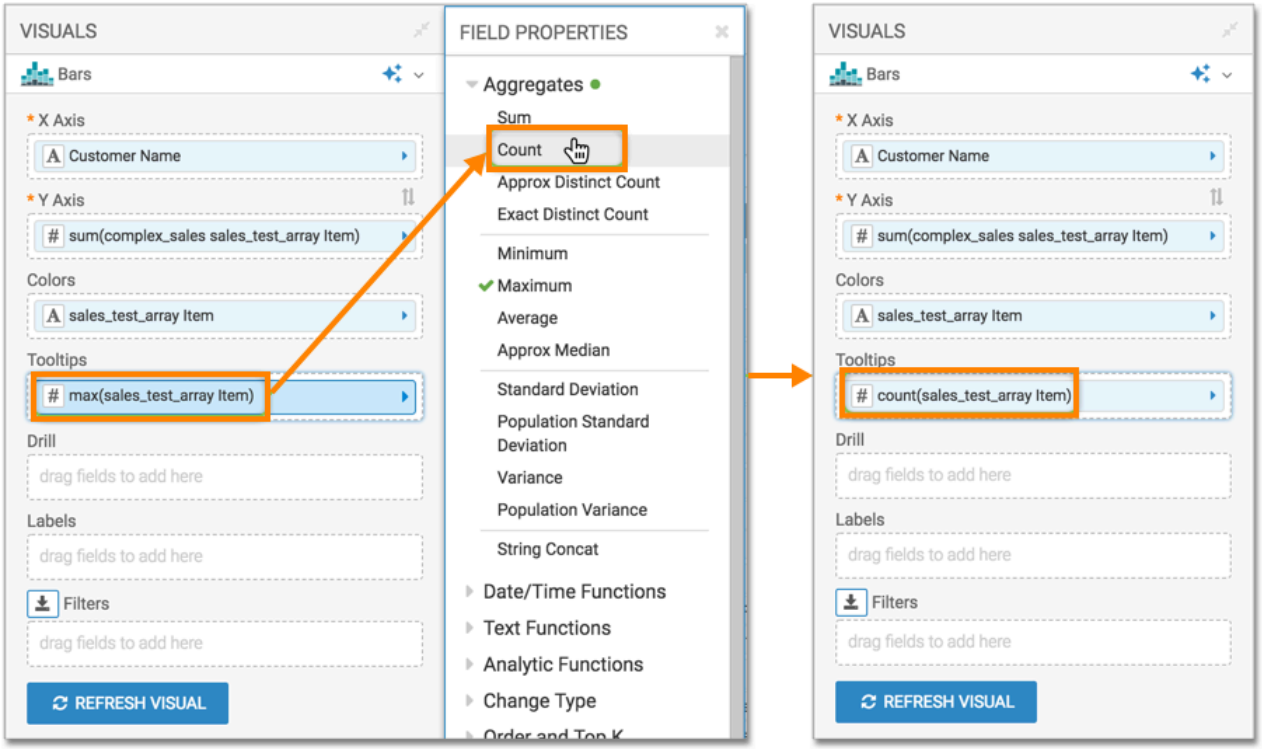

Arrays in Expression Editor

The expression editor fully supports the use of Arrays, both in the Dataset and the Visual interfaces. This enables advanced data manipulation and customization, leveraging the capabilities of ARRAY data type in your data visualization tasks.

MAP data type

This article describes the specifics of the MAP complex data type.

Syntax for MAP

```
column_name MAP < primitive_type, type >
type ::= primitive_type | complex_type
```

MAP data types represent sets of key-value pairs. These collections have an arbitrary number of elements, with each element being the same type. A MAP type is similar a miniature table, the table equivalent of scalar (primitive) values with two columns.

KEY

- A scalar type, such as BIGINT, STRING, or TIMESTAMP
- It enables you to define non-continuous sequences or categories with arbitrary names.
- Keys in a map column may represent a numeric sequence of events during a manufacturing process or TIMESTAMP values that correspond to sensor observations.
- Access as map_name.key.

VALUE

- The second part of a key-value pair in a map.
- May be a scalar, or another complex type (an ARRAY, a STRUCT, or another MAP).
- Access as map_name.value.
- If a map contains a STRUCT, access as map_name.value.field_name, or as map_name.field_name.

MAPs in the Dataset Fields interface

In the **Dataset Fields** interface, an example of a basic MAP data type may look something like the following image. Each level of a complex data type can be expanded to show component details, or collapsed.

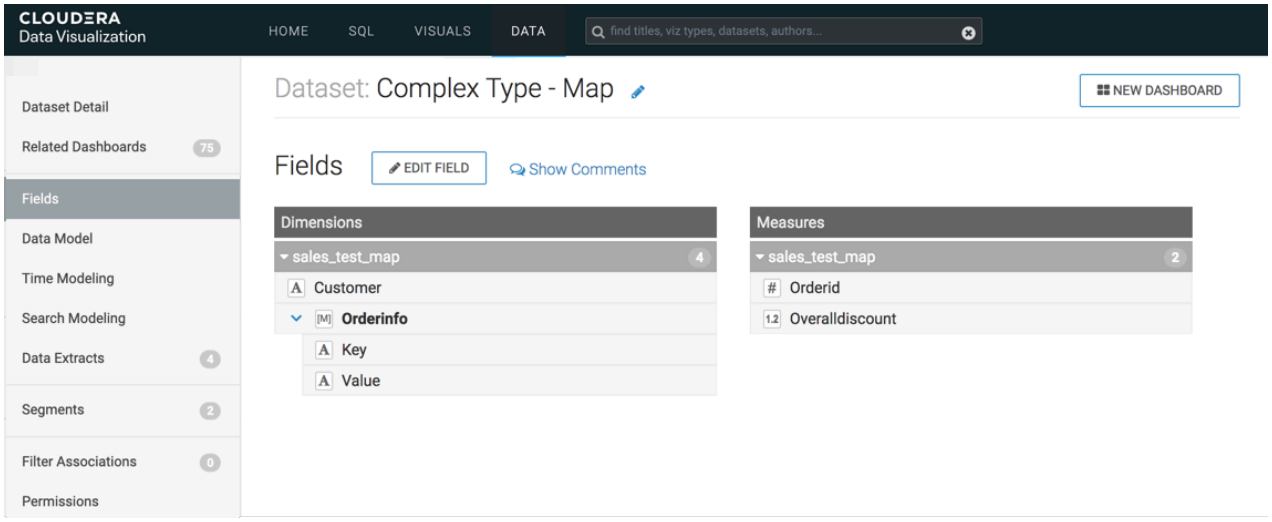

In the example of the dataset Complex Type - Map, you can see the following:

- Dimension Customer (a String, symbolized by A)
- Measures Orderid (an Integer, symbolized by #)
- Overalldiscount (a Real, symbolized by 1.2)

These are primitive types. However, Dimension Orderinfo is a Map data type, symbolized by [M].

When you click Edit Fields, you can see that:

- Primitive types can be cast as alternate data types (such as Integer into Real).
- Complex data type Array cannot be changed to another type.
- Primitive components of the array may be cast as other primitive data types.
- Cloudera Data Visualization uses complex datatypes only as Dimensions. They or their components cannot be redefined as Measurements of the dataset.

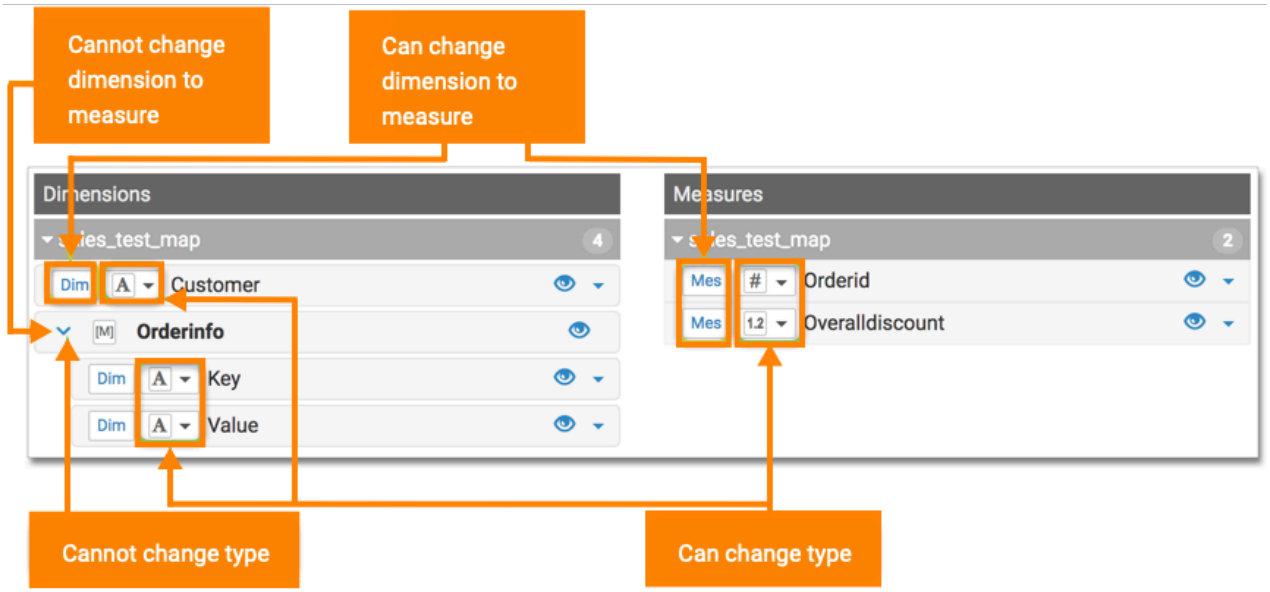

MAPs in visuals

When building a visual with complex data, you cannot use the complex type directly. However, you can add the primitive components of the complex type to the shelves of the visual.

Changing field properties

Changing field properties for a component of a complex data type is straightforward. You can change the aggregation function, type, display options, and so on, just like a standalone primitive type field.

MAPs in Expression Editor

The expression editor supports the full use of MAPs, both in the **Dataset** and **Visual** interfaces.

Dimension hierarchies

Cloudera Data Visualization provides full support for dimensional hierarchy modelling at dataset level, enabling smooth natural transition between granularity levels of data.

Hierarchies are a natural form of data organization. For example, you can break up any given time interval into intervals of shorter duration based on years, months, weeks, days, and so on. Similarly, the territory of United States consists of well-defined states, then counties, then municipalities and incorporated and unincorporated areas.

Hierarchies may only be defined through columns that are classified as dimensions. If necessary, change the column attribute from Measure to Dimension. Alternatively, clone a column, change it to a dimension, and only then use it in defining a dimensional hierarchy.

Just like with column names, the name of the dimensional hierarchy must be unique to the dataset.

Columns that define a dimension hierarchy are retained as a reference, not a copy. Therefore, changes to the basic definition of the column (such as name, type, calculation) propagate into the dimensional hierarchy.

To build out dimensional hierarchies, use 'drag and drop' action to add dimension columns, and to prioritize them. Cloudera recommends that you use the delete function to permanently remove items from a hierarchy. You can also move elements from one hierarchy to another; note that this action removes the dimension from the source hierarchy and adds it to the target hierarchy.

Segments

With Cloudera Data Visualization, you can create and manage segments within your data.

Segments can be used to analyze subsets of the data that have specified properties. Normally, segments are defined by filtering data based on dimensions or measures. Discovering trends within segments and making comparisons across segments provides powerful insights into the data and the business it represents.

To create and manage segments in the Dataset Segments user interface, see the following topics:

- Creating Segments
- Cloning Segments
- Editing Segments
- Creating Entity Segments
- Deleting Segments

You can also segments from filter definitions in a visual. See, Creating segments from filters

To view how segments work in a visual, see Using segments in visuals.

To see how you can use natural segments to restrict data access through filter association, see Setting segments .

Related Information

Creating segments
Cloning segments
Editing segments
Creating entity segments
Deleting segments
Creating segments from filters
Using segments in visuals
Setting segments

Events

With Cloudera Data Visualization, you can identify and manage events within your time-stamped data.

Defining events in a time-stamped dataset enables you to sequence them chronologically, and to define business scenarios that can be modelled by flow and funnel visuals.

Related Information

Creating events
Cloning events
Editing events
Deleting events
Correlation flows

Visuals for analytics

Cloudera Data Visualization has a rich catalog of visual types to assist you in analyzing your data.

The following visual types may be of more interest for advanced analysis:

Related Information

Combo visuals
Gauges
Bullets
Box plots
Networks
Dendograms
Correlation flows
Histograms

Combined bar/line visuals

In Cloudera Data Visualization, Combined Bar/Line (Combo) visuals unite bar and line plots to enable easy comparison of trends and measures across the same dimensions.

Note: This visual type supports smooth rendering of streaming data.

The bars and the line can share the same vertical axis, or use a secondary axis. For more information, see Creating basic combo visuals and Assigning a measure to a secondary axis.

You can optionally render the line portion of the visual as an area. For more details, see Change the line to an area.

Related Information

Creating basic combos
Assigning a measure to a secondary axis
Change the line to an area

Advanced visualization techniques

Cloudera Data Visualization offers advanced analytical and visualization options that work with its applications.

The following examples illustrate some advanced analysis techniques:

Related Information

Analyzing multiple measures
Creating line visuals with dual axes
Working with dimension hierarchies

Working with hierarchical cross-tabulation visuals
Adding highlights
Highlighting elements by hover-over legend

Trellised visuals

Trellis charts are grouped sets of visuals that represent different partitions (segments) of the dataset in the same chart type, scale, and axes. This makes the comparison of sub-groups very intuitive. Trellis charts are sometimes called lattice charts, grid charts, panel charts, or small multiples.

Any chart type that uses the X and Y axes explicitly may be trellisable, depending on the dataset. What is required is an extra entry on either the X or Y shelf; the first field on the shelf represents the segment definition of the trellis, and the second field is the actual dimension. Some charts types have optional X and Y shelves so that trellises may be enabled.

A dataset may be double-trellisable — you could declare a 'partition' on both X and Y shelves at the same time.

There are two approaches for trellises:

- Commonly, we use Trellising on dimensions to show the sections of data in separate but related charts.
- We can also show different data aggregates, side by side, by placing extra measurements on the Measures shelf, to Trellis on measures.

You can find specific examples of trellised charts in the following articles:

- Radials
- Areas
- Word clouds

Some visual types (bars, lines, areas, and grouped bars) support trellising on measures.

Related Information

Trellising on dimensions
Trellis on measures
Radials
Areas
Word clouds

Derived data

Derived data lets you to reference results in new queries, query stacking, and eases cohort analysis. Cloudera Data Visualization uses derived data for computed fields in data modeling, weighted sums and averages, custom binning, for set-based and group-based analysis, and for combining data from different data sources.

Derived data enables you to reference query results in new queries, in essence "stacking" results from sub-select queries. The derived data feature also supports cohort analysis, where a set of data from a report is used (joined back) in another report, and allows you to build computed columns for re-use.

Note: By default, the Derived Data option is turned off. It can be turned on by a user with administrative privileges; see Enabling derived data.

Derived Data is very useful in determining weighted averages and other, more complex calculations. For example, in the dataset World Life Expectancy, life expectancy is reported at the level of each country, for each year. If you wanted to determine the life expectancy by region or subregion, you have to calculate a weighted average of life expectancies. You can also parametrize derived data definitions using bracket notation.

The following steps demonstrate how to use derived data on a table visual based on the dataset World Life Expectancy [data source samples.world_life_expectancy].

1. Place the fields un_region, un_subregion and country on the Dimension shelf.
2. Place the field population, as sum(population), on the Measures shelf.
3. Place the field year on the Filters shelf, and change the expression to [year]=<<year_param:2000>>. This enables you to dynamically change derived data calculations. You must specify a default value in the parametrized expression.

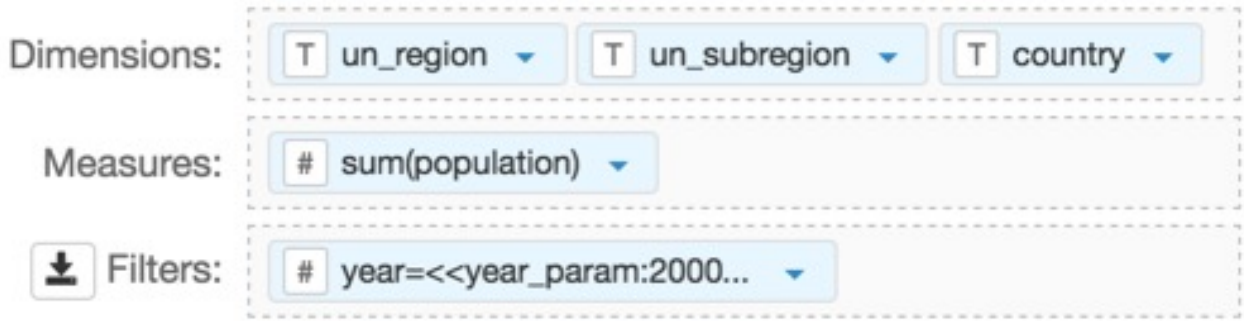

Related Information

Defining derived data
Defining additional derived data
Using derived data
Viewing derived data definitions
Saving derived data
Deleting derived data definitions

Cloudera Data Visualization 7.2.9

Advanced Visualization Techniques

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Trellis on dimensions....4

Trellis on measures....7

Multiple measure analysis.... 9

Line visuals with dual axes....11

Working with dimension hierarchies.... 13

Hierarchical cross tabulation visuals....17

Adding highlights to visuals.... 26

Trellis on dimensions

Trellising charts on dimensions produces multiple charts that focus on a distinct section of data.

Trellising on one dimension

In this example, the field un_region becomes a trellising dimension when it is placed on the X shelf of a map visual.

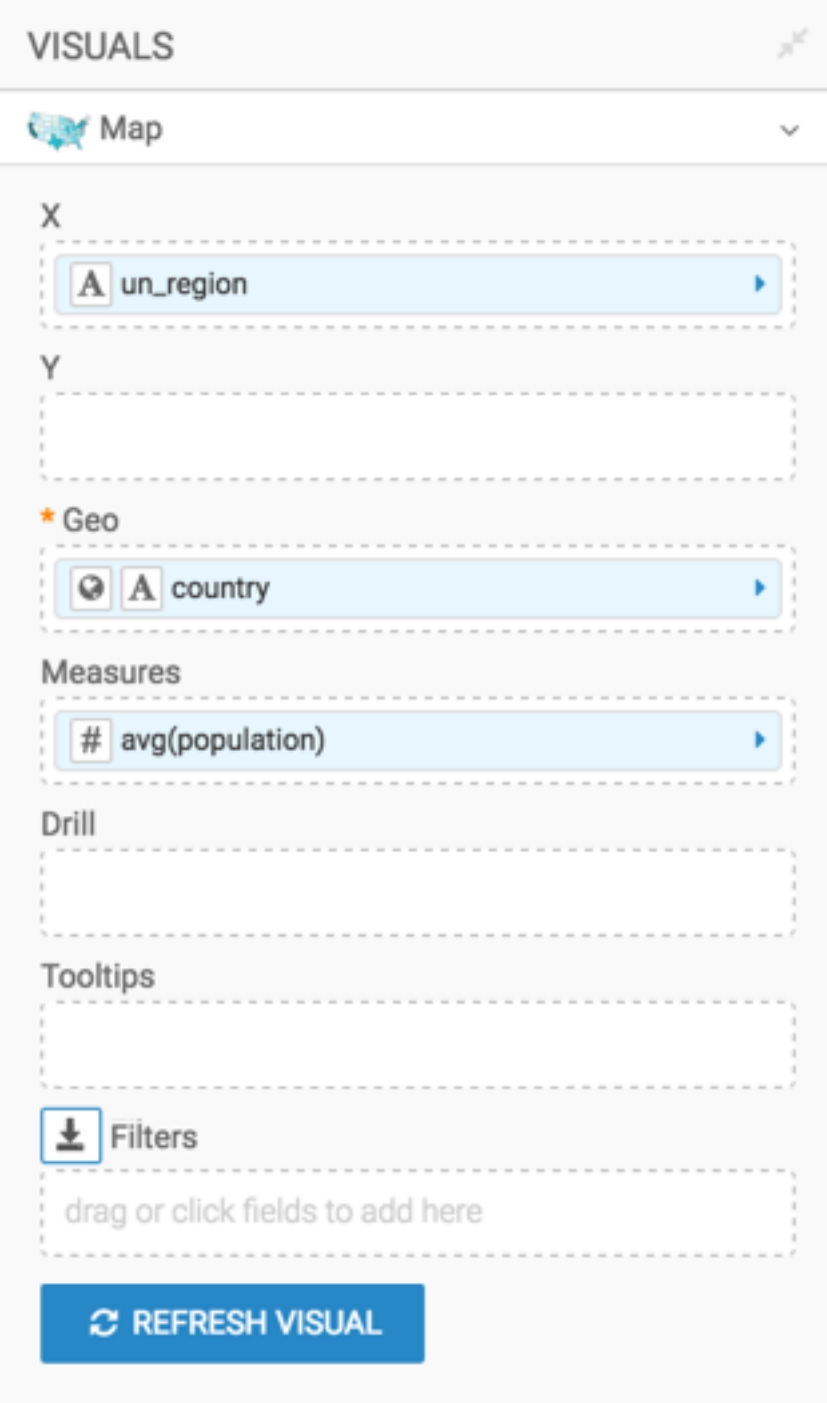

This setup produces a trellised version of a map visual, on one dimension.

You can trellis many of the charts where the X and Y shelves are optional, by placing the field that represents the partition there.

When using mandatory shelves, such a X Axis and Y Axis on bar visuals, place two dimensions on the shelf. Cloudera Data Visualization uses the first field to trellis the visual, and the second field in the visual itself.

Trellising on two dimensions

You can take it one step further, and define a secondary trellis for your visuals by placing yet another dimension on the Y shelf.

Here, the year field is used both on the Y shelf and on the Filters shelf, selecting years 1910, 1960, and 2010 to demonstrate the comparable populations across continents, and in 50-year increments.

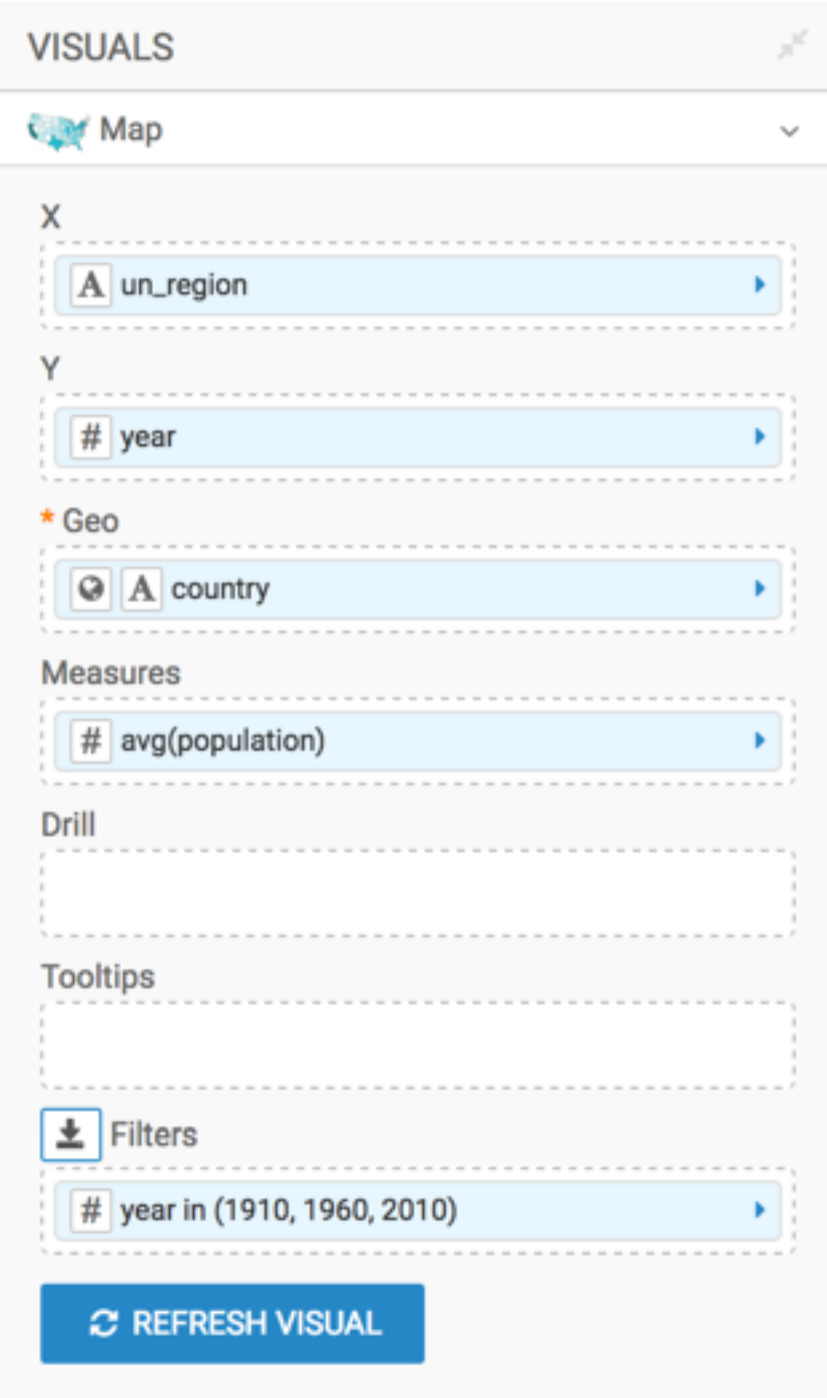

This setup produces a map visual that is trellised on two dimensions, UN Region and Year.

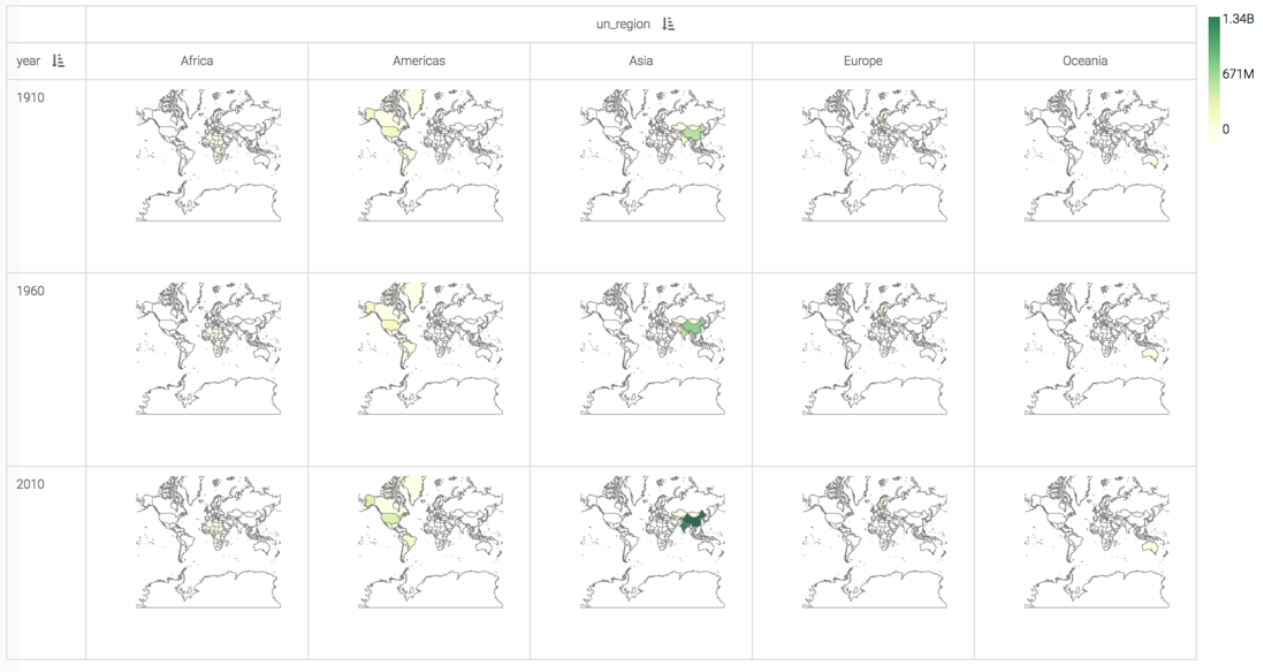

Trellis on measures

Some visual types (bars, lines, areas, and grouped bars) may be trellised on measures, simply by adding more fields to the measurement shelf.

About this task

You can change the chart developed in *Bars* (Country / Life Expectancy) into a visual trellised on three measures: Life Expectancy, Population, and GDP per Capita. You can accomplish this by placing all three measurements on the Y Axis shelf.

Procedure

1. Click the Y Axis shelf. Life Expectancy is already on the shelf.
2. With the shelf active, click the population field in the DATA menu.
3. Click the gdp_per_capita field in the DATA menu.

4. Click REFRESH VISUAL.

Note: As the magnitudes of the measurements vary, Cloudera recommends that you specify independent measurements and then click REFRESH VISUAL again.

Results

This setup produces a measurement-trellised bar chart visual.

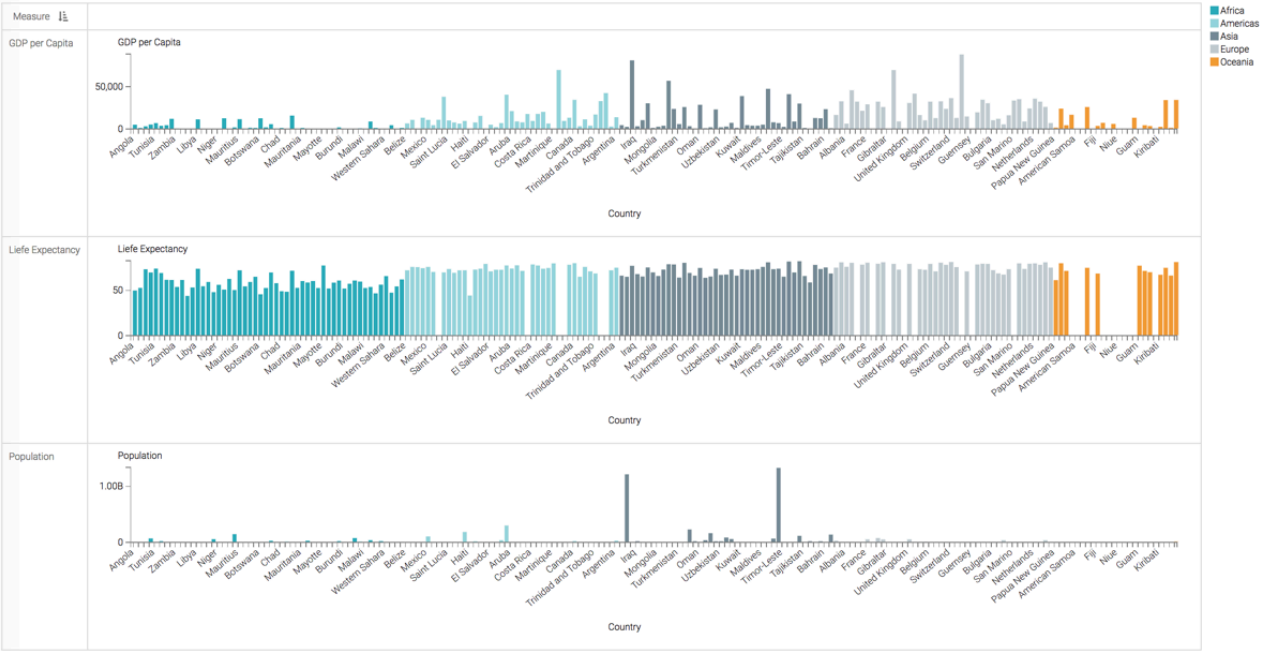

To compare this approach with plotting two measures onto the same line chart, see *Line visuals with dual axes*.

Related Information

Line visuals with dual axes

Bars

Multiple measure analysis

Cloudera Data Visualization supports measures in the trellis, for comparing multiple measures at the same time. This feature works in bar, line, area, and grouped bar visual types.

About this task

To implement this feature, simply add all the required measurement fields to the trellising shelves of the visual.

Follow these steps to create a multi-measure representation of data in the World Life Expectancy dataset.

Procedure

1. Start a new visual based on dataset World Life Expectancy [data source samples.world_life_expectancy].

   For more information, see Creating a visual.
2. In the visuals menu, choose the Lines visual type.
3. Populate the shelves from the available fields (Dimensions, Measures, and so on) in the Data menu.
   - Under Dimensions, select year and place it on the X Axis shelf. Specify ascending order, and alias the field as 'Year'.
   - Under Measures, select gdp_per_capita and place it on the Y Axis shelf.
   - Under Measures, select life_expectancy and place it on the Y Axis shelf.
   - Under Measures, select population and place it on the Y Axis shelf.
   - Under Measures, select un_region and place it on the Colors shelf. Alias the field as 'UN Region'.

4. Change the calculations and alias for each of the fields on the Y Axis using the Expression Editor:
   - Change gdp_per_capita to the following expression, and click Validate & Save.

     ```
     sum([gdp_per_capita] * [population]) / sum([population]) as 'GDP per Cap
     ita'
     ```

   - Change life_expectancy to the following expression, and click Validate & Save.

     ```
     sum([life_expectancy] * [population])/sum([population]) as 'Life Expecta
     ncy'
     ```

   - Change population to the following expression, and click Validate & Save.

     ```
     sum([population]) as 'Population'
     ```

5. In the Settings menu, open the Axes group and select the Independent Aggregate Scale option.
6. In the Settings menu, open the Axes group and select the Gridlines Enables option.
7. In the Settings menu, open the Axes group and deselect the Include Y Axis Label option.
8. Click Refresh Visual.

   The trellised line visual appears. This approach makes it easy to see the values of all measurements along the same horizontal dimension (in this case, year).

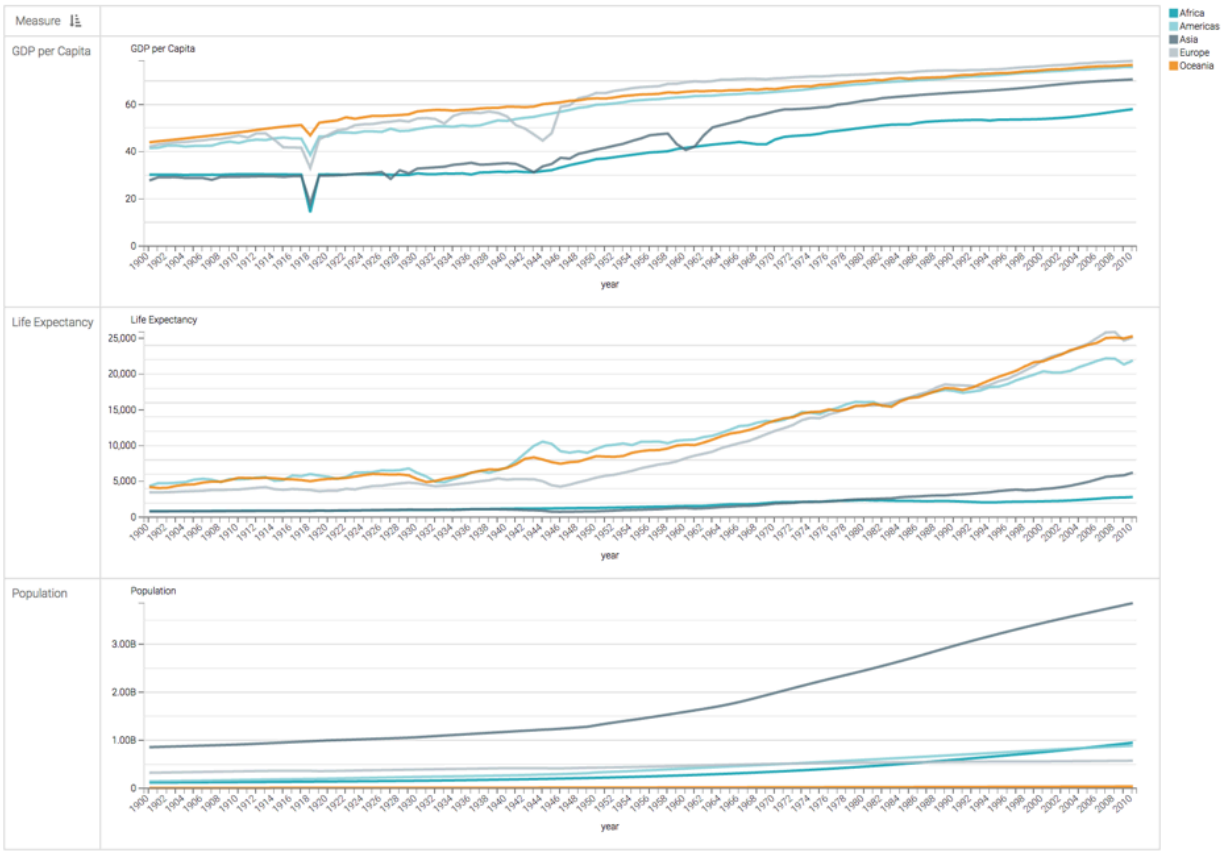

Related Information

Creating a visual

Line visuals with dual axes

It is often enlightening to see related dimensions on the same line chart; it clearly shows both correlation and pattern. However, values may be measuring completely different things, or the range of values (scale) is too different to show both lines clearly. Cloudera Data Visualization offers a dual axis option for situations like these.

About this task

The following steps demonstrate how to create a line visual that uses a double axis; it is based on dataset World Life Expectancy [data source samples.world_life_expectancy].

Procedure

1. Start a new visual based on dataset World Life Expectancy [data source samples.world_life_expectancy].

   For more information, see Creating a visual.
2. In the visuals menu, choose the Lines visual type.
3. Populate the shelves from the available fields (Dimensions, Measures, and so on) in the Data menu.
   - Under Dimensions, select year and place it on the X Axis shelf. Specify ascending order, and alias the field as 'Year'.
   - Under Measures, select life_expectancy and place it on the Y Axis shelf. Alias it.
   - Under Measures, select Population and place it on the Colors shelf. Alias it, too.
4. Click Refresh Visual.

   The line visual appears.

   It is impossible to see the values on the life expectancy line because the population line sets the scale for the Y axis.

   In this case, the label does not correspond to the scale shown, because this information comes from the Y axis shelf.

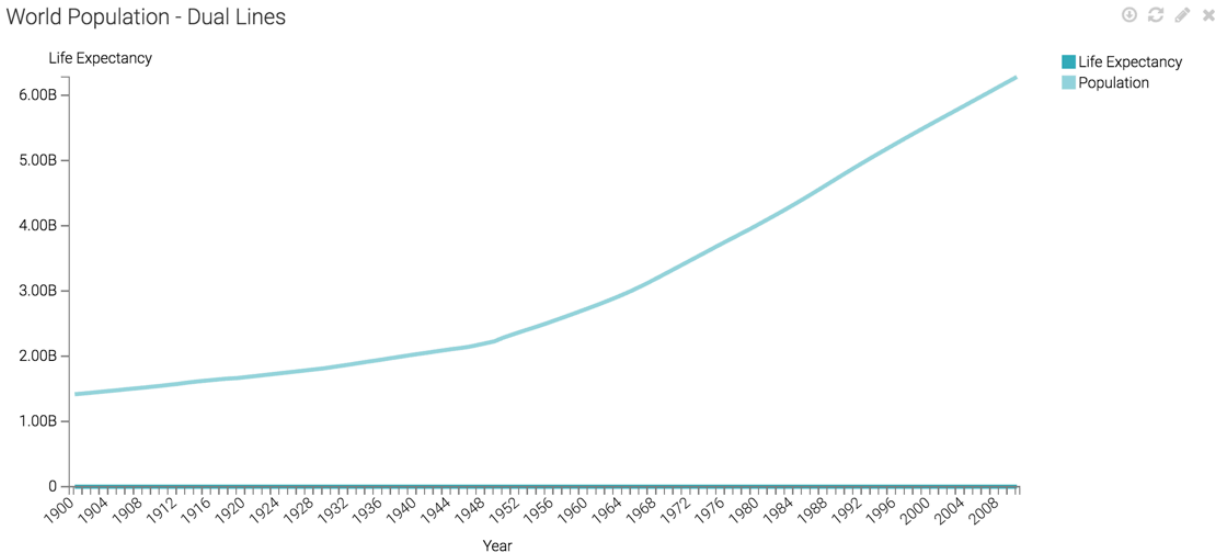

5. On the Colors shelf, on sum(population) field, click the Down arrow icon, select Axis, and then select Secondary Axis.

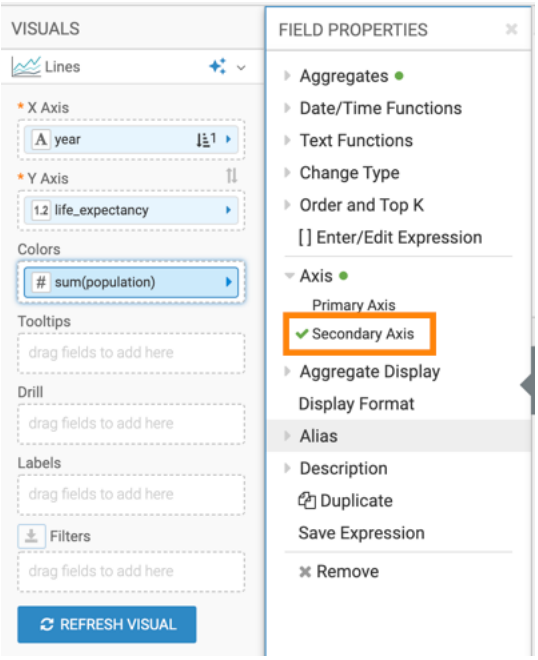

6. Click Refresh Visual.

   The two lines can now be seen clearly.

   There is now a vertical axis both on the left and the right side of the visual. For convenience, the color of the axis labels match the line.

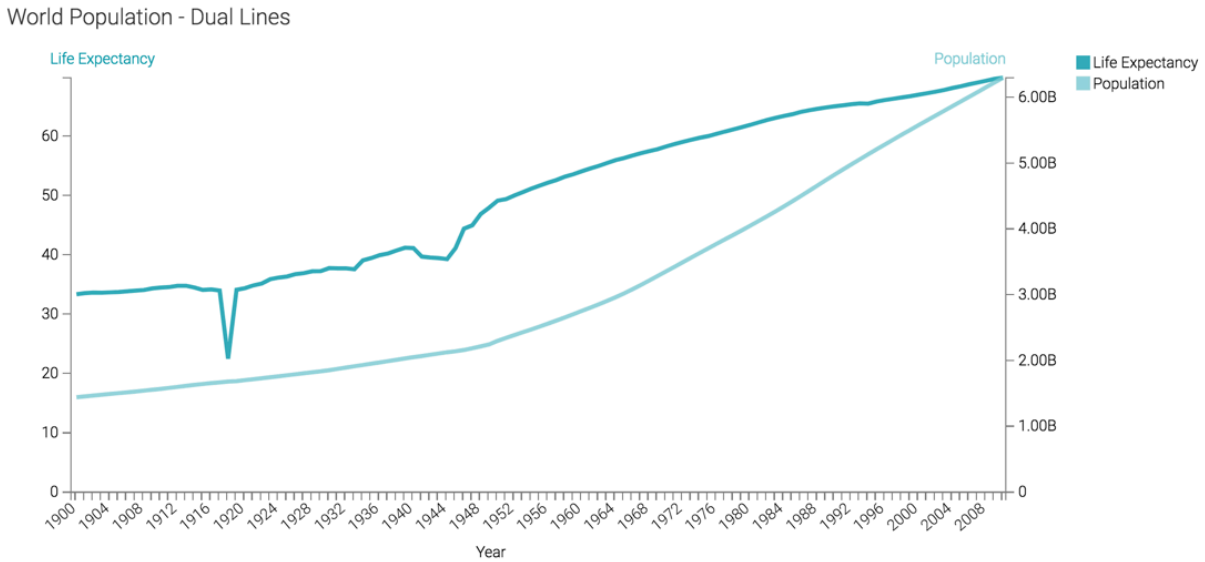

7. Save the visual as World Population - Dual Lines.

   To compare this approach with plotting multiple measures as trellised charts, see Trellis on measures.

Related Information

Trellis on measures

Creating a visual

Working with dimension hierarchies

Dimension hierarchies enable you to create complete visualizations with powerful drill down capability within a single visual, instead of relying on a complex interaction of two or more visuals inside an app.

About this task

Dimension hierarchies can be used with bar, line, area, and grouped bar visual types. Cloudera Data Visualization also provides an alternate solution for hierarchical dimension representation in cross tabulation visuals. For more information, see *Working with hierarchical cross-tabulation visuals*.

The following steps demonstrate how to use Region, a dimensional hierarchy based on dataset World Life Expectancy [data source samples.world_life_expectancy], to build a bar chart that can switch between representations at three different levels of data granularity. To create this hierarchy, see Creating new dimension hierarchies.

Procedure

1. Start a new bar visual on the World Life Expectancy dataset.

   For more information, see *Bars*.
2. In the Visual Designer interface, populate the shelves of the visual.
   - Under Dimensions, select the field Region (which is a dimensional hierarchy) and drag it onto the X Axis shelf.
   - Under Measures, select population and drag it onto the Y Axis shelf.

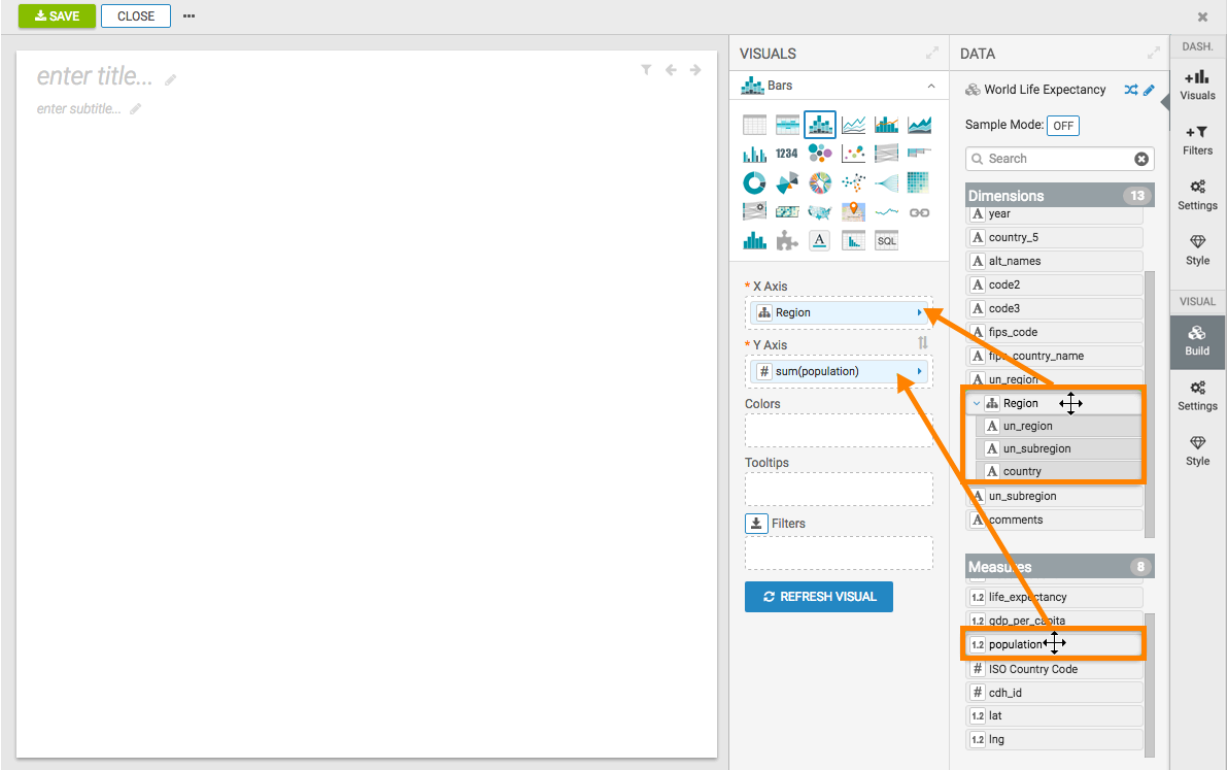

3. Click REFRESH VISUAL.

   The bar chart appears. The legend along the X axis corresponds to the top level of the Region hierarchy, un_region.

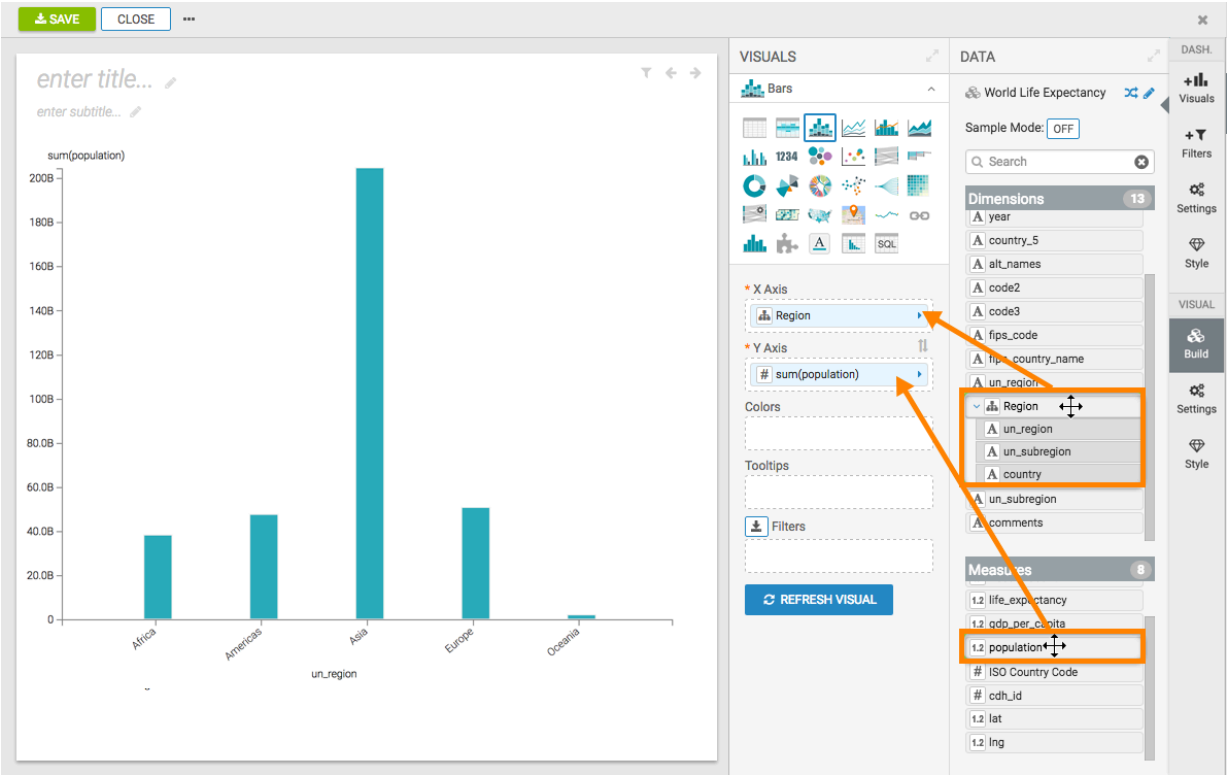

4. Change the title to World Population - Hierarchy.
5. Click Close to switch to View mode, so you can examine the operation of the dimensional hierarchy in the visual.

6. Put the pointer over one of the data labels along the horizontal axis, and note that it changes color; this is because the data labels are link-enabled. Click one of the choices to drill down to a lower level of the hierarchy.

   In the following graphic, we drilled down from Level 1: un_region = 'Americas', to Level 2: un_subregion = ' South America', and then to Level 3: country.

   Every time you move between the hierarchy levels, the scale of the vertical axis adjusts automatically.

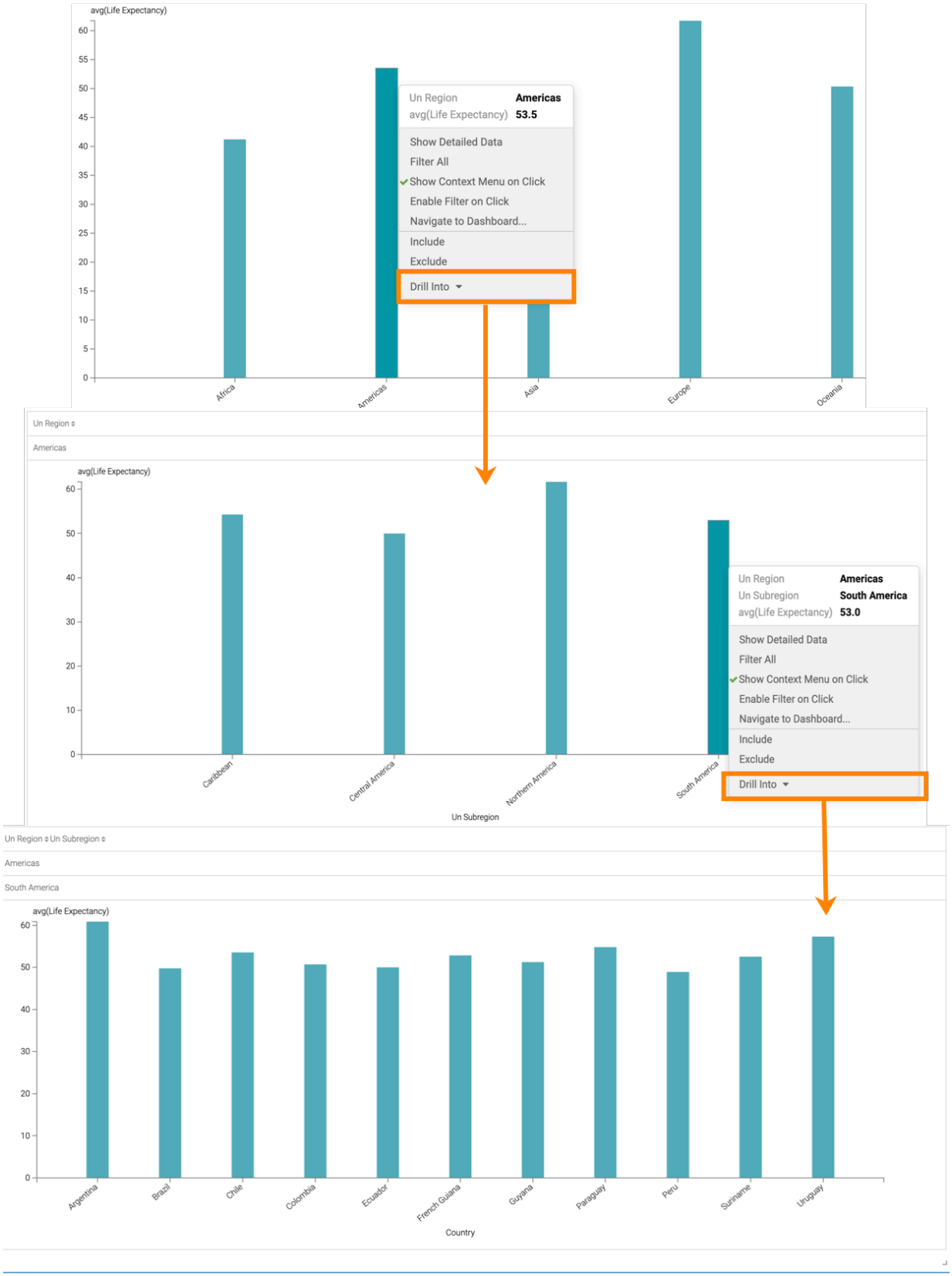
avg(Life Expectancy)
Un Region Americas
avg(Life Expectancy) 53.5
Show Detailed Data
Filter All
Show Context Menu on Click
Enable Filter on Click
Navigate to Dashboard...
Include
Exclude
Drill Into
Africa
Americas
Asia
Europe
Oceania
Un Region
Americas
Un Subregion South America
avg(Life Expectancy) 53.0
Caribbean
Central America
Northern America
South America
Un Subregion
Un Region Un Subregion
South America
Argentina
Brazil
Chile
Colombia
Ecuador
French Guiana
Guyana
Paraguay
Peru
Suriname
Uruguay
Country

7. Put the pointer over the label of the horizontal axis, and note that it change color; it is also link-enabled. You can click the horizontal axis label to go up the hierarchy levels.

   In the following graphic, we moved from Level 3: country where un_region = 'America' and un_subregion = 'Sout h America' to Level 2: un_subregion = 'South America' where un_region = 'America', and then to Level 1: un_r egion.

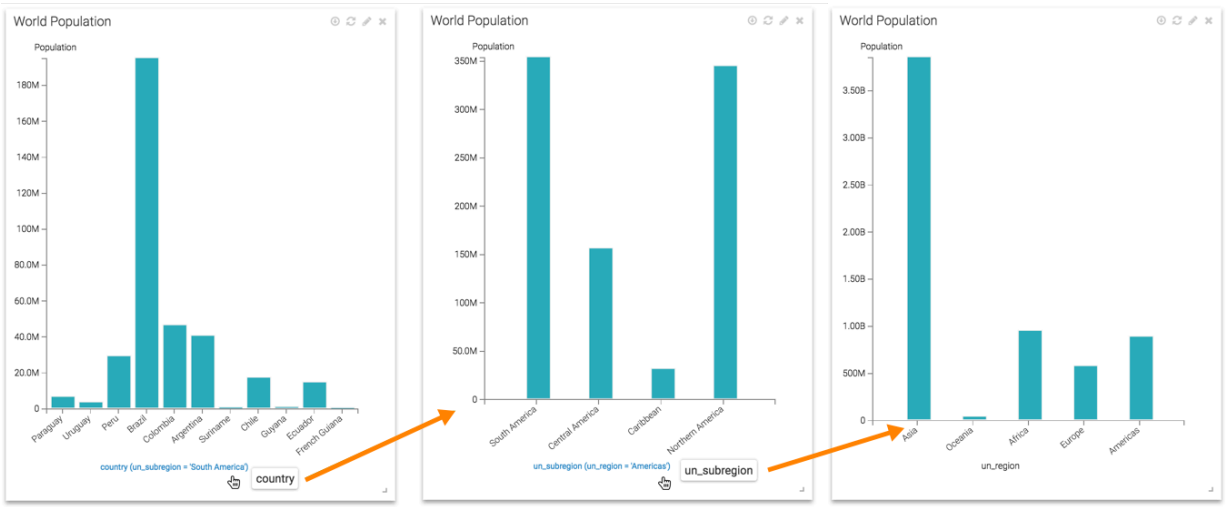

Related Information

Hierarchical cross tabulation visuals

Creating new dimension hierarchies

Bars

Hierarchical cross tabulation visuals

Cloudera Data Visualization has an alternate solution for hierarchical dimension representation in cross tabulation visuals.

About this task

The following steps demonstrate how to create and configure a de facto dimensional hierarchy on the shelves of a cross tabulation visual. We are using dataset World Life Expectancy [data source samples.world_life_expectancy], to build a cross tabulation table that can switch between representations at three different levels of data granularity.

Procedure

1. Start a new cross tabulation visual on the World Life Expectancy dataset.

   For more information, see *Cross tabulation*.

2. Populate the shelves from the available Fields under Dimensions and Measures in the Data menu.
   - Under Dimensions, select un_region and drag it over the Dimensions shelf. Repeat with un_subregion and country, adding them under un_region. Note that your placement of these fields is from the largest to the smallest, top to bottom.
   - Under Measures, select population and drag it over Measures shelf. Ensure that the default aggregation is sum(population).

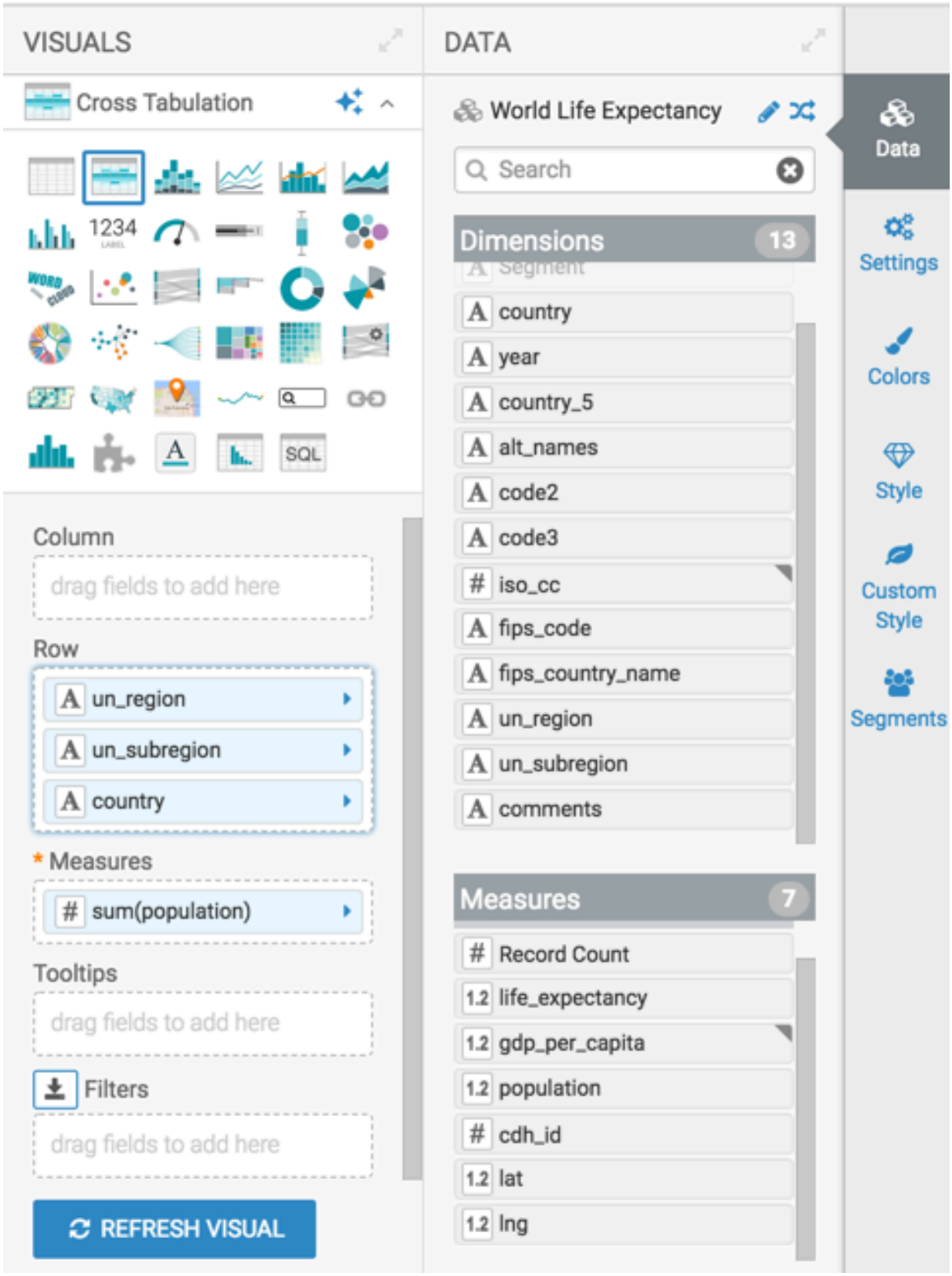
VISUALS
DATA
Cross Tabulation
World Life Expectancy
Data
Search
Settings
Dimensions 13
Segment
country
year
country_5
alt_names
code2
code3
iso_cc
fips_code
fips_country_name
un_region
un_subregion
comments
Colors
Style
Custom Style
Segments
Column
drag fields to add here
Row
un_region
un_subregion
country
* Measures
sum(population)
Tooltips
drag fields to add here
Filters
drag fields to add here
REFRESH VISUAL
Measures 7
Record Count
life_expectancy
gdp_per_capita
population
cdh_id
lat
lng

3. On the Row shelf, select the un_region field and select Enable Expansion option in the Field Properties panel.

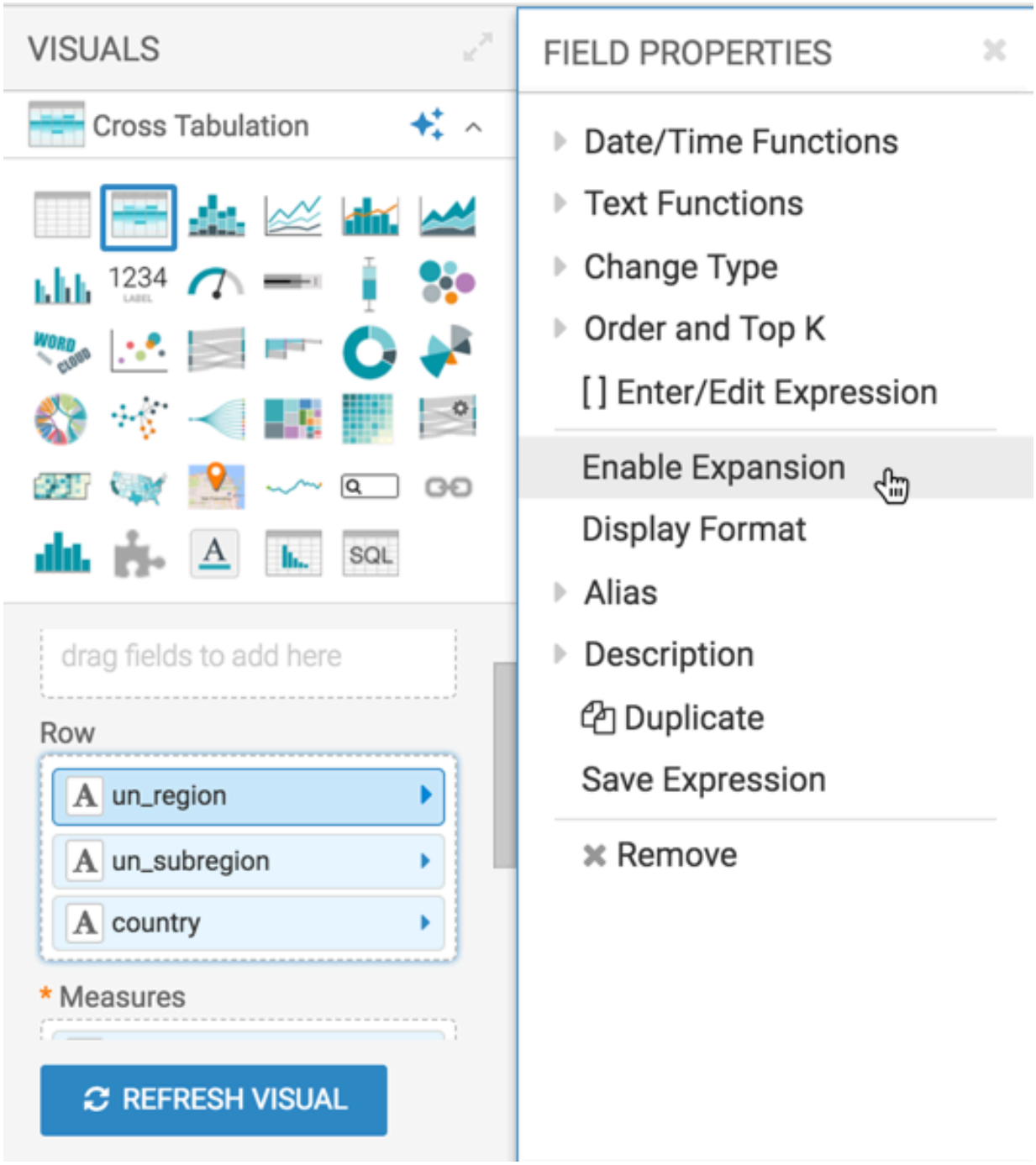

4. On the Row shelf, on the un_subregion and the country field, click the Down arrow icon, and select Enable Expansion.

5. Click REFRESH VISUAL.

   The cross tabulation table appears with a Plus icon in the un_region, un_subregion, and country columns. This indicates that you can expand the data in these columns.

   The sum(population) column shows the combined population for all un_regions.

un_region	un_subregion	country	sum(population)
⊞	⊞	⊞	342B

6. To see more details, expand the un_region column of the visual by clicking the Plus icon under the column header.

un_region	un_subregion	country	sum(population)
⊞	⊞	⊞	342B

un_region	un_subregion	country	sum(population)
⊟	⊟	⊟	342B
Africa	⊞	⊞	38.0B
Americas	⊞	⊞	47.4B
Asia	⊞	⊞	204B
Europe	⊞	⊞	50.5B
Oceania	⊞	⊞	1.88B

7. Expand the un_subregion column of the visual by clicking the Plus icon in the field next to one of the un_region. We chose to expand Africa.

   When you expand Africa, it generates a complete tabulation aggregated by un_subregion.

un_region	un_subregion	country	sum(population)
⊟	⊟	⊟	342B
Africa	⊞	⊞	38.0B
Americas	⊞	⊞	47.4B
Asia	⊞	⊞	204B
Europe	⊞	⊞	50.5B
Oceania	⊞	⊞	1.88B

un_region	un_subregion	country	sum(population)
⊟	⊟	⊟	342B
Africa	⊟	⊟	38.0B
Africa	Eastern Africa	⊞	10.9B
Africa	Middle Africa	⊞	4.43B
Africa	Northern Africa	⊞	8.92B
Africa	Southern Africa	⊞	2.60B
Africa	Western Africa	⊞	11.2B
Americas	⊞	⊞	47.4B
Asia	⊞	⊞	204B
Europe	⊞	⊞	50.5B

8. Expand the country column of the visual by clicking the Plus icon in the field next to one of the un_subregions. We chose to expand Middle Africa.

   When you expand Middle Africa, it generates a complete tabulation aggregated by country.

un_region	un_subregion	country	sum(population)
⊟	⊟	⊟	342B
Africa	⊟	⊟	38.0B
Africa	Eastern Africa	⊞	10.9B
Africa	Middle Africa	⊞	4.43B
Africa	Northern Africa	⊞	8.92B

un_region	un_subregion	country	sum(population)
⊟	⊟	⊟	342B
Africa	⊟	⊟	38.0B
Africa	Eastern Africa	⊞	10.9B
Africa	Middle Africa	⊟	4.43B
Africa	Middle Africa	Angola	723M
Africa	Middle Africa	Cameroon	767M
Africa	Middle Africa	Chad	405M
Africa	Middle Africa	Congo	2.43B
Africa	Middle Africa	Equatorial Guinea	28.6M
Africa	Middle Africa	Gabon	68.4M
Africa	Middle Africa	Sao Tome and Principe	8.56M

9. Change the title to Hierarchical Cross Tabulation of Population.
10. At the top left corner of the Visual Designer, click SAVE.

Related Information

Cross tabulation

Adding highlights to visuals

Highlights are a practical way to bring attention to values that exceed a particular threshold. Cloudera Data Visualization currently supports highlight marks at specific values on the combined bar/line chart type, and enables you to customize these highlights.

About this task

The following steps demonstrate how add highlights to an existing chart. To start, we are using the visual World Population - Dual Bar and Line, developed on dataset World Life Expectancy [data source samples.world_life_expectancy]. For more information, see *Combined bar/line visuals*.

Procedure

1. Open the visual World Population - Dual Bar and Line in Edit mode.
2. Clone the visual.
3. Click the Pencil icon next to the title of the visualization to edit it, and enter the new name.
4. Click the Pencil icon below the title of the visualization to add a brief description of the visual.
5. Change the title to World Population - Highlights on Dual Bar and Line.
6. At the top left corner of the Visual Designer, click Save.
7. Add the un_region field to the Filters shelf, and select Europe from the list of possible values.

   This graph has more variation than the unified World data.

8. Click Refresh Visual.

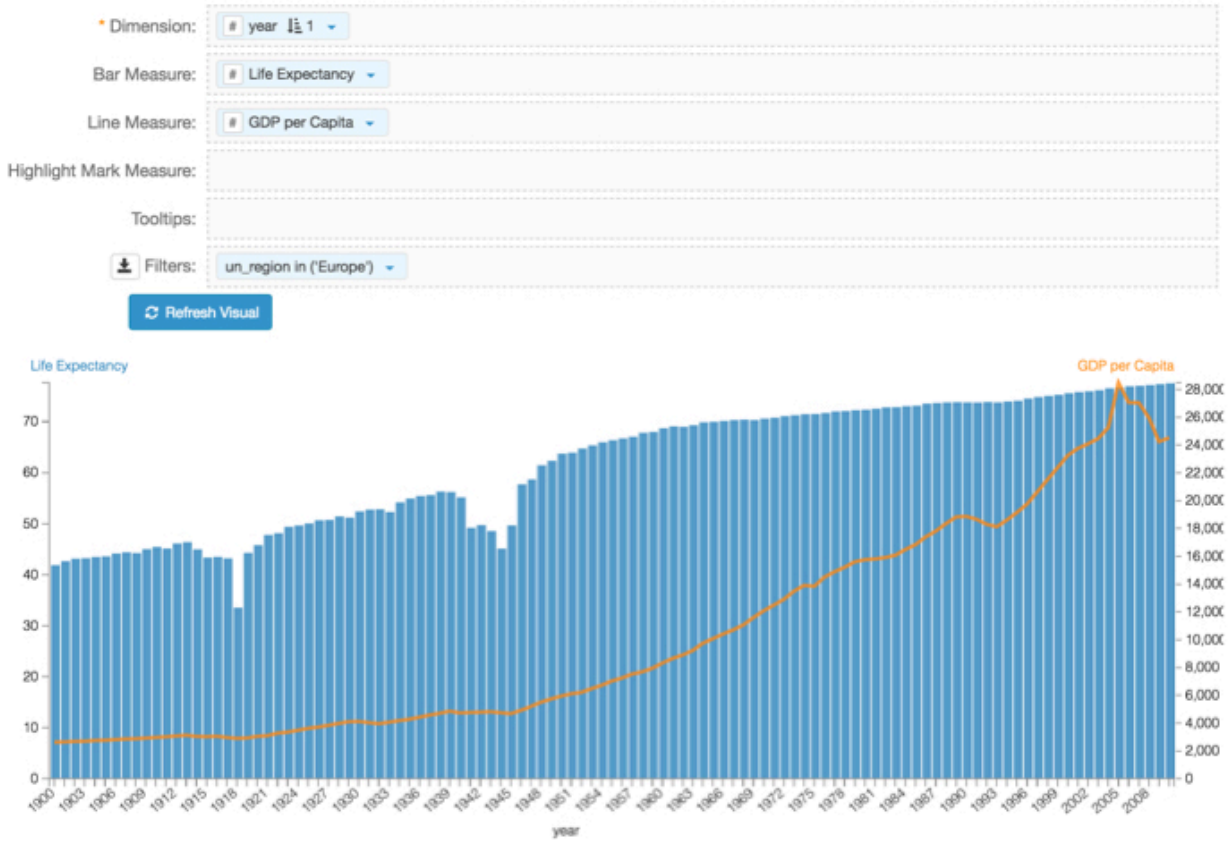

You can see the pronounced peak in the GDP per Capita line, centered around the year 2005.

9. Based on the graph, highlight the values where GDP per Capita is over $25,000.
10. Place the field gdp_per_capita on the shelf Highlight Mark Measure.
11. On the same shelf, click the Down arrow icon on the gdp_per_capita field, and choose Enter/Edit Expression.

12. In the Enter/Edit Expression window modal, delete the text in the default expression, change the function groupings to Conditional, and select the if statement.

Note:

Depending on the SQL dialect of the connection you are using, the syntax of the if statement may be different.

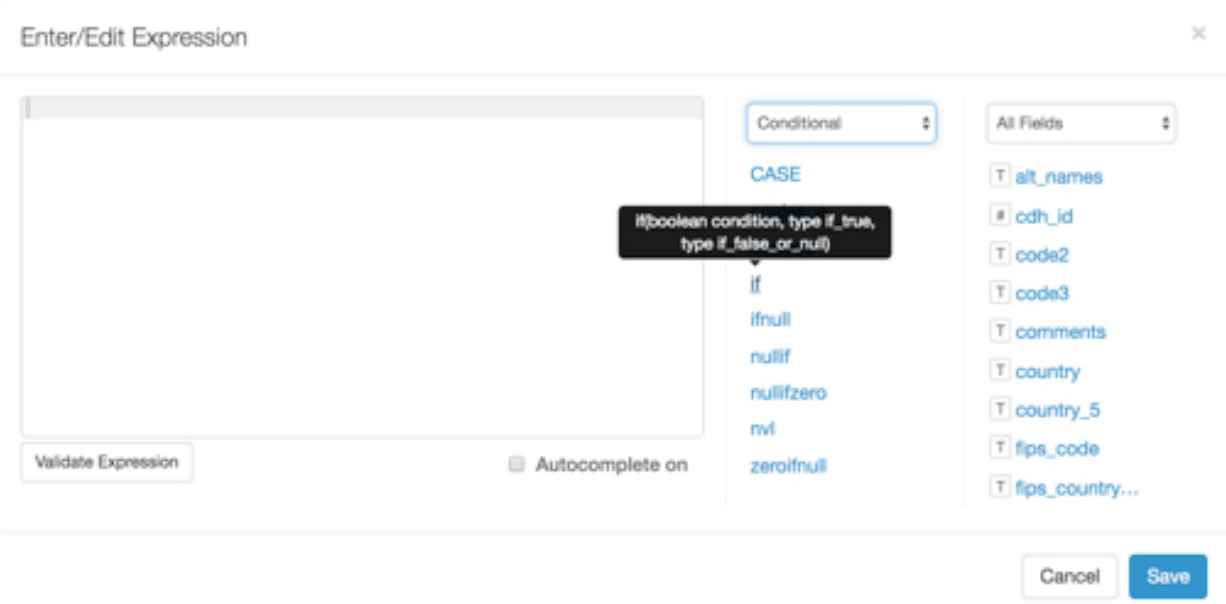

13. Complete the expression template.

It should look similar to the following expression:

```
if(avg([gdp_per_capita])>=25000, avg([gdp_per_capita]), null)
```

Optionally, you can alias the highlight; see the bold font in the changed expression.

```
if(avg([gdp_per_capita])>=25000, avg([gdp_per_capita]), null)
            as 'GDP per Capita > 25K'
```

14. Click Save.

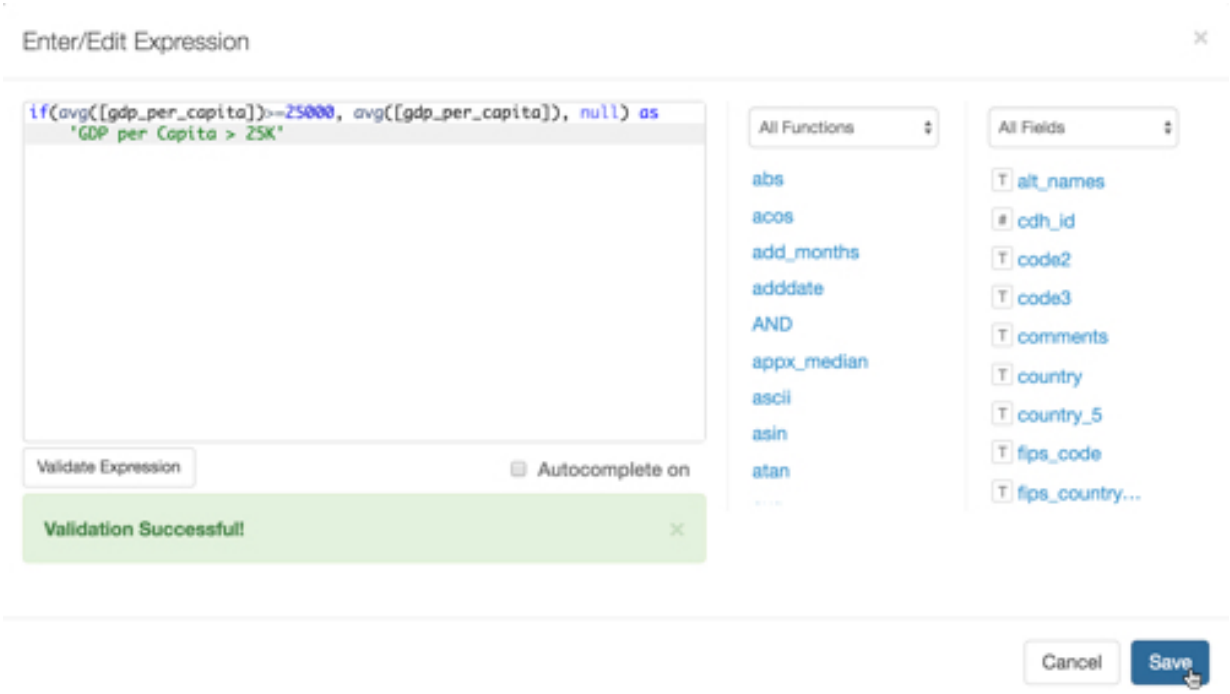

15. Save the visual, and then click View to change to View mode.

Note: The graph has green circle highlights when GDP per Capita values are equal or greater than 25,000.

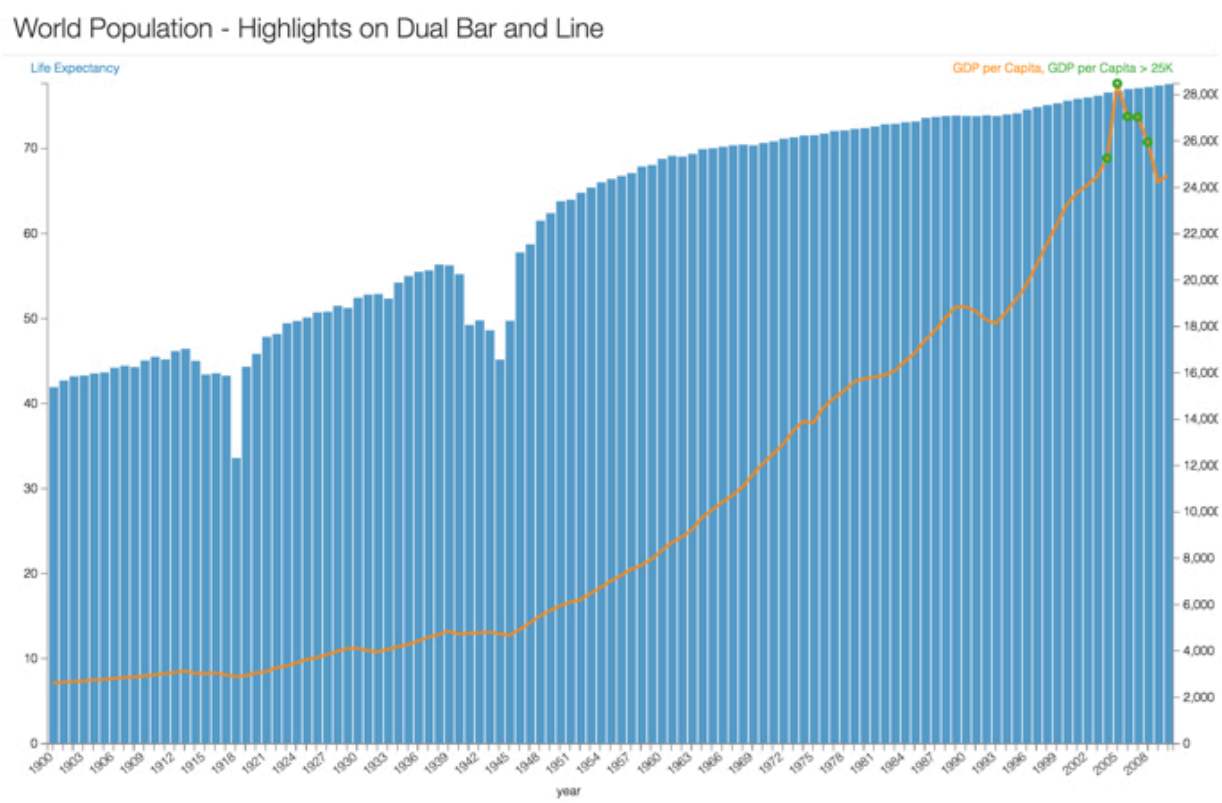

16. The default highlight mark is a circle. To convert it to another type, use the visual Settings interface as described in *Change highlight type*. You can also change the relative size of the mark as described in Change highlight type.

Related Information

Combo visuals

Cloudera Data Visualization 7.2.9

Applications

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What is an application? **4**

App Designer **4**

What is an application?

An application (also called an app) is a collection of dashboards tied together that can be launched as a standalone, branded data visualization tool.

The App Designer interface enables you to build applications, customize the tabs of the apps, and populate the tabs with relevant dashboards.

Related Information

App Designer

Creating a new application

App Designer

Cloudera Data Visualization App Designer is an intuitive and highly configurable interface for building applications. You can use it to build applications, customize the tabs of the apps, populate the tabs with relevant dashboards. You can also style your apps with custom logos, colors, and text.

The App Designer page has the following parts in the main area:

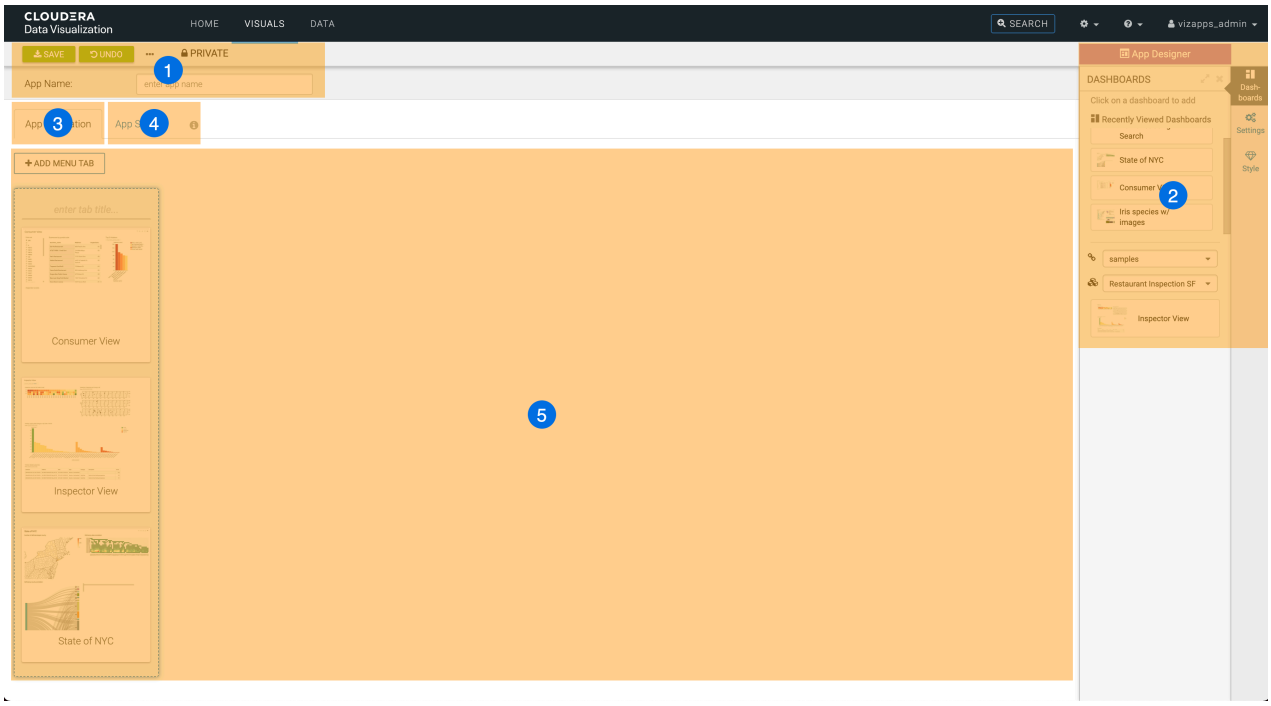

1. App Designer top menu bar
2. App Designer side menu bar
3. App Navigation tab
4. App Style tab
5. Preview area

App Designer top menu bar

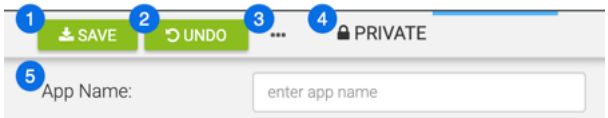

1. The SAVE button saves all changes to the application.
2. The UNDO button reverts all changes back to the last save operation.
3. The Supplemental menu provides access to additional functions:
   - The Clone App option lets you clone the current application.
   - The New Dashboard option opens the Dashboard Designer interface so you can create a new dashboard.
   - The New App option opens on a fresh/blank view of App Designer, so you can create a new application.
   - The Delete App option lets you delete the current application.
4. The workspace selector lets you assign the application to a Private, Public, or Custom workspace. For more information on workspaces, see Workspaces in Cloudera Data Visualization.
5. The App Name text box lets you specify the name of the application. You can change this name at any time.

Note: This name appears in the browser tab. The ID appears after the first time you save the application. Last updated information appears after the first save, and indicates how recently the app was updated. Modified by information appears after the first save, and names the user who made the most recent changes to the application.

App Designer sidebar

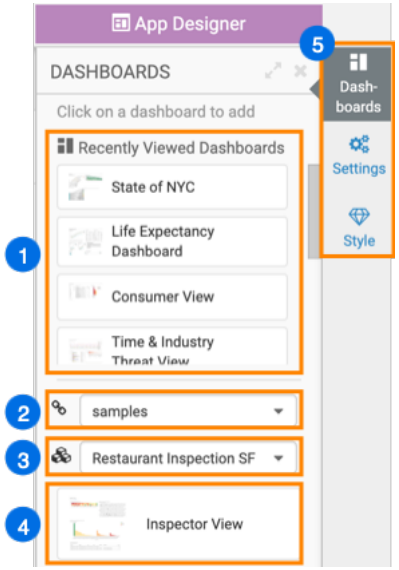

1. Recently Viewed Dashboards appear at the top of the Dashboards menu.
2. The Connection drop-down list lets you select an alternate connection.
3. The Dataset drop-down list lets you select an alternate dataset.
4. Dashboards available in the specified dataset appear under the Dataset drop-down list.
5. The menu selector enables you to choose between Dashboards, Settings, and Style menus. The Dashboards menu is usually used for configuring the navigation of the application, and the other two menus for styling the application.

App Navigation tab

The App Navigation view exposes all elements necessary to configuring the structure of the app: the tabs and their order, and the dashboards that appear in each tab. You can use the elements of the Dashboards menu to find the relevant dashboards for your app.

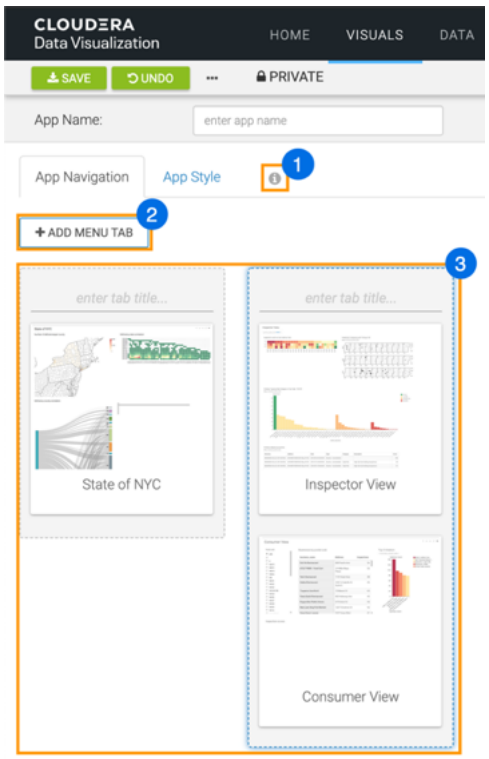

1. The information icon provides in-browser instructions on how to use the App Designer. If you hover over the icon, the following tooltip appears:

   How to create a new App

   - **Name the app.**
   - **Add a new menu section to your app:** click **+ Add Menu Tab**, and enter the title of the menu section. The menu name appears as a tab on the menu bar.
   - **Add dashboards to each tab:** select the tab section, and add existing dashboards by selecting them in the **Dashboards** menu. These dashboards appear as dropdown selections on the menu bar.
   - **Organize the app:** drag-and-drop menu tabs to order them in the app menu, and drag-and-drop dashboards in menu sections to specify dashboard order.
   - **Style the app:** switch to the **App Style** tab. In the **Style** menu, select the desired font options, colors, images, and other options.
   - **Save the app.**
2. The ADD MENU TAB button enables you to add additional tabs to your application.
3. The preview area shows the tabs of your application as tiles.

   You can specify the name of each tab, add dashboards to the tabs, change the order of dashboards in one tab, or move the dashboards between the tabs. You can also move the tiles around to change the order of the tabs, or remove a tab from the application by clicking the delete icon at the top of the tile.

   You can click a dashboard to open it in a new browser tab, in View mode if you want to make changes to it.

App Style tab

The App Style view focuses on how to style an application.

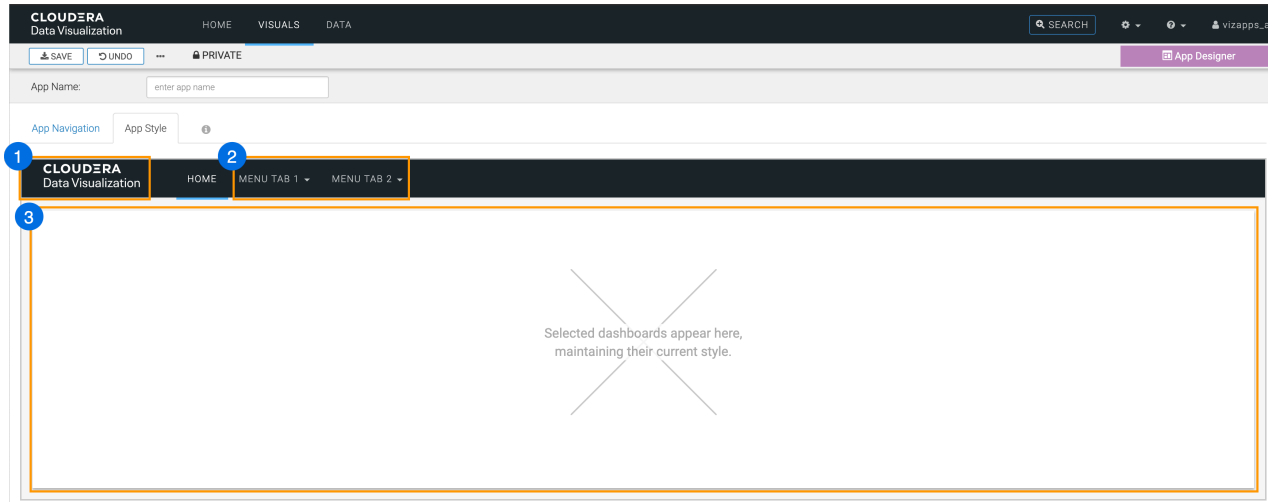

1. The logo of the application. You can set a custom logo from the Style menu in the sidebar.
2. The application tabs added in the App Navigation tab appear here. If more than one dashboard is added to a tab a dropdown menu will let you choose between them.
3. The preview area of the view. The selected dashboards are added here with their current style.

Related Information

Creating a new application

Workspaces in CDP Data Visualization

Cloudera Data Visualization 7.2.9

Click Behavior Between Visuals

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating control visual.....4

Creating target visual.....4

Creating application and enabling Click Send.....5

Enabling Click Receive on target visual..... 7

Exploring Click behavior.....9

Creating control visual

In Cloudera Data Visualization, you can have one (source) visual act as a filter that controls the behavior of the target visual, inside the same application.

Procedure

1. Create a visual based on the US State Population Over Time dataset.

   The specifics for a sample visual follow:

   - Dataset: US State Population Over Time
   - Visual type: map
   - Geo shelf: state field
   - Color shelf: population field, as sum(population)
   - Filters shelf: year field, as year in (2010)
2. Click REFRESH VISUAL.

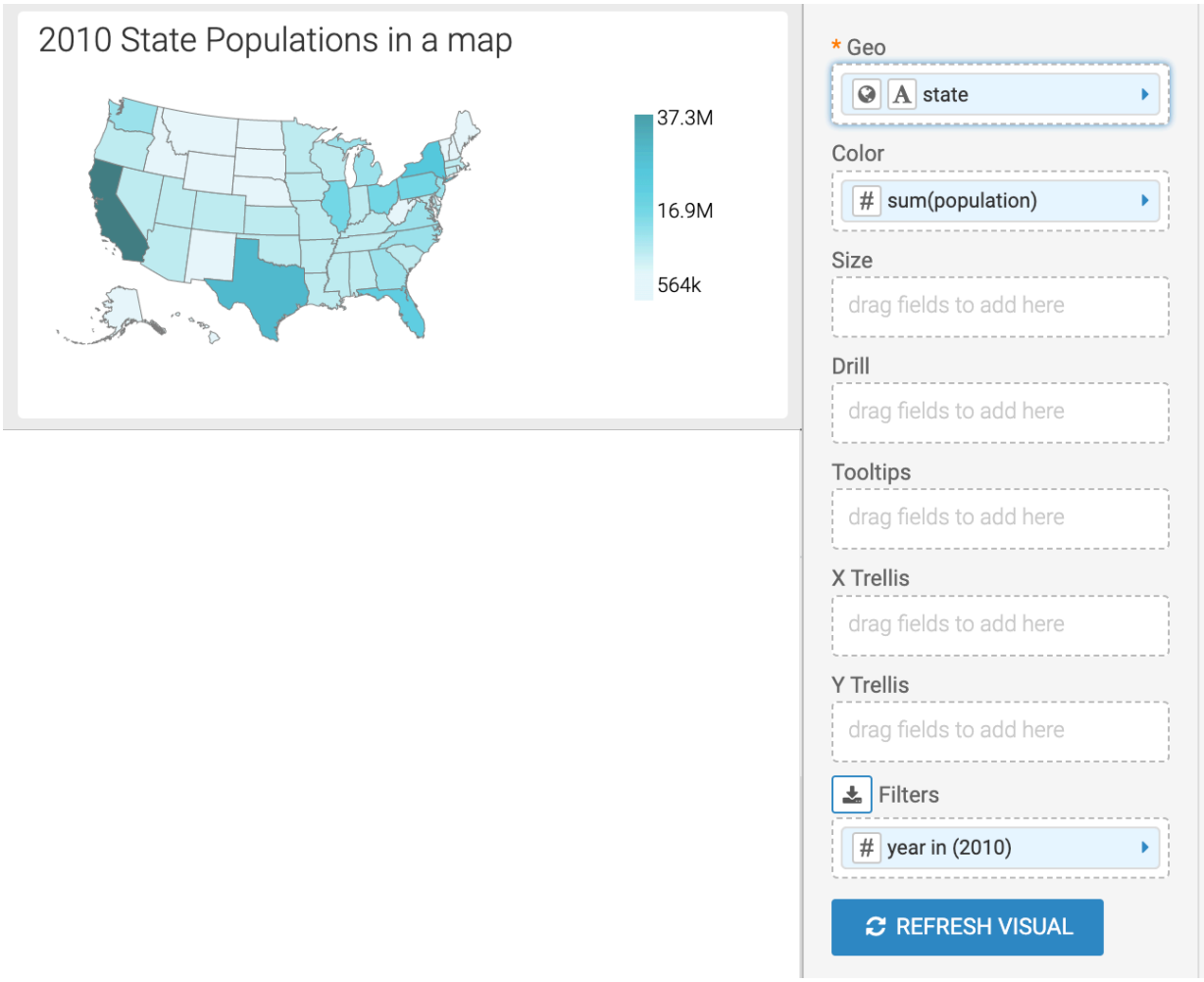

3. Name the visual as Control Visual and save it.

Creating target visual

Procedure

1. Create a visual based on the US County Population dataset.

   The specifics for the sample visual follow:

   - Dataset: US County Population
   - Visual type: map
   - Geo shelf: stname field, aliased as state, and ctyname field, aliased as country
   - Color shelf: add tot_pop field, then change the expression to sum([tot_pop]) as 'population'
   - Tooltips shelf: add tot_male field and change the expression to sum([tot_male]) as 'males', and add tot_female field and change the expression to sum([tot_female]) as 'females'
   - On the Color and Tooltips shelves, change the Display Format for each field by adding the comma thousands separator.
   - Filters shelf: for the year field, change the expression to year=5, and for the agegrp field, change the expression to agegrp=0
2. Click REFRESH VISUAL.

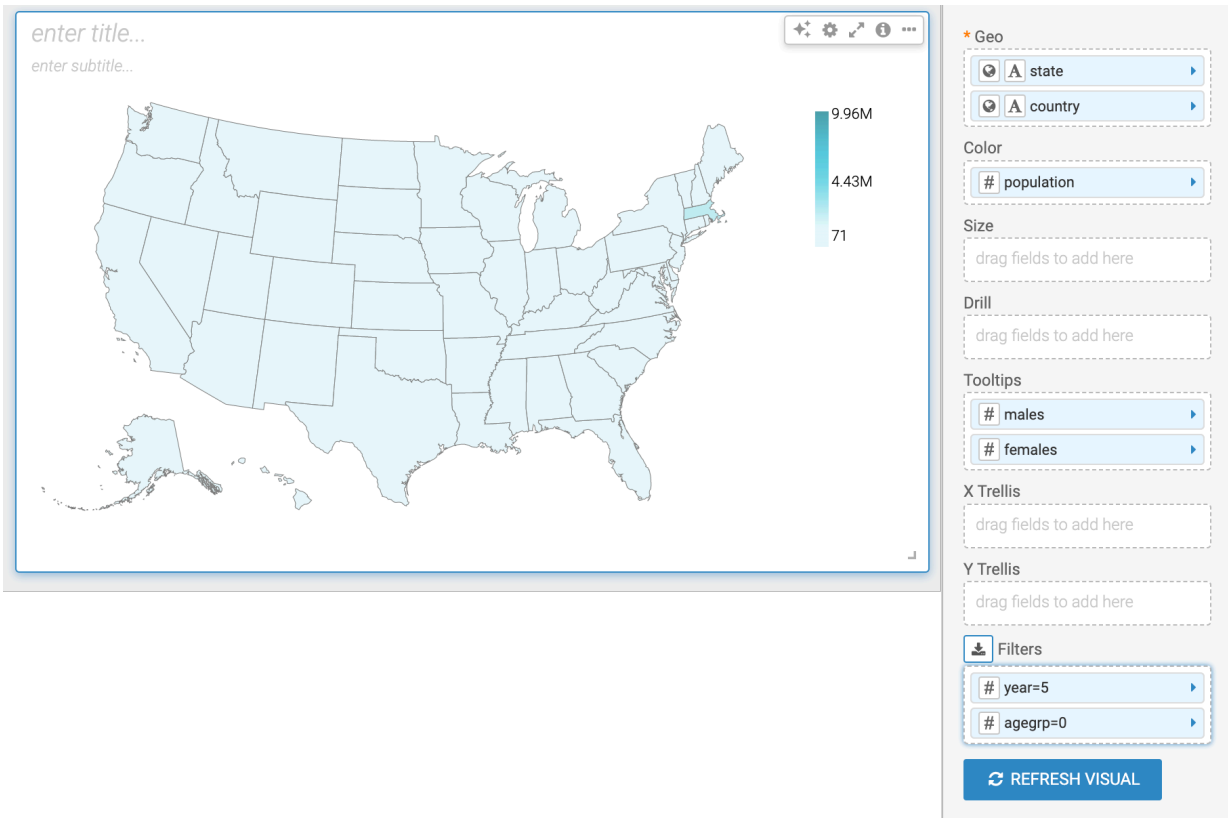

3. Name the visual as Target Visual and save it.

Creating application and enabling Click Send

About this task

Follow these steps to create an application for the two visuals, Control Visual and Target Visual.

Procedure

1. Click New App and name it Control-Target Application.

2. Select the Visuals tab.
3. Click to expand the US State Populations Over Time grouping, and select Control Visual.

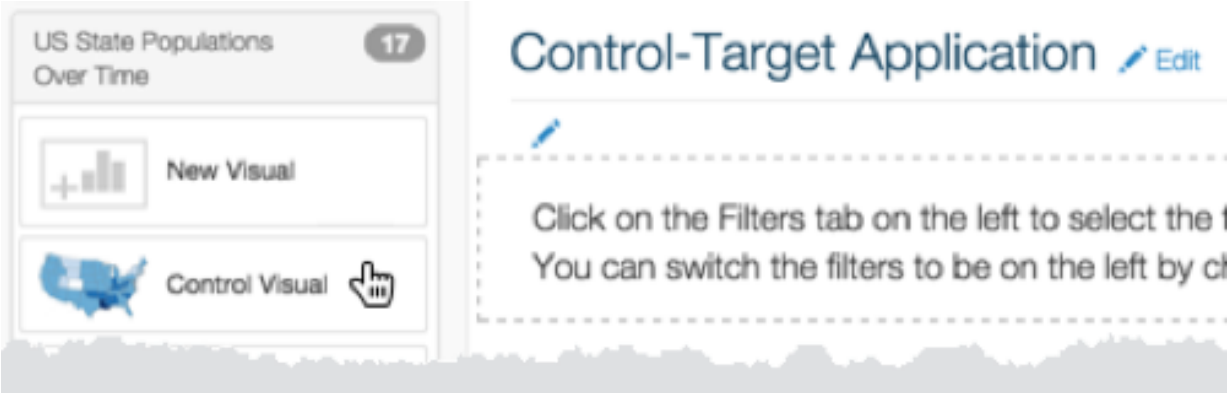

4. Click to expand the US County Population grouping, and select Target Visual.

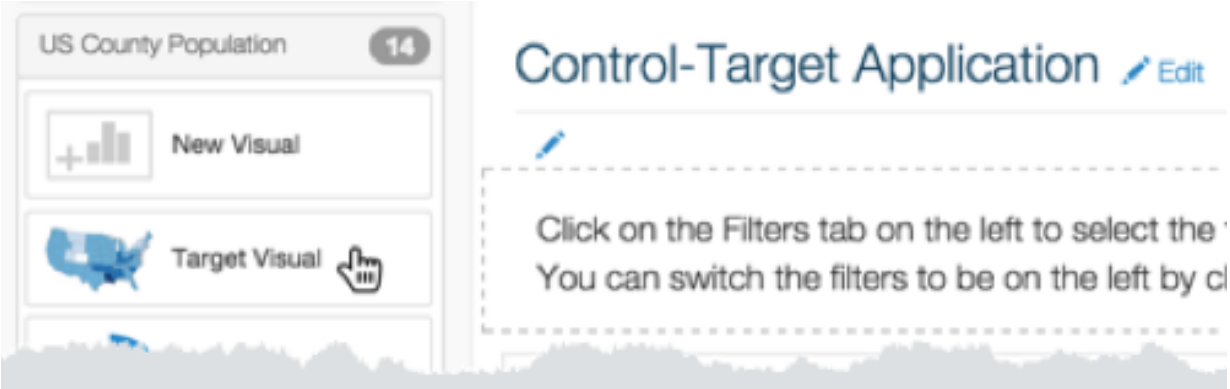

5. To enable parameter passing between the two visuals (in this case the parameter is state), left-click on one of the states in the Control Visual until the control menu appears.

6. Click Enable filter on click.

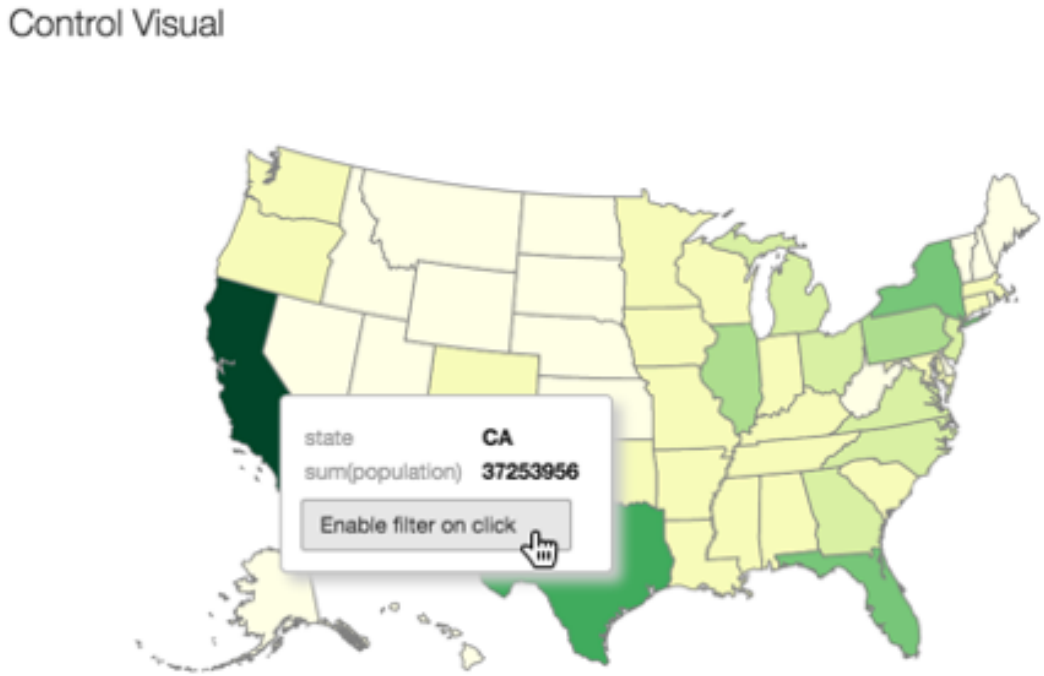

7. Save the app.

Enabling Click Receive on target visual

Procedure

1. Edit the Target Visual.

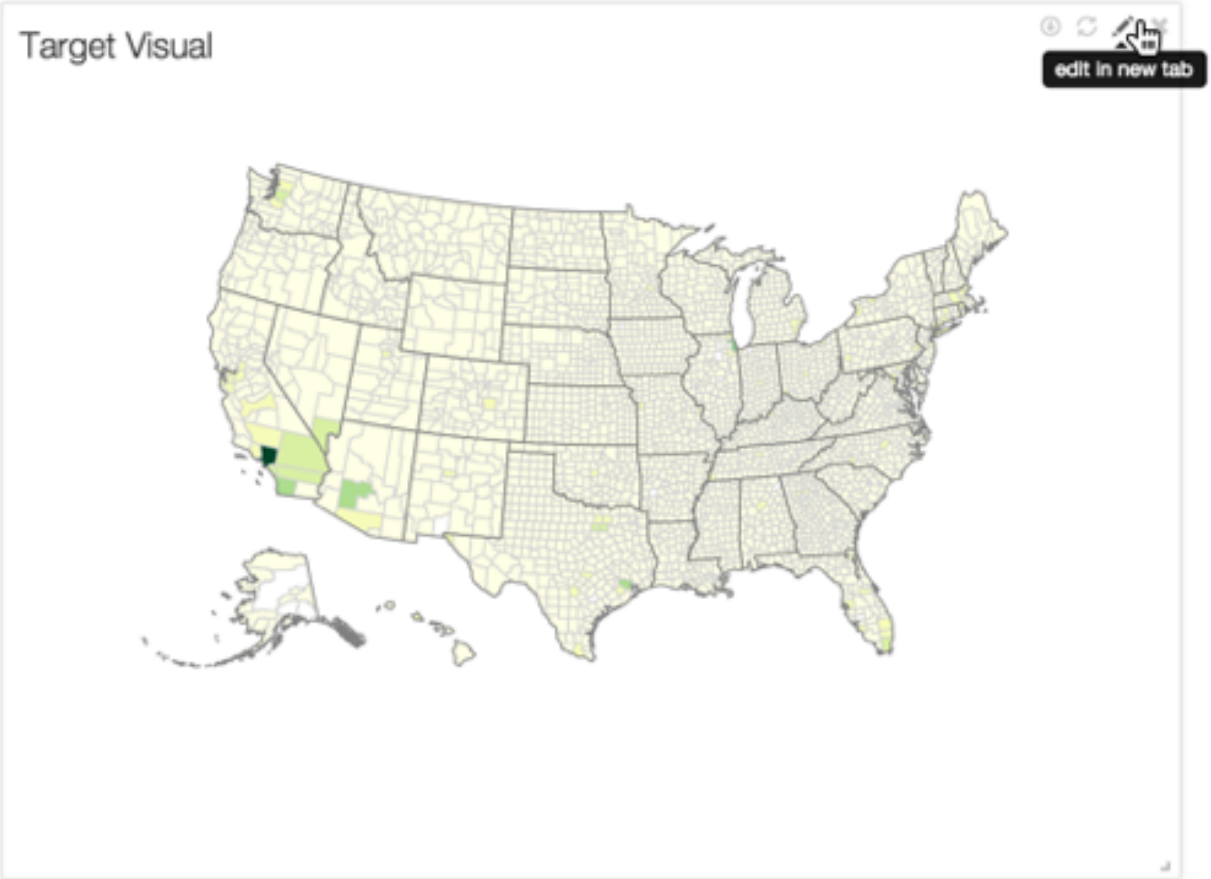

2. Add the field state_names state to the Filters shelf.
3. Edit the expression for the new filter to [state_names state] = ('<<state>>'), validate, and save.

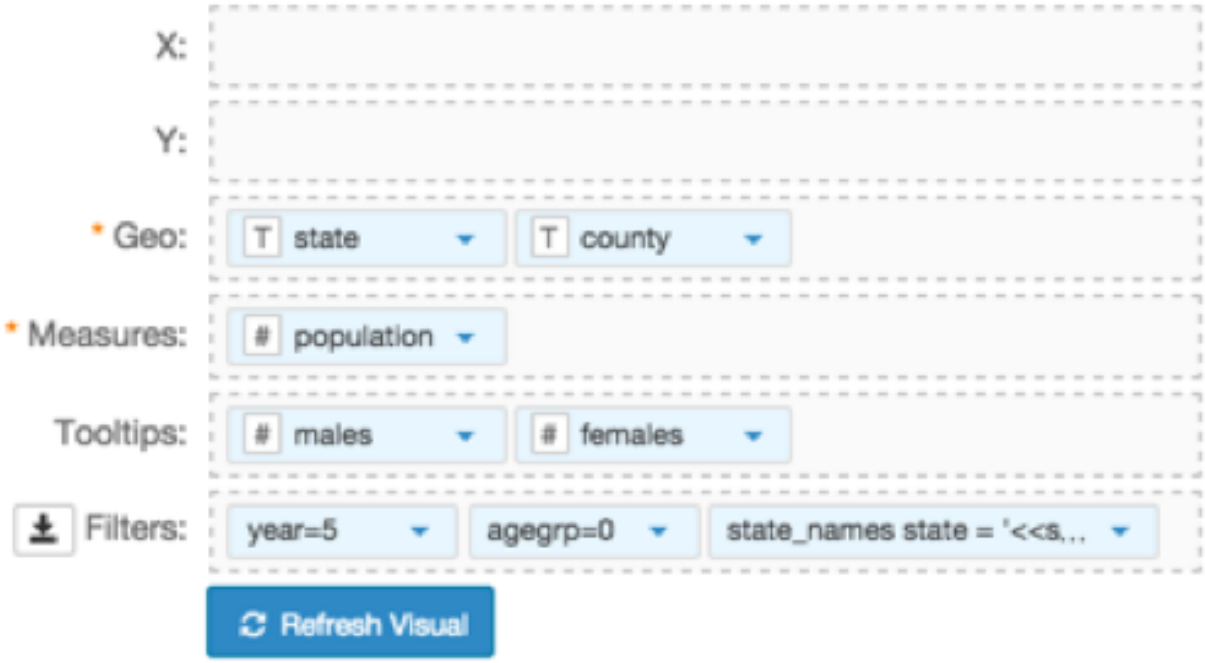

4. Save the visual.

Exploring Click behavior

Before you begin

Before starting, make sure to refresh the browser page to load the updated Target Visual. Note that it does not show a map.

Procedure

1. In the Control Visual, left-click one of the states (California) until the control menu appears, then click Test click behavior.

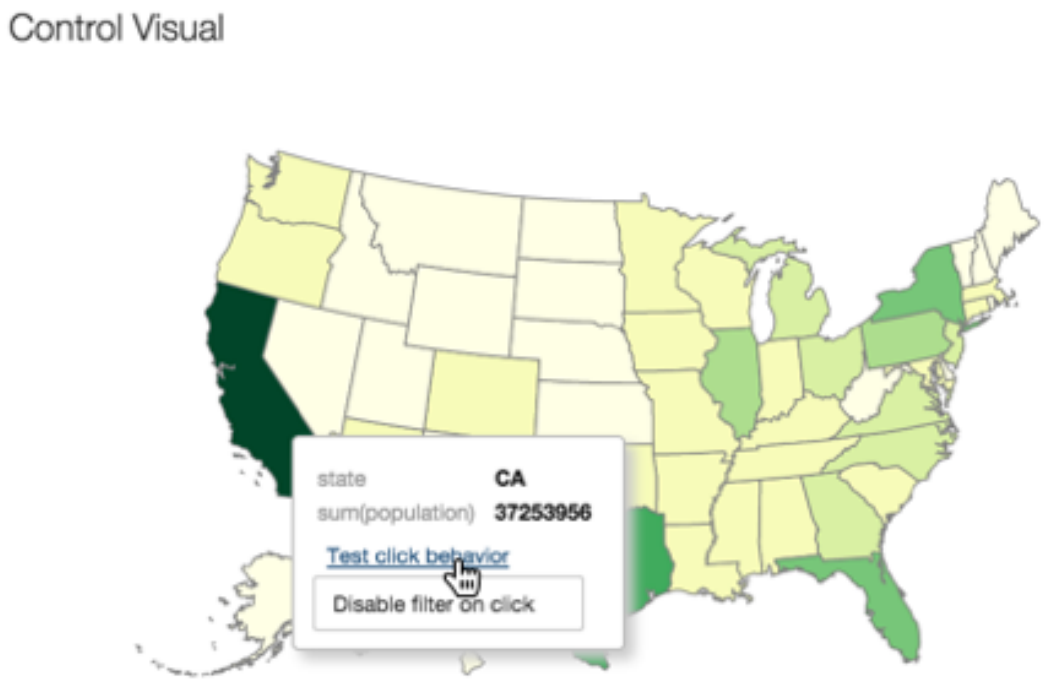

The Target Visual changes, zooming in on the California portion of the map, displaying population density in county choropleth mode.

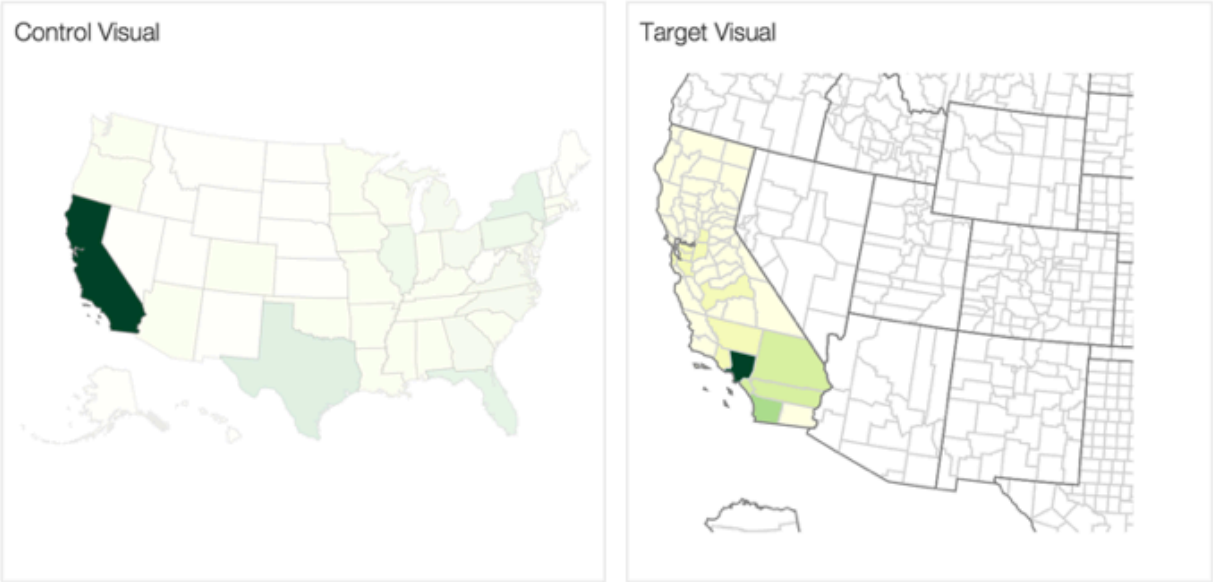

2. Save the app and open it in View mode.

Note: Every click on a state region in the Control Visual refreshes the Target Visual.

Cloudera Data Visualization 7.2.9

Configuring Cascading Filters

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating the dashboard **4**

Adding filters to the dashboard **6**

Configuring filters and enabling cascading filter interaction **10**

Exploring the dashboard filter interaction **12**

Creating the dashboard

About this task

The following steps use a calendar heat map visual, such as the one described in Calendar heat maps, which is built on the dataset SFPD Incidents.

Procedure

1. Start a new dashboard by clicking New Dashboard.
2. In the Dashboard Designer, open SFPD Incidents - Calendar Heat Map, built according to the instructions in Calendar heat maps. Ensure that you are using the SFPD Incidents dataset.

   The visual appears in the dashboard.
3. To edit the visual, click the Pencil icon in the top right corner of the visual.

4. Under Dimensions, select pddistrict and add it to the Filters shelf. Repeat the action with category and date.

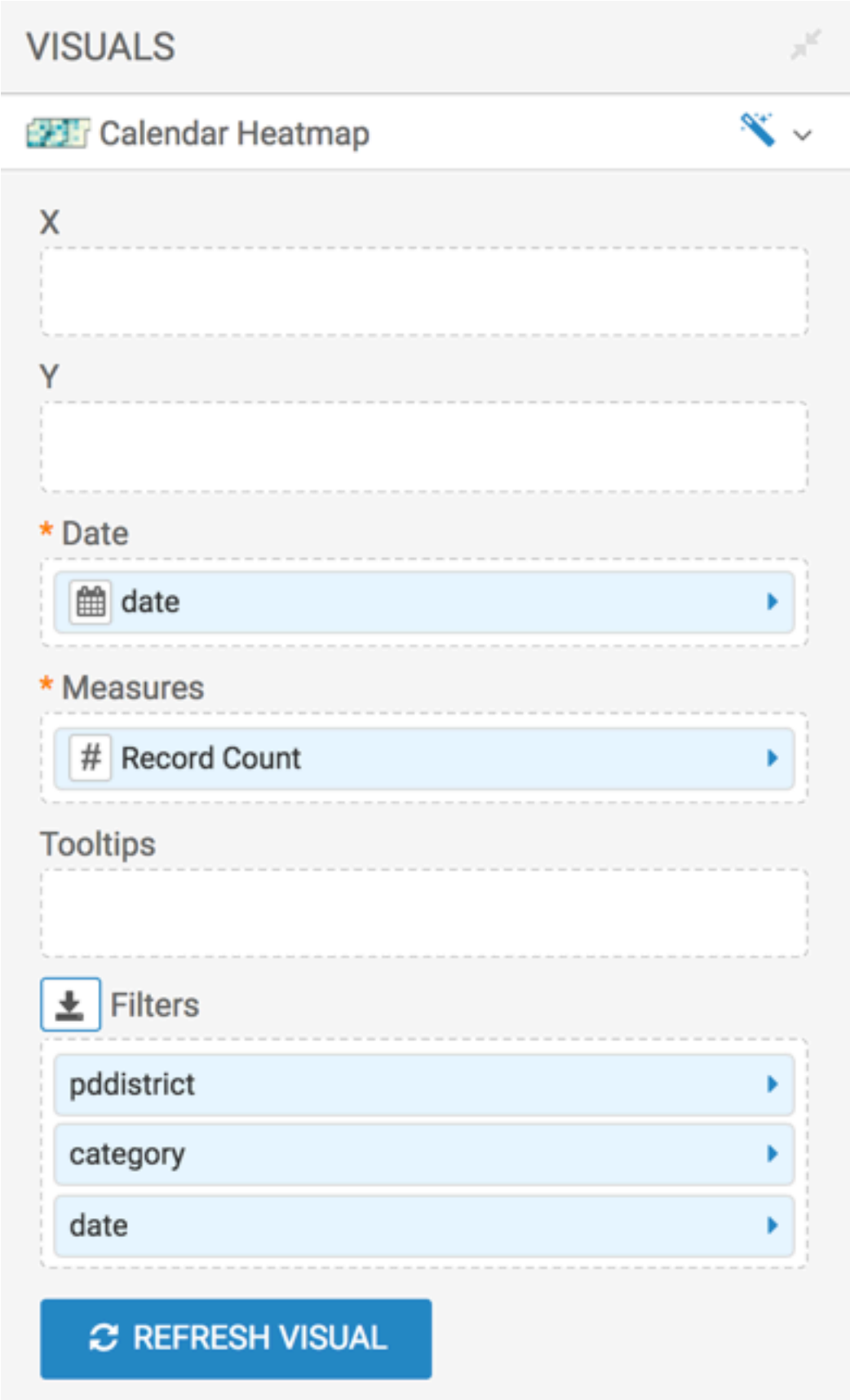

5. Name and save the dashboard.

Related Information

Calendar heat maps

Adding filters to the dashboard

About this task

Add the three filters to the dashboard.

Procedure

1. In the Dashboard Designer, click the Filters tab on the left.

2. Under Dimensions, click the date field.

   This adds a filter to the dashboard. Alternatively, you can also drag the field you want to use as a filter and drop it into the filter bar.

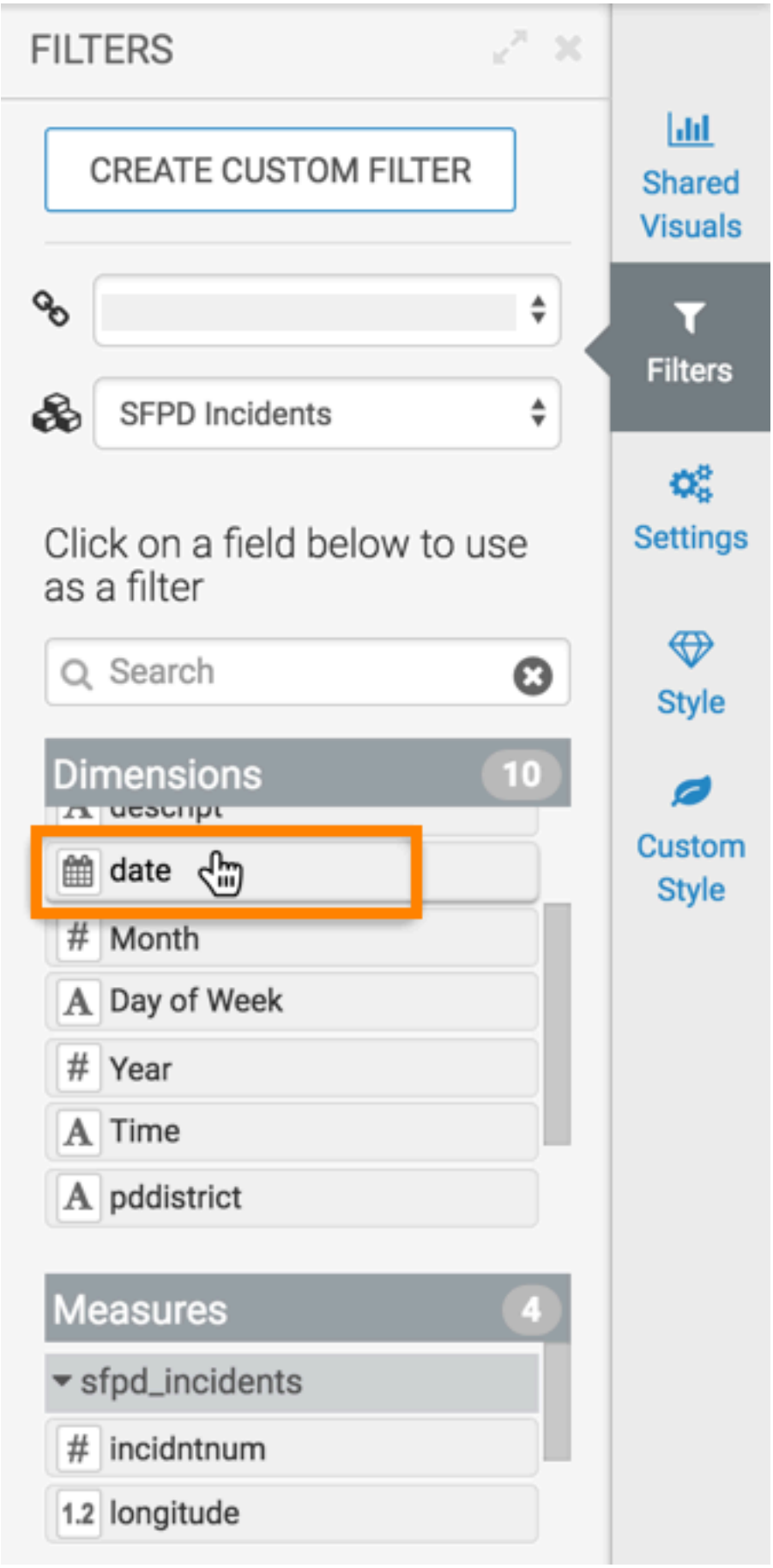
FILTERS
CREATE CUSTOM FILTER
SFPD Incidents
Click on a field below to use as a filter
Search
Dimensions 10
date
Month
Day of Week
Year
Time
pddistrict
Measures 4
sfpd_incidents
incidntnum
longitude
Shared Visuals
Filters
Settings
Style
Custom Style

This filter is based directly on the field date, and has a type timestamp.

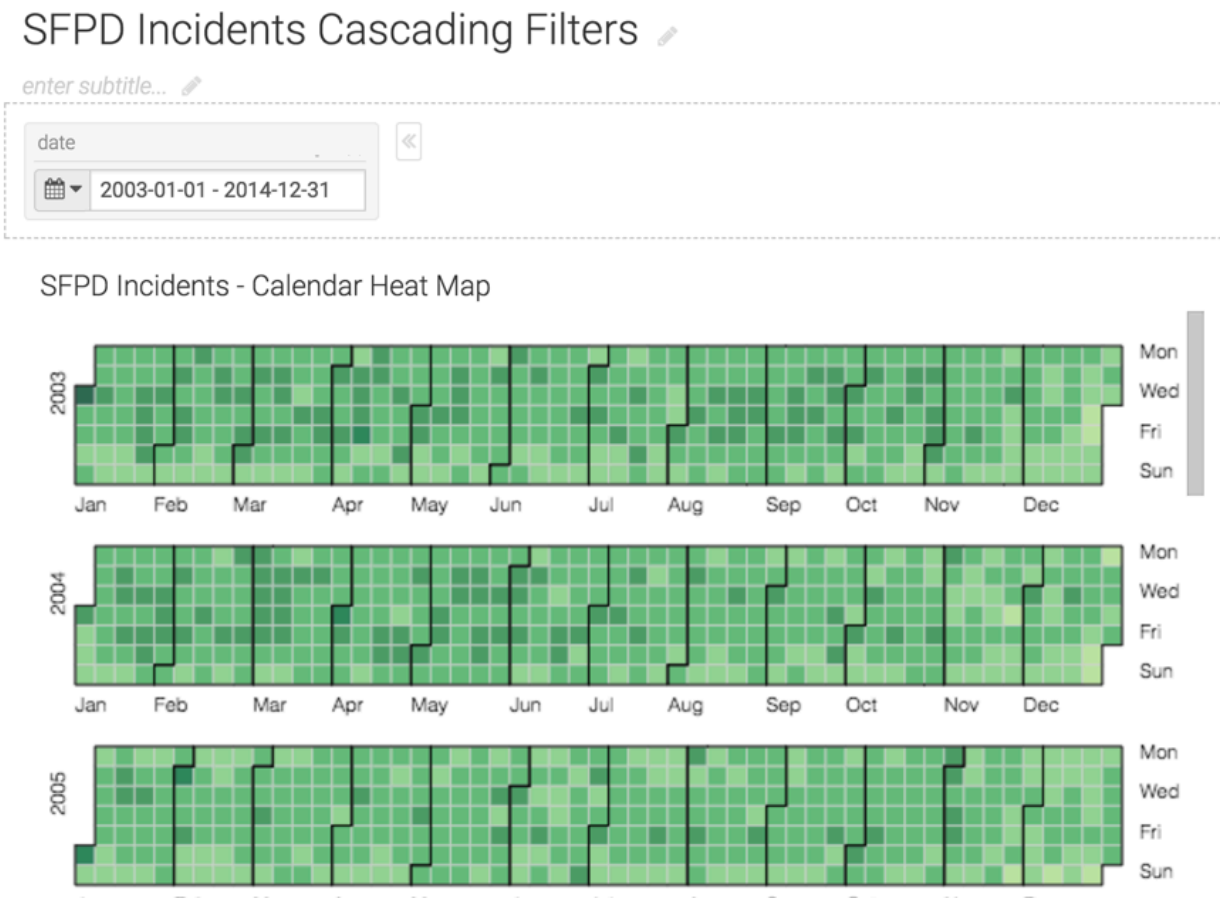

3. Add the other two filters. Under Dimensions, click the category field to add a filter based on that field. Similarly, click the pddistrict field to create the next filter.

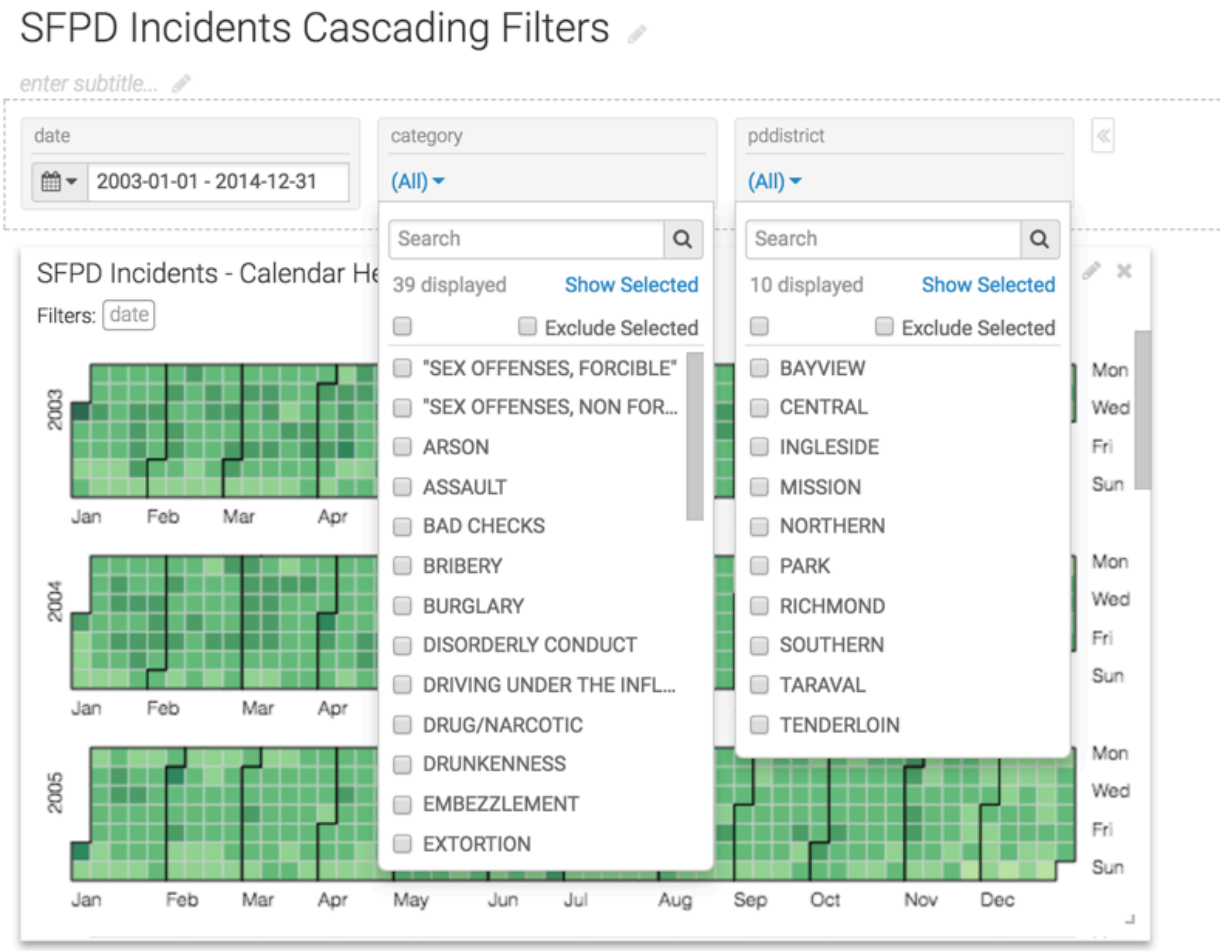

4. Save the dashboard.

Configuring filters and enabling cascading filter interaction

Procedure

1. On the category application filter, click the Gear icon.

   The Settings modal window appears.

2. Click the Display Settings tab and select the Allow only one item to be selected at a time option.

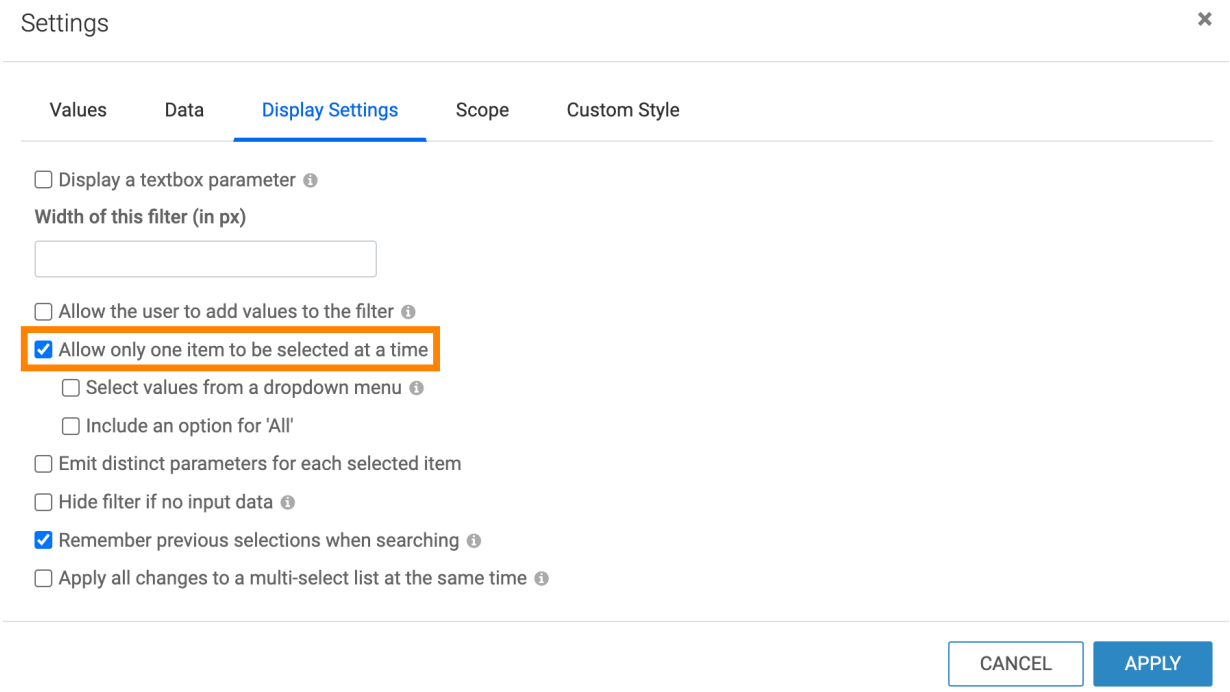

3. Switch to the Values tab and enter the value pddistrict,date for Filter Fields.

   Note that this matches the fields that are the basis of the other filters.

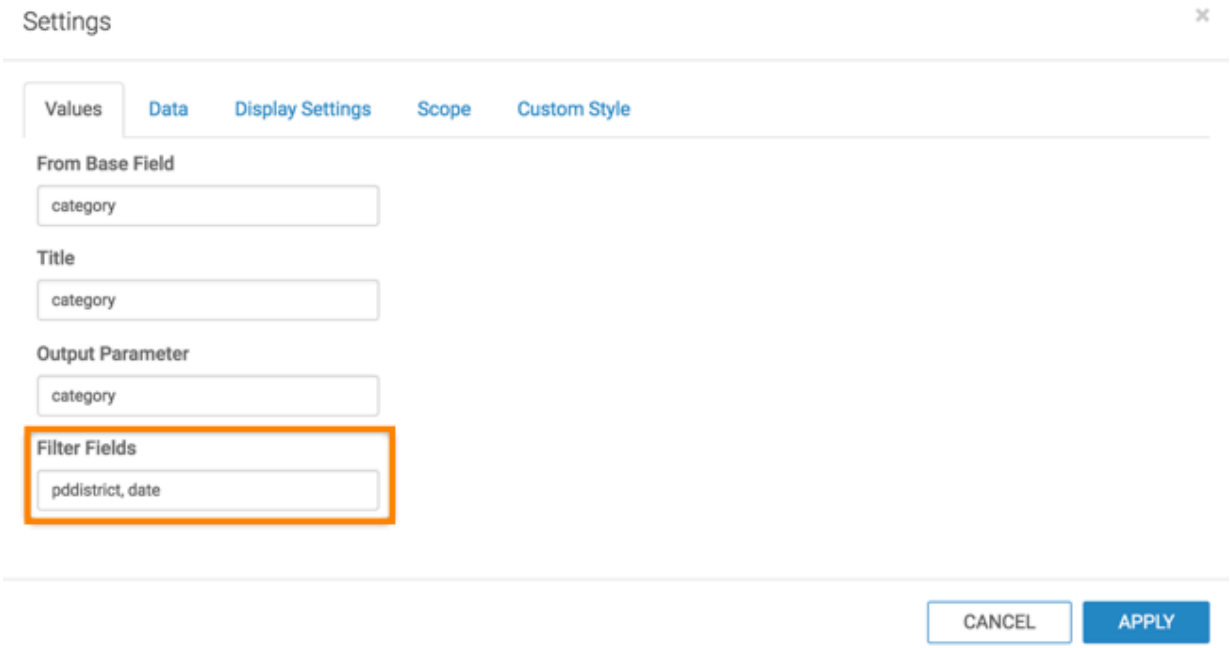

4. Click APPLY.
5. On the pddistrict application filter, click the Gear icon.

   The Settings modal window appears.

6. Click the Values tab and enter the value category,date for Filter Fields.

   This matches the fields that are the basis of the other filters.

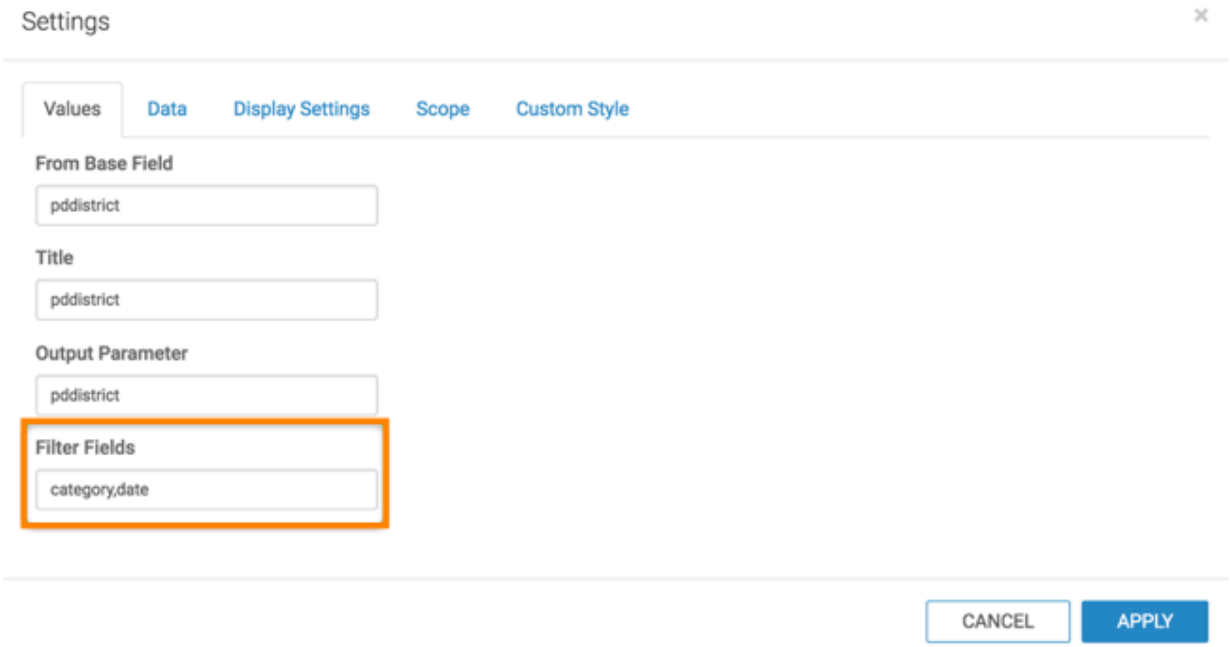

7. Click APPLY.
8. Save the dashboard.

Exploring the dashboard filter interaction

About this task

You can use the cascading filters to determine how the selections interact. Remember that there are 39 possible categories of police incidents, distributed across 10 policing districts.

Procedure

1. Open the dashboard in View mode.
2. Use the first application filter, datetime, to limit the date range to the 2010 calendar year. Start on 2010-01-01, end with 2010-12-31.
3. Click APPLY.
4. In the second filter, category, select the value BAD CHECKS.

   In the third filter, pddistrict, you can see that the value for PARK is missing. There are only 9 policing districts available in the filter.

   This is the result of filtering based on category BAD CHECK and date range 2010-01-01 through 2010-12-31. The resulting subset of data in the dataset does not include rows where pddistrict equals PARK.

   If you select another date range, such as years 2008 through 2014, all 10 districts show up on the pddistrict filter.

Cloudera Data Visualization 7.2.9

Configuring User Authentication Using LDAP

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enabling LDAP authentication..... 4

Using LDAPS..... 7

Complex matching logic for group queries using LDAPGroupQuery().....8

Enabling LDAP authentication

Cloudera Data Visualization by default uses local account (basic) authentication, where users must be created manually through the UI using the default admin user. This authentication method can be supplemented to also enable LDAP authentication so that corporate credentials can be used to login to Cloudera AI Data Visualization.

Before you begin

Prepare your installation by collecting the values of the following LDAP configuration parameters:

Configuration Item	Description
AUTH_LDAP_SERVER_URI	LDAP server URI, for example: ldap://ldap.example.com
AUTH_LDAP_BIND_DN	Username Distinguished Name (DN) of the bind user account. This needs to be the full DN for the Bind User, not just the bind username.
AUTH_LDAP_BIND_PASSWORD	Password of the bind user account
AUTH_LDAP_USER_SEARCH	The DN of the subtree that contains users, often an Organizational Unit (OU)
AUTH_LDAP_GROUP_SEARCH	The DN of the subtree that contains groups, often an OU
AUTH_LDAP_REQUIRE_GROUP	The DN of a group to which users must belong to have login privileges
LDAP Group for Admins	The DN of the Admins group. Users in this group have admin access.

LDAP can be configured for non-public applications, but it may lead to a double-login scenario, so consider using public applications that can be accessed by unauthenticated users. For more information on public applications in Cloudera AI, see Securing Applications.

For more information on how you can configure Cloudera Data Visualization security and authentication settings, see Security and Authentication

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu and select Site Settings.
2. Select Advanced Settings from the left navigation.

   You can configure LDAP authentication using one of the following options:

   Option

   Option 1: Configuring LDAP authentication with a bind user

   Bind User authentication with LDAP offers more flexiblity for users and group lookups, allowing you to map users in ML Data Viz to roles automatically. This means that when these users log in, they are granted access to specific dashboards and datasets based on their assigned roles. Note that additional steps in the ML Data Viz Roles setup section are required to achieve this functionality.

   Implementing Bind User authentication requires you to request and maintain a Bind User. While this setup may take a bit more time initially, it offers enhanced flexibility.

   With Bind User authentication, you may need to periodically update the Bind User password since it is likely to expire over time.

   This is a code snippet illustrating a simple search/bind approach that completes an anonymous bind, searches the OU for an object that matches the UID of the user's name, and attempts to bind using the DN obtained from the search and the user's password. The authentication succeeds only if the search returns exactly one result. If anonymous search is not possible, set

Option

the AUTH_LDAP_BIND_DN to the DN of an authorized user, and AUTH_LDAP_BIND_PASSWORD to the password for authentication.

```
import ldap
from django_auth_ldap.config import LDAPSearch

AUTH_LDAP_BIND_DN = ""
AUTH_LDAP_BIND_PASSWORD = ""
AUTH_LDAP_USER_SEARCH = LDAPSearch("ou=users,dc=example,dc=
com", ldap.SCOPE_SUBTREE, "(uid=%(user)s)")
```

Option 2: Direct Bind Approach

Direct Bind explicitly passes the user's credentials to authenticate with the LDAP server. This method does not require a Bind User account, but does not support group lookups for logged-in users. So you cannot configure automatic User-Role mapping using this method.

This is a code snippet illustrating a simple direct bind approach:

```
AUTH_LDAP_BIND_AS_AUTHENTICATING_USER = True
AUTH_LDAP_USER_DN_TEMPLATE = "uid=%(user)s,ou=users,dc=example
,dc=com"
```

Example

Example: Bind User with Environmental Variables

Store LDAP_DN and LDAP_PASSWORD environmental variables in the Project Settings under the Engine tab for easier management.

```
import ldap
from django_auth_ldap.config import LDAPSearch, NestedActiveDirectoryGroupT
ype, ActiveDirectoryGroupType

# Connection options
#AUTH_LDAP_START_TLS = True  # Optional for LDAPS but normally not needed
AUTH_LDAP_SERVER_URI = "ldap://ldap.example.com:389"

# Bind user setup
AUTH_LDAP_BIND_DN = os.getenv('LDAP_DN')
AUTH_LDAP_BIND_PASSWORD = os.getenv('LDAP_PASSWORD')

# Required Group for all users to access application
#AUTH_LDAP_REQUIRE_GROUP = "CN=All_Staff_WW,OU=Groups,DC=example,DC=local"

# Group for specifying super admins
#AUTH_LDAP_USER_FLAGS_BY_GROUP = {
#  "is_superuser": ["CN=cloud_spend_analysts,OU=Groups,DC=example,DC=local"]
#}

# User and group search objects and types
AUTH_LDAP_USER_SEARCH = LDAPSearch("CN=users,DC=example,DC=local",
 ldap.SCOPE_SUBTREE,"(sAMAccountName=%(user)s)")
AUTH_LDAP_GROUP_SEARCH = LDAPSearch("OU=Groups,DC=example,DC=local",
 ldap.SCOPE_SUBTREE,"(objectClass=group)")

# Map LDAP attributes to Django
```

```
AUTH_LDAP_USER_ATTR_MAP = {
"first_name": "givenName",
"last_name": "sn",
"email": "mail"
}

# Cache settings
# Note this may cause a delay when groups are changed in LDAP
AUTH_LDAP_CACHE_GROUPS = True
AUTH_LDAP_GROUP_CACHE_TIMEOUT = 3600*4  # Cache for 4 hours
REMOTE_GROUP_CACHE_TIMEOUT = 3600*4

# Group Settings
AUTH_LDAP_GROUP_TYPE = ActiveDirectoryGroupType()
AUTH_LDAP_FIND_GROUP_PERMS = True
AUTH_LDAP_MIRROR_GROUPS = False

# Some optional TLS/SSL options when enabling LDAPS

#AUTH_LDAP_GLOBAL_OPTIONS = {
#ldap.OPT_X_TLS_CACERTFILE: "/etc/bla.cert",        # Point to CA Cert file
#ldap.OPT_X_TLS_REQUIRE_CERT: ldap.OPT_X_TLS_NEVER, # Disable cert checking
#}

AUTH_LDAP_CONNECTION_OPTIONS = {
ldap.OPT_DEBUG_LEVEL: 1,  # 0 to 255
ldap.OPT_REFERRALS: 0,  # For Active Directory
}

# If there is no Bind User you can use these settings, but it's not the pr
eferred way
#AUTH_LDAP_BIND_AS_AUTHENTICATING_USER = True
#AUTH_LDAP_USER_DN_TEMPLATE = "example\%(user)s"

# The backend needed to make this work.
AUTHENTICATION_BACKENDS = (
'arcweb.arcwebbase.basebackends.VizBaseLDAPBackend',
'django.contrib.auth.backends.ModelBackend'
)
```

Example

Example: Direct Bind Configuration

```
import ldap
from django_auth_ldap.config import LDAPSearch, NestedActiveDirectoryGroupT
ype, ActiveDirectoryGroupType

# Connection options
#AUTH_LDAP_START_TLS = True  # Optional for LDAPS but normally not needed
AUTH_LDAP_SERVER_URI = "ldap://ldap.example.com:389"

# Bind user setup
#AUTH_LDAP_BIND_DN = os.getenv('LDAP_DN')
#AUTH_LDAP_BIND_PASSWORD = os.getenv('LDAP_PASSWORD')

# Required Group for all users to access application
#AUTH_LDAP_REQUIRE_GROUP = "CN=All_Staff_WW,OU=Groups,DC=example,DC=local"

# Group for specifying super admins
#AUTH_LDAP_USER_FLAGS_BY_GROUP = {
#  "is_superuser": ["CN=cloud_spend_analysts,OU=Groups,DC=example,DC=local"]
```

```
#}

# User and group search objects and types
#AUTH_LDAP_USER_SEARCH = LDAPSearch("CN=users,DC=example,DC=local",
ldap.SCOPE_SUBTREE,"(sAMAccountName=%(user)s)")

AUTH_LDAP_GROUP_SEARCH = LDAPSearch("OU=Groups,DC=example,DC=local", ldap
.SCOPE_SUBTREE,"(objectClass=group)")

# Map LDAP attributes to Django
AUTH_LDAP_USER_ATTR_MAP = {
"first_name": "givenName",
"last_name": "sn",
"email": "mail"
}

# Cache settings
# Note this may cause a delay when groups are changed in LDAP
AUTH_LDAP_CACHE_GROUPS = True
AUTH_LDAP_GROUP_CACHE_TIMEOUT = 3600*4  # Cache for 4 hours
REMOTE_GROUP_CACHE_TIMEOUT = 3600*4

# Group Settings
AUTH_LDAP_GROUP_TYPE = ActiveDirectoryGroupType()
AUTH_LDAP_FIND_GROUP_PERMS = True
AUTH_LDAP_MIRROR_GROUPS = False

# Some optional TLS/SSL options when enabling LDAPS

#AUTH_LDAP_GLOBAL_OPTIONS = {
#ldap.OPT_X_TLS_CACERTFILE: "/etc/bla.cert",        # Point to CA Cert file
#ldap.OPT_X_TLS_REQUIRE_CERT: ldap.OPT_X_TLS_NEVER, # Disable cert checking
#}

AUTH_LDAP_CONNECTION_OPTIONS = {
ldap.OPT_DEBUG_LEVEL: 1,  # 0 to 255
ldap.OPT_REFERRALS: 0,  # For Active Directory
}

# If there is no Bind User you can use these settings, but it's not the p
referred way

AUTH_LDAP_BIND_AS_AUTHENTICATING_USER = True
AUTH_LDAP_USER_DN_TEMPLATE = "example\%(user)s"

# The backend needed to make this work.
AUTHENTICATION_BACKENDS = (
'arcweb.arcwebbase.basebackends.VizBaseLDAPBackend',
'django.contrib.auth.backends.ModelBackend'
)
```

Using LDAPS

If you plan to configure authentication using LDAPS instead of LDAP there are 3 extra steps in the configuration that need to be considered.

Procedure

1. Update the LDAP Server URI and port to use LDAPS protocol.

```
AUTH_LDAP_SERVER_URI = "ldaps://ad-readonly.sjc
.cloudera.com:636"
```

2. Uncomment this section and add a valid path to a SSL certificate file.

```
AUTH_LDAP_GLOBAL_OPTIONS = {
ldap.OPT_X_TLS_CACERTFILE: "/path/to/bla.cert", #
Point to CA Cert file
ldap.OPT_X_TLS_REQUIRE_CERT: ldap.OPT_X_TLS_NEVER,
 # Disable cert checking
}
```

3. [Optional] Enable TLS if not already running

Note: Test setting up LDAPS with Steps 1 and 2 and restart Cloudera AI Data Visualization without this turned on first to avoid unnecessary debugging.

Complex matching logic for group queries using LDAPGroupQuery()

You can use LDAP to restrict user access to Cloudera Data Visualization resources by using multiple require groups.

About this task

Note: This feature depends on the LDAPGroupQuery() class, which is available starting with the django_auth_ldap release 1.2.12.

Procedure

To use a complex group query, implement the LDAP authorization requirement group, as we demonstrate in the following code snippet:

```
AUTH_LDAP_REQUIRE_GROUP = (
(LDAPGroupQuery("cn=enabled,ou=groups,dc=example,dc=com") |
LDAPGroupQuery("cn=also_enabled,ou=groups,dc=example,dc=com")) &
~LDAPGroupQuery("cn=disabled,ou=groups,dc=example,dc=com")
)
```

What to do next

For more information, see the django-auth-ldap reference documentation.

Cloudera Data Visualization 7.2.9

Configuring Natural Language Search

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enabling Natural Language Search in Site Settings.....4

Enabling a dataset for NLS.....5

Specifying a group for a dataset in NLS.....7

Specifying fields for a dataset in NLS.....8

Specifying synonyms for a dataset in NLS.....9

Specifying default aggregation field for a dataset in NLS.....10

Specifying suggested questions for a dataset in NLS.....13

Specifying word substitutions for a dataset in NLS.....14

Configuring date and time for search requirements.....15

Specifying date/time field format for a dataset..... 16

Specifying default time interval in dataset..... 17

Enabling Natural Language Search in Site Settings

Before using Natural Language Search (NLS) in Cloudera Data Visualization, you must enable the feature in the Site Settings.

About this task

Note: This setting is only available for users with administrative privileges.

Procedure

1. Click the Gear icon in the upper right corner to open the Site Administration menu.
2. Click Site Settings.

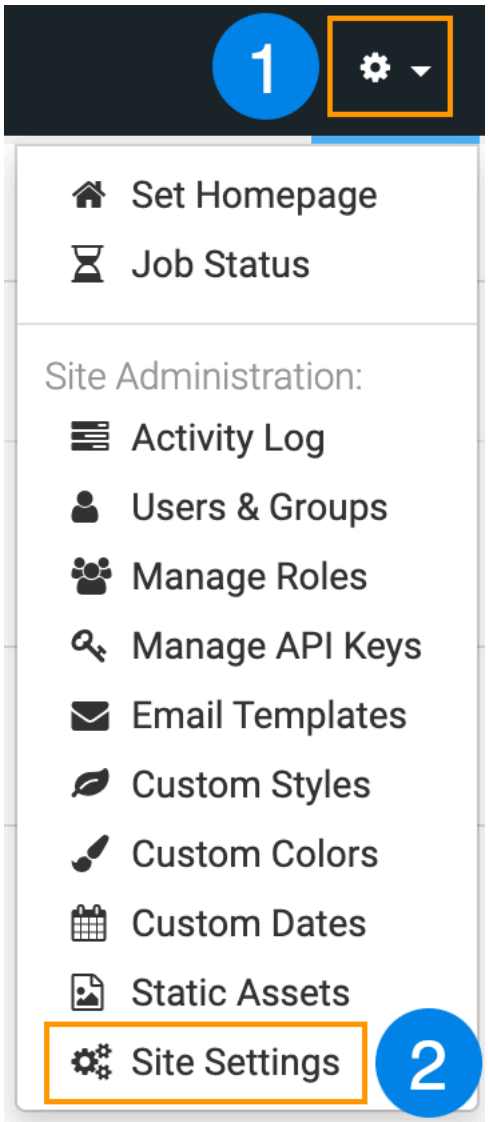

The settings page appears.

3. Select the following options under Features:
   - Enable Search - You have to enable this option for the other two to take effect.
   - Enable Search in Menu Bar - This option allows you to open the search modal on any screen from the icon from the Search button in the top right corner.
   - Enable Search Visual - This option allows you to add search visuals to a dashboard.

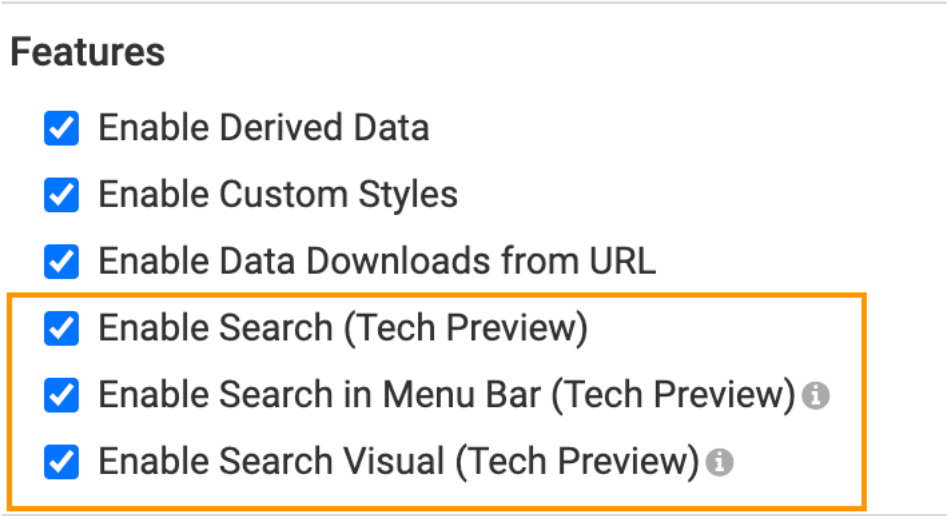

4. Click SAVE in the upper left corner.

Enabling a dataset for NLS

Before configuring Natural Language Search (NLS) in Cloudera Data Visualization, you must enable and configure the datasets.

Procedure

1. On the main navigation bar, click DATA.

2. In the list of datasets, click the one you want to enable for NLS.

   The Dataset Detail page appears.

   In the figure below, the Restaurant Inspection SF dataset is selected.

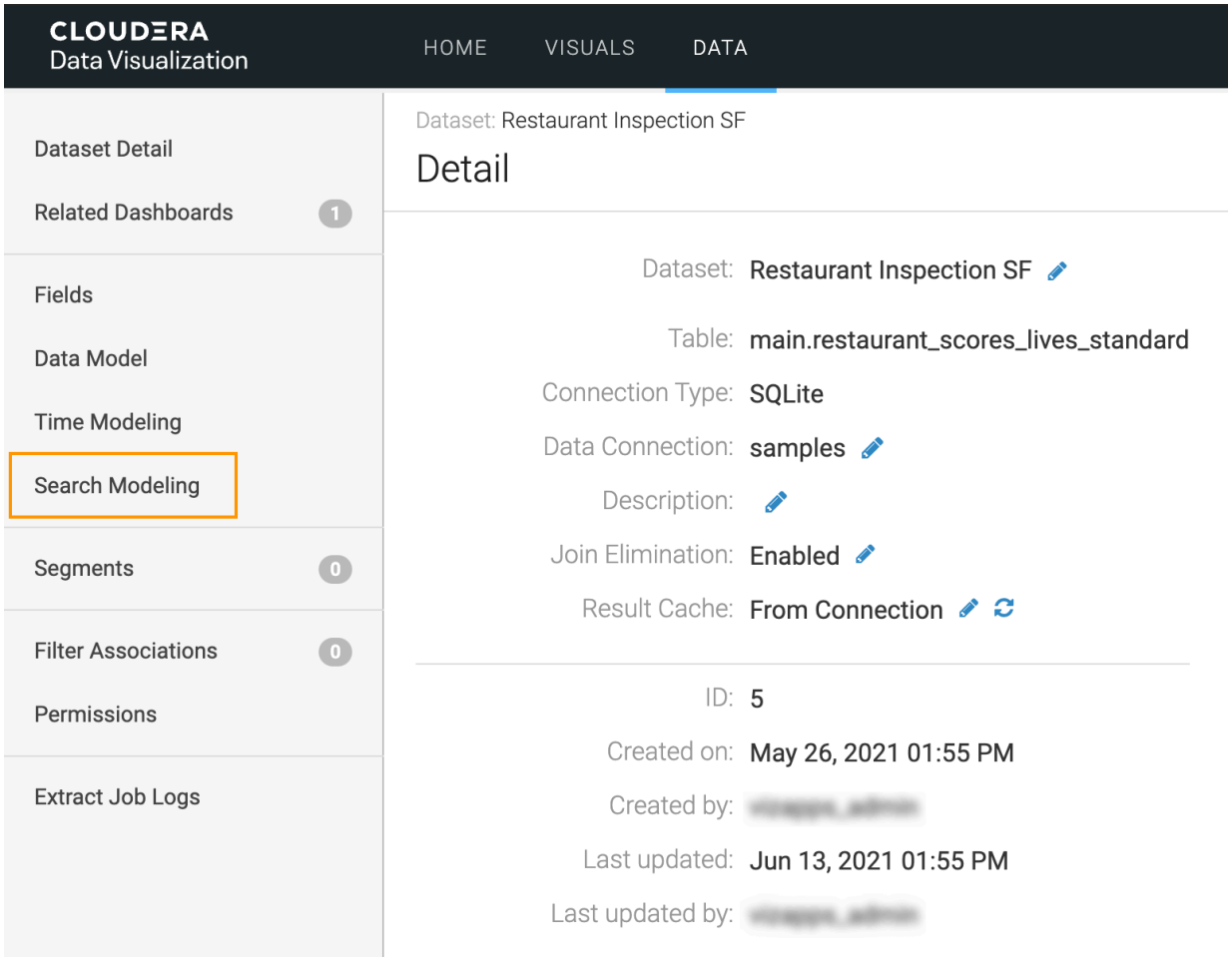

3. Click Search Modeling.
4. In the Search Modeling page, select Enable search on this dataset.

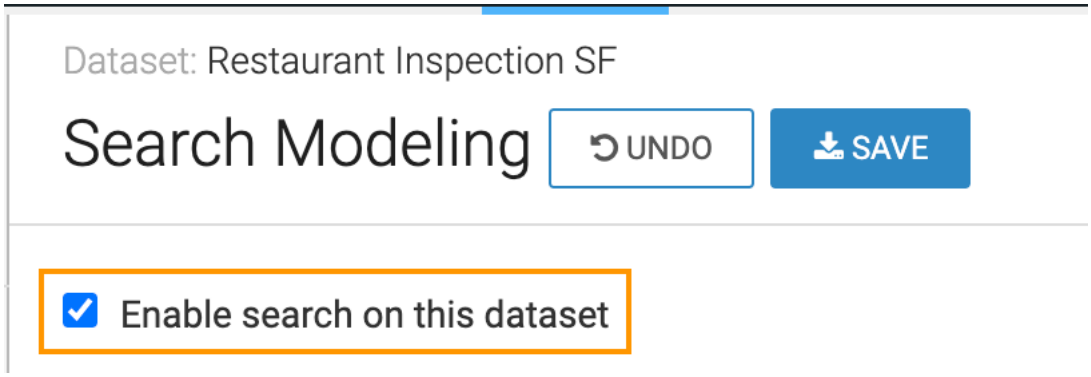

5. Click SAVE.

Note: On the list of datasets, a search-enabled dataset has an active Search icon.

Note: To disable the search on a dataset, clear the Enable search on this dataset checkbox.

Specifying a group for a dataset in NLS

Cloudera Data Visualization enables you to group datasets for Natural Language Search (NLS). You can use a group to more easily combine datasets during search operations.

Procedure

1. Under Search Group, add a new name for the group or choose one from the already existing ones.

   In the figure below, the dataset is added to the group 'Inspections'.

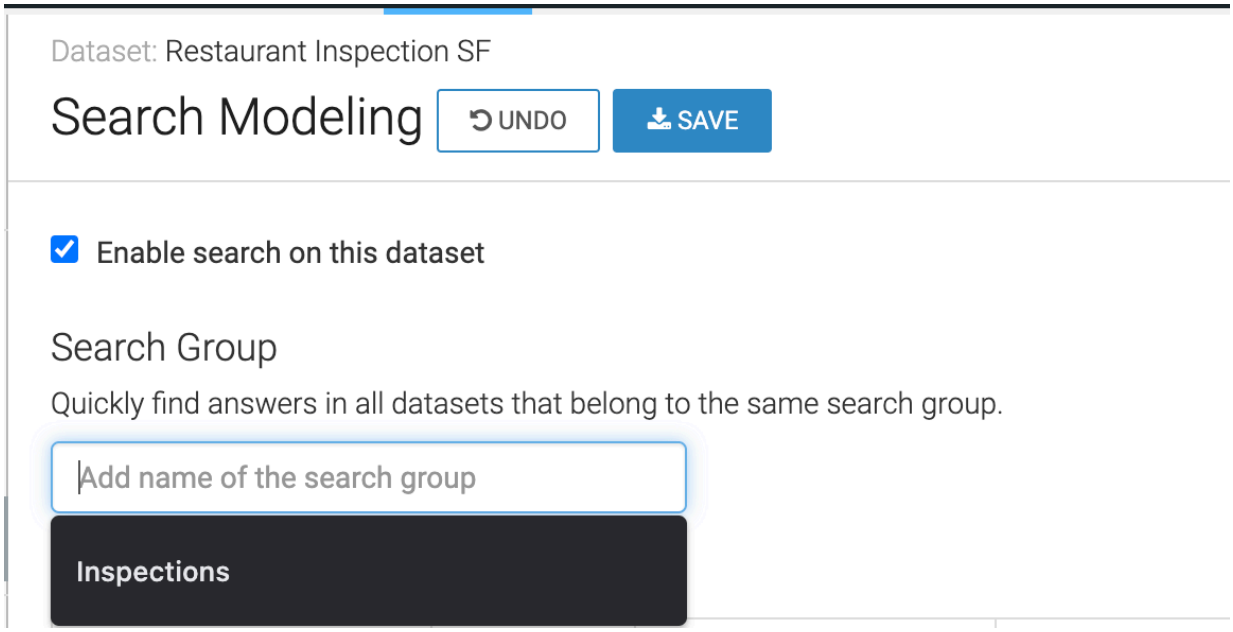

2. Click SAVE.

Results

You can see the datasets grouped in the drop-down list in the Search modal window.

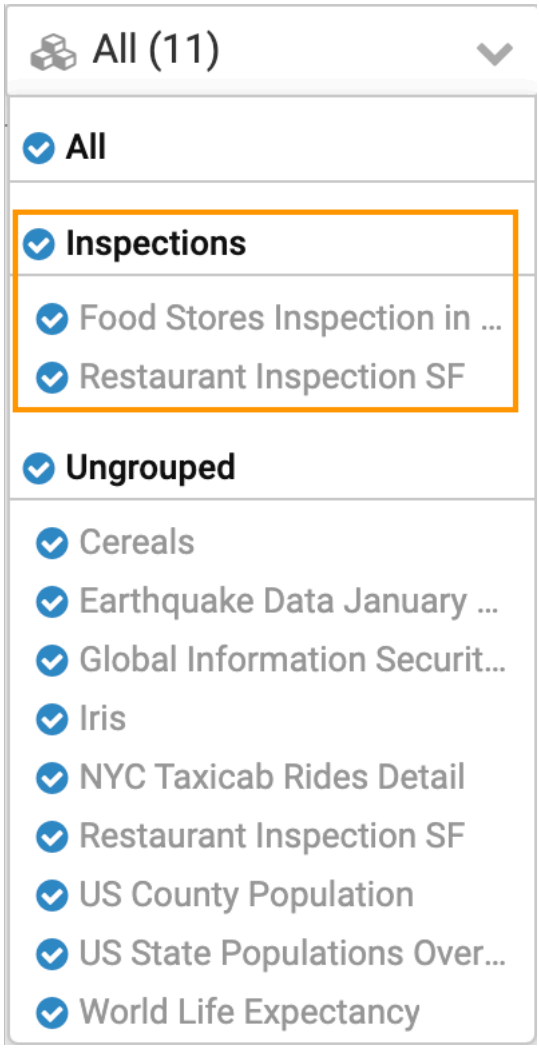

Specifying fields for a dataset in NLS

Cloudera Data Visualization makes it possible to limit Natural Language Search (NLS) to values of specified fields of the dataset.

About this task

You can restrict the search for a number of reasons, either from operational considerations, or for security reasons.

Procedure

1. Under Fields, specify which fields may be searched through NLS.

   In the example below, most of the fields are selected. If you want to disable search for a specific field in a dataset, simply remove all words and phrases from Matching search terms of that field. In the example below, the terms from Record Count were removed.

2. Choose an option from the drop-down list under Auto Search Priority.

   Higher priority columns are searched first, then medium, and finally low.

   Note: Enabling Auto Search Priority will result in additional queries even if the column is not specifically mentioned in the search and can affect performance. Only enable it for columns from which users want to pull information.

   Fields

Field	Data Type	Dimension or Aggregate	Matching search terms	Auto Search Priority
[Record Count]	BIGINT	Aggregate		NONE
[business_id]	BIGINT	Dimension	business_id	HIGH
[business_name]	STRING	Dimension	business establishment restaurant	LOW
[business_address]	STRING	Dimension	business_address	NONE
[business_city]	STRING	Dimension	business_city city	HIGH
[business_state]	STRING	Dimension	business_state state	MEDIUM
[business_postal_code]	STRING	Dimension	"postal code" zip	LOW
[business_latitude]	DOUBLE	Dimension	"business latitude" lat	LOW

3. Click SAVE.

Specifying synonyms for a dataset in NLS

With Cloudera Data Visualization, you can specify synonyms for dataset fields; these words can be used in Natural Language Search (NLS).

Procedure

1. To specify alternate words for the dataset fields, you can add them to the column Matching search terms. Separate synonyms by including a single space, and use quotations for phrases.
2. Under Fields, add synonymous terms that can be searched through NLS.

   In the example below the terms 'business name', 'establishment', 'business', 'restaurant', and 'cafe' are added for the column ‘Business Name’.

   Fields

Field	Data Type	Dimension or Aggregate	Matching search terms	Auto Search Priority
[Record Count]	BIGINT	Aggregate		NONE
[business_id]	BIGINT	Dimension	business_id	NONE
[business_name]	STRING	Dimension	“business name” establishment business restaurant cafe	LOW

3. Click SAVE.

Specifying default aggregation field for a dataset in NLS

In Cloudera Data Visualization, you can specify which field's values report the default aggregation.

About this task

When a search phrase does not specify an aggregate measure, Cloudera Data Visualization reports results of a default column as an aggregate. In the majority of cases, this default column is Record Count, a default aggregation measurement generated by the system.

In this example, inspection_score is selected to illustrate this functionality.

Procedure

1. Navigate to the Fields page on the dataset page.
2. Click Edit Fields.

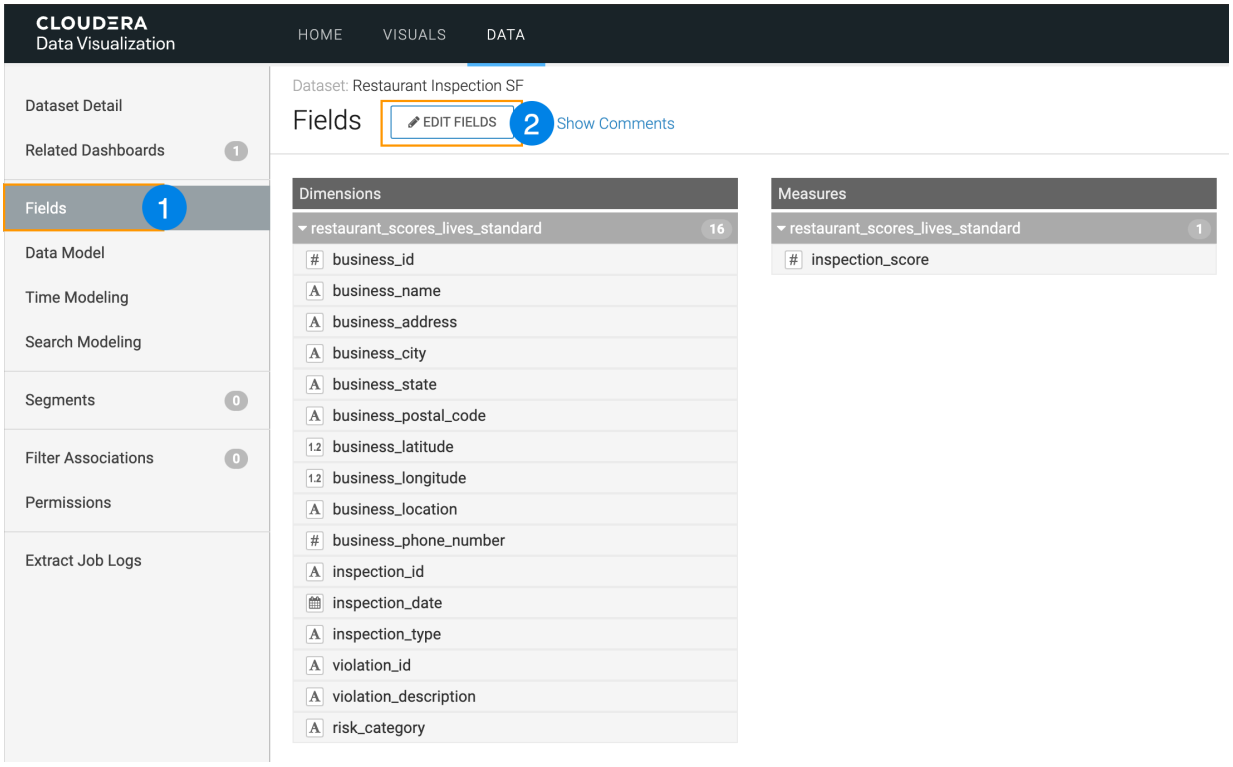

3. Click Edit Field for the field that you want to use as an aggregate.

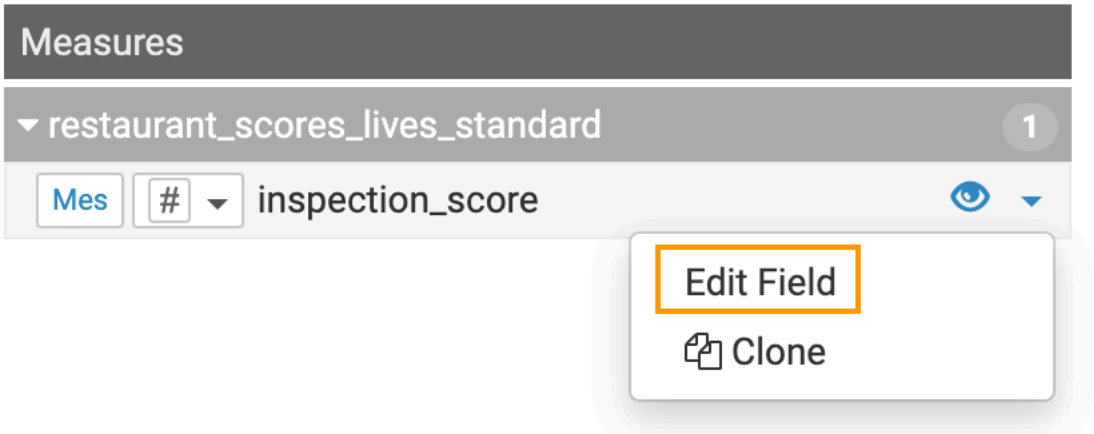

4. Choose a Default Aggregation type from the drop-down list.

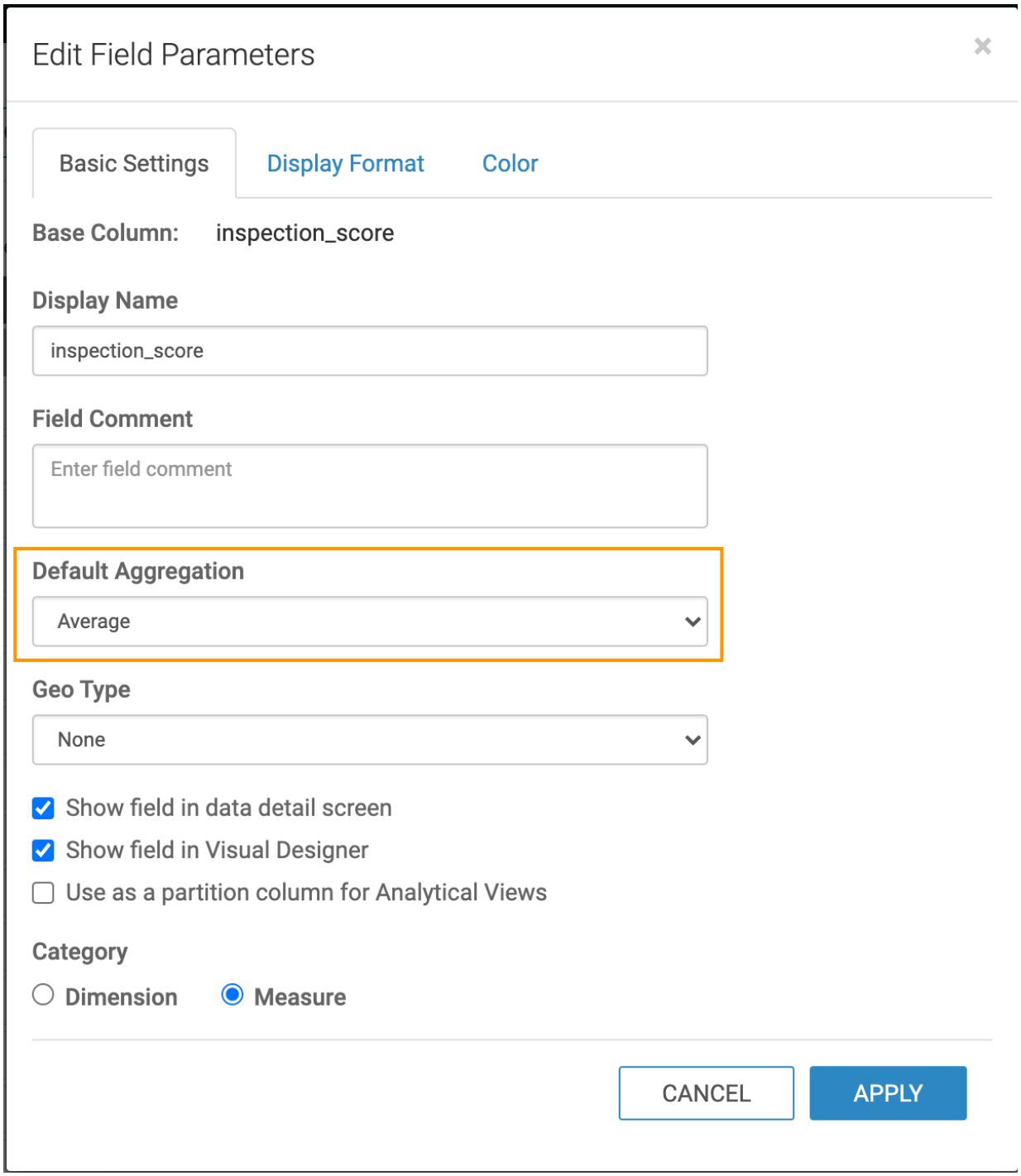

5. Click APPLY.
6. Navigate to the Search Modeling page.

7. Select the Default Aggregation Field from the drop-down list.

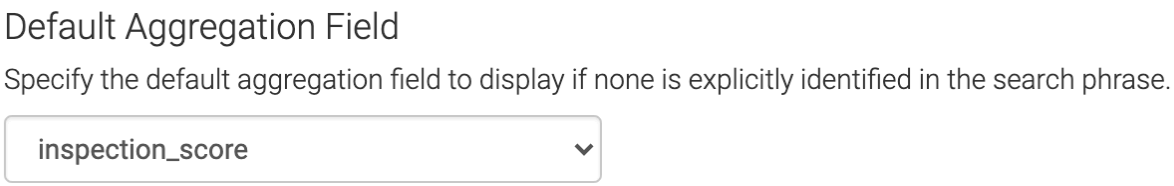

8. Click SAVE.

Results

If you type 'inspection score' in the search bar you get an average score as a result.

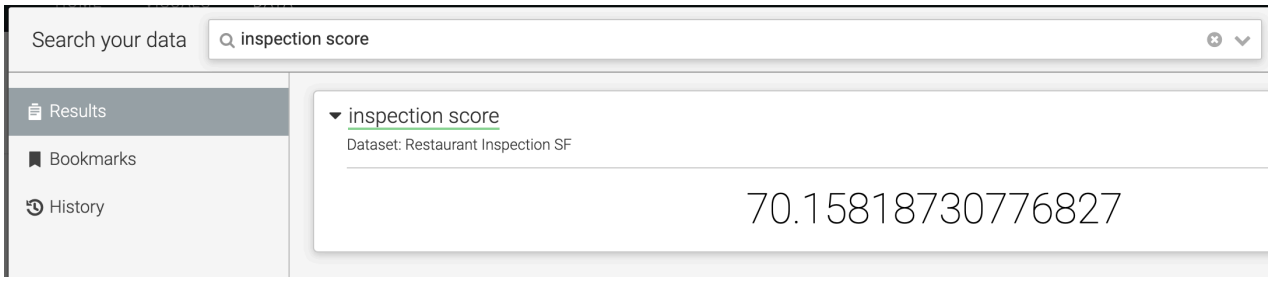

Specifying suggested questions for a dataset in NLS

You can help users explore the dataset by pre-populating the dropdown with some search words and phrases in the dataset. You can also preview the results of the searches.

Procedure

1. Below Suggested Questions, click Add New Item.

2. In the text box, enter the search phrase.

   In this example, several suggestions have been added.

   Suggested Questions

   List the questions the users see when using search for this dataset.

Suggestion	Try it	
top 10 violations		
trends by year		
top 10 restaurant violations by zip		
restaurant location where score is < 50		
restaurant trends in downtown sf as line		
top 10 restaurants where risk is "high risk"		
top 20 inspection count by business		

   + Add New Item

3. Click the Search icon to preview the visual results.

   A separate modal window appears, showing your results.
4. Click SAVE.

   Note: To remove a search phrase, click the Trash icon next to it.

Specifying word substitutions for a dataset in NLS

Word substitutions enable you to connect common words with valid data specifications when using Natural Language Search (NLS) on Cloudera Data Visualization datasets.

About this task

Common words and phrases have specific meaning in each dataset. In this example, you can see how to specify the meaning of the phrase 'current year' and the word 'domestic' into internal search terms.

Procedure

1. Under Word Substitutions, click Add New Item.
2. In the new row, add a common speech component in the Input word or phrase column.

3. In the same row, under Internal search term, specify the equivalent term in the dataset.

   In this example, the phrase 'current year' was added with the clause 'year:2019', setting the current year to 2019.

   Word Substitutions

   List simple word or phrase substitutions that translate into internal search terms.

   Examples: "domestic" → "country:'US'", "this year" → "year:2021", "location" → "latitude and longitude"

Input word or phrase	Internal search term	
current year	year:2019	

   + Add New Item

4. Click Add New Item again, if you want to add more word substitutions.

Configuring date and time for search requirements

To configure date and time settings on datasets in Cloudera Data Visualization, follow the steps below.

Procedure

1. On the main navigation bar, click DATA.
2. In the list of datasets, click the one you want to enable for Natural Language Search (NLS).

   The Dataset Detail page appears.

3. On the left navigation menu of the Dataset Detail page, click Time Modeling.

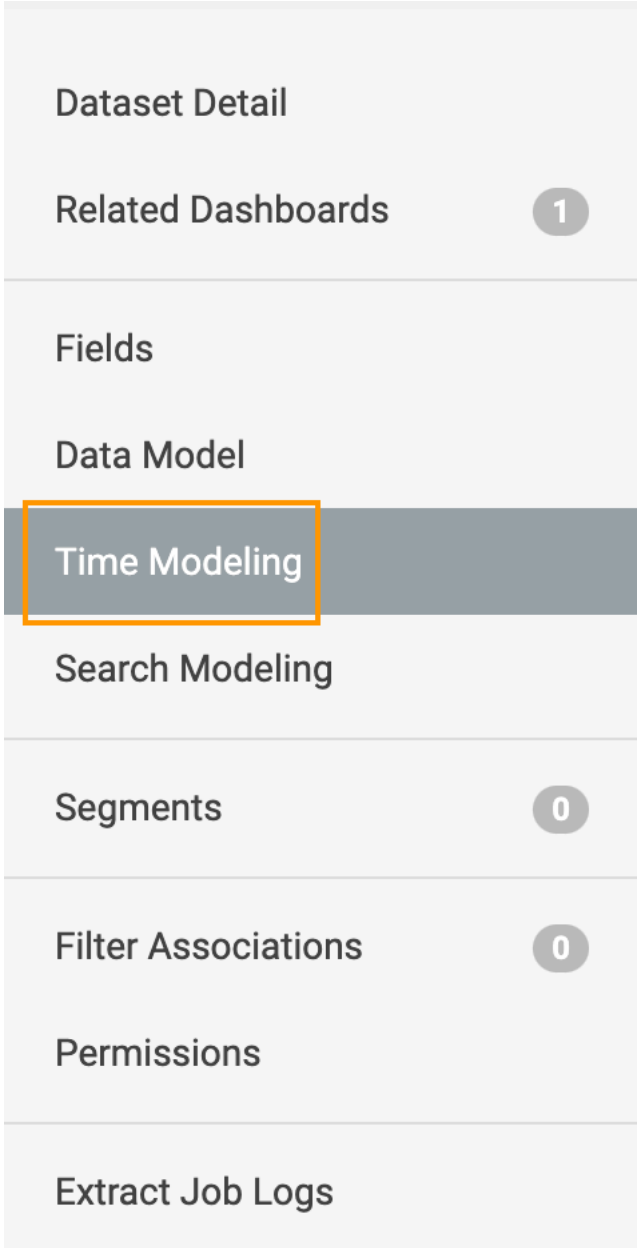

Specifying date/time field format for a dataset

About this task

On the Time Modeling page, specify the Date/Time Field of the dataset to ensure that Cloudera Data Visualization can correctly interpret and utilize the date/time information for search and analysis.

Procedure

1. At the top of the page, click EDIT.

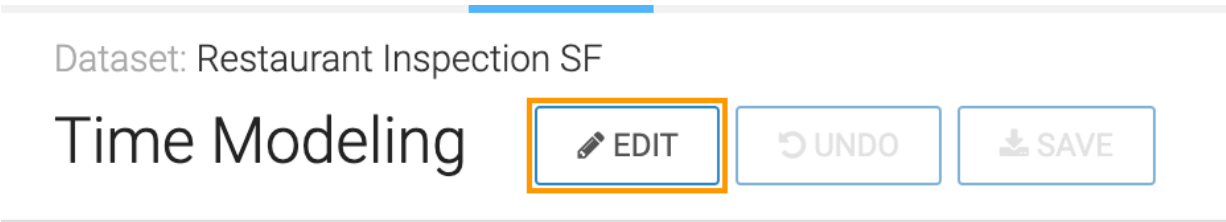

2. In the Date/Time Field section, select the name of the correct field that contains the date information from the first drop-down list.

   The system groups dataset fields in timestamp or other time formats at the top, under the heading Date Fields.

   In this example, the 'inspection_date' field has been selected.

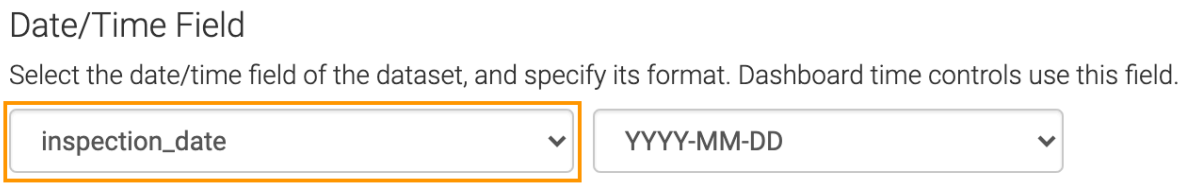

3. In the Date/Time Field section, select the appropriate date format mask from the second drop-down list.

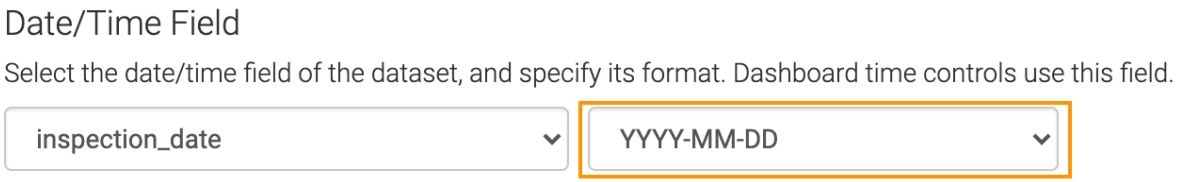

   Depending on your location, the masks may have a different format. For example, the two options in this example are YYYY-MM-DD and YYYY-MM-DD HH:mm:ss. In this example, the dataset does not contain the hour, minute, and second information, so the default format YYYY-MM-DD is kept.

   Note: If the dataset's date format differs from the options provided, you must create a computed column to parse the date into the desired format.

4. Click SAVE to apply your changes..

Specifying default time interval in dataset

About this task

On the Time Modeling page, specify the default time interval for data aggregation and analysis to ensure that time-based filters, visualizations, and searches are applied consistently across the dataset, based on the selected interval.

Procedure

1. In the Date/Time Field section, click the third drop-down list for Default Interval.

2. From the options provided, select the appropriate interval for analyzing metric trends.

   Depending on your location, you may have different options. For example: Month, Quarter, or Year.

   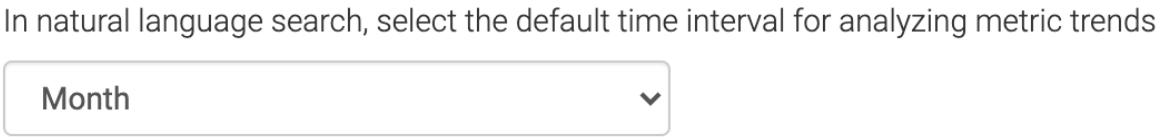

3. Click SAVE to confirm your selection.

Cloudera Data Visualization 7.2.9

Connections

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Data connections in Cloudera Data Visualization.....4

User impersonation in Cloudera Data Warehouse..... 4

Data connections in Cloudera Data Visualization

Cloudera Data Visualization allows you to create connections to many types of external data sources. You can set up connections to the different data sources that you can then use to build visuals.

The following connection types are supported:

- Hive
- Impala
- MariaDB
- MySQL
- PostgreSQL
- Solr [Technical Preview]
- Spark SQL
- SQLite (not supported in Cloudera Data Warehouse)
- Snowflake [Technical Preview]

For more information, see *Connecting to data*.

Related Information

Connecting to data

User impersonation in Cloudera Data Warehouse

User impersonation allows Cloudera Data Visualization to execute queries in Cloudera Data Warehouse on behalf of logged in users through a trusted service account.

When setting up a Hive or Impala data connection within the same Cloudera Data Warehouse environment as your Cloudera Data Visualization instance, the domain is considered secure and trusted and a default service account can be used for authentication. Impersonation in the Advanced settings is turned on by default. During setup, when a trusted connection is selected, the necessary information is auto-populated. These are passwordless connections, so only the username field is populated with the service user.

As a result of impersonation all incoming queries are evaluated against the data security policies of the users' setup in Ranger. From the query logs and profiles, you can see the trusted service user appears as the Connected User for authentication. The Cloudera Data Visualization user appears as the Delegated User for authorization, and their permissions will be evaluated before query runtime to ensure they have access to the data they are requesting.

Cloudera Data Visualization 7.2.9

Connecting to Data

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Connecting to a data source in Cloudera Data Visualization..... 4
Creating a Cloudera AI data connection to Impala..... 4
Creating a Cloudera AI data connection to a Hive data warehouse..... 12
Creating a Cloudera Data Warehouse data connection in Cloudera Data Visualization..... 18
Creating a Cloudera Data Science Workbench data connection to a data warehouse..... 21

Managing data connections..... 23
Editing a data connection..... 23
Setting concurrency for a data connection..... 26
Setting a warning if the query loads for too long..... 27
Deleting a data connection..... 29

Using the Connection Explorer..... 30
Discovering the Connection Explorer interface..... 31
Previewing data table details..... 33
Sample data..... 33
Datasets..... 33

Using the Direct Access interface..... 34
Running a SQL query in Direct Access..... 34
Downloading the results of a Direct Access query..... 36

Adding data..... 36
Importing data in CSV format..... 37
Adding data through URL..... 40
Changing the locale setting of an imported file..... 43
Preparing Excel files for data import..... 44

Connecting to a data source in Cloudera Data Visualization

Cloudera Data Visualization allows you to create connections to many types of external data sources, enhancing your data analysis and visualization capabilities.

Cloudera Data Visualization currently supports the following connection types:

- Hive
- Impala
- MariaDB
- MySQL
- PostgreSQL
- Solr [Technical Preview]
- Spark SQL
- SQLite (not supported in CDW)
- Snowflake [Technical Preview]

This range of supported connection types provides flexibility and versatility in connecting to various data sources for comprehensive data analysis.

When using Cloudera Data Visualization with Cloudera Data Warehouse, the data connection is automatically set up, but you can connect to various other data sources as well to suit your specific requirements.

In Cloudera AI, you can connect to an Impala or a Hive data warehouse, or tie in data from predictive Cloudera AI models.

Related Information

Connections

Datasets

Data modeling

Creating a Cloudera AI data connection to Impala

Learn how to connect natively to data stored in Impala when using Cloudera Data Visualization in Cloudera AI.

About this task

Before you start using data modeling and visualization functions, you must connect to your data. The following steps show you how to create a new Cloudera AI data connection in Cloudera Data Visualization to an Impala data warehouse.

Note:

You must have the Manage data connections privilege or be an admin to be able to create and manage connections in Cloudera Data Visualization.

Setting user privileges requires administrator-level access. You can log in as an administrator using the default admin account with the following credentials:

- Username: vizapps_admin
- Password: vizapps_admin

When you create a connection, you automatically get privileges to create and manage the associated datasets. You can also build dashboards and visuals within these datasets.

- For more information on the Manage data connections privilege, see *RBAC permissions*.

- For instructions on how to define privileges for a specific role, see *Setting role privileges*.
- For instructions on how to assign the administrator role to a user, see *Promoting a user to administrator*.

Before you begin

If you are using a Cloudera Private Cloud Base cluster running Impala with Kerberos for authentication, make sure that Kerberos credentials are configured in Cloudera AI before creating a Cloudera AI data connection to the Impala data warehouse. This ensures seamless integration and authentication between Cloudera Data Visualization and the Impala cluster. If you add Kerberos credentials after launching the Cloudera Data Visualization app, you need to restart the app for the changes to take effect.

For more information on using Kerberos for authentication in Cloudera AI, see *Hadoop Authentication for AI Workspaces*.

Procedure

1. On the main navigation bar, click DATA.

   The DATA interface opens, displaying the Datasets tab.

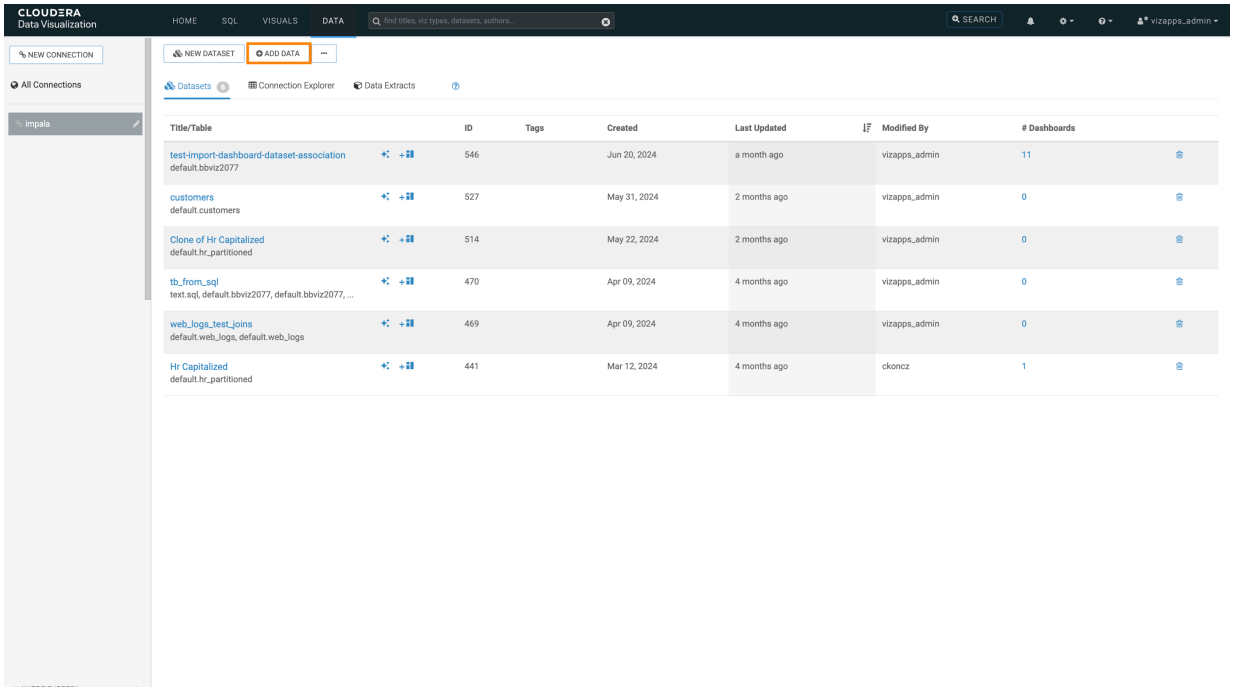

2. On the side menu bar, click NEW CONNECTION.

Note: Only users with Manage data connections privilege or administrators can access the NEW CONNECTION button.

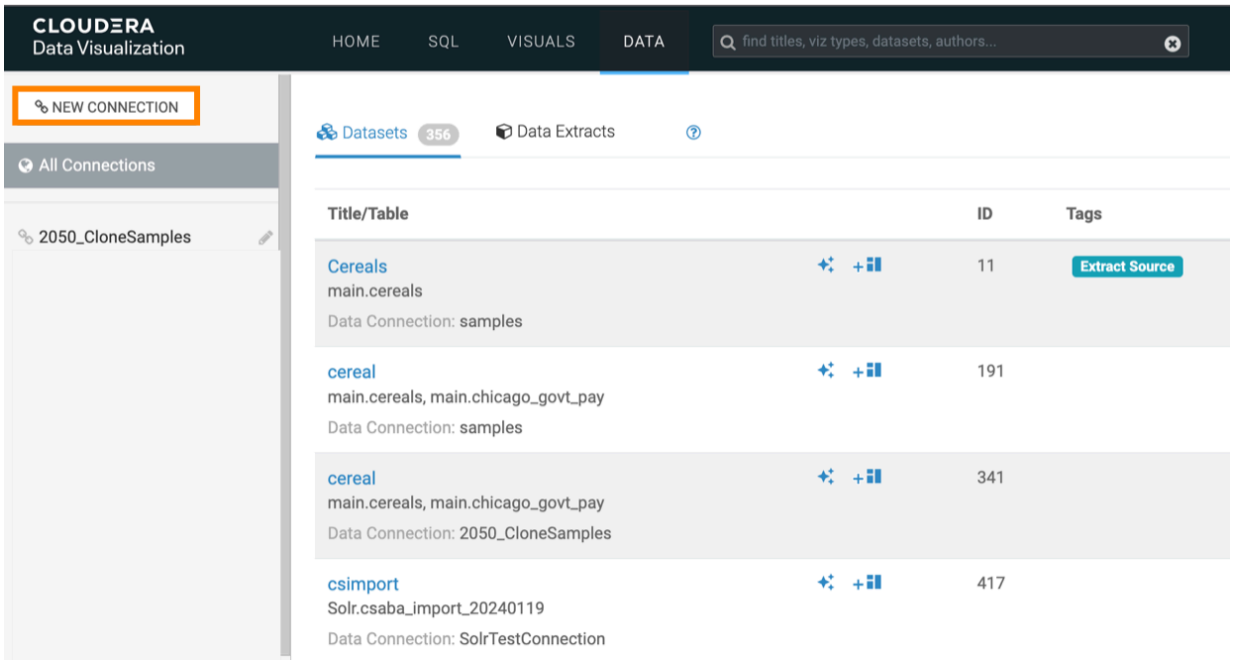

The Create New Data Connection modal window appears.

3. Choose Impala from the Connection type drop-down list and assign a name to your connection.

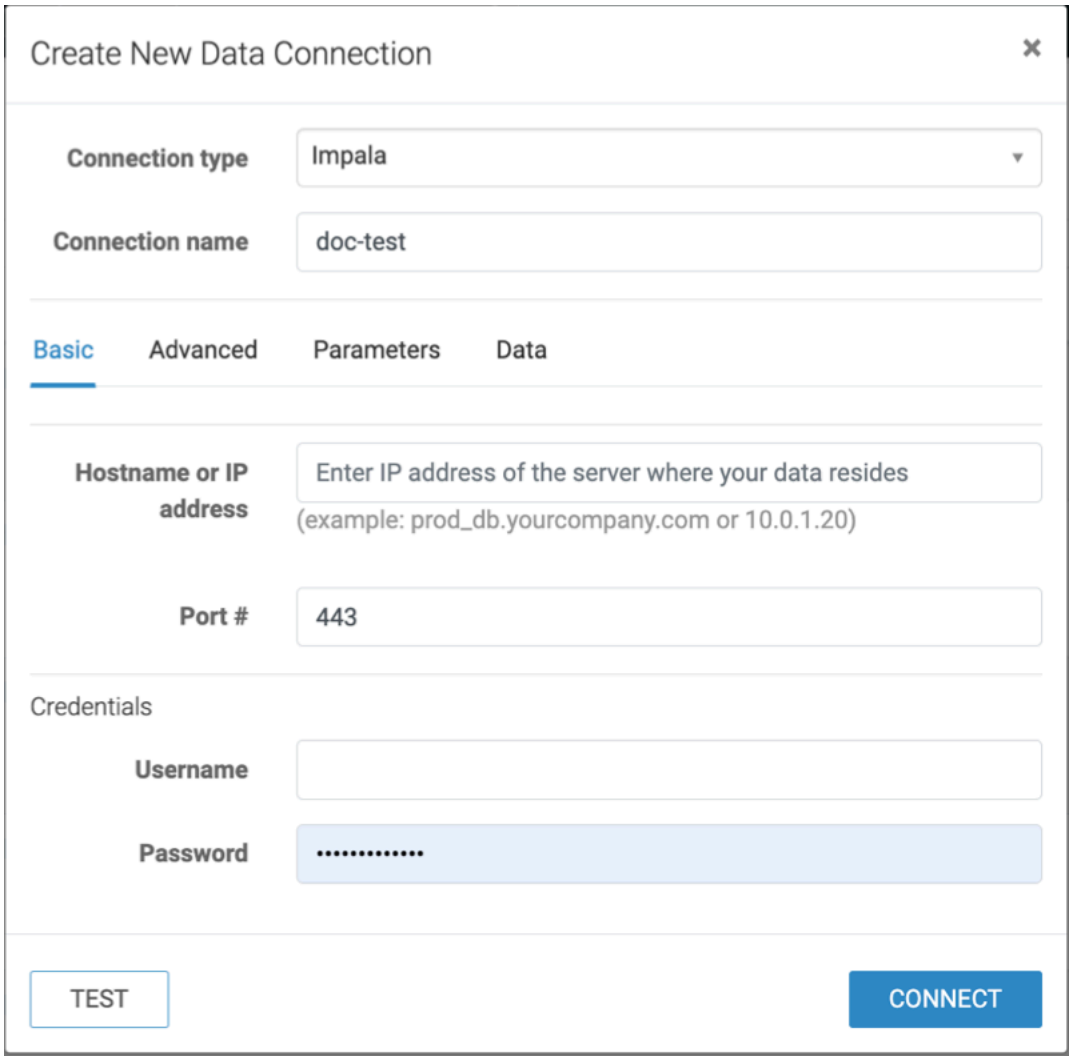

In this example, the Impala connection is made through Knox. Knox always uses TLS encryption and port 443 is the default HTTPS port.

4. Enter the hostname or IP address of the running coordinator.

   You can retrieve this information from the JDBC URL of the Impala DW.

5. Add 443 in the Port # field.
6. Enter your workload username and password as credentials.

7. Click the Advanced tab to configure additional details.

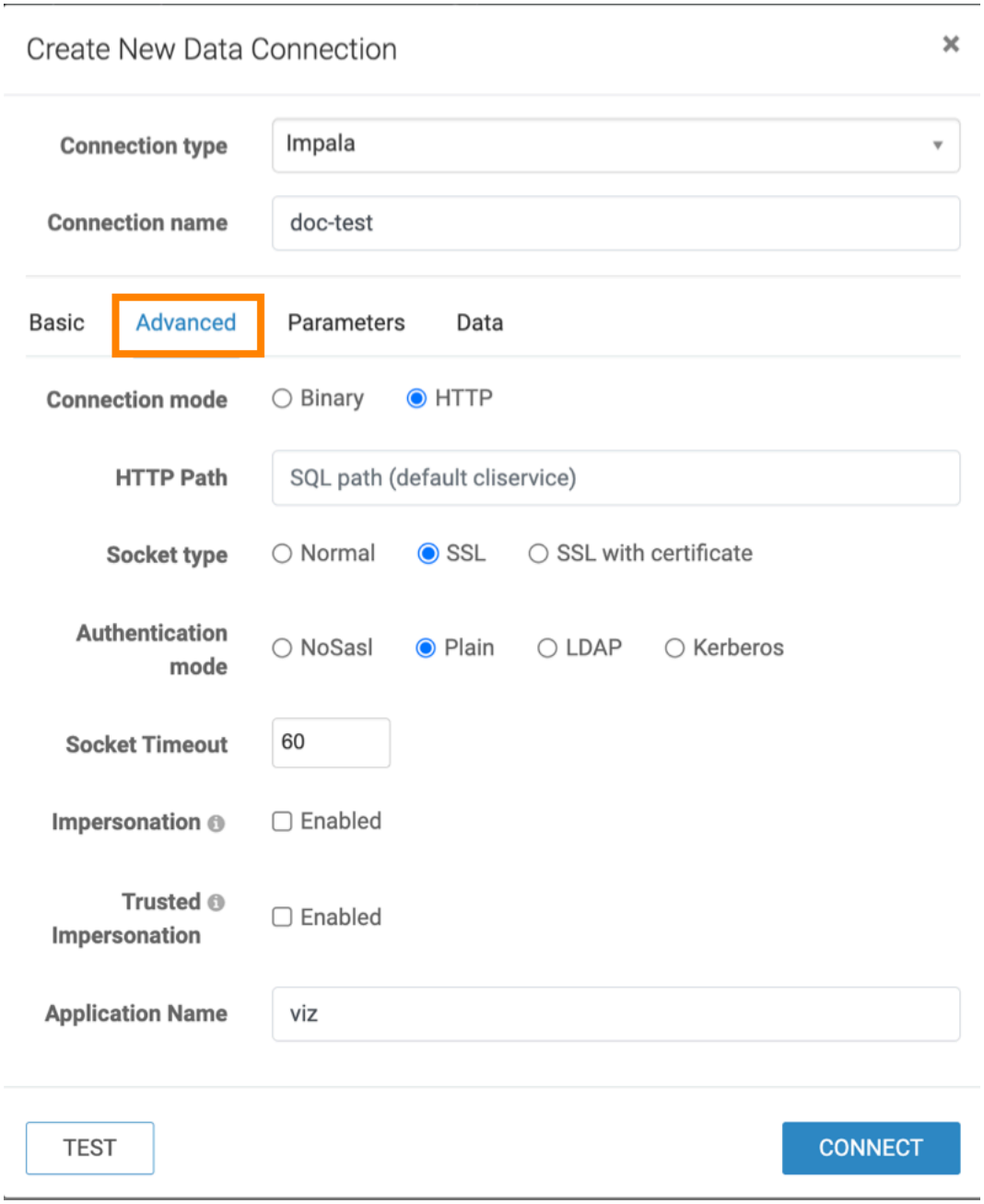

a) For HTTP connection mode, locate the Impala Endpoint for the Data Hub.

b) Copy and paste it into the HTTP Path field.

c) Set any additional details as required.

8. Check the Parameters and Data tabs for more configuration options.

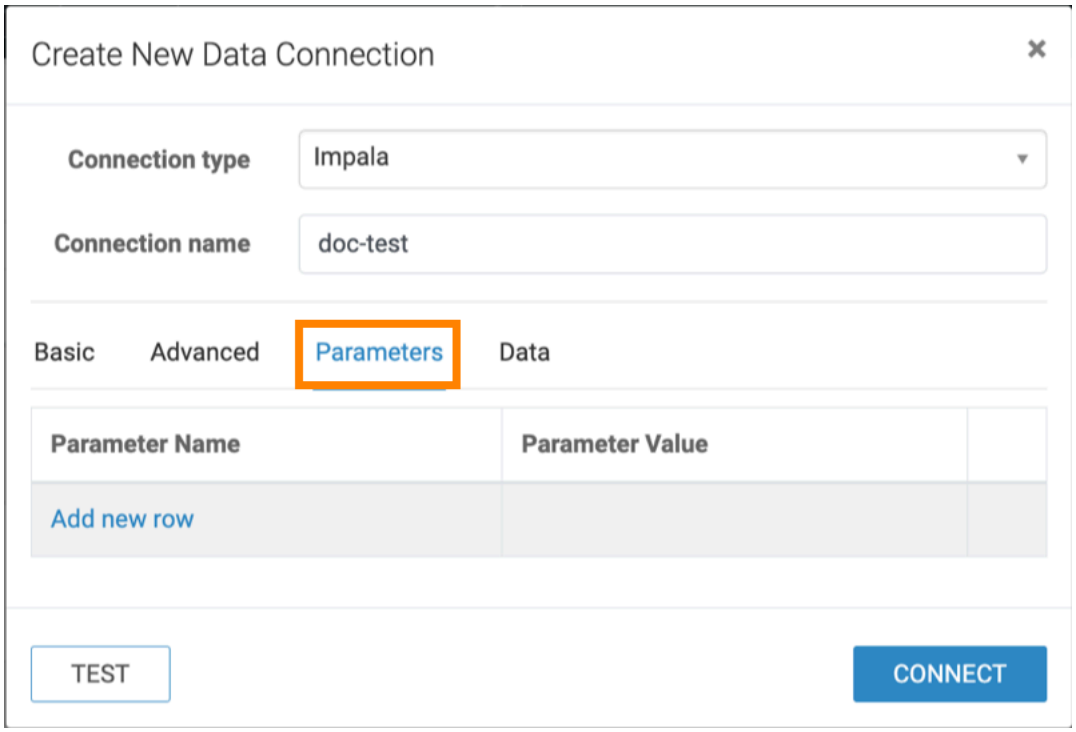

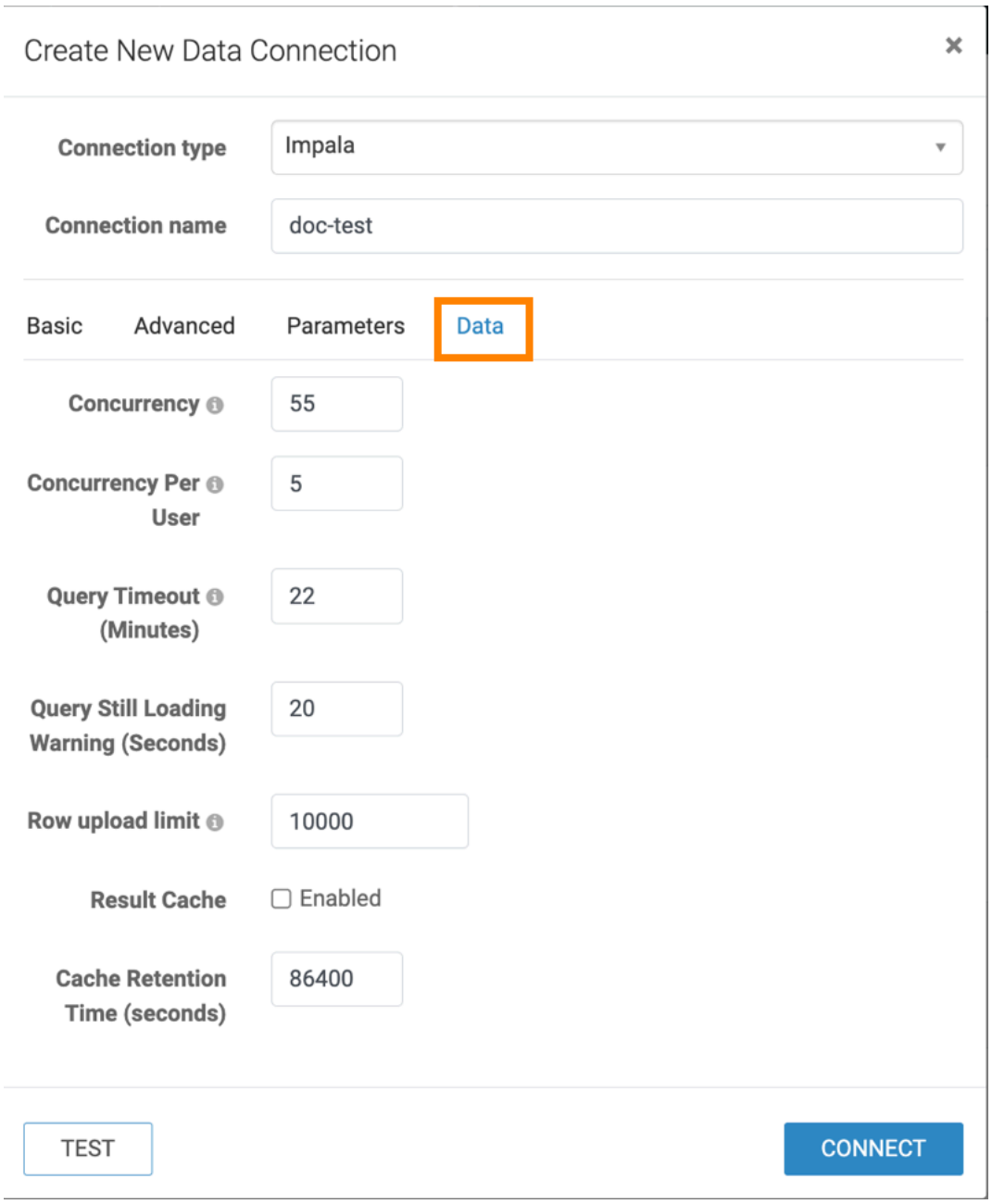

9. Once you finish configuring the settings, click TEST to check the connection.
10. Click CONNECT to establish the connection.

Results

You have successfully set up a connection to a running Impala DW.

Related Information

RBAC permissions
Setting role privileges
Promoting a user to administrator
Hadoop Authentication for ML Workspaces

Creating a Cloudera AI data connection to a Hive data warehouse

Learn how to connect natively to data stored in Hive when using Cloudera Data Visualization in Cloudera AI.

About this task

Before you start using data modeling and visualization functions, you must connect to your data. The following steps show you how to create a new Cloudera AI data connection in Cloudera Data Visualization to a Hive data warehouse.

Note:

You must have the Manage data connections privilege or be an admin to be able to create and manage connections in Cloudera Data Visualization.

Setting user privileges requires administrator-level access. You can log in as an administrator, using the default admin account with the following credentials:

- Username: vizapps_admin
- Password: vizapps_admin

When you create a connection, you automatically get privileges to create and manage the associated datasets. You can also build dashboards and visuals within these datasets.

- For more information on the Manage data connections privilege, see *RBAC permissions*.
- For instructions on how to define privileges for a specific role, see *Setting role privileges*.
- For instructions on how to assign the administrator role to a user, see *Promoting a user to administrator*.

Before you begin

If you are using a Cloudera Private Cloud Base cluster running Hive with Kerberos for authentication, make sure that Kerberos credentials are configured in Cloudera AI before creating a Cloudera AI data connection to the Hive data warehouse. This ensures seamless integration and authentication between Cloudera Data Visualization and the Hive cluster. If you add Kerberos credentials after launching the Cloudera Data Visualization app, you need to restart the app for the changes to take effect.

Procedure

1. On the main navigation bar, click DATA.

   The DATA opens, displaying the Datasets tab.

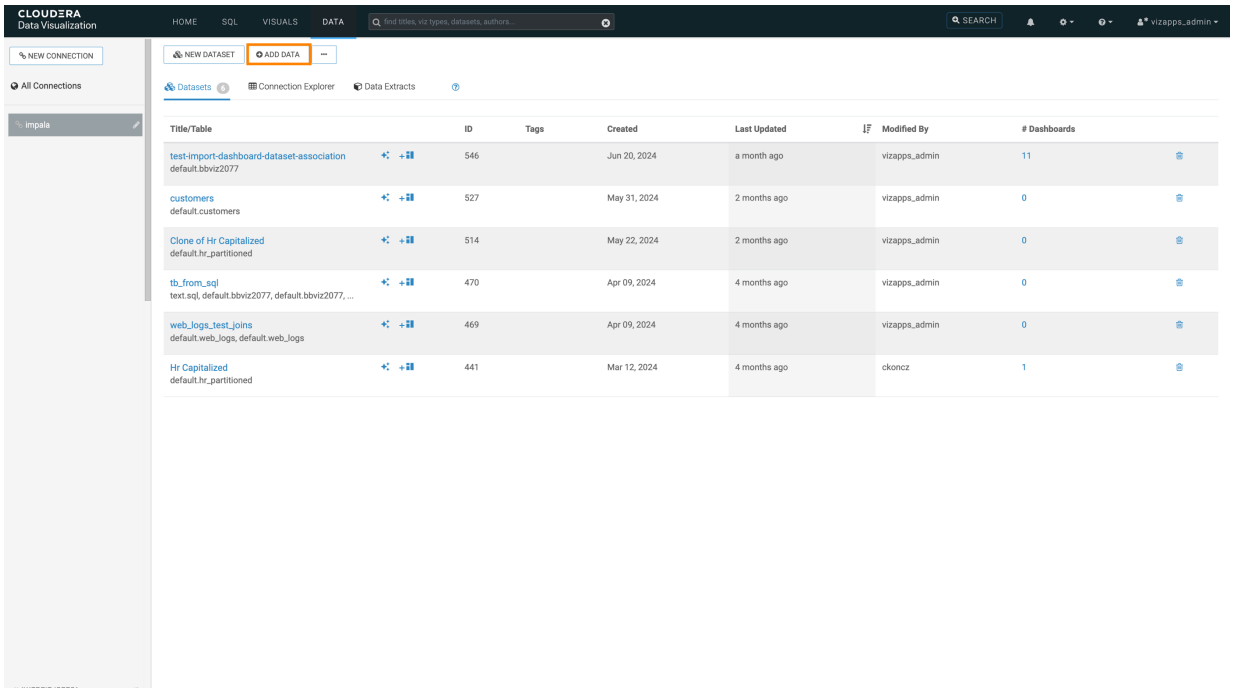

2. On the side menu bar, click NEW CONNECTION.

   Note: The NEW CONNECTION button is only accessible to users assigned to roles with the Manage data connections privilege and to administrators.

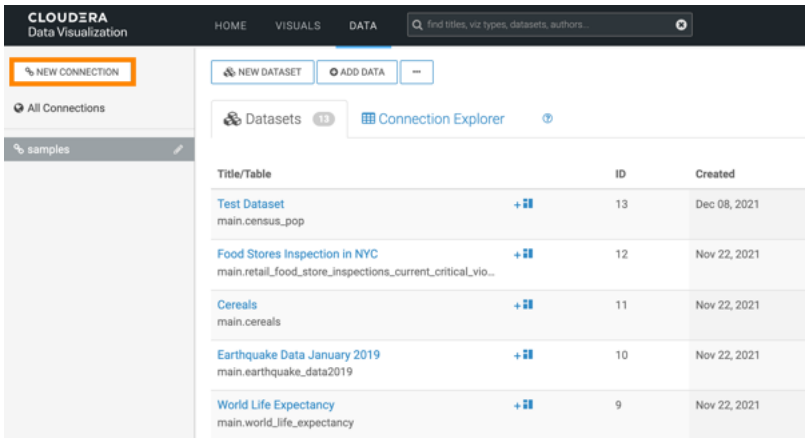

   The Create New Data Connection modal window appears.

3. Choose Hive from the Connection type drop-down list and assign a name to your connection.

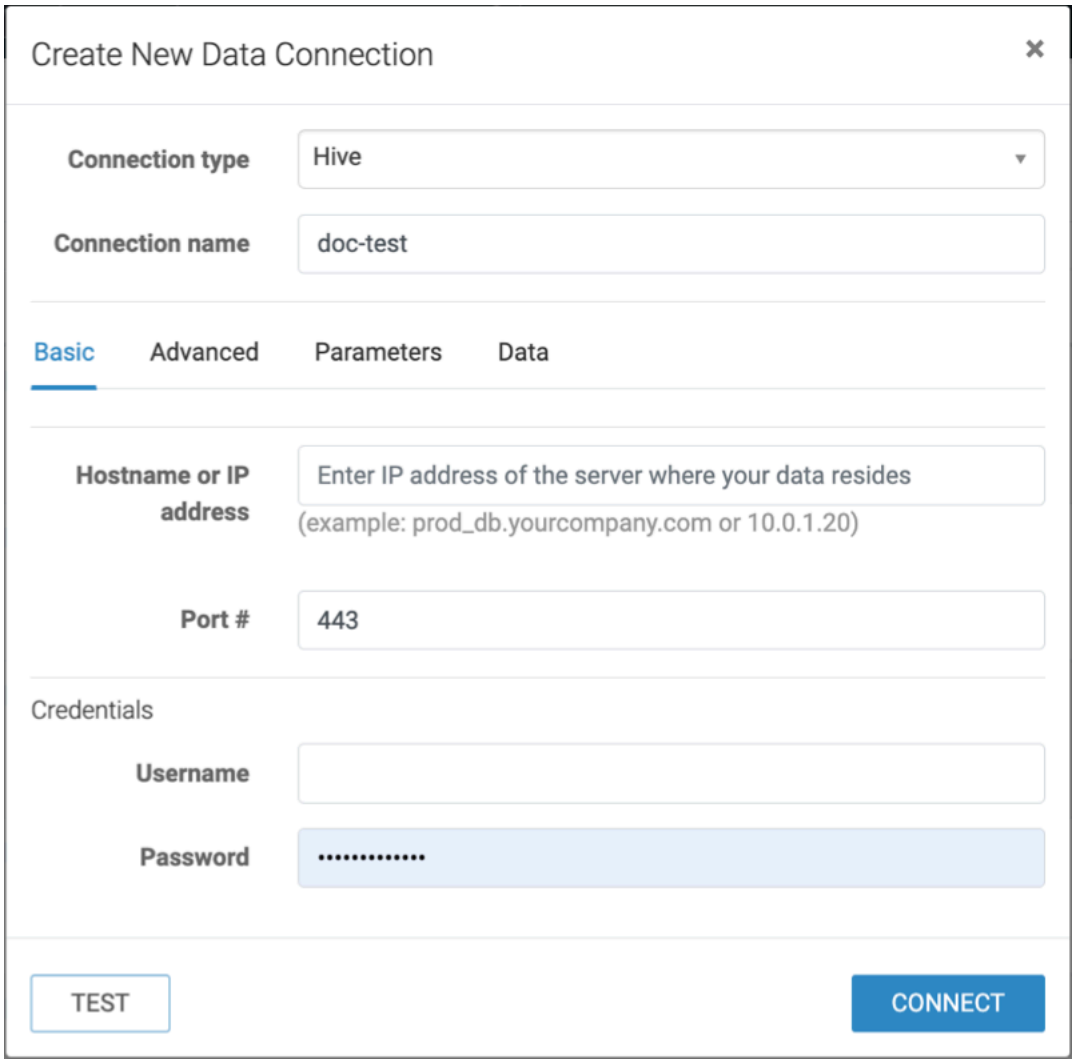

4. Enter the hostname or IP address of the running coordinator.
   You can get the coordinator hostname from the JDBC URL of the Hive DW.
5. Use port 443.
6. Enter your workload username and password as credentials.

7. Click the Advanced tab to configure the additional details.

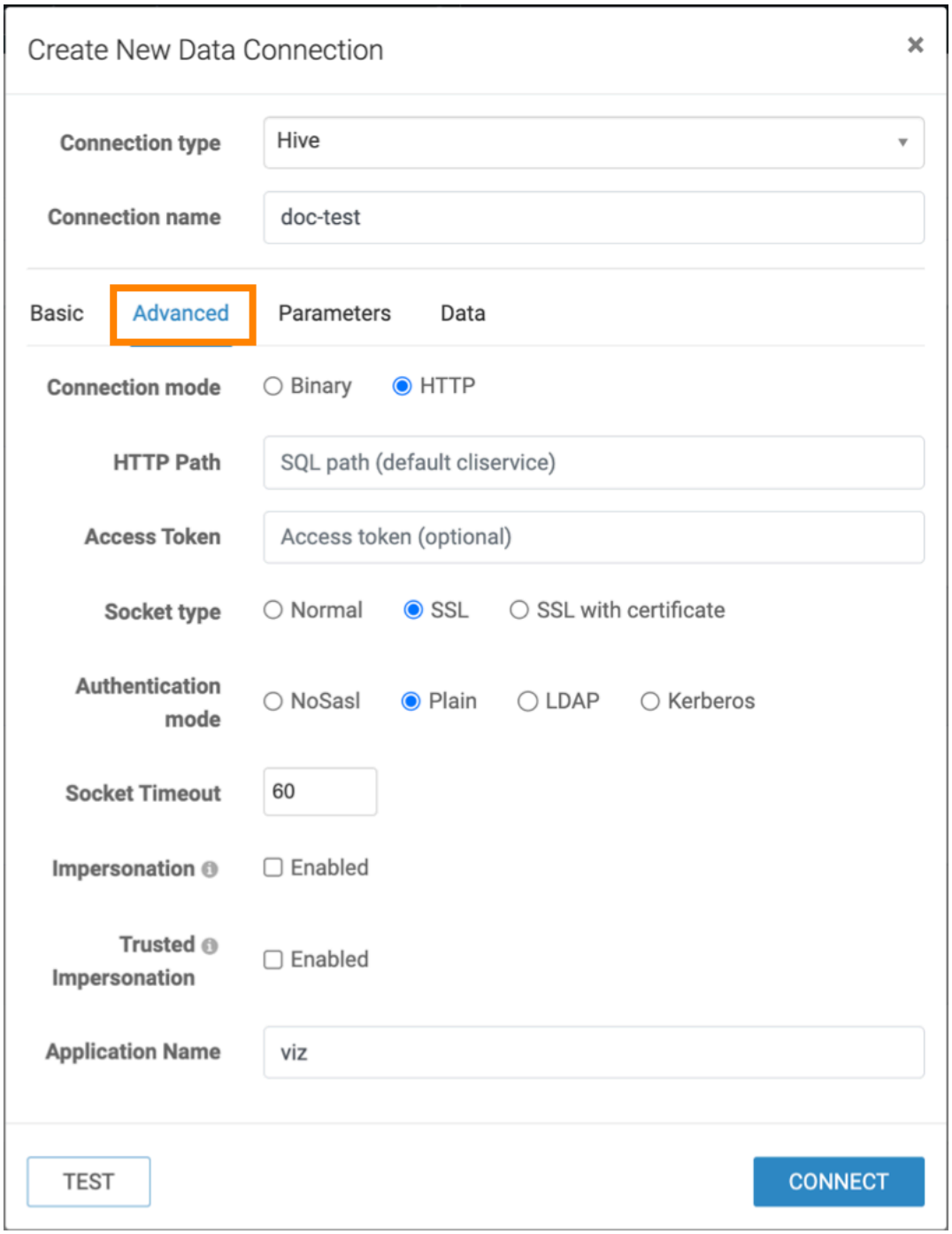

8. Click the Parameters tab and set the hive.server2.async.exec.async.compile parameter to false.

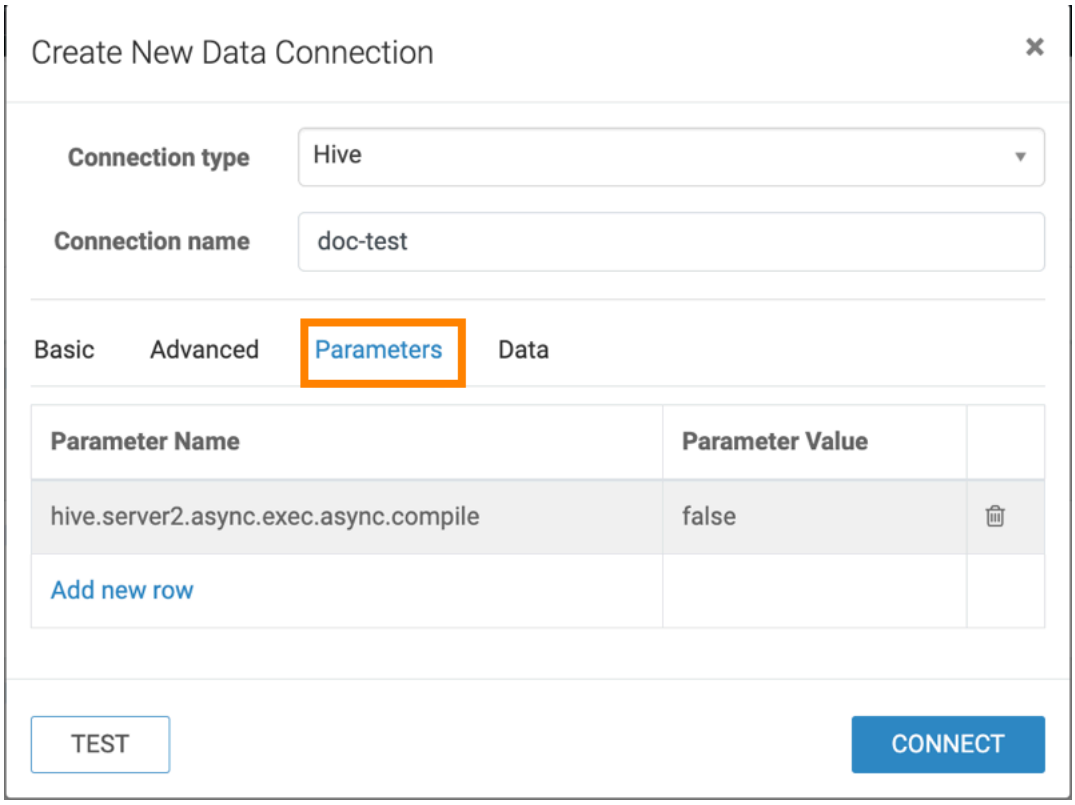

9. Check the Data tab for more configuration options.

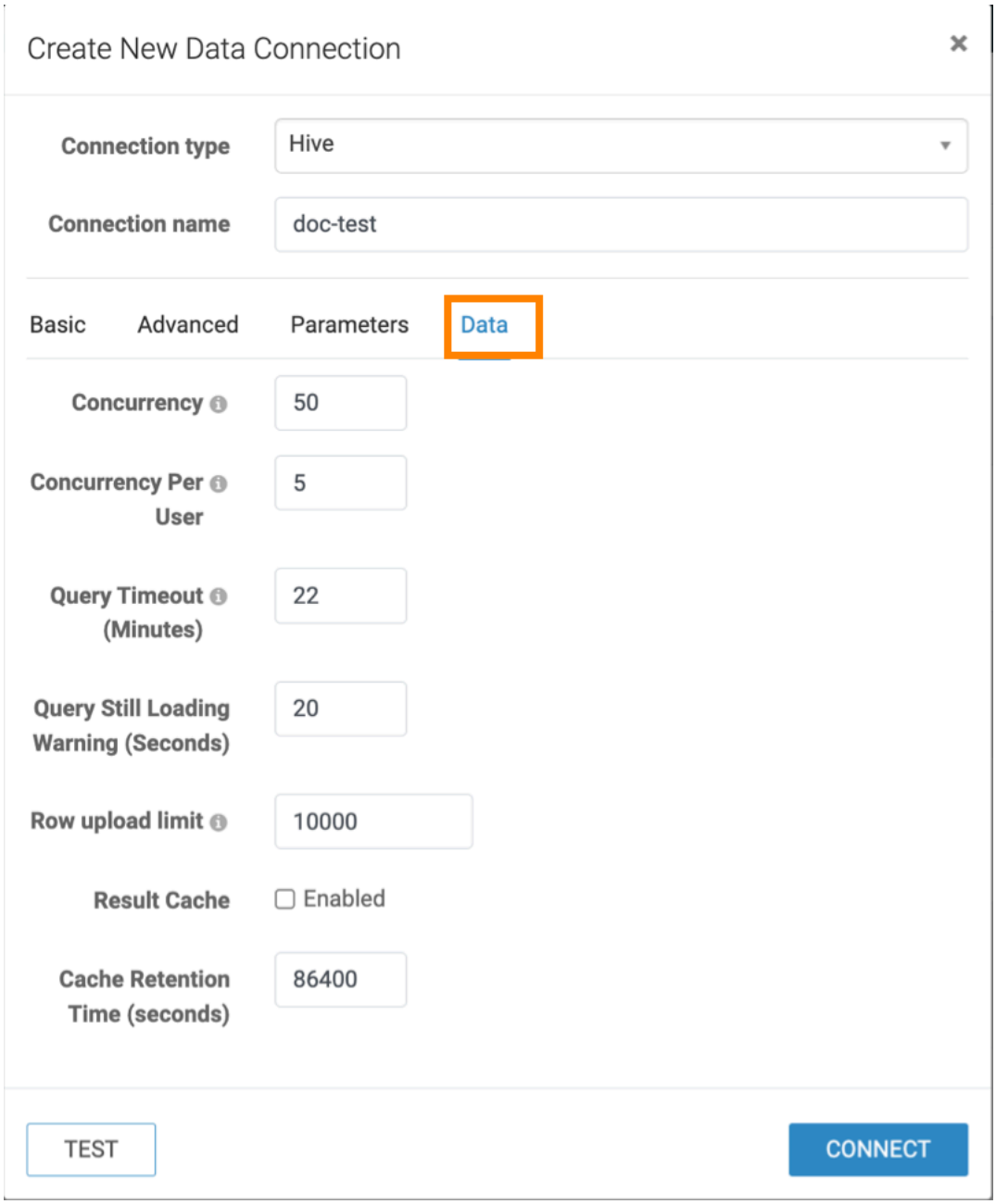

10. Once you finish configuring the settings, click TEST to test the connection.

11. Click CONNECT to establish the connection.

Results

You have successfully set up a connection to a running Hive DW.

Related Information

RBAC permissions

Setting role privileges

Promoting a user to administrator

Creating a Cloudera Data Warehouse data connection in Cloudera Data Visualization

Learn how to connect to data when using Cloudera Data Visualization in Cloudera Data Warehouse data service. You can connect Cloudera Data Visualization to a Virtual Warehouse to visualize your data. Similar to using a BI client, you can configure and connect to Virtual Warehouses from different clusters.

About this task

You must connect to your data prior to using the data modeling and visualization functions. You make the connection to the Virtual Warehouse when you select your warehouse in the steps below. The Cloudera Data Warehouse URL has the same compute instance ID as your Virtual Warehouse.

Note:

To create and manage connections in Cloudera Data Visualization, you must have the Manage data connections privilege or hold administrative privileges. In Cloudera Data Warehouse, these are the members of the Admin Groups associated with the Cloudera Data Visualization instance.

When you create a connection, you automatically gain privileges to create and manage datasets associated with this connection, and to build dashboards and visuals within these datasets.

- For more information on the Manage data connections privilege, see *RBAC permissions*.
- For instructions on how to define privileges for a specific role, see *Setting role privileges*.
- For instructions on how to assign the administrator role to a user, see *Promoting a user to administrator*.

When you are creating a Hive or Impala data connection within the same cluster, the connection is considered secure and trusted, and the connection details can be auto-populated with a default authentication user.

Procedure

1. Start Cloudera Data Visualization from the left navigation panel in Cloudera Data Warehouse.
2. On the main navigation bar, click DATA.

   The DATA interface appears, open on the Datasets tab.

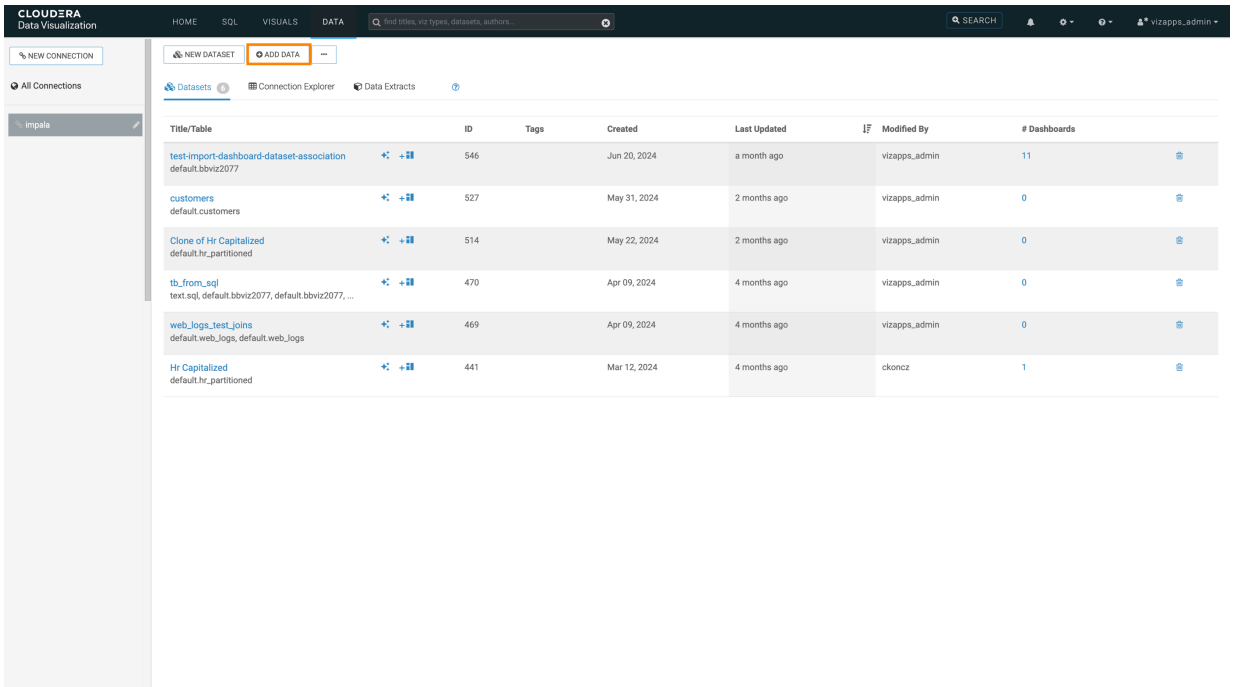

3. In the side menu bar of DATA, click NEW CONNECTION.

Note: The NEW CONNECTION button is only accessible to users assigned to roles with the Manage data connections privilege and to administrators.

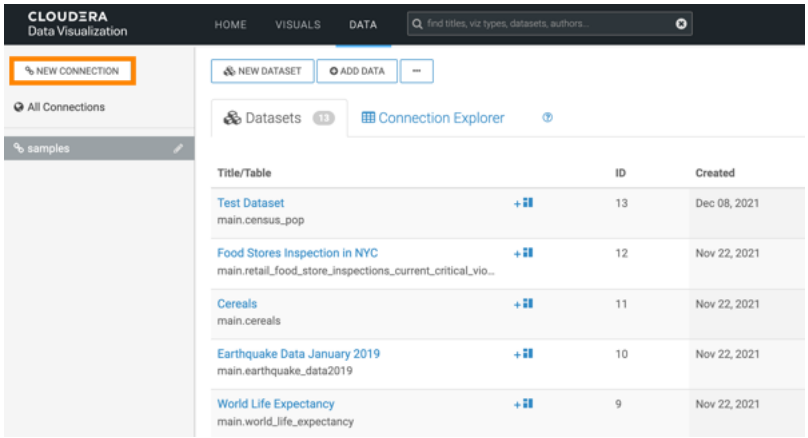

The **Create New Data Connection** modal window appears.

4. In Connection type, select CDW Hive or CDW Impala, and provide a name for your connection.

   Note: SQLite connection is not supported in Cloudera Data Warehouse, and the option is disabled in the connection list.

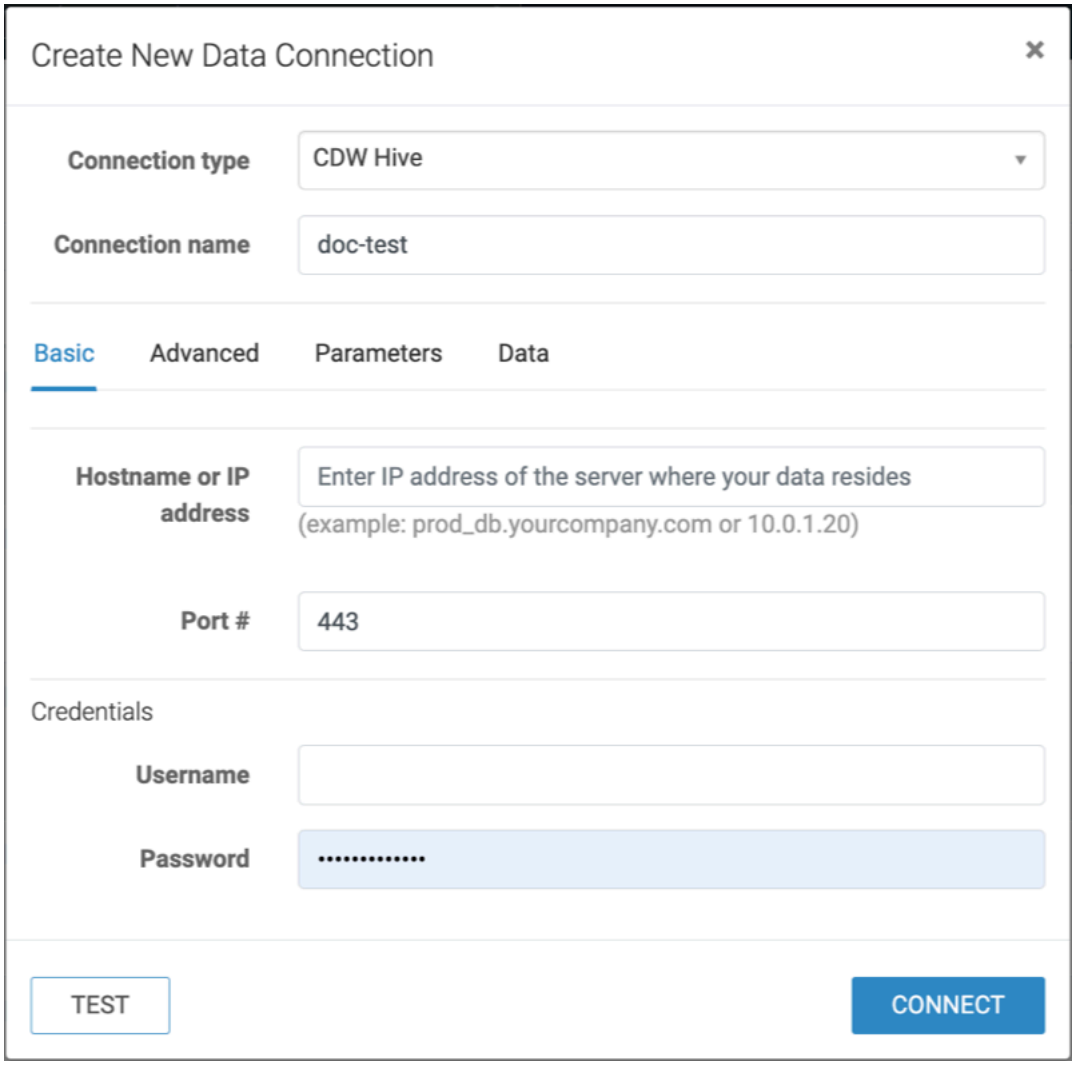

5. Select a Cloudera Data Warehouse warehouse to connect to.

   For Data connection within the same cluster

   The following fields are auto-populated:

   - Hostname or IP address
   - Port #
   - Username

   For Data connection outside the cluster

   Enter the following information:

   - Hostname or IP address
   - Port #
   - Username
   - Password

6. Click the Advanced tab and configure the additional details.

   Important: Depending on the type of connection you are creating, there can be additional tabs in the Create New Data Connection modal window where you have to adjust further settings.

7. Click TEST.

   If the connection is valid, the system returns a Connection Verified message.

8. Click CONNECT.

What to do next

You can create a data set, and then start creating visuals, dashboards, and applications. For more information, see *Creating datasets* and *Creating a visual*.

Related Information

RBAC permissions

Setting role privileges

Promoting a user to administrator

Creating datasets

Creating a visual

Creating a Cloudera Data Science Workbench data connection to a data warehouse

Learn how to connect natively to data stored in a data warehouse when using Cloudera Data Visualization in Cloudera Data Science Workbench.

About this task

You must connect to your data prior to using the data modeling and visualization functionalities. The following steps show you how to create a new Cloudera Data Science Workbench data connection to a running Impala system.

Note:

To create and manage connections in Cloudera Data Visualization, you must have the Manage data connections privilege or hold administrative privileges.

Setting user privileges requires administrator-level access. To log in as an administrator, you can use the default admin account with the following credentials:

- Username: vizapps_admin
- Password: vizapps_admin

When you create a connection, you automatically gain privileges to create and manage datasets associated with this connection, and to build dashboards and visuals within these datasets.

- For more information on the Manage data connections privilege, see *RBAC permissions*.
- For instructions on how to define privileges for a specific role, see *Setting role privileges*.
- For instructions on how to assign the administrator role to a user, see *Promoting a user to administrator*.

Procedure

1. On the main navigation bar, click DATA.

   The DATA interface appears, open on the Datasets tab.

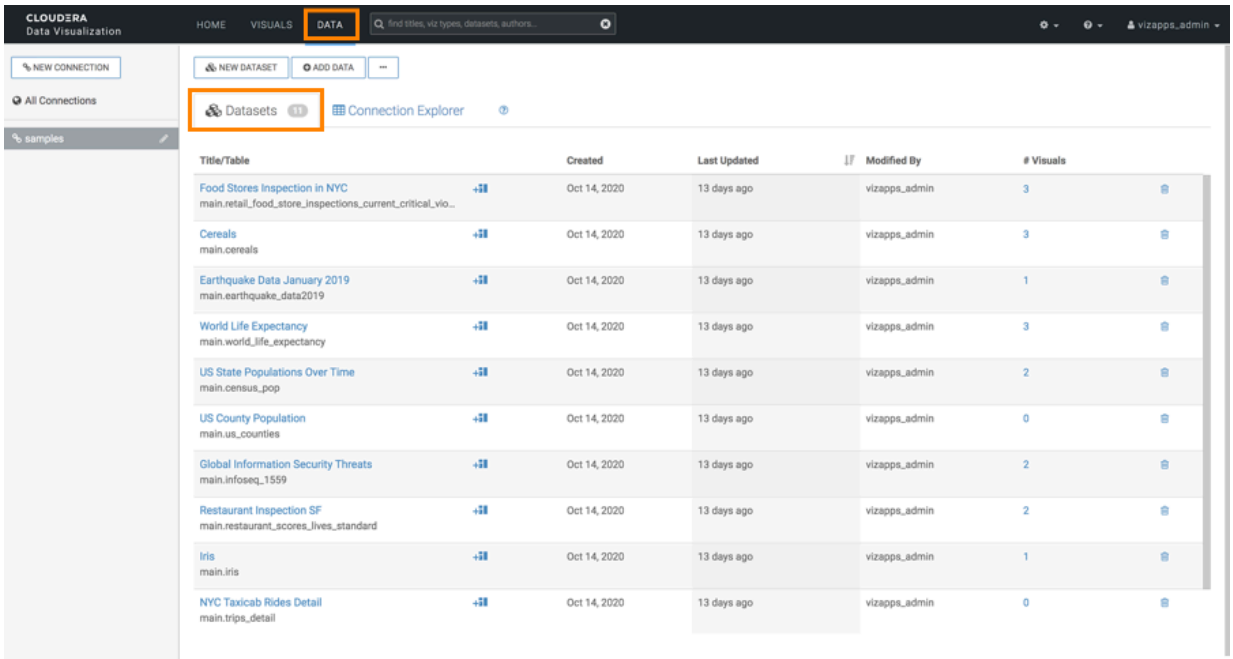

2. In the Data side menu bar, click NEW CONNECTION.

   Note: The NEW CONNECTION button is only accessible to users assigned to roles with the Manage data connections privilege and to administrators.

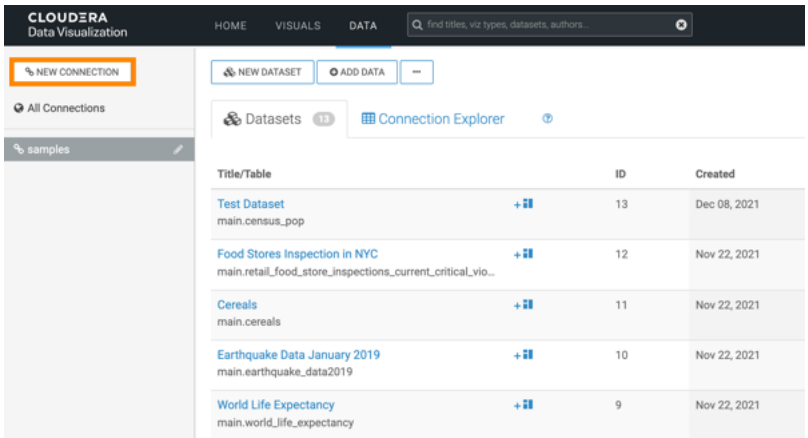

   The Create New Data Connection modal window appears.

3. Select a Connection type from the drop-down list, and provide a name for the connection.
4. Enter the hostname or IP address of the running coordinator.
5. Under Port #, enter the port number.
6. Use your workload username and password as credentials.
7. Click the Advanced tab and make the appropriate selections.

   Important: Depending on the type of connection you are creating, there can be additional tabs in the Create New Data Connection modal window where you have to adjust further settings.

8. Click TEST.

   If the connection is valid, the system returns a Connection Verified message.

9. Click CONNECT.

Results

You have set up a connection to a running data warehouse.

Related Information

RBAC permissions

Setting role privileges

Promoting a user to administrator

Managing data connections

You can change the properties of existing data connections and you can also delete connections that are not associated with any dataset.

Editing a data connection

Learn how to edit a data connection in Cloudera Data Visualization.

About this task

The following steps demonstrate how to edit existing data connections. The example shows changing an Impala connection to a Hive connection.

Procedure

1. On the main navigation bar, click DATA.

   The Data view appears, open on the Datasets tab.

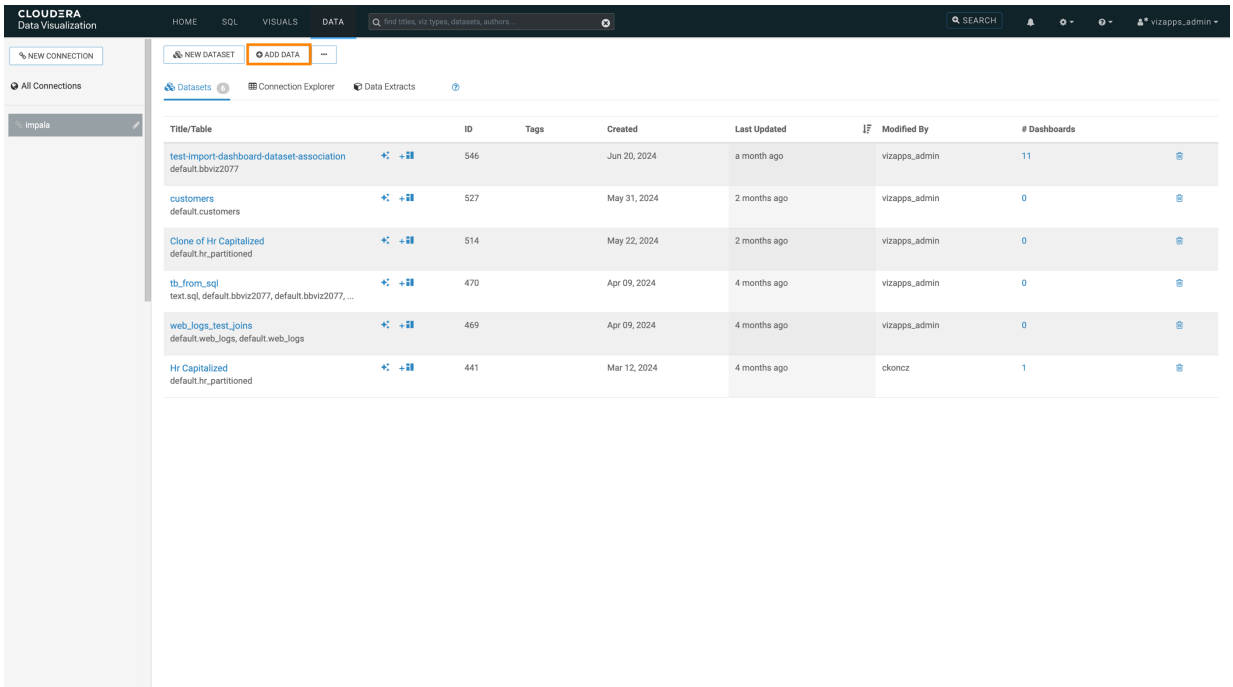

2. In the side bar, click the Edit button to the right of the connection you want to change.

   The Edit Data Connection modal window appears.

3. Edit the connnection details according to the connection type change you want to implement.

   In this example, an Impala connection is changed to a Hive connection.

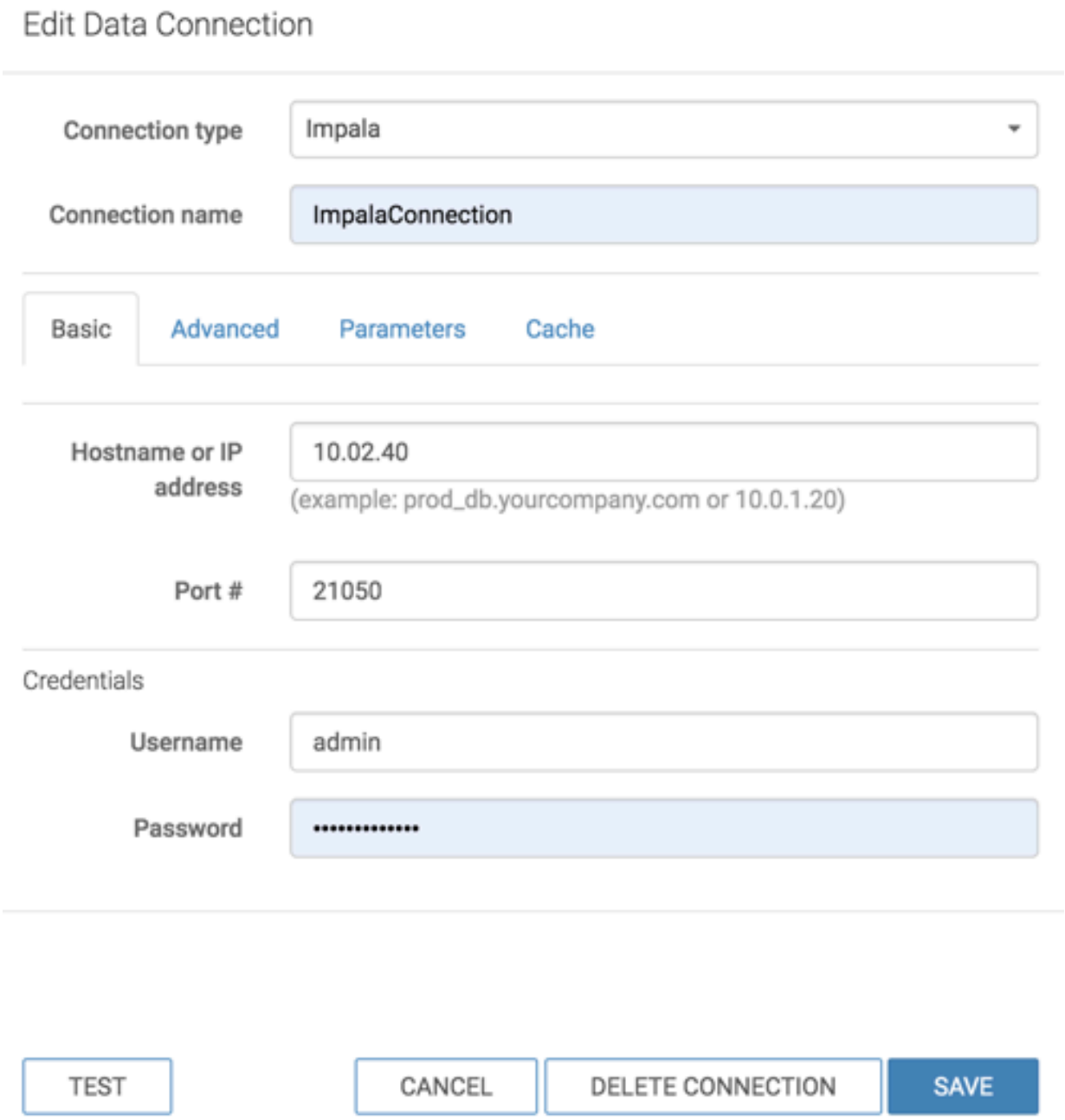

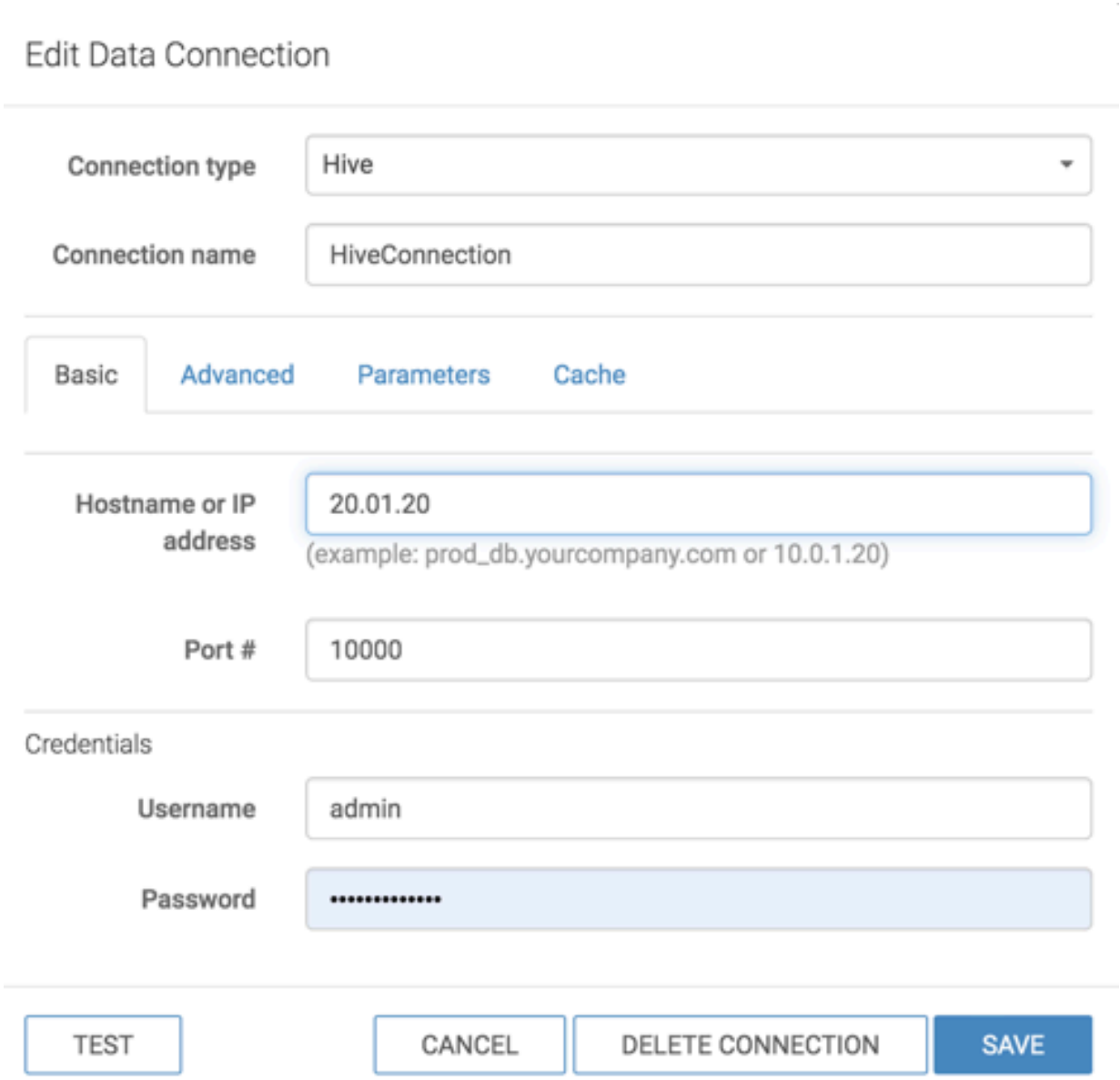

4. At the bottom of the modal, click TEST.
5. If the connection is verified, click SAVE.

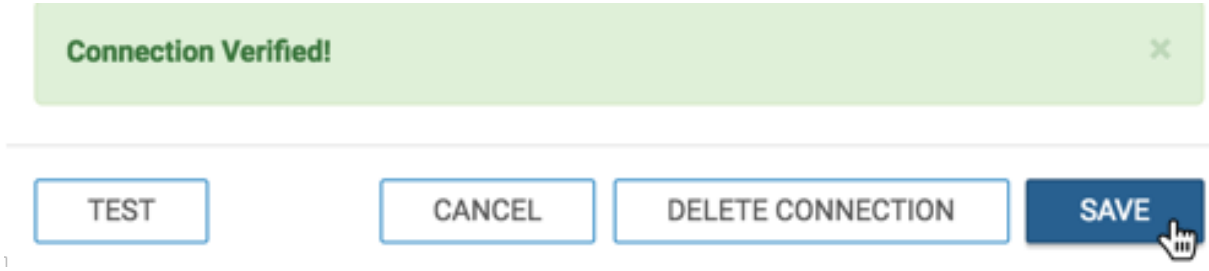

Results

After the operation succeeds, the name of the new type of connection appears on the side navigation bar.

Setting concurrency for a data connection

You can improve resource management in Cloudera Data Visualization by restricting the number of simultaneous connections on specific datasets.

About this task

Note: This setting is only applicable to Impala, Hive, and SQLite connections.

Procedure

1. Open the Edit Data Connection modal.

   For more information, see *Editing a data connection*.
2. Click the Data tab.
3. In the Concurrency field, provide a number to set the maximum limit for simultaneous requests.

   Note: The default maximum concurrency setting per connection is 100. You can change it to a different value in Site Settings Advanced Settings . If you change the default value, ensure that the new maximum concurrency aligns with the requirements of your data connection.

   ```
   DATACONNECTION_CONCURRENCY_DEFAULT = 100
   DATACONNECTION_CONCURRENCY_MAX = 100
   ```
4. In the Concurrency Per User field, provide a number to limit the maximum number of simultaneous requests a user can send.

   Note: You can use this setting to prevent a single user from blocking access for a connection. The default concurrency setting per user is 5, with a default maximum concurrency setting per user at 100. You can adjust these values in Site Settings Advanced Settings .

   ```
   DATACONNECTION_CONCURRENCY_USER_DEFAULT = 5
   DATACONNECTION_CONCURRENCY_USER_MAX = 100
   ```

5. Click SAVE to apply the changes.

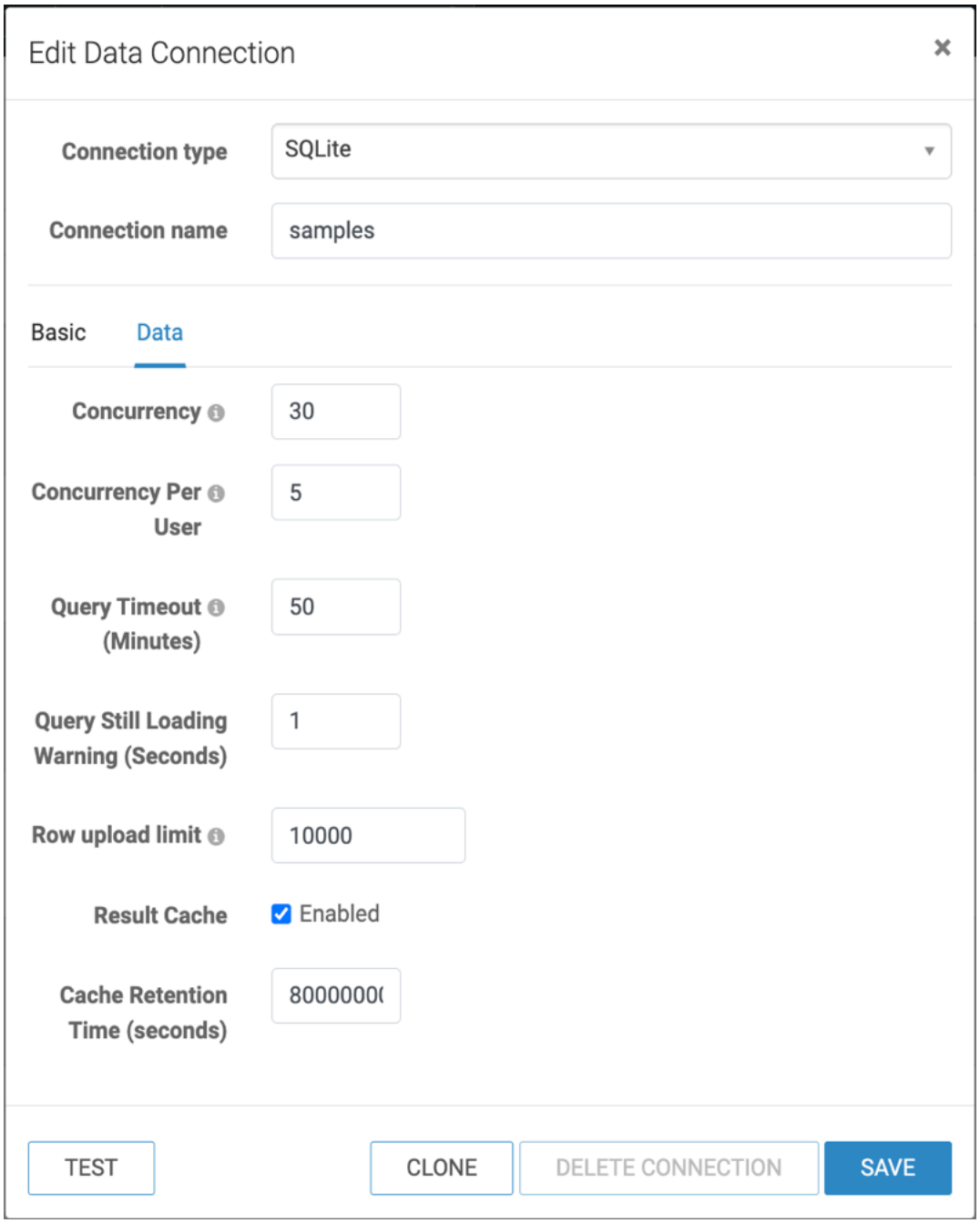

Related Information

Editing a data connection

Setting a warning if the query loads for too long

A data connection admin user can set the time limit for every connection before a warning is shown when data is loaded through a query.

About this task

For more information on creating or editing data connections, see *Connection to a data source* and *Editing a data connection*.

Procedure

1. Select the Data tab in the Data Connection modal.
2. Enter the number of seconds you want to pass before the warning appears in Query Still Loading Warning (Seconds).

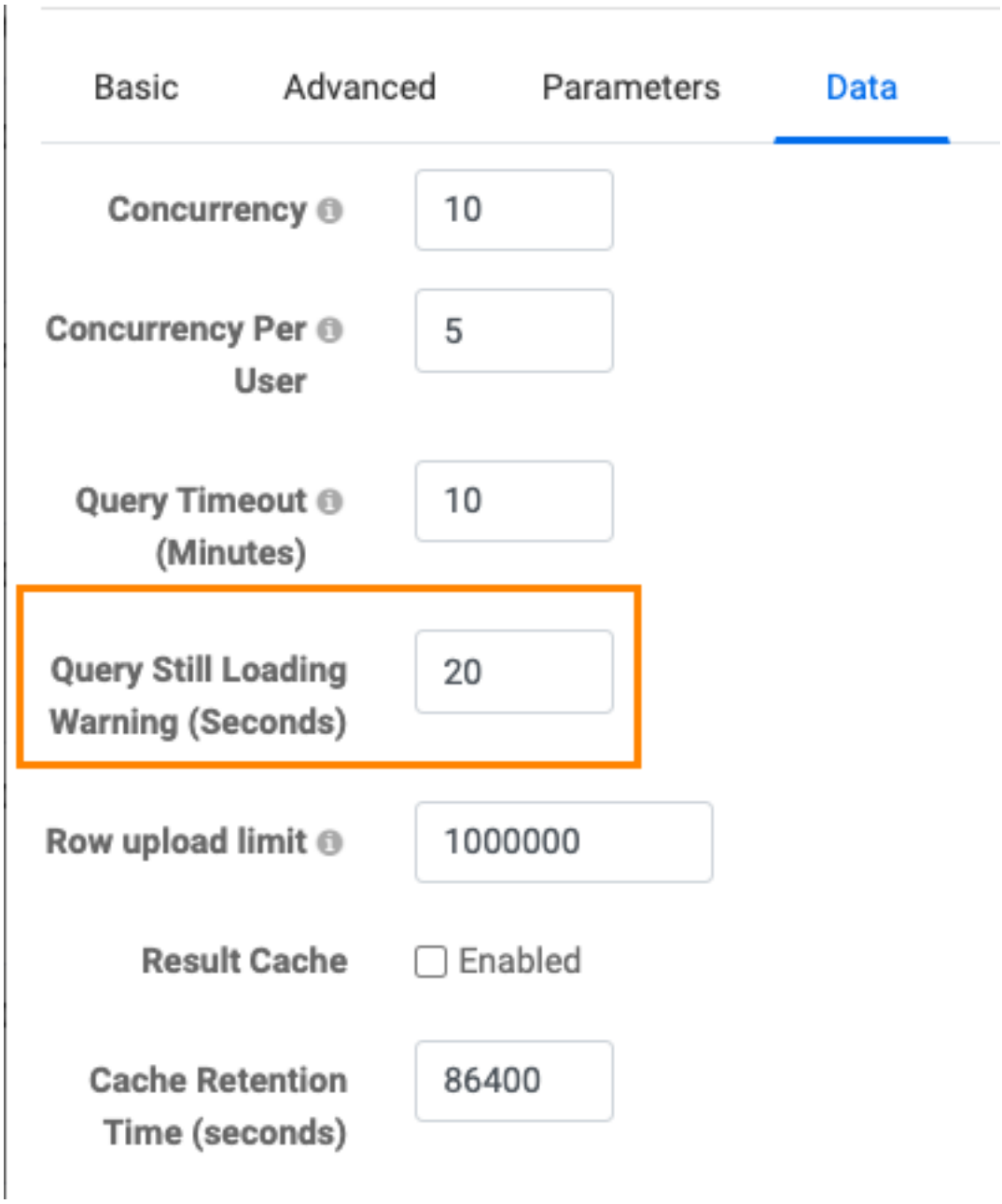

It is also possible to have this message appear when viewing a dashboard, see *Dashboards*.

Related Information

Dashboards
Connecting to a data source in Cloudera Data Visualization
Editing a data connection

Deleting a data connection

Learn how you can remove an existing data connection in Cloudera Data Visualization.

About this task

Tip: You can only delete connections that are not associated with any datasets. To learn how to delete datasets, see *Deleting datasets* .

Procedure

1. On the main navigation bar, click DATA.

   The Data view appears, open on the Datasets tab.

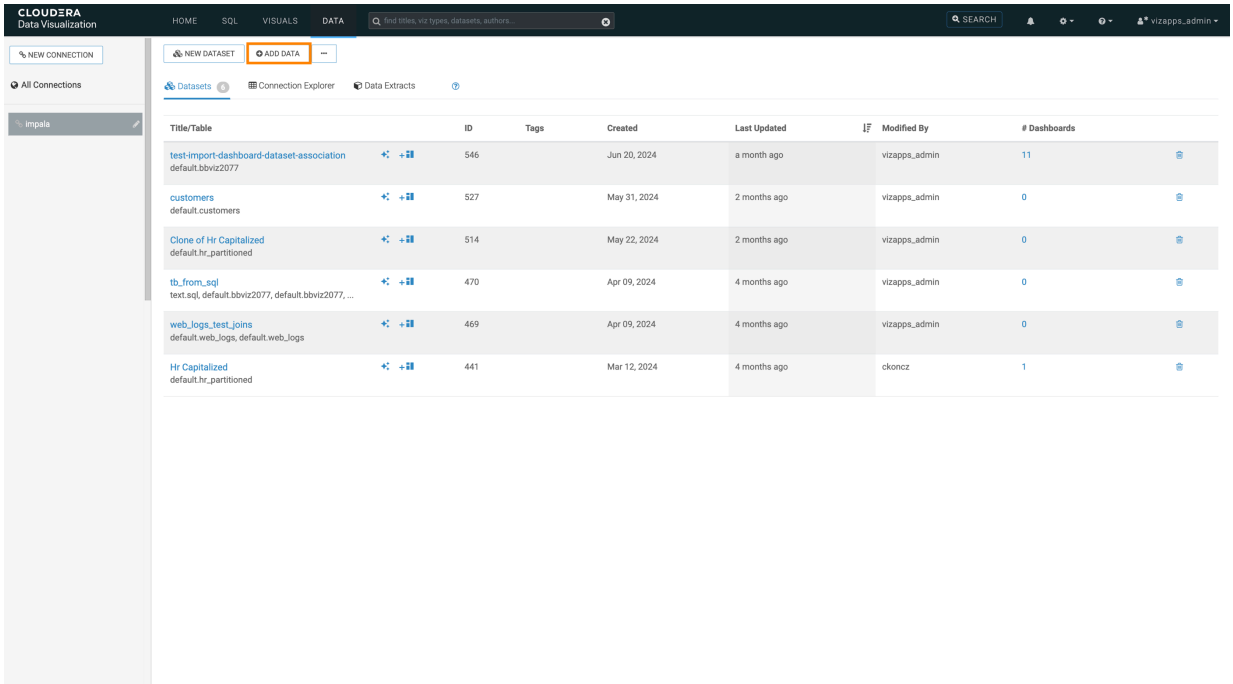

2. In the side bar, click the Edit Connection (pencil) button to the right of the connection you want to delete.

   The Edit Data Connection modal window appears.

3. At the bottom of the Edit Data Connection modal window, click DELETE CONNECTION.

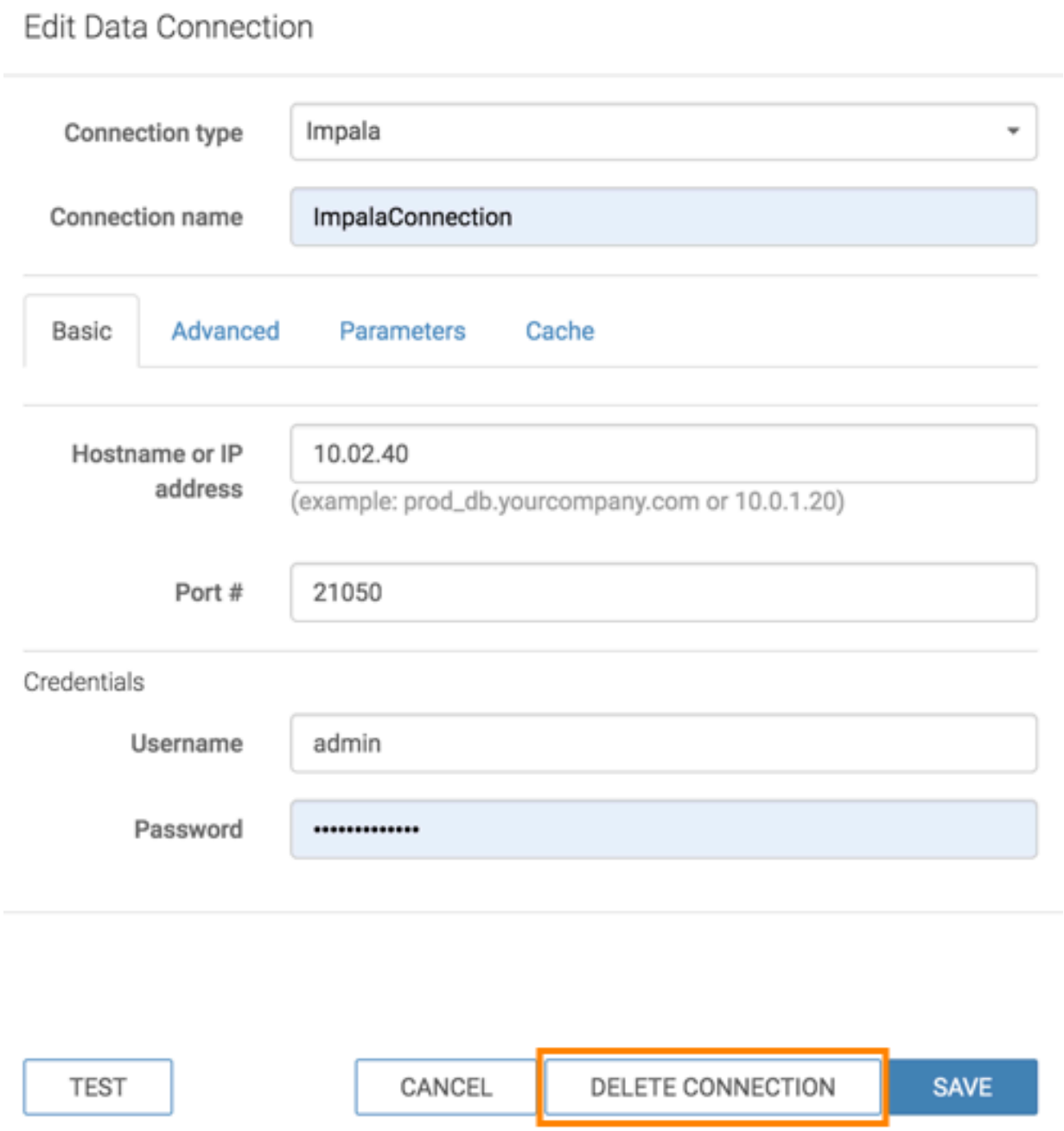

Results

After the operation succeeds, the connection is deleted and its name no longer appears on the side navigation bar.

Related Information

Deleting datasets

Using the Connection Explorer

Cloudera Data Visualization enables you to view existing data connections and all data tables accessible through them. In the Connection Explorer interface, you can create new connections to data sources, preview that data, create

new datasets, navigate to these datasets, import supplemental data, and locate existing dashboards and visuals based on specific datasets.

Discovering the Connection Explorer interface

Learn how you can navigate to the Connection Explorer interface and use it to connect to data in Cloudera Data Visualization.

Procedure

1. On the main navigation bar, click DATA.

   The Data view appears, open on the Datasets tab. The Datasets tab lists all existing datasets on the connection.

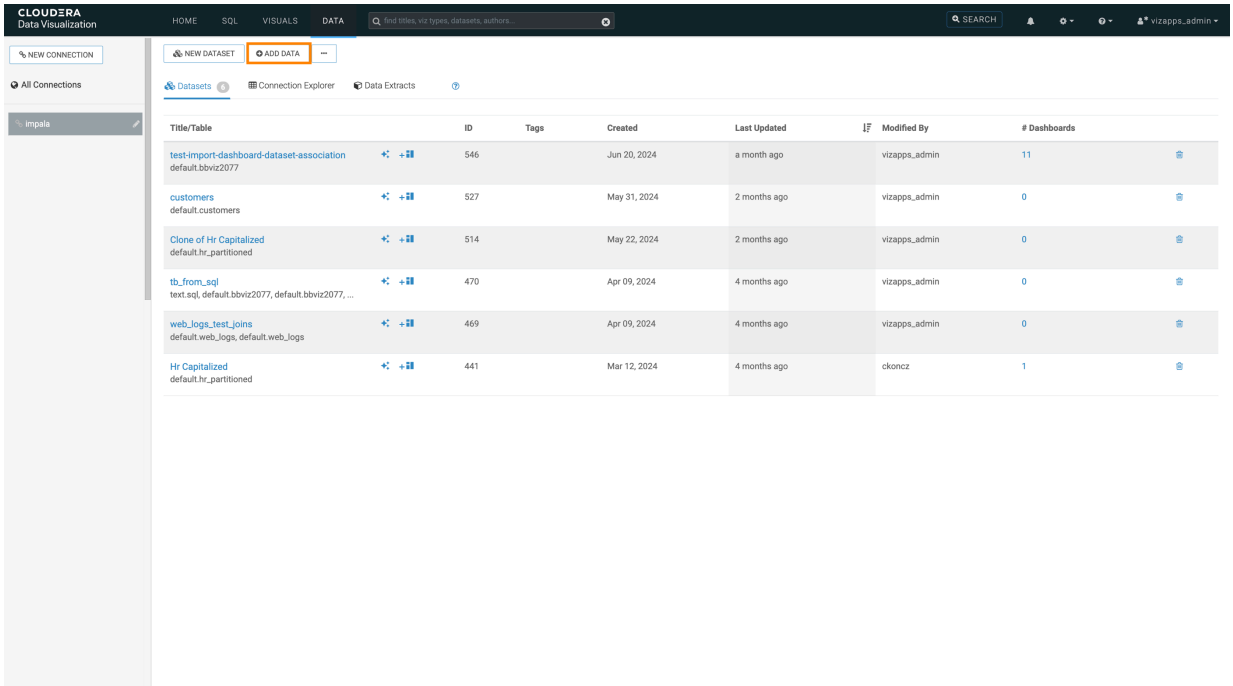

2. In the main area, click the Connection Explorer tab.

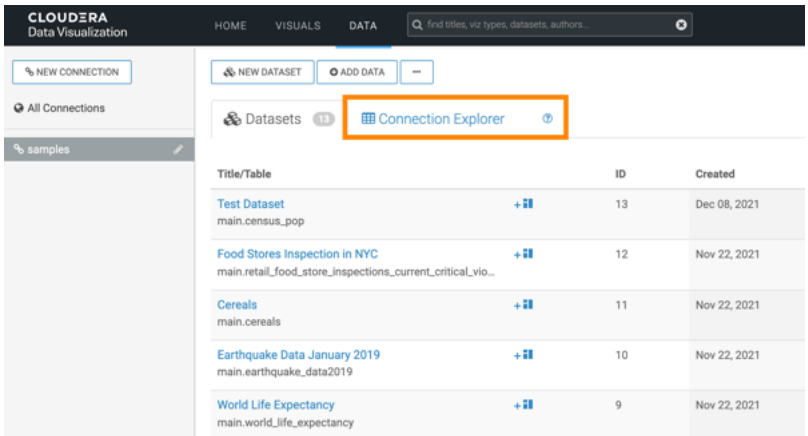

   The Connection Explorer interface appears, where you can explore the databases and tables available on the connection and you can also manage all functions related to the data on the connection. You can click a

connection in left navigation that you want to explore and select a database. You can also select a specific table from that database, and explore its details.

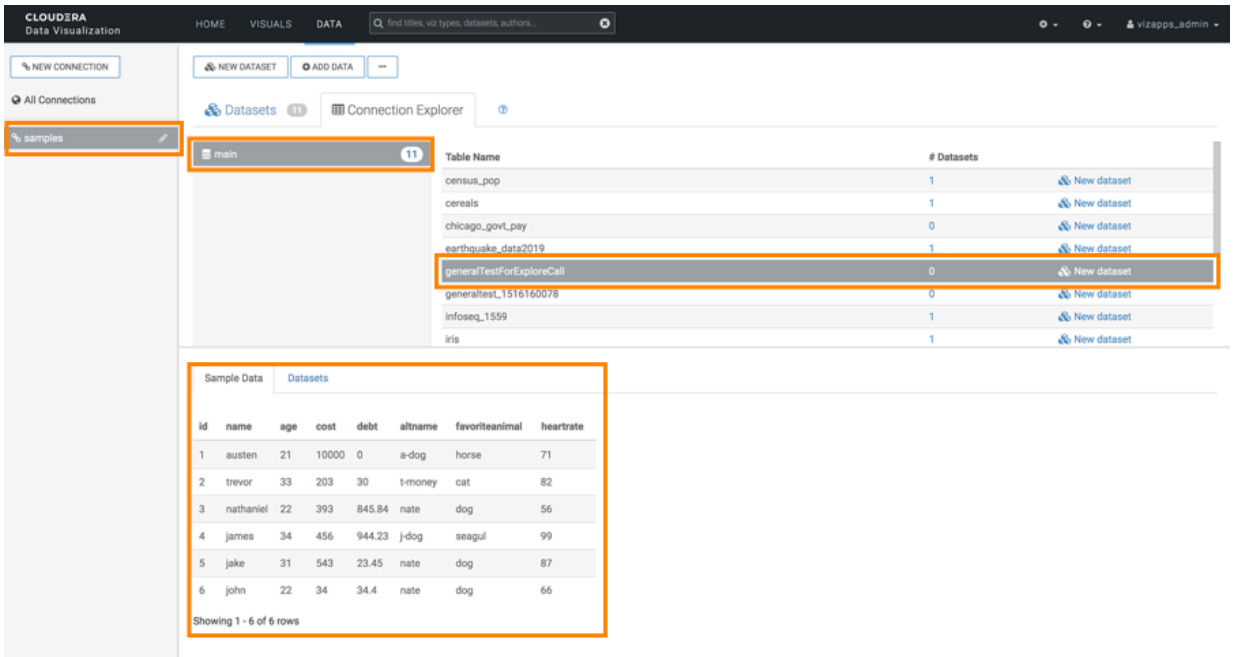

The Connection Explorer interface contains the following items:

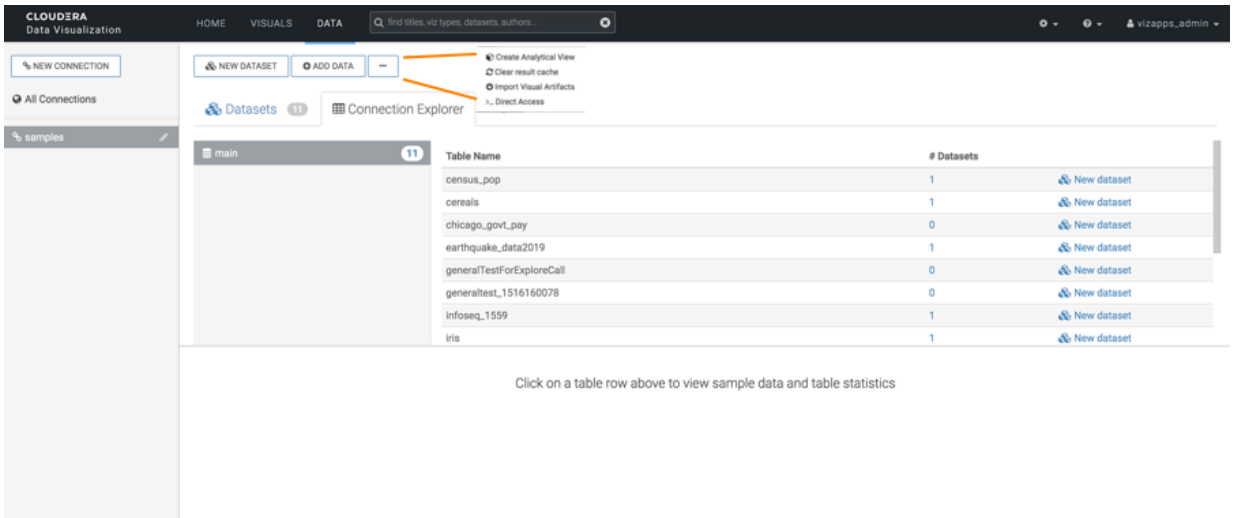

- New Connection is for connecting to any source of data.
- New Dataset is for creating datasets, which are necessary for developing dashboards and visuals. You can also start a new dataset from a specified table.
- For SQLite connections, the Add Data option enables you to introduce data that enriches your datasets from outside sources.
- The Supplemental menu, under the (ellipsis) icon, opens new options.

  a. For Impala connections, clicking the Clear result cache option under the supplemental menu reloads the full table definition.

  b. Import Visual Artifacts option under the supplemental menu enables you to restore or import visual artifacts from a backup JSON file.

  c. Direct Access enables you to access data directly by running SQL queries. You can build datasets from specific SQL queries, as opposed to starting with an existing table.

- The databases area of the screen shows all databases that you can access through the current connection. In our example, there is one called main (selected). Selecting a database shows its tables.
- In the list of tables, the # Datasets column lists the number of datasets that use the particular table as their initial definition.
- New Dataset is for creating a dataset on a specific table.

Previewing data table details

Learn how you can preview table details directly in the Connection Explorer interface.

To see more information about data tables in the Connection Explorer, click the row of a table. When you click a row, two tabs, Sample Data and Datasets appear below the list of tables.

Sample data

When you click a table, you can preview the table data in the Sample Data view.

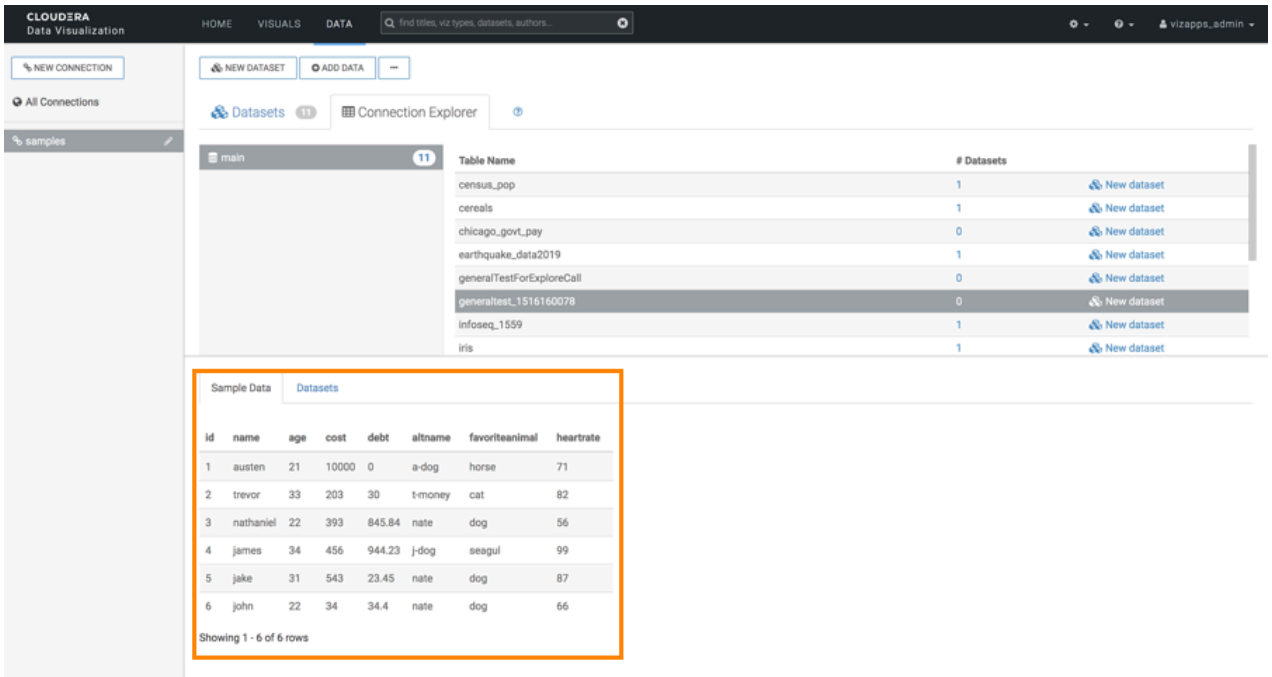

Datasets

When you click a table, you can check the following data in the Datasets view:

- Title/Table
- Created date
- Last Updated date
- Modified by username
- # Visuals for a link to the dashboards and visuals based on the dataset.

You can also perform the following actions:

- Navigate directly to the dataset interface, where you can rename the dataset, modify fields and other parameters, create joins, and so on.
- Start a new dashboard based on the dataset.
- Order datasets based on any of the table columns.
- Delete datasets.

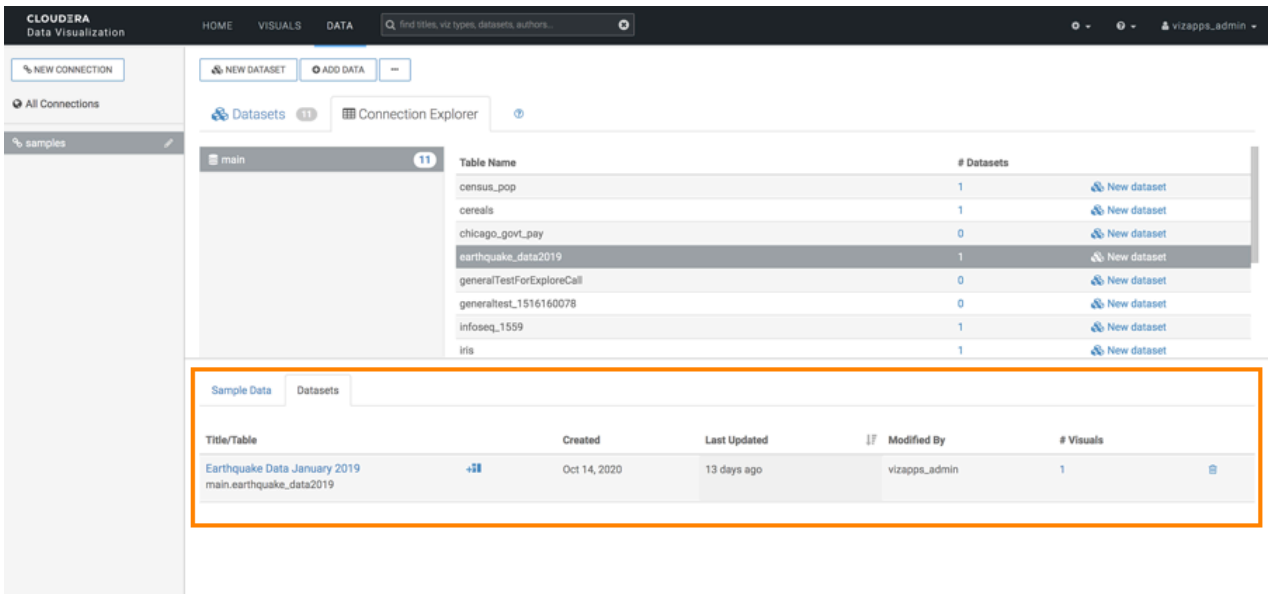

Using the Direct Access interface

The Direct Access interface of Cloudera Data Visualization enables you to run SQL queries on data connections directly on the DATA page. You can quickly examine the structure of tables that are available on the connection, build a query using standard SQL syntax, preview its results, and then create a dataset on the query. You can also download the data, if needed.

Running a SQL query in Direct Access

Learn how you can run a SQL query in the Direct Access interface of Cloudera Data Visualization.

Procedure

1. On the main navigation bar, click SQL.

   Alternatively, you can click DATA ••• >_ Direct Access .
2. Select the database in the left-side panel.

3. Add your SQL data query to the Enter SQL below code-enabled text box.

   The Autocomplete option is on by default. It validates the syntax of the SQL query you enter.

   If you want to limit the number of, you have two options:

   a. You can add a limit clause in the SQL query syntax.
   b. You can mark the Add in a "LIMIT 100" clause to any SQL select query that does not have a limit clause option. This limitation is on by default. If you set a record limit in the SQL query, it will override this default option, even if it is checked.

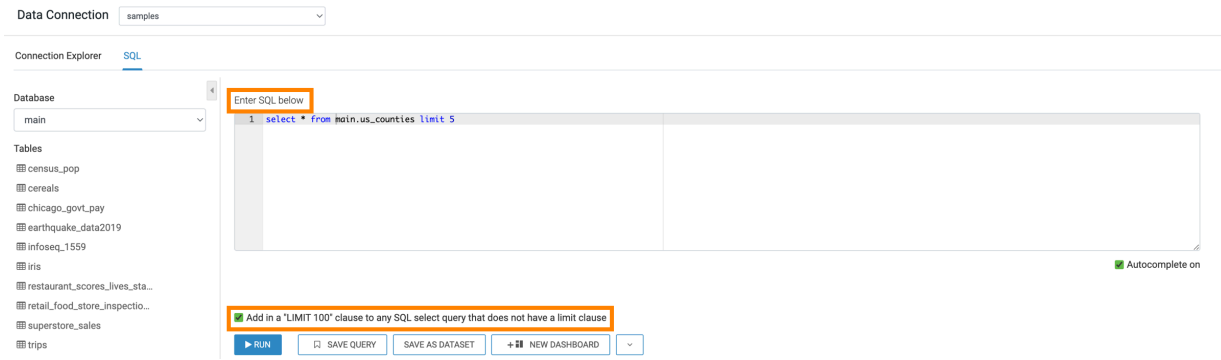

4. Click RUN to initiate the SQL query.

   After the query runs, the results tabe shows the query results.

   In this example, the following query has been run: select * from main.us_counties limit 5

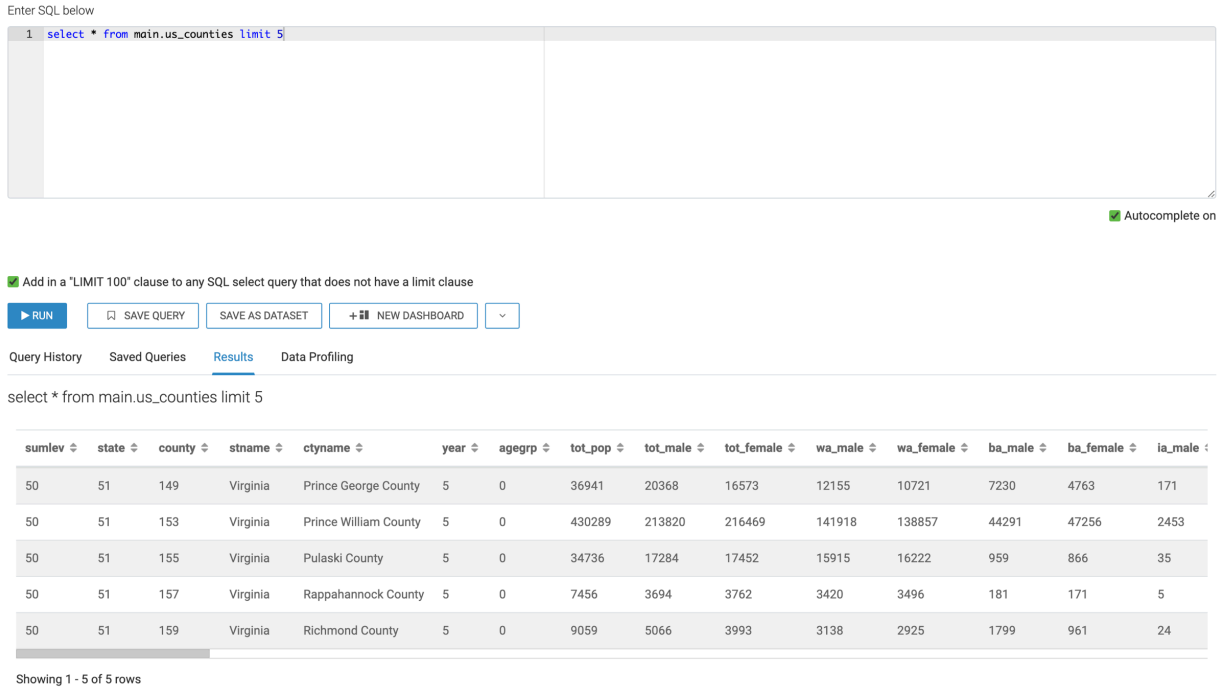

sumlev	state	county	stname	ctyname	year	agegrp	tot_pop	tot_male	tot_female	wa_male	wa_female	ba_male	ba_female	ia_male
50	51	149	Virginia	Prince George County	5	0	36941	20368	16573	12155	10721	7230	4763	171
50	51	153	Virginia	Prince William County	5	0	430289	213820	216469	141918	138857	44291	47256	2453
50	51	155	Virginia	Pulaski County	5	0	34736	17284	17452	15915	16222	959	866	35
50	51	157	Virginia	Rappahannock County	5	0	7456	3694	3762	3420	3496	181	171	5
50	51	159	Virginia	Richmond County	5	0	9059	5066	3993	3138	2925	1799	961	24

Note: If there is a error in the query the line number with the error will be displayed if the database returns it.

For more information about the content available on the other tabs, see *Cloudera Data Visualization, SQL interface*.

Related Information

Cloudera Data Visualization, SQL interface

Downloading the results of a Direct Access query

After obtaining query results in the Direct Access interface, you can download the records in CSV format.

You can download the results in CSV format, by clicking DOWNLOAD CSV. The system saves the CSV file to your default download directory.

If you only want to download a subset of the query results, you can specify the Number of rows in CSV. This will limit the number of records in the CSV file.

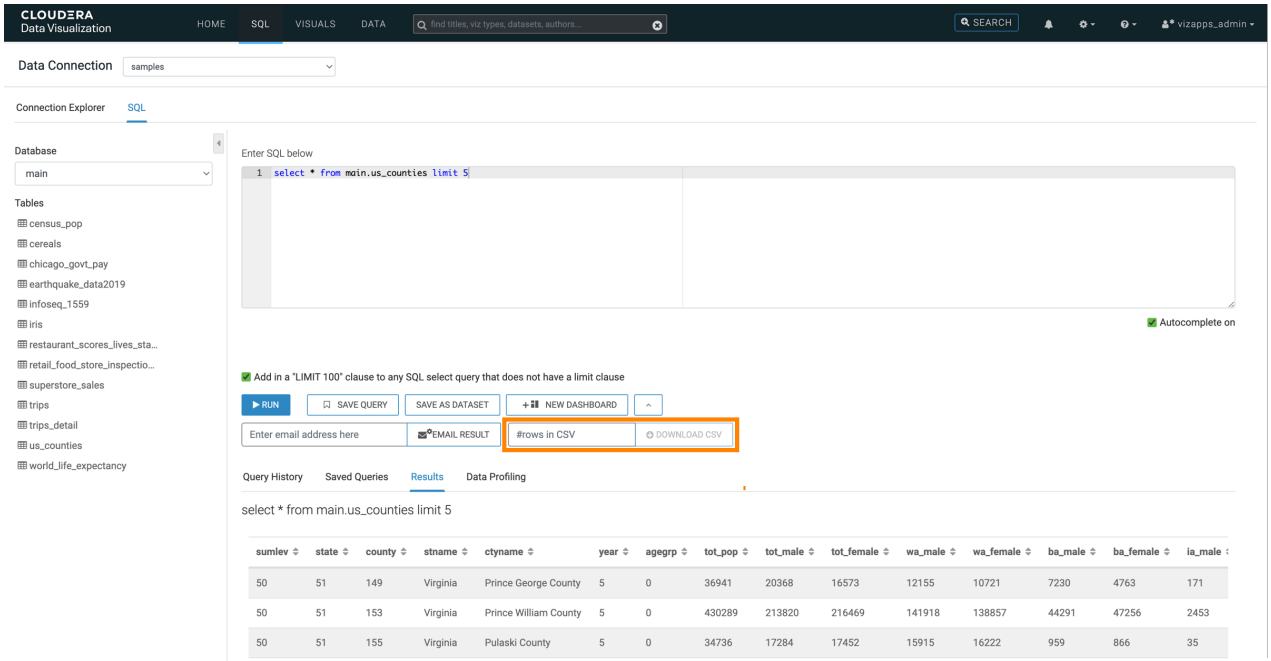

The system saves the CSV file to your default download directory.

Adding data

Experience the flexibility of importing and integrating data to enhance your data visualization projects. In Cloudera Data Visualization, you have the capability to add data from flat files. The imported data is processed as a new table and it is integrated into an existing data repository. Subsequently, you can use this data as a supplementary information source for your existing datasets by establishing new table joins. You can also create new datasets that reference the imported data as their primary tables.

Data import is available for the following connections:

- Hive
- Impala
- MariaDB
- MySQL
- PostgreSQL
- Spark SQL

Note: Data import is not supported in Cloudera Data Warehouse as the connection is read-only.

- SQLite

  Note: This connection is not supported in Cloudera Data Warehouse.

- Snowflake [Technical Preview]

Cloudera Data Visualization supports two primary forms of data sources: CSV and URL.

You can use the Import Data functionality to enhance your existing datasets, incorporating this data by creating joins within the data model. The choice of creating a new dataset based on the imported data depends on your specific business requirements.

For importing data stored in Excel format, see *Preparing Excel files for data import*.

Importing data in CSV format

Learn how to add data to Cloudera Data Visualization from flat CSV files.

About this task

You can use comma-delimited files or files with other delimiting characters. If you want to import data stored in Excel format, see *Preparing Excel files for data import*.

This feature is available on the following connections:

Enabled by default:	Disabled by default:
• SQLite • Hive • Impala • Snowflake [Technical Preview] • Solr [Technical Preview] • Spark SQL	• DuckDB • MS SQL • Teradata

Note: To import data into Cloudera Data Visualization, ensure that you use a data connection with write permissions.

Procedure

1. On the main navigation bar, click DATA.

   The Data view appears, showing the Datasets tab.

2. Click ADD DATA.

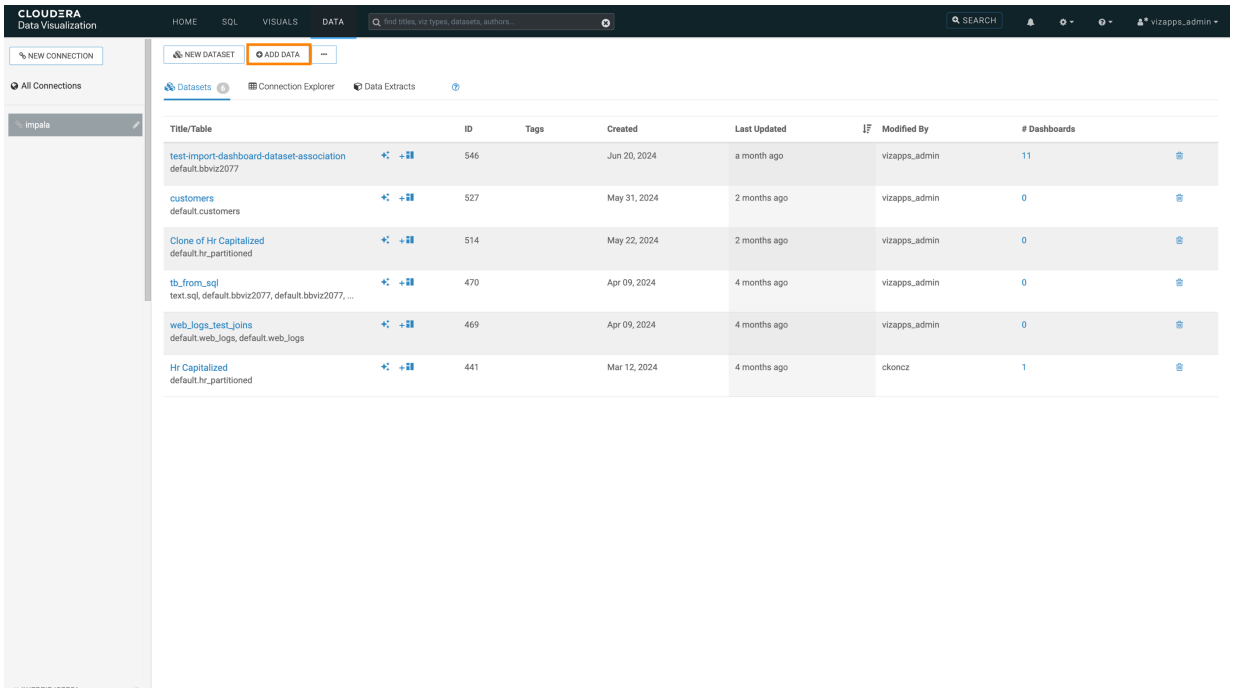

The Add Data modal window appears.

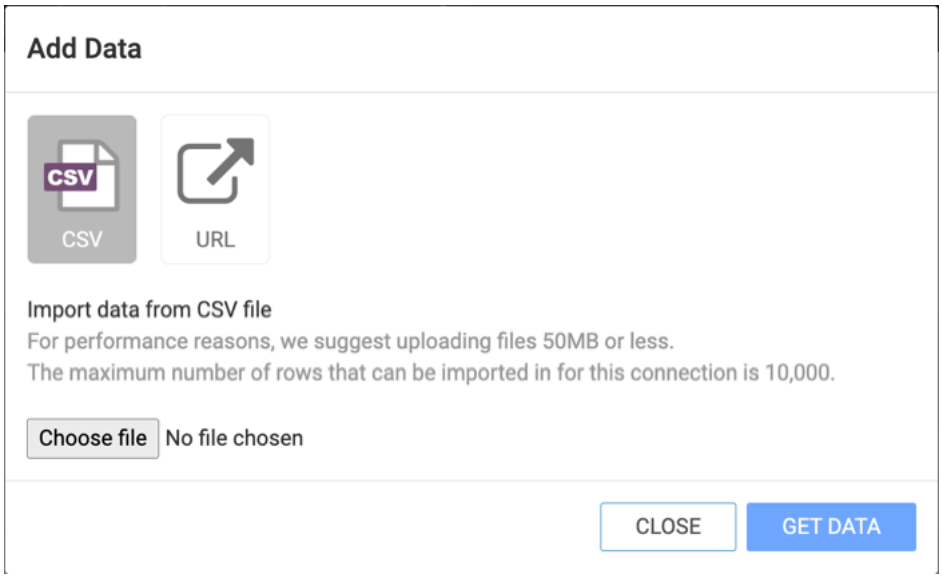

3. Add the CSV file that contains the data you want to import.
   a) Select the CSV option and click Choose file.
   b) Using your computer's file browser, locate and select the data file.
   c) Click GET DATA.

4. In the Imported File modal window, ensure that the data you are adding is configured correctly.
   a) Verify the name of the file.
   b) Under Database, select which database you want to add data to.
   c) Under Table Name, specify a table. The system usually assigns a numerical suffix, which you can remove.
   d) Under Upon Import, select whether you only want to import data or create a dataset, or create a dataset and a dashboard based on the data.
      - If you select Create Dataset, you are directed to the Detail page of the dataset you have created after the import is completed.
      - If you select Create Dataset and Dashboard, you are directed to the newly created dashboard when the import is completed.
   e) For Column Delimiter, define the delimiter used in the source file. It can be Comma, Tab, Space, Semicolon, Colon, Pipe, Control A, or Other.
   f) Under Locale Setting, you can select one of the common locales: United States, United Kingdom, Sweden, Norway. Or you can select Other to add a separate field next to the locale setting, where you can specify the supported locale. For more information, see *Changing the locale setting of an imported file*.
   g) Under Options, select the data clean-up tasks that Cloudera Data Visualization provides prior to data import. These include Create "EXTERNAL" table, File contains headers, Fill missing columns, Skip malformed rows, and Use '\' as escape character.

      Note: Cloudera Data Visualization can detect many items, for example the presence of header rows, automatically.

   h) In the Data Table preview, each column of the table appears with its data type and name.

      The data types available are Boolean, Integer, Real, String, and Timestamp.

      Cloudera Data Visualization detects the data type automatically, but you can modify them as needed (for example store numerical categories as strings).

      You can change the name of the table column before importing.
   i) Click APPLY CHANGES to save the adjustments you have made.

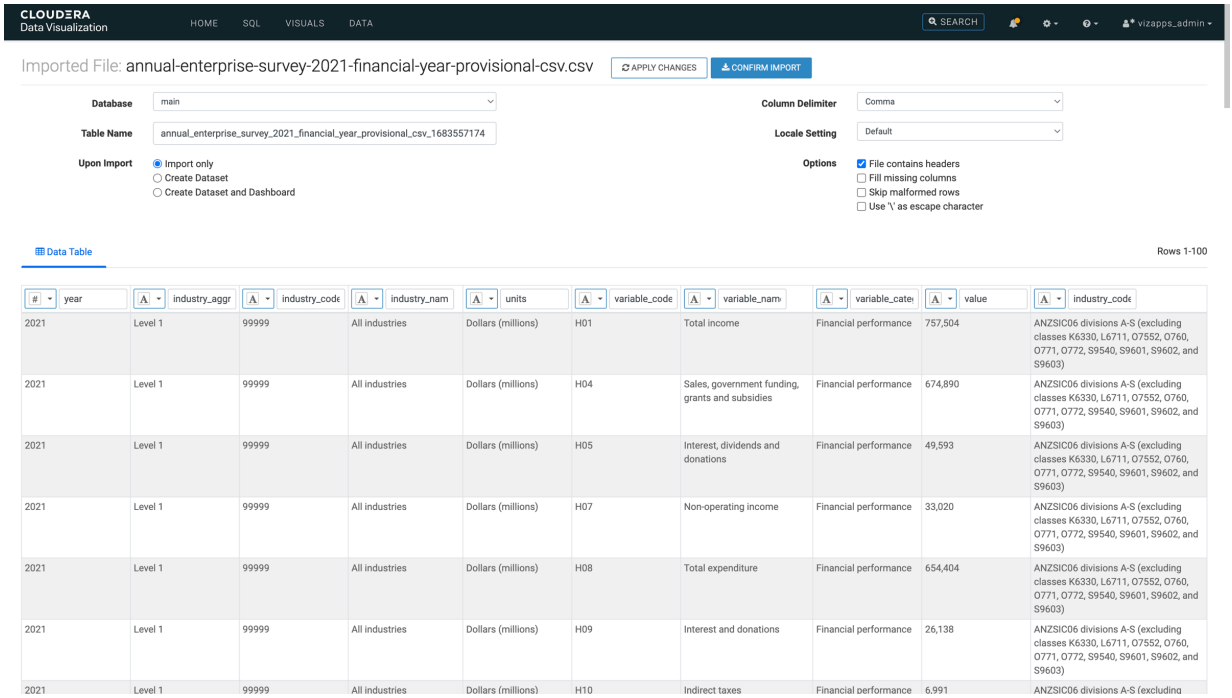

5. Click CONFIRM IMPORT.

   The Confirm File Import modal window appears, where you have to click CONFIRM IMPORT again to finalize the import.

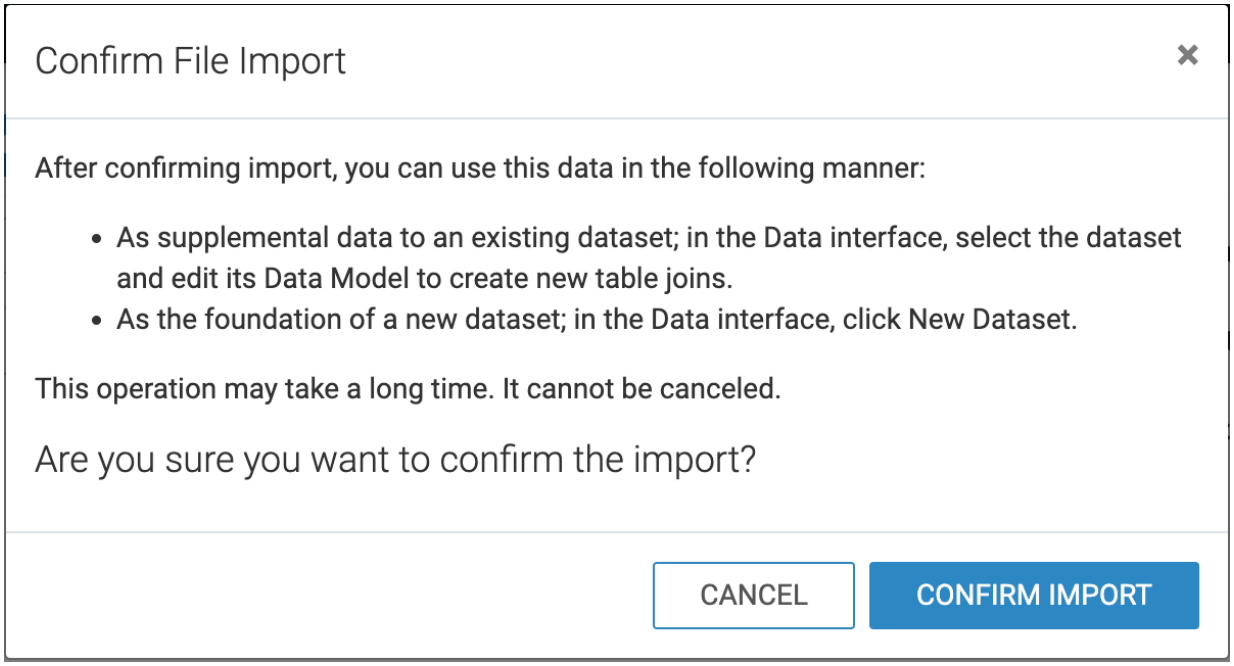

6. To verify that the data has been imported correctly into your system, check the Connection Explorer interface.

Related Information

Preparing Excel files for data import

Changing the locale setting of an imported file

Adding data through URL

Learn how to how to add a data file using a URL in Cloudera Data Visualization.

About this task

Data files are typically in JSON format. The files may be comma-delimited or they use other delimiting characters.

Note: This feature is available on the following connections:

Enabled by default:

- SQLite
- Hive
- Impala
- Snowflake
- Spark SQL

Disabled by default:

- DuckDB
- MS SQL
- Teradata

Procedure

1. On the main navigation bar, click DATA.

   The Data view appears, open on the Datasets tab.

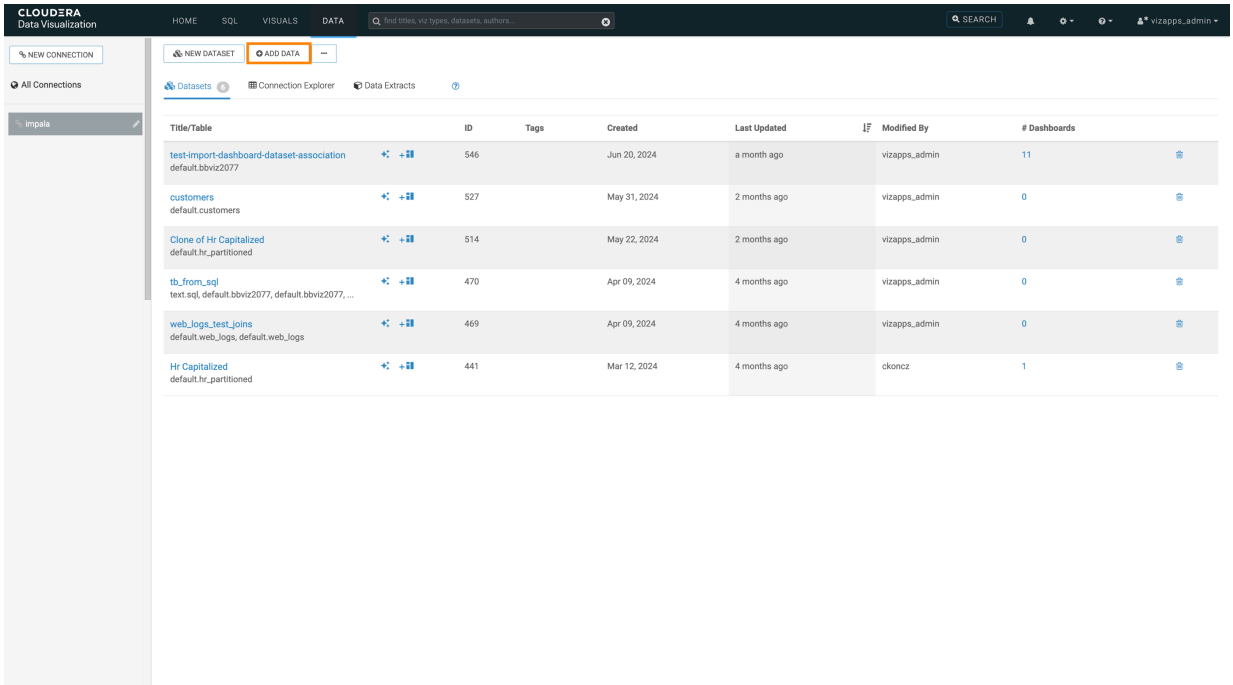

2. Click ADD DATA.

   The Add Data modal window appears.

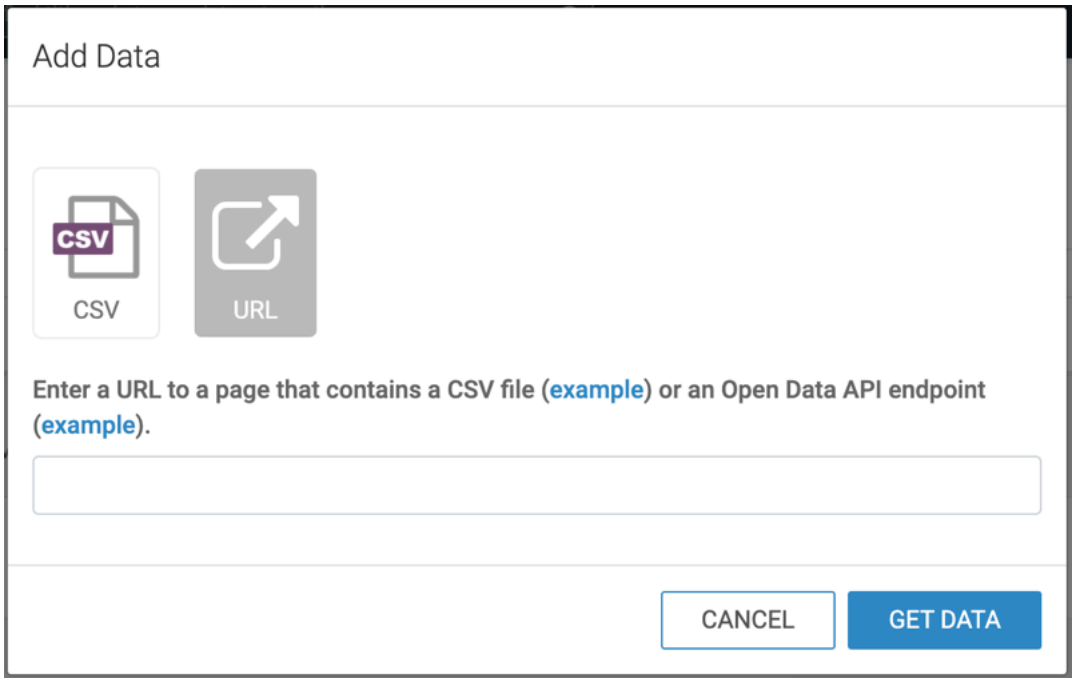

3. Select the URL option, and add the address of the file to the text window, and click GET DATA.

4. Ensure that your data is configured correctly before confirming the import:
   a. Under Database, specify documentation. Alternatively, select another database.
   b. Under Table Name, specify city_chicago_staff. The system typically assigns a numerical suffix that you can remove.
   c. Under Upon Import, select whether you only want to import the data or create a dataset and a dashboard based on the data.
      - If you select Create Dataset, you are taken to the Detail page of the dataset you have created when the import is completed.
      - If you select Create Dataset and Dashboard, you are taken to the newly created dashboard when the import is completed.
   d. For Column Delimiter, define the delimiter used in the source file. In this case, the column delimiter is Comma, but it can also be Tab, Space, Semicolon, Colon, Pipe, Control A, or Other. Make the necessary adjustments to this field.
   e. Under Locale Setting, you can select one of the common options: United States, United Kingdom, Sweden, Norway, or Other. Selecting Other adds a separate field next to the locale setting where you can specify the supported locale. For more information, see *Changing the locale setting of an imported file*.
   f. Under Options, select the data clean-up tasks that Cloudera Data Visualization provides prior to data import. These include File contains headers, Fill missing columns, Skip malformed rows, and Use '\' as escape character.

      Note: Cloudera Data Visualization can detect many items automatically, such as the presence of header rows.

   g. In the Data Table preview, each column of the table appears with its data type and name. Cloudera Data Visualization detects the data type automatically, but you can change it in this interface prior to import. For example, you may wish to store numerical categories as strings. The data types are Boolean, Integer, Real, String, and Timestamp.
   h. You can change the name of the table column before importing.

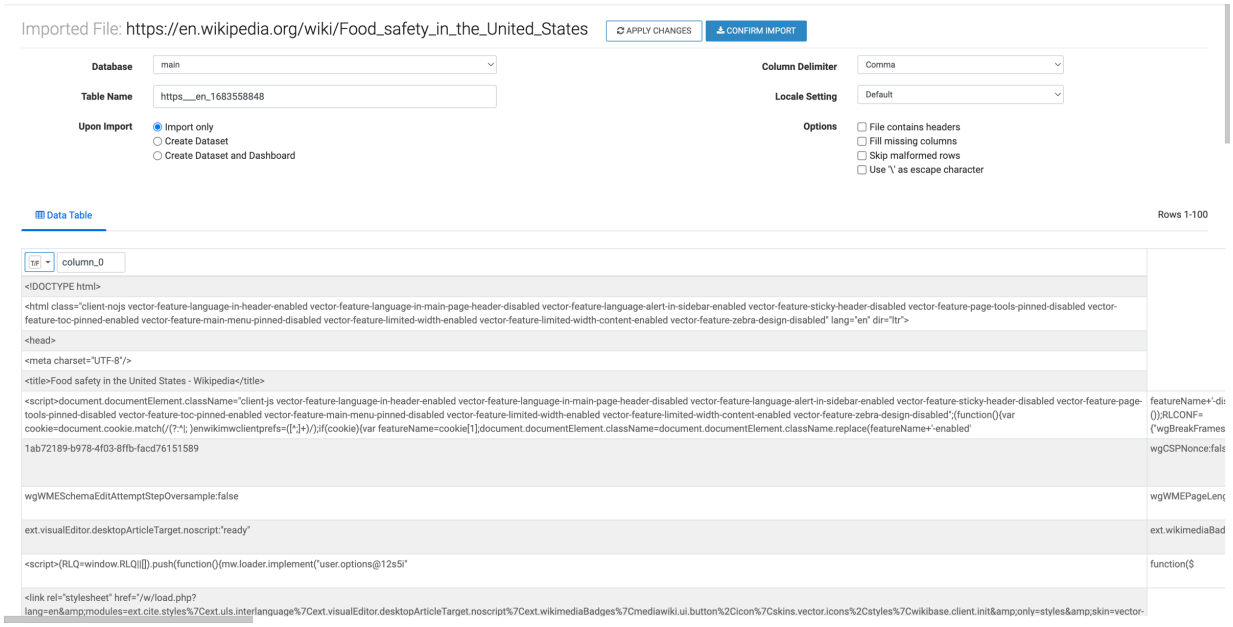

5. Click APPLY CHANGES to ensure that all adjustments you made are saved.

6. Click CONFIRM IMPORT.

   The Confirm File Import modal window appears, where you have to click CONFIRM IMPORT again.

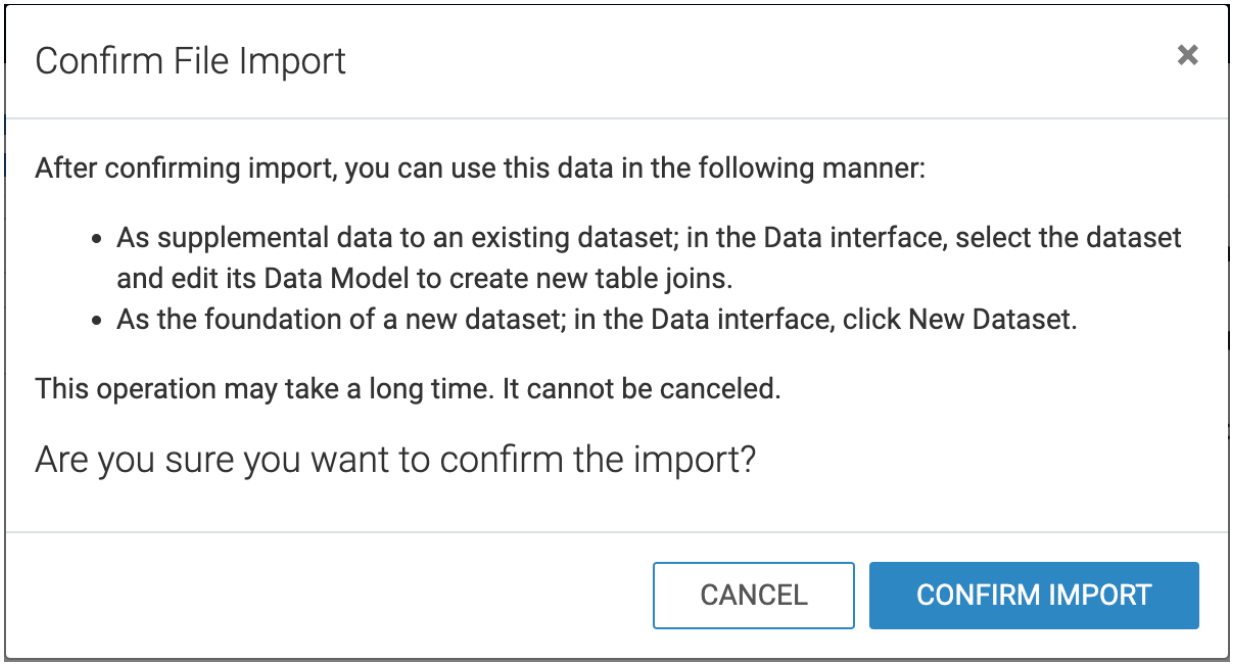

7. To verify that the data has been imported correctly into your system, check the Connection Explorer interface.

Related Information

Changing the locale setting of an imported file

Changing the locale setting of an imported file

When importing a data file, you can specify the localization for the data. This will specifically handle differences in specifying floating-point number.

Under Locale Setting, which is the default for your system, select one of the common options: United States, United Kingdom, Sweden, Norway, or Other.

Selecting Other brings adds a new text entry box. Clicking on the information icon opens a separate menu, where you can select another supported locale. In our case, we selected Russian, ru_RU.

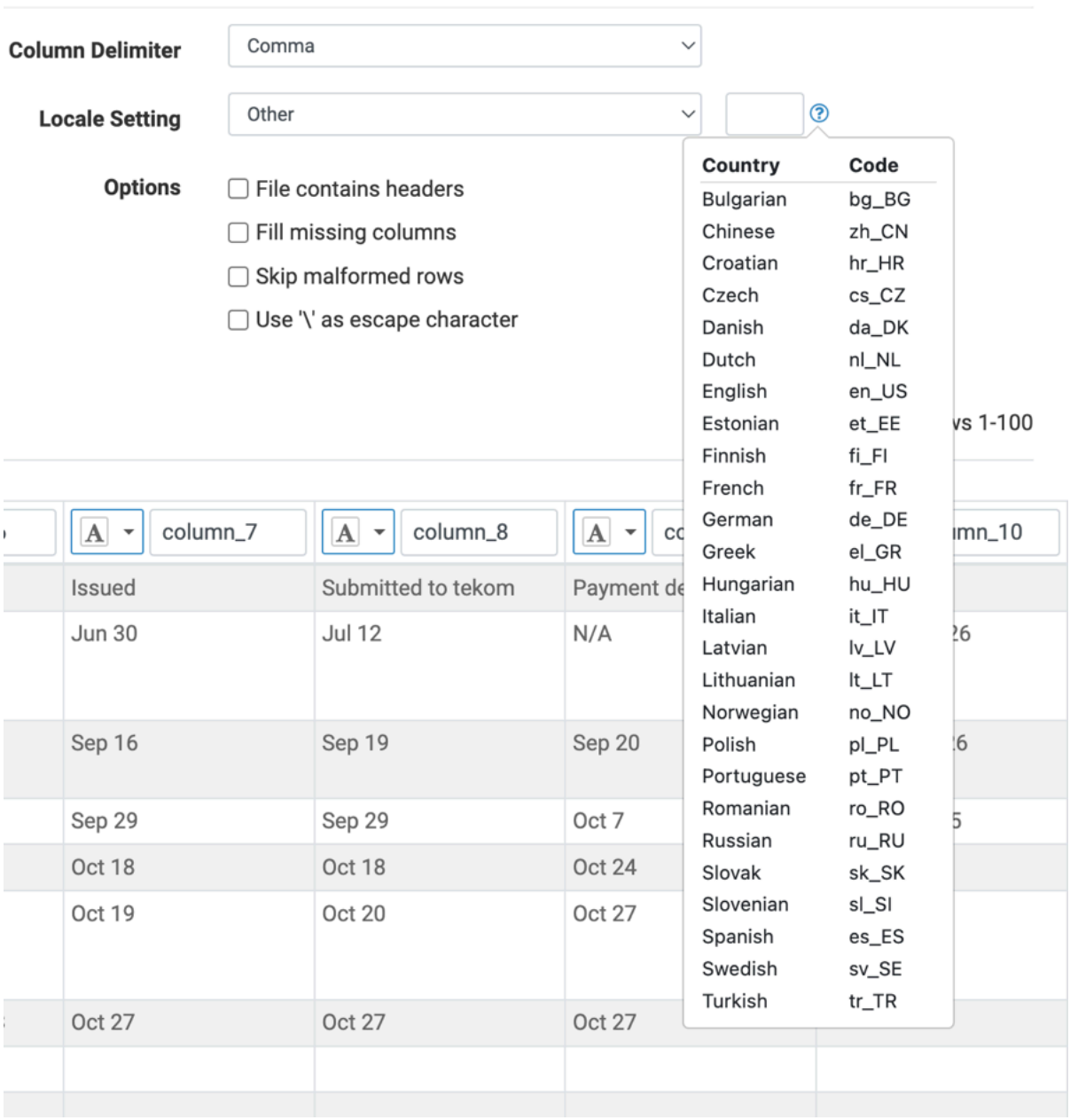

Preparing Excel files for data import

Learn how you can import data from Microsoft Excel or other similar formats into Cloudera Data Visualization.

About this task

You must save the files in character-delimited format.

Procedure

1. In Microsoft Excel, open the file that contains the data you are importing.
2. Click Save as, then select Comma Separated Values (.csv).
3. Choose a file name, such as MyCSVdata, and click Save.

Cloudera Data Visualization 7.2.9

Dashboards

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What is a dashboard? **4**

Dashboard Designer **4**

What is a dashboard?

A Cloudera Data Visualization dashboard is a collection of multiple visuals with the option to apply filters to them. You can use dashboards to consolidate related visualizations. For example, you can display and link visuals that are based on different datasets across different connections while offering run-time filtering on all referenced information.

Once you have created a dashboard, you can add visuals and filter controls, fine-tune settings and styles, and incorporate custom design elements to tailor the dashboard to your specific requirements. You can use sheets to organize the visuals on your dashboards.

Sheets

A sheet functions as the ‘canvas’ or a single page within a dashboard where you can design and display visualizations. Dashboards can contain one or more sheets, with each serving as a container for organizing and presenting specific sets of visual elements. Sheets allow users to organize information in a structured and meaningful way, providing a clear layout for data exploration and analysis.

Sheets give you the flexibility to consolidate all pertinent information about a dataset, a line of inquiry, or a business on a single screen while also providing quick access to related or supplemental visualizations on other sheets. The arrangement of sheets is entirely customizable to meet your unique needs and preferences.

By default, sheets are named Sheet 1, Sheet 2, and so on, but you can assign more descriptive names to your sheets to enhance clarity.

Each sheet is also assigned a unique identifier, visible in the sheet’s URL. This ID serves as a reference point, helping users to easily distinguish between different sheets within one dashboard.

Related Information

Creating a dashboard

Dashboard Designer

Creating a new sheet

Dashboard Designer

The Dashboard Designer of Cloudera Data Visualization provides an intuitive interface for building dashboards that help you to collect, organize, and display related visuals. You can add or link the visuals you have created based on datasets from the same connection or across different connections.

Using the Dashboard Designer interface, you can create visuals and arrange them on dashboards with ease.

The Dashboard Designer page has the following main parts:

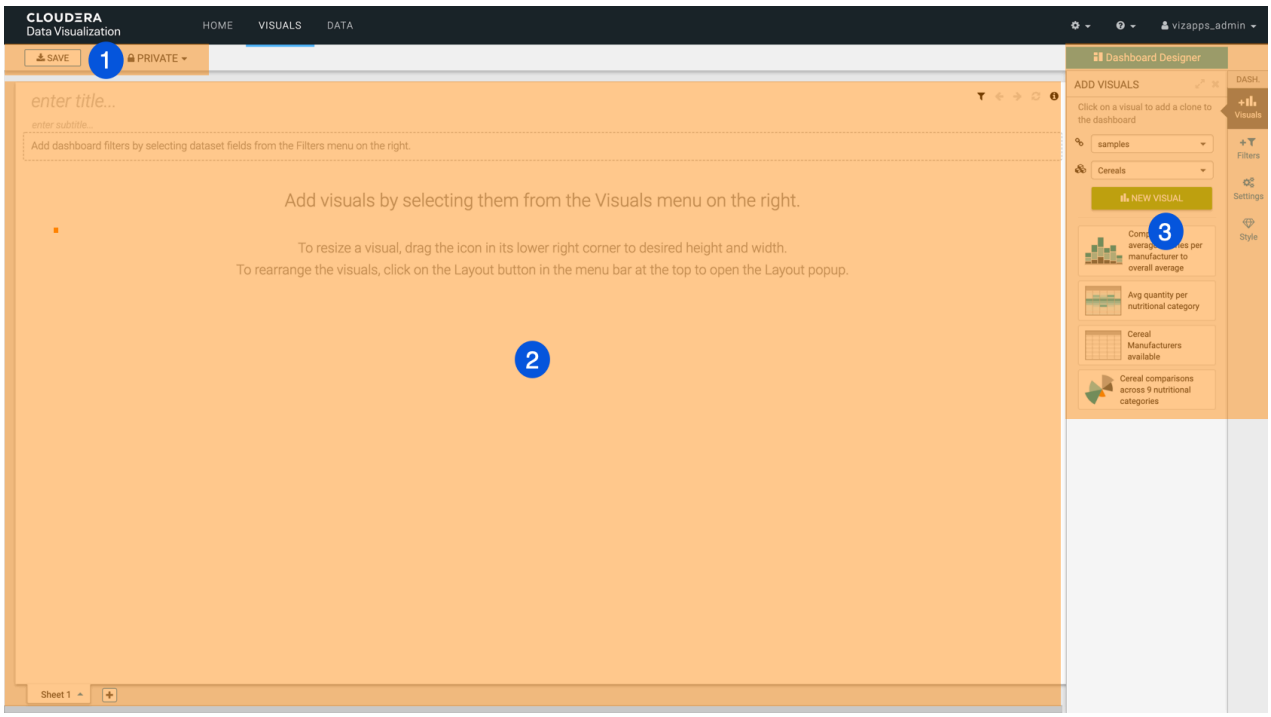

1. Dashboard Designer top menu bar
2. Dashboard preview area
3. Dashboard Designer side menu bar

Dashboard Designer top menu bar

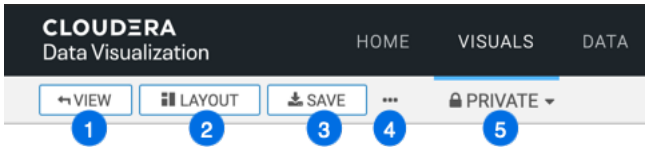

This menu bar houses various functions and options for designing and customizing your dashboards.

1. The VIEW button allows you to preview the dashboard at runtime. It becomes accessible on the menu bar after saving your dashboard. When you open an existing dashboard, it is initially displayed in View mode. To make edits or add new visuals, simply switch to Edit mode by clicking the EDIT button. You can switch back to View mode by clicking the VIEW button.
2. The LAYOUT button allows you to modify the layout of the dashboard. It appears on the menu bar once your dashboard has been saved. For more information on adjusting the arrangement of visuals on a dashboard, see *Customizing dashboard layout*.
3. The SAVE button allows you to save all changes made to the dashboard.

4. The Supplemental menu offers access to additional functions:

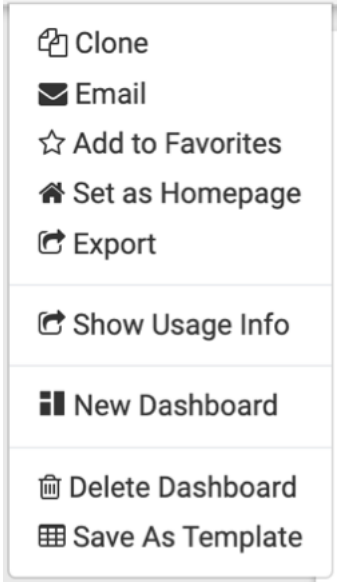

   - Clone lets you clone the current dashboard.
   - Email allows you to create an email job for the current sheet.
   - Add to Favorites lets you mark frequently-used dashboards as favorites for quick access. This option becomes available after you have initially saved your dashboard.
   - Set as Homepage lets you change your home screen to the current dashboard. This option is accessible once you have saved your dashboard.
   - Export allows you to export all dashboard elements. This feature is enabled after saving your dashboard.
   - Show Usage Info provides access to profile data, providing insights into usage. For more information, see *Showing usage information*.
   - New Dashboard lets you create a new dashboard.
   - Delete Dashboard lets you remove the current dashboard.
   - Save As Template allows you to use the current dashboard as a new temaple. Templates facilitate the rapid creation of dashboards from a dataset.
5. The workspace selector allows you to assign your dashboard to a different workspace. Dashboards can belong to public, private, or a custom workspace, but they can only exist in one workspace at any time.

Dashboard preview area

The dashboard preview area shows the contents of the dashboard, that is all filters and visuals you have added to it. You can also find here the title and subtitle of the dashboard. When you start building a new dashboard, this area is empty.

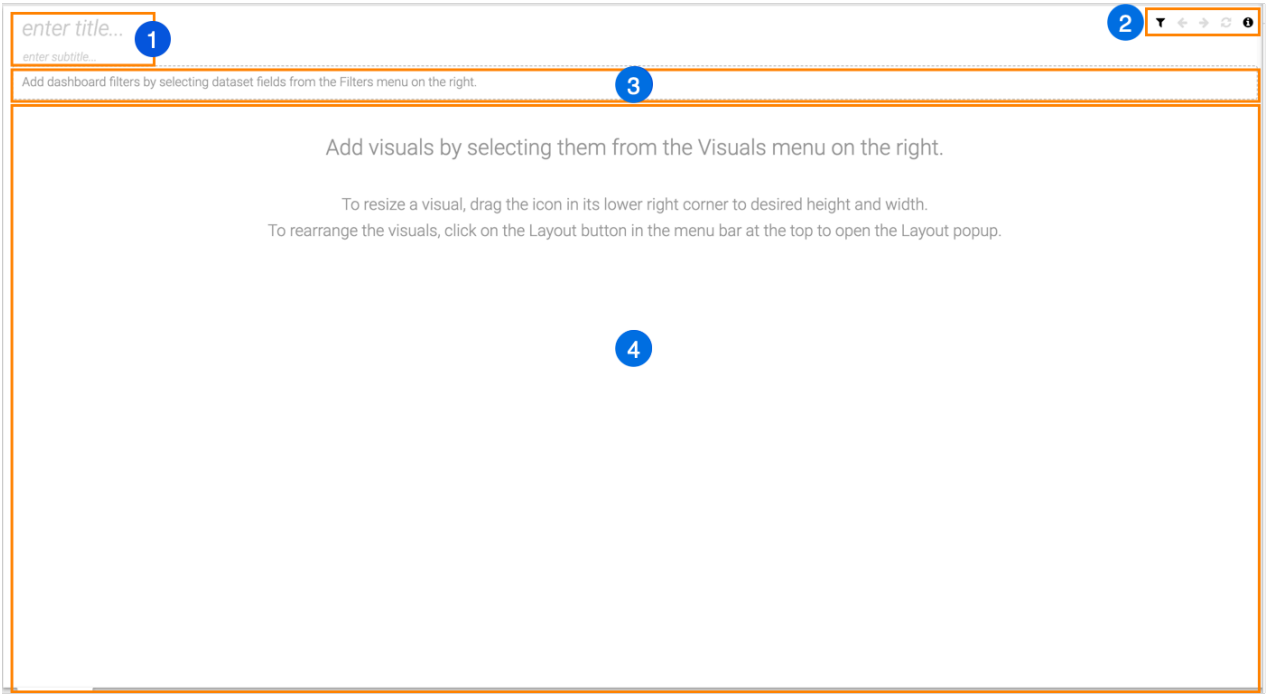

1. The title area contains two editable fields where you can add the title and subtitle of your choice. Both are empty when you create a new dashboard.
   - enter title...
   - enter subtitle...
2. Dashboard actions
   - Parameters
   - Back
   - Forward
   - Reset
   - Information

   When you save a new dashboard the supplemental menu icon is added to the dashbord actions offering the following options:

   - Email – You can use it to send the dashboard in an email. For more information, see *Emailing a dashboard*
   - Download as... – You can use it to download the dashboard as an image or in PDF format. For more information, see *Downloading a dashboard*
   - Get URL – You can use it to obtain the default URL of the dashboard or generate a short URL for it. For more information, see *Getting the dashboard URL*.
3. The filter area contains the filter controls that allow you to dynamically change the dashboard presentation at run time. Filters can be defined on any column of the dataset by pulling the appropriate field from the Filters menu.
4. The visuals area contains the visuals of the dashboard. It is empty when you create a new dashboard. To add visuals, use the Visuals menu.

Dashboard Designer side menu bar

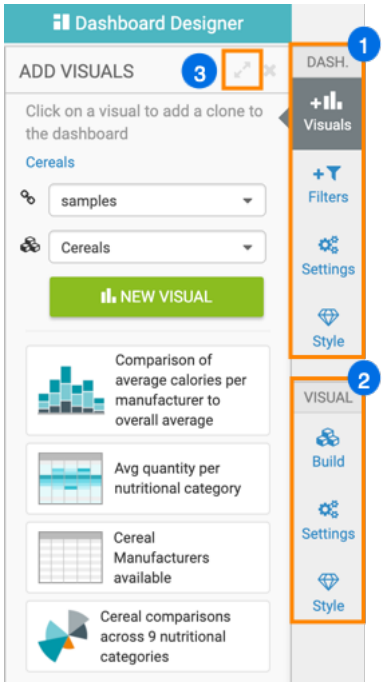

1. The dashboard menu selector enables you to choose between different menus in the Dashboard Designer.
   - Visuals – This menu allows you to add visuals to your dashboard.
     - Recently Created Visuals show you the new visuals added recently.
     - The Connection drop-down list lets you select an alternate connection. It is available in both the Visuals and Filters menus.

       Note: A dashboard can contain visuals from several connections.
     - The Dataset drop-down list lets you select an alternate dataset. It is available in both Visuals and Filters menus.

       Note: A dashboard can contain visuals from multiple datasets that share the same connection or from different datasets across different connections. When you change the dataset, a new set of existing visuals appears in the menu.
     - The NEW VISUAL button enables you to start a new visual from within the dashboard.
     - Existing visuals that belong to the specified dataset are displayed lower on the menu. Changing the dataset gives access to other visuals. You can click the visuals to add them to the dashboard.
   - Filters – This menu allows you to add filters to your dashboard.
     - The CREATE CUSTOM FILTER button allows you to create a custom filter based on any parameter and add it to your dashboard.
     - The Connection drop-down list lets you select an alternate connection. It is available in both Visuals and Filters menus.

       Note: A dashboard can contain visuals from several connections.
     - The Dataset drop-down list lets you select an alternate dataset. It is available in both Visuals and Filters menus.

       Note: A dashboard can contain visuals from multiple datasets that share the same connection or from different datasets across different connections. When you change the dataset, a new set of existing visuals appears in the menu.
     - Dimensions show you all fields by dataset that you can use as a filter.
     - Measures show you all fields by dataset that you can use as a filter.

       Note: You can also use the Search text box to look for a field.
   - Settings – This menu provides you with several options for dashboard display settings.
     - General
     - Filter Settings
     - Downloads

     See *Customizing general settings for dashboards* for more information.
   - Style – This menu provides you with various styling options for dashboards.
     - Dashboard Basics
     - Dashboard Title
     - Dashboard Subtitle
     - Filters
     - Visual Borders
     - Custom Style

     See *Customizing style for dashboards* for more information.

Note: You can minimize the side menu bar to see the full dashboard preview area by clicking the selected menu icon again.

2. If you already have a visual or visuals available in the dashboard and you are working on one of them, you can also see the visual menu selector in the side menu bar. It enables you to choose between different menus in the Visual Builder.
   - Build
   - Settings
   - Style

   For more information about these visual menu options, see *Visual Builder*.
3. At the top right corner of the Dashboard Designer side menu, you can click the [expand] icon to increase its width. To decrease the size of the pane, you can click the [compress] icon.

Related Information

Showing usage information
Emailing a dashboard
Downloading a dashboard
Getting the dashboard URL
Customizing general settings for dashboards
Customizing style for dashboards
Visual Builder

Cloudera Data Visualization 7.2.9

Data

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Datasets in Cloudera Data Visualization..... 4

Data Extracts in Cloudera Data Visualization..... 4

Data modeling in Cloudera Data Visualization.....6

Data joins in Cloudera Data Visualization..... 7

Datasets in Cloudera Data Visualization

Datasets are the foundation and starting point for visualizing your data. They are defined based on the connections to your data and provide access to the specific tables in the data store.

A dataset is the logical representation of the data you want to use to build visuals. It acts as a logical pointer to a physical table or a defined structure in your data source. Datasets may represent the contents of a single data table or a data matrix from several tables that may reside in different data stores on the same connection.

Beyond providing access to data, datasets offer several features to facilitate data usage and visualization, including (but not limited to):

- Table joins allow you to supplement the primary data with information from various other data sources. For more information, see *Data modeling*.
- Derived fields/attributes support the creation of flexible expressions, both for dimensions and aggregates. For more information, see *Creating calculated fields*.
- Hiding fields allows you to remove unnecessary fields or obscure sensitive data without affecting the underlying tables. For more information, see *Hiding dataset fields from applications*.
- Changing data types of field attributes can help proper data type handling or correct processing of numeric codes (like event IDs). For more information, see *Changing data type*.
- Changing default aggregation of fields at the dataset level prevents common mistakes when building visuals by setting appropriate field aggregations at the dataset level. For more information, see *Changing field aggregation*.
- Providing user-friendly names for columns or derived attributes can simplify the visualization process by applying meaningful names to columns or derived attributes, reducing the need for manual aliases. For more information, see *Automatically renaming dataset fields* and *Custom renaming dataset fields*.
- Using dataset versioning allows you to manage different versions of your datasets, which is especially useful when dealing with frequent updates or iterative changes. Versioning helps you to track changes and roll back to an earlier version if an error is introduced or an earlier version is preferred, ensuring data consistency and reliability. It also adds a layer of security and control, ensuring that data integrity is maintained even as datasets evolve. For more information, see *Tracking dataset versions*.

Note: Dataset versioning is currently available as a technical preview in Cloudera Data Visualization.

Related Information

Data modeling
Creating calculated fields
Hiding dataset fields from applications
Changing data type
Changing field aggregation
Automatically renaming dataset fields
Custom renaming dataset fields
Tracking dataset versions

Data Extracts in Cloudera Data Visualization

Data Extracts are saved subsets of data that you can use for data discovery and analytics.

Key concepts

When you create a Data Extract, you reduce the total amount of data you work with by selecting certain dimensions and measures. After you create an extract, you can refresh it with data from the original dataset. With the help of data

extracts you can manage the analytical capabilities, performance, concurrency, and security of data access in your system.

Data Extracts are created from a single dataset but a dataset can have multiple extracts associated with it. You can create a Data Extract in Cloudera Data Visualization on any target connection that supports it. Since the target of an extract is a simple representation of data on the target data connection, you can use it independently as part of a different dataset, or even use it as the source for a further transformation into a second Data Extract.

When a Data Extract is created, it targets a data connection as the new location of the data. This data connection can be the same or a different connection that you have used to build the extract. A second dataset built on the target data will be used as the dataset for building visuals.

You can refresh Data Extracts on a schedule. This functionality creates a new kind of job and uses the standard job scheduling mechanism to schedule and monitor ongoing refreshes. Alternatively, you can do the refresh manually when you need it.

You can use extracts to model data within or across data connections. You can also use extracts to materialize the data models on the same or different data connections. Data modeling is done in the context of a dataset built on the source connection. The modeled form of the data can be materialized on that same or on a different data connection. Modeling includes everything that is possible within a dataset, including joins, scalar transformations, hidden columns and complex aggregations.

Note: Data Extracts are run on a data connection with the permissions of the user who sets up the Data Extract. This user needs to have all the permissions associated with running a Data Extract, including Manage AVs/Data Extracts permissions on both the source and target data connections, and the permissions in the backend data connection to drop and create the target table, select data from the source table, and insert data into the target table.

Key features

Improving analytical capabilities

Data Extracts support large data sets. You can create extracts from data sets that contain billions of rows of data. Extracts permit joins across data connections. You can move data first and then join the data on the target system.

Data Extracts enable you to use different data connection capabilities for analysis. You can move data from an analytical system to an event-optimized or search-optimized store.

Improving performance

Data extraction offers increased performance when large tables or datasets are slow to respond to queries. When you work with views that use extracted data sources, you experience better performance than when interacting with views based on the original dataset.

With Data Extracts, you can do the following:

- Model storage of data into columnar forms
- Move data to a different query system that performs better

When source data requires expensive join and transformation operations for reporting use cases, you can pre-compute join and transformation operations.

When base tables are large but reporting requires a subset of data, or is only needed on well understood roll-ups, you can pre-compute filters and aggregations as Data Extracts.

Improving workload management

You can use isolated systems for performance characterization on particular workloads by moving data from a busy system to another isolated system.

Improving security

You can provide access only to the relevant reporting data by masking column names and schemas from end users through building dashboards off a derived copy of the data with masking enforced.

You can move data that is relevant for reporting from one system to another and ensure that end users only have access to the target system. This ensures that only the data that is relevant for reporting is accessible to end users.

Workflow

The following diagram shows you the workflow of building visualizations on top of data extracts:

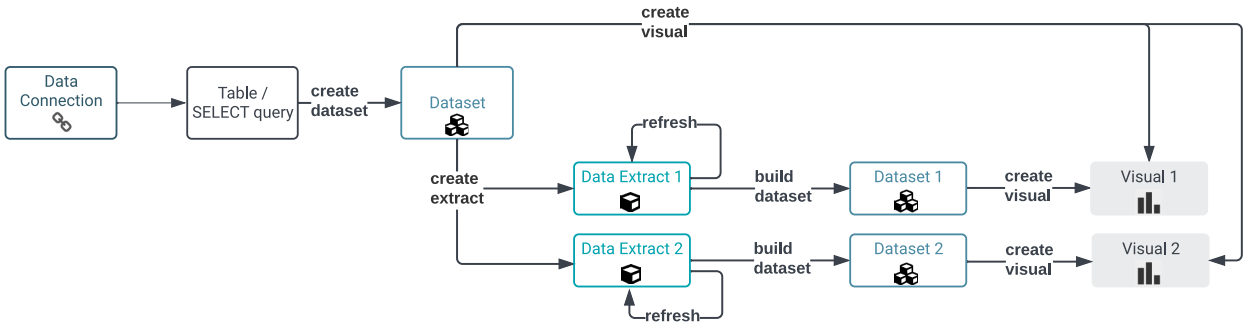

Supported sources

Cloudera Data Visualization supports the following sources for Data Extracts:

- Hive
- Impala
- MariaDB
- MySQL
- PostgreSQL
- SQLite (not supported in Cloudera Data Warehouse)
- Snowflake [Technical Preview]

Supported targets

Cloudera Data Visualization supports the following targets for Data Extracts:

- Hive
- Impala
- MariaDB
- MySQL
- PostgreSQL
- SQLite (not supported in Cloudera Data Warehouse)
- Snowflake [Technical Preview]

Data modeling in Cloudera Data Visualization

With Cloudera Data Visualization, you can create logical datasets that model semantic relationships across multiple data sources. These datasets enable you to build blended visuals, dashboards, and applications.

You can expand a basic, one-table dataset by creating joins with other relevant tables from the same source or from other data stores. The joins created in Cloudera Data Visualization are not materialized. They are calculated during run-time.

Combining data across multiple tables enriches the dataset considerably; it enables you to conduct much more meaningful research and produce insightful visual analytics.

There are two distinct approaches to create data joins for visualization:

- Defining in UI is ideal for datasets that include star-type schemas.
- Defining on back-end ETL is preferable for fact-fact joins, so they may be pre-materialized.

See *Working with data models* for instructions on how to create and manage joins.

See topics in *How To: Advanced Analytics* for more advanced data modeling details, such as dimension hierarchies, segments, and events.

See *Setting filter associations* for information about filter associations related to data modeling and management.

Related Information

Working with data models
How To: Advanced Analytics
Setting filter associations

Data joins in Cloudera Data Visualization

With Cloudera Data Visualization, you can create joins between different table columns.

Cloudera Data Visualization supports five types of column connections, also known as joins.

Note: The types of joins available depend on the underlying database. For example: MySQL connections do not support FULL OUTER JOIN operations. SQLite connections do not support FULL OUTER JOIN and RIGHT OUTER JOIN operations.

Inner join

It is the most common join type. Rows in the result set contain the requested columns from both tables for all combinations of rows where the selected columns of the tables have identical values.

Left join

The result set contains all rows from the left table and the matching data from the right table. Whenever no matching data is available on the right side of the join, the corresponding columns in the result set have the value NULL.

Right join

The result set contains all rows from the right table and the matching data from the left table. Whenever no matching data is available on the left side of the join, the corresponding columns in the result set have the value NULL.

Outer join

An outer join returns all rows from both tables. Whenever no matching data is available on one of the sides of the join, the corresponding columns in the result set have the value NULL.

Left outer join

It supports join elimination in datasets that contain more than one table. By default, this setting is on, but it can be disabled.

Related Information

Working with data models

Cloudera Data Visualization 7.2.9

Working with Derived Data

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Defining derived data **4**

Defining additional derived data **6**

Using derived data **10**

Viewing derived data definitions **14**

Saving derived data with full context **16**

Saving derived data expression only 16

Deleting derived data definitions **16**

Defining derived data

Procedure

1. Inside a visual that generates a subset of data, select a measure field on a shelf.

   In this example, sum(population) is used.
2. Click the Right Arrow icon to the right of the field.

3. In the dropdown, click Save Expression.

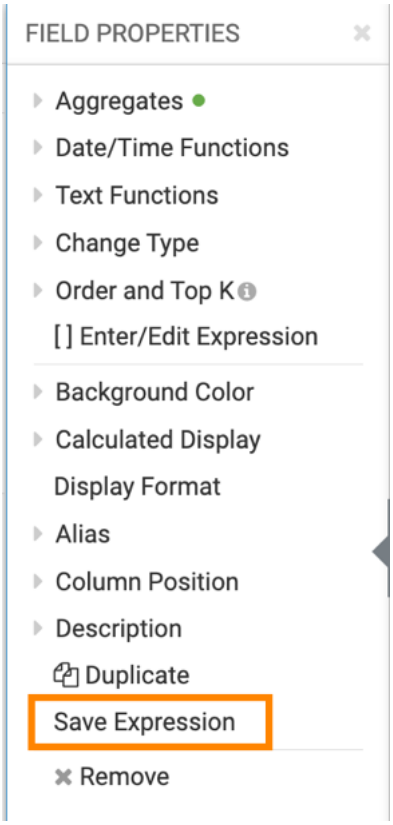

4. In the Save expression to dataset or as derived data modal window, under Name, type POP.

5. Select Calculated field in visual derived data section and click SAVE.

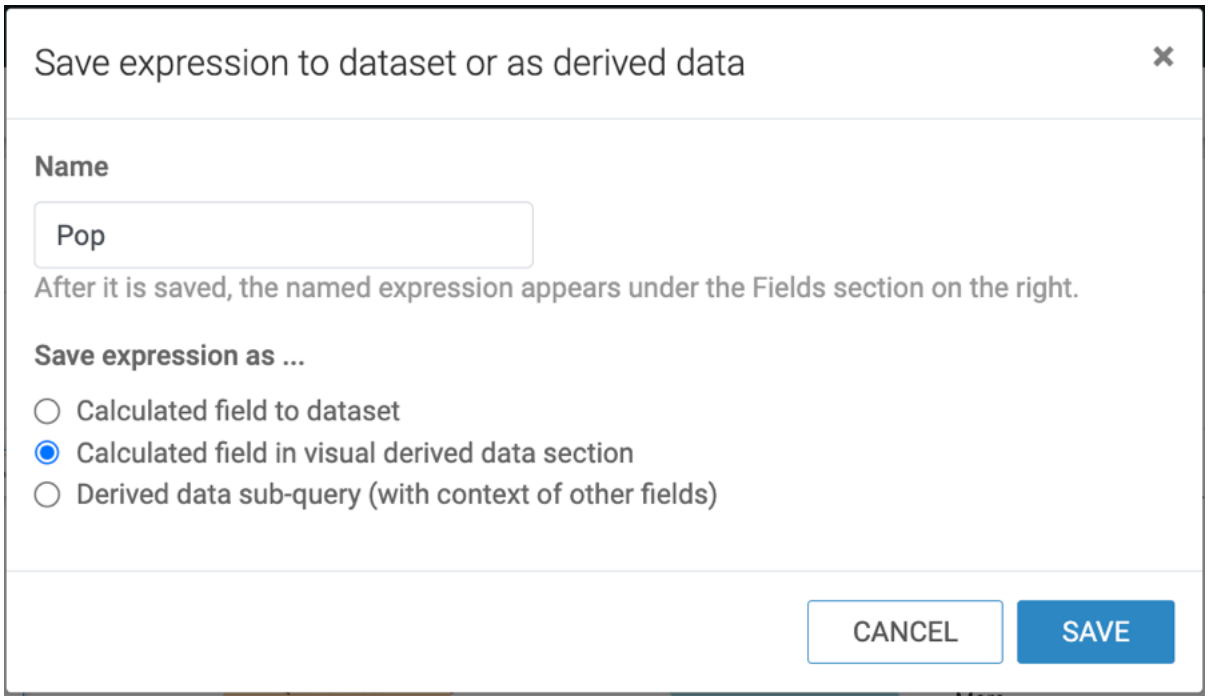

You can see that in the right navigation under Measures, you have a new entry, POP.

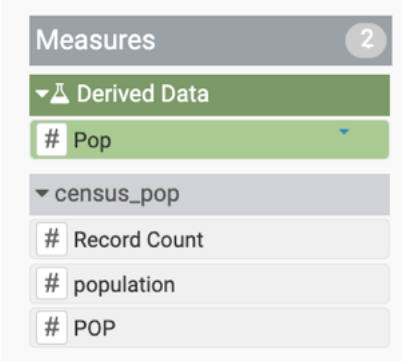

6. To check the definition of the saved derived data POP, click the down arrow to the right of its name, and click View Definition.

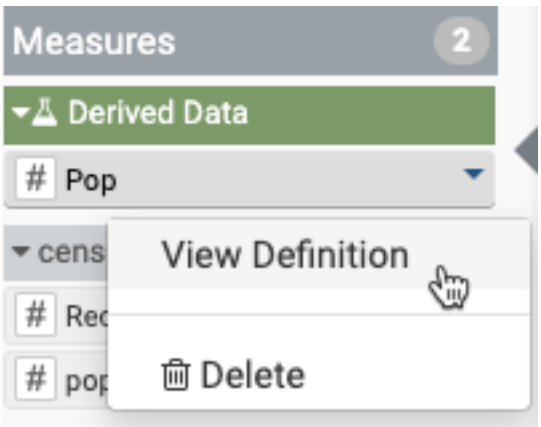

The Derived Data Definition modal window appears. You can see that the definition includes a parameterized filter for year.

7. Click SAVE to save both the visual and the new derived data definitions.

Results

You can now use Pop as any other measure.

Defining additional derived data

About this task

Next, define other derived data in the same shelf set-up and save it as derived data LifeExpXPop.

Procedure

1. On the Dimensions shelf, place the following fields: un_region, un_subregion, and country.
2. Add the derived data Pop to the Dimensions shelf.

   You should do this because fields on the Measure shelf must be aggregated, but the definition of Pop already includes the summation over the population field.
3. Place the field life_expectancy on the Measures shelf and change the aggregate to avg(life_expectancy).

4. Place the field year on the Filters shelf and change the expression to [year]=<<year_param:2000>>.

5. Duplicate Derived data Pop in place, on the Dimensions shelf.
6. Click the Down Arrow icon to the right of the cloned field and click Enter/Edit Expression in the dropdown.
7. In the Enter/Edit Expression modal window, change the expression to [life_expectancy]*[Pop] and save it.

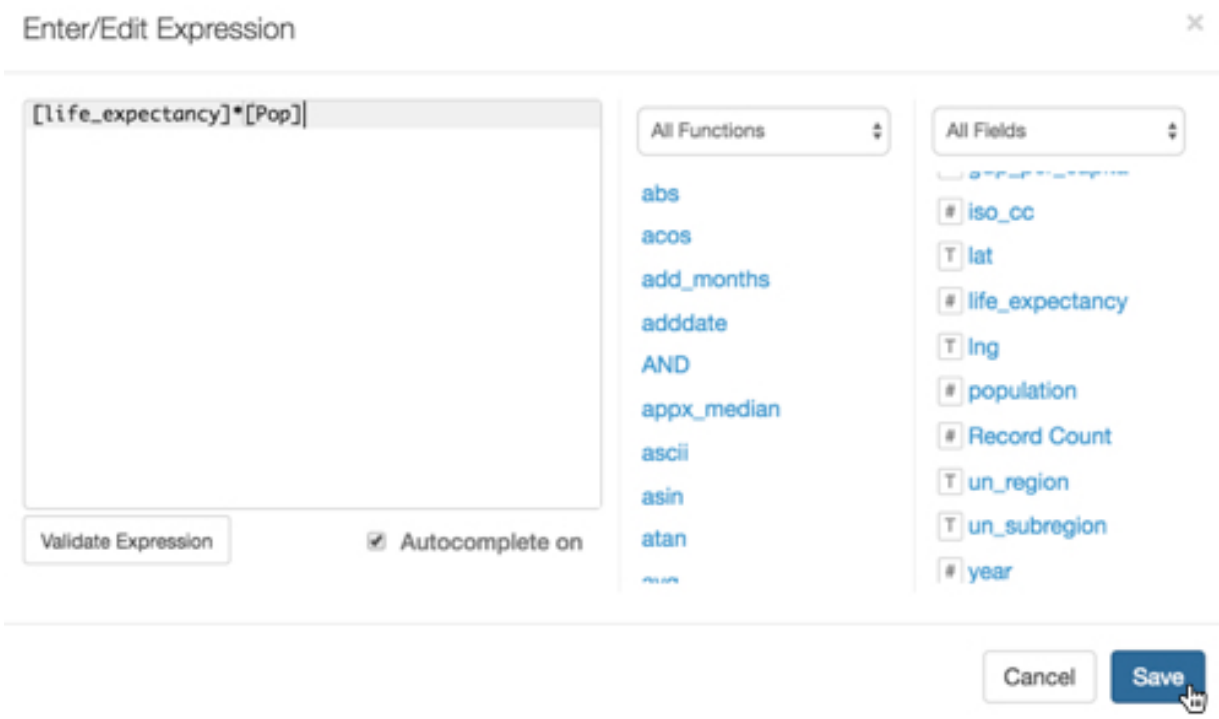

8. Save the same field as derived data, named LifeExpXPop.

   You can see that in the Data menu, under Dimensions, you have a new category Derived Data that contains Life ExpXPop.

9. To check the definition of the saved derived data LifeExpXPop, click the down arrow to the right of its name and click View Definition.

   The Derived Data Definition modal window appears. Note that the definition includes a parametrized filter for year, and the Pop derived data field.

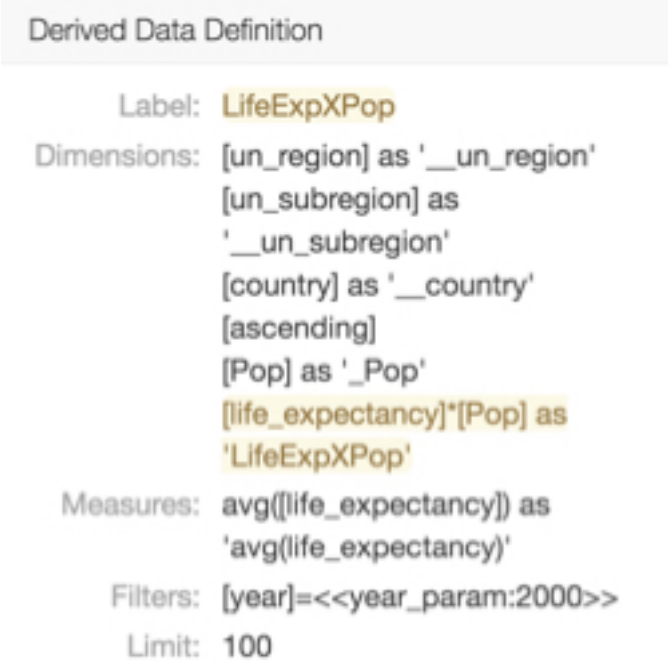

10. Replace the field on the Dimensions shelf with the new derived data field, click Refresh Visual, and Save the application.

    The visual should look similar to this one. Note that we aliased the fields and made some changes to display formats:

UN Region	UN Subregion	Country	Pop	LifeExpXPop	LifeExpectancy
Asia	Southern Asia	Afghanistan	22,856,302	1,254,811,040	54.90
Europe	Southern Europe	Albania	3,071,856	227,931,710	74.20
Africa	Northern Africa	Algeria	30,533,828	2,106,834,175	69.00
Africa	Middle Africa	Angola	13,926,373	629,472,083	45.20
Americas	Caribbean	Antigua and Barbuda	77,656	5,707,716	73.50
Americas	South America	Argentina	36,930,708	2,725,486,419	73.80
Asia	Western Asia	Armenia	3,076,098	219,941,012	71.50
Americas	Caribbean	Aruba	90,271	6,662,000	73.80
Oceania	Australia and New Zealand	Australia	19,164,352	1,531,231,785	79.90

< 1 2 3 4 5 >

Using derived data

About this task

To use the derived data measurements in a calculation, treat it as any other measurement. In this example, the derived data LifeExpXPop is used to accurately calculate the weighted life expectancy for an aggregation dimension, such as UN Region or UN Subregion. You can do this by dividing the sum of Life Expectancy X Population by sum(population) at the appropriate aggregate level.

Procedure

1. Place the following fields on the Dimensions shelf: un_region and un_subregion.
2. Add the derived data field Pop.
3. Place the field year on the Filters shelf , and change the expression to [year]=<<year_param:2000>>.
4. Place the derived data field LifeExpXPop onto the Meausres shelf. Notice that it appears as an aggregation, sum(LifeExpXPop).

5. Using the Enter/Edit Expression window modal, change the definition of the field as follows,and click Save. You can see that the field has been aliased.

```
(sum([LifeExpXPop])/sum([population])) as 'Weighted Life Expectancy'
```

6. Use the Alias option to change the column names to UN Region, UN Subregion

7. Click Refresh Visual.

   The visual should look similar to this one:

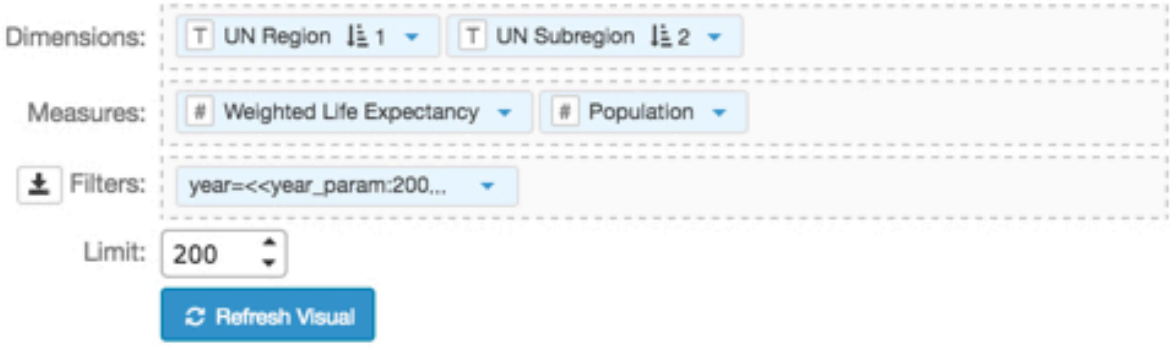

UN Region	UN Subregion	Weighted Life Expectancy	Population
Africa	Eastern Africa	52.37	108,167,918
Africa	Middle Africa	47.42	89,208,823
Africa	Northern Africa	68.72	98,182,244
Africa	Southern Africa	50.60	1,757,925
Africa	Western Africa	54.14	49,296,775
Americas	Caribbean	68.85	23,593,157
Americas	Central America	70.25	27,565,249
Americas	Northern America	79.38	30,723,560
Americas	South America	71.35	279,783,195
Asia	Central Asia	64.17	19,911,544
Asia	Eastern Asia	72.97	1,401,620,301
Asia	South-Eastern Asia	67.01	225,842,357
Asia	Southern Asia	62.31	1,206,917,948
Asia	Western Asia	71.56	54,154,366
Europe	Eastern Europe	70.78	28,274,513
Europe	Northern Europe	75.83	18,353,582
Europe	Southern Europe	78.82	79,244,298
Europe	Western Europe	78.50	159,577,216
Oceania	Australia and New Zealand	79.90	19,164,352
Oceania	Melanesia	67.60	811,718
Oceania	Micronesia	71.35	239,157

8. [Optional] To see the correctly calculated Life Expectancy for a UN Region, remove UN Subregion from the Dimensions shelf.

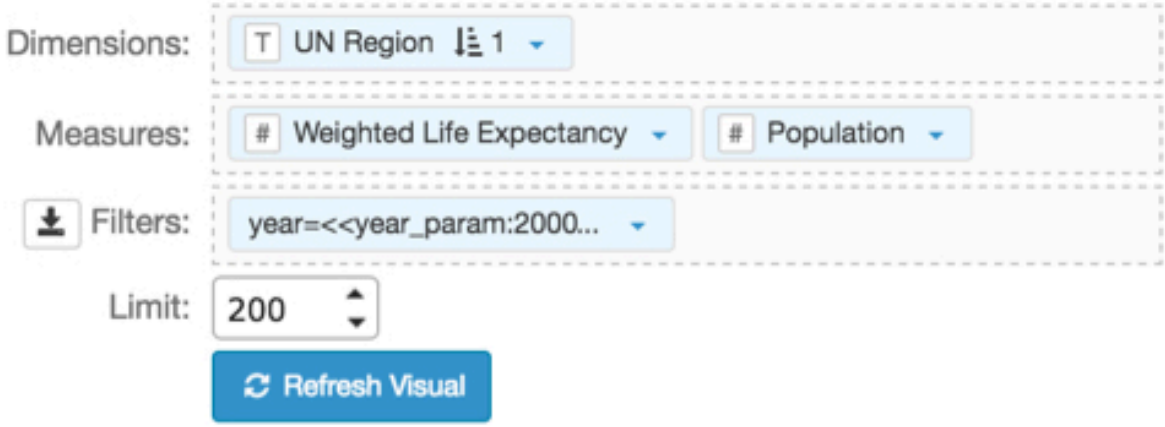

UN Region	Weighted Life Expectancy	Population
Africa	55.97	346,613,685
Americas	71.79	361,665,161
Asia	68.00	2,908,446,516
Europe	77.65	285,449,609
Oceania	79.22	20,452,865

Viewing derived data definitions

Procedure

1. In the left navigation, under Measures Derived Data , select the derived measure you want to check.

   In this example, it is Pop.
2. Click the Down Arrow icon to the right of the field.

3. In the drop-down, select View Definition.

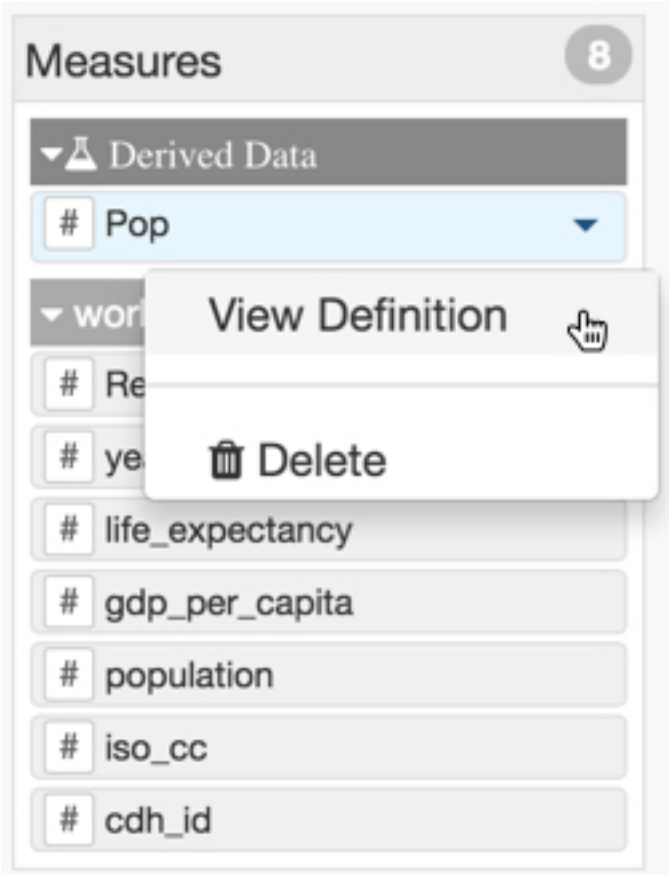

The Derived Data Definition modal window appears.

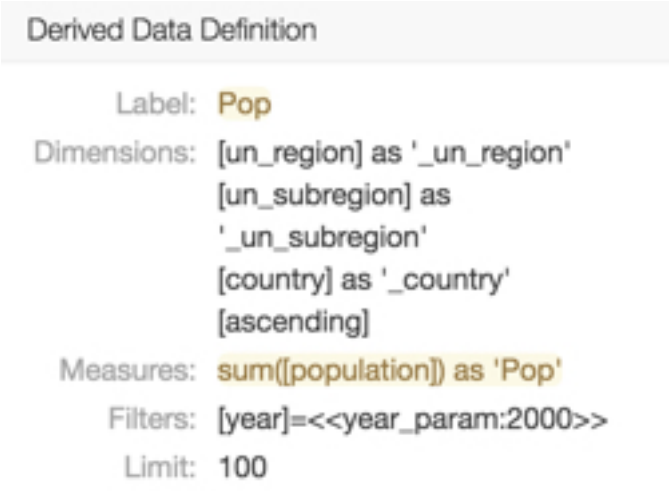

Note: You cannot edit the definition of derived data. If you want to change it, you must first delete it and then create and save a new definition. For more information, see *Deleting derived data definitions* and *Defining derived data*.

Related Information

Deleting derived data definitions

Defining derived data

Saving derived data with full context

This is the default option, when the entire context of the derived data expression can be captured and re-used. In he earlier example, the definition of derived data includes Label, Dimensions, Measures, and Filters:

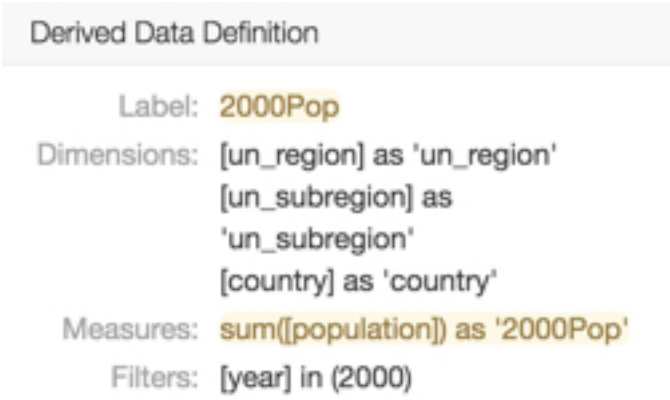

Saving derived data expression only

Procedure

Deleting derived data definitions

Procedure

1. In the left navigation under Measures Derived Data , identify the derived measure you want to delete.

   In this example, it is 2000Pop.
2. Click the Down Arrow icon to the right of the field.

3. In the drop-down, select Delete.

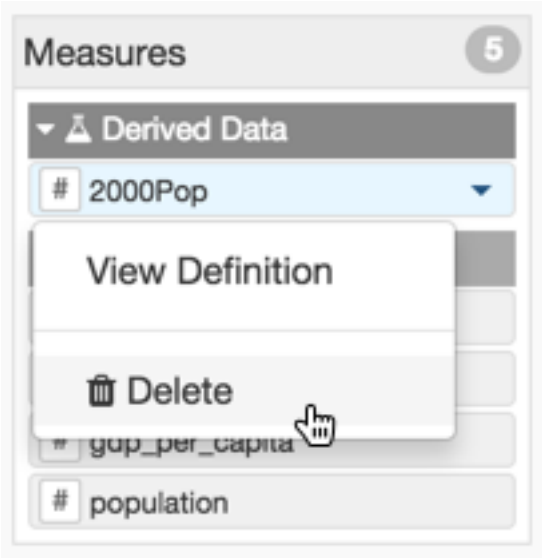

4. Click Save to save both the visual and the changes to derived definitions.

Cloudera Data Visualization 7.2.9

Working with Dimension Hierarchies

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating new dimension hierarchies **4**

Changing the name of the hierarchy **8**

Changing the order of components in the hierarchy **8**

Adding levels to the hierarchy **9**

Removing levels from the hierarchy **9**

Deleting dimension hierarchies **9**

Hierarchy drill into support on the color shelf **10**

Creating new dimension hierarchies

About this task

You may find it useful to define a hierarchy of dimensions, to enable visuals with variable levels of granularity over the same range of data.

Note:

This feature is available only to users with administrative privileges.

The following steps demonstrate how to define such a hierarchy on the dataset World Life Expectancy [data source samples.world_life_expectancy]. In this example the dimensions un_region, un_subregion, and country are used to define a dimensional hierarchy called Region.

Procedure

1. On the main navigation bar, click DATA.

   The Data view appears open on the Datasets tab.
2. In the left navigation menu, click samples.
3. In the Datasets area, select World Life Expectancy (samples.world_life_expectancy).

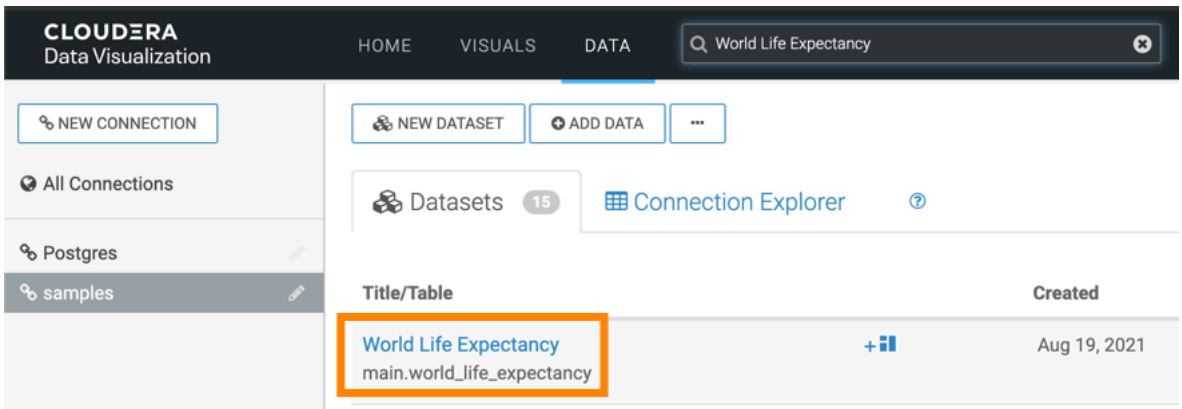

4. In the Dataset Detail menu, select Fields.

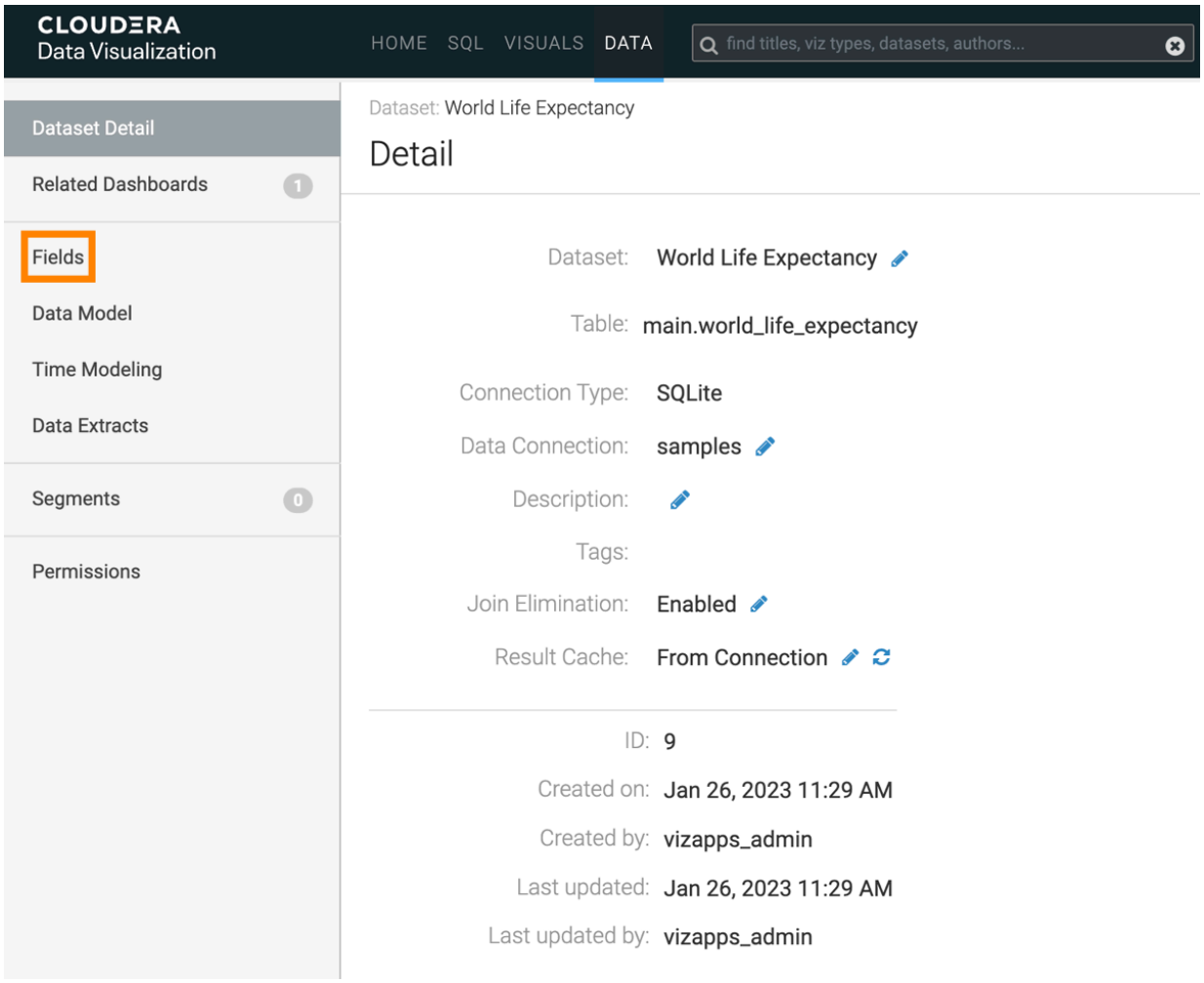

5. In the Fields interface, select Edit fields.

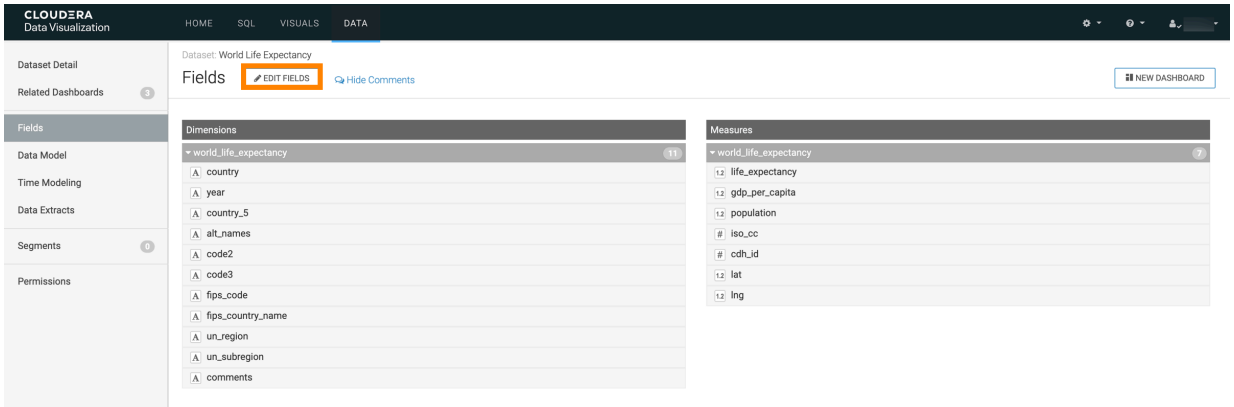

6. In the editable Fields interface, under Dimensions, find the field un_region.

7. Click Down icon on the un_region line, and select Create Hierarchy.

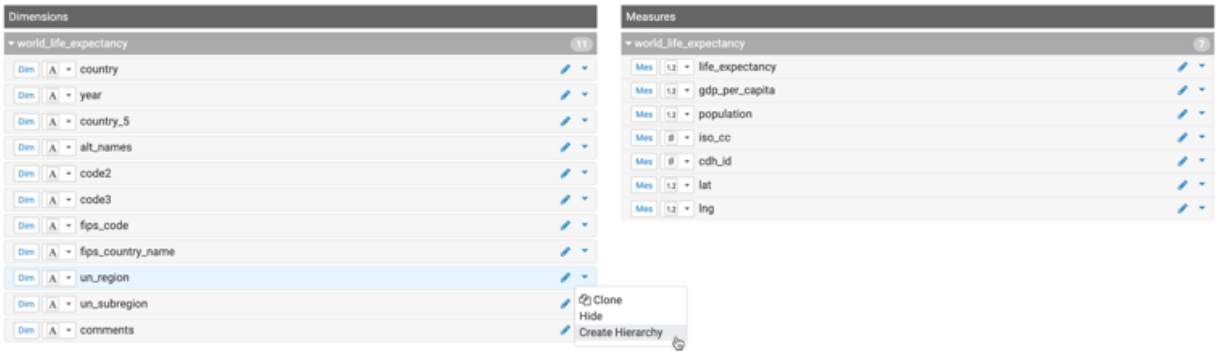

8. In the Create Hierarchy modal window, enter the values for Hierarchy Name and Hierarchy Comment, and click CREATE.

   The example shows a hierarchy named Region, described as Geographical Granularity.

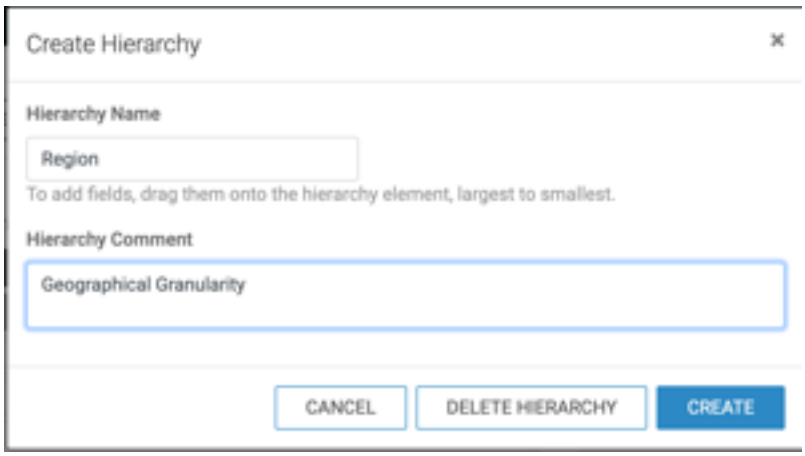

9. Note that Measures now contain a hierarchy Region, denoted by the Hierarchy icon. The hierarchy contains a single element, un_region.

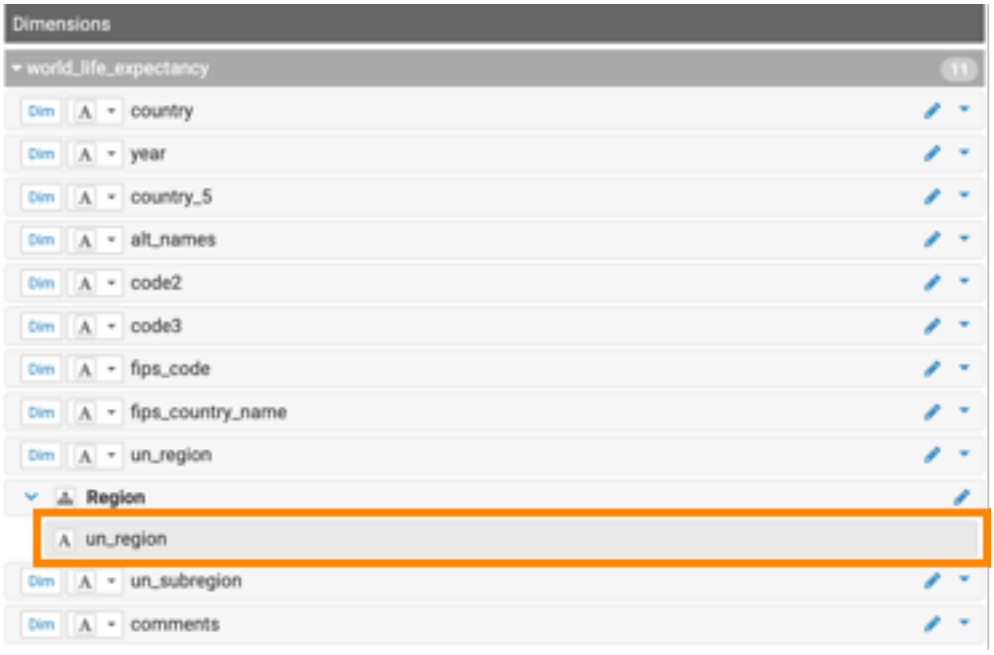

10. To add more levels to the hierarchy, simply click and drag the relevant dimensions or measures to the newly created hierarchy. Make sure to arrange them in order of scale, from largest to smallest.

Below un_region, un_subregion was added.

Below un_subregion, country was added.

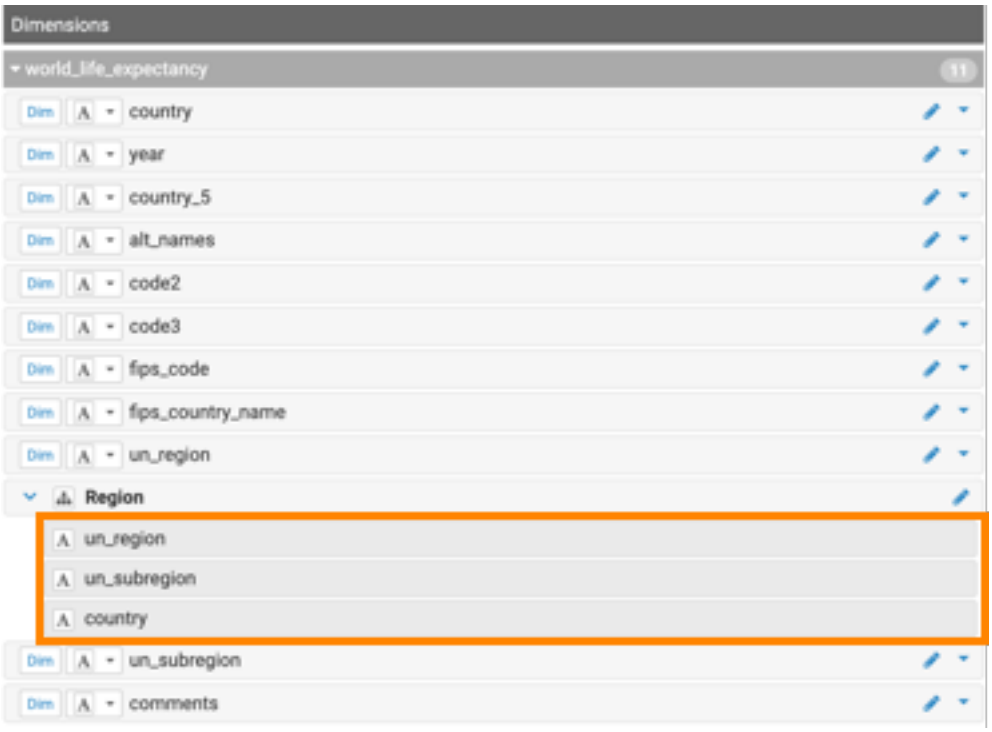

11. Click SAVE.

Results

Defined dimension hierarchies, such as Region created here, can be used just as any other dimension field of the dataset.

Changing the name of the hierarchy

Procedure

1. Click Edit Fields.
2. In the list of Dimensions, find the Region hierarchy, and click the Pencil icon next to it.

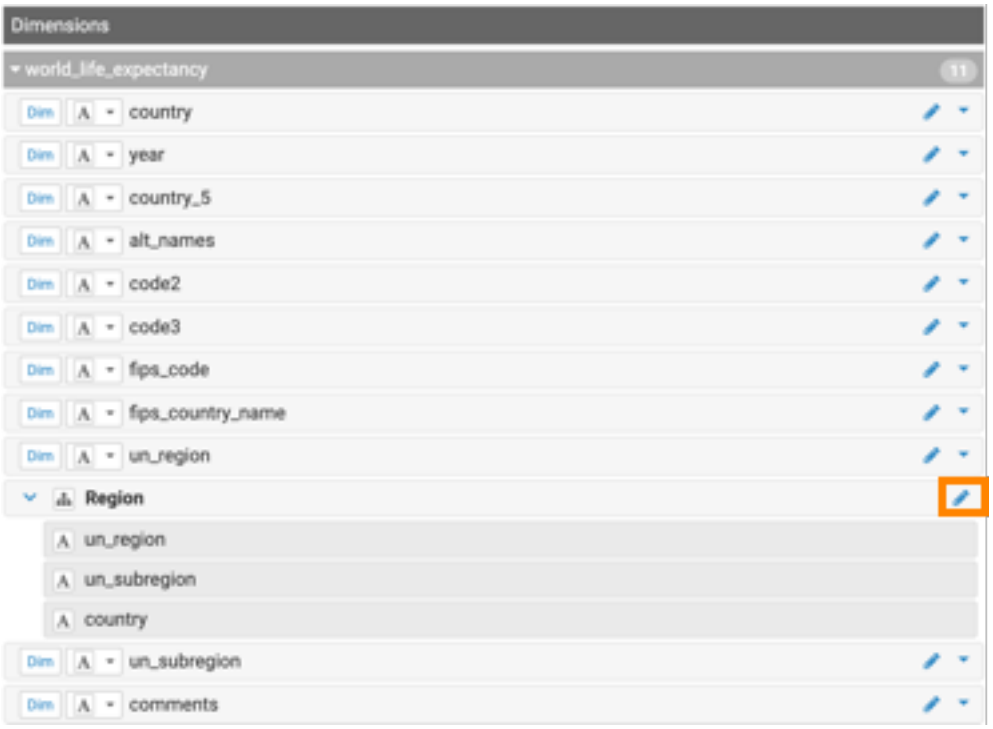

3. In the Edit Hierarchy modal window, change Hierarchy Name.
4. Click Update.

   The hierarchy title Geography now appears in the list of dimensions, instead of Region.
5. Click SAVE.

Changing the order of components in the hierarchy

Procedure

1. Click Edit Fields.
2. Click one of the components of the hierarchy, and move it up or down the list.
3. Click Save.

Adding levels to the hierarchy

Procedure

1. Click Edit Fields.
2. Pick a component to add to the hierarchy, such as un_subregion, click it, and drag into the desired position inside the hierarchy area.
3. Click Save.

Removing levels from the hierarchy

Procedure

1. Click Edit Fields.
2. Pick a component of the hierarchy to remove, and click its Remove icon.
3. Click Save.

Deleting dimension hierarchies

About this task

To delete the hierarchy, you must make changes in the dataset Fields interface.

Procedure

1. Click Edit Fields.

2. In the list of Dimensions, find the hierarchy to delete, and click the Pencil icon next to it.

   We are deleting the Region hierarchy.

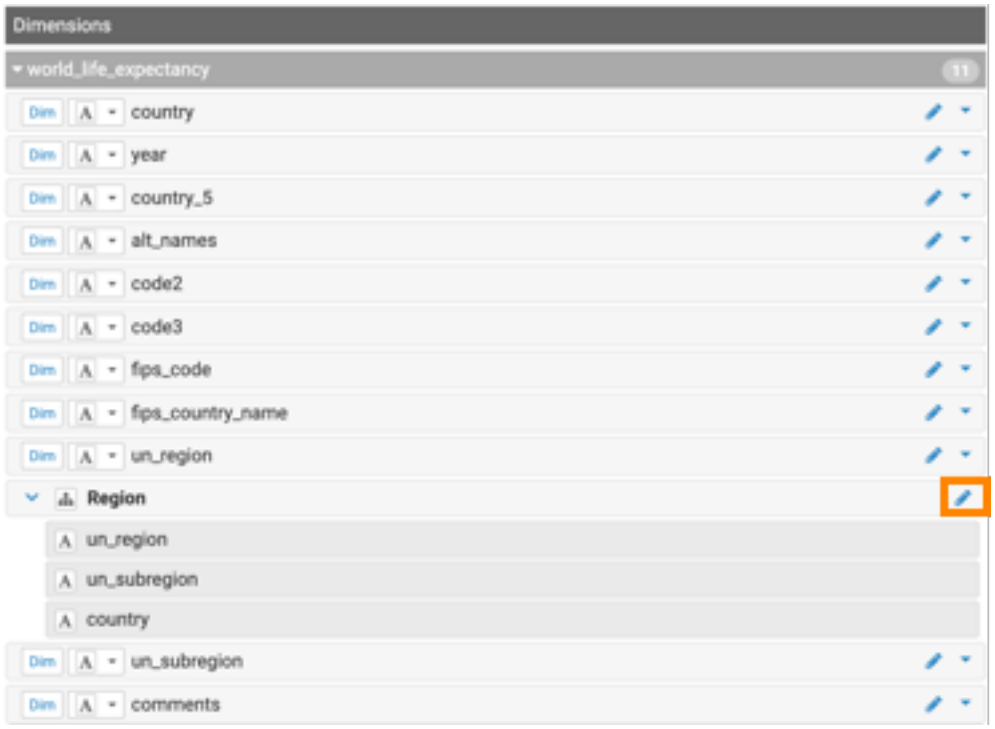

3. In the Edit Hierarchy modal window, click Delete Hierarchy.

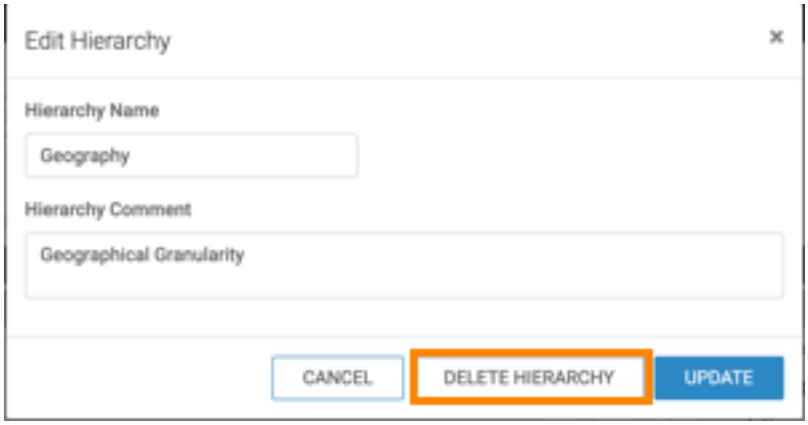

4. Click Save.

   The hierarchy no longer appears in the list of dimensions.

Hierarchy drill into support on the color shelf

About this task

You can use the hierarchy drill-down data discovery on the Colors shelf of the Lines, Bars, Areas, Grouped Bars and Scatter visuals. To use the drill-down, click the legend colors.

Note: You must create the hierarchy in advance, on the Dataset Field interface.

Cloudera Data Visualization 7.2.9

Embedding apps with client pages

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Embedding apps with client pages 4
Embedding apps with unauthenticated access 4
Embedding apps with trusted authentication 5
Overview of trusted authentication process 5
Enabling trusted authentication 6
Authenticating embedded requests 7
Embedding a dashboard without header 8
Embedding a jobs status page without header 9

Embedding apps with client pages

Cloudera Data Visualization allows you to embed applications into client pages using an HTML iframe component. You have two options: open access, which makes the app available to all users without requiring authentication or login, or trusted authentication for added security.

About this task

When you embed a Cloudera Data Visualization app into an iframe on a web page, you need to specify the frame-an cestor attribute, otherwise the browser security policy prevents the application from rendering on the page. The host sources you list will be the locations where the app can be embedded.

Before you begin

- If you want to embed a Cloudera Data Visualization app into a client page iframe without requiring authentication, enable the Enable Unauthenticated Access option for the app. For more information and instructions, see the *Public Applications* section in Securing Applications.
- Turn on the correct cookie settings by setting the environment variable IS_EMBED='true'.
- If you want to maintain the embedding URL and prevent links from opening the application outside the embedded environment, add EMBED_URL_PREFIX=embeddingurl to the advanced site settings and restart the application. This prefix will be added to all Cloudera Data Visualization URLs.

  For example: EMBED_URL_PREFIX=https://embeddingsite.com# will open the url https://embeddingsite.com#/arc/apps/data if you open the data page in a new tab.

Procedure

1. Navigate to Project Settings Advanced in Cloudera AI/Cloudera Data Science Workbench.
2. Set the CDSW_FRAME_ANCESTORS environment variable: specify one or more website locations where you want to embed the application in <host-source> format.

   If you are adding multiple websites, use a comma-separated list.

Embedding apps with unauthenticated access

In Cloudera Data Visualization, providing unauthenticated access when embedding apps can be a valuable aspect of your configuration. Unauthenticated access simplifies user interactions, making it seamless for viewers to engage with your embedded content. By adding specific login settings in the Advanced Site Settings, you can enable a smooth experience for your users.

Procedure

1. In Cloudera Data Visualization, add the following login settings in the Advanced Site Settings text box in key = value format under Site settings Advanced Settings .
   - AUTOLOGIN_ENABLED = False
   - AUTOLOGIN_USERNAME = None
   - AUTOLOGIN_PASSWORD = None
2. Restart the Cloudera Data Visualization application for the changes to take effect.

Embedding apps with trusted authentication

Embedding Cloudera Data Visualization apps within client pages can enhance the user experience and extend the functionality of your applications. One option for embedding is trusted authentication, which introduces an additional layer of security, ensuring that only authorized requests are processed.

Overview of trusted authentication process

Cloudera Data Visualization supports embedding applications within client pages through an HTML iframe component. You have two options: one for open access and another with trusted authentication for enhanced security.

In trusted authentication, the Cloudera Data Visualization Server authenticates the <iframe> request made by the client and then returns the visualization. To achieve this authentication, Cloudera Data Visualization uses the trusted authentication protocol, which involves the following steps:

1. **User Browser requests an App page.**

   The user requests a web page from the parent web server, which includes an embedded Cloudera Data Visualization visual within an <iframe> element.

2. **App Server requests a ticket from the Cloudera Data Visualization Server.**

   The parent App Server makes a POST ticket request to the Cloudera Data Visualization Server, including the Cloudera Data Visualization username for authenticating the <iframe>.

   The ticket request can be authenticated through one of two methods:

   - Ticket-granting user: The ticket request includes the Cloudera Data Visualization username and password of the trusted ticket granter. This account does not normally have admin or superuser privileges. For more information, see *Post ticket request using a ticket-granting user*.
   - Trusted IP: The parent App Server is listed among trusted IPs. The POST request includes only the Cloudera Data Visualization username to obtain the ticket-granting user's full credentials. For more information, see *Post ticket request using an IP*.

   By default, the ticket may only be used once. However, it can be configured for multiple uses for debugging purposes. The ticket is valid for a configurable time period before expiring.

3. **Cloudera Data Visualization Server authenticates the request and returns a unique ticket.**

   - If the request is valid, the Cloudera Data Visualization Server creates a ticket and returns it as a response to the POST request.
   - If the request is invalid, it returns the value of -1 as a response to the POST request.

4. **App Server returns an HTML page that contains an iframe tag with Cloudera Data Visualization URL and the ticket.**

   The parent App Server uses the ticket to generate a unique URL containing the ticket for the embedded visual. This URL is used for the visual's <iframe> element in the HTML returned to the client. For more information, see Request Visual from Cloudera Data Visualization Server.

5. **User Browser requests the iframe from the Cloudera Data Visualization Server, including the ticket.**

   The client browser uses the iframe URL obtained in the previous step to request the App from the Cloudera Data Visualization Server.

6. **Cloudera Data Visualization Server authenticates User Browser based on the ticket and returns the visualization for the iframe.**

   The Cloudera Data Visualization Server authenticates the <iframe> request based on the ticket that is part of the request URL. If the ticket is valid, it automatically logs in the username specified in the original POST request and then sends the visual to the client.

After the user is logged in using the ticket, they can request any other URL until that session expires. The login session expires at the end of the browser session.

Enabling trusted authentication

Before embedding Cloudera Data Visualization Services within client pages, you must first enable trusted authentication on the Cloudera Data Visualization Server.

Procedure

1. Add the following settings to the advanced site settings.

```
INSTALLED_APPS = INSTALLED_APPS + ('trustedauth',)

AUTHENTICATION_BACKENDS = (
      'django.contrib.auth.backends.ModelBackend',
      'trustedauth.backends.TrustedAuthBackend'
)

TRUSTED_AUTH = {
      'trusted_ips': ['127.0.0.1'],
      'trusted_users': ['tadmin'],
      'timeout': 120,
      'single_use': True,
      'session_expiry': 0,
      'allow_superuser': True
```

```
}
```

Settings explanation:

trusted_ips:

A list of trusted IPs. Ticket requests from these IP addresses are validated. You can either specify a list of trusted_ips, a list of trusted_users, or both.

trusted_users:

A list of trusted ticket-granting usernames. You can either specify a list of trusted_users, a list of trusted_ips, or both.

timeout:

The time that the ticket remains valid, in seconds.

single_use:

The ticket can be used only one time.

session_expiry:

The duration time of the user session, in seconds. A setting of 0 ends the user session when the browser closes.

allow_superuser:

Allows authentication of a user with admin privileges using a ticket. Set this to False to disable this feature.

2. Restart the application to apply the new configuration.

Authenticating embedded requests

For embedding apps within client pages, Cloudera Data Visualization uses the trusted authentication protocol to authenticate embedded requests.

About this task

Follow these steps to authenticate an embedded request from the client:

Procedure

1. Request a ticket from the Cloudera Data Visualization Server.

   The parent Application Server sends a POST ticket request to the Cloudera Data Visualization Server, either by using the ticket-granting Cloudera Data Visualization username, an IP address, or both. The ticket request has the following syntax:

   ```
   https://<appserver>/arc/trustedauth/getticket
   ```

   - Posting a ticket request using a ticket-granting user:

     To authenticate the ticket request using the trusted ticket granter's Cloudera Data Visualization username and password, use the following syntax:

     ```
     curl --data \ "username=cdvuser&trustedusername=ticketgranter&trustedpas
     sword=trustedpass" \
     ```

```
http://127.0.0.1:8000/arc/trustedauth/getticket
```

If the request is valid, the Cloudera Data Visualization Server returns the ticket cYvvmRSHSWOOWNCO eve1vA.

- Posting a ticket request using an IP Address:

  In this case, the Cloudera Data Visualization Server already has the parent Application server IP in the list of trusted IPs. The POST request includes only the Cloudera Data Visualization username to get the ticket-granting user's full credentials. To authenticate the ticket request using an IP address, use the following syntax:

```
curl --data "username=cdvuser" http://127.0.0.1:8000/arc/trustedauth/get
ticket
```

  If the request is valid, the Cloudera Data Visualization Server returns the ticket cYvvmRSHSWOOWNCO eve1vA.

The following POST parameters are used in the preceding examples:

- Username: User identifier for automatic login.
- Trustedusername: User identifier for ticket-granting user (optional when using trusted IP authentication).
- Trustedpassword: Password for ticket granting user.

2. Generate a unique URL.

   The parent Application Server uses the ticket to generate a unique URL, which contains the <iframe> tag and the ticket for the embedded visual, and sends it to the client.

   For example, the URL address in the <iframe> would be:

```
http://127.0.0.1:8000/arc/trustedauth/trusted/cYvvmRSHSWOOWNCOeve1vA/app/1
```

3. Request visual from the Cloudera Data Visualization Server.

   The client browser uses the <iframe> URL obtained from the Application Server and forwards the same URL to the Cloudera Data Visualization Server, requesting the visual.

```
http://127.0.0.1:8000/arc/trustedauth/trusted/cYvvmRSHSWOOWNCOeve1vA/app/1
```

4. Return Cloudera Data Visualization visual.

   The Cloudera Data Visualization Server authenticates the <iframe> request based on the ticket that is part of the request URL. If the ticket is valid, it automatically logs in the username specified in the original POST request and then sends the visual to the client.

Embedding a dashboard without header

You can embed a Cloudera Data Visualization dashboard within a client web page without displaying the Cloudera Data Visualization header, which includes the navigation bar and logo.

About this task

Follow these steps to remove the header from a dashboard.

Note: You can only remove the header in View mode.

Procedure

1. Open a dashboard in View mode.

   You can see that the dashboard has a Cloudera header at the top.

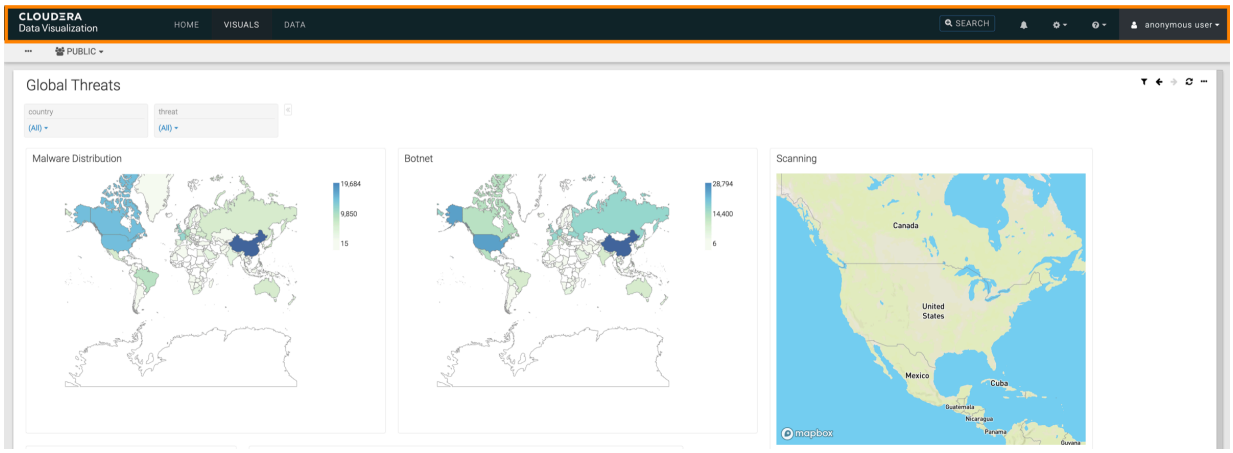

2. Identify the URL of the dashboard.

   It typically follows this format:

```
http://<appserverip>:<port>/arc/apps/app/<dashboard ID>
```

   For example:

```
http://127.0.0.1:8000/arc/apps/app/3851
```

3. Add ?embed=true at the end of the URL and click enter.

   For example:

```
http://127.0.0.1:8000/arc/apps/app/3851?embed=true
```

   The header is removed from the dashboard.

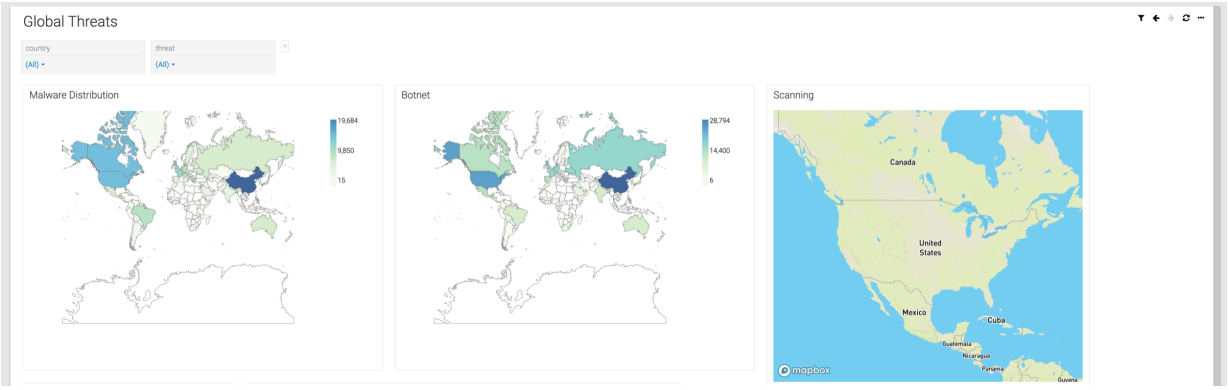

4. You can now embed the dashboard into your web page using an <iframe> tag.

Embedding a jobs status page without header

You can embed a Cloudera Data Visualization Jobs status page within a client web page without displaying the header, which includes the navigation bar and logo.

About this task

Follow these steps to remove the header from the Jobs status page.

Note: Depending on your role, you have access to different information and can perform different actions on this interface.

Procedure

1. Navigate to the Jobs status page.

   You can see that the page has a Cloudera header at the top.

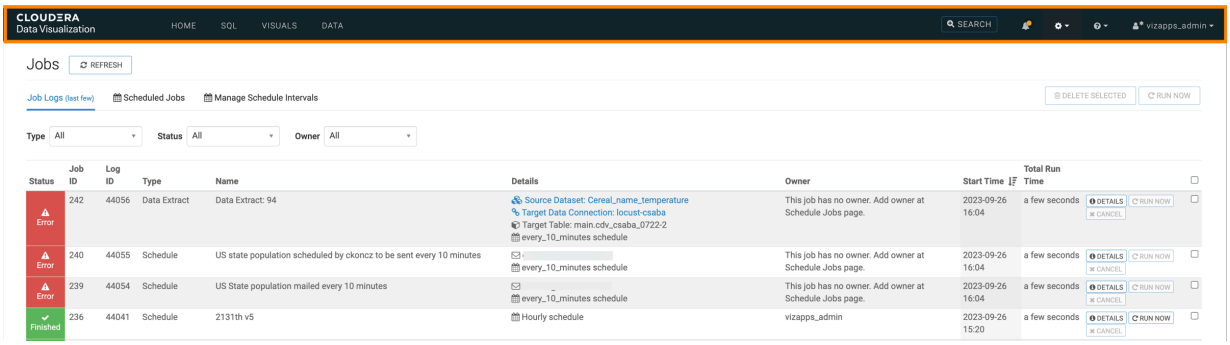

2. Identify the URL of the status page.

   It typically follows this format:

```
http://<appserverip>:<port>/arc/jobs/<jobs_ID>
```

   For example:

```
http://127.0.0.1:8000/arc/jobs/adminlogs
```

3. Add ?embed=true at the end of the URL and click enter.

   For example:

```
http://127.0.0.1:8000/arc/jobs/adminlogs?embed=true
```

   The header is removed from the Jobs status page.

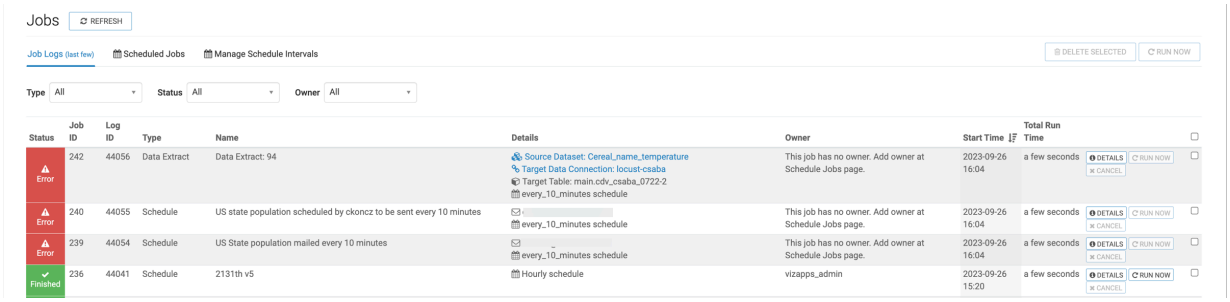

4. You can now embed the report page into your web page using an <iframe> tag.

Cloudera Data Visualization 7.2.9

Events

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating events **4**

Cloning events **8**

Editing events **12**

Deleting events **16**

Creating events

Cloudera Data Visualization allows you to create new events in a dataset.

About this task

To demonstrate how to create new events, the Campaign dataset is used in this example, based on data previously imported into Cloudera Data Visualization from a data file.

To create a new event in the dataset Campaign, follow these steps:

Procedure

1. On the main navigation bar, click DATA.

   The Data view appears open on the Datasets tab.
2. Find the dataset in the list of datasets, either by scrolling or by using search, click on it, and select the Campaign dataset.

   Dataset side navigation appears open at Dataset Detail view.
3. In the side navigation menu, click Events.

   The Events interface appears.

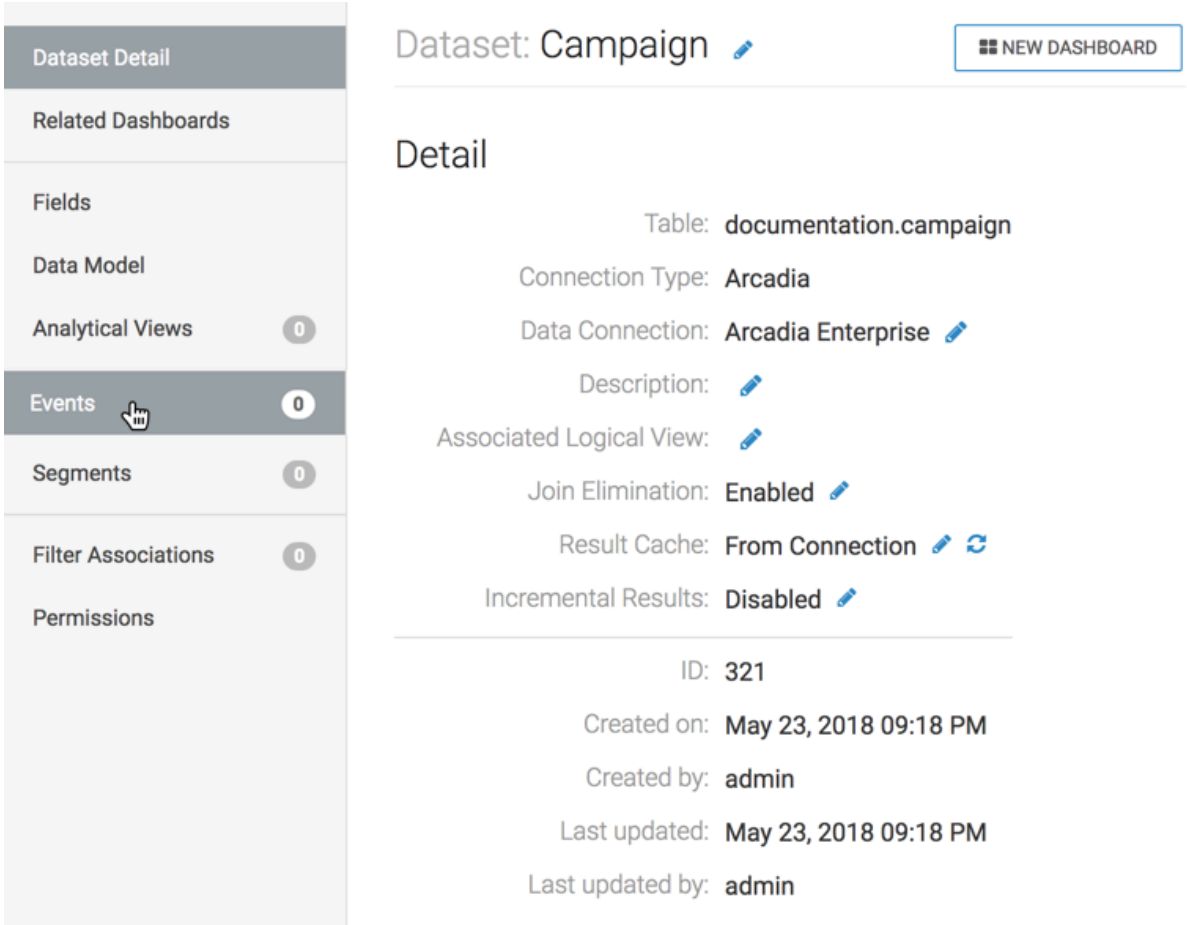

4. In the Events interface, click New Event.
   The Add Event modal window appears.

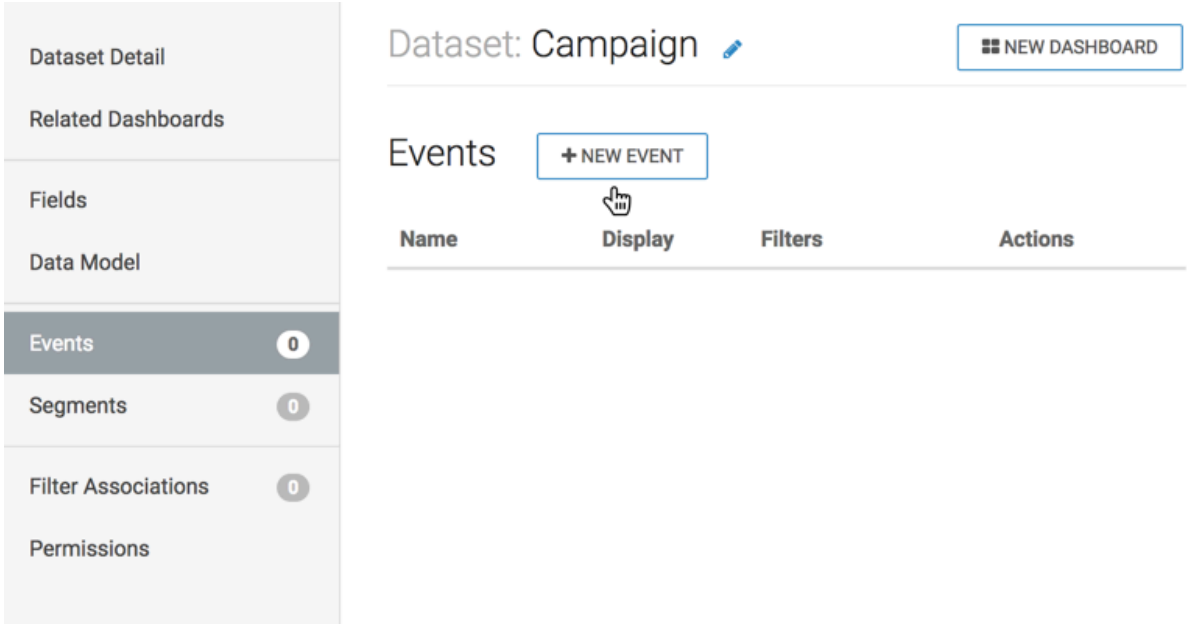

5. In the Add Event modal window, add the following:
   - Under Event Name, enter Start.
   - Under Event Display, enter name.
   - Under Filters, click the text box to open the Filter Expression modal window.

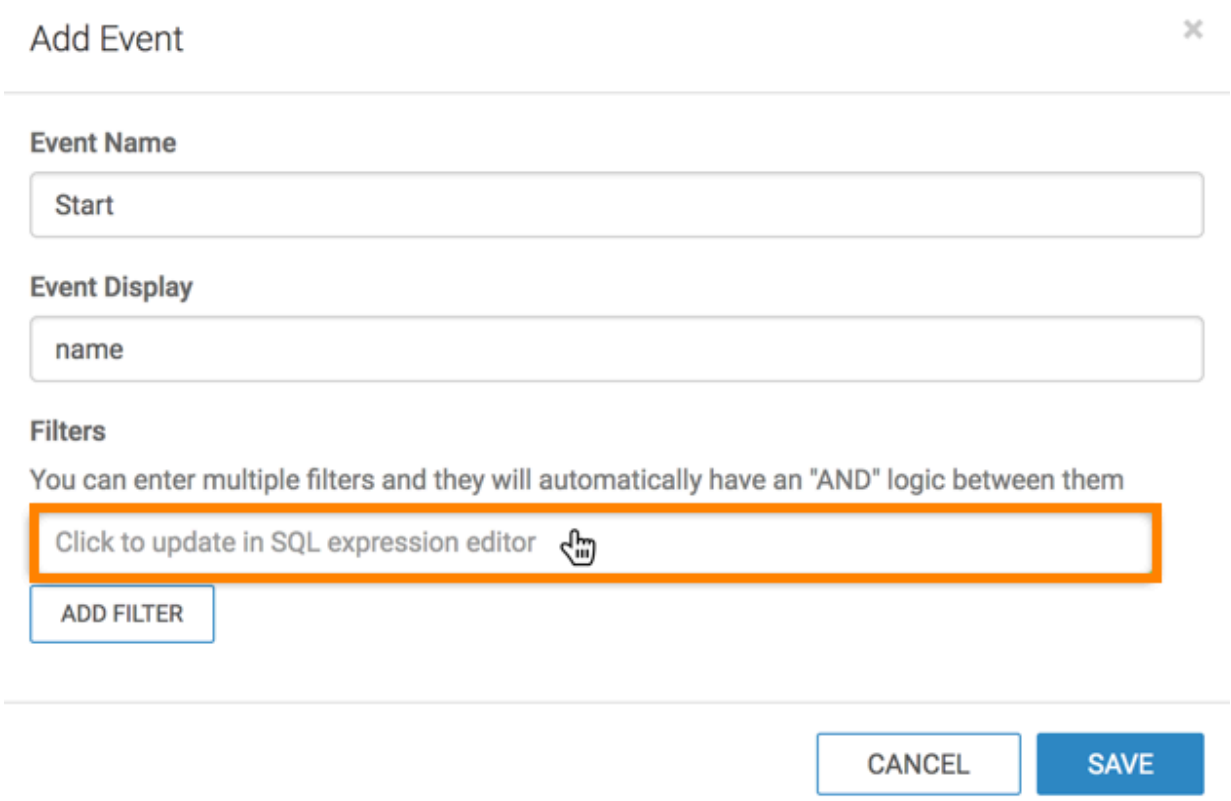

6. In the The Filter Expression modal window, enter the following expression: name='$session_started'

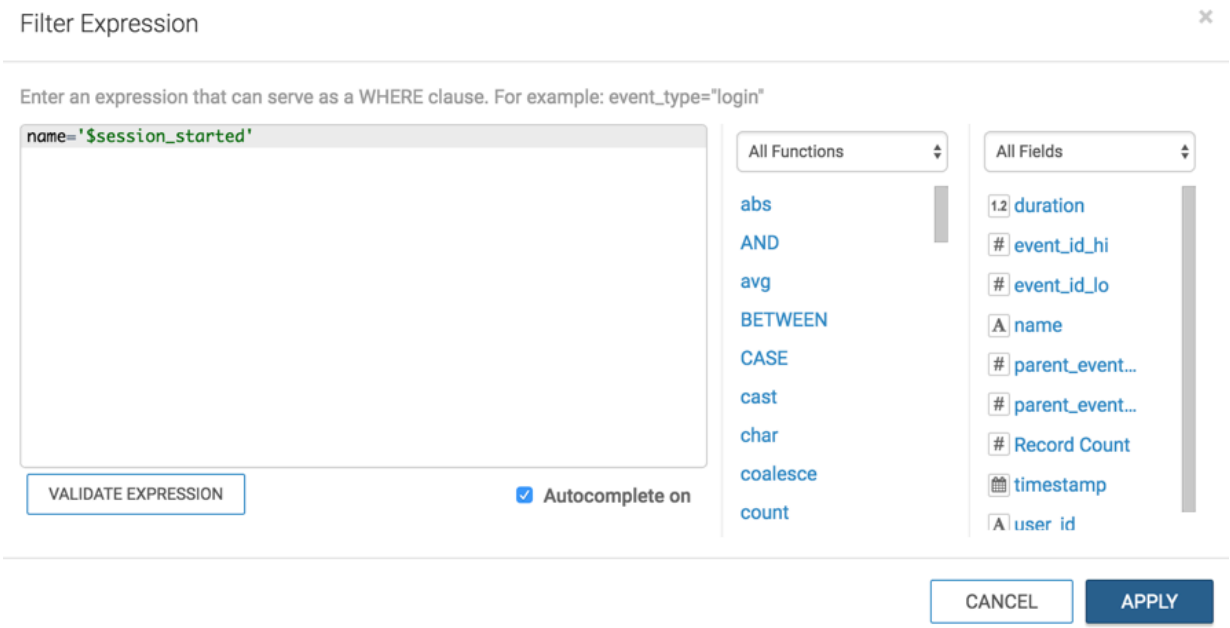

7. Click APPLY to save the filter expression and return to the Add Event modal window.

   In the Add Event window modal, you can see that the Filters field contains the specified expression.

8. Click SAVE.

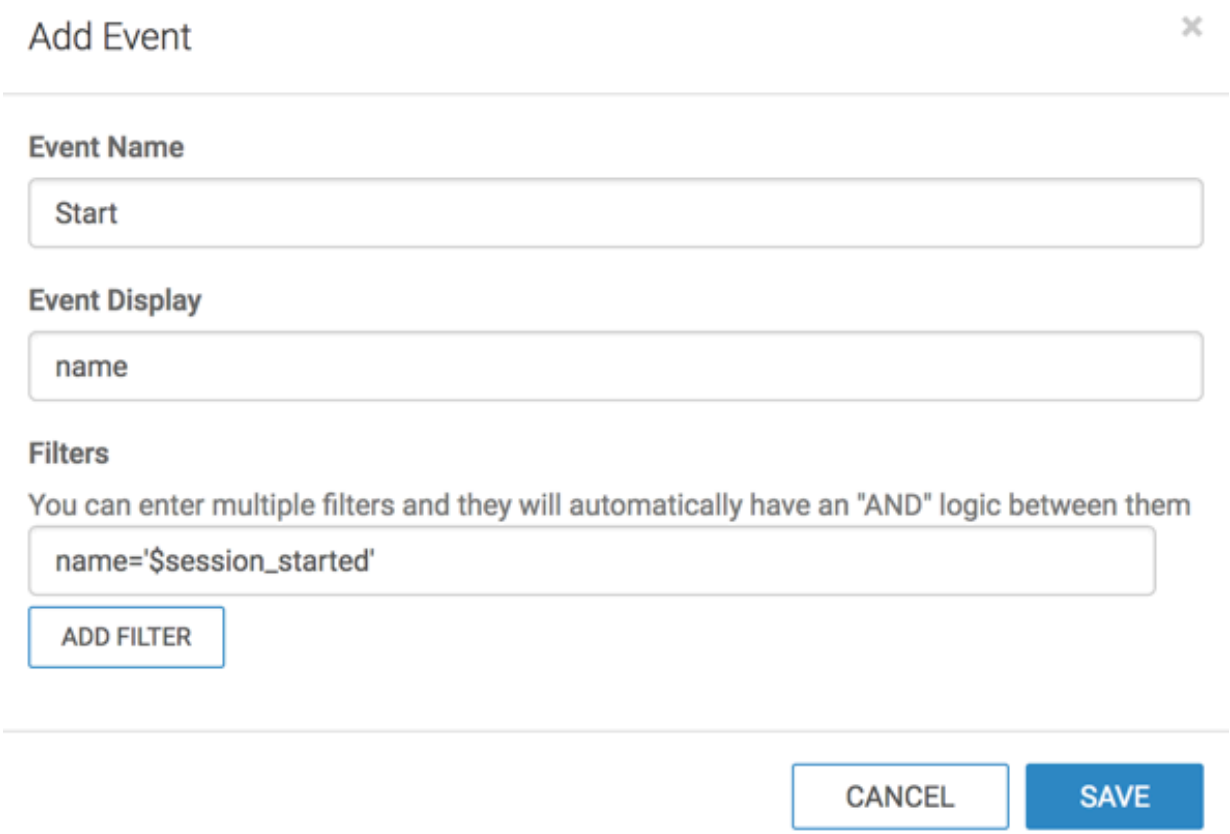

The new event definition appears in the Events interface. The following actions are available for this event: Edit, Clone, and Delete.

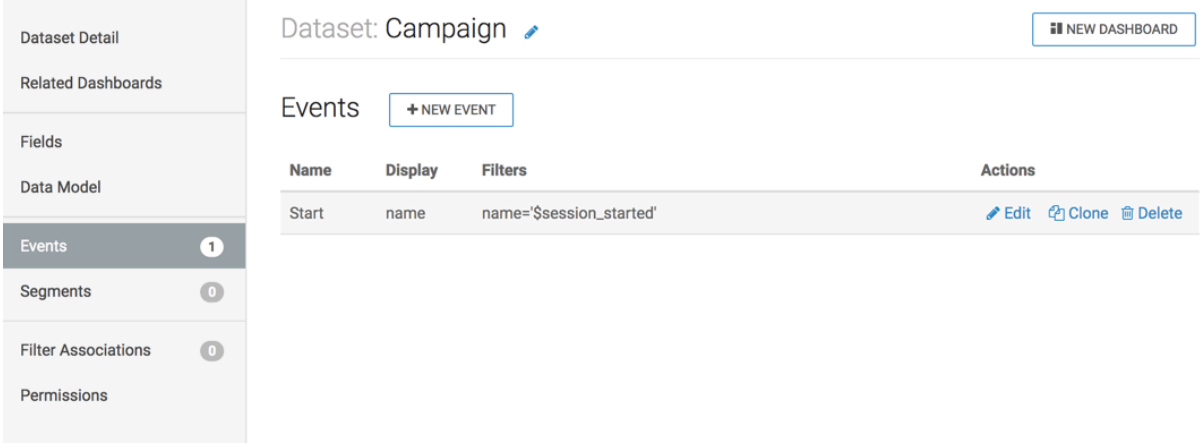

9. Repeat the preceding steps to create six additional events for this dataset, as defined in the following table.

   You already created the event in the first row. Alternatively, you can create new events by cloning existing events. For more information, see *Cloning Events*.

Table 1: Events of the Campaign dataset

Event Name	Event Display	Filters
Start	name	name='$session_started'

Event Name	Event Display	Filters
All application events	name	name!='$session_started' AND name!='$session_ended'
First app use	name	name='$first_used_app'
Not a purchase	name	name!='$purchase'
All non-purchase application events	name	name!='$session_started' name!='$session_ended' name!='$purchase'
Purchase	name	name='$purchase'
End	name	name='$session_ended'

When you finish defining all events for the dataset, your list of events should look similar to the following:

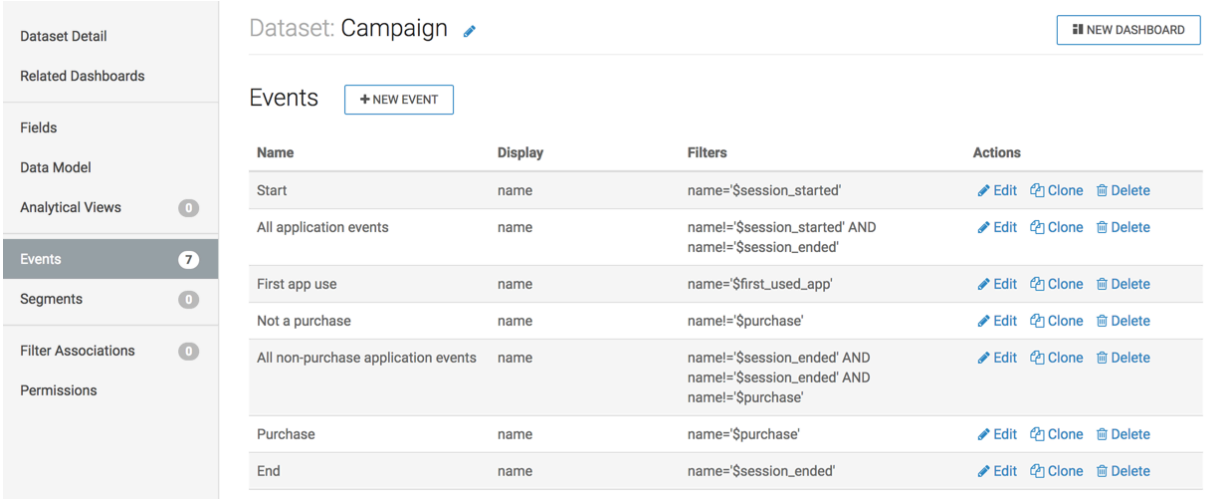

Related Information

Cloning events

Cloning events

Cloudera Data Visualization allows you to clone events in a dataset.

About this task

The following steps demonstrate how to create a new event in the dataset by cloning an existing one. You can use the event Start that was set up in *Creating Events*.

Procedure

1. On the main navigation bar, click DATA.
2. Select Campaign dataset.

   Dataset side navigation appears, open at Dataset Detail view.
3. In the side navigation menu, click Events.

4. In the Events list, find the event to clone, and click Clone.

   In this example, the Start event is cloned.

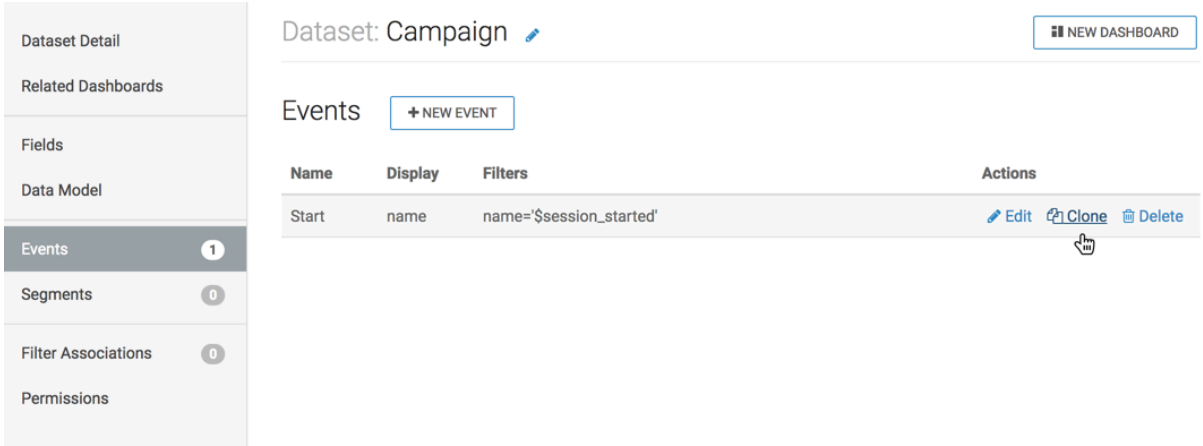

   A populated Add Event modal window appears.

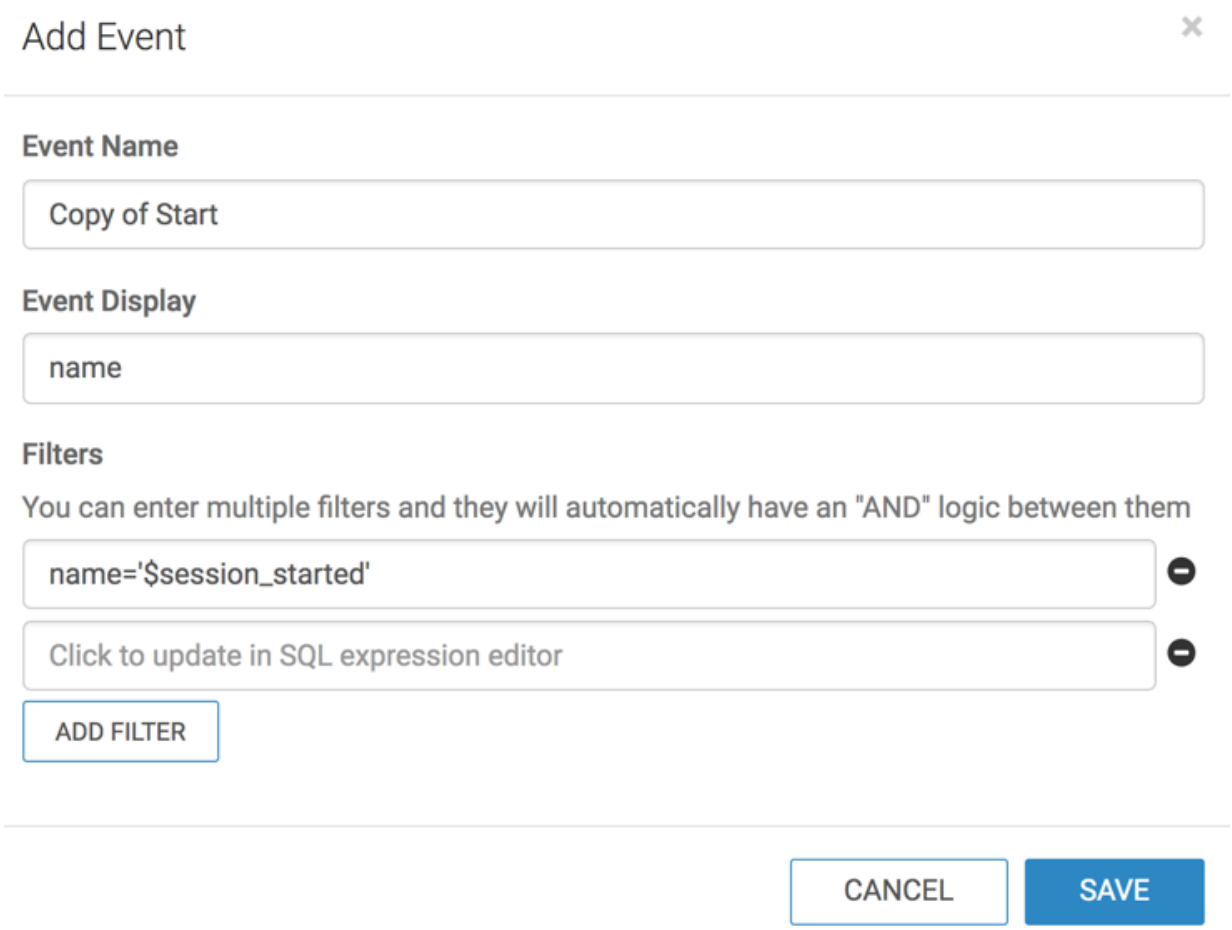

5. In the Add Event modal window, update the following entries:
   - Change Event Name from Copy of Start to End.
   - Under Filters, click the filter expression text box that contains name='$session_started', to open a Filter Expression modal window and edit the expression.

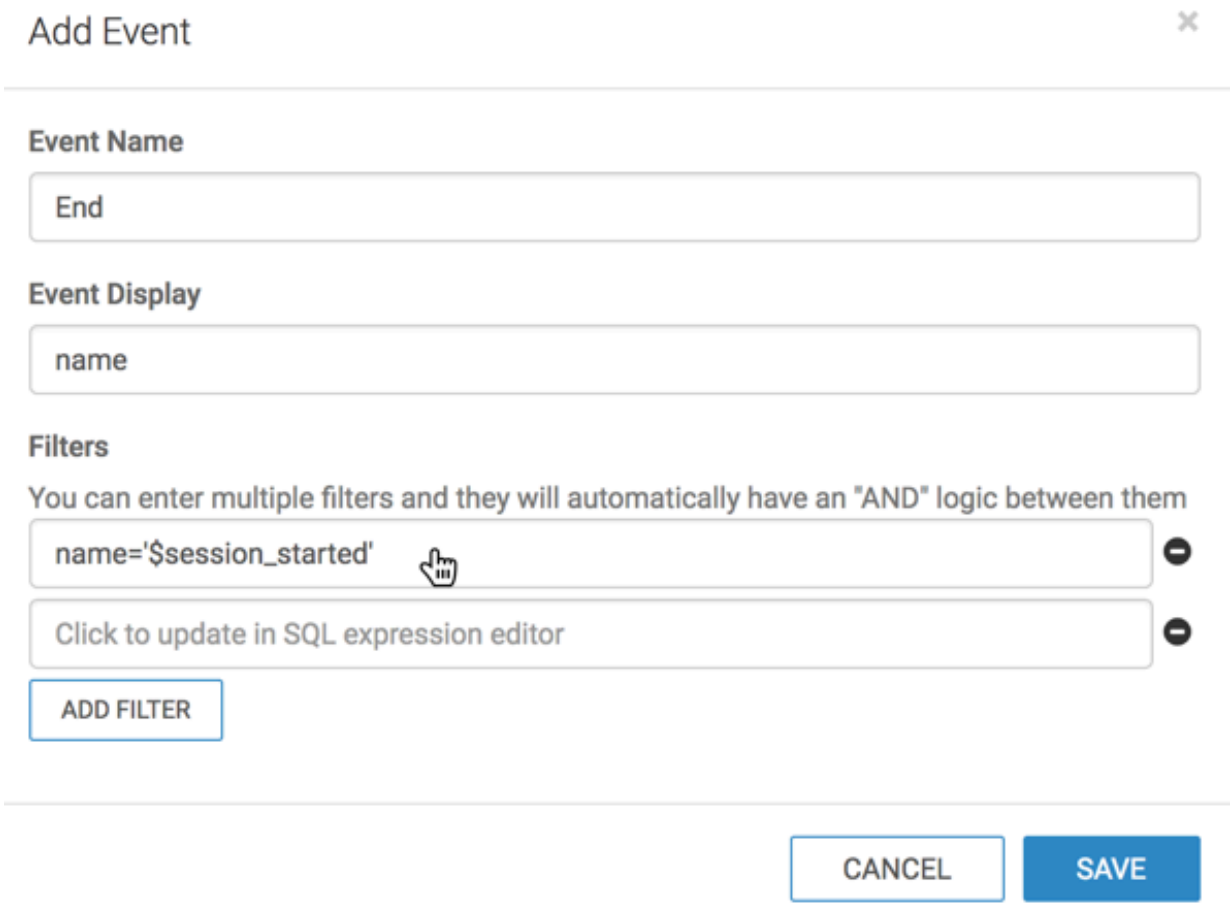

6. In the Filter Expression modal window, enter the following expression: name='$session_ended'
7. Click Validate Expression to test the expression.

8. Click APPLY to save the new filter expression and return to the Add Event modal window.

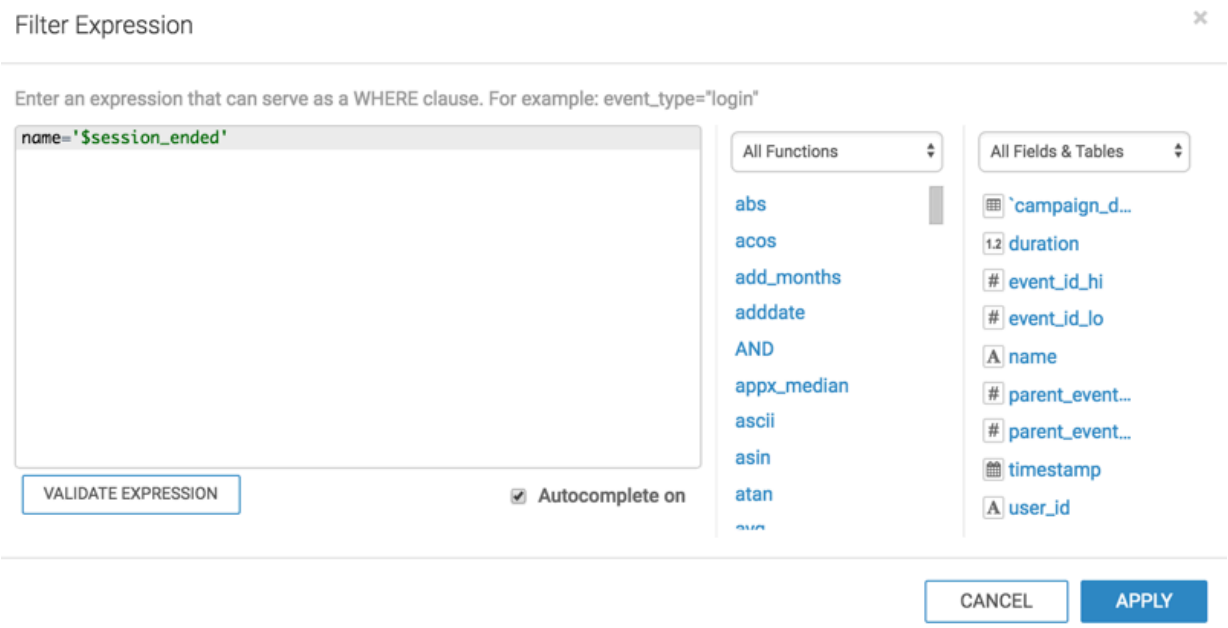

In the Add Event modal window, you can see that the Filters field contains the specified expression.

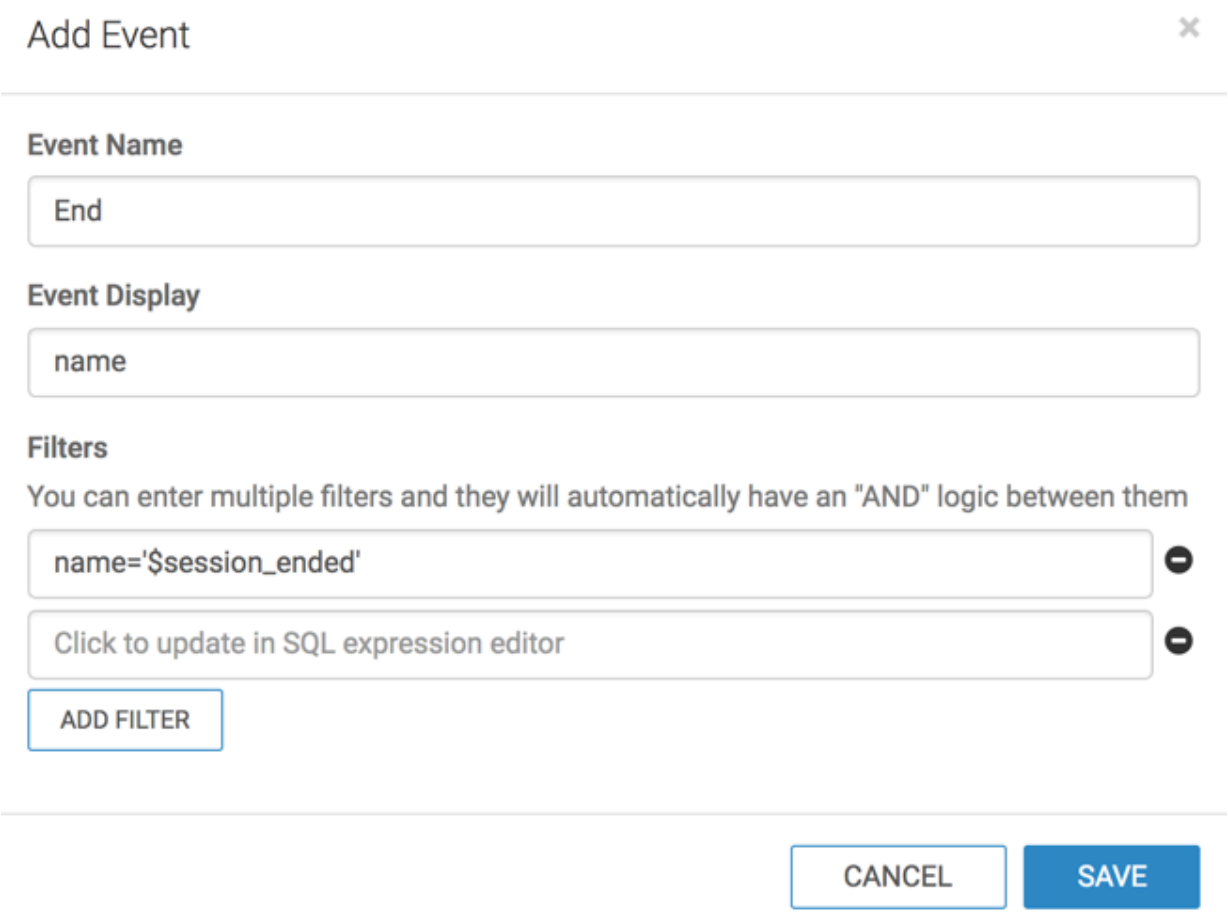

9. Click SAVE.

   The new event definition appears in the Events interface.

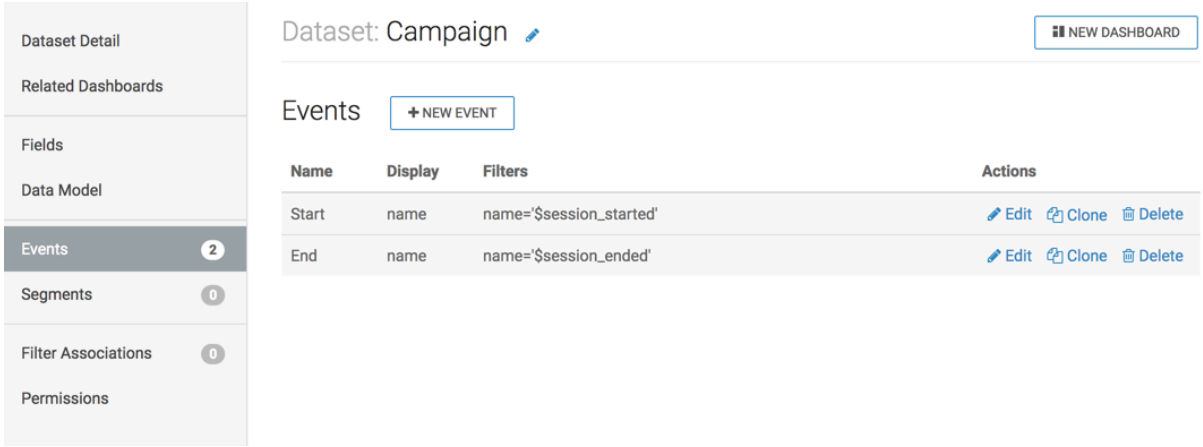

Related Information

Creating events

Editing events

Cloudera Data Visualization allows you to edit the details of an event defined on a dataset.

About this task

The following steps demonstrate how to edit an event in a dataset. You can use the event All application events that was created in *Creating Events*.

Procedure

1. On the main navigation bar, click DATA.
2. Select Campaign dataset.

   Dataset side navigation appears, open at Dataset Detail view.
3. In the side navigation menu, click Events.
4. In the Events list, find the event to edit, and click Edit.

   In this example, an event called All application events is edited.

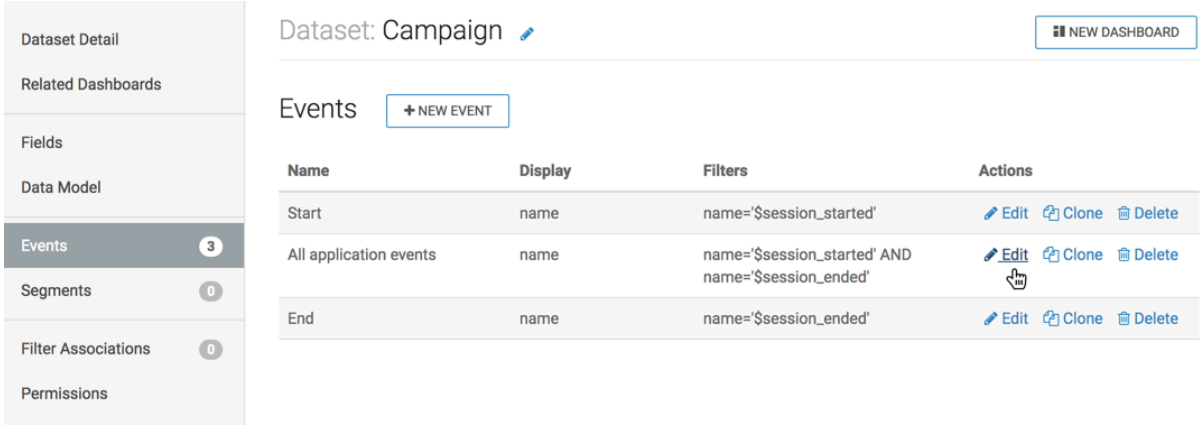

   An Edit Event modal window appears.

5. You can change all the entries in Edit Event modal window.

   In this example:

   - Leave Event Name 'as is'.
   - Leave Event Display 'as is'.
   - Under Filters, click the filter expression text box name='$session_started'AND name='$session_ended, to open a Filter Expression modal window and edit the expression.

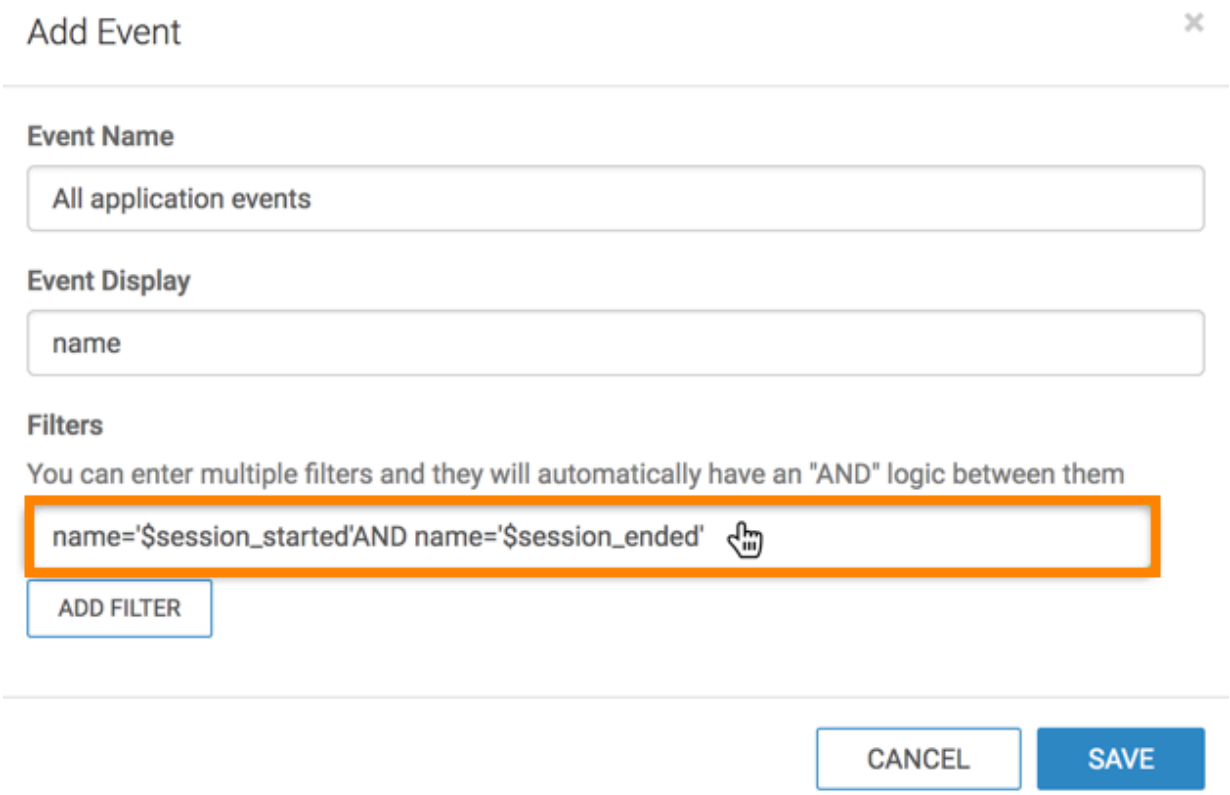

   The Filter Expression modal window appears.

6. In the Filter Expression modal window, enter the following expression:

```
name='$session_started'AND name='$session_ended'AND name='$purchase'
```

7. Click APPLY to save the new filter expression and return to the Edit Event modal window.

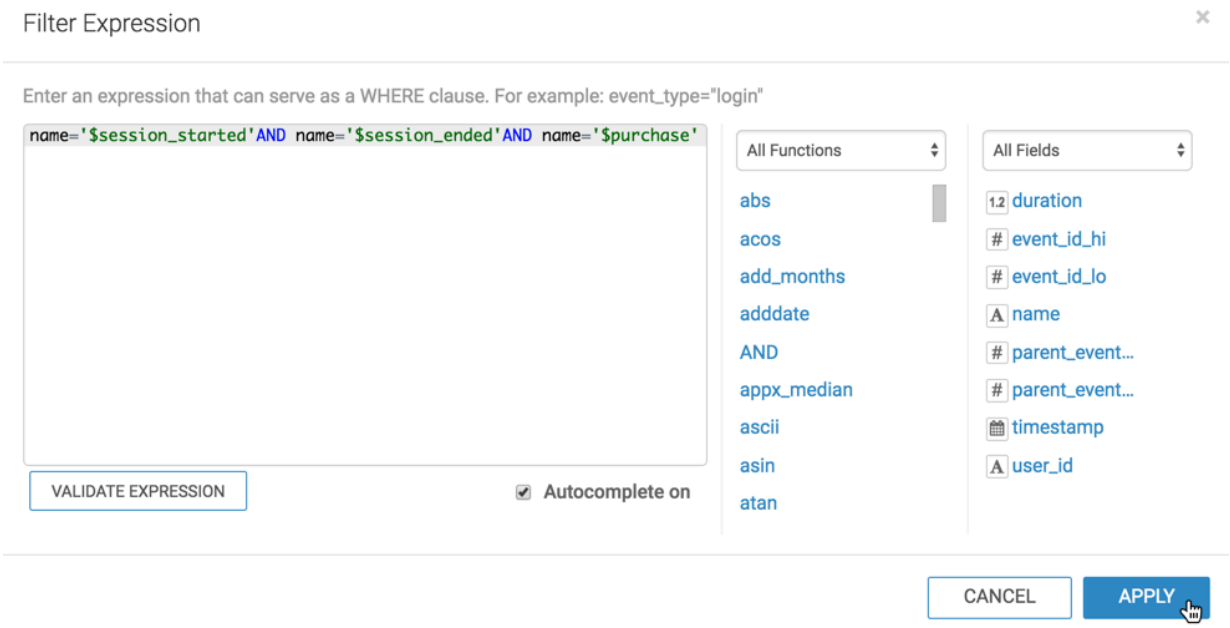

8. In the Edit Event modal window, click SAVE.

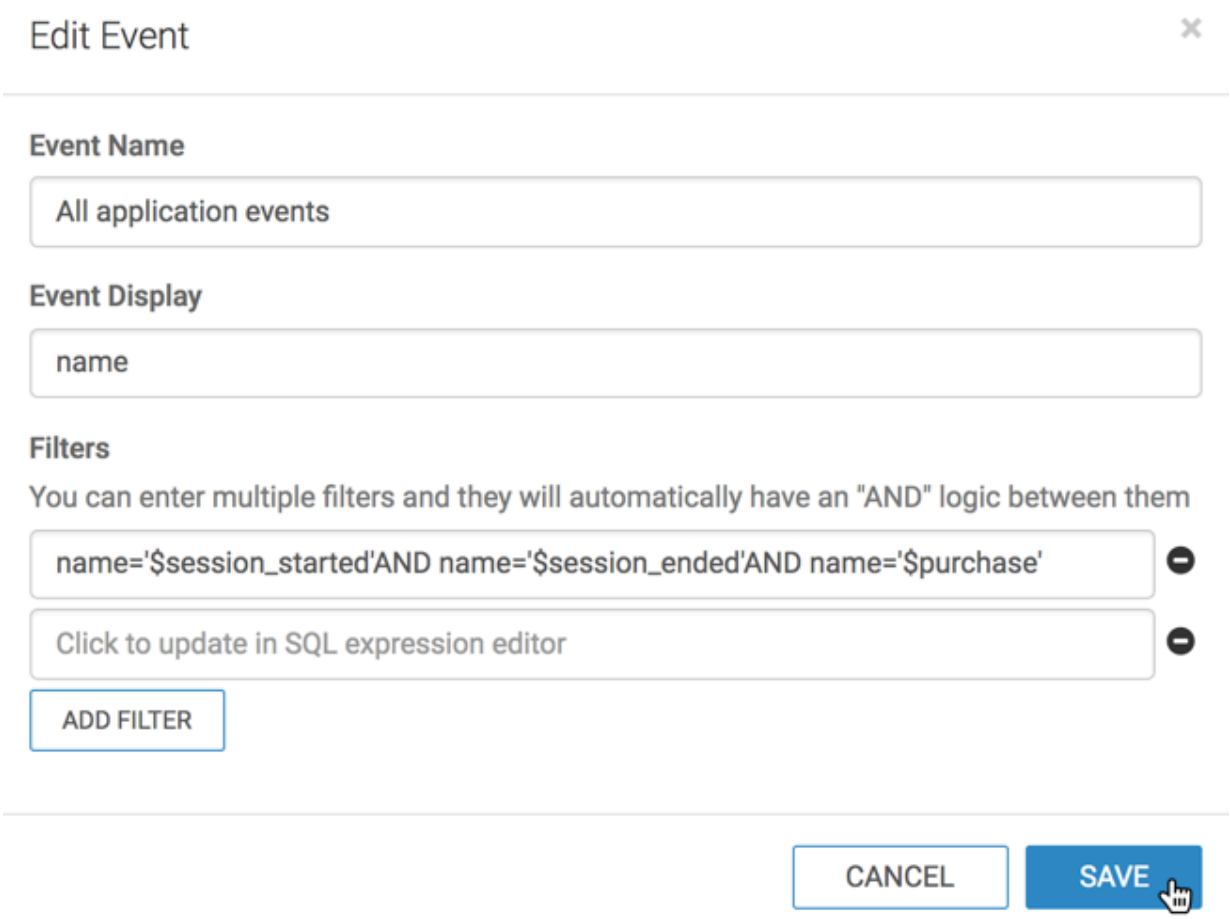

The renamed/edited event appears in the Events interface.

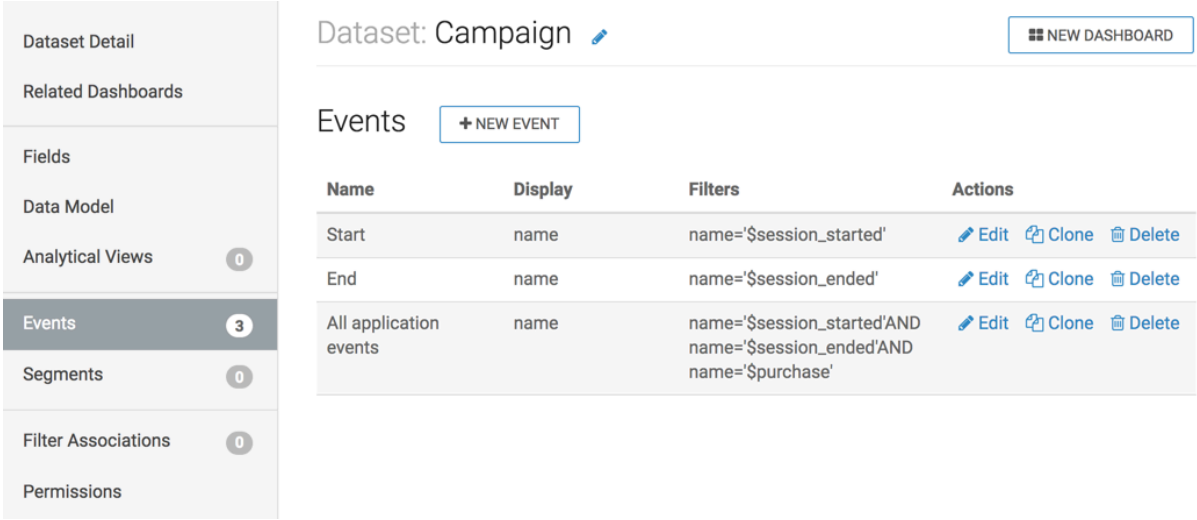

Related Information

Creating events

Deleting events

Cloudera Data Visualization allows you to delete an event defined on a dataset.

About this task

In this example, the event All application events will be deleted.

Procedure

1. On the main navigation bar, click DATA.
2. Select the Campaign dataset.

   Dataset side navigation appears, open at Dataset Detail view.
3. In the side navigation menu, click Events.
4. In the Events list, find the event to delete, and click Delete.

   In this example, the event All application events will be deleted.

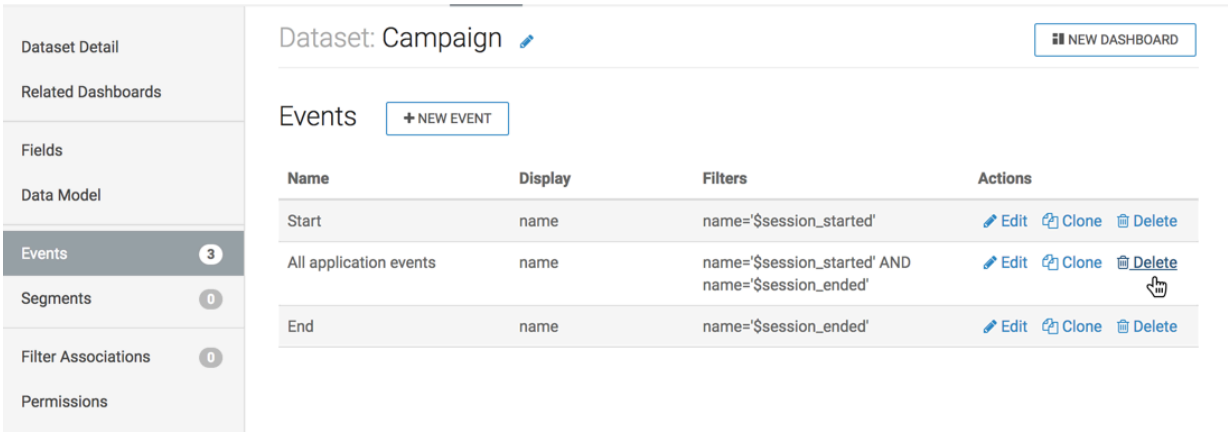

5. In the confirmation window modal, click DELETE.

Results

The event End no longer appears in the Events interface.

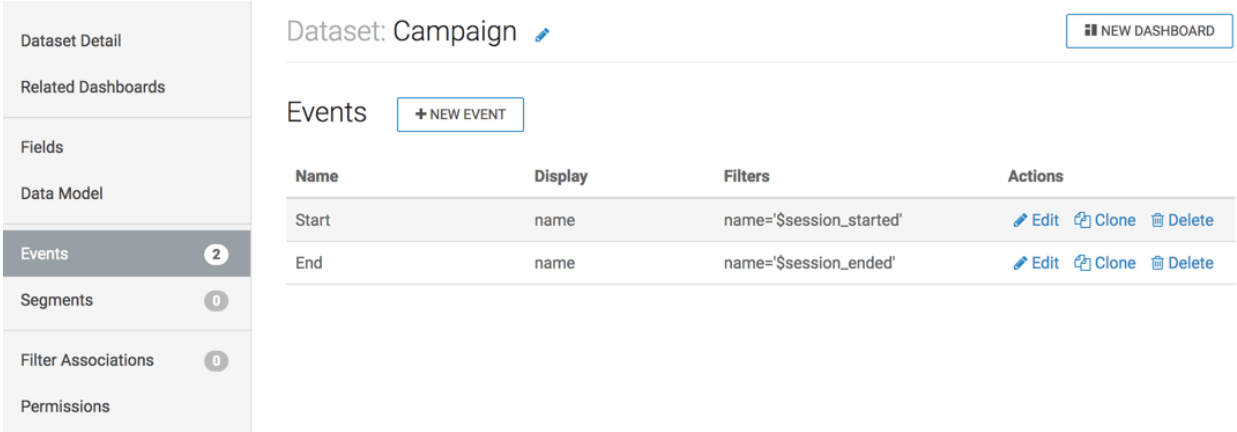

Cloudera Data Visualization 7.2.9

Filter Scope

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Setting explicit scope for filters.....4

Setting dataset scope for filters.....4

Setting app scope for filters.....5

Setting explicit scope for filters

About this task

To enable explicit filter scope, which acts on individual visuals, you must set the Explicit Scope option.

Procedure

1. Open a dashboard in Edit mode that has a filter widget.

   If you do not have a dashboard with filters, add a new filter widget. For instructions, see *Creating dataset filters on dashboards* or *Creating custom filters on dashboards*.
2. Hover over the top right corner of the filter widget and click the Gear icon to configure the filter.

   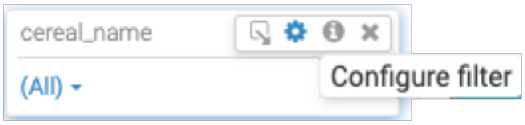

   The Settings modal window appears.
3. Select the Scope tab.
4. Set the Scope of parameters by this filter option to Explicit.
5. Click APPLY.

Related Information

Using parameters with explicit scope

Setting dataset scope for filters

In Cloudera Data Visualization, you must set the dataset scope in your dashboard.

Procedure

1. Open a dashboard in Edit mode that has a filter widget.

   If you do not have a dashboard with filters, add a new filter widget. For instructions, see *Creating dataset filters on dashboards* or *Creating custom filters on dashboards*.
2. Hover over the top right corner of the filter widget and click the Gear icon to configure the filter.

   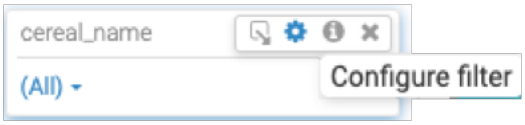

   The Settings modal window appears.
3. Select the Scope tab.
4. Set the filter visibility to Sheet.

5. Set the Scope of parameters by this filter option to All Datasets.

   This option ensures that the dashboard filtering is consistent for all visuals that share this filter's dataset.

   This is the default for new filters.

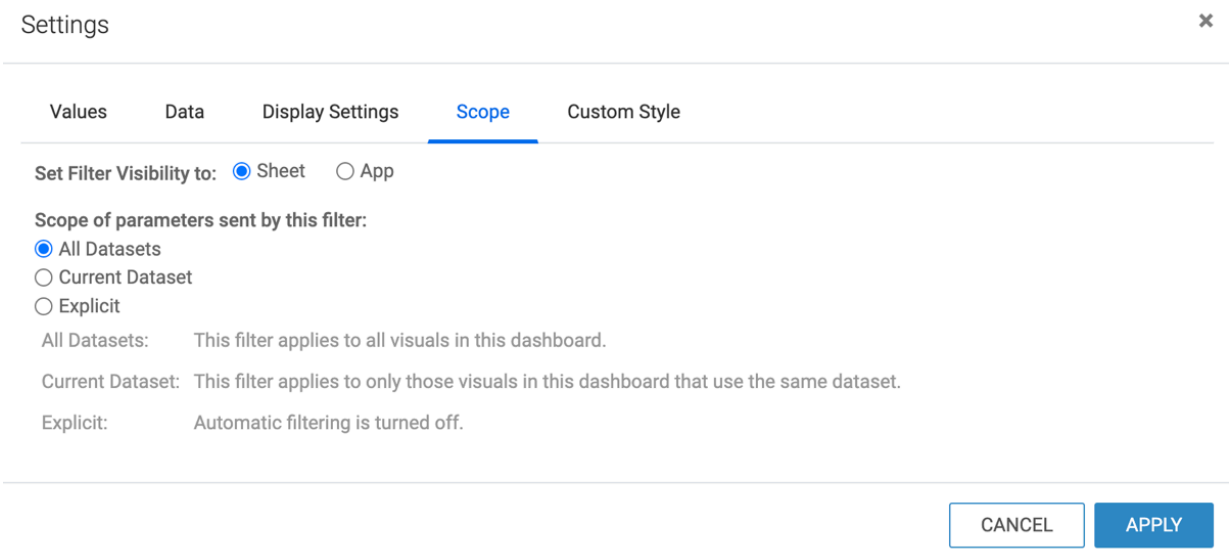

6. Click APPLY.

Related Information

Creating dataset filters on dashboards

Creating custom filters on dashboards

Setting app scope for filters

Cloudera Data Visualization enables you to extend the scope of a filter so that all dashboards in the app are automatically filtered by it

About this task

All apps that contain the dashboard inherit the selections and behavior of this filter. All sheets in a dashboard also inherit the filter behavior. App scope dashboard filters appear only in the filter area, either across the top or on the left side; you cannot move them into the main grid area.

Follow these steps to set App scope filter in a dashboard:

Procedure

1. Open a dashboard in Edit mode that has a filter widget.

   If you do not have a dashboard with filters, add a new filter widget. For instructions, see *Creating dataset filters on dashboards* or *Creating custom filters on dashboards*.

2. Hover over the top right corner of the filter widget, and click the Gear icon to configure the filter.

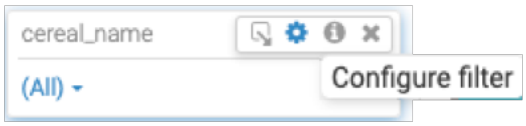

   The Settings modal window appears.

3. Select the Scope tab.
4. Set the filter visibility to App.
5. Select the All Datasets option.

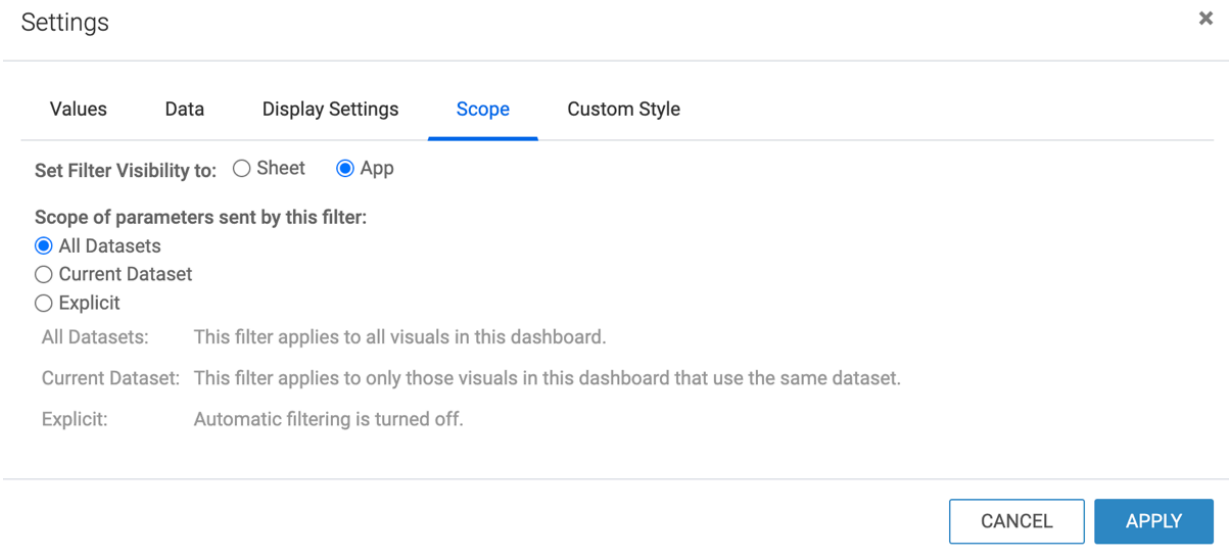

Note: All apps that contain the dashboard inherit the selections and behavior of this filter. All sheets in a dashboard also inherit the filter behavior.

6. Click APPLY.

After you select the App option for visibility, the filter widget on the dashboard displays a gray triangle on the top left corner of the filter. If you hover over the triangle you can see a tooltip that says 'Has App Scope'.

Note: App scope dashboard filters appear only in the filter area, either across the top or on the left side. You cannot move them into the main grid area.

Related Information

Creating dataset filters on dashboards
Creating custom filters on dashboards

Cloudera Data Visualization 7.2.9

Managing Filter Shelves

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Selecting discrete values on filter shelves.....4

Selecting a range of number values on filter shelves.....6

Selecting a string pattern for values on filter shelves.....9

Selecting a range of dates on filter shelves.....11

Selecting values by using an expression on filter shelves.....16

Selecting discrete values on filter shelves

About this task

It is very simple to select discrete values on the filter shelf for all data types: numerical, string, date, and so on.

For selecting discrete numbers in a filter, the example from building *Cross tabulation* is used, where several discrete years of the dataset World Life Expectancy are specified.

Procedure

1. On the Filters shelf, click the Down Arrow on the year field placed there earlier, then click Select values.

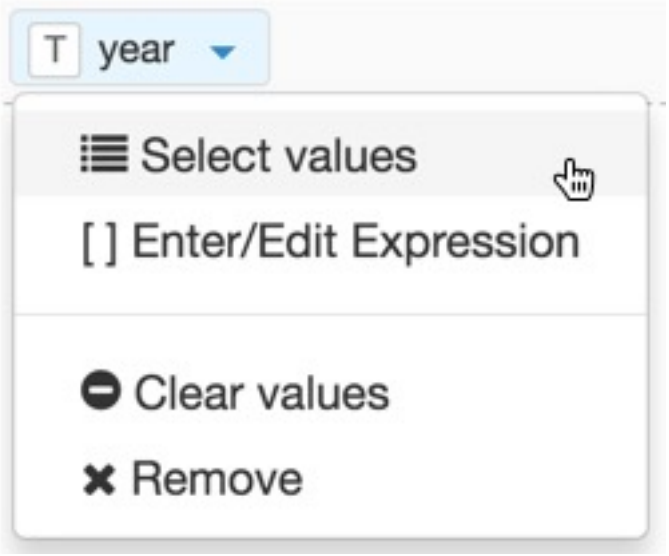

2. In the Filter for year modal window, under the Values tab, select 1900, 1910, 1920, 1930, 1940, 1950, 1960, 1970, 1970, 1980, 1990, 2000, and 2010.

3. Click APPLY.

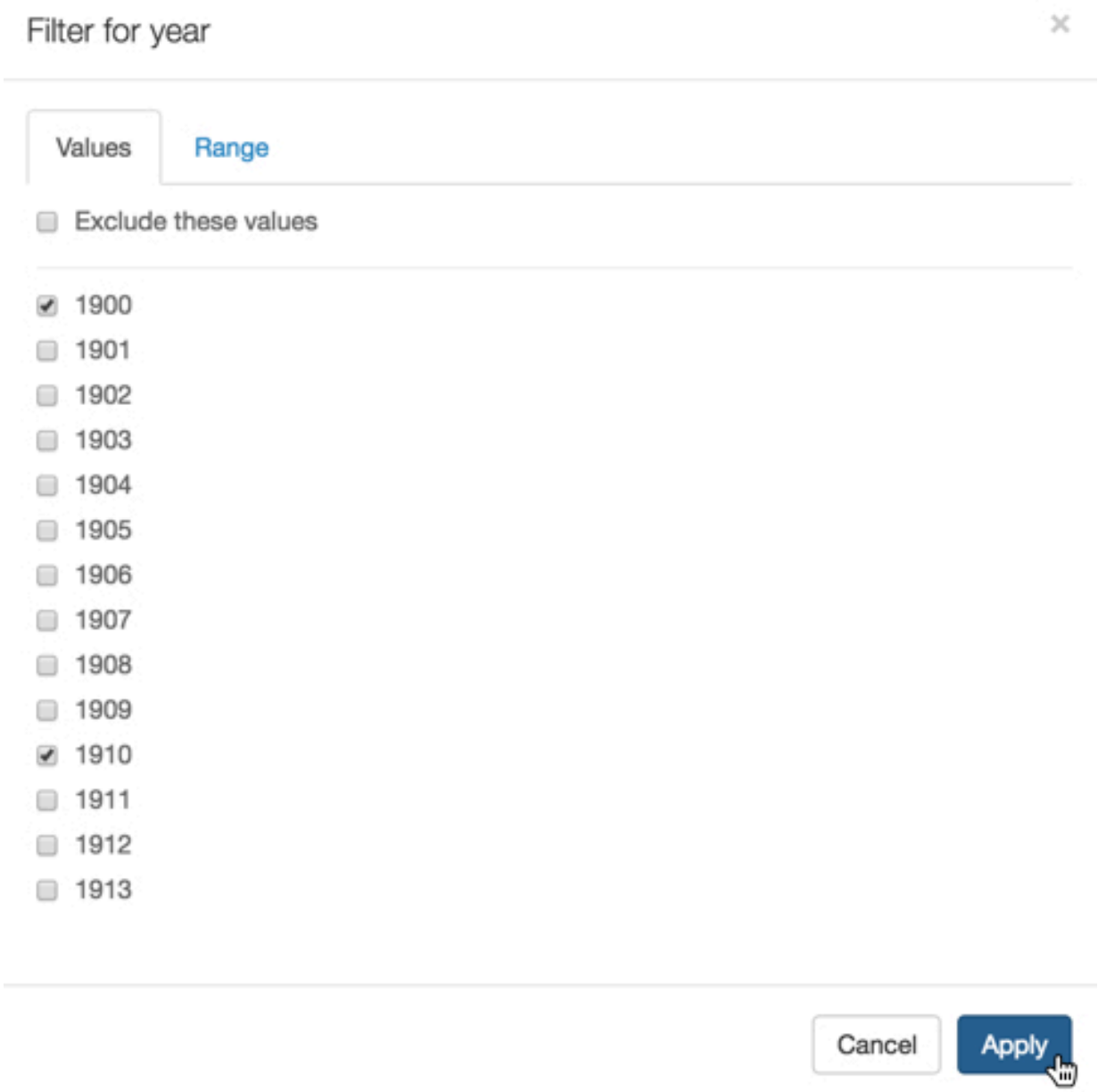

4. Click REFRESH VISUAL.

   The cross tabulation visual appears with the years specified in the previous step.

	country						
	Angola	Cameroon	Chad	Congo	Equatorial Guinea	Gabon	Sao Tome and Principe
year	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)
1900	27.00	28.80	30.90	31.60	29.80	30.60	31.00
1910	27.00	28.80	30.90	31.60	29.80	30.60	31.00
1920	27.00	28.80	30.90	31.60	29.80	30.60	31.00
1930	27.00	28.80	30.90	31.60	29.80	30.60	31.00
1940	27.00	28.80	30.90	31.60	29.80	30.60	35.80
1950	29.20	37.90	35.60	38.30	33.90	36.00	45.50
1960	33.00	41.50	38.00	41.10	36.70	39.60	50.40
1970	37.00	46.10	41.30	43.90	39.80	46.70	55.90
1980	40.20	51.20	44.70	46.10	43.00	54.90	60.60
1990	41.20	53.60	46.40	47.50	46.50	61.40	61.80
2000	45.20	52.00	46.70	46.40	47.70	59.70	63.30
2010	50.70	53.70	49.80	49.00	51.50	62.30	65.90

   Alternatively, with numerical data, you can specify a range of numerical values. For more information, see Selecting a range of number values on filter shelves on page 6. You can also choose the desired combination of filters by specifying an expression. For more information, see Selecting values by using an expression on filter shelves on page 16.

Related Information

Cross tabulation

Selecting a range of number values on filter shelves

About this task

For selecting a range of numerical values in a filter shelf, let's use the example from building *Cross tabulation*, where we specify several discrete years of the dataset World Life Expectancy.

Procedure

1. On the Filters shelf, click Down Arrow on the year field placed there earlier, then click Select values.

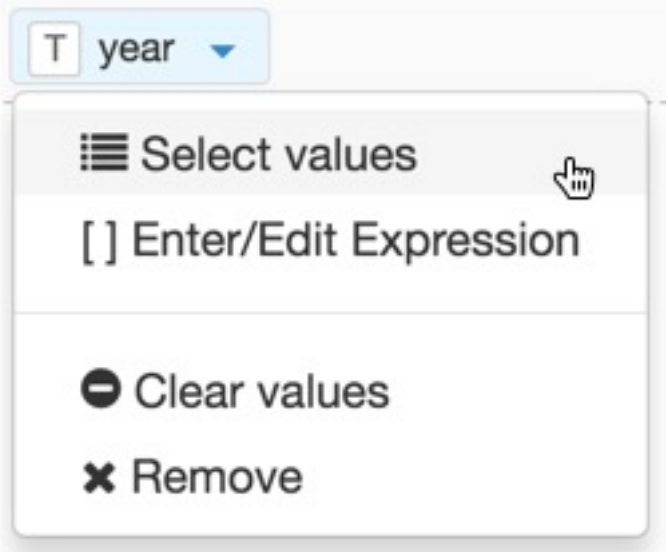

2. In the Filter for year modal window, click the Range tab, and then slide the range upper and lower values to select the range of values used by the visual.

   By default, the lowest value of the range is at the extreme left of the slide control (set to >=), and the highest value is on the extreme right (set to <=). You can change the end-point specifications of the range.

   The valid operators for the bottom of the range are:

   - Greater than or equal to (>=)
   - Greater than (>)
   - Equal to (=)
   - Not equal to (!=)
   - Do not use minimum range

   The valid operators for the top of the range are:

   - Less than or equal to (<=)
   - Less than (<)
   - Do not use maximum range

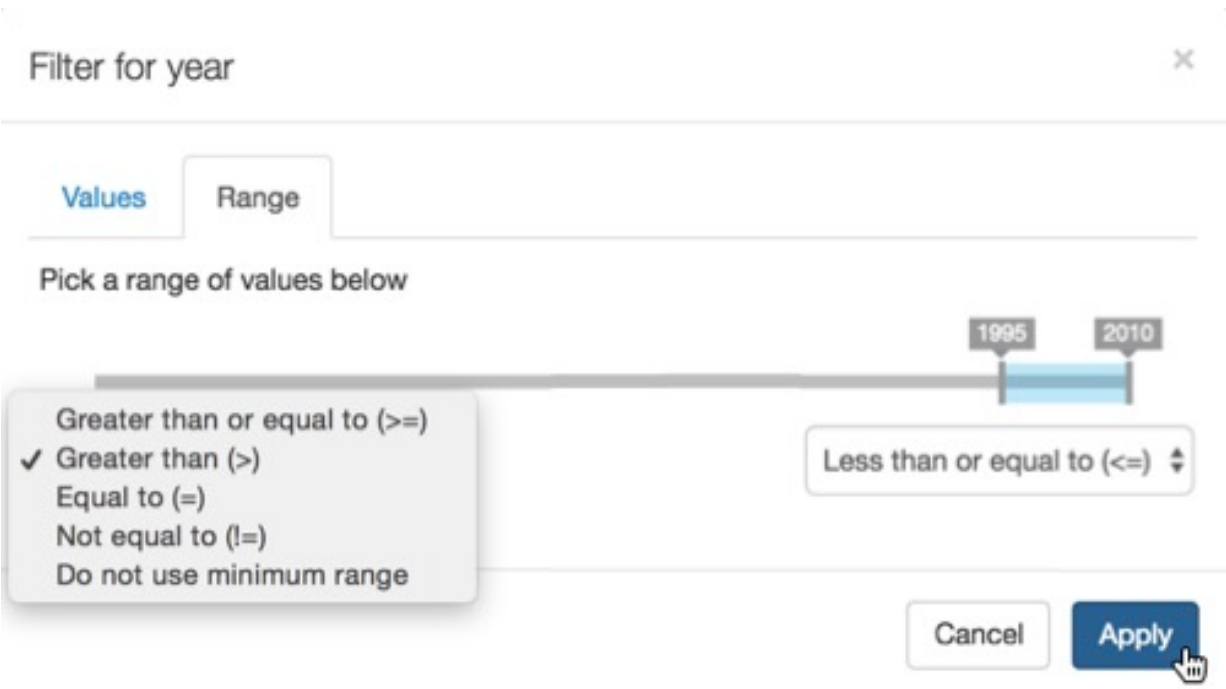

3. After clicking Refresh Visual, the cross tabulation visual appears. Note the years we specified in the previous step.

	country						
	Angola	Cameroon	Chad	Congo	Equatorial Guinea	Gabon	Sao Tome and Principe
year	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)
1996	42.50	53.00	46.50	46.20	47.70	60.90	62.80
1997	43.10	52.80	46.50	46.10	47.70	60.70	62.90
1998	43.70	52.50	46.60	46.00	47.70	60.30	63.00
1999	44.50	52.20	46.60	46.10	47.70	60.00	63.20
2000	45.20	52.00	46.70	46.40	47.70	59.70	63.30
2001	46.00	51.80	46.80	46.70	47.80	59.60	63.50
2002	46.70	51.70	47.00	47.10	47.90	59.50	63.80
2003	47.40	51.70	47.10	47.40	48.10	59.50	64.00
2004	48.00	51.70	47.40	47.60	48.40	59.70	64.30
2005	48.60	51.90	47.70	47.90	48.80	60.10	64.60
2006	49.00	52.20	48.10	48.00	49.30	60.50	65.00
2007	49.50	52.50	48.50	48.20	49.80	60.90	65.30
2008	49.90	52.90	48.90	48.50	50.30	61.40	65.50
2009	50.30	53.30	49.30	48.70	50.90	61.90	65.70
2010	50.70	53.70	49.80	49.00	51.50	62.30	65.90

Related Information

Cross tabulation

Selecting a string pattern for values on filter shelves

About this task

For selecting a range of numerical values in a filter shelf, let's use the example from building *Cross tabulation*, where we specify several discrete years of the dataset World Life Expectancy.

Procedure

1. On the Filters shelf, click Down Arrow on the year field placed there earlier, then click Select values.

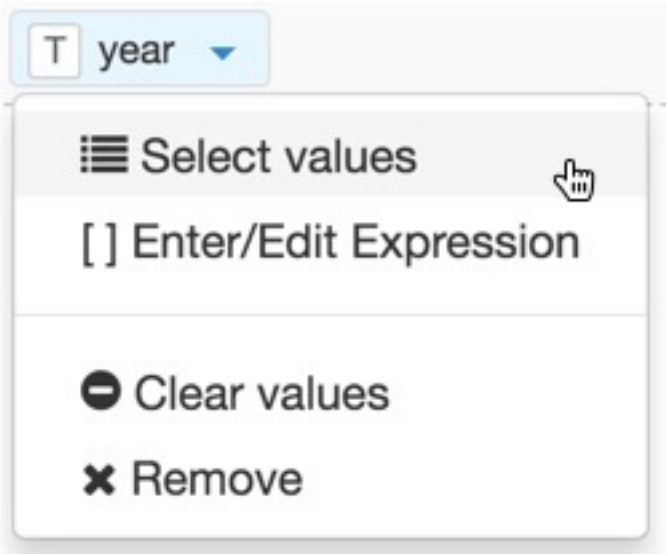

2. In the Filter for country modal window, click the Pattern tab, and then enter the necessary information.

   The patterns can be matched in the following manner:

   - Starts With
   - Ends With
   - Contains

3. To select all countries that start with Z, select Starts With, enter Z in the text box, and click Apply.

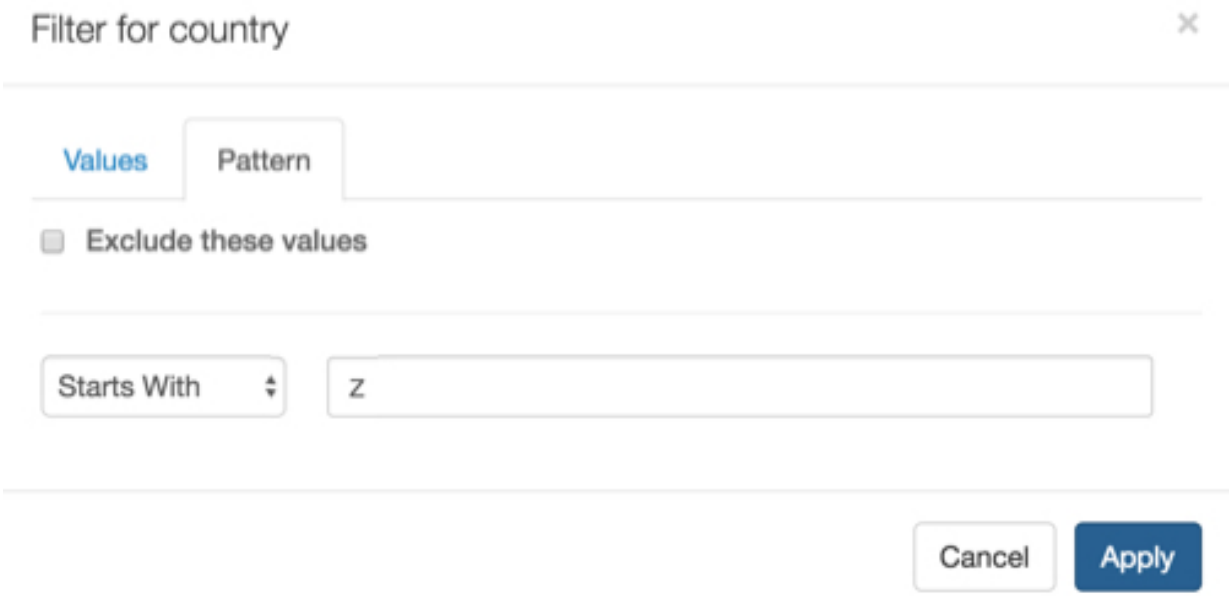

4. After clicking Refresh Visual, the cross tabulation visual appears. Note that only two countries, Zambia and Zimbabwe, match the filter conditions.

	country	
	Zambia	Zimbabwe
year	avg(life_expectancy)	avg(life_expectancy)
2001	42.50	43.30
2002	43.50	43.00
2003	44.50	42.90
2004	45.80	43.20
2005	47.20	44.00
2006	48.60	45.20
2007	50.10	46.80
2008	51.60	48.90
2009	53.10	51.20
2010	54.50	53.70

5. To select all countries that end with 'stan', select Ends With, enter stan in the text box, and click Apply.

	country						
	Afghanistan	Kazakhstan	Kyrgyzstan	Pakistan	Tajikistan	Turkmenistan	Uzbekistan
year	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)	avg(life_expectancy)
2001	55.30	63.80	66.30	64.10	63.90	64.00	67.10
2002	55.70	64.20	66.50	64.40	64.30	64.20	67.20
2003	56.10	64.60	66.50	64.70	64.70	64.30	67.20
2004	56.60	64.90	66.60	64.90	65.10	64.30	67.30
2005	57.10	65.20	66.60	65.20	65.40	64.40	67.40
2006	57.60	65.40	66.70	65.40	65.80	64.50	67.50
2007	58.10	65.60	66.70	65.60	66.10	64.60	67.50
2008	58.60	65.80	66.80	65.80	66.40	64.70	67.60
2009	59.10	66.00	66.90	65.90	66.60	64.90	67.80
2010	59.60	66.10	67.10	66.10	66.80	65.00	67.90

Related Information

Cross tabulation

Selecting a range of dates on filter shelves

About this task

For selecting a range of dates in a filter shelf, let's use a visual built in *Creating joins*.

Procedure

1. Create a new field crs_dep_timestamp directly on the dataset.

   It combines the fl_date and crs_dep_time data into a new timestamp field that has the following definition:

```
concat(substr(cast([fl_date] as string),1,11),
if(length(cast([crs_dep_time] as string))=3,
concat('0',substr(cast([crs_dep_time] as string),1
,1),':',
substr(cast([crs_dep_time] as string),2,2),':00'),
concat(' ',substr(cast([crs_dep_time] as string)
,1,2),':',
substr(cast([crs_dep_time] as string),3,2),':00')
))
```

2. Create a new cross tabulation visual on the dataset Flight Delays with the following configuration:
   - On the X shelf, add the dimension unique_carrier. Alias the field as Airline.
   - On the Y shelf, add the dimension fl_date.

     Click the filed name, and under the Field Properties, change the Date/Time Functions to Date.

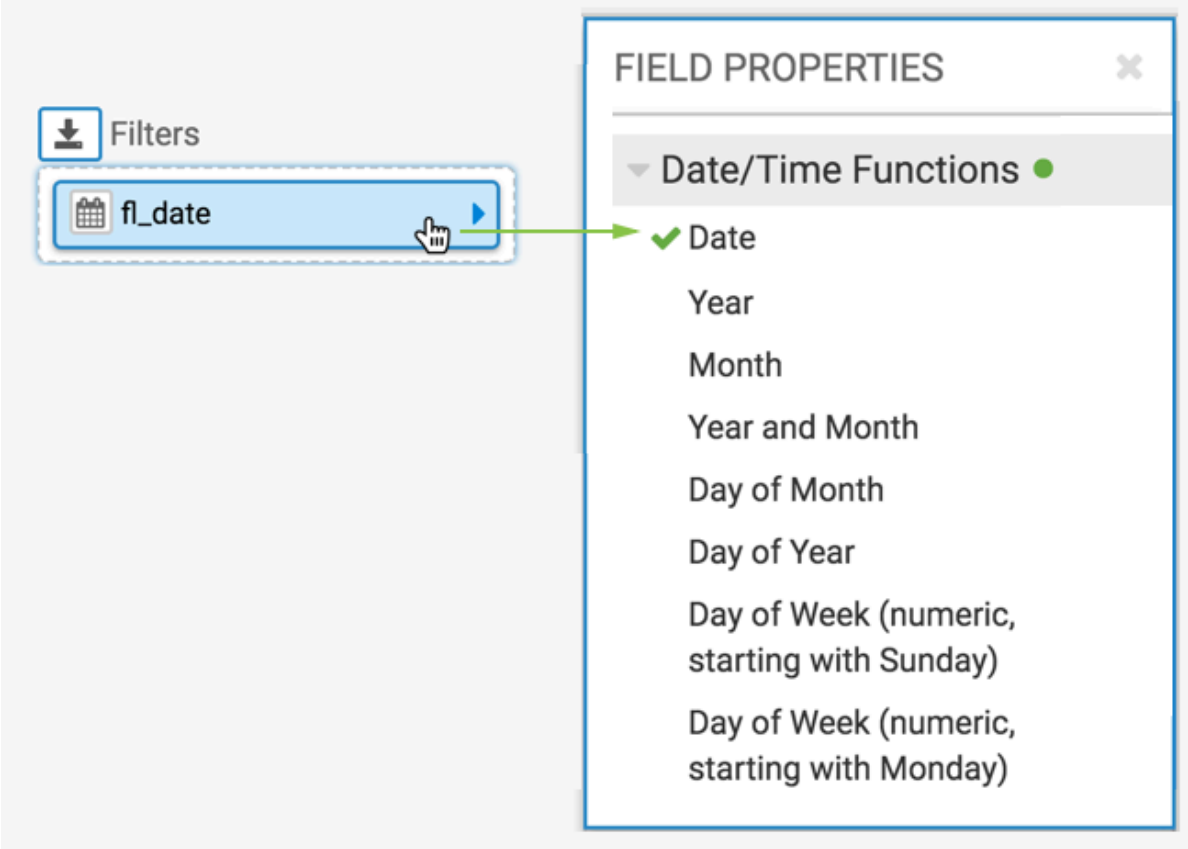

   - On the Measures shelf, add the measure dep_delay, and ensure that it uses the avg(dep_delay) aggregate. Alias the field as Delay.
   - On the Filters shelf, add crs_dep_timestamp.

3. Click Refresh Visual. Note that all the date values in the dataset, starting with 2015-01-01 and ending with 2015-02-28.

	Airline													
	AA	AS	B6	DL	EV	F9	HA	MQ	NK	OO	UA	US	VX	WN
Date	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay
2015-01-01	18.0	1.21	5.62	3.42	5.48	14.7	4.22	13.9	9.95	13.0	14.9	3.60	1.56	10.5
2015-01-02	21.4	8.04	2.08	6.30	6.99	16.6	22.1	18.5	15.5	14.3	18.2	3.10	5.38	16.5
2015-01-03	29.9	9.12	20.6	14.3	32.1	52.4	22.3	34.3	30.9	25.4	29.1	15.3	6.31	27.6
2015-01-04	36.1	10.9	34.4	21.3	37.6	65.3	1.58	56.3	36.7	31.2	38.4	17.0	16.8	31.6
2015-01-05	25.6	6.35	21.1	6.99	29.5	31.9	6.02	49.1	47.0	26.4	23.6	7.13	7.40	18.7
2015-01-06	16.0	8	33.4	13.6	21.8	26.1	-1.59	35.3	30.7	15.8	25.5	26.1	11.4	29.0
2015-01-07	14.3	5.76	15.7	6.01	14.5	35.2	-1.81	32.0	23.7	16.3	25.7	6.42	8.62	12.9
2015-01-08	17.9	5.10	6.73	11.5	19.8	18.1	1.55	33.1	17.5	20.7	29.9	8.71	1.89	13.9
2015-01-09	14.1	0.289	27.8	11.8	17.1	33.4	2.63	28.2	20.2	22.8	28.6	5.79	7.38	7.60
2015-01-10	12.8	-1.79	7.32	4.49	5.98	19.9	-0.592	12.3	12.7	11.1	16.4	3.84	0.642	6.97
2015-01-11	15.3	8.38	7.89	4.63	11.4	20.1	2.37	21.8	10.9	17.7	18.1	6.99	23.0	18.8
2015-01-12	11.1	3.20	13.2	27.8	25.4	21.7	-2.71	31.0	33.2	22.5	19.7	9.73	18.6	16.8
2015-01-13	5.83	-0.420	0.405	3.93	4.55	3.34	1.12	22.3	7.18	16.4	6.78	2.57	-0.669	4.91
[illegible]	[illegible]	-0.651	[illegible]	11.7	14.1	[illegible]	1.79	[illegible]	14.6	10.2	[illegible]	[illegible]	5.31	[illegible]
2015-02-19	4.98	3.02	19.7	11.2	9.85	26.1	2.34	19.3	12.8	12.2	15.5	9.62	11.7	11.1
2015-02-20	7.48	5.41	15.8	15.5	11.2	23.2	1.52	31.7	13.0	8.58	15.7	13.9	8.91	9.28
2015-02-21	12.1	4.08	22.9	17.6	8.31	35.8	2.21	17.8	19.3	9.07	21.7	14.2	-1.16	11.3
2015-02-22	21.4	2.29	50.1	18.6	18.5	72.5	5.93	20.1	28.8	12.8	21.2	28.3	9.94	11.8
2015-02-23	14.0	3.53	13.5	10.3	17.7	30.1	-1.80	25.8	48.7	15.7	13.7	8.96	6.98	12.2
2015-02-24	10.4	-3.49	3.68	35.3	23.8	16.1	-4.00	34.1	23.3	6.10	8.99	10.2	-1.05	11.2
2015-02-25	11.9	-1.57	14.7	16.6	11.9	31.8	-1.15	17.9	14.3	8.63	13.4	3.09	-2.35	9.58
2015-02-26	22.9	1.28	12.4	12.4	26.8	28.3	0.721	39.9	23.2	21.4	24.9	14.2	4.98	25.3
2015-02-27	34.1	3.66	10.8	8.20	8.01	12.2	3.88	30.9	15.9	16.8	16.8	7.91	22.5	12.4
2015-02-28	37.3	1.01	15.8	6.20	5.27	54.8	2.31	21.7	31.4	10.8	17.0	9.51	5.53	9.57

4. On the Filters shelf, click the crs_dep_timestamp field.
5. Click Select values.

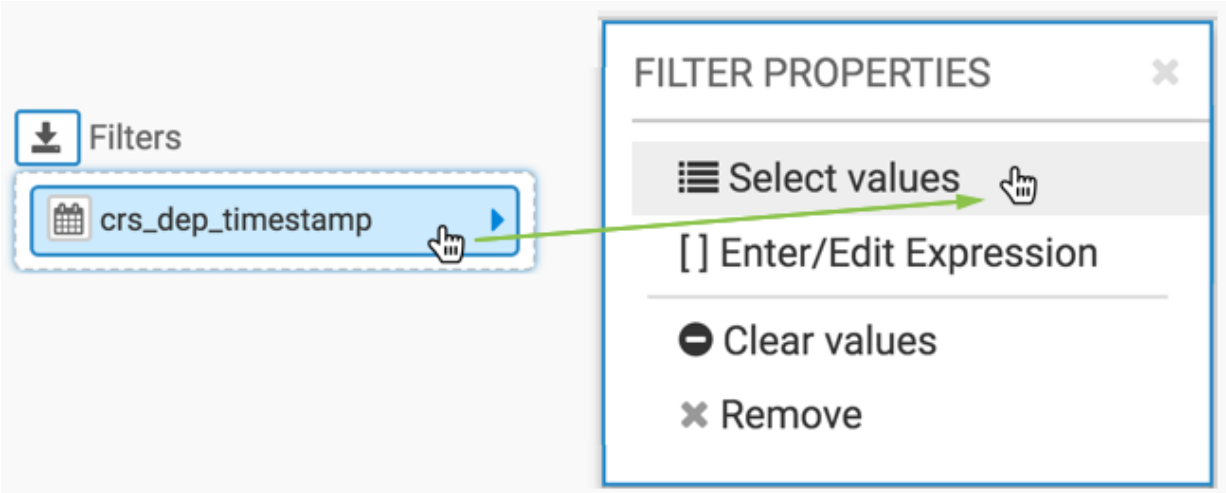

6. In the Filter for crs_dep_timestamp modal window, click the Date Range tab.
7. Click inside the textbox that contains the text.
8. Click to select range.

   Note the selection options in date range/calendar interface:

   - Pre-set ranges, such as Today, Yesterday, Last 7 Days, Last Week, Last 30 Days, This Month, and Last Month.
   - Custom Range, that can be configured either through the FROM and TO entry boxes, or by manipulating the calendar widgets
   - Time of Day control for beginning and end of the time period.

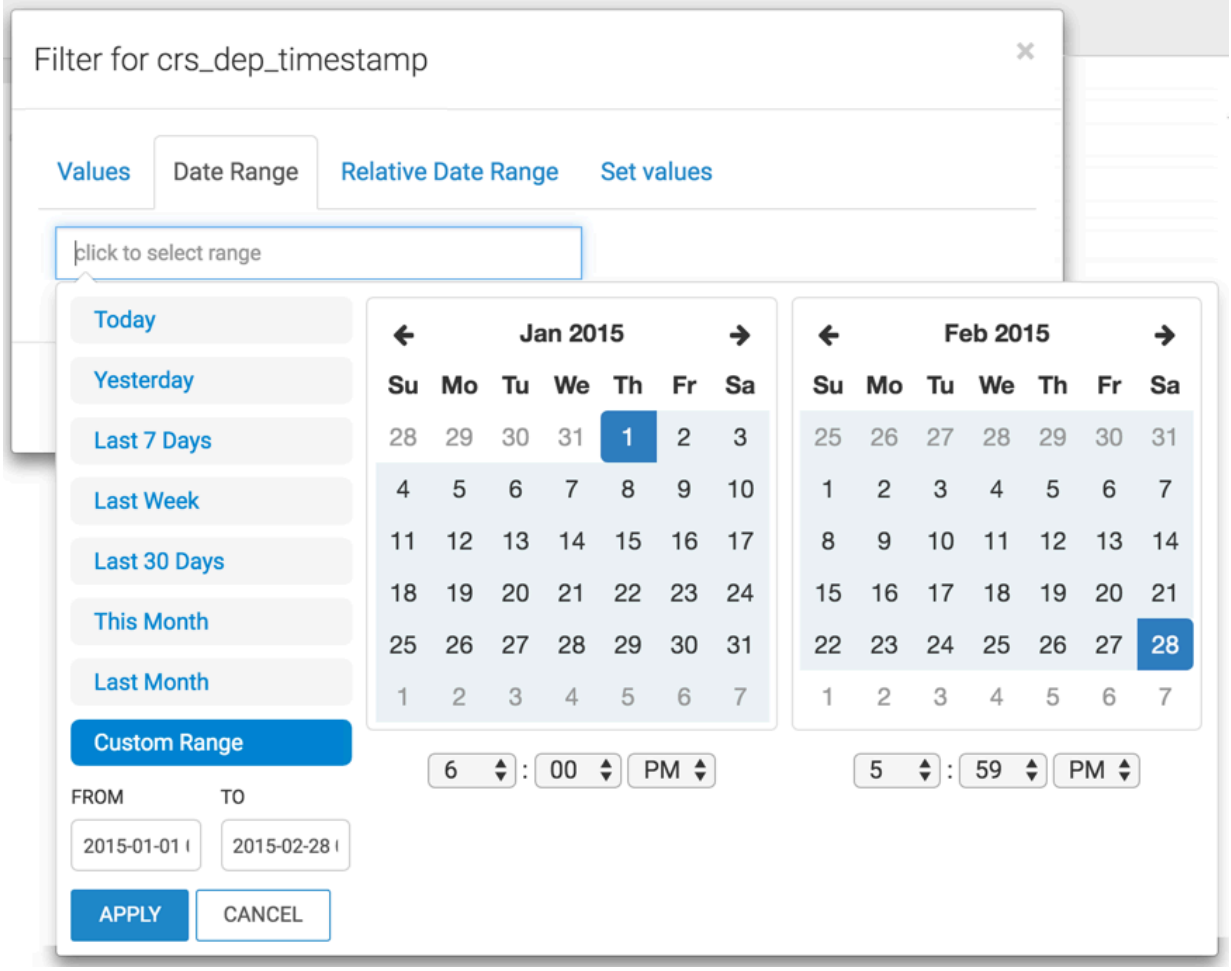

9. In the calendar widget, select the date range of 24th of January 2015 (10:00 AM) through 17th of February 2015 (9:30 PM).

10. Click Apply.

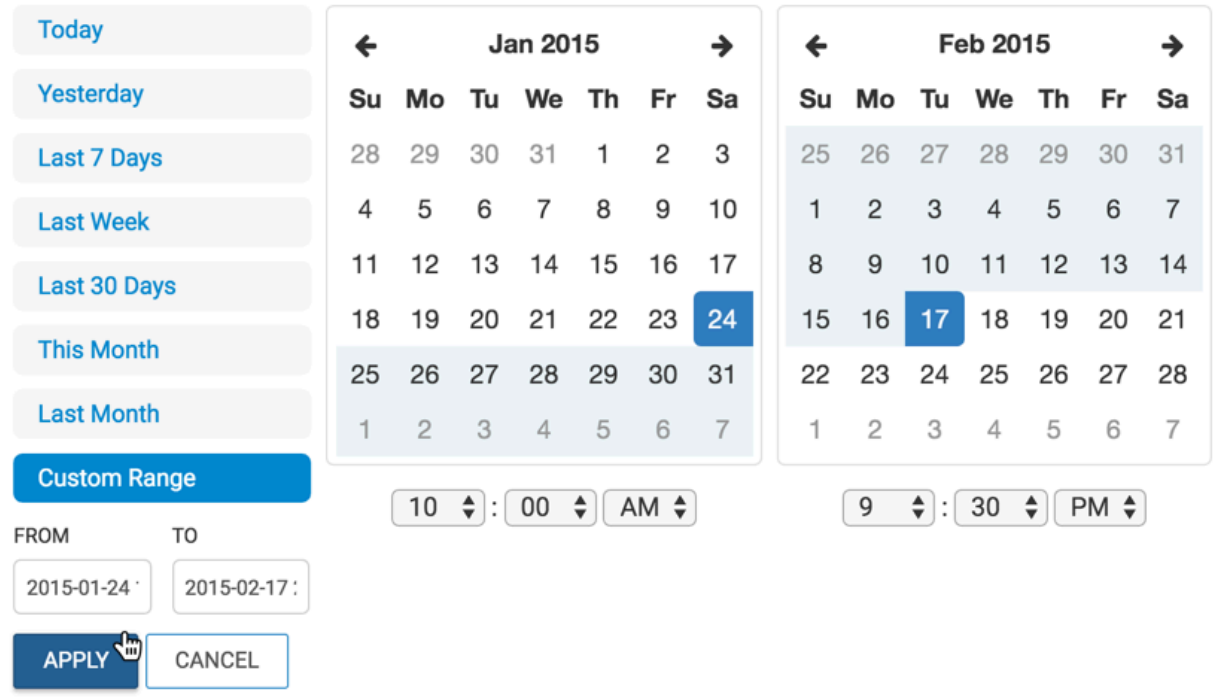

11. After clicking Refresh Visual, the updated visual appears. Note the range of dates we specified in the previous step.

	Airline													
	AA	AS	B6	DL	EV	F9	HA	MQ	NK	OO	UA	US	VX	WN
Date	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay	Delay
2015-01-25	5.20	-2.05	0.618	0.373	1.59	18.9	-3.57	16.0	11.5	8.32	7.52	1.09	7.03	3.47
2015-01-26	6.30	4.57	21.4	2.86	5.23	5.04	-5.21	12.7	5.94	4.26	7.98	3.51	4.49	4.47
2015-01-27	0.697	-2.34	4.56	-0.569	-0.959	-4.06	-5.09	5.13	-1.27	0.307	1.80	0.304	-3.47	-1.42
2015-01-28	2.38	1.09	7.77	1.23	-0.037	3.16	-1.50	2.02	0.929	0.132	2.63	2.61	5.08	1.73
2015-01-29	2.61	5.02	6.43	2.65	2.40	7.72	4.87	5.48	1.11	2.97	7.34	-0.920	-1.12	5.54
2015-01-30	4.56	15.4	11.4	7.15	5.53	14.5	-0.258	10.1	8.30	12.7	13.5	14.3	12.6	12.2
2015-01-31	4.70	10.0	5.80	1.92	-0.804	8.26	3.89	0.617	3.77	6.75	6.54	2.85	-1.11	2.38
2015-02-01	9.74	3.93	7.25	10.7	8.67	51.7	-2.32	9.66	29.2	19.3	12.7	31.9	-1.27	13.6
2015-02-02	13.7	10.2	54.0	30.2	22.7	43.1	-1.56	24.6	28.4	16.6	21.0	14.8	11.9	10.7
2015-02-03	4.83	-1.49	44.0	10.4	11.7	32.0	-3.90	21.2	16.2	12.0	15.6	7.04	8.10	6.92
2015-02-04	3.67	-1.29	9.20	5.37	13.2	47.0	-0.764	25.4	18.4	20.5	13.2	0.112	20.1	6.83
2015-02-05	13.2	7.25	32.7	8.43	9.06	15.8	0.188	33.7	22.4	23.1	15.5	5.96	24.7	12.8
2015-02-06	3.52	6.36	1.74	3.53	1.53	7.58	5.50	5.86	2.85	6.90	20.9	1.31	87.0	7.06
2015-02-07	1.41	2.62	4.77	1.83	-1.36	0.132	5.10	0.698	-1.78	4.57	6.65	-0.890	9.45	1.48
2015-02-08	8.31	6.98	23.7	5.18	3.31	4.55	4.96	9.90	4.08	11.9	12.1	2.90	57.5	8.01
2015-02-09	7.41	2.57	18.2	12.9	7.59	11.1	12.0	25.4	9.20	9.13	9.32	2.81	7.72	5.09
2015-02-10	2.58	2.60	18.1	2.62	2.51	7.06	-1.12	1.39	8.81	1.43	2.43	2.93	-1.92	3.05
2015-02-11	0.371	-1.17	7.60	2.83	0.999	8.68	4.80	3.56	-0.486	2.85	5.45	1.10	-1.62	2.19
2015-02-12	3.95	6.40	10.1	5.54	6.18	7.44	-1.40	3.82	3.95	5.04	19.7	3.59	2.67	7.57
2015-02-13	6.21	4.29	7.56	10.5	5.94	7.19	1.86	9.96	12.7	5.67	13.2	7.18	11.6	8.42
2015-02-14	5.45	3.50	16.4	9.99	4.64	41.4	33.4	10.9	21.5	3.19	13.0	3.33	6.64	10.4
2015-02-15	6.45	0.529	21.8	7.06	12.7	22.8	0.510	22.7	18.4	9.11	18.5	9.89	9.69	8.74
2015-02-16	18.3	-1.14	56.2	21.4	21.0	41.2	3.66	24.7	21.9	13.2	20.6	11.5	5.64	18.1
2015-02-17	10.8	4.02	38.6	26.1	23.4	33.4	-2.19	23.1	21.4	8.67	17.2	32.5	16.4	13.0

Related Information

Creating a join

Selecting values by using an expression on filter shelves

About this task

The Enter/Edit Expression interfaced may be used on a filter shelf to fine-tune value filtering, and to incorporate information about values from multiple rows of the dataset. To demonstrate this approach, let's use the example from building *Cross tabulation*, where we specify several discrete years of the dataset World Life Expectancy.

Procedure

1. On the Filters shelf, click Down Arrow on the year field placed there earlier.

2. Click [ ] Enter/Edit Expression

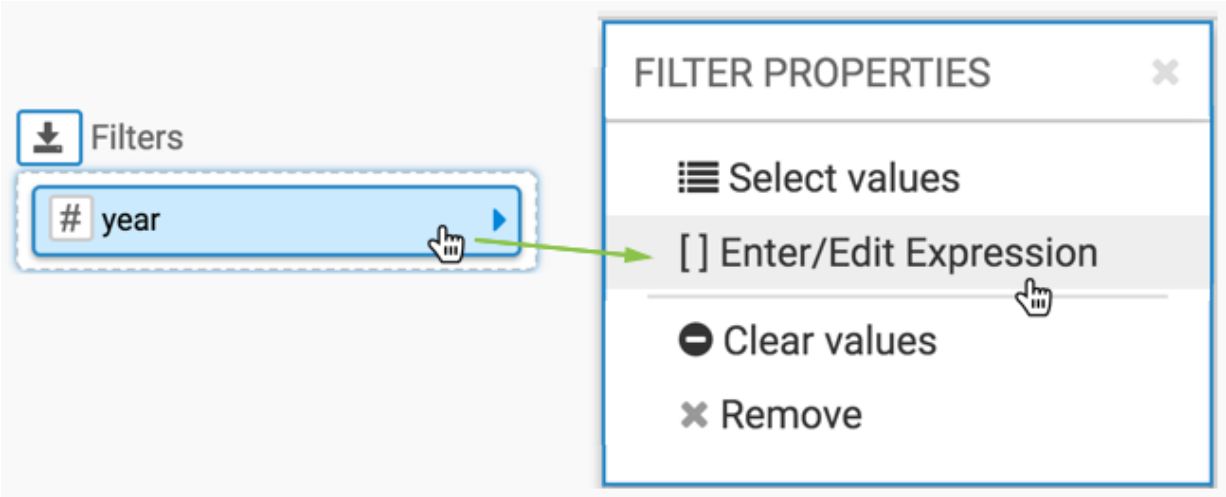

3. In the Enter/Edit Expression modal window, build an expression.

   We used the following expression to specify a range of year values:

   [year]BETWEEN 1950 AND 2010

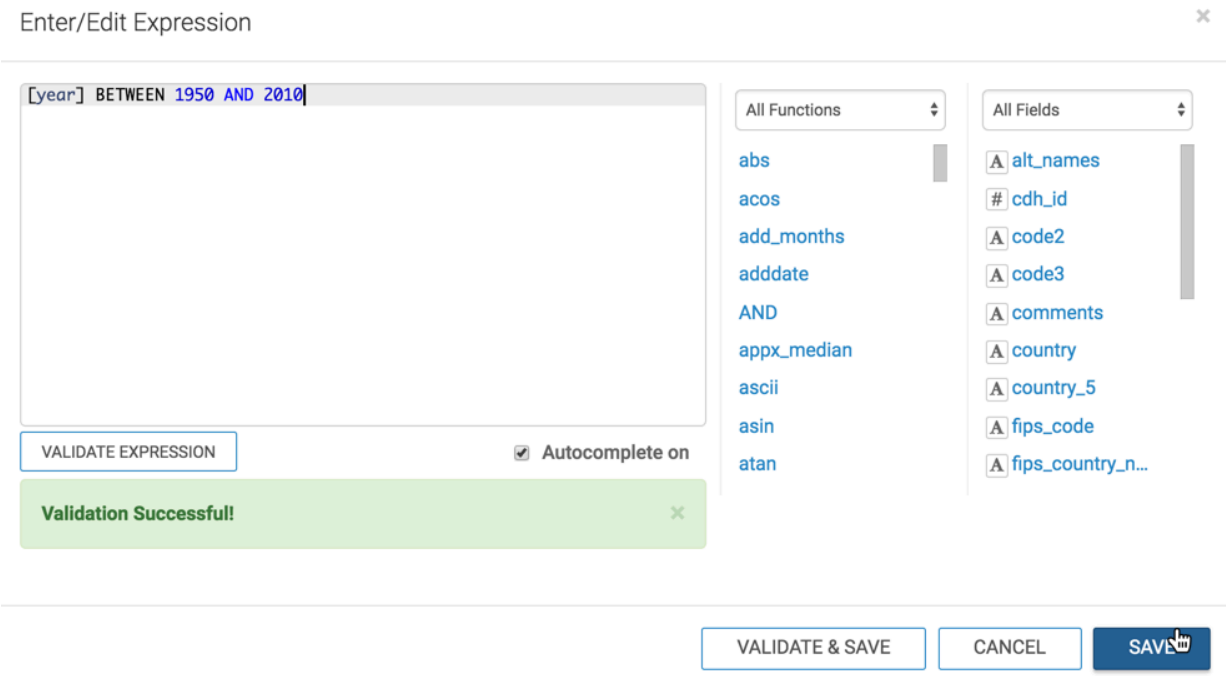

4. Click Validate Expression.
5. Click Save.

6. After clicking Refresh Visual, the cross tabulation visual appears.

   Note the range of years we specified in the previous step, and that the columns stop reporting life expectancy when it reaches the threshold of 50 years for all seven countries in Middle Africa UN sub-region.

	country						
	Angola	Cameroon	Chad	Congo	Equatorial Guinea	Gabon	Sao Tome and Principe
year	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy
1950	29.2	37.9	35.6	38.3	33.9	36.0	45.5
1951	29.4	38.0	35.7	38.5	34.0	36.2	45.7
1952	29.8	38.4	35.9	38.9	34.3	36.8	46.2
1953	30.2	38.7	36.2	39.2	34.6	37.3	46.6
1954	30.6	39.1	36.4	39.6	34.9	37.7	47.1
1955	31.0	39.4	36.7	39.9	35.2	38.1	47.6
1956	31.4	39.8	37.0	40.1	35.5	38.4	48.1
1957	31.8	40.3	37.2	40.4	35.8	38.7	48.6
1958	32.2	40.7	37.5	40.7	36.1	39.0	49.2
1959	32.6	41.1	37.8	40.9	36.4	39.3	49.8
1960	33.0	41.5	38.0	41.1	36.7	39.6	50.4
1961	33.4	42.0	38.3	41.3	37.0	39.9	50.9
1962	33.8	42.4	38.5	41.5	37.3	40.4	51.5
1963	34.2	42.9	38.8	41.8	37.6	40.9	52.1
1964	34.6	43.3	39.1	42.0	37.9	41.6	52.6
1965	35.0	43.7	39.4	42.3	38.2	42.3	53.1
1966	35.4	44.2	39.7	42.5	38.5	43.2	53.6
1967	35.8	44.6	40.0	42.9	38.8	44.1	54.2
1968	36.2	45.1	40.4	43.2	39.1	44.9	54.7
1969	36.6	45.5	40.[illegible]	43.6	39.5	45.8	55.3
1996	42.5	53.0	46.5	46.2	47.7	60.9	62.8
1997	43.1	52.8	46.5	46.1	47.7	60.7	62.9
1998	43.7	52.5	46.6	46.0	47.7	60.3	63.0
1999	44.5	52.2	46.6	46.1	47.7	60.0	63.2
2000	45.2	52.0	46.7	46.4	47.7	59.7	63.3
2001	46.0	51.8	46.8	46.7	47.8	59.6	63.5
2002	46.7	51.7	47.0	47.1	47.9	59.5	63.8
2003	47.4	51.7	47.1	47.4	48.1	59.5	64.0
2004	48.0	51.7	47.4	47.6	48.4	59.7	64.3
2005	48.6	51.9	47.7	47.9	48.8	60.1	64.6
2006	49.0	52.2	48.1	48.0	49.3	60.5	65.0
2007	49.5	52.5	48.5	48.2	49.8	60.9	65.3
2008	49.9	52.9	48.9	48.5	50.3	61.4	65.5
2009	50.3	53.3	49.3	48.7	50.9	61.9	65.7
2010	50.7	53.7	49.8	49.0	51.5	62.3	65.9

7. Let us change the expression to report only when life expectancy is at 50 years or more for the same year range.

   ([year]BETWEEN 1950 AND 2010) AND ([life_expectancy]>=50)

   Note the following results:

   - The first row is for year 1960, the first time a country in Middle Africa reported a life expectancy over 50 years.
   - Only five out of seven countries are reporting; this means that life expectancy for Chad and Congo never reaches 50 years over the period covered by our dataset, through the year 2010.

	country				
	Angola	Cameroon	Equatorial Guinea	Gabon	Sao Tome and Principe
year	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy
1960					50.4
1961					50.9
1962					51.5
1963					52.1
1964					52.6
1965					53.1
1966					53.6
1967					54.2
1968					54.7
1969					55.3
1970					55.9
1971					56.5
1972					57.1
1973					57.8
1974					58.4
1975				50.8	59.0
1976				51.6	59.5
1977				52.4	59.9
1978		50.3		53.2	60.2
1979		50.8		54.1	60.4
1980		51.2		54.9	60.6
1981		51.6		55.8	60.6
1982		52.0		56.6	60.7
1983		52.3		57.5	60.8
1984		52.6		58.3	60.8
1985		52.9		59.1	61.0
1986		53.1		59.8	61.1
1987		53.3		60.3	61.3
1988		53.4		60.8	61.5
1989		53.5		61.1	61.6

Related Information

Cross tabulation

Cloudera Data Visualization 7.2.9

Filters and Parameters

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Filter shelves **4**

Filter scope **4**

Filter widgets on dashboards **5**

Syntax for using parameters **5**

Filter shelves

Cloudera Data Visualization uses filters and parameters to change the analysis criteria at run-time by specifying the data selected for the visuals. Filters define the subsets of data that is examined in a visual. By placing either measures or dimensions on the filter shelf of a visual, you can exclude or include specific values or a range of values from the data used.

You can specify filters by using one of the following methods:

- Select a value or values from a list, or through an interface specific to the data type
- Specify an expression that represents the filtering function

For more information on filter expressions, see *Syntax for using parameters*.

Related Information

Syntax for using parameters

Filter scope

Cloudera Data Visualization uses filters and parameters to change the analysis criteria at run-time by specifying the data selected for the visuals. Filter scope determines the level at which filters are active.

Filter scope has the following three options:

Explicit scope

Explicit scope is defined when a dashboard or visual sends out parameters to other visuals, and the visuals accepts these parameters in their Filters shelf. This can be enabled at the level of each visual.

Explicit scope is expected to be used sparingly, for a small number of visuals in each dashboard that represent a high-level look at a metric and that should not be filtered by visible filters.

Note: Explicit scope overrides dataset scope.

The scope of click behavior is explicit.

Tip: If you want visuals to be updated based on selected elements, make sure to explicitly place these columns in the filter shelf of the target visuals.

For more information, see:

- *Setting explicit scope for filters*
- *Using parameters with explicit scope*

Dataset scope

Dataset scope filter applies to all visuals in a dashboard that do not have explicit scope turned on. Filters with dataset scope apply to all visuals in the dashboard that are using the same dataset. This is the default for new filters.

Tip: Dataset scope filters continue to use the parameter passing mechanism, and these parameters may be used within visuals in many ways, for example in the title, within expressions.

For more information, see:

- *Setting dataset scope for filters*
- *Sending parameters with dataset scope*

App scope

App scope automatically filters all dashboards in the app. For more information, see *Setting app scope for filters*.

Related Information

Setting explicit scope for filters
Using parameters with explicit scope
Setting dataset scope for filters
Sending parameters with dataset scope
Setting app scope for filters

Filter widgets on dashboards

In Cloudera Data Visualization, filters and parameters allow you to adjust analysis criteria in real-time by specifying the data displayed in visuals. A filter is a control element linked to the values of a specific table column.

When building a dashboard, you can add filter widgets to the Filters area. You also have the option to create and add custom filters. After you define a custom filter, you can apply it to visual shelves to dynamically update the displayed data.

For detailed instructions on creating, configuring and using filter widgets, see *Filter widgets on dashboards*.

Related Information

Filter widgets on dashboards

Syntax for using parameters

You can define a filter by specifying an expression that represents the filtering function.

You have the following syntax format options:

- Adding the value of the output parameter variable to the visual:

  ```
  <<variable:default-value>>
  ```

  You can define it by URL parameters, clicked values within dashboards, or single value/textbox pick lists.

  Some examples:

  - By fields: (<<field-name:dimension-column>>)
  - By discrete single values: (<<state:'AL'>>)
  - By discrete multiple values: (<<state.data:‘NC’, ‘KY’>>)
- User-defined pick lists also emit an alias parameter:

  ```
  <<variable.alias:default-value>>
  ```
- When dashboard filters function in multi-select mode, you must also specify the include-exclude operators:

  ```
  <<variable.data:'defalut-value'>>
  <<variable.exclude:default-value>>
  ```
- For date-range functions, the filter expression that uses the timestamp variable ts has the following pattern:

  ```
  [ts] between <<ts.start:"2013-08-03">> and <<ts.end:"2014-04-28">>
  ```

- When the dashboard filter is multi-select, you must use different parameters:
  - The selected values are available through the <<variable.data:default-value>> expression.
  - When implementing the include or exclude operator in a multi-select filter, use the <<variable.exclude:*DEFAULT-VALUE*>> expression.
- To customize the title of any visual, you can change its name to include a parameter. For a practical example, see *Creating visuals with variable dimensions*.

Related Information

Creating visuals with variable dimensions

Cloudera Data Visualization 7.2.9

Filter Widgets on Dashboards

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating dataset filters on a dashboard 5

Creating custom filters on a dashboard 10

Creating date/time filters on a dashboard 15
- Using fixed dates (date range picker) mode 18
- Using relative dates mode 24
- Using quick filters 28
- Including null values in date filters 31
  - Creating a dashboard with date filters 31
  - Including null values in a visual 32

Setting parameters through URL 37

Customizing output parameters 40

Configuring ranges on a date/time filter 44

Configuring data options for a dashboard filter 46
- Changing number of initial options in dashboard filters 47
- Changing number of options in search output of dashboard filters 48
- Disabling incremental queries in filters 48
- Configuring sort order in filter values 48

Configuring display settings for a dashboard filter 49
- Displaying a textbox parameter 51
- Changing the width of the filter text box 53
- Entering values manually 54
- Using single value mode with radio buttons 58
- Selecting all values 59
- Using single value with dropdown menu 61
- Displaying distinct parameters 62
- Hiding filters 64
- Remembering previous selections on dashboard filters when searching 65
- Using multi-select mode 67

Configuring dashboard filter scope 69

Configuring custom style for a dashboard filter 72
- Configuring custom style using CSS code 74

Moving filter widgets on a dashboard.....76

Creating dataset filters on a dashboard

Cloudera Data Visualization enables you to add a filter widget to the dashboard based on any field in the dataset.

Procedure

1. Open an existing dashboard in Edit mode.
2. Go to the Filters tab on the Dashboard Designer side menu bar.

3. Click any field of the dataset from Dimensions or Measures to add a widget to the dashboard that filters on that field.

Alternatively, you can also drag the field that you want to use as a filter and drop it into the filter bar on the dashboard.

A new filter widget is added in the filter area of the dashboard, named after the field you have used. You can hover over the widget to access filter actions in the top-right corner.

- Clicking the icon moves the filter to the grid area.
- Clicking

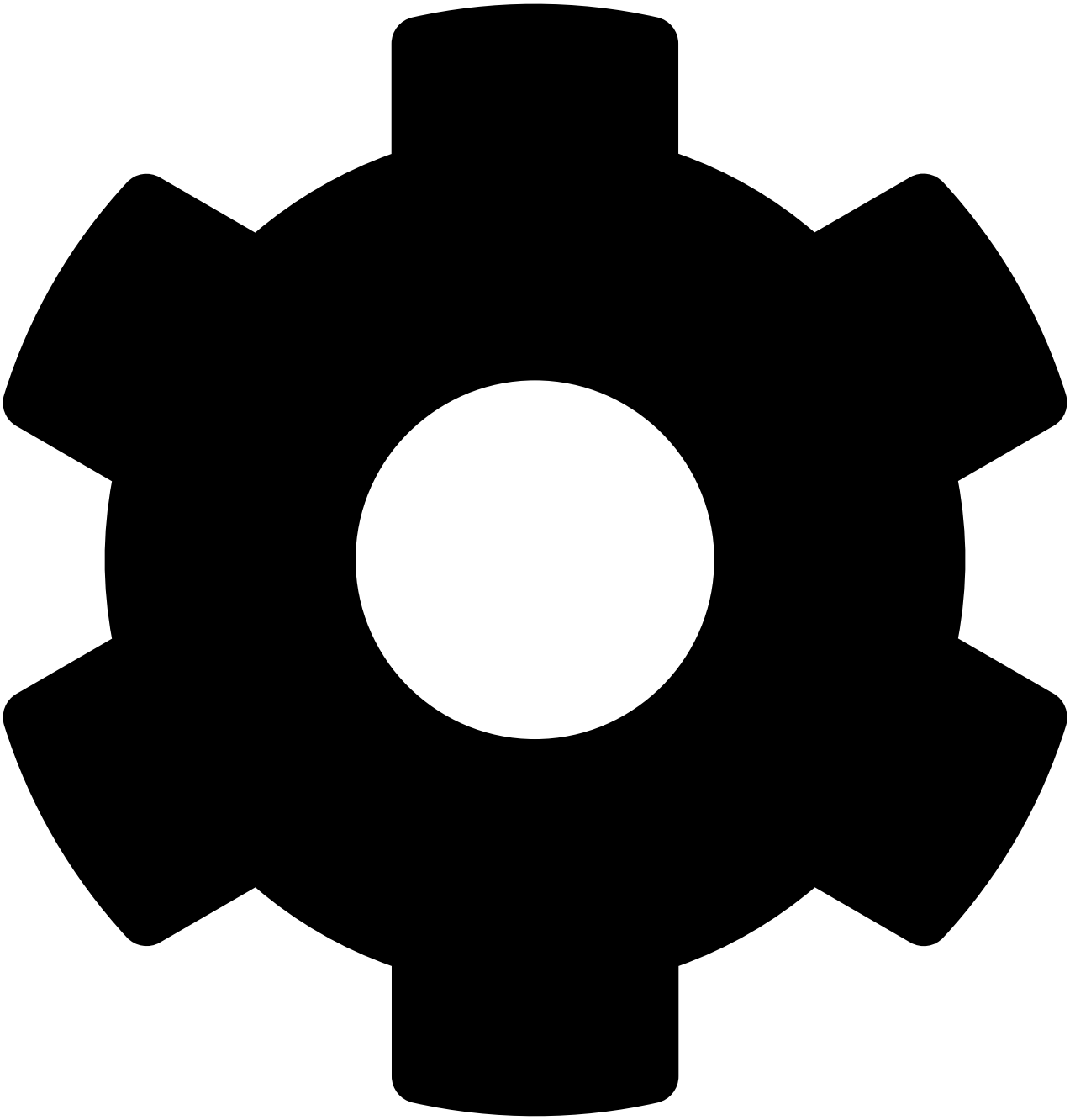

opens the Settings modal window that allows you to configure the filter.
- Hovering over the Info icon displays the filter's information modal, providing information about the visual and the dataset.

- Clicking the ✖ icon removes the filter from the dashboard.

4. To set the basic configuration for the new dashboard filter, click

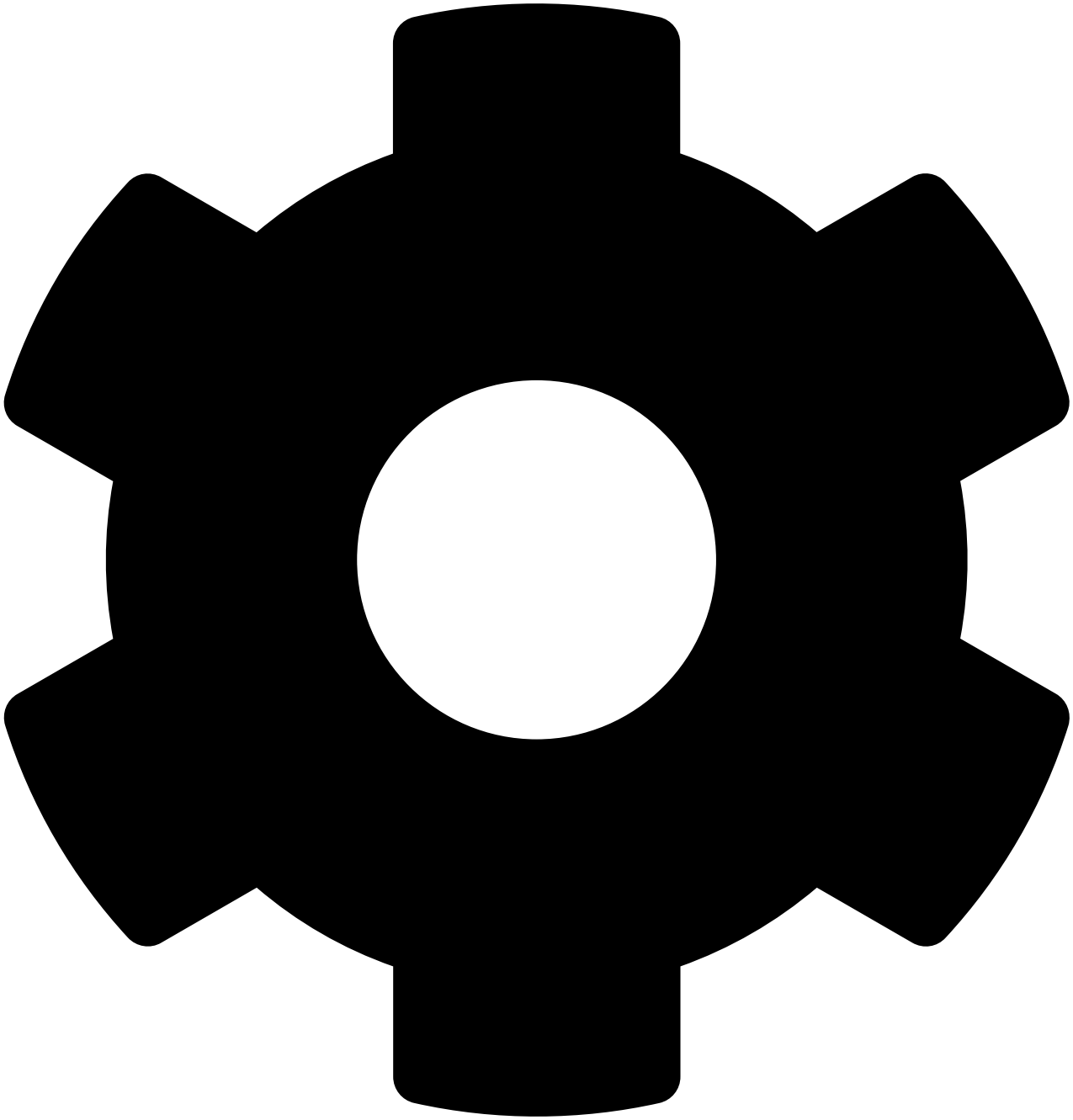

in the top right corner.

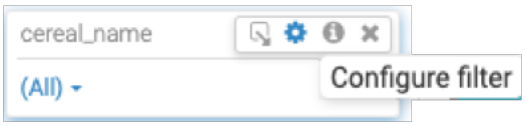

The Settings modal window opens on the Values tab, showing the dataset of the filter and several configuration options.

a) To switch the filter to another dataset and connection, click .
b) To preserve the adjusted filter settings for View mode, select the Save selections made in edit mode option. You can use it to save all changes made in Edit mode.
c) To exclude null values for all filter types, use the Permit selection of NULL values setting.

d) From Base field specifies the filter-controlling field. You can change the base field to another field of the dataset. This option is available only in dataset-based filters.

e) Title shows the name of the filter as it appears in the application at run-time. Keep it short and descriptive providing information on the filter's function.

f) Output Parameter is the parameter sent to the target visual to determine which field to display. For optional or variable parameters, output parameter is used on the shelves of the target visual, such as <<[dim], or <<[agg], or in an expression that contains the variable.

   When used in filters that are based on existing fields, the output parameter ensures field name match across datasets. For example, a filter that is based on the state field in the US State Populations Over Time dataset must have the output parameter stname to control visuals in the US County Population dataset.

g) Filter Fields specifies the base fields of another field, based on which options of the current filter can be filtered. You can enter multiple base fields separated by a comma. This option is available only in dataset-based filters.

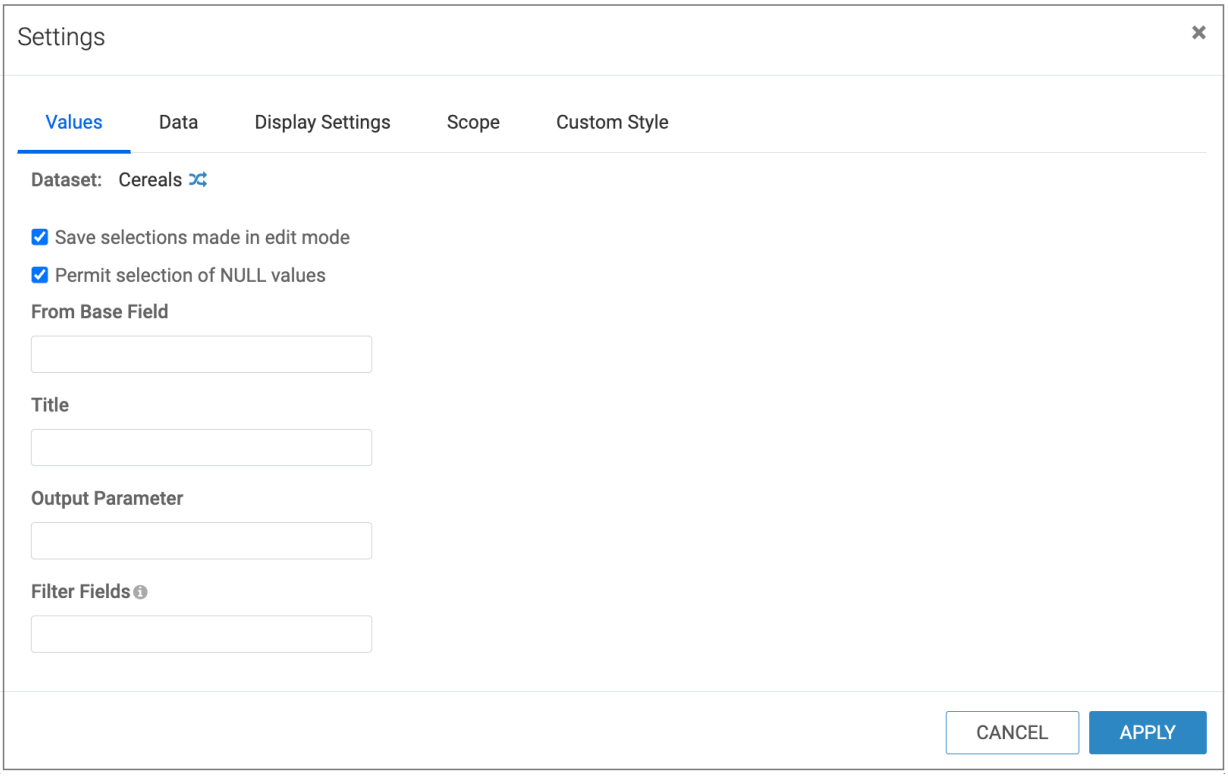

5. Click APPLY to save the configuration settings.

What to do next

For more information on further configuration options, see the following documentation.

Related Information

Setting parameters through URL
Customizing output parameters
Configuring data options for a dashboard filter
Configuring display settings for a dashboard filter
Configuring dashboard filter scope
Configuring custom style for a dashboard filter

Creating custom filters on a dashboard

Cloudera Data Visualization enables you to create a custom filter based on any parameter.

Procedure

1. Open an existing dashboard in Edit mode.
2. Go to the Filters tab on the Dashboard Designer side menu bar.

3. Click CREATE CUSTOM FILTER.

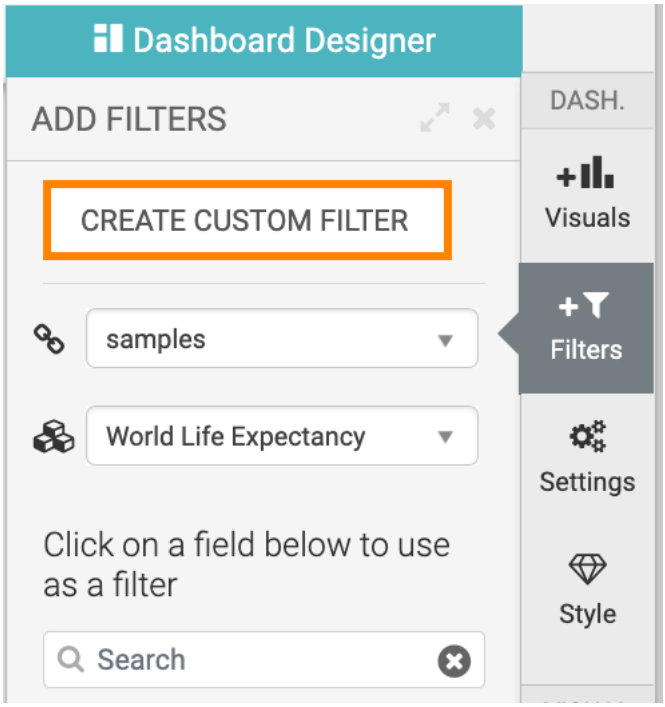

A new filter widget named New Filter is added to the dashboard. You can hover over the widget to access filter actions in the top-right corner.

- Clicking the Move icon moves the filter to the grid area.

- Clicking

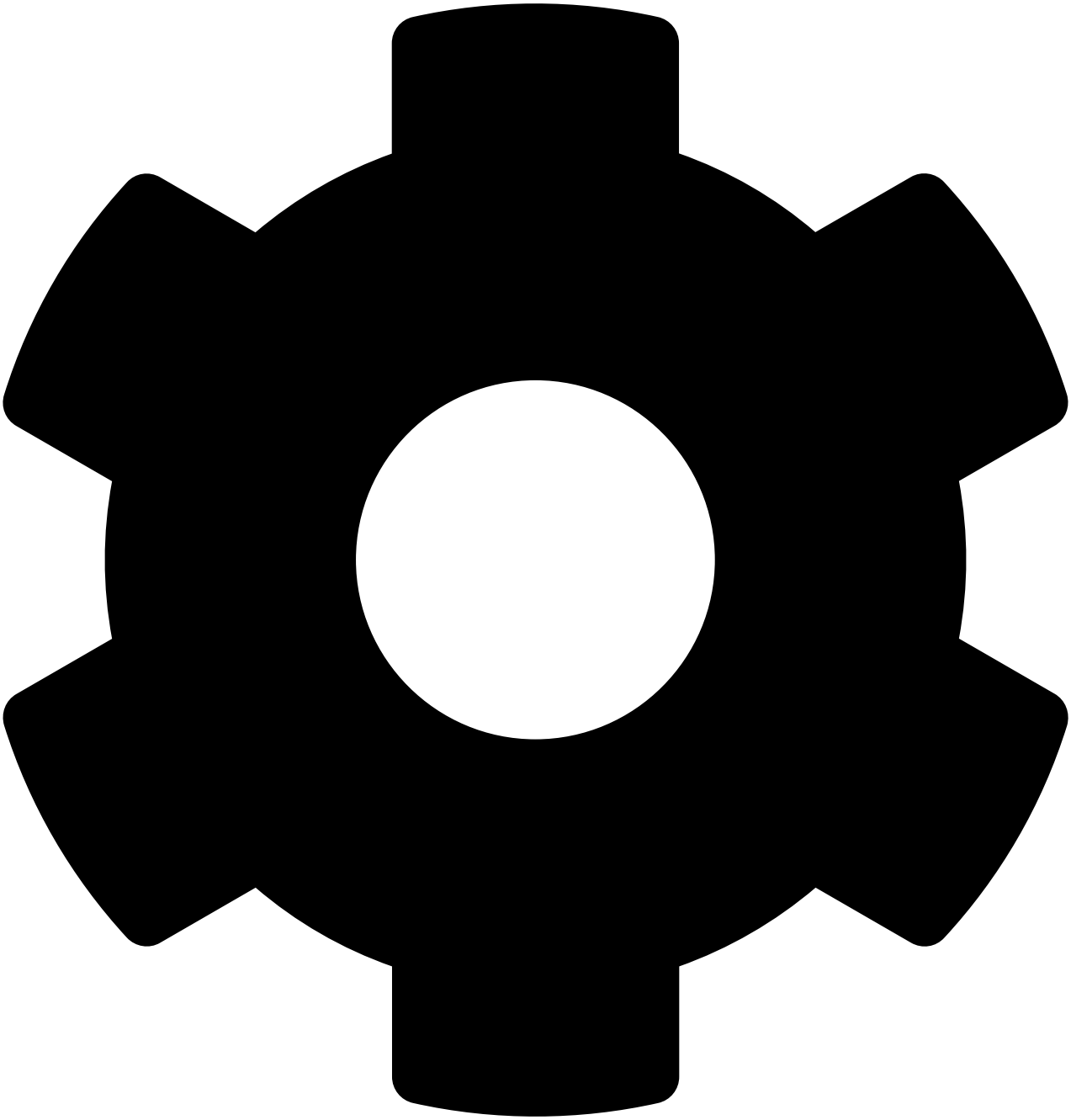

  opens the Settings modal window that allows you to configure the filter.
- Hovering over the Info icon displays the filter's information modal, providing information about the visual and the dataset.
- Clicking the Cross icon removes the filter from the dashboard.

4. To set the basic configuration for the new dashboard filter, hover over the filter widget and click

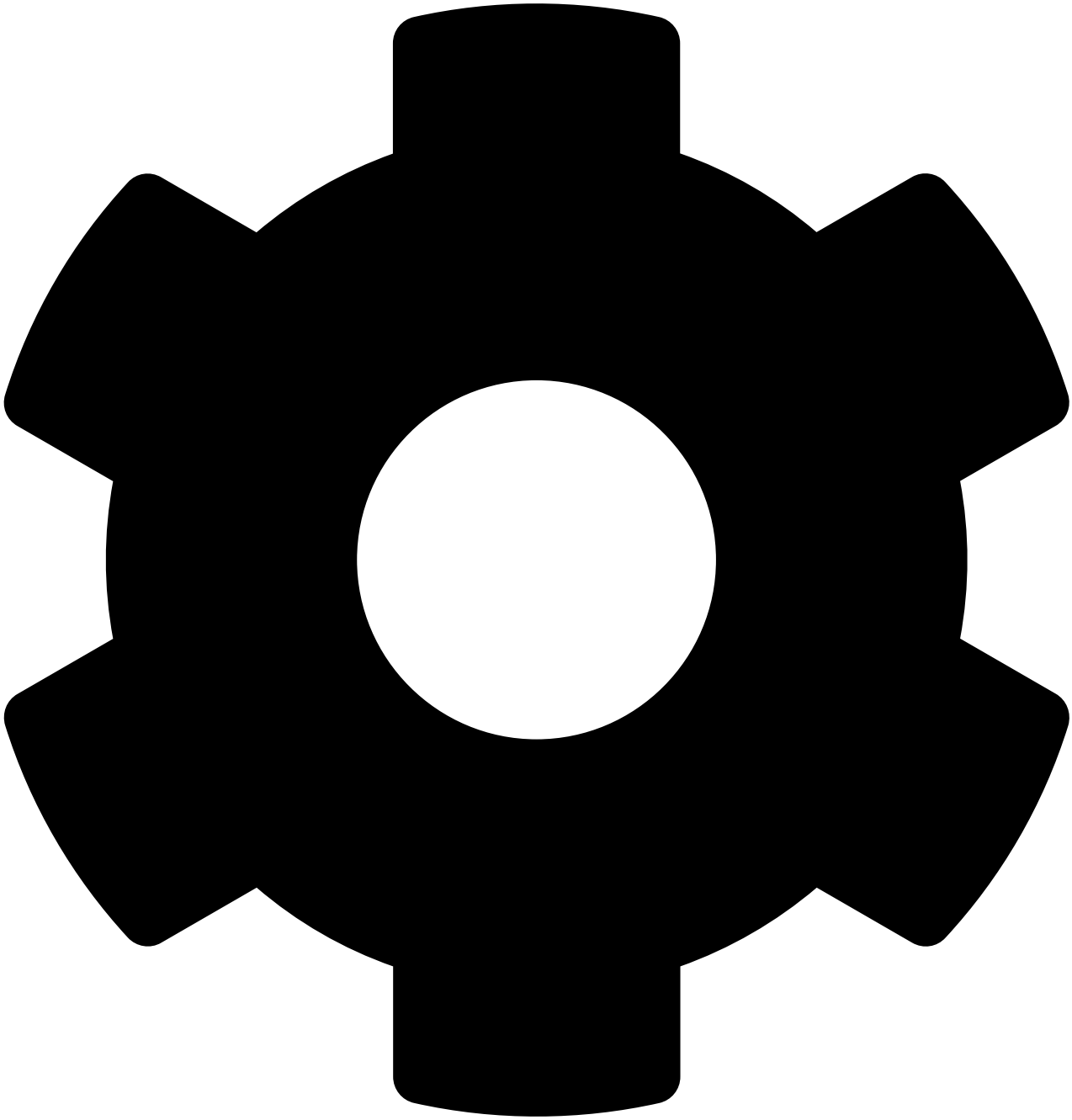

in the top right corner.

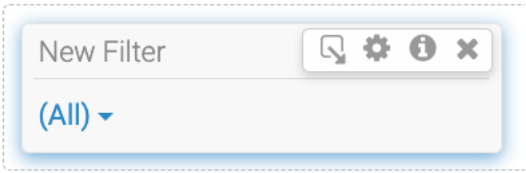

The Settings modal window opens on the Values tab, showing several configuration options.

a) To preserve the adjusted filter settings for View mode, select the Save selections made in edit mode option. You can use it to save all changes made in Edit mode.
b) Title shows the name of the filter as it appears in the application at run-time. Change it from New Filter to something that provides information on the filter's function. Keep the name short and descriptive.

c) Output Parameter is the parameter sent to the target visual to determine which field to display. In cases of optional or variable parameters, output parameter is used on the shelves of the target visual, such as <<[dim], or <<[agg], or in an expression that contains the variable.

When used in filters that are based on existing fields, the output parameter ensures field name match across datasets. For example, a filter that is based on the state field in the US State Populations Over Time dataset must have the output parameter stname to control visuals in the US County Population dataset.

d) Specified values specifies the distinct values of the output parameter on which the filter acts, and labels them.

For example, a <<mes>> output parameter can have the following specified values:

- Value: gdp_per_capita, Label: GDP per Capita
- Value: life_expectancy, Label: Life Expectancy

This option is available only in custom filter widgets. For more information see *Setting parameters through URL*

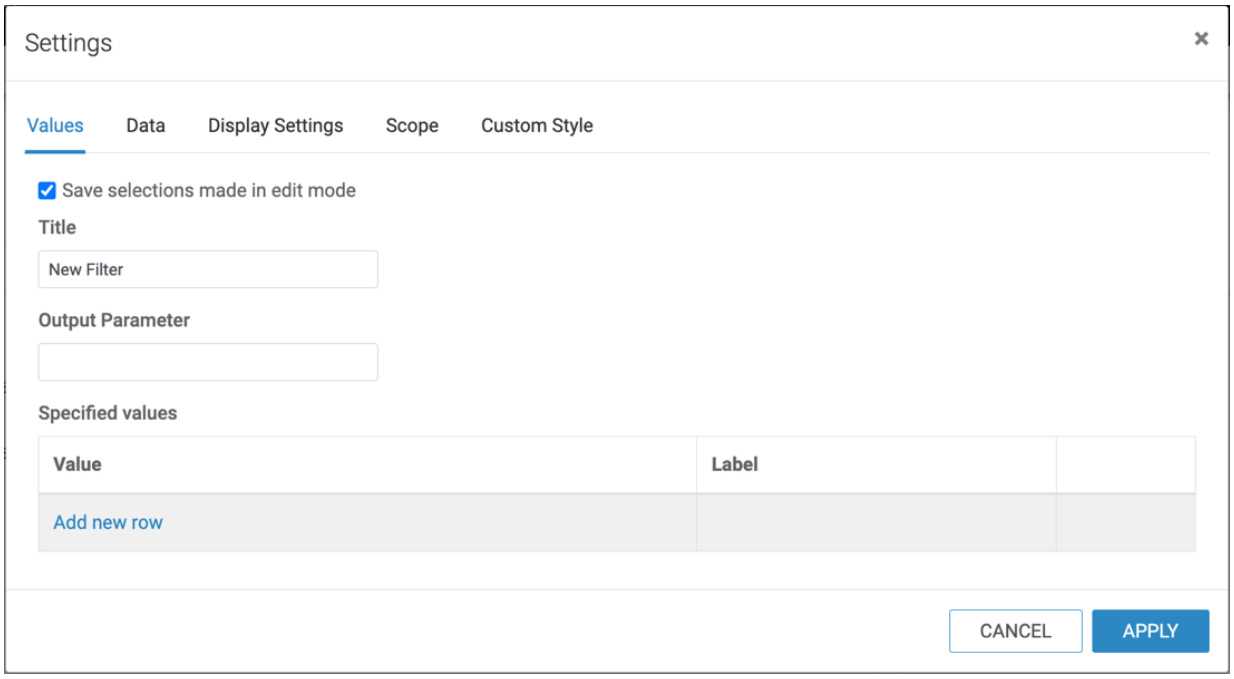

5. Click APPLY to save the configuration settings.

What to do next

For more information on further configuration options, see the following documentation.

Related Information

Setting parameters through URL
Customizing output parameters
Configuring data options for a dashboard filter
Configuring display settings for a dashboard filter
Configuring dashboard filter scope
Configuring custom style for a dashboard filter

Creating date/time filters on a dashboard

In Cloudera Data Visualization, date and timestamp filters are essential for dynamically adjusting your dashboard visuals based on time-related data. By using date/time filters, you can allow users to easily explore and analyze records filtered by specific dates or time ranges. These instructions walk you through the process of creating and configuring date/time filters on a dashboard, including basic settings, time range selection, and customization options. You will learn how to add filters, adjust their configurations, and optimize them for a more tailored user experience.

Procedure

1. Open an existing dashboard in Edit mode.
2. Go to the Filters tab on the Dashboard Designer side menu bar.

3. Click any date type field from the dataset under Dimensions or Measures.

   You can see a new date/time filter widget added to the dashboard named after the selected field. You can hover over the widget to access filter actions in the top-right corner.

   - Clicking the Move icon moves the filter to the grid area.
   - Clicking

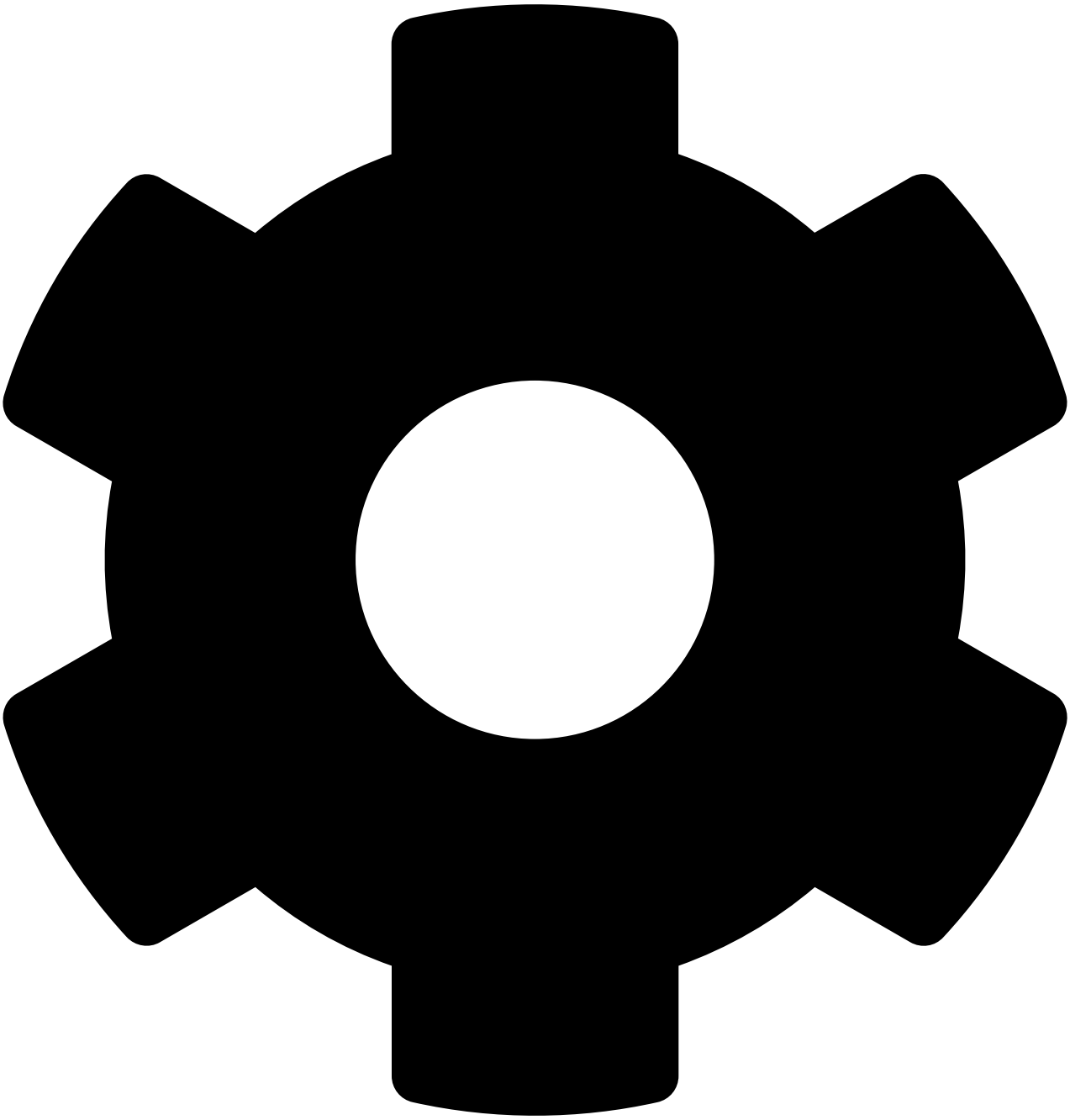

     opens the Settings modal window that allows you to configure the filter.
   - Hovering over the Info icon displays the filter's information modal, providing information about the visual and the dataset.
   - Clicking the Cross icon removes the filter from the dashboard.

   For instructions on how to configure date ranges, see *Configuring ranges on dashboard filters*.

4. To set basic configuration for the new dashboard filter, hover over the filter widget and click

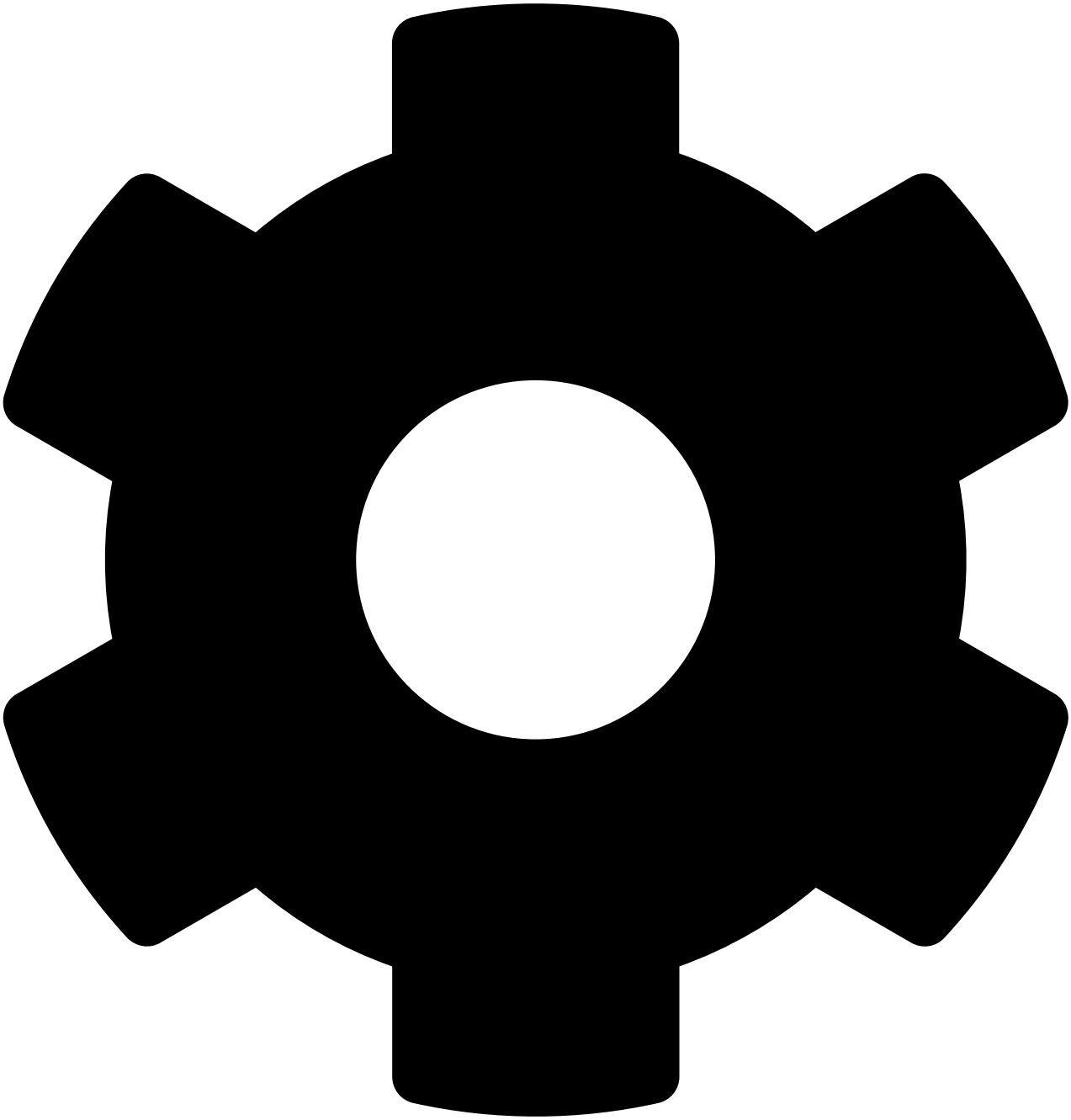

in the top right corner.

The Settings modal window opens on the Basic tab, showing the dataset of the filter and several configuration options.

a) You can switch the filter to another dataset and connection by clicking .

b) You can filter data by specifying a start and end date/time range on the dashboard using Permit selection of time range.

   Time Zone enables you to adjust the date/time filter to a specific time zone. To activate this, first check Permit selection of time range then choose the time zone from the drop-down. The default is UTC.

c) You can use the Permit selection of NULL values setting to exclude null values for all filter types.

   For more information, see *Including null values in date filters* .

d) To show only the latest timestamp, select the Default to the latest available timestamp option.

Note: If you select this option, the Save date range selections made in edit mode becomes unavailable.

e) To preserve the adjusted filter settings for View mode, select the Save date range selections made in edit mode option. You can use it to save the changes made in Edit mode.
f) Base field specifies the filter-controlling field. You can change the base field to another field of the dataset. This option is available only in dataset-based filters.
g) Title shows the name of the filter as it appears in the application at run-time. Keep it short and descriptive providing information on the filter's function.

Important:

Avoid adding the same field to the dashboard filter if it is used as both a date range picklist AND a time control on the dashboard.

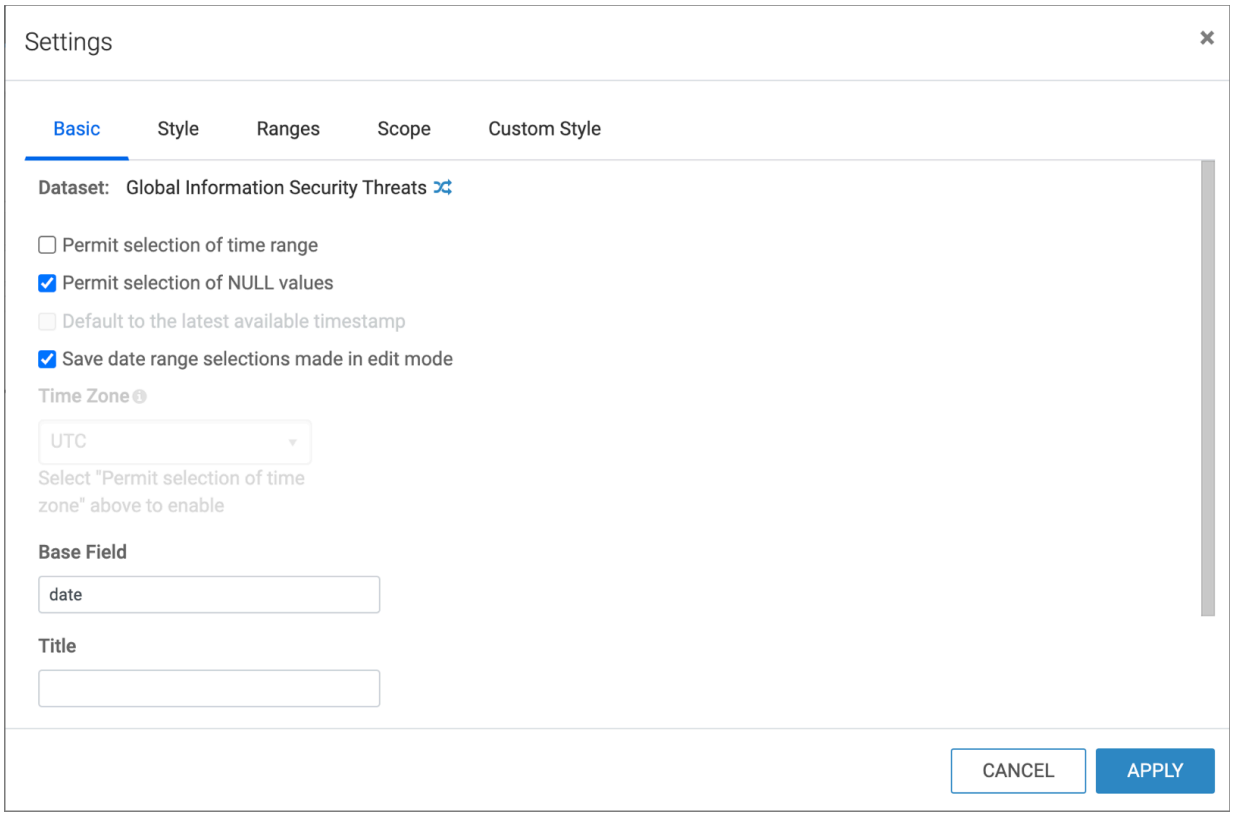

5. Click APPLY to save the configuration settings.

What to do next

For more information on further configuration options, see the following documentation.

Related Information

Configuring ranges on a date/time filter
Configuring dashboard filter scope
Configuring custom style for a dashboard filter

Using fixed dates (date range picker) mode

In Cloudera Data Visualization, using fixed dates in a date filter enables you to select a subset range of dates for visualizing your data.

About this task

The following steps show you how to use the date range picker option of a date and time filter. The instructions use a calendar heatmap visual as an example.

Before you begin

- You have created a calendar heat map visual using the SFPD Incidents dataset. For more information on this visual type, see Calendar heatmaps.
- You have added this visual to a dashboard.

Procedure

1. Open your dashboard that contains the heatmap visual in Edit mode.
2. Switch to the Filters tab on the Dashboard Designer side menu bar.

3. Under Dimensions, click the date field.

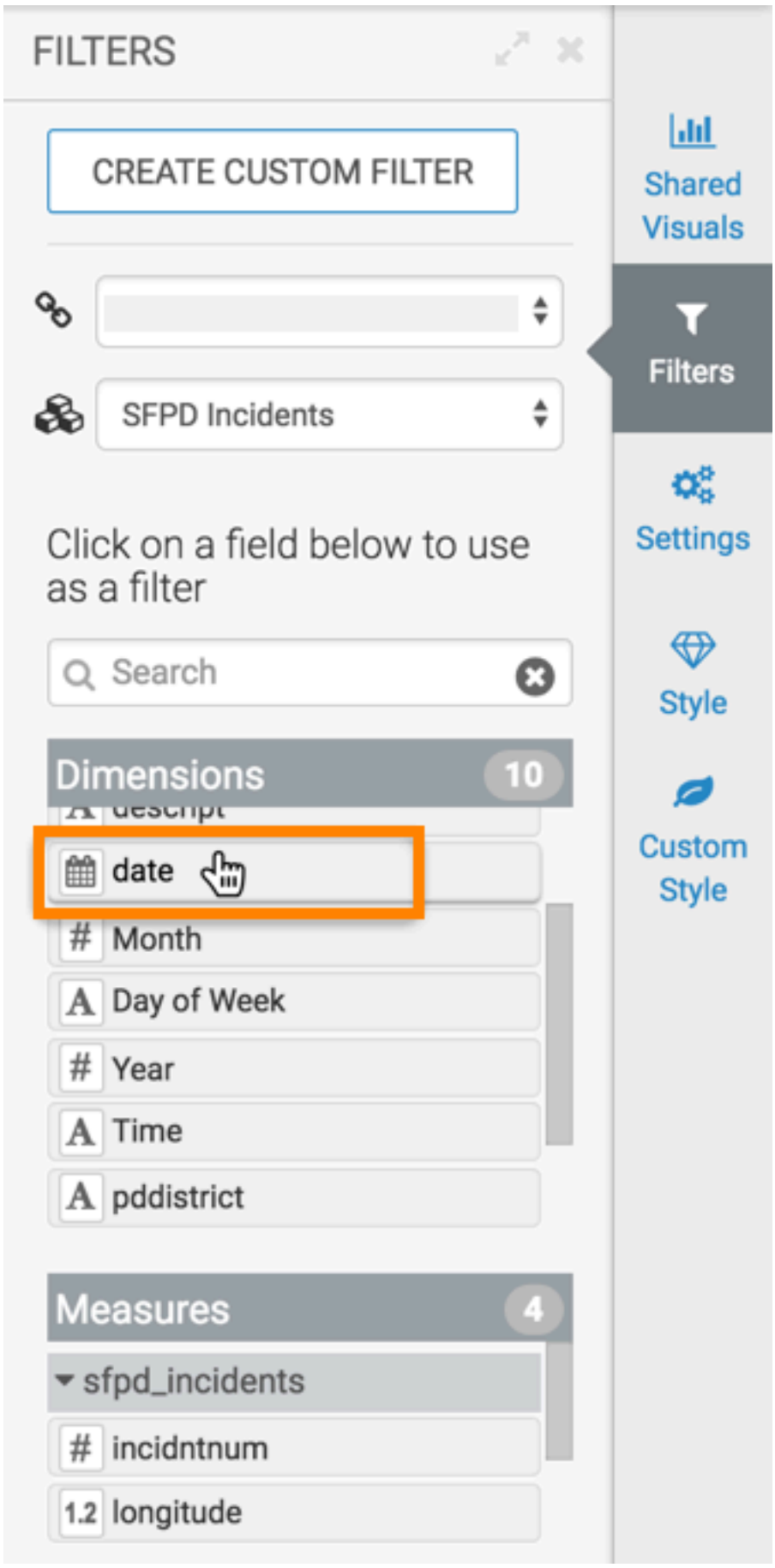
FILTERS
CREATE CUSTOM FILTER
SFPD Incidents
Click on a field below to use as a filter
Search
Dimensions 10
date
Month
Day of Week
Year
Time
pddistrict
Measures 4
sfpd_incidents
incidntnum
longitude
Shared Visuals
Filters
Settings
Style
Custom Style

This adds a filter to the dashboard. This filter is based directly on the date field and has a timestamp type.

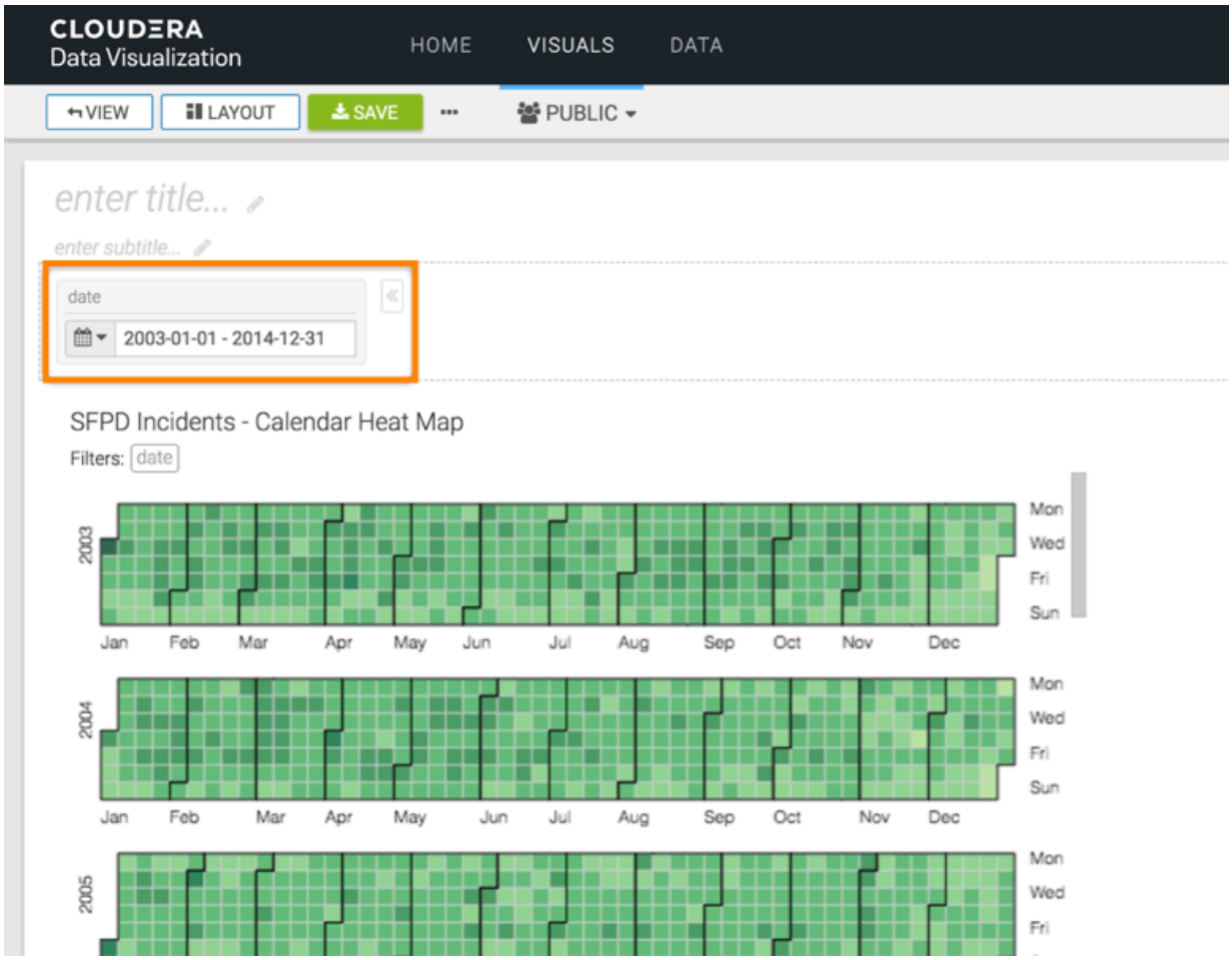

The dashboard filter contains the date range for the entire dataset; in this case, it is from 2003-01-01 through 2014-12-31.

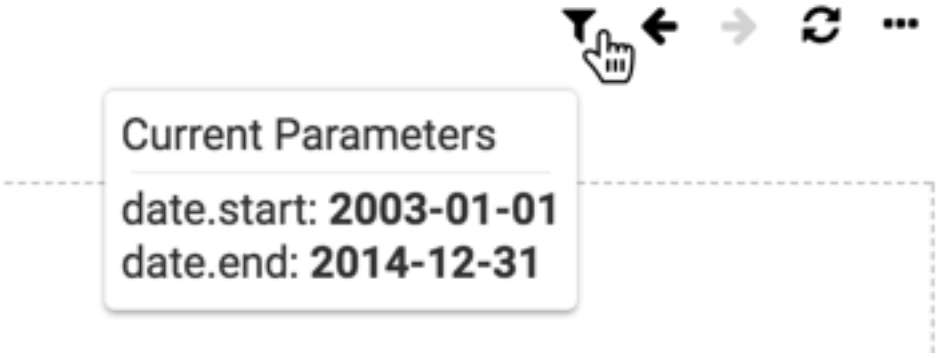

4. To use the date range picker interface, click the Calendar icon on the filter, and select Fixed Dates from the menu.

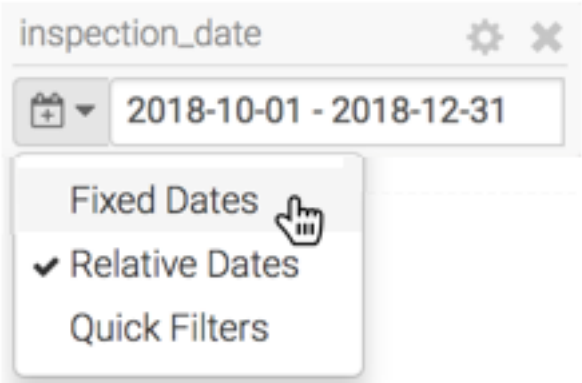

You can also click the date field of the dashboard filter to see the date range picker interface.

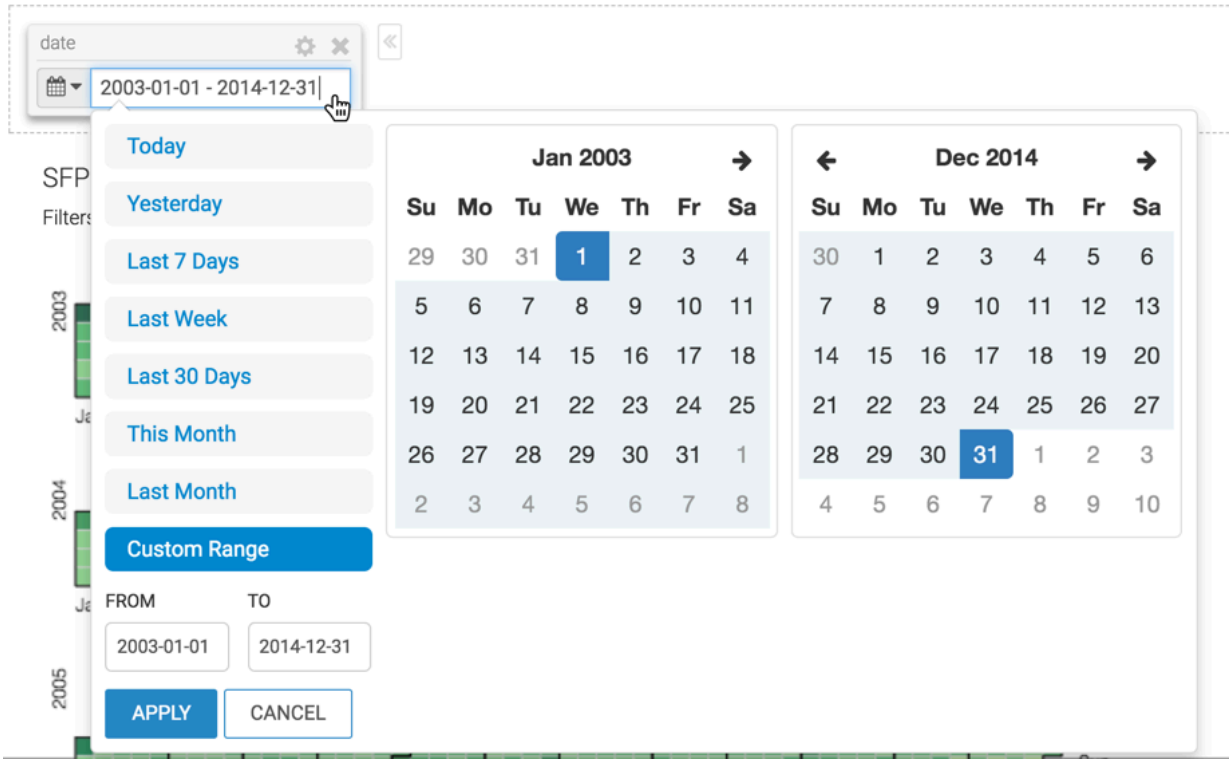

You can change the date range in various ways:

- By selecting the default options:
  - Today
  - Yesterday
  - Last 7 Days
  - Last Week
  - Last 30 Days
  - This Month
  - Last Month
- By changing the date entries for FROM and TO fields
- By picking the start and end dates directly on the calendar interface

5. Change the date range to 2010-01-01 through 2010-12-31, and click APPLY.

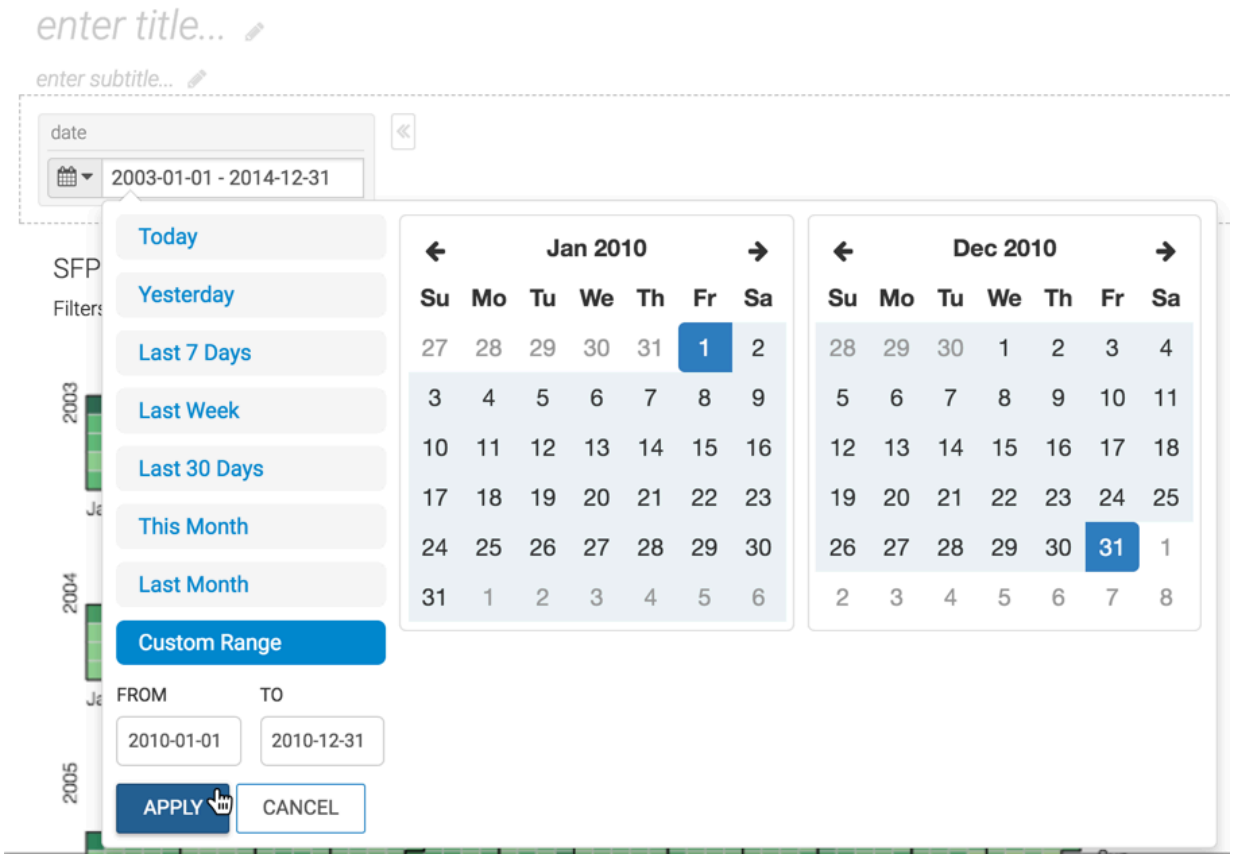

Results

The visual output is now limited to the 2010 calendar year and the application filter shows this new date range.

Related Information

Calendar heat maps

Using relative dates mode

In Cloudera Data Visualization, the relative date mode in a date filter enables you to select a subset range of dates anchored to a particular date.

About this task

The following steps show you how to use relative date mode of a date and time filter. The instrucions use a calendar heat map visual as an example. You can learn how to anchor a date to 2015-01-01 and limit the visual output to the next two years from this date.

Before you begin

- You have created a a calendar heat map visual using the Food Stores Inspection in NYC dataset with the inspecti on_date dimension on the Date shelf and the Record Count measure on the Measures shelf.

  For more information on this visual type, see Calendar heatmaps.
- You have added this visual to a dashboard.

Procedure

1. Open your dashboard in Edit mode.
2. Switch to the Filters tab on the Dashboard Designer side menu bar.

3. Under Dimensions, click the inspection_date field.

   This adds a filter to the dashboard. This filter is based directly on the inspection_date field, and has a timestamp type.

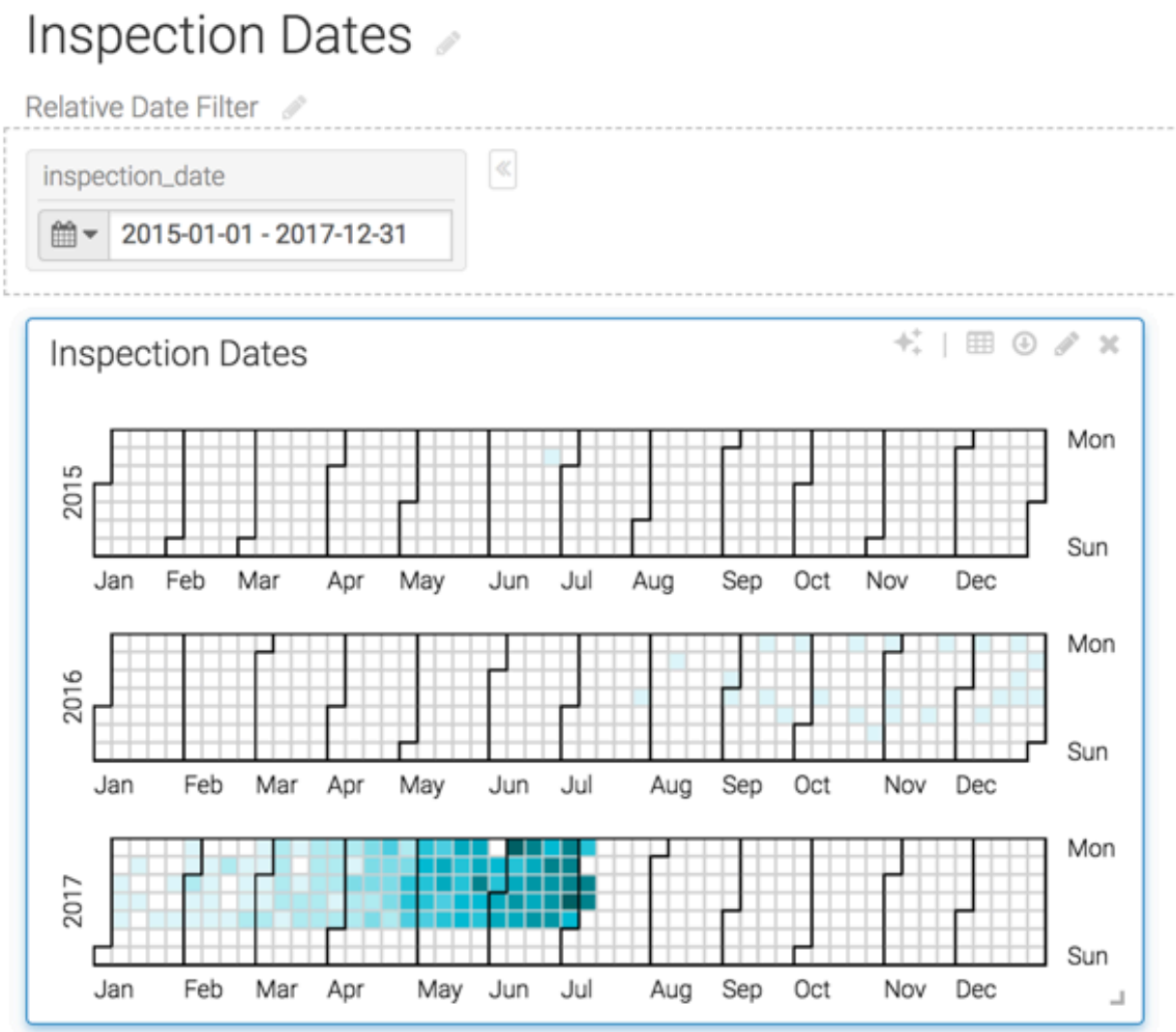

The dashboard filter contains the date range for the entire dataset. In this case, it is for three years from 20015-01-01 through 2017-12-31.

4. Cick the Calendar icon on the filter and select Relative Dates from the menu.

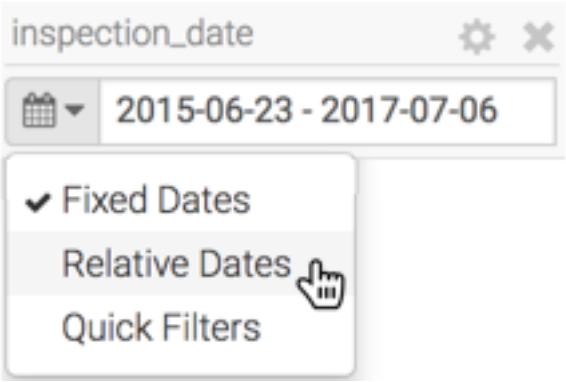

The Relative Date modal window appears.

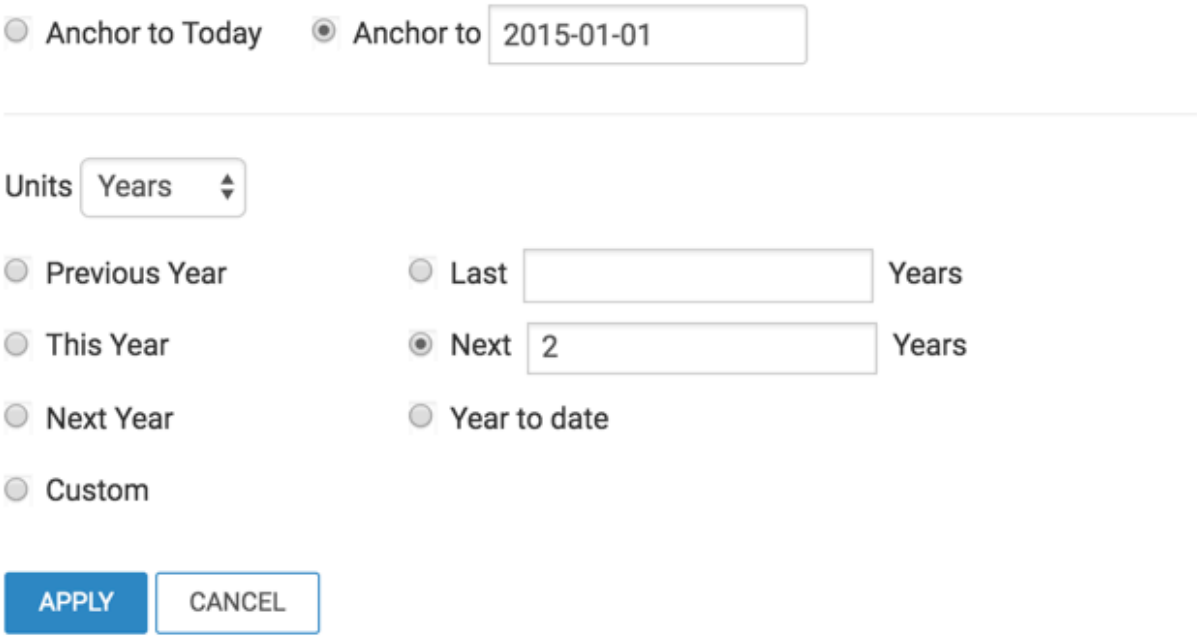

5. Anchor the date to 2015-01-01 and limit the visual output to the next two years from this date:
   - Select Anchor to and click in the date field to select 2015-01-01 from the calendar interface.
   - The other option is Anchor to Today (default).
6. Under Units, select Years.

   The other options are:
   - Days
   - Weeks
   - Months
   - Quarters

7. Select Next and enter '2' in the field next to it.

   The other options are:

   - Previous Year
   - This Year
   - Next Year
   - Custom
   - Last <number> Years
   - Next <number> Years
   - Year to date

8. Click APPLY.

Results

The visual output is now limited to two years, starting from the anchor year of 2015 and the application filter shows the new date range of 2015-01-01 - 2016-12-31.

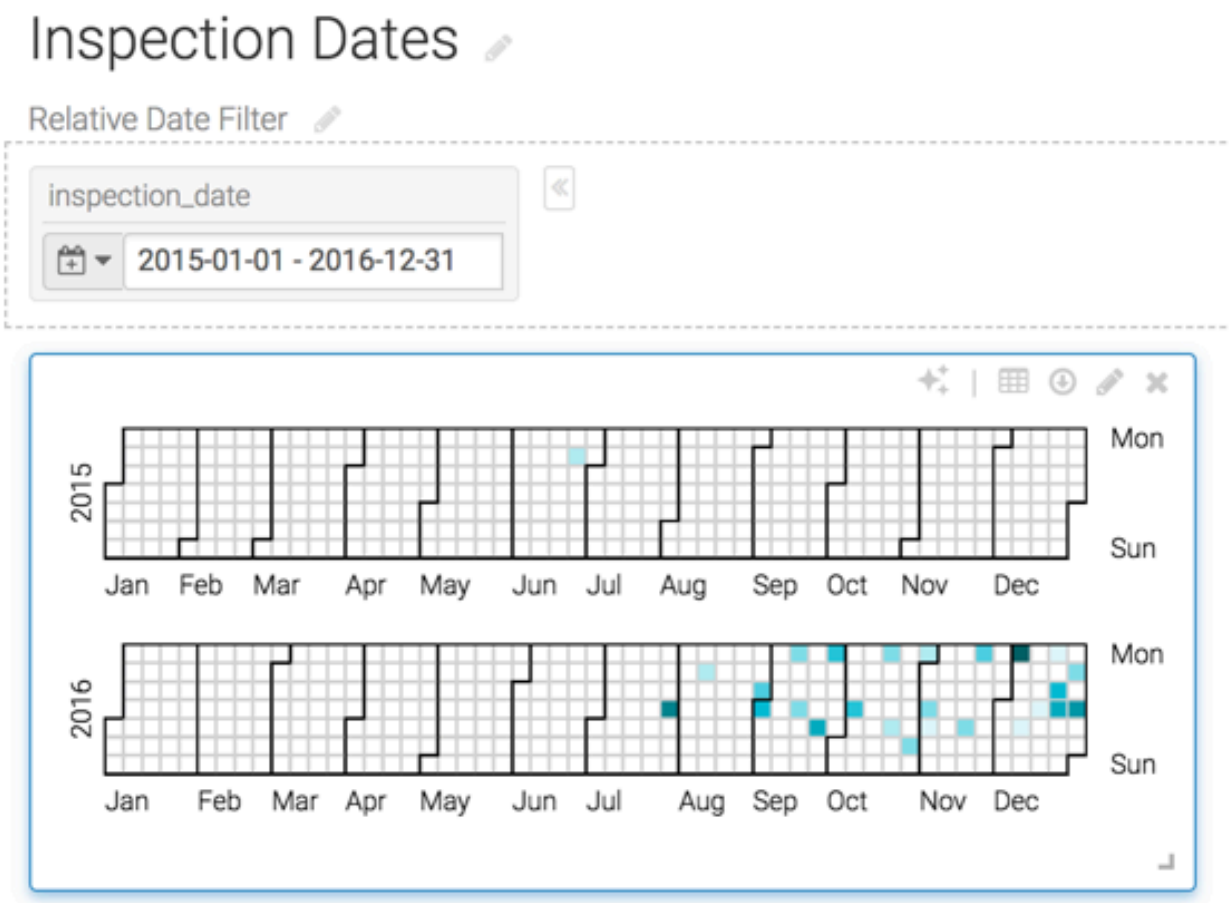

Using quick filters

In Cloudera Data Visualization, the quick date filter enables you to provide a quick selection of date ranges for visualizing your data.

About this task

The following steps show you how to use the quick filters option for a date and time filter. The instructions use a calendar heatmap visual as an example.

Before you begin

You have created a dashboard with quick filters. For instructions, see Configuring ranges on a date/time filter.

Procedure

1. Open your dashboard in Edit mode.
2. Click the Calendar icon on the filter and select Quick filters from the menu.

   The appearance of the filter changes to show the set of custom date ranges.

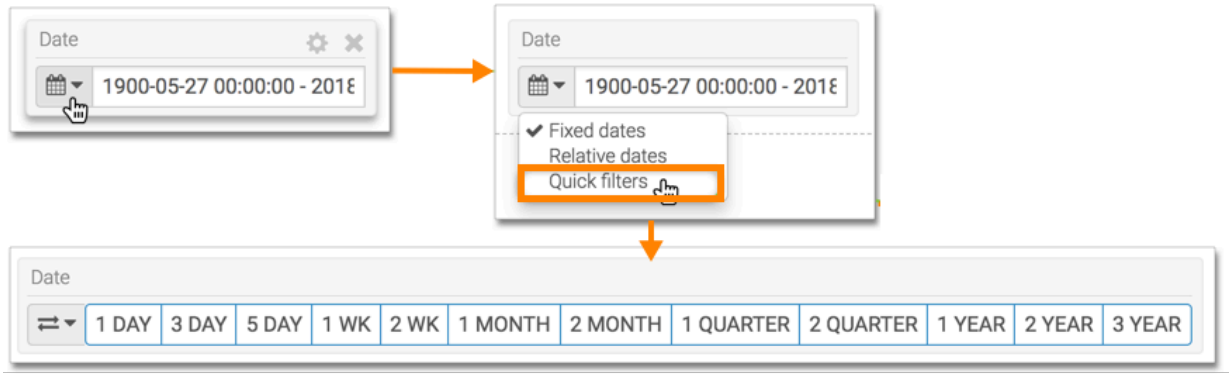

3. To use the quick filter, save the dashboard and switch to View mode.
4. Explore how the visual display changes when you click the different filter ranges.

Example

In these examples, you can see how the visual changes when the following filter date ranges are applied: none (all data), 3 years, and 1 quarter.

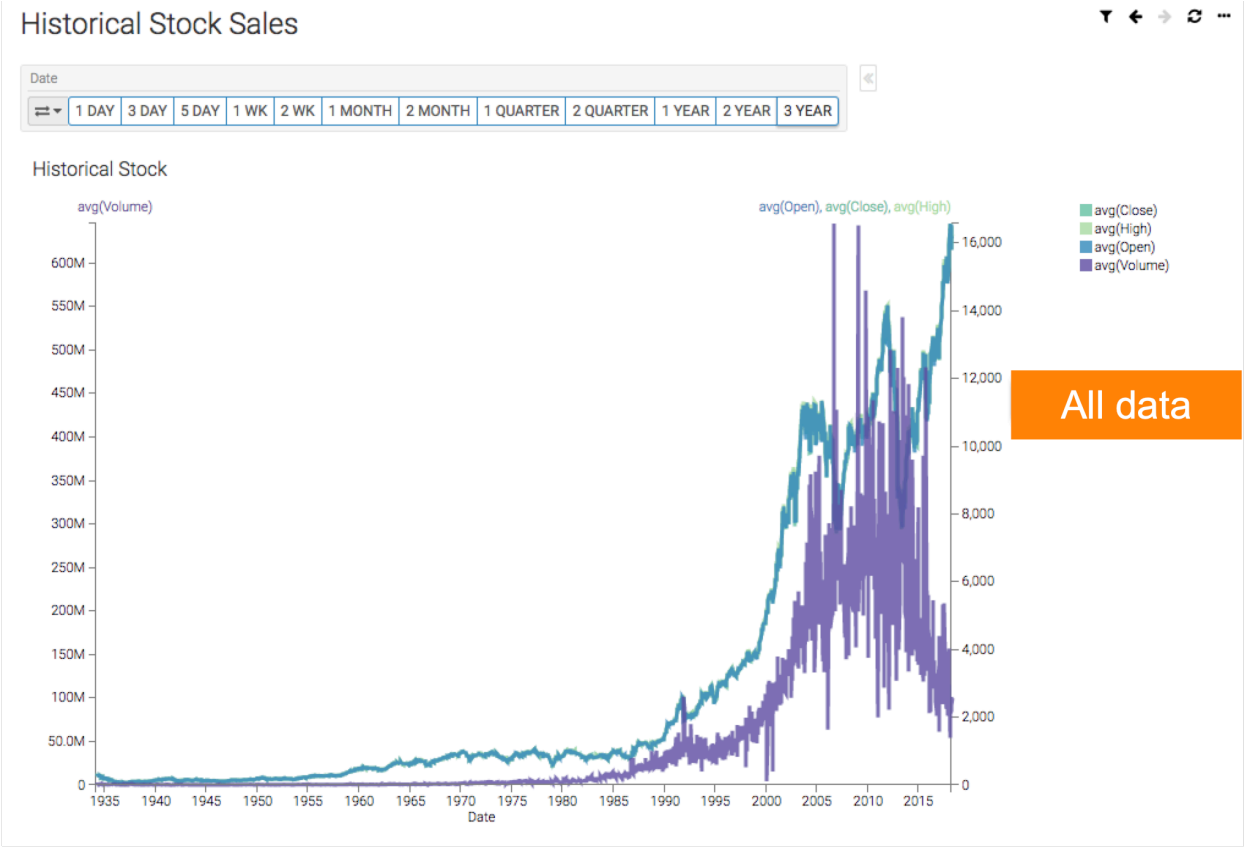
Historical Stock Sales
Date
1 DAY 3 DAY 5 DAY 1 WK 2 WK 1 MONTH 2 MONTH 1 QUARTER 2 QUARTER 1 YEAR 2 YEAR 3 YEAR
Historical Stock
avg(Volume)
avg(Open), avg(Close), avg(High)
avg(Close)
avg(High)
avg(Open)
avg(Volume)
All data
Date

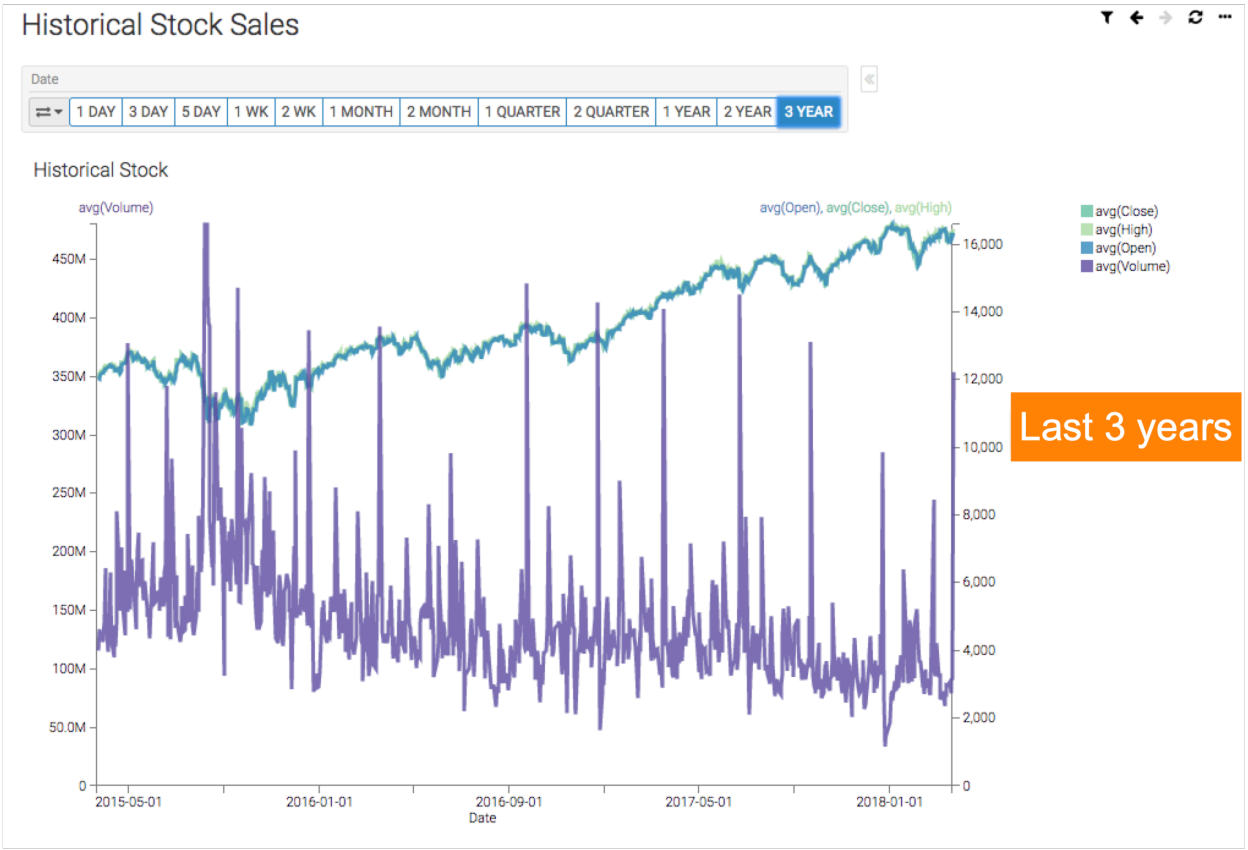
Historical Stock Sales
Date
1 DAY 3 DAY 5 DAY 1 WK 2 WK 1 MONTH 2 MONTH 1 QUARTER 2 QUARTER 1 YEAR 2 YEAR 3 YEAR
Historical Stock
avg(Volume)
avg(Open), avg(Close), avg(High)
avg(Close)
avg(High)
avg(Open)
avg(Volume)
Last 3 years
Date

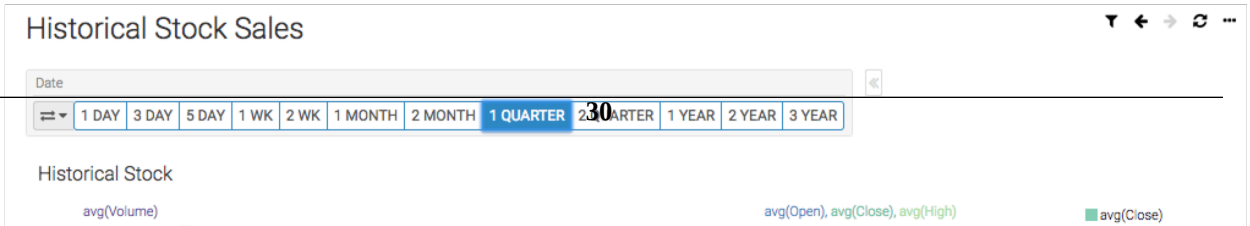
Historical Stock Sales
Date
1 DAY 3 DAY 5 DAY 1 WK 2 WK 1 MONTH 2 MONTH 1 QUARTER 1 YEAR 2 YEAR 3 YEAR
Historical Stock
avg(Volume)
avg(Open), avg(Close), avg(High)
avg(Close)

Related Information

Managing custom dates
Configuring ranges on a date/time filter

Including null values in date filters

The date filter enables you to select null values for visualization. The following instructions use a Scatter visual, which is built on the Issue Tracker dataset. This dataset includes issues that are created and resolved and also issues that are created but still open. Resolved issues have a timestamp and open issues have a 'null' value. By default, visuals only display values with a timestamp. This feature enables you to configure the date filter and display 'null' values in the visuals.

Creating a dashboard with date filters

The quick date filter enables you to provide a quick selection of date ranges for visualizing your data. This example shows a dashboard with a Scatter visual that plots created and resolved issues.

Procedure

1. Create a new dashboard and name it 'Issue Completeness'.
2. In the Dashboard Designer, create a Scatter visual based on the Issue Tracker dataset.
3. Populate the shelves from the available Dimensions and Measures fields in the DATA menu.
   a. Under the X shelf, add created from Measures.
   b. Under the Y shelf, add resolved from Measures.
   c. Click the field to open the Field Properties menu and make the following changes:
      - Select Enter/Edit Expression, enter the following expression, and click SAVE:

        ```
        if([resolved] IS NOT NULL,datediff([resolved],[created]),datediff(no
        w(),[created]))
        ```
      - Select Alias and name the field 'Days Open'.
   d. Under Colors, add assignee from Dimensions.
   e. Under Tooltips, add the following fields:
      - Description
      - Issue Key
      - Assignee
      - Date Open
      - Date Closed
   f. Under Filters, add created and resolved from Dimensions.
4. Click REFRESH VISUAL.
5. Click SAVE.

   The visual appears in the main area of the dashboard interface.
6. Add two date filters, created and resolved in the dashboard, with a date range from 2018-06-25 to 2018-09-12.

   For information on how to create date filters in a dashboard, see *Using fixed dates (date range picker) mode*.
7. Click SAVE.

Results

The following image shows the new dashboard you have created. This visual does not include the null values in the dataset.

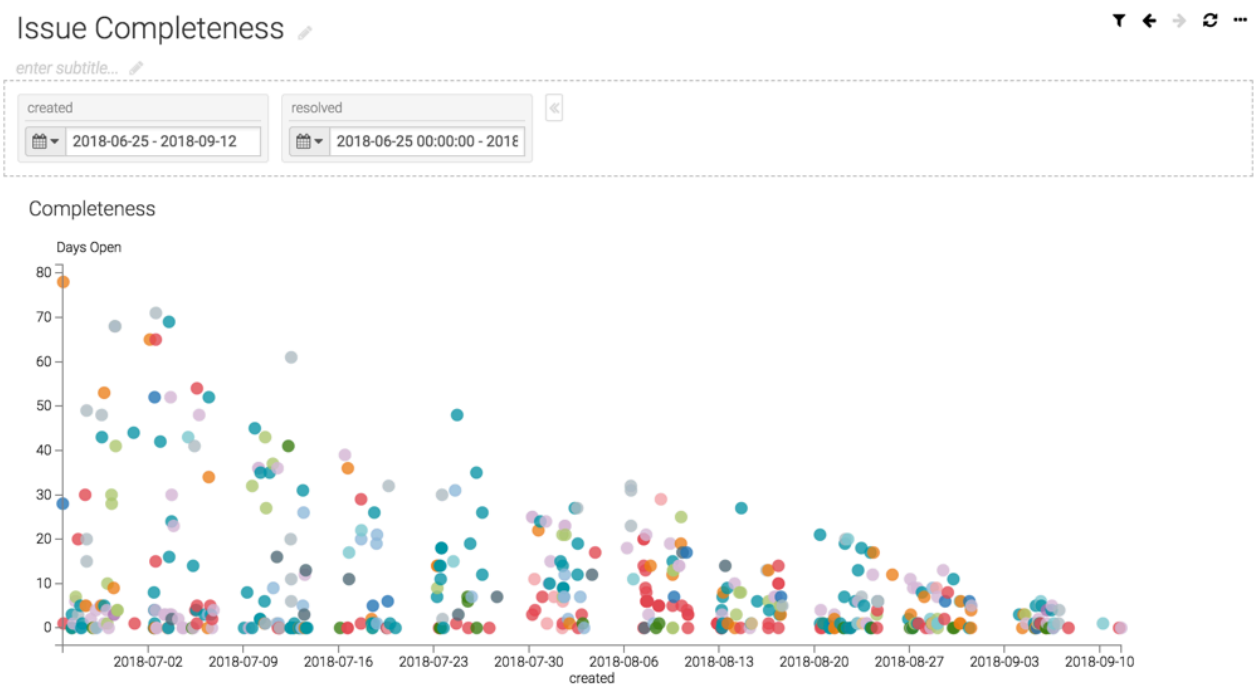

What to do next

Proceed to the next section to learn how to include null values in a visual.

Including null values in a visual

The quick date filter enables you to provide a quick selection of date ranges for visualizing your data.

Before you begin

You have created dashboard with a scatter visual that plots created and resolved issues.

Procedure

1. Hover over the top right corner of the resolved filter and click

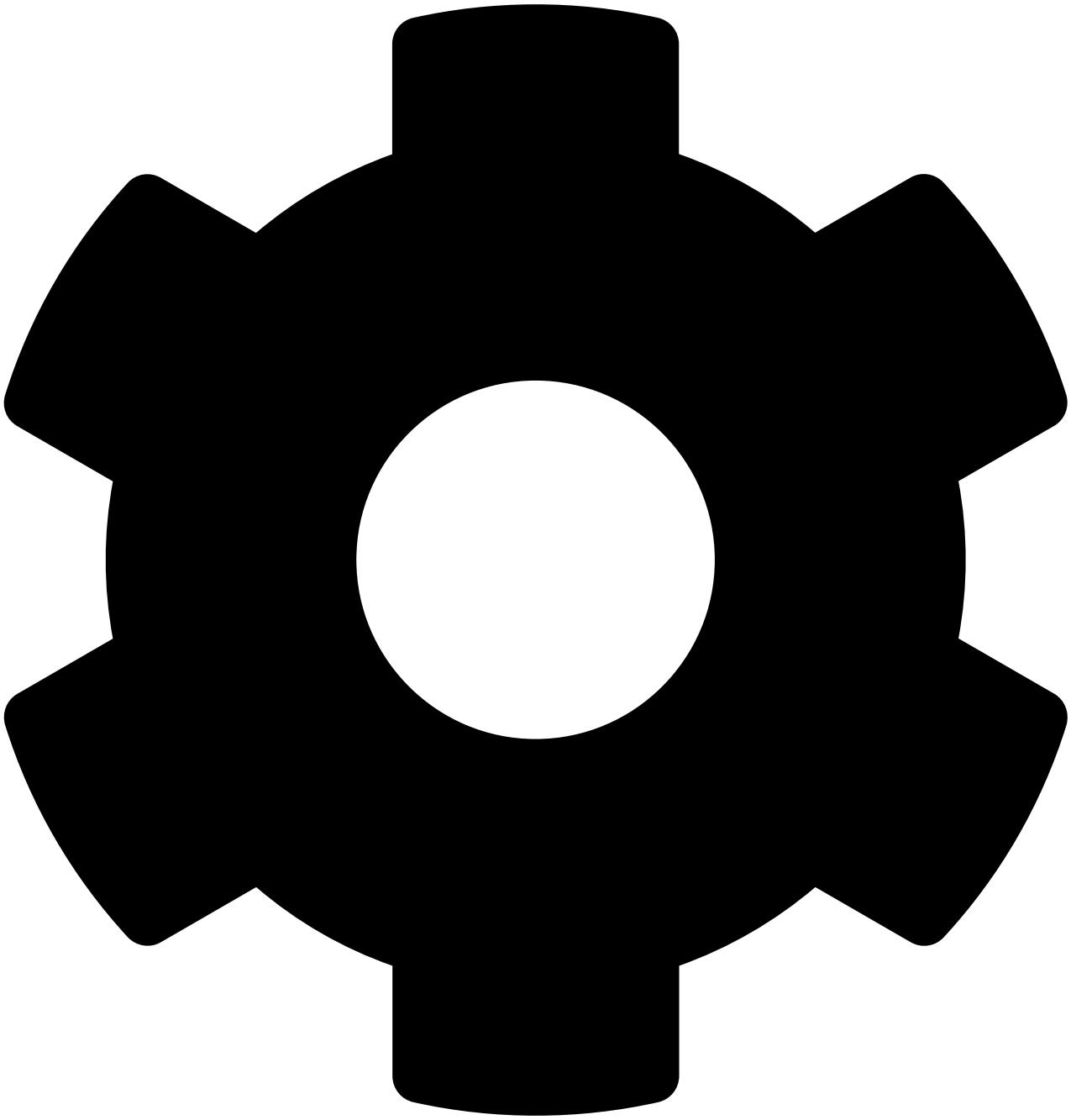

to configure the filter.

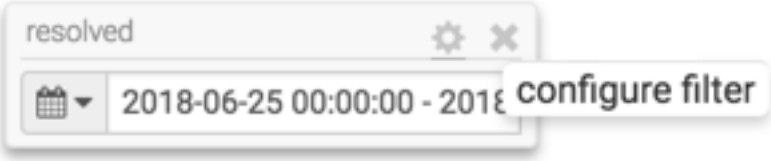

2. On the Values tab of the Settings modal window, select the Permit selection of NULL values option.

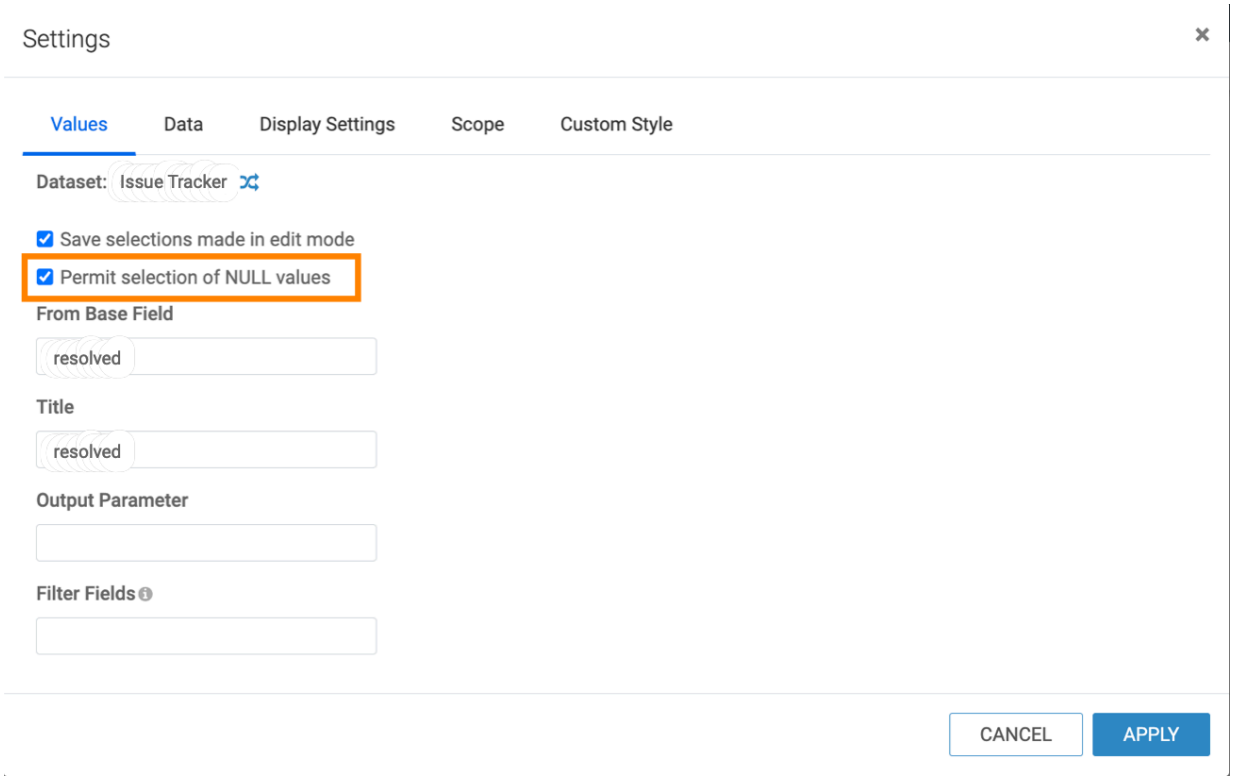

3. Click APPLY.

4. Click the Calendar icon on the resolved filter.

   The Include NULLs option appears in the drop-down menu.

   Note: If the Permit selection of Null values option is not selected in the previous step, the Include NULLs option is not visible in the dropdown.

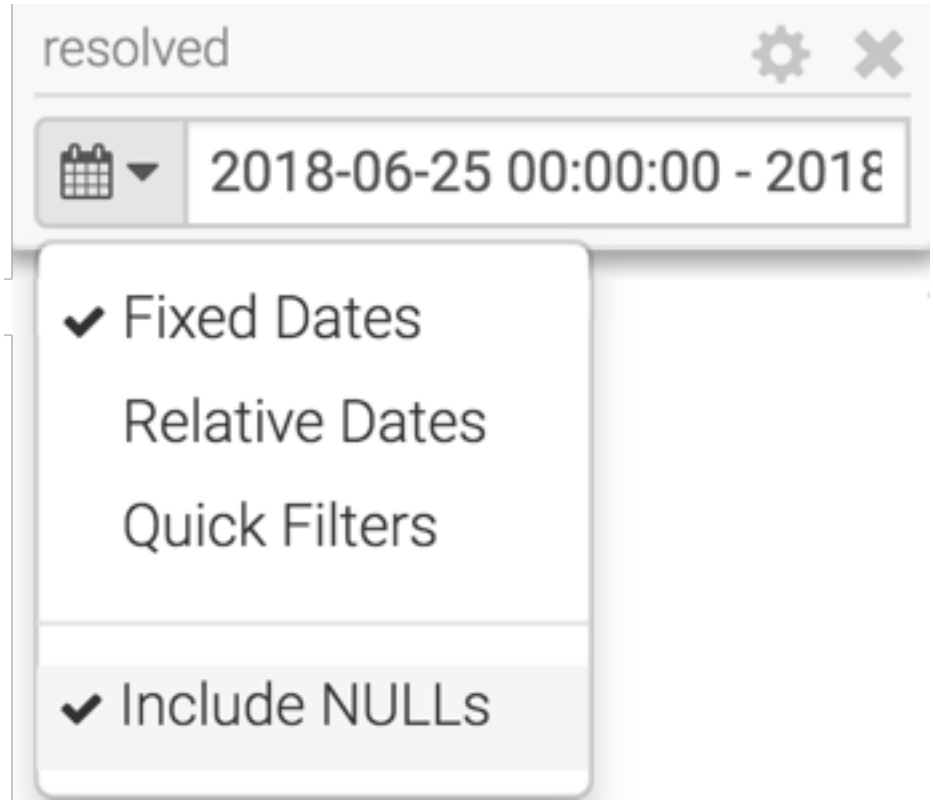

5. Close the drop-down menu without selecting the Include NULLs option.

   The following image shows that the Current Parameters on the top right corner shows resolved.include_nulls: false, indicating that the Include NULLs option was not selected in the filter.

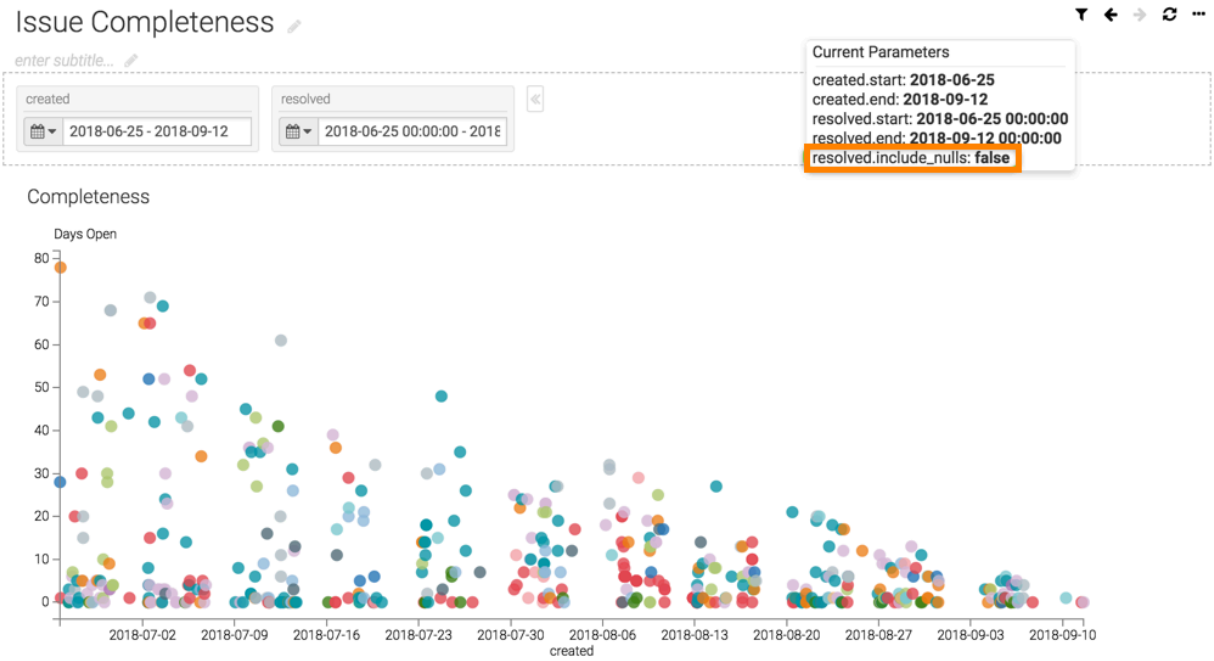

6. Click the Calendar icon on the resolved filter and select the Include NULLs option.

   The following image shows that the Current Parameters modal in the top right corner shows resolved.include_nulls: true, indicating that the Include NULLs option was selected in the filter. The visual now includes the null values in the dataset.

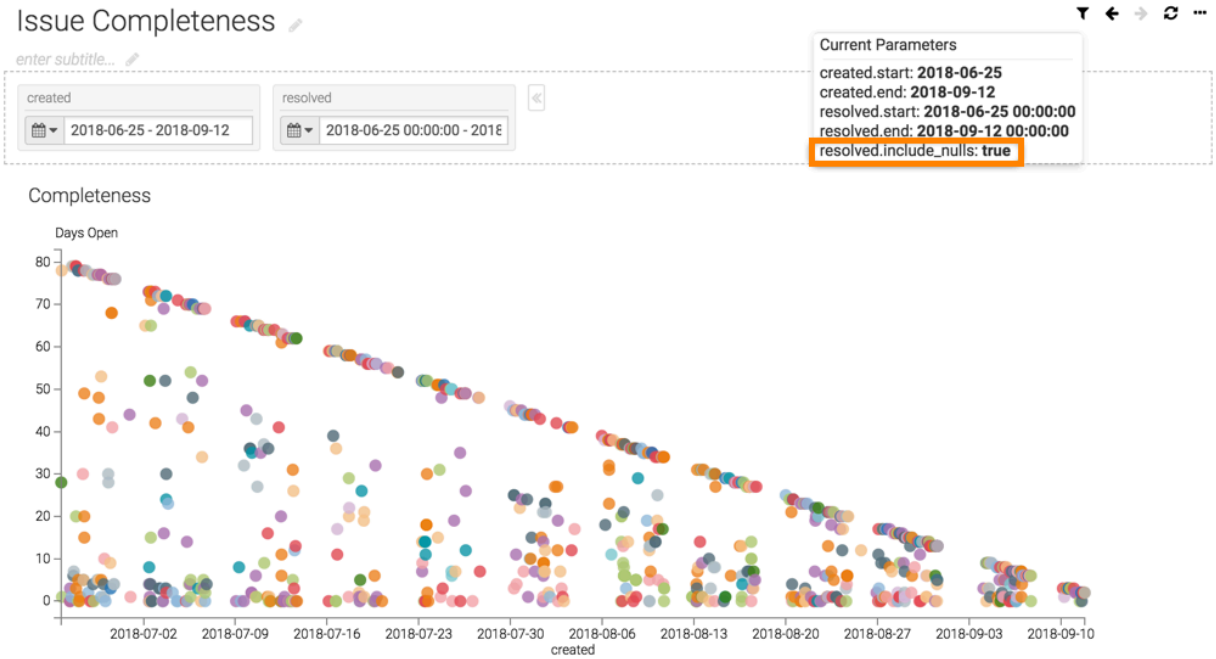

7. To verify that the visual includes null values, hover over one of the selections to view the tooltip.

   The following image shows that Date Closed has a 'NULL' value.

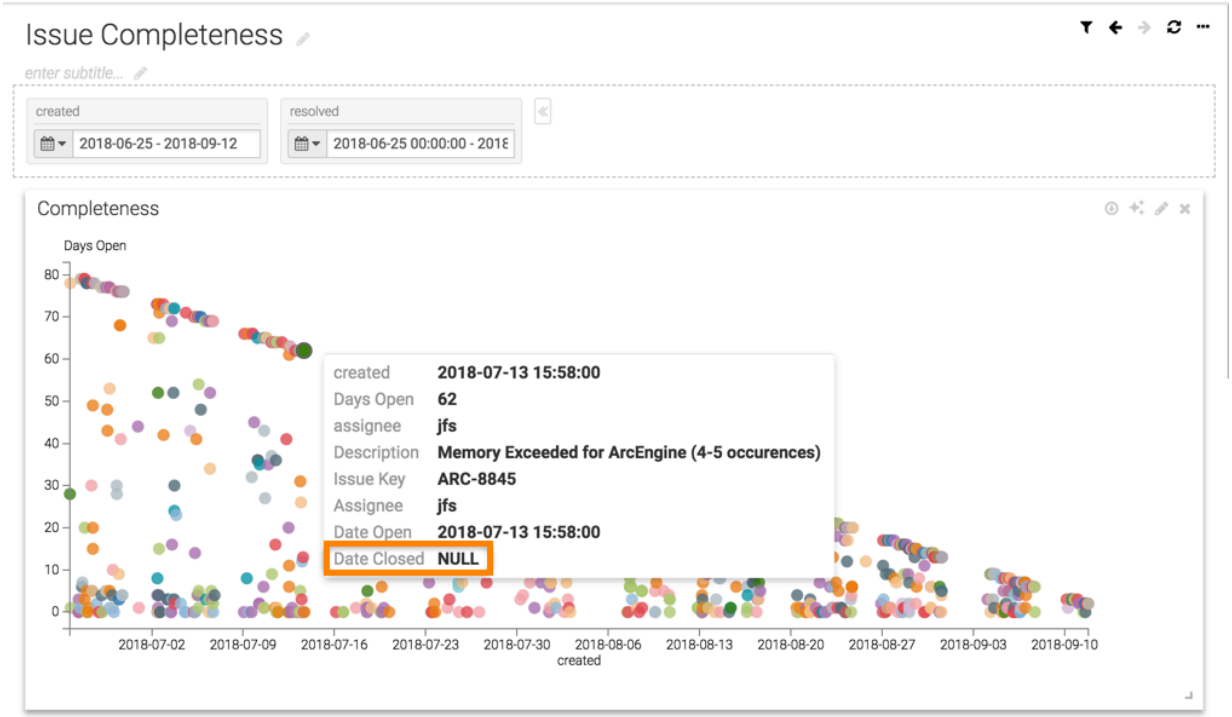

Setting parameters through URL

In Cloudera Data Visualization, setting parameters through the URL allows you to dynamically add selection options without altering the app's saved values. These settings can also be easily shared by including them in emails, documents, or similar mediums.

About this task

Important: URL strings must use single quotes (be a valid JSON array), and usual character limitations on URL addresses apply.

The basic syntax for URL formatting:

```
http://baseurl/arc/apps/app/id?config.output={setting: value}
```

baseurl

The IP address of the site, such as 27.0.0.1:8000, or sf.mycompany.org.

id

Unique number of the dashboard or the visual.

output

The output of the parameter or the picklist.

setting

The available option is VALUE.

value

An array of value-label pairs in the format [["value1","label1"],["value2","label2"],["value3","label3"], ...,].

The following steps demonstrate how to use parameters through URLs:

Procedure

1. On the Values tab of the Settings modal window, specify the Title and Output Parameter.

   In this example, test and result are used, respectively.
2. You can specify one or more Value:Label pairs.

   In this example, result1:1 is used.

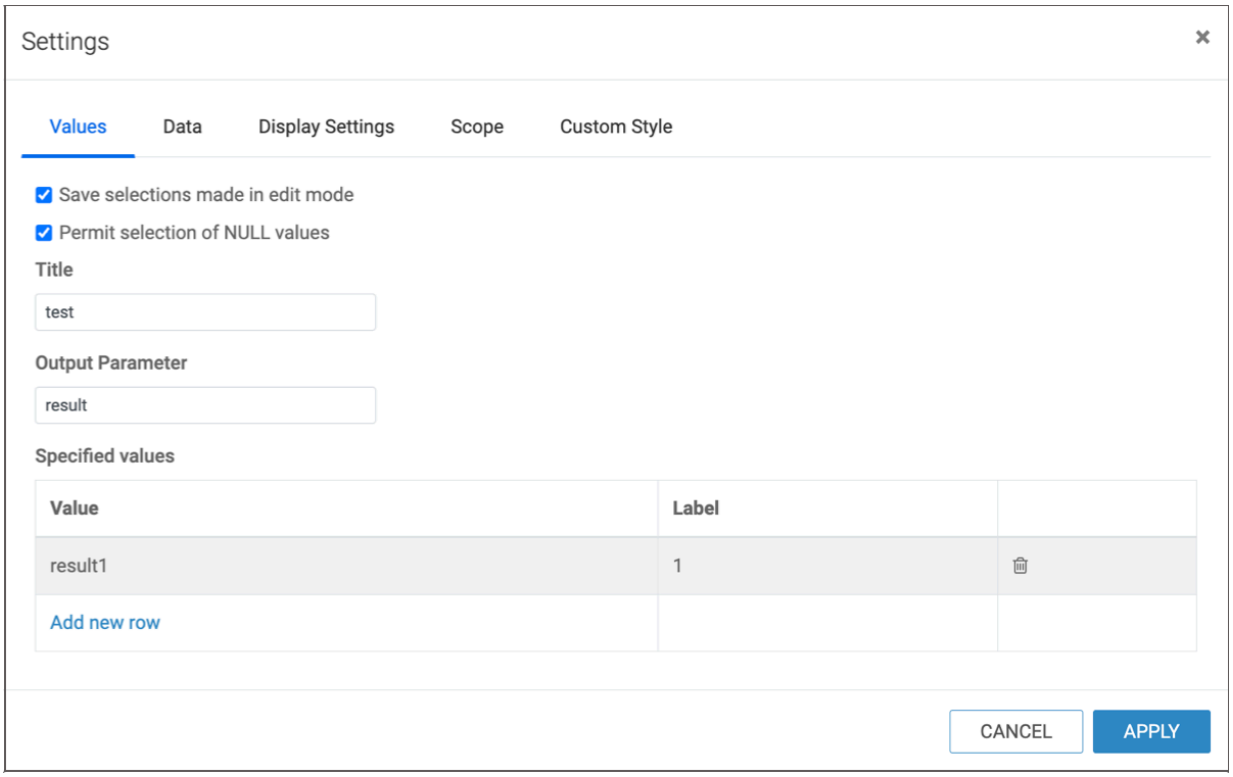

3. Click APPLY.
4. In the dashboard, check the possible options for the parameter.
5. In the test parameter, click the Down Arrow icon.

   There are two options: (All) and 1.

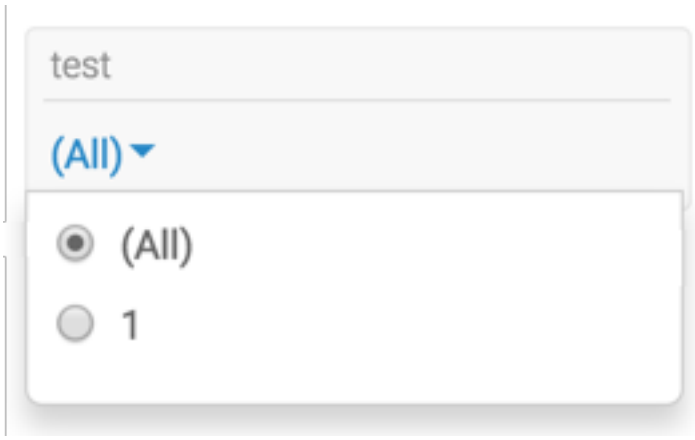

6. Select 1 and hover the pointer over the Filter icon at the top right of the dashboard.

   The Current Parameters are set to result: result1 and result.alias: 1.

7. Save the dashboard.
8. Switch to View mode.
9. Copy the URL.

   In this example, it is http://127.0.0.1:8000/arc/apps/app/38.
10. In another browser window, paste the value saved in the previous step, add the code specific to setting parameters through URL, and click enter.

    In this example, two more pairs: result2:2, and result3:3 are added. The new code is in bold font, starting with '?'.

```
http://127.0.0.1:8000/arc/apps/app/38?config.result={values:
[["result2","2"],["result3","3"]]}
```

11. In the dashboard, check the possible options for the parameter.

    Options 2 and 3 appear in the selection list.
12. In the test parameter, click Select.

    There are two additional options: 2 and 3.

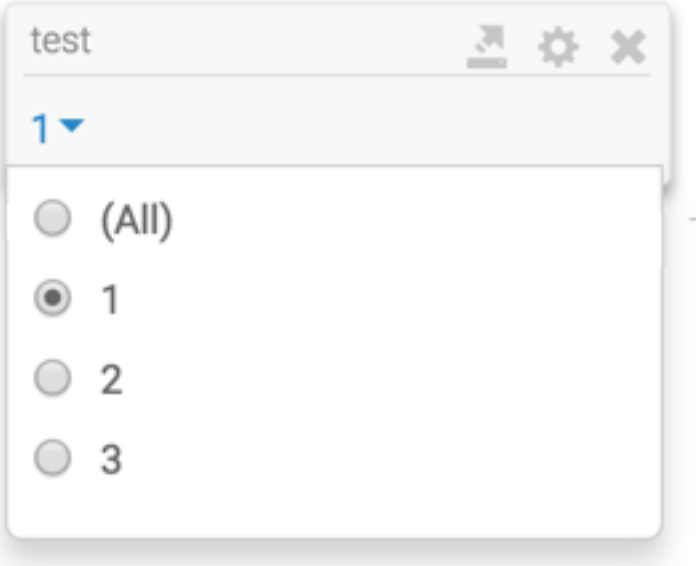

13. Select 2 and hover the pointer over the Filter icon at the top right of the dashboard.

    Current Parameters are set to result: result2 and result.alias: 2.

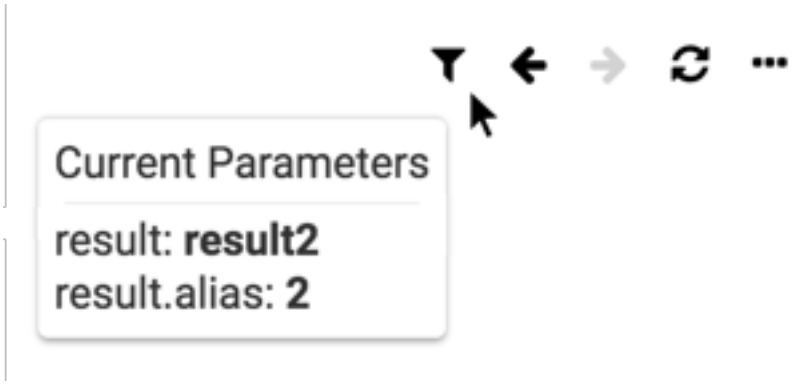

Customizing output parameters

In Cloudera Data Visualization, application controls can use custom output parameters. You can use custom output parameters for application controls. This allows you to apply a filter from one dataset to control the visual presentation of another dataset.

About this task

Often, the field names between the source (driver) and target datasets do not match. You can resolve this by mapping the filter's value to the correct field in the target dataset.

The following steps show you how to configure a filter based on the US State Populations Over Time dataset to control a visual based on the US County Population dataset.

Procedure

1. Open an existing dashboard in Edit mode or create a new one.
2. Ensure that includes visuals from both datasets. If not, click the Visuals tab on the left navigation and add the required visuals.
3. Click the Filters tab and select the US State Populations Over Time dataset.
4. Under Dimensions, click the state field to add it as a filter.

5. Hover over the added filter until controls appear in the top right corner, then click

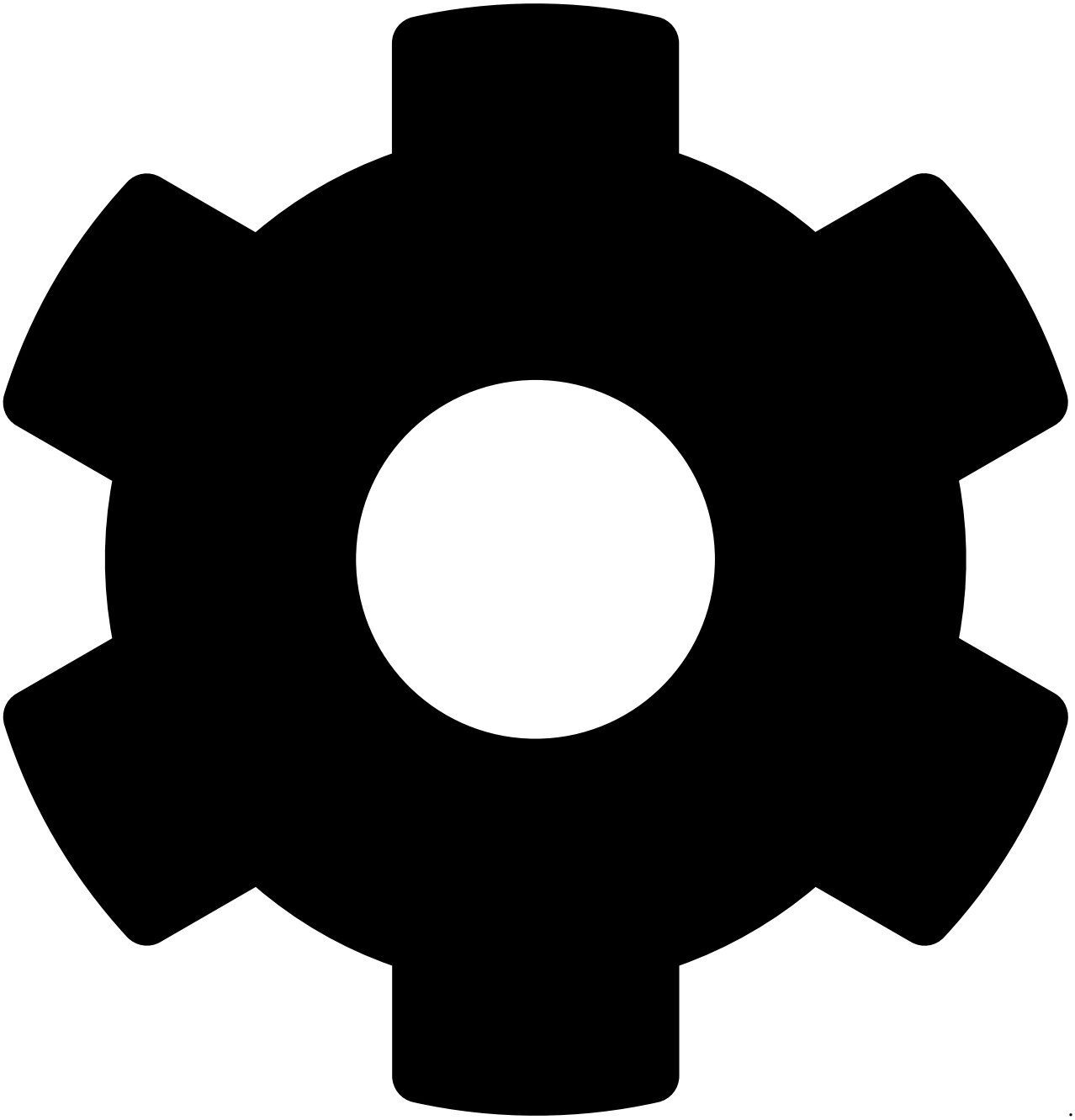

The Settings modal window opens on the Values tab.

6. Configure the following values:
   - Title: Set to States.
   - Output Parameter: Set to stname.

   Note: From Base Field is set to state by default. Do not change this setting!

7. Click APPLY.
8. In the state filter, click Select and chose the following:
   - AK
   - AL
   - AR

9. In the top right corner of the application, hover over to review the current filter parameters.

This reports the current filtering parameters. The output parameter is correctly named stname, and the filtering conditions applied earlier are displayed.

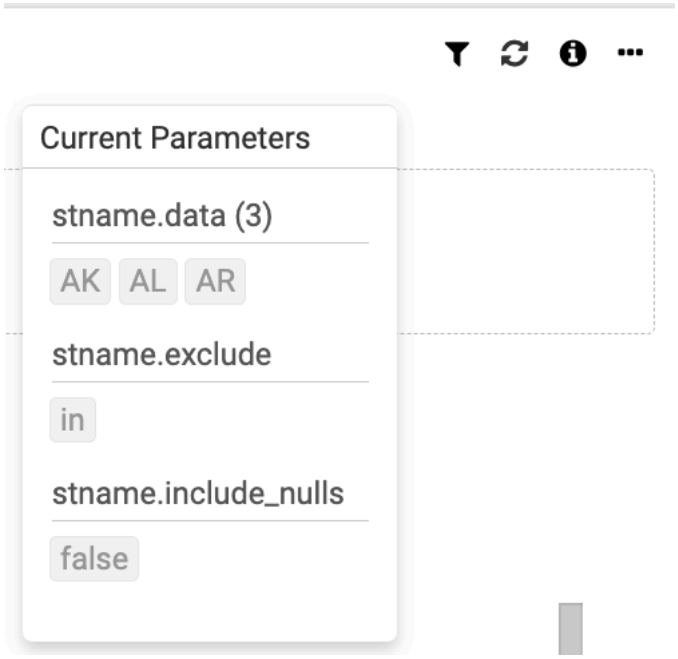

If the Emit distinct parameters for each item selected option is selected on the Display Settings tab, the filter reports each selection as distinct key/value pairs. By default, the label for each key is derived from its value. However, if labels are specified (for example, when using optional parameters), they will be displayed instead.

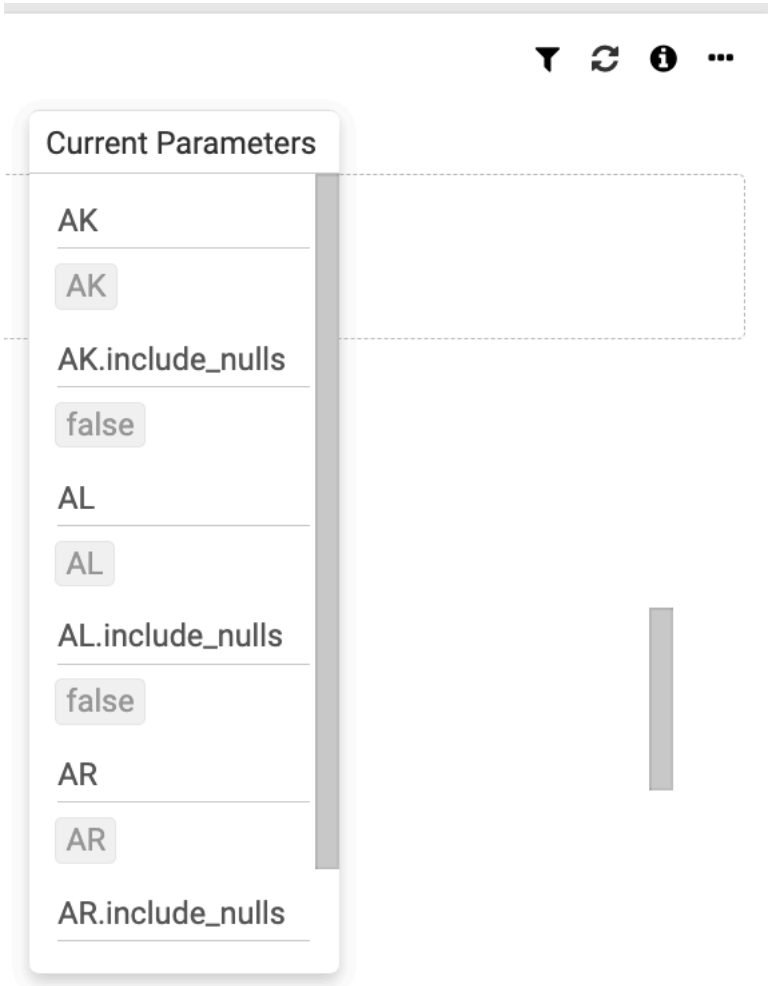

Configuring ranges on a date/time filter

In Cloudera Data Visualization, you can easily configure real-time filters that enable you to navigate across visuals built on time series. Both historical data and ongoing data feeds are supported.

Before you begin

You have created a date/time filter widget. For instructions, see Creating date/time filters on a dashboard.

Procedure

1. Open the dashboard in Edit mode.

2. Hover over the upper right corner of the filter and click

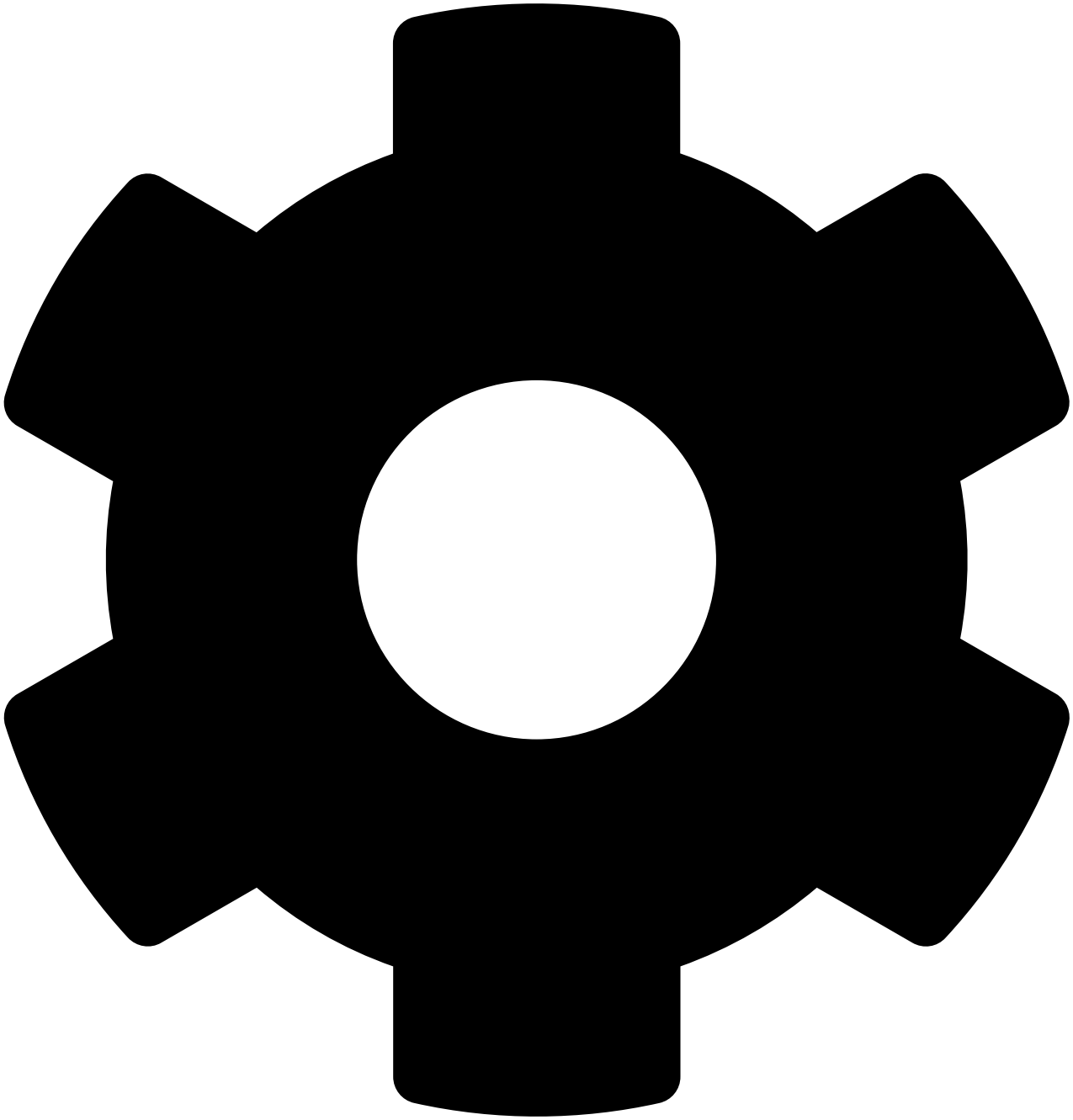

to configure the filter.

The Settings modal window opens on the Basic tab.

3. Select the option Permit selection of time range.
4. Click the Ranges tab and select the the appropriate range.
5. Click APPLY.

6. On the filter, click the Calendar icon and select the Quick filters option from the menu.

   The appearance of the filter changes to show the set of defined date ranges.

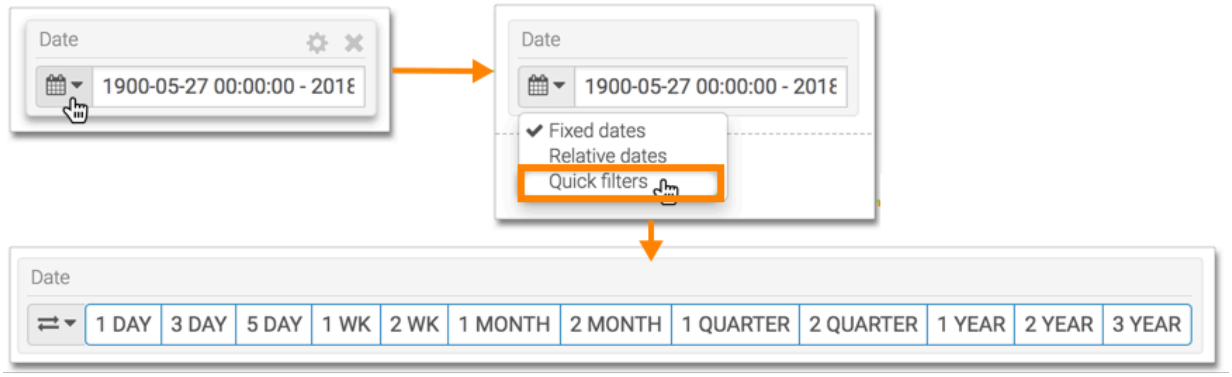

   For instructions on how to use the quick filter, see *Using quick filters*.

Related Information

Using quick filters

Configuring data options for a dashboard filter

Cloudera Data Visualization allows you to configure data options of a newly created field-based or custom filter in a dashboard.

Procedure

1. Click

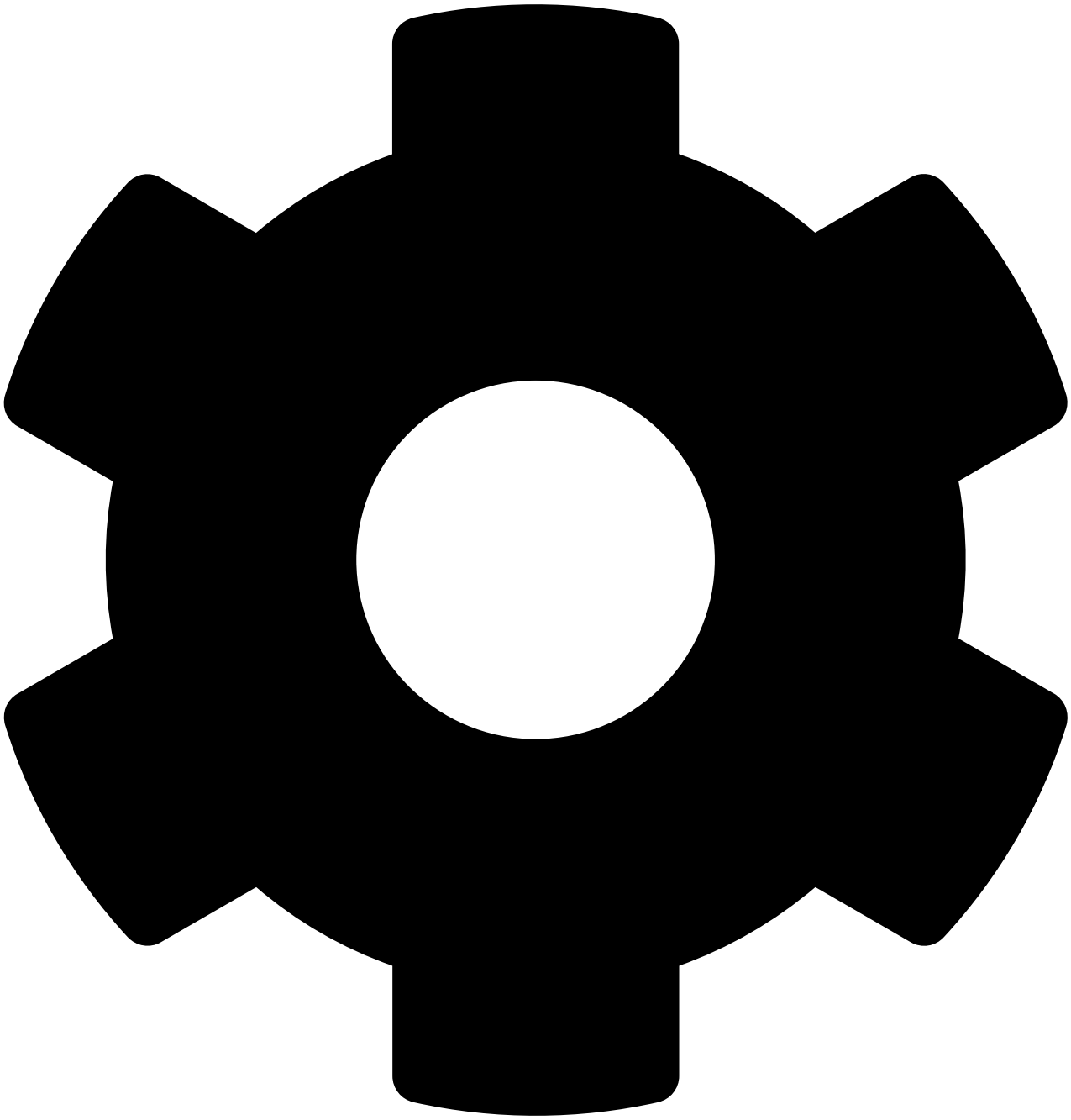

   on the filter widget.

   The Settings modal window opens on the Values tab.
2. Click the Data tab.
3. Configure the following settings.

Changing number of initial options in dashboard filters

In a dashboard filter, you can control the number of initial options that appear in the dashboard filter at loading time. This limits the display of options for datasets with a very large variability.

Procedure

1. Enter a value in the Maximum default number of values displayed.

   The default is 1000.
2. Click APPLY.

Changing number of options in search output of dashboard filters

In a dashboard filter, you can change the maximum number of options that appear during search. This limits the display of options for datasets with a very large variability.

Procedure

1. Enter a value in the Maximum number of search results displayed.

   The default is 1000.
2. Click APPLY.

Disabling incremental queries in filters

Cloudera Data Visualization allows you to disable incremental queries at the level of the dashboard filter. Incremental queries return preliminary results quicker and paints the visuals on the dashboard, and then adjusts the level of detail as more results are incorporated into the calculations. Disabling this feature delays the painting of the visuals until all results are in.

Procedure

1. Select the Disable incremental queries, overriding field-based-level setting option.
2. Click APPLY.

Configuring sort order in filter values

Cloudera Data Visualization allows you to select the sort order of filter values. There are different sort order options available on field-based filters and custom filters.

Procedure

1. Select the Filter Options Sort Order option.

   Sort Order of Filter Widget Values on Field-based Filters

   For a field-based filter, there are two sort options available under the Filter Options Sort Order setting:

   - Ascending
   - Descending

   By default, filter widget values are displayed in ascending order (image on the left). In this example, to see the most recent population of Australia first, select the Descending option (image on the right).

   By default, filter widget values are displayed in ascending order. In our example, to see the most recent population of Australia first, select the Descending option.

   Sort Order of Filter Widget Values on Custom Filters

   For a custom filter, there are three sort options available under the Filter Options Sort Order setting:

   - Ascending
   - Descending
   - Default

   The default setting of the sort order in a custom filter is Default. In the following image, notice that the default sort order in the filter widget is the same as the lables of the custom filter added in the Values tab: Country, Year, Region, and GDP.

   You can select the Ascending option to change the sort order to country, GDP, Region, and Year, or select the Descending option to change the sort order to Year, Region, GDP, and Country.

2. Click APPLY.

Configuring display settings for a dashboard filter

Cloudera Data Visualization enables you to specify display settings of a custom filter in the dashboard.

Procedure

1. Click

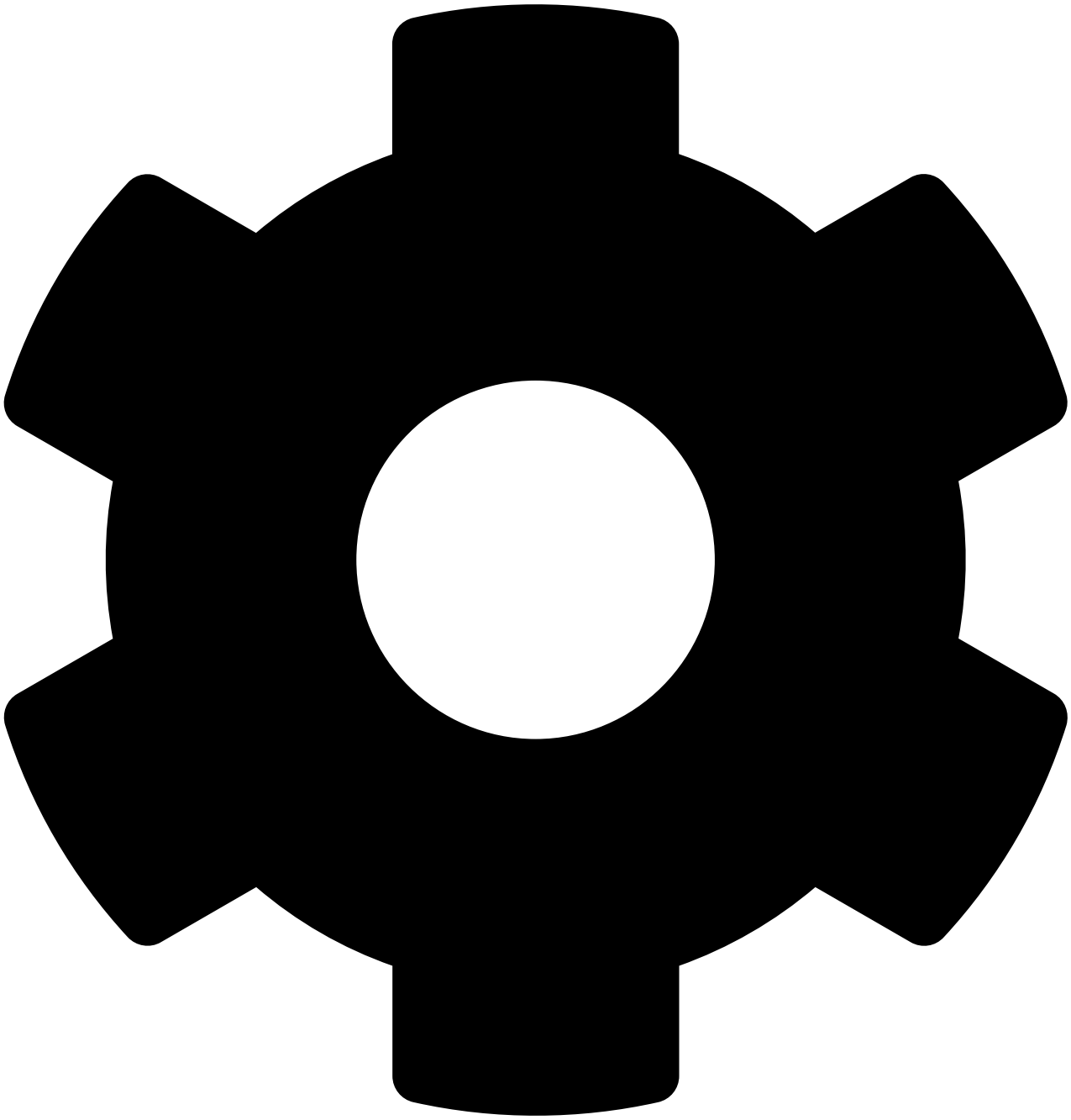

on the filter widget.

2. In the Settings modal window, click the Display Settings tab to examine the available options:

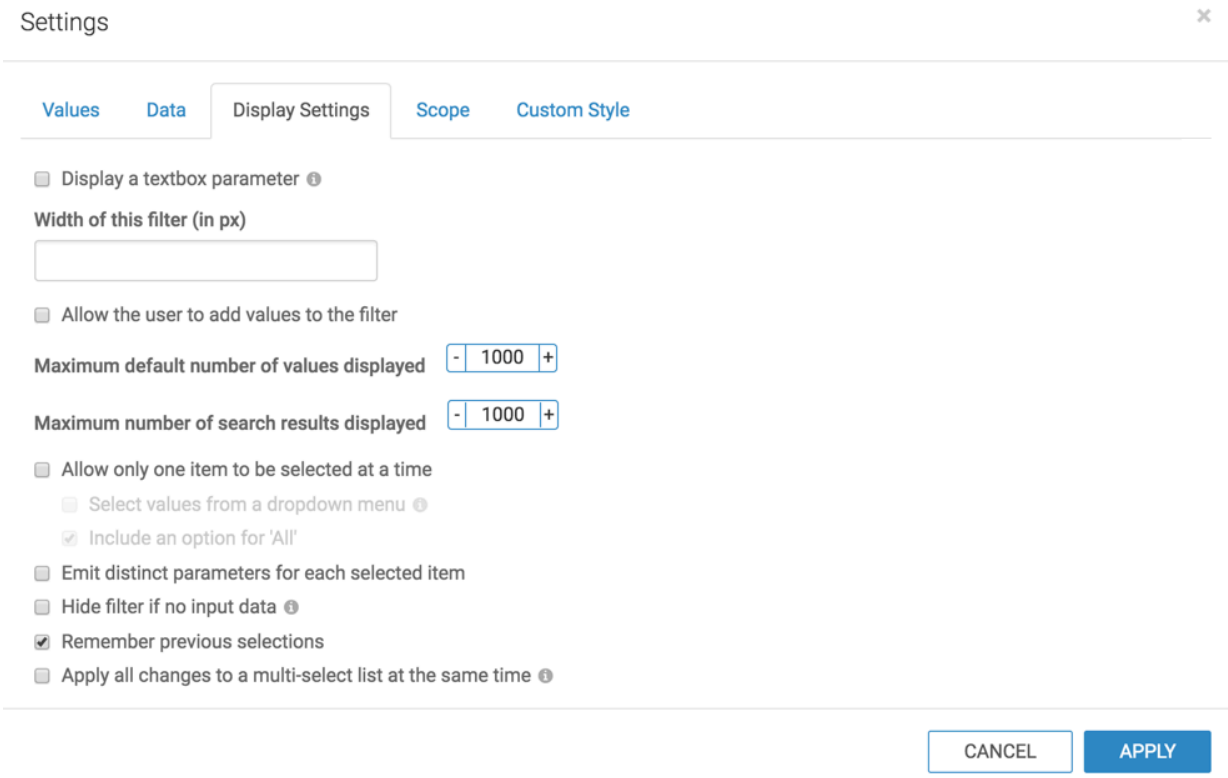

Displaying a textbox parameter

This option allows you to add a customizable parameter text box in a dashboard filter, enabling users to enter or search for specific values during runtime.

About this task

When enabled, the Display a text box parameter option overrides all predefined values included in the Specified values option on the Values tab of the Settings modal window. Instead of selecting from a list, you can enter a parameter value that you want to use at runtime, or search for data that matches certain values. It creates a simple search box filter.

Before you begin

You have added a filter widget to your dashboard.

Procedure

1. Open the dashboard in Edit mode.

2. Hover over the filter and click

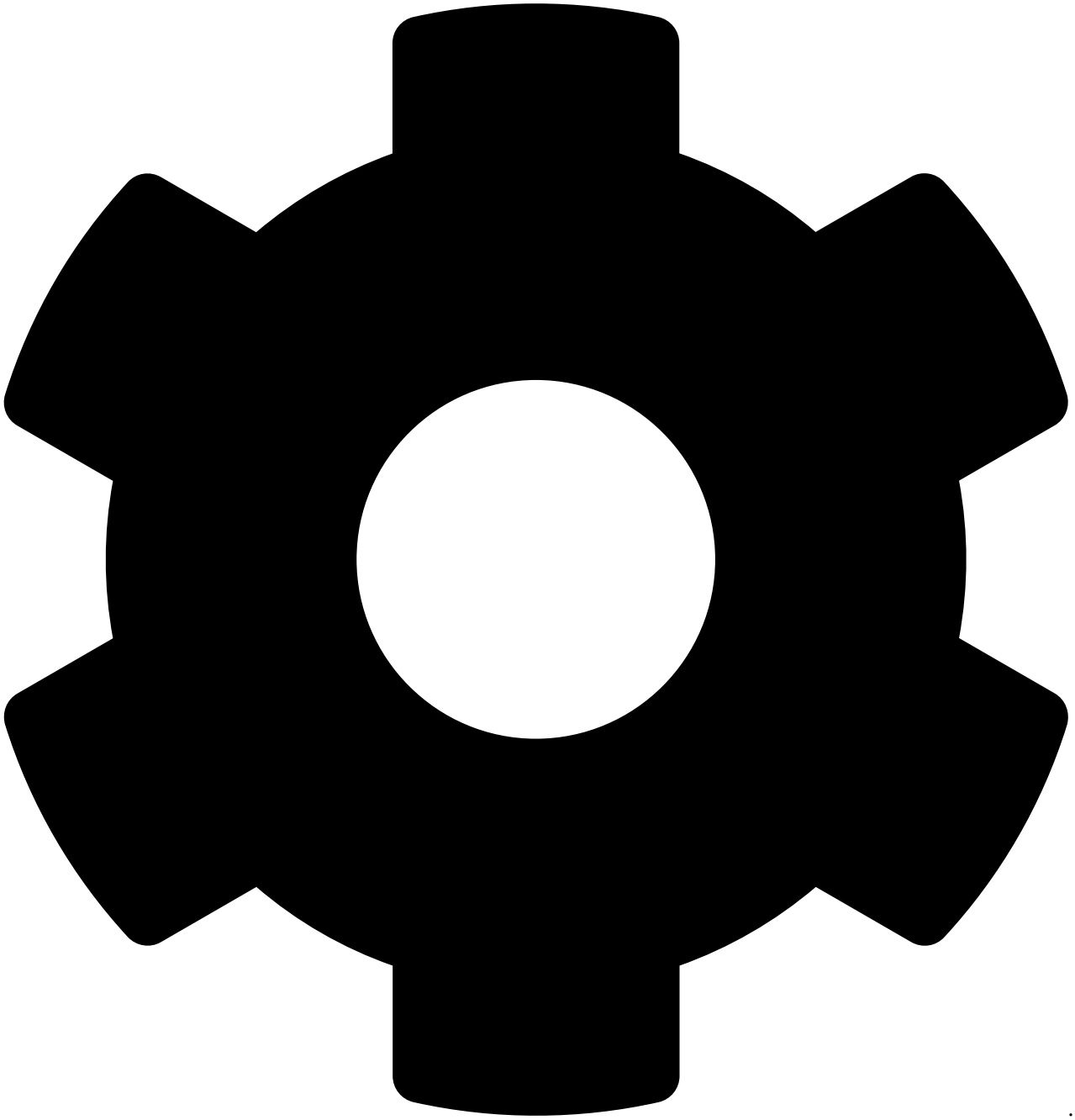

.

   The Settings modal window appears.
3. Go to the Display Settings tab and check the Display a text box parameter option.
4. Click APPLY.
   A parameter filter textbox appears as a filter.

5. Enter a value in the textbox.
   In this example All-Bran is added.

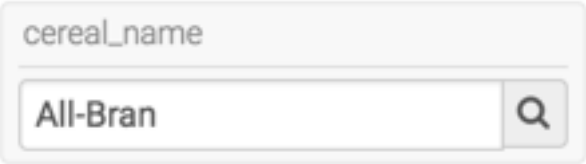

Results

The visual automatically gets updated and displays data related only to All-Bran.

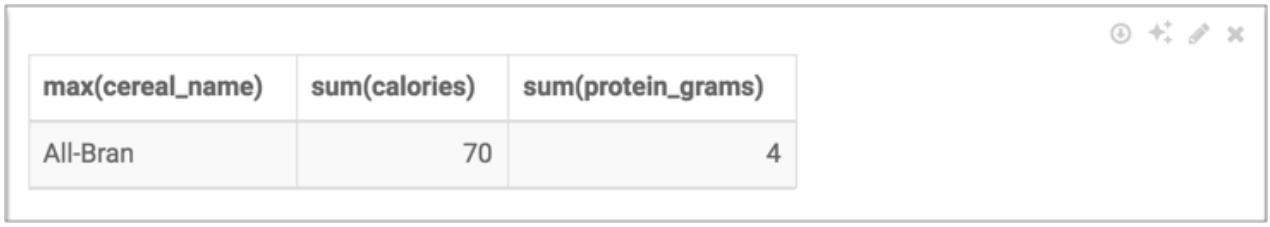

max(cereal_name)	sum(calories)	sum(protein_grams)
All-Bran	70	4

Changing the width of the filter text box

This option allows you to change the width of the filter text box in a dashboard filter.

Before you begin

You have added a filter widget to your dashboard.

Procedure

1. Open the dashboard in Edit mode.

2. Hover over the filter and click

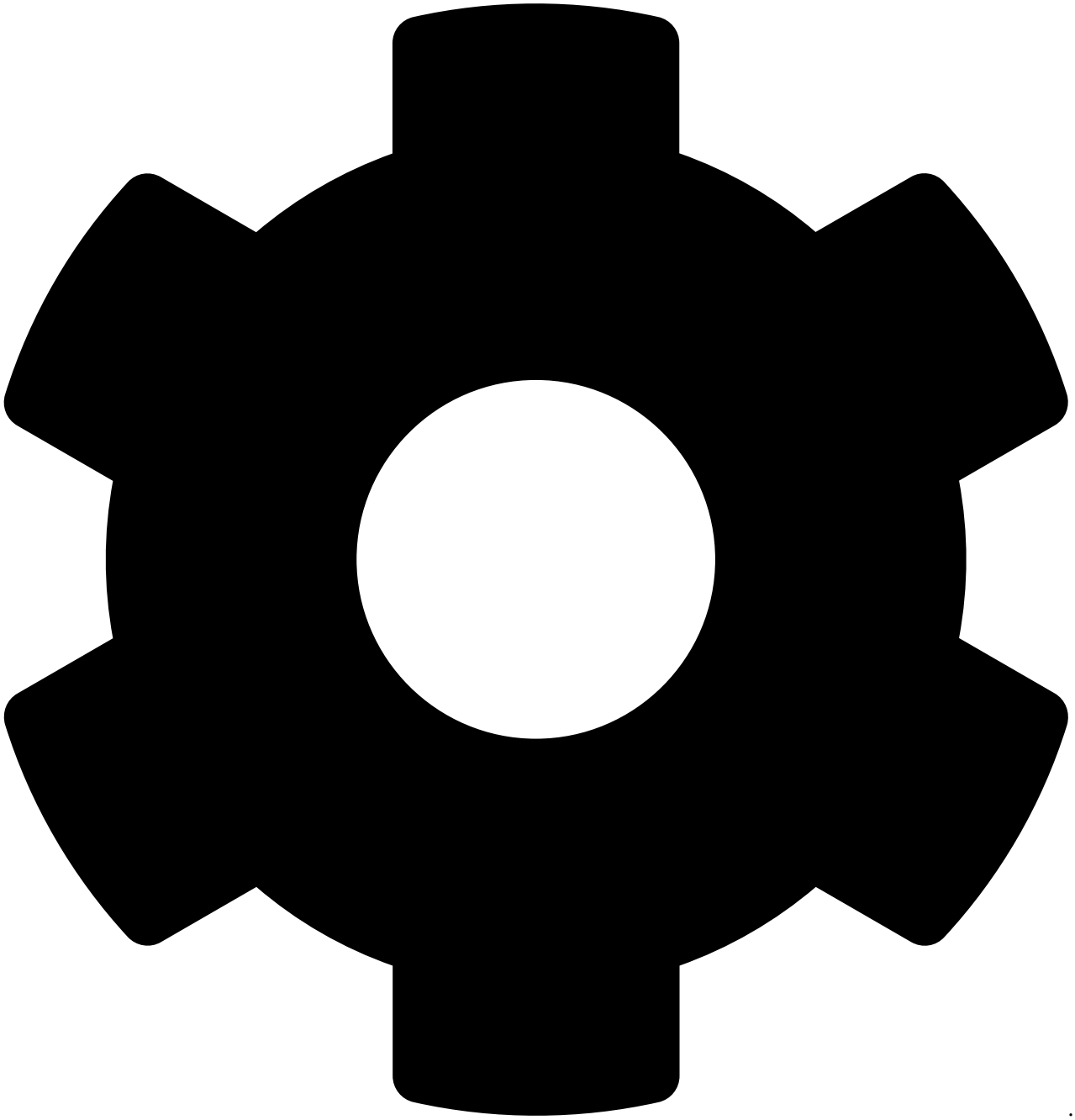

.

The Settings modal window appears.

3. Click Display Settings and enter a value in the Width (in filter mode) box to change the width of the filter box.
4. Click APPLY.

Entering values manually

This option allows you to change the display settings of a custom filter in a dashboard. You can select multiple values in bulk, instead of selecting them individually from a menu. For instance, you can copy multiple values from a table and paste them in a text box.

Before you begin

You have added a filter widget to your dashboard.

Procedure

1. Open the dashboard in Edit mode.
2. Hover over the filter and click

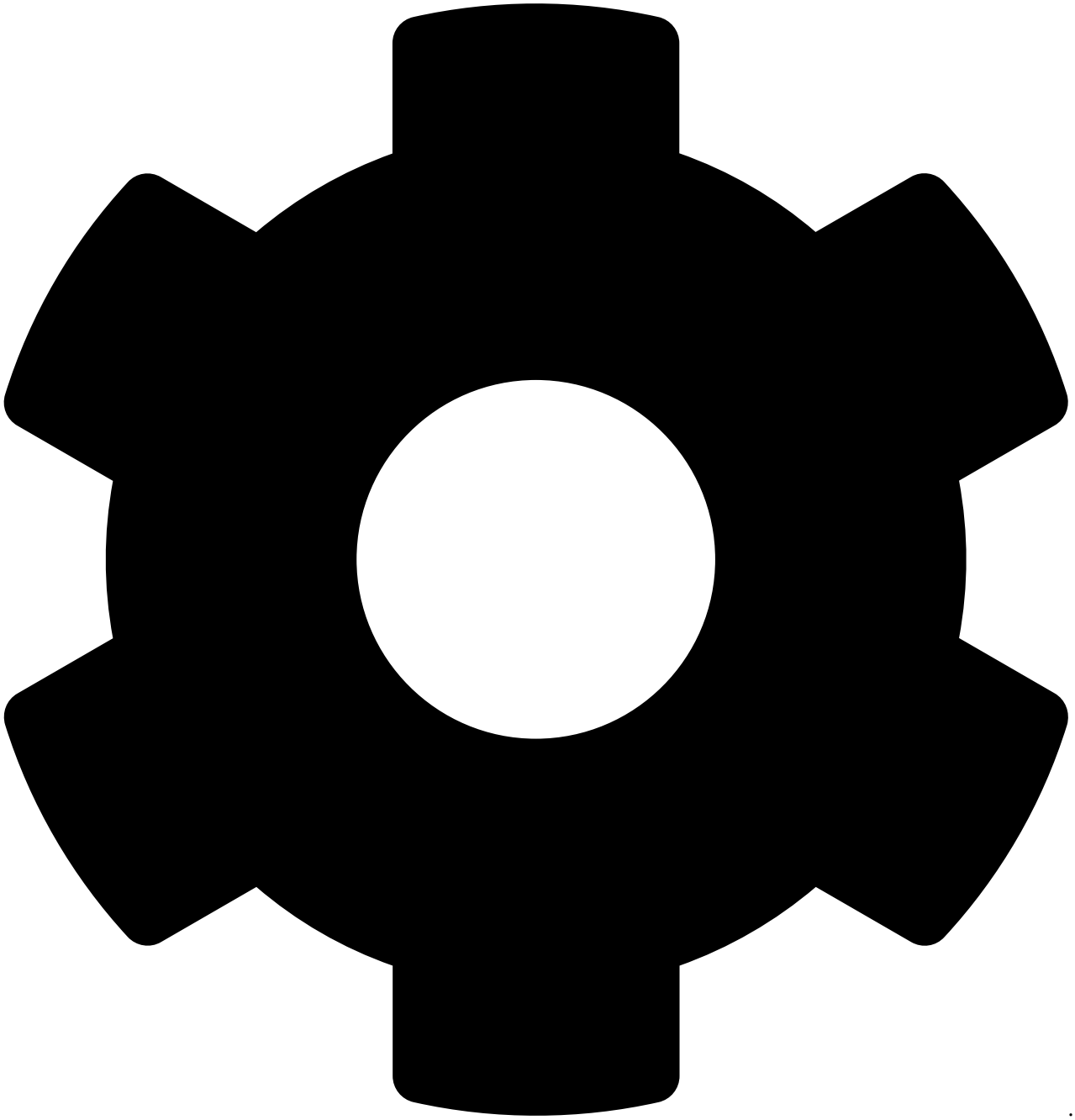

.

   The Settings modal window appears.
3. Click the Display Settings tab and check the Allow the user to add values to the filter option.

4. Click APPLY.
   The Enter values link appears.

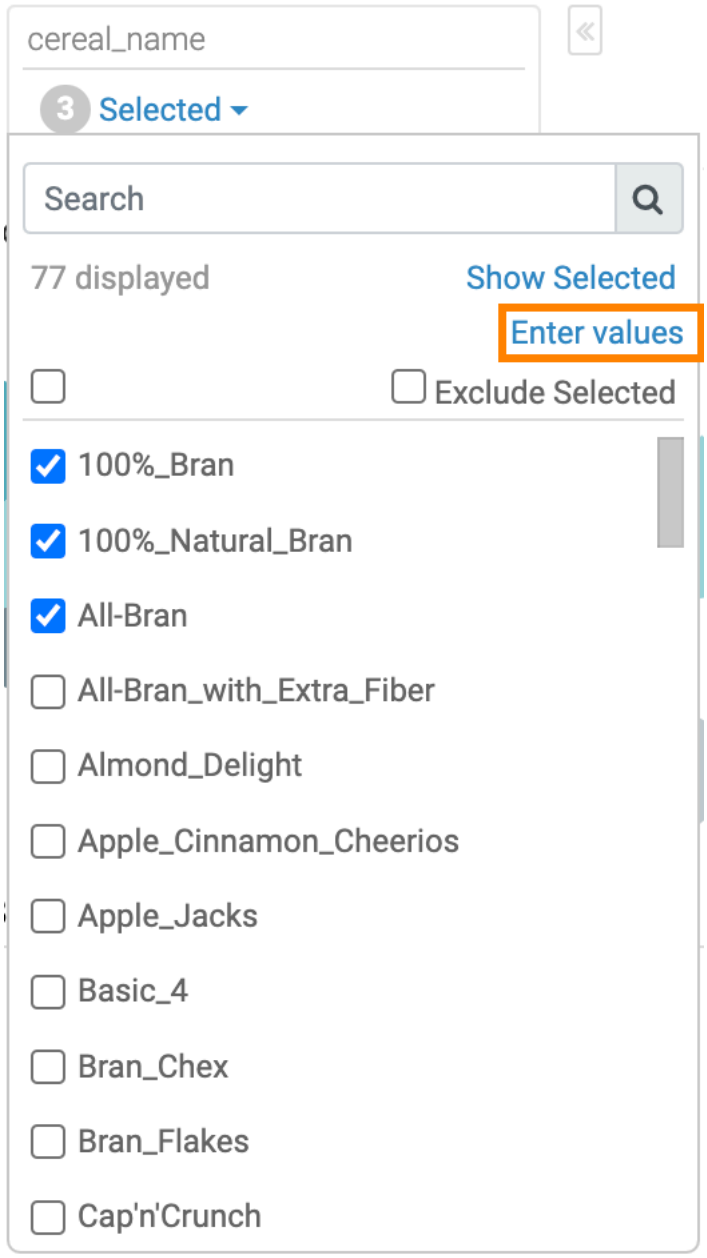

5. Click Enter Values.
   The Set Filter Values modal appears.

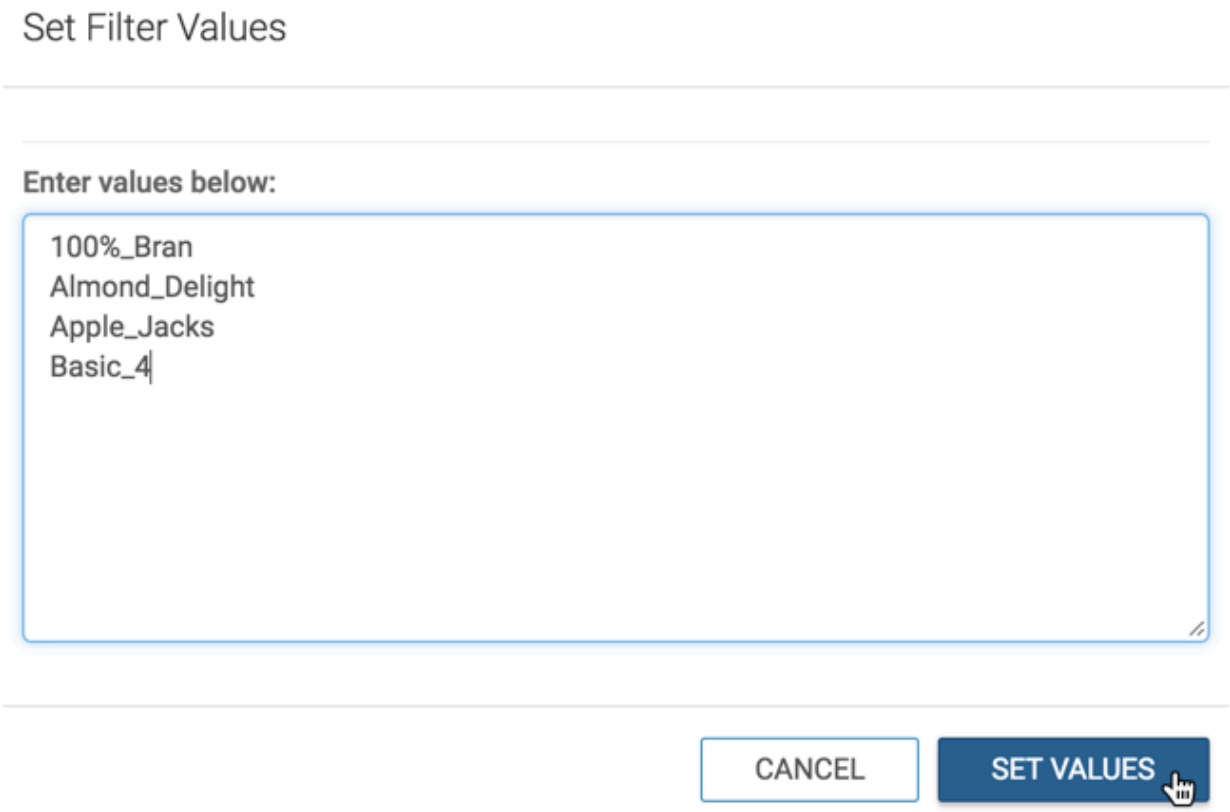

6. Enter values manually in the text box.
7. Click SET VALUES.

Results

All four values that were entered manually in the text box are selected.

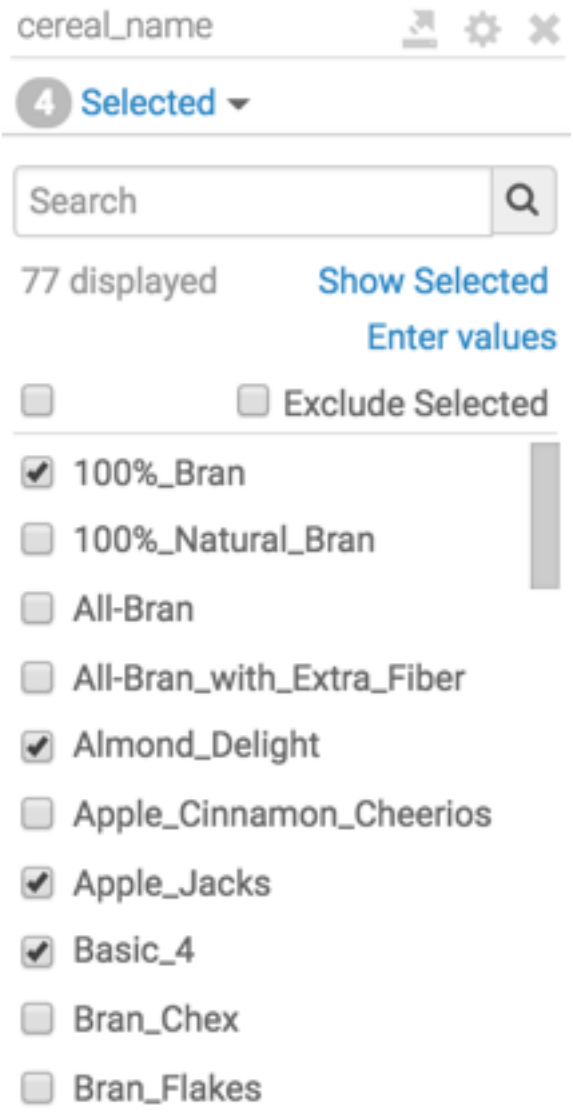

Using single value mode with radio buttons

The single value mode allows you to select a single value from the menu in a dashboard filter. This mode of dashboard controls uses radio button choices to let the user select a single value.

Before you begin

You have added a filter widget to your dashboard.

Procedure

1. Open the dashboard in Edit mode.

2. Hover over the filter and click

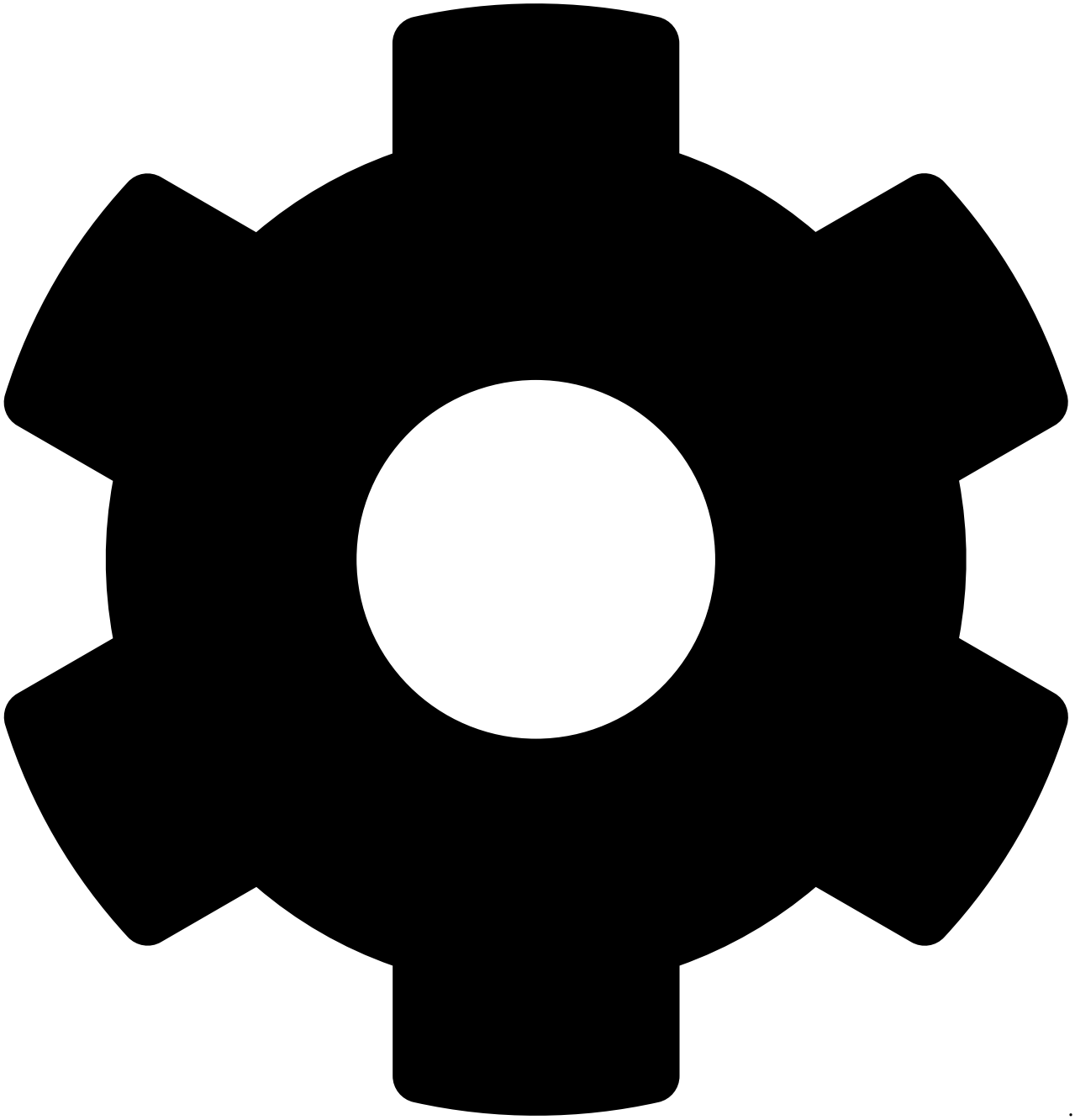

.

The Settings modal window appears.

3. Click Display Settings tab and check the Allow only one item to be selected at a time option.
4. Click APPLY.

Selecting all values

This option allows you to ensure that all data may be examined at once in a dashboard filter. For example, in single select mode on a cereal name filter, you can only look at details of one cereal. It may be interesting to look at information of all cereals at the same time.

Before you begin

You have added a filter widget to your dashboard.

Procedure

1. Open the dashboard in Edit mode.
2. Hover over the filter and click

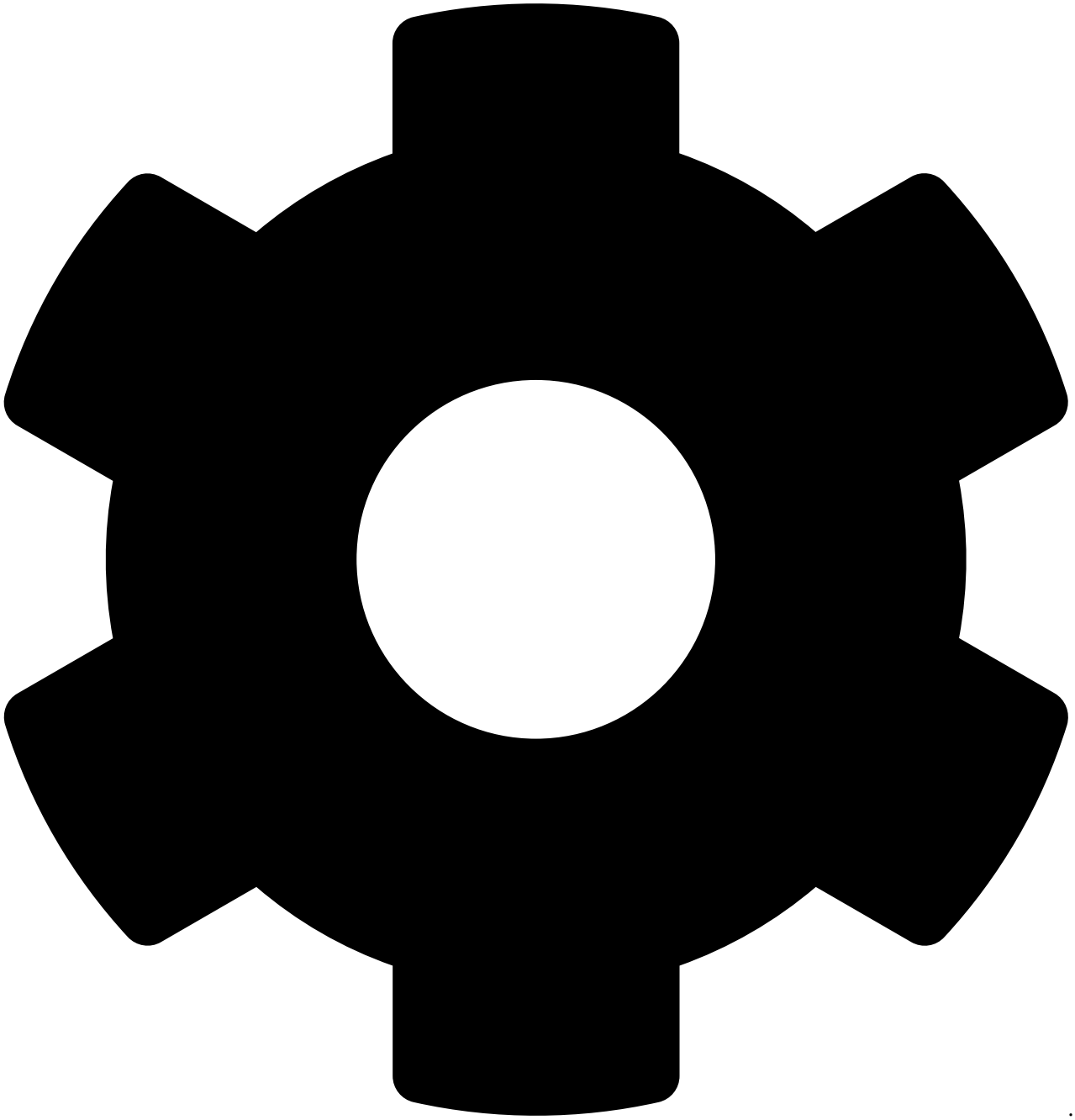

.

   The Settings modal window appears.
3. Click the Display Settings tab, then check the Allow only one item to be selected at a time option and the Include an option for '(All)' sub-option.
4. Click APPLY.

Results

The All option appears in the filter box.

Using single value with dropdown menu

You can use this option to select a single value from a dropdown menu in a dashboard filter.

Before you begin

You have added a filter widget to your dashboard.

Procedure

1. Open the dashboard in Edit mode.

2. Hover over the filter and click

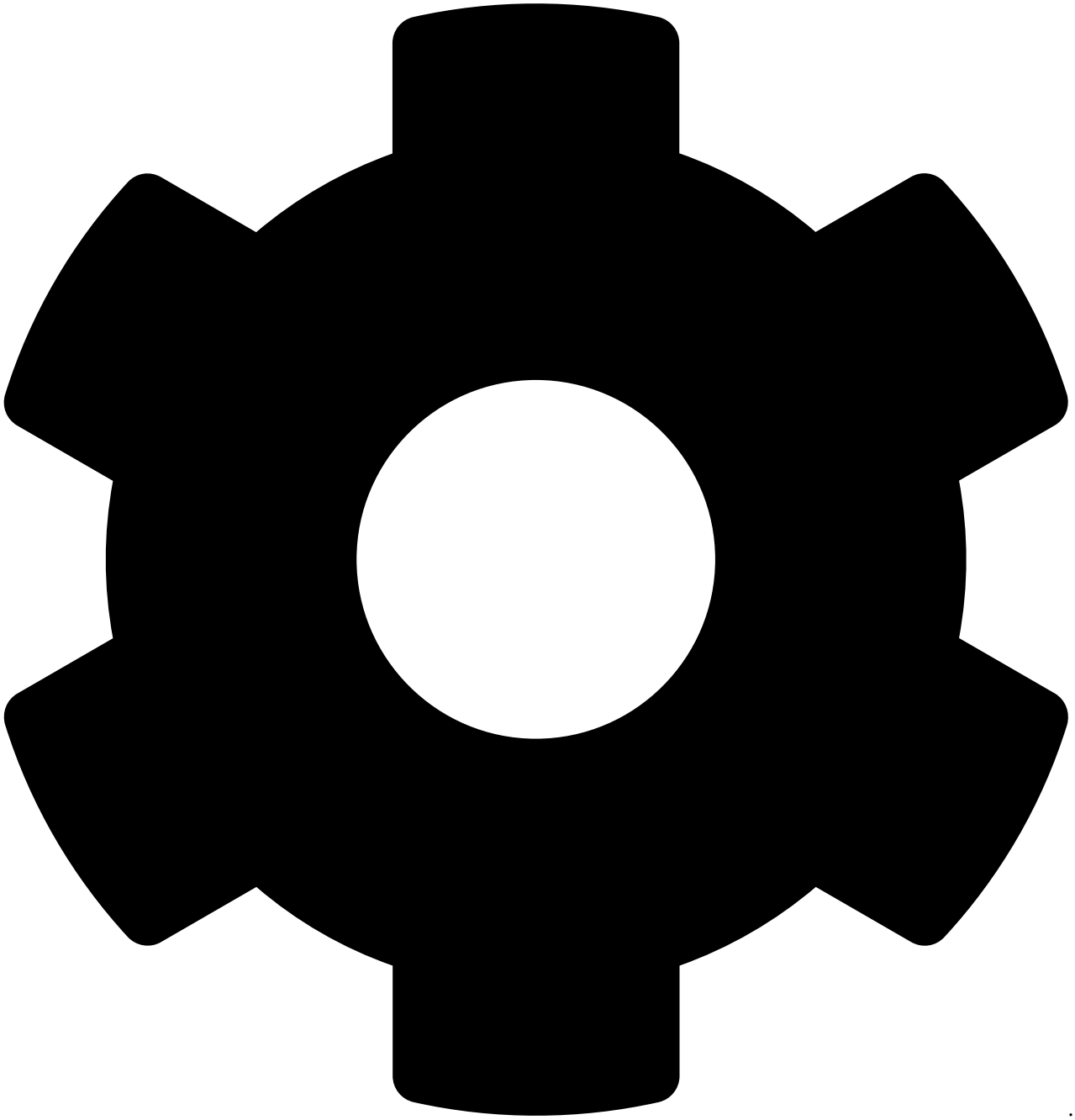

.

The Settings modal window appears.

3. Click the Display Settings tab, then check the Allow only one item to be selected at a time option and the Select values from a dropdown menu sub-option.
4. Click APPLY.

Displaying distinct parameters

You can use this option to ensure that parameters show a distinct 'key':'value' pair in a dashboard filter.

About this task

This option ensures that each selection is processed by the filter as a distinct 'key':'value' pair. By default, the selections are passed as a 'key':'list of values'. It enables a single filter to be potentially used for multiple purposes.

Before you begin

You have added a filter widget to your dashboard.

Procedure

1. Open the dashboard in Edit mode.
2. Hover over the filter and click

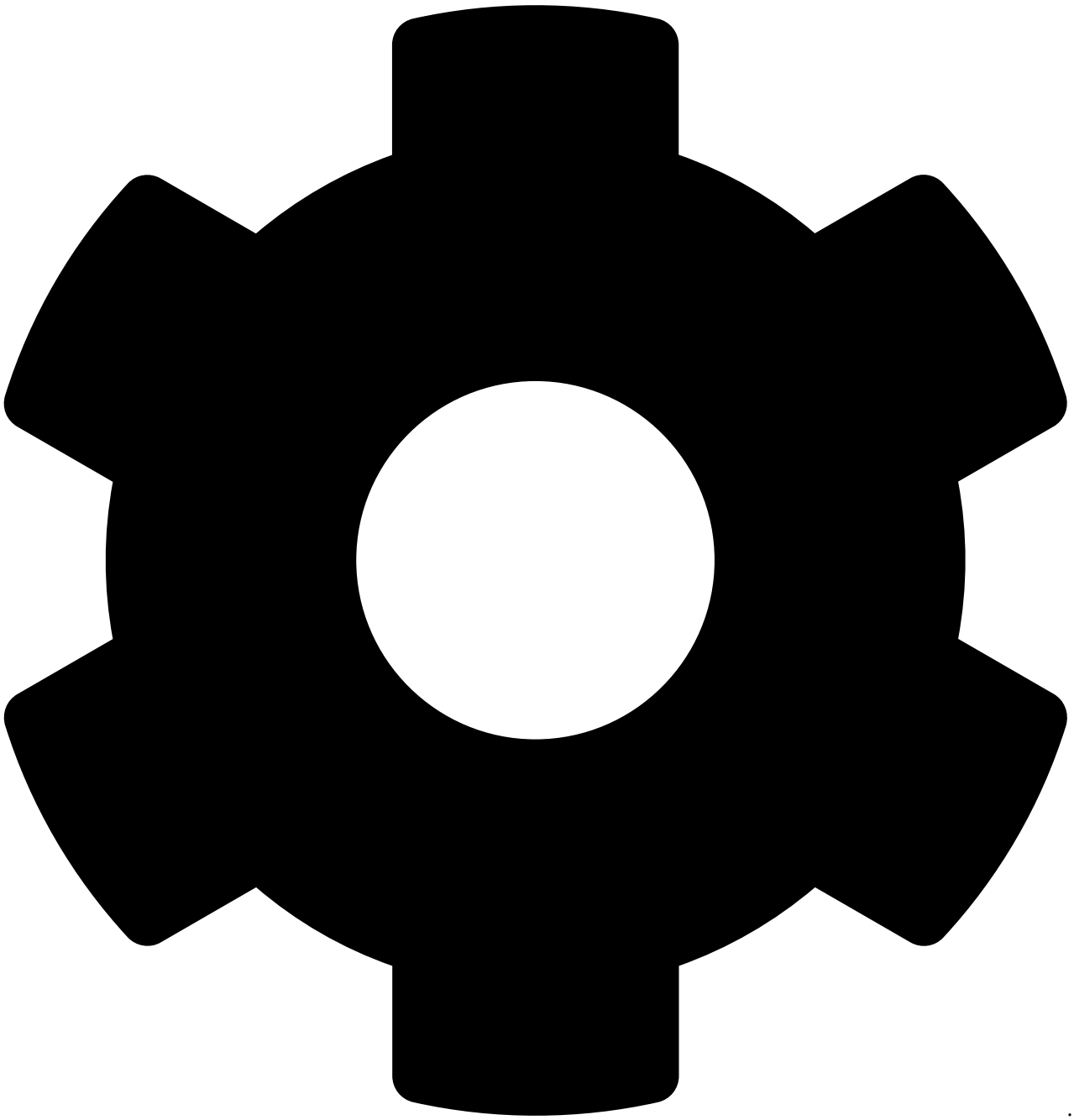

.

   The Settings modal window appears.
3. Click the Display Settings tab and check the Emit distinct parameters for each item selected option.

4. Click APPLY.

Hiding filters

This option allows you to hide the custom filter in a dashboard. If no viable selection data exists, this option hides the filter in View mode.

Before you begin

You have added a filter widget to your dashboard.

Procedure

1. Open the dashboard in Edit mode.

2. Hover over the filter and click

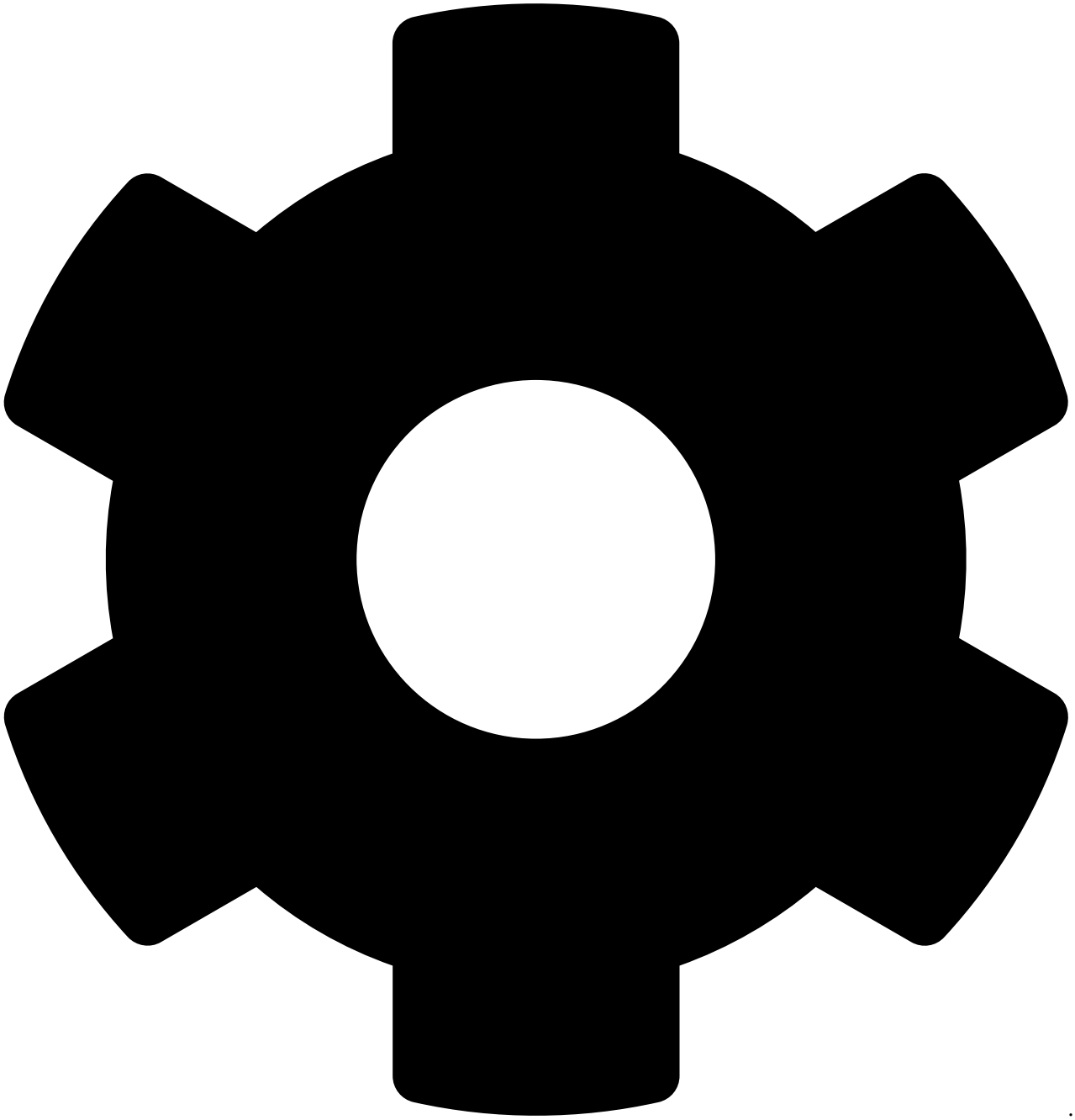

.

The Settings modal window appears.

3. Click the Display Settings tab and check the Hide filter if no input data option.
4. Click APPLY.

Remembering previous selections on dashboard filters when searching

This option allows you to view previous selections when searching for a new value in a dashboard filter.

About this task

This option is on by default.

The following examples show you how to enable or disable the option and how the filter behaves at run time both with and without the option.

Before you begin

You have added a filter widget to your dashboard.

Enabling 'Remember previous selections when searching'

1. Open the dashboard in Edit mode.
2. Hover over the filter and click

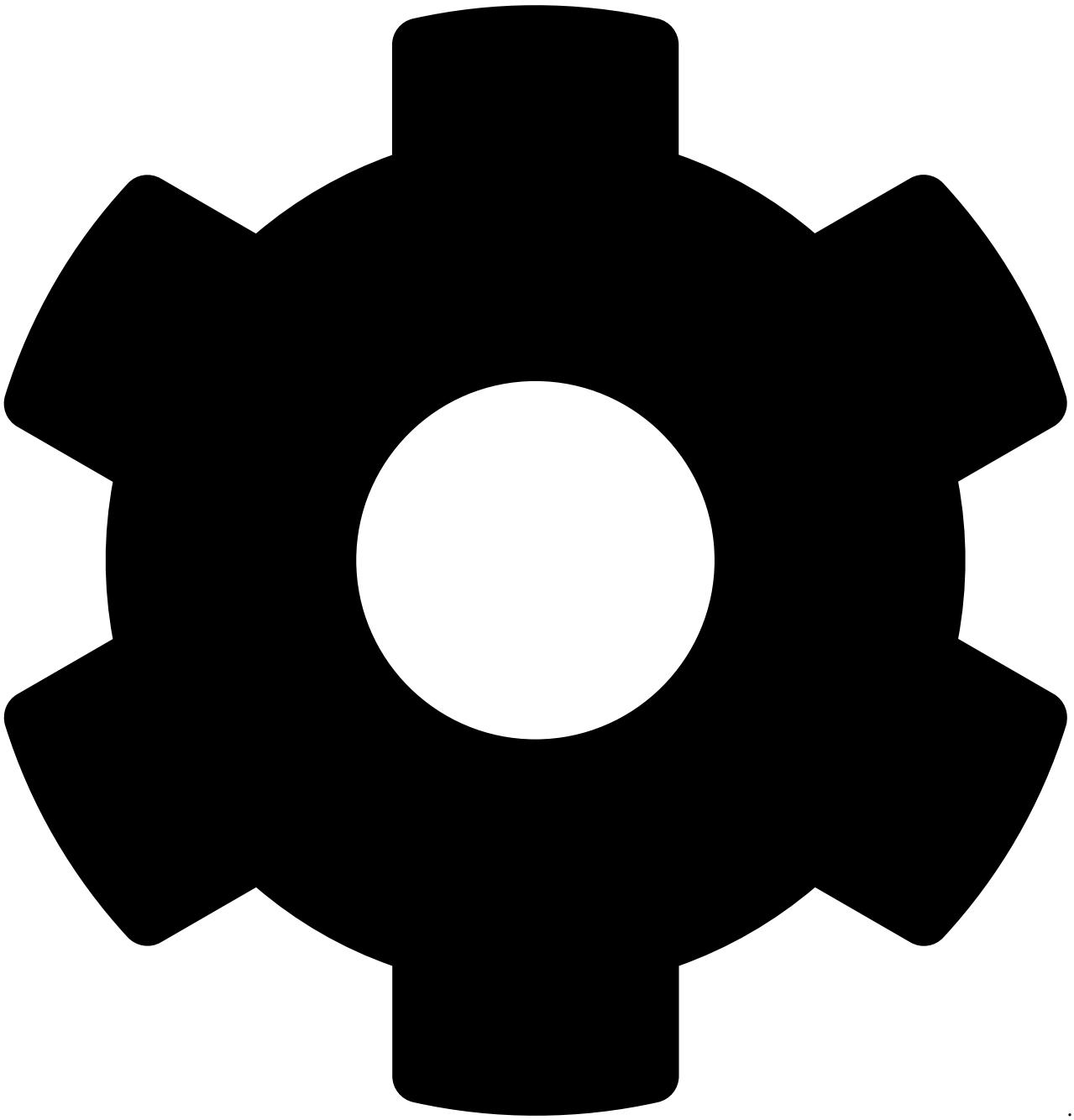

.

   The Settings modal window appears.
3. Click the Display Settings tab and select the Remember previous selections option. By default, it is turned on.
4. Click APPLY.
5. Save the changes to the dashboard.

Using Filters with 'Remember previous selections when searching'

Compare this with the next section, *Using Filters Without 'Remember Previous Selection'*. For this option, a dashboard based on Cereals dataset and a dashboard filter based on the dataset field cereal_name are used.

1. Select 'All-Bran' and 'Almond-Delight' from the dropdown menu.
2. Enter Basic_4 in the search box, and click the Search icon to find the value.
3. Click Basic_4 to confirm the selection.
4. Click the Show Selected option.

Using Filters Without 'Remember previous selections when searching'

When this option is disabled, you are unable to see the previous selections. In this example the same dashboard and filter on cereal_name are used as in the *Using Filters with 'Remember Previous Selection'* section, but with the option disabled. To use the filter without the option, follow these steps:

1. Select 'All-Bran' and 'Almond-Delight' from the dropdown menu.
2. Enter Basic_4 in the search box, and click the Search icon to find the value.
3. Click Basic_4 to confirm the selection.
4. Click Show Selected.

   Note that the top bar and the list at the bottom shows only the last selection, Basic_4.

Using multi-select mode

This option allows you to select multiple options in the dropdown, then apply all changes to the filter together by clicking outside the table. This collapses the list. For multi-select list, this is a recommended option as it minimizes the query refresh time. It is the default mode for application controls.

Before you begin

You have added a filter widget to your dashboard.

Procedure

1. Open the dashboard in Edit mode.

2. Hover over the filter and click

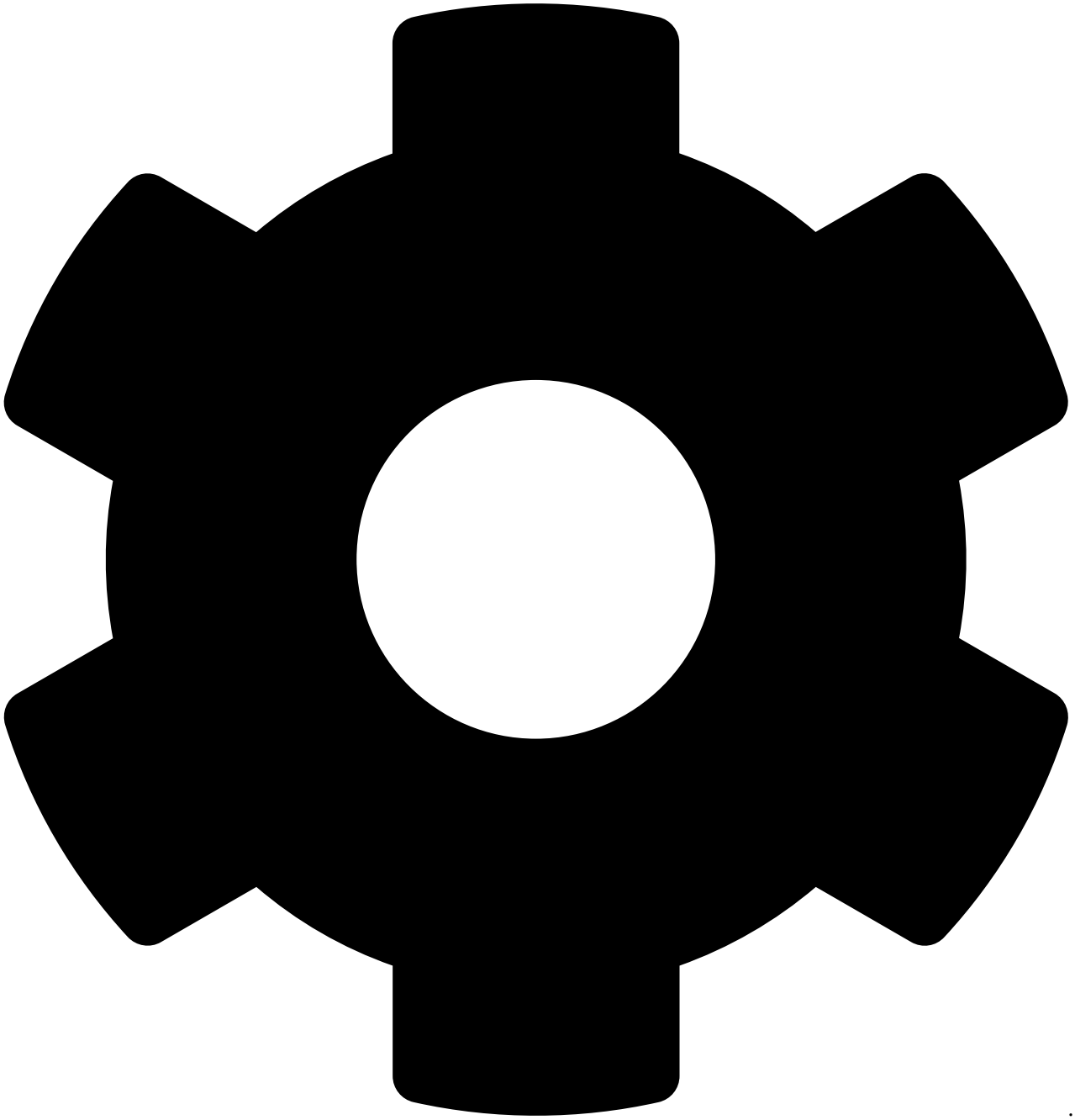

.

   The Settings modal window appears.
3. Click the Display Settings tab and select Apply all changes to a multi-select list at the same time.
4. Click APPLY.

5. If you want to edit the filter widget, click to display the available choices.

   The application control expands.

   - In the Search box, enter your search term and click .
   - To view only the selected values, click Show Selected located just below the search bar. This option toggles with Show All to switch between selected values and the entire list.
   - To include all selections, click the unlabeled check box in the top left corner.
   - To exclude specific selections, check the Exclude Selected box in the top-right corner.
6. In the filter, click Select.

   The application control expands in multi-select mode.

Configuring dashboard filter scope

In Cloudera Data Visualization, you must specify the scope of a newly created custom filter after adding it to a sheet in a dashboard.

Procedure

1. Open the dashboard in Edit mode.

2. Hover over the filter you want to update and click

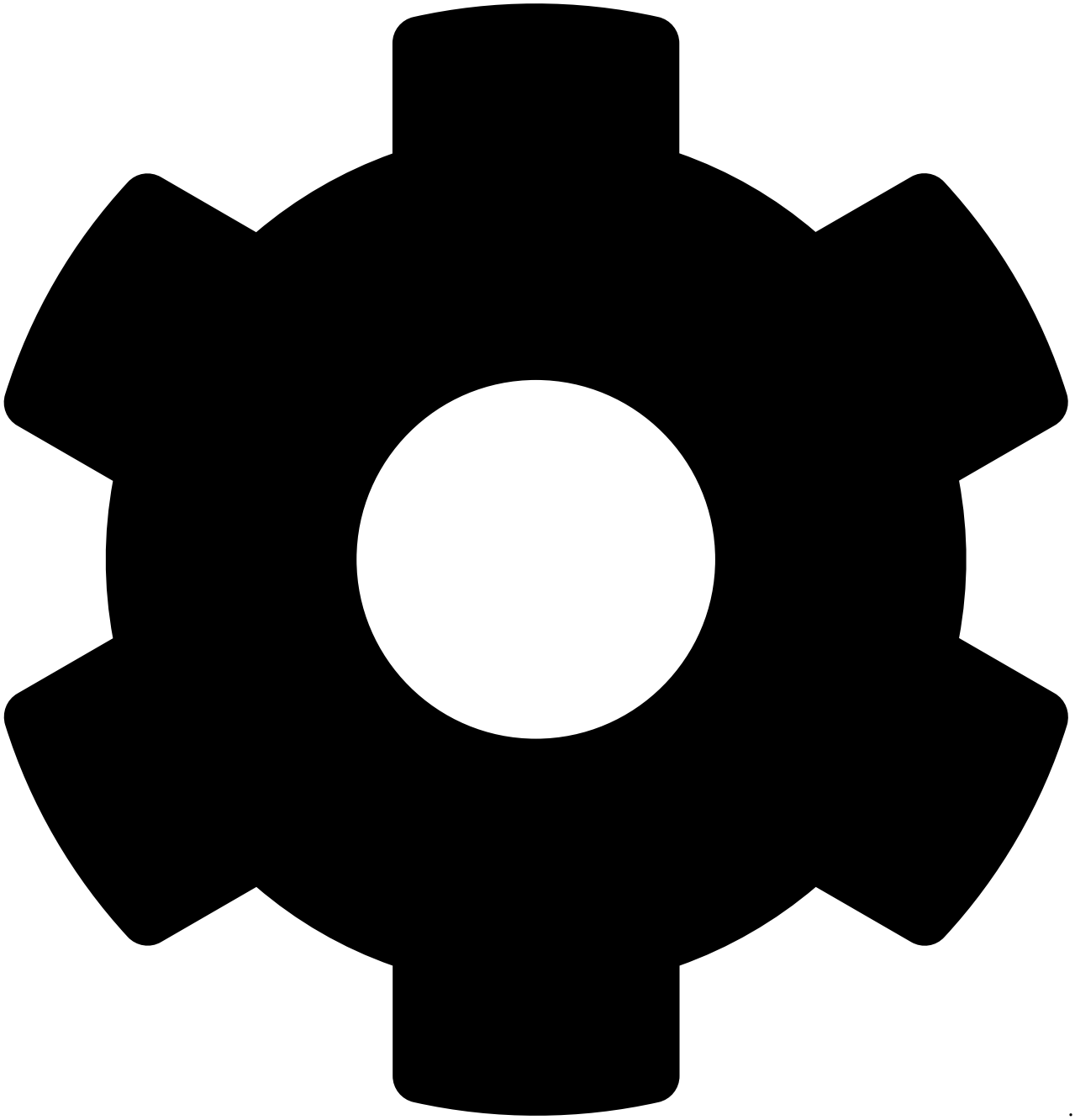

.

The Settings modal window appears.

3. Select the Scope tab.

4. Select the desired filter visibility.

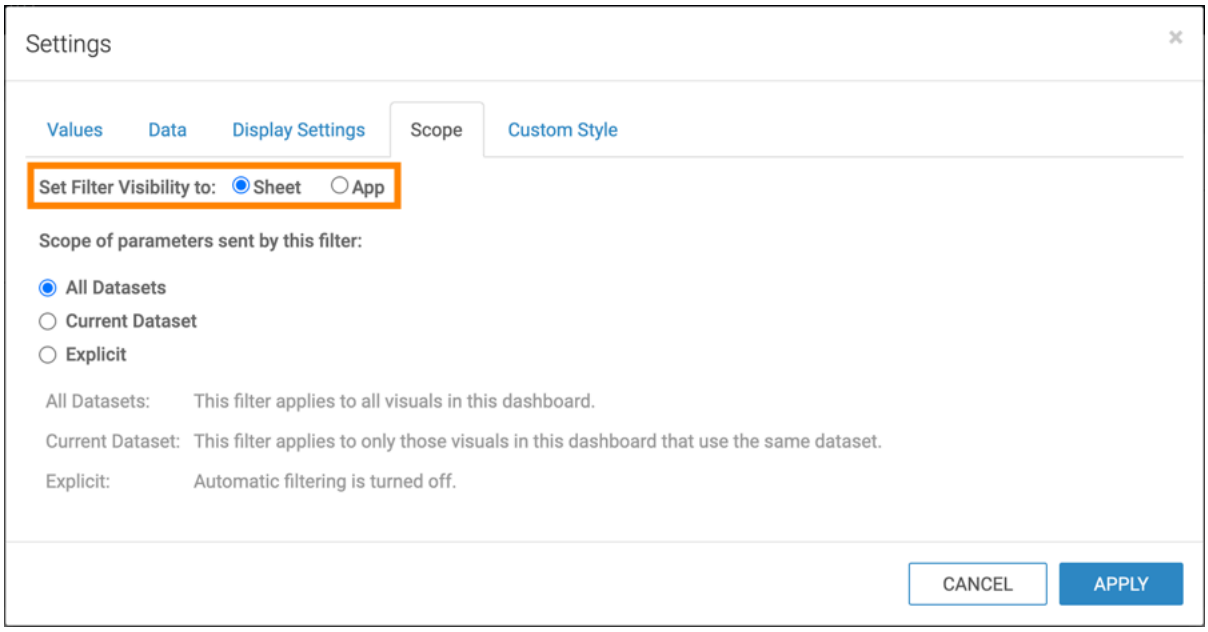

Note:

If App scope is selected all apps that contain the dashboard and all sheets in the dashboard also inherit the selections and behavior of this filter. When you select this option in Edit mode, the filter widget on the dashboard displays a gray triangle on the top left corner of the filter. Hover over the triangle to view the tooltip.

5. Select the desired scope of the filter.

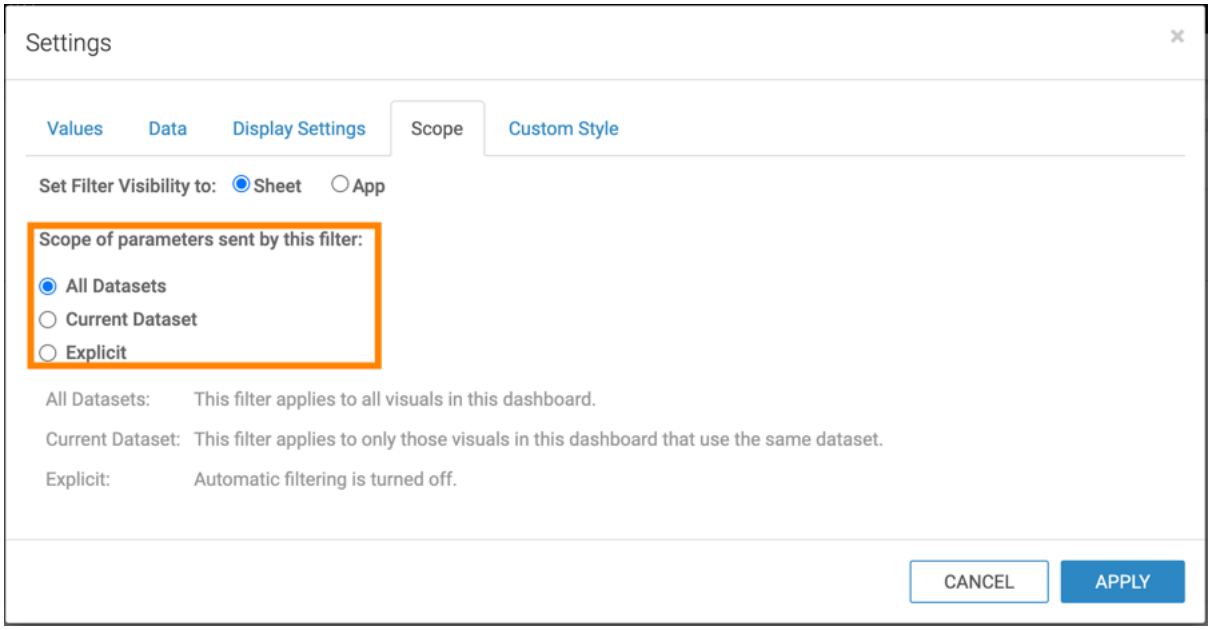

All Datasets

The filter applies to all visuals in this dashboard.

Current Datasets

This filter applies to only those visuals in this dashboard that use the same dataset.

Explicit

Automatic filtering is turned off.

6. Click APPLY.

Configuring custom style for a dashboard filter

Cloudera Data Visualization enables you to apply custom styles to a newly created filter in a dashboard.

About this task

When applying custom styles, there are several approaches. For information on setting styles for a visual or at the dashboard level, see Adding custom styles to visuals. For information on setting styles for a dashboard filter, see the instructions below.

Before you begin

By default, the Custom Styling option is disabled. Only users with administrative privileges can enable it.

Procedure

1. Hover over the filter you want to update and click

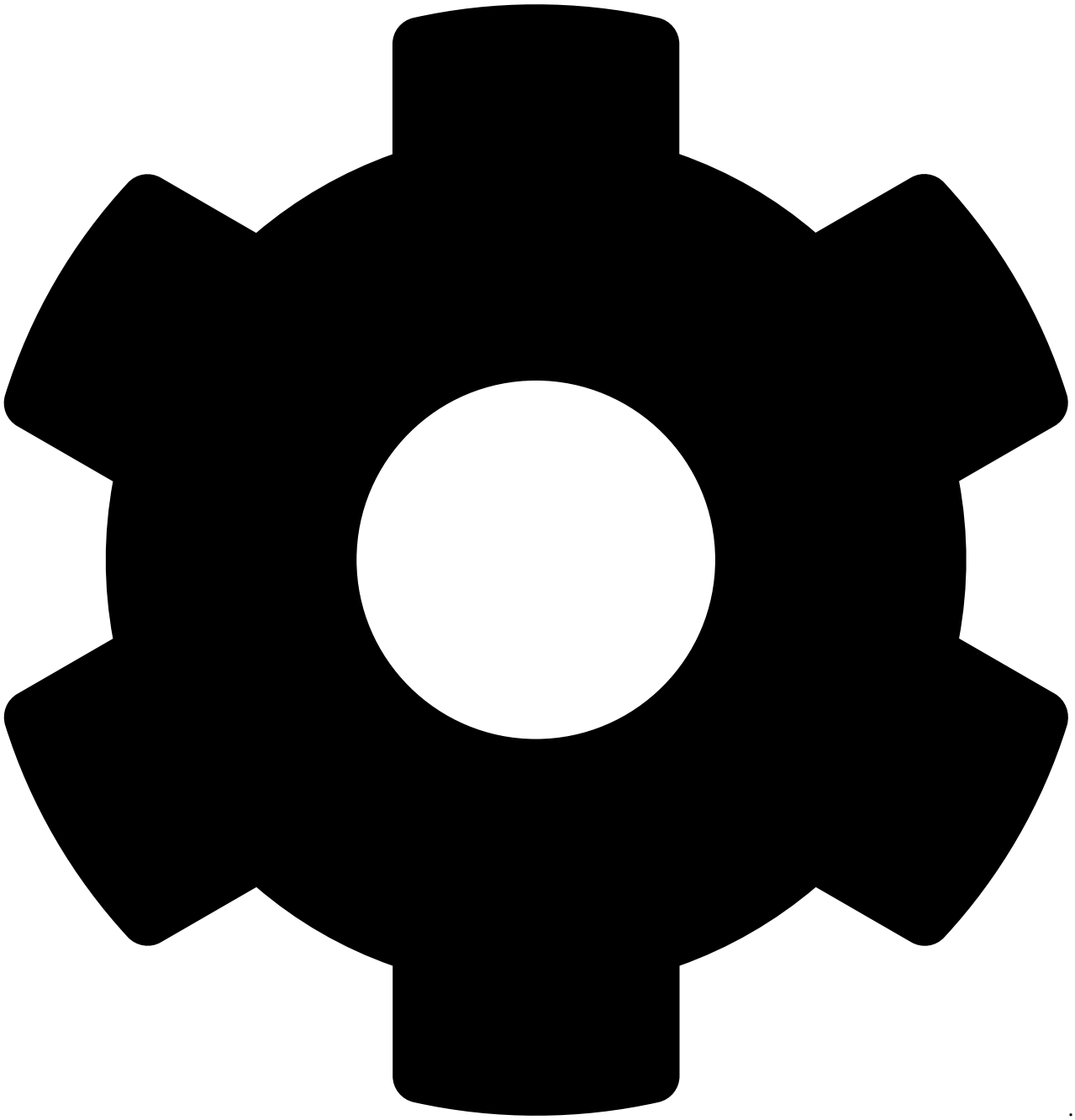

.

2. In the Settings modal window, navigate to the Custom Style tab.

3. Review and apply the available styling options as needed.

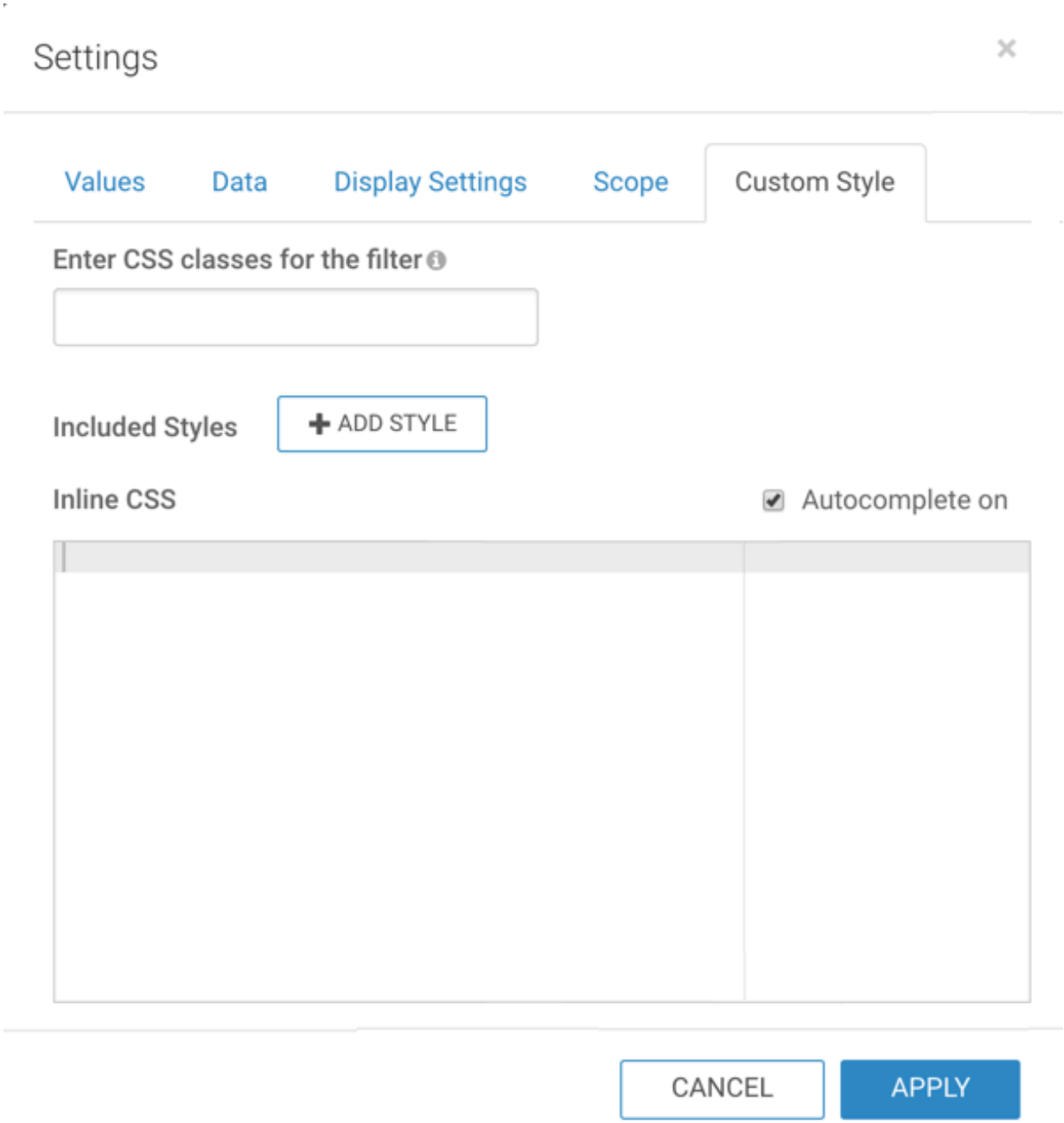

Related Information

Adding custom styles to visuals
Adding included styles to dashboards

Configuring custom style using CSS code

In Cloudera Data Visualization, you can customize the appearance of application filters by applying custom CSS. This allows you to enhance the visual styling of filters to match your dashboard design preferences.

Procedure

1. Open an existing or new application in Edit mode that has an application filter.

2. Hover over the filter and click

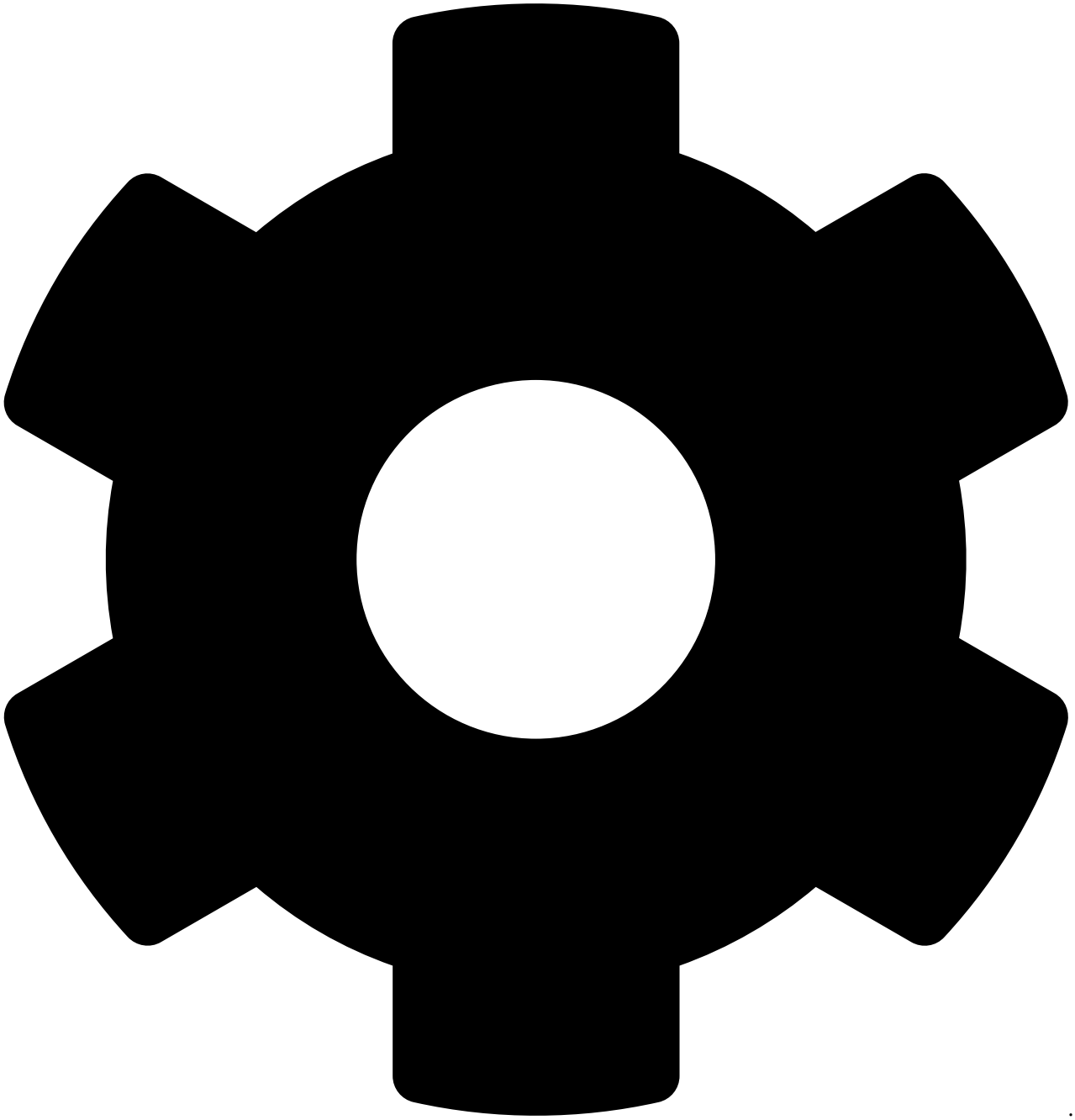

.

The Settings modal window appears.

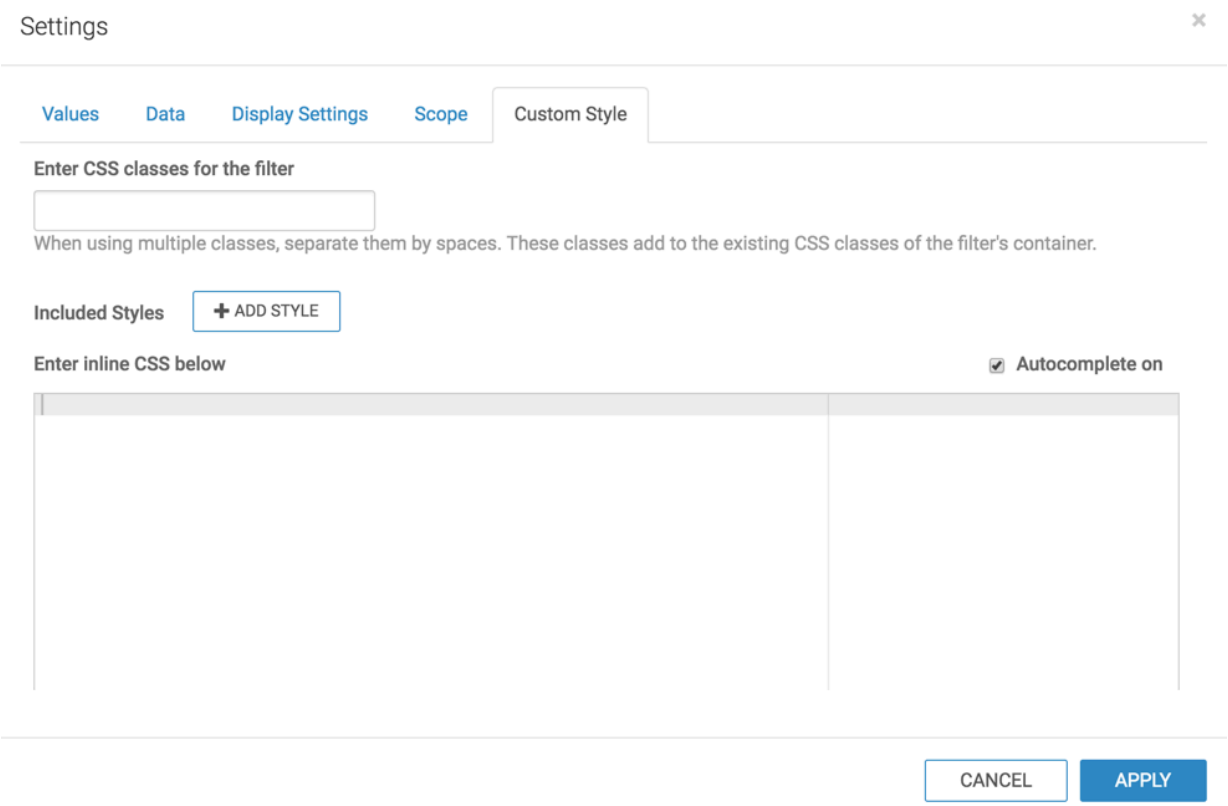

3. In the Settings modal window, navigate to the Custom Styling tab.
4. Set CSS class for visual to list-larger-font.
5. Enter the following code in the Enter inline CSS rules here to text box:

```
.list-larger-font .picklist-list li {
                        font-size:14px;
                        padding:3px;
                        border-bottom:dotted 1px #ddd;
                        }
```

6. Click APPLY.
7. Click SAVE to save the application.

Moving filter widgets on a dashboard

Cloudera Data Visualization displays dashboard filters across the top of the dashboard by default, but you can move them around as needed.

Moving dashboard filters from filter area into main grid area and back

1. Hover over the filter. When the configuration options appear, click the Move icon to move the filter to the grid area of the dashboard.

   The filter is now available among the other visuals of the dashboard. The appearance of the filter has changed significantly. It is represented as a simple white rectangle without search options.

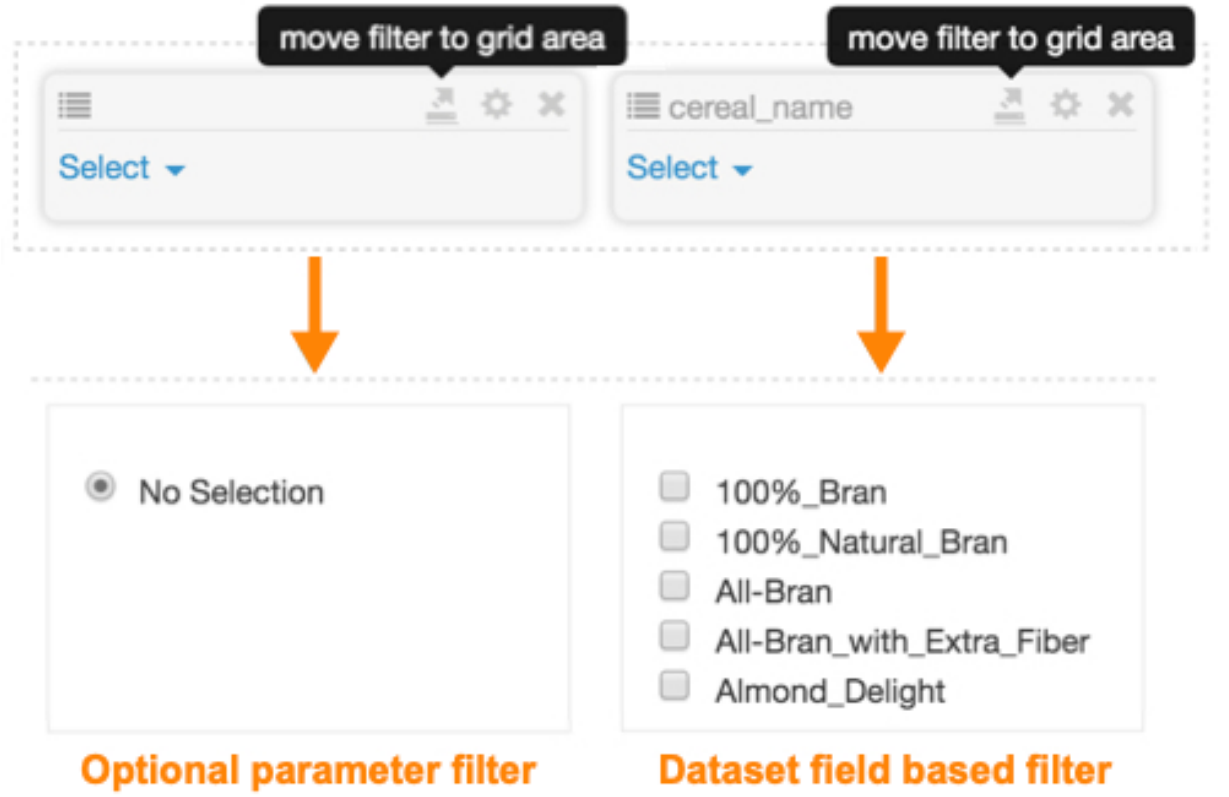

2. To return the filter to the filter area, hover over the filter. When the configuration options appear, click the Move icon to move the filter onto the filter area of the application.

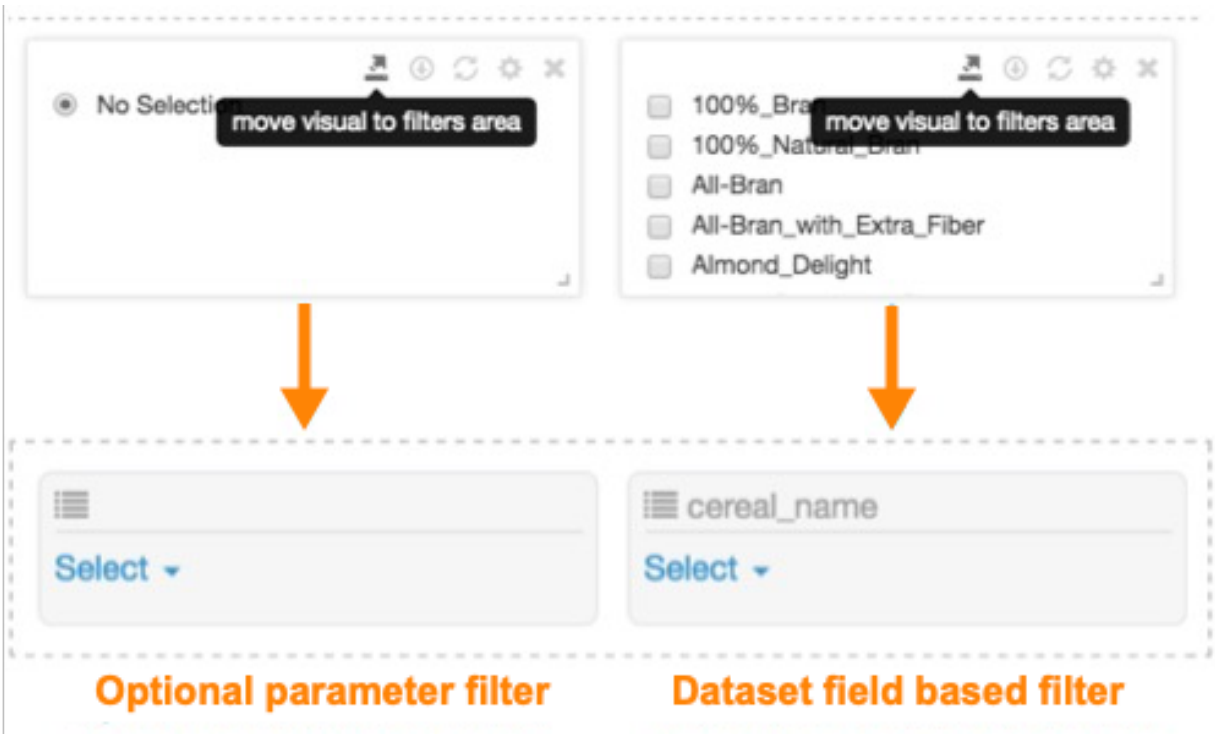

Moving dashboard filters from the top to the left of the dashboard

1. On the right side of the Dashboard Designer, click Style Filters .

2. Select Align filters on the left.

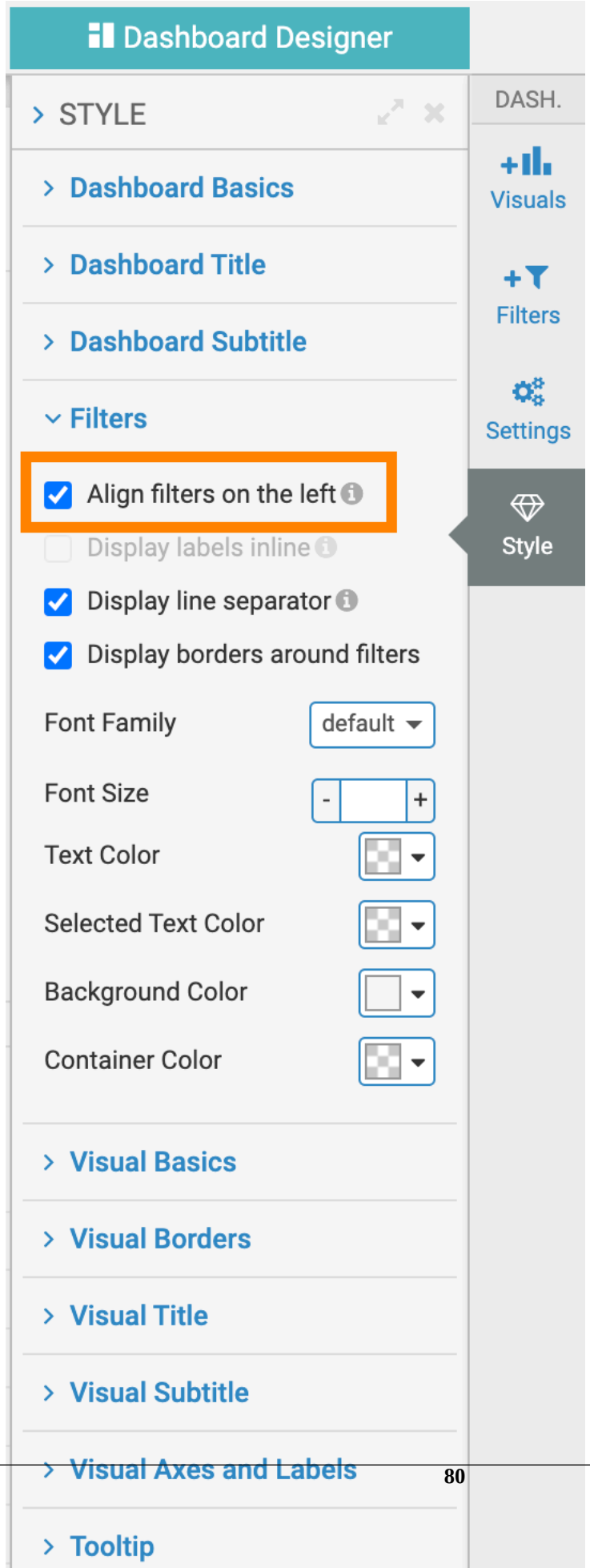
Dashboard Designer
STYLE
DASH.
Visuals
Filters
Settings
Style
Dashboard Basics
Dashboard Title
Dashboard Subtitle
Filters
Align filters on the left
Display labels inline
Display line separator
Display borders around filters
Font Family
default
Font Size
Text Color
Selected Text Color
Background Color
Container Color
Visual Basics
Visual Borders
Visual Title
Visual Subtitle
Visual Axes and Labels
Tooltip

To return the filters to the top, deselect this option.

Results

The visual on the left has filters across the top, and the visual on the right has filters on the left side.

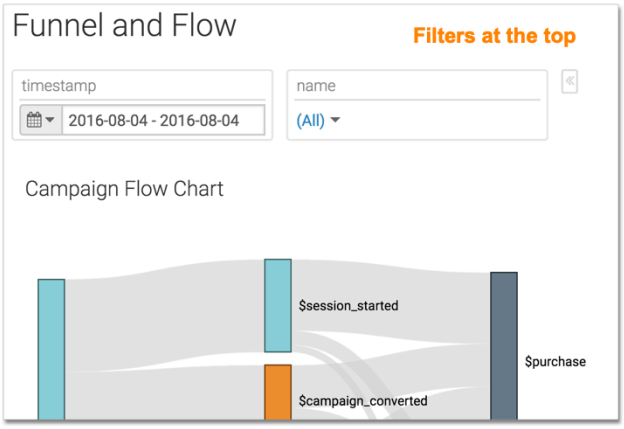

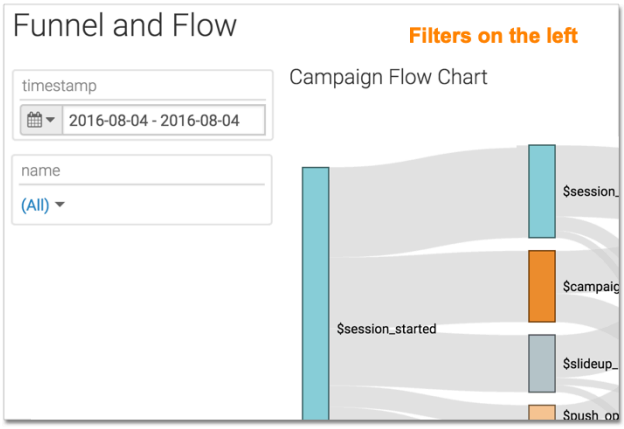

Cloudera Data Visualization 7.2.9

Working with Admin APIs

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enabling admin API support **4**

Admin API demo page **4**

Setting up a session **4**

Retrieving or changing logging level **5**

Enabling admin API support

By default, admin API URL support is disabled Cloudera Data Visualization. A platform administrator can enable this support, at the level of individual data types, through site-specific settings.

Procedure

1. Open Site Settings.
2. Scroll to Advanced Settings at the bottom of the left navigation.
3. To enable specific data types, add them to the assignment line.

   See the example below:

Option	
ADMIN_API_URL_LIST = ['visuals', 'datasets']	Add the data types to enable visuals and datasets.
ADMIN_API_URL_LIST = ['visuals', 'datasets', 'connections', 'users', 'groups','roles', 'segments', 'filterassociations']	To enable all data type support, list them all in the text box.
ADMIN_API_URL_LIST = ['*']	Use the wildcard to specify all options.

4. Click SAVE.

Admin API demo page

This demo page serves as a practical tool for understanding how to interact with the Cloudera Data Visualization admin APIs. It provides a comprehensive view of how network requests are formed for accessing the admin APIs.

Procedure

1. Open Site Settings.
2. Scroll down to the Advanced Settings located at the bottom of the left navigation pane.
3. Add the following configuration setting.

```
ADMIN_API_DEMO_LIST = ['visuals', 'datasets', 'connections', 'users', 'g
roups', 'roles', 'segments', 'workspaces', 'filterassociations']
```

4. Click SAVE.

Results

Once this configuration is enabled, the Cloudera Data Visualization page is available on *[***DOMAIN***]*/arc/apps/apidemo.

Setting up a session

Cloudera Data Visualization Admin API supports two alternatives for setting up a session: standard login access and APIKey access.

Standard login access

To establish a session with username/password login, use the following code:

```
username = "" #(user name as obtained from external source)
password = "" #(password as obtained from external source)
session = requests.session()
response = session.get(login_url)
session.headers['referer'] = response.url
payload = {'username':username,'password':password,'csrfmiddlewaretoken':
session.cookies['arccsrftoken']}
session.post(login_url, data = payload)
```

Note: The login URL has the form [http|https]://*[***HOST***]*:*[***PORT***]*/arc/apps/login.

APIKey access

To establish a session through the Cloudera API Key system and avoid the login process, use the following code:

```
apikey = (apikey string)
session = requests.session()
session.headers['AUTHORIZATION'] = 'apikey %s' % apikey
```

In this approach, the client code must obtain the Cloudera Data Visualization access key through a client-controlled mechanism, then add the APIKey to the request header and avoid explicit login.

Fetching data from all datasets in the system

After setting up a session, you can fetch the entire data from all datasets in the system by using the following code:

```
response = session.get(api_url + 'datasets?detail=1')
datasets = response.json()
```

Note: The API URL has the form [http|htttps]://*[***HOST***]*:*[***PORT***]*/arc/adminap i/*[***VERSION***]*. Use the URL option 'detail=true' to fetch all information in the GET call.

Retrieving or changing logging level

Cloudera Data Visualization Admin API provides a convenient way to manage logging levels for specific loggers, allowing you to control the verbosity of log messages as needed. It supports retrieving and changing the logging level for a specified logger within the system.

About this task

You can query the current logging level for a logger or modify it using the following API endpoints.

- GET: /arc/adminapi/loglevel/<logger-name>
- POST: /arc/adminapi/loglevel/<logger-name>

POST parameters

The POST endpoint expects a JSON document in the request body. The JSON document should contain a single field, level, specifying the desired log level. It accepts one of the predefined names in the Python logging module:

- CRITICAL
- DEBUG
- ERROR
- FATAL
- INFO

- WARN
- WARNING

Authentication

Both API key and session-based authentication are accepted. The retrieval of the logging level does not require any special permissions. However, updating the logging level requires the sys_viewlogs permission.

Procedure

1. Use the curl command to send a GET request to the specified endpoint to retrieve the current logging level for a specific logger.

```
curl -H 'content-type: application/json' -H 'authorization:
apikey [***API-KEY***]' [***VIZ_URL***]/arc/adminapi/loglev
el/[***LOGGER_NAME***]
```

   Make sure to replace *[***API-KEY***]* with your actual API key, *[***VIZ_URL***]* with the Cloudera Data Visualization URL, and *[***LOGGER_NAME***]* with the name of the specific logger you want to check.

   After executing the command, you receive a JSON response indicating the logger name and its current logging level.
2. Review the 'level' field in the response to determine the current logging level for the specified logger.
3. Use the curl command to send a POST request to the specified endpoint to change the logging level for a specific logger.

   Provide the desired log level in the request body.

```
curl -H 'content-type: application/json' -H 'authorization:
apikey [***API-KEY***]' [***VIZ_URL***]/arc/adminapi/loglev
el/[***LOGGER_NAME***]
 -d '{"level":"DEBUG"}'
```

   After executing the command, you receive a JSON response confirming the changes, including the logger name and the new logging level.
4. Verify that the 'level' field in the response now reflects the updated logging level for the specified logger.

Example

Getting the current loglevel for the arcweb logger:

1. Request

```
curl -H 'content-type:application/json' -H 'authorization: apikey <API-K
EY>'
          <VIZ_URL>/arc/adminapi/loglevel/arcweb
```

2. Response

```
{
    "logger": "arcweb",
    "level": "INFO"
}
```

Setting the log level to DEBUG for arcweb:

1. Request

```
curl -H 'content-type:application/json' -H 'authorization: apikey <API-K
EY>' <VIZ_URL>/arc/adminapi/loglevel/arcweb -d '{"level":"DEBUG"}'
```

2. Response

```
{
  "logger": "arcweb",
  "level": "DEBUG"
}
```

Cloudera Data Visualization 7.2.9

API Keys

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enabling API keys **4**

Creating new API keys **4**

Changing API keys **5**

Deleting API keys **6**

Enabling API keys

About this task

To enable the Data API keys, follow these steps:

Procedure

1. Edit the settings.py file (typically, settings_local_dev.py) file to include the following setting: DATA_API_ENABLED = True
2. Save the change.
3. Restart Cloudera Data Visualization.
4. Reload Cloudera Data Visualization in the browser window.

Related Information

Data API

Creating new API keys

You can create API keys in Cloudera Data Visualization.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.

2. Select Manage API Keys.

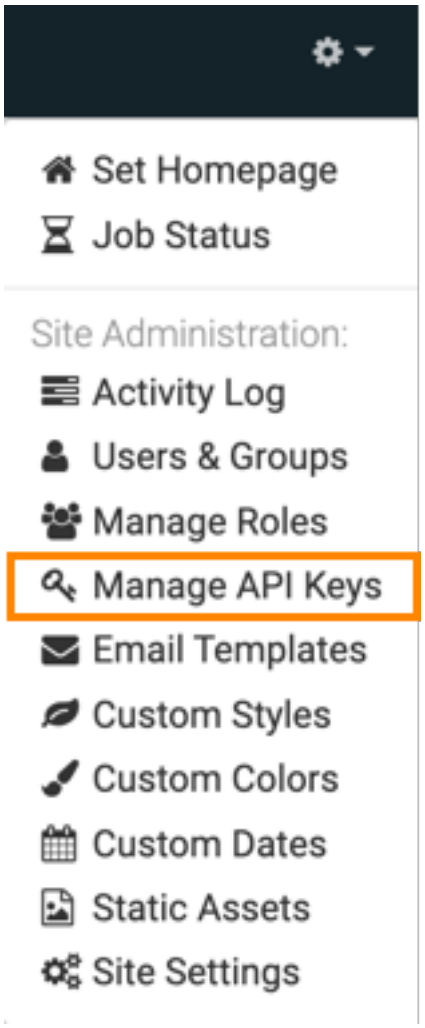

3. In the Manage API Keys interface, click New API Key.

   New API Key window modal appears.
4. Make the following selections:
   - Select the Username from the menu.
   - Under Expiration Date, either select the date from the calendar drop-down (by default, one year from current date), or check the Never Expire option.
   - Under Status, select Active.
5. Click Save
6. Review API Key window modal appears. There are two keys: API Key, and Secret API Key. This is the only time that the second (secret) key appears. Copy both of them from the interface, and then click OK to dismiss the modal.

Changing API keys

Cloudera Data Visualization enables you to edit some of the properties of existing API.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.

2. Select Manage API Keys.

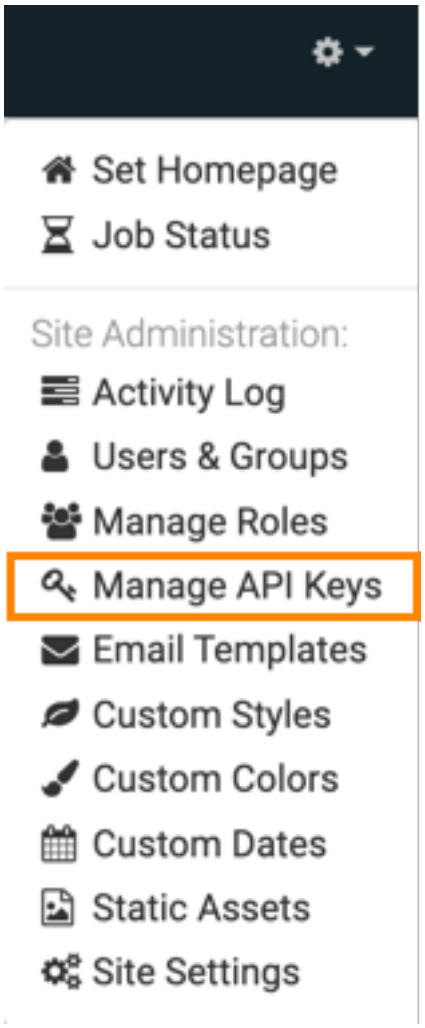

3. In the Manage API Keys interface, click the Pencil icon of the key you want to edit.

   Edit API Key window modal appears.
4. You can make the following changes and click Save:
   - Select a different user in the Username menu.
   - Under Expiration Date, either select a different date in the calendar drop-down, or check the Never Expire option.
   - To deactivate the key, select Inactive option under Status.

   Review API Key window modal appears.
5. Click OK.

Deleting API keys

About this task

If you want to delete an API Key authorization, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Manage API Keys.

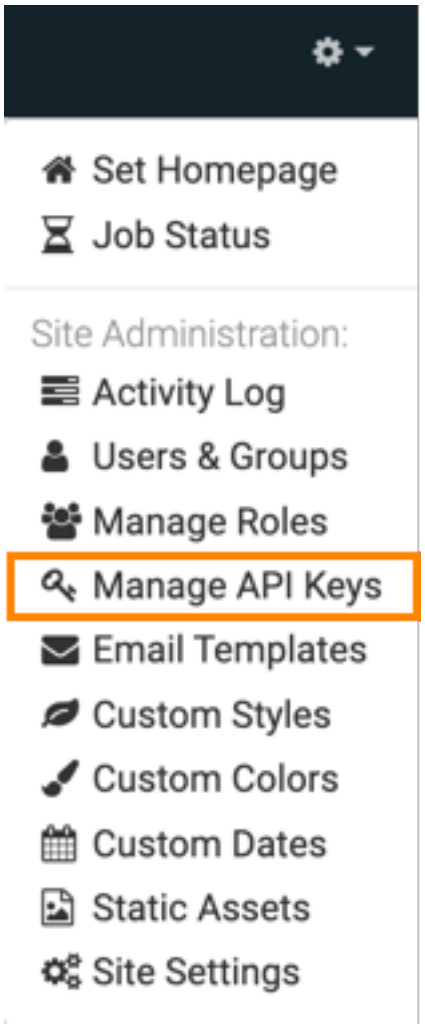

3. In the Manage API Keys interface, click the Trashcan icon of the key you want to delete.

   The delete confirmation modal appears.
4. Click DELETE.

Cloudera Data Visualization 7.2.9

Applications

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a new application 4

Adding tabs to applications 6

Removing a tab from an application 7

Adding dashboards to tabs 8

Removing a dashboard from a tab 10

Changing datasets in applications 11

Rearranging dashboards 13

Rearranging tabs 15

Launching applications 15

Moving apps between workspaces 17

Cloning an application 19

Exporting an app 20

Importing an application 23

Deleting an app 26

Creating a new application

Learn how to create an application in Cloudera Data Visualization.

Procedure

1. Click NEW APP in the top right corner of the main HOME interface.

   The App Designer interface opens on the App Navigation tab and it has a default menu tab.

   Alternatively, you can click Create New App in the top right corner of the preview area in the VISUALS interface.

2. In the Dashboards menu of the App Designer, select a dataset.

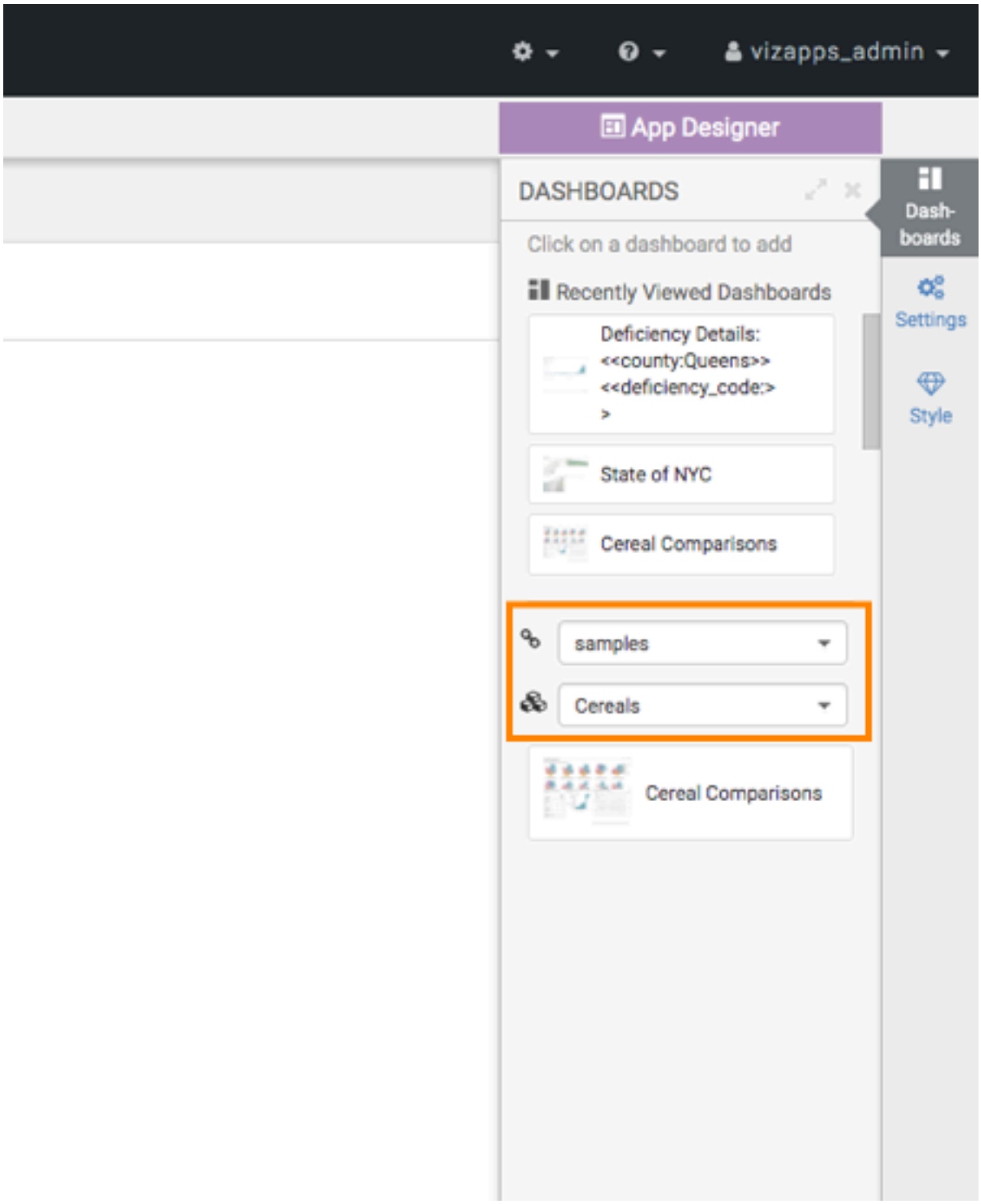

3. Enter a name for your app in the App Name text box.

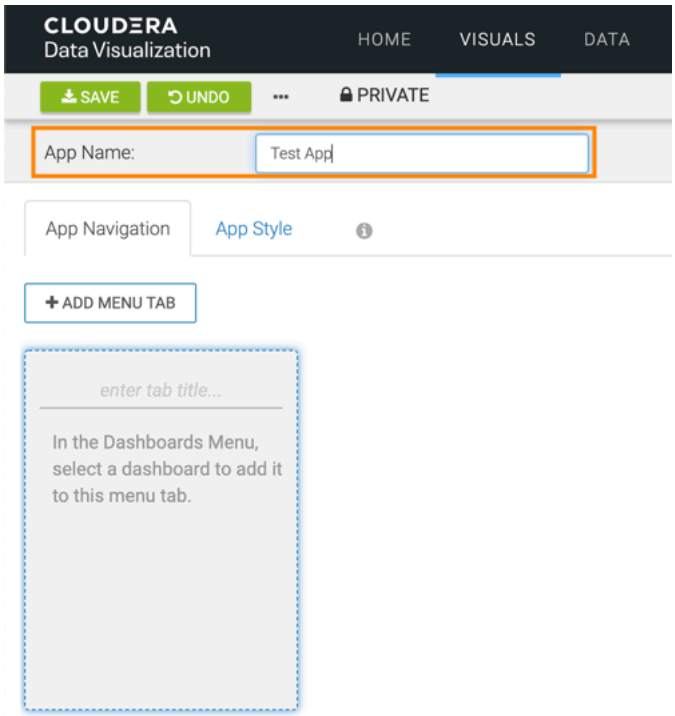

4. In the highlighted empty tab, enter the tab title.
5. In the Dashboards menu, select a dashboard to add it to this tab.
6. Switch to the App Style tab.
7. Design the appearance of your new app.
8. Click SAVE.

Results

Your new application is saved. To the right of the App Name, you can see the ID of the app, its update time information, and the last user who modified it.

Adding tabs to applications

In a Cloudera Data Visualization application, you can have several named tabs that contain one or more dashboards.

Procedure

1. In your application, click Add Menu Tab.

2. Name the new tab.

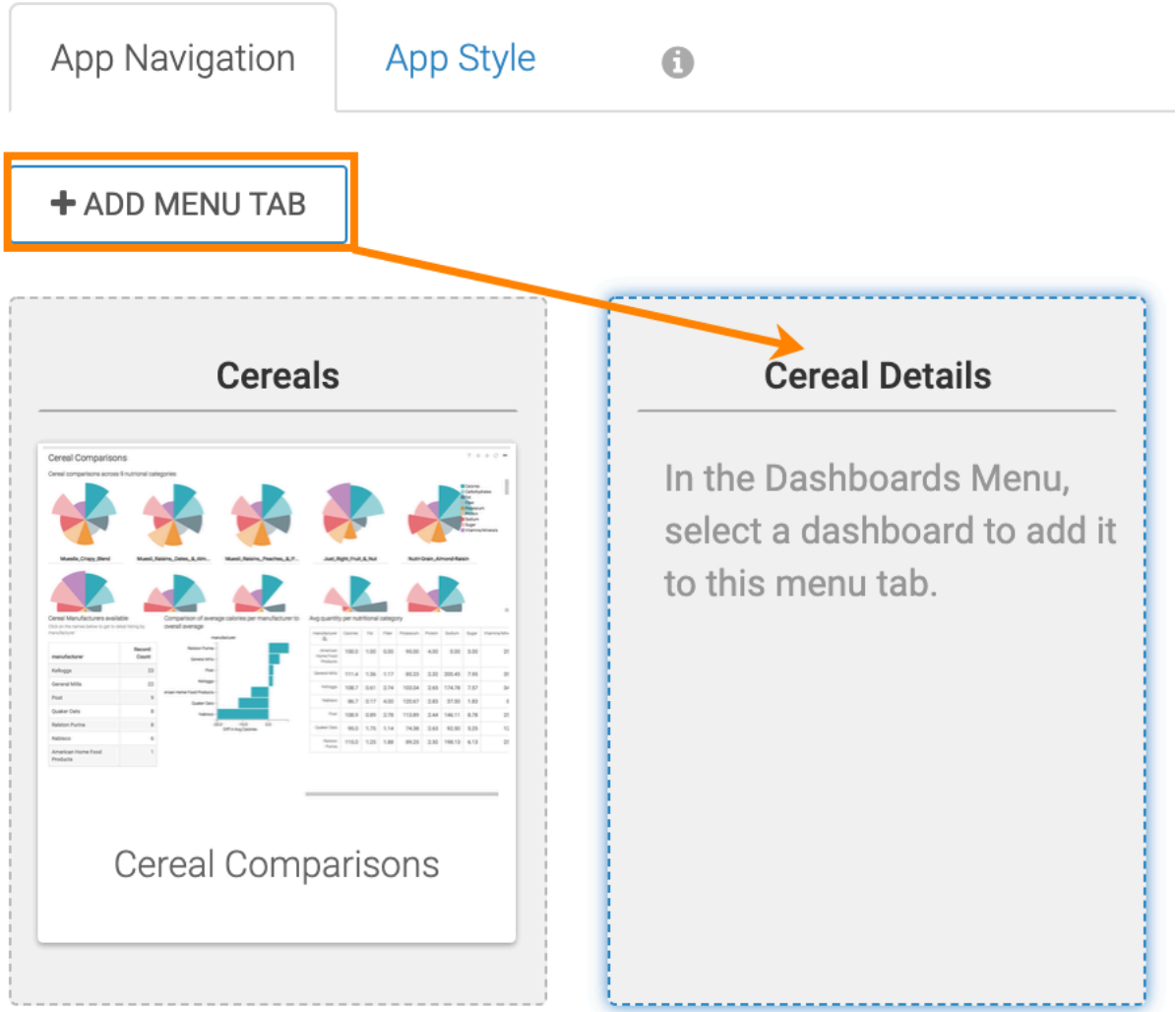

Note: Remember that you can use dashboards from any connection or dataset in the same app, provided that the creater/owner has the appropriate privileges.

Removing a tab from an application

In a Cloudera Data Visualization application, you can easily delete tabs and the dashboards they contain.

Procedure

1. In the App Navigation interface, hover over the top of a tab, and click the (remove) icon.

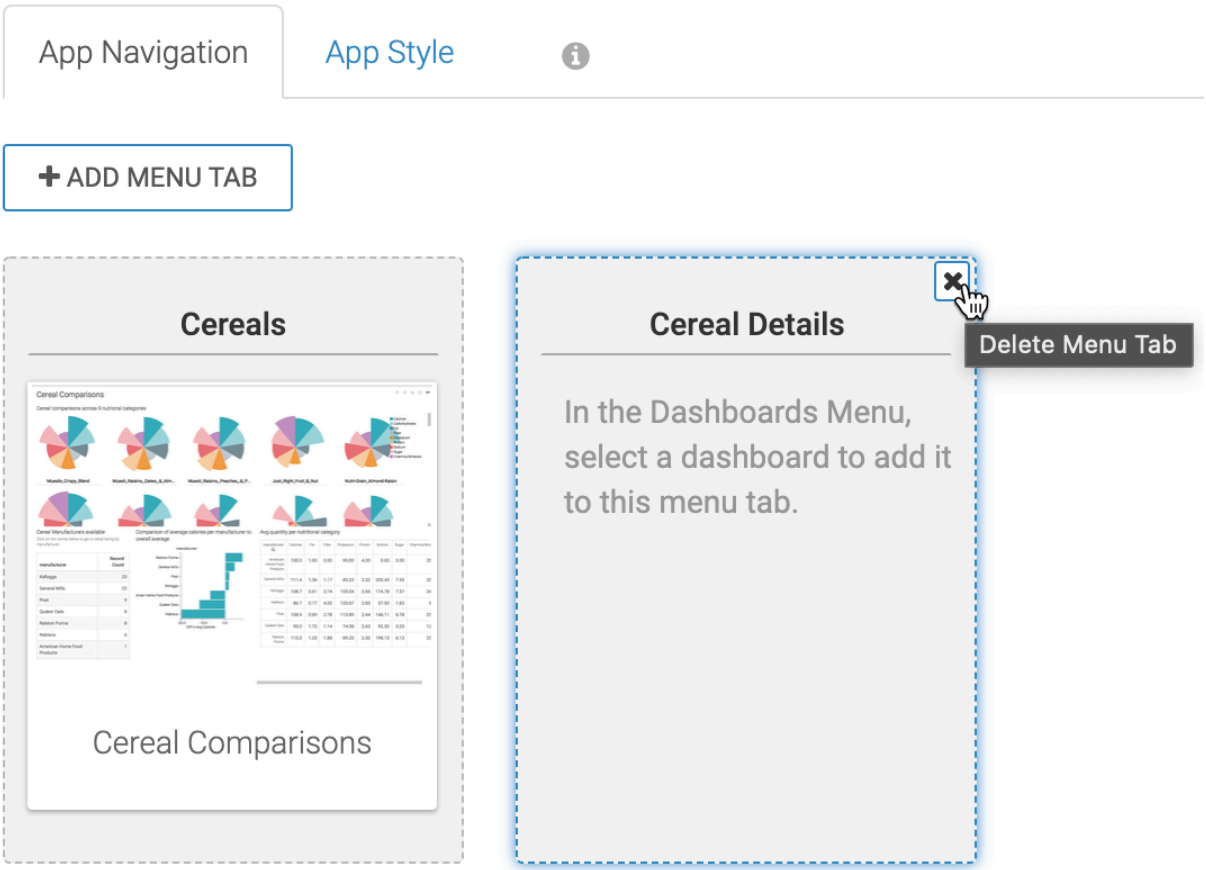

   The App Navigation interface no longer shows the deleted tab.

2. To ensure that the change persists, click Save.

   Note: Remember that you can use dashboards from any connection or dataset in the same app, provided that the creater/owner has the appropriate privileges.

Adding dashboards to tabs

Each tab in a Cloudera Data Visualization application contains one or more dashboards.

Procedure

1. Add a dashboard to the first tab in your application by clicking a dashboard in the Dashboards menu on the right side.

   You can preview the information about the dashboard by hovering over it.

   Note: You can select recently viewed dashboards, or the dashboards in the selected dataset.

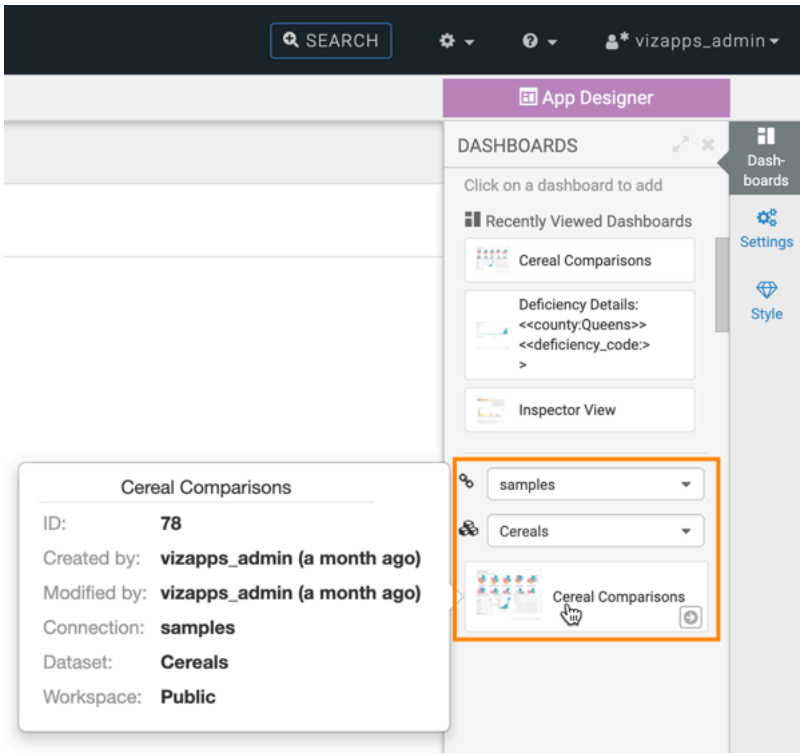

   The newly added dashboard appears in the tab container.

2. Click Save.

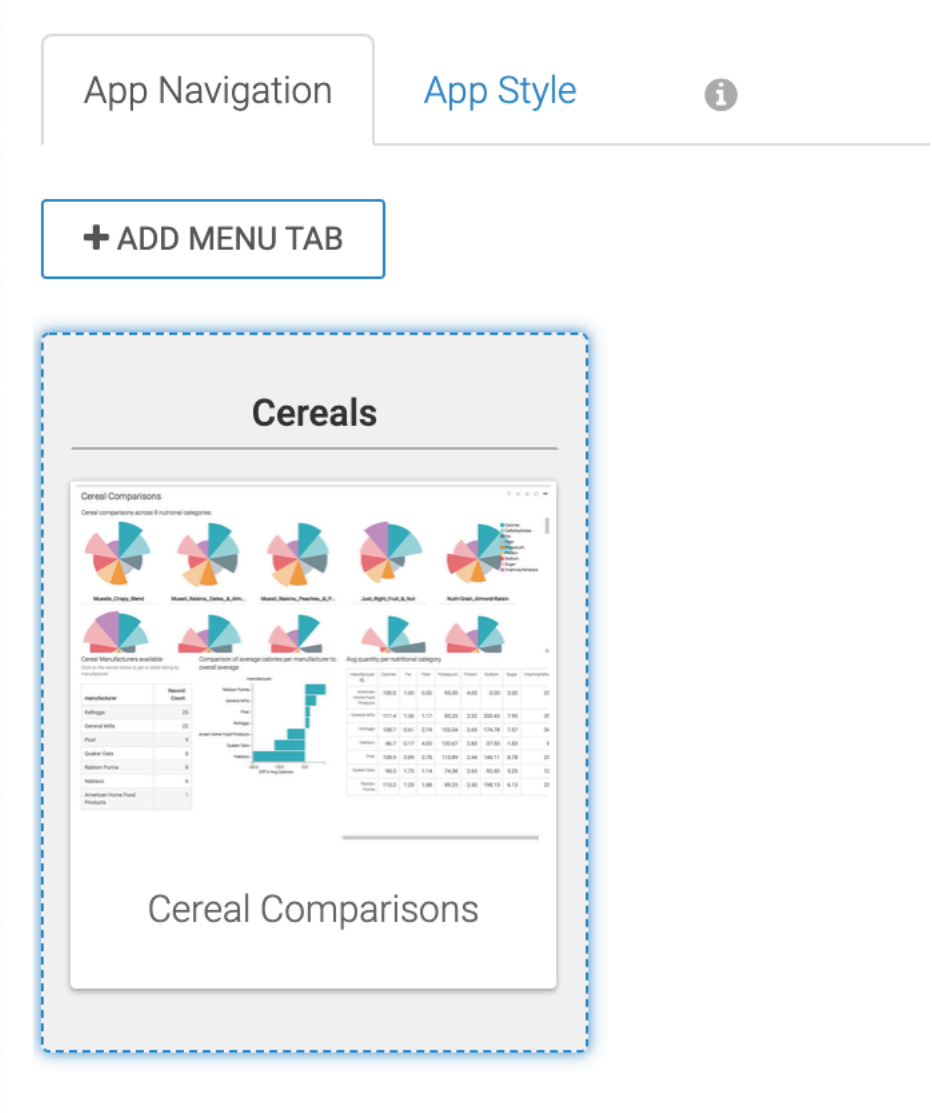

Removing a dashboard from a tab

In the App Designer of Cloudera Data Visualization, it is very easy to remove dashboards.

Procedure

1. In the App Navigation interface, hover over a dashboard in a tab, and click the (remove) icon on the dashboard.

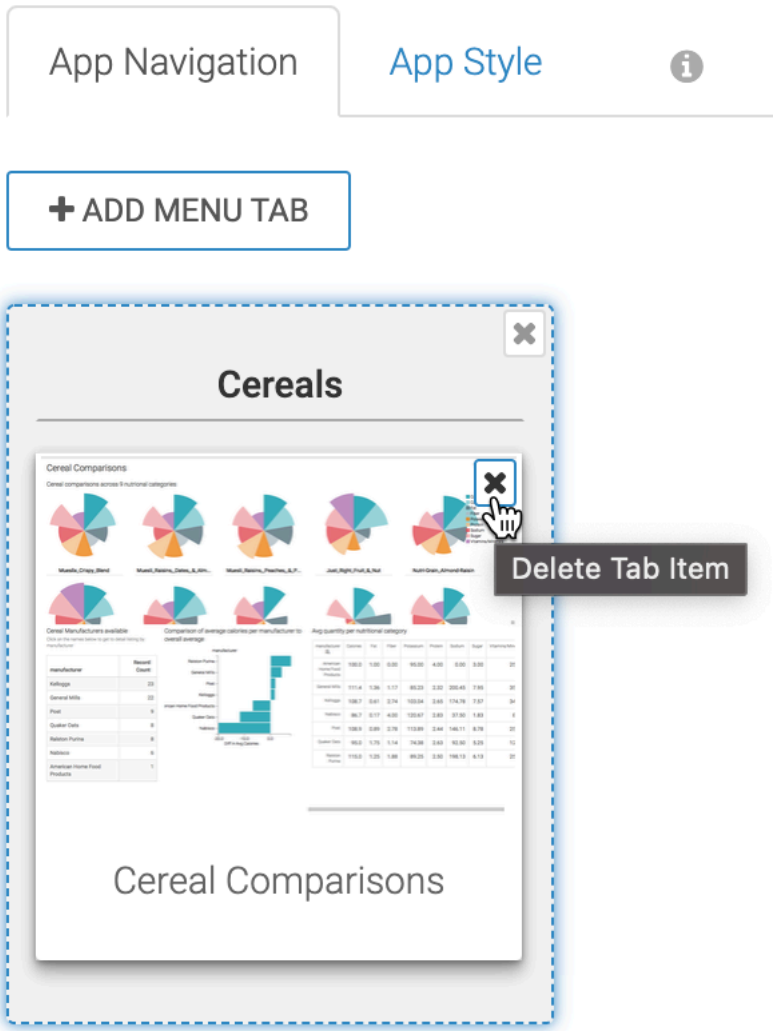

   At this point, the App Navigation interface still has the tab you used, but the dashboard has been deleted.

2. To ensure that the change persists, click Save.

Changing datasets in applications

In a Cloudera Data Visualization application, you can switch between datasets and use dashboards from different datasets, even when sourced from different connections.

Procedure

1. In the Dashboards menu, select a different dataset.

   In this example, the World Life Expectancy dataset is used.

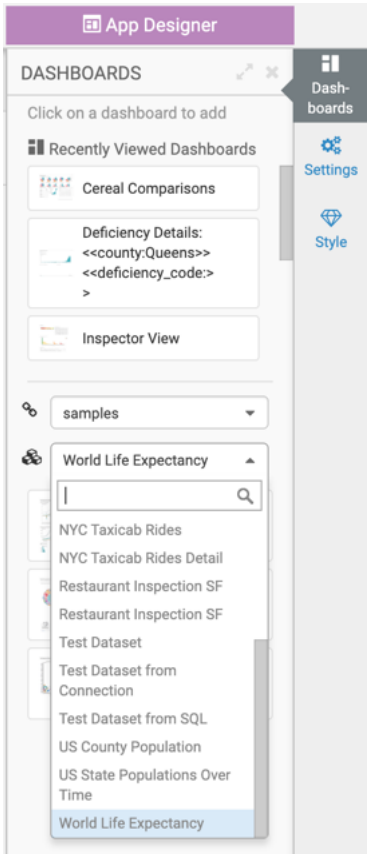

2. Select a dashboard available on the dataset you used.
   In this example, the Life Expectancy Dashboard is selected.

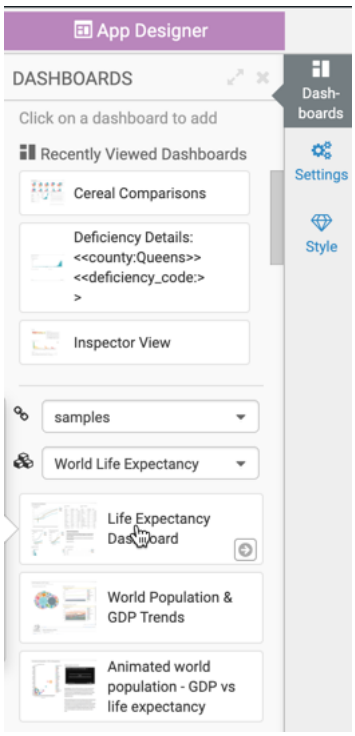

   The newly added dashboard appears in the tab.

Rearranging dashboards

You can easily move dashboards in a Cloudera Data Visualization application, both within the same tab, or to a new tab.

Procedure

1. Click to select the dashboard that you plan to move. We chose the Maps dashboard.

2. Drag the dashboard to a different tab, and drop it.

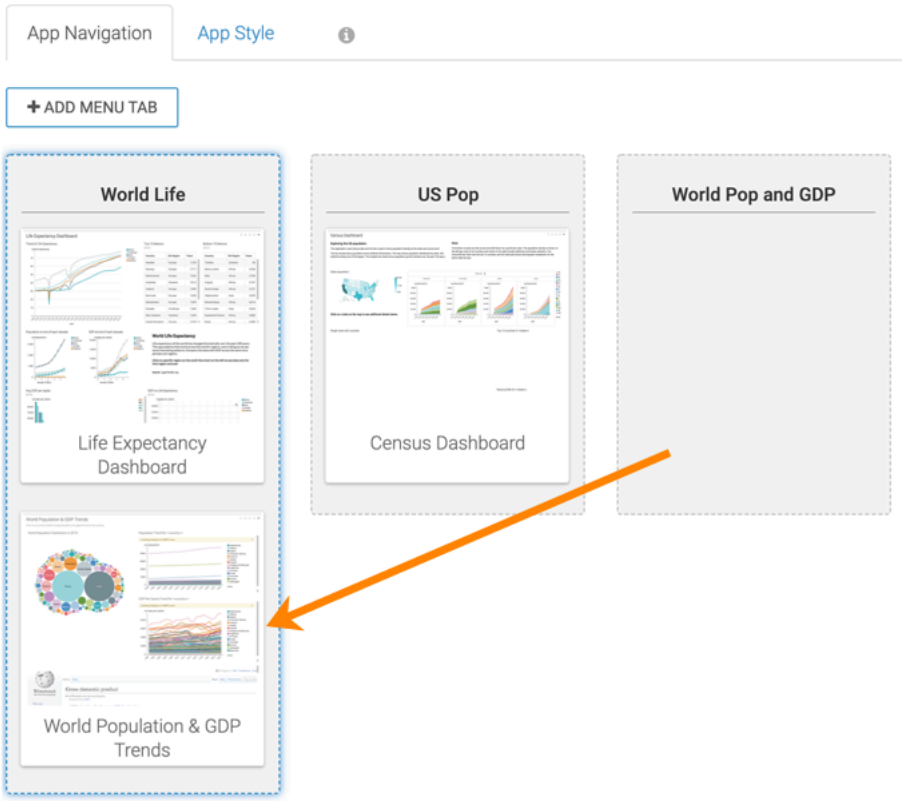

3. You can change the order of dashboards within a tab simply by dragging them within the tab container.

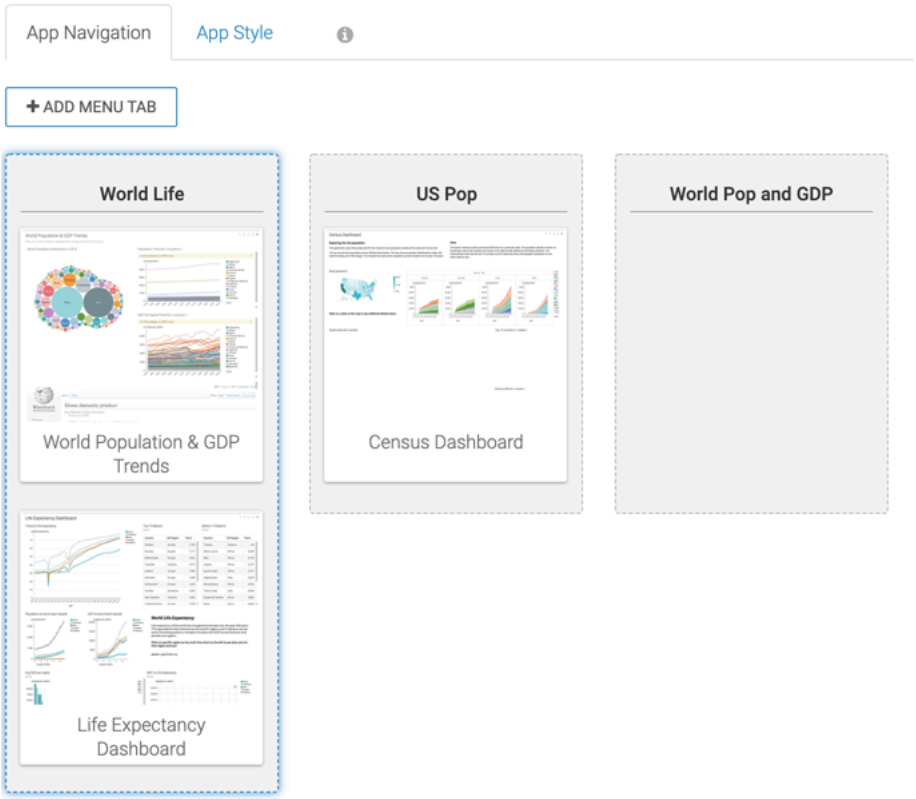

4. You can add the same dashboard to different tabs.

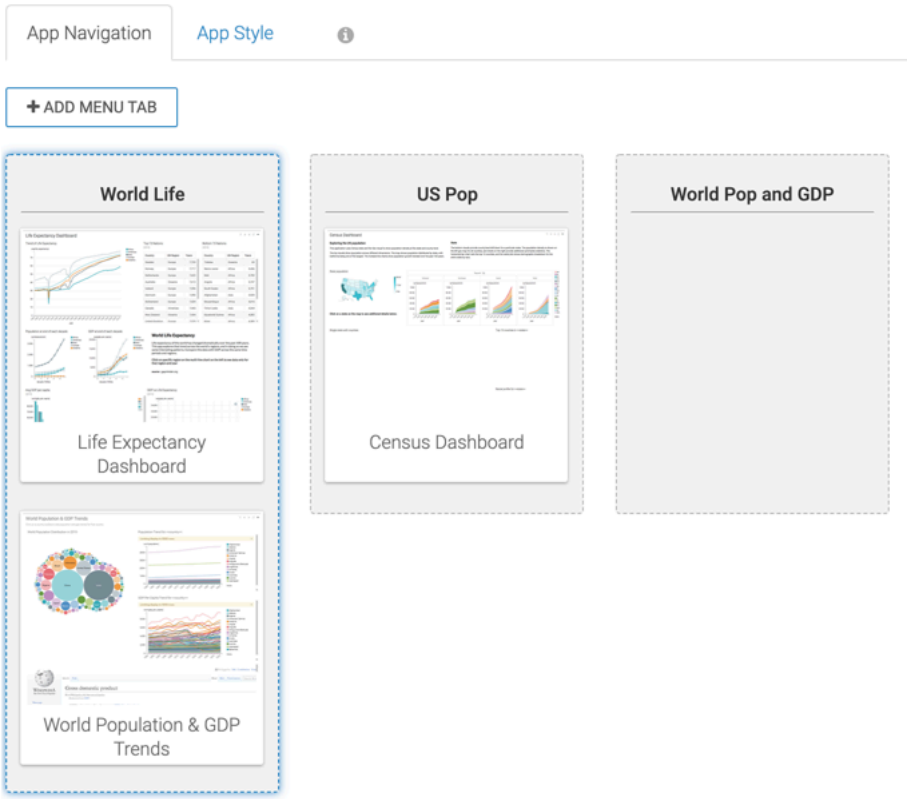

Note: You cannot add a dashboard to the same tab twice. You may move a dashboard from a tab into another tab that contains the dashboard; but the duplicate dashboard does not persist after saving the app.

Rearranging tabs

In the App Designer of Cloudera Data Visualization, you can easily change the order of tabs through drag and drop interaction.

Procedure

1. In the App Navigation interface, click to select the tab that you plan to move.
2. Drag the tab to the first position, and drop it.
3. Click Save.

What to do next

To familiarize yourself with app customization, see *Customizing applications*.

To learn how to launch the app you created, see *Launching applications*.

Launching applications

Cloudera Data Visualization enables you to externalize an app by launching it.

Procedure

1. Navigate to the app you plan to launch.

   The following example uses the app Exploring National Parks.
2. In the top navigation bar of App Designer, click Launch App.

   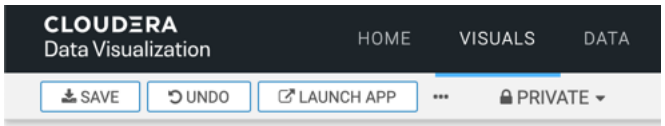

   A new tab appears in your browser window.
3. Open the tab to see the standalone application you just created.

   The externalized, standalone application is fully interactive and has the following features:

   a. The tabs across the top correspond to the names of the apps you defined in the App Designer. The sub-menus correspond to the names of the dashboards; they appear only when there is more than one dashboard in each tab.

   b. The tabs at the bottom correspond to the individual sheets in each dashboard. They appear only if the dashboard has two or more sheets.

   c. The (filter) icon at the top-right corner correspond to the same navigation elements as you would see in a dashboard, Current Parameters.

   d. The Back navigation icon, (back arrow), moves the dashboard backward to the previous parameter state. You can also use the shortcut keys CTRL + B.

   e. The Forward navigation icon, (forward arrow), moves the dashboard forward to the next parameter state. You can also use the shortcut keys CTRL + F.

   f. The Reset navigation icon resets the parameters to the original state of the dashboard. You can also use the shortcut keys CTRL + R.

   g. The Supplemental menu, marked with the (ellipsis) icon, has additional options, such as Get URL. It shows the specific URL address that you can copy and share with others. Pasting this URL into a browser brings up the identical app, opens at the very same visual or dashboard, with specified parameters.

   The URL is of the following type:

   ```
   IP_ADDRESS:PORT/arc/appgroup/APP-ID?initialapp=VIS-ID&sheet=SHEET_ID
     &param.PARAMETER_NAME.data='DATA_LIST'&param.PARAMETER_NAME.
   ```

```
exclude=[in|out]
```

where:

- *IP_ADDRESS* and *PORT* specify the server
- *APP-ID* is the integer that specifies the id of the externalized (launched) app
- *VIS-ID* is the integer that specifies the id of the dashboard or visual
- *SHEET_ID* is the integer that represents the sheet order
- *PARAMETER_NAME* is the name of the filter parameter, if present
- *DATA_LIST* is the list of values that apply to the parameter_name variable
- in the exclude setting, in|out specifies the include/exclude setting of the parameter.

h. At the level of visuals inside the dashboards, hovering over the top right corner enables you to download the data as CSV by clicking the (download CSV)icon.

i. You can click the (expand visual) icon to bring up the visual in a window modal, superimposed over the application.

j. The zoom in - zoom out control is part of the control mechanism for particular visual types, for example interactive maps.

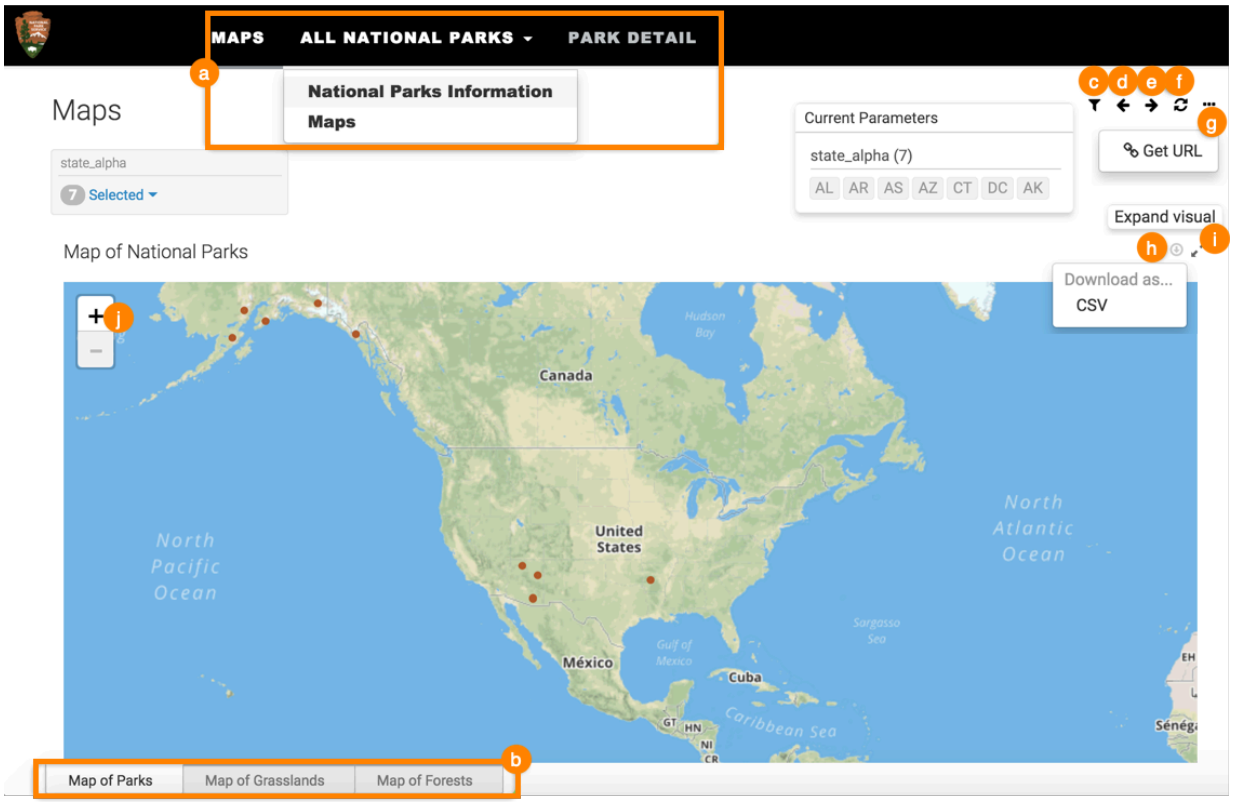

Moving apps between workspaces

Whether you are transitioning your app for personal use, or want to share it with a broader audience, or customizing your workspace organization, Cloudera Data Visualization enables you to manage your apps easily across different workspaces.

About this task

You can create an app in the Private workspace and then change the assigned workspace before sharing it.

Note: Cloudera Data Visualization utilizes Role-Based Access Control (RBAC) permissions to govern access to functionalities in the system. Administrators or users with Additional system privilege can move dashboards and apps to any workspace, including private ones.

On the VISUALS interface

If you do no want to open the app, you can move it to a different workspace on the VISUALS interface.

1. Hover over the app tile or row and use the Move to Workspace function from the Supplemental menu of the tile. You can also use Move to Workspace from the actions available in the top right corner of the preview area of the interface.

   Note: Note The action buttons are only active if you select at least one artifact in the preview area.

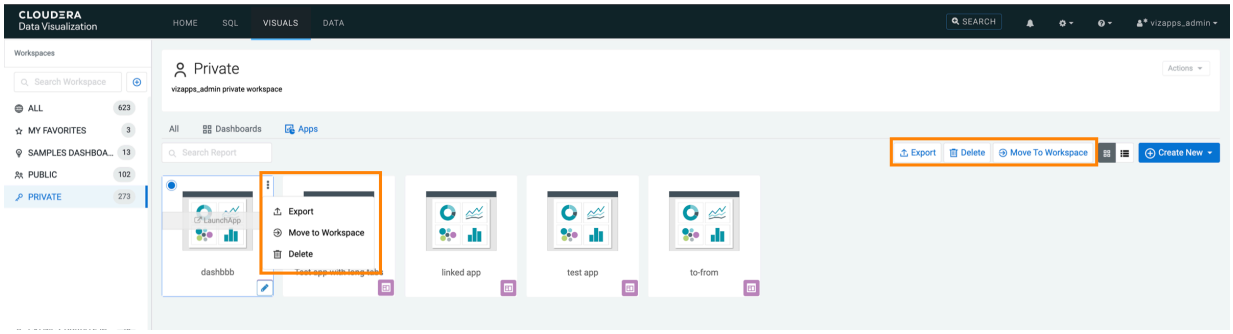

   The Move to Workspace modal window is displayed. It provides information about the app you want to move.

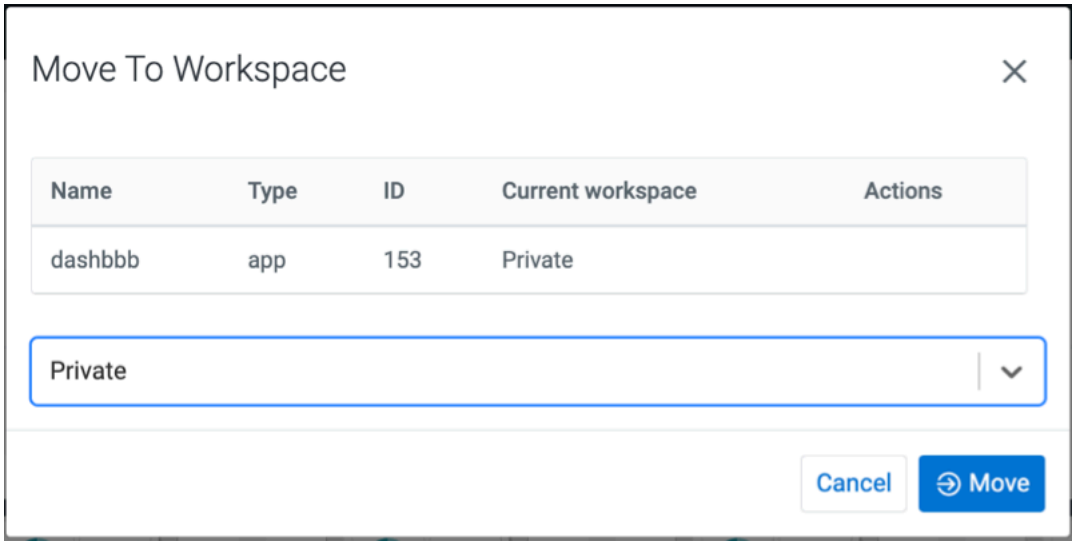

2. Selet the new workspace from the drop-down list and clickMove.

After opening the app

Procedure

1. Find the app you want to move and open it in Edit mode.
2. Click the workspace indicator in the top navigation bar of the App Designer.
3. From the drop-down menu, choose the workspace where you want to move the app.

   In this example, the app is in the Private workspace. Select Public.

4. After selecting the new workspace, click MOVE.

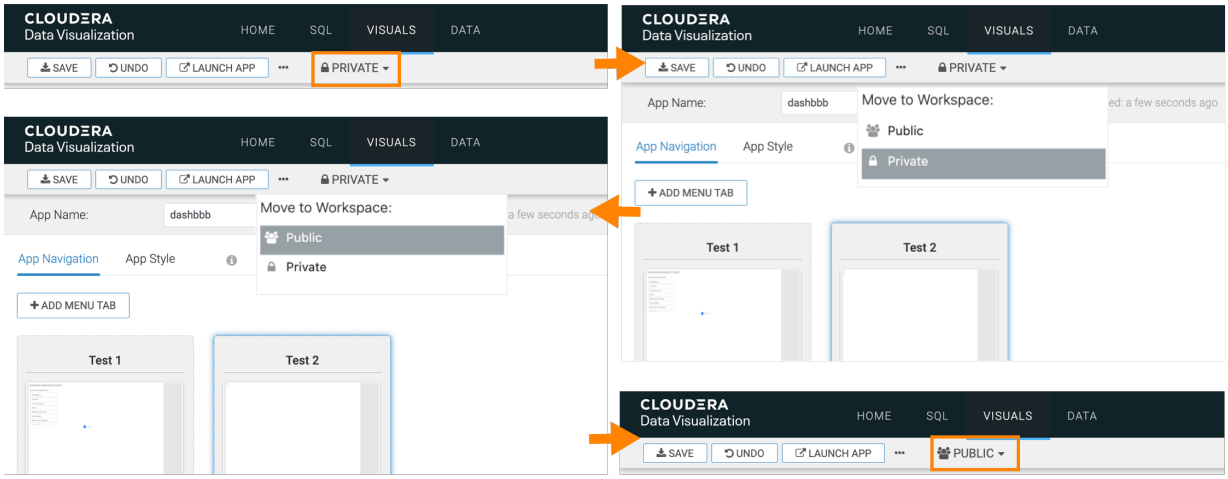

The app is now in the Public workspace. You can verify it by navigating to the Visuals interface, and then selecting and expanding the destination workspace.

This process is flexible and works for moving dashboards between private, public, and custom workspaces, in any direction. Just ensure you have the necessary privileges to perform the action.

Cloning an application

Cloudera Data Visualization enables you to clone an existing application.

Procedure

1. Navigate to the app.
2. In the top navigation bar of App Designer, click the supplemental menu indicated by the (ellipsis) icon, and then select Clone App.

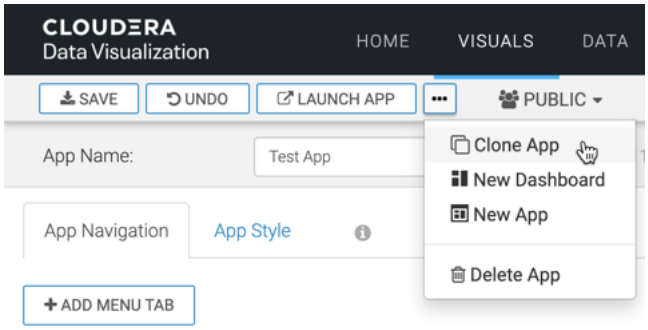

After a brief success message, the new clone app appears on the screen.

The App Name is Clone of …, it has its own ID, and it is in the Private workspace by default.

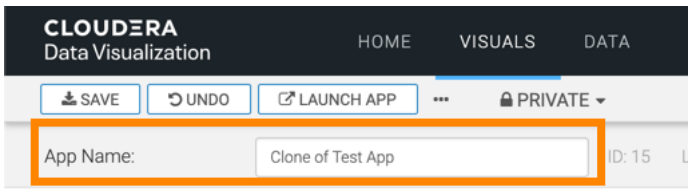

3. Change the App Name and other properties of the clone and save the new app.

Exporting an app

In Cloudera Data Visualization, you can export dashboards, apps, and other visual artifacts. When you export a whole app, the process captures all the dependencies, such as visuals, dashboards, datasets, and custom styles.

Procedure

1. Click VISUALS in the main navigation bar.
2. Select the app that you want to export.

   This updates the action buttons available in the top right corner of the preview area and the Export button becomes active.

   Note: The Export button is only active if you select at least one artifact in the preview area.

3. Click Export.

   This action exports the app to your default download folder.

   You can also use the Suppmental menu of the app tile to initiate the export.

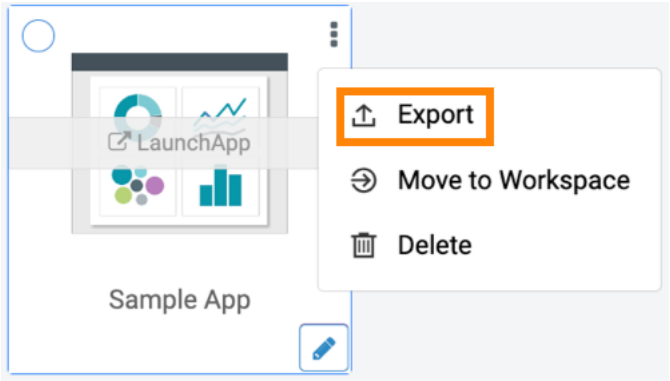

   The Export Visual Artifact modal window appears.

   In this example, the Sample App is used.

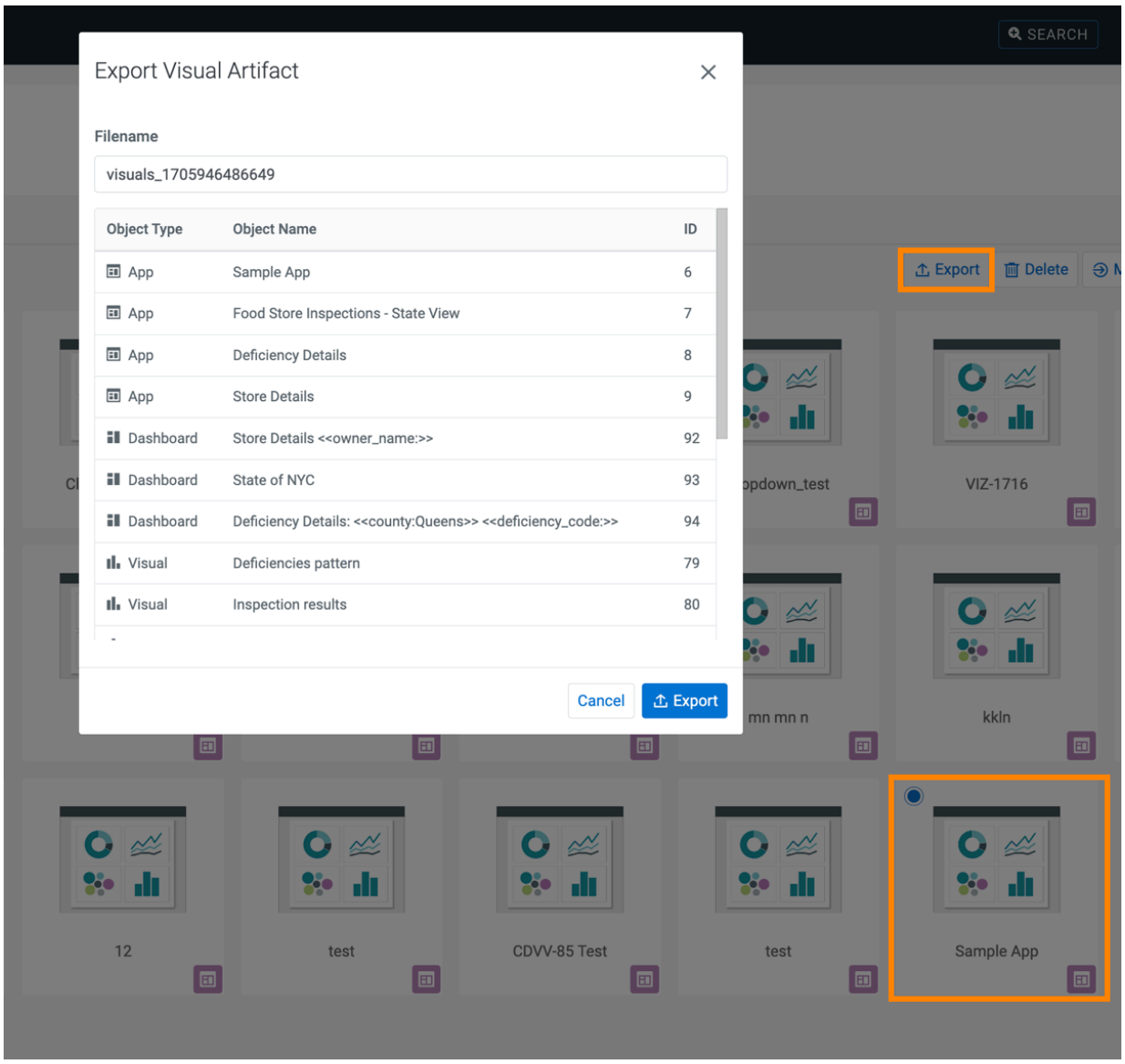

The export file includes the definition of the app and any subgroups that it may have, the visuals and dashboards in the app, and all necessary datasets. When available, the export file also includes custom style dependencies.

4. Change the name of the file you are exporting.
5. Click Export.

   This starts the download of the export file to your default download folder. You can confirm the success of the export by finding the file on your system.

What to do next

For instructions on how to import an app, see *Importing an application*.

For more information about the visual artifact migration utility, see *Exporting and importing visual artifacts*.

Related Information

Importing an application

Exporting and importing visual artifacts

Importing an application

In Cloudera Data Visualization, you can import dashboards, apps and other visual artifacts exported from another system. All you need is the appropriate JSON file that captures the complete definition of the app, including its dependencies: its own app definition, subgroup app definitions, visuals, dashboards, datasets, and custom styles.

Procedure

1. Click DATA on the main navigation bar.
2. Select the connection where to import the artifacts.

   In this example Samples has been selected showing connecting to the same data source for both systems.
3. At the top of the screen, click the Supplemental menu.
4. In the Supplemental menu, click the Import Visual Artifacts option.

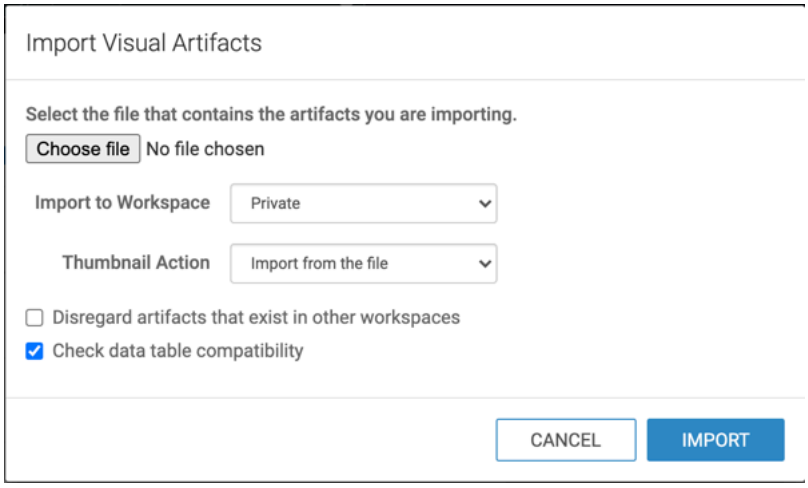

   The Import Visual Artifacts modal window appears.

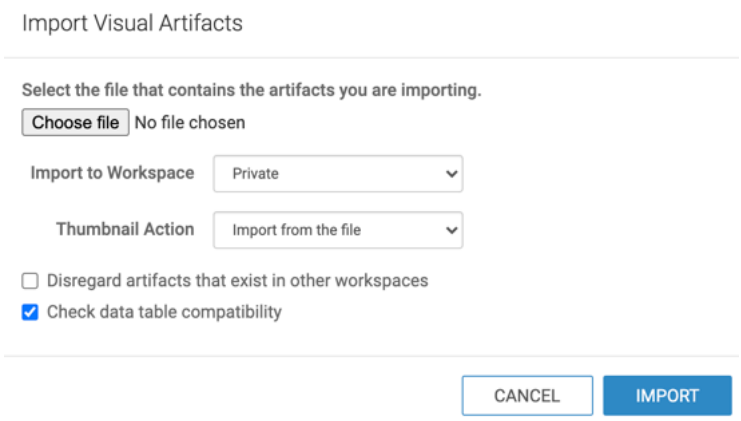

5. Click Choose File.
6. Select the file you want to import.
7. Specify the destination workspace in the Import to Workspace drop-down menu.

   By default, Cloudera Data Visualization imports the visual artifacts into your Private workspace. However, you can choose the Public workspace, or another custom workspace as well.

8. Specify how you want to handle thumbnails.

   There are three options:

   - Import from the file
   - Generate new thumbnails
   - Skip thumbnail import

   By default, Cloudera Data Visualization imports the thumbnails from the file. However, you can choose to generate new thumbnails during the import process, or skip thumbnail import.

9. Define the Disregard artifacts that exist in other workspaces option.

   Before importing an artifact, Cloudera Data Visualization searches for it on the system. If it already exists, it is either updated (if changed from a previous import), or left 'as is'. If it does not exist, it is imported as a new visual artifact. By default, the search domain is the entire system. By selecting this option, you are limiting the search to the specified workspace.

10. Select the Check data table compatibility option.

    This option is 'on' by default.

11. Click IMPORT.

    After processing the import JSON file, Cloudera Data Visualization opens the Import interface to show its visual artifacts. In this example, it displays the visual artifacts that you saved earlier into a JSON file during export. All items saved into the app_documentation_collateral.json file during export appear on the interface. The Planned Action column shows that this is a new import for some artifacts and they don't have an assigned ID.

    In the Importing a dashboard example, the dashboard Flight Delay Information, and its dependencies: visuals Departure Delay by Airline & Day of the week, Originating Airport, and Originating Airport by State, the Custom Style Category colors in tables, and the dataset Flight Data were all imported. For details, see *Importing a dashboard*. The dataset has changed (the description parameter was edited), so the Planned Action is to Replace. The rest of these artifacts is unchanged, so they appear with Planned Action set to None.

12. Click ACCEPT AND IMPORT.

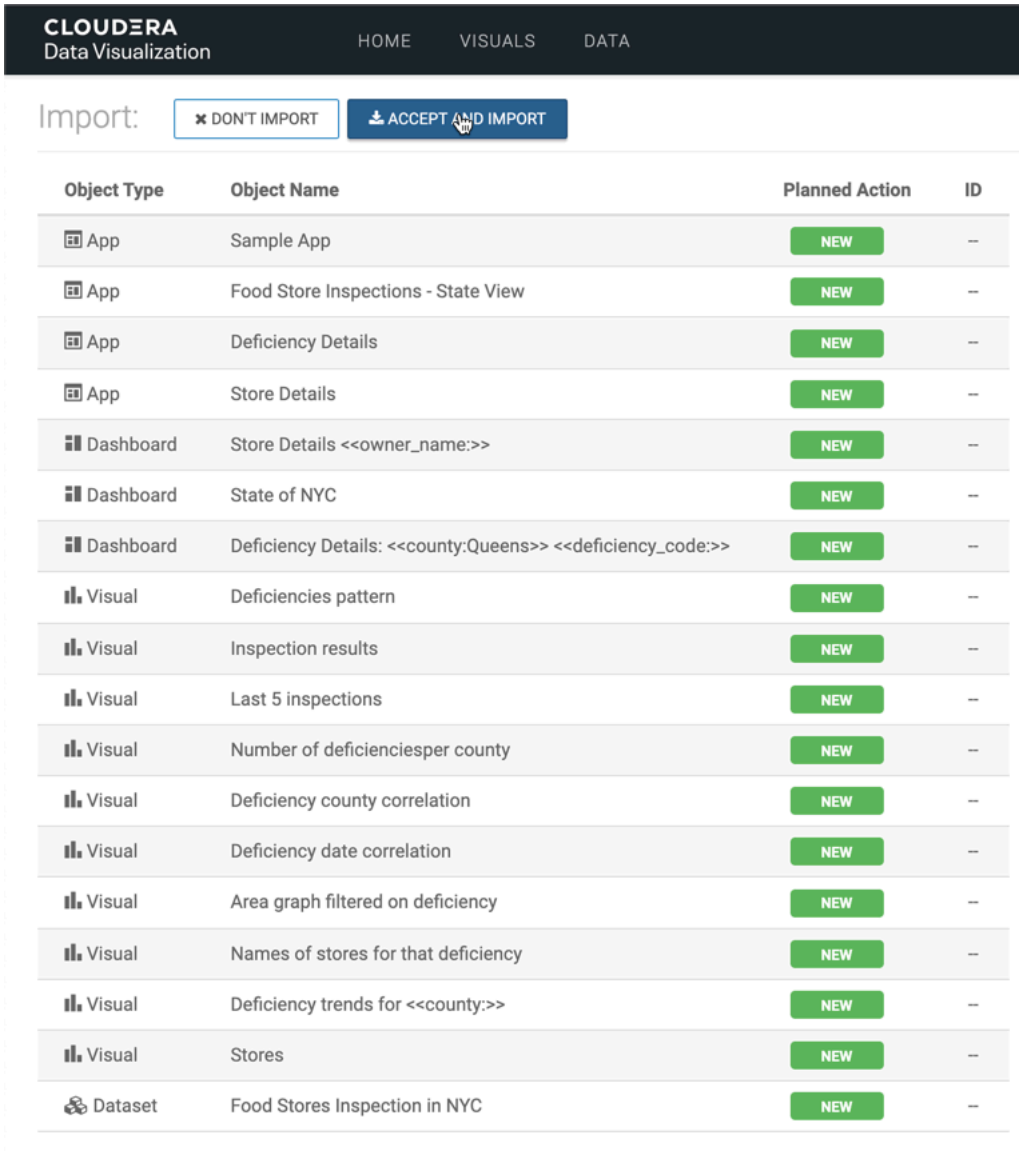

Object Type	Object Name	Planned Action	ID
App	Sample App	NEW	--
App	Food Store Inspections - State View	NEW	--
App	Deficiency Details	NEW	--
App	Store Details	NEW	--
Dashboard	Store Details <<owner_name:>>	NEW	--
Dashboard	State of NYC	NEW	--
Dashboard	Deficiency Details: <<county:Queens>> <<deficiency_code:>>	NEW	--
Visual	Deficiencies pattern	NEW	--
Visual	Inspection results	NEW	--
Visual	Last 5 inspections	NEW	--
Visual	Number of deficienciesper county	NEW	--
Visual	Deficiency county correlation	NEW	--
Visual	Deficiency date correlation	NEW	--
Visual	Area graph filtered on deficiency	NEW	--
Visual	Names of stores for that deficiency	NEW	--
Visual	Deficiency trends for <<county:>>	NEW	--
Visual	Stores	NEW	--
Dataset	Food Stores Inspection in NYC	NEW	--

Results

After the import completes, a success message appears on the Import interface. All the artifacts have an assigned ID, which are generated by the system, sequentially. Visuals/Dashboards and datasets have separate ID queues.

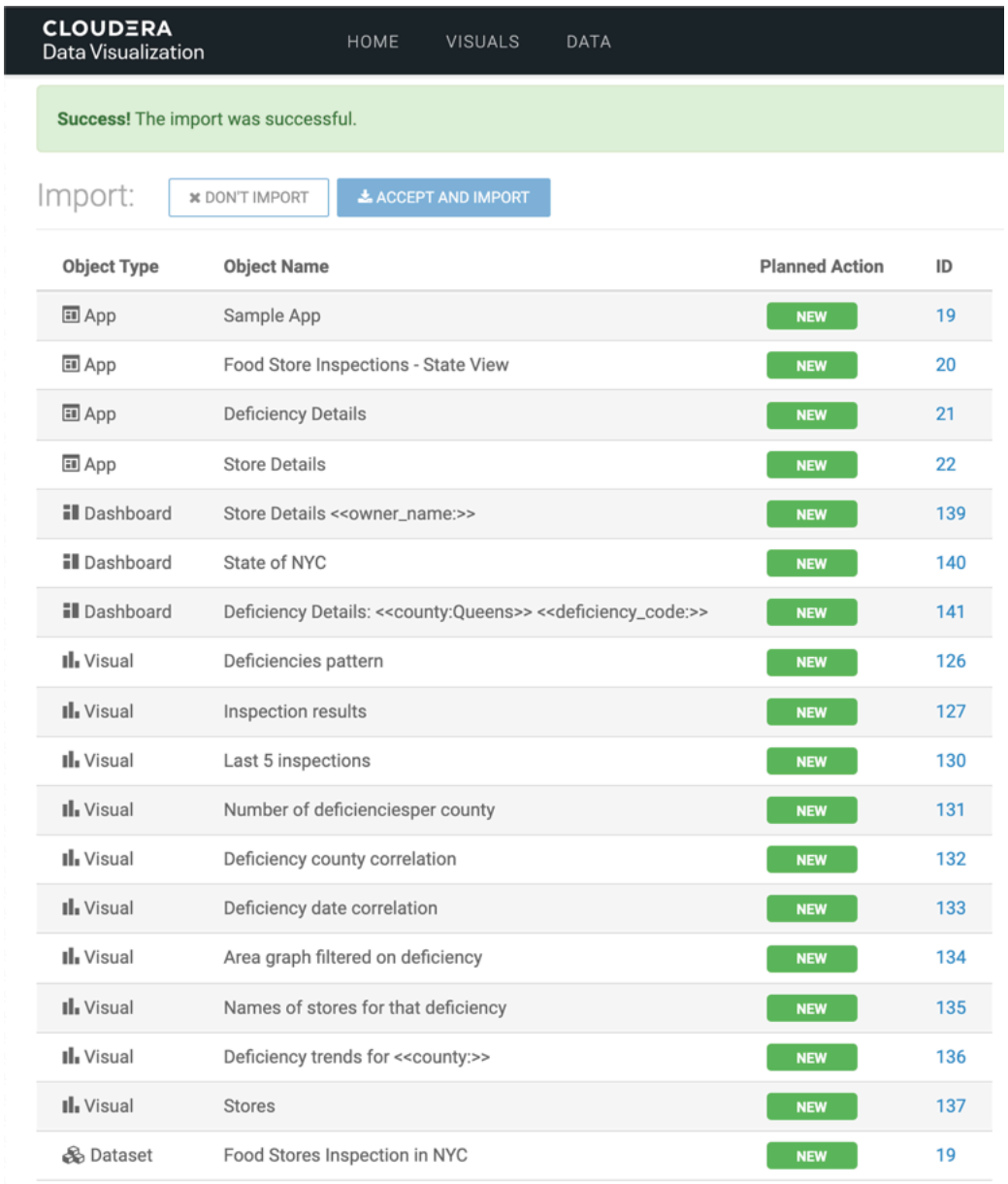

Object Type	Object Name	Planned Action	ID
App	Sample App	NEW	19
App	Food Store Inspections - State View	NEW	20
App	Deficiency Details	NEW	21
App	Store Details	NEW	22
Dashboard	Store Details <<owner_name:>>	NEW	139
Dashboard	State of NYC	NEW	140
Dashboard	Deficiency Details: <<county:Queens>> <<deficiency_code:>>	NEW	141
Visual	Deficiencies pattern	NEW	126
Visual	Inspection results	NEW	127
Visual	Last 5 inspections	NEW	130
Visual	Number of deficienciesper county	NEW	131
Visual	Deficiency county correlation	NEW	132
Visual	Deficiency date correlation	NEW	133
Visual	Area graph filtered on deficiency	NEW	134
Visual	Names of stores for that deficiency	NEW	135
Visual	Deficiency trends for <<county:>>	NEW	136
Visual	Stores	NEW	137
Dataset	Food Stores Inspection in NYC	NEW	19

For instructions on how to export an app, see *Exporting an application*.

For more information about the visual artifact migration utility, see *Exporting and importing visual artifacts*.

Related Information

Exporting an application

Exporting and importing visual artifacts

Deleting an app

It is very simple to delete an app in Cloudera Data Visualization.

Procedure

1. Click VISUALS in the main navigation bar.

2. Select the app that you want to delete.

   This updates the action buttons available in the top right corner of the preview area and the Delete button becomes active.

   Note: The Delete button is only active if you select at least one artifact in the preview area.

3. Click Delete.

   You can also use the Suppmental menu of the app tile to initiate the export.

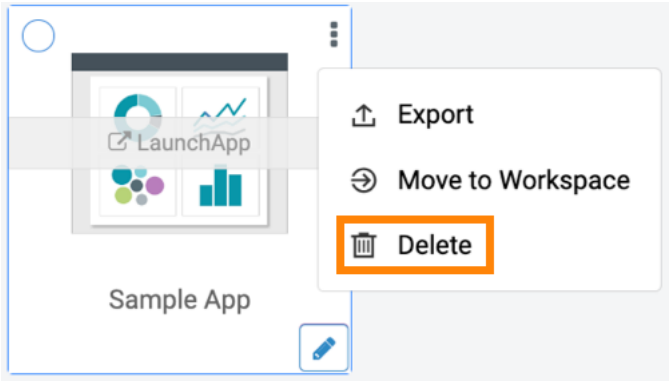

   The Delete Visual Artifacts modal window appears.

   In this example, the Sample App is used.

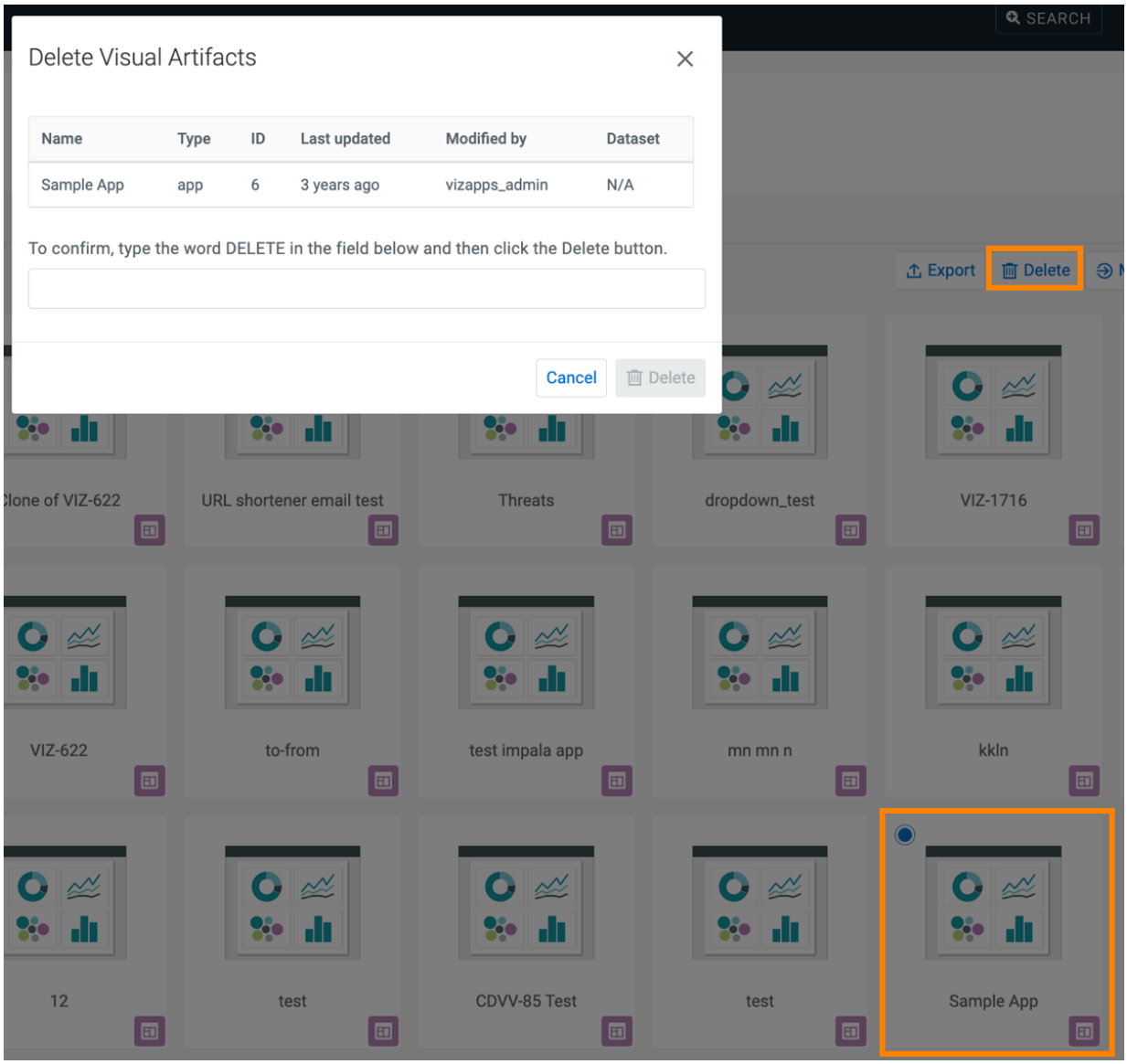

4. Type DELETE in the text box to confirm the action and click Delete.

   The Delete Confirmation modal window appears.

Results

Cloudera Data Visualization refreshes the Apps view of the VISUALS interface. The deleted application is not available any more.

Cloudera Data Visualization 7.2.9

Data Discovery with Context Menus

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using context menu options....4

Drill-down and drill-up data discovery....5
Drill-down data discovery....5
Drill-up data discovery....7

Drill-through data discovery....8

Drill-into data discovery....13
Limiting drill-into options....15

Filtering all visuals in a dashboard....17
Prepare target visuals....18
Prepare source visuals....18
Prepare and use the dashboard....18

Discovering detailed data behind a visual....23

Filter data using the include option in the context menu....27

Filter data using the exclude option in the context menu....28

Filtering data using legends....30

Using context menu options

You have several approaches to data discovery from the context menu of the dashboard in Cloudera Data Visualization.

You can view the context menu at both the dashboard and the visual level. Context menus can be specific to the dashboard or visuals, or common to both. Additionally, you can use context menus directly on the graphical elements of the visual (for example, a particular bubble in the packed bubbles visual), or on the legend for the visual.

- Context menu options only in dashboards on page 4
- Context menu options only in visuals on page 4
- Context menu options common to both dashboards and visuals on page 5

In addition to using the context menu options to discover your data, you can drill up and down the hierarchical tree by Creating new dimension hierarchies. For more information, see Drill-down and drill-up data discovery.

Context menu options only in dashboards

In dashboard Edit mode, you can enable the following three options. The setting of these options control the behavior of context menu options in View mode.

No action on click

This option shows the context menu in View mode, but does not filter the target visuals on clicking any mark of the source visual. You must click the Filter All option in the context menu to filter target visuals in the dashboard, based on the selected element of the source visual.

Show context menu on click

When you enable context menus at the dashboard level through the Displaying context menus in visuals setting option in the General menu, the No Action on Click context menu appears as Show Context Menu on Click. If you disable the context menus, the option appears as No Action on Click.

Enable filter on click

In View mode, if you click on a mark of a visual, this option filters the target visuals in the dashboard, based on the selected element of the source visual. Context menu is not visible in this option. In Edit mode, you can test the filter behavior by clicking on Test click behavior.

Navigate to Dashboard

In View mode, if you click on a mark of a visual, it navigates you to a different dashboard. For more information, see Drill-through data discovery.

Context menu options only in visuals

The following context menu options are only available in visual Edit mode. These options are enabled by default.

Include

This option enables you to click a mark of the visual and include selected values in the filter shelf of the visual designer.

For more information, see Filter all data discovery.

Exclude

When you click on a mark of a visual, this option enables you to add an expression in the filter shelf of the visual designer that excludes the selected values in the visual. For more information, see Filter data using the exclude option in the context menu.

Context menu options common to both dashboards and visuals

The following options appear at both the dashboard and the visual level.

Filter All

This option is enabled by default.

This option filters all the target visuals in the dashboard, based on the selected element of the source visual. For more information, see Filter all data discovery.

Drill Into

In Visual Designer, you must first enable Show Drill Into context menu option in the General menu. This setting is available only on Bar, Line, Area, and Grouped Bar visual types. For more information, see Enabling Drill Into data discovery.

This option enables you to view subsets of data across other dimensions. For more information, see Drill into data discovery.

Show Detailed Data

In Visual Designer, you must first enable Show detailed data option in the Marks menu. For more information, see Showing detail data button in context menu.

This option enables you to click on a mark of a visual, such as specific bar, slice of radial, and so on, to view the data directly associated with that portion of the plot. For more information, see Detailed data discovery.

Drill-down and drill-up data discovery

The following conditions apply to Drill-down/up functionality:

- This feature is only available on dimensions that are part of predefined hierarchies.
- These hierarchies may only be defined by users with administrative privileges, and at the level of the dataset.
- Hierarchies, as implied, combine related and 'nested' dimensions of the dataset, progressing from largest to smallest measurement.
- Dimensional hierarchies by their very nature do not help discover how potentially unrelated dimensions influence the results.

To help with the use of this feature, you can find here some examples for bar charts.

Drill-down data discovery

To discover your data in Cloudera Data Visualization, you can navigate up and down the hierarchical tree.

Procedure

1. To use drill-down, place the pointer over one of the data labels along the horizontal axis, and note that it changes color. This is because the data labels are link-enabled.

2. Click the label to drill down to a lower level of the hierarchy.

   In the following graphic, we drilled through these levels:

   - Level 1: un_region = 'Americas'
   - Level 2: un_subregion = 'South America'
   - Level 3: country

   Note: Every time you move down a hierarchy level, the scale of the vertical axis adjusts automatically.

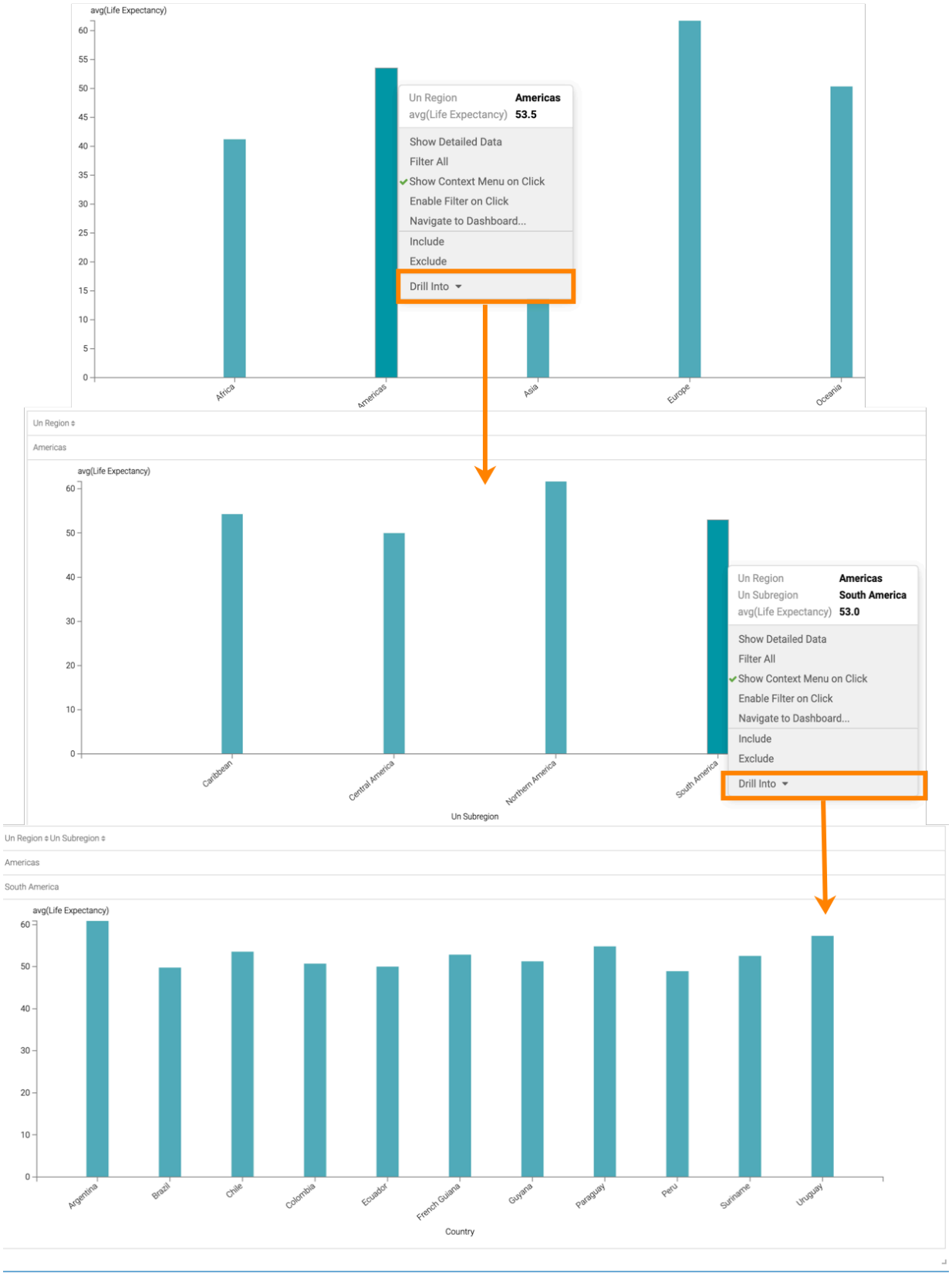

Drill-up data discovery

Procedure

1. Place the pointer over the label of the horizontal axis itself, and note that it changes color. It is also link-enabled.
2. Click the horizontal axis label to go up the hierarchy levels.

   To use drill-up,

   In the following graphic, we moved up along these levels:

   - Level 3: country where un_region = 'America' and un_subregion = 'South America'
   - Level 2: un_subregion = 'South America' where un_region = 'America'
   - Level 1: un_region

   Note: Every time you move up a hierarchy level, the scale of the vertical axis adjusts automatically.

Drill-through data discovery

Cloudera Data Visualization lets you navigate from one dashboard to another, with parameter passing.

About this task

Dashboards support drill through data discovery, where clicking on an element of the visual in a dashboard opens a target dashboard. You can configure drill-through to pass parameters to the target visual.

Drill-through navigation has the following restrictions:

- The navigation link is at the level of the dashboard; you cannot specify a particular visual as a target.
- The parameters used in drill-through data discovery include these:
  - Parameters passed to the source visual from a URL
  - Parameters that correspond to the clicked element
  - When navigating within an application, application scope parameters
- The parameters not supported in drill-through data discovery are:
  - Parameters outside the application scope
  - Explicit parameters defined on the source visual

Follow this procedure to enable drill-through data discovery.

Procedure

1. With the dashboard in Edit mode, click an element of a visual, and select the Navigate to Dashboard option from the menu.

The Pick a dashboard window modal. is displayed.

2. Specify the option Pass current parameters when navigating.

3. Click the Open Target Dashboard in menu, and select one of the options:
   - Current Tab
   - New Tab
   - New Window

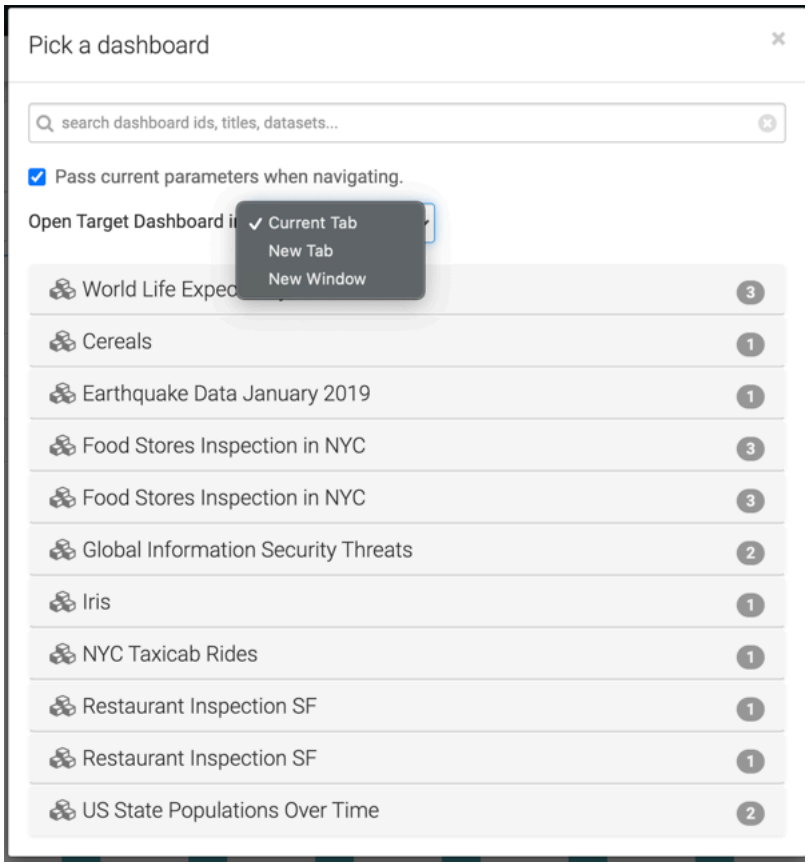

4. You can use the Search bar to find the desired target dashboard, and then select it by clicking one of the results.

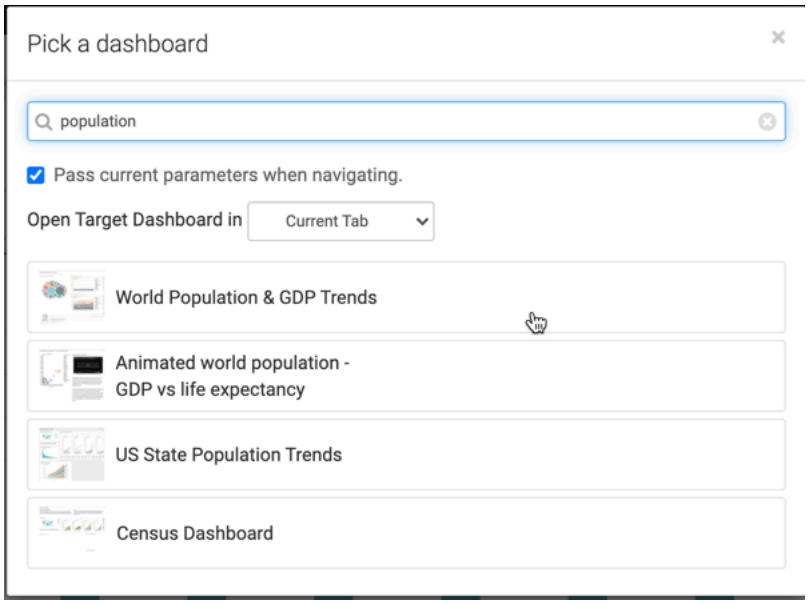

Alternatively, you can expand one of the datasets, and select one of its dashboards.

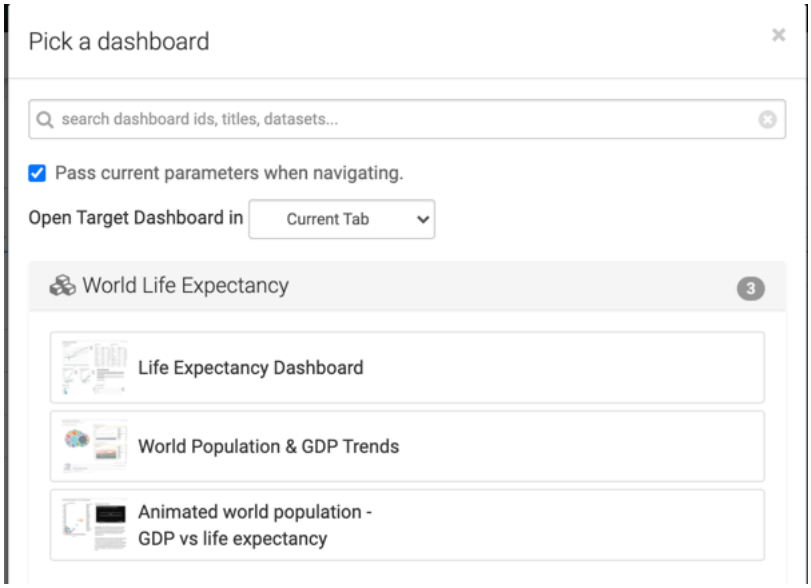

5. To check that the configuration leads to the correct dashboard, with correct options, click on an element in the dashboard, and note the Navigate to ID:145 in the drop-down menu.

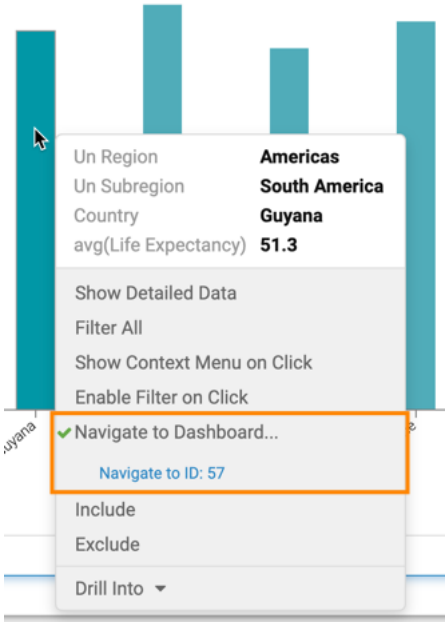

6. Click Save to preserve the changes to the dashboard.
7. Click View to change to View mode.
8. In View mode, click an element of a graph. In this example, we are clicking on a bubble that represents Argentina.

   When the target dashboard is displayed, it preserves the parameters of the click, and builds the target visuals accordingly.

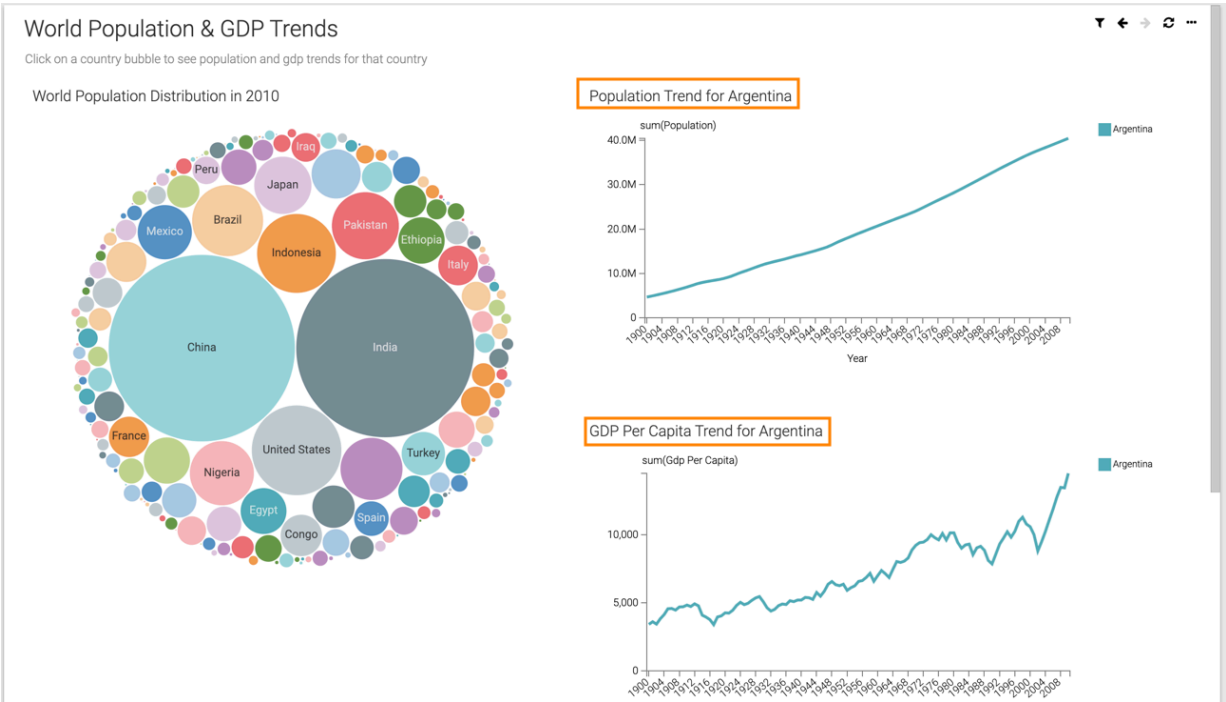

Drill-into data discovery

In Cloudera Data Visualization, You can drill into your data to view subsets of data across other dimensions.

About this task

Drill-into enables you to discover the more granular information for a particular data point by examining other dimensions of your data. It uses the exact portion of the graph you click to determine the filters, and the selected field as the new horizontal axis. You may further limit the fields available in drill-into.

Note: Drill-into data discovery is available on Bar, Line, Area, Grouped Bars, and Map (choropleth only) visuals, and also on trellised charts of these visual types.

Before you begin

To discover data, you must first enable the Show Drill Into context menu option in the General menu. For instructions, see Enabling Drill Into data discovery.

Procedure

1. Open a dashboard that contains the visual you are examining. This example shows it in Edit view.
2. Hover over the visual to determine what portion of the graph to examine further.

   In this example, the line for Europe, in year 1990 is selected. These values become filters in the visual generated by the drill-into choice.

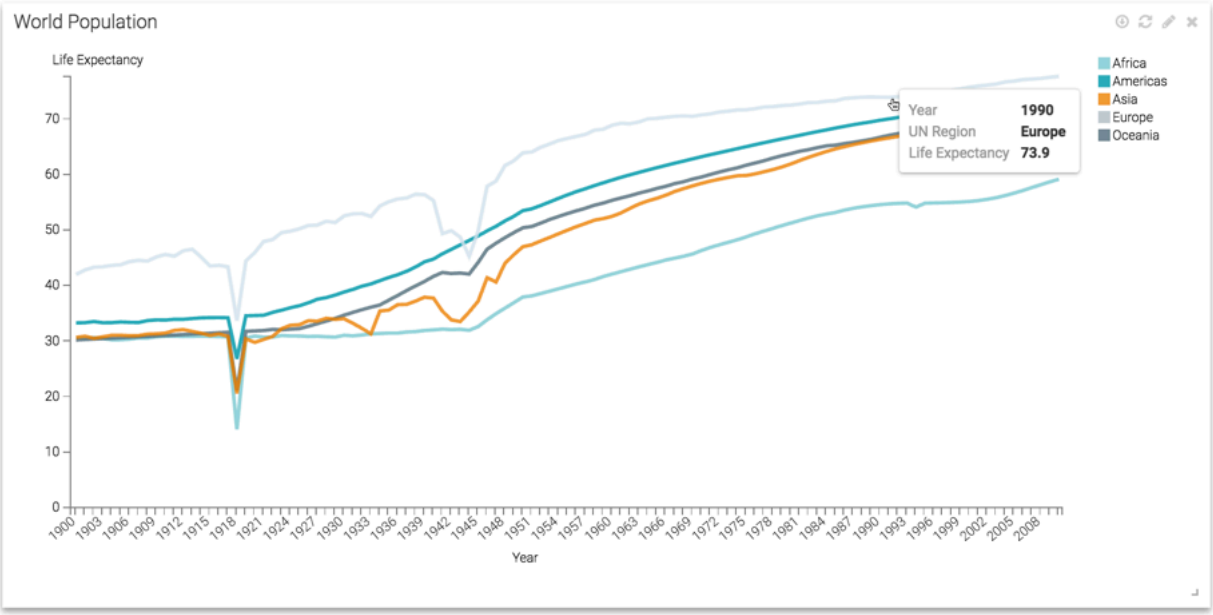

3. Click the selected part of the graph.

   Cloudera Data Visualization expands to show the available data discovery options.

4. Select the Drill Into option, and then select the appropriate field from the drop-down menu.

Note: You have an option to limit the available fields. See *Limiting drill-into options*.

In this example, the country field is selected.

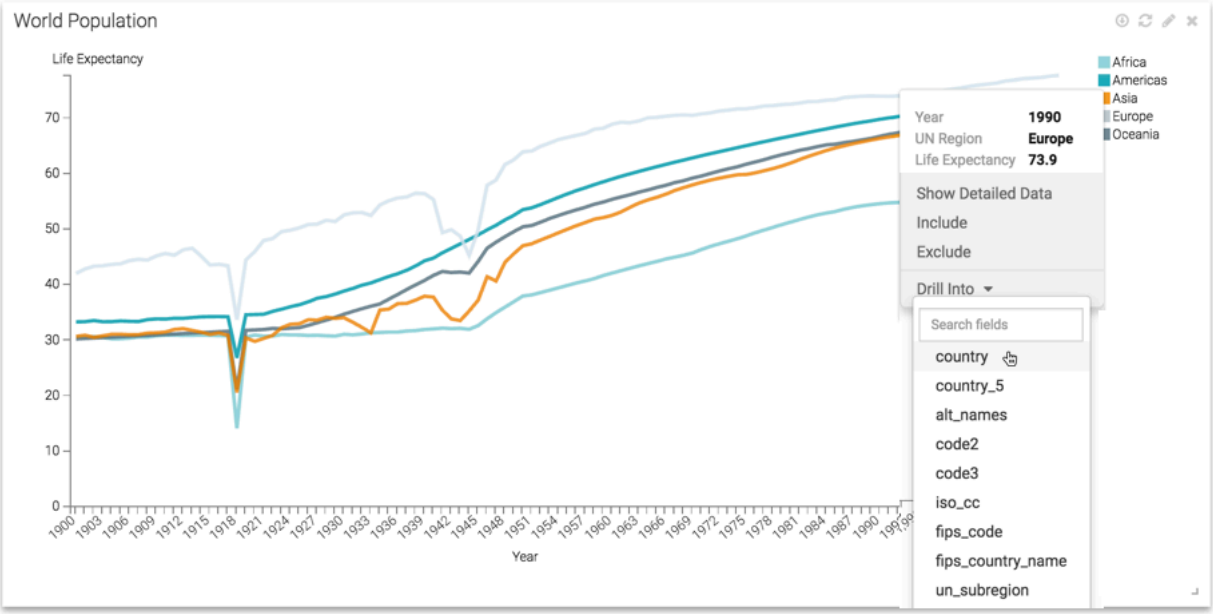

Results

The original visual is replaced with a version that implements the filters specified earlier: Europe in year 1990. The field chosen earlier specifies the horizontal axis, where appropriate. In Choropleth Map visuals, it changes the granularity of regions represented, moving from County, to State, to County, and so on.

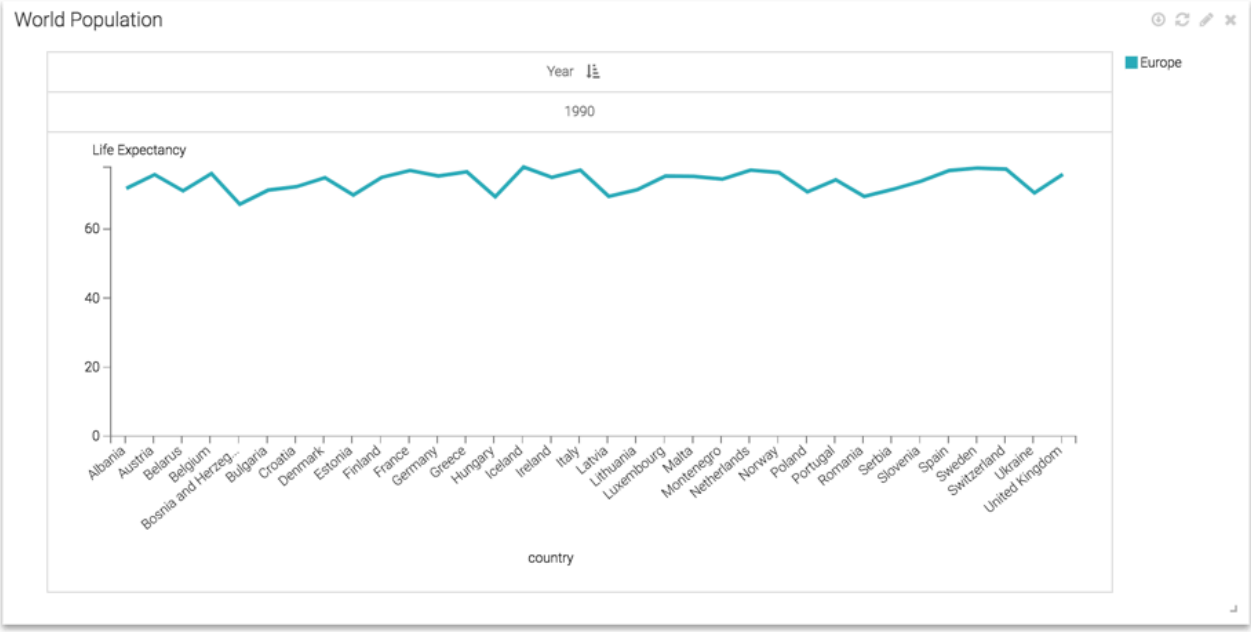

Limiting drill-into options

By default, all dimensions are visible when implementing drill-into data discovery, but you can limit the available fields to a smaller subset.

About this task

To limit the fields available for drill-into to a smaller subset, you must place these fields onto the Drill shelf. When specifying a limit for Choropleth Map fields, these fields must be Geo Type fields, or cast to Geo Type.

Procedure

1. Open the dashboard in Edit mode and click the (pencil/edit) icon of the visual to edit it.

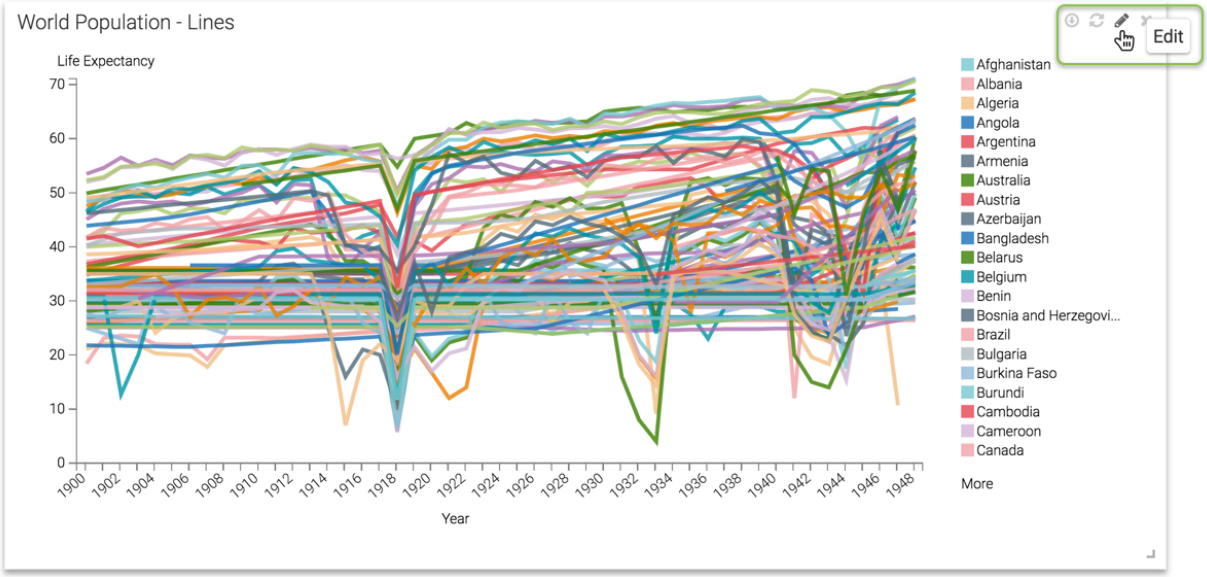

2. In the visual, add the desired fields to the Drill shelf.

   In this example, the fields un_region and un_subregion have been added.

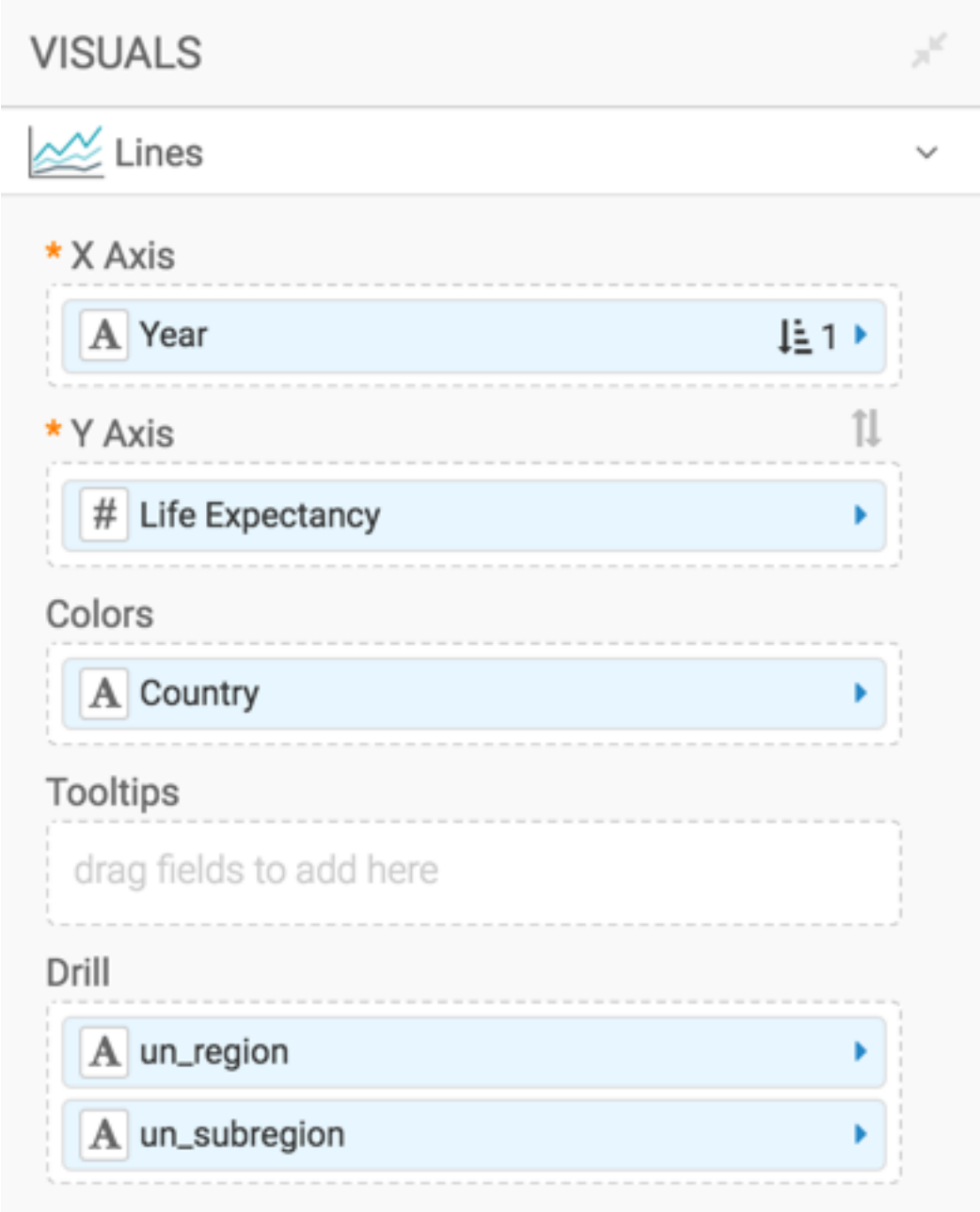

3. SAVE and CLOSE the visual.

Results

If you use the drill-into option, the menu shows only the fields you specify. In this example these are: un_region and un_subregion.

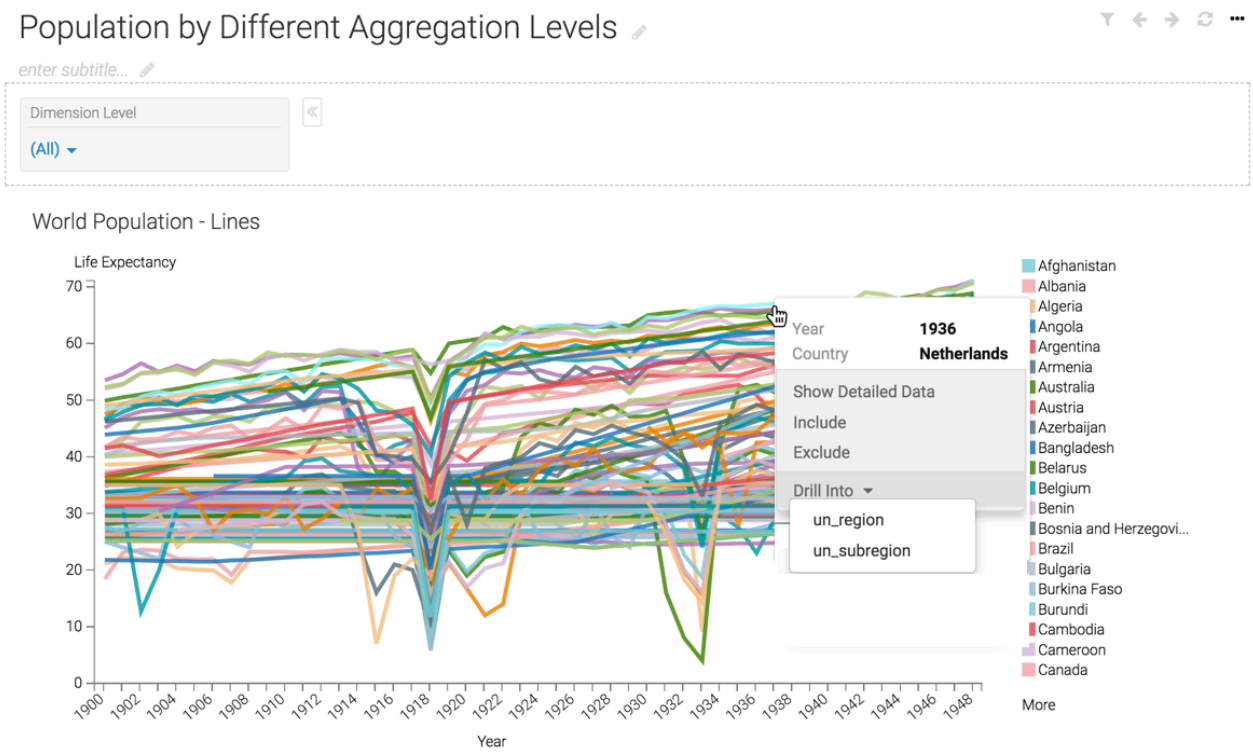

Filtering all visuals in a dashboard

You can discover your data, by filtering all visuals in the dashboard, based on the selected element of the source visual.

About this task

The Filter All data discovery mode lets you to define a visual in the dashboard as a 'source', so clicking on a specific element of the source controls the appearance of other visuals in the same dashboard. The source sends filter parameters to the target visuals, which they use to define the output. Keep in mind that any visual can be both a source and a target when passing filter parameters.

To discover data in View mode, you must first enable the context menu, as described in Displaying context menus in visuals.

The following conditions apply to the Filter All functionality:

- This method of data discovery works across visuals in a single dashboard.
- Scope matters:
  - By default, the source visual sends parameters with Explicit Scope. To change this, select the Dataset Scope. For more information, see Sending parameters with dataset scope.
  - By default, the target visuals receive parameters with Dataset Scope. To change this select the Explicit Scope. For more information, see Using parameters with explicit scope.

    When the target visual receives filter parameters with Explicit Scope, it must have the corresponding dataset fields on the Filters shelf.
- Depending on your use case, you may have to de-select the sending of aggregates as parameters. For more information, see Sending aggregates as parameters.

  The following steps show you how to set up a dashboard with Filter All data discovery options, using dataset scope.

Prepare target visuals

In this example, you can see the settings for two sample target visuals.

Procedure

- The Life Expectancy line visual has the following settings:
  - X Axis shelf: year, sorted in chronological order.
  - Y Axis shelf: life_expectancy, as avg(life_expectancy).
  - Colors shelf: country.
  - Under Settings, select the Parameters menu, and make sure that the, Receive parameters with explicit scope option is not enabled.
- The Life Expectancy and GDP line visual has the following settings:
  - X Axis shelf: year, sorted in chronological order.
  - Y Axis shelf: life_expectancy, as avg(life_expectancy).
  - Colors shelf: gdp_per_capita, as avg(gdp_per_capita); select the field on the shelf to view the Field Properties menu, expand Axis, and select Secondary Axis.
  - Under Settings, select the Parameters menu, and make sure that the, Receive parameters with explicit scope option is not enabled.

Prepare source visuals

In this example, a bar visual World Population is used as the source (controlling) visual.

Procedure

- The visual has the following settings:
  - X Axis shelf: un_region. You can also choose to use a dimensional hierarchy that represents the [un_region | un_subregion | country] tuple.
  - YAxis shelf: population, as sum(population); alias it as Population.
  - Filters shelf: year, set to 2010
  - Under Settings, select the Parameters menu, and enable the Send parameters with dataset scope option. Do not select other options in this menu.

Prepare and use the dashboard

In this example, a new dashboard is created that displays both the source and the target visuals and filters are also used.

Procedure

1. Create a new dashboard, and add the three visuals to it.

   In this example, the source visual is resized so it occupies the left side of the dashboard, while the target visuals share the right side of the dashboard.

2. Switch to View mode.

3. Click one of the bars of the source visual (in this example, Oceania), and select the Filter All option.

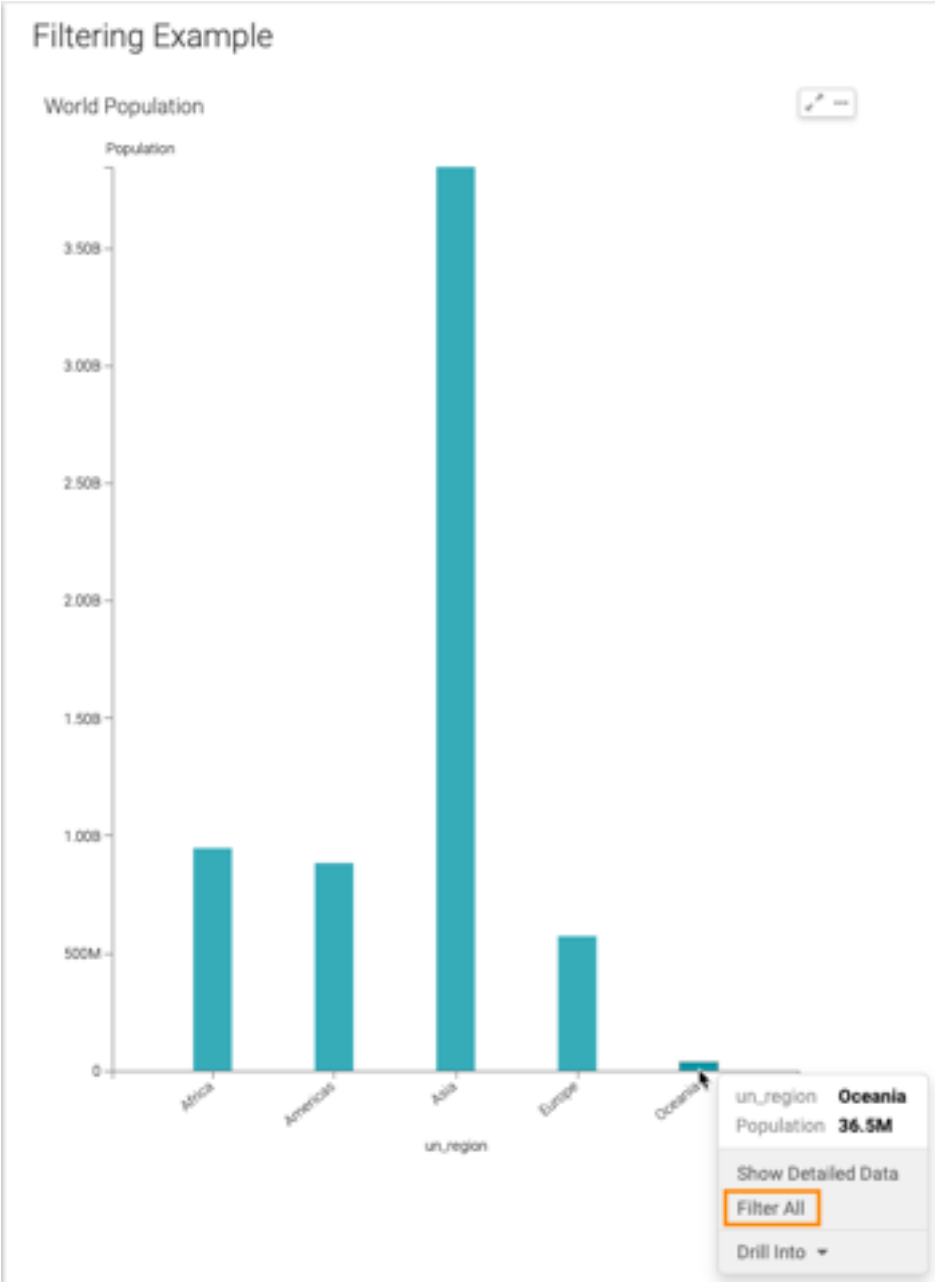

Cloudera Data Visualization refreshes the dashboard to show the target visuals that contain information about the selected un_region only. You can see the value of the filters on each visual when you hover the pointer over them, or the value of the dashboard parameters when you hover over the (filter) icon.

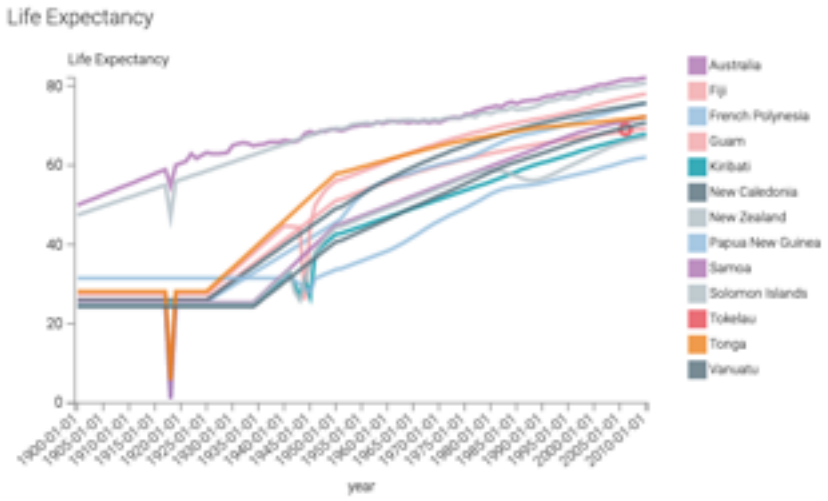

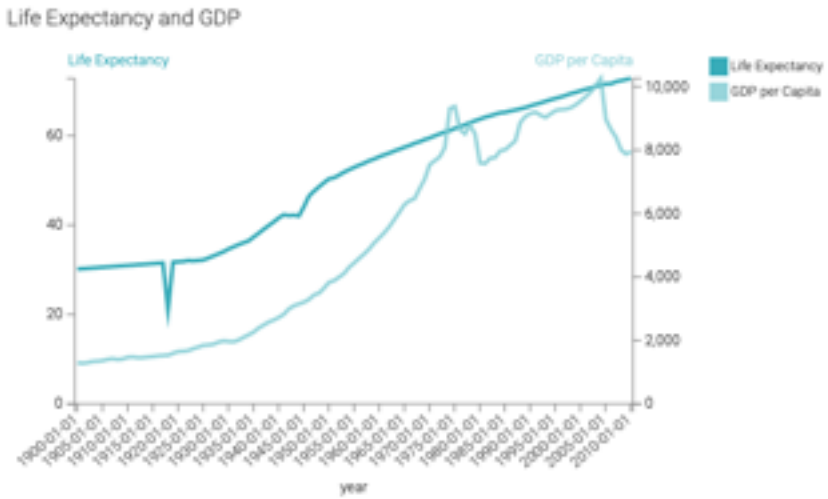

Note: Any visual can be both a source and a target when passing filter parameters.

4. To examine data outliers in one of the 'target' visuals, click on that element. In this example, we can check for more details in the Life Expectancy visual by investigating the drop reported for Turkey in 1915.

5. Click that part of the line chart, and select Filter All.

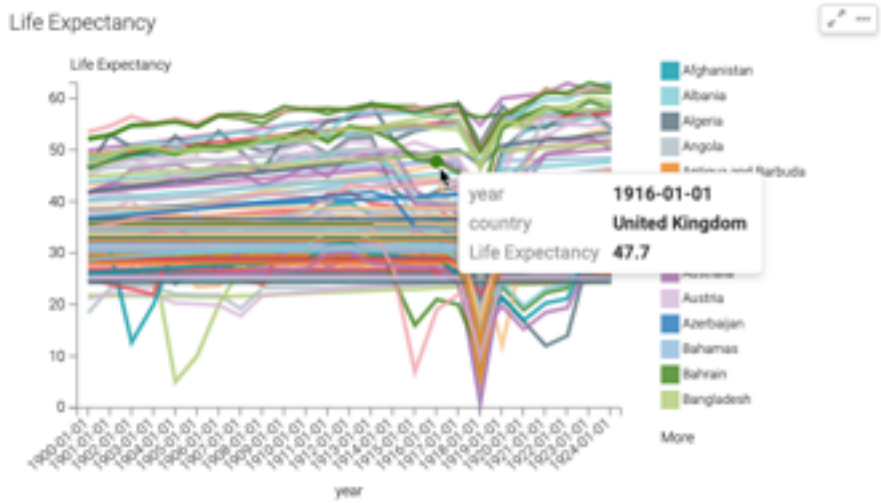

Cloudera Data Visualization refreshes the dashboard to show the effect on the other visuals. For example, the Life Expectancy and GDP shows that the GDP per Capita in Turkey for 1915 was 1.52k.

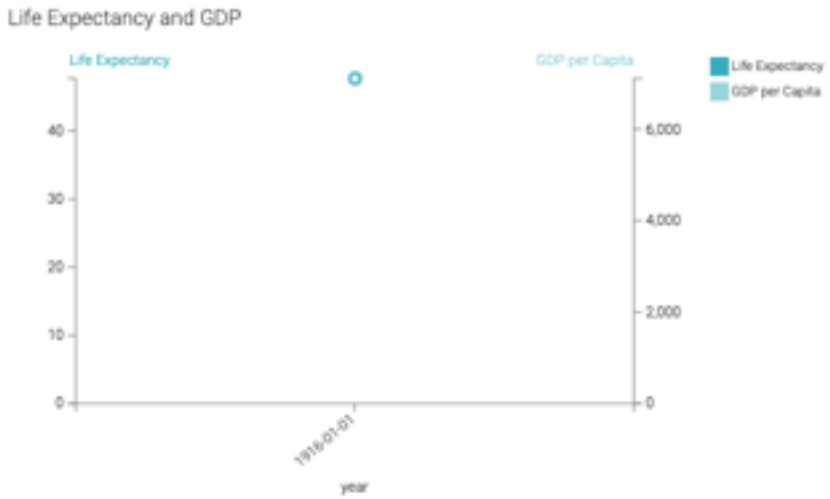

Discovering detailed data behind a visual

You can click on a part of a visual and see the matching rows of data, in detail.

About this task

The following steps describe how to access detailed data on a sample visual through. This visual shows the number of social media touch points, on a quarterly basis. While Twitter and Tumblr elements seem relatively constant, Facebook numbers for Q1 are low relative to other quarters. in the example, you can examine the underlying data.

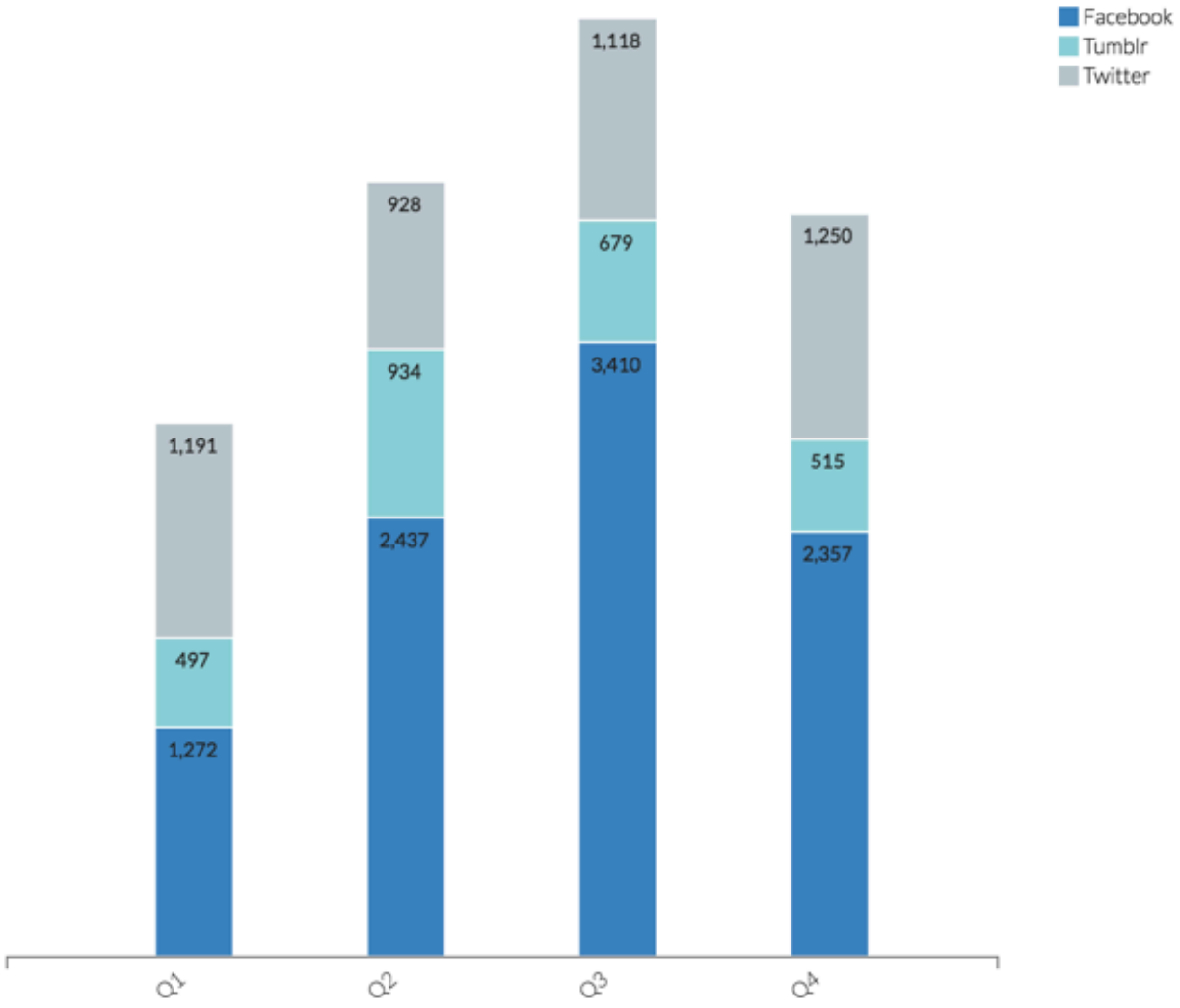

Before you begin

To use this option, you must first enable it, as described in Showing detail data button in context menu.

Procedure

1. Hover over the Facebook portion of the Q1 bar, and examine the totals that appear on the tooltip.

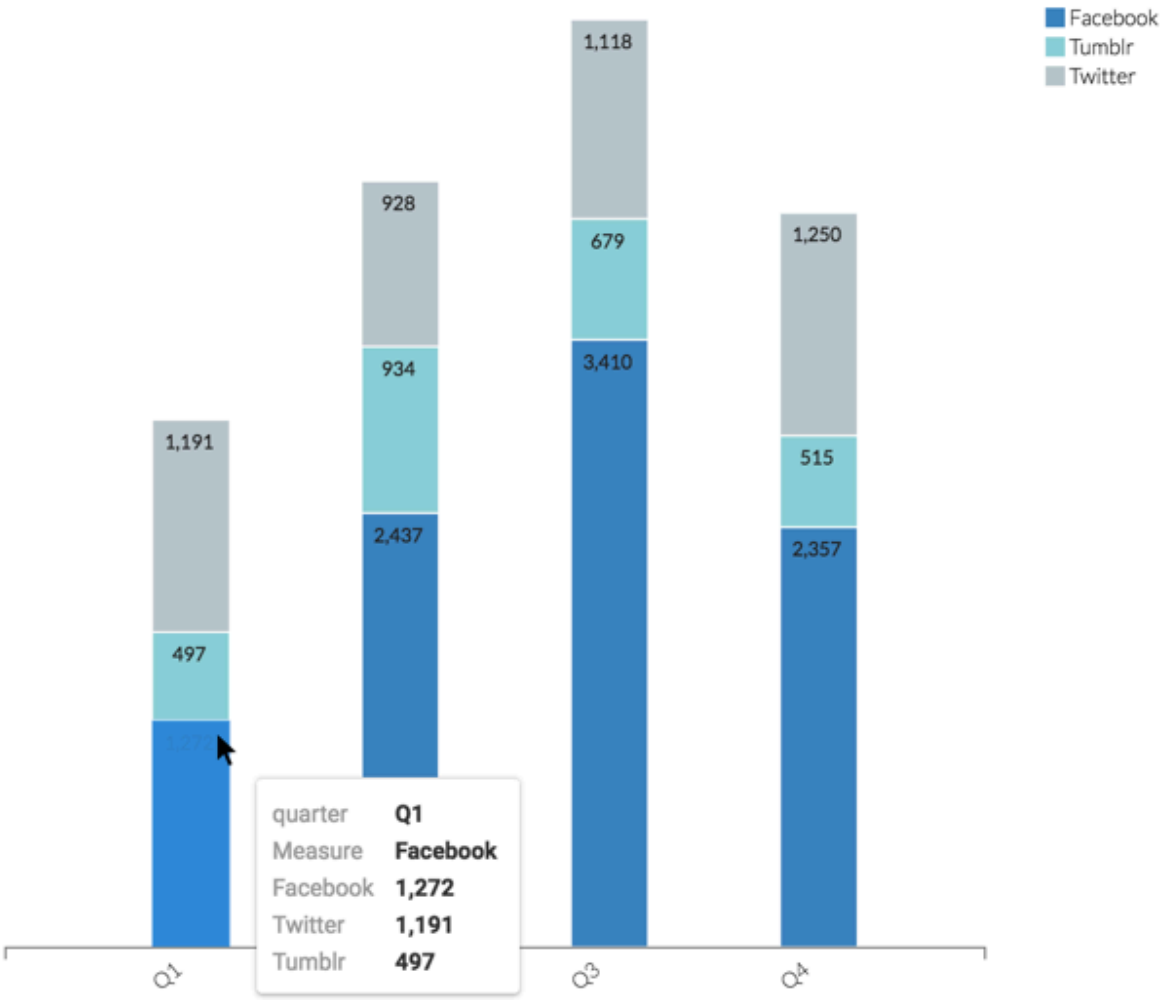

2. Click the element to see the context menu, and select Show Detailed Data option.

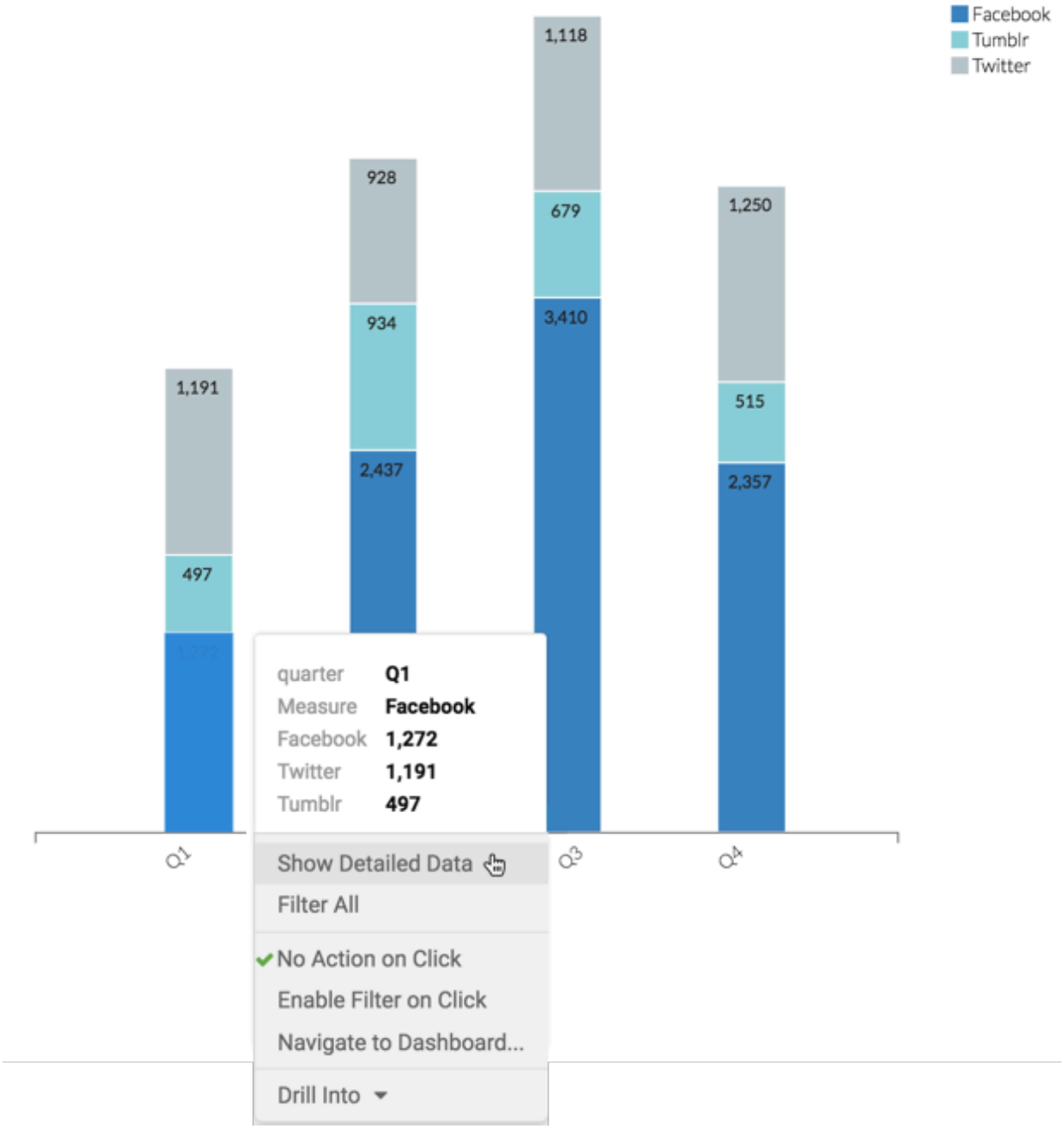

The Detail Data modal window appears. It has two tabs:

- The Data tab shows the raw data that makes up the clicked visual element, as well as all applicable filters.
- The Queries tab shows the SQL queries used to create the visual.

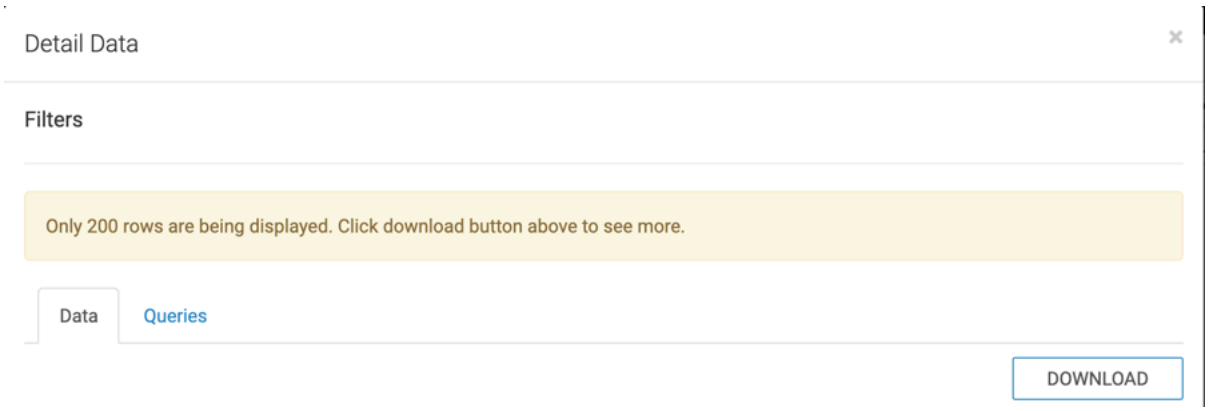

3. Click Download to save the data as a *.csv file locally.

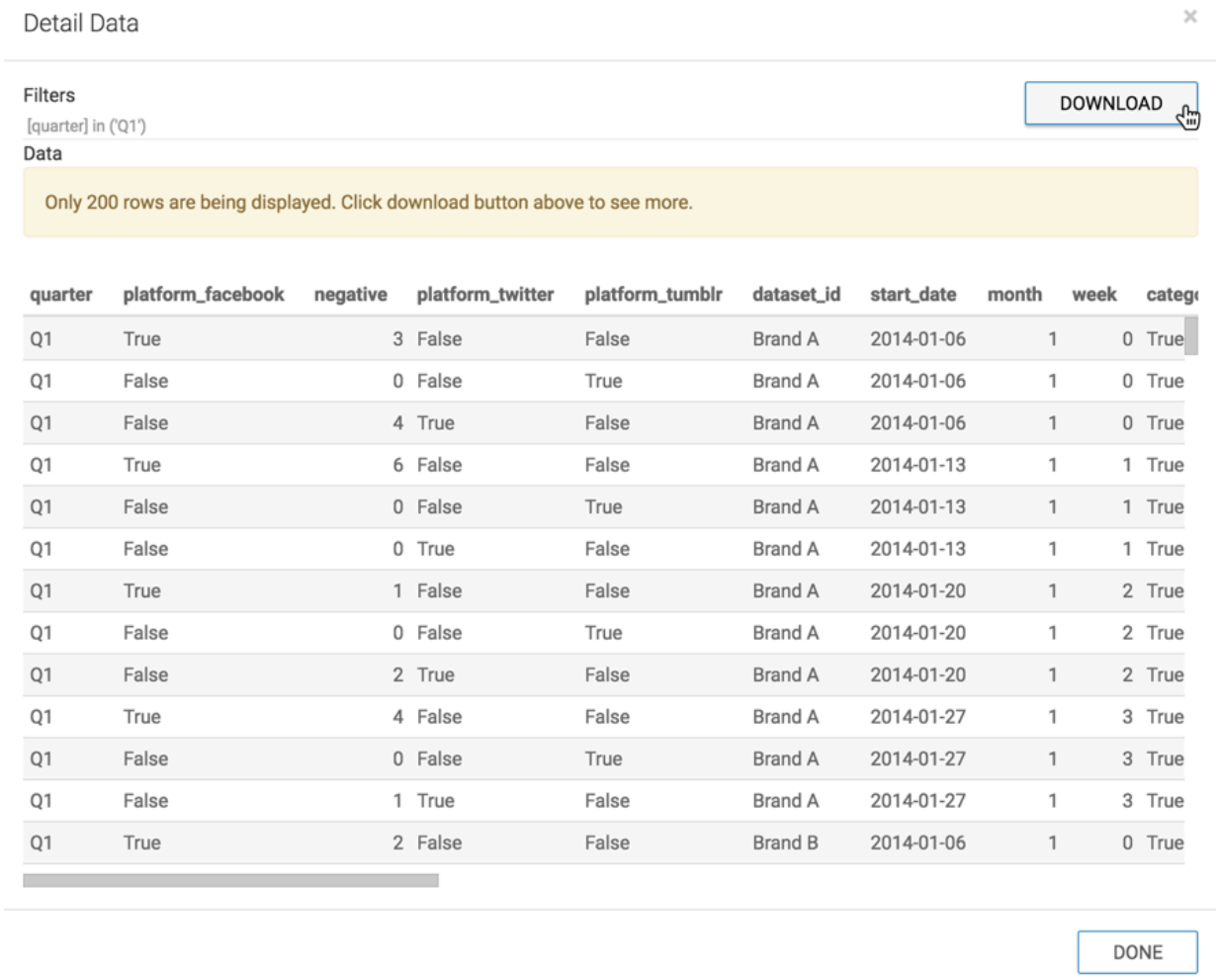

quarter	platform_facebook	negative	platform_twitter	platform_tumblr	dataset_id	start_date	month	week	catego
Q1	True	3	False	False	Brand A	2014-01-06	1	0	True
Q1	False	0	False	True	Brand A	2014-01-06	1	0	True
Q1	False	4	True	False	Brand A	2014-01-06	1	0	True
Q1	True	6	False	False	Brand A	2014-01-13	1	1	True
Q1	False	0	False	True	Brand A	2014-01-13	1	1	True
Q1	False	0	True	False	Brand A	2014-01-13	1	1	True
Q1	True	1	False	False	Brand A	2014-01-20	1	2	True
Q1	False	0	False	True	Brand A	2014-01-20	1	2	True
Q1	False	2	True	False	Brand A	2014-01-20	1	2	True
Q1	True	4	False	False	Brand A	2014-01-27	1	3	True
Q1	False	0	False	True	Brand A	2014-01-27	1	3	True
Q1	False	1	True	False	Brand A	2014-01-27	1	3	True
Q1	True	2	False	False	Brand B	2014-01-06	1	0	True

4. After examining the data, click Done to close the modal window.

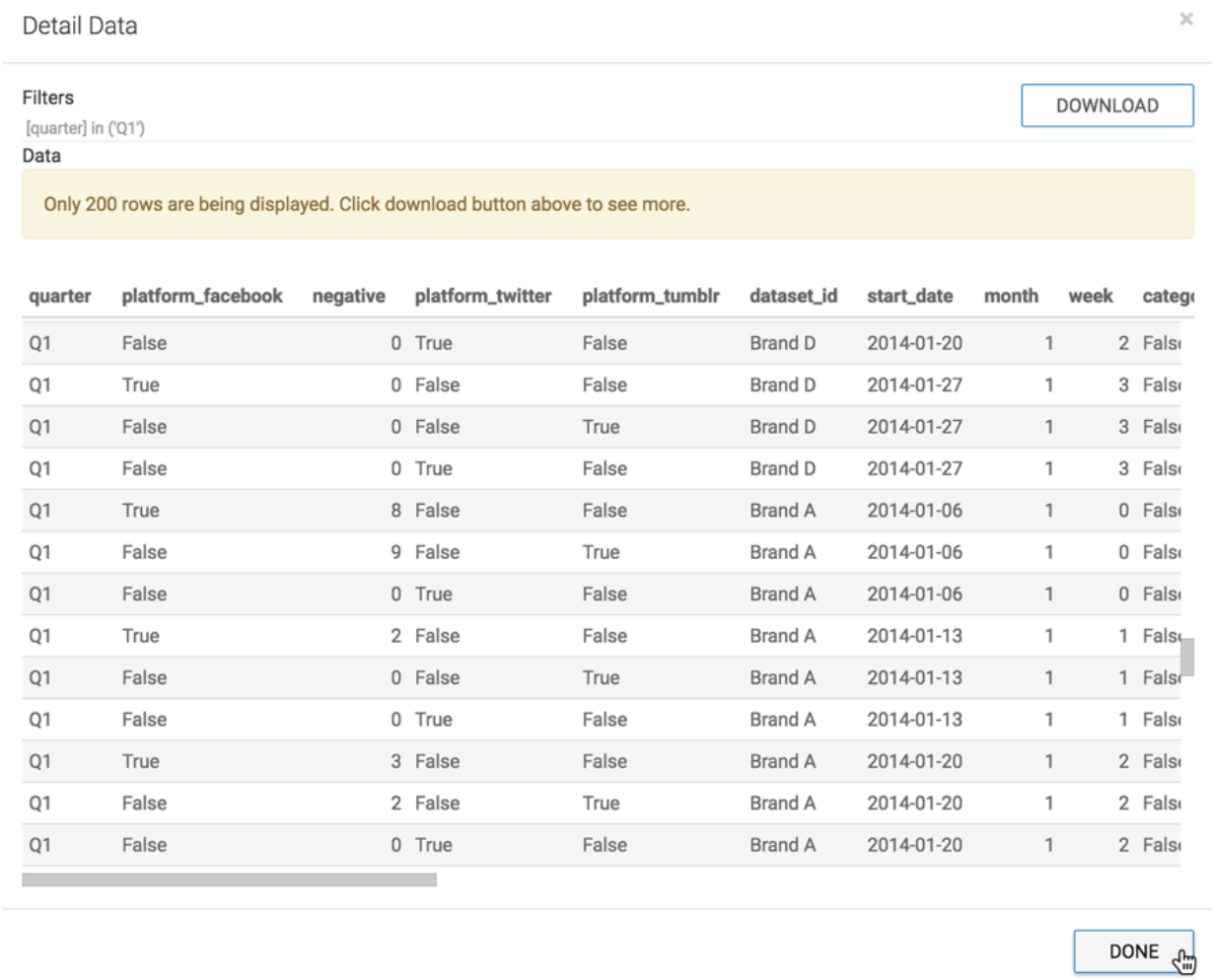

quarter	platform_facebook	negative	platform_twitter	platform_tumblr	dataset_id	start_date	month	week	catego
Q1	False	0	True	False	Brand D	2014-01-20	1	2	Falsı
Q1	True	0	False	False	Brand D	2014-01-27	1	3	Falsı
Q1	False	0	False	True	Brand D	2014-01-27	1	3	Falsı
Q1	False	0	True	False	Brand D	2014-01-27	1	3	Falsı
Q1	True	8	False	False	Brand A	2014-01-06	1	0	Falsı
Q1	False	9	False	True	Brand A	2014-01-06	1	0	Falsı
Q1	False	0	True	False	Brand A	2014-01-06	1	0	Falsı
Q1	True	2	False	False	Brand A	2014-01-13	1	1	Falsı
Q1	False	0	False	True	Brand A	2014-01-13	1	1	Falsı
Q1	False	0	True	False	Brand A	2014-01-13	1	1	Falsı
Q1	True	3	False	False	Brand A	2014-01-20	1	2	Falsı
Q1	False	2	False	True	Brand A	2014-01-20	1	2	Falsı
Q1	False	0	True	False	Brand A	2014-01-20	1	2	Falsı

Filter data using the include option in the context menu

You can click on a mark of a visual and include selected values in the filter shelf of the visual designer.

About this task

The Include option is enabled by default in all visual types except Table and Queries visual types.

Follow these steps to include selected values in a visual.

Procedure

1. For Table and Queries visual types, in Edit mode, navigate to the Marks menu, and select the Show Context Menu.

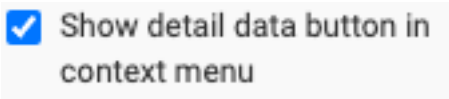

2. Click any mark of the visual to open the tooltip and the context menu.

3. In the context menu that appears, click Include. Here, we are selecting the line for Africa, in year 1985.

   These values are added in the filter shelf of the visual designer.

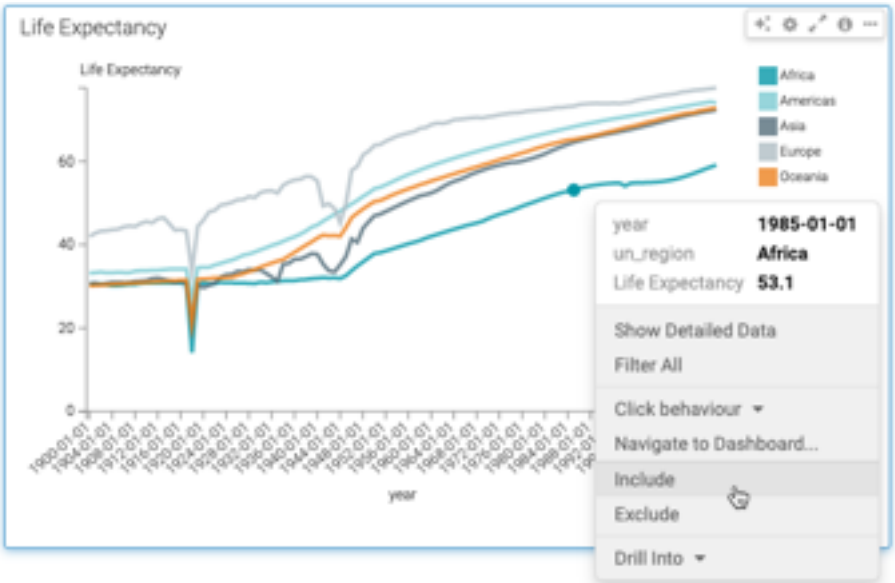

4. Click REFRESH VISUAL.

   After applying the Include filter, the following image only shows the average life expectancy in Africa in year 1985.

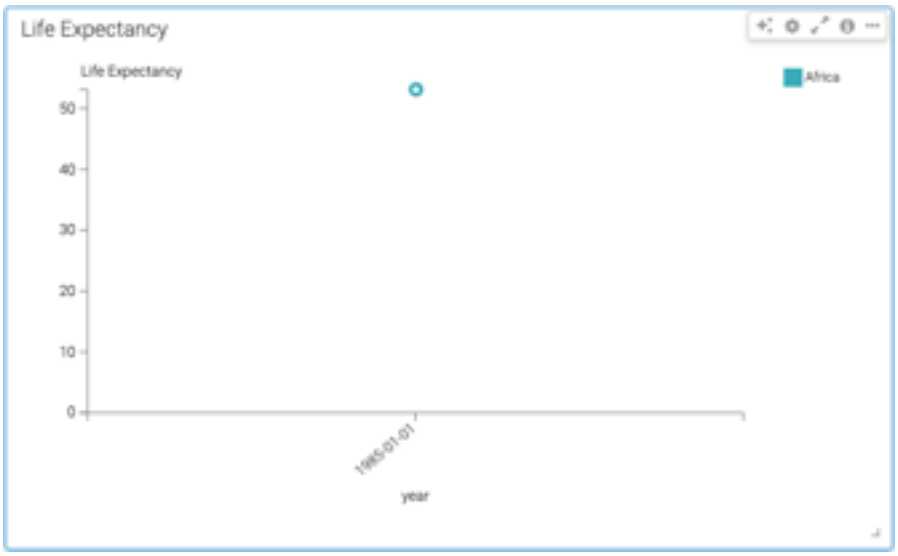

Filter data using the exclude option in the context menu

You can click on a mark of a visual and add an expression in the filter shelf of the visual designer to exclude selected values in a visual.

About this task

The Exclude option is enabled by default in all visual types except Table and Queries visual types.

Follow these steps to exclude selected values in a visual.

Procedure

1. For Table and Queries visual types, in Edit mode, navigate to the Marks menu and select the Show Context Menu.

   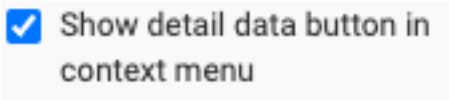

2. Click any mark of the visual to open the tooltip and the context menu.
3. In the context menu that appears, click Exclude.

   The selected values appear in the filter shelf of the visual, excluding year not in (1900) and un_region not in ('Europe').

4. After applying the exclude filter, notice that the visual removes the information for year 1900 and region Europe.

   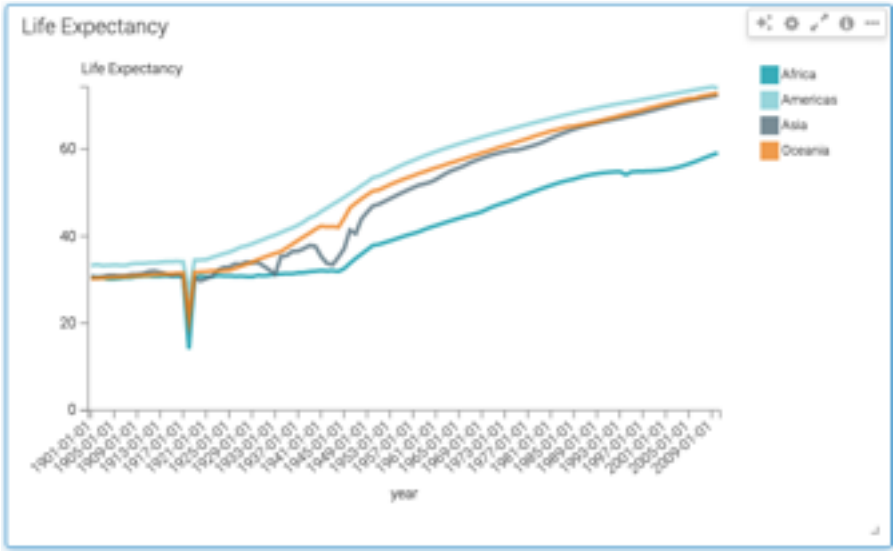

Filtering data using legends

With large datasets, a visual may become very crowded, making it hard to understand what graphical elements represent. In Cloudera Data Visualization, you can easily filter on a particular element in the legend.

About this task

Cloudera Data Visualization legends are interactive. In View mode, you can easily find an element, by either hovering over an element, or by clicking to filter on a particular element.

In dashboard Edit mode, you can filter on a particular element in the legend. You can also test the behavior of the filter before viewing the filter results in View mode.

The following procedure shows you how to filter a visual from a legend.

Procedure

1. Open an existing dashboard in Edit mode.

   In this example, the Cereal Manufacture Code dashboard with the Enable Filter on Click visual is used.
2. Make sure that the visual displays the legend.
3. Switch to the Filters tab and create a custom filter in the filter area of the application.

   In the below example the manufacture code filter has been added. The manufacture code is currently filtering on the letter K.
4. Hover over option G in the legend and right click to open a dropdown menu.

   Enable Filter on Click is selected by default.

5. Click Test Click Behavior to verify if click behaviour is working on the filter.

   The manufacture code filter changes to letter G.

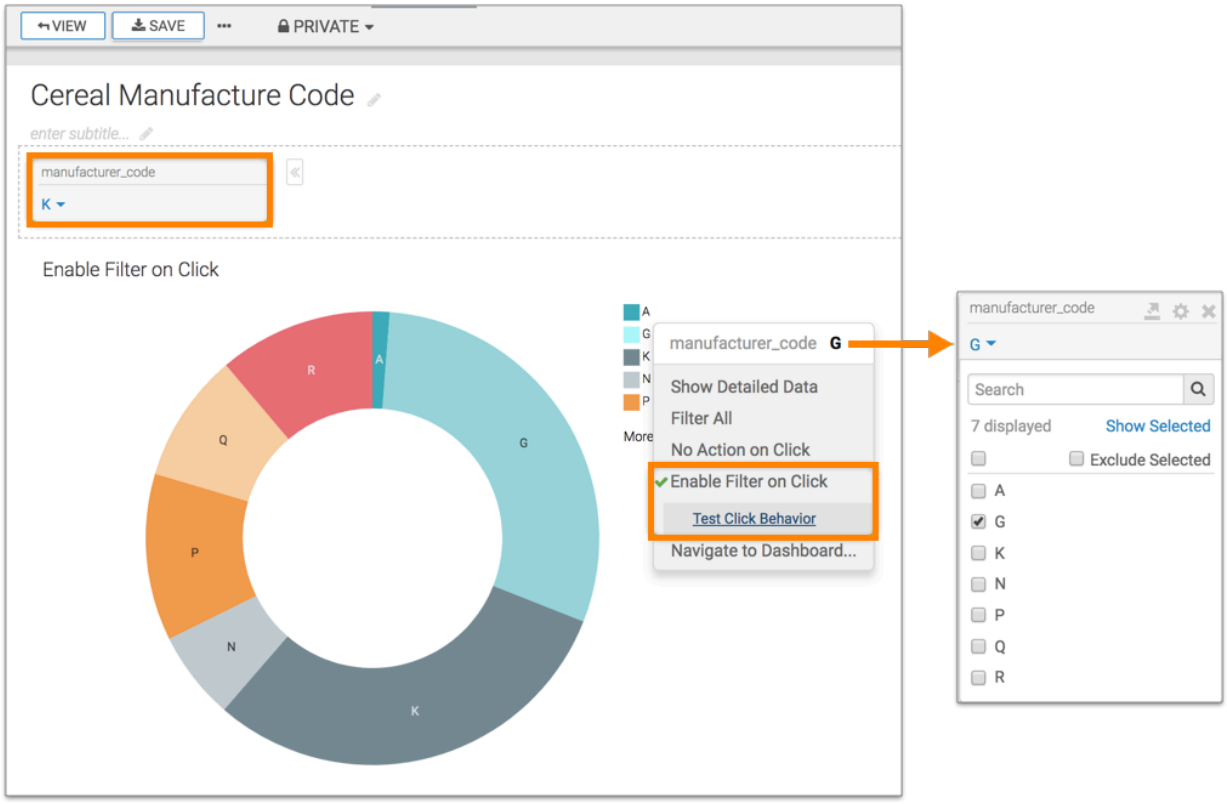

6. Click VIEW to see Enable Filter on Click behavior on one of the legend options.

   The visual is now filtered on manufacture code G.

Cloudera Data Visualization 7.2.9

Customizing Applications

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Changing application style....4
Customizing menu bar style for applications....4
Displaying menu on left....5
Changing menu background color....6
Changing menu bottom bar color....8
Changing menu selected color....9
Customizing menu text for apps....10
Changing menu font family....11
Changing menu font size....13
Changing menu font weight....13
Changing menu text color....14
Changing menu text selected color....15
Changing menu text in uppercase....16
Customizing logo for apps....17
Changing logo image....18
Changing logo width....19
Changing logo height....19
Changing logo top margin....20
Changing logo left margin....20
Customizing app background....20
Changing app background color....21
Removing dashboard shadow....23

Adding custom styles to apps....26
Adding CSS classes to applications....28
Adding included styles to applications....29
Adding inline CSS code to apps....31

Enabling navigation from logo....31

Changing application style

The Style menu contains a large selection of options for customizing the appearance of apps. To use the Style menu, follow these steps.

Procedure

1. On the right side of App Designer, click the Style menu.

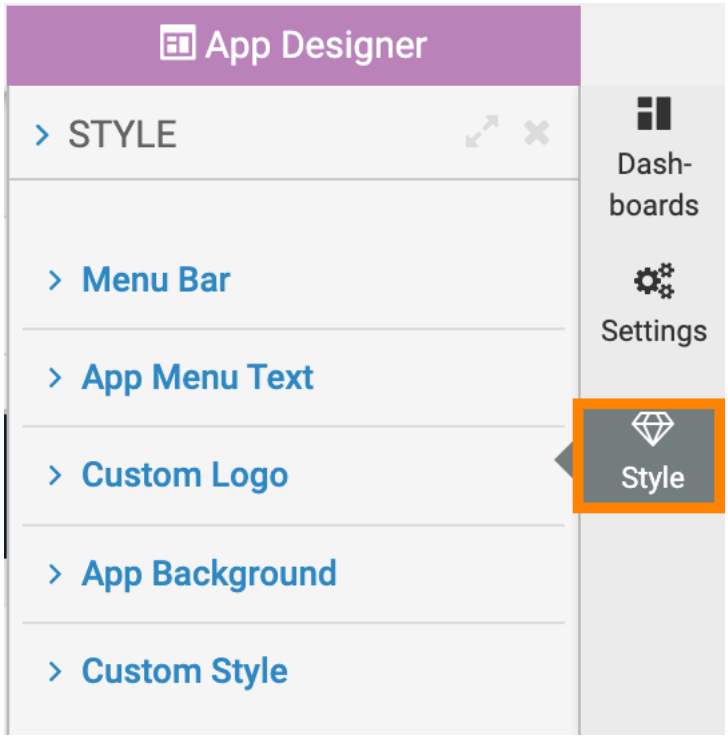

2. In the Style menu, review the additional menus and select the relevant option according to your needs.

Customizing menu bar style for applications

About this task

To get to the options in the Menu Bar style menu of the App Designer, follow these steps.

Procedure

1. On the right side of App Designer, click the Style menu.

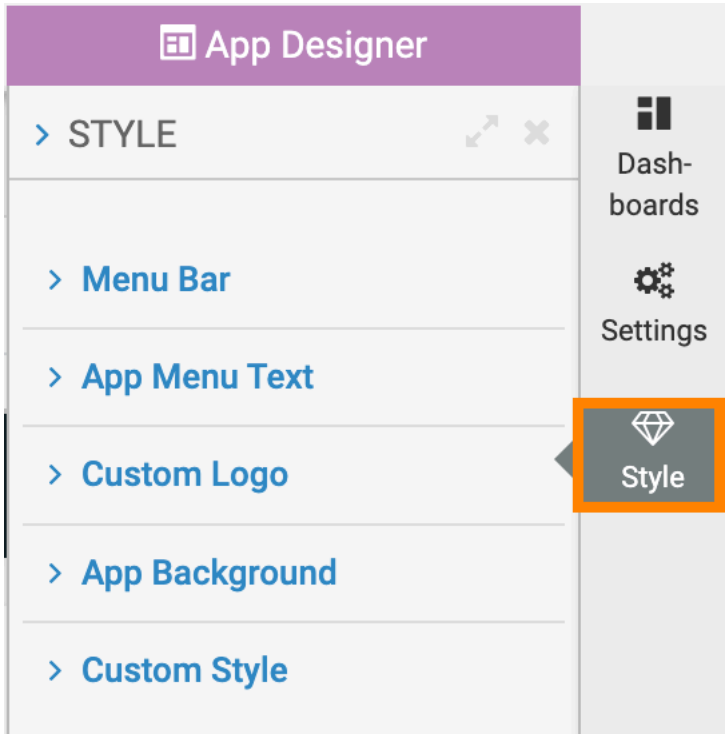

2. In the Style menu, click Menu Bar.

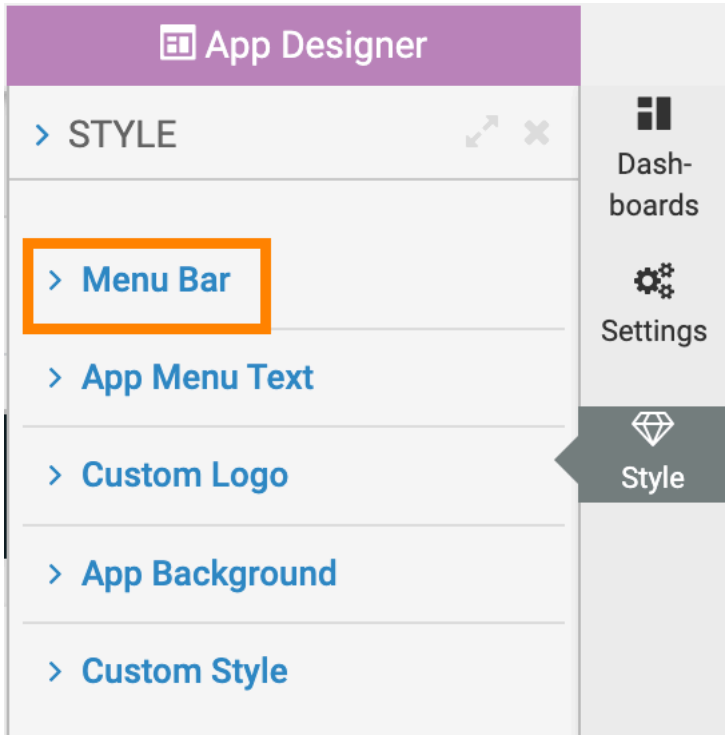

3. Select one of the following options.

Displaying menu on left

To move the app menu from the top of the app to the left, in App Designer interface, navigate to the Menu Bar menu of the Style menu, and select the option Display menu on left.

By default, this option is not enabled.

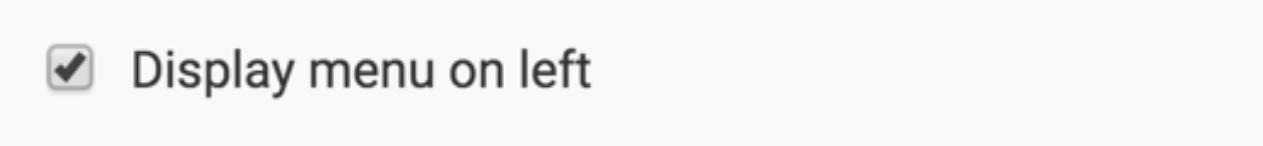

You can see how the menu is moved from the top to the left of the application:

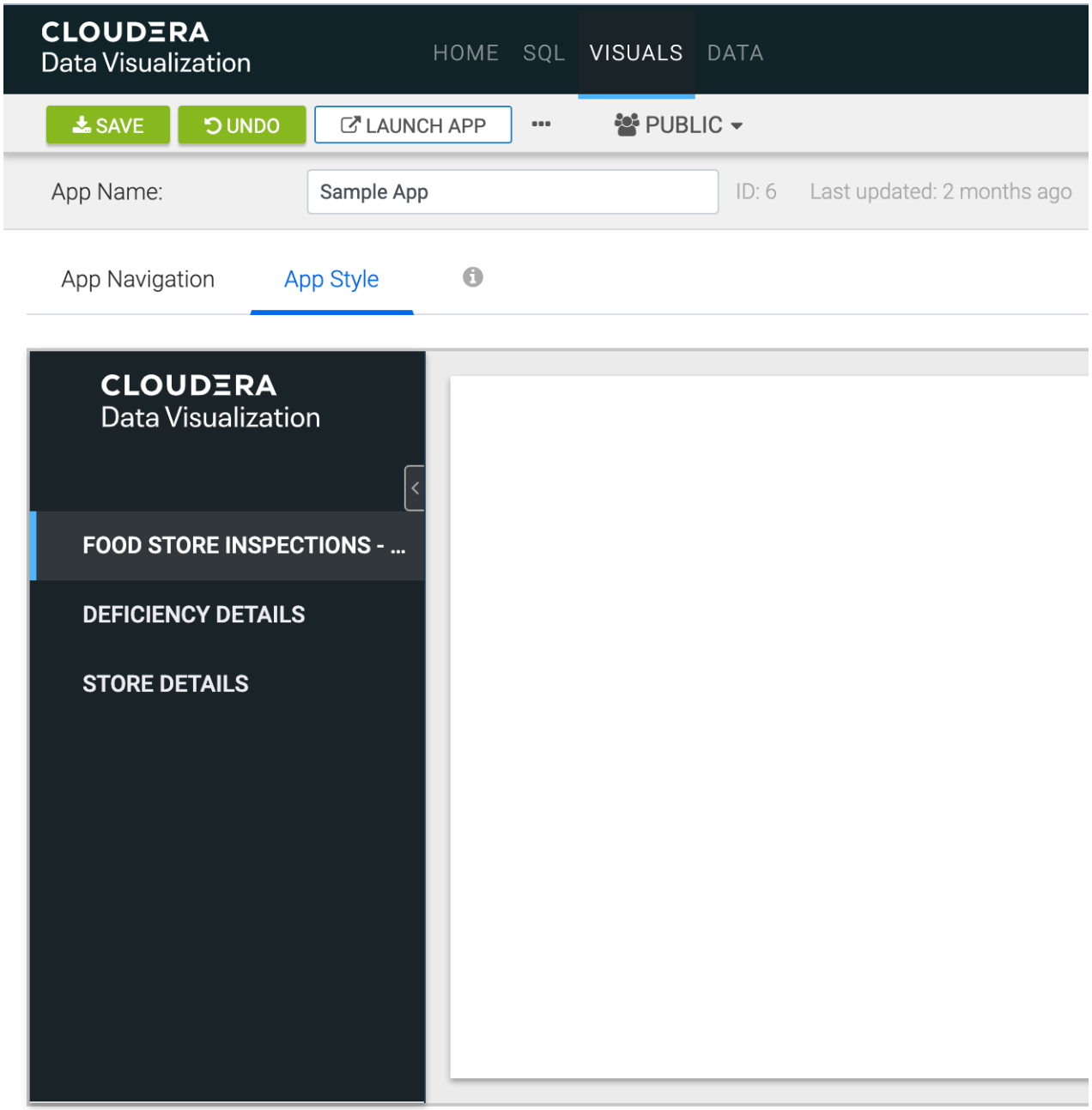

Changing menu background color

To change the background color of the app menus, in App Designer, navigate to the Menu Bar menu of the Style menu, and change the color in the Background Color selector.

You can select one of the palette colors, or specify a custom color.

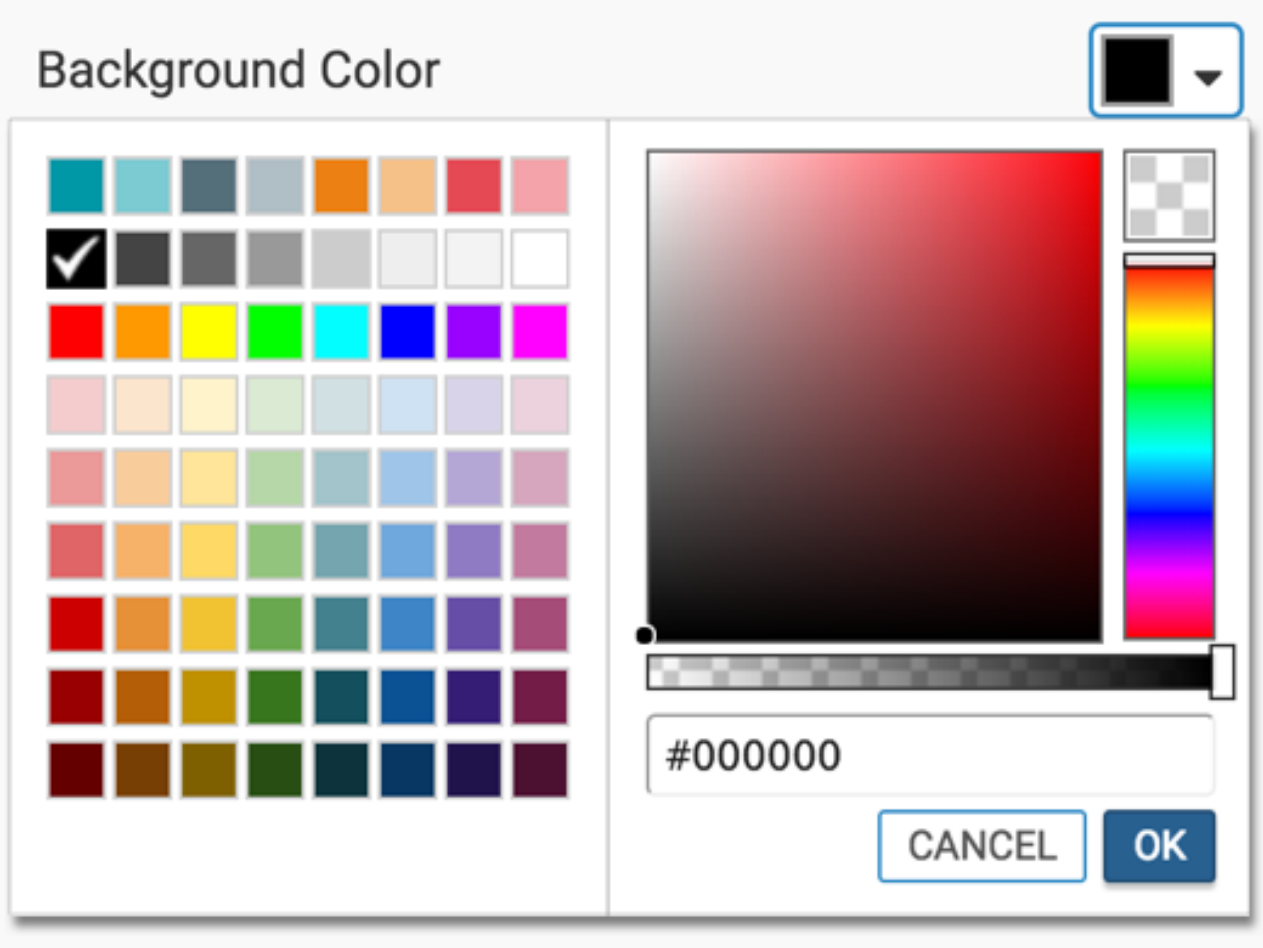

For example, you can see the effect of changing the default menu color (medium grey) to black:

Changing menu bottom bar color

To change the bottom bar color of the app menus, in App Designer, navigate to the Menu Bar menu of the Style menu, and change the color in the Bottom Bar Color selector.

You can select one of the palette colors, or specify a custom color.

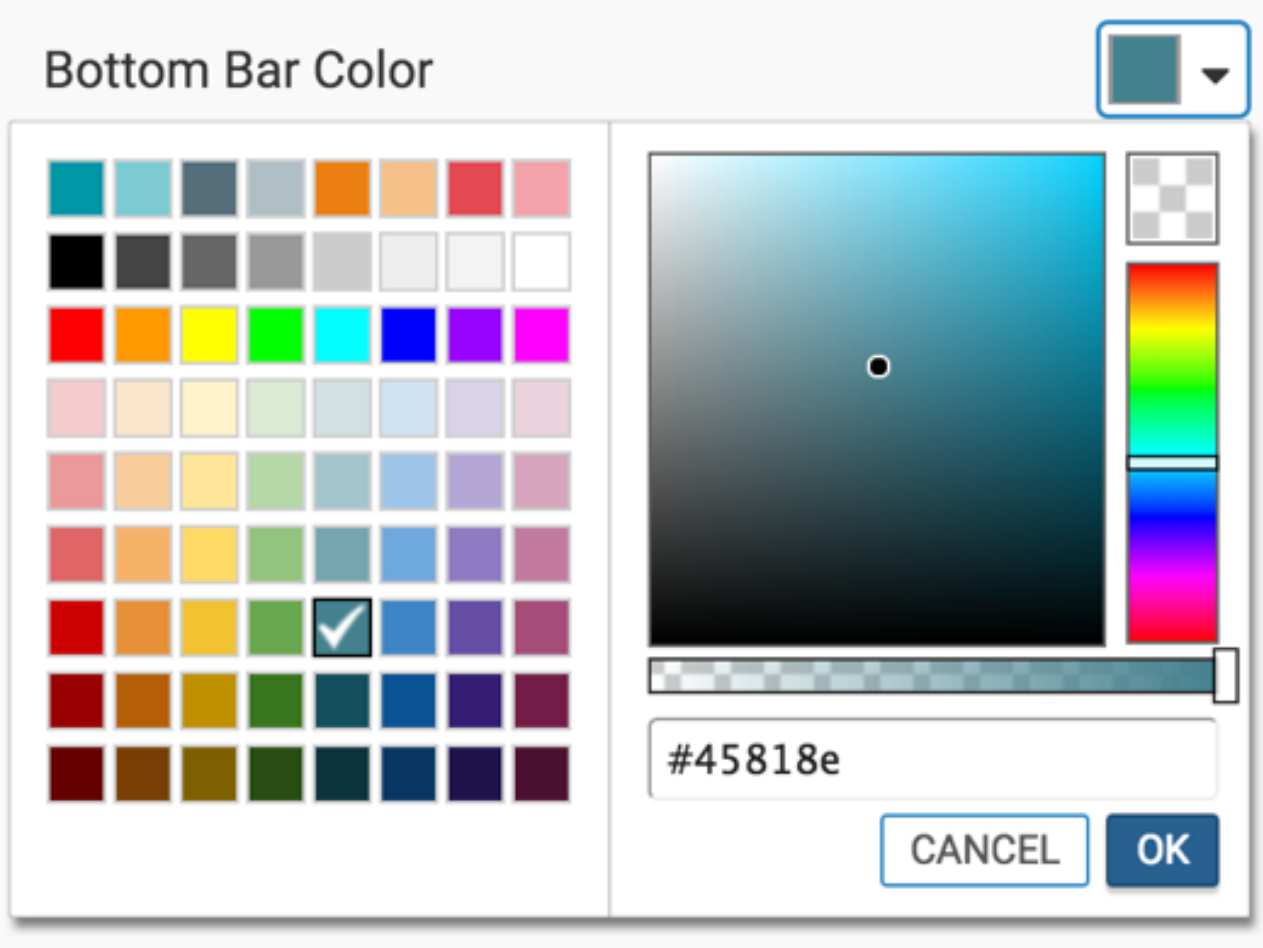

For example, you can see the effect of changing the default bottom bar color (clear) to teal:

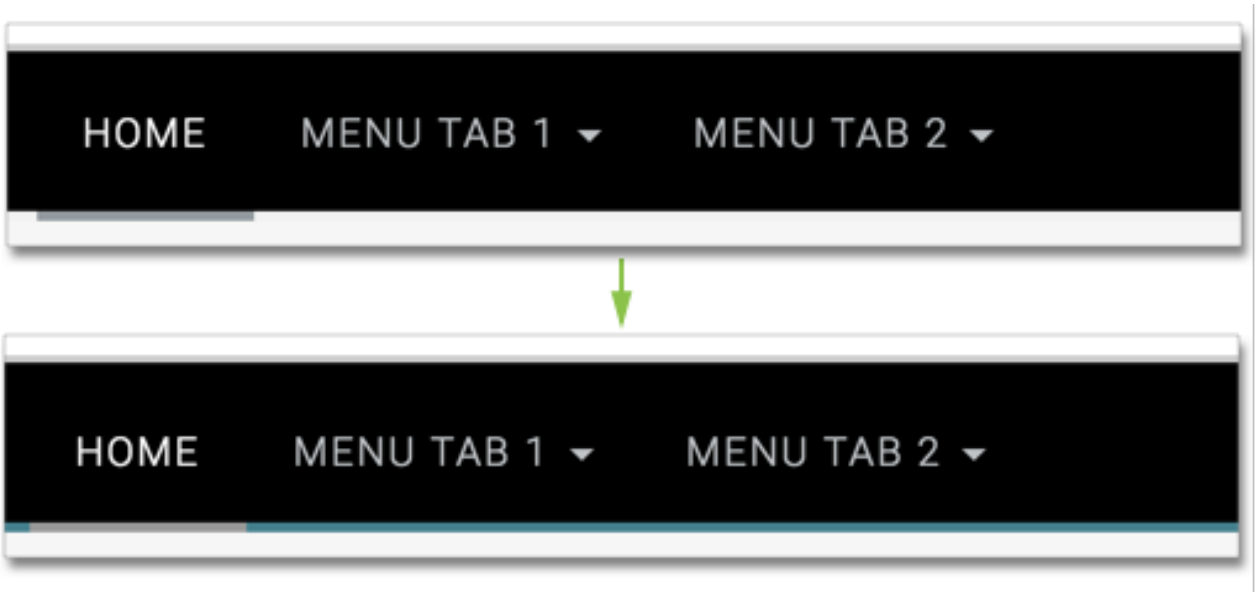

Changing menu selected color

To change the color of the selection indicator on the bottom bar of the app menus, in App Designer, navigate to the Menu Bar menu of the Style menu, and change the color in the Bar Color - Selected selector.

You can select one of the palette colors, or specify a custom color.

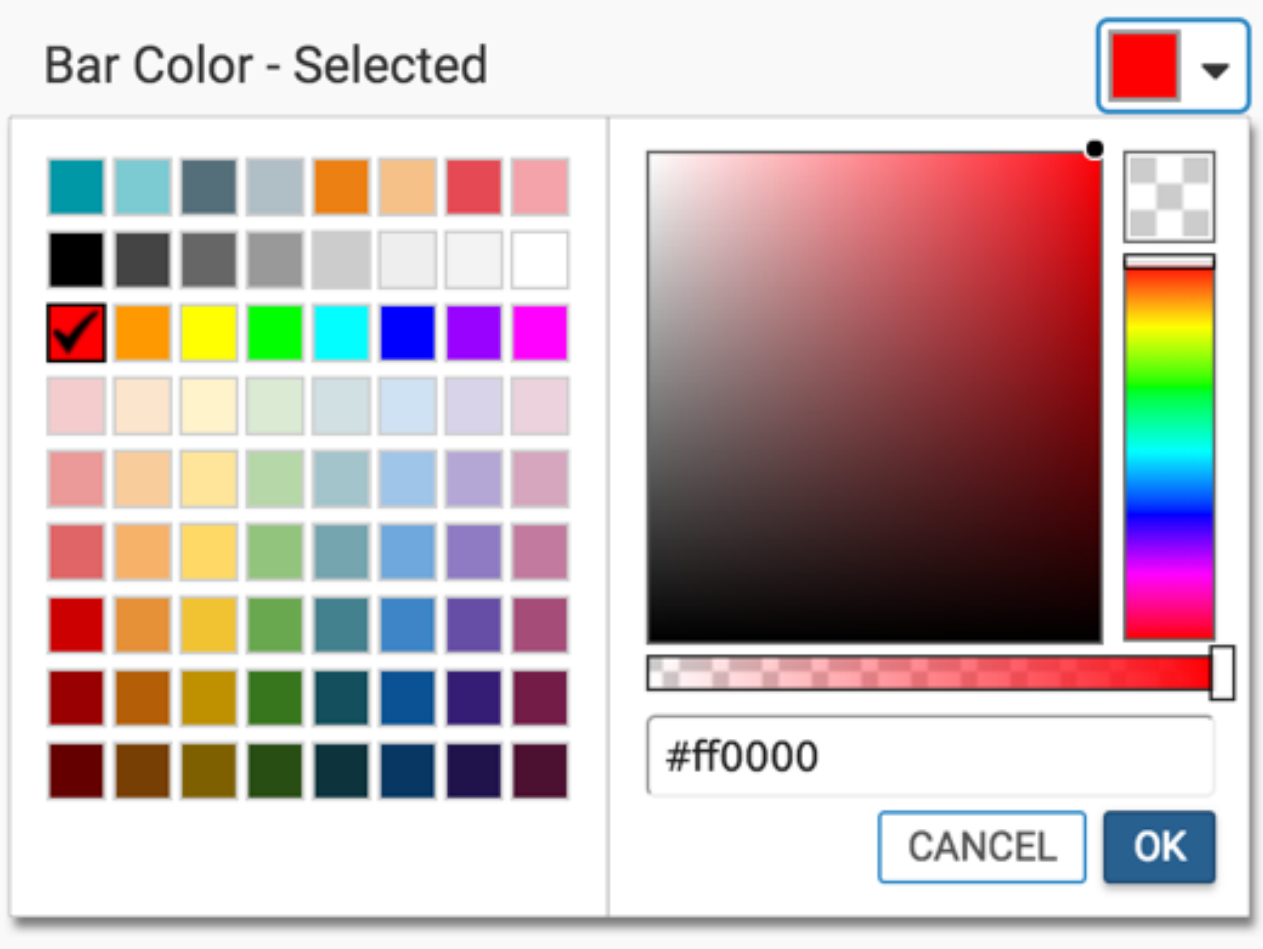

For example, you can see the effect of changing the default selection color (light grey) to red:

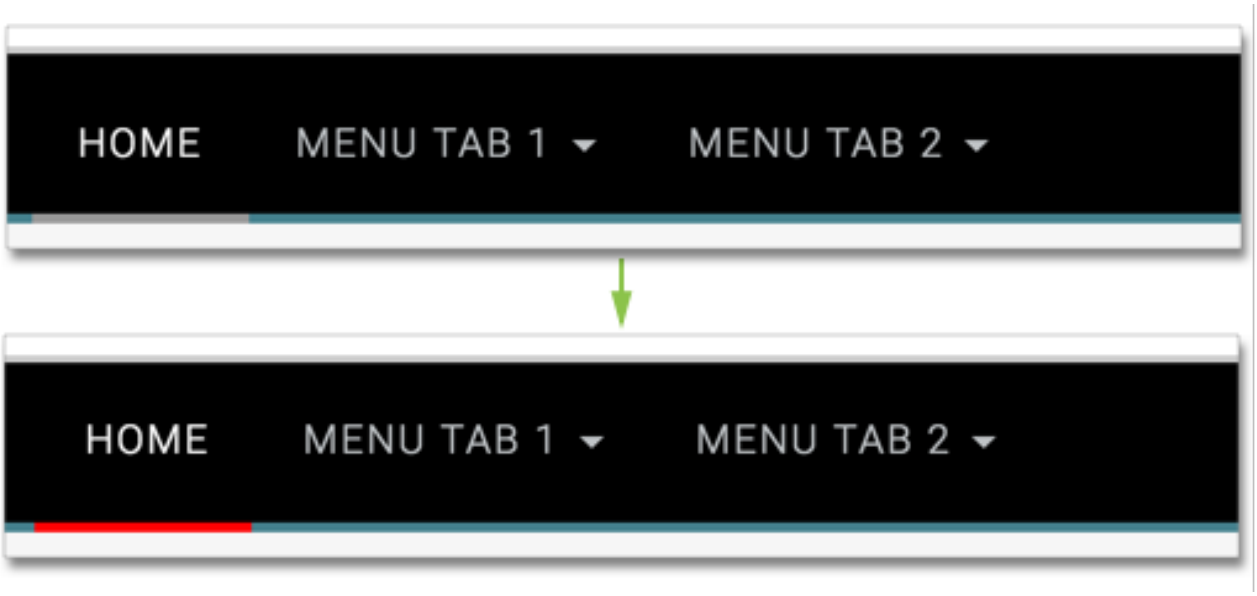

Customizing menu text for apps

About this task

To get to the options in the App Menu Text style menu of the App Designer, follow these steps.

Procedure

1. On the right side of App Designer, click the Style menu.

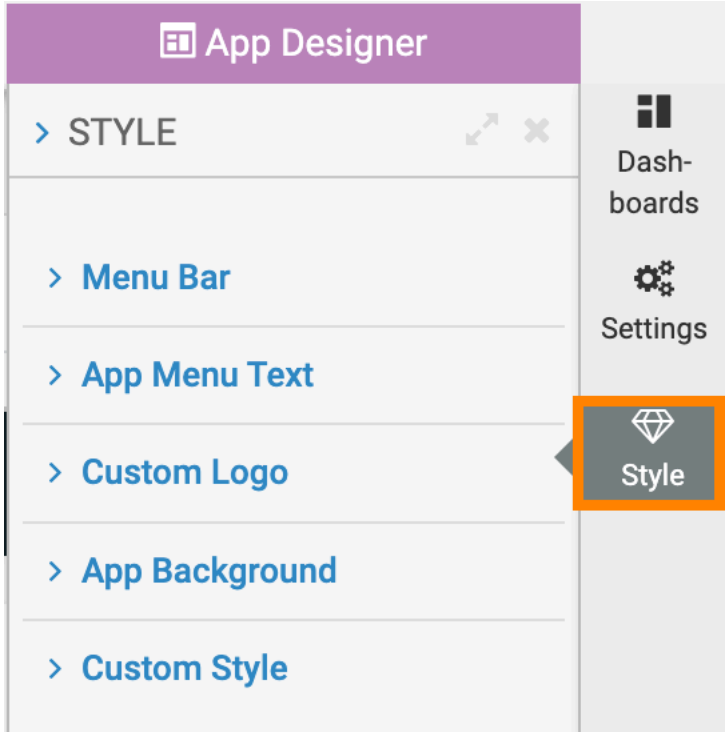

2. In the Style menu, click App Menu Text.

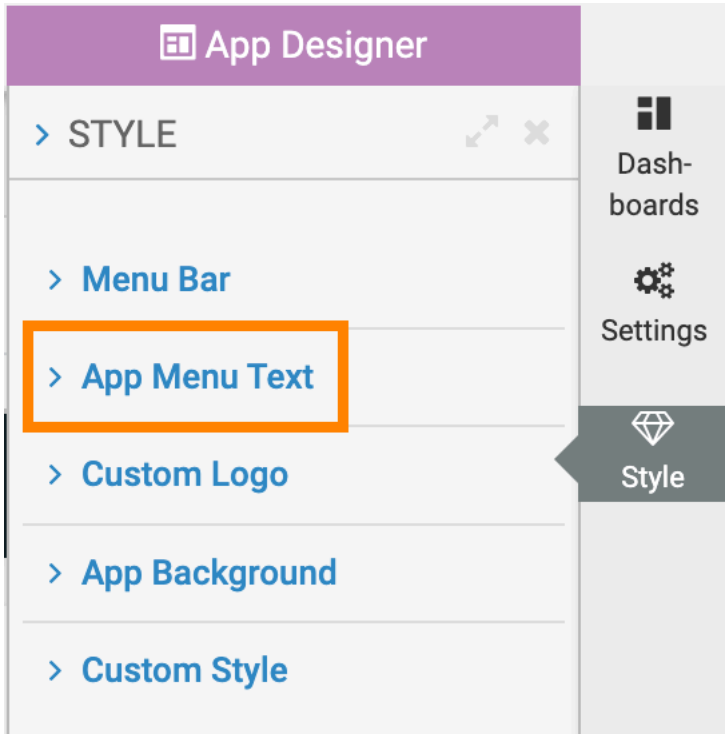

3. Select one of the following options.

Changing menu font family

To change the font of app menus in the App Designer, navigate to the App Menu Text menu of the Style menu, and select a new option from the Font Family menu.

The system specifies the default font size.

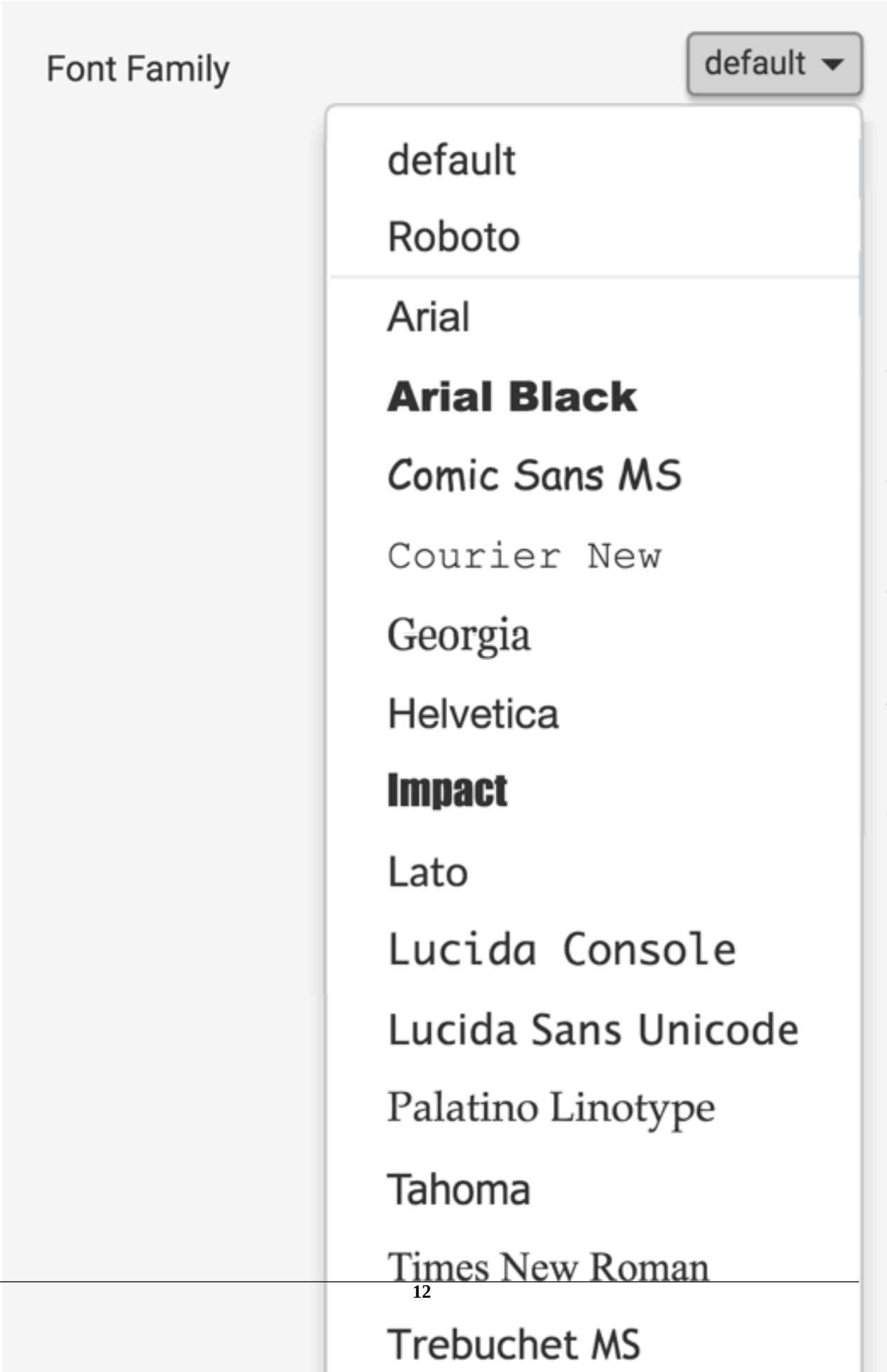
Font Family
default
default
Roboto
Arial
Arial Black
Comic Sans MS
Courier New
Georgia
Helvetica
Impact
Lato
Lucida Console
Lucida Sans Unicode
Palatino Linotype
Tahoma
Times New Roman
Trebuchet MS

You can see the effect of changing the font from the system default to Arial Black:

Changing menu font size

To change the size of the font in app menus in the App Designer, navigate to the App Menu Text menu of the Style menu, and change the selector for the Font Size option.

The system specifies the default font size.

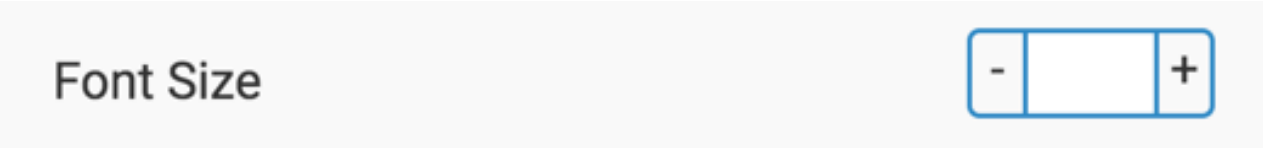

You can see the effect of changing the font size of the app menu from 12 to 20:

Changing menu font weight

To change the weight of the font in app menus in the App Designer, navigate to the App Menu Text menu of the Style menu, and change the selector for the Font Weight option.

Other than default, the options are Lighter, Normal, and Bold.

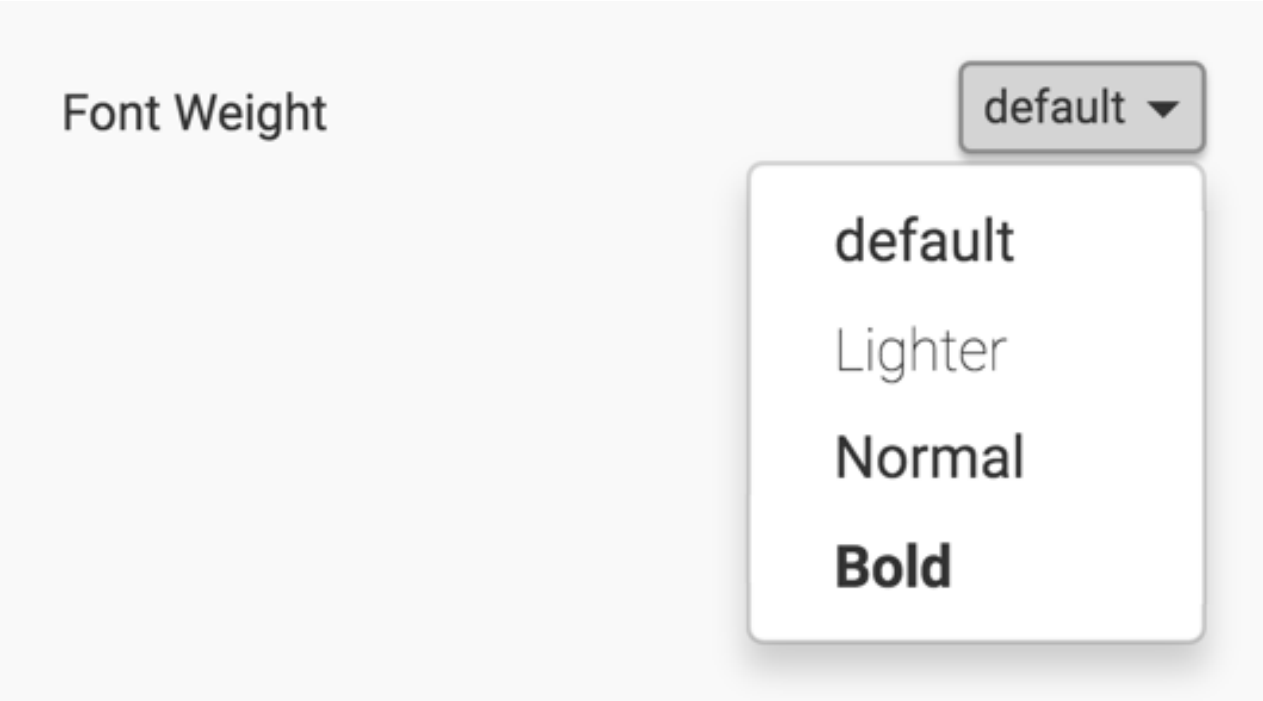

You can see the appearance of the app menu with all font weight options: default, light, normal, and bold:

Changing menu text color

To change the color of the font in app menus in the App Designer, navigate to the App Menu Text menu of the Style menu, and change the color in the Text Color selector.

You can select one of the palette colors, or specify a custom color.

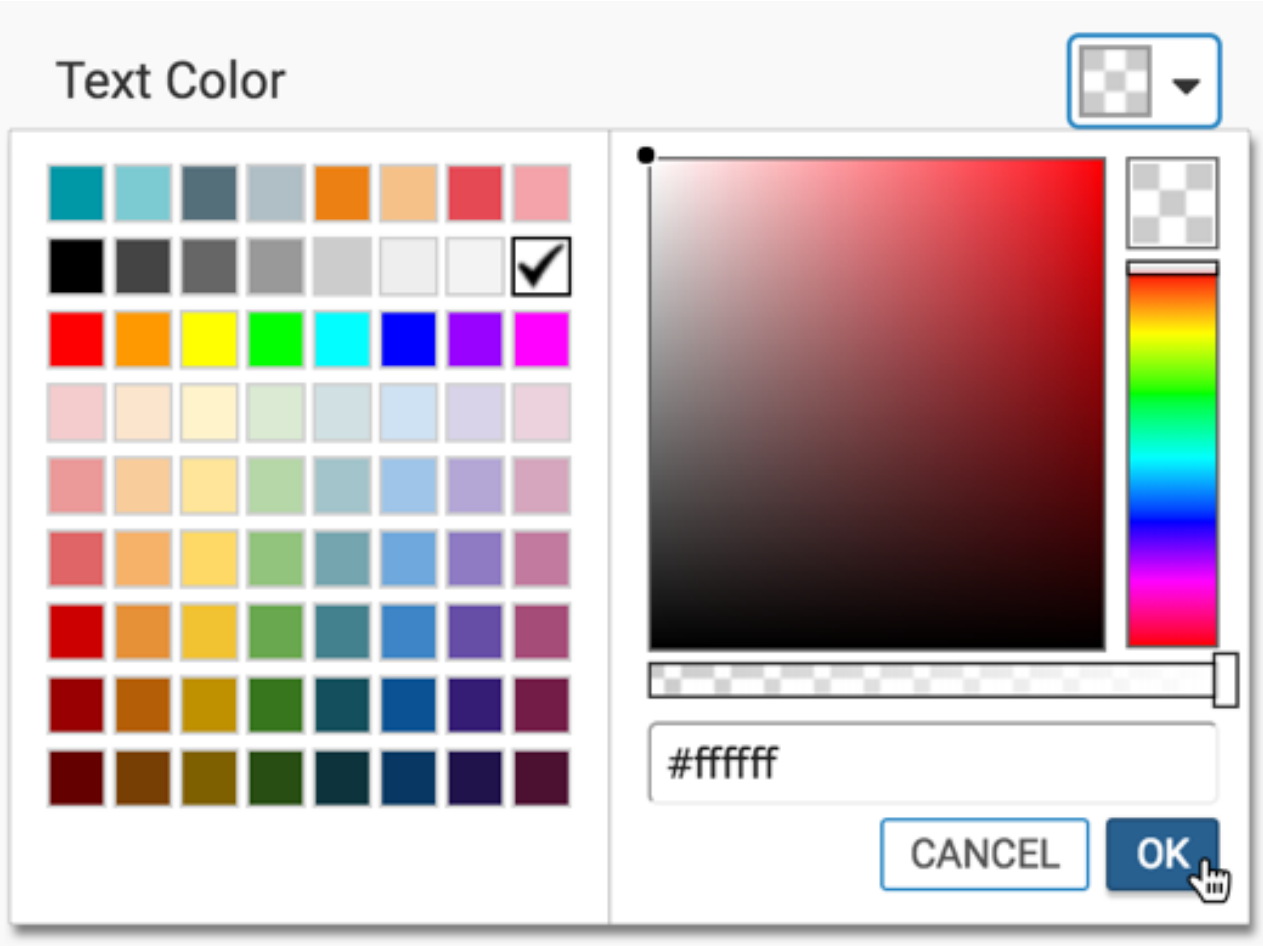

You can see the appearance of the app menu with the default text color (light grey) and with white text color:

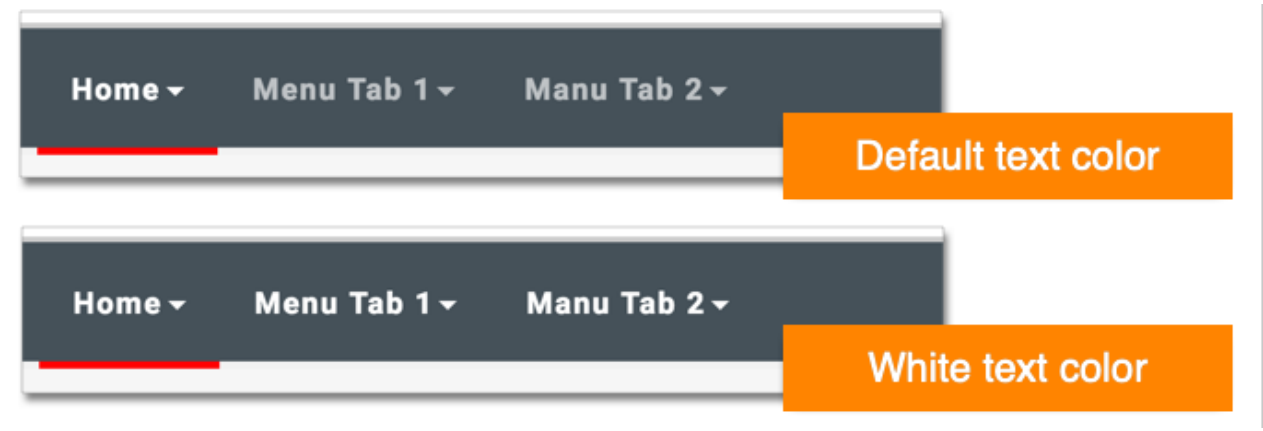

Changing menu text selected color

To change the color of the text for the selected item in app menus in the App Designer, navigate to the App Menu Text menu of the Style menu, and change the color in the Text Color - Selected selector.

You can select one of the palette colors, or specify a custom color.

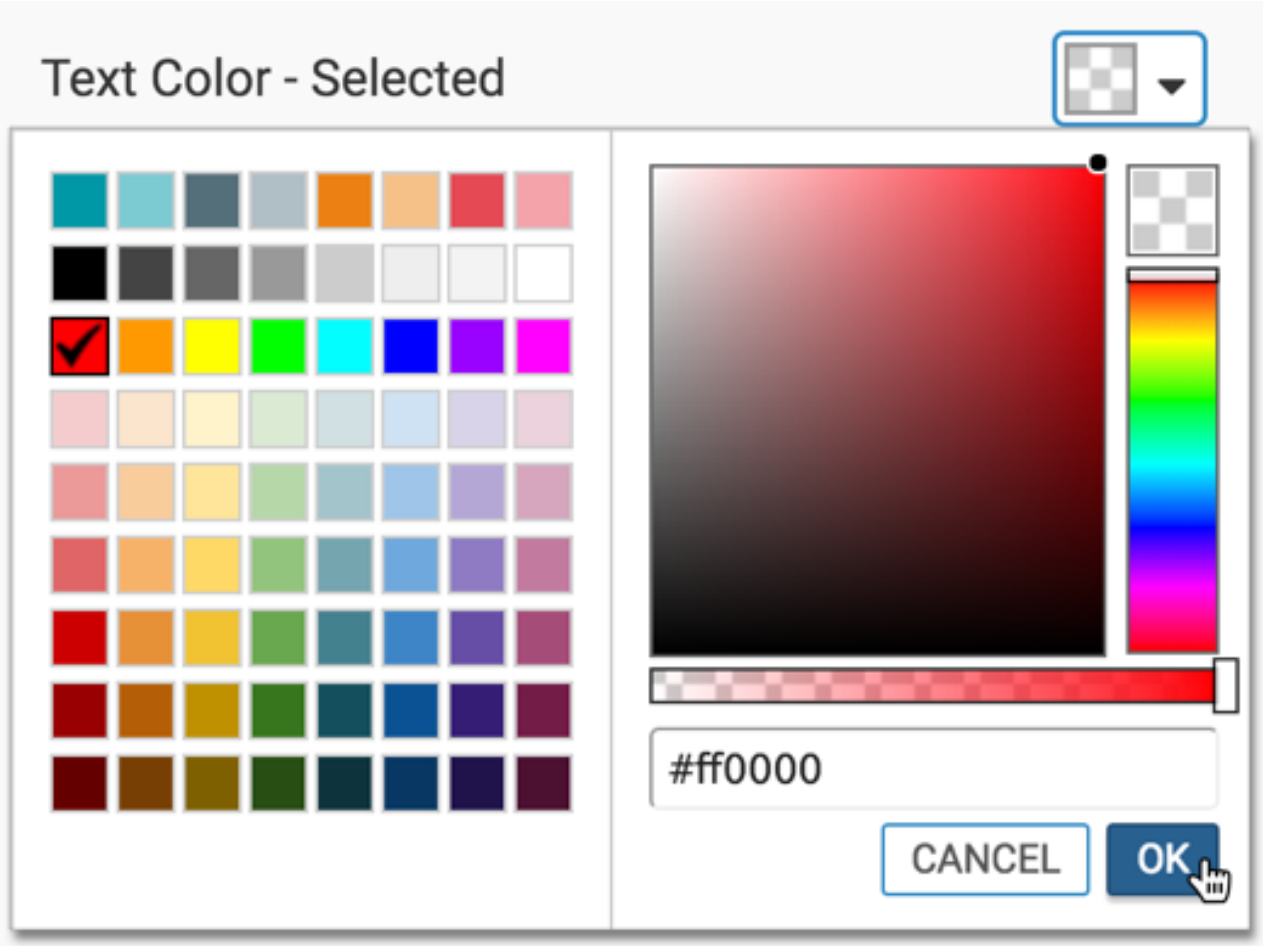

You can see the appearance of the selected menu item when using the default text color (white), and when using red text color:

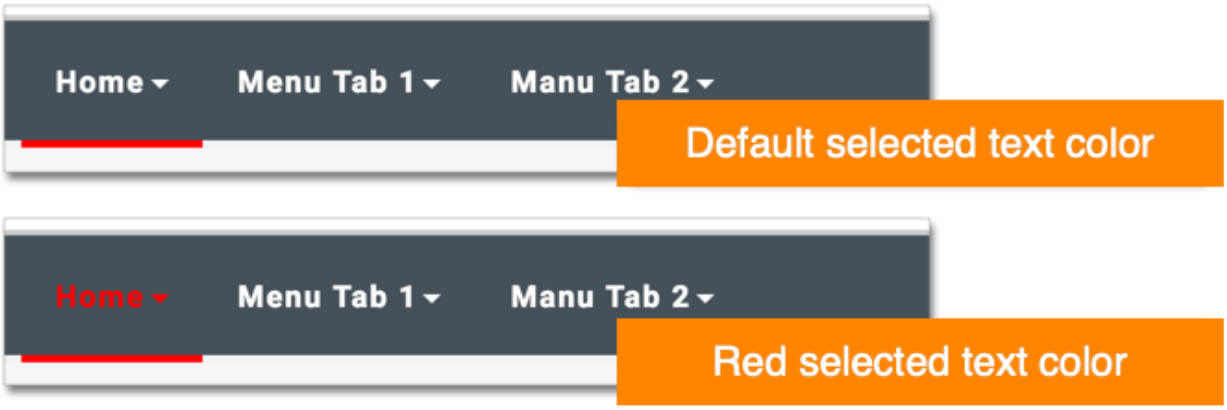

Changing menu text in uppercase

To change text display to uppercase font in the App Designer, navigate to the Menu Bar menu of the Style menu, and select the option Uppercase.

By default, this is not enabled.

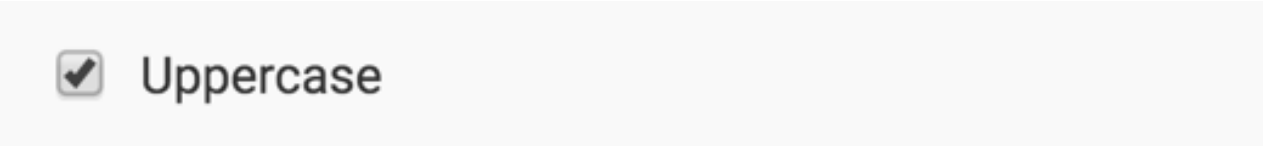

You can see the effect of changing the font to uppercase:

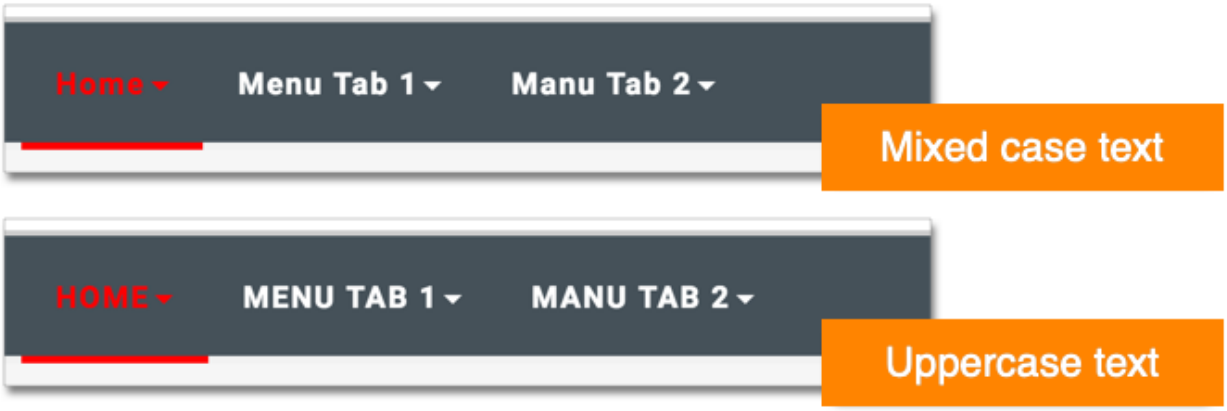

Customizing logo for apps

About this task

To get to the options in the Logo style menu of the App Designer, follow these steps.

Procedure

1. On the right side of App Designer, click the Style menu.

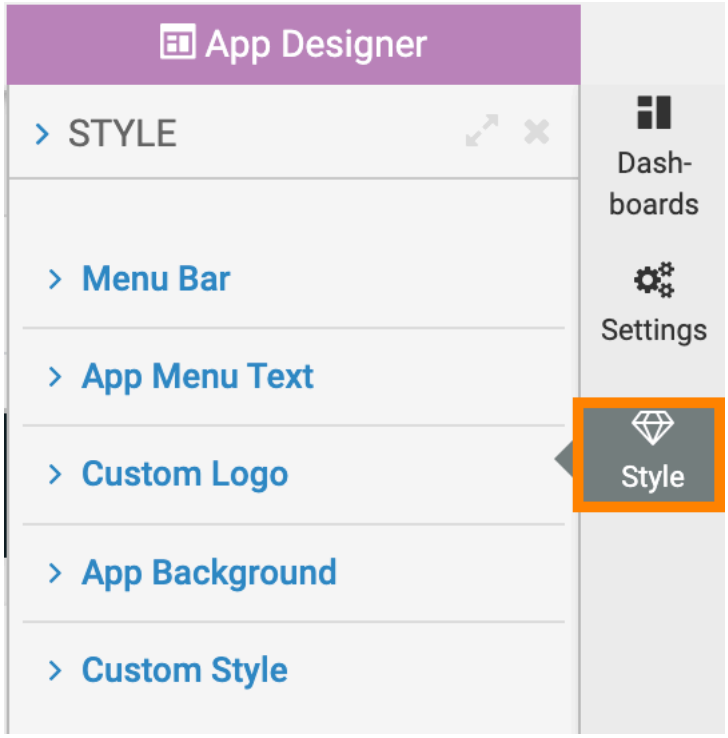

2. In the Style menu, click Custom Logo.

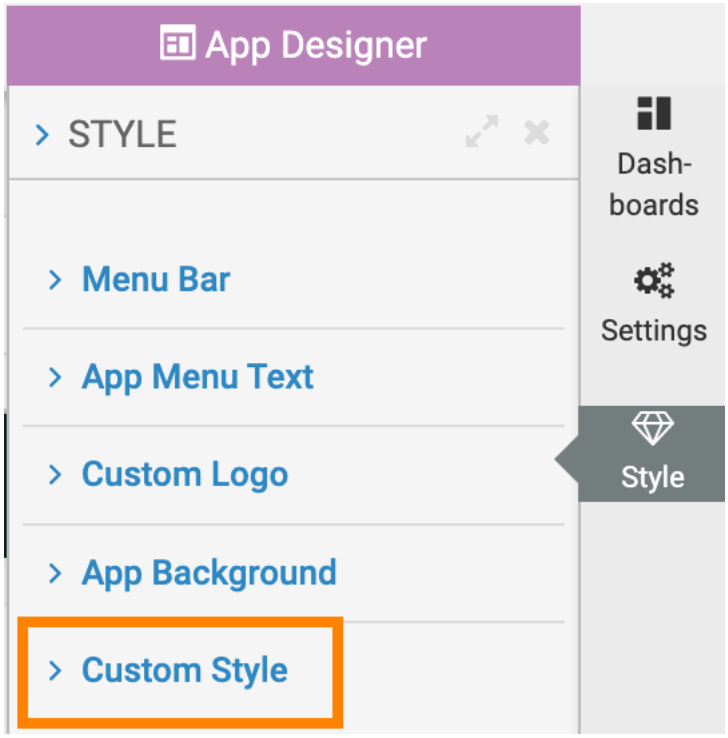

3. Select one of the following options.

Changing logo image

To change the logo that appears in the app, navigate to the Logo menu of the Style menu. In the Logo Image URL text box, enter the address of an image previously loaded into Cloudera Data Visualization as a static asset.

The entry must have the following form, or you can use an external imag URL address.

```
http://IP_ADDRESS:PORT/arc/reports/staticasset/raw/ASSET-ID.png
```

Where *ASSET-ID* is a numerical value that represents the image, and *IP_ADDRESS*:*PORT* is the full address of the Cloudera Data Visualization Server that contains the image. The value depends on the order of loading static assets.

In this example, the Wikipedia logo has been added as the app logo using the following URL: https://www.wikipedia.org/portal/wikipedia.org/assets/img/Wikipedia-logo-v2@2x.png

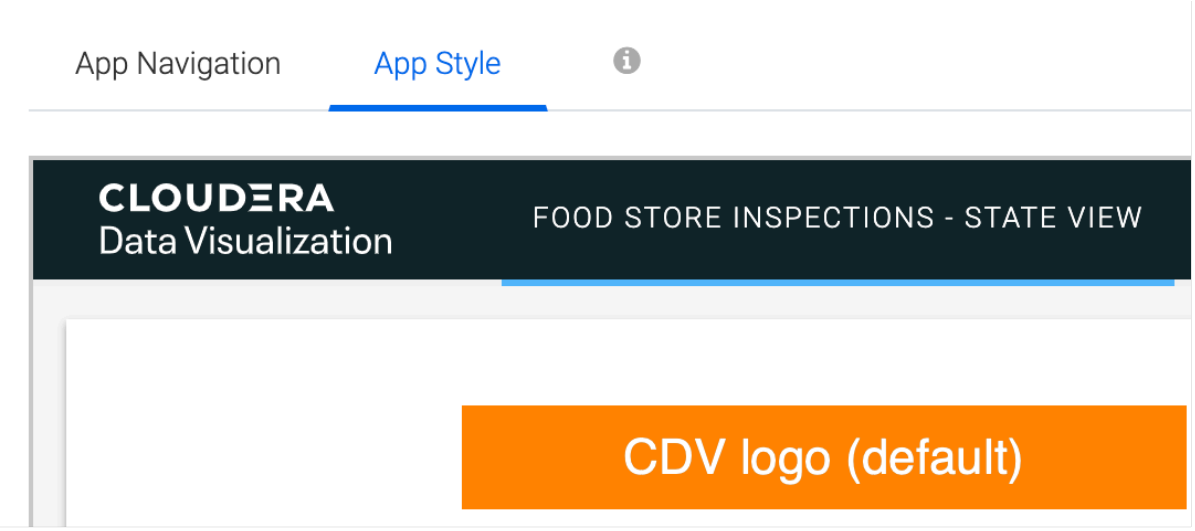

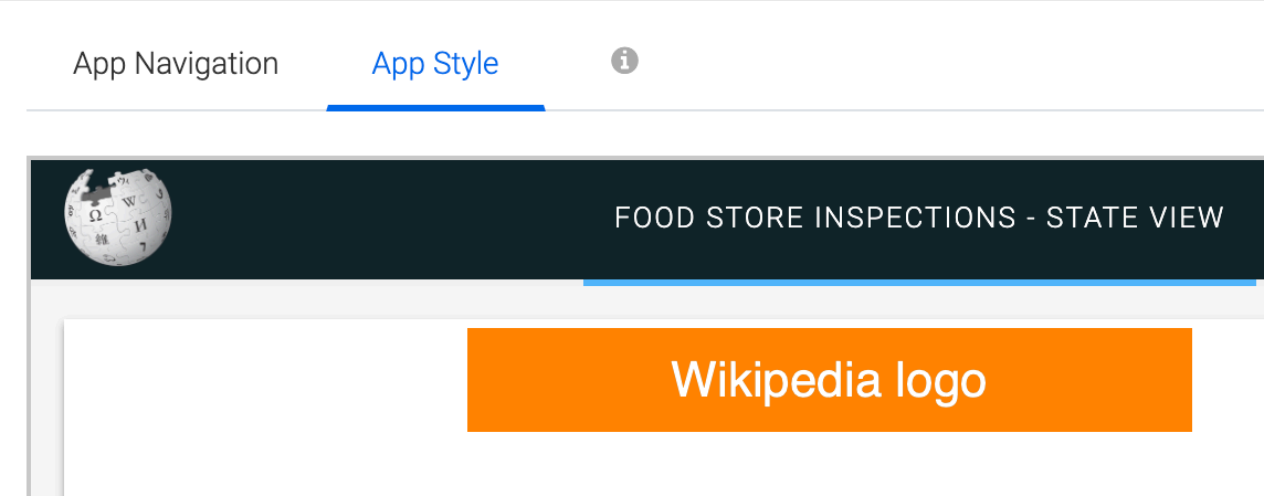

Changing logo width

To change the width of the logo that appears in the app, in the App Designer, navigate to the Logo menu of the Style menu. In the Width selector, specify the new value for width.

The default value is 170 px.

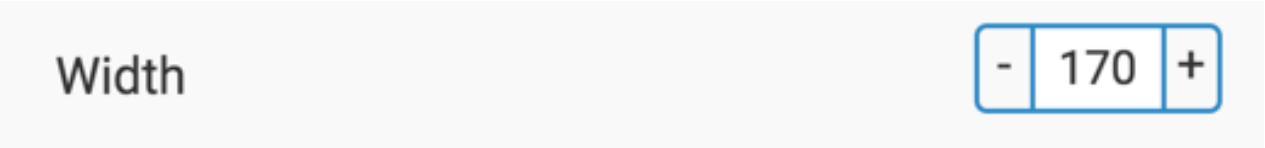

Changing logo height

To change the height of the logo that appears in an app in the App Designer, navigate to the Logo menu of the Style menu. In the Height selector, specify the new value for width.

The default value is 60 px.

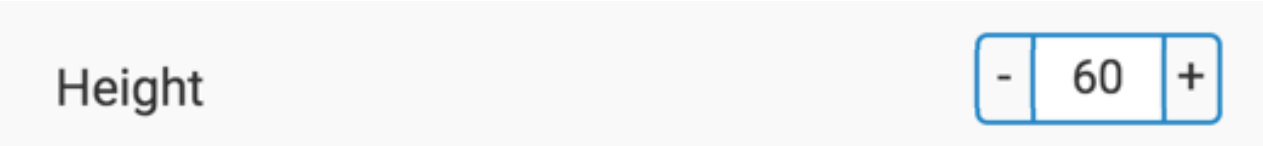

Changing logo top margin

To change the top margin of the logo that appears in the app in the App Designer, navigate to the Logo menu of the Style menu. In the Top Margin selector, specify the new value.

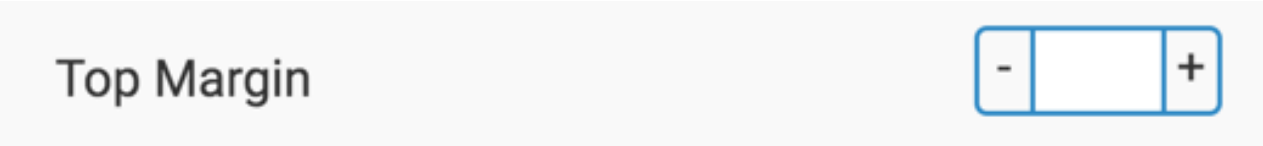

Changing logo left margin

To change the left margin of the logo that appears in the app in the App Designer, navigate to the Logo menu of the Style menu. In the Left Margin selector, specify the new value.

Customizing app background

About this task

To get to the options in the Background style menu of the App Designer, follow these steps.

Procedure

1. On the right side of App Designer, click the Style menu.

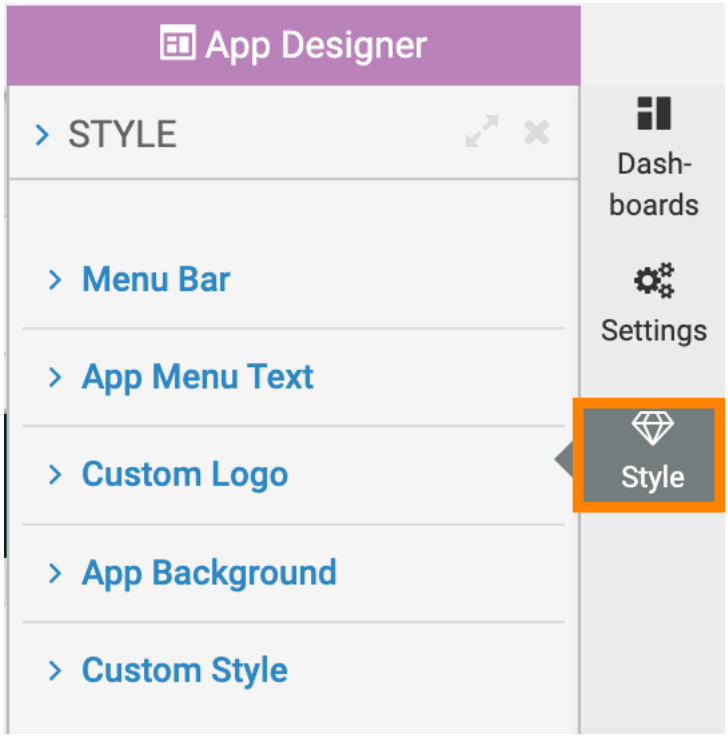

2. In the Style menu, click Background.

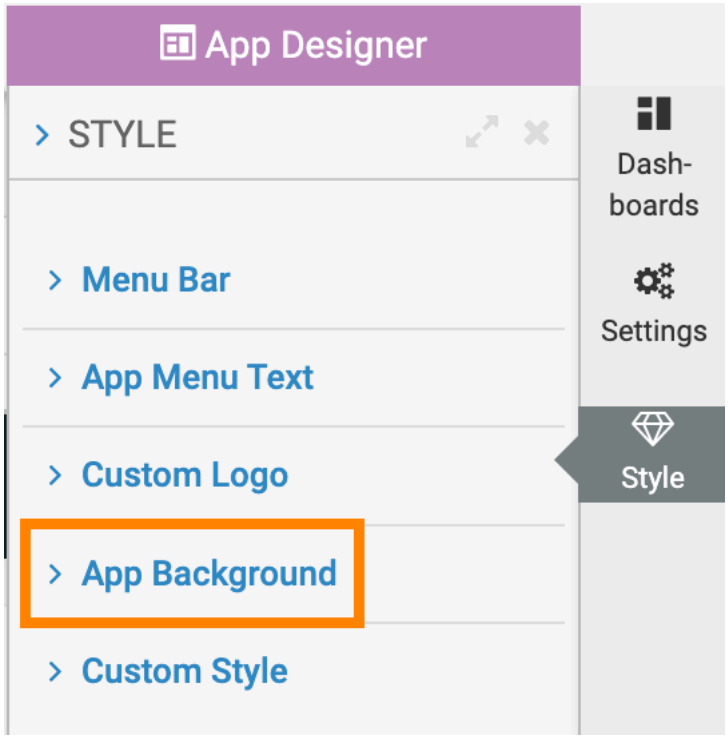

3. Select one of the following options.

Changing app background color

To change the color of the app background, which extends beyond the borders of the dashboard, navigate to the App Background menu of the Style menu, and change the color in the Background Color selector.

You can select one of the palette colors, or specify a custom color.

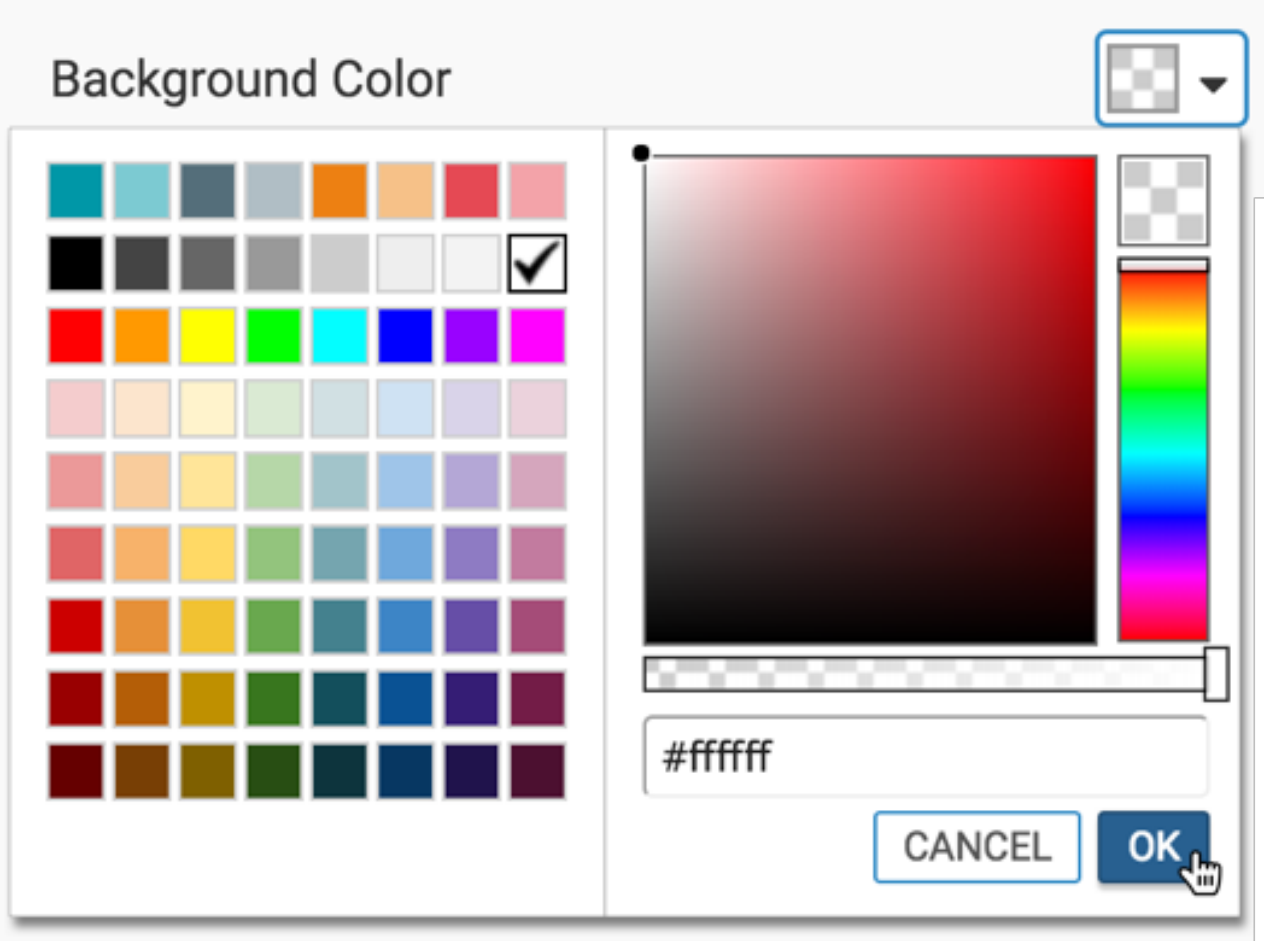

For example, you can see the effect of changing the default background color (light grey) to white:

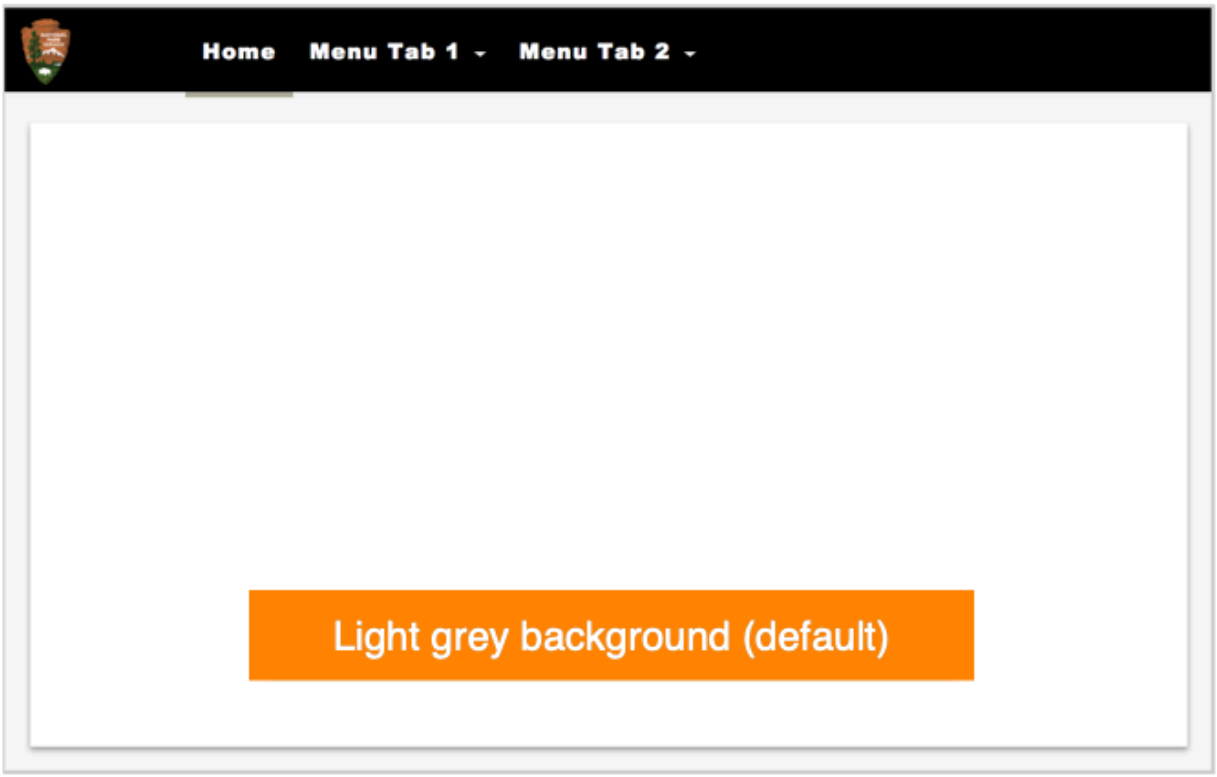

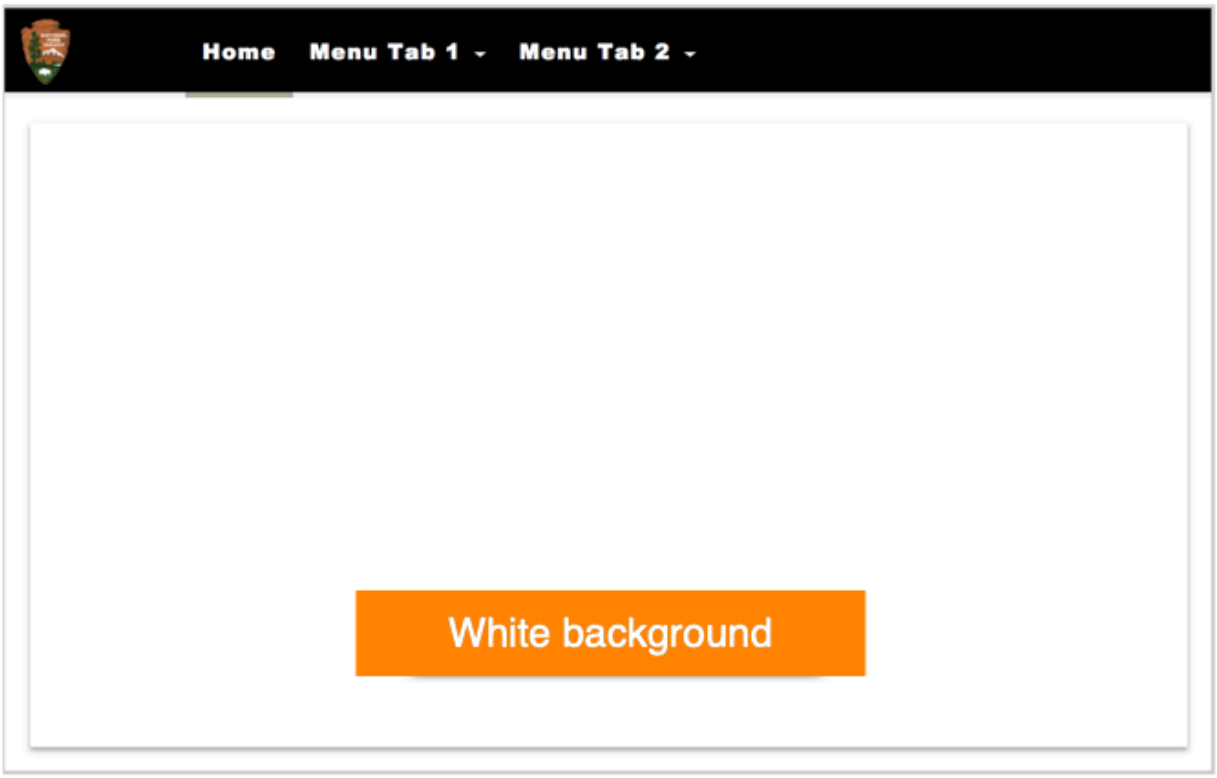

Removing dashboard shadow

To remove the dashboard shadow in apps, navigate to the App Background menu of the Style menu, and select the Hide Dashboard drop shadow option.

This option is not on by default.

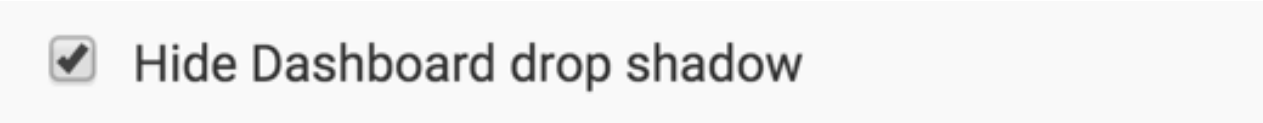

For example, you can see the effect of changing the default background color (light grey) to white:

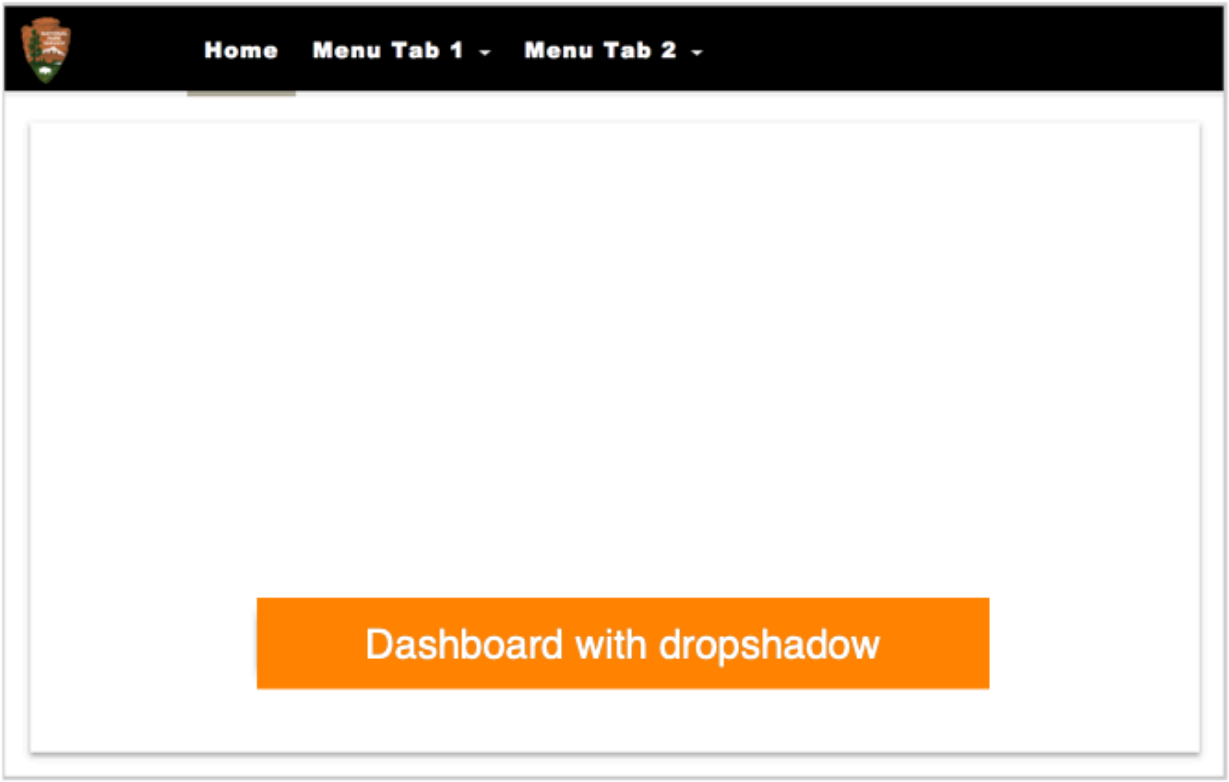
Home
Menu Tab 1
Menu Tab 2
Dashboard with dropshadow

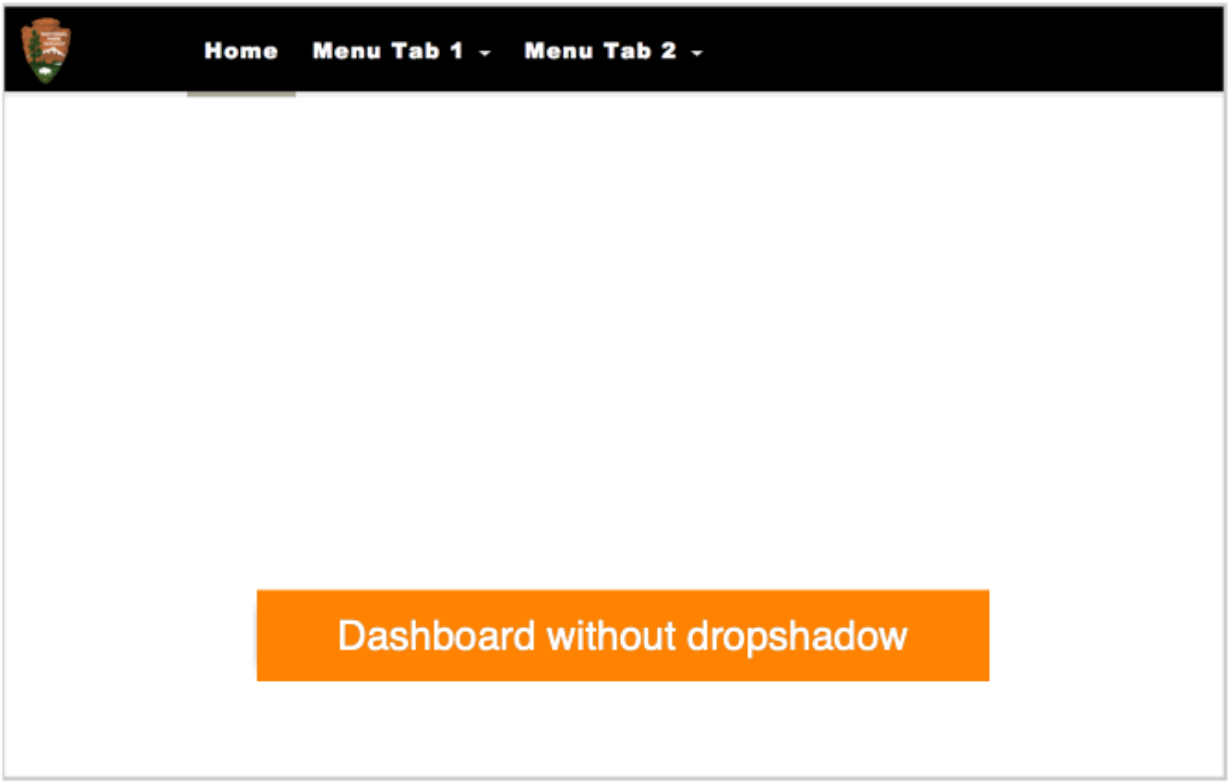
Home
Menu Tab 1
Menu Tab 2
Dashboard without dropshadow

Adding custom styles to apps

Cloudera Data Visualization enables you to add custom styling to applications.

About this task

To use the Custom Style menu, follow these steps.

Procedure

1. On the right side of App Designer, click the Style menu.

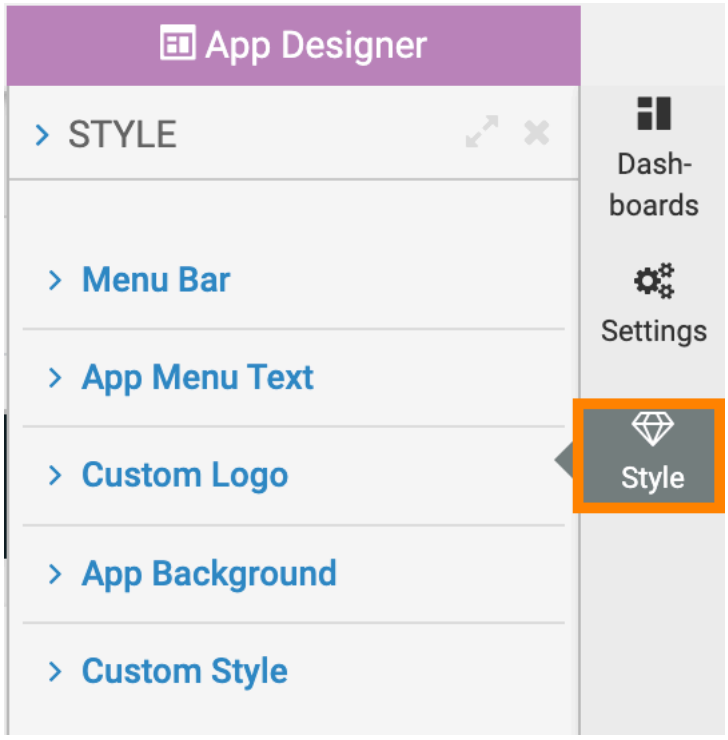

2. In the Style menu, click Custom Style.

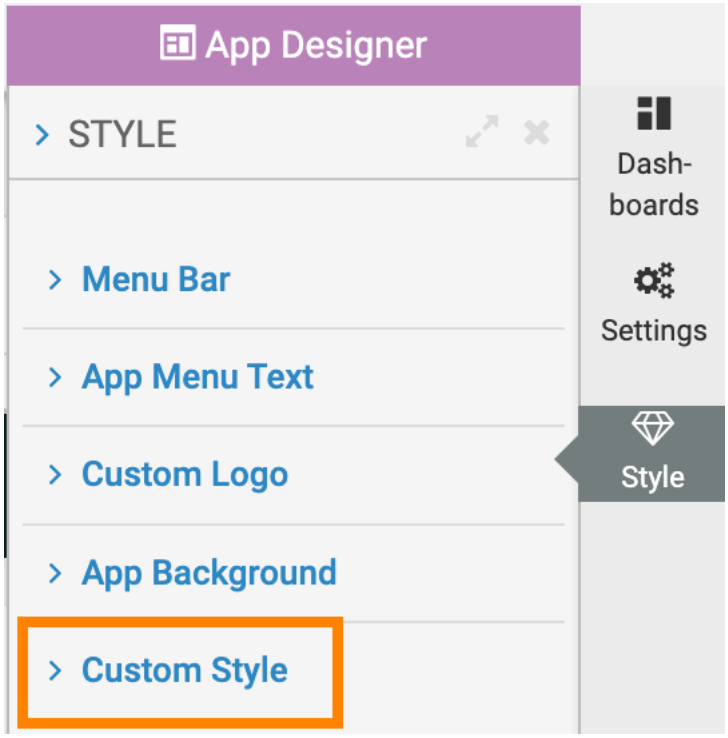

3. Select one of the following options.

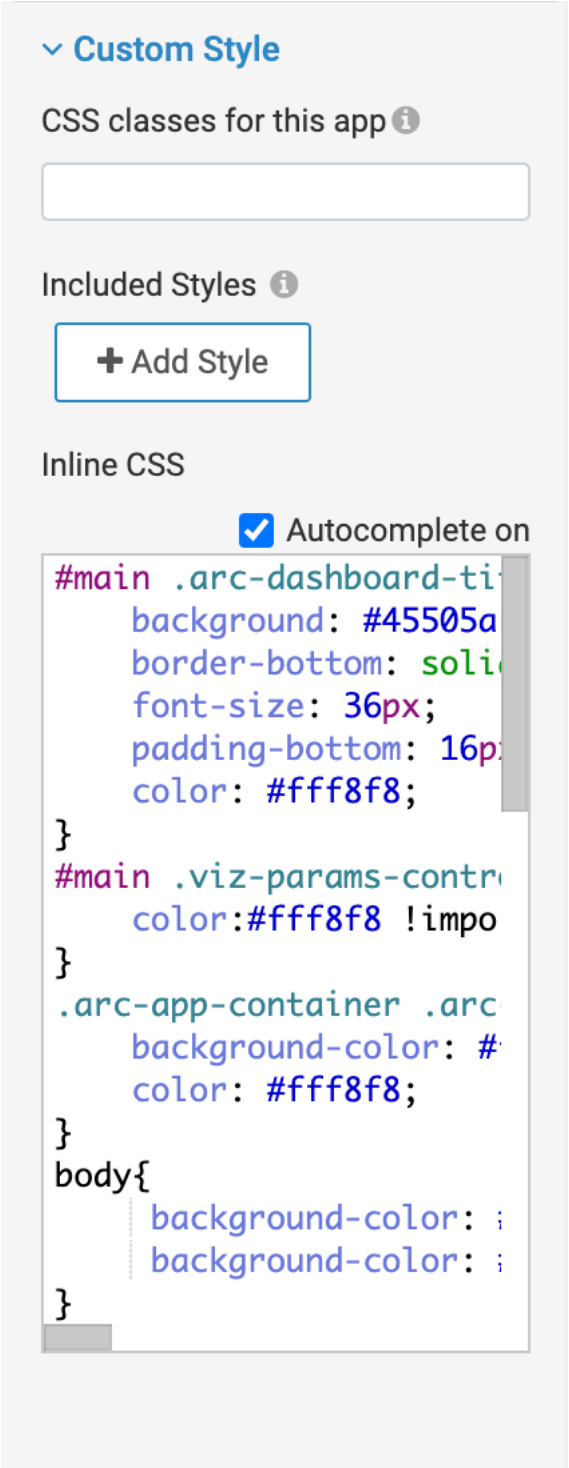

Adding CSS classes to applications

In the App Designer interface, navigate to the Custom Style menu, and enter the name(s) of existing CSS classes in the text box CSS classes for this app.

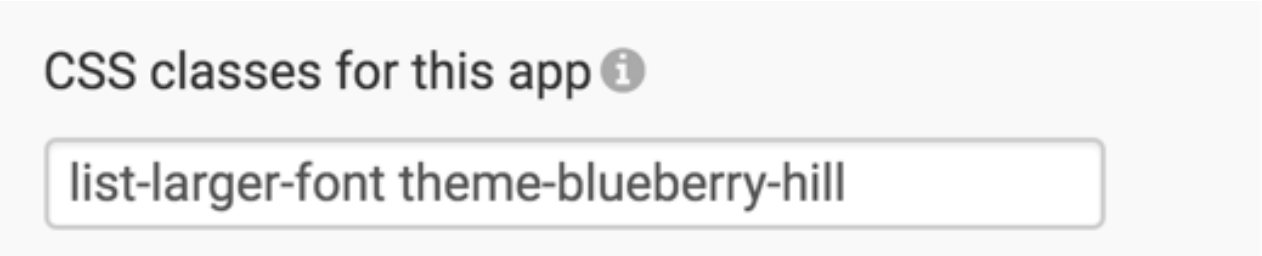

Adding included styles to applications

Procedure

1. In the App Designer interface, navigate to the Custom Style menu.
2. Under Included Styles, click Add Style.

3. In the Pick Custom CSS window modal, select the styles to add to the app and click ADD.

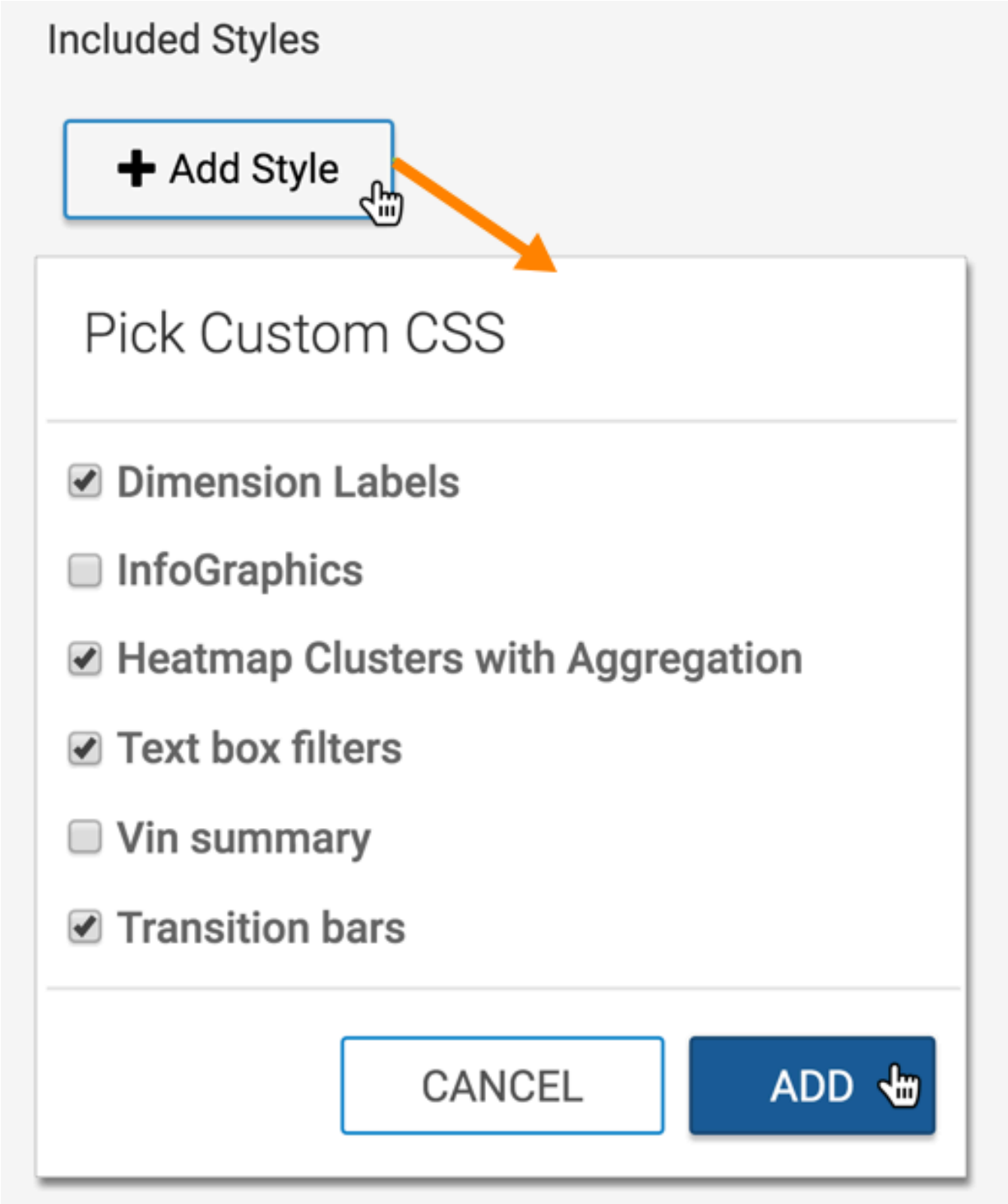

To add styles to a specific visual or dashboard, see *Adding included styles to visuals* and *Adding included styles to dashboards*.

Related Information

Adding included styles to visuals

Adding included styles to dashboards

Adding inline CSS code to apps

To add simple formatting to all the visuals and dashboards in the app, in the App Designer interface, navigate to the Custom Style menu of an app, and enter a CSS code snippet in the text box under the Inline CSS option.

You can select the Autocomplete option, too.

To apply this operation at the level of a specific visual or dashboard, see *Adding inline CSS code to visuals* and *Adding inline CSS code to dashboards*.

Related Information

Adding inline CSS code to visuals
Adding inline CSS code to dashboards

Enabling navigation from logo

The Settings menu contains the general options for customizing the behavior of apps. You can use it to provide users access to the Visuals page by clicking the logo in the menu bar of the navigation.

Procedure

1. On the right side of the App Designer, click the Settings (cogwheel) menu.

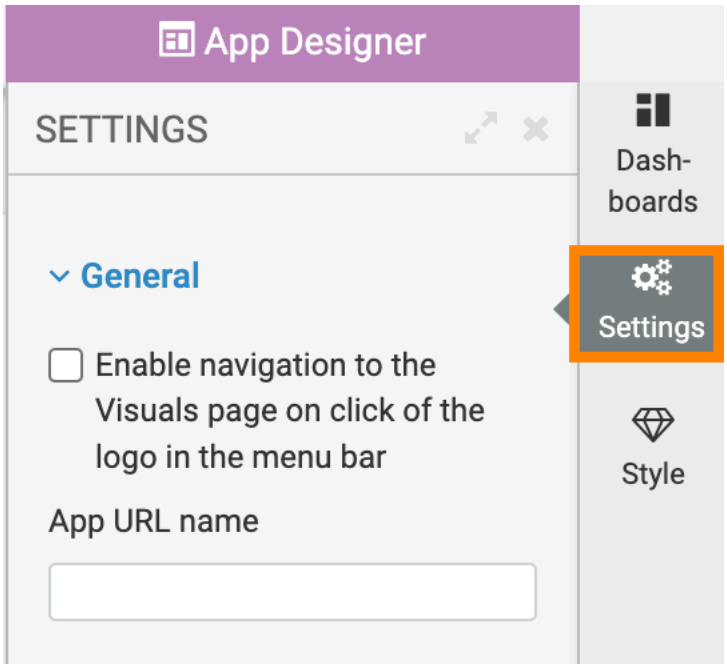

2. In the Settings menu, click General.
3. Select the following option: Enable navigation to the Visuals page on click of the logo in the menu bar.

   By default, this option is not enabled.

Cloudera Data Visualization 7.2.9

Customizing Dashboards

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Customizing general settings for dashboards.....5

Changing dashboard width.....5
Enabling animations.....5
Enabling manual refresh of visuals in a dashboard.....6
Adjusting horizontal margins.....6
Adjusting vertical margins.....7
Displaying a visual in full screen.....8
Adjusting detailed data columns.....8
Displaying filter selections for each visual.....9
Displaying context menus in visuals.....10
Overriding visual style settings.....10
Adjusting automatic refresh.....11
Enabling or disabling case sensitive dashboard filters.....12
Enabling viewing data in dashboard visuals.....12
Rendering visuals after filters in dashboards.....13

Customizing download settings for dashboards.....13

Enabling PNG and PDF download for dashboards.....14
Changing dashboard scale for PDFs.....14
Selecting page size for PDF downloads.....14
Selecting page orientation for PDF downloads.....14
Enabling headers/footers when printing in PDF.....14
Enabling CSV/Excel download for dashboard visuals.....14
Enabling email for a dashboard.....15

Showing borders around visuals.....15

Customizing dashboard style.....15

Basics styles.....16
Dashboard text color.....17
Dashboard background color.....18
Dashboard page background color.....20
Dashboard titles styles.....21
Dashboard titles font family.....21
Dashboard titles font size.....21
Dashboard titles font weight.....22
Dashboard titles text color.....22
Dashboard titles background color.....22
Dashboard titles alignment.....22
Hiding dashboard titles.....22
Dashboard subtitles styles.....22
Dashboard subtitles font family.....22
Dashboard subtitles font size.....22
Dashboard subtitles text color.....23
Dashboard subtitles background color.....23
Dashboard subtitles alignment.....23

Dashboard filter styles....23
Aligning dashboard filters....23
Using filter labels inline....23
Hiding filters line separator....24
Hiding borders for filters in dashboards....24
Changing dashboard filter font....24
Changing dashboard filter font size....24
Changing text color for dashboard filter....24
Changing selected text color for dashboard filter....24
Changing container color for dashboard filter....24
Changing background color for dashboard filter....24
Visual basics styles in dashboards....25
Visual basic font family in dashboards....25
Visual basic font size in dashboards....25
Visual basic background color in dashboards....25
Visual border styles in dashboards....25
Show borders on visuals in dashboards....25
Change visual border color in dashboards....26
Change visual border radius in dashboards....26
Visual titles styles in dashboard....26
Visual title font family in dashboards....26
Visual title font size in dashboards....26
Visual title font weight in dashboards....26
Visual title text color in dashboards....27
Visual title background color in dashboards....27
Visual title alignment in dashboards....27
Visual subtitles styles in dashboard....27
Visual subtitle font family in dashboards....27
Visual subtitle font size in dashboards....27
Visual subtitle text color in dashboards....27
Visual subtitle background color in dashboards....28
Visual subtitle alignment in dashboards....28
Visual axes and labels styles in dashboards....28
Axis label font size in dashboards....28
Axis ticks font size in dashboards....28
Visual label font size in dashboards....28
Visual label title text color in dashboards....29
Visual trellis font size in dashboards....29
Visual legend font size in dashboards....29
Tooltips in dashboards....29
Tooltip font family in dashboards....29
Tooltip font size in dashboards....29

Adding custom styles to dashboards....30
Adding CSS classes to dashboards....30
Adding included styles to dashboards....30
Searching custom styles in dashboards....30
Sorting custom styles in dashboards....30
Adding inline CSS code to dashboards....31

Resetting dashboard style....31

Creating a dashboard with multi-page table visuals....32

Customizing general settings for dashboards

Cloudera Data Visualization supports several options for display settings for dashboards. To change the options for dashboard settings, follow these steps.

Procedure

1. On the right side of Dashboard Designer, click the Settings menu.
2. Open the General menu.
3. Select the option you need.

Changing dashboard width

You can change the width of a dashboard in Cloudera Data Visualization.

To change the width of a dashboard, navigate to the General menu, and adjust the value of the Dashboard width option. The default value is 1024. To have dashboards extend to the full width of the page, clear the field.

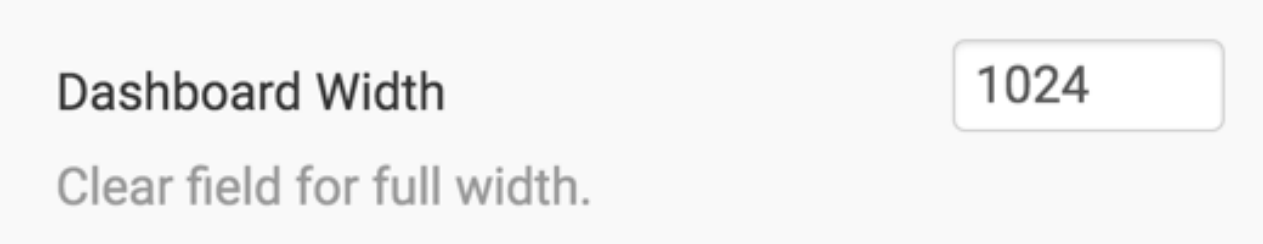

For example, we decreased the dashboard width from 1024 to 500 in the following dashboard:

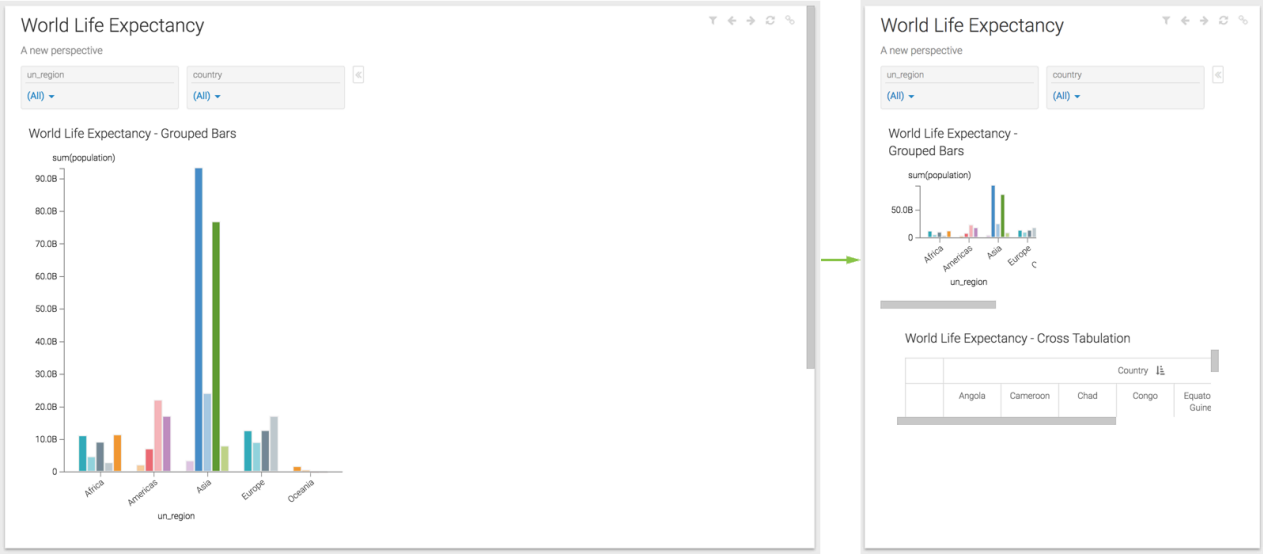

Enabling animations

Cloudera Data Visualization enables visual transitions in dashboards for the following visual types: bars, lines, combo, area, scatter, grouped bar, histogram, sparklines, pie, packed bubbles, and correlation heat map.

About this task

Note: This feature is only available when you enable visual transitions at site level. For more information, see *Enabling visual transitions*.

To enable animations on specific dashboard visuals, follow these steps:

Procedure

1. Open the dashboard.
2. In Dashboard Designer, select the Settings menu.
3. Click the General menu to expand it.
4. Select the Enable animations option.

Related Information

Enabling visual transitions

Enabling manual refresh of visuals in a dashboard

About this task

If you have a slow back-end data connection and your dashboard is loading slowly, you can choose to load the filters first, specify their settings, and then manually load the visuals. This topic describes how to enable manual refresh.

To enable manual refresh of visuals in a dashboard, follow these steps:

Procedure

1. Open the dashboard in Edit mode.
2. In the Dashboard Designer, select the Settings menu.
3. Click the General menu to expand it.
4. Select the Refresh visuals manually option.

Results

This adds a REFRESH VISUALS button to the dashboard.

Adjusting horizontal margins

You can change the horizontal margins of a dashboard in Cloudera Data Visualization.

To change the horizontal margins of a dashboard, navigate to the General menu, and adjust the value in the selector for the Horizontal Margins option. The default value is 10.

For example, we increased the horizontal margins from 10 to 35 in the following dashboard. Notice that both visuals are shifted to the right, but the filter controls remain stationary.

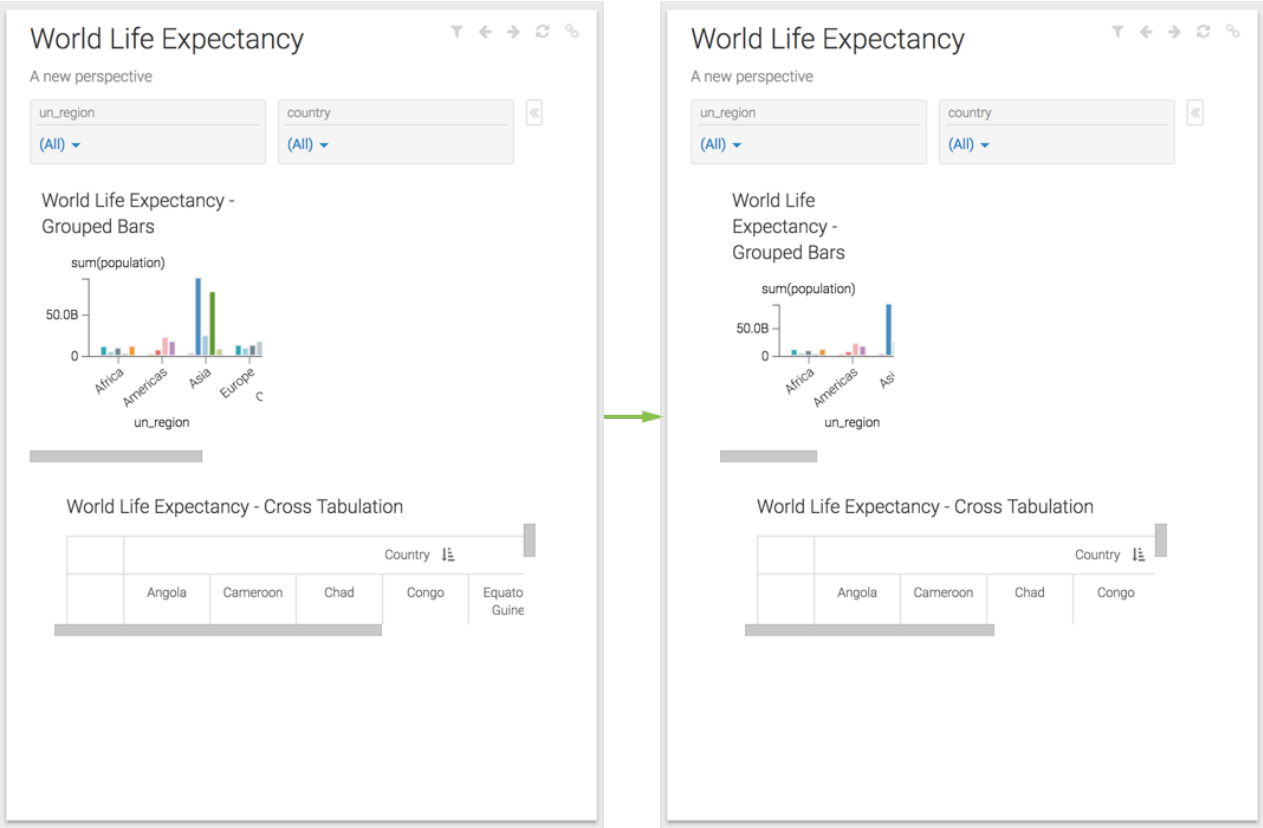

Adjusting vertical margins

You can change the vertical margins of a dashboard in Cloudera Data Visualization.

To change the horizontal margins of a dashboard, navigate to the General menu, and adjust the value in the selector for the Vertical Margins option. The default value is 10.

For example, we increased the vertical margins from 10 to 35 in the following dashboard. Notice that the first visual is shifted downwards, and both visuals are vertically 'squashed'. The filter controls remain stationary.

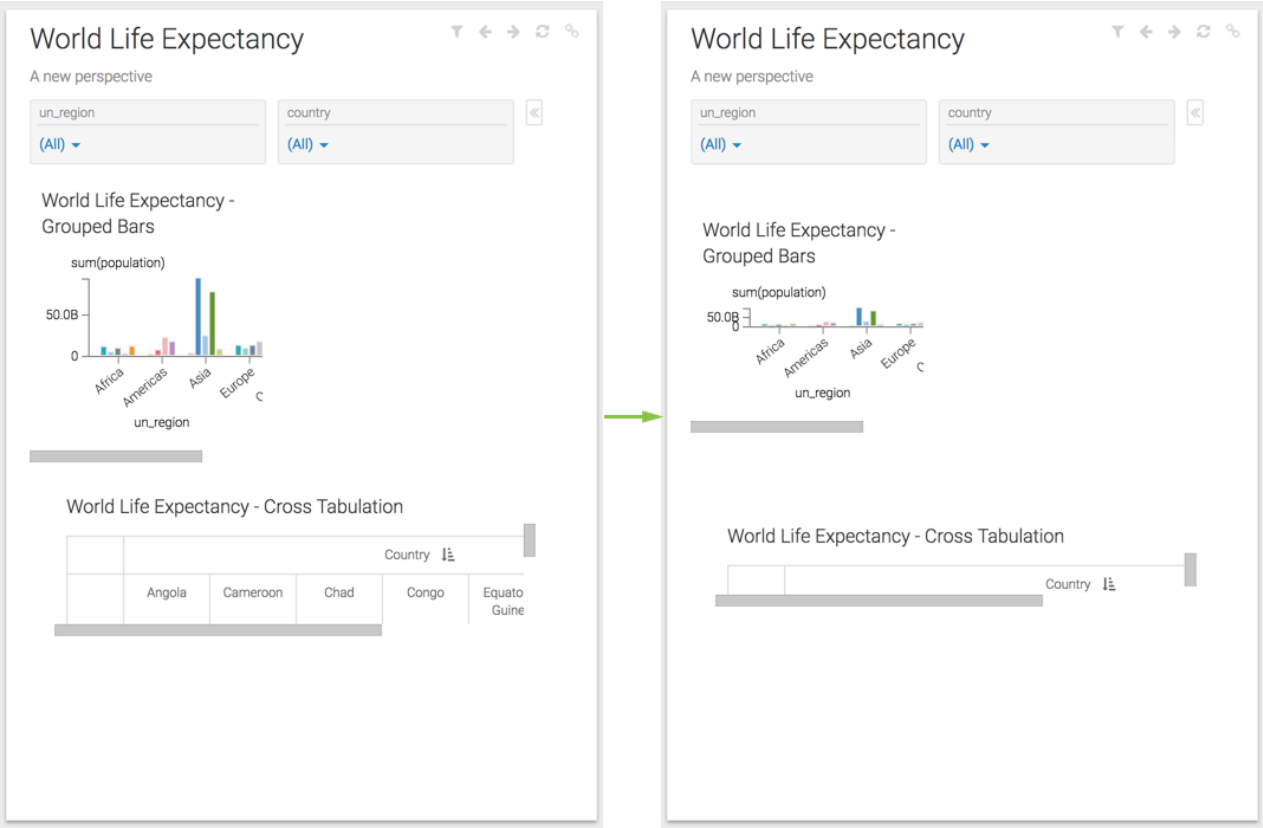

Displaying a visual in full screen

You can display a visual in full screen in Cloudera Data Visualization.

To extend a visual in a dashboard to the full width of the page, navigate to the General menu, and select the Make full screen for single visual option. This option is available if there is only one visual in the dashboard.

For example, compare the two visuals: one in its original size, and the other in full-screen mode.

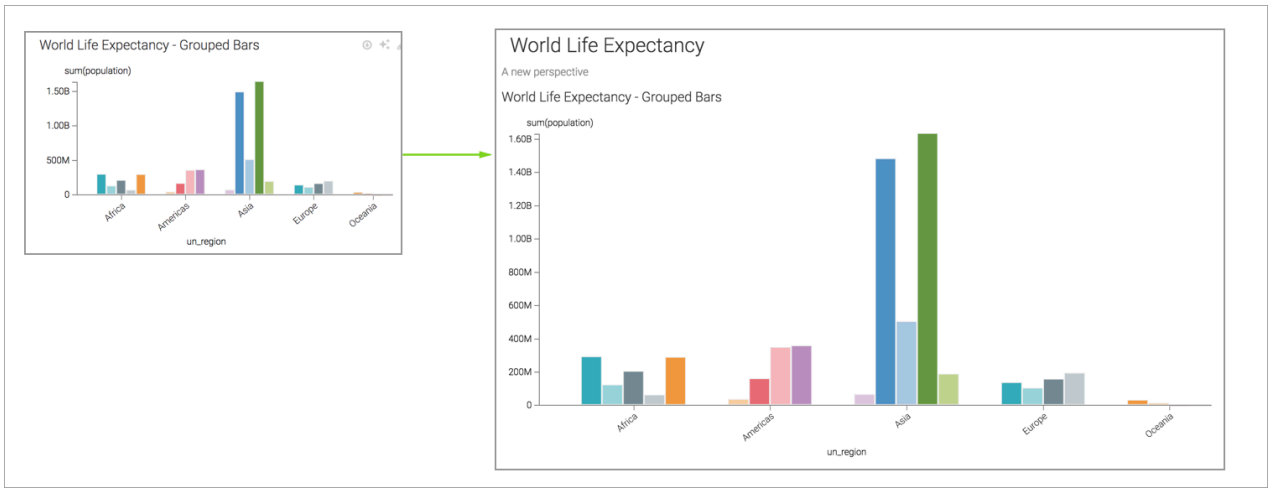

Adjusting detailed data columns

You can change detailed data columns in Cloudera Data Visualization.

To change the number of columns that appear when selecting Detailed Data in a context menu of a dashboard visual, navigate to the General menu, and adjust the value in the selector for the Detailed Data Columns option.

Displaying filter selections for each visual

You can display filter selections for the visuals of a dashboard in Cloudera Data Visualization.

To show active filter selections for each visual in the dashboard, navigate to the General menu, and select the Display filter selections for each visual option.

For example, compare the two dashboards:

- The top dashboard displays filters parameters when you hover the pointer over the filter icon at the top right corner of the dashboard, but not on the visuals themselves.
- The bottom dashboard displays filters and parameters both at the top right corner, and on each visual modified by the filter.

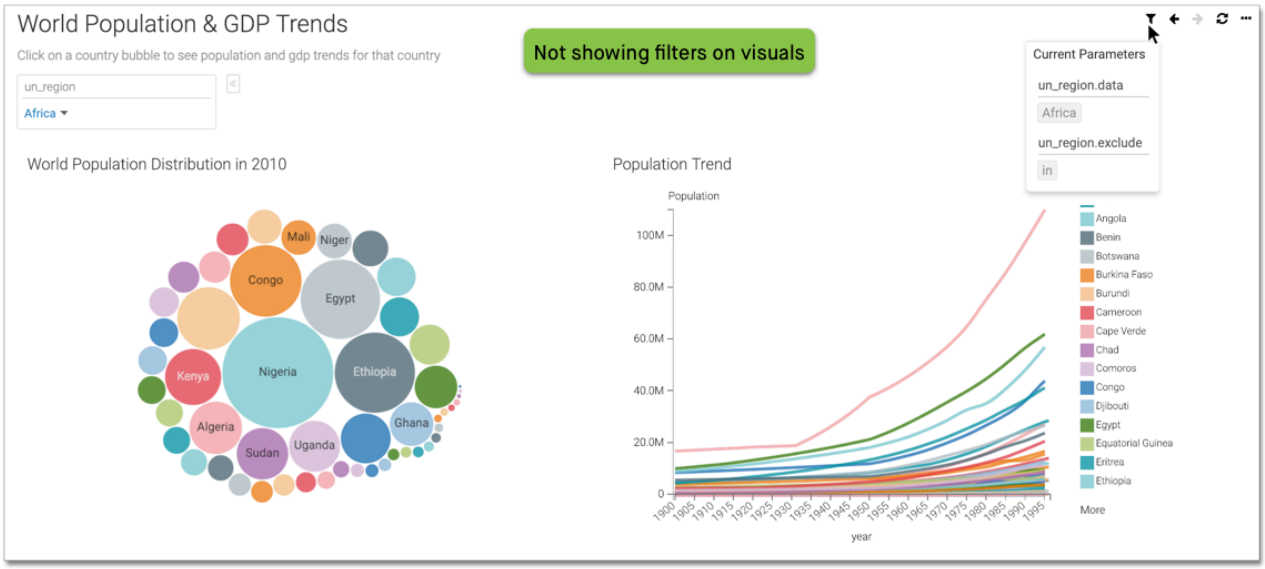

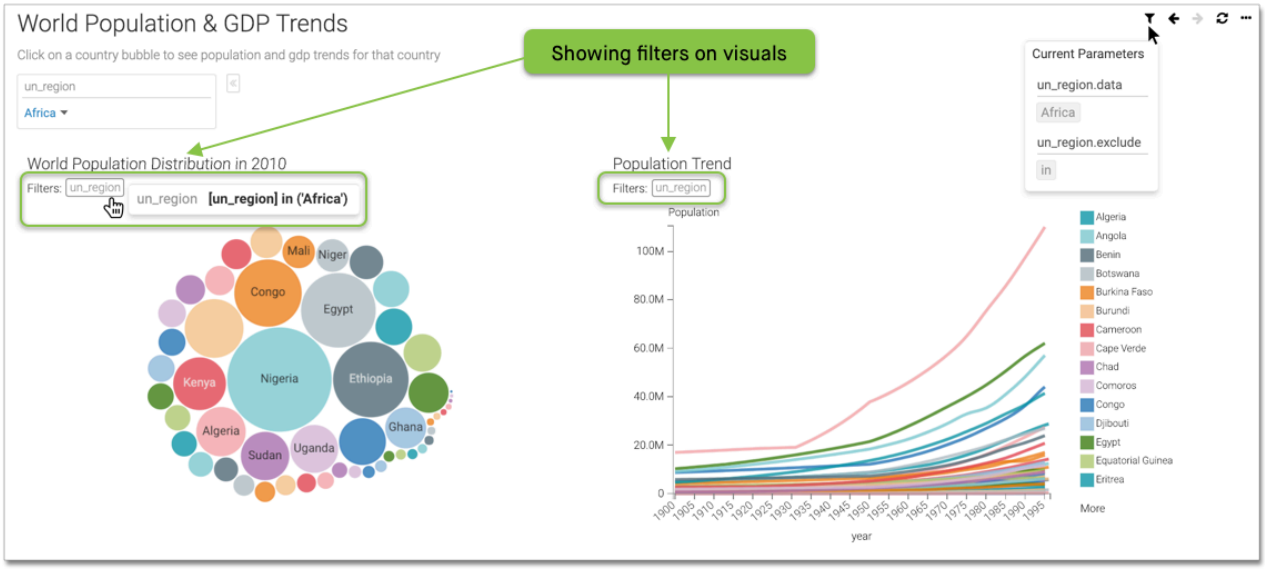

Displaying context menus in visuals

About this task

In a dashboard, you can see the context menu within each visual in both Edit and View mode.

In Edit mode, you can see context menus by default.

Procedure

1. To enable the context menu in View mode, navigate to the General menu.
2. Select the Display context menus within visuals option.
3. Click the selected part of the graph in View mode, to see the context menu.

   Notice the three data discovery options in the context menu, Show Detailed Data, Filter All, and Drill Into.

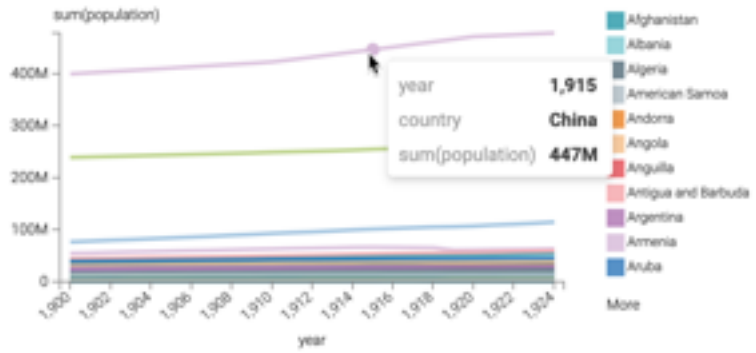

4. Switch to Edit mode, and click the selected part of the graph to show the context menus.

   In Edit mode, you can see more data discovery options in the context menu as compared to View mode.

   In the following image, notice that when you enable context menus at the dashboard level through the Display context menus within visuals setting option, the No Action on Click context menu appears as Show Context Menu on Click.

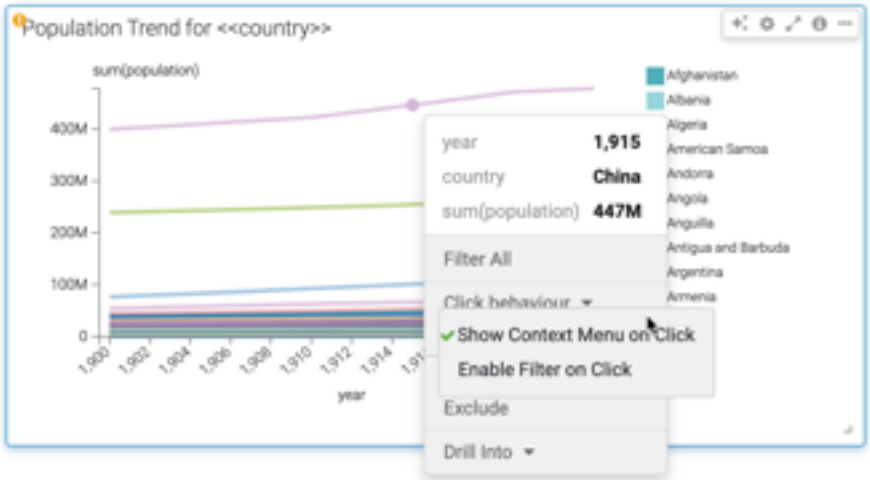

Overriding visual style settings

To achieve consistent styling of all visuals in a dashboard, you can override all individual visual styling choices using a single dashboard setting in Cloudera Data Visualization.

To override the settings of the visuals by the dashboard settings, navigate to the General menu, and select the Override visual style settings option.

To change the styles at dashboard level, see *Dashboard Style Customization*.

Related Information

Customizing style for dashboards

Adjusting automatic refresh

Cloudera Data Visualization offers an automatic refresh feature that refreshes all visuals on the dashboard simultaneously, regardless of the type of visual used (for example: charts, external links, tables). This is especially helpful for dashboards that contain dynamic data, such as stock tickers or real-time event updates.

To enable automatic refresh for your entire dashboard, navigate to the General menu and adjust the value in the selector for the Auto-refresh seconds option.

This sets the refresh rate in seconds for the entire dashboard.

Note: By default, this setting is disabled, and the selector is empty. Once activated, all visuals on the dashboard will refresh at the specified interval.

Example

This example demonstrates how to create a simple dashboard with a refresh rate through using an external link visual to access live information about the 2017 French Presidential election.

While this example uses a link visual, the automatic refresh setting affects the entire dashboard, meaning all visuals will refresh based on the set interval.

1. Start a new dashboard based on any dataset.
2. In the Visual Designer, click New Visual.
3. Hover over the Visual Controls in the top right corner of the new visual and click the (pencil) icon to edit the visual.
4. Select External Link from the Visual Type Catalog.
5. In the URL field, enter https://en.wikipedia.org/wiki/French_presidential_election,_2017.
6. Name and save the new visual.

   In this example, the name of the visual is French Election 2017.
7. In the dashboard Settings, set the Auto-refresh seconds option to 300 seconds, which is equivalent to an update every 5 minutes.

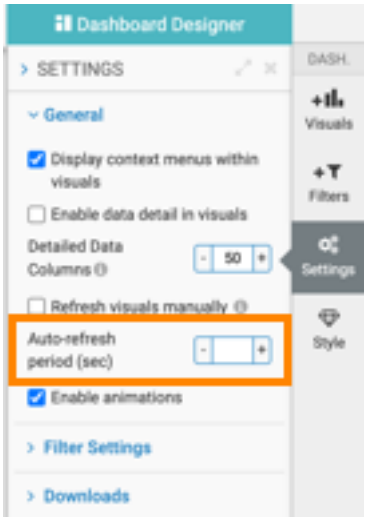

8. Name the new dashboard.

   In this example, the name of the dashboard is News.
9. Click Save.
10. Click ViewSwitch to View mode to observe the periodic updates of the dashboard.

Enabling or disabling case sensitive dashboard filters

By default, Cloudera Data Visualization processes all parameter data in case-sensitive manner. It also has an option for disabling case-sensitive comparison on dashboard settings.

About this task

This example demonstrates how to create a simple dashboard with a refresh rate through using a URL link visual to access live information about the 2017 French Presidential election.

Procedure

- To enable dashboards to use case-sensitive names in output parameters, navigate to the General menu, and select the Parameter names are case sensitive option.

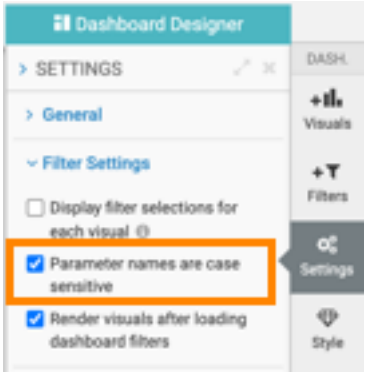

- To disable case-sensitive comparison, de-select this option.

Enabling viewing data in dashboard visuals

You can view the detailed data in each visual of a dashboard Cloudera Data Visualization.

To see the detailed data of each visual in the dashboard, navigate to the Settings General menu, and select the option Enable viewing data within each visual. By default, this feature is turned off.

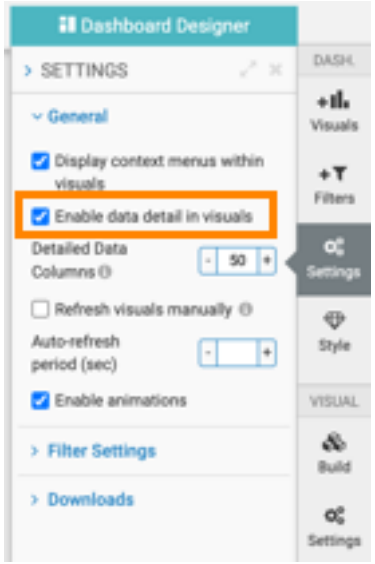

In dashboard Edit mode, note the difference between the available widgets on a focus visual when the option is disabled, and when it is enabled. Clicking the (table) icon opens the Detailed Data window modal, which enables you to examine and download this data.

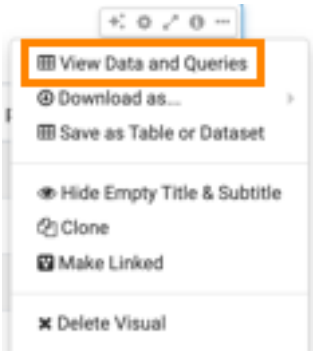

The (table) icon also appears in view-enabled dashboards in View mode.

Rendering visuals after filters in dashboards

You can render the visuals of a dashboard after all filters load in Cloudera Data Visualization.

To load the dashboard visuals after the filtering completes, navigate to the Settings General menu, and select the option Render visuals after loading dashboard filters.. By default, this feature is turned off.

Customizing download settings for dashboards

Cloudera Data Visualization supports several options for dashboard download settings. To change the options, follow these steps.

Procedure

1. On the right side of Dashboard Designer, click the Settings menu.
2. In the SETTINGS menu, click Downloads to open the download options.
3. Select the option you need.

Enabling PNG and PDF download for dashboards

You can download the visuals of a dashboard in PNG and PDF format in Cloudera Data Visualization.

To enable the download of the dashboard as a PNG image, navigate to the Downloads menu and select the Enable "Download as Image/PDF" option.

Changing dashboard scale for PDFs

You can change the dashboard scale for PDF downloads in Cloudera Data Visualization.

To change the scale of the PDF file, navigate to the Downloads menu and adjust the numerical value of the option Print PDF Scale option.

Selecting page size for PDF downloads

You can select the page size when downloading a PDF in Cloudera Data Visualization.

To change the page size of PDF downloads, navigate to the Downloads menu, and select the appropriate option in the PDF Page Size menu.

Cloudera Data Visualization supports the Letter and A4 standard sizes.

Selecting page orientation for PDF downloads

About this task

You can select the page orientation when downloading a PDF file in Cloudera Data Visualization.

To change the page orientation of PDF downloads, navigate to the Downloads menu, and select the appropriate option in the PDF Page Orientation menu.

Cloudera Data Visualization supports Portrait and Landscape.

Enabling headers/footers when printing in PDF

You can enable headers and footers for PDF files in Cloudera Data Visualization.

To add headers and footers to PDF, navigate to the Downloads menu and select the Enable PDF Header and Footer option.

Enabling CSV/Excel download for dashboard visuals

You can enable CSV/Excel download for visuals in a dashboard in Cloudera Data Visualization.

To enable the download of the data in the visuals as CSV or Excel files, navigate to the Downloads menu and select the Enable "Download as CSV/Excel" for visuals option.

In View mode, the visuals of the dashboards show the CSV download option, the icon when you hover the pointer over the Visual Controls.

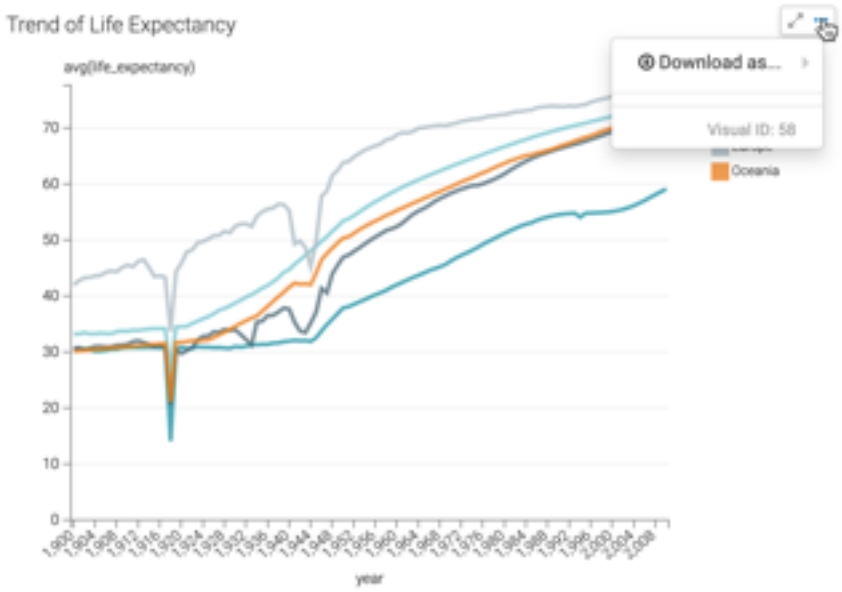

Enabling email for a dashboard

You can email dashboards in Cloudera Data Visualization.

Note: Before using the email function, you must enable the feature in site settings. Only site administrators can customize email settings to allow users to share visuals through email. For instructions, see *Email settings*.

To allow the user to email dashboards, navigate to the Downloads menu and select the Enable emails option.

Related Information

Email settings

Showing borders around visuals

You can add borders to create a frame around visuals in a dashboard in Cloudera Data Visualization.

To show borders for all visuals, navigate to the Style > Visual Borders menu, and select the Show borders around visuals option. By default, visuals in a dashboard do not show borders.

For example, compare the two visuals: one with a hidden border, and the other with a border showing.

Customizing dashboard style

Cloudera Data Visualization supports several options for styling on dashboards.To change the options for dashboard style, follow these steps.

Procedure

1. On the right side of Dashboard Designer, click the Style menu.

2. In the Style menu, you can see a list of options that contain all the style adjustments.

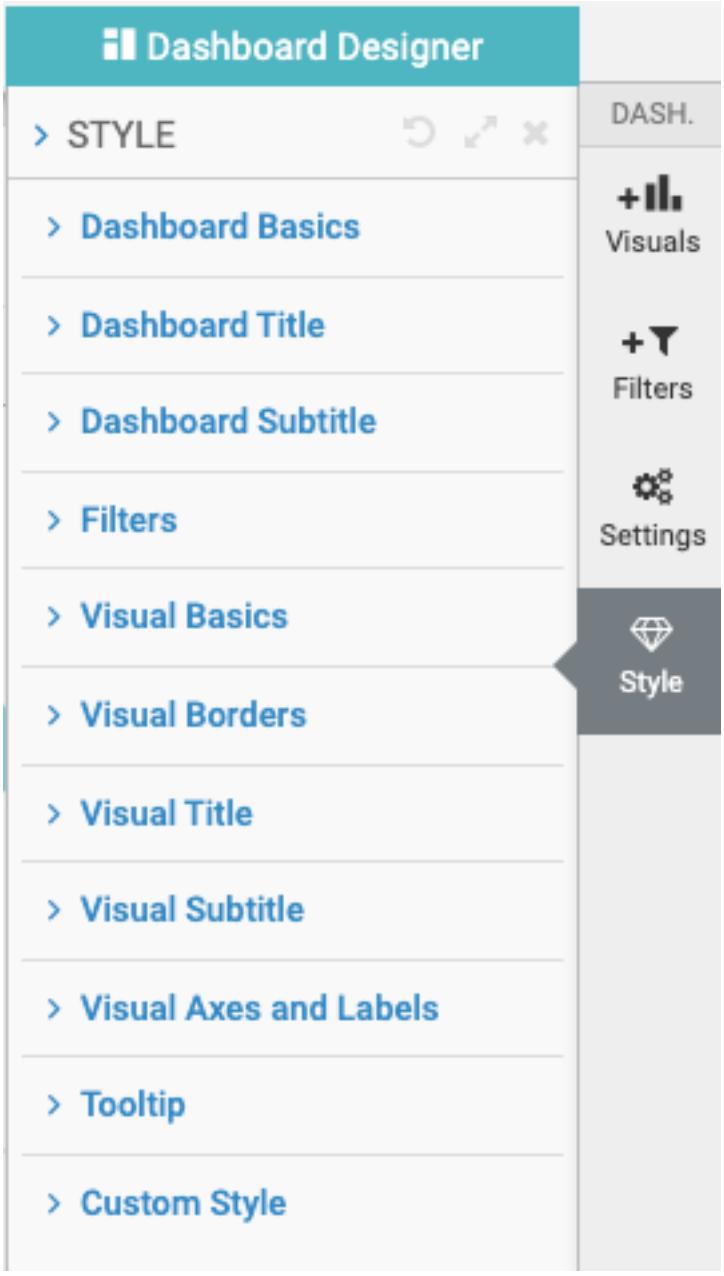

3. Select the option you need.

Basics styles

About this task

You can adjust several style options for basics in Cloudera Data Visualization dashboards. To get to the options in the Dashboard Basics menu, follow these steps:

Procedure

1. On the right side of Dashboard Designer, click the Style menu.
2. In the Style menu, click Dashboard Basics.

3. Select one of the options.

Dashboard text color

You can change the color of the text in the dashboard in Cloudera Data Visualization.

To change the color of the text in the dashboard, navigate to the dashboard Style menu, select the Dashboard Basics menu, and select a new color in the Text Color selector.

You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette. You can also adjust color opacity. The default is no color.

The following image shows the dashboard text before changing the color.

The following image shows the dashboard text after changing the color.

Dashboard background color

You can change the color of the background in the dashboard in Cloudera Data Visualization.

To change the color of the background in the dashboard, navigate to the dashboard Style menu, select the Dashboard Basics menu, and select a new color in the Background Color selector.

You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity). The default is no color.

The following image shows the dashboard background before changing the color.

The following image shows the dashboard background after changing the color.

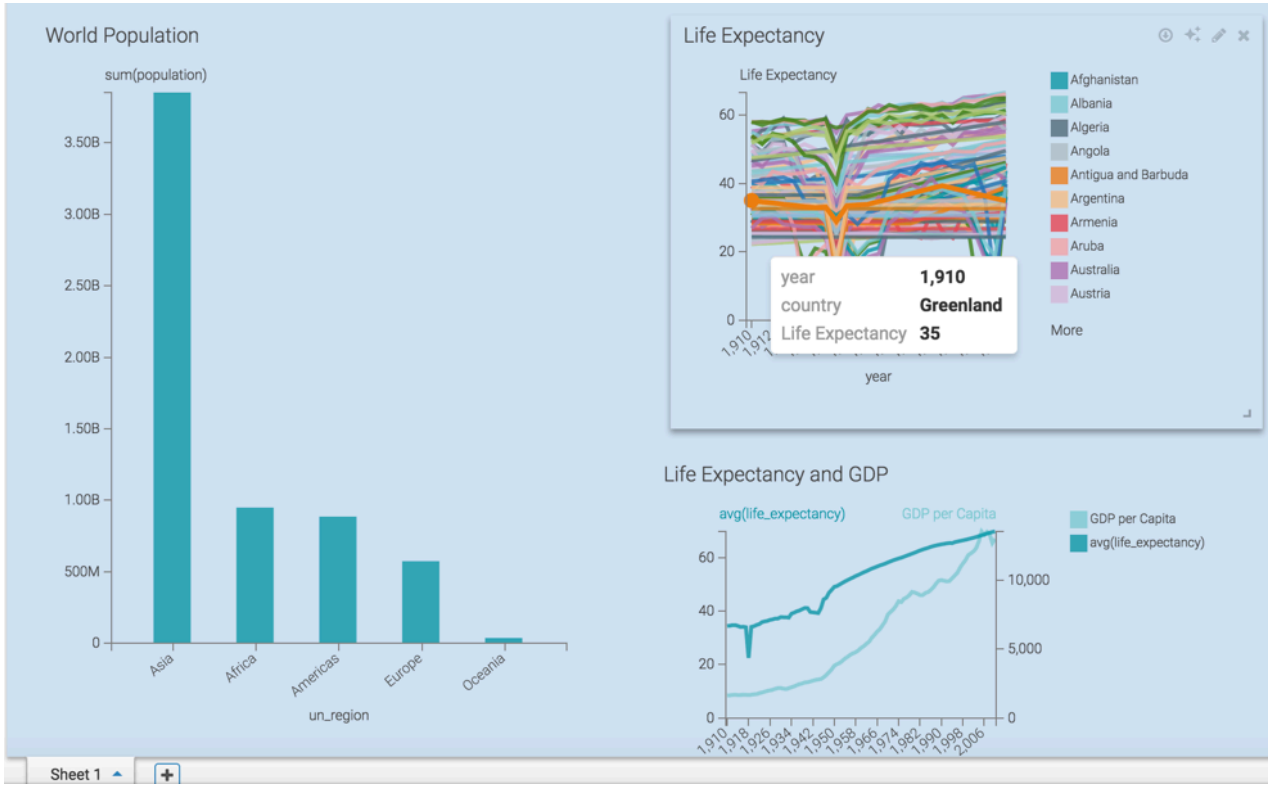

Dashboard page background color

You can change the color of the page background in the dashboard in Cloudera Data Visualization.

To change the color of the page background in the dashboard, navigate to the dashboard Style menu, select the Dashboard Basics menu, and select a new color in the Page Background Color selector.

You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette. You can also adjust color opacity. The default is no color.

The following image shows the dashboard page background before changing the color.

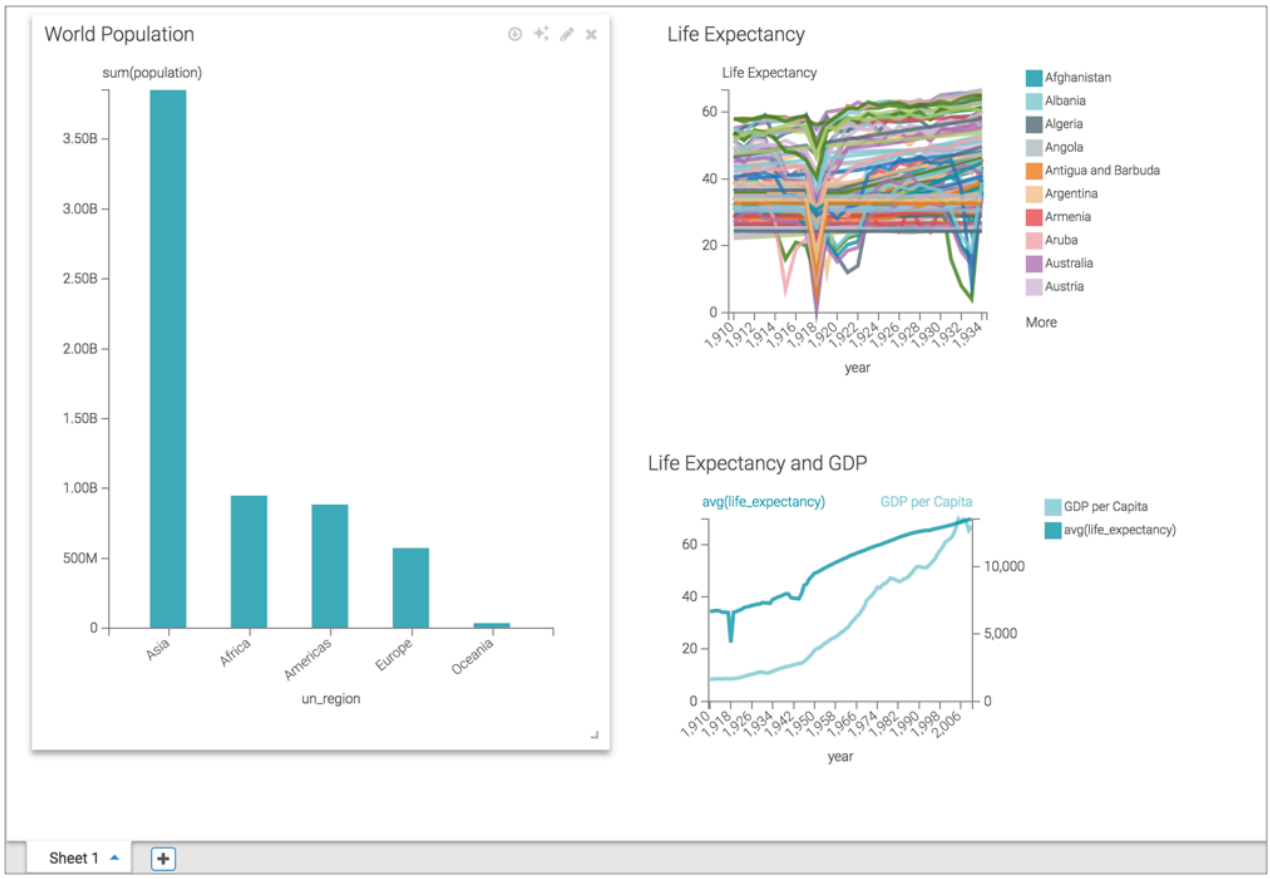

The following image shows the dashboard page background after changing the color.

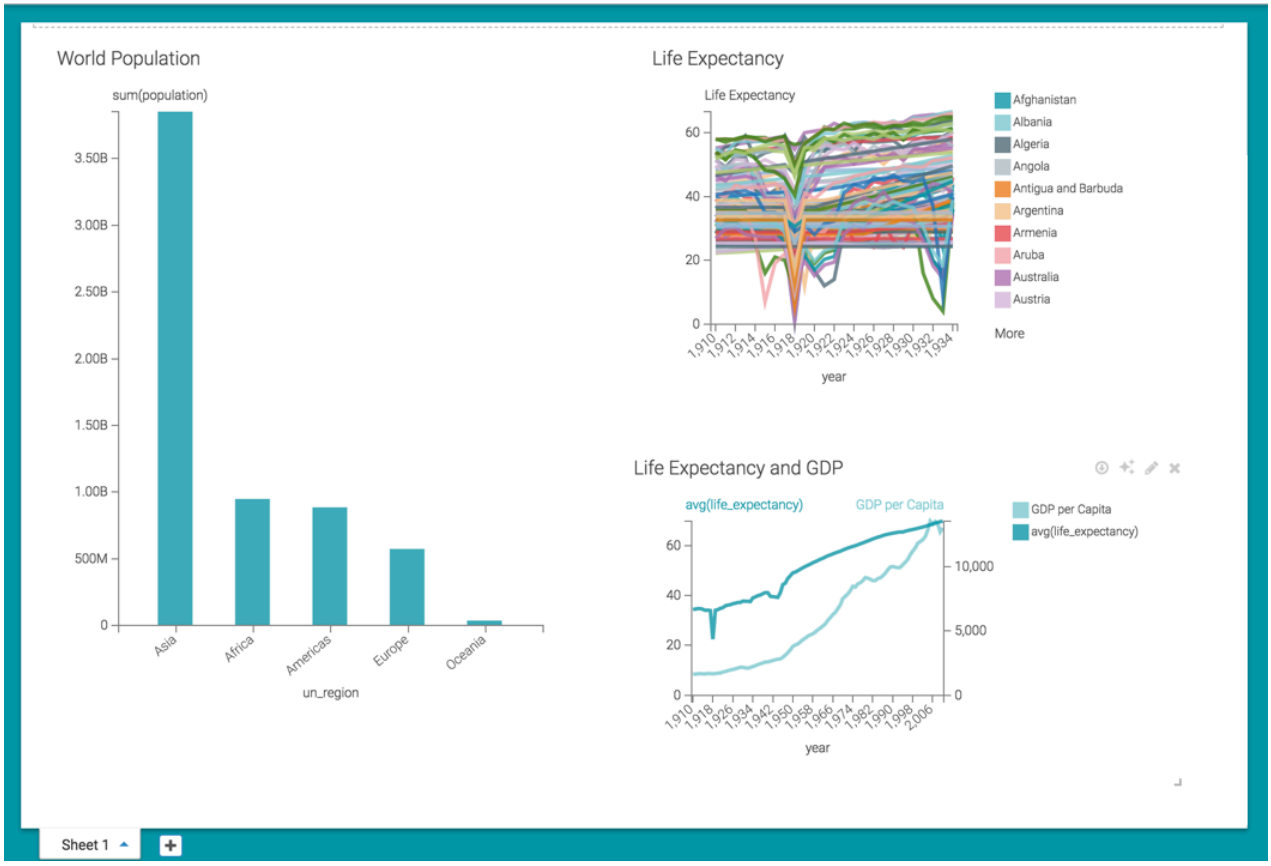

Dashboard titles styles

About this task

You can adjust several style options for dashboard titles in Cloudera Data Visualization dashboards. To get to the options in the Dashboard Title menu, follow these steps:

Procedure

1. On the right side of Dashboard Designer, click the Style menu.
2. In the Style menu, click Dashboard Title.
3. Select one of the options.

Dashboard titles font family

You can change the font of the dashboard title in Cloudera Data Visualization.

To change the font of the dashboard title, navigate to the dashboard Style menu, select the Dashboard Title menu, and then select a new font in the Font Family selector.

Dashboard titles font size

You can change the font size of the dashboard title in Cloudera Data Visualization.

To change the font size of the dashboard title, navigate to the dashboard Style menu, select the Dashboard Title menu, and then adjust the Font Size selector.

Dashboard titles font weight

You can change the font weight of the dashboard title in Cloudera Data Visualization.

To change the font weight of the dashboard title, navigate to the dashboard Style menu, select the Dashboard Title menu, and then select the new font weight in the Font Weight selector.

Dashboard titles text color

You can change the text color of the dashboard title in Cloudera Data Visualization.

To change the color of the text in the dashboard title, navigate to the dashboard Style menu, select the Dashboard Title menu, and then select a new color in the Text Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity). The default is no color.

Dashboard titles background color

You can change the background color of the dashboard title in Cloudera Data Visualization.

To change the color of the background in the dashboard title, navigate to the dashboard Style menu, select the Dashboard Title menu, and then select a new color in the Background Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity). The default is no color.

Dashboard titles alignment

You can change the alignment of the dashboard title in Cloudera Data Visualization.

To change the alignment of the dashboard title, navigate to the dashboard Style menu, select the Dashboard Title menu, and then select the desired Alignment option. By default, the title is left-aligned.

Hiding dashboard titles

You can hide a dashboard title in Cloudera Data Visualization.

To hide the dashboard title, navigate to the dashboard Style menu, select the Dashboard Title menu, and then select the Hide title in view mode option. By default, the title is on.

Dashboard subtitles styles

About this task

You can adjust several style options for dashboard subtitles in Cloudera Data Visualization dashboards. To get to the options in the Dashboard Subtitle menu, follow these steps:

Procedure

1. On the right side of Dashboard Designer, click the Style menu.
2. In the Style menu, click Dashboard Subtitle.
3. Select one of the options.

Dashboard subtitles font family

You can change the font of the dashboard subtitle in Cloudera Data Visualization.

To change the font of the dashboard subtitle, navigate to the dashboard Style menu, select the Dashboard Subtitle menu, and then select a new font in the Font Family selector.

Dashboard subtitles font size

You can change the font size of the dashboard subtitle in Cloudera Data Visualization.

To change the font size of the dashboard subtitle, navigate to the dashboard Style menu, select the Dashboard Subtitle menu, and then select a new font in the Font Size selector.

Dashboard subtitles text color

You can change the text color of the dashboard subtitle in Cloudera Data Visualization.

To change the color of the text in the dashboard subtitle, navigate to the dashboard Style menu, select the Dashboard Subtitle sub-menu, and then select a new color in the Text Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity). The default is no color.

Dashboard subtitles background color

You can change the background color of the dashboard subtitle in Cloudera Data Visualization.

To change the color of the background in the dashboard subtitle, navigate to the dashboard Style menu, select the Dashboard Subtitle sub-menu, and then select a new color in the Background Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity). The default is no color.

Dashboard subtitles alignment

You can change the alignment of the dashboard subtitle in Cloudera Data Visualization.

To change the alignment of the dashboard subtitle, navigate to the dashboard Style menu, select the Dashboard Subtitle sub-menu, and then select the desired Alignment option. By default, the title is left-aligned.

Dashboard filter styles

About this task

You can adjust several style options for dashboard filters in Cloudera Data Visualization.

To get to the options in the Filters menu, follow these steps:

Note: These styling choices affect the filter appearance when it is in the filter area of the dashboard. When you choose to place the filter inside the main dashboard area, it takes on the default style choices of the visuals in the dashboard.

Procedure

1. On the right side of Dashboard Designer, click the Style menu.
2. In the Style menu, click Filters.
3. Select one of the options.

Aligning dashboard filters

You can change the alignment of the dashboard filter controls in Cloudera Data Visualization. To show filters on the left of the dashboard, navigate to the dashboard Style menu, select the Filters menu, and select the Align filters on the left option. By default, the filter controls of a visual appear across the top.

Note: Cloudera recommends that you do not use this option with quick filters. For more information, see *Using quick filters*.

Related Information

Using quick filters

Using filter labels inline

You can change the position of the dashboard filter labels in Cloudera Data Visualization. To show labels inline with selectors, navigate to the dashboard Style menu, select the Filters menu, and select the Display labels inline option. By default, the filter labels appearover its selection controls.

Note: This option is only available when the dashboard has filters across the top.

Hiding filters line separator

You can hide the line separator in dashboard filetesr in Cloudera Data Visualization. To hide the separator, navigate to the dashboard Style menu, select the Filters menu, and select the Display line separator option. By default, the filter shows the line separator.

Note: This option is only available when the filter labels appear over the selection controls.

Hiding borders for filters in dashboards

You can hide filter borders in dashboards in Cloudera Data Visualization. To hide the filter borders, navigate to the dashboard Style menu, select the Filters menu, and select the Show borders around filters option. By default, the borders are on.

Changing dashboard filter font

You can change filter font family in dashboards in Cloudera Data Visualization. To change the font used in a dashboard filter, navigate to the dashboard Style menu, select the Filters menu, and select the Font Family selector.

Changing dashboard filter font size

You can change filter font size in dashboards in Cloudera Data Visualization. To change the size of the font used in a dashboard filter, navigate to the dashboard Style menu, select the Filters sub-menu, and select the Font Size selector.

Changing text color for dashboard filter

You can change the text color of the filter in dashboards in Cloudera Data Visualization. To change the color of the text in all filters of the dashboard, navigate to the dashboard Style menu, select the Filters menu, and select a new color in the Text Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity).

Changing selected text color for dashboard filter

You can change the color of the selected text in dashboard filters in Cloudera Data Visualization. To change the color of the selected text in all filters of the dashboard, navigate to the dashboard Style menu, select the Filters menu, and select a new color in the Selected Text Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity).

Changing container color for dashboard filter

You can change the color of the container in dashboard filters in Cloudera Data Visualization. To change the color of the containers in all filters of the dashboard, navigate to the dashboard Style menu, select the Filters menu, and select a new color in the Container Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity).

Changing background color for dashboard filter

You can change the background color of the dashboard filters in Cloudera Data Visualization. To change the color of the background in the filters of the dashboard, navigate to the dashboard Style menu, select the Filters menu, and

select a new color in the Background Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity).

Visual basics styles in dashboards

About this task

You can find several basic style options for dashboard visuals in Cloudera Data Visualization.

To get to the options in the Visual Basics menu, follow these steps:

Procedure

1. On the right side of Visual Designer, click the Style menu.
2. In the Style menu, click Visual Basics.
3. Select one of the options.

Visual basic font family in dashboards

You can change the font of the visuals of a dashboard in Cloudera Data Visualization. To change the font for the visuals in the dashboard, navigate to the dashboard Style menu, select the Visual Basics sub-menu, and then select a new font in the Font Family selector.

Visual basic font size in dashboards

You can change the font size of the visuals of a dashboard in Cloudera Data Visualization. To change the size of the font in the visuals in the dashboard, navigate to the dashboard Style menu, select the Visual Basics sub-menu, and then select a new font in the Font Size selector.

Visual basic background color in dashboards

You can change the font size of the visuals of a dashboard in Cloudera Data Visualization. To change the color of the background in the visual, navigate to the dashboard Style menu, select the Visual Basics sub-menu, and then select a new color in the Background Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity). The default is no color.

Visual border styles in dashboards

About this task

You can find several border style options for dashboard visuals in Cloudera Data Visualization.

To get to the options in the Visual Borders menu, follow these steps:

Procedure

1. On the right side of Dashboard Designer, click the Style menu.
2. In the Style menu, click Visual Borders.
3. Select one of the options.

Show borders on visuals in dashboards

You can show visual borders in dashboards in Cloudera Data Visualization. To show the borders for all visuals, navigate to the dashboard Style menu, select the Visual Borders sub-menu, and select the Show borders around visuals option. By default, the borders are off.

Note: The width of the border is 1px.

Change visual border color in dashboards

You can change the visual border color in dashboards in Cloudera Data Visualization. To change the color of borders for all visuals that appear on this dashboard, navigate to the dashboard Style menu, select the Border Color sub-menu, and select a new color in the Border Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity).

Note: This option is not available if the Show Borders is not on. When you turn on the Show Borders option, the color is not specified in the selector. However, the default that shows on the dashboard is light grey, or #E6E6E6.

Change visual border radius in dashboards

You can change the visual border radius in dashboards in Cloudera Data Visualization. To change the radius of borders on visuals or how curvy the corners are, navigate to the dashboard Style menu, select the Visual Borders sub-menu, and select a new color in the Border Radius option.

Note: This option is available only when Show Borders is enabled. When you turn on the Show Bordersoption, the default radius is 3. To make the corners curvier then the default, increase the radius setting. To set the borders to square corners, use the radius value of 0.

Visual titles styles in dashboard

About this task

You can adjust several style options for visual titles in Cloudera Data Visualization dashboards.

To get to the options in the Visual Title menu, follow these steps:

Procedure

1. On the right side of Dashboard Designer, click the Style menu.
2. In the Style menu, click Visual Title.
3. Select one of the options.

Visual title font family in dashboards

You can change the font of the visual title in Cloudera Data Visualization dashboards. To change the font of the text in titles of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Title menu, and then select a new font in the Font Family selector.

Visual title font size in dashboards

You can change the font size of the visual title in Cloudera Data Visualization dashboards. To change the font size of the text in titles of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Title menu, and then adjust the Font Size selector.

Visual title font weight in dashboards

You can change the font weight of the visual title in Cloudera Data Visualization dashboards. To change the font weight of the text in titles of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Title menu, and then select the new font weight in the Font Weight selector.

Visual title text color in dashboards

You can change the text color of the visual title in Cloudera Data Visualization dashboards. To change the color of the text in titles of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Title menu, and then select a new color in the Text Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity).

Visual title background color in dashboards

You can change the background color of the visual title in Cloudera Data Visualization dashboards. To change the color of the background in the title of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Title menu, and then select a new color in the Background Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity).

Visual title alignment in dashboards

You can change the alignment of the visual title in Cloudera Data Visualization dashboards. To change the alignment of the text in titles of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Title menu, and then select the desired Alignment option. By default, the title is left-aligned.

Visual subtitles styles in dashboard

About this task

You can adjust several style options for visual subtitles in Cloudera Data Visualization dashboards.

To get to the options in the Visual Subtitle menu, follow these steps:

Procedure

1. On the right side of Dashboard Designer, click the Style menu.
2. In the Style menu, click Visual Subtitle.
3. Select one of the options.

Visual subtitle font family in dashboards

You can change the font of the visual subtitle in Cloudera Data Visualization dashboards. To change the font of the text in subtitles of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Subtitle menu, and then select a new font in the Font Family selector.

Visual subtitle font size in dashboards

You can change the font size of the visual subtitle in Cloudera Data Visualization dashboards. To change the font size of the text in subtitles of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Subtitle menu, and then adjust the Font Size selector.

Visual subtitle text color in dashboards

You can change the text color of the visual subtitle in Cloudera Data Visualization dashboards. To change the color of the text in subtitles of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Subtitle

menu, and then select a new color in the Text Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity).

Visual subtitle background color in dashboards

You can change the background color of the visual subtitle in Cloudera Data Visualization dashboards. To change the color of the background in the subtitle of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Subtitle menu, and then select a new color in the Background Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity).

Visual subtitle alignment in dashboards

You can change the alignment of the visual subtitle in Cloudera Data Visualization dashboards. To change the alignment of the text in subtitles of all visuals of the dashboard, navigate to the dashboard Style menu, select the Visual Subtitle menu, and then select the desired Alignment option. By default, the title is left-aligned.

Visual axes and labels styles in dashboards

About this task

You can adjust several style options for visual axes and labels in Cloudera Data Visualization dashboards.

To get to the options in the Visual Axes and Labels menu, follow these steps:

Procedure

1. On the right side of Dashboard Designer, click the Style menu.
2. In the Style menu, click Visual Axes and Labels.
3. Select one of the options.
4. To style the visuals individually, de-select the Override visual style settings option, and specify the corresponding setting in each visual.

Axis label font size in dashboards

You can change the font size of the axis labels in visuals in Cloudera Data Visualization dashboards. To change the font size used by the axis labels in all visuals that appear on this dashboard, navigate to the dashboard Style menu, select the Visual Axes and Labels sub-menu, and make adjustments in the Axis Label Font Size selector.

To style the visuals individually, de-select the Override visual style settings option, and specify the corresponding setting in each visual.

Axis ticks font size in dashboards

You can change the font size of the axis ticks in visuals in Cloudera Data Visualization dashboards. To change the font size used by the axis ticks in all visuals that appear on this dashboard, navigate to the dashboard Style menu, select the Visual Axes and Labels sub-menu, and make adjustments in the Axis Ticks Font Size selector.

To style the visuals individually, de-select the Override visual style settings option, and specify the corresponding setting in each visual.

Visual label font size in dashboards

You can change the font size of the visual labels in Cloudera Data Visualization dashboards. To change the font size used by the labels in all visuals that appear on this dashboard, navigate to the dashboard Style menu, select the Visual Axes and Labels sub-menu, and make adjustments in the Label Font Size selector.

To style the visuals individually, de-select the Override visual style settings option, and specify the corresponding setting in each visual.

Visual label title text color in dashboards

You can change the text color of a trelllis used in visuals in Cloudera Data Visualization dashboards. To change the font size used of a trellis in all visuals that appear on this dashboard, navigate to the dashboard Style menu, select the Visual Axes and Labels sub-menu, and make adjustments in the Label Text Color selector. You can choose from one of the standard system colors, specify a hexadecimal or RGB color, or select one in the continuous color palette (you can also adjust color opacity).

Note: This style option overrides the selection at the level of the individual visual. To style the visuals individually, de-select the Override visual style settings option, and specify the corresponding setting in each visual.

Visual trellis font size in dashboards

You can change the font size of a trellis in visuals in Cloudera Data Visualization dashboards. To change the font size used of a trellis in all visuals that appear on this dashboard, navigate to the dashboard Style menu, select the Visual Axes and Labels sub-menu, and make adjustments in the Trellis Font Size selector.

To style the visuals individually, de-select the Override visual style settings option, and specify the corresponding setting in each visual.

Visual legend font size in dashboards

You can change the font size of a legend in visuals in Cloudera Data Visualization dashboards. To change the font size used of the legend in all visuals that appear on this dashboard, navigate to the dashboard Style menu, select the Visual Axes and Labels sub-menu, and make adjustments in the Legend Font Size selector.

Tooltips in dashboards

About this task

You can adjust several style options for tooltips in Cloudera Data Visualization dashboards.

To get to the options in the Visual Axes and Labels menu, follow these steps:

Procedure

1. On the right side of Dashboard Designer, click the Style menu.
2. In the Style menu, click Tooltips.
3. Select one of the options.

Tooltip font family in dashboards

You can change the font of the tooltips in Cloudera Data Visualization dashboards. To change the font of the tooltips in all visuals of the dashboard, navigate to the dashboard Style menu, select the Tooltip sub-menu, and then select a new font in the Font Family selector.

Tooltip font size in dashboards

You can change the font size of the tooltips in Cloudera Data Visualization dashboards. To change the font size of the tooltips used in all visuals of the dashboard, navigate to the dashboard Style menu, select the Tooltip sub-menu, and adjust the Font Size selector.

Adding custom styles to dashboards

Cloudera Data Visualization supports several options for custom styling on dashboards.

About this task

To change the options for custom styles on dashboards, follow these steps:

Procedure

1. On the right side of Dashboard Designer, click the Custom Style menu.
2. In the Style menu, choose from the available custom style options.

Adding CSS classes to dashboards

You can add CSS classes to Cloudera Data Visualization dashboards.

To add CSS classes to a dashboard, navigate to the dashboard Custom Style menu, and enter the name(s) of classes in the text box for the option CSS classes for this dashboard.

Adding included styles to dashboards

You can add defined styles to all visuals in Cloudera Data Visualization dashboards.

To add previously defined styles to all visuals in a dashboard, navigate to the dashboard Custom Style menu, click the Add Style button, select one or more existing styles from the menu, and click Add.

You can also search or sort the custom styles in the Pick Custom CSS interface.

Searching custom styles in dashboards

Cloudera Data Visualization enables you to search custom styles in dashboards by Name and Modified By fields. You can also search by the content of CSS and Javascript code.

About this task

When you start typing characters in the search text box, the search results display the exact match on the top, followed by the closest matches. You must type at least two characters to start the search.

To demonstrate how to search custom styles in dashboards, follow these steps:

Procedure

1. Navigate to the Custom Style menu, click the Add Style button.

   The Pick Custom CSS interface appears.
2. To demonstrate how to search the list of custom styles, enter Pie in the Search text box.
3. To search the list of custom styles by JavaScript code, follow these steps:
   a) Click the (pencil)icon next to the Pie Chart Legend Label to view the JavaScript code of this style.
   b) Enter Not Applicable in the Search text box.

Sorting custom styles in dashboards

Cloudera Data Visualization enables you to sort custom styles in dashboards by ID, Name, Last Updated, and Modified By fields. You can sort in ascending or descending order.

About this task

Note: You cannot sort by the Description field.

To demonstrate how to sort custom styles in dashboards, follow these steps:

Procedure

1. Navigate to the Custom Style menu, click the Add Style button.
   The Pick Custom CSS interface appears.
2. Click the Name field.
   When you hover over a field, notice that a (hand icon) appears only on the fields that you can sort.
3. To sort the field in descending order, click the Name field again.

Adding inline CSS code to dashboards

You can apply simple formatting to visuals with CSS code snippets in Cloudera Data Visualization dashboards.

To add simple formatting to all the visuals in the dashboard, navigate to the dashboard Custom Style menu, and enter a CSS code snippet under the Enter in-line CSS option. You can select the Autocomplete option, too.

To apply this operation at the level of a specific visual, see *Adding inline CSS code to visuals*.

Related Information

Adding inline CSS code to visuals

Resetting dashboard style

You can reset the styling of a dashboard to its default settings.To change the style back, follow these steps.

Procedure

1. Open the dashboard in Edit mode.
2. On the right side of the Dashboard Designer, click the Style menu.

3. In the Style menu, click at the top right corner to reset the dashboard to its original style.

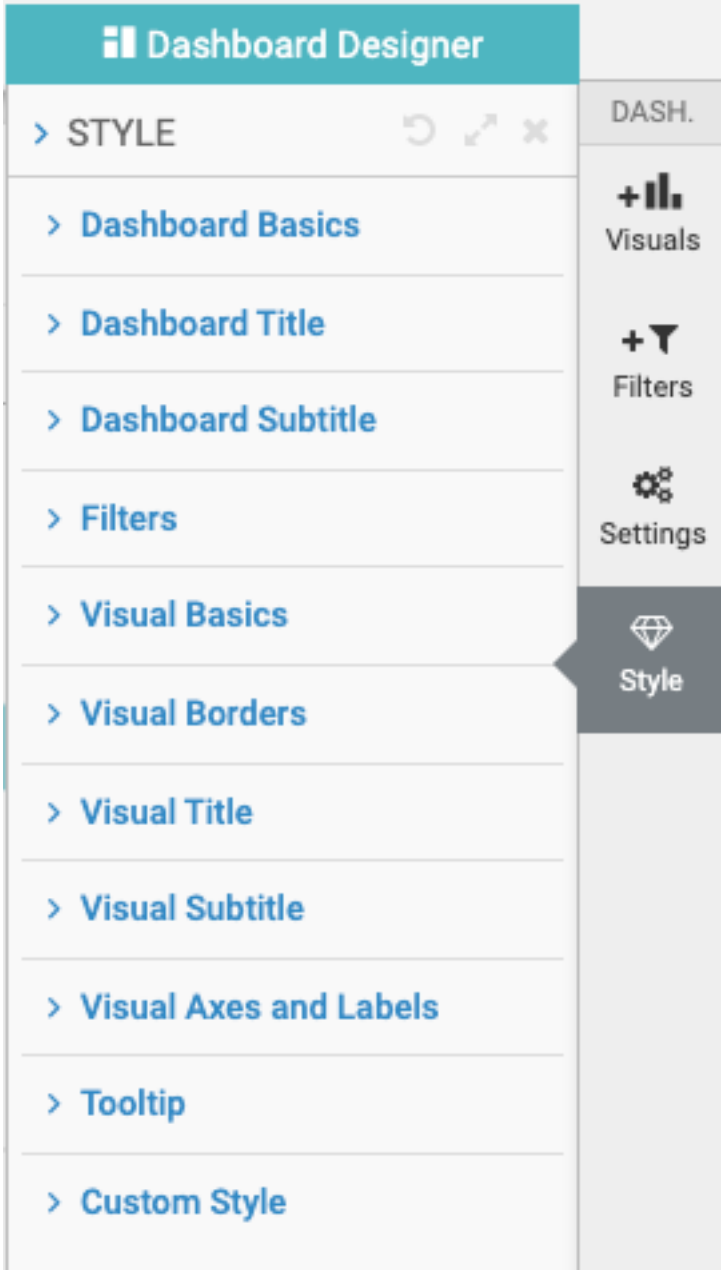

4. Save the dashboard to preserve the new style.

Creating a dashboard with multi-page table visuals

This customization example shows you how to build an embedded dashboard that prints the full contents of a table as part of a dashboard print. This dashboard contains visuals at the top, and a table at the bottom. The table prints as many pages as necessary to display all rows. Table headers repeat at the top of each page. Several custom styles are used to achieve the desired outcome.

About this task

Complete these tasks to develop a dashboard with a large table that prints over multiple pages.

Procedure

1. Create the custom table style.

   These steps show how to create and add a dashboard style for printing long tables over multiple pages. This table must be the last visual in the dashboard.

   a) Create a new style, named Print PDF table - CSS/JS.

      For help on how to create a new style, see *CSS custom styles*. The optional description for this style can be as follows:

      ```
      Add this Style to the table to be printed on multiple pages.
      The table must be the last visual in the dashboard.
      ```

   b) In the CSS tab, add the following code.

      You can download it as multi-page-table.css.

      ```
      @charset "UTF-8";
      /* CSS Document */

      /* To disable the known csslint warning for this file. */
      /*csslint
      box-sizing: false,
      important: false,
      adjoining-classes: false,
      overqualified-elements: false,
      box-model: false,
      known-properties: false
      */

      /* screen table style */
      .print-pdf-table table.table-bordered.dataTable {
        border-collapse: collapse !important;
      }

      .print-pdf-table .dataTables_scrollBody table tbody tr td {
        height: 0.53cm !important;  /* table row height */
        box-sizing: border-box;
        font-size: 10px;
      }

      @media print {
        @page {
          size: letter landscape;
          margin: 1.27cm 1.27cm 1.27cm 1.27cm;
        }

        .print-pdf-table .dataTables_scrollBody table thead tr th {
          border-bottom: solid #000 2px !important;  /* bottom border of table
       header */
          font-size: 12px;
          font-weight: bold;
          height: 1.48cm !important;  /* table header height */
          line-height: 1;
          padding: 12px 8px !important;
        }
        .print-pdf-table .dataTables_scrollBody table thead tr th > div {
          height: 1.48cm !important;  /* table header height */
          display: flex;
          justify-content: center;
      ```

```
    flex-direction: column;
  }

  .print-pdf-table table.dataTable.table-viz thead > tr > th.sorting {
    padding: inherit;
    padding-right: 8px; /* remove extra space where sort icon is disp
layed */
  }

  .print-pdf-table .dataTables_scrollBody table tbody tr td {
    height: 0.53cm !important;  /* table row height */
    box-sizing: border-box;
    font-size: 10px;
  }

  .gs-w.print-pdf-table .dataTables_scrollBody {
    max-height: none !important;
  }

/* this class prints the table on the second page */
/* remove to print normally */
  .gs-w.print-pdf-table {
     position: absolute;
     top: 40px; /* add landscape page height + top margin */
  }

  .gs-w.print-pdf-table {
    height: unset !important;
  }

  .print-pdf-table .dash-viz-wrapper {
    display: contents;
  }
  .print-pdf-table .arc-viz-container {
    overflow: unset !important;
  }

  .main-app.arc-app-container.view-mode {
    display: table;
    background-color: transparent;
  }

  .control-container {
    background-color: transparent;
  }

  .arc-dashboard-title-container {
    padding-top: 1cm;  /* adjust if want more or less space at the top
of the first page  */
  }

  .builder-container.view-mode,
  .builder-container-app,
  .builder-container {
    display: table;
  }

  .main-app-view,
  .main-view {
    overflow: visible !important;
    top: 0;
  }

  .main-view.view-mode {
```

```
    top: 0;
  }

  .print-pdf-table table th,
  .print-pdf-table table td {
    border: 1px solid #bbb !important;  /* border color */
  }

  .print-pdf-table table.table-bordered.dataTable {
    border-collapse: collapse !important;
  }

  .print-pdf-table  .arc-viz-title-container {
    display: none;  /* hide table title to print properly */
  }
  .print-pdf-table .dash-viz-wrapper {
    padding: 0;
  }

  .print-pdf-table .dataTables_scrollHead {
    height: 0;
  }

  .print-pdf-table .table {
    border-width: 0;
  }
  .print-pdf-table thead {
    break-inside: avoid;
    display: table-header-group;
  }

  .print-pdf-table table {
    page-break-after: always;
  }
  .print-pdf-table tfoot {
    break-inside: avoid;
    display: table-header-group;
  }

/* hide elements when printing  */
  #extension-print-btn,  /* extension buttons */
  #toggle-filter-container /* filter collapse icon */
  {
    display: none;
  }

}
```

c) In the JS tab, add the following code.

You can download it as multi-page-table.js.

```
return function() {
    var f = function() {};
    f.version = "1";
    f.afterDraw = function() {
      var $sel = $('#' + arcapi.chartId());
      $sel.closest('.gs-w').addClass('print-pdf-table');
    };
   return f;
}();
```

d) Save the new style.

2. Create the dashboard.
   a) Create a dashboard on your target dataset. For more information, see *Creating a dashboard*.
   b) Add several visuals to the dashboard, such as KPIs, summary tables, pie charts, and so on.
   c) Add all necessary filters to the dashboard.
3. Specify dashboard general settings to achieve correct layout in dashboard.
   a) On the right side of Visual Designer, click the Settings menu.
   b) In the Settings menu, click General to open that menu.
   c) In the General menu, clear the Dashboard Width specification. For more information, see *Changing dashboard width*.
4. Specify dashboard downloads settings to print the dashboard using the PDF options.
   a) On the right side of Visual Designer, click the Settings menu.
   b) In the Settings menu, click Downloads to open that menu.
   c) In the Downloads menu, make the following selections and changes:
      - Enable Download as Image/PDF
      - Enable print style PDF
      - Set "Print PDF Scale" to 0
      - Set "PDF Page Size" to Letter
      - Set "PDF Page Orientation" to Landscape
      - Enable PDF Header and Footer
5. Add time stamp.
   a) Create the extension visual using the following definition:

```
<span style="color:#aaa">Today:</span>
<span id="tdate"></span>

<script>
$(document).ready(function() {
  var today = new Date();
  $('#tdate').html(
  $.datepicker.formatDate('MM dd, yy', today));
});
</script>
```

   When you add this extension visual to the dashboard, it will look similar to this (in View mode).

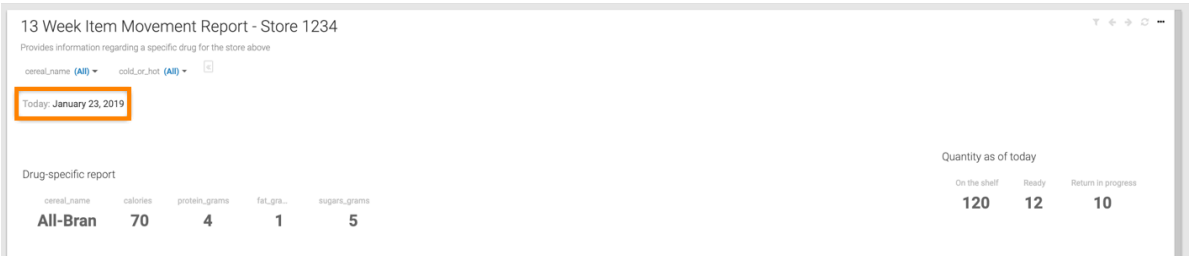

6. Add capabilities to print dashboard and download PDF.

   Use the following definition to create the buttons.

   - Print button with an icon, Print
   - Download PDF button with an icon, Download PDF

```
<div id="extension-print-btn">
  <button class="btn btn-xs btn-primary"
  onclick="window.print();return false;">
  <span class="fa fa-print"></span> Print</button>
```

```
  <button class="pdf-download btn btn-xs btn-primary"
  onclick="$('.viz-params-more').find('[data-arc-imageformat=&quot;pdf-cur
rent&quot;]')[0].click();”>
  <span class="fa fa-download"></span> Download PDF</button>
  <div class="image-dropdown-submenu"></div>

</div>
```

When you add this extension visual to the dashboard, it would look similar to this (in View mode).

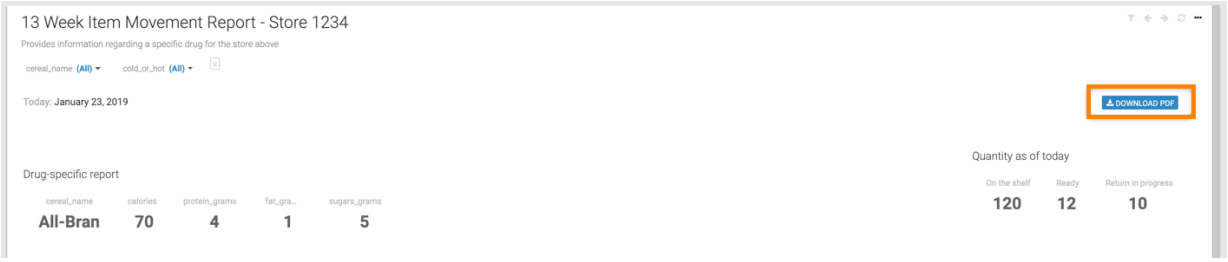

7. Create and customize the table visual.

These steps show you how to create and customize the main table visual. This visual should be the last (lowest) one in the dashboard.

a) Create the table visual on the target dataset. For more information, see *Tables*.
b) On the right side of Visual Designer, click the Style menu.
c) In the Style menu, click Table Style.
d) In the Table Style menu, make the following selections and changes:

- *Choose the first table template*
- *Table Header Alignment* set to Align with contents
- *Enable Full Table Width*
- *Wrap Column Contents*

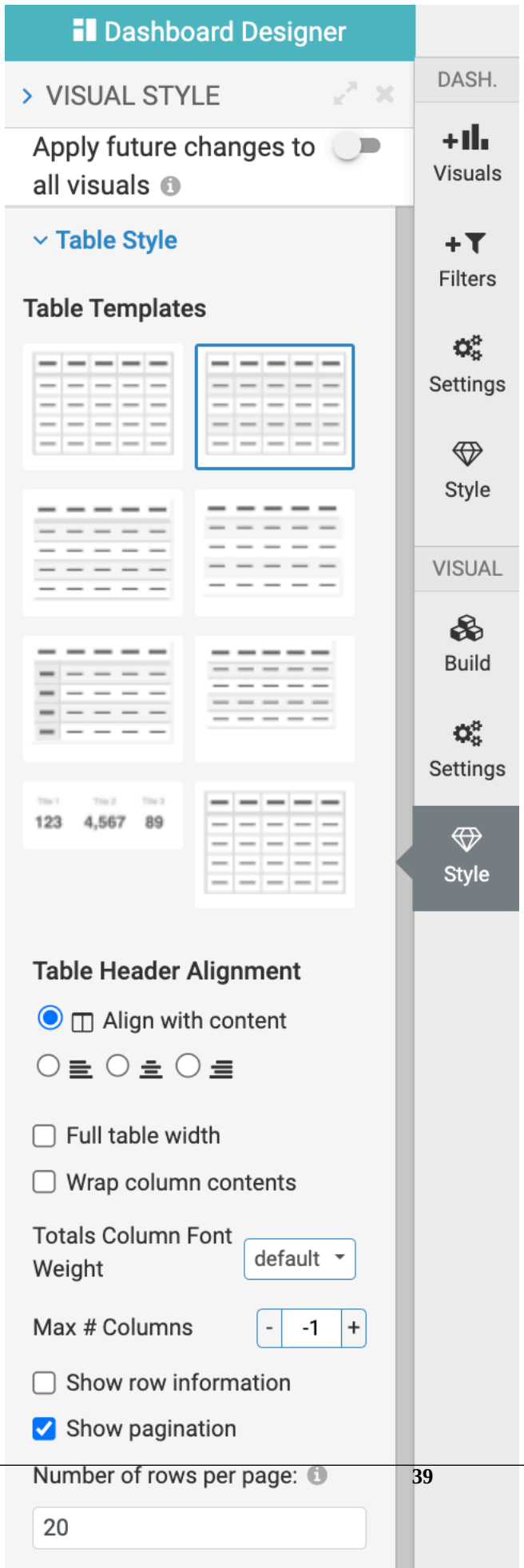
Dashboard Designer
VISUAL STYLE
Apply future changes to all visuals
Table Style
Table Templates
123
4,567
89
Table Header Alignment
Align with content
Full table width
Wrap column contents
Totals Column Font Weight
default
Max # Columns
-1
Show row information
Show pagination
Number of rows per page:
20
DASH.
Visuals
Filters
Settings
Style
VISUAL
Build
Settings
Style

e) Switch to the Custom Style menu, and make the following changes:

- Add the style arc-print-last-table to the visual. For more information, see *Adding custom styles to visuals* .
- Add the style we created earlier, Print PDF table - CSS/JS, to the visual. For more information, see *Adding included styles to visuals* .

Note: The ID of your custom style may be different. Cloudera Data Visualization generates sequential ID on each instance.

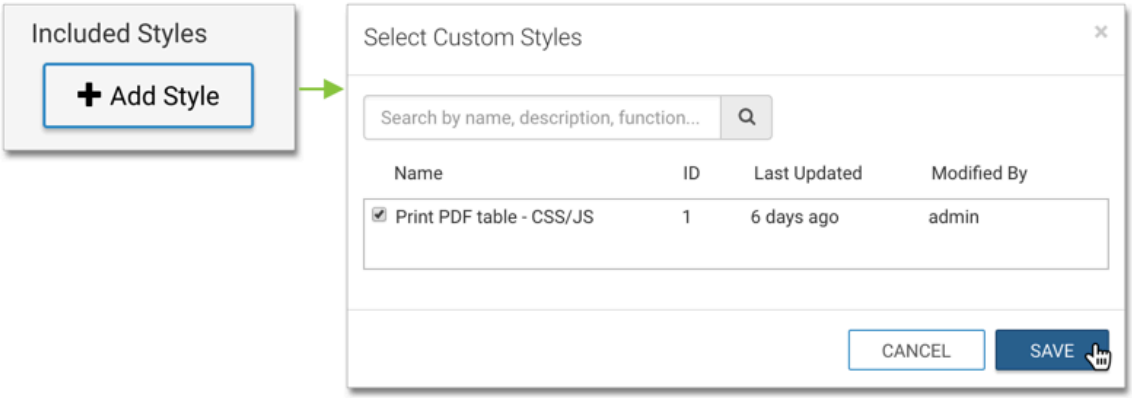

Your Custom Style menu selections should look something like this:

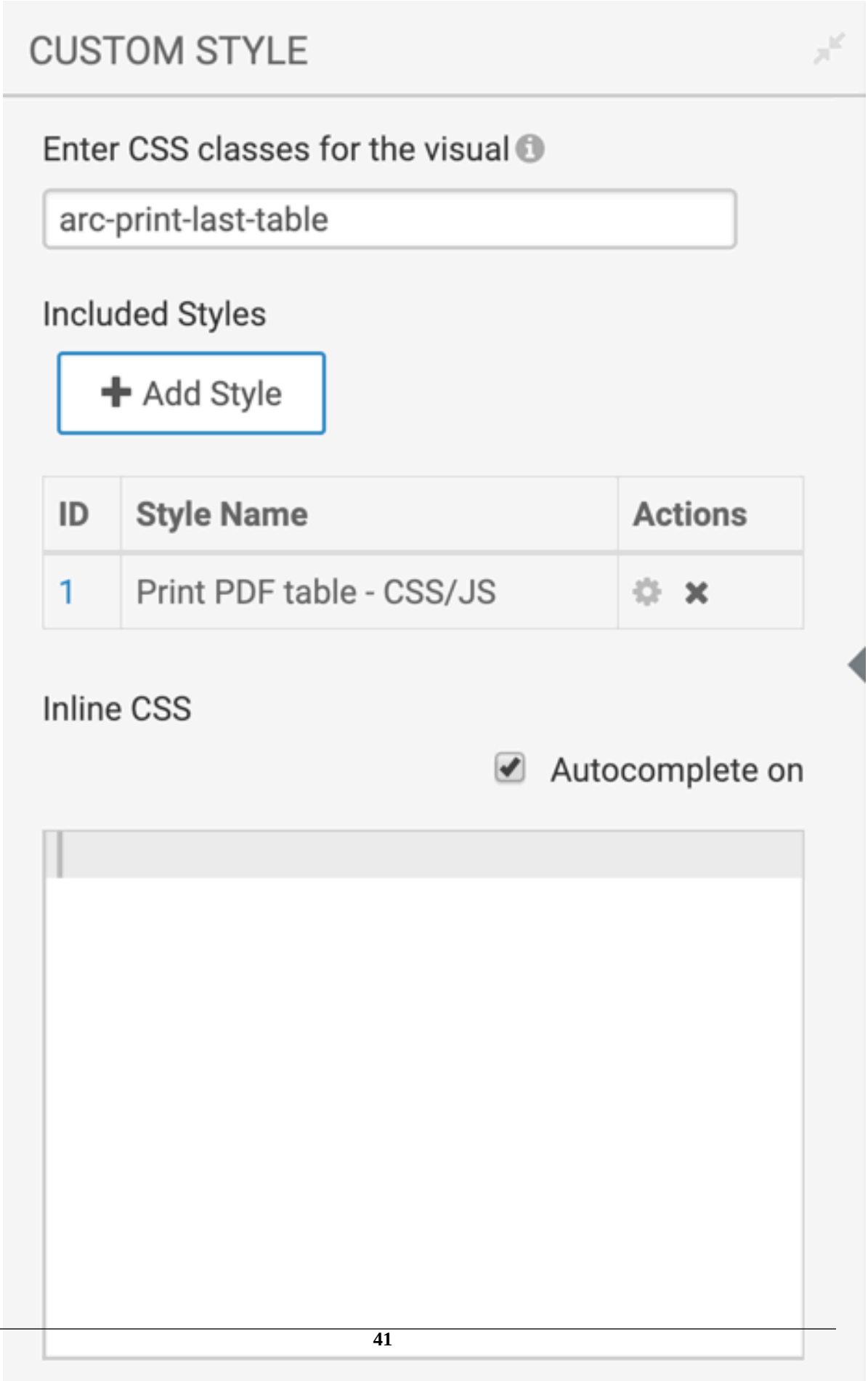
CUSTOM STYLE
Enter CSS classes for the visual
arc-print-last-table
Included Styles
Add Style
ID
Style Name
Actions
1
Print PDF table - CSS/JS
Inline CSS
Autocomplete on

f) When you finished configuring the table, it would look similar to this in View mode of the dashboard. Ensure that this table visual is the last item on the dashboard.

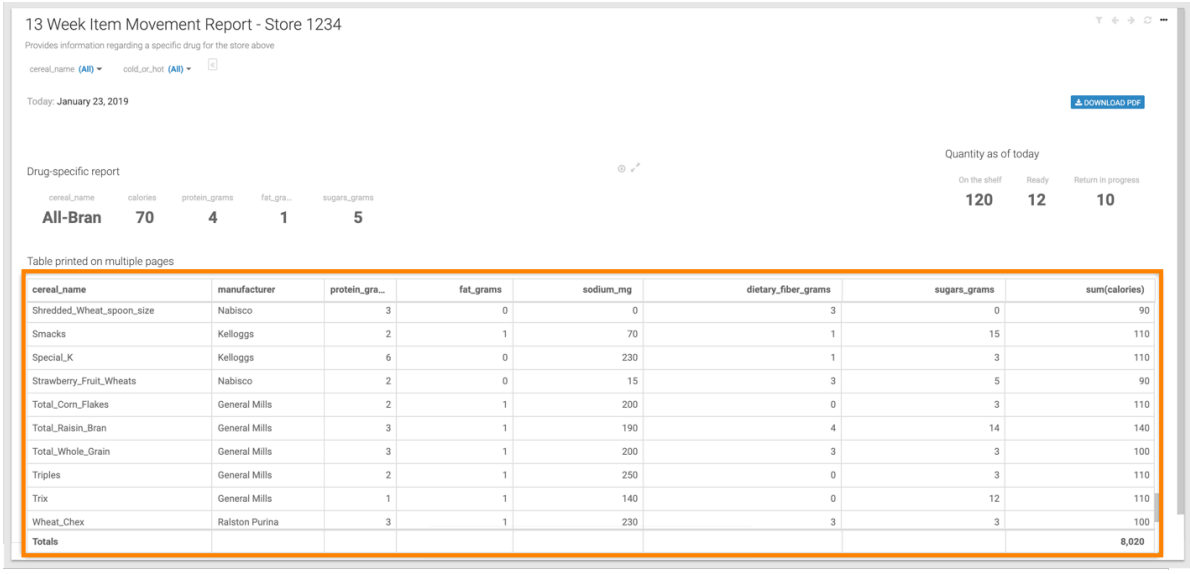

cereal_name	manufacturer	protein_gra...	fat_grams	sodium_mg	dietary_fiber_grams	sugars_grams	sum(calories)
Shredded_Wheat_spoon_size	Nabisco	3	0	0	3	0	90
Smacks	Kelloggs	2	1	70	1	15	110
Special_K	Kelloggs	6	0	230	1	3	110
Strawberry_Fruit_Wheats	Nabisco	2	0	15	3	5	90
Total_Corn_Flakes	General Mills	2	1	200	0	3	110
Total_Raisin_Bran	General Mills	3	1	190	4	14	140
Total_Whole_Grain	General Mills	3	1	200	3	3	100
Triples	General Mills	2	1	250	0	3	110
Trix	General Mills	1	1	140	0	12	110
Wheat_Chex	Ralston Purina	3	1	230	3	3	100
Totals							**8,020**

8. Print and download.

a) After saving the dashboard, switch to View mode.

b) Click PRINT on the dashboard, and then follow system prompts to complete the printing process.

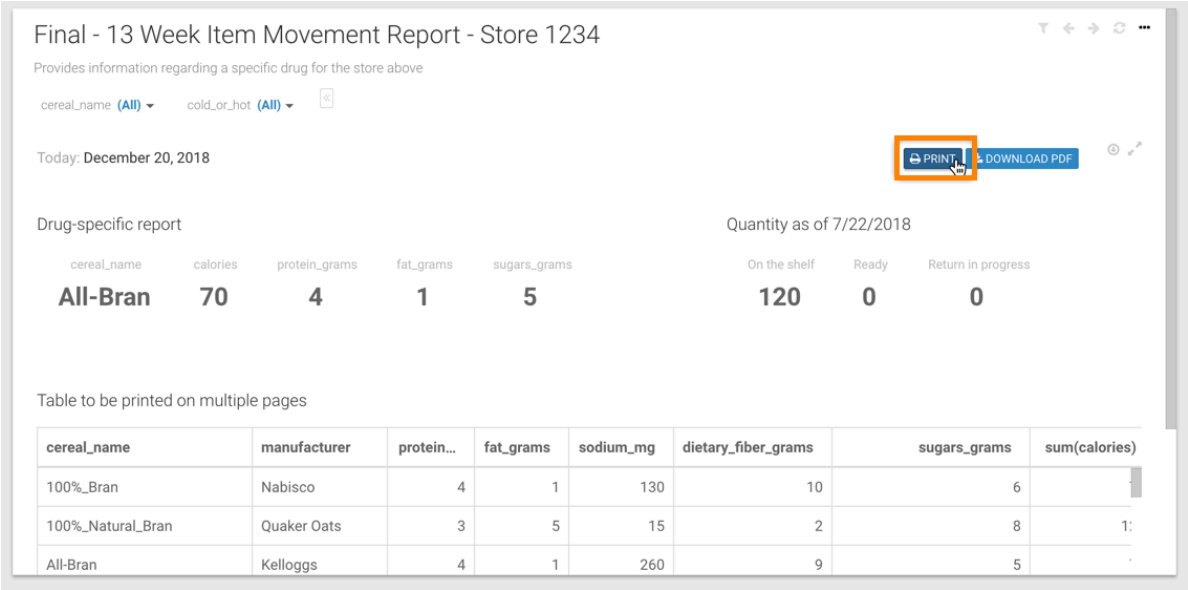

c) Click DOWNLOAD PDF on the dashboard, and then follow system prompts to complete the PDF download of the dashboard.

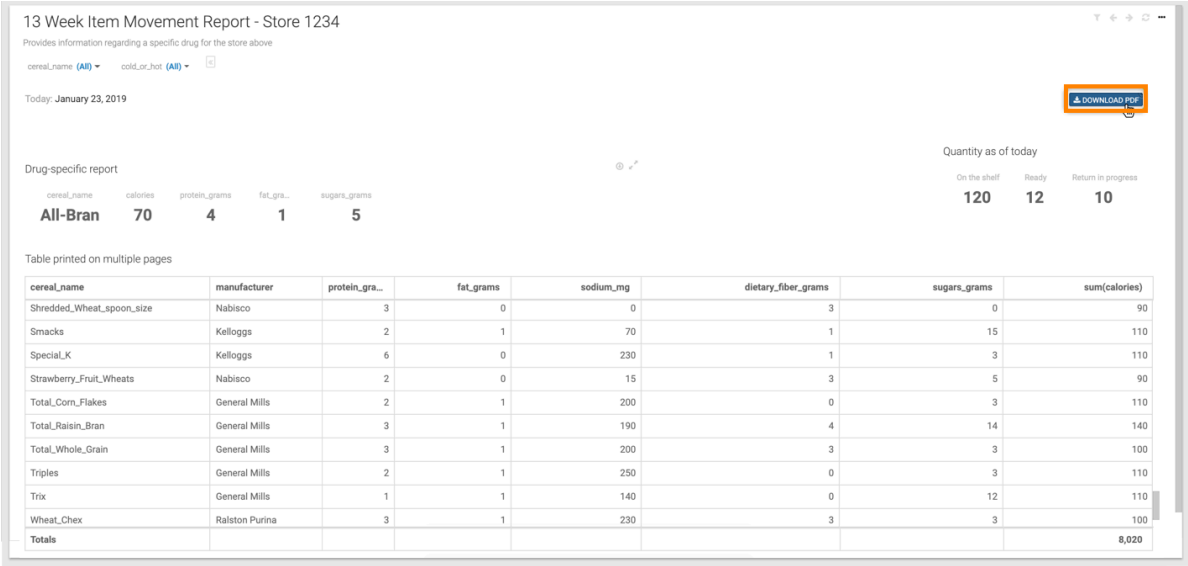

Related Information

CSS custom styles

Creating a dashboard

Changing dashboard width

Tables

Adding custom styles to visuals

Adding included styles to visuals

Cloudera Data Visualization 7.2.9

Customizing Field Properties

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Adding and removing aggregates 5

Customizing date/time functions 6

Adjusting text display options 8

Customizing analytic functions 9

Changing field type 10

Customizing order and top K 14

Customizing enter/edit expressions 15

Customizing calculated display 19

Enabling expansion 22

Customizing display format 23

- Real number display format 23
- Integer display format 24
- Percentage display format 25
- Scientific display format 27
- Currency display format 30
- Date/time display format 31
- Using custom display configurations 34
  - Using custom display formats for large numbers 38
- Customizing display format using Javascript 41

Using alias 43

Changing column position 44

Customizing description 46

Duplicating an item..... 49

Adding and removing aggregates

In visuals, when you add a field to one of the shelves in Edit mode, you can also configure the properties for that field. Aggregate functions calculate a return value across all the items in a result set. Cloudera Data Visualization provides optional aggregation operations on data values, such as summation, string concatenation, and common statistical measurements of data.

About this task

To add an aggregate function of change the selection for aggregates, follow these steps:

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.
2. Under Field Properties, click to expand the Aggregates menu.

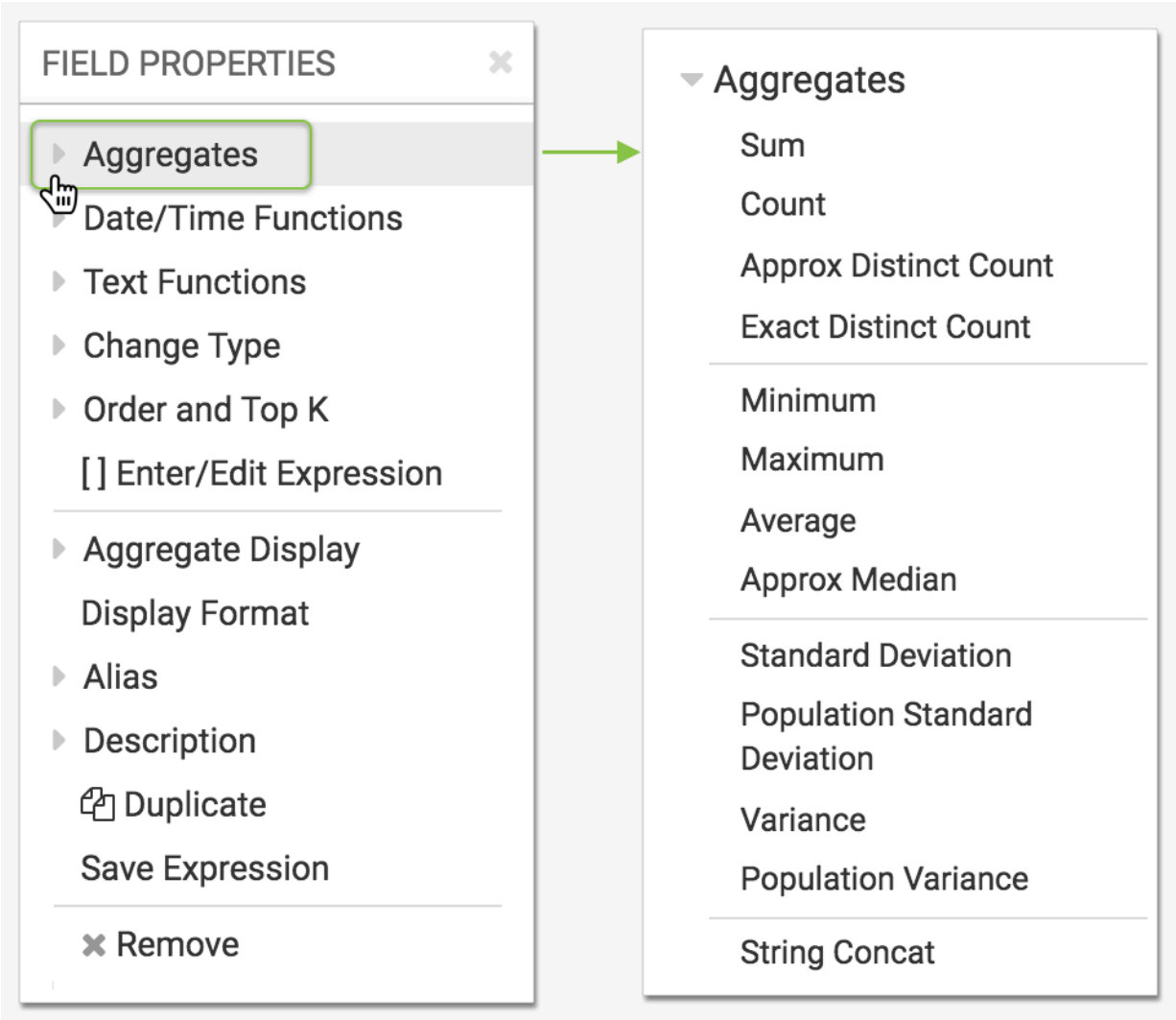

3. From the Aggregates menu, select one of the options.

   The available aggregation functions depend on the connection type and data type. A check mark precedes the selected aggregate function.

4. Click Refresh Visual to see the changes.

   Fields that use an aggregation function have a green dot on the Aggregates menu.

   To remove an aggregate from a field, click the selected aggregate function. It removes the check mark.

Customizing date/time functions

Cloudera Data Visualization has several built-in transformations for date and time that can be applied before rendering visual charts.

About this task

To use a date and time functions, follow these steps:

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.

2. Under Field Properties, click to expand the Date/Time Functions menu.

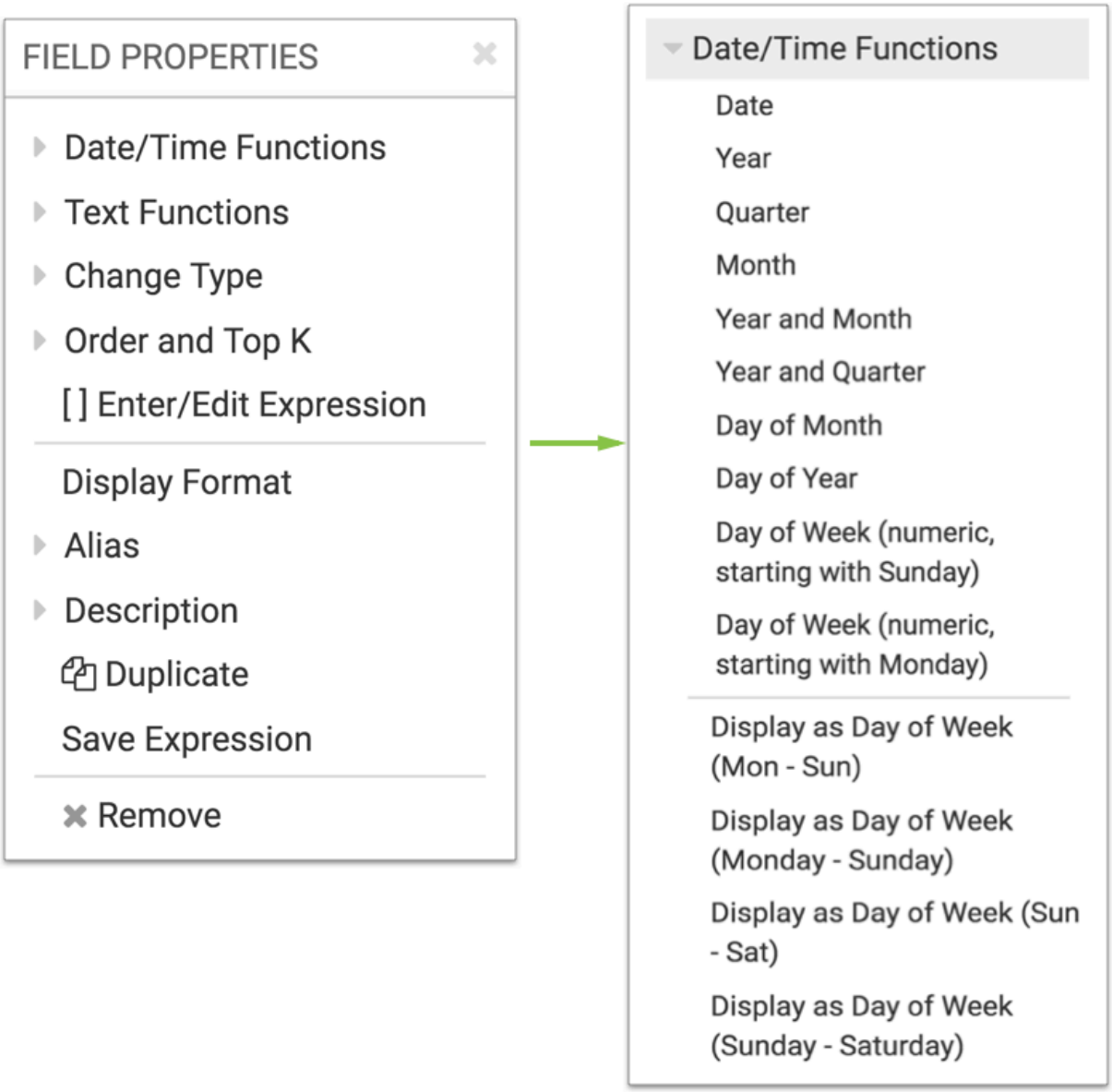

3. From the Date/Time Functions menu, select one of the options.
   - Date
   - Year
   - Month
   - Year and Month
   - Year and Quarter
   - Day of Month
   - Day of Year
   - Day of Week (numeric, starting with Sunday)
   - Day of Week (numeric, starting with Monday)
   - Display as Day of Week (Mon - Sun)
   - Display as Day of Week (Monday - Sunday)
   - Display as Day of Week (Sun - Sat)
   - Display as Day of Week (Sunday - Saturday)

   A check mark precedes the selected aggregate function.
4. Click Refresh Visual to see the changes.

Adjusting text display options

Cloudera Data Visualization enables you to adjust the text display options for field values before rendering visuals.

About this task

To use text functions, follow these steps:

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.

2. Under Field Properties, click to expand the Text Functions menu.

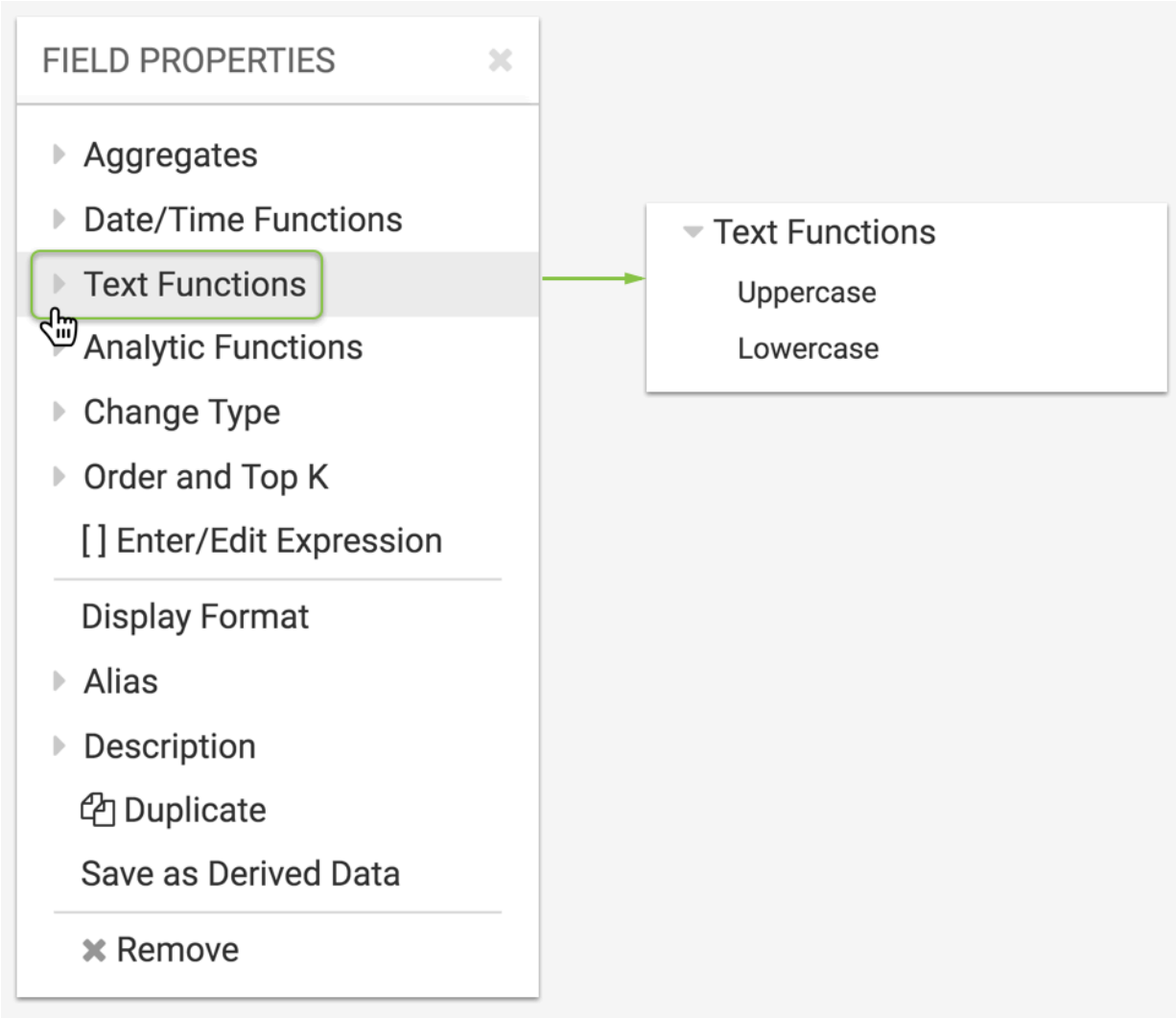

3. From the Text Functions menu, select one of the options.
   - Uppercase converts text to upper case; returns the value of upper([field])
   - Lowercase converts text to lower case; returns the value of lower([field]).

   A check mark precedes the selected aggregate function.
4. Click Refresh Visual to see the changes.

   To the text mask from a field, click on the selected mask. It removes the check mark.

Customizing analytic functions

Cloudera Data Visualization uses analytic functions in advanced analysis. You can move the app menu from the top to the left of the App Designer interface.

See the full description of analytic functions in the *Advanced Analysis* section.

Changing field type

Cloudera Data Visualization visuals enable you to change the type of the field as a cast operation of the form cast(expr as type), directly on the fields of the visual. Similarly, you can ensure that a field on a Geo shelf can be used even if it does not automatically satisfy Geo naming conventions.

About this task

To change the field type, follow these steps:

Procedure

1. Click the field you plan to modify, to open the FIELD PROPERTIES menu.

2. Under FIELD PROPERTIES, click to expand the Change Type menu.

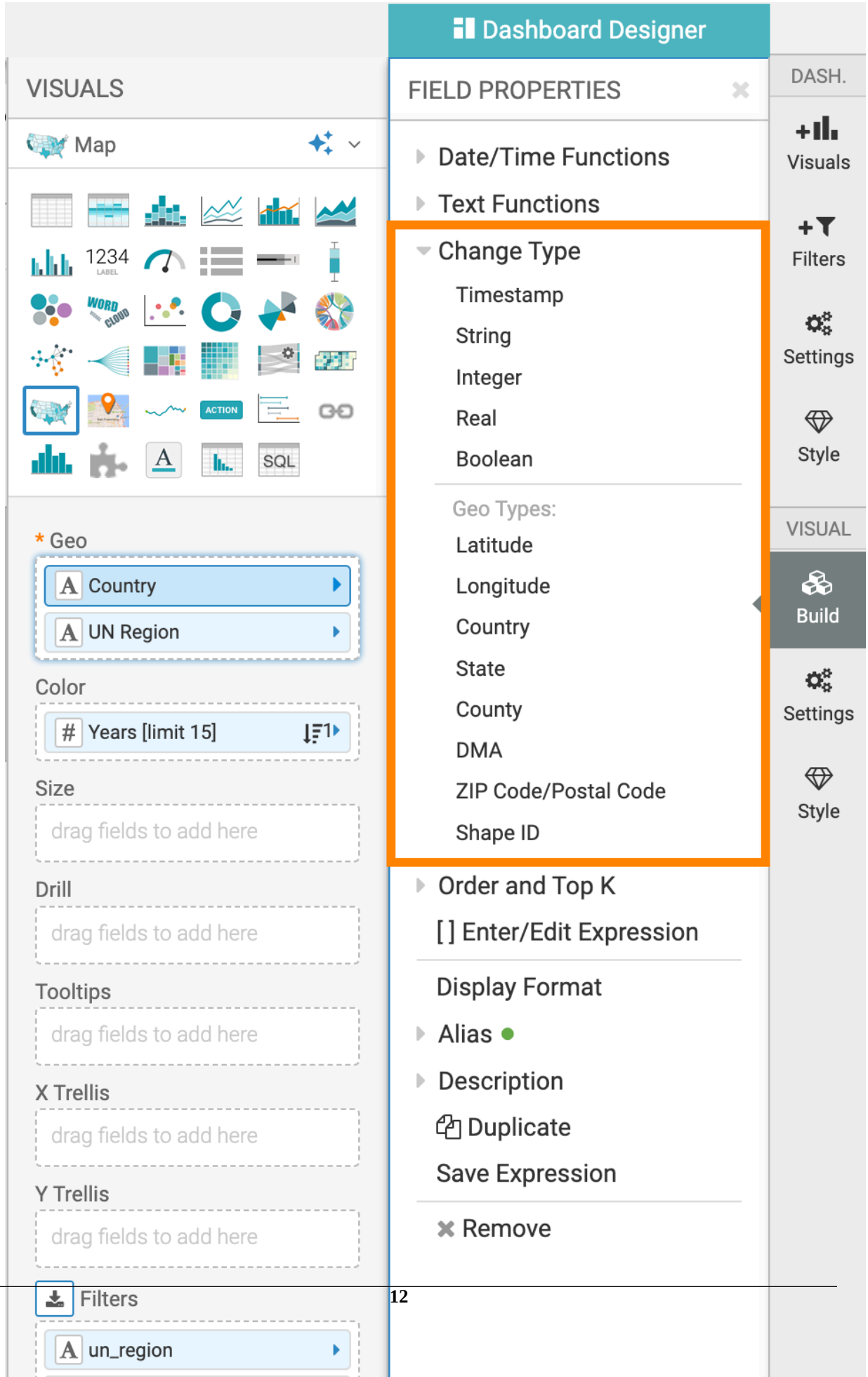
Dashboard Designer
VISUALS
Map
* Geo
Country
UN Region
Color
Years [limit 15]
Size
drag fields to add here
Drill
drag fields to add here
Tooltips
drag fields to add here
X Trellis
drag fields to add here
Y Trellis
drag fields to add here
Filters
un_region
FIELD PROPERTIES
Date/Time Functions
Text Functions
Change Type
Timestamp
String
Integer
Real
Boolean
Geo Types:
Latitude
Longitude
Country
State
County
DMA
ZIP Code/Postal Code
Shape ID
Order and Top K
[ ] Enter/Edit Expression
Display Format
Alias
Description
Duplicate
Save Expression
Remove
DASH.
Visuals
Filters
Settings
Style
VISUAL
Build
Settings
Style

3. From the Change Type menu, select one of the options.

Timestamp

Timestamp casts data to TIMESTAMP, cast([field] as TIMESTAMP) of the form year-month-Date hour:minute:second.

For example: 2015-06-09 14:15:45.

String

String casts data as STRING, cast([field]as STRING).

Integer

Integer casts data to BIGINT, cast([field]as BIGINT).

Real

Real casts data to REAL, cast([field]as REAL).

Note: REAL is an alias for DOUBLE.

Boolean

Boolean casts data to BOOLEAN, cast([field]as BOOLEAN).

Geo Types

Geo Types is a grouping of types that make the field usable on the Geo shelf of Map and Interactive Map visuals.

- Latitude makes the field useable as a latitude measurement
- Longitude makes the field useable as a longitude measurement
- Country makes the field useable as a country region
- State makes the field useable as a state region, for United States of America
- County makes the field useable as a county region, for United States of America
- DMA makes the field useable as a Nielsen Designated Market Area Region, for United States of America
- ZIP Code/Postal Code makes the field useable as a defined zipcode region, for United States of America or postal code for other countries
- Shape ID makes the field useable as a shape ID

Note that after a field becomes useable for geo visuals, it is marked on the shelf by the (globe) icon.

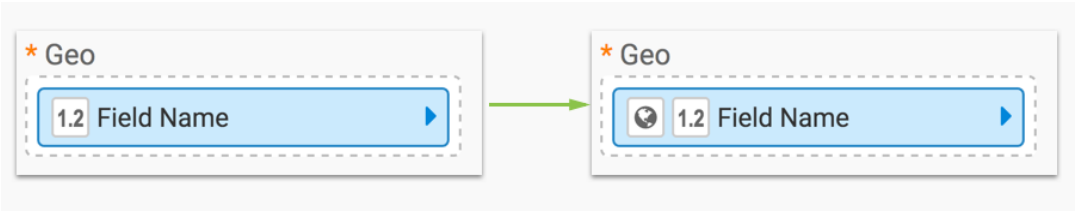

Alternatively, a user with sufficient privileges should declare the field a Geo Type at the level of the dataset. See *Specifying Geographic Fields*.

4. Click REFRESH VISUAL to see the changes.

Fields that carry a type change have a green dot on the Change Type menu.

To remove a type cast from a field, click on the selected type cast. It removes the check mark.

Related Information

Specifying geographic fields

Customizing order and top K

Sorting is one of the most common tools for data management. In Cloudera Data Visualization, you can sort your data in ascending or descending order by implementing the 'ORDER BY' clause on the query. You can also choose to limit the number of records displayed to examine the most significant entries. This can be implemented through the 'LIMIT' clause.

Procedure

1. Click the field you want to modify to open the Field Properties menu.
2. Click Order and Top K to expand the menu.

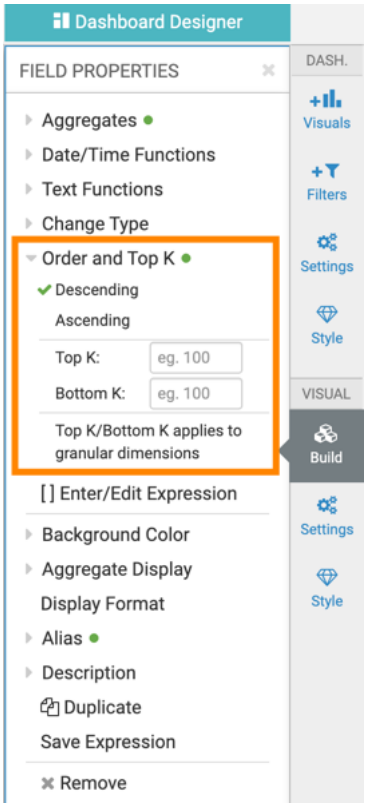

3. Review the following options and configure the sorting and limiting according to your needs:
   - Sorting
     Descending

     This option sorts [field] in descending order.

     Ascending

     This option sorts [field] in ascending order.

     You can turn on sorting if you click Ascending or Descending. A checkmark precedes the selected option. If you want to clear the sort, click the selected option again.
   - Limiting
     Top K

     You can specify the number of records that you want to display, starting from the top of the specified sort.

     Note: This setting overrides the Limit shelf setting. So, if the value of Limit is 100 and Top K is set to 50, the query returns 50 rows.

     Bottom K

     You can specify the number of values that you want to display, starting from the bottom of the specified sort.

     Note: This setting overrides the Limit shelf setting. If the value of Limit is 100 and Bottom K is set to 50, the query returns 50 rows.

   Additionally, you can also apply Top K/Bottom K to granular dimensions.
4. Click REFRESH VISUAL to update your visual.

Customizing enter/edit expressions

Cloudera Data Visualization provides you with an option to manually adjust expressions that modify the field values of the visual, the Enter /Edit Expression interface.

About this task

developer:

- Ensure that column names in the expressions are bracketed, like this: [field].
- When manually editing expressions, you can apply changes that are equivalent to all Field Properties transformations except Order and Top K and Display Format. Entering these changes causes an error.

To set the customize expressions, follow these steps.

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.

2. Under Field Properties, click to expand the [ ] Enter/Edit Expression menu to open the Expression Editor.

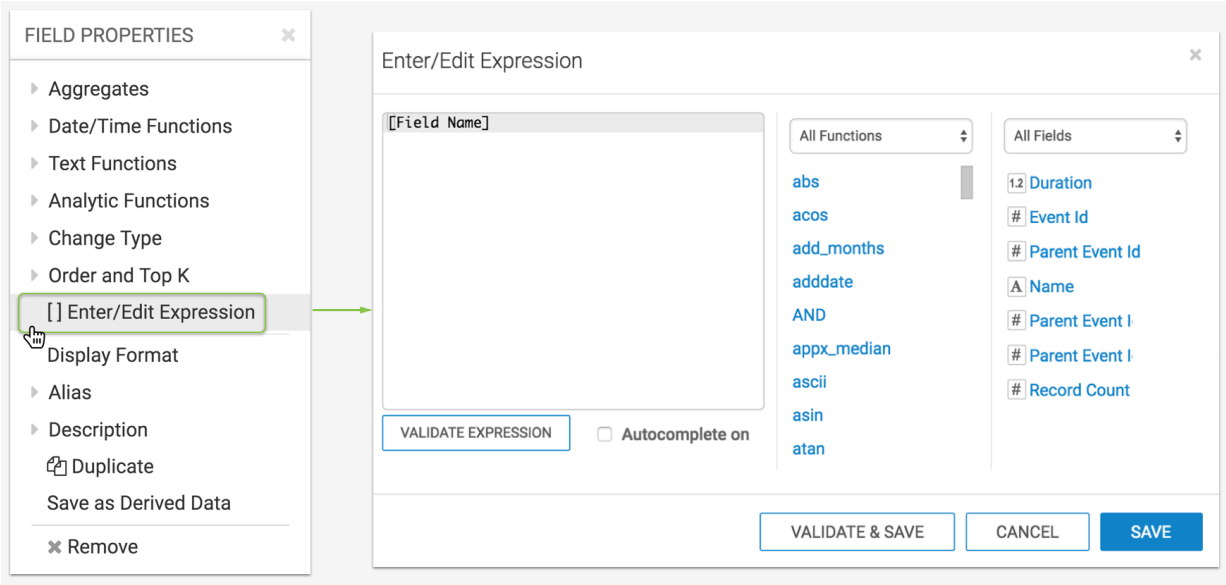

3. In the Enter/Edit Expression editor, change the expression.

   See *Simple Alias expression* and *CASE expression* examples for demonstration of some of the varied expressions you can build using the expression editor.

   a. Expression Text: Enter/modify the expression directly in the text field.
   b. Autocomplete: Click this field to turn expression autocomplete on or off.
   c. Functions: By default, all available functions show. Scroll down to find a specific function, and click it to add it to the Expression Text field. Note the location of the cursor.

   Alternatively, click the All Functions menu to see the function groupings and select the functions from these. Available expression groupings include Aggregate, Conditional, Mathematical, Miscellaneous, String, Filter, and Date and Time.

   Note that different connections have their own SQL syntax, and therefore a different catalog of functions.
   d. All Fields: By default, all available fields show. Scroll down to find a specific field, and click it to add it to the Expression Text field. Note the location of the cursor.

   Alternatively, click the All Fields menu to see the Measures and Dimensions groupings, and select the fields from these.

4. Validate Expression: Validate the expression that you build.

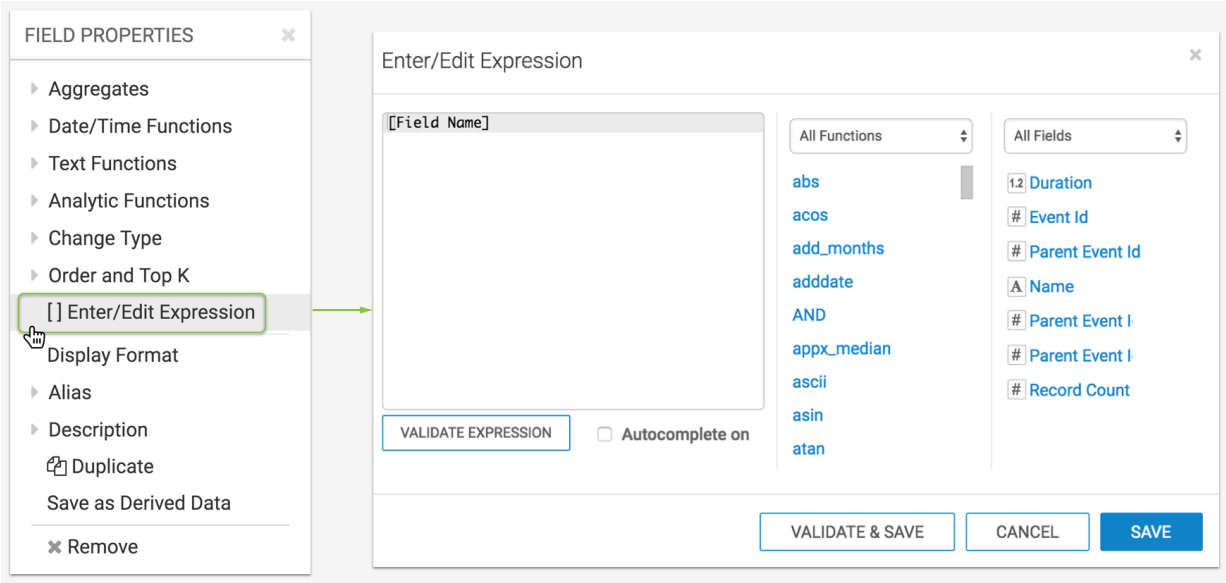

5. Click Save.
6. Click Refresh Visual.

Simple Alias expression

You can add an alias to the expression simply by typing it in the text field:

```
sum([population]) as 'Population'
```

Remember to Validate Expression and click Save.

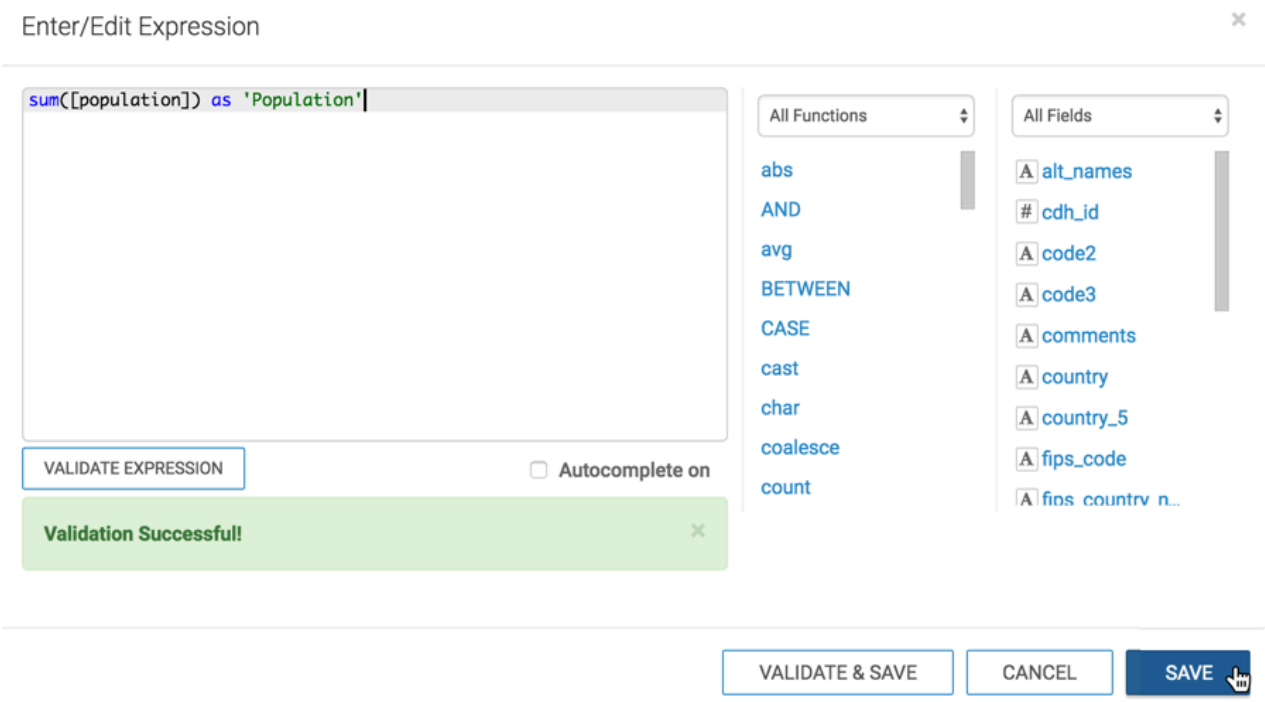

After clicking Refresh Visual, you will see a change similar to the one in the following image:

un_subregion	country	sum(population)
Northern Africa	Algeria	35,468,208
Northern Africa	Egypt	81,121,080
Northern Africa	Libya	6,355,112
Northern Africa	Morocco	31,951,412
Northern Africa	Sudan	34,136,520
Northern Africa	Tunisia	10,480,934
Northern Africa	Western Sahara	530,500

un_subregion	country	Population
Northern Africa	Algeria	35,468,208
Northern Africa	Egypt	81,121,080
Northern Africa	Libya	6,355,112
Northern Africa	Morocco	31,951,412
Northern Africa	Sudan	34,136,520
Northern Africa	Tunisia	10,480,934
Northern Africa	Western Sahara	530,500

CASE expression

You can add a CASE expression to categorize the values, by selecting CASE in the functions area, and using autocomplete. The expression in this example follows; if you paste it into the editor, remove hard line returns:

```
CASE WHEN sum([population])<1000000 THEN 'very low'
  WHEN (sum([population])<10000000 and sum([population])>=1000000) THEN 'lo
w'
  WHEN (sum([population])<20000000 and sum([population])>=30000000) THEN 'me
dium'
  WHEN (sum([population])<40000000 and sum([population])>=30000000) THEN
'high'
  ELSE 'very high'
END
as 'Population'
```

Remember to validate the expression and click Save.

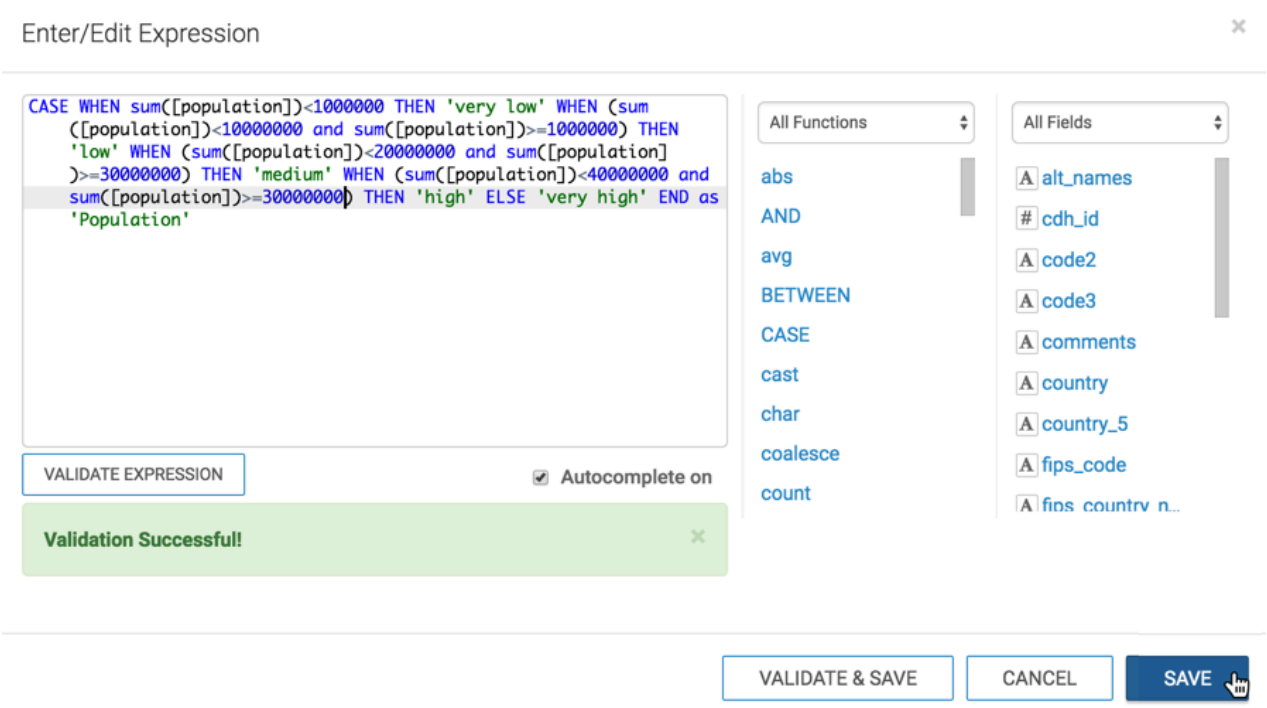

After clicking Refresh Visual, you will see a change similar to the one in the following image:

un_subregion	country	Population
Northern Africa	Algeria	35,468,208
Northern Africa	Egypt	81,121,080
Northern Africa	Libya	6,355,112
Northern Africa	Morocco	31,951,412
Northern Africa	Sudan	34,136,520
Northern Africa	Tunisia	10,480,934
Northern Africa	Western Sahara	530,500

un_subregion	country	Population
Northern Africa	Algeria	medium
Northern Africa	Egypt	very high
Northern Africa	Libya	low
Northern Africa	Morocco	medium
Northern Africa	Sudan	medium
Northern Africa	Tunisia	low
Northern Africa	Western Sahara	low

Customizing calculated display

Cloudera Data Visualizationn allows you to display calculated functions in several ways: as is, cumulative (ascending or descending), and as difference (ascending and descending).

About this task

To add the calculated display option for fields on measurement shelves, follow these steps:

Procedure

1. Click the field you plan to modify to open the FIELD PROPERTIES menu.

2. Click Calculated Display to expand its menu.

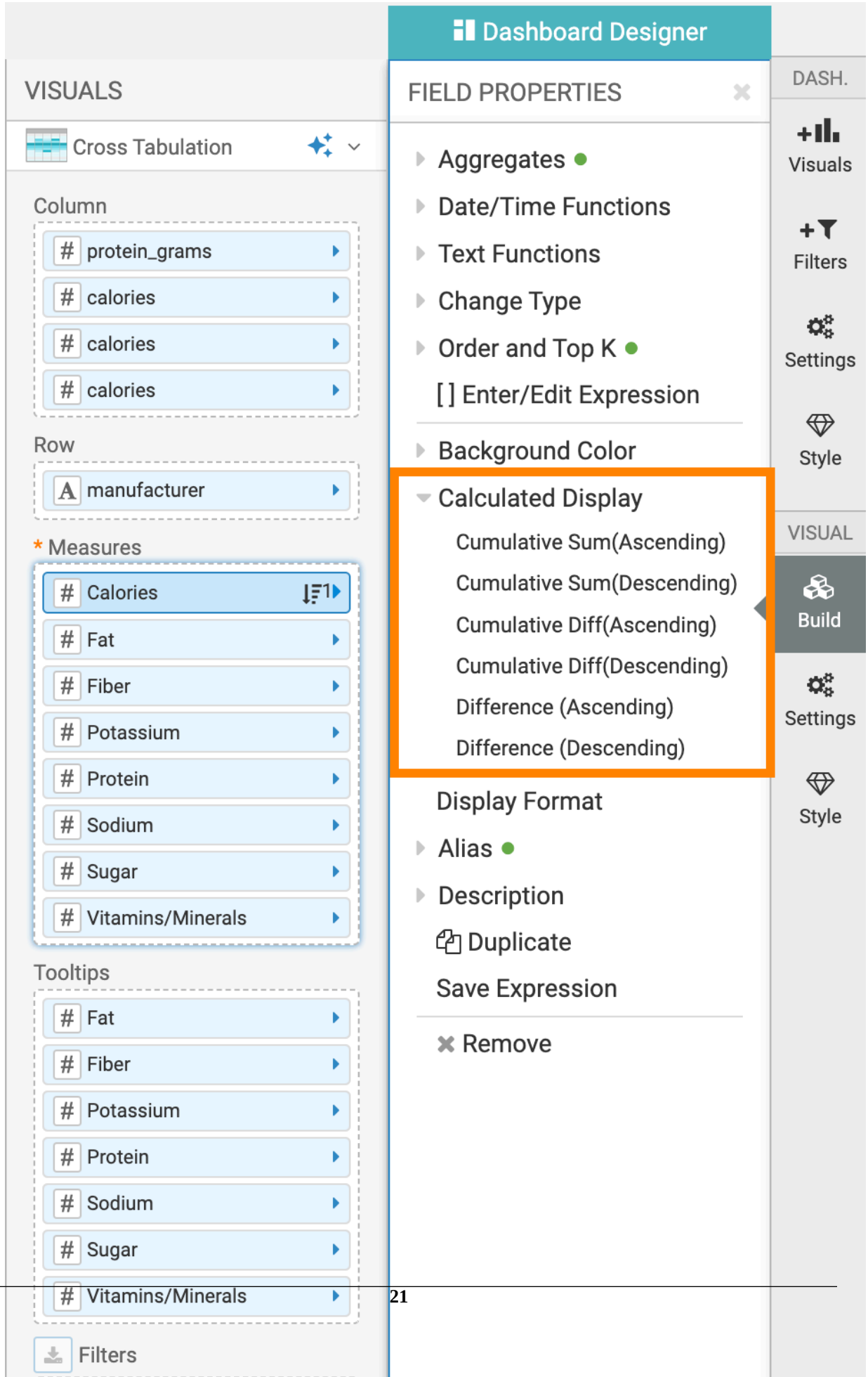
Dashboard Designer
VISUALS
Cross Tabulation
Column
protein_grams
calories
calories
calories
Row
manufacturer
* Measures
Calories
Fat
Fiber
Potassium
Protein
Sodium
Sugar
Vitamins/Minerals
Tooltips
Fat
Fiber
Potassium
Protein
Sodium
Sugar
Vitamins/Minerals
Filters
FIELD PROPERTIES
Aggregates
Date/Time Functions
Text Functions
Change Type
Order and Top K
[ ] Enter/Edit Expression
Background Color
Calculated Display
Cumulative Sum(Ascending)
Cumulative Sum(Descending)
Cumulative Diff(Ascending)
Cumulative Diff(Descending)
Difference (Ascending)
Difference (Descending)
Display Format
Alias
Description
Duplicate
Save Expression
Remove
DASH.
Visuals
Filters
Settings
Style
VISUAL
Build
Settings
Style

3. From the Calculated Display menu, select one of the following options:
   - Cumulative Sum (Ascending)
   - Cumulative Sum (Descending)
   - Cumulative Diff (Ascending)
   - Cumulative Diff (Descending)
   - Difference (Ascending)
   - Difference (Descending)

   A check mark precedes the selected calculated display option.
4. Click REFRESH VISUAL to see the changes.

Enabling expansion

In cross tabulation visuals, you can enable expansion to display subtotal and combined total of the dimensions entered in the Row shelf.

About this task

To enable expansion, follow these steps:

Procedure

1. Click the field you plan to modify to open the Field Properties menu.
2. Under Field Properties, click Enable Expansion.

   A green dot appears to the right of the option.

   Note: The Enable Expansion option only appears on the Row shelf of cross tabulation visuals, to make the field usable in that context.

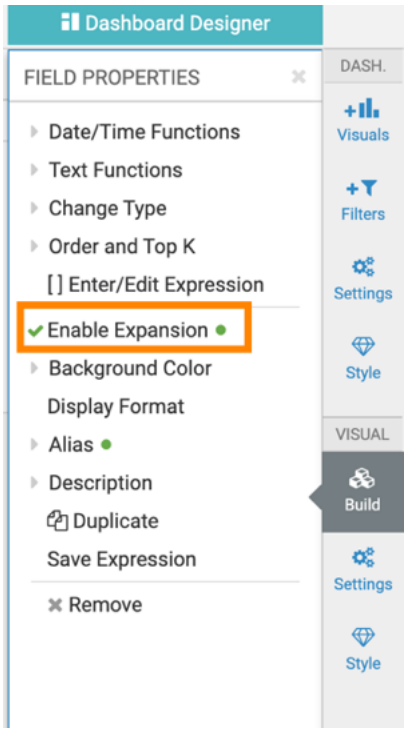

3. Click Refresh Visual to see the changes.

   To disable expansion from a field, click the Enable Expansion option again; that removes the green dot.

   To demonstrate how expansion works, see *Displaying totals with data expansion*.

Related Information

Displaying totals with data expansion

Customizing display format

Cloudera Data Visualization has robust support for formatting the display of individual values.

About this task

To change the display format of a field, follow these steps:

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.
2. Under Field Properties, click Display Format.

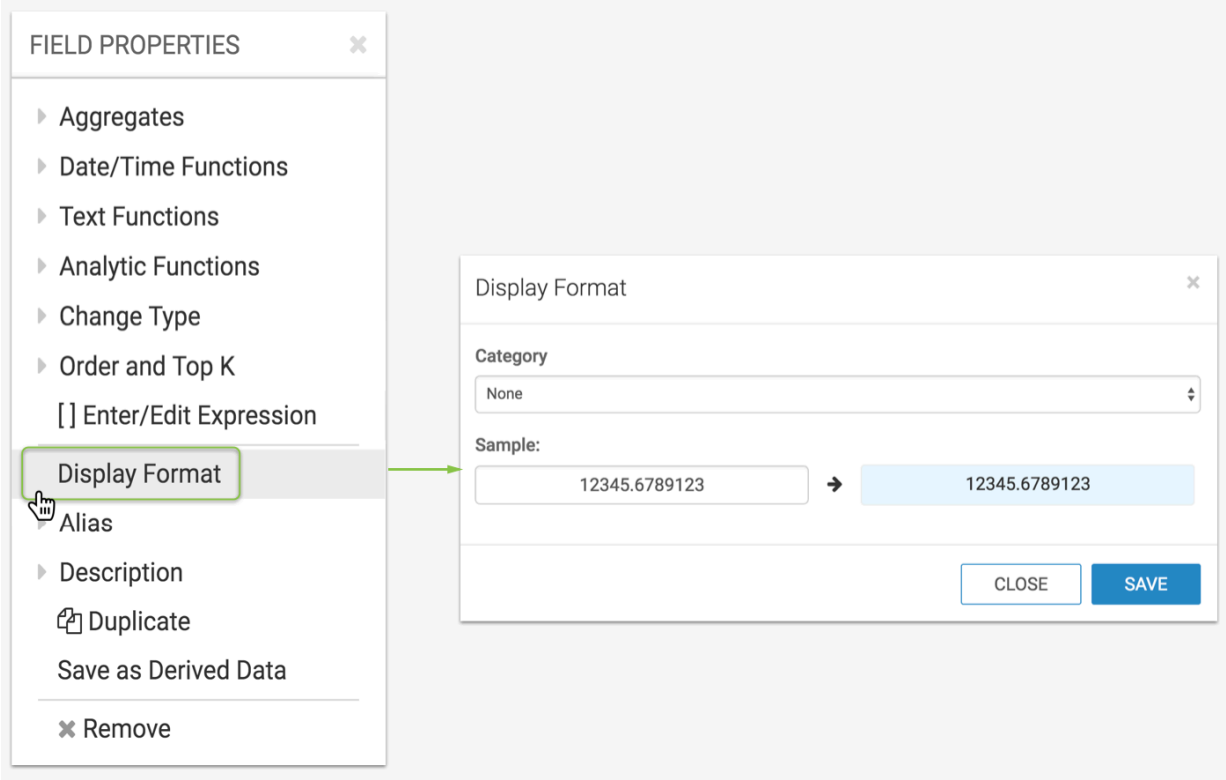

3. Change the display format to another category with specialized options:

   By default, the Category for display format is None.

   For a complete list of supported formats, see *Display Format Reference*.

Real number display format

About this task

Cloudera Data Visualization enables you to specify real number format for a numerical field.

The following steps demonstrate how to adjust the display format for a real number inside a visual: by setting a comma thousands separator and specifying the number of decimals.

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.
2. Under Field Properties, click Display Format.
3. In the Display Format interface, select Real Number from the Category menu.
4. Make adjustments as needed:
   - Under Decimal Places, change the number of decimals.
   - To show no decimals, clear the Decimal Places text box.
   - To show all decimals, select the All option.
   - Select or de-select the Use 1000 separator option.
5. Click Save.

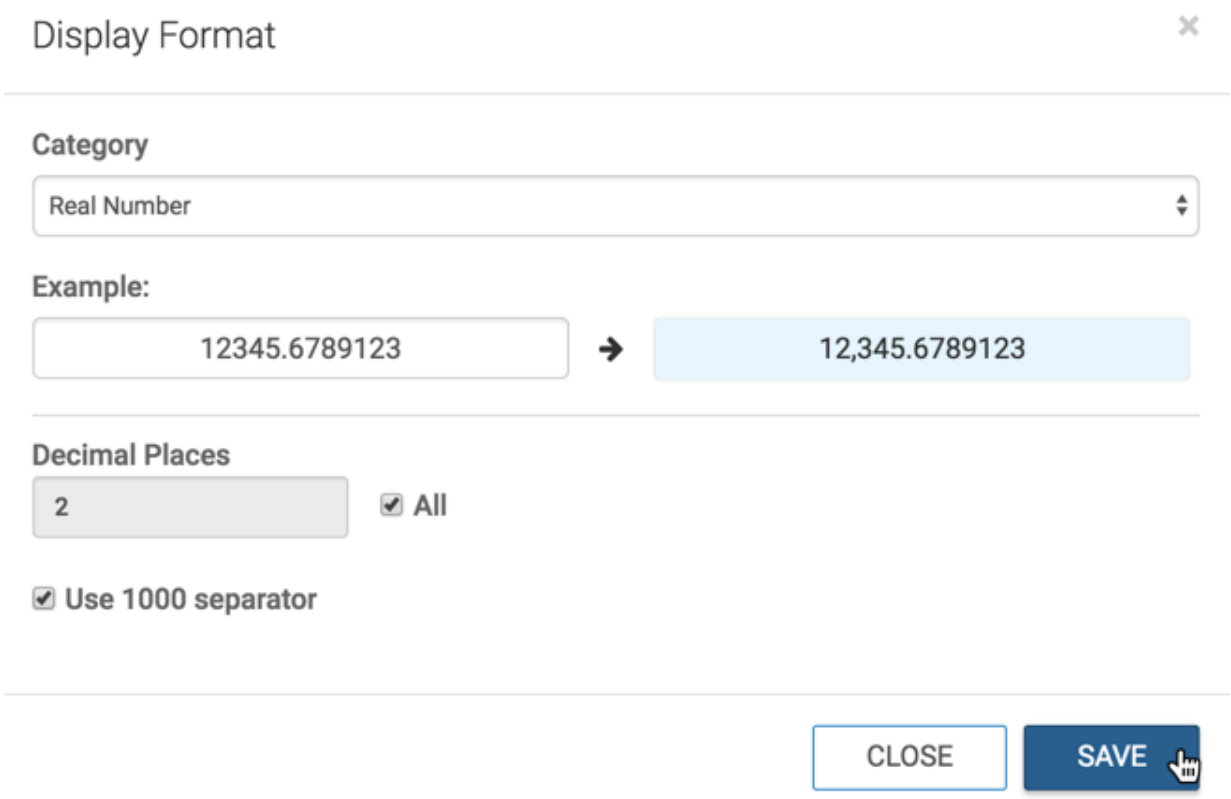

6. Click Refresh Visual.

Integer display format

About this task

Cloudera Data Visualization enables you to specify integer format for a numerical field.

The following steps demonstrate how to adjust the display format for an integer inside a visual, by setting a comma thousands separator.

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.
2. Under Field Properties, click Display Format.
3. In the Display Format interface, select Integer from the Category menu.
4. Make adjustments as needed: Select or de-select the Use 1000 separator option.
5. Click Save.

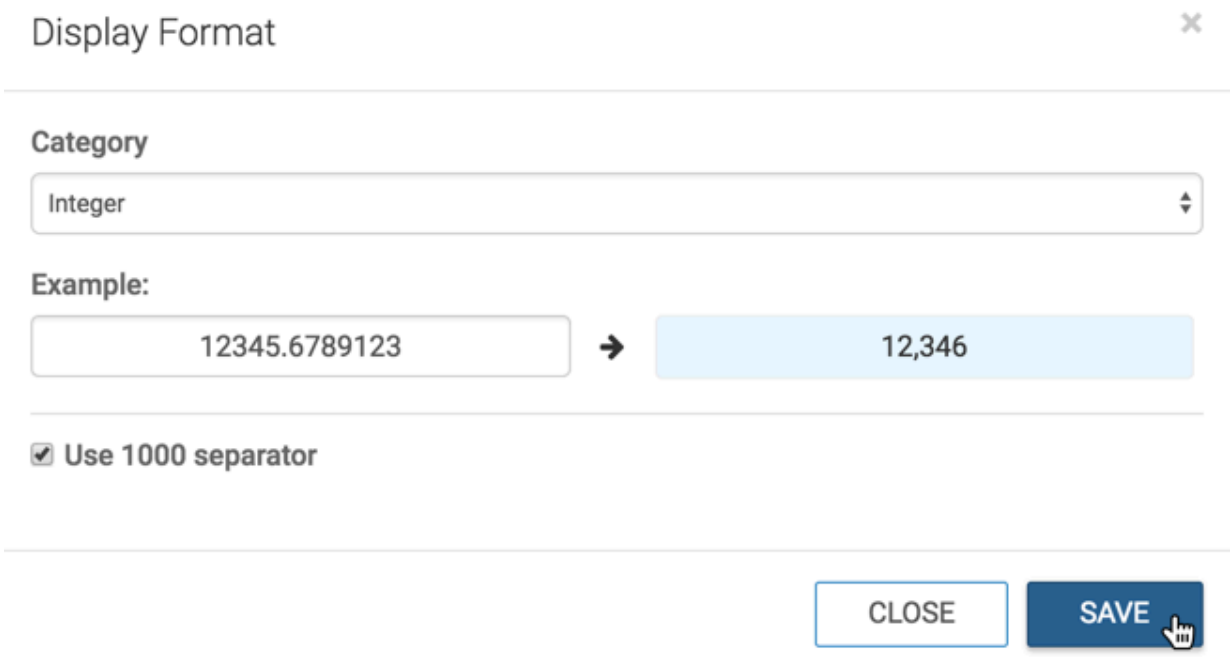

6. Click Refresh Visual.

Percentage display format

About this task

Cloudera Data Visualization enables you to specify percentage format for a numerical field.

Tip: The % operation automatically multiplies the number by 100. If your data source reports percentages as whole numbers ('2%' is stored as '2' instead of '0.02'), you must divide by 100 first. Cloudera recommends that you make this adjustment globally, in the Dataset Attribute Definition interface, as described in *Changing Column Calculation.*

The following steps demonstrate how to properly use percentages, the % format option, with display format.

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.
2. Under Field Properties, click Display Format.
3. In the Display Format interface, select Percentage from the Category menu.

4. Make adjustments as needed:
   - Under Decimal Places, change the number of decimals.
   - To show no decimals, clear the Decimal Places text box.
   - To show all decimals, select the All option.
   - Select the Use 1000 separator option.
5. Click SAVE.

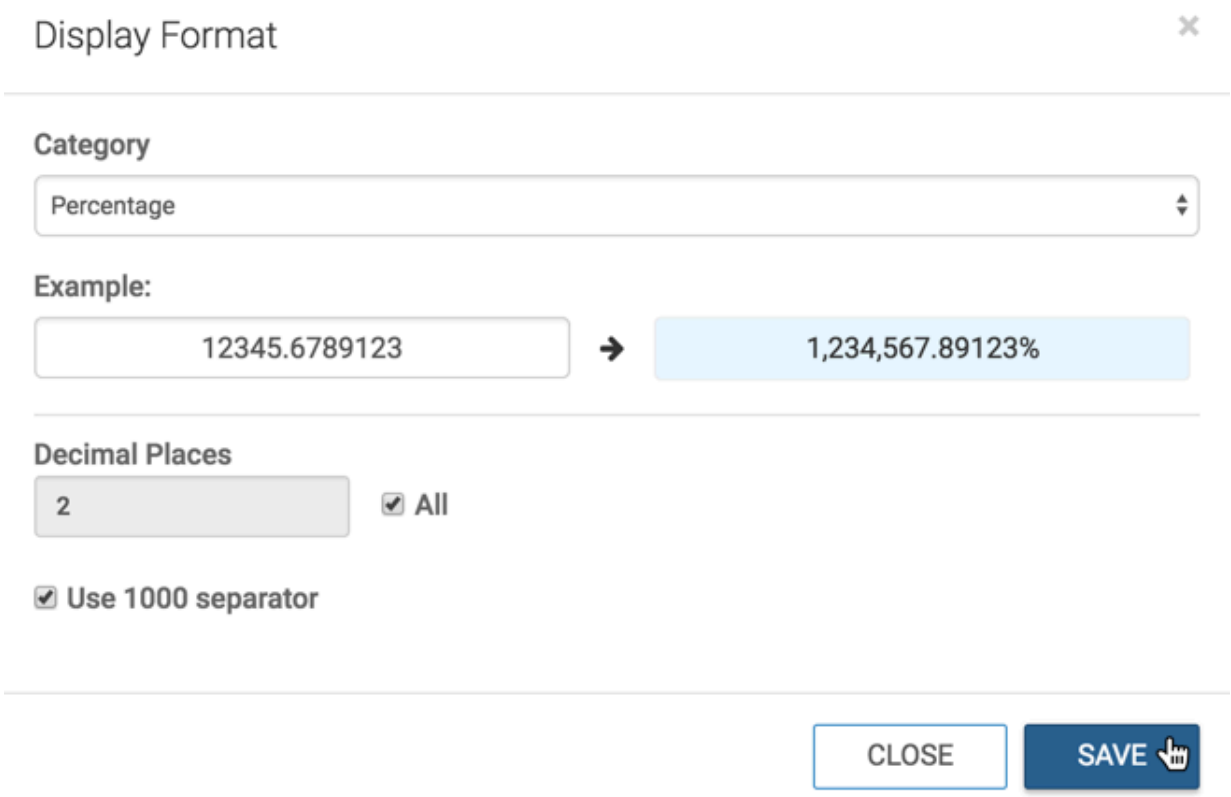

6. Click REFRESH VISUAL.

Percentage Display with Cereals Dataset

Here, we are using a basic table visual built in the Cereals dataset.

1. Start a Table visual on the Cereals dataset.
2. Populate the shelves from the available fields (Dimensions, Measures, and so on) in the Data menu.
   - Place cereal_name on the Dimensions shelf.
   - Place calories, protein_grams, fat_grams, dietary_fiber_grams, complex carbohydrates_grams, sugars_grams, and sodium_mg on the Measures shelf.
3. [Optional] Use Alias to rename the fields on the shelf to % Daily Calories, % Daily Protein, and so on.
4. Ensure that the aggregation function used by the measure is Average, for calculations in the next step. See *Aggregates*.
5. To calculate the percentage of daily recommended intake that a portion of each cereal represents, edit the expression of each field to divide by daily values.

Table 1: Daily Intake Recommendations

Calories	Protein	Fats	Fiber	Carbohydrates	Sugar	Sodium
2000 cal	50 grams	70 grams	30 grams	310 grams	90 grams	2300 milligrams

6. Set the display format for the re-calculated fields.

   For all the fields on the Measures shelf, change the Display Format to Percentage, select a 2- decimal display, and click SAVE.
7. Click REFRESH VISUAL.

   Note that the visual's display has the specified format, the % sign, and 2 decimal places.

cereal_name	% Daily Calories	% Daily Protein	% Daily Fats	% Daily Sodium	% Daily Fiber	% Daily Carbohydrates	% Daily Sugar
100%_Bran	3.50%	8.00%	1.43%	5.65%	33.33%	1.61%	6.67%
100%_Natural_Bran	6.00%	6.00%	7.14%	0.65%	6.67%	2.58%	8.89%
All-Bran	3.50%	8.00%	1.43%	11.30%	30.00%	2.26%	5.56%
All-Bran_with_Extra_Fiber	2.50%	8.00%	0.00%	6.09%	46.67%	2.58%	0.00%
Almond_Delight	5.50%	4.00%	2.86%	8.70%	3.33%	4.52%	8.89%
Apple_Cinnamon_Cheerios	5.50%	4.00%	2.86%	7.83%	0.00%	0.00%	11.11%
Apple_Jacks	5.50%	4.00%	0.00%	5.43%	3.33%	3.55%	15.56%
Basic_4	6.50%	6.00%	2.86%	9.13%	6.67%	5.81%	8.89%
Bran_Chex	4.50%	4.00%	1.43%	8.70%	13.33%	4.84%	6.67%
Bran_Flakes	4.50%	6.00%	0.00%	9.13%	16.67%	4.19%	5.56%

Related Information

Creating calculated fields

Scientific display format

About this task

Cloudera Data Visualization enables you to specify scientific display format for a numerical field, both in exponential form, and using unit abbreviation.

Scientific number notation is fully supported by the Scientific category. The following steps demonstrate how to use this display format.

Cloudera Data Visualization offers two built-in options for Scientific formatting: Exponential, and Unit Abbreviation.

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.
2. Under Field Properties, click Display Format.
3. In the Display Format interface, select Scientific from the Category menu.
4. Make adjustments as needed:
   - Under Unit Notation, select either Exponential or Unit abbreviation.
   - Under Decimal Places, change the number of decimals.
   - To show no decimals, clear the Decimal Places text box.
   - To show all decimals, select the All option.

5. Click Save.

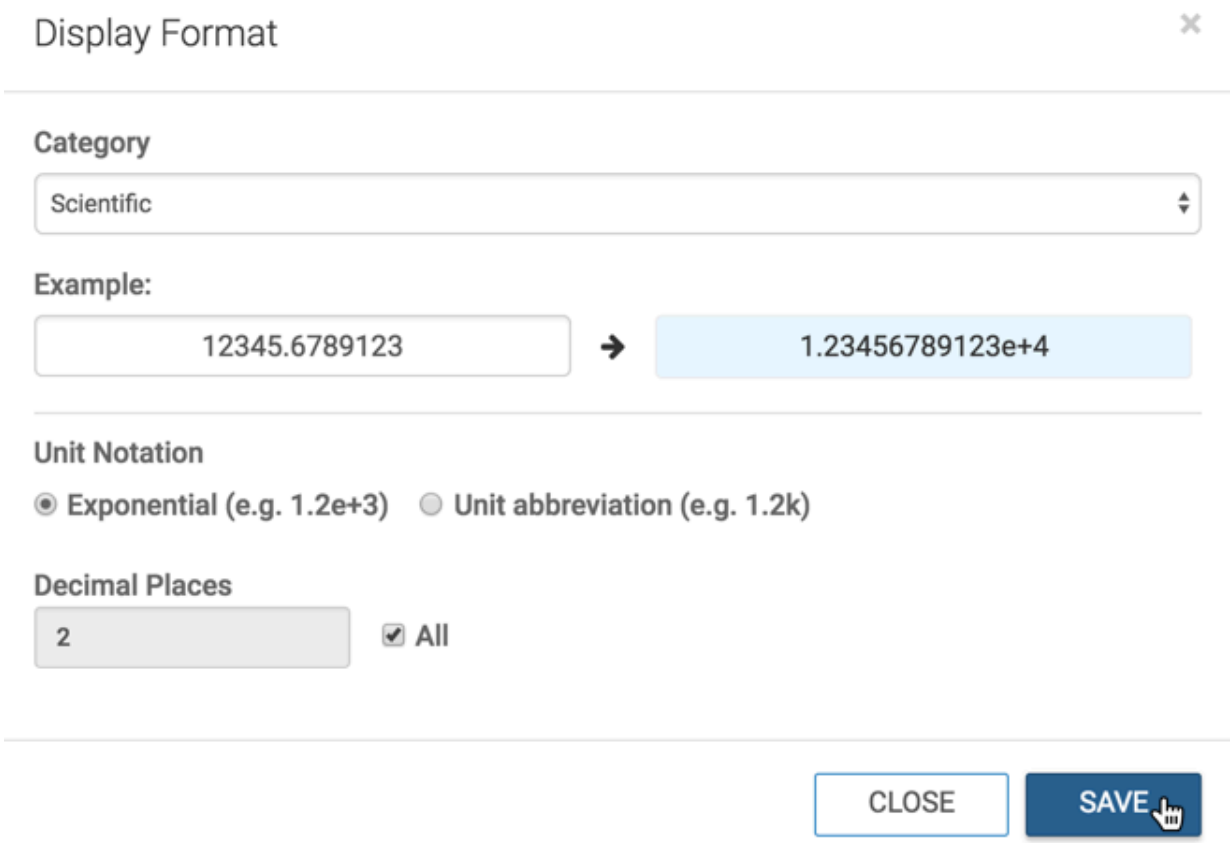

6. Click Refresh Visual.

Scientific format with exponential notation

We are using the Encyclopedia of Starsdataset, based on a datafile.

1. Create a new table visual on the dataset.
2. Place the field proper_name on the Dimensions shelf, and the field lightyears on the Measures shelf.
3. Change the Format Display for lightyears to Scientific.
4. Under Unit Notation, select Exponential, and click Save.
5. Click Refresh Visual.

6. Notice the change to the number format in column lightyears.

proper_name	lightyears
268 G. Cet	23.41897927
3C 273	326156
82 G. Eri	19.7109117
96 G. Psc	24.31460364
Acamar	161.2238031
Achernar	139.4424531
Acrux	321.9704401
Adhara	405.1627248
Albaldah	509.61875

proper_name	lightyears
268 G. Cet	2.34e+1
3C 273	3.26e+5
82 G. Eri	1.97e+1
96 G. Psc	2.43e+1
Acamar	1.61e+2
Achernar	1.39e+2
Acrux	3.22e+2
Adhara	4.05e+2
Albaldah	5.10e+2

Scientific format with unit abbreviation

We are using the Encyclopedia of Starsdataset, based on a datafile.

1. Create a new table visual on the dataset.
2. Place the field proper_name on the Dimensions shelf, and the following fields on the Measures shelf:lightyears, cartesian_x, cartesian_y, cartesian_z, cartesian_velocity_x, cartesian_velocity_y, and cartesian_velocity_z.
3. Change the Format Display for all fields on the Measures shelf to Scientific, with the Unit Abbreviation option, and click Save.
4. Click Refresh Visual.

5. Notice the change to the number format between the tables:

proper_name	lightyears	cartesian_x	cartesian_y	cartesian_z	cartesian_velocity_x	cartesian_velocity_y	cartesian_velocity_z
268 G. Cet	23.41897927	5.538462	4.488223	0.861013	-0.00002374	0.00006168	0.00005304
3C 273	326156	-99130.70606	-12660.05862	3581.344963	-0.00060072	0.00530397	0.00212194
82 G. Eri	19.7109117	2.839238	3.381499	-4.127499	-0.00001715	0.00011797	-0.00004501
96 G. Psc	24.31460364	7.258368	1.555495	0.686093	-0.00001198	0.00002545	-0.00004201
Acamar	161.2238031	26.85749	26.453029	-31.974932	0.00001851	2.3e-7	-0.00000324
Achernar	139.4424531	21.065573	9.568417	-35.951967	-0.00000584	0.00001738	-0.00001826
Acrux	321.9704401	-44.363755	-5.171957	-88.034513	0.00000934	0.00001813	0.00000684
Adhara	405.1627248	-27.497987	105.141608	-60.171969	-0.00000781	0.00002362	-0.00001217
Albaldah	509.61875	43.714239	-139.143552	-56.055078	-0.00000671	0.00001839	-0.00002237
Albireo	434.295631	45.350466	-108.51838	62.430012	-0.00001193	0.00001667	-0.00001472
Alcor	81.72295198	-13.393436	-5.223608	20.521915	0.00000861	-0.00001233	-0.00000864
Alcyone	403.1594747	61.664507	94.489136	50.483636	0.00000116	0.00002299	-0.0000194
Aldebaran	66.64410779	7.02722	18.287604	5.806661	0.00001528	0.00005709	-0.00000214

proper_name	lightyears	cartesian_x	cartesian_y	cartesian_z	cartesian_velocity_x	cartesian_velocity_y	cartesian_velocity_z
268 G. Cet	23	5.5	4.5	860m	-24µ	62µ	53µ
3C 273	330k	-99k	-13k	3.6k	-600µ	5.3m	2.1m
82 G. Eri	20	2.8	3.4	-4.1	-17µ	120µ	-45µ
96 G. Psc	24	7.3	1.6	690m	-12µ	25µ	-42µ
Acamar	160	27	26	-32	19µ	230n	-3.2µ
Achernar	140	21	9.6	-36	-5.8µ	17µ	-18µ
Acrux	320	-44	-5.2	-88	9.3µ	18µ	6.8µ
Adhara	410	-27	110	-60	-7.8µ	24µ	-12µ
Albaldah	510	44	-140	-56	-6.7µ	18µ	-22µ
Albireo	430	45	-110	62	-12µ	17µ	-15µ
Alcor	82	-13	-5.2	21	8.6µ	-12µ	-8.6µ
Alcyone	400	62	94	50	1.2µ	23µ	-19µ
Aldebaran	67	7.0	18	5.8	15µ	57µ	-2.1µ

Currency display format

About this task

Cloudera Data Visualization enables you to specify currency display format for a numerical field by selecting a common currency symbol.

You can customize the display of measures that represent currency. Direct support is available for the following currencies: $ (Dollar), £ (Pound), ¥ (Yen/Yuan), # (Rupee), € (Euro), and # (Cedi).

The following steps demonstrate how to adjust the display format for currency, using the Currency category.

Procedure

1. Click the field you plan to modify, to open the FIELD PROPERTIES menu.
2. Under FIELD PROPERTIES, click Display Format.

3. In the Display Format modal window, select Currency from the Category menu.
4. Select the appropriate currency symbol: $ (Dollar), £ (Pound), ¥ (Yen/Yuan), # (Rupee), € (Euro), or # (Cedi).
5. Select the Basic Format for your records.

   You can also define and apply a custom format. Enter a valid format mask in the Customize Currency text box. For a full list of options, see *Display Format Reference*.

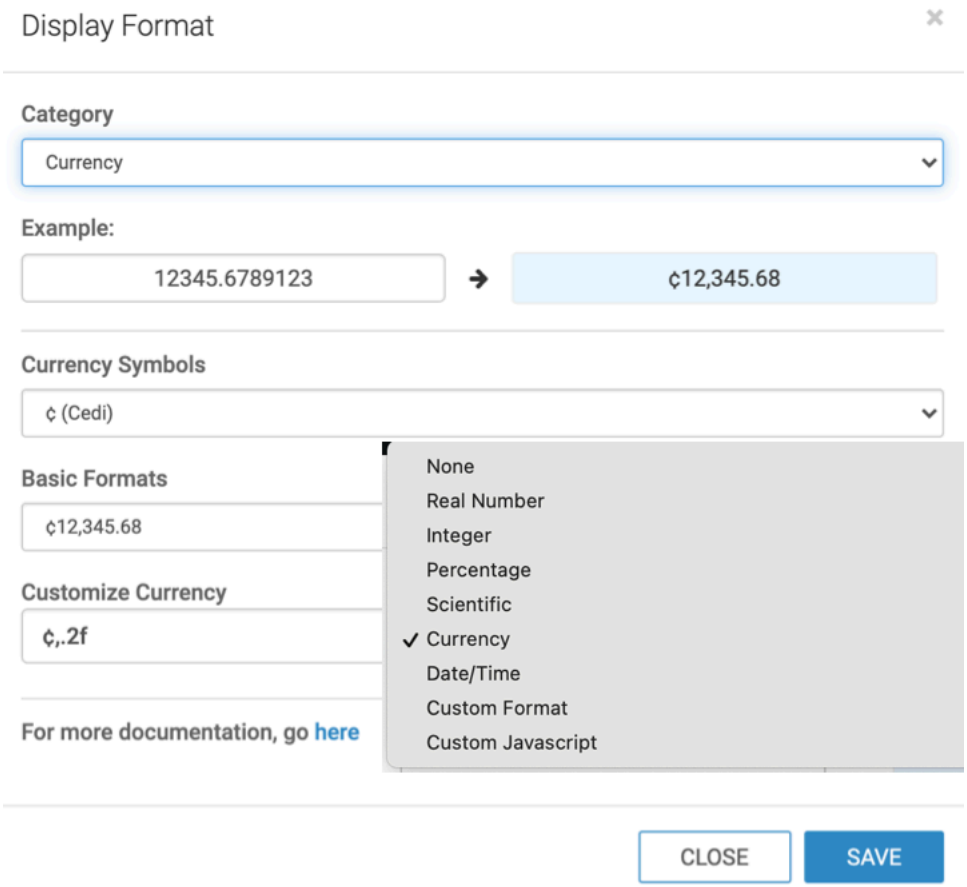

6. Click SAVE.

Related Information

Display format reference

Date/time display format

About this task

Cloudera Data Visualization enables you to specify date/time display formats for a numerical field by selecting from a number of common formats, or by specifying a custom format.

You have common notation options for formatting Date/Time fields. Cloudera Data Visualization also offers a simple customization component that transforms date/time information.

By default, data/time records appear in the form [YYYY]-[MM]-[DD] [hh]:[mm]:[ss].[sss], in most-to-least significant order. This is consistent with the *ISO 8601 Date and Time Specification*.

The following steps demonstrate how to adjust the display format for date/time fields:

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.
2. Under Field Properties, click Display Format.
3. In the Display Format interface, select Date/Time from the Category menu.
4. Make a selection from one of the Basic Formats. These include most common forms, such as 2010-01-05, 01/05/2010, 1/5/10, Jan 05, 2010, Friday, January 5th, 2010, 10-Jan-05, Q1 2010, 1:05 pm, 01:05:53 PM, 13:05:53.010, 1/5/10 1:05:53 PM, and 01/05/2010 13:05:53.

   Alternatively, specify select the Custom Date Format option in the Basic Formats menu, and then use the Customize Date/Time text filed to enter the custom format.

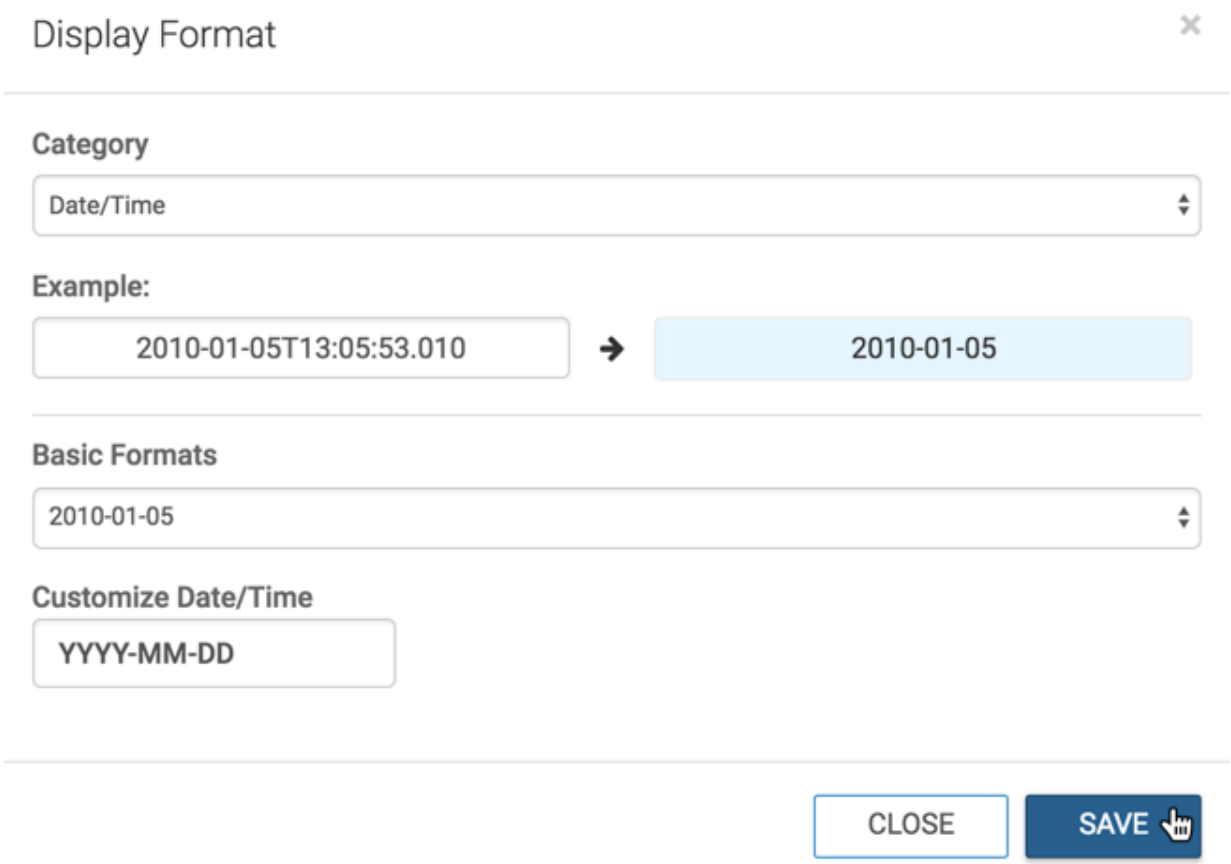

5. Click Save.
6. Click Refresh Visual.

   The following examples demonstrate the use of date/time display formatting interface for *Basic Formats for Date/Time* and *Custom Formats for Date/Time*. We are using the NYC Taxicab Rides Detail dataset, based on the table main.trips-detail.

Basic Formats for Date/Time

1. Create a new Table visual on the NYC Taxicab Rides Detail dataset.
2. Place the field pickup_neighborhood on the Dimensions shelf, and the field pickup_datetime on the Measures shelf.
3. Change the format display for pickup_datetime field by selecting the mask 2010-01-05 from the Basic Formats menu of the Display Format: Date/Time interface.
4. Click Refresh Visual.

5. Notice the change to the number format in column pickup_datetime.

pickup_neighborhood	pickup_datetime
Airport	2013-10-07 10:01:15
Astoria	2013-10-07 03:46:00
Baisley Park	2013-10-07 13:01:00
Bath Beach	2013-10-07 02:55:00
Battery Park City-Lower Manhattan	2013-10-07 19:48:01
Bay Ridge	2013-10-07 22:35:00
Bayside-Bayside Hills	2013-10-07 05:13:00
Bedford	2013-10-07 20:08:44
Bedford Park-Fordham North	2013-10-07 19:28:33
Bellerose	2013-10-07 17:14:00
Belmont	2013-10-07 15:02:12

pickup_neighborhood	pickup_datetime
Airport	2013-10-07
Astoria	2013-10-07
Baisley Park	2013-10-07
Bath Beach	2013-10-07
Battery Park City-Lower Manhattan	2013-10-07
Bay Ridge	2013-10-07
Bayside-Bayside Hills	2013-10-07
Bedford	2013-10-07
Bedford Park-Fordham North	2013-10-07
Bellerose	2013-10-07
Belmont	2013-10-07

Custom Formats for Date/Time

1. Create a new Table visual on the NYC Taxicab Rides Detail dataset.
2. Place the field pickup_neighborhood on the Dimensions shelf, and the field pickup_datetime on the Measures shelf.
3. Change the format display for pickup_datetime field by entering the mask hh:mm aselecting the mask 2010-01-05 under the Customize Date/Time option of the Display Format: Date/Time interface.
4. In the Basic Formats menu, select Custom Date Format.
5. Click Refresh Visual.
6. Notice the change to the number format in column pickup_datetime.

pickup_neighborhood	pickup_datetime
Airport	2013-10-07 10:01:15
Astoria	2013-10-07 03:46:00
Baisley Park	2013-10-07 13:01:00
Bath Beach	2013-10-07 02:55:00
Battery Park City-Lower Manhattan	2013-10-07 19:48:01
Bay Ridge	2013-10-07 22:35:00
Bayside-Bayside Hills	2013-10-07 05:13:00
Bedford	2013-10-07 20:08:44
Bedford Park-Fordham North	2013-10-07 19:28:33
Bellerose	2013-10-07 17:14:00
Belmont	2013-10-07 15:02:12

pickup_neighborhood	pickup_datetime
Airport	10:01 am
Astoria	03:46 am
Baisley Park	01:01 pm
Bath Beach	02:55 am
Battery Park City-Lower Manhattan	07:48 pm
Bay Ridge	10:35 pm
Bayside-Bayside Hills	05:13 am
Bedford	08:08 pm
Bedford Park-Fordham North	07:28 pm
Bellerose	05:14 pm
Belmont	03:02 pm

Using custom display configurations

About this task

Cloudera Data Visualization has a free-format option that enables you to specify numerical data formats. This Custom Format option provides a wide range of custom display configurations.

The following steps outline how to use the Display Format: Custom Format interface.

Procedure

1. Click the field you want to modify to open the FIELD PROPERTIES menu.
2. Under FIELD PROPERTIES, click Display Format.
3. In the Display Format modal window, select Custom Format from the Category menu.

4. In the Customize Format text box, enter a valid format mask based on the examples provided in the modal window.

   For a full list of options, see *Display Format Reference*.

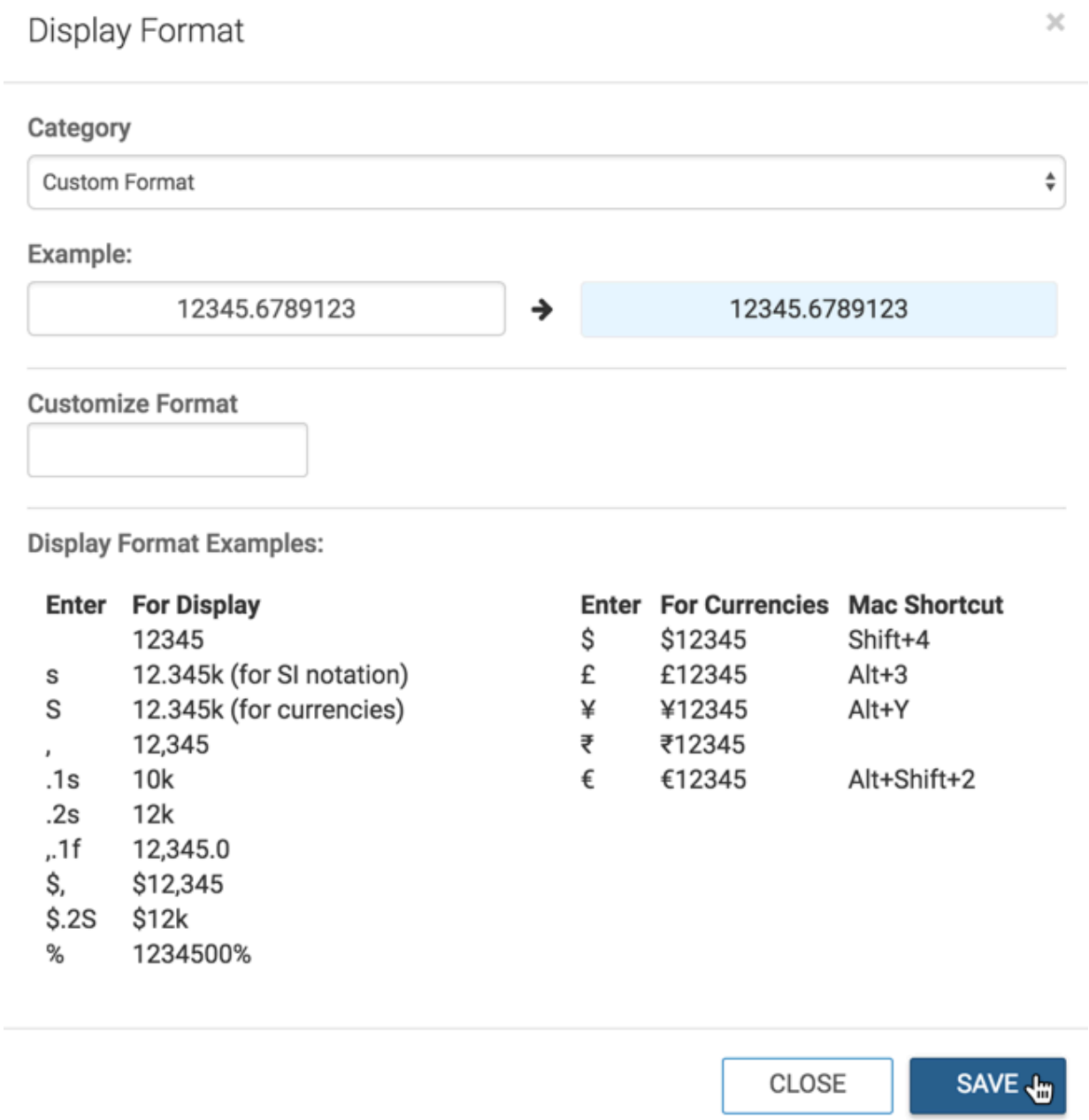

5. Click SAVE.

Common number formatting

Some examples of common display formats follow:

- Entering 1s produces decimal format with Roman numerals for unit abbreviation.

  For example, 37,253,956 renders as 37.253956M.

	state				
year	CA	CO	NV	OR	WA
2010	37.253956M	5.029196M	2.700551M	3.831074M	6.72454M
2000	33.871648M	4.301261M	1.998257M	3.421399M	5.894121M
1990	29.760021M	3.294394M	1.201833M	2.842321M	4.866692M
1980	23.667764M	2.889735M	800.508k	2.633156M	4.132353M
1970	19.971069M	2.209596M	488.738k	2.091533M	3.413244M
1960	15.717204M	1.753947M	285.278k	1.768687M	2.853214M
1950	10.586223M	1.325089M	160.083k	1.521341M	2.378963M
1940	6.907387M	1.123296M	110.247k	1.089684M	1.736191M

- Entering ,.1f produces a format with 1 digit after the decimal, and comma for the thousands separator.

  So, 37,253,956 renders as 37,253,956.0.

	state				
year	CA	CO	NV	OR	WA
2010	37,253,956.0	5,029,196.0	2,700,551.0	3,831,074.0	6,724,540.0
2000	33,871,648.0	4,301,261.0	1,998,257.0	3,421,399.0	5,894,121.0
1990	29,760,021.0	3,294,394.0	1,201,833.0	2,842,321.0	4,866,692.0
1980	23,667,764.0	2,889,735.0	800,508.0	2,633,156.0	4,132,353.0
1970	19,971,069.0	2,209,596.0	488,738.0	2,091,533.0	3,413,244.0
1960	15,717,204.0	1,753,947.0	285,278.0	1,768,687.0	2,853,214.0
1950	10,586,223.0	1,325,089.0	160,083.0	1,521,341.0	2,378,963.0
1940	6,907,387.0	1,123,296.0	110,247.0	1,089,684.0	1,736,191.0

- Entering ,.2f produces a format with 2 digits after the decimal, and comma for the thousands separator.

  So, 37,253,956 renders as 37,253,956.0.

	state				
year	CA	CO	NV	OR	WA
2010	37,253,956.00	5,029,196.00	2,700,551.00	3,831,074.00	6,724,540.00
2000	33,871,648.00	4,301,261.00	1,998,257.00	3,421,399.00	5,894,121.00
1990	29,760,021.00	3,294,394.00	1,201,833.00	2,842,321.00	4,866,692.00
1980	23,667,764.00	2,889,735.00	800,508.00	2,633,156.00	4,132,353.00
1970	19,971,069.00	2,209,596.00	488,738.00	2,091,533.00	3,413,244.00
1960	15,717,204.00	1,753,947.00	285,278.00	1,768,687.00	2,853,214.00
1950	10,586,223.00	1,325,089.00	160,083.00	1,521,341.00	2,378,963.00
1940	6,907,387.00	1,123,296.00	110,247.00	1,089,684.00	1,736,191.00

To configure unit abbreviation for large numbers, see *Using custom display formats for large numbers*.

Using custom display formats for large numbers

Currency magnitudes use expressions such as thousands (103), millions (106), billions (109), and trillions (1012). In contrast, scientific notation uses prefixes: k: kilo- (103), M: Mega- (106), G: Giga- (109), and T: Tera- (1012). viz-display-format-custom.png has an option that helps you to distinguish between scientific and currency unit abbreviations.

About this task

The following steps demonstrate how to adjust the display format for currency, using the S format option.

Procedure

1. Create a new Bars visual on the Capital Expenditure by Industry dataset.
2. Add the industry field on the X Axis shelf and the expenditure field on the Y Axis shelf.
3. Click the expenditure field to open the FIELD PROPERTIES menu.
4. Under FIELD PROPERTIES, click Display Format.
5. In the Display Format modal window, select Currency from the Category menu and change the format to display the $ (dollar) currency symbol.
6. Click SAVE.

7. Click REFRESH VISUAL.

   You can see the display of numbers on the axis scale and in the tooltips.

   Obviously, the standard currency form is inadequate for representing very large numbers and quickly reviewing the difference in their magnitude.

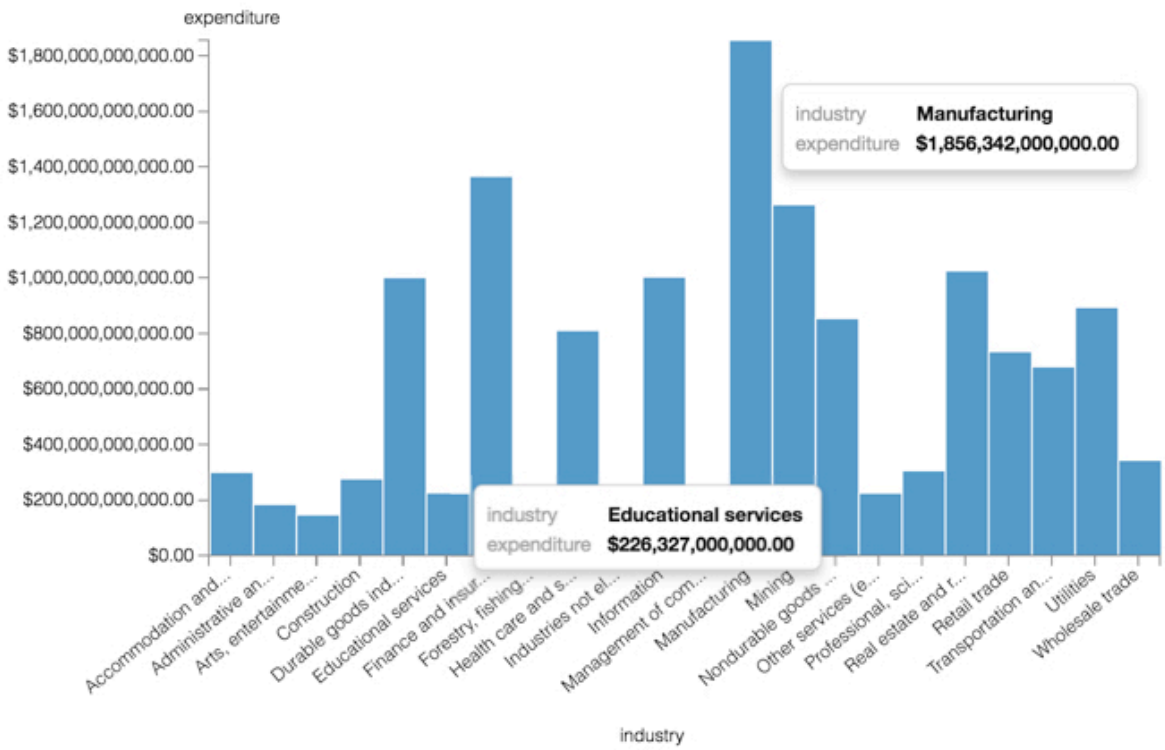

8. Change the display format of the expenditure field to a custom format by setting the value of Customize Date/Time to $,S (currency, ',' thousands separator, and 'S' for currencies).
9. Click SAVE.

10. Click REFRESH VISUAL.

The new display format categorizes the magnitude of the currency more clearly , with Educational Services expenditure valued at approximately $226 Billion, contrasting with Manufacturing at nearly $2 Trillion, for the years covered by the data.

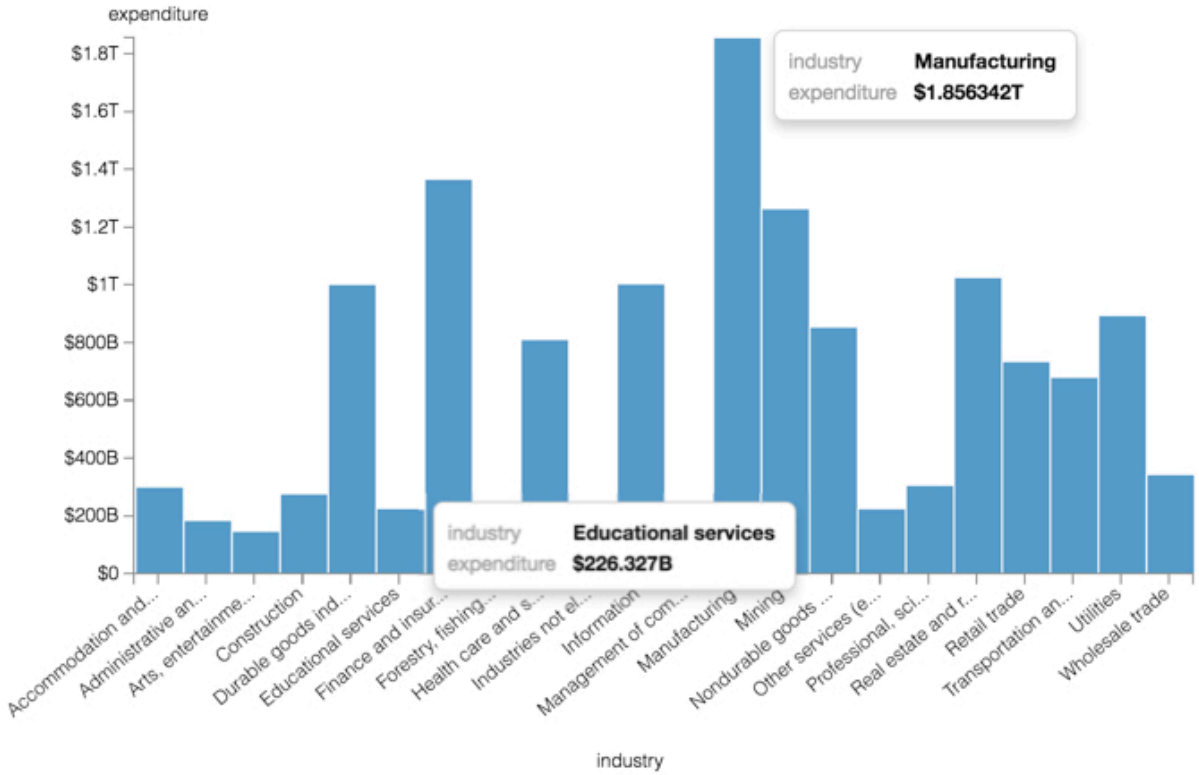

11. You can see how the data format appears across the data set by creating a dashboard that contains both this bar chart and a table visual.

- Clone the visual and change the chart type to table.
- Name and save that visual.
- Click NEW DASHBOARD, and add both visuals to it.
- Name and save the dashboard.

Some column values have the suffix B (billions), while larger columns have the suffix T (trillions).

Customizing display format using Javascript

About this task

In Cloudera Data Visualization, you can customize the display format of fields in visuals using Javascript. The Javascript editor includes auto-complete functionality, and checks for valid code.

This feature is available with the following conditions:

- Site-wide setting for customization are turned on; see *Enabling Custom Styling*.
- The user is either an Admin, or has *Manage custom styles* permissions.

In addition to other categories, Cloudera Data Visualization has a Custom Javascript option that enables a wide range of custom display configurations.

The following steps outline how to use the Display Format: Custom Javascript interface.

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.
2. Under Field Properties, click Display Format.
3. In the Display Format interface, select Custom Javascript from the Category menu.

4. In the Customize JS Format Function code entry box, enter valid JavaScript code.

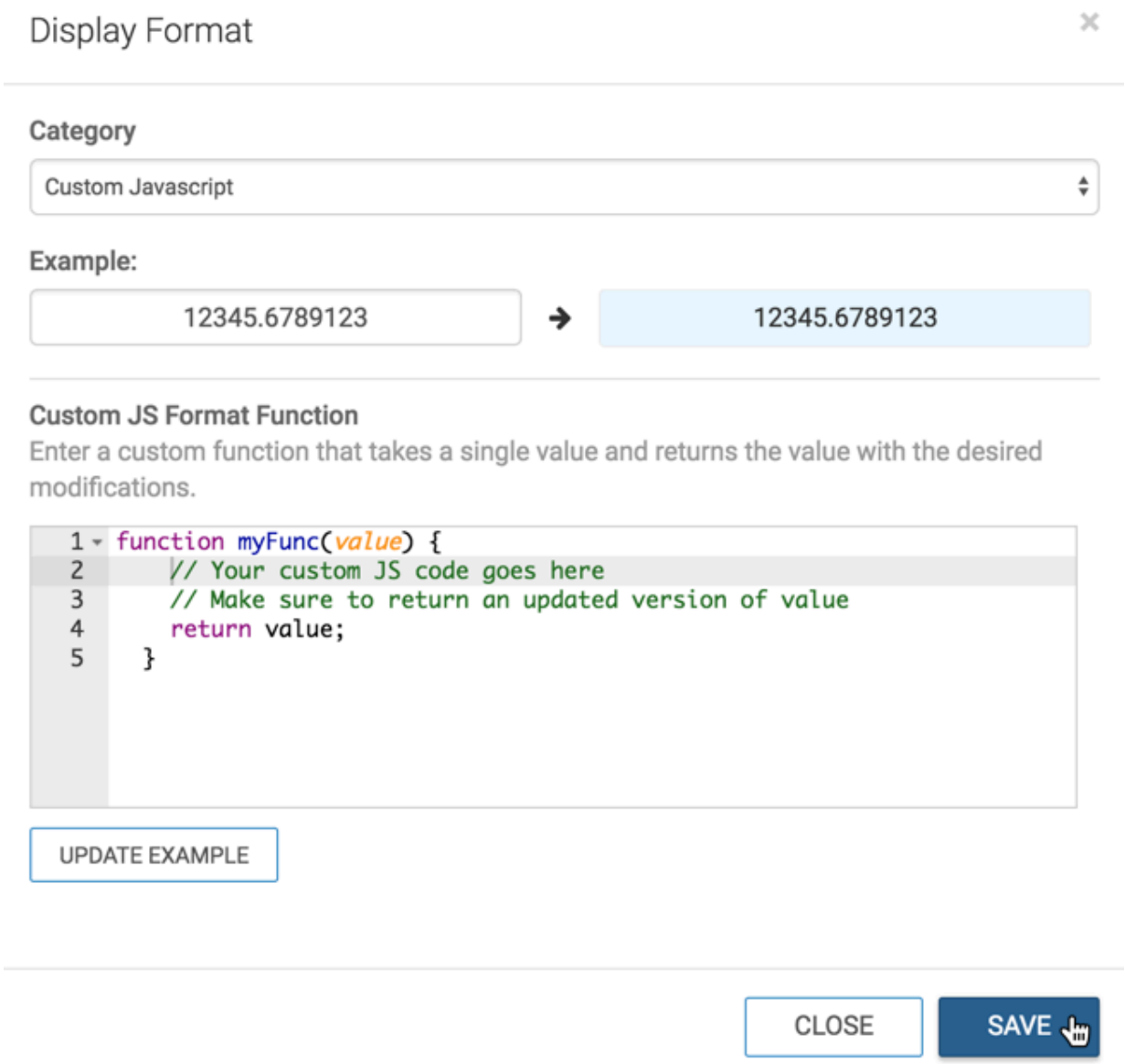

5. Click Save.

Specifying text color based on a scalar threshold

In the following example, we adapted the custom JS format to more clearly identify which US States have a Hispanic population over or under the national average of 16%.

```
function myFunc(value) {
    // change display color based on value
    // being under or over national average of 16%
    // expressed as percentage, with two decimal places
    if (value < .16) {
        // less than 16% is blue
        return `<div style="color: blue">${Number(value * 100).toFixed(2)}%<
/div>`;
        }
    else {
        // 16% or greater is green
```

```
        return `<div style="color: green">${Number(value * 100).toFixed(2)}
%</div>`;
        }
  }
```

Compare the results in the % Hispanic column (uses percentage display and the Rel. to 16 National Average column, that uses custom Javascript code in this example.

State	Population	Hispanic Population	% Hispanic	Rel. to 16% National Average
Alabama	4,822,023	196,032	4.07%	4.07%
Alaska	731,449	44,869	6.13%	6.13%
Arizona	6,553,255	1,976,106	30.15%	30.15%
Arkansas	2,949,131	199,693	6.77%	6.77%
California	38,041,430	14,537,666	38.22%	38.22%
Colorado	5,187,582	1,088,744	20.99%	20.99%
Connecticut	3,590,347	510,645	14.22%	14.22%
Delaware	917,092	78,813	8.59%	8.59%
District of Columbia	632,323	62,726	9.92%	9.92%
Florida	19,317,568	4,484,199	23.21%	23.21%

Using alias

Cloudera Data Visualization provides you with an option to alias fields to remove uninformative field names (or overly informative expressions) from the visual.

About this task

To set the alias, follow these steps:

Procedure

1. Click the field you plan to modify to open the FIELD PROPERTIES menu.

2. Under FIELD PROPERTIES, click Alias.

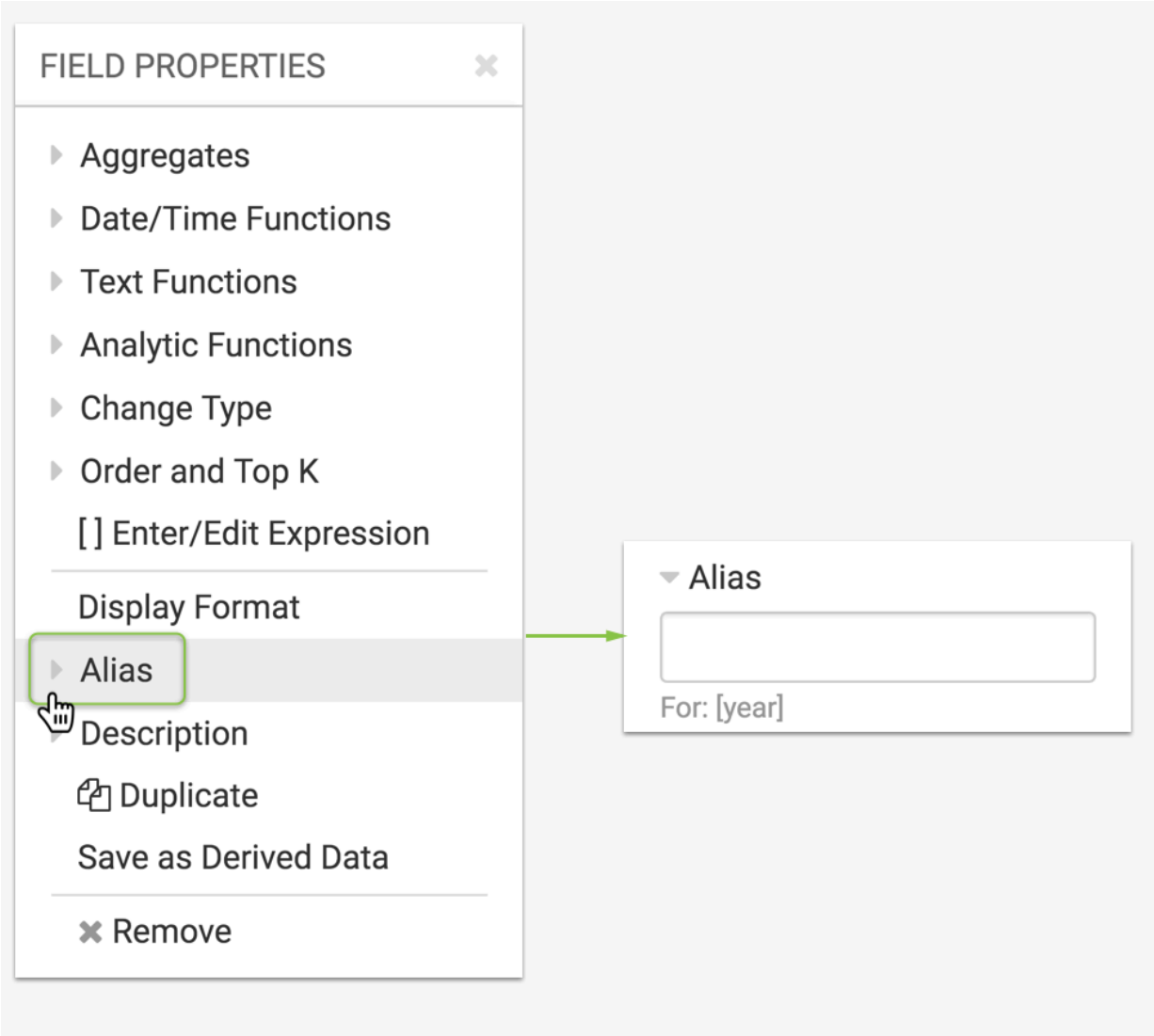

3. Enter the alias in the text field.

   Note: To improve behavior of dataset fields as filters, parenthesis are not allowed in the display name. You can add parenthetical information as a field comment.

4. Click REFRESH VISUAL to see the changes.

   Fields that use an alias have a green dot on the Alias menu.

   To remove an alias from a field, repeat the previous steps and clear the text field under Alias.

Changing column position

In a table visual, you can change the position of a column for better visualization of data in Cloudera Data Visualization.

About this task

By default, measure columns appear after all the dimension columns. In a table with large number of dimensions, it may be difficult to analyse the data if measure columns appear towards the end of the table.

Let's take an example of the cereal data and demonstrate how changing the position of the column improves the analyis of data.

To change the position of a column in a table, follow these steps:

Procedure

1. Click the measure that you plan to move to open the Field Properties menu.

   In this example, we are moving calories.
2. Under Field Properties, click to expland the Column Position menu.

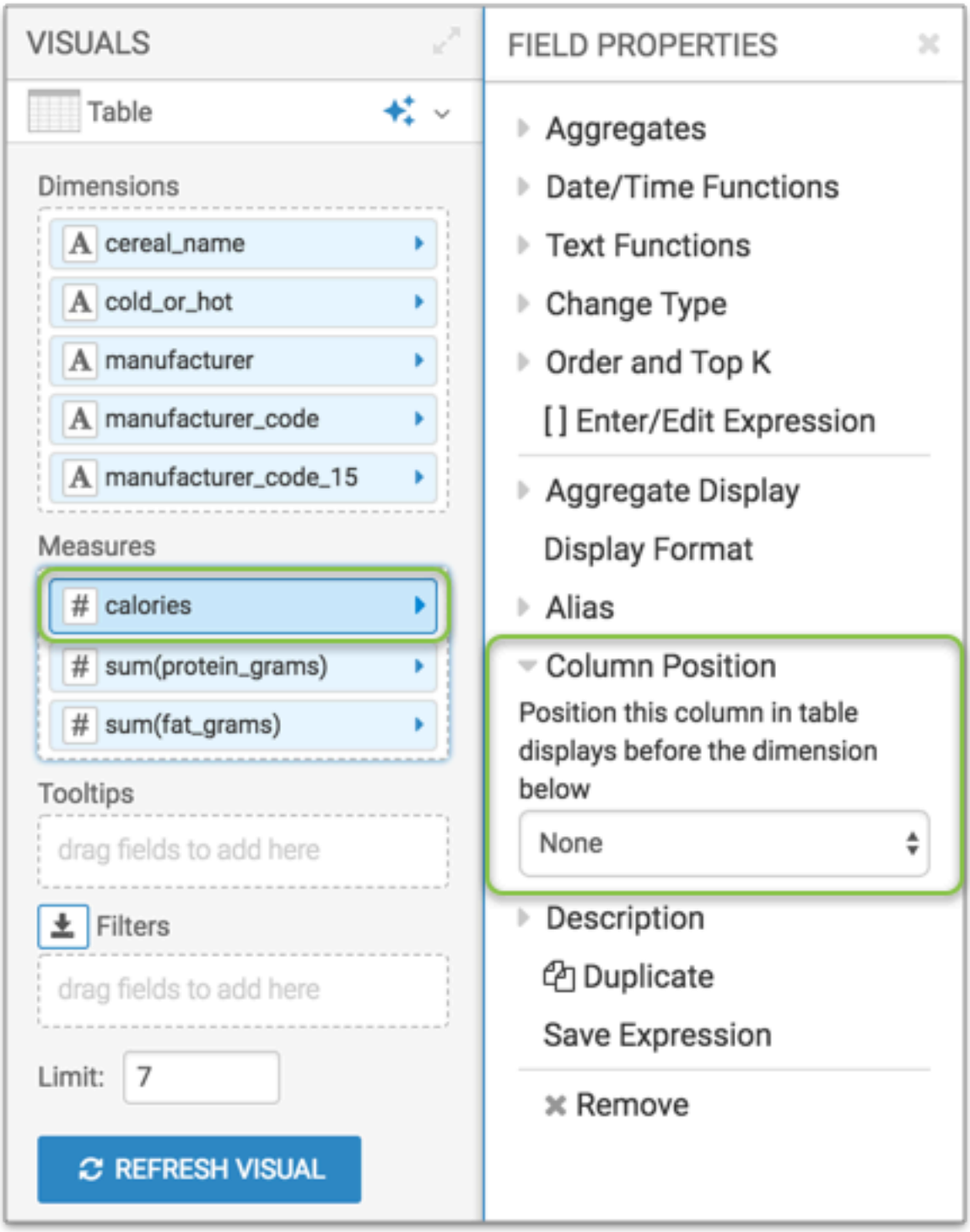

3. From the list of dimensions in the dropdown menu, select cold_or_hot.

   A green dot appears beside the field indicating a change in the value.

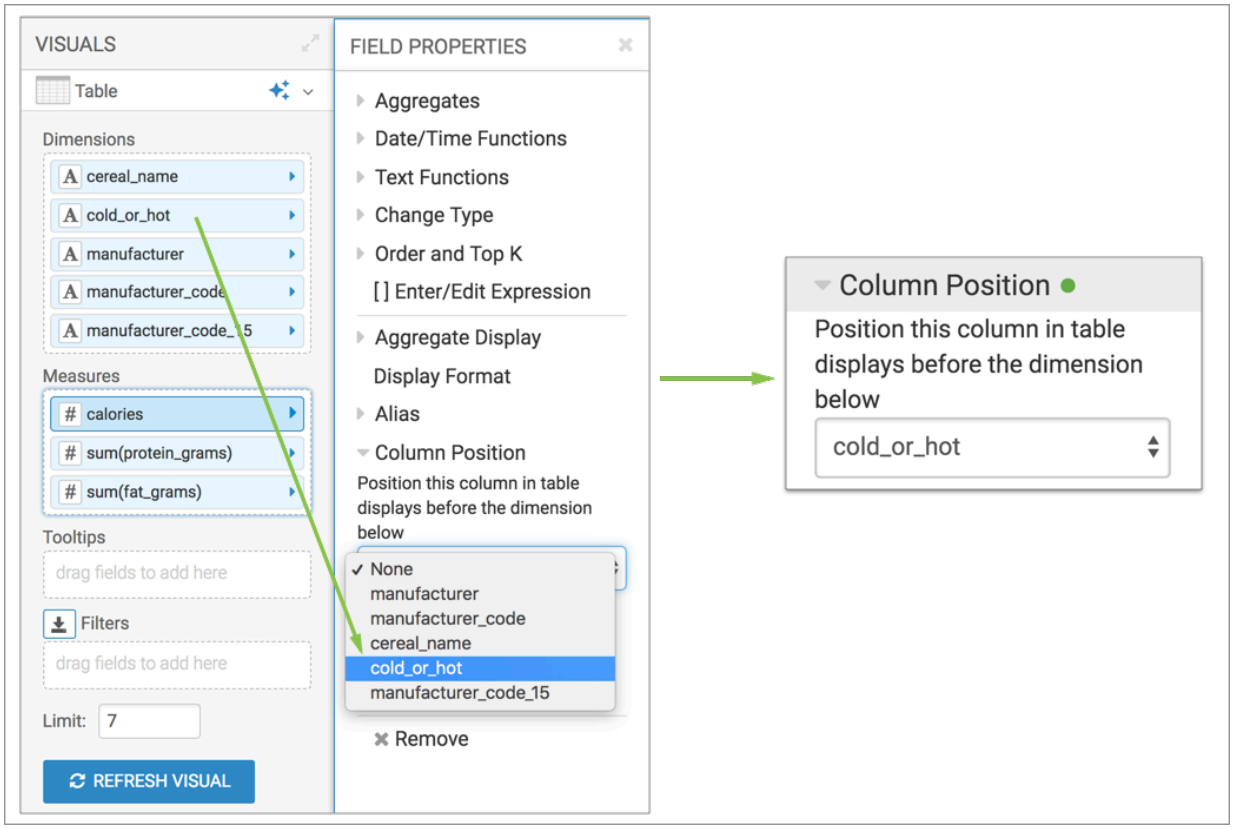

4. Click Refresh Visual to see the changes.

   Notice in the first image, calories column appears after all the dimension columns. However, if you want to compare the calories in each cereal, by moving the calories column before the cold_or_hot. column (second image), it becomes easier to view that data.

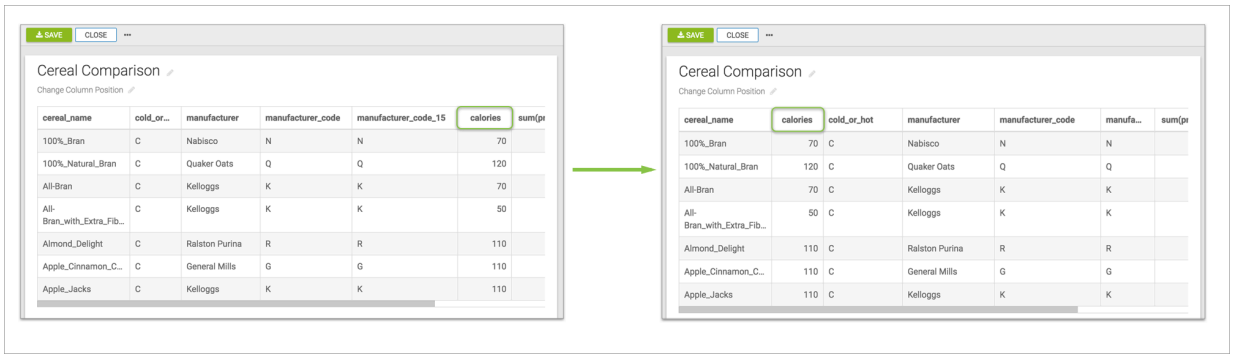

Customizing description

Cloudera Data Visualization provides you with an option to add a meaningful description to chart elements; this description can be viewed when hovering the pointer over the table or chart.

About this task

The following image demonstrates how you can (or cannot) customize descriptions on fields that define the following elements:

1. axes
2. trellis dimensions
3. data points (suppressed by tooltip functionality)
4. segment elements (suppressed by tooltip functionality)

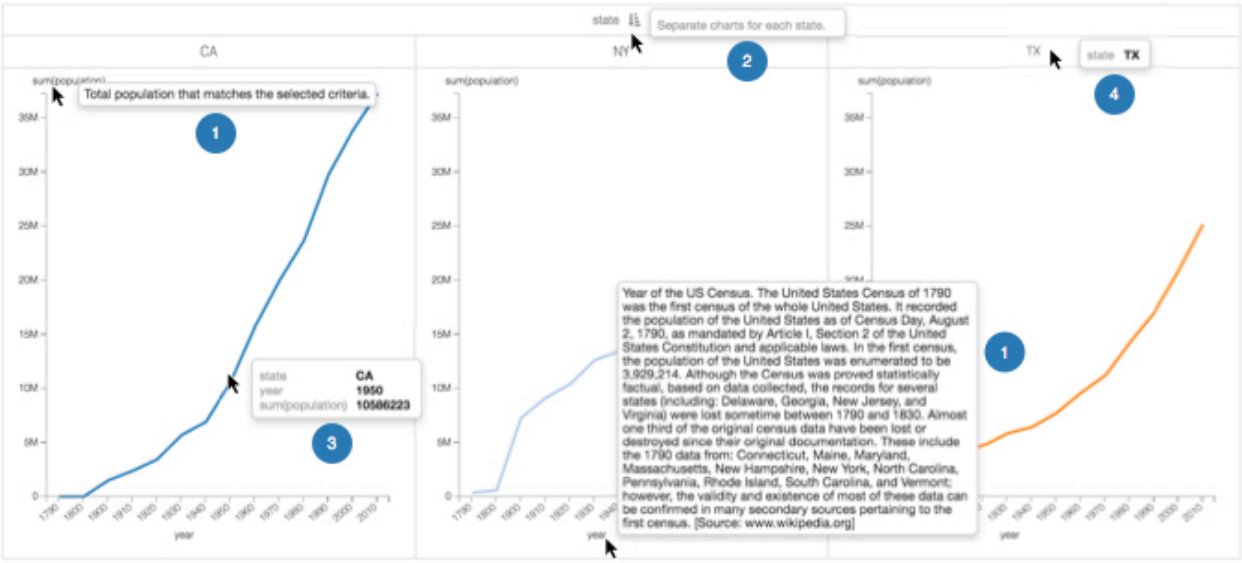

By default, a field's description is identical to how it is represented on a shelf of the visual.

To set a custom description, follow these steps:

Procedure

1. Click the measure that you plan to move to open the Field Properties menu.

   In this example, we are moving calories.

2. Under Field Properties, click to expland the Description menu.

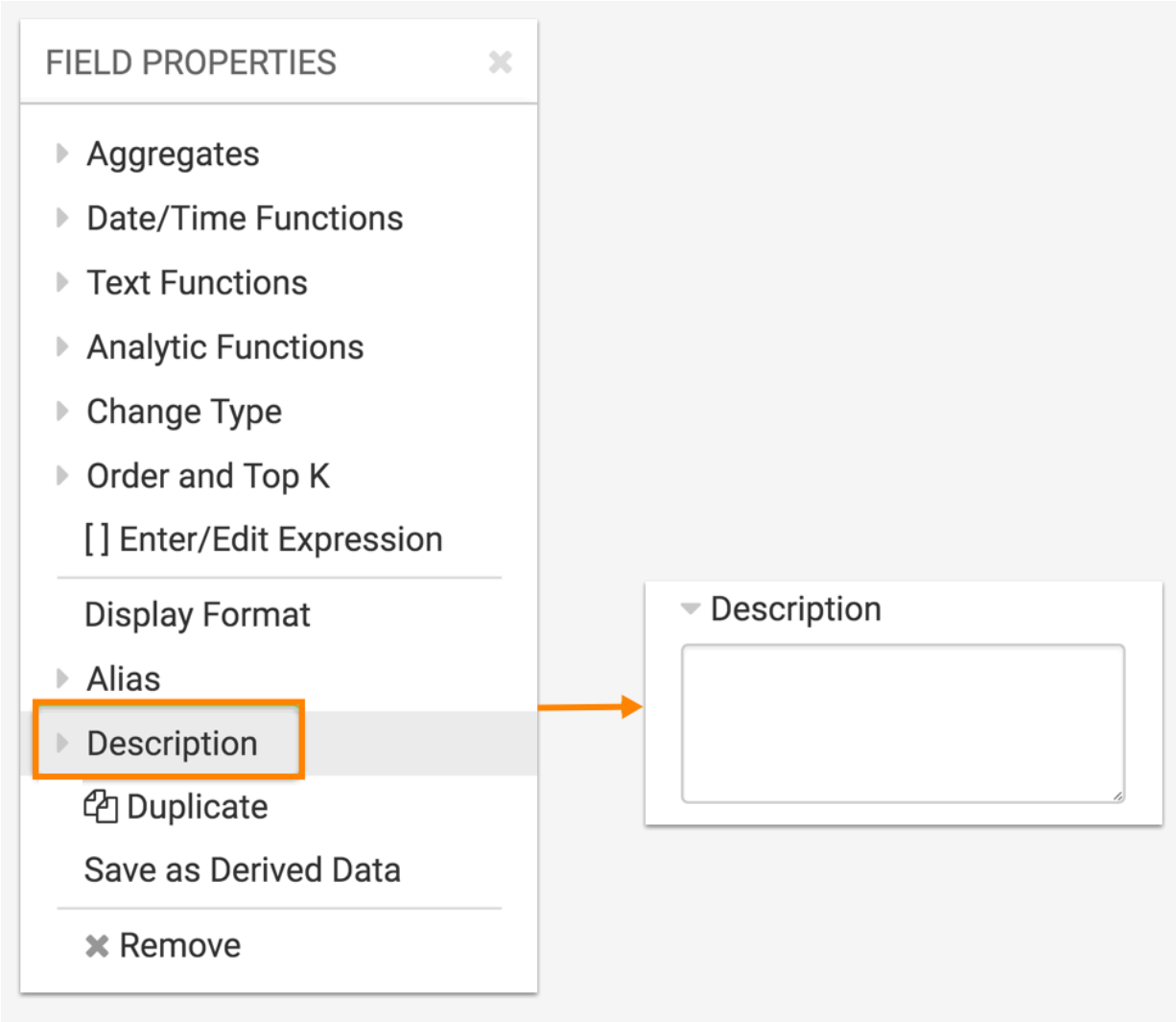

3. Enter the description in the large text field.
4. Click Refresh Visual to see the changes.

   Fields that use a description have a green dot on the Description menu.

Adding a description for a Ffeld

We entered the description as follows:

```
Total population that matches the selected criteria.
```

Note the change in the description of the field, between the default and the one we just added.

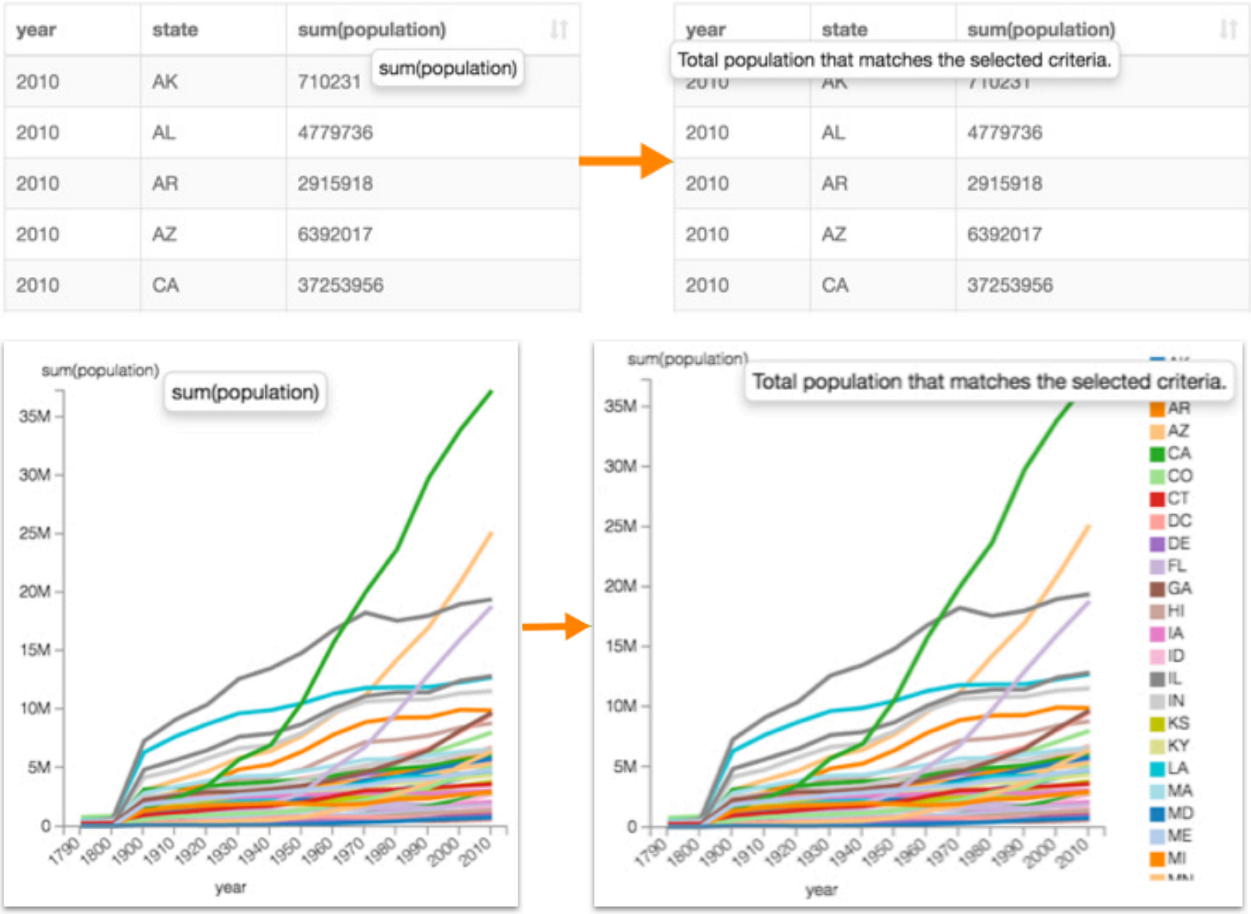

Duplicating an item

In Cloudera Data Visualization you can duplicate an item on the shelf, and then re-use it for another calculation or for placement on another shelf of the visual.

About this task

To duplicate a field, follow these steps.

Procedure

1. Click the field you plan to modify, to open the Field Properties menu.

2. Under Field Properties, click Duplicate.

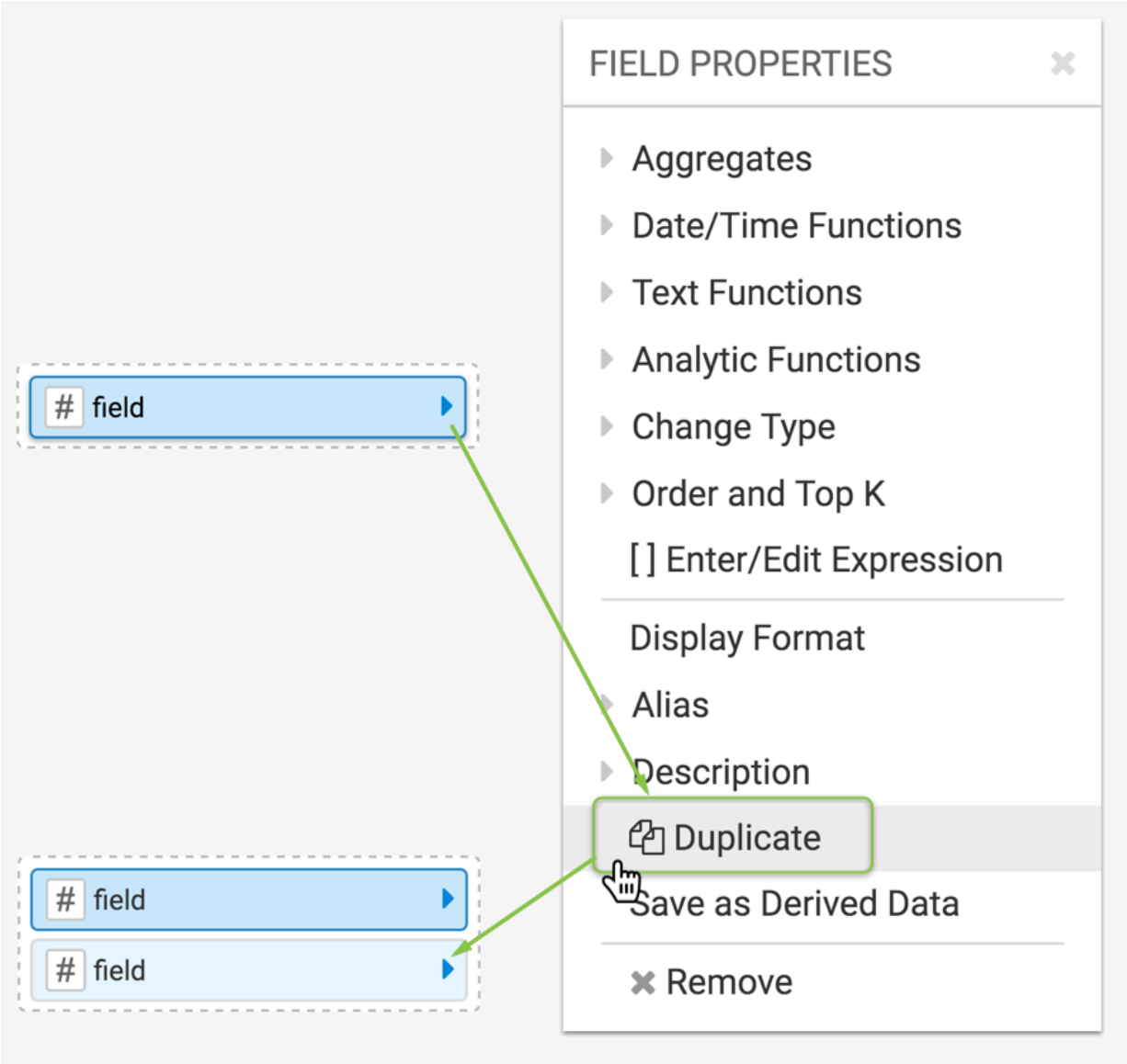

3. Depending on the requirements of your work, edit the field, and/or move it to another shelf.

Cloudera Data Visualization 7.2.9

Customizing Homepages

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Setting your homepage from a dashboard.....4

Setting your homepage on the Set Homepage interface..... 5

Setting a default homepage for all users.....5

Resetting homepage to default..... 6

Setting your homepage from a dashboard

In Cloudera Data Visualization, all users can change their home screen by selecting the current dashboard.

About this task

Important:

- The Set as Homepage option is only available if the administrator has not disabled homepage user-override. For more information, see *Setting a default homepage for all users*.
- The homepage set by a general user overwrites the default set by an administrator.

The following steps demonstrate how to set your own homepage directly from the dashboard you are viewing or editing.

Procedure

1. While viewing or editing a dashboard, click the supplemental menu, and select Set as Homepage.

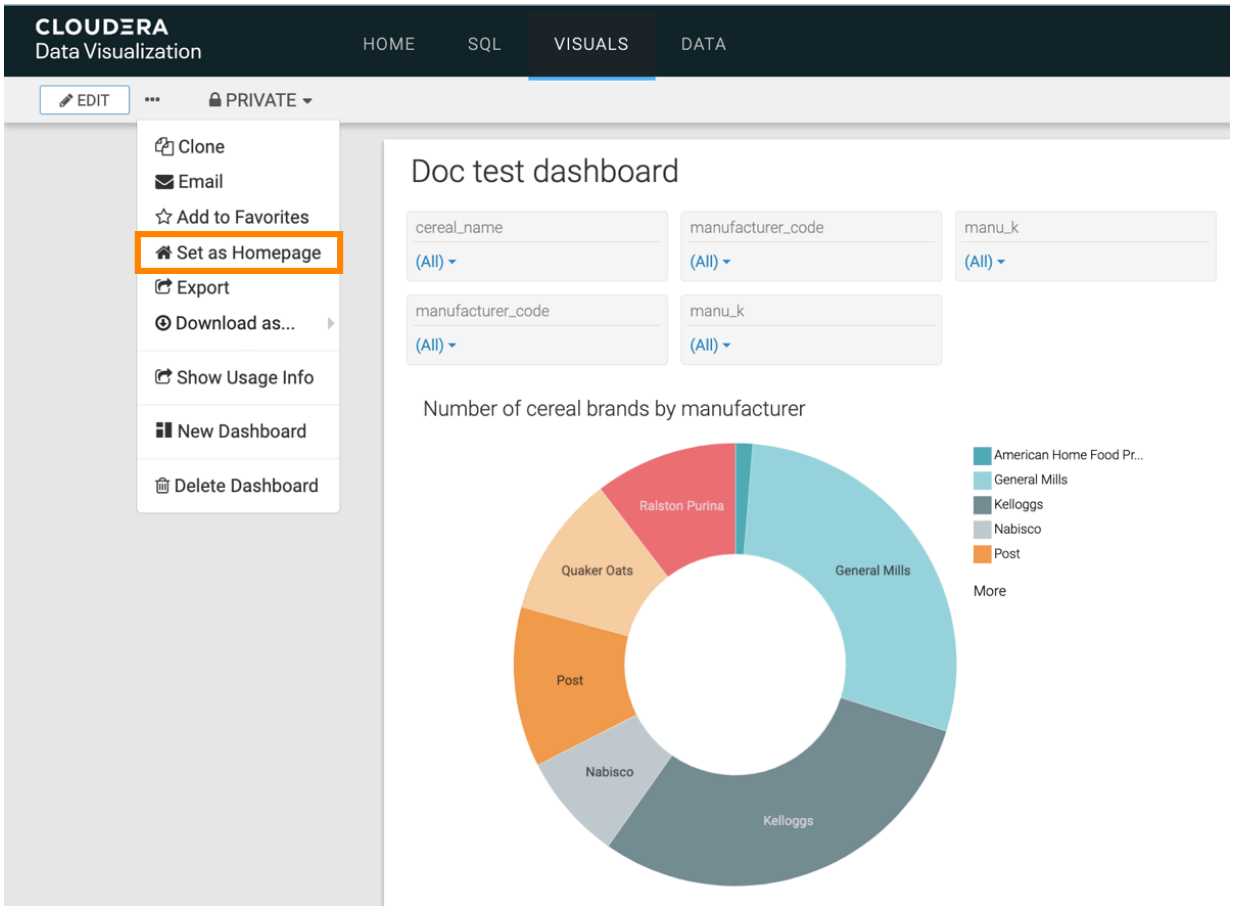

A message confirming the success of setting the dashboard as your homepage appears briefly.

2. Click HOME at the top navigation bar to check that the dashboard you just specified appears.

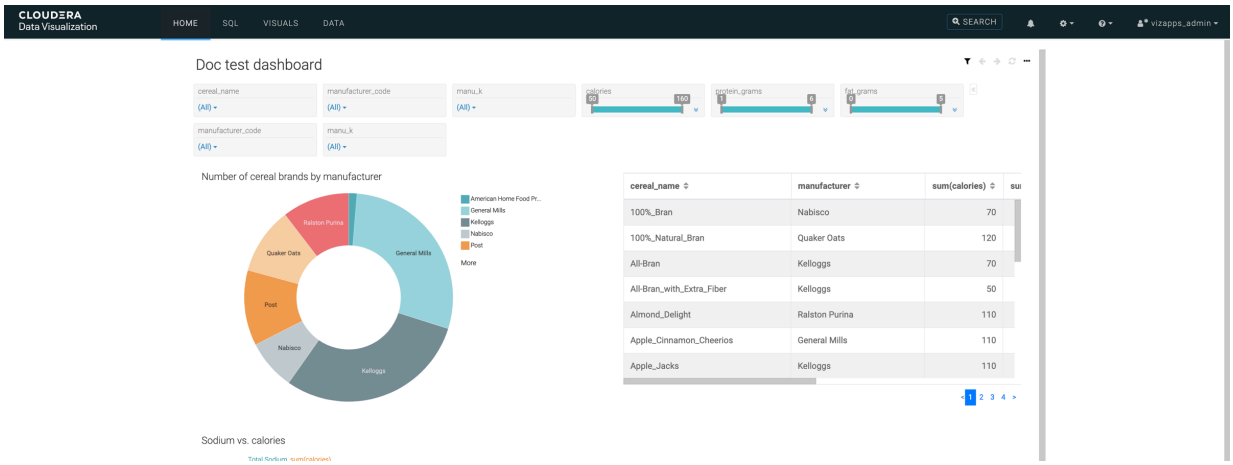

Related Information

Setting a default homepage for all users

Setting your homepage on the Set Homepage interface

In Cloudera Data Visualization, all users can change their home screen on the Set Homepage interface.

About this task

Important:

- The Set Homepage option is only available if the administrator has not disabled homepage user-override. For more information, see *Setting a default homepage for all users*.
- The homepage set by a general user overwrites the default set by an administrator.

The following steps show you how to set your own homepage from the Settings menu.

Procedure

1. On the main navigation bar, click the Gear icon to open the Settings menu.
2. Click Set Homepage in the drop-down menu.

   The Set Homepage interface appears.
3. Select the dashboard from the list to use for the homepage.
4. Click SAVE.

   A message confirming the success of setting the dashboard as your homepage appears briefly.
5. Click HOME at the top navigation bar to check that the dashboard you just specified appears.

Related Information

Setting a default homepage for all users

Setting a default homepage for all users

In Cloudera Data Visualization, administrators can change the users' default landing page globally.

About this task

Note: This feature is only available to users with administrative privileges.

The following steps show you how to set the default start page to a specific dashboard for all users.

Procedure

1. Log in as an admin user.
2. On the main navigation bar, click the Gear icon to open the Settings menu.
3. Click Set Homepage in the drop-down menu.

   The Set Homepage interface appears.
4. Scroll down to the Admin Only section and select the dashboard that you want to use as the new default homepage.
5. Click SAVE.

   A message confirming the success of setting the dashboard as your homepage appears briefly.
6. If you mark Disable user-override of homepage, you can prevent users from overriding the default homepage that you have set.
7. Log out from CDV and log in as a user with simple viewing privileges.
8. If this user does not have a custom homepage set, clicking HOME at the top navigation bar takes you to the new default homepage you have just specified.

Resetting homepage to default

Both users and administrators can change the Cloudera Data Visualization homepage back to its default value.

Procedure

Reseting your own homepage to default

When a simple user resets their homepage to default, it reverts to the setting that applies to all users.

The following steps demonstrate how you can set your homepage back to default:

1. Log in as a user with simple viewing privileges.
2. On the main navigation bar, click the Gear icon to open the Settings menu.
3. Click Set Homepage in the drop-down menu.

   The Set Homepage interface appears.
4. Click Reset to admin-defined homepage.

   You can see that the dashboard changes to the default global homepage.

   A message confirming the success of setting the dashboard to the admin-defined default appears briefly.
5. Click HOME at the top navigation bar to check that the admin-defined dashboard you have just specified appears.

   The HOME view changes to the default homepage defined by the administrator.

Resetting all user's homepages to Cloudera Data Visualization default

When an administrator resets the homepage to the default homepage, the new homepage will be the original Cloudera Data Visualization starting page for all users.

6. Log in as an admin user.
7. On the main navigation bar, click the Gear icon to open the Settings menu.

8. Click Set Homepage in the drop-down menu.

   The Set Homepage interface appears.
9. Scroll down to the Admin Only section.
10. Click Reset to default homepage.

    A message confirming the success of setting the homepage to default appears briefly.
11. If you mark Disable user-override of homepage, you can prevent users from overriding the default homepage that you have set.

    A message confirming the success of setting the dashboard as your homepage appears briefly.
12. Log out from CDV and log in as a user with simple viewing privileges.
13. Log in as a user with simple viewing privileges.
14. If the user does not have a custom homepage set, clicking HOME at the top navigation bar takes you to the Cloudera Data Visualization Welcome page you as an admin have just reset.

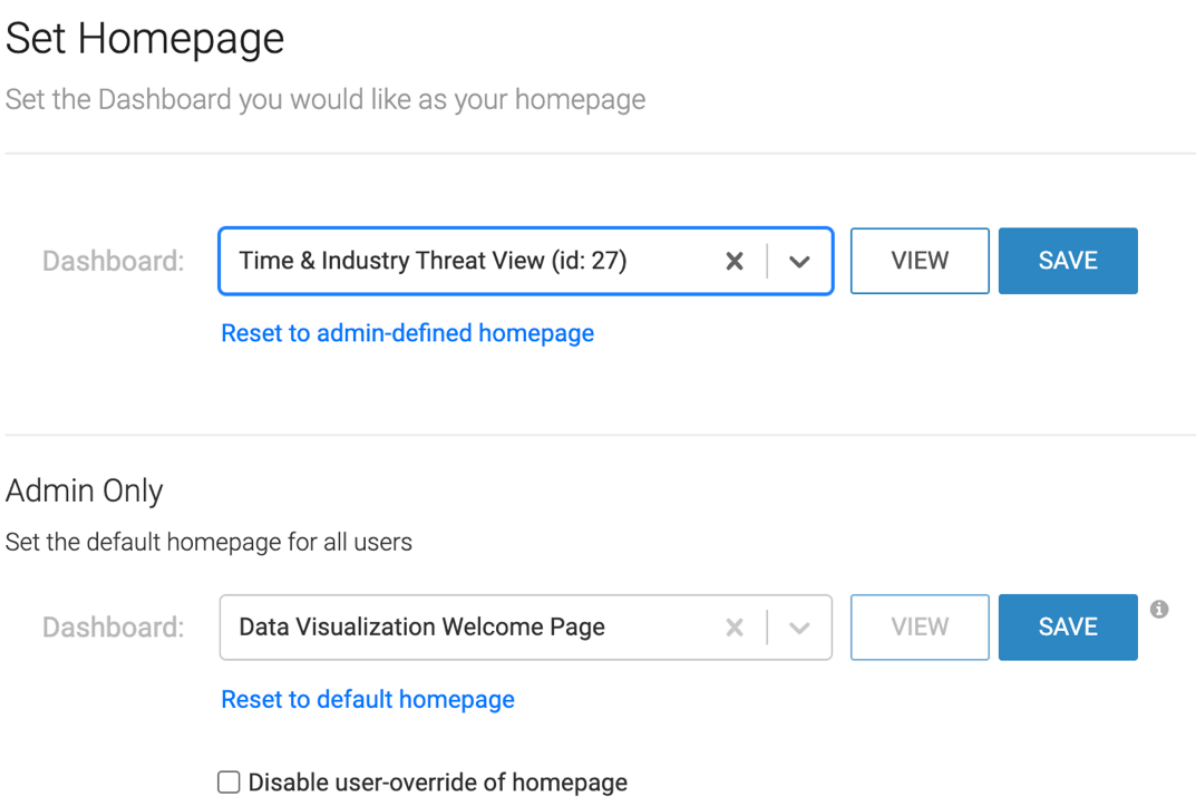

Cloudera Data Visualization 7.2.9

Customizing Visuals

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Customizing visual settings....9
Customizing general settings....9
Enabling animations....10
Enabling Drill Into data discovery....10
Customizing basic visual settings....11
Changing the bucket count....11
Showing normalized histograms....13
Using the cumulative option....15
Customizing downloads settings....17
Enabling CSV download....17
Enabling quotation marks in CSV download....19
Enabling Excel download in visuals....20
Setting the locale for CSV downloads....22
Customizing axes for visuals....23
Changing the axis scale....26
Changing the axes scale for radial graphs....27
Enabling gridlines....30
Enabling continuous axis for date/time values....30
Changing the padding on the X axis....31
Changing the padding on the Y axis....31
Starting aggregate scale at zero....32
Changing the min/max of the aggregate scale....32
Showing or hiding axis labels....33
Changing the size of dimension text....34
Showing or hiding an axis....35
Reversing the axis scale....35
Dynamically sizing a graph....36
Setting maximum row label length....36
Setting column label length....37
Enabling a continuous X axis....37
Changing scale of a continuous X axis....38
Enabling a continuous Y axis....39
Changing scale of a continuous Y axis....40
Enabling a continuous axis on dimensions....41
Changing the scale of a continuous axis on dimensions....42
Customizing data for visuals....43
Ordering NULL rows....44
Changing number of rows to download....46
Changing maximum number of rows to fetch....47
Disabling incremental queries....48
Displaying NULL rows....48
Enabling URL links....51
Specifying qualitative range as percentages....52
Customizing links....53
Changing link color....54
Changing link opacity....56
Changing link width range....57
Changing link distance....58
Changing link strength....58

Adding arrow direction to links.....59
Changing direction arrow size.....59
Adding link legend.....59
Changing the name of the links legend.....60
Customizing map options.....60
Selecting map view.....61
Customizing marks.....62
Displaying tooltips.....63
Showing detail data button in context menu.....65
Showing the context menu.....66
Showing table data as HTML.....67
Showing adjacent duplicate values.....67
Showing expansion totals above expansion rows.....67
Displaying title for subtotal rows.....69
Adding and removing legend.....70
Changing legend item width.....73
Changing legend width.....73
Reversing legend order.....74
Displaying a linear trendline.....75
Separating trendlines by color.....76
Changing trendline number of decimals.....77
Showing a selection brush for ranges.....79
Showing measure values inside bars.....79
Changing bar size range.....79
Separating histogram columns.....80
Showing dimension values on hovering.....80
Showing points.....81
Ignoring missing date values.....82
Changing line shape.....83
Changing line style.....89
Changing highlight type.....90
Converting line to area visual.....91
Changing highlight of radius.....92
Changing mark size range.....93
Changing transition delay.....94
Showing autoplay transition animation.....95
Showing dimensions as marks.....95
Showing percentages as marks.....96
Changing number of decimals.....97
Changing display to overlap.....97
Showing the same source and destination values.....98
Changing size of chord visual.....99
Displaying bubbles on maps.....99
Changing layout gravity.....101
Changing layout theta.....101
Showing node labels.....102
Tokenizing input.....103
Displaying gauge needle.....104
Displaying gauge line on qualitative arc.....104
Displaying axis ticks.....105
Changing outer width.....105
Changing qualitative thickness.....105
Changing arc gap.....106
Changing arc thickness.....106
Changing label font size.....107
Changing label offset.....107

Changing sublabel font size.... 108
Changing sublabel offset.... 108
Making box plots horizontal.... 109
Changing outline width.... 110
Changing font size of main indicator.... 110
Changing value of main indicator.... 111
Using same color for main indicator as arrow.... 112
Displaying first measure as main indicator.... 113
Changing compare indicator value.... 114
Changing compare indicator position.... 115
Changing arrow type.... 117
Flipping compare indicator.... 119
Changing label indicator position.... 120
Changing label indicator font size.... 121
Ignoring negative values.... 121
Changing approximate tick count.... 123
Orienting bars vertically.... 124
Removing default label.... 125
Staying at same map location.... 126
Setting minimum cell size.... 128
Customizing nodes.... 129
Remembering node positions.... 129
Showing the same source and target values.... 130
Changing source node mark.... 131
Changing target node mark.... 134
Changing combined node mark.... 137
Changing node size range.... 139
Changing node friction.... 141
Changing node charge.... 143
Changing node opacity.... 145
Showing node labels.... 146
Adding nodes legend.... 147
Changing nodes legend name.... 149
Customizing parameters.... 150
Using parameters with explicit scope.... 151
Sending parameters with dataset scope.... 152
Sending aggregates as parameters.... 152
Customizing map server.... 153
Customizing map server.... 154
Specifying mapbox key for a visual.... 155
Specifying atlas mapbox server for a visual.... 156
Customizing map style.... 157
Changing the map style of a visual.... 157
Adding custom mapbox style.... 161
Customizing choropleth.... 162
Specifying choropleth layer ID.... 163
Specifying choropleth opacity.... 164
Displaying legend in choropleth map.... 165
Remembering location of map.... 167
Customizing heatmap.... 170
Displaying heatmap.... 171
Changing heatmap radius.... 172
Changing intensity magnification.... 175
Showing heatmap legend.... 178
Changing visible zoom levels in heatmap.... 182
Customizing cluster.... 185

Displaying cluster.... 186
Changing cluster color....187
Changing cluster radius.... 190
Customizing circles....192
Displaying circles....192
Displaying color legend....194
Displaying area legend.... 196
Changing circle radius range....198
Changing visible zoom levels in circles.... 204
Customizing routes and pins.... 207
Enabling routes.... 207
Enabling pins.... 210
Selecting pin types....212
Selecting start pin type....215
Selecting middle pin type....218
Selecting end pin type.... 221
Specifying a custom icon.... 224
Changing custom icon size....227
Customizing tooltip....229
Showing percentage.... 229
Specifying label for percentage....230
Changing number of decimals for percentage.... 232
Changing tooltip font size.... 234
Changing tooltip font family....236
Customizing trellis.... 241
Displaying trellis borders....241
Enabling independent axis scale.... 242
Setting minimum trellis cell width....245
Setting minimum trellis cell height....246
Customizing zoom.... 248
Enabling pan and zoom....248
Enabling zoom to active areas.... 250

Customizing visual style....253
Customizing visual basics.... 253
Changing font family in a visual.... 253
Changing font size in a visual.... 258
Changing background color of a visual....260
Customizing table style.... 262
Changing table templates....262
Changing cross tabulation header alignment.... 268
Adjusting table width.... 273
Wrapping column contents....276
Showing column totals.... 279
Showing header for column totals.... 281
Showing column page totals.... 283
Showing header for column page totals....285
Specifying totals column font weight.... 287
Specifying maximum number of columns.... 289
Showing row information....291
Showing pagination.... 294
Specifying number of rows per page....298
Showing header column.... 300
Showing header column borders....302
Adjusting table column width.... 304

Customizing cross tabulation style....305
Adjusting cross tabulation width....305
Adjusting cross tabulation column width....305
Changing cross tabulation header font....306
Changing table header font size....306
Changing cross tabulation header font weight....306
Changing cross tabulation cell padding....307
Changing cross tabulation border width....307
Showing cross tabulation column totals....307
Showing cross tabulation row totals....307
Specifying font weight for totals column in cross tabulation....310
Specifying font weight for totals row in cross tabulation....310
Changing cross tabulation header alignment....310
Changing cross tabulation content alignment....311
Customizing table and cross tabulation colors....311
Changing table header background color....311
Changing table header text color....314
Changing table odd rows background color....317
Changing table even rows background color....320
Changing table odd rows text color....323
Changing table even rows text color....326
Changing table border color....329
Changing totals row background color....332
Changing totals row text color....335
Customizing visual title....338
Changing font family in the title of a visual....338
Changing font size in the title of a visual....343
Changing font weight in the title of a visual....345
Changing text alignment in the title of a visual....347
Hiding the title of a visual....350
Customizing visual subtitle....352
Changing font family in the subtitle of a visual....352
Changing font size in the subtitle of a visual....356
Changing text alignment in the subtitle of a visual....358
Customizing visual axes and labels....360
Axis label font size in a visual....360
Ticks font size in a visual....362
Label font size in a visual....364
Label text color in a visual....365
Trellis font size in a visual....368
Legend font size in a visual....370
Customizing tooltip....373
Changing tooltip font family....373
Changing tooltip font size....377
Customizing aggregate row style....381
Changing font style in the aggregate row of a visual....381
Changing font weight in the aggregate row of a visual....385

Customizing visual colors....389
Enabling reverse color palette of visuals....389
Changing visual color palette....391
Enabling colors....392
Changing foreground color....393
Changing pointer color....394
Changing background color....395

Changing arrow color....397
Changing color opacity on visuals....398
Changing link color....399
Changing drop off color....400
Changing highlight color....402
Changing projection opacity....404
Changing background opacity....405
Ensuring consistent colors on dimension values....406
Assigning color to specified values....408
Changing colors for KPI visuals....410

Adding custom styles to visuals....416

Adding CSS classes to visuals....419
Adding included styles to visuals....420
Searching custom styles in visuals....422
Sorting custom styles in visuals....428
Adding inline CSS code to visuals....433
Inline CSS usage....436

Customizing visual settings

Cloudera Data Visualization supports several options for visual display settings.

Note: Keep in mind that not all visual setting adjustments make sense for all visual types. Cloudera Data Visualization only shows setting options that are meaningful in the context of a particular visual type.

Customizing general settings

About this task

You can manage several options from the General menu, depending on the chart type of the visual.

To get to the options in the General menu, follow these steps:

Procedure

1. On the right side of Dashboard Designer, click Settings General from the VISUAL menu.

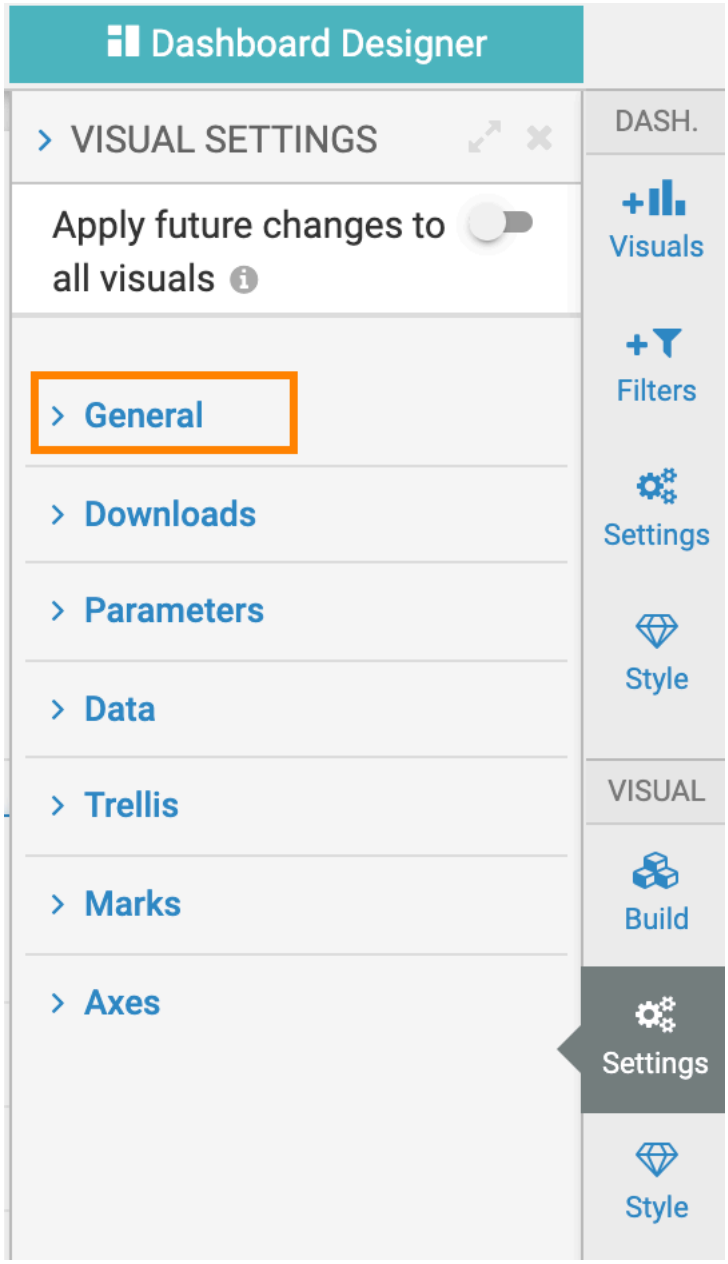

2. Select one of the options:

Enabling animations

To enable animations on your visual, navigate to the General menu, and select the Enable animations option.

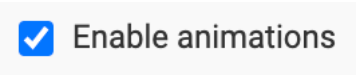

Related Information

Data Discovery with Context Menus

Enabling Drill Into data discovery

You can use this feature to drill into data in the context menu of a visual.

About this task

Note: This setting is available only on Bar, Line, Area, and Grouped Bar visual types.

For more information, see *Data Discovery Through Context Menus*.

To enable the Drill Into data discovery option, select the Show Drill Into context menu option in the General menu.

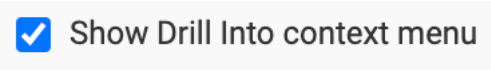

If this option is selected, Drill Into appears in the context menu. To disable the drill-into behavior, de-select this option.

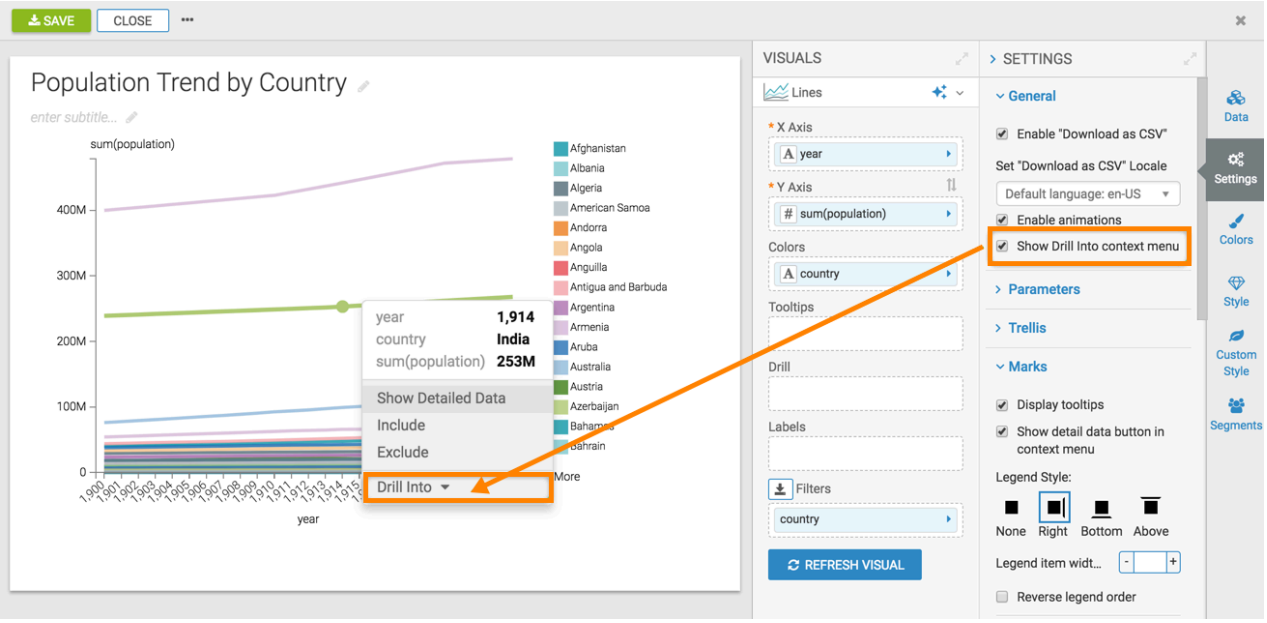

Customizing basic visual settings

Changing the bucket count

The default number of histogram buckets is 10, but you can easily change the number of buckets.

To change the number of histogram buckets, click Settings in the VISUAL menu, open the Basic section, and adjust the selector for the Bucket count option. The default value is 10.

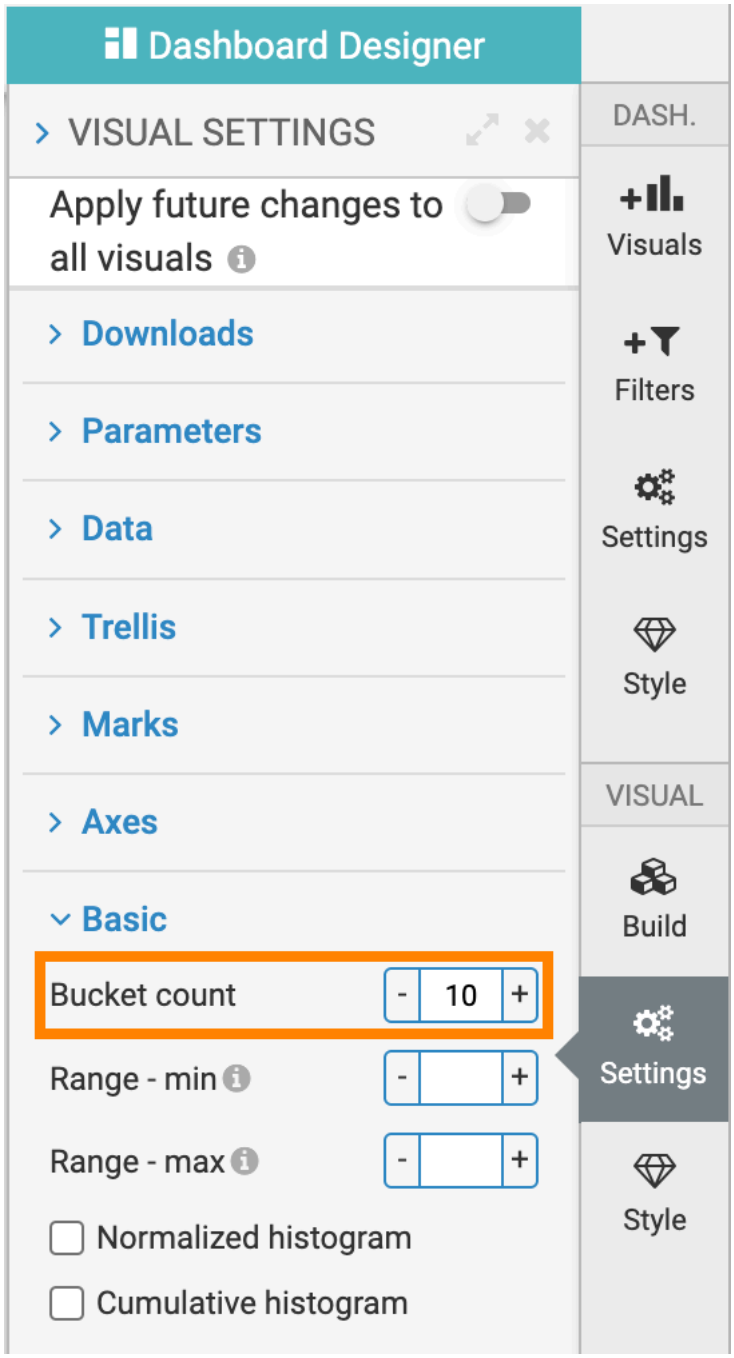

The following image contrasts the appearance of a histogram with 10 buckets and a histogram with 20 buckets. Notice that as the number of buckets doubles, the range of values covered by each bucket is reduced by half.

Showing normalized histograms

To report the histogram as a percentage of a whole, the histogram count is normalized to add up to 1, and then the bars represent the proportionate frequency.

To normalize a histogram, click Settings in the VISUAL menu, open the Basic section, and select the Normalized histogram option.

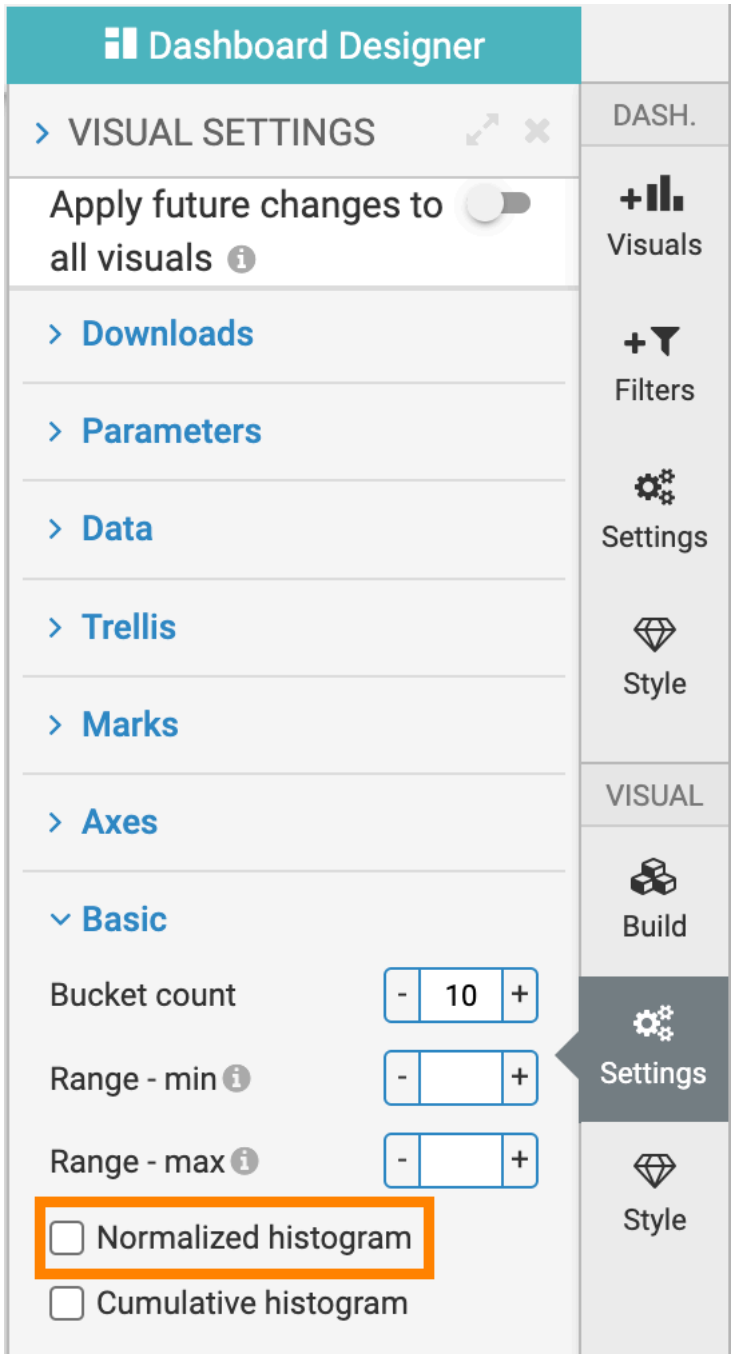

The following image shows the two versions of the histogram. You can see that the change appears in the vertical axis and the tooltip, where frequency is reported as a number of incidents on the upper graph, and as a percentage on the lower graph.

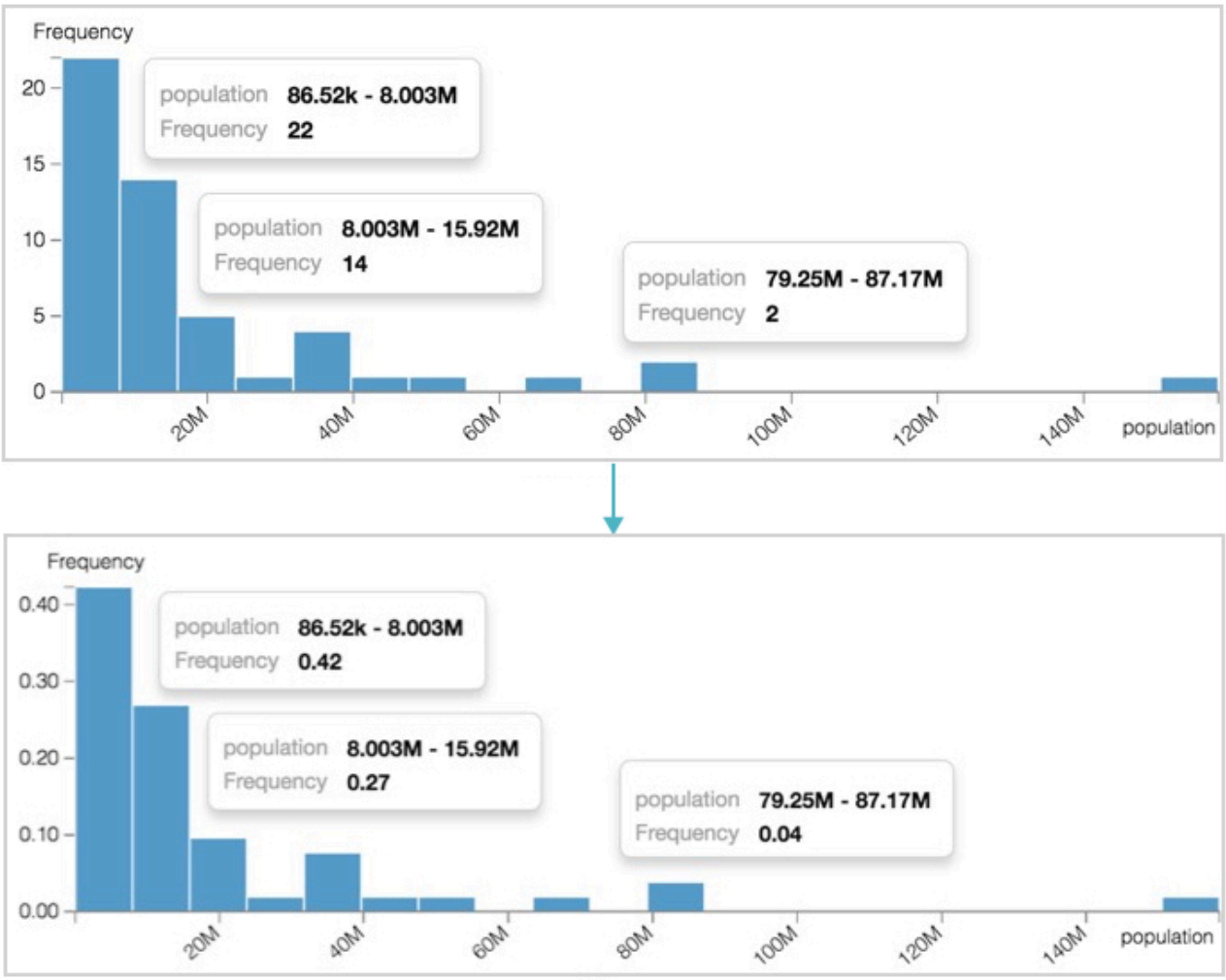

Using the cumulative option

About this task

By default, the histogram reports each bucket individually. The cumulative option adds each bucket's count or frequency to the running total, so that the right-most bucket reports the total count or 1 (100%), depending on whether the normalized histogram option is active. You can turn on the cumulative option on a histogram in the Basic menu.

Procedure

1. On the right side of Dashboard Designer, click Settings in the VISUAL menu.
2. In the Settings menu, click Basic.

3. Under Basic, select the Cumulative histogram option.

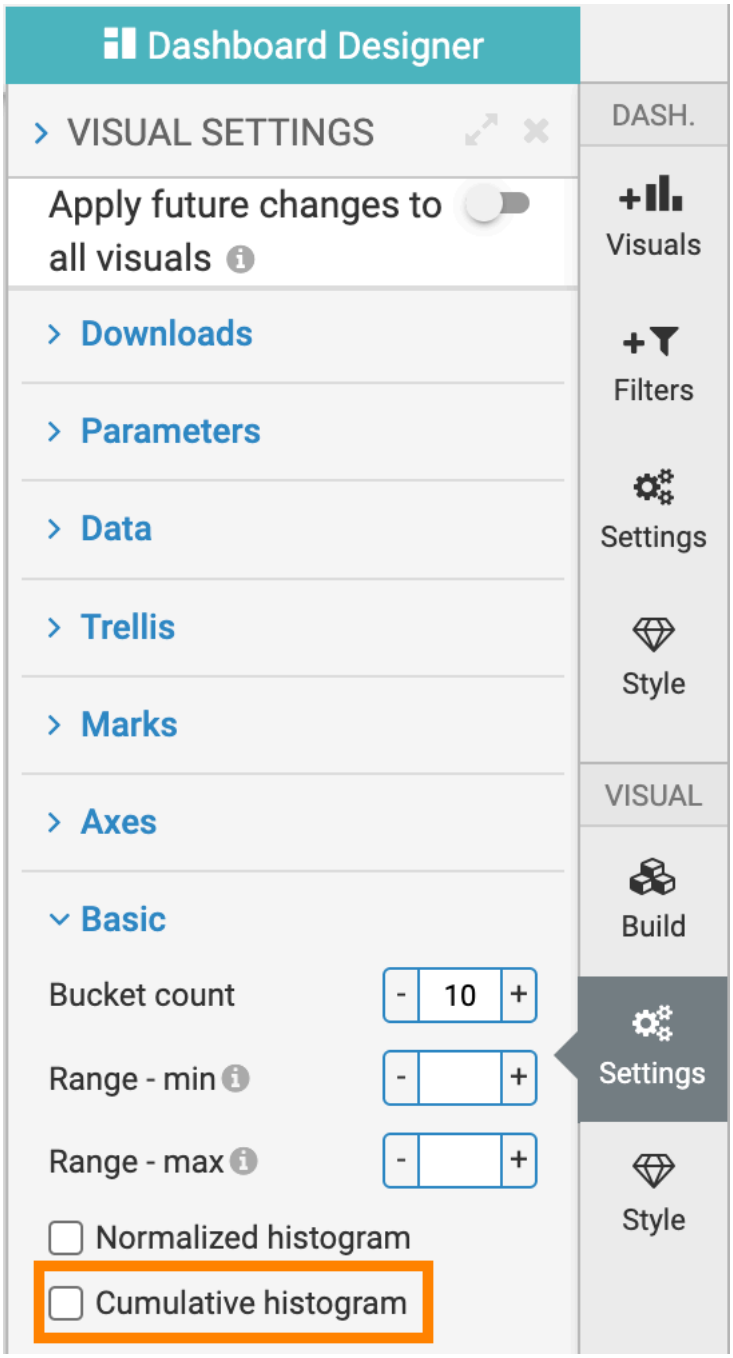

Example

You can see that the difference in the vertical axes between the upper and the lower images; in the lower image, the accumulated values approach 1 on the right side of the graph.

Customizing downloads settings

In Cloudera Data Visualization, there are several dowload setting options may be managed from the Downloads menu, depending on the chart type of the visual.

Enabling CSV download

In Cloudera Data Visualization, you can manage several dowload setting options from the Downloads menu, depending on the chart type of the visual.

About this task

You can download data in a visual, in CSV or Excel format. This setting is available on all visual types.

Note: The site administrator can override this setting at the site level. For more information, see *Downloading maximum number of rows in CSV files*.

Procedure

1. On the right side of Dashboard Designer, click Settings Downloads from the VISUAL menu.

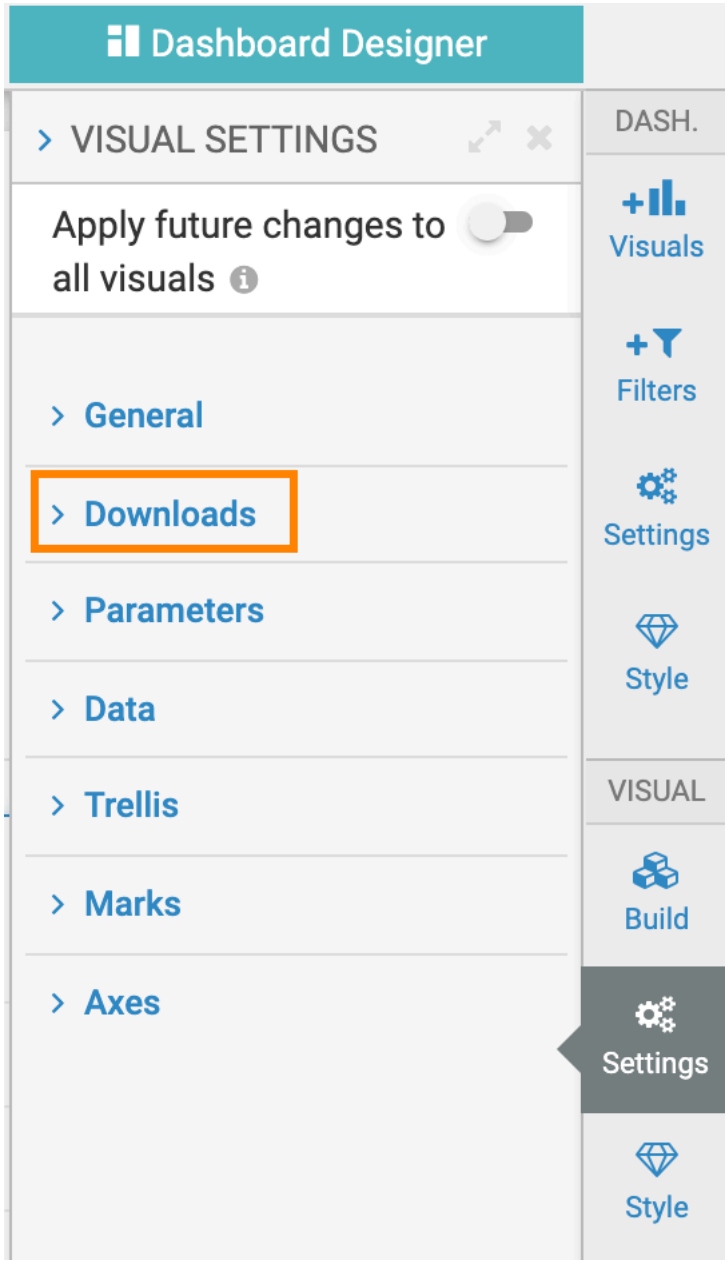

2. Select Enable "Download as CSV/Excel".

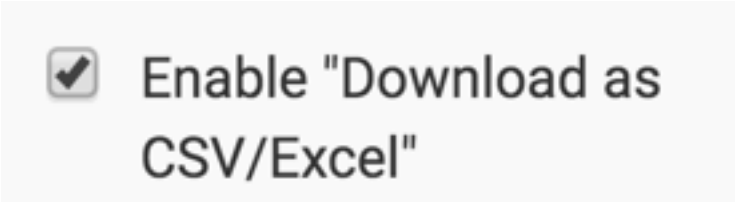

To disable CSV download, de-select the Enable "Download as CSV/Excel" option.

Related Information

Downloading maximum number of rows in CSV files

Enabling quotation marks in CSV download

If you download a CSV file and open in excel, leading zeros in a numeric string are truncated. To see the leading zeros, Cloudera Data Visualization enables you to add quotes in a numeric string.

Before you begin

Before enabling this setting, you must first enable CSV download, in the Downloads menu at the visual level.

Note: The site administrator can override the visual-level setting at the site level. For more information, see *Downloading maximum number of rows in CSV files*.

Procedure

1. On the right side of Dashboard Designer, click Settings Downloads from the VISUAL menu.

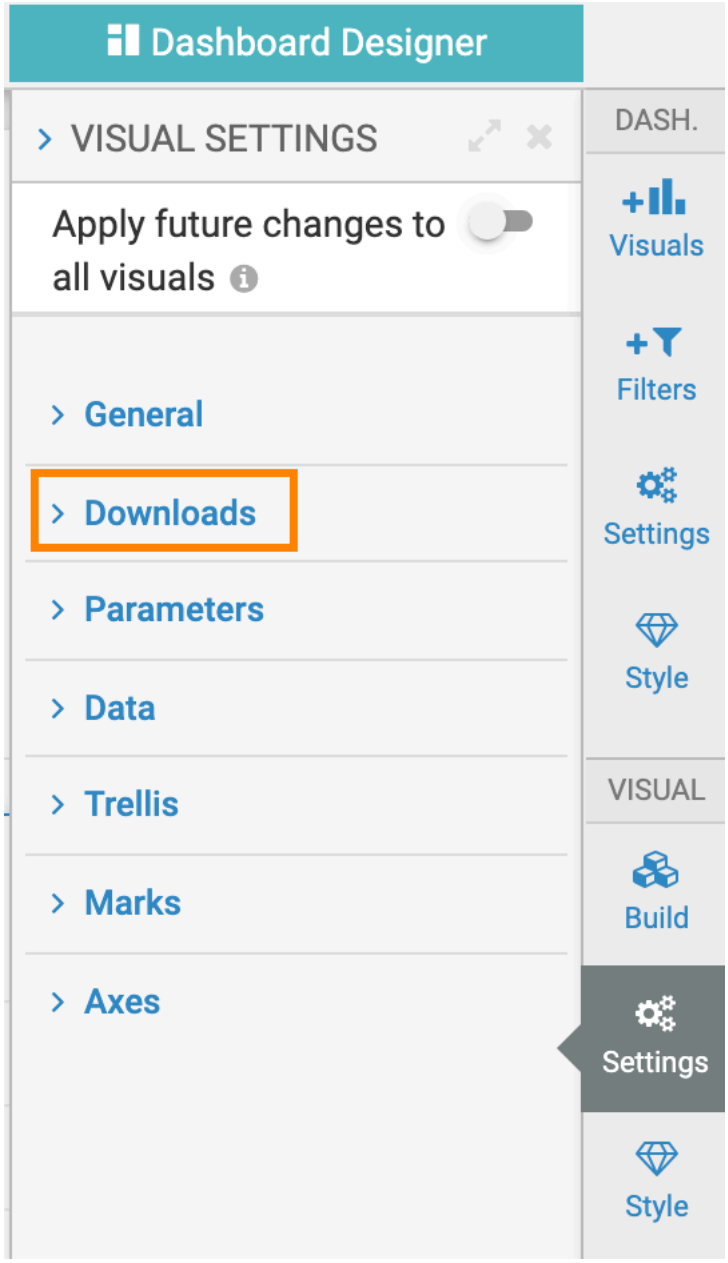

2. Select Quote numeric strings in CSV downloads.

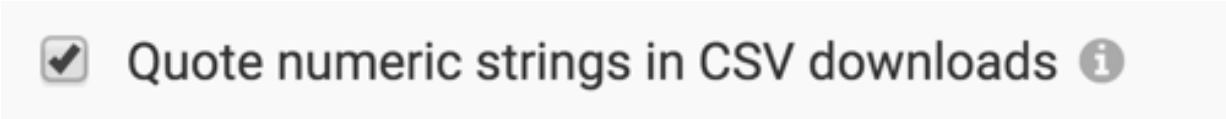

To disable the support of quotation marks in a CSV file, de-select the Quote numeric strings in CSV download option.

Related Information

Downloading maximum number of rows in CSV files

Enabling Excel download in visuals

You can enable or disable Excel data download for each visual in an XLS file.

About this task

This setting is available on all visual types.

Note: The site administrator can override this setting by defining Enable "Download as CSV/Excel" in the Site Settings interface. For more information, see *Downloading maximum number of rows in CSV files*.

Procedure

1. On the right side of Dashboard Designer, click Settings Downloads from the VISUAL menu.

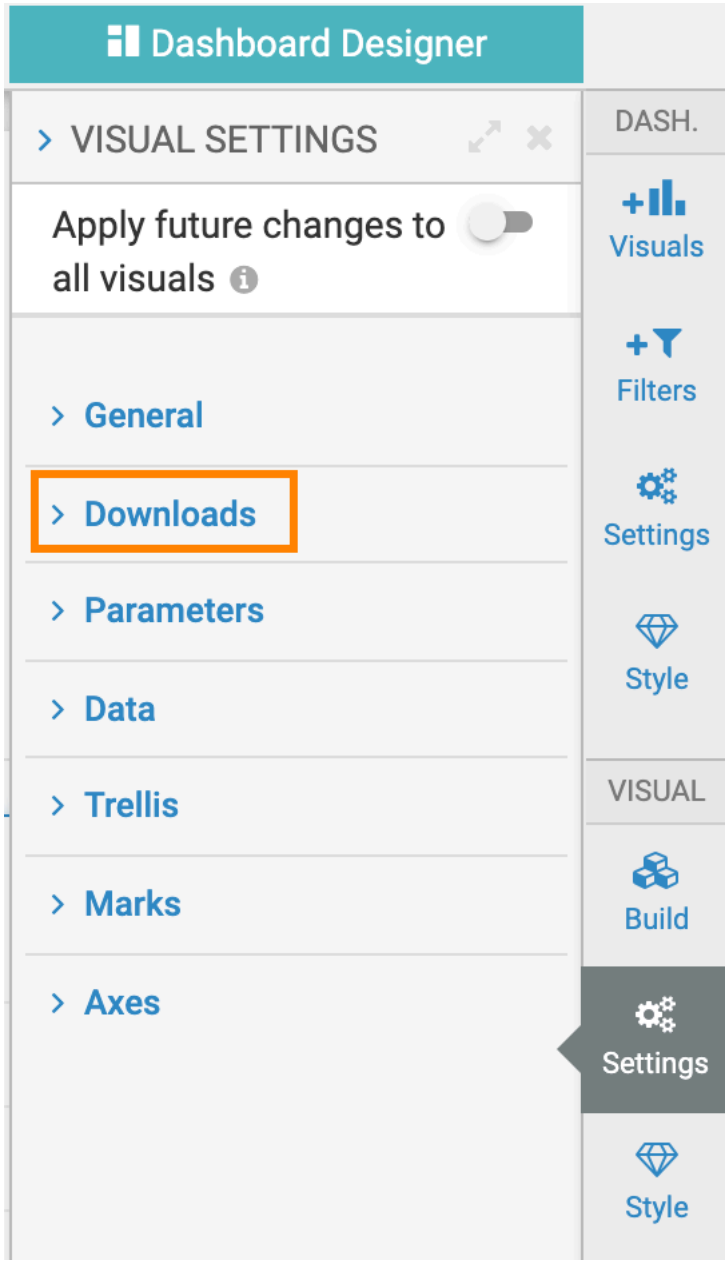

2. Select Enable "Download as CSV/Excel".

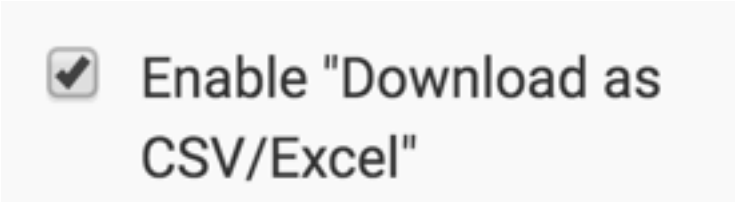

To disable Excel download, de-select this option.

Related Information

Downloading maximum number of rows in CSV files

Setting the locale for CSV downloads

By default, the locale settings for CSV download format match the system settings of the Internet browser. However, it is possible to specify a different locale for download.

About this task

This setting is available on all visual types.

Procedure

1. On the right side of Dashboard Designer, click Settings Downloads from the VISUAL menu.

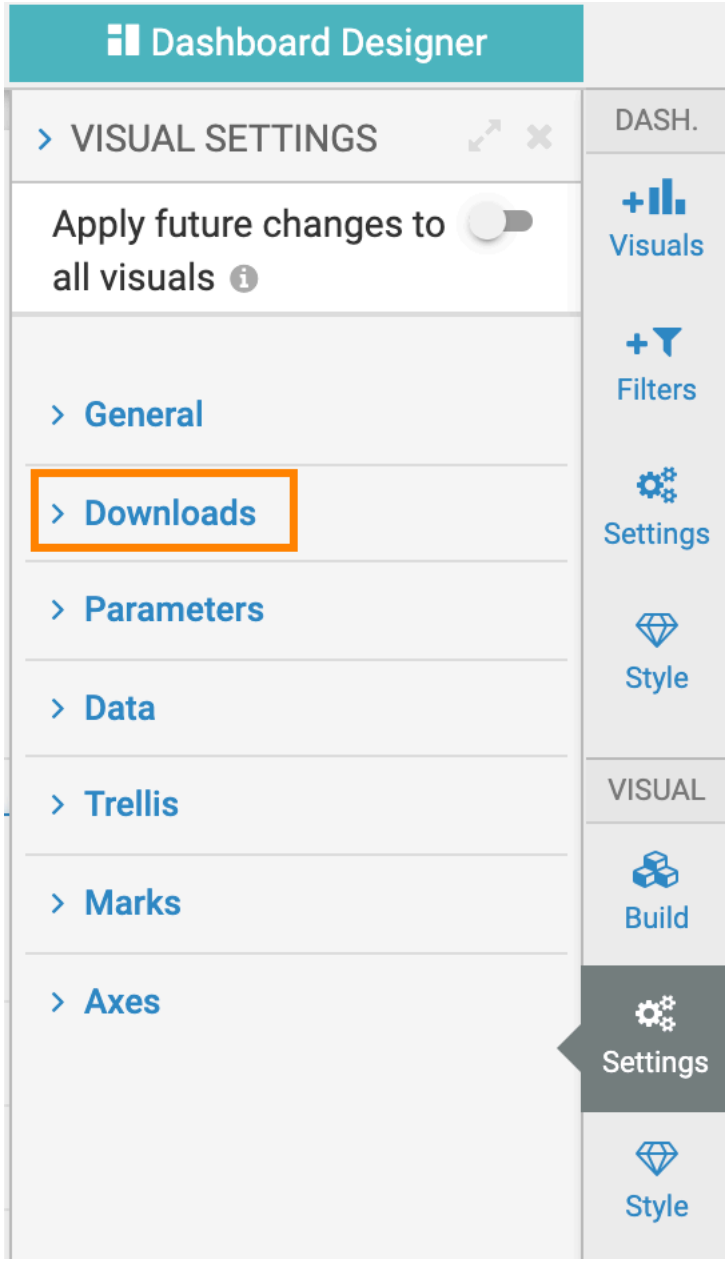

2. Set the language in the Set "Download as CSV" Locale drop-down menu.

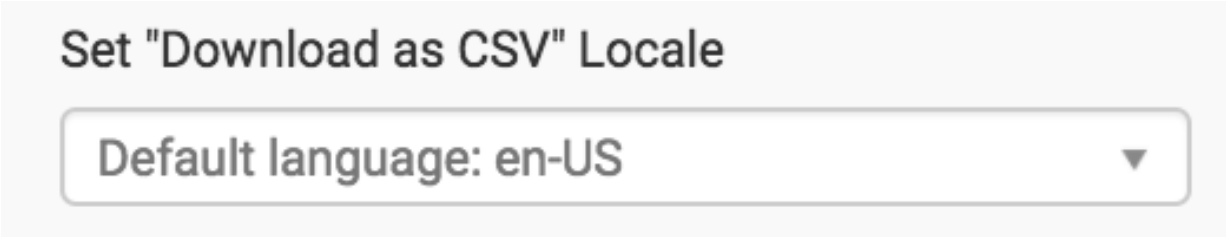

Example

Here, the default locale is en-US, and we use the Set "Download as CSV" Locale menu to select an alternate locale,Germany.

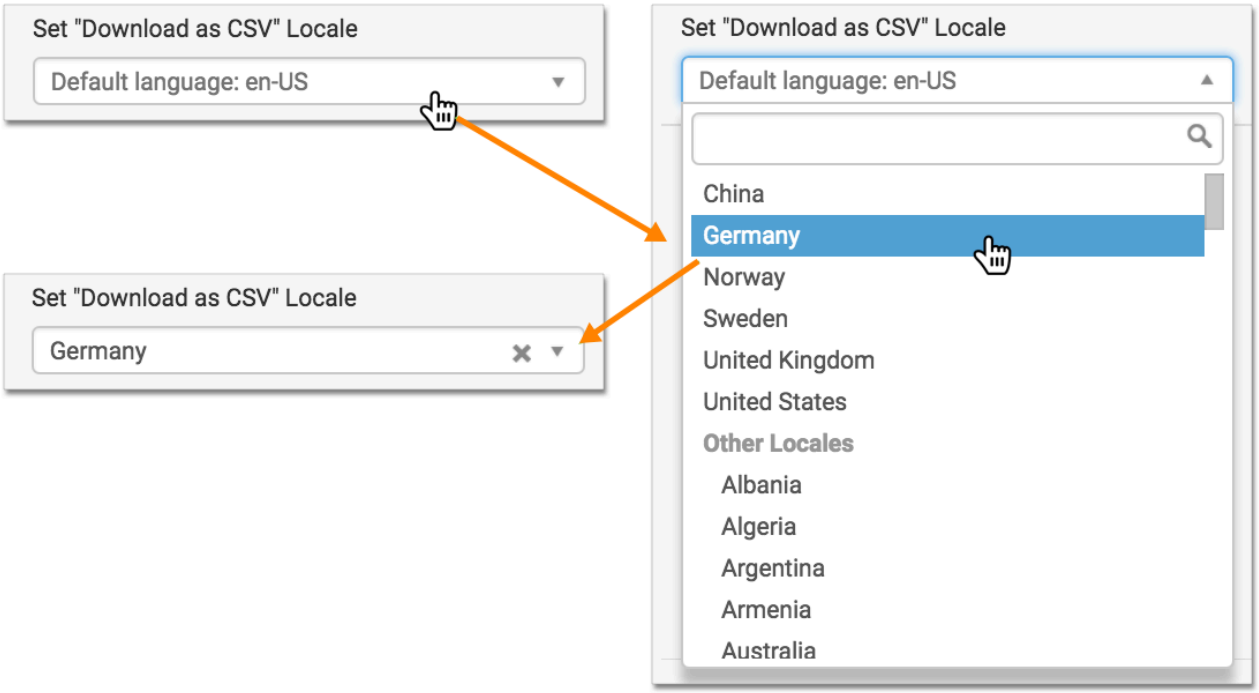

Customizing axes for visuals

About this task

You can mnage several display options from the Axes menu, depending on the chart type in the visual.

Note: The Axes is only available if the visual uses axes.

To get to the options of the menu, follow these steps:

Procedure

1. On the right side of Visual Designer, click the Settings menu.

2. In the Settings menu, click Axes.

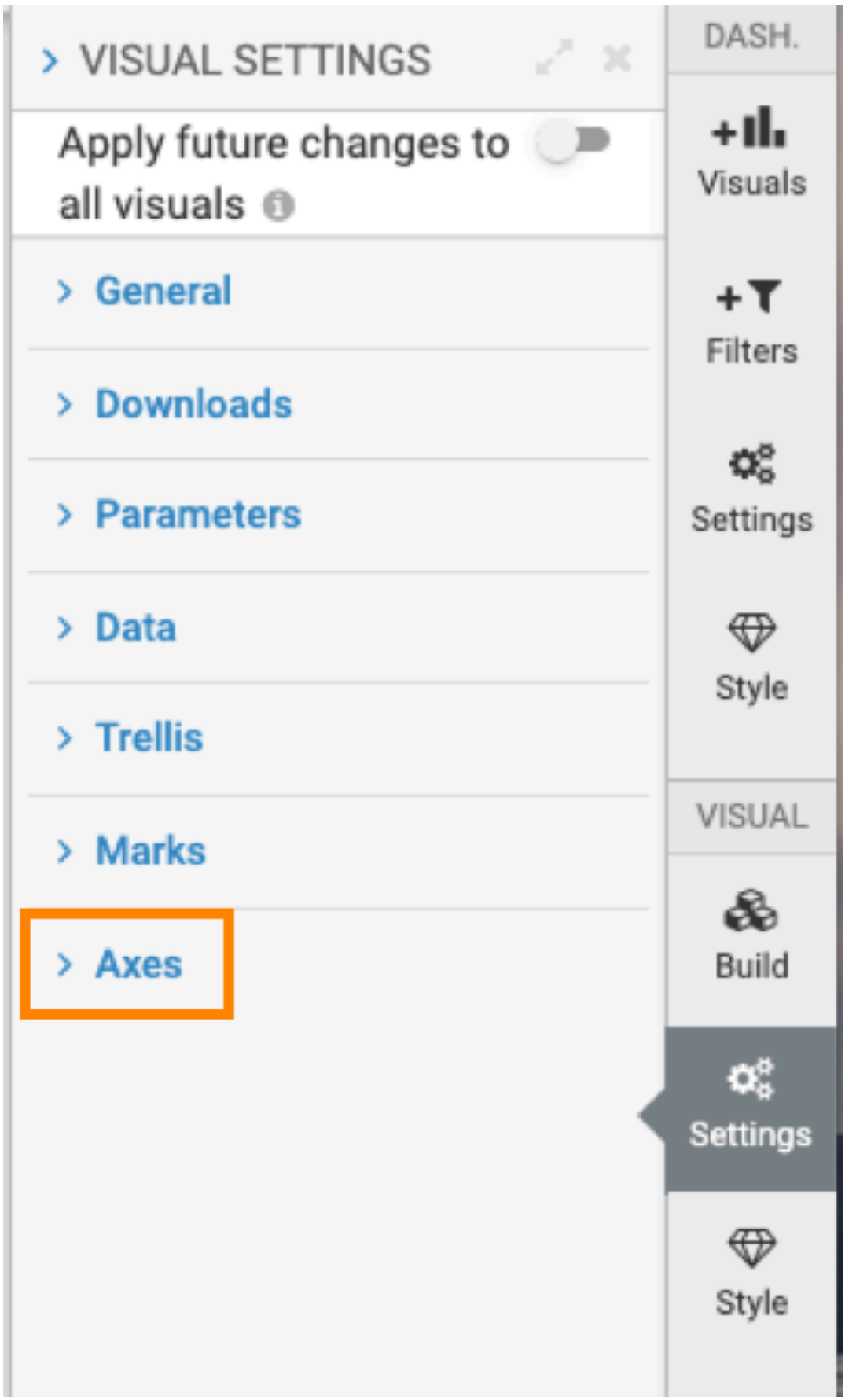
VISUAL SETTINGS
Apply future changes to all visuals
General
Downloads
Parameters
Data
Trellis
Marks
Axes
DASH.
Visuals
Filters
Settings
Style
VISUAL
Build
Settings
Style

3. Select one of the options:

Changing the axis scale

To change the scale for a visual, navigate to the Axes menu, and select one of the options under the heading Axis Scale.

- Linear (default)
- log10
- Percentage of dimension
- Percentage of color

For example, selecting a log10 scale changes the visual by re-calibrating the Y axis, and helps compare the relative values of smaller aggregates, such as for Alaska and Delaware. In the case of Alaska, it clearly shows the relatively lower number of females vs. males in the survey data.

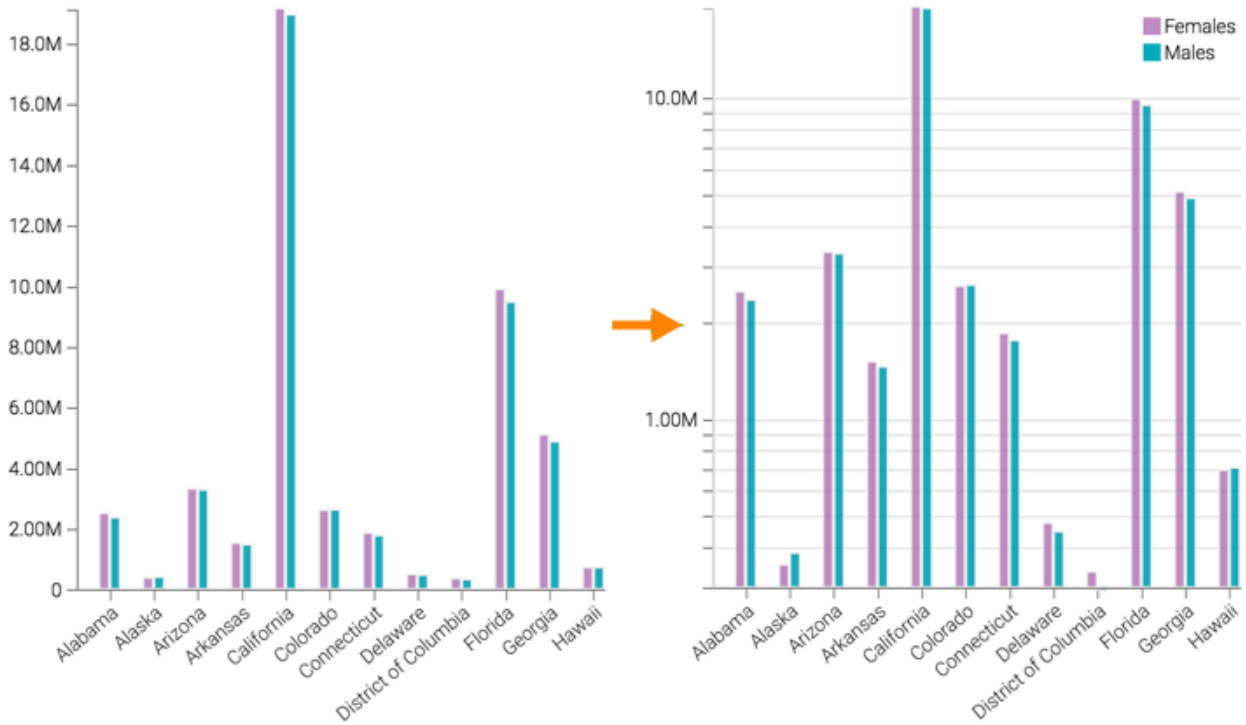

Note that not all options are available on all visual types that support changing axis scale. For example, the Scatter visual has both X Axis Scale and Y Axis Scale options, but only for Linear and log10.

In this example, The Oscars viewing audience is more than twice the size of the next 'runner-up'. This squeezes the plot towards the origin, and makes it very difficult to understand the data. Switching from Linear to log10 axis scale enables us to see the lower-performing categories better.

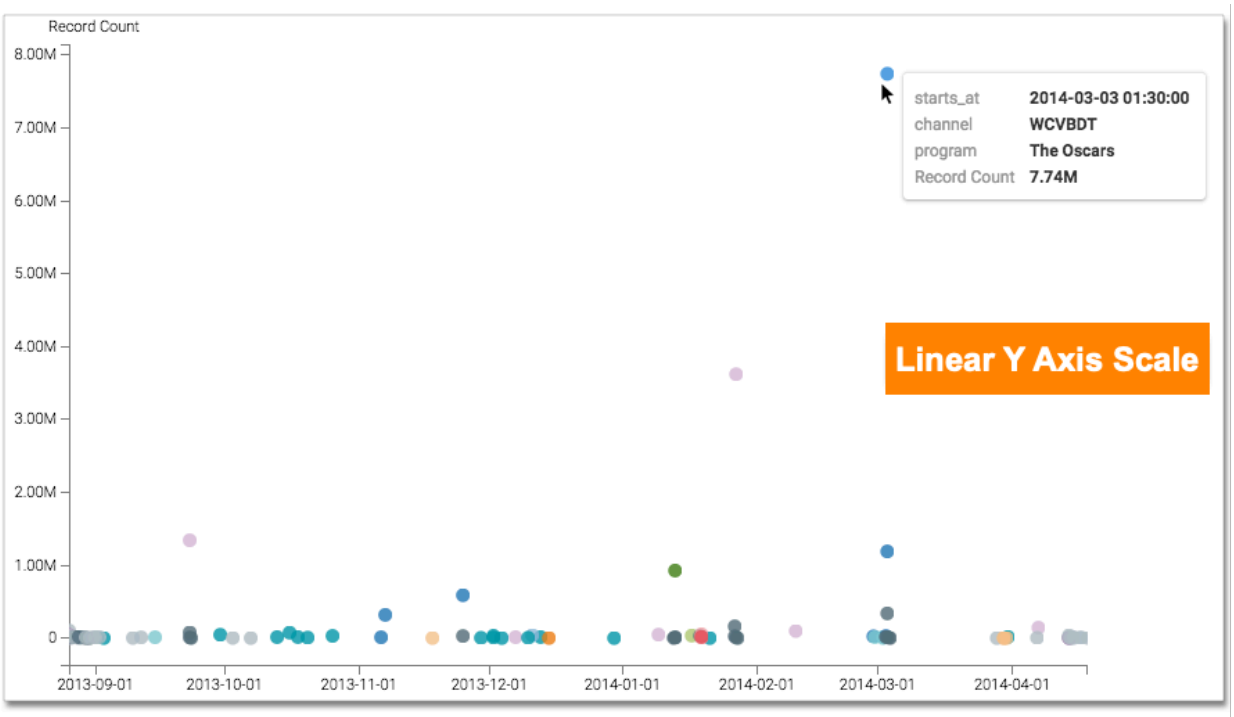

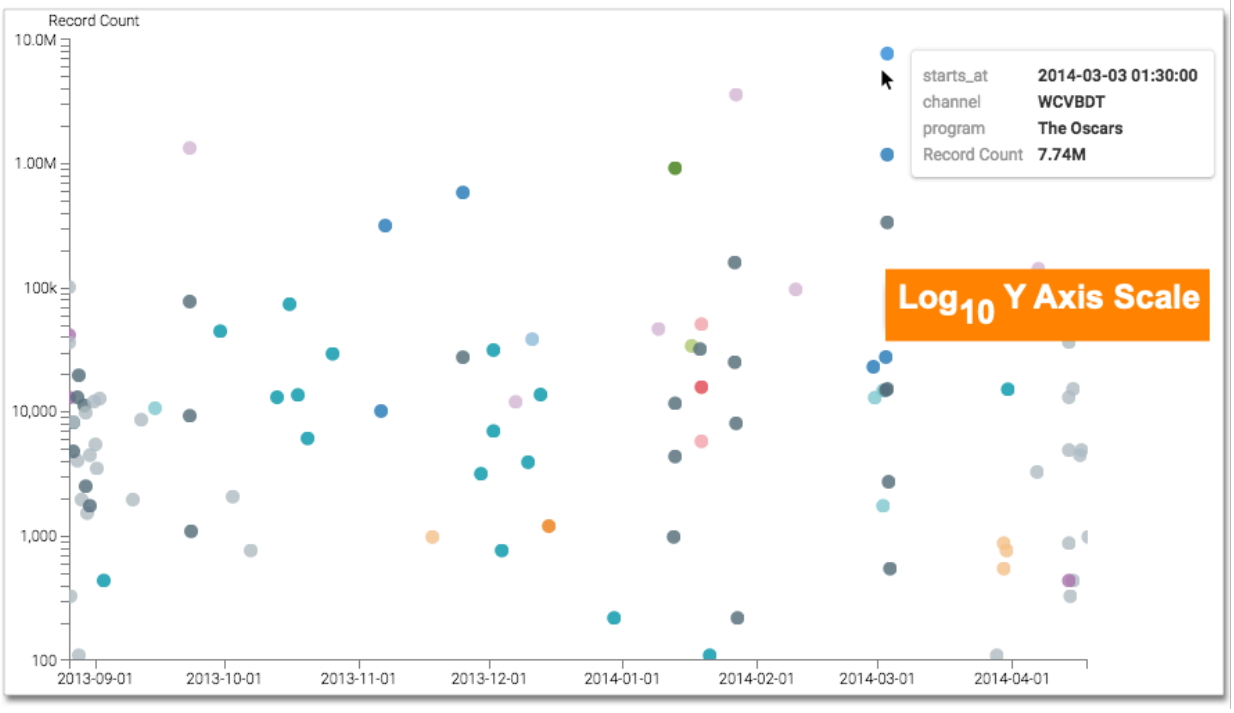

Changing the axes scale for radial graphs

To choose among options for axis display on radial charts, navigate to the Axes menu, and select one of the options under the heading Axis Scale.

- Measure values compare to values in the same measure across all dimensions
- Measure values compare to values across all measures
- Measure values compare to other measure values in the same dimension

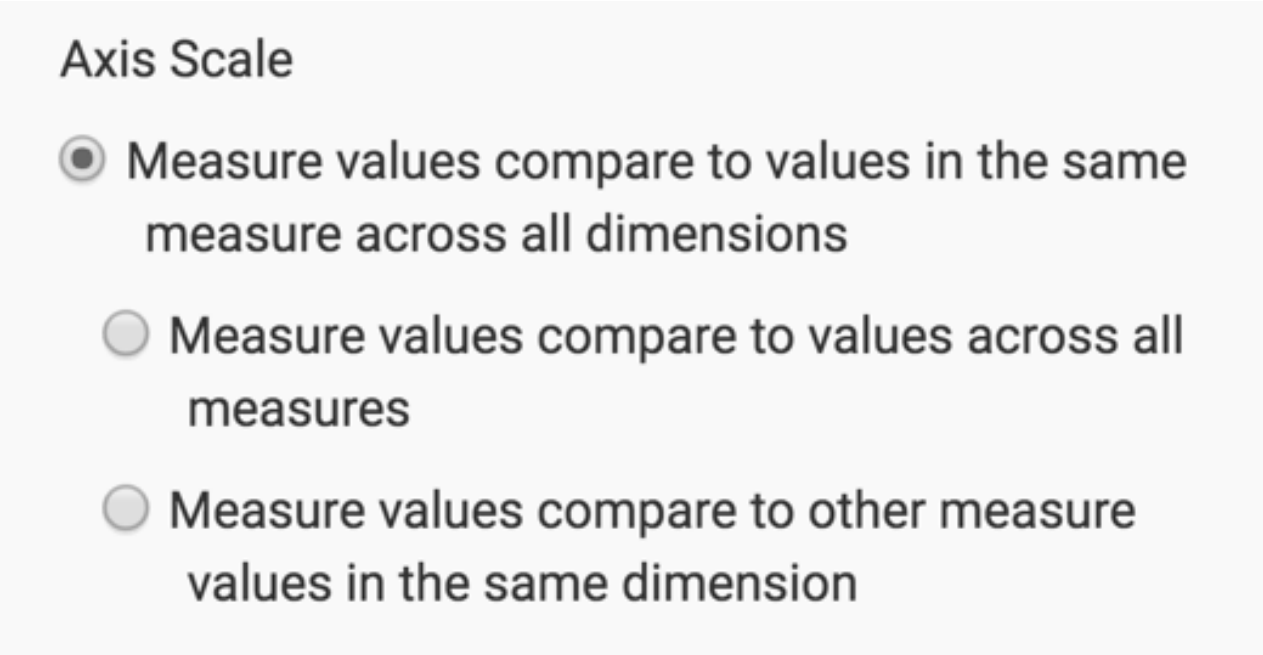

Compare the appearance of the radial charts with each of the options:

Figure 1: Measure values compare to values in the same measure across all dimensions

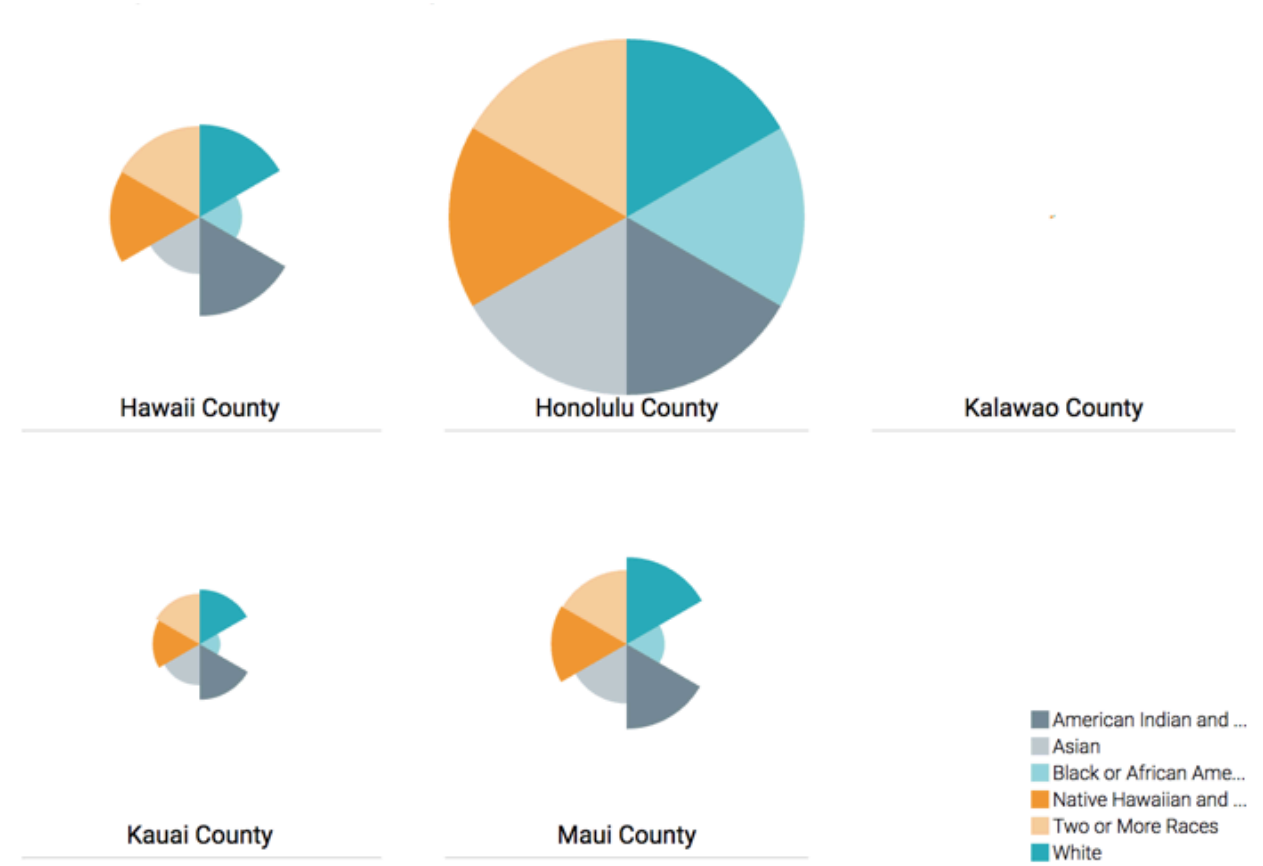

Figure 2: Measure values compare to values across all measures

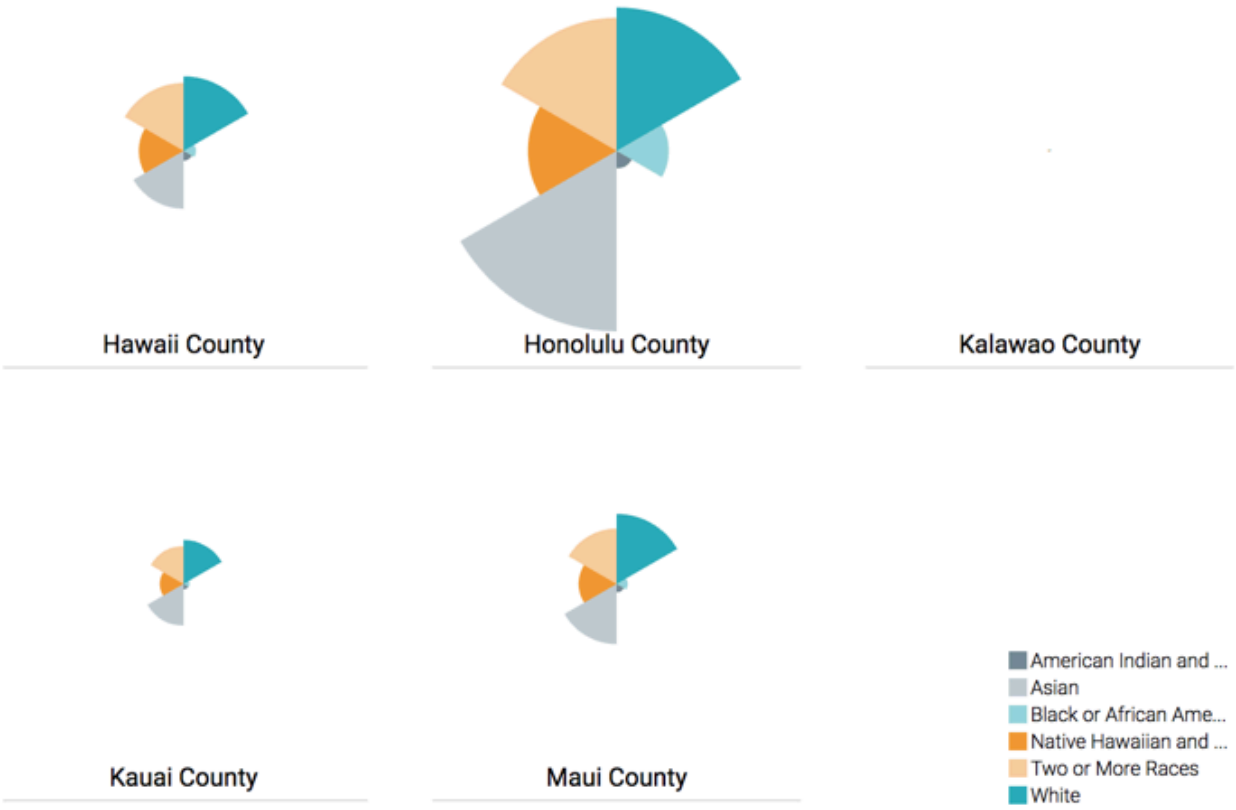

Figure 3: Measure values compare to other measure values in the same dimension

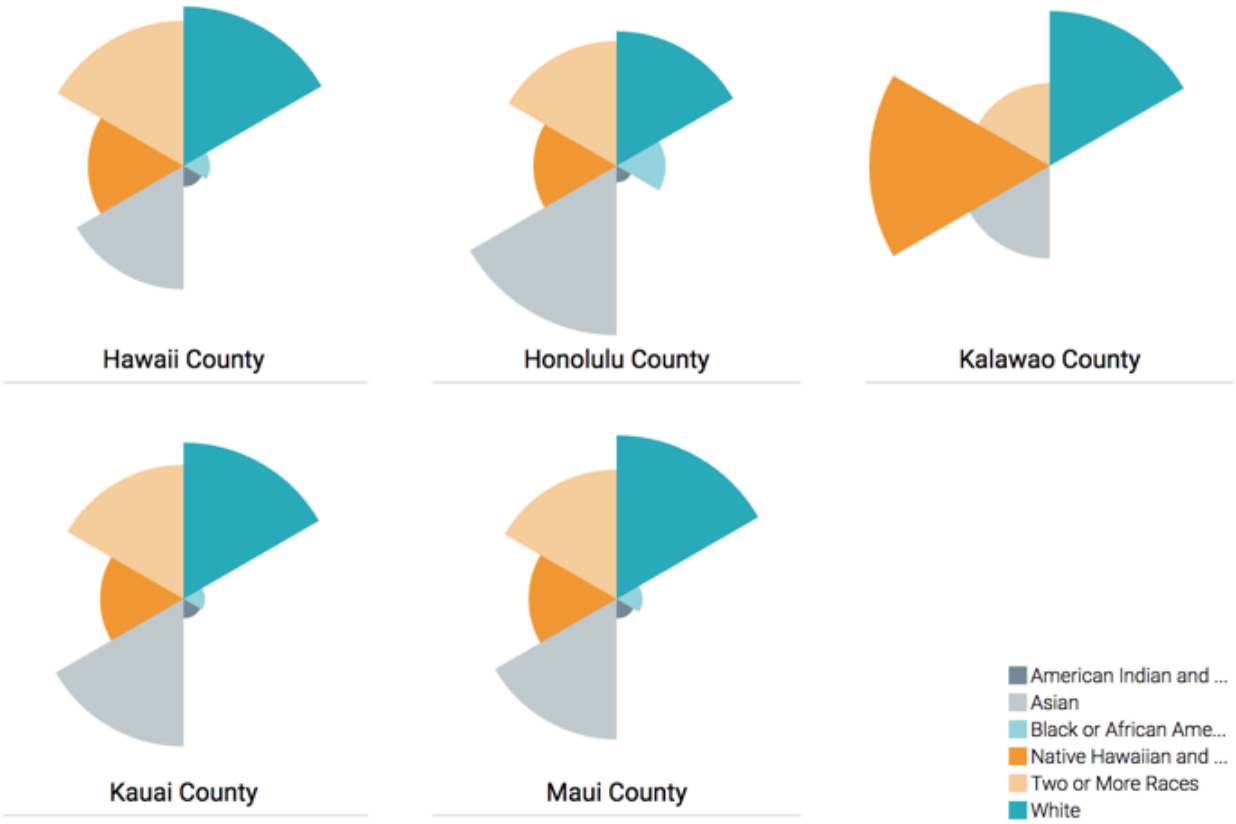

Enabling gridlines

About this task

You can enable horizontal and vertical gridlines for more accurate approximation of the values.

Procedure

1. Click SettingsAxes in the Dashboard Designer.
2. Select the desired gridline axes to show.

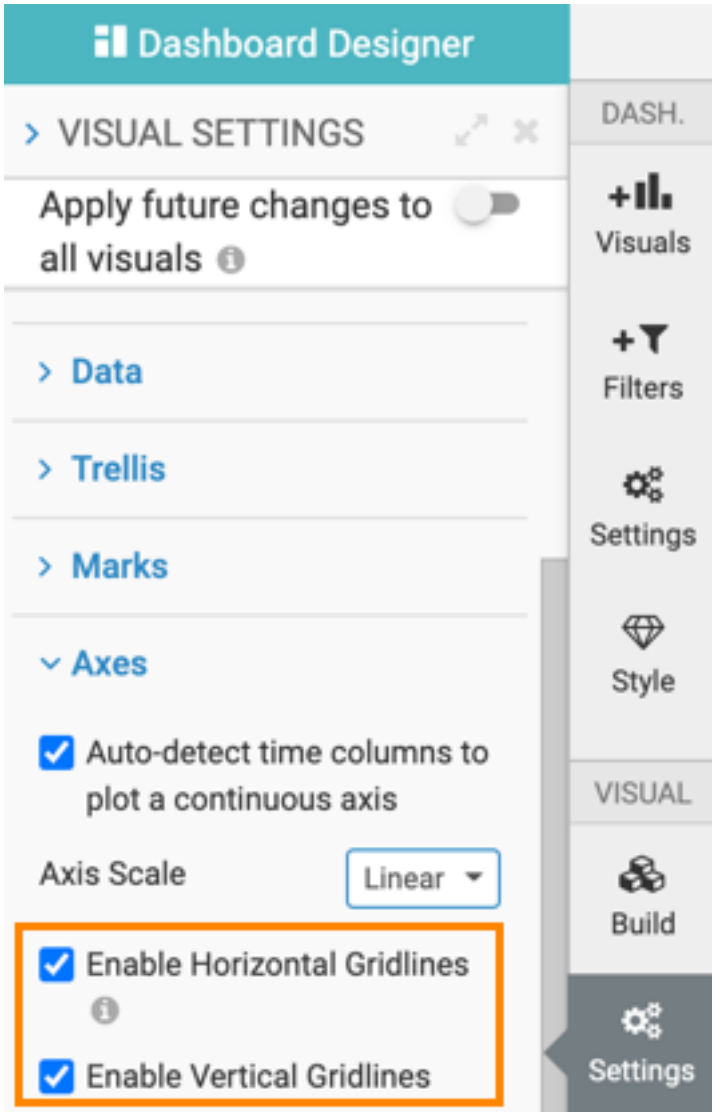

Enabling continuous axis for date/time values

When using date/time measurements on the horizontal axis, values tend to be lengthy. This often distracts from the visual plot.

To use regular intervals, navigate to the Axes menu, and select the Auto-detect time columns to plot a continuous axis option.

You can see how enabling this feature results in a cleaner and clearer visual. Also, the plot produces conveniently rounded-up axis labels.

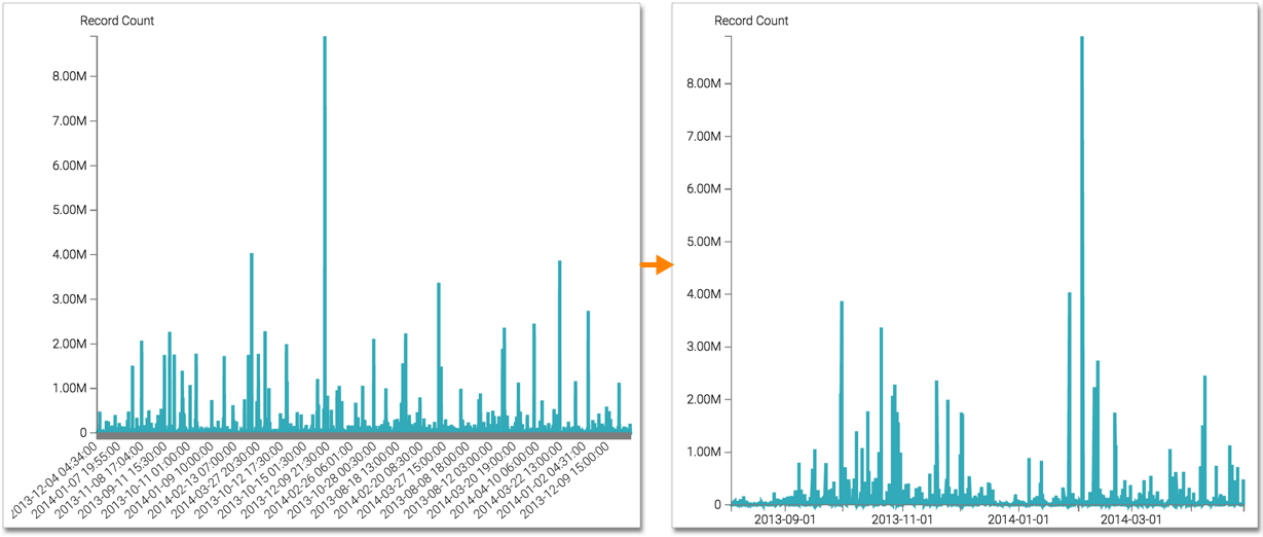

Changing the padding on the X axis

To change the extra space to the sides of the plot (X Axis padding), adjust the value on the selector X Axis Padding (%). The default is 5%, the range is from 0% to 1,000%.

Changing the padding on the Y axis

To change the extra space at the top and at the bottom of the plot (Y Axis padding), adjust the value on the selector Y Axis Padding (% of range). The default is 5%, the range is from 0% to 1,000%.

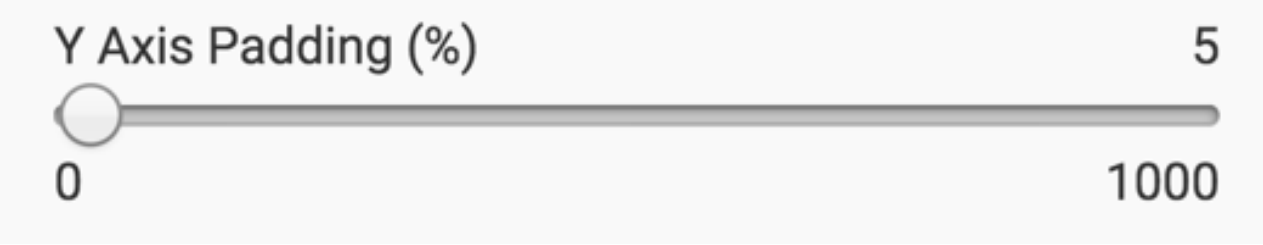

For example, contrast the Y Axis padding of the same graph at 5% and at 15%.

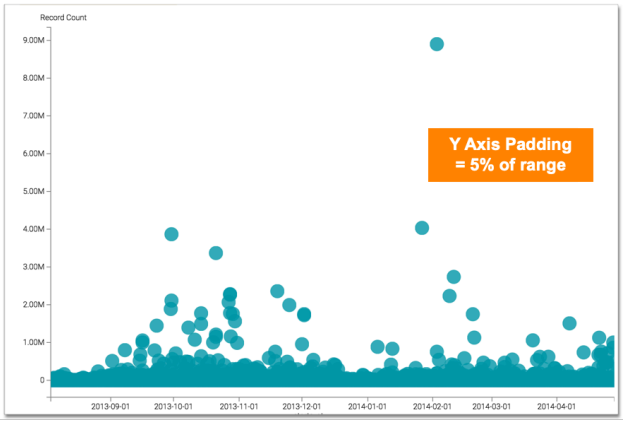

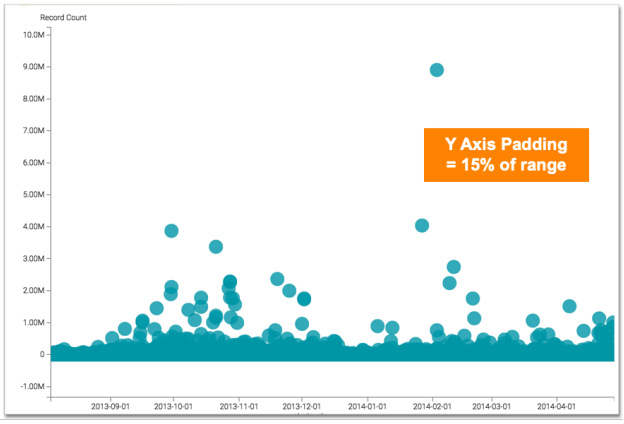

Starting aggregate scale at zero

By default, the aggregate scale starts at zero. However, you can change this by de-selecting this option.

To disable the zero start for aggregate scale, navigate to the Axes menu, and de-select the option Positive aggregate scales start at zero.

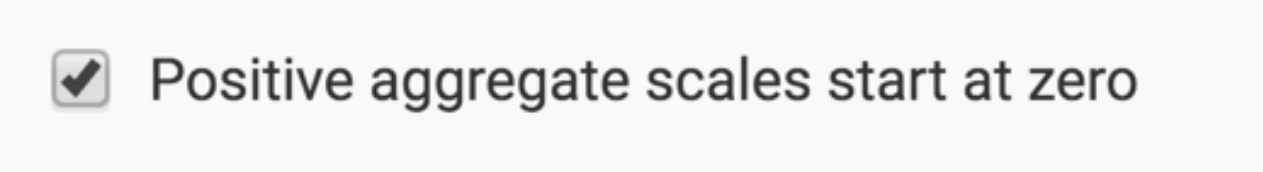

Notice that the aggregates with smaller values barely show over the X Axis and are very difficult to read: Alaska, Delaware, and especially District of Columbia.

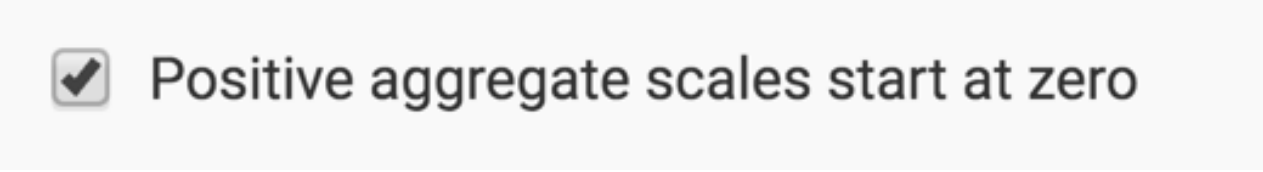

Changing the min/max of the aggregate scale

You can configure independent aggregate scales for visuals with primary and secondary axes.

The line, combined bar/line, and grouped bar visuals support primary and secondary axes with independent minimum and maximum aggregate scales.

By default, the aggregate scale starts at zero, and rises to a number just higher than the maximum aggregate value. This can be adjusted, which is useful in dashboards that contain side-by-side visuals for different segments, time periods, and so on.

To change the minimum and maximum aggregate scale values, navigate to the Axes menu, and make adjustments to the four selectors:

- Min Agg Scale - for the Primary Axis
- Max Agg Scale - for the Primary Axis
- Sec. Min Agg Scale - for the Secondary Axis
- Sec. Max Agg Scale - for the Secondary Axis

For example, raising the Minimum aggregate scale value to 250,000 has the effect of shrinking the values for the smaller aggregates.

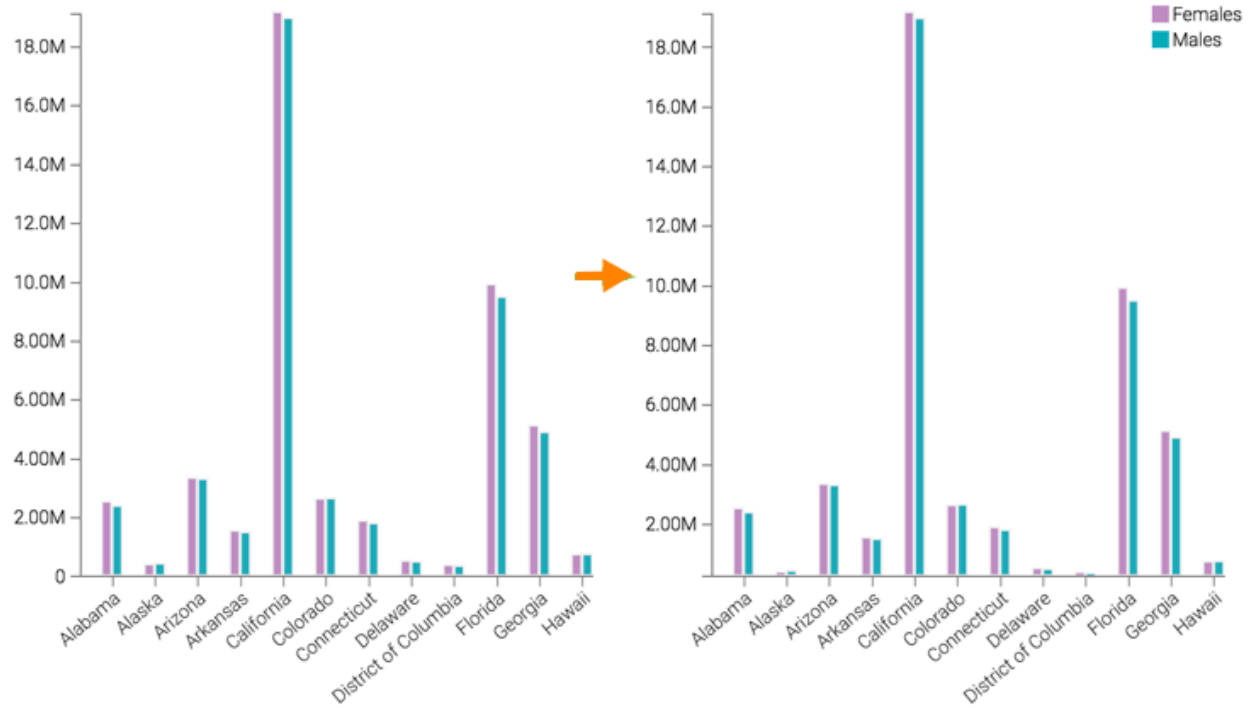

Meanwhile, raising the Maximum aggregate scale value to 26 Million adds blank space over the top aggregate, and scales the whole graph down.

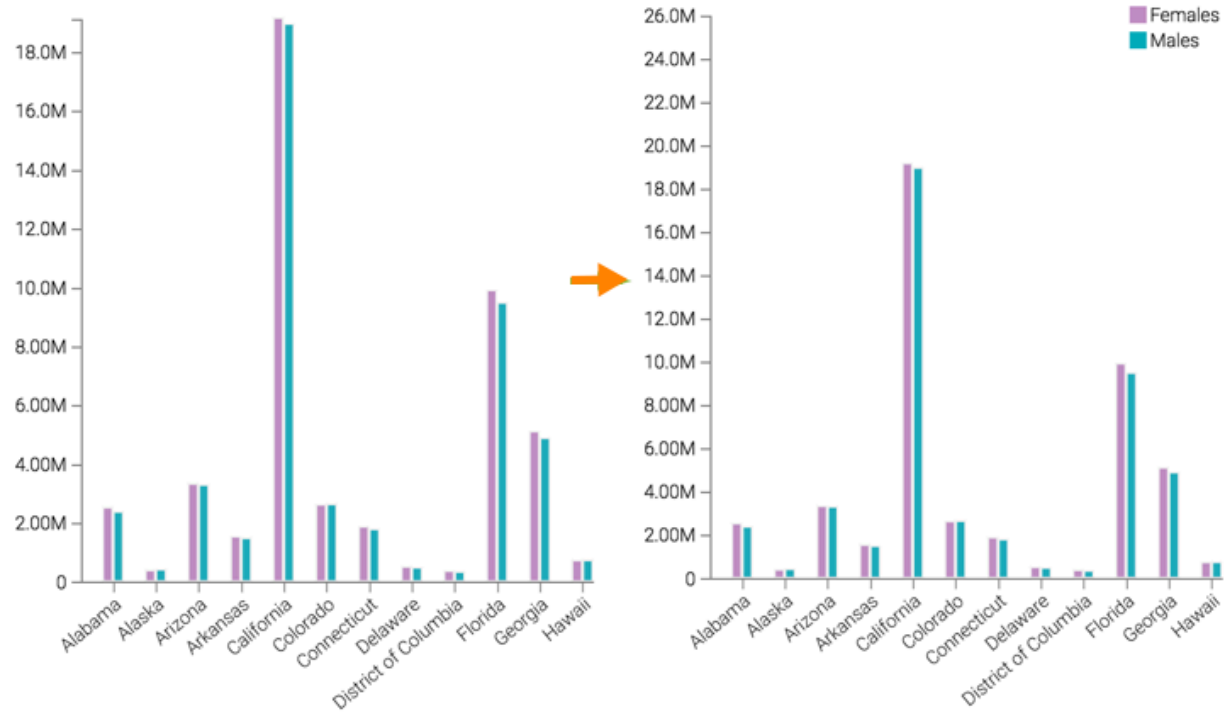

Note that the minimum cannot be larger than the smallest aggregate, and the maximum cannot be smaller than the largest aggregate.

Showing or hiding axis labels

Cloudera Data Visualization enables you to show or hide axis lables. By default, visuals with vertical and horizontal axes have labels on these axes.

To remove the labels on visuals that use X Axis and Y Axis, navigate to the Axes menu, and de-select: Include X Axis label, or Include Y Axis label, or both.

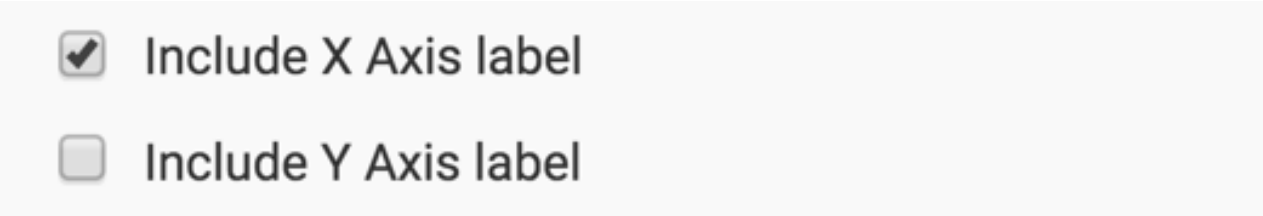

To remove the labels on visuals that use Dimensions and Measures, navigate to the Axes menu, and de-select: Include dimension axis label, or Include measure axis label, or both.

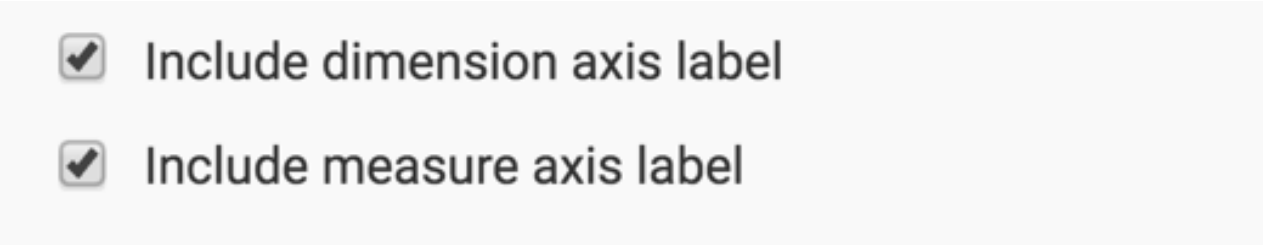

Changing the size of dimension text

Occasionally, the textual description for a dimension on the graph is very long. The options are either to increase the size of the text length, or to decrease it and leave it up to the user to understand the context.

To change the length of the text for dimensions, navigate to the Axes menu, and adjust the value on the selector Max tick length.

In this example, the text length has been decreased from 25 to 15. The impact is visible on the columns labeled District of Columbia:

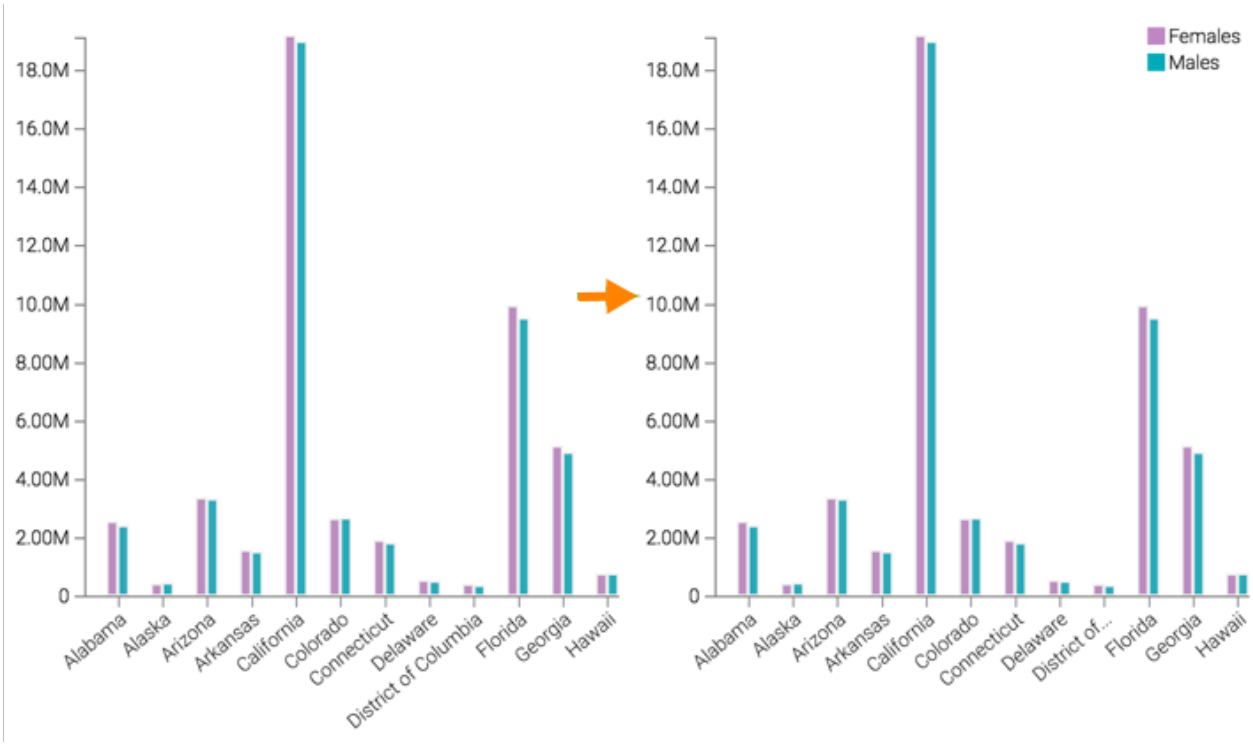

Showing or hiding an axis

Cloudera Data Visualization enables you to show or hide an axis. By default, all visuals display horizontal and vertical axes when they have directionality.

A visual may have a Dimension Axis and Measure Axis, or an X Axis and Y Axis.

To hide an axis, navigate to the Axes menu, and de-select the appropriate options.

For visuals that use Dimensions and Measures, the options are Show dimension axis, or Show measure axis, or both.

For visuals that use X axis and Y axis, the options are Show X Axis, or Show Y Axis, or both.

For example, hiding the Y axis label changes a visual in the following way:

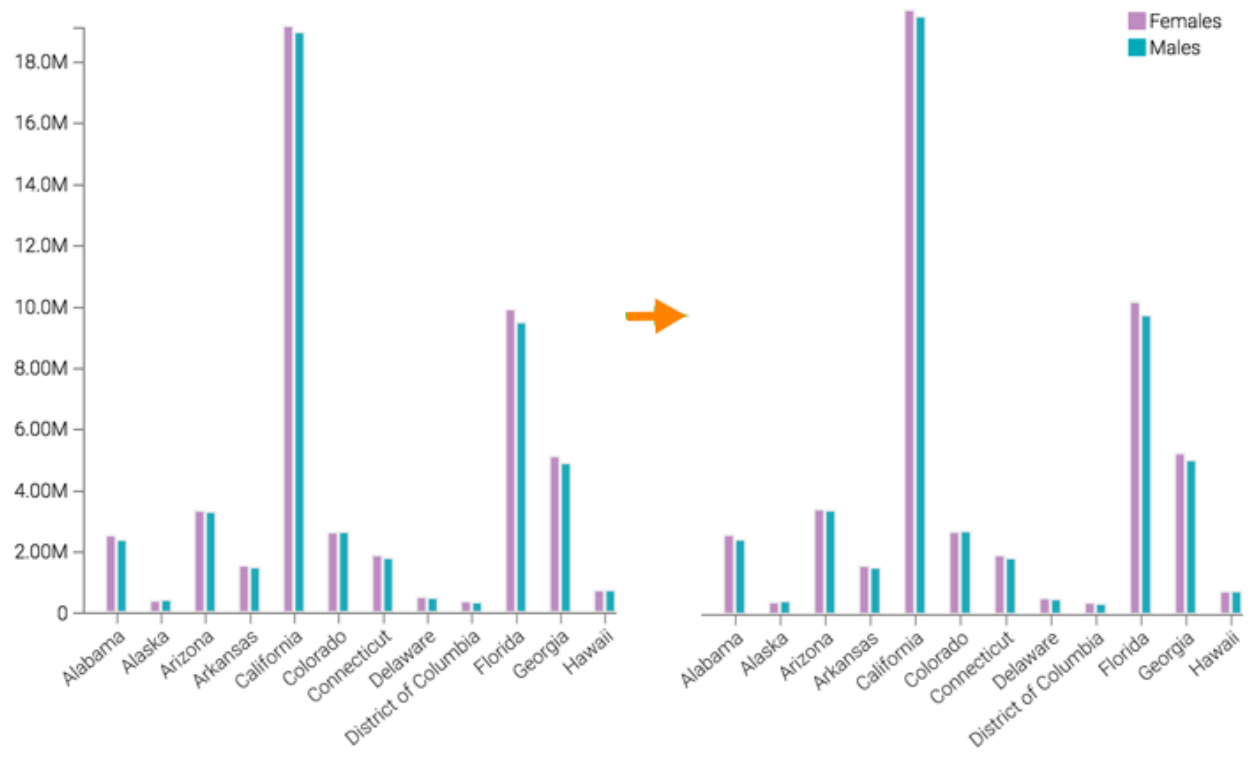

Reversing the axis scale

By default, all visuals display horizontal and vertical axes with origin in the bottom left of the visual. However, it is possible to reverse both the X and Y Axis, separately and together.

To reverse an axis, navigate to the Axes menu, and select the appropriate options: Reverse x scale, Reverse y scale, or both.

The following image demonstrates the possible scale direction options for a scatter plot: original, reversed x axis, reversed y axis, and both x and y axes reversed.

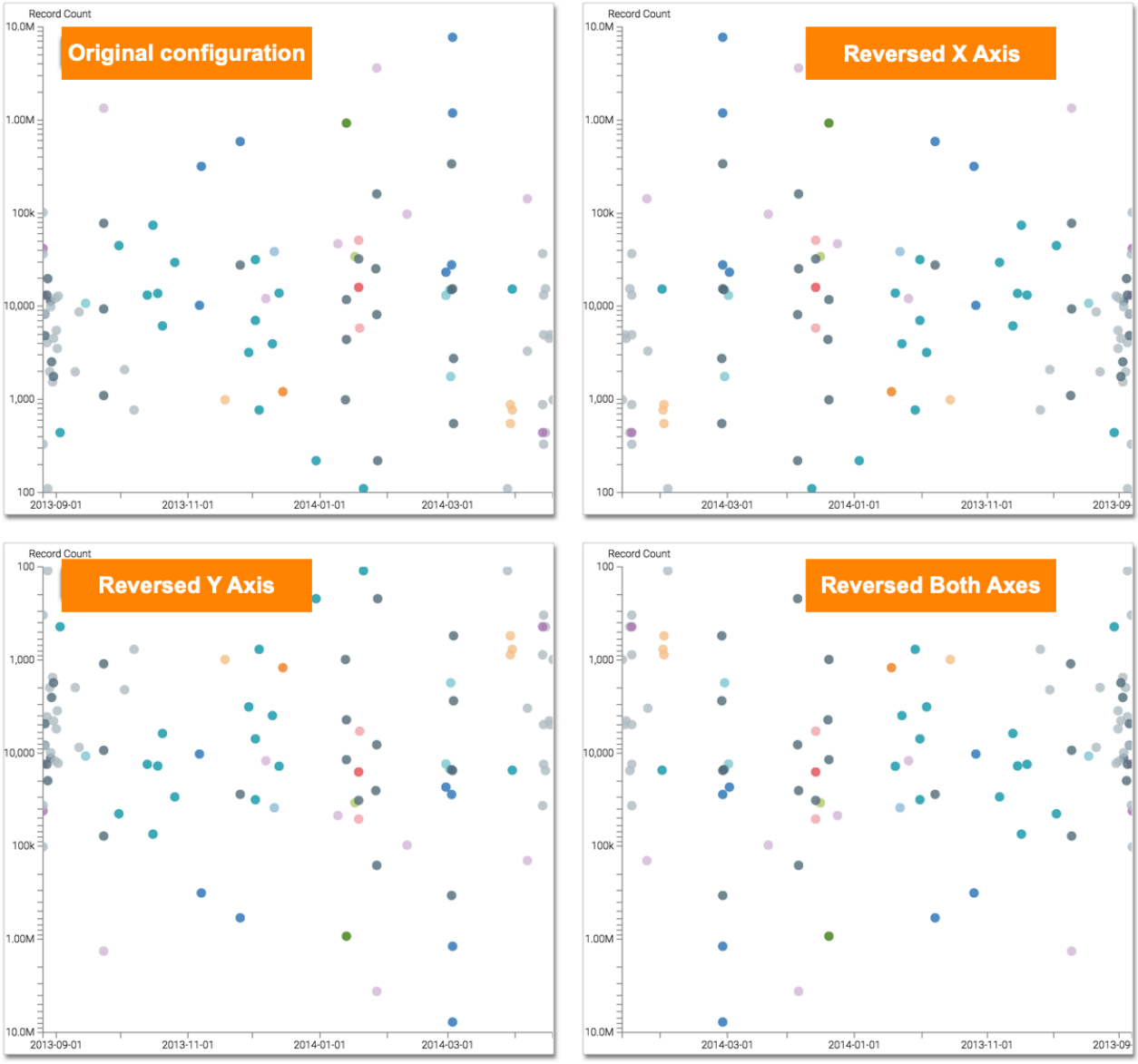

Dynamically sizing a graph

To turn on or off the option for dynamic graph sizing, navigate to the Axes menu, and change the selection for the Dynamically size graph to window size option.

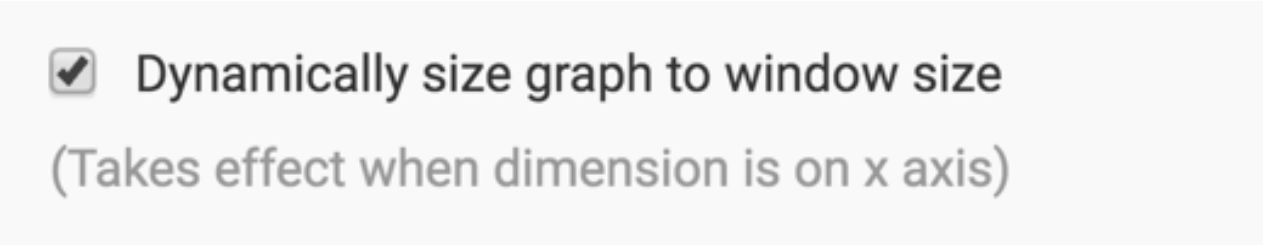

Setting maximum row label length

availability:

- This setting is for the correlation heatmap visual type.

To change the maximum height of the row label, navigate to the Axes menu, and change the value of the Row label length option. The default is 400 px, the range is from 1 px to 1,000 px.

Setting column label length

availability:

- This setting is for the correlation heatmap visual type.

To change the maximum length of the column label, navigate to the Axes menu, and change the value of the Column Label Length option. The default is 400 px, the range is from 1 px to 1,000 px.

Enabling a continuous X axis

You can customize a *scatter*visual according to your data type. To look at trends and the rate of change, use a continuous axis, that shows data at equal intervals.

Compare the continuous axis to a categorical axis that only plots values in the dataset.

Note:

- Available only on *scatter* visuals.
- Available only on numeric and non-aggregate fields.

To use regular intervals on the X axis, navigate to the Axes menu, and select the Make X axis continuous option.

This option is on by default.

The image on the left shows a categorical X axis with each label representing a value in the dataset, from 125 to 290 in irregular intervals. Image on the right shows a continuous X axis where labels are auto-generated to use equal intervals, from 50 to 350. Notice how enabling this feature gives a clearer visual representation of where the sodium levels fall within a range of values.

Example of a continuous X Axis

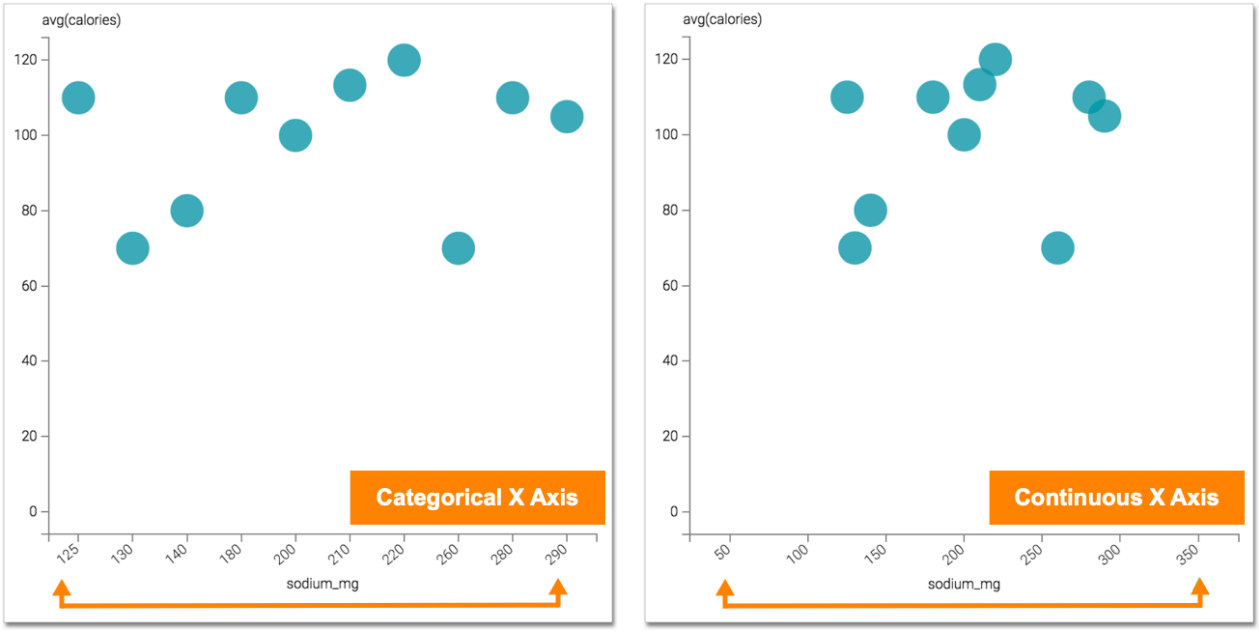

Changing scale of a continuous X axis

You can change the minimum and maximum scale of a continuous X-axis in a Scatter visual.

A continuous axis scale has an infinite number of possible values, in equal intervals. Compare it with a categorical axis scale, which has a finite number of evenly spaced values, limited to the values in the dataset.

To enable continuous X-axis, see *Enabling a continuous X axis*.

To change the minimum and maximum scale of a continuous X-axis, navigate to the Axes menu, and make adjustments to the two selectors for the Min X Scale and Max X Scaleoptions.

The image on the left shows the minimum and maximum scale of the X axis from 50 to 350. Notice that in the right image the scale changes when we changed the values from 0 to 1000. Increasing the scale also gives a clearer visual representation of where the sodium levels fall within a range of values.

Figure 4: Minimum and Maximum Values of a Continuous X Axis

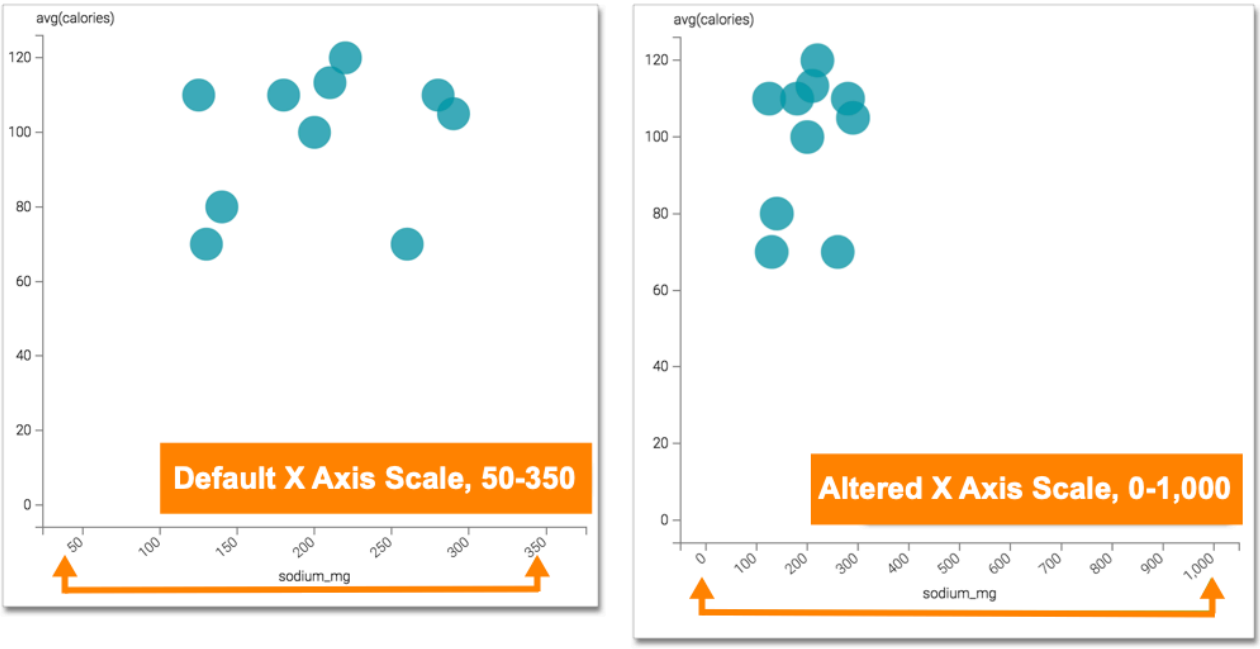

Related Information

Enabling a continuous X axis

Enabling a continuous Y axis

You can customize a Scatter visual according to your data type. To look at trends and the rate of change, use the continuous axis that shows data at equal intervals. Compare the continuous axis to a categorical axis, which only plots values in the dataset.

Note:

- Available only on Scatter visuals.
- Available only on numeric and non-aggregate fields.

To use regular intervals on the Y axis, navigate to the Axes menu, and select the Make Y axis continuous option. This option is on by default.

The image on the left shows a categorical Y axis with each label representing a data point, starting from 10 to 23. Image on the right shows a continuous Y axis where labels are auto-generated in equal intervals, from 0 to 24. Notice how enabling this feature gives a clearer visual representation of where the carbohydrate levels fall within a range of values.

Figure 5: Contrast between a Continuous Y Axis

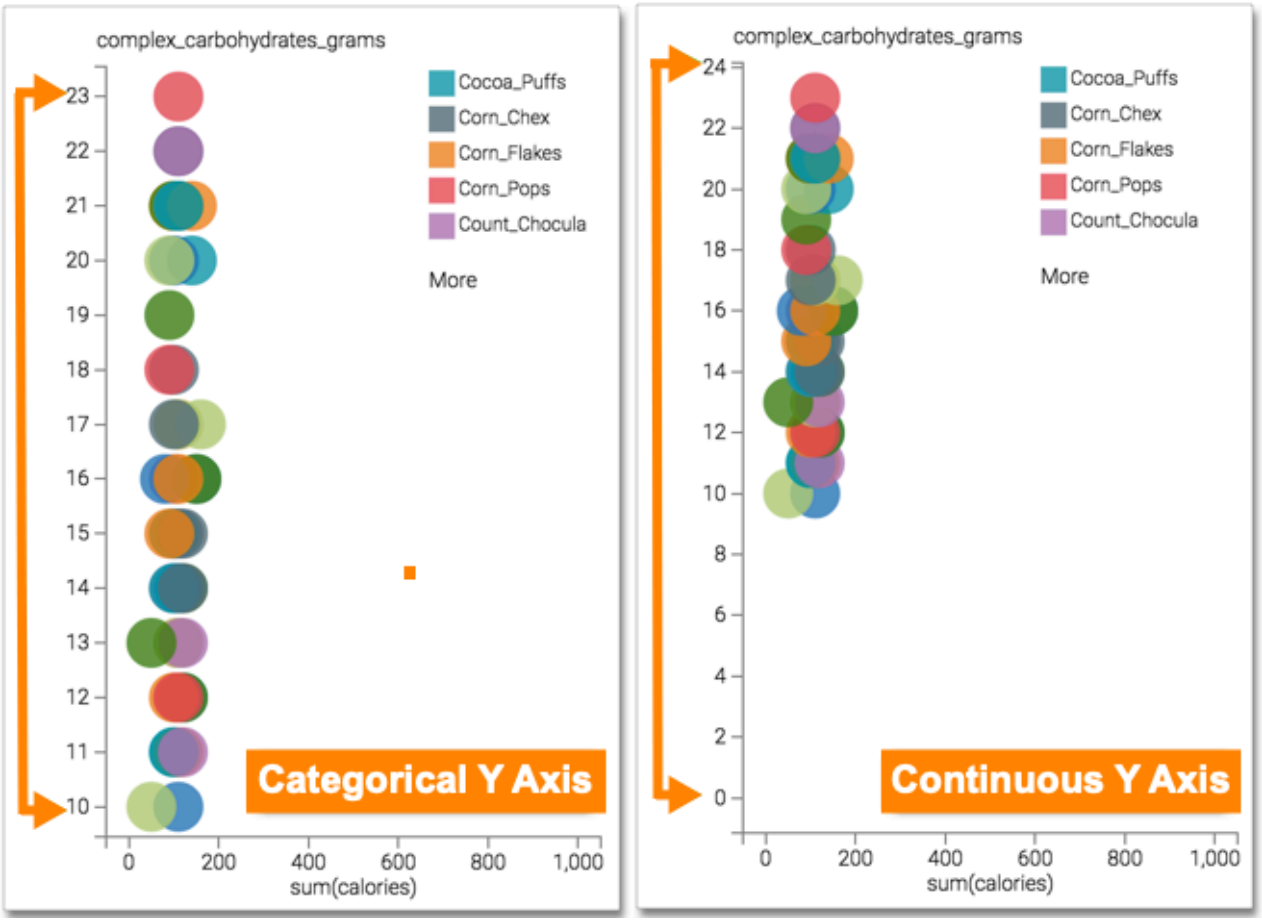

Changing scale of a continuous Y axis

You can change the minimum and maximum scale of a continuous Y axis in a Scatter visual.

A continuous axis scale has an infinite number of possible values in equal intervals. Compare it with a categorical axis scale, which has a finite number of evenly spaced values, limited to the values in the dataset.

To enable continuous Y axis, see *Enabling a continuous Y axis*.

To change the minimum and maximum scale of a continuous Y axis of a Scatter visual, navigate to the Axes menu, and make adjustments to the two selectors, Min Y Scale and Max Y Scale.

The image on the left shows the minimum and maximum scale of the Y axis from 0 to 24. Notice in the right image, how the scale changes when we changed the values from 10 to 30. Increasing the scale gives a clearer visual representation of where the carbohydrate levels fall within a range of values.

Figure 6: Minimum and Maximum Values of Continuous Y axis

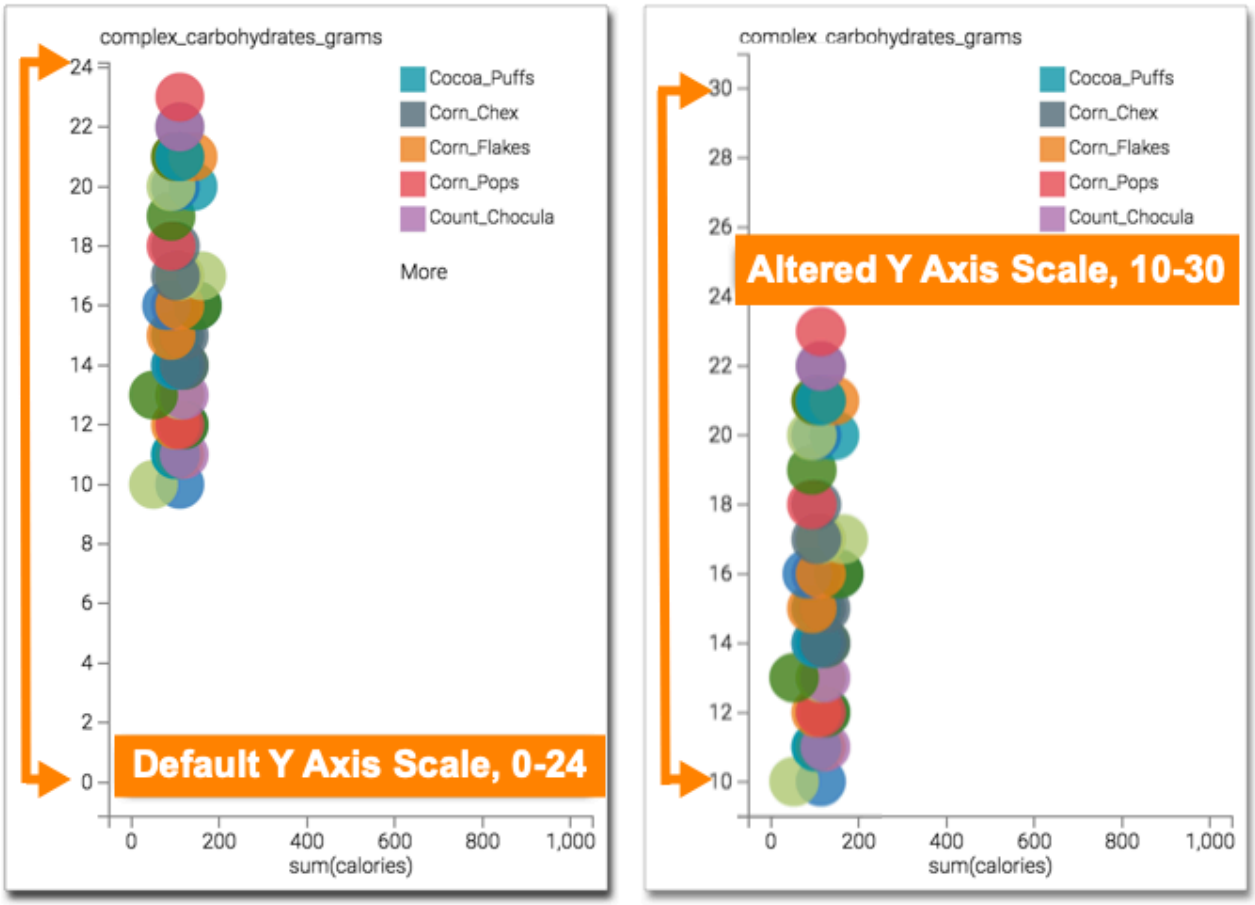

Related Information

Enabling a continuous Y axis

Enabling a continuous axis on dimensions

Enabling a continuous axis on dimensions

You can customize your visuals according to their data type. To look at trends and the rate of change, use a continuous axis on numerical dimension, which shows data at equal intervals. Compare visuals that use a continues dimensional axis to results with a categorical axis, which only plots dataset values.

availability:

- Available only for numerical dimensions on Bars, Grouped bars, Line, Area, and Combo visuals.

To use regular intervals, navigate to the Axes menu, and select the Make Dimension Axis Continuous option.

The image on the left shows a categorical X axis where each label represents a value in the dataset, from 50 to 160. The image on the right shows a continuous X axis where labels are auto-generated, in equal intervals starting from 0 to 160. Notice how enabling this feature gives a clearer visual representation of where the calorie levels fall within a range of values.

Figure 7: Continuous Dimension Axis (Horizontal)

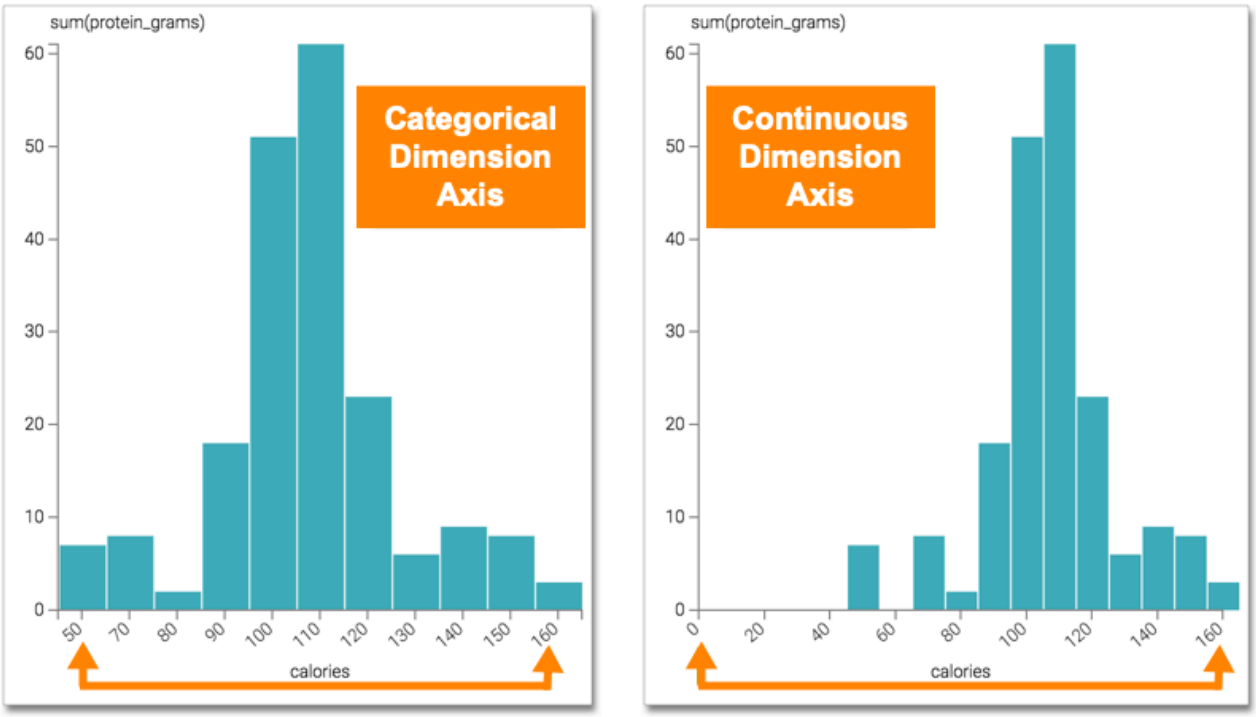

Changing the scale of a continuous axis on dimensions

You can change the minimum and maximum scale of a continuous axis enabled on a dimension.

A continuous axis scale has an infinite number of possible values, in equal intervals. Compare it with a categorical axis scale, which has a finite number of evenly spaced values, limited to the values in the dataset.

To enable continuous dimension axis, see *Enabling a continuous axis on dimensions*.

availability:

- Available only for numerical dimensions on Bars, Grouped bars, Line, Area, and Combo visuals.

To change the minimum and maximum scale of a continuous axis on dimensions, navigate to the Axes menu, and make adjustments to the two selectors: Min Dim Scale and Max Dim Scale options.

The image on the left shows the minimum and maximum scale of the X axis from 50 to 160. You can see that that in the right image, the scale changes when you change the values from 0 to 200. Increasing the scale also gives a clearer visual representation of where the calorie levels fall within a range of values.

Figure 8: Example: Minimum and Maximum Values of Continuous Axis on Dimensions

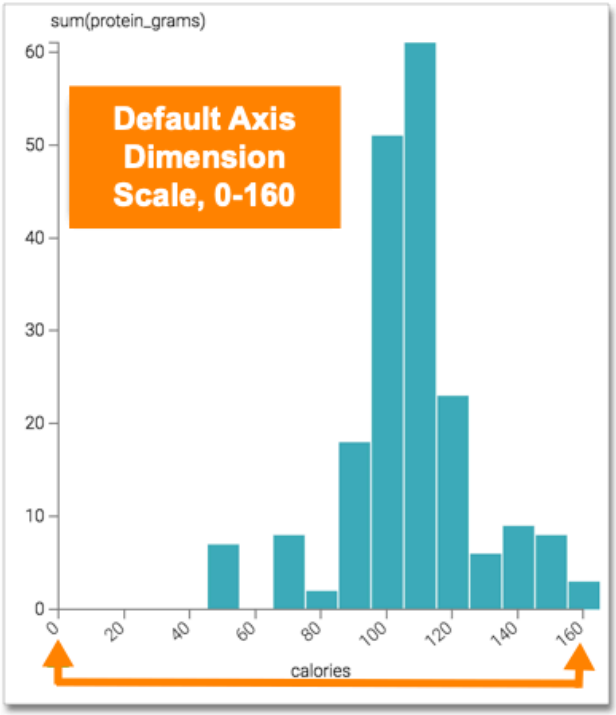

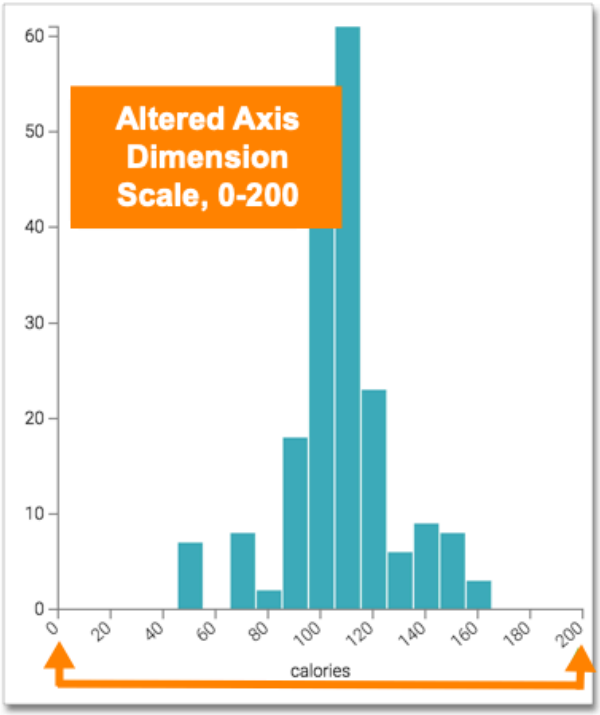

Customizing data for visuals

About this task

Follow these steps to get to the data options:

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Data.

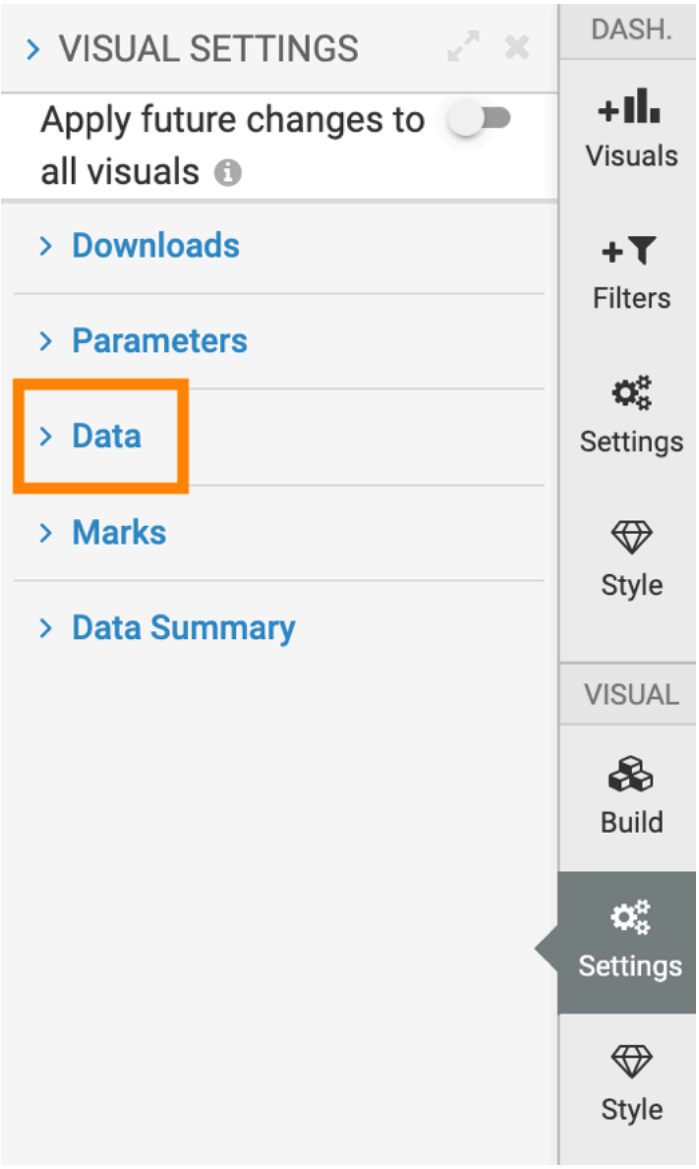

3. Under Data, select one of the following options:

Ordering NULL rows

Note: This feature is only available on Impala or PostgreSQL connections.

To change the order in which NULL values appear in the visual, navigate to the Data menu, and select one of styles under Show data with NULL values: Default, First, or Last.

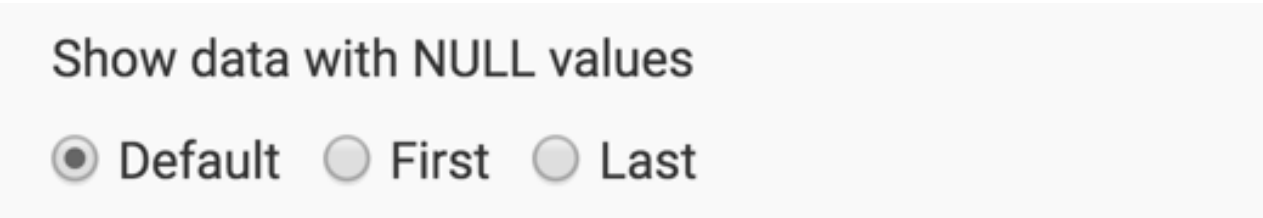

- Default NULL ordering shows NULL data 'as is', without sorting the results.

State	Year	previous percent(Population)
AK	1900	1.07277573467
AK	1920	1.30001181056
AK	1970	1.0424191733
AK	1950	1.04646509017
AK	1790	NaN
AK	1910	1.00129136651
AK	1980	1.00069975347
AK	1800	NaN
AK	1990	1.00210242872
AK	1960	1.00126615342
AK	1930	1.00307973467
AK	2000	1.0019033434

< 1 2 3 ... 9 >

State	Year	previous percent(Population)
AR	2000	1.01026056515
AR	1960	1.00612425714
AR	1970	1.00723274179
AR	1950	1.00221067664
AR	1900	1.00633928284
AR	1920	1.00046248874
AZ	1790	null
AZ	1990	1.00133977211
AZ	1930	1.00276260265
AZ	1940	1.01109598973
AZ	2010	1.00317823975
AZ	1910	1.01004339617

< 1 ... 4 ... 9 >

State	Year	previous percent(Population)
CT	1980	1.00028937784
CT	1900	1.00689983873
DC	1990	1.00414131771
DC	1800	1.07157894737

< 1 ... 7 8 9 >

- First NULL ordering shows NULL data first, after the sort of the specified column.

State	Year	previous percent(Population)
KS	1790	null
KS	1800	NaN
AL	1790	NaN
ID	1790	NaN
IA	1800	NaN
UT	1800	NaN
SD	1790	NaN
NM	1790	NaN
NM	1800	NaN
OK	1800	NaN
AK	1790	NaN
IA	1790	NaN

< 1 2 3 ... 9 >

State	Year	previous percent(Population)
WV	1790	NaN
AL	1800	NaN
IL	1800	1.9664
IN	1800	1.07078925956
MI	1800	1.42743161094
MS	1800	2.02289060421
DC	1800	1.07157894737
TN	1790	4.38249017682
OH	1800	1.18122215685
NV	1900	1.00417467207
AK	1920	1.30001181056
DE	1790	1.07376989607

< 1 ... 6 ... 9 >

State	Year	previous percent(Population)
NV	1950	1.03637069886
ID	1900	1.01055077678
GA	1800	1.00564992706
NH	1800	1.13014027021

< 1 ... 7 8 9 >

- Last NULL ordering shows NULL data last, after the sort of the specified column.

State	Year	previous percent(Population)
AK	1910	1.00129136651
VT	1800	1.00301296745
AK	1930	1.00307973467
NV	1900	1.00417467207
RI	1800	1.00431529241
GA	1800	1.00564992706
GA	1790	1.00821984733
ID	1900	1.01055077678
DE	1800	1.0107088942
HI	1900	1.01504096389
KY	1790	1.01589818543
WY	1900	1.01617650289

< 1 2 3 ... 9 >

State	Year	previous percent(Population)
MS	1800	2.02289060421
TN	1790	4.38249017682
KS	1800	NaN
AL	1790	NaN
ID	1790	NaN
IA	1800	NaN
UT	1800	NaN
SD	1790	NaN
NM	1790	NaN
NM	1800	NaN
OK	1800	NaN
AK	1790	NaN

< 1 ... 4 ... 9 >

State	Year	previous percent(Population)
ND	1790	NaN
WV	1790	NaN
AL	1800	NaN
KS	1790	null

< 1 ... 7 8 9 >

Changing number of rows to download

About this task

At the visual level, you can change the limit on the number of rows to download in a CSV file.

By default, the setting in the Maximum number of rows to fetch option specifies the number of rows to download in a CSV file. For table visuals only, the value of the Limit shelf is used first, but if this option does not specify a value, then the Maximum number of rows to fetch value applies. See .*Changing maximum number of rows to fetch*

Procedure

- To override the default, you can specify your own limit by navigating to the Data menu, and changing the setting in the Number of rows downloaded during CSV export and Detailed Data option.

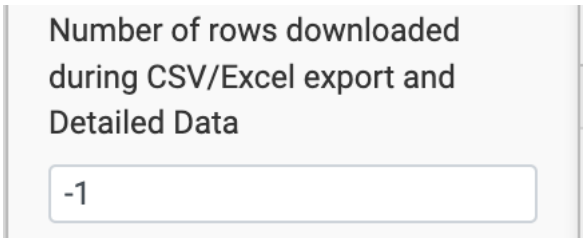

This setting overrides the Maximum number of rows to fetch option in the Data menu. For table visuals only, it also overrides the value in the Limit shelf.

Limiting rows downloaded to CSV file

Set the value of this option to 10.

After downloading this visual's data as CSV (see *Downloading visual data to CSV files*), check the number of rows.

Note that compared to the default download with value of -1 (left side), only 10 rows are present in the exported data, to match the number of rows specified earlier.

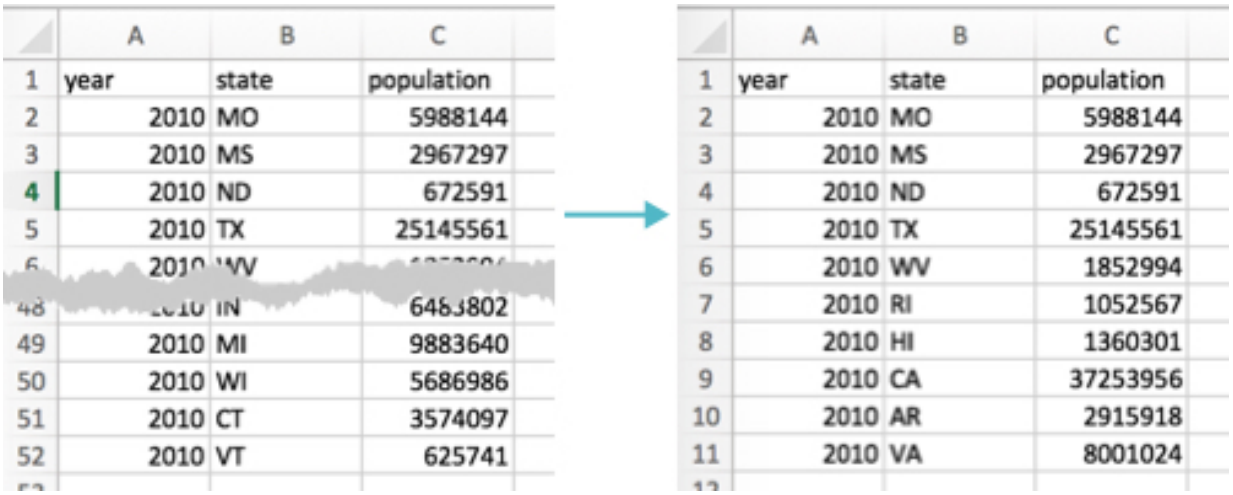

	A	B	C
1	year	state	population
2	2010	MO	5988144
3	2010	MS	2967297
4	2010	ND	672591
5	2010	TX	25145561
6	2010	WV	[illegible]
48	2010	IN	6483802
49	2010	MI	9883640
50	2010	WI	5686986
51	2010	CT	3574097
52	2010	VT	625741

	A	B	C
1	year	state	population
2	2010	MO	5988144
3	2010	MS	2967297
4	2010	ND	672591
5	2010	TX	25145561
6	2010	WV	1852994
7	2010	RI	1052567
8	2010	HI	1360301
9	2010	CA	37253956
10	2010	AR	2915918
11	2010	VA	8001024

Note: In Edit mode, a warning about the limit of the fetched rows is shown after opening a visual or dashboard with a limit. In View mode, the warning is suppressed.

Related Information

Changing maximum number of rows to fetch
Downloading visual data to CSV file

Changing maximum number of rows to fetch

About this task

You can provide an upper limit on the number of rows in a visual request. At the individual visual level, the number of rows in a visual request defaults to 5,000. So by default, the system fetches 5,000 rows at a time. At the individual visual level, the setting in the Maximum number of rows to fetch option overrides this default behaviour.

Note: For table visuals only, the value on the Limit shelf in the Visual Designer overrides this setting.

Procedure

To change the default setting, navigate to the Data menu, and change the value in the text box for Maximum number of rows to fetch.

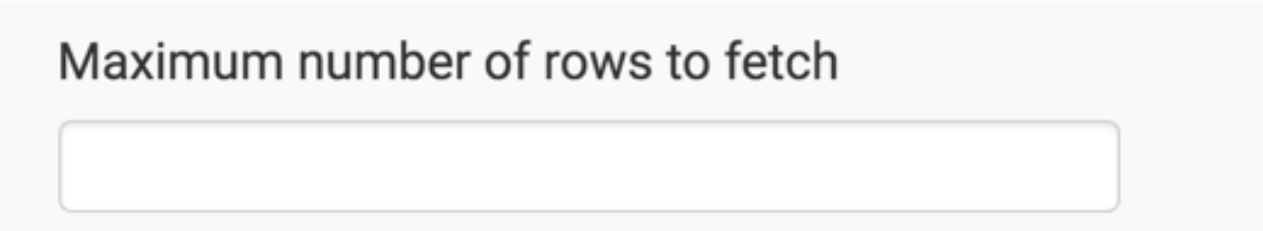

At the dashboard level, you can change this setting on the Data tab of a filter widget.

Note: In Edit mode, a warning about the limit of the fetched rows is shown after opening a visual or dashboard with a limit. In View mode, the warning is suppressed.

Disabling incremental queries

To disable incremental queries for this visual, navigate to the Data menu, and select Disable incremental queries, overriding dataset level settings.

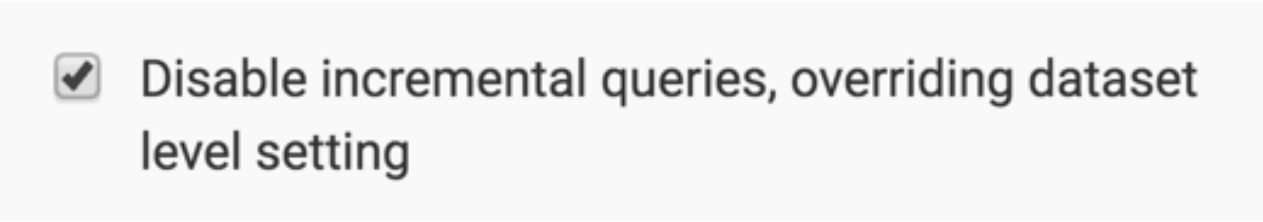

Displaying NULL rows

To change the data display of NULL values in the visual, navigate to the Data menu, and change the mask under the heading Display null value as:.

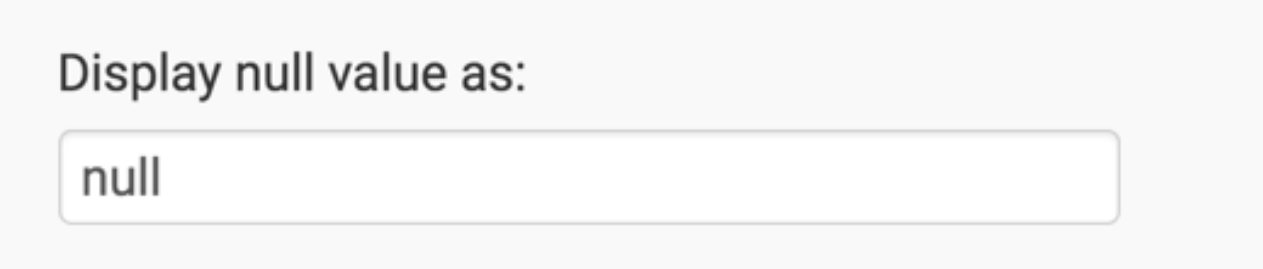

By default, this value is null. This is how the visual appears:

State	Year	previous percent(Population)
KS	1790	null
KS	1800	NaN
AL	1790	NaN
ID	1790	NaN
IA	1800	NaN
UT	1800	NaN
SD	1790	NaN
NM	1790	NaN
NM	1800	NaN
OK	1800	NaN
AK	1790	NaN
IA	1790	NaN

If the value is cleared (a blank), this is how the visual appears:

State	Year	previous percent(Population)
KS	1790	
KS	1800	NaN
AL	1790	NaN
ID	1790	NaN
IA	1800	NaN
UT	1800	NaN
SD	1790	NaN
NM	1790	NaN
NM	1800	NaN
OK	1800	NaN
AK	1790	NaN
IA	1790	NaN

If the value is set to a few dashes (---), this is how the visual appears:

State	Year	previous percent(Population)
KS	1790	—
KS	1800	NaN
AL	1790	NaN
ID	1790	NaN
IA	1800	NaN
UT	1800	NaN
SD	1790	NaN
NM	1790	NaN
NM	1800	NaN
OK	1800	NaN
AK	1790	NaN
IA	1790	NaN

Enabling URL links

Procedure

- To render values that contain usable http addresses as navigation links, navigate to the Data menu, and select Show values starting with http(s):// as links.

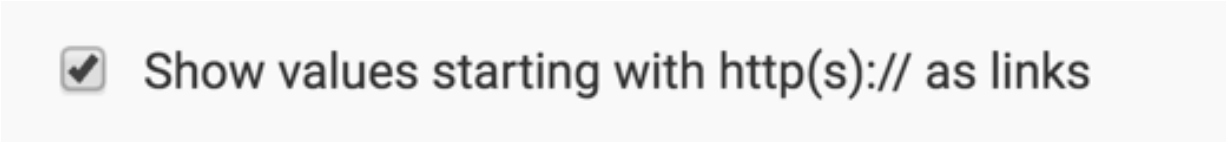

- Compare the appearance of the table visual with http... content before and after the preceding selection.

Figure 9: Column of URL Addresses Converted to Active Links

Specifying qualitative range as percentages

About this task

The Gauge and Bullet visuals use the Qualitative Ranges shelf to define the fit of the main value (on the Measure shelf) relative the value specified by the Compare To shelf. An alternate approach is to specify these ranges as percentages.

Procedure

- To use the percentage option, navigate to the Data menu, and select Use percentages of "Compare To" value if no qualitative data provided. Then specify the percentage value for the Lowest, Middle, and Highest ranges.

  By default, these are 60%, 80%, and 100%, respectively. Note that the scale accommodates up to 200%, to track outperforming categories.

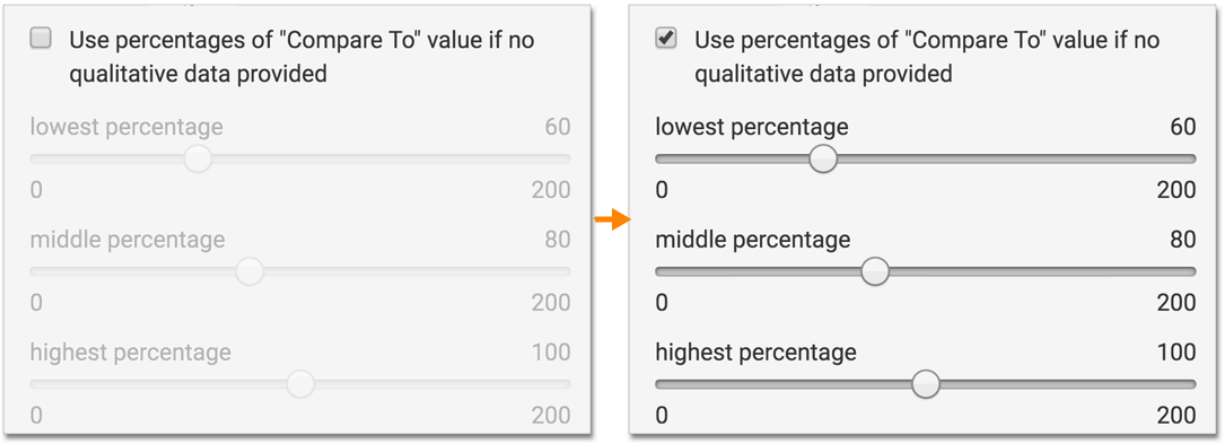

The visual on the left shows the ranges when they are explicitly defined as scalar values 600, 700, and 800. The visual on the right shows the qualitative ranges as percentages: 60%, 80%, and 100%.

Customizing links

About this task

Several display options may be managed from the Links menu, depending on the type of the visual.

To get to the options in the Links menu, follow these steps:

Procedure

1. On the right side of the Visual Designer, click the Settings menu.

2. In the Settings menu, click Links.

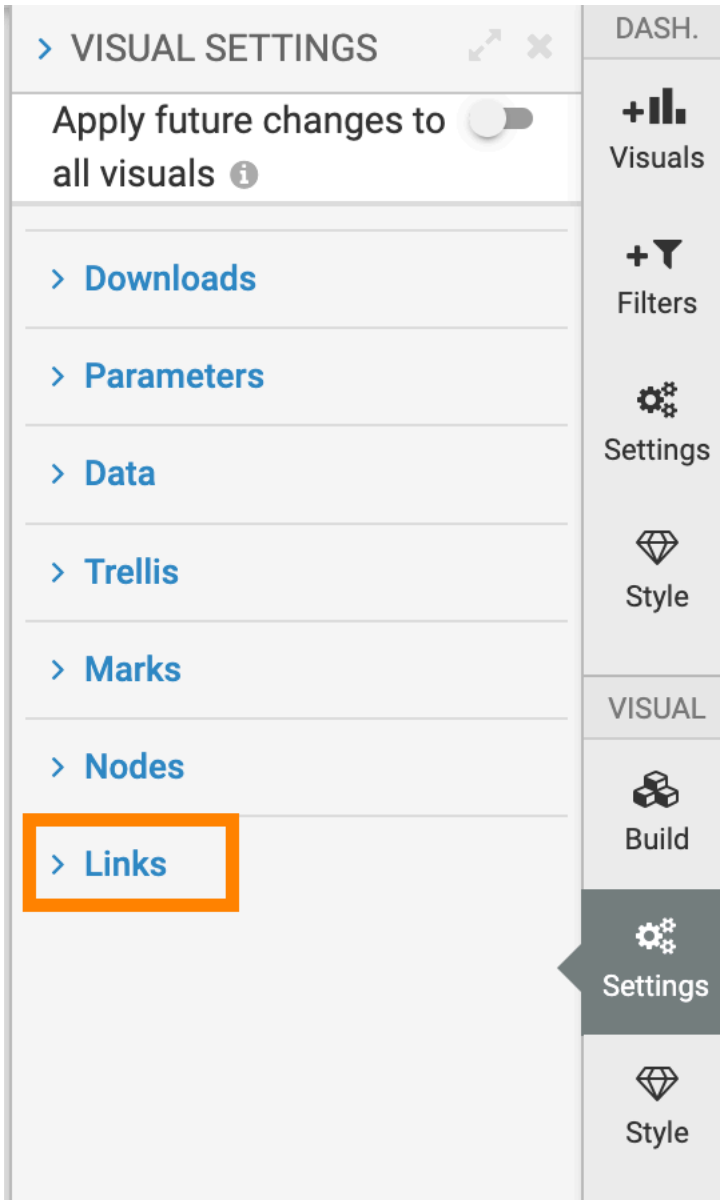

3. Under Links, select one of the following options:

Changing link color

To change the color of the links, navigate to the Links menu, and change the selection in the Default Link Color menu. These options are:

- source
- target
- link width (default)
- fixed

Compare the appearance of a network visual with links based on source node, and links based on target node.

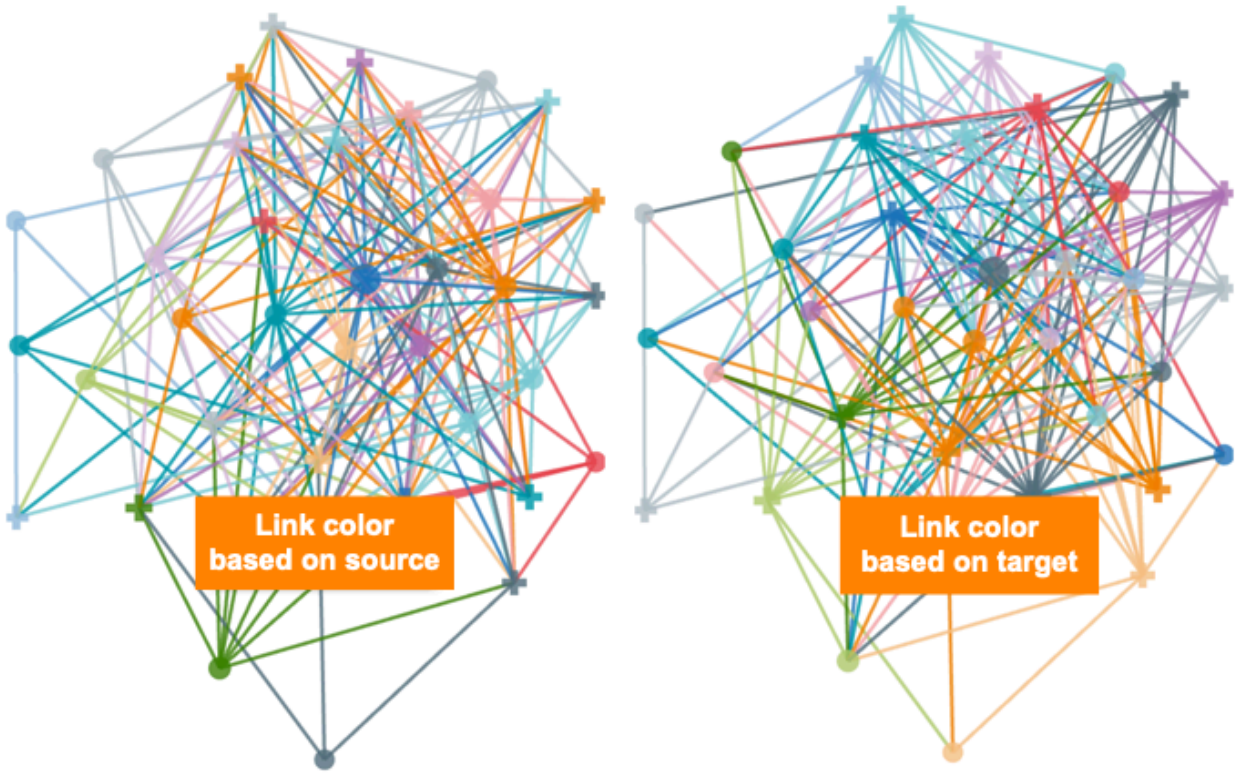

Here, compare the same dendrogram visual with links based on source, target, aggregate, and fixed.

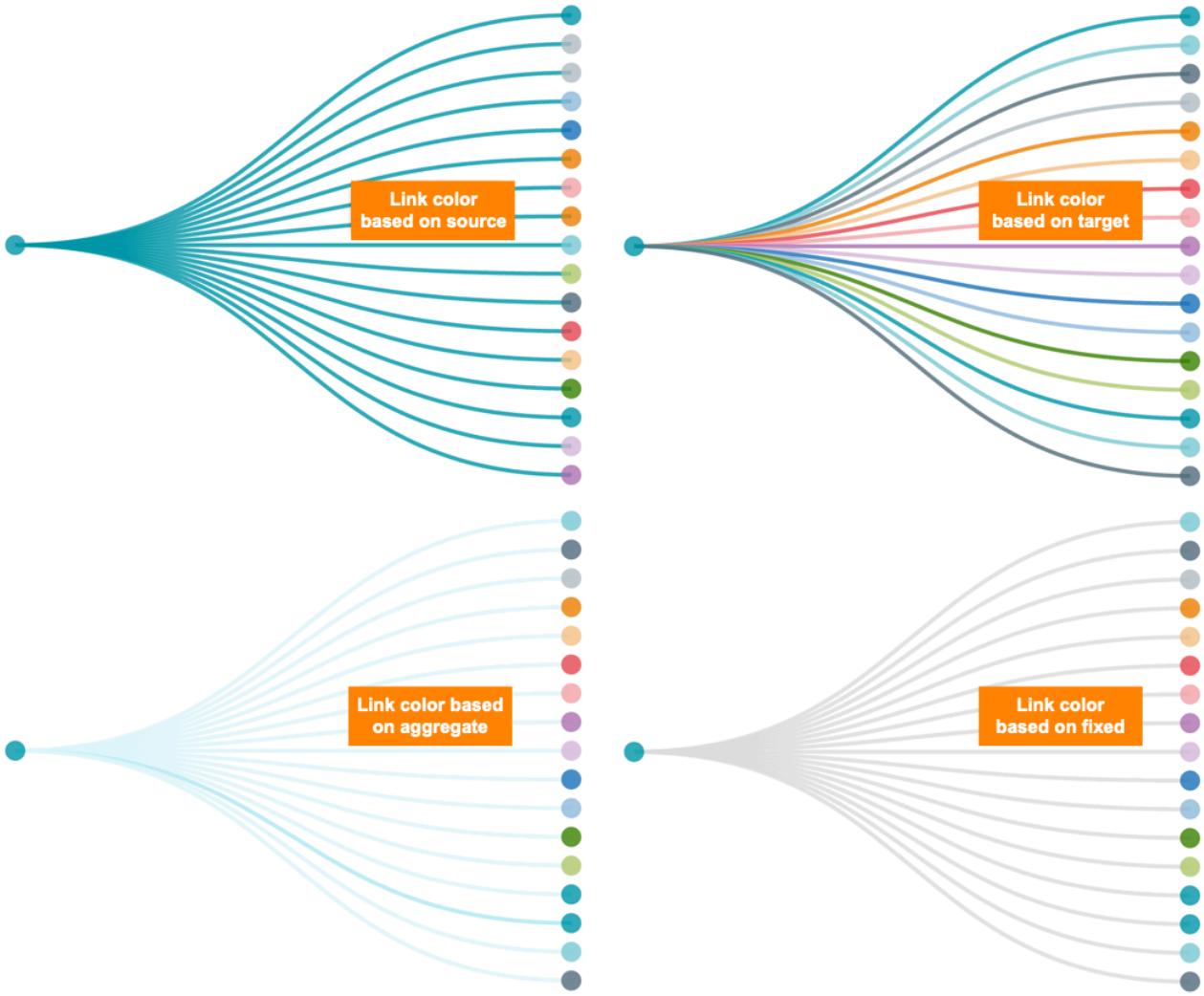

Changing link opacity

To change the opacity (color saturation) of links, navigate to the Marks menu, and adjust the selector for the Link Opacity option. These measurements are 80 by default.

Compare the appearance of a visual with link opacity of 80 and 20.

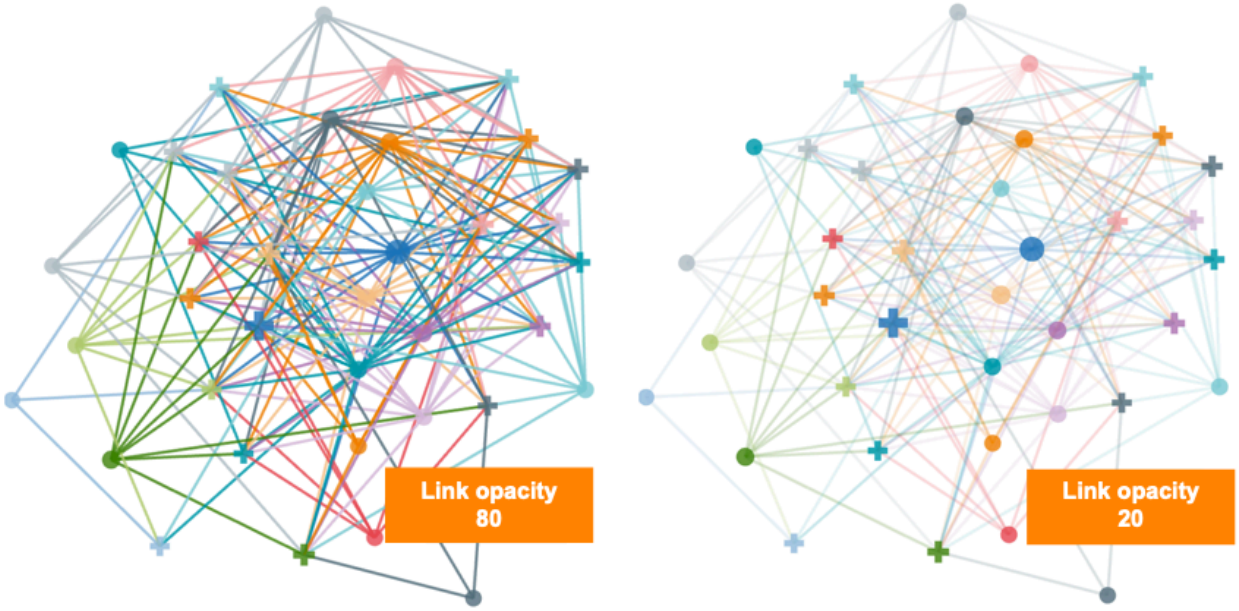

Changing link width range

To change the width of the link, navigate to the Links menu, and change the minimum and the maximum in the Link width range option. The default setting is 4-10 pixels.

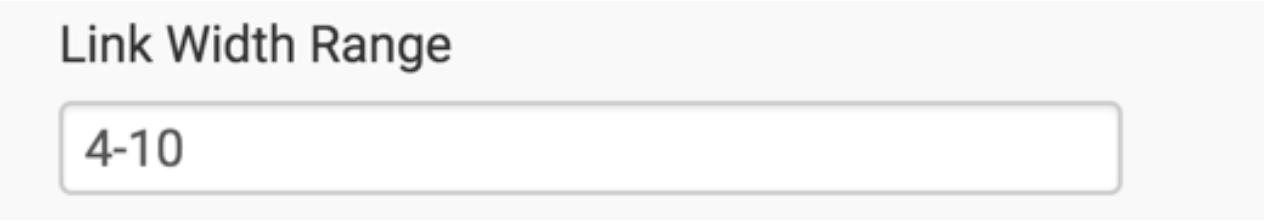

Compare the appearance of a visual with link width range of 2-2 (on the left) with a visual that uses a link width range of 4-4 (on the right).

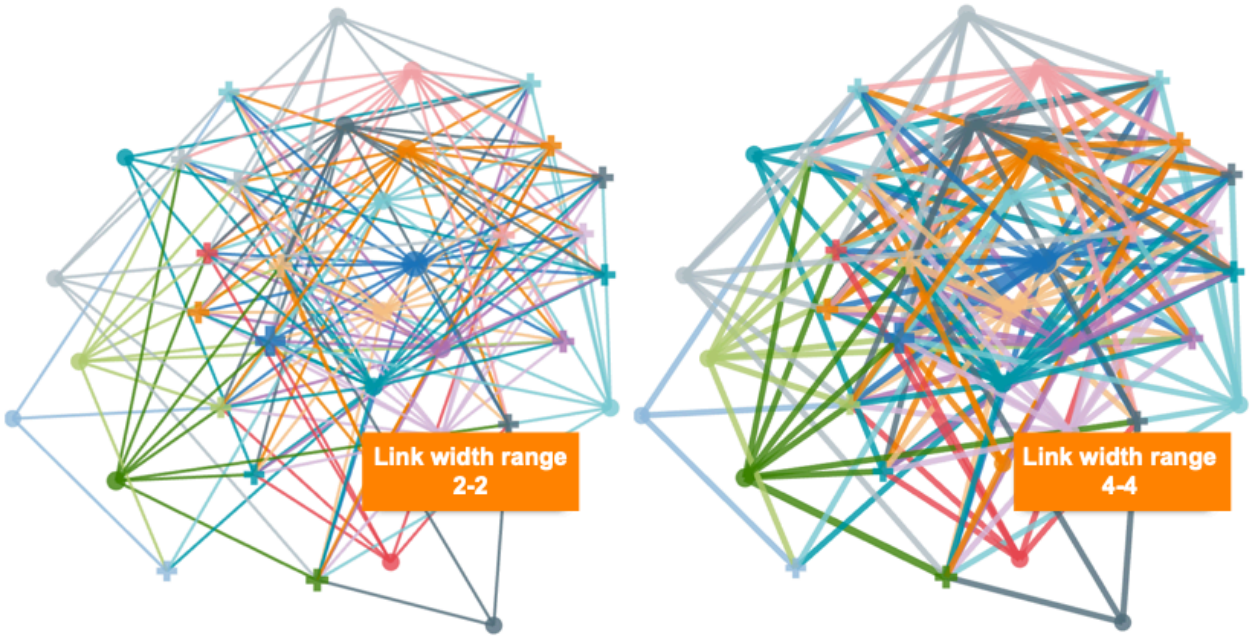

Changing link distance

To specify the distance between links, navigate to the Links menu, and change the value in the Link distance selector.

The default value is 80, and the valid range is between 0 and 500.

Note how changing the link distance setting alters the visual. Here is the same network visual, with link distance 5 and 100.

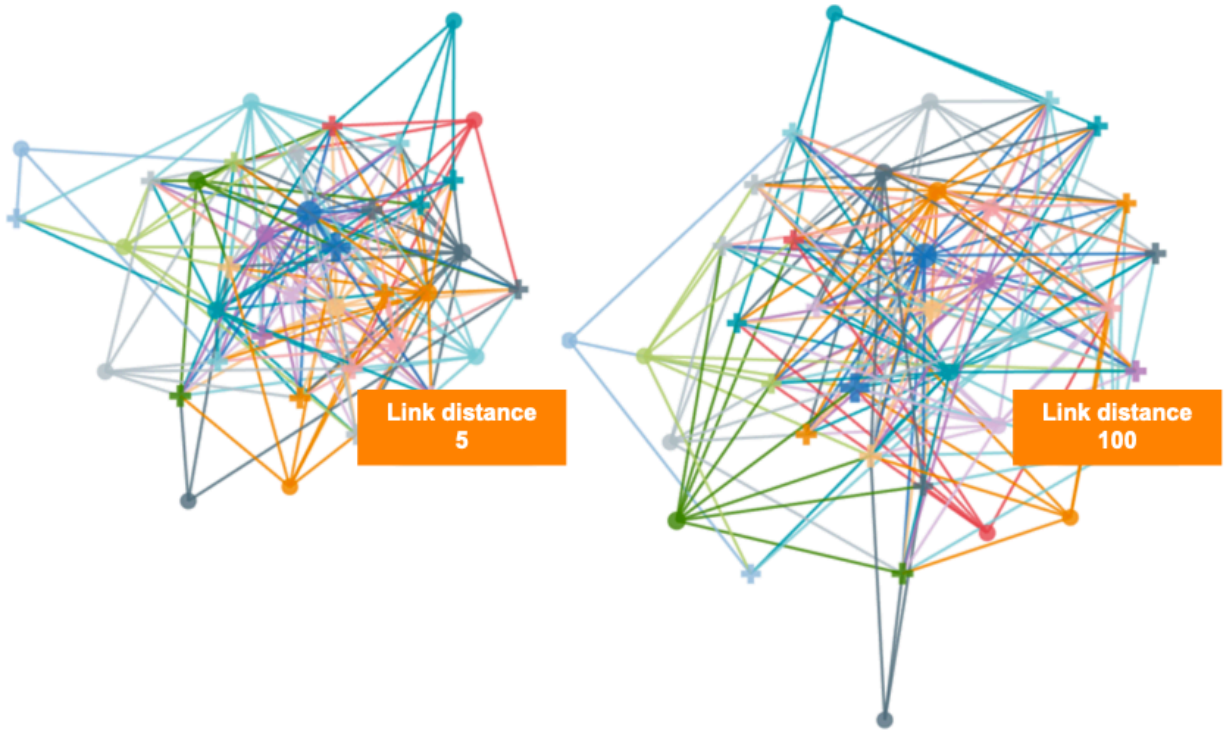

Changing link strength

To change the strength of a link, navigate to the Links menu, and change the minimum and the maximum in the Link strength option. The default setting is 0.1, and the valid range is 0 to 1.

Compare the appearance of a visuals with link strength of 0.01, 0.1, and 1.

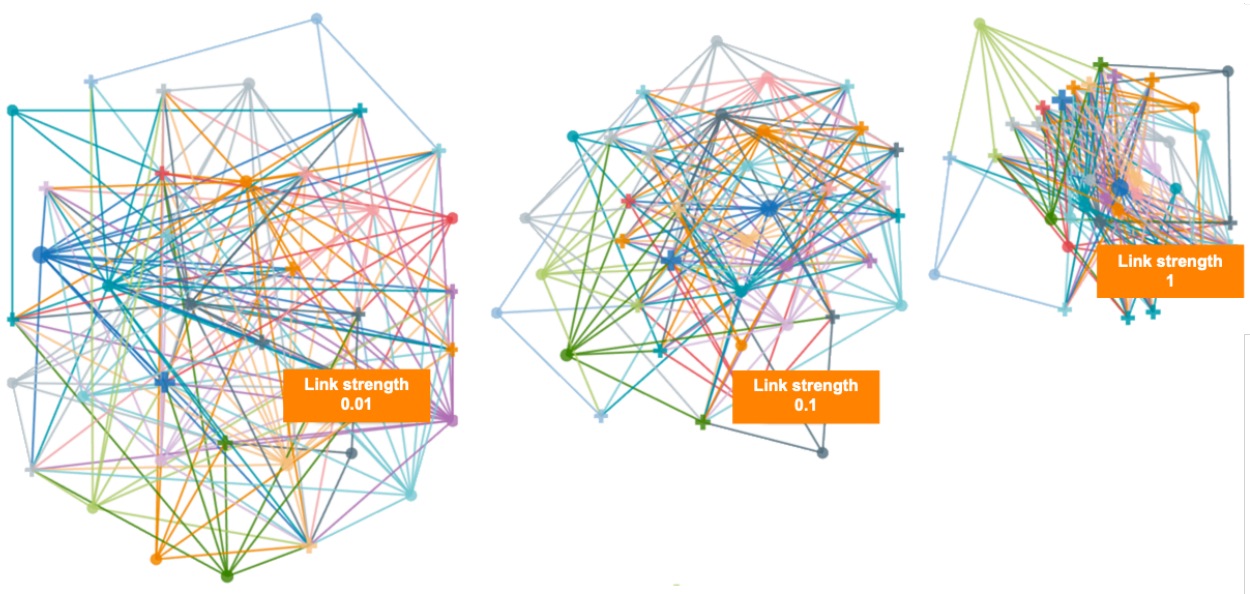

Adding arrow direction to links

To add a direction to the links, navigate to the Links menu, and select the Add direction arrows to links option.

After you select this option, Direction Arrow Size selector can be changed.

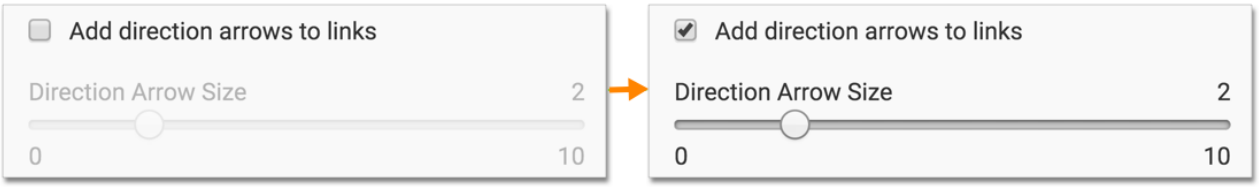

Changing direction arrow size

To change the size of the direction arrow, navigate to the Links menu, and change the Direction Arrow Size option on the selection slider.

The default value is 2, and the valid range is 0 to 10.

Note that you can change this value only after selecting the Add direction arrows to links option, as described in *Adding Arrow Direction to Links*.

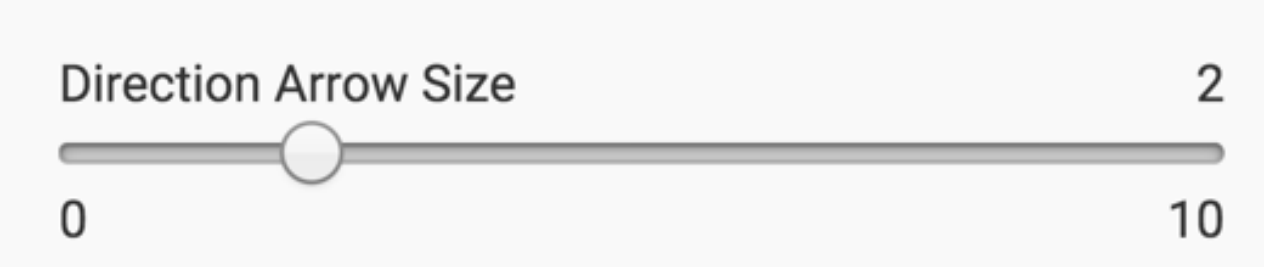

Related Information

Adding arrow direction to links

Adding link legend

In Cloudera Data Visualization, you can enable a legend for network links.

To add a legend for links, navigate to the Links menu, and select the Add link legend option.

After you select this option, you can change the Legend Title value. This value is Link colors by default.

In the following image, notice that selecting this option adds a legend to the links of a network visual.

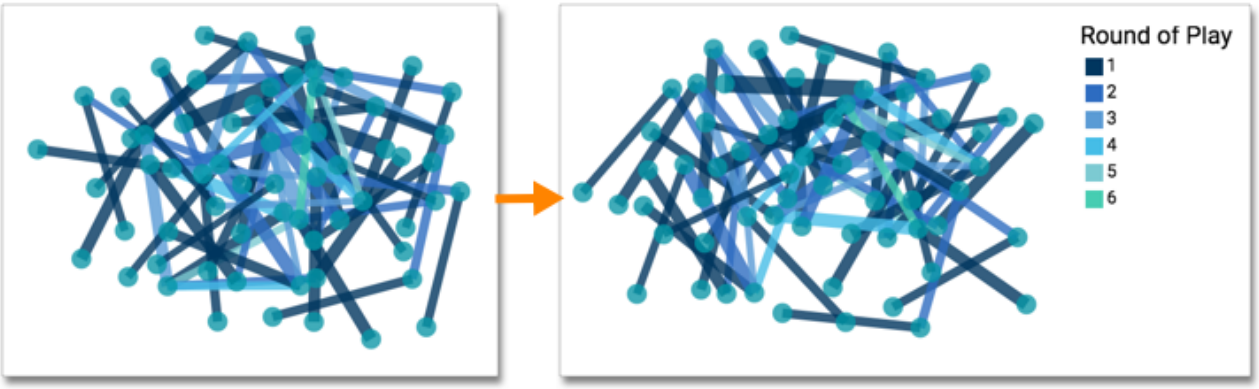

Changing the name of the links legend

To change the name of the links legend, navigate to the Links menu, and change the text in the Legend Title option.

The default value is Link colors.

Note that you can change this value only after selecting the Add link legend option, as described in *Adding the Link Legend*.

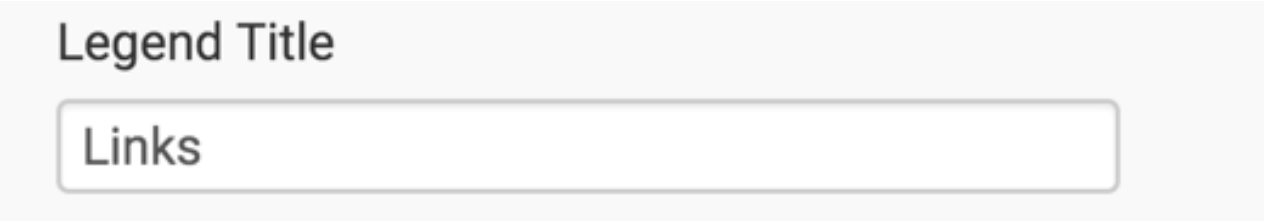

Related Information

Adding link legend

Customizing map options

About this task

The Map menu specifies the underlying map view selection for choropleth maps. To get to the options in the Map menu, follow these steps:

Procedure

1. On the right side of the Visual Designer, click the Settings menu.

2. In the Settings menu, click Map.

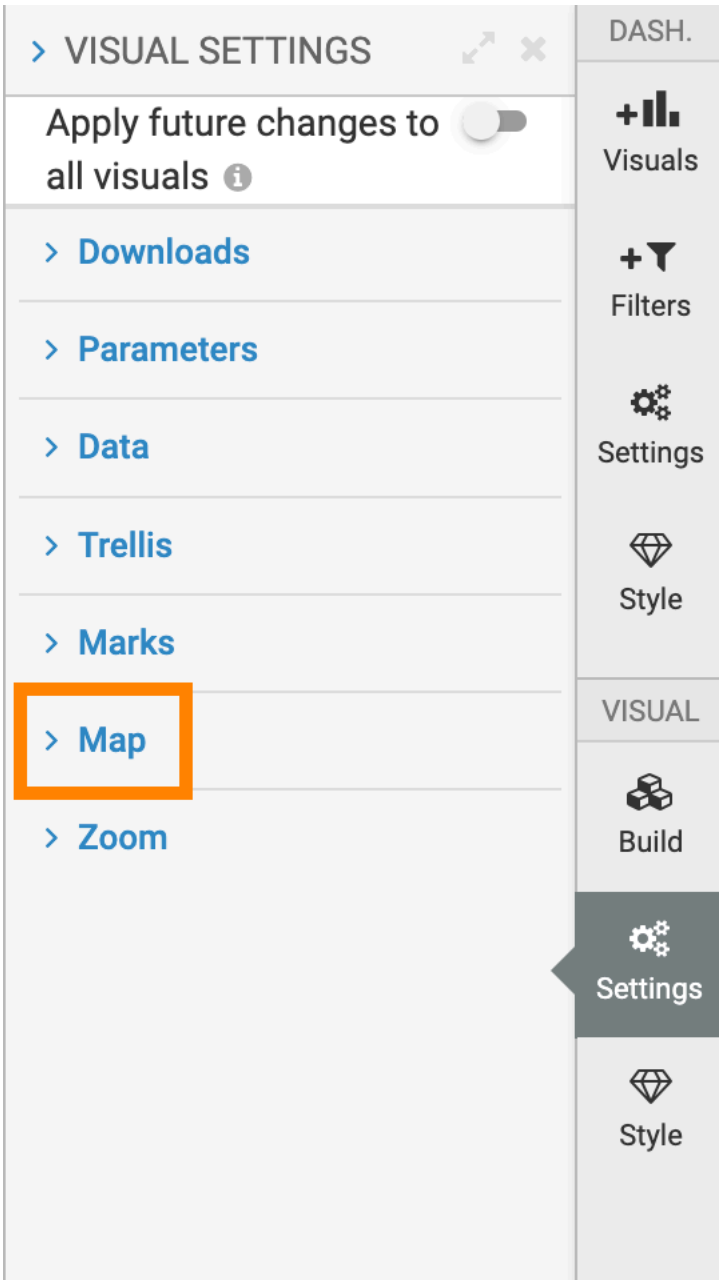

3. Under Map, select the Map View option.

Selecting map view

To change the map shapefile (World, United States, Canada), navigate to the Map menu, and change the selection in the Select Map View menu. This value is Auto-Detect, by default.

Example

In this example, we are specifying that the visual must use a Canadian map.

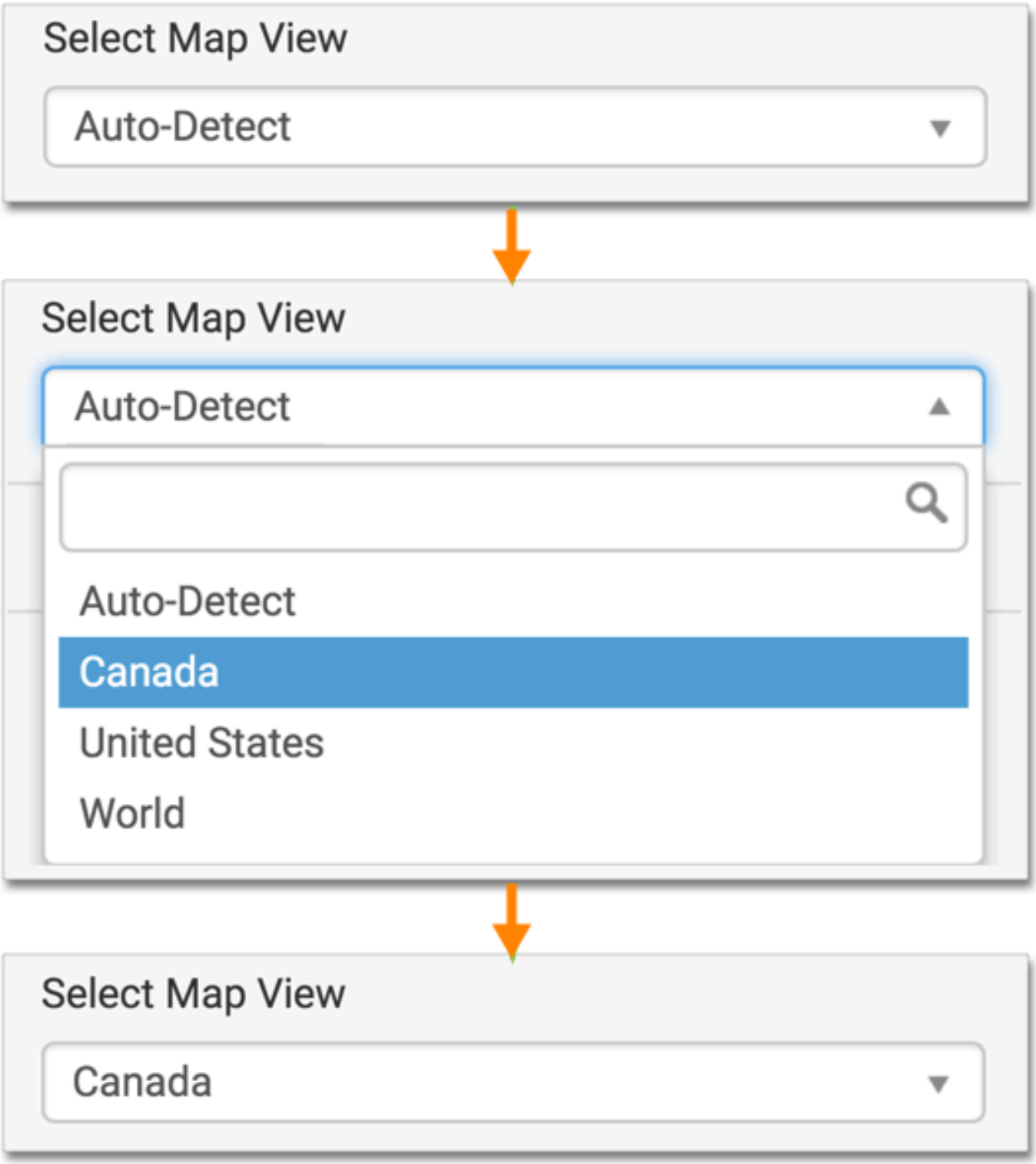

Customizing marks

About this task

The Marks menu, offers various display options based on your visual type, allowing you to fine-tune the appearance of chart elements. To access the Marks settings, follow these steps:

Procedure

1. On the right side of the Visual Designer, click the Settings menu.
2. In the Settings menu, click Marks to expand the available options.

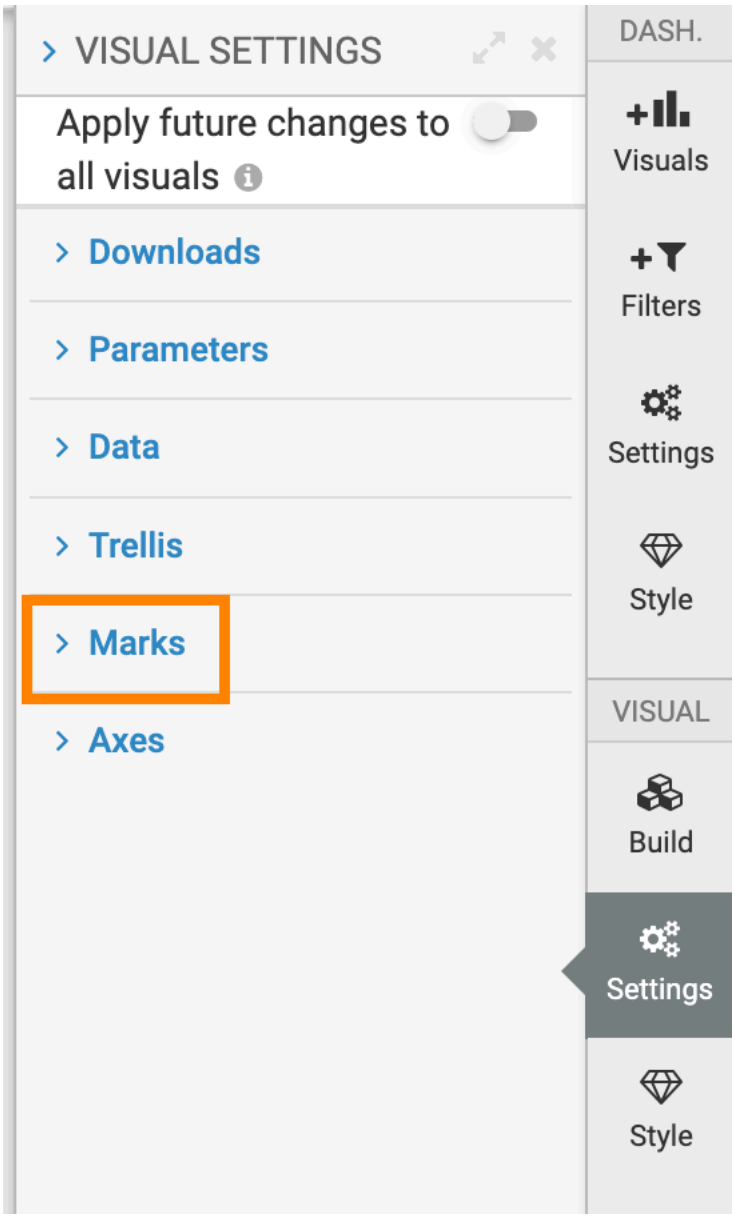

The Marks menu provides the following options.

> **Remember:** This is the full list, but the available settings in the UI will vary depending on the type of visual you are customizing.

Displaying tooltips

About this task

Cloudera Data Visualization offers you the option to add tooltips to your visuals. Tooltips provide additional information when you hover over data points.

They allow you to pick measure fields from the data menu to add additional information to the graph's tooltip. The Display Tooltips option is available for all visuals, and on most visual types, this shelf accepts multiple fields.

For certain visuals like sparklines, adding tooltips to your visual provides access to additional options to show values in the tooltip. You can customize your tooltip settings by the selecting specific options as needed.

To turn tooltips on, navigate to Settings Marks and select the Display Tooltips option.

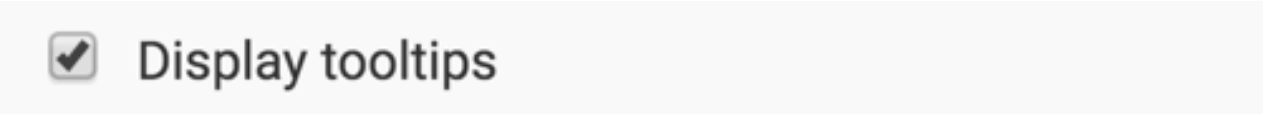

Enabling minimum value in tooltips

You can enhance your certain visuals (for example Sparklines) by displaying the minimum value in tooltips.

To display the minimum value in a tooltip, navigate to Settings Marks , select Display Tooltips, and then select Enable minimum value in tooltips.

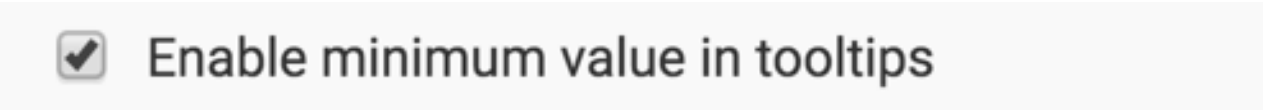

Once enabled, the minimum value will be included in the tooltip when hovering over your visual.

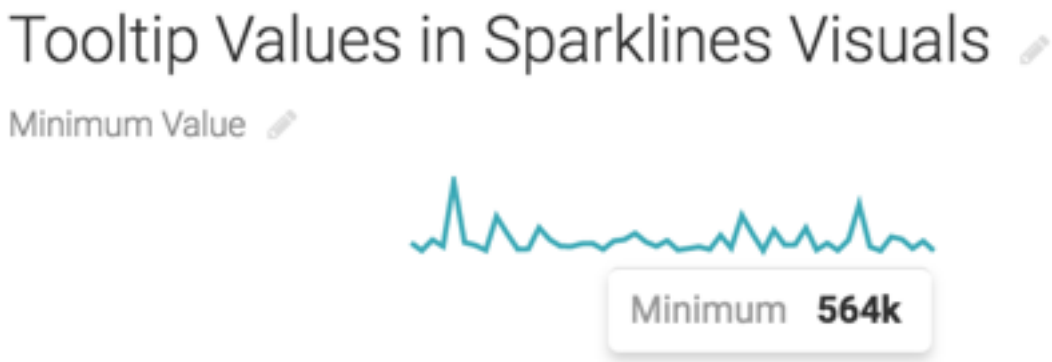

Enabling maximum value in tooltips

You can enhance your certain visuals (for example Sparklines) by displaying the maximum value in tooltips.

To display the minimum value in a tooltip, navigate to Settings Marks , select Display Tooltips, and then select Enable maximum value in tooltips.

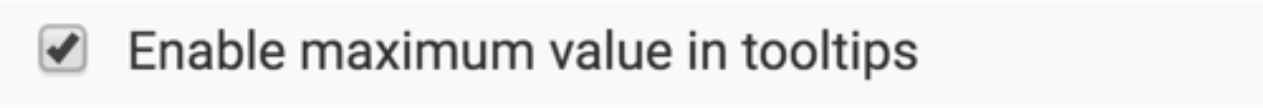

Once enabled, the maximum value will be included in the tooltip when hovering over your visual.

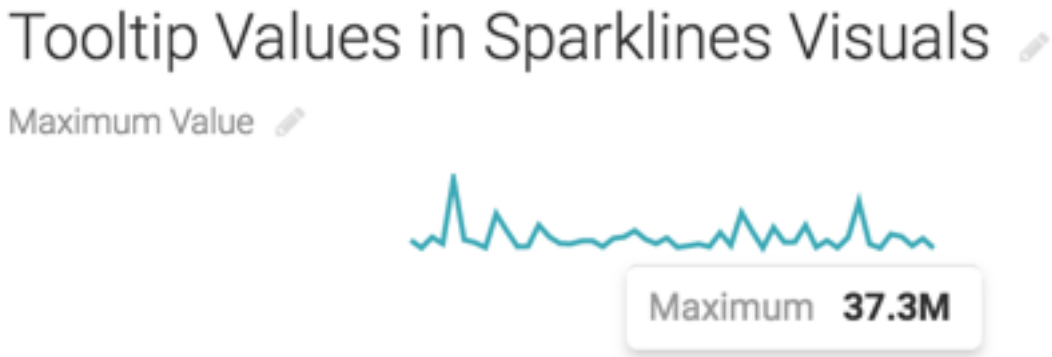

Enabling first value in tooltips

You can enhance your certain visuals (for example Sparklines) by displaying the first value in tooltips.

To display the first value in a tooltip, navigate to Settings Marks , select Display Tooltips, and then select Enable first value in tooltips.

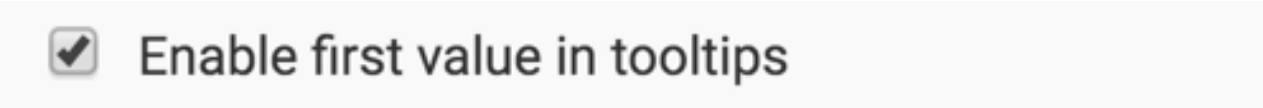

Once enabled, the first value will be displayed in the tooltip when hovering over your visual.

Enabling last value in tooltips

You can enhance your certain visuals (for example Sparklines) by displaying the last value in tooltips.

To display the first value in a tooltip, navigate to Settings Marks , select Display Tooltips, and then select the Enable last value in tooltips option.

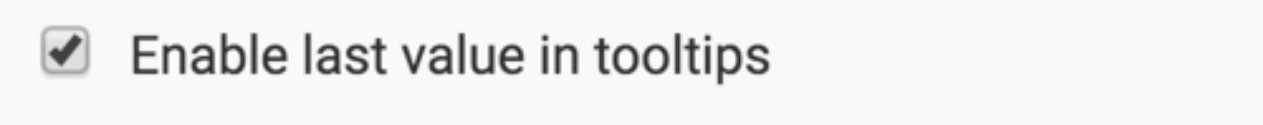

Once enabled, the last value will be displayed in the tooltip when hovering over your visual.

Showing detail data button in context menu

About this task

You can click on a part of a visual and see the matching rows of data, in detail. This setting is available on all visual types.

Use this feature to ensure that the Detail Data selector appears in the context menu. In Edit mode, navigate to the Marks menu, and select the Show detail data button in context menu option. It is on by default.

If this option is selected, notice that the Show Detailed Data option appears in the context menu.

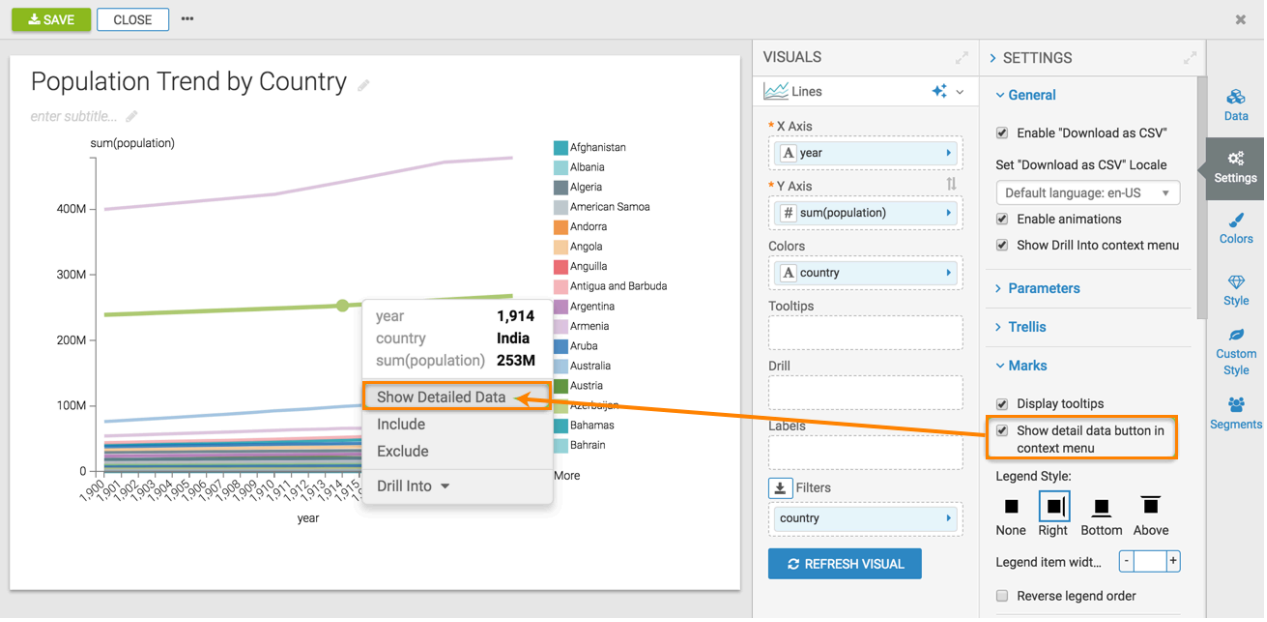

- To see how this option works, see *Discovering detailed data behind a visual*.
- To hide detail data, de-select the Show detail data button in context menu option.

Related Information

Discovering detailed data behind a visual

Showing the context menu

About this task

You can click on a part of a visual and see the context menu. This option is only available for Table and Queries visuals.

To turn the context menu option on or off, navigate to the Marks menu, and select Show detail data button in context menu.

The following image shows the context menu in a table visual after the option is selected. To hide the context menu in a visual, de-select this option.

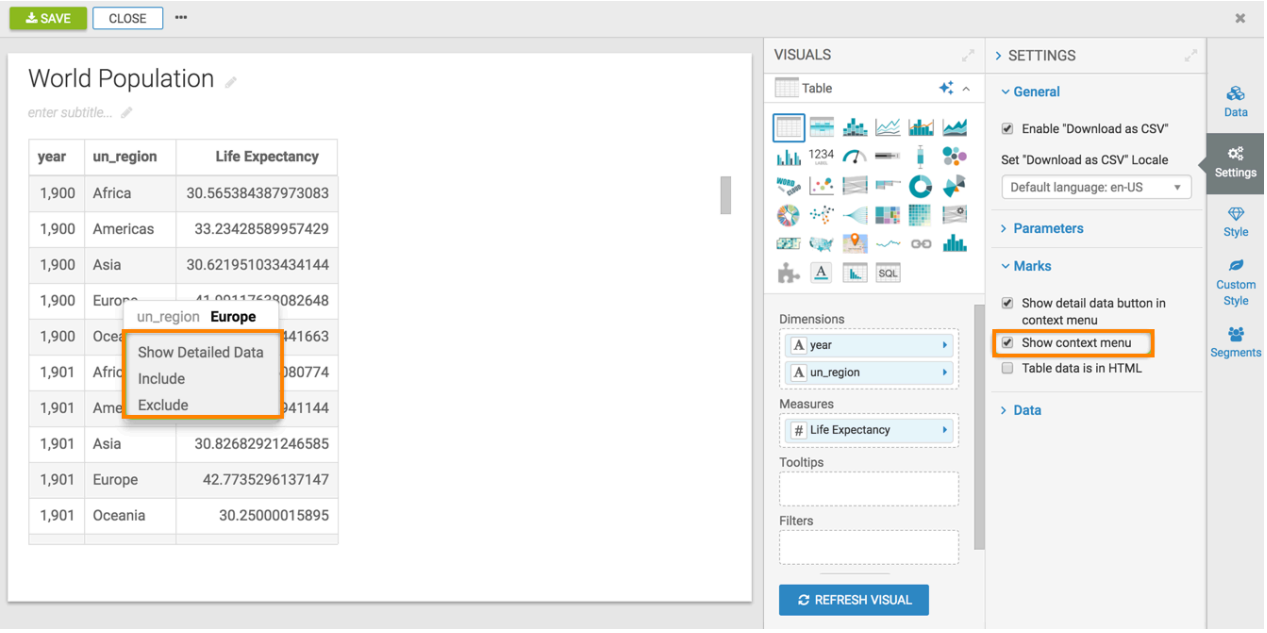

The context menu includes the following options:

- Show Detailed Data
- Include: Filter Data Using the Include Option in the Context Menu
- exclude: Filter Data Using the Exclude Option in the Context Menu

Showing table data as HTML

To turn this option on or off, navigate to the Marks menu, and select Table Data is expected to be in HTML.

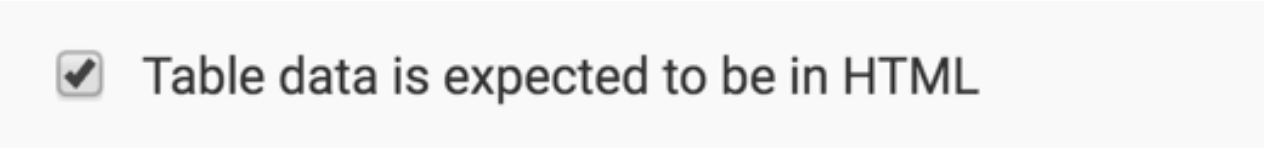

Showing adjacent duplicate values

To turn this option on or off, navigate to the Marks menu, and select Show adjacent duplicate values.

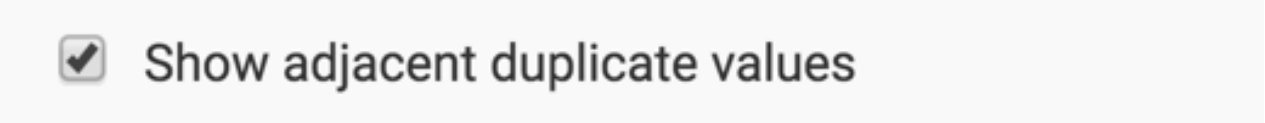

Showing expansion totals above expansion rows

In cross tabulation visuals, the subtotals and combined total appear under the detailed data, by default. To display subtotals and combined total over the expanded rows, navigate to the Marks menu, and select Show expansion totals above the expanded rows.

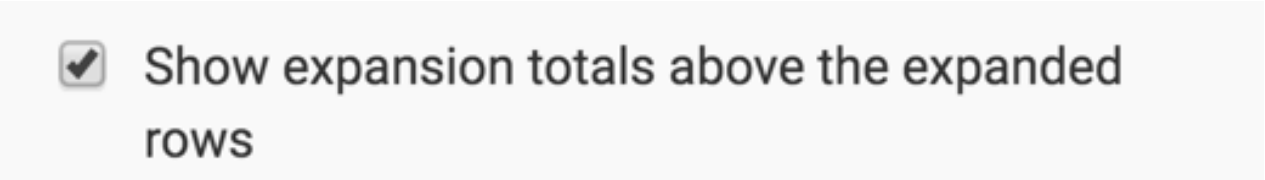

Before selecting this option, notice that the subtotals and combined total appear at the bottom of expanded rows.

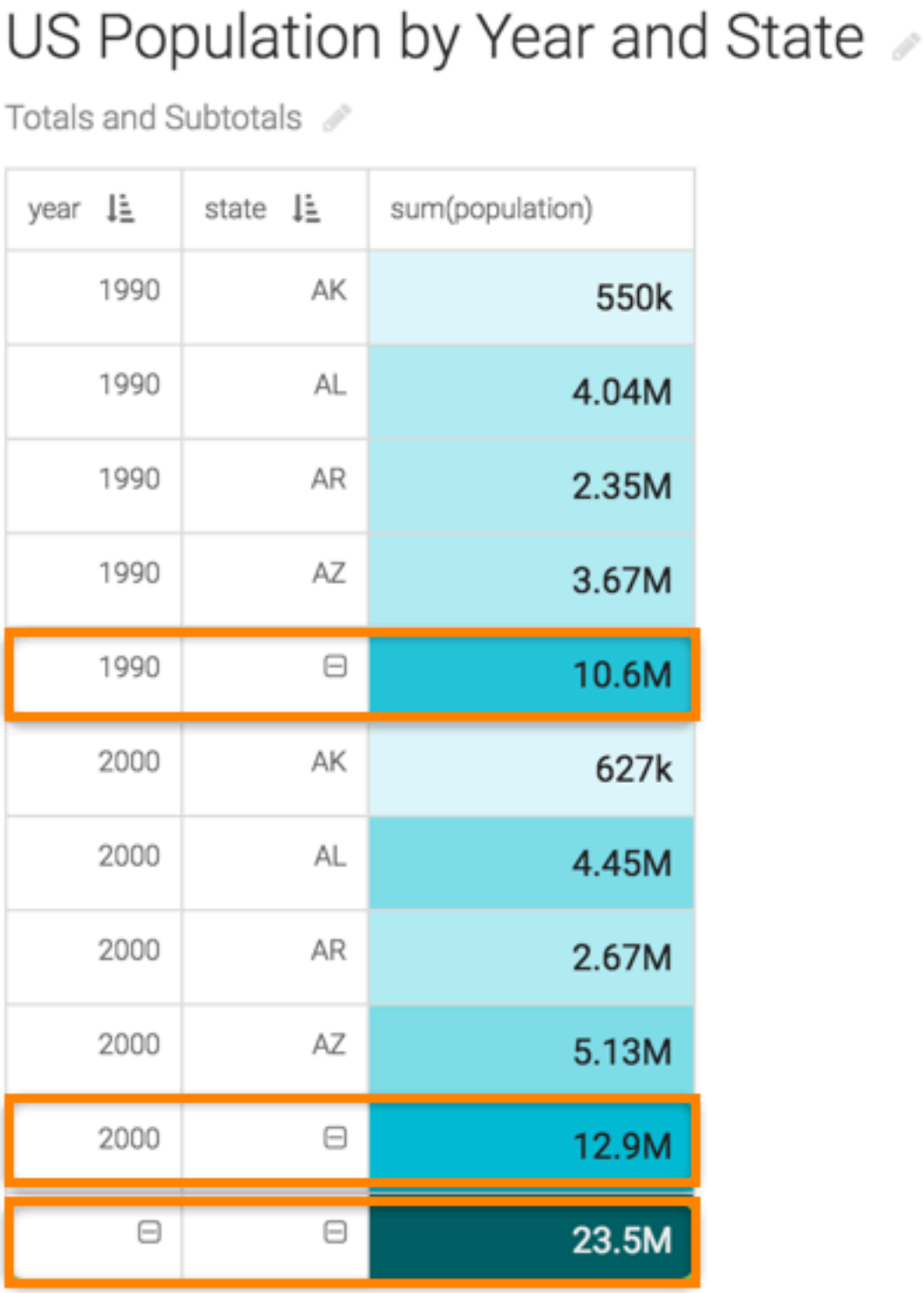

After selecting this option, notice that the subtotals and combined total appear over the expanded rows.

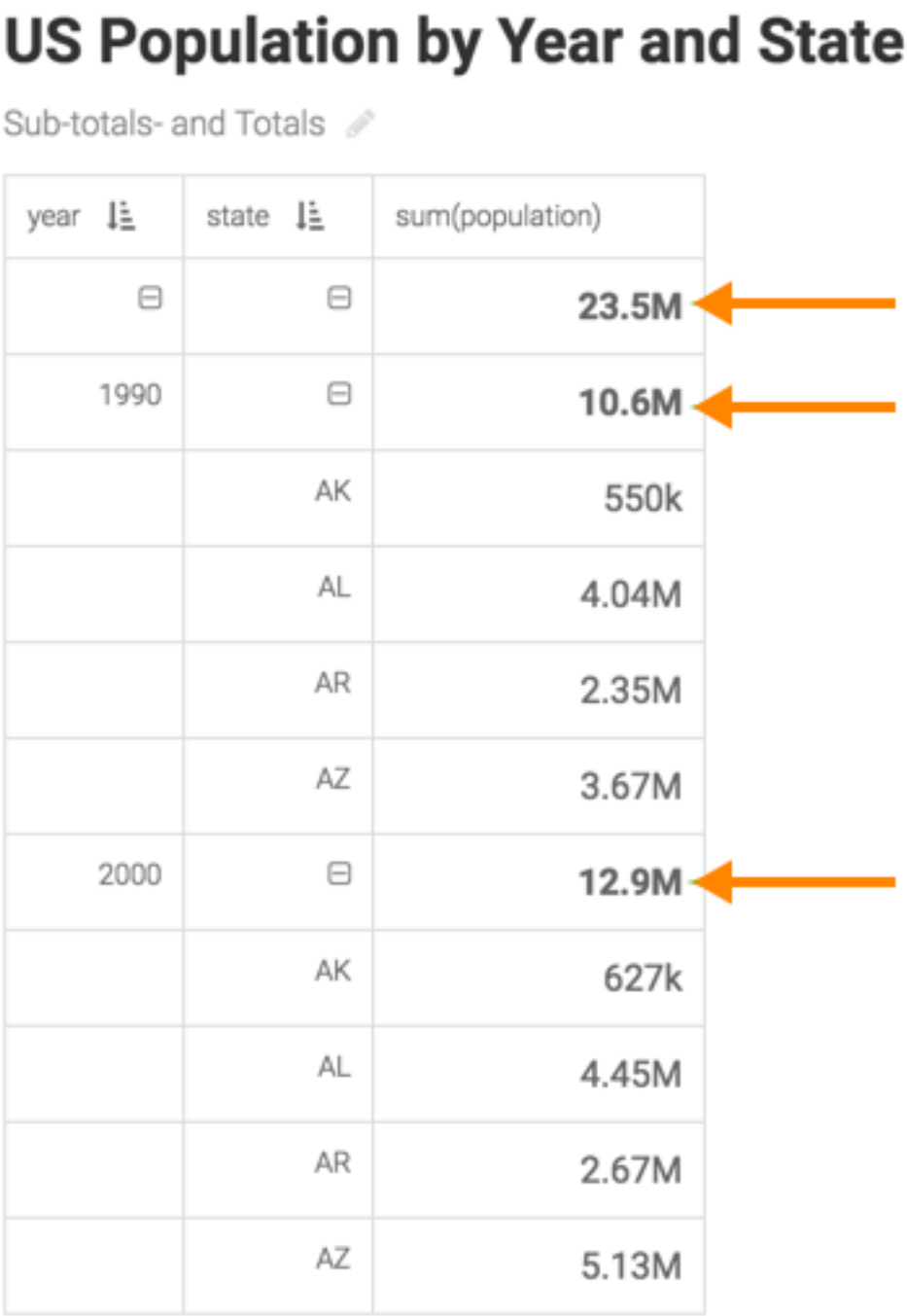

Displaying title for subtotal rows

You can add a title for the subtotal rows. This option is available only on cross tabulation visuals when expansion is enabled on at least one of the Row shelf dimensions.

Navigate to the Marks menu and enter a title for the subtotal row under the Subtotal Title option.

In the following example, compare the visual on the left without a title to the one on the right with a title. In our example, we added the title Subtotal.

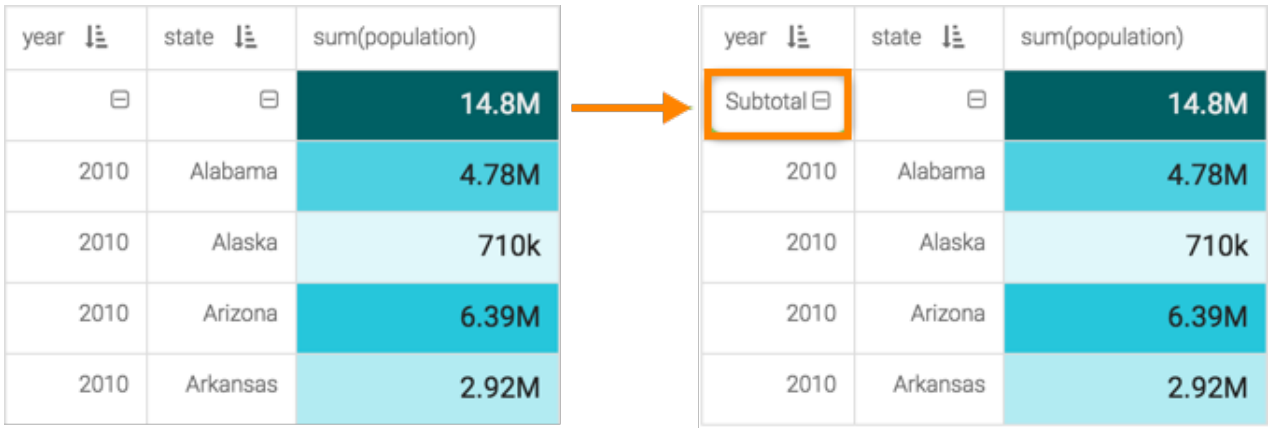

year	state	sum(population)
⊟	⊟	14.8M
2010	Alabama	4.78M
2010	Alaska	710k
2010	Arizona	6.39M
2010	Arkansas	2.92M

year	state	sum(population)
Subtotal ⊟	⊟	14.8M
2010	Alabama	4.78M
2010	Alaska	710k
2010	Arizona	6.39M
2010	Arkansas	2.92M

Adding and removing legend

Procedure

- Legend is on by default. It appears on the right side of the visuals. To add a legend below or above a visual, navigate to the Marks menu, and select the appropriate Legend Style.

- To remove the legend, select Marks Legend Style None .

Example

Here is an example of a line chart with a legend showing. The legend represents dimensions as distinct colors.

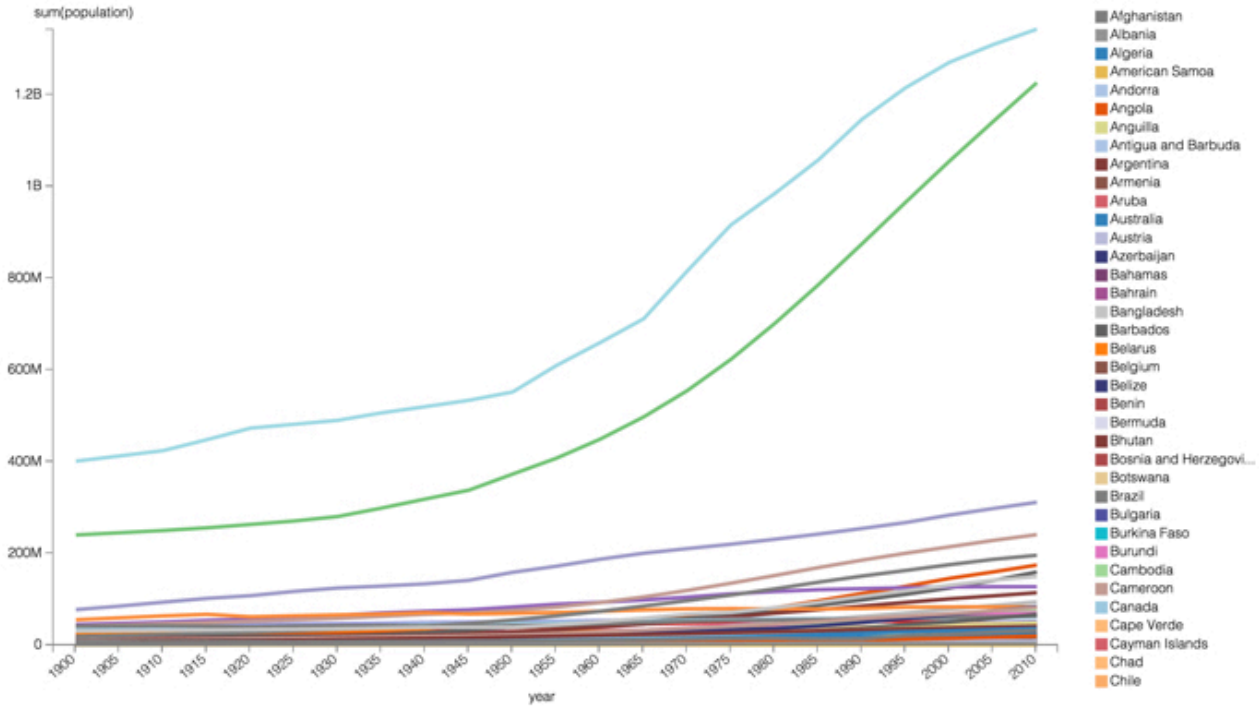

For graphs that map a continuum of values, such as choropleth maps, the legend shows a sliding scale of color and values at the minimum, median, and maximum.

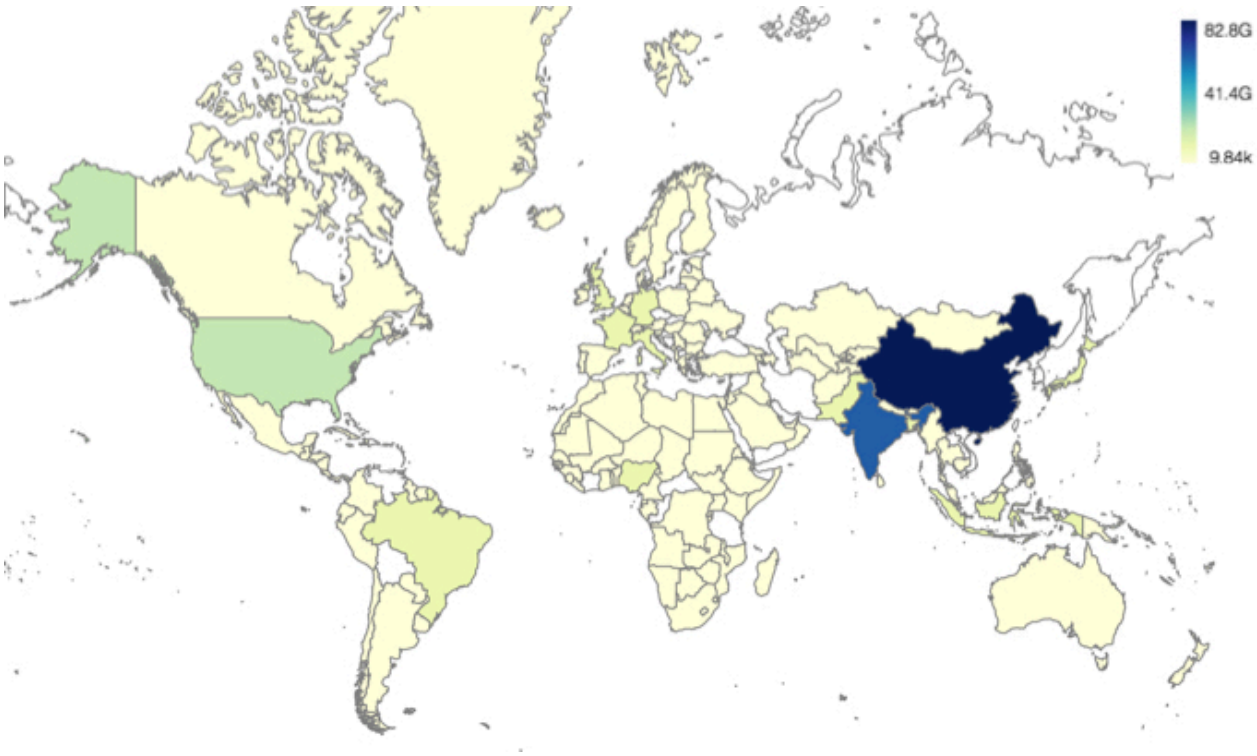

For timeline visuals, you can add a legend by adding a dimension to the color shelf. This example shows you a timeline where the different latitude values are marked by different colors, and the legend is displayed on the right side of the visual.

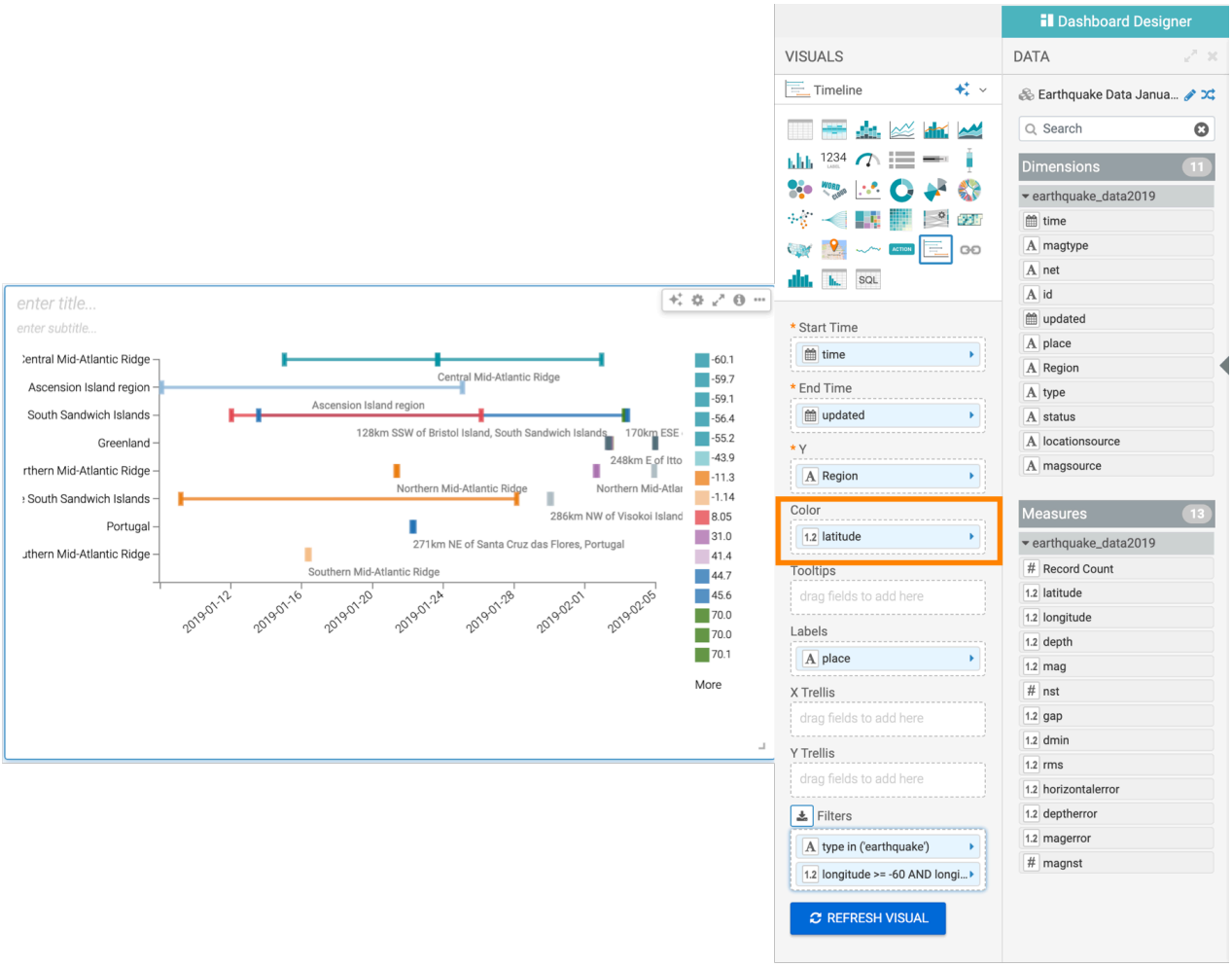

Changing legend item width

To change the width of the items inside a legend, navigate to the Marks menu, and make adjustments to the selector for Legend Item Width. The scale is in pixels.

Changing legend width

To change the width of the legend, navigate to the Marks menu, and make adjustments to the selector for Legend width. The valid range is 100 to 250, and the default value is 150 pixels.

Reversing legend order

To turn this option on or off, navigate to the Marks menu, and select the Reverse legend order option.

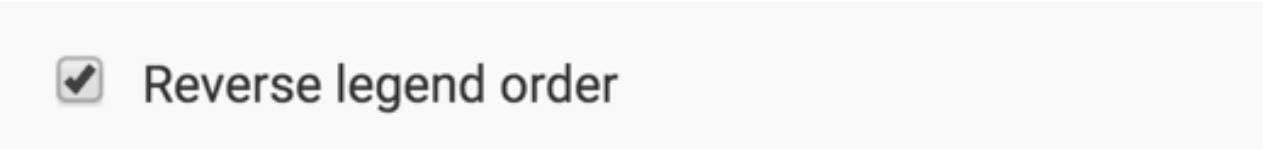

In the case of distinct dimension mapping, the default legend appears in ascending alphabetical order; reversing it renders the legend in descending alphabetical order. Note that the dimension-color mapping remains the same.

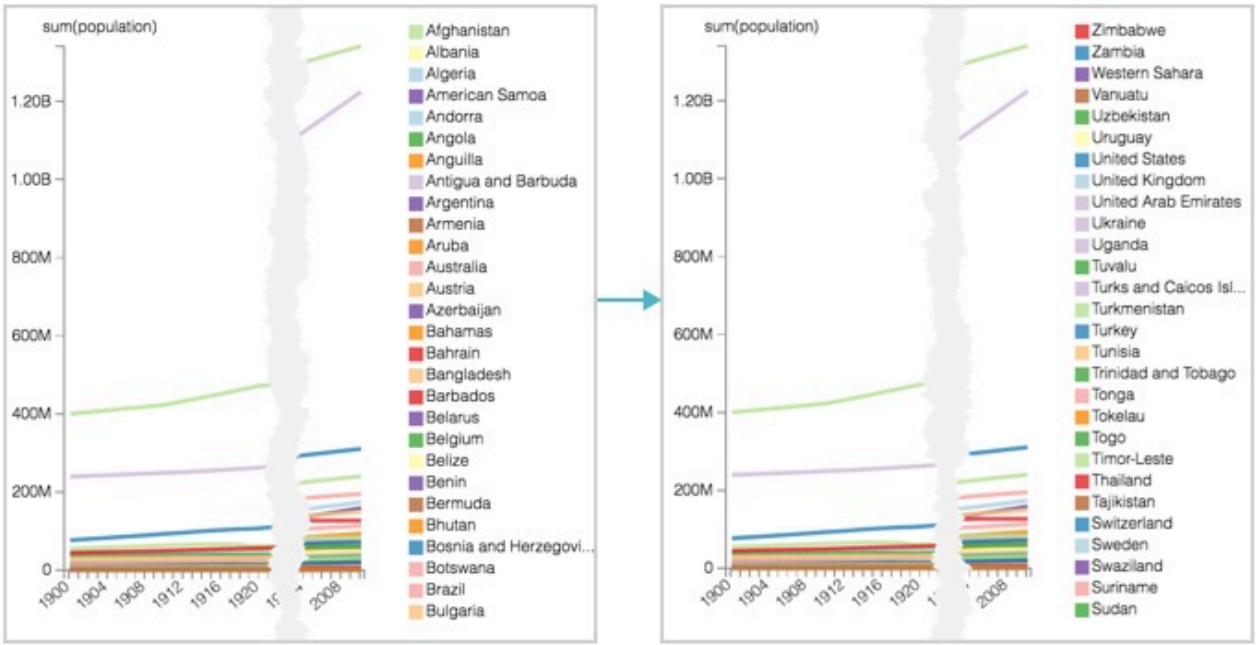

For graphs that map a continuum of values, such as choropleth maps, the legend shows a sliding scale of color and values at the minimum, median, and maximum. The reverse legend option simply changes the orientation of the legend from maximum value on top, to minimum value on top.

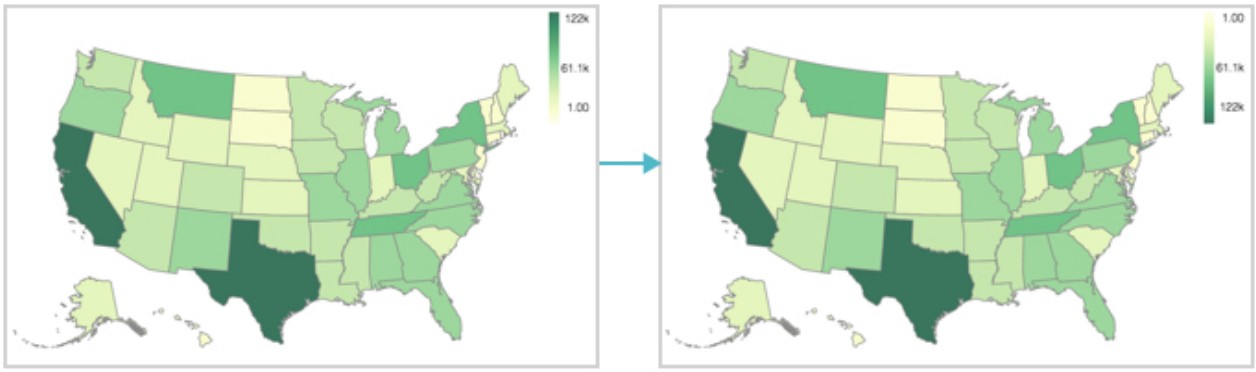

Displaying a linear trendline

Cloudera Data Visualization enables you to create a linear trendline. A linear trendline is a best-fit straight line that is used with simple linear data sets. The data is linear if the system data points resemble a line. A linear trendline usually shows that something is increasing or decreasing at a steady rate.

To display a linear trendline for the graph, navigate to the Trendline menu, and select Display linear trendline.

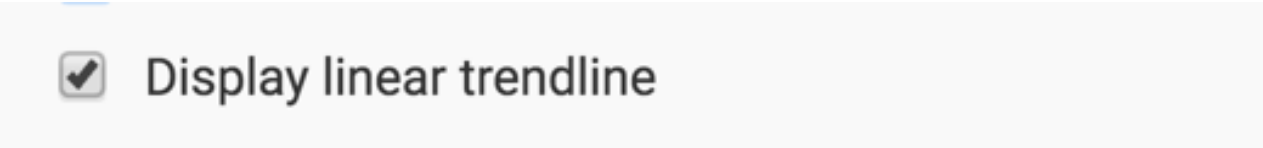

Note that the trendlines appear on the visual. To see the specific values related to the calculation of the trendline, hover your mouse over the trendline and examine the information in the Tooltip window.

Here is a line visual with trendlines. In the following example, notice that a linear trendline clearly shows that population of China has consistently risen over the years.

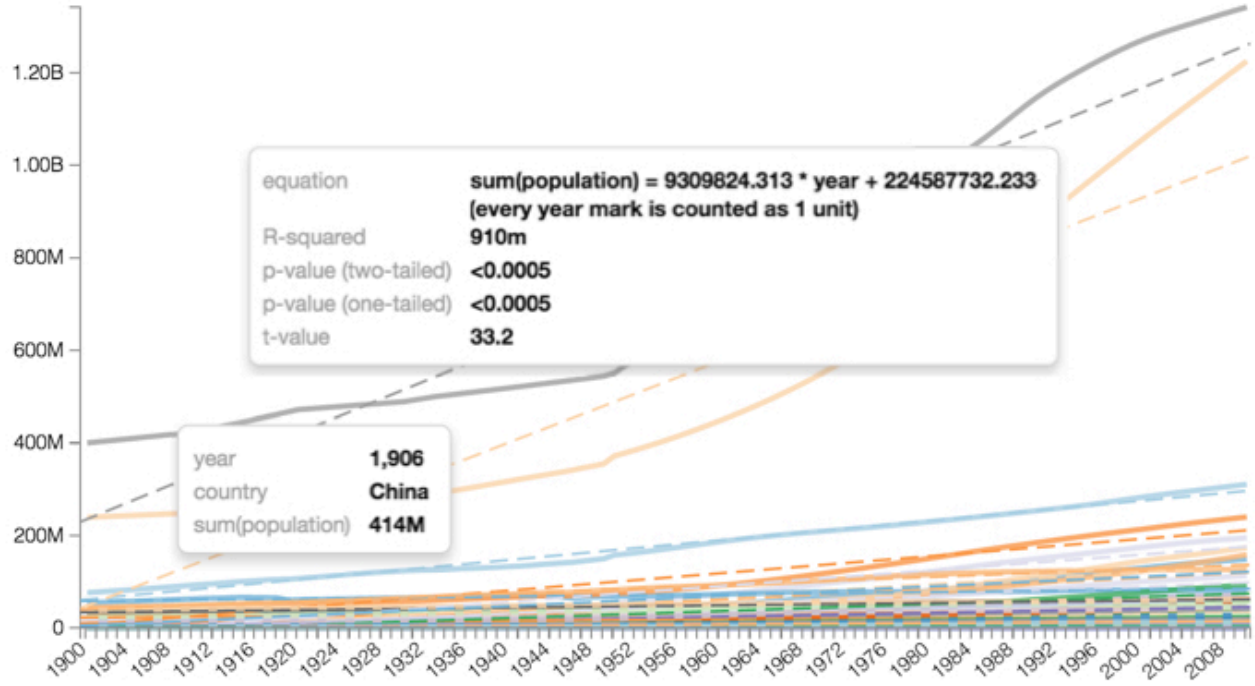

Here is a bar visual with trendlines. In the following example, notice that a linear trendline clearly shows that the population of Southern Asia has consistently risen over the years.

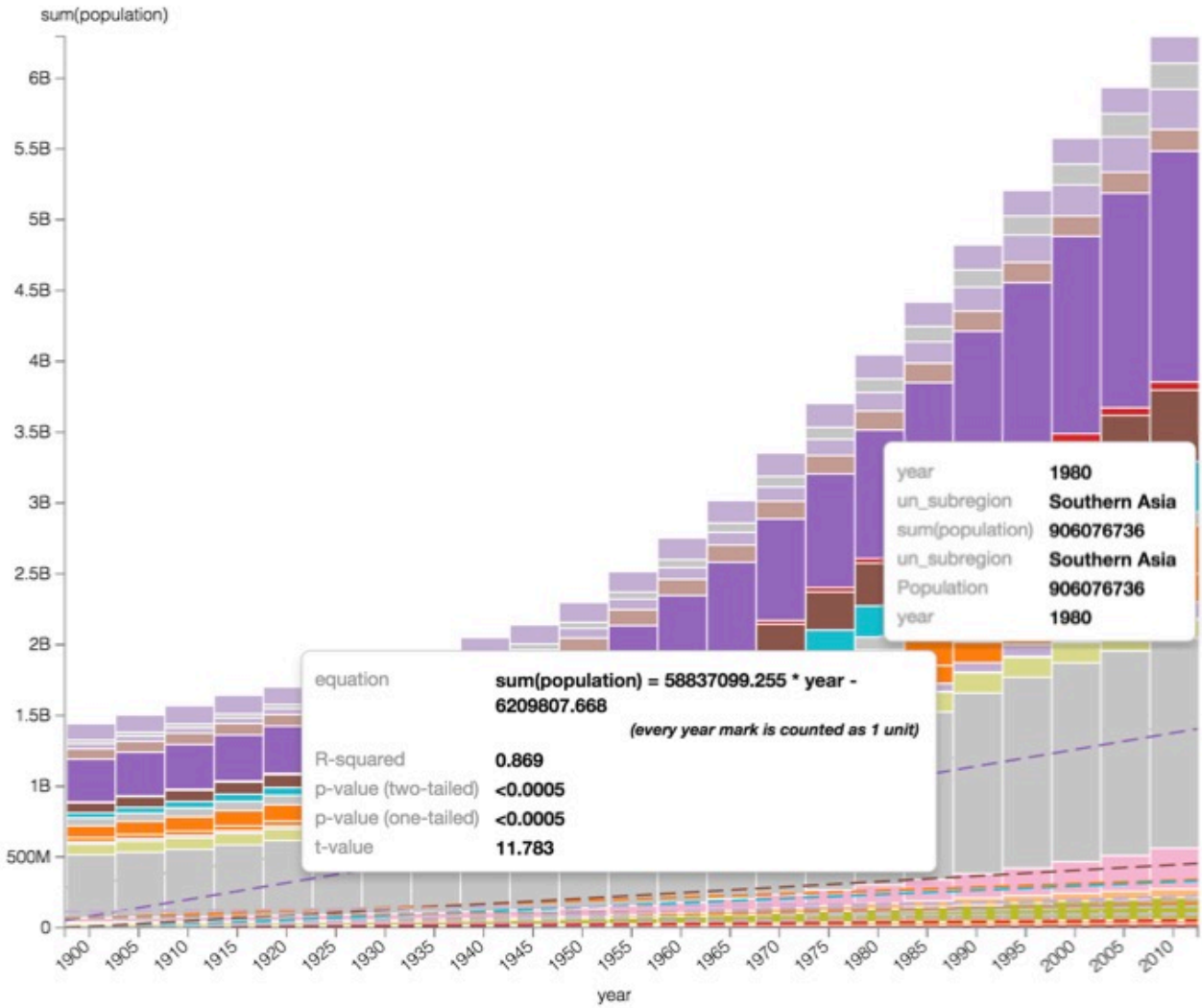

Separating trendlines by color

Cloudera Data Visualization enables you to separate all linear trendlines and plot them in the same color as main data they describe.

To separate all the linear trendlines and plot them in the same color as main data they describe, navigate to the Marks menu, and select Separate trendlines by color.

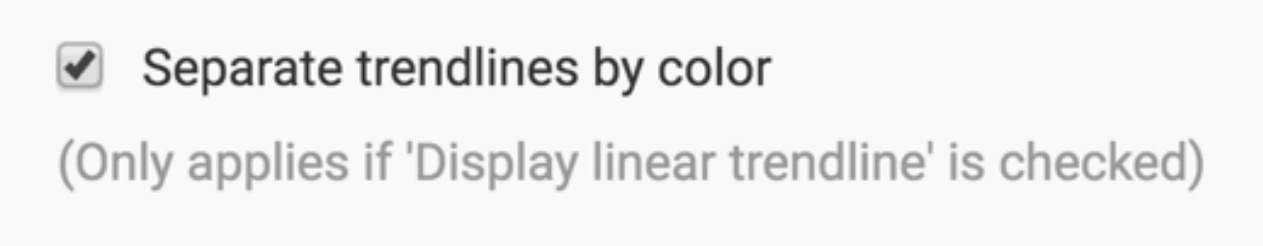

Note that the trendlines of visual match the color of the line dimension they describe.

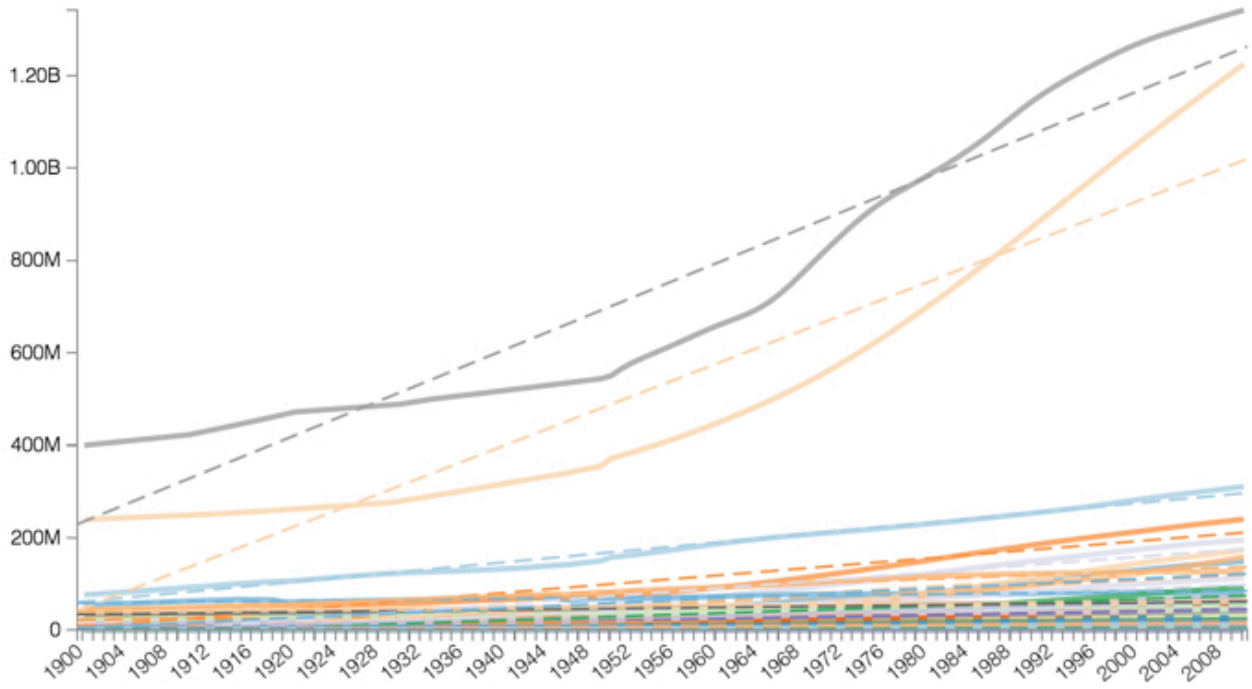

Changing trendline number of decimals

Cloudera Data Visualization enables you to change the number of decimals in the trendline equation and r-square value display.

To change the number of decimals in the trendline equation and r-square value display, navigate to the Trendlines menu, and adjust the selector for Decimals included when displaying trendline.

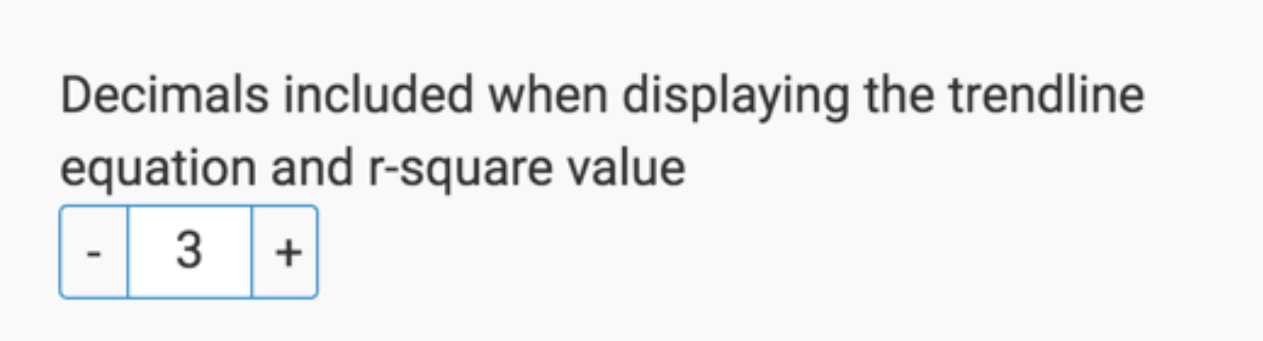

You can see the change to the tooltip of the trendline, visible when you hover the cursor over it.

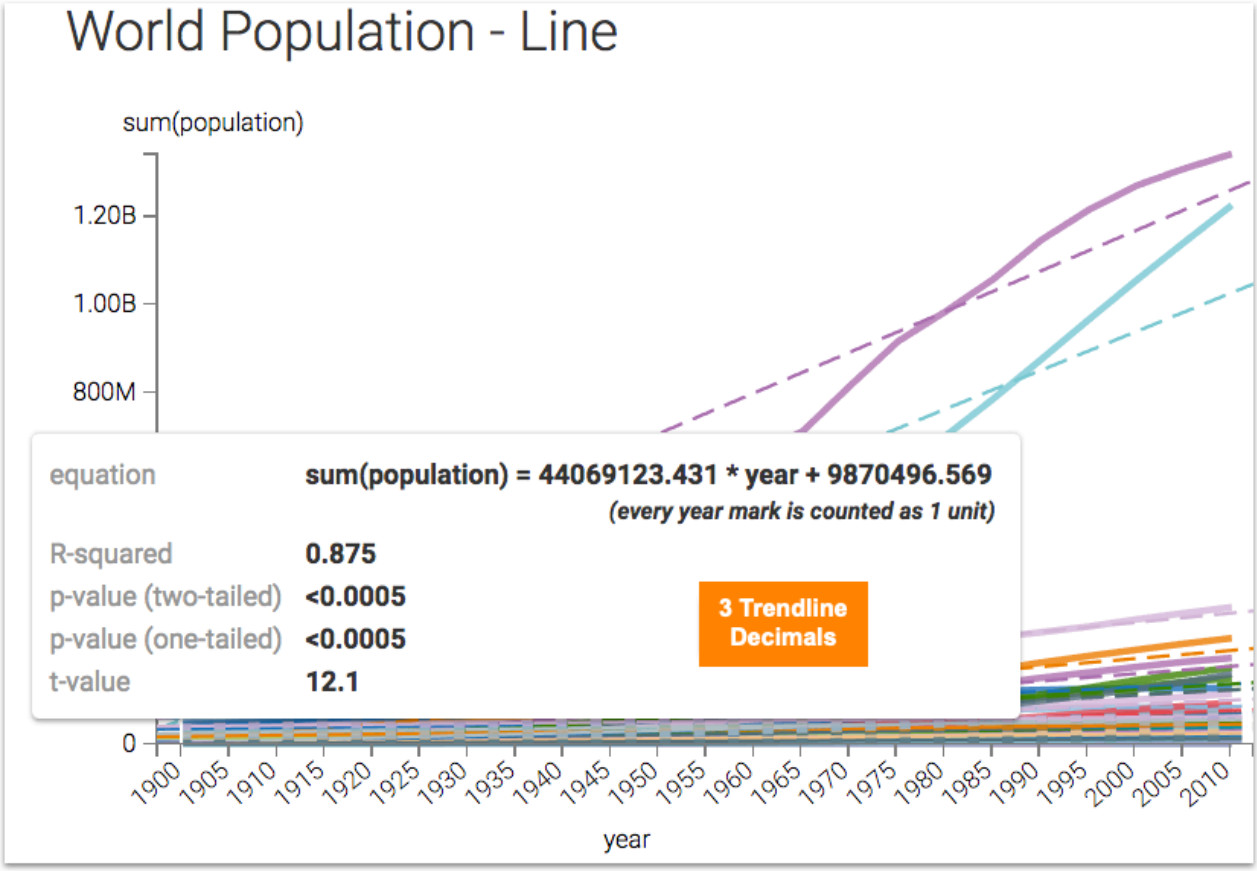
World Population - Line
sum(population)
1.20B
1.00B
800M
0
equation
sum(population) = 44069123.431 * year + 9870496.569
(every year mark is counted as 1 unit)
R-squared
0.875
p-value (two-tailed)
<0.0005
p-value (one-tailed)
<0.0005
t-value
12.1
3 Trendline Decimals
1900 1905 1910 1915 1920 1925 1930 1935 1940 1945 1950 1955 1960 1965 1970 1975 1980 1985 1990 1995 2000 2005 2010
year

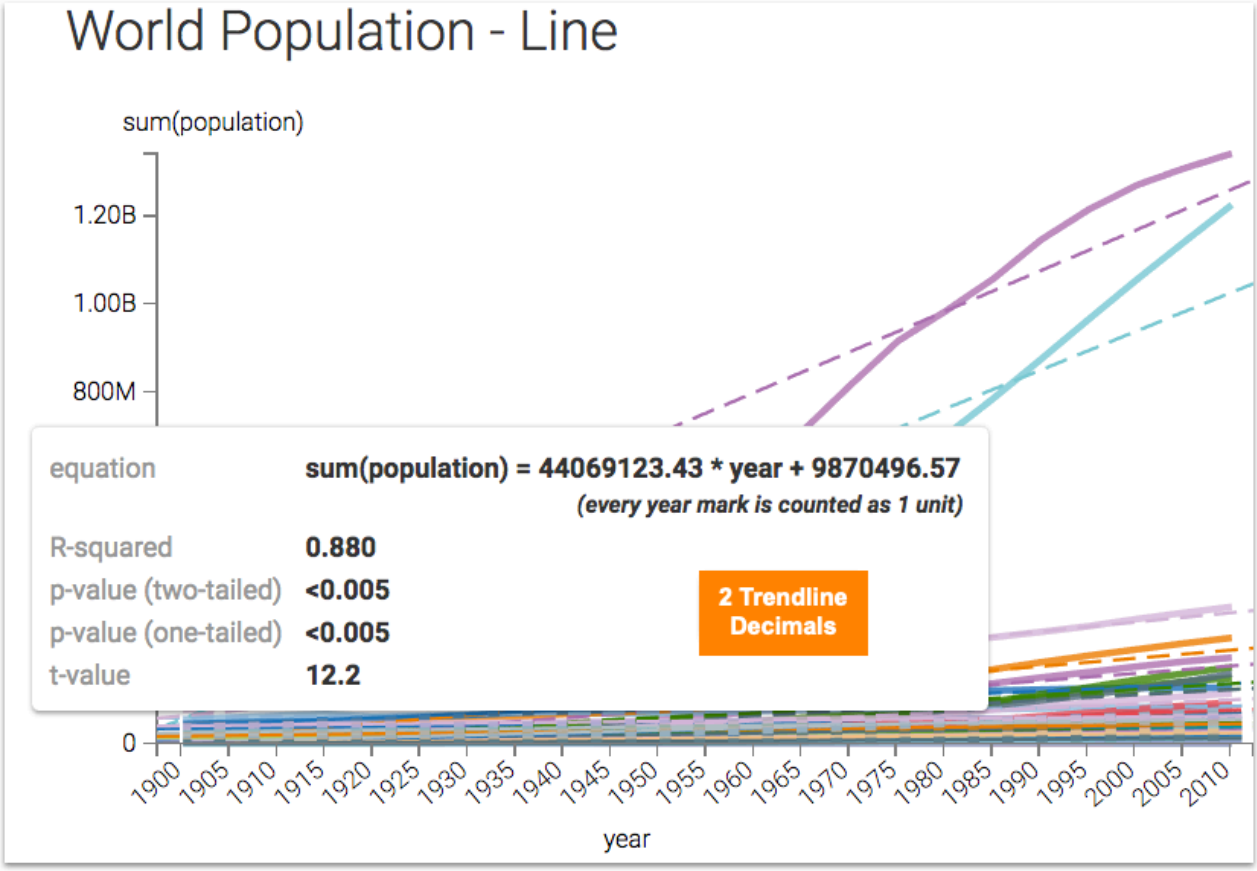
World Population - Line
sum(population)
1.20B
1.00B
800M
0
equation
sum(population) = 44069123.43 * year + 9870496.57
(every year mark is counted as 1 unit)
R-squared
0.880
p-value (two-tailed)
<0.005
p-value (one-tailed)
<0.005
t-value
12.2
2 Trendline Decimals
1900 1905 1910 1915 1920 1925 1930 1935 1940 1945 1950 1955 1960 1965 1970 1975 1980 1985 1990 1995 2000 2005 2010
year

Showing a selection brush for ranges

To enable a selection brush interface for a range of values, navigate to the Marks menu, select the Show a selection brush for a range of values option.

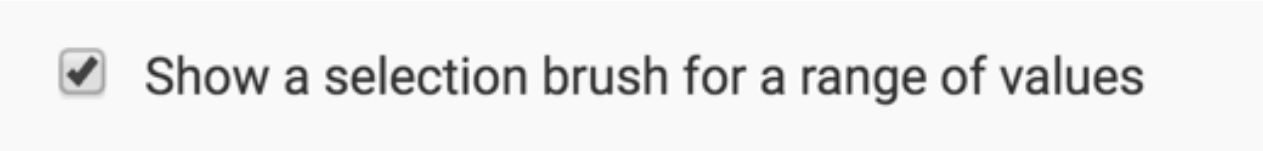

Showing measure values inside bars

If you adjust the width of the bars, Cloudera Data Visualization enables you to see measure values inside bars.

To show measure values inside bars, navigate to the Marks menu, and select the Label measure values for bars option.

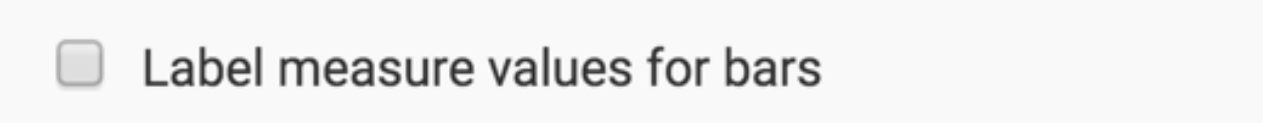

Note the measure values inside the bars in the second image.

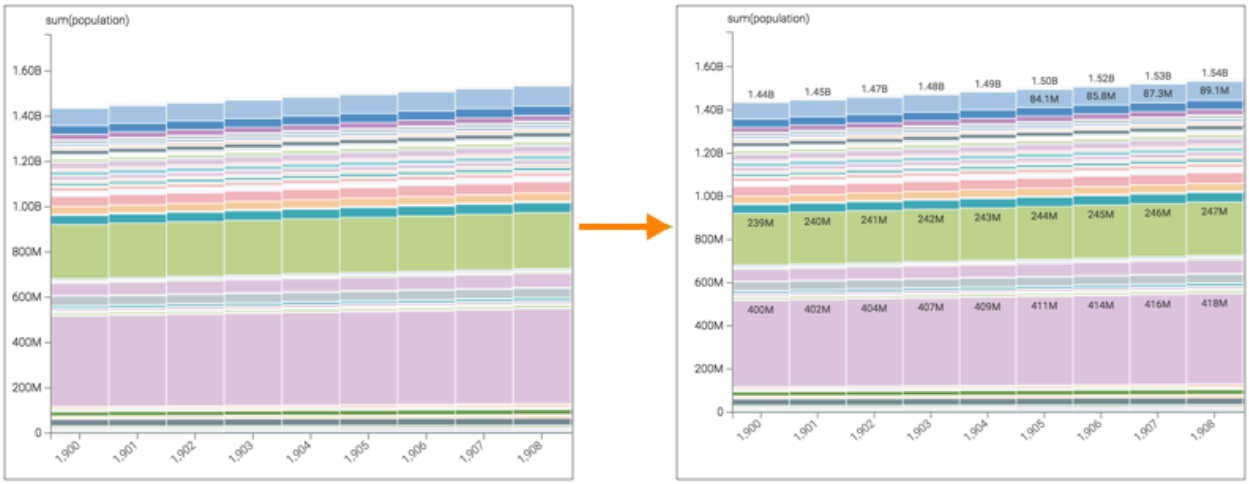

You may need to adjust the width of the bars to accommodate the value display. See *Change bar size range*.

Related Information

Changing bar size range

Changing bar size range

To change the range of widths for bars, navigate to the Marks menu, and change the values in the Bar size range option. The first number is the minimum width, and the second number is the maximum width.

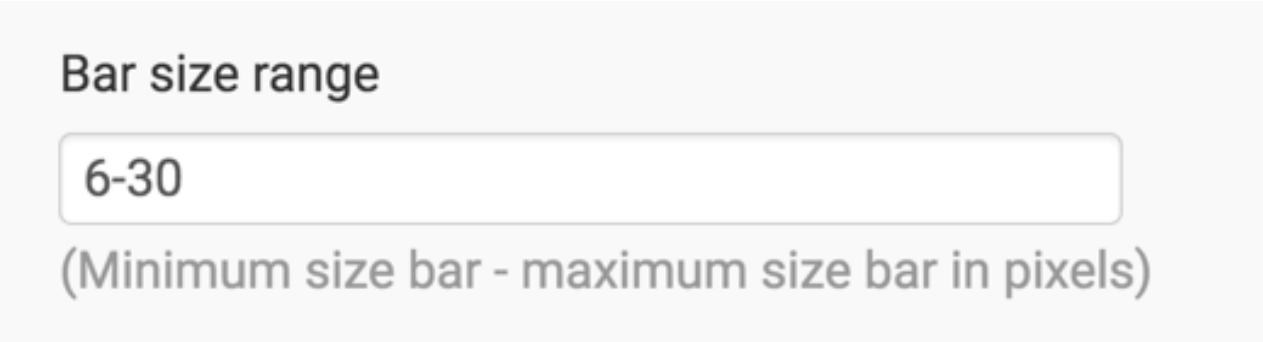

For example, compare the relative widths of the bars in the following two graphs. Note that the wider setting allows enough space to label each bar element.

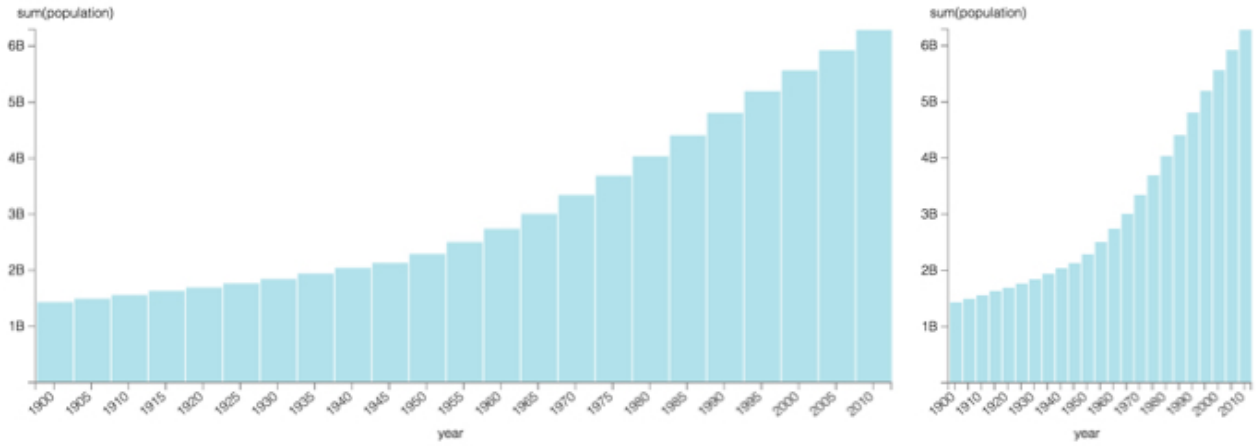

Separating histogram columns

About this task

Cloudera Data Visualization allows you to add outlines around histogram bars, making each bar more distinct.

To enable this feature, go to Settings Marks and select the Separate columns option.

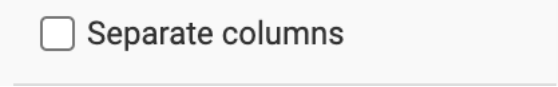

Showing dimension values on hovering

Cloudera Data Visualization enables you to see dimension values when hovering over a dimension point.

To show dimension values when you hover over a dimension point, navigate to the Marks menu, and select the When hovering over a dimension, show each point with that dimensional value option.

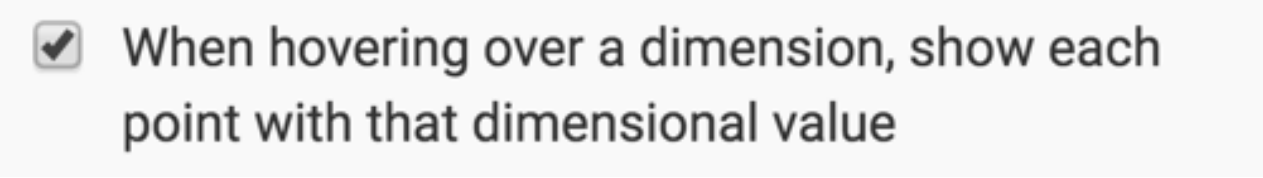

In the following image, you can see the display of the value of the dimension, when you hovered over year 1933.

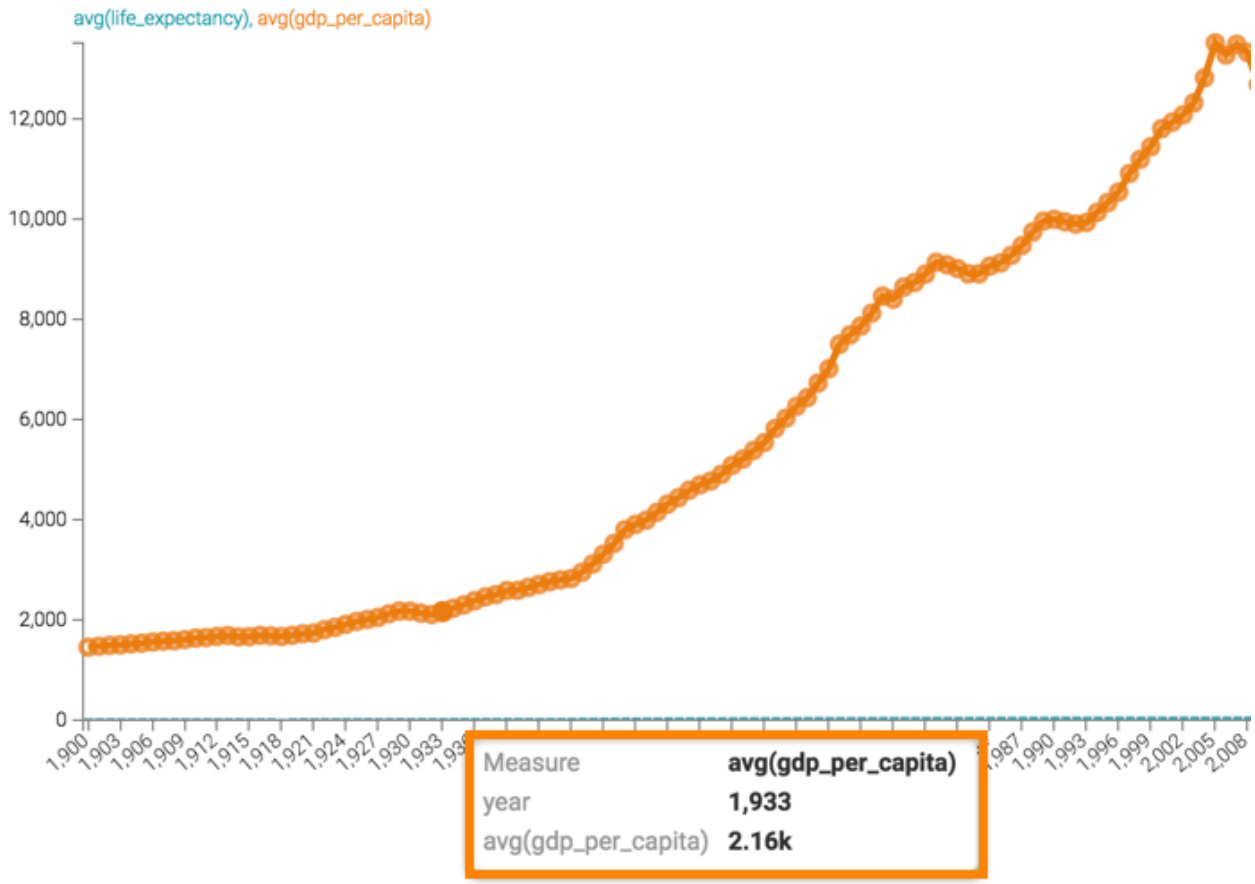

Showing points

This task applies to Line charts, Area charts, and Combined Bar/Line charts.

To show the individual data points, navigate to the Marks menu, and select the Show Points option.

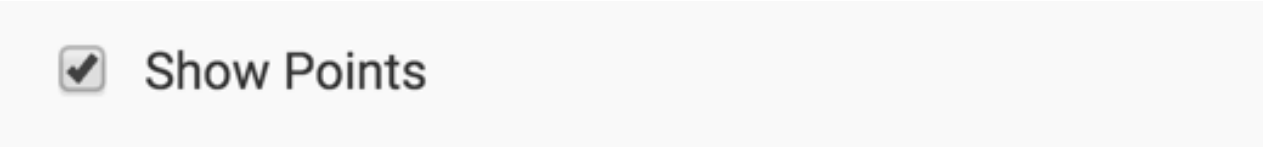

Note how the points appear on a visual.

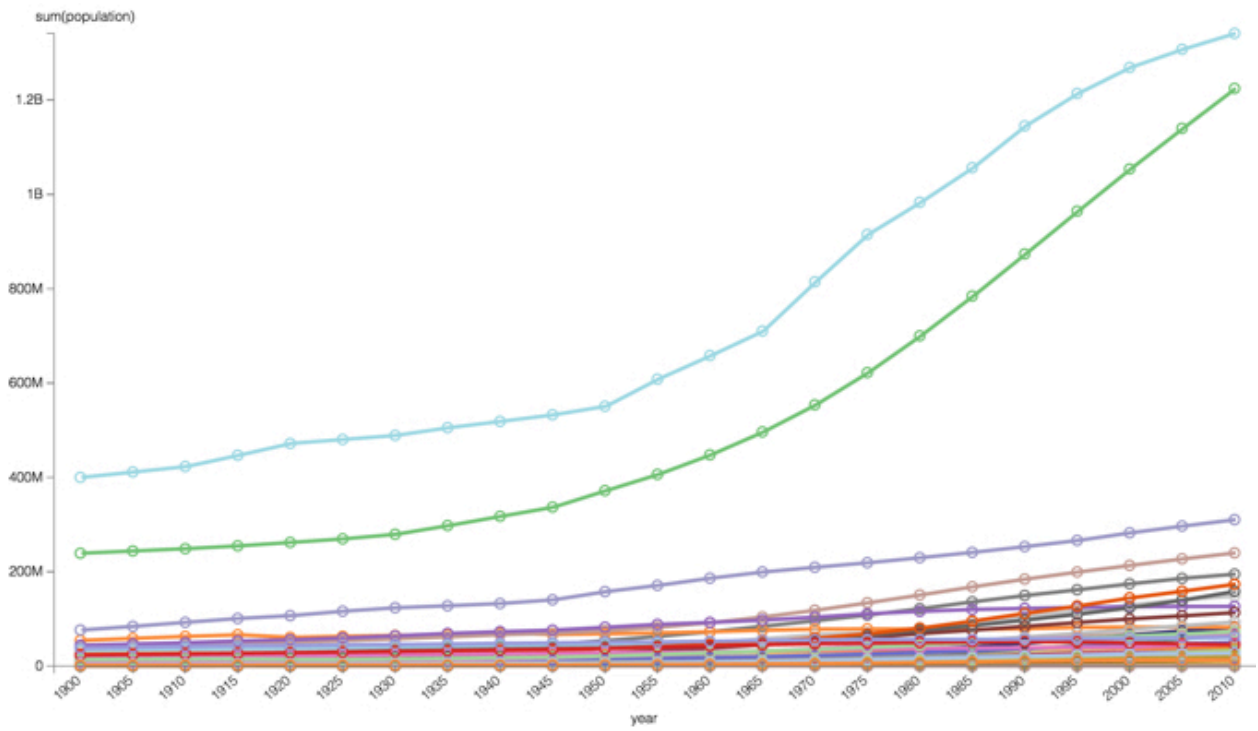

Ignoring missing date values

When date/time information is missing from a dataset, the corresponding values plot as zeros on line and area graphs. It is possible to "smooth" the data by interpolating the missing values.

To ignore missing dates in a plot, navigate to the Marks menu, and select the Ignore missing dates option.

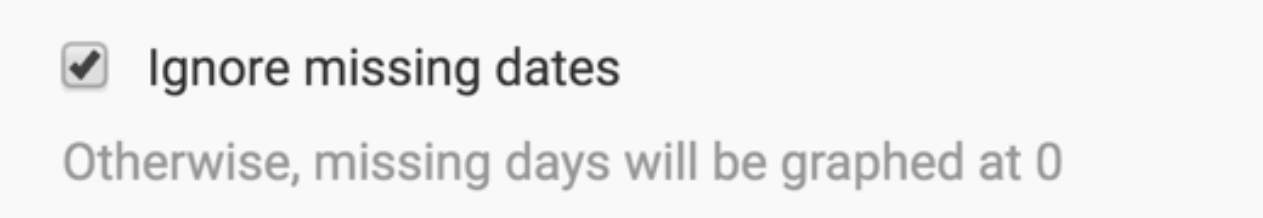

Note that some of the spikes are 'smoothed' in the second image.

Changing line shape

To change the style of a line (linear by default), navigate to the Marks menu, and select one of the options from the Line Style menu:

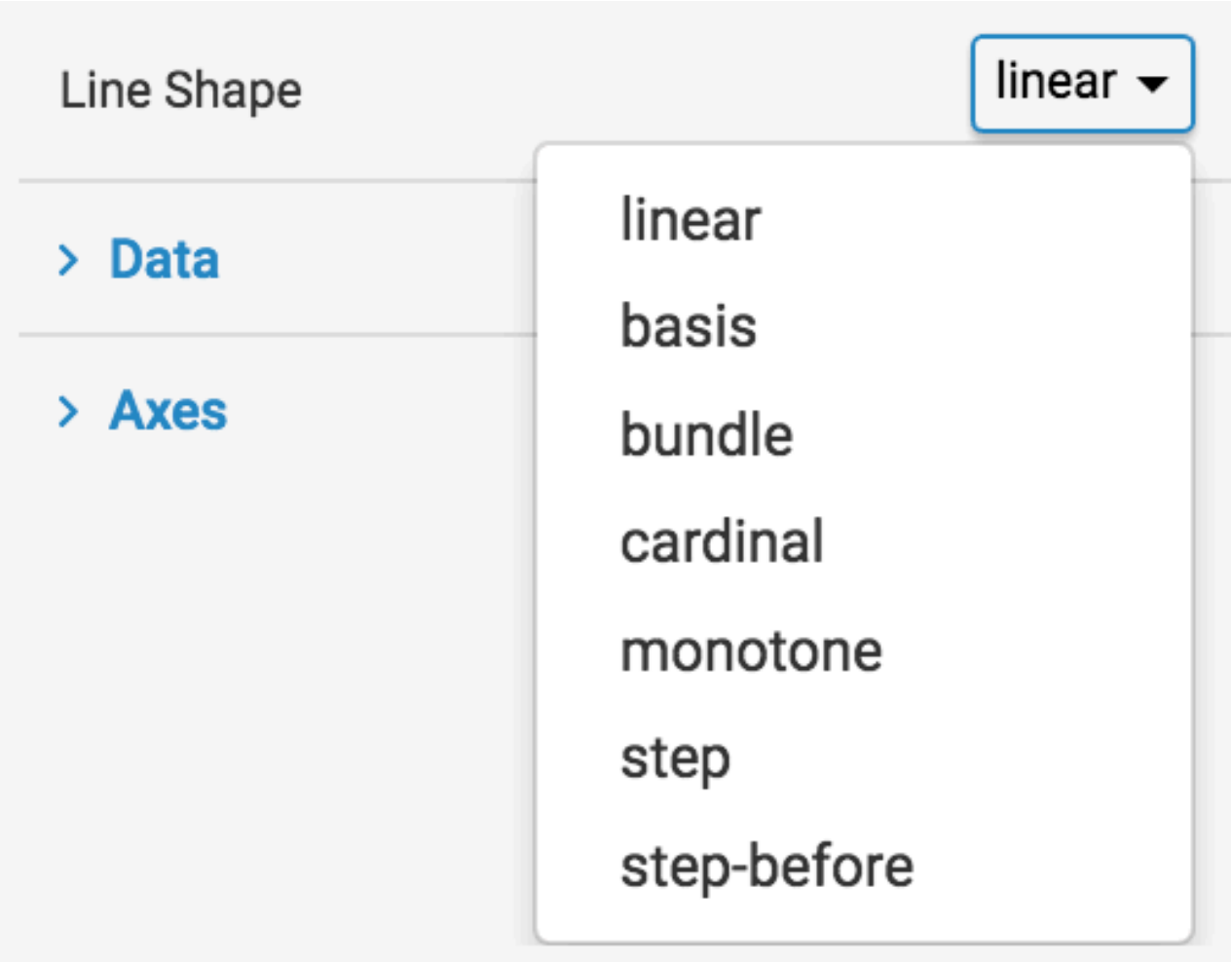

Linear Line Shape

The linear line shape, the default setting, is a normal jagged line comprised of piecewise line segments that connect the data points.

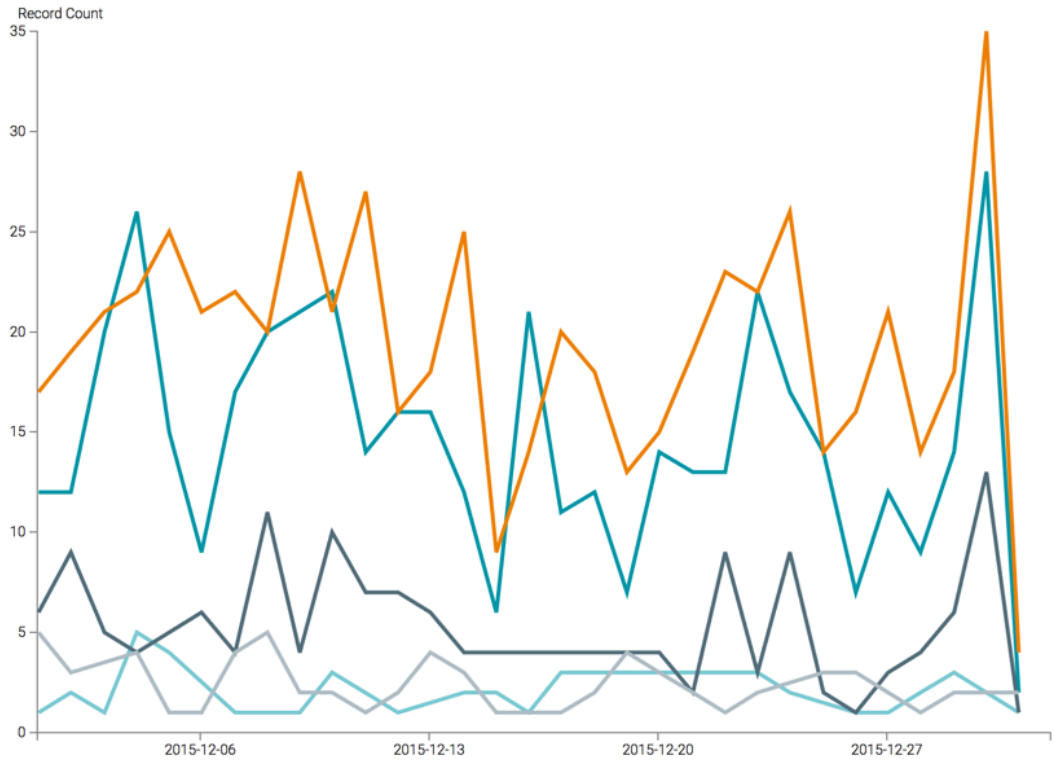

Basis Line Shape

The basis line shape implements curve-fitting using a B-spline interpolation formulas (generalization of the Bézier curve), with control point duplication on the ends. Contrast this rendering with the one in Bundle Line Shape.

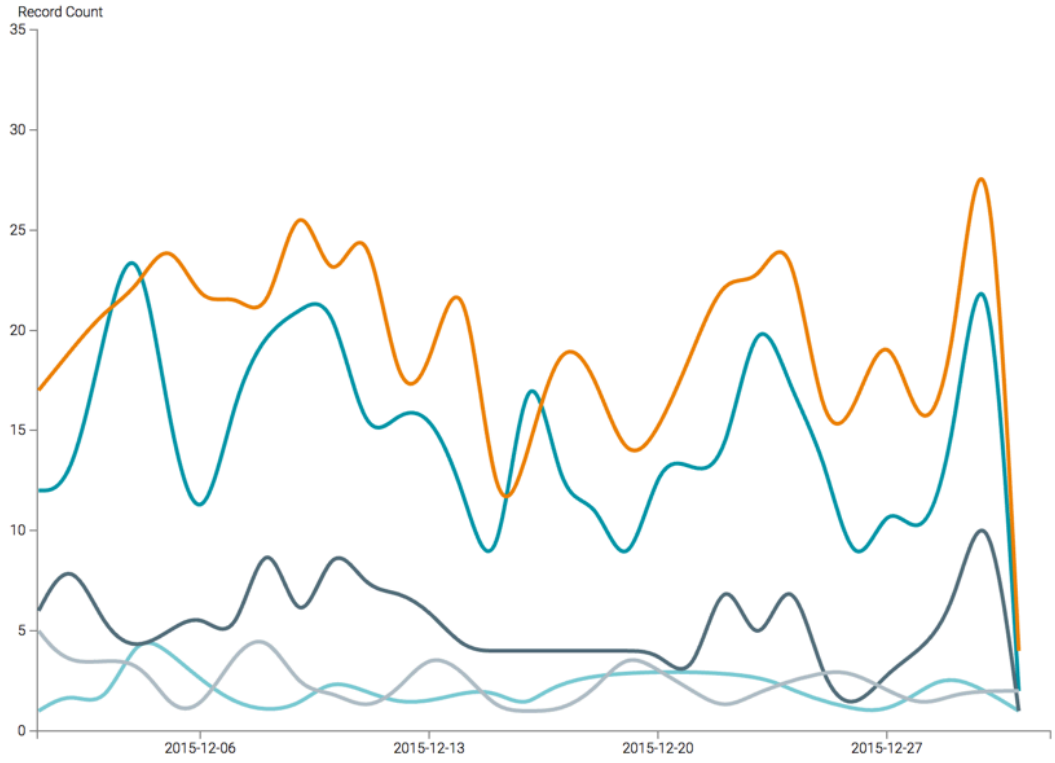

Bundle Line Shape

The bundle line shape implements curve-fitting using a B-spline interpolation formulas (generalization of the Bézier curve), with the tension parameter to straighten the spline. Contrast this rendering with the one in Basis Line Shape.

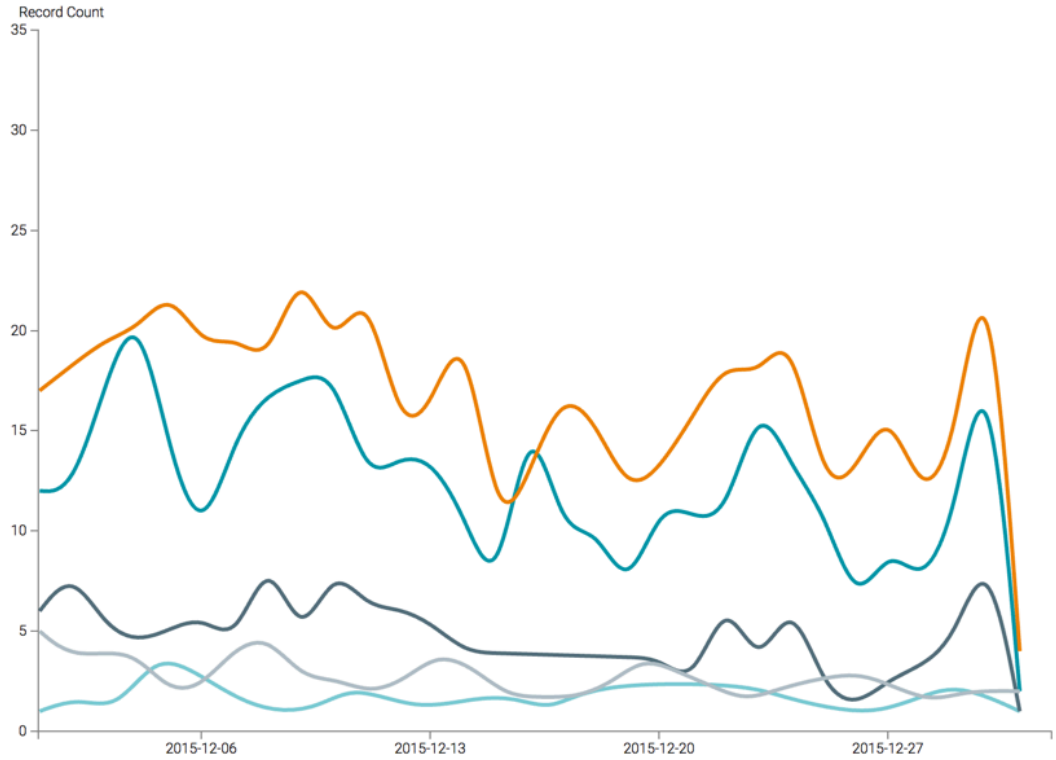

Cardinal Line Shape

The cardinal line shape implements curve-fitting using a Cardinal spline (or canonical spline) interpolation formulas that incorporate a tension parameter, with control point duplication on the ends. Contrast this rendering with the one in Basis Line Shape.

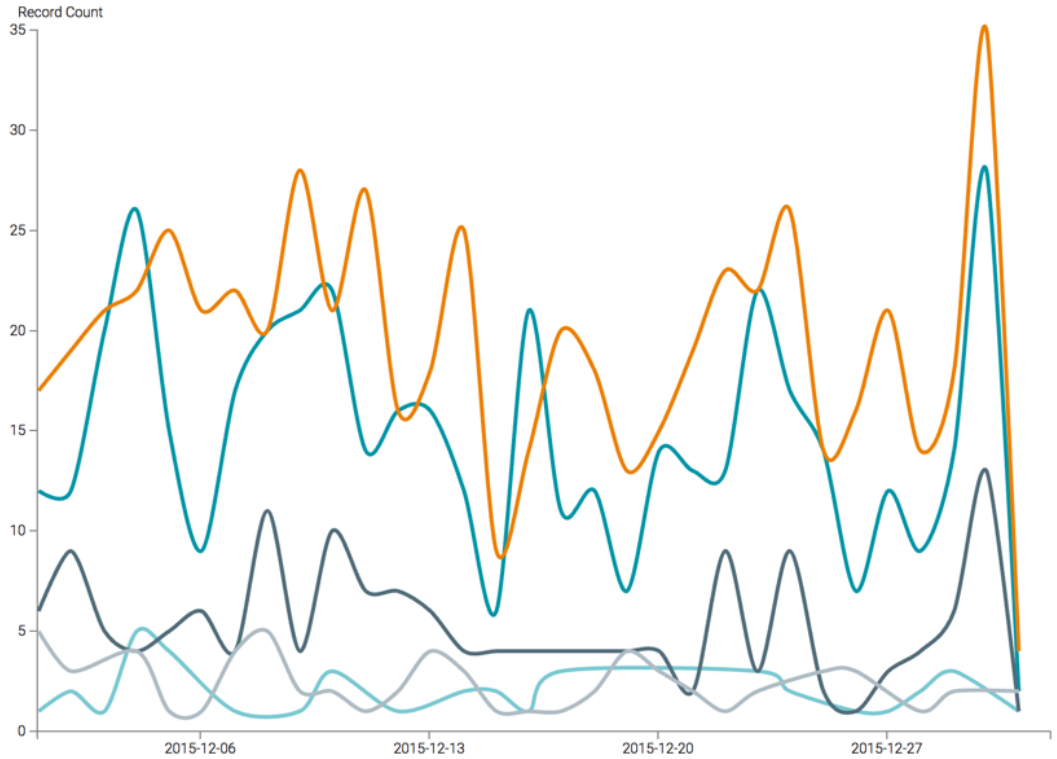

Monotone Line Shape

The monotone line shape is a cubic interpolation that preserves the monotonicity on the vertical axis as it calculates the curve fit.

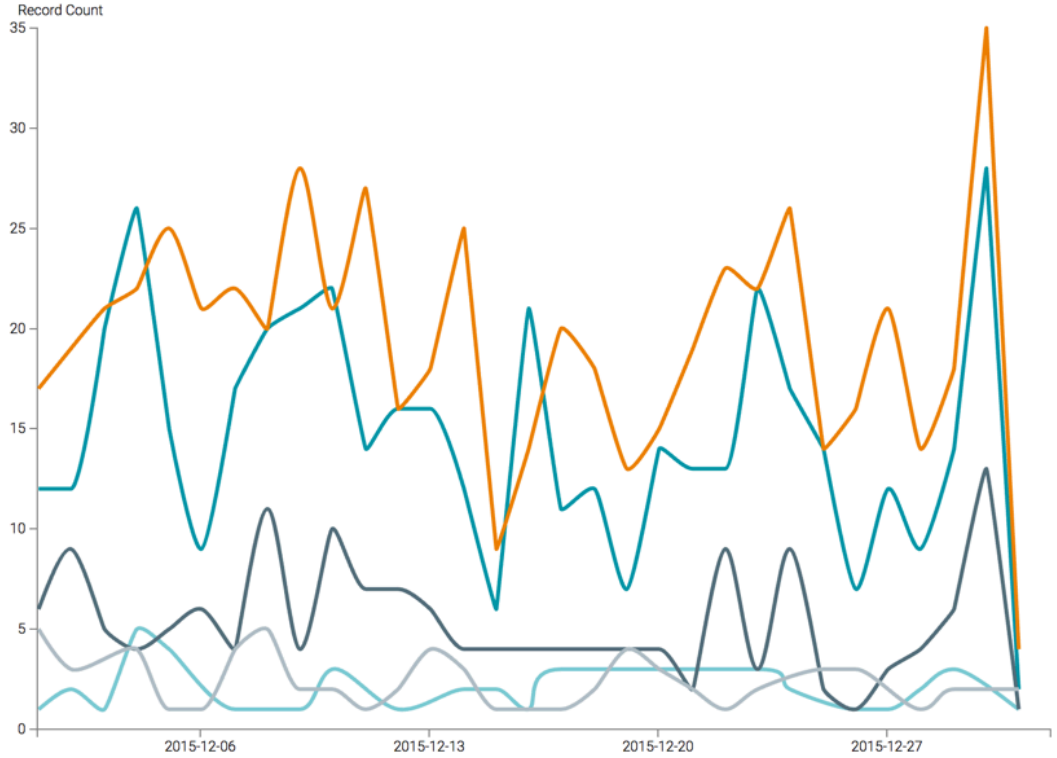

Step Line Shape

The step line shape, just as the name implies, alternates between horizontal and vertical segments, starting with a horizontal segment. Contrast this rendering with the one in Step Before Line Shape.

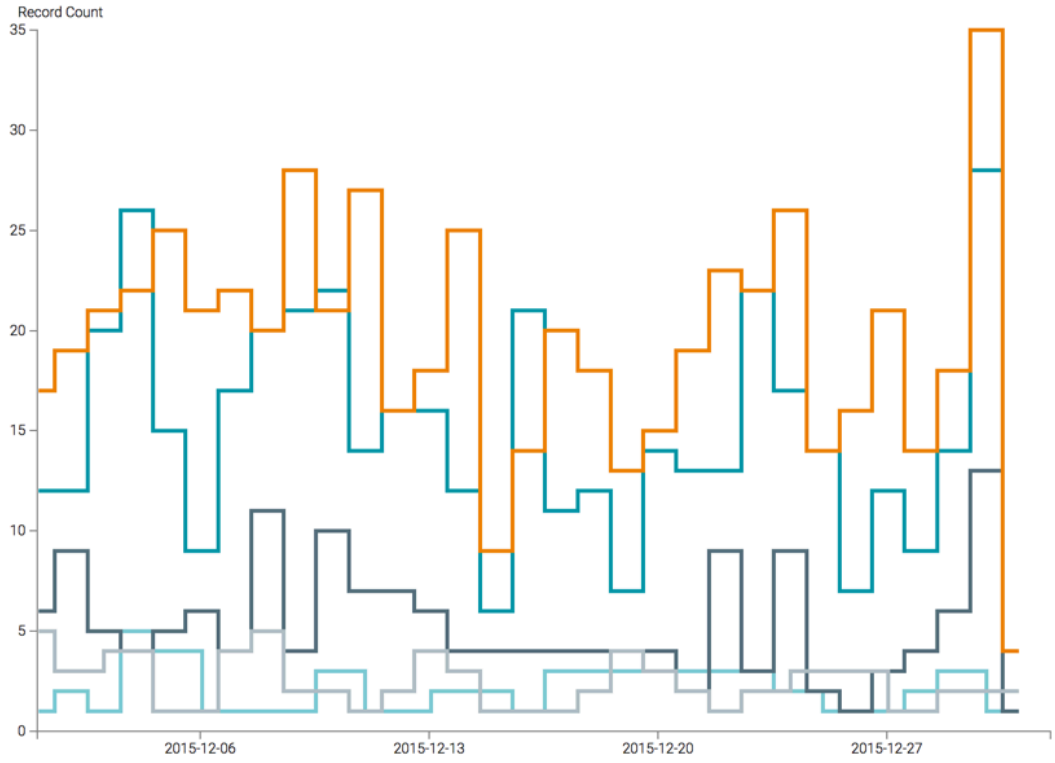

Step Before Line Shape

The step before line shape, just as the name implies, alternates between vertical and horizontal segments, starting with a vertical segment. Contrast this rendering with the one in Step Line Shape.

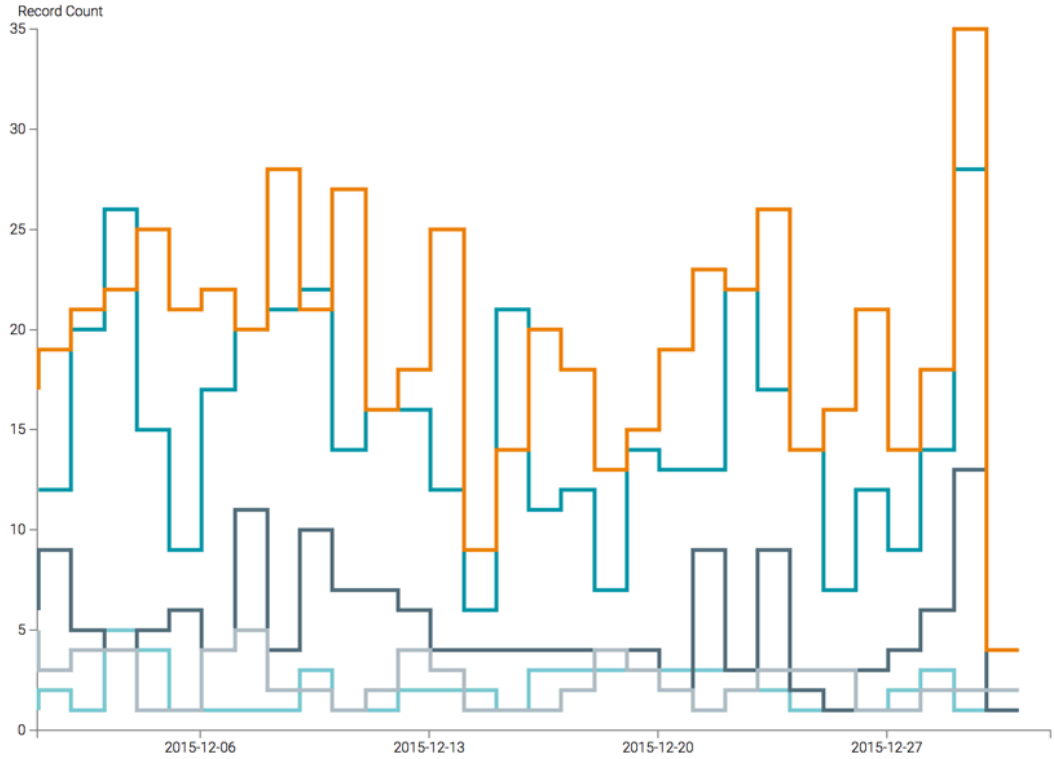

Changing line style

In a line visual, you can display a different line style for each measure in the visual.

To modify each line of a visual to a different style, navigate to the Marks menu, and check the Modify line styles per color option.

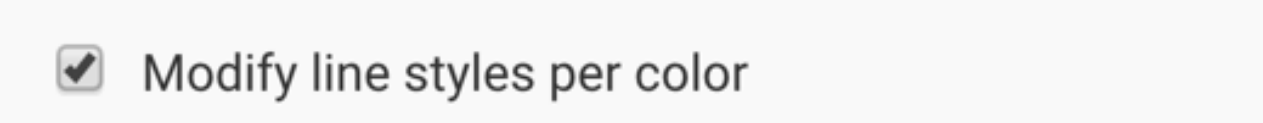

In the following image, notice the different line styles for each measure:

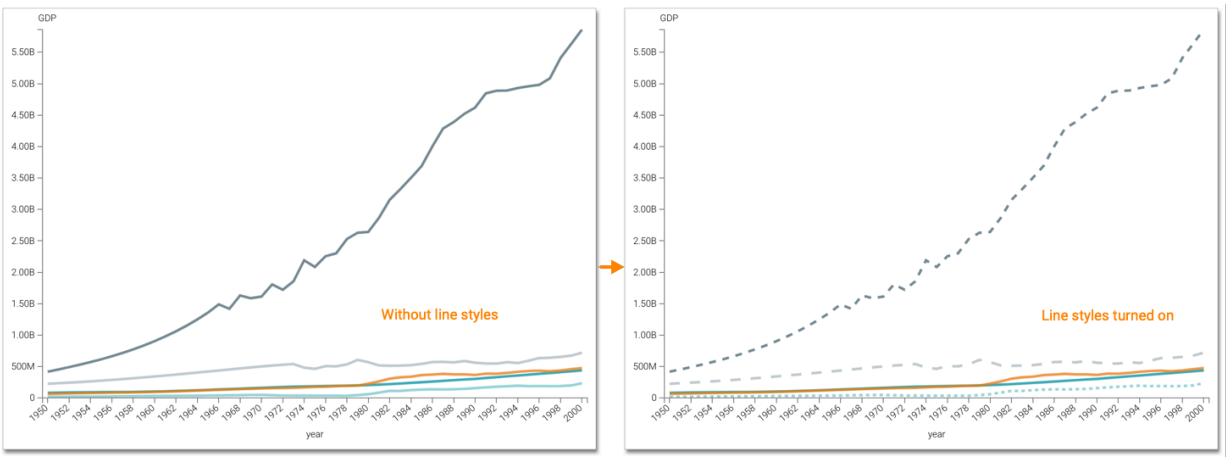

If you hover over a line, the line style changes from a dotted (or dashed) line to a solid line, and the tooltip appears.

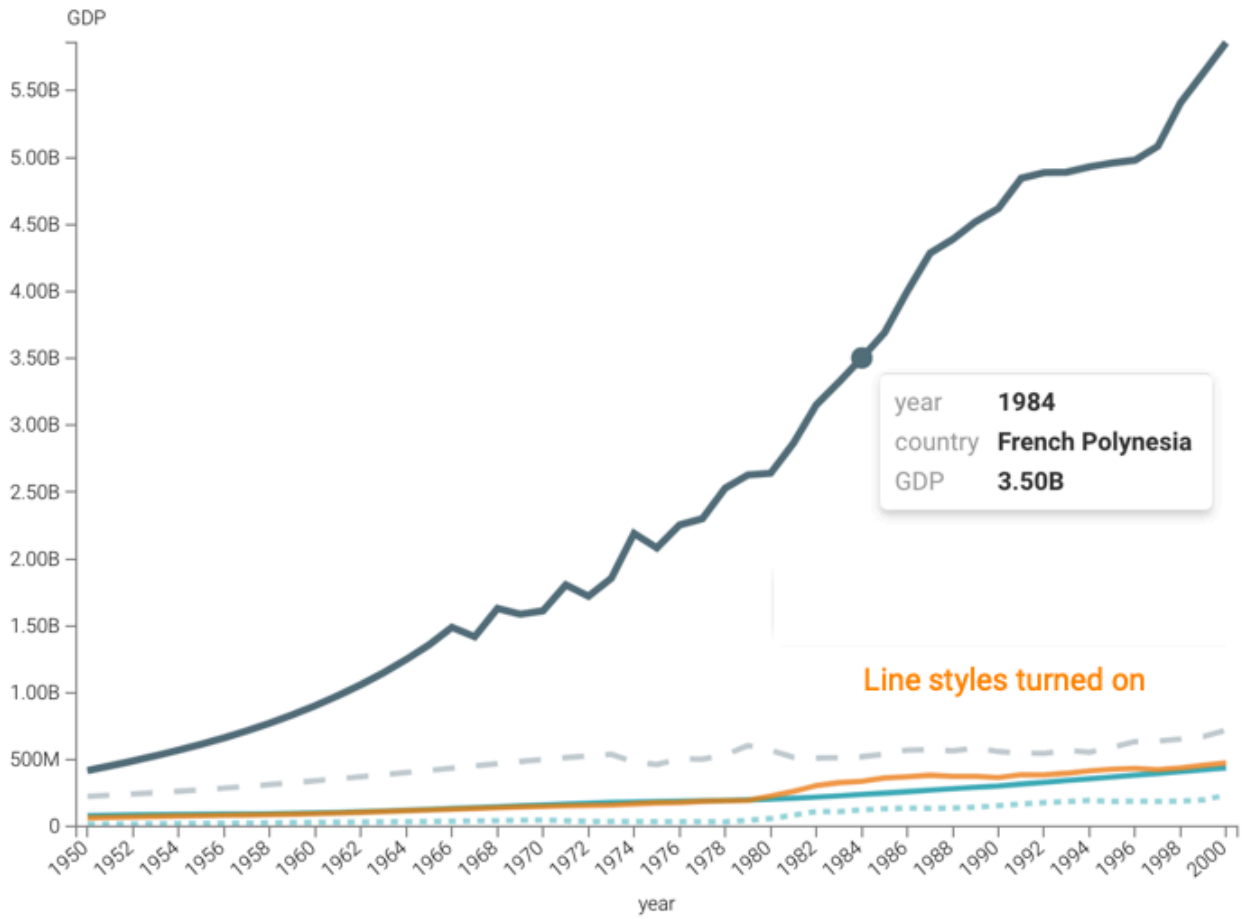

Changing highlight type

To change the style of a highlight (circle by default), navigate to the Marks menu, and select one of the options from the Highlight Mark Type menu:

- circle (default)
- dot
- star

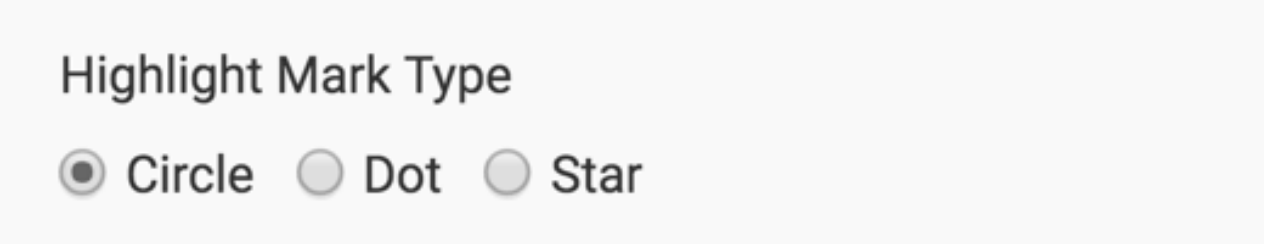

Notice how changing the highlight from a circle to a star changes the visual.

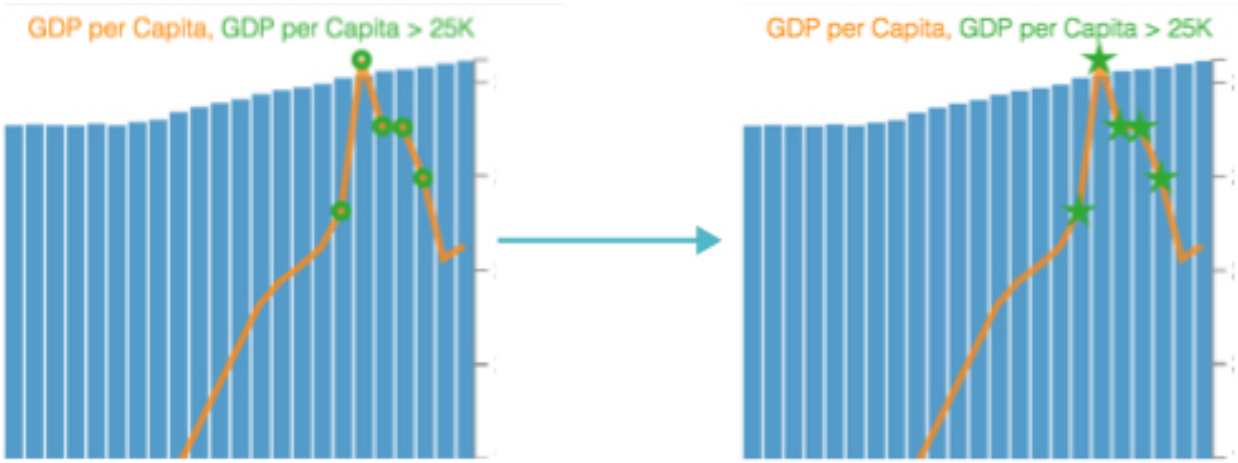

Converting line to area visual

In Combo visuals, for better representation of data, Cloudera Data Visualization enables you to convert the line to a semi-transparent area.

It applies to Combined Combo charts.

To change the combo visual style for the second dimension display from line (default) to an area, navigate to the Marks menu, and select the Convert Line To Area Visual option.

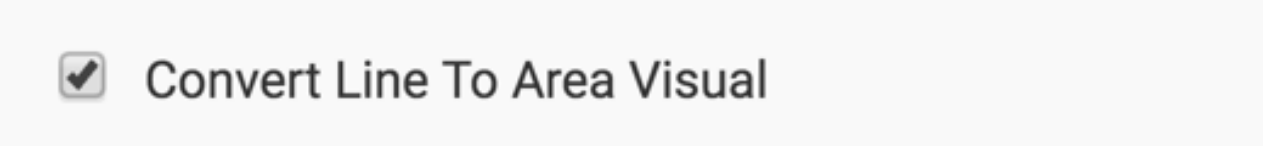

Notice how this change alters the combo visual.

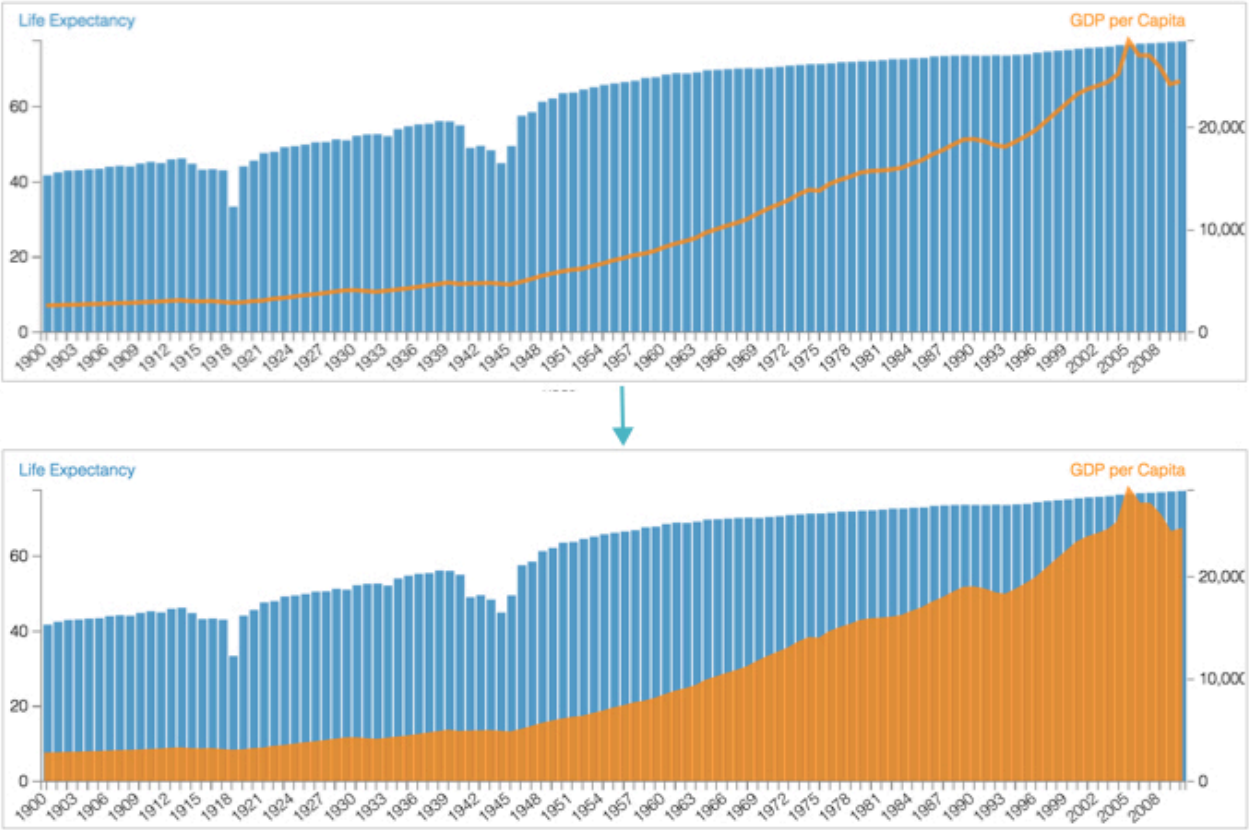

Changing highlight of radius

For clearer appearance of the highlight in a visual, Cloudera Data Visualization enables you to adjust the highlight of the radius.

It applies to Combined Bar/Line charts.

To change the size of a highlight, navigate to the Marks menu, and adjust the selector for the Max Highlight Radius option.

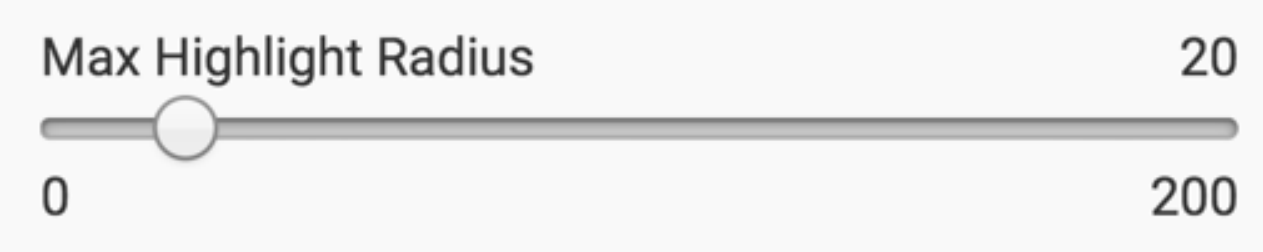

Note how the change of the maximum radius from 20 to 40 changes the appearance of the visual.

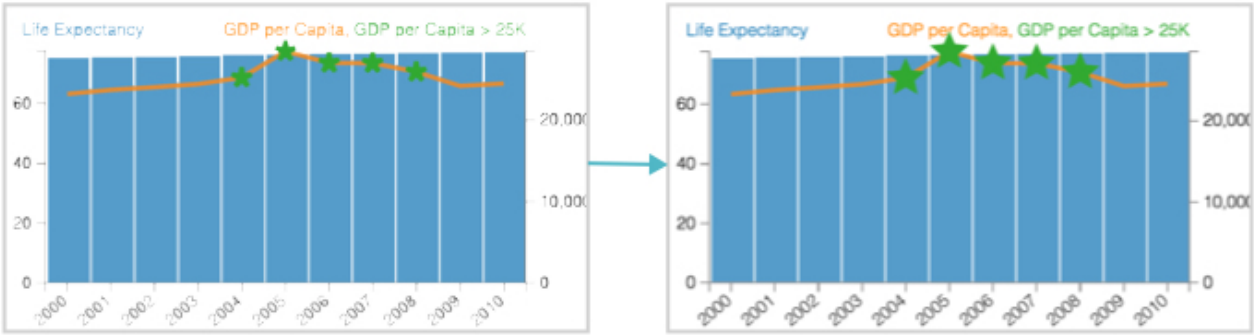

Changing mark size range

Use the Mark size range option to adjust the size of circles or bubbles on scatter and map visuals.

This setting applies to Map and Scatter visuals.

To adjust the relative size of marks on the visual, such as circles in scatters or bubbles in maps, navigate to the Marks menu, and adjust the minimum and maximum numbers for the Mark size range option.

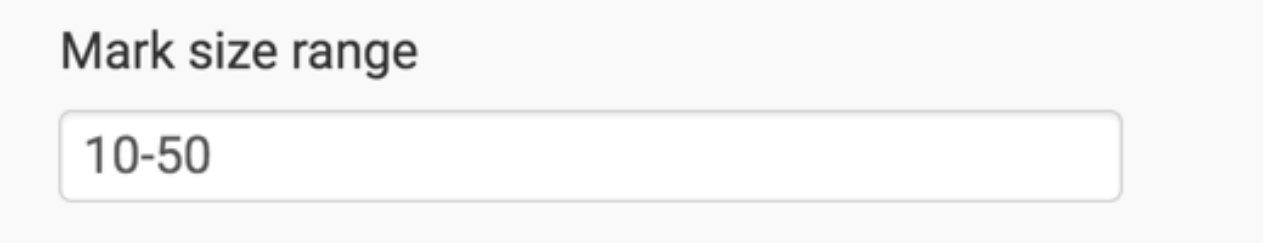

Mark Size Range in Scatter Visuals

Compare the appearance of a scatter plot that has marks at size 10-50 (the default) to the one at size 10-20 bubbles.

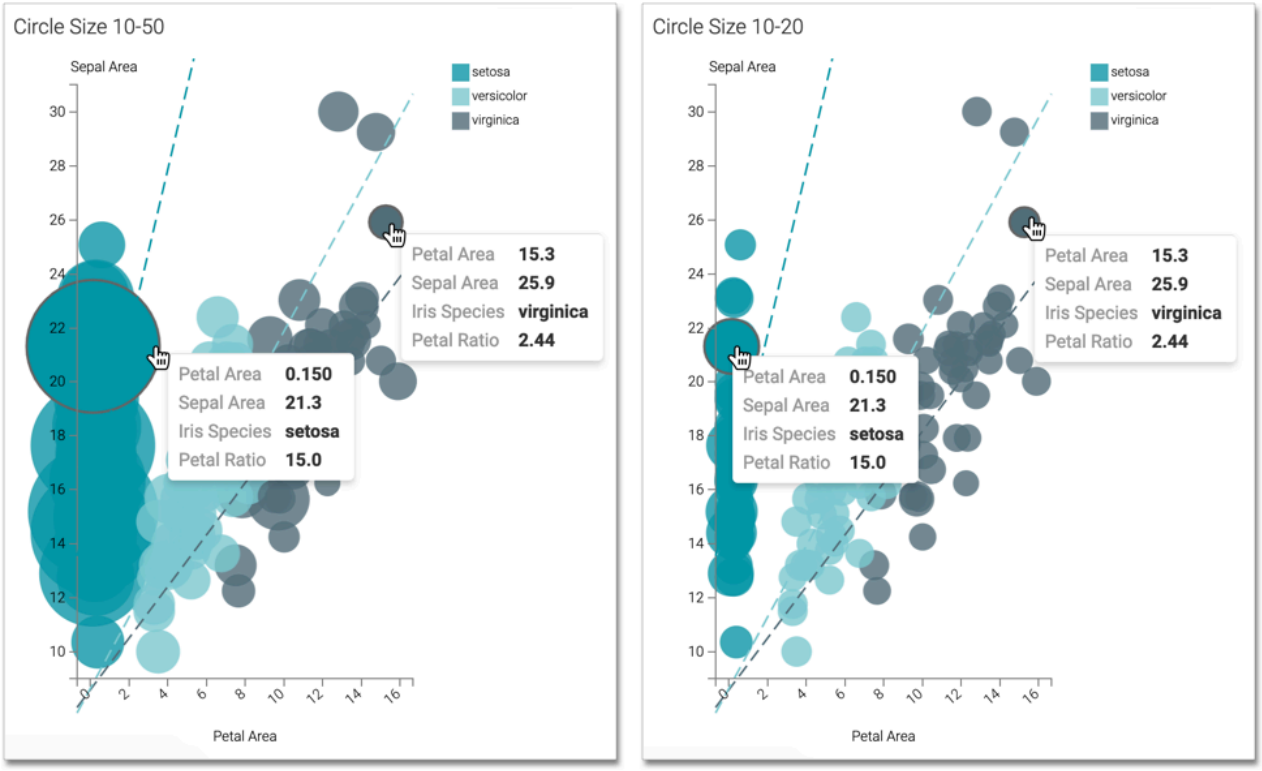

Mark Size Range in Map Visuals

Compare the appearance of world map visual with marks at size 10-50 (the default) to the one at size 1-30 bubbles.

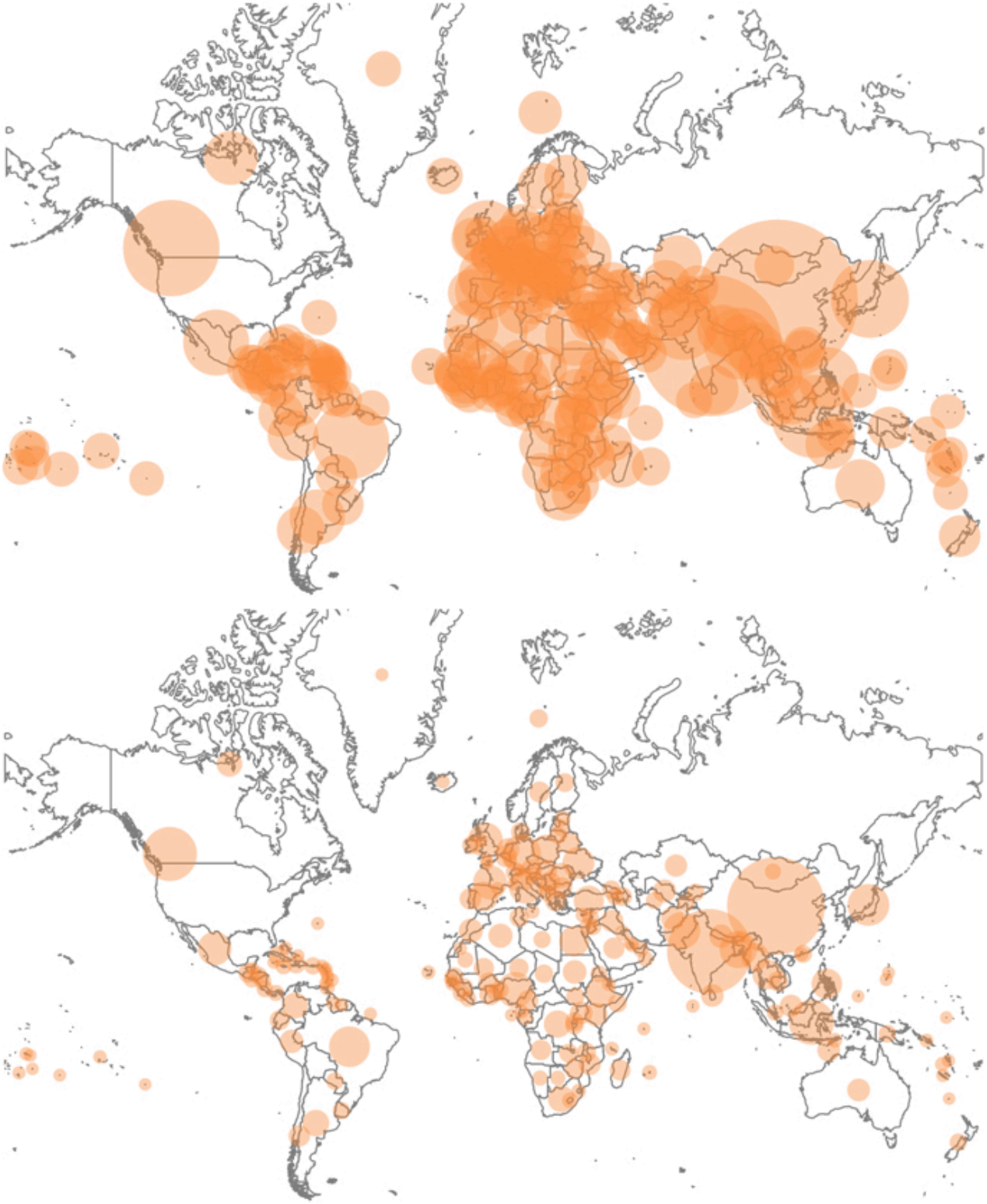

Changing transition delay

This task applies to Scatter visuals. It controls the speed of animation described in *Showing Autoplay Transition Animation*.

To adjust the speed of animation, navigate to the Marks menu, and adjust the value for the Transition Delay (ms) option.

Larger numbers specify longer delays, while smaller numbers generate faster animations. The default setting is 1,000 milliseconds; to speed up the animation 10-fold, change the value to 100.

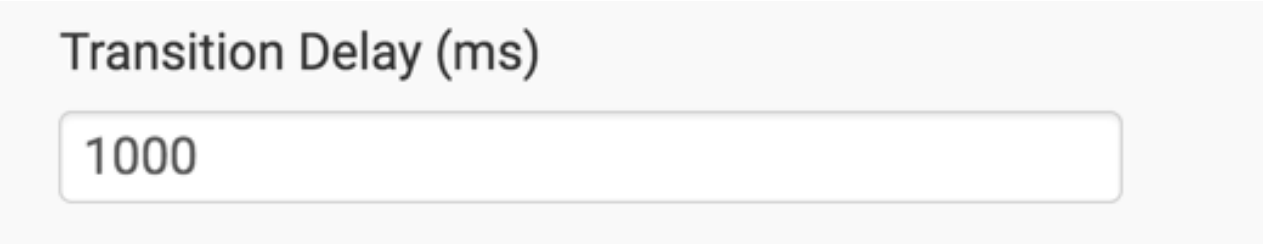

Showing autoplay transition animation

Placing a periodic dimension on the Transition shelf of a scatter visual enables you to animate a scatter visual. This is very useful when dimensions represent dates and times. This task applies to Scatter visuals.

To turn on this feature, navigate to the Marks menu, select the Autoplay transition animation option.

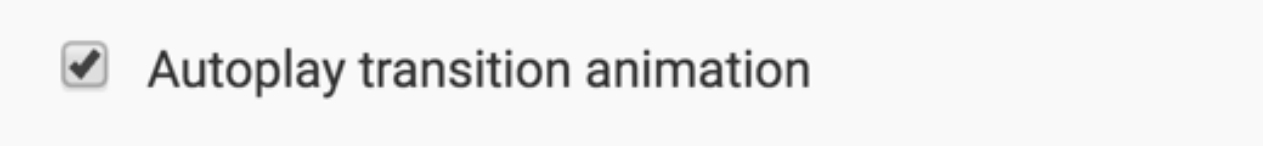

Note that appearance of controls common to all video animations: 'play' (which changes to 'pause' when the animation is active), 'back to start', 'back-up', 'advance', and 'to end'.

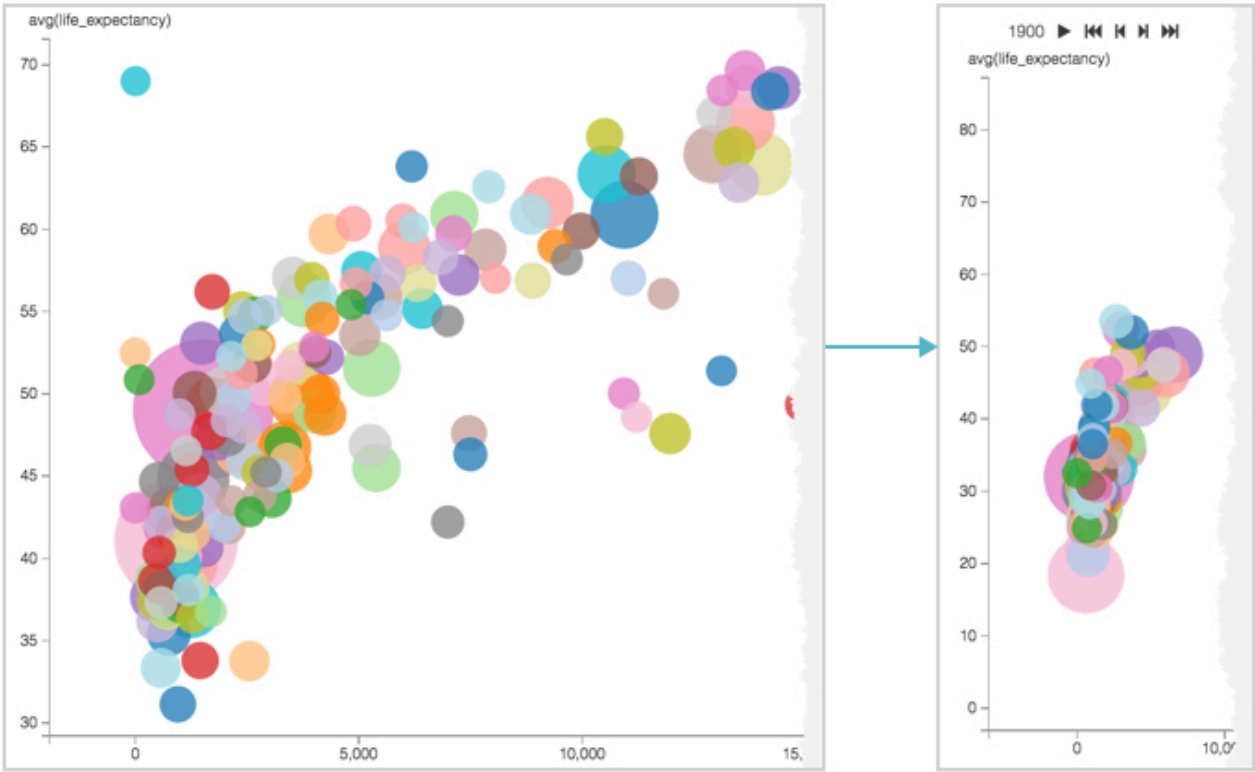

Showing dimensions as marks

Tis task applies to Pie charts.

To show the dimension label for each section of the visual, navigate to the Marks menu, and select the Show Dimensions option.

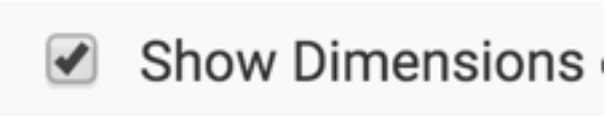

Note how turning on the dimension display changes the visual.

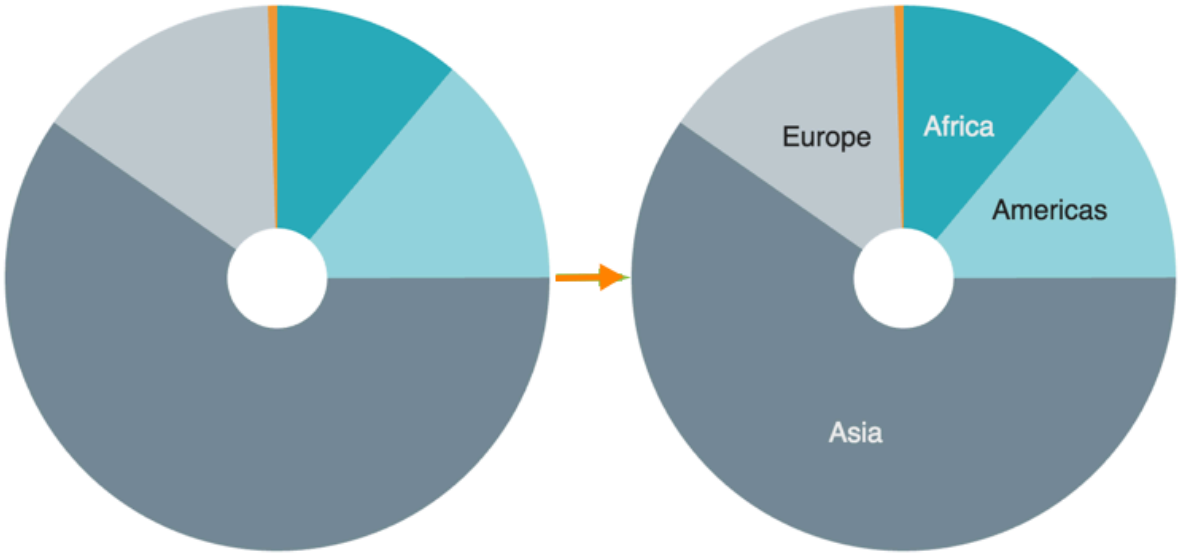

Showing percentages as marks

This task applies to Pie charts.

To show the percentage of ring/pie for each dimension, navigate to the Marks menu, and select the Show Percentages option.

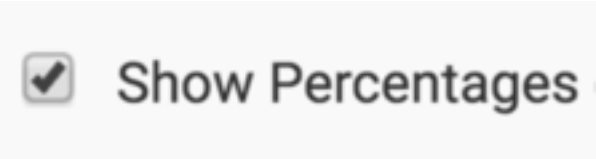

Note how turning on the percentage display changes the visual.

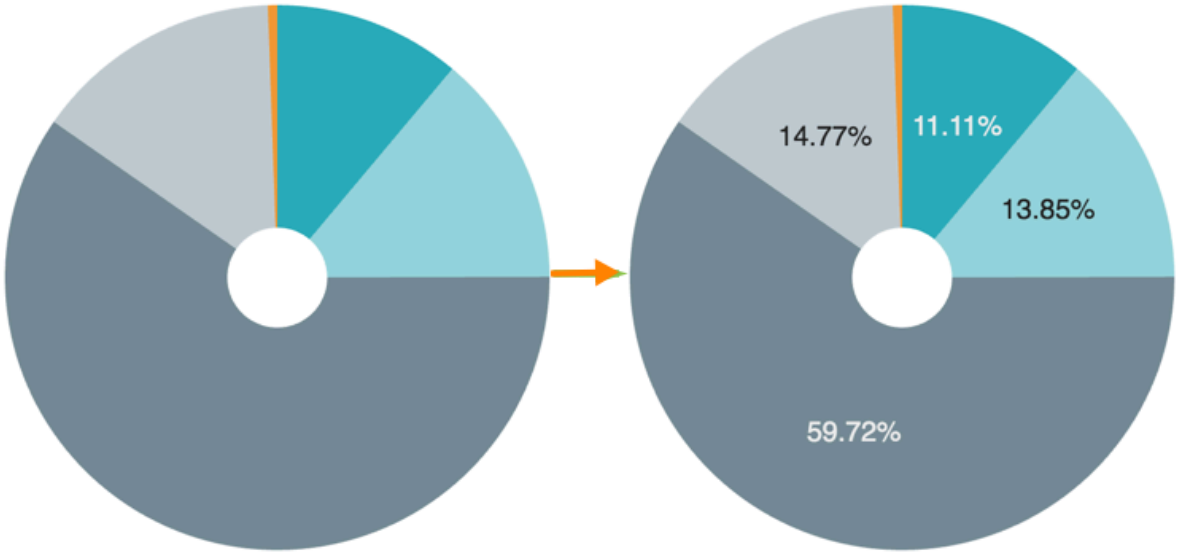

Changing number of decimals

This task applies to KPI visuals.

To change the number of decimals in a KPI visual, navigate to the Marks menu, and enter a new value in the Decimals in Display selector The default value is 1.

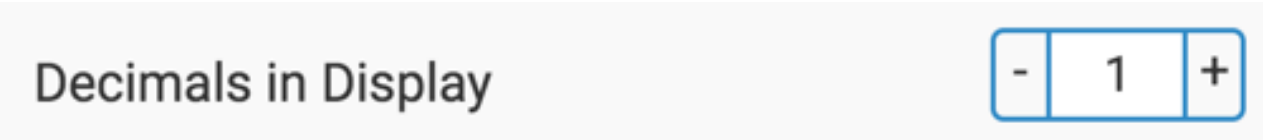

Note how changing the number of decimals changes the display.

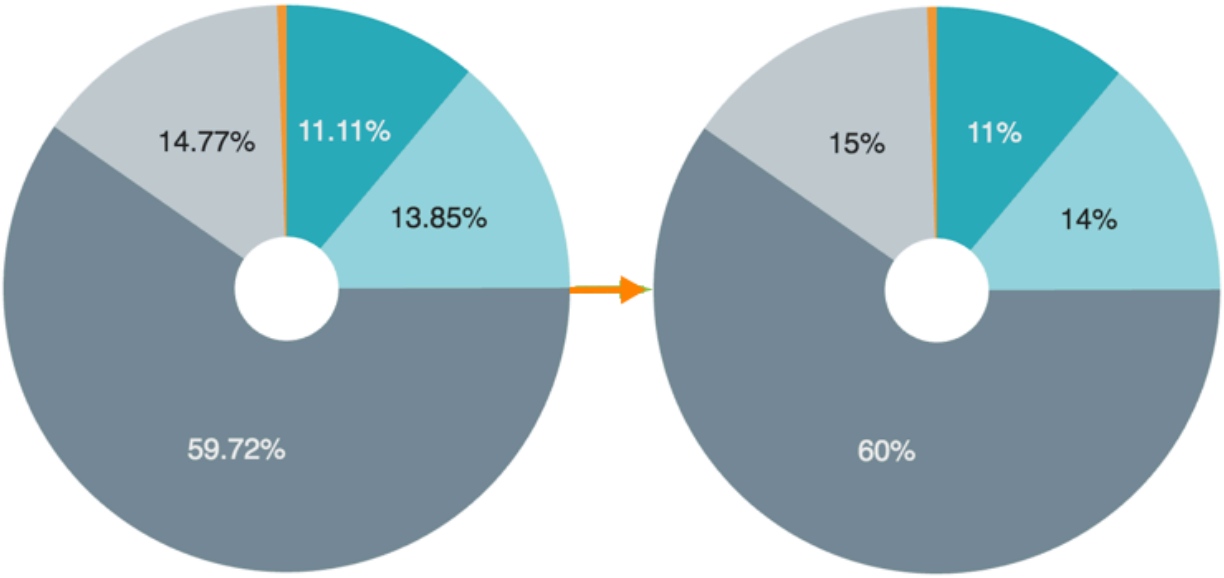

Changing display to overlap

This task applies to Radial Charts.

To view radial plots as full circles instead of wedges, navigate to the Marks menu, and select the Overlap marks option.

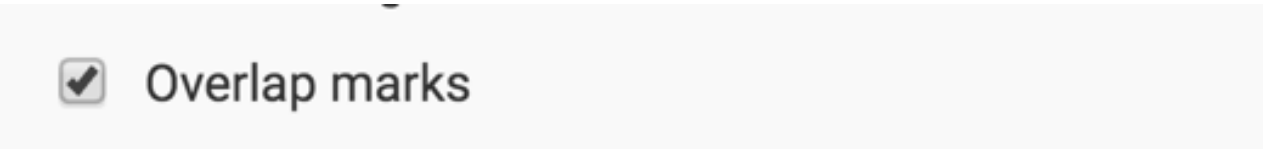

Example

Here is a radial visual with default display.

Note how turning on the overlap option changes the visual.

Showing the same source and destination values

This task applies to Chord and Dendrogram visuals.

To ensure that the end points are defined by the same source and destination values, navigate to the Marks menu, and select the Source & destination data come from the same base values option.

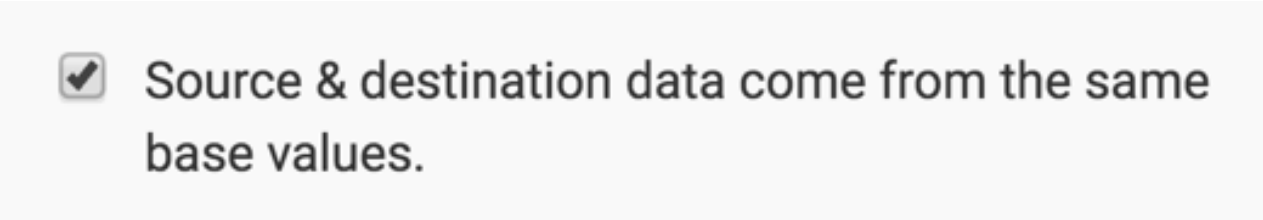

Changing size of chord visual

Cloudera Data Visualization enables you to change the size of the chord visual.

To change the size of a chord visual, navigate to the Marks menu, and change the value of the Minimum size option.

Compare the appearance of the chord visual when we change the size from 190 to 290 pixels.

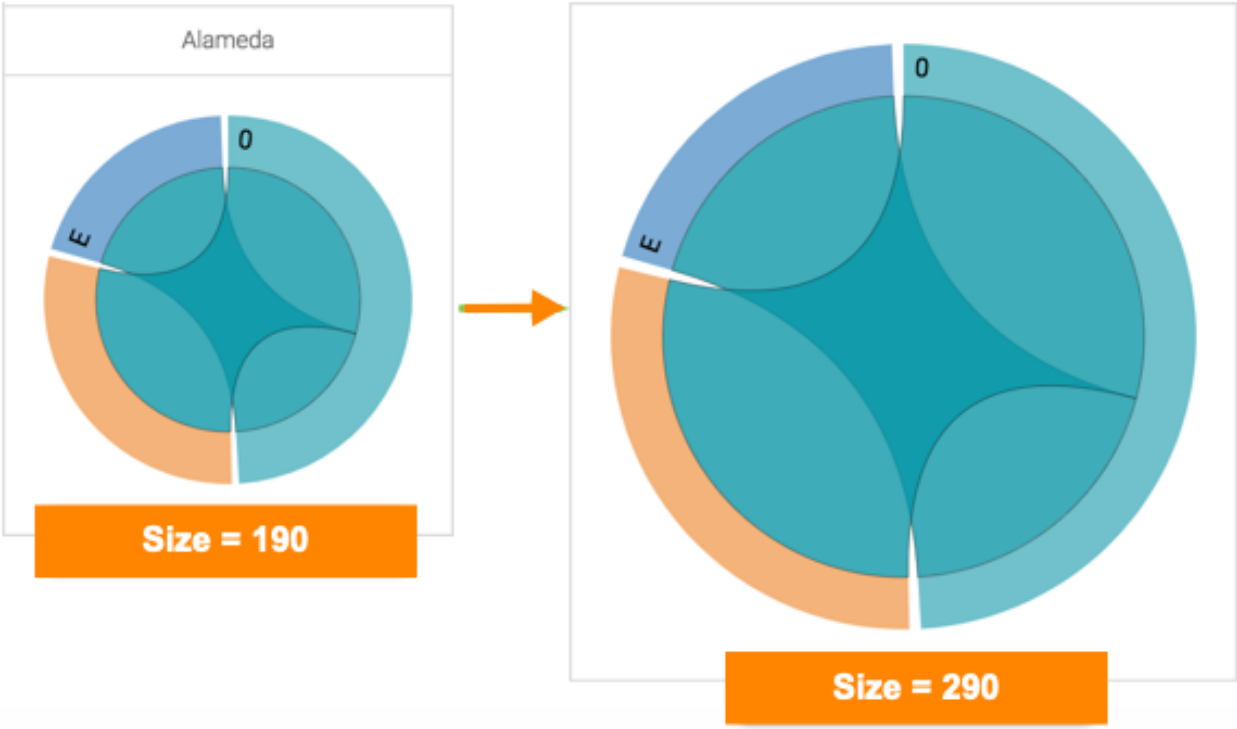

Displaying bubbles on maps

Cloudera Data Visualization enables you to display measurements over maps as bubbles. This task applies to Maps.

Many map visuals render as choropleths by default. To display measurements over maps as bubbles, click the Settings menu, select the Marks sub-menu, and select the Always display marks as bubbles option.

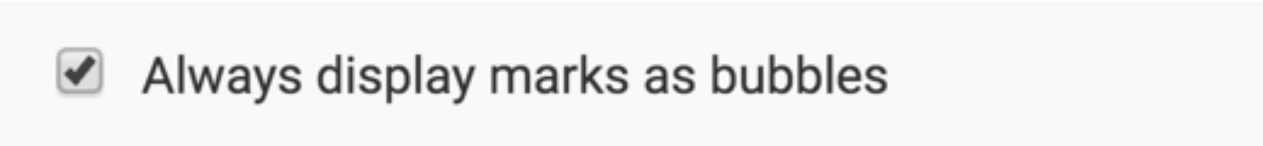

Compare the appearance of world map as a choropleth map, and as a map with bubble marks.

To adjust the size of the bubbles, see *Change mark size range*.

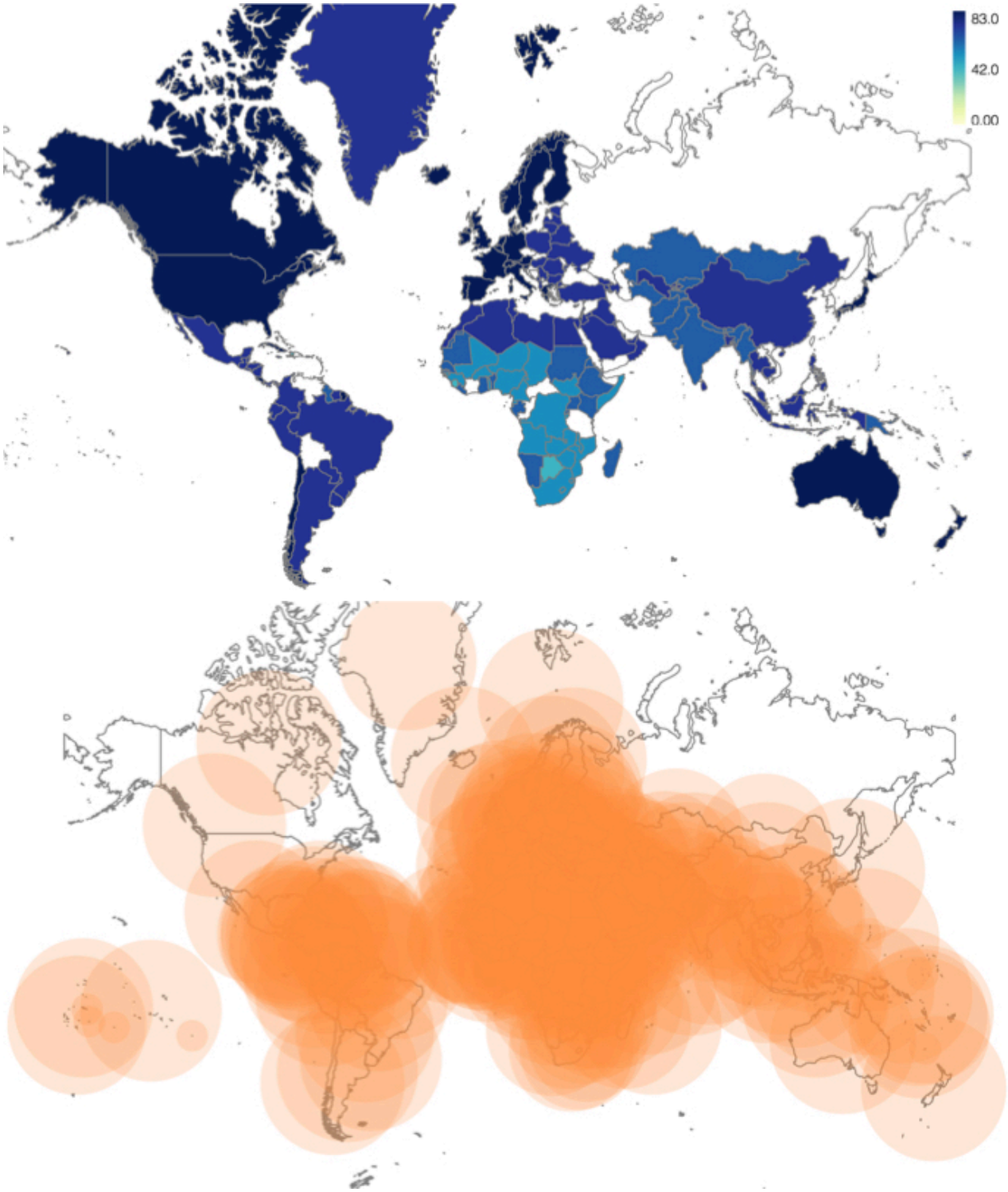

Related Information

Changing mark size range

Changing layout gravity

The Layout Gravity models a force that keeps nodes centered in the visible area, and keeps the disconnected subgraphs in the visual.

To change the layout gravity, navigate to the Marks menu, and change the value for the Layout gravity option. The default setting is 0.1, the range is between 0 and 1.

If setting the layout gravity to 0, use other constraints.

Example

Compare the appearance of a visuals with layout gravity of 0, 0.5, and 1. In each of these cases, the Layout Theta option is at the default value of 0.8, and all other settings remain the same.

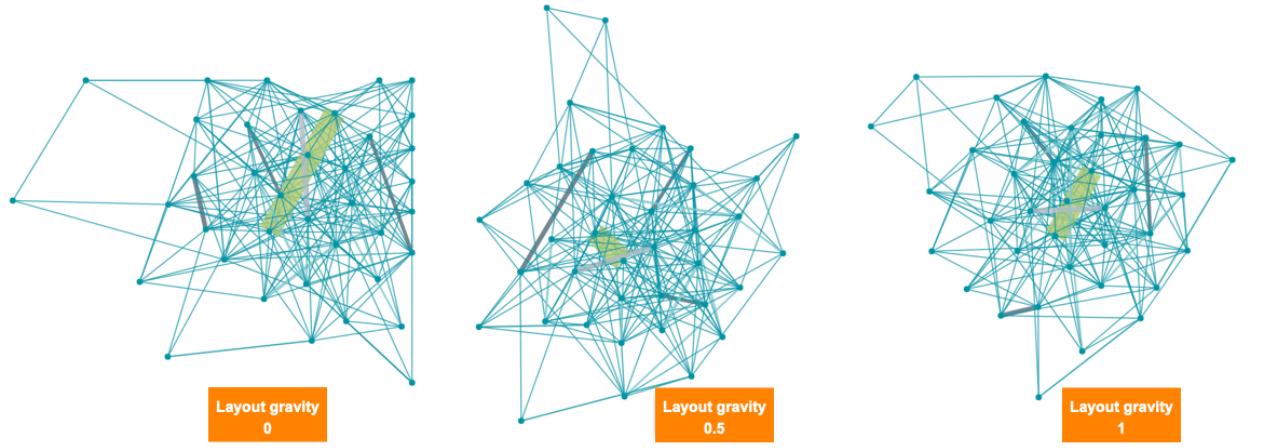

Changing layout theta

The layout theta determines the accuracy of the Barnes–Hut approximation computation; larger values increase the speed of the simulation but lower its accuracy. When theta is 0, the algorithm becomes a direct-sum algorithm.

To change the layout theta setting, navigate to the Marks menu, and change value for Layout theta option. The default setting is 0.8.

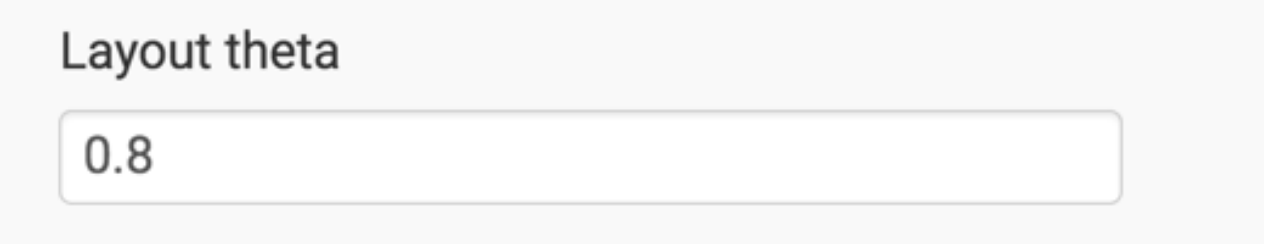

Example

Compare the appearance of a visuals with layout theta settings of 0, 0.8, and 1.

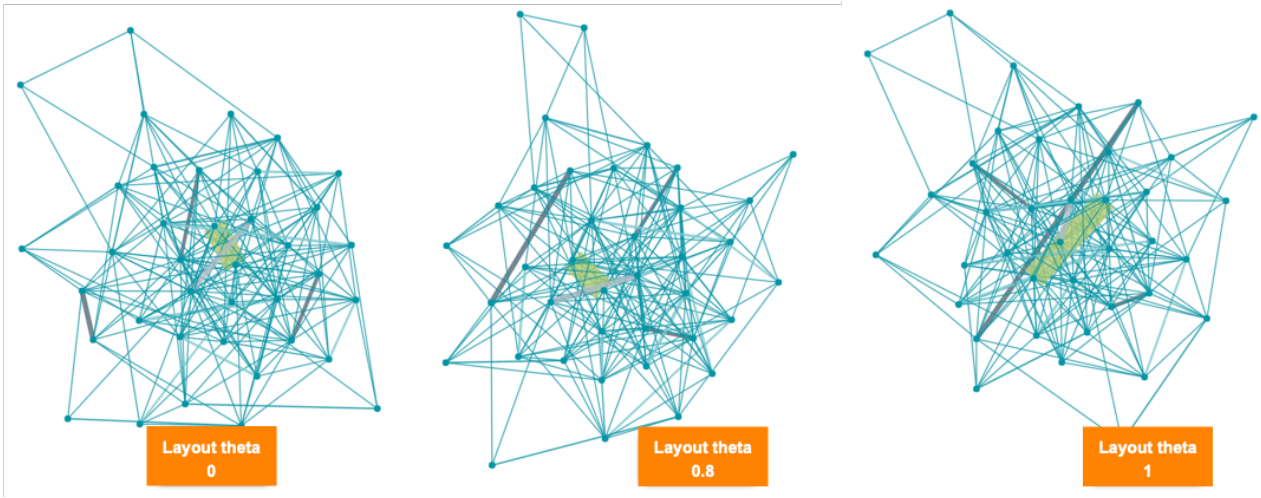

Showing node labels

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

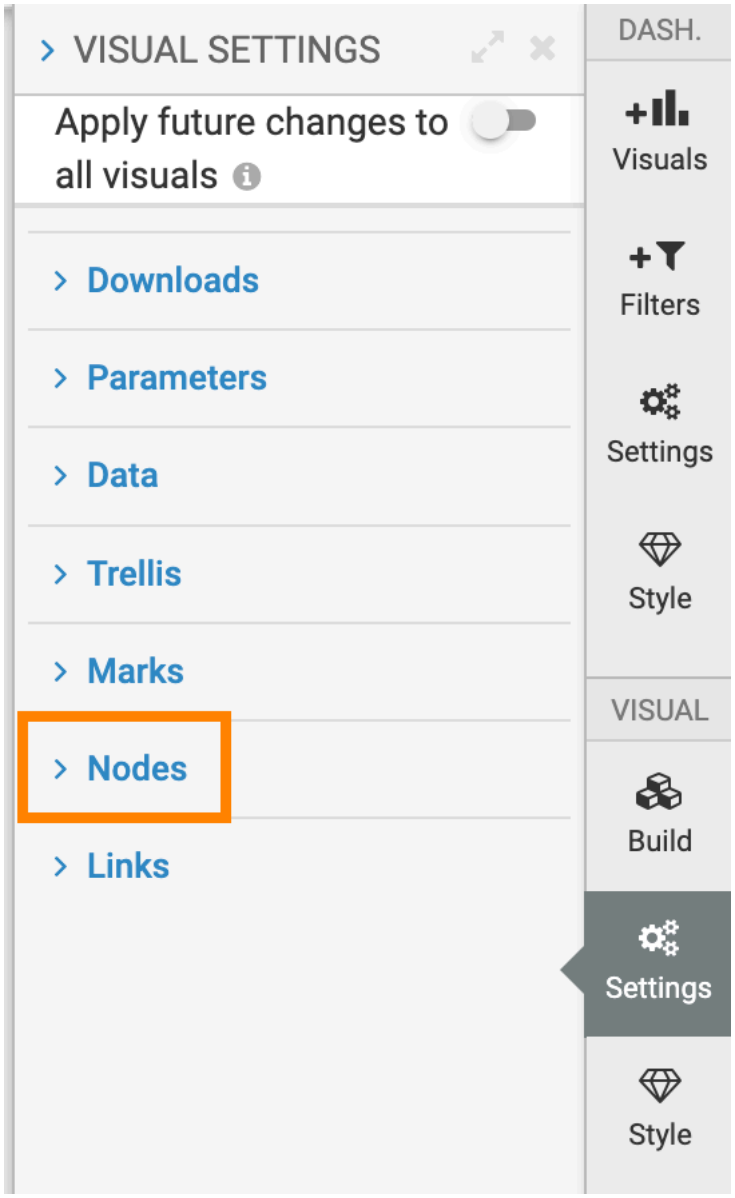

3. To change the opacity (color saturation) of nodes, select the Show node labels option.

Tokenizing input

This setting is for the Word Cloud visual type.

To tokenize input for World Cloud visuals, navigate to the Marks menu, and select the Tokenize Input option.

Displaying gauge needle

In a Gauge visual, Cloudera Data Visualization helps you to display the measure mark as a needle or a line from the center to the measure value on the arc of a Gauge visual.

To show a gauge arrow from the center to the measure value on the arc, navigate to the Marks menu, and select Needle under the Key measure mark option. To remove the gauge needle select None. To show a gauge line instead of an arrow, select Line.

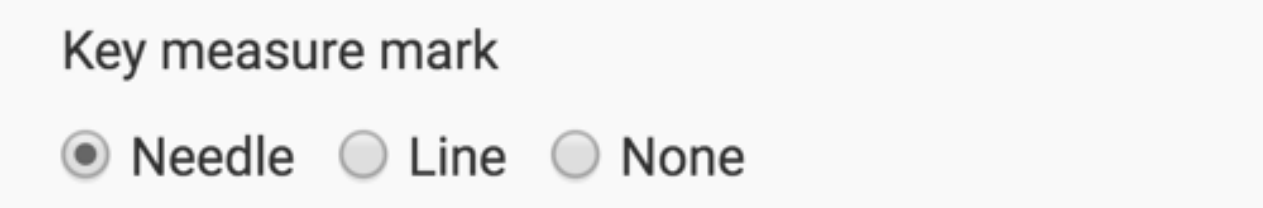

Example

Note the Gauge needle in the second image after selecting this option.

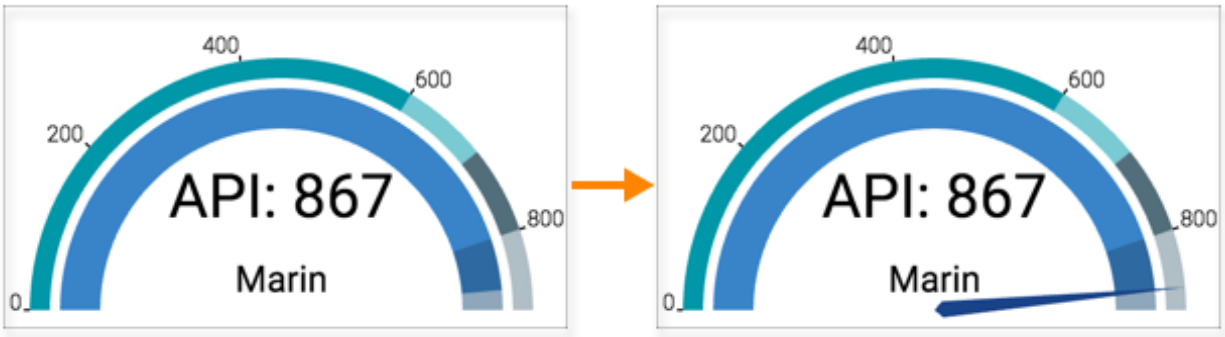

Displaying gauge line on qualitative arc

In a Gauge visual, Cloudera Data Visualization enables you to compare the measure value to the maximum qualitative range.

To show a compare line on the qualitative arc, navigate to the Marks menu, and select the Display compare line on qualitative arc option.

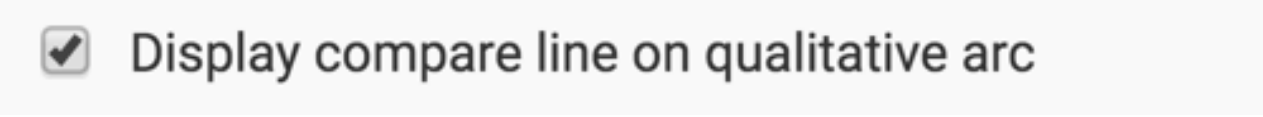

Example

In the second image, note the presence of a gauge line on the qualitative arc.

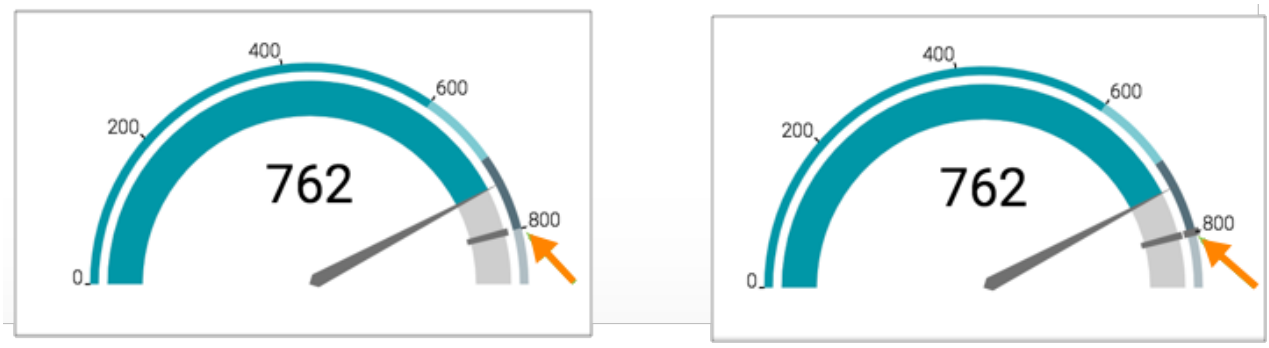

Displaying axis ticks

This task applies to Gauge charts.

By default, the axis wraps around the outer arc of a gauge chart, and shows measurements as evenly-spaced tick marks. To hide the tick-marks, navigate to the Marks menu, and de-select the Display axis ticks option.

Note how hiding axis ticks changes the visual.

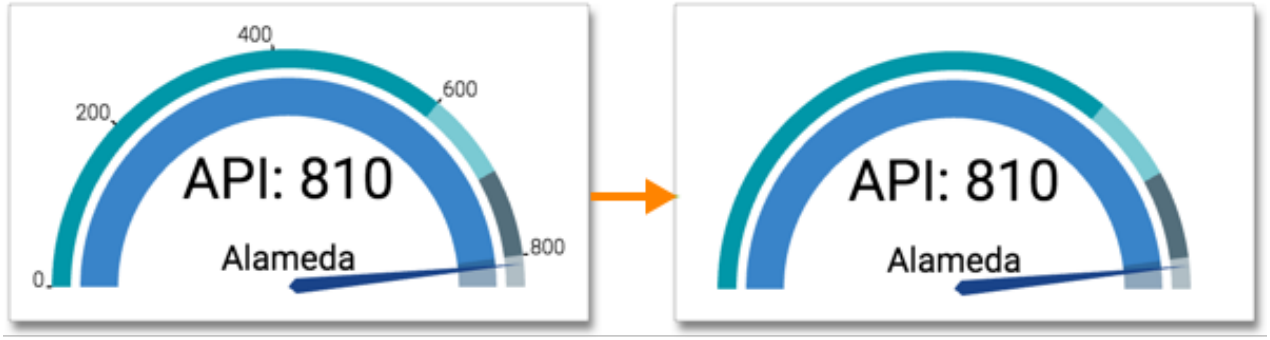

Changing outer width

This task applies to Gauge charts.

By default, the outer width of a gauge chart is set to the range of 200-250. To change it, navigate to the Marks menu, and enter new values for the Outer Width option.

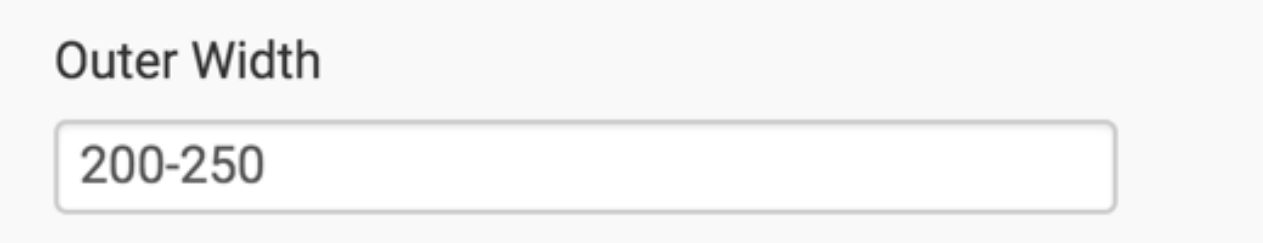

Note how adjusting the outer width changes the appearance of the visual, at the same magnification. While the smaller ranges produce compact charts that work well for trellised visual groupings, it is challenging to use them with descriptive labels.

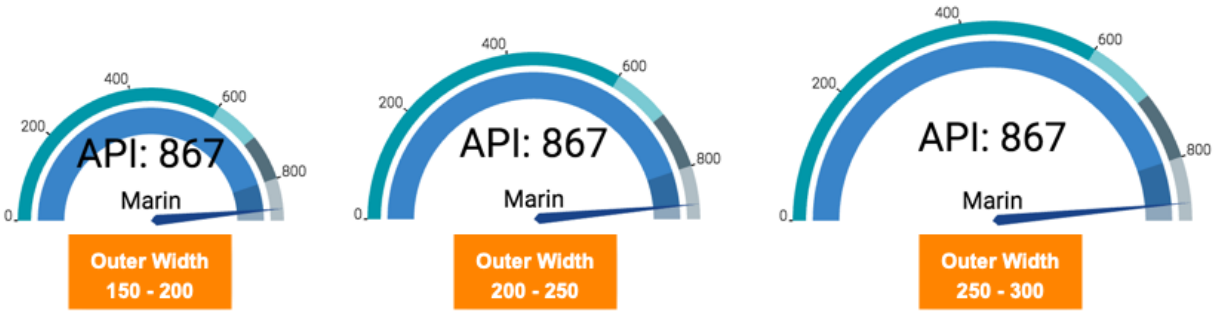

Changing qualitative thickness

This task applies to Gauge charts.

By default, the width of the qualitative (outer) arc of the visual is set to 10 pixels. To change it, navigate to the Marks menu, and enter the new value for the Qualitative Thickness option.

Note how adjusting this setting changes the appearance of the visual.

Changing arc gap

This task applies to Gauge charts.

By default, the gap between the measurement (inner) and qualitative (outer) arcs is 5 pixels. To change it, navigate to the Marks menu, and select a new value for the Arc Gap option.

Note how adjusting this setting changes the appearance of the visual. The outer (qualitative) arc is fixed, while the inner (measurement) arc radius changes to accommodate the arc gap setting. When this setting is at 0, a gap does not appear.

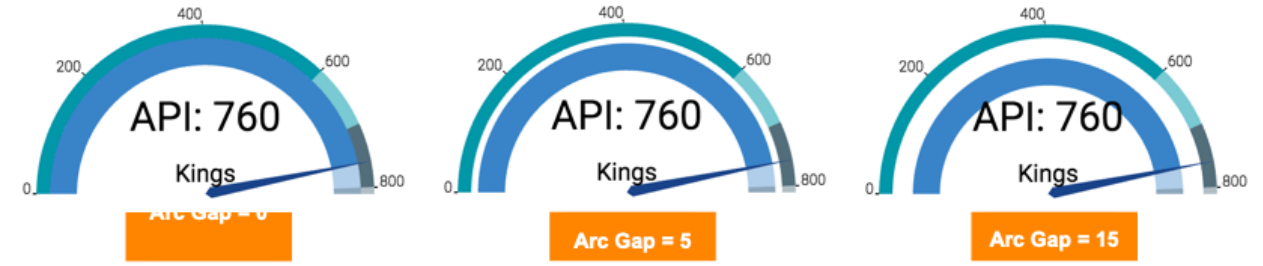

Changing arc thickness

This task applies to Gauge charts.

By default, the thickness of the measurement (inner) arc is 20 pixels. To change it, navigate to the Marks menu, and select a new value for the Arc Thickness option.

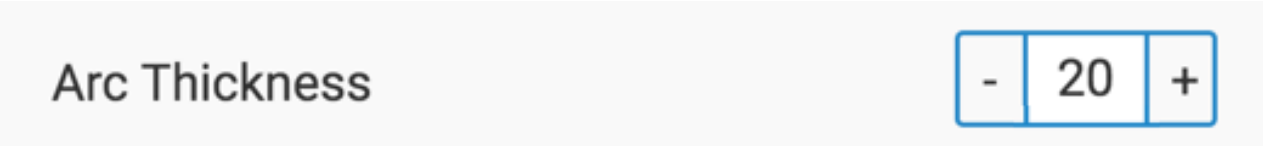

Note how adjusting this setting changes the appearance of the visual.

Changing label font size

This task applies to Gauge charts.

By default, the font size of the primary label in gauge charts is 30. To change it, navigate to the Marks menu, and select a new value for the Label Font Size option.

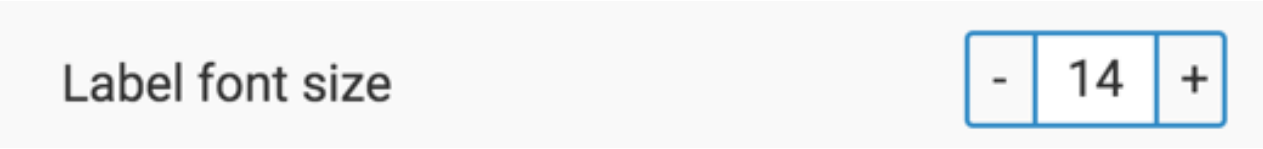

Note how adjusting this setting changes the appearance of the visual.

Changing label offset

This task applies to Gauge charts.

By default, the offset (vertical position) for the primary label in gauge charts is 0. To change it, navigate to the Marks menu, and select a new value for the Label Offset option.

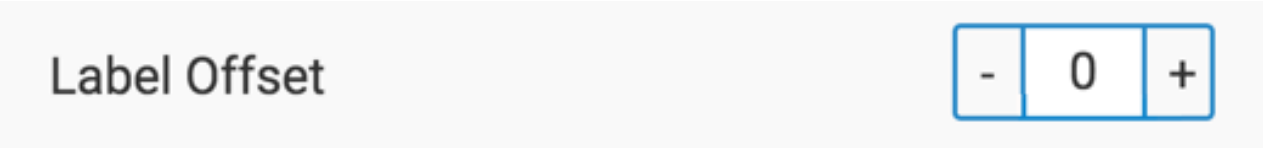

Note how adjusting this setting changes the appearance of the visual: negative values move the primary label upward, and positive values move it downward.

Changing sublabel font size

This task applies to Gauge charts.

By default, the font size of the secondary label in gauge charts is 18. To change it, navigate to the Marks menu, and select a new value for the Sublabel Font Size option.

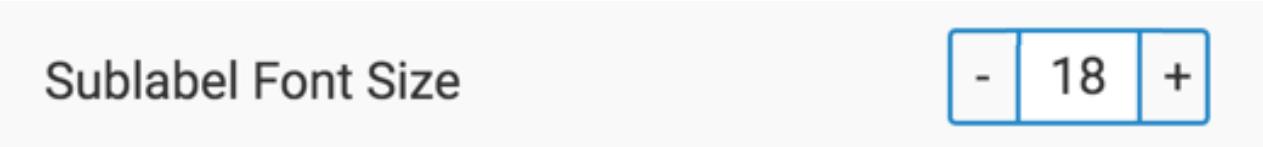

Note how adjusting this setting changes the appearance of the visual.

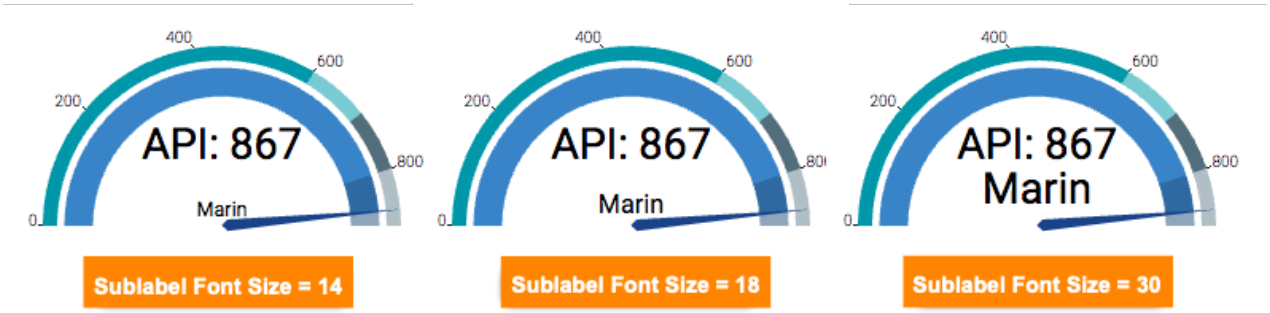

Changing sublabel offset

This task applies to Gauge charts.

By default, the offset (vertical position) for the secondary label in gauge charts is 0. To change it, navigate to the Marks menu, and select a new value for the Sublabel Offset option.

Note how adjusting this setting changes the appearance of the visual: negative values move the secondary label upward, and positive values move it downward.

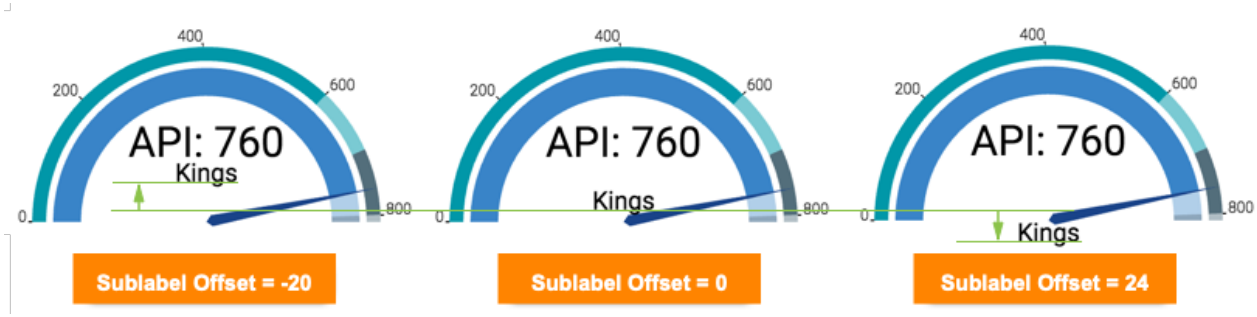

Making box plots horizontal

To make box plot visuals that are vertical by default display horizontally, navigate to the Marks menu, and select the Make box plots horizontal option.

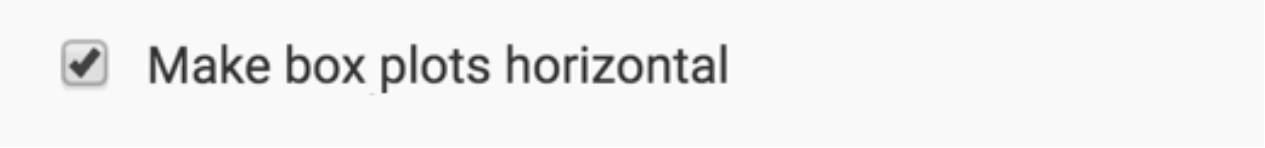

Here is an example of how the visual changes when the box plot orientation changes from vertical to horizontal.

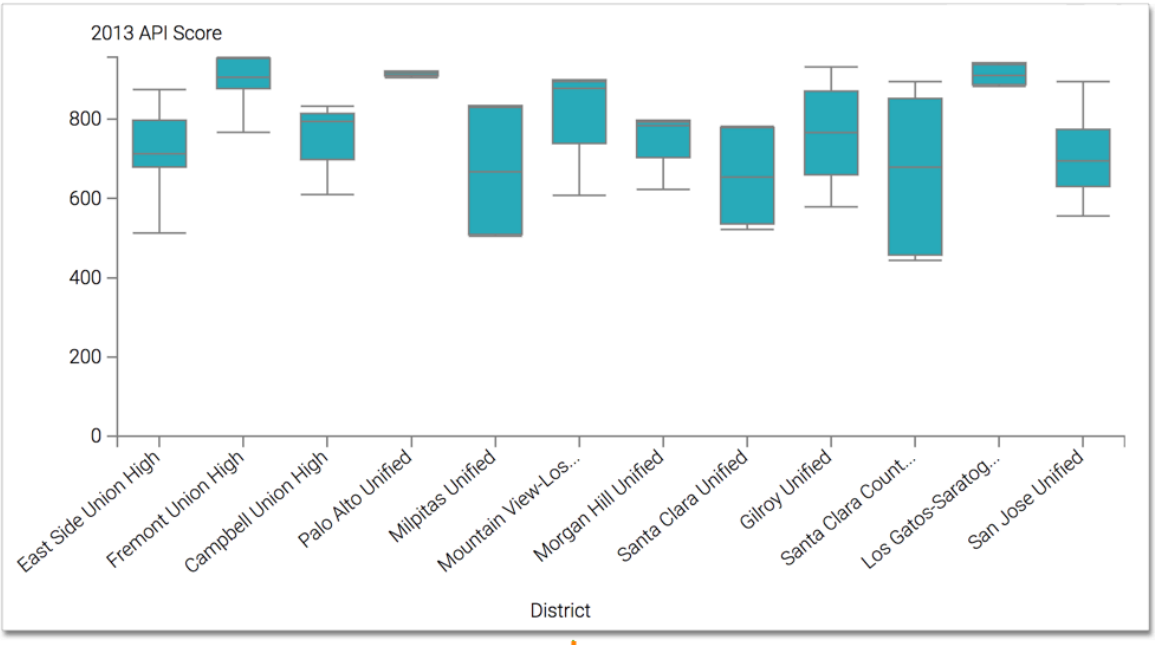

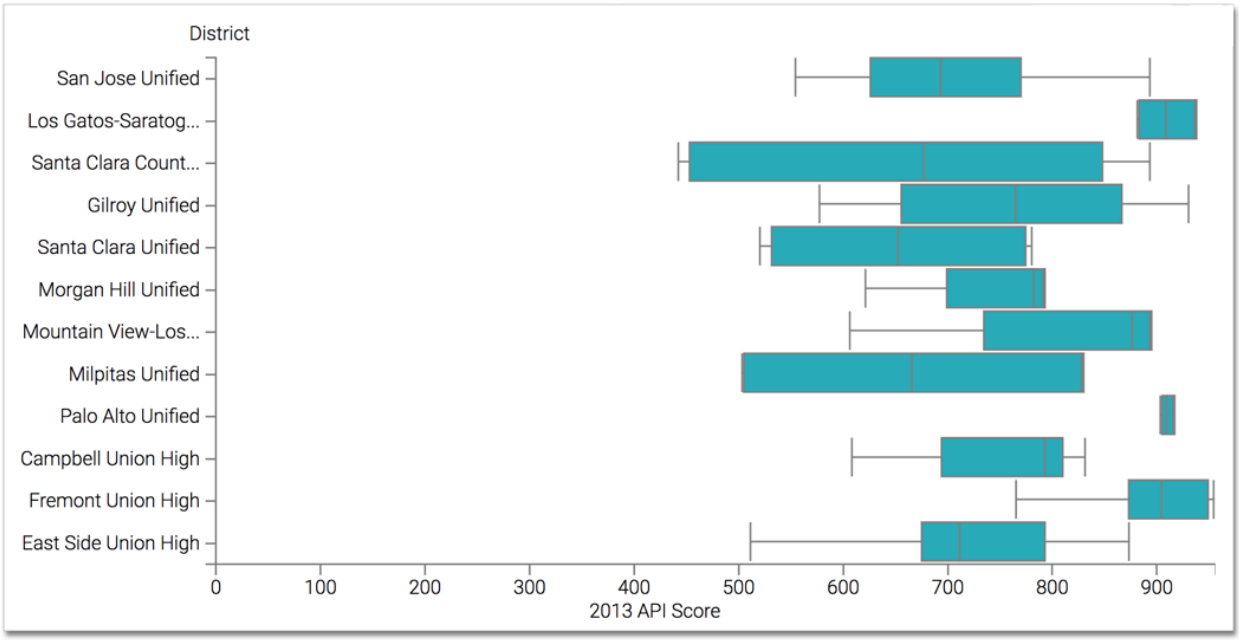

Changing outline width

To change the outline width of displayed items, such as boxes on box plot visuals, navigate to the Marks menu, and adjust the selector for the Outline width option.

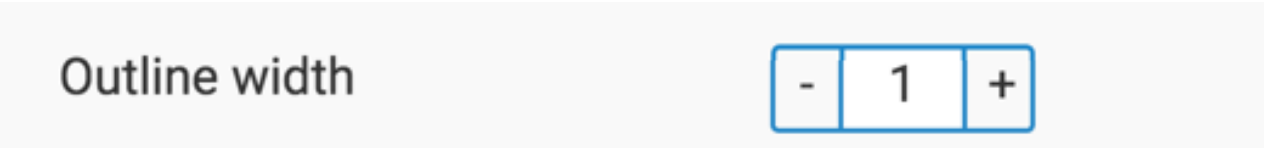

Changing font size of main indicator

This setting is specific to KPI visuals.

To change the font size of the main indicator, navigate to the Marks menu, and adjust the selector for the Font size main value option. The default value is 36.

This image illustrates the visual with font size of 16, 26, 36 (default), and 46 for the main indicator.

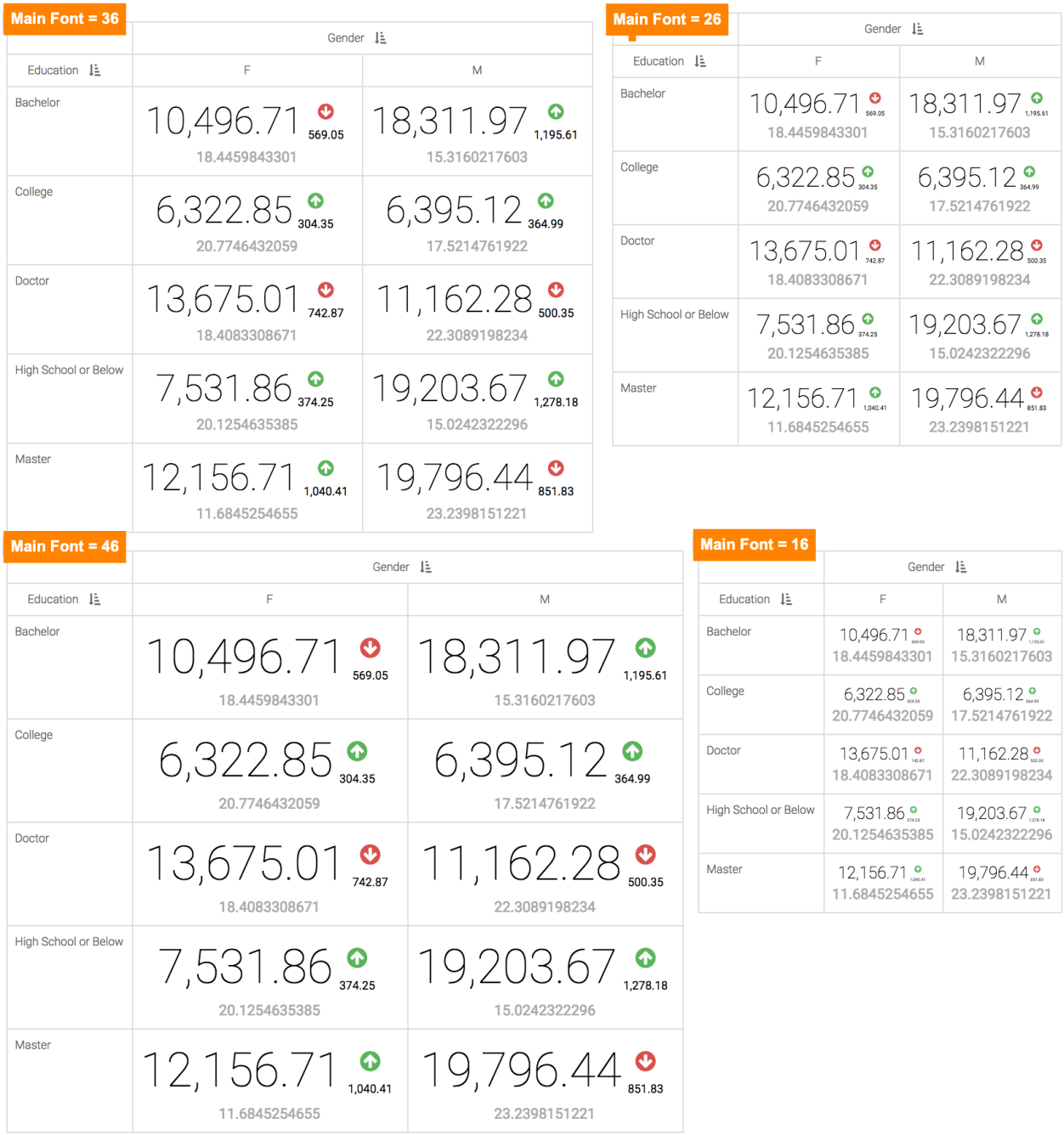

Changing value of main indicator

This setting is specific to KPI visuals.

To specify what is reported by the main indicator, navigate to the Marks menu, and select in Main value format one of the options:

- Main Measure (default), that appears on either the Measure or Dimension shelves
- Percent Changed, difference between the Measure and Compare To values as percentage
- Delta, the arithmetic difference between the Measure and Compare To values

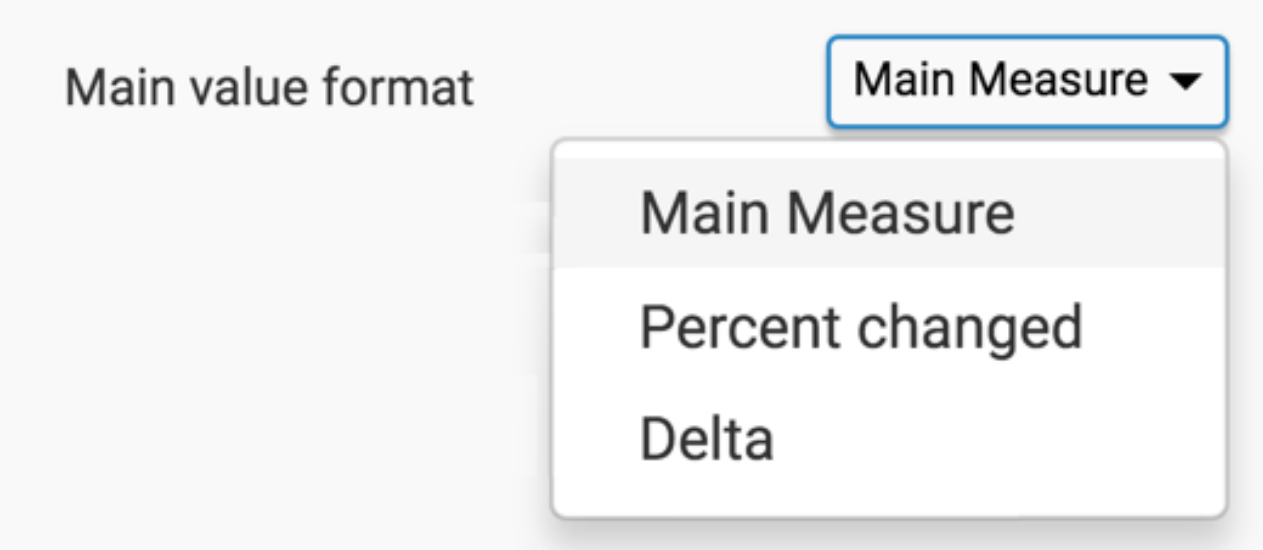

This image illustrates the visual with main indicator displayed as a Main Measure, Percent Changed, and Delta.

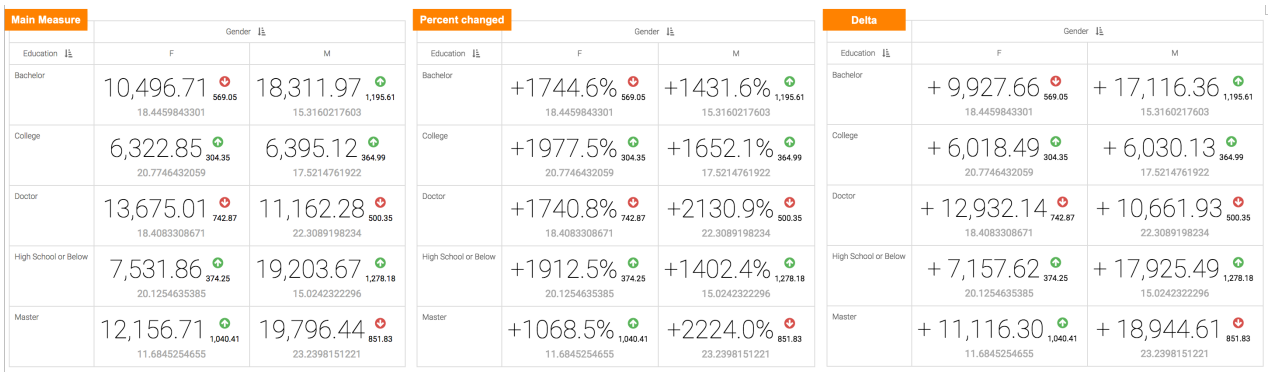

Main Measure

Education	Gender F	Gender M
Bachelor	10,496.71 569.05 18.4459843301	18,311.97 1,195.61 15.3160217603
College	6,322.85 304.35 20.7746432059	6,395.12 364.99 17.5214761922
Doctor	13,675.01 742.87 18.4083308671	11,162.28 500.35 22.3089198234
High School or Below	7,531.86 374.25 20.1254635385	19,203.67 1,278.18 15.0242322296
Master	12,156.71 1,040.41 11.6845254655	19,796.44 851.83 23.2398151221

Percent changed

Education	Gender F	Gender M
Bachelor	+1744.6% 569.05 18.4459843301	+1431.6% 1,195.61 15.3160217603
College	+1977.5% 304.35 20.7746432059	+1652.1% 364.99 17.5214761922
Doctor	+1740.8% 742.87 18.4083308671	+2130.9% 500.35 22.3089198234
High School or Below	+1912.5% 374.25 20.1254635385	+1402.4% 1,278.18 15.0242322296
Master	+1068.5% 1,040.41 11.6845254655	+2224.0% 851.83 23.2398151221

Delta

Education	Gender F	Gender M
Bachelor	+ 9,927.66 569.05 18.4459843301	+ 17,116.36 1,195.61 15.3160217603
College	+ 6,018.49 304.35 20.7746432059	+ 6,030.13 364.99 17.5214761922
Doctor	+ 12,932.14 742.87 18.4083308671	+ 10,661.93 500.35 22.3089198234
High School or Below	+ 7,157.62 374.25 20.1254635385	+ 17,925.49 1,278.18 15.0242322296
Master	+ 11,116.30 1,040.41 11.6845254655	+ 18,944.61 851.83 23.2398151221

Using same color for main indicator as arrow

This setting is specific to KPI visuals.

To match the color of the main indicator to the arrow, navigate to the Marks menu, and select the Use arrow color for main font color options.

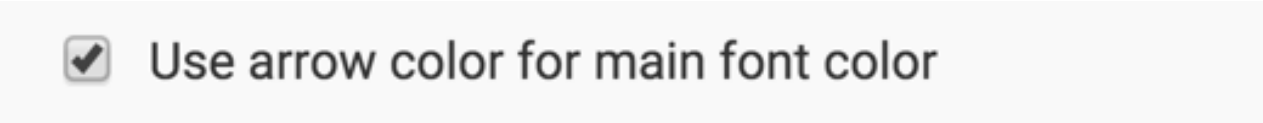

This image illustrates the visual with main indicator displayed in default color, and matching the arrow.

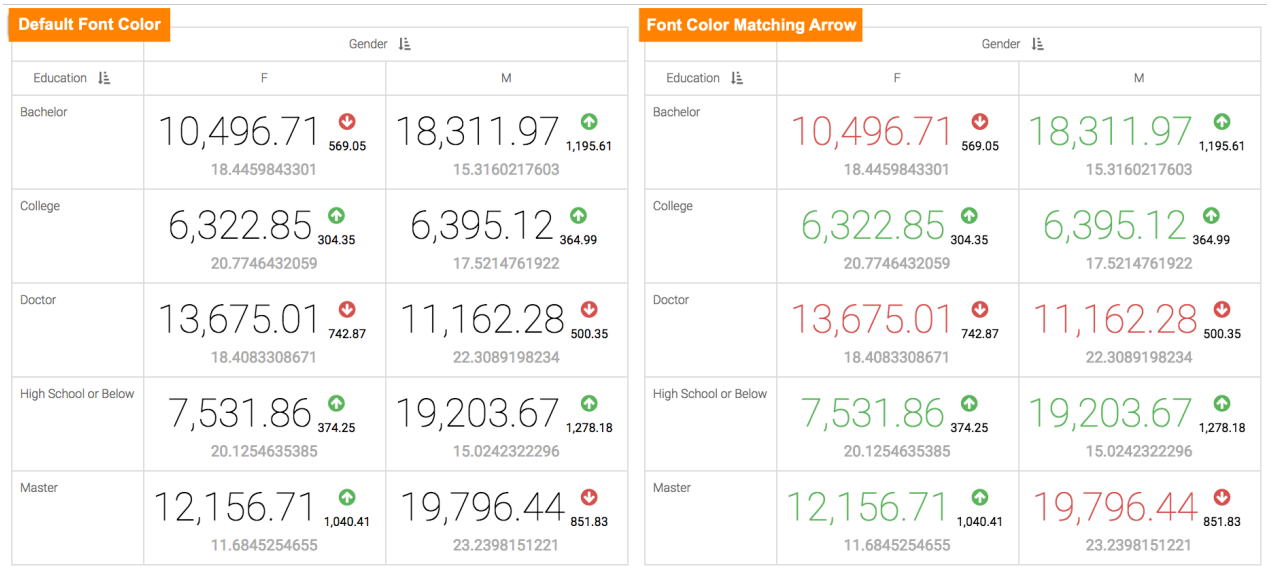

Displaying first measure as main indicator

This setting is specific to KPI visuals.

To display the first value on the Measure shelf as the main indicator, navigate to the Marks menu, and select the Display the first measure option.

If this option is not selected, the visual displays the first value on the Dimensions shelf as primary indicator.

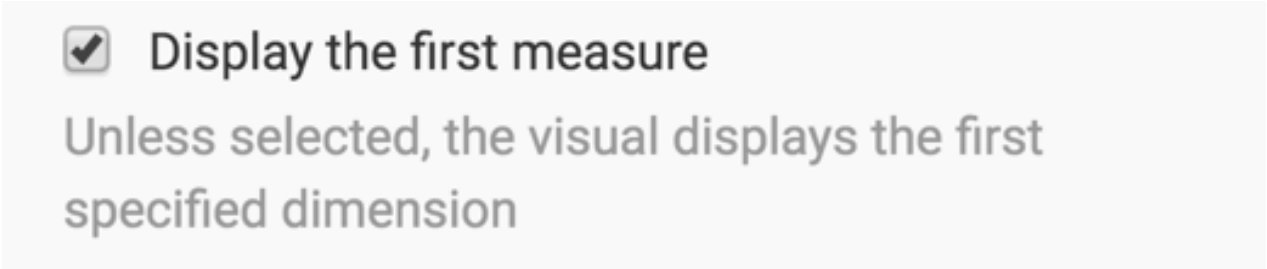

This image illustrates the visual with main indicator displayed as Measure (default), or Dimension.

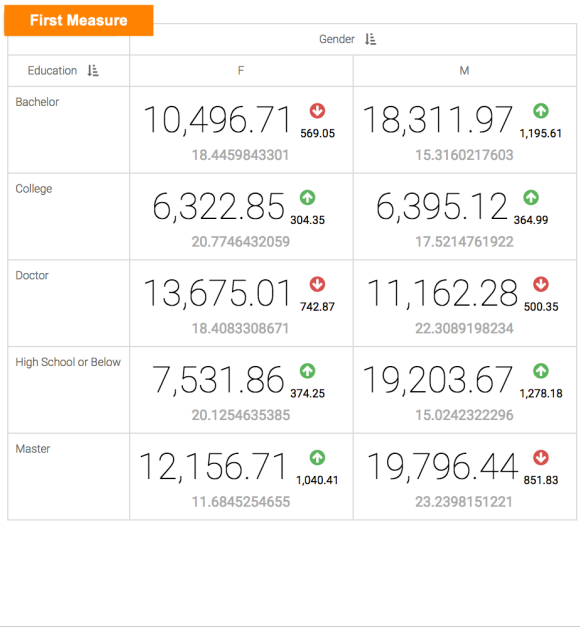

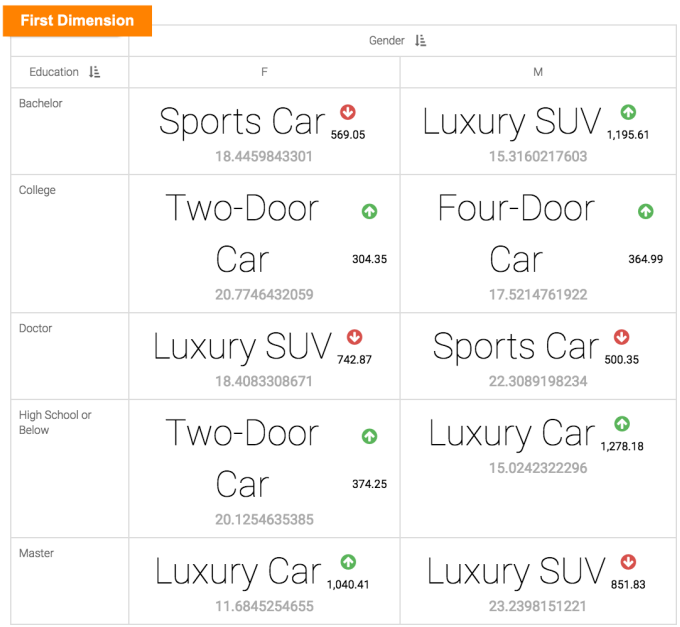

Changing compare indicator value

This setting is specific to KPI visuals.

To specify what is reported by the compare indicator, navigate to the Marks menu, and select in Compare display format one of the options:

- Value (default), that appears on the Compare To shelf
- Percent changed, difference between the main indicator value and Compare To value, as percentage
- Delta, the arithmetic difference between the main indicator value and Compare To value
- Do Not Show, to remove this indicator from the visual

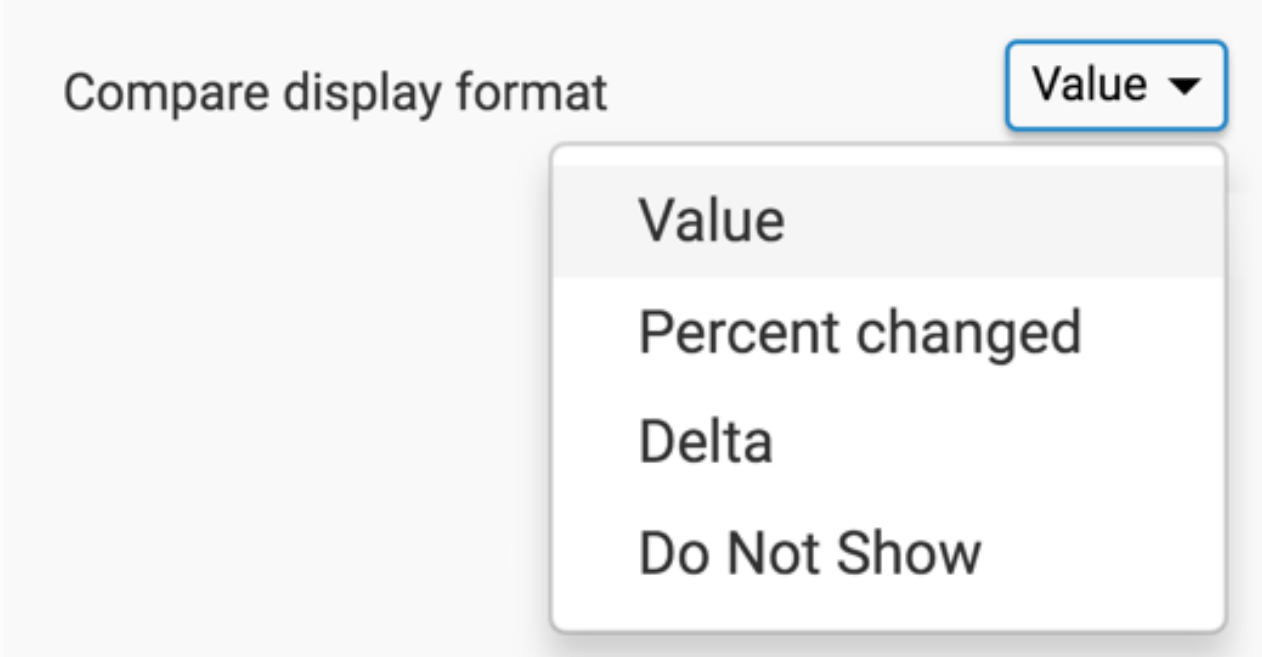

This image illustrates the visual with compare indicator displayed as a Value, Percent Changed, Delta, and Do Not Show.

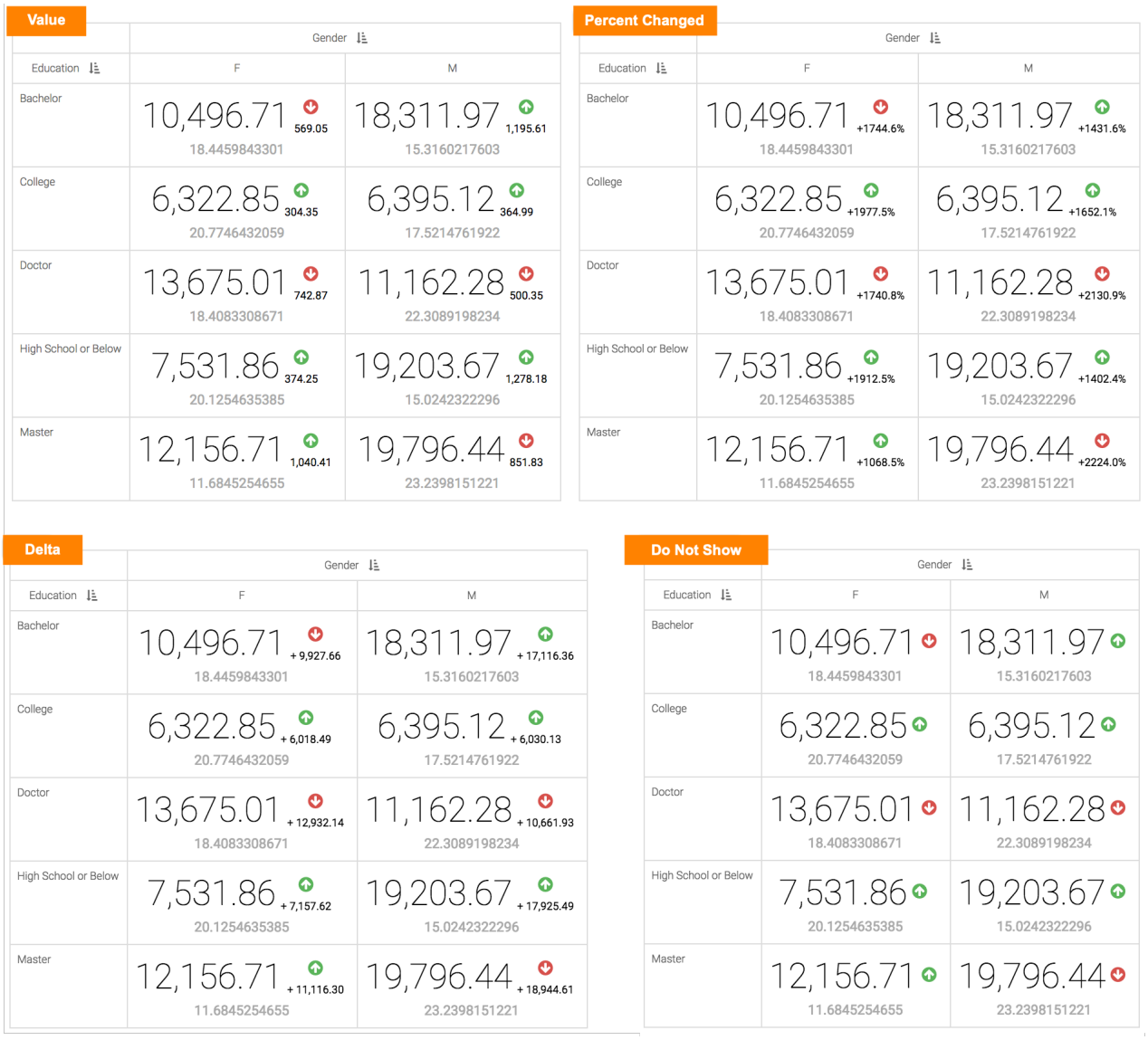

Changing compare indicator position

This setting is specific to KPI visuals.

To change the location of the compare indicator, navigate to the Marks menu, and select in Compare position one of the options:

- Right (default)
- Left
- Below
- Above

This image illustrates the visual with compare indicator displayed to the right, left, below, and above the main indicator.

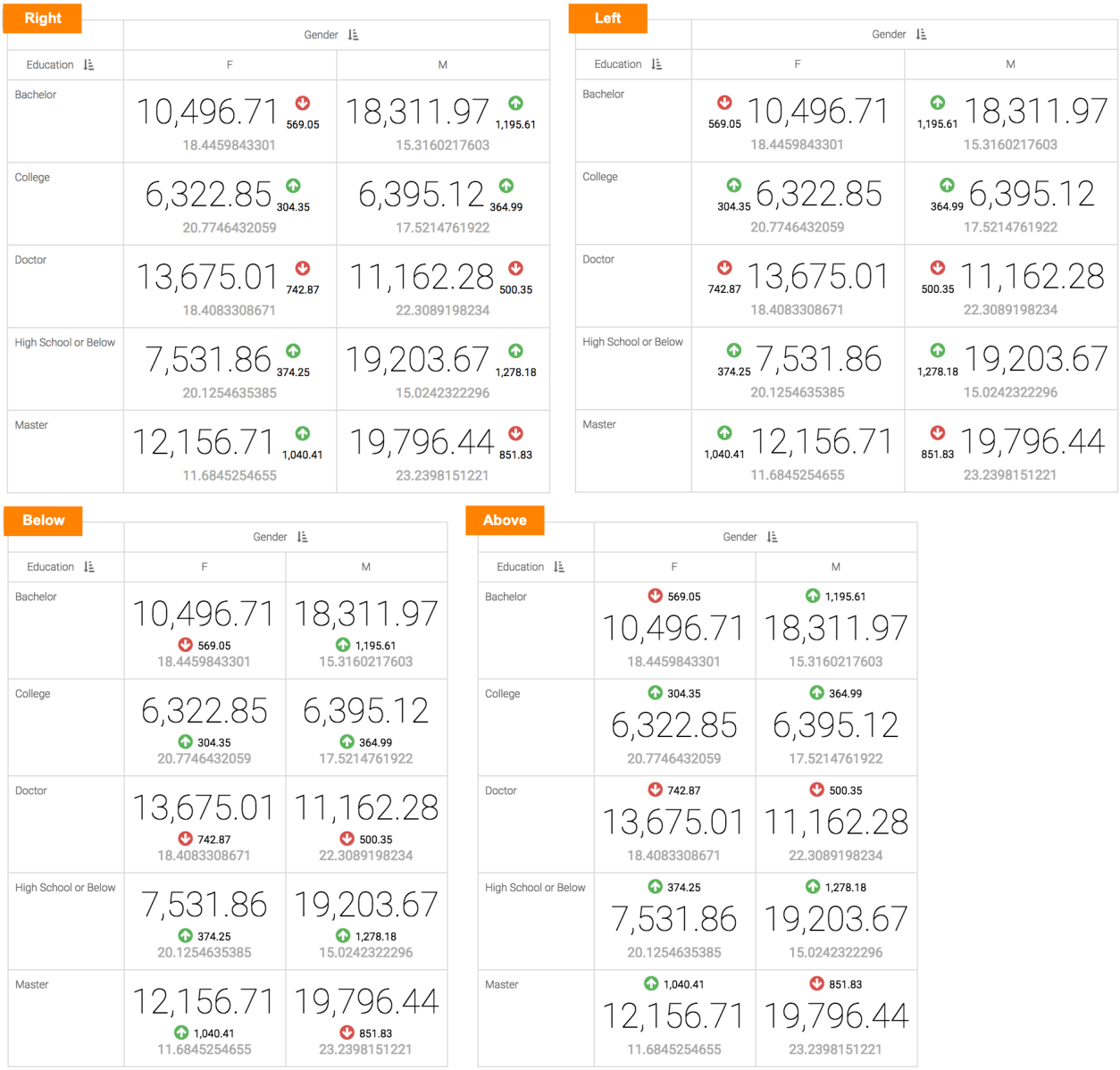

Changing arrow type

This setting is specific to KPI visuals.

To change the location arrow type on the KPI display, navigate to the Marks menu, and select one of the options in Arrow type. Note that the last option is None, which means not to display the arrow.

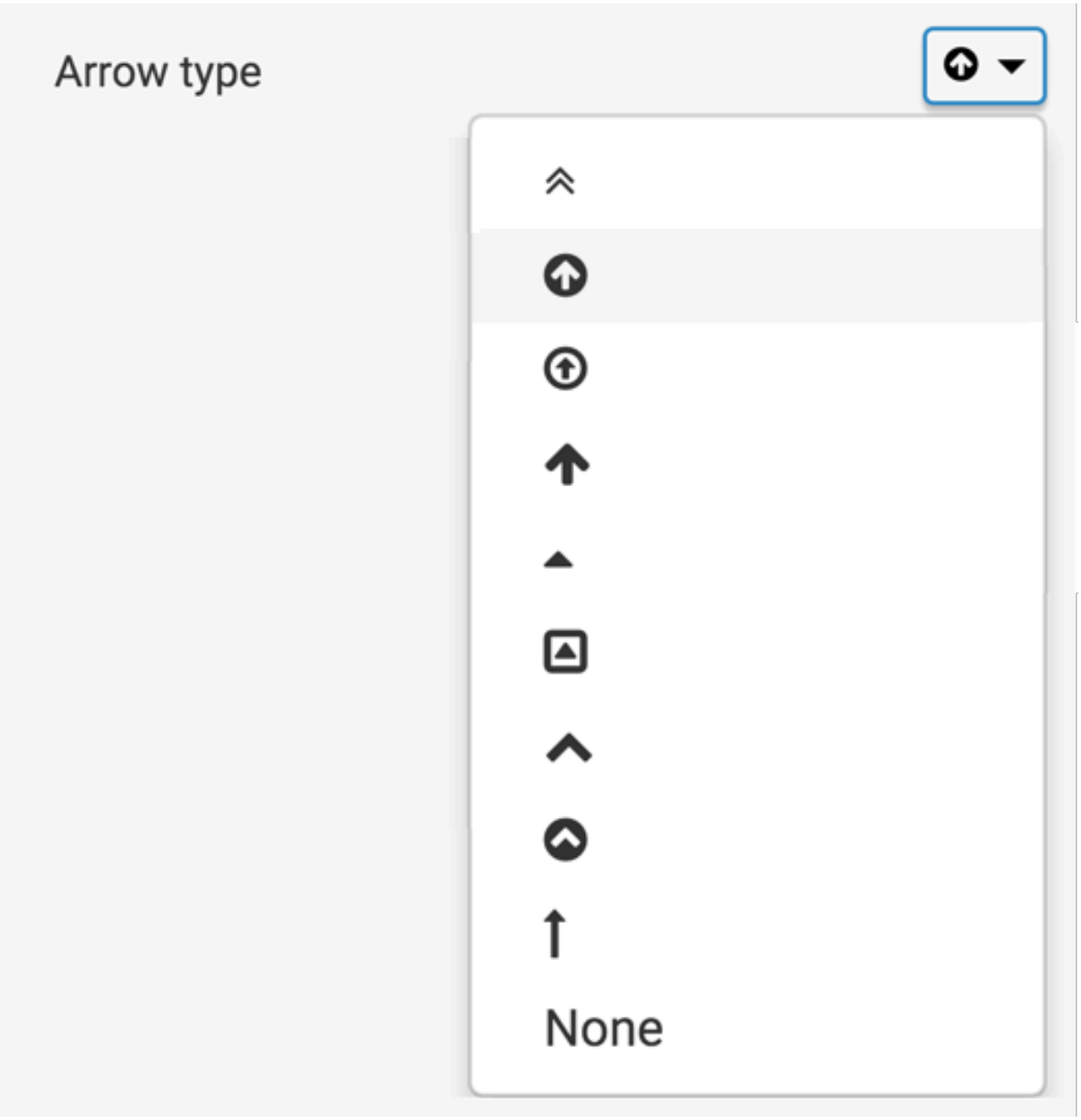

This image illustrates some of the different arrow type options.

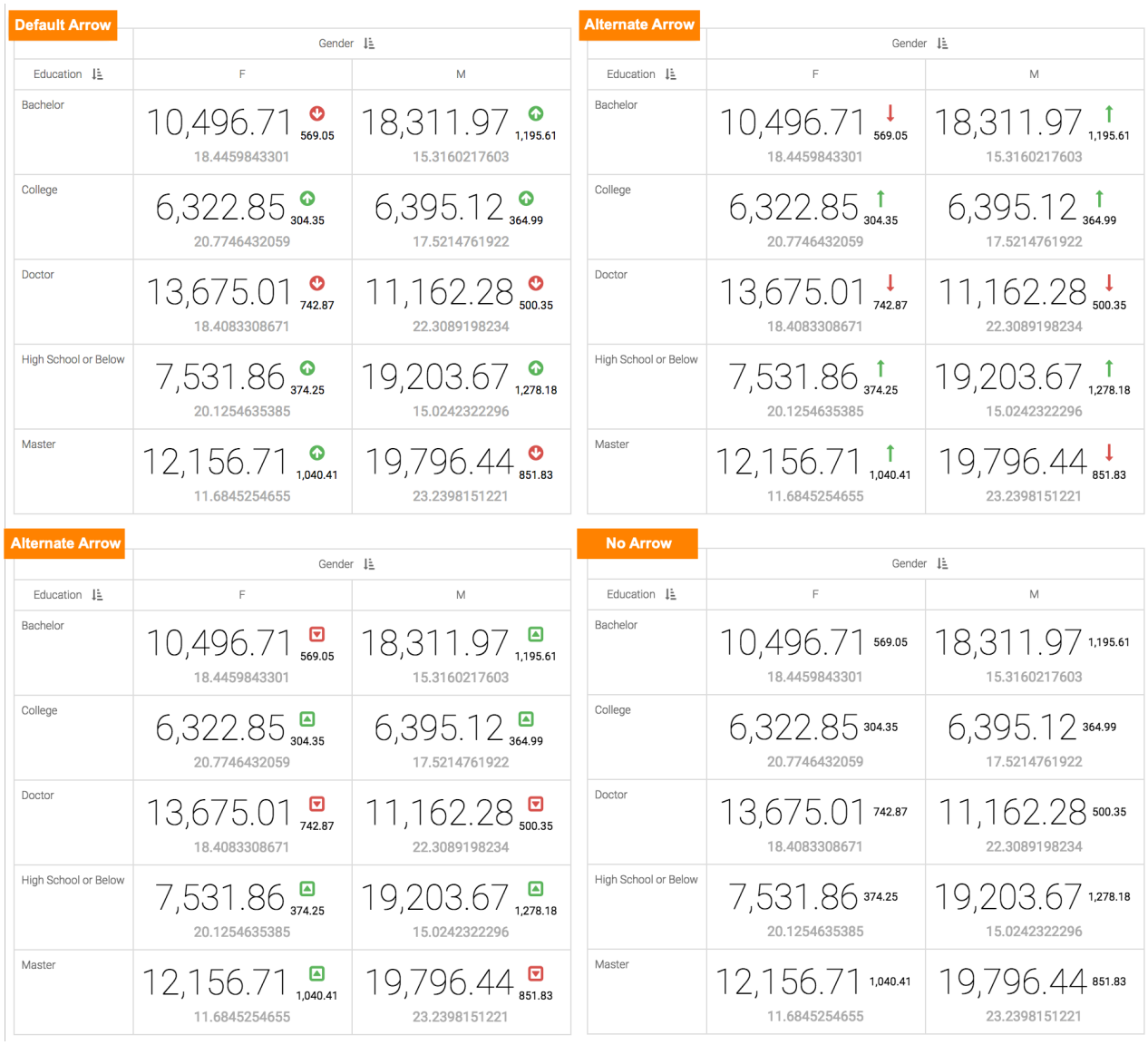

Flipping compare indicator

This setting is specific to KPI visuals.

To change the direction of the arrow (and the semantic meaning of the differences between the indicators), navigate to the Marks menu, and selection the Flip comparison option.

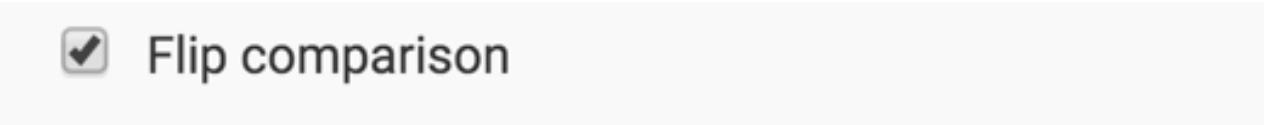

This image illustrates the visual with the default comparison setting, and with the flipped comparison. Note that the arrows flipped.

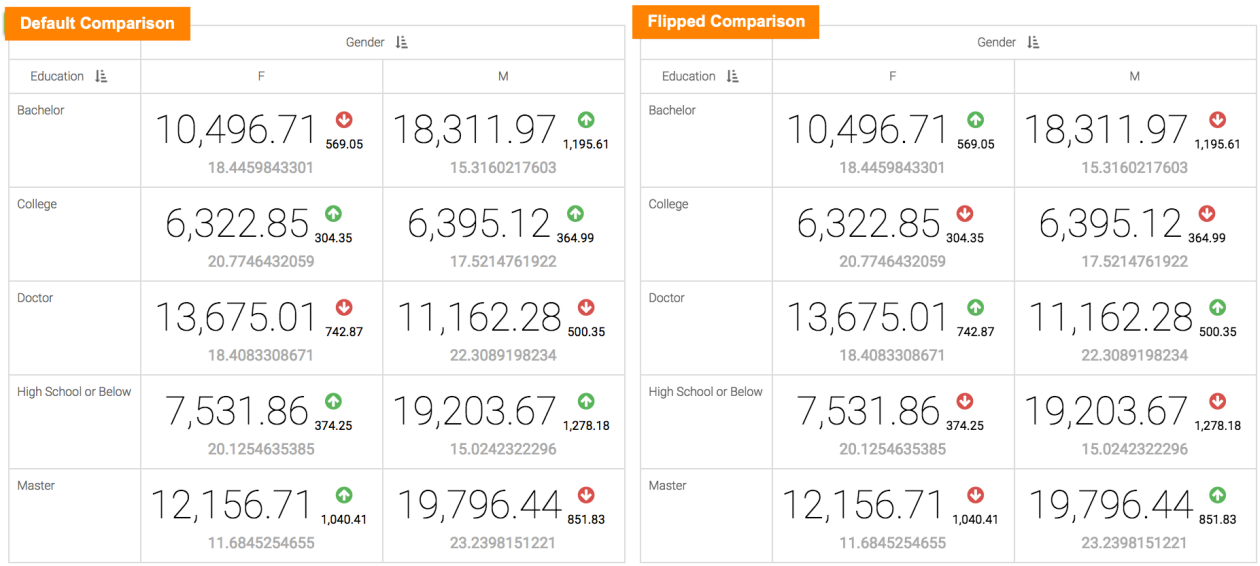

Changing label indicator position

This setting is specific to KPI visuals.

To change the location of the label indicator, navigate to the Marks menu, and make a selection in the Label location option:

- Below (default)
- Above

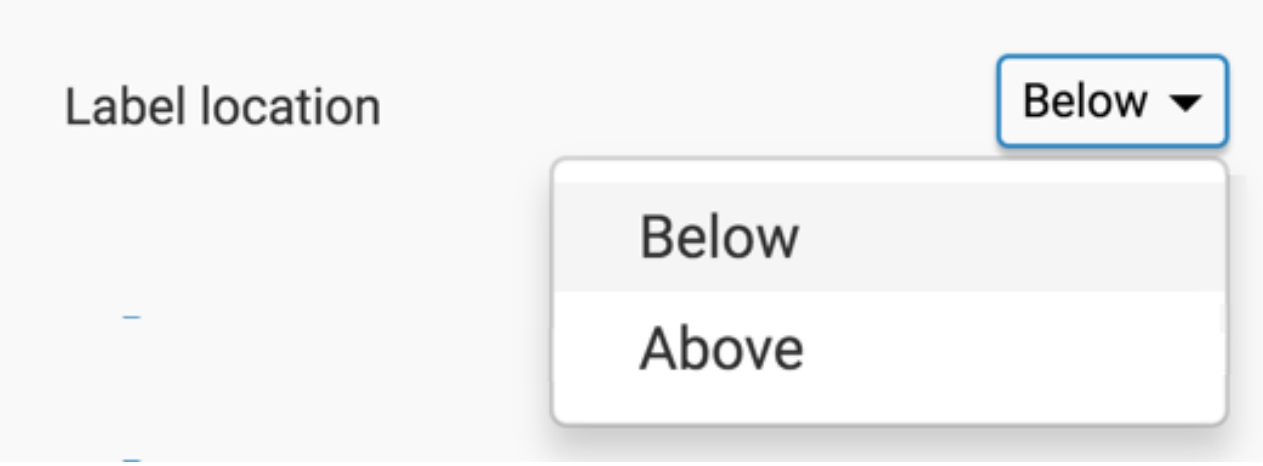

This image illustrates the visual with two options:

- the label indicator appears below the main indicator
- the label indicator appears above the main indicator.

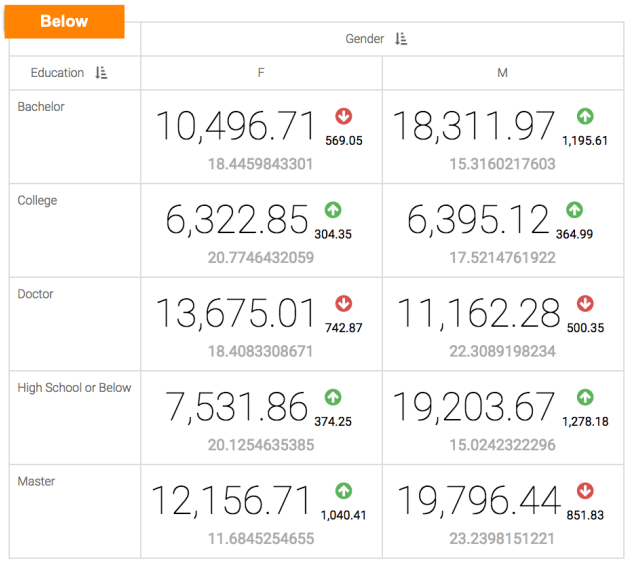

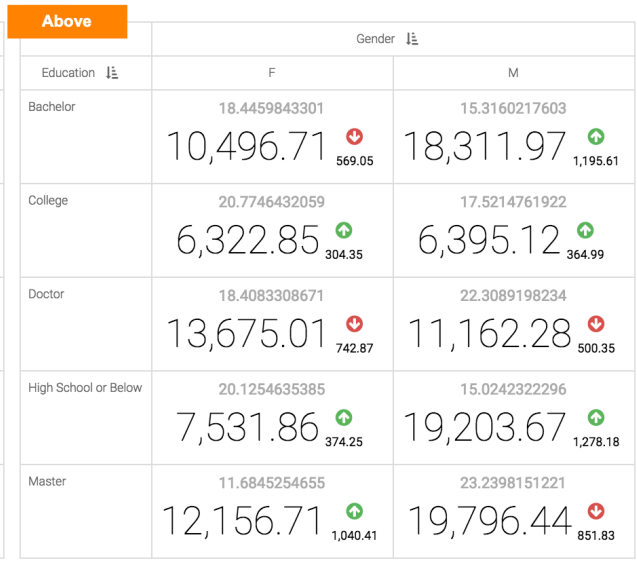

Changing label indicator font size

This setting is specific to KPI visuals.

To change the font size of the label indicator, navigate to the Marks menu, and adjust the selector for the Label font size option. The default value is 14.

This image illustrates the visual with label font size of 14 (default), and 30.

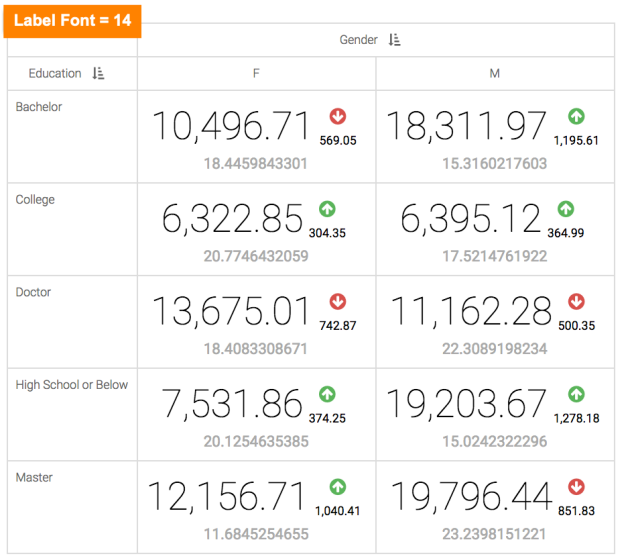

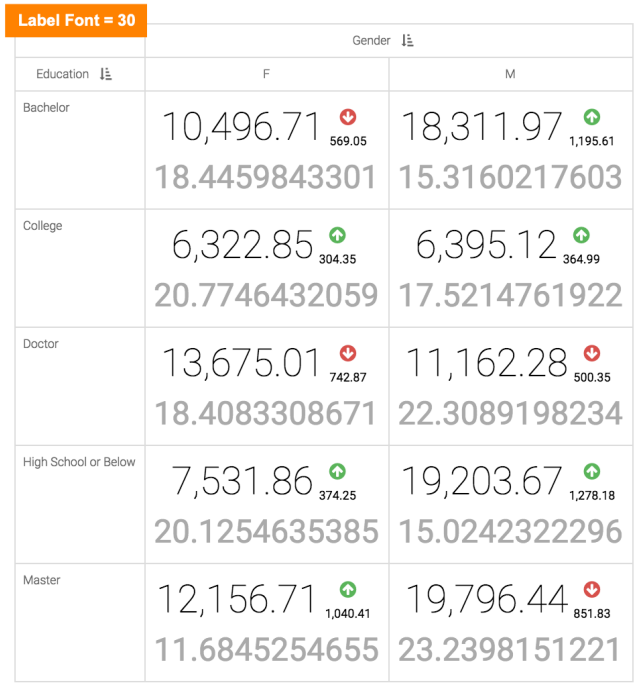

Ignoring negative values

If a visual has both negative and postive values, you have an option not to display negative values. This task applies to stacked Bars and Area visuals.

To ignore negative values in stacked bar and area visuals, navigate to the Marks menu, and select the When both positive and negative measure values are present in stacked bars/area, ignore the negative measure values option.

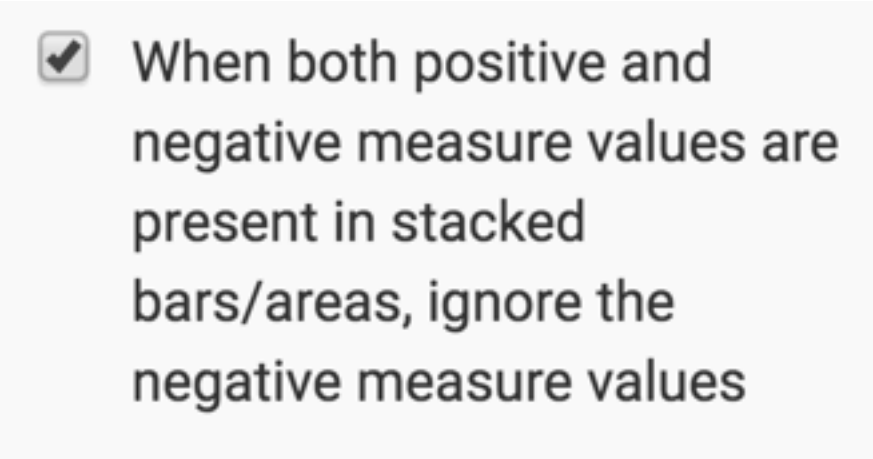

The following image shows the stacked bars without the negative values, when this option is selected.

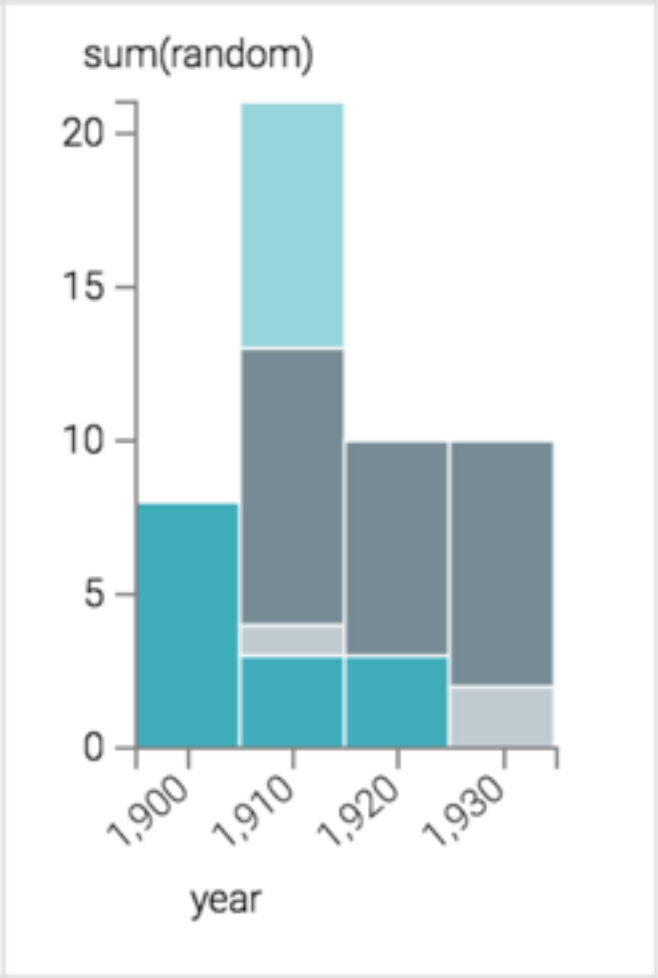

The following image shows the stacked bars with negative values, when this option is not selected.

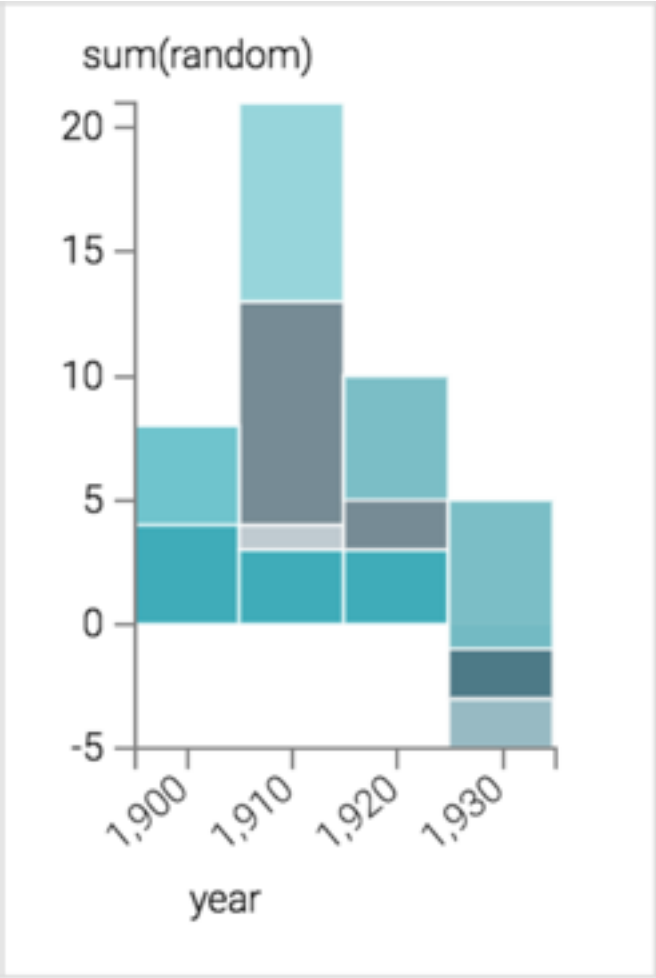

Changing approximate tick count

At times, the ticks are too dense for your visualization needs, or too sparse.

To change the number of ticks from a default value, navigate to the Marks menu, and move the slider in the Approximate tick count option.

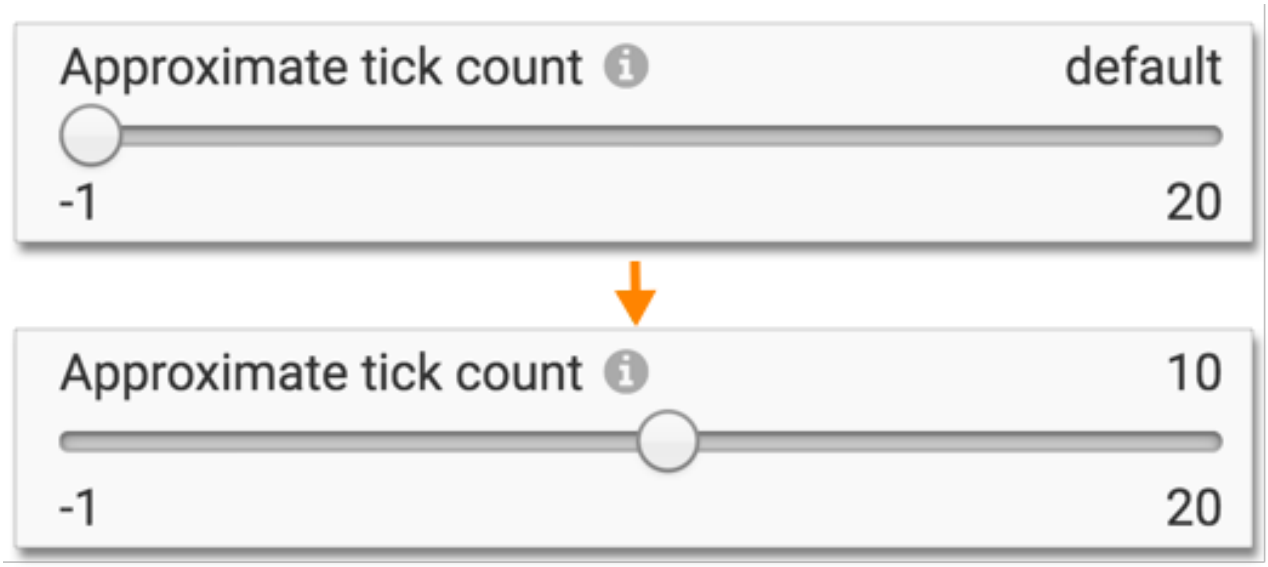

In the following example, compare the default value of ticks on a bullet visual with the appearance of the same visual when the option is set to 5, and then to 15.

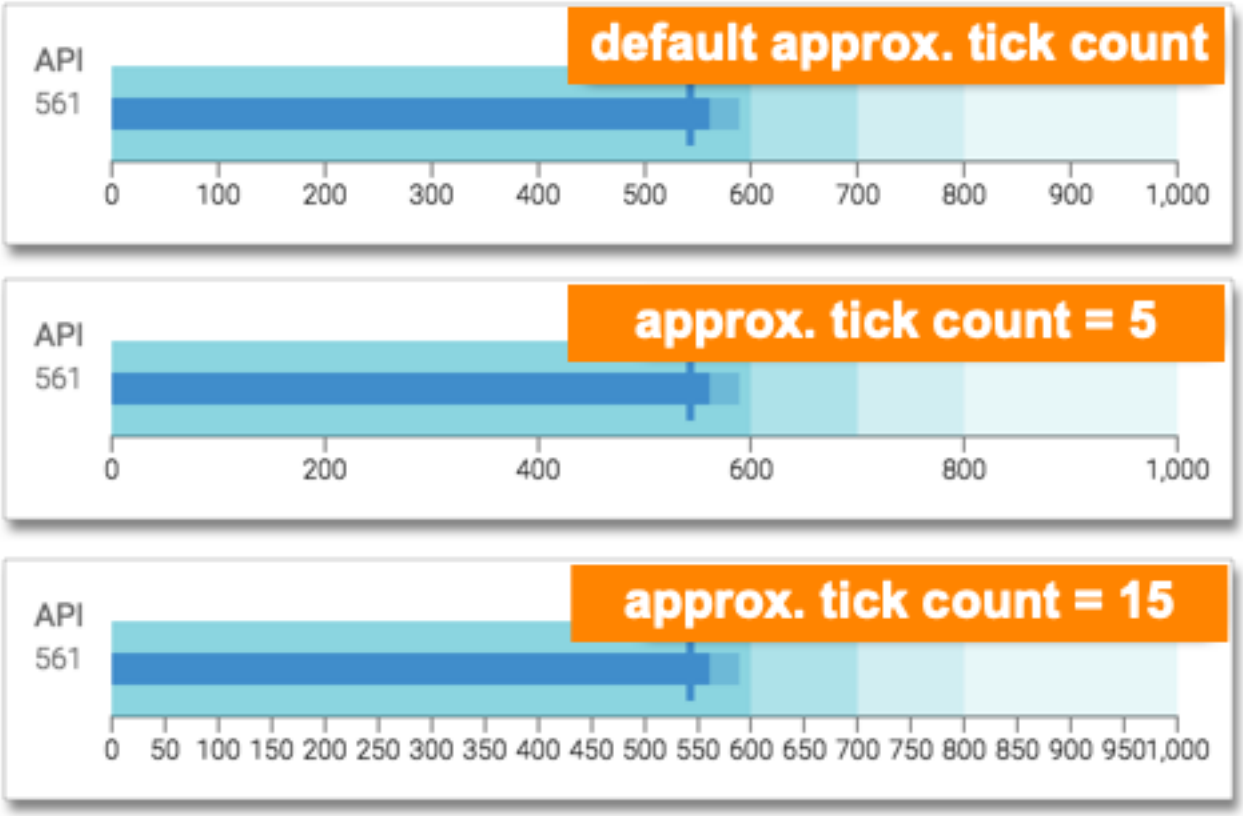

Orienting bars vertically

By default, some visual types, such as bullets, have a natural horizontal orientation.

To change the display to show bullets as vertical bars, navigate to the Marks menu, select the Orient bars vertically option.

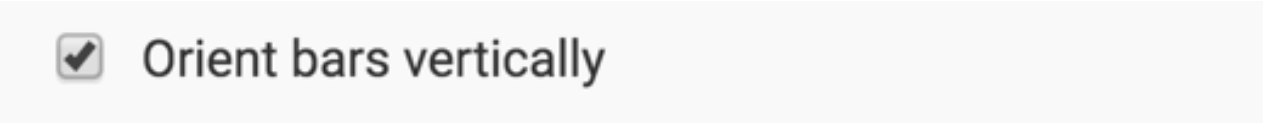

In the following example, compare the default horizontal display of a bullet visual with the appearance of the same visual when displayed vertically.

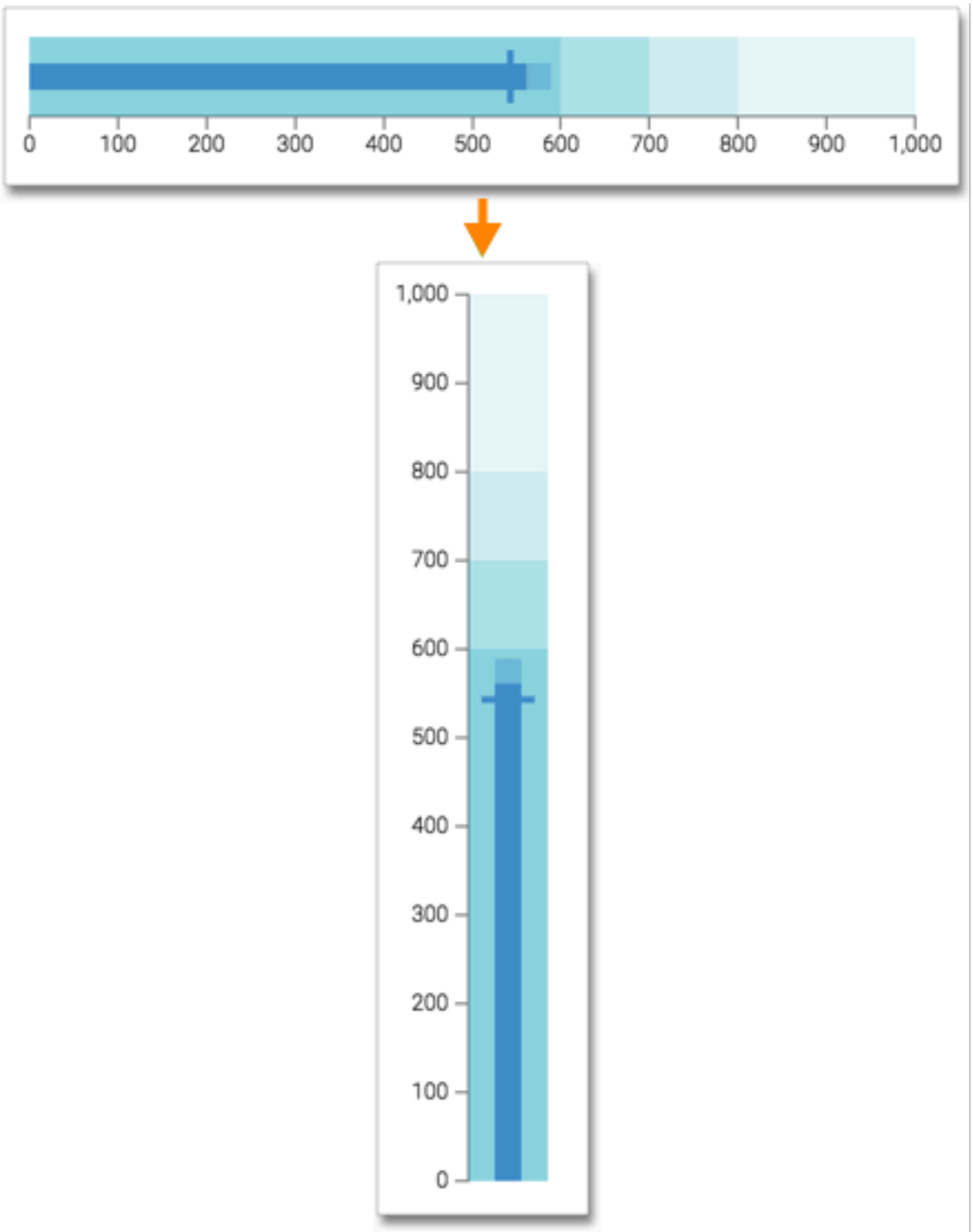

Removing default label

By default, some visual types have default labels.

To remove the default label, navigate to the Marks menu, and select the Remove default label option.

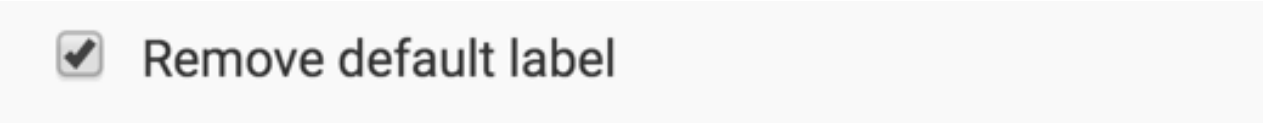

In the following example, compare the visual with a default label, to the one without a label.

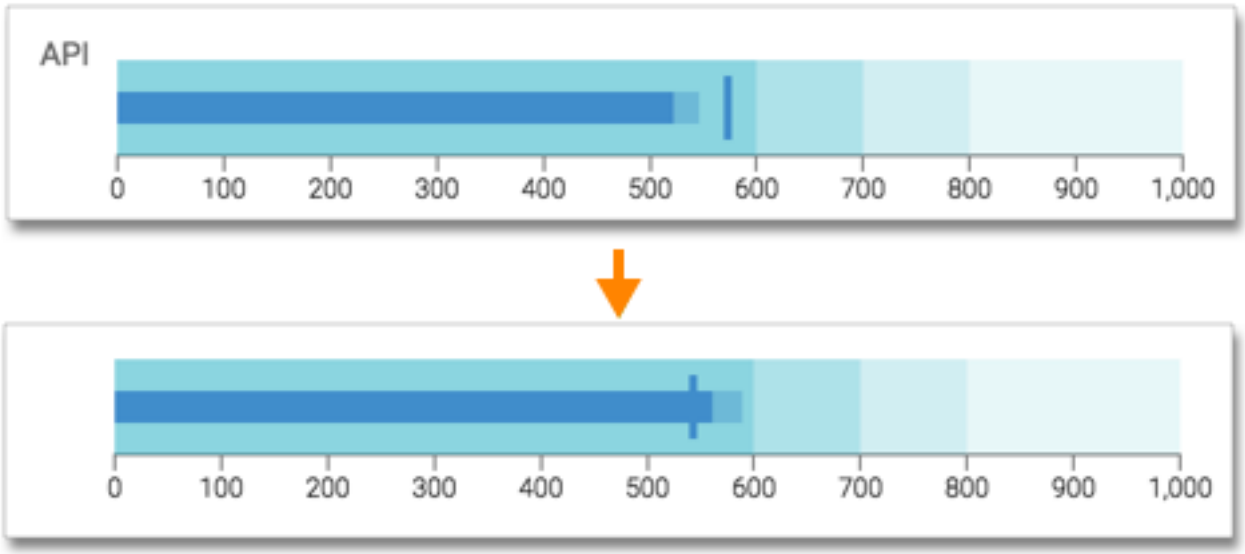

Staying at same map location

About this task

While a visual is streaming, if you manually move or change the zoom level in the map, Cloudera Data Visualization enables you to stay at the same location in the map.

When a visual is set to auto-refresh, it periodically updates the map with new data, and displays a new location on refresh. Now you can stay at the same location in the map, if you manually move or change the zoom level in the map while the visual is streaming.

To stay at the same location in the map, navigate to the Marks menu, and select the Do not auto-update map location option. To automatically update the location of the map, de-select this option.

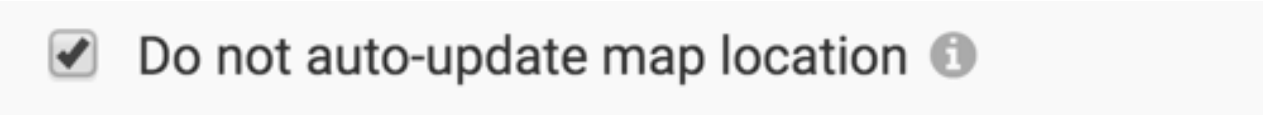

To demonstrate this feature, let us create a dashboard with two visuals based on the dataset Food Stores Inspection in NYC.

The following image displays two visuals and a custom filter, city, with All values. The map visual shows the heatmap of New York, and the table visual lists all the cities in the state of New York.

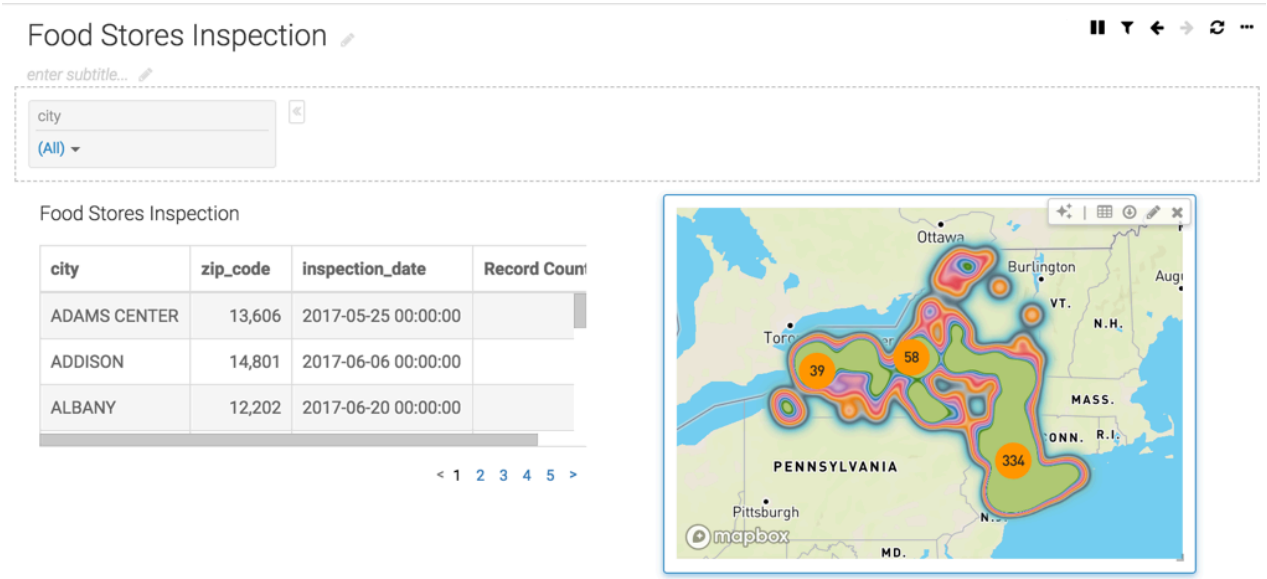

While the visual is streaming, let us zoom in to see the JFK airport in the map. Then select Albany in the filter. In the following image, notice that the table visual refreshes to show only the Albany data. However, the map visual still shows the same location where we zoomed in, the JFK airport.

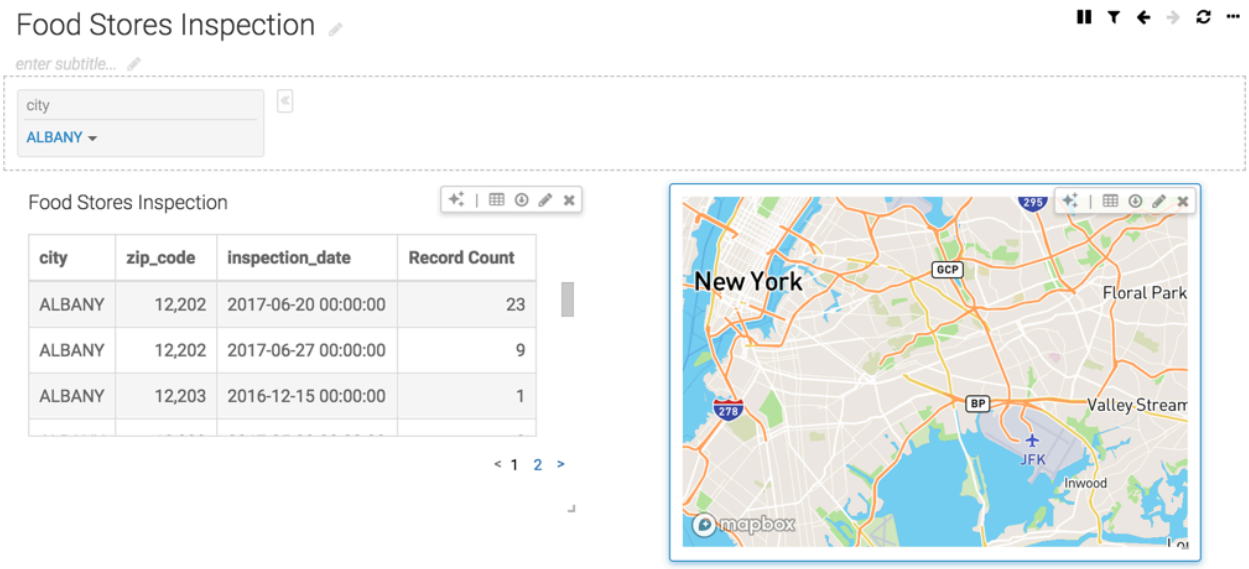

To update the location of the map while streaming, let us de-select the Do not update map location option. In the following image, notice that the map visual now updates to show the filter value, Albany and not JFK airport, where we zoomed in while streaming.

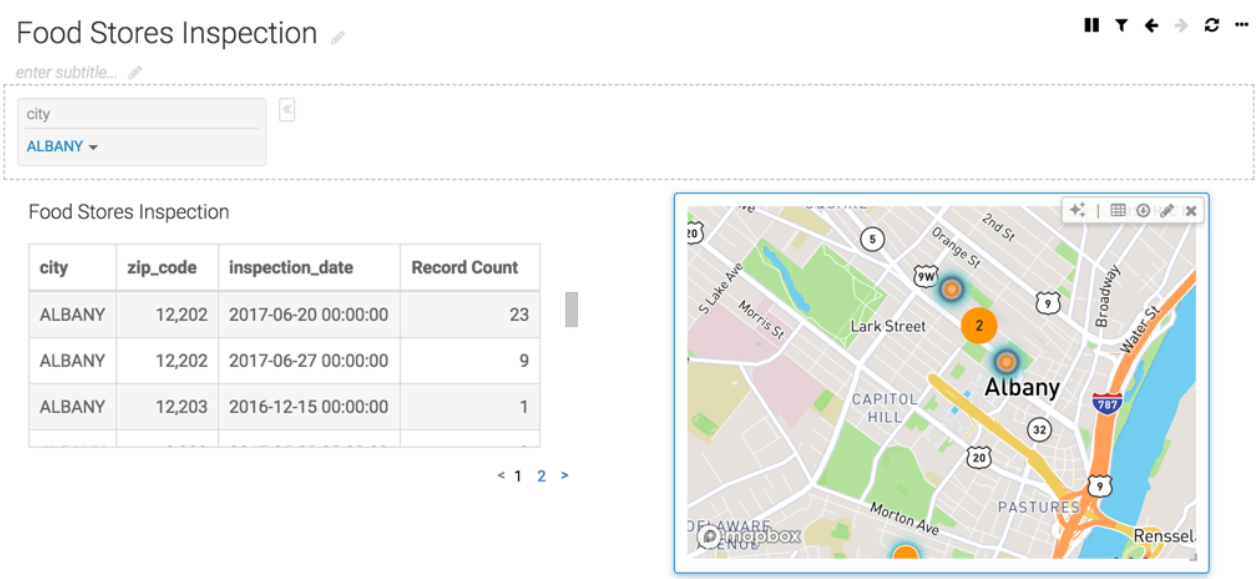

Setting minimum cell size

About this task

In Correlation Heatmap visuals, Cloudera Data Visualization enables you to change the minimum size of the component cells of the visual.

This setting is only available on Correlation Heatmap visuals.

To change the size of a cell in a visual, navigate to the Marks menu, and change the value of the Min Cell Size option. The default is 13 px, and the range is from 1 px to 500 px.

In the following example, compare the appearance of the same visual when the minimum cell size is 15, and when it is 25.

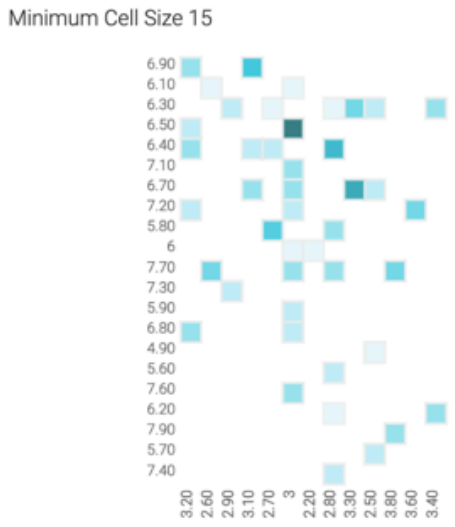

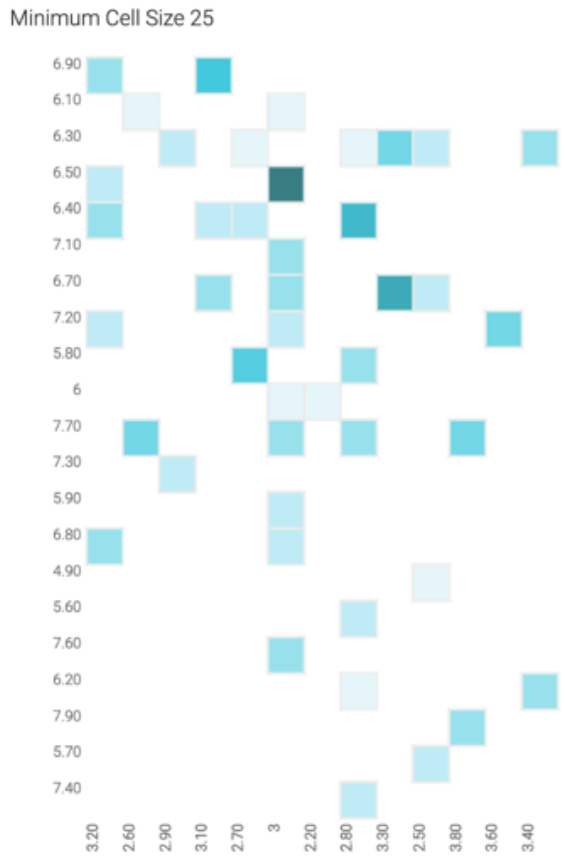

Customizing nodes

Several display options may be managed from the Nodes menu, depending on the chart type in the visual. The Nodes interface has the following options:

Remembering node positions

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

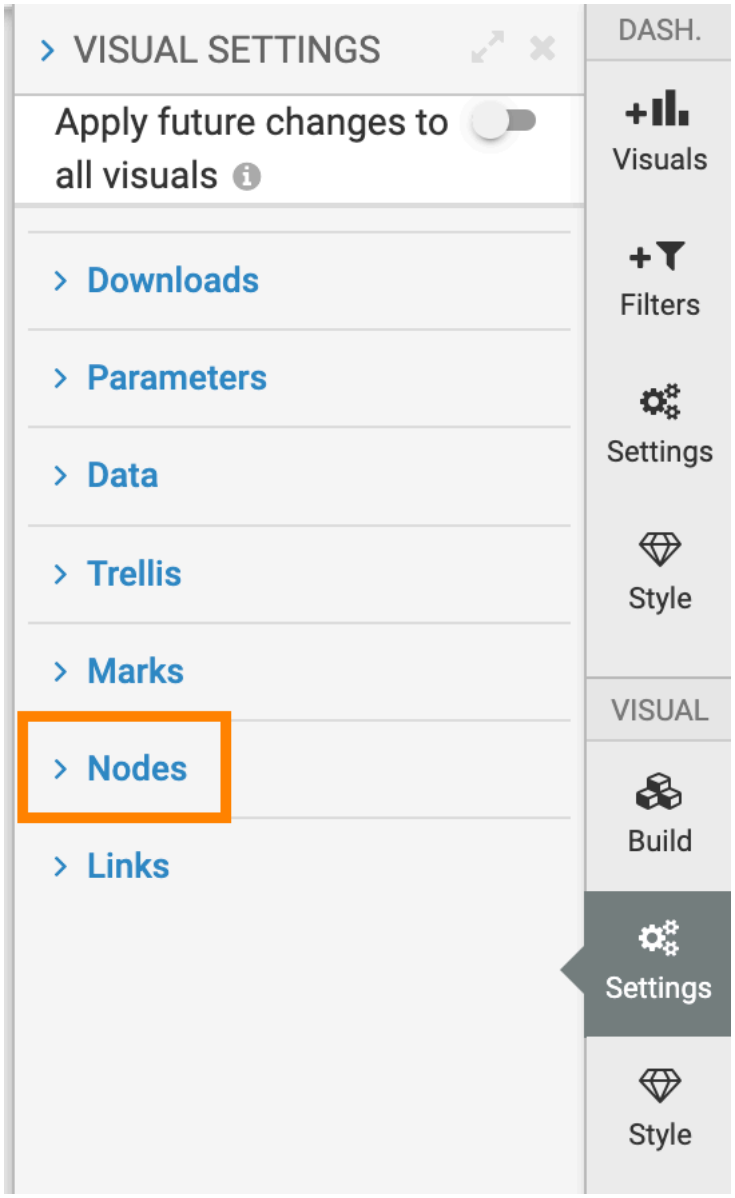

3. Select the Remember the positions of fixed nodes option.

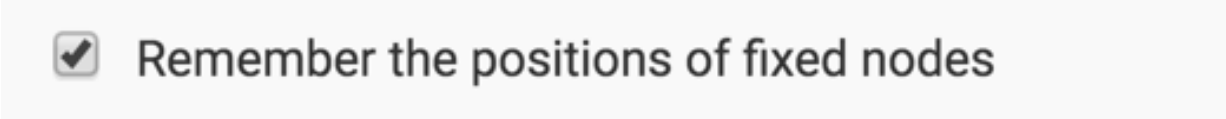

Showing the same source and target values

About this task

This task applies to Network visuals.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

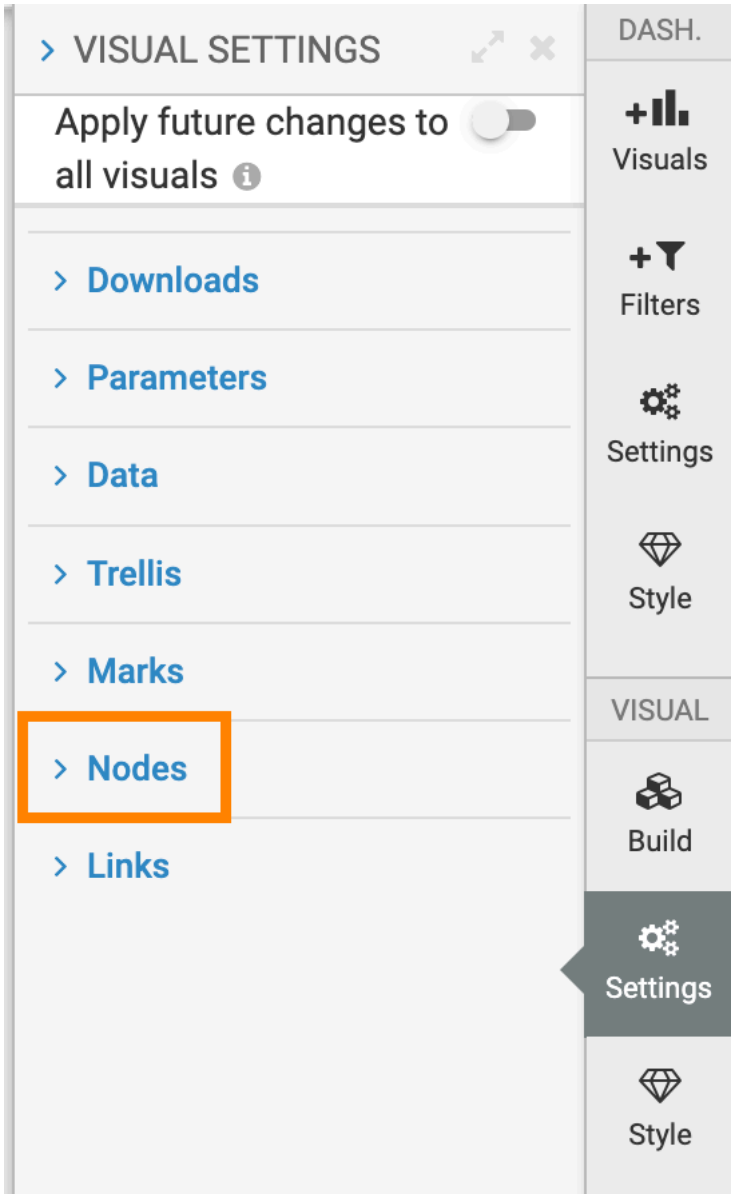

3. To ensure that the end points are defined by the same source and destination values, select the Source & target node data come from the same base values option.

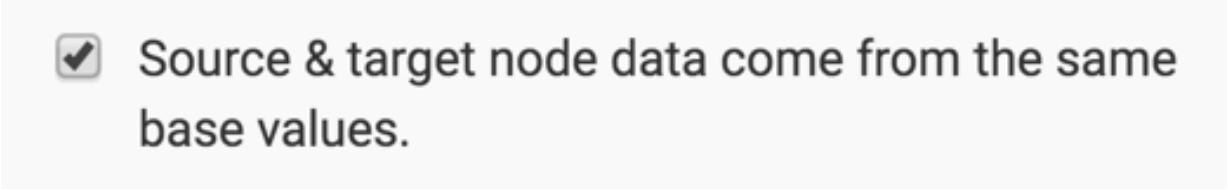

Changing source node mark

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

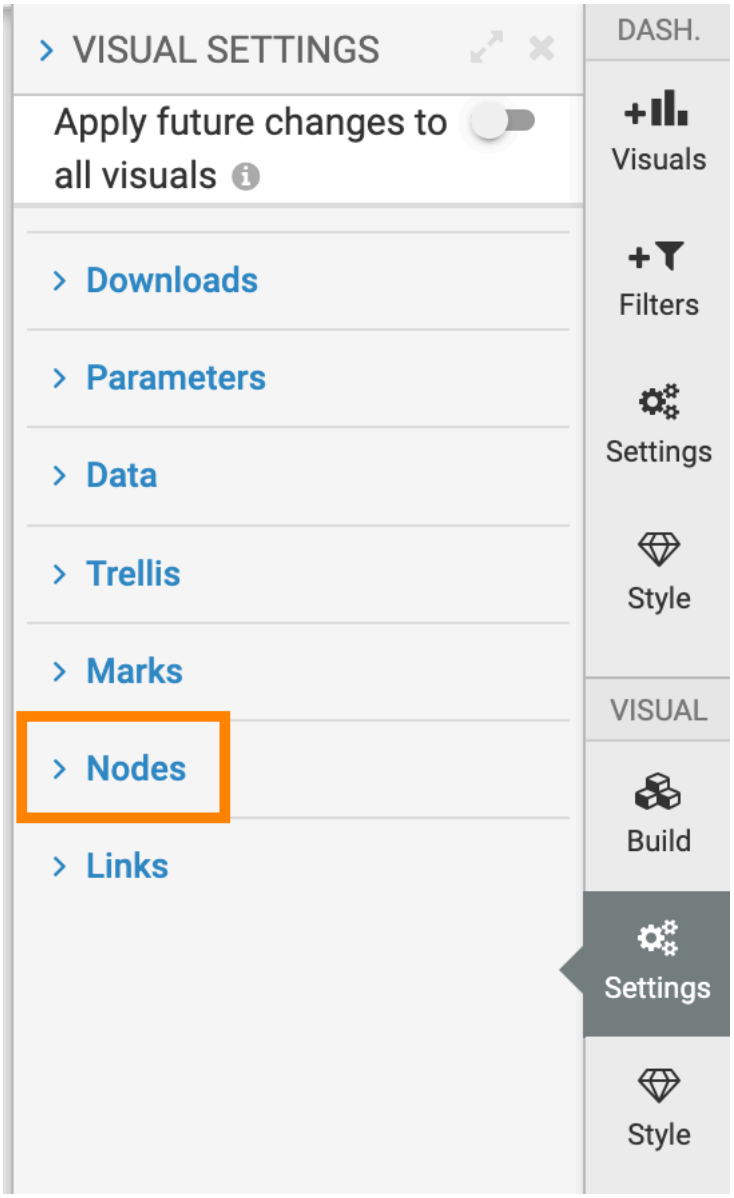

3. To change the mark of the source node, change the selection in the Source Node Mark menu.

   The options are:

   - circle (default)
   - square
   - diamond
   - cross
   - triangle-up
   - triangle-down

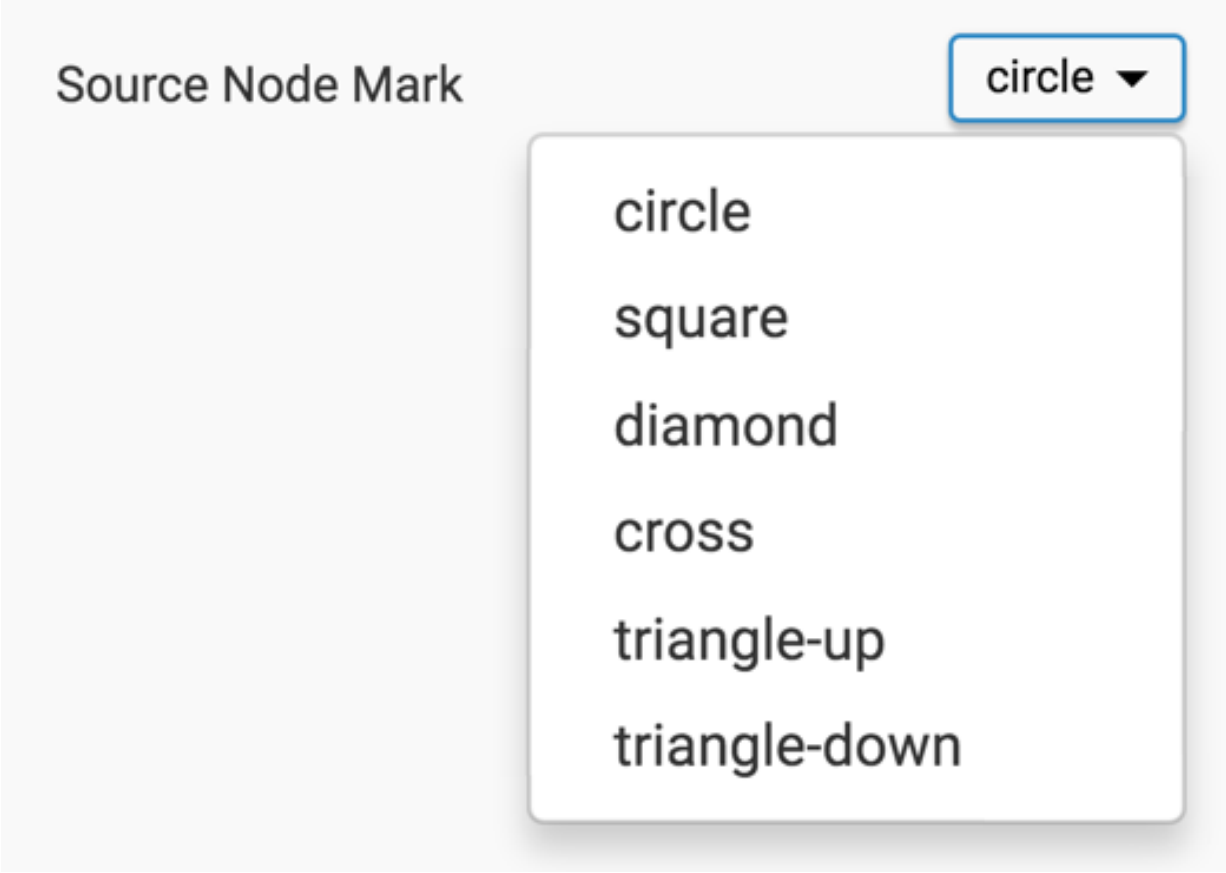

Compare the appearance of a visual with source node circle mark, and source node square mark.

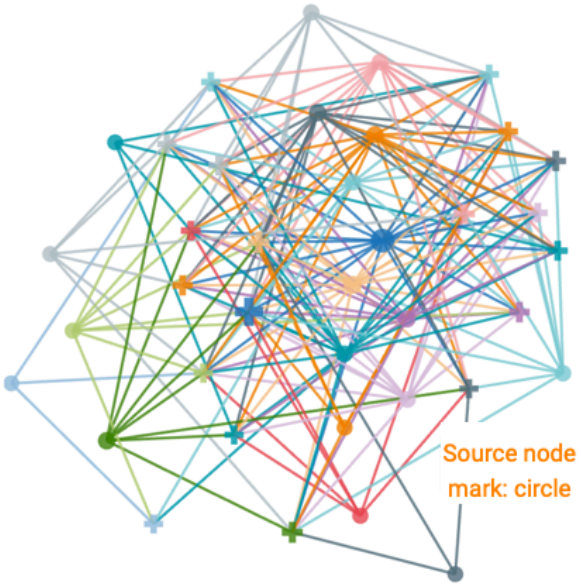

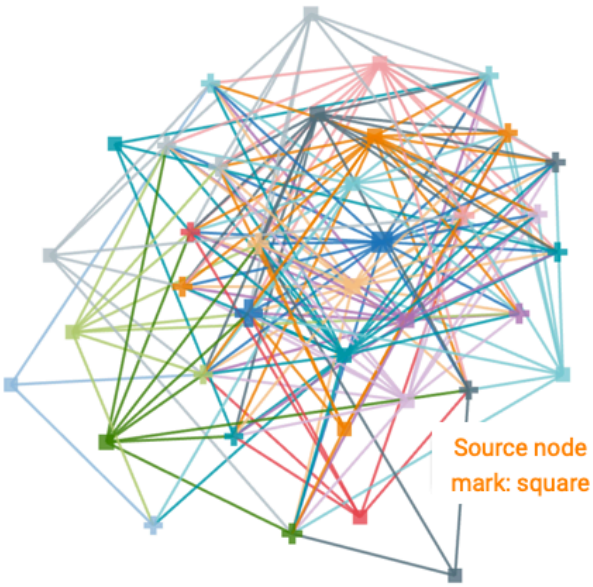

Changing target node mark

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

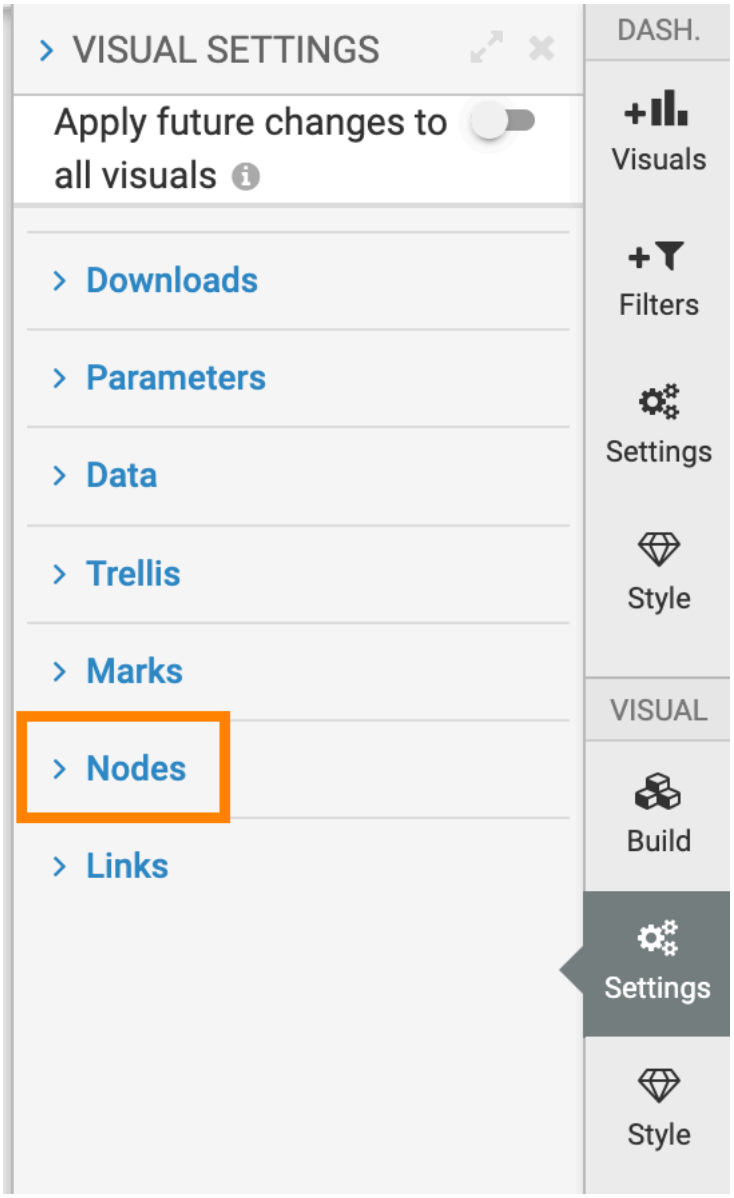

3. To change the mark of the target node, change the selection in the Target Node Mark menu.

   These options are:

   - circle (default)
   - square
   - diamond
   - cross
   - triangle-up
   - triangle-down

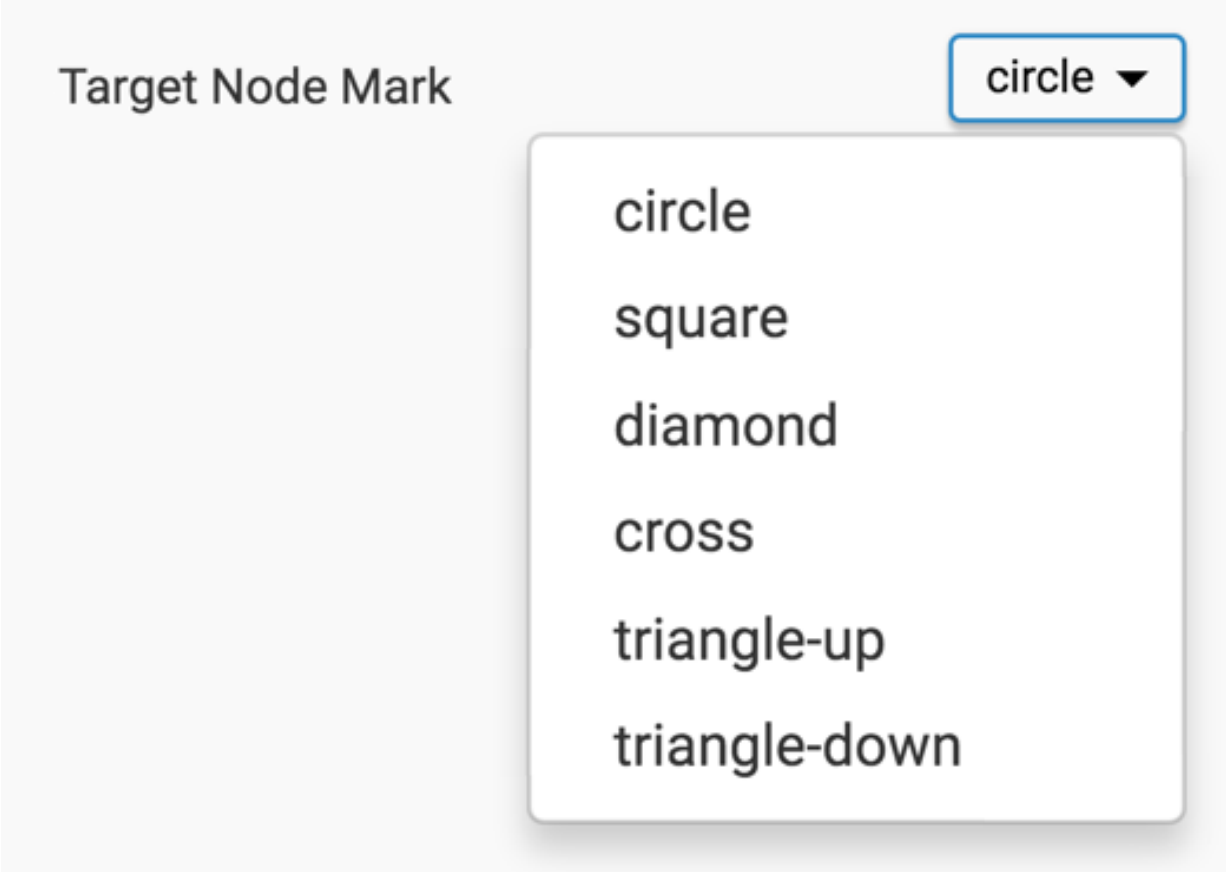

Compare the appearance of a visual with target node cross mark, and target node diamond mark.

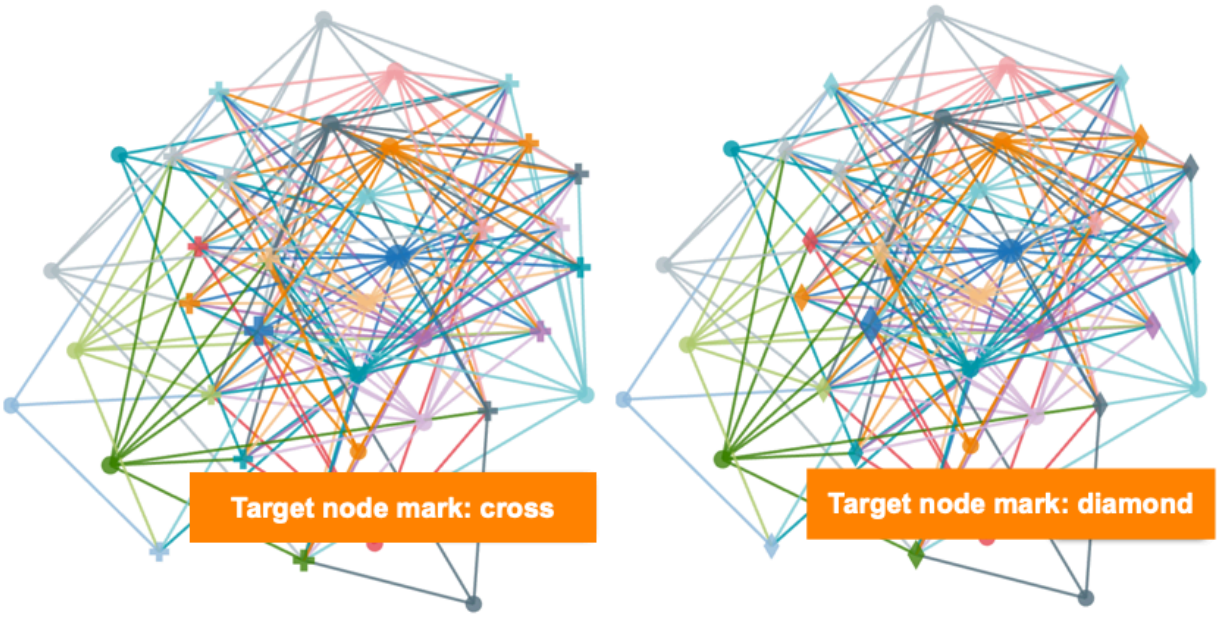

Changing combined node mark

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

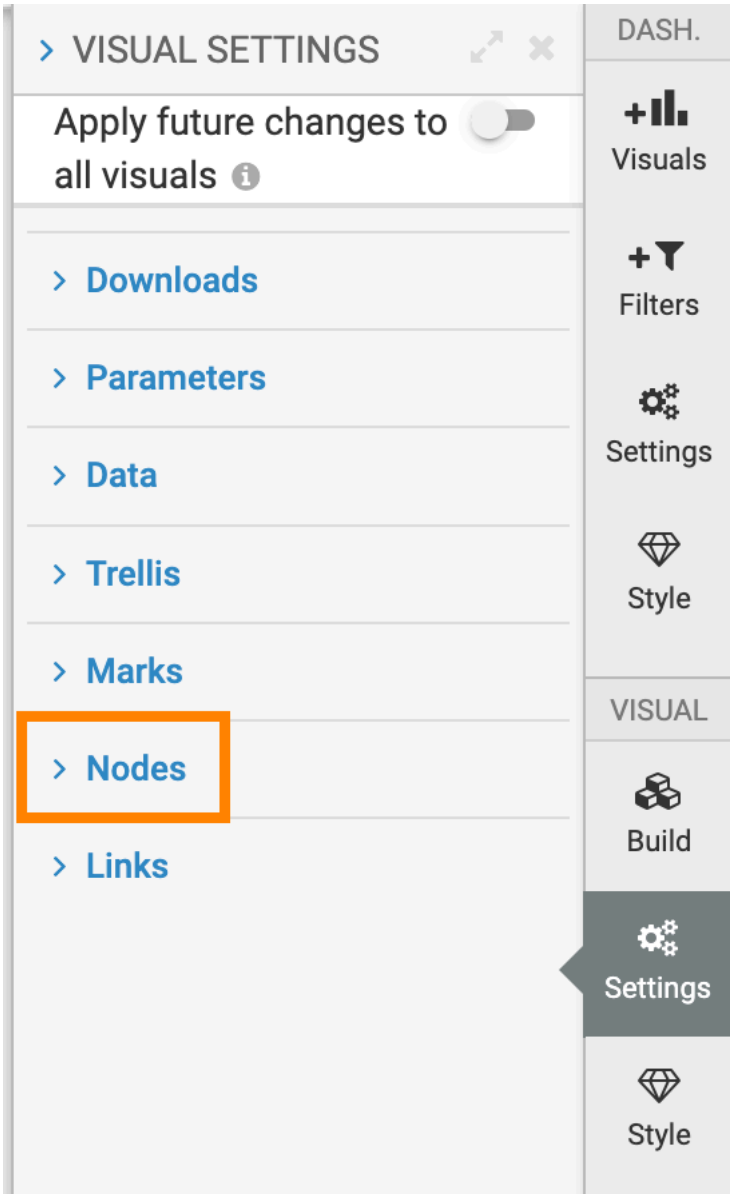

3. To change the mark that applies to both the source and target node, change the selection in the Combined Node Mark menu.

   These options are:

   - circle (default)
   - square
   - diamond
   - cross
   - triangle-up
   - triangle-down

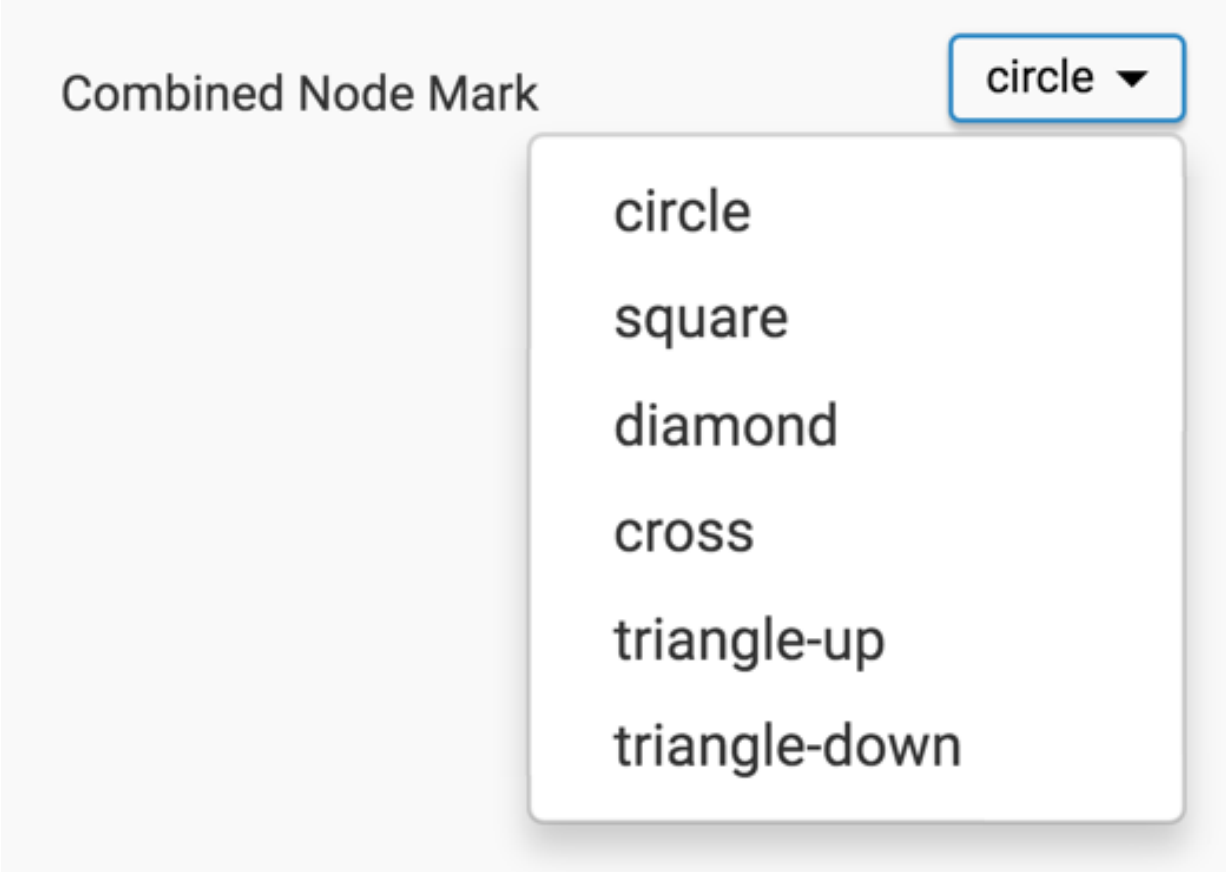

Changing node size range

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

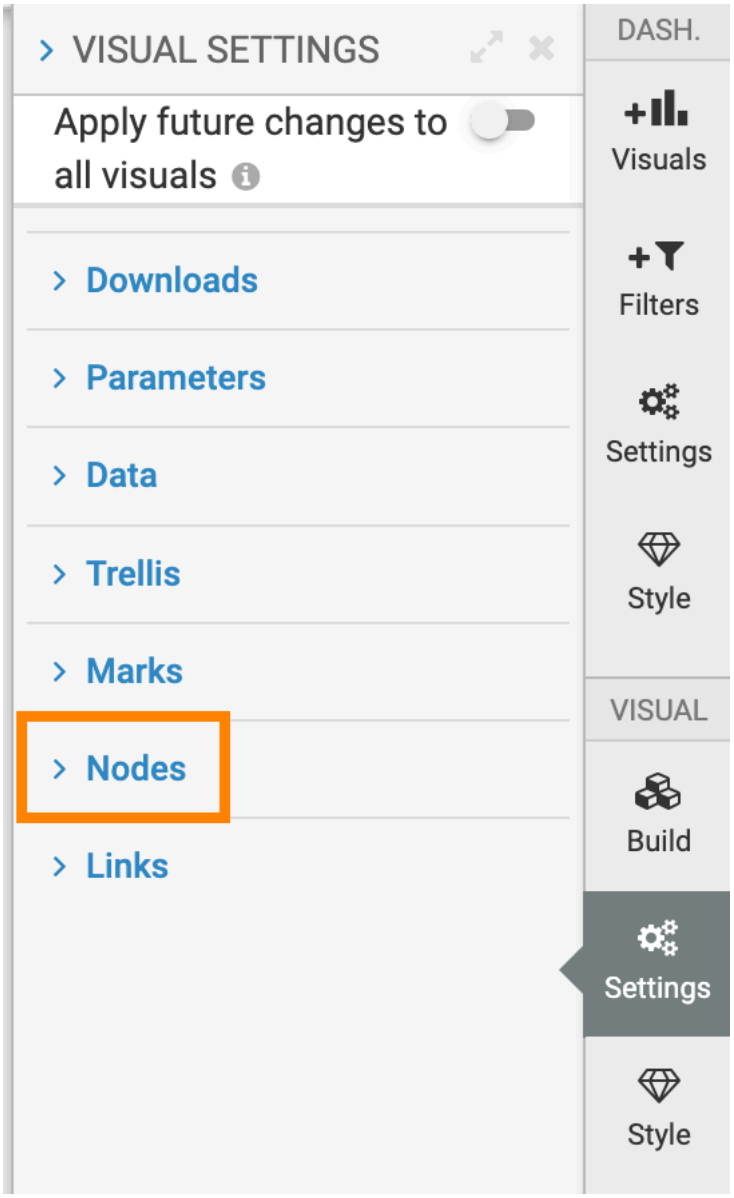

3. To change the width of the link, change the minimum and the maximum in the Node size range option.

   The default setting is 10-25.

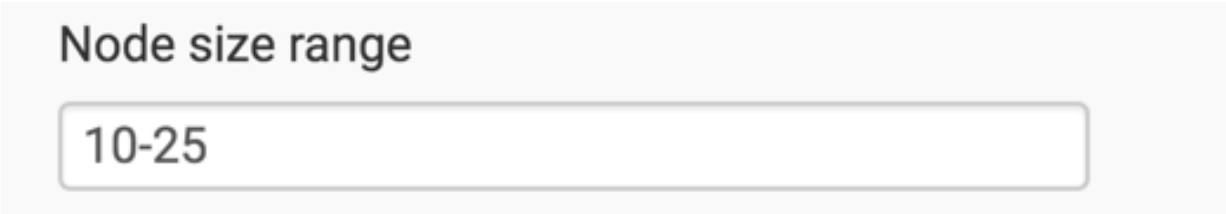

   Compare the appearance of a visual with node size range of 10-15 (on the left) with a visual that uses a node size range of 2-30 (on the right).

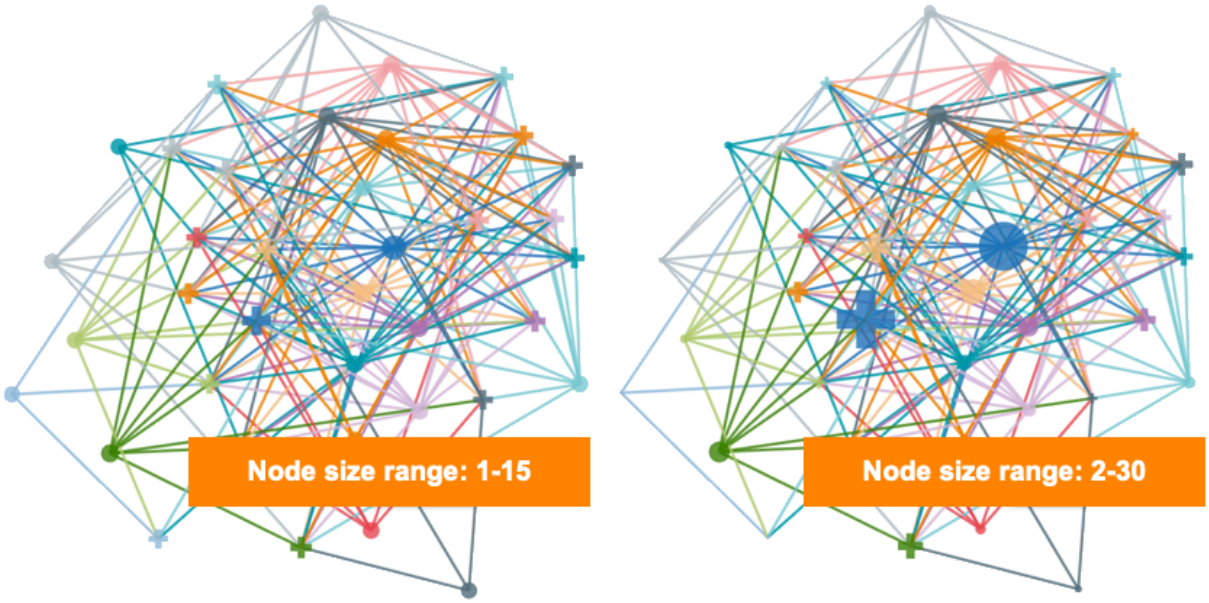

Changing node friction

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

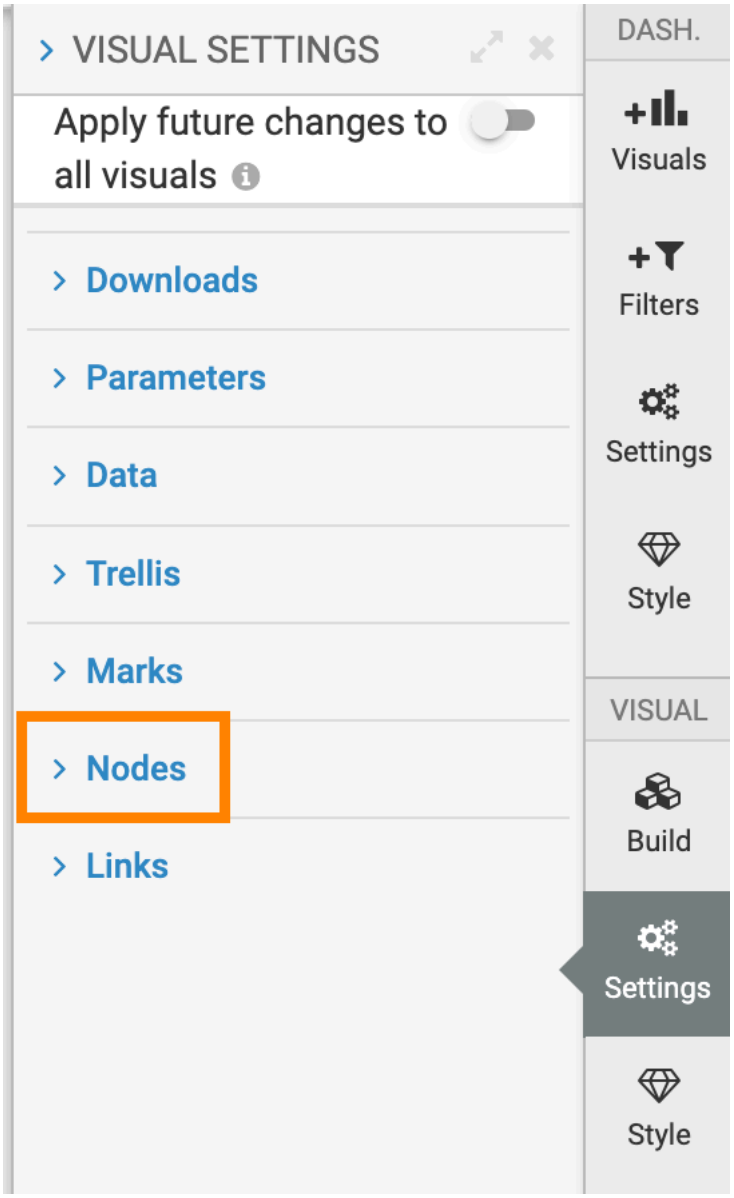

3. To change the friction of a node, change the value of the Node friction option.
   The default setting is 0.3, and the valid range is 0 to 1.

Compare the appearance of a visuals with node friction of 0.09, 0.5, and 0.9.

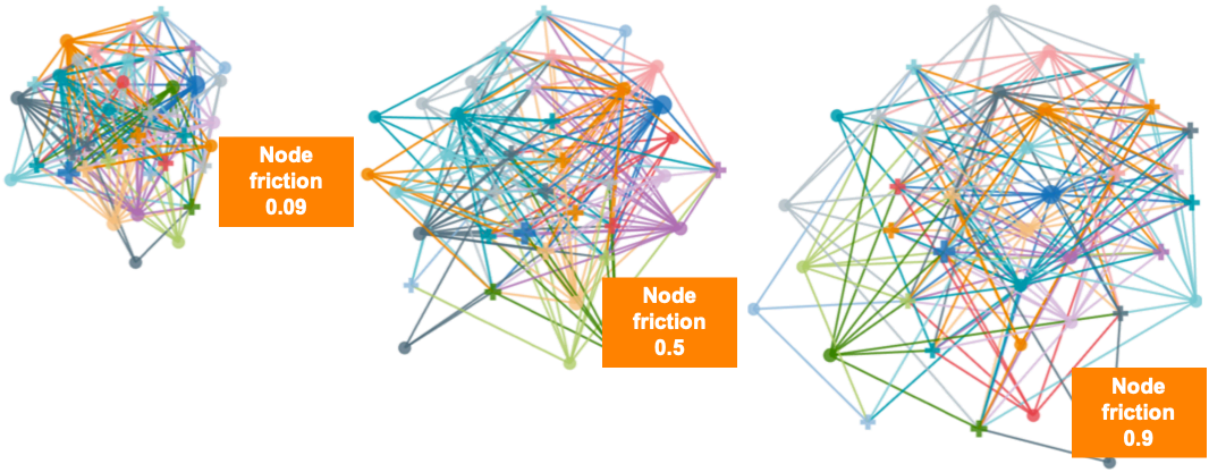

Changing node charge

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

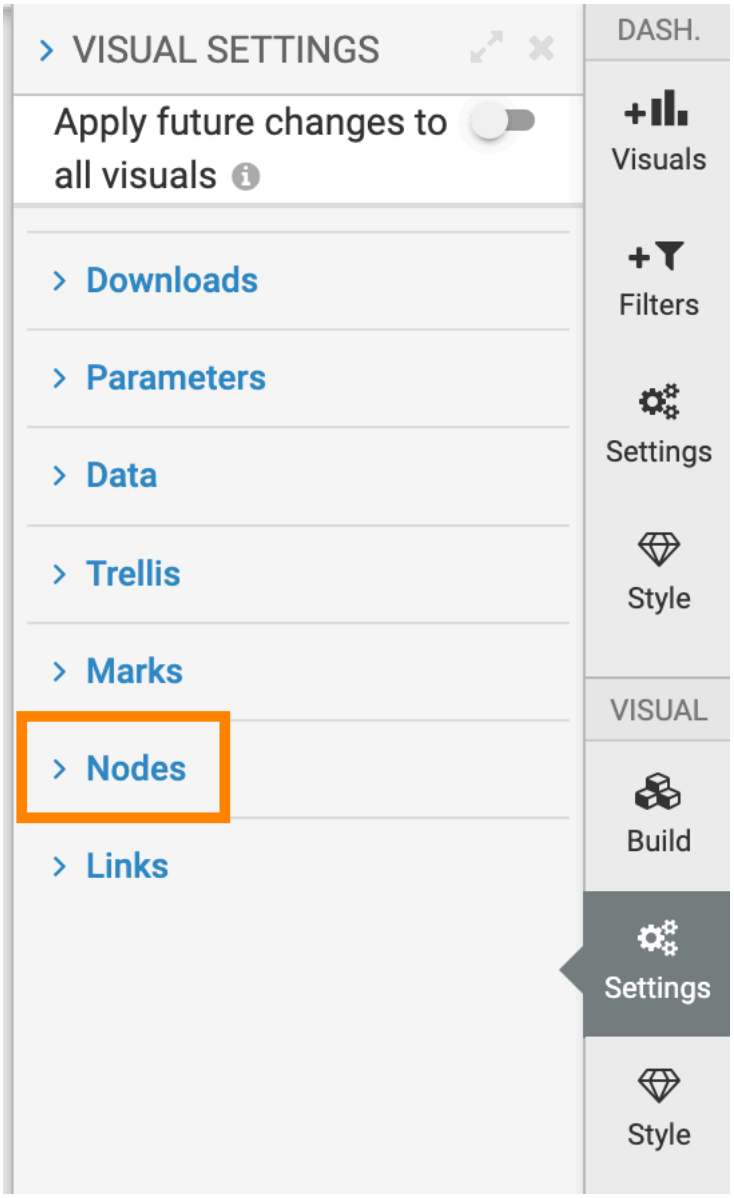

3. To change the charge of a node, adjust the selector for the Node charge option.
   The default setting is -1000. The valid range is -3000 to 0.

Compare the appearance of visuals with node charge of -100, -300, and -500.

Changing node opacity

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

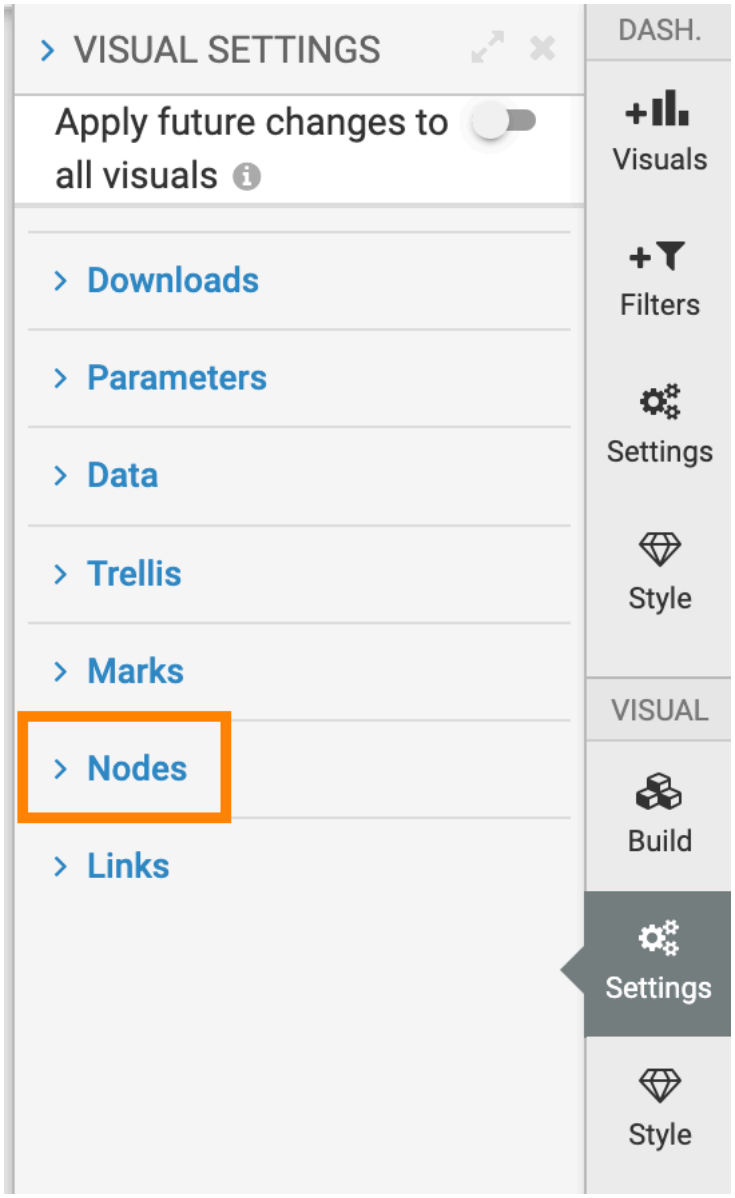

3. To change the opacity (color saturation) of nodes, adjust the slider for the Node Opacity option. The default value is 80, and the valid range is between 0 and 100.

Showing node labels

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

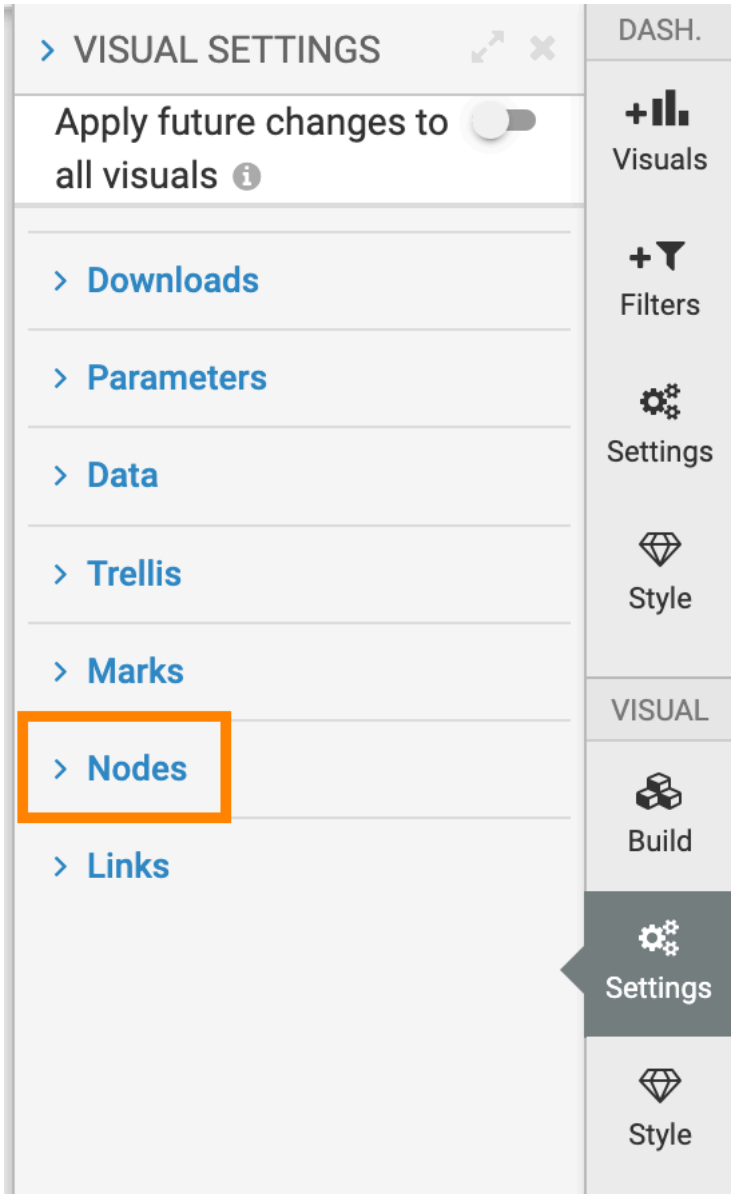

3. To change the opacity (color saturation) of nodes, select the Show node labels option.

Adding nodes legend

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

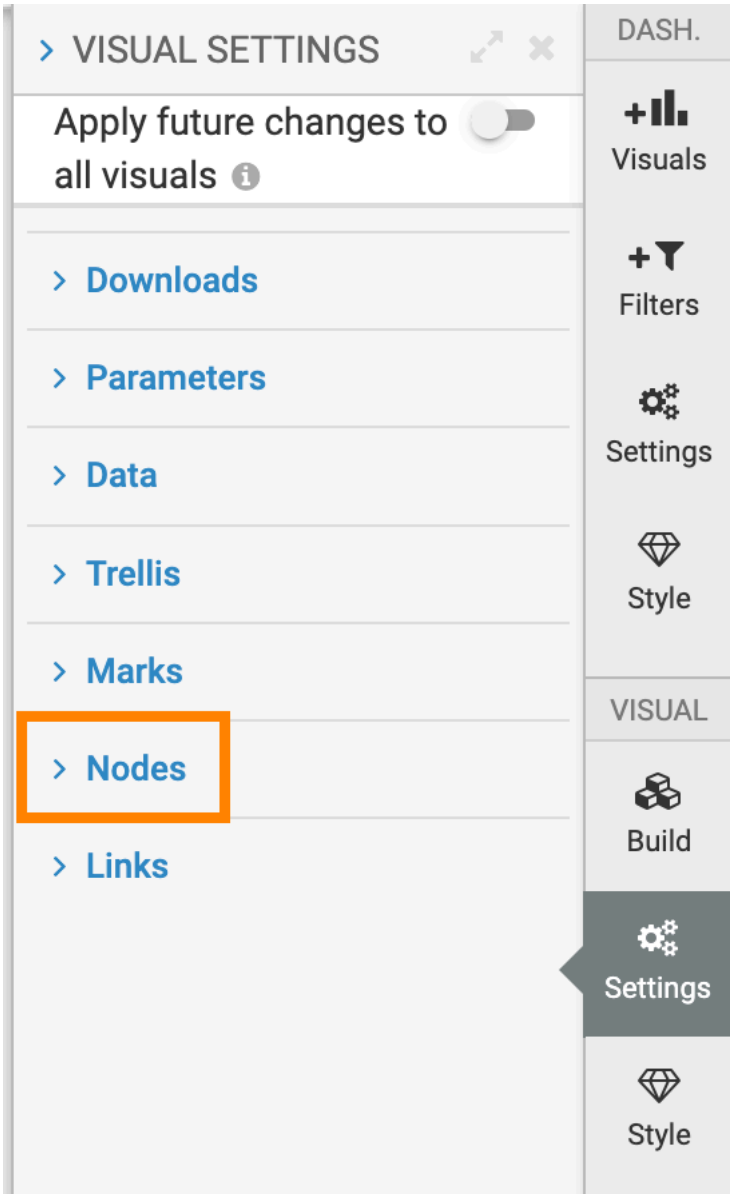

3. To add a legend for links, select the Add node legend option.

   After you select this option, you can change the Legend Title value. This value is Node colors by default.

   In the following example, we added a legend to the nodes of a network visual.

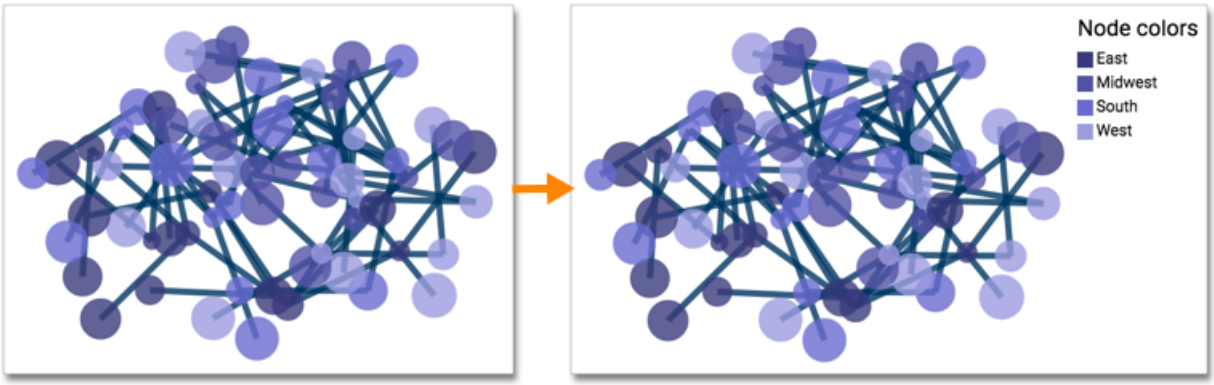

Changing nodes legend name

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Nodes.

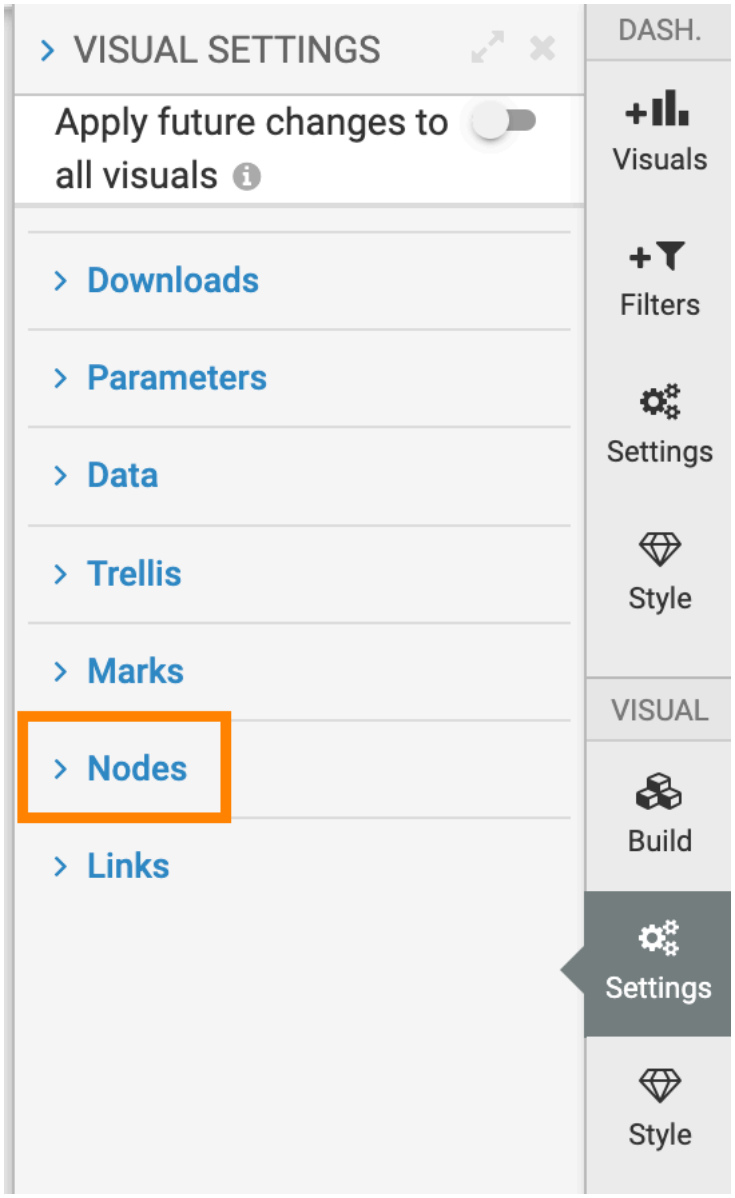

3. To change the name of the nodes legend, change the text in the Legend Title option.

   The default value is Link colors.

   Note that you can change this value only after selecting the Add link legend option, as described in *Adding the Nodes Legend*.

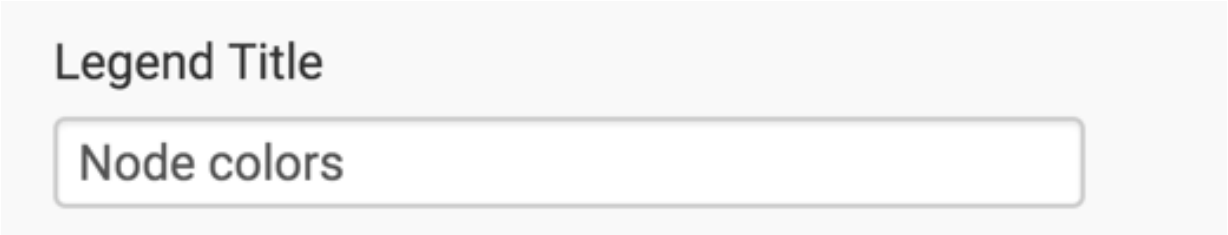

Customizing parameters

In the Parameters settings of Cloudera Data Visualization, you can do any of the following:

Using parameters with explicit scope

About this task

This setting protects the visual from being changed by dataset scope filters.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Parameters.

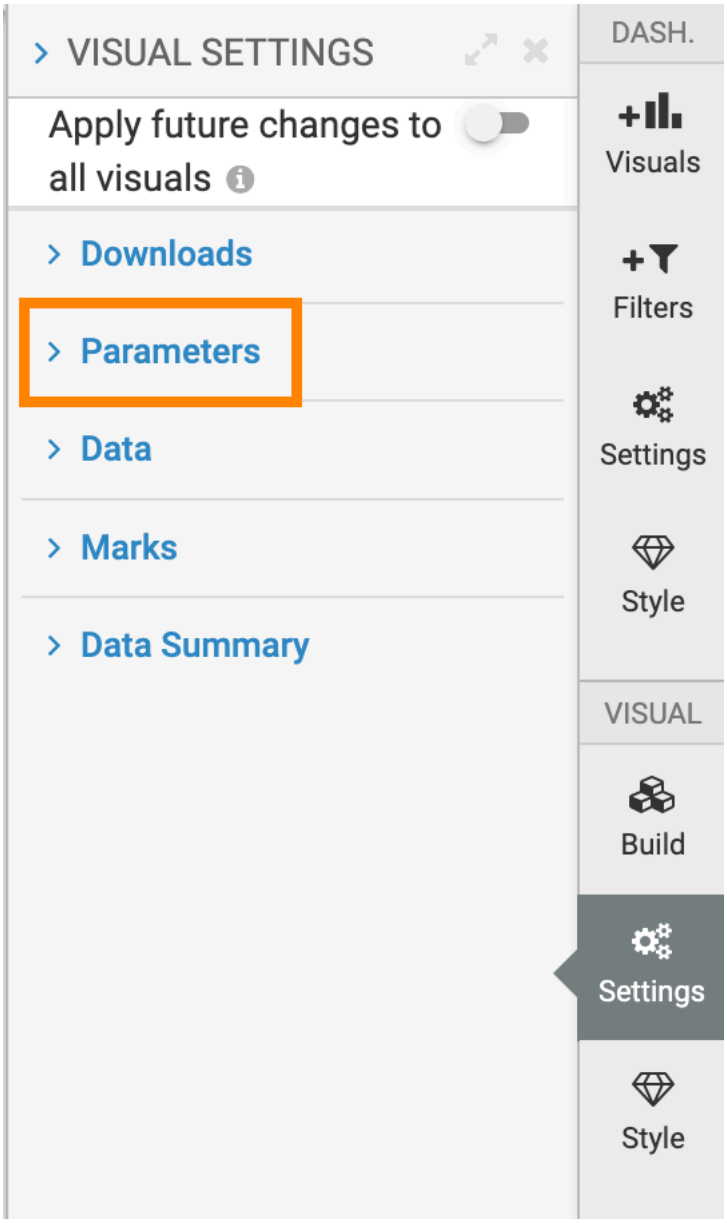

3. To set the explicit scope for a specific visual, select Receive parameters with explicit scope.

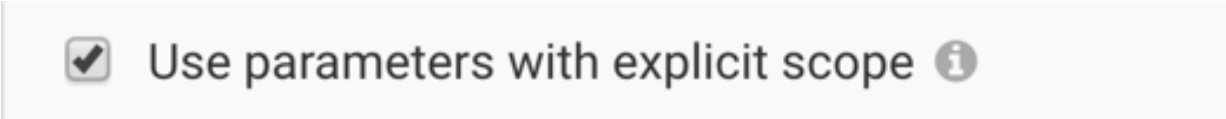

Sending parameters with dataset scope

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Parameters.

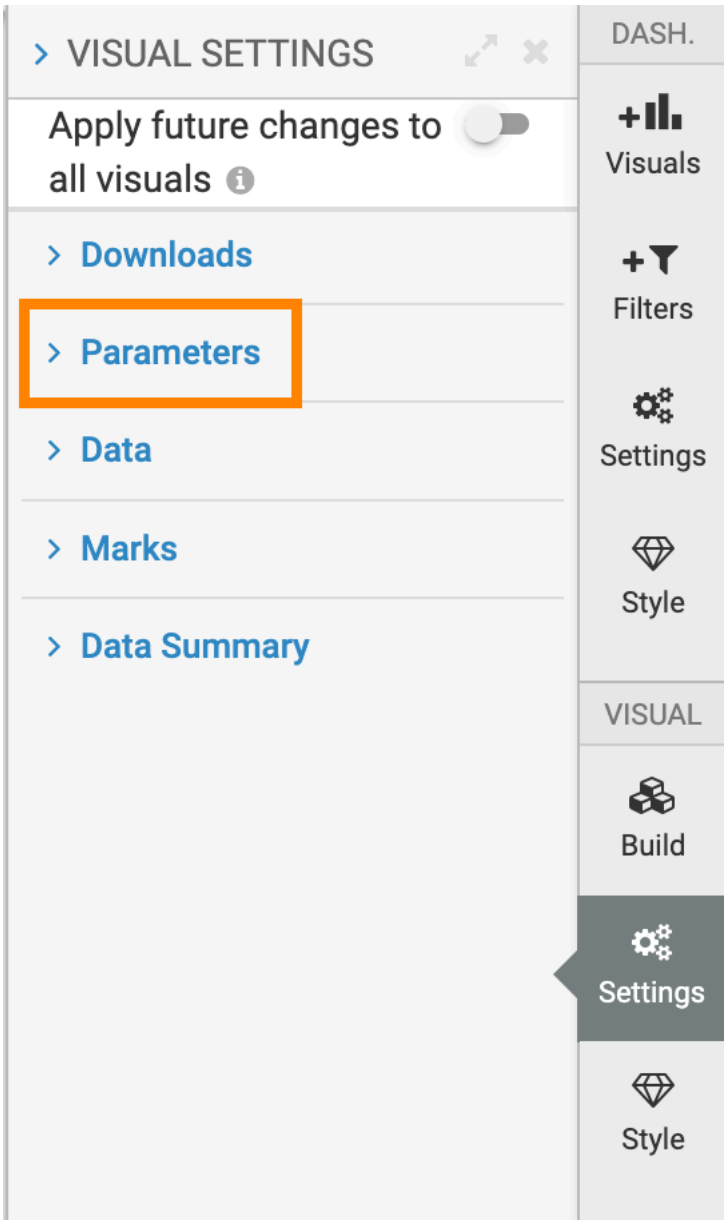

3. To ensure that the parameters emitted by this visual are available to all visuals in the dashboard that use the same dataset, select Send parameters with dataset scope.

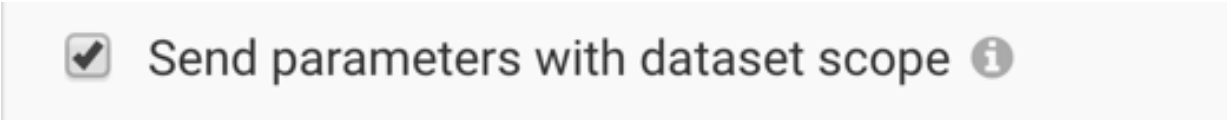

Sending aggregates as parameters

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Parameters.

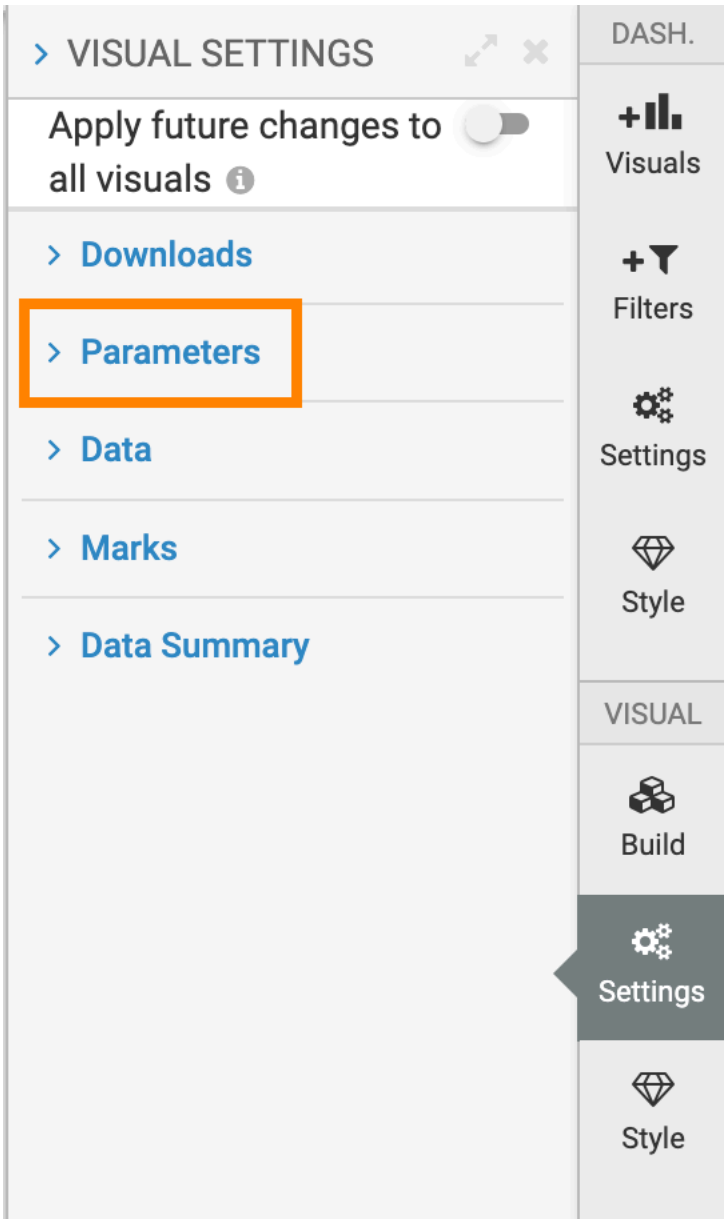

3. To ensure that the visual emits parameters that contain aggregate values in addition to dimensions, select Send Aggregates as Parameters.

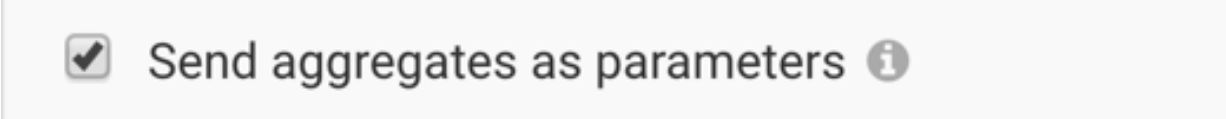

Customizing map server

In the Map Server menu, you can select which digital map service to use for interactive maps.

Customizing map server

Cloudera Data Visualization provides two digital map services: Google Maps and Mapbox.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Map Server.

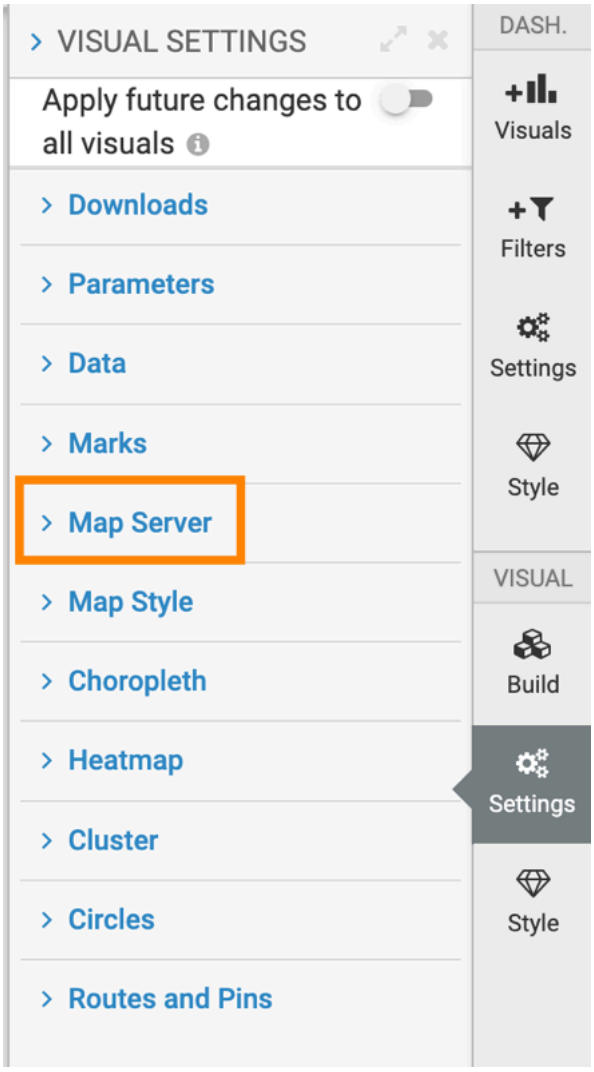

3. To change the map server, change the selection in the Map Server option.

   The options are Google Maps and Mapbox. Note that Mapbox uses the Mapbox GL library.

   Important: The administrator must obtain a Premium Google Maps API key or a Client ID from Google Maps API or a Mapbox Token, and the Mapbox Style ID from Mapbox. Then specify this key in the Site SettingsAPI Keys for Interactive Map Visual option.

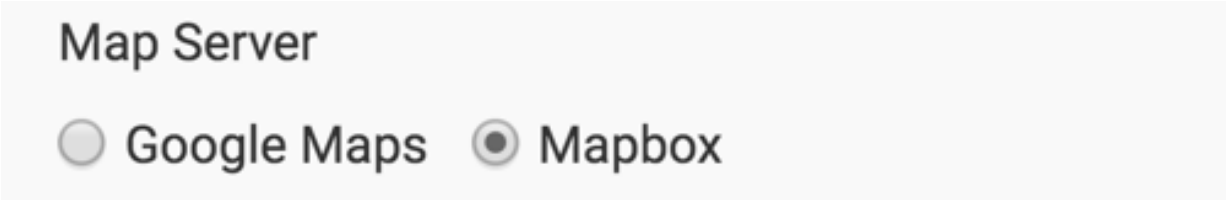

Specifying mapbox key for a visual

You can override the Mapbox token in the Site Settings interface by specifying a Mapbox token at the visual level.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Map Server.

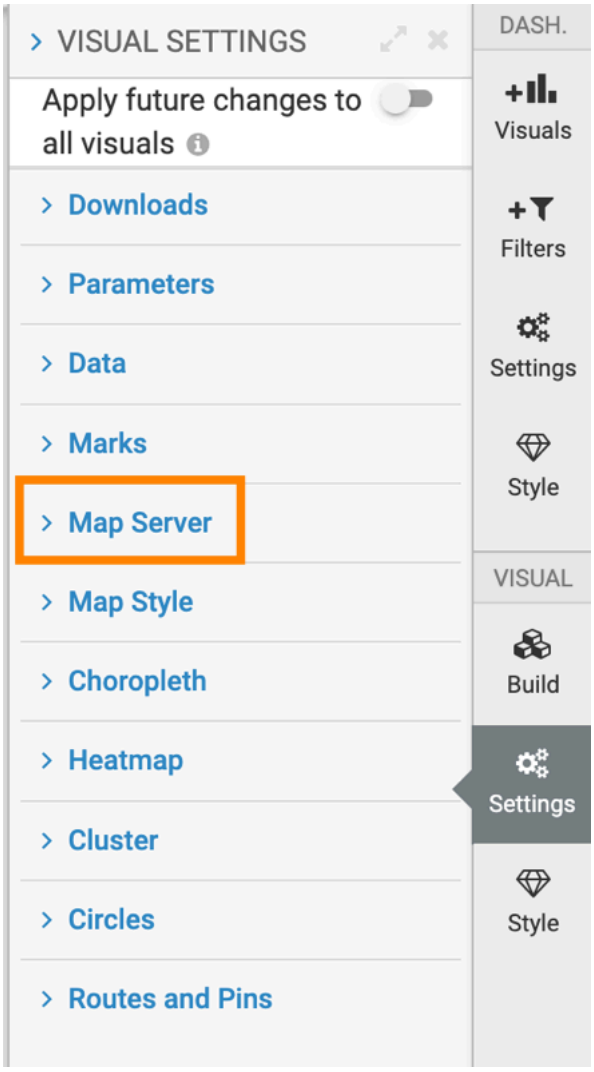

3. Enter a mapbox key for the visual in the Map key for visual text box.

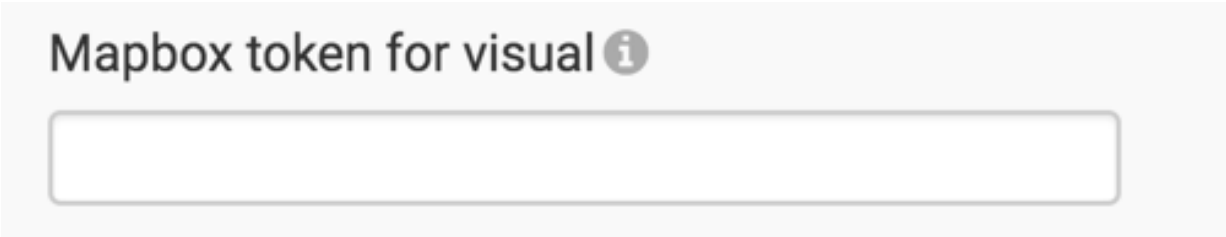

This setting will override the Mapbox Token setting in the Site Settings menu.

Important: Your organization must obtain its own credentials for Google Maps and MapBox. See Site SettingsAPI Keys for Interactive Map Visual.

Specifying atlas mapbox server for a visual

You can specify an Atlas Mapbox server at the visual level. With Atlas Mapbox server, you can run Mapbox APIs on your own infrastructure, whether on your own private cloud or on-premises.

About this task

This is an optional task.

To access Interactive Maps visuals, you can also use a Mapbox server or specify an Atlas Mapbox server at the site level.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Map Server.

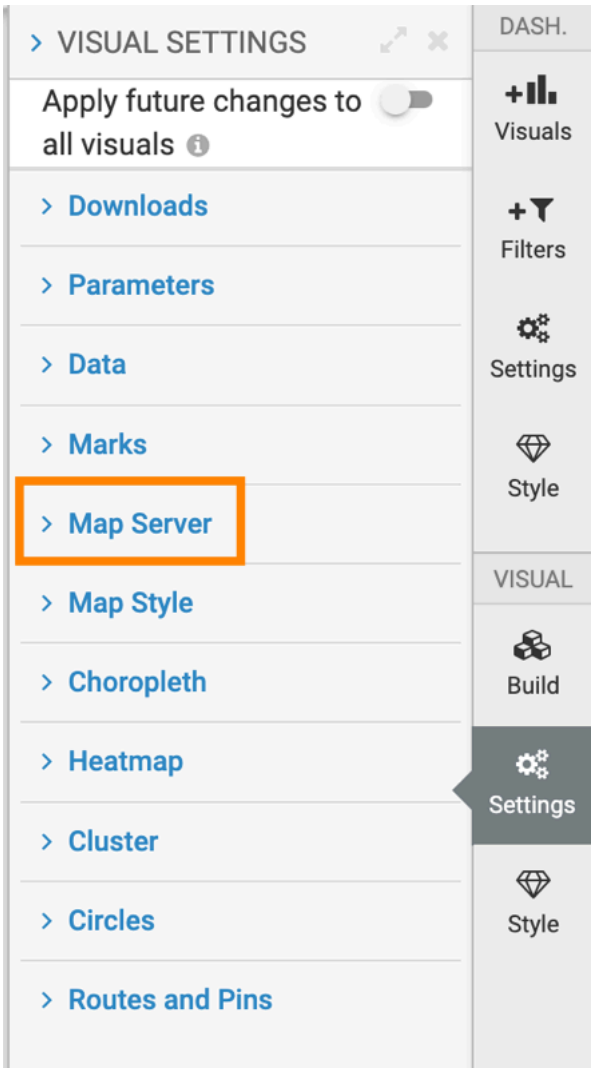

3. Enter the IP address or hostname of the Atlas Mapbox server in the Atlas Mapbox server for visual text box.

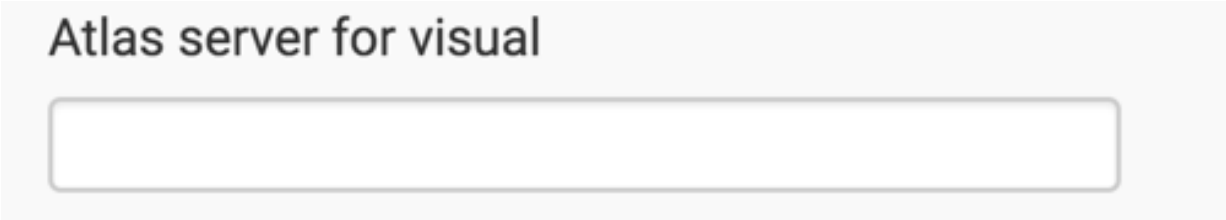

This setting will override the IP/Hostname of Atlas Mapbox server (optional) setting in the Site Settings menu.

Customizing map style

In interactive map visuals, Cloudera Data Visualization enables you to choose from six different map styles or add a custom mapbox style. A map style defines the visual appearance of a map.

Changing the map style of a visual

Cloudera Data Visualization provides six different interactive map style options.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Map Style.

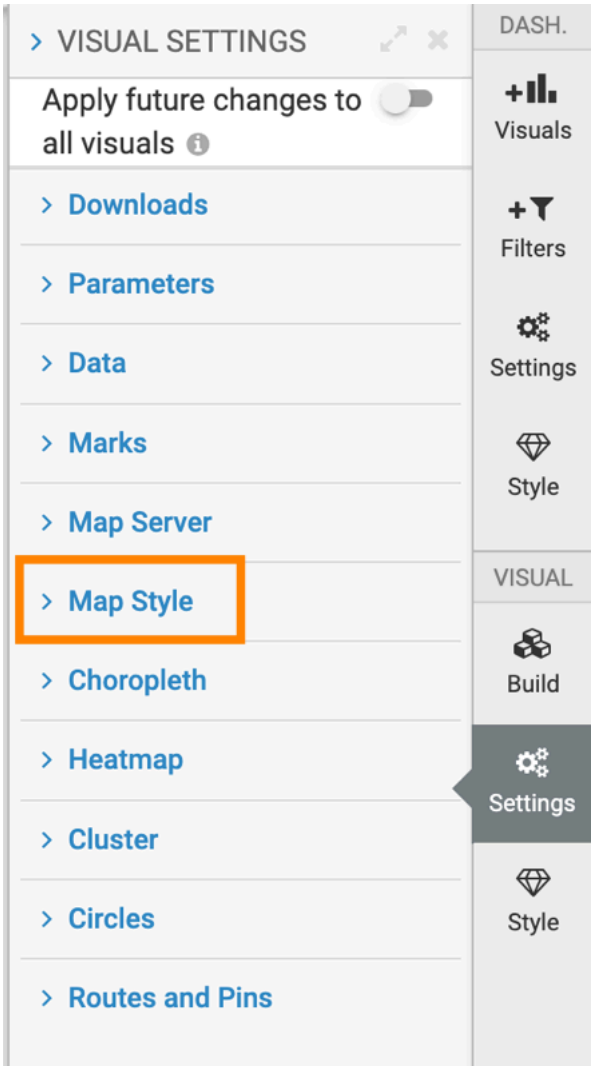

3. To change the map style of an interactive map visual, select one of the options in the Map Style dropdown menu. The options are Default, Subtle, Dark, Light, Muted, and Satellite.

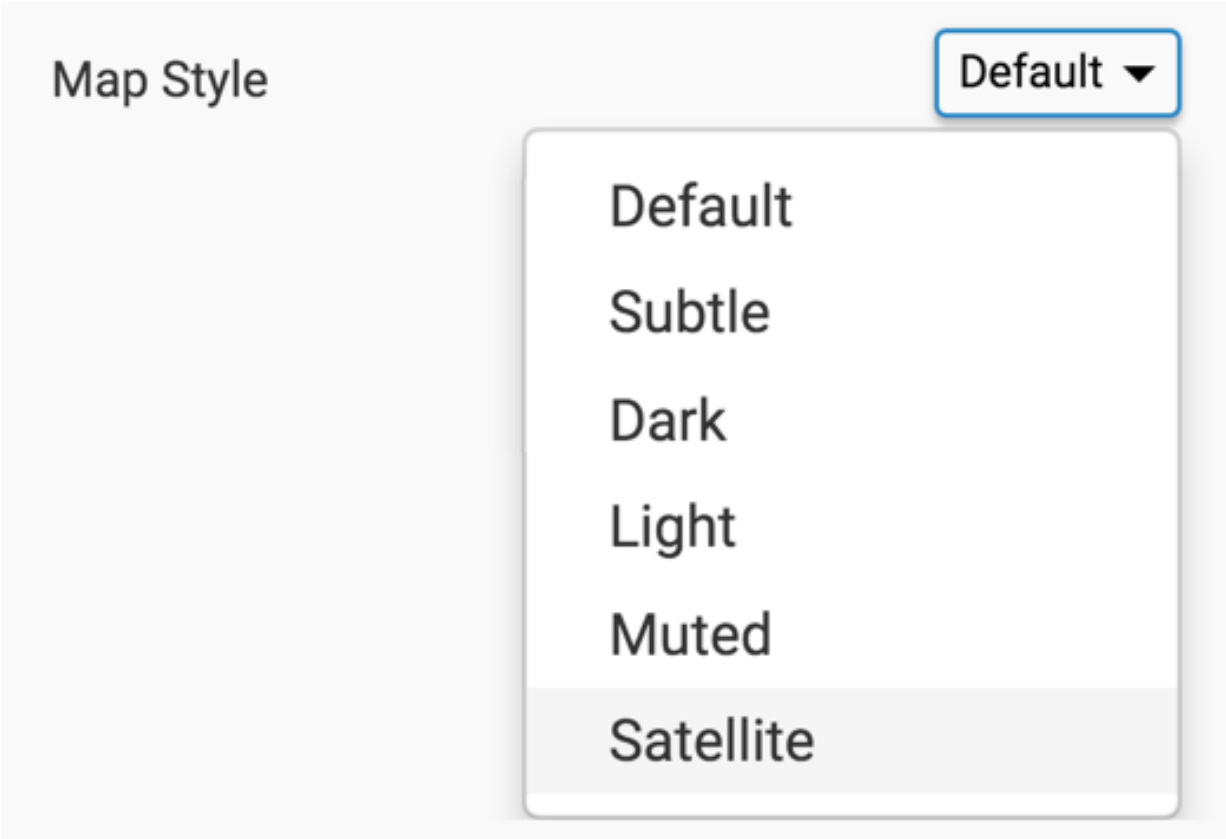

Example

Here is an example of style options available when you choose the Mapbox Server:

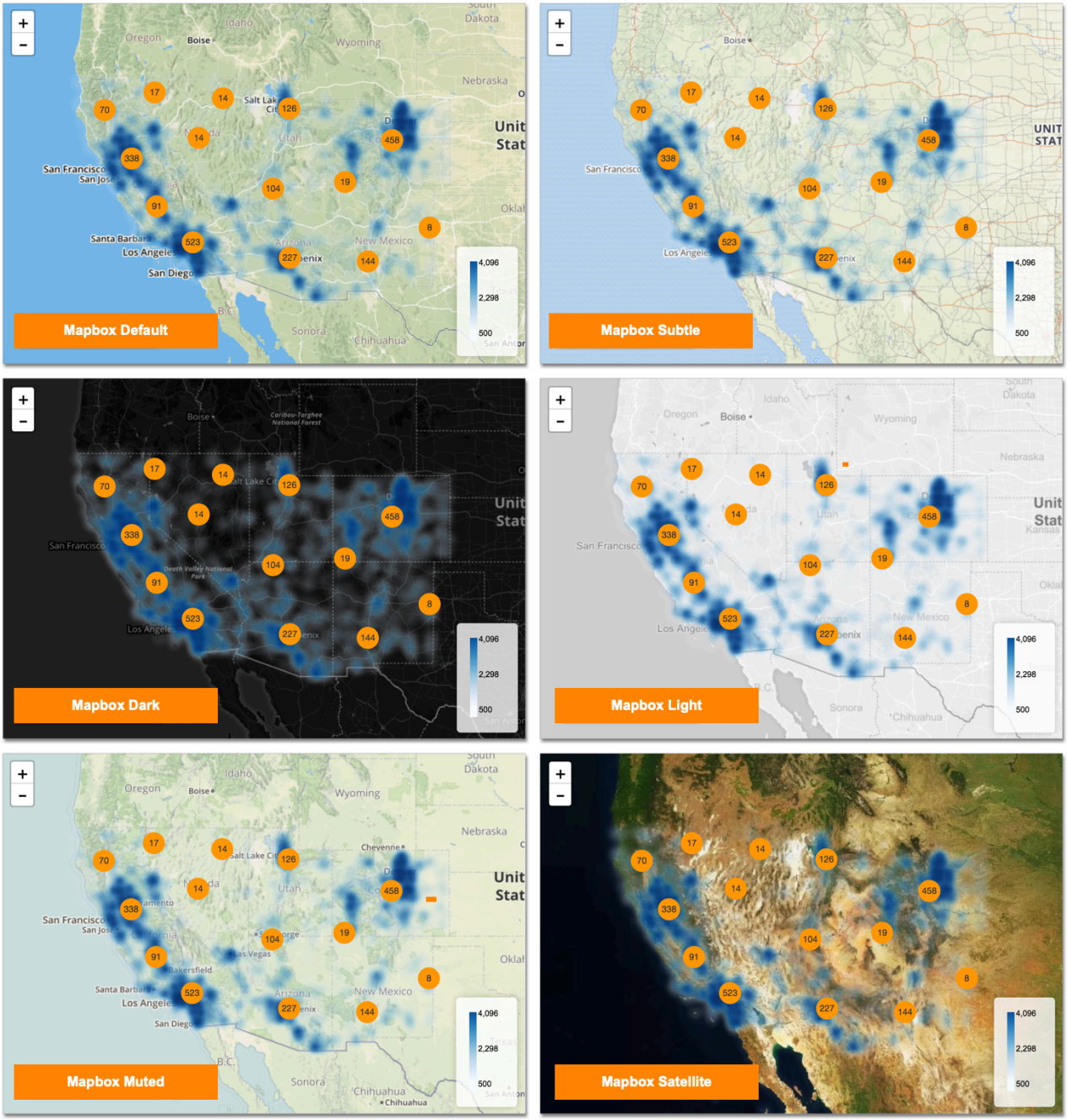

Here is an example of style options available when you choose the Google Maps Server:

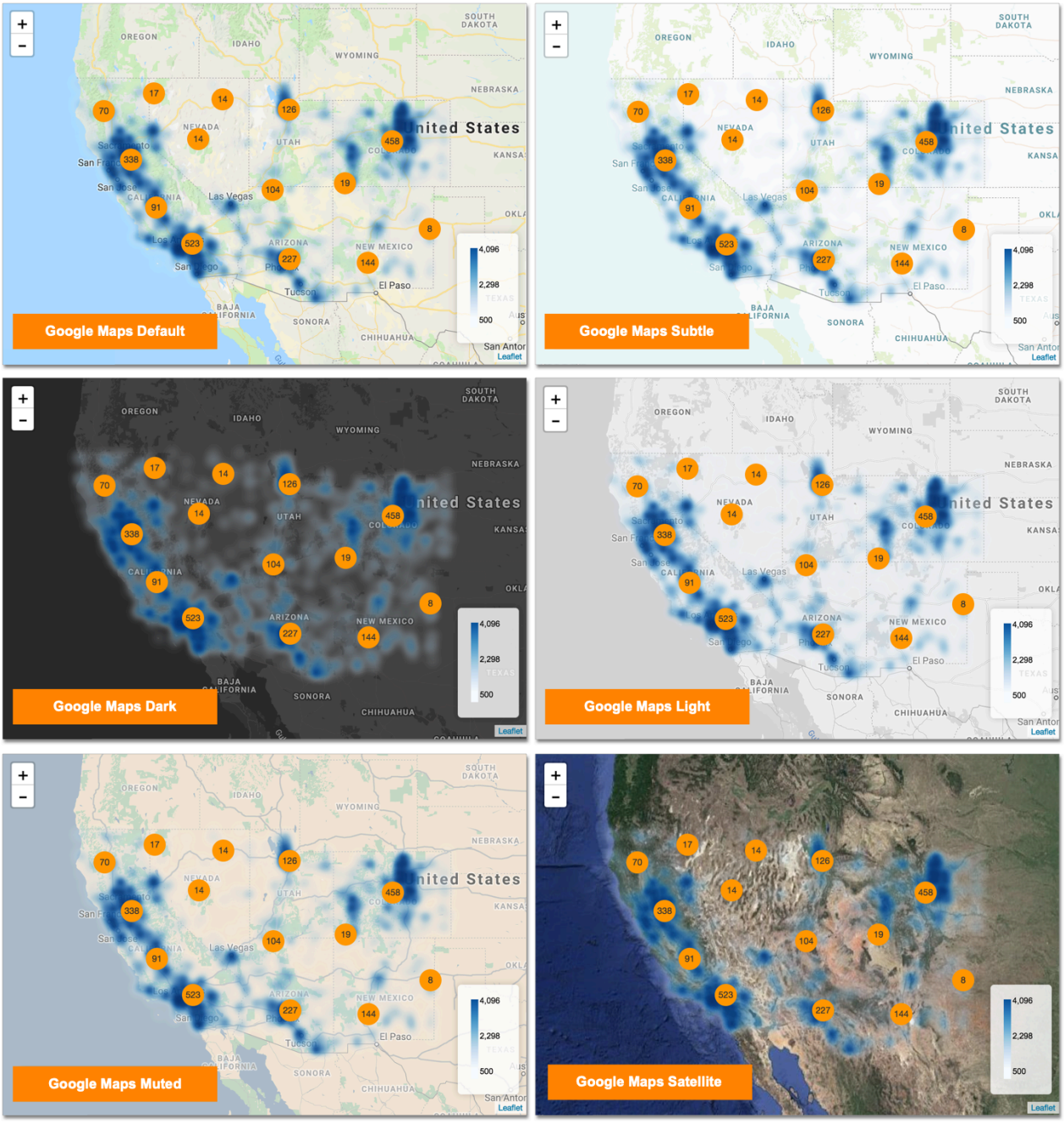

Adding custom mapbox style

Cloudera Data Visualization enables you to add a custom mapbox style in an interactive map visual.

About this task

To create a choropleth map in an Interactive Map visual, you need a Custom Mapbox style and a Layer ID.

When you request for a custom style from Mapbox, make sure that this style has a choropleth shape file in it. This shape file has the choropleth Layer ID, which is the name of a unique layer within the Custom Mapbox style that identifies the shapes that you would like to color by a measure in your data. See *Specifying a Choropleth Layer ID*.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Map Style.

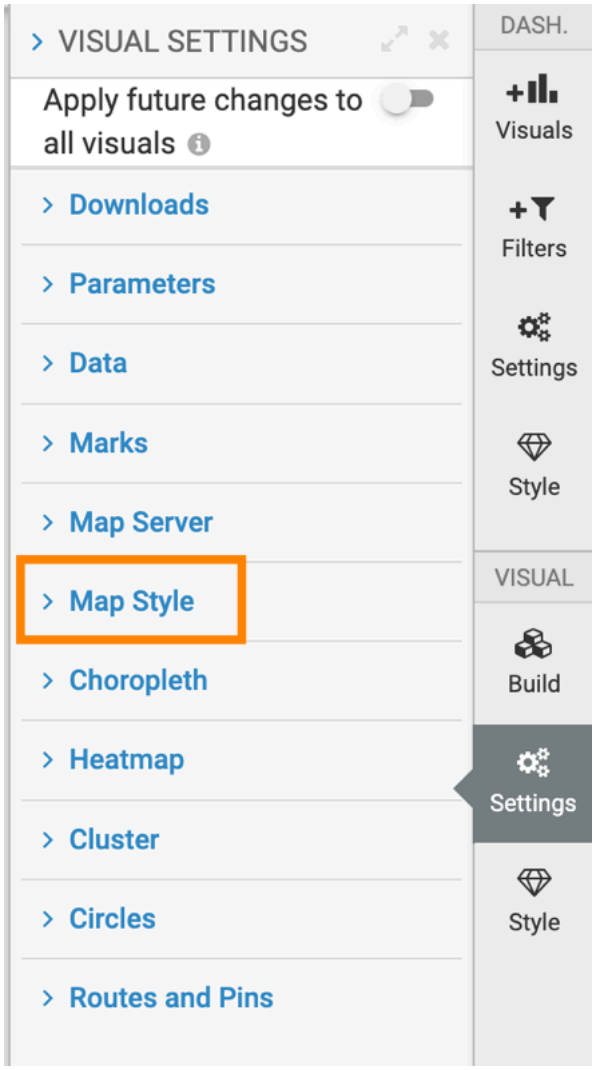

3. To add a custom mapbox style that you got from Mapbox, enter the custom mapbox style in the Custom Mapbox style option text box.

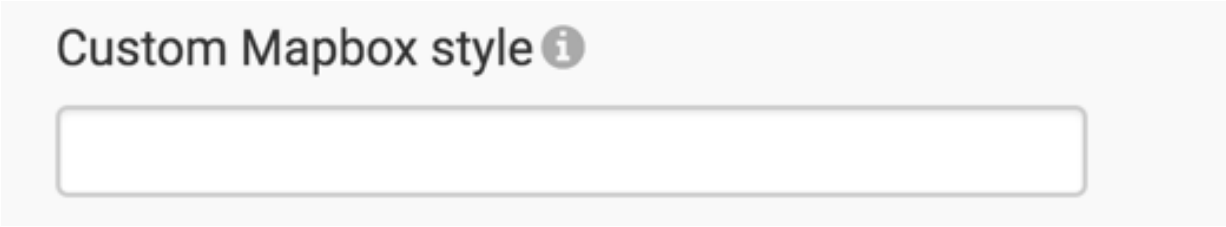

Related Information

Specifying choropleth layer ID

Customizing choropleth

A choropleth map shows geographical areas that are shaded in proportion to the value of the measurement that is displayed on the map. In interactive map visuals, Cloudera Data Visualization enables you to create and customize choropleth maps.

Specifying choropleth layer ID

To create a choropleth map in an interactive map visual, you must specify a choropleth Layer ID.

About this task

When you request for a custom style from Mapbox, make sure that this style has a choropleth shape file in it. This shape file has the choropleth Layer ID, which is the name of a unique layer within the Custom Mapbox style that identifies the shapes that you would like to color by a measure in your data. See *Custom mapbox style*.

To create a choropleth map in an Interactive Map visual, you need a Custom Mapbox style and a Layer ID.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Choropleth.

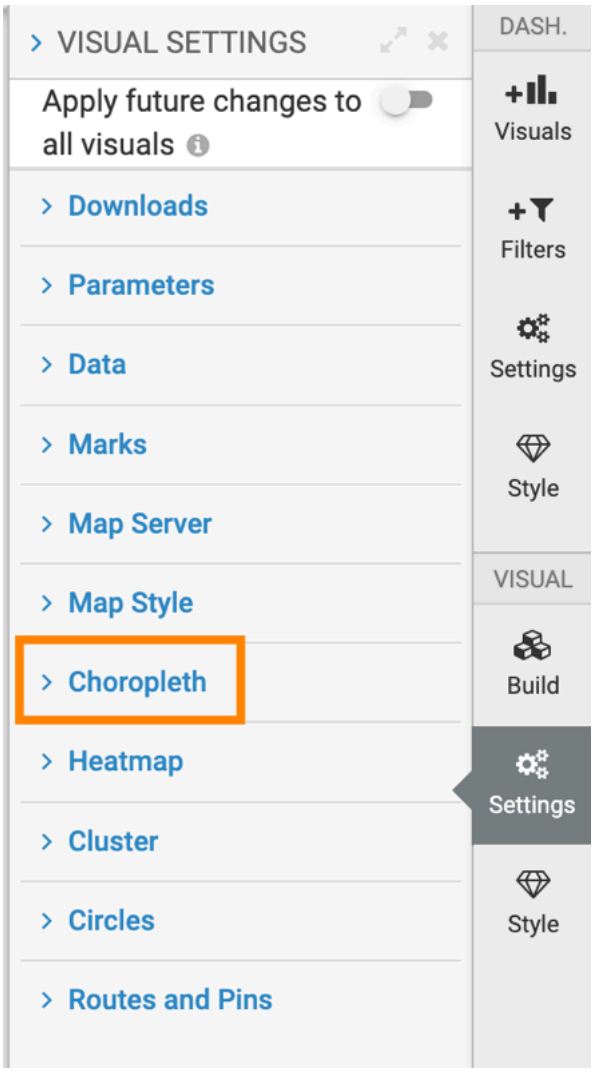

3. To specify a layer ID, enter a choropleth layer ID in the Layer ID text box.

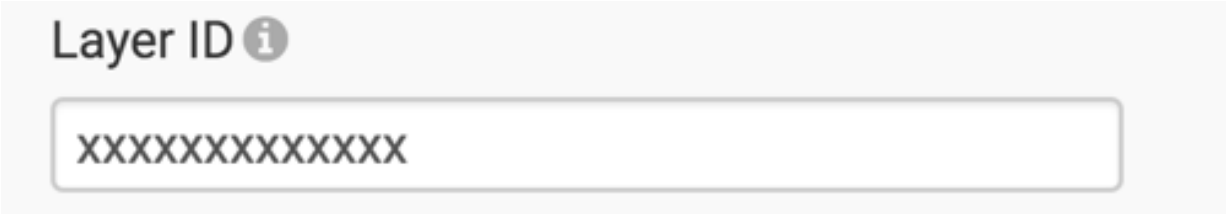

Related Information

Adding custom mapbox style

Specifying choropleth opacity

In an interactive map visual, Cloudera Data Visualization enables you to change the opacity of choropleth maps.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Choropleth.

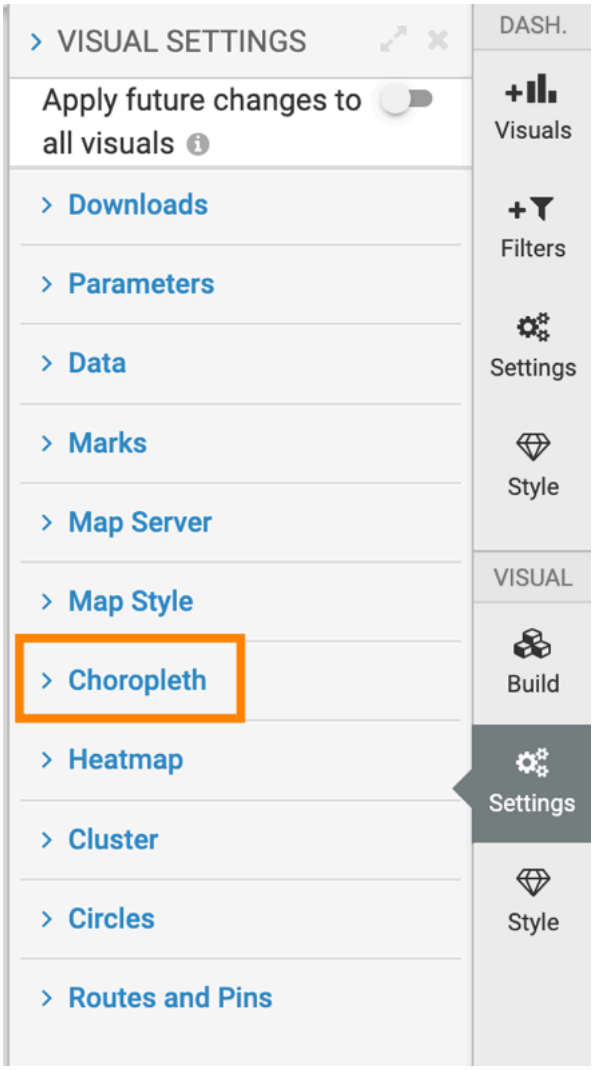

3. To change the opacity (color saturation) of choropleths, adjust the selector for the Choropleth Opacity. The valid range is from 0 through 1.

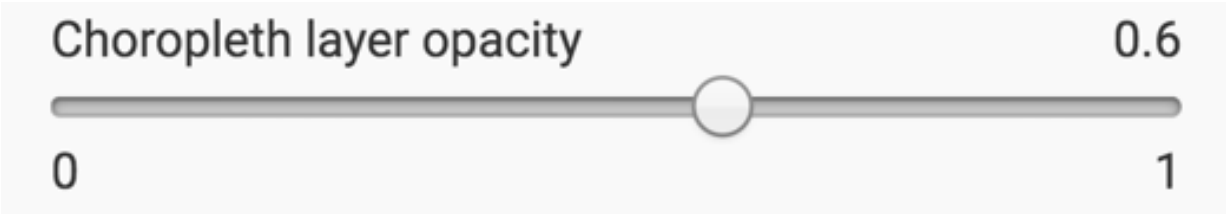

Compare the appearance of the choropleth map when the opacity is set to 1 and 0.5.

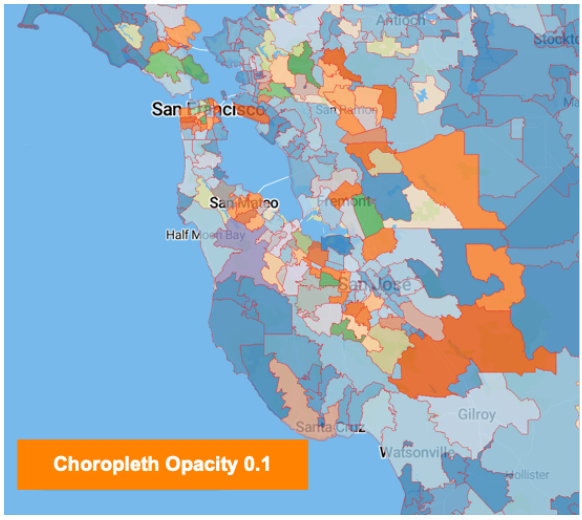

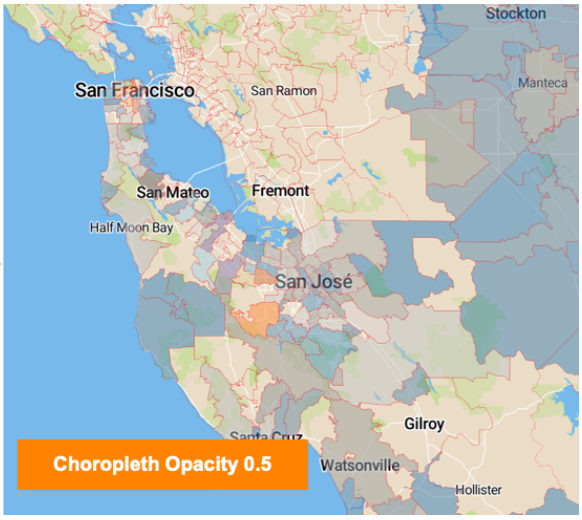

Displaying legend in choropleth map

In an Interactive Map visual, Cloudera Data Visualization enables you to display a legend in a choropleth map.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Choropleth.

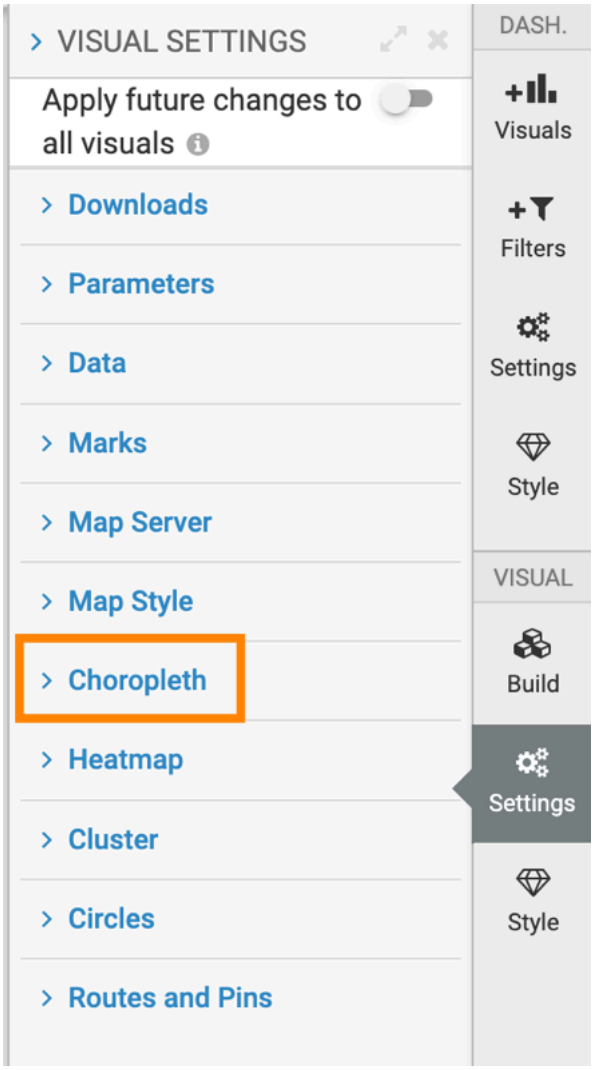

3. To display a legend, select the Include legend option.

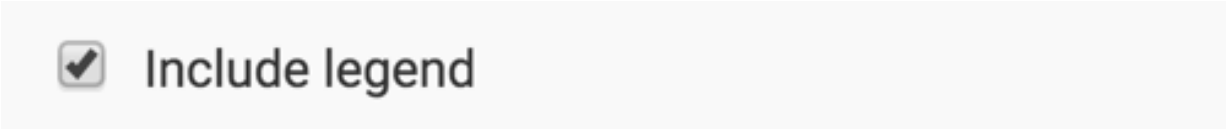

In the following image, notice the legend at the bottom right corner, with gradient of colors showing the average tax liability in California by zip code.

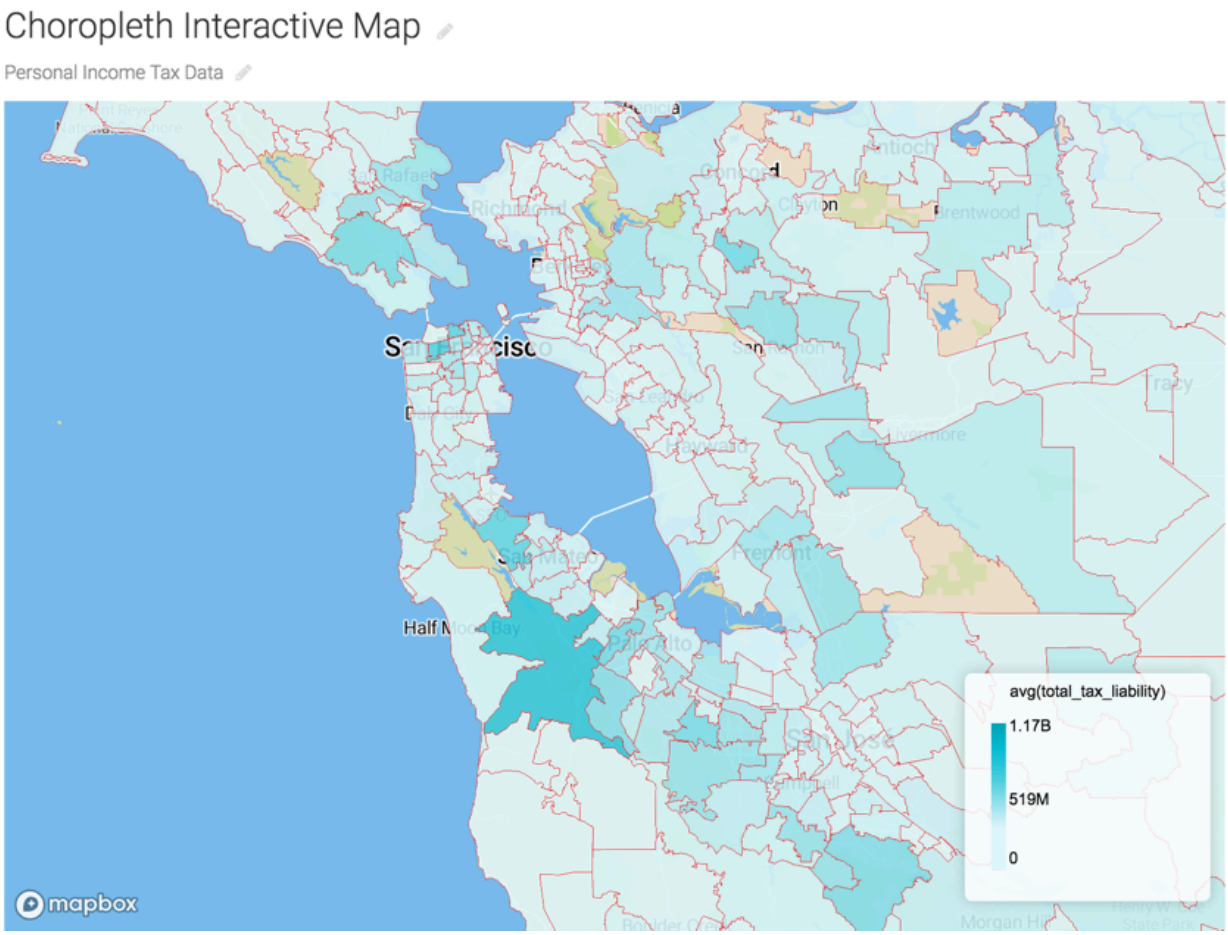

Remembering location of map

In interactive map visuals, Cloudera Data Visualization enables you to remember the location of the map.

About this task

In Edit mode, if you save and close the visual, and open the same visual again, the map opens at the same location where you closed the visual.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Choropleth.

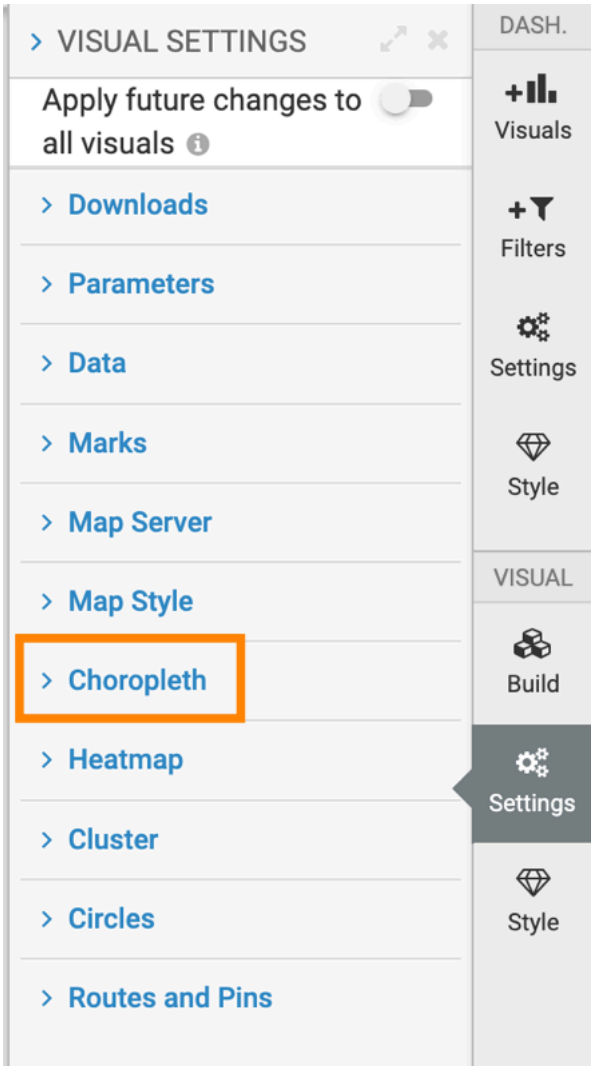

3. To remember the location of the map, select the Remember map location option.

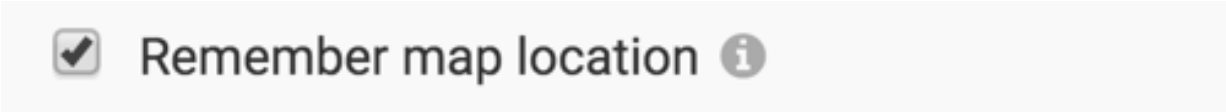

To demonstrate how this feature works, follow these steps:

- Create a choropleth interactive visual. See *Creating Choropleth Interactive Map Visuals*.
- In Edit mode, navigate to a location in the map. In our example, the tooltip shows our location as zip_code 94027.
- Click Save.
- Click Close.
- Open the same visual again. The visual opens at the same location, zip_code 94027, where we closed the visual.

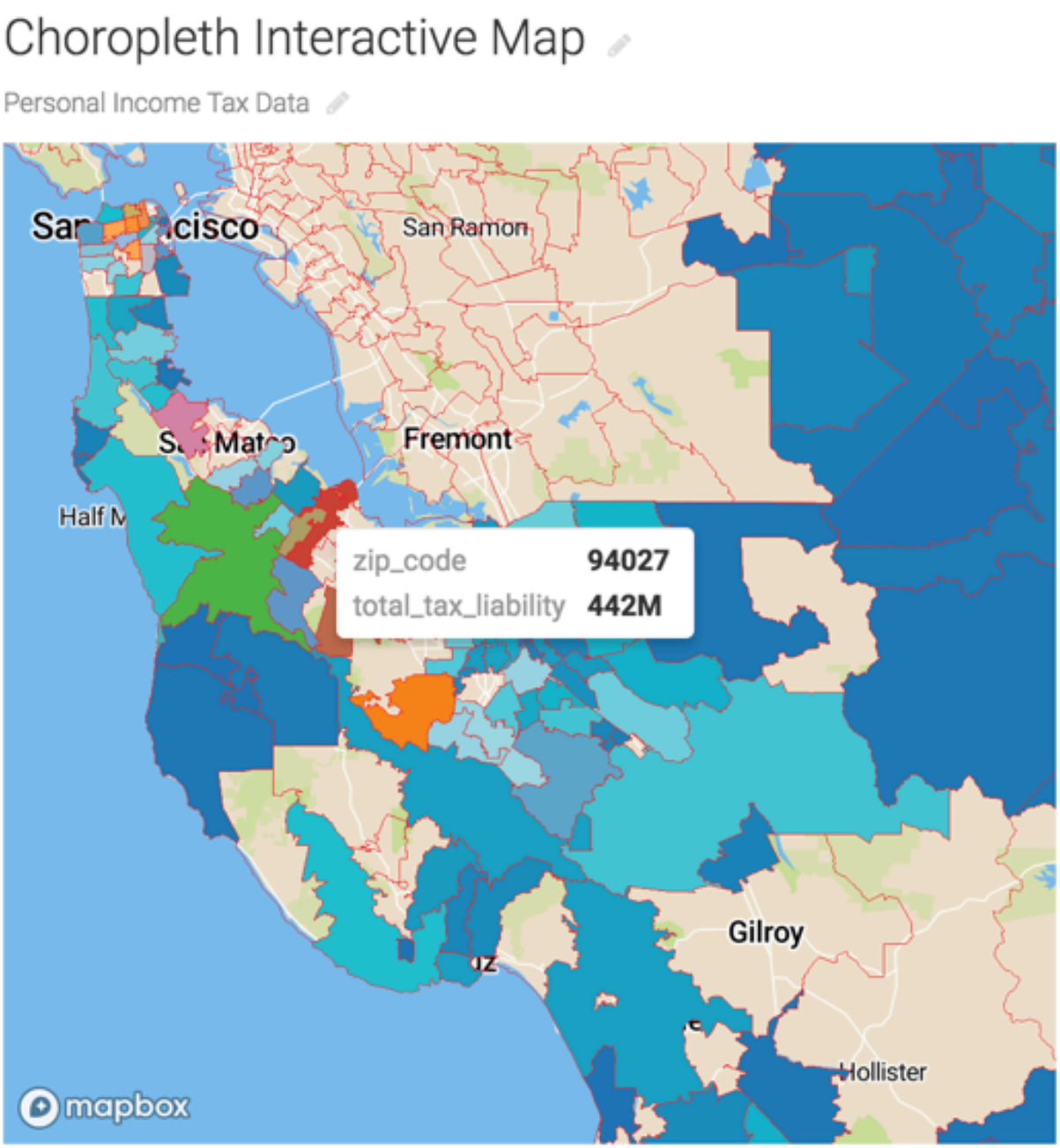

Customizing heatmap

In an interactive map visual, Cloudera Data Visualization enables you to change Heatmap settings.

The Heatmap is on by default, along with Cluster.

The color of the heatmap indicates the value of the aggregate on the Measurements shelf, and it is determined by the color palette. The heatmap includes an optional color legend, which is in gradient form and is based on the color palette. The intensity magnification multiplies the value of the measure by a constant, and adjust the color display by pushing it 'higher' up the selected color palette. The heatmap radius displays, in pixels, the size of each point.

Make the appropriate changes under the following Heatmap options:

Displaying heatmap

About this task

In an Interactive Map visual, Cloudera Data Visualization enables you to show or hide the Heatmap.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Heatmap.

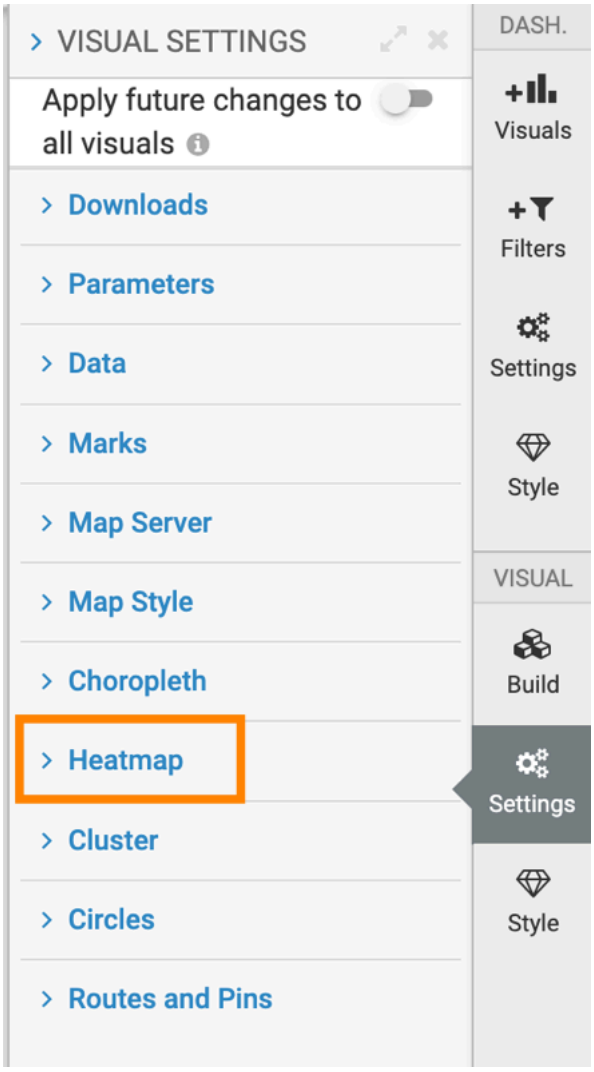

3. To show or hide the Heatmap, select or unselect Enable Heatmap option. This option is on by default.

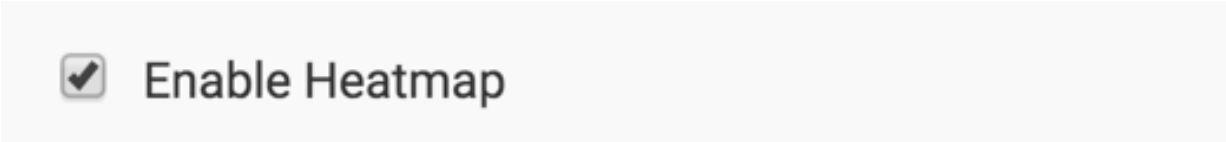

If you disable the Heatmap option, Google map renders only with the Cluster option, the other default.

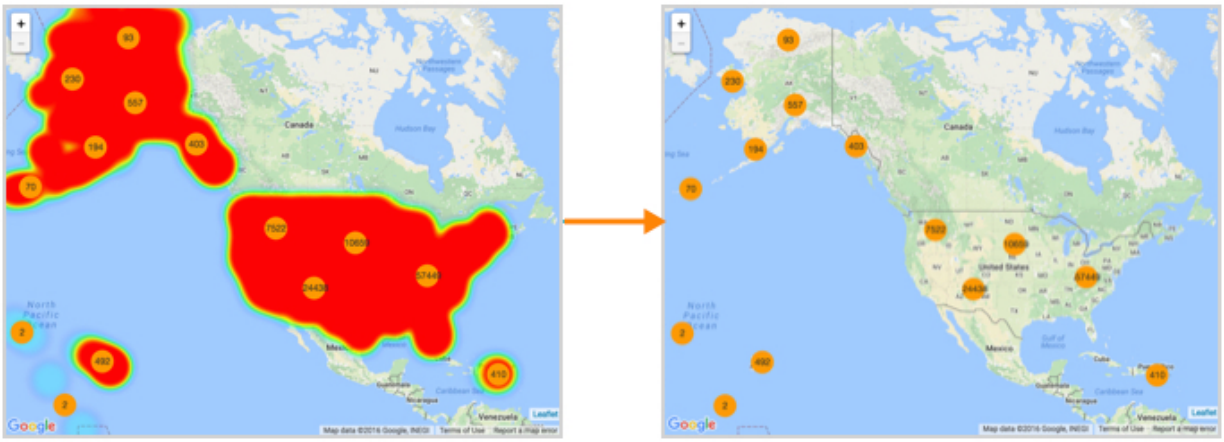

Similarly, the Mapbox map renders without the Heatmap option.

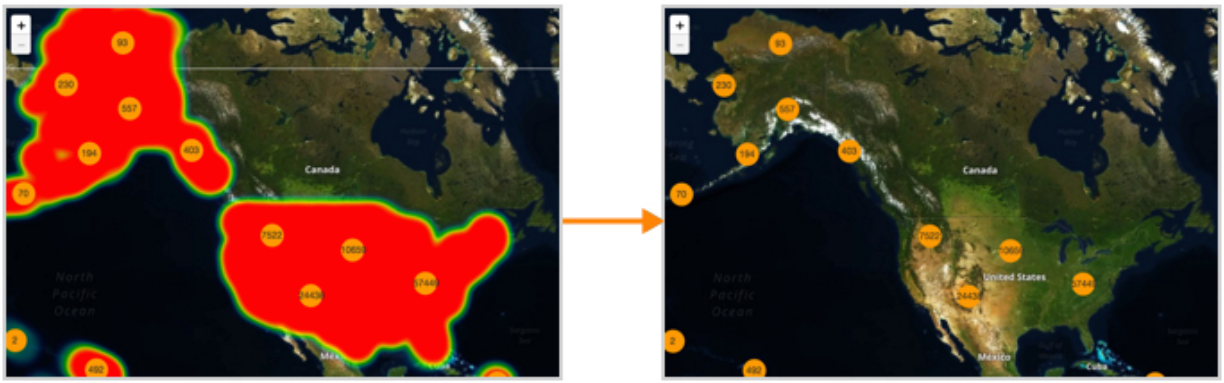

Changing heatmap radius

About this task

In an interactive map visual, Cloudera Data Visualization enables you to change the granularity of the heatmap.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Heatmap.

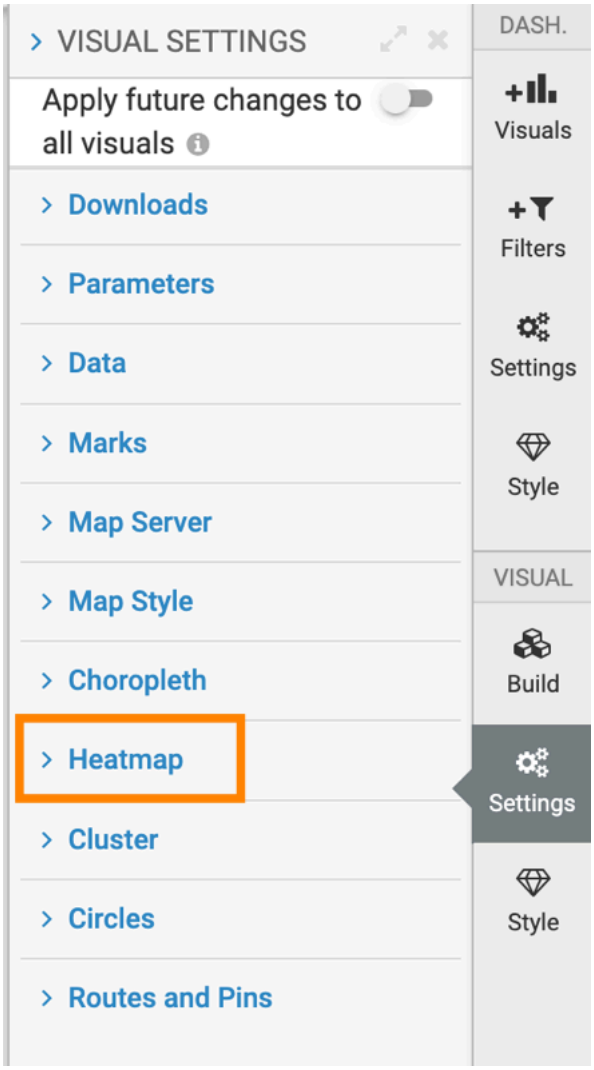

3. To change the granularity of a Heatmap, change the value for the Heatmap Radius option. The default value is 25.

The default value of heatmap radius is 25. Compare the results of setting heatmap radius on Google Map to 5, 10, 15, 20, 25, and 30. In the following figure, the intensity magnification for all cases is 100.

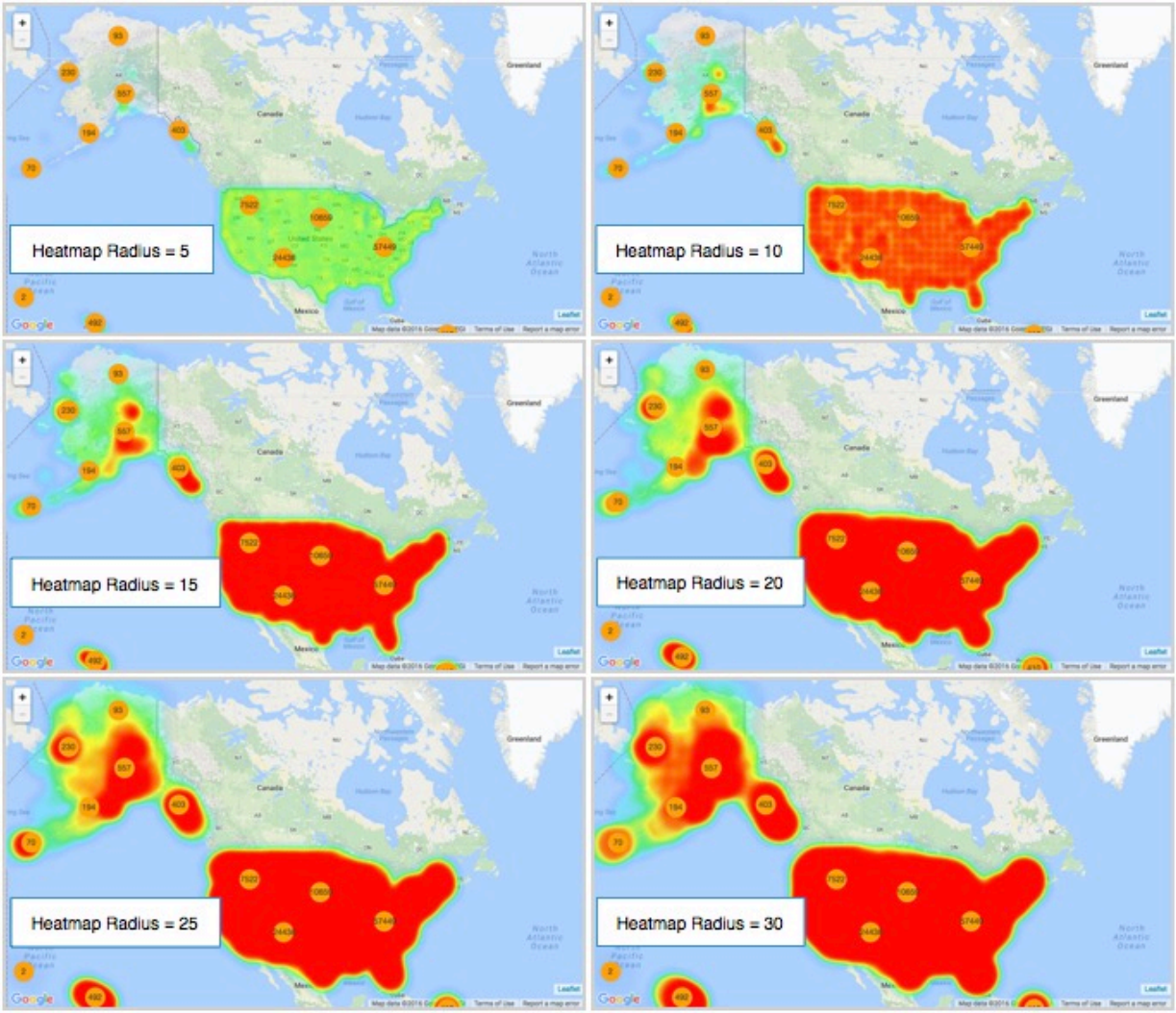

Similarly, compare the effect of the same heatmap radius settings on the Mapbox map.

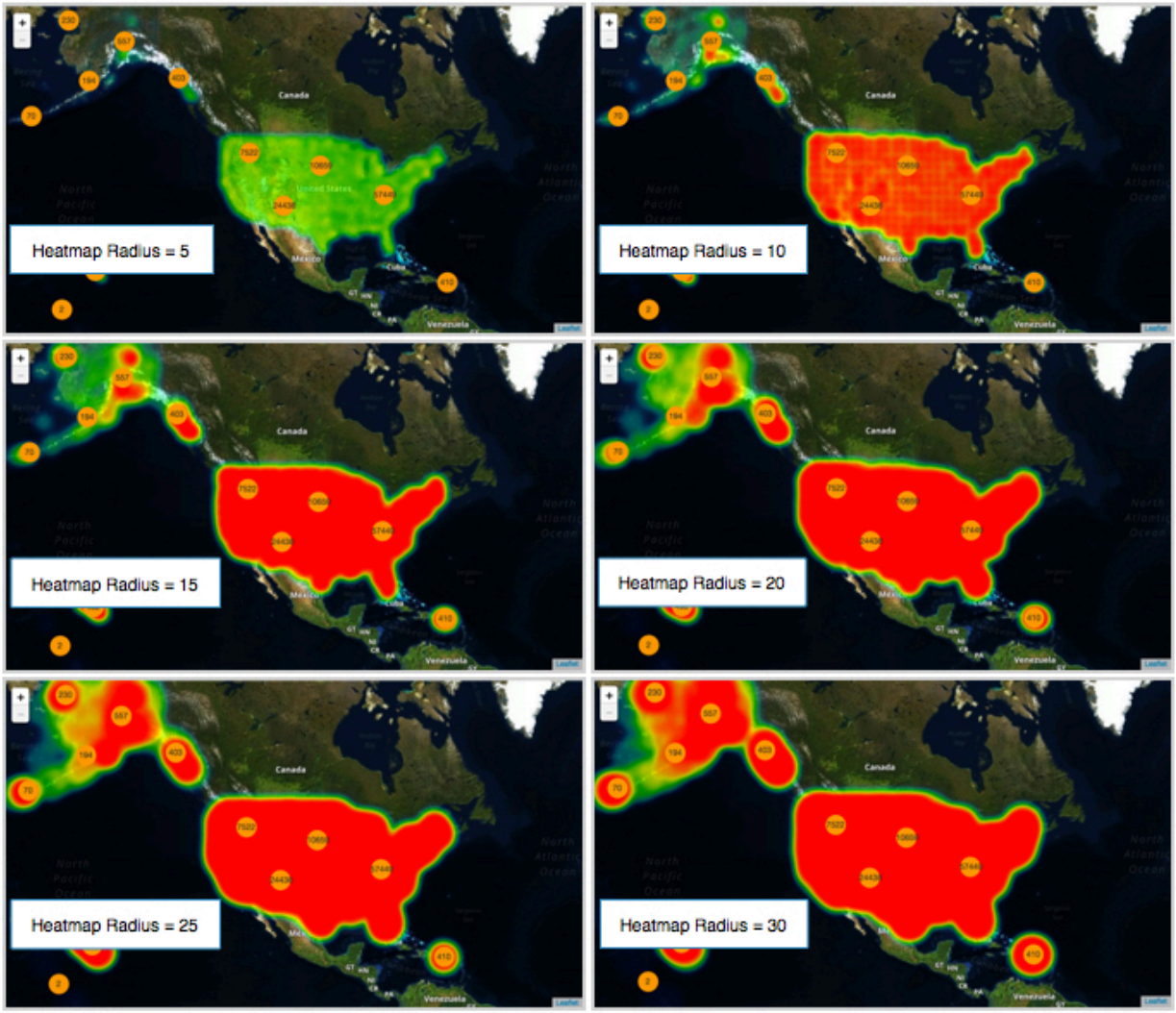

Changing intensity magnification

About this task

In an interactive map visual, Cloudera Data Visualization enables you to intensity magnification for heatmaps.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Heatmap.

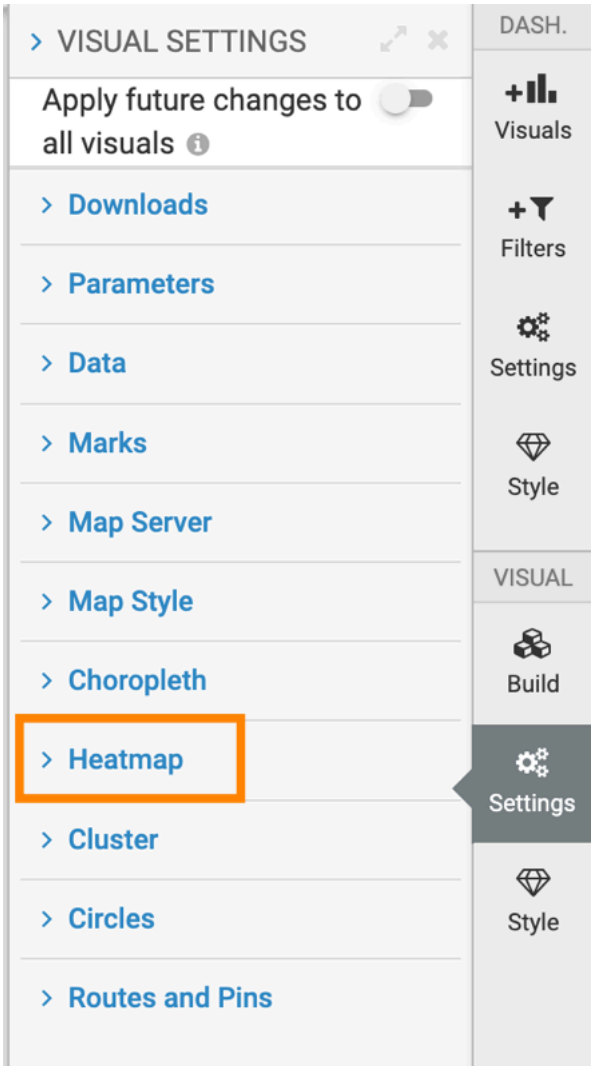

3. To change the intensity magnification of the Heatmap, change the value for the Intensity Magnification option. The default value is 1000.

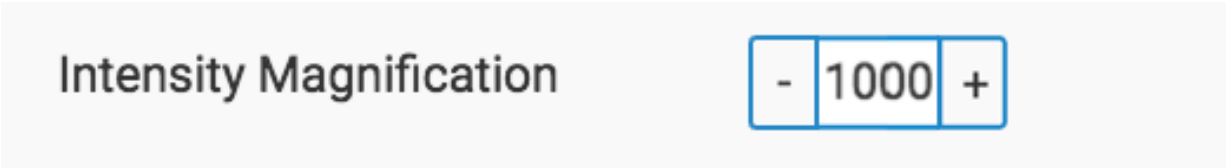

The default intensity magnification value is 1000. Compare the results of setting intensity magnification on Google Map to 10, 100, 1,000, and 10,000. In the following figures, the heatmap radius for all cases is 25.

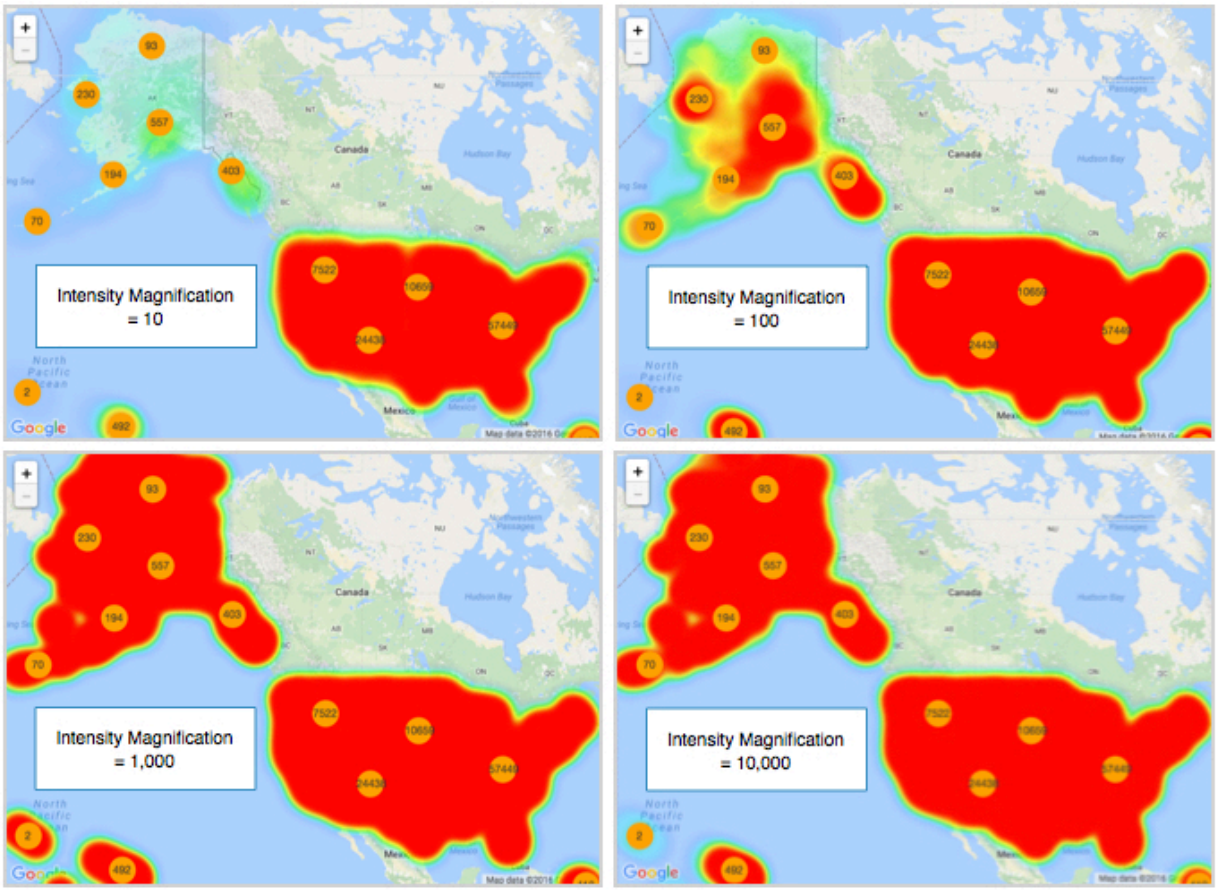

Similarly, compare the effect of the same heat intensity magnification settings on the Mapbox map.

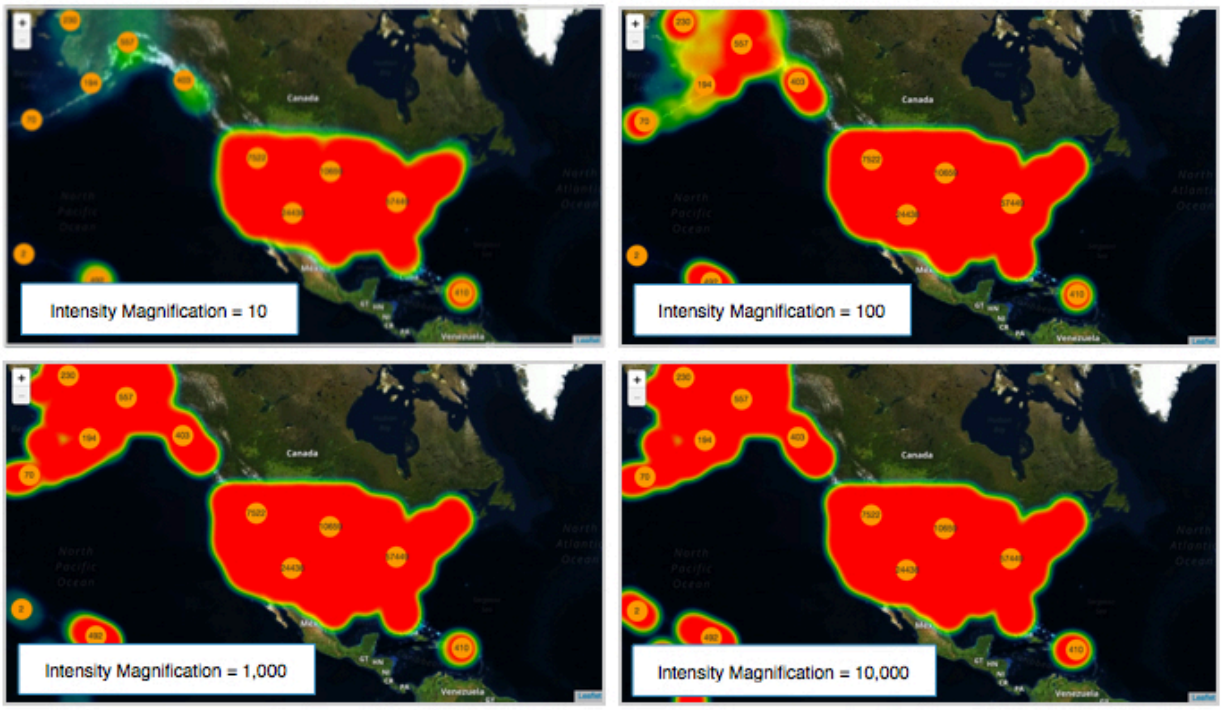

Showing heatmap legend

About this task

In an interactive map visual, Cloudera Data Visualization enables you to show or hide a legend for heatmaps.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Heatmap.

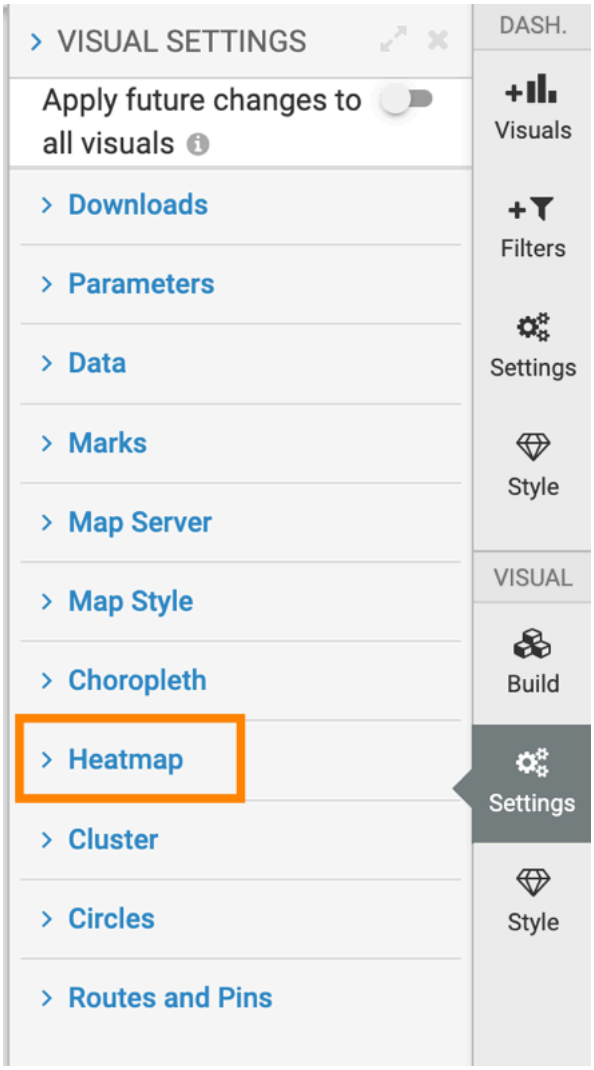

3. To show the legend for Heatmap, select or unselect Add Heatmap Legend option. This option is off by default.

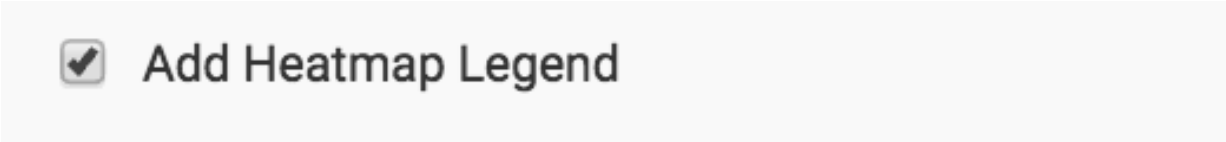

After you enable the legend option, the legend appears in the lower right-corner of the Google map.

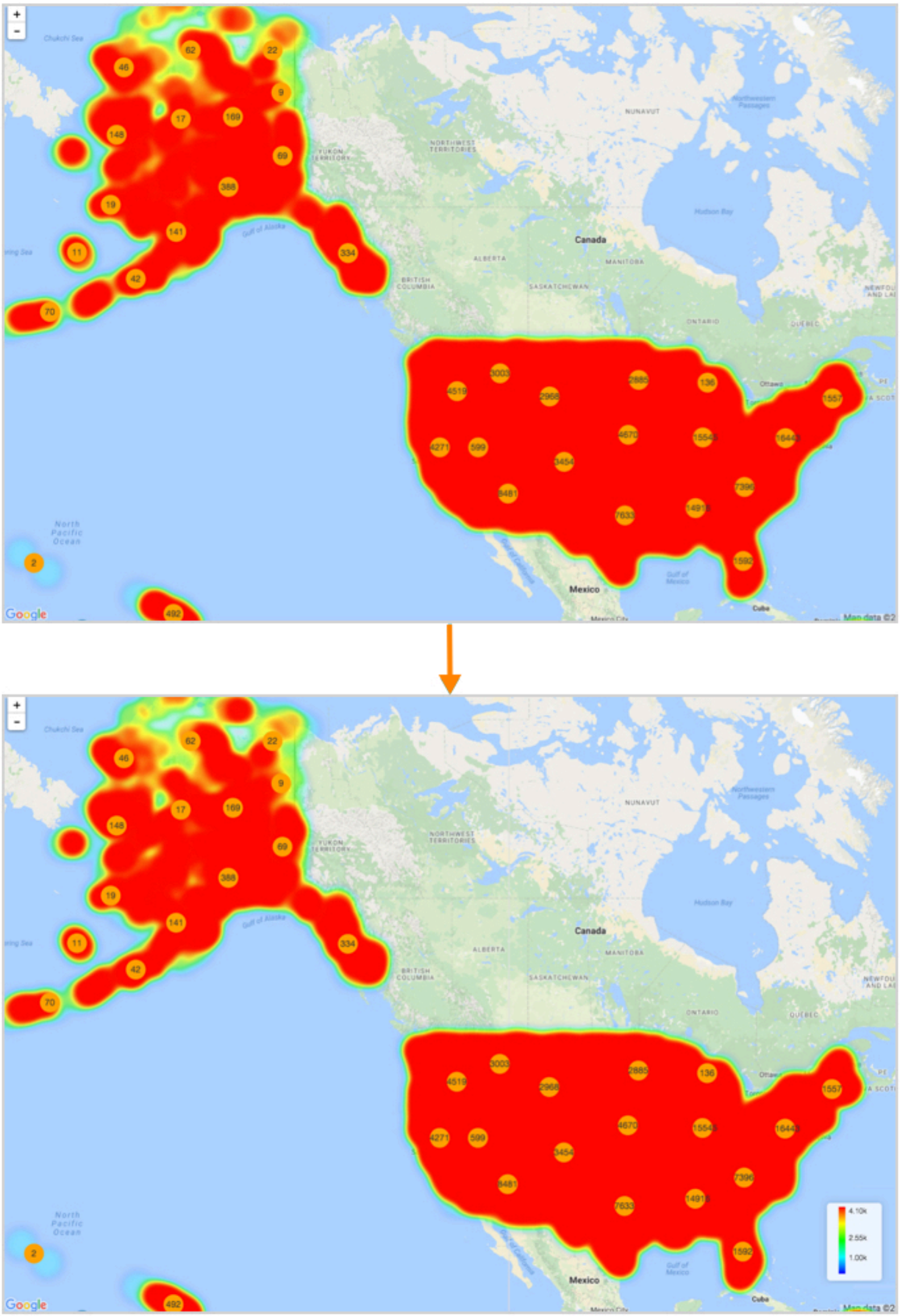
Canada
Mexico
North Pacific Ocean
Google
4.10k
2.55k
1.00k

Similarly, the legend appears on the Mapbox map.

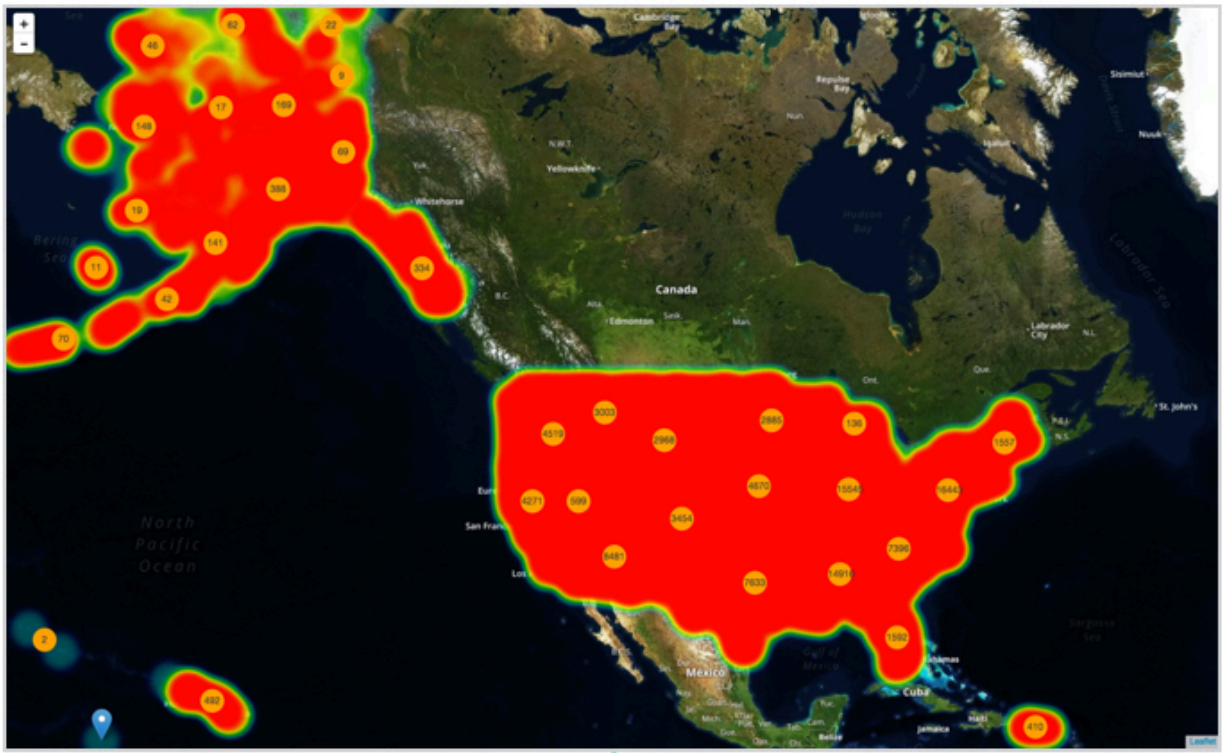

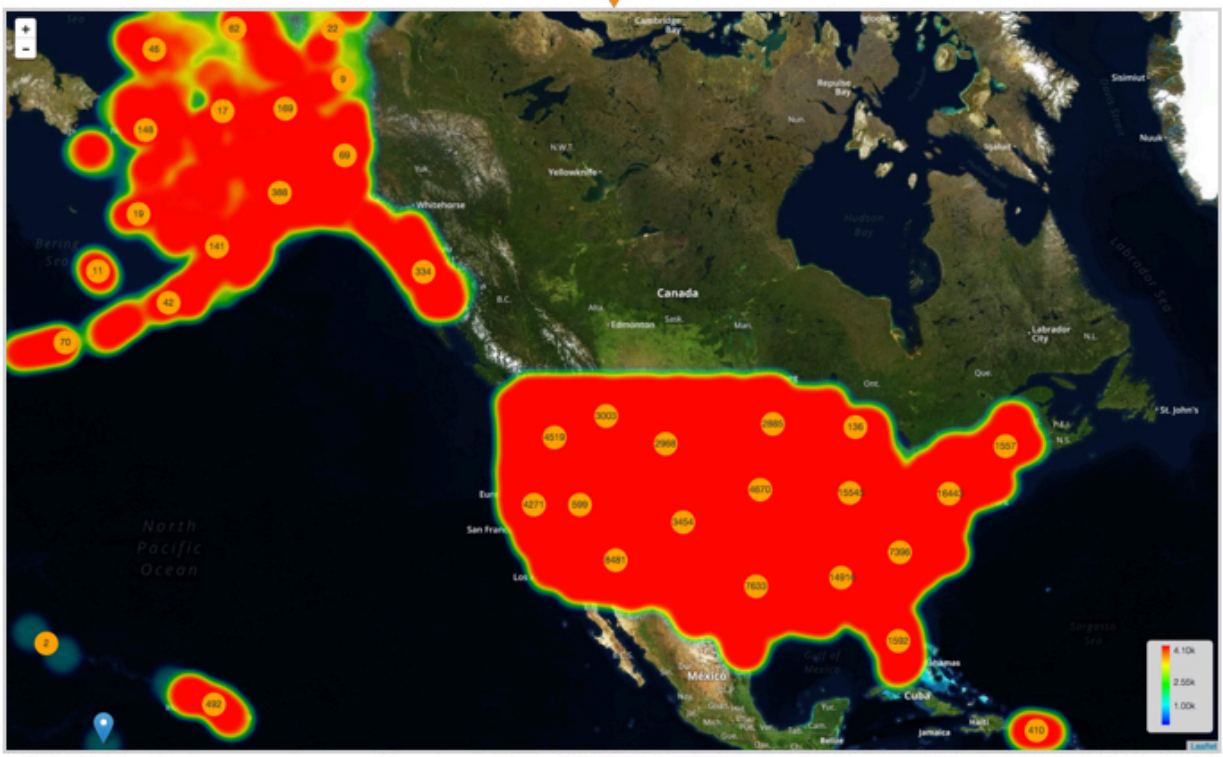

Changing visible zoom levels in heatmap

About this task

In an interactive map visual on MapBox, it is recommended to adjust the zoom level within the minimum and maximum zoom levels. Valid values are 0 through 22. For example, if the zoom level is set to more than the maximum level, heatmap will not be visible.

Sometimes, it is not possible to display data legibly at a given zoom level. Cloudera Data Visualization enables you to adjust the visible zoom levels in a heatmap.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Heatmap.

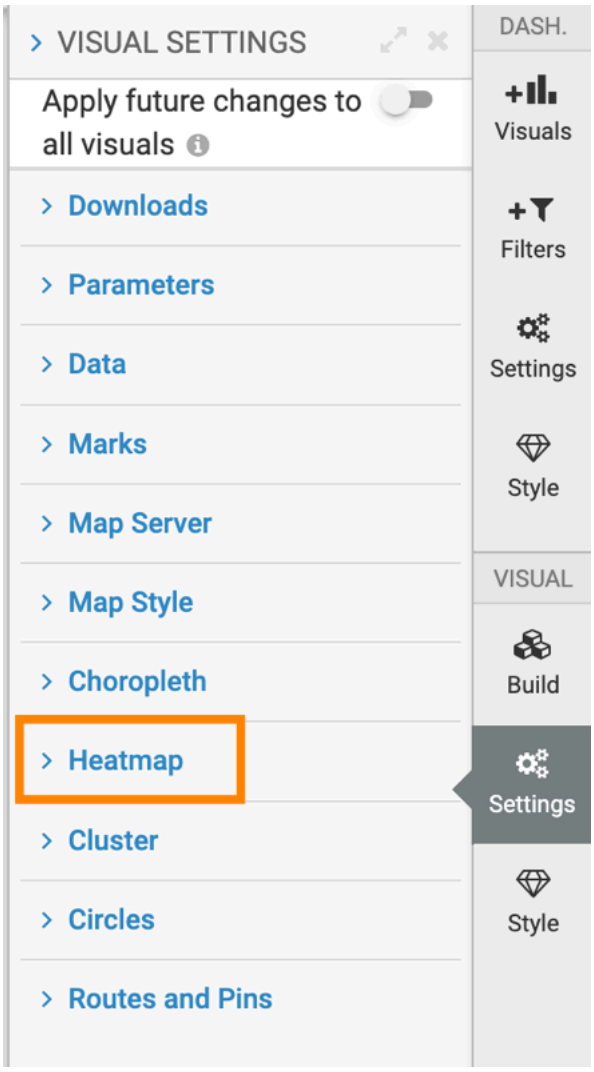

3. To adjust the zoom level of a heatmap, navigate to the Heatmap menu under Settings, and adjust the minimum and maximum values in the Visible Zoom Levels option.

   Let's set the minimum level to 1 and maximum level to 3.

   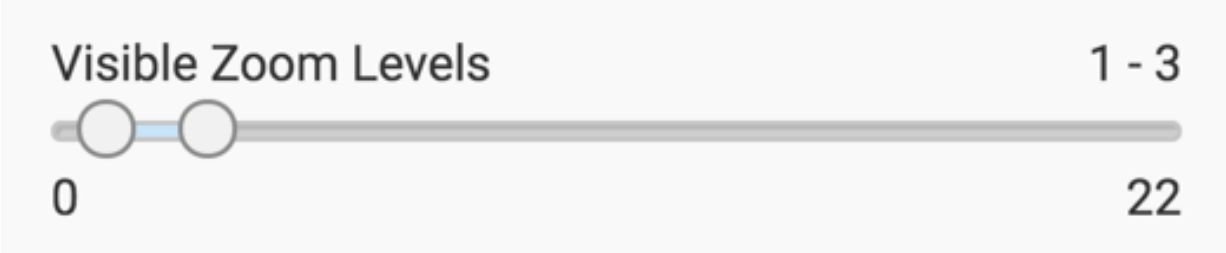

   In the following example, notice when the zoom level is set to a value less than the maximum level 2.95, heatmap is visible. However, if the zoom level is set to more than the maximum level 3.02, heatmap is not visible and the circles are smaller.

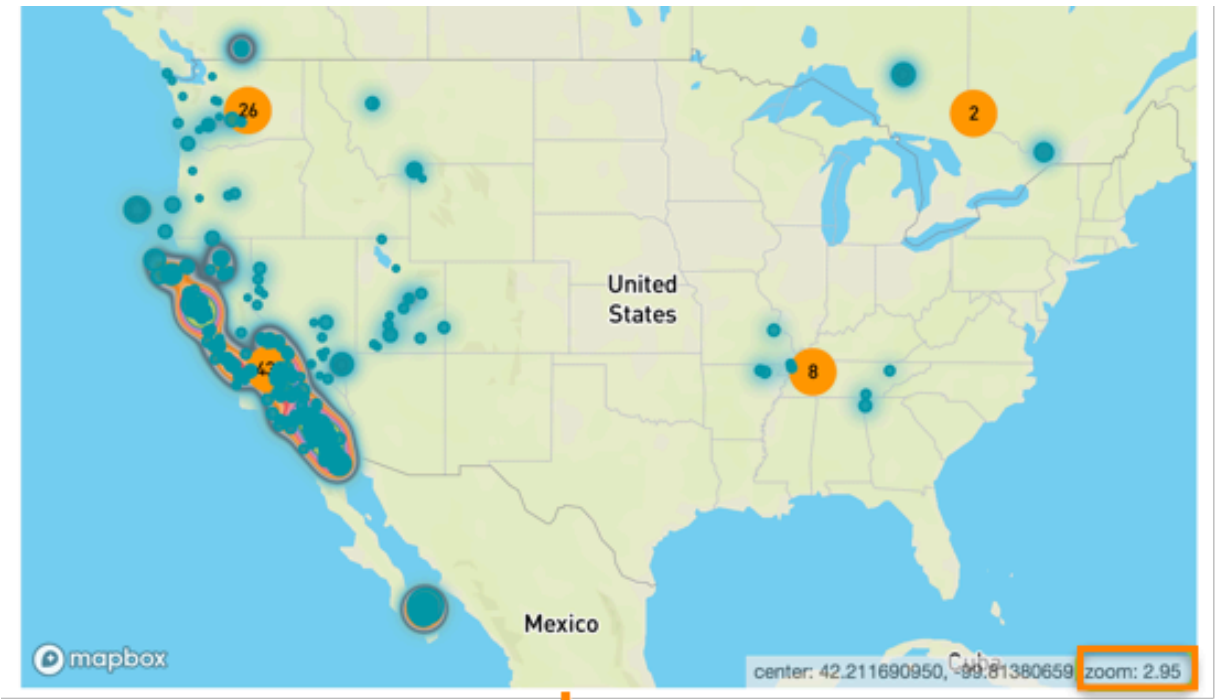

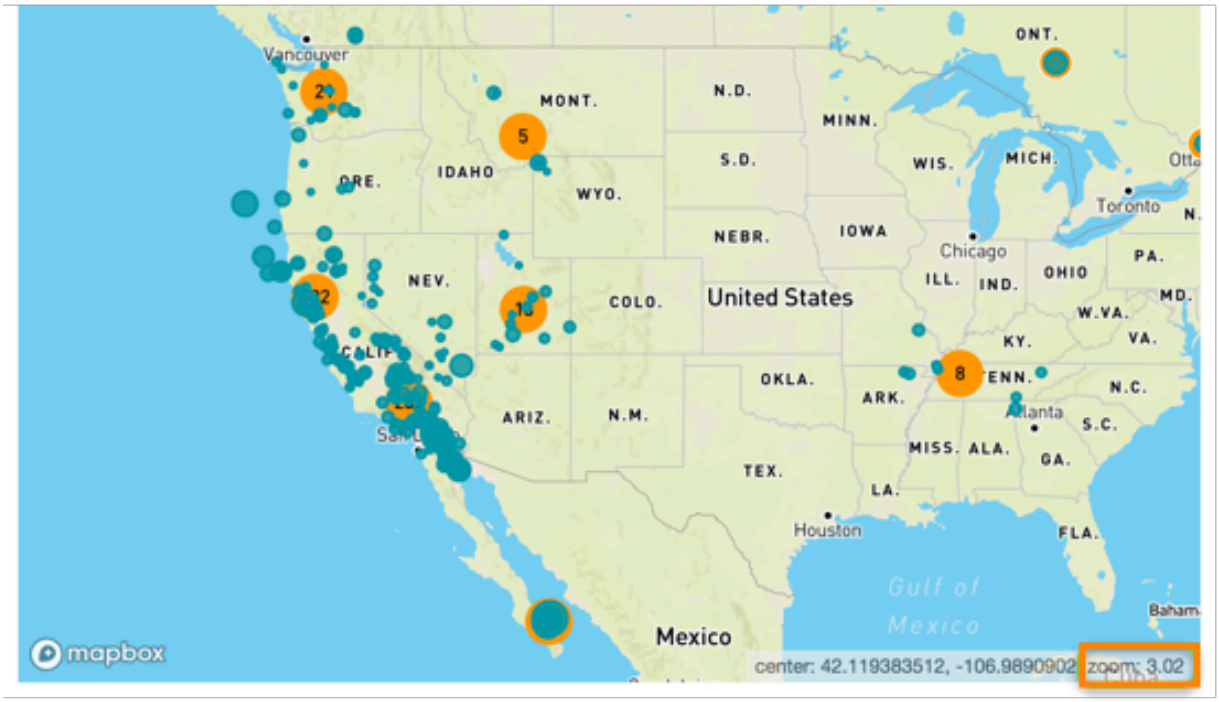

Customizing cluster

In an interactive map visual, Cloudera Data Visualization enables you to change cluster settings.

Cluster shows aggregate values within a certain cluster radius, so changing the radius changes the aggregation grouping. Note that this radius is relative to the magnification on the map, so the number of aggregations change as you zoom in and out. The cluster color specifies the color of the circle that reports the number of items in each

aggregate. At sufficiently high magnifications, when individual data points are visible as marks, the cluster supports click behavior.

The cluster is on by default, along with heatmap.

Displaying cluster

In an interactive map visual, Cloudera Data Visualization enables you to show cluster.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Cluster.

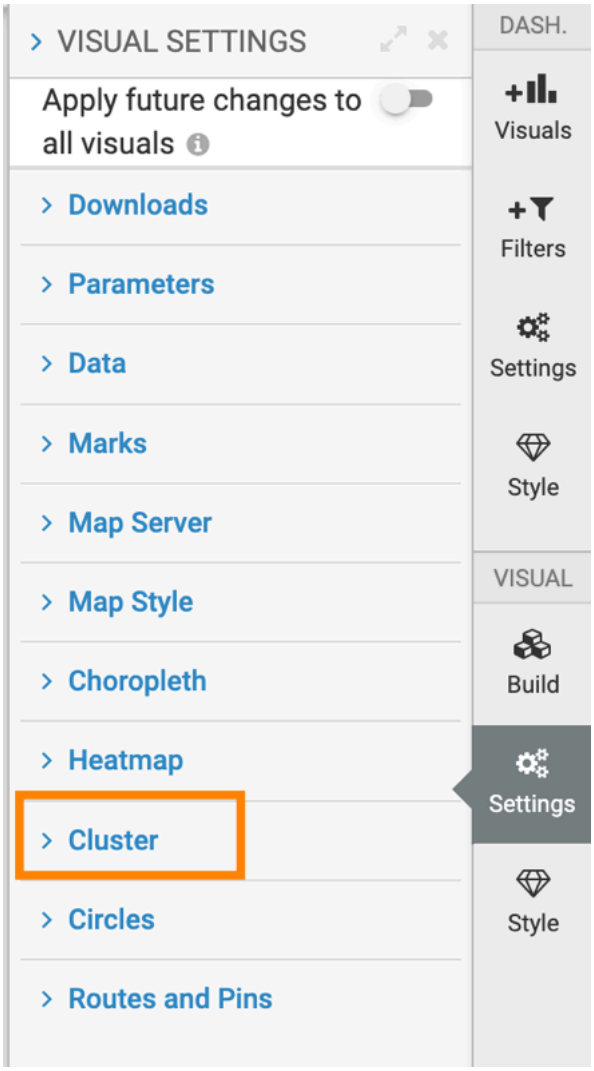

3. To show or hide the Cluster option, select or unselect Enable Cluster option. This option is on by default.

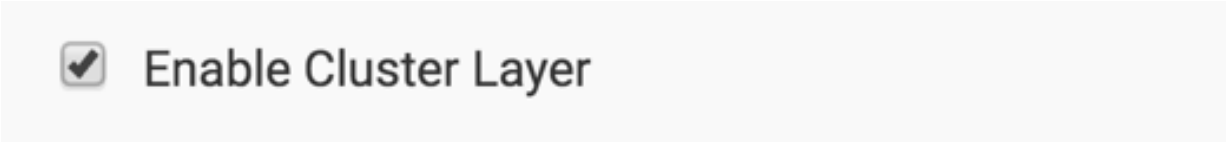

After you disable the cluster option, Google map renders only with the heatmap option, the other default.

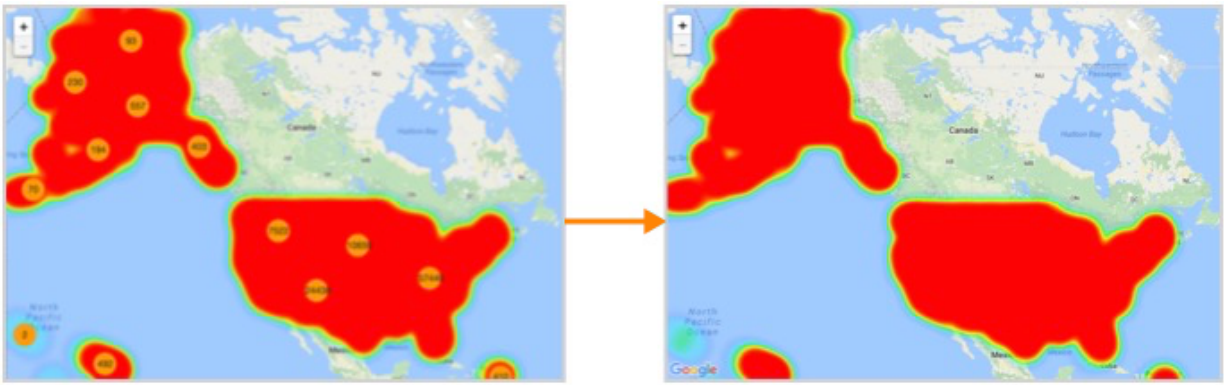

Similarly, the Mapbox map renders without the cluster option.

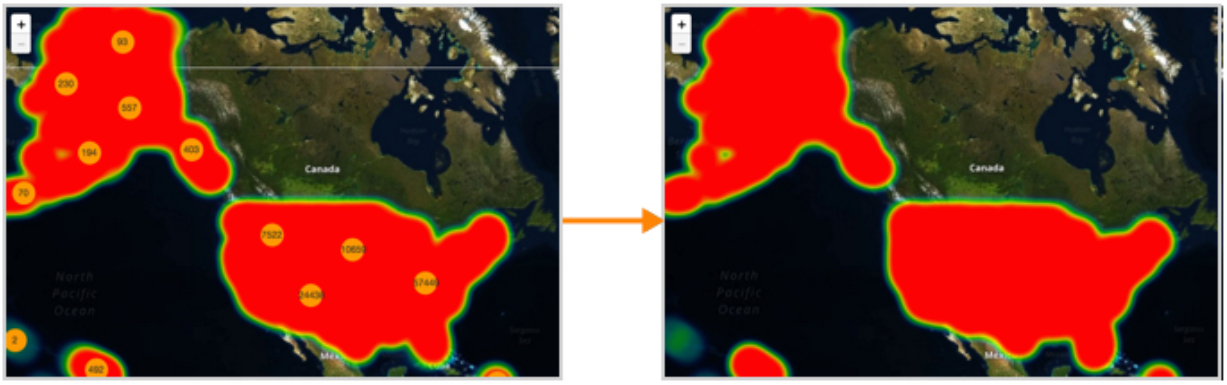

Changing cluster color

In an interactive map visual, Cloudera Data Visualization enables you to change the color of a cluster.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Cluster.

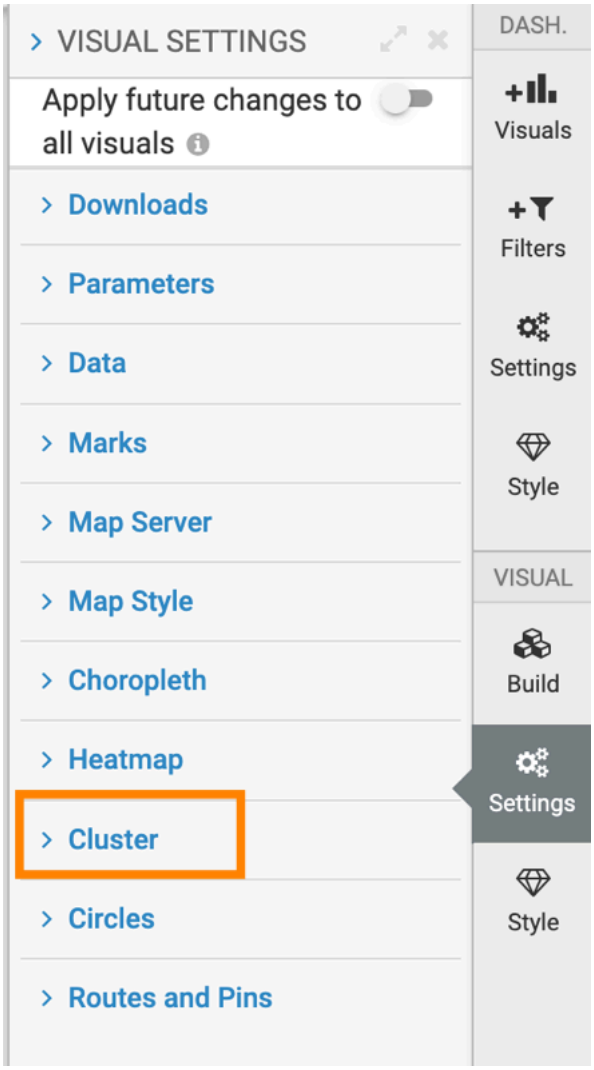

3. To change the color of Cluster, change the value of the Cluster Color option, either by changing the hexadecimal value, or by selecting the new color from the color palette.

Note the cluster color change from the default of #ff9900 to custom #42a9b9 on the Google map rendering. We turned off the Heatmap option.

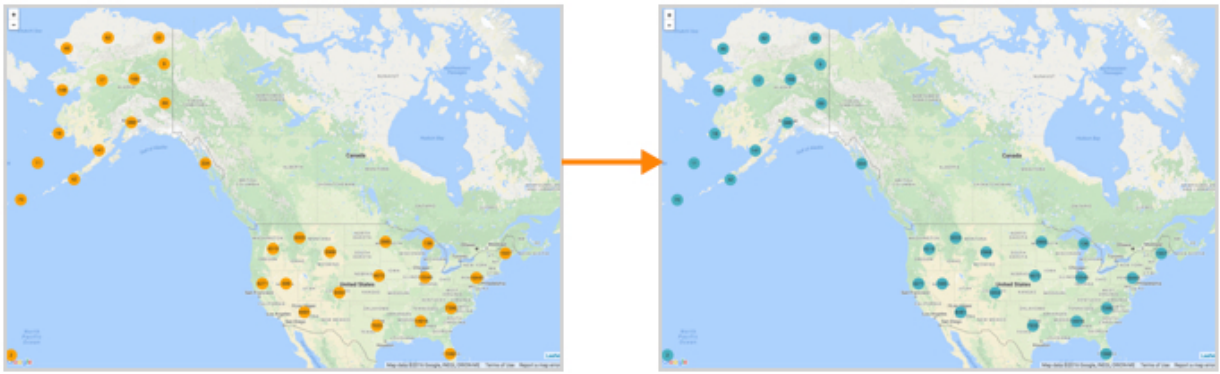

Similarly, note the color change for cluster layer on the Mapbox map.

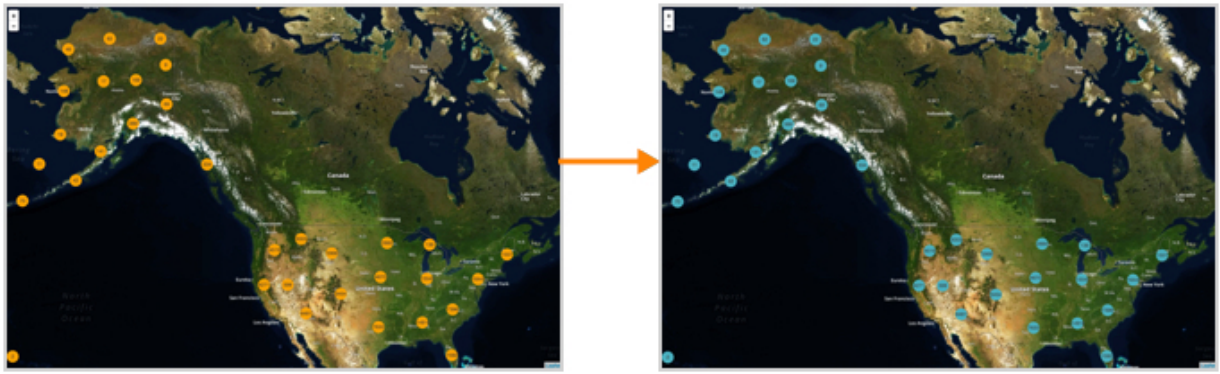

Changing cluster radius

In an interactive map visual, Cloudera Data Visualization enables you to change the radius of a cluster.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Cluster.

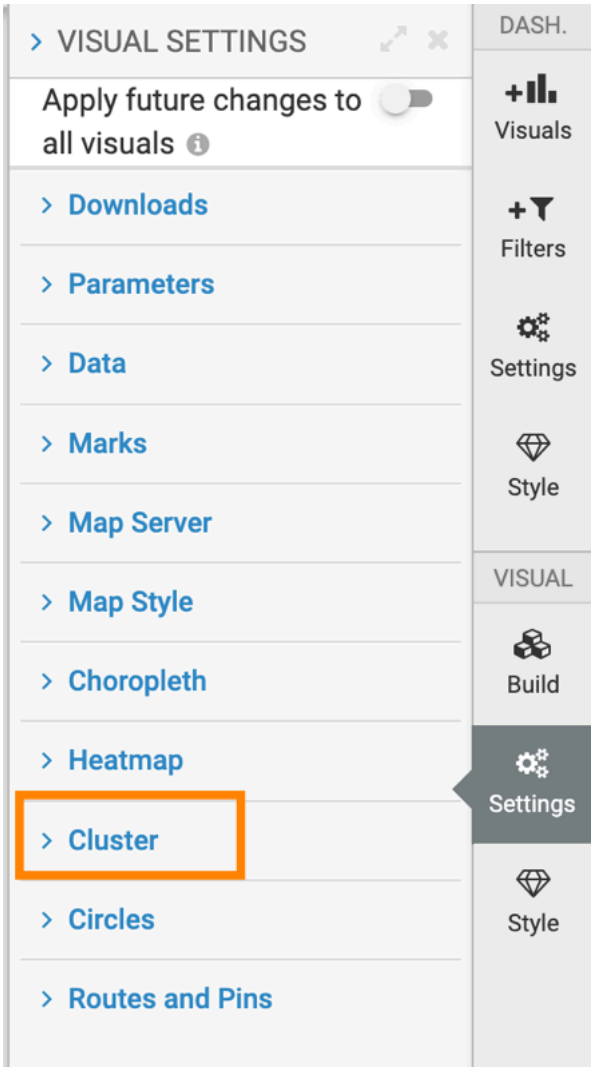

3. To change the clustering granularity for the map, you must adjust the cluster Radius option. The default value is 80, and the minimum value is 30.

The default cluster radius value is 80. Compare the results of setting cluster radius on Google Map to 30, 60, 90, and 120.

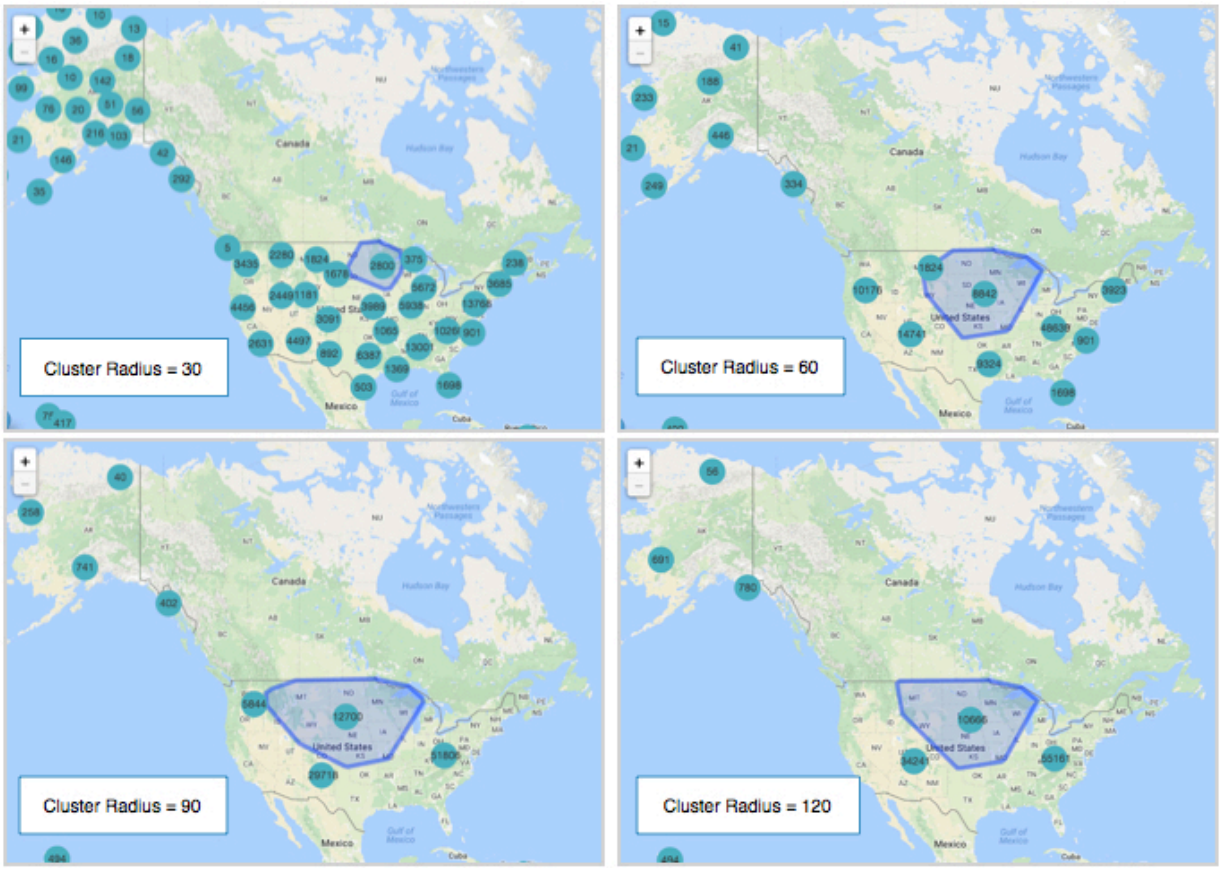

Similarly, note the change for the same cluster radius options on the Mapbox map.

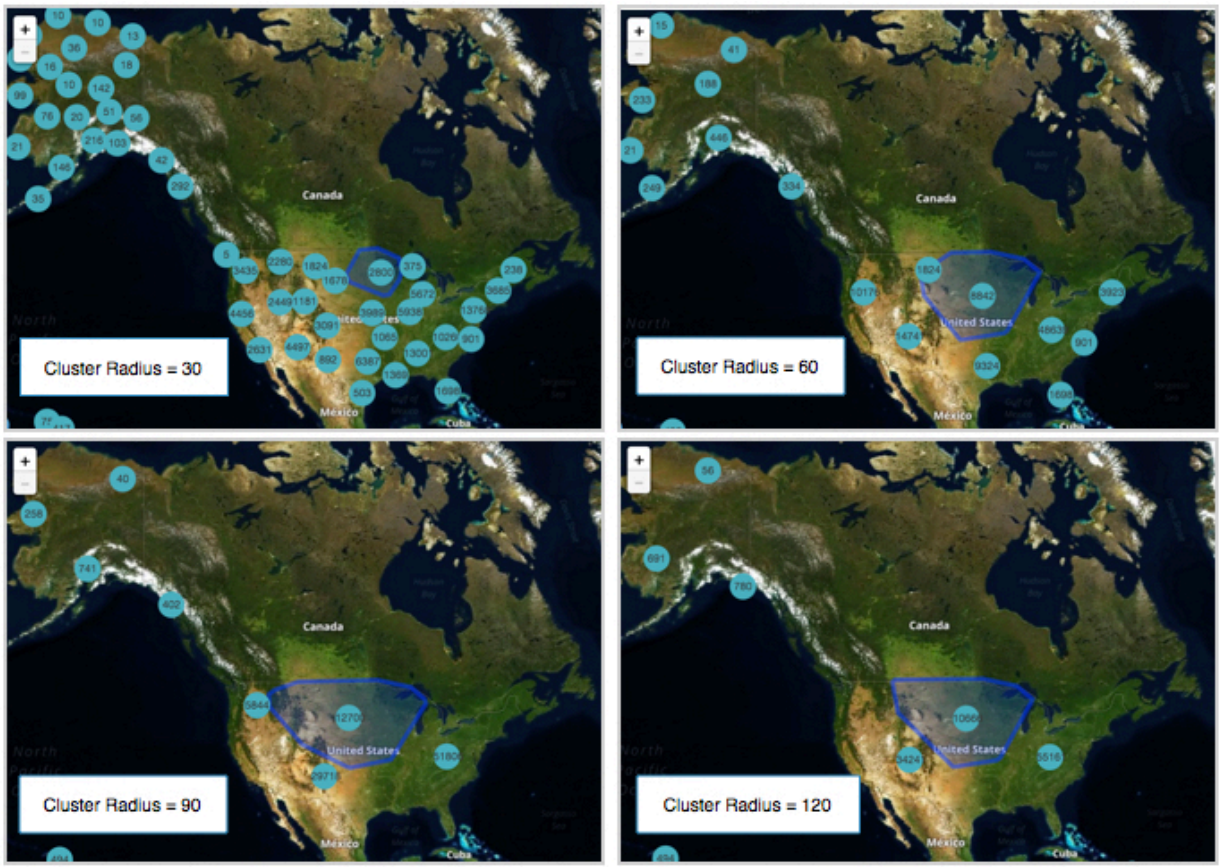

Customizing circles

In an interactive map visual, Cloudera Data Visualization enables you to change circle settings.

Circles paint circles that are linearly proportional to the value of the aggregate. When using one measure, both the size and color of the circle correspond to the value of the aggregate measure. When using two measures, the first measurement specifies the size of the circle, and the second measurement specifies its color.

Circles are not on by default.

Displaying circles

In an interactive map visual, Cloudera Data Visualization enables you to display circles.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Circles.

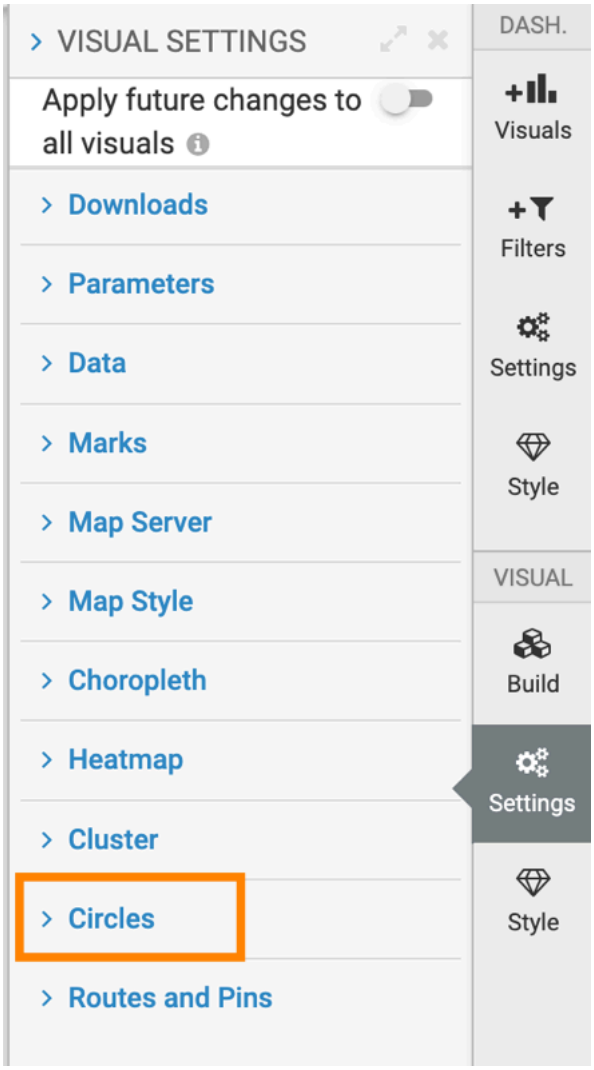

3. To show or hide Circles, select or unselect the Enable Circle option. This option is off by default.

   The following image shows the Interactive Google Map with the Circles option on.

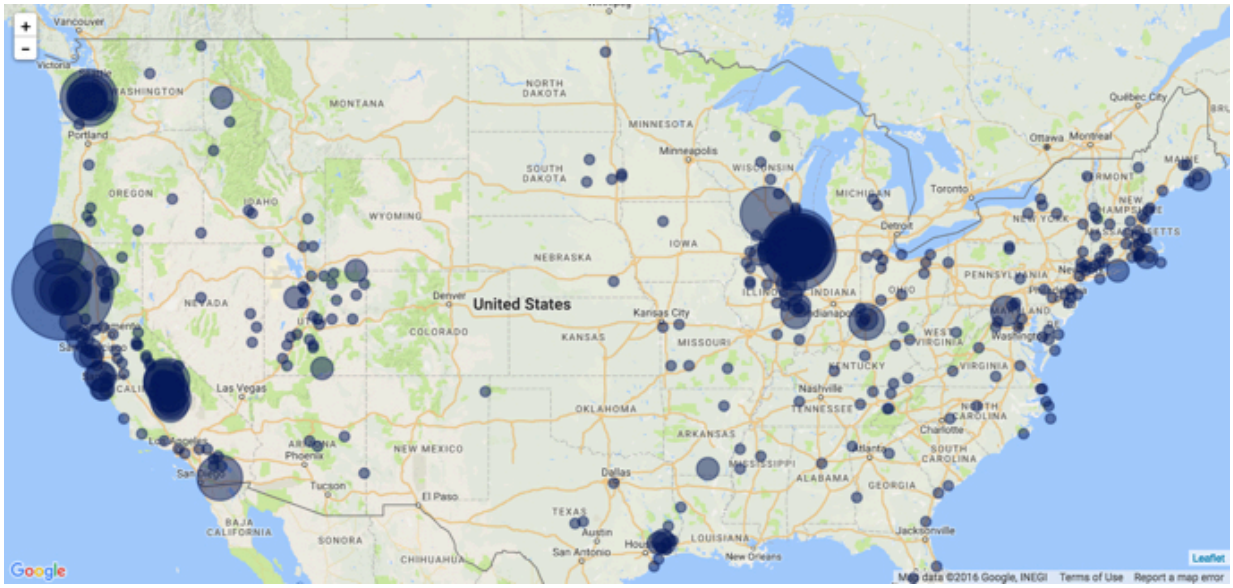

The following image shows the Mapbox Interactive Map with the Circles option on.

Displaying color legend

In an interactive map visual, Cloudera Data Visualization enables you to display color legend of circles.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Circles.

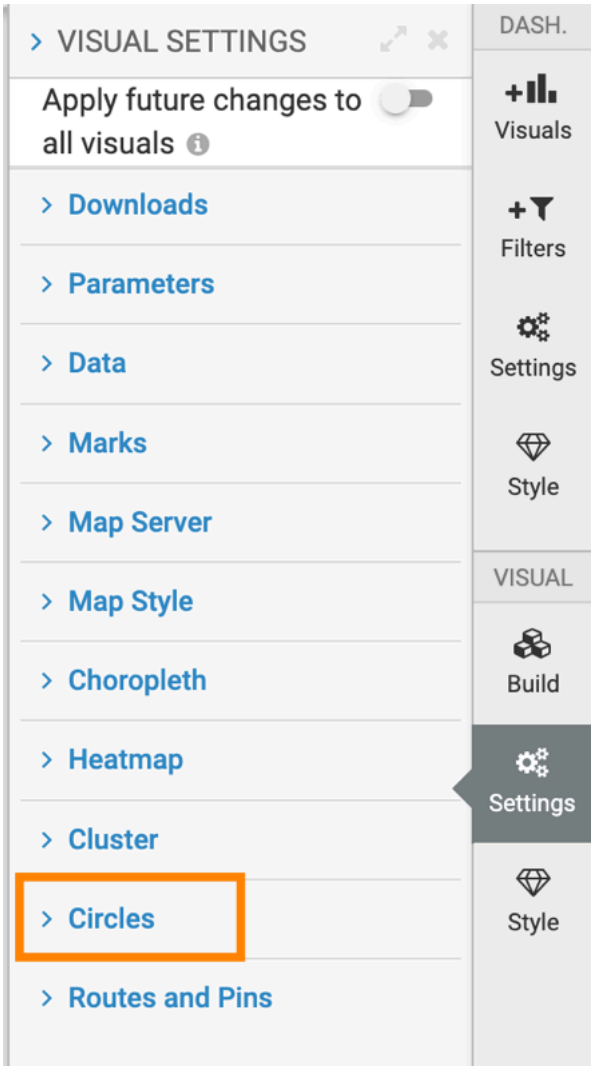

3. To show the color legend for Circles, select the Add Color Legend option.

   This option is off by default.

   Note: This setting is only available if the Colors shelf is populated. Make sure you have one aggregate field in the Colors shelf.

   The following image shows a Google Map with Circles, plotting two measures: elevation and count of features. You can see that the first measure appears as colors that you can check in the color legend, while the second measurement displays as size that you can see in the area legend.

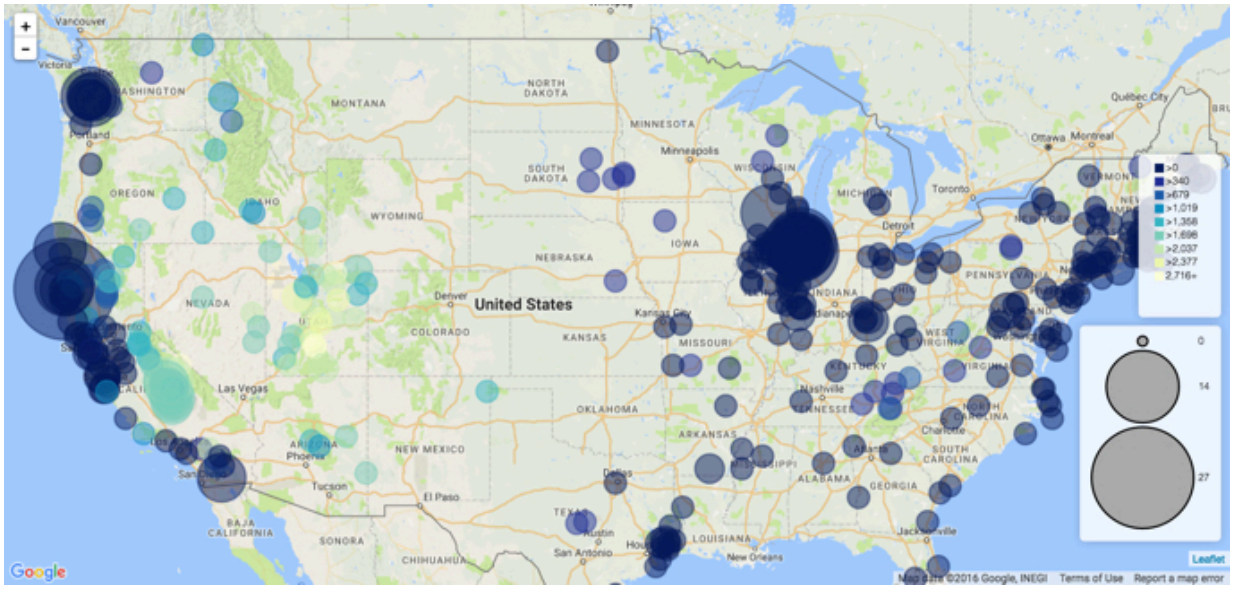

   The following image shows a Mapbox Map with Circles and two measures: elevation and count of features. Elevation appears as colors, and feature count is represented by the area of the circle.

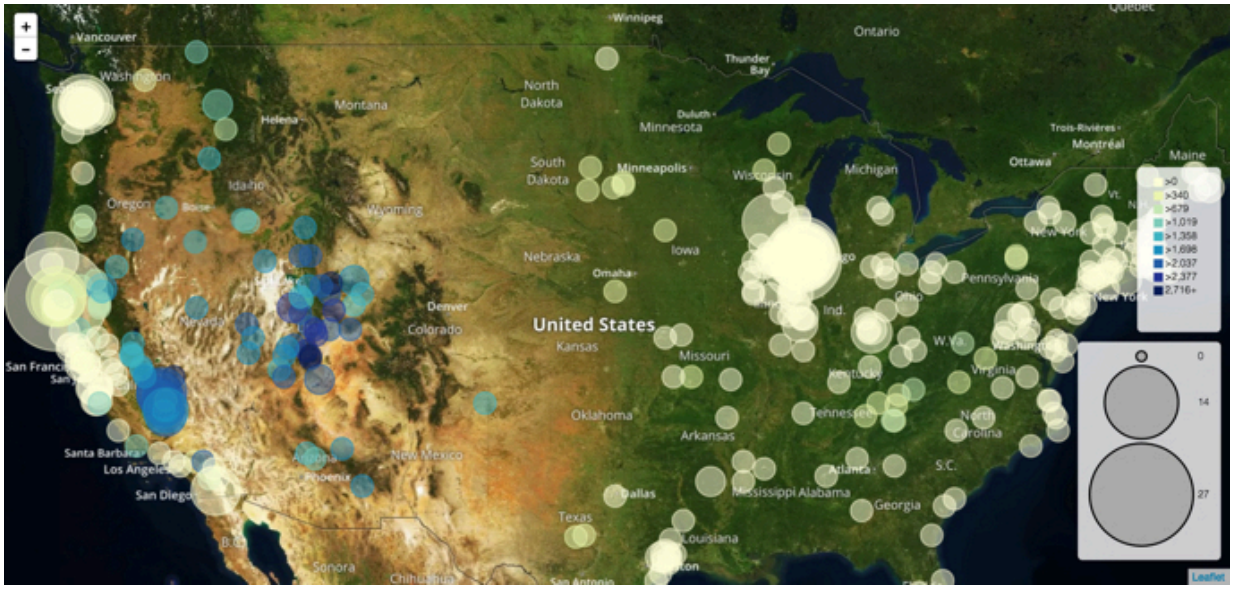

Displaying area legend

In an interactive map visual, Cloudera Data Visualization enables you to display area legend of circles.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Circles.

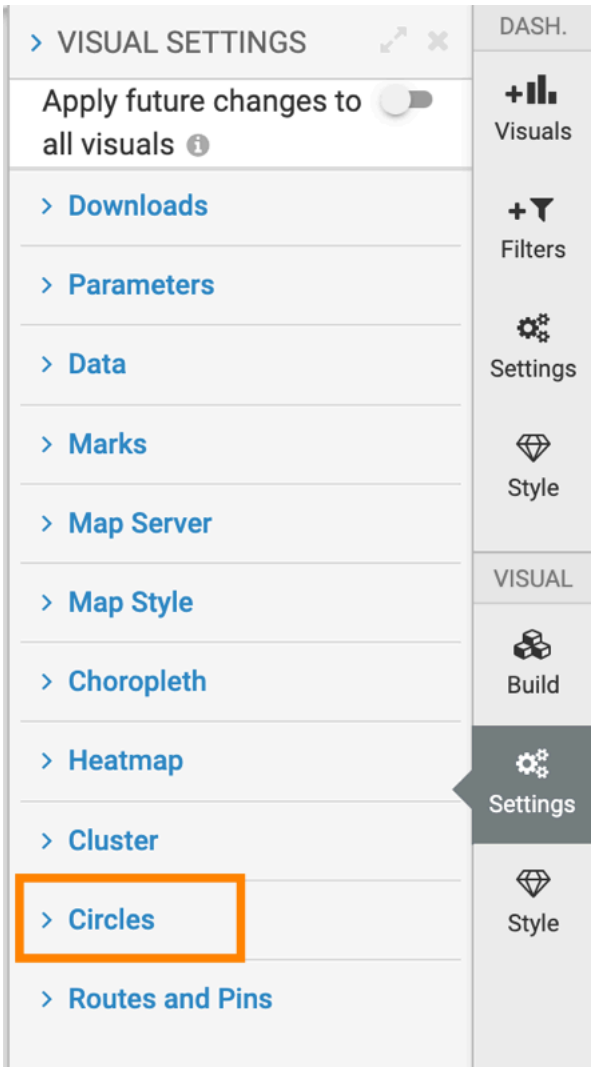

3. To show the area legend for Circles, select the Add Area Legend option.

   This option is off by default.

   Note: This setting is only available if the Measures shelf is populated. Make sure you have one aggregate field in the Measures shelf.

   The following image shows a Google Map with Circles where the area legend now appears on the lower-right corner of the map.

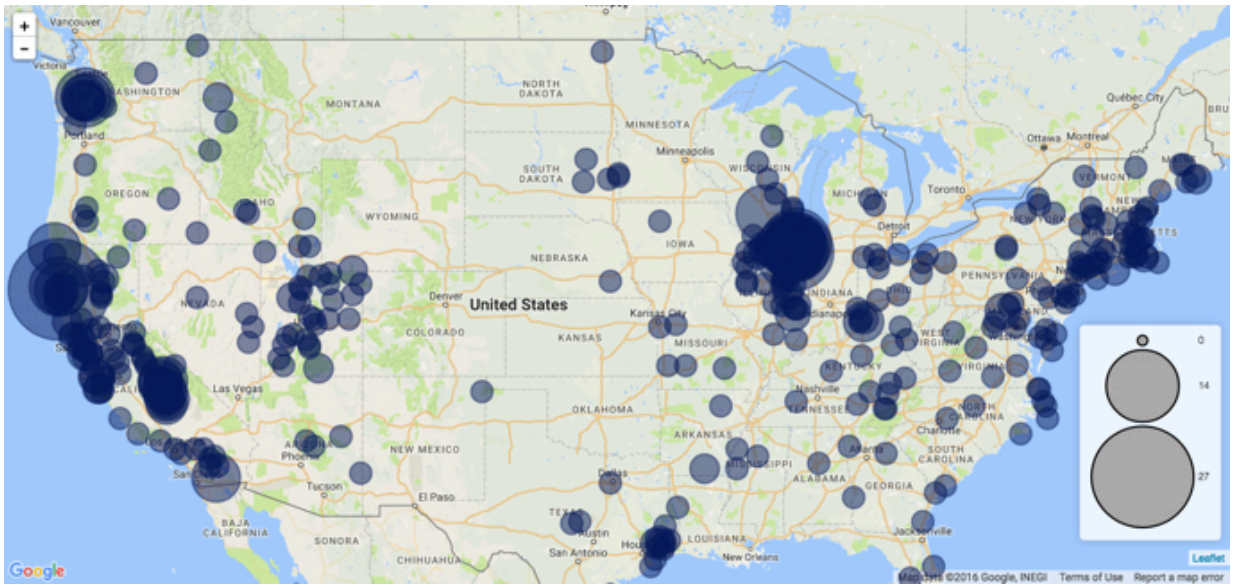

   The following image shows a Mapbox Map with Circles and an area legend.

Changing circle radius range

In an interactive map visual, Cloudera Data Visualization enables you to change the size of circles.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Circles.

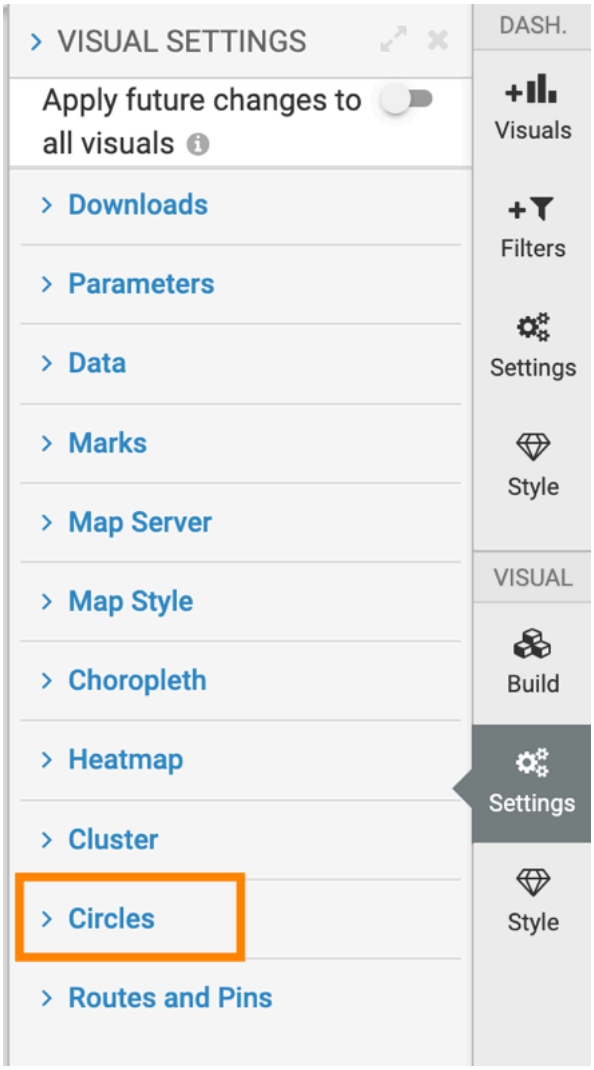

3. To change the range of sizes for Circles, adjust the minimum and maximum values in the Radius Range option.

   The default minimum is 1, and the default maximum is 100. Valid values are between these two numbers.

   The following image shows a Google Map with Circles, contrast the maximum size setting of circles from 24, to the new maximum setting of 5.

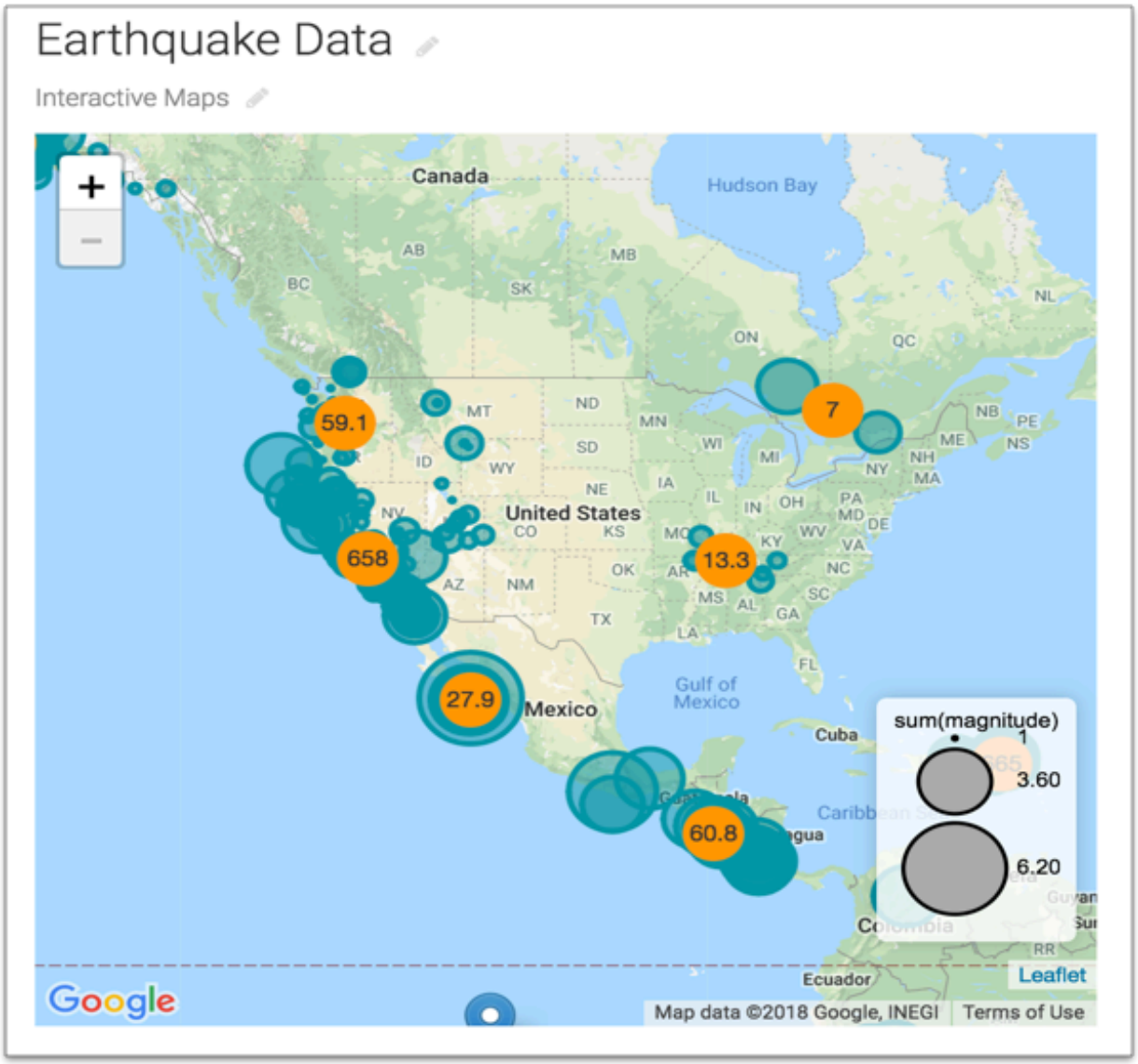
Earthquake Data
Interactive Maps
Canada
Hudson Bay
United States
Mexico
Gulf of Mexico
Cuba
Ecuador
59.1
658
27.9
60.8
13.3
7
sum(magnitude)
1
3.60
6.20
Google
Leaflet
Map data ©2018 Google, INEGI
Terms of Use

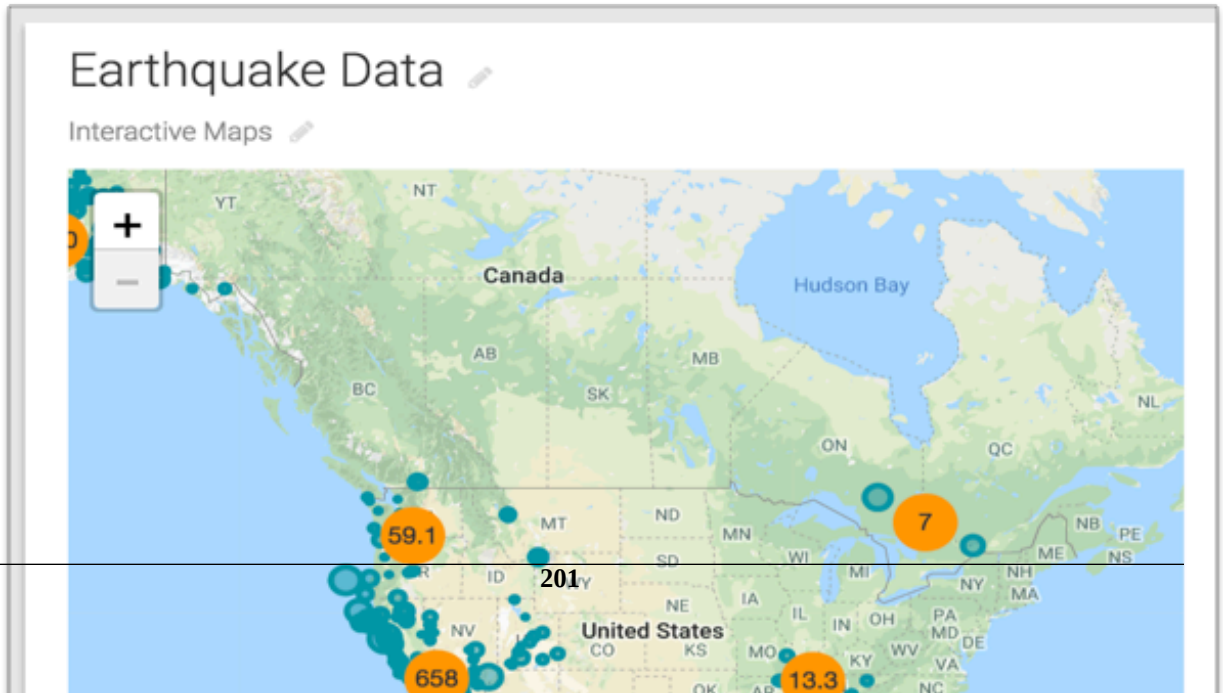
Earthquake Data
Interactive Maps
Canada
Hudson Bay
United States
59.1
658
13.3
7

The following image shows a Mapbox with Circles, contrast the maximum size setting of circles from 24, to the new maximum setting of 5.

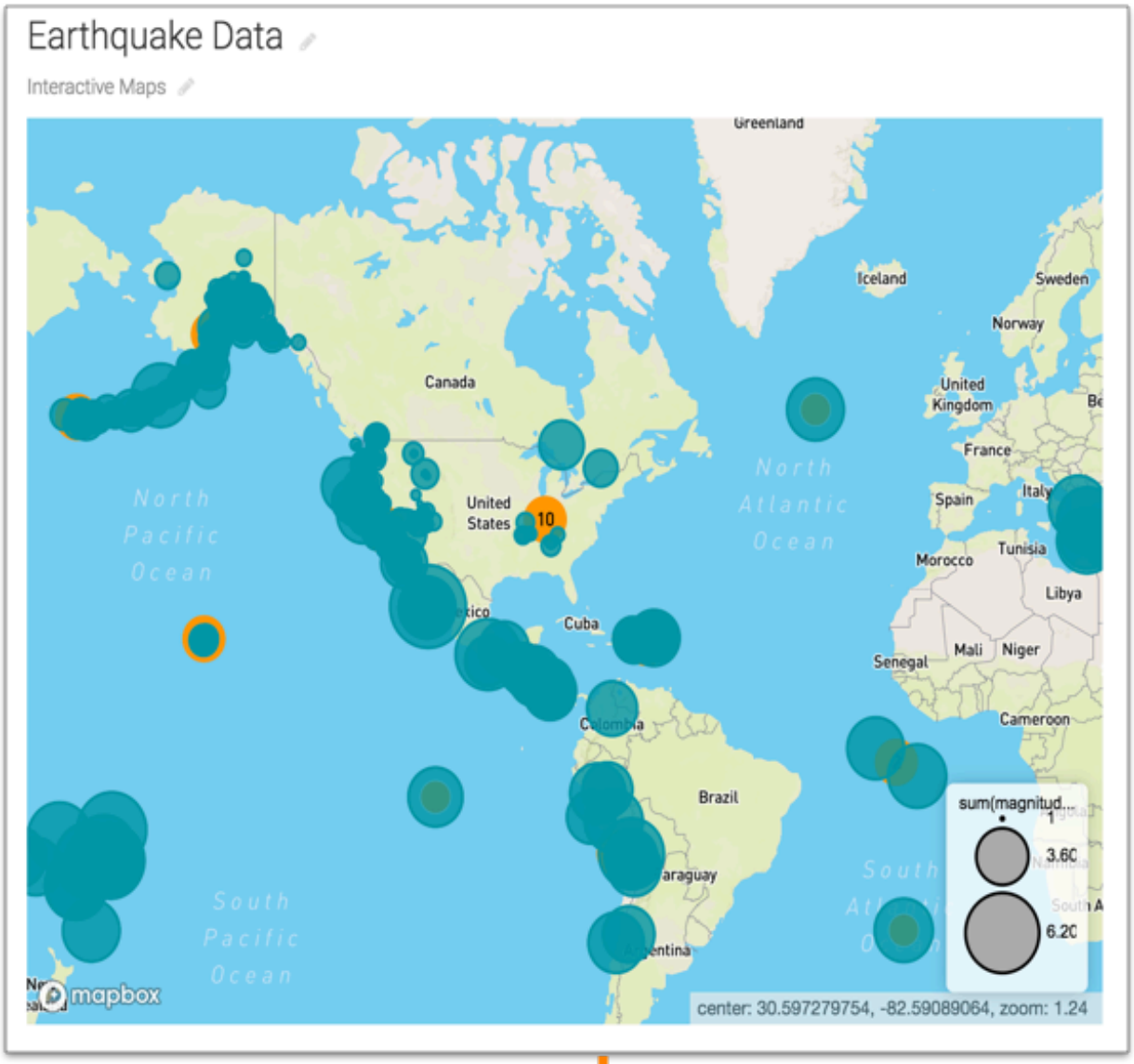
Earthquake Data
Interactive Maps
Greenland
Iceland
Sweden
Norway
Canada
United Kingdom
France
Spain
Italy
North Pacific Ocean
North Atlantic Ocean
United States
10
Morocco
Tunisia
Libya
Cuba
Mali
Niger
Senegal
Cameroon
Brazil
sum(magnitud...
1
3.60
6.20
South Pacific Ocean
mapbox
center: 30.597279754, -82.59089064, zoom: 1.24

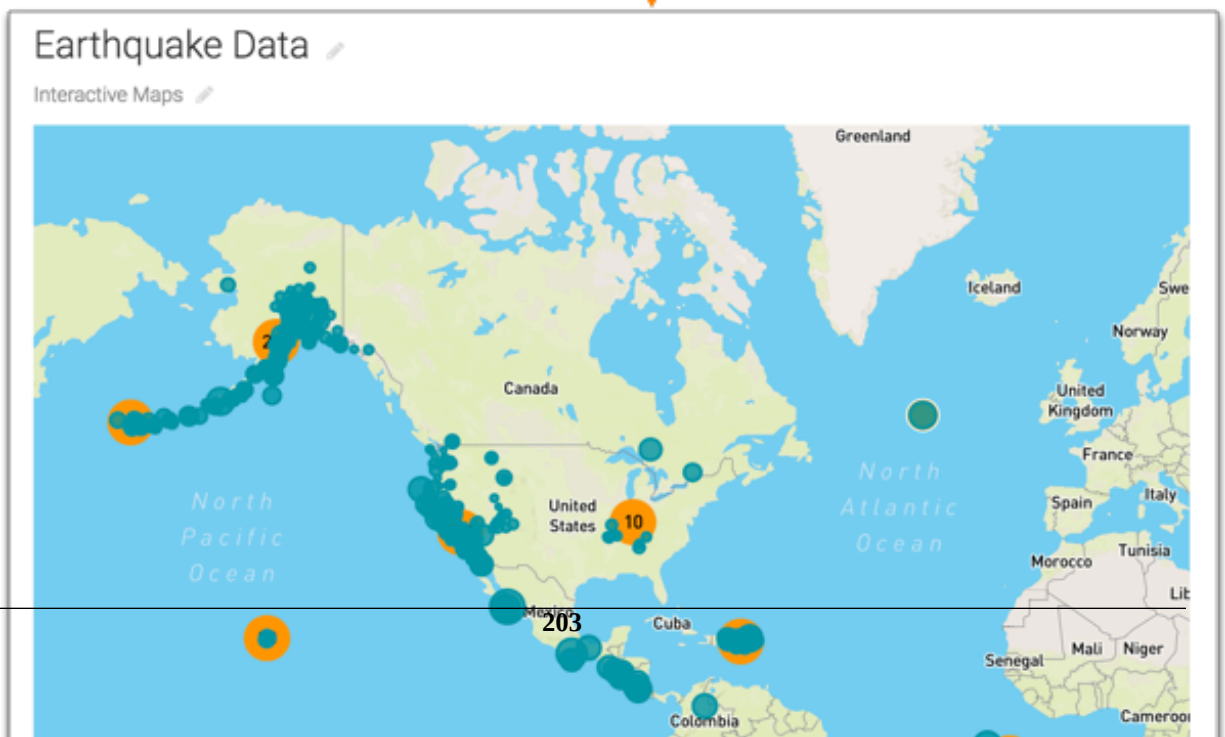
Earthquake Data
Interactive Maps
Greenland
Iceland
Norway
Canada
United Kingdom
France
North Pacific Ocean
North Atlantic Ocean
United States
10
Spain
Italy
Morocco
Tunisia
Cuba
Senegal
Mali
Niger
Colombia

Changing visible zoom levels in circles

Sometimes, it is not possible to display data legibly at a given zoom level. Cloudera Data Visualization enables you to adjust the visible zoom levels in circles.

About this task

In an interactive map visual on MapBox, it is recommended to adjust the zoom level within the minimum and maximum zoom levels. Valid values are 0 through 22. For example, if the zoom level is set to more than the maximum level, heatmap will not be visible.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Circles.

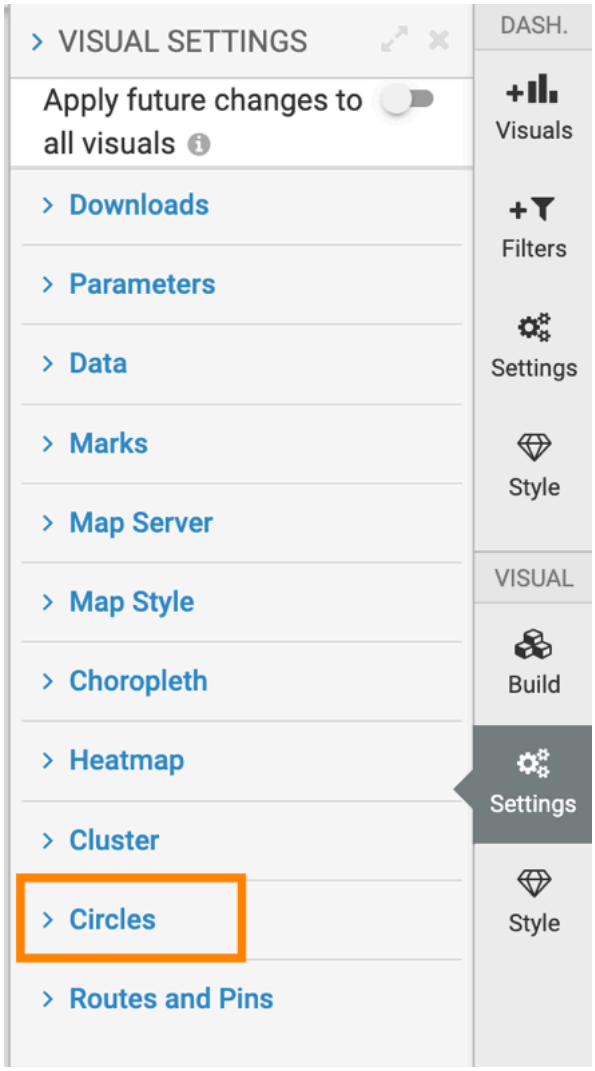

3. To adjust the zoom level of a heatmap, navigate to the Heatmap menu under Settings, and adjust the minimum and maximum values in the Visible Zoom Levels option.

   In the following example, you can see that when the zoom level is set to a value less than the maximum level 4.95, circles are visible. However, if the zoom level is set to more than the maximum level 5.02, circles are not visible.

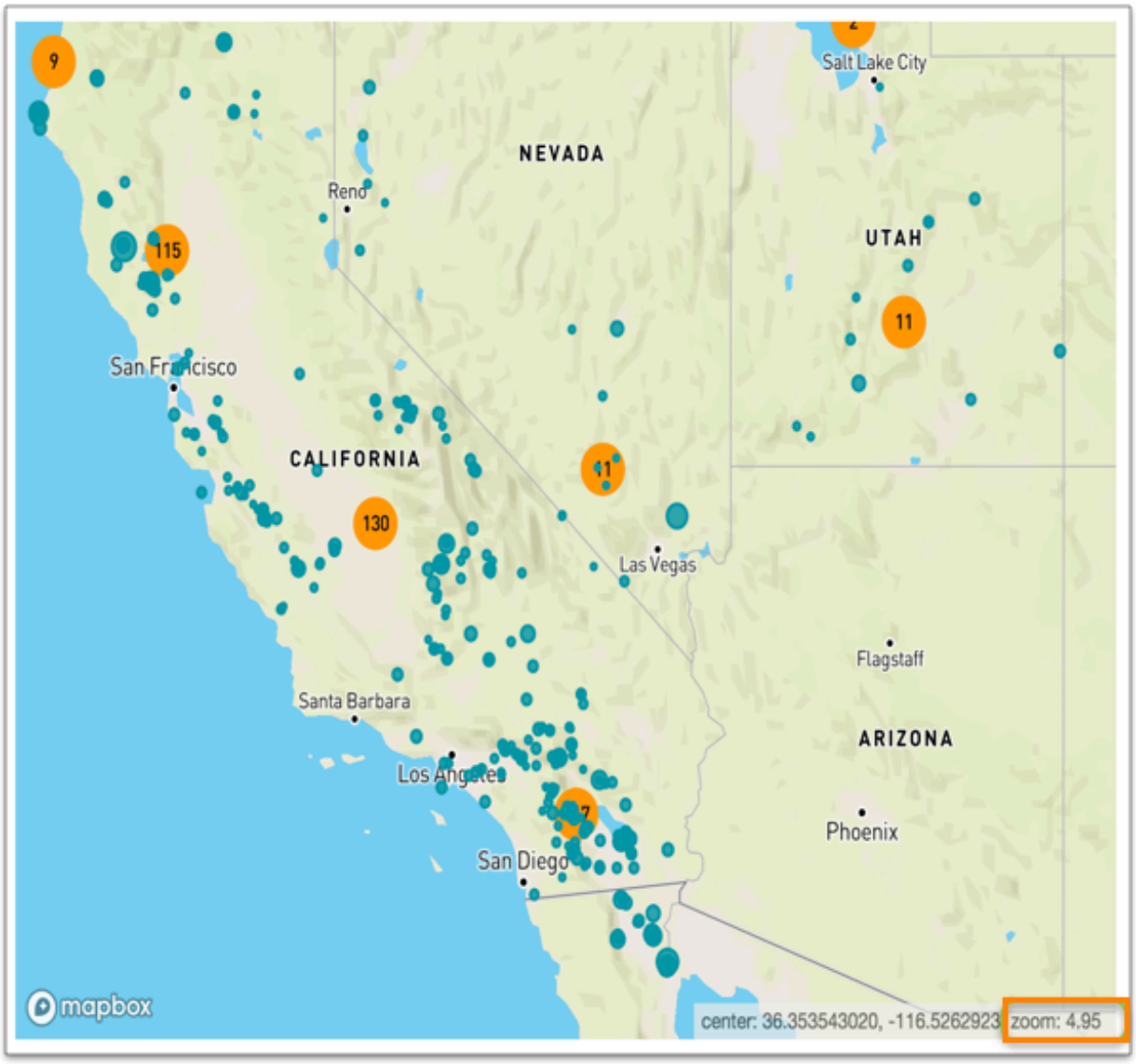
9
Salt Lake City
NEVADA
Reno
115
UTAH
11
San Francisco
CALIFORNIA
11
130
Las Vegas
Flagstaff
Santa Barbara
ARIZONA
Los Angeles
Phoenix
San Diego
mapbox
center: 36.353543020, -116.5262923 zoom: 4.95

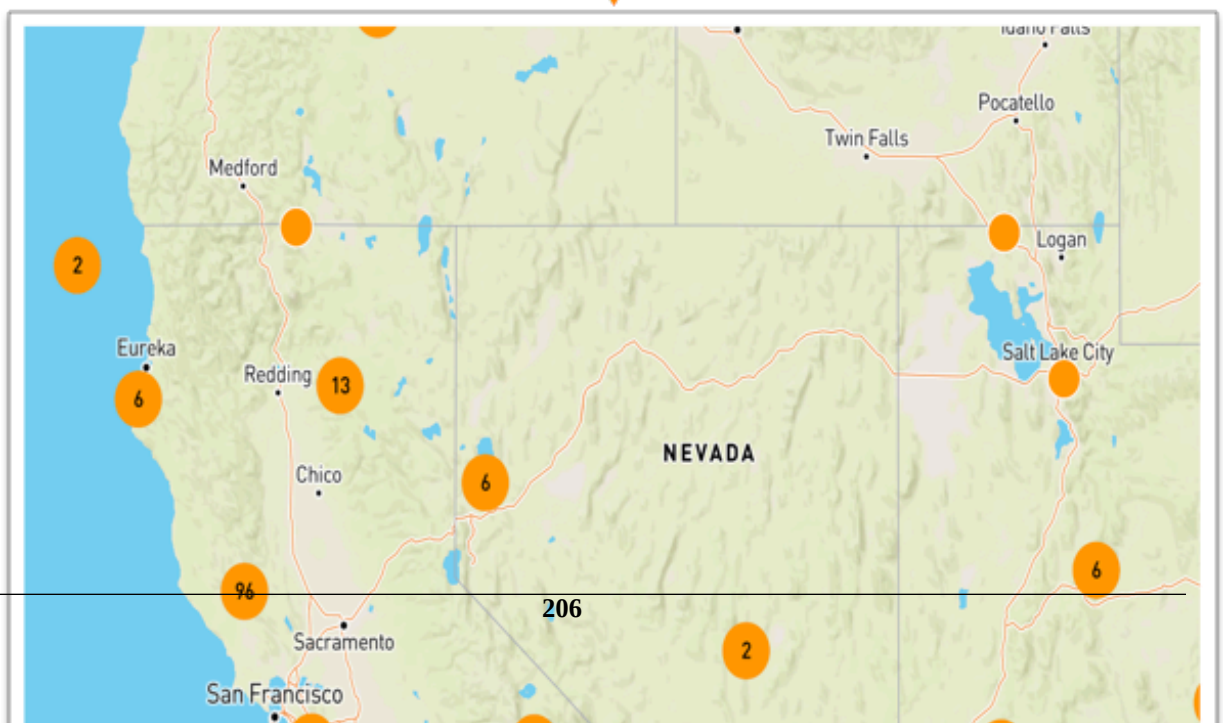
Pocatello
Twin Falls
Medford
Logan
2
Eureka
Salt Lake City
Redding
13
6
NEVADA
Chico
6
6
96
Sacramento
2
San Francisco

Customizing routes and pins

In an interactive map visual, you can plot a route and pins for each point in the route.

In addition to the following scenarios of using the Routes/Pinsoptions by itself, it is often informative to combine it with another option, such as heatmap.

Circles are not on by default.

Enabling routes

In an interactive visual, you can choose show and hide routes.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Circles.

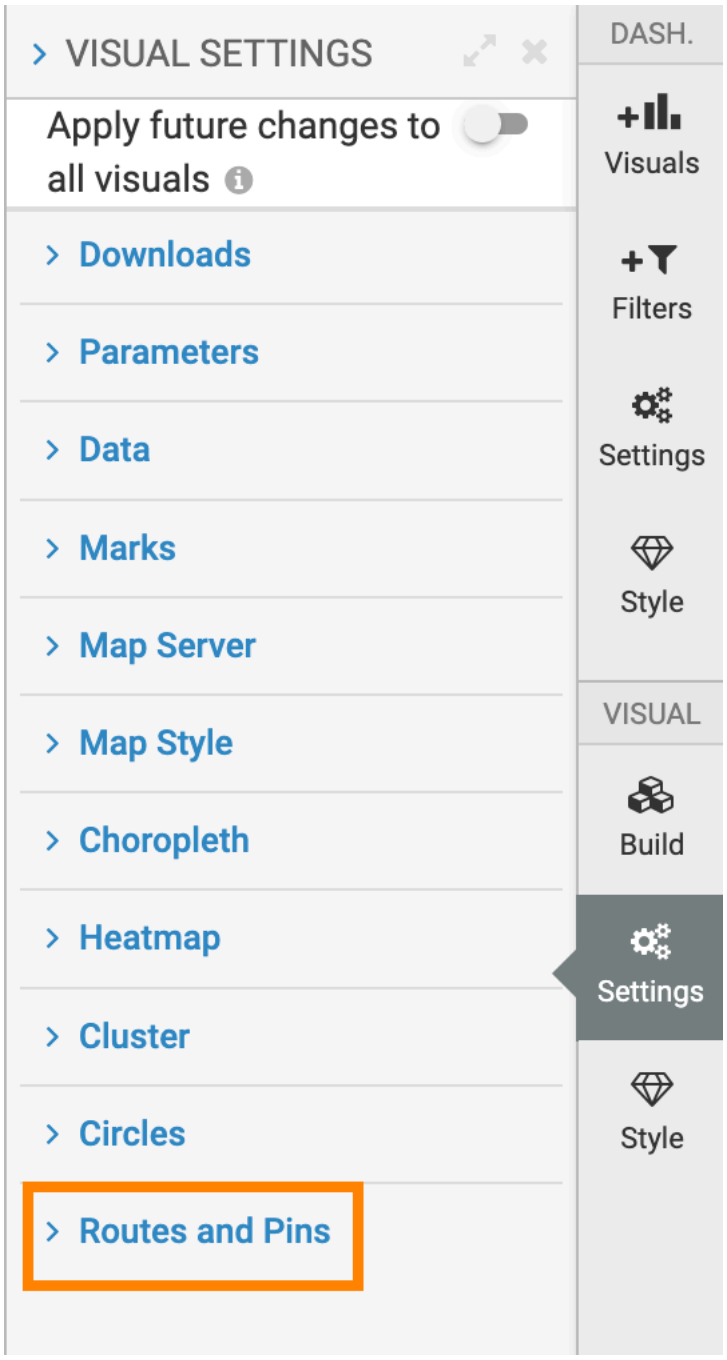

3. In an Interactive Map visual, to show or hide Routes, select or unselect the Enable Routes option.

   This option is off by default.

   The following image shows a Google Map with the Routes option.

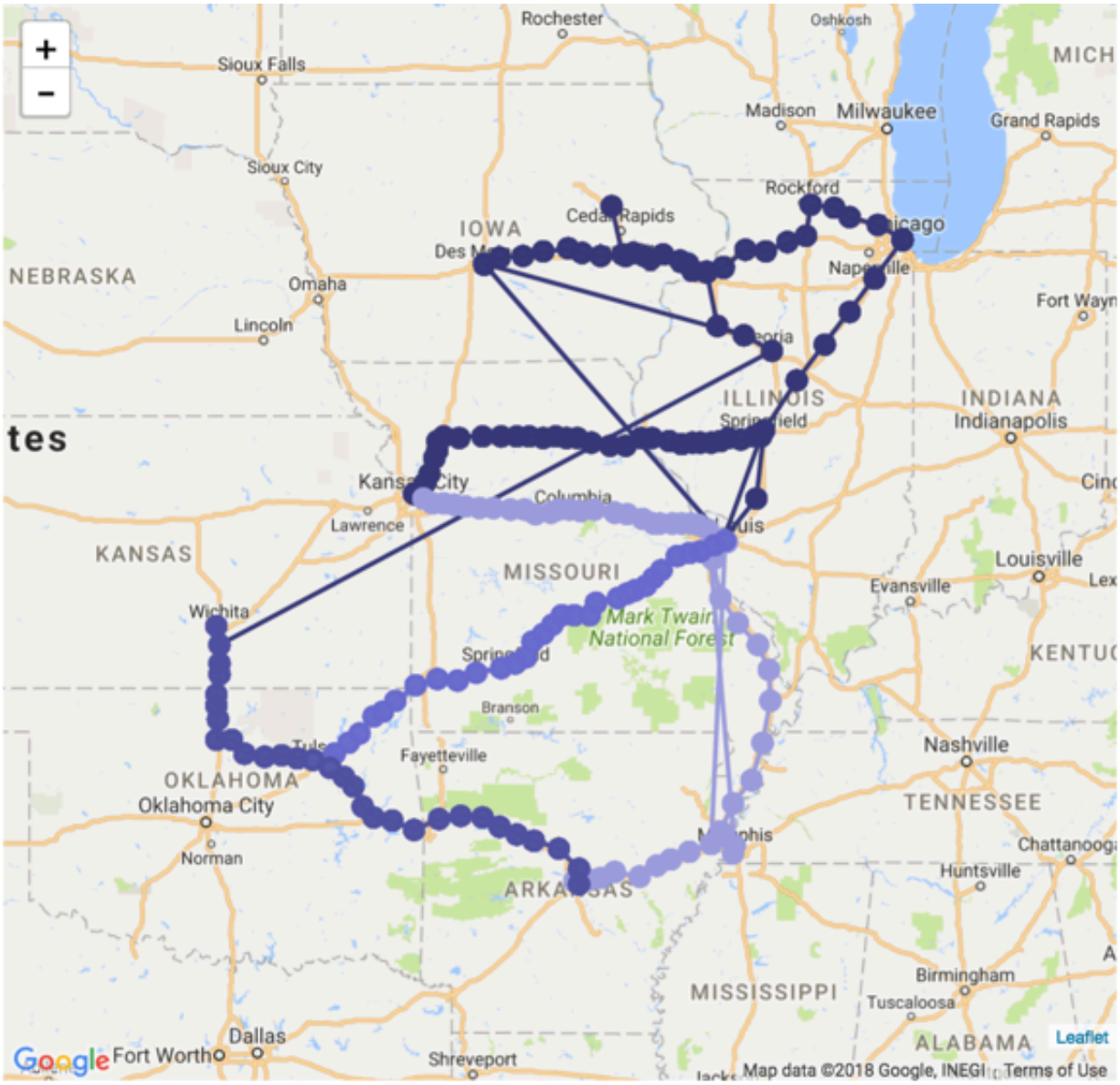

   The following image shows a Mapbox Interactive Map with the Routes option.

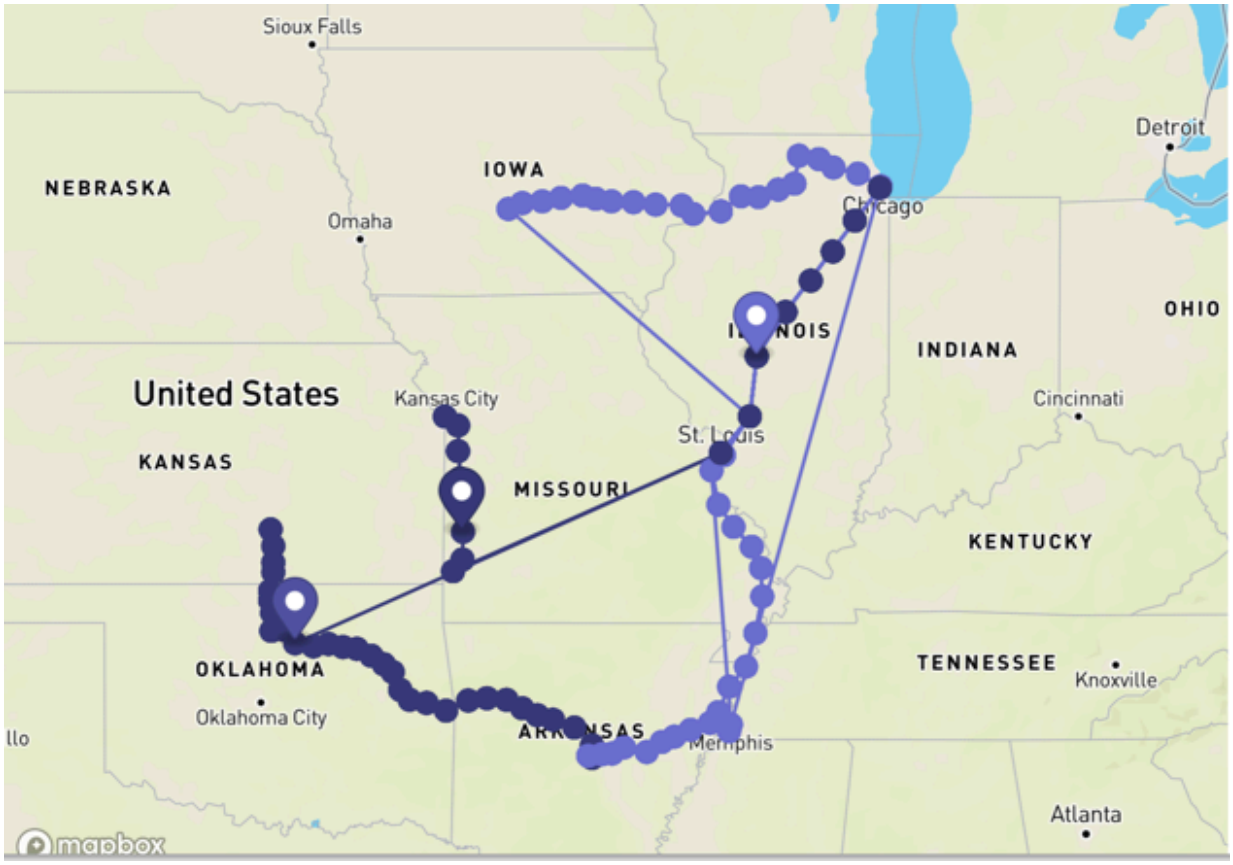

Enabling pins

In an intercative visual, you can shoa and hide pins.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Routes/Pins.

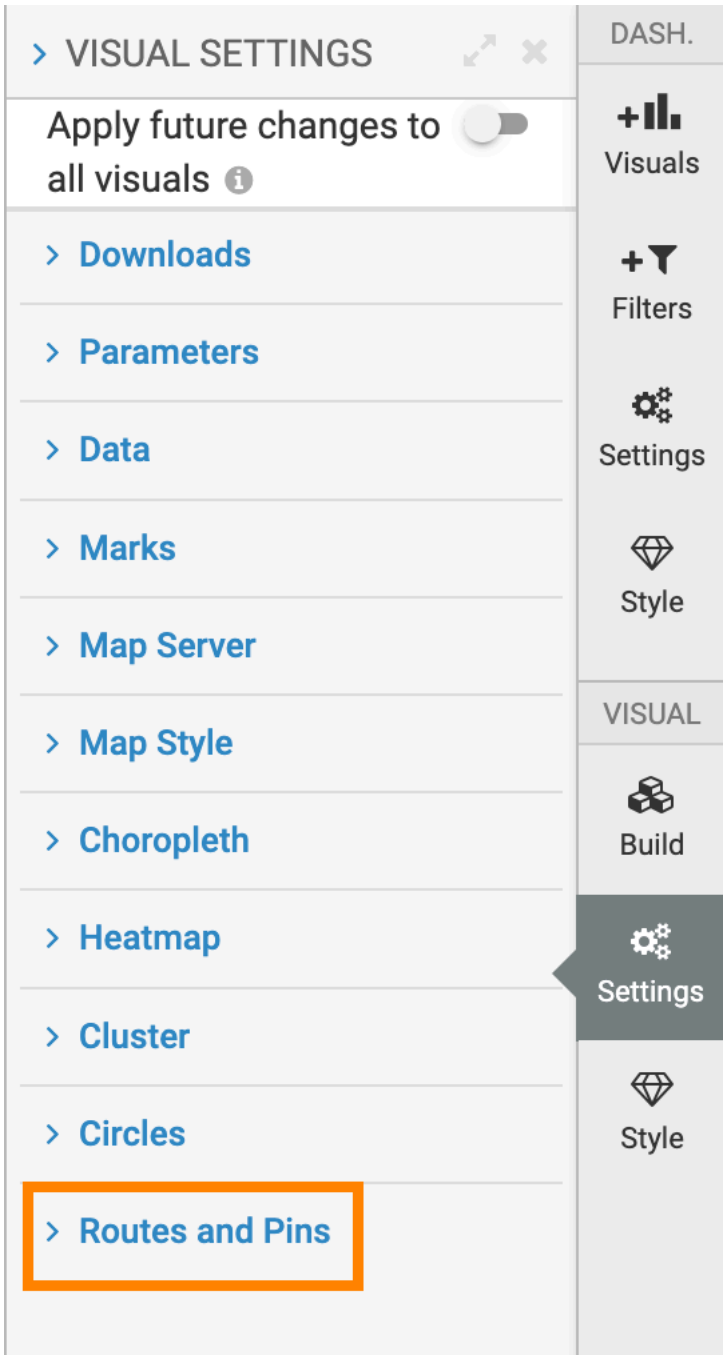

3. In an interactive map visual, to show or hide Pins, select or unselect the Enable Pins option.

   This option is off by default.

   The following image shows a Google Map with pins enabled.

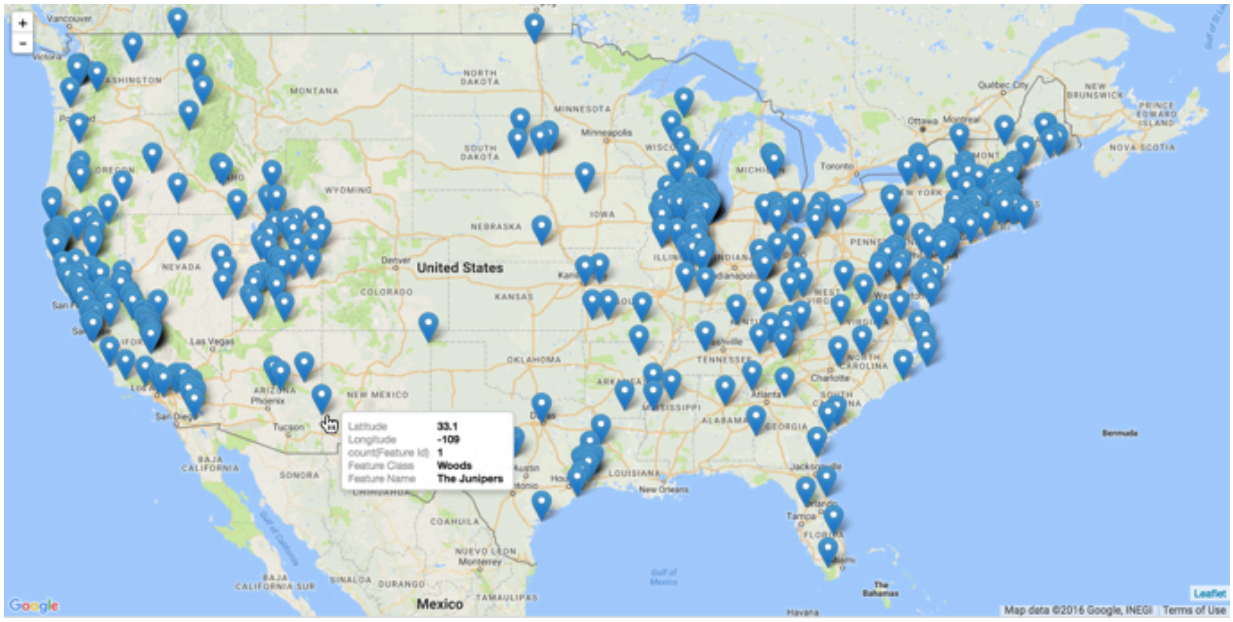

   The following image shows a Mapbox with pins enabled.

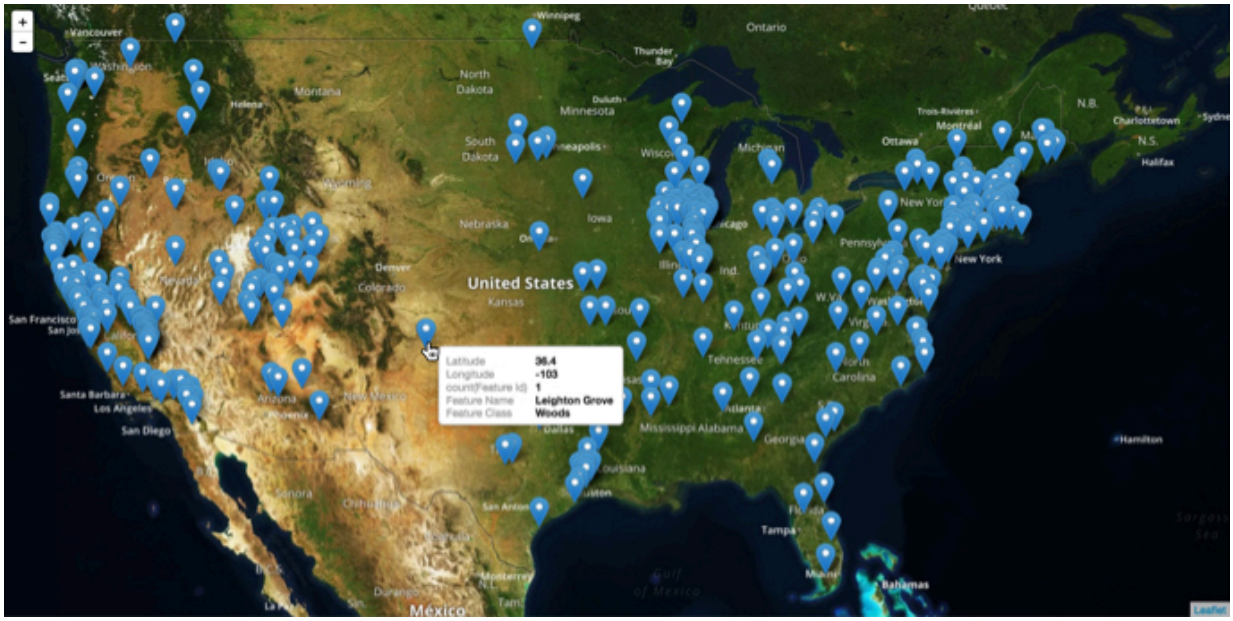

Selecting pin types

In a map route, you can choose pin types from a dropdown or add your own custom pin.

Before you begin

Before you select a pin type in an interactive map visual, make sure you enable the pins. See Enabling Pins.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Routes/Pins.

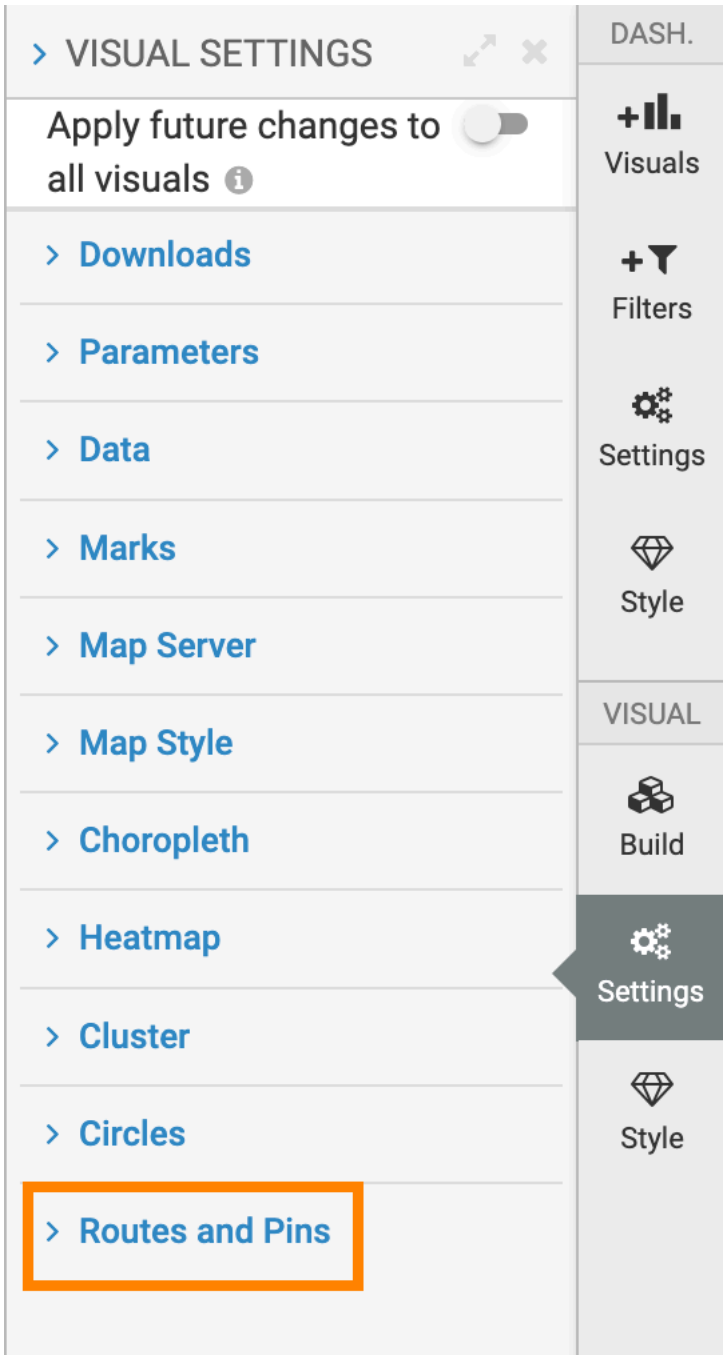

3. Select one of the following pin types from the Pin Type dropdown menu: Default, Circle, and Custom.

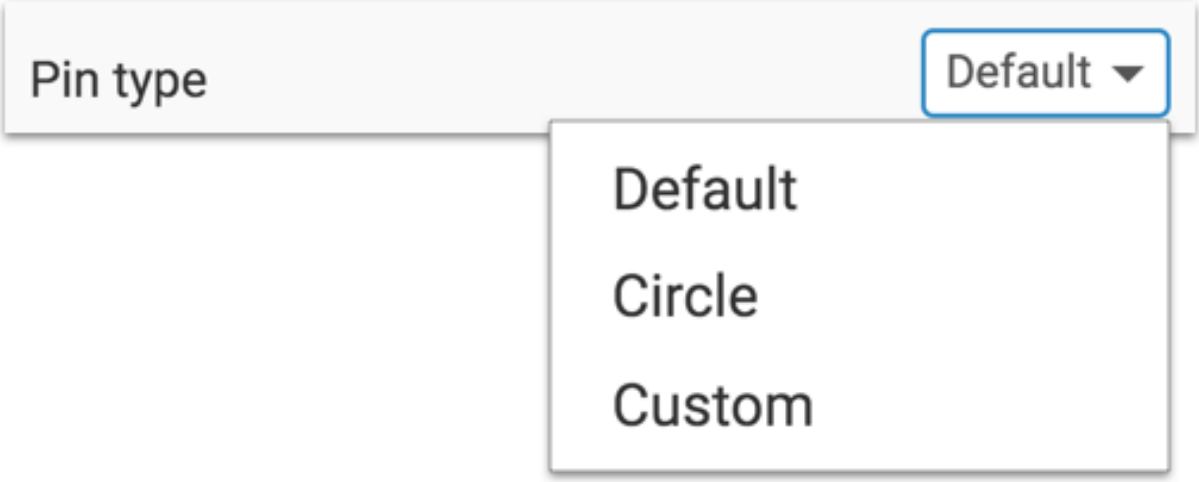

The following image shows the Default pin type.

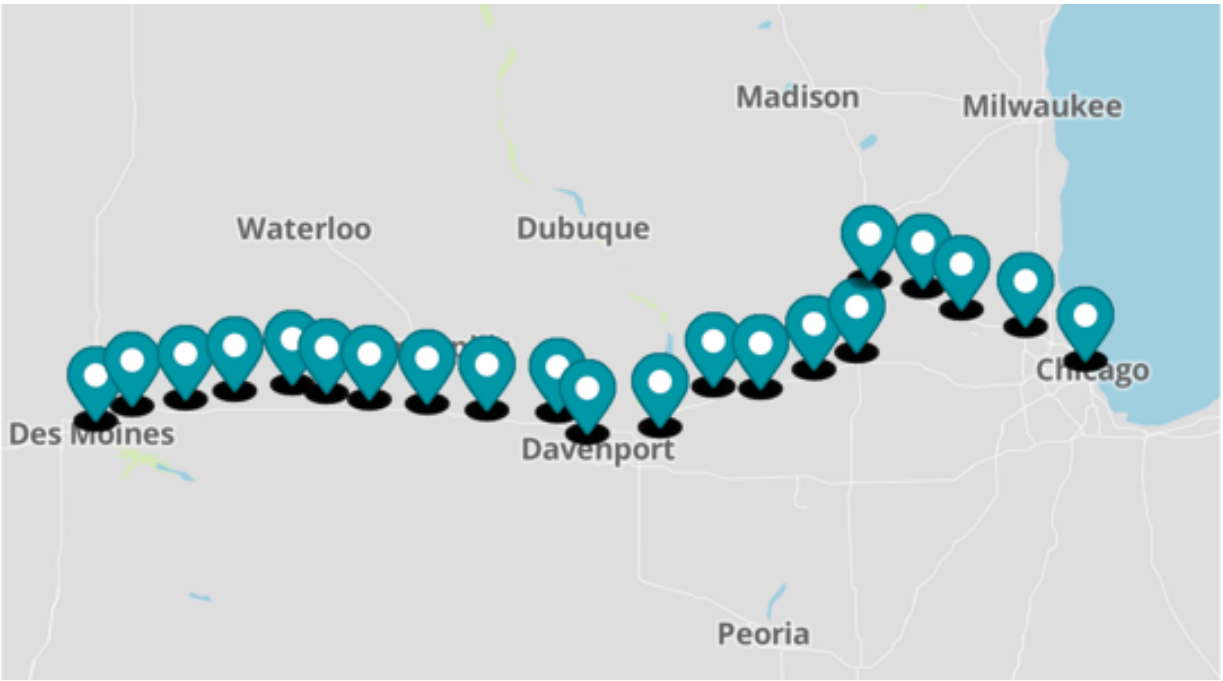

The following image shows the Circle pin type.

The following image shows trucks as a Custom pin type. To add a custom pin type, see Specifying a custom icon.

Related Information

Enabling pins

Selecting start pin type

In a map route, you can choose an existing start pin type from a dropdown or add your own custom pin.

Before you begin

In an Interactive Map visual, before you select the Start Pin Type option, make sure you enable routes. Once enabled, the route shows a default route and a default pin at the end of the route.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Routes/Pins.

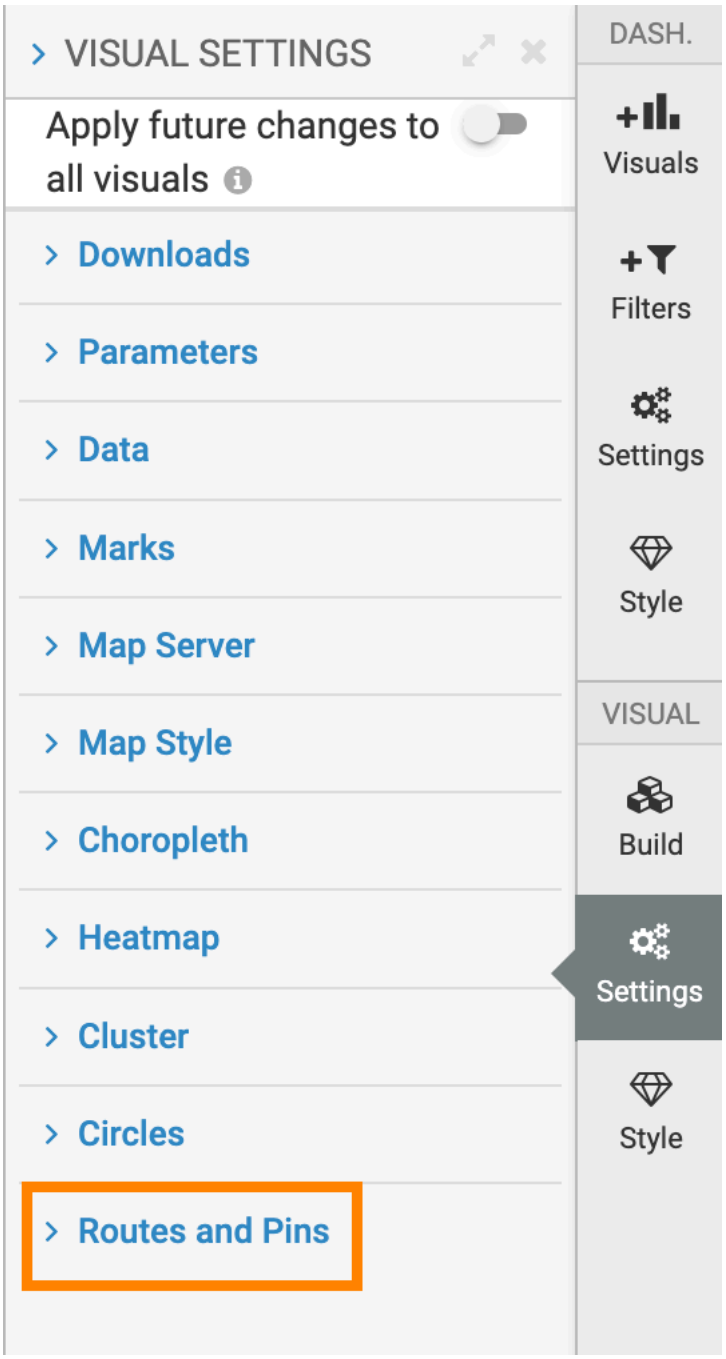

3. To change the start pin type, you can select one of the following pin types from the Start Pin Type dropdown menu: Default, Circle, None, and Custom.

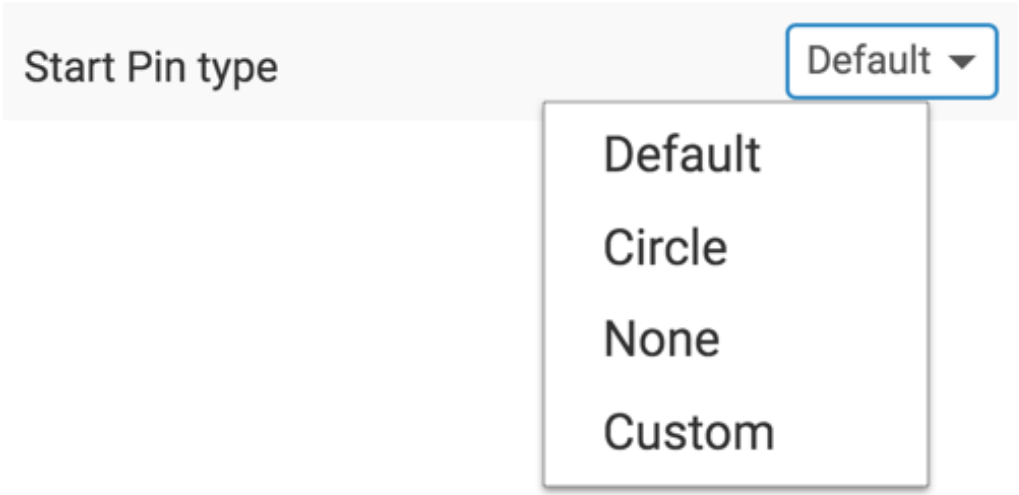

a) Select Circle from the dropdown menu to display the start pin as a circle.

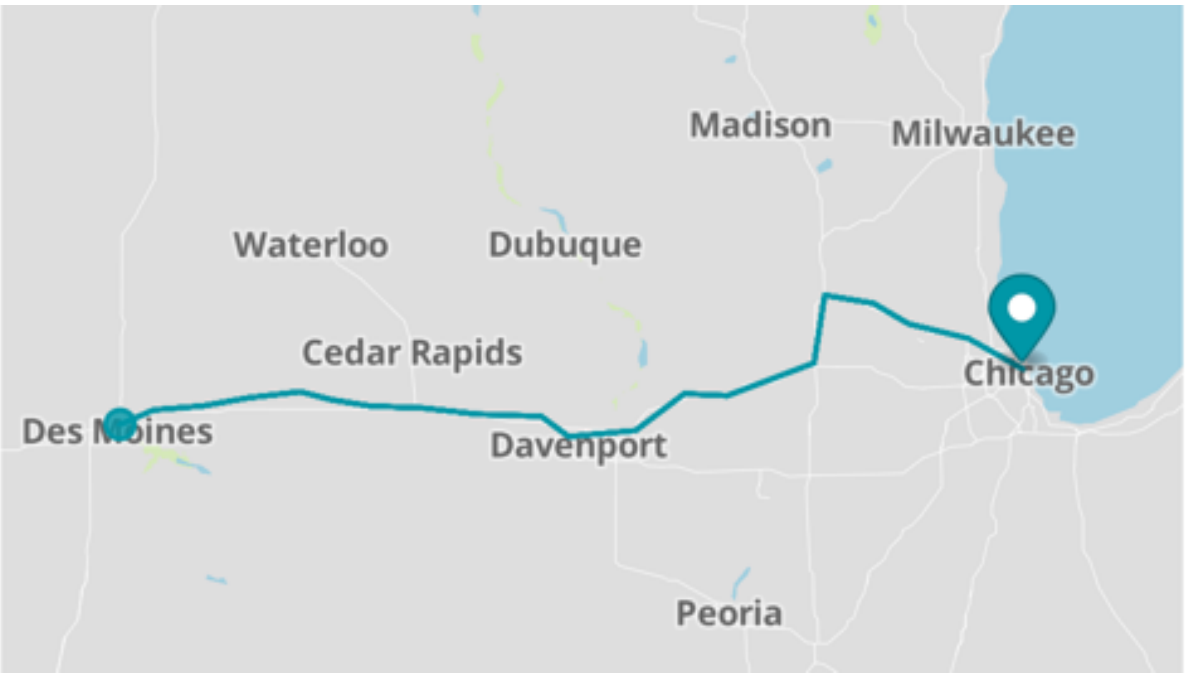

b) Select Custom from the dropdown menu to display the start pin as a truck.

To add a custom icon, see *Specifying a Custome Icon*.

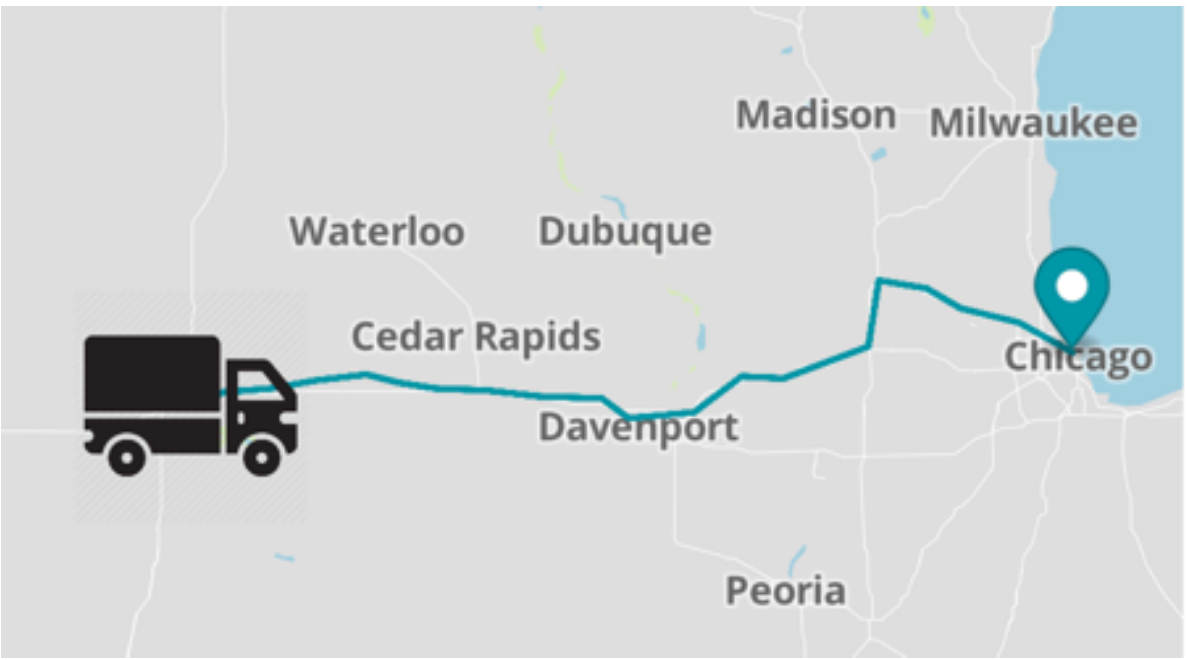

c) Select None from the dropdown menu to remove the start pin.

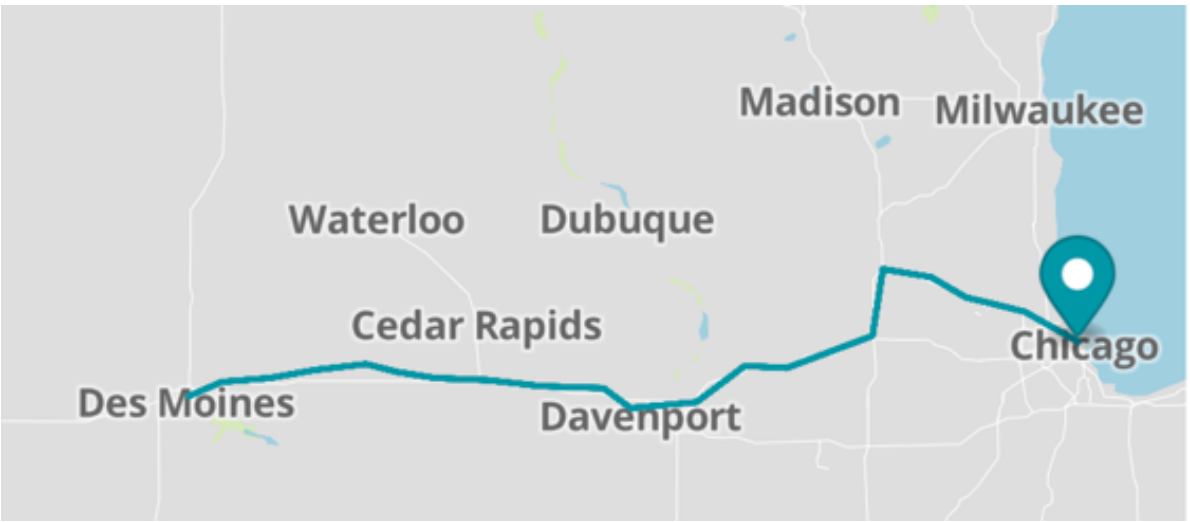

Related Information

Specifying a custom icon

Selecting middle pin type

In a map route, you can choose an existing middle pin type from a dropdown or add your own custom pin.

Before you begin

In an Interactive Map visual, before you select the Middle Pin Type option, make sure you enable routes. Once enabled, the route shows a default route and a default pin at the end of the route.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Circles.

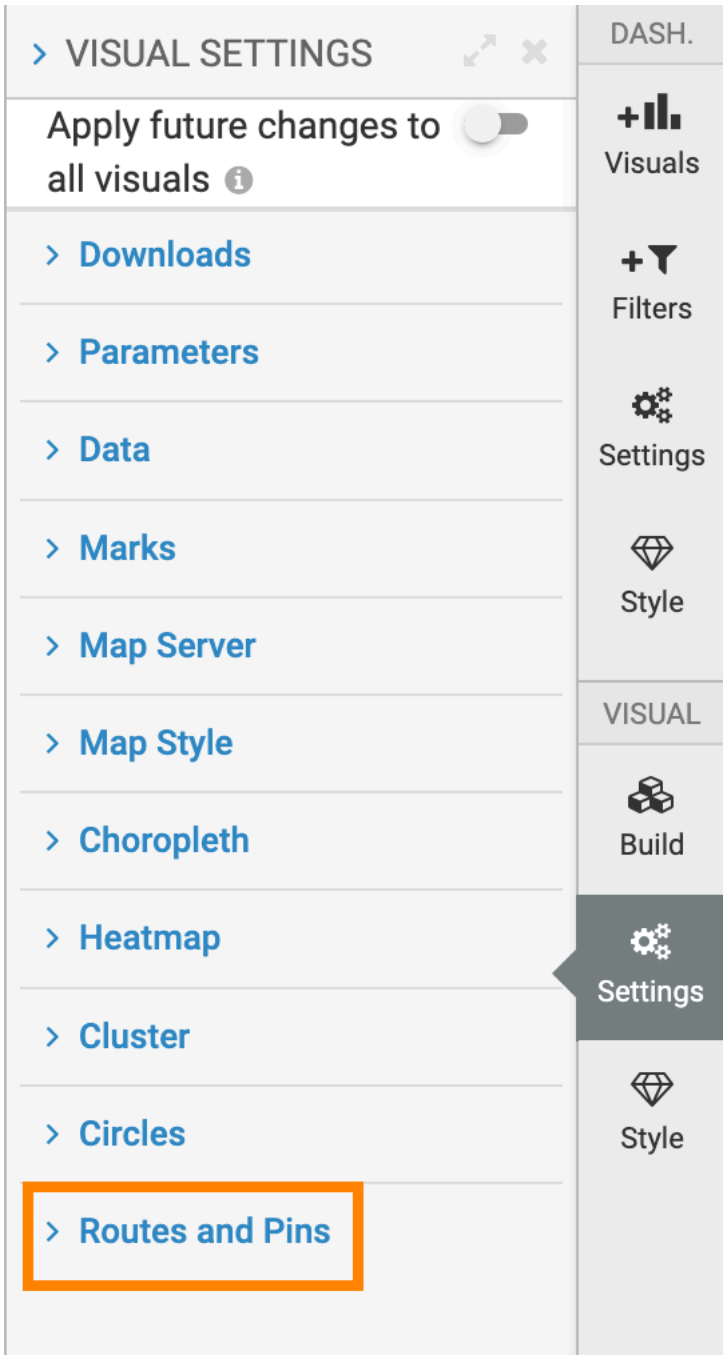

3. To change the middle pin type, you can select one of the following pin types from the Middle Pin Type dropdown menu: Default, Circle, None, and Custom.

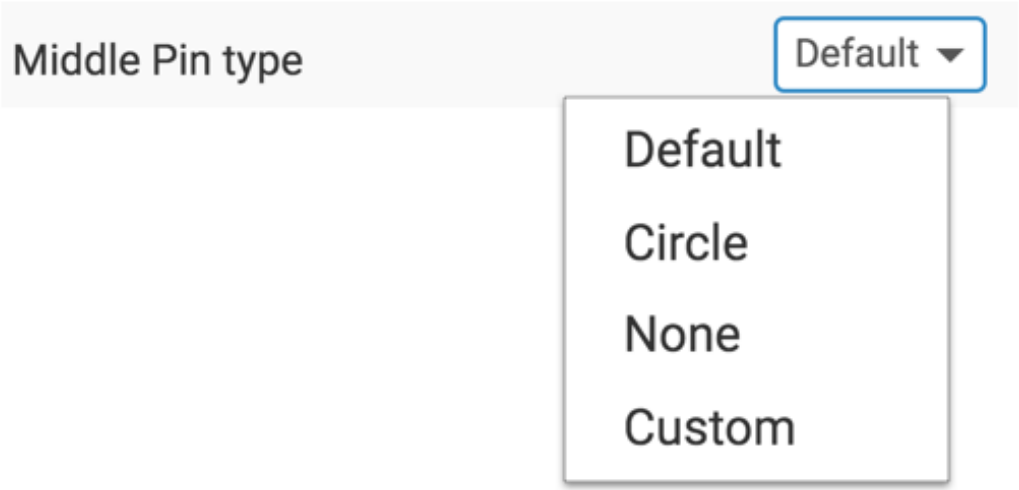

a) Select Circle from the dropdown menu to display the middle pins as circles.

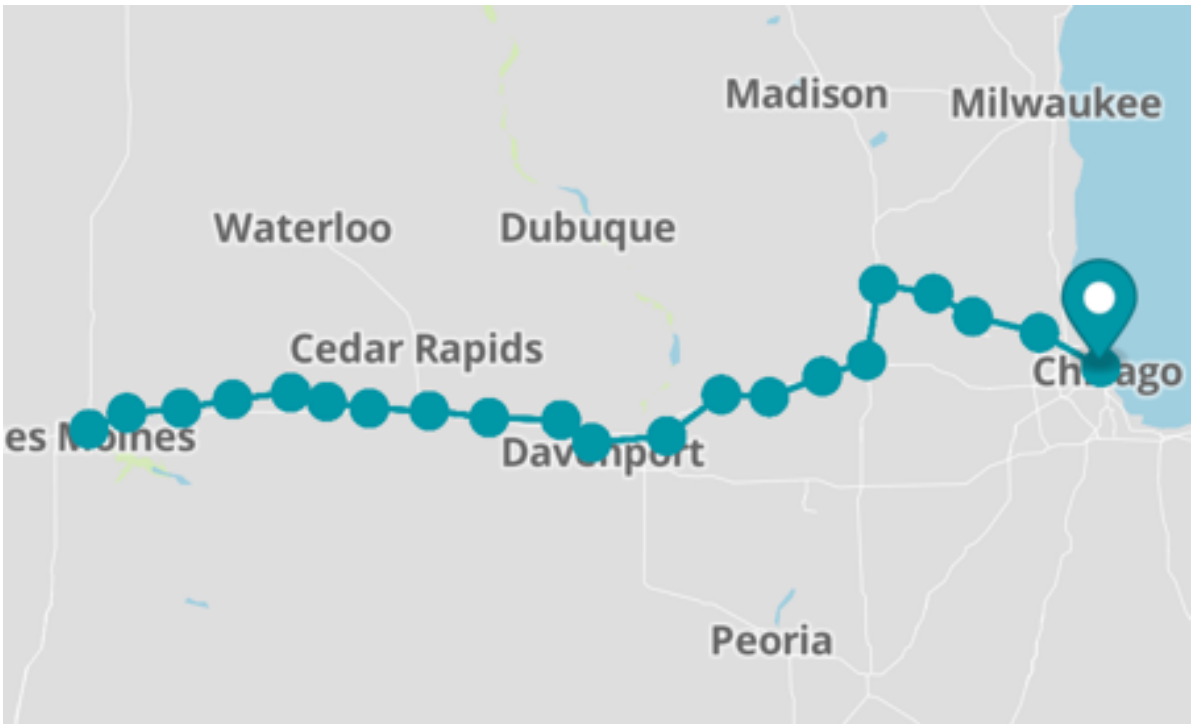

b) Select Custom from the dropdown menu to display the middle pin as trucks.

To add a custom icon, see *Specifying a Custom Icon*.

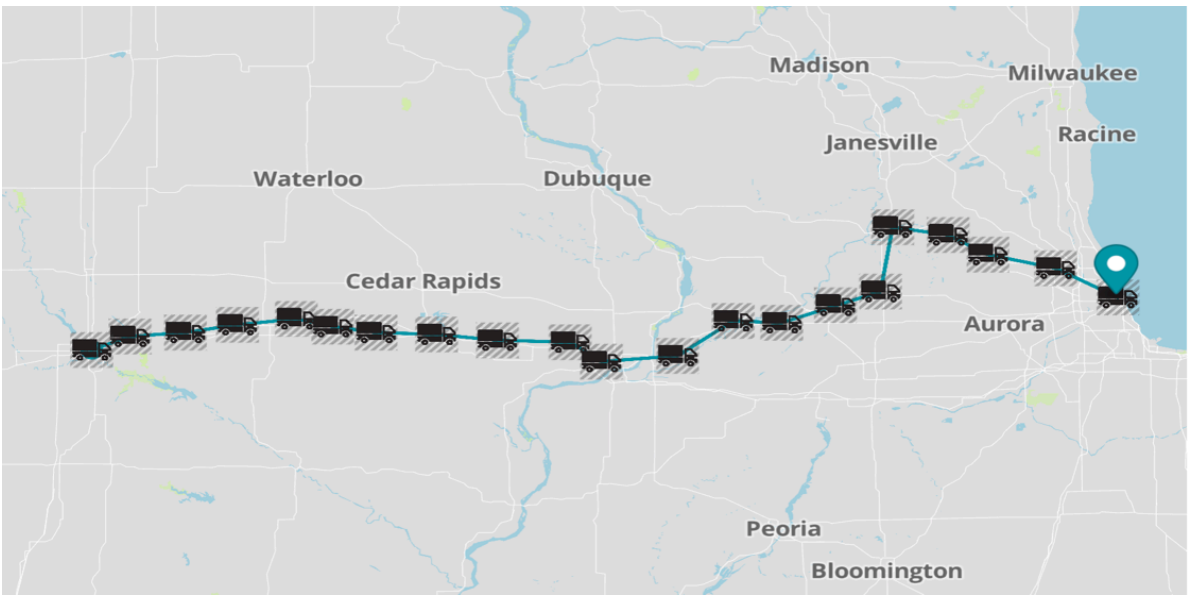

c) Select None from the dropdown menu to remove all middle pins.

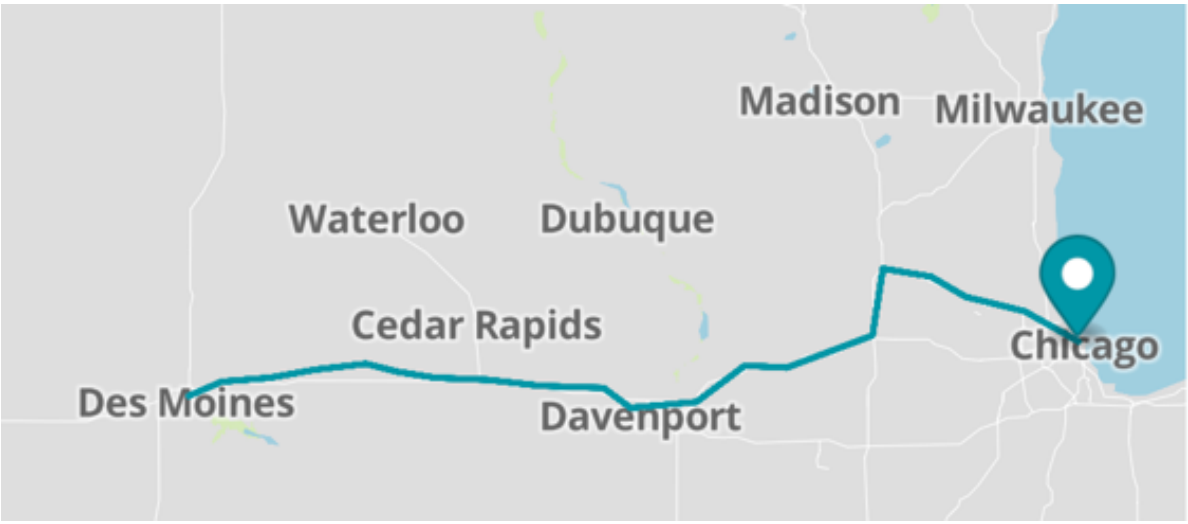

Related Information

Specifying a custom icon

Selecting end pin type

In a map route, you can choose an existing end pin type from a dropdown or add your own custom pin.

Before you begin

In an Interactive Map visual, before you select the End Pin Type option, make sure you enable routes. Once enabled, the route shows a default route and a default pin at the end of the route.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Routes/Pins.

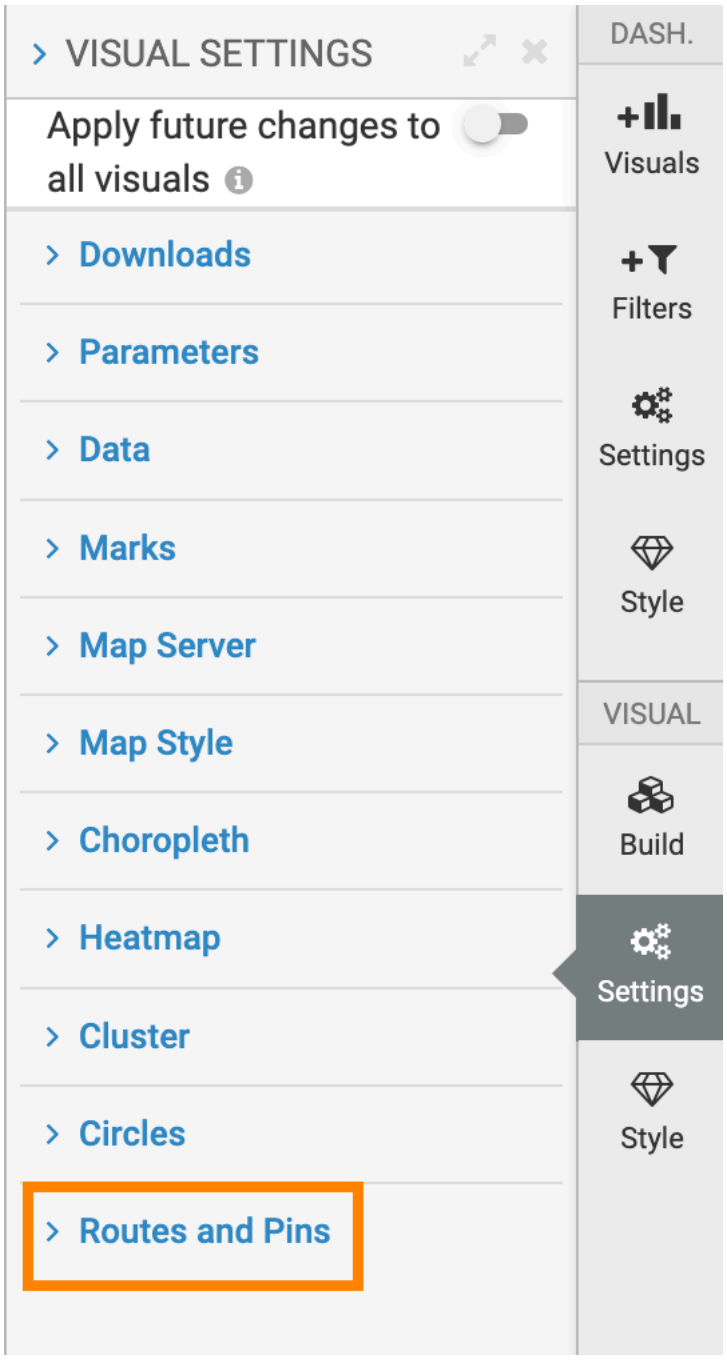

3. To change the end pin type, you can select one of the following pin types from the End Pin Type dropdown menu: Default, Circle, None, and Custom.

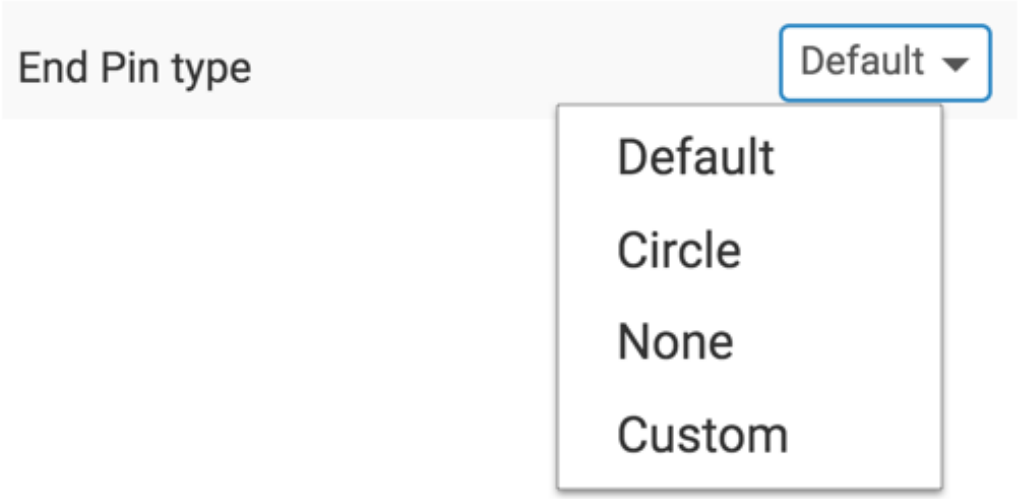

a) Select Circle from the dropdown menu to display the end pin as a circle.

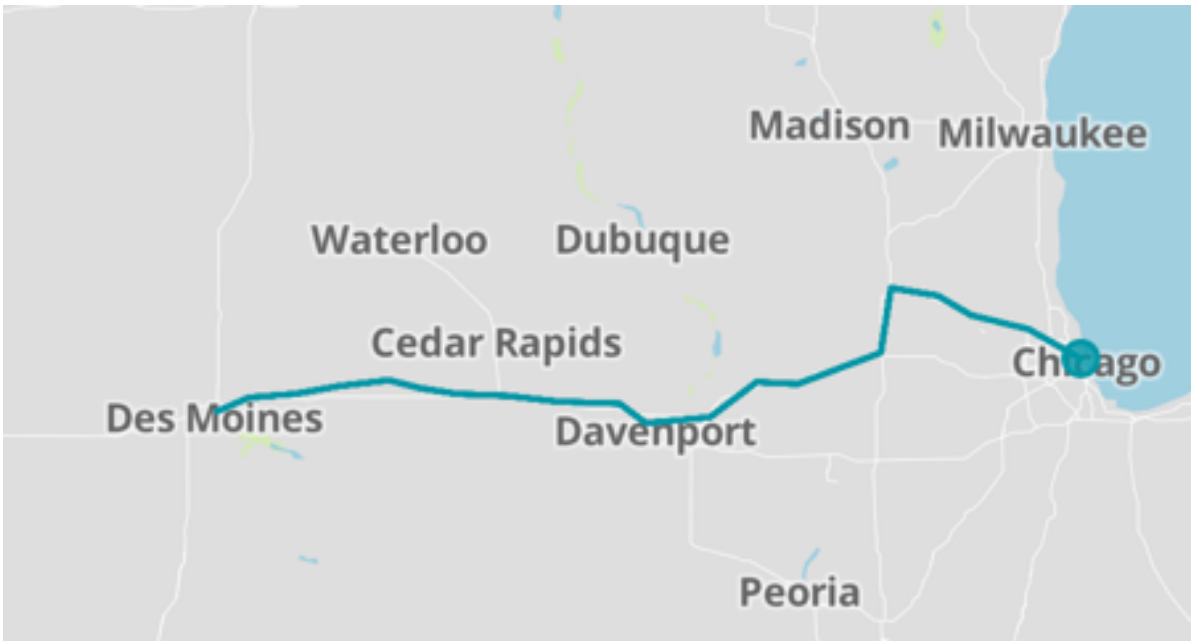

b) Select Custom from the dropdown menu to display the end pin as a truck.

To add a custom icon, see *Specifying a Custom Icon*.

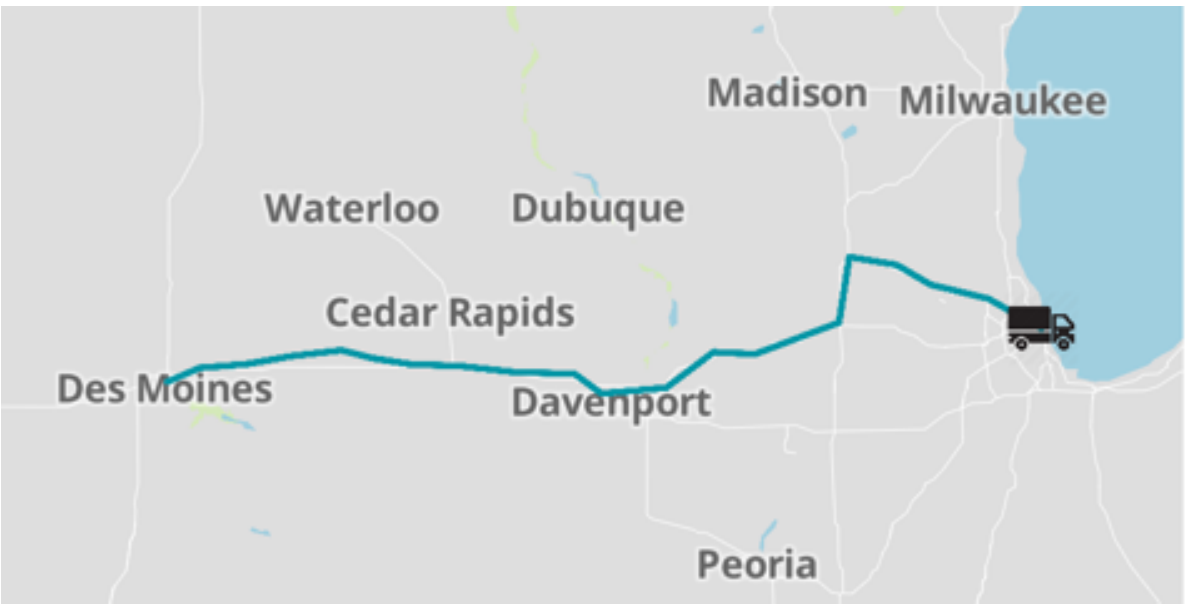

c) Select None from the dropdown menu to remove the end pin.

Related Information

Specifying a custom icon

Specifying a custom icon

In an interactive map visual, you can add a custom icon as a pin.

About this task

When using Mapbox, you can use any file type that can be used as an image in CSS, as a pin on your map.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Routes/Pins.

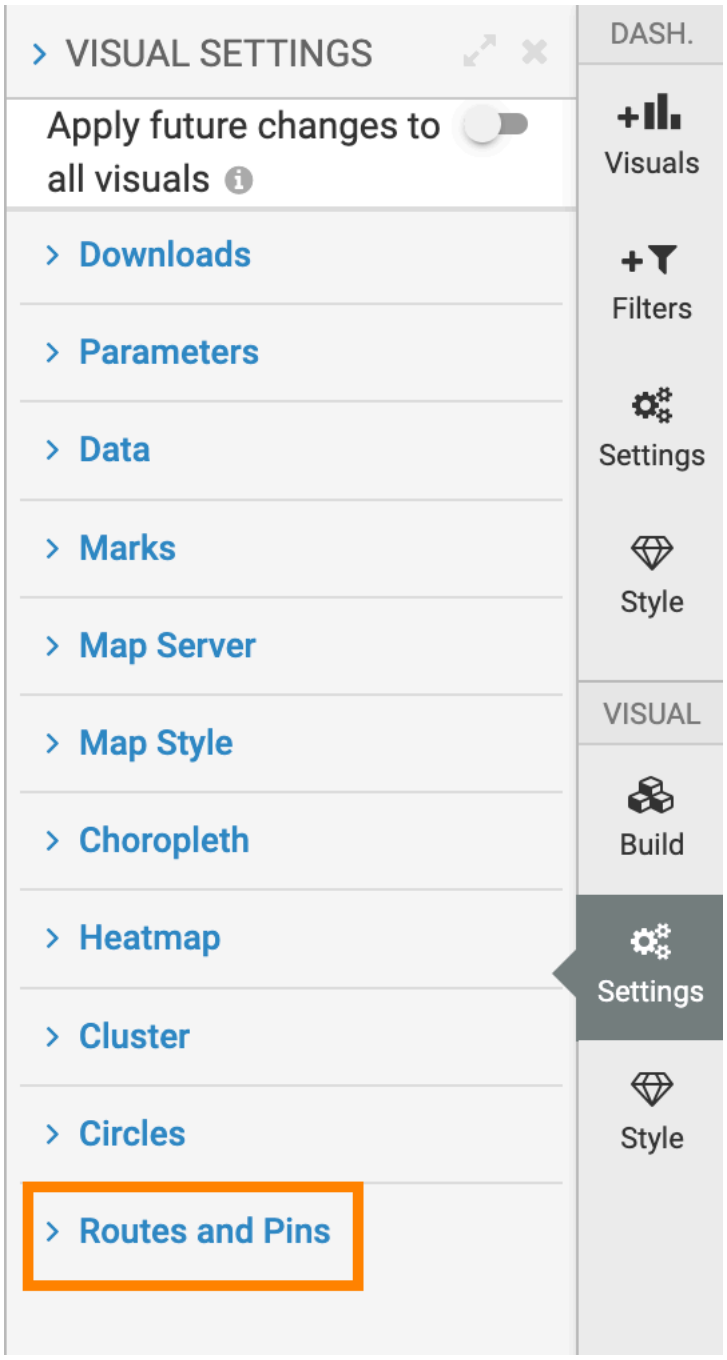

3. To specify a custom icon for the pin, enter the link in the Custom Icon text box.

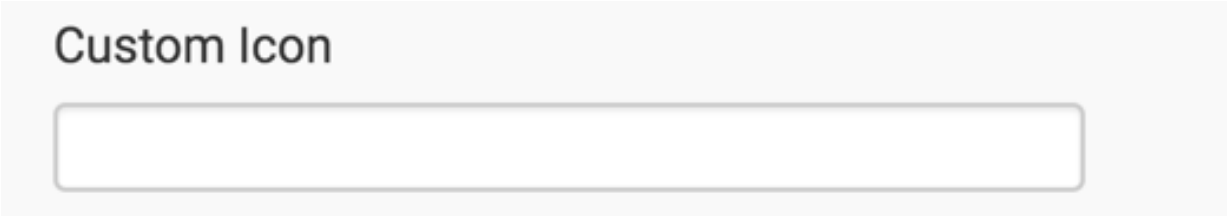

Example

Follow these steps to display pins as trucks on the entire route from start to end:

1. Select Custom in Start Pin Type, Middle Pin Type, and End Pin Type options.
2. Enter the following link in the Custom Icon option: https://cdn4.iconfinder.com/data/icons/eldorado-transport/40/truck_1-512.png.

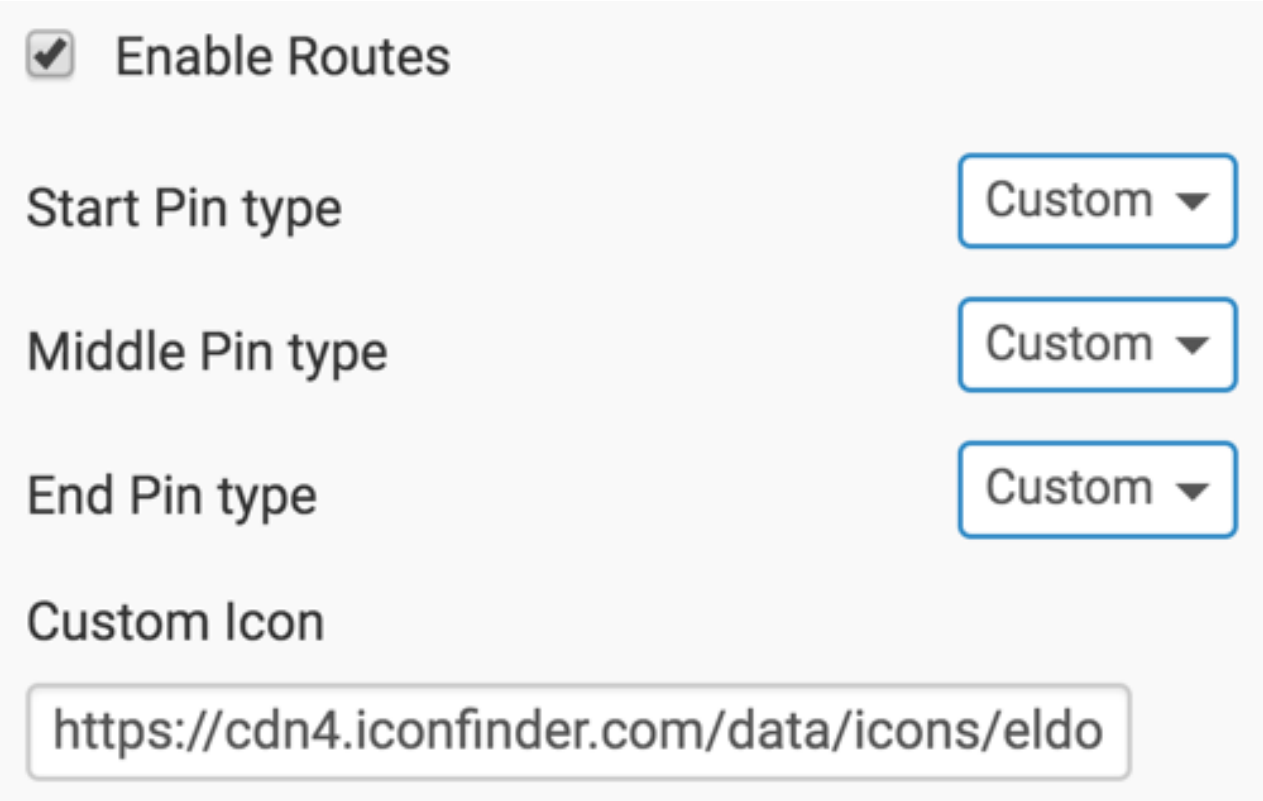

You can see that the image shows trucks on the entire route from start to end.

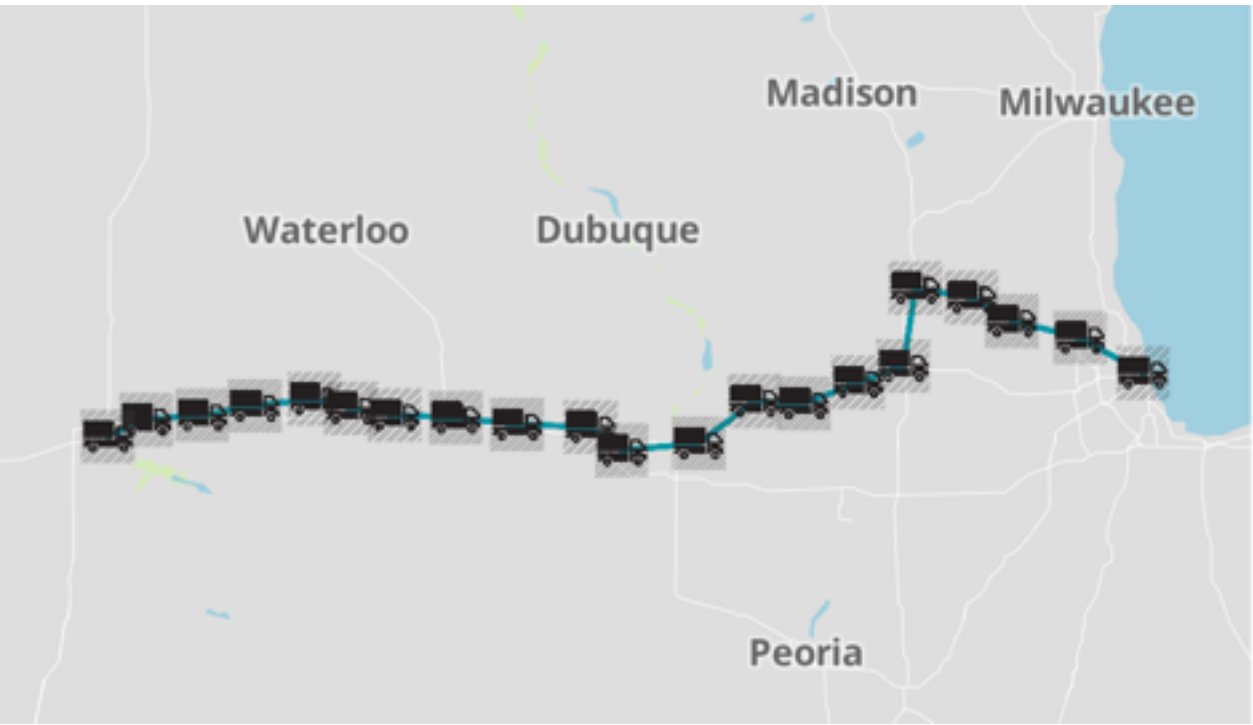

Changing custom icon size

In an interactive map visual, you can change the size of the custom icon.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Routes/Pins.

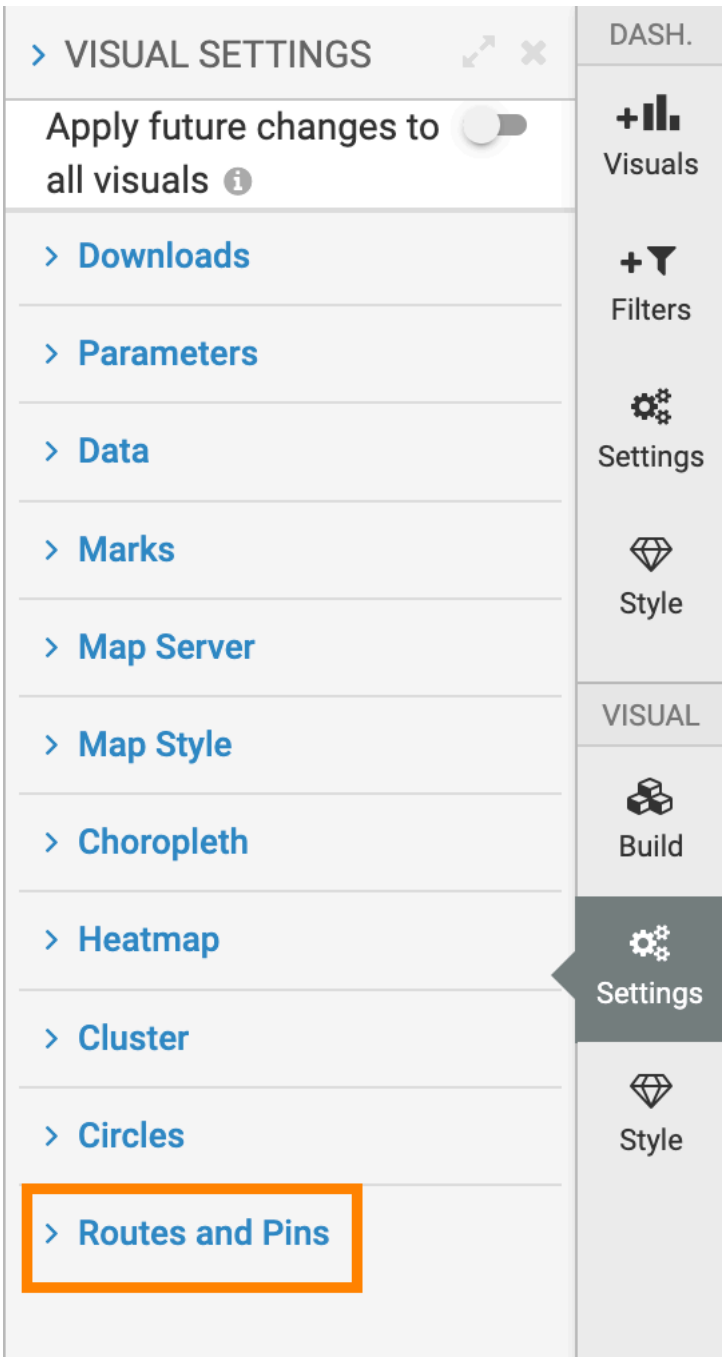

3. To adjust the size of a custom icon, adjust the minimum and maximum values in the Custom Icon Size option. Valid values are 0 through 100.

Example

In this example, a custom icon (truck) is used, and the icon size is set to 4.

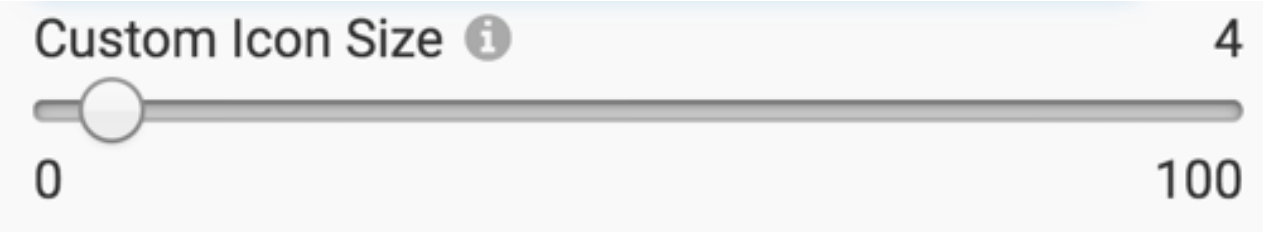

In the following image, you can see the size of the trucks when the custom icon size is 4.

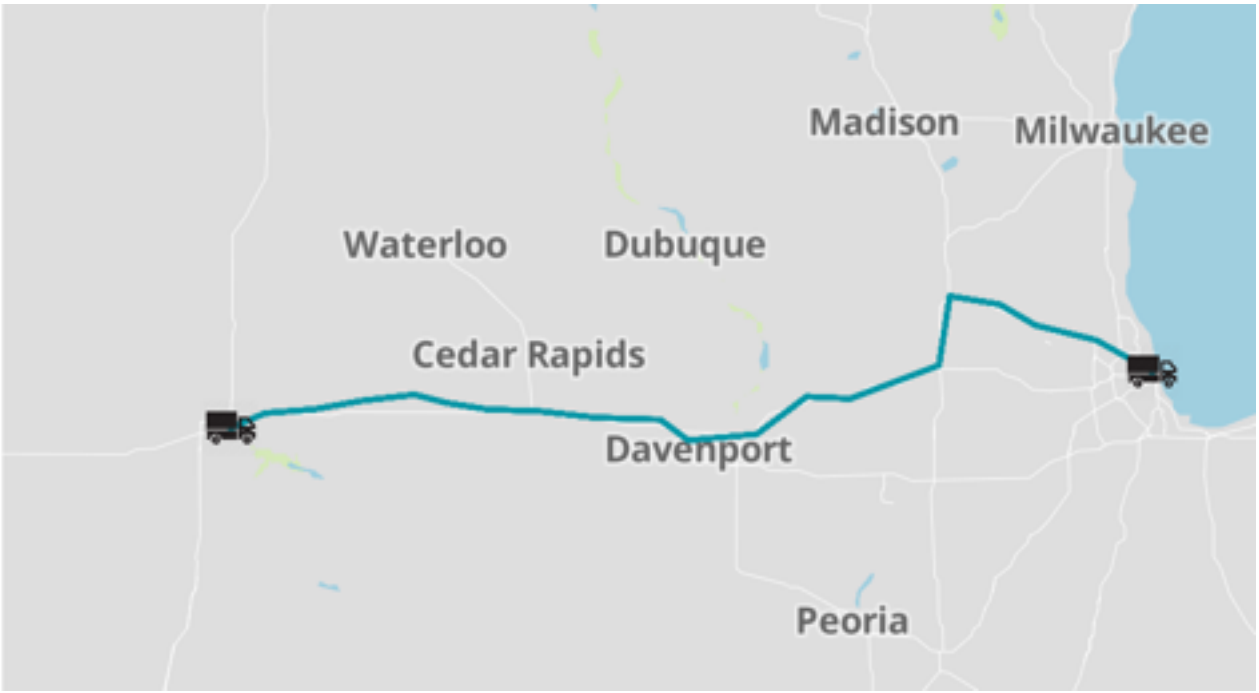

You can increase the custom icon size to 16.

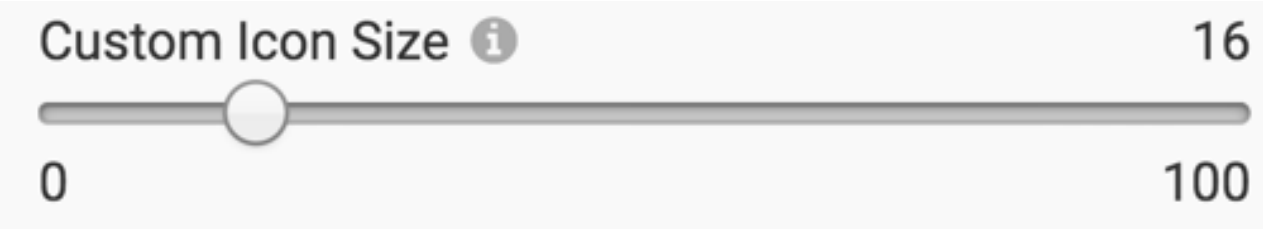

In the following image, you can see the bigget custom icon (truck) after the custom icon size is increased to 16.

Related Information

Specifying a custom icon

Customizing tooltip

The tooltip setting allows you to control display and formatting for the tooltip of a visual.

Showing percentage

Procedure

1. On the right side of Visual Designer, click the Settings menu.

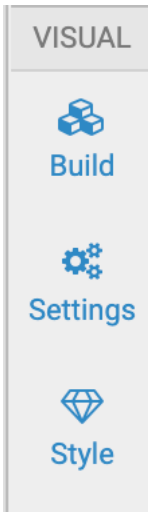

2. In the Settings menu, click Tooltip.

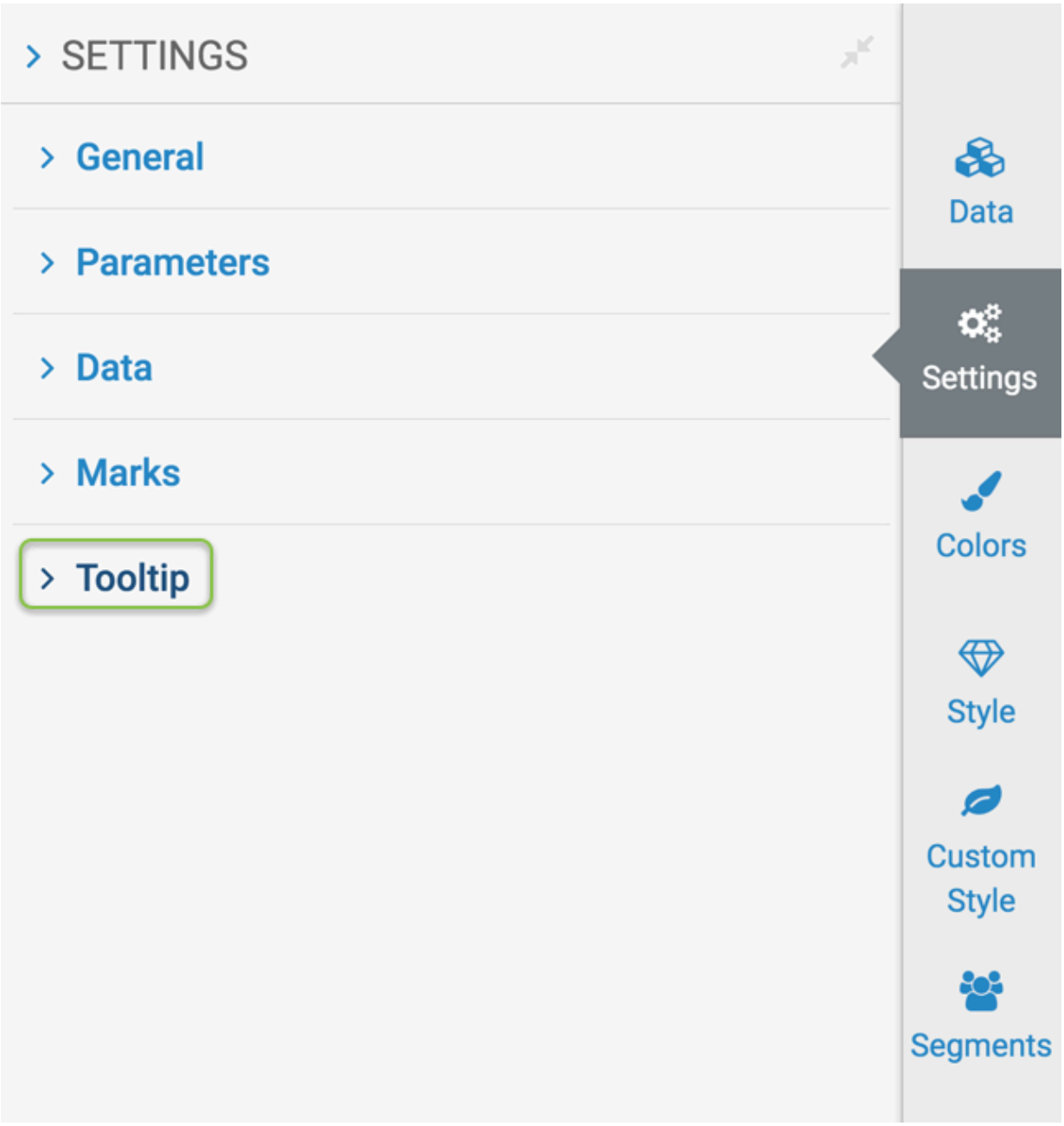

3. To show a percentage in a tooltip of a pie visual, select the Include percentage of total in tooltip option.

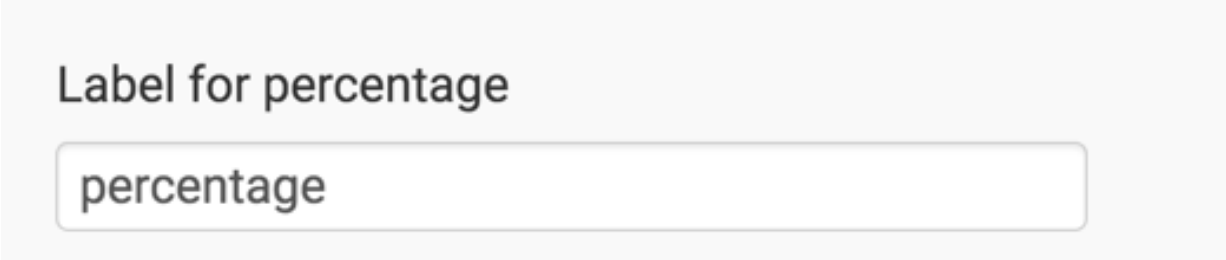

Specifying label for percentage

Procedure

1. On the right side of Visual Designer, click the Settings menu.

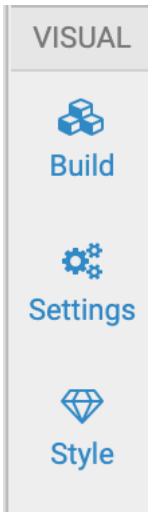

2. In the Settings menu, click Tooltip.

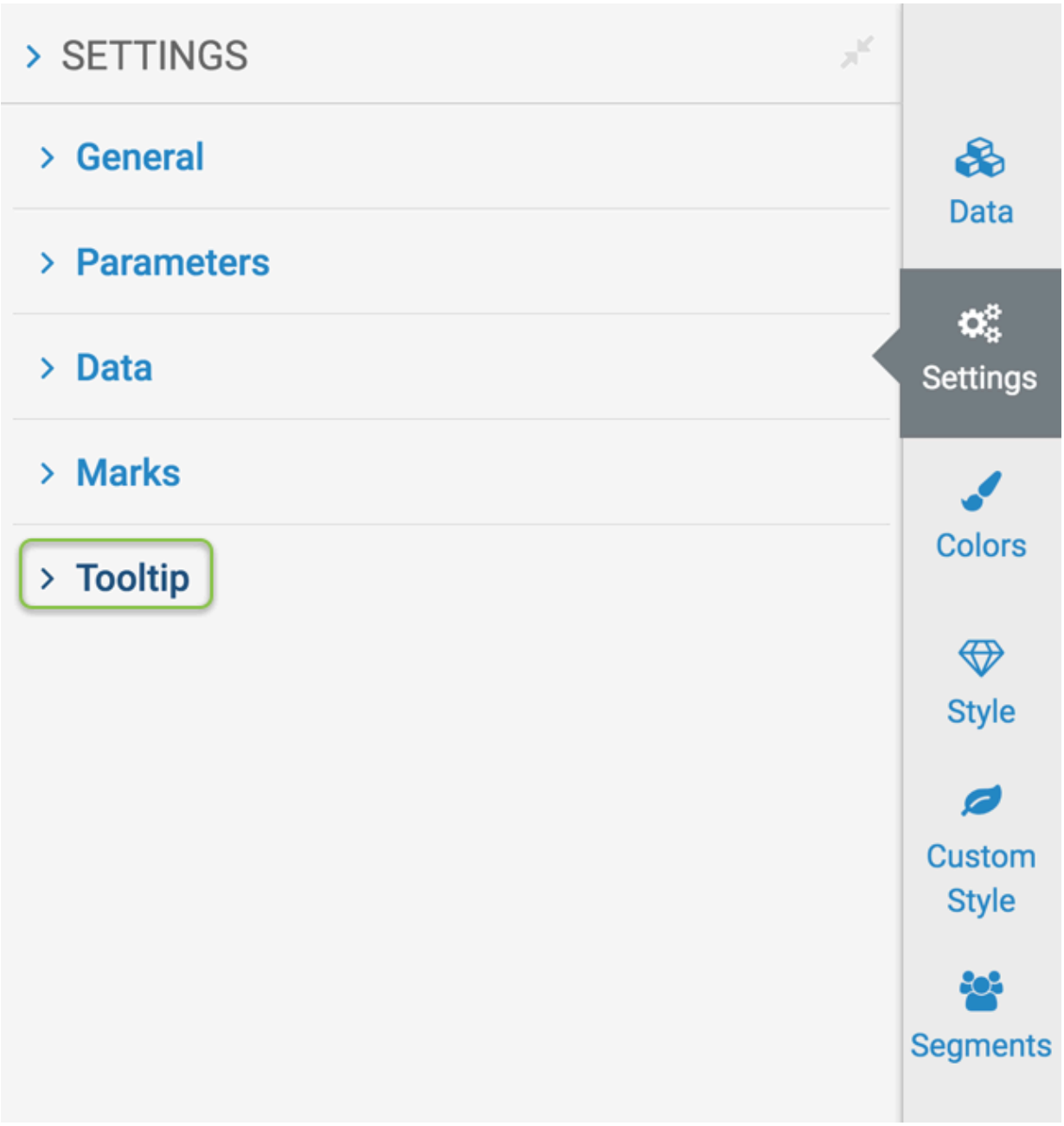

3. To specify a label for the percentage measurement in a tooltip of a visual, change the value in the Label for percentage option. The default value is percentage.

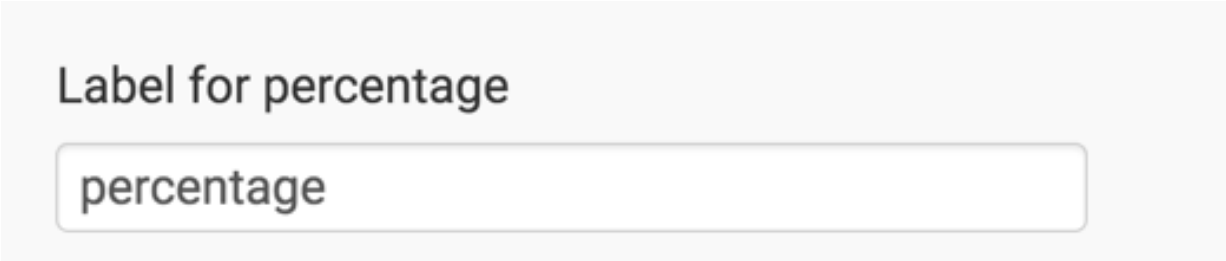

Changing number of decimals for percentage

Procedure

1. On the right side of Visual Designer, click the Settings menu.

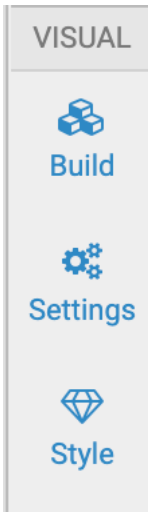

2. In the Settings menu, click Tooltip.

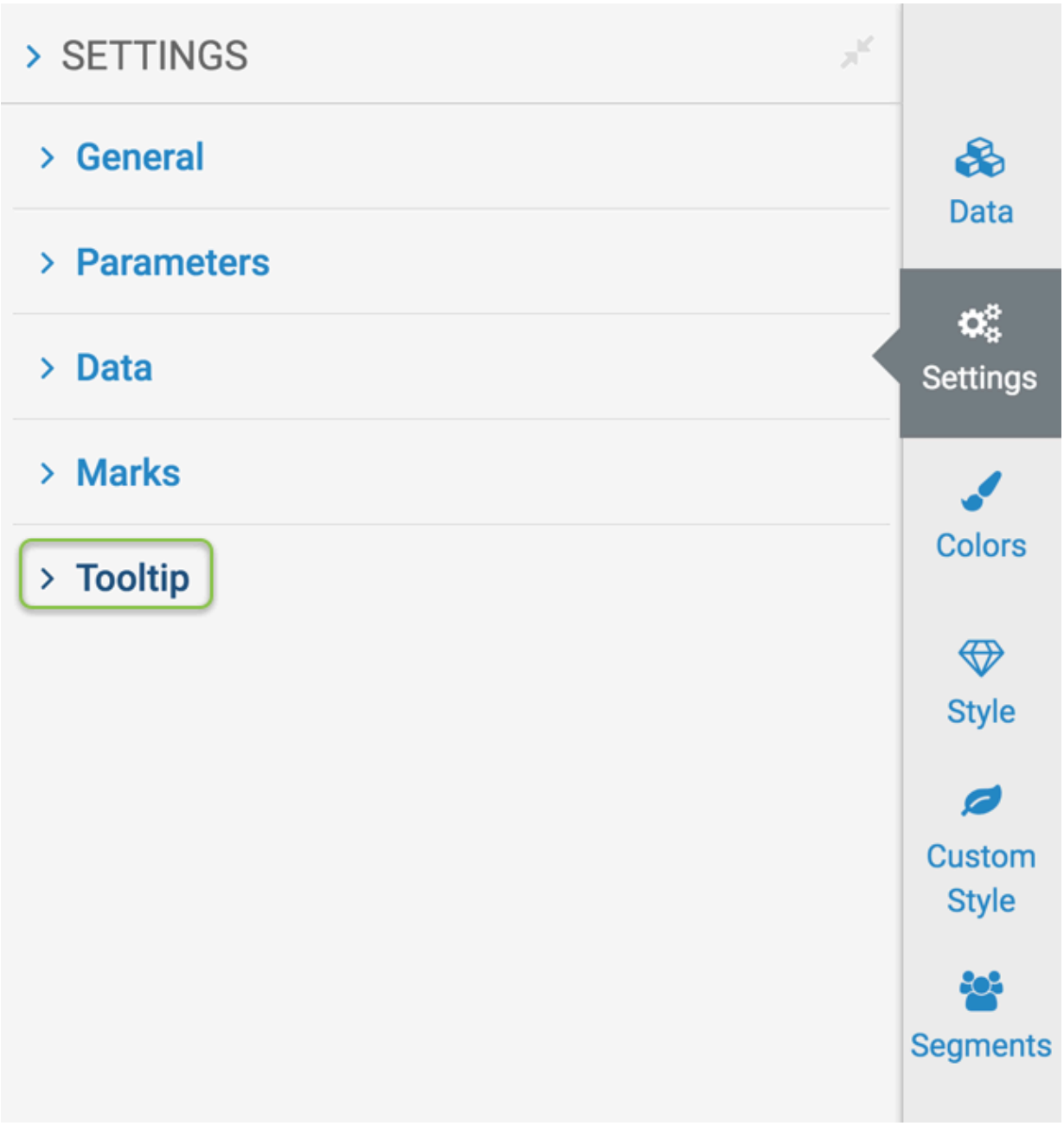

3. To change the number of decimal places in the percentage measurement for a tooltip of a visual, adjust the selector for the Decimals included in percentage option. The default value is 2.

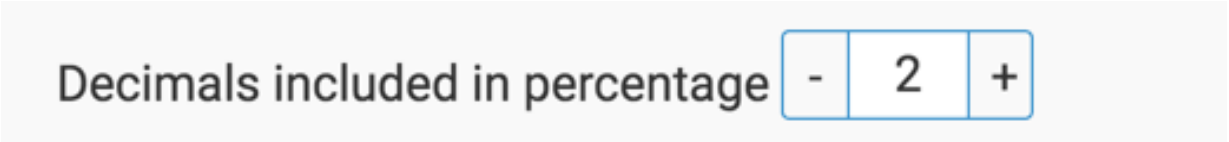

Changing tooltip font size

Procedure

1. On the right side of Visual Designer, click the Settings menu.

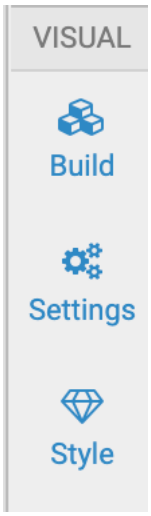

2. In the Settings menu, click Tooltip.

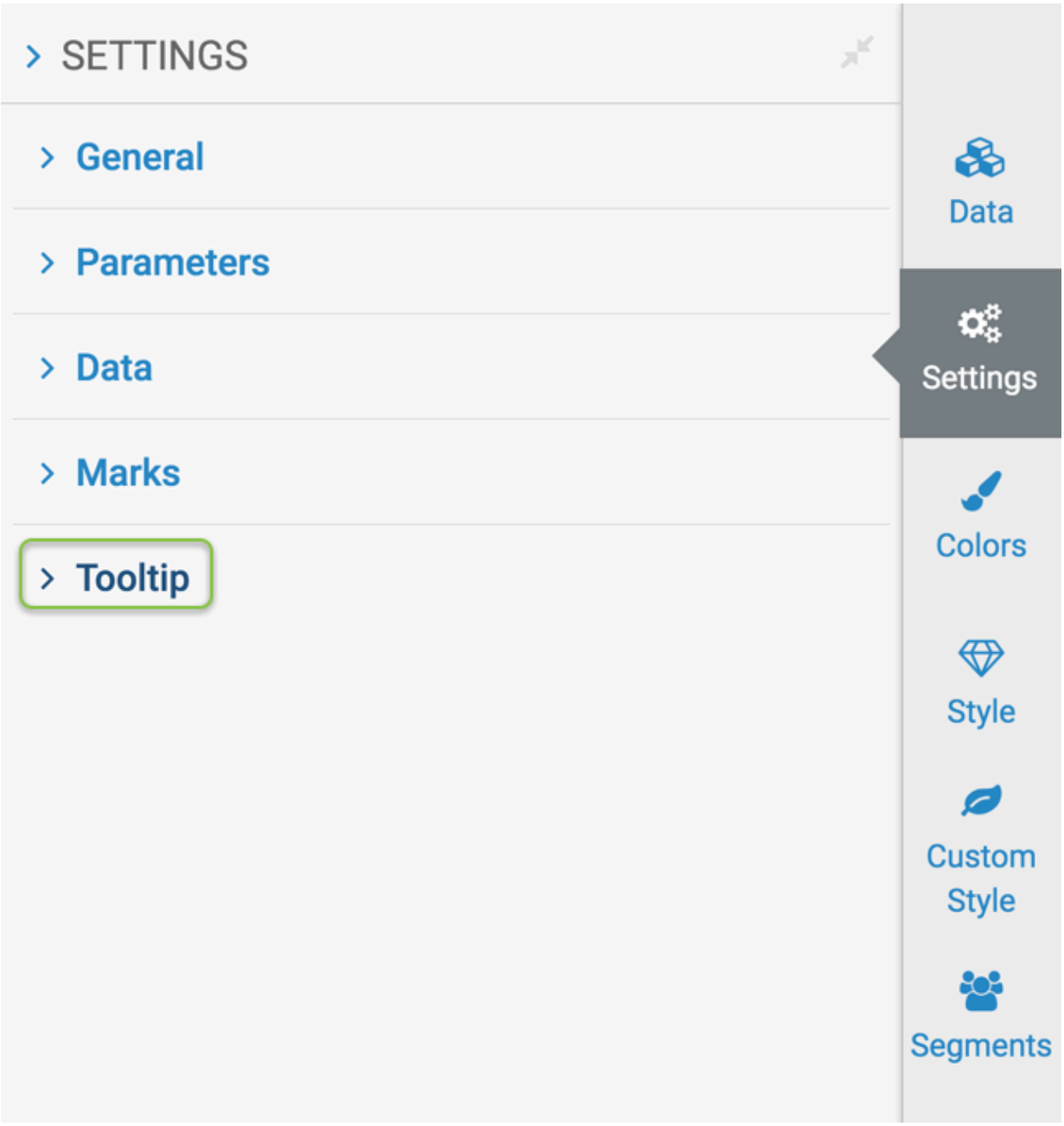

3. To change the font size used by the tooltip in a visual, make adjustments in the Font Size selector.

Changing tooltip font family

Procedure

1. On the right side of Visual Designer, click the Settings menu.

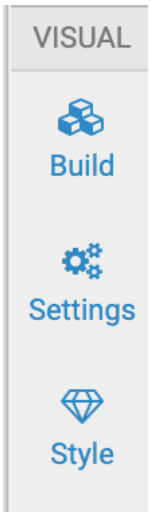

2. In the Settings menu, click Tooltip.

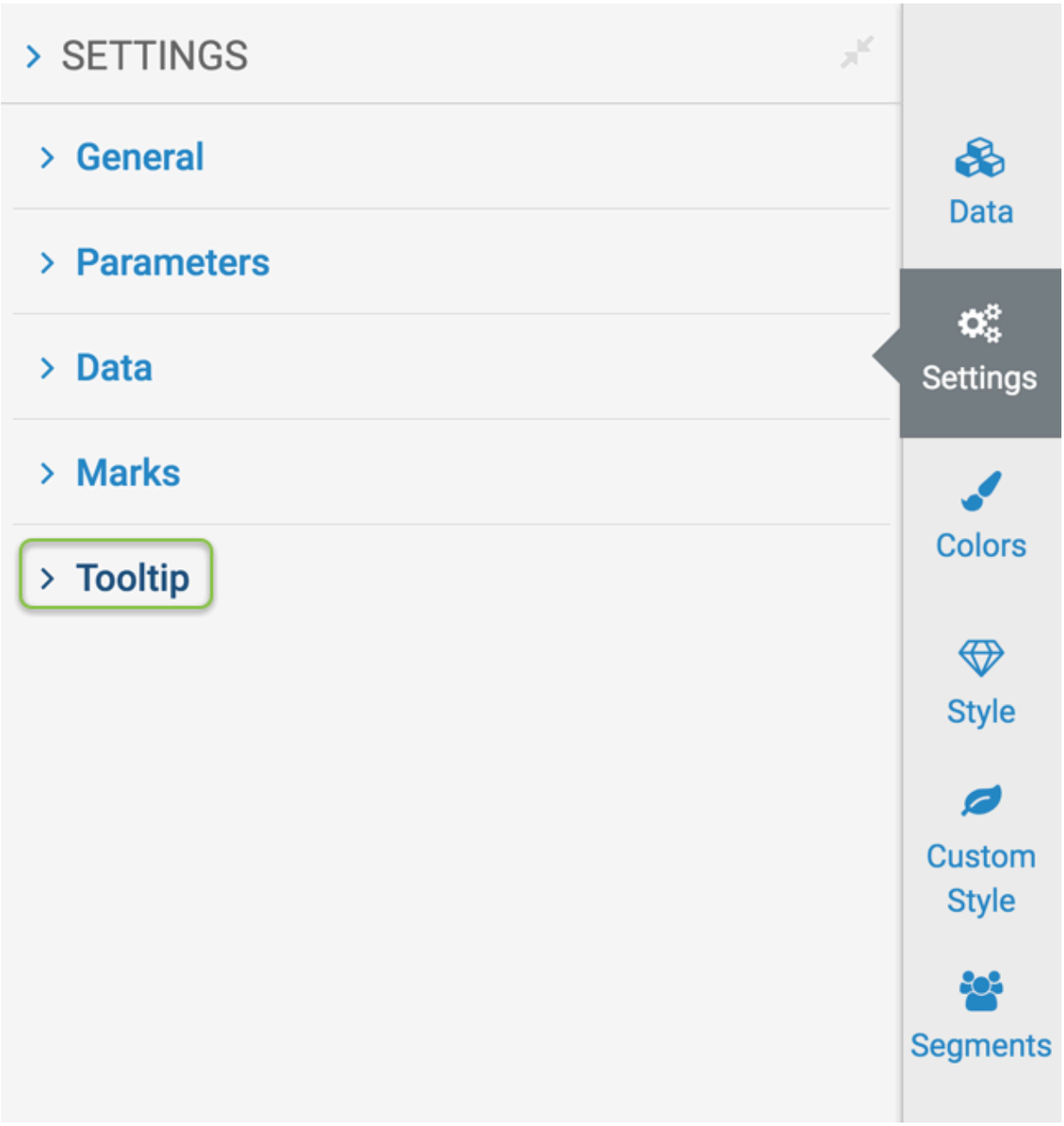

3. To change the tooltip font inside a visual, open the Font Family menu and make a selection there.

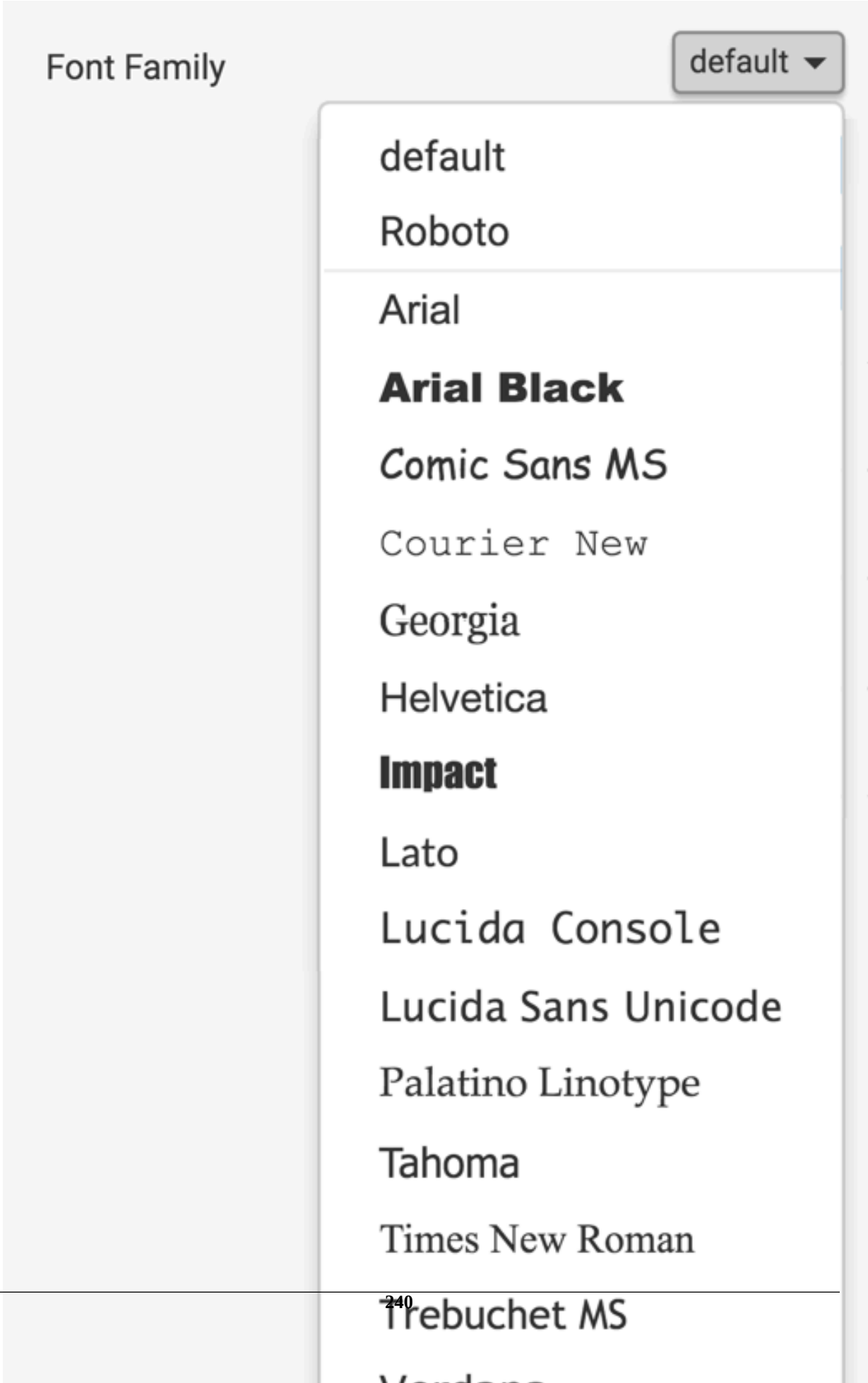
Font Family
default
default
Roboto
Arial
Arial Black
Comic Sans MS
Courier New
Georgia
Helvetica
Impact
Lato
Lucida Console
Lucida Sans Unicode
Palatino Linotype
Tahoma
Times New Roman
Trebuchet MS

Customizing trellis

You can manage several display options from the Trellis menu, depending on the chart type in the visual.

Displaying trellis borders

About this task

This setting is available on most visual types.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Trellis.

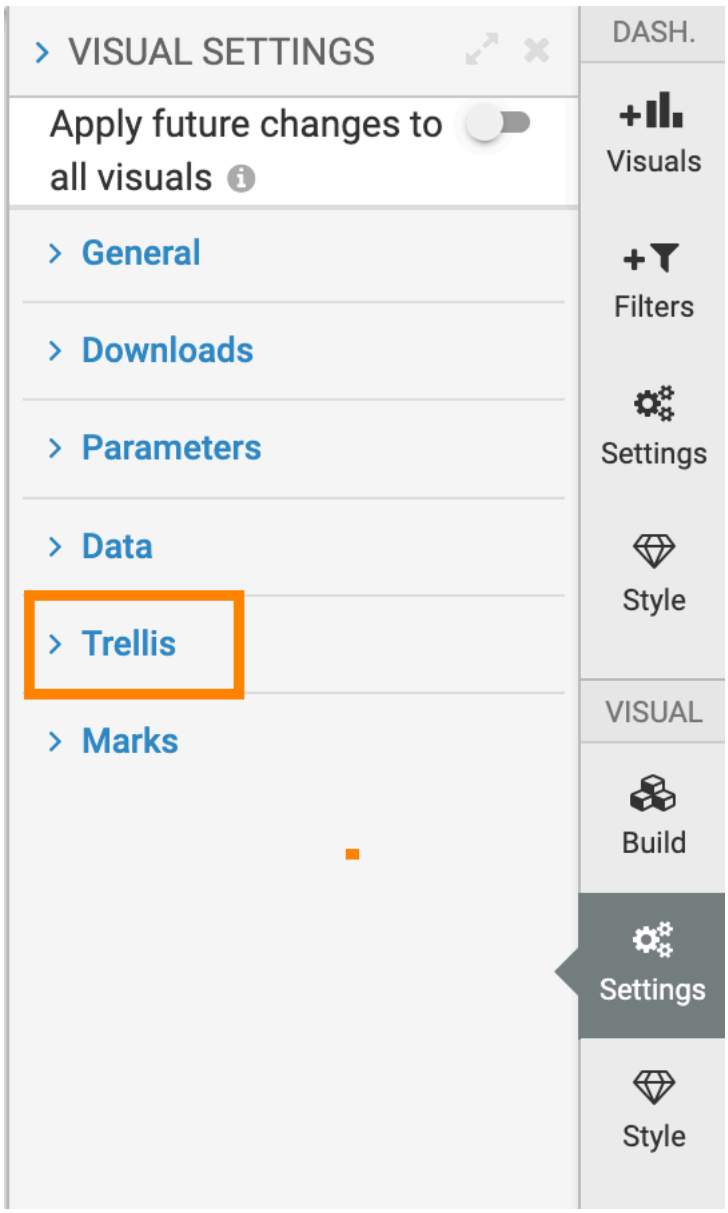

3. To turn the trellis borders on or off, select the Display trellis borders option.

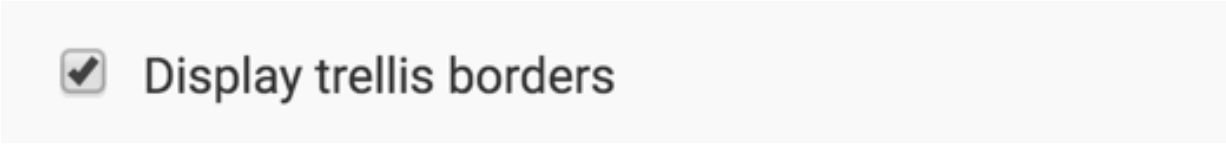

Enabling independent axis scale

About this task

Cloudera Data Visualization lets you apply independent axis to trellised visuals. Trellis charts are very powerful visualization tools. However, they do not work well when one trellis partition contains values that are several orders of magnitude larger than in other trellis partitions. The charts that represent segments with smaller values are very difficult to interpret. You can mitigate this problem by enabling independent axis.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Trellis.

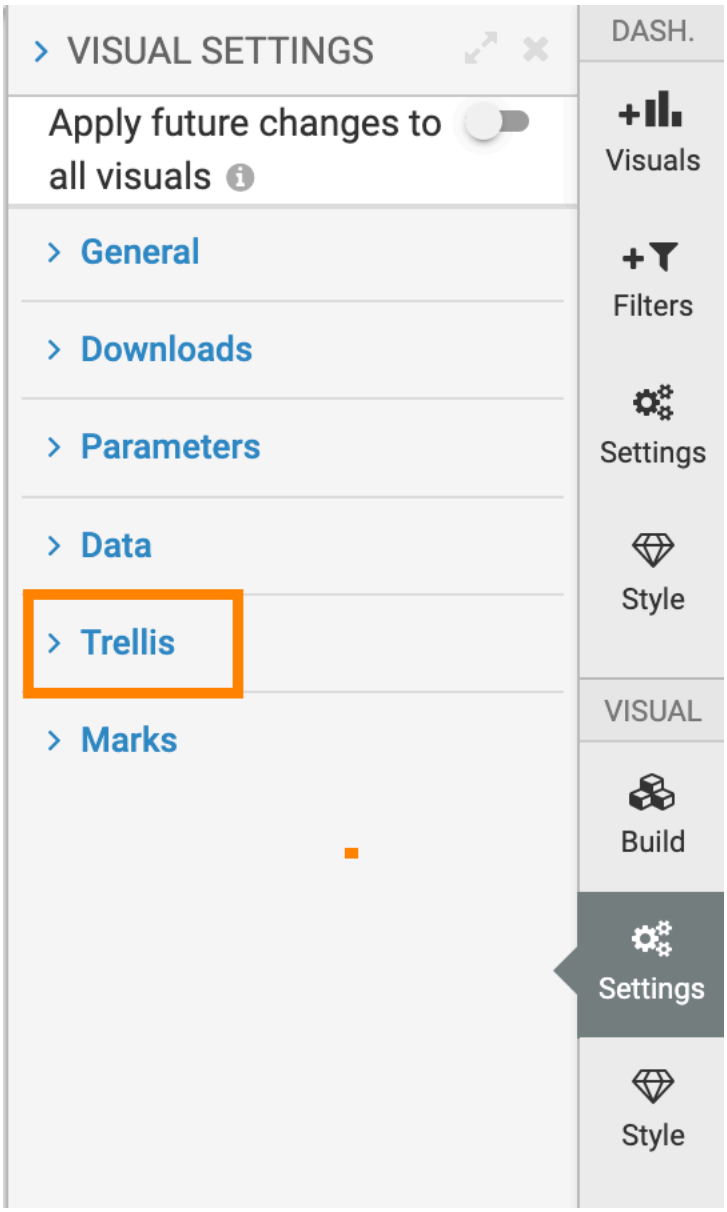

3. To enable independent axis scale, select the relevant option.

   Depending on the type of visual, this option may appear as one of:

   Independent Dimension Scale

   For Bar, Lines, Areas, Grouped Bars, and Box Plot visuals, to make dimension scale independent, navigate to the Trellis menu, and select Independent Dimension Scale.

   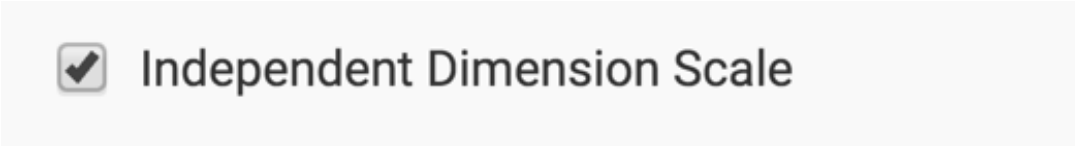

   Independent Measure Scale

   For Box Plot visual, to make the measure scale independent, navigate to the Trellis menu, and select Independent Measure Scale.

   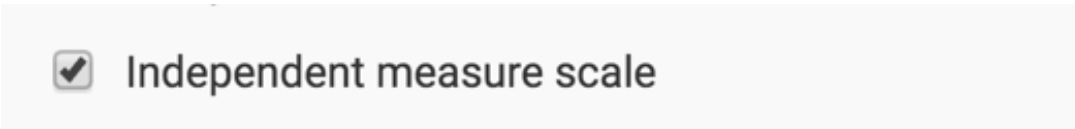

   Independent Aggregate Scale

   For Bar, Lines, Areas, and Grouped Bars visuals, to make the aggregate scale independent, navigate to the Trellis menu, and select Independent Aggregate Scale.

   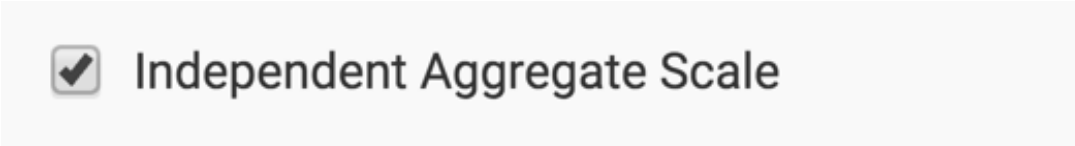

   Independent X Scale
   Independent Y Scale

   For Scatter and Histogram visuals, to make the X and/or Y scales independent, navigate to the Trellis menu, and select Independent X Scale and/or Independent Y Scale.

   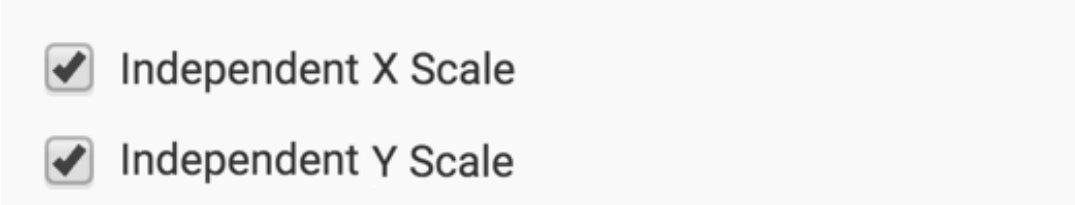

   Scale independently across trellis

   For Gauge and Bullet visuals, to scale independently across trellis, navigate to the Trellis menu, and select Scale independently across trellis.

   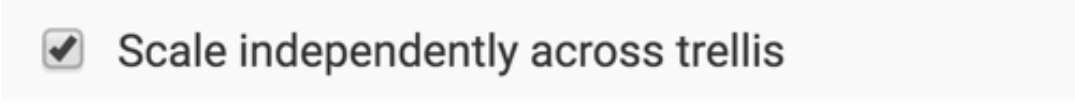

   Independent scales

   For Map visual, to scale independently across trellis, navigate to the Trellis menu, and select Independent scales.

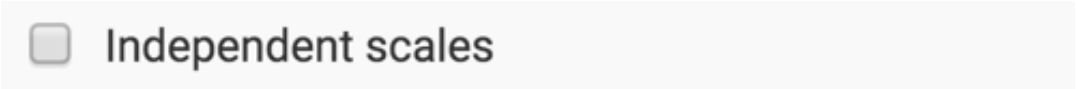

Independent date scale

For Calendar Heatmap visual, to make the date scale independent, navigate to the Trellis menu, and select Independent date scale.

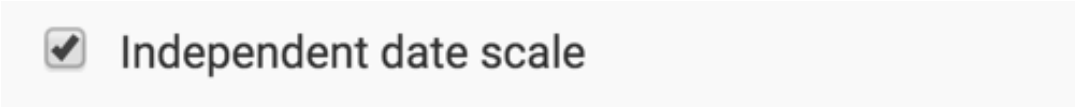

To learn how to make scales independent, see *Trellis option with independent scale*.

Related Information

Trellis option with independent scale

Setting minimum trellis cell width

About this task

This setting is available on most visual types.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Trellis.

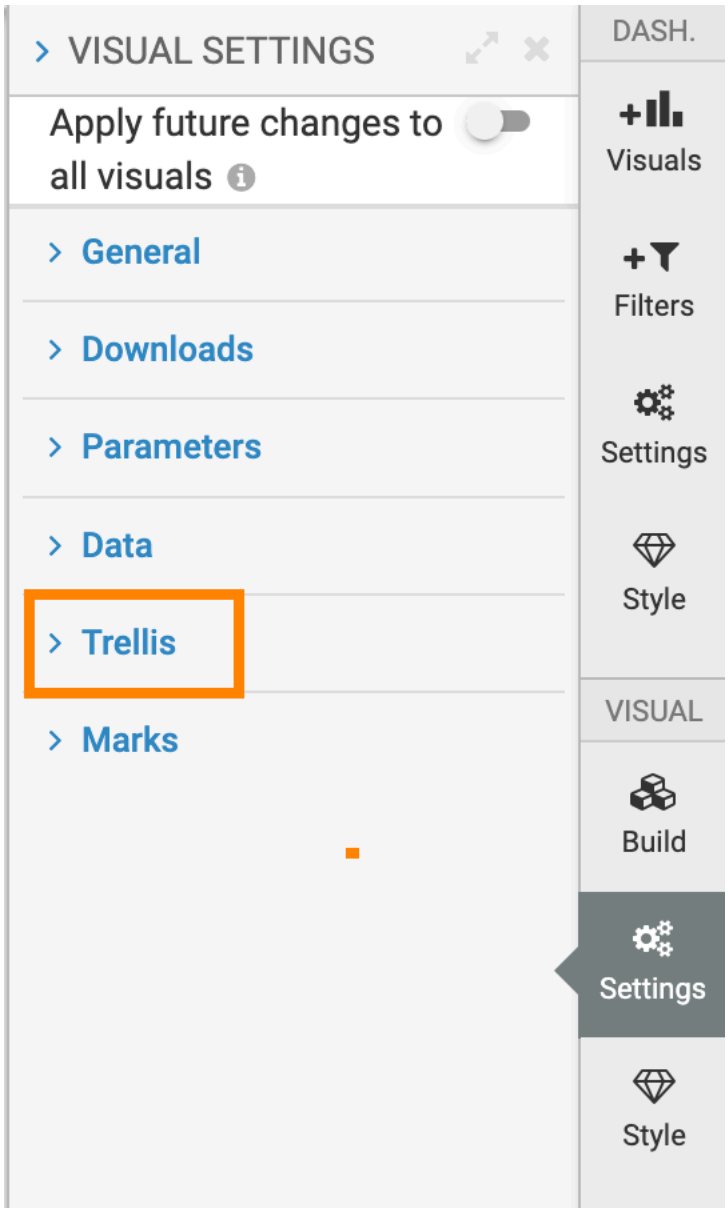

3. To change the horizontal size of the trellis cell in a visual, change the value of the Min Cell Width option.

   The default is 100 px, the range is from 50 px to 2,000 px.

Setting minimum trellis cell height

About this task

This setting is available on most visual types.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Trellis.

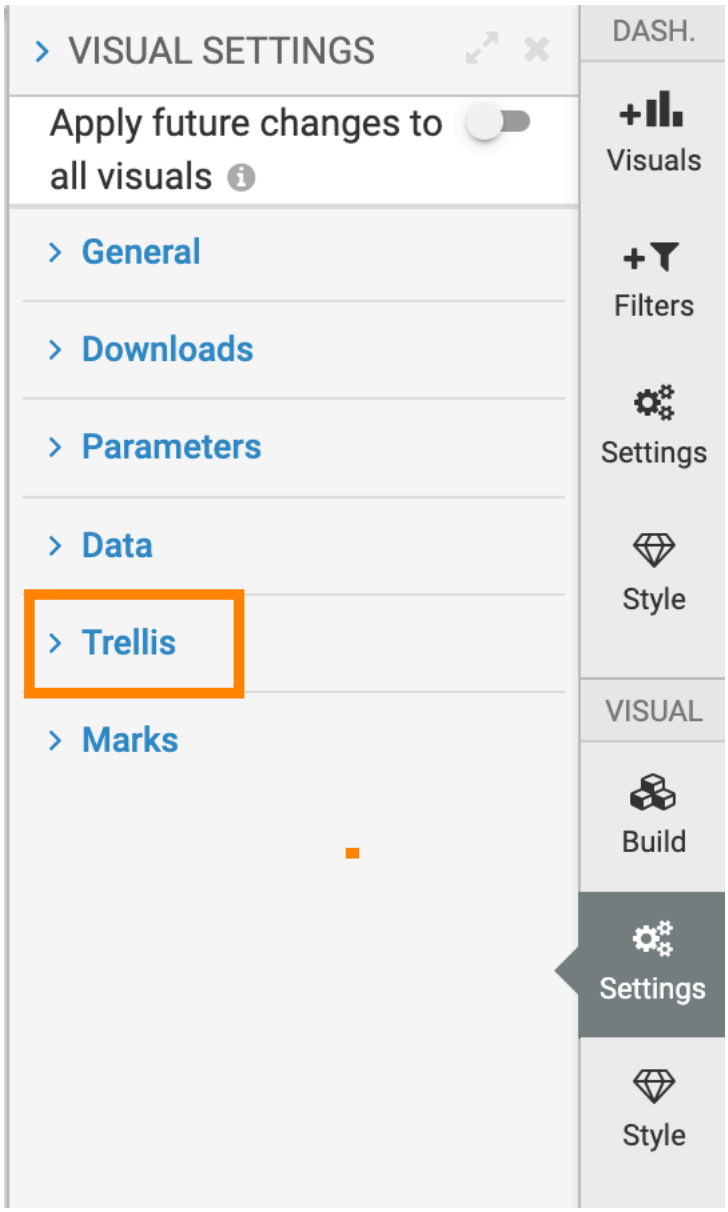

3. To change the vertical size of the trellis cell in a visual, change the value of the Min Cell Height option.

   The default is 100 px, the range is from 50 px to 2,000 px.

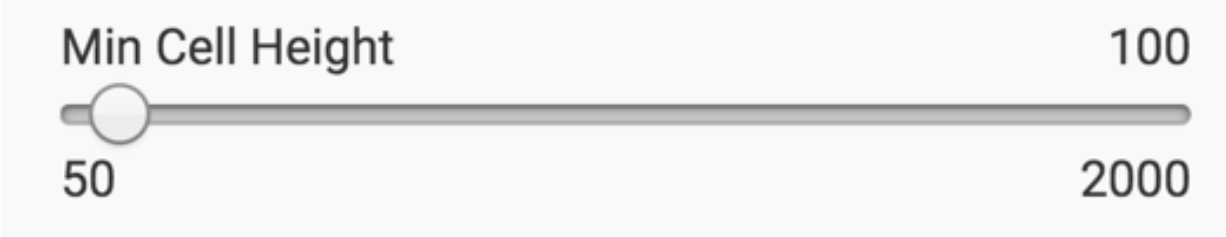

Customizing zoom

Cloudera Data Visualization enables you to customize zoom options for your map visual.

Enabling pan and zoom

To manipulate the appearance of the map, Cloudera Data Visualization enables you to pan and zoom your map visual.

Procedure

1. On the right side of Visual Designer, click Settings.
2. In the Settings menu, click Zoom.

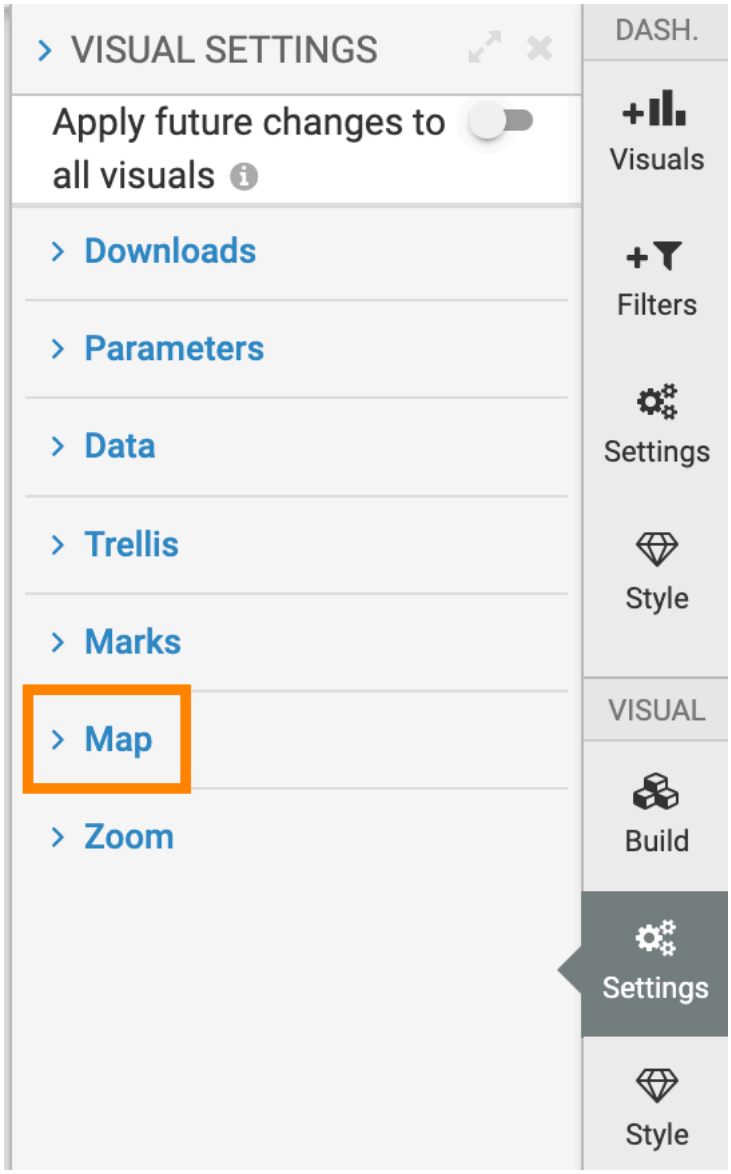

3. To enable pan and zoom on Maps, select the Enable pan and zoom option.

   If Enable pan and zoom is checked, you can manipulate the appearance of the map in the following ways:

   - Pan to move the visual, to change the focus to a specific part of the visual.

     Click and hold the primary mouse button (usually, the left button) on the map in the visual, and move the graphic to the desired location.

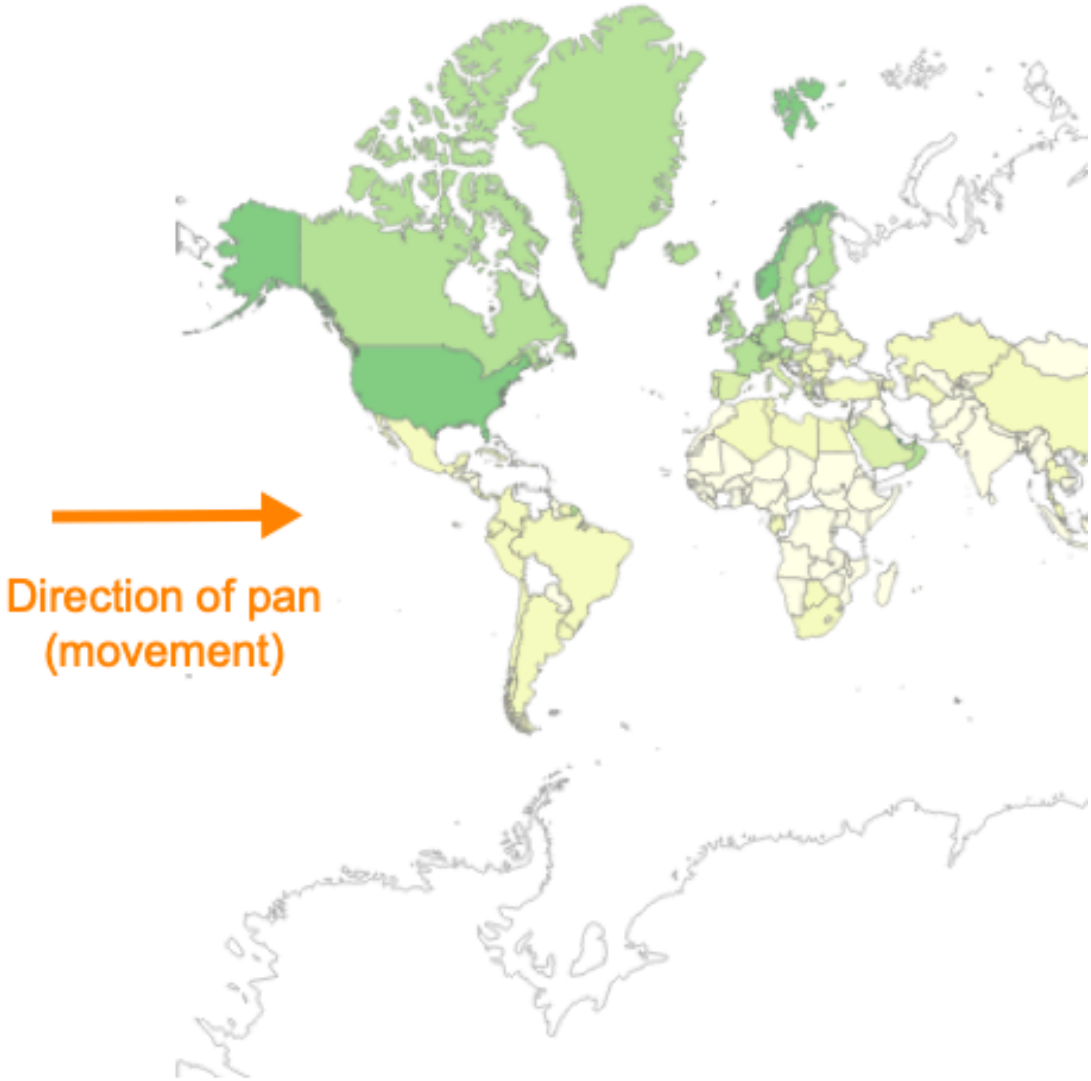

   - Zoom to change the scale of the visual. You must zoom in to make the visual larger and show more detail, or zoom out to make the visual smaller and show less detail.

     Click and hold the secondary mouse button (usually, the right button) on the map in the visual. Moving it down zooms in, while moving it up zooms out.

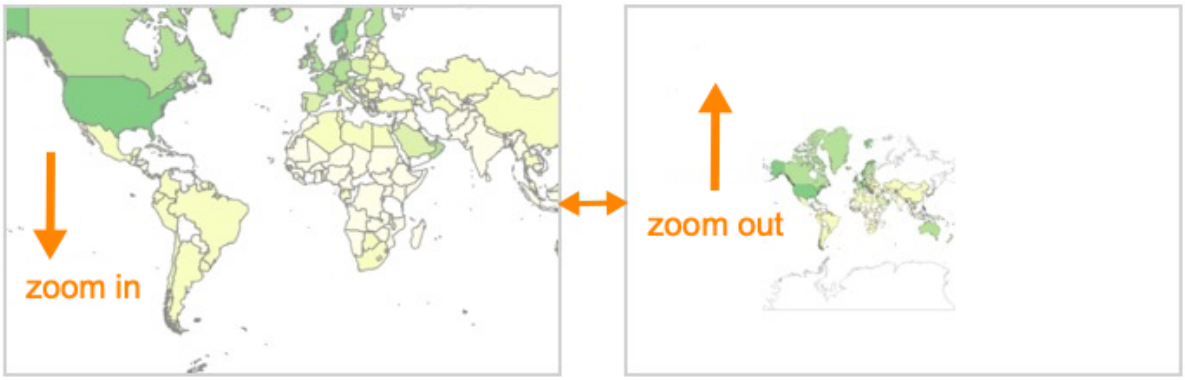

Enabling zoom to active areas

In Cloudera Data Visualization, Map visuals can enable you to zoom to active areas in.

Procedure

1. On the right side of Visual Designer, click Settings.

2. In the Settings menu, click Zoom.

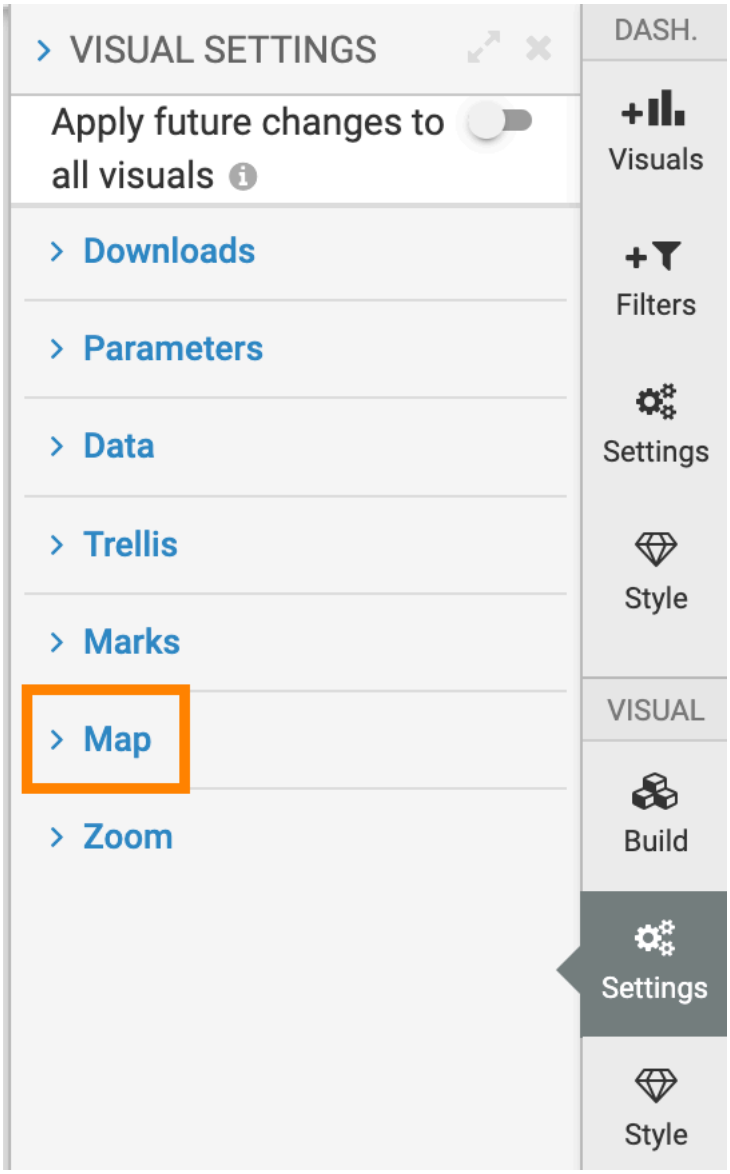

3. To enable active area zoom on Maps, select the Zoom to active area option.

   The default is on.

   If Zoom to active areas is checked, the visual automatically focuses on the parts of the map that have plotted dimensions. In the example of the world map used here, this includes all the continents except Antarctica.

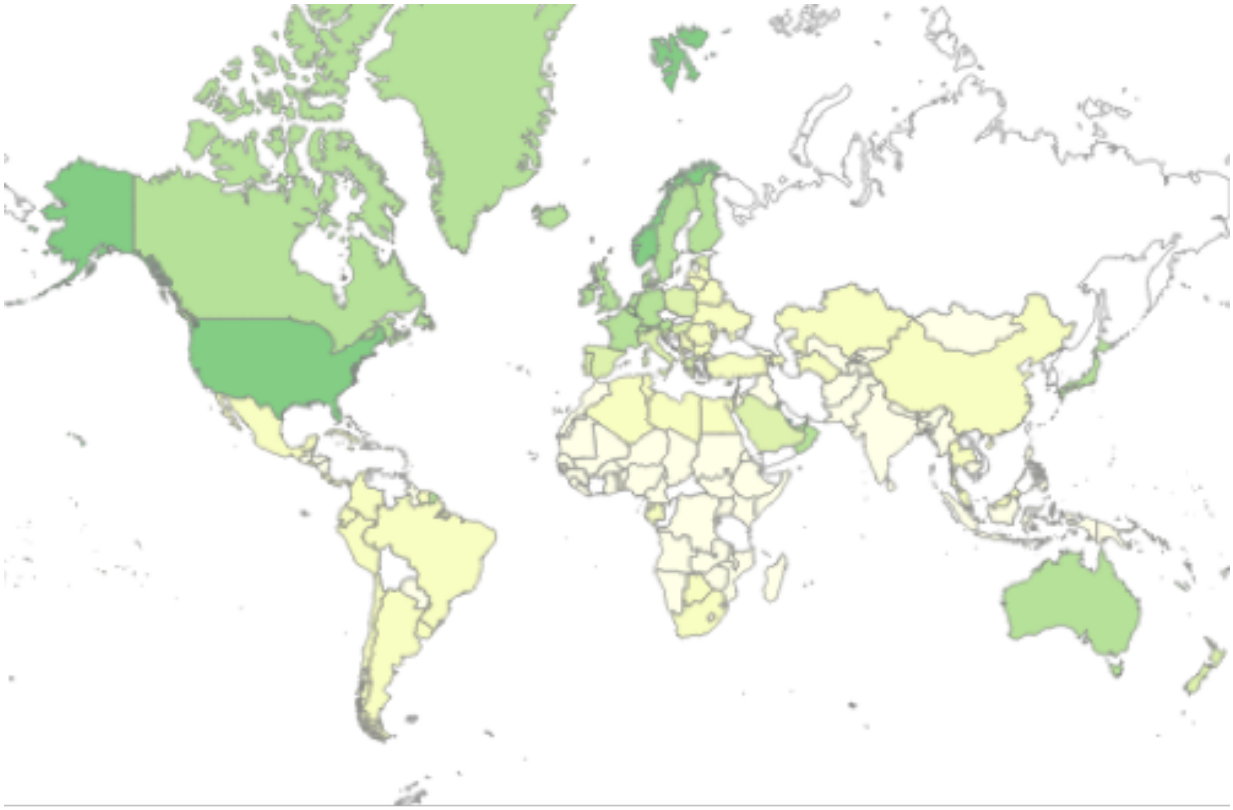

   If this option is de-selected, the visual shows the entire map, including the over-size continent of Antarctica.

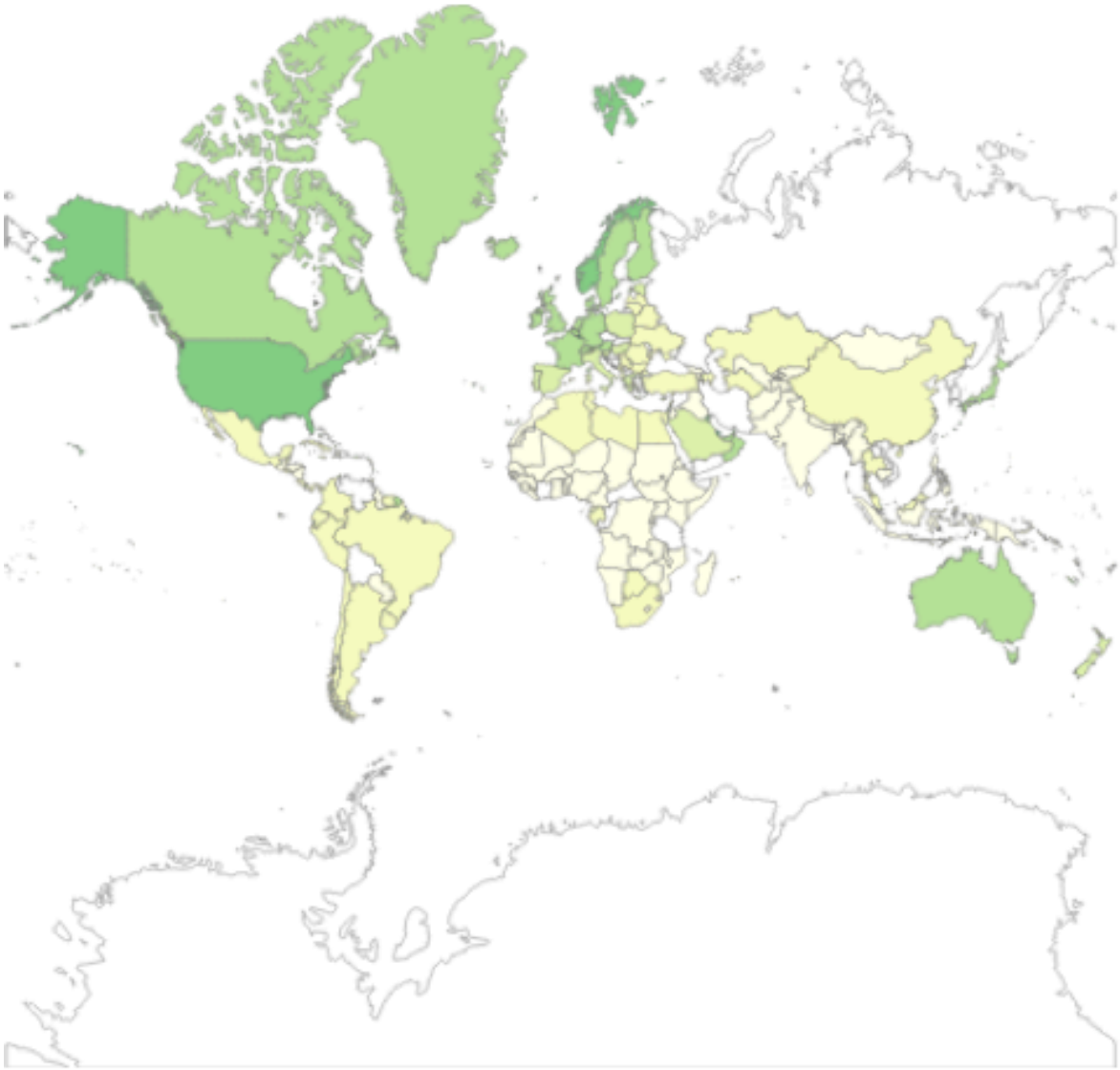

Customizing visual style

In Cloudera Data Visualization, Visual Style settings determine the general display of the visual. See the following topics for working with different style options.

Customizing visual basics

In Cloudera Data Visualization, you can modify a visual's background color and also choose from various font display options for the text that appears in the visual.

Changing font family in a visual

Procedure

1. On the right side of Visual Designer, click Style in the VISUAL section of the menu bar.

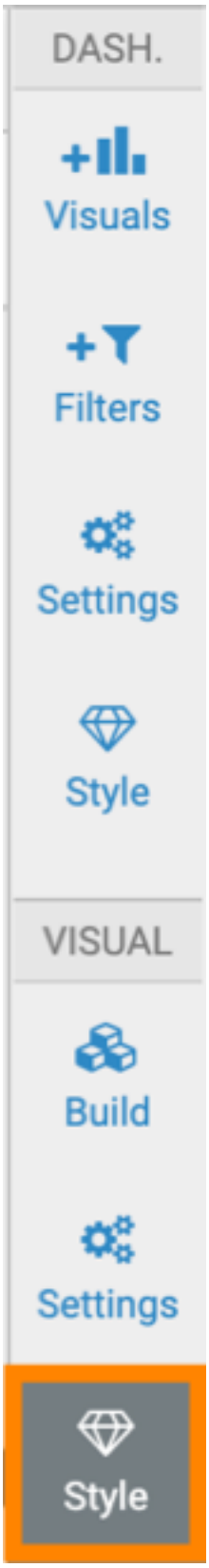

2. In the Style menu, click Visual Basics.

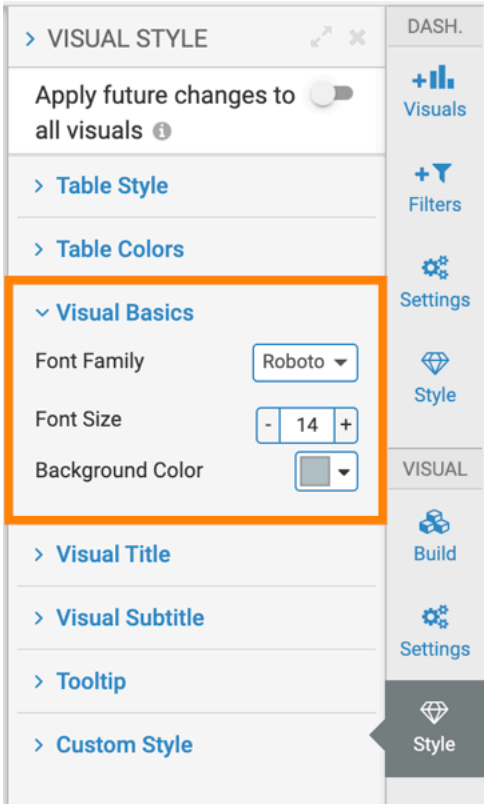

3. To change the font used in a visual, open the Font Family menu and select the required font type.

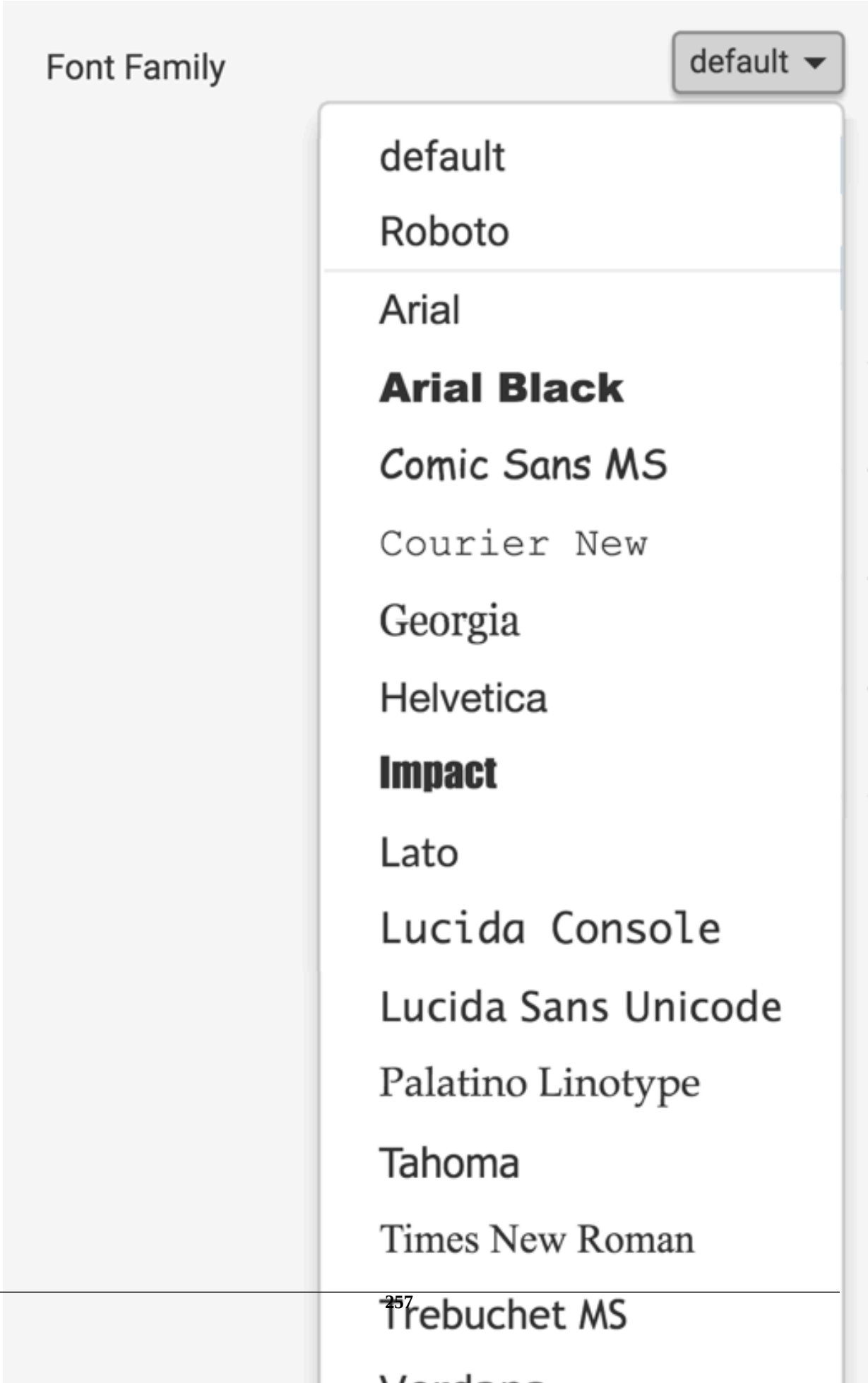
Font Family
default
default
Roboto
Arial
Arial Black
Comic Sans MS
Courier New
Georgia
Helvetica
Impact
Lato
Lucida Console
Lucida Sans Unicode
Palatino Linotype
Tahoma
Times New Roman
Trebuchet MS

Changing font size in a visual

Procedure

1. On the right side of Visual Designer, click Style in the VISUAL section of the menu bar.

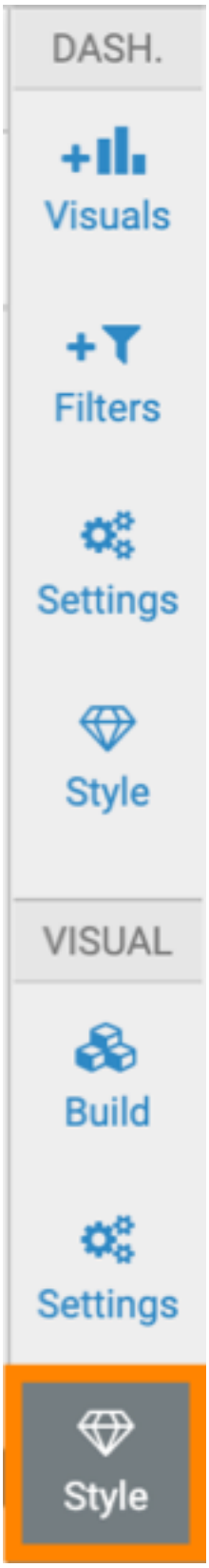

2. In the Style menu, click Visual Basics.

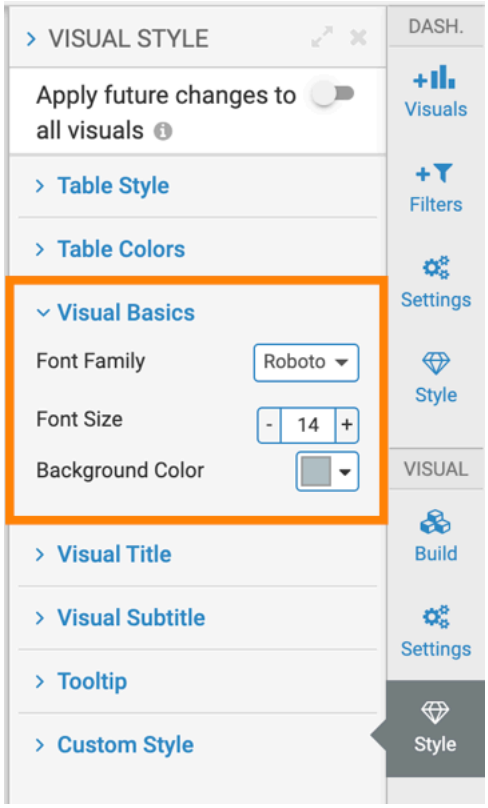

3. To change the font size used in a visual, adjust the size value in the Font Size selector.

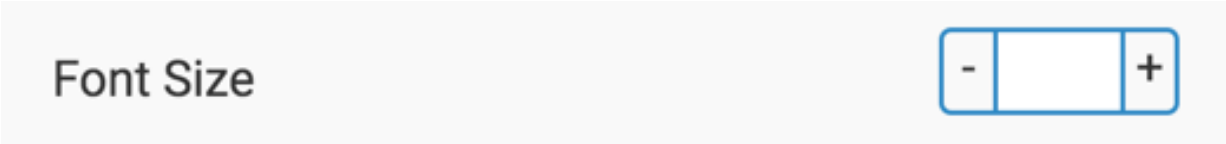

Changing background color of a visual

Procedure

1. On the right side of Visual Designer, click Style in the VISUAL section of the menu bar.

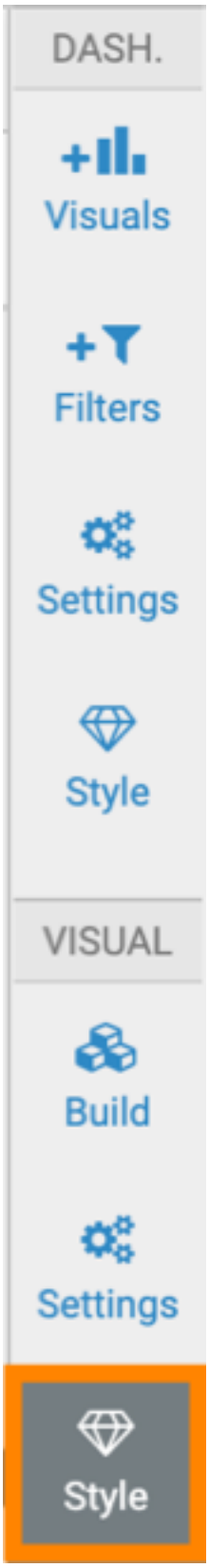

2. In the Style menu, click Visual Basics.

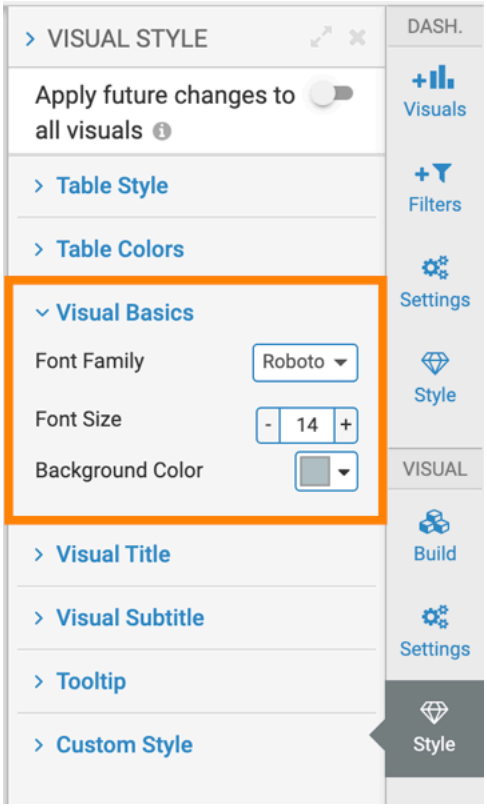

3. To change the background color of the visual, use the color selector.

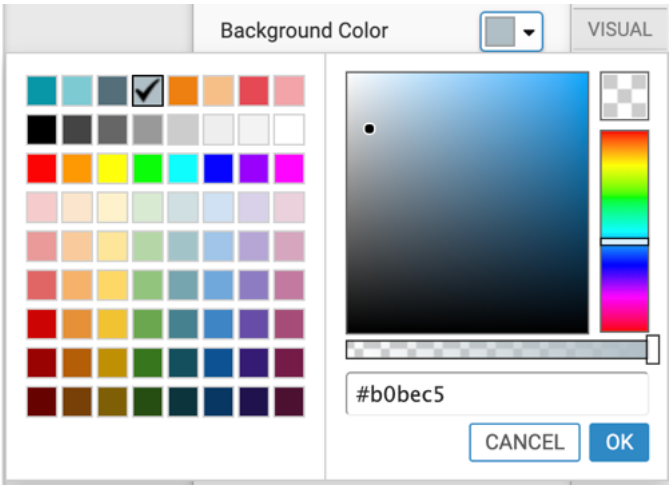

Customizing table style

In Cloudera Data Visualization, you can define various style options for your table visuals.

Changing table templates

In Cloudera Data Visualization, you can choose from various table templates when creating a Table or a Queries visual.

Procedure

1. On the right side of Visual Designer, click Style in the VISUAL section of the menu bar.

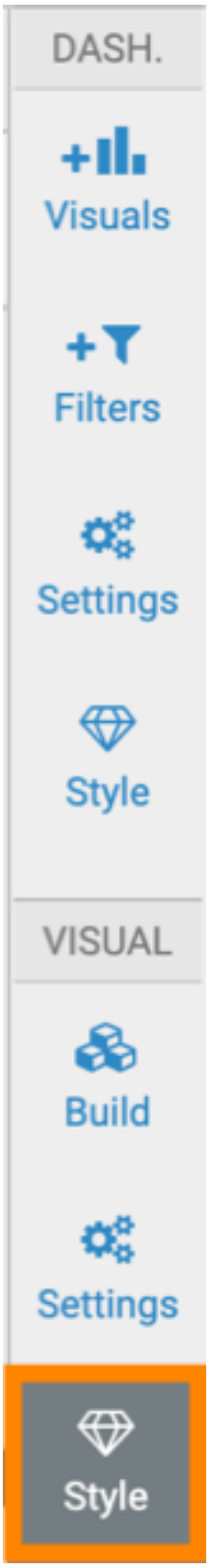

2. In the Style menu, click Table Style.

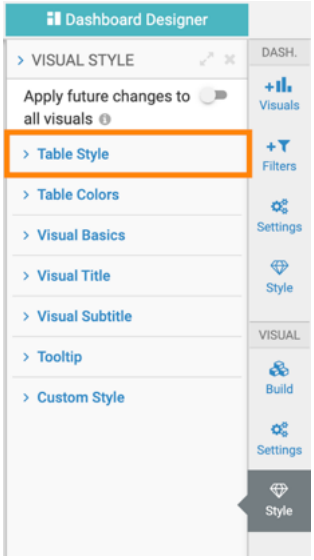

3. In the Table Templates section, select one of the styles.

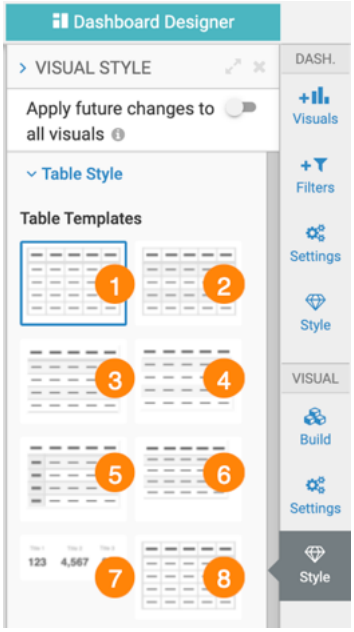

Style 1

Table with internal and external borders

year	state	sum(population)
2010	AK	710231
2010	AL	4779736
2010	AR	2915918
2010	AZ	6392017
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DC	601723
2010	DE	897934
2010	FL	18801310

Style 2

Table with internal and external borders, and shading on alternating rows

year	state	sum(population)
2010	AK	710231
2010	AL	4779736
2010	AR	2915918
2010	AZ	6392017
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DC	601723
2010	DE	897934
2010	FL	18801310

Style 3

Table with borders between rows only and shading on alternating rows

year	state	sum(population)
2010	AK	710231
2010	AL	4779736
2010	AR	2915918
2010	AZ	6392017
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DC	601723
2010	DE	897934
2010	FL	18801310

Style 4

Table with a single border between the header row and record rows, and shading on alternating rows

year	state	sum(population)
2010	AK	710231
2010	AL	4779736
2010	AR	2915918
2010	AZ	6392017
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DC	601723
2010	DE	897934
2010	FL	18801310

Style 5

Table with borders between rows, shading on alternating rows, and deeper shading and header format on the first column

year	state	sum(population)
2010	AK	710231
2010	AL	4779736
2010	AR	2915918
2010	AZ	6392017
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DC	601723
2010	DE	897934
2010	FL	18801310

Style 6

Table with borders between rows only, shading on alternating rows, and condensed spacing on rows

year	state	sum(population)
2010	AK	710231
2010	AL	4779736
2010	AR	2915918
2010	AZ	6392017
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DC	601723
2010	DE	897934
2010	FL	18801310

Style 7

Large and bold font on data rows, header style – ideal where a single table row can be used in a visual

year	state	sum(population)
2010	AK	710231

Style 8

Table with internal and external borders, shading on alternating rows, and condensed spacing on columns

year	state	sum(po...
2010	AK	710231
2010	AL	4779736
2010	AR	2915918
2010	AZ	6392017
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DC	601723
2010	DE	897934
2010	FL	18801310

Changing cross tabulation header alignment

In Cloudera Data Visualization, you can change the header alignment for cross tabulation visuals.

Procedure

1. On the right side of Visual Designer, click Style in the VISUAL section of the menu bar.

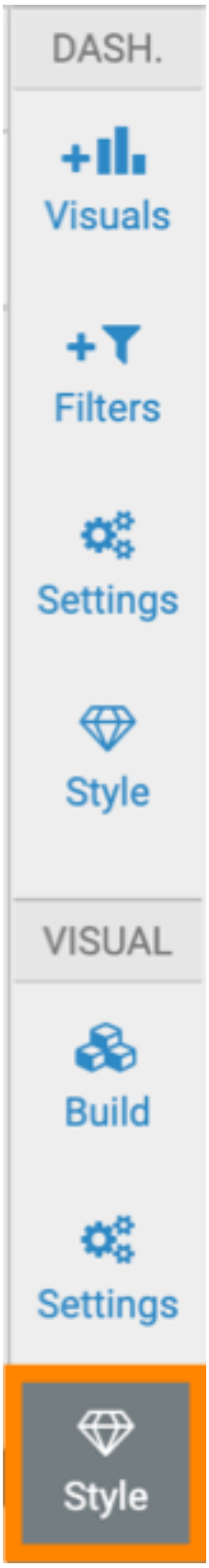

2. In the Style menu, click Table Style.

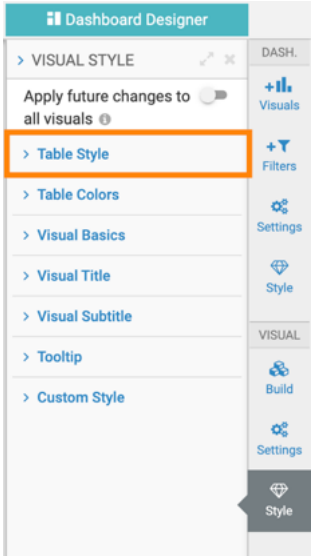

3. In the Table Header Alignment section of the Table Style menu, select one of the options.

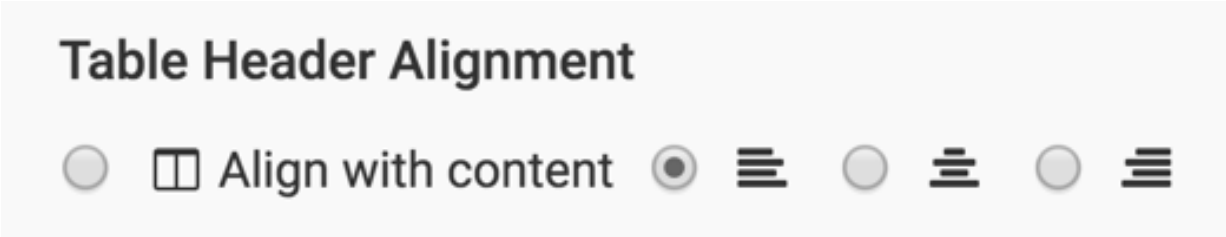

Align with content

This setting shows table headers aligned in the same manner as the column data. This is the default setting.

In the following example, states are aligned to the left, and population data is aligned to the right.

year	state	population
2010	AL	4779736
2010	AK	710231
2010	AZ	6392017
2010	AR	2915918
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DE	897934
2010	DC	601723
2010	FL	18801310

Left alignment

This settings shows table headers aligned to the left in each column.

year	state	population
2010	AL	4779736
2010	AK	710231
2010	AZ	6392017
2010	AR	2915918
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DE	897934
2010	DC	601723
2010	FL	18801310

Right alignment

This setting shows table headers aligned to the right in each column.

year	state	population
2010	AL	4779736
2010	AK	710231
2010	AZ	6392017
2010	AR	2915918
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DE	897934
2010	DC	601723
2010	FL	18801310

Center alignment

This setting shows table headers aligned at the center of each column.

year	state	population
2010	AL	4779736
2010	AK	710231
2010	AZ	6392017
2010	AR	2915918
2010	CA	37253956
2010	CO	5029196
2010	CT	3574097
2010	DE	897934
2010	DC	601723
2010	FL	18801310

Adjusting table width

You can set a Table, a Cross Tabulation or a Queries visual to full width to ensure that they fill the entire width of the page.

Procedure

1. On the right side of Visual Designer, click Style in the VISUAL section of the menu bar.

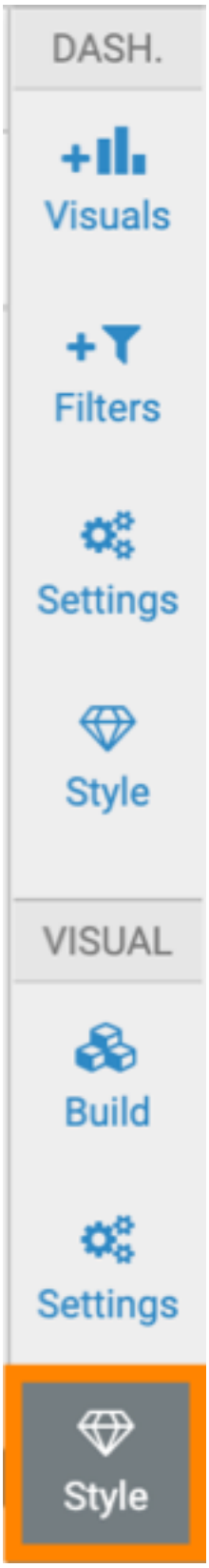

2. In the Style menu, click Table Style.

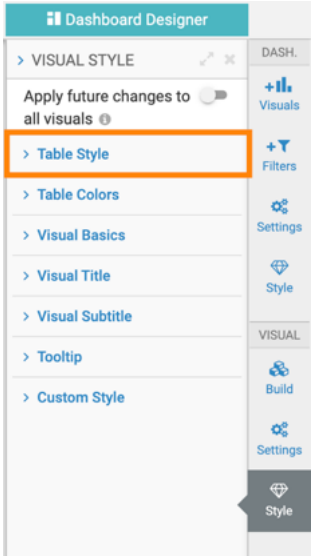

3. Select the Full table width option.

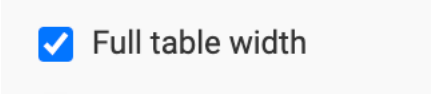

Example

In the following example, compare the two visuals and note the change in the appearance of the table. The table on the right is set to full width.

year	state	population
2010	AK	710,231
2010	AL	4,779,736
2010	AR	2,915,918
2010	AZ	6,392,017
2010	CA	37,253,956
2010	CO	5,029,196
2010	CT	3,574,097
2010	DC	601,723
2010	DE	897,934
2010	FL	18,801,310

year	state	population
2010	AK	710,231
2010	AL	4,779,736
2010	AR	2,915,918
2010	AZ	6,392,017
2010	CA	37,253,956
2010	CO	5,029,196
2010	CT	3,574,097
2010	DC	601,723
2010	DE	897,934
2010	FL	18,801,310

Important: The Full table width option always uses the current viewport as a reference. This means that the result will be different depending on the size of the screen and the visual.

Wrapping column contents

In Cloudera Data Visualization, you can wrap content in table columns. If the text does not wrap, you can hover over the text to view full text.

Wrapping content

To wrap content in the columns of Table and Queries visuals:

1. On the right side of the Visual Designer, click Style in the VISUAL section of the menu bar.

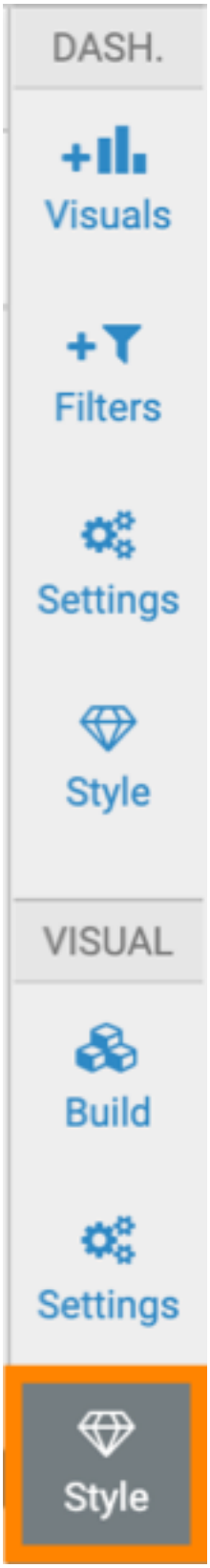

2. In the Style menu, click Table Style.

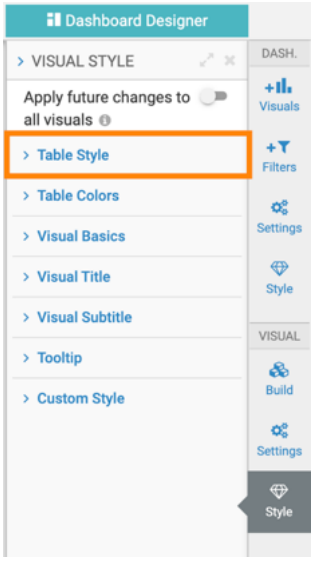

3. Select the Wrap column contents option.

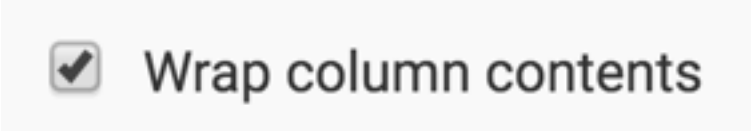

In the following example, compare the two visuals – in the table on the right, the content in the first column wraps after selecting this option.

pickup_neighborhood	sum(pickup_ho...
Airport	276
Astoria	276
Baisley Park	209
Bath Beach	2
Battery Park City-Lower Manhattan	276
Bay Ridge	116

pickup_neighborhood	sum(pickup_ho...
Airport	276
Astoria	276
Baisley Park	209
Bath Beach	2
Battery Park City-Lower Manhattan	276
Bay Ridge	116

Viewing full text

If the Wrap column contents option is not selected, and the text does not fit the width of the column, the text is displayed with a series of three dots (ellipsis – …) at the end of the table cell. Hover over the text to view the full text.

In the following example, the highlighted column text is displayed as Battery Park City.... If you hover over the text to view the full text, you can see the full text: Battery Park City-Lower Manhattan.

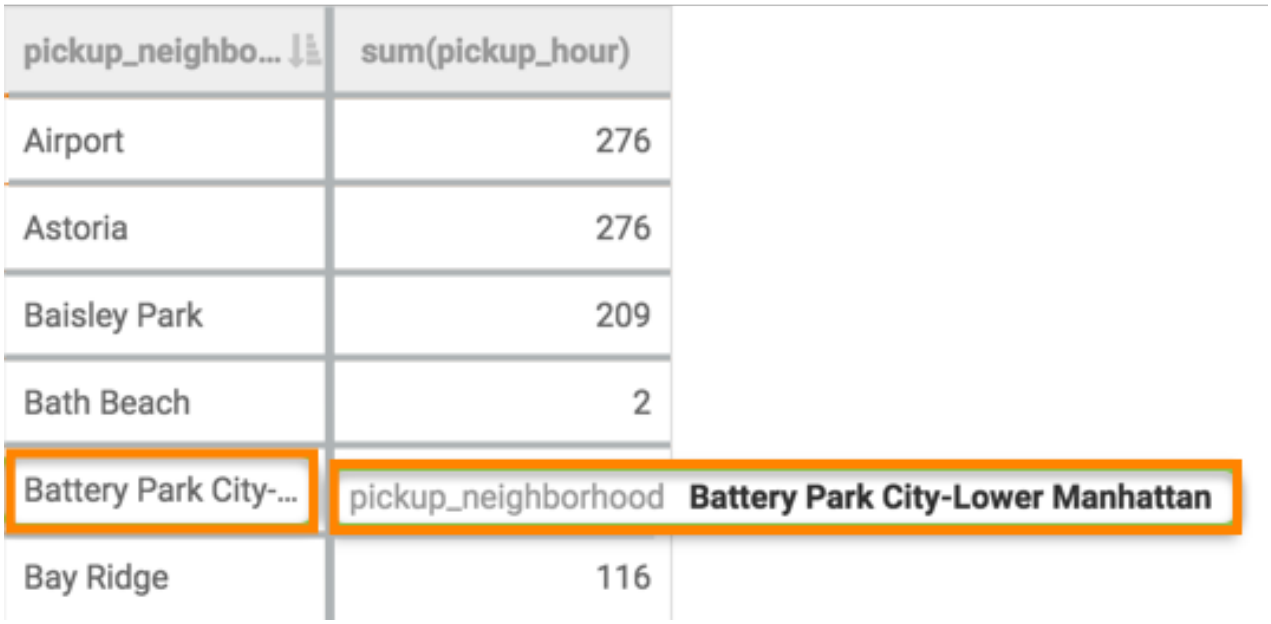

pickup_neighbo...	sum(pickup_hour)
Airport	276
Astoria	276
Baisley Park	209
Bath Beach	2
Battery Park City-...	pickup_neighborhood Battery Park City-Lower Manhattan
Bay Ridge	116

Showing column totals

In Cloudera Data Visualization, you can add table and queries visuals report values in rows by default. But there is an option of adding a totals row to a table and use it to present column totals.

Procedure

1. On the right side of Visual Designer, click Settings in the VISUAL section of the menu.

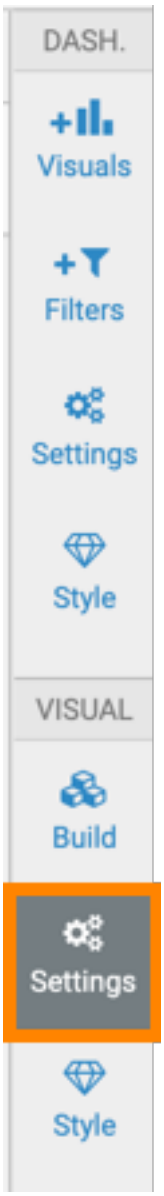

2. Open the Data Summary menu and select the Show totals option.

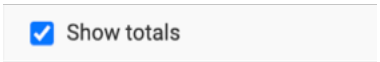

Note: The Data Summary menu is different for Cross Tabulation visuals. You can configure both column totals and row totals. Use the Show column totals options to add column totals.

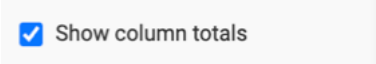

For information on row totals, see *Showing row totals*.

In this table, you can see the totals reported for numerical measurement columns:

year	state	sum(population)
2010	AK	710,231
2010	AL	4,779,736
2010	AR	2,915,918
2010	AZ	6,392,017
2010	CA	37,253,956
2010	CO	5,029,196
2010	CT	3,574,097
2010	DC	601,723
2010	DE	897,934
2010	FL	18,801,310
		80,956,118

Showing header for column totals

In Cloudera Data Visualization, if you use column totals in your Table and Queries visuals, you can also add a header for column totals.

Procedure

1. On the right side of Visual Designer, click Settings in the VISUAL section of the menu.

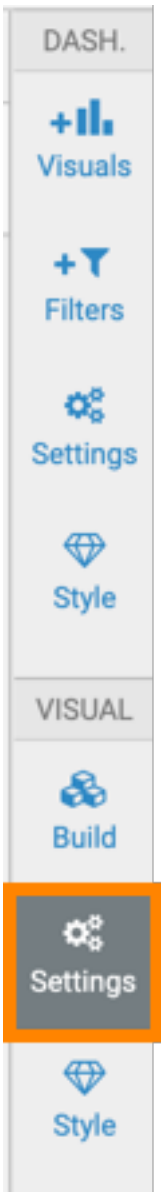

2. Open the Data Summary menu and select the Show totals header option.

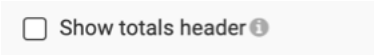

Note: This option is only applied to the visual if Show totals is selected.

In this table, you can see the total number of population reported for the sum(population) column in the Totals row:

year	state	sum(population)
2010	AK	710,231
2010	AL	4,779,736
2010	AR	2,915,918
2010	AZ	6,392,017
2010	CA	37,253,956
2010	CO	5,029,196
2010	CT	3,574,097
2010	DC	601,723
2010	DE	897,934
2010	FL	18,801,310
Totals		**80,956,118**

Showing column page totals

In Cloudera Data Visualization, you can add Table, and Queries visuals report values in rows by default. But there is an option of adding a subtotals row to a table and use it to present column page totals, which is the total of the currently viewed page.

Procedure

1. On the right side of Visual Designer, click Settings in the VISUAL section of the menu.

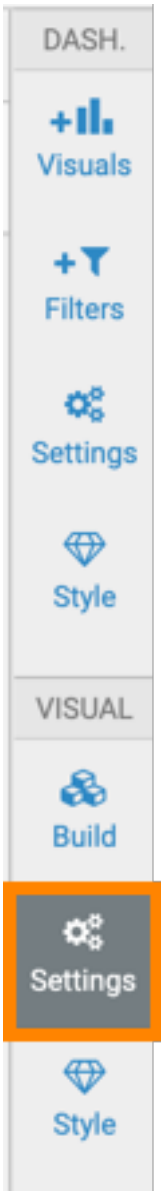

2. Open the Data Summary menu and select the Show page totals option.

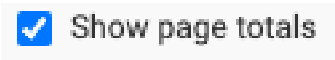

Results

The visual now has a row showing the total of currently viewed page:

year	state	sum(population)
2010	AK	710,231
2010	AL	4,779,736
2010	AR	2,915,918
2010	AZ	6,392,017
2010	CA	37,253,956
2010	CO	5,029,196
2010	CT	3,574,097
2010	DC	601,723
2010	DE	897,934
2010	FL	18,801,310
2010	GA	9,687,653
2010	HI	1,360,301
		134,205,929

< 1 2 3 >

Showing header for column page totals

In Cloudera Data Visualization, if you use column totals in your Table and Queries visuals, you can also add a header for column totals.

Procedure

1. On the right side of Visual Designer, click Settings in the VISUAL section of the menu.

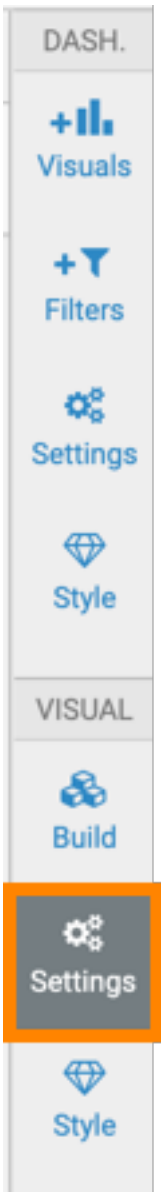

2. Open the Data Summary menu and select the Show totals header option.

Results

The visual now has a row with heading showing the total of currently viewed page:

year	state	sum(population)
2010	AK	710,231
2010	AL	4,779,736
2010	AR	2,915,918
2010	AZ	6,392,017
2010	CA	37,253,956
2010	CO	5,029,196
2010	CT	3,574,097
2010	DC	601,723
2010	DE	897,934
2010	FL	18,801,310
2010	GA	9,687,653
2010	HI	1,360,301
Page totals		134,205,929

< 1 2 3 >

Specifying totals column font weight

Cloudera Data Visualization allows you to adjust the font style of the totals column.

1. On the right side of the Visual Designer, click Style in the VISUAL section of the menu bar.

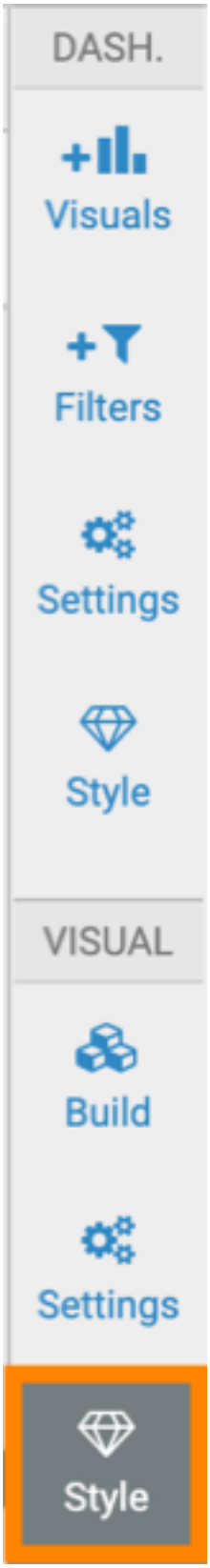

2. In the Style menu, click Table Style.

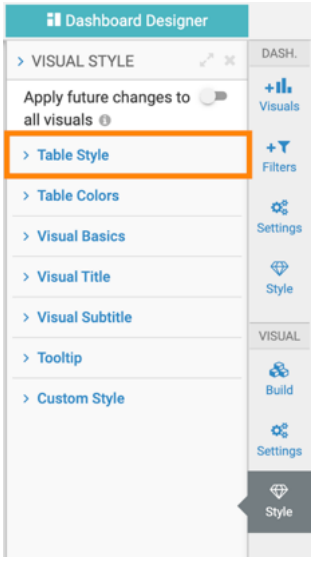

3. Adjust the font style from the Totals Column Font Weight menu.

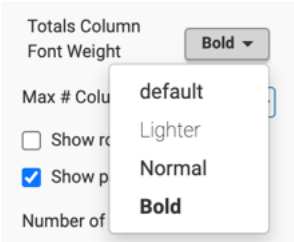

Specifying maximum number of columns

Cloudera Data Visualization allows you to limit the number of columns showing for wide tables.

1. On the right side of the Visual Designer, click Style in the VISUAL section of the menu bar.

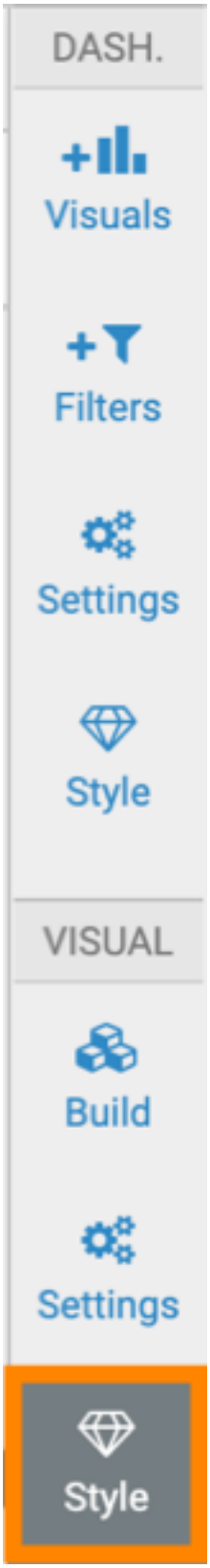

2. In the Style menu, click Table Style.

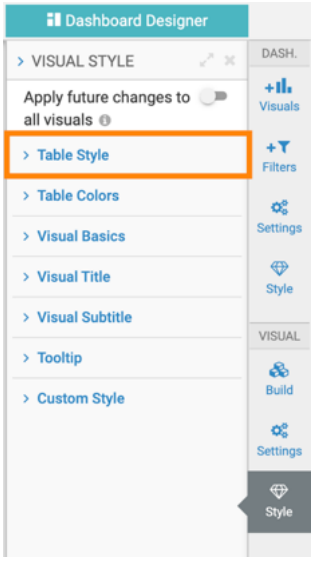

3. adjust the selector for the Max # Columns option. The default is 20.

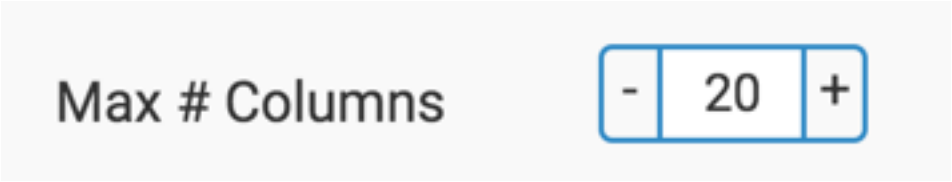

If the value of this option is set to 5, a 10-column table shows the first 5 columns.

id	sepal_length	sepal_width	petal_length	petal_width	species	sepal_area	petal_area	sepal_ratio	petal_ratio
1	5.1	3.5	1.4	0.2	setosa	17.85	0.28	1.45714285714	7
2	4.9	3	1.4	0.2	setosa	14.7	0.28	1.63333333333	7
3	4.7	3.2	1.3	0.2	setosa	15.04	0.26	1.46875	6.5
4	4.6	3.1	1.5	0.2	setosa	14.26	0.3	1.48387096774	7.5
5	5	3.6	1.4	0.2	setosa	18	0.28	1.38888888889	7
6	5.4	3.9	1.7	0.4	setosa	21.06	0.68	1.38461538462	4.25
7	4.6	3.4	1.4	0.3	setosa	15.64	0.42	1.35294117647	4.66666666667
8	5	3.4	1.5	0.2	setosa	17	0.3	1.47058823529	7.5
9	4.4	2.9	1.4	0.2	setosa	12.76	0.28	1.51724137931	7
10	4.9	3.1	1.5	0.1	setosa	15.19	0.15	1.58064516129	15
11	5.4	3.7	1.5	0.2	setosa	19.98	0.3	1.45945945946	7.5
12	4.8	3.4	1.6	0.2	setosa	16.32	0.32	1.41176470588	8
13	4.8	3	1.4	0.1	setosa	14.4	0.14	1.6	14
14	4.3	3	1.1	0.1	setosa	12.9	0.11	1.43333333333	11
15	5.8	4	1.2	0.2	setosa	23.2	0.24	1.45	6
16	5.7	4.4	1.5	0.4	setosa	25.08	0.6	1.29545454545	3.75
17	5.4	3.9	1.3	0.4	setosa	21.06	0.52	1.38461538462	3.25
18	5.1	3.5	1.4	0.3	setosa	17.85	0.42	1.45714285714	4.66666666667

id	sepal_length	sepal_width	petal_length	petal_width
1	5.1	3.5	1.4	0.2
2	4.9	3	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4
7	4.6	3.4	1.4	0.3
8	5	3.4	1.5	0.2
9	4.4	2.9	1.4	0.2
10	4.9	3.1	1.5	0.1
11	5.4	3.7	1.5	0.2
12	4.8	3.4	1.6	0.2
13	4.8	3	1.4	0.1
14	4.3	3	1.1	0.1
15	5.8	4	1.2	0.2
16	5.7	4.4	1.5	0.4
17	5.4	3.9	1.3	0.4
18	5.1	3.5	1.4	0.3

Showing row information

Cloudera Data Visualization allows you to display the number of rows in your tables.

To show row information for table and queries visuals:

1. On the right side of the Visual Designer, click Style in the VISUAL section of the menu bar.

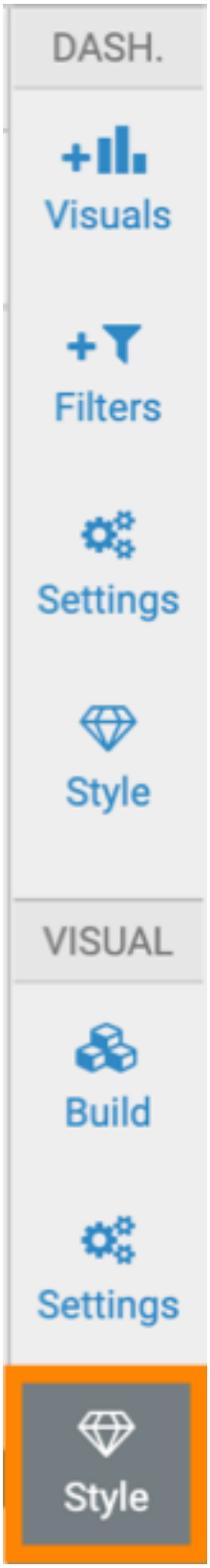

2. In the Style menu, click Table Style.

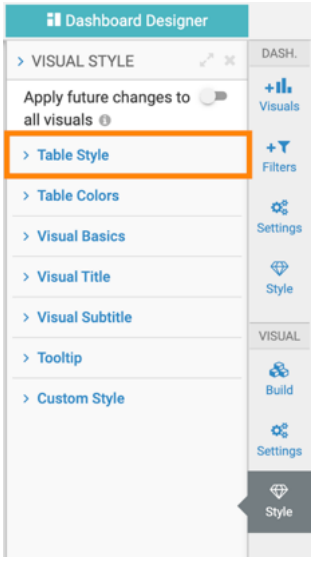

3. Select the Show row information option.

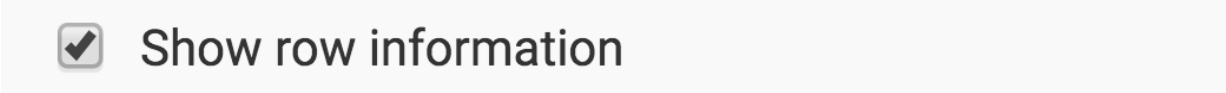

This visual displays the number of table rows below the table.

year	state	population
2010	AK	710,231
2010	AL	4,779,736
2010	AR	2,915,918
2010	AZ	6,392,017
2010	CA	37,253,956
2010	CO	5,029,196
2010	CT	3,574,097
2010	DC	601,723
2010	DE	897,934
2010	FL	18,801,310
2010	GA	9,687,653
2010	HI	1,360,301
2010	IA	3,046,355
2010	ID	1,567,582

Showing all 51 rows

Showing pagination

With very long tables, scrolling down can become annoying. Cloudera Data Visualization allows you to display pagination for your tables, but by default, tables do not display page numbers.

To turn pagination on for tables and queries:

1. On the right side of the Visual Designer, click Style in the VISUAL section of the menu bar.

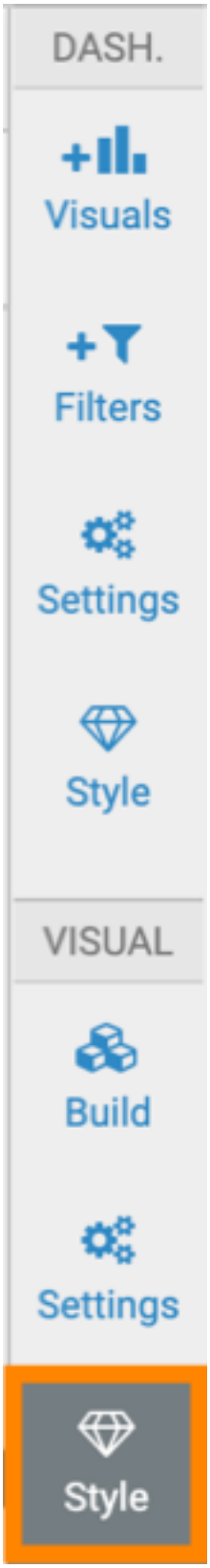

2. In the Style menu, click Table Style.

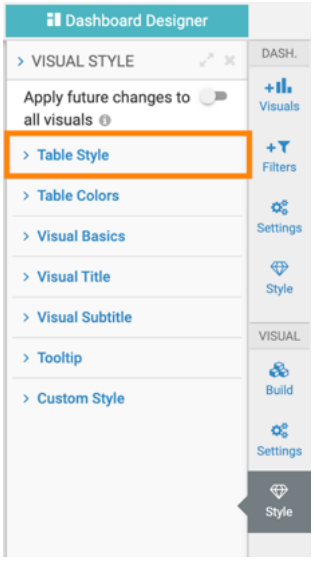

3. Select the Show pagination option. You must also specify the number of rows per page.

   By default, pagination is off.

Pagination, with 10 rows per page

In this example, pagination is turned on, and the number of rows per page is set to 10. The table shortens, and an interactive page selector appears at the bottom right corner of the visual.

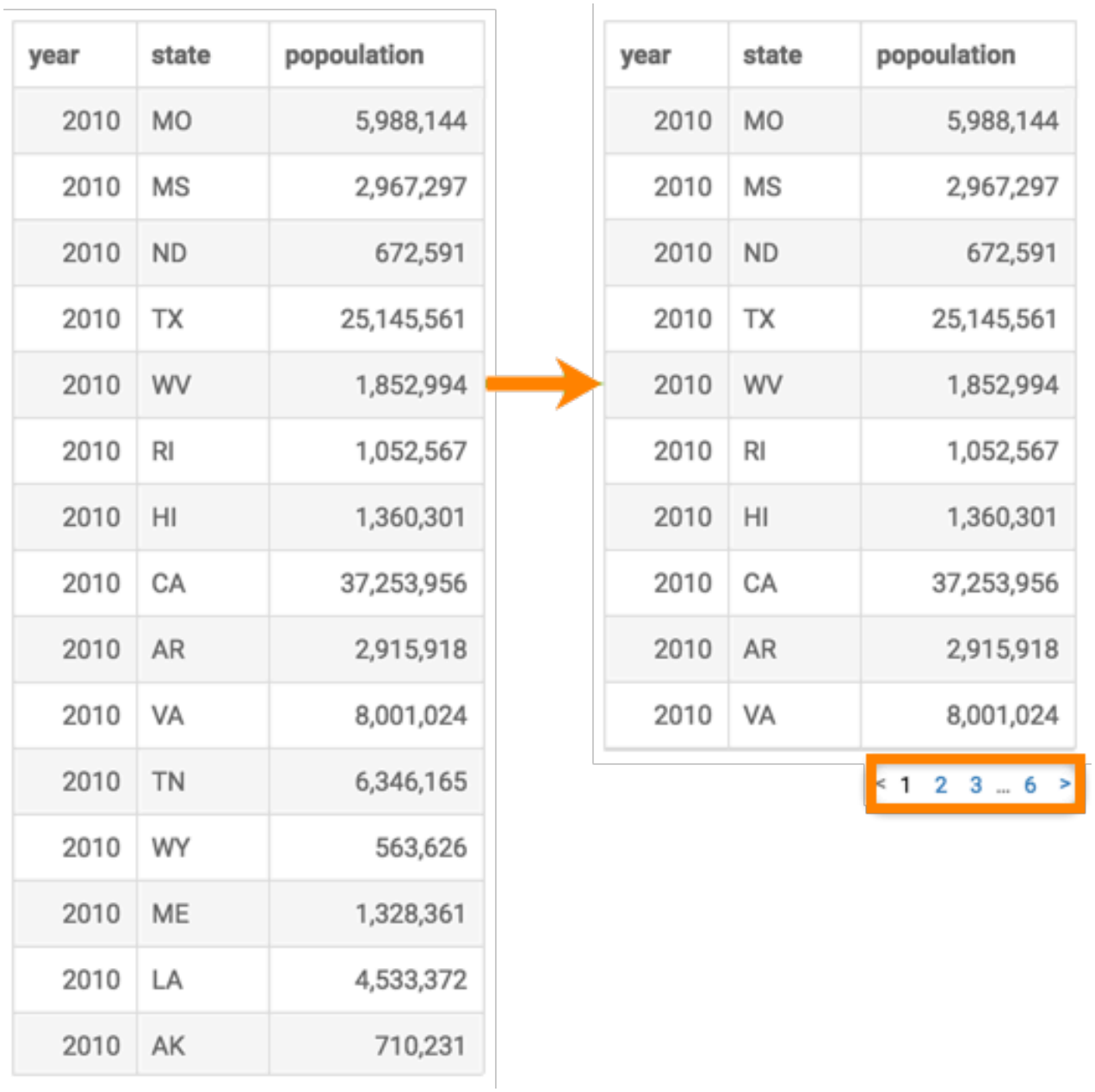

year	state	popoulation
2010	MO	5,988,144
2010	MS	2,967,297
2010	ND	672,591
2010	TX	25,145,561
2010	WV	1,852,994
2010	RI	1,052,567
2010	HI	1,360,301
2010	CA	37,253,956
2010	AR	2,915,918
2010	VA	8,001,024
2010	TN	6,346,165
2010	WY	563,626
2010	ME	1,328,361
2010	LA	4,533,372
2010	AK	710,231

year	state	popoulation
2010	MO	5,988,144
2010	MS	2,967,297
2010	ND	672,591
2010	TX	25,145,561
2010	WV	1,852,994
2010	RI	1,052,567
2010	HI	1,360,301
2010	CA	37,253,956
2010	AR	2,915,918
2010	VA	8,001,024

< 1 2 3 ... 6 >

Pagination and totals

If you select both the pagination and totals options, the table reports the same total on each page, and that is the overall total.

year	state	popoulation
2010	MO	5,988,144
2010	MS	2,967,297
2010	ND	672,591
2010	TX	25,145,561
2010	WV	1,852,994
2010	RI	1,052,567
2010	HI	1,360,301
2010	CA	37,253,956
2010	AR	2,915,918
2010	VA	8,001,024
Totals		**308744815**

< 1 2 3 ... 6 >

Specifying number of rows per page

To specify the number of rows per page:

1. On the right side of the Visual Designer, click Style in the VISUAL section of the menu bar.

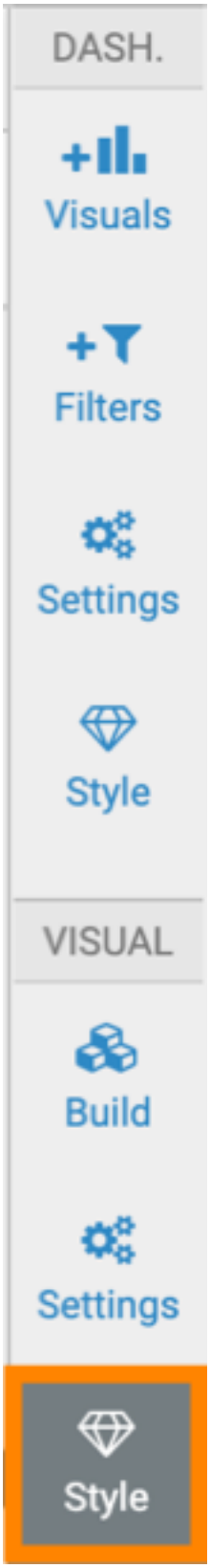

2. In the Style menu, click Table Style.

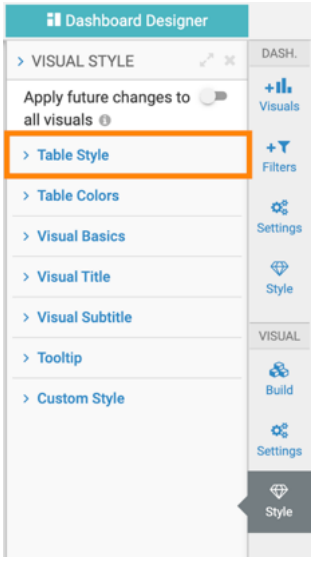

3. Enter a positive value for the Number of rows per page option.

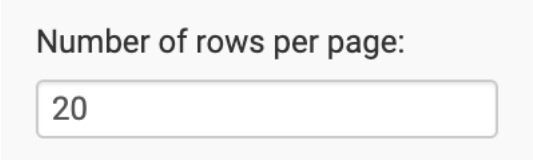

For information on how to set the pagination for your visual, see *Showing pagination*.

Related Information

Showing pagination

Showing header column

In Cloudera Data Visualization, header columns of table and queries visuals have a border by default.

To show or hide header column borders for Tables and Queries:

1. On the right side of the Visual Designer, click Style in the VISUAL section of the menu bar.

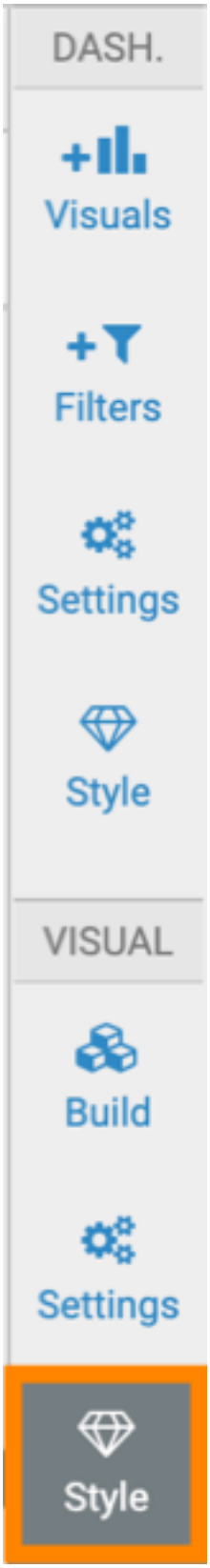

2. In the Style menu, click Table Style.

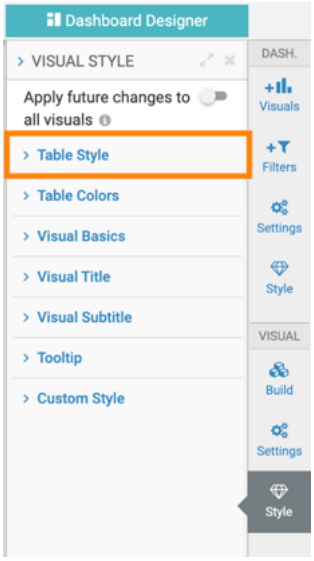

3. Select or deselect the Display header option.

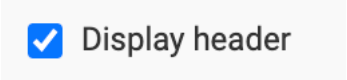

Example

In the following example, you can see a table without a table header and a table after selecting this option.

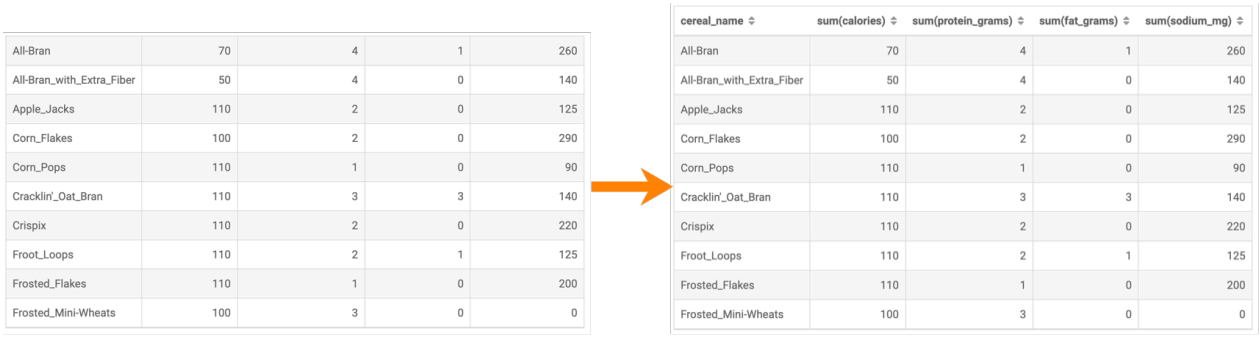

All-Bran	70	4	1	260
All-Bran_with_Extra_Fiber	50	4	0	140
Apple_Jacks	110	2	0	125
Corn_Flakes	100	2	0	290
Corn_Pops	110	1	0	90
Cracklin'_Oat_Bran	110	3	3	140
Crispix	110	2	0	220
Froot_Loops	110	2	1	125
Frosted_Flakes	110	1	0	200
Frosted_Mini-Wheats	100	3	0	0

cereal_name	sum(calories)	sum(protein_grams)	sum(fat_grams)	sum(sodium_mg)
All-Bran	70	4	1	260
All-Bran_with_Extra_Fiber	50	4	0	140
Apple_Jacks	110	2	0	125
Corn_Flakes	100	2	0	290
Corn_Pops	110	1	0	90
Cracklin'_Oat_Bran	110	3	3	140
Crispix	110	2	0	220
Froot_Loops	110	2	1	125
Frosted_Flakes	110	1	0	200
Frosted_Mini-Wheats	100	3	0	0

Showing header column borders

In Cloudera Data Visualization, header columns in table and queries visuals have a border by default.

To show or hide header column borders for tables and queries:

1. On the right side of the Visual Designer, click Style in the VISUAL section of the menu bar.

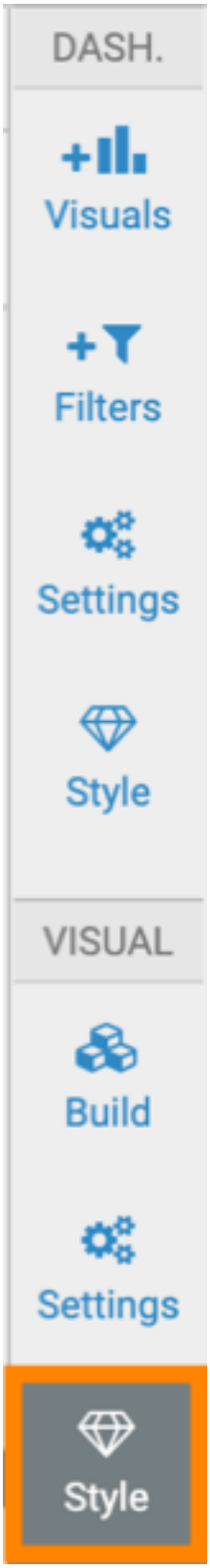

2. In the Style menu, click Table Style.

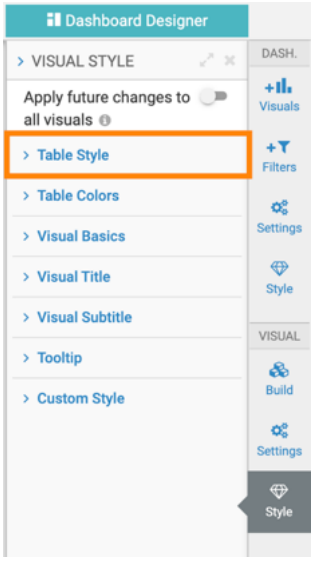

3. Select or deselect the Display header column borders option.

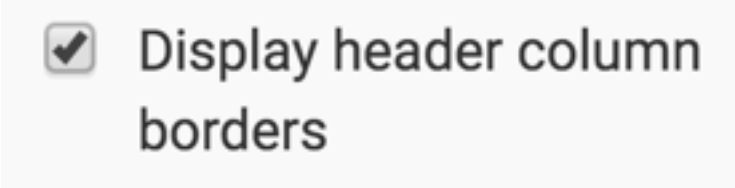

Example

In the following example, you can see the column header borders before and after selecting this option.

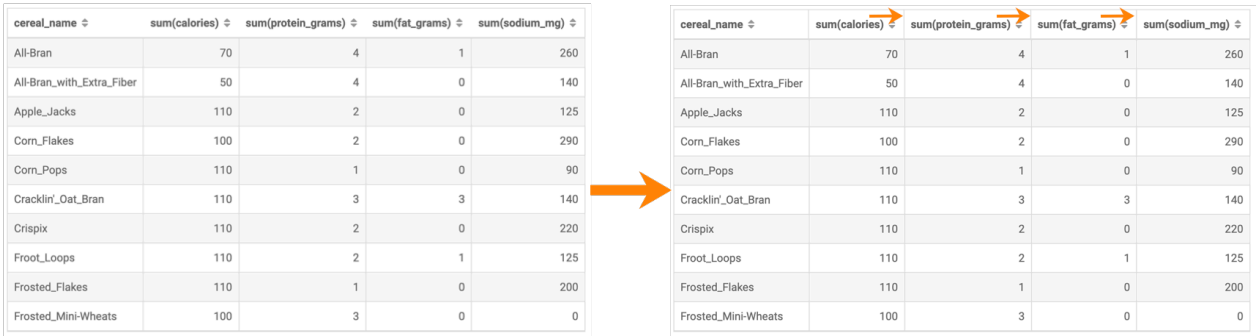

cereal_name	sum(calories)	sum(protein_grams)	sum(fat_grams)	sum(sodium_mg)
All-Bran	70	4	1	260
All-Bran_with_Extra_Fiber	50	4	0	140
Apple_Jacks	110	2	0	125
Corn_Flakes	100	2	0	290
Corn_Pops	110	1	0	90
Cracklin'_Oat_Bran	110	3	3	140
Crispix	110	2	0	220
Froot_Loops	110	2	1	125
Frosted_Flakes	110	1	0	200
Frosted_Mini-Wheats	100	3	0	0

cereal_name	sum(calories)	sum(protein_grams)	sum(fat_grams)	sum(sodium_mg)
All-Bran	70	4	1	260
All-Bran_with_Extra_Fiber	50	4	0	140
Apple_Jacks	110	2	0	125
Corn_Flakes	100	2	0	290
Corn_Pops	110	1	0	90
Cracklin'_Oat_Bran	110	3	3	140
Crispix	110	2	0	220
Froot_Loops	110	2	1	125
Frosted_Flakes	110	1	0	200
Frosted_Mini-Wheats	100	3	0	0

Adjusting table column width

In Cloudera Data Visualization, you can manually adjust, either increase or decrease, the width of columns in a table.

Procedure

1. In the table header, hover your cursor on the right side of the column border until the [cross-hairs] icon appears.
2. Click column border, drag it the to the desired width, and then release the cursor.

Example

In this example the width of a column in a table is increased.

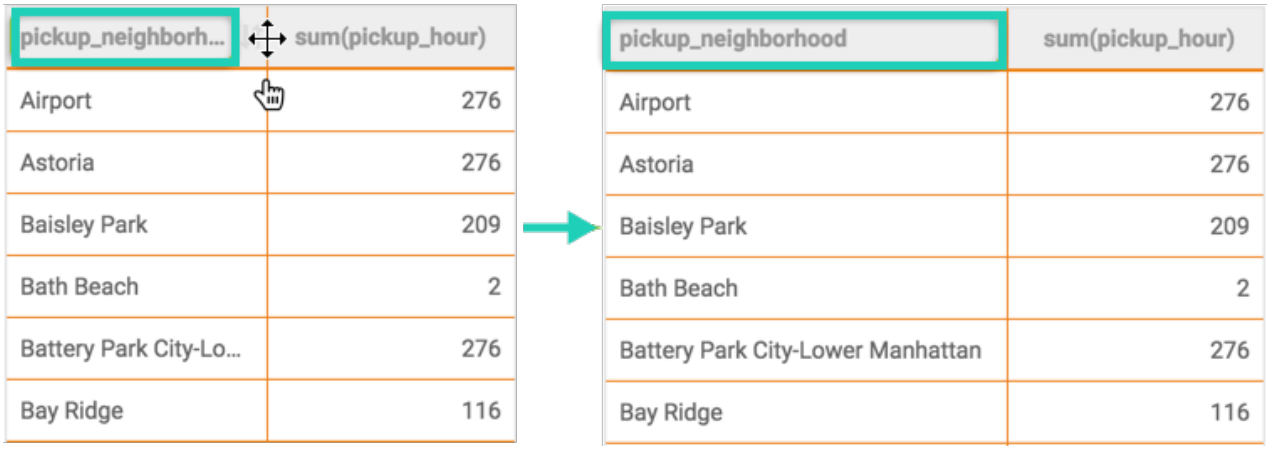

When the column width is smaller than the length of the text in the cell, the text is abbreviated, and a series of three dots (ellipsis – …) appear at the end of the line. The first column heading text of the left visual is displayed as 'pickup_neighborh...'. In such cases, you can hover over the table cell to see the full text.

Customizing cross tabulation style

In Cloudera Data Visualization, you can define various style options for your cross tabulation visuals.

Adjusting cross tabulation width

You can set a cross-tabulation visual to fill the entire width of the page in Cloudera Data Visualization.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.
4. Enable the Full table width option.

Important: The Full table width setting ensures that the table adjusts to the full width of the current viewport, meaning the display will vary based on the screen size and visual dimensions.

Adjusting cross tabulation column width

In Cloudera Data Visualization, you can enable automatic adjustment of column widths in a cross tabulation to ensure that header text is fully displayed, enhancing readability and maintaining the sorting feature.

About this task

When the column width is narrower than the text length in the header cell, the text is abbreviated with an ellipsis (…) at the end, and the sorting function is unavailable. By enabling auto sizing, you can ensure that all columns display the full header text and allow for sorting functionality.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.
4. Enable the Auto Size Columns by header option.

Changing cross tabulation header font

In Cloudera Data Visualization, you can customize the font used in cross-tabulation table headers.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.
4. In the Header Font menu, choose a new font.

   Options include the system default, Roboto, Arial, and more.

   For example, switching the font from the system default to Arial Black significantly changes the appearance of the headers.

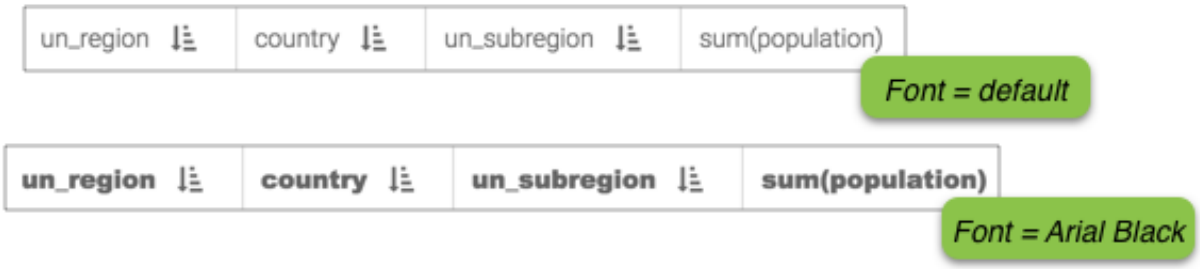

Changing table header font size

In Cloudera Data Visualization, you can adjust the font size of table headers in cross-tabulation visuals.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.
4. Locate the Header Font Size option and use the selector to adjust the size.

   The default size is system-specified.

   For example, increasing the header font size from 12 to 20 makes the headers more prominent and easier to read.

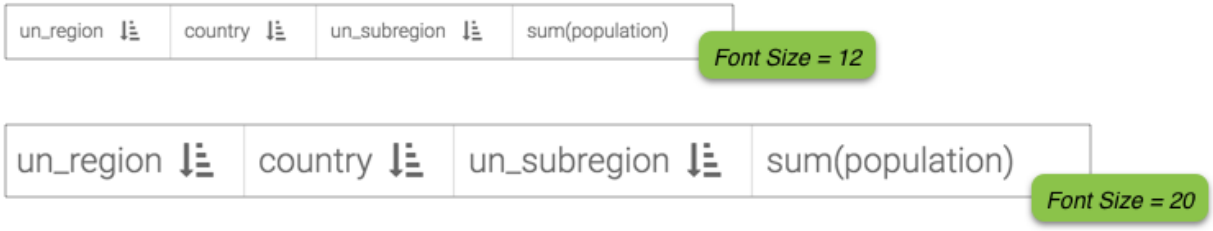

Changing cross tabulation header font weight

In Cloudera Data Visualization, you can adjust the font weight of table headers in cross-tabulation visuals.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.

4. Under the Header Font Weight option, select a new weight.

   Options include Lighter, Normal, and Bold, with the default weight being system-specified.

   For example, switching the font weight to Bold enhances the header's visibility and emphasis.

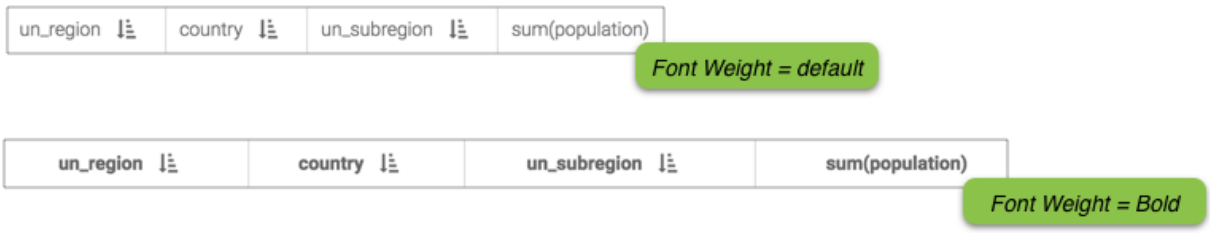

Changing cross tabulation cell padding

In Cloudera Data Visualization, you can customize the padding of cells in cross-tabulation visuals.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.
4. Adjust the Table Cell Padding value as needed.

   The default value is 8.

Changing cross tabulation border width

In Cloudera Data Visualization, you can customize the border width of cells in cross-tabulation visuals.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.
4. Adjust the Border Width value as needed.

   The default value is 1.

Showing cross tabulation column totals

In Cloudera Data Visualization, cross-tabulation visuals report row values by default. However, you can add a totals row to display column totals.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Data Summary.
4. Select the Show totals option.

Showing cross tabulation row totals

In Cloudera Data Visualization, you can add a totals row to cross tabulation visuals.

Procedure

1. Open the Dashboard Designer.

2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Data Summary.

4. Select the Show row totals option.

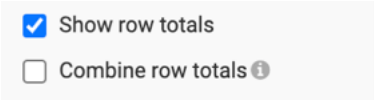

Once selected, the Combine row totals option becomes available. It is valid when there are multiple measures and at least one dimension on the Column shelf.

In the following example, the first visual shows data without row totals, while the second visual includes row totals.

	Country					
	European Union		NAFTA with Canada (Consump)		OPEC	
Year	Imports	Exports	Imports	Exports	Imports	Exports
1999	-$230B	$170B	-$120B	$58B	-$67B	$19B
2000	-$230B	$160B	-$110B	$60B	-$60B	$20B
2001	-$230B	$150B	-$120B	$61B	-$53B	$19B
2002	-$250B	$160B	-$120B	$63B	-$68B	$17B
2003	-$280B	$170B	-$130B	$67B	-$92B	$22B
2004	-$310B	$190B	-$150B	$76B	-$130B	$32B
2005	-$330B	$210B	-$160B	$83B	-$150B	$39B
2006	-$350B	$240B	-$160B	$87B	-$170B	$48B
2007	-$370B	$270B	-$170B	$89B	-$240B	$65B
2008	-$280B	$220B	-$110B	$72B	-$110B	$50B
2009	-$320B	$240B	-$150B	$86B	-$150B	$54B
2010	-$370B	$270B	-$160B	$93B	-$190B	$65B
2011	-$380B	$270B	-$180B	$99B	-$180B	$82B
2012	-$390B	$260B	-$180B	$100B	-$150B	$85B
2013	-$420B	$280B	-$170B	$100B	-$130B	$83B
2014	-$430B	$270B	-$140B	$93B	-$66B	$73B
2015	-$420B	$270B	-$130B	$90B	-$78B	$71B
2016	-$430B	$280B	-$130B	$110B	-$72B	$59B

	Country							
	European Union		NAFTA with Canada (Consump)		OPEC		Total Imports	Total Exports
Year	Imports	Exports	Imports	Exports	Imports	Exports		
1999	-$230B	$170B	-$120B	$58B	-$67B	$19B	-$420B	$250B
2000	-$230B	$160B	-$110B	$60B	-$60B	$20B	-$400B	$240B
2001	-$230B	$150B	-$120B	$61B	-$53B	$19B	-$400B	$230B
2002	-$250B	$160B	-$120B	$63B	-$68B	$17B	-$440B	$240B
2003	-$280B	$170B	-$130B	$67B	-$92B	$22B	-$510B	$260B
2004	-$310B	$190B	-$150B	$76B	-$130B	$32B	-$580B	$290B
2005	-$330B	$210B	-$160B	$83B	-$150B	$39B	-$640B	$330B
2006	-$350B	$240B	-$160B	$87B	-$170B	$48B	-$680B	$380B
2007	-$370B	$270B	-$170B	$89B	-$240B	$65B	-$780B	$430B
2008	-$280B	$220B	-$110B	$72B	-$110B	$50B	-$510B	$340B
2009	-$320B	$240B	-$150B	$86B	-$150B	$54B	-$610B	$380B
2010	-$370B	$270B	-$160B	$93B	-$190B	$65B	-$720B	$430B
2011	-$380B	$270B	-$180B	$99B	-$180B	$82B	-$740B	$450B
2012	-$390B	$260B	-$180B	$100B	-$150B	$85B	-$720B	$450B
2013	-$420B	$280B	-$170B	$100B	-$130B	$83B	-$730B	$460B
2014	-$430B	$270B	-$140B	$93B	-$66B	$73B	-$630B	$440B
2015	-$420B	$270B	-$130B	$90B	-$78B	$71B	-$630B	$430B
2016	-$430B	$280B	-$130B	$110B	-$72B	$59B	-$640B	$450B

You check the Combine row totals option, if there are multiple measures and at least one dimension in the Column shelf.

In this example, the first visual shows uncombined totals for the Import and Export columns, while the second one combines them to display the actual trade deficit.

	Country							
	European Union		NAFTA with Canada (Consump)		OPEC		Total Imports	Total Exports
Year	Imports	Exports	Imports	Exports	Imports	Exports		
1999	-$230B	$170B	-$120B	$58B	-$67B	$19B	-$420B	$250B
2000	-$230B	$160B	-$110B	$60B	-$60B	$20B	-$400B	$240B
2001	-$230B	$150B	-$120B	$61B	-$53B	$19B	-$400B	$230B
2002	-$250B	$160B	-$120B	$63B	-$68B	$17B	-$440B	$240B
2003	-$280B	$170B	-$130B	$67B	-$92B	$22B	-$510B	$260B
2004	-$310B	$190B	-$150B	$76B	-$130B	$32B	-$580B	$290B
2005	-$330B	$210B	-$160B	$83B	-$150B	$39B	-$640B	$330B
2006	-$350B	$240B	-$160B	$87B	-$170B	$48B	-$680B	$380B
2007	-$370B	$270B	-$170B	$89B	-$240B	$65B	-$780B	$430B
2008	-$280B	$220B	-$110B	$72B	-$110B	$50B	-$510B	$340B
2009	-$320B	$240B	-$150B	$86B	-$150B	$54B	-$610B	$380B
2010	-$370B	$270B	-$160B	$93B	-$190B	$65B	-$720B	$430B
2011	-$380B	$270B	-$180B	$99B	-$180B	$82B	-$740B	$450B
2012	-$390B	$260B	-$180B	$100B	-$150B	$85B	-$720B	$450B
2013	-$420B	$280B	-$170B	$100B	-$130B	$83B	-$730B	$460B
2014	-$430B	$270B	-$140B	$93B	-$66B	$73B	-$630B	$440B
2015	-$420B	$270B	-$130B	$90B	-$78B	$71B	-$630B	$430B
2016	-$430B	$280B	-$130B	$110B	-$72B	$59B	-$640B	$450B

	Country						
	European Union		NAFTA with Canada (Consump)		OPEC		Total
Year	Imports	Exports	Imports	Exports	Imports	Exports	
1999	-$230B	$170B	-$120B	$58B	-$67B	$19B	-$170B
2000	-$230B	$160B	-$110B	$60B	-$60B	$20B	-$160B
2001	-$230B	$150B	-$120B	$61B	-$53B	$19B	-$170B
2002	-$250B	$160B	-$120B	$63B	-$68B	$17B	-$210B
2003	-$280B	$170B	-$130B	$67B	-$92B	$22B	-$250B
2004	-$310B	$190B	-$150B	$76B	-$130B	$32B	-$290B
2005	-$330B	$210B	-$160B	$83B	-$150B	$39B	-$310B
2006	-$350B	$240B	-$160B	$87B	-$170B	$48B	-$300B
2007	-$370B	$270B	-$170B	$89B	-$240B	$65B	-$350B
2008	-$280B	$220B	-$110B	$72B	-$110B	$50B	-$160B
2009	-$320B	$240B	-$150B	$86B	-$150B	$54B	-$230B
2010	-$370B	$270B	-$160B	$93B	-$190B	$65B	-$300B
2011	-$380B	$270B	-$180B	$99B	-$180B	$82B	-$290B
2012	-$390B	$260B	-$180B	$100B	-$150B	$85B	-$270B
2013	-$420B	$280B	-$170B	$100B	-$130B	$83B	-$270B
2014	-$430B	$270B	-$140B	$93B	-$66B	$73B	-$200B
2015	-$420B	$270B	-$130B	$90B	-$78B	$71B	-$190B
2016	-$430B	$280B	-$130B	$110B	-$72B	$59B	-$190B

Specifying font weight for totals column in cross tabulation

Cloudera Data Visualization enables you to customize the font weight of the totals column in your cross-tabulation visuals.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.
4. From the Totals Column Font Weight menu, select your desired font weight to apply the changes.

Specifying font weight for totals row in cross tabulation

Cloudera Data Visualization enables you to customize the font weight of the totals row in your cross-tabulation visuals.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.
4. From the Totals Row Font Weight menu, select your desired font weight to apply the changes.

Changing cross tabulation header alignment

In Cloudera Data Visualization, you can customize the alignment of the header in cross tabulation visuals.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.

4. In the Crosstab Header Alignment section, select one of the alignment options.

   Default

   Integers in the headers are aligned to the right and text is aligned to the left in each column.

   Left

   All content in the headers is aligned to the left in each column.

   Center

   All content in the headers is centered in each column.

   Right

   All content in the headers is aligned to the right in each column.

Changing cross tabulation content alignment

In Cloudera Data Visualization, you can customize the alignment of cell content in cross tabulation visuals.

Procedure

1. Open the Dashboard Designer.
2. Click Settings in the VISUAL section of the right-side menu bar.
3. In the VISUAL SETTINGS menu, click Table Style.
4. In the Crosstab Content Alignment section, select one of the alignment options.

   Default

   Integers in the headers are aligned to the right and text is aligned to the left in each column.

   Left

   All content in the headers is aligned to the left in each column.

   Center

   All content in the headers is centered in each column.

   Right

   All content in the headers is aligned to the right in each column.

Customizing table and cross tabulation colors

In Cloudera Data Visualization, you can redesign the colors that appear in a table or a cross tabulation visual.

Changing table header background color

About this task

You can change the background color of a table header.

Procedure

1. On the right side of Visual Designer, click the Settings menu.

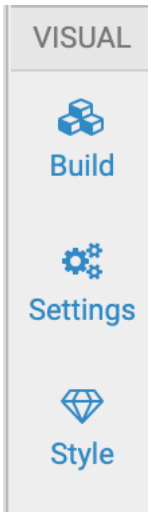

2. In the Settings menu, click Table Colors.

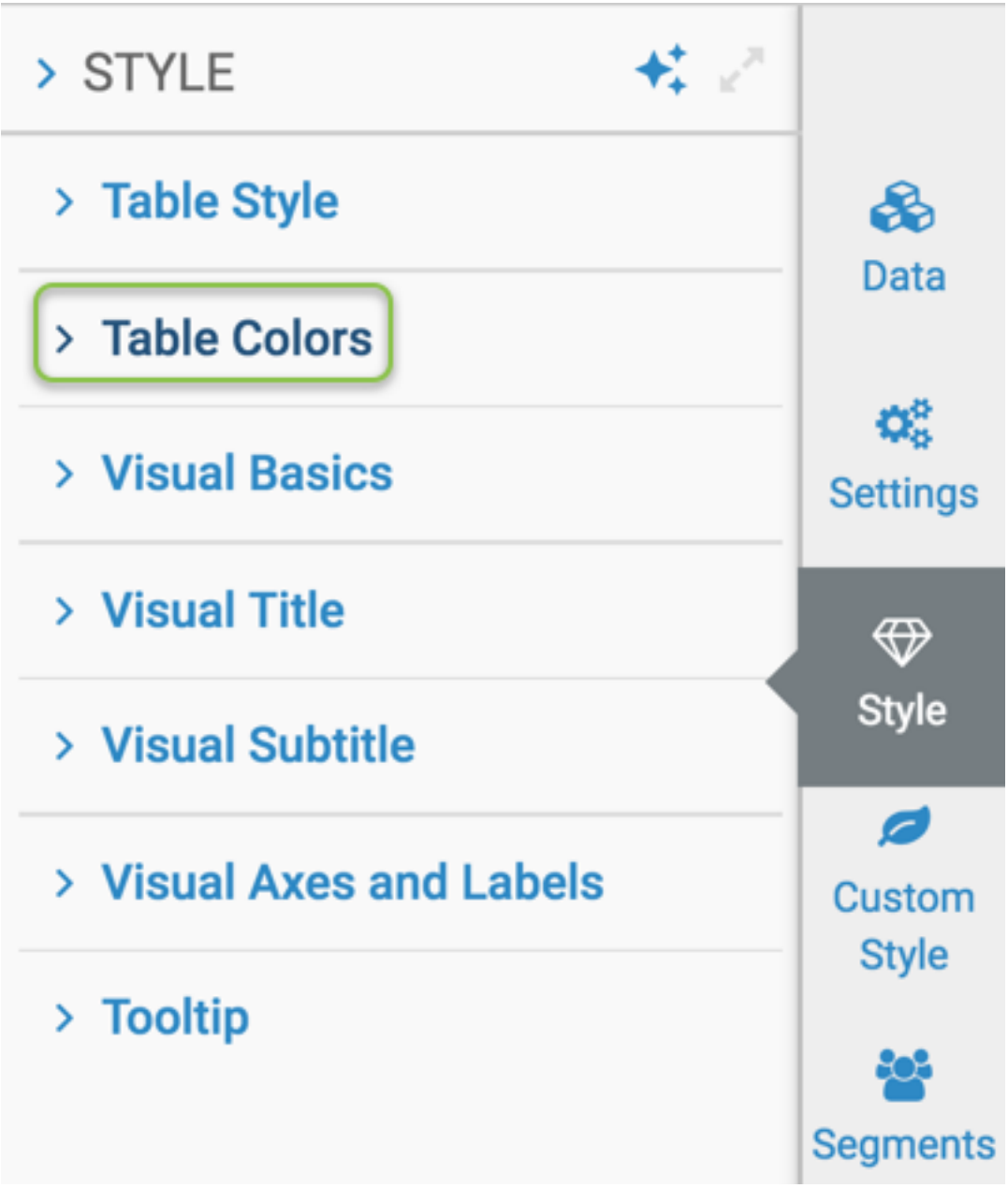

3. To change the backround color of the header in a table visual, make changes in the Header Background selector. You can select one of the palette colors, or specify a custom color.

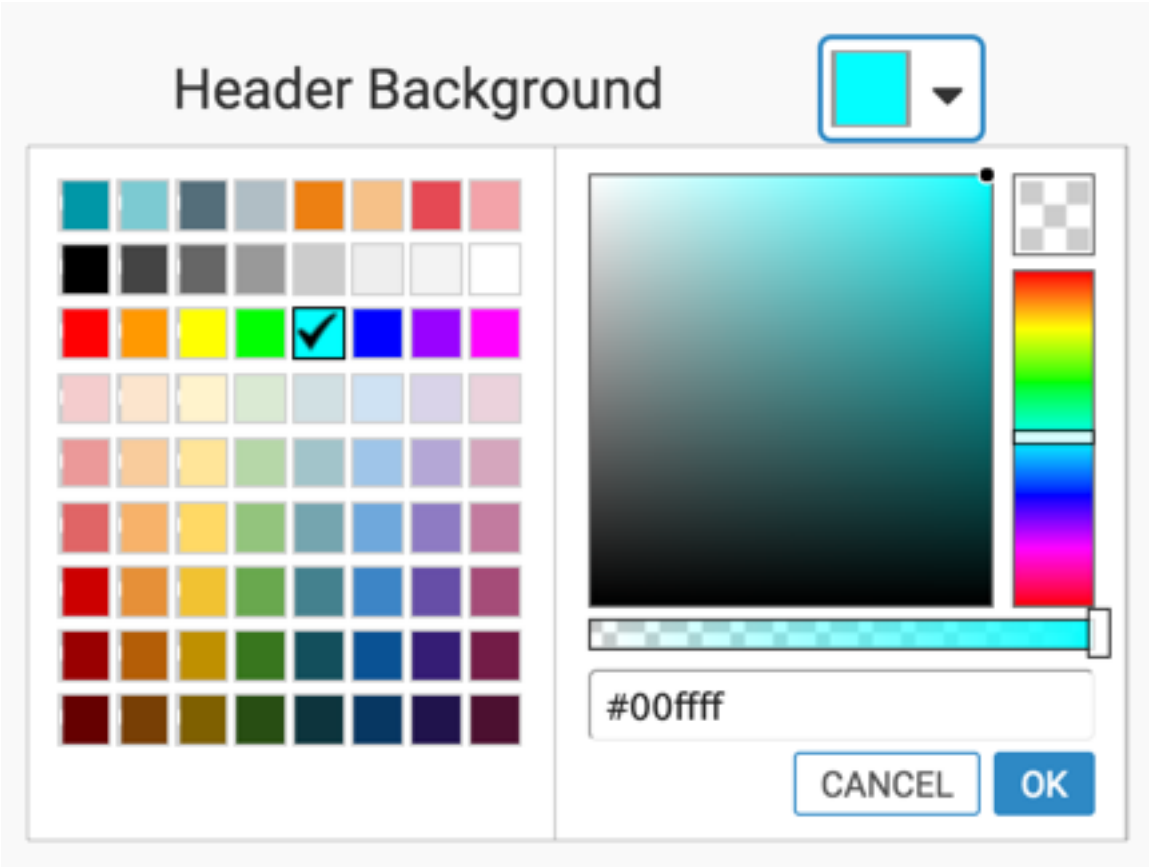

Example

In the following example, notice the difference in the tables before and after changing the background color of the header.

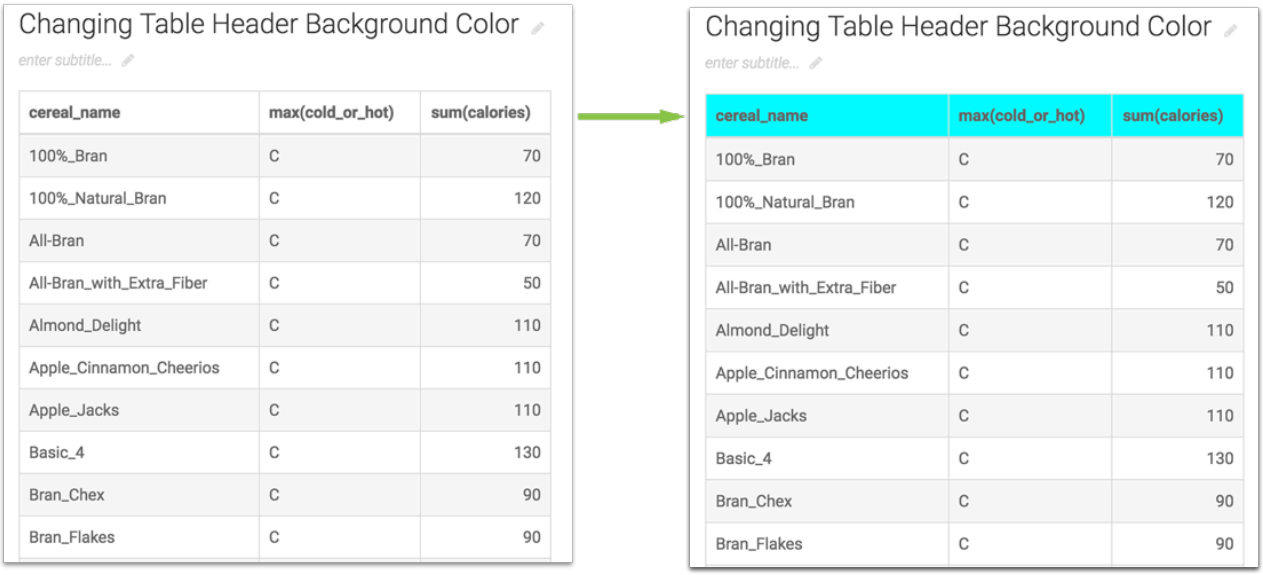

Changing table header text color

About this task

You can change the header text color in table visuals.

Procedure

1. On the right side of Visual Designer, click the Settings menu.

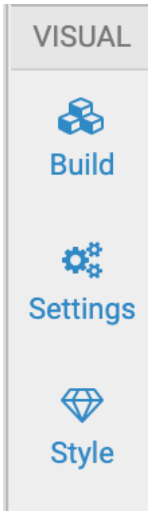

2. In the Settings menu, click Table Colors.

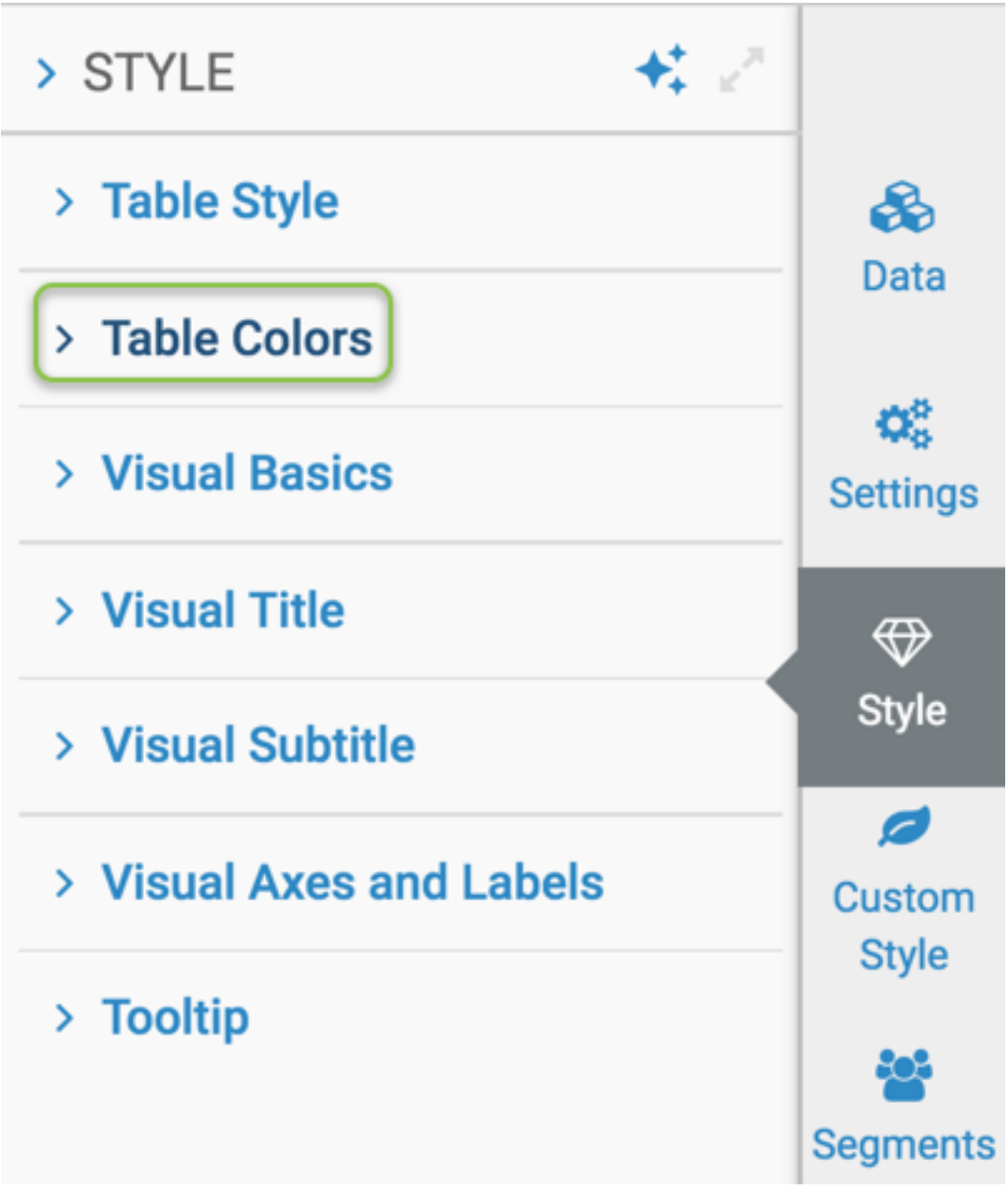

3. To change the color of the header text in a table visual, make changes in the Header Text selector. You can select one of the palette colors, or specify a custom color.

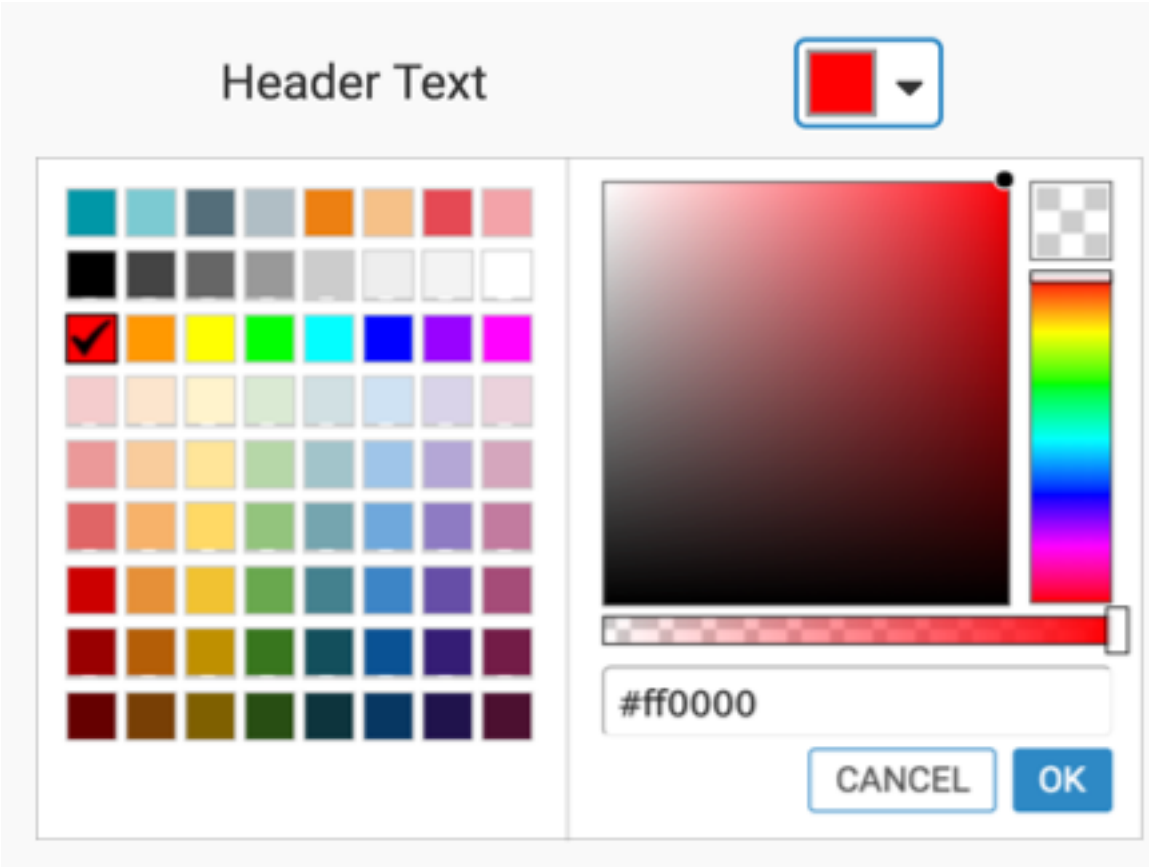

Example

In the following example, notice the difference in the table header before and after changing the text color of the header.

Changing Table Colors

enter subtitle...

cereal_name	max(cold_or_hot)	sum(calories)
100%_Bran	C	70
100%_Natural_Bran	C	120
All-Bran	C	70
All-Bran_with_Extra_Fiber	C	50
Almond_Delight	C	110
Apple_Cinnamon_Cheerios	C	110
Apple_Jacks	C	110
Basic_4	C	130
Bran_Chex	C	90
Bran_Flakes	C	90

Changing Table Colors

enter subtitle...

cereal_name	max(cold_or_hot)	sum(calories)
100%_Bran	C	70
100%_Natural_Bran	C	120
All-Bran	C	70
All-Bran_with_Extra_Fiber	C	50
Almond_Delight	C	110
Apple_Cinnamon_Cheerios	C	110
Apple_Jacks	C	110
Basic_4	C	130
Bran_Chex	C	90
Bran_Flakes	C	90

Changing table odd rows background color

About this task

You can change the background color of the odd rows in a table visuals.

Procedure

1. On the right side of Visual Designer, click the Settings menu.

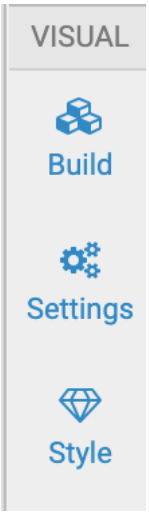

2. In the Settings menu, click Table Colors.

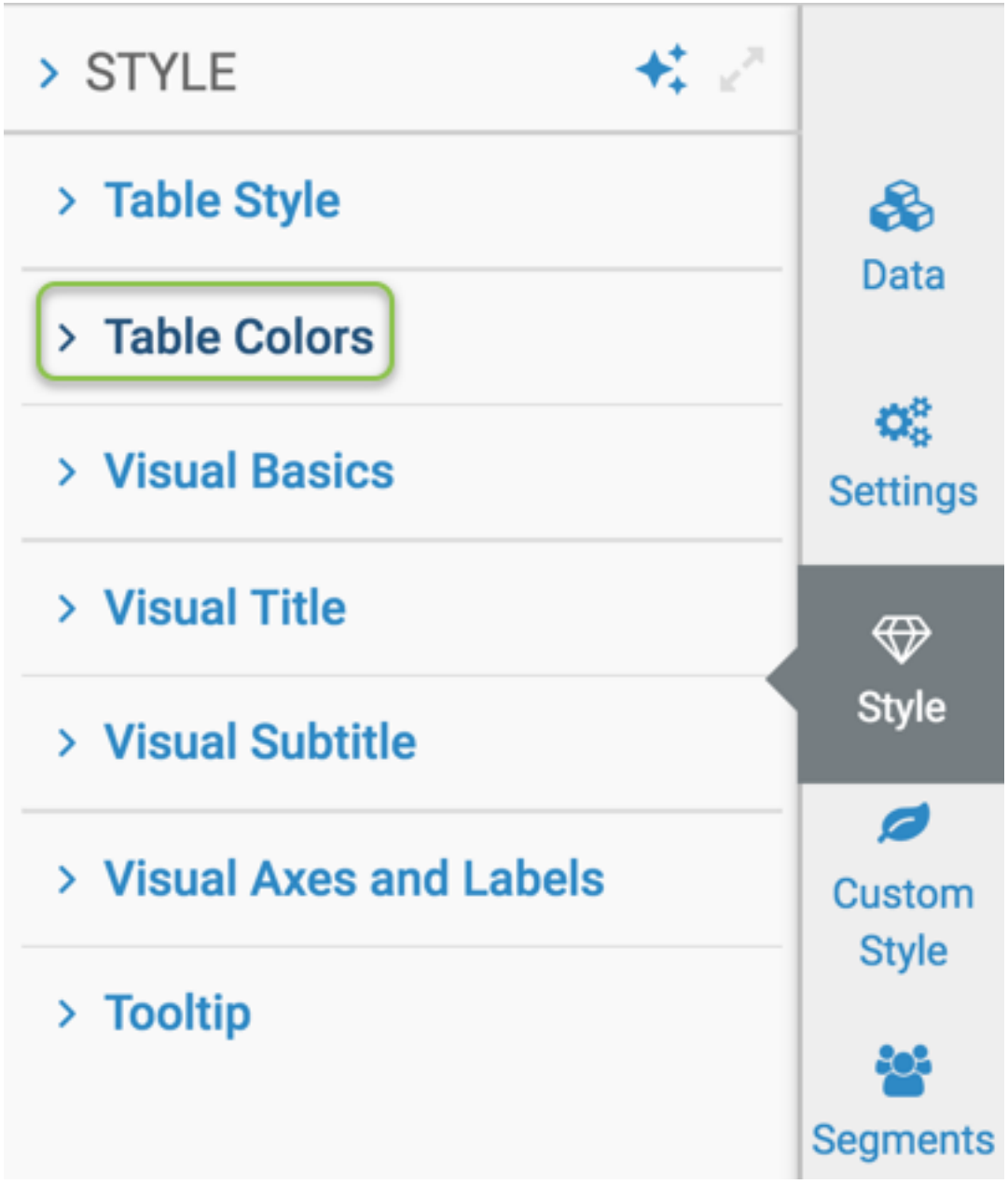

3. To change the background color of the odd rows in a table visual, make changes in the Odd Rows Background selector. You can select one of the palette colors, or specify a custom color.

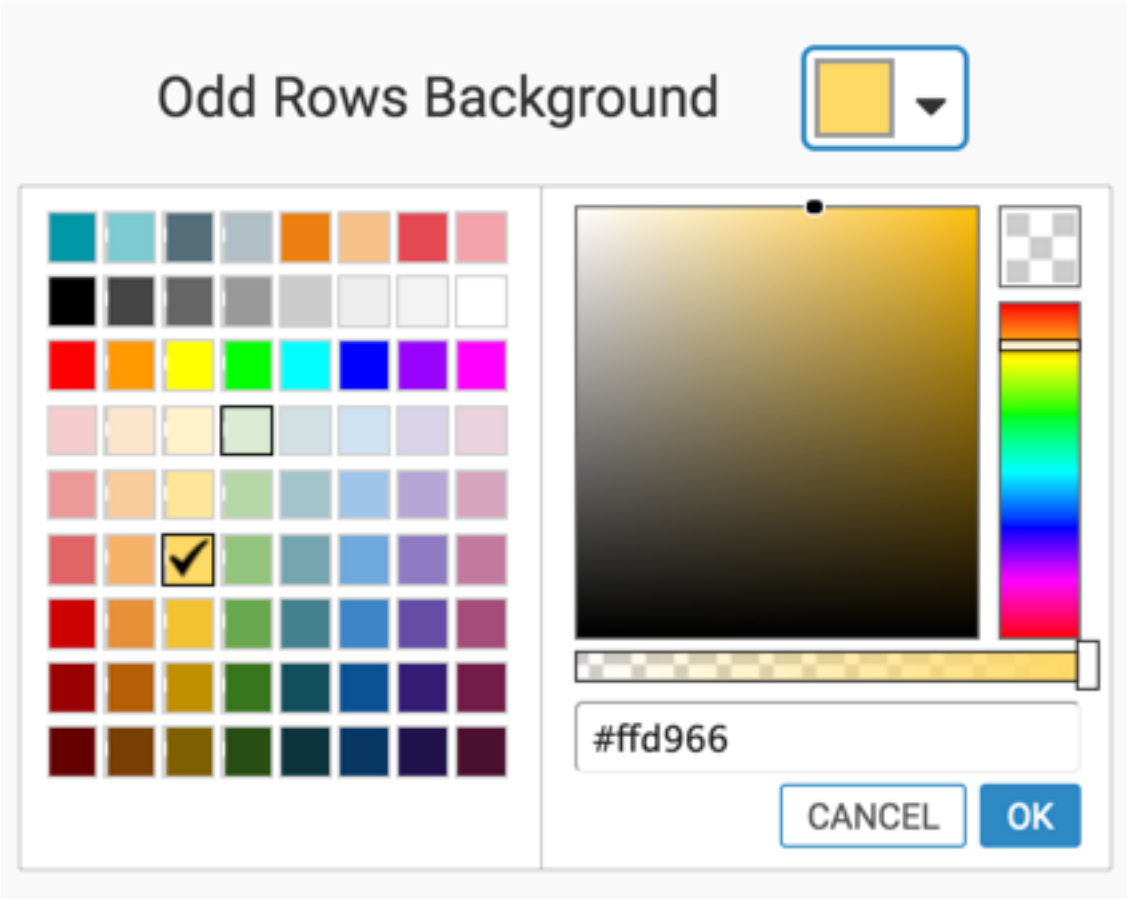

Example

In the following example, notice the difference in the table rows before and after changing the background color of the odd rows.

Changing Table Colors

enter subtitle...

cereal_name	max(cold_or_hot)	sum(calories)
100%_Bran	C	70
100%_Natural_Bran	C	120
All-Bran	C	70
All-Bran_with_Extra_Fiber	C	50
Almond_Delight	C	110
Apple_Cinnamon_Cheerios	C	110
Apple_Jacks	C	110
Basic_4	C	130
Bran_Chex	C	90
Bran_Flakes	C	90
Cap'n'Crunch	C	120

Changing Table Colors

enter subtitle...

cereal_name	max(cold_or_hot)	sum(calories)
100%_Bran	C	70
100%_Natural_Bran	C	120
All-Bran	C	70
All-Bran_with_Extra_Fiber	C	50
Almond_Delight	C	110
Apple_Cinnamon_Cheerios	C	110
Apple_Jacks	C	110
Basic_4	C	130
Bran_Chex	C	90
Bran_Flakes	C	90
Cap'n'Crunch	C	120

Changing table even rows background color

About this task

You can change the background color of the even rows in a table visual.

Procedure

1. On the right side of Visual Designer, click the Settings menu.

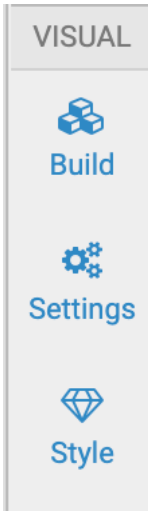

2. In the Settings menu, click Table Colors.

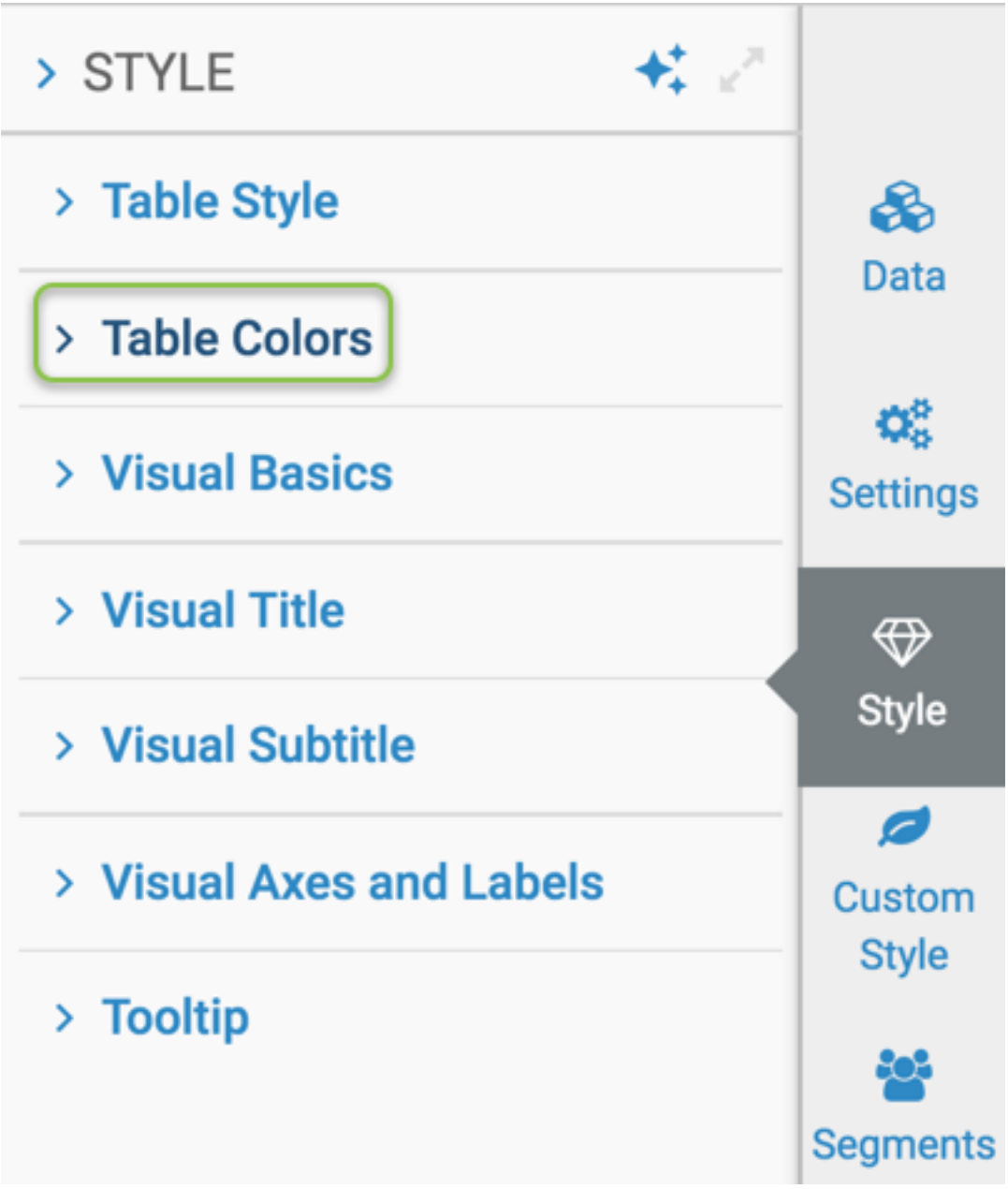

3. To change the background color of the even rows in a table visual, make changes in the Even Rows Background selector. You can select one of the palette colors, or specify a custom color.

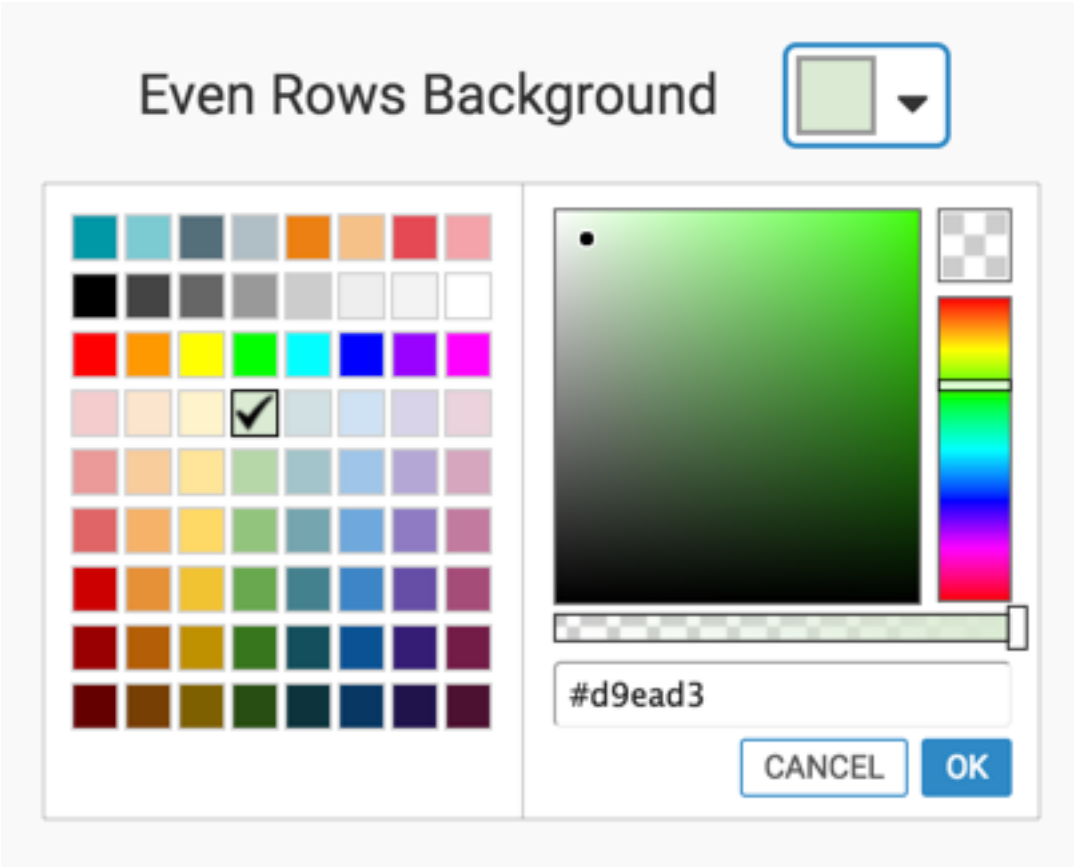

Example

In the following example, notice the difference in the background color of the even rows before and after changing the background color.

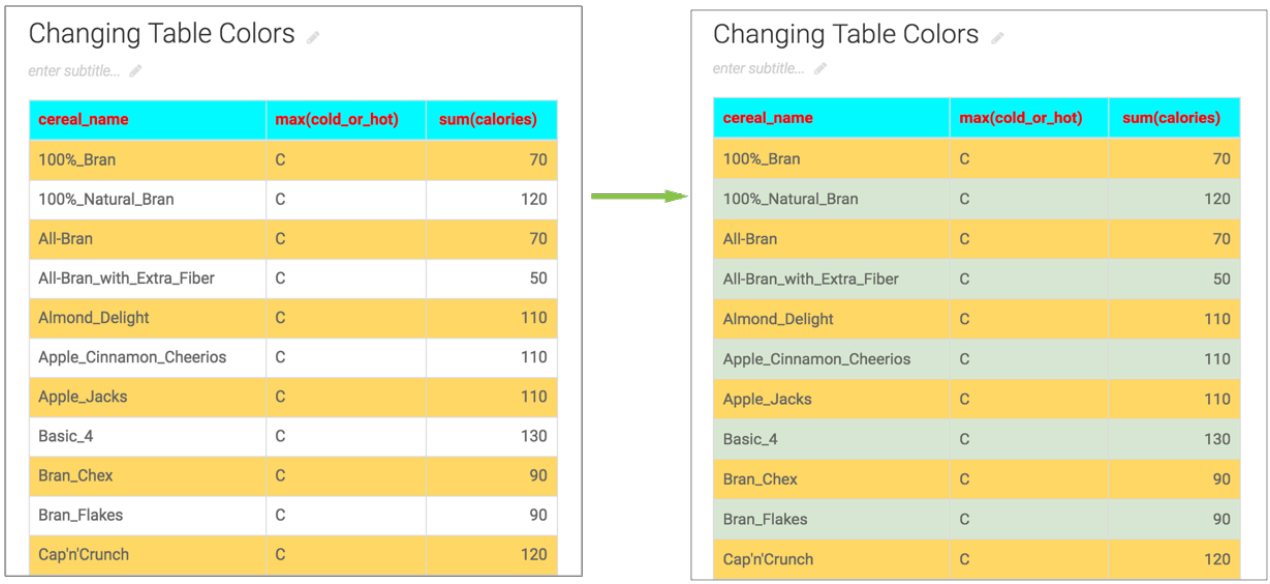

cereal_name	max(cold_or_hot)	sum(calories)
100%_Bran	C	70
100%_Natural_Bran	C	120
All-Bran	C	70
All-Bran_with_Extra_Fiber	C	50
Almond_Delight	C	110
Apple_Cinnamon_Cheerios	C	110
Apple_Jacks	C	110
Basic_4	C	130
Bran_Chex	C	90
Bran_Flakes	C	90
Cap'n'Crunch	C	120

Changing table odd rows text color

About this task

You can change the text color of the odd rows in a table visual.

Procedure

1. On the right side of Visual Designer, click the Settings menu.

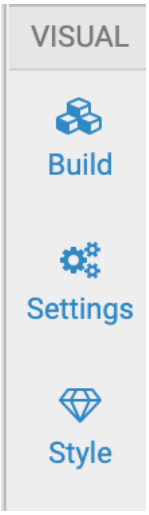

2. In the Settings menu, click Table Colors.

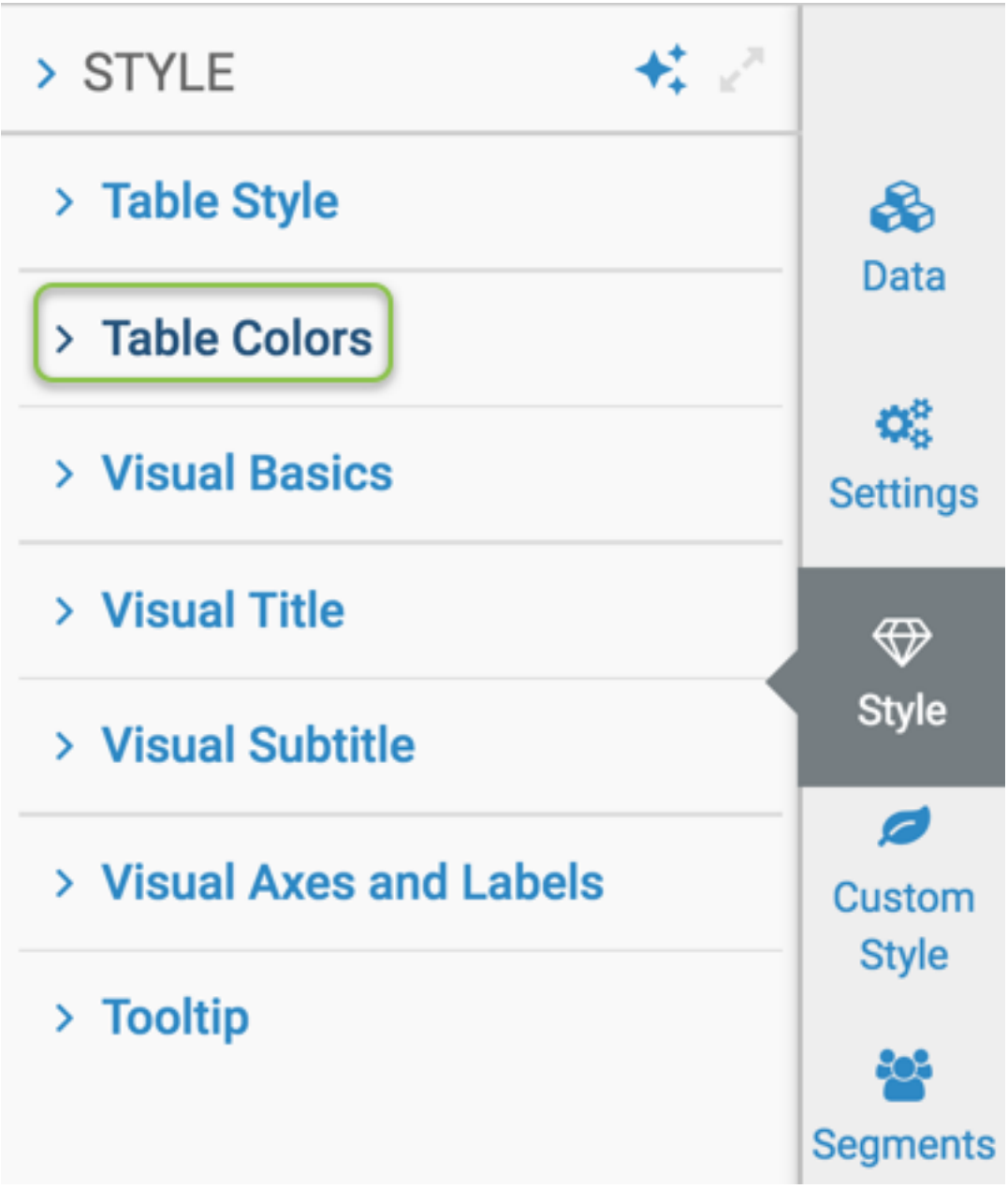

3. To change the text color of the odd rows in a table visual, make changes in the Odd Rows Text selector. You can select one of the palette colors, or specify a custom color.

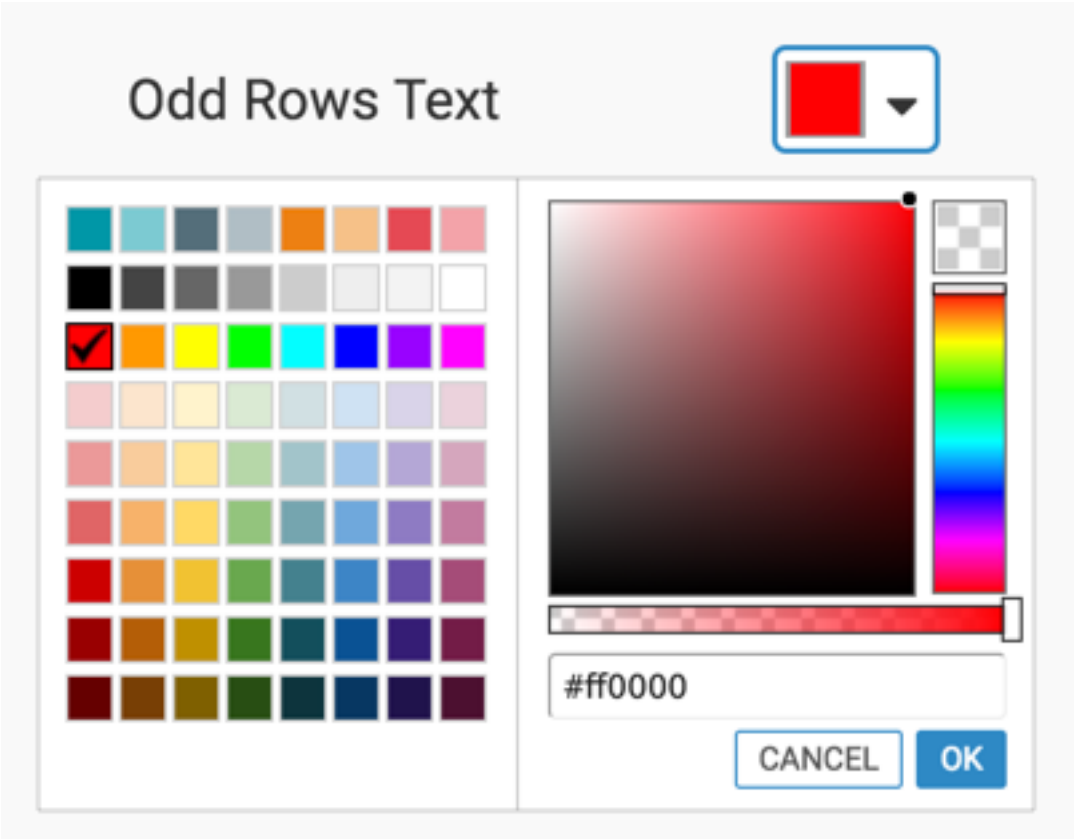

Example

In the following example, notice the difference in the odd rows text color before and after changing the color of the text.

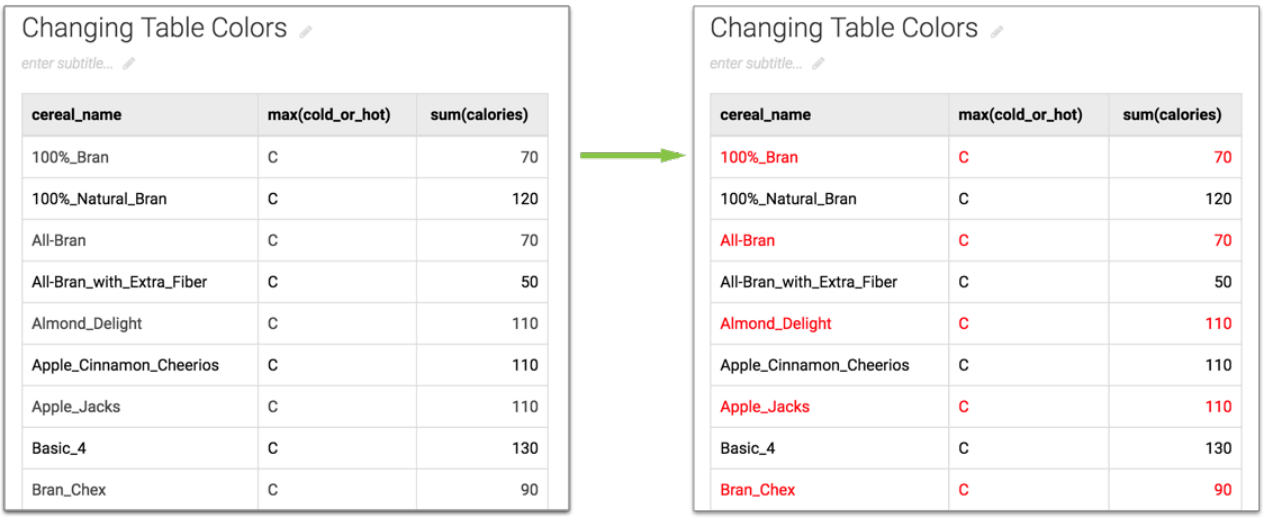

cereal_name	max(cold_or_hot)	sum(calories)
100%_Bran	C	70
100%_Natural_Bran	C	120
All-Bran	C	70
All-Bran_with_Extra_Fiber	C	50
Almond_Delight	C	110
Apple_Cinnamon_Cheerios	C	110
Apple_Jacks	C	110
Basic_4	C	130
Bran_Chex	C	90

cereal_name	max(cold_or_hot)	sum(calories)
100%_Bran	C	70
100%_Natural_Bran	C	120
All-Bran	C	70
All-Bran_with_Extra_Fiber	C	50
Almond_Delight	C	110
Apple_Cinnamon_Cheerios	C	110
Apple_Jacks	C	110
Basic_4	C	130
Bran_Chex	C	90

Changing table even rows text color

About this task

You can change the text color of the even rows in a table visual.

Procedure

1. On the right side of Visual Designer, click the Settings menu.

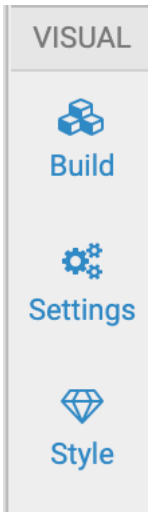

2. In the Settings menu, click Table Colors.

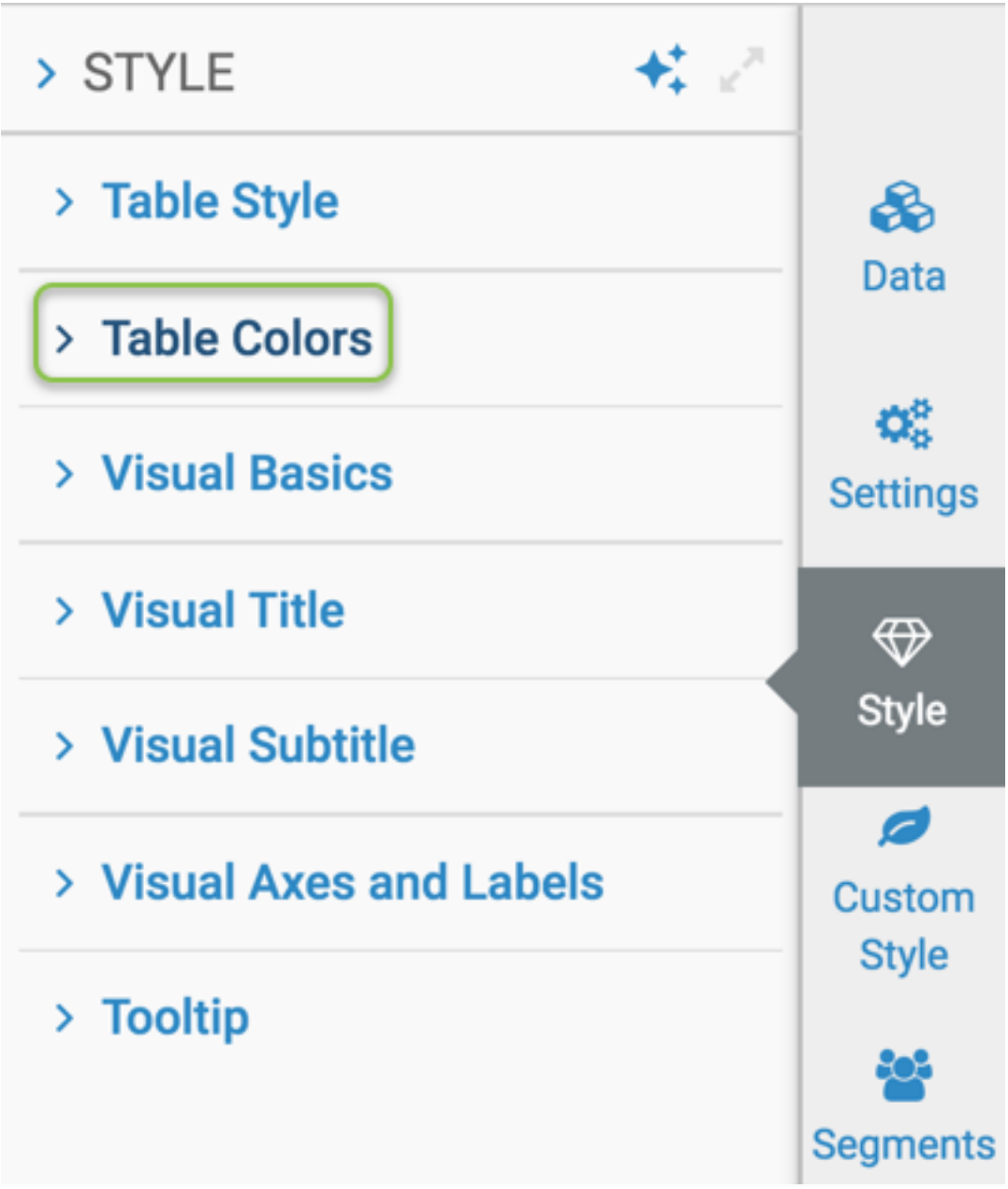

3. To change the text color of the even rows in a table visual, make changes in the Even Rows Background selector. You can select one of the palette colors, or specify a custom color.

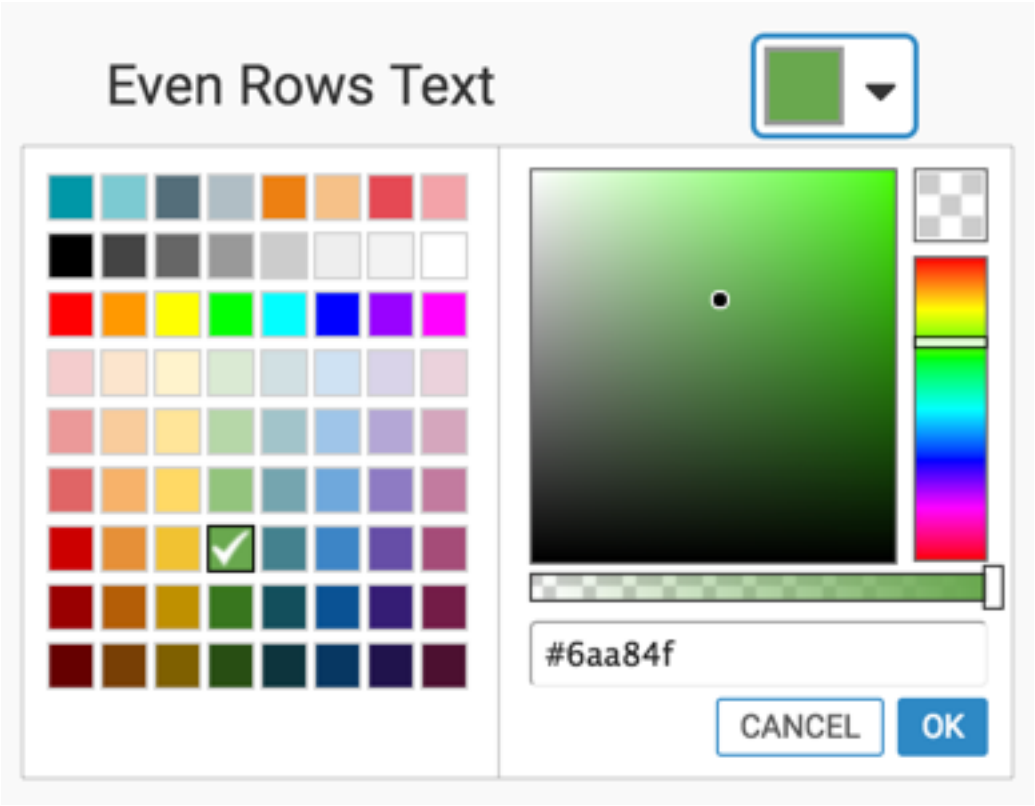

Example

In the following example, notice the difference in the even rows text color before and after changing the color of the text.

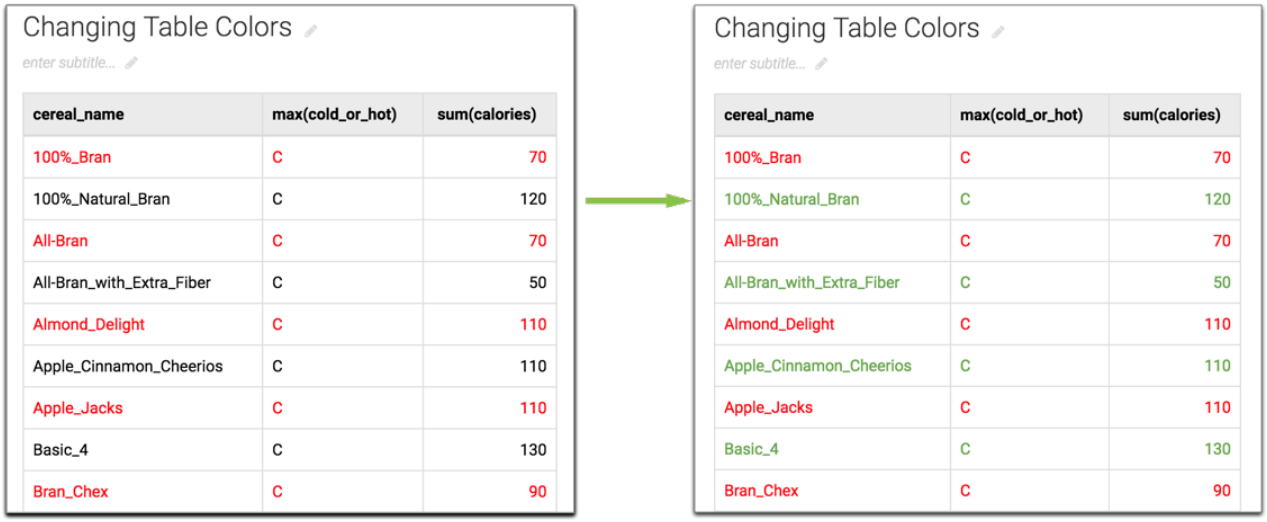

Changing table border color

About this task

You can change the color of the cell borders in a table visual.

Procedure

1. On the right side of Visual Designer, click the Settings menu.

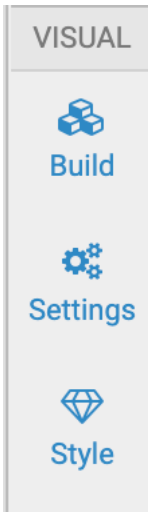

2. In the Settings menu, click Table Colors.

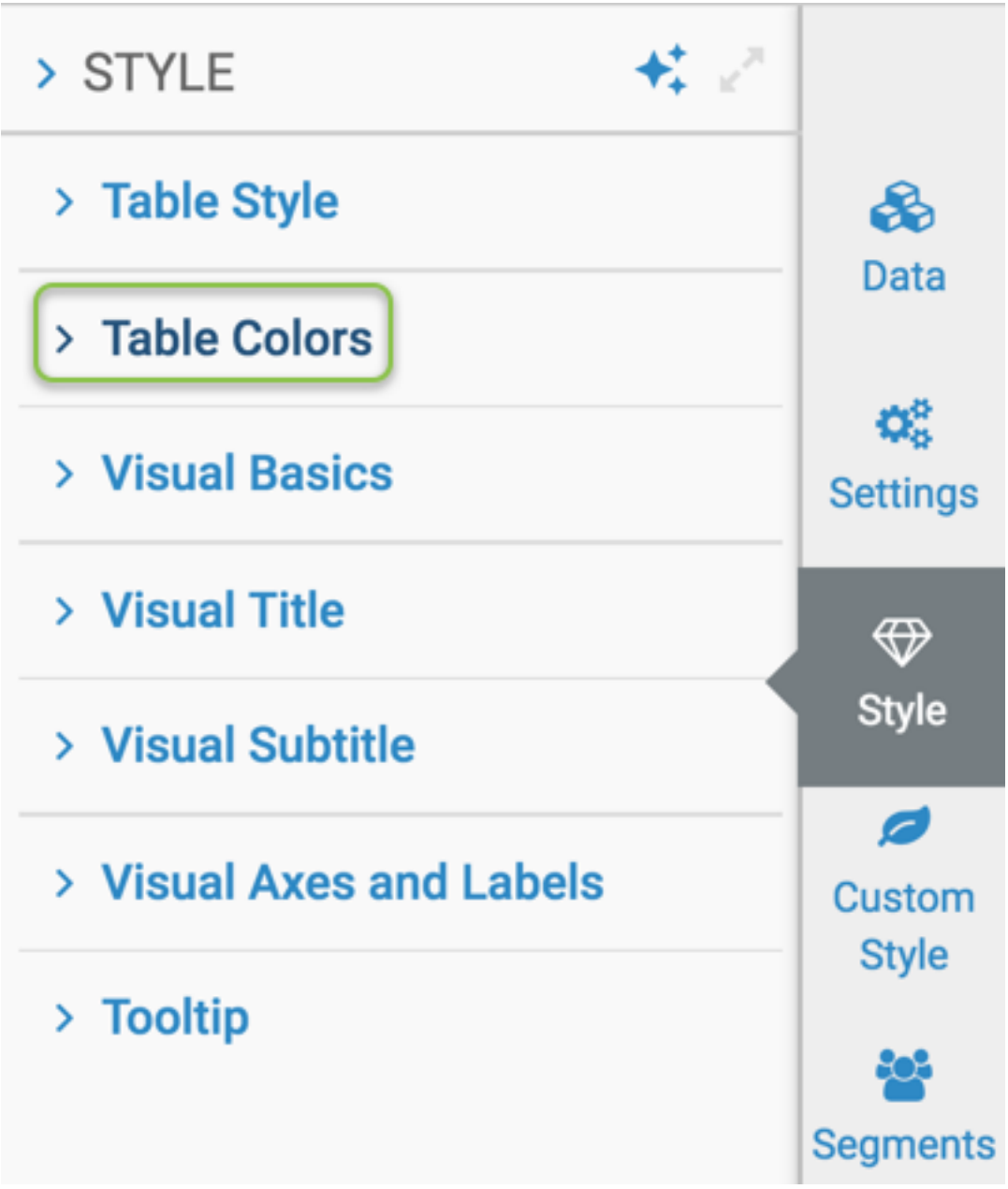

3. To change the color of the cell borders in a table visual, make changes in the Border selector. You can select one of the palette colors, or specify a custom color.

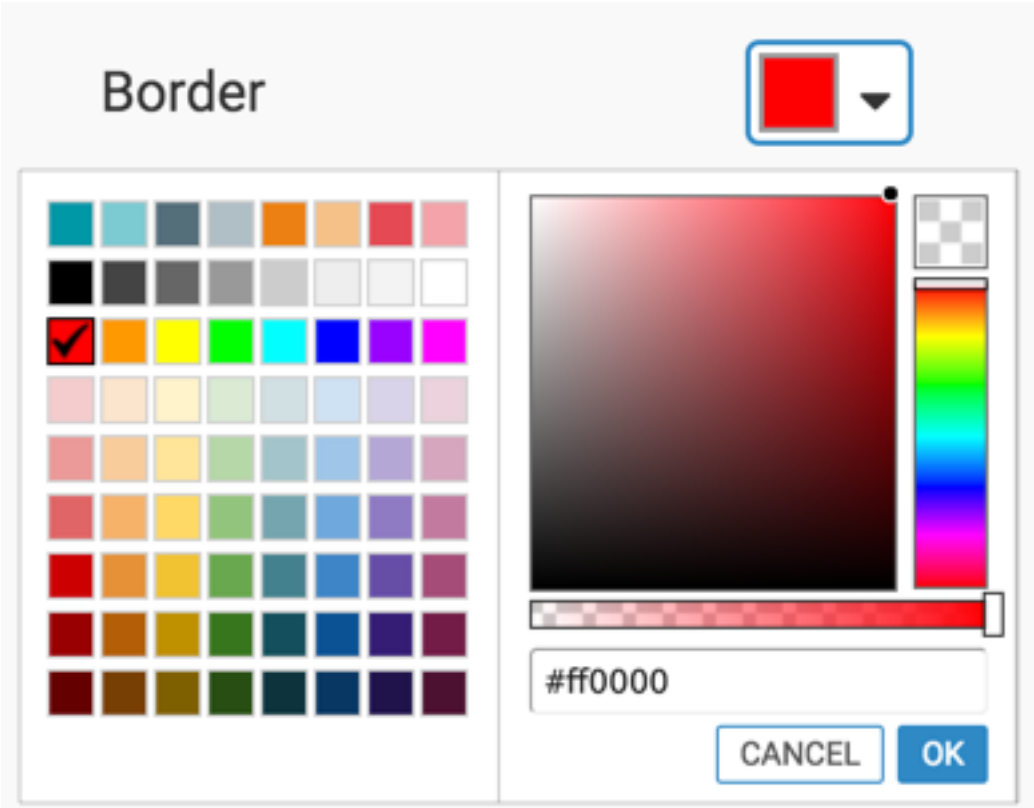

Example

In the following example, notice the difference in the table cell borders before and after changing the border color.

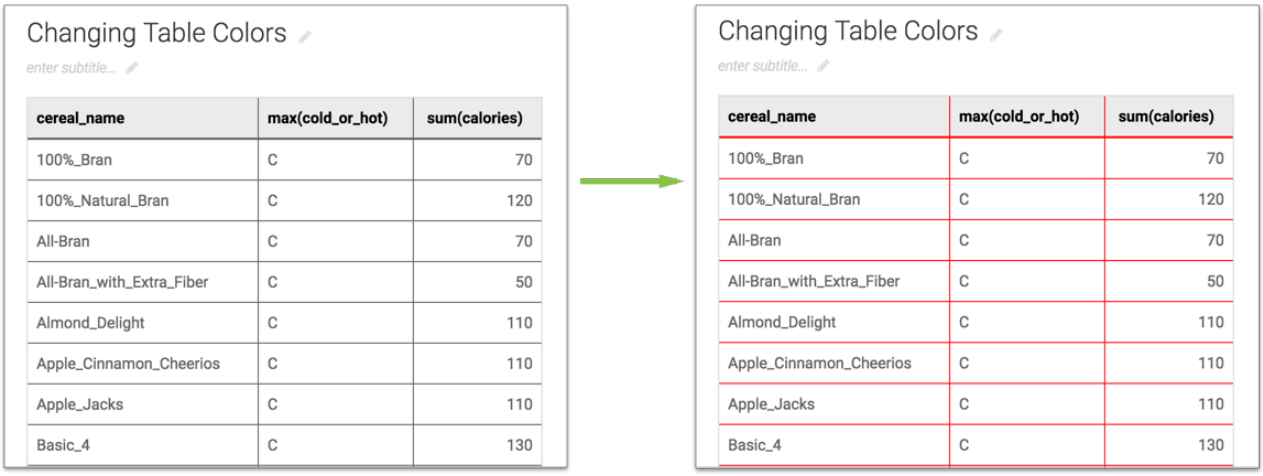

Changing Table Colors

enter subtitle...

cereal_name	max(cold_or_hot)	sum(calories)
100%_Bran	C	70
100%_Natural_Bran	C	120
All-Bran	C	70
All-Bran_with_Extra_Fiber	C	50
Almond_Delight	C	110
Apple_Cinnamon_Cheerios	C	110
Apple_Jacks	C	110
Basic_4	C	130

Changing Table Colors

enter subtitle...

cereal_name	max(cold_or_hot)	sum(calories)
100%_Bran	C	70
100%_Natural_Bran	C	120
All-Bran	C	70
All-Bran_with_Extra_Fiber	C	50
Almond_Delight	C	110
Apple_Cinnamon_Cheerios	C	110
Apple_Jacks	C	110
Basic_4	C	130

Changing totals row background color

About this task

You can change the background color of the totals row in a table visual.

Procedure

1. On the right side of Visual Designer, click the Settings menu.

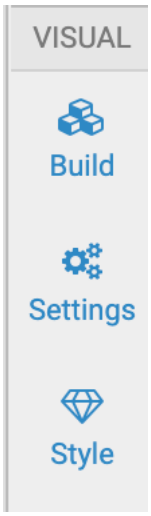

2. In the Settings menu, click Table Colors.

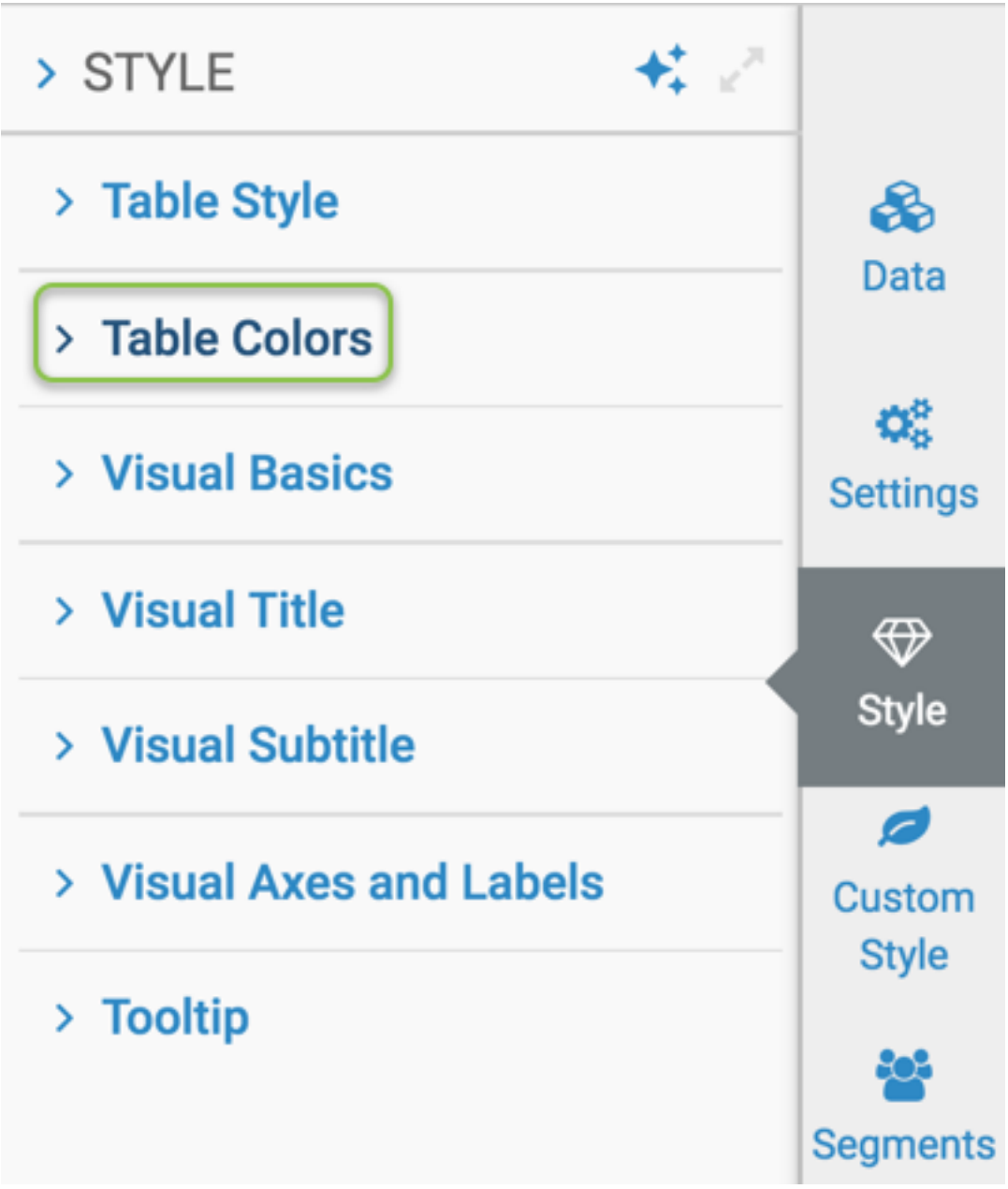

3. To change the background color of the totals rows in a table visual, make changes in the Totals Row Background selector. You can select one of the palette colors, or specify a custom color.

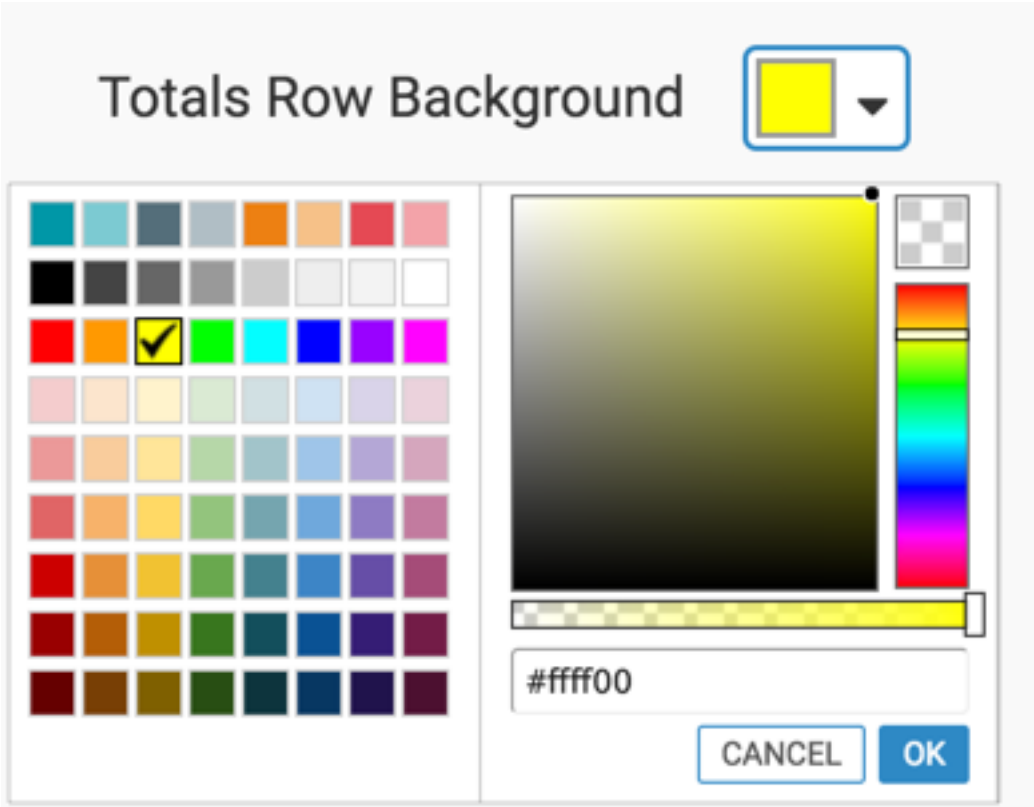

Example

In the following example, notice the difference in the totals row before and after changing the background color of the totals row.

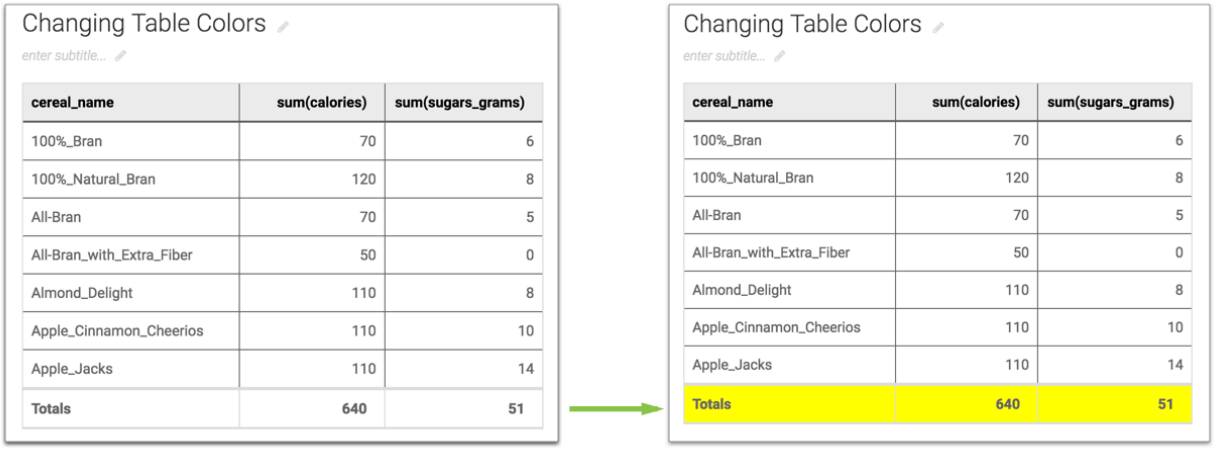

cereal_name	sum(calories)	sum(sugars_grams)
100%_Bran	70	6
100%_Natural_Bran	120	8
All-Bran	70	5
All-Bran_with_Extra_Fiber	50	0
Almond_Delight	110	8
Apple_Cinnamon_Cheerios	110	10
Apple_Jacks	110	14
Totals	**640**	**51**

cereal_name	sum(calories)	sum(sugars_grams)
100%_Bran	70	6
100%_Natural_Bran	120	8
All-Bran	70	5
All-Bran_with_Extra_Fiber	50	0
Almond_Delight	110	8
Apple_Cinnamon_Cheerios	110	10
Apple_Jacks	110	14
Totals	**640**	**51**

Changing totals row text color

About this task

You can change the text color of the totals row in a table visual.

Procedure

1. On the right side of Visual Designer, click the Settings menu.

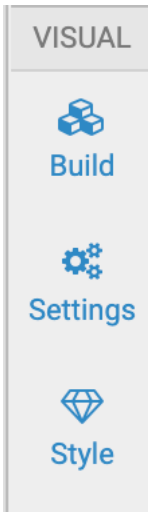

2. In the Settings menu, click Table Colors.

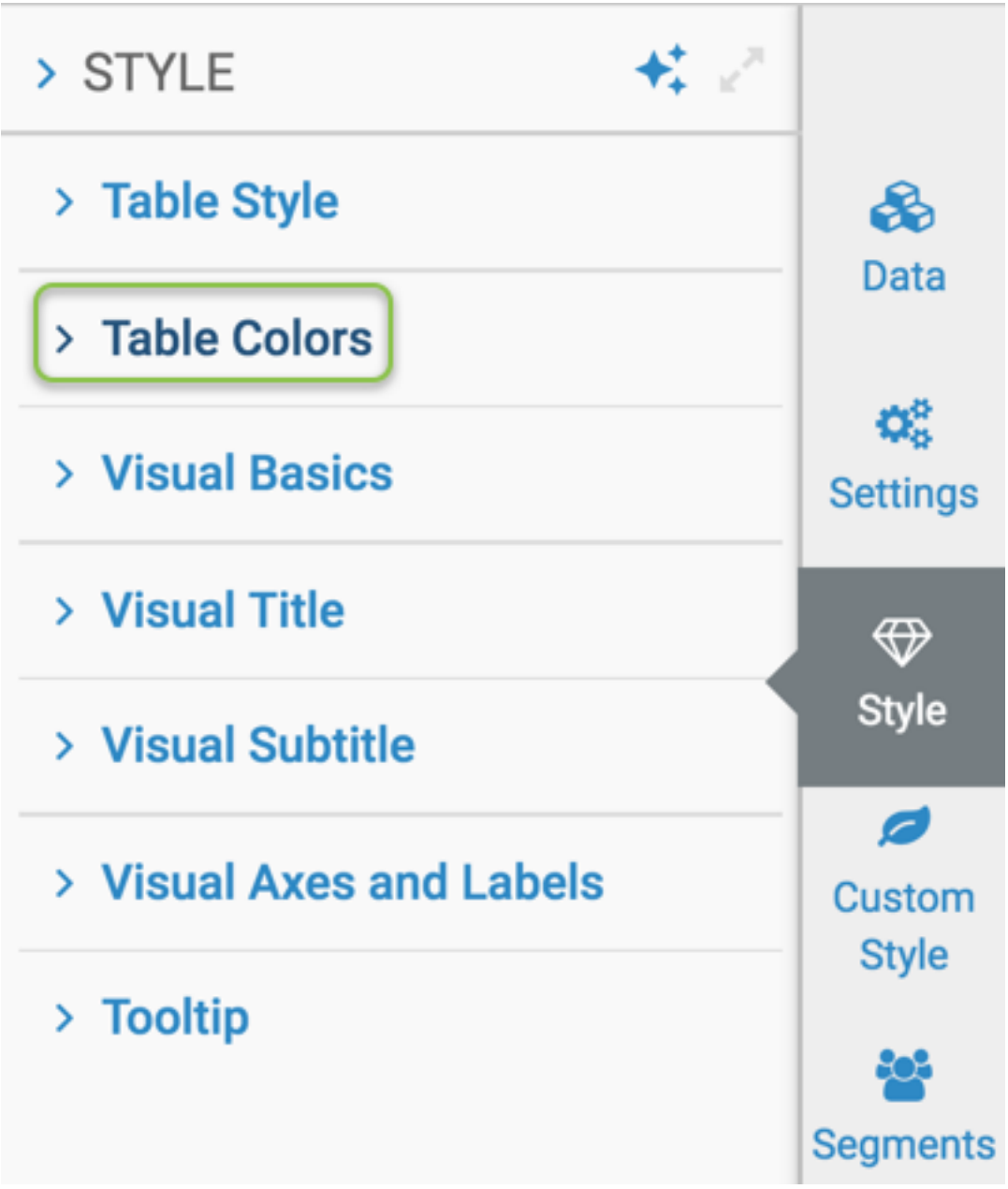

3. To change the text color of the totals row in a table visual, make changes in the Totals Row Text selector. You can select one of the palette colors, or specify a custom color.

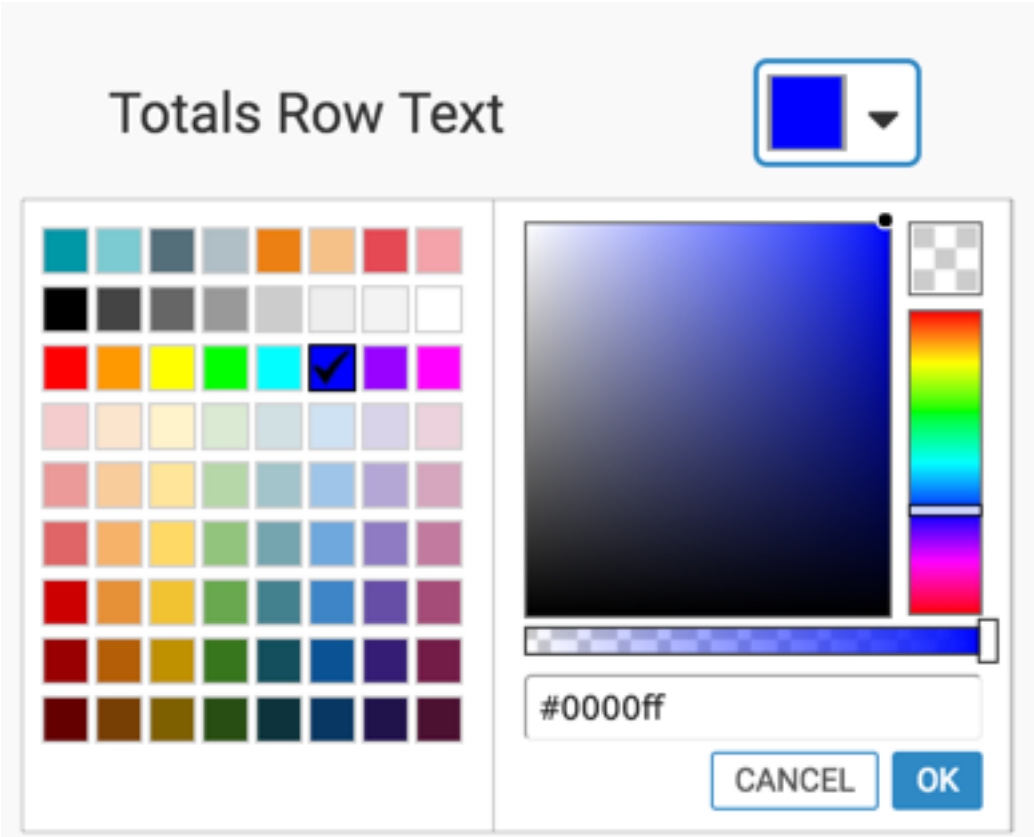

Example

In the following example, notice the difference in the totals row before and after changing the text color of the totals row.

Changing Table Colors

enter subtitle...

cereal_name	sum(calories)	sum(sugars_grams)
100%_Bran	70	6
100%_Natural_Bran	120	8
All-Bran	70	5
All-Bran_with_Extra_Fiber	50	0
Almond_Delight	110	8
Apple_Cinnamon_Cheerios	110	10
Apple_Jacks	110	14
Totals	**640**	**51**

Changing Table Colors

enter subtitle...

cereal_name	sum(calories)	sum(sugars_grams)
100%_Bran	70	6
100%_Natural_Bran	120	8
All-Bran	70	5
All-Bran_with_Extra_Fiber	50	0
Almond_Delight	110	8
Apple_Cinnamon_Cheerios	110	10
Apple_Jacks	110	14
Totals	**640**	**51**

Customizing visual title

In Cloudera Data Visualization, you can manage modify the display options of visual titles.

Changing font family in the title of a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

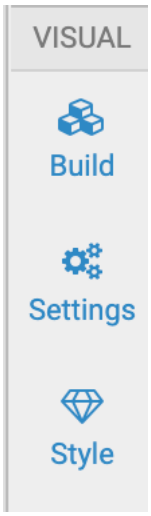

2. In the Settings menu, click Visual Title.

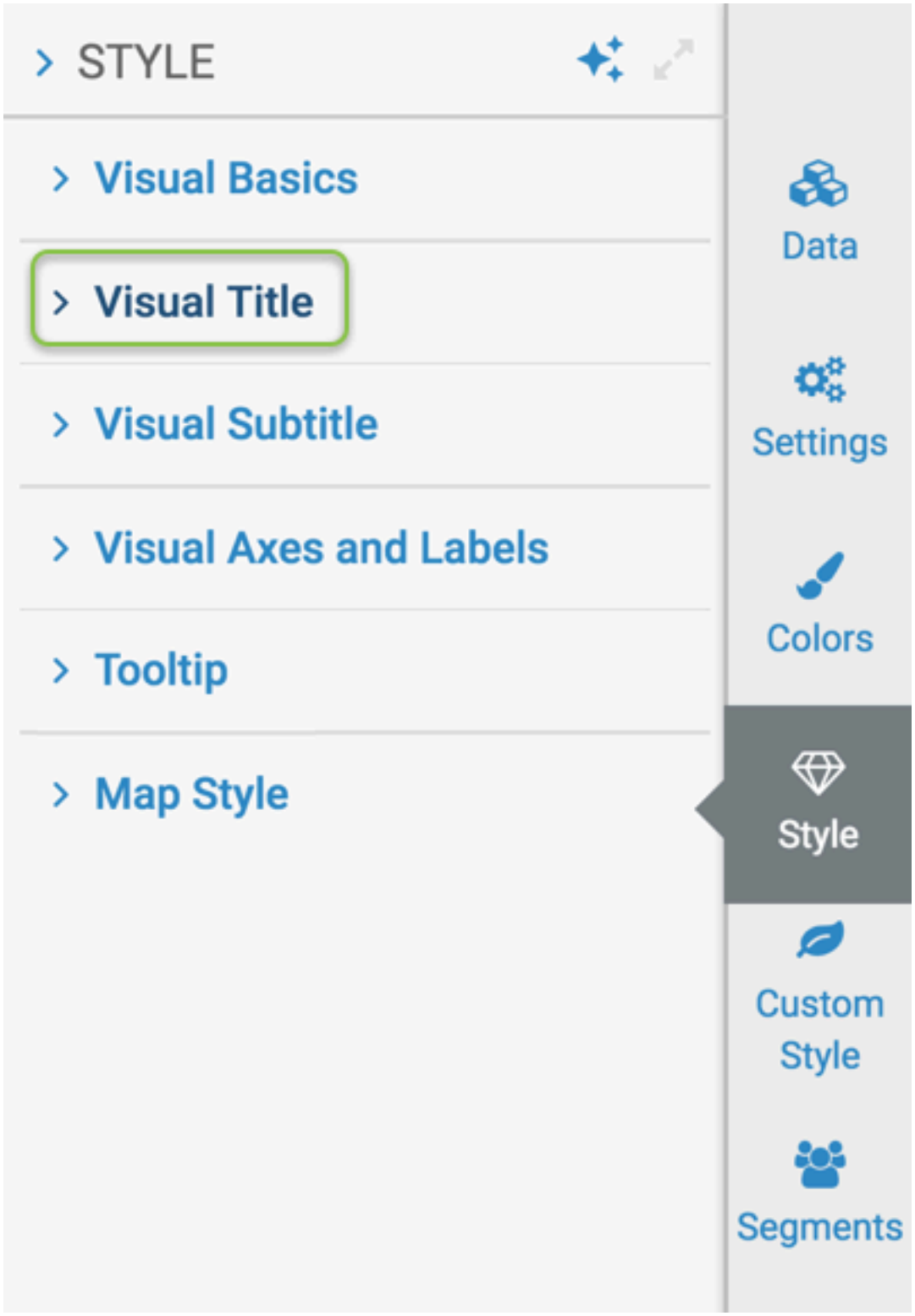

3. To change the font used by the title of a visual, navigate to the Visual Titlemenu, and make a selection from the Font Family menu.

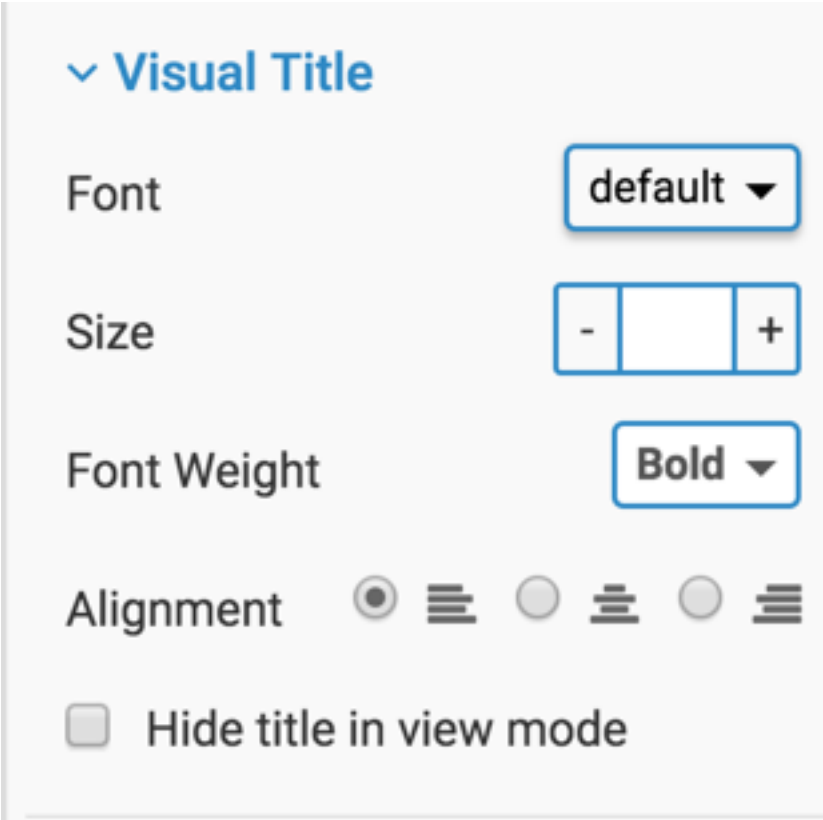

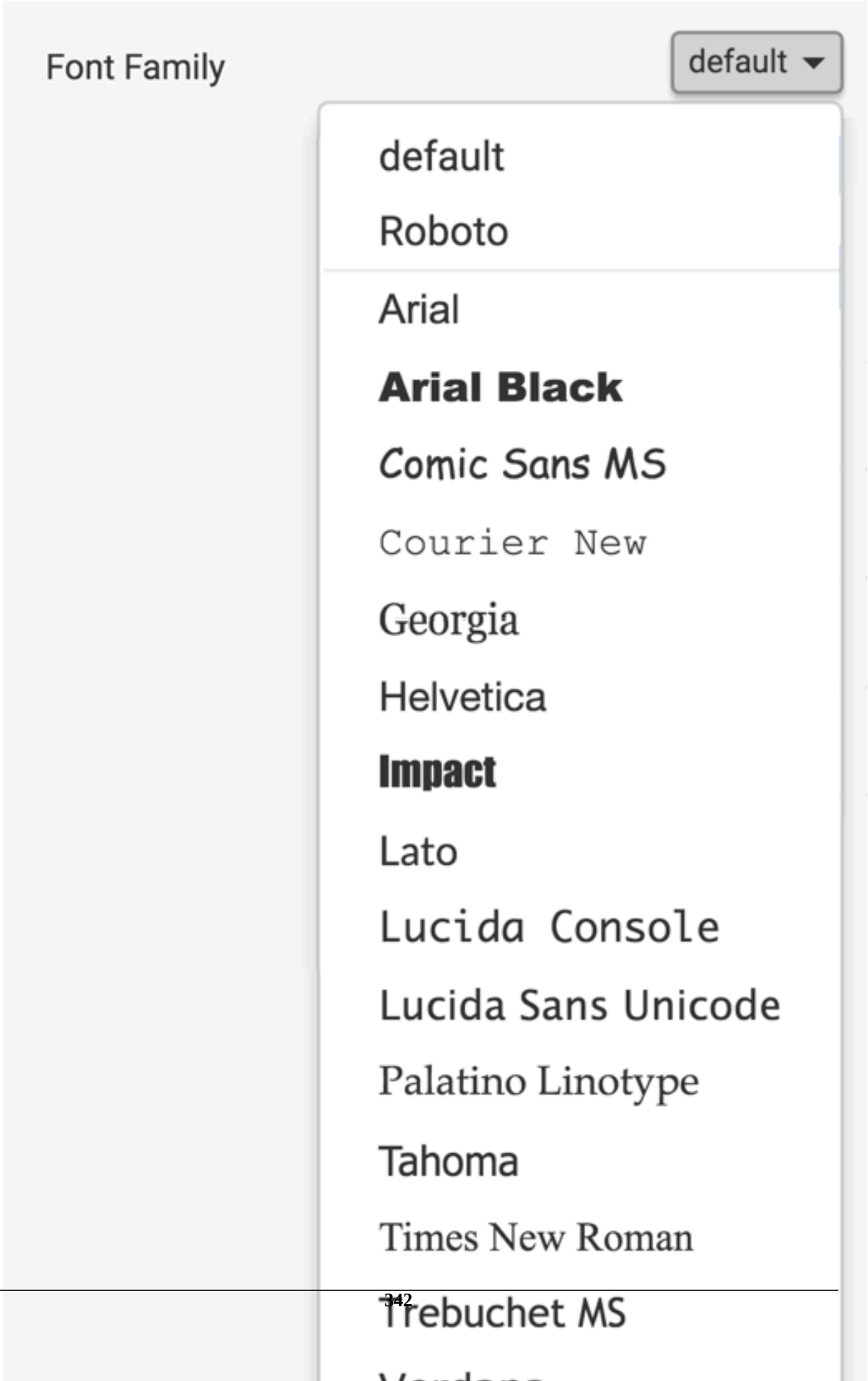
Font Family
default
default
Roboto
Arial
Arial Black
Comic Sans MS
Courier New
Georgia
Helvetica
Impact
Lato
Lucida Console
Lucida Sans Unicode
Palatino Linotype
Tahoma
Times New Roman
Trebuchet MS

Changing font size in the title of a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

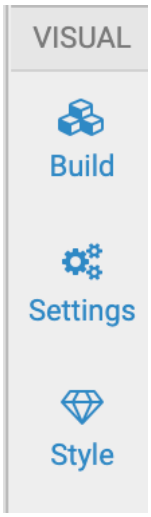

2. In the Settings menu, click Visual Title.

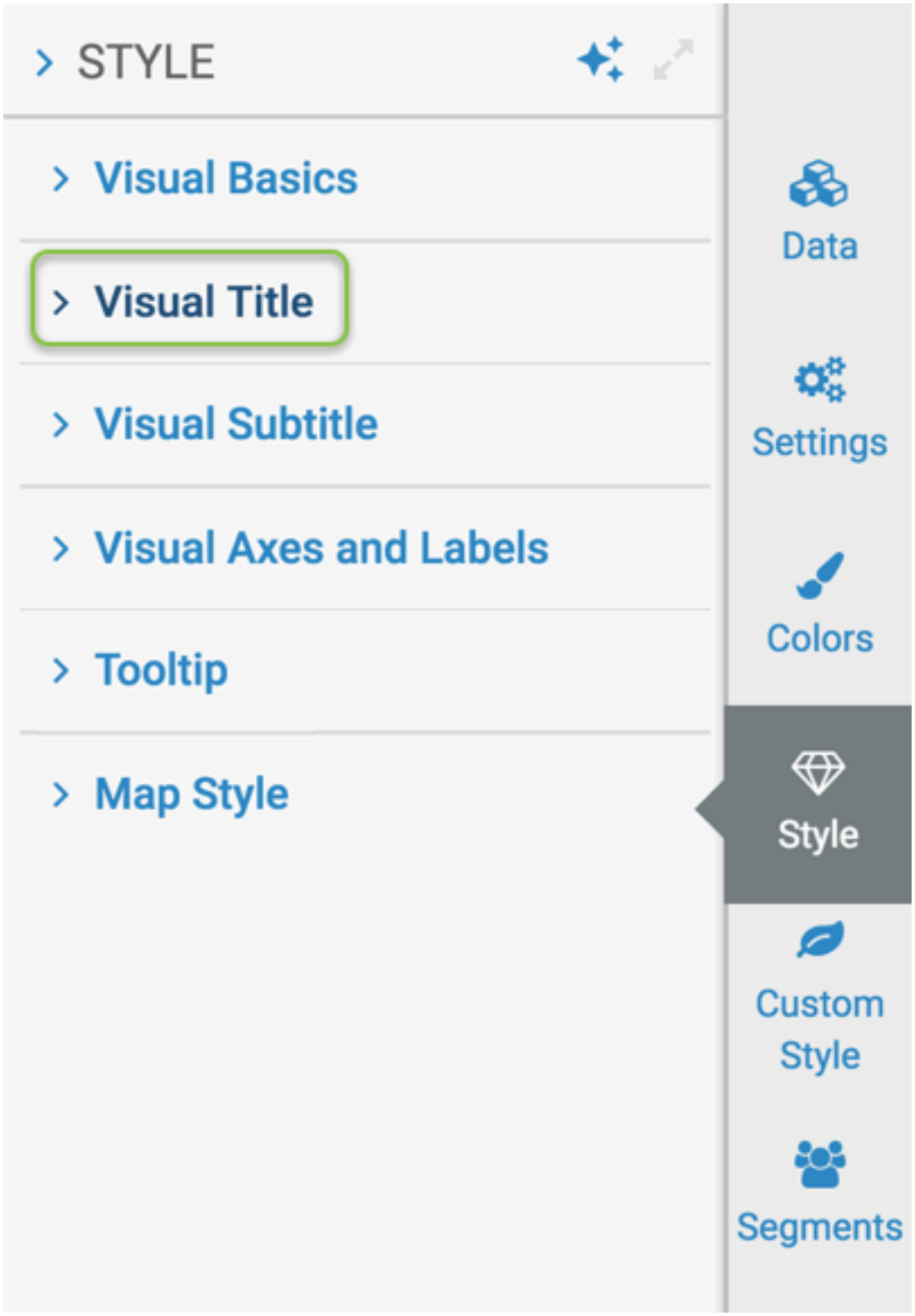

3. To change the font size used by the title of a visual, navigate to the Visual Title menu, and make adjustments in the Font Size selector.

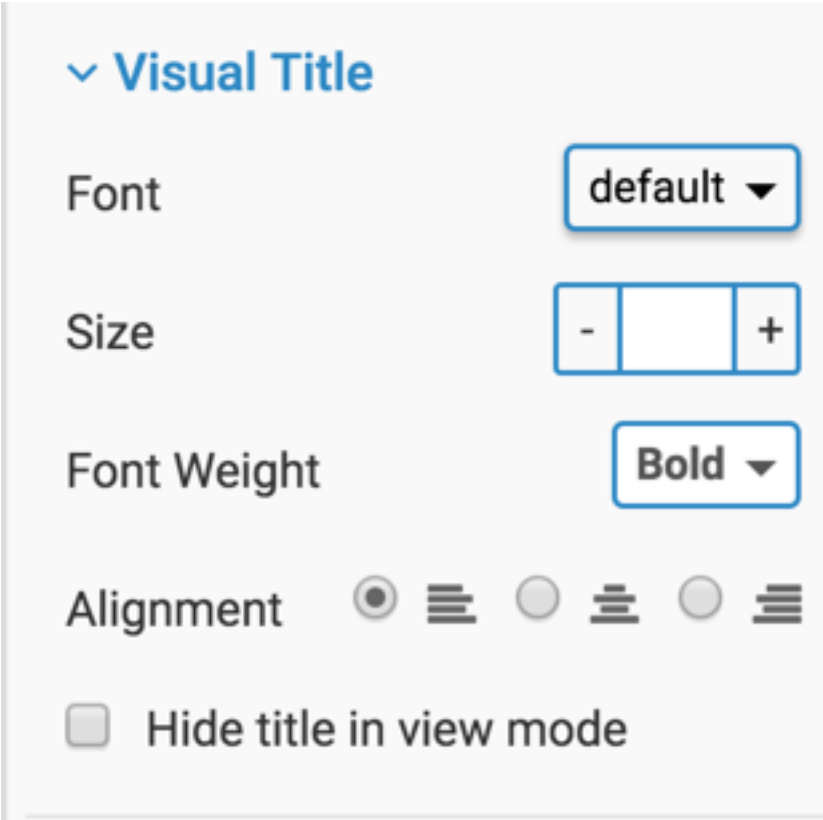

Changing font weight in the title of a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

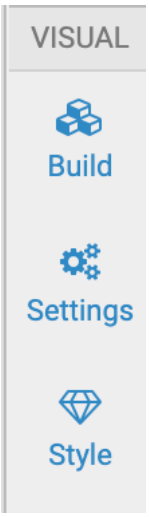

2. In the Settings menu, click Visual Title.

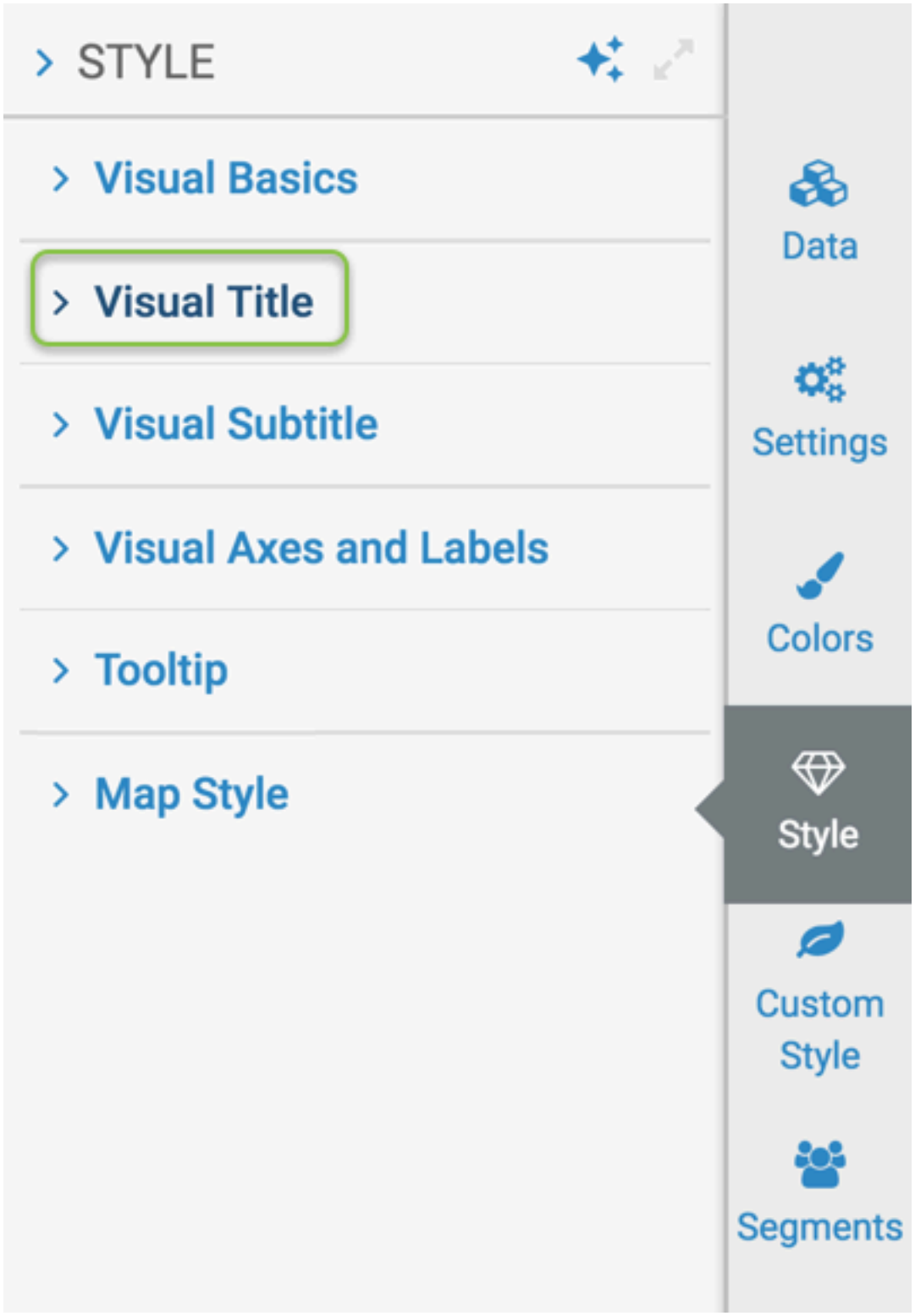

3. To change the font weight used by the title of a visual, navigate to the Visual Title menu, open the Font Weight menu, and make a selection.

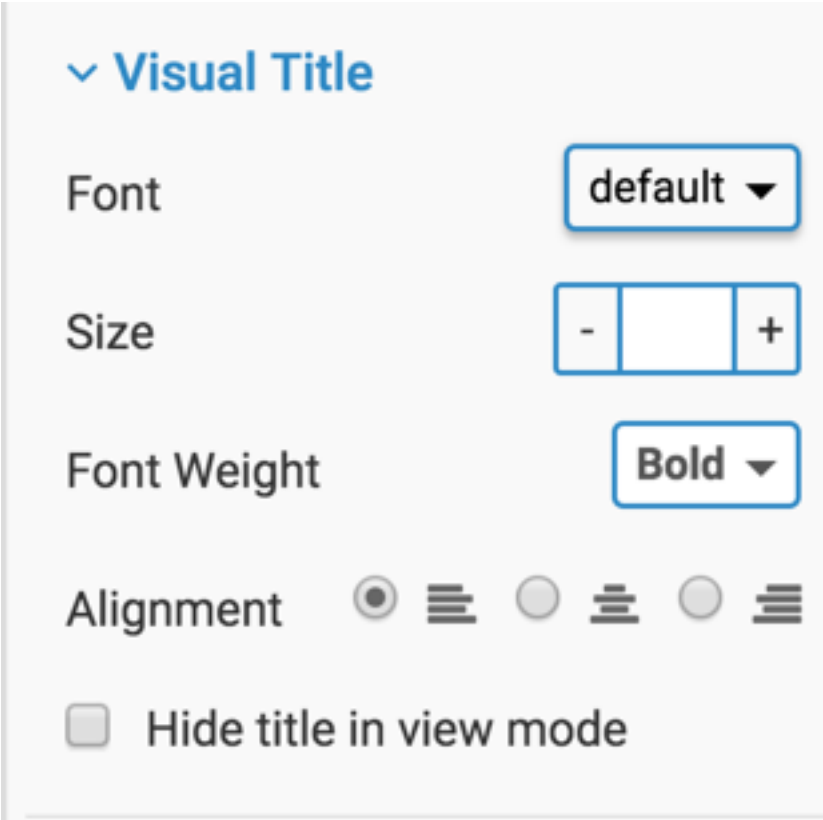

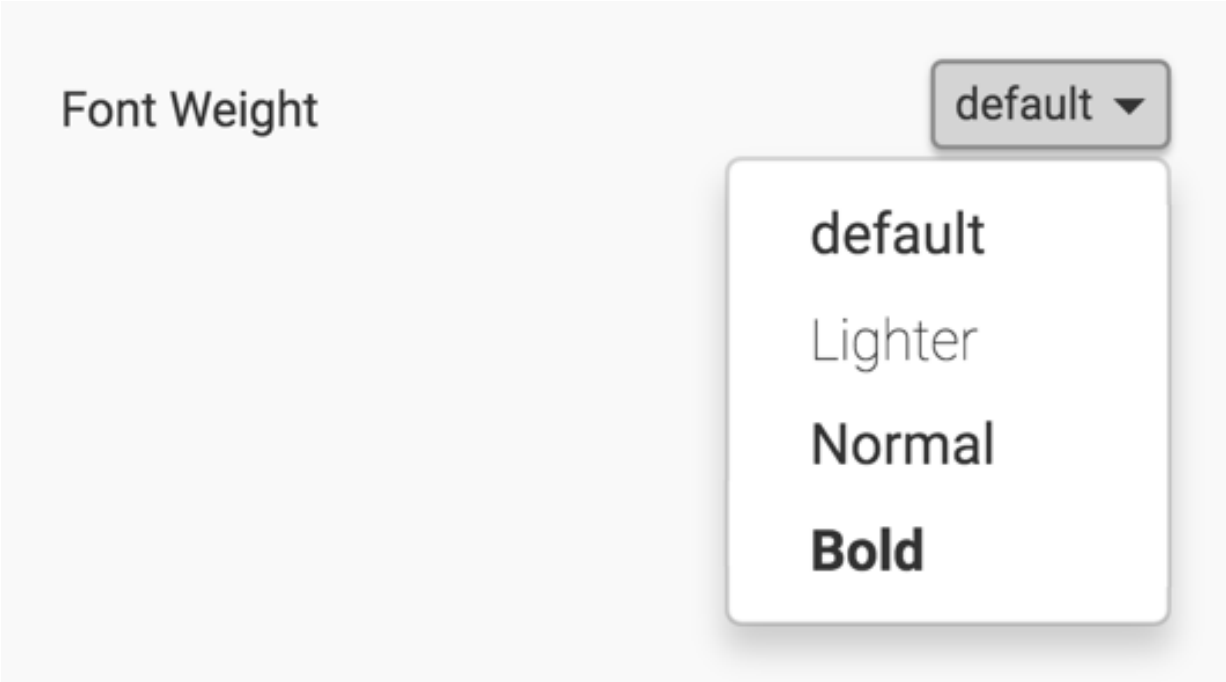

Changing text alignment in the title of a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

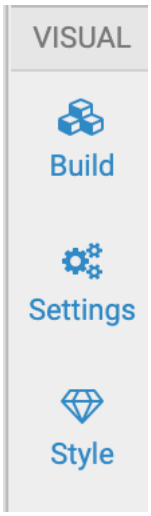

2. In the Settings menu, click Visual Title.

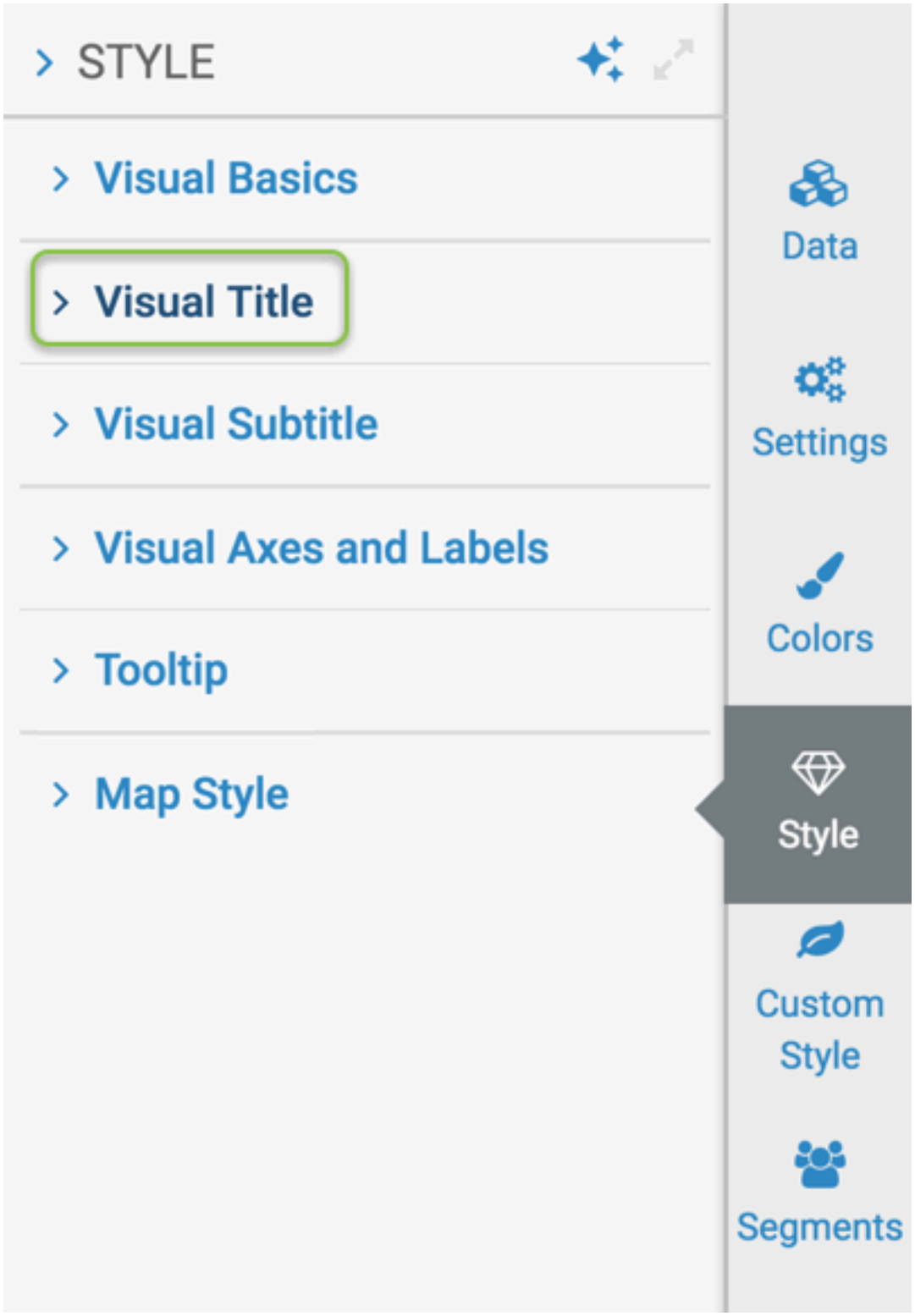

3. To change the text alignment of the title of a visual, navigate to the Visual Title menu, and change the selection of the Alignment option. Text is aligned left by default, and has the additional options for center and right text alignment.

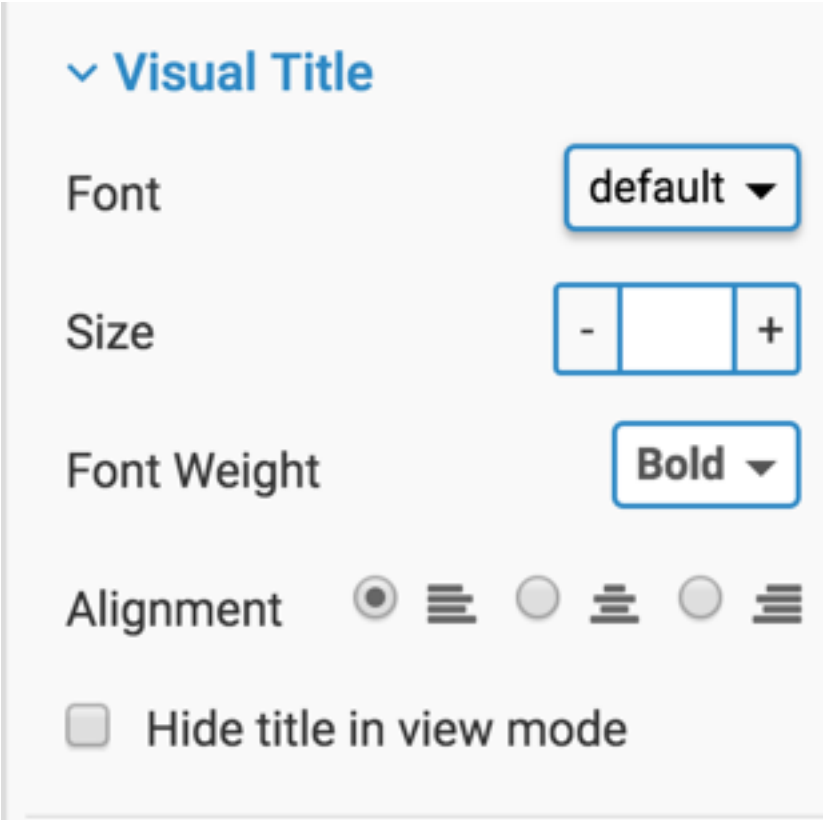

Hiding the title of a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

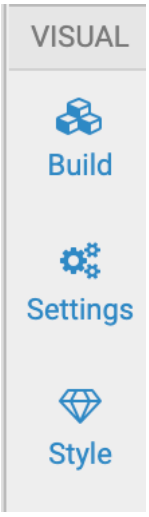

2. In the Settings menu, click Visual Title.

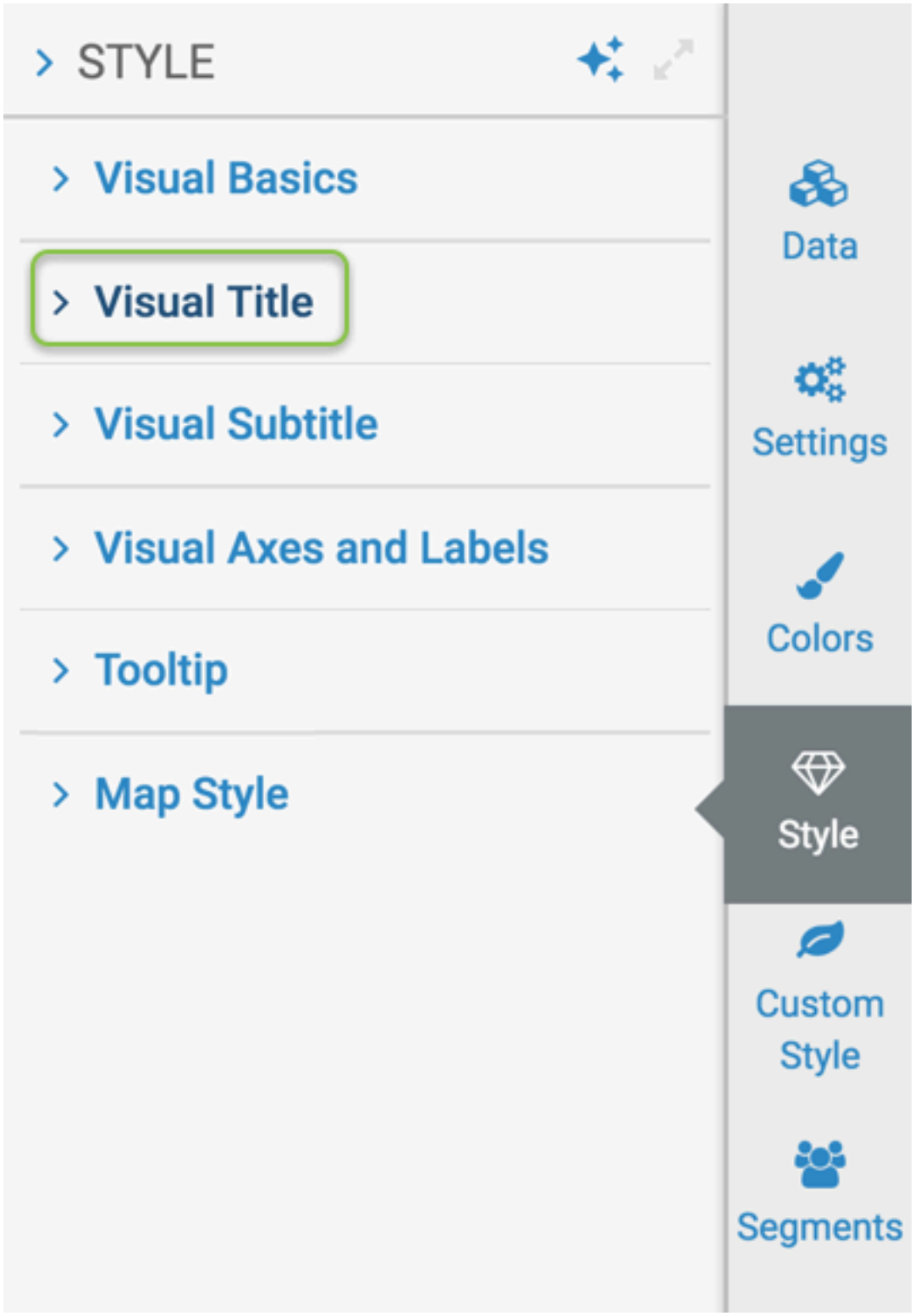

3. To hide the title of a visual in View mode (run time), navigate to the Visual Title menu, and select the Hide title in view mode option.

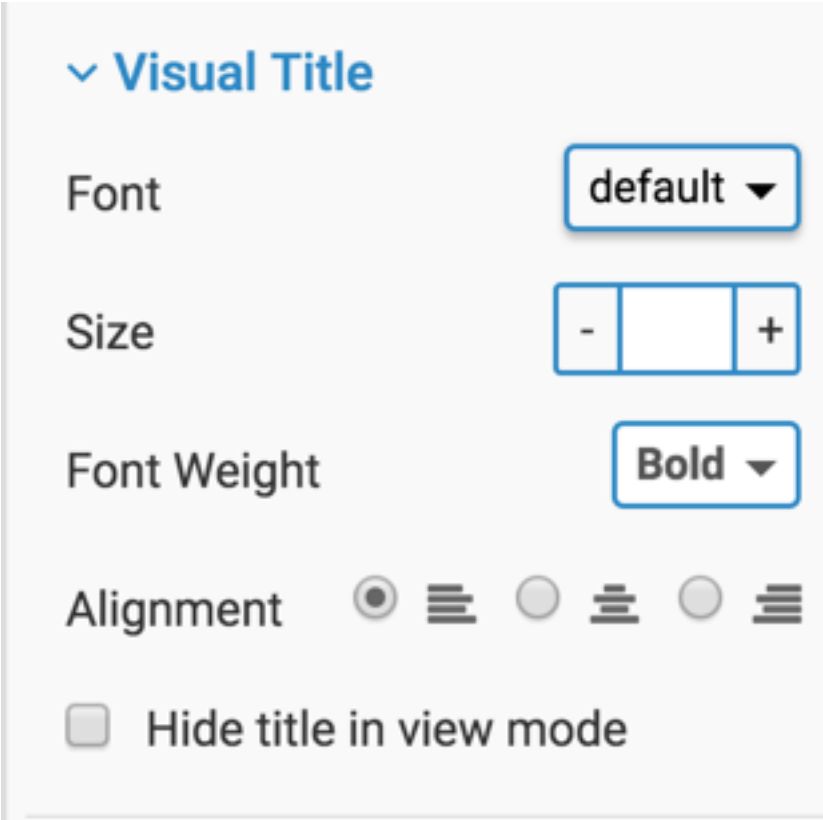

Customizing visual subtitle

In Cloudera Data Visualization, you can manage modify the display options of visual subtitles.

Changing font family in the subtitle of a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

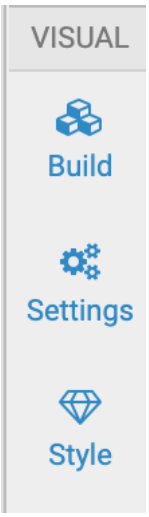

2. In the Settings menu, click Visual Title.

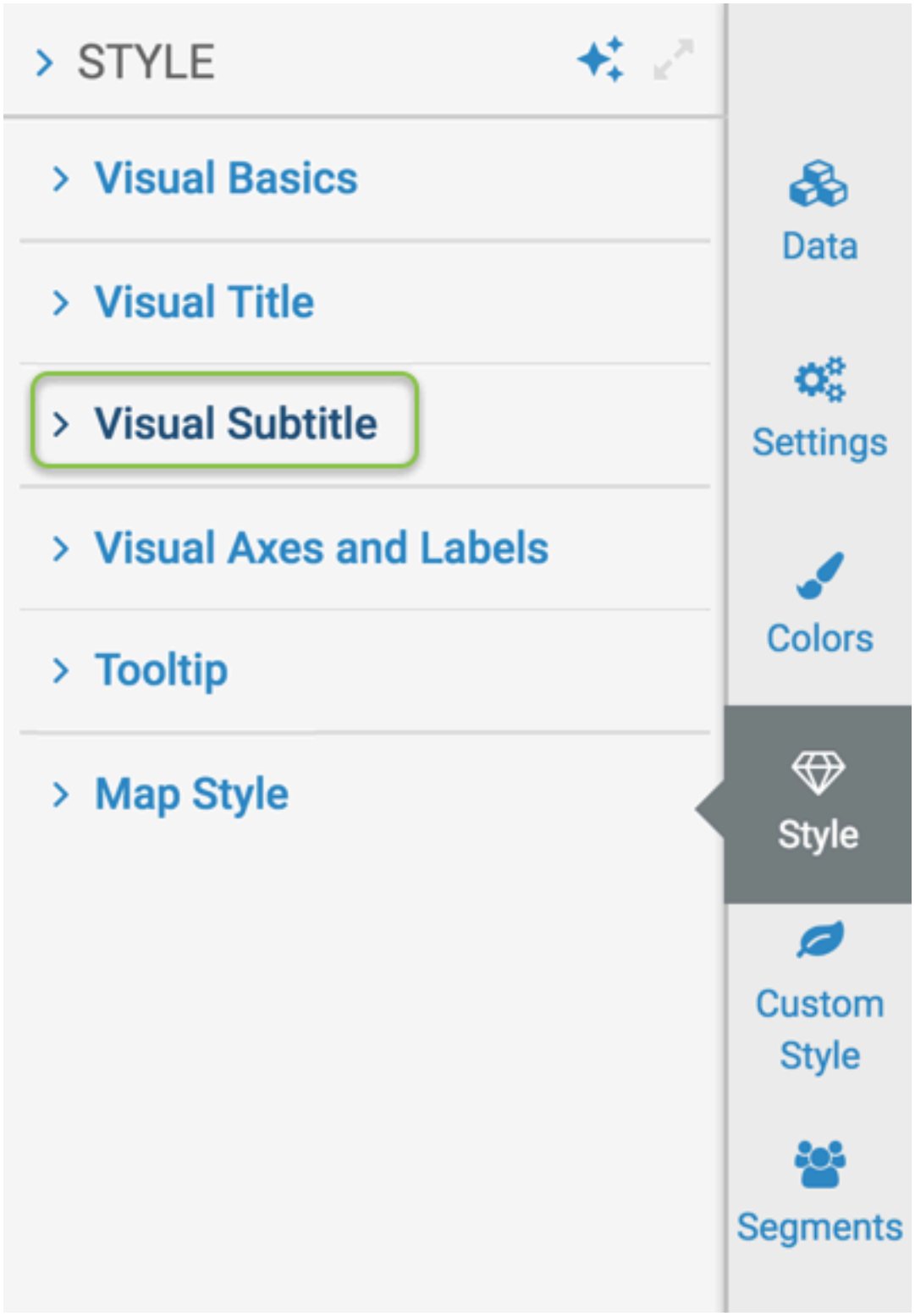

3. To change the font used by the subtitle of a visual, navigate to the Visual Subtitle menu, open the Font Family menu, and make a selection.

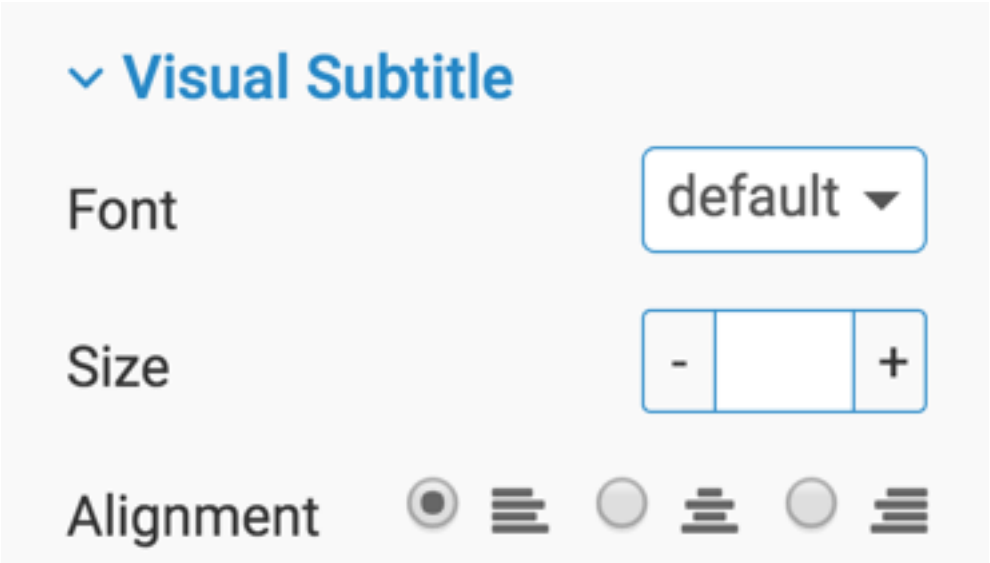

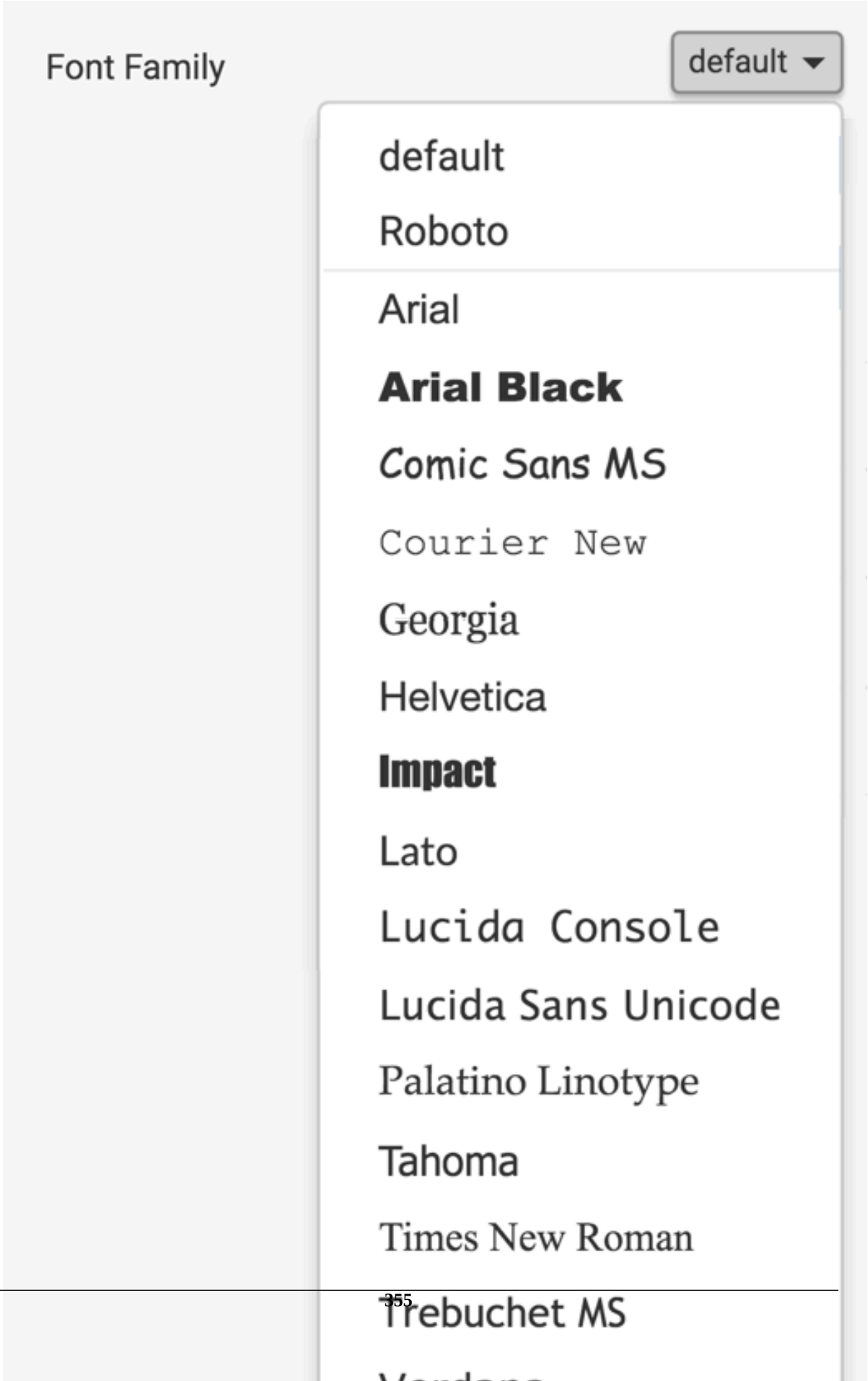
Font Family
default
default
Roboto
Arial
Arial Black
Comic Sans MS
Courier New
Georgia
Helvetica
Impact
Lato
Lucida Console
Lucida Sans Unicode
Palatino Linotype
Tahoma
Times New Roman
Trebuchet MS

Changing font size in the subtitle of a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

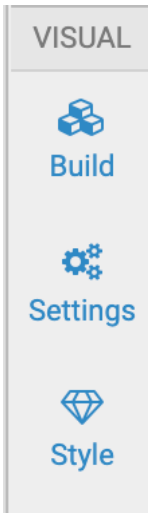

2. In the Settings menu, click Visual Subtitle.

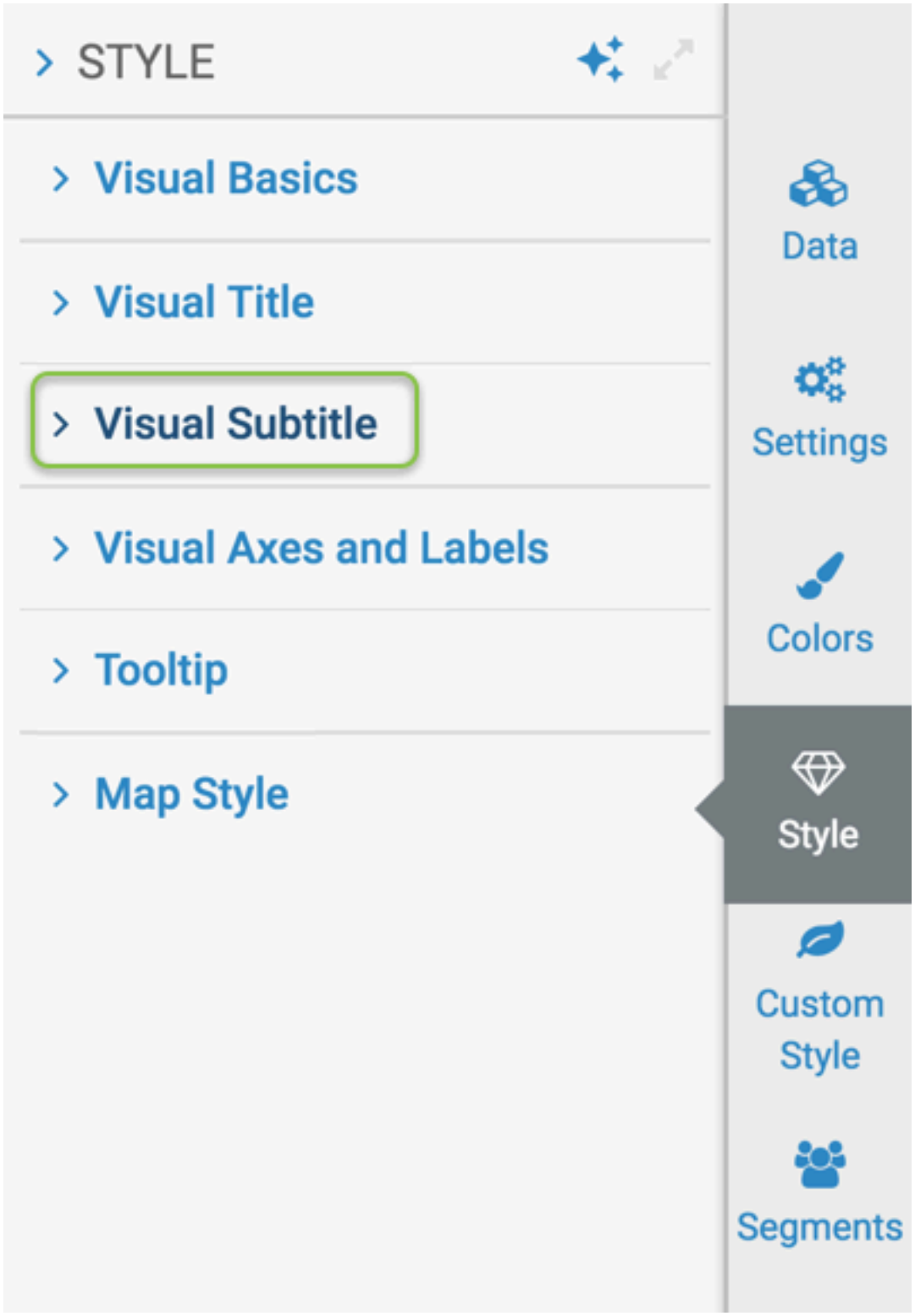

3. To change the font size used by the subtitle of a visual, navigate to the Visual Subtitle menu, and make adjustments in the Font Size selector.

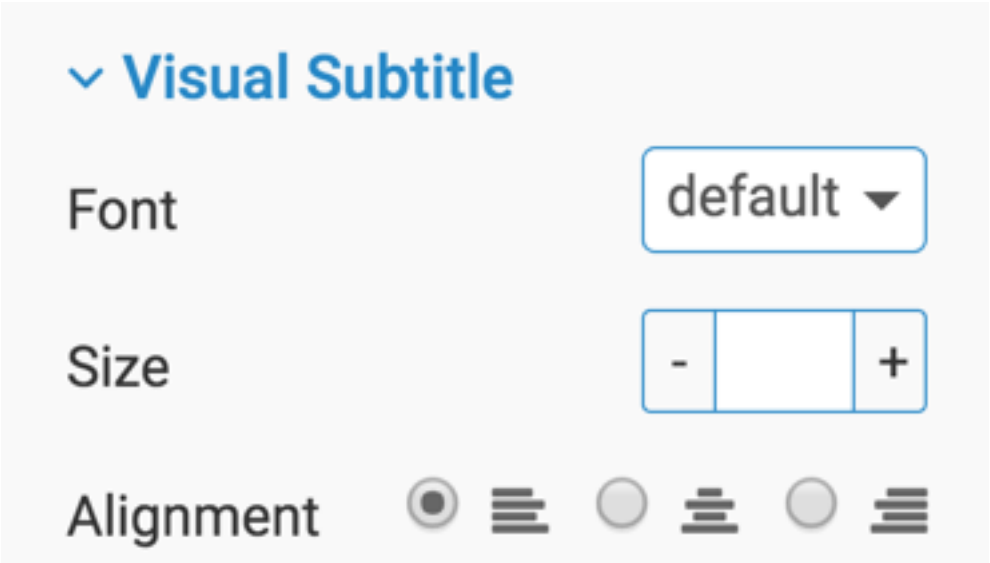

Changing text alignment in the subtitle of a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

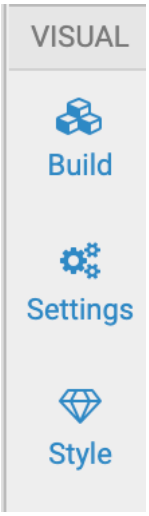

2. In the Settings menu, click Visual Subtitle.

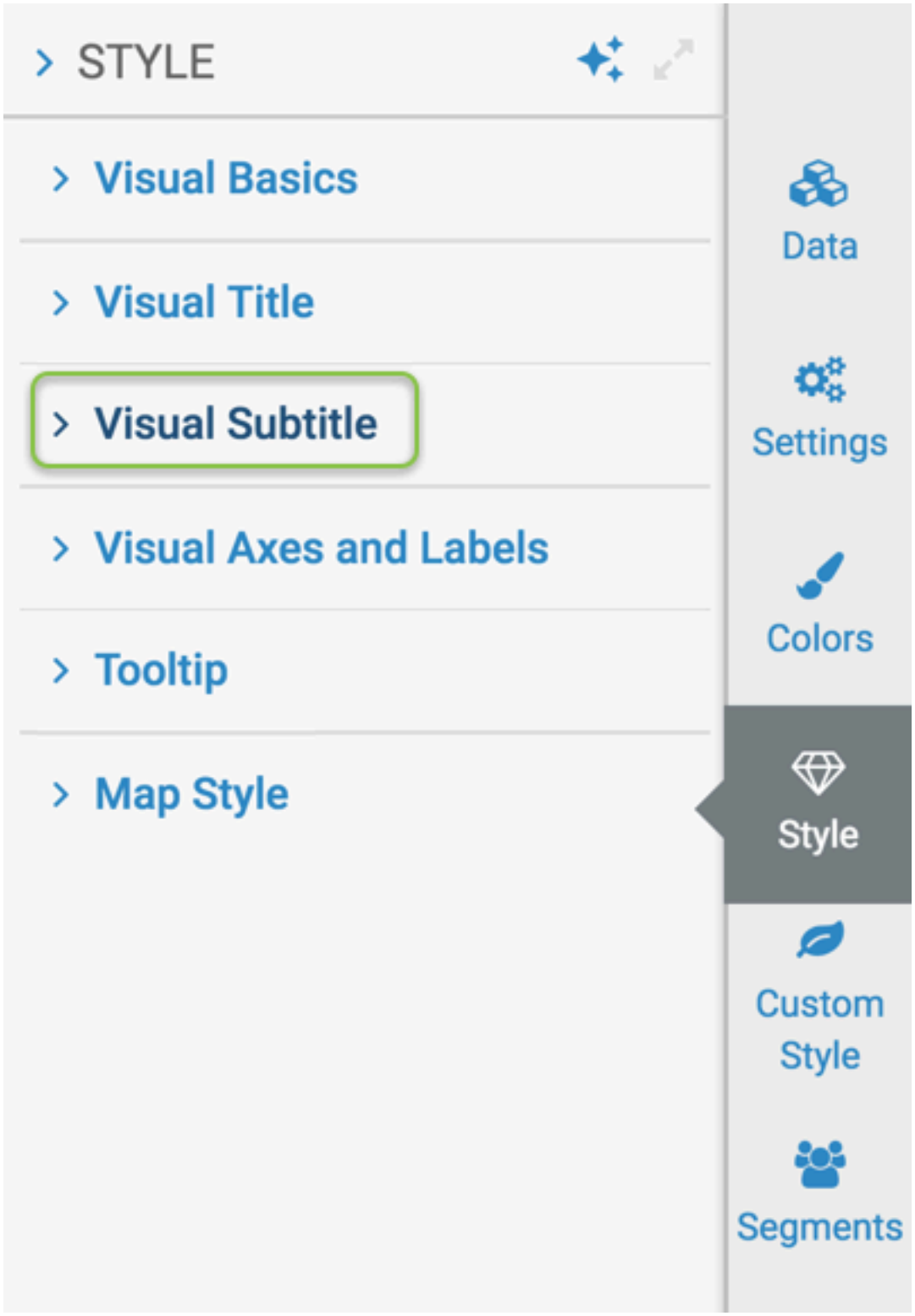

3. To change the text alignment of the subtitle of a visual, navigate to the Visual Subtitle menu, and change the selection of the Alignment option. Text is aligned left by default, and has the additional options for center and right text alignment.

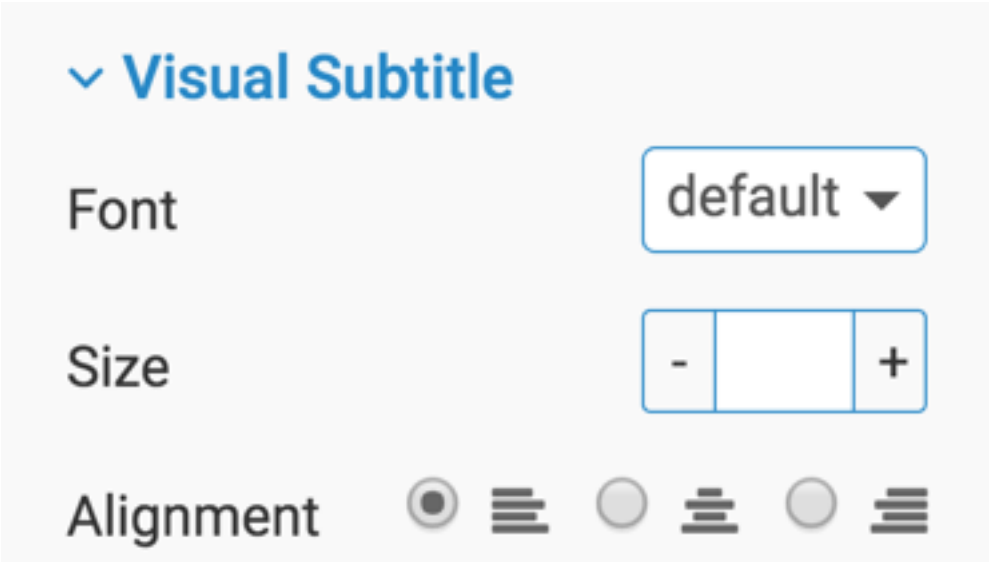

Customizing visual axes and labels

In Cloudera Data Visualization, you can manage several display options for the axes and lables of your visuals.

Axis label font size in a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

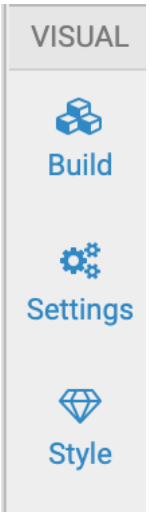

2. In the Settings menu, click Visual Title.

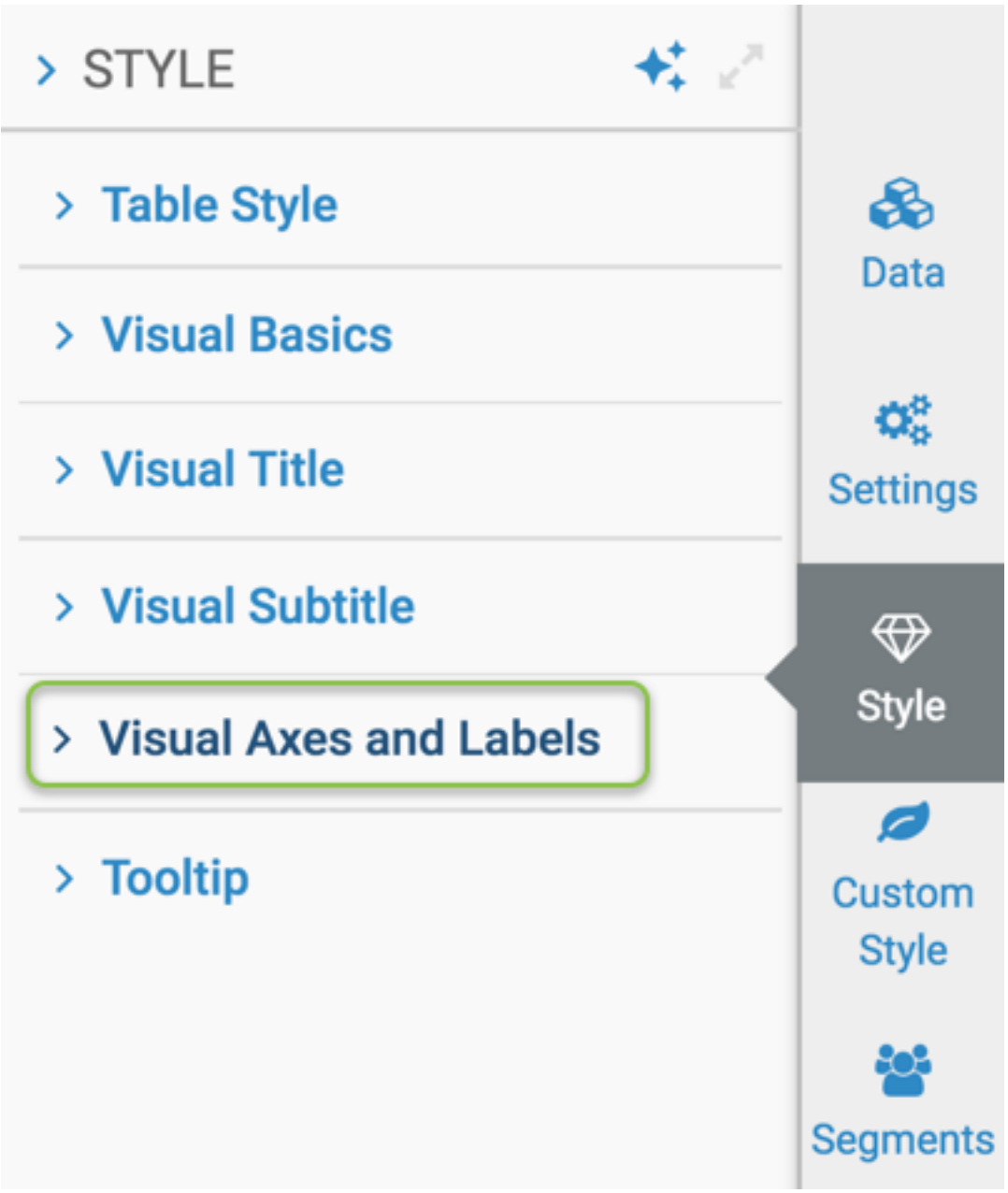

3. To change the font size used by the axis labels of a visual, navigate to the Customizing Visual Axes and Labels menu, and make adjustments in the Axis Label Font Size selector.

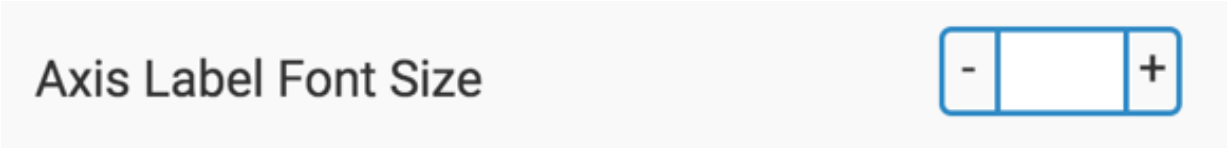

Example

In the following example, compare the two visuals after changing the Axis Label Font Size to 17.

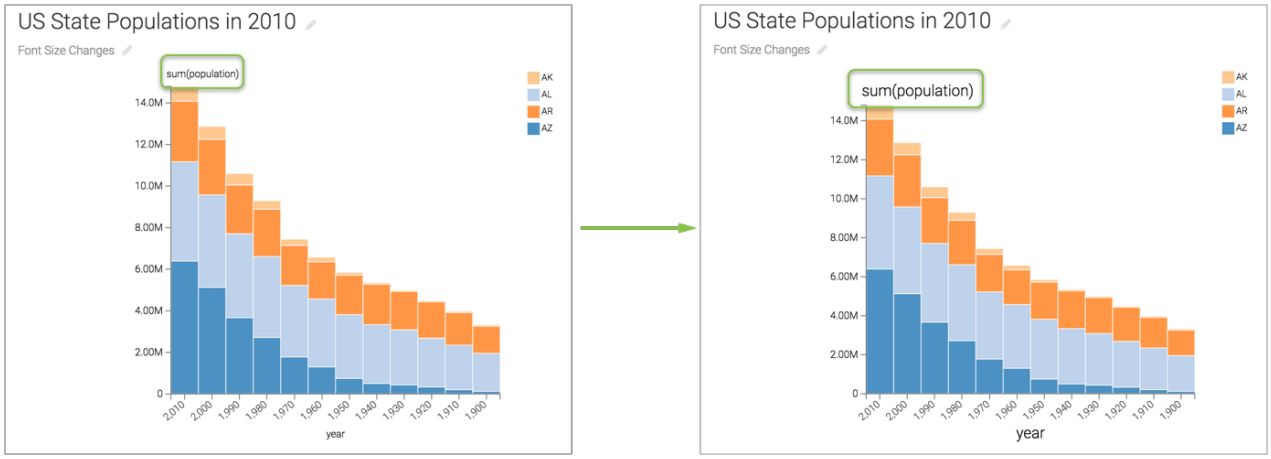

Ticks font size in a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

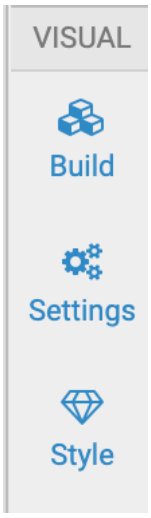

2. In the Settings menu, click Visual Title.

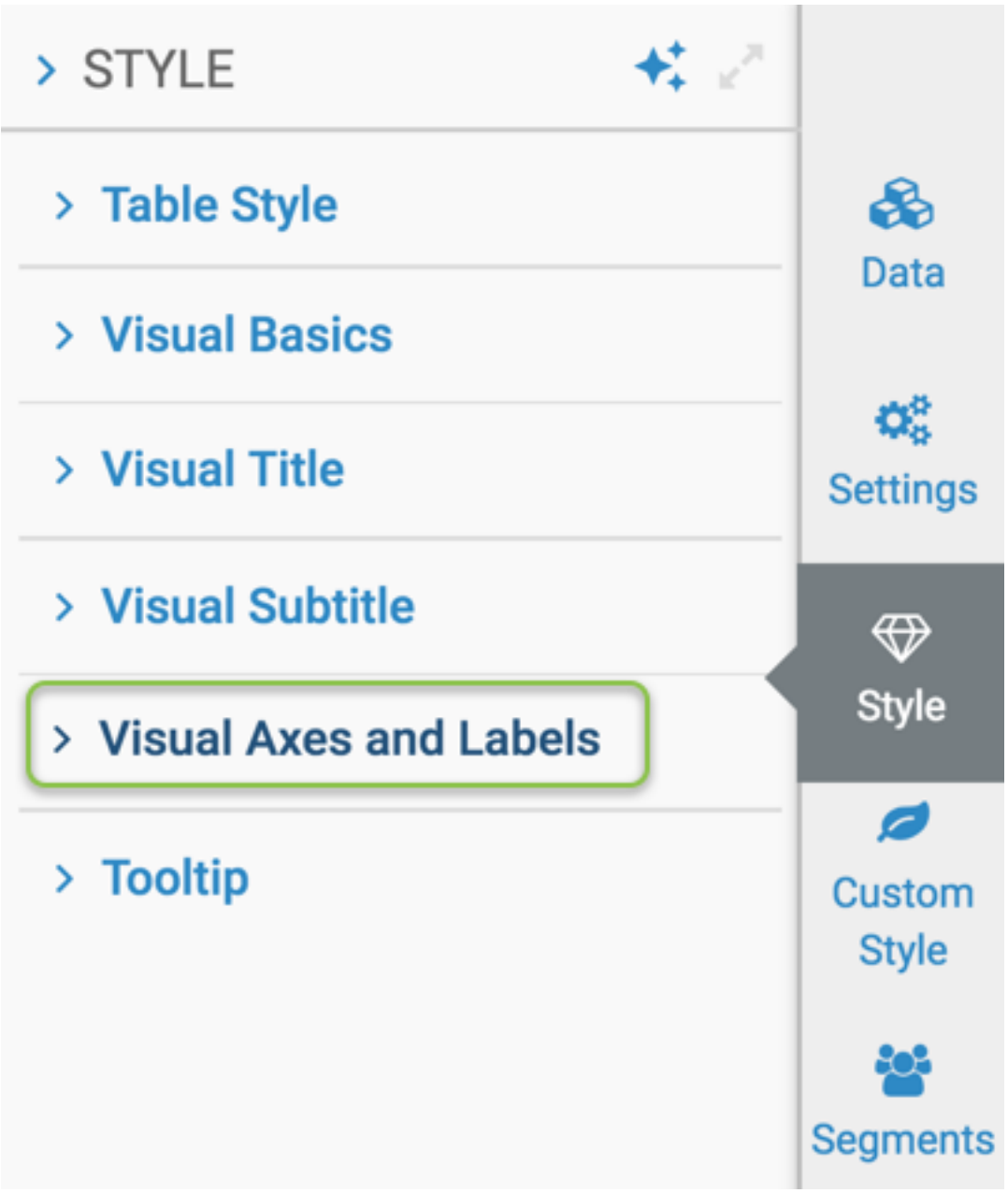

3. To change the font size of ticks in a visual, navigate to the Customizing Visual Axes and Labels menu, and make adjustments in the Axis Ticks Font Size selector.

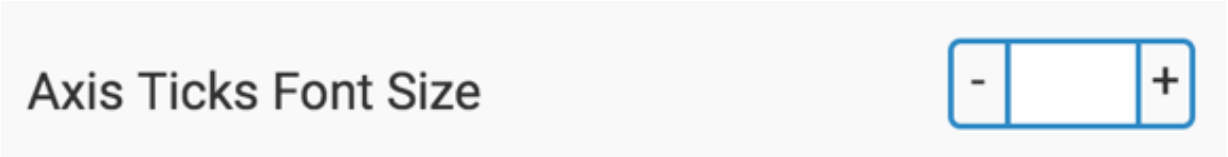

Example

In the following example, compare the two visuals after changing the Axis Ticks Font Size to 17.

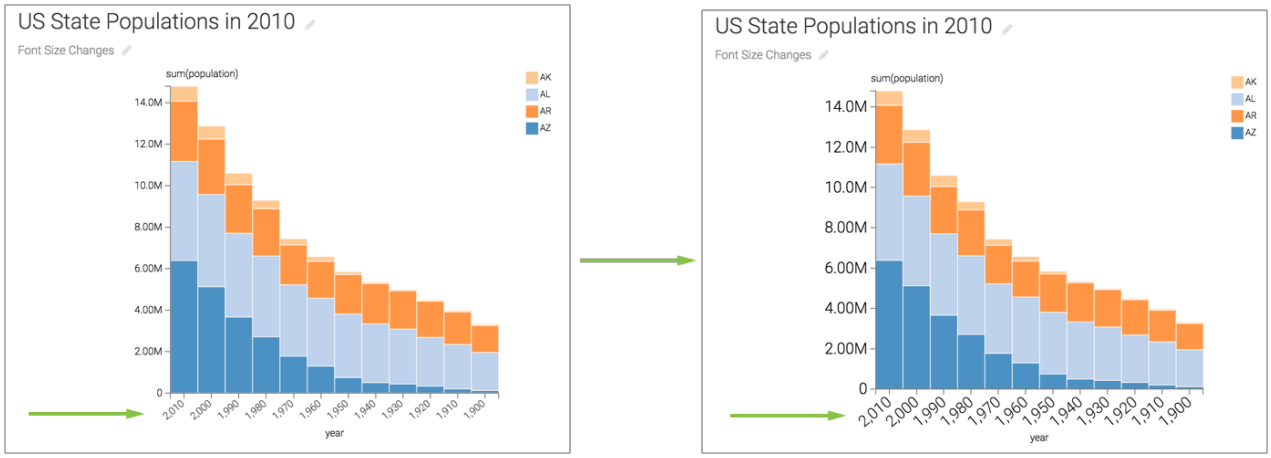

Label font size in a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

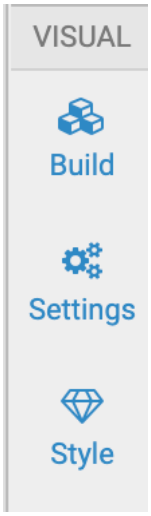

2. In the Settings menu, click Visual Title.

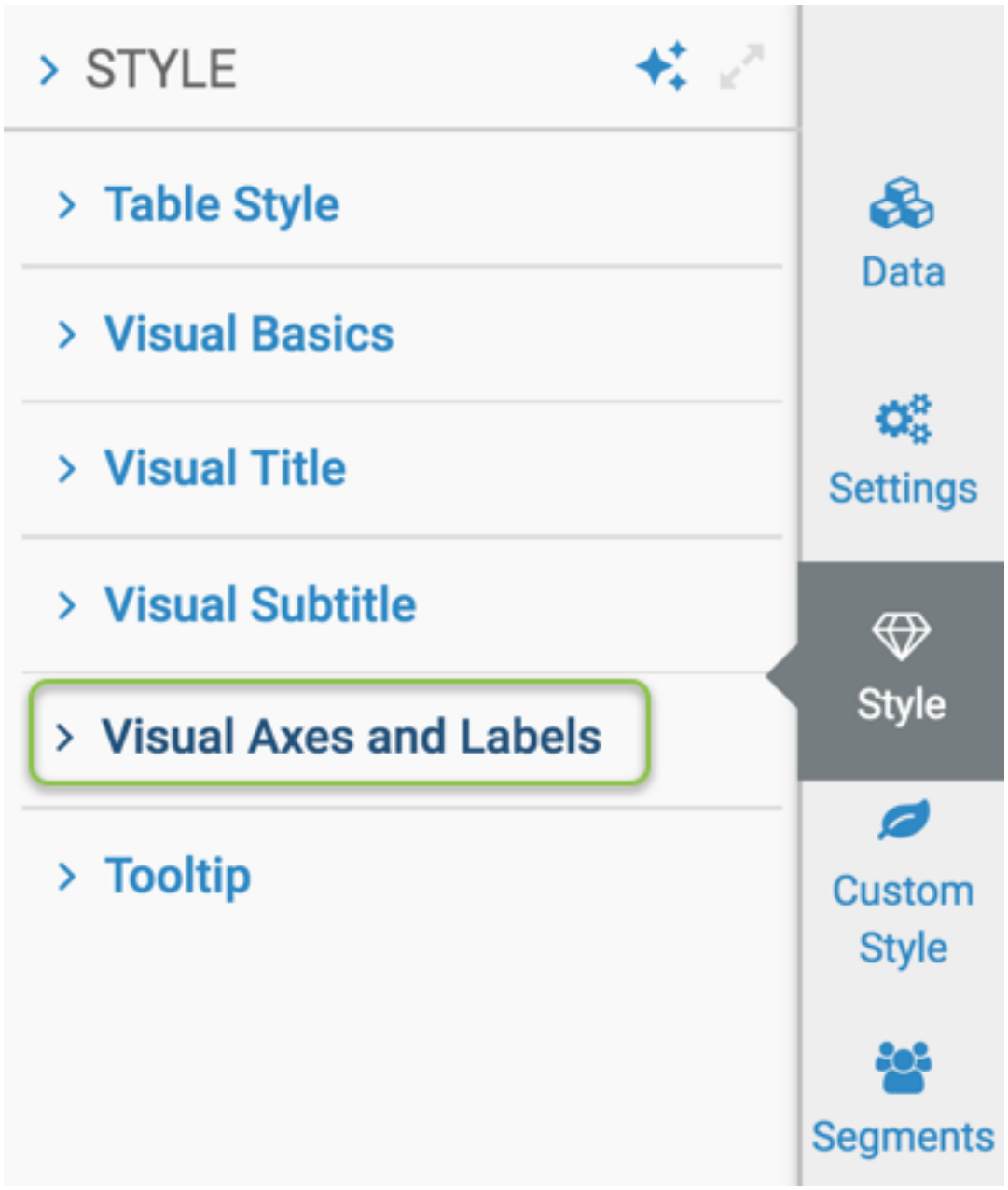

3. To change the font size used by the labels of a visual, navigate to the Customizing Visual Axes and Labels menu, and make adjustments in the Label Font Size selector.

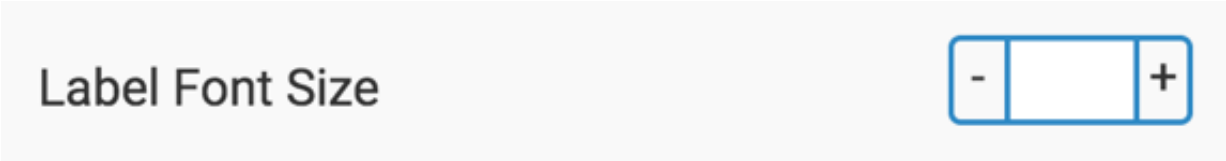

Label text color in a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

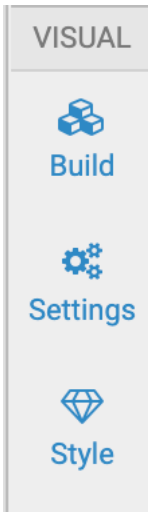

2. In the Settings menu, click Visual Title.

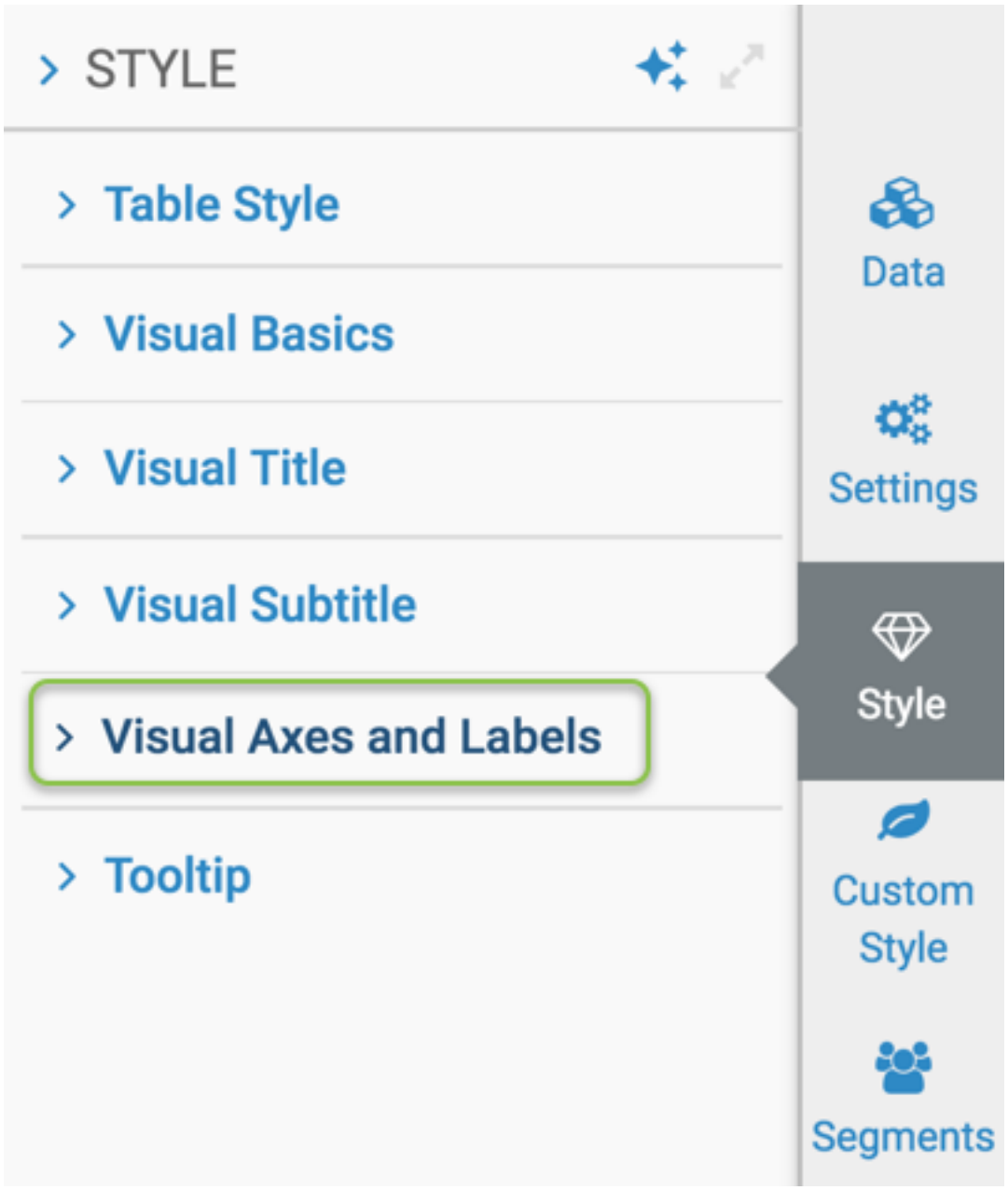

3. To change the color used by the legend of a visual, navigate to the StyleVisual Axes and Labels menu, and make adjustments in the Label Text Color selector.

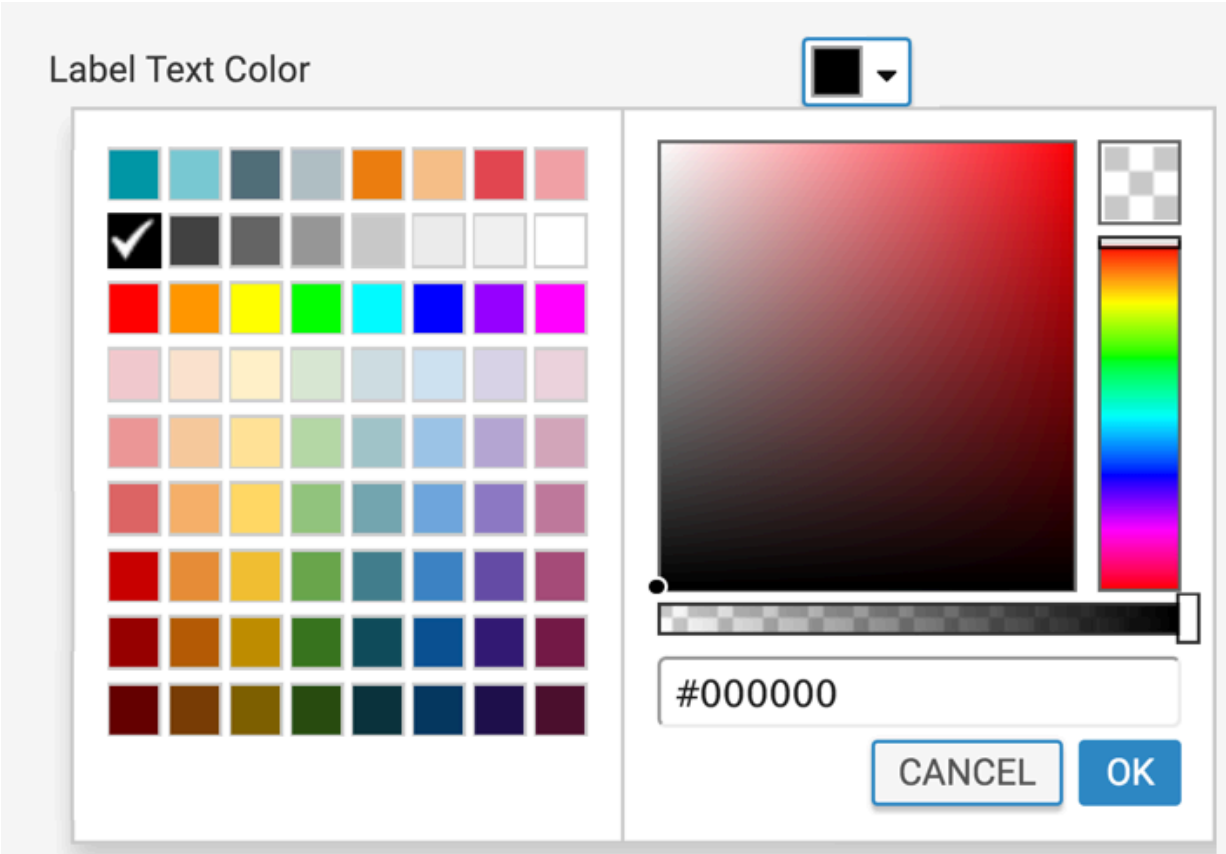

Example

In the following example, compare the two visuals; the first one uses the default color scheme in labels, and the second one uses black font color.

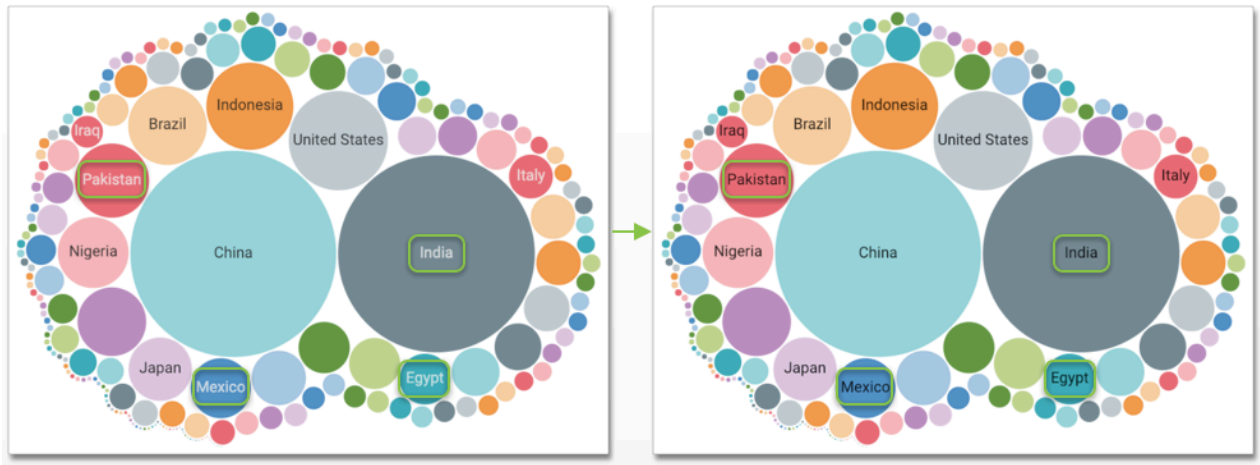

Trellis font size in a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

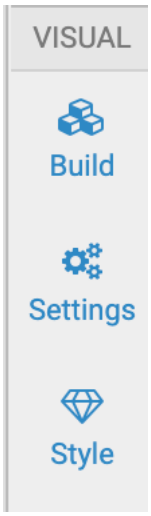

2. In the Settings menu, click Visual Title.

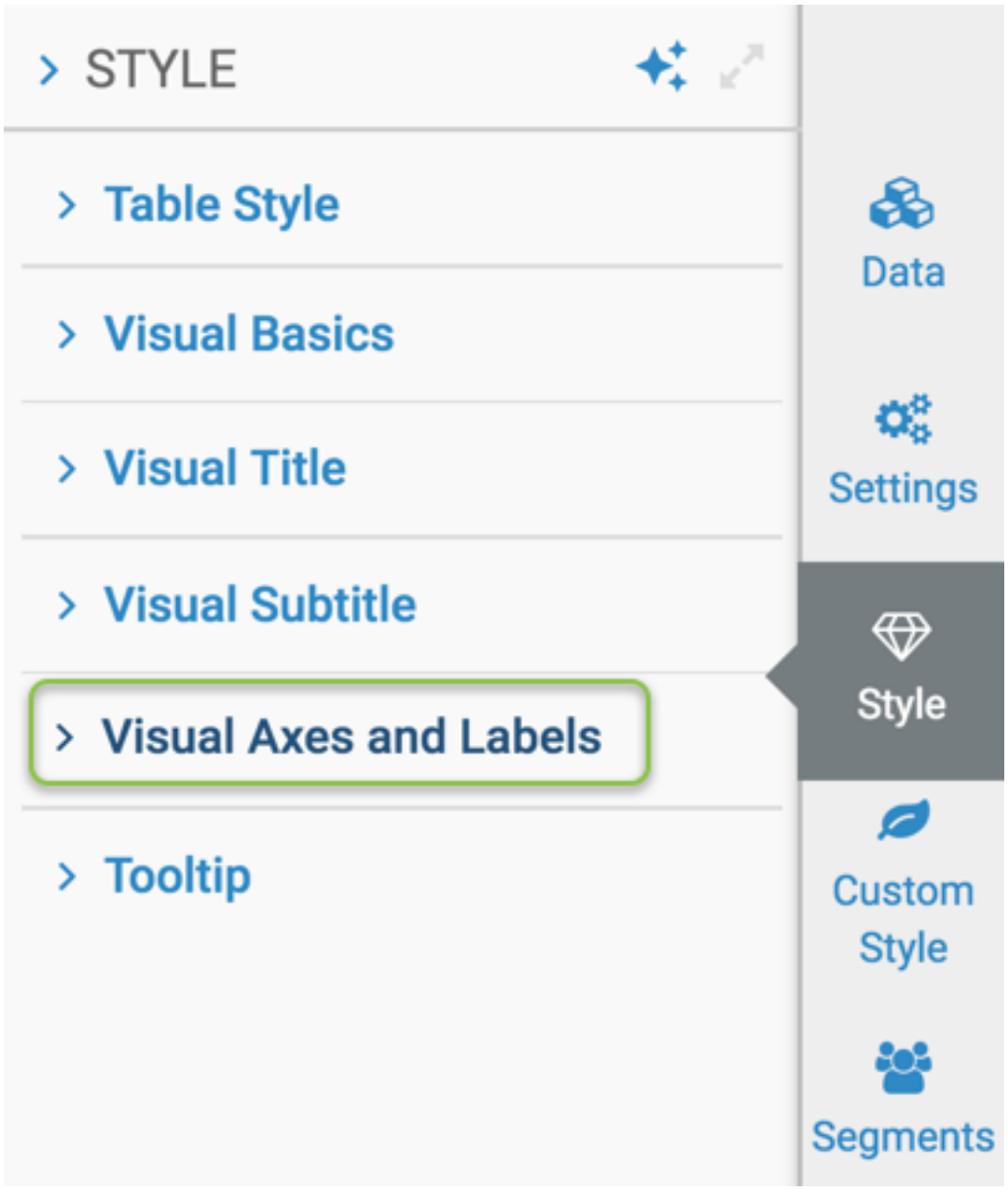

3. To change the font size used of a trellis in a visual, navigate to the Customizing Visual Axes and Labels menu, and make adjustments in the Trellis Font Size selector.

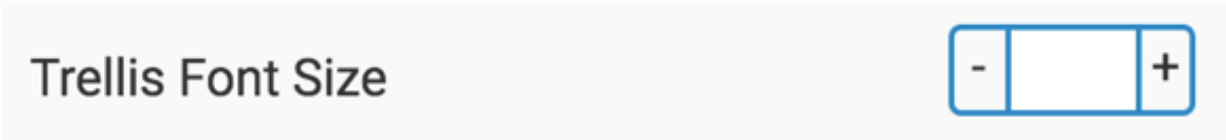

Legend font size in a visual

Procedure

1. On the right side of Visual Designer, click the Settings menu.

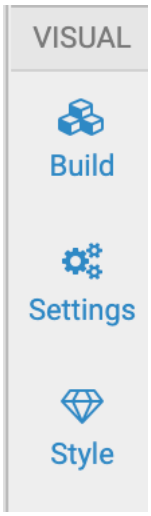

2. In the Settings menu, click Visual Title.

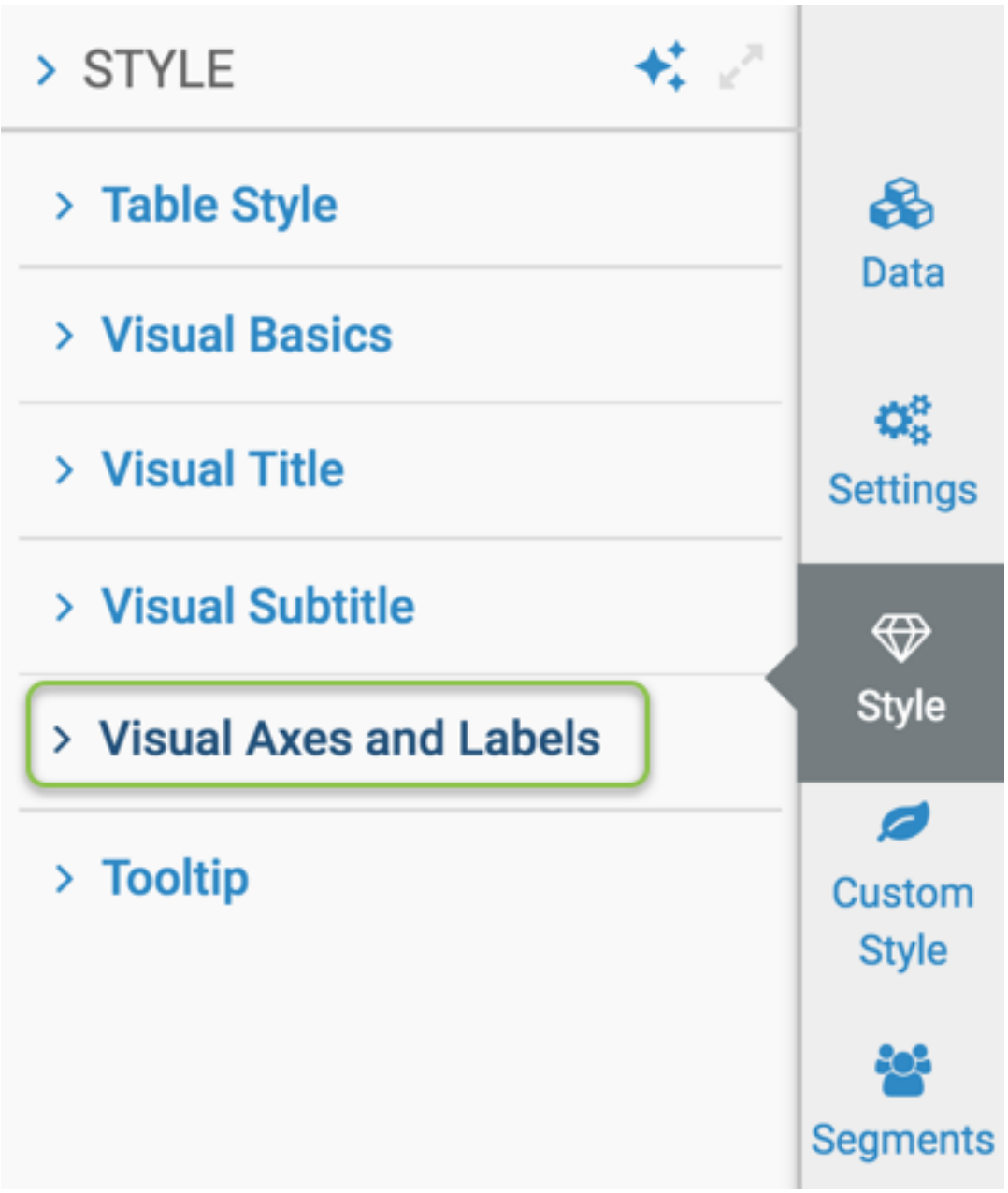

3. To change the font size used by the legend of a visual, navigate to the Customizing Visual Axes and Labels menu, and make adjustments in the Legend Font Size selector.

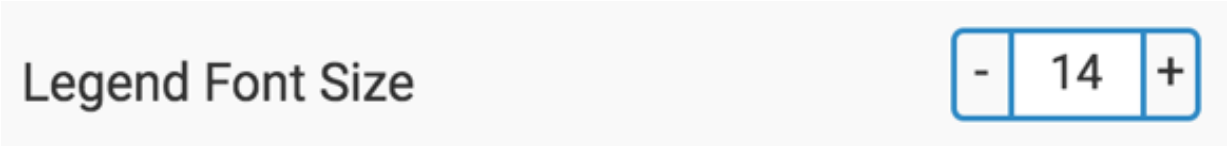

Example

In the following example, compare the two visuals after changing the Legend Font Size to 17.

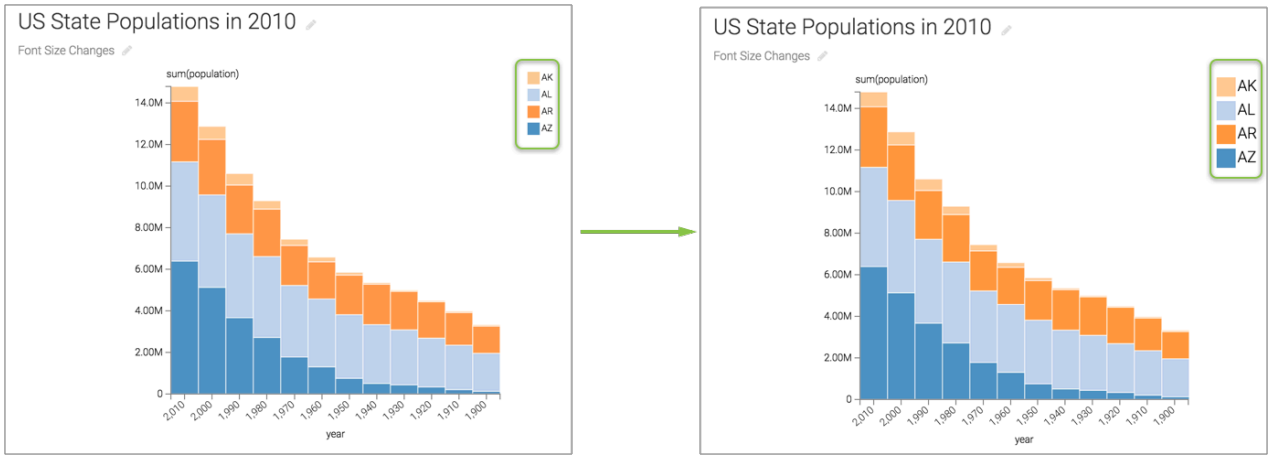

Customizing tooltip

In Cloudera Data Visualization, you can manage several visual display options for your tooltips.

Changing tooltip font family

Procedure

1. On the right side of Visual Designer, click the Settings menu.

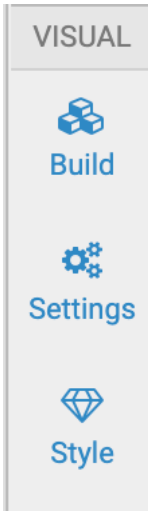

2. In the Settings menu, click Tooltip.

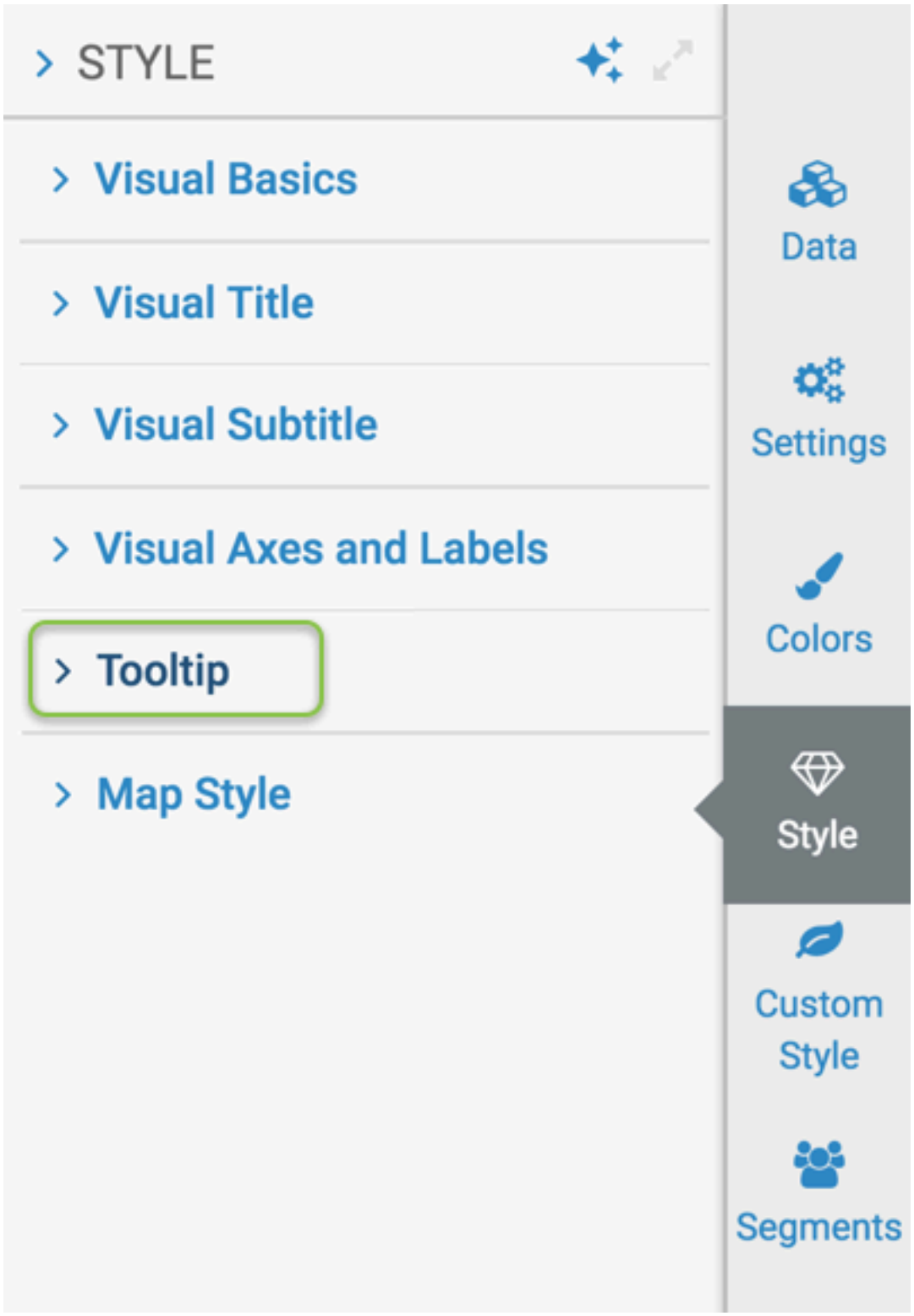

3. To change the tooltip font inside a visual, open the Font Family menu and make a selection there.

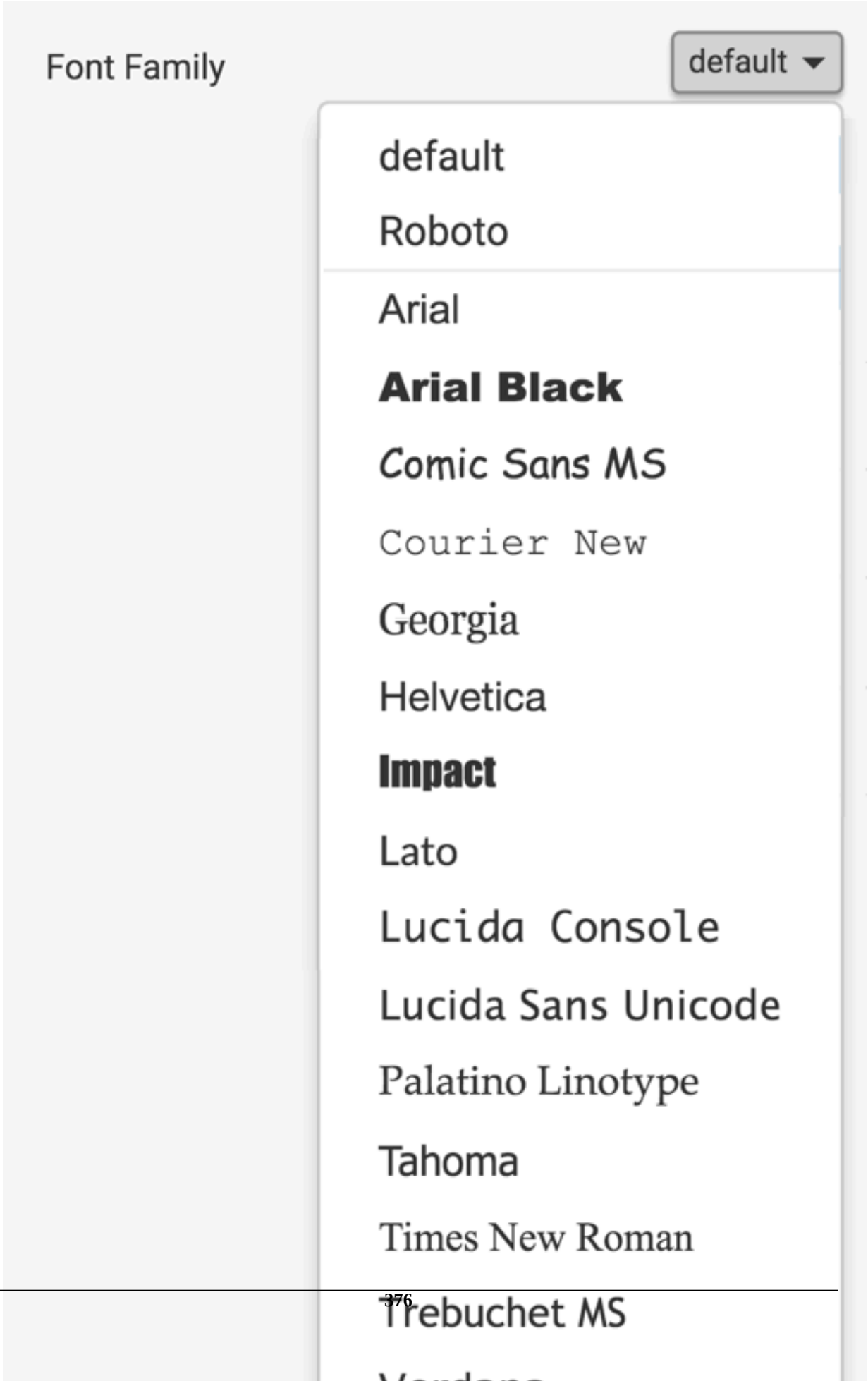
Font Family
default
default
Roboto
Arial
Arial Black
Comic Sans MS
Courier New
Georgia
Helvetica
Impact
Lato
Lucida Console
Lucida Sans Unicode
Palatino Linotype
Tahoma
Times New Roman
Trebuchet MS

Changing tooltip font size

Procedure

1. On the right side of Visual Designer, click the Style menu.

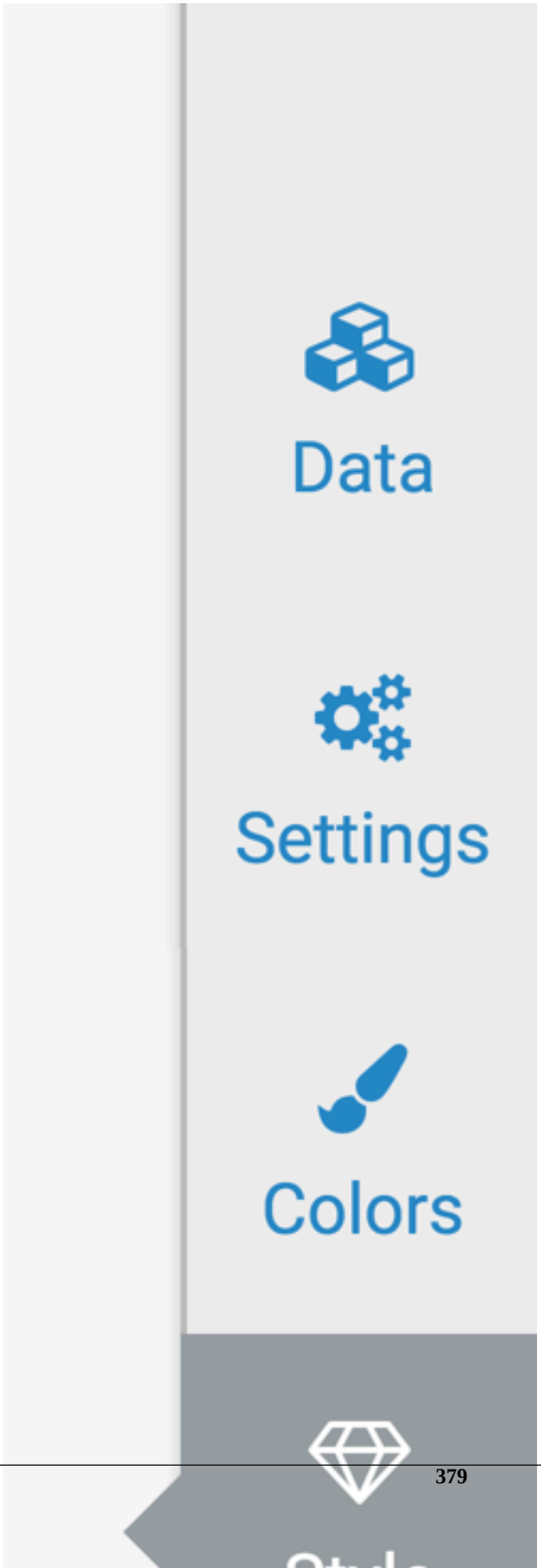
Data
Settings
Colors

2. In the Settings menu, click Tooltip.

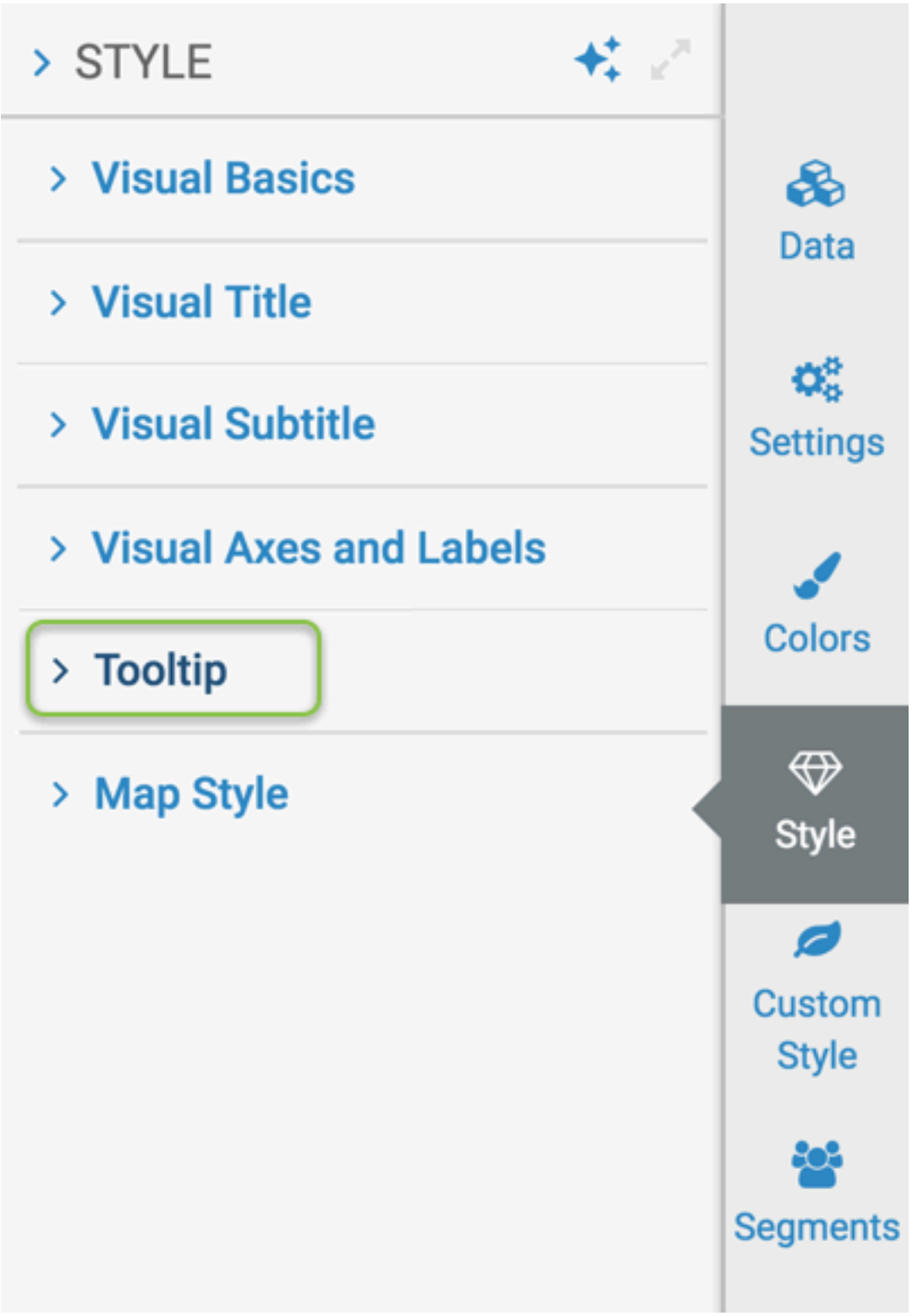

3. To change the font size used by the tooltip in a visual, make adjustments in the Font Size selector.

Customizing aggregate row style

In Cloudera Data Visualization, you can manage several visual display options from the Aggregate Row Style menu.

Changing font style in the aggregate row of a visual

About this task

This applies to Cross Tabulation visuals.

Procedure

1. On the right side of Visual Designer, click the Style menu.

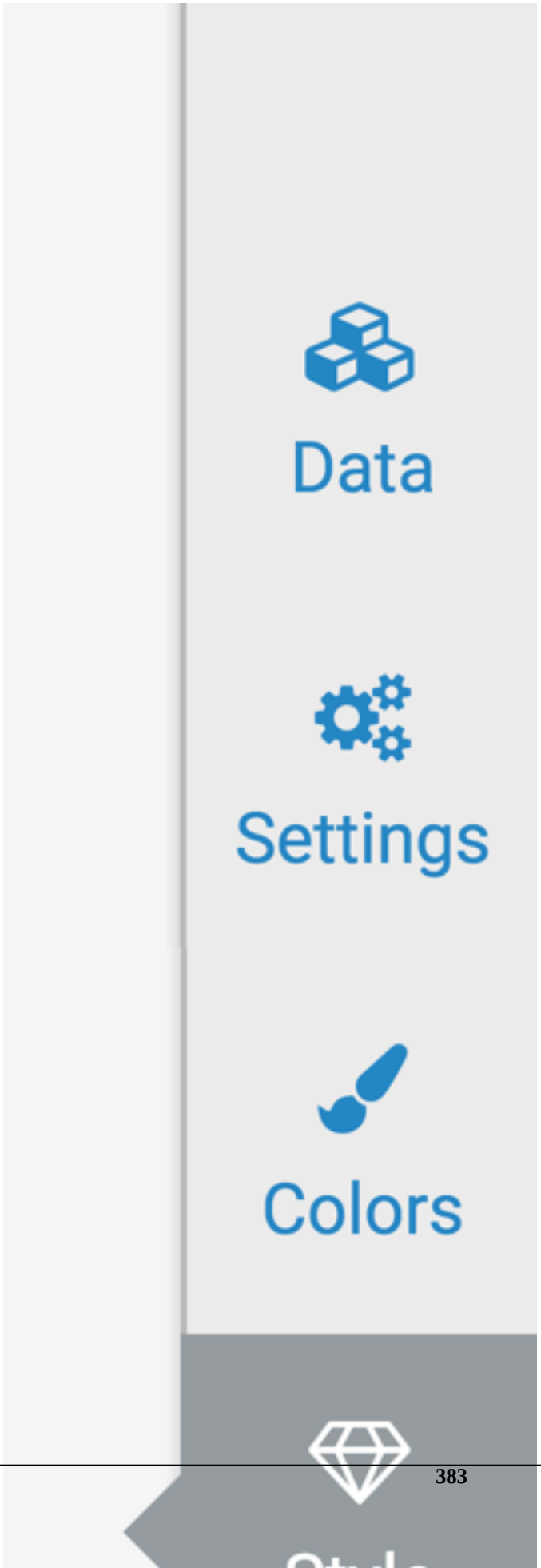
Data
Settings
Colors

2. In the Settings menu, click Aggregate Row Style..

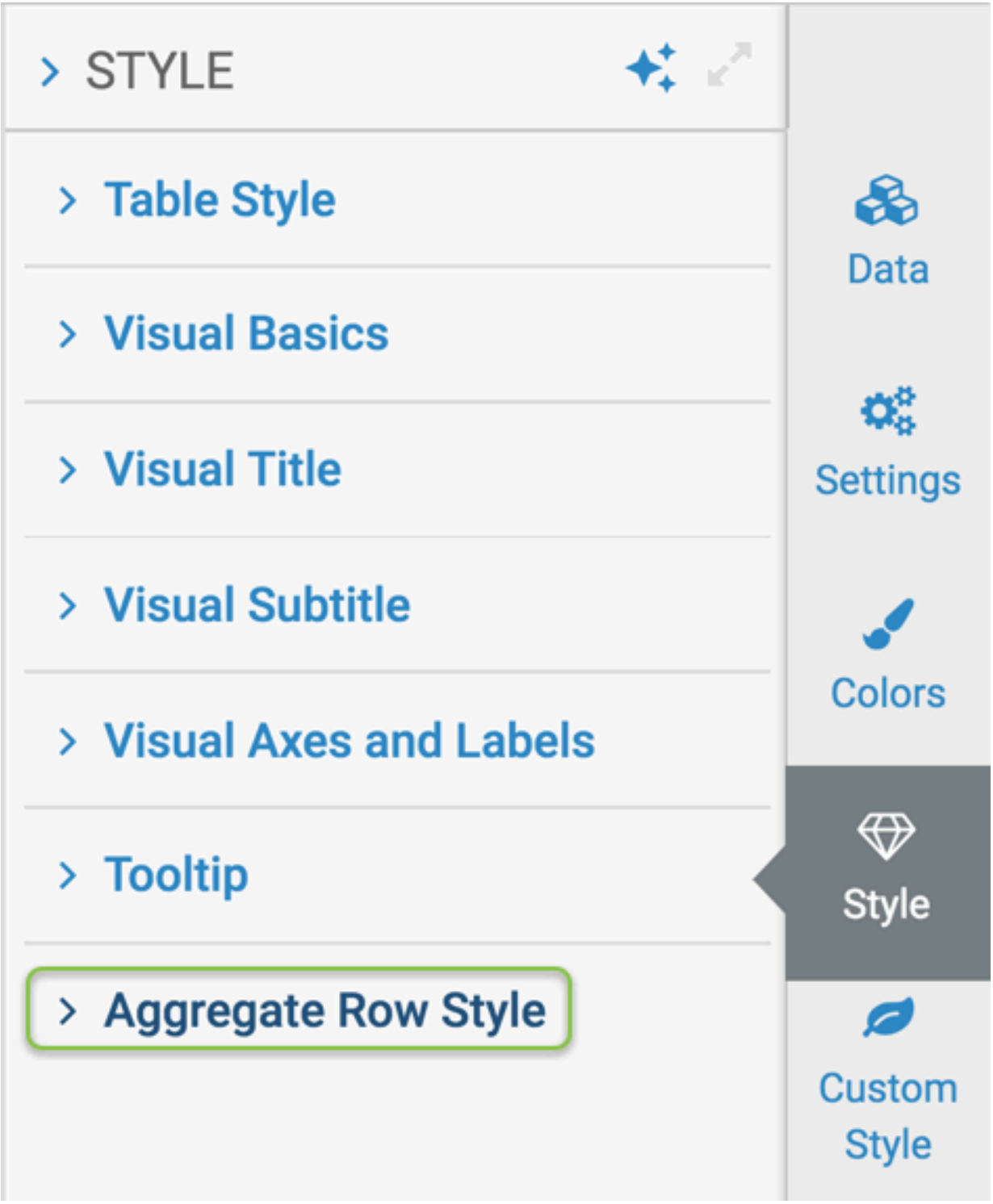

3. To change the font used by the aggregate row of a visual, select Font Style from the Aggregate Row Style menu and choose a font style.

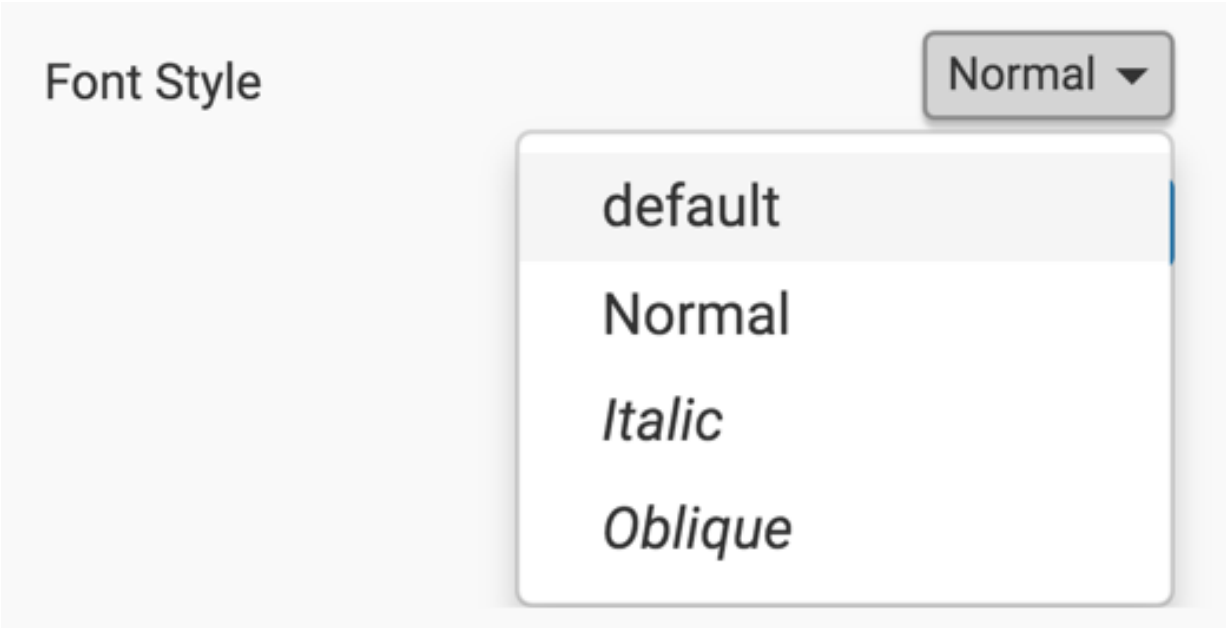

Changing font weight in the aggregate row of a visual

About this task

This applies to Cross Tabulation visuals.

Procedure

1. On the right side of Visual Designer, click the Style menu.

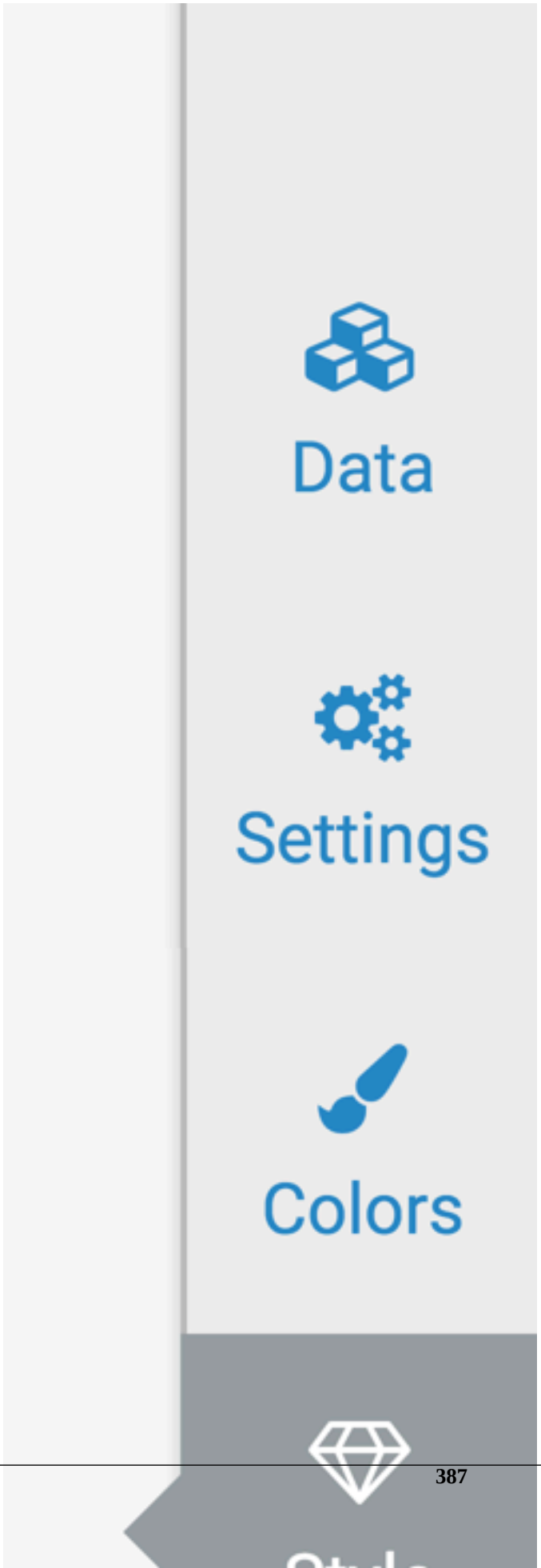
Data
Settings
Colors

2. In the Settings menu, click Aggregate Row Style..

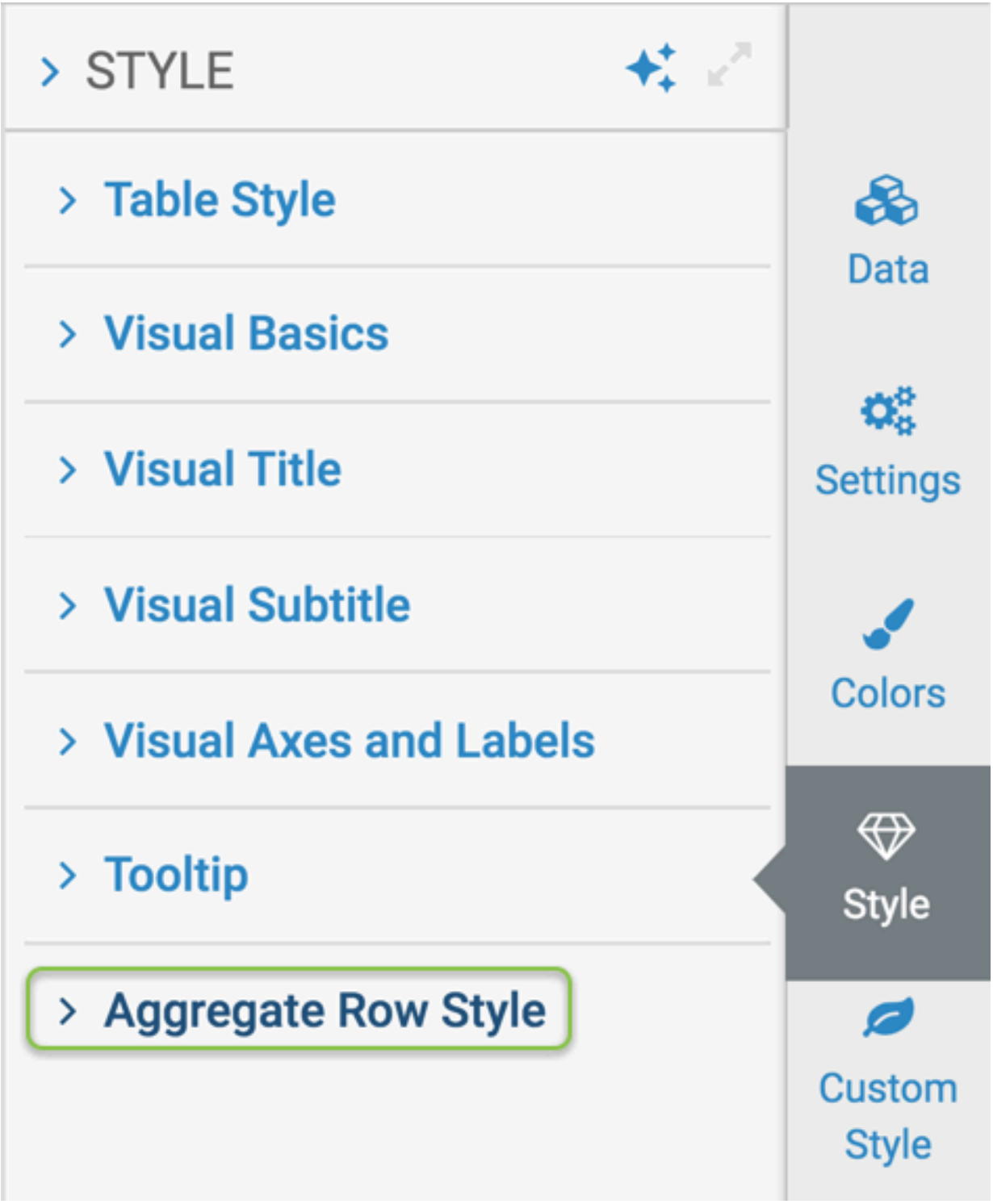

3. To change the font weight used by the aggregate row of a visual, navigate to the Style menu, and select Font Weight from the Aggregate Row Style menu and choose a fonnt weight.

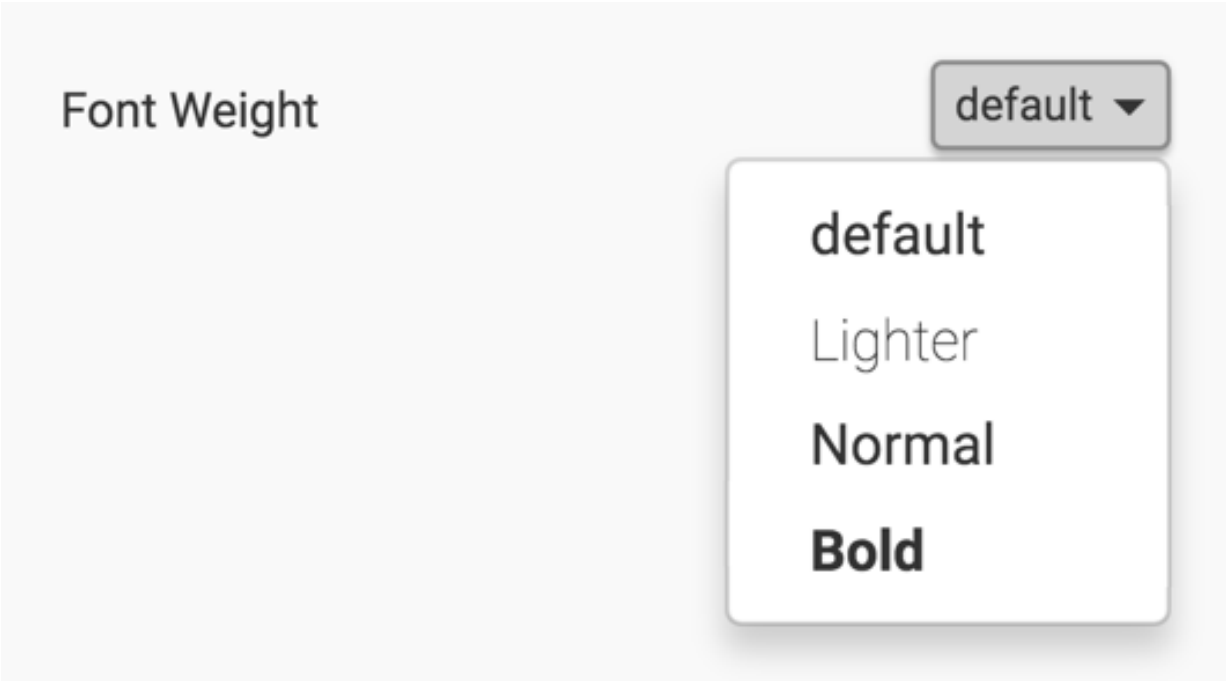

Customizing visual colors

The type of color customization available for each visual in Cloudera Data Visualization depends on the visual type.

For example:

- Most visuals that support color have the option of changing the color palette, or to reverse the color palette.
- Visuals with optional color, such as cross tabulation, have an option to enable or disable colors.
- Visuals that rely on color as a principle method of conveying information, such as a bar chart, have an option to customize color opacity.
- Most commonly-used visuals support color assignment based on dimension value.
- All visuals that use the Color shelf also support specific color assignment for dimension values.
- Some visuals, such as extension and rich text, do not support color customization at all.

For details, see the following color customization options:

Enabling reverse color palette of visuals

Cloudera Data Visualization enables you to reverse the color palette in your visualizations.

Procedure

To reverse the color palette in your visualization, select Reverse order of colors palettes.

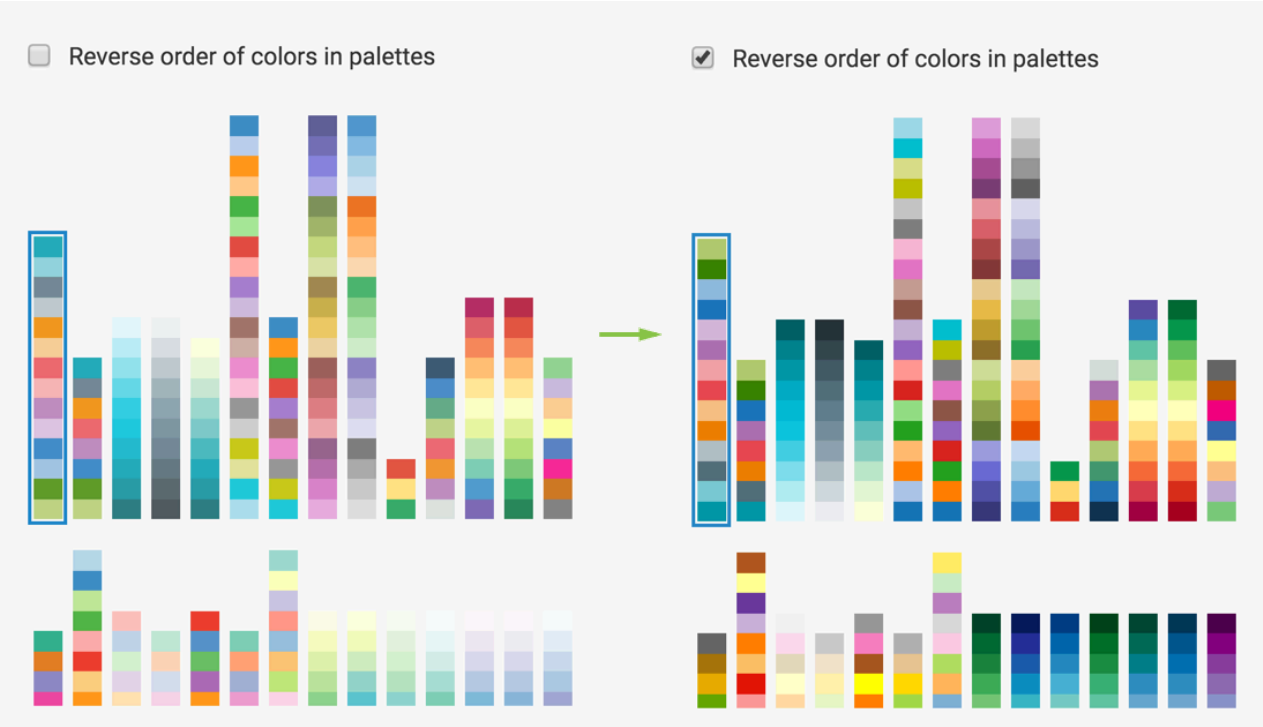

Note that the colors in the visual flip; the legend is a good indicator of this.

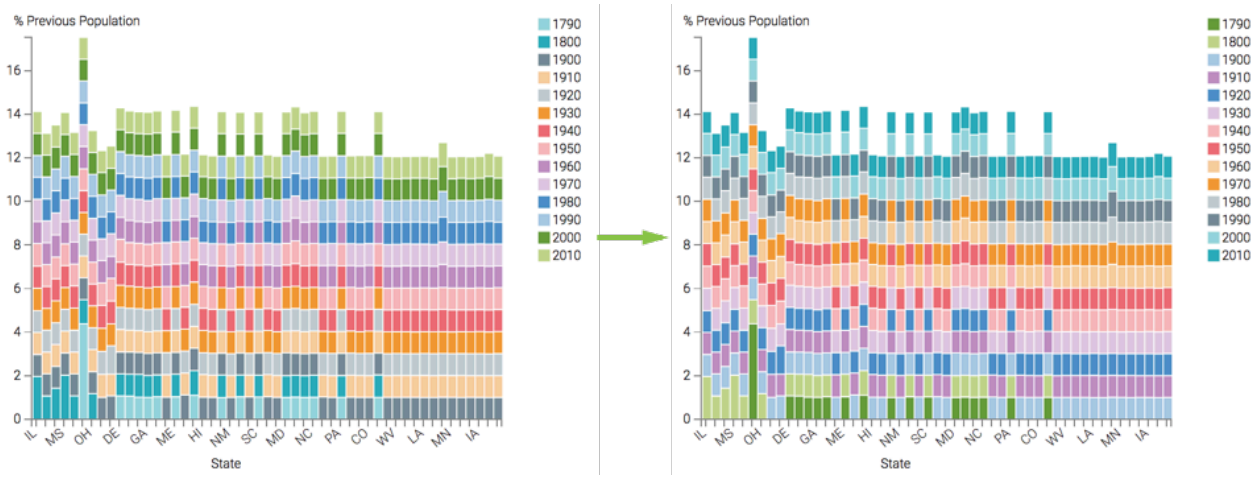

The color reverse effect is particularly noticeable on monochromatic and dichromatic color palettes, which typically use a gradient legend.

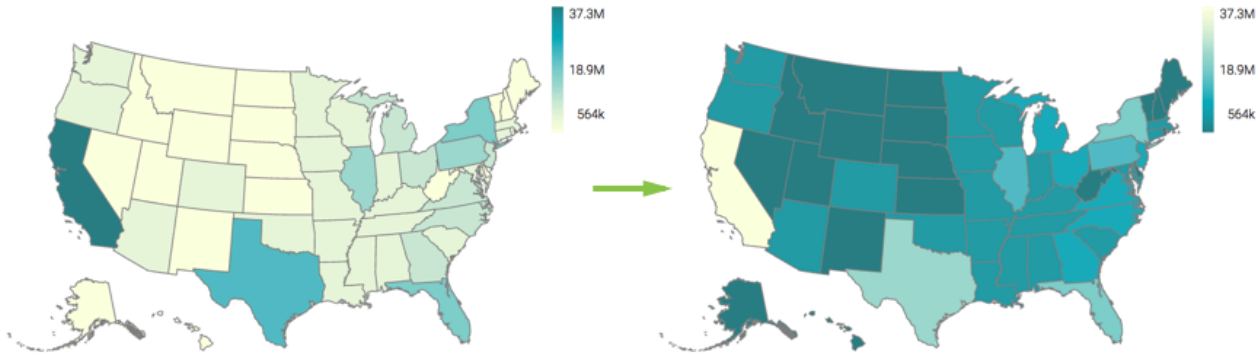

Changing visual color palette

Procedure

Click on a different color palette in the Colors catalog to choose a different color palette for your visualization.

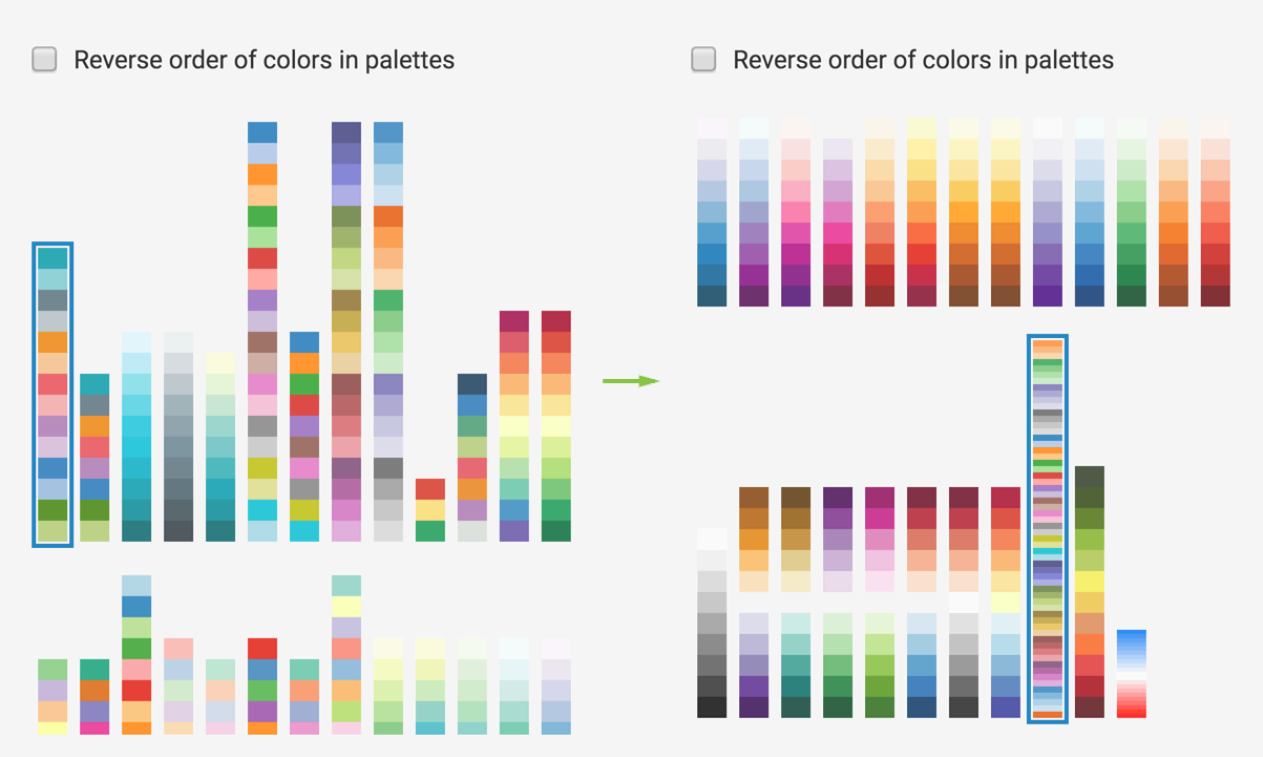

Tip:

The default color of a palette is the top one. To change the color used, click on another one in the palette.

For visuals that only use one color, this is the one that will be used.

For visuals that use categorical colors, the palette will start at this color and wrap around.

Results

The colors in the visual change. The legend is a good indicator of this.

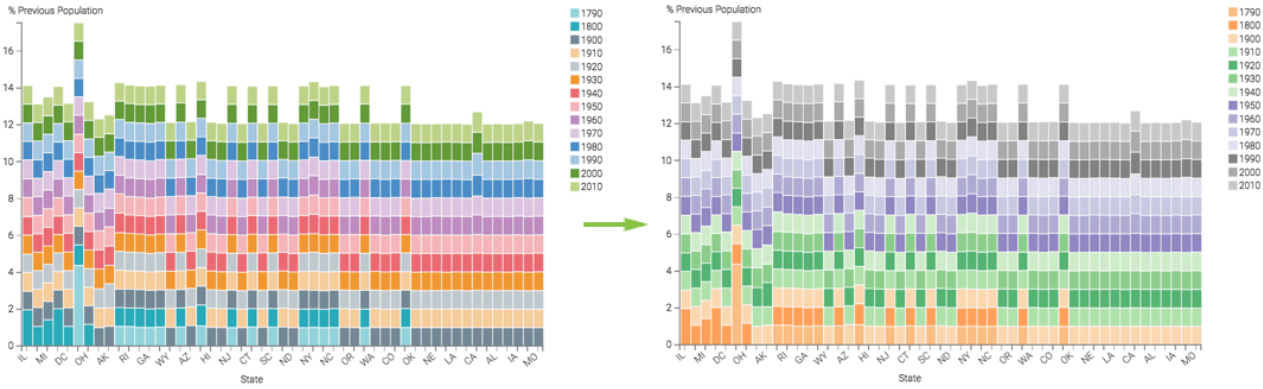

For gauge visuals, this change affects the representation of qualitative ranges.

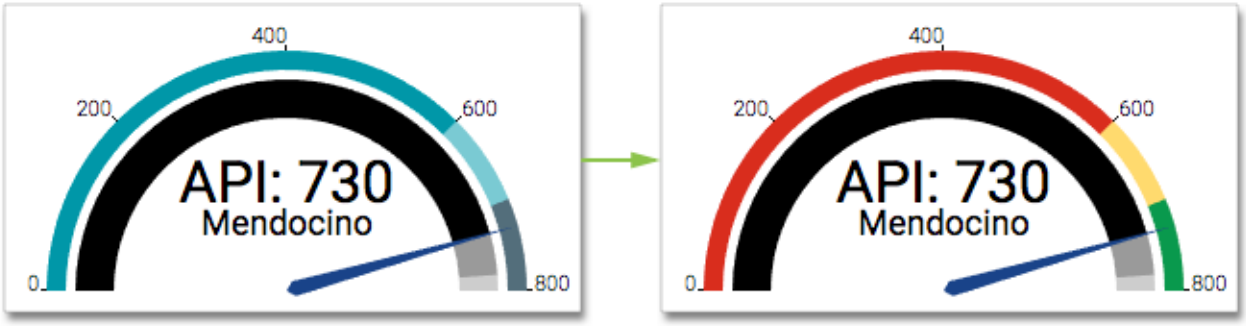

In network visuals, you can specify two separate palettes: one for the nodes, and one for the links:

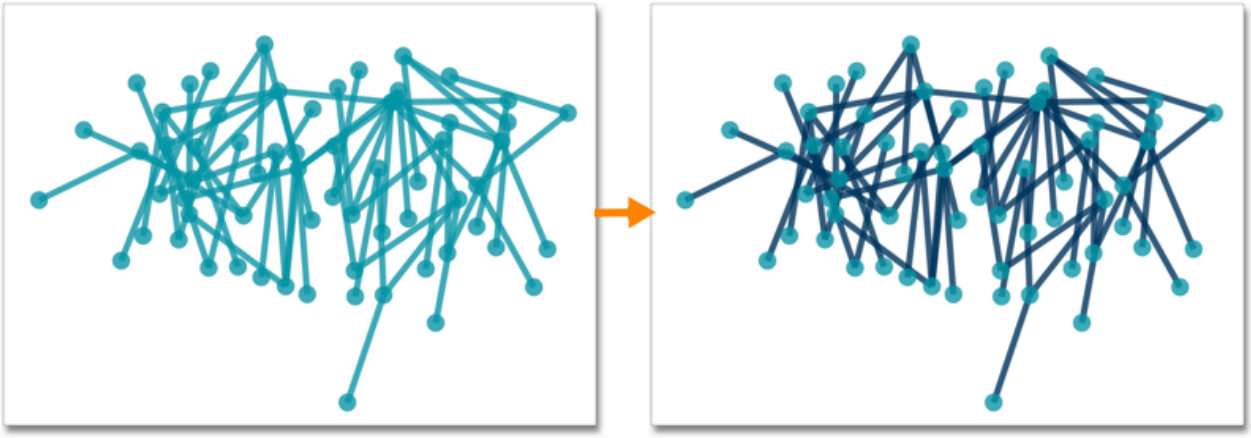

Enabling colors

Procedure

To enable or disable the color option for your visual, such as cross tabulation and check Enable Colors catalog.

Compare the 'enabled-color' cross tabulation visual (default behavior) with the appearance of the 'disabled color' visual.

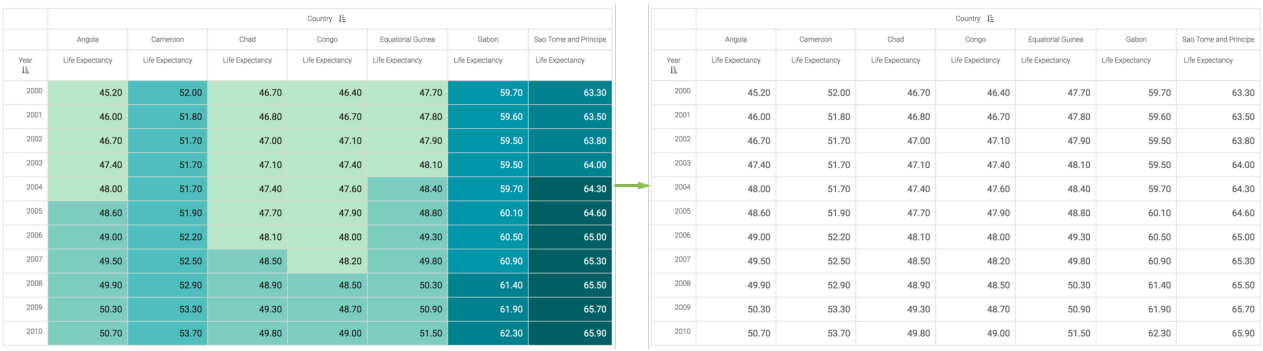

	Country						
	Angola	Cameroon	Chad	Congo	Equatorial Guinea	Gabon	Sao Tome and Principe
Year	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy
2000	45.20	52.00	46.70	46.40	47.70	59.70	63.30
2001	46.00	51.80	46.80	46.70	47.80	59.60	63.50
2002	46.70	51.70	47.00	47.10	47.90	59.50	63.80
2003	47.40	51.70	47.10	47.40	48.10	59.50	64.00
2004	48.00	51.70	47.40	47.60	48.40	59.70	64.30
2005	48.60	51.90	47.70	47.90	48.80	60.10	64.60
2006	49.00	52.20	48.10	48.00	49.30	60.50	65.00
2007	49.50	52.50	48.50	48.20	49.80	60.90	65.30
2008	49.90	52.90	48.90	48.50	50.30	61.40	65.50
2009	50.30	53.30	49.30	48.70	50.90	61.90	65.70
2010	50.70	53.70	49.80	49.00	51.50	62.30	65.90

	Country						
	Angola	Cameroon	Chad	Congo	Equatorial Guinea	Gabon	Sao Tome and Principe
Year	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy	Life Expectancy
2000	45.20	52.00	46.70	46.40	47.70	59.70	63.30
2001	46.00	51.80	46.80	46.70	47.80	59.60	63.50
2002	46.70	51.70	47.00	47.10	47.90	59.50	63.80
2003	47.40	51.70	47.10	47.40	48.10	59.50	64.00
2004	48.00	51.70	47.40	47.60	48.40	59.70	64.30
2005	48.60	51.90	47.70	47.90	48.80	60.10	64.60
2006	49.00	52.20	48.10	48.00	49.30	60.50	65.00
2007	49.50	52.50	48.50	48.20	49.80	60.90	65.30
2008	49.90	52.90	48.90	48.50	50.30	61.40	65.50
2009	50.30	53.30	49.30	48.70	50.90	61.90	65.70
2010	50.70	53.70	49.80	49.00	51.50	62.30	65.90

Changing foreground color

About this task

This applies to Gauge and Bullet visuals.

Procedure

To choose a different color for the value on the Measurement shelf and make changes in the Foreground Color selector. You can select one of the palette colors, or specify a custom color.

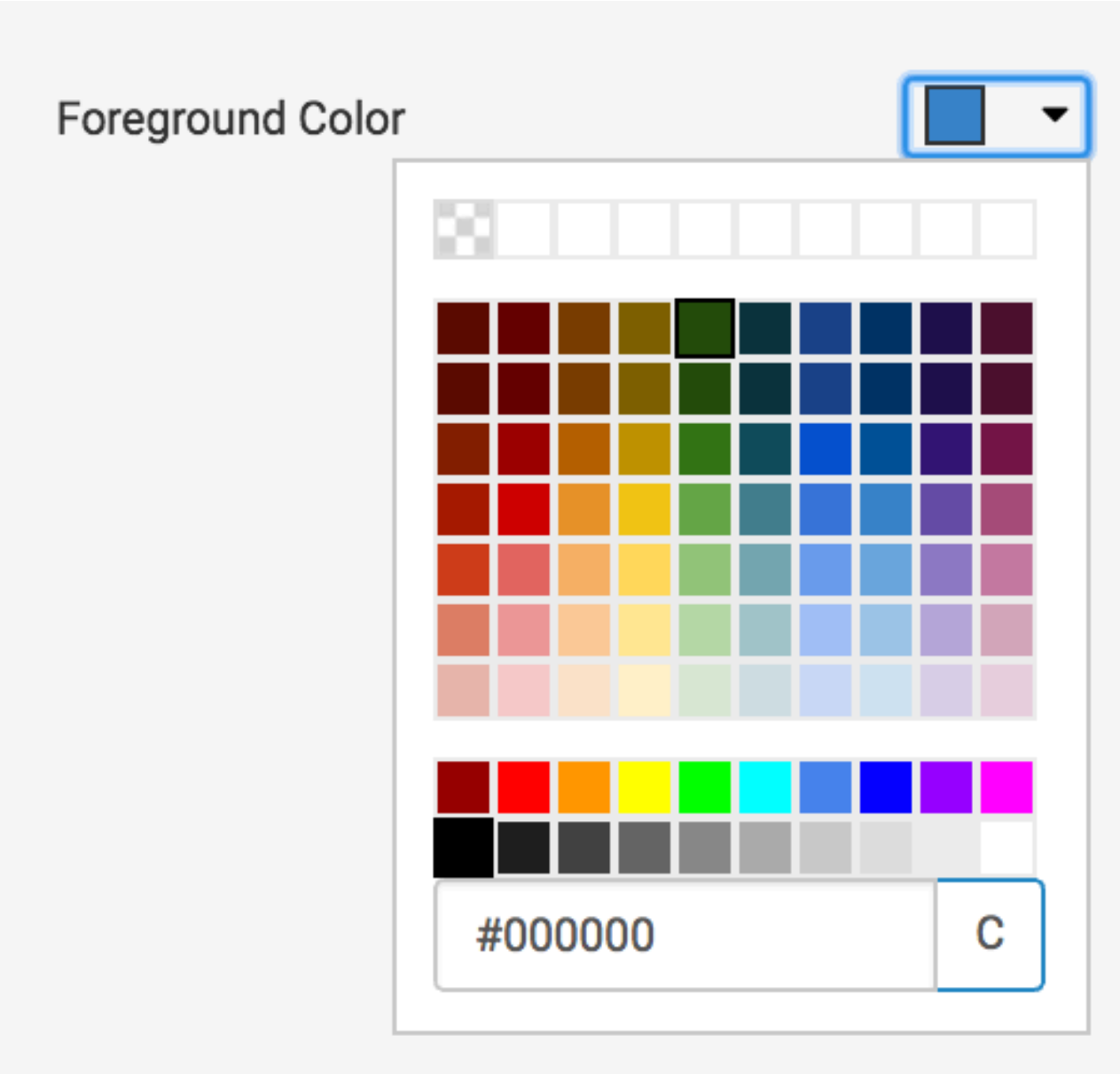

Note how the change in foreground color changes the gauge visual; the main measurement that this option affects is on the inner arc.

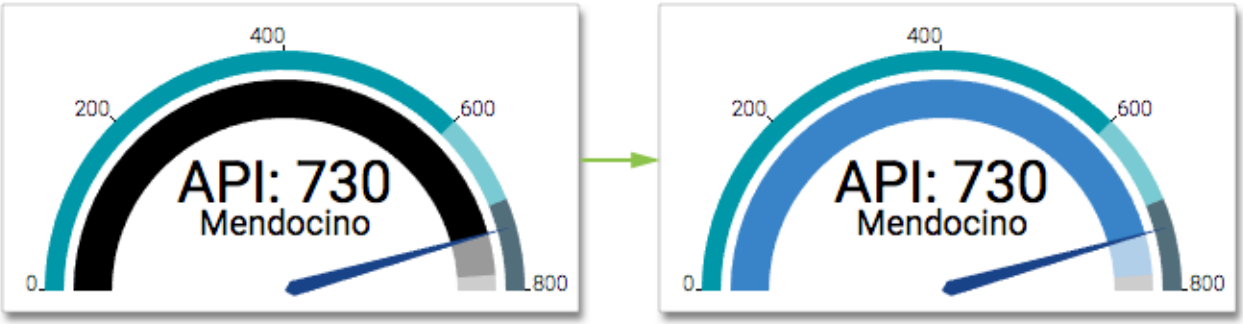

Note how the change in foreground color changes the bullet visual; the main measurement is the narrow central bar.

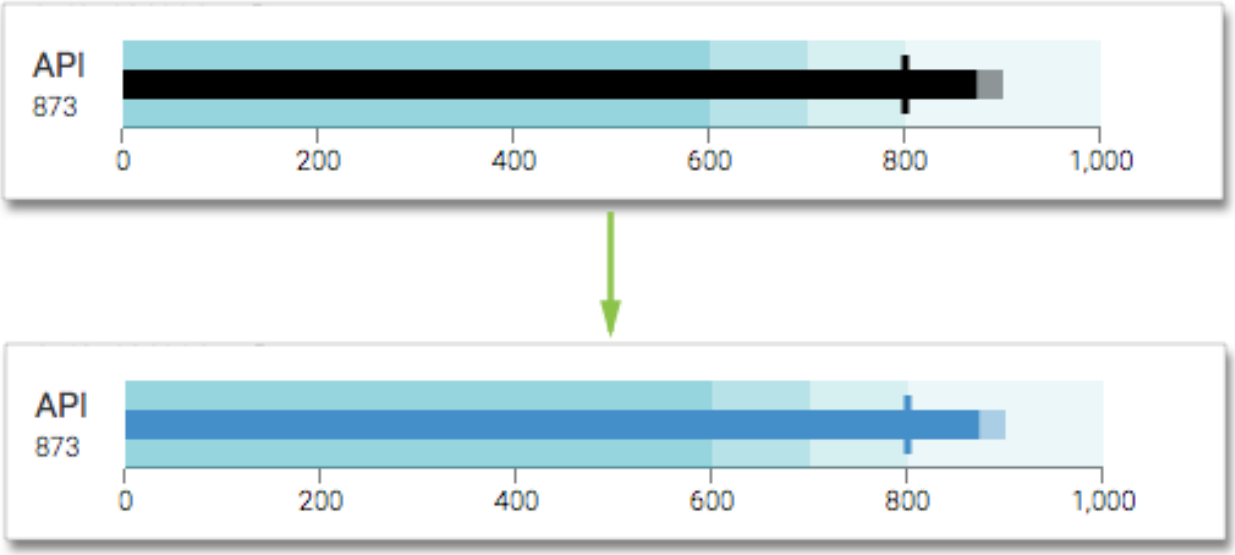

Changing pointer color

About this task

This applies to Gauge visuals.

Procedure

To choose a different color for the pointer on the dial of gauge visuals and make changes in the Pointer Color selector. You can select one of the palette colors, or specify a custom color.

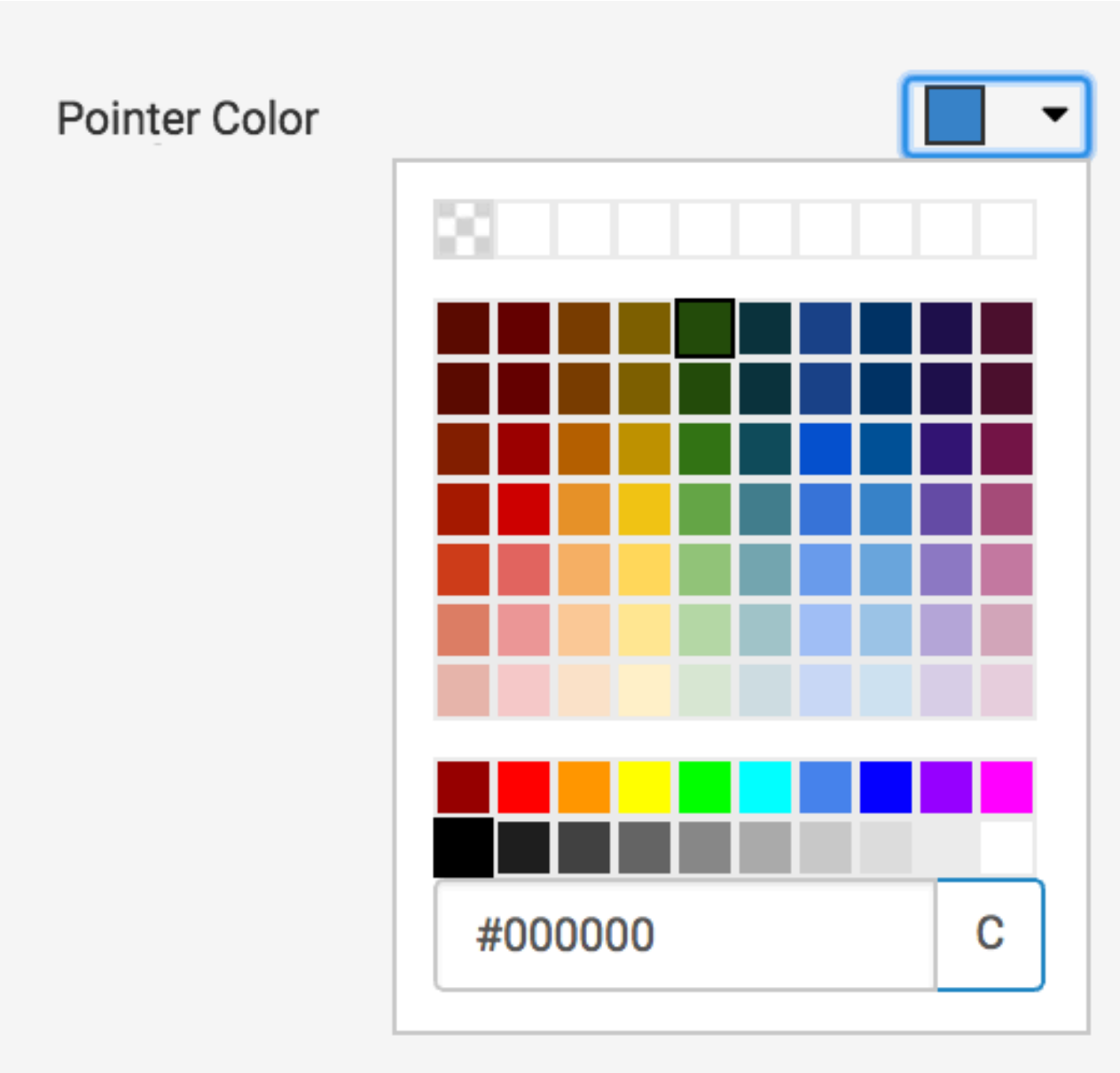

Note how the change in pointer color changes the appearance of the visual.

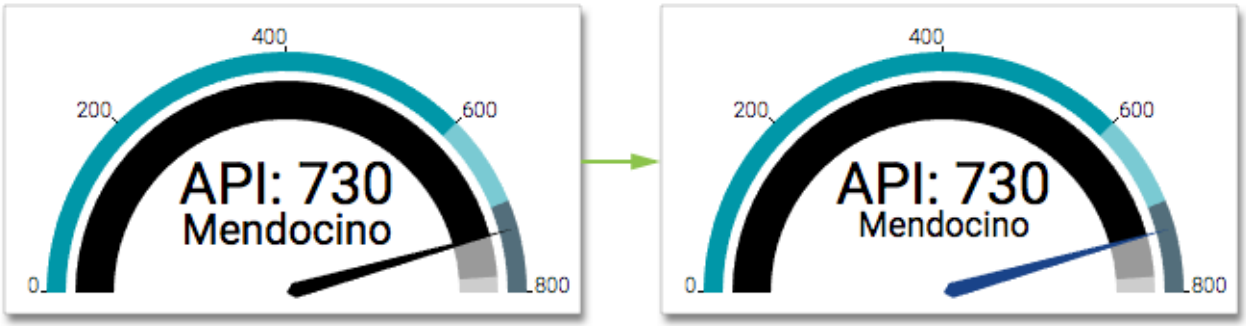

Changing background color

About this task

This applies to Gauge and Bullet visuals.

Procedure

To choose a different color for the background of gauge or bullet visuals and make changes in the Background Color selector. You can select one of the palette colors, or specify a custom color.

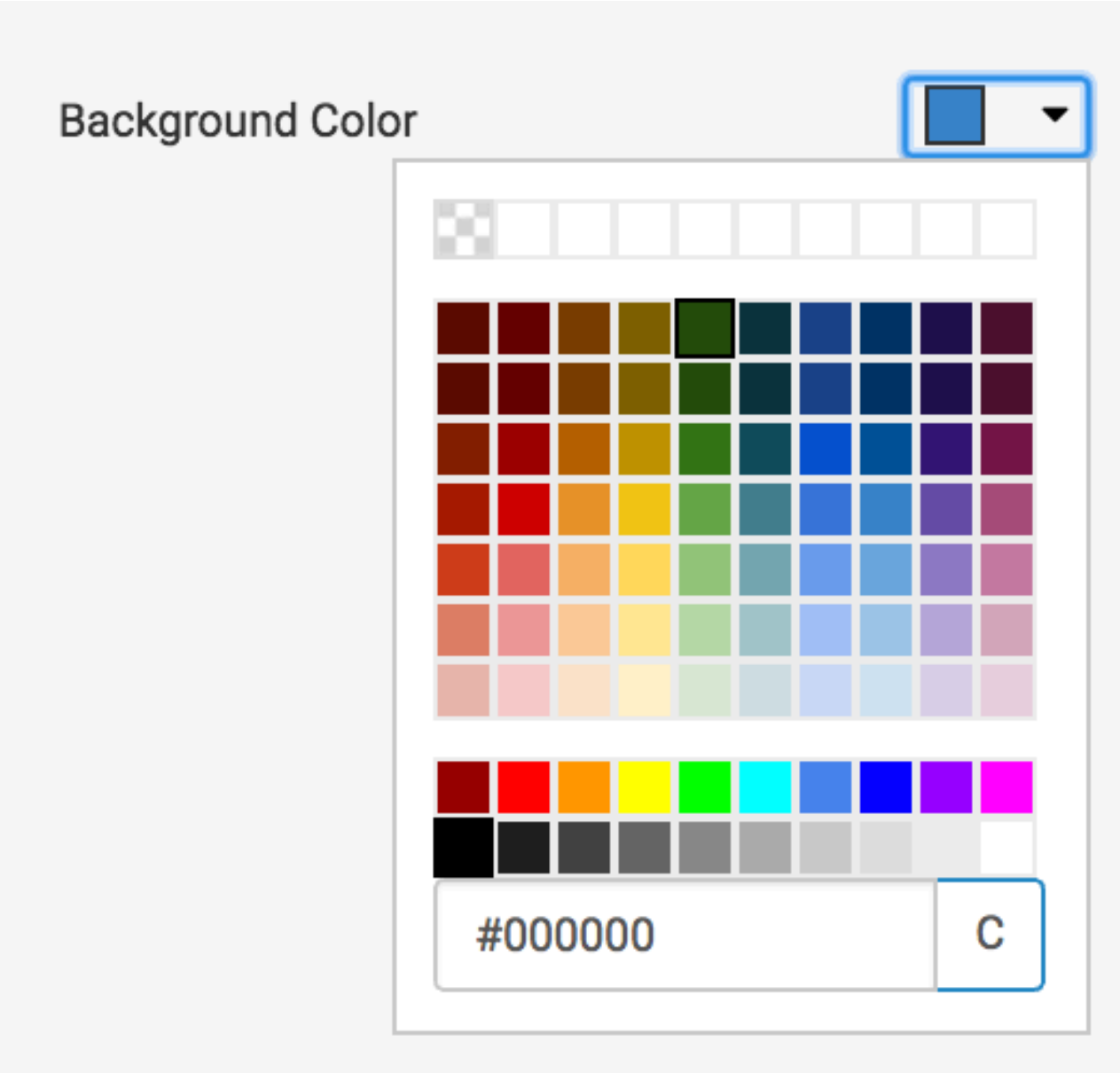

Note how the change in background color changes the gauge visual; this option affects the inner arc, at the at the maximum end. It does not change the appearance of qualitative ranges. To do that, change the color palette.

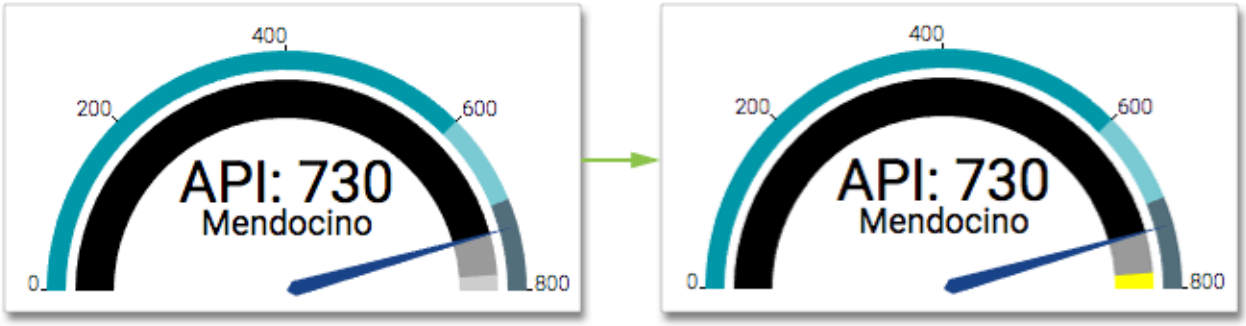

Note how the change in background color changes the bullet visual: in this visual, it controls the appearance of the qualitative ranges.

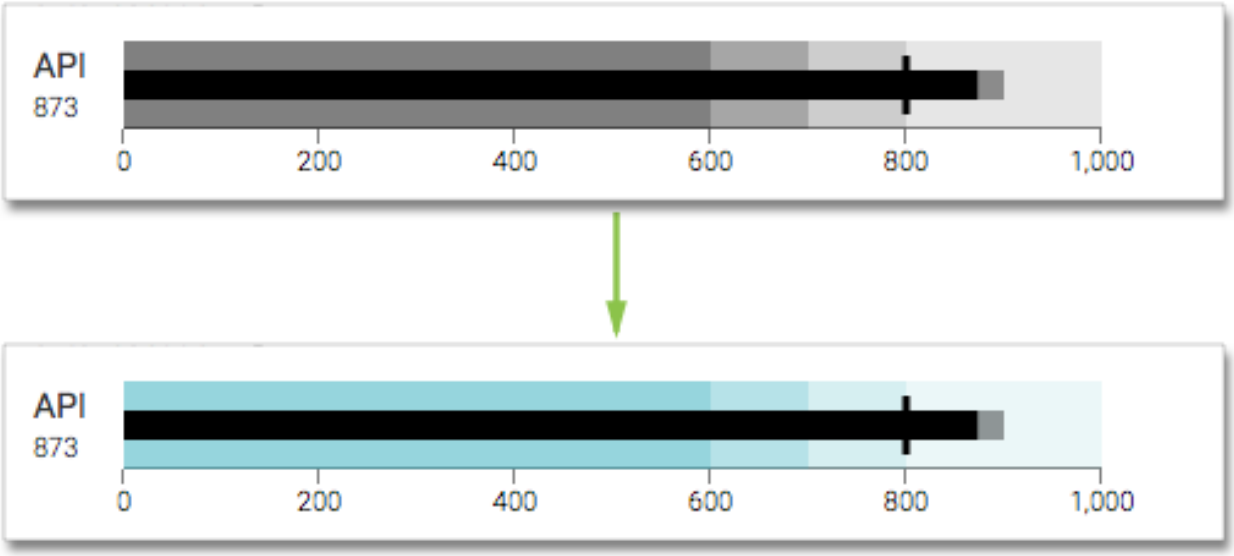

Changing arrow color

Procedure

To choose a different color for the direction arrows, which by default corresponds to the first color in the link color palette, expand the Link Colors menu, and make changes in the Direction arrow color selector. You can select one of the palette colors, or specify a custom color.

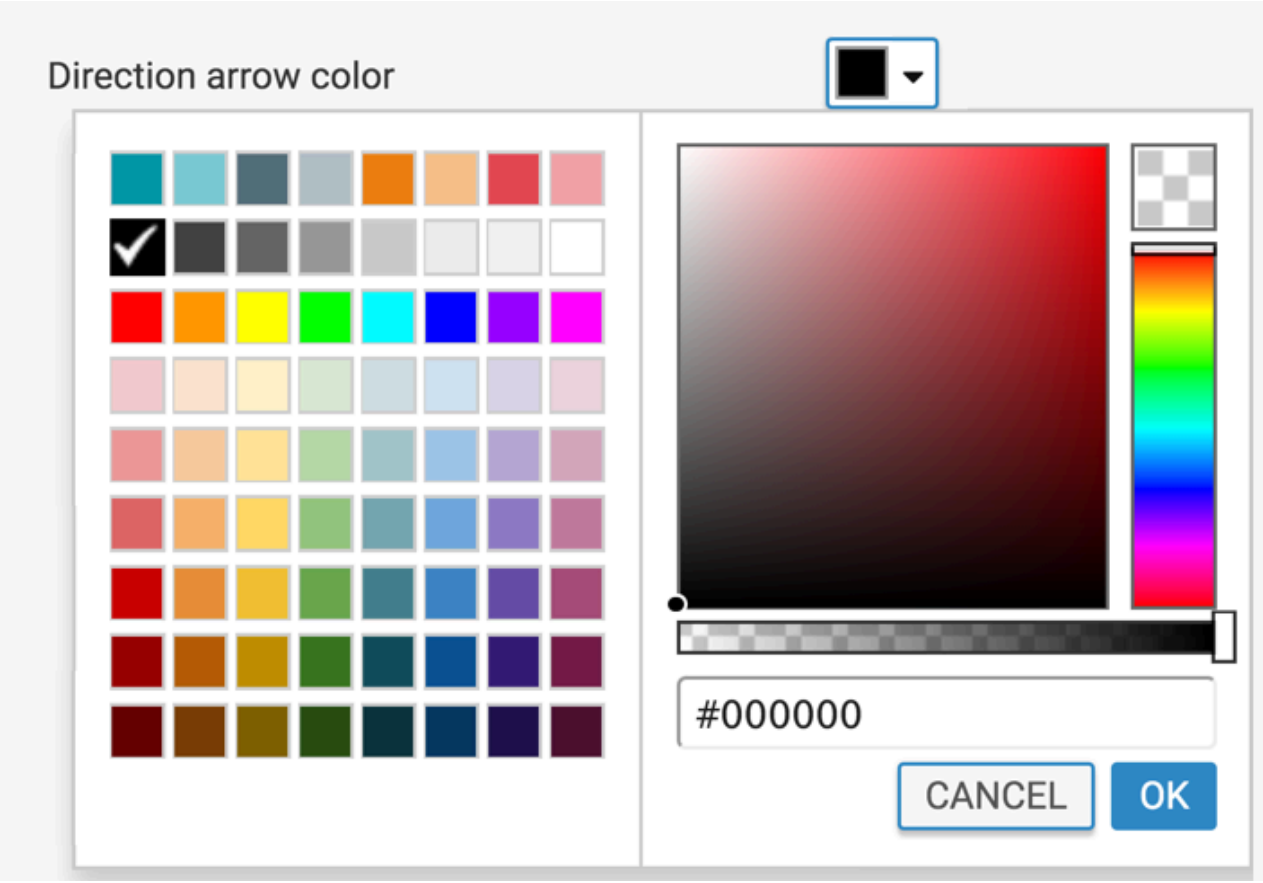

Changing color opacity on visuals

Procedure

To choose a different opacity (color saturation) for your visual and make changes to the Color Opacity selector.

The default value is 80 %. Click the - sign to make the colors more transparent by decreasing color opacity by 10% with each click. To increase color saturation, click the + sign to increase color opacity by 10%. Edit the number in the center of the selector to get a custom value. Valid values range from 1% for the palest through 100% for the brightest colors. A value of 0 generates a visual with full saturation.

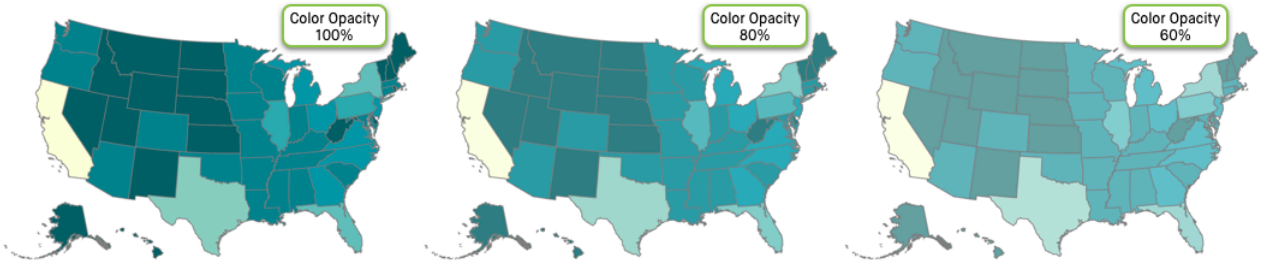

Changing link color

Procedure

To choose a different color for the link between nodes, such as in flow visuals, click the Link Colors color selector widget, select one of the palette colors or specify a custom color, and click OK.

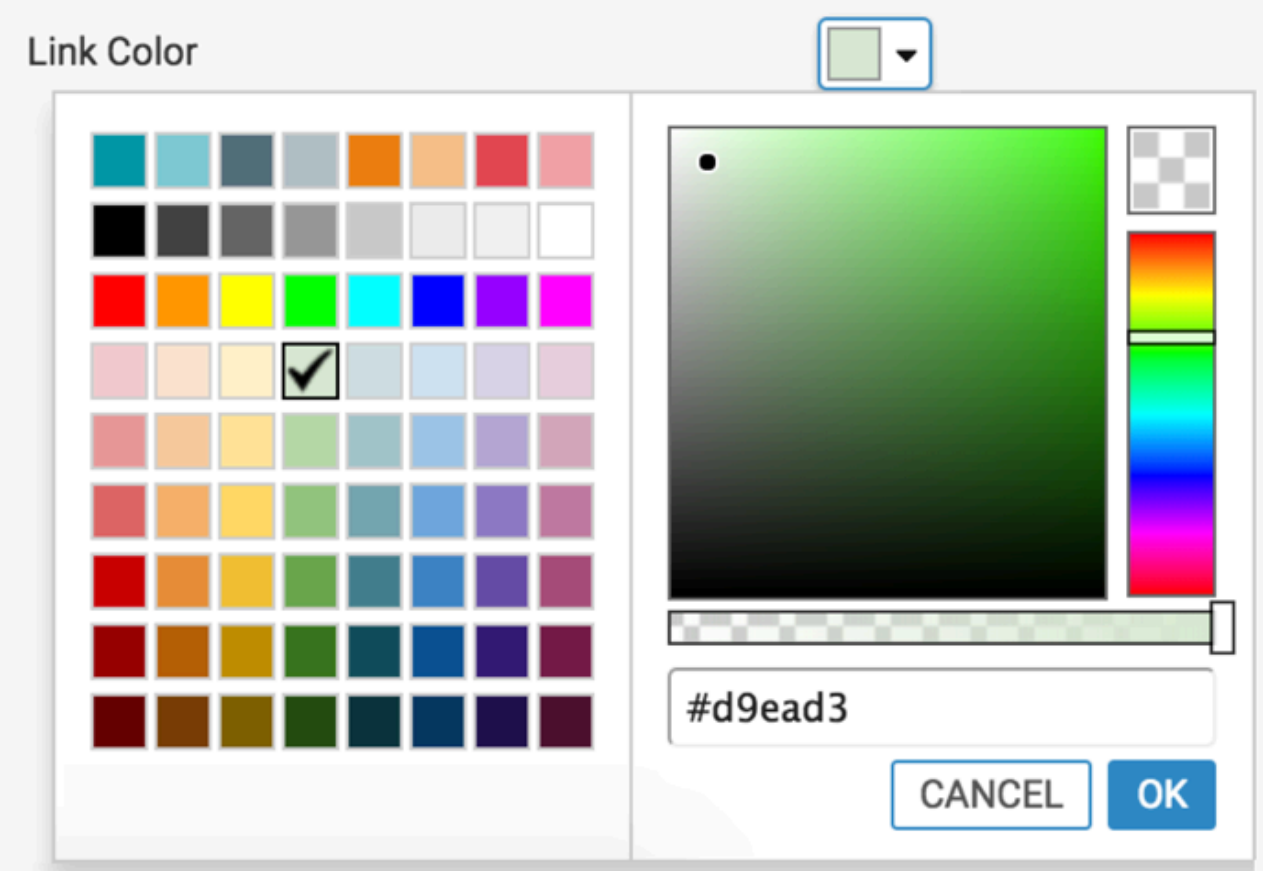

The following image illustrates how a flow visual changes when we change the link color.

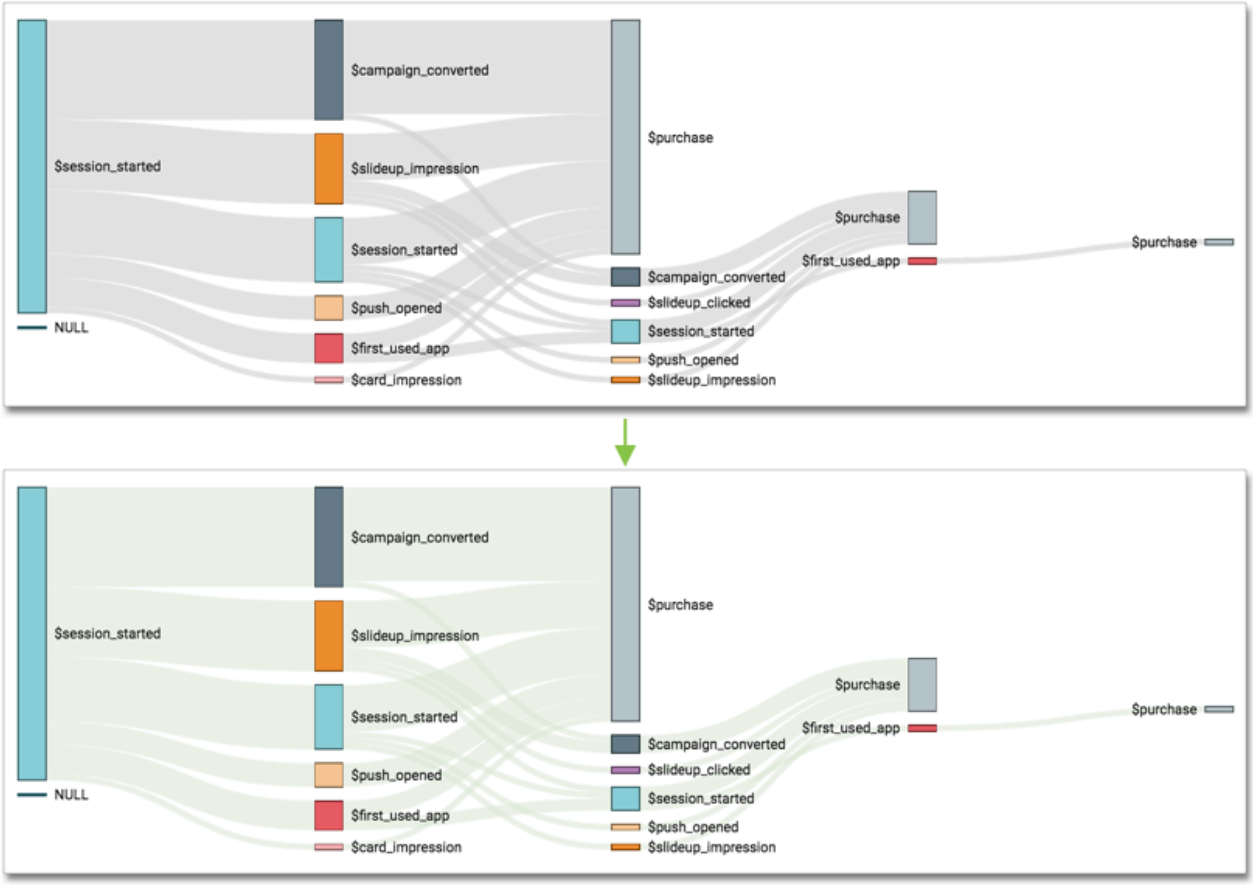

Changing drop off color

Procedure

To choose a different color for the drop off node, such as in funnel visuals, click the Drop off Color color selector widget, select one of the palette colors or specify a custom color, and click OK.

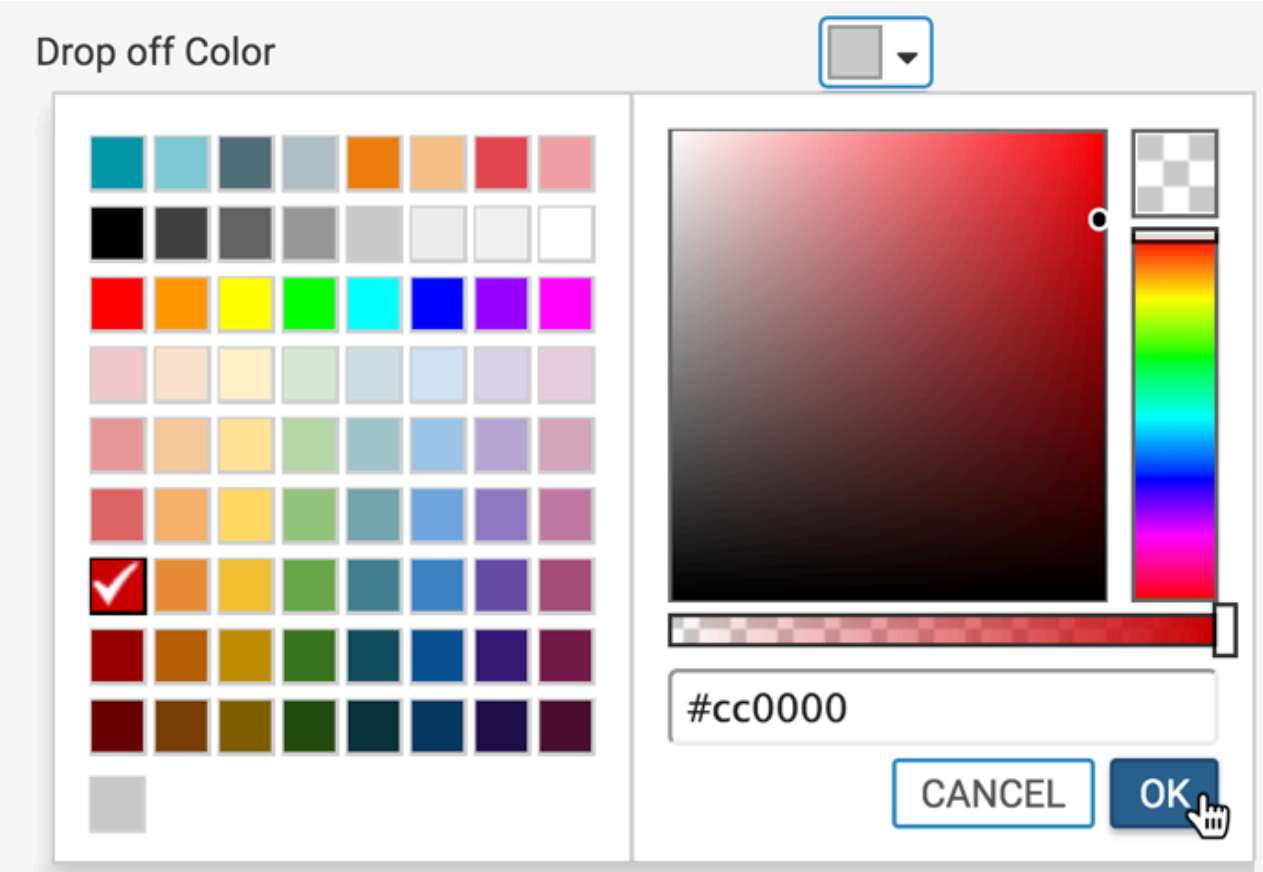

The following image illustrates how a funnel visual changes when we change the drop off color.

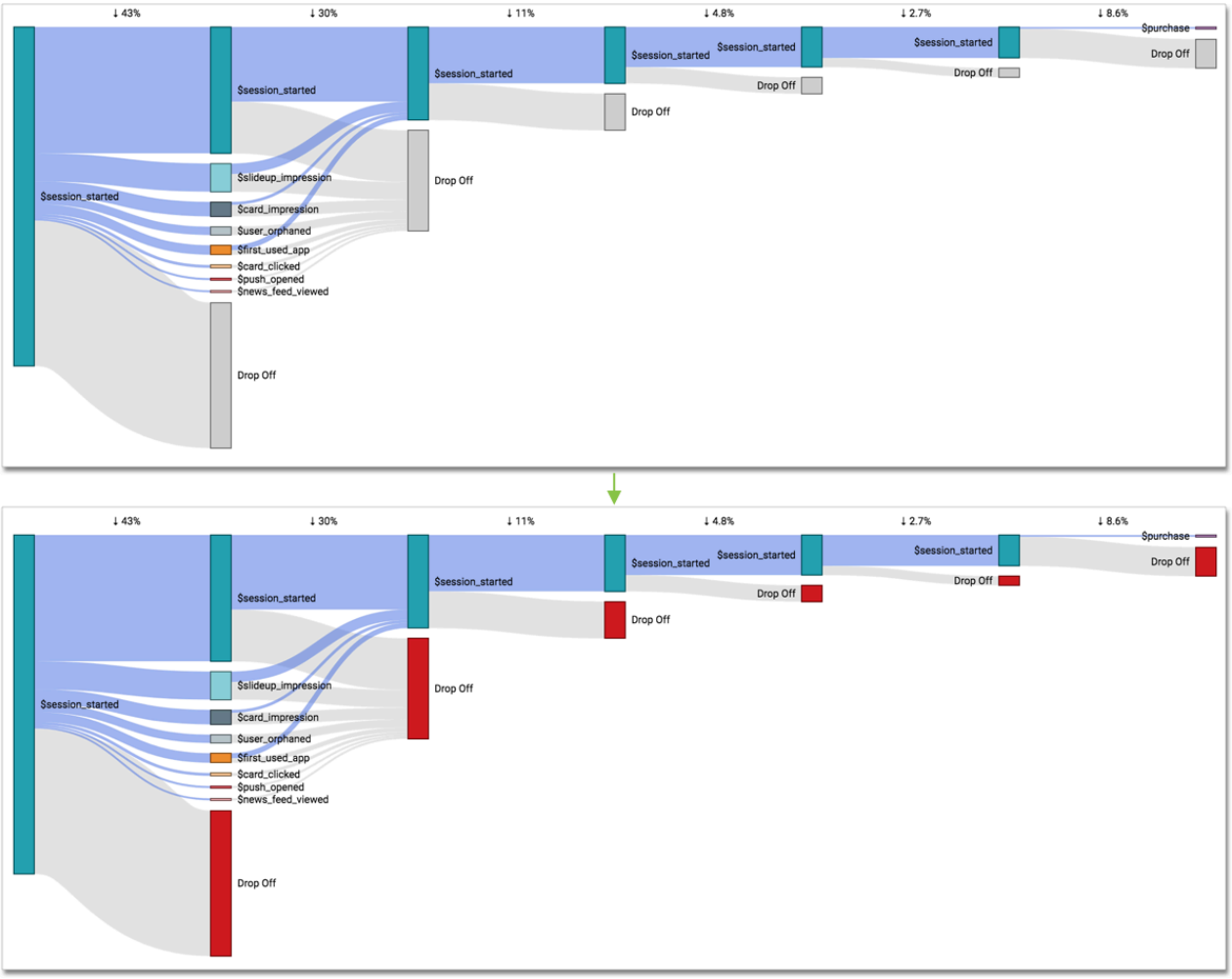

Changing highlight color

Procedure

To choose a different color for the highlight on the link of a visual, such as in flow or funnel visuals, click the Highlight Color color selector widget, select one of the palette colors or specify a custom color, and click OK.

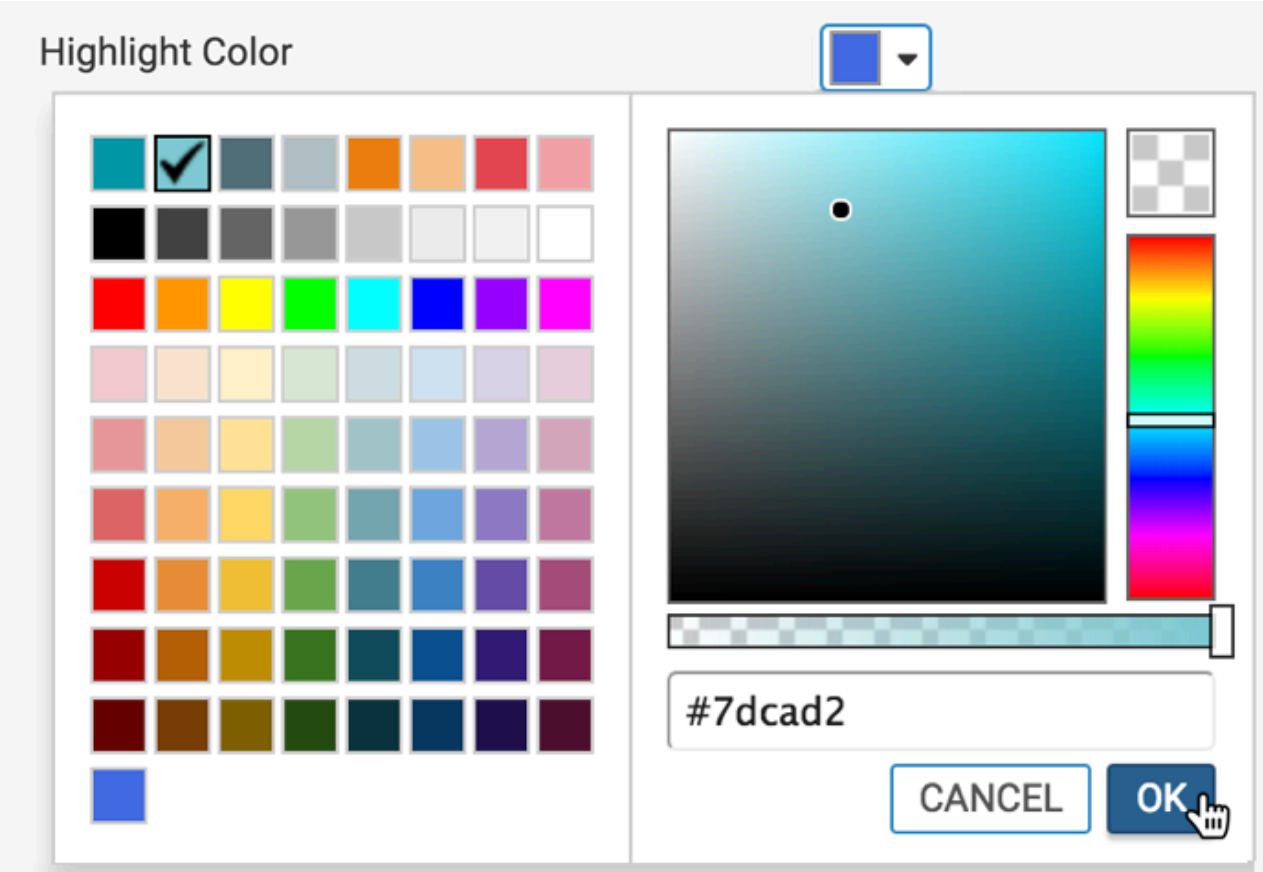

The following image illustrates how a funnel visual changes when we change the highlight color.

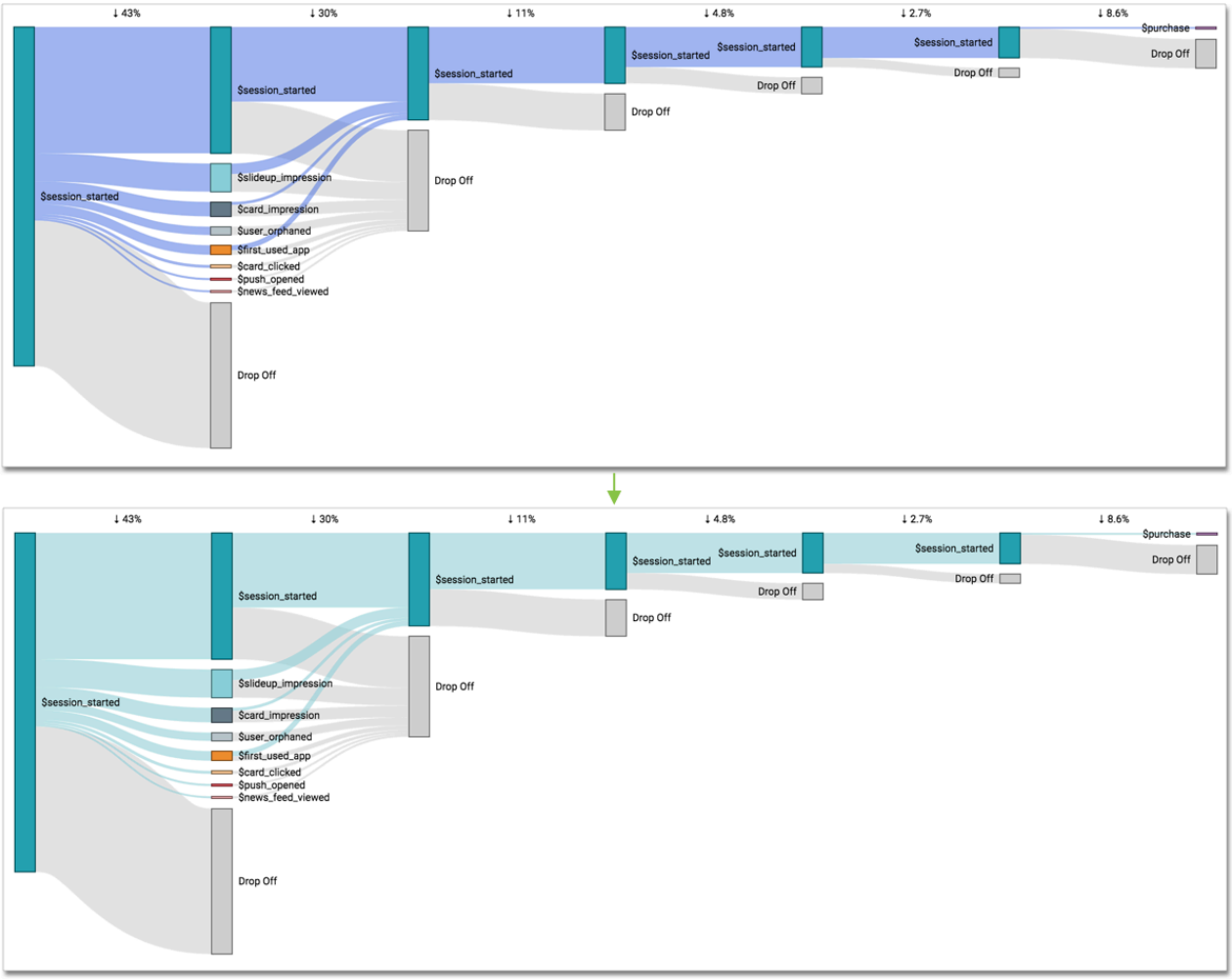

Changing projection opacity

About this task

This applies to Gauge and Bullet visuals.

Procedure

To choose a different opacity (color saturation) for the projection indicator of your visual and make changes to the Projection Opacity selector.

For example, examine the differences in the appearance of the projection for the bullet visual

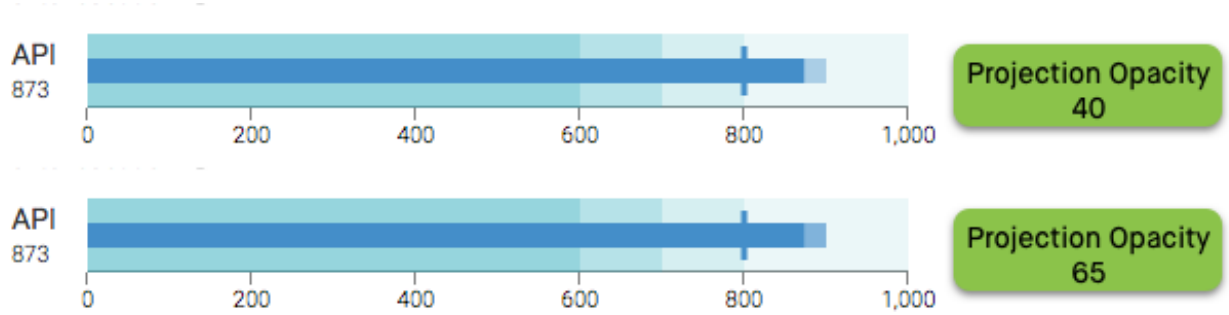

Remember that the projection appears on the inner arc of a gauge visual.

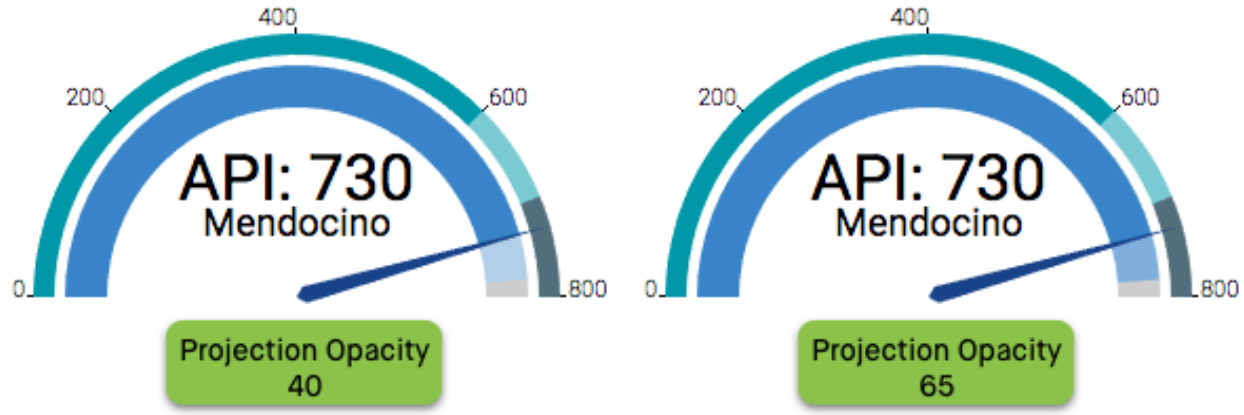

Changing background opacity

About this task

This applies to Gauge visuals.

Procedure

To choose a different opacity (color saturation) for the background of the measurement arc in gauge visuals and make changes to the Background Opacity selector. The default value is 20.

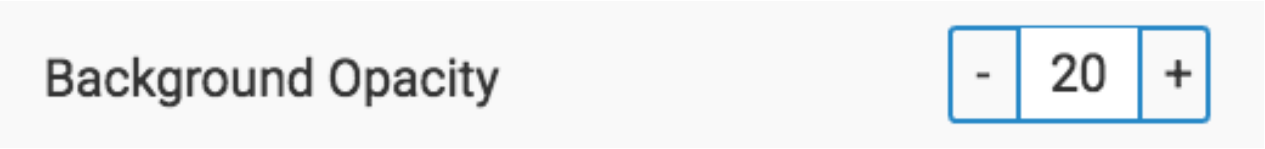

For example, examine the differences here in the appearance of the background for the gauge visual. Note also that because the projection color is semi-transparent, a 'stronger' background also changes the appearance of the projection color.

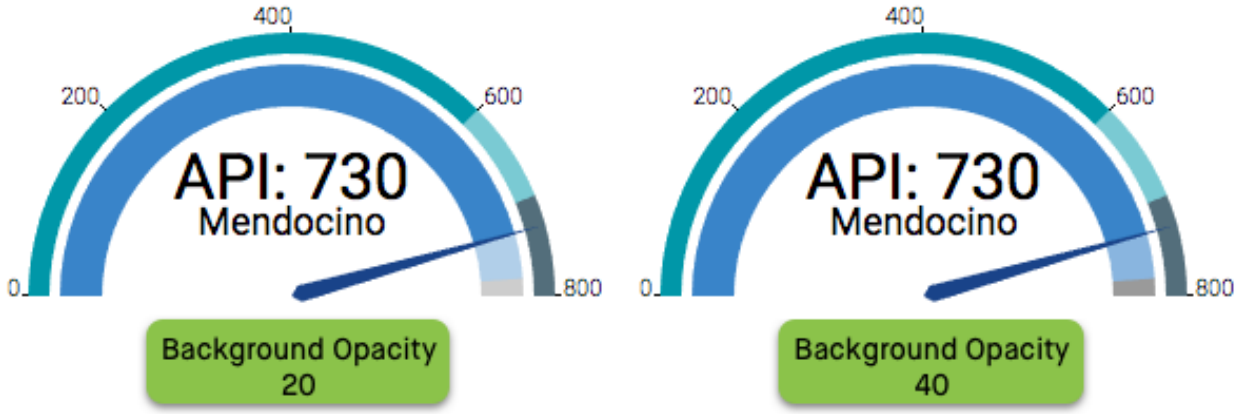

Ensuring consistent colors on dimension values

Cloudera Data Visualization enables you to make colors consistent on dimension values. In dimensions where several visuals include same dimension values, it is helpful if these values are colored identically. It also removes the need to add multiple legends on the same dashboard.

About this task

This feature applies to bar visuals, line visuals, area visuals, combined bar/line visuals, and pie visuals.

For this feature to work properly, you must ensure that the target visuals use the same color palette.

Procedure

To ensure that the colors are consistent across the visuals when they appear in the same dashboard and select the option Base colors on dimension values. You must do this for each visual in the dashboard.

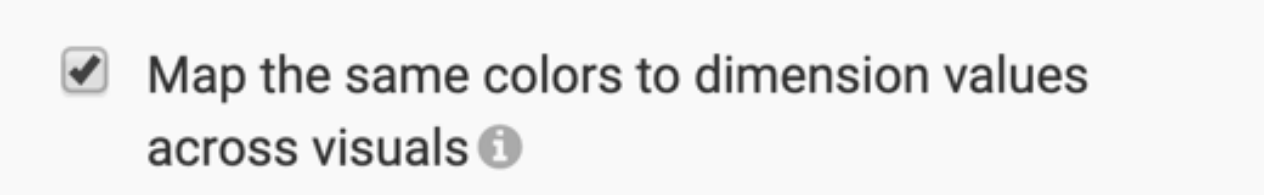

Example

In the following example, we show how to standardize colors for a US State Population dashboard that uses a bar visual and a trellised line visual.

Compare the color values for the two visuals in the top graph. It can be easily seen that the legends for the visuals don't match. However, they are identical in the bottom dashboard, where we enabled the color consistency feature.

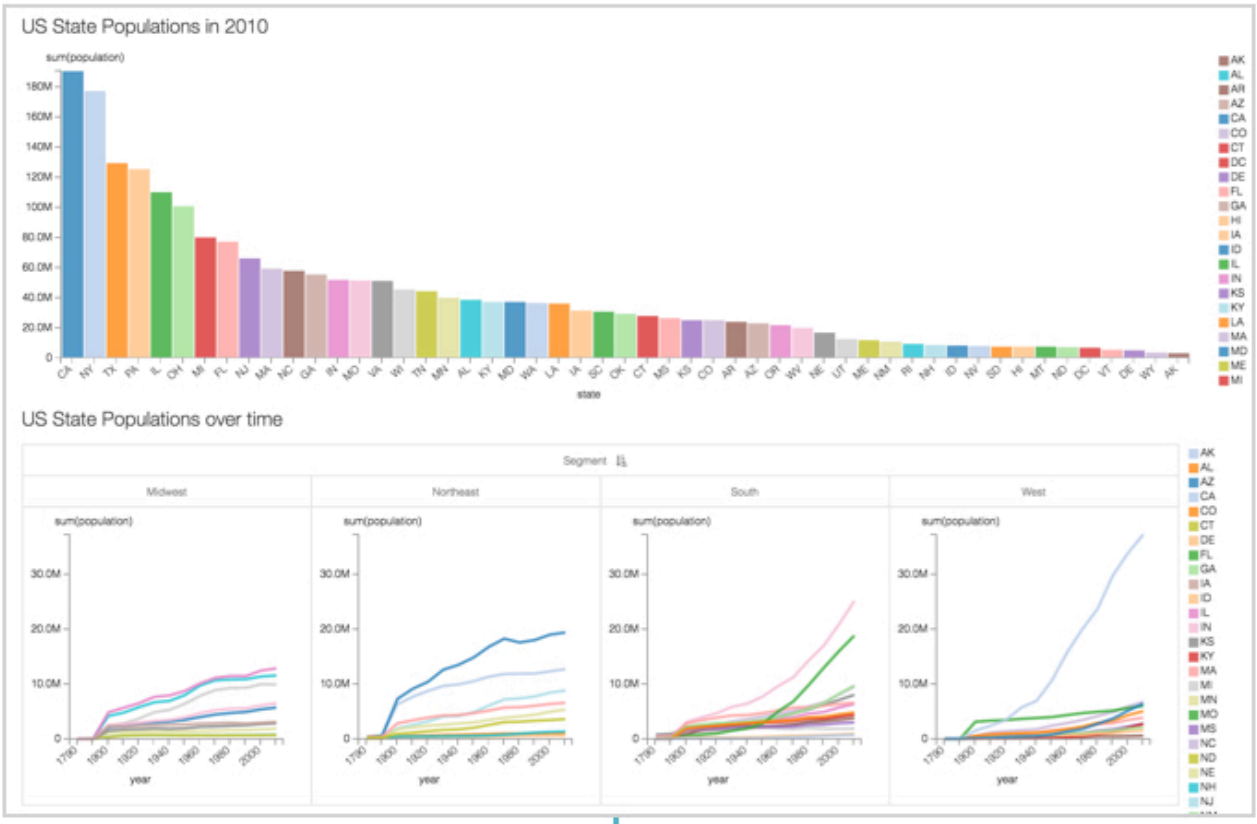
US State Populations in 2010
sum(population)
state
US State Populations over time
Segment
Midwest
Northeast
South
West
year

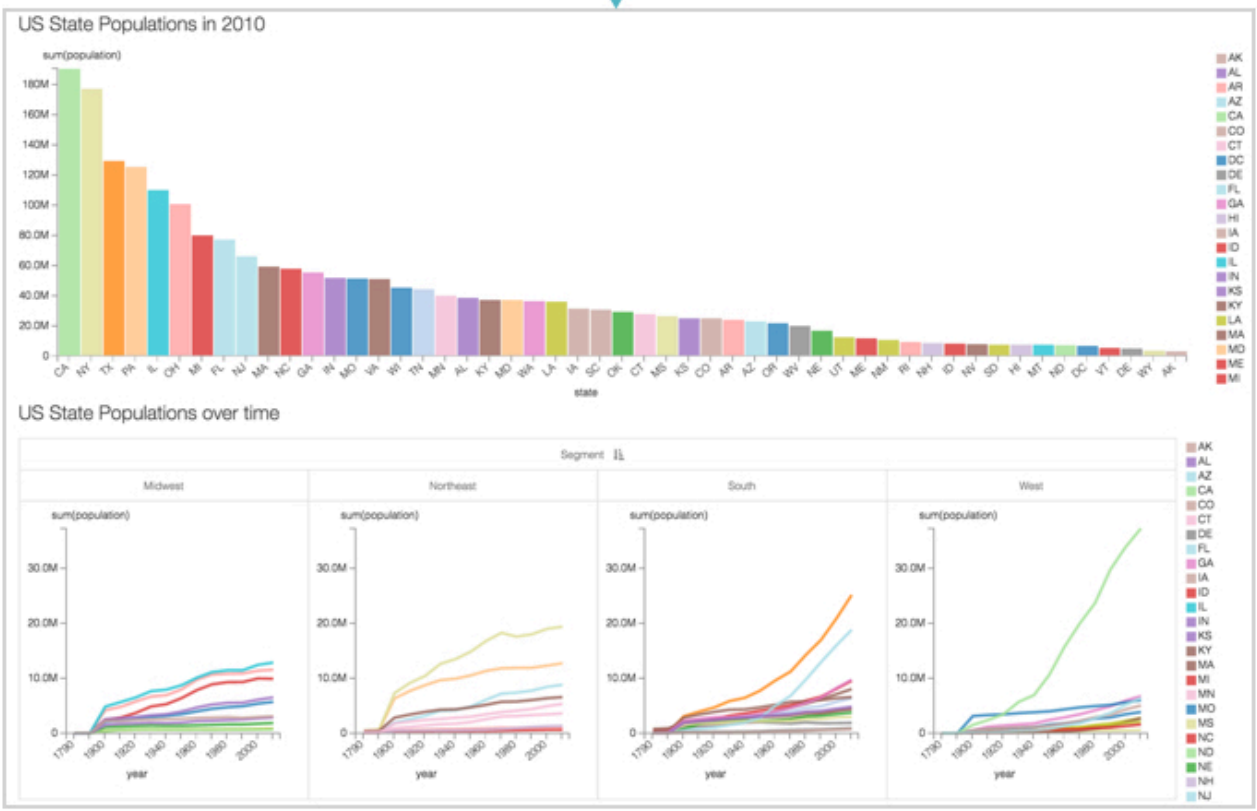
US State Populations in 2010
sum(population)
state
US State Populations over time
Segment
Midwest
Northeast
South
West
year

Assigning color to specified values

About this task

Mapping colors to a specific dimension's values enables you to choose and modify a meaningful representation of values, such as 'green | red | amber' for 'success | failure | soft-fail'.

This feature applies to all visual types that use the Colors shelf.

Procedure

1. To adjust the colors, select the option Fix colors for specified values.

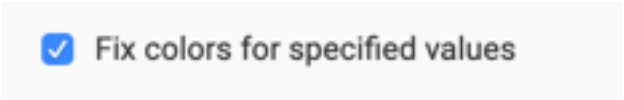

2. In the colors that appear for each value, make changes.

   You can ensure that these colors are used consistently across all visuals within a dashboard by combing this feature with Base colors on dimension values, For details, see *Consistent colors on dimension values*.

   If the color you want to use is not in any of the palettes available to you, you can define it through the Custom Colors interface.

Example

For example, in a grouped bar visual that shows values for Female and Male segments, we can adjust colors in the following manner:

1. Below the selected option Fix colors for specified values, click on the color you want to change. Let's change the color for Females.

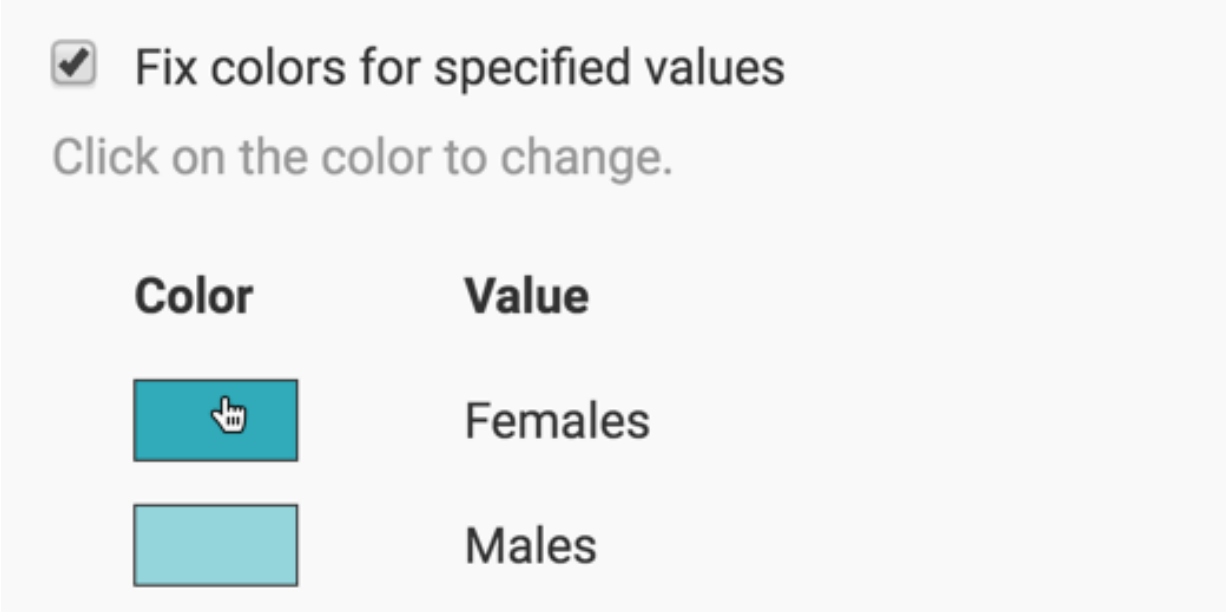

2. In the expanded palette, select the new color. Here, we are changing the color for Females from dark turquoise to mauve.

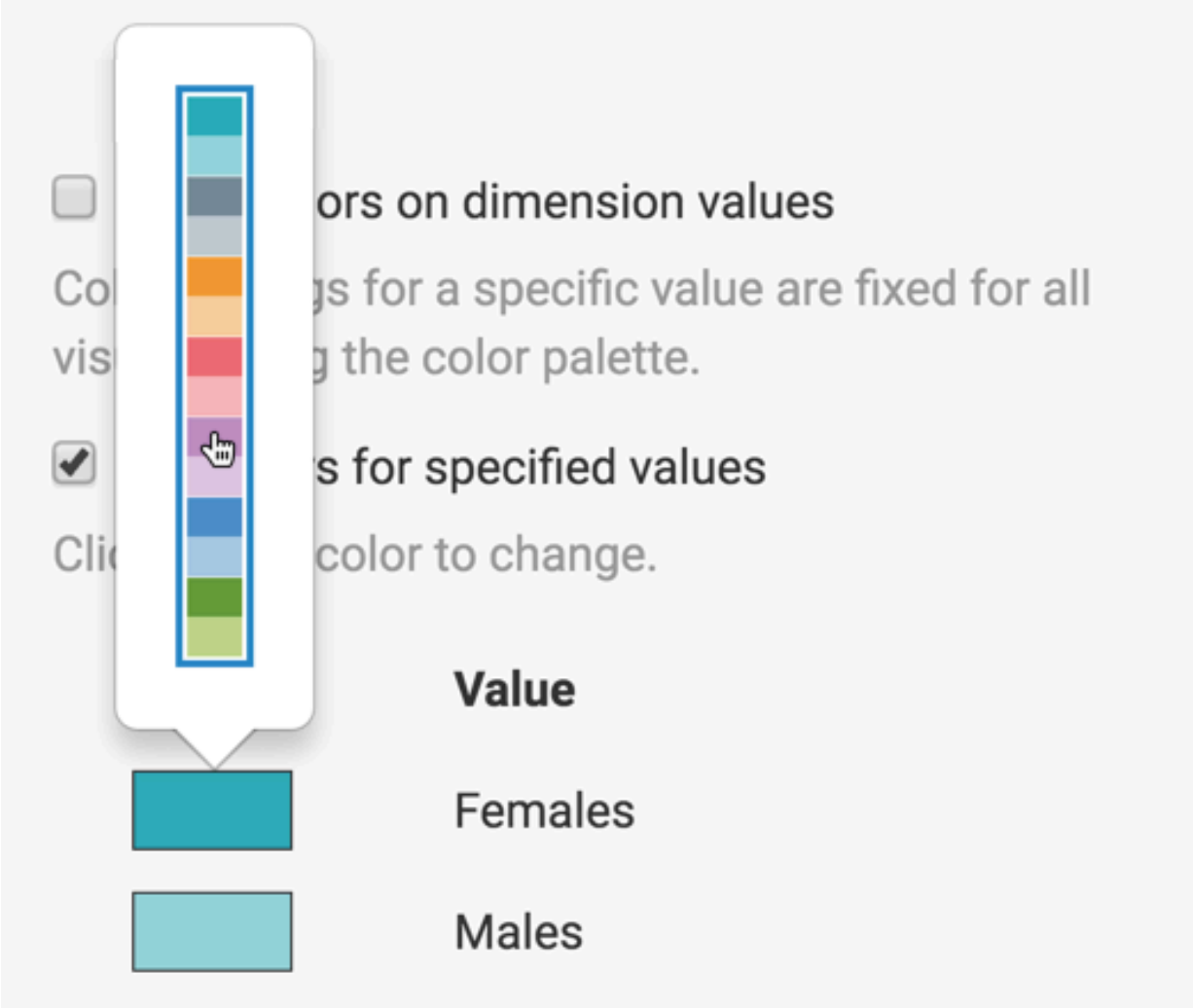

3. Similarly, change the color for Males from light turquoise to dark turquoise.

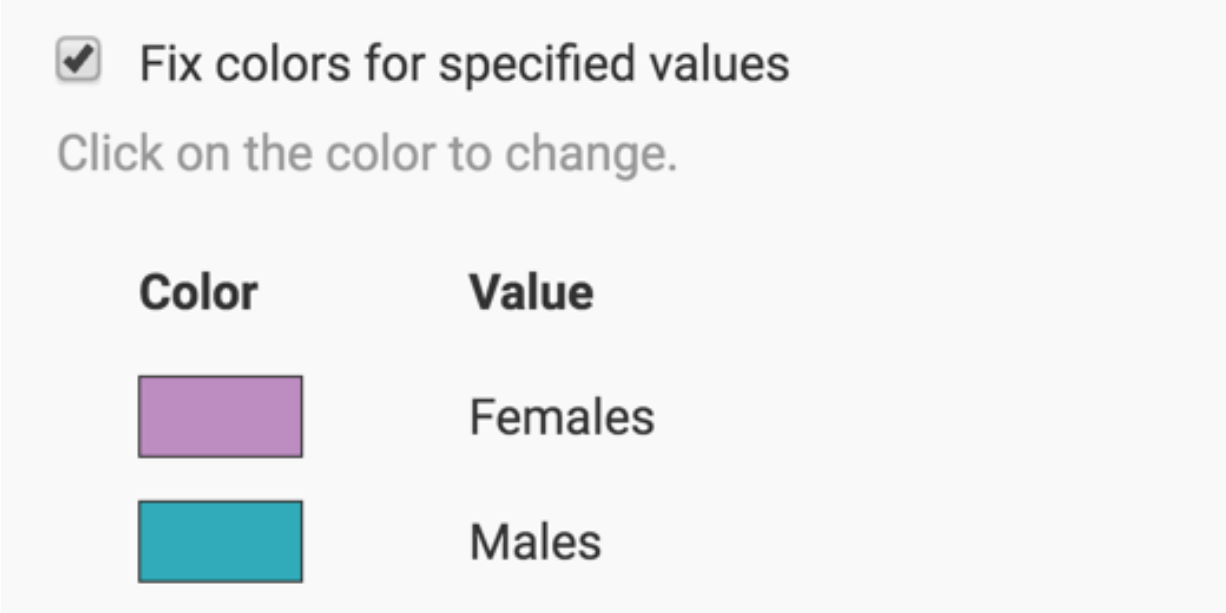

4. This is how the color indicators appear after your make the changes:

After refreshing the visual, notice the change to its appearance:

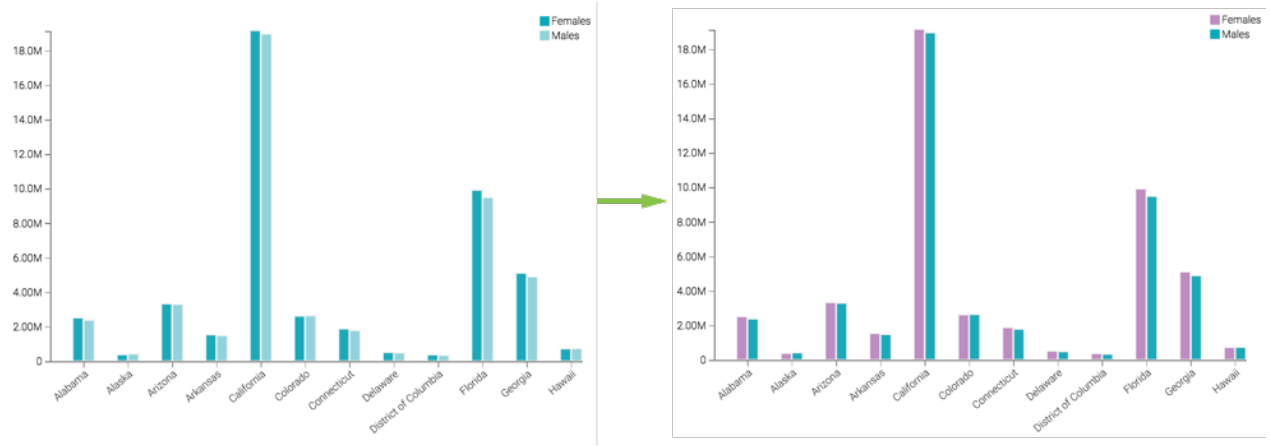

Related Information

Ensuring consistent colors on dimension values

Changing colors for KPI visuals

About this task

This applies to KPI visuals.

Click the Style menu on the right side of the Dasboard Designer.

Click Colors to open the drop down menu.

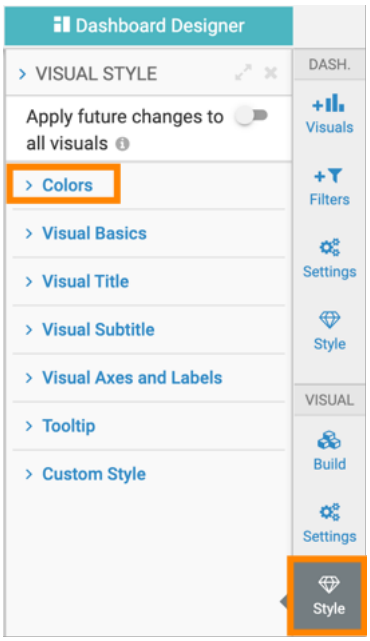

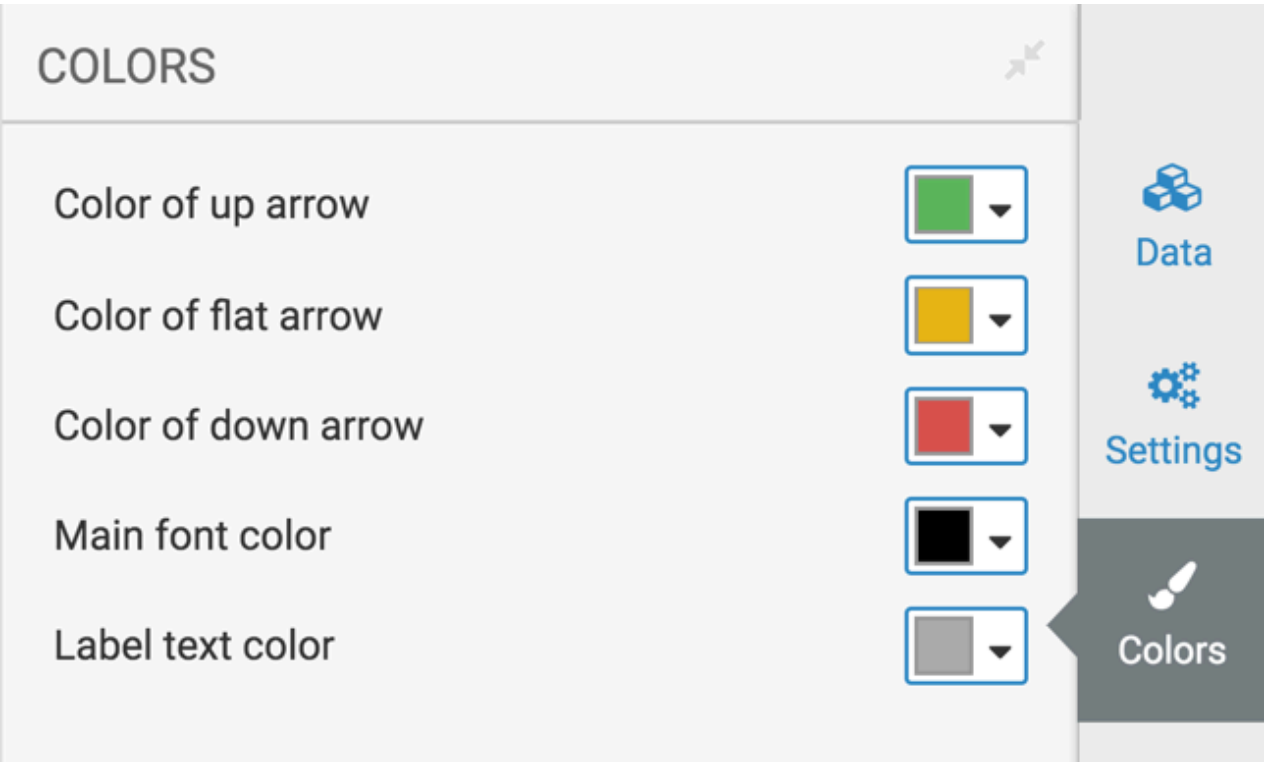

Changing the color of the up arrow

To choose a different color for the up arrow of the visual (the relative difference between the main value and the compare value), make changes in the Color of up arrow selector. You can select one of the palette colors, or specify a custom color.

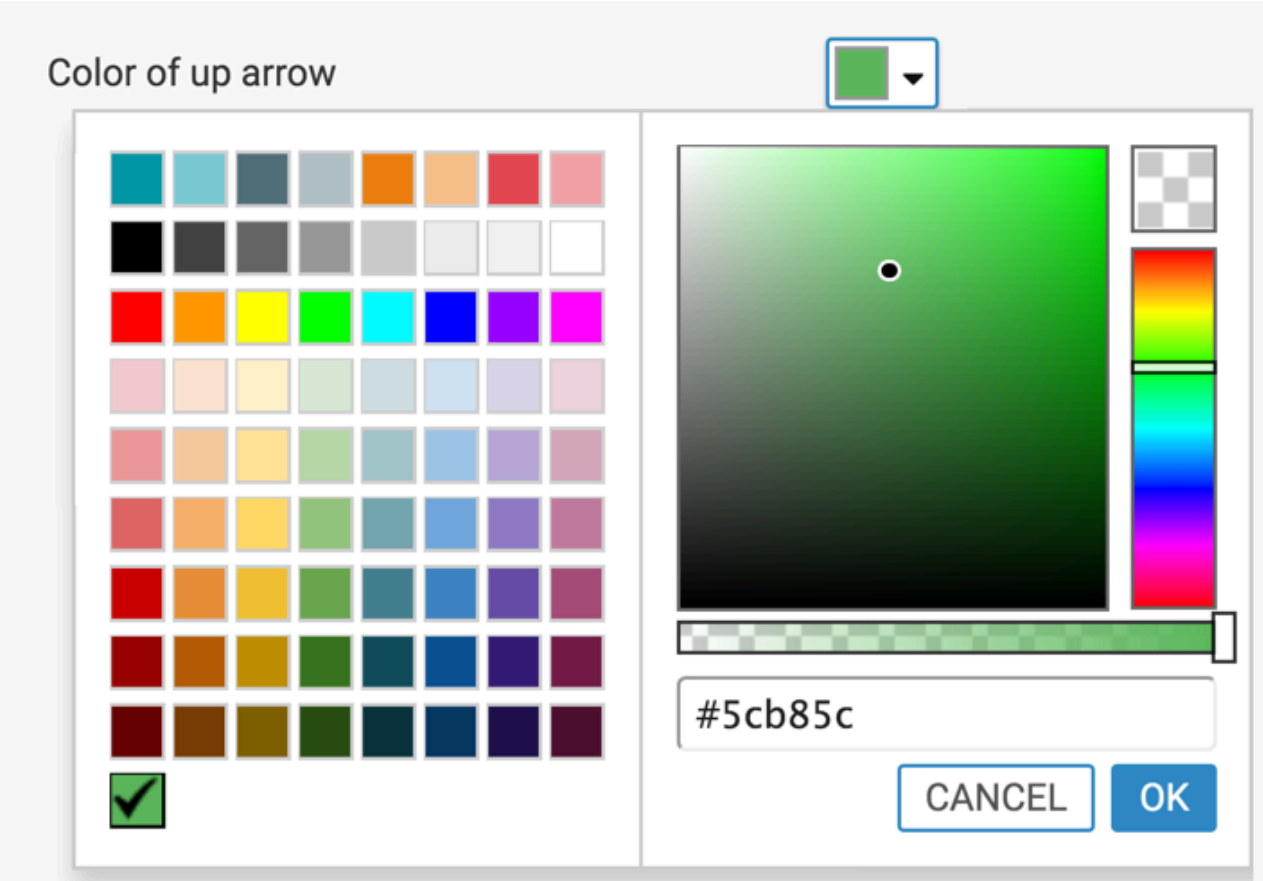

Changing the color of the flat arrow

To choose a different color for the flat arrow of the visual (when there is no difference between the main and comparison values), make changes in the Color of flat arrow selector. You can select one of the palette colors, or specify a custom color.

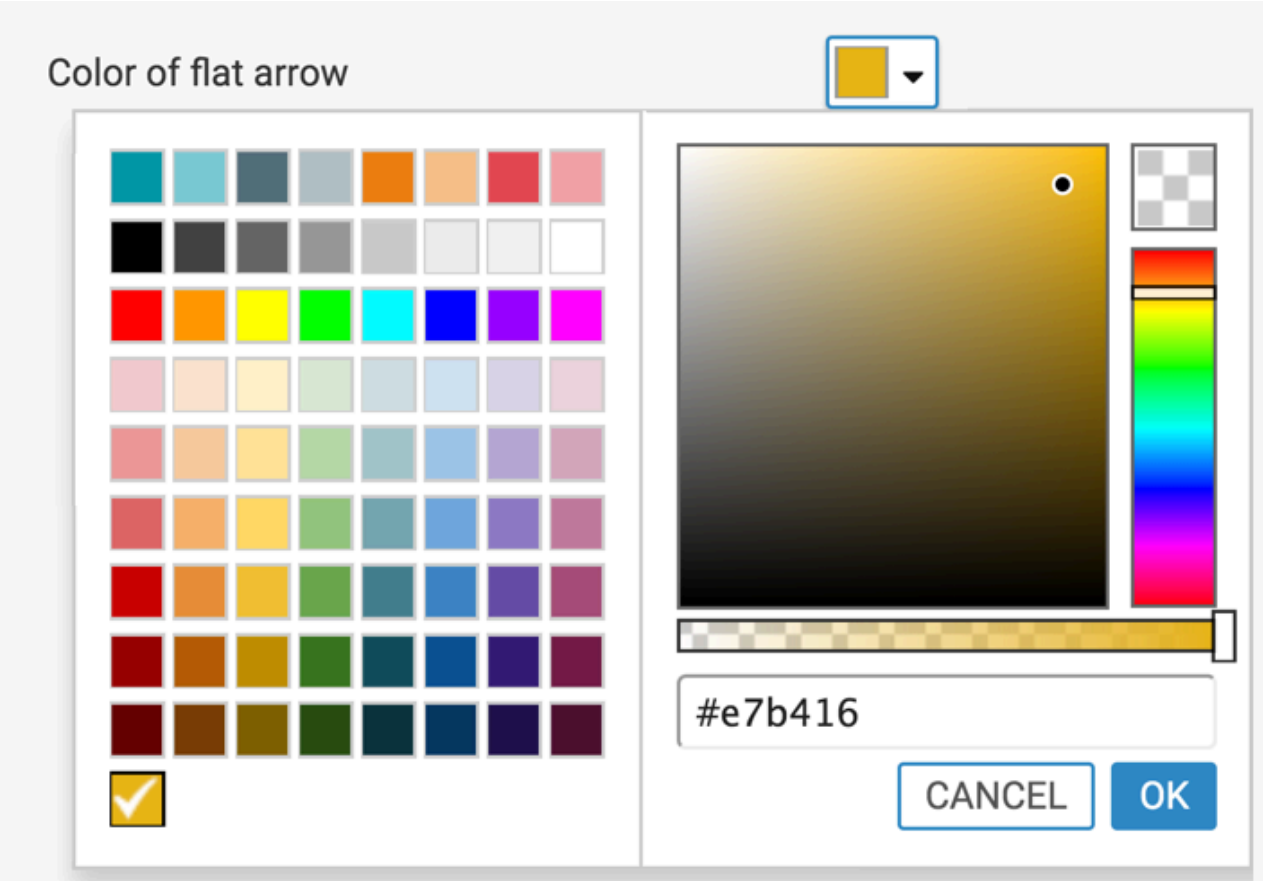

Changing the color of the down arrow

To choose a different color for the up arrow of the visual (the relative difference between the main value and the compare value), make changes in the Color of down arrow selector. You can select one of the palette colors, or specify a custom color.

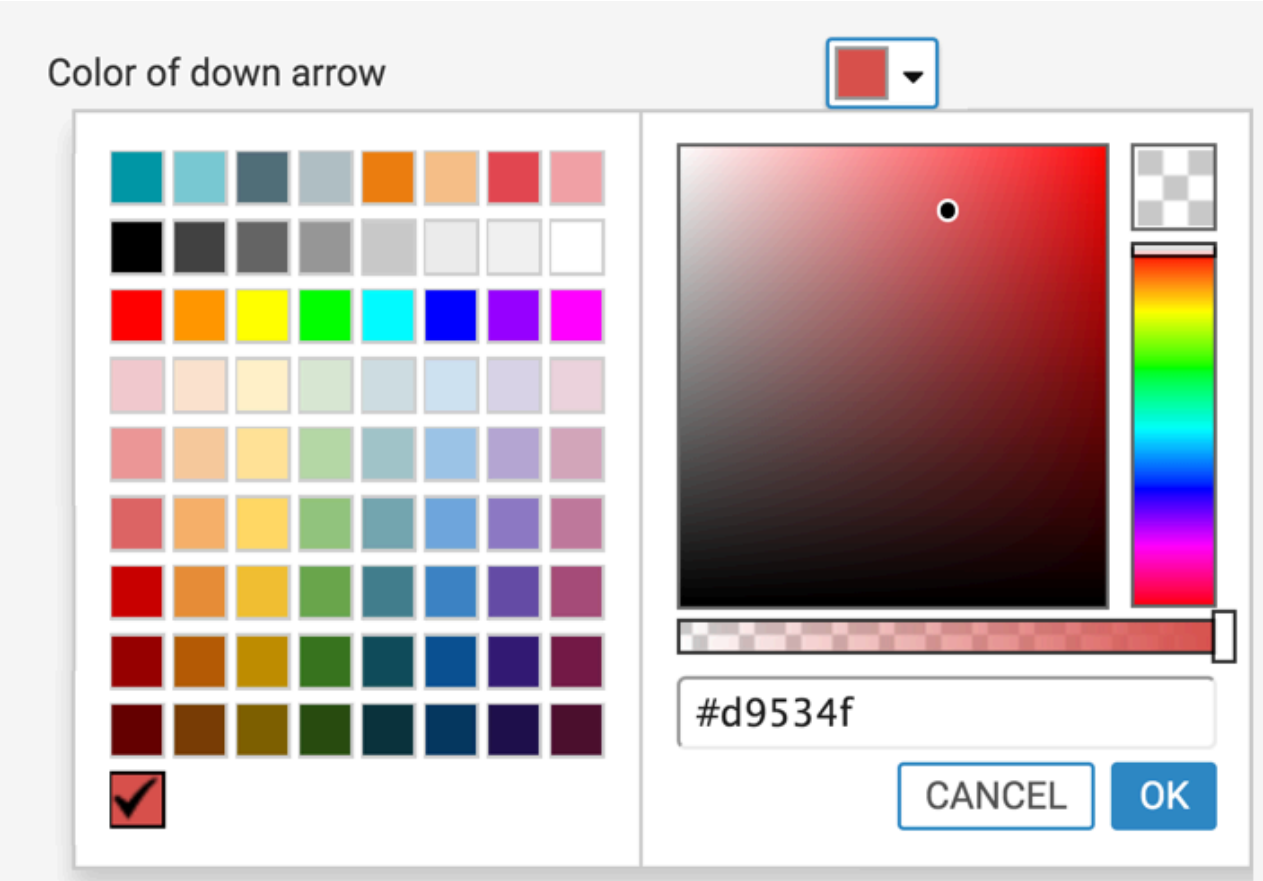

Changing the color of the main font

To choose a different color for the font of the main measure of the visual, make changes in the Main text color selector. You can select one of the palette colors, or specify a custom color.

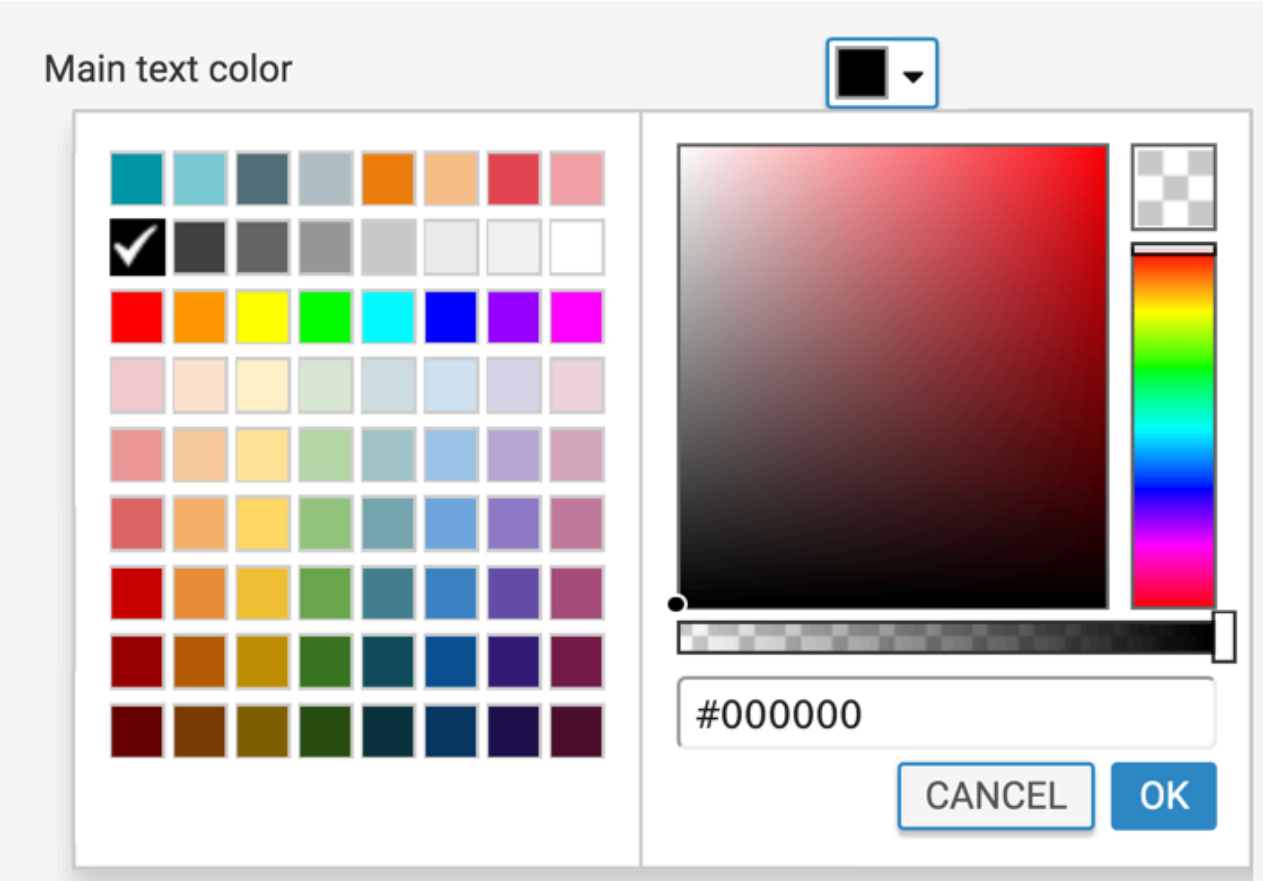

Changing the color of the label font

To choose a different color for the font of the label of the visual, make changes in the Label text color selector. You can select one of the palette colors, or specify a custom color.

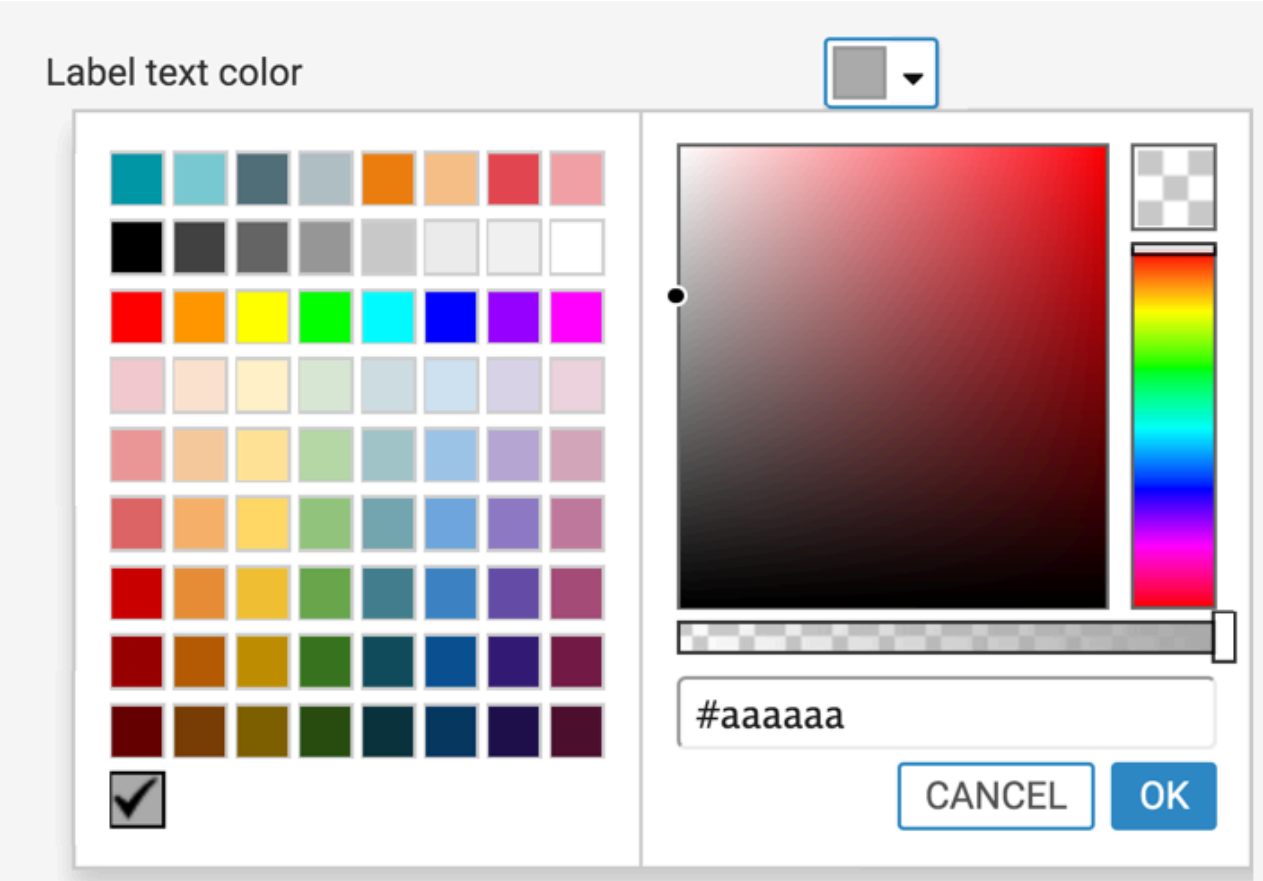

Adding custom styles to visuals

Custom styles enable you to apply styling to a particular element. You can customize each Cloudera Data Visualization visual to use CSS classes, included styles, and in-line CSS code.

About this task

When applying custom styles in Cloudera Data Visualization, you can use several approaches: setting site-wide custom styles, setting custom styles at the dashboard level, at an app level, setting styles for a filter, or setting styles for a visual, described here. Custom styles enable the user to isolate the styling to a particular element.

Note: By default, the Custom Styling option is turned off. It can be turned on by a user with administrative privileges, see *Enabling custom styling*.

To get to the options in the Custom Style menu of the visual, follow these steps:

Procedure

1. On the right side of Visual Designer, click the Custom Style menu.

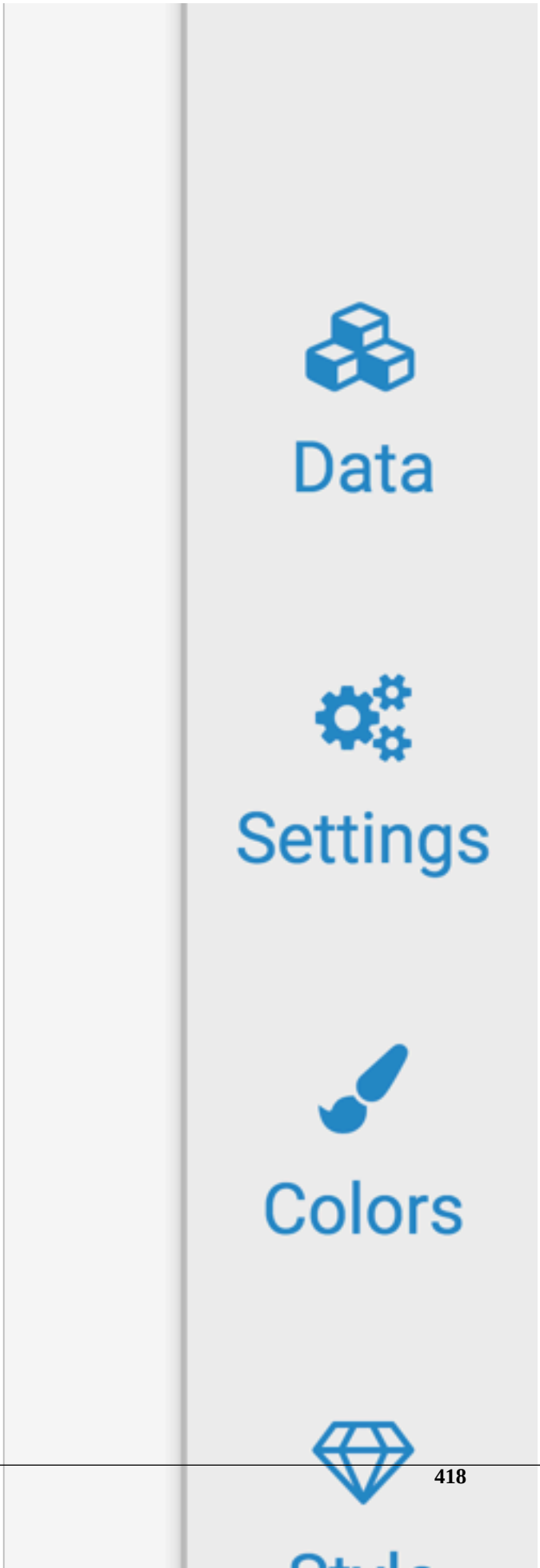
Data
Settings
Colors
Style

2. Consider the following options available on the Custom Style menu:

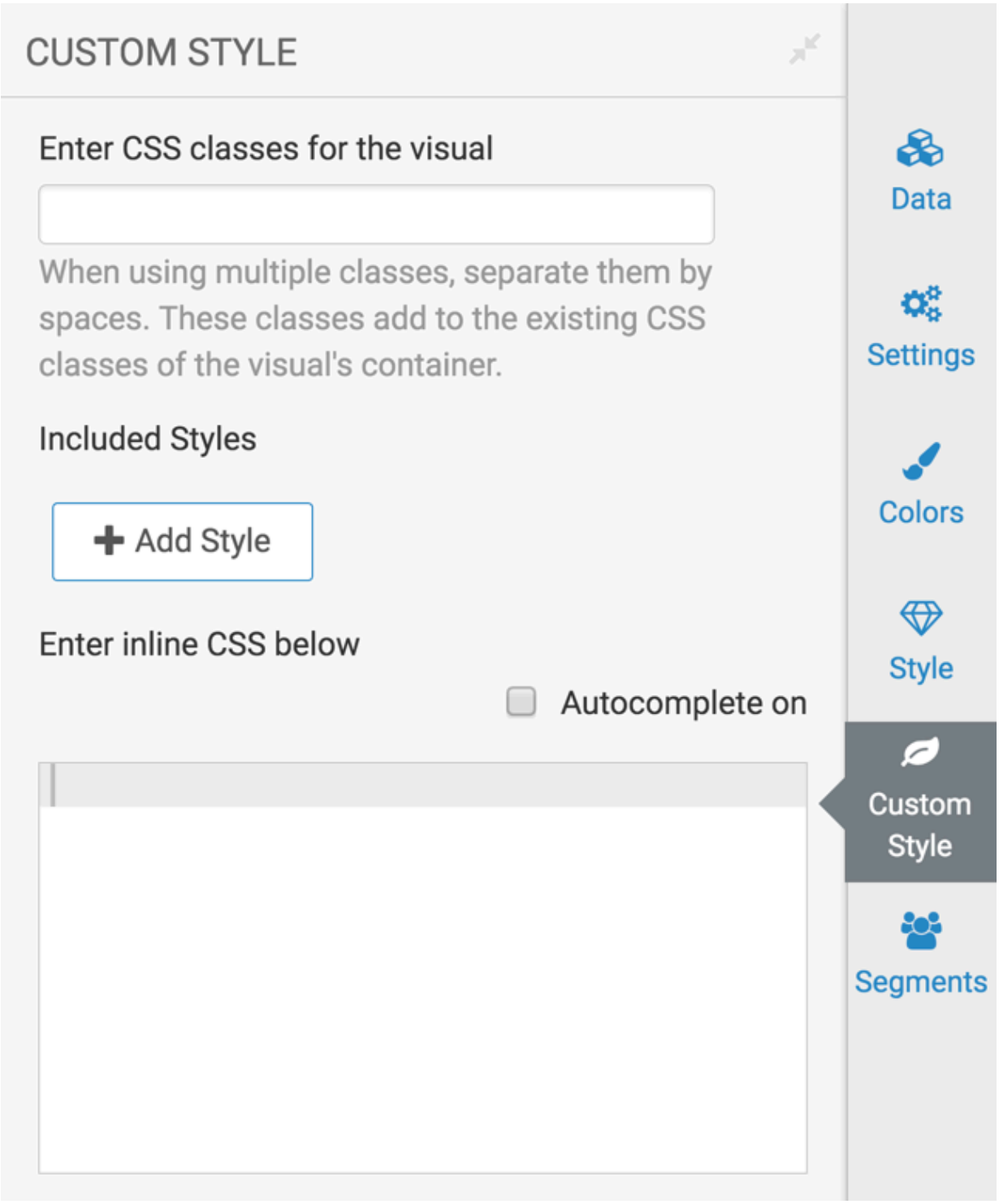

3. Select one of the options.

Adding CSS classes to visuals

Procedure

To add CSS classes to a visual, enter the name(s) of classes in the text box for the option Enter CSS classes for the visual in the Custom Style menu.

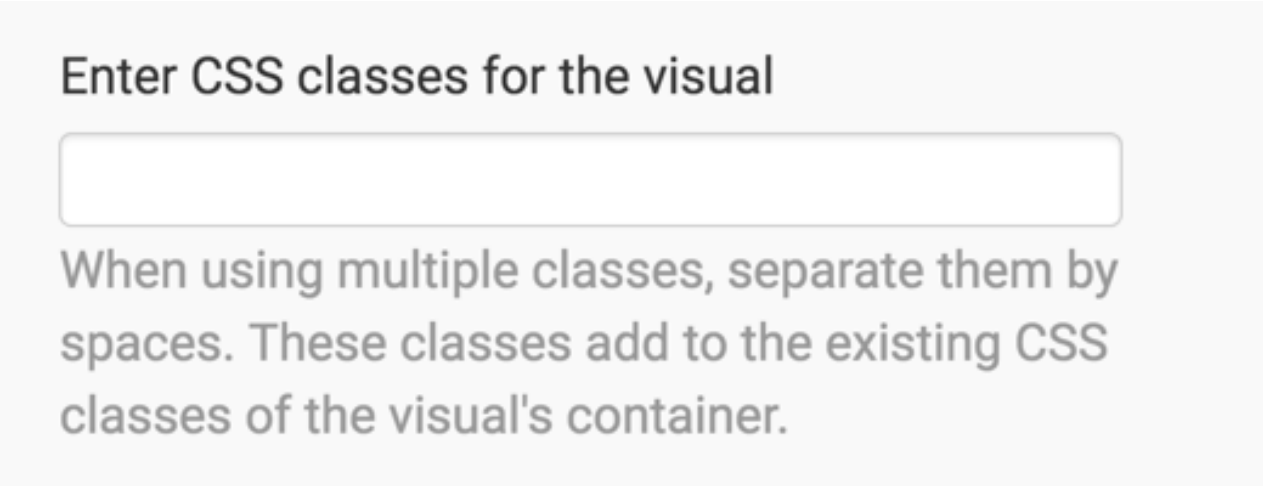

To add CSS classes to a dashboard, see *Adding CSS Classes to Dashboards*.

Related Information

Adding CSS classes to dashboards

Adding included styles to visuals

Procedure

1. To add previously defined styles to the visual, click the Add Style button in the Custom Style menu.
2. Select one or more existing style from the menu.

3. Click Add.

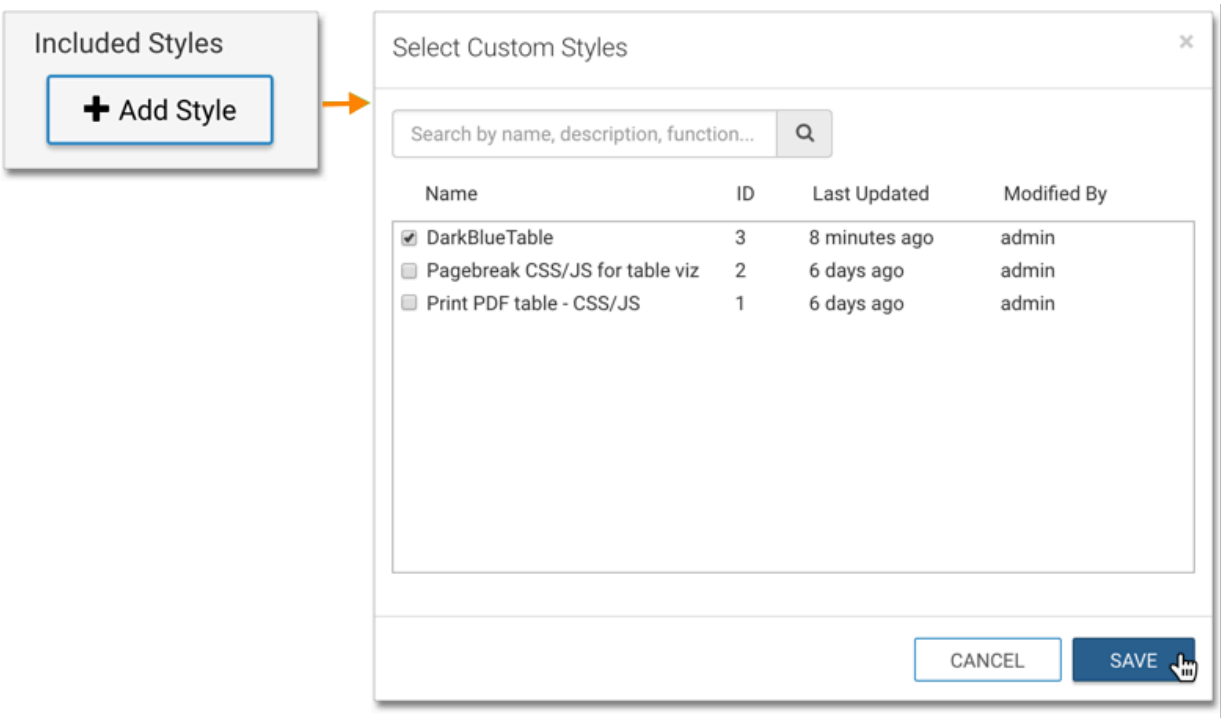

The added styles appear in the Pick Custom CSS interface. Depending on the type of the style, you can configure the style options by clicking the (gear) icon, or remove it by clicking the (x) icon.

To apply an included style to a dashboard, see *Adding Included Styles to Dashboards.*

Example

In this example, the styles Word Cloud and 3D Donut have been added.

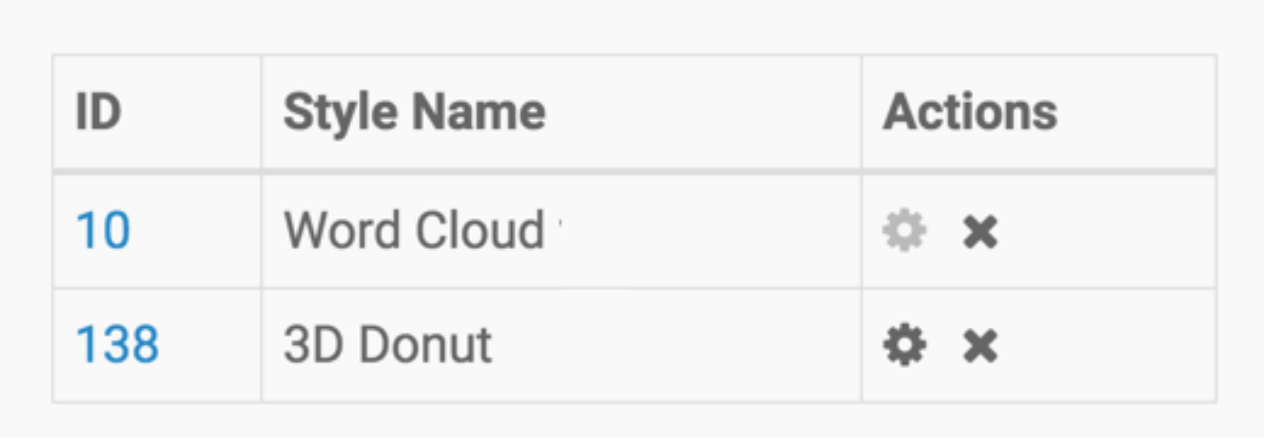

ID	Style Name	Actions
10	Word Cloud	
138	3D Donut	

After refreshing a pie visual, it will look something like this:

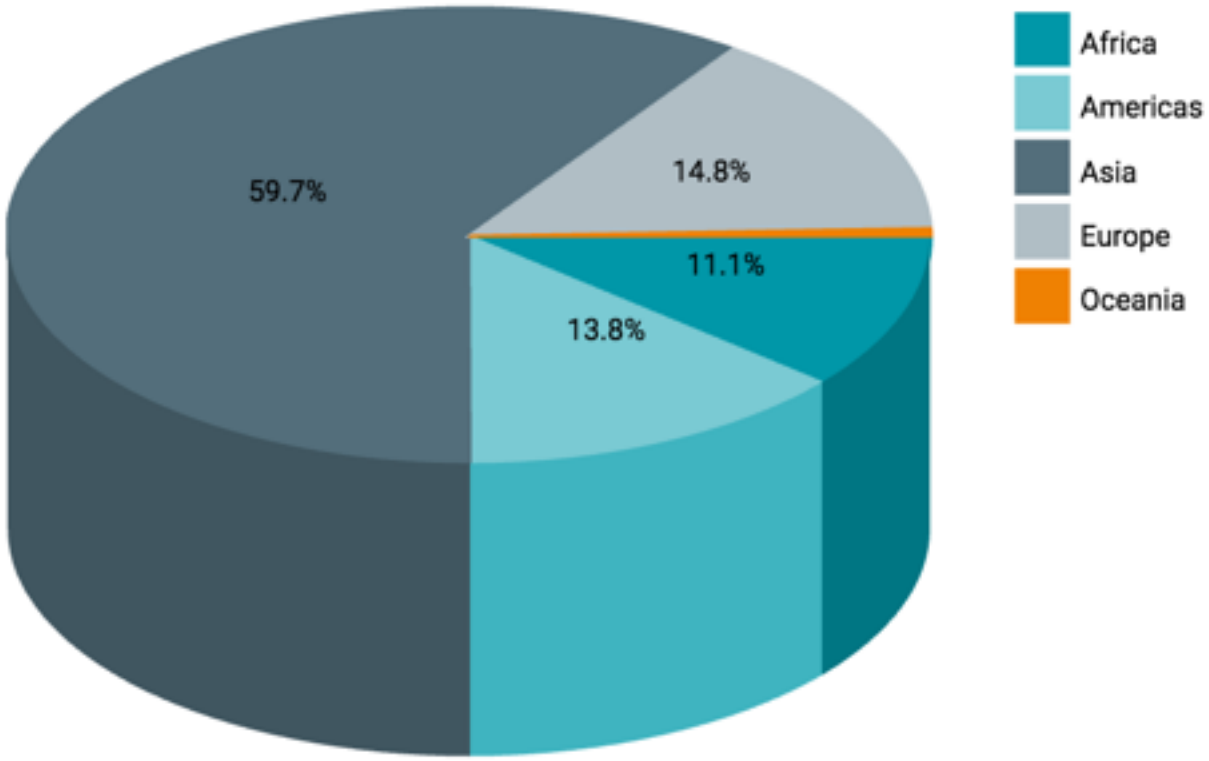

You can also search or sort the custom styles in the Pick Custom CSS interface.

Related Information

Adding included styles to dashboards

Searching custom styles in visuals

Cloudera Data Visualization enables you to search custom styles in visuals by Name and Modified By fields. You can also search by the content of CSS and Javascript code.

About this task

When you start typing characters in the search text box, the search results display the exact match on the top, followed by the closest matches. You must type at least two characters to start the search.

To demonstrate how to search custom styles in visuals, follow these steps:

Procedure

1. On the right side of Visual Designer, click the Custom Style menu.

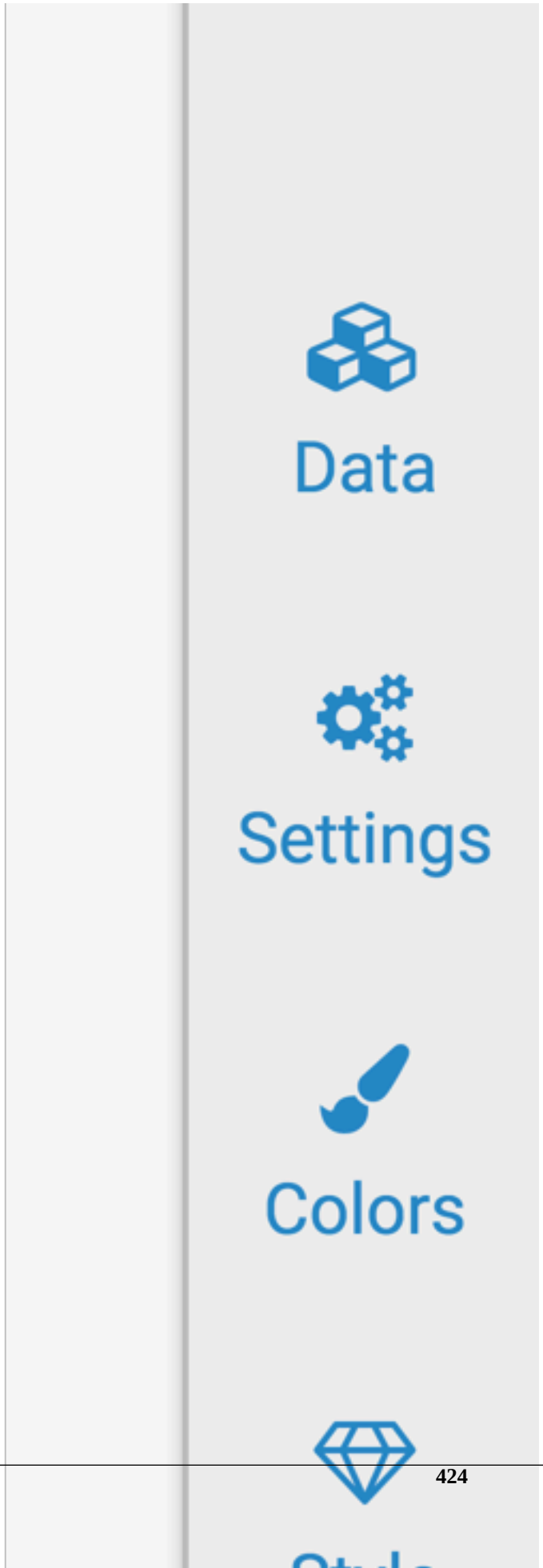
Data
Settings
Colors

2. Click the Add Style button.

   The Pick Custom CSS interface appears.

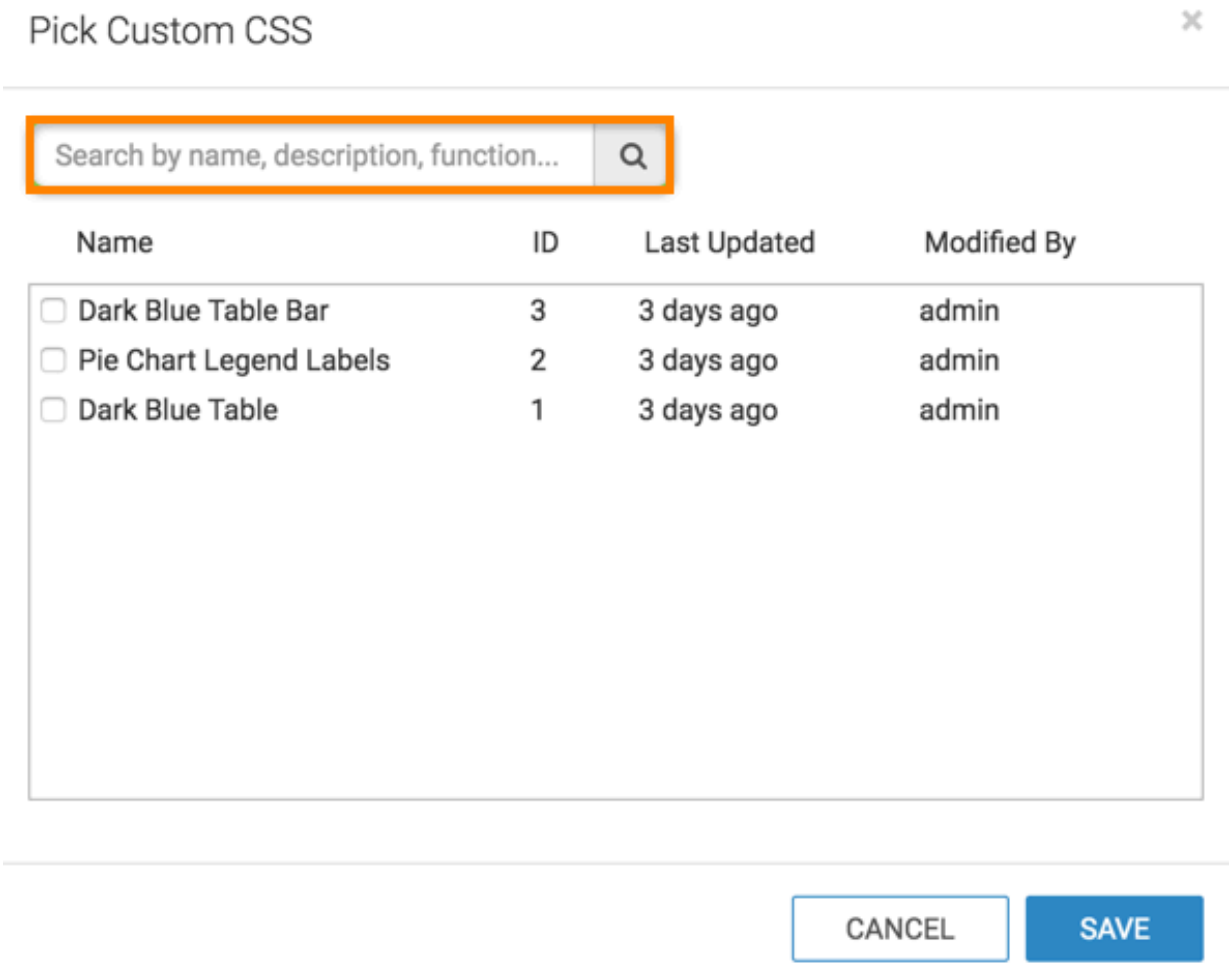

3. To demonstrate how to search the list of custom styles, enter Pie in the Search text box.

   In the following image, you can see that the Pie Chart Legend Label custom style is filtered from the list.

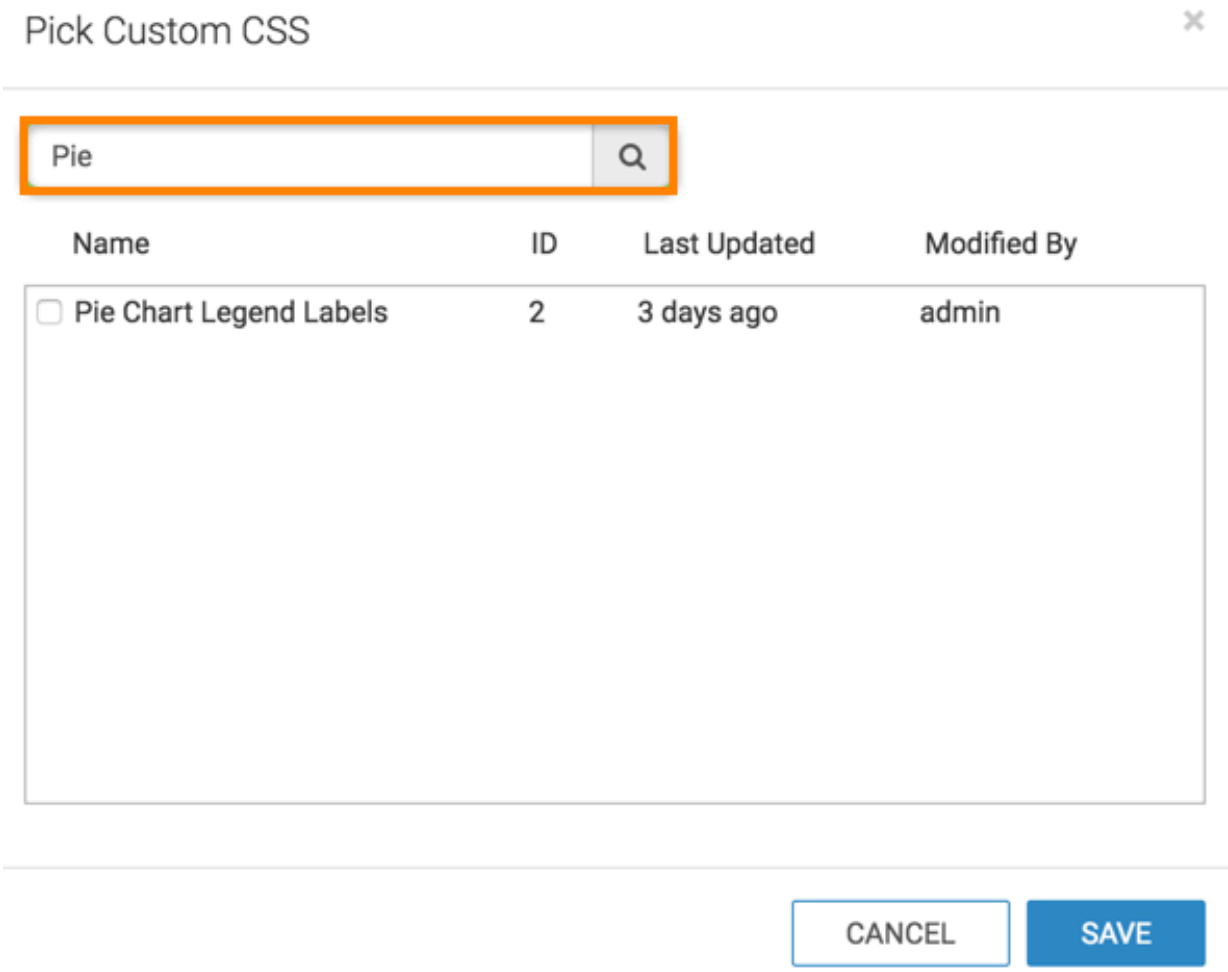

4. To search the list of custom styles by JavaScript code, follow these steps:
   a) Click the (pencil) icon next to the Pie Chart Legend Labelto view the JavaScript code of this style. In our example, we will search by Not Applicable from the JavaScript code.

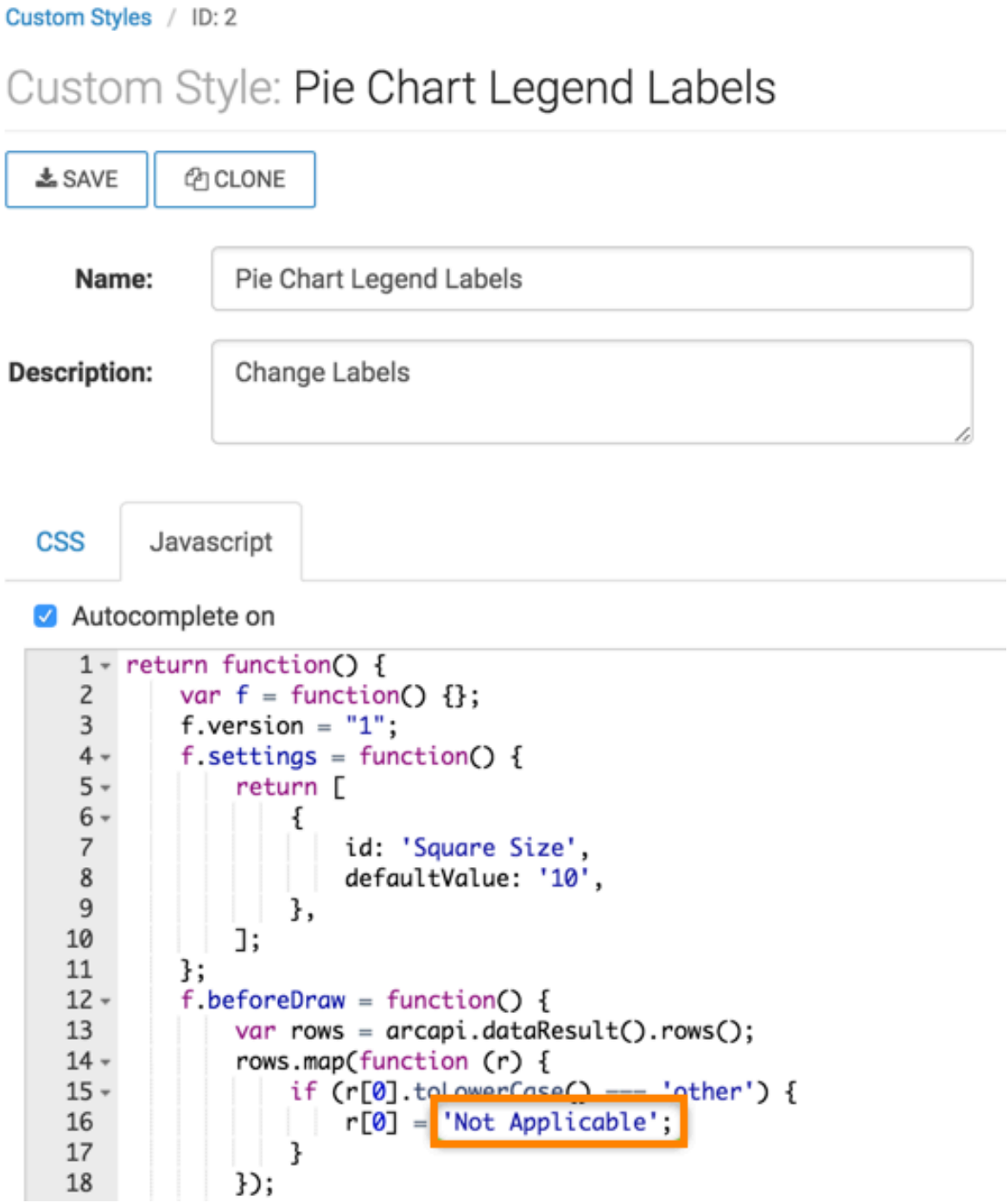

   b) Enter Not Applicable in the Search text box.

      In the following image, you can see that the Pie Chart Legend Label custom style is filtered out of the list of custom styles.

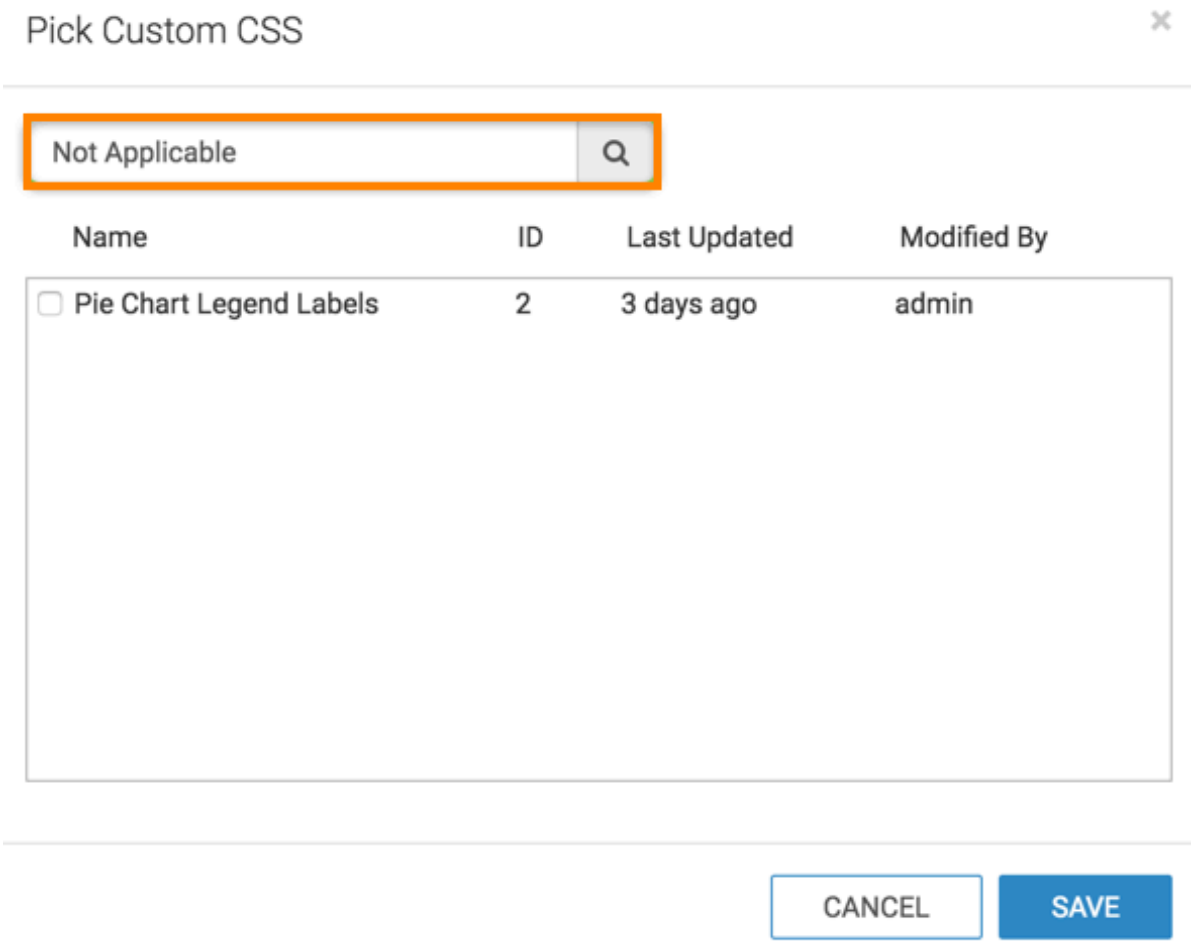

Sorting custom styles in visuals

Cloudera Data Visualization enables you to sort custom styles in visuals by ID, Name, Last Updated, and Modified By fields. You can sort in ascending or descending order.

About this task

Note: You cannot sort by the Description field.

To demonstrate how to sort custom styles by the Name field, follow these steps:

Procedure

1. On the right side of Visual Designer, click the Custom Style menu.

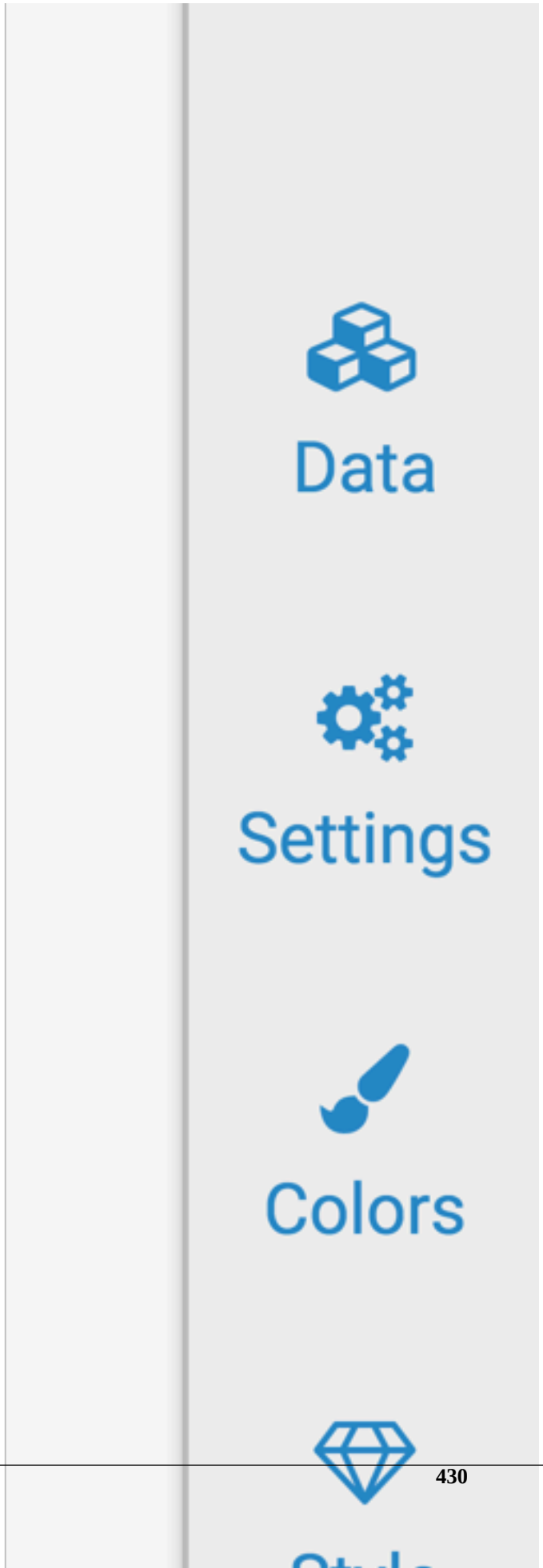
Data
Settings
Colors

2. Click the Add Style button.

   The Pick Custom CSS interface appears.

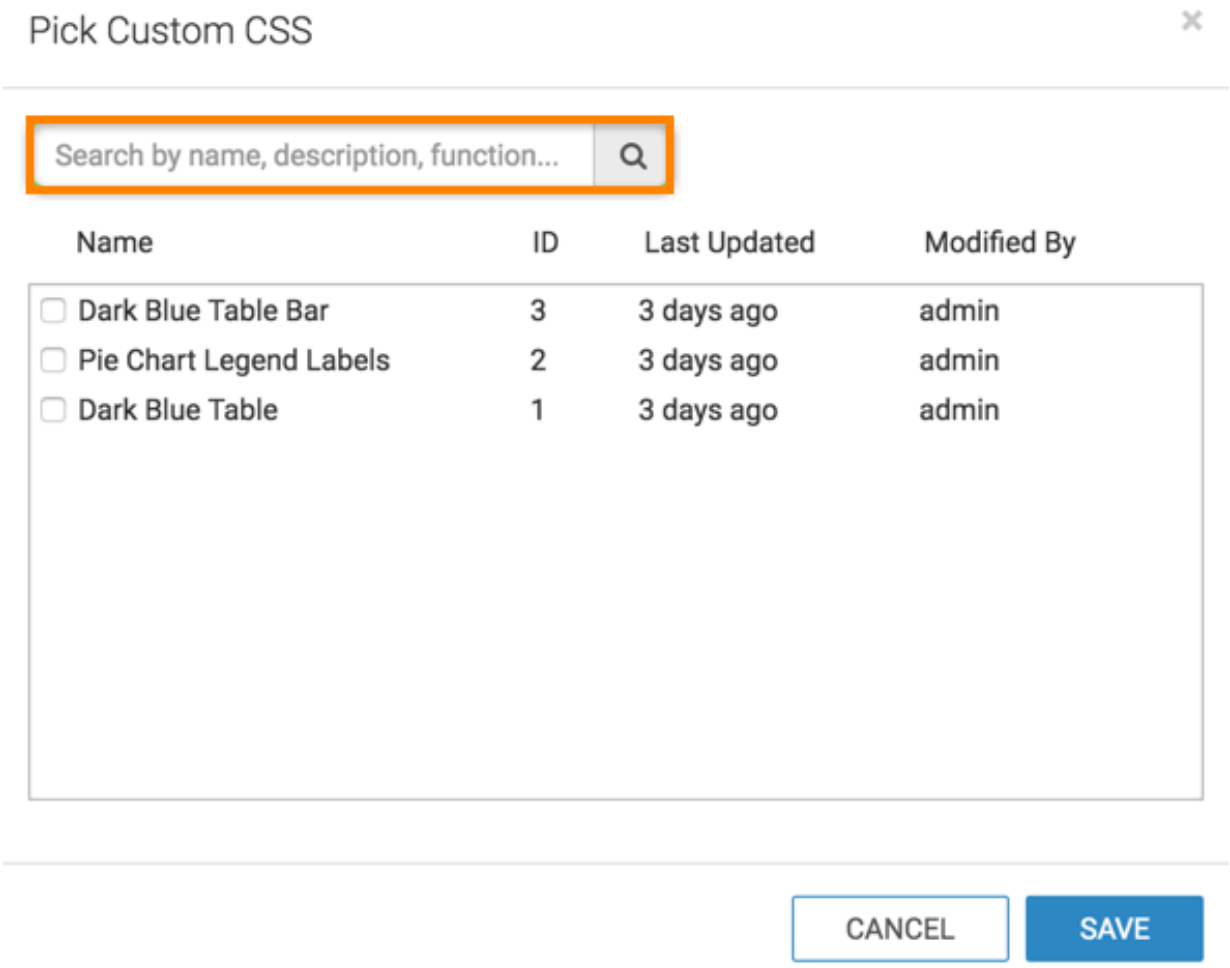

3. Click the Name field.

Note: When you hover over a field, a hand icon appears only on the fields that you can sort.

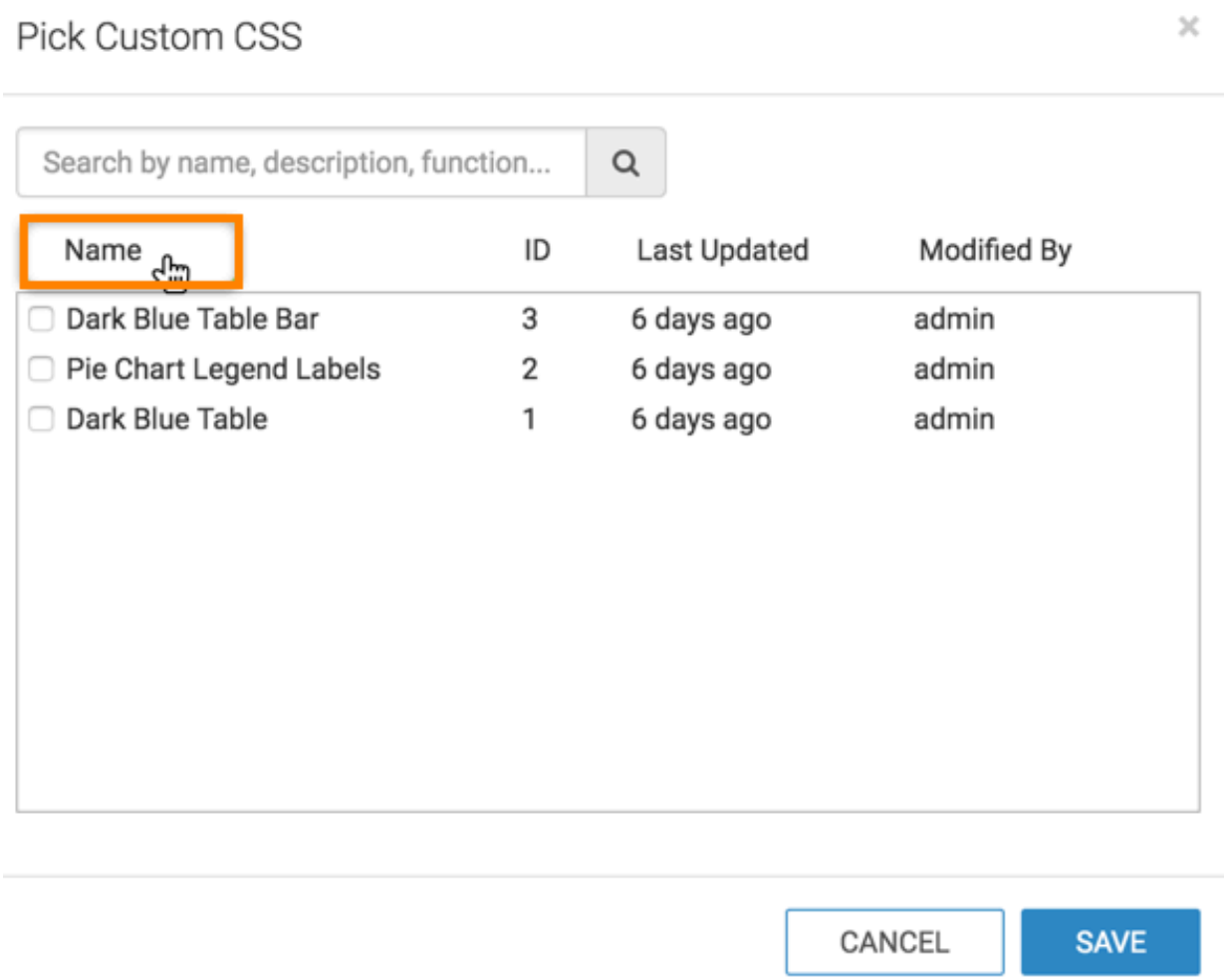

In the following image, the Name field is sorted in ascending order and a sort icon appears next to the field.

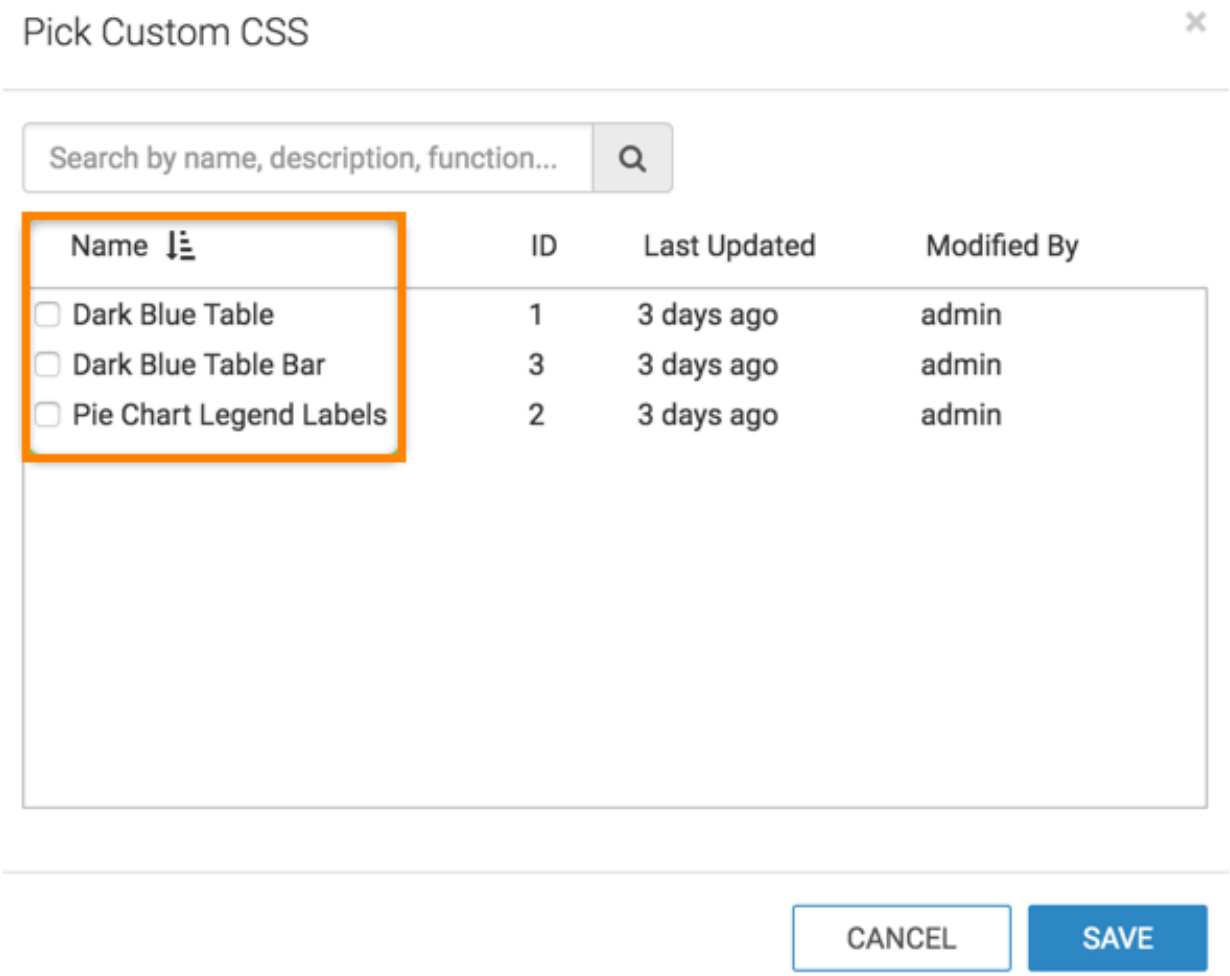

4. To sort the field in descending order, click the Name field again.

Adding inline CSS code to visuals

About this task

To add simple formatting to a visual, follow thsese steps:

Procedure

1. On the right side of Visual Designer, click the Custom Style menu.

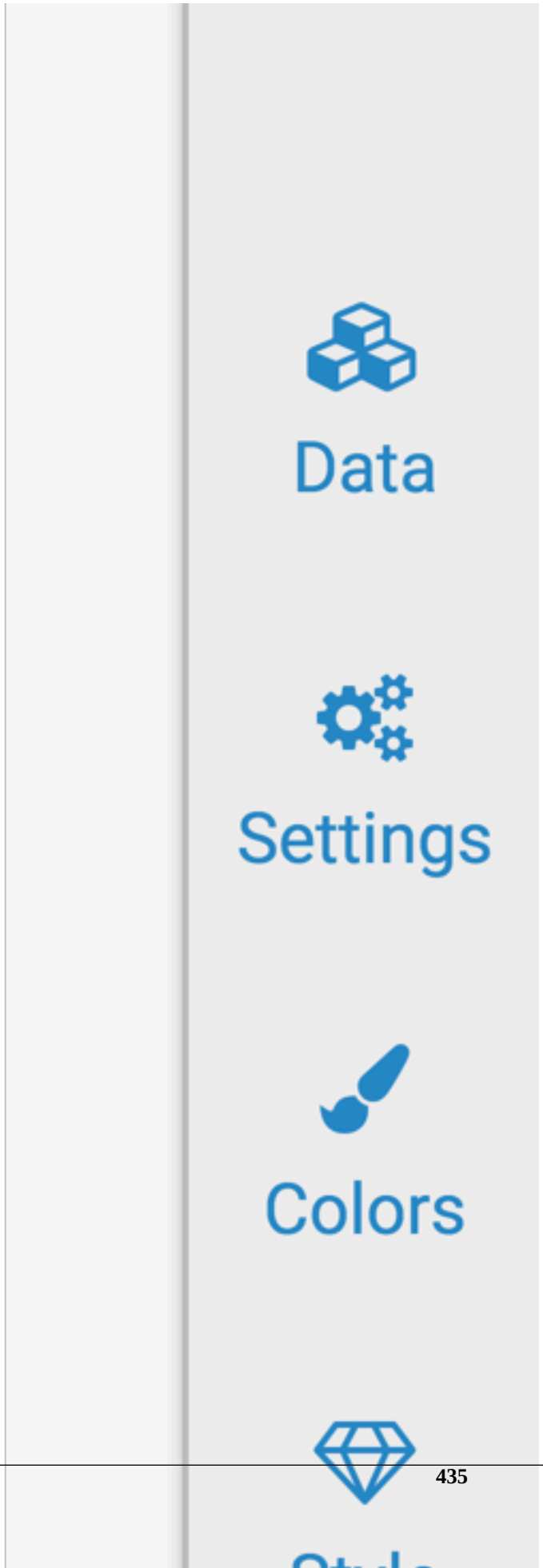
Data
Settings
Colors

2. Enter a CSS code snippet under the Enter in-line CSS option. You can select the Autocomplete option, too.

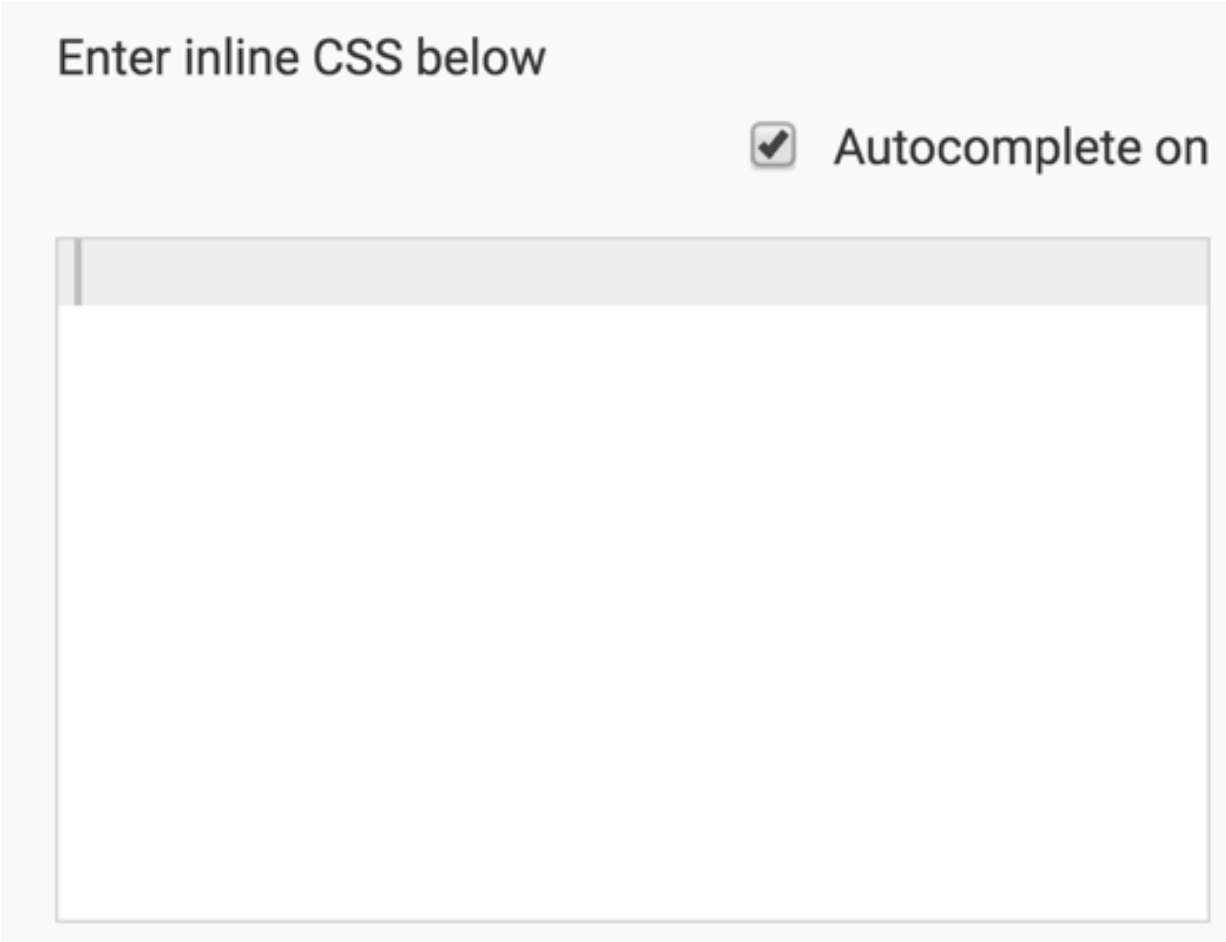

To apply an inline style to a dashboard, see *Adding Inline CSS code to Dashboards*.

Related Information

Adding inline CSS code to dashboards

Inline CSS usage

Under Enter in-line CSS, enter the following code to set the background and font color for table headings:

```
th{
  background-color: #336699;
  color: #ffffff;
}
```

After refreshing a table visual, it will look something like this:

Country	Year	Life Expectancy
Afghanistan	2010	59.60
Albania	2010	76.80
Algeria	2010	70.60
American Samoa	2010	null
Andorra	2010	null
Angola	2010	50.70
Anguilla	2010	null
Antigua and Barbuda	2010	75.40
Argentina	2010	75.80
Armenia	2010	74.30
Aruba	2010	75.10
Australia	2010	82.10
Austria	2010	80.60
Azerbaijan	2010	70.50
Bahamas	2010	74.80

Cloudera Data Visualization 7.2.9

Custom Styles

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Adding custom styles to visuals 4

Defining site-wide custom styles 6

CSS custom styles 6

JavaScript custom styles 8

Modifying site-wide custom styles 12

Deleting site-wide custom styles 13

Searching site-wide custom styles 15

Sorting site-wide custom styles 17

Adding custom styles to visuals

Custom styles enable you to apply styling to a particular element. You can customize each Cloudera Data Visualization visual to use CSS classes, included styles, and in-line CSS code.

Note: This feature is only available to users with Manage custom styles or Administrator privileges.

When applying custom styles in Cloudera Data Visualization, you may use several approaches:

- Defining site-wide custom styles
- Adding custom styles to dashboards
- Adding custom styles to apps
- Configuring custom styles for dashboard filters

Custom styles enable you to isolate the styling to a particular element.

Note: The custom styling option is turned off by default. It can be turned on by a user with administrative privileges.

To get to the options in the Custom Style menu of the visual, follow these steps:

1. On the right side of Visual Designer, click the Custom Style menu.

2. Consider the following options available on the Custom Style menu:
   - Adding CSS Classes to Visuals
   - Adding Included Styles to Visuals
   - Adding Inline CSS code to Visuals

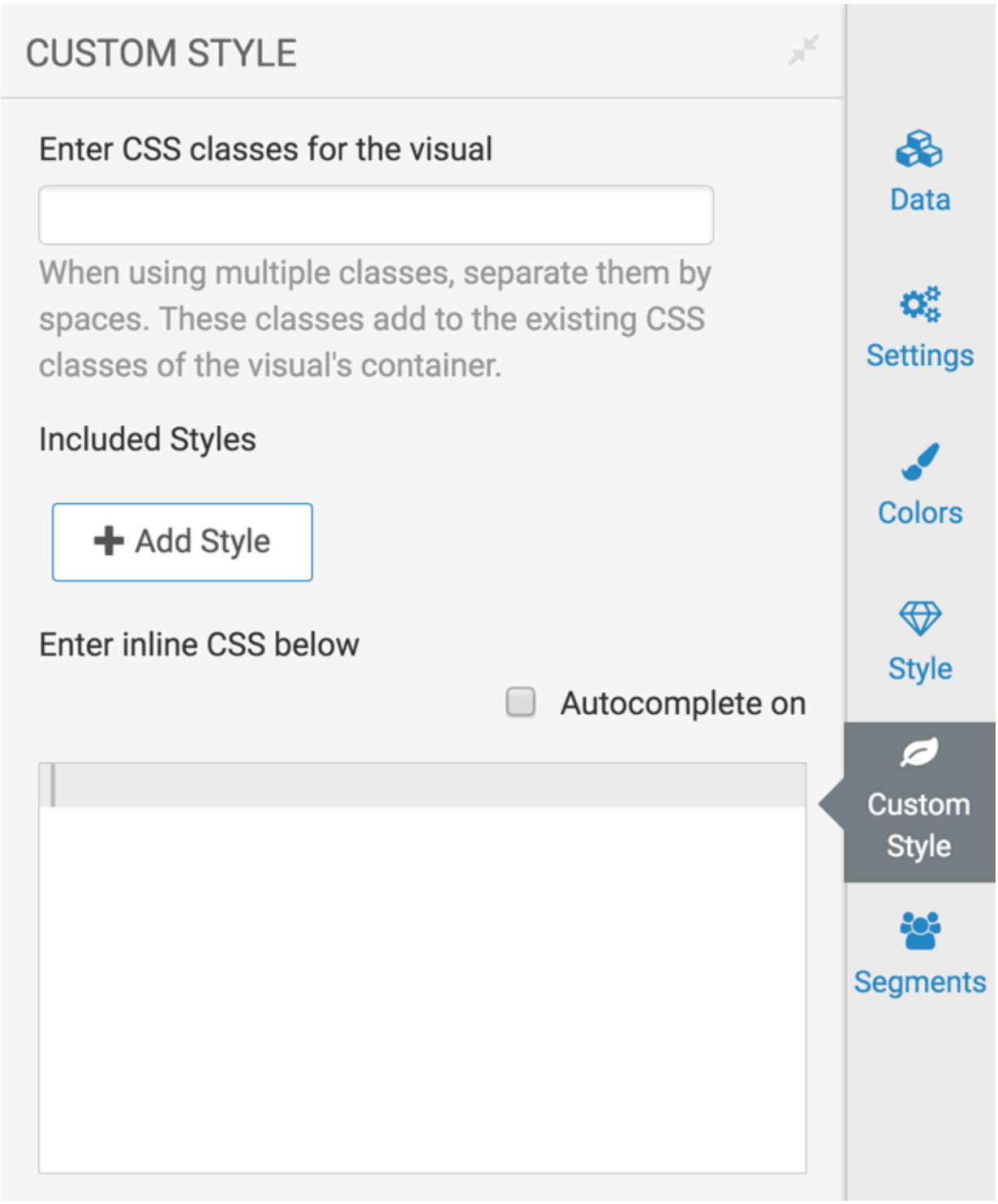

Related Information

Defining site-wide custom styles

Adding custom styles to dashboards

Adding custom styles to apps
Configuring custom styles for dashboard filters
Adding CSS classes to visuals
Adding included styles to visuals
Adding inline CSS code to visuals

Defining site-wide custom styles

About this task

To define styling for the entire site, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Custom Styles.
3. In the Manage Custom Styles interface, click New Custom Style.

   The New Custom Style interface appears.
4. Add the Name of the new custom style, and optionally, its Description.
5. Click SAVE.
6. Depending on your permissions, you can add CSS and Javascript (JS) styling:
   - CSS custom styles on page 6
   - JavaScript custom styles on page 8

CSS custom styles

About this task

To define styling for the entire site using CSS code, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Custom Styles.
3. In the Manage Custom Styles interface, under Name, enter DarkBlueTable.
4. To add CSS instructions, enter the following code:

```
.table{
          background-color:#336699;
```

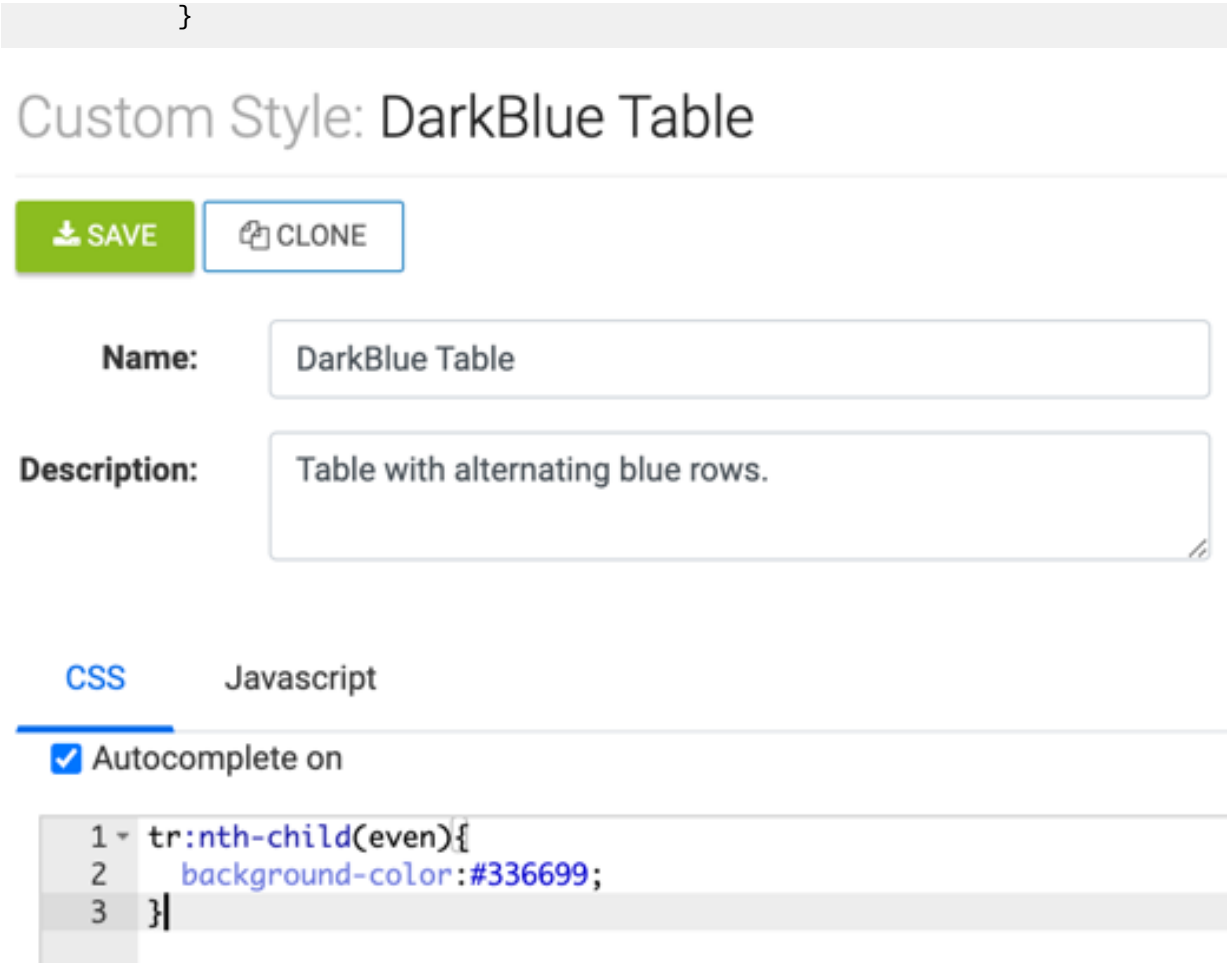

5. Click Save.
6. Notice that the Manage Custom Styles window contains the new style.

   For information on how to include the new style in visual, see *Adding custom styles to visuals*.

   When selected for a visual that contains a table, the style DarkBlueTable gives the table the specified coloring scheme, much like the following image.

cereal_name	manufacturer_code	cold_or_hot	calories	protein_grams	fat_grams	sodium_mg	dietary_fibe
100%_Bran	N	C	70	4	1	130	
100%_Natural_Bran	Q	C	120	3	5	15	
All-Bran	K	C	70	4	1	260	
All-Bran_with_Extra_Fiber	K	C	50	4	0	140	
Almond_Delight	R	C	110	2	2	200	
Apple_Cinnamon_Cheerios	G	C	110	2	2	180	
Apple_Jacks	K	C	110	2	0	125	
Basic_4	G	C	130	3	2	210	

Note that the Manage Custom Styles interface keeps track of all visuals that use each defined style. In the following image, you can see the reporting after applying the style DarkBlueTable to visualization 15. Note also that the number 15 is an active link to that visualization.

Related Information

Defining site-wide custom styles
Adding custom styles to visuals

JavaScript custom styles

About this task

Note: This feature is limited to users who have both Manage custom styles and Manage dataset privileges. For more information on role-based access in Cloudera Data Visualization, see *RBAC Permissions*.

To define styling for the entire site using JavaScript (JS) code, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Custom Styles.
3. In the Manage Custom Styles interface, click New Custom Style, and then select the JavaScript tab.

4. If you enable the Autocomplete on option, and then type arc, you can choose from a list of Arcadia-specific APIs, such as arcapi.settings, arcapi.getSettings, arcapi.sendParameters, arcapi.getParameters, and so on.

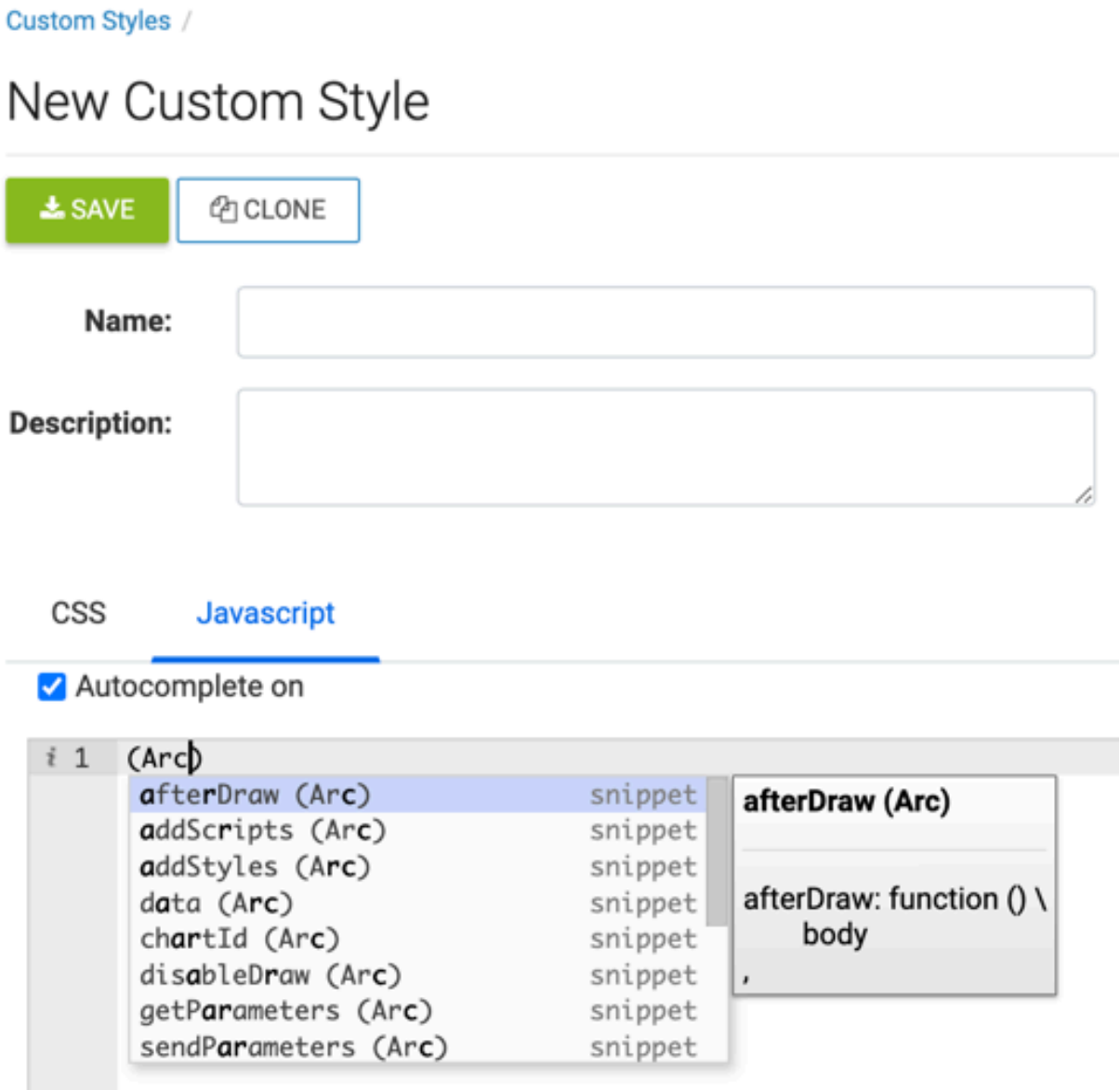

5. Under Name, enter Pie Chart Labels.
6. Under Description, enter Specifying labels and values in the legend.
7. Under the JavaScript tab, add the following code to change the name of one the labels and add values after the label name.

```
return function() {
            var f = function() {};
            f.version = "1";
            f.settings = function() {
            return [
            {
            id: 'Square Size',
            defaultValue: '10',
            },
            ];
            };
            f.beforeDraw = function() {
```

```
            var rows = arcapi.dataResult().rows();
            rows.map(function (r) {
            if (r[0].toLowerCase() === 'all-bran') {
            r[0] = 'Not Applicable';
            }
            });
            arcapi.dataResult().rows(rows);
            };
            f.afterDraw = function() {
            var rows = arcapi.dataResult().rows();
            var length = $('#' + arcapi.chartId() + ' .viz-base-legend-g')
.children('text').length;
            var total = d3.sum(rows, function(row) { return +row[1]; });
            var legendlabel;
            var fmt = d3.format(",");

            rows.forEach(function(d, i) {
            for (var c=0; c < length; c++ ){
            legendlabel = $('#' + arcapi.chartId() + ' .viz-base-legend-g
text')[c].innerHTML;
            if(legendlabel == d[0]){
            $('#' + arcapi.chartId() + ' .viz-base-legend-g text')[c].inne
rHTML = legendlabel + ' (' + fmt(+d[1]) + ')';
            }
            }
            });
            };
            return f;
            }();
```

8. Click Save.

Results

The following image shows the pie chart before applying the JavaScript code. The legend in the pie chart displays label names without values. Also, one of the labels is named All-Bran.

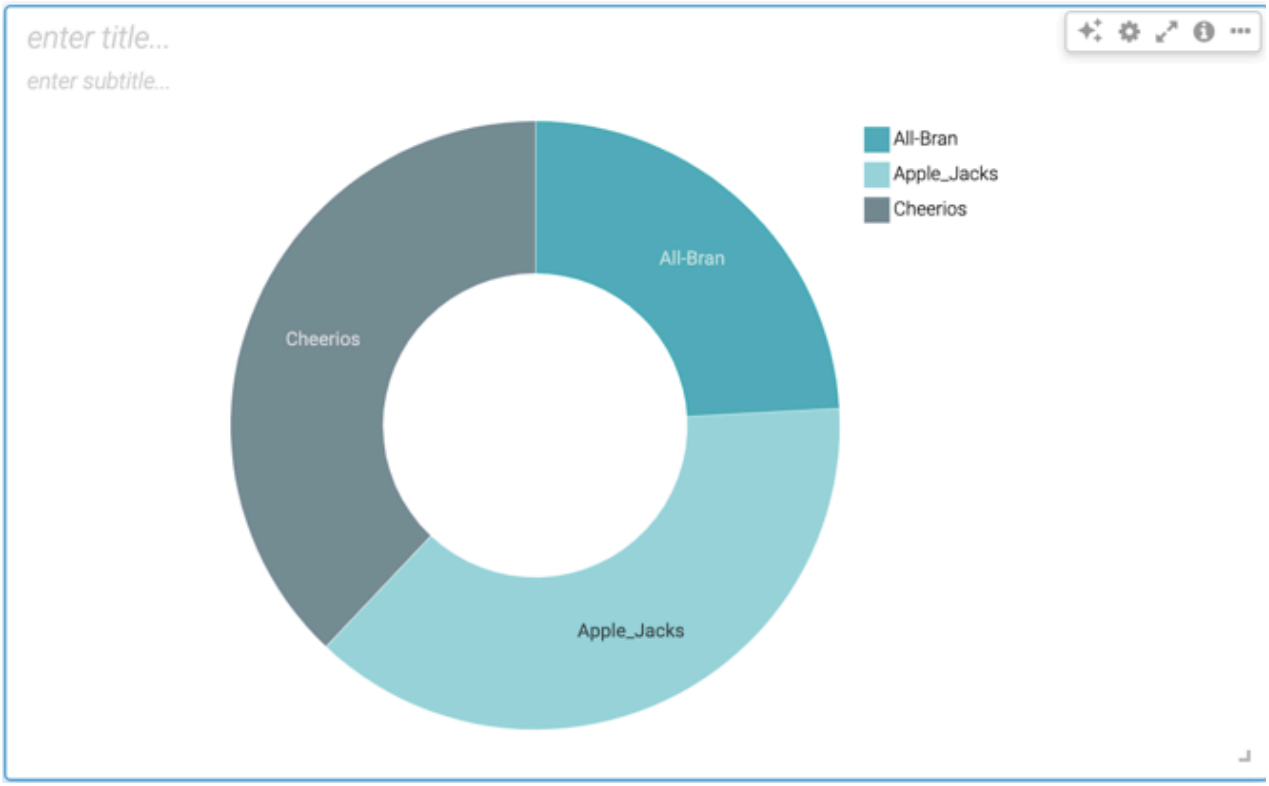

The following image shows the pie chart after applying the JavaScript code. The legend in the pie chart now displays the values in the legend. The All-Bran legend label name is renamed to Not Applicable.

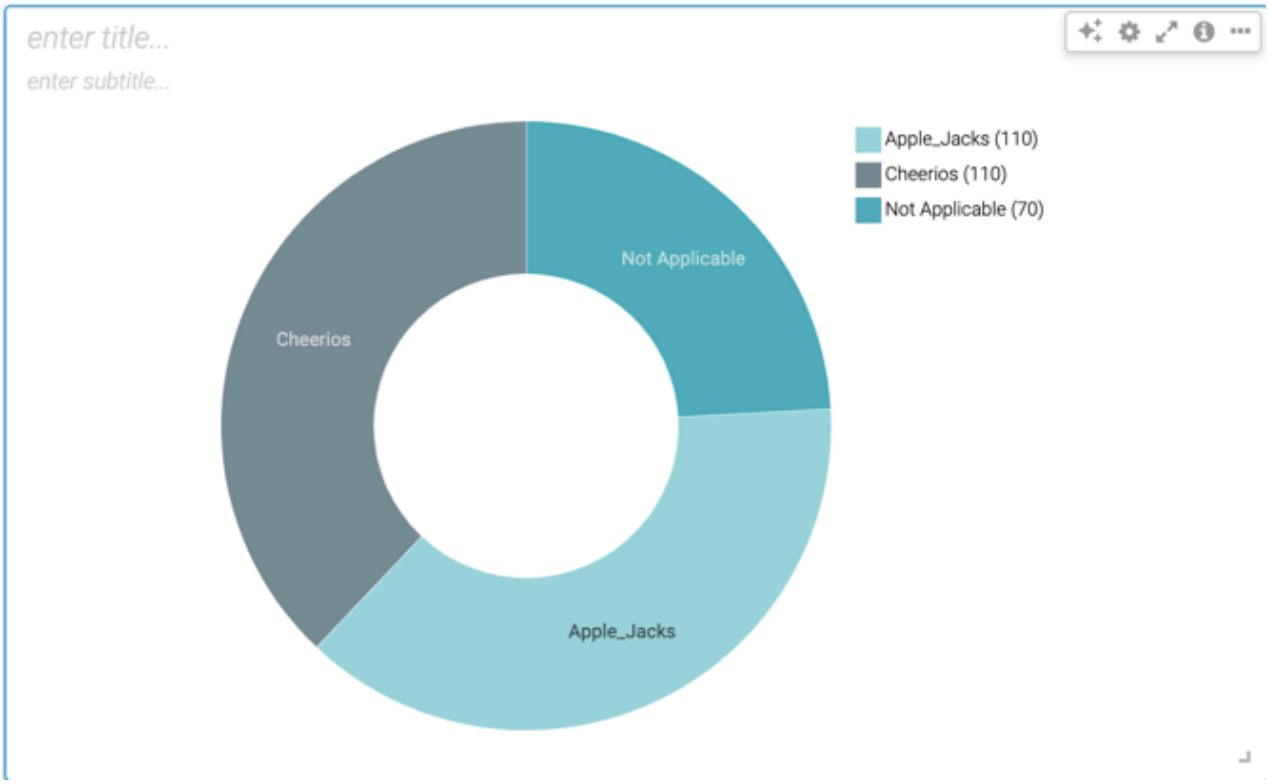

Note that the Custom Styles interface keeps track of all visuals that use each defined style.

Here, you can see that the Custom Styles interface lists the new style, Pie Chart Labels, and that it applies to visualization with ID 6700. Note also that this ID is an active link to the visualization.

Related Information

Defining site-wide custom styles
RBAC Permissions

Modifying site-wide custom styles

About this task

To edit the existing site-wide custom styles, follow these steps.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Custom Styles.
3. In the Manage Custom Styles window, click the Pencil icon next to the custom style you want to edit.

   In this case, the custom style DarkBlueTable is edited.
4. In the Custom Style: DarkBlueTable window, make the necessary changes, and then click Save.

   You can also rename the style to DarkBlueTableBar.

   In this example, the Javascript tab is selected, and some code is added to convert the last aggregate column to a proportionally sized horizontal bar.

```
return {
            version: "1",

            afterDraw: function() {
            var d = arcapi.dataResult();
            var rows = d.rows();
            var aggColIndex = -1;
            d.columns().forEach(function(col) {
            if (col.colproperty() == 2) {
            aggColIndex = col.colindex();
            }
            })

            if (aggColIndex < 0) {
            return;
            }

            console.log(aggColIndex);
            var extent = d3.extent(rows, function(row) {
            return +row[aggColIndex];
            });

            var scale = d3.scale.linear()
            .domain(extent)
            .range([5, 80]);

            console.log(extent);
            d3.selectAll("#" + arcapi.chartId() + " .table tbody tr :nth-
child(" + (aggColIndex + 1) + ")")
            .text("")
            .append("div")
            .style("background-color", "black")
            .style("height", "15px")
            .style("width", function(d, i) {
```

```
            var m = +rows[i][aggColIndex];
            var w = scale(m);
            return w + "px";
            });
            }
            };
```

The changes in the table visual:

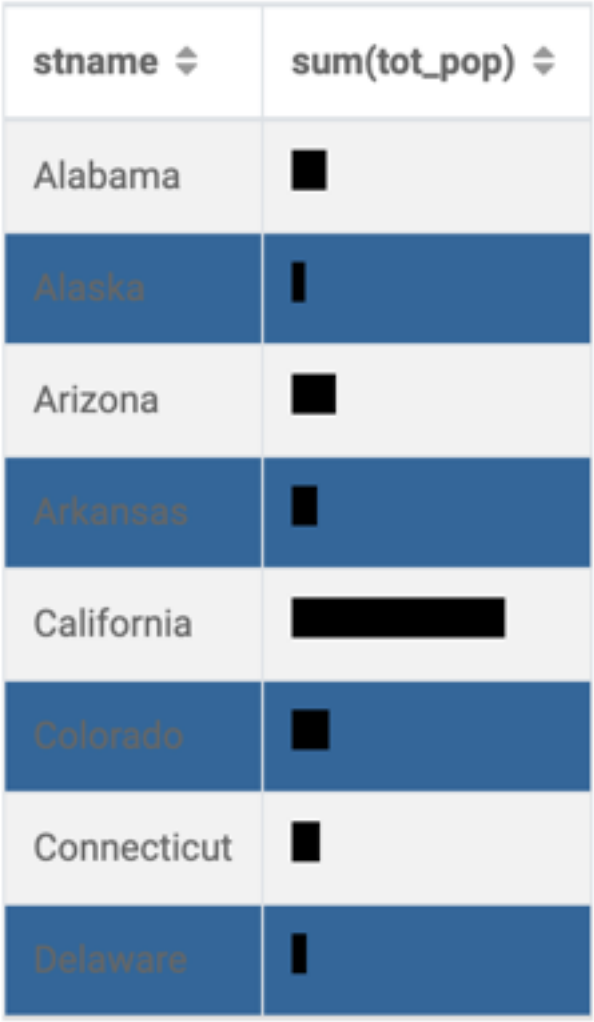

As the name of an existing custom style has been changed to DarkBlueTableBar, there is a corresponding update in the Manage Custom Styles window.

Deleting site-wide custom styles

About this task

To remove existing site-wide custom styles, follow these steps.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.

2. Select Custom Styles.
3. In the Manage Custom Styles interface, click the Trashcan icon next to the custom style that you want to edit.

   In this example, the custom style BlueAllOver will be deleted.

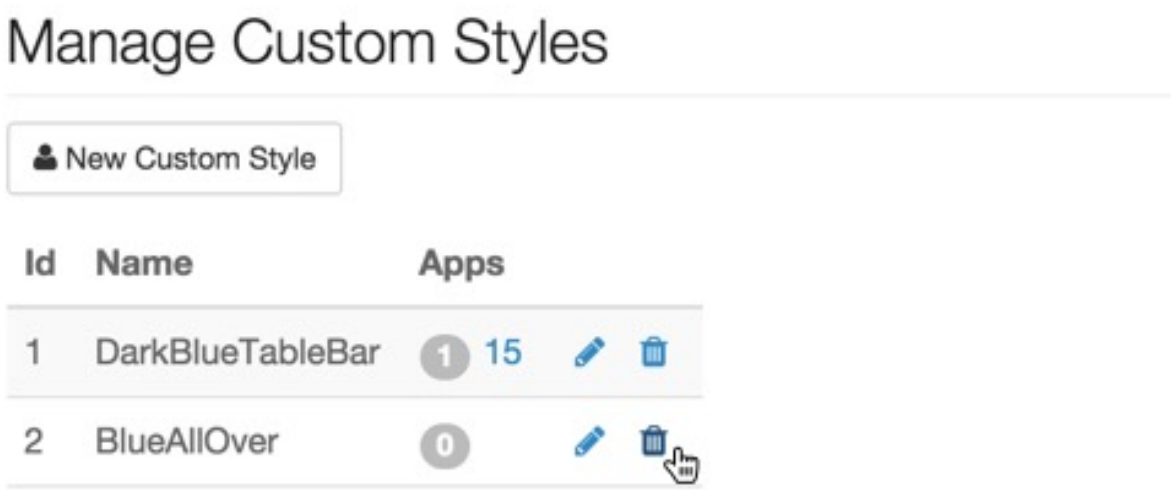

4. In the Delete Confirmation window, type DELETE, and then click Delete.

Results

The recently deleted style is no longer listed in the Manage Custom Styles window.

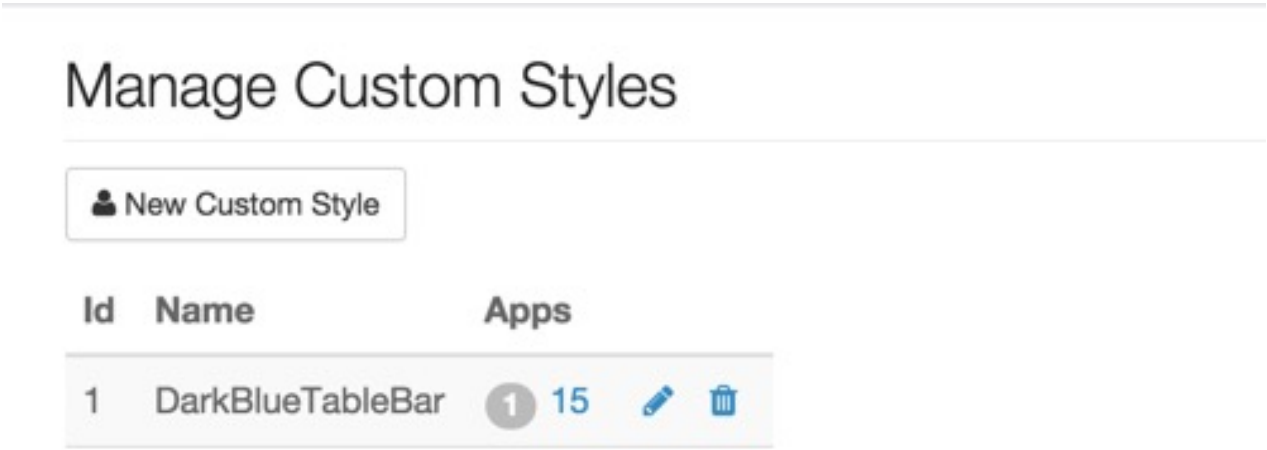

Searching site-wide custom styles

Cloudera Data Visualization enables you to search site-wide custom styles by Name, Description, Author, and Modified By fields. You can also search by the content of CSS and Javascript code.

About this task

When you start typing characters in the search text box, the search results display the exact match on the top, followed by the closest matches. You must type at least two characters to start the search.

To demonstrate how to search site-wide custom styles, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Custom Styles.
3. In the Manage Custom Styles interface, create new custom styles or use the existing styles.

   You can find the search text box on the top right.
4. To demonstrate how to search the list of custom styles, enter Pie in the Search text box.

   In this example, the Pie Chart Legend Label custom style is filtered from the list.

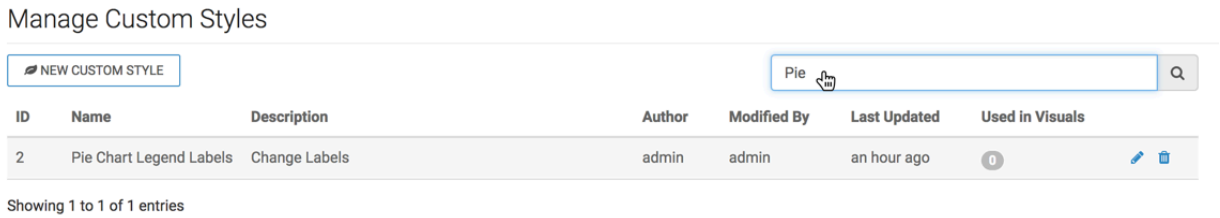

5. To search the list of custom styles by javascript code, follow these steps:
   a) Click the Pencil icon next to the Pie Chart Legend Label to view the javascript code of this style.

      In our example, we search by Not Applicable from the javascript code.

   b) Enter Not Applicable in the Search text box.

      In this example, the Pie Chart Legend Label custom style is filtered out of the list of custom styles.

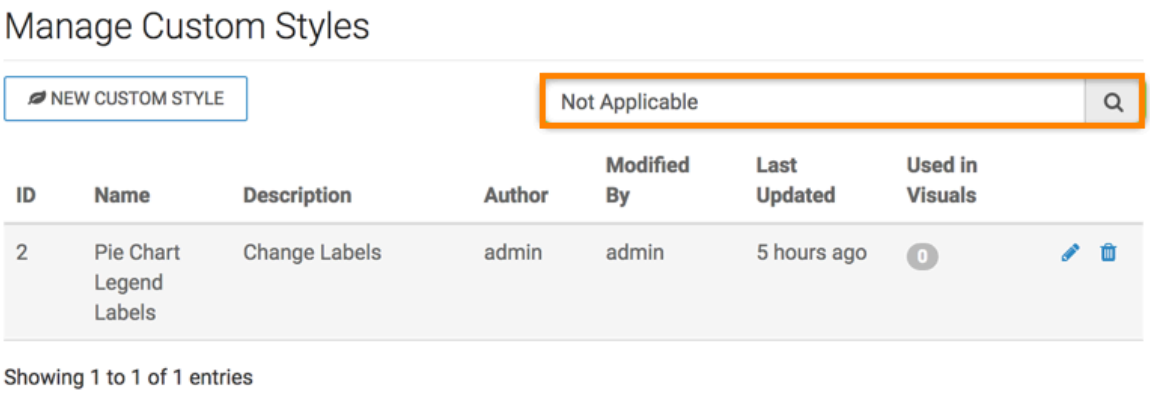

Sorting site-wide custom styles

Cloudera Data Visualization enables you to sort site-wide custom styles by ID, Name, Author, Modified By, Last Updated, and Used in Visuals fields. You can sort in ascending and descending order.

About this task

Note: You cannot sort by the Description field.

To demonstrate how to sort site-wide custom styles, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Custom Styles.
3. In the Manage Custom Styles interface, create new custom styles or use the existing styles.

   In this example, styles are sorted by the name of the custom styles.

4. Click the Name field.

   When you hover over a field, a hand icon appears only on the fields that you can sort.

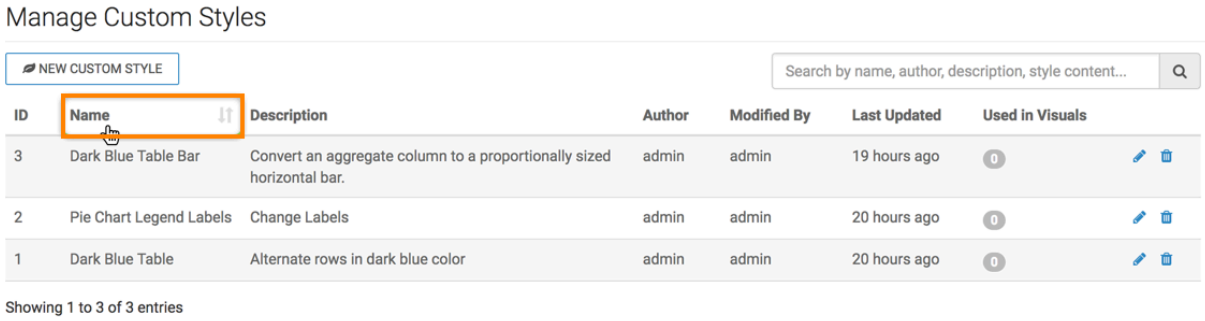

   In the following image, the Name field is sorted in ascending order and a sort icon appears next to the field.

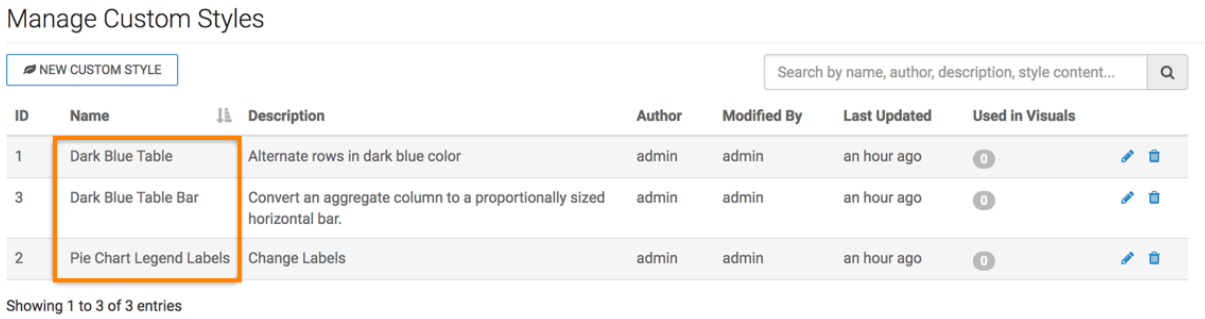

5. To sort the field in descending order, click the Name field again.

Cloudera Data Visualization 7.2.9

Dashboards

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a dashboard 4

Customizing dashboard layout 5

Opening a dashboard 8

Cloning a dashboard 9

Adding a dashboard to favorites 11

Setting a dashboard as your homepage 13

Getting the dashboard URL 15

Emailing a dashboard 17

Downloading a dashboard 20

- Downloading dashboard sheets as PNG files 20
- Downloading dashboards and sheets as PDF files 22
  - Downloading current sheet as a PDF file 23
  - Downloading all sheets as a PDF File 25
  - Downloading current sheet with one visual per page as a PDF file 28

Exporting a dashboard 32

Importing a dashboard 34

Moving a dashboard 38

- Cloning linked visuals 38
- Moving linked visuals 39
- Moving dashboards between workspaces 40

Refreshing visuals manually in dashboards 42

Deleting a dashboard 44

Creating a dashboard

You can easily create new dashboards in Cloudera Data Visualization.

Procedure

1. There are several possible starting points for creating a new dashboard:
   - Click NEW DASHBOARD in the top right corner of the main HOME interface.
   - Click Create New Dashboard in the top right corner of the preview area in the VISUALS interface.
   - Click the icon next to the name of a particular dataset on the DATA interface.
   - Click NEW DASHBOARD in the top right corner of the Dataset Detail view for a particular dataset.

   The Dashboard Designer opens and displays an untitled dashboard. See the *Dashboard Designer* documentation to familiarize yourself with the interface.
2. Change the connection and/or dataset.

   The dashboard is automatically connected to the parent connection and dataset, but you can add visuals and filters based on other connections and datasets.

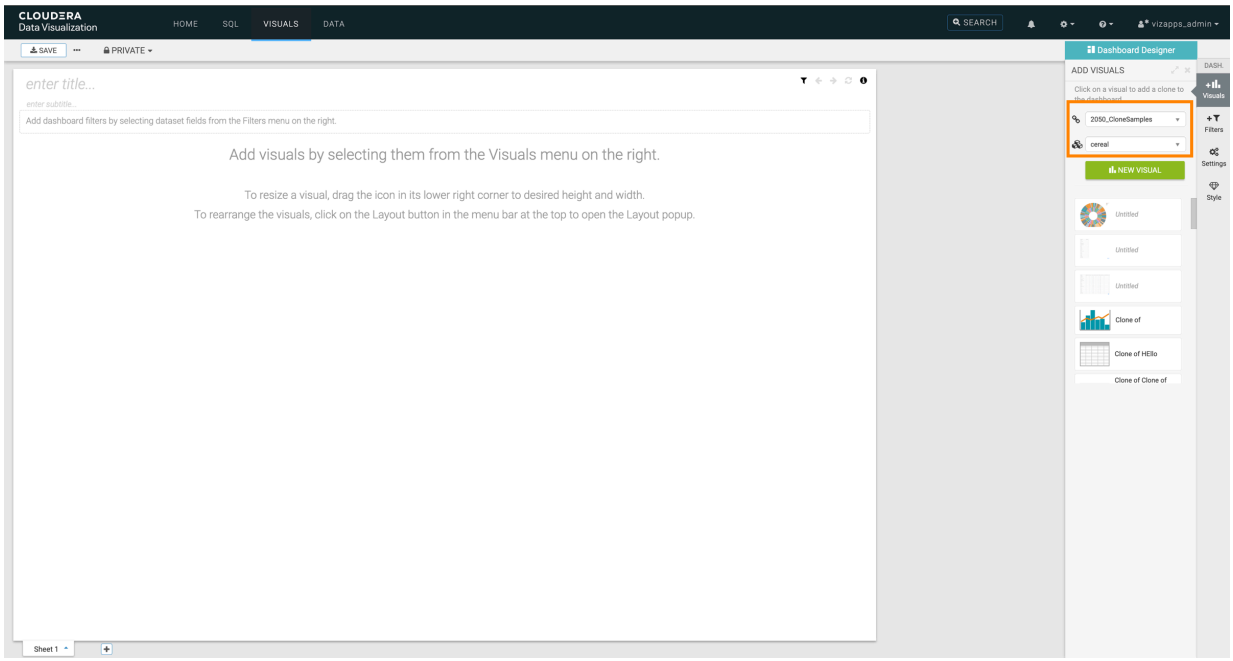

3. Add a title for the dashboard.

   Optionally, you can also add a subtitle.

   In this example, World Life Expectancy and A new perspective are used as title and subtitle.

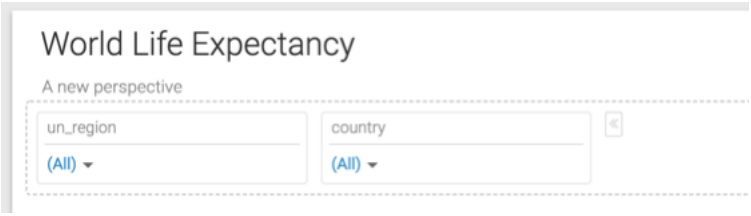

4. You can also customize settings, styles, and create custom styles for the dashboard.

5. In the VISUALS menu, select an existing visual you want to add to the dashboard, either from Recently Viewed Visuals, or from the listed visuals of this dataset. Alternatively, you can create a new visual by clicking NEW VISUAL.
6. Click SAVE to save the dashboard.
7. Click VIEW to preview the dashboard at runtime.

Related Information

Dashboard Designer

Customizing dashboard layout

When you have a dashboard with multiple visuals, you might need to resize and reorganize the visuals on the dashboard to make it work better for your users. In Cloudera Data Visualizationn, you can easily change the layout of a dashboard.

About this task

By default, visuals are positioned sequentially, aligning with the order in which you added them to the dashboard. You can reorganize the layout by resizing or moving around the individual widgets on the dashboard.

Procedure

1. Open the dashboard you wish to modify in Edit mode.
2. On the Dashboard Designer top menu bar, click LAYOUT.

   The Layout of Dashboard Visuals modal window is displayed.

   In the modal window there are widgets that provide the following information about visuals:

   - Top left color squares represent the datasets associated with the visuals.
   - The title of each visual is displayed at the top of its corresponding widget.

     Note: If the widget is too small, the title may not be fully visible.

   - Visual snapshots appear as thumbnails making it easy to identify the individual visuals on the dashboard.

     Note: If a visual has been modified but not saved, the thumbnail displayed maybe be outdated or unavailable. When a thumbnail is unavailable, an icon representing the visual type is used instead.
3. To reorganize visuals on the canvas, click and drag the widgets to the desired positions.
4. To resize a widget, click and drag its bottom-right corner and set the desired dimensions.
5. Once you have achieved the desired layout, click APPLY LAYOUT.

Results

The dashboard layout is updated.

Example

In this example, you can see how the dashboard layout changes after rearranging the widgets on the layout modal.

This is the original dashboard, showing the arrangement of visuals the way it was initially designed:

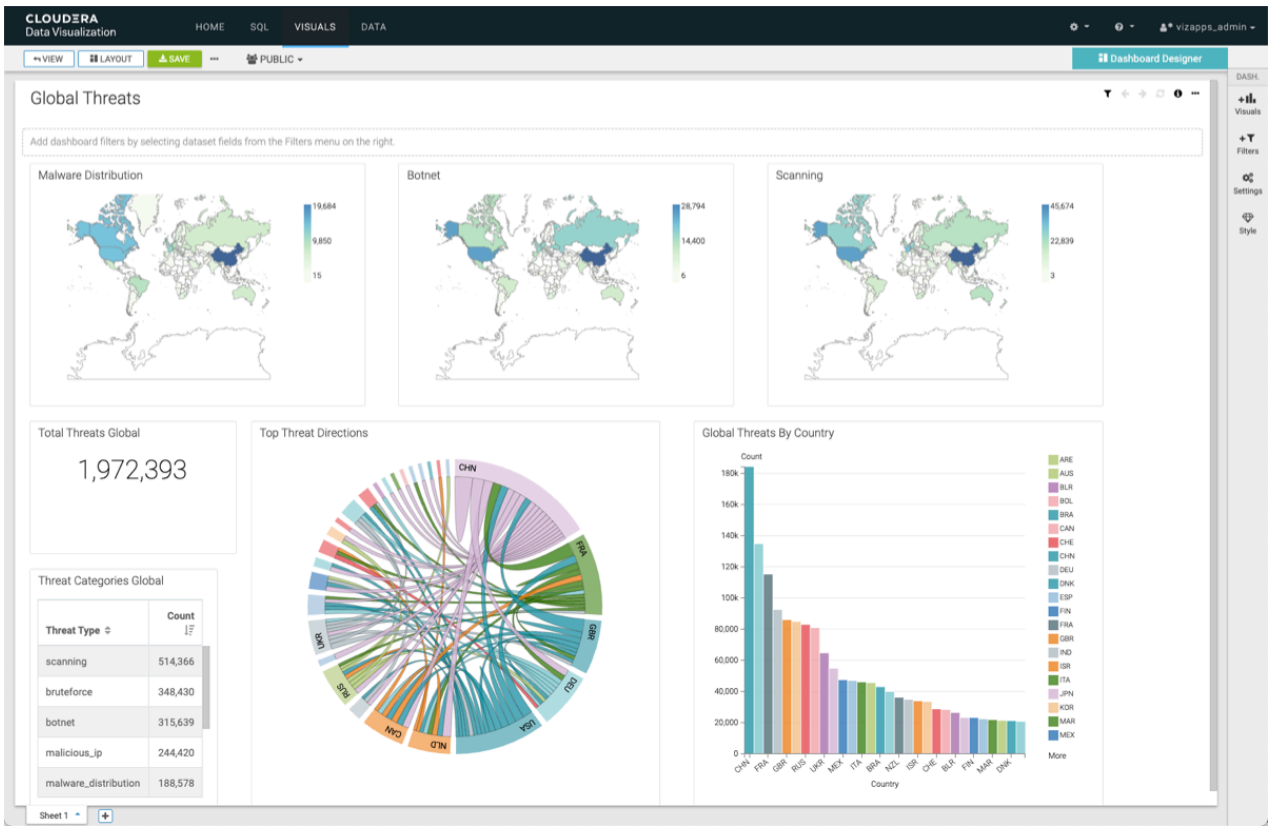

This is the layout modal of the original dashboard:

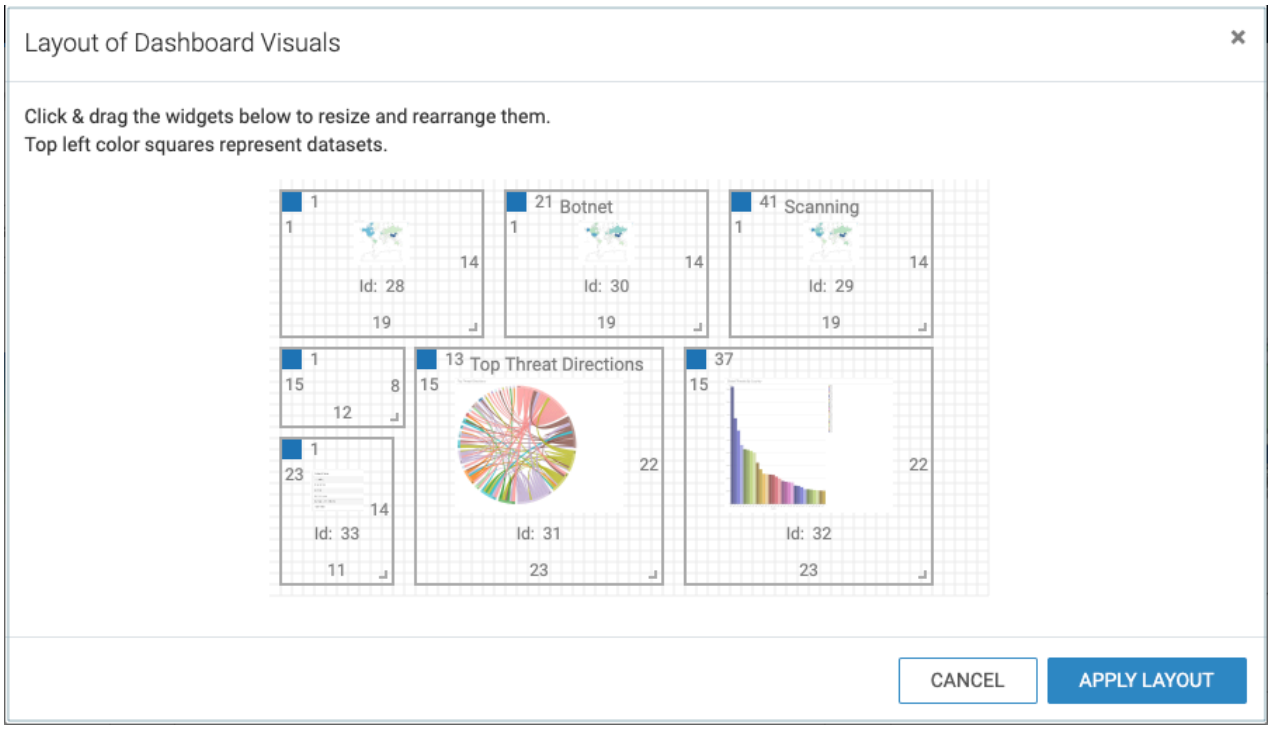

This is the new dashboard layout after rearranging and resizing the widgets representing the visuals in the dashboard:

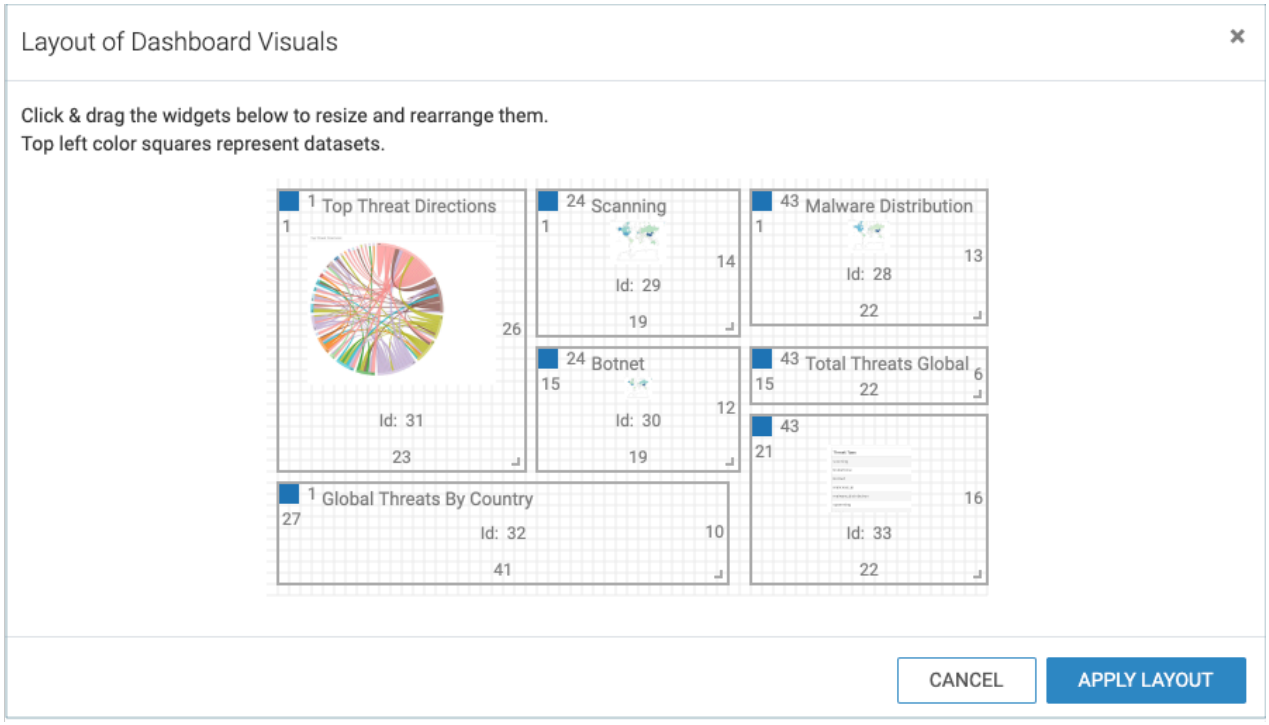

This is the updated dashboard after applying the layout changes:

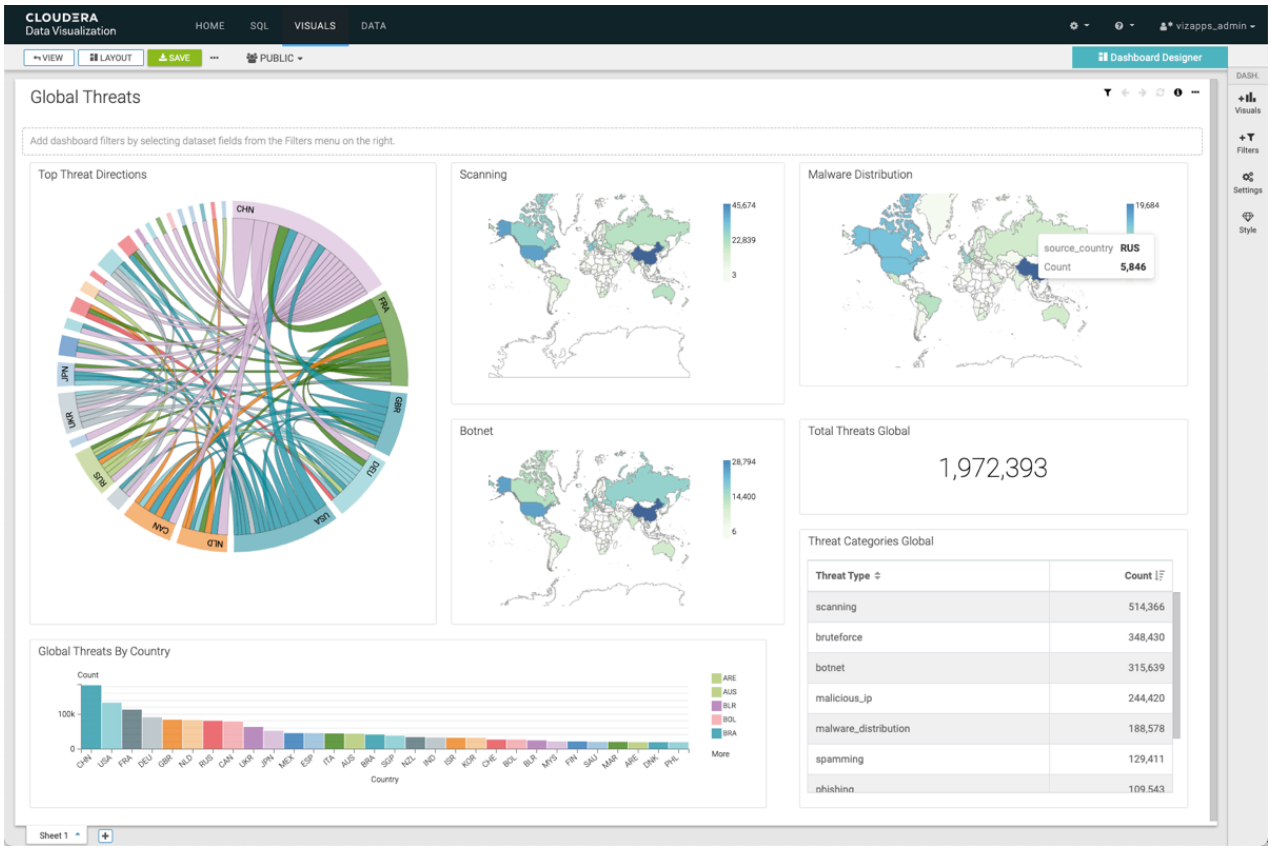

Opening a dashboard

A Cloudera Data Visualization dashboard is a collection of multiple visuals. You can access existing dashboards on the VISUALS interface and open them in either View or Edit mode. If a dashboard is loading slowly, a notification is displayed. This data load message provides a link to the Profile Data modal for more information on the requests and operations in progress.

View mode

View View mode provides a read-only view of a dashboard and its visuals. Despite being read-only, this mode is dynamic and responsive, which is primarily achieved by filters and dashboard controls.

If you open a dashboard in View mode, you can access the following within the dashboard preview area:

1. Visuals displayed on the dashboard
2. Dashboard controls
3. Filters (if added)

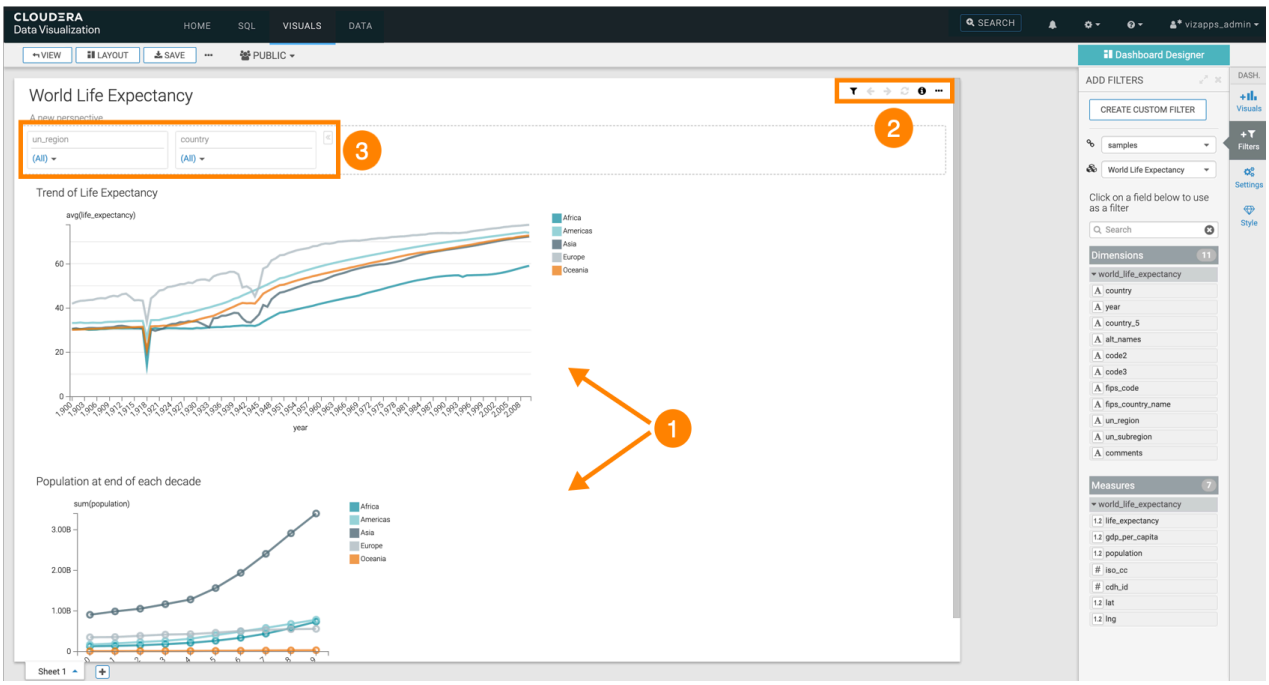

Dashboard controls

Cloudera Data Visualization offers a set of dashboard controls in View mode, some of which have keyboard shortcuts available.

- Current Parameters: Shows the current values of dashboard-level filters.
- Previous Parameters (left arrow icon): Navigates backward through filter settings, resetting the filter values to the previous state. Keyboard shortcut: Ctrl/Cmd + B.
- Next Parameters (right arrow icon): Navigates forward through filter settings, advancing the dashboard to the next state (if available). Keyboard shortcut: Ctrl/Cmd + F.
- Reset Parameters (refresh icon): Resets the dashboard to its initial state with the original filter values. To change these initial values, you must edit the dashboard and save the changes. Keyboard shortcut: Ctrl/Cmd + F.
- Supplemental Menu: Provides options to email the current sheet, or to get the full dashboard URL so you can copy it, share it, or create an alias for it.

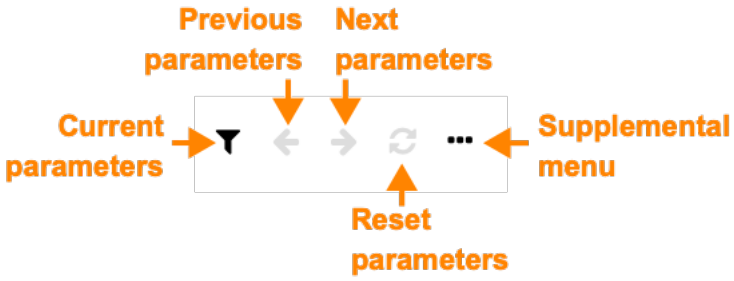

Edit mode

Edit mode is accessible to the dashboard owner and all users who have been granted access permissions. It allows you to edit the dashboard, including editing existing visuals and adding new ones to the dashboard.

To edit an existing dashboard:

- Open it in View mode and click EDIT in the top-left corner.
- On the VISUALS interface, hover over the dashboard tile and click the (pencil) icon in the bottom-right corner.

Cloning a dashboard

In Cloudera Data Visualization, you can create an identical dashboard that you may choose to customize or move to another workspace.

Procedure

1. In the Dashboard Designer top menu bar, click the (ellipsis) icon and select Clone.

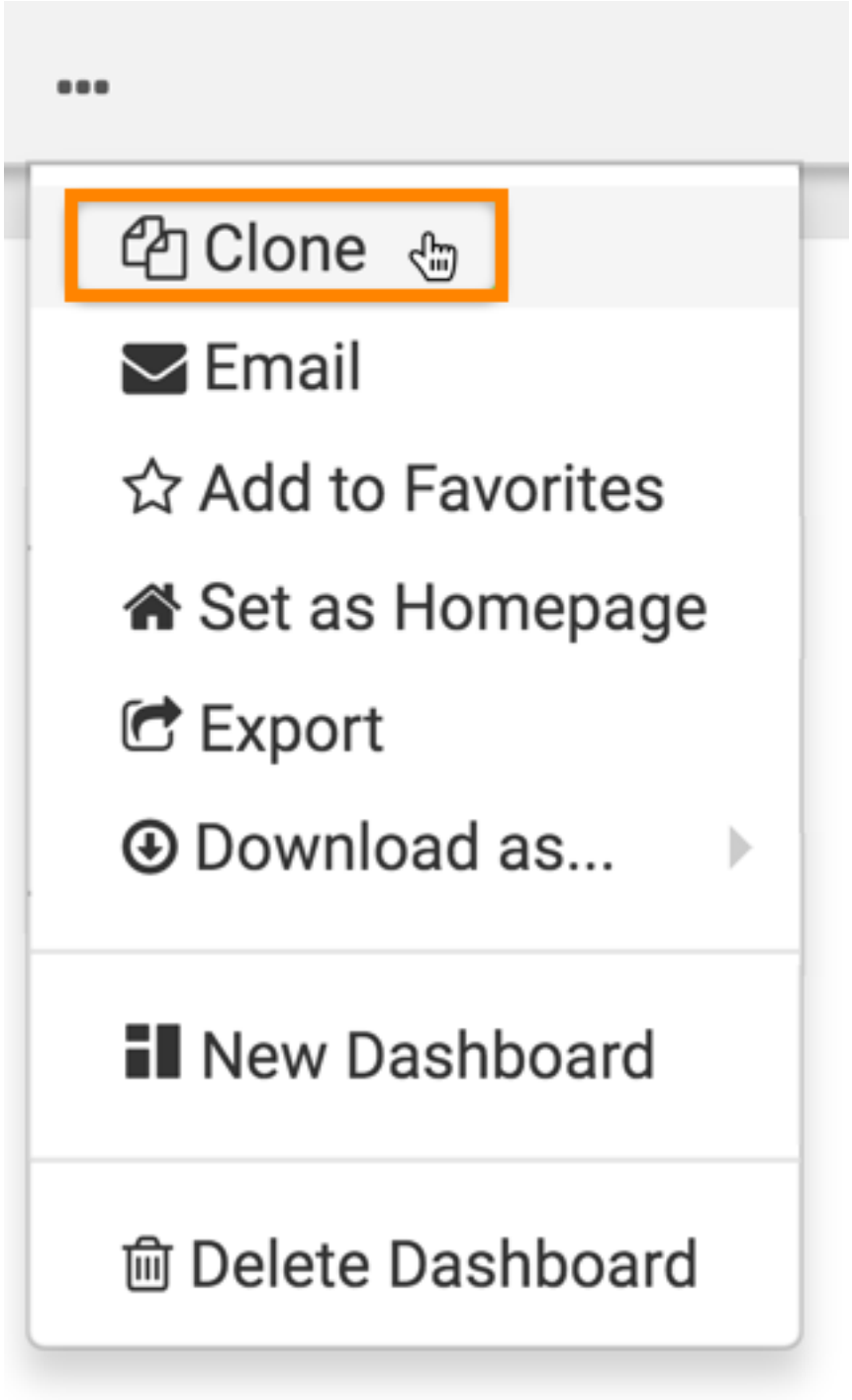

Note: Unsaved changes on the parent dashboard are persisted when cloning the dashboard.

If the dashboard contains linked visuals, the Clone Dashboard with Linked Visuals modal window appears.

2. By default, Cloudera Data Visualization clones all visuals, both linked and dashboard-specific. However, you can choose the Maintain linked association option for any of the linked visuals, and click CLONE NOW.

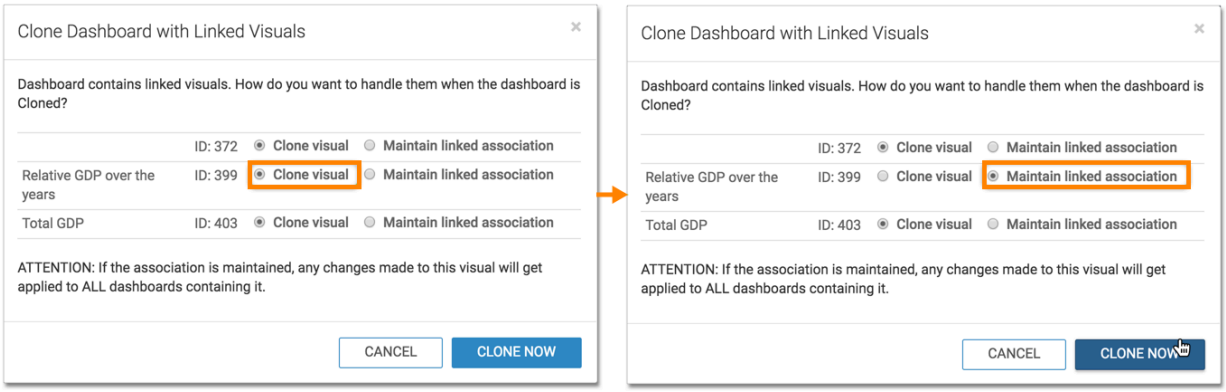

Adding a dashboard to favorites

In Cloudera Data Visualization, you can mark your frequently used dashboards as favorites. These bookmarks make it easy to find your most-used content in a subsequent work session

About this task

When you mark a dashboard as favorite, you can access it quickly from the My Favorites list. Favorites are also visible on the HOME view.

Procedure

1. In the Dashboard Designer top menu bar, click the (ellipsis) icon, and select Add to Favorites.

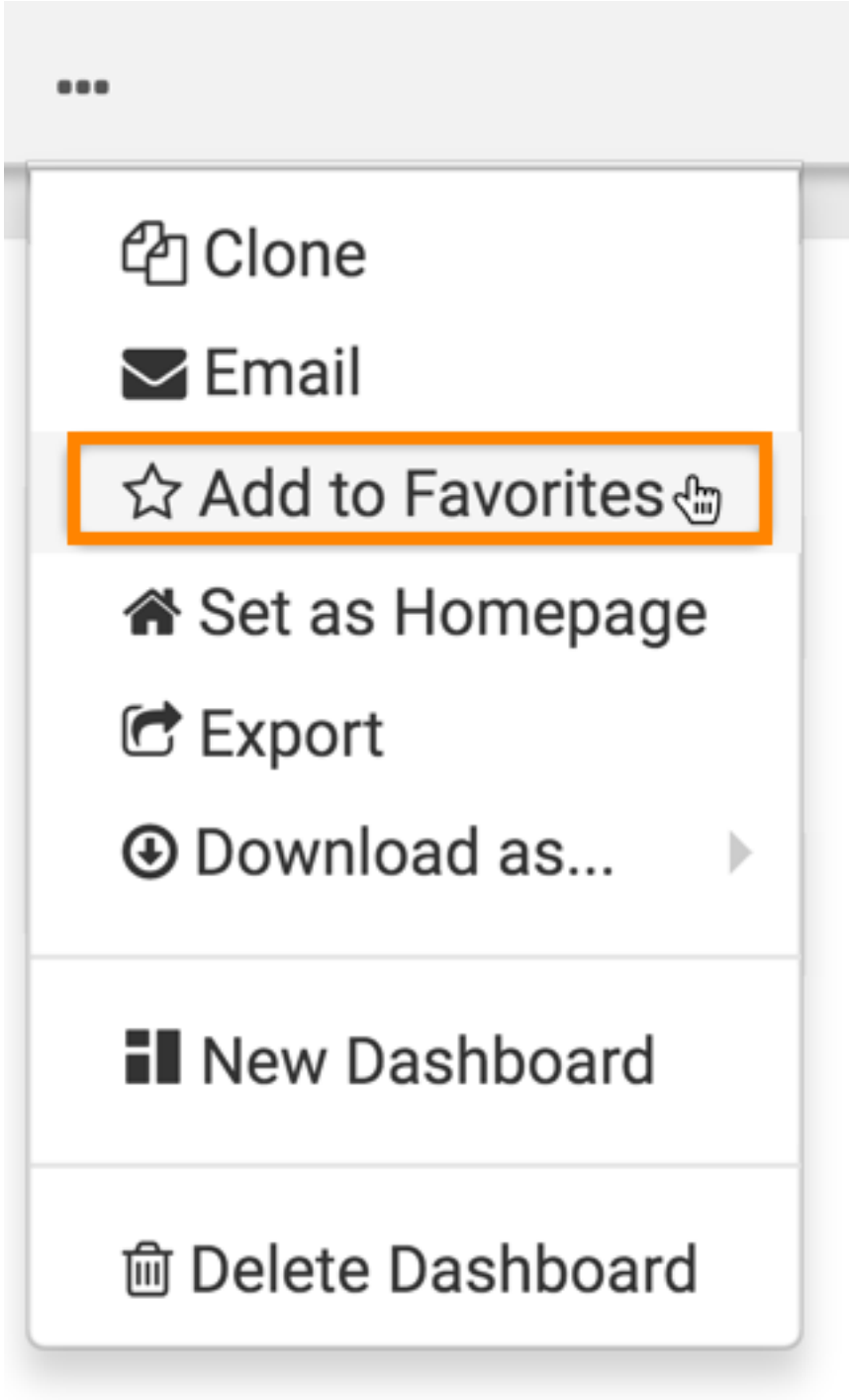

A Success message appears.

You can now find this visual in the Favorites list. All dashboards marked as favorite appear in the interface.

2. To view your favorite dashboards, click VISUALS on the main navigation bar and click My Favorites in the side menu bar.

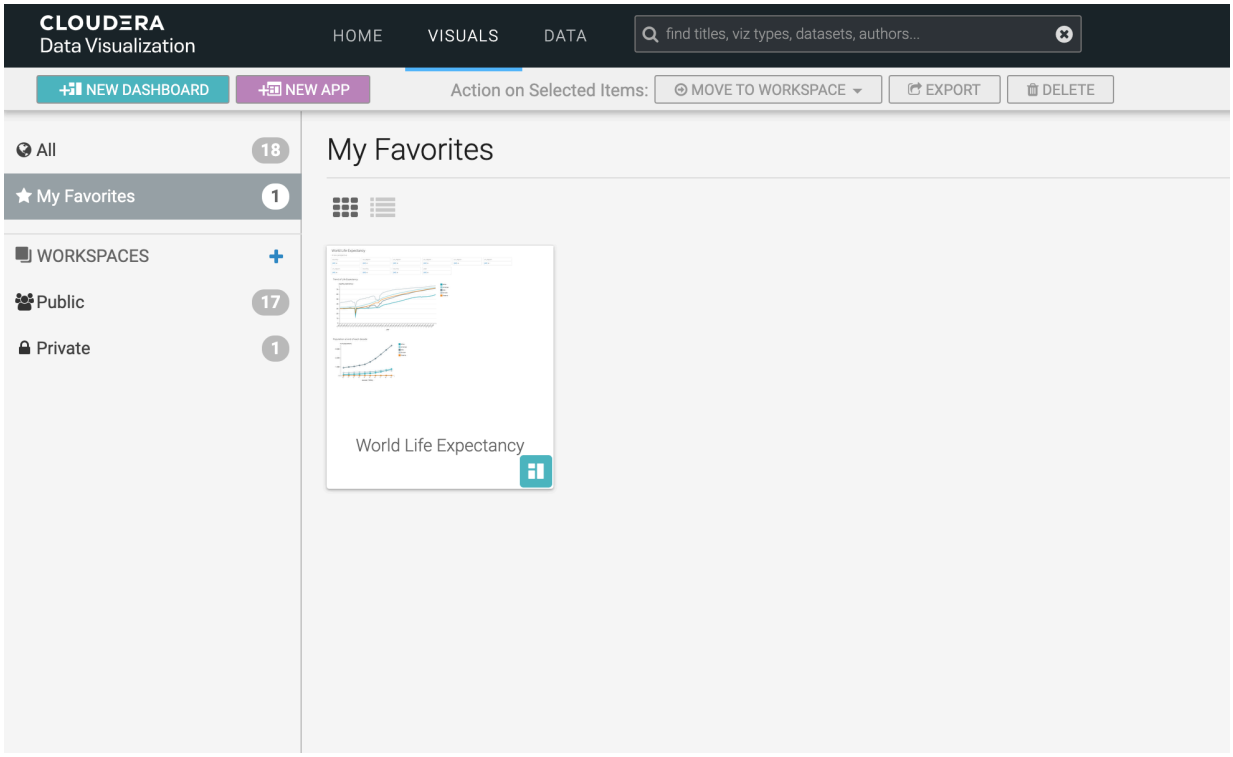

3. If you no longer use a dashboard as often as before, you can unfavorite it.

   When you unfavorite content, it is removed from your My Favorites list but not from CDV.

   Favorites is a toggle setting. In the Dashboard Designer top menu bar, click the (ellipsis) icon and select Remove from Favorites.

Setting a dashboard as your homepage

In Cloudera Data Visualization, you can change the home screen to the currently viewed or edited dashboard.

Before you begin

You need edit privileges to perform this task.

Procedure

1. Click the supplemental menu in the top menu bar and select Set as Homepage.

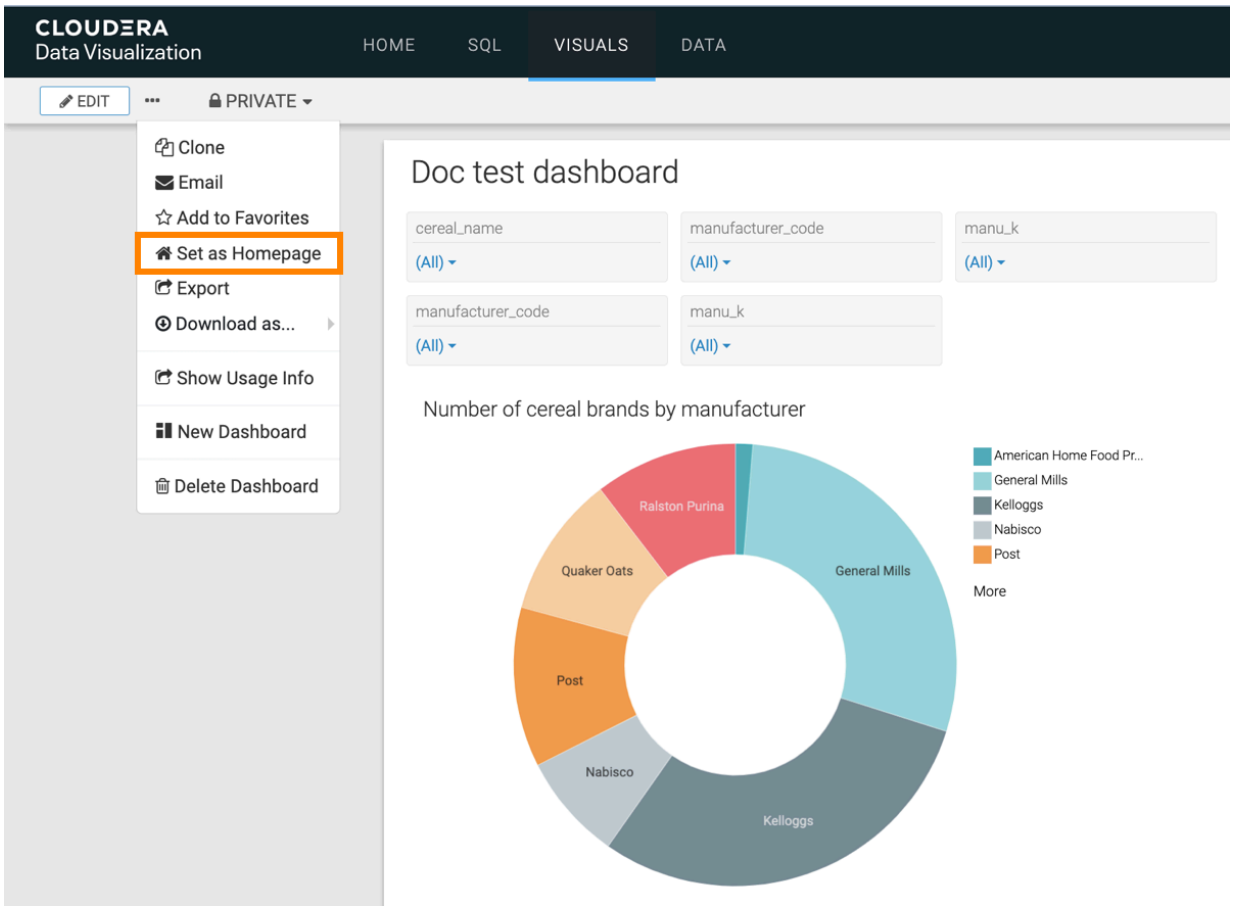

A message confirming the success of setting the dashboard appears briefly.

2. Click HOME on the main navigation bar.

The dashboard that you have specified as homage is displayed.

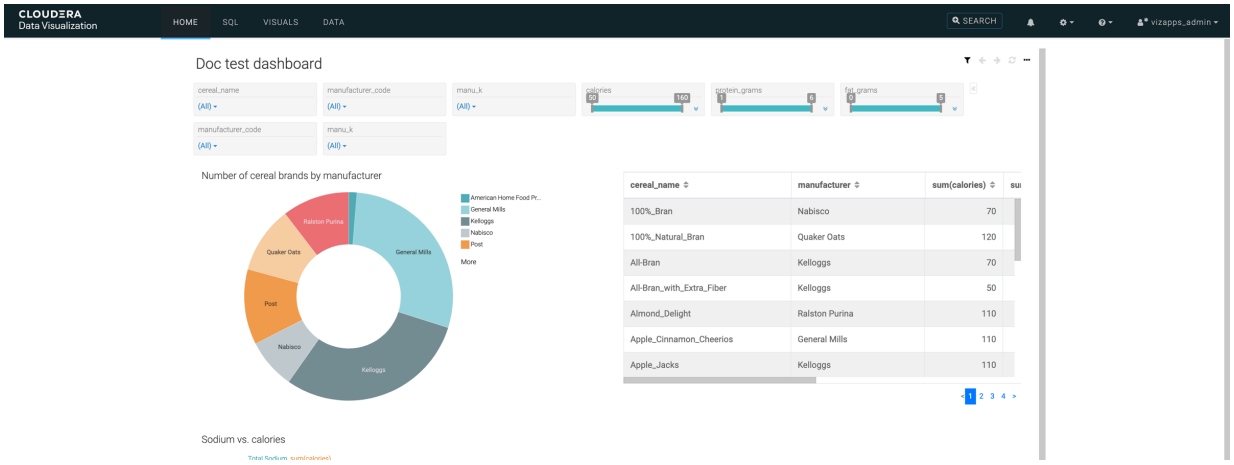

Getting the dashboard URL

Cloudera Data Visualization enables you to share a dashboard by providing a direct link to it. You can share the default URL of the dashboard or use a short version of the link. You also have the option to customize the short URL, which allows you to use more descriptive and more readable/memorable links. You can create your short links for example to indicate where the link leads or what the dashboard contains.

About this task

Short URLs can help you when adding dashboard links in emails or embedding the links in applications. When the user clicks the short version, they are automatically forwarded to the destination URL. The Get url action is available by default in Cloudera Data Visualization.

Procedure

1. Open a dashboard.
2. Click the (ellipsis) icon in the top right corner of the dashboard to open the supplemental menu and click Get URL.

   The Dashboard URL modal window is displayed.

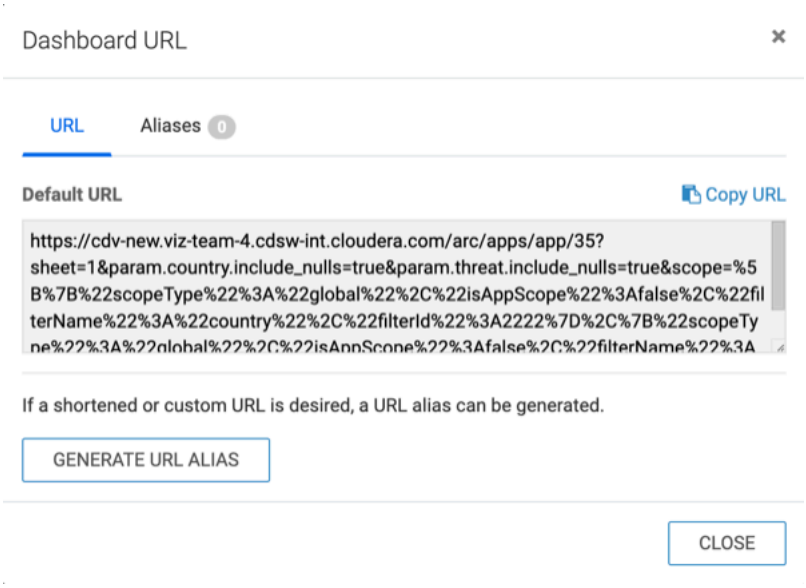

3. If you want to use the default URL, click Copy URL to copy the full URL to your clipboard.

   If you want to use a short URL, follow these steps:

   a. Click GENERATE URL ALIAS.

      A random short URL will be generated. You can customize the short URL by changing the last section to your custom string. Make sure to only use allowed characters (A-Z a-z 0-9 _-)

      Note: If you are trying to add an alias that is already being used, CDV will ask you to change it to a unique one.

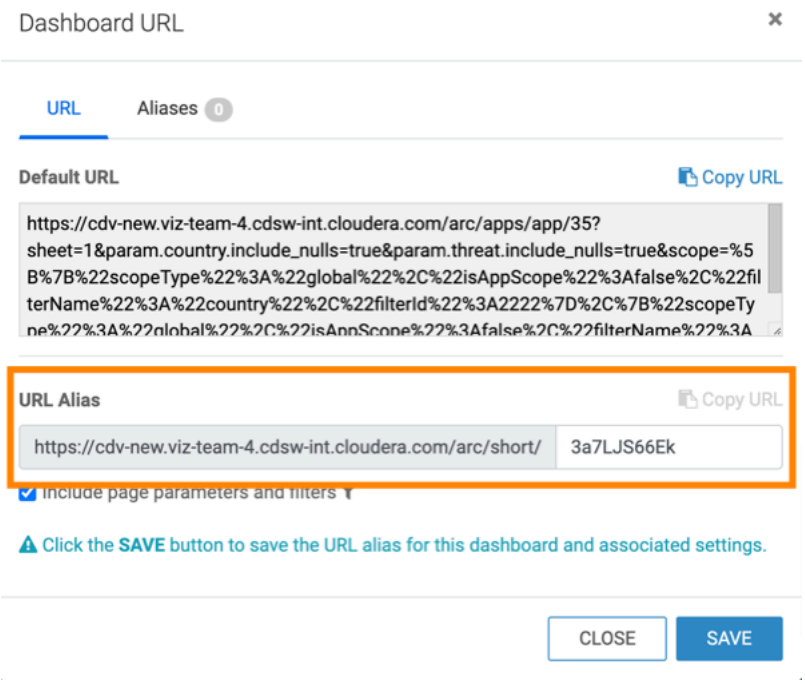

   b. Decide if you want to include the dashboard parameters and filters in the short URL and set the option accordingly.

   c. Click SAVE.

      - The URL alias and the associated settings are saved.
      - The Copy URL option becomes available for the newly created URL alias.
      - The new URL alias is added to the Aliases tab.

        It lists all short URLs created for the dashboard.

        - shows you the parameters and filters included.
        - allows you to delete the short URL.
        - allows you to copy the short URL to your clipboard.

   d. Click Copy URL above the URL alias on the URL tab to copy the short URL to your clipboard.

   e. Click CLOSE.

   This example shows an alias where a custom URL is used and a dashboard filter has been included in the generated URL.

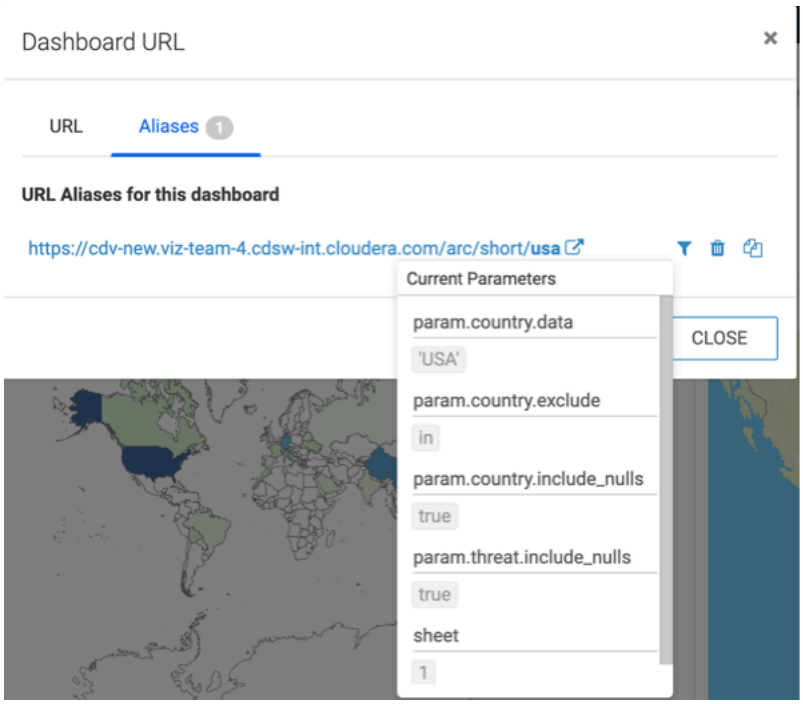

Emailing a dashboard

Cloudera Data Visualization enables you to share dashboards through email. You can send them immediately, schedule them at specific times and intervals, or through a triggered response when reaching a specific monitored threshold.

Before you begin

Before using the email function, you must enable the feature in site settings. For instructions, see *Email settings*.

Note: This setting is only available for users with administrative privileges.

Complete these prerequisites in your system configuration:

- Enable screenshot and thumbnail downloads. For instructions, see *Snapshots*.
- Enable email functionality. For instructions, see *Email settings*.
- Enable scheduled jobs. For instructions, see *Enabling scheduled jobs*.

Note: You can only email one dashboard sheet at a time, but if you have more sheets on your dashboard, you can repeat this procedure for each sheet.

Procedure

1. Find the dashboard that you want to email. Open it either in Edit or in View mode.

2. In the top menu bar, click the ••• icon to open the supplemental menu and select Email.

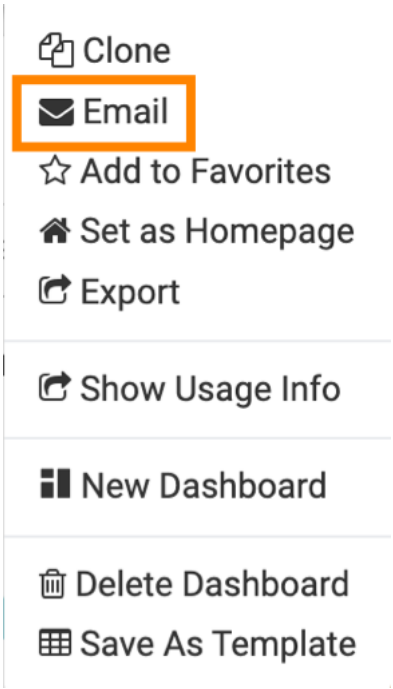

The Send sheet modal window appears.

3. Choose one of the two options for sending email.

   Email now – For instructions on how to provide email details, see *Immediate email*.

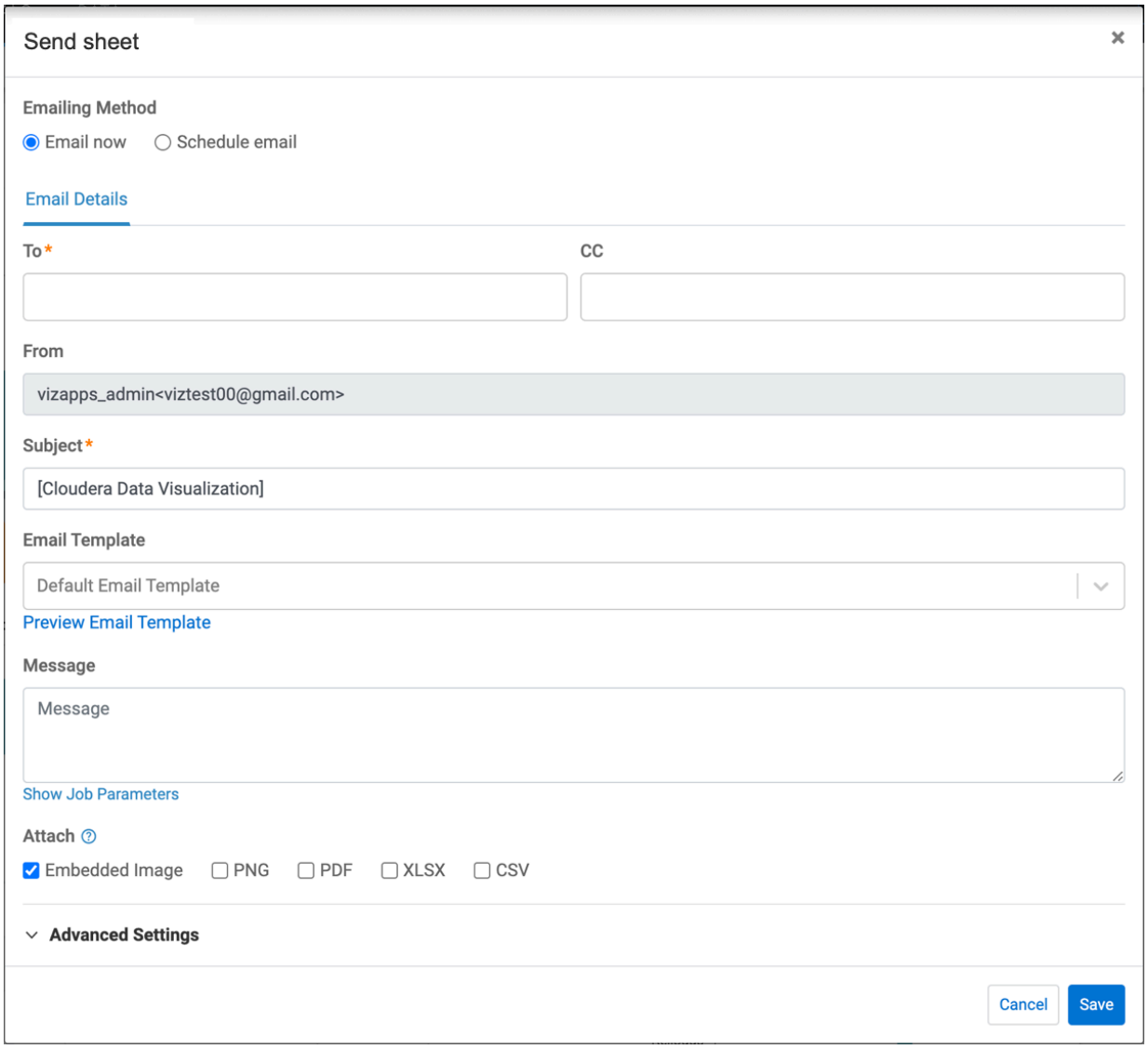

   Schedule email – For instructions on how to provide email and schedule details, see *Schedule-based email*.

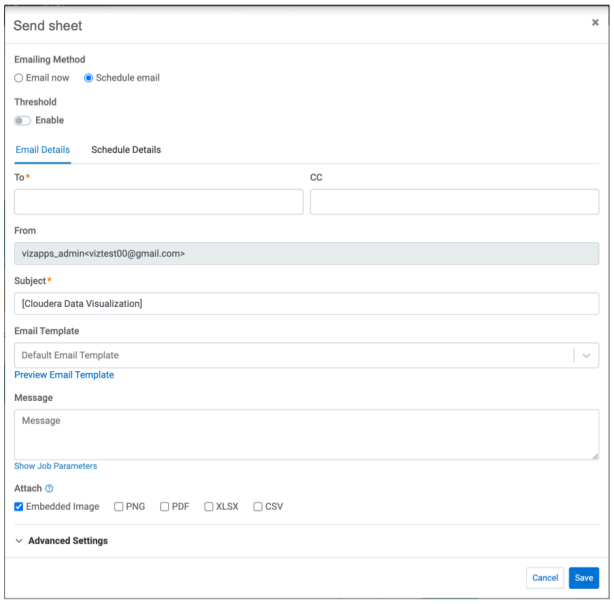

4. After creating emails, you can monitor them in the Jobs interface, review their details, rerun, cancel, and so on. For more information, see *Managing jobs*.

Related Information

Email settings
Snapshots
Enabling scheduled jobs
Immediate email
Schedule-based email
Threshold-based email
Managing jobs

Downloading a dashboard

Cloudera Data Visualization enables you to save the current sheet of a dashboard in different file formats.

Downloading dashboard sheets as PNG files

Cloudera Data Visualization enables you to save the current sheet of a dashboard as a PNG file. The filename is the same as the name of the dashboard, and it has a timestamp specifying the date and time when the file is created.

About this task

This method is better than using a desktop tool because it handles images that are too long or wide to fit the screen.

Note:

- This feature is available only when thumbnails are enabled.
- For snapshots of custom JS visuals, Cloudera Data Visualization supports only ECMA Script 5 for Javascript.
- Cloudera Data Visualization does not support PNG download on the Windows platform.

Procedure

1. In the Dashboard Designer top menu bar, click the (ellipsis) icon.
2. Click Download as..., and select Image (png) from the secondary menu.

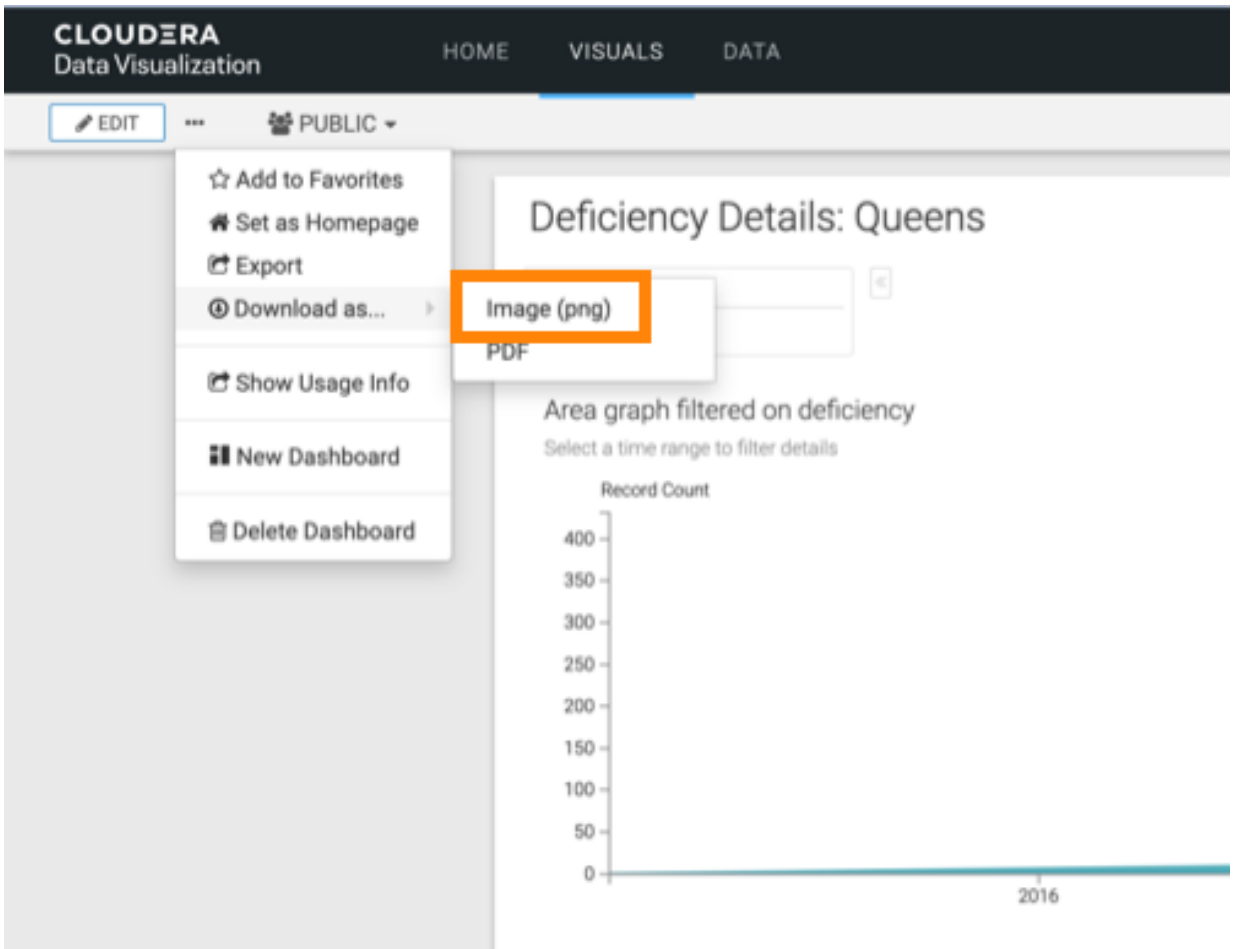

A Generating the requested file message appears.

3. After the download is complete, open the default download folder on your computer.

   Cloudera Data Visualization saves the current sheet to a PNG file.

   In our example, the generated filename, Example close_open issue_20190207094420, has the following components:

   - *Example close_open issue*

     Same name as the dashboard
   - *20190207*

     Signifies the date the file is created, in YYYYMMDD format
   - *094420*

     Signifies the time the file is created, in hhmmss format

   Note: Cloudera Data Visualization renames sheets with names that contain characters not supported by the file system. For example, for the sheet Example close/open issue, it generates the file as Example close_open issue_xxxxxxxxxxxxxx.pdf, replacing the forward-slash (ASCII #47) character with a space.

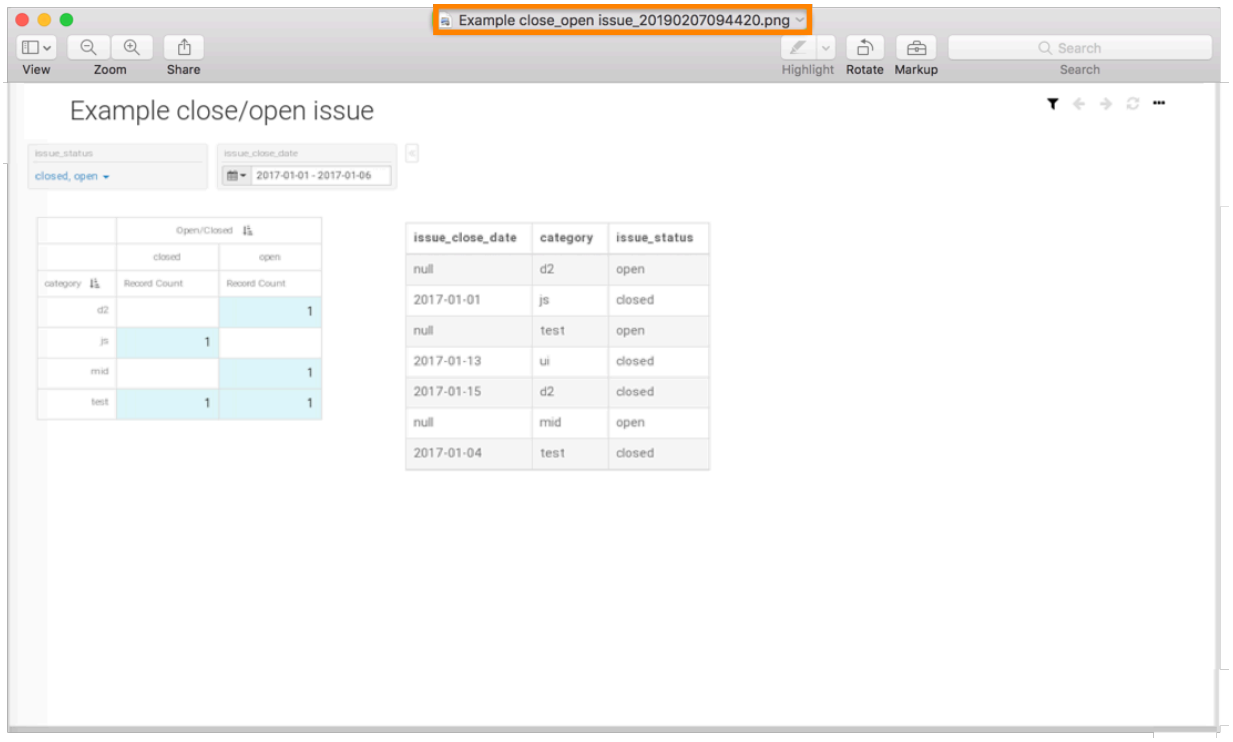

Downloading dashboards and sheets as PDF files

Cloudera Data Visualization enables you to save your dashboards, and individuals sheets as PDF files.

About this task

Note:

- This feature is available only when thumbnails are enabled.
- For snapshots of custom JS visuals, we support only ECMA Script 5 for Javascript.

Before you begin

Before you download the dashboard as a PDF file, under Settings, navigate to the General menu, and select the Enable "Download as Image/PDF" option.

Procedure

1. In a dashboard, click the (ellipsis) icon at the top left corner of the interface.
2. Click Download as..., and select PDF from the secondary menu.

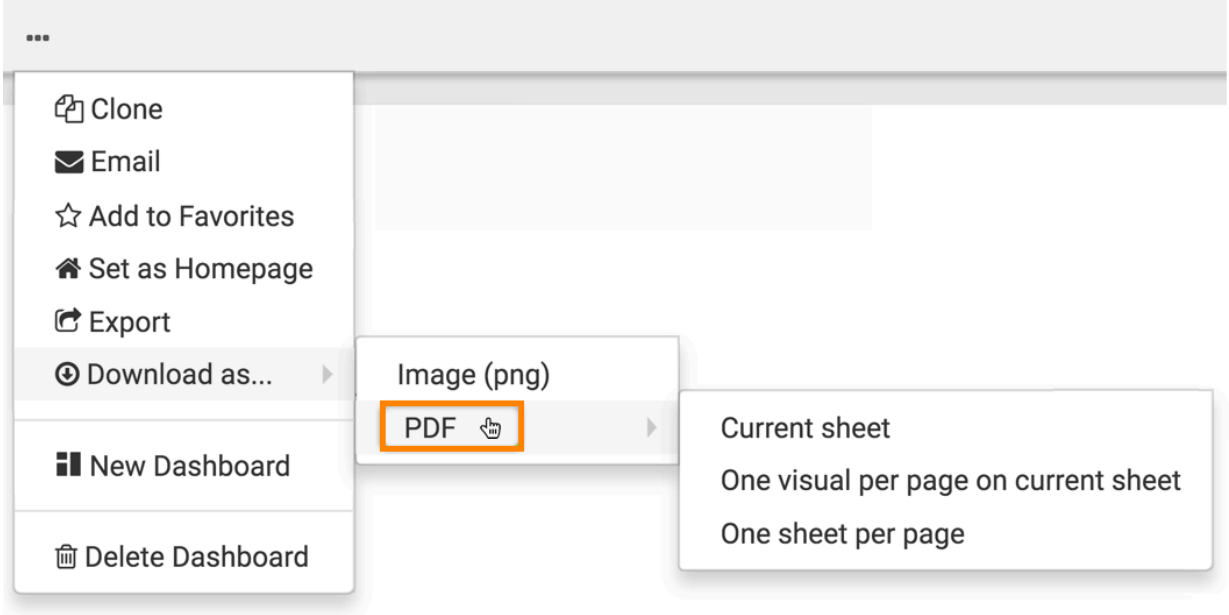

3. From the dropdown menu, select one of the following options:
   - Current sheet
   - One visual per page on current sheet
   - One sheet per page

Downloading current sheet as a PDF file

Cloudera Data Visualization enables you to download the current sheet of a dashboard as a PDF file with all visuals in the current sheet on one page.

About this task

Note:

- This feature is available only when thumbnails are enabled .
- For snapshots of custom JS visuals, we support only ECMA Script 5 for Javascript.
- For filtered visuals, the filter must have app scope.

Before you begin

Before you download the dashboard as a PDF file, make sure that the Enable "Download as Image/PDF" option is enabled. You can find it in Site Administration Site Settings Snapshots .

Procedure

1. Open any dashboard.
2. Click the (ellipsis) icon at the top left corner of the interface.

3. Select Download as... PDF Current Sheet .

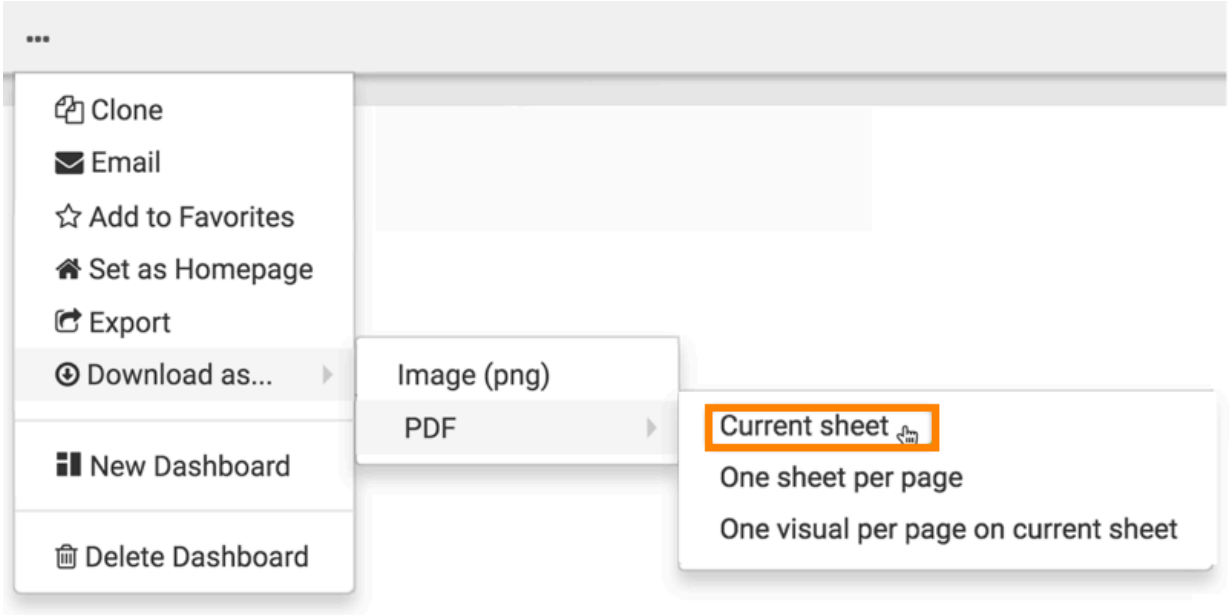

A Generating the requested file message appears.

4. After the download is complete, open the default download folder on your computer. Cloudera Data Visualization saves the current sheet to a PDF file.

   In this example, the generated filename, PDF Export_20190205135933, has the following components:

   - *PDF Export*

     Same name as the dashboard
   - *20190205*

     Signifies the date the file is created, in YYYYMMDD format
   - *135933*

     Signifies the time the file is created, in hhmmss format

   Note: Cloudera Data Visualization renames sheets with names that contain characters not supported by the file system. For example, for the sheet PDF/Export, it generates the file as PDF Export_xxxxxxxxx xxxxx.pdf, replacing the forward-slash (ASCII #47) character with a space.

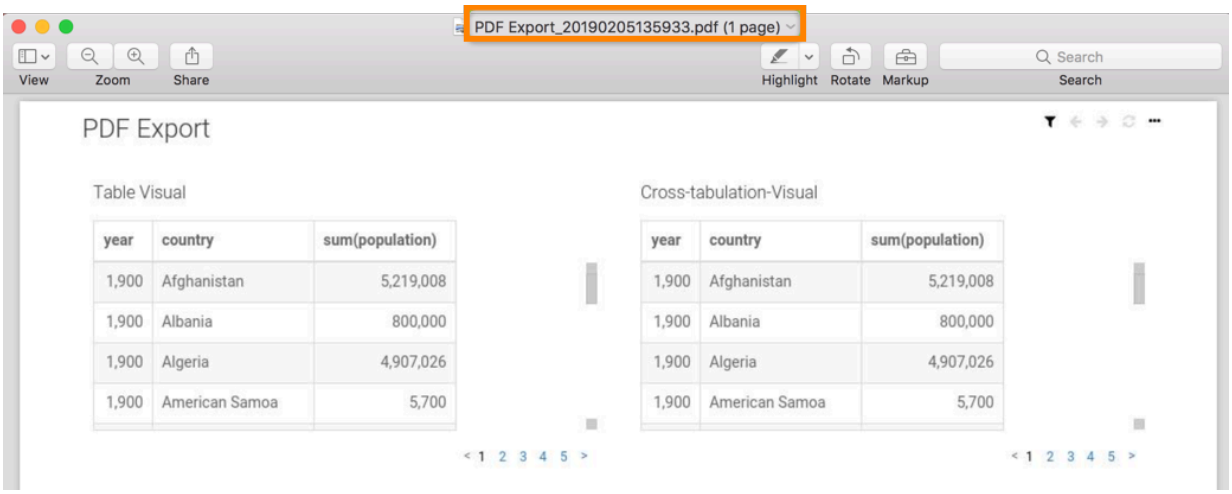

The filename is the same as the name of the dashboard, and it has a timestamp specifying the date and time when the file is created.

5. All visuals in the current sheet (Sheet 1) are on one page. Repeat the preceding steps to download Sheet 2.

Downloading all sheets as a PDF File

Cloudera Data Visualization enables you to download the entire dashboard as a PDF file with each sheet on a separate page.

About this task

Note:

- This feature is available only when thumbnails are enabled .
- For snapshots of custom JS visuals, we support only ECMA Script 5 for Javascript.

In the following example, you can see how to download Sheet 1 and Sheet 2 of the dashboard on separate pages.

Before you begin

Before you download the dashboard as a PDF file, make sure that the Enable "Download as Image/PDF" option is enabled. You can find it in Site Administration Site Settings Snapshots .

Procedure

1. Open any dashboard. In this example, the PDF Export dashboard that has two sheets is used.

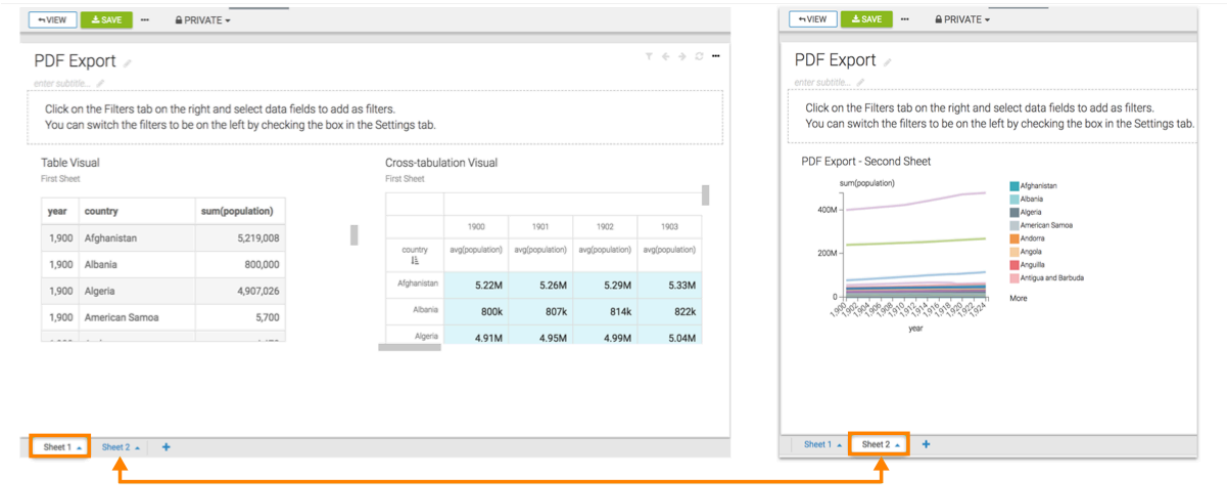

2. Click the (ellipsis) icon at the top left corner of the interface.

3. Select Download as... PDF One sheet per page .

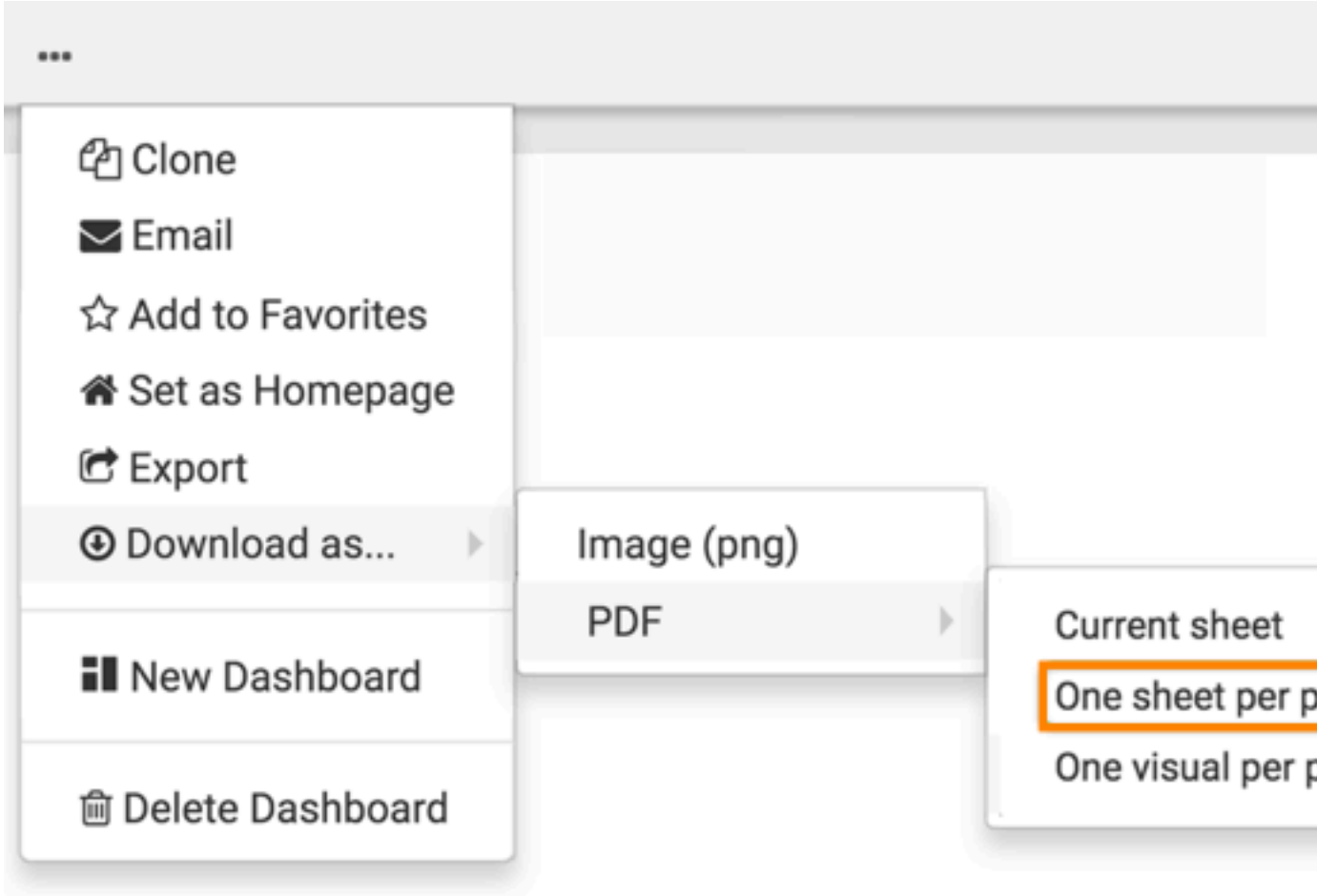

A Generating the requested file screenshot message appears.

4. After the download is complete, open the default download folder on your computer. Cloudera Data Visualization saves the current sheet to a PDF file.

   In this example, the generated filename, PDF Export_20190205142719, has the following components:

   - *PDF Export*

     Same name as the dashboard
   - *20190205*

     Signifies the date the file is created, in YYYYMMDD format
   - *142719*

     Signifies the time the file is created, in hhmmss format

   Note: Cloudera Data Visualization renames sheets with names that contain characters not supported by the file system. For example, for the sheet PDF/Export, it generates the file as PDF Export_xxxxxxxxx xxxxx.pdf, replacing the forward-slash (ASCII #47) character with a space.

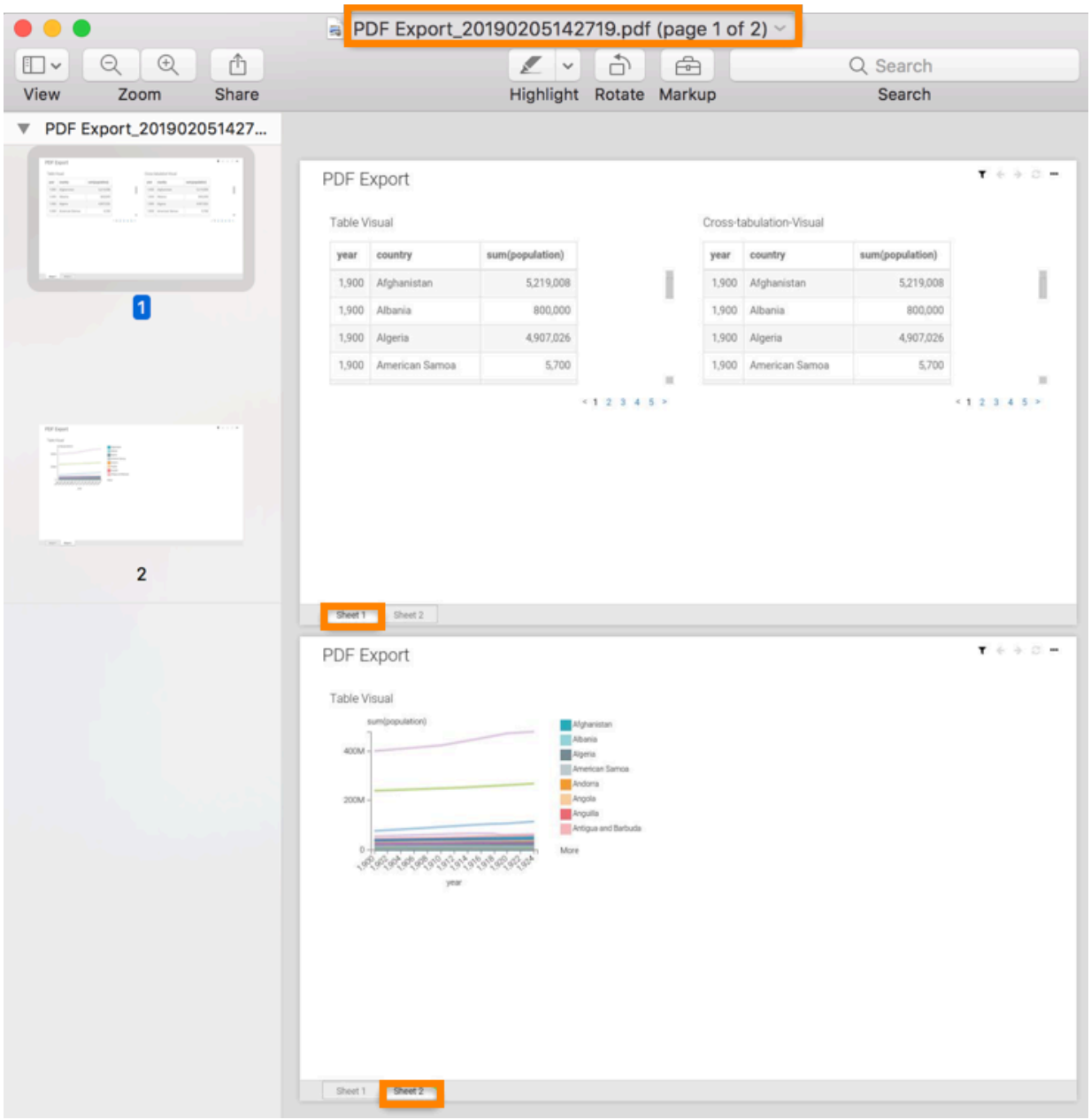

The filename is the same as the name of the dashboard, and it has a timestamp specifying the date and time when the file is created. Each sheet of the dashboard is on a separate page.

Downloading current sheet with one visual per page as a PDF file

Cloudera Data Visualization enables you to download the current sheet as a PDF file with each visual on a separate page.

About this task

In the following example, you can see how to download a PDF file with each visual of the current sheet on a separate page.

Before you begin

Before you download the dashboard as a PDF file, make sure that the Enable "Download as Image/PDF" option is enabled. You can find it in Site Administration Site Settings Snapshots .

Procedure

1. Open any dashboard. In this example, the PDF Export dashboard is used, which has two visuals.

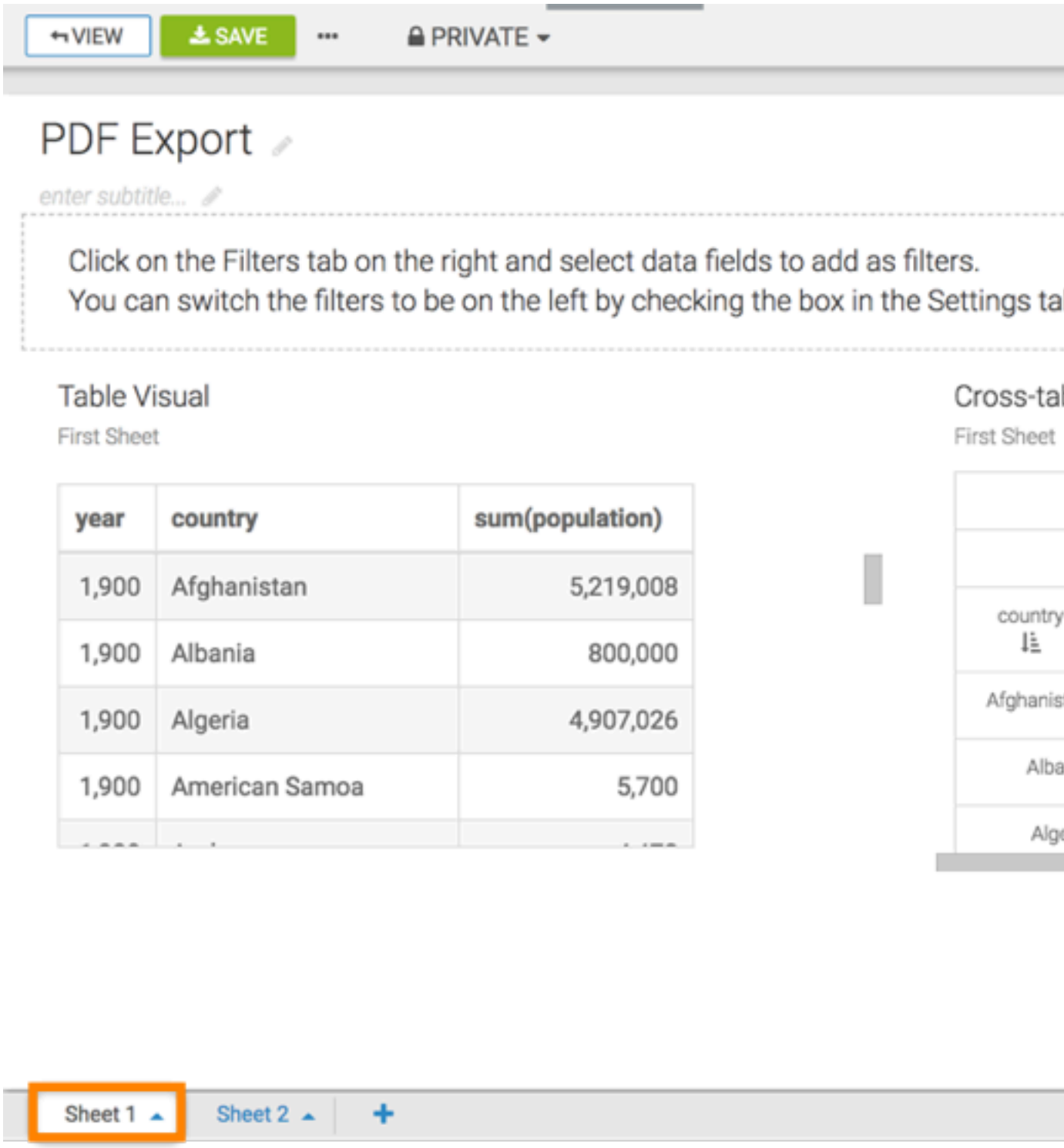

2. Click the (ellipsis) icon at the top left corner of the interface.

3. Select Download as... PDF One visual per page on current sheet .

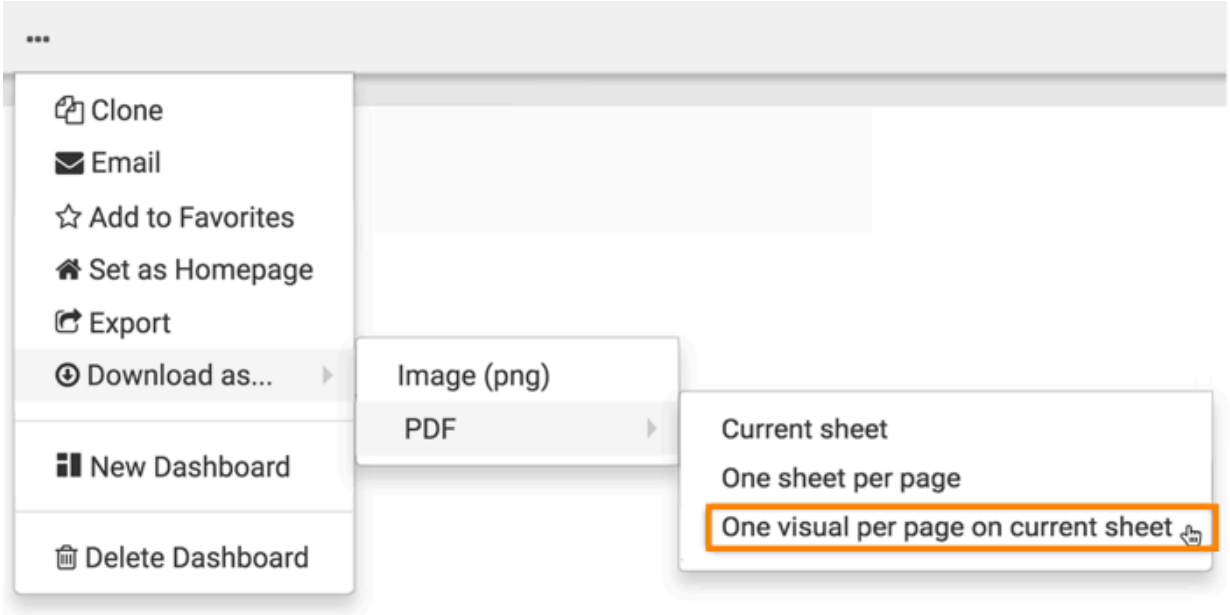

A Generating the requested file screenshot message appears.

4. After the download is complete, open the default download folder on your computer. Cloudera Data Visualization saves the current sheet to a PDF file.

   In this example, the generated filename, PDF Export_20190205142003, has the following components:

   - *PDF Export*

     Same name as the dashboard
   - *20190205*

     Signifies the date the file is created, in YYYYMMDD format
   - *142003*

     Signifies the time the file is created, in hhmmss format

   Note: Cloudera Data Visualization renames sheets with names that contain characters not supported by the file system. For example, for the sheet PDF/Export, it generates the file as PDF Export_xxxxxxxxx xxxxx.pdf, replacing the forward-slash (ASCII #47) character with a space.

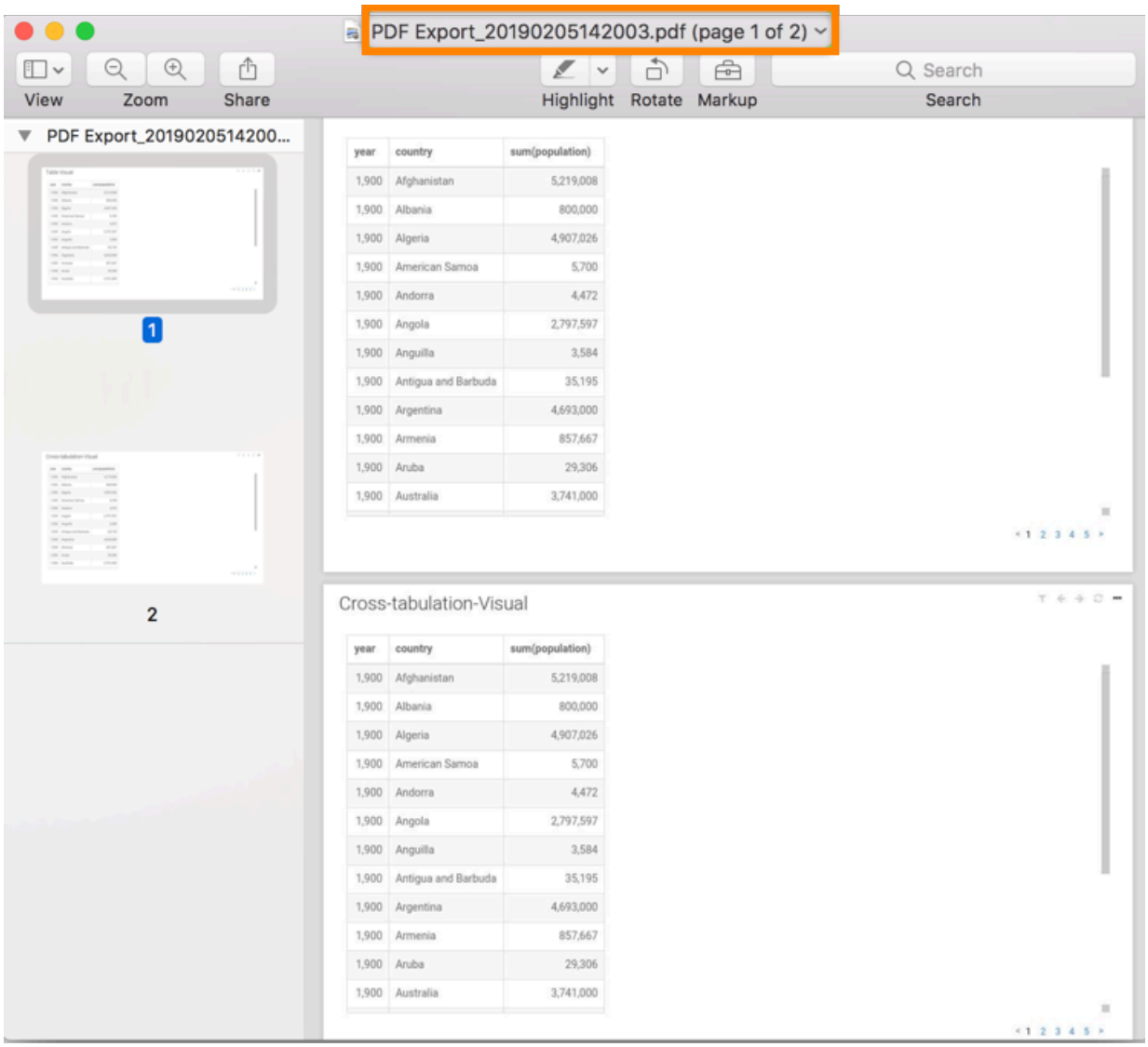

The filename is the same as the name of the dashboard, and it has a timestamp specifying the date and time when the file is created. Each visual in the current sheet is on a separate page.

Exporting a dashboard

In Cloudera Data Visualization, you can export visuals, dashboards and other visual artifacts. You can use the export function to port these artifacts from development/testing environments to production deployments or for creating back-ups.

About this task

Exporting a dashboard from a Cloudera Data Visualization instance generates a JSON file and saves it into your download directory. You can use this file can to import the visual artifacts to a new system, and deploy them if the new environment has identical data connections.

Note: You need the Manage visuals and dashboards privilege to export visual artifacts.

Procedure

1. Find the dashboard you want to export. Open it either in Edit or View mode.

2. Open the (supplemental) menu from the top menu bar and click Export.

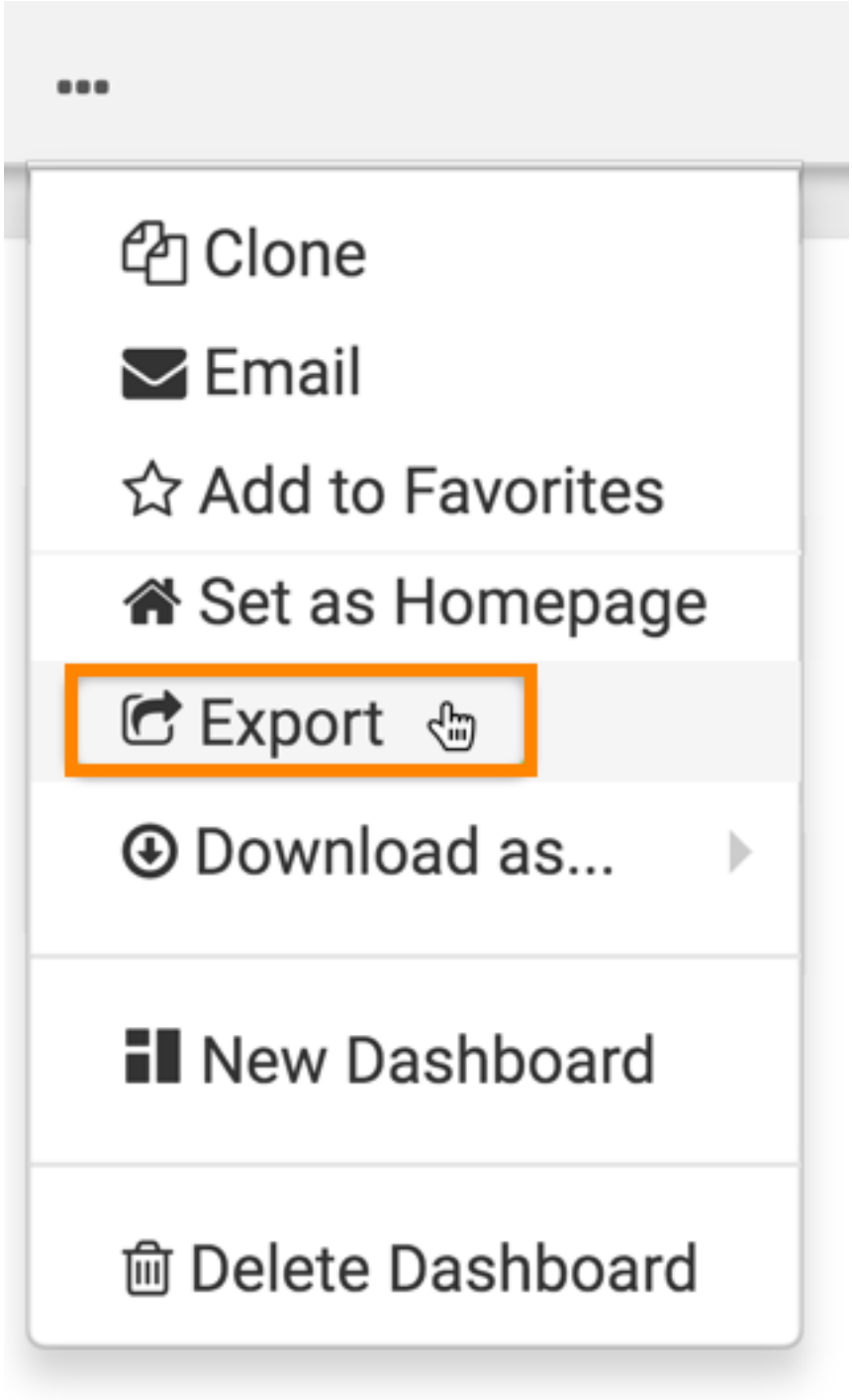

The Export Visual Artifacts modal window appears.

Note: In addition to the dashboard itself, the utility exports all dashboard elements: visuals, datasets, static assets, custom styles, and custom color.

3. Click EXPORT to confirm the export of the dashboard.

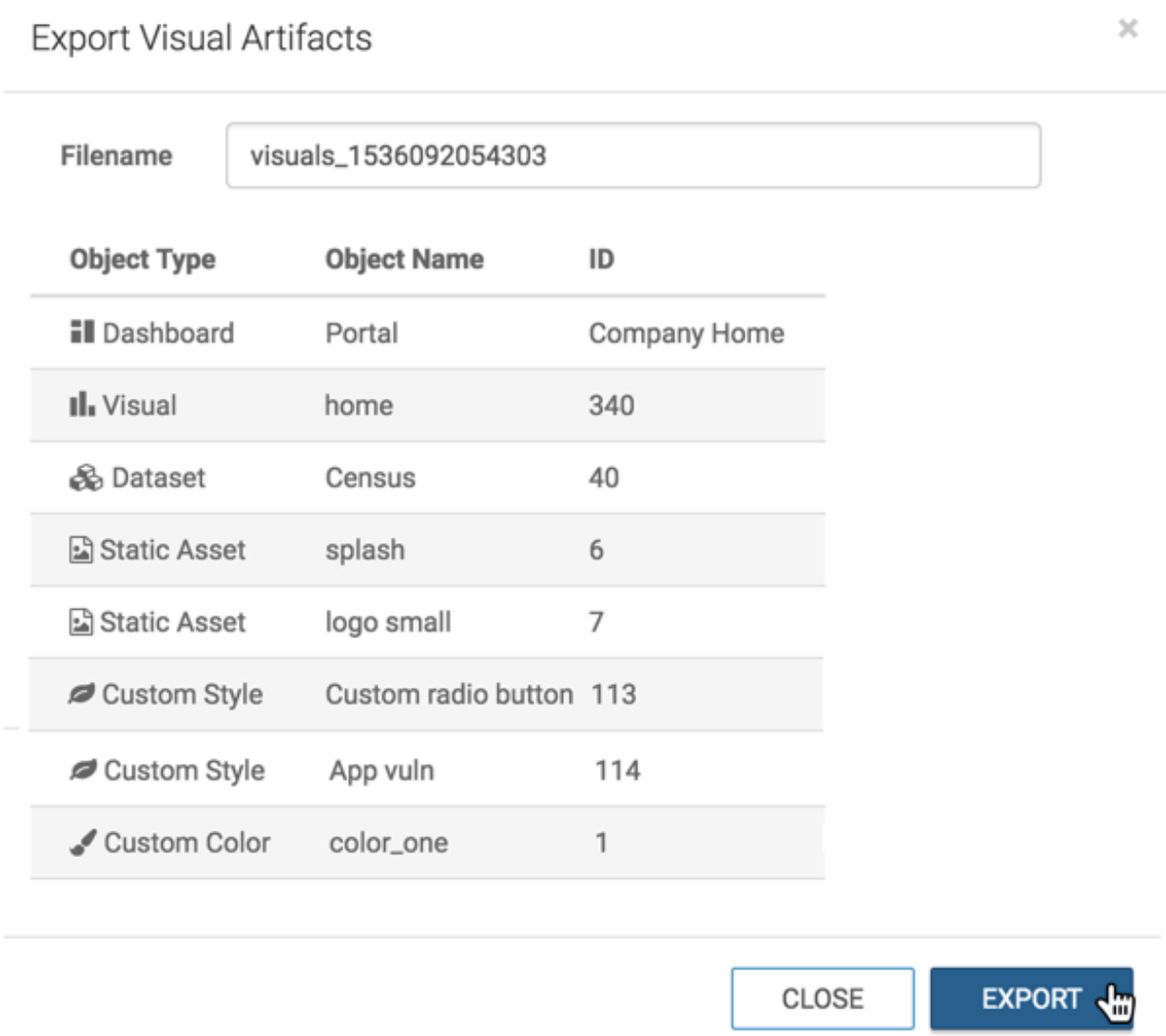

What to do next

For instructions on how to import a dashboard, see *Importing a dashboard*.

For more information about the visual artifact migration utility, see *Exporting and importing visual artifacts*.

Related Information

Importing a dashboard
Exporting and importing visual artifacts

Importing a dashboard

In Cloudera Data Visualization, you can import visuals, dashboards and other visual artifacts exported from another system. You can use the migration utility to port these artifacts from development/testing environments to production deployments or for creating back-ups. All you need is the appropriate JSON file that captures the definitions of these visual artifacts and all their dependencies (such as datasets and custom styles).

Procedure

1. Click DATA on the main navigation bar.
2. Select the connection where you want to import the artifacts.

   In this example, the Samples connection has been selected showing connecting to the same data source for both systems.
3. At the top of the screen, open the (supplemental) menu and select Import Visual Artifacts.

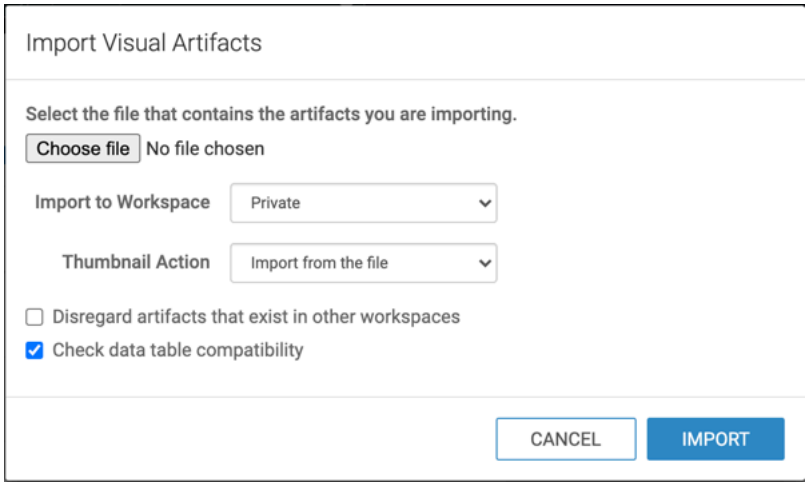

   The Import Visual Artifacts modal window appears.

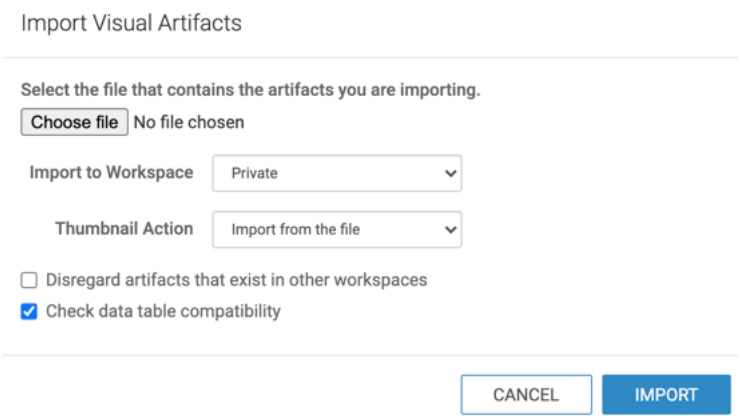

4. Click Choose File.

5. Select the file you want to import.

   In this example, visuals_only_param_population.json is selected.

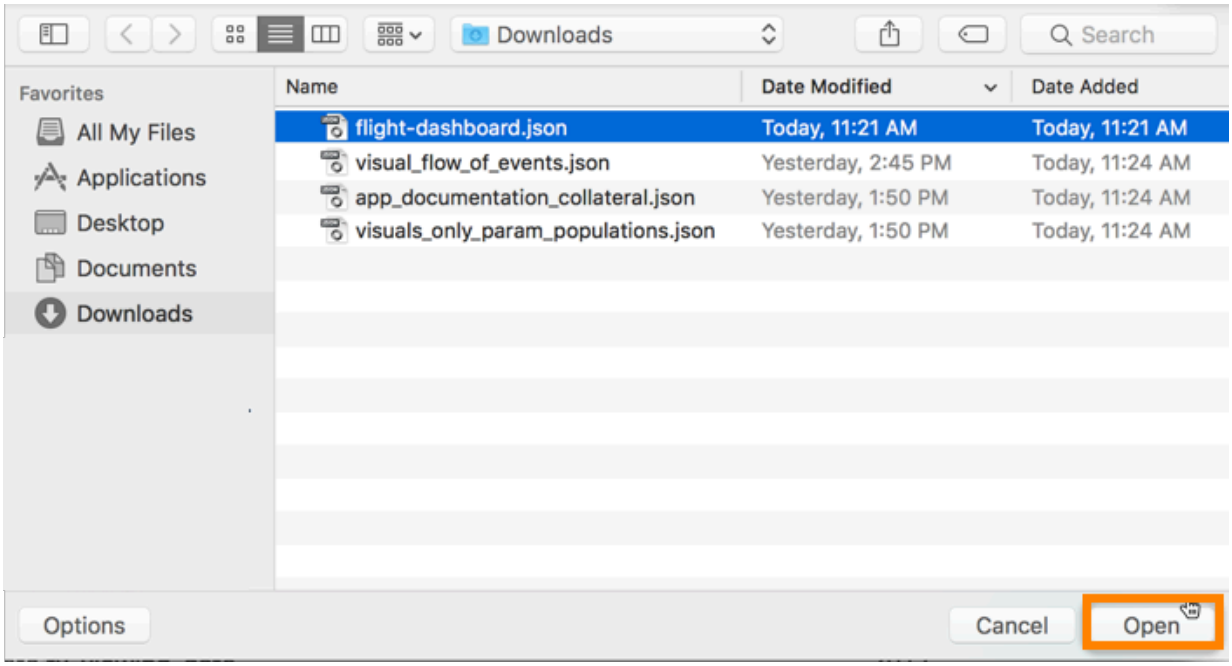

6. Specify the destination workspace in the Import to Workspace drop-down menu.

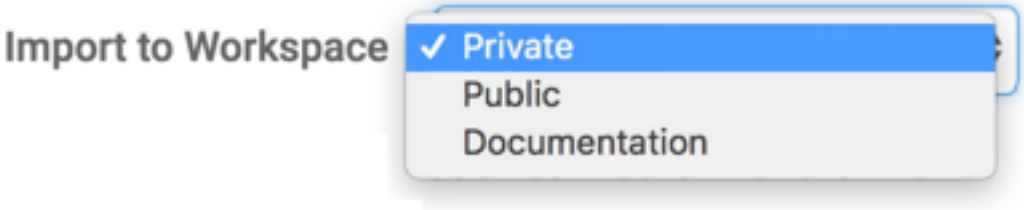

   By default, Cloudera Data Visualization imports the visual artifacts into your Private workspace. However, you can choose the Public workspace, or another workspace.

7. Specify how you want to handle thumbnails.

   There are three options:

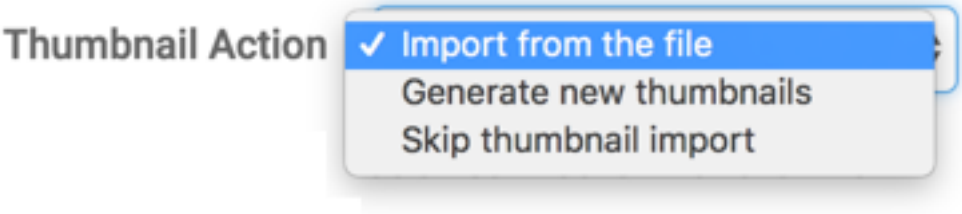

   By default, Cloudera Data Visualization imports the thumbnails from the file. However, you can choose to generate new thumbnails during the import process, or skip thumbnail import.

8. Define the Disregard artifacts that exist in other workspaces option.

   Before importing an artifact, Cloudera Data Visualization searches for it on the system. If it already exists, it is either updated (if changed from a previous import), or left 'as is'. If it does not exist, it is imported as a new visual artifact. By default, the search domain is the entire system. By selecting this option, you are limiting the search to the specified workspace.

9. Check data table compatibility.

   This is to confirm that tables suitable for the incoming dataset(s) exist on the system. It means that the database and tables have the same names on the source and target systems. This option is 'on' by default.

10. Click IMPORT.
After processing the import JSON file, Cloudera Data Visualization opens the Import interface to show its visual artifacts. It displays the visual artifacts that you saved earlier into a JSON file during export. The Planned Action column shows that this is a new import for all artifacts. At this point, none of them have an assigned ID.

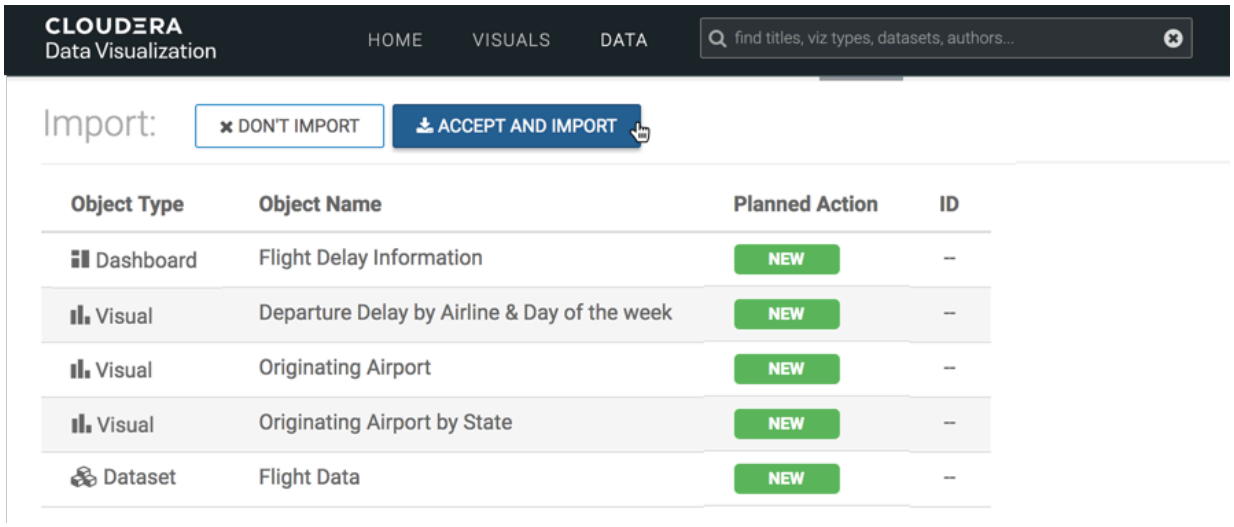

11. Click ACCEPT AND IMPORT.

Results

After the import completes, a success message appears on the Import interface. All the artifacts have an assigned ID, which are generated by the system, sequentially. Visuals/Dashboards and datasets have separate ID queues.

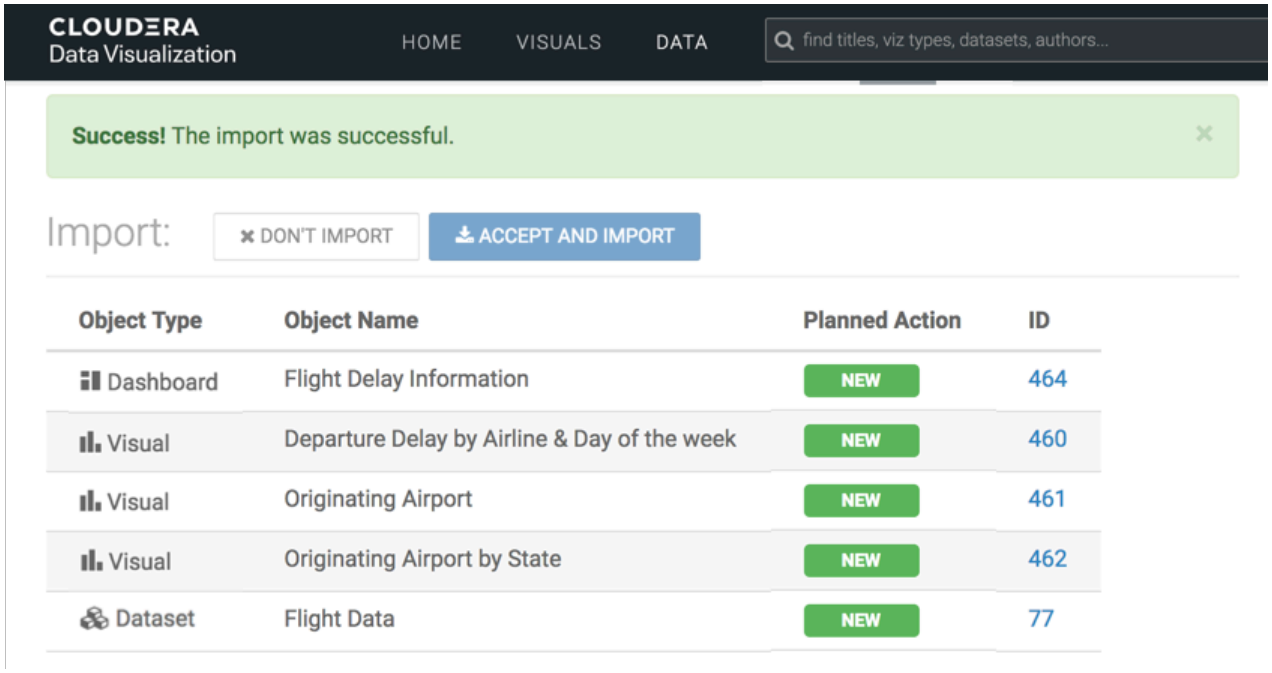

For instructions on how to export a dashboard, see *Exporting a dashboard*.

For more information about the visual artifact migration utility, see *Exporting and importing visual artifacts*.

Related Information

Exporting a dashboard
Exporting and importing visual artifacts

Moving a dashboard

If you have a dashboard in a development workspace (it can be your private or a custom workspace), you can move it to a public workspace so that everyone across the organization can access the analytical insights that the dashboard provides. Depending on your needs and your Cloudera Data Visualization installation, you can also designate the public workspace as the collection of customer-facing dashboards.

Cloudera Data Visualization utilizes Role-Based Access Control (RBAC) permissions to govern access to functionalities in the system. Administrators or users with Additional system privilege can move dashboards and apps to any workspace, including private ones.

When the dashboard you are moving contains a linked visual that is used by another dashboard in the workspace, you have two options: to clone it or to move it. For more information on what happens in each case, see the following diagram. In this example, dashboard 2 is moved from WORKSPACE I (private) to WORKSPACE II (public). Visual c is a linked visual, used by both dashboard 2 and dashboard 3.

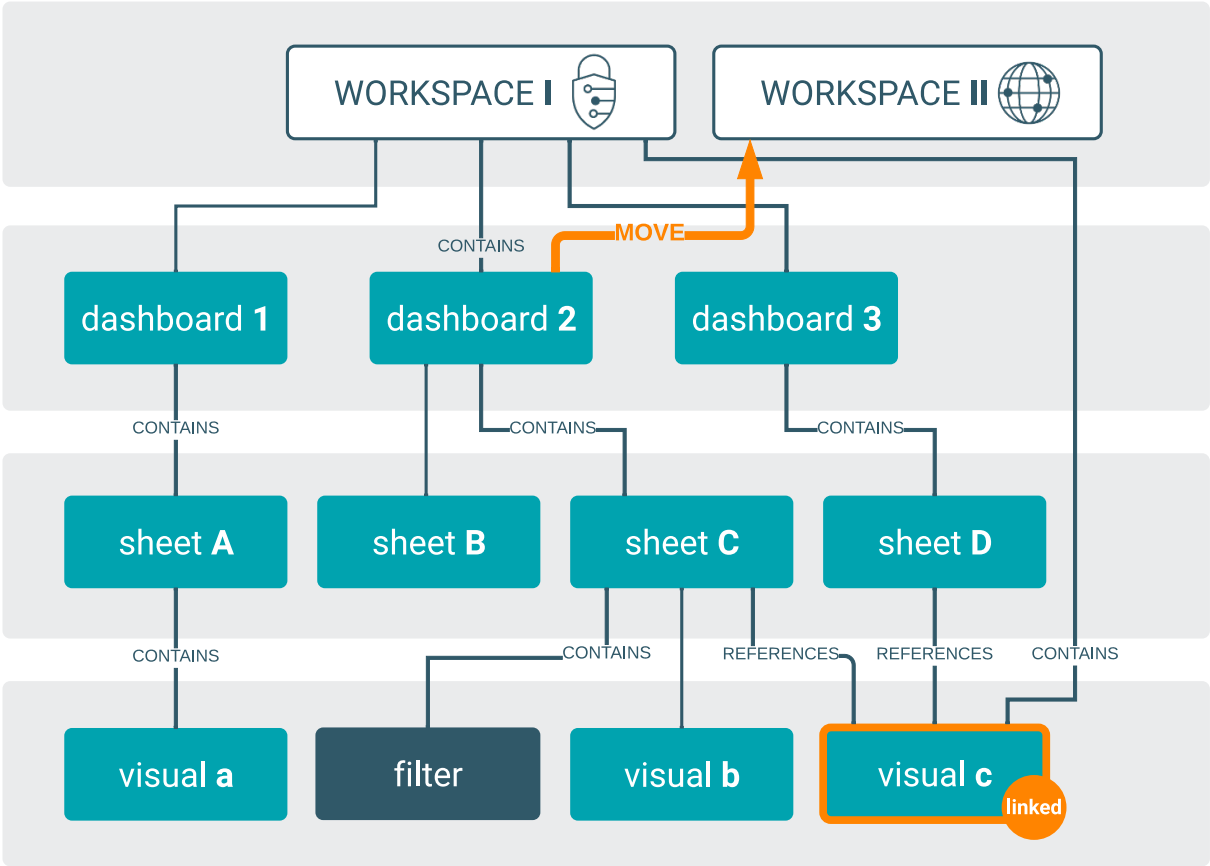

Cloning linked visuals

Cloning linked visuals allows you to create a duplicate of a linked visual. This creates a new linked visual on a specific dashbboard in a designated workspac without changing the original dashboard.

Visual c is a linked visual artifact on dashboard 3 of WORKSPACE 1. If you clone visual c, you create a new visual (visual d) in WORKSPACE II. The new visual d is also linked. Dashboard 3 in WORKSPACE I remains unchanged: visual c is still there, and it is still a linked visual.

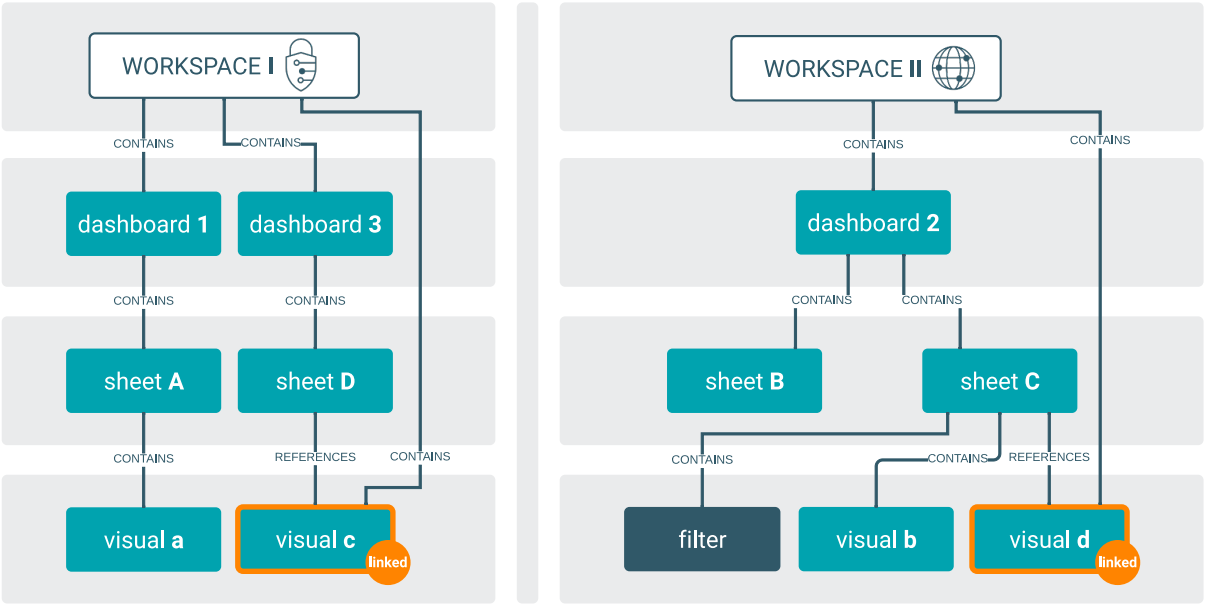

Moving linked visuals

Relocating a linked visual between workspaces provides you with flexibility in organizing, managing and sharing your visual elements.

If you move the linked visual c betweeen the workspaces, it remains a linked visual in its new location in WORKSPACE II. However, this has an impact on dashboard 3 in WORKSPACE I. While visual c is still part of the dashboard, it is referenced in an external workspace. Because visual c is no longer part of the same workspace, users who do not have sufficient permissions for WORKSPACE II will not be able to see the visual from WORKSPACE I. This is more likely to happen when moving dashboards from the public workspace into a custom workspace.

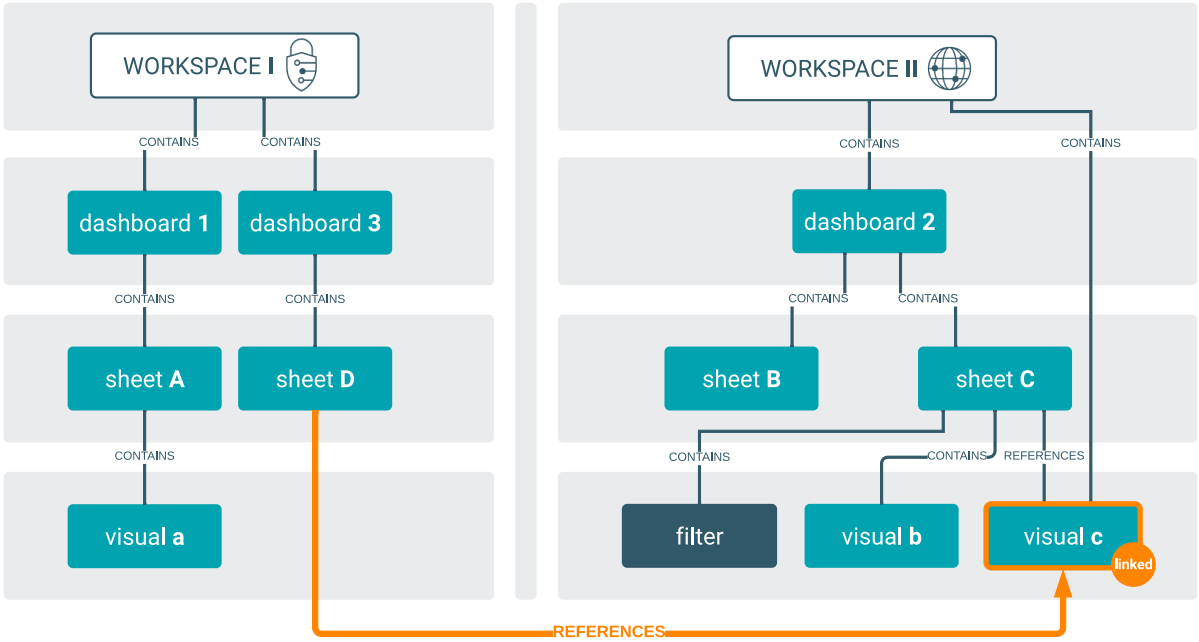

Moving dashboards between workspaces

You can move dashboards between the private, public, and custom workspaces in any direction. Whether you are transitioning dashboards for personal use, sharing them with a broader audience, or customizing your workspace organization, Cloudera Data Visualization enables you to manage your dashboards easily across different workspaces.

About this task

As a dashboard owner, you can easily designate a private dashboard as public by shifting it from your private workspace to a public workspace. Moreover, you have the option to move dashboards in various directions, leveraging private or custom workspaces.

This dynamic capability allows you to create and test dashboards in your private workspace and, when ready, share them with others by moving them to a public workspace. By doing so, individuals with at least viewing privileges for the relevant datasets gain access to the dashboard.

Note: Cloudera Data Visualization utilizes Role-Based Access Control (RBAC) permissions to govern access to functionalities in the system. Administrators or users with Additional system privilege can move dashboards and apps to any workspace, including private ones.

On the VISUALS interface

If you do no want to open the dashboard, you can move it to a different workspace on the VISUALS interface.

1. Hover over the dashboard tile or row and use the Move to Workspace function from the Supplemental menu of the tile. You can also use Move to Workspace from the actions available in the top right corner of the preview area of the interface.

   Note: Note The action buttons are only active if you select at least one artifact in the preview area.

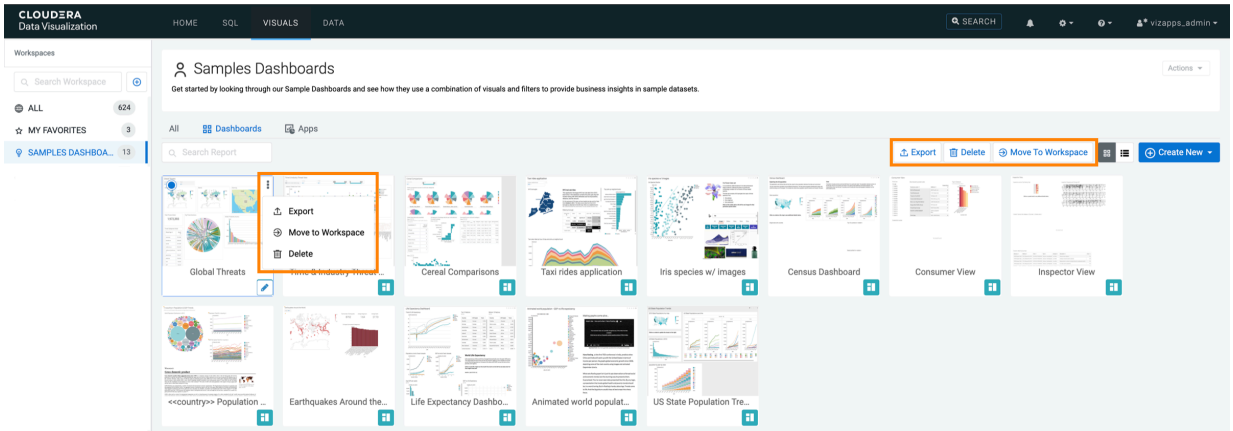

   The Move to Workspace modal window is displayed. It provides information about the dashboard you want to move.

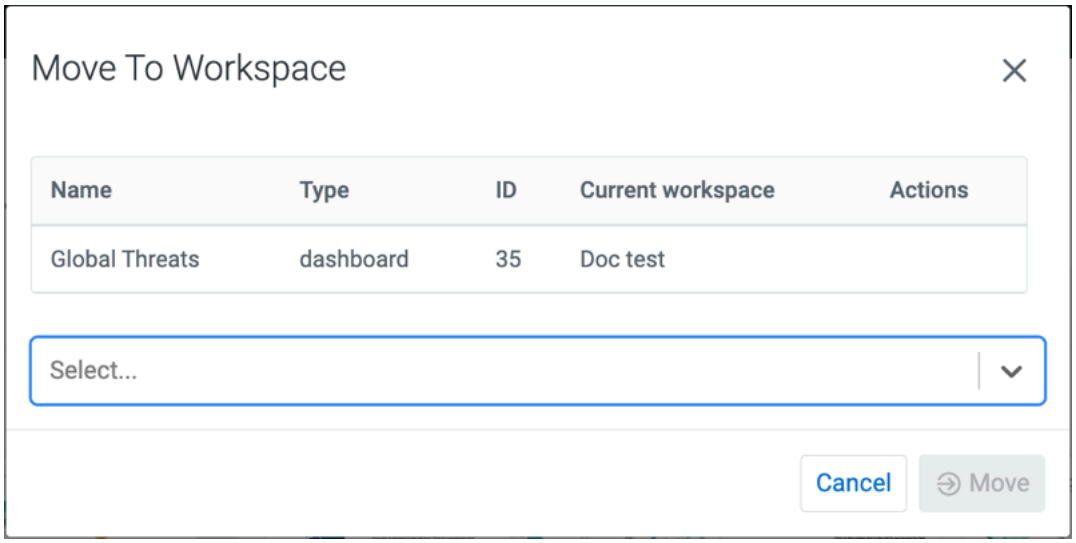

2. Selet the new workspace from the drop-down list and click Move.

After opening the dashboard

Procedure

1. Find the dashboard you want to move and open it in Edit or View mode.
2. Click the workspace indicator in the top menu of the dashboard.
3. From the drop-down menu, choose the workspace where you want to move the dashboard.

   In this example, the dashboard is in the Private workspace. Select Public.

4. After selecting the new workspace, click MOVE.

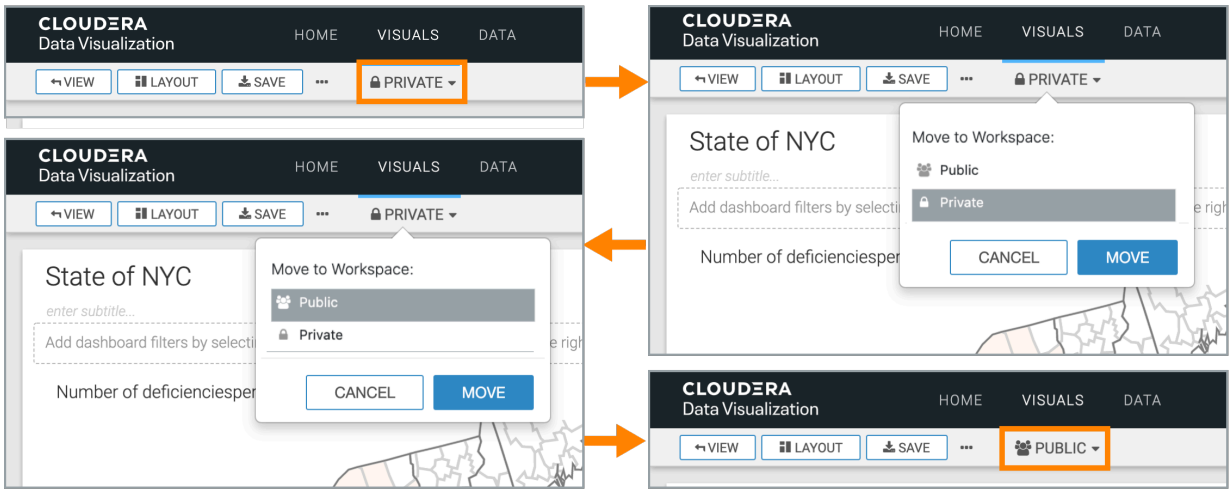

The dashboard is now in the Public workspace. You can verify it by navigating to the Visuals interface, and then selecting and expanding the destination workspace.

This process is flexible and works for moving dashboards between private, public, and custom workspaces, in any direction. Just ensure you have the necessary privileges to perform the action.

Important: Once a dashboard is added to an app, it cannot be reverted to Private.

Refreshing visuals manually in dashboards

Cloudera Data Visualization enables you to manually refresh visuals in dashboards.

About this task

If you have a slow back-end data connection and your dashboard is loading slowly, you can choose to load the filters first, specify their settings, and then manually load the visuals.

Procedure

1. Open the dashboard in Edit mode.

   In this example, the World Life Expectancy dashboard is used.
2. In the Dashboard Designer, select the Settings menu.
3. Click the General menu to expand it.
4. Select the Refresh visuals manually option.

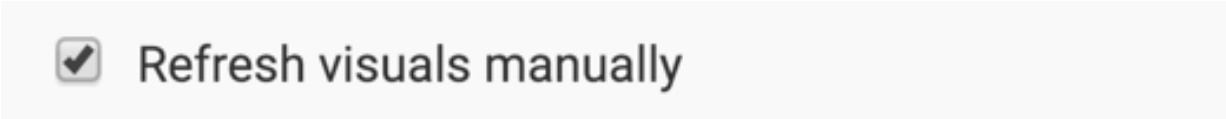

   This adds a REFRESH VISUALS button to the dashboard.
5. Under Filters, click the year field to add a new filter widget to the dashboard and select 1900 from the dropdown.
6. Click SAVE.
7. Switch to View mode.

   Only the filter widget appears on the dashboard without any visuals.

8. To view visuals, click REFRESH VISUALS in the top right corner of the dashboard.

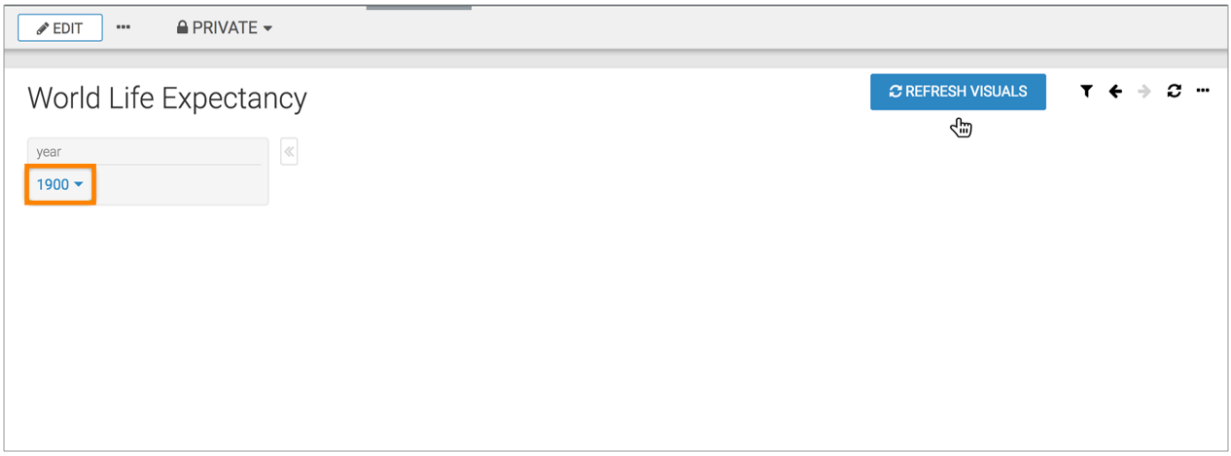

The visual now appears in the dashboard along with the defined filter.

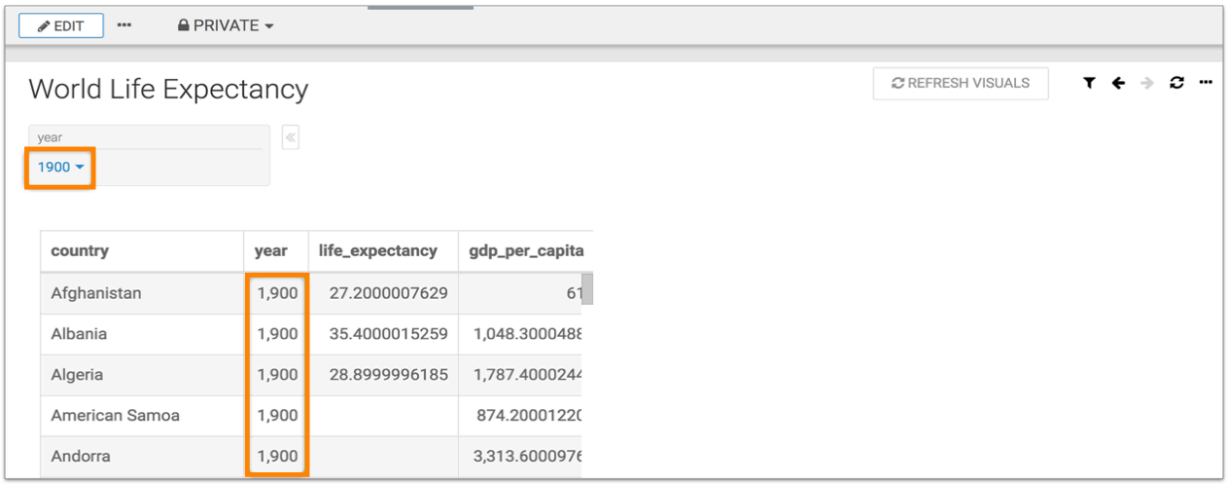

9. Add the year 1901 to the filter.

As soon as you change the filter, the filter refreshes to show both the values 1900, 1901, but the visual does not refresh. The year column continues to show only 1900. Also, the REFRESH VISUALS button again appears in the top right corner of the dashboard.

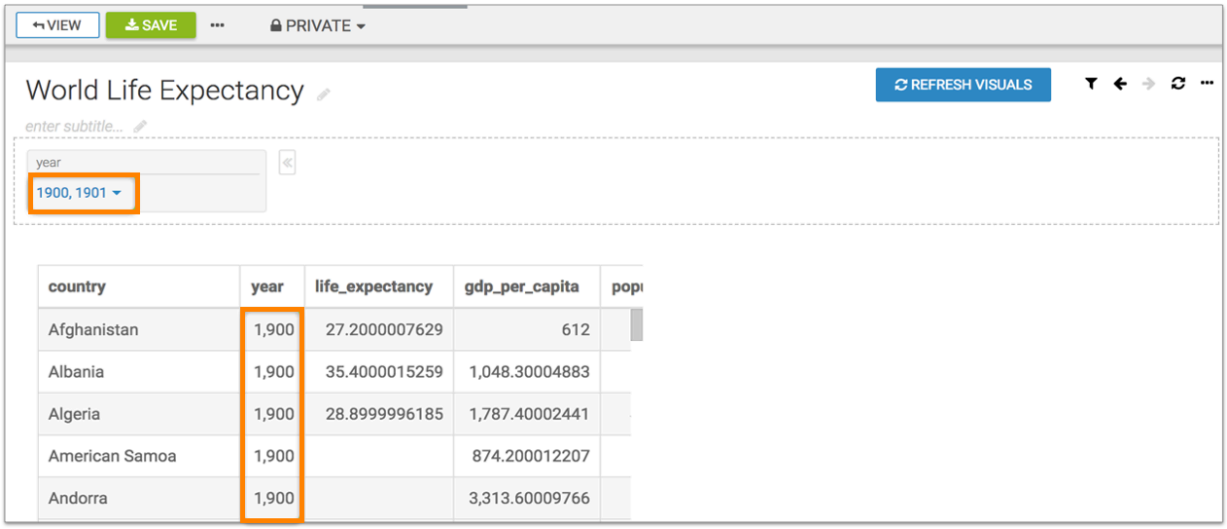

10. To render the updated visual, click REFRESH VISUALS.

The year column now shows the updated values, 1900 and 1901.

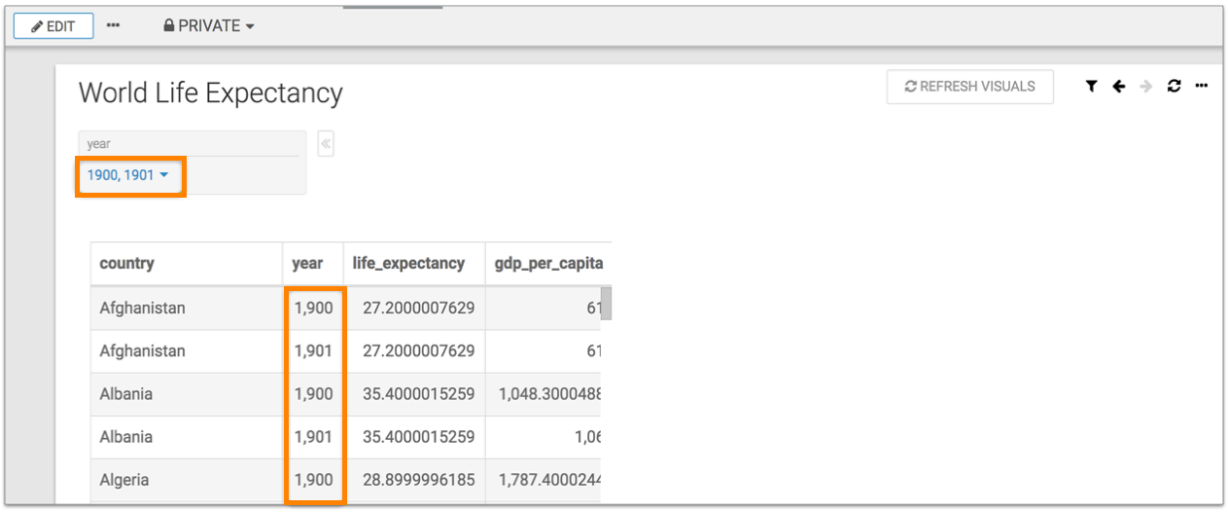

11. Click SAVE.

Deleting a dashboard

You can easily delete dashboards that you no longer use in Cloudera Data Visualization.

Procedure

1. Find the dashboard you want to delete and open it either in Edit or View mode.

2. Click the (supplemental) menu in the top menu bar and select Delete Dashboard.

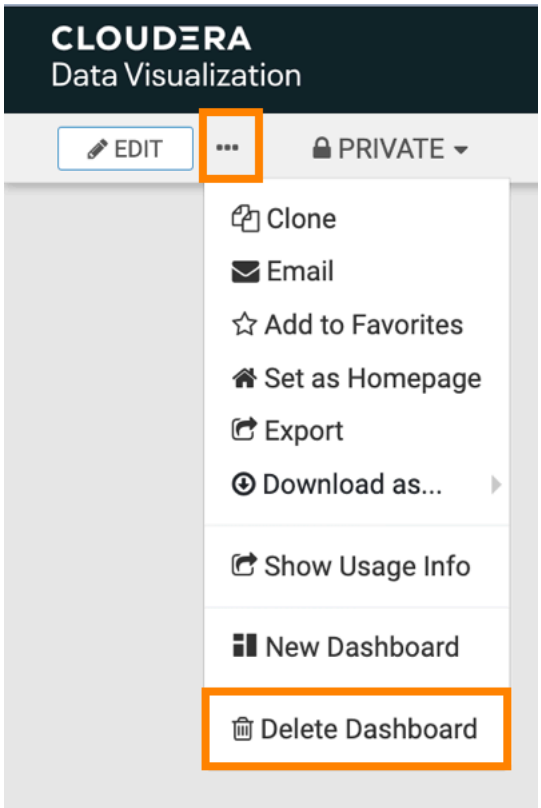

Alternatively, if you do no want to open the dashboard, you can delete on the VISUALS interface as well. Just hover over the dashboard tile or row and click Delete from its Supplemental menu. You can also use Delete from the actions available in the top right corner of the preview area of the interface.

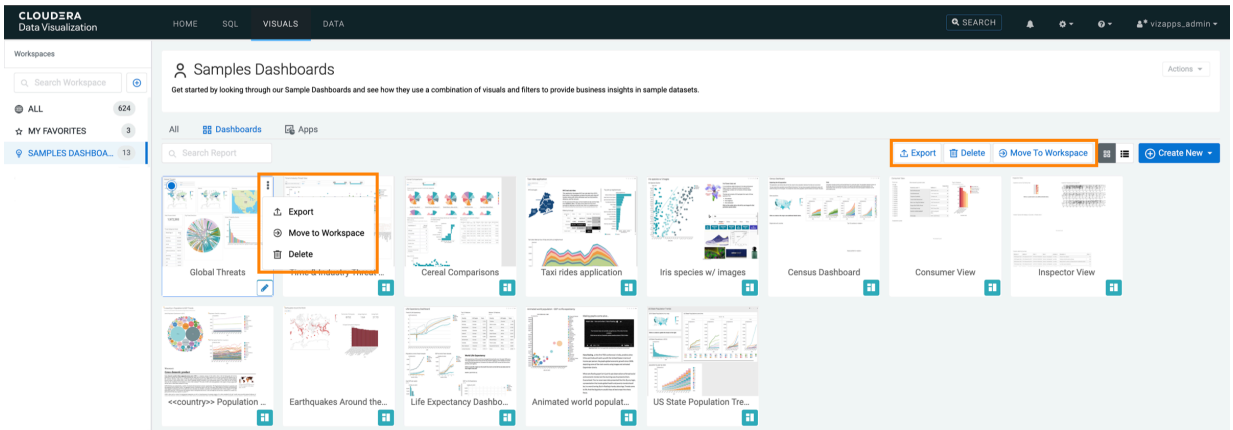

The Delete Visual Artifacts modal window appears providing information about the dashboard you are about to delete.

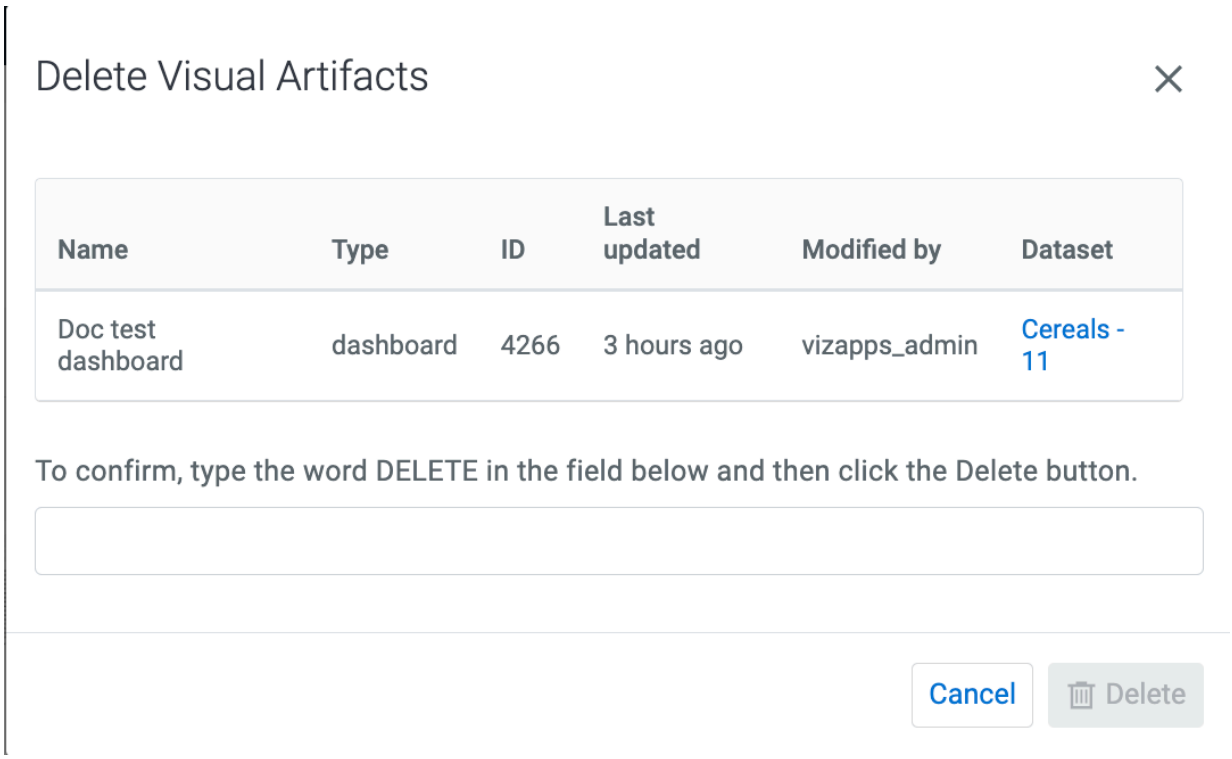

3. Type DELETE to confirm deleting the dashboard and click the Delete button.

Cloudera Data Visualization 7.2.9

Working with Data APIs

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enabling Data APIs **4**

Using API keys **4**

Accessing Data API request payload **5**

Migrating visual artifacts with REST APIs **6**

Exporting visual artifacts with REST APIs 7

Importing visual artifacts with REST APIs 7

Enabling Data APIs

To enable Cloudera Data Visualization Data API, update the configuration settings and restart the Cloudera Data Visualization instance.

About this task

Note: This is only possible for Cloudera Data Visualization instances running in Cloudera AI and Cloudera Data Science Workbench.

Before you begin

Before invoking the platform-name Data API, you must first enable it in the configuration file of the Web Server and obtain an API key.

If you want to access the API outside of Cloudera AI or Cloudera Data Science Workbench jobs or sessions, you have to enable Enable Unauthenticated Access for the application. For more information and instructions, see the Public Applications section in Securing Applications.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu and select Site Settings.
2. Select Advanced Settings from the left navigation and scroll down to the Advanced Site Settings section.
3. Add the ENABLE_API_KEYS and DATA_API_ENABLED settings.

```
ENABLE_API_KEYS = True  # If you plan to access via API keys
DATA_API_ENABLED = True
```

4. Save your changes and restart the Cloudera Data Visualization service.

Using API keys

When you do not log into the visual layer of Cloudera Data Visualization directly, you must obtain an API key and implement it when using Data APIs.

Before you begin

Create a new API key. For instructions, see *Creating new API keys*.

Procedure

1. Use the API key that you have obtained to authenticate the user to access the dataset.
2. Invoke the data layer of Cloudera Data Visualization at the following endpoint and proceed with using the Data API interface.

```
http(s)://server:port/arc/api/data
```

For more information, see *Example of Data API usage*.

Related Information

Creating new API keys

Example of Data API usage

Accessing Data API request payload

To avoid creating Data API requests from scratch, Cloudera Data Visualization provides access to the details of the data requests.

About this task

This task is optional.

Procedure

1. Open Site Settings and scroll to Advanced Settings at the bottom of the left navigation.
2. Add the following settings.

```
ENABLE_DSREQ_PERF_DISPLAY = True
COMPRESS_DSREQ = True
```

3. Restart the Cloudera Data Visualization service.
4. Open an existing dashboard.
5. Use the keyboard shortcut Shift + Control/Cmd + G to open the Performance Profile interface.

6. Hover over a line that represents a visual ID and view the duration, type, and details of the query that runs against that visual.

   The query uses the same API and includes the details of the data requests. You can use this dataset request as the payload for your Data API call.

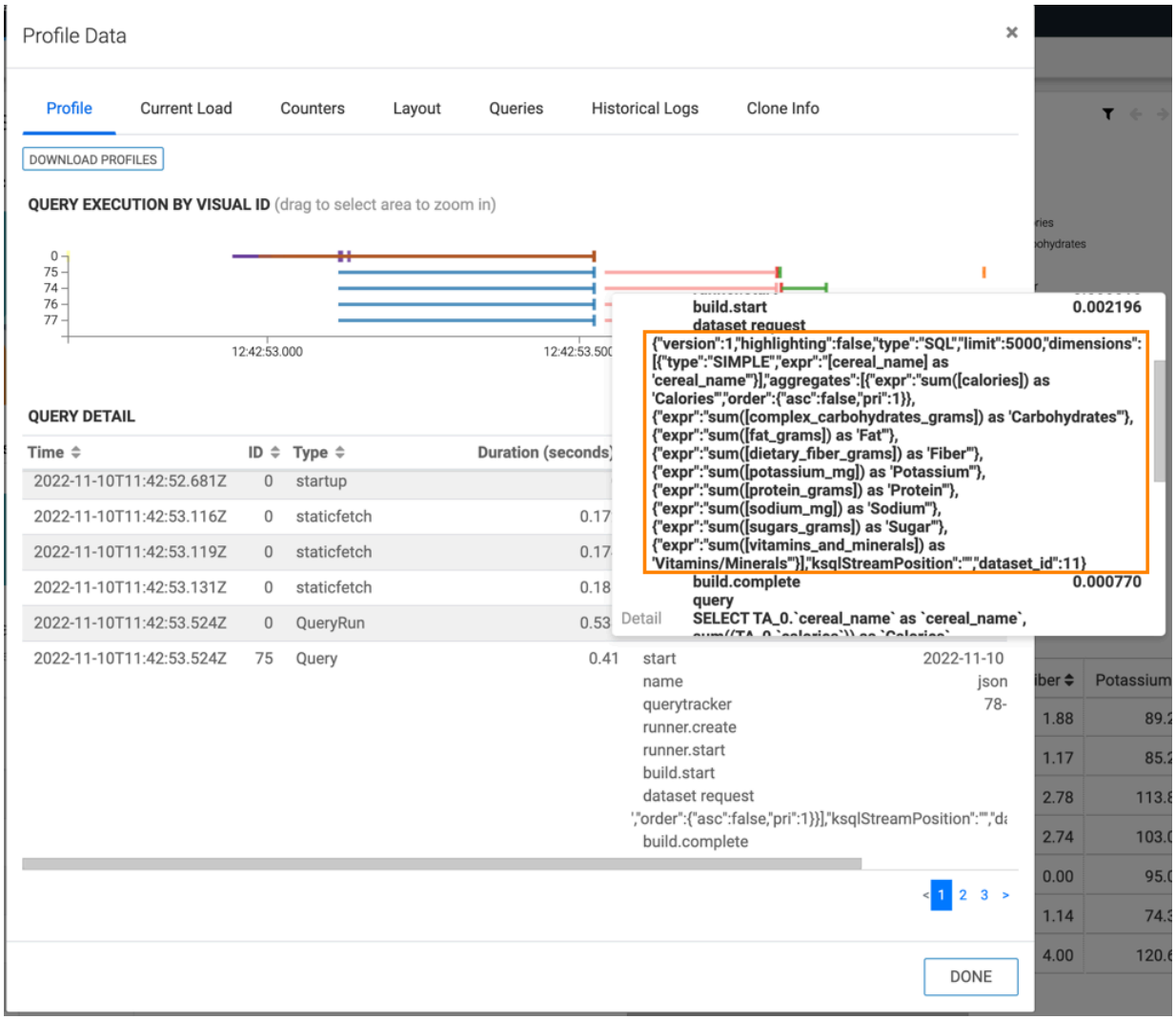

Migrating visual artifacts with REST APIs

In addition to UI-enabled export and import, Cloudera Data Visualization provides REST APIs that give more flexibility in choosing what objects to migrate. While migrating artifacts using the UI allows you to directly specify export of visuals, dashboards, and apps, REST APIs enable you to specify a wider range of migratable objects.

About this task

Note: You do not migrate events and segments separately, but as part of dataset migration. Similarly, you do not migrate thumbnails along with dashboards or visuals that they represent.

Before you begin

Enable and create the relevant data API keys, both for the source and destination machines. For instructions, see *Enabling API keys* and *Creating new API keys*. The examples used in these instructions use the secret API key that appears when you generate the keys for the first time, as in *Creating new API keys*.

Procedure

1. Export the objects by submitting a GET request on the source machine. For instructions, see *Exporting visual artifacts with REST APIs*.
2. Import the objects by submitting a POST request on the target machine. For instructions, see *Importing visual artifacts with REST APIs*.

Related Information

Enabling API keys

Creating new API keys

Exporting visual artifacts with REST APIs

Procedure

1. Authenticate the user using the data API.
2. Submit a GET request to the following address for a source machine: *SOURCE_ARCVIZ_IP*/arc/migration/api/export/

   For instructions on import, see *Importing visual artifacts with REST APIs*.

Example

This is a sample Python code snippet for authenticating and submitting the GET request that exports dashboards with IDs 2115 and 2110, and then writing the response to the my_app.json file.

```
export requests
headers = {'AUTHORIZATION':'apikey secret_api_key'}
payload = {'dashboards': '[2115, 2110]', 'filename': 'apitestmigration',
'dry_run':'False'}
r = requests.get('source_arcviz_ip/arc/migration/api/export/', headers=hea
ders, params=payload)
with open('my_app.json', 'w') as f:
        f.write(r.text)
```

Related Information

Importing visual artifacts with REST APIs

Importing visual artifacts with REST APIs

Before you begin

Make sure that the JSON file with information about exported visual artifacts is on the destination machine.

Procedure

1. Authenticate the user using the data API.

2. Submit a POST request to the following address for a destination machine: *DESTINATION_ARCVIZ_IP*/arc/migration/api/import/

   For instructions on export, see *Exporting visual artifacts with REST APIs*.

Example

This is a sample Python code snippet for authenticating and submitting the POST request that imports dashboards with IDs 2115 and 2110, which were earlier saved (from a metadata perspective) as my_app.json. This file is uploaded to the Cloudera Data Visualization server during the import.

```
import requests
headers = {'AUTHORIZATION':'apikey secret_api_key'}
payload = {'dry_run': False, "dataconnection_name":"data_connection"}
files = {'import_file': open('/Users/my_name/Downloads/my_app.json','r')}
r = requests.post('destination_arcviz_ip/arc/migration/api/import/',files=fi
les, data=payload, headers=headers)
print r.status_code # 200 is success
```

Related Information

Exporting visual artifacts with REST APIs

Cloudera Data Visualization 7.2.9

Emails and Email Templates

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating new email templates 4

Changing email templates 8

Cloning email templates 9

Deleting email templates 11

Creating new email templates

About this task

To define a new email template, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Email Templates.

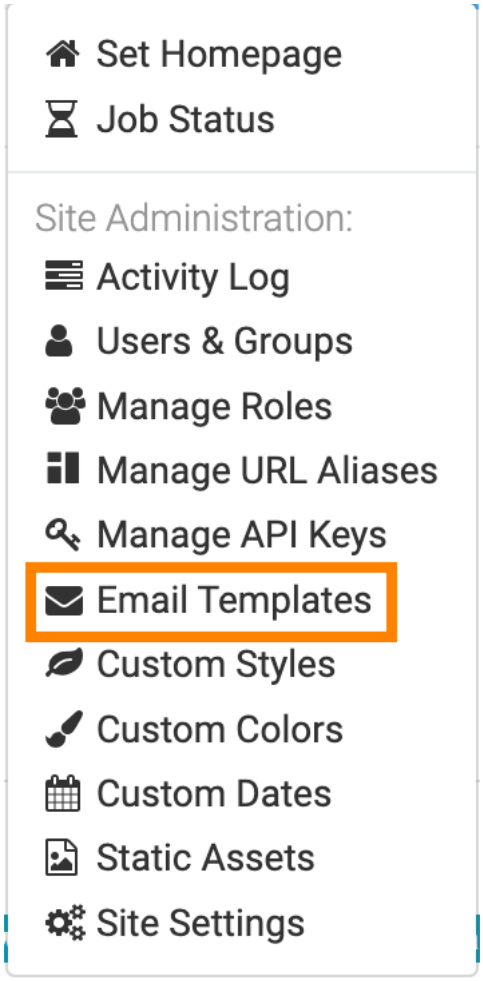

3. In the Manage Email Templates interface, click NEW EMAIL TEMPLATE.

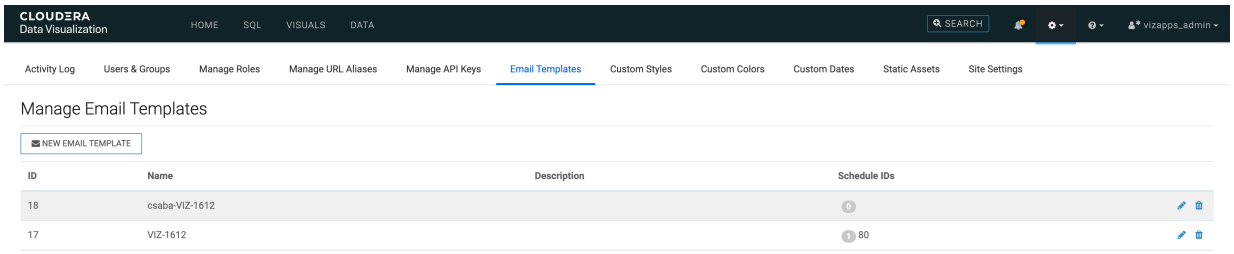

Before creating a new template, it is recommended that you explore the available sample templates. Click SAMPLE TEMPLATE 1 or SAMPLE TEMPLATE 2 to explore the available templates. You may find it easier to create your own template by starting from a sample template and making necessary adjustments.

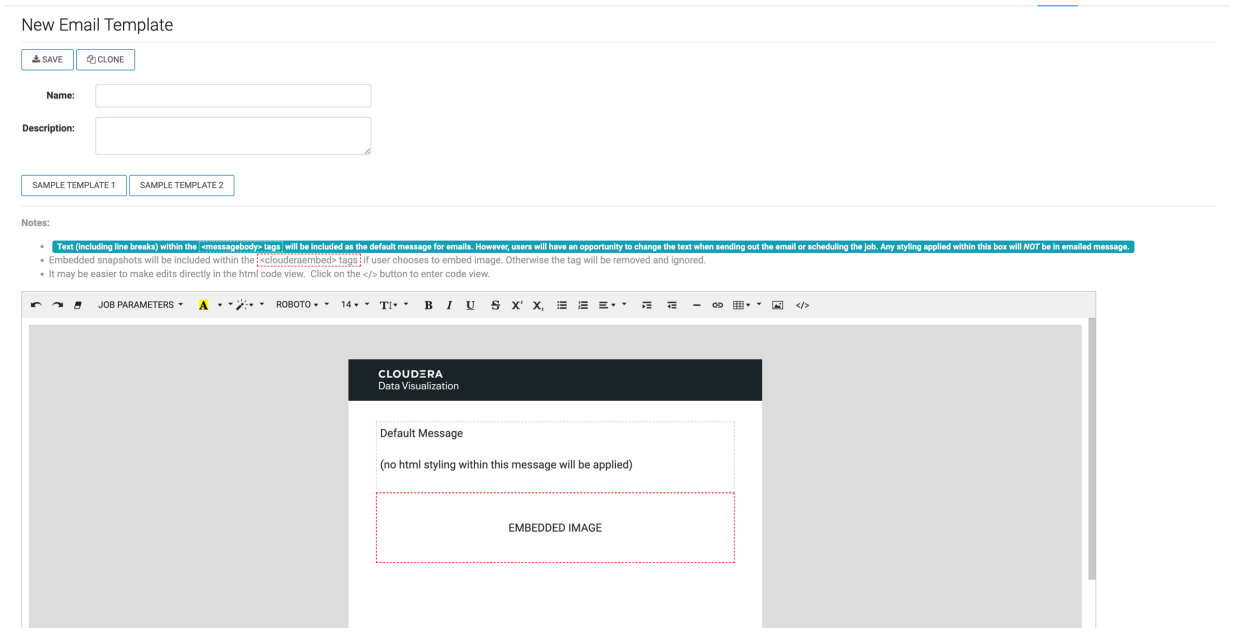

4. Under Name, enter the name of the template.
5. Under Description, enter the information that helps the user select the appropriate template.
6. Use the Rich-Text Editor to create the message template.

   In addition to a large text entry box and a set of editing options on the toolbar, the following key options are also included:

   - *Job Parameters*

     This menu provides access to some standard Job Parameters. For more infornation, see *Parameters for jobs*.
   - *Code View*

     To edit the template directly in HTML format, click </> on the toolbar. The HTML used for email formatting is somewhat different from standard HTML. For example, you can format the content as a set of nested

tables. Before working directly in HTML mode, see *Email development recommendations* for design and development best practices.

- *Logo*
  - You can change the size of the logo by clicking on it, and selecting one of the standard sizes: 100%, 50%, or 25%.
  - To delete the logo image, click on it, and select the trash icon.

    Note that the code view option has a placeholder for the logo, even when you deleted the image.
  - To insert a new logo, click on the placeholder in the top bar, and then click the Picture option.

    In the Insert Image modal window, add the image either through the Select from files option (browsing through local directory), or through the Image URL option.

    Note: The Image URL option should point to a storage location that is open to all recipients of emails generated from this template. Otherwise, the image will not load.
  - To change the background color (so the logo blends in), open Code View, and change the bgcolor parameter. In this example, the background is set to black:

```
<td align="center" valign="middle" bgcolor="#000000" style="padding:
10px;font-size:24px">
                    <!-- company logo goes here -->
                    <img src="image_location/file_name.png" alt="C
loudera Data Visualization" style="max-width: 200px; max-height: 50p
x;">
                    </td>
```

  - To change the size of the logo image, you may need to change not only the percentage width of the image, but also the max-width and max-height parameters in Code View. For example, The Sample Template uses the following style settings:

```
<td align="center" valign="middle" bgcolor="#000000" style="padding:
10px;font-size:24px">
                    <!-- company logo goes here -->
                    <img src="image_location/file_name.png" alt="C
loudera Data Visualization" style="max-width: 200px; max-height:
 50px;">
                    </td>
```

    To increase the size of the image, increase these parameter values and add the width parameter.

```
<td align="center" valign="middle" bgcolor="#000000" style="padding:
10px;font-size:24px">
                      <!-- company logo goes here -->
                      <img src="image_location/file_name.png" alt="C
loudera Data Visualization"
                      style="max-width: 300px; max-height: 80px;
 width: 150%;">
                      </td>
```

  - To change the alignment of the logo, click the Paragraph icon and select another alignment option. Alternatively, add the text-align parameter in Code View.

```
<td align="center" valign="middle" bgcolor="#000000" style="text-
align:left;padding:10px;font-size:24px">
                    <!-- company logo goes here -->
                    <img src="image_location/file_name.png" alt="C
loudera Data Visualization" style="max-width: 200px; max-height: 50p
x;">
                    </td>
```

- To reduce or increase the height of the top banner that contains the logo, change the padding parameter in Code View.

```
<td align="center" valign="middle" bgcolor="#000000" style="text-ali
gn:left;padding:2px;font-size:24px">
                      <!-- company logo goes here -->
                      <img src="image_location/file_name.png" alt="C
loudera Data Visualization" style="max-width: 200px; max-height: 50p
x;">
                      </td>
```

- *Message Body*

  The message body appears between the <messagebody> … </messagebody> tags. Use <br> tags to separate lines of text.

  You can easily embed Cloudera Data Visualization parameters in the text body.

```
<td style="padding:30px 40px;">
                    <p style="font-size: 16px; min-height: 300px; line-h
eight: 1.4;">
                    <!-- use <br> for line breaks -->
                    <messagebody>This is a general email to notify you
of the key performance metrics tracked by Dashboard
                    <b><span>&lt;&lt;app_id&gt;&gt;:</span> </b><b
>&lt;&lt;app_name&gt;&gt;</b>.<br>
                    <br>Cloudera generates this message automatically
.<br>
                    <br>Most recent run date is: <span>&lt;&lt;st
art_time&gt;&gt;</span>
                    <br>You can see the dashboard here: &lt;&lt;relative
_url&gt;&gt;</messagebody>
                    </p>
```

- *Embedded Images*

  Click inside the EMBEDDED IMAGE area of the template, and then click the picture option. Add an image and configure it in the same way as you did for the logo. Remember to contain it between <clouderaembed> … </clouderaembed> tags.

  In this exmaple, a very large image has been embedded, so the style="width: 200px;" parameter has been applied to it.

```
<clouderaembed><img src="image_location/koda.jpg alt="masco
t" style="width: 200px;"></clouderaembed>
```

- *Footer text*

  Enter and configure footer text.

  In this exmaple "Footer Text" has been replaced with "Greetings from Koda".

- *Copyright text*

  The template contains a default statement about the authorship of the email. You may choose to use it as a copyright statement.

  In this example "This email was sent by Cloudera" has been replaced with "Email © Cloudera".

7. Click SAVE to save all changes to the template.
8. Click the Email Templates tab at the top of the screen to see the new item in the list of templates.

Related Information

Parameters for jobs

Email development recommendations

Changing email templates

About this task

To change an existing email template, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Email Templates.

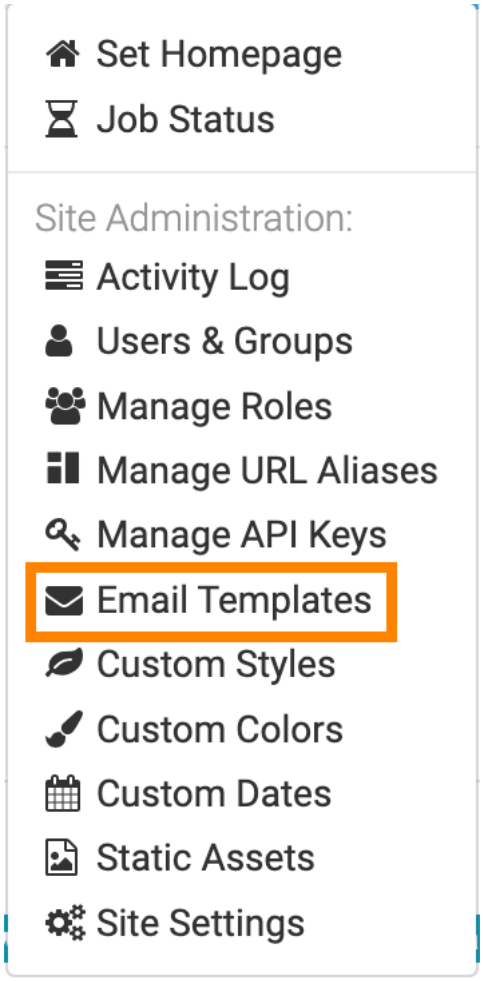

3. In the Manage Email Templates interface, click the Pencil icon to edit the template.

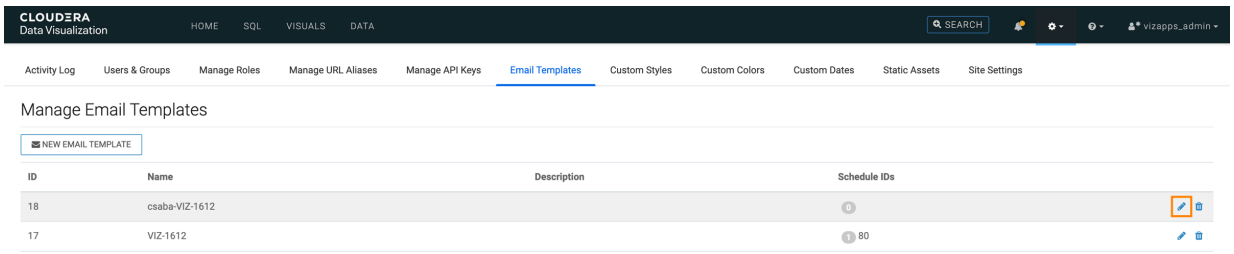

4. Make the necessary changes in the template and click SAVE.

5. Click the Email Templates tab at the top of the screen to see the updated item in the list of templates.

Cloning email templates

About this task

To make a clone from an existing email template, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Email Templates.

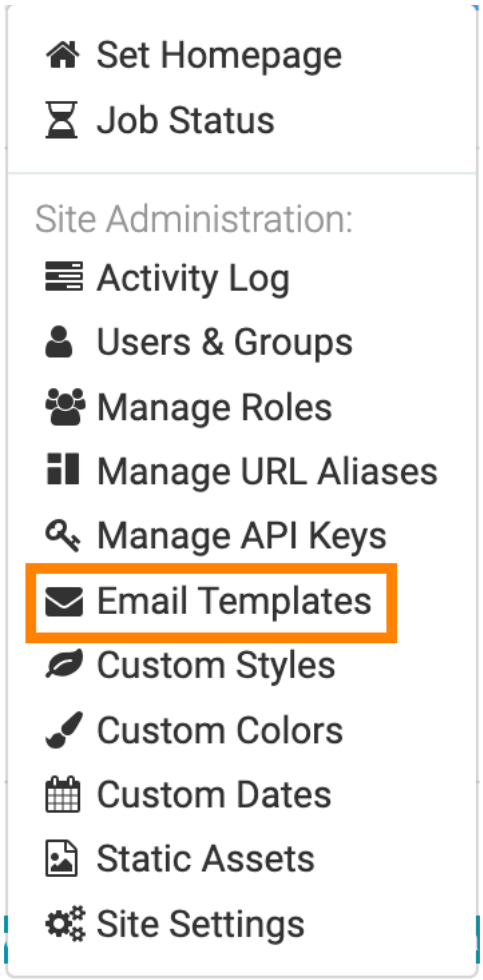

3. In the Manage Email Templates interface, click the Pencil icon to edit the template.

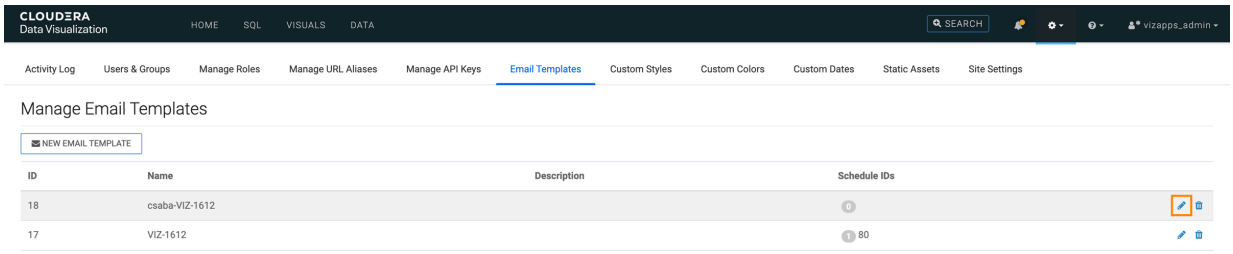

4. Click CLONE.

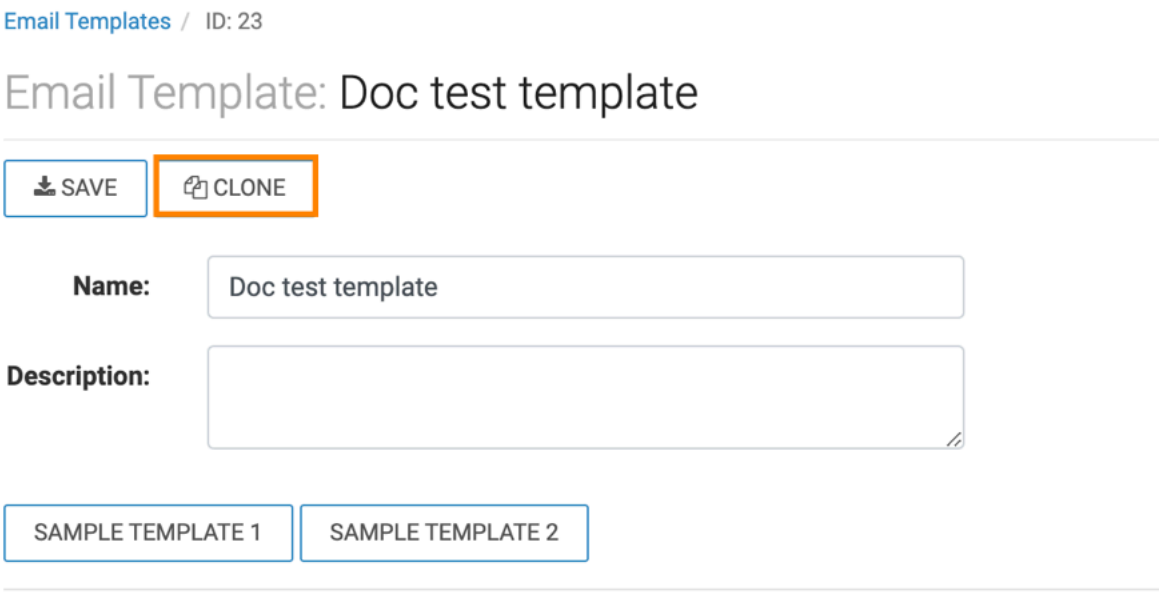

5. In the cloned template, make the necessary changes and click SAVE.

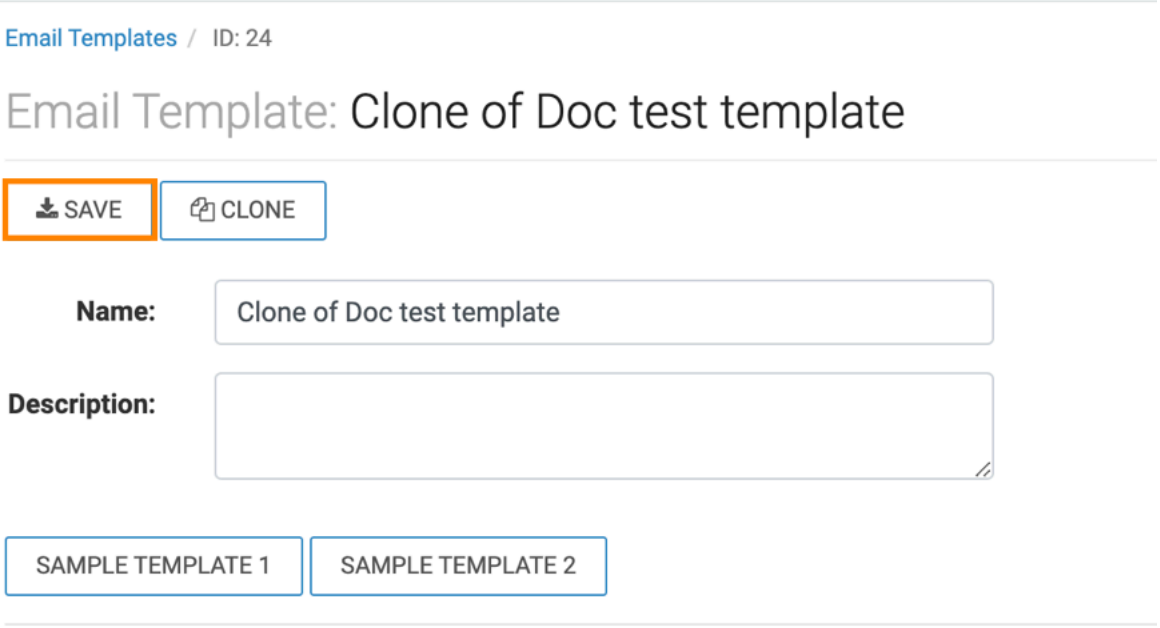

6. Click the Email Templates tab at the top of the screen to see the updated item in the list of templates.

Deleting email templates

About this task

To delete an existing email template, follow these steps.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Select Email Templates.

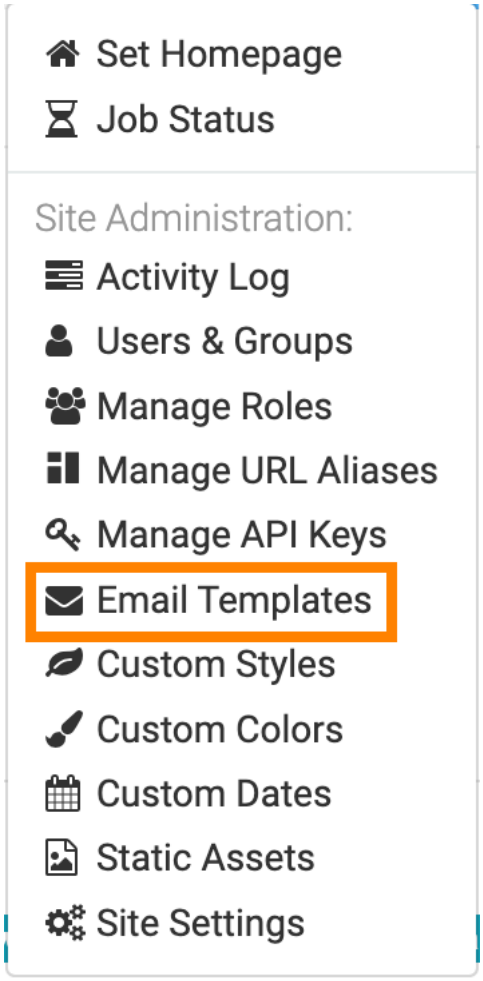

3. In the Manage Email Templates interface, click the Trashcan icon to delete the template.

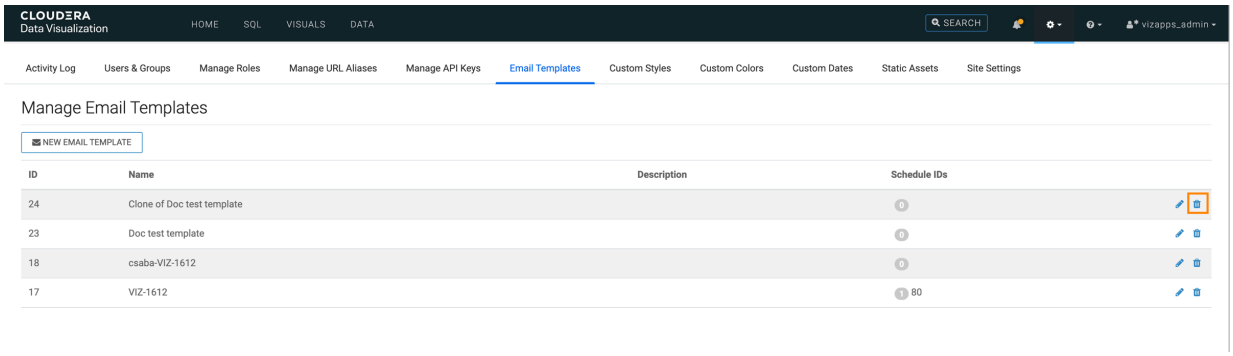

4. In the Delete Confirmation modal window, type DELETE, and then click DELETE.

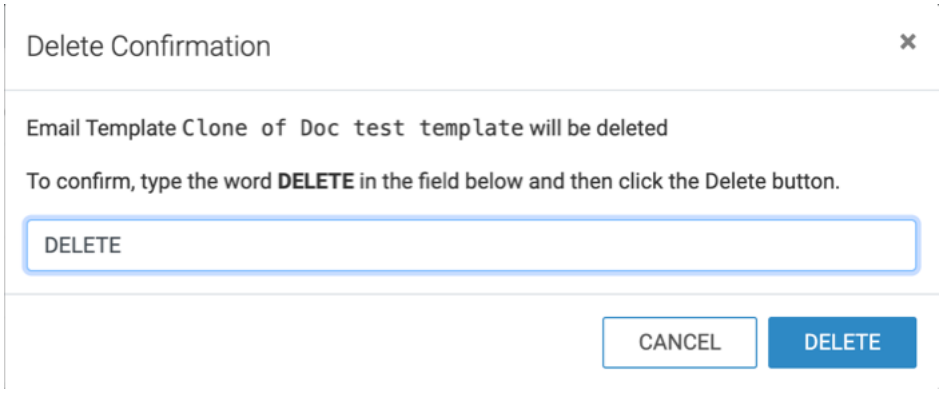

Results

A success message is displayed briefly and the list of templates no longer includes the deleted one.

Cloudera Data Visualization 7.2.9

Custom Colors

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing custom colors 4

Creating new custom colors 4

Creating custom colors with distinct colors 6

Creating custom colors with gradient values 12

Creating custom colors with gradient percentages 17

Editing custom colors 21

Deleting custom colors 25

Managing custom colors

In Cloudera Data Visualization, users with administrative privileges can define custom color palettes as another choice for styling visuals and applications.

Anyone can use a custom color in a visual immediately after and administrator creates it. Administrators can also edit and delete custom colors.

Custom color palettes may be defined as a series of distinct colors, gradients with values, and gradients with percentages.

Note: This feature is only available to users with administrative privileges.

Related Information

Creating new custom colors
Editing custom colors
Deleting custom colors
Creating custom colors with distinct colors
Creating custom colors with gradient values
Creating custom colors with gradient percentages

Creating new custom colors

About this task

To define a custom color, follow these steps.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu, and select Custom Colors.

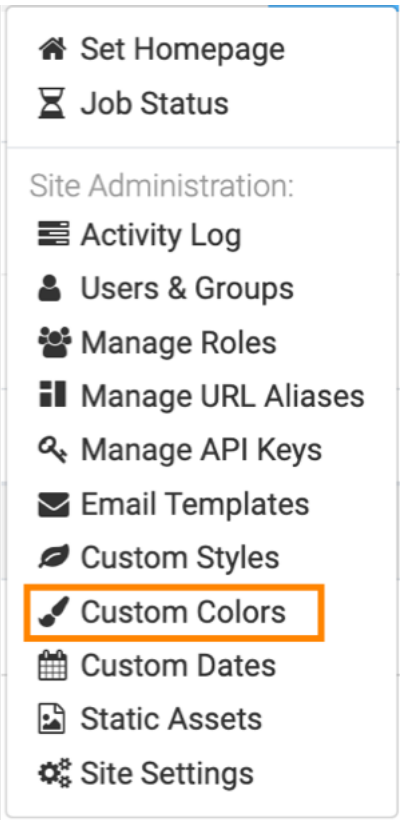

2. In the Manage Custom Colors interface, click NEW CUSTOM COLOR.
3. In the New Custom Color interface, add a name and a description for the new color.
4. Define one of the color types:
   - *Distinct colors*
   - *Gradient colors - with values*
   - *Gradient colors - with percentages*
5. Click SAVE.
6. Click the Custom Colors tabe to return to the Custom Colors interface.

Example

The custom colors appear in the Settings Palette menu, next to the default colors.

Related Information

Creating custom colors with distinct colors
Creating custom colors with gradient values
Creating custom colors with gradient percentages

Creating custom colors with distinct colors

In Cloudera Data Visualization, users with administrative role can define distinct color custom color palettes.

About this task

Perform these actions in the New Custom Color interface:

Procedure

1. Enter the Name of the new color.
   In this example, Leaf is used.
2. You can add a description for the new color.
3. In Color Scale Type, choose Distinct.
4. For Color 1, enter a hexadecimal color code.
   In this example, #25351f is used.

5. Click ADD MORE.

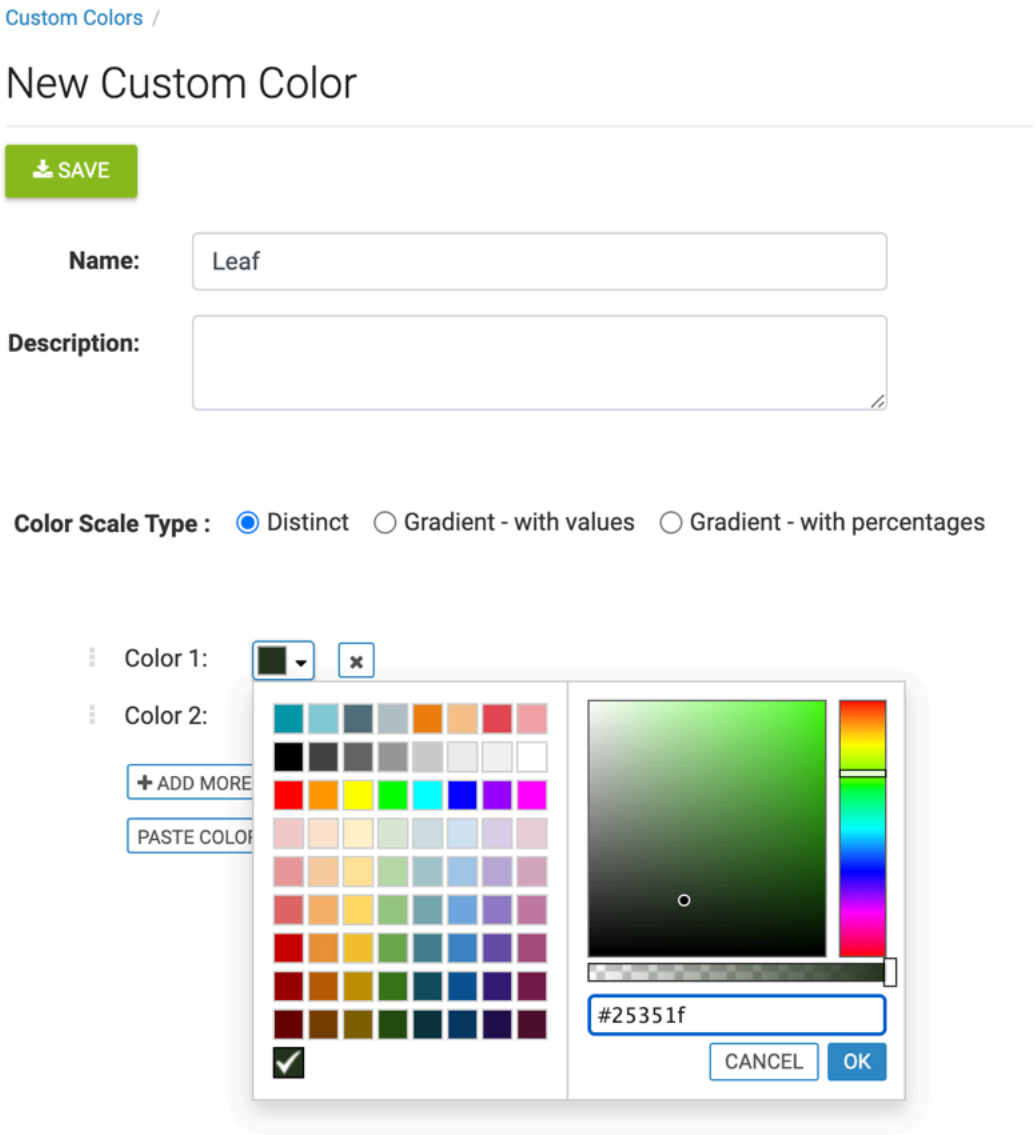

6. For Color 2, use the color picker.

   In this example, the color #76ce1b has been picked.

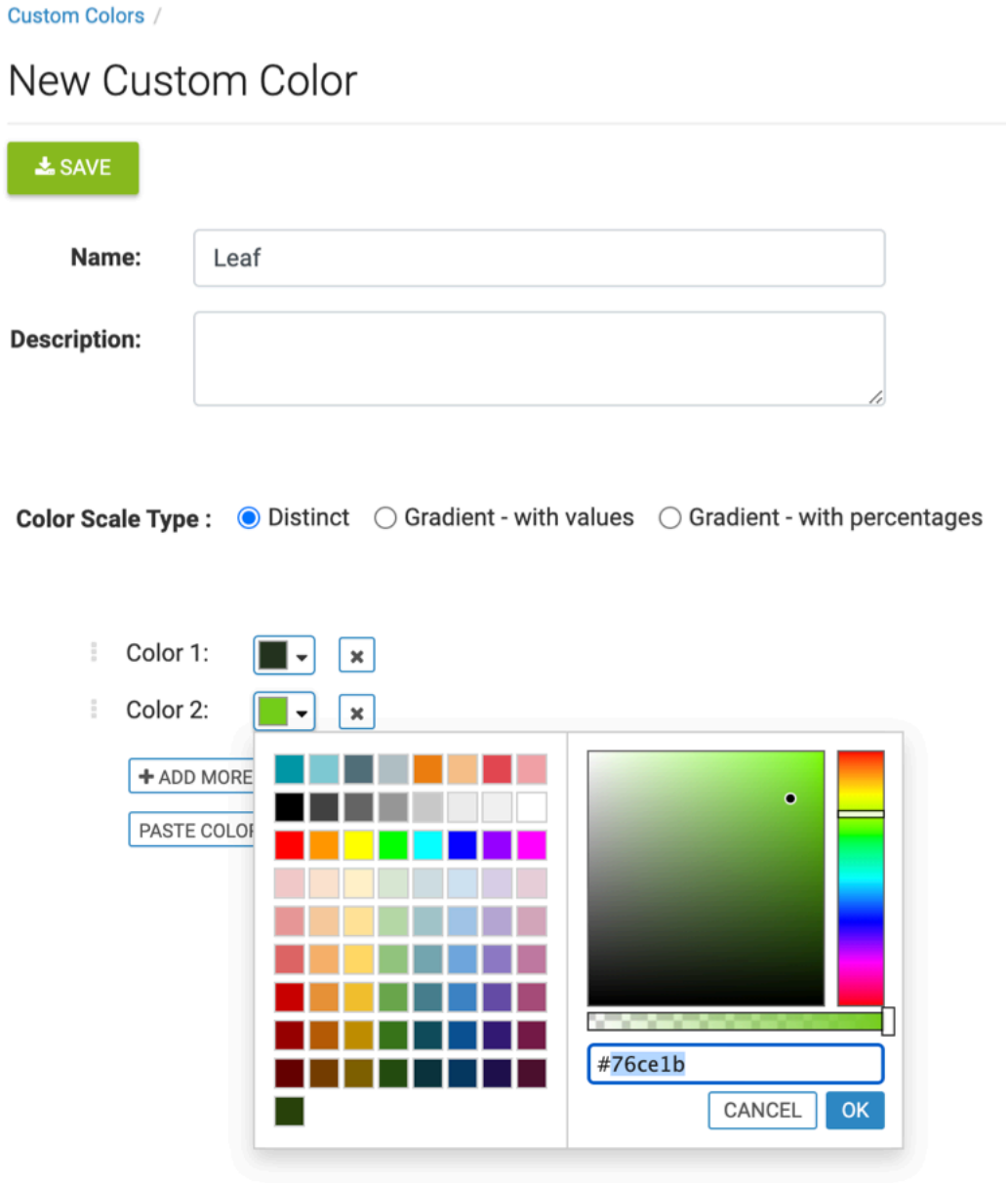

7. You can click Add More for other colors.

   In this example, the following colors have been added to the custom color Leaf:

Color	Hexadecimal Value
Color 1	#25351f
Color 2	#28410e
Color 3	#476b17
Color 4	#82b332
Color 5	#aec551
Color 6	#f7f05d
Color 7	#f1c150

Color	Hexadecimal Value
Color 8	#dc8451
Color 9	#fb5f2d
Color 10	#e03035
Color 11	#a10917

Color	Hexadecimal Value
Color 12	#510812

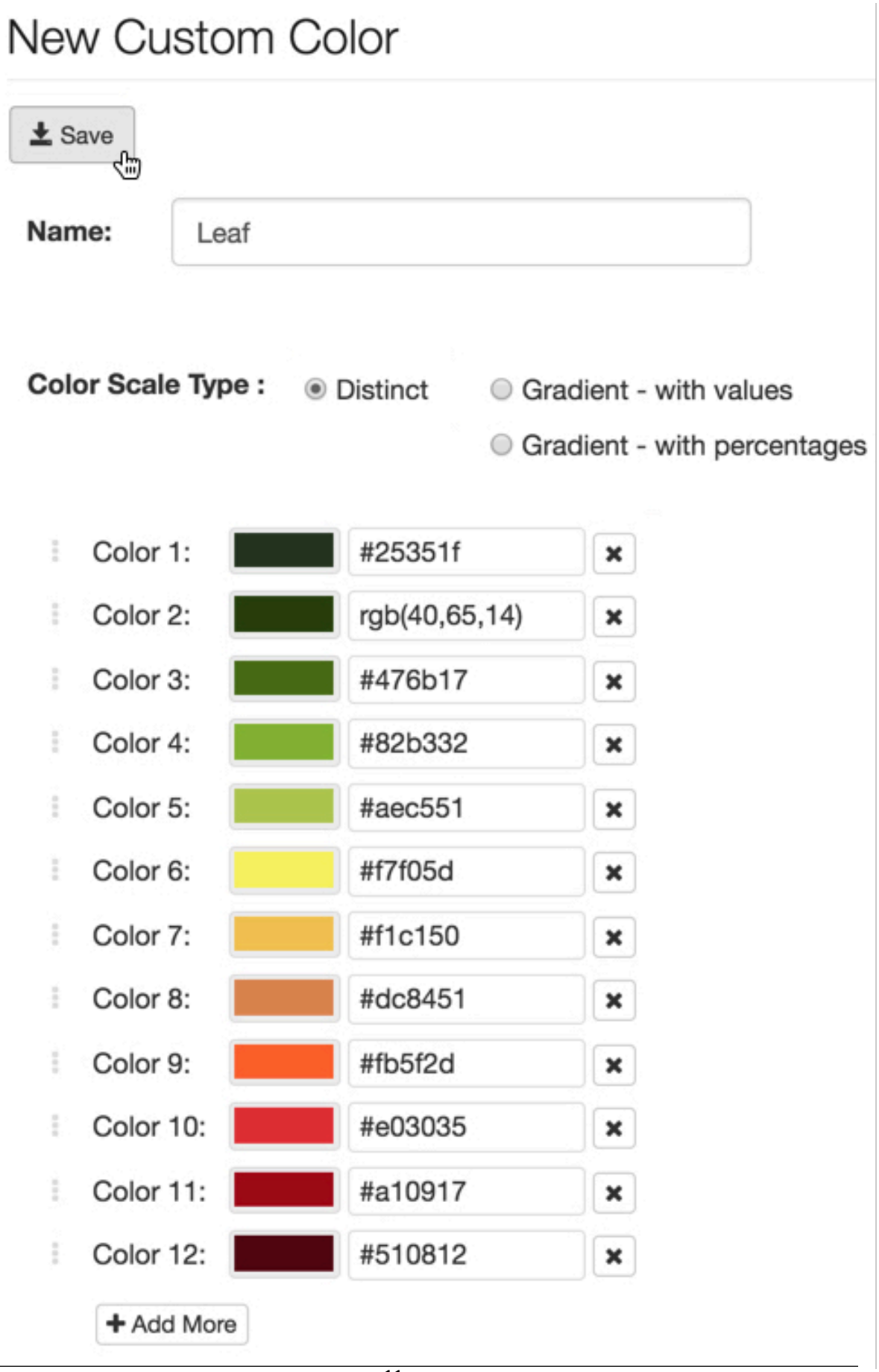
New Custom Color
Save
Name:
Leaf
Color Scale Type :
Distinct
Gradient - with values
Gradient - with percentages
Color 1:
#25351f
Color 2:
rgb(40,65,14)
Color 3:
#476b17
Color 4:
#82b332
Color 5:
#aec551
Color 6:
#f7f05d
Color 7:
#f1c150
Color 8:
#dc8451
Color 9:
#fb5f2d
Color 10:
#e03035
Color 11:
#a10917
Color 12:
#510812
Add More

8. Click SAVE.
9. Click the Custom Colors tab to return to the Custom Colors interface.

Results

The new color, Leaf appears in the list of colors on the Manage Custom Colors interface. Its type is distinct, and you have an option to edit and delete it.

Creating custom colors with gradient values

In Cloudera Data Visualization, users with administrative role can define custom color palettes that are gradients, based on a value.

About this task

Note: Not all of the visual types support gradient value palettes.

Perform these actions in the New Custom Color interface:

Procedure

1. Enter the Name of the new color.
   In this example, Ocean is used.
2. In Color Scale Type, choose Gradient - with values.
3. On the color bar, click the color indicator on the left (the red square over the gray bar).

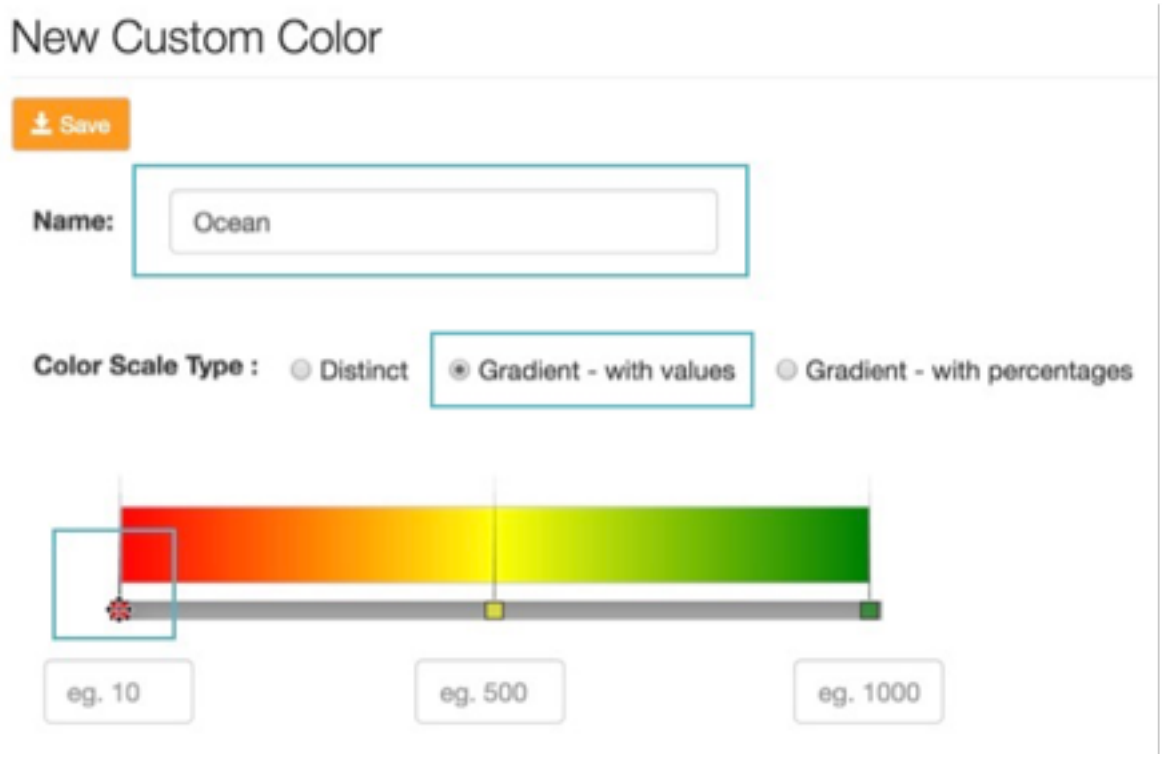

The color selection widget appears.

4. Point to another color and click it.
   You can also enter the color as RGB values or in hexadecimal format.

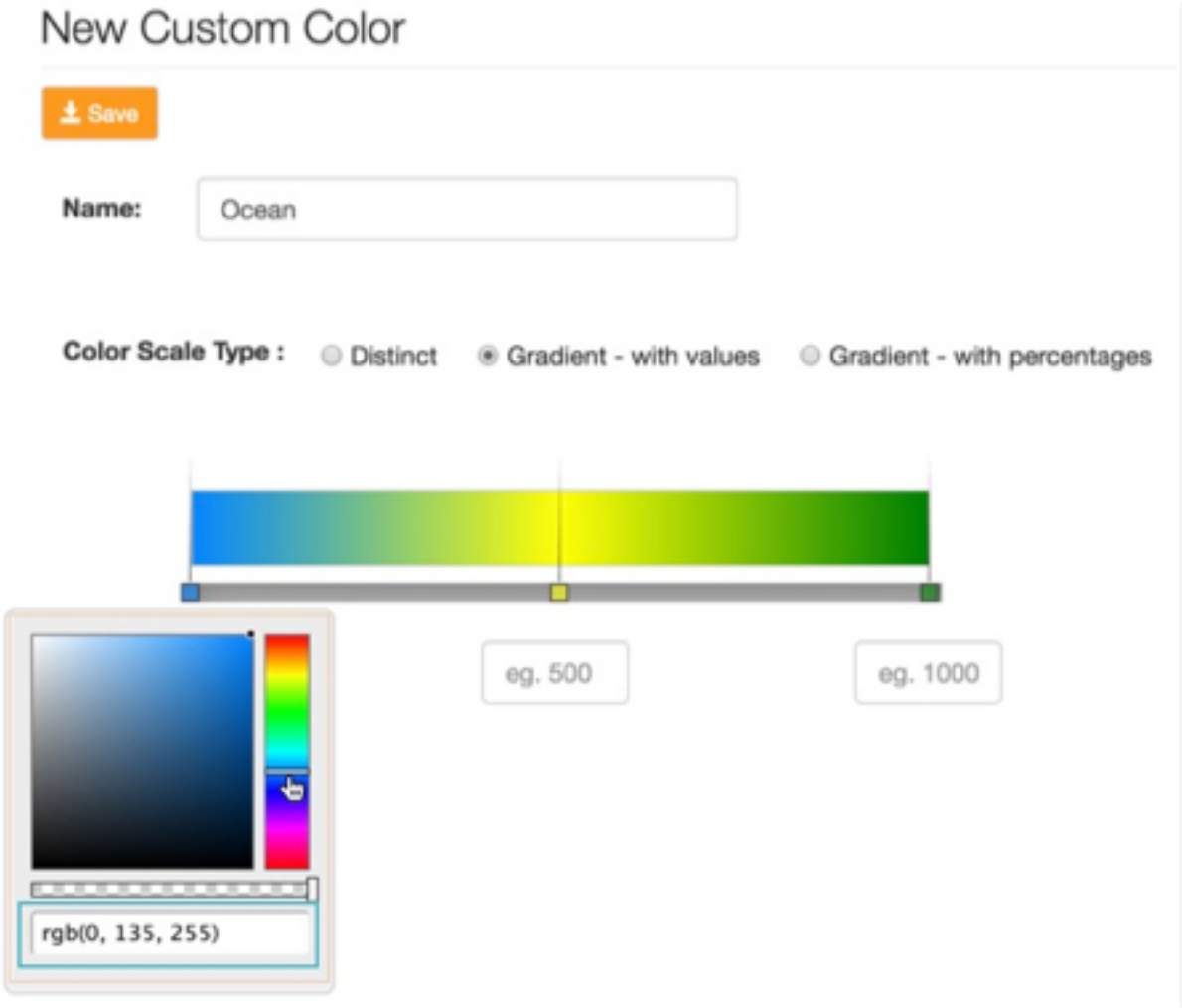

5. To adjust the color shade, slide indicator on the shade bar to the left.

   Values more to the left are progressively more 'Grey'. Placing the indicator on the extreme right (default position) generates the most intense color.

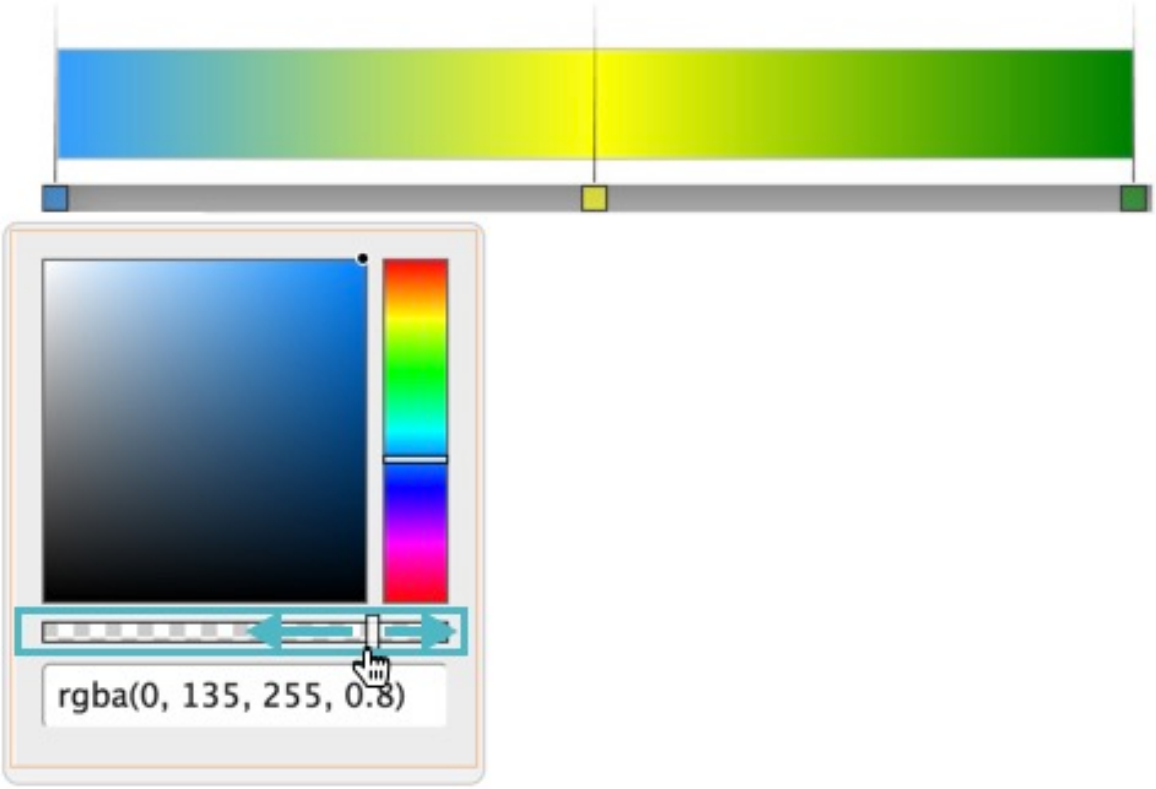

   Another option for color selection is to manually adjust the color by dragging the black dot on the square color selection field to the desired location.

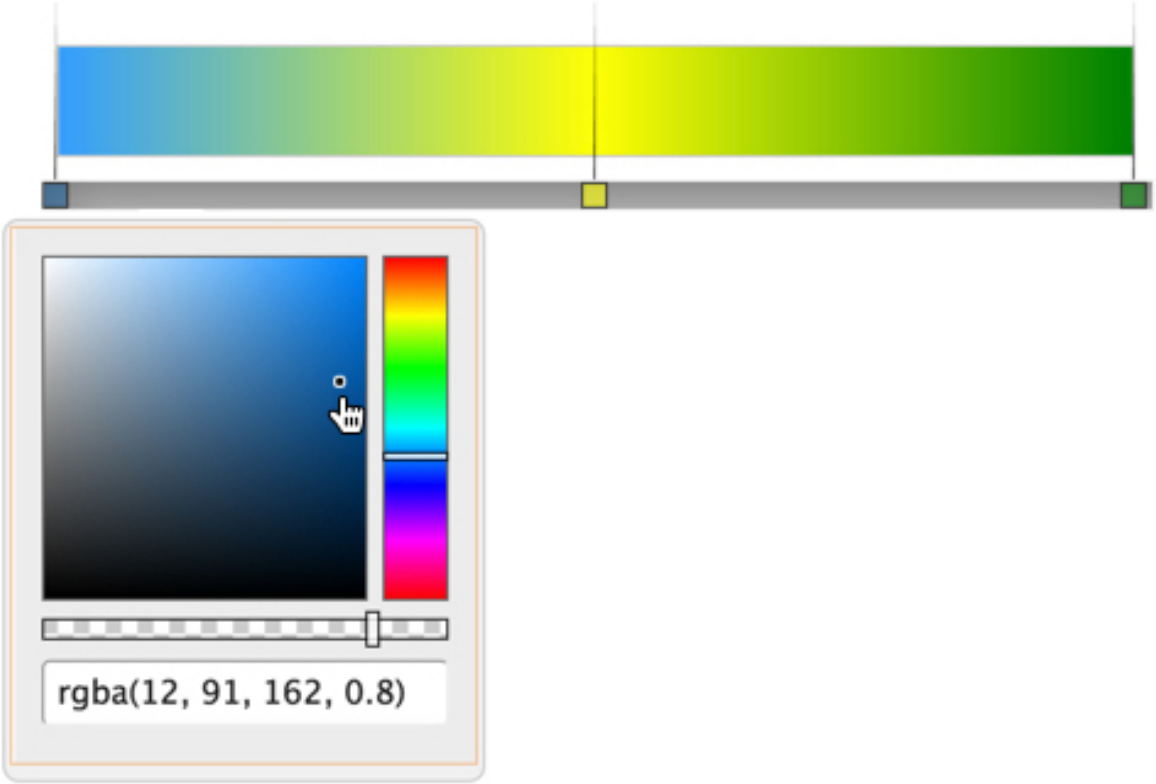

6. Configure the color on the right side of the color gradient.

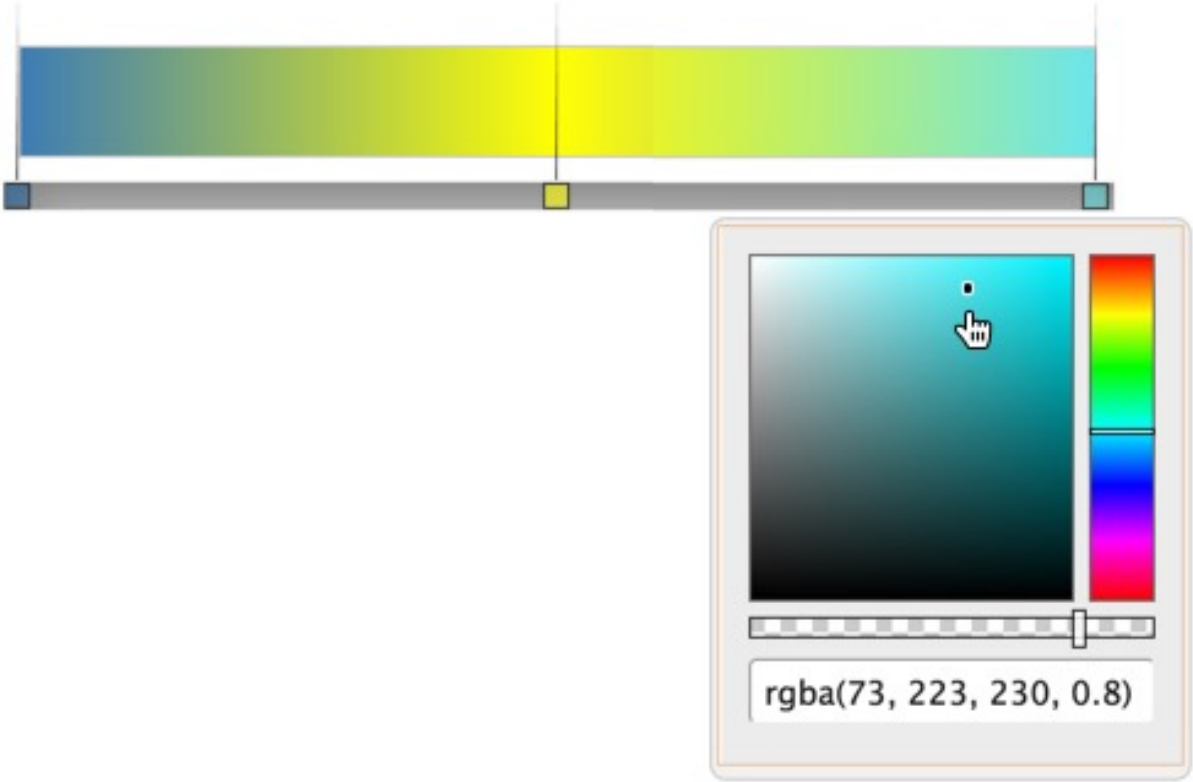

7. Configure the color in the middle of the color gradient.

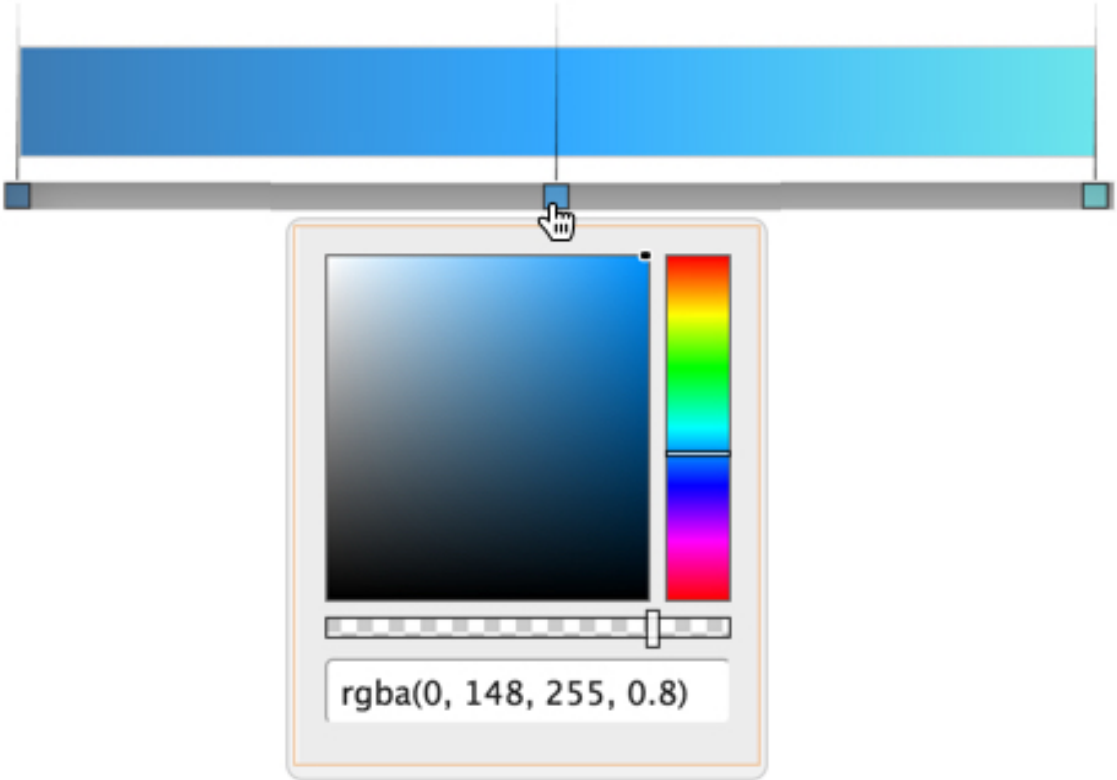

8. You can adjust the position of the middle color by sliding its indicator to the left or the right of the color gradient.

In the text fields under the color gradient, the three color indicators specify the values that correspond to the range and midpoint values in the data you plan to represent by the custom color.

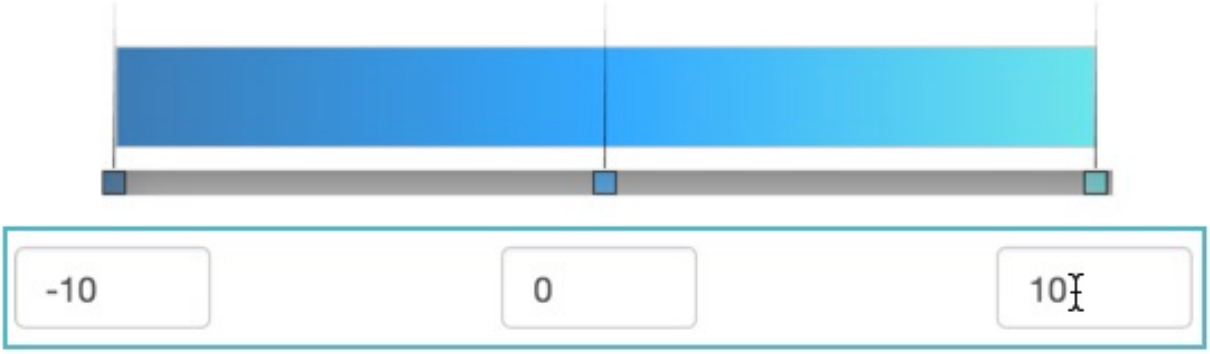

9. Click SAVE.
10. Click Custom Colors to return to the Custom Colors interface.

Results

The new color, Ocean appears in the list of colors on the Manage Custom Colors interface. Its type is Gradient-with values, and you have an option to edit and delete it.

Creating custom colors with gradient percentages

In Cloudera Data Visualization, users with administrative role can define gradient with percentage custom color palettes.

About this task

Perform these actions in the New Custom Color interface:

Procedure

1. Enter the Name of the new color. In this example, Elections is used.
2. In Color Scale Type, choose Gradient - with percentages.

3. On the color bar, click the color indicator on the left (the red square over the gray bar) and select another color from the color menu either by using the color slide bar, the color field, the shade field, or the text box for color code.

   In this example, blue color rgb(0, 136, 255) is used.

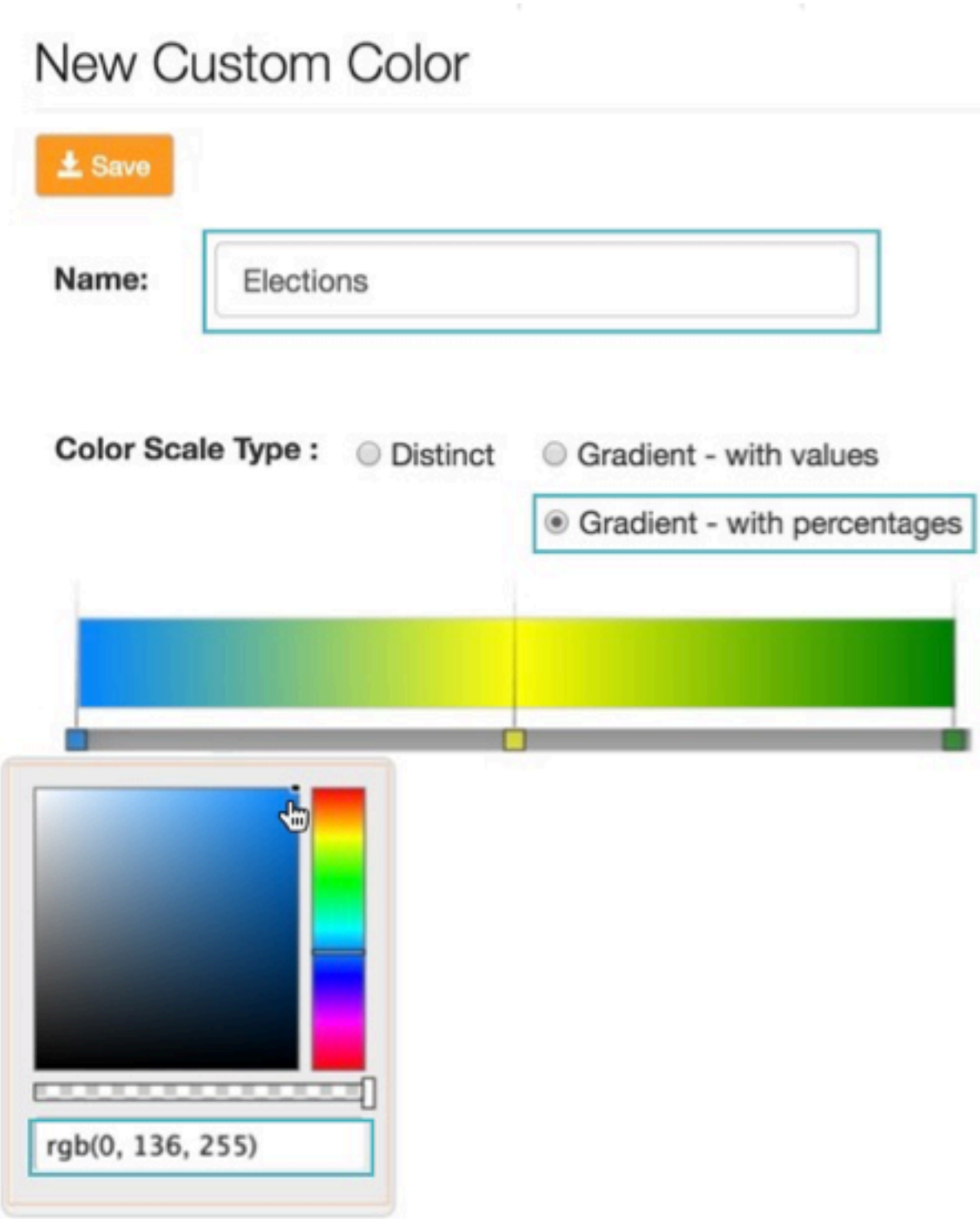

4. Configure the color in the middle of the color gradient.

   In this example, white rgb(255, 255, 255) is used.

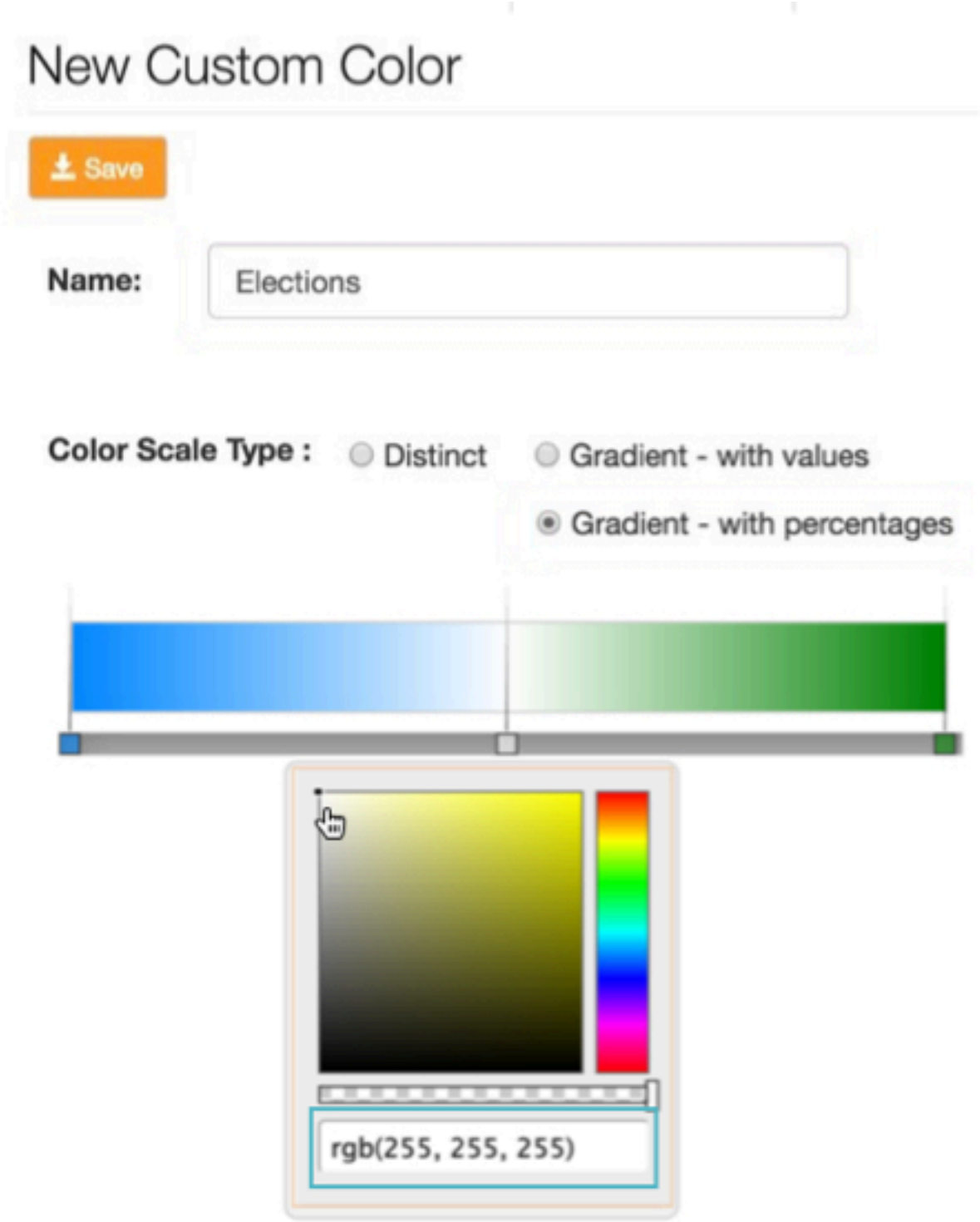

5. Configure the color on the right of the color gradient.

   In this example, red rgb(255, 42, 0) is used.

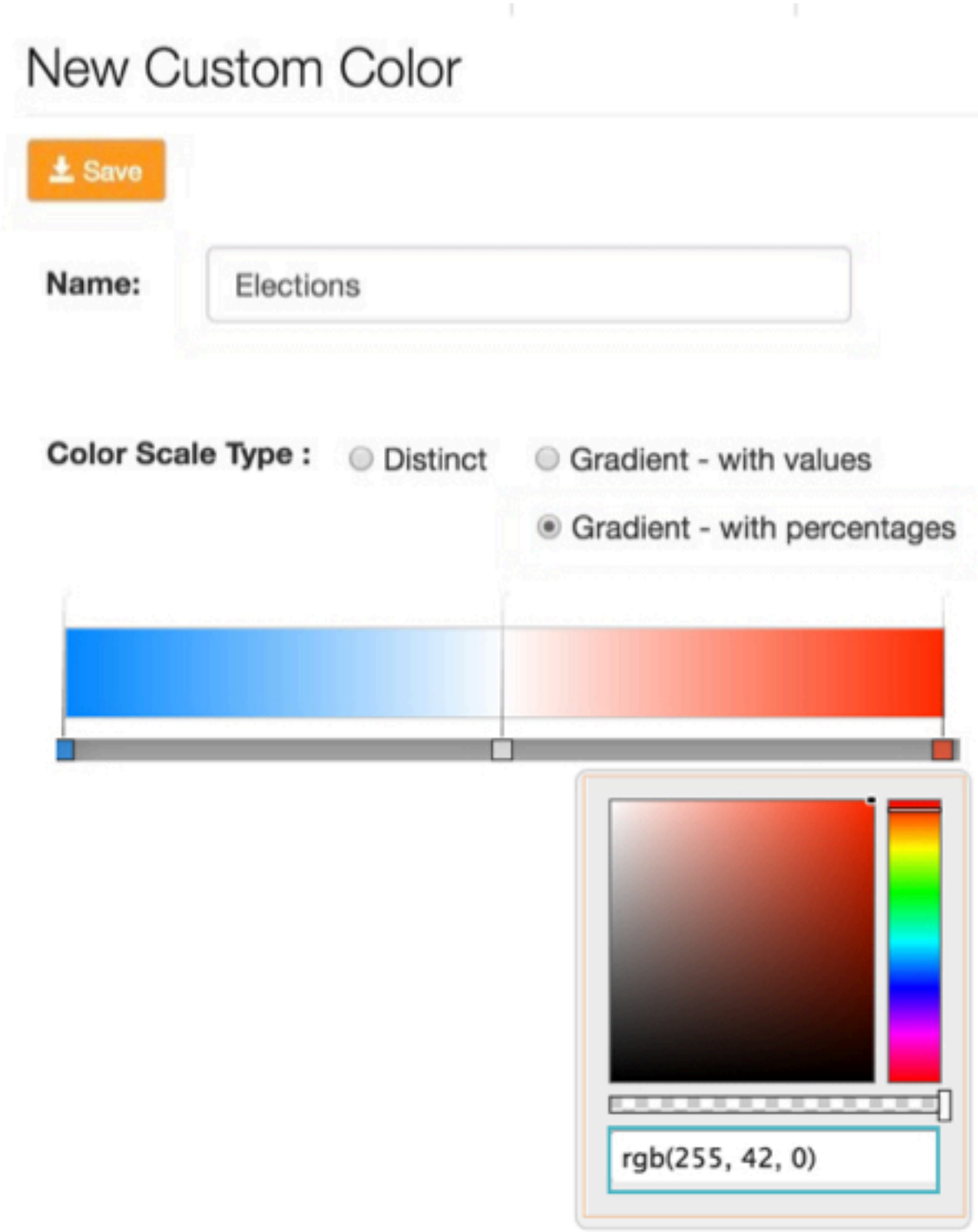

6. Click SAVE.
7. Click Custom Colors to return to the Custom Colors interface.

Results

Example

You can see that the new color, Election appears in the list of colors on the Manage Custom Colors interface. Its type is Gradient-with percentages, and you have an option to edit and delete it.

Editing custom colors

From time to time, it may become necessary to change the definition of a custom color. It is possible to change the name of the color palette, specifications of individual color components, and so on. However, after it is defined, a color palette's type cannot change.

About this task

Note: This feature is only available to users with administrative privileges.

To edit a custom color, follow these steps.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu, and select Custom Colors.

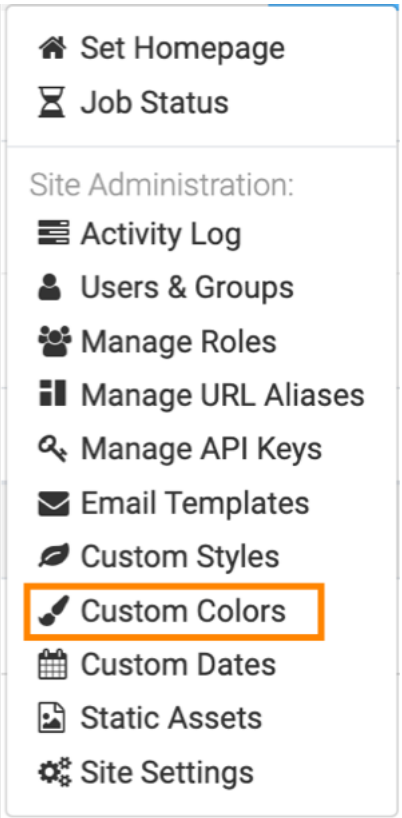

2. In the Manage Custom Colors interface, select the color to edit, and click its Pencil icon.

   In this example, the custom color Elections is edited.

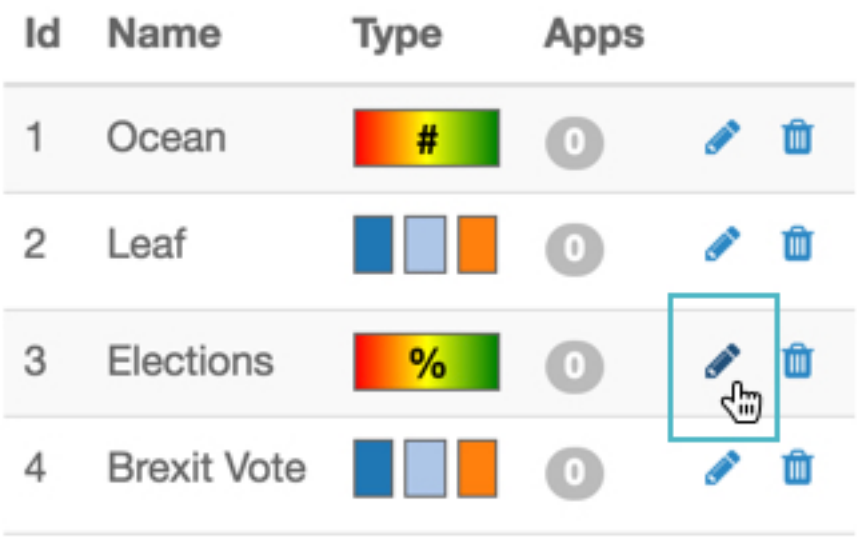

3. In the New Custom Color interface, change the Name.

   In this example, the name from Elections has been changed to US Presidential Elections.

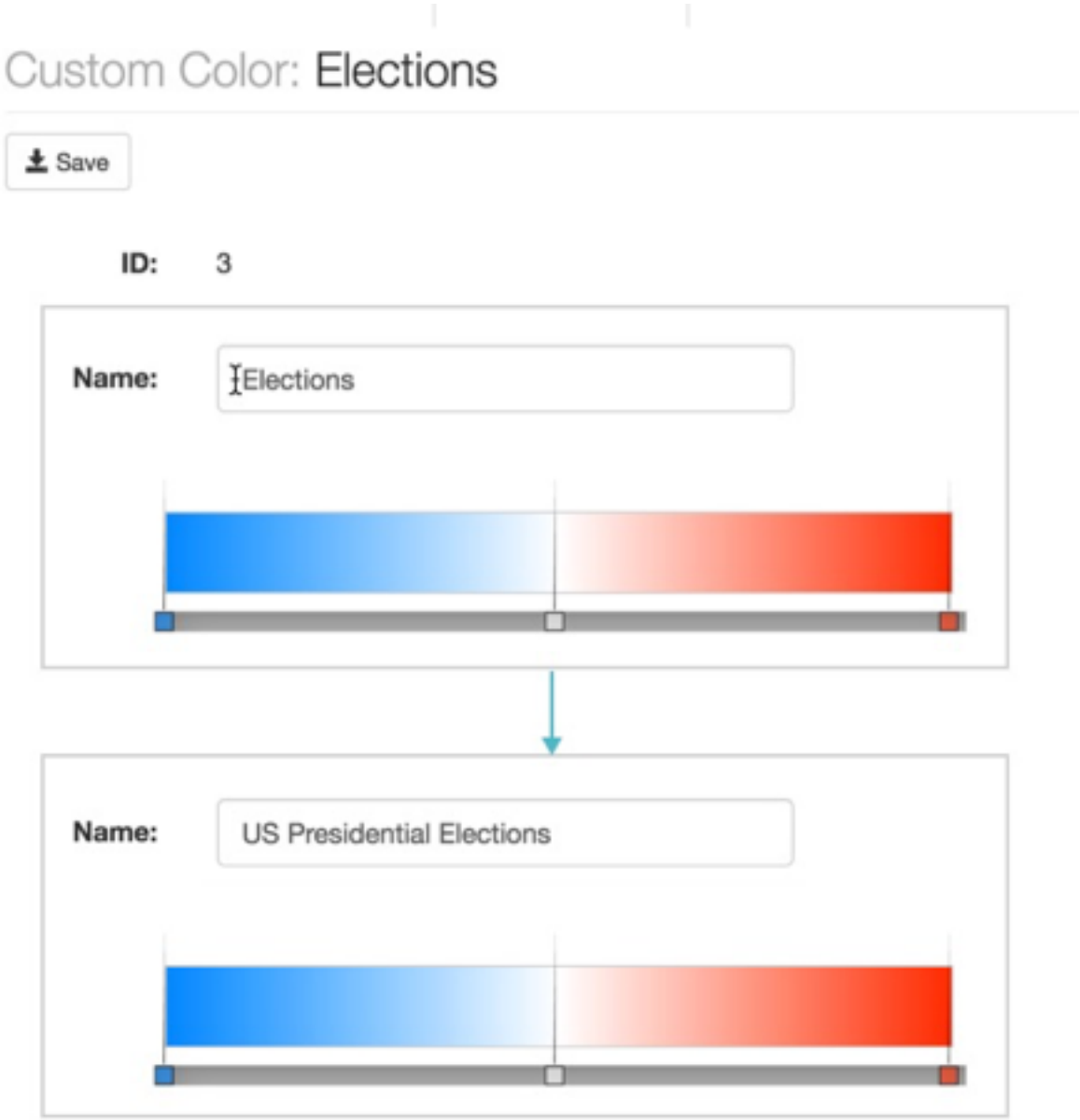

4. To change the definition of a component color:
   a) For gradient options, both gradient values and gradient percentages, select the color indicator, and change the color definition either through the text box, the color slider, the shade slider, or the color selection field.

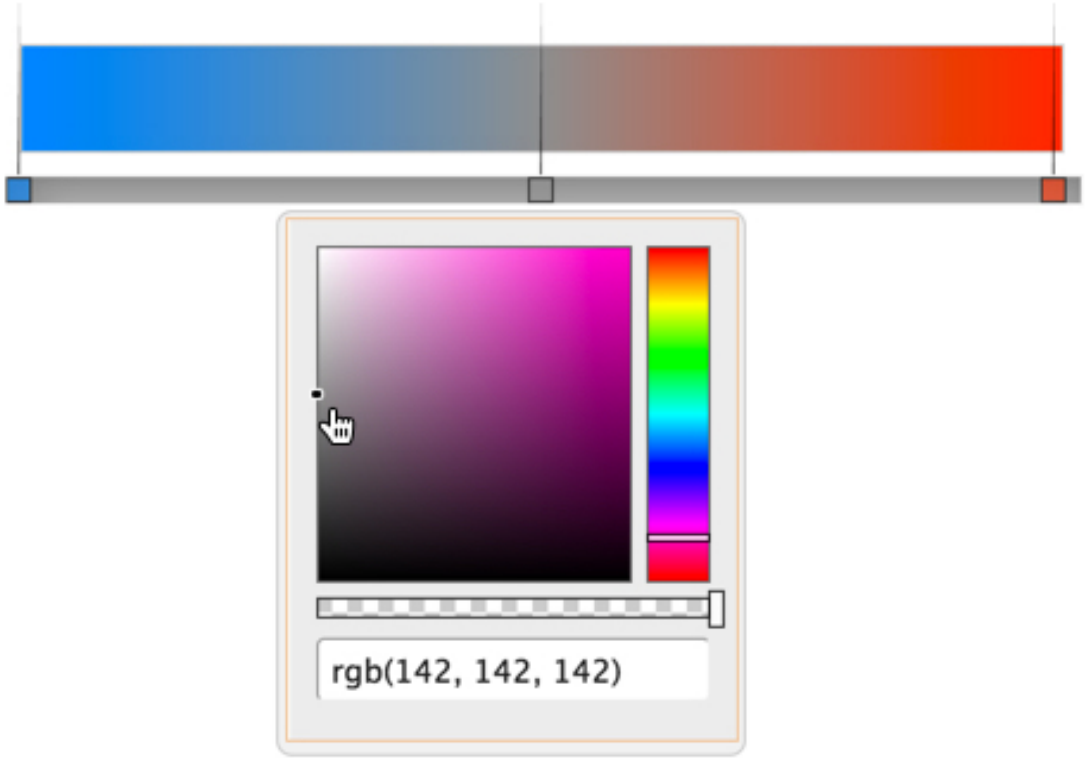

   b) For the distinct color option, edit each color separately. You can either click on the color indicator and use the color selection sliders and fields, or change the numeric color definition.

5. To change the position of the middle color on gradients, slide the selector for that color to the desired position.

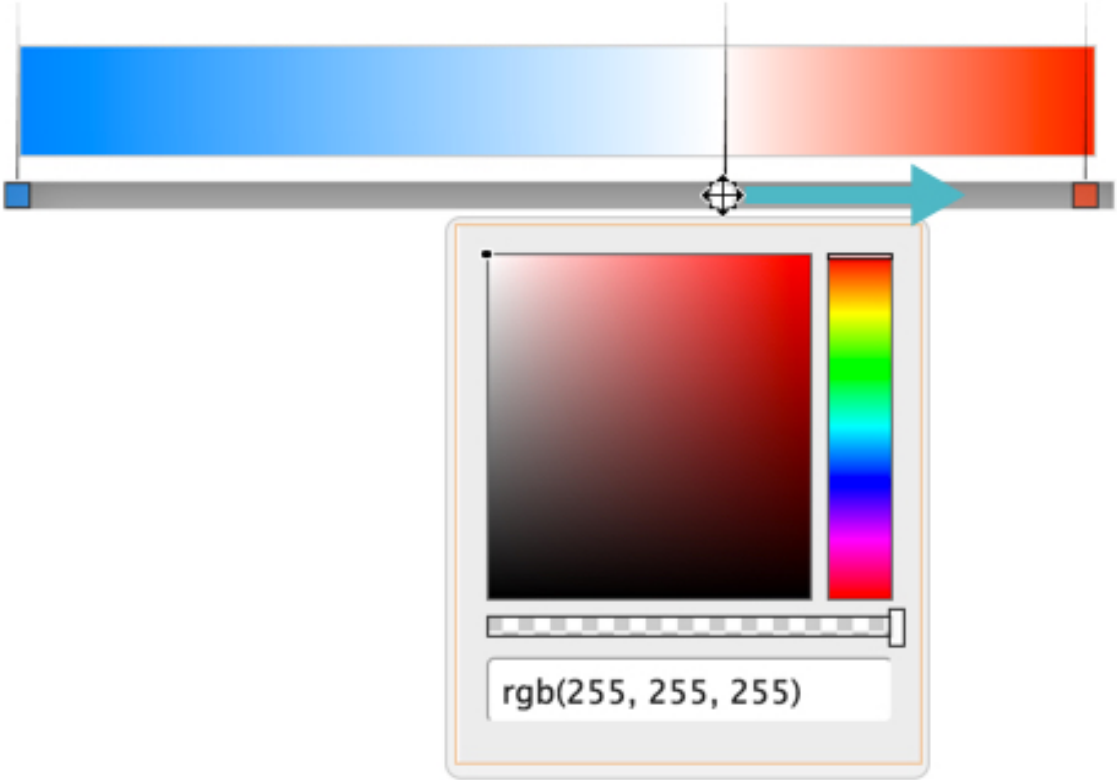

6. To remove a color from a distinct color grouping, click the remove icon.

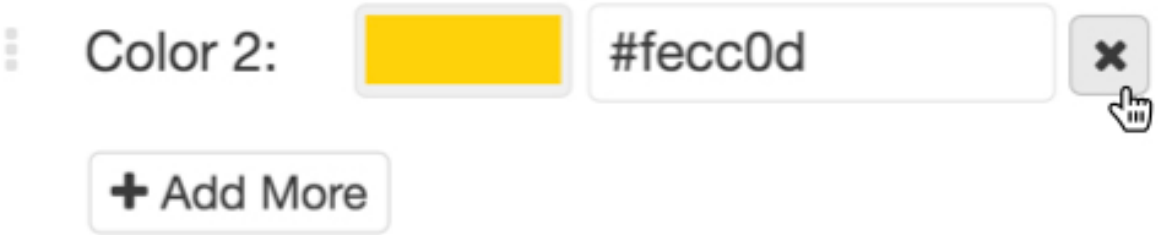

7. Click SAVE.
8. Click the Custom Colors tabe to return to the Custom Colors interface.

Results

You can see that Elections is now renamed US Presidential Elections in the list of custom colors.

Deleting custom colors

Cloudera Data Visualization enables you to delete previously defined colors.

About this task

Note: This feature is only available to users with administrative privileges.

To delete a custom color, follow these steps.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu, and select Custom Colors.

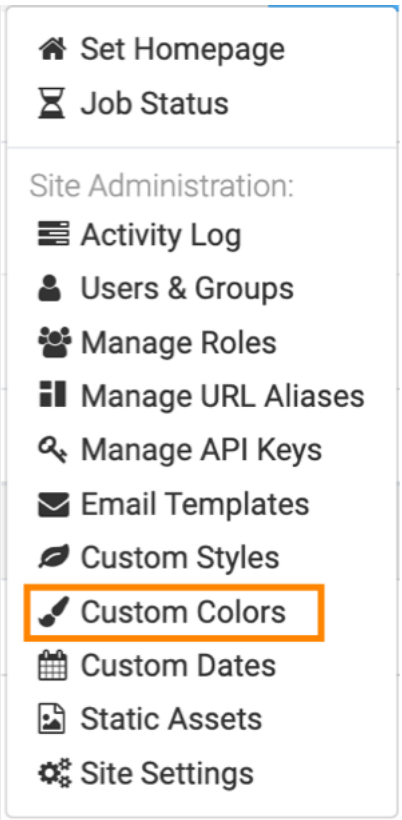

2. In the Manage Custom Colors interface, select the color to delete, and click its Trashcan icon.
3. In the Delete Confirmation modal window, type DELETE, and then click Delete.

Results

The custom color you have deleted is no longer on the list of custom colors.

Cloudera Data Visualization 7.2.9

Custom Dates

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing custom dates.....4

Creating new custom date ranges.....5

Cloning custom date ranges..... 12

Deleting custom date ranges.....13

Managing custom dates

Cloudera Data Visualization enables you to define custom sets of date ranges, and anchor them to a specific date and time. For real-time filters, you can anchor the filters to today, or to now.

About this task

You can use the custom sets of date ranges on dashboards to filter time-based data, and help you analyze streaming input.

Note: This feature is only available to users with administrative privileges.

To use the Manage Custom Dateranges interface, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu and select Custom Dates.

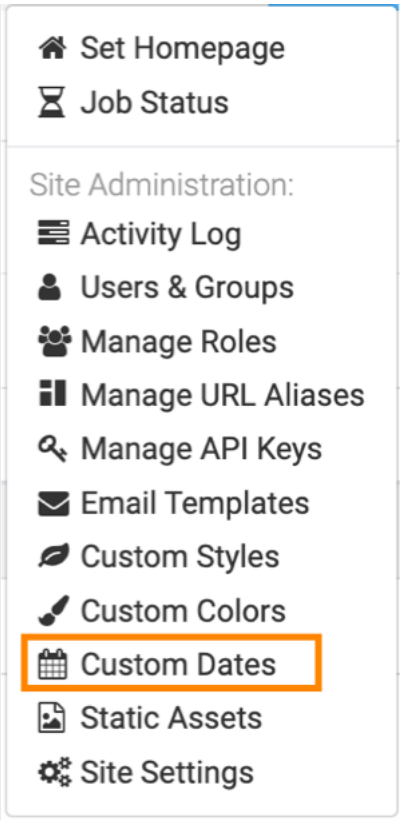

2. You have the following options in the Manage Custom Dateranges interface:
   - Click NEW CUSTOM DATERANGE to create a new asset. For more information, see *Creating new custom date ranges*.
   - Click the Pencil icon to edit the parameters of a set of dateranges, or to start the cloning process. For more information, see *Cloning custom date ranges*.
   - Click the Trashcan icon to delete the asset. For more information, see *Deleting custom date ranges*.

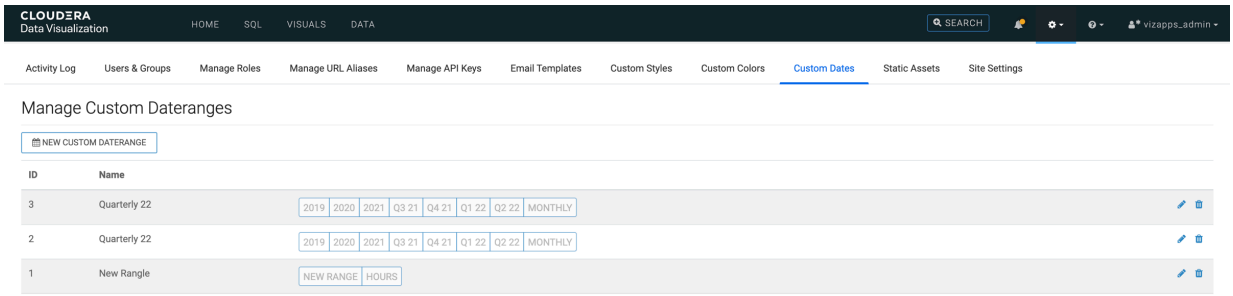

Related Information

Creating new custom date ranges

Cloning custom date ranges

Deleting custom date ranges

Creating new custom date ranges

About this task

To create a new set of custom date range, you must define each custom range in turn.

Here, we demonstrate how to build a set of custom date ranges called My Date Range Set. We define the first one date range, 3 min, in detail. Then, we define the rest of them in similar manner.

Follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu and select Custom Dates.

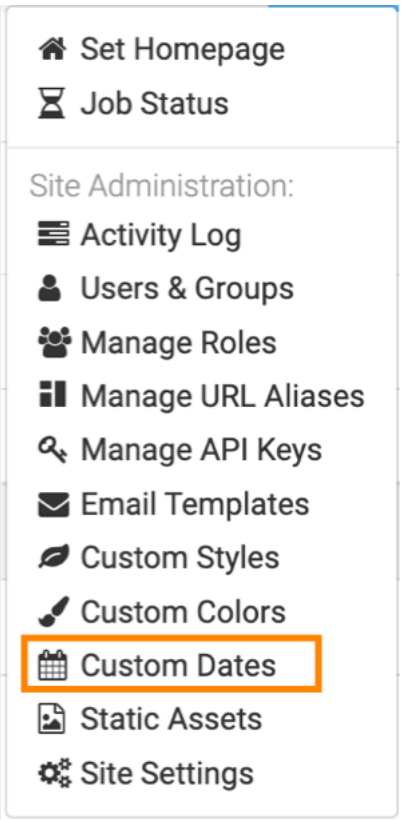

2. In the Manage Custom Dateranges interface, click NEW CUSTOM DATERANGE.
3. In the Custom Date interface, enter the Name of the custom date.

   In this example, My Date Range is used.

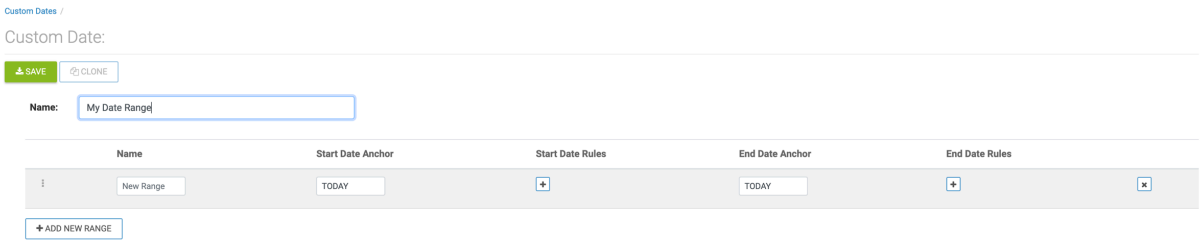

4. In the list of ranges, define the first range:
   a) Enter the Name of the new range.
   b) Under Start Date Anchor, select one of the options:
      - Anchor to Today (default)
      - Anchor to Now
      - Anchor to Date, which is the current date by default. To specify an alternate date, click in the edit field and select from the calendar interface.
      - Anchor to Time, which is the current timestamp by default. To specify an alternate timestamp, click in the edit field and select from the calendar and time interface.

      Click APPLY.
   c) Under Start Date Rules, click the Plus icon and proceed to specify the first stat date rule:
      - Click the text box with operations (Add) and select the operation for the date rule:

        Add (default), Subtract, Startof, or Endof.
      - Click the text box with numbers, and select the appropriate time offset.
      - Click the text box with units, and select one of the time units from the menu:

        Seconds, Minutes, Hours, Days, Weeks, Months, Years, or Quarters (default). We selected Minutes.

      This completes the specification of the first start date rule.

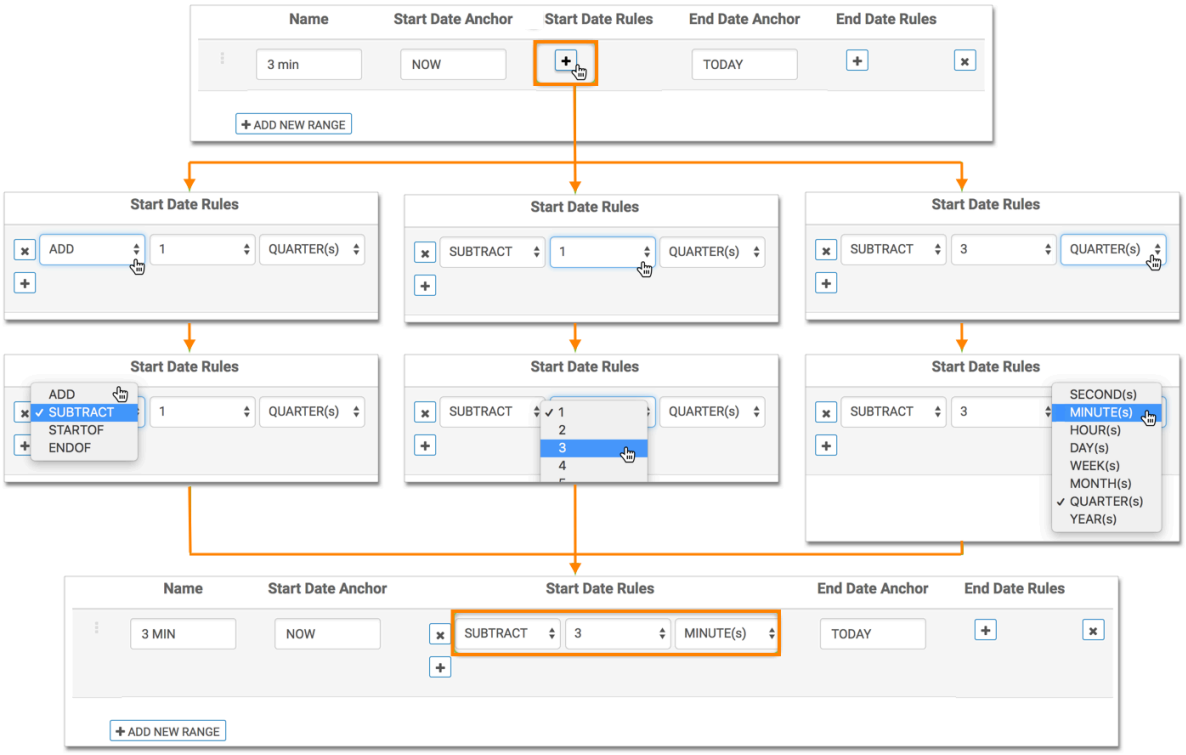

To delete or remove a rule, click the Delete icon beside the rule.

d) Under End Date Anchor, select one of the options:

- Anchor to Today (default)
- Anchor to Now
- Anchor to Date, which is the current date by default. To specify an alternate date, click in the edit field and select from the calendar interface.
- Anchor to Time, which is the current timestamp by default. To specify an alternate timestamp, click in the edit field and select from the calendar and time interface.

This works just like the Start Date Anchor.

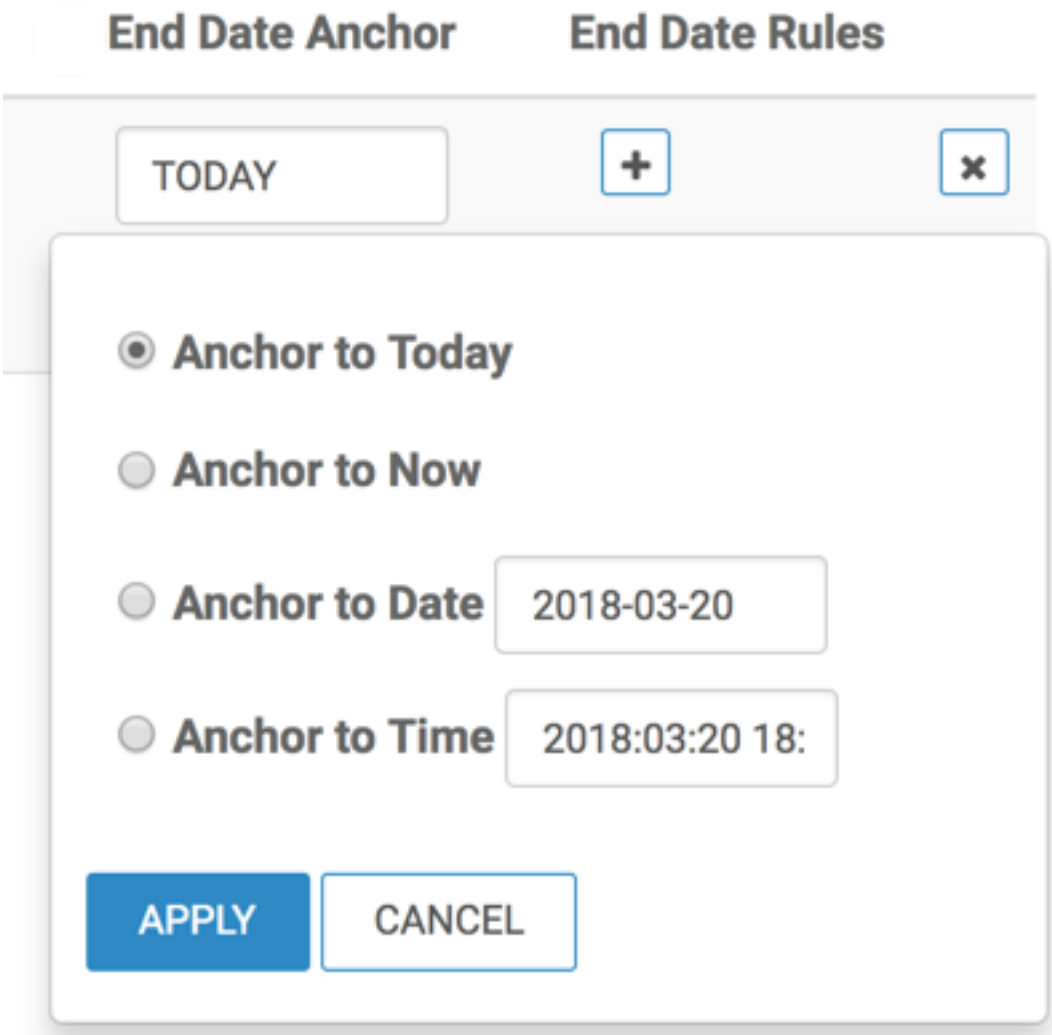

Click APPLY.

e) Under End Date Rules, click the Plus icon.

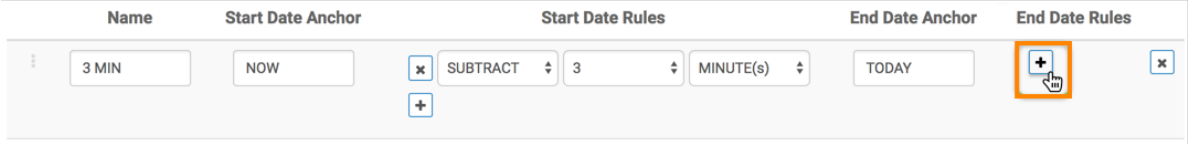

Proceed to specify the first rule:

- Click the text box with operations (Add), and select the operation for the date rule:

  Add (default), Subtract, Startof, or Endof
- Click the text box with numbers, and select the appropriate time offset.
- Click the text box with units, and select one of the time units from the menu:

  Seconds, Minutes, Hours, Days, Weeks, Months, Years, or Quarters (default).

This completes the specification of the first end date rule.

To delete or remove a rule, click the Delete icon beside the rule.

To delete or remove a rule, click the Delete icon beside the rule.

f) To remove the entire date range, click Remove icon at the end of the row that specifies the date range.

5. Click SAVE at the top of the interface to save the first date range.

6. Following the earlier steps, create additional dateranges and click SAVE.

   In this example, the following dateranges have been created:

   3 min5 min, 10 min, 15 min, 30 min, 1 hr, 3 hr, 6 hr, 9 hr, 12 hr, 1 day, 3 day, 5 day, 1 wk, 2 wk, 1 month, 2 month, 1 quarter, 2 quarter, 1 year, 2 year, 3 year

Custom Dates / ID: 1

Custom Date: My Date Range Set

SAVE CLONE

Name: My Date Range Set

Name	Start Date Anchor	Start Date Rules			End Date Anchor	End Date Rules
3 MIN	NOW	SUBTRACT	3	MINUTE(s)	TODAY	
5 MIN	NOW	SUBTRACT	5	MINUTE(s)	TODAY	
10 MIN	NOW	SUBTRACT	10	MINUTE(s)	TODAY	
15 MIN	NOW	SUBTRACT	15	MINUTE(s)	TODAY	
30 MIN	NOW	SUBTRACT	30	MINUTE(s)	TODAY	
1 HR	NOW	SUBTRACT	1	HOUR(s)	TODAY	
3 HR	NOW	SUBTRACT	3	HOUR(s)	TODAY	
6 HR	NOW	SUBTRACT	6	HOUR(s)	TODAY	
9 HR	NOW	SUBTRACT	9	HOUR(s)	TODAY	
12 HR	NOW	SUBTRACT	12	HOUR(s)	TODAY	
1 DAY	NOW	SUBTRACT	1	DAY(s)	TODAY	
3 DAY	NOW	SUBTRACT	3	DAY(s)	TODAY	
5 DAY	NOW	SUBTRACT	5	DAY(s)	TODAY	
1 WK	NOW	SUBTRACT	1	WEEK(s)	TODAY	
2 WK	NOW	SUBTRACT	2	WEEK(s)	TODAY	
1 MONTH	NOW	SUBTRACT	1	MONTH(s)	TODAY	
2 MONTH	NOW	SUBTRACT	2	MONTH(s)	TODAY	
1 QUARTER	NOW	SUBTRACT	1	QUARTER(s)	TODAY	

Results

In the Manage Custom Dateranges interface, you can see that the list contains the new set, My Date Range, with all defined ranges.

Cloning custom date ranges

About this task

To create a new set of custom date ranges based on an existing one, you can make a clone of a set, and edit it. In this example it is demonstrated how to create My Date Range Set - Fine Control, a subset of My Date Range Set that features dateranges from 3 minutes to 1 hour.

Follow these steps:

Procedure

1. In the Manage Custom Dateranges interface, click Pencil icon next to the set of date ranges to clone.

2. In the Custom Date interface, click the Clone button.
3. Change the Name field.

   In this example, Clone of My Date Range Set is renamed to My Date Range Set - Fine Control.

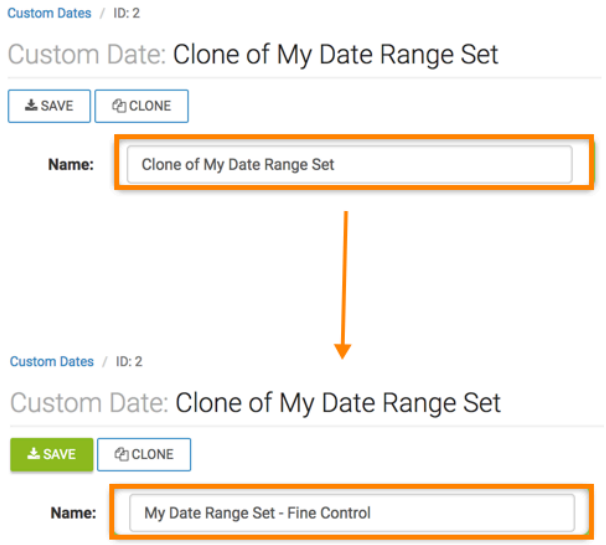

4. In the list of ranges, delete the elements longer than 1 hour.

5. Click Save at the top of the interface.

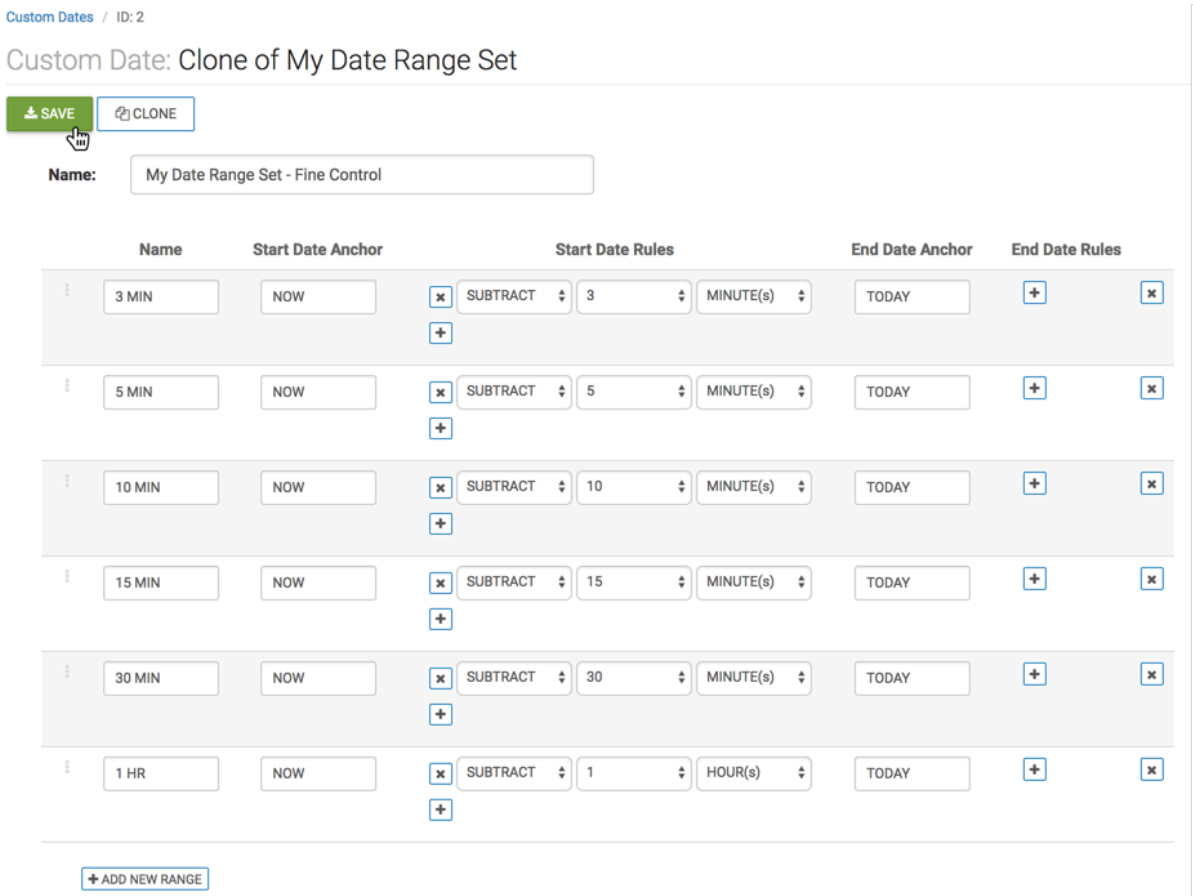

6. Click Custom Dates at the top of the interface.

   In the Manage Custom Dateranges interface, you can see that the list contains the new set, My Date Range Set - Fine Control, with a shorter list of ranges than the original set.

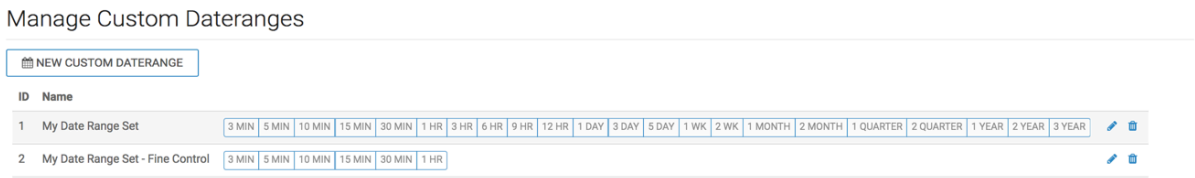

7. Create another set based on My Date Range Set, this time with dateranges of 1 Day or greater. Name it My Date Range Set - Gross Control.

Deleting custom date ranges

About this task

To delete set of custom date ranges, follow these steps:

Procedure

1. In the Manage Custom Dateranges interface, click the Trashcan icon next to the set of date ranges to delete.

   In this example, My Date Range Set - Gross Control will be deleted.

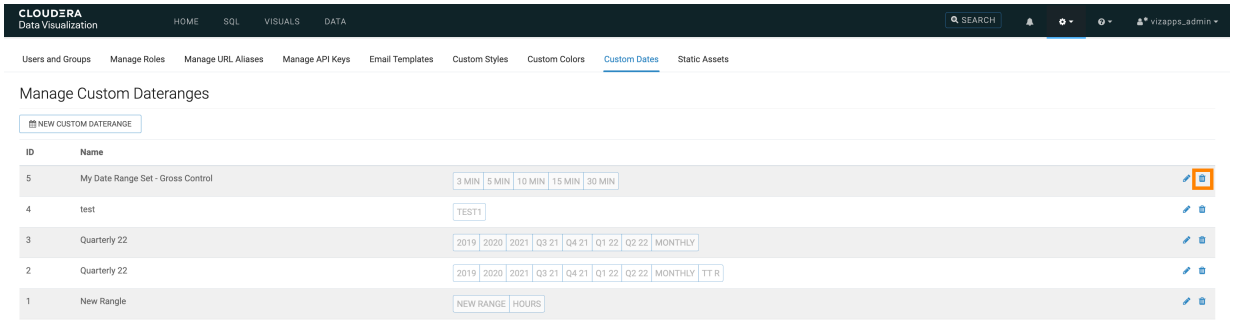

2. In the Delete Confirmation modal window, type DELETE, and then click the DELETE button.

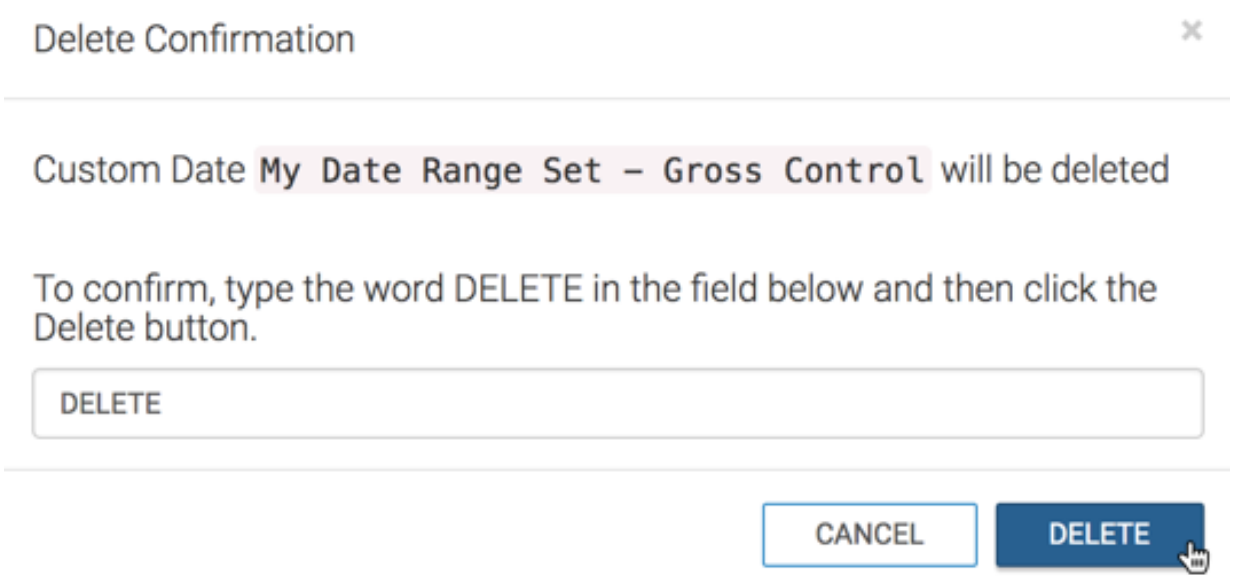

Results

In the Manage Custom Dateranges interface, notice that the list no longer has the deleted set.

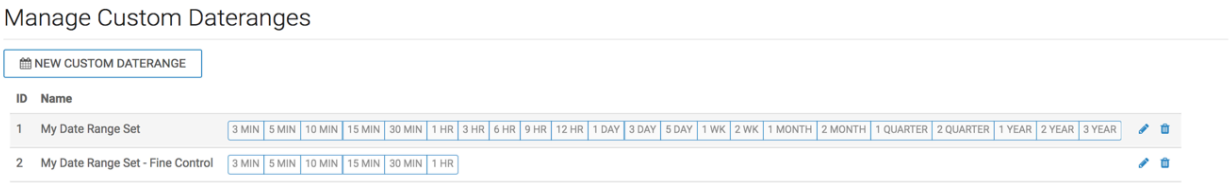

Cloudera Data Visualization 7.2.9

Managing Jobs

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating new email jobs **4**
Immediate email 4
Schedule-based email 6

Rerunning jobs **11**

Deleting job logs **12**

Pausing and resuming jobs **13**
Pausing running jobs 13
Resuming paused jobs 14

Changing the owner of a job **15**

Editing jobs **16**

Deleting scheduled jobs **20**

Creating new schedule intervals **21**

Editing schedule intervals **23**

Deleting schedule intervals **25**

Creating new email jobs

Cloudera Data Visualization supports two types of email jobs:

Immediate email

You can share existing visuals with other users on an ad-hoc basis using email.

About this task

Follow these steps to email a visual, a sheet, or a dashboard of your choice. You can accomplish this task either from Edit mode or from View mode.

Before you begin

You have the option to use an email template for this task, but it is not required. You can send the email as-is, without a message, or include your own custom message. If you decide to use a template, you can choose the default template or create a custom one. For guidance, see Creating new email templates.

Note: Only users with administrative privileges can create and manage email templates. For more information, see *Creating new email templates*.

Procedure

1. Open the dashboard that you want to share through email or navigate to the visual that you want to include in the email.
2. Access the supplemental menu.

   For visuals:

   Click ••• in the upper-right corner of the visual.

   For dashboards:

   Click the same icon (•••) but in the upper-right corner of the dashboard.
3. Select Email from the drop-down menu.

   The Send visual or Send sheet modal appears.

4. Select Email now.

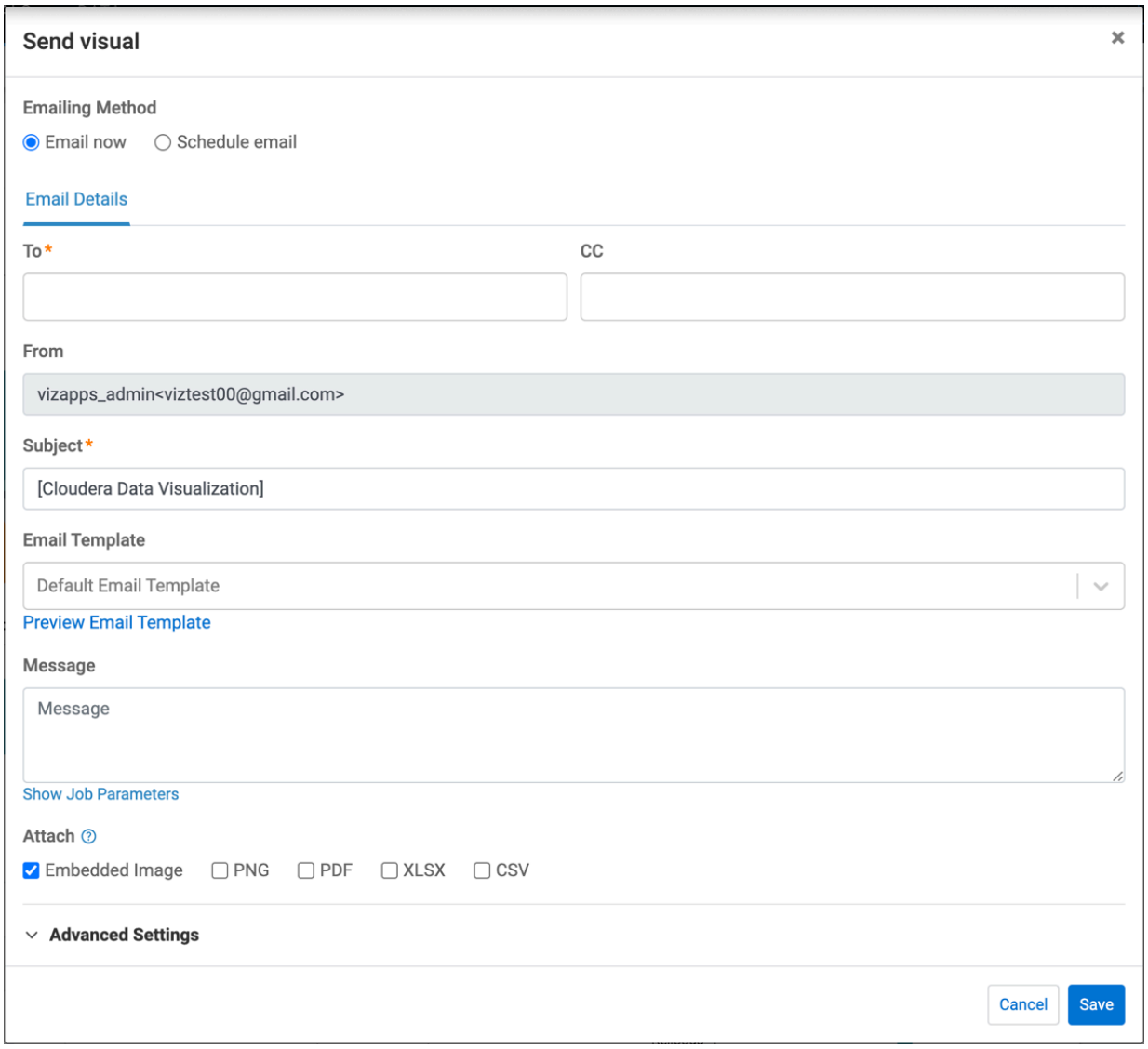

5. Fill in the required details.

   Fields marked with an asterisk are mandatory.

   - To* – Enter the primary recipients' email addresses.
   - CC – Enter the email addresses of additional recipients.
   - From – You can set the value of this field in Site Settings.
   - Subject* – You can modify the default title of the email.

   Note:

   The subject includes a default prefix you specified when enabling email and the title of the visual. The default prefix is [Cloudera Data Visualization].

6. You can set the Email Template that you want to use.

   If you want to see how the template looks, click Preview Email Template.

7. Enter the body text of the email in the Message field.
8. Click Show Job Parameters to view the list of parameters you can add to the message.

9. Choose how the visual is included in the email.

   You can either embed the visual directly into the email body (default option) or attach it in one of the following formats:

   - PNG
   - PDF
   - XLSX
   - CSV

   Important: The attachment type cannot be changed after the job is configured and saved.

   Note: Email larger than 25 MB might not be sent, depending on the limits of the email server.

10. Configure Advanced Settings.

    - Error Notification Emails To – Specify an email address to receive error messages.
    - Reply To – Specify a different reply-to address if needed.
    - Use Alias – Create a new or use an existing alias for easier access.
    - Page Parameters – For visuals that have parameterized fields on their shelves, you can choose:
      - Include page parameters (default)
      - Do not include page parameters

    You can also show or hide the page parameters.

11. Click Save.

    A notification appears while the email is prepared. Once completed, a success message confirms the operation.

What to do next

You can choose to review the list of pending emails, cancel any running email jobs, or resend previously sent emails. For more information, see *Managing jobs*.

Related Information

Creating new email templates

Managing jobs

Schedule-based email

You can schedule regular emails to share visuals, sheets, and dashboards. These jobs run automatically based on a defined schedule or or are triggered when specific dataset measures meet predefined thresholds.

About this task

Follow these steps to schedule sending emails of visual, sheet, or dashboard. You can accomplish this either in Edit or in View mode.

Before you begin

- You have the option to use an email template for this task, but it is not required. You can send the email as-is, without a message, or include your own custom message. If you decide to use a template, you can choose the default template or create a custom one. For guidance, see Creating new email templates.

  Note: Only users with administrative privileges can create and manage email templates. For more information, see *Creating new email templates*.

- Before you schedule an email, check the Enable scheduled jobs option in Site Settings Jobs . For more information, see Enabling scheduled jobs in the Site Settings documentation.

Procedure

1. Open the dashboard that you want to share through email or navigate to the visual that you want to include in the email.
2. Access the supplemental menu.

   For visuals:

   Click ••• in the upper-right corner of the visual.

   For dashboards:

   Click the same icon (•••) but in the upper-right corner of the dashboard.
3. Select Email from the drop-down menu.

   The Send visual or the Send sheet modal appears.
4. Select the Schedule email option.

5. Fill in the fields on the available tabs.
   - If Threshold is not enabled, you will see the Email Details and Schedule Details tabs.
   - If Threshold is enabled, the Threshold Details will also appear.

     With threshold, you can schedule an email job that is triggered by measures on the dataset reaching a defined threshold. It means that email notifications are sent when the specified triggering condition occurs.

For Email Details

Fields marked with an asterisk are mandatory.

- To* – Enter the primary recipients' email addresses.
- CC – Enter the email addresses of additional recipients.
- From – You can set the value of this field in Site Settings.
- Subject* – You can modify the default title of the email.

  Note:

  The subject includes a default prefix you specified when enabling email and the title of the visual. The default prefix is [Cloudera Data Visualization].
- You can set the Email Template that you want to use.

  If you want to see how the template looks, click Preview Email Template.
- Enter the body text of the email in the Message field.
- [Optional] Click Show Job Parameters to view the list of parameters you can add to the message.
- Choose how the visual is included in the email.

  You can either embed the visual directly into the email body (default option) or attach it in one of the following formats:

  - PNG
  - PDF
  - XLSX
  - CSV

  Important: The attachment type cannot be changed after the job is configured and saved.

  Note: Email larger than 25 MB might not be sent, depending on the limits of the email server.
- Configure Advanced Settings.
  - Error Notification Emails To – Specify an email address to receive error messages.
  - Reply To – Specify a different reply-to address if needed.
  - Use Alias – Create a new or use an existing alias for easier access.
  - Page Parameters – For visuals that have parameterized fields on their shelves, you can choose:
    - Include page parameters (default)
    - Do not include page parameters

    You can also show or hide the page parameters.

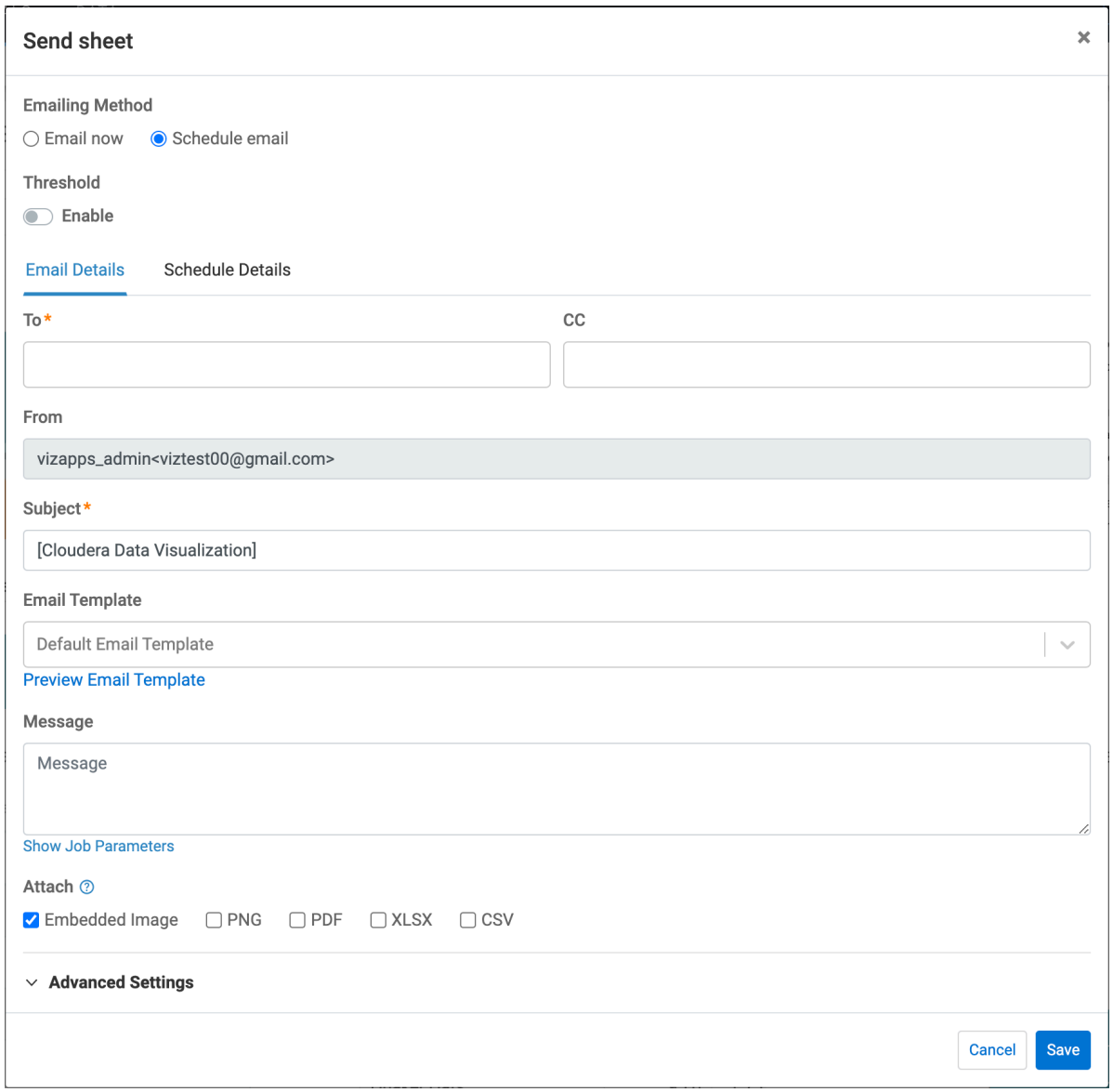

For Schedule Details

- Enter a name for the scheduled job.
- In the Schedule field of the Schedule Details tab, select an existing schedule interval from the following options:
  - Hourly (default)
  - Daily
  - Every Weekday
  - Weekly
  - Monthly
  - Quarterly
  - Custom intervals (available to administrators only)

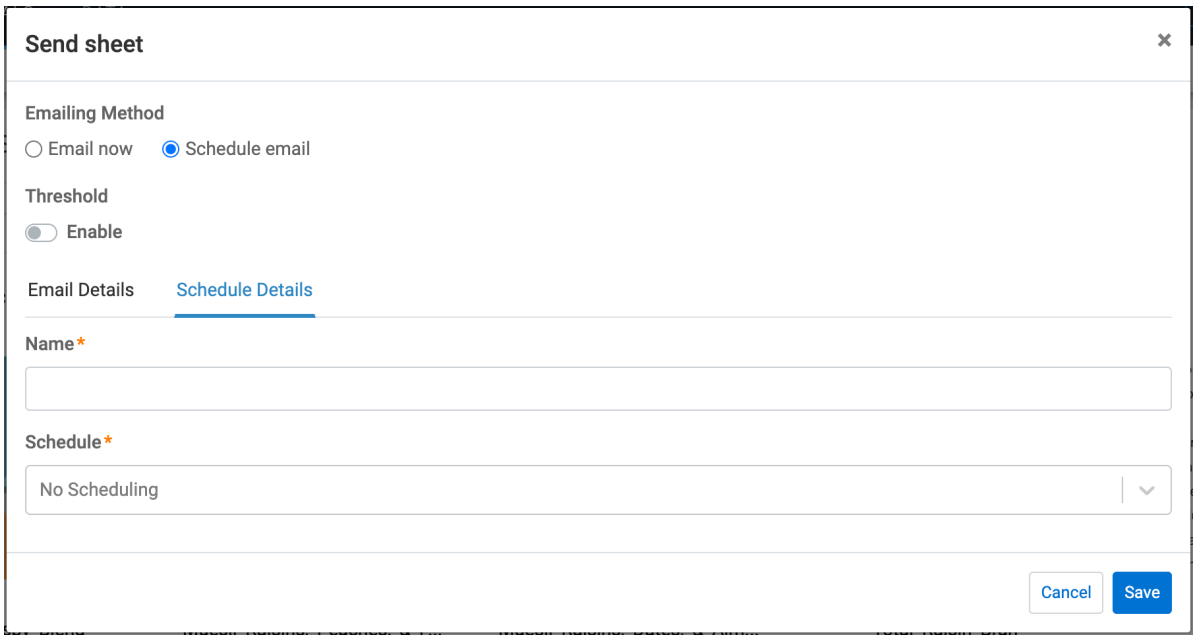

For Threshold Details

You can define a threshold-based trigger to send email notifications when specific conditions are met.

- Trigger Visual ID – Define a new trigger visual or add the ID of an existing table visual.
- Send Email Limit – Specify how often emails should be sent when the condition is met:
  - Only on first trigger
  - When trigger occurs
  - Once a half hour (default)
  - Once an hour
  - Once a day
  - Once a week
  - Once in 2 weeks
  - Once a month
- Trigger Type – Select one of the following options:
  - Threshold met: Triggers an email when the first measure of the first row in the trigger visual meets the conditional expression.
  - Value changed since last assessment: Triggers an email when the change in the first measure of the first row after the last assessment meets the conditional expression.
- Trigger Conditions – Set logical operators (one of <, >, #, #, or =) and values.

  You can add multiple trigger conditions with + Add Condition.

  Note: If you set multiple conditions, All conditions must be true to trigger the job.

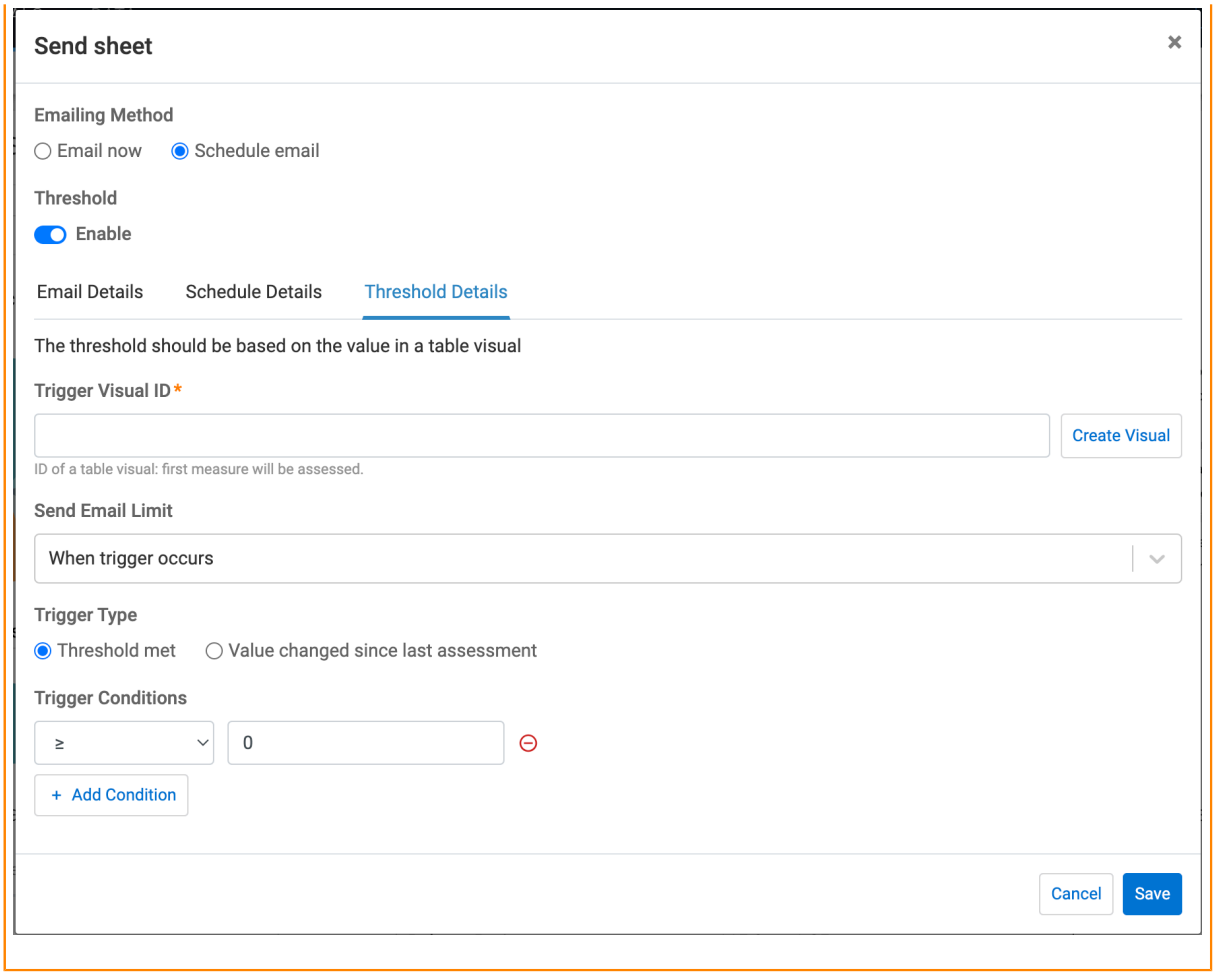

6. Click Save.

   A notification appears while the email is prepared. Once completed, a success message confirms the operation.

What to do next

You can choose to review the list of pending emails, cancel any running email jobs, or resend previously sent emails. For more information, see *Managing jobs*.

Related Information

Creating new email templates

Rerunning jobs

Cloudera Data Visualization allows you to rerun previously defined and performed jobs.

Procedure

1. Navigate to Runs (Latest) or Scheduled Jobs on the Jobs interface.
2. To rerun multiple jobs simultaneously, select them using the checkbox at the beginning of each row.

   Alternatively, you can use the checkbox located on the left side of the list header to select all jobs at once.

3. Click Run in the top-right corner to initiate the rerun process.

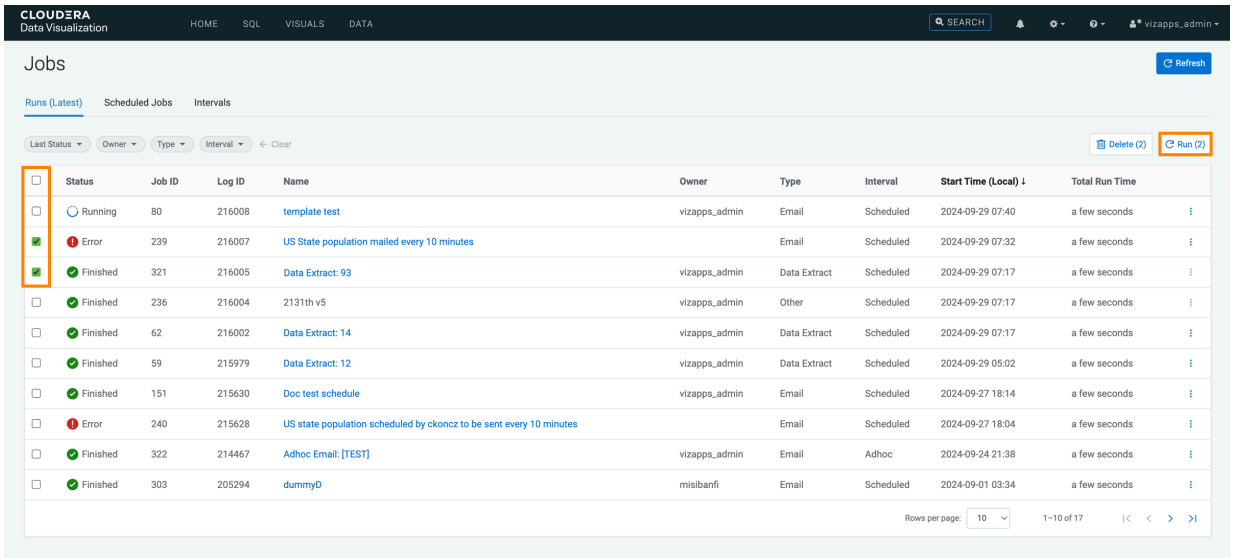

If you wish to rerun a single job, you can click ⋮ located at the end of the job row to open the supplemental menu and select Run Now.

You can also start a rerun from the Actions dropdown on the Job Details page.

4. Click Run jobs in the confirmation modal to proceed with the action.

   The modal lists the IDs and names of the jobs you selected for rerunning.

   As the rerun process progresses, the job log list refreshes automatically to show the status information for the new instances of the selected jobs. These statuses may appear as Running or Pending. Pending jobs do not display Start Time or Total Run Time until they commence operation.

Deleting job logs

Cloudera Data Visualization allows you to remove unnecessary job run logs from your log list.

Procedure

1. Navigate to the Runs (Latest) tab within the Jobs interface.
2. Select the job runs you wish to delete. To select all logs, click the checkbox on the left side of the list header.

3. Click Delete in the top-right corner.

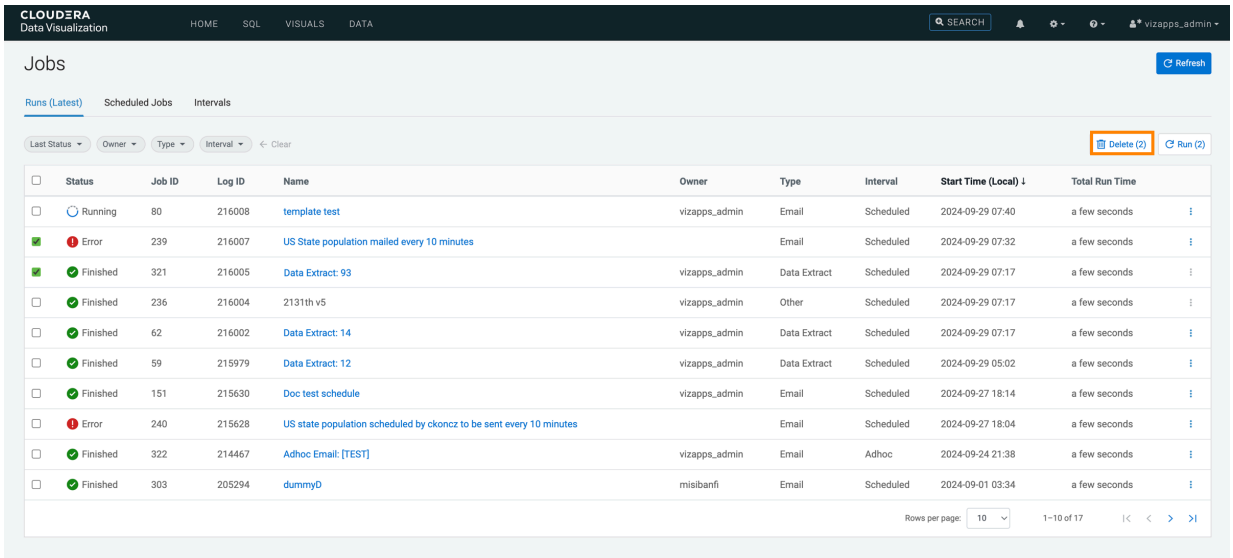

4. Click Delete Runs in the confirmation modal to proceed with the action.

   The modal lists the IDs and names of the run logs you selected for deletion.

Pausing and resuming jobs

Cloudera Data Visualization allows you to suspend and resume scheduled jobs providing flexibility in running your tasks.

Pausing running jobs

Procedure

1. Navigate to Scheduled Jobs on the Jobs interface.
2. Locate the job you wish to pause and click ⋮ located at the end of the job row to open the supplemental menu.

3. To pause a single job, locate the job you wish to pause and click ⋮ at the end of the job row to open the supplemental menu.

   To pause multiple jobs simultaneously, select them using the checkbox at the beginning of each row. If you want to pause all jobs, click the checkbox on the left side of the list header.

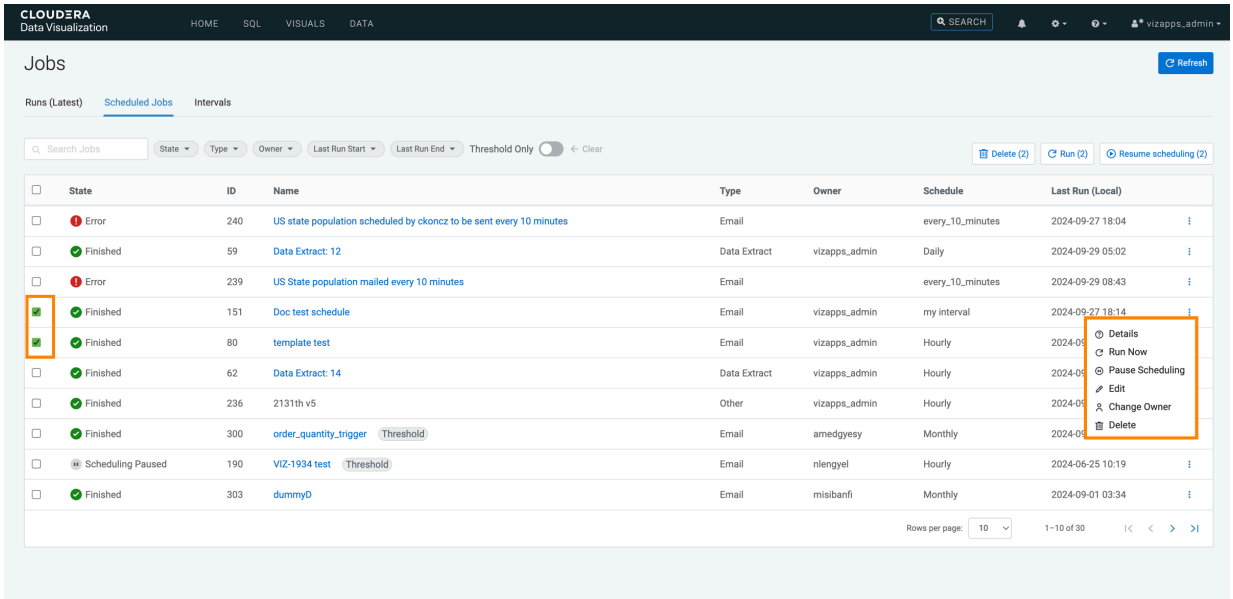

4. Click Pause Scheduling
5. Click Pause in the confirmation modal to proceed with the action.

   The modal lists the IDs and names of the jobs selected for pausing.

   A notification that the job is paused appears briefly, the status of the job changes to Scheduling Paused, and the Pause Scheduling option is replaced with Resume Scheduling in the supplemental menu.

Resuming paused jobs

Procedure

1. Navigate to Scheduled Jobs on the Jobs interface.

2. To rerun a single job, locate the job you wish to resume and click ⋮ at the end of the job row to open the supplemental menu.

   To resume multiple paused jobs simultaneously, select them using the checkbox at the beginning of each row. If you want to resume all jobs, click the checkbox on the left side of the list header.

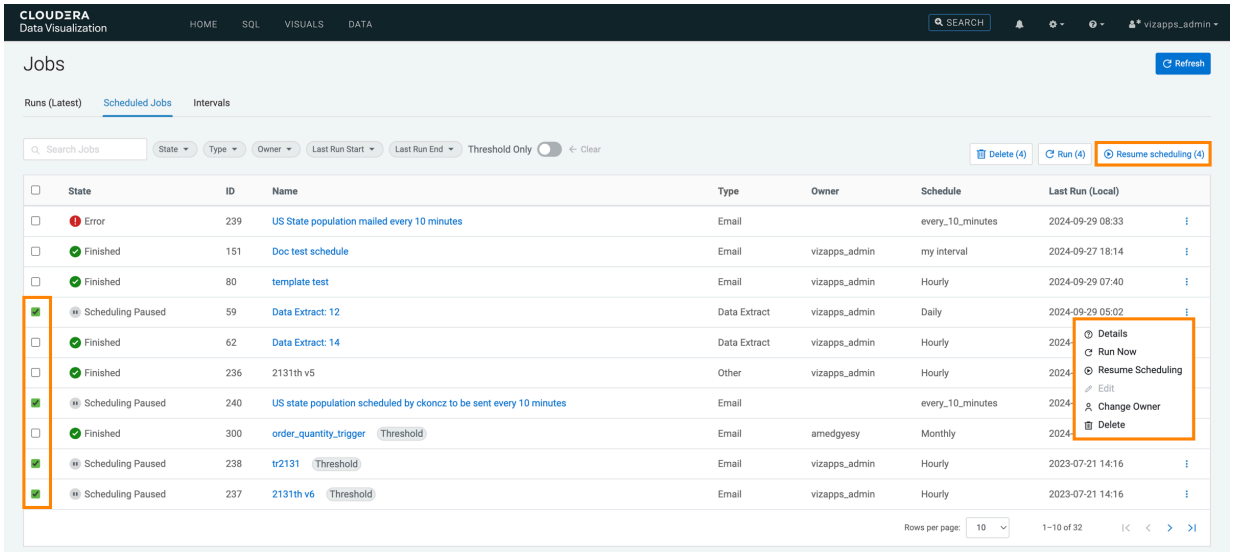

3. Click Resume Scheduling
4. Click Resume in the confirmation modal to proceed with the action.

   The modal lists the IDs and names of the jobs selected for resuming.

   A notification that the job is resumed appears briefly, the status of the job changes, and the Resume Scheduling option is replaced with Pause Scheduling in the supplemental menu.

Changing the owner of a job

Cloudera Data Visualization enables job owners and system administrators to change the owner of an existing job. This functionality is particularly useful when reassigning jobs from deleted users, as these jobs cannot run until the owner is updated.

About this task

You can perfom this task from two locations:

- On the Job Details page, which provides detailed insights into the selected job, helping you understand and manage job operations more effectively, while also offering access to various job-related actions.
- On the Scheduled Jobs page, using the supplemental menu of a job listed on this page.

Procedure

1. Click the name of the job on either the Runs (Latest) or the Scheduled Jobs tab.

   The Job Details page opens.

2. On the Job Details page, click Actions Change Owner in the top-right corner.

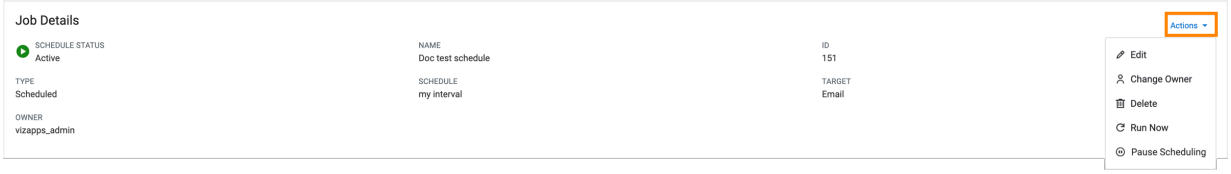

The Change Job Owner modal window appears.

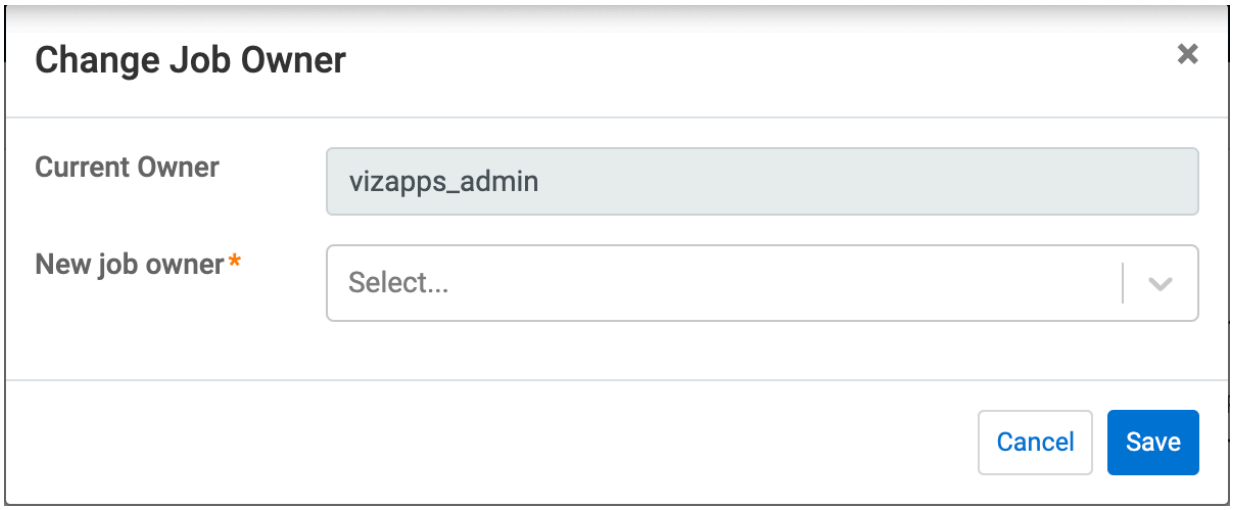

Alternatively, you can open this Change Job Owner modal from the supplemental menu of the job on the Scheduled Jobs page.

3. Select a new job owner from the list of available users.

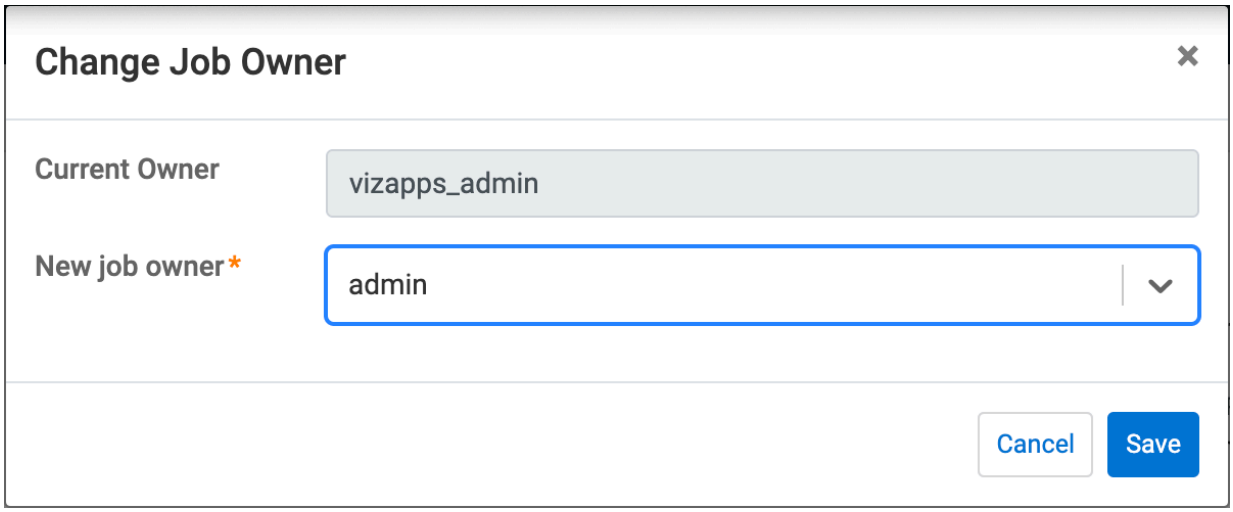

4. Click Save to confirm the change.

Note: When the job owner is changed, the ownership of the related logs is also updated accordingly.

Editing jobs

Cloudera Data Visualization allows you to modify the details of existing jobs.

About this task

You can perfom this task on the Job Details page, which provides detailed insights into the selected job. This page helps you understand and manage job operations more effectively, while also offering access to job-related actions.

Procedure

1. Click the name of the job on either the Runs (Latest) or Scheduled Jobs tab.

   The Job Details page opens.

2. Click Actions Edit in the top-right corner.

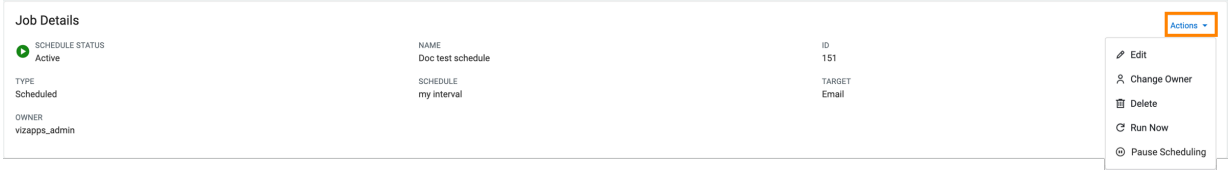

The Edit Job Details modal window appears.

Edit Job Details

ID

151

Name*

Doc test schedule

Schedule

my interval

Email Details

Visual ID

2532

To*

ifeher@cloudera.com ×

CC

Error Notification Emails To

Reply To

Subject*

[Cloudera Data Visualization]

Email Template

Default Email Template

Preview Email Template

Message

doc test email

Show Job Parameters

Page Parameters

Include page parameters Do not include page parameters

Show Page Parameters

Use Alias

Use URL alias

Attach

Embedded Image PNG PDF XLSX CSV

Cancel Save

Note:

The Edit action is not available for data extract jobs. Editing data extracts is only possible on the Data Extracts page. For more information on changing data extract details, see *Editing a data extract*.

3. Modify the fields according to your needs.
4. Once you have finalized the updates, click Save to apply the modifications.

Related Information

Editing a data extract

Deleting scheduled jobs

Cloudera Data Visualization allows you to delete existing scheduled jobs that you do not need any more.

About this task

You can perfom this task from two locations:

- On the Scheduled Jobs page
- On the Job Details page.

Important:

Deleting a scheduled job will delete the job itself along with all associated job run logs.

Procedure

1. Navigate to the Scheduled Jobs tab within the Jobs interface.
2. To delete a single job, locate the job you wish to delete and click ⋮ at the end of the job row to open the supplemental menu.

   To delete multiple jobs simultaneously, select them using the checkbox at the beginning of each row. If you want to delete all jobs, click the checkbox on the left side of the list header.

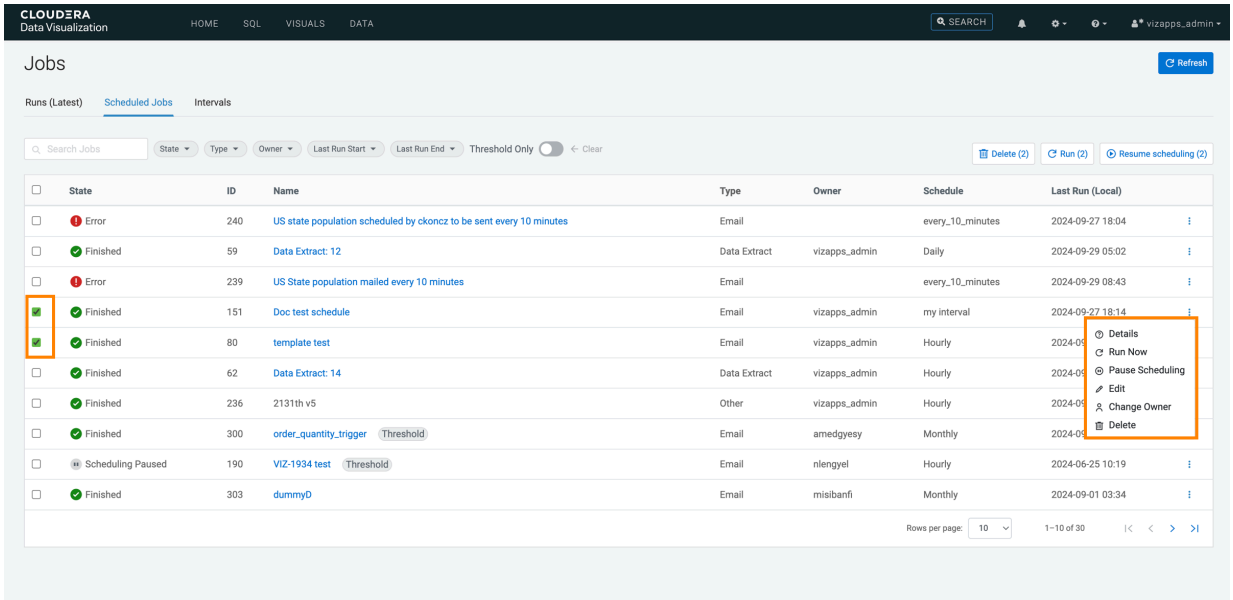

   Alternatively, you can open the Job Details page by clicking the name of a job on either the Runs (Latest) or the Scheduled Jobs tab, and on the Job Details page, click Actions Delete in the top-right corner.

3. Click Delete.
   A confirmation modal appears, listing the IDs and names of the scheduled jobs you selected for deletion.
4. Click Delete again to proceed with the action.

Creating new schedule intervals

Cloudera Data Visualization allows you to create custom schedules for your scheduled jobs.

About this task

Note: This feature is only available to users with administrative privileges.

To create a new schedule that can be used by one or more scheduled jobs, follow the below steps.

You can also create custom schedule intervals when configuring a new scheduled email. These intervals are accessible on the Intervals page, and can be reused for other jobs as well.

Procedure

1. Navigate to the Intervals tab within the Jobs interface.

2. Click Create Schedule Group.
   The Create Interval modal appears.

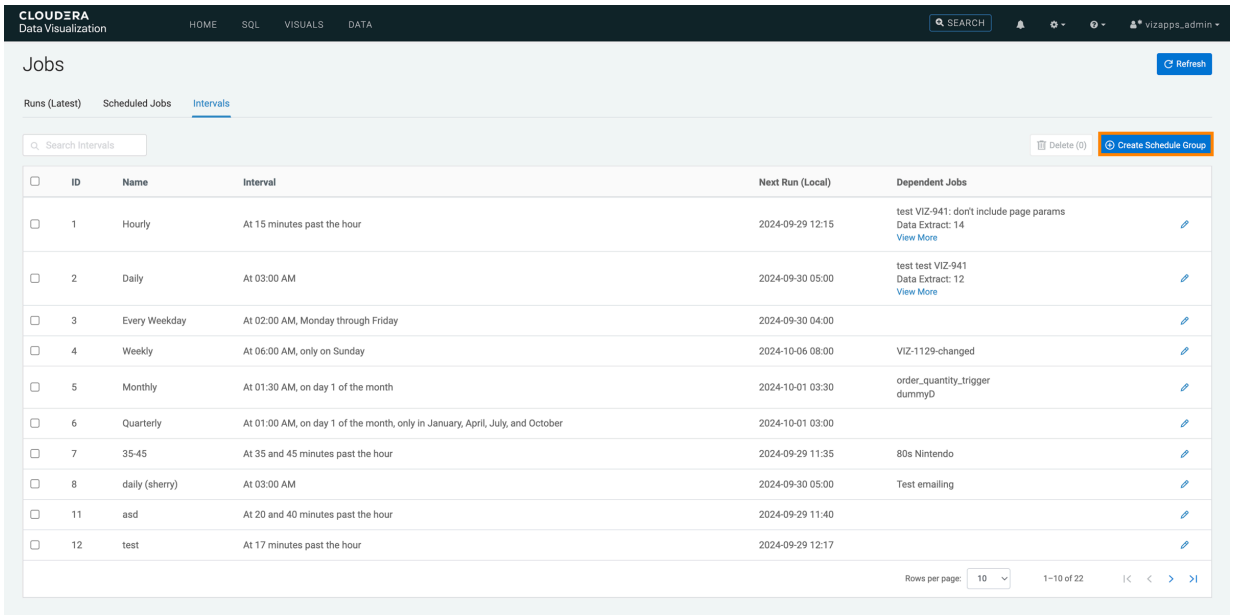

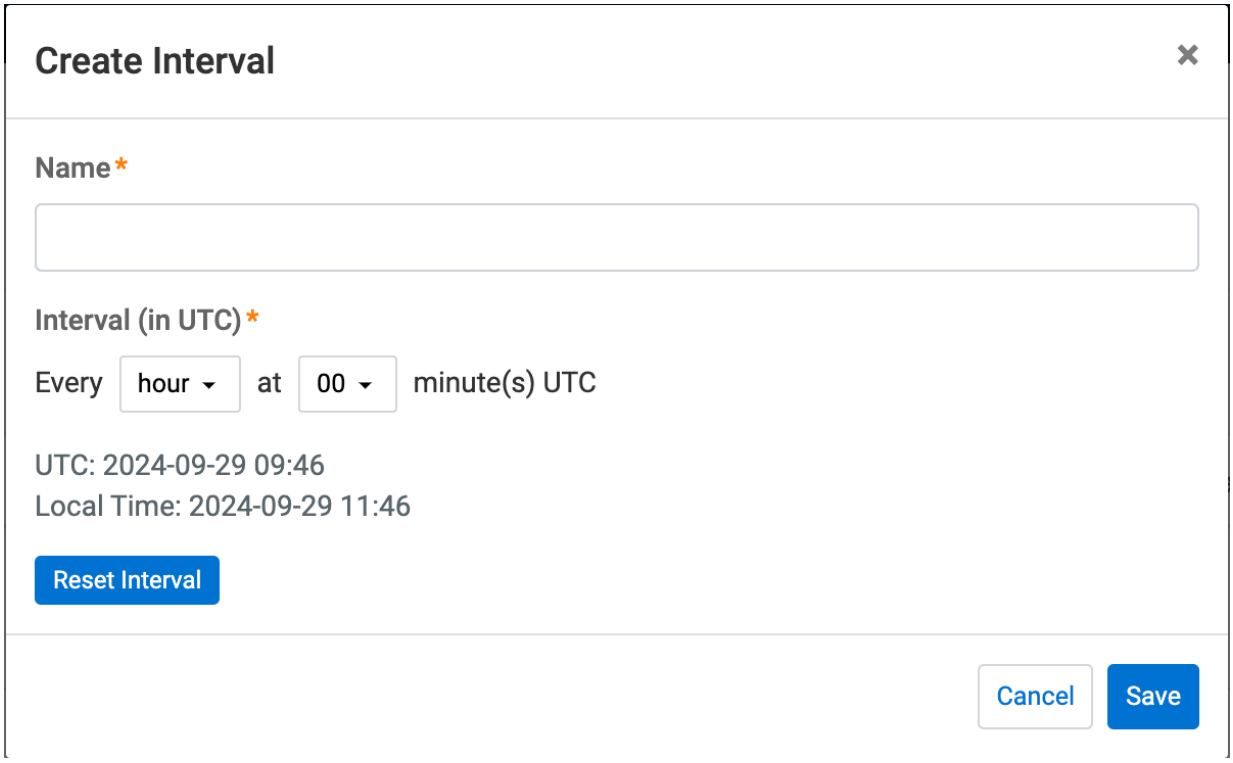

3. Specify the parameters of the schedule:
   - In the Name field, provide a descriptive name for the new schedule interval.
   - When selecting the date/time of the run, remember that the time format is in UTC. Local time format designation is provided for reference.
     - Choose the largest time component in the first selectable field (defaulted to year).
     - For the day of the month, select one or multiple days as needed.
     - Set the hour and minute components of the schedule in the third field, with multiple selections possible based on previous choices.

     Note: The number of selections depends on the granularity of the time interval specified by the first choice.
4. Click Save to create the new schedule.

   A brief success message appears on your screen, confirming the creation of the new interval. The newly created interval is now visible in the Intervals interface for future use.

Editing schedule intervals

Cloudera Data Visualization allows you to change the details of the schedule intervals that you previously defined for your scheduled jobs.

About this task

Note: This feature is only available to users with administrative privileges.

Procedure

1. Navigate to the Intervals tab within the Jobs interface.

2. Locate the job you wish to edit and click at the end of the schedule row.

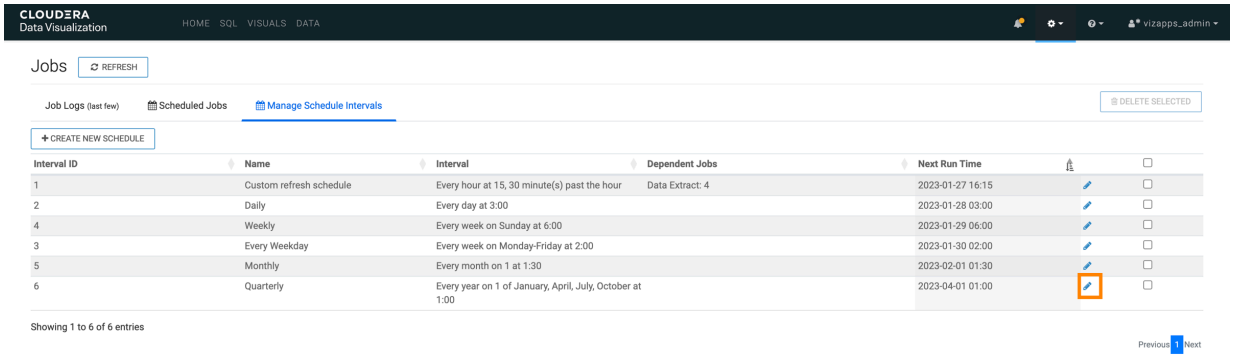

The Edit Interval modal appears.

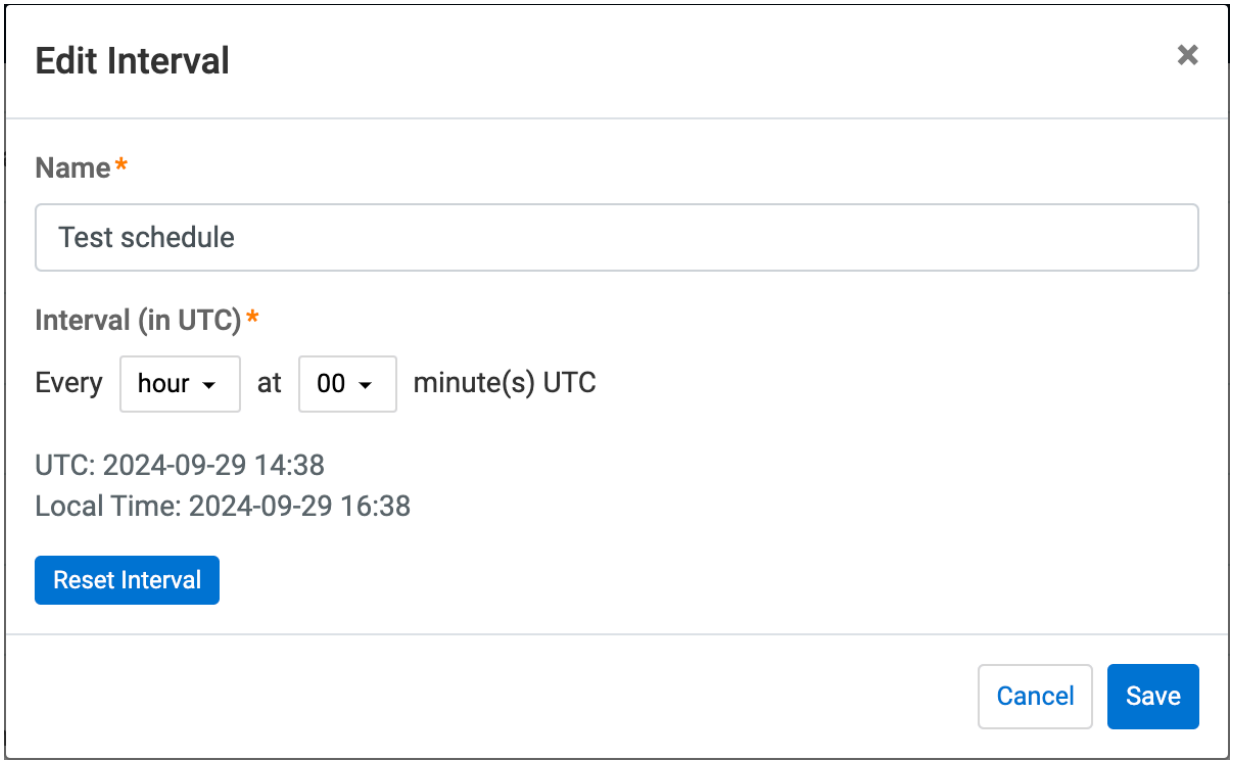

3. Adjust the name or the time selection according to your needs.

   In the below example, the time selection is adjusted to account for the 7-hour difference between UTC and local time (PDT). To achieve the intended notification times of 04:00 and 16:00 local time (PDT), adjust the hour parameter from 4 and 16 to 11 and 23, respectively.

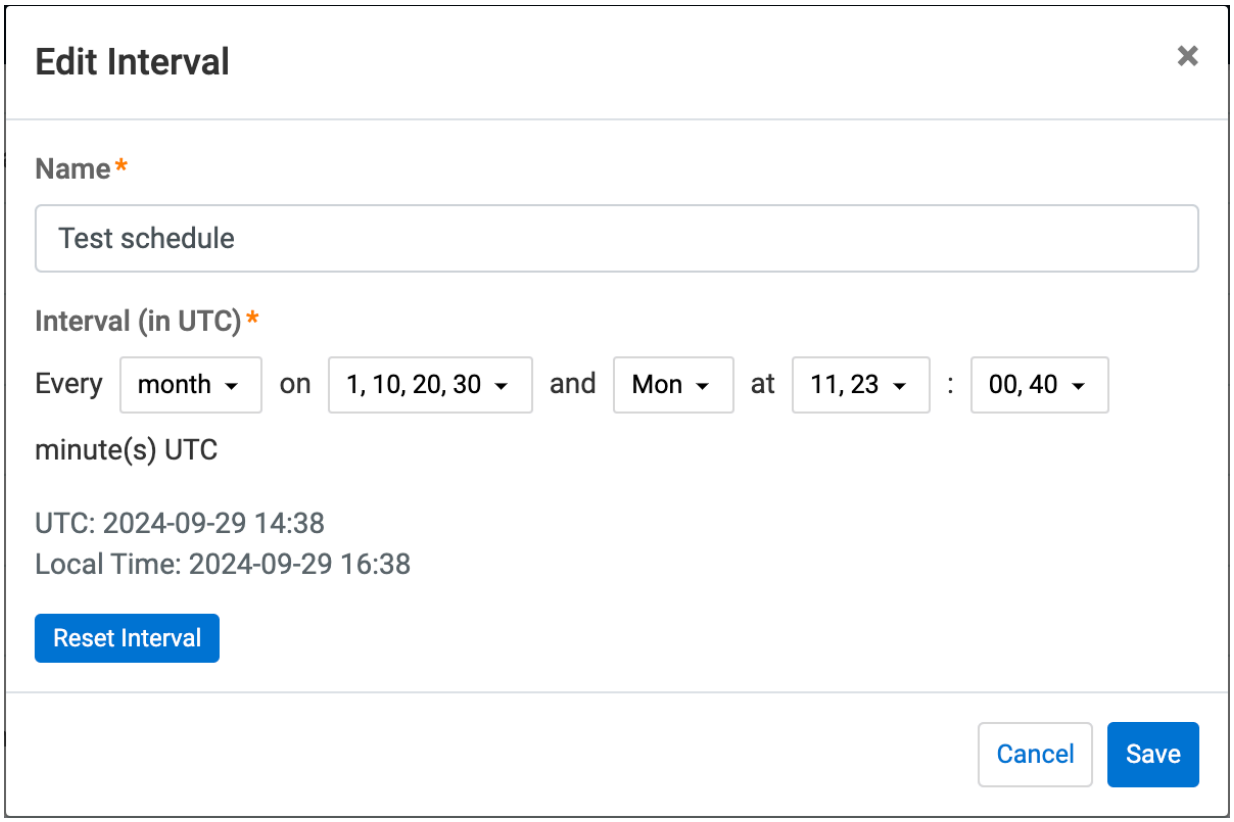

4. Click Save to apply the modifications.

Deleting schedule intervals

Cloudera Data Visualization allows you to delete previously defined job schedule intervals that are no longer needed.

About this task

Note: This feature is only available to users with administrative privileges.

Procedure

1. Navigate to the Intervals tab within the Jobs interface.

2. Select the intervals you wish to delete by using the checkbox at the beginning of each row.
   To select all intervals, use the checkbox on the left side of the list header.

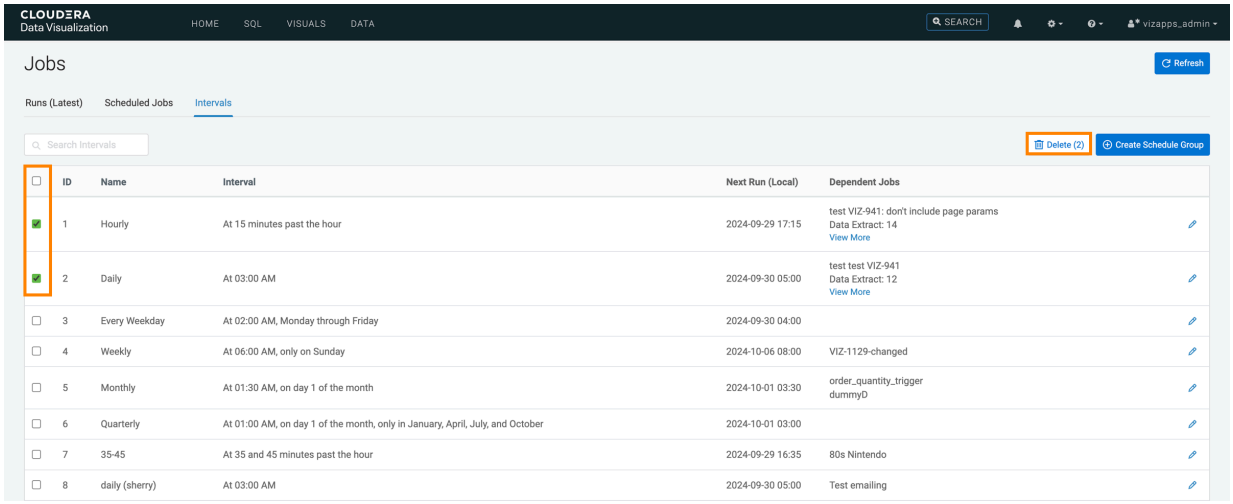

3. Click Delete in the top-right corner.

   The Delete Interval modal appears, listing the IDs and names of the intervals you selected for deletion.

4. Click Delete to proceed with the action.

Cloudera Data Visualization 7.2.9

Site Settings

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing data-related site settings 4

Managing dashboard site settings 7
- Visuals 10
- Snapshots 13
- Config 14
- Interactive Map 14

Managing email related site settings 15

Managing site configuration 19

Managing AI settings [Technical Preview] 21
- General 23
- Embeddings 23
- Completion 25

Managing security and authentication site settings 27

Managing debugging and logs site settings 28

Managing version control site settings 29

Managing help and support site settings 30
- Customizing Help on the UI 30
- Customizing Help through configuration file 32

Managing advanced settings 34
- Advanced site settings 34

Managing data-related site settings

Cloudera Data Visualization offers extensive configurability with various site-wide settings, allowing you to enable or disable specific data features and adjust related settings.

About this task

The Site Settings menu is available only to users with administrative privileges.

Procedure

1. Click the

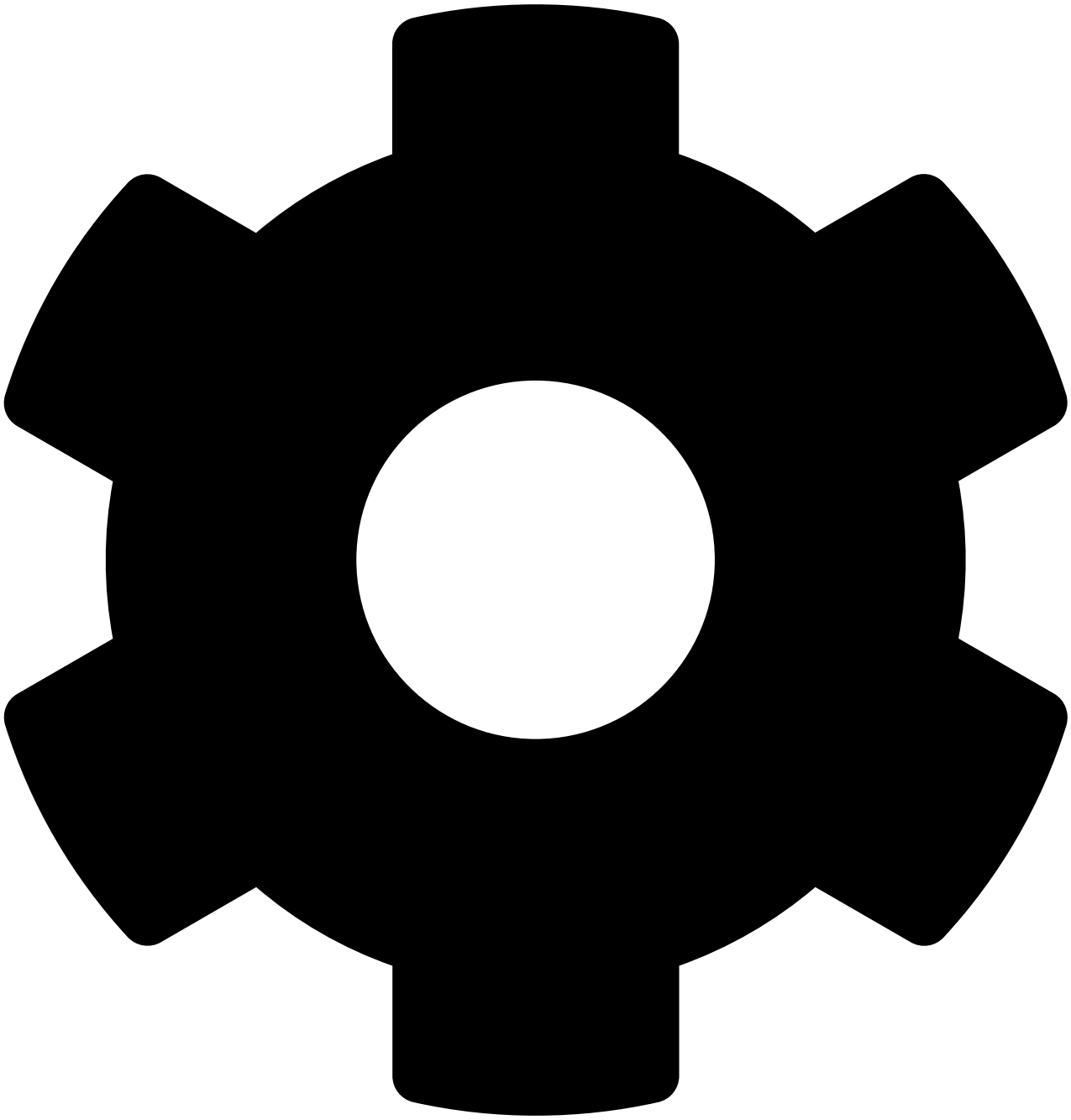

icon on the main navigation bar to open the Administration menu and select Site Settings.

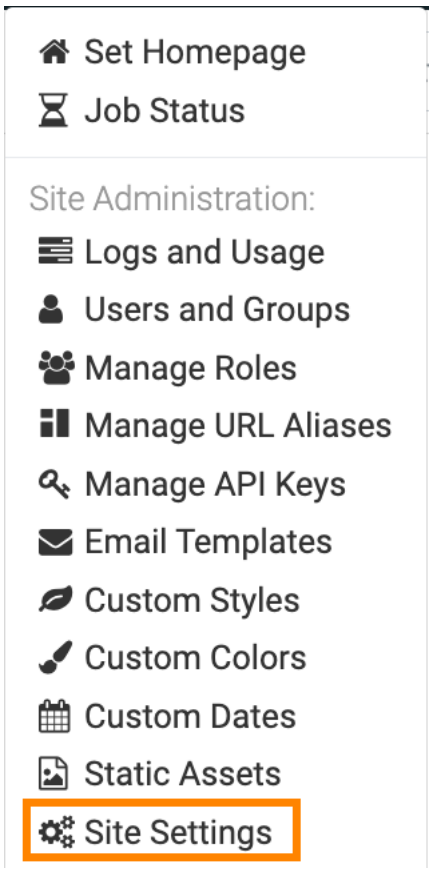

The Site Settings interface opens on the Data page.

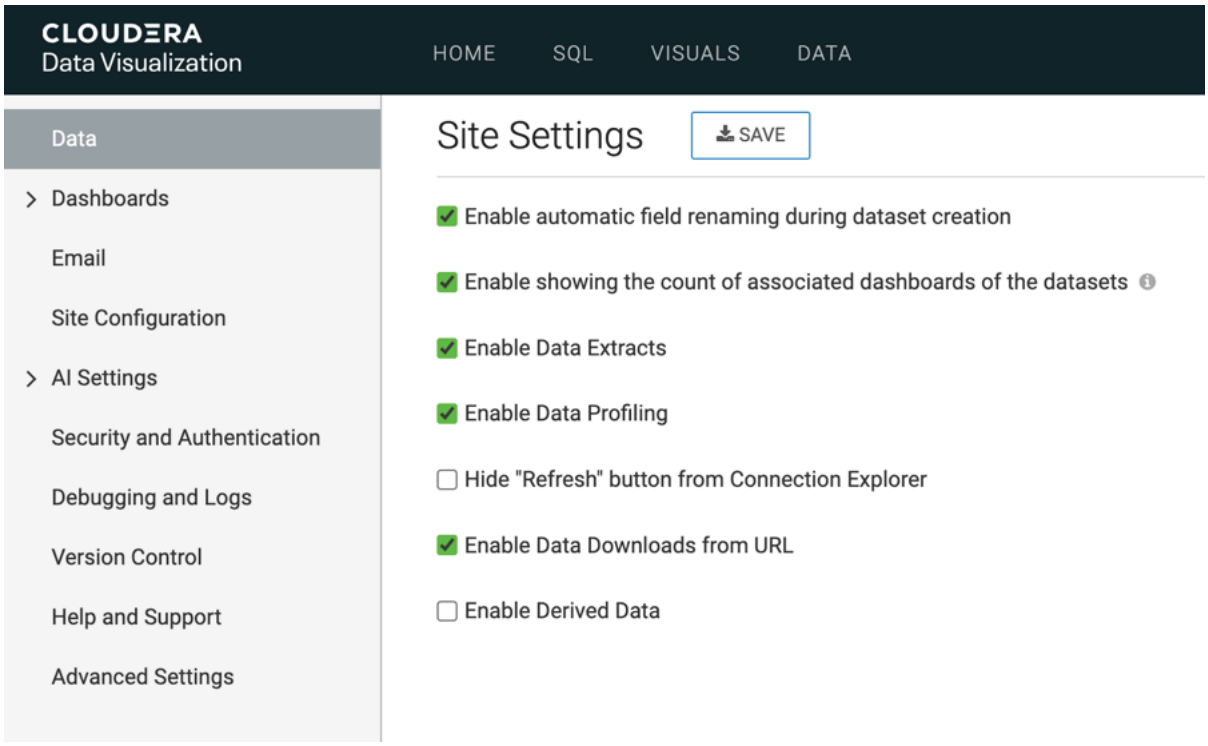

2. Set the option(s) that you want to use.

   Enable automatic field renaming during dataset creation

   This option allows you to automatically rename dataset fields for better readability. Fields with underscores are converted into multi-word fields. For example: un_region becomes Un Region.

   Enable showing the count of associated dashboards of the datasets

   This option allows you to choose to display the count of dashboards associated with each dataset.

   Tip: Disable this option if the Data page is loading slowly.

   Enable Data Extracts

   This option allows you to make the Data Extracts functionality available to users.

   Enable Data Profiling

   This option allows you to enable users to analyze and summarize data characteristics in a dataset. This feature helps to understand data quality, structure, and identify patterns and trends for more informed decision making.

   Hide "Refresh" button from Connection Explorer

   This option allows you to hide the Refresh button to have better control over data updates.

   Enable Data Downloads from URL

   This option allows you to enable users to download data directly from a specified URL address.

   Enable Derived Data

   This option allows you to enable Derived Data for high-level analyses, such as query stacking, cohort analysis, and back joins.

3. Save your changes for the updated site settings to take effect.

   Remember: You do not need to restart the Cloudera Data Visualization application.

Managing dashboard site settings

Cloudera Data Visualization is highly configurable, with many site-wide settings. You can turn specific dashboard related features on and off and adjust various settings for visuals, dashboards, and snapshots.

About this task

The Site Settings menu is only available to users with administrative privileges.

Procedure

1. Click the

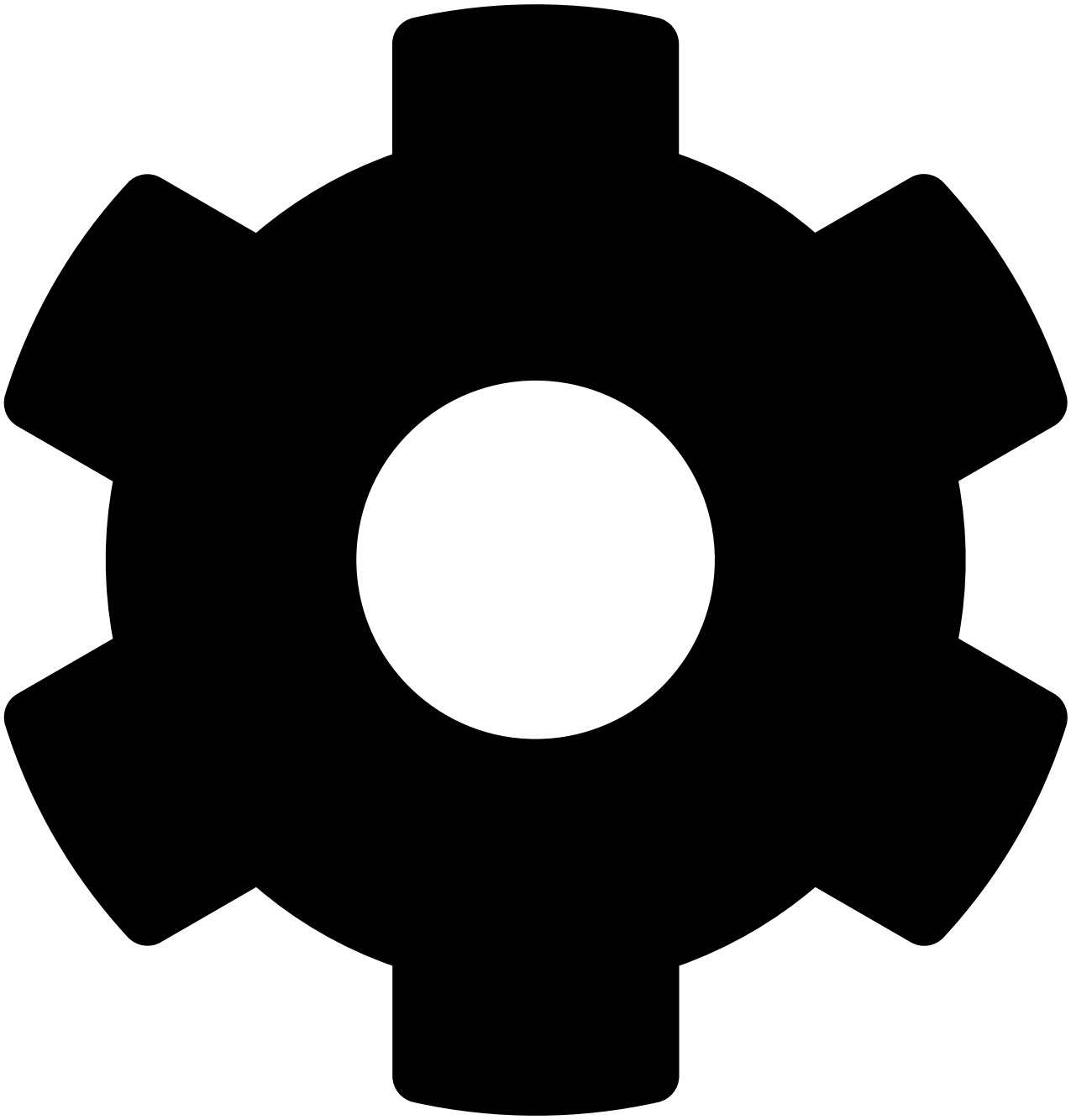

icon on the main navigation bar to open the Administration menu and select Site Settings.

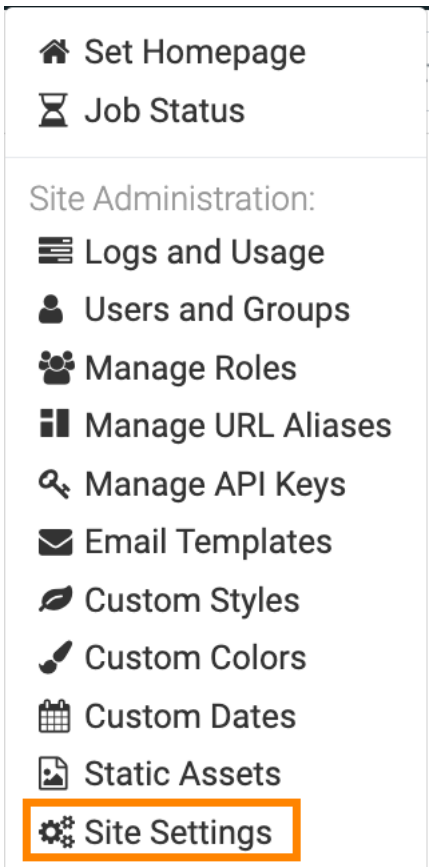

The Site Settings interface opens on the Data page.

2. Open the Dashboards menu from the left navigation and select one of the options available.

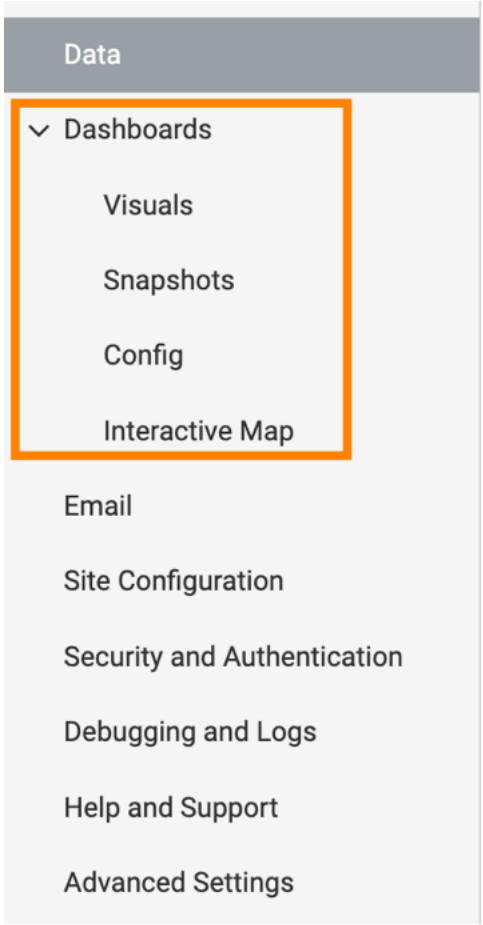

Visuals

You can enable various advanced options for visuals.

About this task

The Site Settings menu is only available to users with administrative privileges.

Remember:

Save your changes for the updated site settings to take effect, but you do not need to restart the Cloudera Data Visualization application.

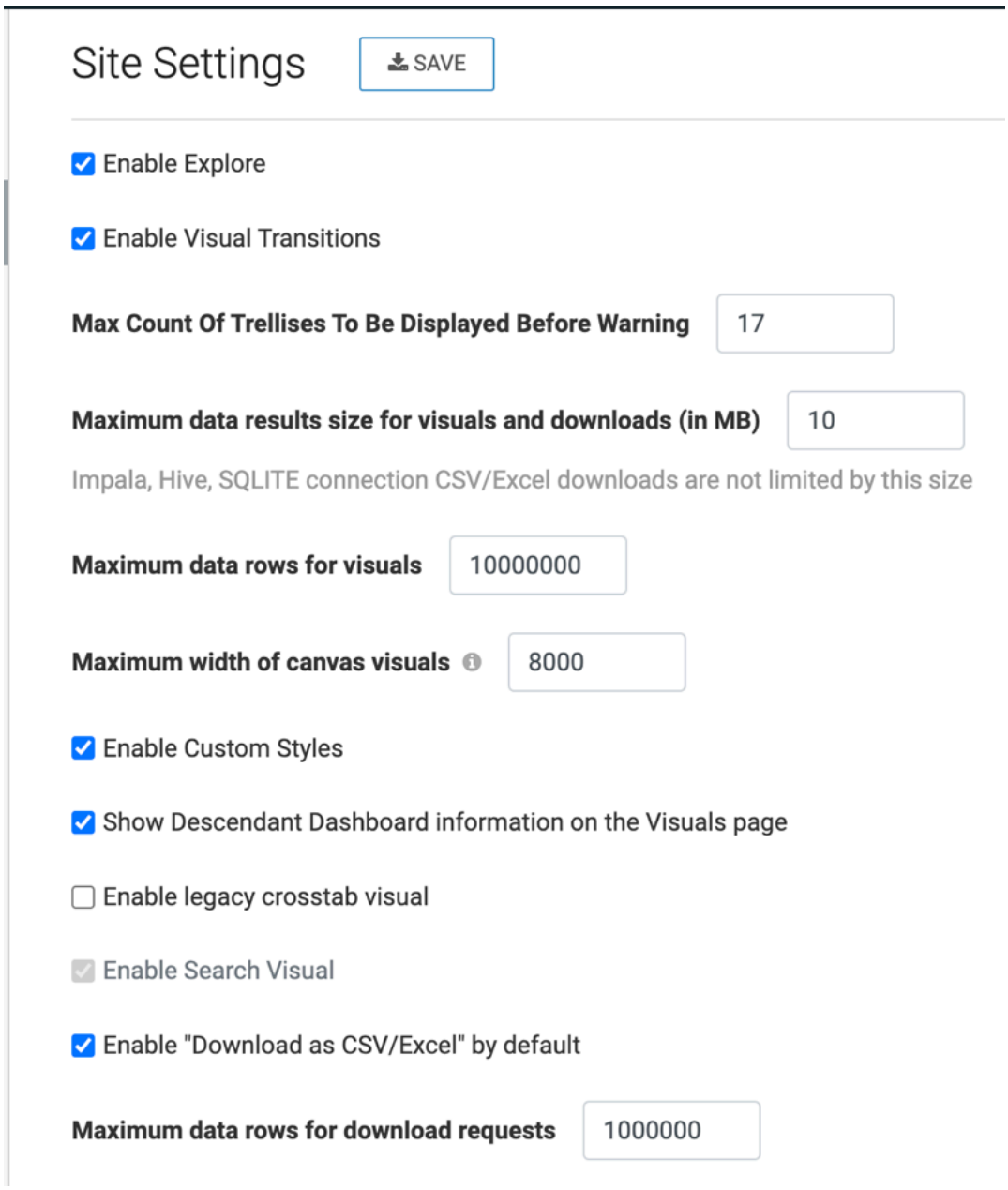

Enable explore

You can use the explore function to find the best visualization for your data request. It suggests new visual types and options based on the information you already use in existing visuals. For more information, see *Exploring visual options*.

To enable this function, select the Enable Explore option under Dashboards Visuals in the site settings.

Enable visual transitions

You can use transitions to animate visuals during loading and refresh operations. The visual types that use visual transitions include bars, lines, combo, area, scatter, grouped bar, histogram, sparklines, pie, packed bubbles, and correlation heat map.

To enable the transition function, select the Enable Visual Transitions option under Dashboards Visuals in the site settings.

Set maximum count of trellises to be displayed before warning

To adjust the maximum count of trellises, select the appropriate value for Max Count Of Trellises To Be Displayed Before Warning under Visuals.

Set maximum data results size for visuals and downloads

To adjust the maximum data results size, enter a value in the Maximum data results size for visuals and downloads (in MB) text box under Dashboards Visuals .

Important: Impala, Hive, SQLITE connection CSV/Excel downloads are not limited by this size.

Set maximum data rows for visuals

You can provide an upper limit for the number of rows rendered in a visual request. Sometimes an inexperienced or malicious user can create a visual requesting billions of rows. This may result in excessive resource consumption at the server level and the result might be too large to render in the client browser.

To avoid this scenario and provide an upper limit on the number of rows rendered, enter a value in the Maximum data rows for visuals text box under Dashboards Visuals in the site settings.

At the individual visual level, the setting in the Maximum number of rows to fetch option in the Data menu of the Visual Designer overrides this default behaviour.

At the site level, the setting in the Maximum data rows for visuals option overrides the setting in the Maximum number of rows to fetch option.

Set maximum width of canvas visuals

Cloudera recommends that you designate a reasonable maximum width for canvas visuals considering performance reasons. Overly wide visuals that display streaming data may cause the Internet browser to crash.

To set the maximum width value, enter a value in the Maximum width of canvas visuals text box under Dashboards Visuals in the site settings.

Enable custom styles

You can enable custom styling to set the appearance of visuals, dashboards, filters, and applications through CSS and JS scripting. For more information on using custom styles, see *Adding custom styles to visuals*.

Note: You must also have the Manage custom styles permission to use CSS and JS customizations.

To enable the feature, check the Enable Custom Styles option under Dashboards Visuals in the site settings.

Show descendant dashboard information

You can choose to show descendant dashboard information on the Visuals page.

To enable the feature, check the Show Descendant Dashboard information on the Visuals page option under Dashboards Visuals in the site settings.

Enable legacy crosstab visual

You can make the legacy crosstab visual type available to users.

To enable the feature, check the Enable legacy crosstab visual option under Dashboards Visuals in the site settings.

Enable the search visual

You can only add the search visual to the Visual Catalog if Search is enable under Site Configuration. Once you have enabled search, the two associated options are also automatically enabled:

Enable Search Visual

You can find it under Dashboards Visuals

Enable Search in Menu Bar

You can find it under Site Configuration.

Set CSV/Excel download as default

In Cloudera Data Visualization, you can download data in a visual in CSV or Excel format.

To make this download option default, check the Enable "Download as CSV/Excel" by default option under Dashboards Visuals .

Set maximum data rows for download requests

Sometimes an inexperienced or malicious user can download a large number of rows. This may result in excessive resource consumption at the server level.

To avoid this scenario and provide an upper limit on the number of rows downloaded in a CSV file, enter a value in the Maximum data rows for download requests text box under Dashboards Visuals .

Important: This setting overrides the setting specified by the user in the Number of rows downloaded during CSV export and Detailed Data option in the Data menu of the Visual Designer. For more information, see *Changing number of rows to download*.

Related Information

Exploring visual options

Adding custom styles to visuals

Changing number of rows to download

Snapshots

The group of snapshot settings enables you to take snapshots of your actual visuals and dashboards, which you can then download as PNG files.

About this task

The Site Settings menu is only available to users with administrative privileges.

Snapshots replace the standard thumbnails that appear in the Visuals interface with miniaturized versions of the visual or dashboard. This customization makes it very easy to locate the visuals and dashboards you want to use.

You can configure the following settings:

- Check Enable snapshots (required for thumbnails, email, and image downloads).
- Check Enable "Download as Image/PDF" by default
- Set the IP/Hostname to use to connect back to the server. You can use the default value or change it to connect to a different server.

The Site Settings menu is only available to users with administrative privileges.

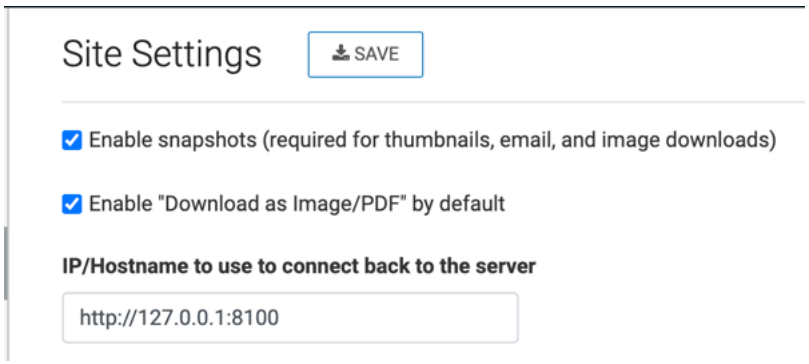

Remember:

Save your changes for the updated site settings to take effect, but you do not need to restart the CDV application.

Config

You can perform the following dashboard configuration actions:

About this task

The Site Settings menu is only available to users with administrative privileges.

Set default width of dashboards

To specify a width for dashboards, enter the measurement for Default Dashboard Width in pixels.

Note: If you clear the field, all dashboards automatically resize to the full width of the browser window.

Set warning that loading dashboards takes too long in view mode

To set the warning, check the Show "This dashboard is still loading data" in view mode option under Dashboards Config in the site settings.

Remember:

Save your changes for the updated site settings to take effect, but you do not need to restart the Cloudera Data Visualization application.

Interactive Map

About this task

The Site Settings menu is only available to users with administrative privileges.

Enable API keys for interactive map visuals

Your organization must obtain its own credentials for Google Maps, either Google API Id or Google Client Id. To obtain Google Maps keys, see Google Maps APIs.

For the Mapbox interface, your organization must get a Mapbox Token and the Mapbox Style ID, specified for each visual in the Settings menu. Cloudera Data Visualization already has a default mapbox token. To obtain new Mapbox tokens and Style IDs, see Mapbox.

Specify atlas mapbox server

You can use a Mapbox server or specify an Atlas Mapbox server at the site level to run Mapbox APIs on your own infrastructure, whether on your own private cloud or on-premises.

To specify an Atlas Mapbox server at the site level, Enter the IP address or hostname of the Atlas Mapbox server in the IP/Hostname of Atlas Mapbox Server (optional) text box.

Note: In an interactive map visual, the Atlas Mapbox server for visual setting under the Settings > Map Server menu, overrides the IP/Hostname of Atlas Mapbox Server (optional) setting defined under Site Settings.

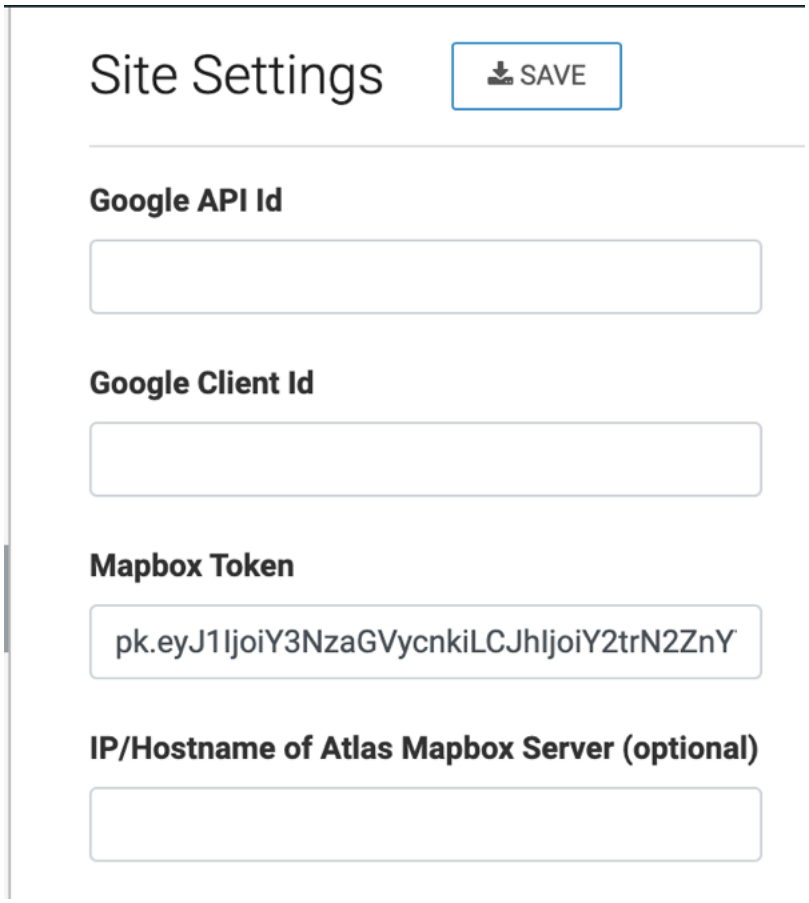

Remember:

Save your changes for the updated site settings to take effect, but you do not need to restart the Cloudera Data Visualization application.

Managing email related site settings

Cloudera Data Visualization is highly configurable, with many site-wide settings. You can configure email related details and customize additional email settings to enable users to share visuals through email.

Procedure

1. Click the

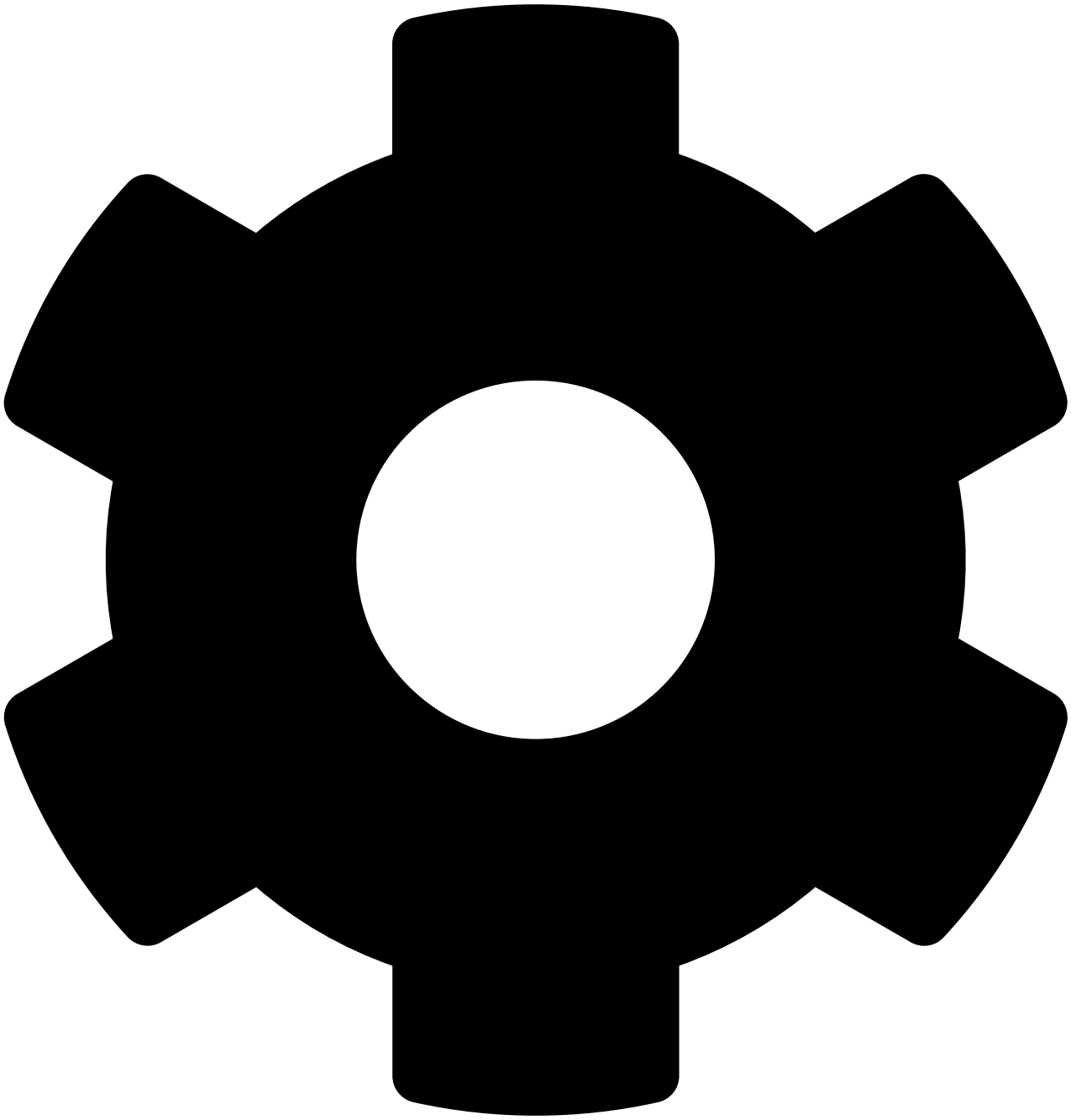

icon on the main navigation bar to open the Administration menu and select Site Settings.

Note: The Site Settings menu is only available to users with administrative privileges.

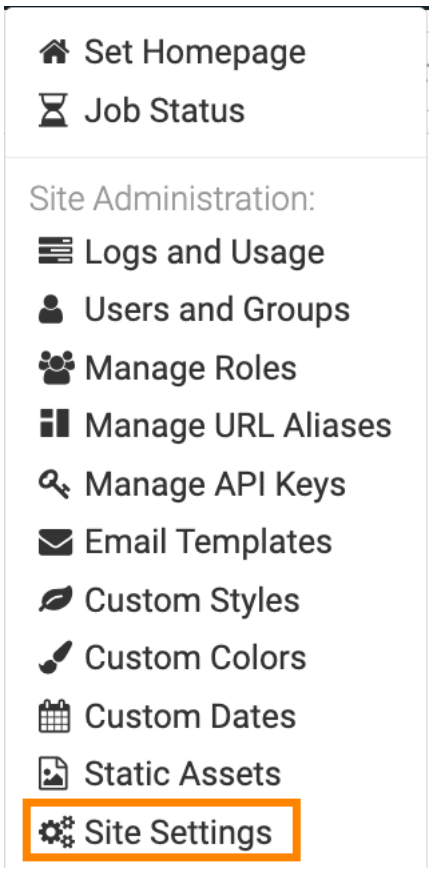

The Site Settings interface opens on the Data menu.

2. Open the Email page from the left navigation.
3. Check the Permit emails option to allow users to email visuals and receive job and extract refresh status messages.
4. Enter the Email Address and Password.

   Optionally, you can also enter your Username. If you do not enter anything in this text box, your email address is used as the default value.

   These details have to belong to a valid account, and all future emails originate from this account.
5. Enter the SMTP Host.
6. Enter the Port.
7. For Email protocol, select the appropriate option based on provider: SSL, TLS, or None.

8. Enter the Default email subject prefix.
   The default prefix value is [Cloudera Data Visualization].

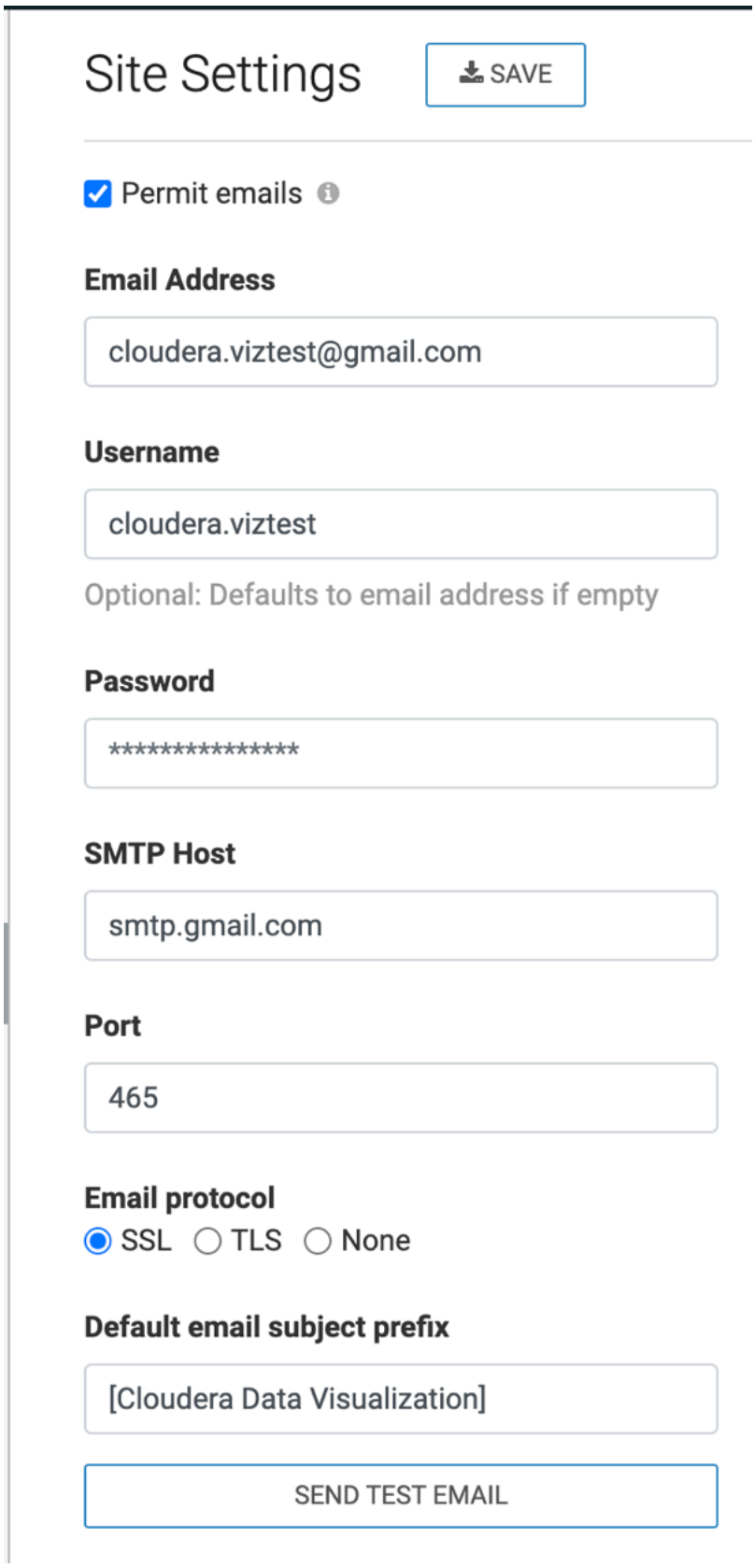

9. Test the email functionality by clicking SEND TEST EMAIL.

   If the settings are correct, you should see a success message:

10. Save your changes for the updated site settings to take effect.

    Remember: You do not need to restart the Cloudera Data Visualization application.

    Note: When using Google Email (Gmail), you may be using 2-factor authentication, which requires an app password. If you are **not** using 2-factor authentication, you must enable access for less secure apps. For more information, review the following external documentation:

    - Less secure apps and your Google Account
    - How to create app passwords

Managing site configuration

Cloudera Data Visualization allows you to define site configuration globally. Site settings configuration is only available to users with administrative privileges.

Notifications

You can mute notifications on the Cloudera Data Visualization site.

To enable the feature, check the Mute Notifications option under Site Configuration and save your changes for the updated site settings to take effect. This prevents users from receiving notifications from Cloudera Data Visualization.

Speech detection (Technical Preview)

You can activate speech detection on the Cloudera Data Visualization site.

To enable the feature, check the Enable Speech Detection on site (Tech Preview) option under Site Configuration and save your changes for the updated site settings to take effect.

Note: This feature is in technical preview.

Search

To enable search in Cloudera Data Visualization, check the Enable Search option under Site Configuration and save your changes for the updated site settings to take effect.

Once search is enabled, the two associated options are also automatically enabled:

Enable Search Visual

You can find it under Dashboards Visuals

Enable Search in Menu Bar

You can find it under Site Configuration

If you do not need the Search visual or Search in the Menu Bar, deselect the corresponding check box.

Scheduled jobs

You can schedule a number of jobs, either at fixed times or as a response to triggering conditions.

To enable the feature, check the Enable scheduled jobs option under Site Configuration.

> **Remember:**
>
> You must save your changes and restart the Cloudera Data Visualization application for the changed site setting to take effect.

Usage analytics (Technical Preview)

If you want to use a dynamic dashboard with visualizations generated from the audit logs displayed on the Activity Log tab, you can add Cloudera Data Visualization Usage Analytics to the Logs and Usage interface.

To enable this feature, check the Enable Cloudera Data Visualization Usage Analytics (Tech Preview) option on the Site Configuration page.

Additionally, you have the option to set parameters for the handling of usage data. You can specify the maixum number of days for usage data to remain valid, and you can also delete all usage data.

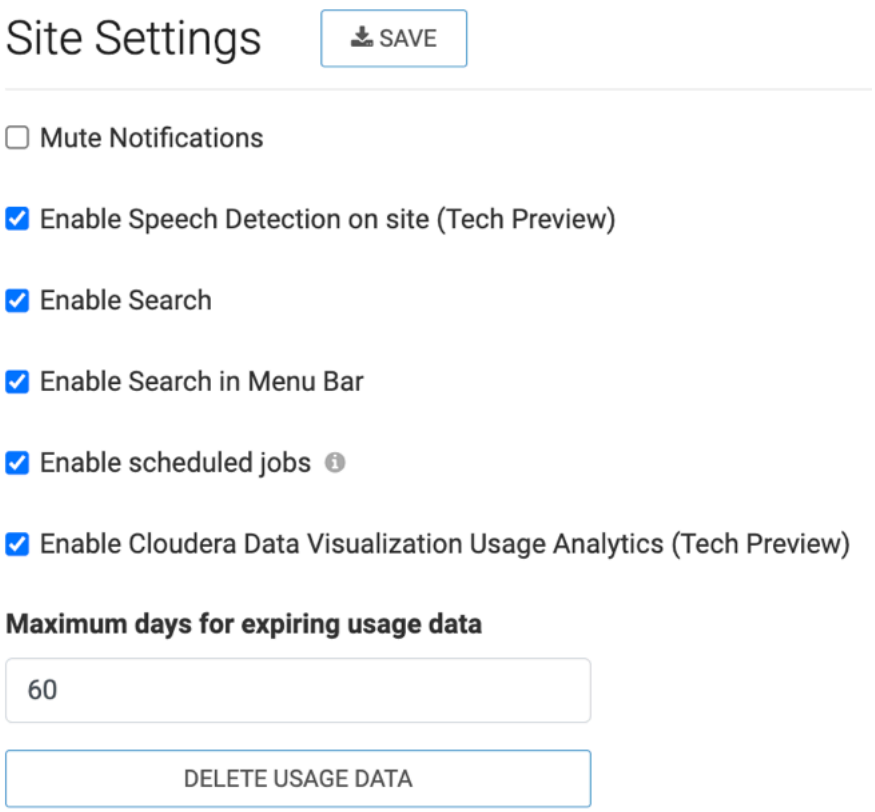

> **Remember:**
>
> You must save your changes and restart the Cloudera Data Visualization application for the changed site setting to take effect.

Managing AI settings [Technical Preview]

Cloudera Data Visualization offers extensive site-wide configuration options, enabling users with administrative privileges to manage various settings crucial for organizational workflows. You can enable the AI visual, select the AI engine, and specify additional settings.

About this task

The AI Settings option is included in the Site Settings menu, which is only available to users with administrative privileges.

Procedure

1. Click

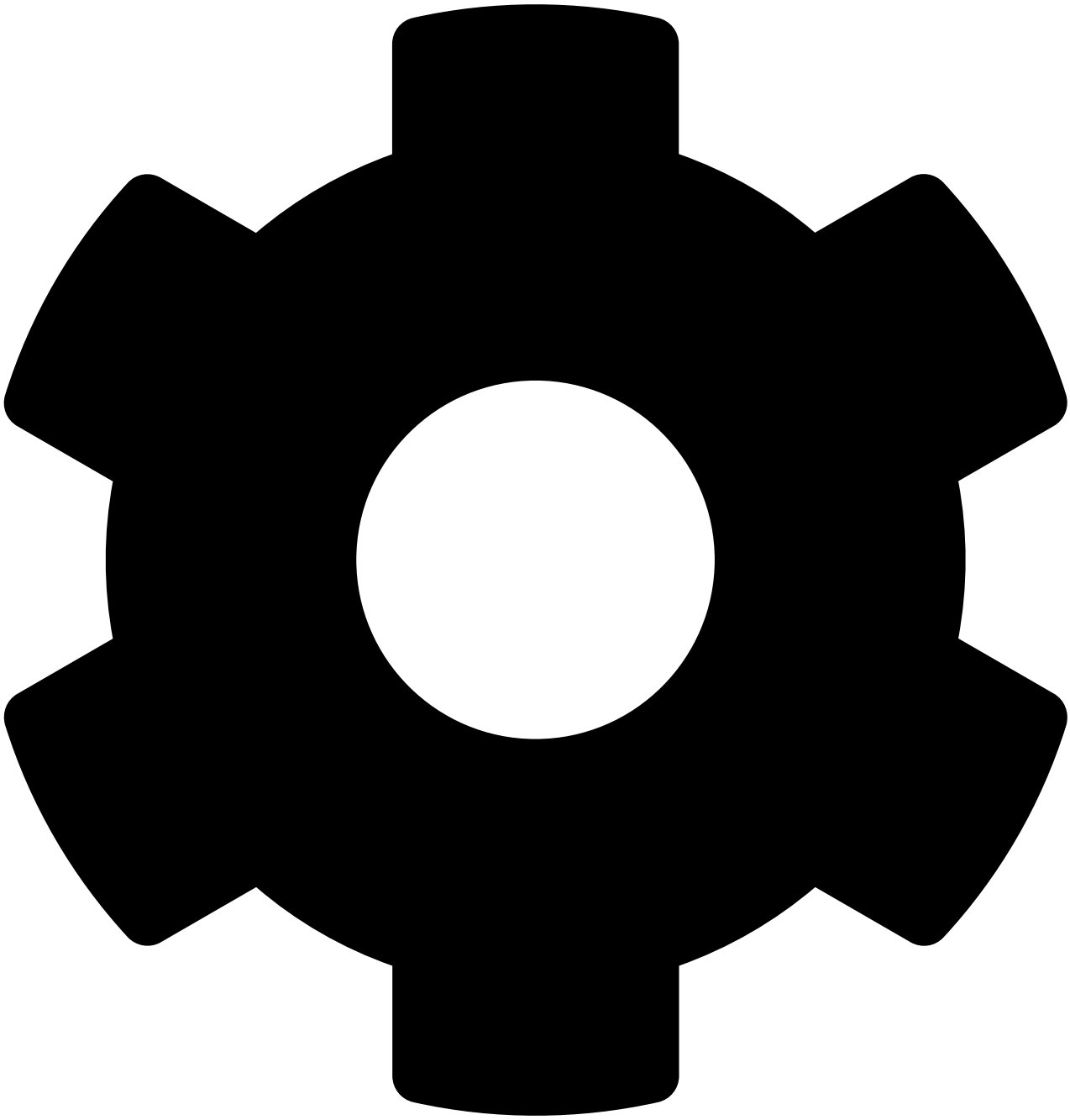

on the main navigation bar to open the Administration menu and select Site Settings.

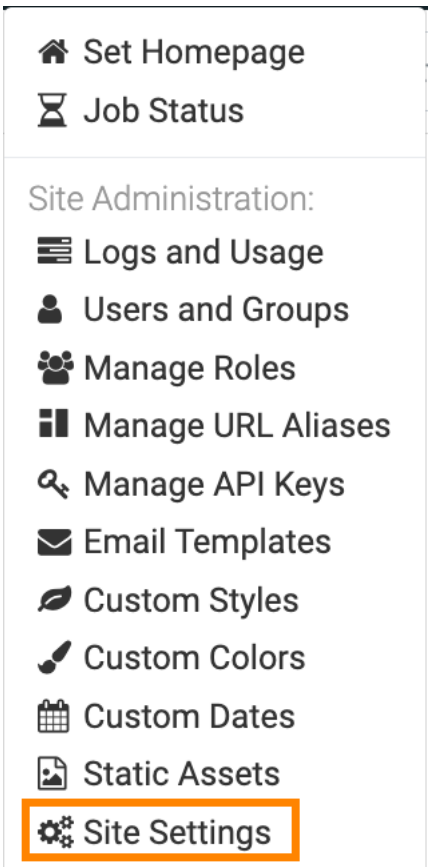

The Site Settings page opens on the Data menu.

2. Open the AI Settings menu from the left navigation and choose from the available options.

General

In this section, you can enable the AI visual and the redaction of AI visual logs.

Note: The AI visual is available as a technical preview feature. Cloudera encourages you to explore it in non-production environments and provide feedback on your experiences.

Before enabling this feature, review the legal terms and conditions of using the AI visual, displayed in the Info tooltip. When you enable the feature for the first time, a modal window prompts you to accept these terms and conditions. The system records the date and the username of the person who accepts the terms. To ensure compliance, users are required to confirm acceptance of the terms each time they enable the AI features, as the user enabling the feature may differ from the original user who enabled the feature before.

Important: If this feature is not enabled, you are not able to view AI visuals on dashboards created by other users.

Additionally, you can also enable the redaction of AI visual logs. If enabled, audit logs are redacted to omit information transmitted by the AI visual.

Embeddings

In this section, you can manage your embedding profiles. You can create new profiles, view the list of existing ones, and perform actions such as cloning, editing, or deleting existing profiles, and setting a new default profile.

Creating a new embeddings profile

1. Click CREATE PROFILE.

   The Create Profile modal window opens.
2. Provide a name for the profile.
3. Select the AI engine you want to use for the embeddings.

   You have the following options available: Cloudera AI Hosted MiniLM, Cloudera AI Inference, OpenAI, OpenAI Azure, and Other
4. Specify the settings for the selected AI engine.

   For Cloudera AI Hosted MiniLM

   - **Access Key**: Required to access the external third-party AI service provider
   - **Authorization API Key**: Required to access the external third-party AI service provider
   - **Service URL**: URL for the embedding service used by the AI engine
   - **Service input parameter name**: Name for the embedding service input
   - **Service response parameter name**: Name for the embedding service response parameter

   For Cloudera AI Inference

   - **API Key**: Required to access the external third-party AI service provider

     Ensure you review the legal terms and conditions provided in the Info tooltip before proceeding.
   - **Service URL**: URL for the embedding service used by the AI engine
   - **Model**: Used for the embeddings

     Note: If the embedding model requires the input_type parameter, add the -query suffix to the model name. For more information, see the NVIDIA documentation.

   For OpenAI

   - **API Key**: Required to access the external third-party AI service provider

     Ensure you review the legal terms and conditions provided in the Info tooltip before proceeding.
   - **Model**: Used for the embeddings

   For OpenAI Azure

   - **API Key**: Required to access the external third-party AI service provider

     Ensure you review the legal terms and conditions provided in the Info tooltip before proceeding.
   - **Service URL**: URL for the embedding service used by the AI engine

   For Other

   - **API Key**: Required to access the external third-party AI service provider

     Ensure you review the legal terms and conditions provided in the Info tooltip before proceeding.
   - **Service URL**: URL for the embedding service used by the AI engine
   - **AI engine settings**: Use this textbox to provide your custom request and response transformer functions in valid Python.
5. [Optional] Click TEST to validate the model using the specified settings.
6. Click SAVE.

Cloning an existing embeddings profile

Alternatively, you can clone an existing embeddings profile to create a new profile instance.

1. Click to open the Create Profile modal window. All details are pre-filled from the original profile, except for the Name field.
2. Enter a new name for the profile.
3. Click SAVE.

Managing embeddings profiles

- **Setting a default profile**: The current default profile that will be used for new AI visuals is marked with the Default label. To change the default, hover over a non-default profile and click the Set as default option.
- Deleting a profile:
  1. Click on the profile row.
  2. Click YES to confirm the deletion.
- **Editing a profile**:
  1. Click on the profile row to open the Edit Profile modal window.
  2. Make the necessary changes to the profile.
  3. [Optional] Click TEST to validate the model using the new settings.
  4. Click SAVE.

Completion

In this section, you can manage completion profiles. You can create new profiles, view the list of existing ones, and perform actions such as copying, editing, deleting existing profiles, or setting a new default profile.

Creating a new completion profile

1. Click CREATE PROFILE.
2. Provide a name for the profile.
3. Select the AI engine you want to use for the completion.

   You have the following options available: Cloudera AI Hosted Llama, Cloudera AI Inference, OpenAI, OpenAI Azure, and Other

4. Specify the settings for the selected AI engine.

For Cloudera AI Hosted Llama

- **Access Key**: Required to access the external third-party AI service provider

  Ensure you review the legal terms and conditions provided in the Info tooltip before proceeding.
- **Authorization API Key**: Required to access the external third-party AI service provider

  Ensure you review the legal terms and conditions provided in the Info tooltip before proceeding.
- **Service URL**: URL for the completion service used by the AI engine
- **Service input parameter name**: Name for the completion service input
- **Service response parameter name**: Name for the completion service response parameter
- **Extra arguments**: Any additional arguments for the completion query

For Cloudera AI Inference

- **API Key**: Required to access the external third-party AI service provider

  Ensure you review the legal terms and conditions provided in the Info tooltip before proceeding.
- **Service URL**: URL for the completion service used by the AI engine
- **Model**: Used for the completion
- **Temperature**: Controls the randomness of the output on a scale from 0 to 2. Lower values produce more focused and deterministic responses, while higher values introduce greater randomness and creativity. The default setting is 1.

For OpenAI

- **API Key**: Required to access the external third-party AI service provider

  Ensure you review the legal terms and conditions provided in the Info tooltip before proceeding.
- **Model**: Used for the completion
- **Temperature**: Controls the randomness of the output on a scale from 0 to 2. Lower values produce more focused and deterministic responses, while higher values introduce greater randomness and creativity. The default setting is 1.

For OpenAI Azure

- **API Key**: Required to access the external third-party AI service provider

  Ensure you review the legal terms and conditions provided in the Info tooltip before proceeding.
- **Service URL**: URL for the completion service used by the AI engine

For Other

- **API Key**: Required to access the external third-party AI service provider

  Ensure you review the legal terms and conditions provided in the Info tooltip before proceeding.
- **Service URL**: URL for the completion service used by the AI engine
- **AI engine settings**: Use this textbox to provide your custom request and response transformer functions in valid Python.

5. [Optional] Click TEST to validate the model using the specified settings.
6. Click SAVE.

Cloning an existing completion profile

Alternatively, you can clone an existing completion profile to create a new profile instance.

1. Click to open the Create Profile modal window. All details are pre-filled from the original profile, except for the Name field.

2. Enter a new name for the profile.
3. Click SAVE.

Managing completion profiles

- Setting a default profile: The current default profile that will be used for new AI visuals is marked with the Default label. To change the default, hover over a non-default profile and click the Set as default option.
- Deleting a profile:
  1. Click on the profile row.
  2. Click YES to confirm the deletion.
- Editing a profile:
  1. Click on the profile row to open the Edit Profile modal window.
  2. Make the necessary changes to the profile.
  3. [Optional] Click TEST to validate the model using the new settings.
  4. Click SAVE.

Managing security and authentication site settings

Cloudera Data Visualization allows you to adjust the security and authentication settings. Security configuration is only available to users with administrative privileges.

About this task

> **Remember:** Save your changes for the updated security site settings to take effect, but there is no need to restart the Cloudera Data Visualization application.

Enable local user creation

You can enable the option to create local users in a Cloudera Data Visualization instance.

To enable the feature, check the Enable Local User Creation option under Security and Authentication.

Set SSO for authentication

Check Permit authentication via the REMOTE_USER http header to enable user authentication with the login credentials and log users in automatically.

- In CML, this SSO option is enabled by default.

Set password requirements

For password control, you can specify the following requirements:

- **Minimum password length**

  Specify the minimum number of characters for a password. The default value is 5.
- **Require that password contains one or more numbers**

  Select this option to enforce the rule that the password must contain a numerical value between 0 and 9.
- **Require that password contains one or more symbols**

  Select this option to enforce the rule that the password must contain a symbol. Symbols include all characters on the keyboard other than letters or numbers.

Set default group for new users

To define the default group membership for new users, enter the group name in the Default group membership for new users text box under Security and Authentication.

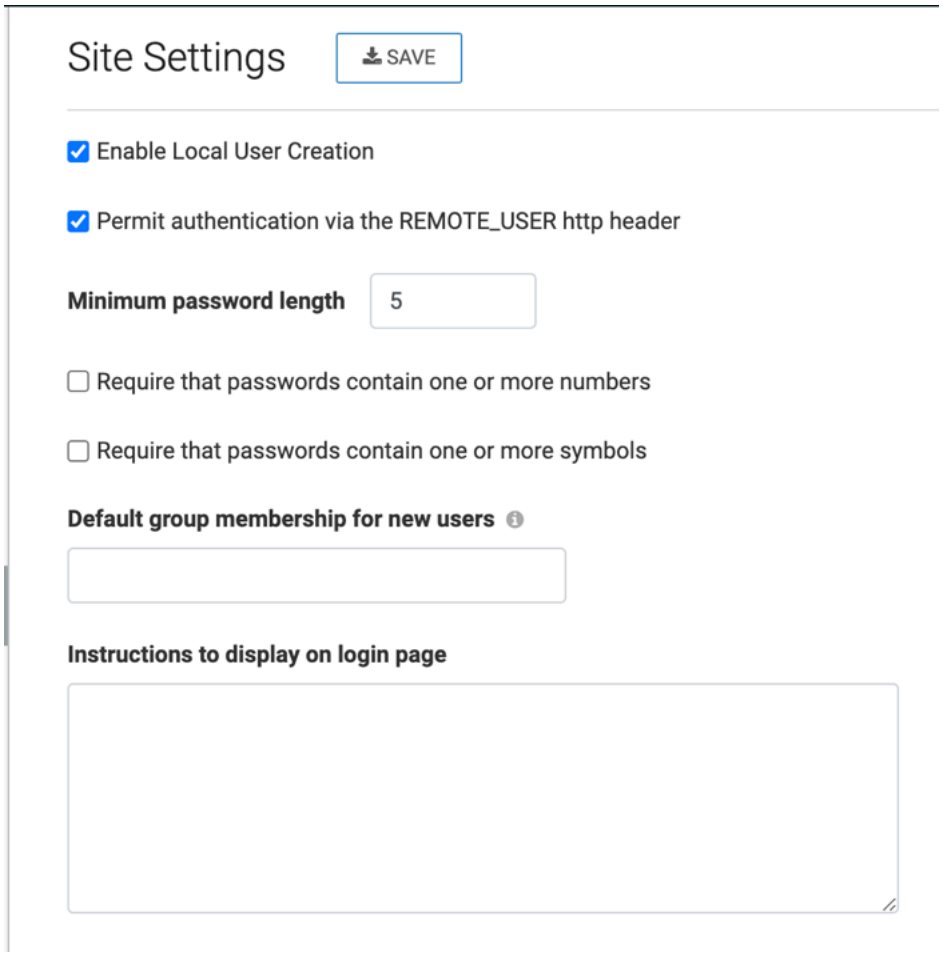

Set instructions for login page

To set what instructions are displayed on the Cloudera Data Visualization login page, enter the information in the Instructions to display on login page text box under Security and Authentication.

Managing debugging and logs site settings

In Cloudera Data Visualization, you configure job log retention on the system and enable various debugging options. This configuration is only available to users with administrative privileges.

You can set the following options:

Server Settings tab

To enable server settings, check the Enable the "Server Settings" tab within the "Activity Log" screen option.

Verbose Error Reporting

Check the Verbose Error Reporting. Display Javascript page errors on screen in app view/edit pages option.

Show Usage Info

Check the Enable the "Show Usage Info" and Ctrl-Shift-G shortcut to view page profiles within all pages option.

Page profiles views within dashboard/app view pages

Check the Additionally enable the Ctrl-Shift-G shortcut to view page profiles within dashboard/app view pages option.

Number of log entries, their retention days, and time interval for log deletion check

Max Entries

Maximum number of job log entries kept after bulk process

Check Interval

Time interval in days for log deletion checks

Note:

No restart required, but the new value only takes effect after the previous interval has elapsed. A restart is required if you need the changes to take effect immediately.

Max Retention Days

Maximum retention time in days for keeping old job logs

Remember: Save your changes for the updated site settings to take effect, but you do not need to restart the Cloudera Data Visualization application.

Managing version control site settings

Cloudera Data Visualization offers extensive site-wide configuration options, allowing users with administrative privileges to manage various settings crucial for organizational workflows.

About this task

The Site Settings menu is only available to users with admin user permissions or site setting system privilege.

Note: Dataset versioning is currently available as a technical preview in Cloudera Data Visualization.

You can set the following options:

Enable Version Control (Tech Preview)

By using this option, you can activate application-wide version control within Cloudera Data Visualization.

Note: This setting is required before enabling dataset-specific version control.

Enable Dataset Version Control (Tech Preview)

Once general version control is enabled, you can enable version control specifically for datasets. This allows you to:

- Track changes and revisions across dataset versions.
- Enable features like rollback to previous dataset versions.

Unnamed Dataset Version Deletion Interval

This setting defines how frequently an automatic cleanup script runs to delete old, unnamed dataset versions. You can set the number of days between checks for deletion of unnamed versions. The minimum interval is 1 day and the default setting is 7 days.

Unnamed Dataset Version Maximum Retention Days

This setting defines the maximum lifespan for unnamed dataset versions before they are automatically deleted. You can set the retention period, which is the maximum number of days a dataset version can exist without a name. The minimum retention period is 1 day, and the default setting is 30 days.

For more information about unnamed dataset versions, see *Tracking dataset versions*.

Related Information

Tracking dataset versions

Managing help and support site settings

As an administrator, you can customize the default settings of the Help menu in Cloudera Data Visualization. You can edit or change the link for Documentation and Support directly on the UI, or in the settings.py file.

Customizing Help on the UI

You can change the default settings of the Help menu in Cloudera Data Visualization. You can customize the links for Documentation and Support on the Site Settings interface.

About this task

In Site Settings Help and Support , you can change the following:

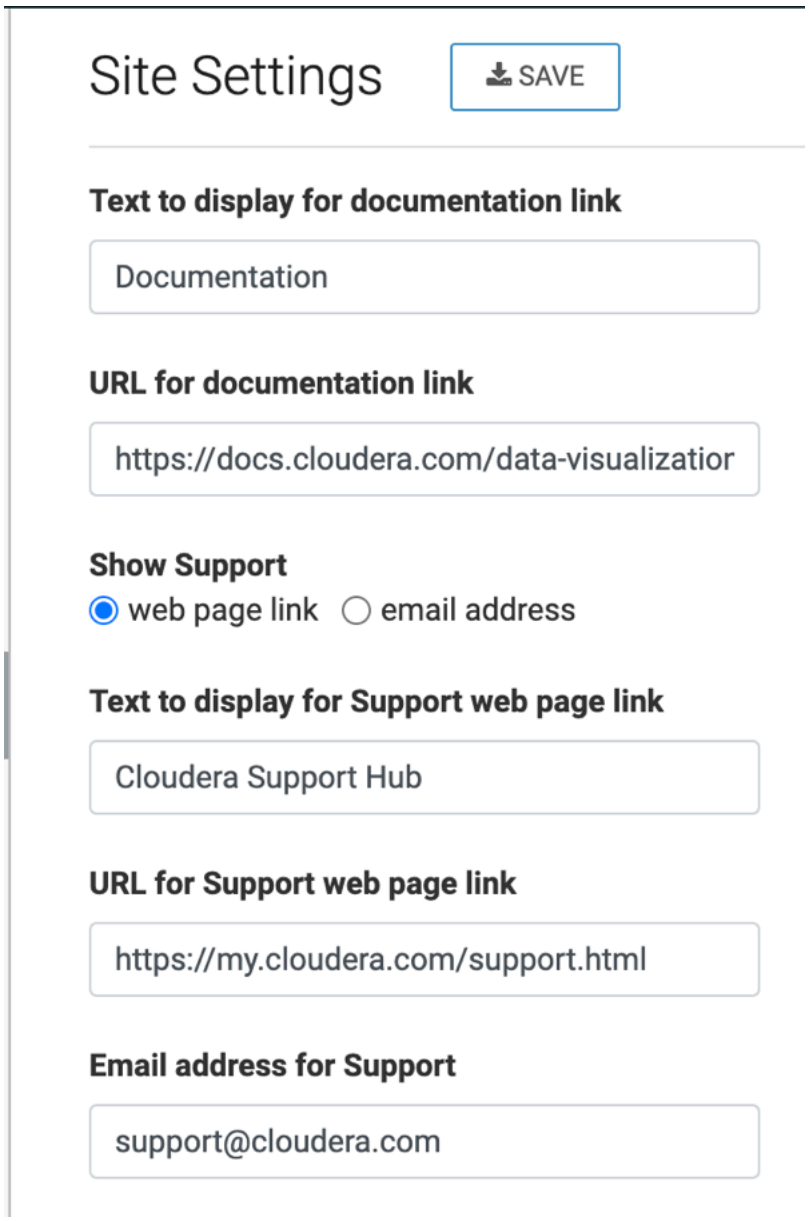

Procedure

- You can change the Text to display for documentation link from the default Documentation to any other title, for example as Customer Documentation.
- You can change the URL for documentation link from the default https://docs.cloudera.com/data-visualization/7/index.html to a different URL, such as http://books.mycompany.com.
- You can set whether you want to show a webpage link or an email address for Support contact information.
  - You can change the Text to display for Support web page link option from the default Cloudera Support Hub and the URL for Support web page link from the default https://my.cloudera.com/support.html.
  - You can change the Email address for Support from the default support@cloudera.com to another email, such as support@mycompany.com.

    Remember:

    Save your changes for the updated site settings to take effect, but there is no need to restart the Cloudera Data Visualization application.

Customizing Help through configuration file

You can customize the Help menu on the main navigation bar by deleting default links provided by Cloudera Data Visualization, or by adding new links.

About this task

To customize the Help menu, modify the local configuration file, such as settings.py, and restart the server.

Components of the Help Menu

By default, the Help menu contains links to What's New in the current release, Documentation, Knowledge Base, YouTube Channel, Support Email, licensing status, and the current version of Cloudera Data Visualization.

To modify this menu, you must change the corresponding settings from TRUE to FALSE.

HELP_SHOW_WHATSNEW

Set to TRUE to show the release number of the running instance of Cloudera Data Visualization.

```
HELP_SHOW_WHATSNEW = True
```

This is TRUE by default. To remove the "What's new" link, set the parameter to FALSE:

```
HELP_SHOW_WHATSNEW = False
```

HELP_SHOW_DOCUMENTATION

Set to TRUE to show the link to our documentation at documentation.arcadiadata.co m/release_number/.

```
HELP_SHOW_DOCUMENTATION = True
```

This is TRUE by default. To remove the "Documentation" link, set the parameter to FALSE:

```
HELP_SHOW_DOCUMENTATION = False
```

HELP_SHOW_SUPPORT_EMAIL

Set to TRUE to show a link to contact our support team at support@arcadiadata.com.

```
HELP_SHOW_SUPPORT_EMAIL = True
```

This is TRUE by default. To remove the "Support" link, set the parameter to FALSE:

```
HELP_SHOW_SUPPORT_EMAIL = False
```

HELP_SHOW_VERSION

Set to TRUE to show the release number of the running instance of Cloudera Data Visualization.

```
HELP_SHOW_VERSION = True
```

This is TRUE by default. To remove the version, set the parameter to FALSE:

```
HELP_SHOW_VERSION = False
```

HELP_SHOW_LICENSE

Set to TRUE to show if your license is valid.

```
HELP_SHOW_LICENSE = True
```

This is TRUE by default. To remove the license information, set the parameter to FALSE:

```
HELP_SHOW_LICENSE = False
```

HELP_MENU_LINKS

To add custom links that supplement the default Cloudera Data Visualization help links, use the HELP_MENU_LINKS parameter and a list of links in the form of type:link, link:url_address, and label:link_label.

You can also insert a divider between these custom links, in the form: type:divider,

```
HELP_MENU_LINKS = [
  {
    "type": "link",
    "link": "url_address_1",
    "label": "link_label_1",
  }
  {
    "type": "link",
    "link": "url_address_2",
    "label": "link_label_2",
  }
  ...
  {
    "type": "divider",
  }
  {
    "type": "link",
    "link": "url_address_n",
    "label": "link_label_n",
  }
]
```

Sample Help Menu Configuration

Here is a sample of configuration settings for the Help menu:

```
# To display the default Data Visualization help links in the Help menu, set
 the flag to "True"
HELP_SHOW_WHATSNEW = True
HELP_SHOW_DOCUMENTATION = True
HELP_SHOW_SUPPORT_EMAIL = True
HELP_SHOW_VERSION = True

# To remove an Arcadia link from the Help menu, set the flag to "False"
HELP_SHOW_LICENSE = False
# To add new links in the Help menu, enter the following code:
#In this example, we are adding links to Knowledge Base and YouTube Channel.
HELP_MENU_LINKS = [
  {
    "type": "link",
    "link": "https://www.arcadiadata.com/resources/knowledge-base/",
    "label": "Knowledge Base",
  },
#  You can also specify a divider between the two rows as follows:
#  {
#    "type": "divider",
```

```
#  },
  {
    "type": "link",
    "link": "https://www.youtube.com/channel/UCmubDZSISQMKg1Q0HXo03HQ",
    "label": "YouTube Channel",
  },
]
```

Managing advanced settings

Cloudera Data Visualization allows users with administration privileges to configure several advanced settings.

Webhook

You can use webhooks to pass parameters to any API endpoint that you might want to update with a dashboard's parameters.

To interact with webhooks, check the Allow users to interact with webhooks option and enter the URLs in the URLs that webhook actions are permitted to interact with text box. If you add multiple values, separate them by a new line.

Remote data

To provide URLs for remote data, enter the URLs in the URLs that remote data columns can be fetched from text box. If you add multiple values, separate them by a new line.

> **Remember:**
>
> Save your changes for the updated site settings to take effect, but there is no need to restart the Cloudera Data Visualization application.

Related Information

Securing Applications

Advanced site settings

You can configure various advanced settings in Cloudera Data Visualization by adding them to the Advanced Site Settings text box.

Autologin

Users with administrative privileges can enable unauthenticated access through automatic login:

In Cloudera Machine Learning and Cloudera Data Science Warehouse, authentication for applications is enforced on all ports by default. If needed, an Admin user can allow users to create public applications accessible by unauthenticated users. This is needed if you do not want users to log into Cloudera Machine Learning or Cloudera Data Science Warehouse. For information on how to enable unauthenticated access in Cloudera Machine Learning/ Cloudera Data Science Warehouse, see *Public Applications* in *Securing Applications*.

To configure autologin for a user in Cloudera Data Visualization, add the following settings in the Advanced Site Settings text box in key = value format:

```
AUTOLOGIN_ENABLED = True
AUTOLOGIN_USERNAME = "username"
AUTOLOGIN_PASSWORD = "password"
```

Important: Make sure that the username and password you provide in the advanced settings match the login credentials of the user designated for automatic login. Since anyone accessing the link will have the same permissions as that user, carefully choose the user and role for configuring unauthenticated access.

Remember:

Save your changes and restart the Cloudera Data Visualization application for the changed advanced settings to take effect.

Concurrency

To configure concurrency, add the desired settings in the Advanced Site Settings text box in key = value. The following options are available:

DATACONNECTION_CONCURRENCY_DEFAULT	Default number of concurrent requests per data connections
DATACONNECTION_CONCURRENCY_MAX	Maximum number of concurrent requests per data connections
DATACONNECTION_CONCURRENCY_USER_DEFAULT	Default number of concurrent requests per user
DATACONNECTION_CONCURRENCY_USER_MAX	Default number of maximum concurrent requests per user

These settings control the number of concurrent requests per data connection and per user.

Remember:

Save your changes and restart the Cloudera Data Visualization application for the changed advanced settings to take effect.

Data extracts

To configure data extracts, add the desired settings in the Advanced Site Settings text box in key = value format. The following options are available:

Prefix for target table names

You can define a prefix for table names. The default value is *CDV_*.

Note: If the value is set to an empty string, no prefix is applied.

```
DATA_EXTRACTS_TARGET_TABLENAME_PREFIX = "cdv_"
```

Maximum number of rows that can be inserted into a data connection by CDV using data extracts or CSV upload

You can set a system wide maximum value for the maximum number of rows for upload/import all connections.

```
UPLOAD_MAXROWS_MAX = 1000000000
```

You can set the default value for the maximum number of rows for upload/import for all connections. You can change the default on each connection in the Edit connection modal. Its value can be only between 0 and UPLOAD_MAXROWS_MAX.

Note: This limit does not apply when the data extract source is on the same connection as data extract target.

```
UPLOAD_MAXROWS_DEFAULT = 1000000
```

Note: This limit does not apply when the data extract source is on the same connection as the data extract target.

Number of rows inserted at a time through data import and data extract inserts

```
IMPORT_BATCH_INSERT = 1000
```

Remember:

Save your changes and restart the Cloudera Data Visualization application for the changed advanced settings to take effect.

Notifications

To configure how long read notifications are kept in the Notifications Center, add the BULK_REMOVE_READ _NOTIFICATION_INTERVAL flag in the Advanced Site Settings text box.

Remember:

Save your changes and restart the Cloudera Data Visualization application for the changed advanced settings to take effect.

Size limit for static asset upload

To configure the size limit for uploading static assets, add the DATA_UPLOAD_MAX_MEMORY_SIZE flag in the Advanced Site Settings text box.

Remember:

Save your changes and restart the Cloudera Data Visualization for the changed advanced settings to take effect.

Filter association configuration

To enable Filter Association (FA) based on Manage dataset permission, add the MANAGE_DS_FA = True flag in the Advanced Site Settings text box.

Note: If you gain the Manage filter associations permission by adding the MANAGE_DS_FA flag, ensure you also have View roles and users permission to be able to work with filter associations.

Cloudera Data Visualization 7.2.9

Static Assets

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing static assets 4

Creating new assets 5

Viewing an existing asset 7

Changing existing assets 7

Automatically uploading assets 10

Deleting an existing asset 10

Managing static assets

Cloudera Data Visualization enables you to save and manage images (PNG and JPG format), CSS files, and JS files as Static Assets. You can subsequently use them across all reports to standardize the appearance and behavior of visuals and dashboards.

About this task

Note: This feature is only available to users with Manage custom styles or Administrator privileges.

To use the Static Assets interface, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Click Static Assets.

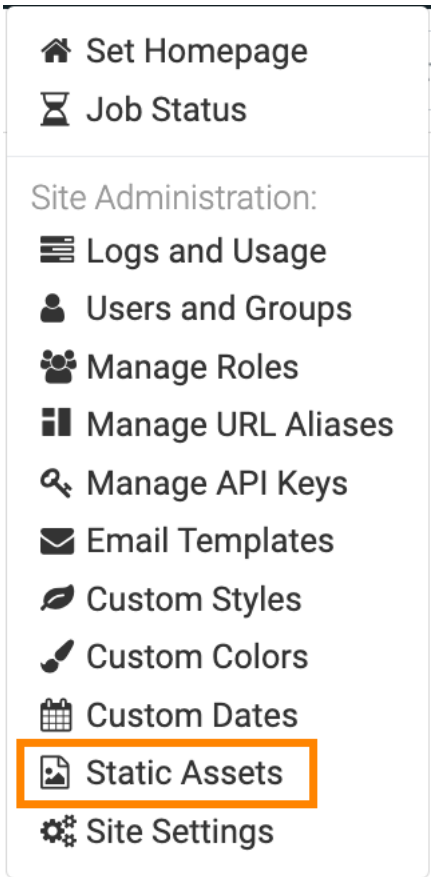

3. In the Manage Image, CSS, and JS Assets interface, you have the following actions available:
   - Click NEW ASSET to create a new asset. For more information, see *Creating new assets*.
   - Click the ID number of a particular asset to view it. For more information, see *Viewing an existing asset*.
   - Click the Pencil icon to edit the information about a particular asset. For more information, see *Changing existing assets*.
   - Click the Trash can icon to delete the asset. For more information, see *Deleting an existing asset*.

What to do next

To use a static asset in a visual, such as Rich-Text visual type, follow the instructions in *Inserting images*.

Related Information

Creating new assets
Viewing an existing asset
Changing existing assets
Deleting an existing asset
Inserting images

Creating new assets

About this task

To create a new asset, follow these steps.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Click Static Assets.

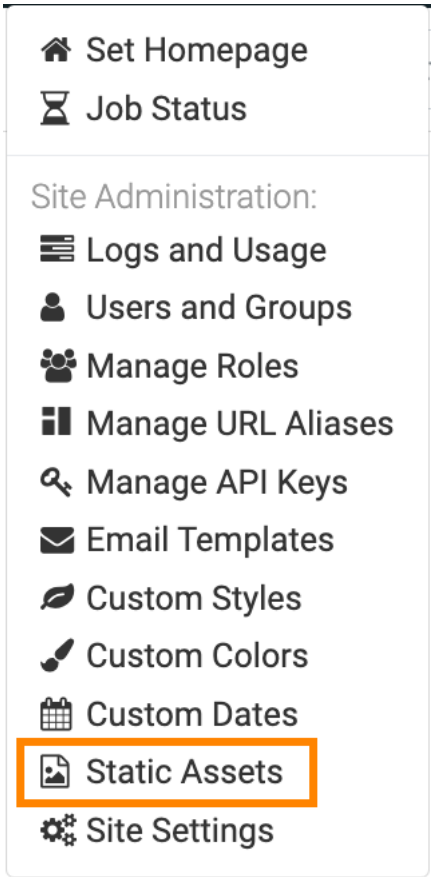

3. In the Manage Image, CSS, and JS Assets interface, click NEW ASSET.

4. In the New Asset interface, define the new asset:
   a) Enter the Name of the asset.

      In this example, company_logo is used.
   b) Provide a Description for the asset.

      In this example, Cloudera Logo is used.
   c) Under File, click Choose File and use your system interface to add the file to Cloudera Data Visualization.
   d) After you add the file, Cloudera Data Visualization adds the correct information to the immutable fields File Type and Size.

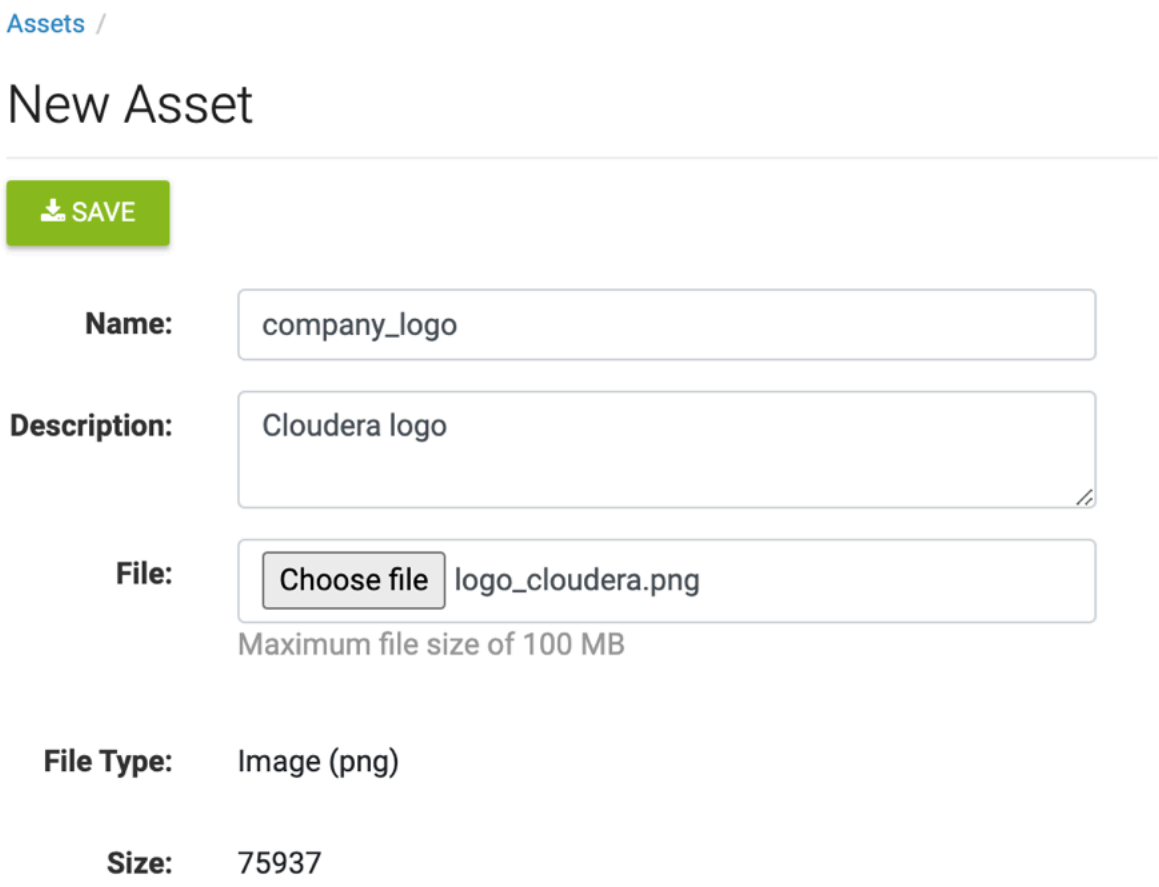

5. Click SAVE.

   The Asset information interface appears.

6. You can change the Name and Description of the asset, and even replace the file, using Change File. The asset also has its assigned ID.

   The asset also has a clickable URL for viewing its contents. This is equivalent to the linked ID on the Manage Image, CSS, and JS Assets interface. Also, note the navigation path to the asset; in this case, it is:

   ```
   http://127.0.0.1:8000/arc/reports/staticasset/raw/1.png
   ```

   In general, the link has the following form:

   ```
   [http|https]://ip_address/arc/reports/staticasset/raw/ID.file_extension
   ```

   Where ID is the ID of the asset as it appears on the interface, and file_extension is one of the supported file formats: png, jpg, css, or js.

Viewing an existing asset

Procedure

You can view an existing asset either by clicking the linked ID of the asset in the Manage Image, CSS, and JS Assets interface, or by clicking URL when editing specific asset information.

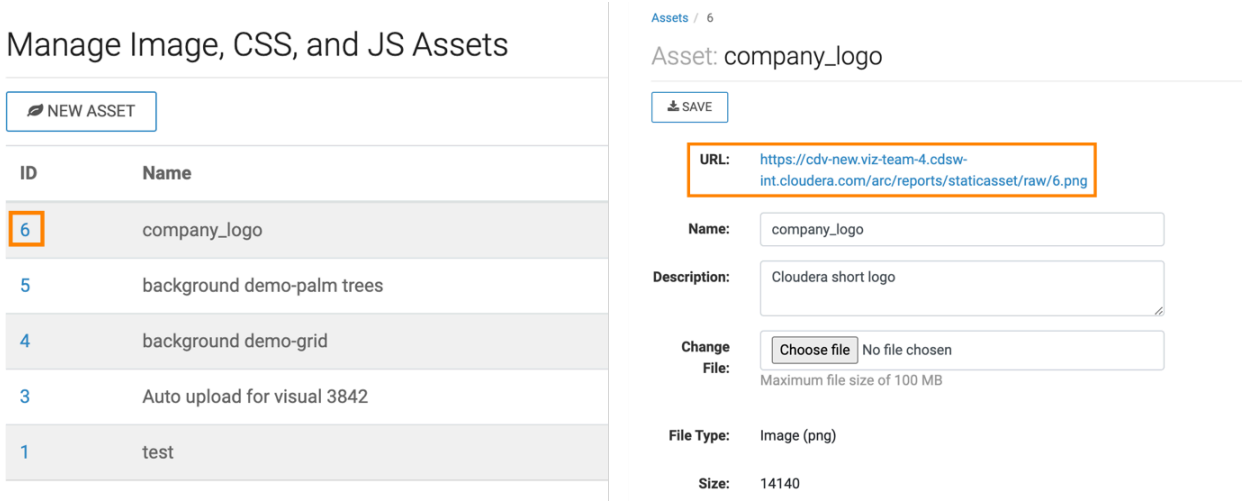

The asset opens in a new tab in your browser window.

Note: CSS and JS assets show the code listing.

Changing existing assets

About this task

To change an existing asset, follow these steps.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.

2. Click Static Assets.

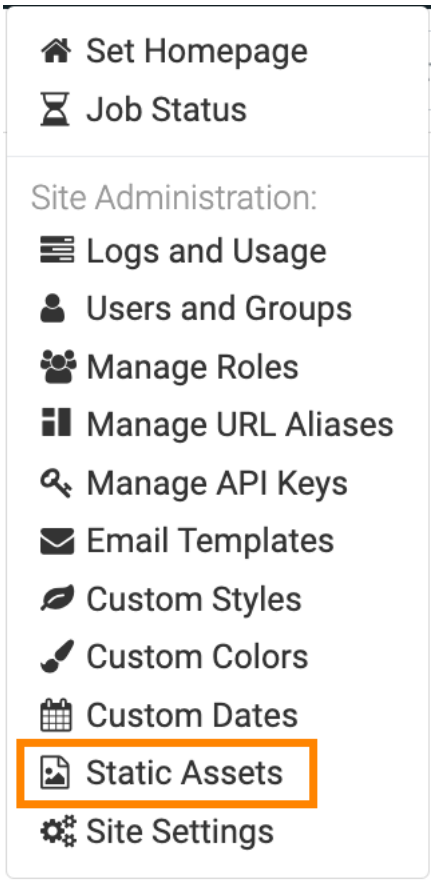

3. In the Manage Image, CSS, and JS Assets interface, click the Pencil icon that corresponds to the asset.

   The Asset information interface appears. It contains the information about the asset.

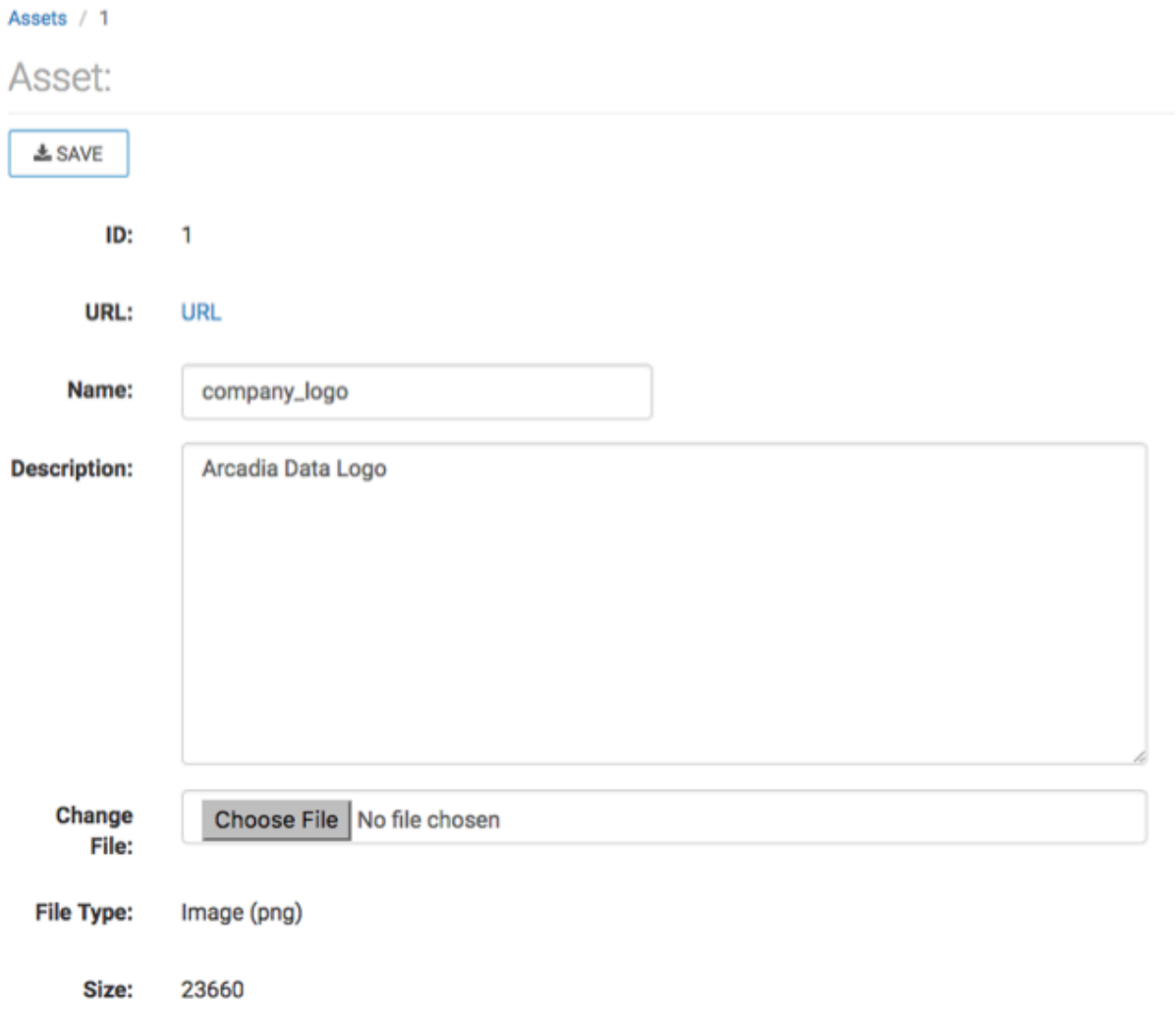

4. In the Asset interface, you can perform the following actions:

   - Change the Name of the asset.
   - Change the Description of the asset.
   - Click URL to view the asset. This is equivalent to the linked ID on the Manage Image, CSS, and JS Assets interface. Also, note the navigation path to the asset; in our case, it is:

     ```
     http://127.0.0.1:8000/arc/reports/staticasset/raw/1.png
     ```

     In general, the link has the following form:

     ```
     [http|https]://ip_address/arc/reports/staticasset/raw/ID.file_extension
     ```

     Where ID is the ID of the asset as it appears on the interface, and file_extension is one of the supported file formats: png, jpg, css, or js.
   - To substitute the file by a newer version, click Choose File and use the file system to select and upload a replacement.
   - Click SAVE after making changes.

Automatically uploading assets

When using a Rich Text visual, Cloudera Data Visualization automatically uploads local files into the static asset database. These assets are identified as "Auto upload" in the static asset interface.

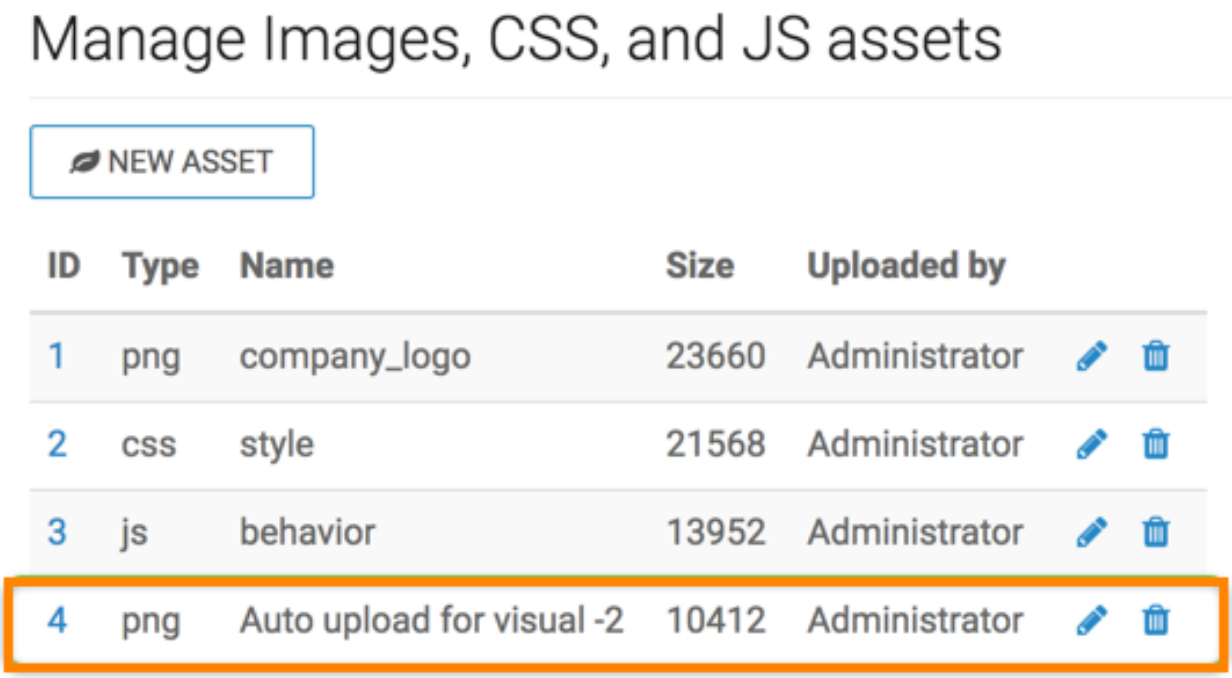

Proceed to editing the static asset description, as explained in *Changing existing assets*.

Related Information

Changing existing assets

Deleting an existing asset

About this task

To delete an existing asset, follow these steps.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.

2. Click Static Assets.

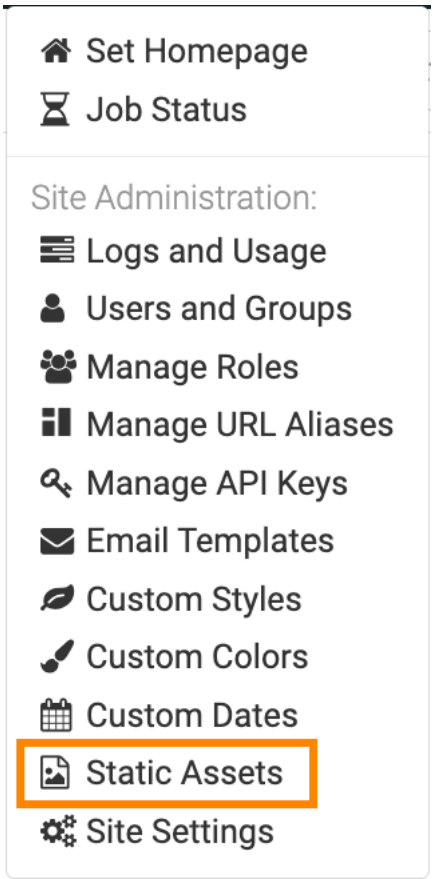

3. In the Manage Image, CSS, and JS Assets interface, click the Trash can icon that corresponds to the asset.

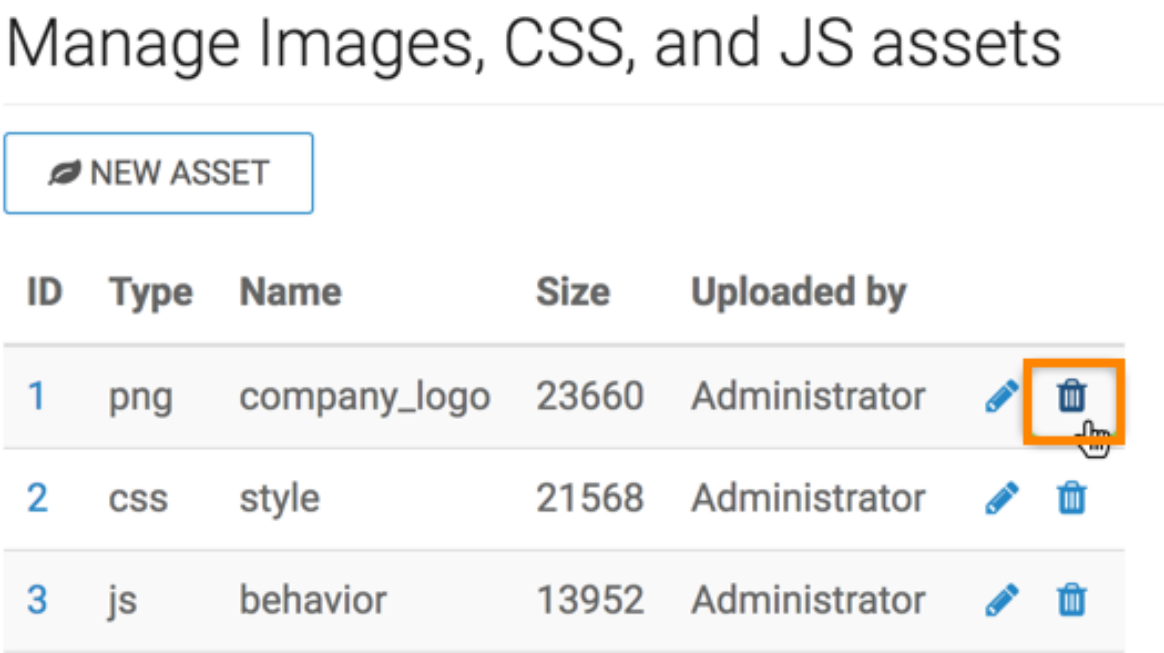

4. The Delete Confirmation modal, type 'DELETE' and click DELETE.

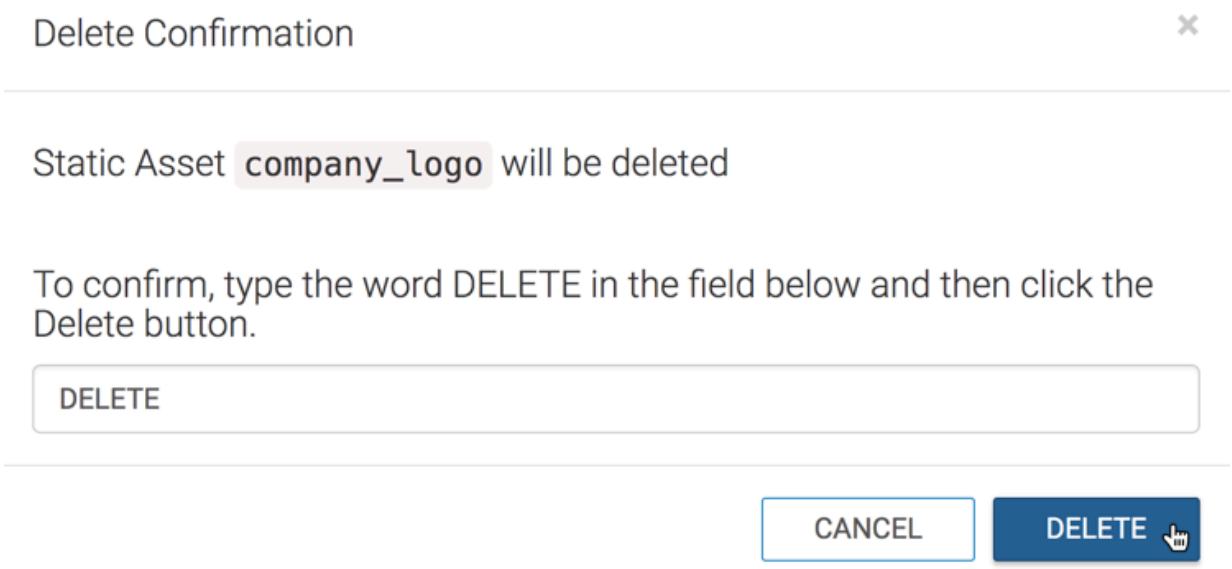

Cloudera Data Visualization 7.2.9

Managing URL aliases

Date published: 2020-10-30
Date modified: 2024-12-18

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing dashboard URL aliases.....4

Editing dashboard URL aliases..... 5

Managing dashboard URL aliases

Cloudera Data Visualization enables you to manage all your URL aliases on one page. You can view, edit, delete, and copy the short URLs.

About this task

Note: You need administrative privileges to manage URL aliases.

You can find the following features on the Managing dashboard URL aliases interface:

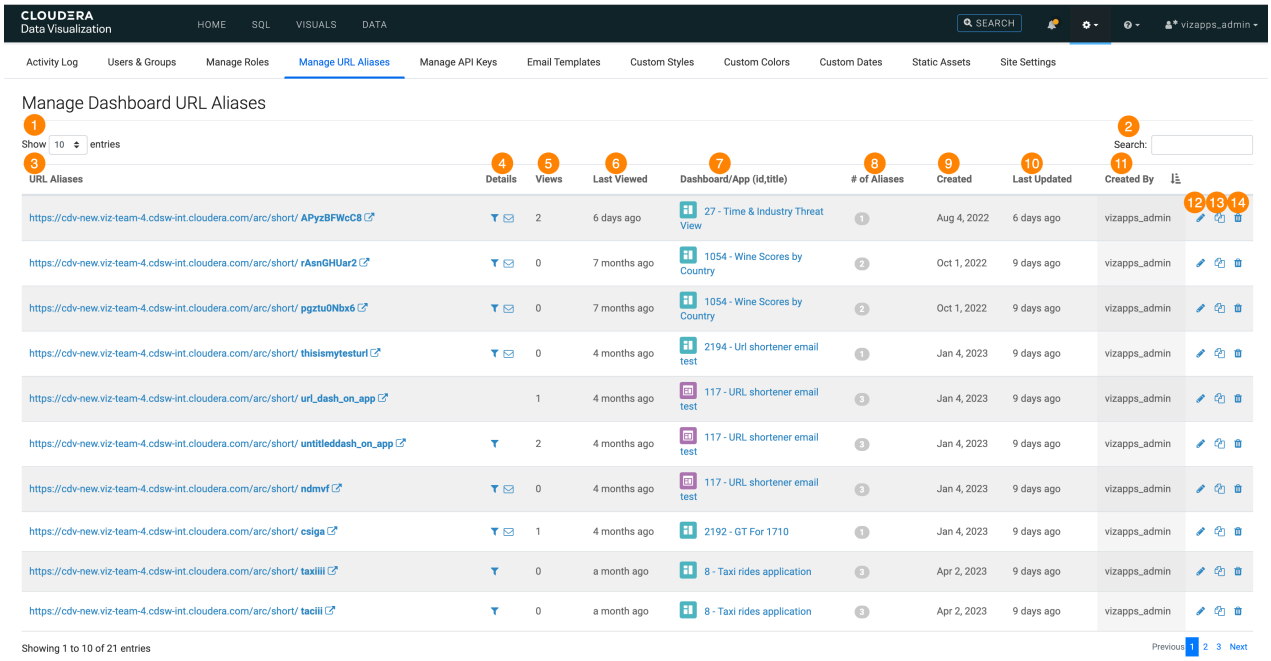

1. Using the Show function, you can adjust the number of entries that appear on the page.
2. Using the Search box, you can search among the listed short URLs by alias, dashboard title, or app title.
3. The first column of the list contains the URL aliases. You can open them in a separate browser tab.
4. The Details column provides information on parameters and usage in emails. The email icon is displayed if the alias was created through the email modal or the alias was sent out in an email.
5. Views shows you the number of views of a particular short URL.
6. Last Viewed provides information on when an alias was last viewed.
7. The Dashboard/App column shows you the ID and title of the dashboard or app. You can open it by clicking on the name.
8. # of Aliases shows you how many aliases the dashboard or the app has.
9. Created gives you the creation date of the alias.
10. Last Updated provides information on when an alias was last edited.
11. Created By shows who created the alias.
12. You can use the Pencil icon to edit the short URL. For more information, see *Editing dashboard URL aliases*.
13. You can use the Copy icon to copy the short URL to the clipboard.
14. You can use the Trash can icon to delete the selected short URL.

Related Information

Editing dashboard URL aliases

Editing dashboard URL aliases

Cloudera Data Visualization enables you to edit a URL alias after you created it.

About this task

Note: You need administrative privileges to create and edit URL aliases.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu and select Manage URL Aliases.
   The Manage Dashboard URL Aliases interface opens.
2. Identify the URL alias that you want to edit and click the pencil icon at the end of the row.
   The Set the URL Alias to a dashboard or an app modal window appears.

3. Edit the URL alias details as needed.

   You can update the following information:

   URL Alias

   This field contains the generated short URL. You can customize the end of the link.

   Type

   You can set whether the URL alias points to a dashboard or an app.

   Note: The subsequent fields depend on the type option selected.

   .

   Associated Dashboard / App

   The text field is pre-filled, the dashboard or app associated with the URL alias. You can switch to a different dashboard or app, if needed.

   Sheet / Menu Tab

   This text field depends on the Associated Dashboard / App field. If there are more sheets on the selected dashboard or the selected app contains several tabs, you can switch between these and select which is associated with the URL alias.

   Current Dashboard / App Parameters

   The text field is pre-filled with previously saved parameters, if the alias has been saved with parameters . Otherwise, the text area is empty.

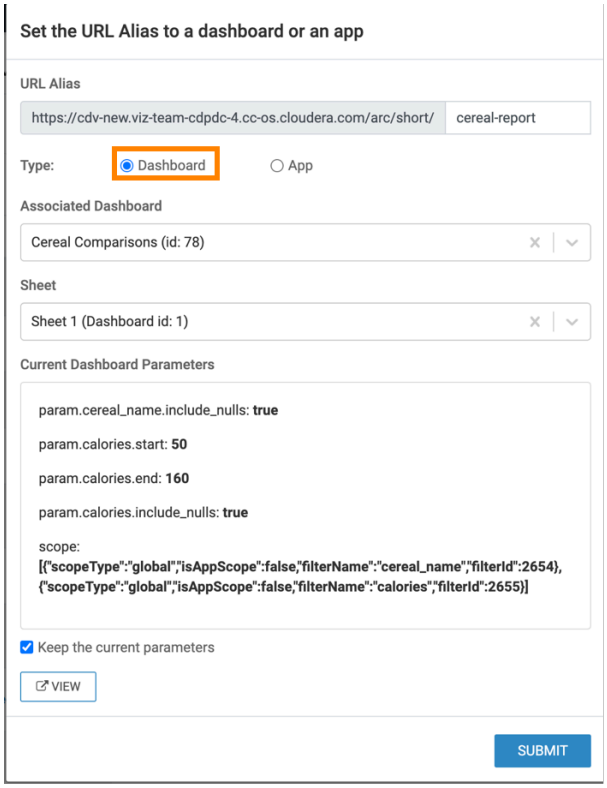

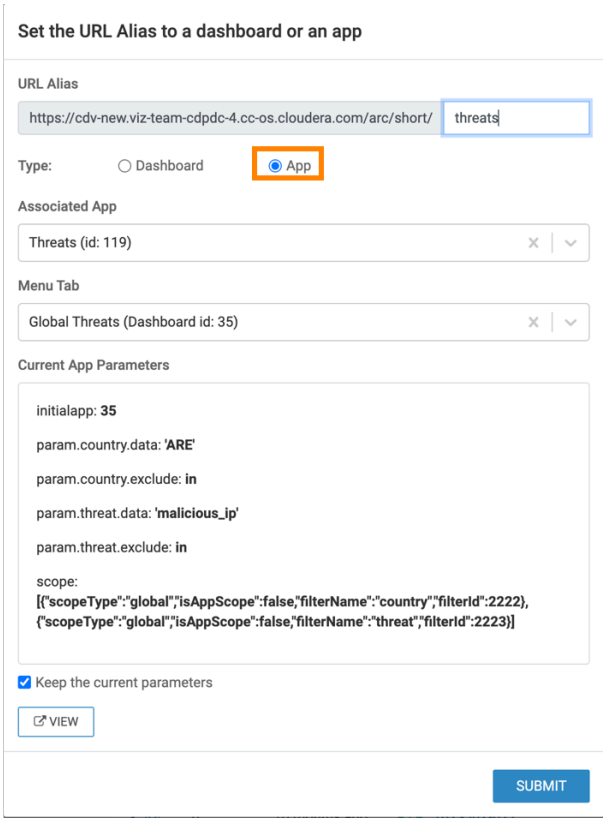

4. Select the Keep the current parameters checkbox if you want the listed parameters assigned to the URL alias and the associated dashboard / app.

   If you do not select the checkbox, the parameters will be overwritten, and only the sheet or initial app parameters will be kept.

5. Click VIEW if you want to check the dashboard / app the URL alias belongs to.

6. Click SUBMIT.

Cloudera Data Visualization 7.2.9

Users and User Groups

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating new user accounts....4

Editing accounts....8

Changing user passwords....12

Promoting a user to administrator....16

Creating new administrator accounts....17

Inactivating user accounts....22

Assigning roles to users....22
- Assigning roles to a single user....22
- Assigning roles to multiple users....24
- Assigning multiple roles to multiple users....28

Deleting user accounts....31
- Deleting individual user accounts....31
- Deleting several user accounts....33

Creating new user groups....34

Editing user groups....37

Adding users to user groups....38
- Adding a user to multiple groups; user modal....39
- Adding multiple users to a single group; group modal....40
- Adding multiple users to multiple groups; users tab....41

Assigning roles to user groups....41

Deleting user groups....43

Creating new user accounts

The following steps demonstrate how to create a new user account.

Before you begin

You must have administrative privileges to perform this action.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Click Users & Groups.

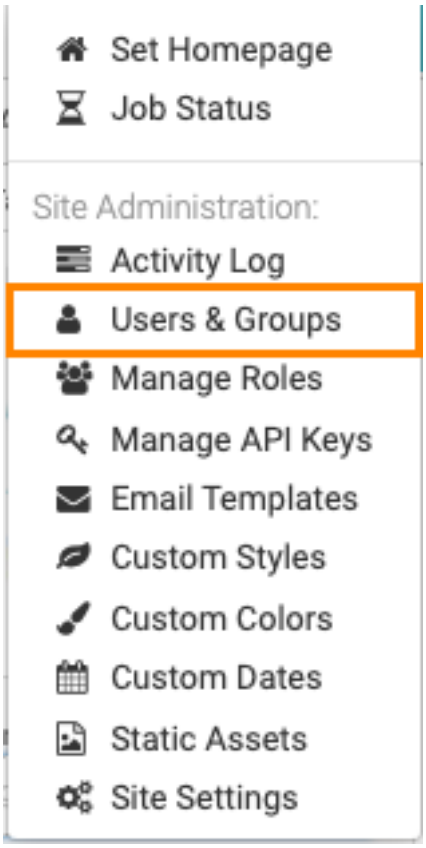

The Manage Users & Groups interface appears, open on the Users tab.

3. Click NEW USER.

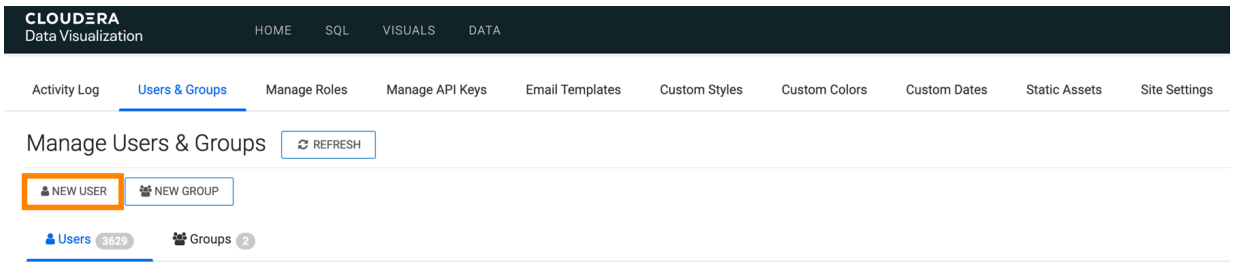

The New User modal window appears.

4. Fill in the new user details:
   - Name & password
     - Username
     - First Name and Last Name

       You can also pick these values from LDAP user definitions.
     - Password

       Important: When you enter your password, ensure that the password meets the minimum security requirements specified for Cloudera Data Visualization.

       For more information, see *Setting user password requirements*.

       After five failed login attempts, Cloudera Data Visualization locks out the user. For instructions on how to change the account lockout configuration settings or to unlock the user, see *Configuring account lockout*. You can check the Show Password option to display the password.
   - Permissions
     - Normal user and Admin user

       Select Normal user.
     - [Optional] On the Groups tab, under Group Membership, you can use the Search box to find user groups, and assign the new user to existing groups by selecting user groups on the left list, and then clicking Add

to move them to the right. For more information on how to add users to groups see, *Creating new user groups*.

- [Optional] On the Roles tab, under Roles, you can use the Search box to find roles, and assign the new user to existing roles by selecting roles on the left list, and then clicking Add to move them to the right. For more information on users and roles, see *Assigning roles to users*.

Note: If you have not defined at least one User Group, the New User modal window does not contain Group Membership information.

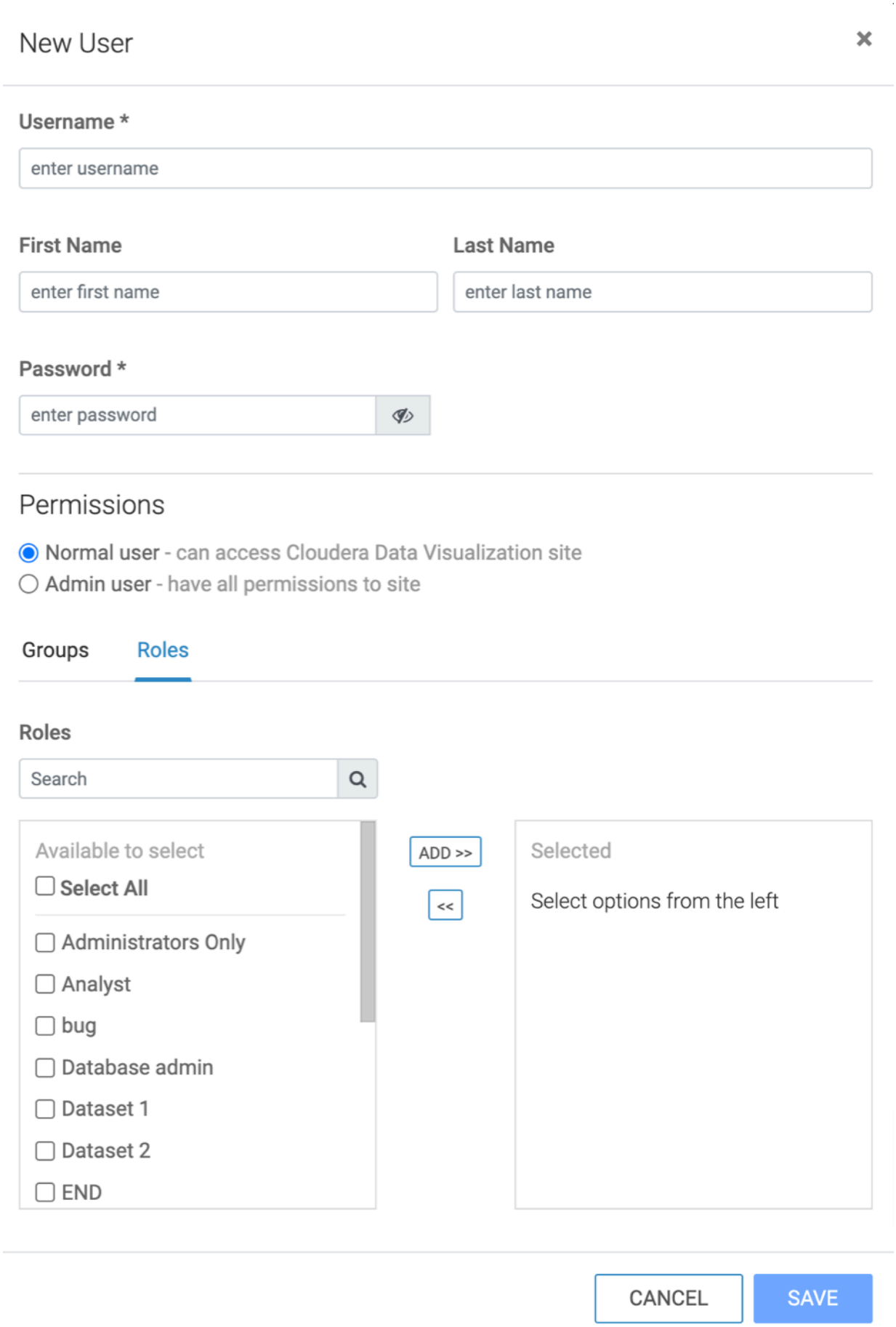
New User
Username *
enter username
First Name
enter first name
Last Name
enter last name
Password *
enter password
Permissions
Normal user - can access Cloudera Data Visualization site
Admin user - have all permissions to site
Groups
Roles
Roles
Search
Available to select
Select All
Administrators Only
Analyst
bug
Database admin
Dataset 1
Dataset 2
END
ADD >>
<<
Selected
Select options from the left
CANCEL
SAVE

5. Click SAVE.

Results

After the operation succeeds, you can see the Manage Users & Groups page, showing the Users tab with the new user's information.

When a user logs in, their first name and last name appear in the top right corner of the navigation bar.

After a new user logs in, notice that the date joined changes and the last login changes from Never to a few seconds ago.

Related Information

Setting user password requirements
Configuring account lockout
Creating new user groups
Assigning roles to users

Editing accounts

You can change the profile of any user or administrator account in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

The following steps demonstrate how to change the username and permissions for a Cloudera Data Visualization user account. The process is the same for changing the administrative account.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.

2. Click Users & Groups.

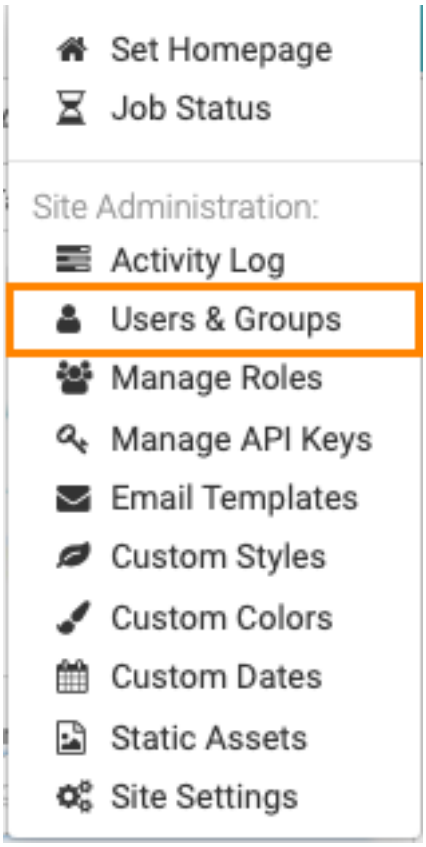

The Manage Users & Groups interface appears, open on the Users tab.

3. In the list of registered users, do one of the following:
   - Click the Username of the user that you want to edit.
   - Click the Pencil icon next to the account that you want to edit.

   The Edit User modal window appears.

4. You can change the following details:
   - Name & password
     - Username
     - First Name and Last Name
     - Password
   - Permissions
     - Normal user and Admin user
     - Group Membership
     - Roles
   - Active or Inactive

Edit User

Username

Anvesh.Koripella

First Name

Anvesh

Last Name

Koripella

Change password

Permissions

Normal user - can access Cloudera Data Visualization site

Admin user - have all permissions to site

Groups Roles

Roles

Search

Available to select

Select All

Analyst

bug

Database admin

Dataset 1

Dataset 2

nur_can't_manage_connections

ADD >>

<<

Selected

Select options from the left

Active Inactive

Date joined: a year ago

Last login: Never

CANCEL DELETE USER SAVE

5. Click SAVE.

Results

After the operation succeeds, you can see the Manage Users & Groups page, showing the Users tab with the updated information.

Changing user passwords

An administrator may reset the password of any user or of another administrator in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

Note: The ‘change password’ option is not available for LDAP and SAML user accounts.

The following steps demonstrate how to change the password for any Cloudera Data Visualization user account.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Click Users & Groups.

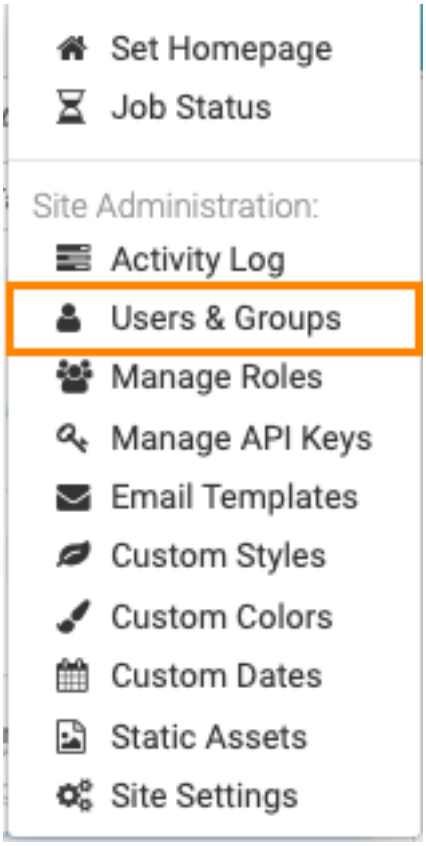

The Manage Users & Groups interface appears, open on the Users tab.

3. In the list of registered users, do one of the following:
   - Click the Username of the account you want to change.
   - Click the Pencil/Edit icon that corresponds to the account you want to change.

   The Edit User modal window appears.

4. In the Edit User modal window, click Change password.

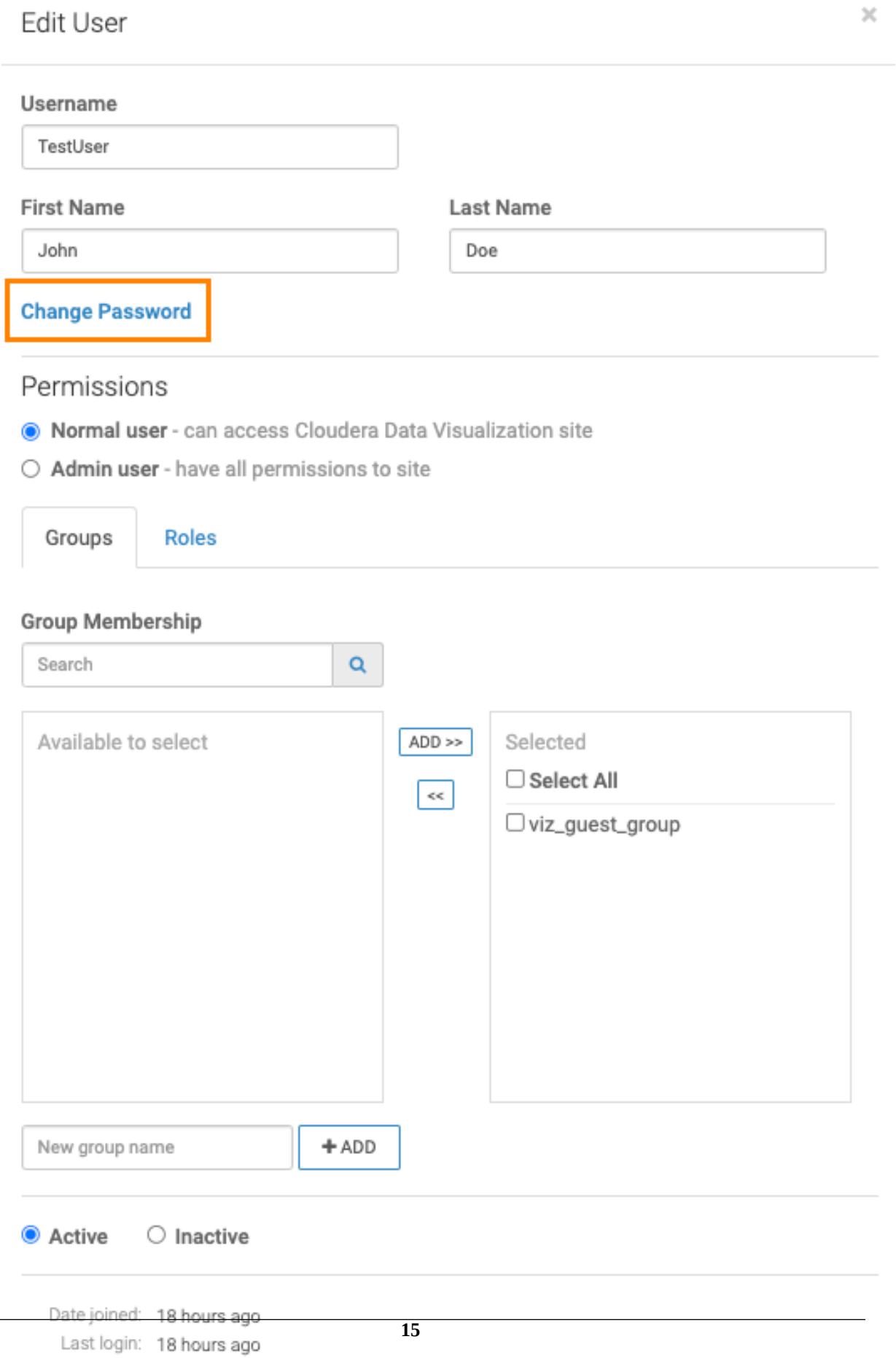
Edit User
Username
TestUser
First Name
John
Last Name
Doe
Change Password
Permissions
Normal user - can access Cloudera Data Visualization site
Admin user - have all permissions to site
Groups
Roles
Group Membership
Search
Available to select
ADD >>
<<
Selected
Select All
viz_guest_group
New group name
+ ADD
Active
Inactive
Date joined: 18 hours ago
Last login: 18 hours ago

5. Enter the new password.

   Consider the following when choosing the password:

   - Ensure that the password meets the minimum security requirements specified for the Cloudera Data Visualization instance. For more information, see *Setting user password requirements*.
   - After five failed login attempts, Cloudera Data Visualization locks out the user. For instructions on how to change the account lockout configuration settings or to unlock the user, see *Configuring account lockout*.

   You can check the password you have added if you check the Show Password option.

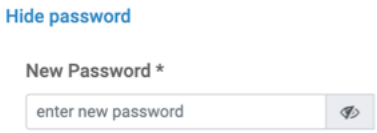

6. Click SAVE.

Results

The user's password is changed and you get back to the Manage Users & Groups interface.

Related Information

Setting user password requirements
Configuring account lockout

Promoting a user to administrator

You can promote any user to an administrator in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

Procedure

1. To promote the user to administrator, open the user account in Edit mode and change the Permissions by selecting Admin user.

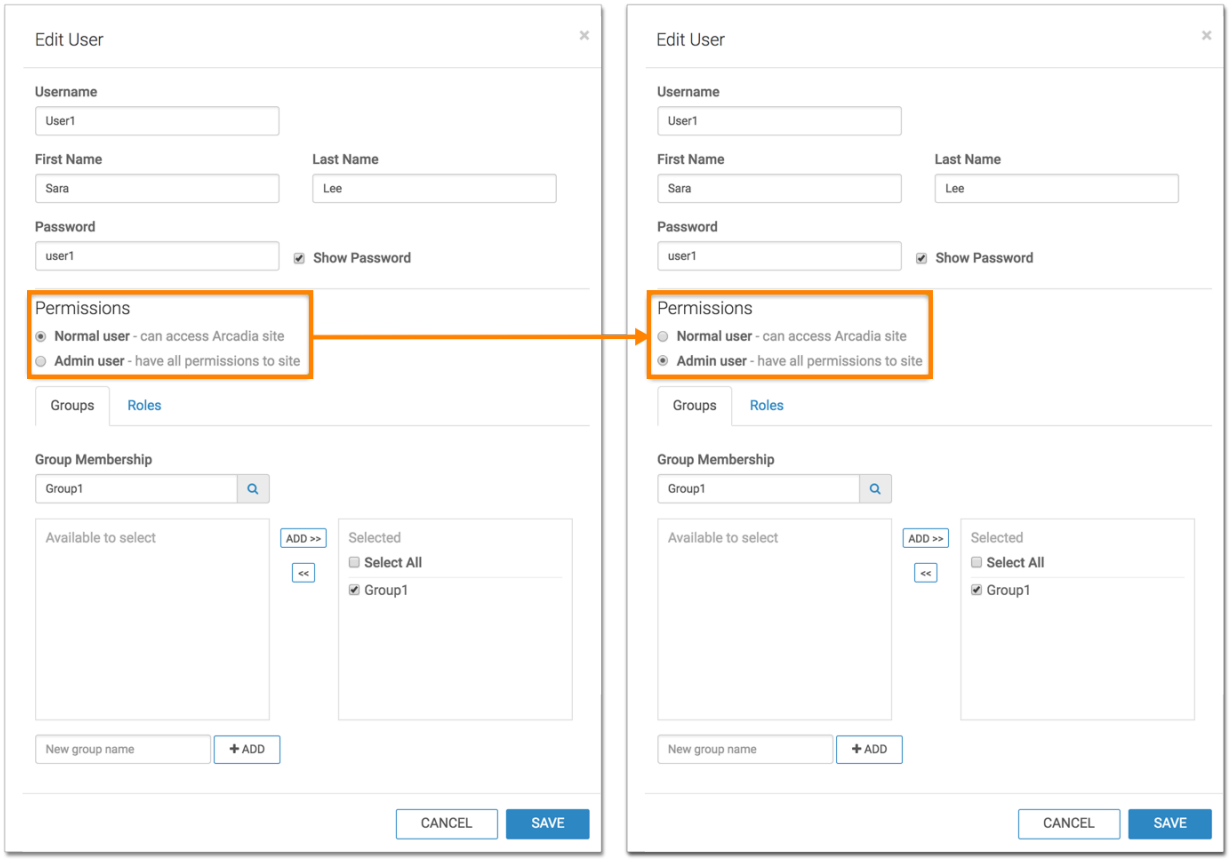

2. To demote an administrator to plain user, select Normal User under Permissions.

Creating new administrator accounts

An administrator can create new administrator accounts in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

The following steps demonstrate how to create a new admin account.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.

2. Click Users & Groups.

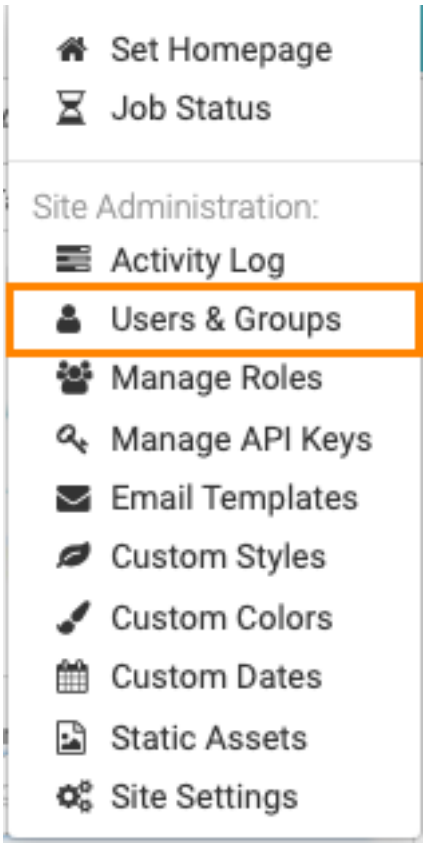

The Manage Users & Groups interface appears, open on the Users tab.

3. Click NEW USER.

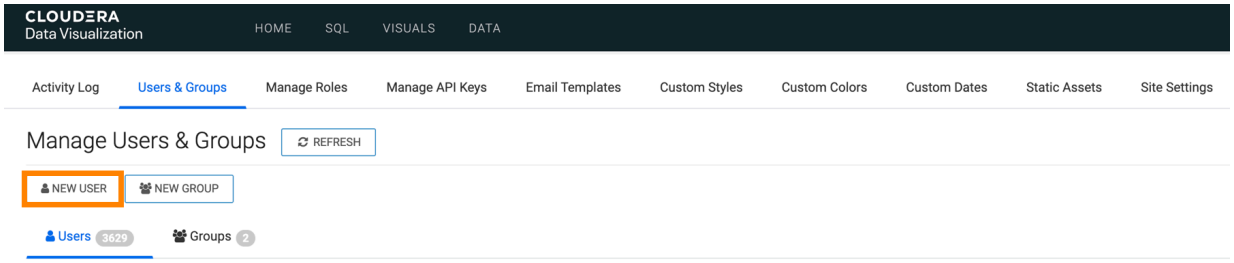

The New User modal window appears.

4. Fill in the new admin details:
   - Name & password
     - Username
     - First Name and Last Name

       You can also pick these values from LDAP user definitions.
     - Password

       Important: When you enter your password, ensure that the password meets the minimum security requirements specified for Cloudera Data Visualization.

       For more information, see *Setting user password requirements*.

       After five failed login attempts, Cloudera Data Visualization locks out the user. For instructions on how to change the account lockout configuration settings or to unlock the user, see *Configuring account lockout*. You can check the Show Password option to display the password.
   - Permissions
     - Normal user and Admin user

       Select Admin user.
     - [Optional] On the Groups tab, under Group Membership, you can use the Search box to find user groups, and assign the new user to existing groups by selecting user groups on the left list, and then clicking Add

to move them to the right. For more information on how to add users to groups see, *Creating new user groups*.

- [Optional] On the Roles tab, under Roles, you can use the Search box to find roles, and assign the new user to existing roles by selecting roles on the left list, and then clicking Add to move them to the right. For more information on users and roles, see *Assigning roles to users*.

Note: If you have not defined at least one User Group, the New User modal window does not contain Group Membership information.

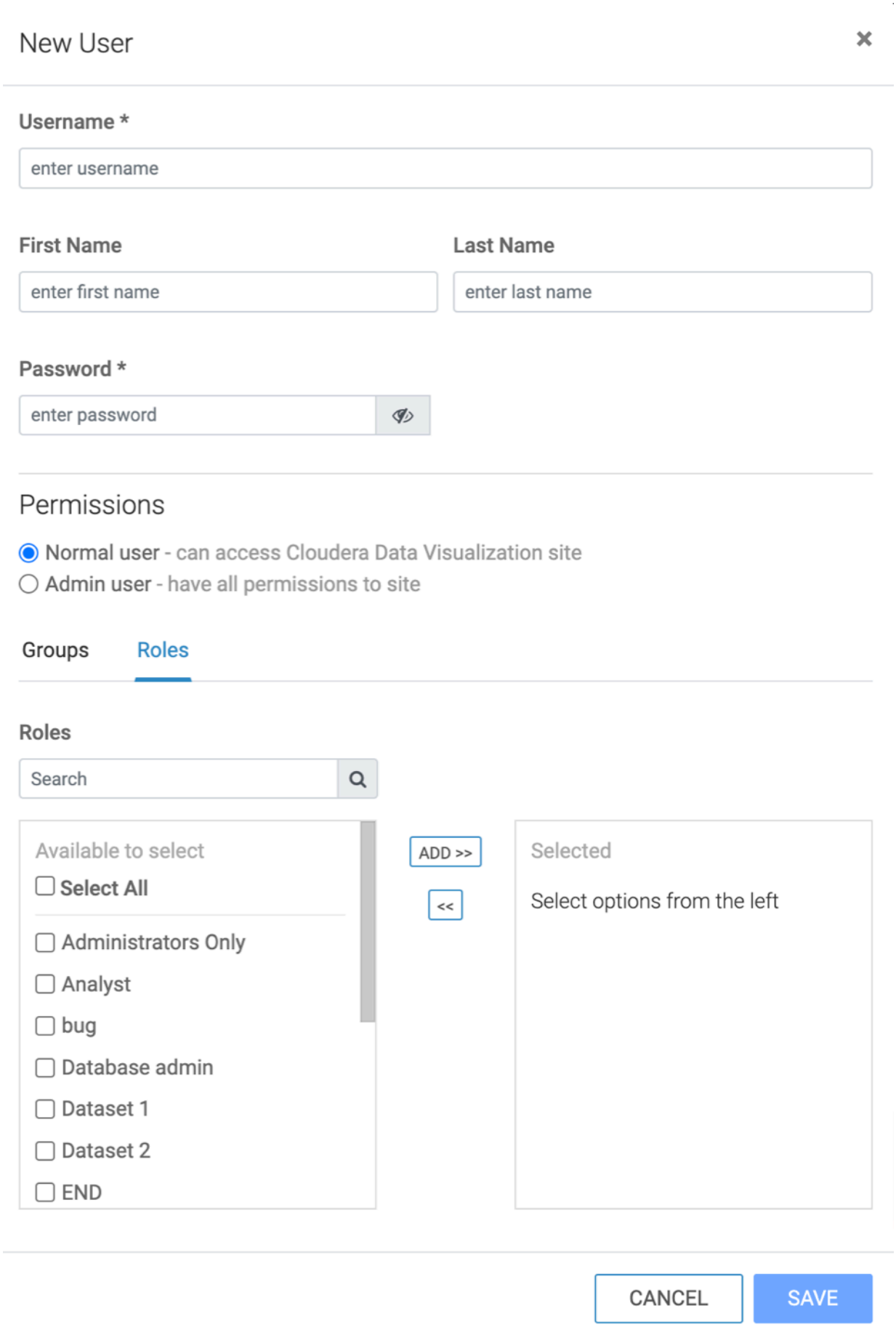
New User
Username *
enter username
First Name
enter first name
Last Name
enter last name
Password *
enter password
Permissions
Normal user - can access Cloudera Data Visualization site
Admin user - have all permissions to site
Groups
Roles
Roles
Search
Available to select
Select All
Administrators Only
Analyst
bug
Database admin
Dataset 1
Dataset 2
END
ADD >>
<<
Selected
Select options from the left
CANCEL
SAVE

Results

After the operation succeeds, you can see the Manage Users & Groups page, showing the Users tab with the new admin user's information.

Related Information

Setting user password requirements

Creating new user groups

Inactivating user accounts

If it is necessary, you can inactivate a user's account without removing it from the system in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

The following steps demonstrate how to inactivate a Cloudera Data Visualization user account.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Click Users & Groups.

   The Manage Users & Groups interface appears, open on the Users tab.
3. In the list of Users, click either the Username of the account you want to inactivate, or the Edit (Pencil) icon.
4. In the Edit User modal window, select Inactive.
5. Click SAVE.
6.

Results

After the operation succeeds, you can see the Users list on the screen. The inactiveated user is part of the list, and the status shown is set to inactive.

Assigning roles to users

You can assign roles to users in Cloudera Data Visualization.

Note: This feature is only available to users with *Manage roles and users* privileges.

An administrator can assign users to roles in several ways:

Assigning roles to a single user

You ocan add security roles to a user's profile in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Click Users & Groups.

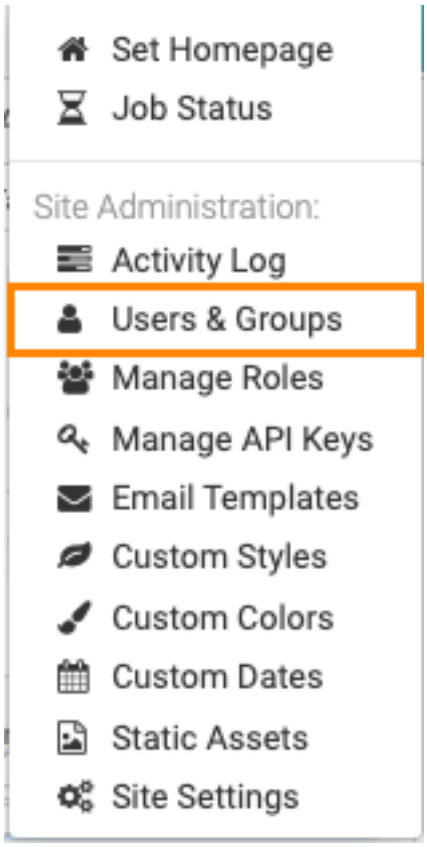

The Manage Users & Groups interface appears, open on the Users tab.

3. In the list of registered users, select the user for which you plan to assign new roles, and click the edit (Pencil) icon that corresponds to that account.

   Alternatively, you can click the Username of the account that you want to change.

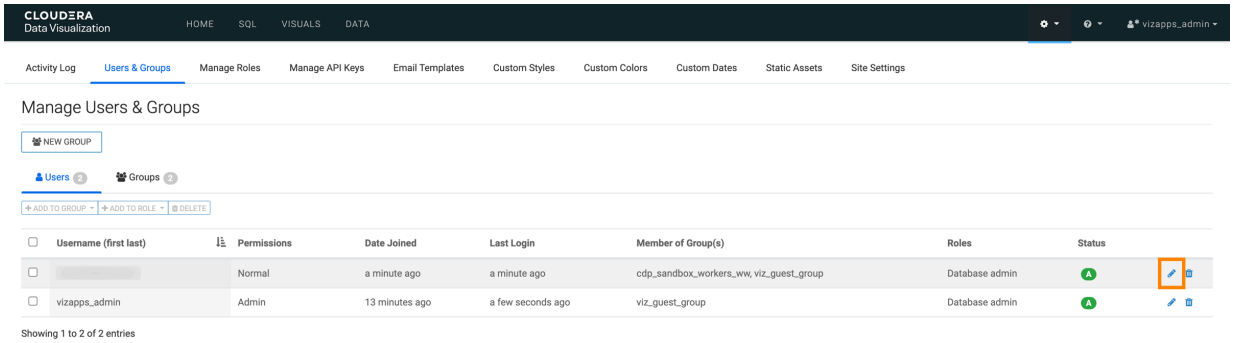

The Edit User modal window appears.

4. Click the Roles tab.

5. Select a new role to assign to the user and click ADD.

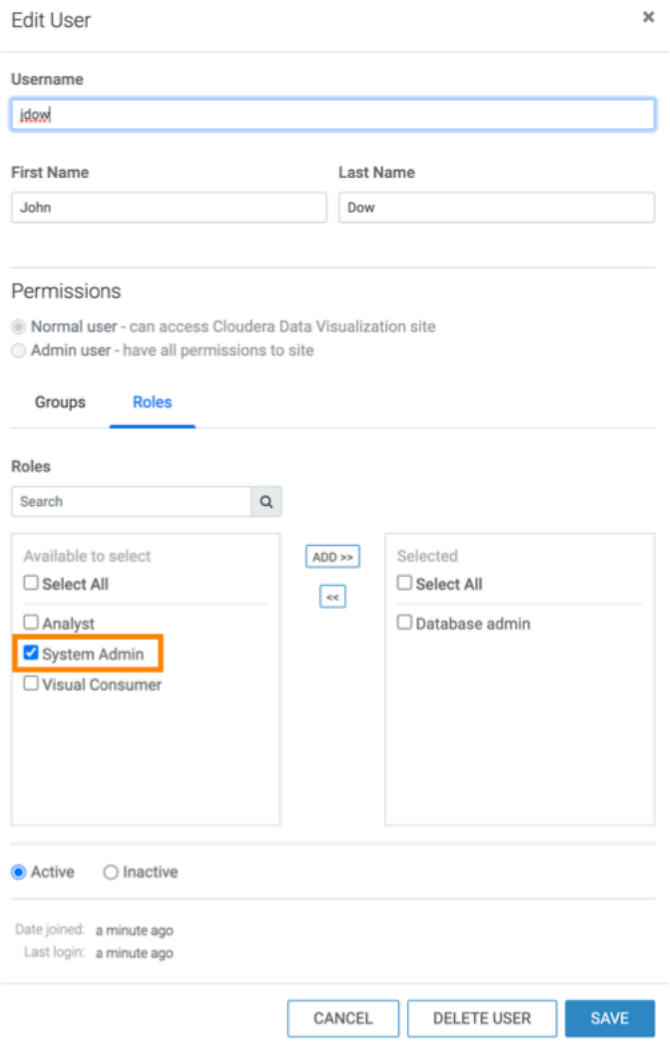

6. After the desired role or roles appear on the right side of the interface, click SAVE.

Results

In the Users & Groups interface, the user now has the new role assignment.

Assigning roles to multiple users

You can add several users to a role at the same time in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

To assign users to existing roles, follow these steps:

Procedure

1. Under the Members tab, in the Users list, click Edit User(s)
2. The Role Assignment modal window for users appears.

3. There are several options for adding a user to a membership:

- *Search*

  If you have a long list of users in the Members section, use the Search box to match user names, select them from the sub-list, then click ADD to move them to the right side of the modal window, and finally click APPLY.

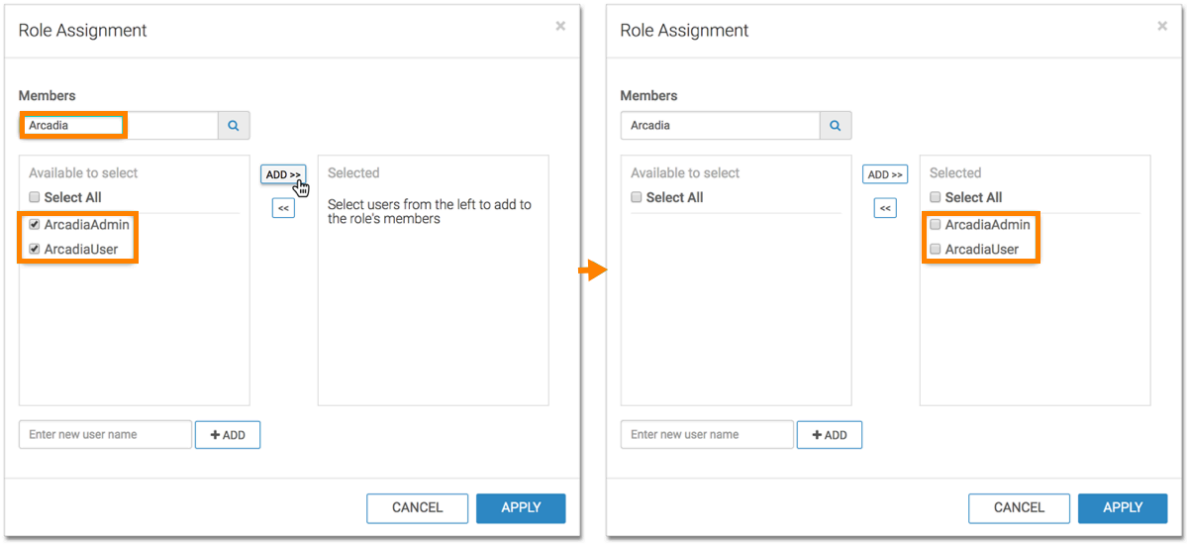

- *Select*

  In the Members section, select the users to assign to the role, click ADD to move them to the right side of the modal window, and finally click APPLY.

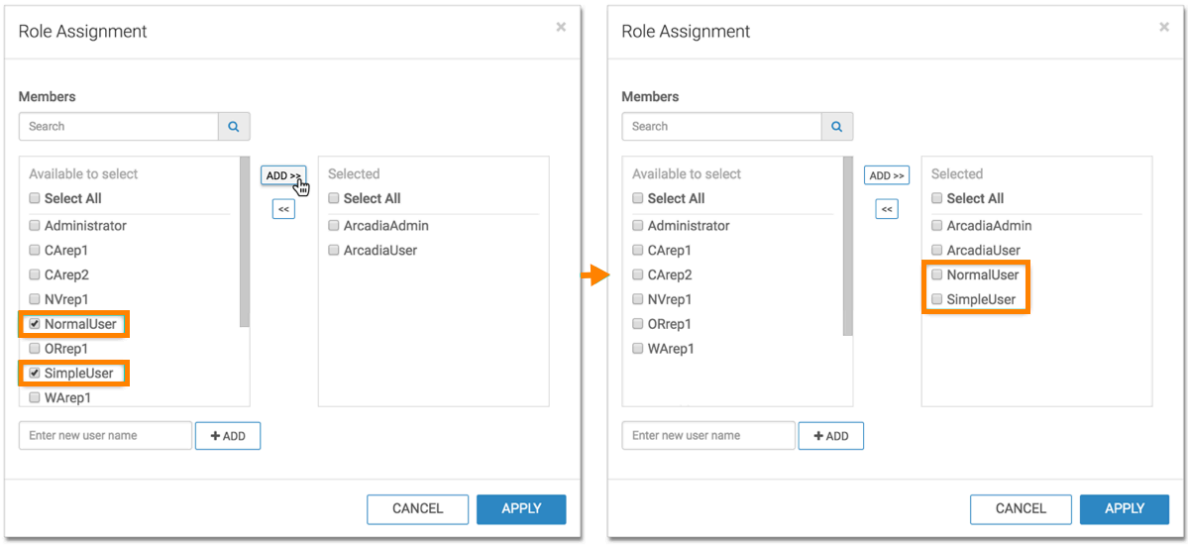

- *Select All*

  To assign all users to Members, select All to get all user names, then click ADD to move them to the right side of the modal window, and finally click APPLY.

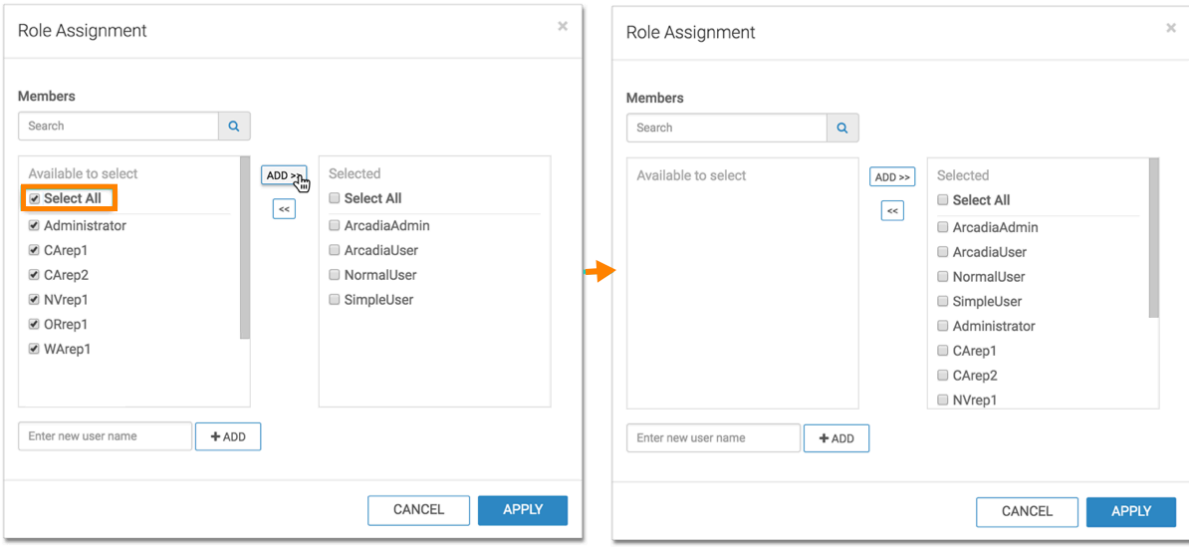

- *Adding Externally Defined Users; LDAP Authentication*

  For convenience, the Role Assignment interface supports adding into the list of assignees usernames that are not stored locally. For example, known usernames available through LDAP authentication may be added in this manner.

  Enter the new user name, and click ADD. After the new user name appears in the Members section, select it and click ADD to move the new user to the right side of the modal window. Click APPLY.

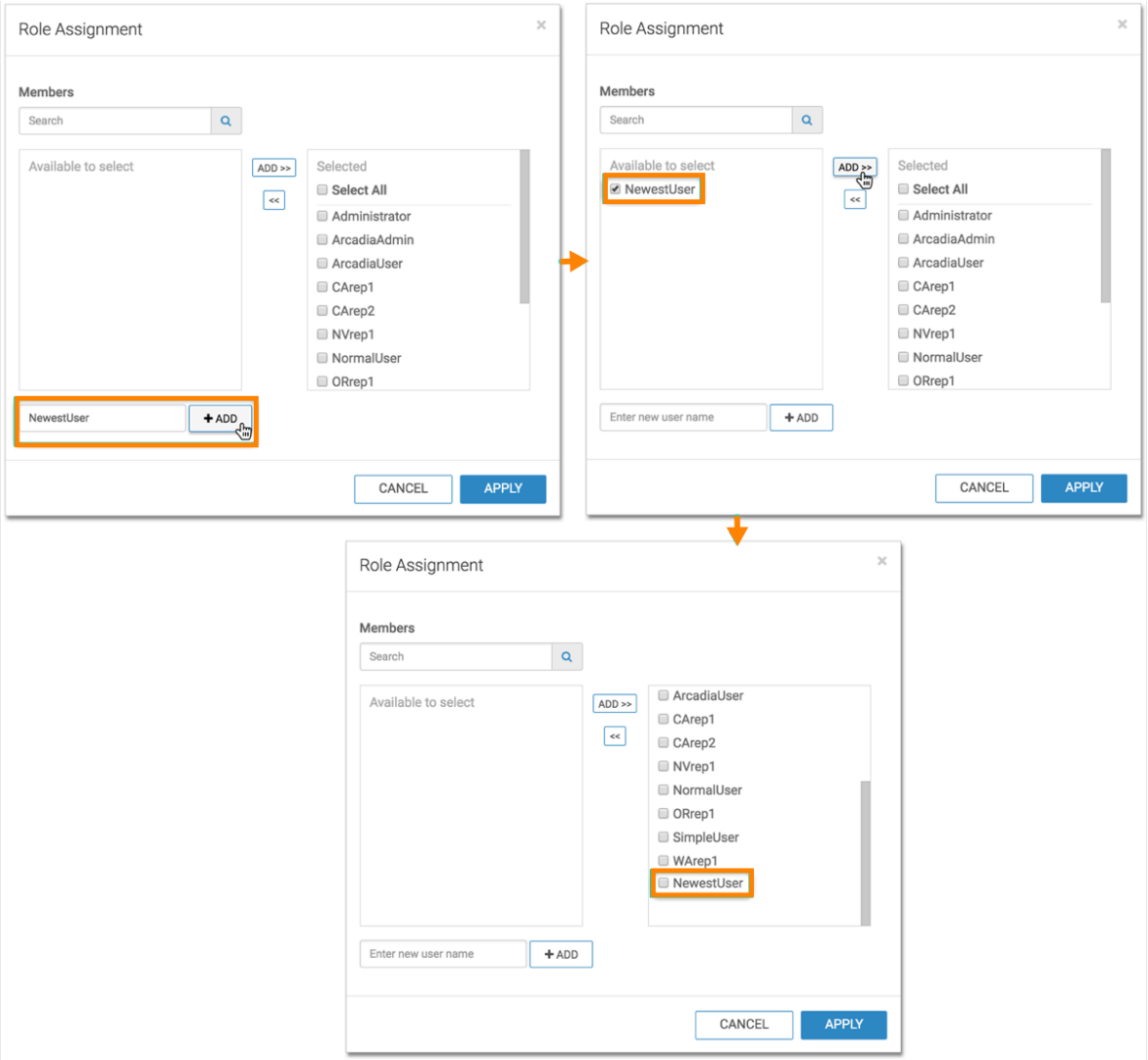

- *Remove*

  To move users out of Members, select the user(s) on the right side panel, then click the Left Arrow, and finally click APPLY.

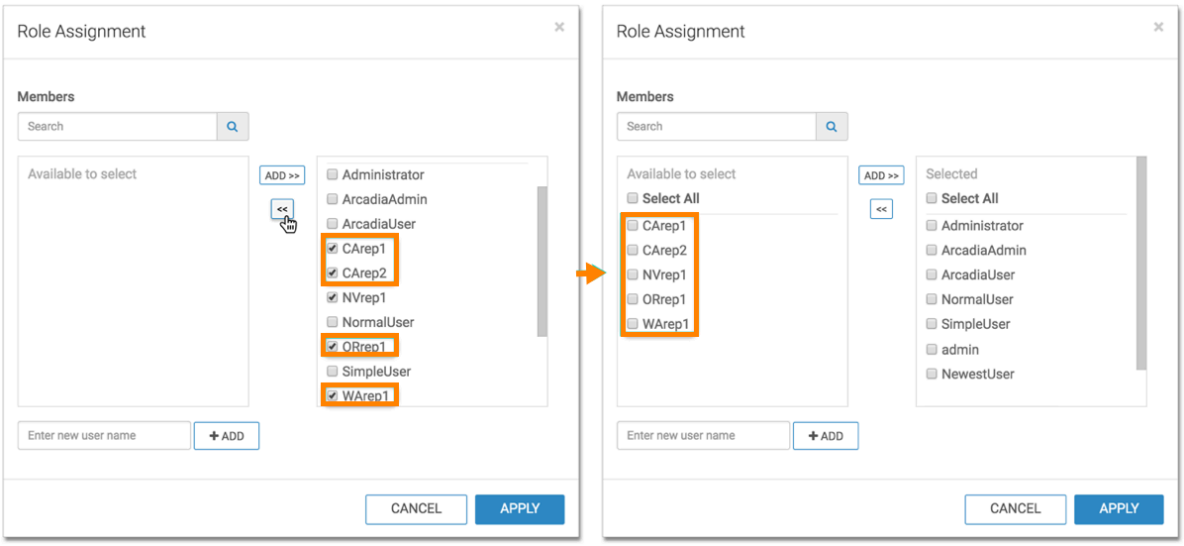

The list of users assigned to the role appears in the Role: Test Role 1 interface, under the Members tab.

4. Click SAVE. A confirmation of role update appears briefly on the screen.

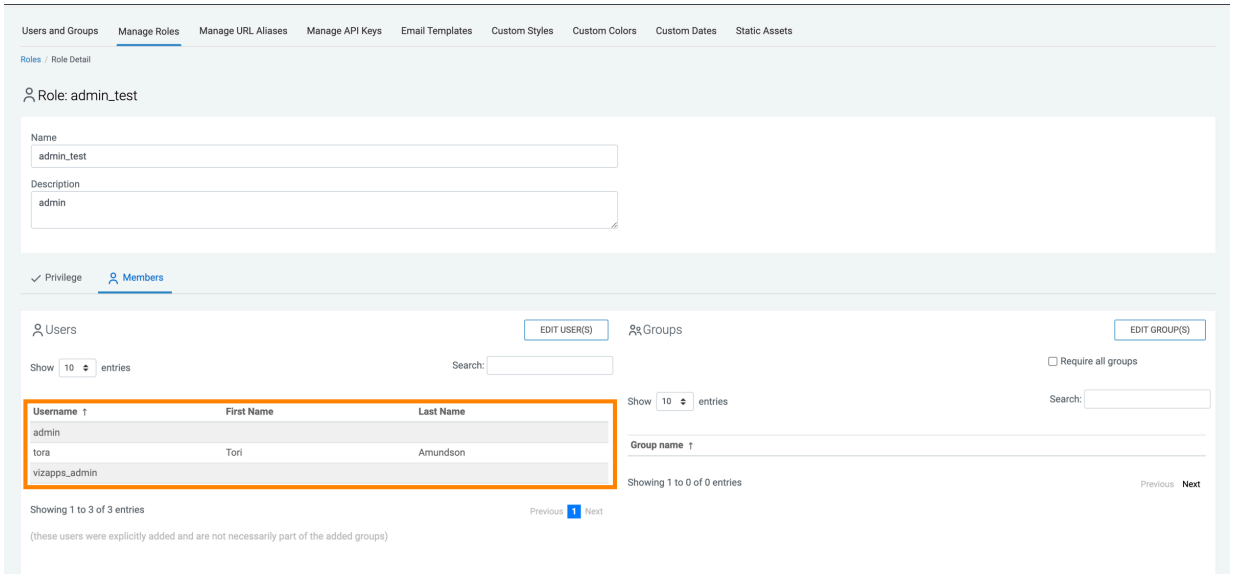

Assigning multiple roles to multiple users

You can add multiple users to multiple role at the same time in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

To assign multiple users to existing roles, follow these steps:

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.

2. Click Users & Groups.

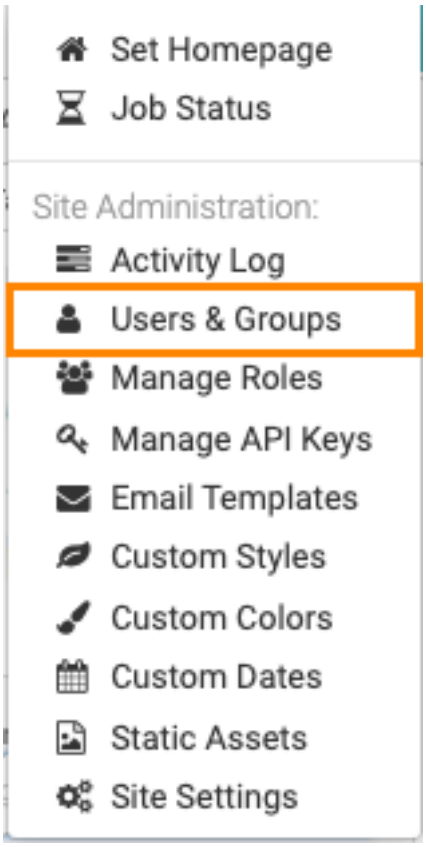

The Manage Users & Groups interface appears, open on the Users tab.

3. Select the users that you want to assign to roles. In this example the following users are used: CArep1, CArep2, NVrep1, ORrep1, and WArep1.

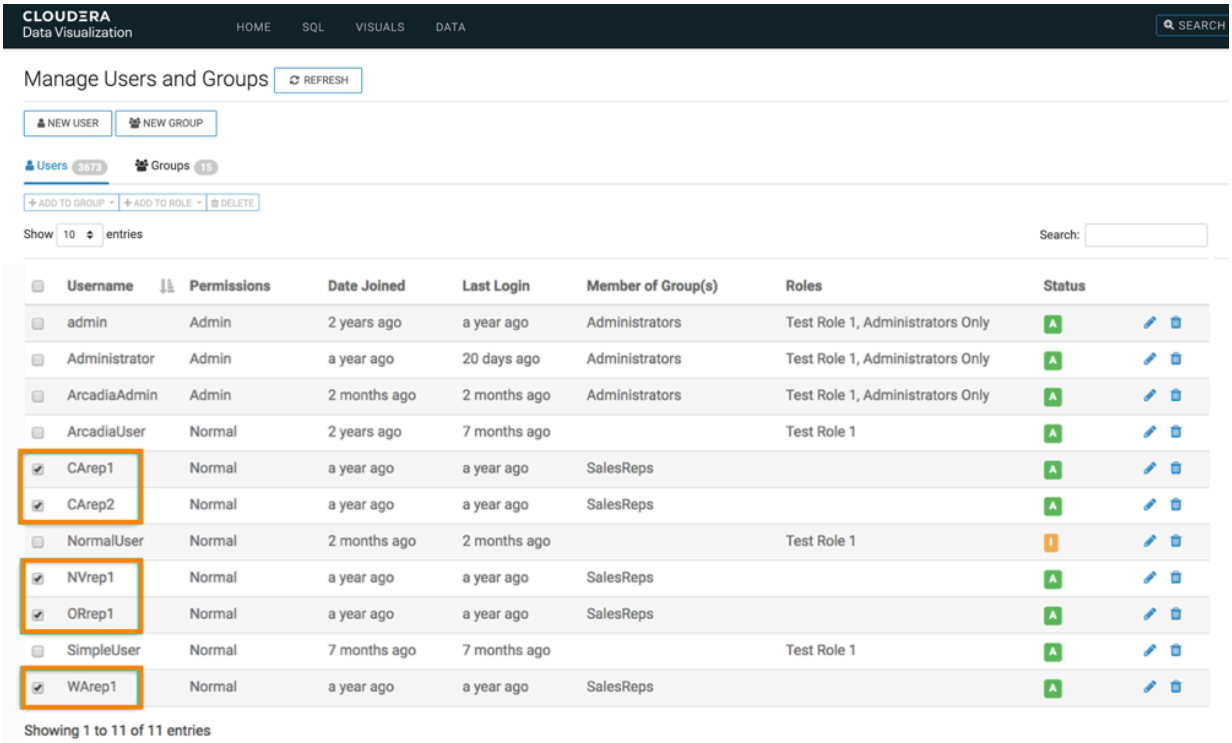

Username	Permissions	Date Joined	Last Login	Member of Group(s)	Roles	Status
admin	Admin	2 years ago	a year ago	Administrators	Test Role 1, Administrators Only	A
Administrator	Admin	a year ago	20 days ago	Administrators	Test Role 1, Administrators Only	A
ArcadiaAdmin	Admin	2 months ago	2 months ago	Administrators	Test Role 1, Administrators Only	A
ArcadiaUser	Normal	2 years ago	7 months ago		Test Role 1	A
CArep1	Normal	a year ago	a year ago	SalesReps		A
CArep2	Normal	a year ago	a year ago	SalesReps		A
NormalUser	Normal	2 months ago	2 months ago		Test Role 1	I
NVrep1	Normal	a year ago	a year ago	SalesReps		A
ORrep1	Normal	a year ago	a year ago	SalesReps		A
SimpleUser	Normal	7 months ago	7 months ago		Test Role 1	A
WArep1	Normal	a year ago	a year ago	SalesReps		A

4. Click ADD TO ROLES.

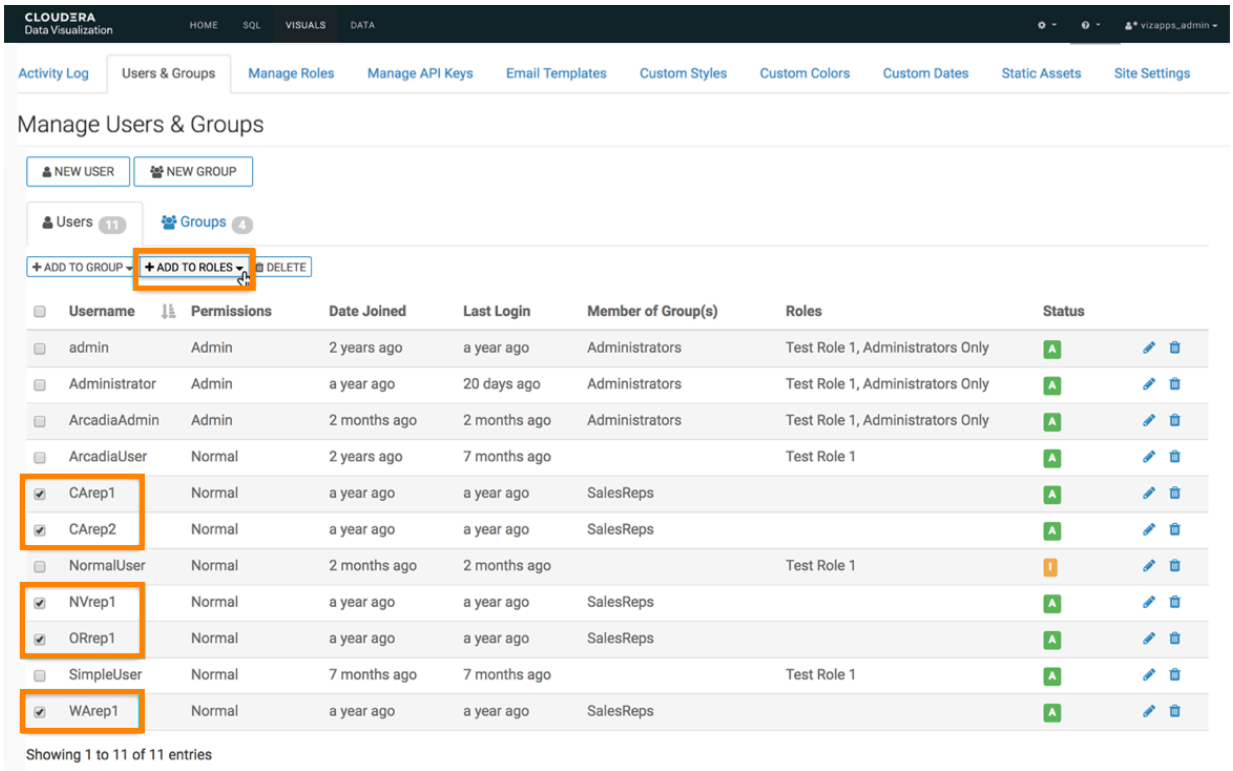

Username	Permissions	Date Joined	Last Login	Member of Group(s)	Roles	Status
admin	Admin	2 years ago	a year ago	Administrators	Test Role 1, Administrators Only	A
Administrator	Admin	a year ago	20 days ago	Administrators	Test Role 1, Administrators Only	A
ArcadiaAdmin	Admin	2 months ago	2 months ago	Administrators	Test Role 1, Administrators Only	A
ArcadiaUser	Normal	2 years ago	7 months ago		Test Role 1	A
CArep1	Normal	a year ago	a year ago	SalesReps		A
CArep2	Normal	a year ago	a year ago	SalesReps		A
NormalUser	Normal	2 months ago	2 months ago		Test Role 1	I
NVrep1	Normal	a year ago	a year ago	SalesReps		A
ORrep1	Normal	a year ago	a year ago	SalesReps		A
SimpleUser	Normal	7 months ago	7 months ago		Test Role 1	A
WArep1	Normal	a year ago	a year ago	SalesReps		A

5. In the drop-down menu, select the roles you plan to assign to the selected users, and click SAVE.

   In this example, Test Role 1 and View Only are used.

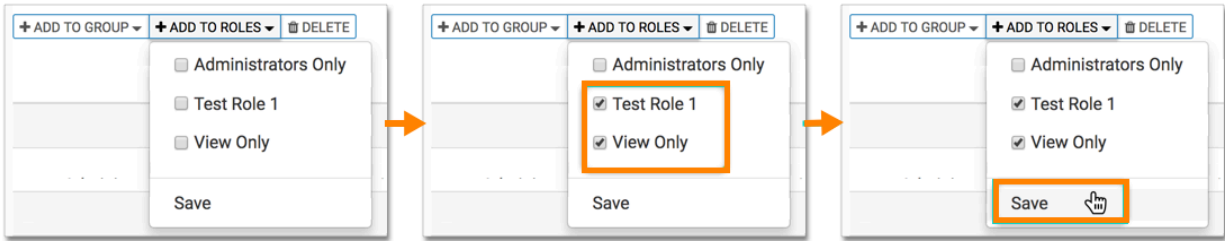

Results

Note the changes to the information in the Users interface.

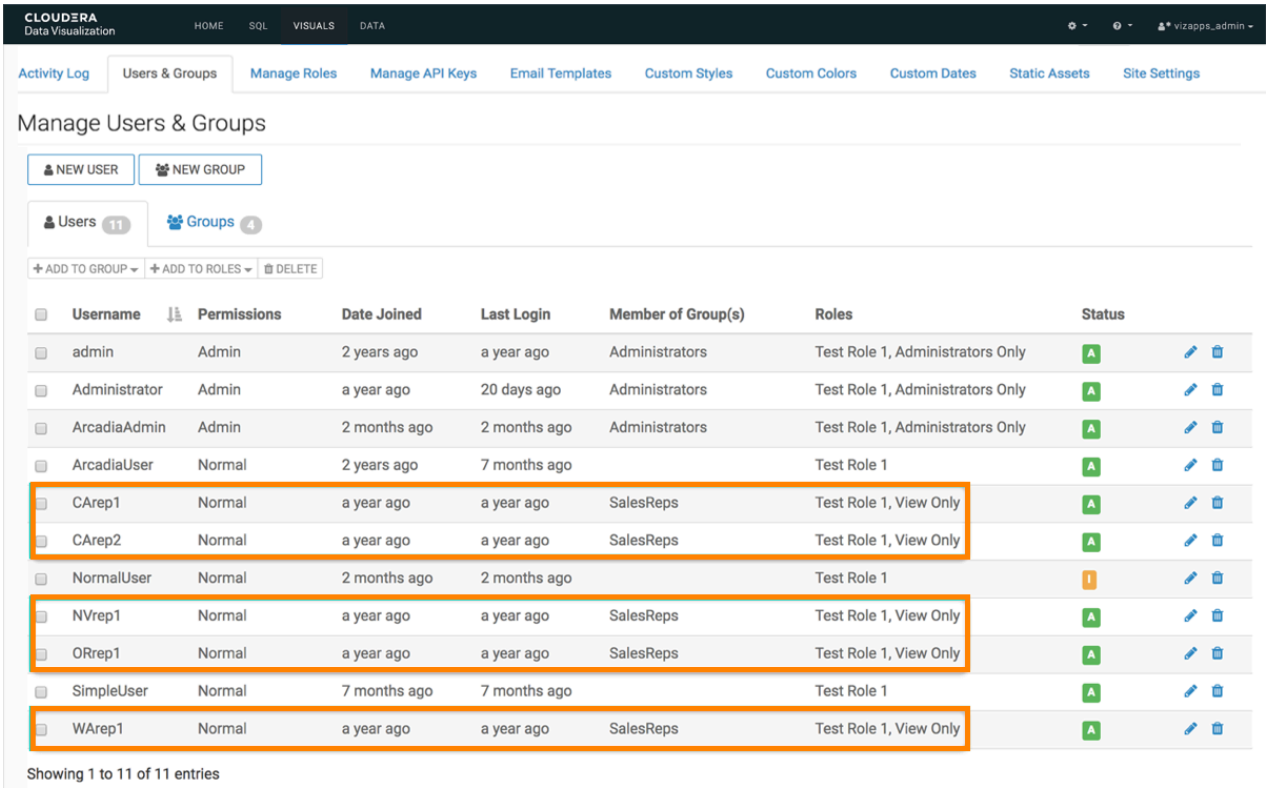

Username	Permissions	Date Joined	Last Login	Member of Group(s)	Roles	Status
admin	Admin	2 years ago	a year ago	Administrators	Test Role 1, Administrators Only	A
Administrator	Admin	a year ago	20 days ago	Administrators	Test Role 1, Administrators Only	A
ArcadiaAdmin	Admin	2 months ago	2 months ago	Administrators	Test Role 1, Administrators Only	A
ArcadiaUser	Normal	2 years ago	7 months ago		Test Role 1	A
CArep1	Normal	a year ago	a year ago	SalesReps	Test Role 1, View Only	A
CArep2	Normal	a year ago	a year ago	SalesReps	Test Role 1, View Only	A
NormalUser	Normal	2 months ago	2 months ago		Test Role 1	I
NVrep1	Normal	a year ago	a year ago	SalesReps	Test Role 1, View Only	A
ORrep1	Normal	a year ago	a year ago	SalesReps	Test Role 1, View Only	A
SimpleUser	Normal	7 months ago	7 months ago		Test Role 1	A
WArep1	Normal	a year ago	a year ago	SalesReps	Test Role 1, View Only	A

Deleting user accounts

You can delete both standard user accounts, and administrator accounts in Cloudera Data Visualization.

Note: This feature is only available to users with administrative privileges.

An administrator can delete user accounts in the following ways:

Deleting individual user accounts

You can delete user accounts in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

The following steps demonstrate how to delete an individual Cloudera Data Visualization user account.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.

2. Click Users & Groups.

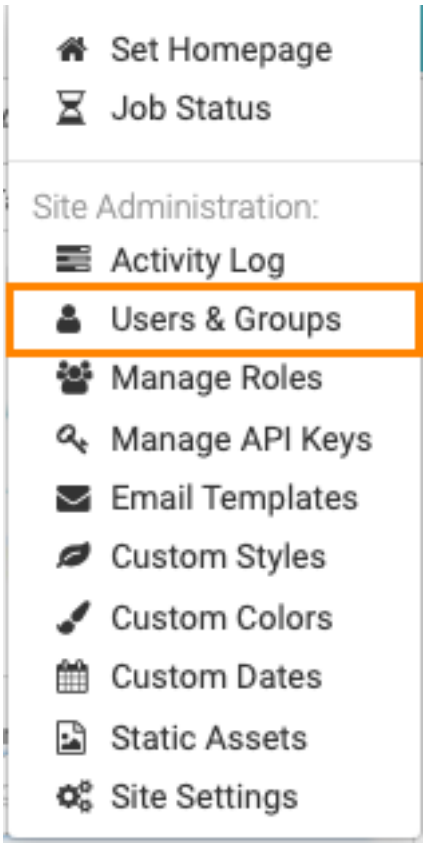

The Manage Users & Groups interface appears, open on the Users tab.

3. In the list of users, do one of the following:

- Find the username of the account you want to delete, and click the Trashcan icon.

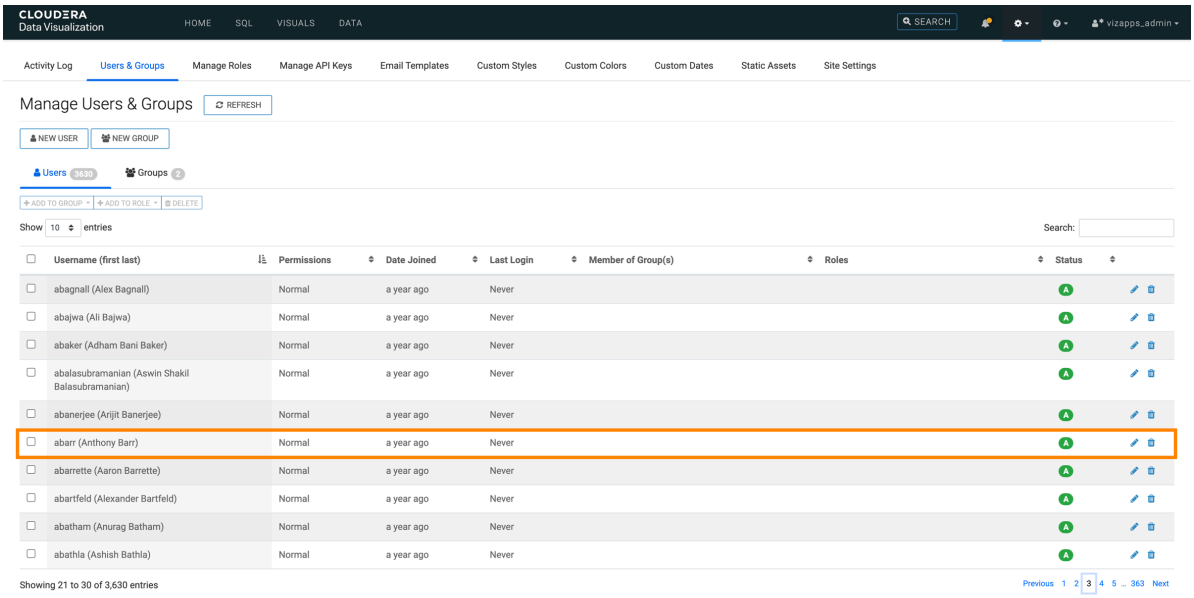

- Click the Username of the account you want to delete, or the edit (Pencil) icon.

The Edit User modal window is displayed.

4. Click DELETE USER.

5. In the confirmation modal window, click DELETE.

Results

After the operation succeeds, you will see the Users screen. The Username of the account you just deleted is no longer on the list.

Deleting several user accounts

You can delete user accounts in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

The following steps demonstrate how to delete several Cloudera Data Visualization user accounts at the same time.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Click Users & Groups.

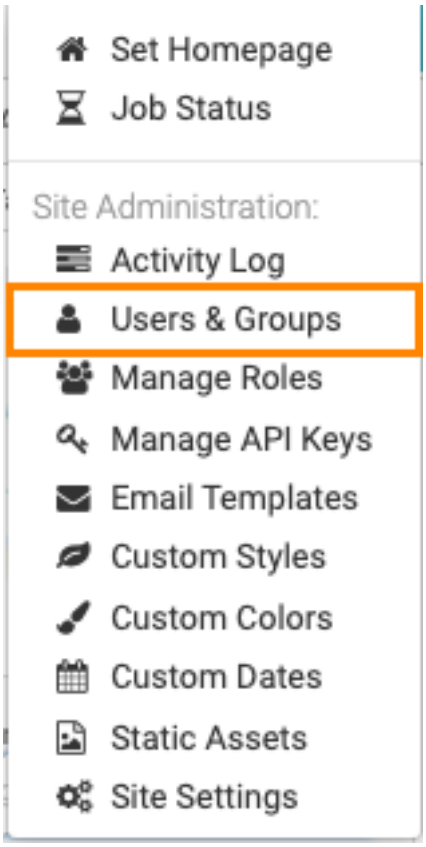

The Manage Users & Groups interface appears, open on the Users tab.

3. In the list of users, click the checkmarks next to the usernames you plan to delete.

4. Click DELET.

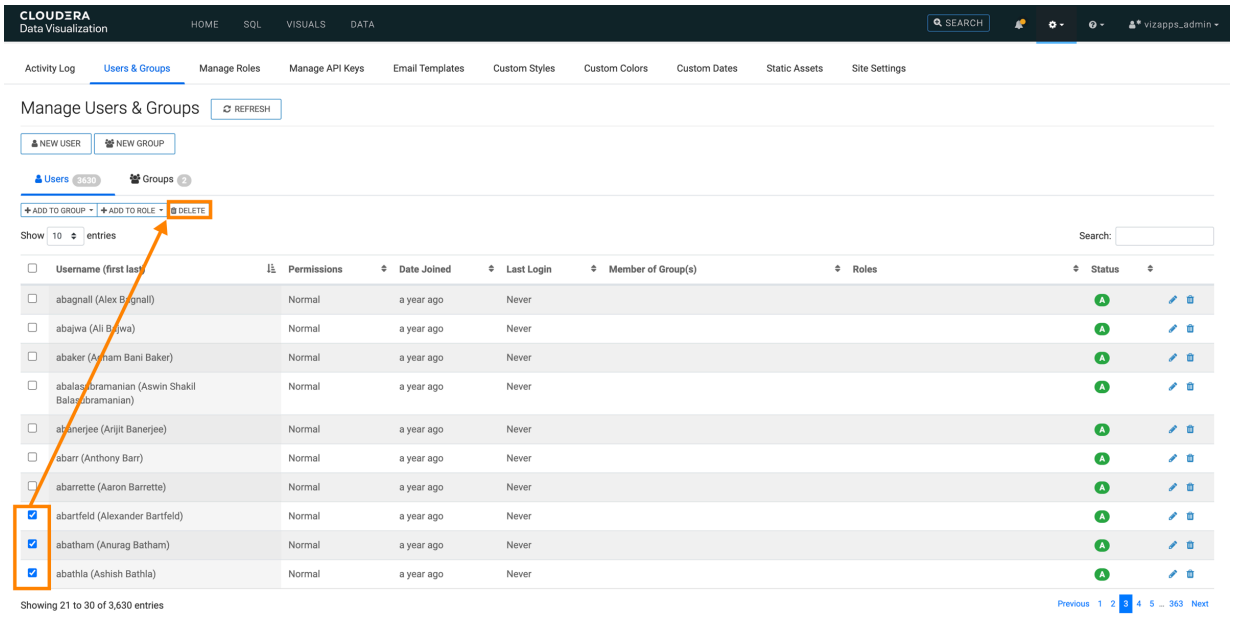

5. In the confirmation modal window, click DELETE.

Creating new user groups

You can create new user groups in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

The following steps demonstrate how to create a new user group.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.

2. Click Users & Groups.

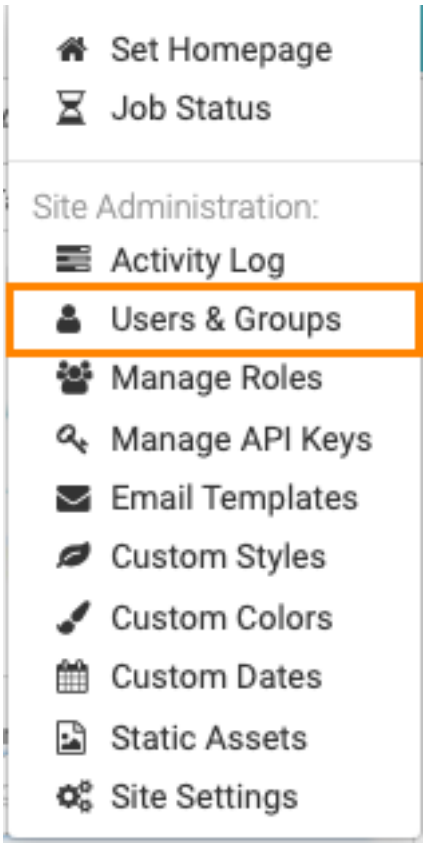

The Manage Users & Groups interface appears, open on the Users tab.

3. Click New Group.

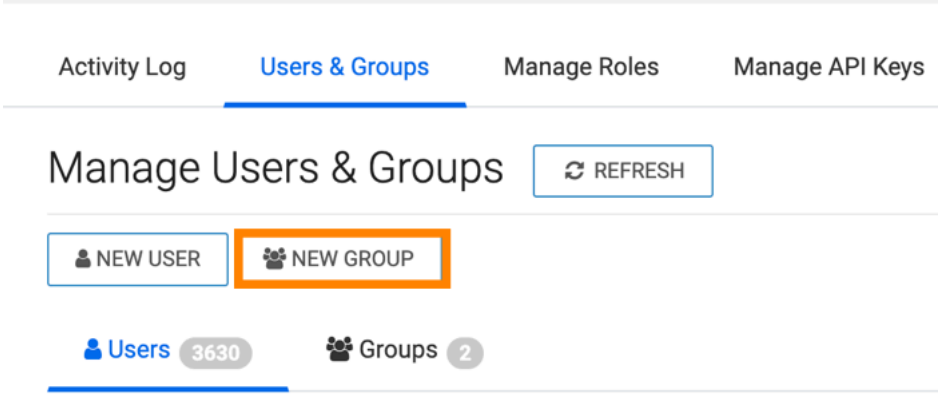

The New Group modal window appears.

4. In the New Group modal window:
   - Enter Group Name.
   - [Optional] Under Members on the Users tab, use the Search box to find users, and assign the new user to existing groups by selecting user groups on the left list, and then clicking ADD to move them to the right.

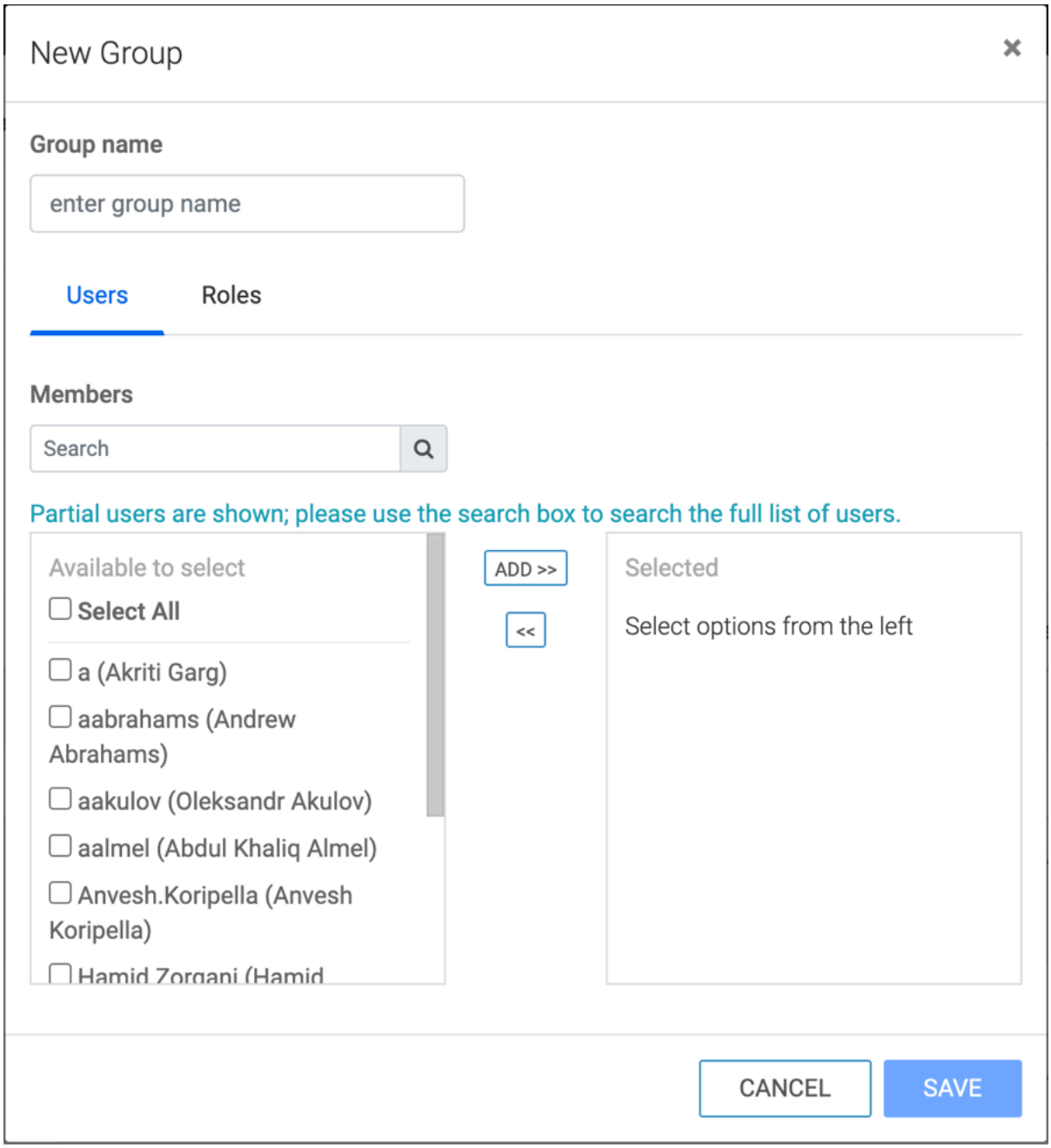

5. Click SAVE.

Results

After the operation succeeds, you can check the Users interface and find that the group assignment applies to the users just assigned to the new group.

Editing user groups

You can edit the user groups in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

When editing a user group, you can change its name, and the users assigned to it.

The following steps demonstrate how to edit user groups.

Procedure

1. Click the Gear icon on the main navigation bar to open the Administration menu.
2. Click Users & Groups.

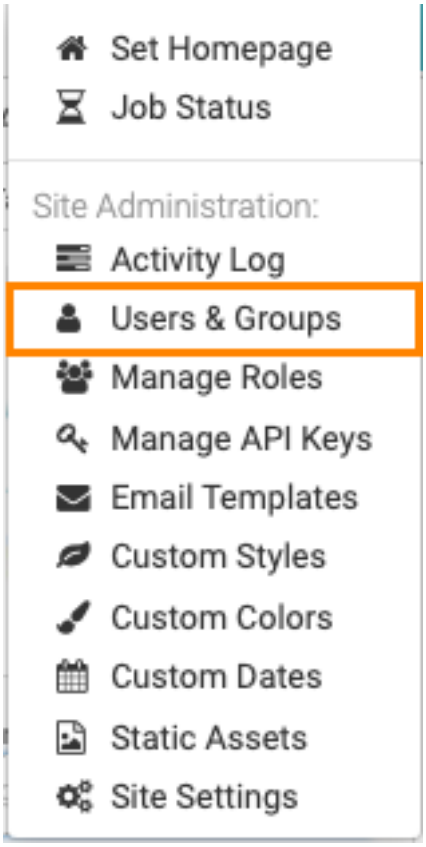

   The Manage Users & Groups interface appears, open on the Users tab.
3. Click the Groups tab.
4. In the list of user groups, click the name of the user group to edit, or the Pencil icon on that row.
5. The Edit Group modal window appears.

6. To change the name of the user group, simply change the text in the Group name field, and click Save.

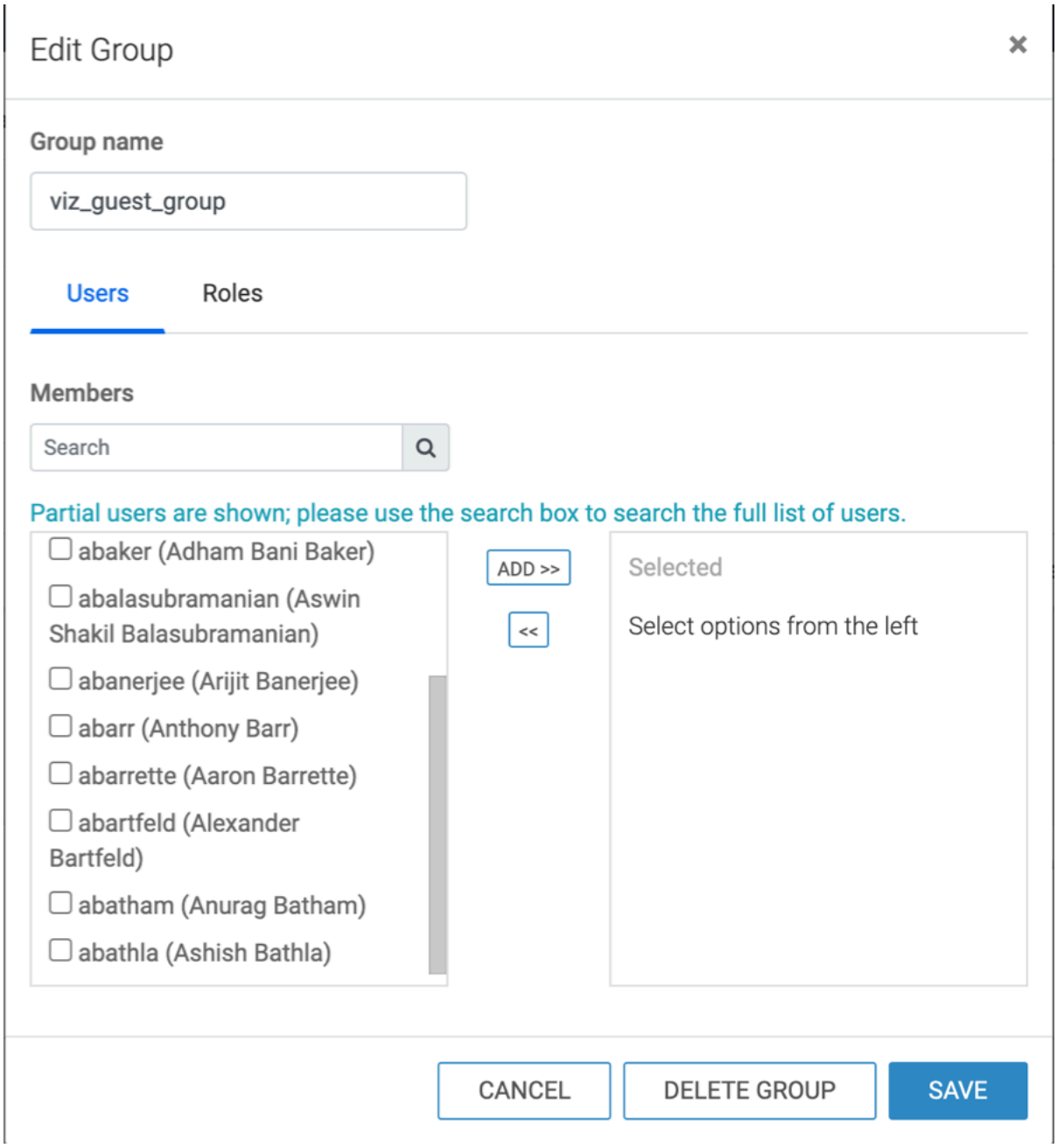

Adding users to user groups

You can add users to groups in Cloudera Data Visualization.

Note: This feature is only available to users with administrative privileges.

Administrators can add users to groups in several ways:

Adding a user to multiple groups; user modal

There are several options for adding a user to user groups when working in the New Group or Edit Group modal window.

Procedure

Choose from the below options:

- If you have a long list of groups in the Group Membership section, use the Search box to match group names, select groups from the sub-list, and then click ADD to move them to the right side of the modal window. Click SAVE.

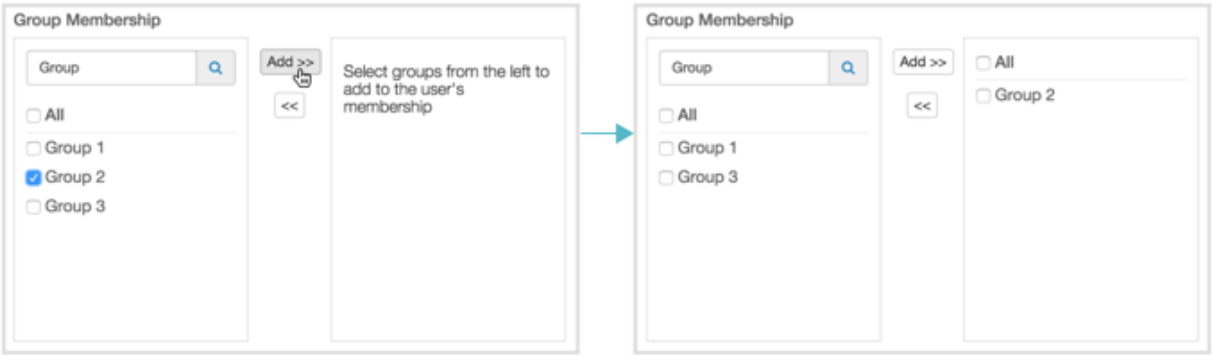

- In the Group Membership section, select the group to assign to the user and click ADD to move them to the right side of the modal window. Click SAVE.

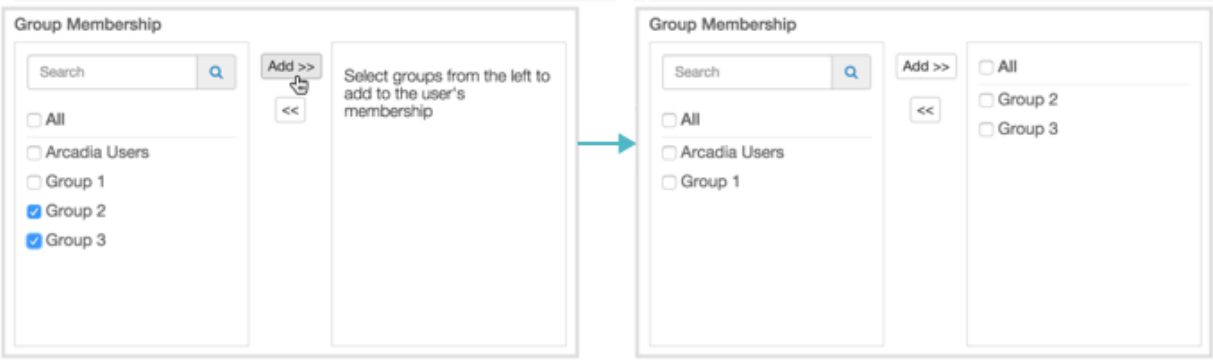

- To assign the user to all user groups, select All to get all the groups, and then click ADD to move them to the right side of the modal window. Click SAVE.

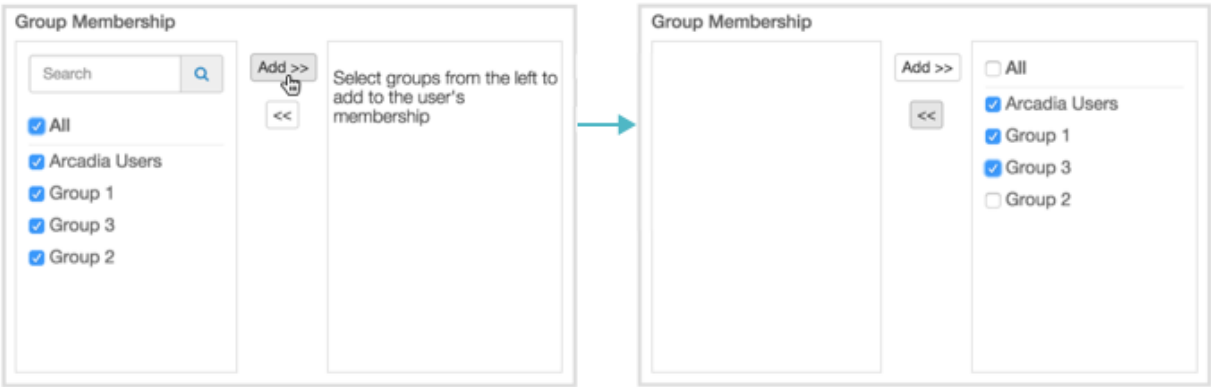

- To move the user out of one or more groups, select the groups on the right side panel, and then click the Left Arrow. Click SAVE.

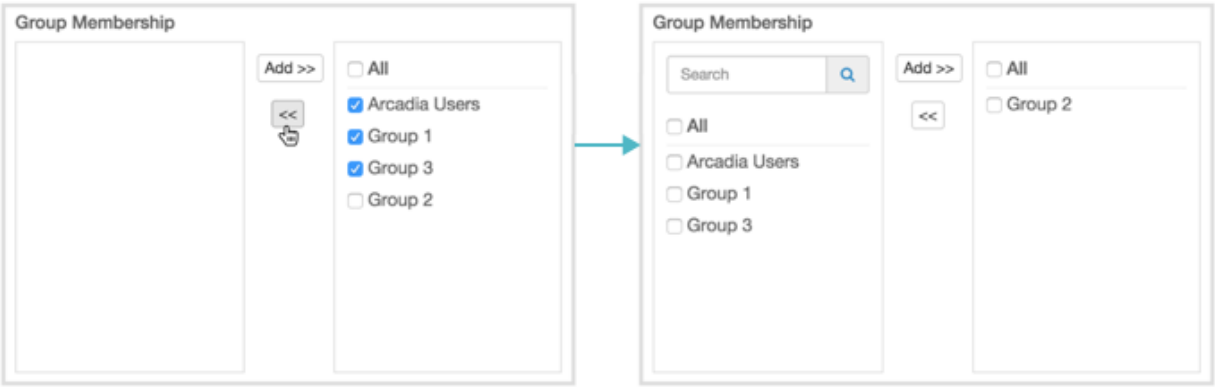

Adding multiple users to a single group; group modal

There are several options for adding users into a specific user group when working in the New Group or Edit Group modal window.

Procedure

Choose from the below options:

- If you have a long list of users in the Members section, use the Search box to match usernames, select users from the sub-list, and then click ADD to move them to the right side of the modal window. Click SAVE.
- In the Members section, select the users to assign to the group and click ADD to move them to the right side of the modal window. Click SAVE.

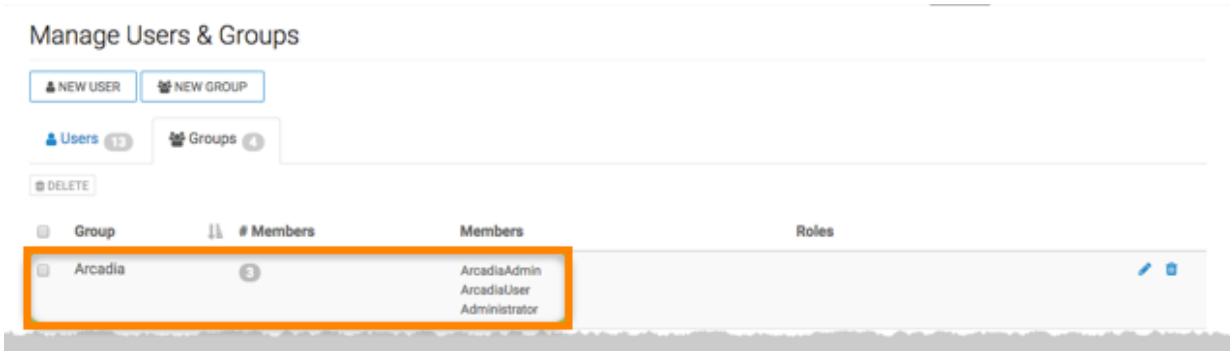

- To add all users to a particular group, select All, and then click ADD. Click SAVE.

- To move users out of the group, select the users on the right side panel, and then click the Left Arrow. Click SAVE.

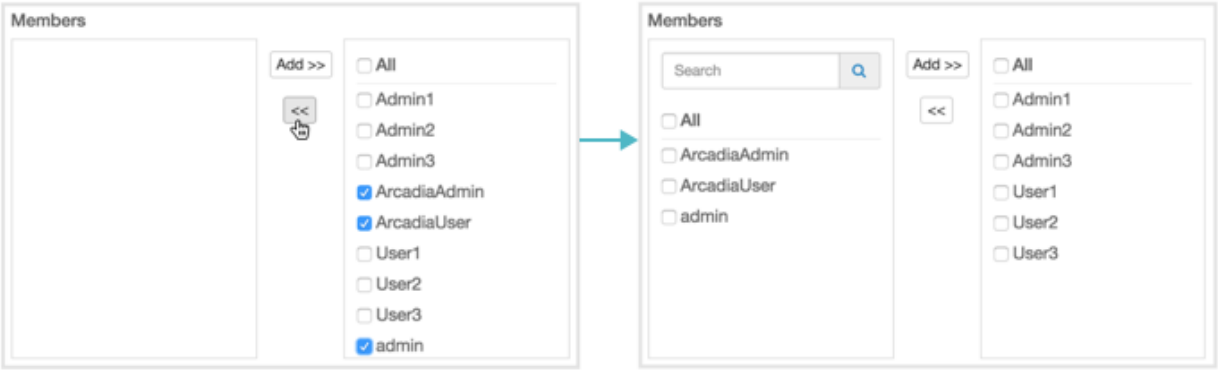

Adding multiple users to multiple groups; users tab

When working in the Users tab, you can add multiple users to one or more user groups.

Procedure

1. Select the users that you want to assign to groups.
2. Click the Down Arrow on the right of Add to Group, and select the groups for the users.
3. Click SAVE.

Assigning roles to user groups

About this task

The following steps demonstrate how to add user groups to a role.

Procedure

1. Navigate to the Members tab of the Role Detail interface.

   The Require all groups option ensures that only members of ALL groups listed in the role membership fields have the role's defined access. It determines whether members must belong to all listed groups or any one of them to access the role's privileges.

   Note:

   - If Require all groups is checked, only users belonging to both groups will receive the role's privileges.
   - Exceptions to group membership include explicitly named users and users imported through Ranger groups.

2. Click EDIT GROUP(S).

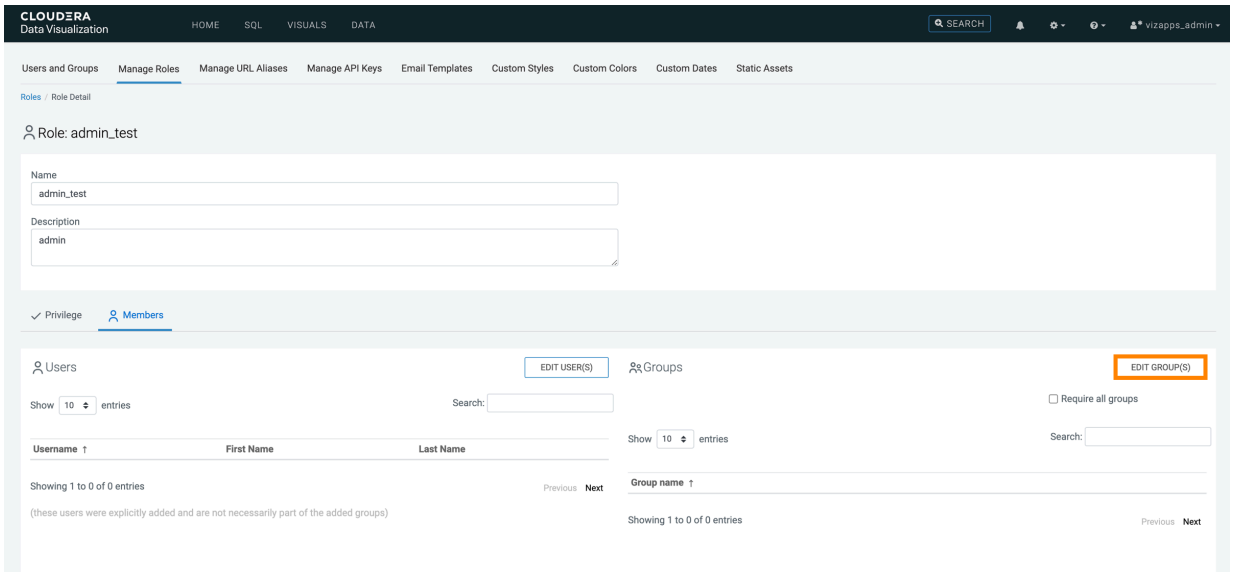

The Role Assignment modal window is displayed.

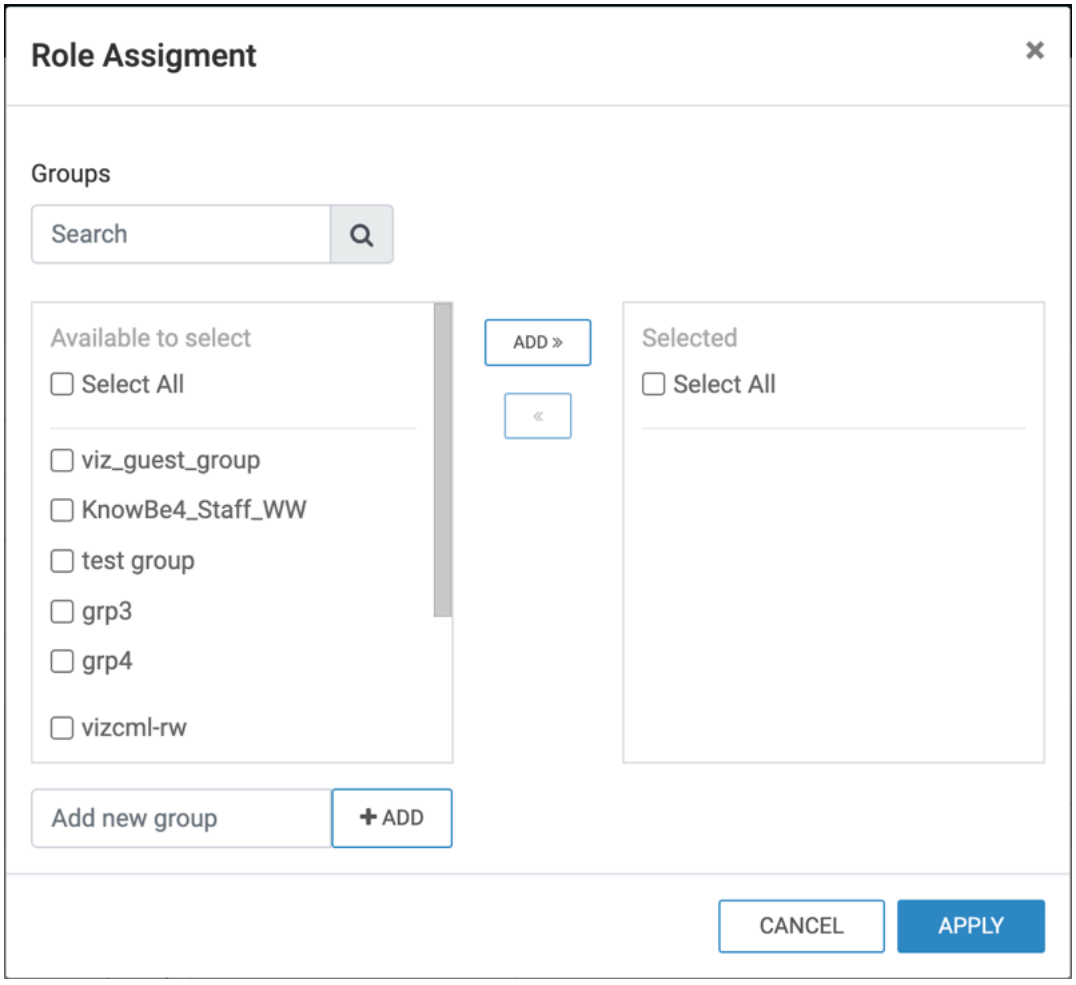

3. There are several options to add groups to role membership:
   - Search

     If you have a long list of groups in the Groups section, use the Search box to find specific group names, select them from the sub-list, and click ADD>> to move them to the right side of the modal window.

     When you are ready, click APPLY.
   - Select

     In the Groups section, select the groups to assign to the role and click ADD>> to move them to the right side of the modal window.

     When you are ready, click APPLY.
   - Select All

     To assign all groups to Groups, mark Select All and click ADD>> to move them to the right side of the modal window.

     When you are ready, click APPLY.
   - Adding externally defined users (LDAP authentication)

     The Role Assignment modal supports adding user groups to the list of assignees that are not stored locally. For example, you can add known groups available through LDAP authentication using this method.

     Enter the new group name and click ADD>>. After the new group name appears in the Members section, select it and click ADD>> to move the new group to the right side of the modal window.

     When you are ready, click APPLY.
   - Remove

     To remove users from the Members list, select them on the right side panel and click <<.

     When you are ready, click APPLY.
4. Click APPLY CHANGES.

   The Users list is updated.

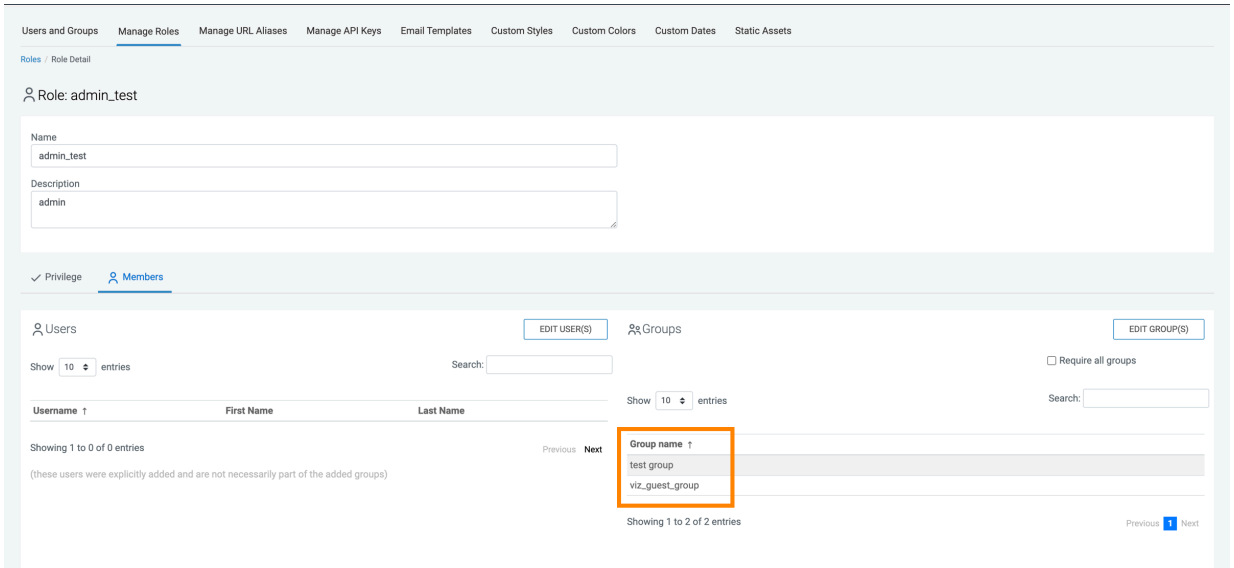

Deleting user groups

You can delete user groups in Cloudera Data Visualization.

Before you begin

You must have administrative privileges to perform this action.

About this task

The following steps demonstrate how to delete a previously defined user group.

Procedure

1. Click the Groups tab.

2. There are several ways for deleting user groups:

   Method 1:

   a. Click the boxes next to the names of the user groups to select them.
   b. Click DELETE.
   c. In the confirmation window modal, click DELETE.

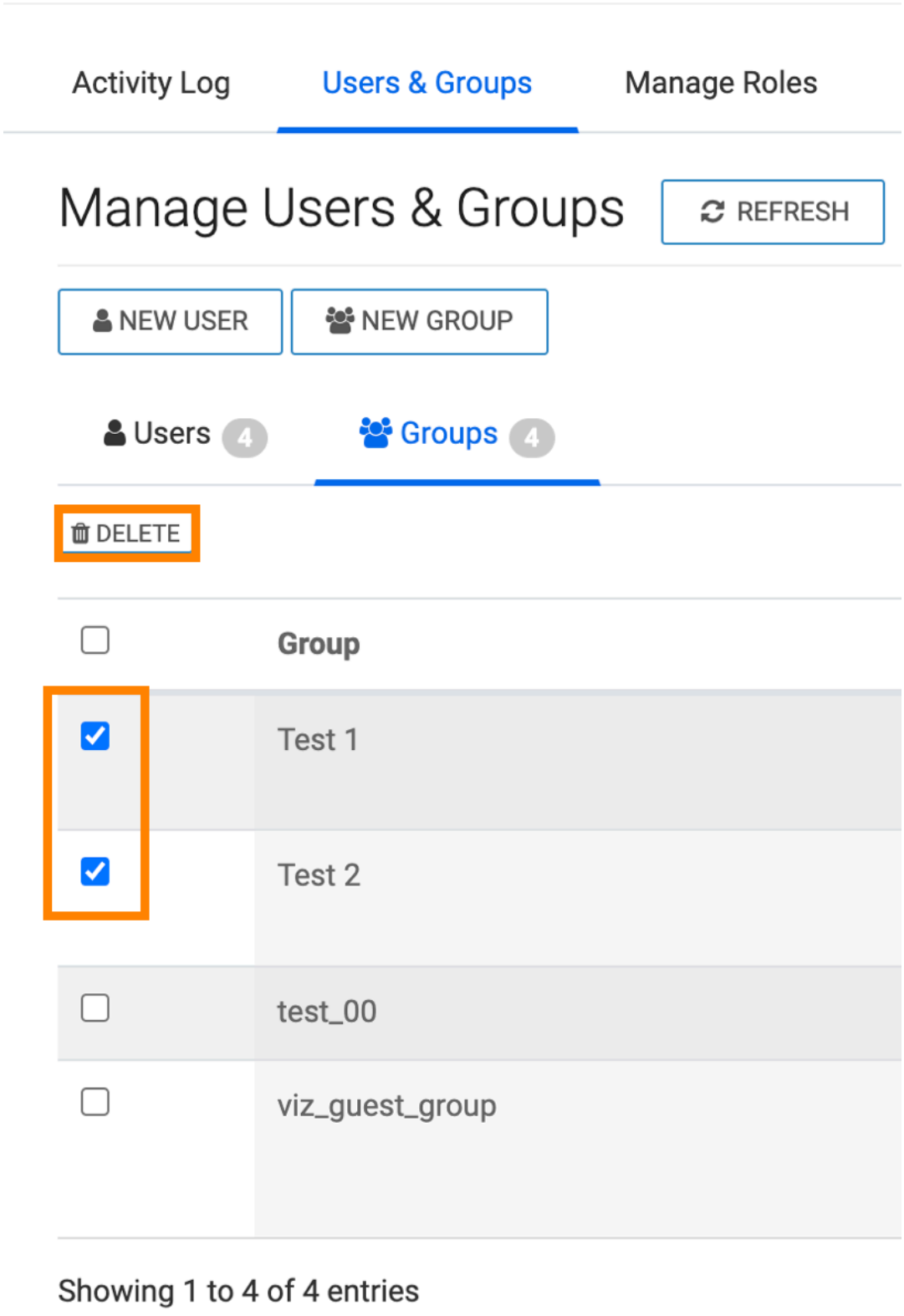

Method 2:

a. Click on the select box next to the table header Group.

This Selects all user groups.

b. Click Delete.
c. In the confirmation window modal, click Delete.

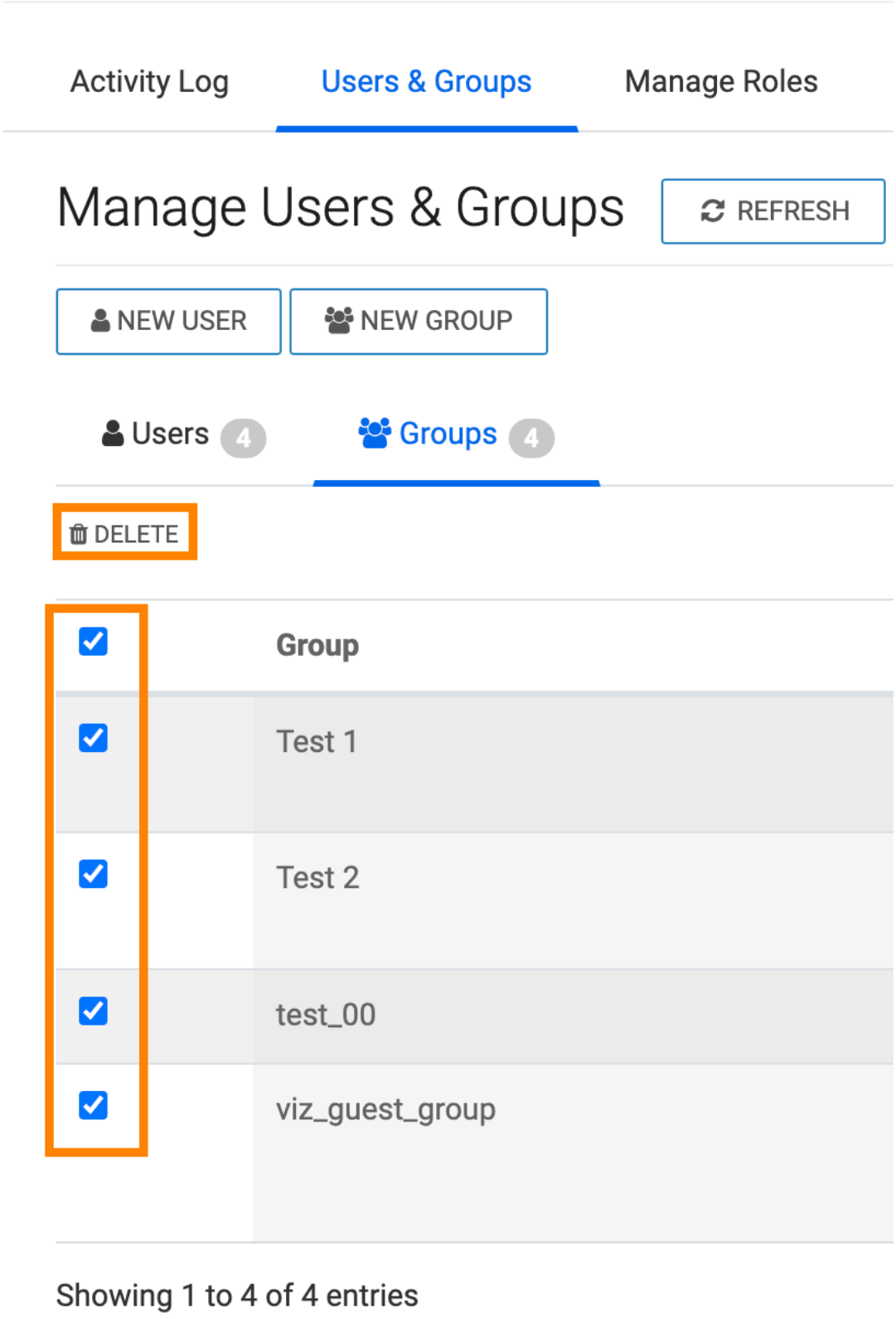

Method 3:

a. Click the name of the user group.

Alternatively, click the Pencil icon.

b. In the Edit Group window modal, click DELETE GROUP.

c. In the confirmation window modal, click DELETE.

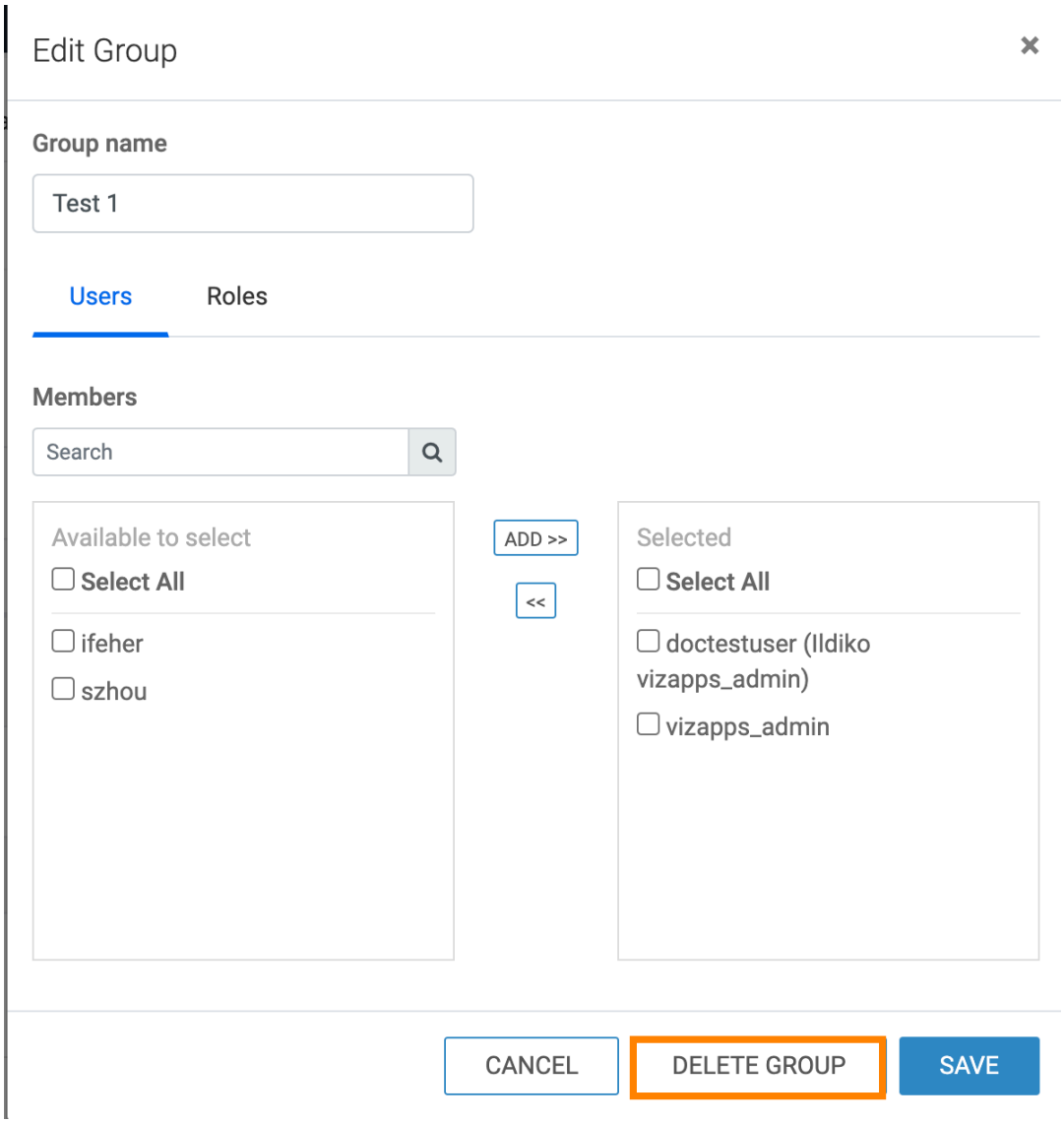

Method 4:

a. Click the Trash can icon that corresponds to the user group.

b. In the confirmation window modal, click DELETE.

Cloudera Data Visualization 7.2.9

Sheets

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a new sheet 4

Adding visuals to a sheet 4

Adding a filter to a sheet 7

Renaming a sheet 11

Reordering sheets 12

Reordering sheet IDs to match positions 15

Cloning a sheet 16
- Cloning an internal sheet in a dashboard 16
- Cloning an external sheet in a dashboard 18

Downloading a sheet 21
- Downloading dashboard sheets as PNG files 21
- Downloading all sheets as a PDF File 23
- Downloading current sheet as a PDF file 27
- Downloading current sheet with one visual per page as a PDF file 28

Emailing sheets 32

Deleting a sheet 34

Creating a new sheet

Cloudera Data Visualization enables you to create and customize sheets within your dashboards.

Procedure

1. Open a dashboard of your choice in Edit mode.

   If you are creating a new dashboard, the default single sheet already contains a basic table visual based on the initial dataset.

   Note: Save the dashboard before proceed with adding a new sheet.

2. Navigate to the bottom left corner of the interface and click the icon next to the tab of the existing sheet.

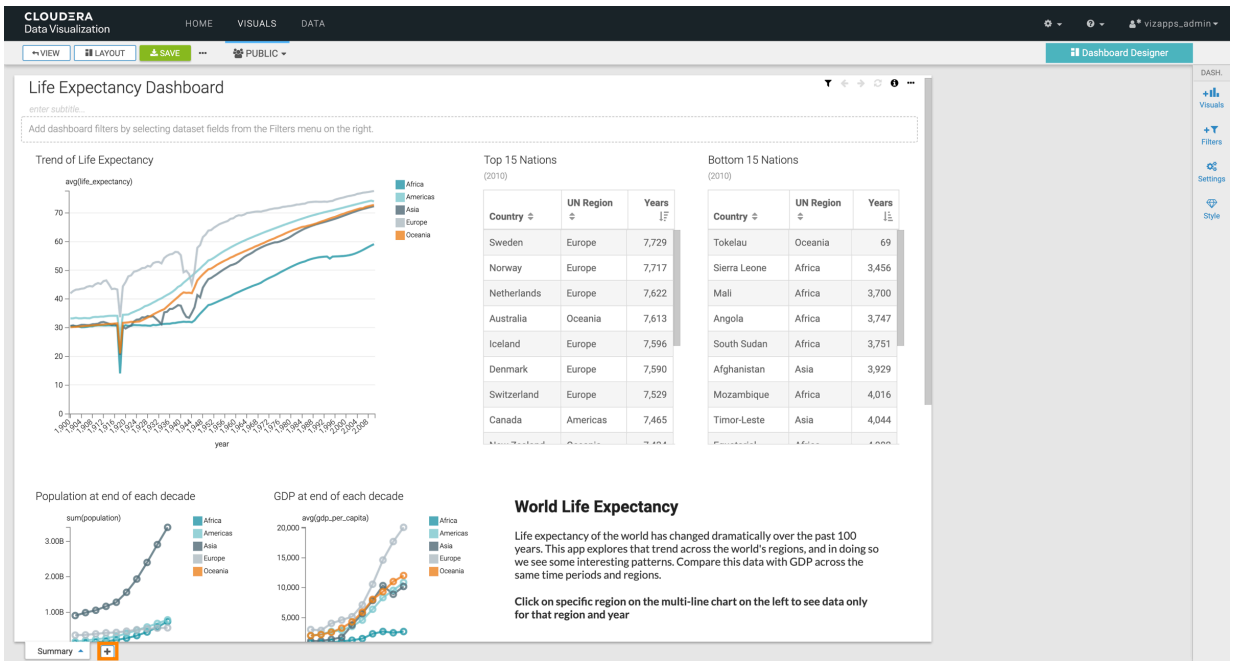

   Cloudera Data Visualization generates a new, empty sheet, titled Sheet 2. The sheet title is automatically highlighted, allowing you to assign a more descriptive name.

3. Change the name of the sheet. In this example, Sheet 2 is renamed to High GDP.

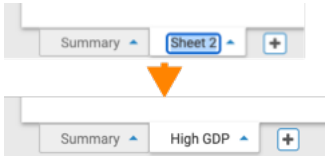

4. Click SAVE to ensure that all changes are saved to the dashboard.

What to do next

Start building out the new sheet by adding filters and visuals.

Adding visuals to a sheet

Cloudera Data Visualization enables you to add visuals to a sheet.

Procedure

1. Open a dashboard of your choice in Edit mode.
2. Choose the specific sheet in the dashboard where you want to add visuals.

   Note: If you are working with a new dashboard, it has a single sheet by default.

   This examples works with a sheet called High GDP.
3. In the Visual Builder side menu bar on the right, select Visuals.

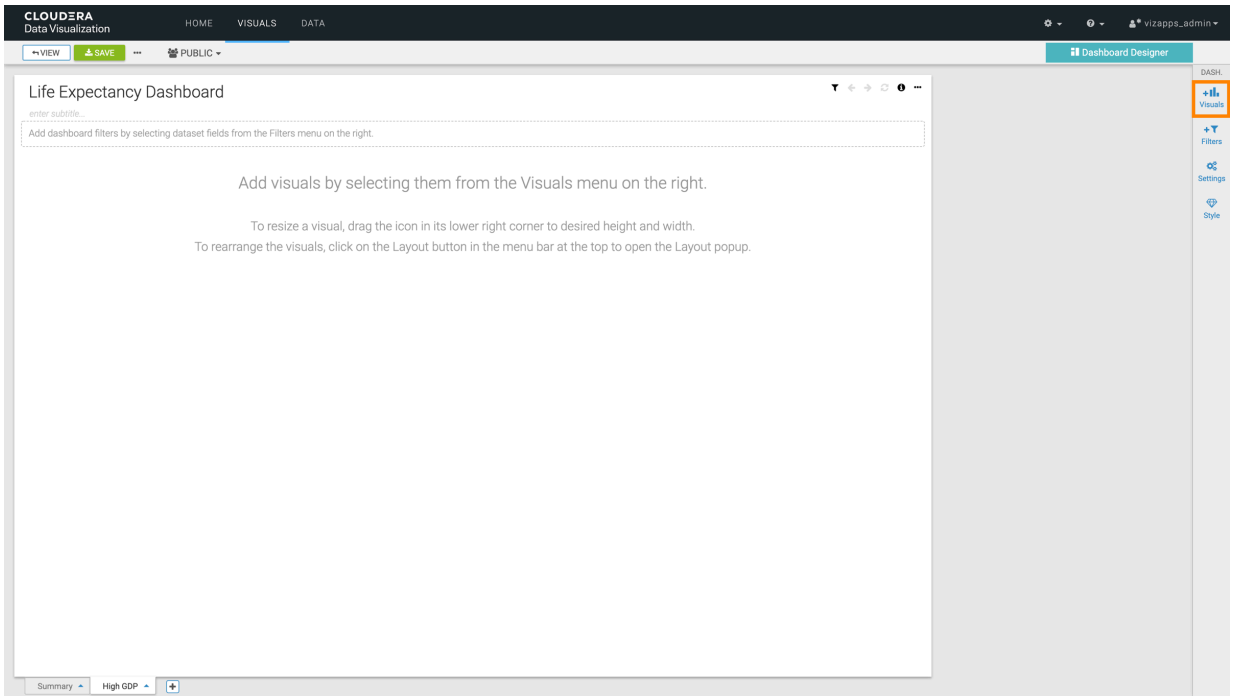

4. In the Visuals menu, configure the visual settings as needed.

   You can set the following:

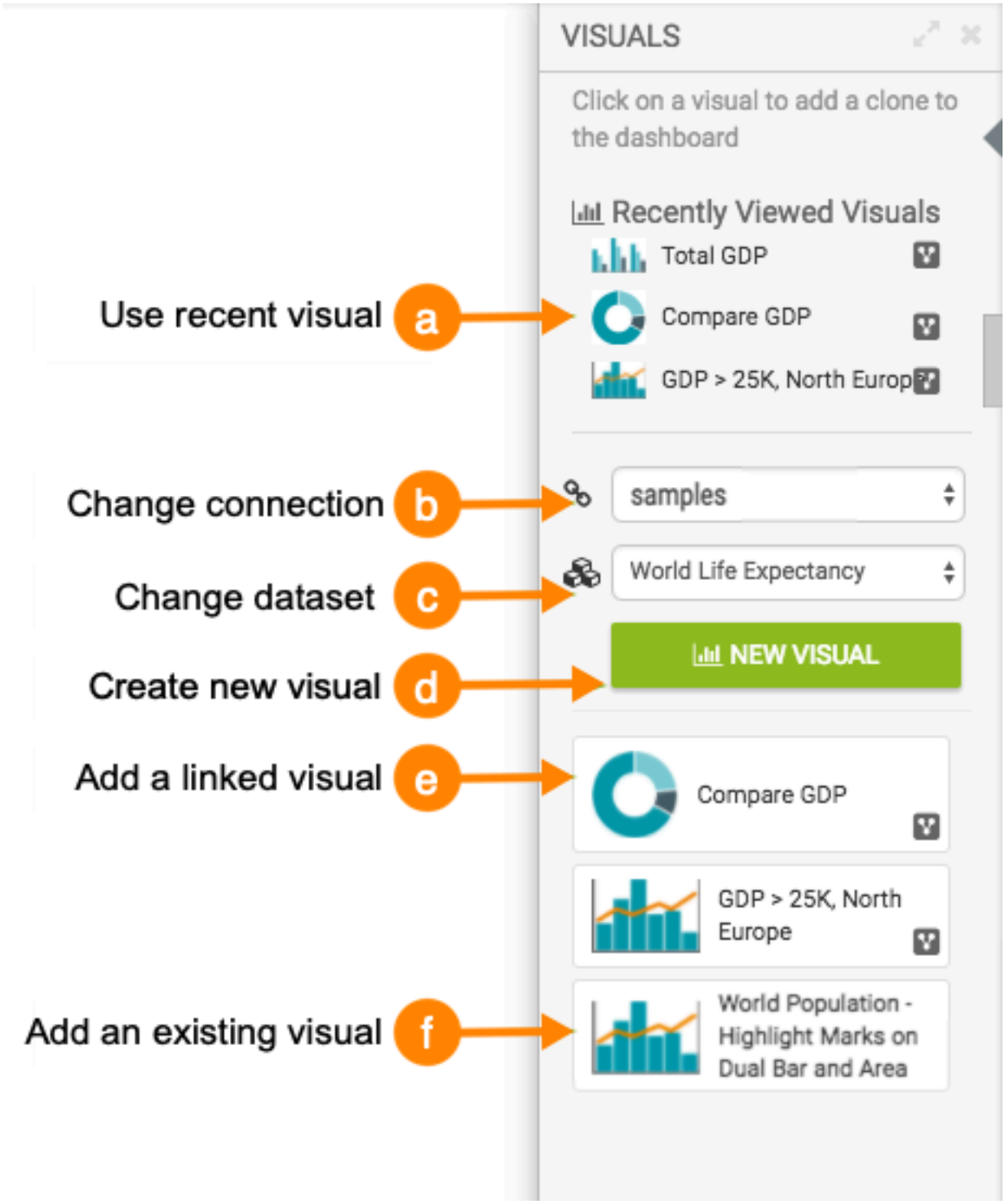

   a. *Recently Viewed Visuals*

      You can quickly access visuals added recently. Click one of the options in the top menu to add a visual to the dashboard sheet.

   b. *Connection*

Change the data connection to access visuals based on different connections and datasets. The Connection dropdown lets you select an alternate connection, affecting the dataset and updating the visuals list. It is available in both the Visuals and Filters menus.

Click the name of the current connection, and select a different connection from the menu. When you change the connection, it changes the dataset to the first one that is available on that connection. This also updates the list of visuals that appear in the lower portion of the menu.

Note: A dashboard can contain visuals from several connections.

c. *Dataset*

Choose a different dataset within the same or different connections. Changing the dataset updates the list of visuals available for addition to the dashboard sheet. This option is available in both Visuals and Filters menus.

Click the name of the current dataset, and select a different dataset from the menu. When you change the dataset, a new set of existing visuals appears in the menu. You can change the source dataset of the visuals that you plan to add to your dashboard sheet. The new dataset may be on the same connection, or on a different connection.

Note: A dashboard can contain visuals from multiple datasets that share the same connection or from different datasets across different connections.

d. *New Visual*

Start a new visual within the dashboard, creating a table visual based on the chosen dataset. You can open this visual in edit mode and make changes by selecting another option from the Visual Catalog, populating the shelves of the visual, and customizing settings, colors, and more.

e. *Linked Visual*

Reuse an existing visual marked as linked. Choose to clone the visual or maintain a link for synchronized changes across dashboards.

1. Click to select a visual that is marked with the (linked) symbol.

   Cloudera Data Visualization opens an Add Linked Visual modal window.
2. Choose one of the following options:
   - Clone visual option, the default, makes a copy of the linked visual. This new visual can be subsequently changed without affecting the original visual.
   - Maintain link option keeps the association; any changes made to the visual in one dashboard appear in all dashboards that feature this same visual.
3. After making your choice, click ADD VISUAL.

f. *Existing Visual*

Access visuals belonging to the specified dataset. You can click visuals to add them to the dashboard sheet. To add a visual that already exists in this dashboard and appears on another sheet, click the icon that represents this visual.

5. Click SAVE to save all changes to the sheet.

What to do next

To further enhance the dashboard sheet, consider adding filters. For instructions, see *Adding filters to a sheet*.

Adding a filter to a sheet

Cloudera Data Visualization enables you to add filters to a sheet.

Procedure

1. Open a dashboard of your choice in Edit mode.
2. Choose a sheet in the dashboard where you plan to add visuals.
   In this example, a sheet called High GDP is used.
3. Select Filters in the Dashboard Designer side menu bar on the right.

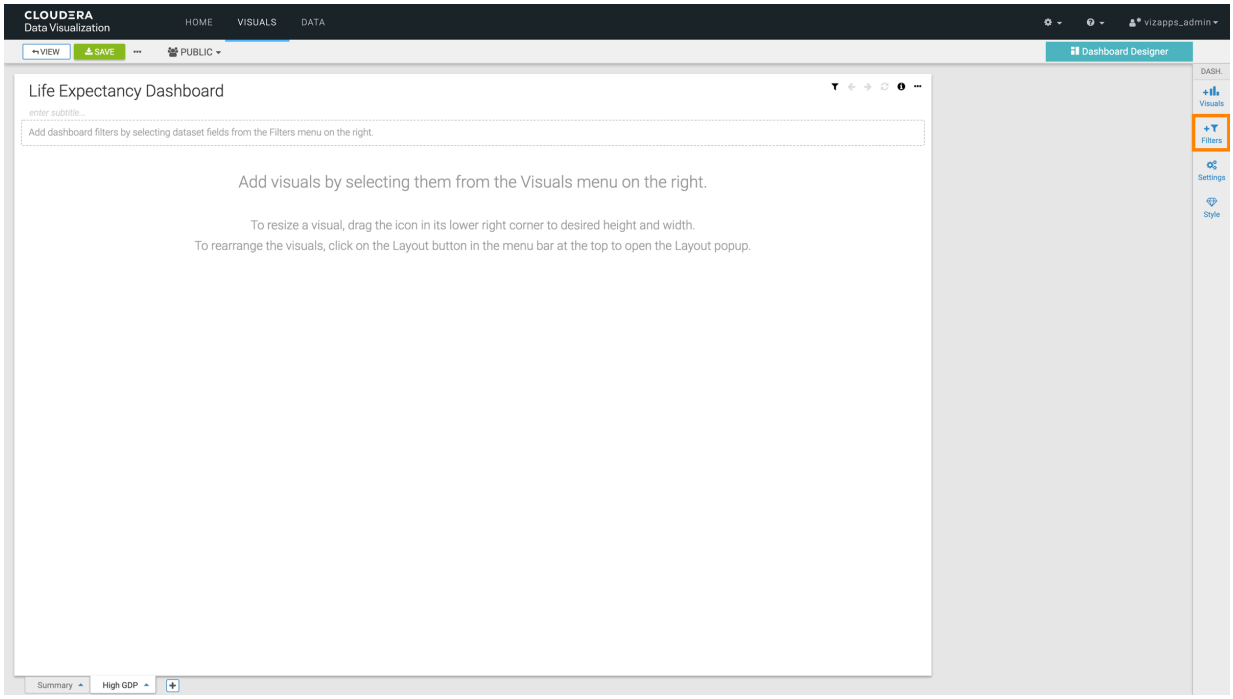

4. In the ADD FILTERS menu, configure your filter as you need.

   You can set the following options:

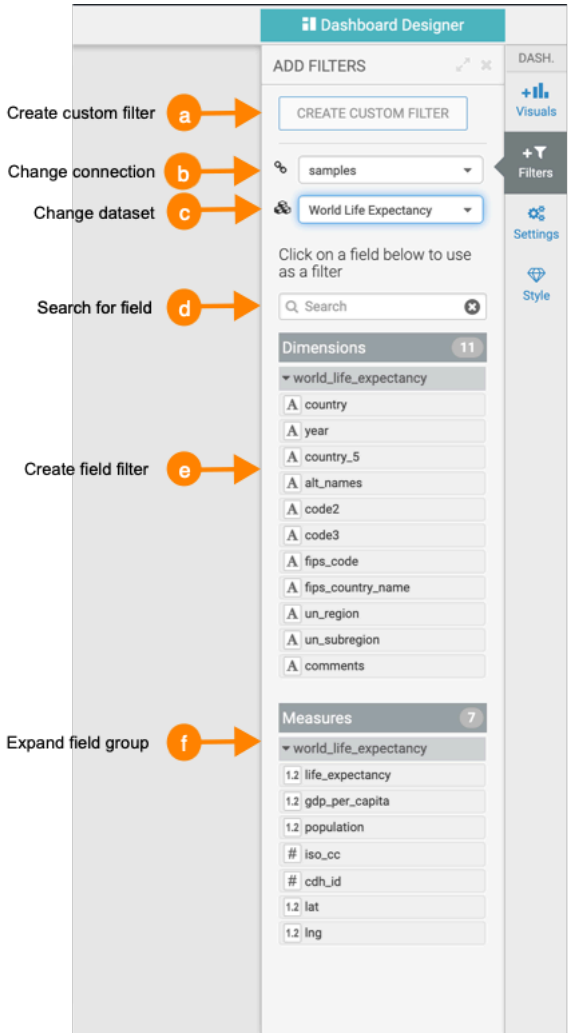

   a. *Custom Filter*

   Click CREATE CUSTOM FILTER at the top of the Filters menu.

   Cloudera Data Visualization adds a new filter to the filter area of the sheet. If you click the gear icon on the filer, the Settings window modal is displayed, where you can name the new filter and specify its value options, data handling, display, scope, and custom styling.

   For information on new dashboard-specific filter scope options, see *Configuring scope of dashboard filters*.

   b. *Connection*

   Cloudera Data Visualization supports access to multiple data sources in the same dashboard, so you can change the connection to access visuals based on different connections and dataset.

   Click the name of the current connection, and select a different connection from the menu. When you change the connection, note that it changes the dataset to the first one that is available on that connection. This also updates the list of visuals that appear on the lower portion of the menu.

   c. *Dataset*

   You can change the source dataset of the visuals that you plan to add to your dashboard sheet. The new dataset may be on the same connection, or on a different connection.

Click the name of the current dataset, and select a different dataset from the menu. This also updates the list of visuals that appear on the lower portion of the menu.

d. *Search*

For datasets built on very wide tables, this option helps you find the base field for your filter.

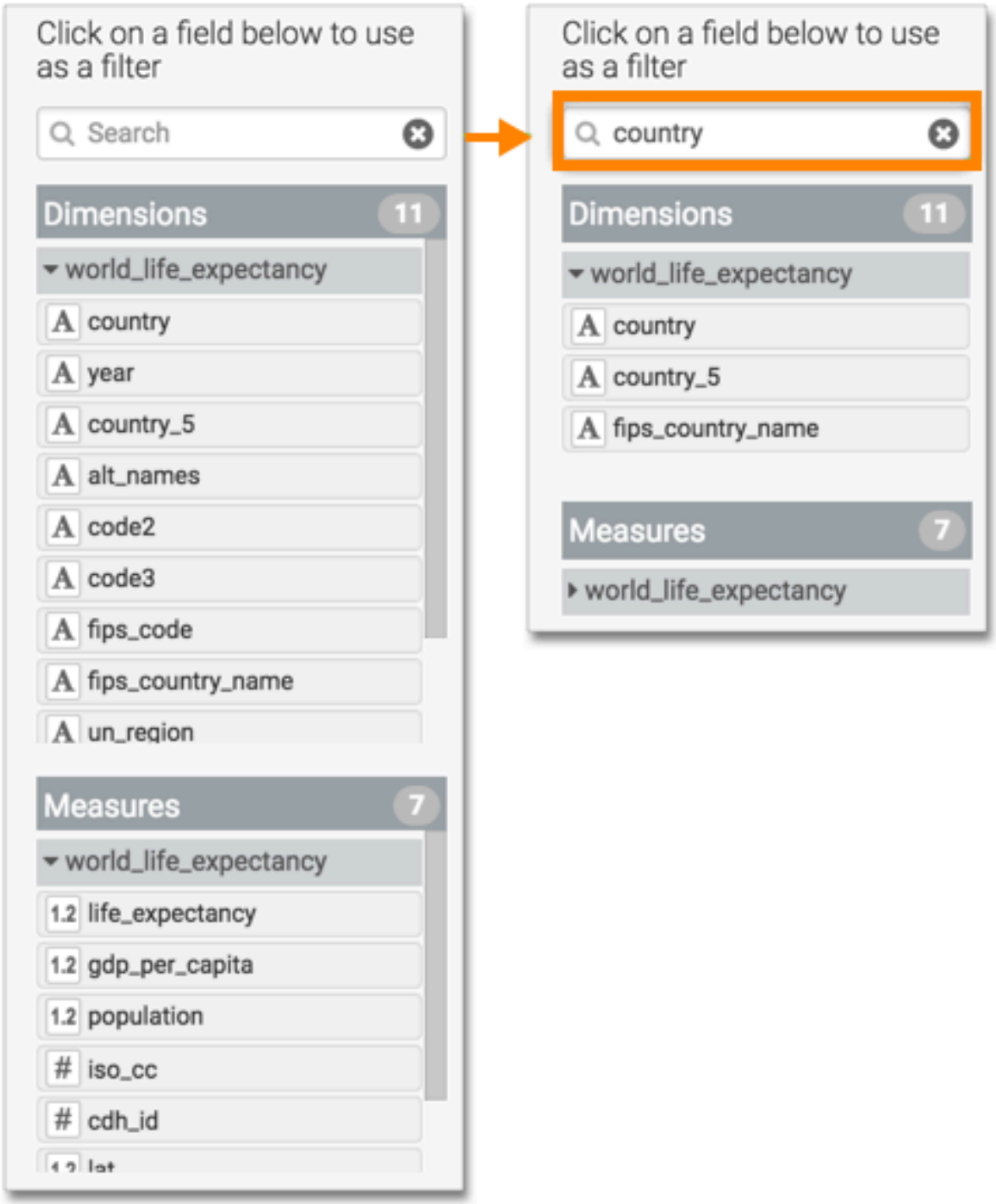

e. *Field Filter*

Clicking one of the fields in either Dimensions or Measures adds a basic filter based on that field into the filter area of the interface. Alternatively, you can also drag the field that you want to use as a filter and drop it into the filter bar on the sheet.

In this example, the filter country has been added. You can use it directly, or open the Settings menu, where you can rename the filter and specify its value options, data handling, display, scope, and custom styling.

f. *Field Groups*

For ease of viewing, you can collapse or expand the list of fields in either Dimensions, Measures, or both.

Additionally, datasets that represent two or more joined tables group the fields of each table separately. Clicking the name of the table under Dimensions or Measures toggles between expanding the list of these fields.

Renaming a sheet

Cloudera Data Visualization enables you to rename sheets to provide context and make it easier to identify the content of each sheet within the dashboard.

Procedure

1. Open a dashboard of your choice in Edit mode.

   By default, each dashboard contains a single sheet, named Sheet 1.

2. At the bottom left corner of the interface, click the icon next to the name of the sheet.

3. In the sheet management menu, click Rename.

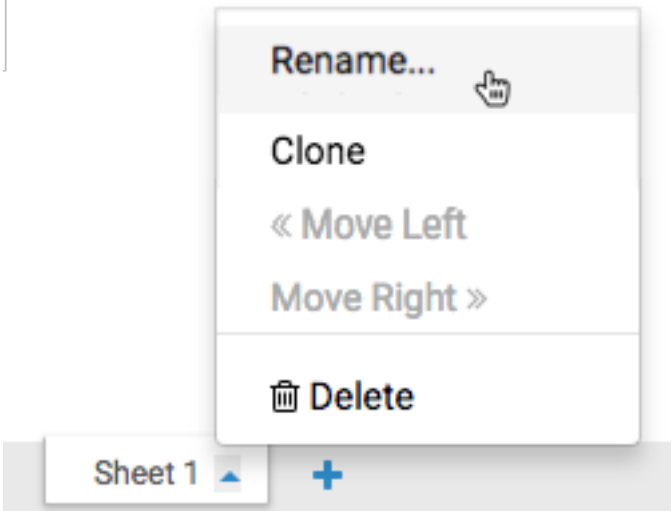

   The sheet title becomes editable.

4. Type the new name and either press enter/return on the keyboard, or click outside the sheet name space.

   In this example, the name of the sheet changes from Sheet 1 to High GDP.

5. Click SAVE to save all changes to the dashboard.

   Alternatively, you can double-click the name of the sheet to change it in Edit mode.

   To preview the changes at runtime, click VEW.

Reordering sheets

Cloudera Data Visualization enables you to move and reorder sheets within a dashboard.

Procedure

1. Open the dashboard in Edit mode.
2. Select the sheet you want to move by clicking on its tab, at the bottom left corner of the interface.

   In this example, the sheet Low GDP is chosen, the second of three sheets of the dashboard.

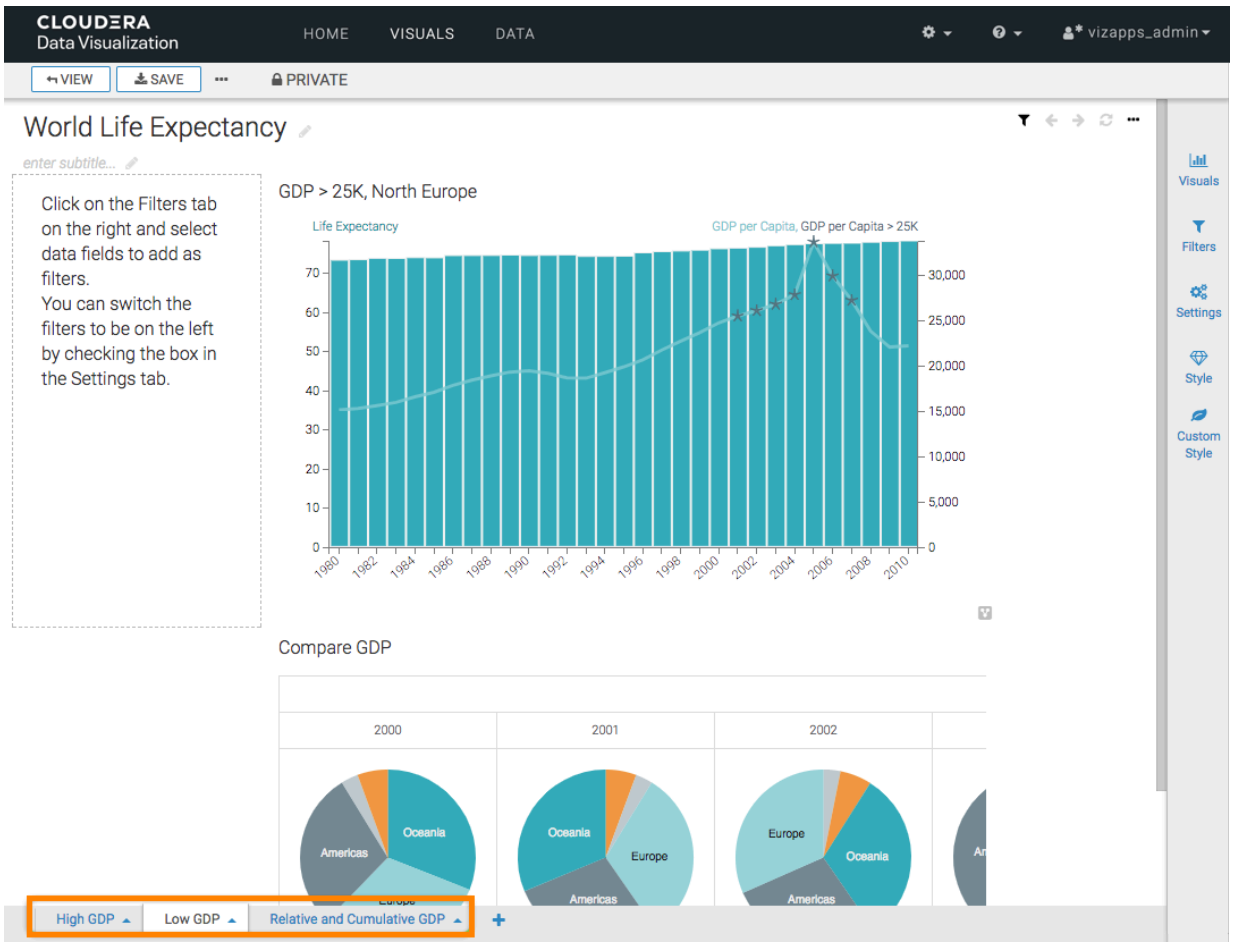

3. To move a sheet to the left, click the (up arrow) icon next to the title of the sheet, and choose the Move Left option.

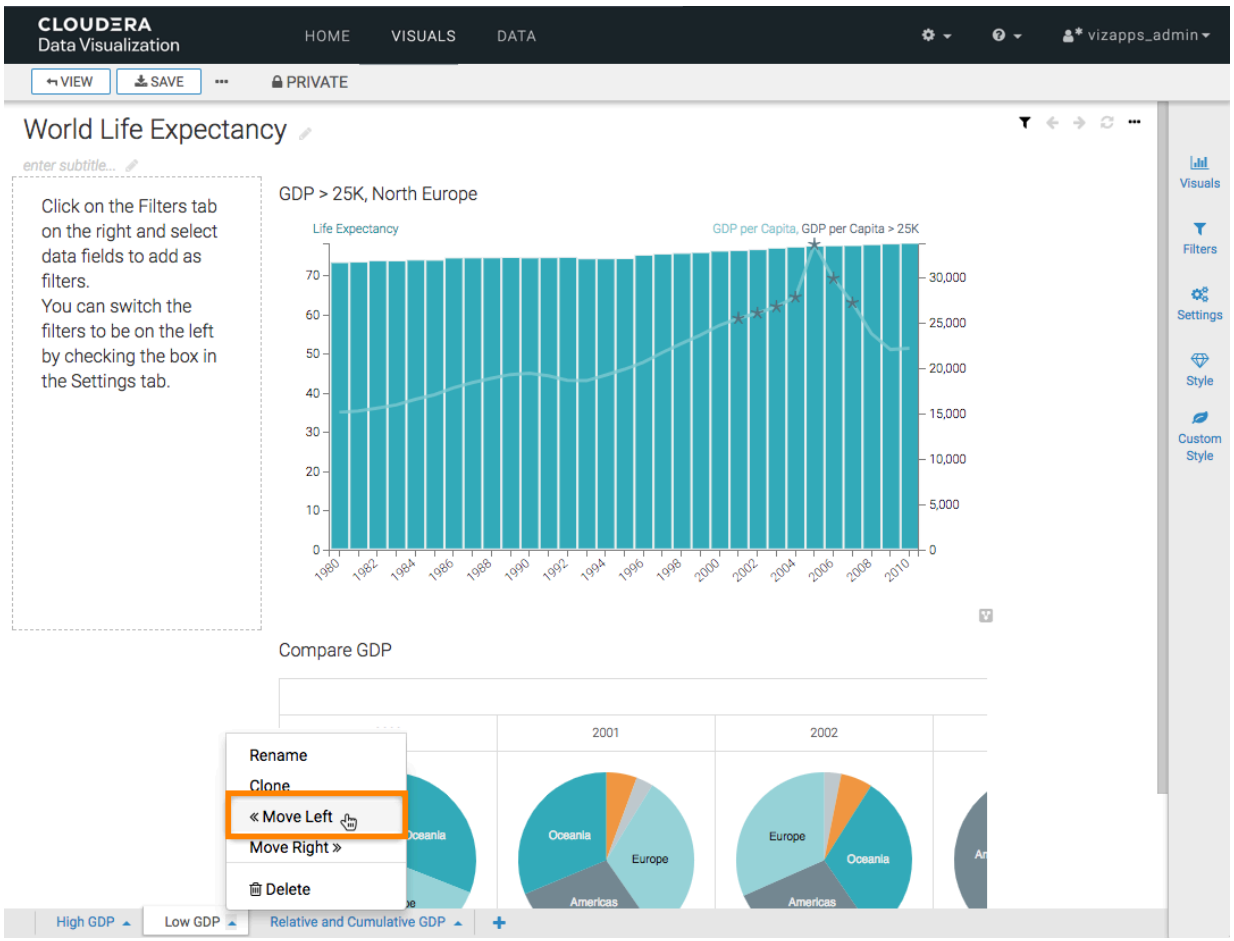

Similarly, if you wanted to move the sheet to the right instead, choose the Move Right option.

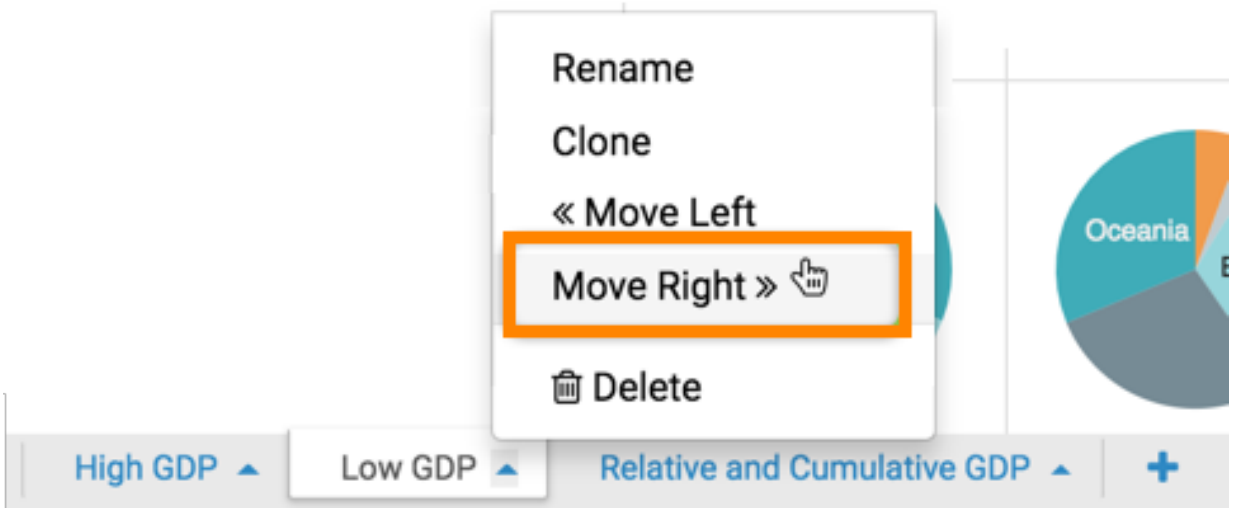

The dashboard moves to the extreme left, or the first sheet in the dashboard.

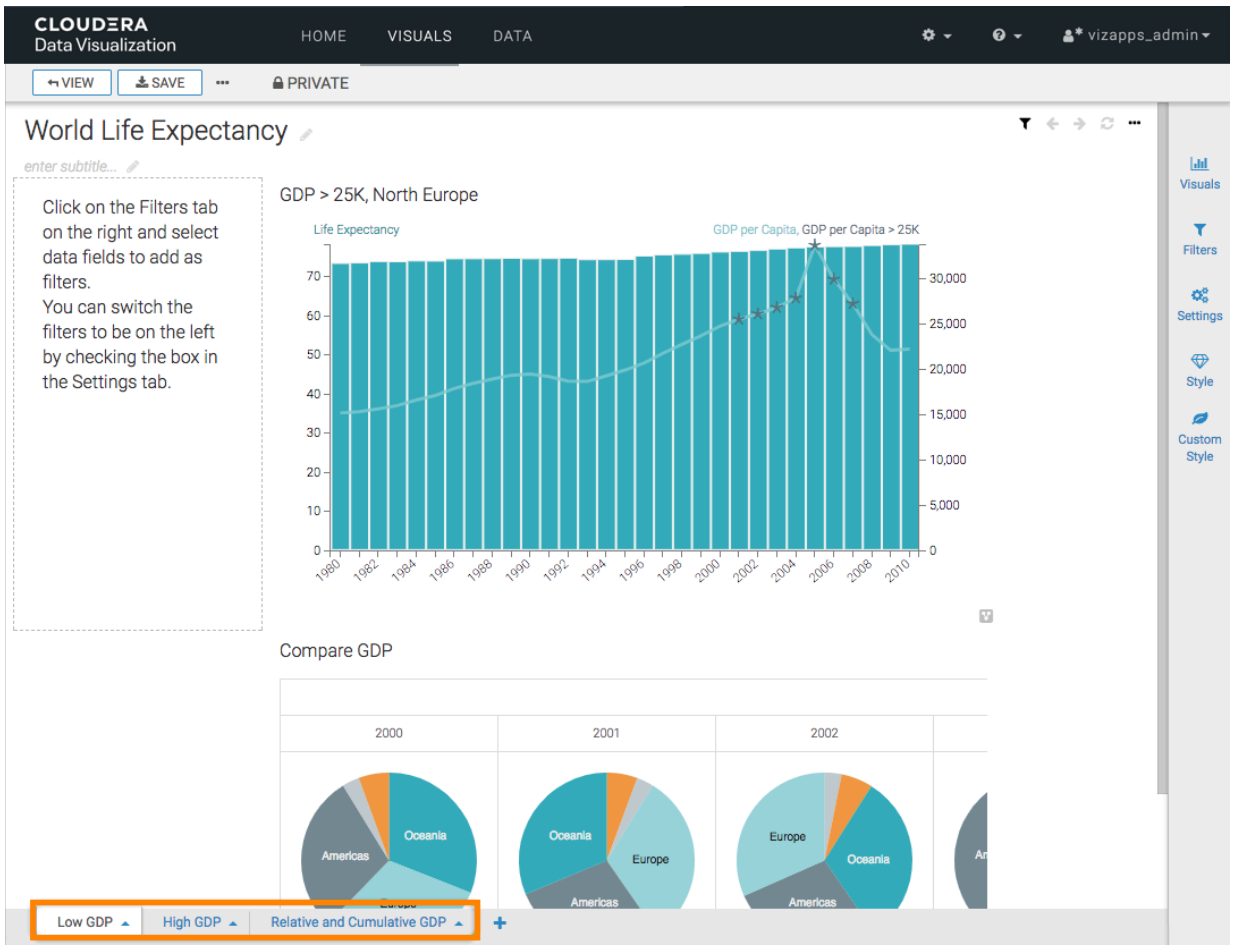

4. Click SAVE to save all changes to the dashboard.

5. To see how the dashboard appears at runtime and to confirm that the order of sheets is correct, click VIEW.

   In this example, the order of sheets in the dashboard is just like intended.

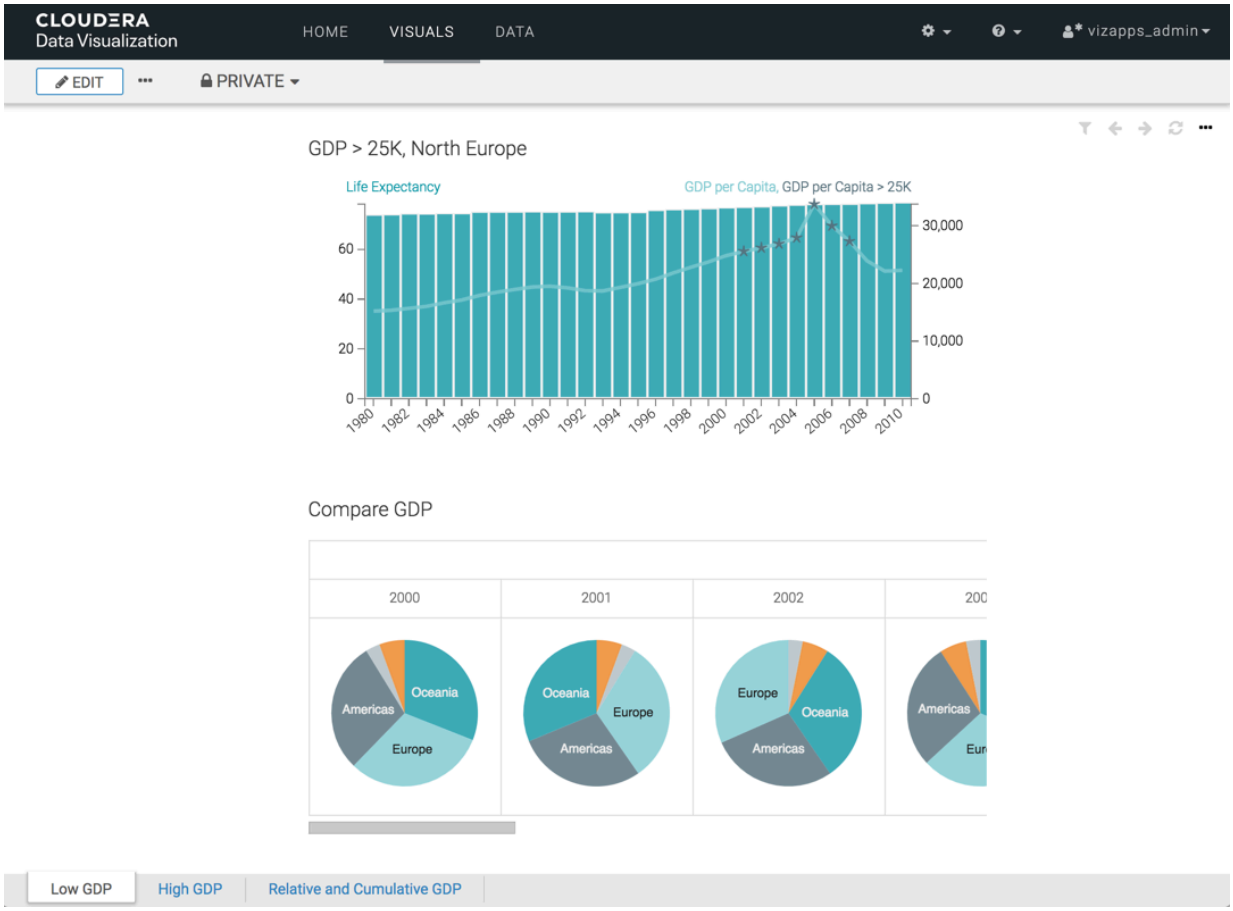

Reordering sheet IDs to match positions

Cloudera Data Visualization allows you to reassign the sheet IDs based on their new order within the dashboard after reordering or deleting sheets. This ensures consistency between sheet IDs and the display order of the actual sheets.

About this task

In addition to their names, each sheet in the dashboard has a unique identifier, which is visible in the sheet's URL. This unique ID helps you to easily differentiate between various sheets within the dashboard.

The IDs are assigned based on the original order of the sheets. When sheets are repositioned or removed, the ID sequence can lose alignment with the current sheet order. To maintain a coherent reference system, you can reassign sheet IDs based on their new order.

Procedure

1. Open the dashboard in Edit mode.

2. Hover over the icon at the bottom left corner of the Dashboard Designer interface and click the icon that appears next to it.

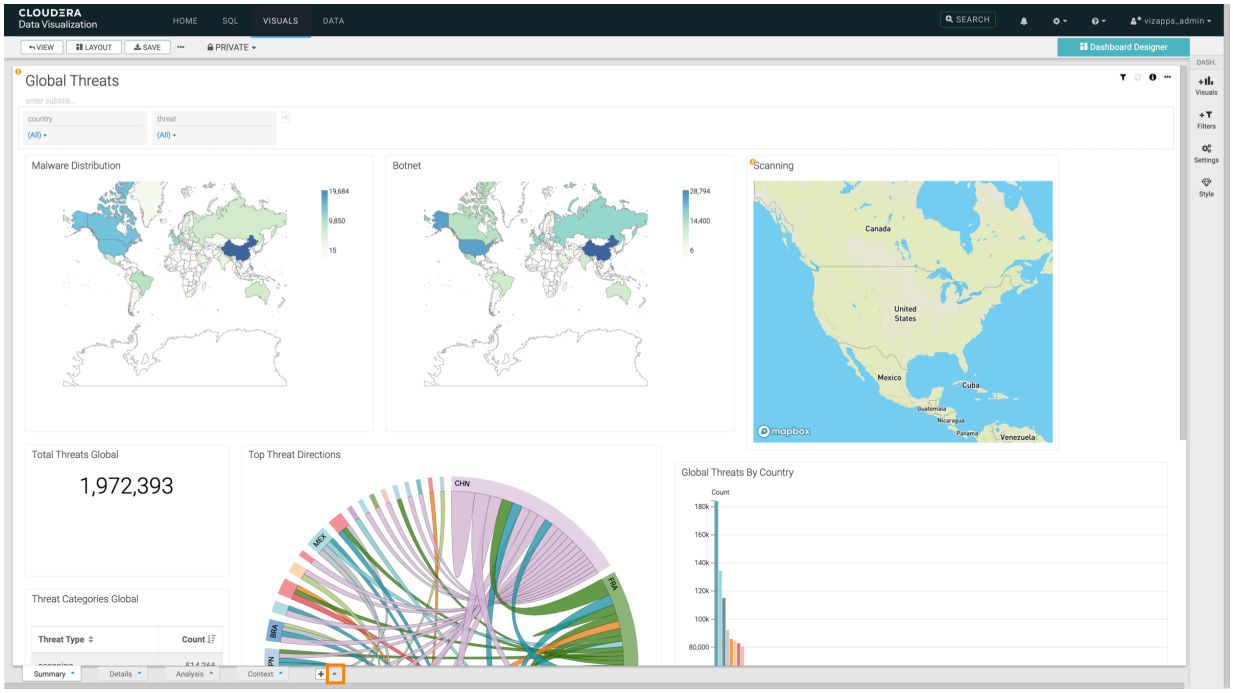

3. Select Reorder sheet IDs to match positions from the pop-up menu.
   Sheet IDs are automatically reassigned based on the new order of sheets.

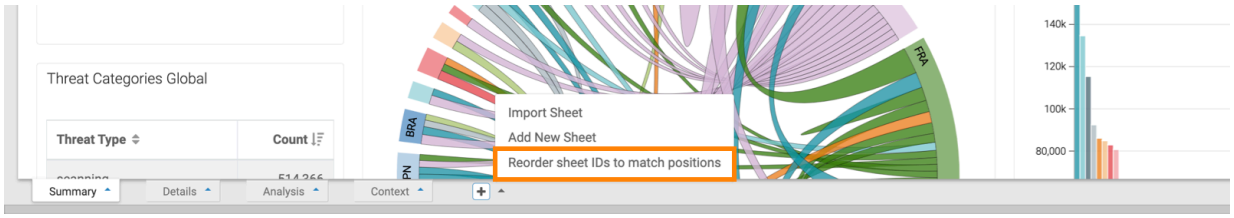

4. Click SAVE to save all changes to the dashboard.

Results

The new sheet ID numbering now corresponds to the order of the sheets on the dashboard.

Cloning a sheet

Cloudera Data Visualization enables you to clone a sheet in a dashboard.

You can clone internal and external sheets.

Cloning an internal sheet in a dashboard

In Cloudera Data Visualization, you can clone sheets within a dashboard.

Procedure

1. Open the dashboard in Edit mode.

2. Select the sheet you want to clone by clicking on its tab, at the bottom left corner of the interface.
   In this example, the sheet Low GDP is chosen.
3. Click the (up arrow) icon next to the title of the sheet, and choose the Clone option.

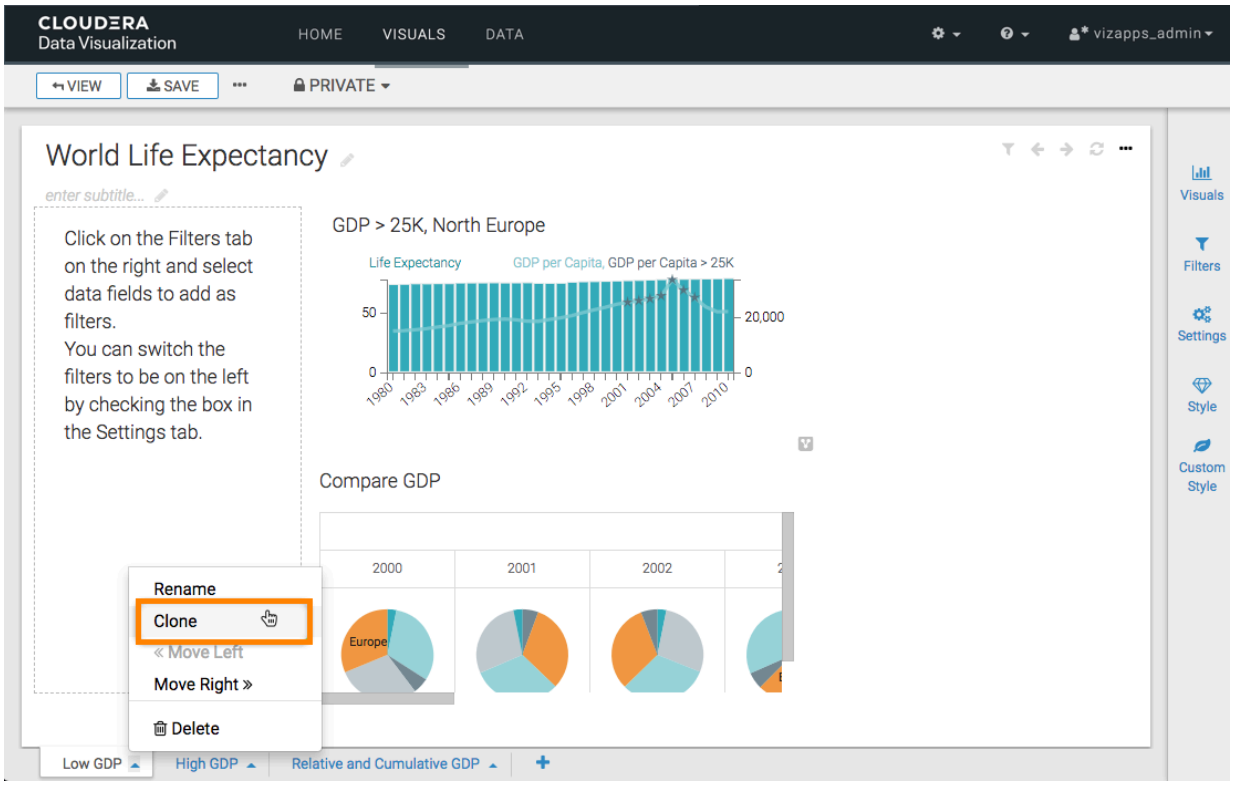

Data Vizualization creates a clone of the sheet, and places it on the extreme right.

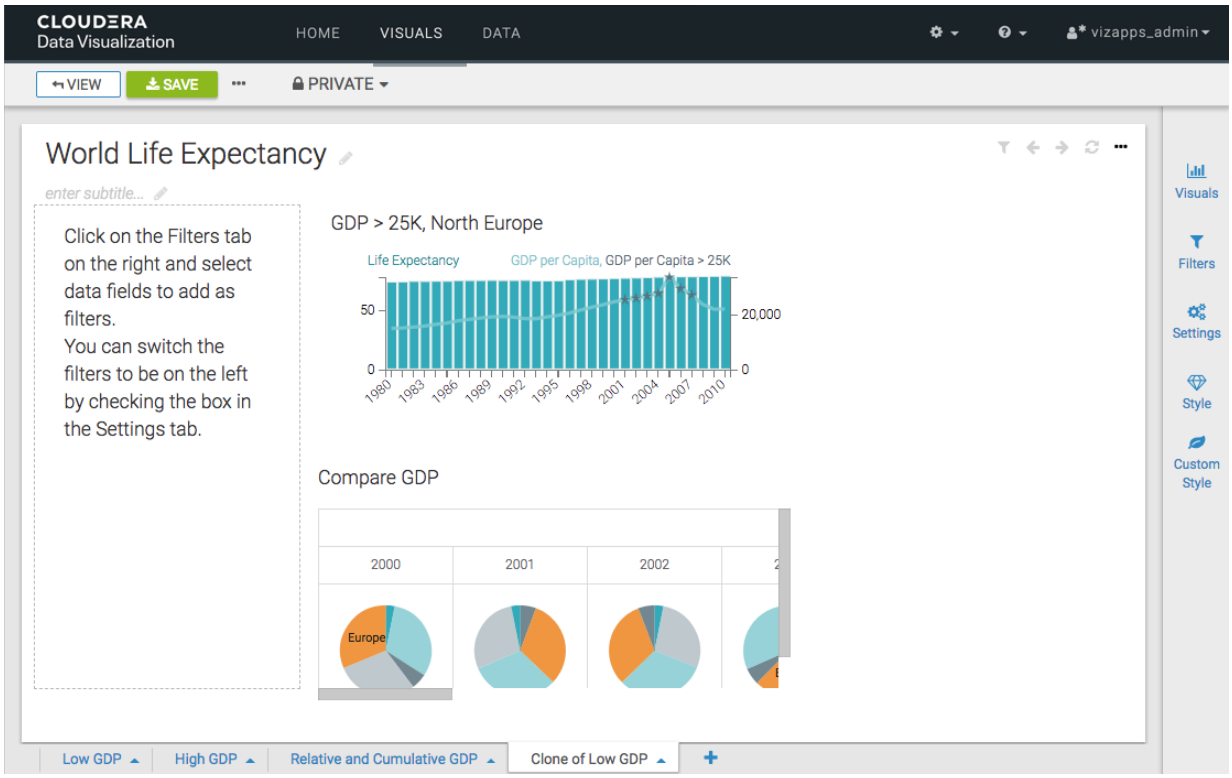

4. Rename the sheet, and click SAVE to save the all changes to the dashboard.

Cloning an external sheet in a dashboard

In Cloudera Data Visualization, you can clone a sheet from another dashboard, and import it into your current dashboard.

Procedure

1. Open the dashboard in Edit mode.
2. At the bottom of the interface, hover over the (plus) icon next to the right-most sheet title.

   A new widget, the (up) icon, appears to the right of the (plus) icon.
3. Click the (up) icon.

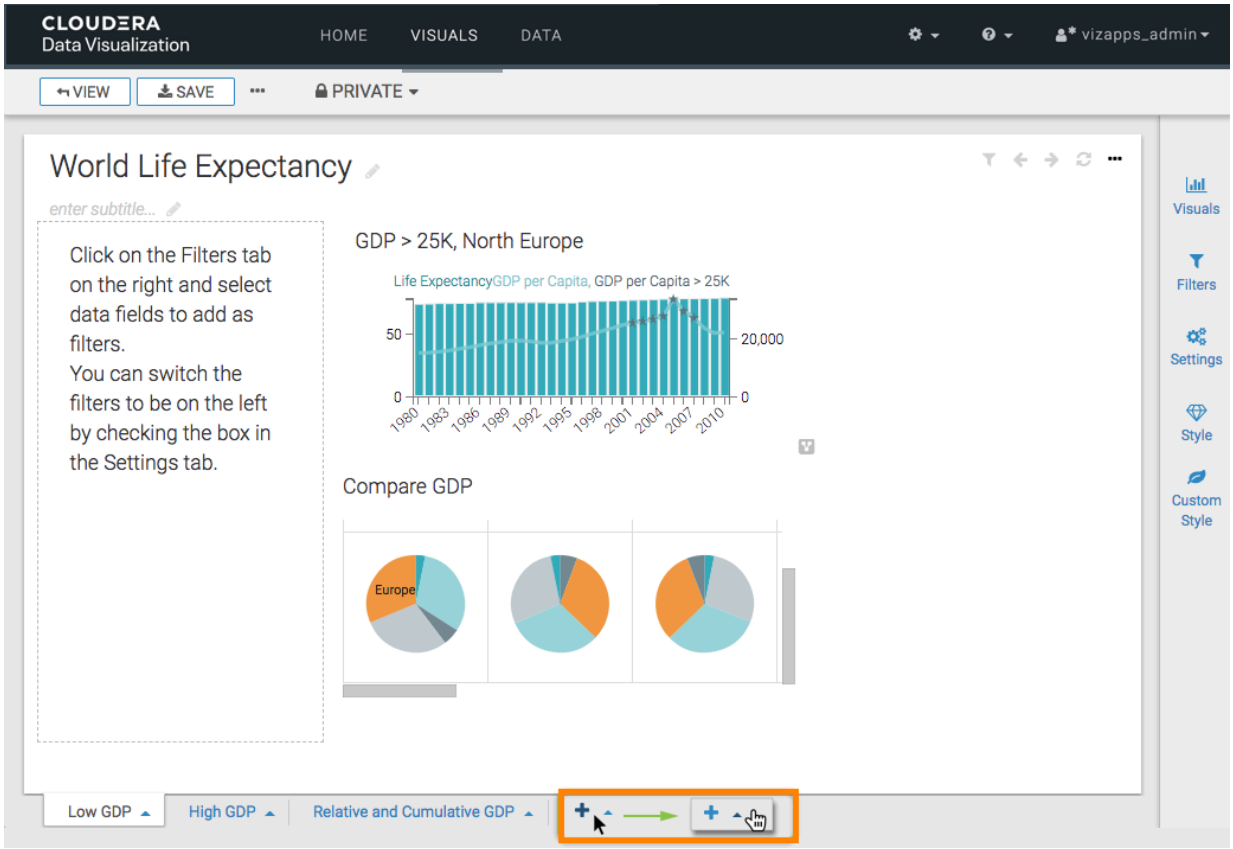

4. In the menu, select Import Sheet.

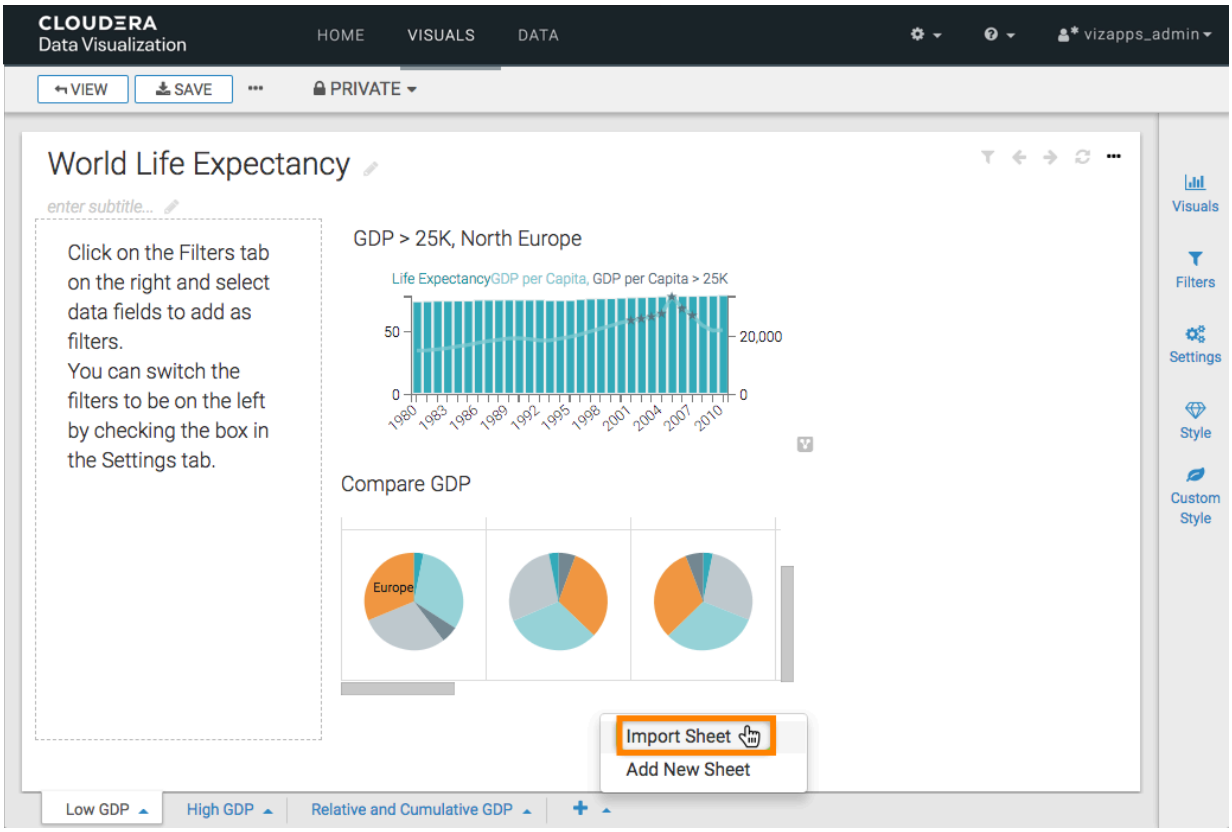

5. In the Pick a dashboard modal window, make the selection in the following manner:
   a. Select the dataset.
   b. Select the dashboard.
   c. Select the particular sheet that you plan to import.
   d. Click the dashboard.

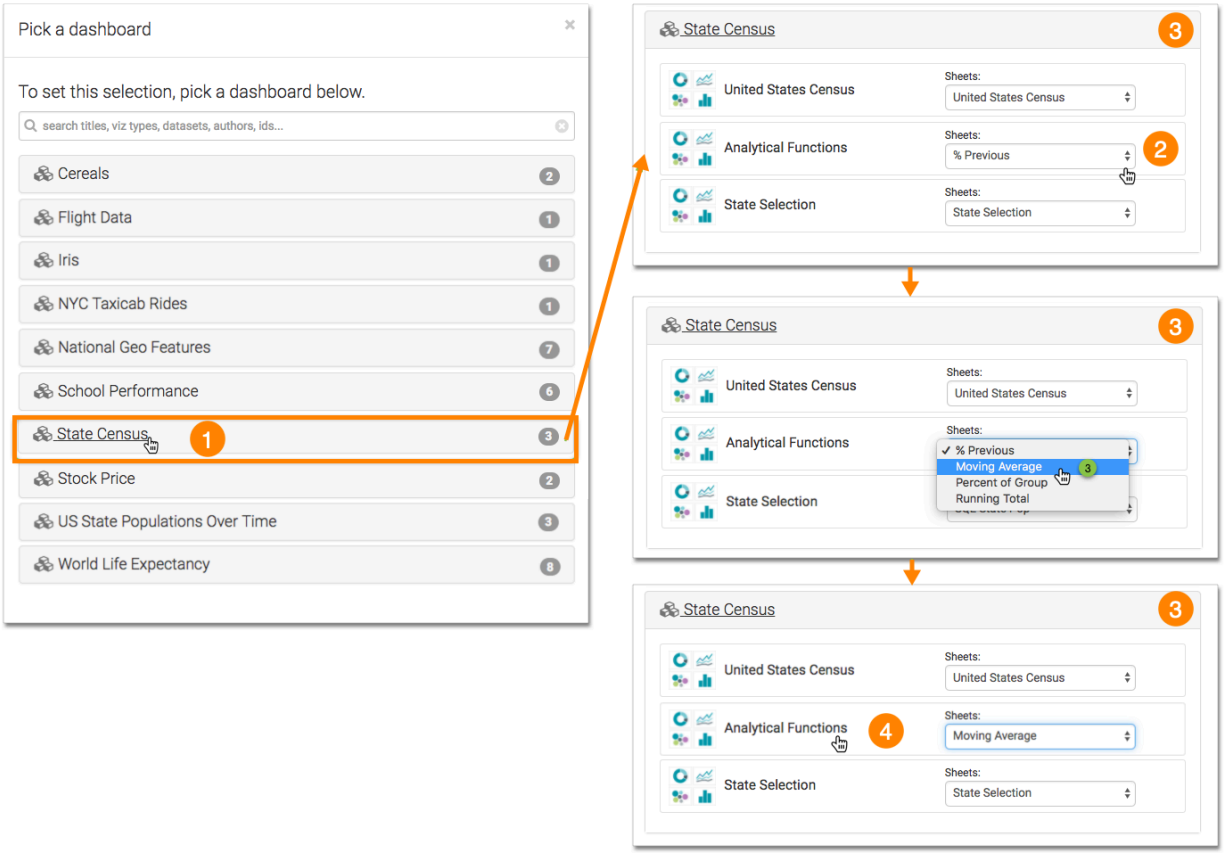

Cloudera Data Visualization creates a clone of the external sheet, and imports it into the dashboard, placing it on the extreme right.

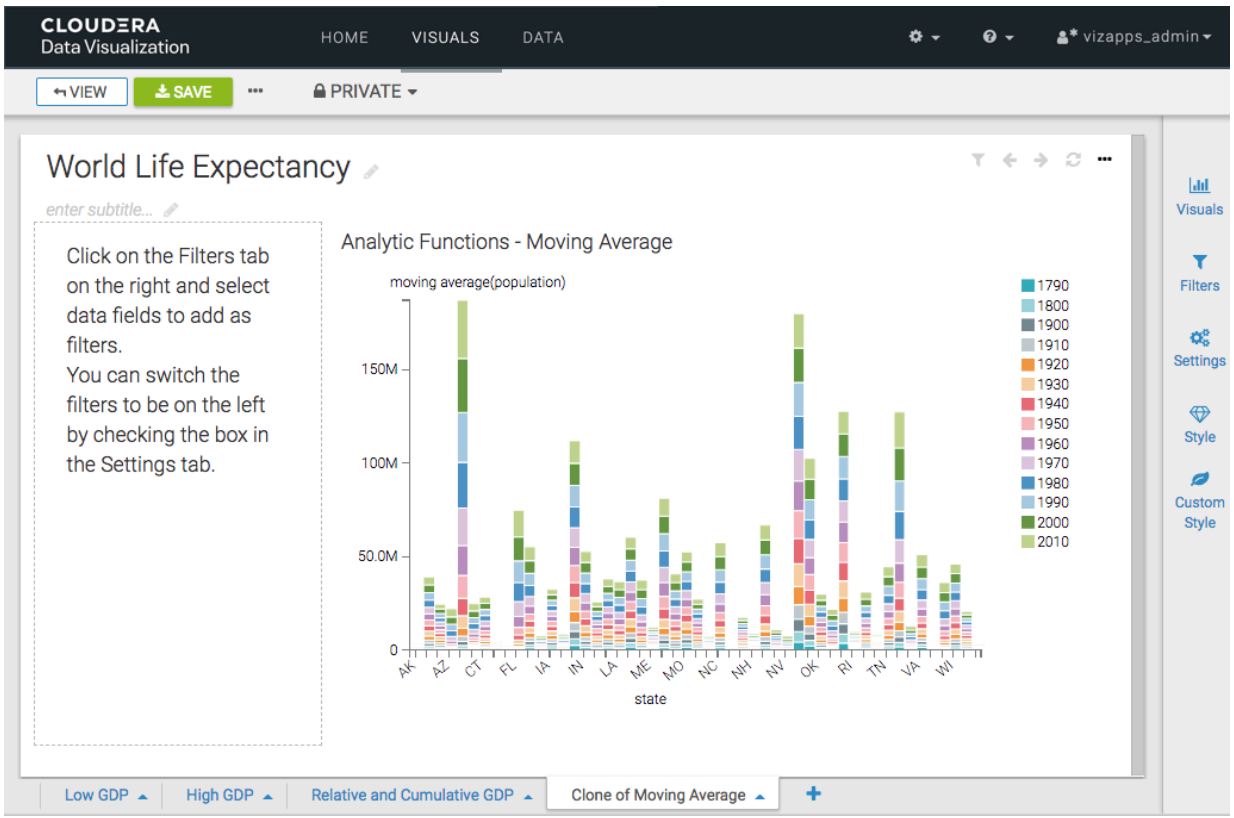

6. Rename the sheet and click SAVE to save the all changes to the dashboard.

Downloading a sheet

Cloudera Data Visualization enables you save the current sheet of a dashboard as a .png file and to download the entire dashboard as a PDF file with each sheet on a separate page.

Downloading dashboard sheets as PNG files

Cloudera Data Visualization enables you to save the current sheet of a dashboard as a PNG file. The filename is the same as the name of the dashboard, and it has a timestamp specifying the date and time when the file is created.

About this task

This method is better than using a desktop tool because it handles images that are too long or wide to fit the screen.

Note:

- This feature is available only when thumbnails are enabled.
- For snapshots of custom JS visuals, Cloudera Data Visualization supports only ECMA Script 5 for Javascript.
- Cloudera Data Visualization does not support PNG download on the Windows platform.

Procedure

1. In the Dashboard Designer top menu bar, click the (ellipsis) icon.

2. Click Download as..., and select Image (png) from the secondary menu.

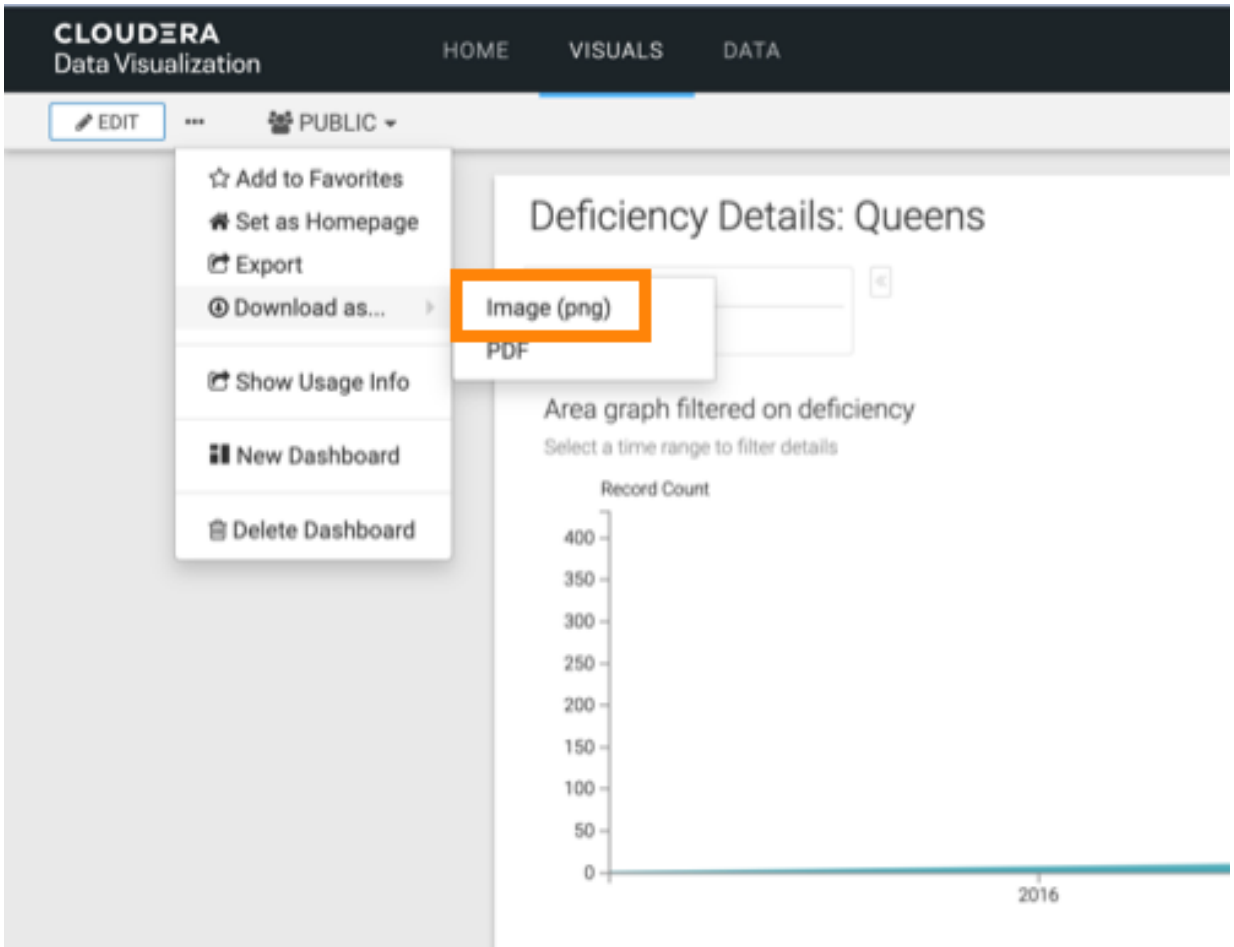

A Generating the requested file message appears.

3. After the download is complete, open the default download folder on your computer.

   Cloudera Data Visualization saves the current sheet to a PNG file.

   In our example, the generated filename, Example close_open issue_20190207094420, has the following components:

   - *Example close_open issue*

     Same name as the dashboard
   - *20190207*

     Signifies the date the file is created, in YYYYMMDD format
   - *094420*

     Signifies the time the file is created, in hhmmss format

   Note: Cloudera Data Visualization renames sheets with names that contain characters not supported by the file system. For example, for the sheet Example close/open issue, it generates the file as Example close_open issue_xxxxxxxxxxxxxx.pdf, replacing the forward-slash (ASCII #47) character with a space.

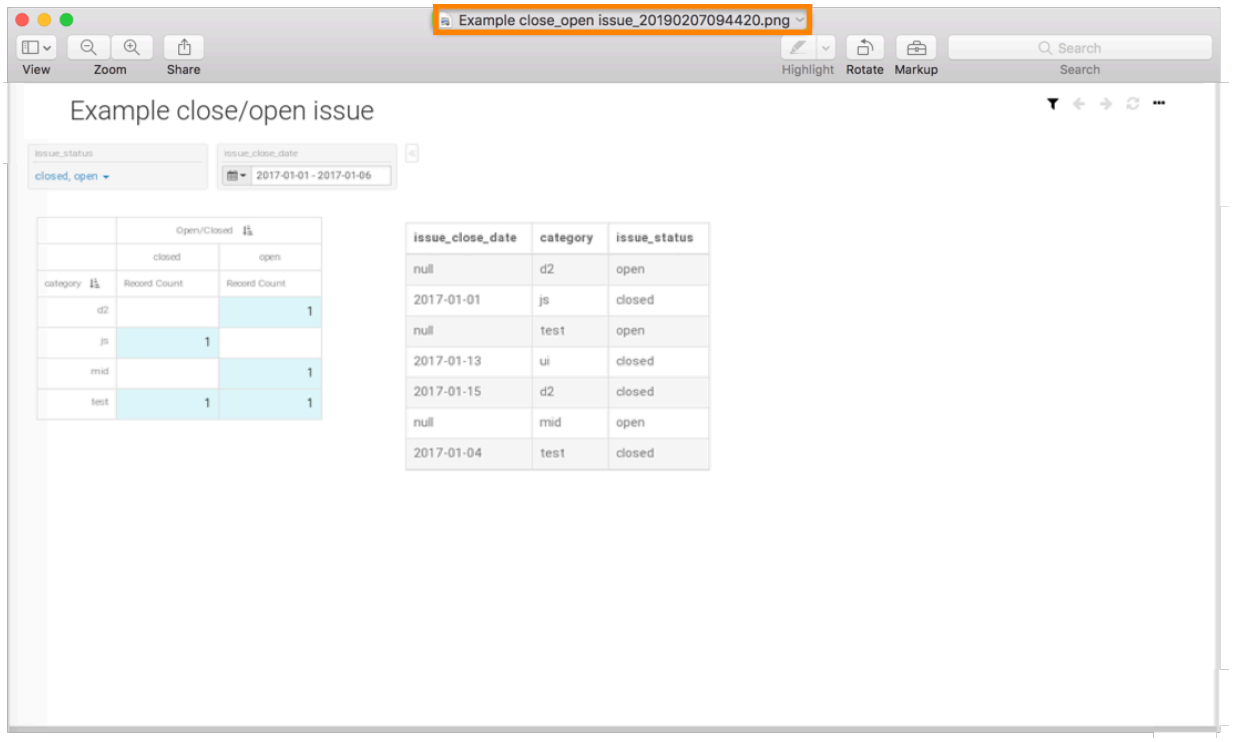

Downloading all sheets as a PDF File

Cloudera Data Visualization enables you to download the entire dashboard as a PDF file with each sheet on a separate page.

About this task

Note:

- This feature is available only when thumbnails are enabled .
- For snapshots of custom JS visuals, we support only ECMA Script 5 for Javascript.

In the following example, you can see how to download Sheet 1 and Sheet 2 of the dashboard on separate pages.

Before you begin

Before you download the dashboard as a PDF file, make sure that the Enable "Download as Image/PDF" option is enabled. You can find it in Site Administration Site Settings Snapshots .

Procedure

1. Open any dashboard. In this example, the PDF Export dashboard that has two sheets is used.

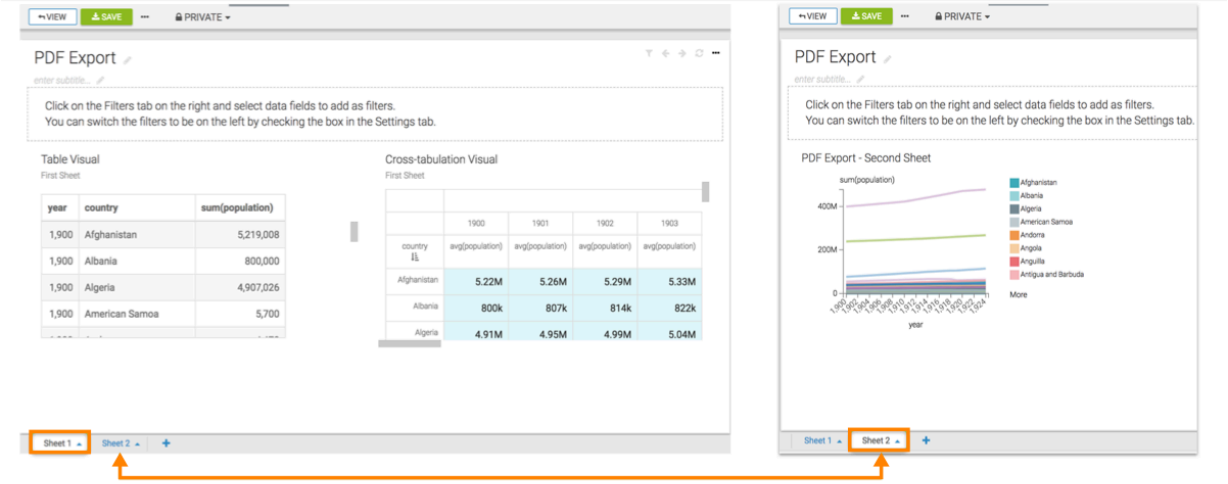

2. Click the (ellipsis) icon at the top left corner of the interface.

3. Select Download as... PDF One sheet per page .

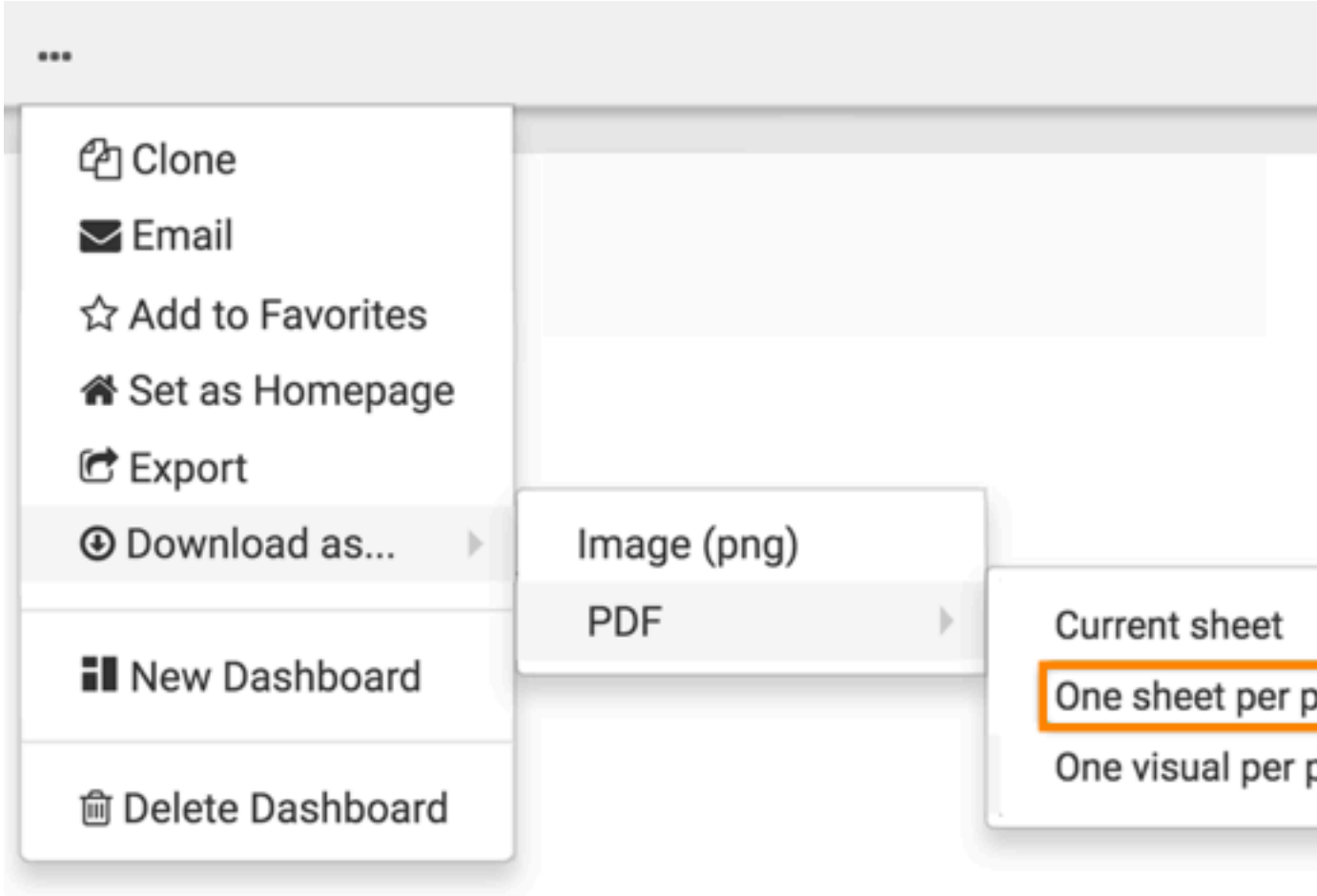

A Generating the requested file screenshot message appears.

4. After the download is complete, open the default download folder on your computer. Cloudera Data Visualization saves the current sheet to a PDF file.

   In this example, the generated filename, PDF Export_20190205142719, has the following components:

   - *PDF Export*

     Same name as the dashboard
   - *20190205*

     Signifies the date the file is created, in YYYYMMDD format
   - *142719*

     Signifies the time the file is created, in hhmmss format

   Note: Cloudera Data Visualization renames sheets with names that contain characters not supported by the file system. For example, for the sheet PDF/Export, it generates the file as PDF Export_xxxxxxxxx xxxxx.pdf, replacing the forward-slash (ASCII #47) character with a space.

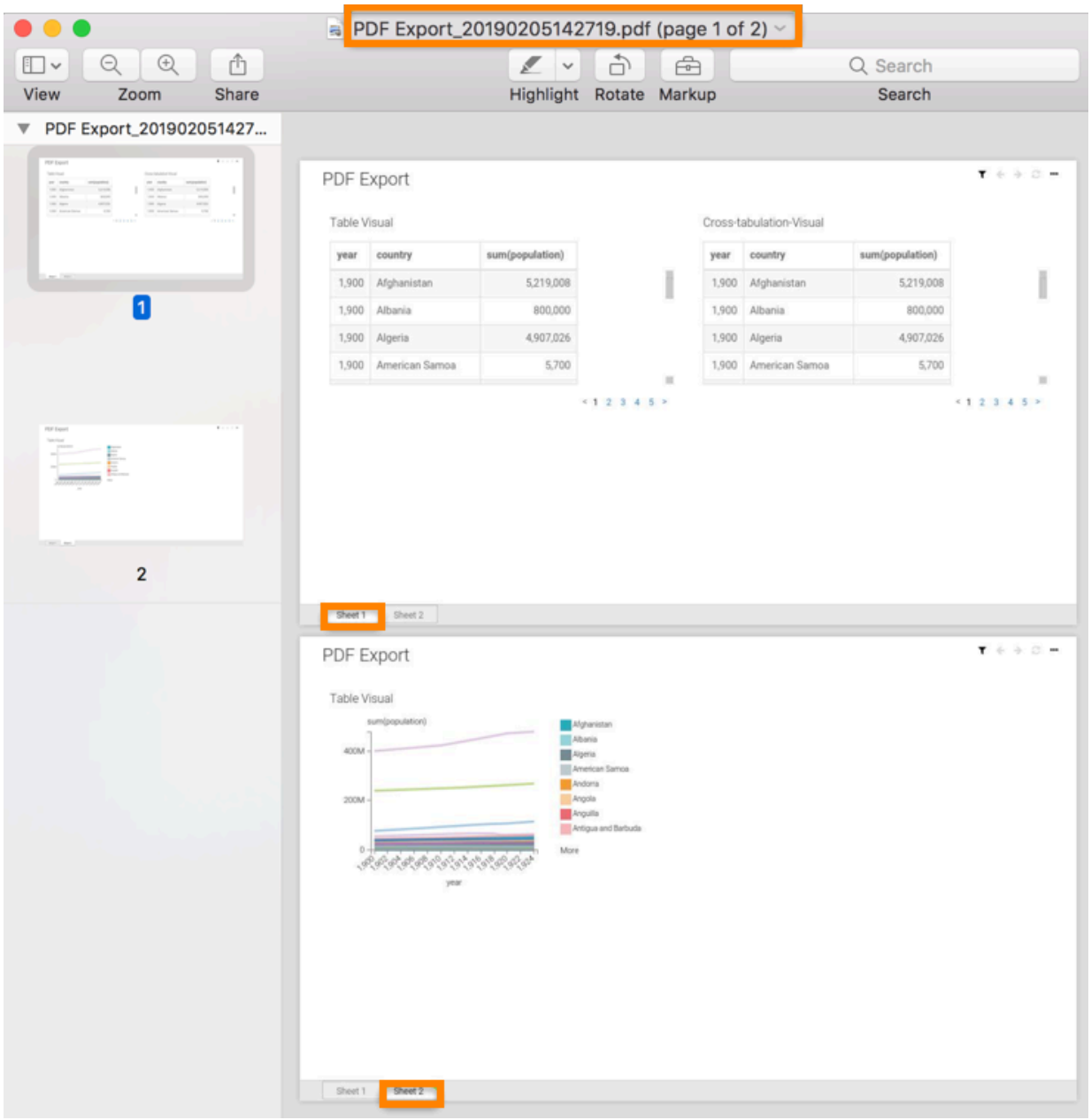

The filename is the same as the name of the dashboard, and it has a timestamp specifying the date and time when the file is created. Each sheet of the dashboard is on a separate page.

Downloading current sheet as a PDF file

Cloudera Data Visualization enables you to download the current sheet of a dashboard as a PDF file with all visuals in the current sheet on one page.

About this task

Note:

- This feature is available only when thumbnails are enabled .
- For snapshots of custom JS visuals, we support only ECMA Script 5 for Javascript.
- For filtered visuals, the filter must have app scope.

Before you begin

Before you download the dashboard as a PDF file, make sure that the Enable "Download as Image/PDF" option is enabled. You can find it in Site Administration Site Settings Snapshots .

Procedure

1. Open any dashboard.
2. Click the (ellipsis) icon at the top left corner of the interface.
3. Select Download as... PDF Current Sheet .

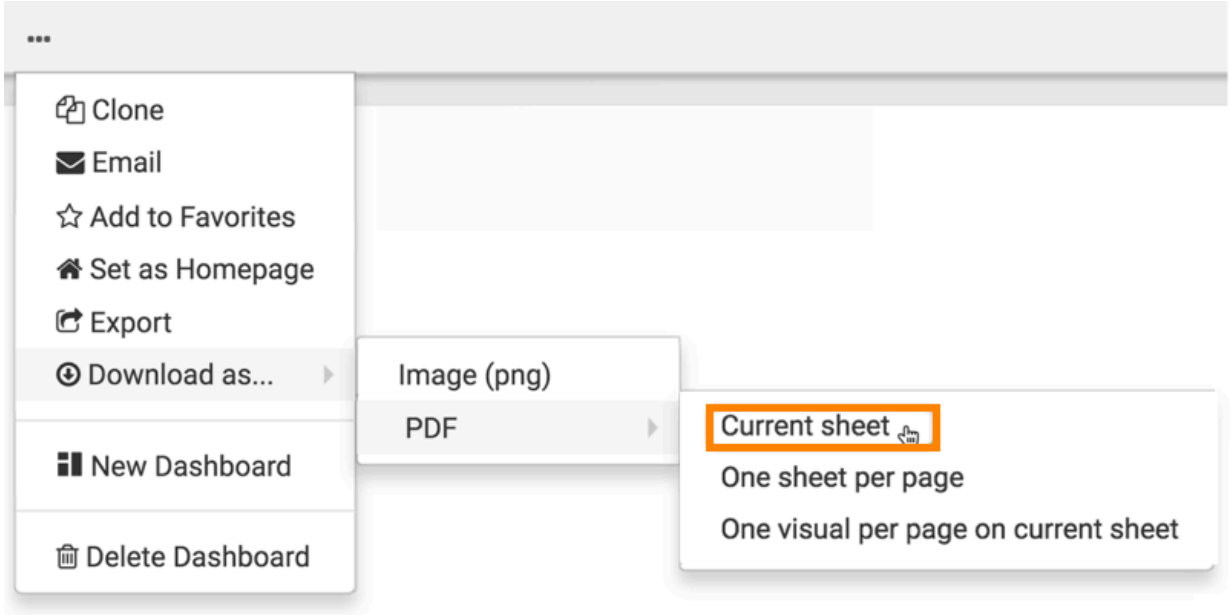

A Generating the requested file message appears.

4. After the download is complete, open the default download folder on your computer. Cloudera Data Visualization saves the current sheet to a PDF file.

   In this example, the generated filename, PDF Export_20190205135933, has the following components:

   - *PDF Export*

     Same name as the dashboard
   - *20190205*

     Signifies the date the file is created, in YYYYMMDD format
   - *135933*

     Signifies the time the file is created, in hhmmss format

   Note: Cloudera Data Visualization renames sheets with names that contain characters not supported by the file system. For example, for the sheet PDF/Export, it generates the file as PDF Export_xxxxxxxxx xxxxx.pdf, replacing the forward-slash (ASCII #47) character with a space.

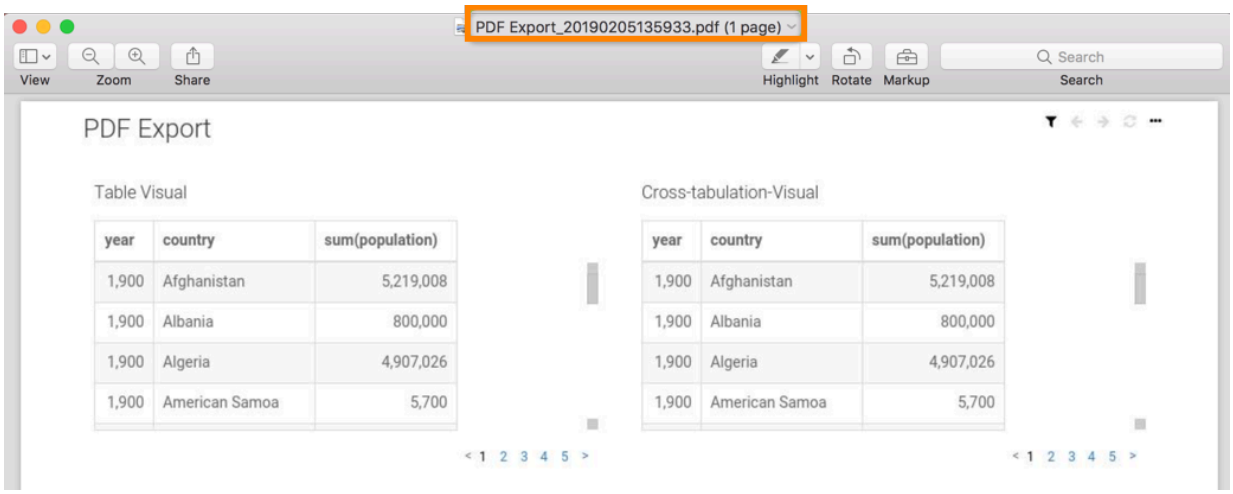

   The filename is the same as the name of the dashboard, and it has a timestamp specifying the date and time when the file is created.

5. All visuals in the current sheet (Sheet 1) are on one page. Repeat the preceding steps to download Sheet 2.

Downloading current sheet with one visual per page as a PDF file

Cloudera Data Visualization enables you to download the current sheet as a PDF file with each visual on a separate page.

About this task

In the following example, you can see how to download a PDF file with each visual of the current sheet on a separate page.

Before you begin

Before you download the dashboard as a PDF file, make sure that the Enable "Download as Image/PDF" option is enabled. You can find it in Site Administration Site Settings Snapshots .

Procedure

1. Open any dashboard. In this example, the PDF Export dashboard is used, which has two visuals.

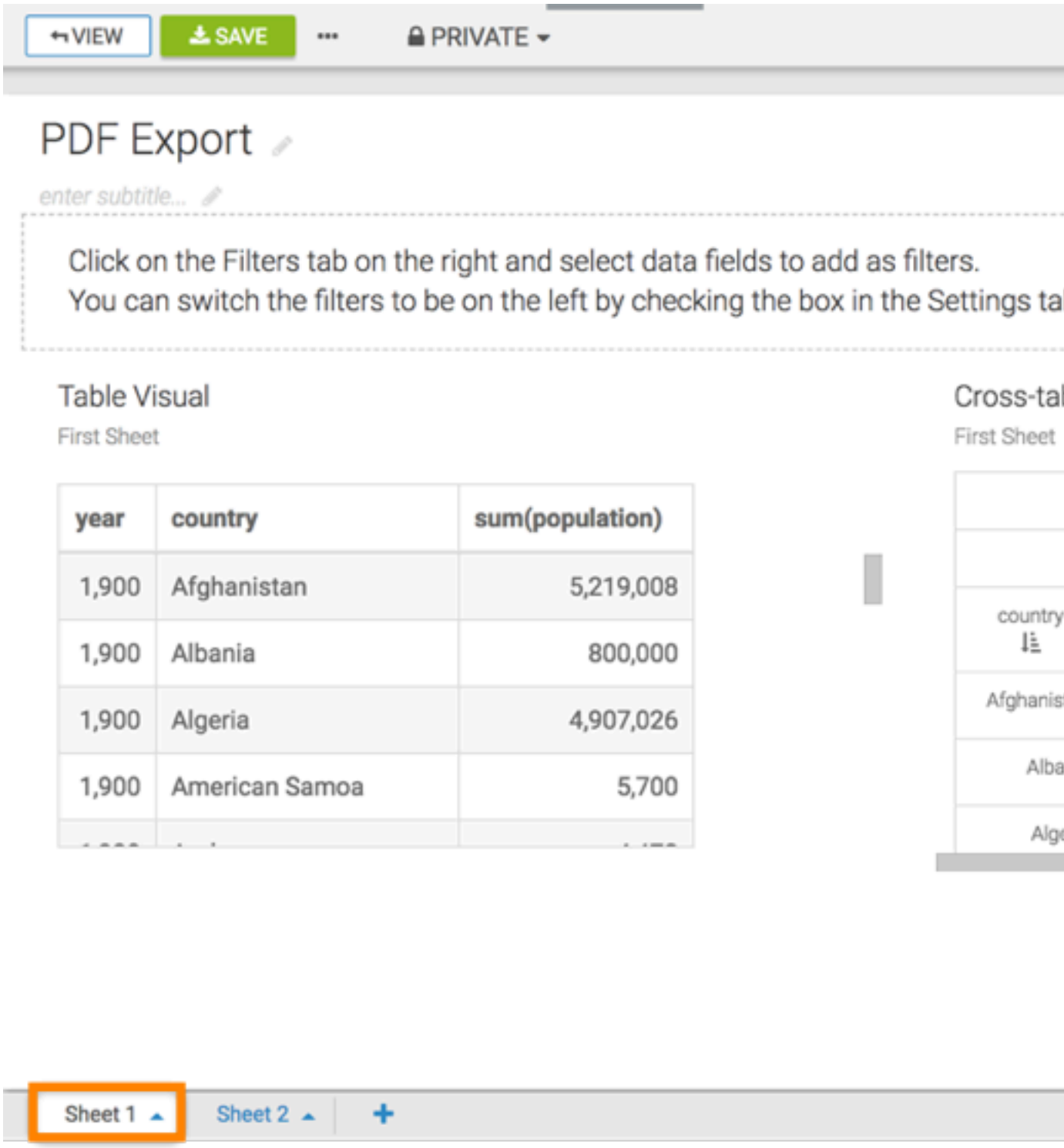

2. Click the (ellipsis) icon at the top left corner of the interface.

3. Select Download as... PDF One visual per page on current sheet .

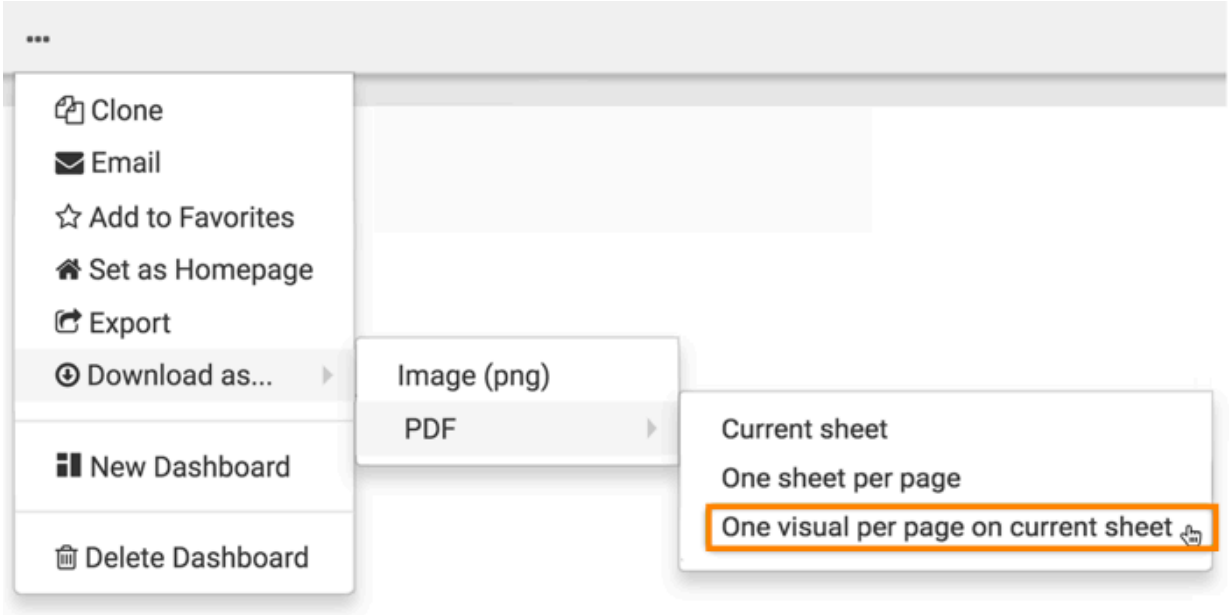

A Generating the requested file screenshot message appears.

4. After the download is complete, open the default download folder on your computer. Cloudera Data Visualization saves the current sheet to a PDF file.

   In this example, the generated filename, PDF Export_20190205142003, has the following components:

   - *PDF Export*

     Same name as the dashboard
   - *20190205*

     Signifies the date the file is created, in YYYYMMDD format
   - *142003*

     Signifies the time the file is created, in hhmmss format

   Note: Cloudera Data Visualization renames sheets with names that contain characters not supported by the file system. For example, for the sheet PDF/Export, it generates the file as PDF Export_xxxxxxxxx xxxxx.pdf, replacing the forward-slash (ASCII #47) character with a space.

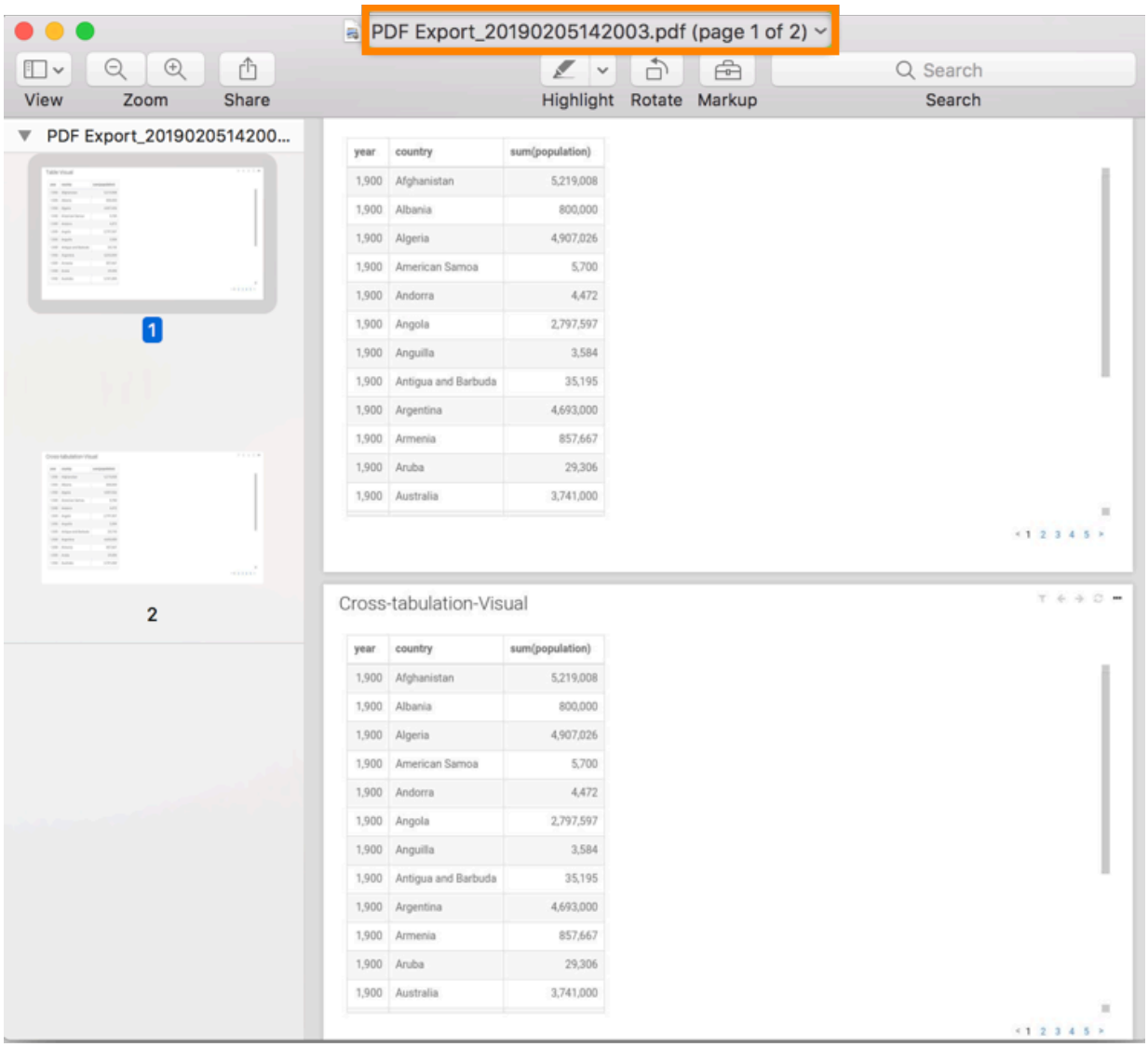

The filename is the same as the name of the dashboard, and it has a timestamp specifying the date and time when the file is created. Each visual in the current sheet is on a separate page.

Emailing sheets

Cloudera Data Visualization enables you to share dashboards through email. You can send them immediately, schedule them at specific times and intervals, or through a triggered response when reaching a specific monitored threshold.

Before you begin

Before using the email function, you must enable the feature in site settings. For instructions, see *Email settings*.

Note: This setting is only available for users with administrative privileges.

Complete these prerequisites in your system configuration:

- Enable screenshot and thumbnail downloads. For instructions, see *Snapshots*.
- Enable email functionality. For instructions, see *Email settings*.
- Enable scheduled jobs. For instructions, see *Enabling scheduled jobs*.

Note: You can only email one dashboard sheet at a time, but if you have more sheets on your dashboard, you can repeat this procedure for each sheet.

Procedure

1. Select the sheet you want to share in email by clicking on its tab at the bottom left corner of the Dashboard Designer interface.
2. In the top menu bar, click the ••• icon to open the supplemental menu and select Email.

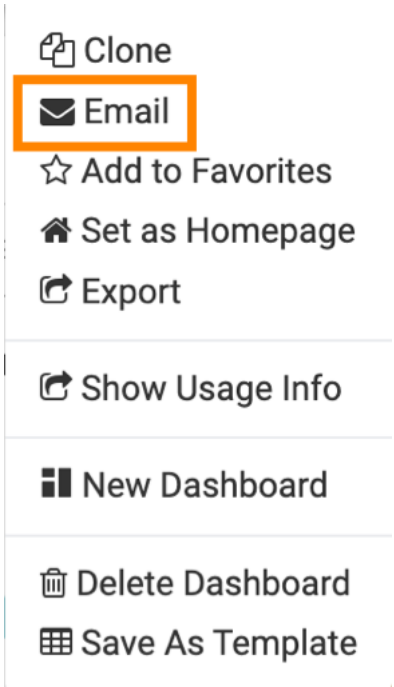

The Send sheet modal window appears.

3. Choose one of the two options for sending email.

   Email now – For instructions on how to provide email details, see *Immediate email*.

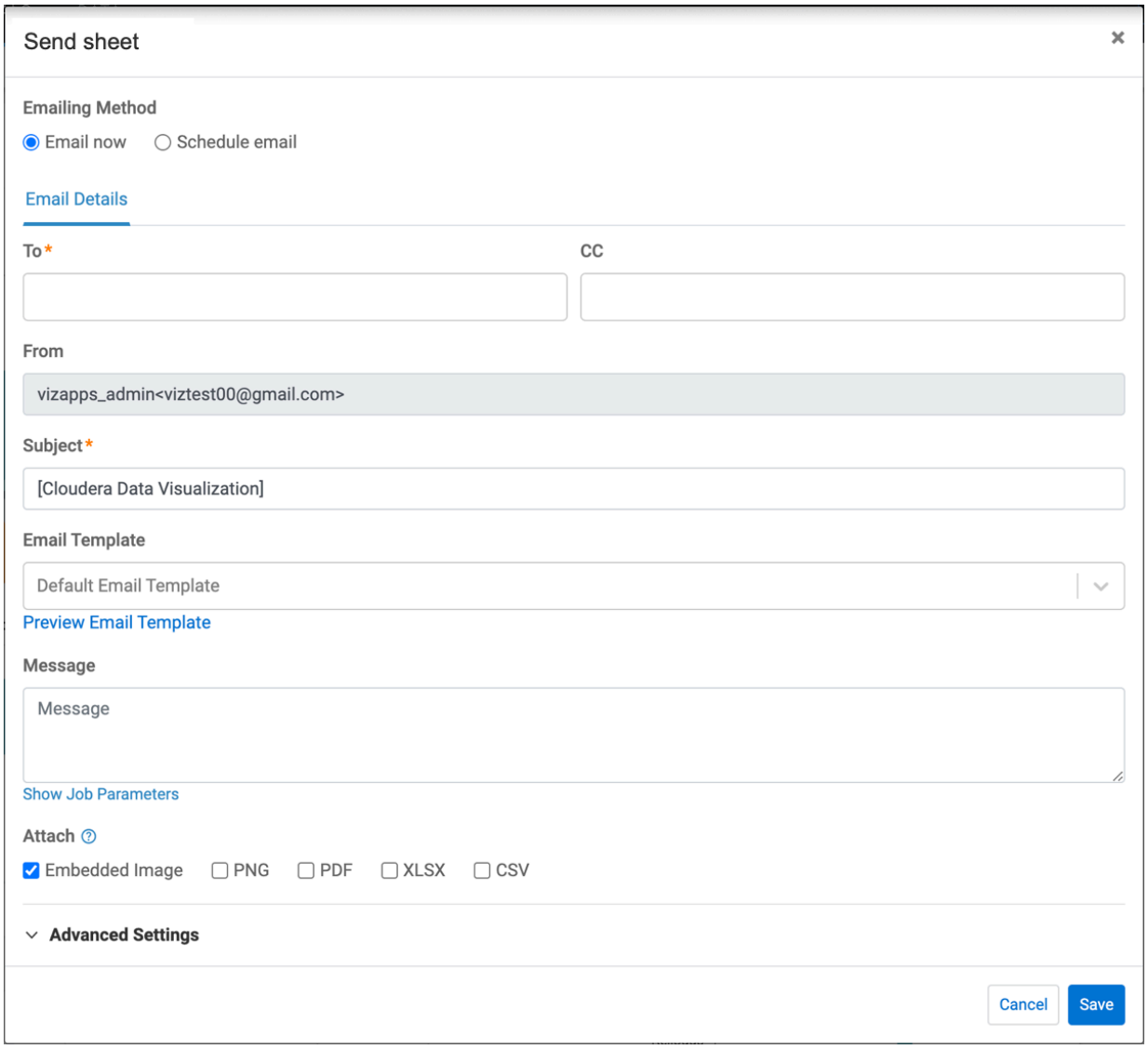

   Schedule email – For instructions on how to provide email and schedule details, see *Schedule-based email*.

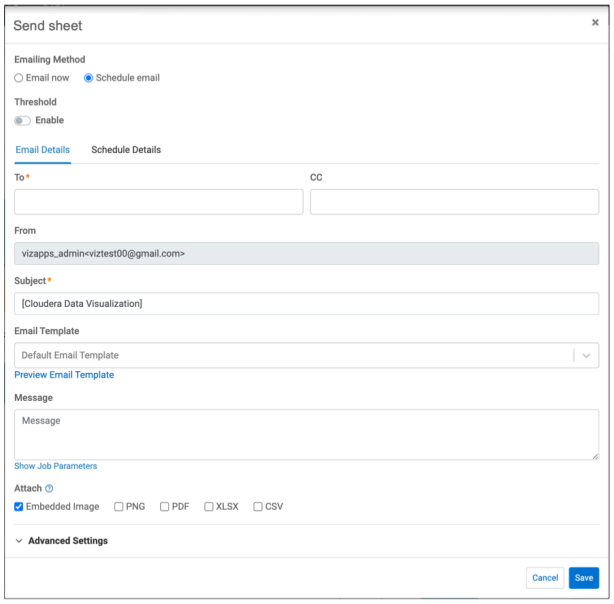

4. After creating emails, you can monitor them in the Jobs interface, review their details, rerun, cancel, and so on. For more information, see *Managing jobs*.

Related Information

Email settings
Snapshots
Enabling scheduled jobs
Immediate email
Schedule-based email
Threshold-based email
Managing jobs

Deleting a sheet

Cloudera Data Visualization enables you to delete sheets from a dashboard.

Procedure

1. Open a dashboard in Edit mode.
2. Click the icon on tab of the sheet that you want to remove.

3. In the sheet management menu, click Delete.

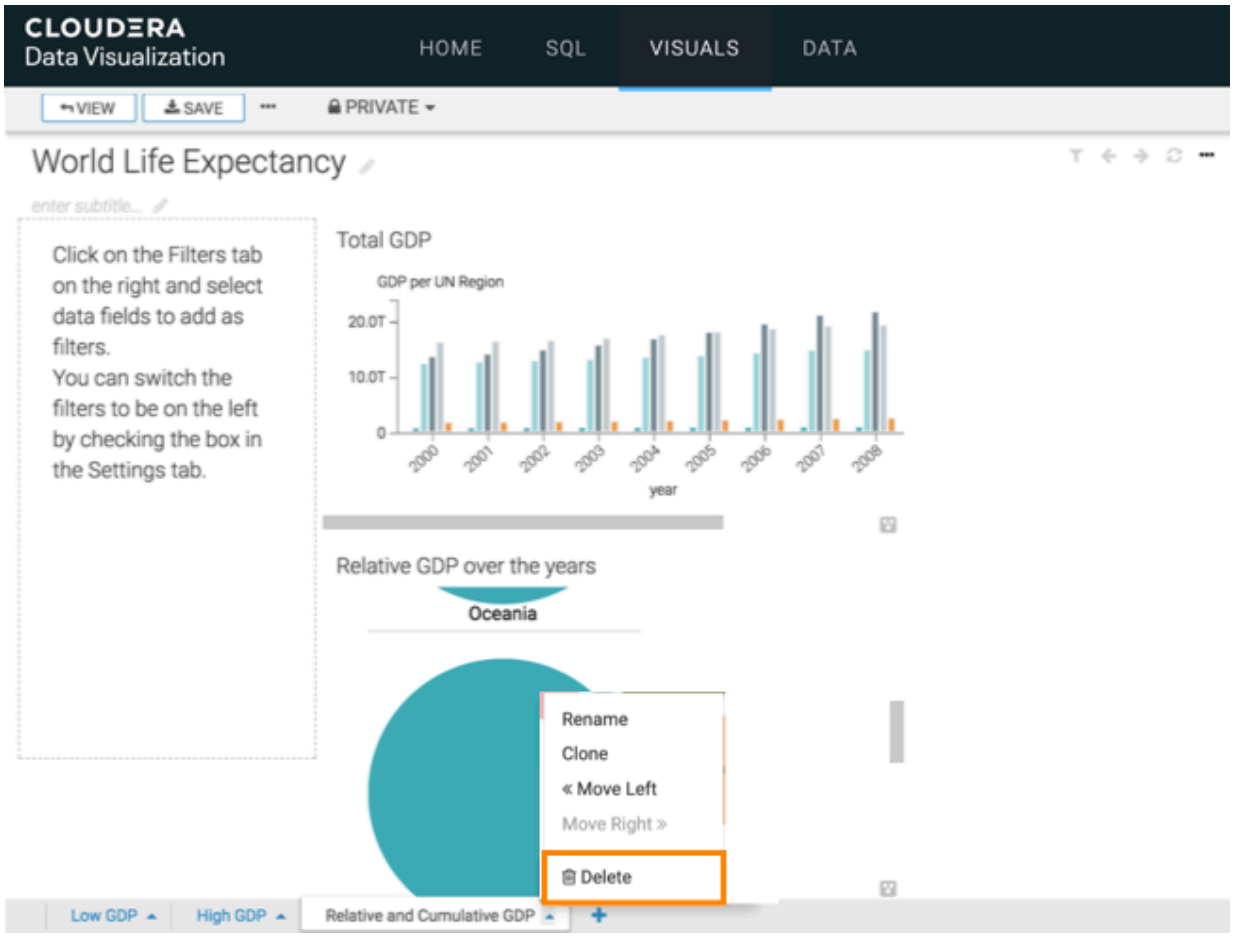

4. Click SAVE to save all changes made.

Results

After saving the dashboard, you can see that the deleted sheet is no longer visible.

Cloudera Data Visualization 7.2.9

Visualization

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a visual 7

Cloning a visual 10

Linking visuals 10

Unlinking visuals 12

Emailing a visual 12

Exporting a visual 15

Importing a visual 17

Downloading visuals 20
- Downloading a visual as PNG file 21
- Downloading a visual as PDF file 22
- Downloading visual data to CSV file 24
- Downloading visual data to Excel file 26

Removing a visual 30

Deleting a visual 30

Adding filters to a visual 31

Exploring visual options 33

Visual type catalog 35
- Table 37
  - Shelves for tables 45
  - Saving tables 46
- Cross tabulation 52
  - Displaying totals with data expansion 58
  - Shelves for cross tabulation 65
- Bars 65

Filtering 1: Bar chart by year....72
Filtering 2: bar chart by region....74
Horizontal bars....77
Stacked bars....79
Labelled bars....81
Shelves for bars....84
Lines....84
Lines with dual axes....89
Shelves for lines....91
Combined bar/line visuals....92
Creating basic combos....93
Assigning a measure to a secondary axis....97
Change line to area....99
Shelves for combos....99
Areas....100
Using the trellis option with areas....104
Using the trellis option with independent scale....105
Shelves for areas....106
Grouped bars....107
Shelves for grouped bars....111
KPI....112
Shelves for KPIs....117
Gauge....118
Shelves for gauges....127
Row listing....128
Shelves for row listing visuals....130
Bullet....130
Shelves for bullets....139
Box plot....140
Shelves for box plots....143
Packed bubbles....144
Shelves for packed bubbles....149
Word cloud....149
Trellised word clouds....153
Shelves for word clouds....154
Scatter....155
Creating a basic scatter visual....155
Adding size to scatter visuals....161
Adding transition animation to scatter visuals....164
Adding trellis to scatters....168
Shelves for scatters....170
Pie....171
Shelves for pies....175
Radial....176
Trellised radial....180
Shelves for radials....181
Chord....182
Shelves for chords....185
Network....186
Shelves for networks....193
Advanced options for networks....194
Dendrograms....202
Shelves for dendrograms....207
Treemap....207
Shelves for treemaps....211
Correlation heatmap....212

Shelves for correlation heatmaps.... 215
Correlation flow.... 216
Shelves for correlation flows.... 219
Calendar heatmaps.... 220
Shelves for calendar heatmaps.... 225
Maps.... 225
Choropleth country map.... 226
Country map with bubbles.... 231
Choropleth US state map.... 236
US state map with bubbles.... 240
Choropleth US county map.... 241
US county map with bubbles.... 243
US zipcode map with bubbles.... 245
Latitude and longitude map with bubbles.... 249
Choropleth DMA map.... 250
Choropleth Canadian province map.... 253
Canadian province map, bubbles.... 256
Choropleth Canadian postal code map.... 259
Canadian postal code map, bubbles.... 263
Shelves for maps.... 266
Interactive maps.... 267
Basic interactive map.... 268
Choropleth interactive maps.... 275
Interactive map with multiple dimensions.... 281
Interactive map with multiple measures.... 282
Changing the map server for interactive maps.... 284
Changing layer options for interactive maps.... 285
Plotting routes on interactive maps.... 286
Using alphabetic values in interactive maps.... 288
Segmenting data qualitatively in interactive maps.... 291
Shelves for interactive maps.... 294
Sparklines.... 295
Using sparklines.... 296
Shelves for sparklines.... 298
AI visual [Technical Preview].... 299
Shelves for AI visual.... 301
Action.... 302
Search.... 308
Timeline.... 309
Shelves for timelines.... 311
External link visuals.... 312
Setting up dynamic external links.... 314
Shelves for external links.... 317
Histograms.... 317
Specifying the bucket count.... 320
Specifying data range.... 322
Normalizing the histogram.... 325
Using the cumulative option.... 326
Shelves for histograms.... 328
Extensions.... 329
Examples of extensions.... 332
Rich texts.... 333
Inserting URL links.... 336
Inserting tables.... 337
Inserting images from local machine.... 337
Inserting images from static assets.... 339

Resizing images.....340
Inserting videos.....340
Field statistics.....342
SQL queries.....344

Creating a visual

You can easily create new visual representations for your data in Cloudera Data Visualization.

About this task

The following steps demonstrate how to create a new visual.

Procedure

1. On the main navigation bar, click VISUALS.
2. Click Create New Dashboard to open a new, blank dashboard.

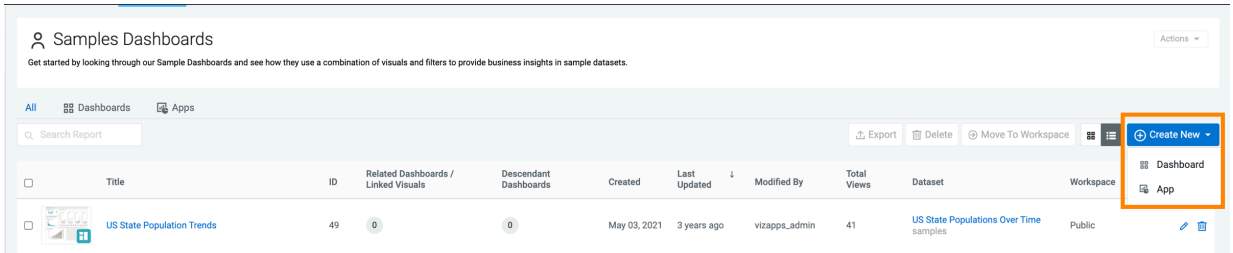

If you already have an existing dashboard and you just want to add a new visual, open the dashboard and click EDIT to make changes to it.

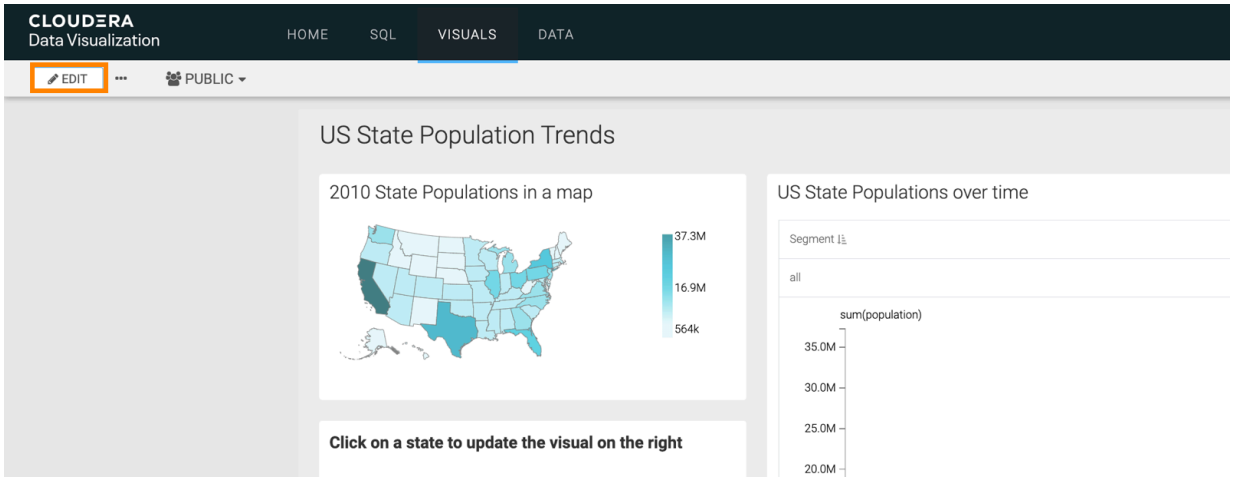

3. In the Dashboard Designer interface, open the Visuals menu from the side menu bar and click NEW VISUAL.

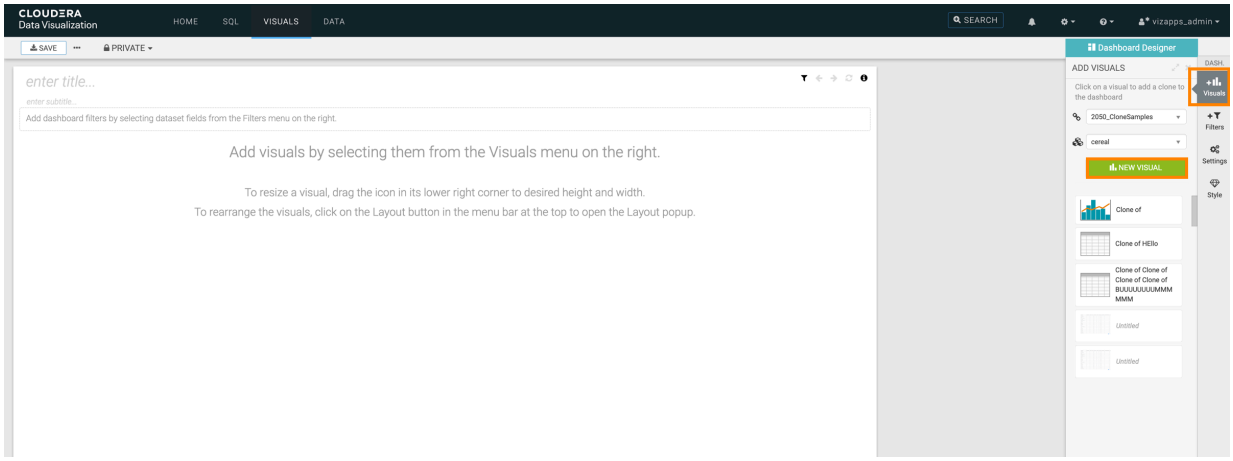

4. The Visual Builder appears.
   By default, a table visual is selected. For more information, see the *Visual Builder* overview.
5. Choose the visual type you want to build.
   In this example, the Bars visual has been selected.

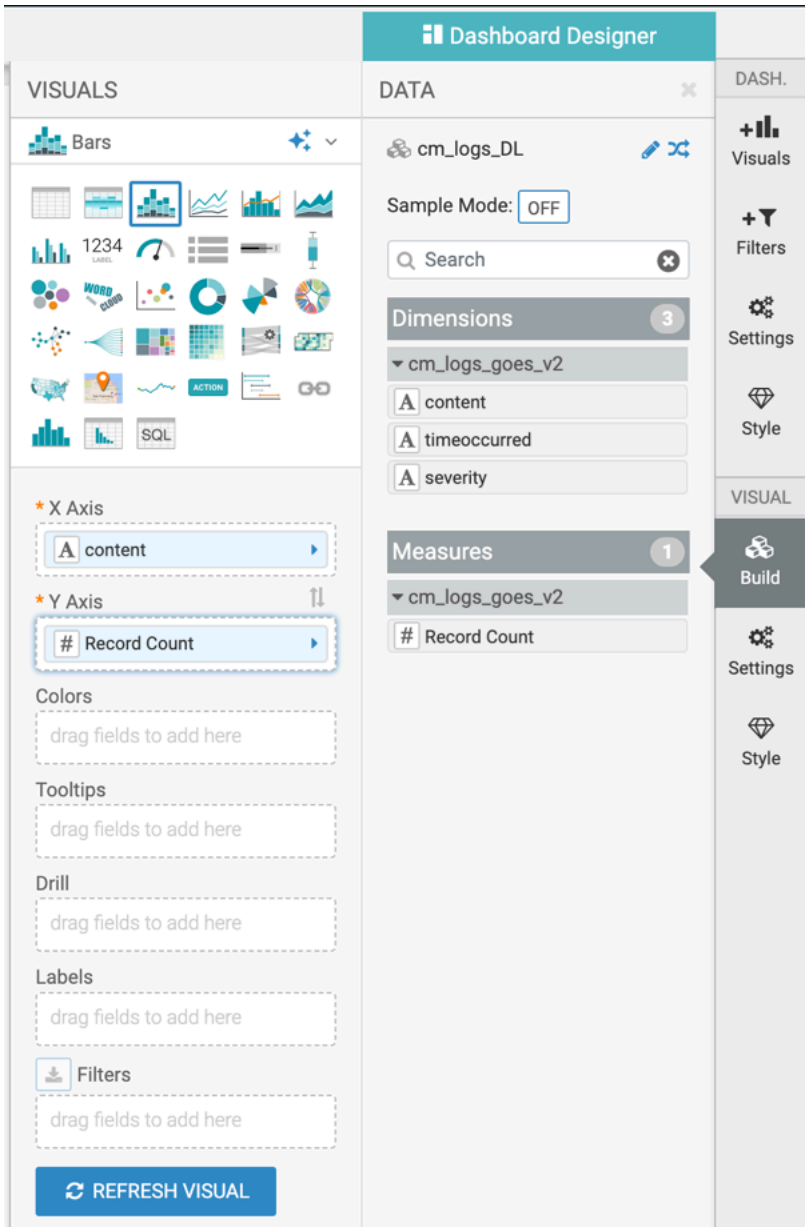

6. You can switch on Sample Mode for the visual you are building.

   With sample mode, you create the visual on a subset of your data to test it. You can define this subset as a percentage of the original data.

   Note: This feature is only available on Impala, Hive, and Spark connections.

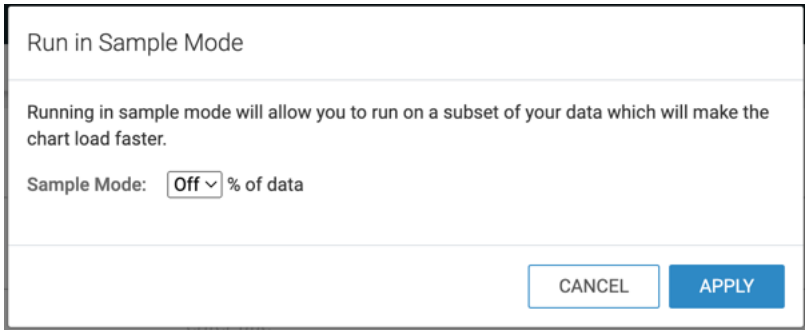

7. Populate the shelves of the visual from the Dimensions and Measures fields:

   In this example, the following fields have been added:

   - state dimension on the X Axis shelf
   - population measure on the Y Axis shelf

     Note: The population measure appears with its default aggregation, as sum(population).

   - year measure on the Colors shelf
   - year measure on the Filters shelf

     Note: The Filter for year module opens automatically. Years 1970, 1980, 1990, 2000, and 2010 have been selected.

8. Click REFRESH VISUAL.

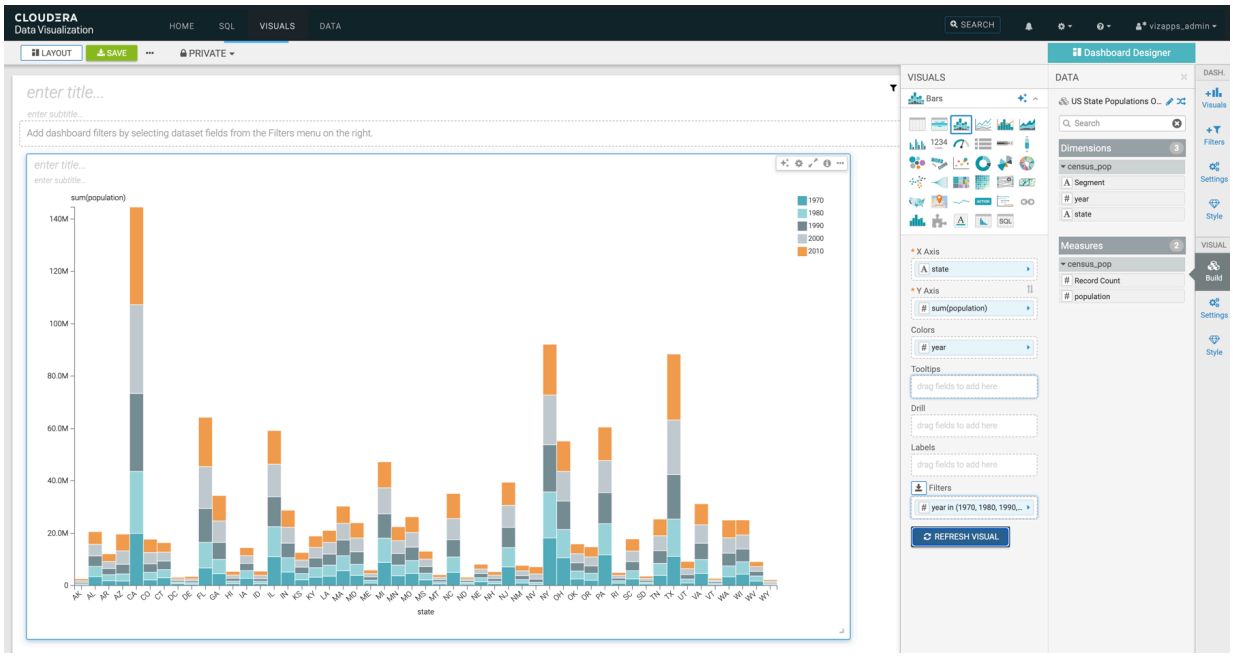

9. Click the enter title... field and enter a name for the visual.
   - In this example the title 'Recent US Census Results by State' has been added.
   - You can also click the enter subtitle... field below the title of the visualization to add a brief description of the visual.
10. Click SAVE at the top left corner of the Dashboard Designer.

Related Information

Visual Builder

Cloning a visual

In Cloudera Data Visualization, you can create a new visual that is a clone of an existing visual.

About this task

The following steps demonstrate how to clone an existing visual.

Procedure

In the Dashboard Designer, click the (ellipsis) icon at the top of the interface, and select Clone.

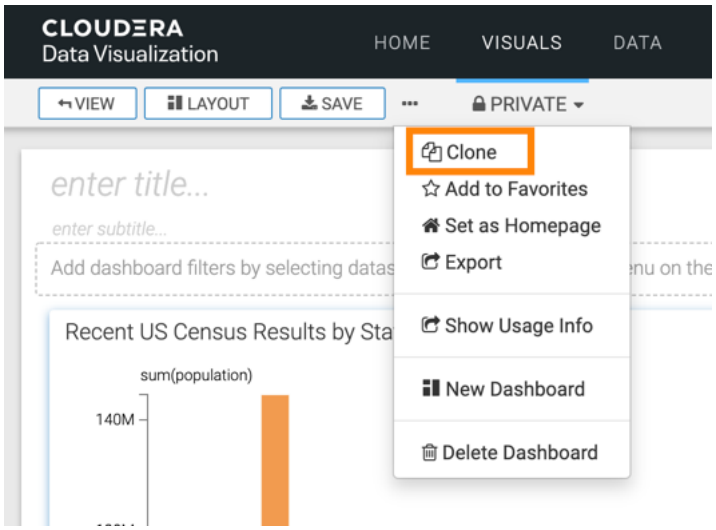

You should briefly see a green success notification. The main area of the screen now has a new visual titled Clone of...[Source Visual].

The SAVE button is not highlighted, because the process adds a cloned, unlinked version of the visual to the bottom of the dashboard. This cloned visual includes any setting or shelf changes you made before clicking Clone.

After working on the visual, you can rename and save it.

Linking visuals

In Cloudera Data Visualization, you can link visuals and these linked visuals can be shared across various dashboards and sheets within one dashboard. The following steps demonstrate how to make an existing visual as linked.

Procedure

1. Open a dashboard in Edit mode and hover over a visual until its control widgets appear.

2. Click the (ellipsis) icon at the top of the visual to open the supplemental menu.
3. Select Make Linked.

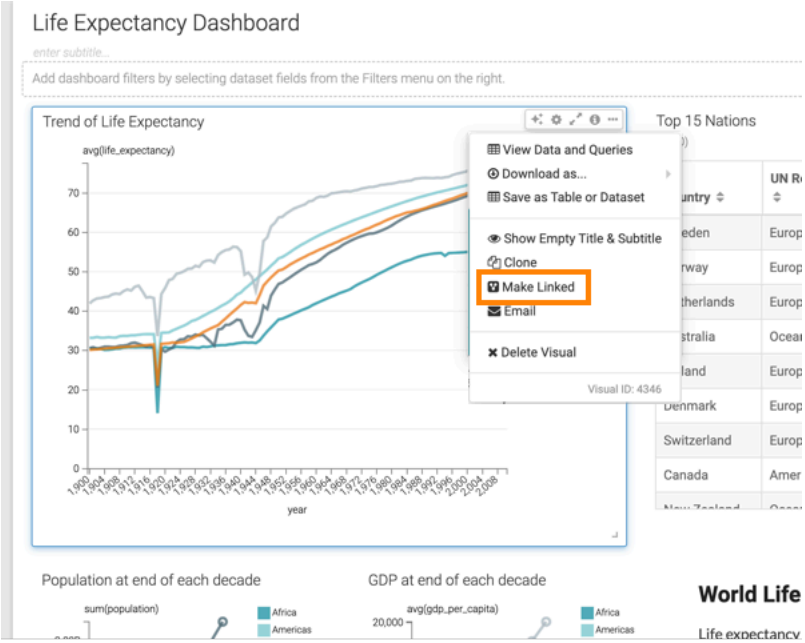

Note: Linked visuals cannot be deleted when they are dropped from all dashboards. They are still available to be added in future work.

4. Confirm the operation by clicking LINK.

Results

The visual can now be used in other dashboards. You can see a small icon in the lower right corner of the visual showing the linked designation.

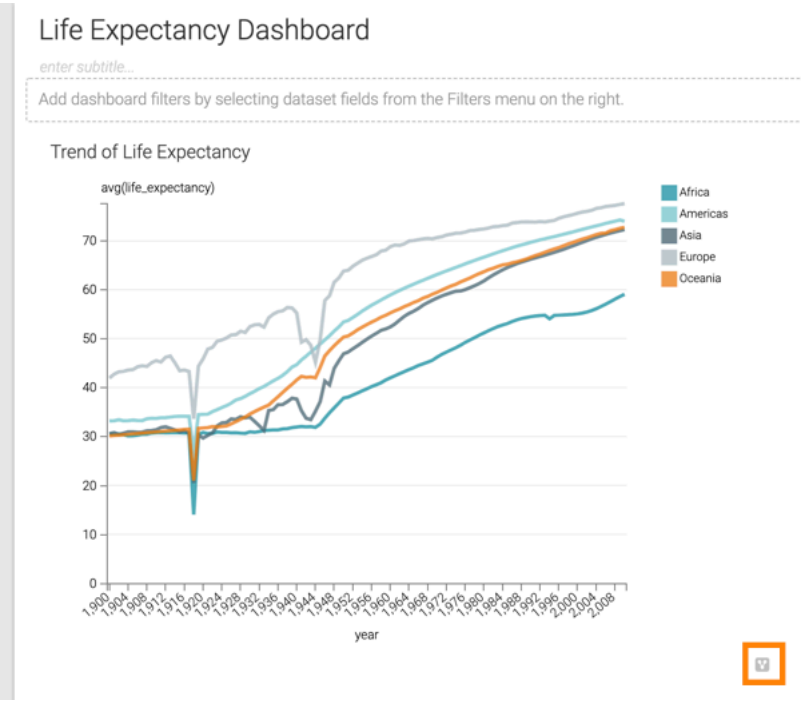

For instructions on how to add a linked visual to a dashboard, see *Adding visuals to a sheet*.

For information on how to unlink a visual, see *Unlinking visuals*.

Unlinking visuals

In Cloudera Data Visualization, you can unlink linked visual to break the sharing across dashboards.

About this task

The following steps demonstrate how to unlink a visual.

Procedure

1. In the Dashboard Desginer interface, hover over the visual until its control widgets appear.
2. Click the (ellipsis) icon at the top of the visual to open the supplemental menu.
3. Select Unlink.

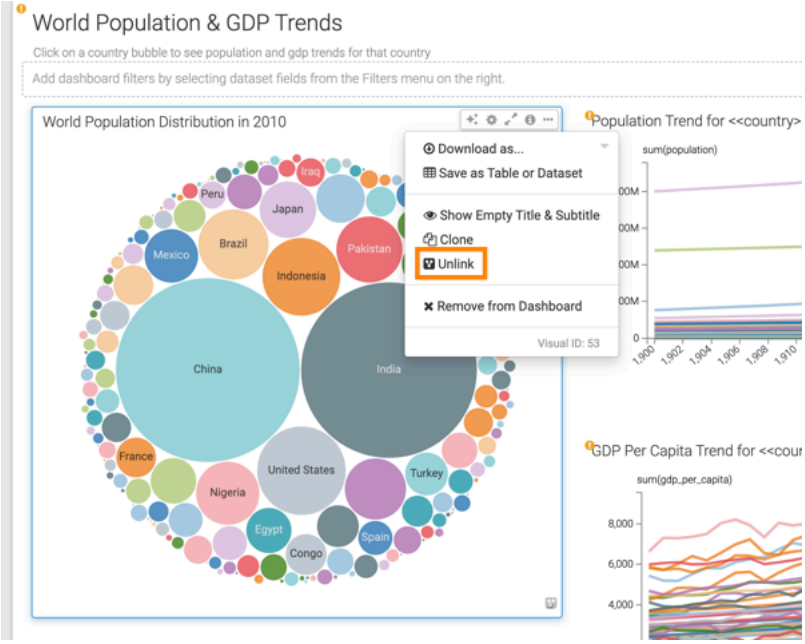

4. Confirm the operation by clicking UNLINK.

   When a visual appears in multiple dashboards, it cannot be unlinked. You can clone it instead. Confirm the operation by clicking CLONE.

   After you unlinked the visual by cloning it, it no longer has a linked designation. it is completely independent of the original visual, and its title is Clone of …[Original Name].
5. You can change the title of the visual to the original title, or use something different.
6. At the top left corner of the Dashboard Designer, click SAVE.

Results

The visual no longer has the small icon in the lower right corner showing the linked designation.

Emailing a visual

Cloudera Data Visualization enables you to share visuals in an email. You can send them immediately, schedule them at specific times and intervals, or through a triggered response when reaching a specific monitored threshold.

Before you begin

Before using the email function, you must enable the feature in site settings. For instructions, see *Email settings*.

Note: This setting is only available for users with administrative privileges.

Complete these prerequisites in your system configuration:

- Enable screenshot and thumbnail downloads. For instructions, see *Snapshots*.
- Enable email functionality. For instructions, see *Email settings*.
- Enable scheduled jobs. For instructions, see *Enabling scheduled jobs*.

Procedure

1. Find the visual you want to send in an email and open it in Edit or in View mode.
2. Click the ••• icon to open the supplemental menu in the top right corner of the visual and select Email from the drop-down.

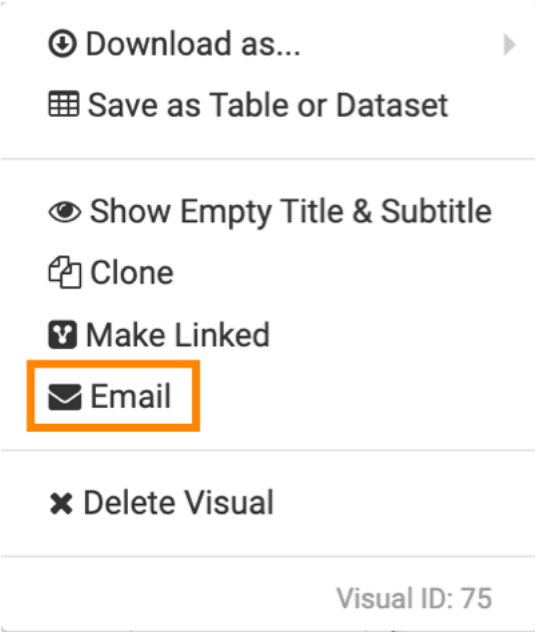

The Send visual modal window appears.

3. Choose one of the two options for sending email.

   Email now – For instructions on how to provide email details, see *Immediate email*.

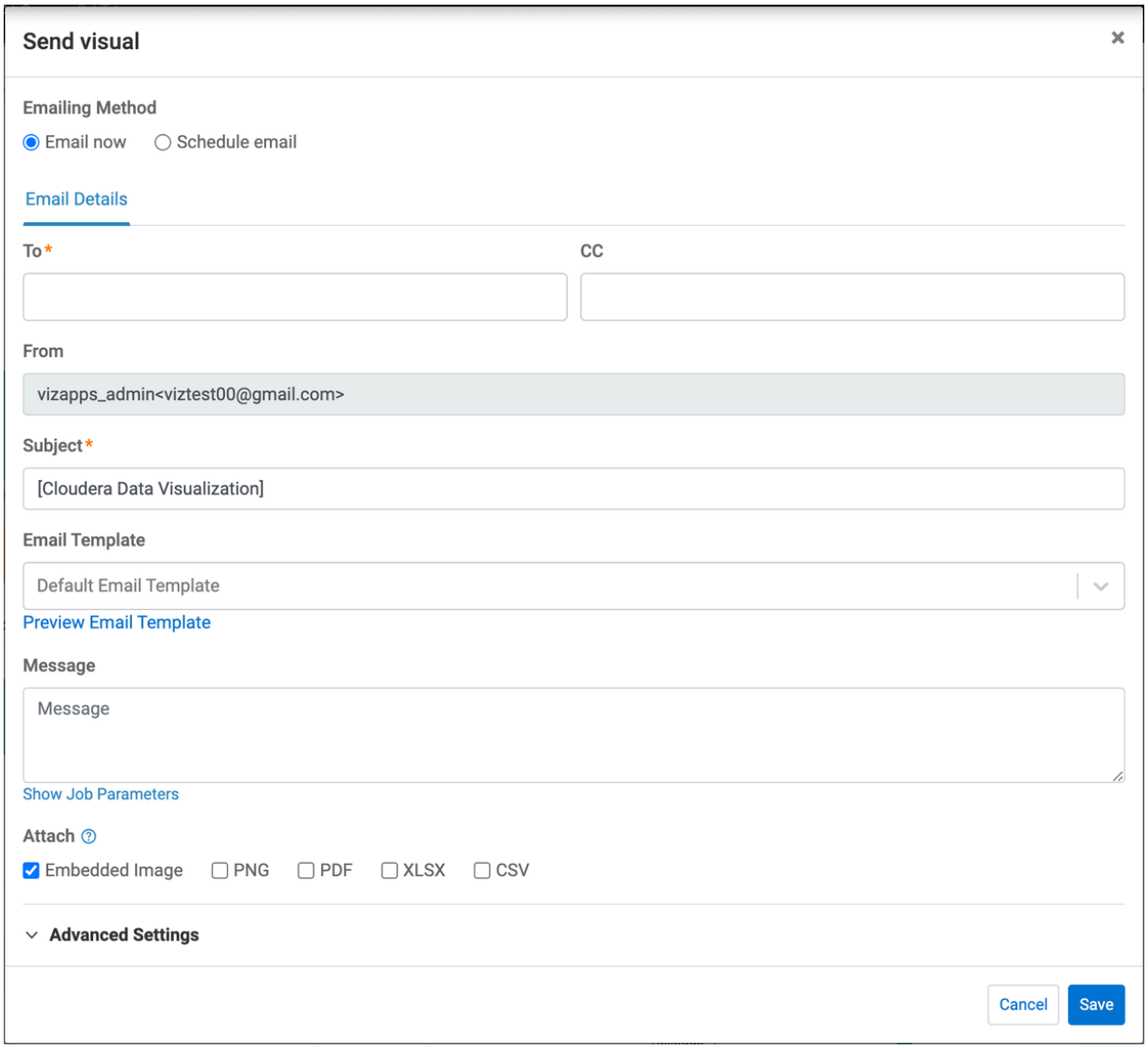

   Schedule email – For instructions on how to provide email and schedule details, see *Schedule-based email*.

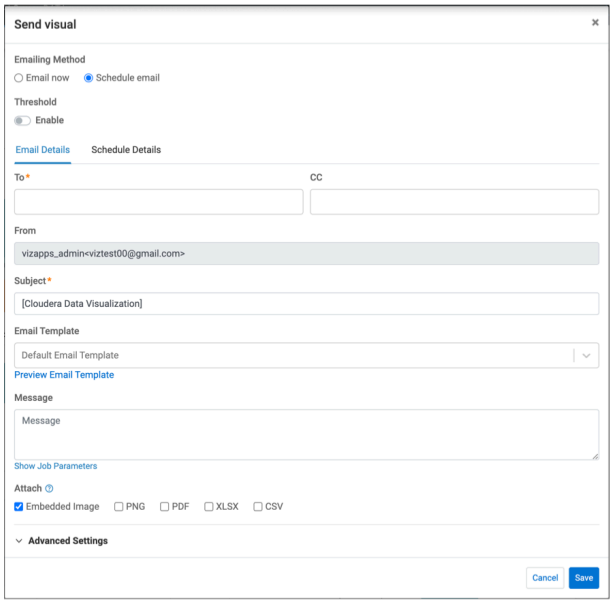

4. After creating emails, you can monitor them in the Jobs interface, review their details, rerun, cancel, and so on. For more information, see *Managing jobs*.

Related Information

Email settings
Snapshots
Enabling scheduled jobs
Immediate email
Schedule-based email
Threshold-based email
Managing jobs

Exporting a visual

In Cloudera Data Visualization, you can export selected visuals. The export process captures all the dependencies, such as datasets and custom styles.

About this task

To export a visual, follow these steps.

Procedure

1. In the top menu bar of the Dashboard Designer interface, click the (ellipsis) icon for the Supplemental menu.

2. Select Export.

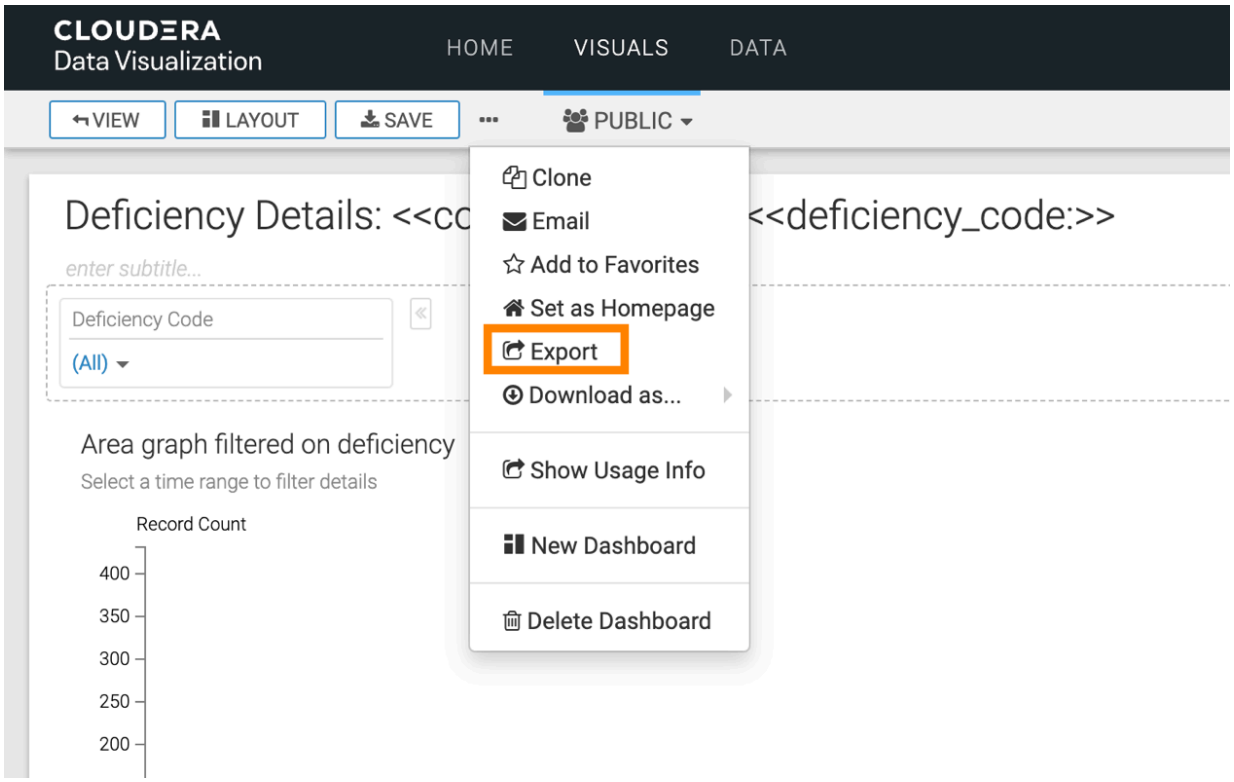

The Export Visual Artifacts modal window appears. It contains a default Filename of visuals_*RANDOM_NUMBER*. It also contains object information about the visual to be exported.

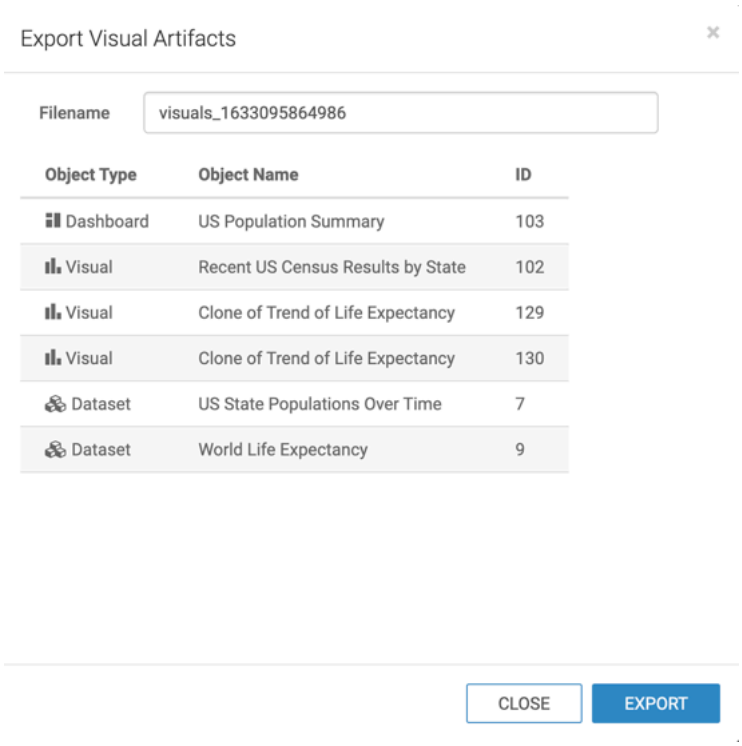

3. You can change the name of the file.

4. Click EXPORT.

   This starts the download of the export file to your default download folder. You can confirm the success of the export by locating the file in your system.

What to do next

For instructions on how to import a visual, see *Importing a visual*.

For more information about the visual artifact migration utility, see *Exporting and importing visual artifacts*.

Related Information

Importing a visual
Exporting and importing visual artifacts

Importing a visual

In Cloudera Data Visualization, you can import linked visuals from another system using an appropriate .json file that captures the definitions of these visual artifacts and all their dependencies (such as datasets and custom styles).

About this task

Follow these steps to import a visual.

Procedure

1. Click DATA in the Main navigation bar.
2. Select the connection where to import the artifacts.

   In this example the samples connection has been selected, showing connecting to the same data source for both systems.

3. At the top of the screen, click the (ellipsis) icon for the (supplemental) menu.

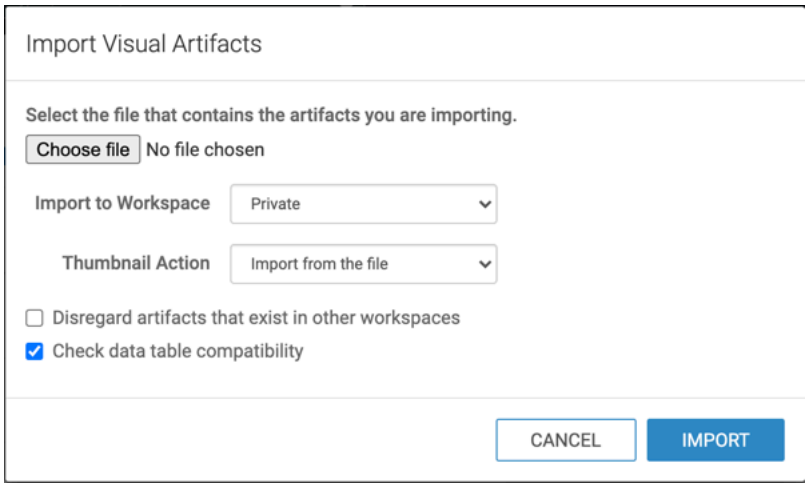

The Import Visual Artifacts modal window appears.

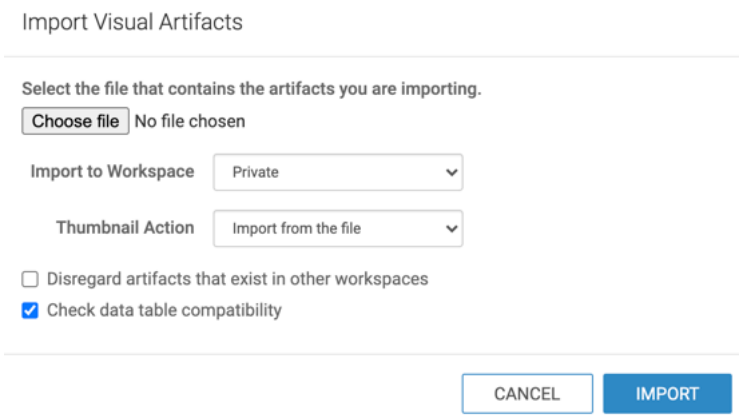

4. Click Choose File.

5. Select the file you want to import.

   In this example, visuals_flow_of_events.json is selected.

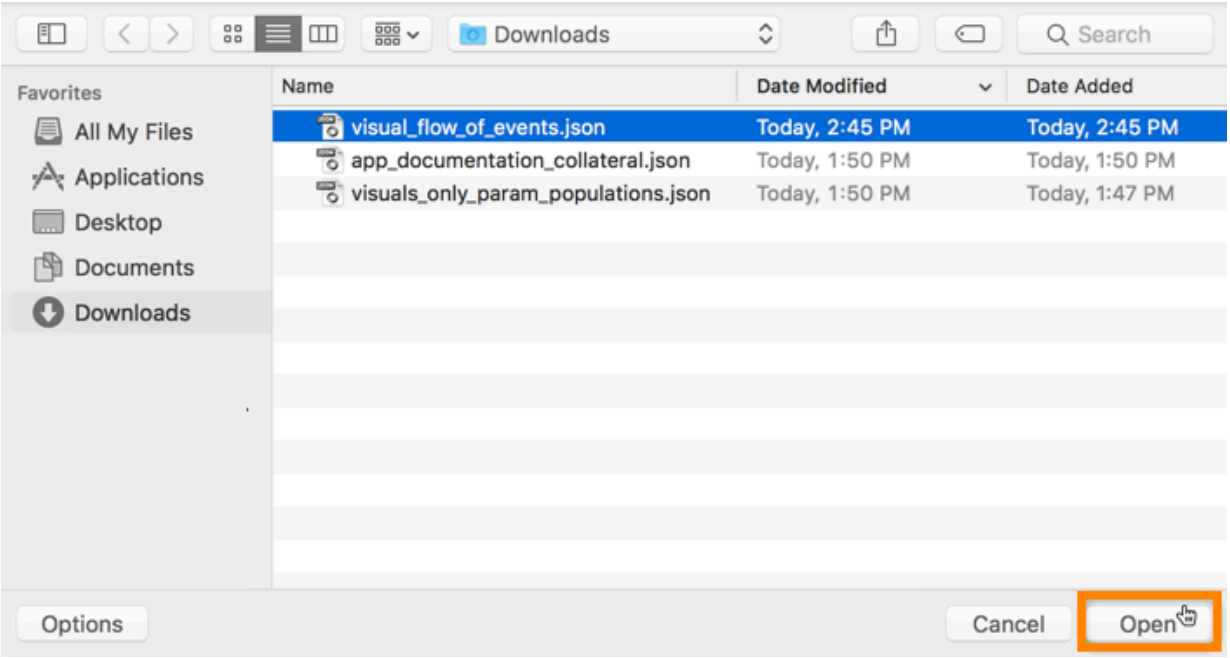

6. Click Open.
7. Specify the destination workspace in the Import to Workspace drop-down menu.

   By default, Cloudera Data Visualization imports the visual artifacts into your Private workspace. However, you can choose the Public workspace, or another workspace.

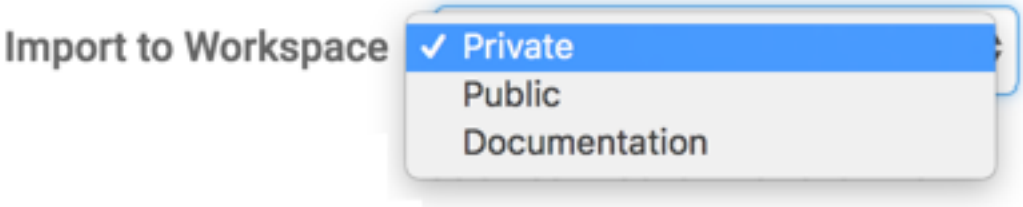

8. Specify how you want to handle thumbnails.

   By default, Cloudera Data Visualization imports the thumbnails from the file. However, you can choose to generate new thumbnails during the import process, or skip thumbnail import.

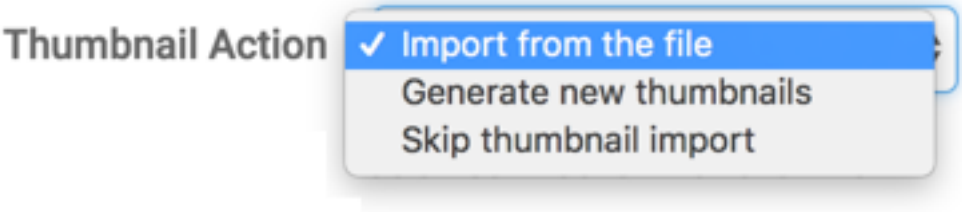

9. Define the Disregard artifacts that exist in other workspaces option.

   Before importing an artifact, Cloudera Data Visualization searches for it in the system. If it already exists, it is either updated (if changed from a previous import), or left 'as is'. If it does not exist, it is imported as a new visual artifact. By default, the search domain is the entire system. By selecting this option, you are limiting the search to the specified workspace.
10. Check data table compatibility.

    This is to confirm that tables suitable for the incoming dataset(s) exist on the system. It means that the database and tables have the same names on the source and target systems. This option is 'on' by default.

11. Click IMPORT.

After processing the import JSON file, Cloudera Data Visualization opens the Import interface to show its visual artifacts.

In this example, the Planned Action column shows that this is a new import for all artifacts. At this point, none of them have an assigned ID.

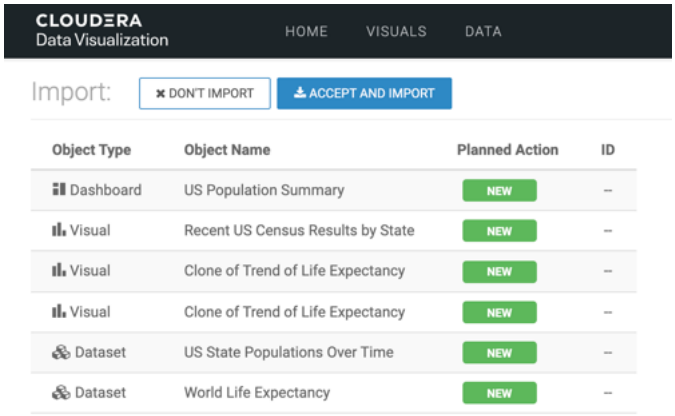

12. Click ACCEPT AND IMPORT.

Results

After the import completes, a success message appears on the Import interface. All artifacts have an assigned ID generated by the system sequentially. Visuals/Dashboards and datasets have separate ID queues.

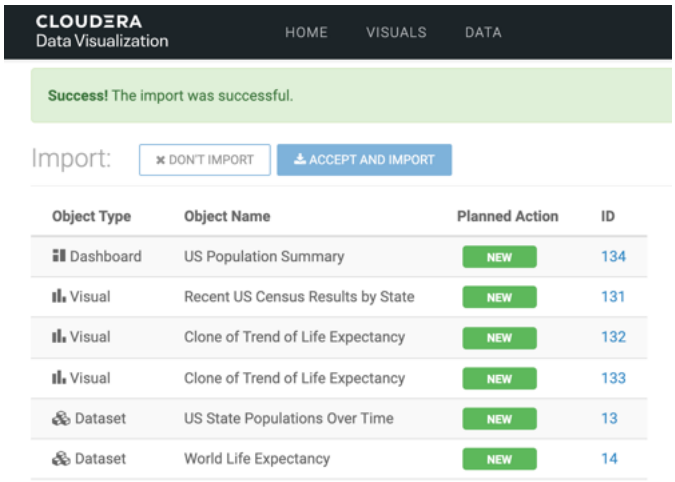

For instructions on how to export a visual, see *Exporting a visual*.

For more information about the visual artifact migration utility, see *Exporting and importing visual artifacts*.

Related Information

Exporting a visual

Exporting and importing visual artifacts

Downloading visuals

Cloudera Data Visualization enables you to download visuals in various file formats.

Downloading a visual as PNG file

Cloudera Data Visualization enables you to save a screenshot of a visual as a PNG file. The filename is the same as the name of the dashboard, and has a timestamp specifying the date and time when the file is created.

About this task

Cloudera recommends this method rather than using a desktop tool, because it handles visuals that are very long or very wide, and do not fit on the screen.

Note:

- This feature is available only when thumbnails are enabled. See *Enabling snapshots and thumbnails* for instructions.
- For snapshots of custom JS visuals, only ECMA Script 5 for Javascript is supported.
- PNG download on the Windows platform is not supported.

The following steps demonstrate how to save a visual as a PNG file.

Procedure

1. Open a visual in Edit mode, click the (ellipsis) icon at the top of the interface and select Download as... Image (png) .

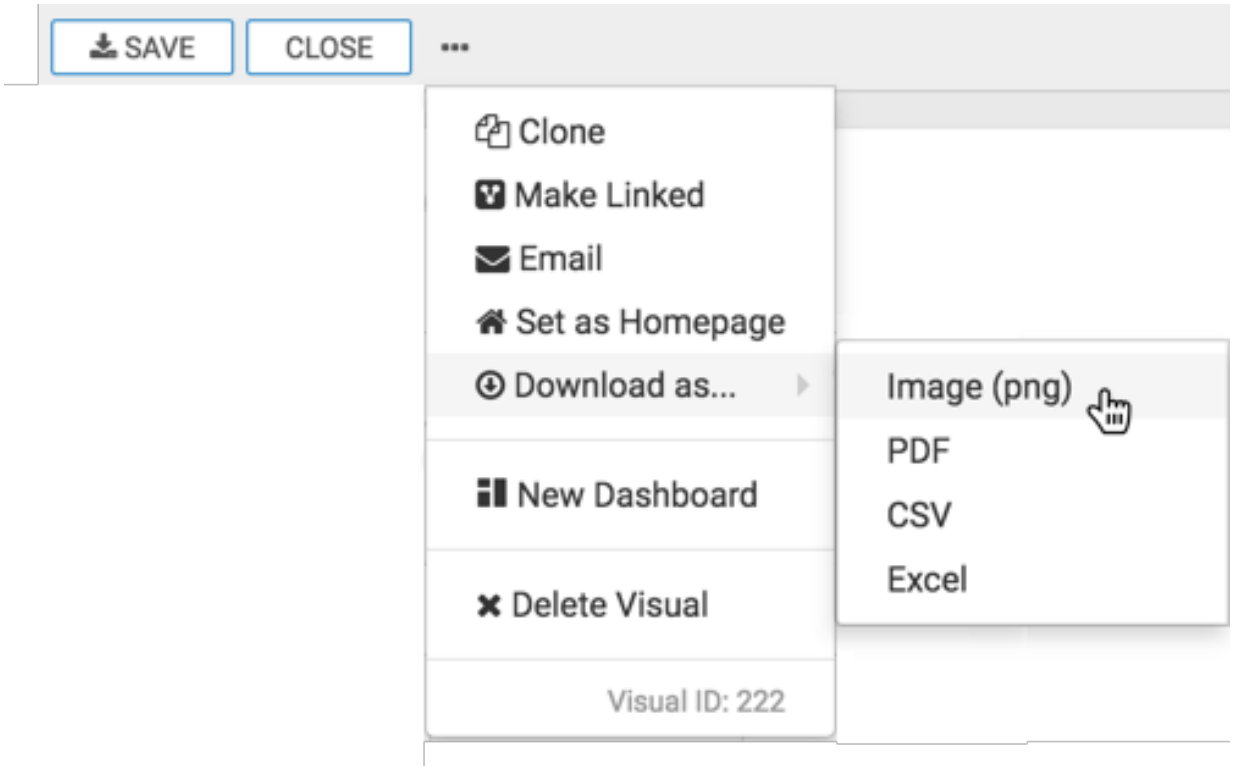

A Generating the requested file message appears. Cloudera Data Visualization saves the current sheet to a PNG file.

2. After the download is complete, open the default download folder on your computer.

   In this example, the generated filename (Table_Visual_20190207105500), has the following components:

   Table_Visual

   Same name as the dashboard

   20190207

   Shows the date the file is created, in YYYYMMDD format

   105500

   Shows the time the file is created, in hhmmss format

   Cloudera Data Visualization renames sheets that have names with characters not supported by the file system. For example, for the sheet Table/Visual, it generates the file as Table_Visual_xxxxxxxxxxxxxx.png, replacing the forward-slash (ASCII #47) character with an underscore (ASCII #95) character.

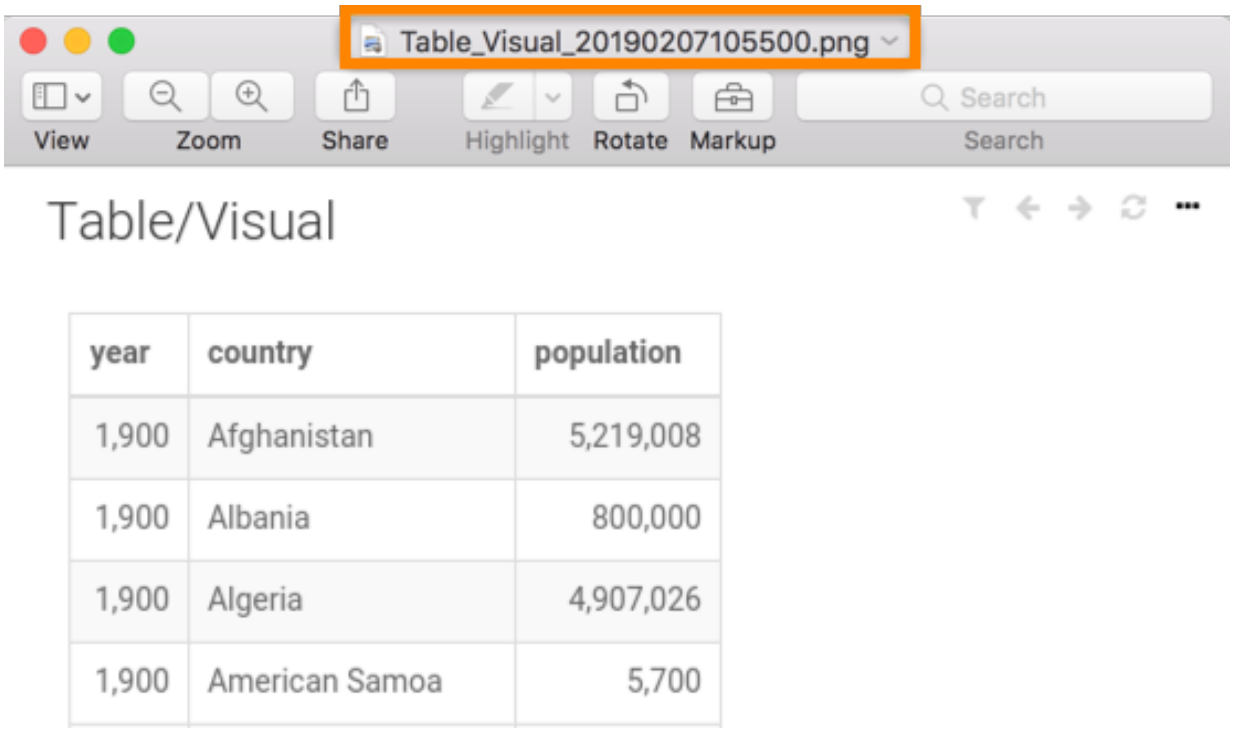

Downloading a visual as PDF file

Cloudera Data Visualization enables you to save a screenshot of a visual as a PDF file. The filename is the same as the name of the visual, and has a timestamp specifying the date and time when the file is created.

About this task

Note:

- This feature is available only when thumbnails are enabled. See *Enabling snapshots and thumbnails* for instructions.
- For snapshots of custom JS visuals, only ECMA Script 5 for Javascript is supported.
- PDF download on the Windows platform is not supported.

Before you begin

Before you download the visual as a PDF file, navigate to the General menu, and select the Enable "Download as PDF" option.

Procedure

1. Open a visual in Edit mode, click the (ellipsis) icon at the top of the interface and select Download as... PDF .

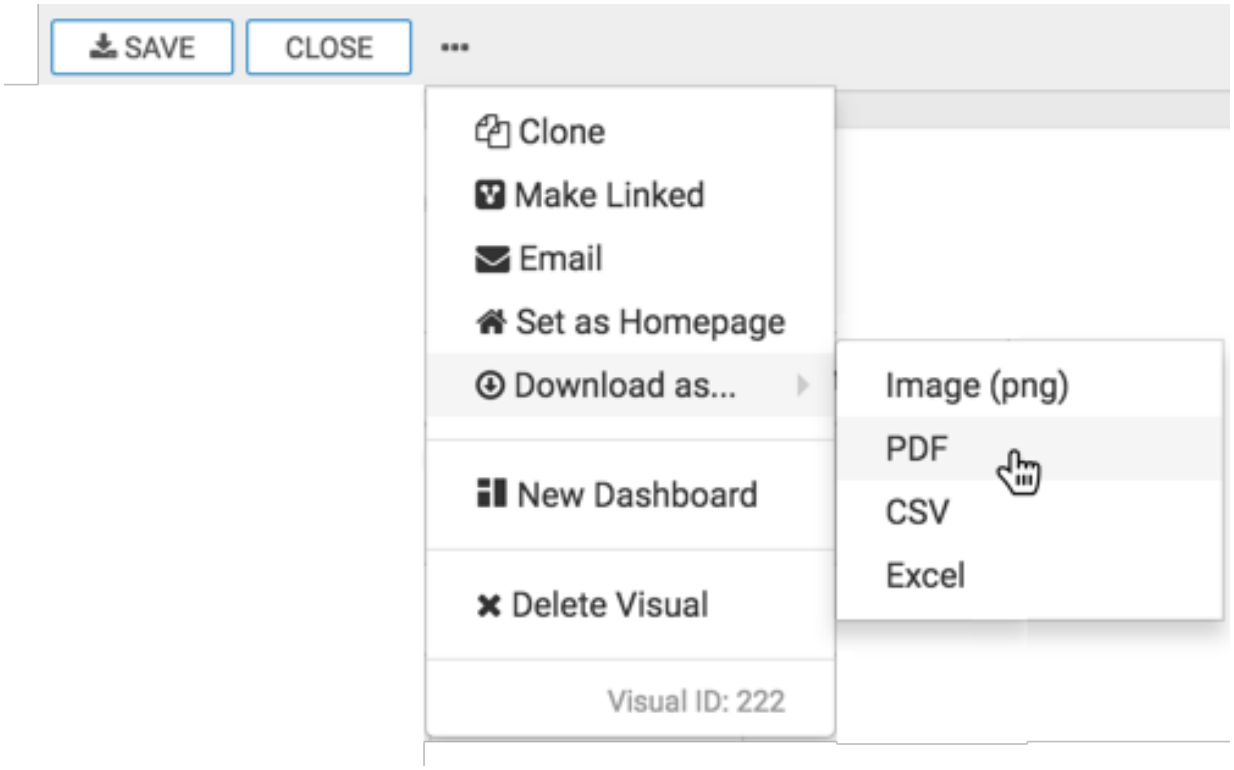

A Generating the requested file message appears. Cloudera Data Visualization saves the current sheet to a PDF file.

2. After the download is complete, open the default download folder on your computer.

   In this example, the generated file name, PDF Export_20190207105249, has the following components:

   PDF Export

   Same name as the dashboard

   20190207

   Shows the date the file is created, in YYYYMMDD format

   105249

   Shows the time the file is created, in hhmmss format

   Cloudera Data Visualization renames sheets that have names with characters not supported by the file system. For example, for the sheet Table/Visual, it generates the file as Table_Visual_xxxxxxxxxxxxxx.pdf, replacing the forward-slash (ASCII #47) character with an underscore (ASCII #95) character.

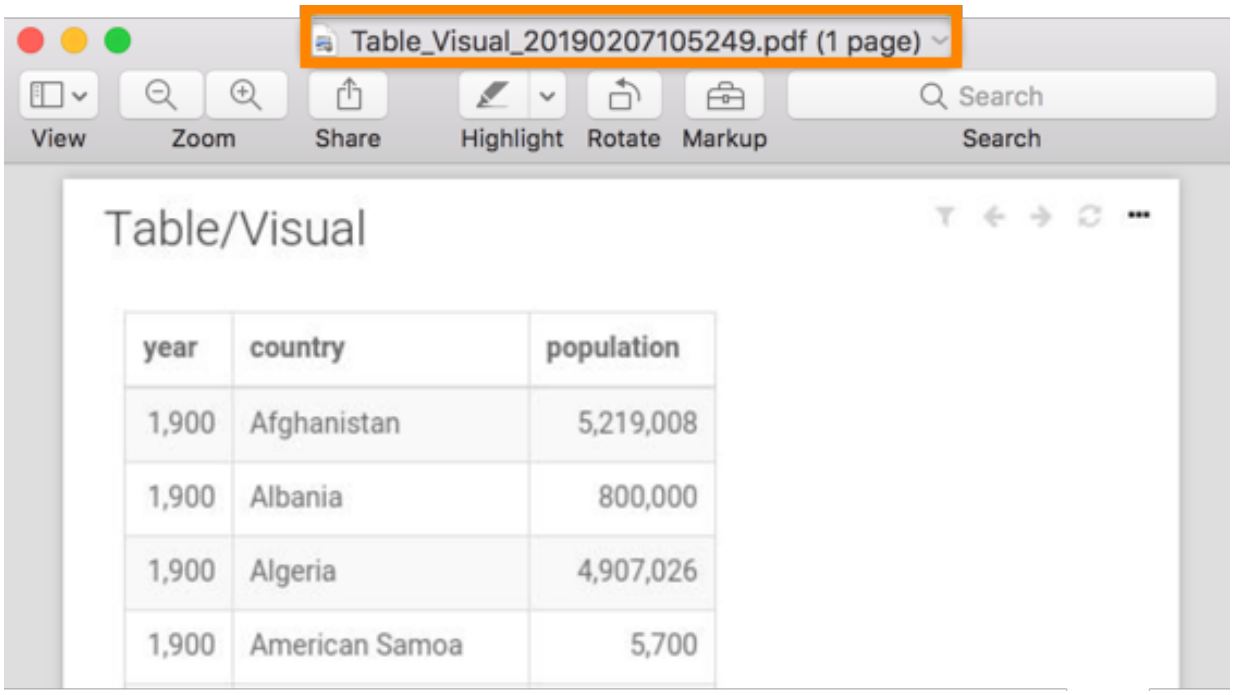

Downloading visual data to CSV file

Cloudera Data Visualization enables you to save the data used in visualizations as CSV files. The filename is the same as the name of the visual, and has a timestamp specifying the date and time when the file is created.

Before you begin

Before you begin to download a CSV file, you must first *Enable CSV Download*. The site administrator can override this setting in the Enable "Download as CSV/Excel" by default in the Site Settings interface. See *Downloading maximum number of rows in CSV files*.

Procedure

1. Open a visual in Edit mode, click the (ellipsis) icon at the top of the interface and select Download as... CSV .

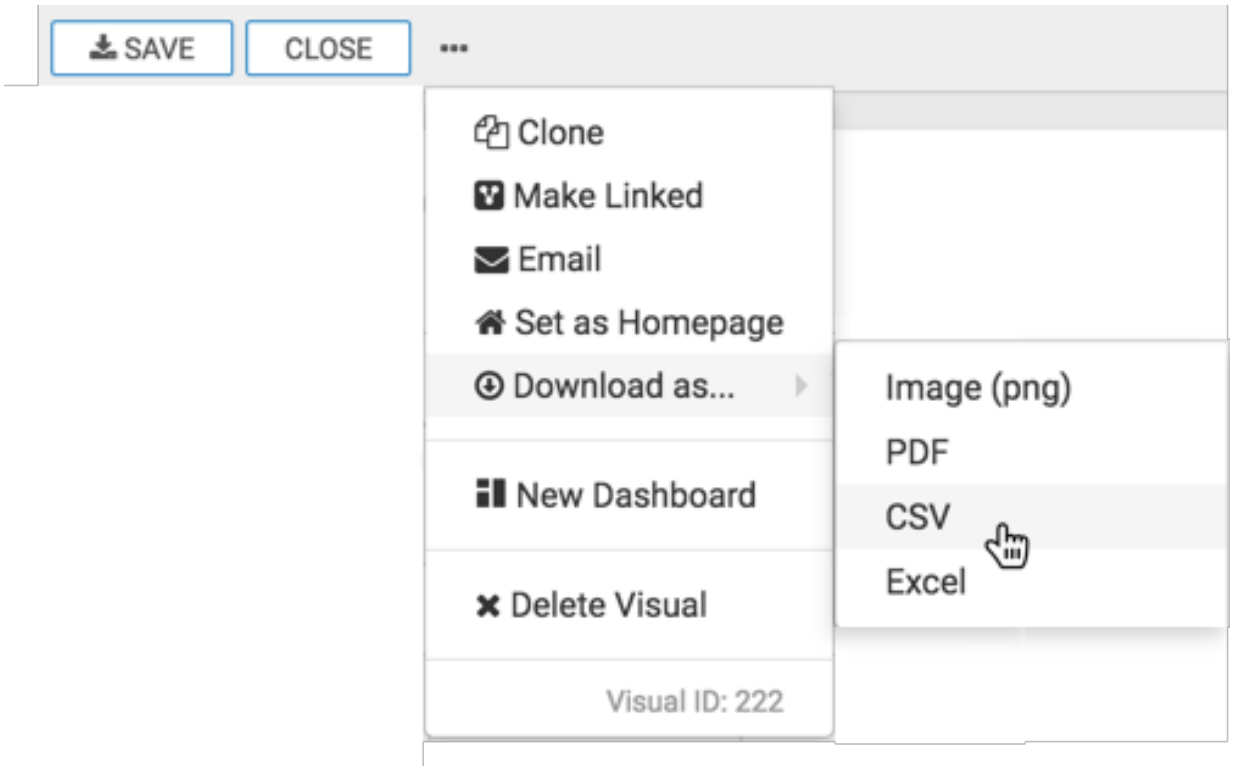

A Generating the requested file message appears. Cloudera Data Visualization saves the current sheet to a CSV file.

2. After the download is complete, open the default download folder on your computer.

   In this example, the generated file name (Table_Visual_20190207110756) has the following components:

   Table_Visual

   Same name as the dashboard

   20190207

   Shows the date the file is created, in YYYYMMDD format

   110756

   Shows the time the file is created, in hhmmss format

   Cloudera Data Visualization renames sheets that have names with characters not supported by the file system. For example, for the sheet Table/Visual, it generates the file as Table_Visual_xxxxxxxxxxxxxx.csv, replacing the forward-slash (ASCII #47) character with an underscore (ASCII #95) character.

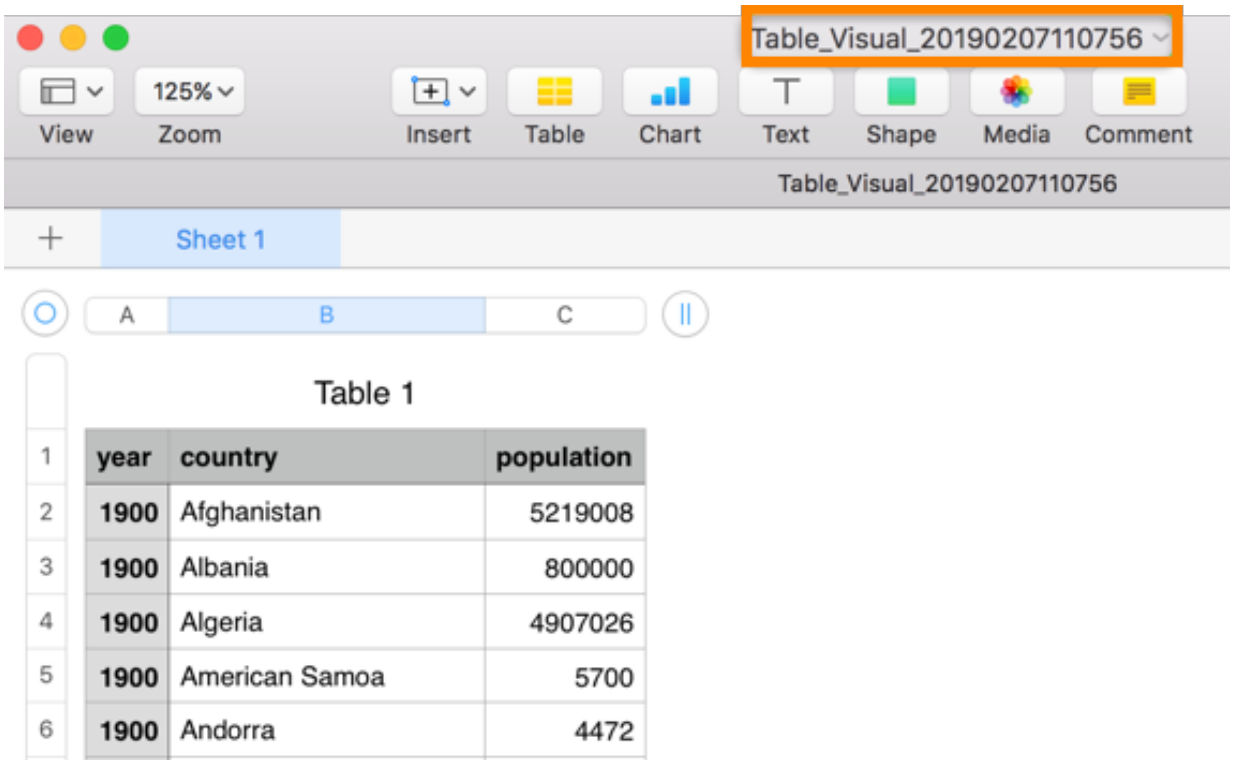

Downloading visual data to Excel file

Cloudera Data Visualization enables you to save data in your visuals as an Excel file. The filename is the same as the name of the visual, and has a timestamp specifying the date and time when the file is created.

About this task

Excel download is available on all visuals. In the following example, we demonstrate how to download visual data to an Excel file. The resulting Excel worksheet also contains a separate sheet showing parameter definitions.

The example uses a table visual with two filters. Year in 1900, 1901 and country in Afghanistan and Albania. To show totals in the visual, enable *Showing column totals*.

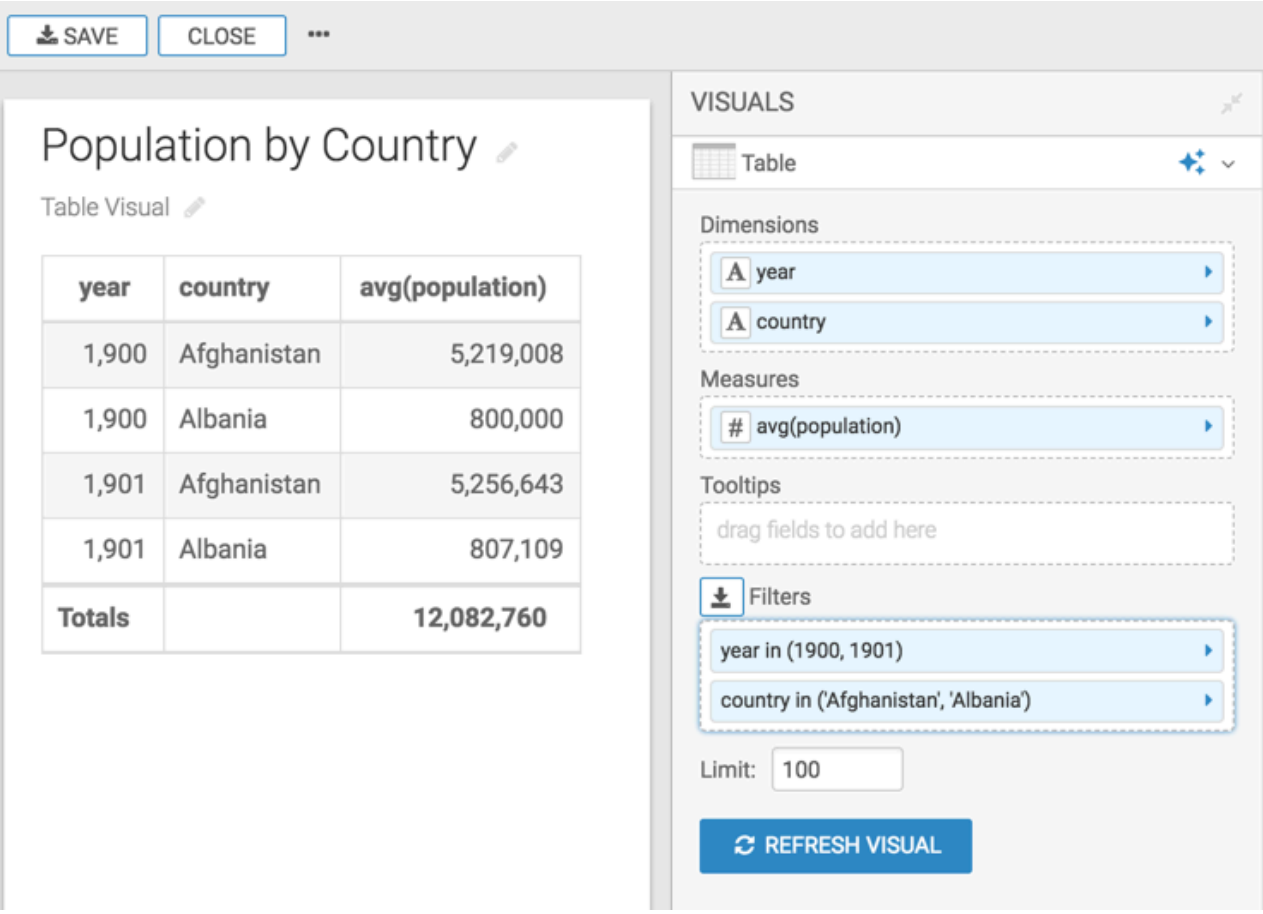

Before you begin

Before you begin to download a CSV file, you must *Enable CSV Download*. `The site administrator can override this setting in the Enable "Download as CSV/Excel" by default in the Site Settings interface. See *Downloading maximum number of rows in CSV files*.

Procedure

1. Open a visual in Edit mode, click the (ellipsis) icon at the top of the interface and select Download as... Excel .

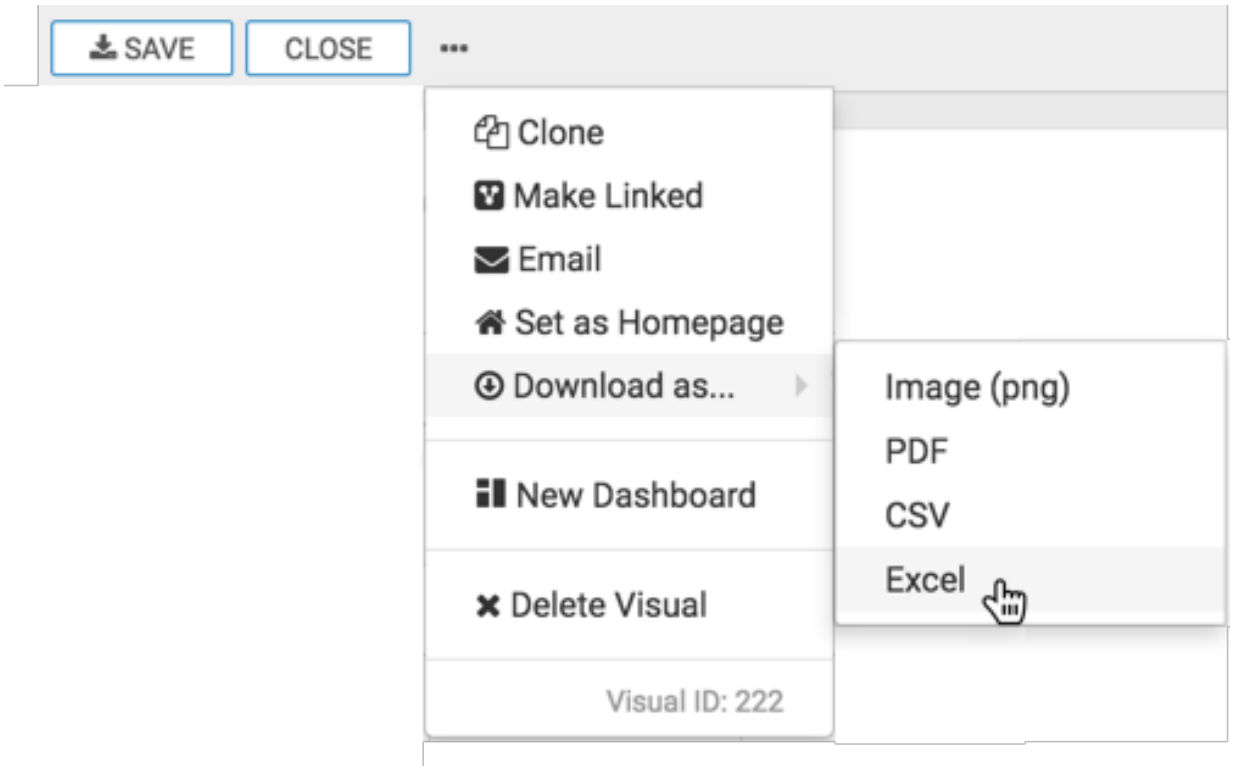

A Generating the requested file message appears. Cloudera Data Visualization saves the current sheet to an Excel file.

2. After the download is complete, open the default download folder on your computer.

   In this example, the generated file name (Population by Country_20190207113450) has the following components:

   Population by Country

   Same name as the visual

   20190207

   Shows the date the file is created, in YYYYMMDD format

   113450

   Shows the time the file is created, in hhmmss format

   Totals appear in the table only if you enable totals, as described in *Showing column totals*.

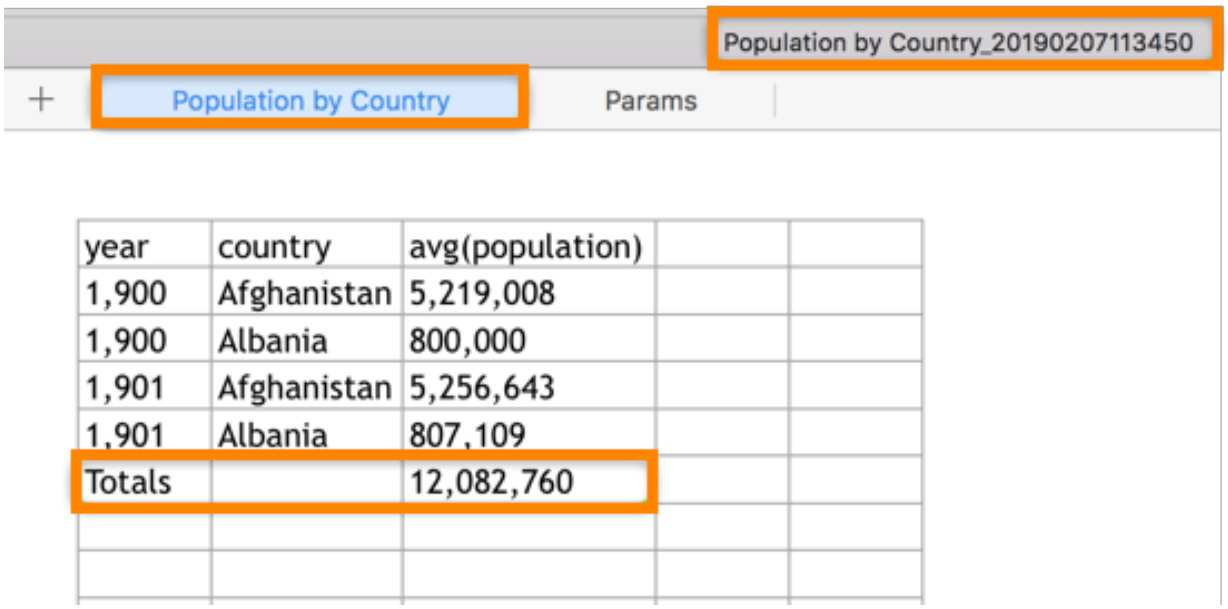

year	country	avg(population)		
1,900	Afghanistan	5,219,008		
1,900	Albania	800,000		
1,901	Afghanistan	5,256,643		
1,901	Albania	807,109		
Totals		12,082,760		

3. Click the Params sheet in the Excel file to see the two parameters that were defined earlier to create the visual.

year.data	year.exclude	country.data	country.exclude
1900	in	Afghanistan	in
1901		Canada	
		Bhutan	

Removing a visual

Cloudera Data Visualization enables you to remove a linked visual from a dahsboard in case you do not need it any more.

About this task

These instructions show you how you can remove a linked visual from a dashboard. When you remove a linked visual from the dashboard, you should use the 'remove...' menu option. After removal, the visual is still available in Cloudera Data Visualization and you can add it to other dashboards.

When you remove an unlinked visual from the dashboard, you should use the 'delete...' menu option. After deletion, the visual no longer exists in the system. For more information, see *Deleting visuals*.

Procedure

1. Open the dashboard in Edit mode in the Dashboard Designer.
2. Click the (ellipsis) icon at the top right corner of the linked visual.
3. Select Remove from Dashboard.

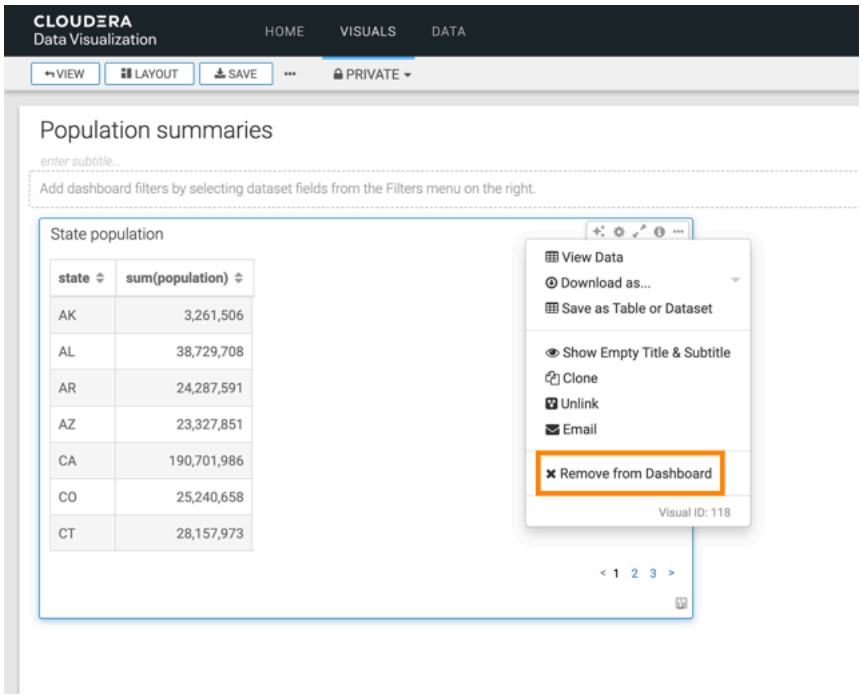

Deleting a visual

Cloudera Data Visualization enables you to delete an unlinked visual.

About this task

These instructions show you how you can delete a visual. When you remove an unlinked visual from a dashboard, you should use the 'delete...' menu option. After deletion, the visual will no longer exist in the system.

When you remove a linked visual from a dashboard, you should use the 'remove...' menu option. After removal, the visual is still available in Cloudera Data Visualization and you can add it to other dashboards. For more information, see *Removing a visual*.

Procedure

1. In Edit mode, click the Supplemental menu at the top right corner the visual you want to delete.
2. Click Delete Visual.

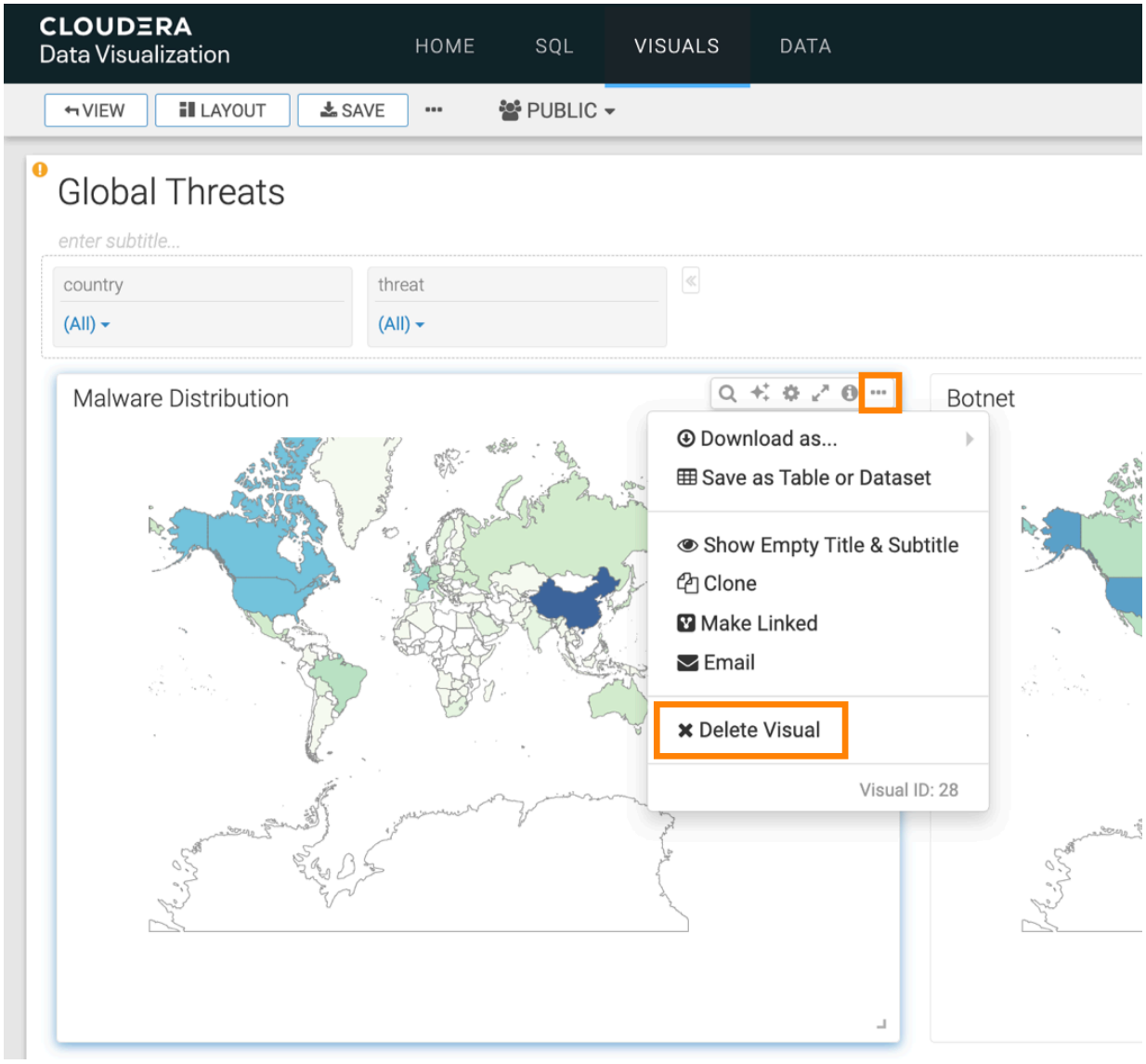

Adding filters to a visual

Cloudera Data Visualization enables you to add filters to a visual if the visual has a Filters shelf available.

Procedure

1. Open a visual of your choice in Edit mode.
   In this example, a visual called Deficiency date correlation overview is used.

2. Populate the Filters shelf from the available fields (Dimensions, Measures) in the DATA menu.

   In this example, the country field is added to the filter shelf. A modal window pops up where you can set the values you want to filter for. You can choose discrete values, ranges, patterns, or use the Select All option for a bulk selection of the shown filter values.

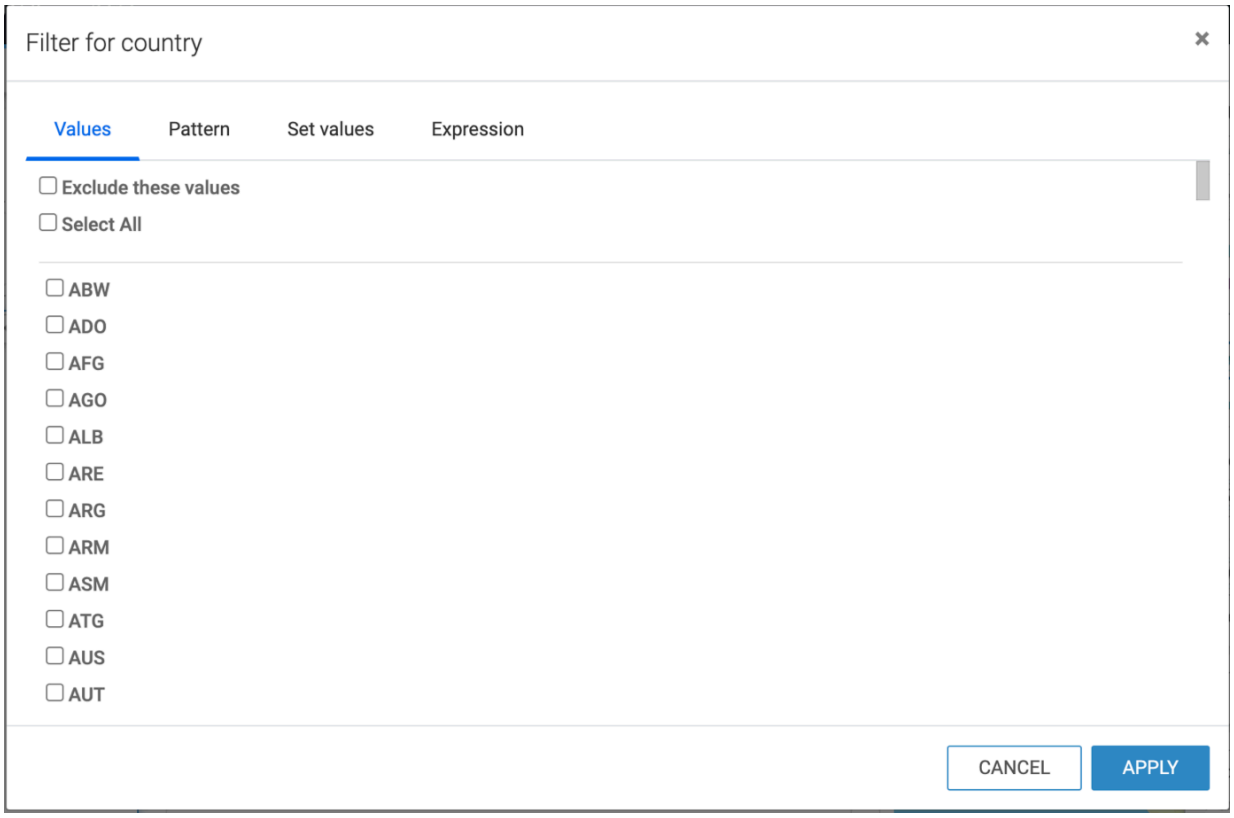

You can also create your own custom filter expressions by using the Expression tab. In this example, the deficiency_code field is added to the filter shelf with a custom filter expression.

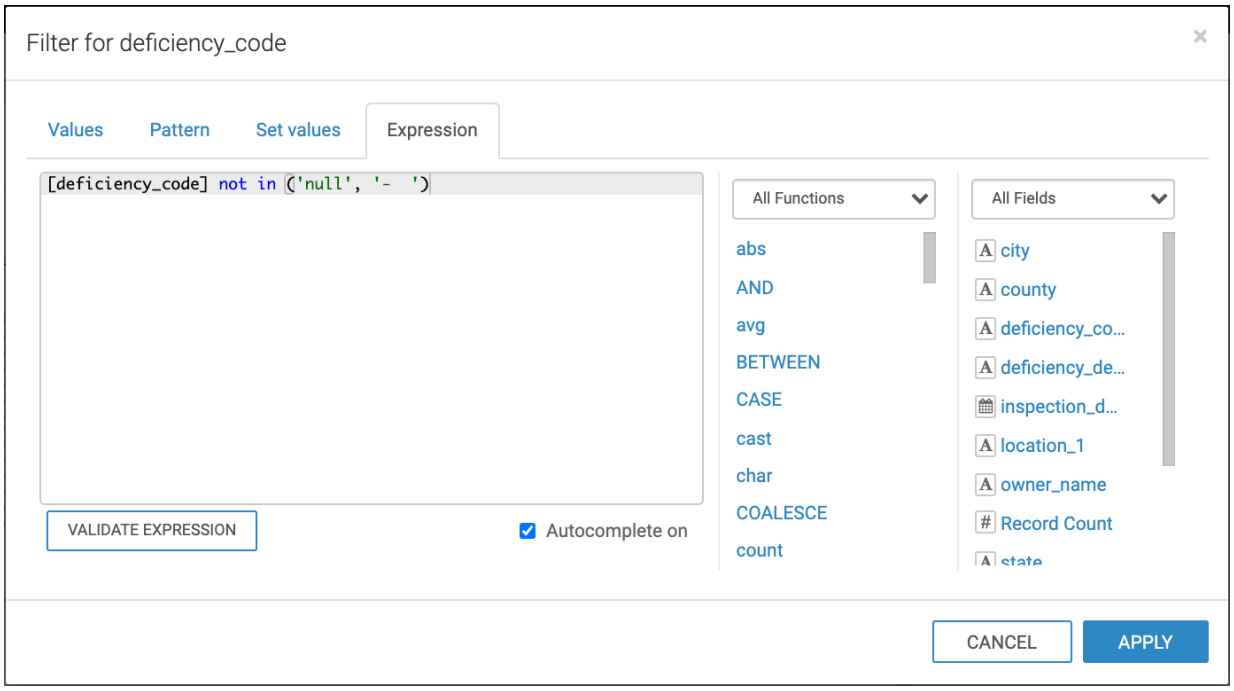

You can use the filter with its default settings, or you can open the Settings menu from the side menu bar to configure your filter.

Exploring visual options

Cloudera Data Visualization helps you to develop valuable insights into your data by suggesting new visual types and options based on the information you already use in your existing visuals.

Before you begin

To view visual type and option recommendations, you must activate Enable Explore in the Site Settings Dashboards Visuals menu. For more information, see *Enabling visual options*.

Procedure

1. Open the dashboard in Edit mode.

2. Select an existing visual and hover over its top right corner.

   Click Explore Options and choose one from the available recommendations. The options presented depend on the visual type.

   - Visual Types
   - Colors
   - Styles/Fonts
   - Settings

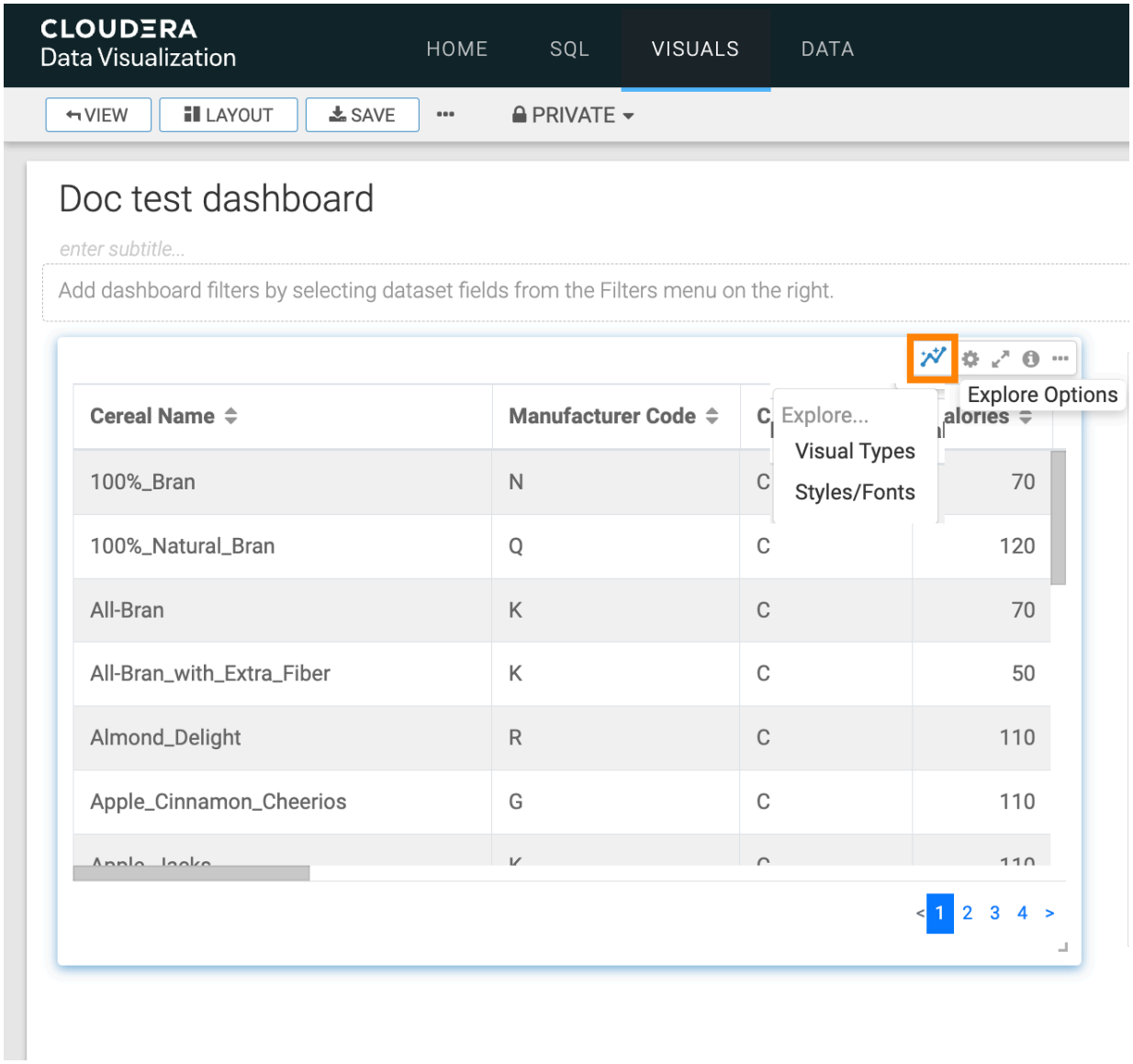

3. You can also select the Explore Visuals icon in the Dashboard Designer.

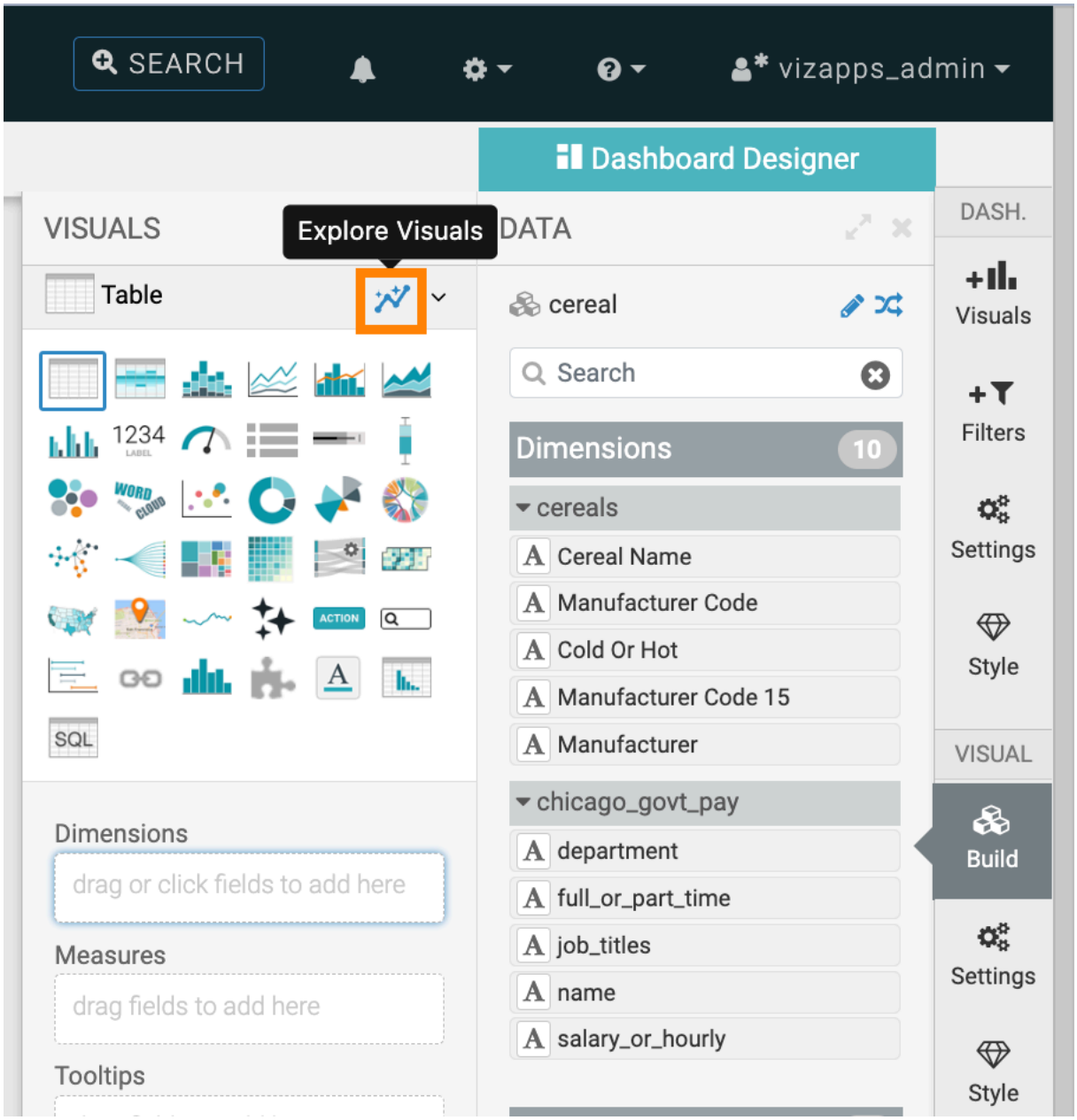

Visual type catalog

Cloudera Data Visualization has a rich offering of visualization types to assist you in analyzing data.

Click each visual to see how to build and use the visual type.

availability:

The following visual types support smooth rendering of streaming data: *Lines*, *Bars*, *Areas*, *Stacked bars*, *Combo*, *Calendar heatmap*, *Correlation heatmap*, *Packed bubbles*, *Pie*, and *Scatter*.

Table
Cross tabulation
Bars
Lines
Combo
Areas
Grouped bars
Box plot
Bullet
Row listing
Packed bubbles
WORD
VISUAL
CLOUD
Word cloud
Scatter
Pie
Radial
Chord

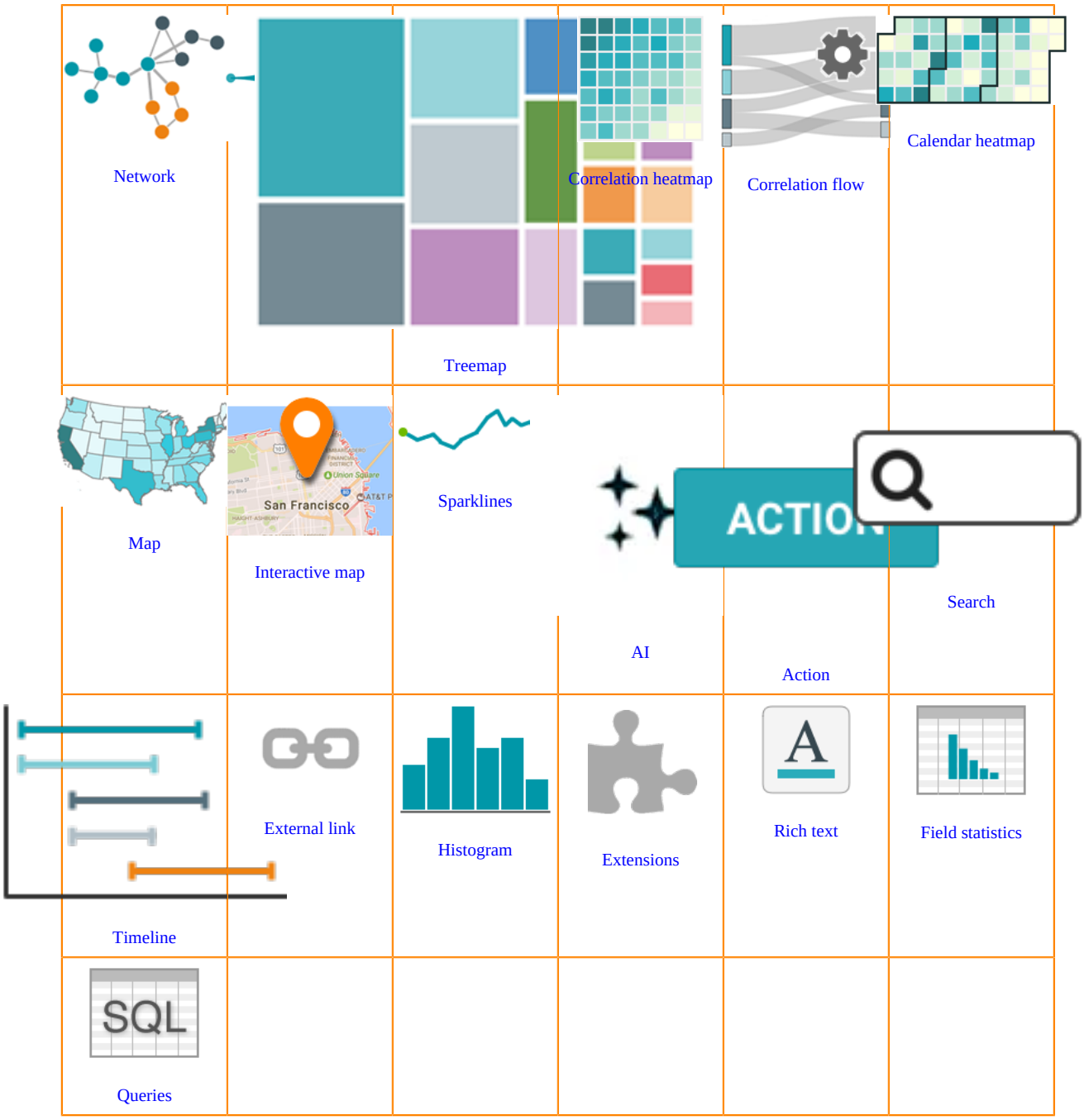

Table

A Table visual displays the fields of the underlying dataset. It may be sorted along any measure or dimension, and its cells can be formatted using format strings.

About this task

The following steps demonstrate how to create a new table visual representation on the World Life Expectancy dataset. For an overview of shelves that specify this visual, see *Shelves for tables*.

Tip:

The default Table visual request selects only columns from the first table in a dataset with a join, up to the maximum number of columns specified in the Table Style menu.

- To change the maximum number of columns, see *Specifying maximum number of columns*.
- To show columns from other tables of the dataset, add them to the shelves of visual from the Dimensions and Measures of the DATA menu.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.

   For instructions, see *Creating a visual*. By default, the table shows all fields in the dataset. If your starting point is another visual type, find and click Table in the VISUALS menu.

   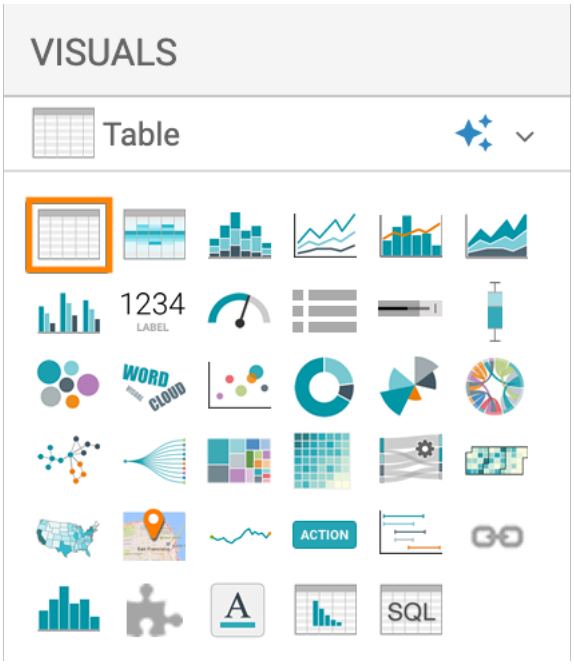

   The shelves of the Table visual are Dimensions, Measures, Tooltips, Filters, and Limit.

   There are no mandatory shelves. If you do not specify a value in any of the shelves, all columns appear in the visual.

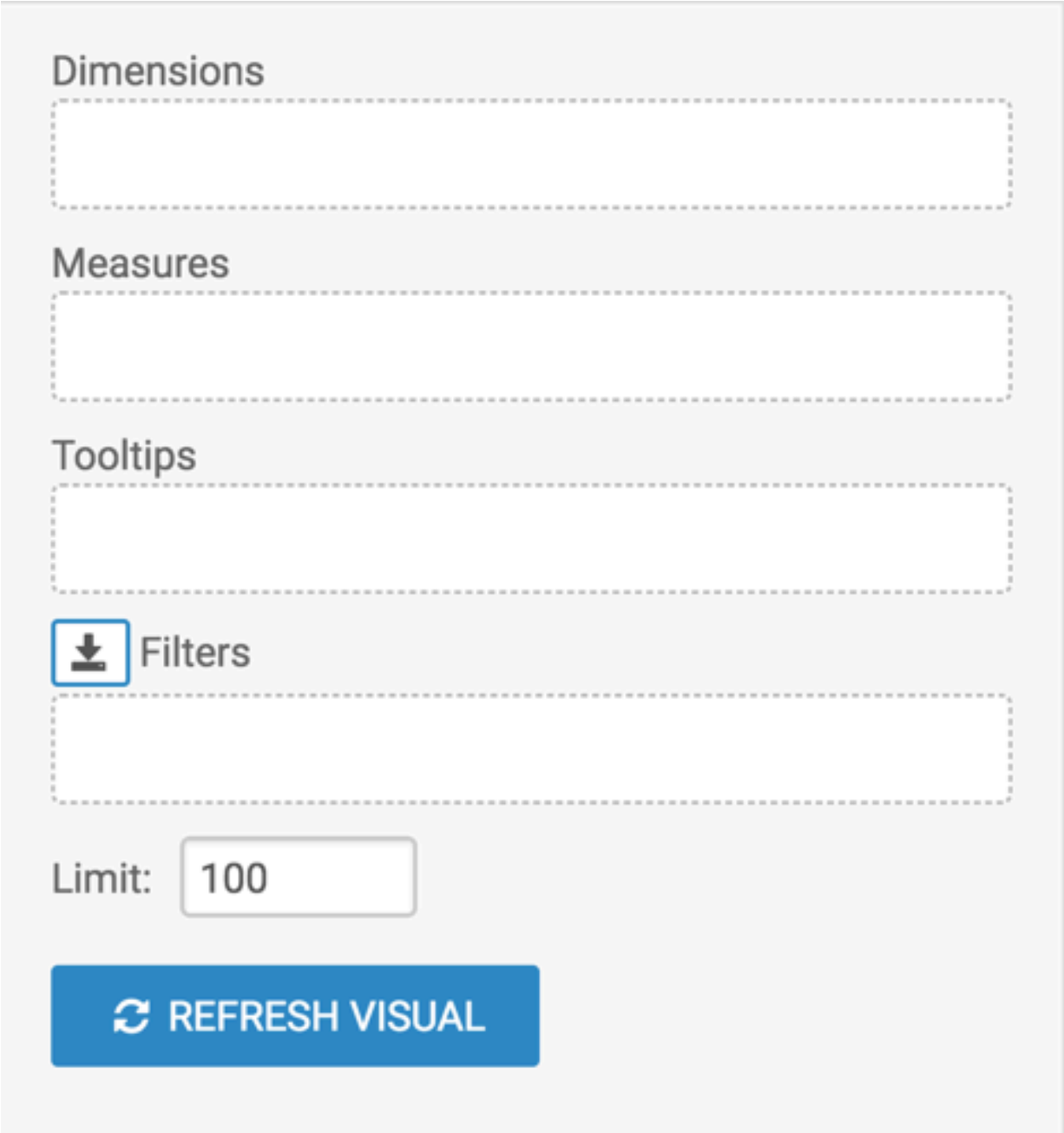
Dimensions
Measures
Tooltips
Filters
Limit:
100
REFRESH VISUAL

2. To show only specific items, populate the shelves from the fields available in the DATA menu.
   - Under Dimensions, select 'country' and drag it to the Dimensions shelf.
   - Under Measures, select 'life_expectancy' and drag it to the Measures shelf.

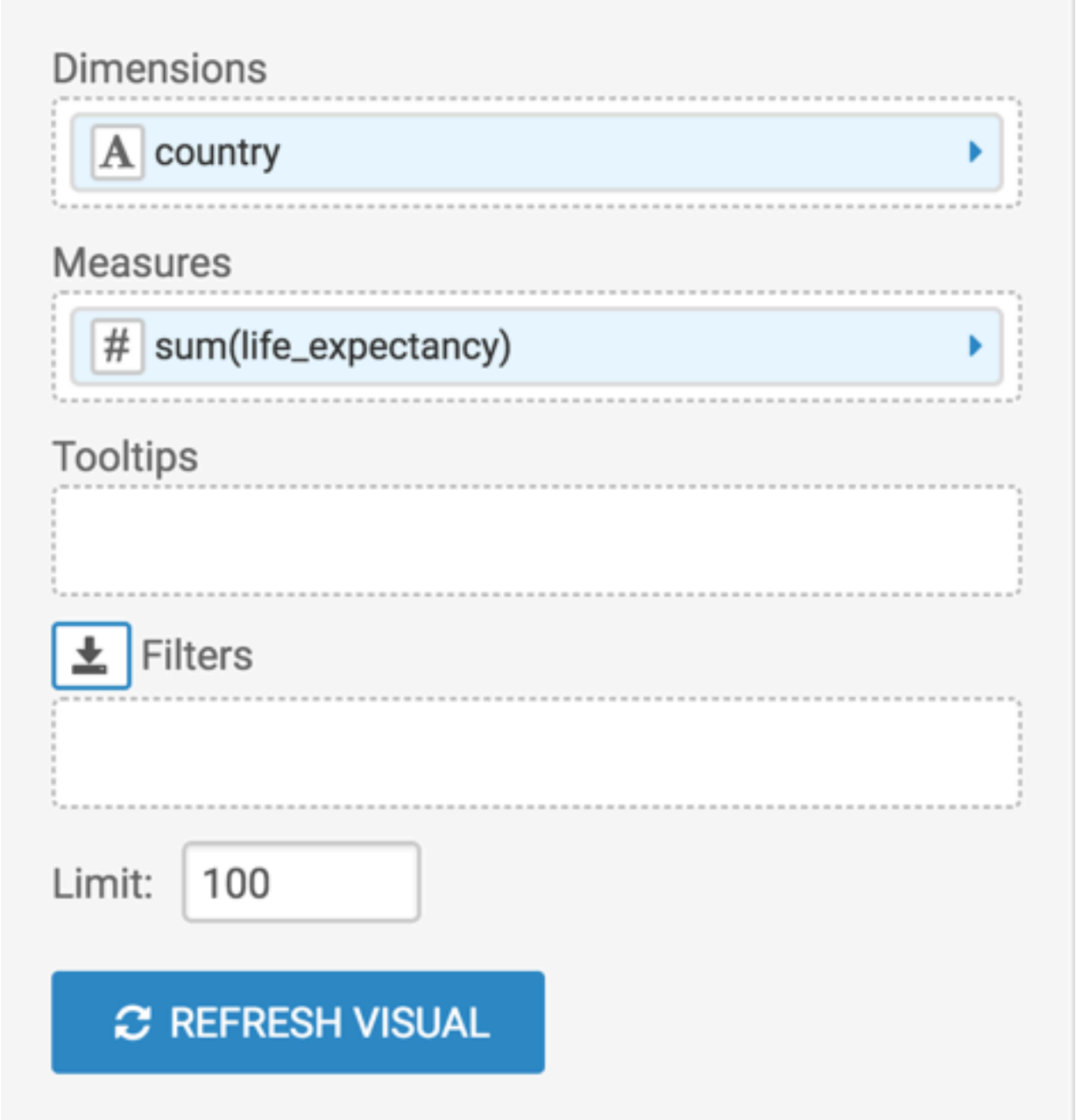

3. Change the aggregate of the 'life_expectancy' field from 'sum(life_expectancy)' to 'avg(life_expectancy)'.
   a. On the Measures shelf of the visual, click the caret-right icon to the right of the 'sum(life_expectancy)' field.
   b. In the FIELD PROPERTIES menu, click the caret-right icon next to Aggregates to open a drop-down menu.
   c. From the list of aggregate functions, select Average.

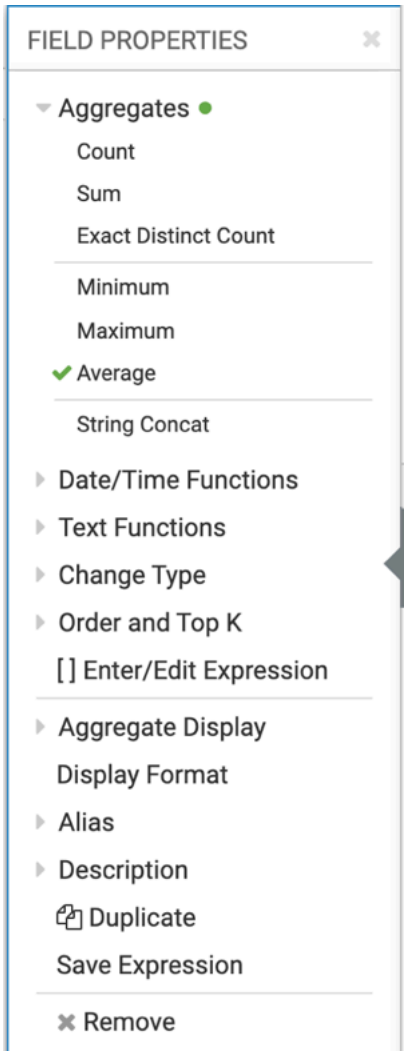

   d. Click the x icon at the top of the FIELD PROPERTIES menu to close it.

   The shelf now contains the modified field with the avg() aggregation function.

   Alternatively, you can change the aggregation at the level of the dataset, as described in *Changing field aggregation*.

4. Click REFRESH VISUAL.

   The updated Table visual appears.

Country	Life Expectancy
Afghanistan	35.4
Albania	54.7
Algeria	45.3
American Samoa	
Andorra	
Angola	33.8
Anguilla	
Antigua and Barbuda	55.4
Argentina	60.9
Armenia	56.2
Aruba	57.0
Australia	68.6

   Tip: If no default display format is specified, comma separation will be applied. See, Customizing display format.

5. To improve the appearance of the visual, alias the 'country' field as 'Country' and the 'life_expectancy' field as 'Life Expectancy'.

   a. On the shelf of the visual, click the caret-right icon to the right of the field.
   b. In the FIELD PROPERTIES menu, click the caret-right icon next to Alias.
   c. In the text box below Alias, enter the alias name of the column, as it should appear in the visual.
   d. Click the x icon at the top of the FIELD PROPERTIES menu to close it.

   The shelves now show the columns with their alias names.

6. On the Filters shelf, you can add several Dimensions and Measures from the DATA menu.

   This enables you to dynamically control the data input and discover the data at a more granular level.

   For example, from Dimensions, drag the 'un_region' and 'year' fields to the Filters shelf.

   Note: The filter selection wizard appears each time you add a field to the Filters shelf. You can specify filter values at this time, or click CANCEL and select them later.

   In this example, 'un_region' is set to 'Americas'. It should appear as un_region in ('Americas') on the Filters shelf. 'year' is set to '2001 through 2010'. It should appear as year in (2001,2002,…,2010) on the Filters shelf.
7. Change the value on the Limit shelf to another number.

   In this example, the new value is '12'.
8. Click REFRESH VISUAL.

   The updated Table visual appears.

Country	Life Expectancy
Anguilla	
Antigua and Barbuda	74.6
Argentina	74.9
Aruba	74.4
Bahamas	73.8
Barbados	74.1
Belize	72.1
Bermuda	
Brazil	71.9
Canada	80.4
Cayman Islands	
Chile	78.1

9. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'World Population - Table'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
10. At the top left corner of the Visual Builder, click SAVE.

Results

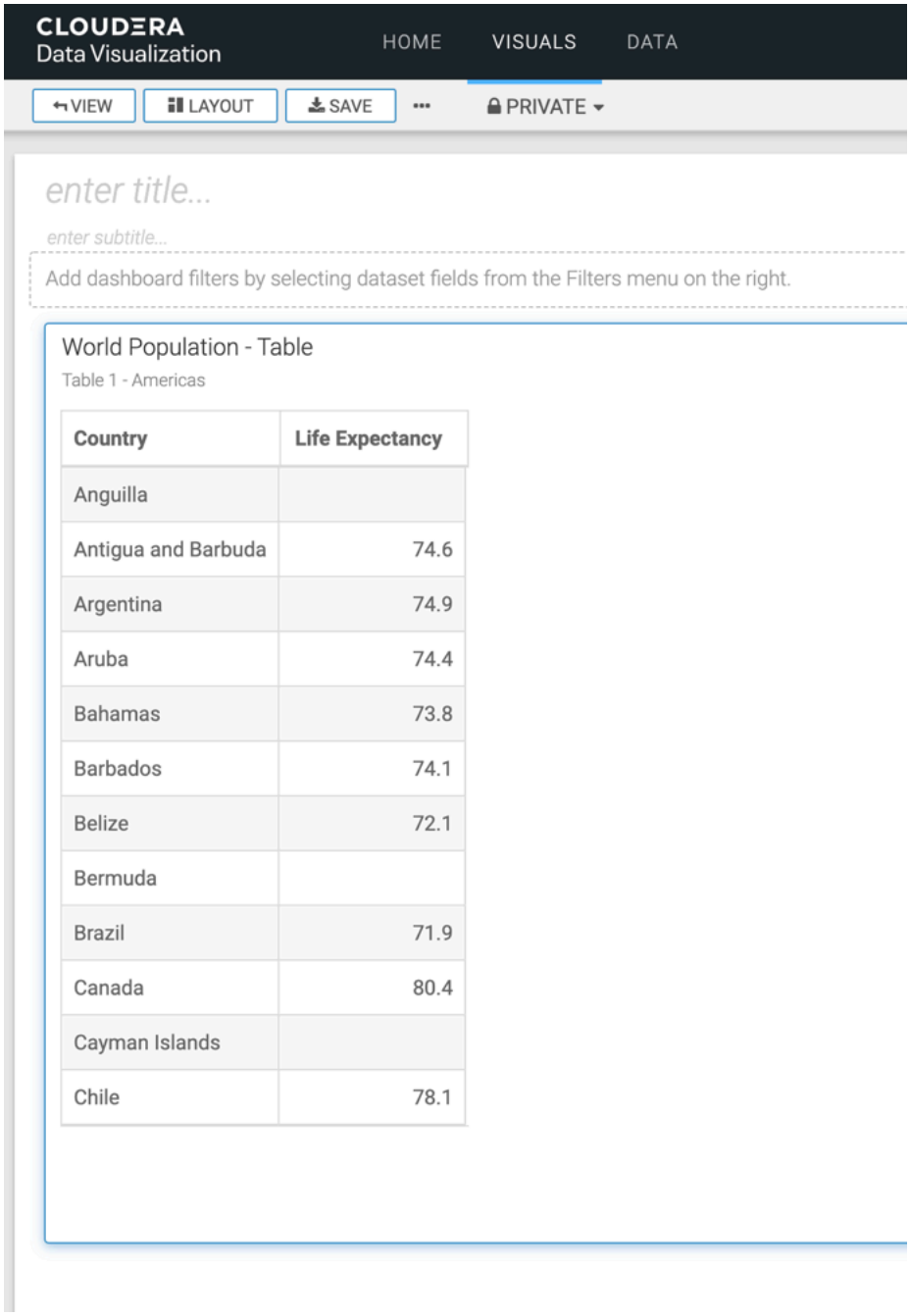

Country	Life Expectancy
Anguilla	
Antigua and Barbuda	74.6
Argentina	74.9
Aruba	74.4
Bahamas	73.8
Barbados	74.1
Belize	72.1
Bermuda	
Brazil	71.9
Canada	80.4
Cayman Islands	
Chile	78.1

Related Information

Specifying maximum number of columns
Creating a visual
Changing field aggregation
Customizing display format

Shelves for tables

Overview of shelves for Cloudera Data Visualization Table visuals.

A Table visual has the following shelves:

Dimensions

Add a Dimension field from the DATA menu to include a dimension column in the table.

This shelf accepts multiple fields.

Measures

Add a Measure field from the DATA menu to include an aggregate column in the table.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Limit

This shelf specifies the number of records displayed in the table.

The default value is 100.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

This field is necessary to see measures that do not appear in the table.

Saving tables

Cloudera Data Visualization enables you to save tabular data as its own table, and writes it back to the database. It also enables you to produce a dataset based on data exposed in a visual, which may be used for self-referenced application stacking.

About this task

This feature is available only when:

- Examining data in either table or query visuals
- Using Impala connections
- User has Data Import/Load privileges

Saving a table as a materialized table

Cloudera Data Visualization allows you to save a table as pre-computed (materialized) data. Use a materialized table if you want to permanently store your query results.

Before you begin

You have a table visual that you want to save as a materialized table.

Procedure

1. On the top right menu, click the (ellipsis) icon and select Save Table from the menu.

2. In the Create a Table or Dataset modal window, fill out these fields:
   a) Choose Materialized table for Type to permanently save query results.
   b) Under Create in database, choose an existing database where you want to save the table.
   c) Enter the name of the new table.
   d) Choose Parquet or Text file as Storage format.
   e) When creating a dataset as part of this operation, select the Create a new dataset based on new table/view option and enter the Dataset name.
   f) Click CREATE.

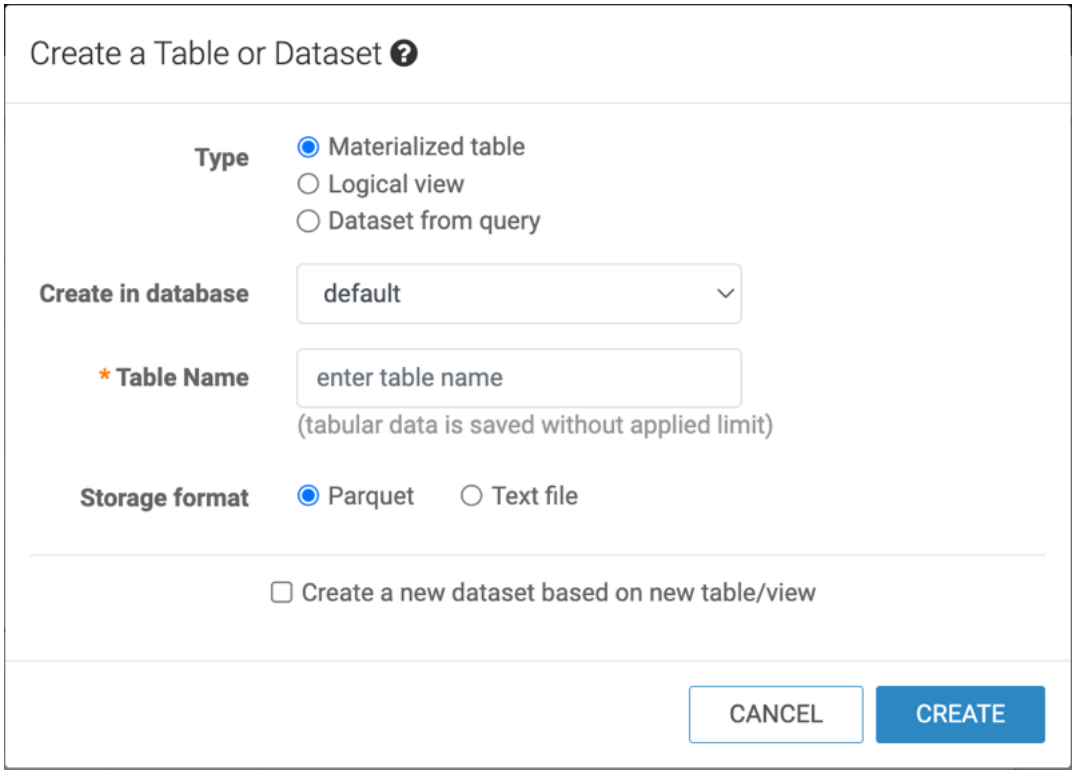

A success confirmation modal appears, with a link to the new default visual on the new dataset.

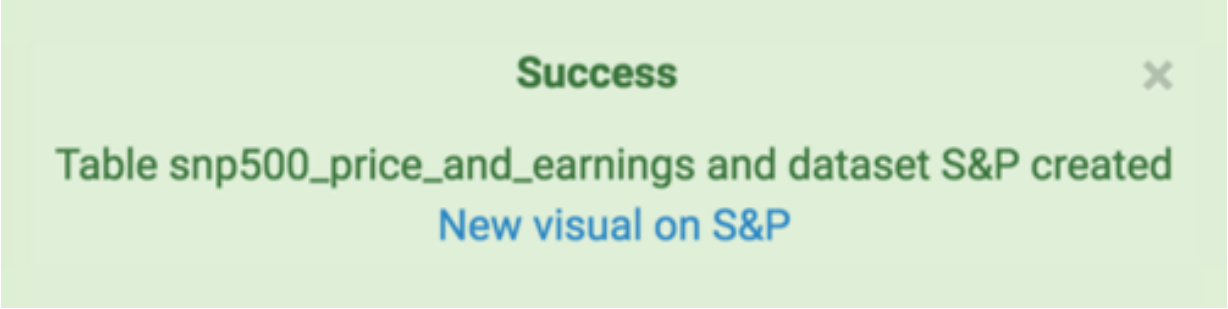

3. Click the link to open a new default visual on the new dataset in Edit mode.

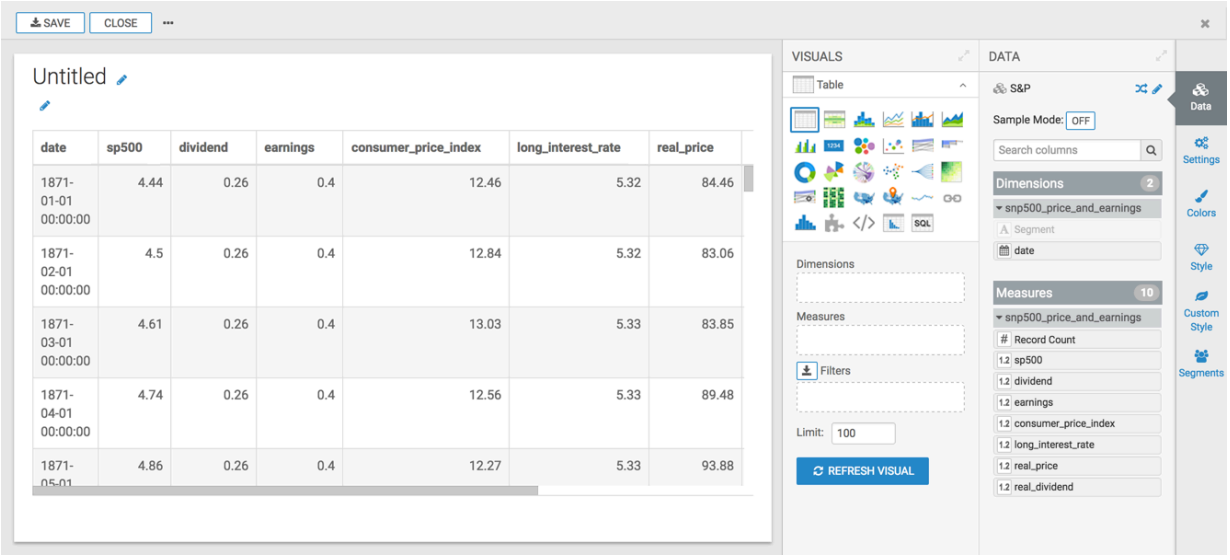

Saving a table as a logical view

Cloudera Data Visualization allows you to create logical tables. Use a logical view if you want updated query results every time you run the visualization.

Before you begin

You have a table visual that you want to save as a logical view.

Procedure

1. On the top right menu, click the (ellipsis) icon and select Save Table from the menu.

2. In the Create a Table or Dataset modal window, fill out these fields:
   a) Choose Logical view for Type to generate updated query results every time the visualization is loaded or refreshed.
   b) Under Create in database, choose an existing database where you want to save the table.
   c) Enter the name of the new logical view.
   d) When creating a dataset as part of this operation, select the Create a new dataset based on new table/view option and enter the Dataset name.

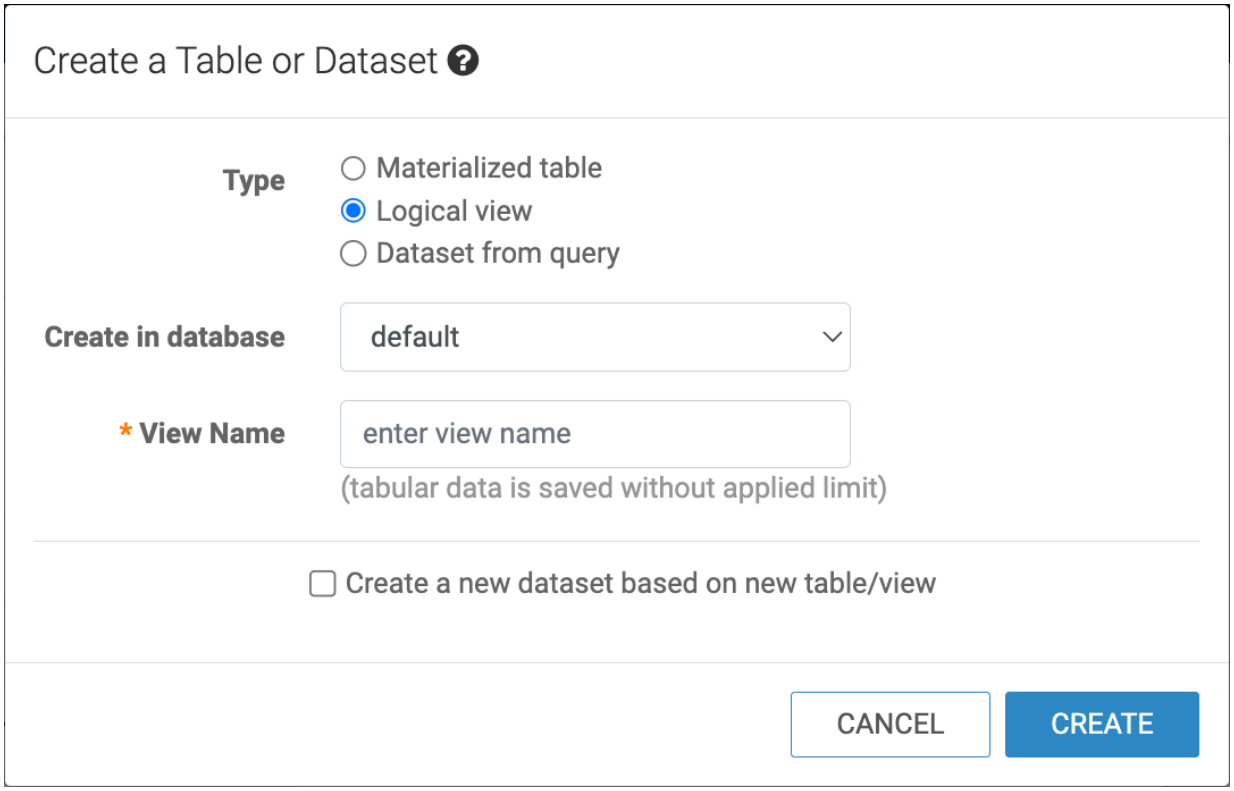

A success confirmation modal appears, with a link to the new default visual on the new dataset.

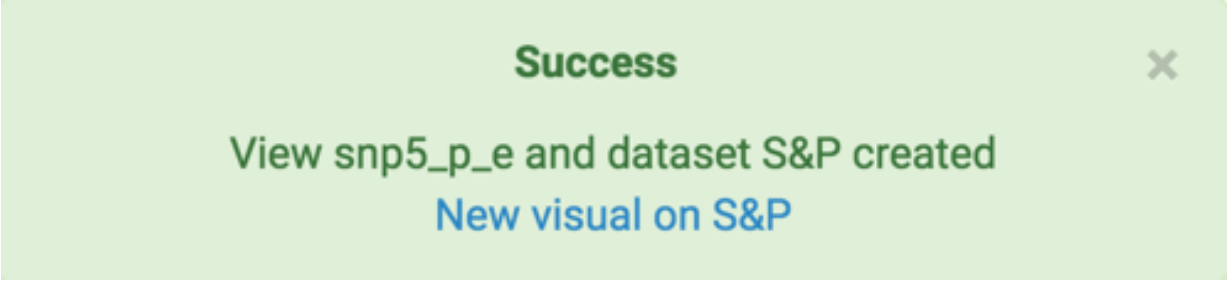

3. Click the link to open a new default visual on the new dataset, in Edit mode.

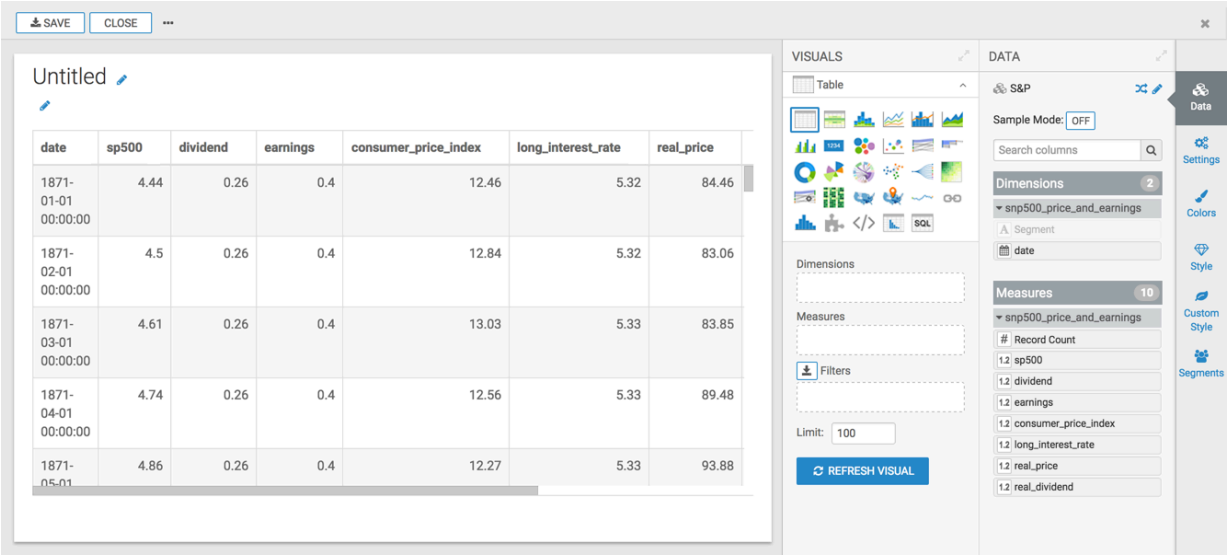

Saving a table as a dataset based on the query

Cloudera Data Visualization allows you to save a query result as a separate dataset. Unlike a logical view, which is created on the server, this method uses the current query as the basis for the dataset. This feature is particularly useful when working with subsets of data, specific sections, or when constructing datasets with self-joins.

Before you begin

You have a table visual that you want to save as a dataset.

Procedure

1. Add the necessary fields on the table shelves, and apply functions, for example sort, order, alias, or filters.
2. On the top right menu, click the (ellipsis) icon and select Save Table from the menu.

3. In the Create a Table or Dataset modal window, fill out these fields:
   a) Choose Dataset from query for Type.
   b) Enter the name of the new dataset.
   c) In the SQL for dataset creation field, you can edit the SQL expression used for creating the dataset.
   d) Click CREATE.

A success confirmation modal appears, with a link to the new default visual on the new dataset.

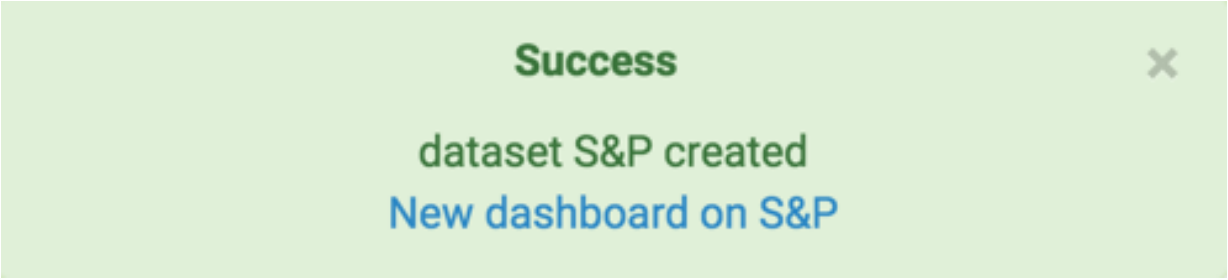

4. Click the link to open a new default visual on the new dataset, in Edit mode.

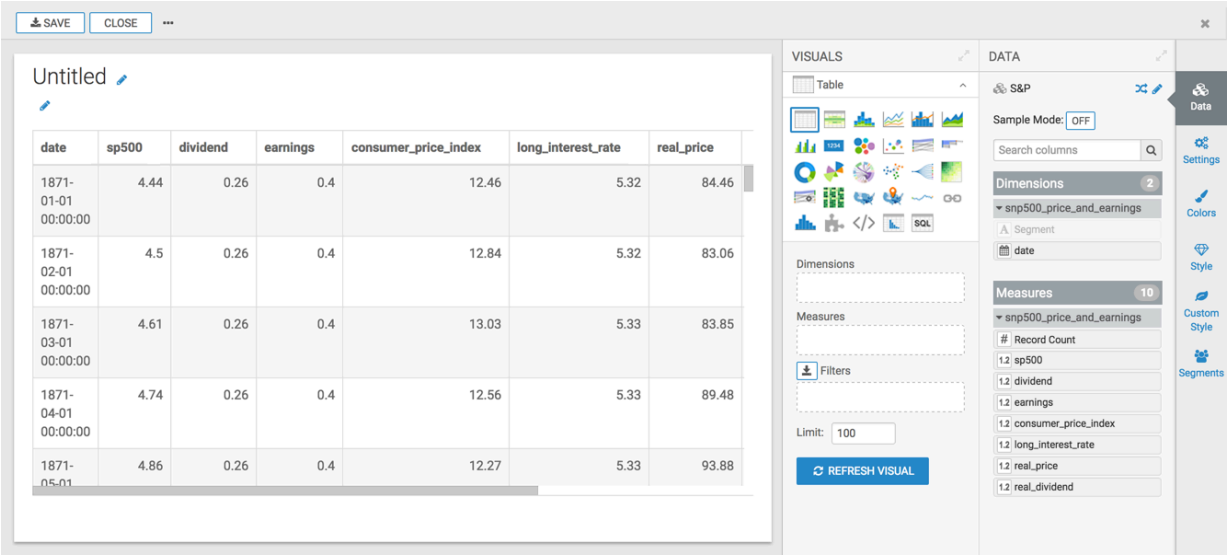

Cross tabulation

Cross Tabulation visual is a tool that allows you to compare the relationship between two variables in table format. Cloudera Data Visualization enhances the understanding of relative value size by assigning more saturated colors to the table fields with larger values.

About this task

The following steps demonstrate how to create a new Cross Tabulation visual on the World Life Expectancy dataset. For an overview of shelves that specify this visual, see *Shelves for cross tabulation*.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Cross Tabulation.

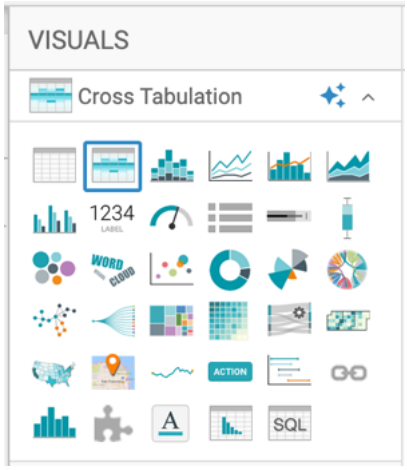

The shelves of the visual changed. They are now Column, Row, Measures, Tooltips, and Filters. The Measures shelf is mandatory.

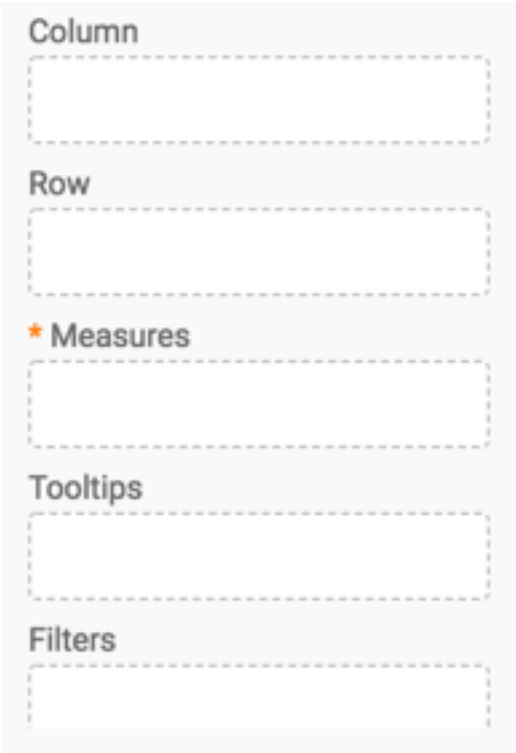

3. To show specific items, populate the shelves from the available fields in the DATA menu.
   - Under Dimensions, select 'country' and drag it to the Column shelf.
   - Under Dimensions, select 'year' and drag it to the Row shelf.
   - Under Measures, select 'life_expectancy' and drag it to the Measures shelf.

     By default, the sum aggregate of the field appears on the shelf, as 'sum(life_expectancy)'.

4. Change the aggregate of the 'life_expectancy' field from 'sum(life_expectancy)' to 'avg(life_expectancy)'.
   a. On the Measures shelf of the visual, click the caret-right icon to the right of the field.
   b. In the FIELD PROPERTIES menu, click the caret-right icon next to Aggregates to open a drop-down menu..
   c. From the list of aggregate functions, select Average.

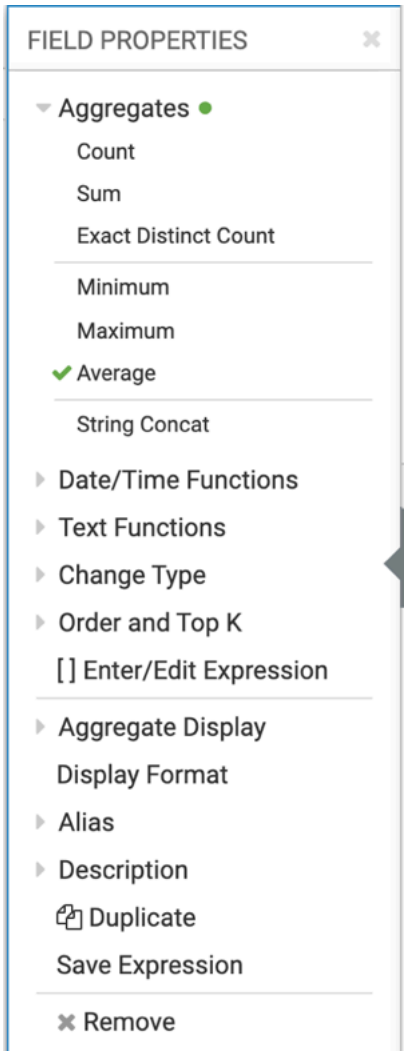

   d. Click the x icon at the top of the FIELD PROPERTIES menu to close it.

   The shelf now contains the modified field with the 'avg()' aggregation function.

   Alternatively, you can change the aggregation at the level of the dataset, as described in *Changing field aggregation*.

5. Click REFRESH VISUAL.

   The updated Cross Tabulation visual appears. It shows life expectancy information for each row of the dataset. By default, Cross Tabulation visuals show smaller values in light colored cells, and larger values in darker colored cells.

	country														
	Afghanistan	Albania	Algeria	American Samoa	Andorra	Angola	Anguilla	Antigua and Barbuda	Argentina	Armenia	Aruba	Australia	Austria	Azerbaijan	Bahamas
year	avg(life_expectancy)														
1900	27.2	35.4	28.9			27		33.5	36.8	35.9	34.4	49.9	41.5	31.1	35.2
1901	27.2	35.4	28.9			27		33.5	37.4	36.1	34.4	50.5	42	31.3	35.2
1902	27.2	35.4	28.9			27		33.5	38	36.4	34.4	51	41.1	31.6	35.2
1903	27.2	35.4	29.9			27		33.5	38.5	36.6	34.4	51.5	40.1	31.8	35.2
1904	27.1	35.4	24.1			27		33.5	39.1	36.9	34.4	52	40.7	32.1	35.2
1905	27.1	35.4	26.7			27		33.5	39.7	37.1	34.4	52.6	41.4	32.3	35.2
1906	27.1	35.4	28			27		33.5	40.5	37.4	34.4	53.1	42	32.6	35.2
1907	27.1	35.4	27.9			27		33.5	41.2	37.6	34.4	53.6	42.6	32.8	35.2
1908	27.1	35.4	27.8			27		33.5	42	37.9	34.4	54.2	43.2	33.1	35.2
1909	27.1	35.4	29.3			27		33.5	42.7	38.1	34.4	54.7	43.9	33.3	35.2
1910	27.1	35.4	30.8			27		33.5	43.5	38.4	34.4	55.2	44.5	33.6	35.2
1911	27.1	35.4	30.5			27		33.5	44	41.4	34.4	55.7	45.1	36.5	35.2
1912	27.1	35.4	31.8			27		33.5	44.6	41.8	34.4	56.3	45.7	36.9	35.2
1913	27.1	35.4	29.7			27		33.5	45.1	39.6	34.4	56.8	46.2	34.8	35.2
1914	27	35.4	29.2			27		33.5	45.7	39.2	34.4	57.3	46.8	34.4	35.2
1915	27	35.4	28.7			27		33.5	46.2	38.8	34.4	57.9	47.4	34	35.2
1916	27	35.4	28.3			27		33.5	47	38.8	34.4	58.4	47.9	34	35.2
1917	27	35.4	28.3			27		33.5	47.8	35.8	34.4	58.9	48.5	31	35.2
1918	7	19.4	22.1			10.4		21.8	41.8	27	22.3	54.8	32.4	22	22.8
1919	27	35.4	28.3			27		33.5	49.4	37	34.4	60	49.6	32	35.2
1920	27	35.4	27.5			27		33.5	50.2	28	34.4	60.5	50.2	30	35.2
1921	27	35.4	27.5			27		33.5	50.7	38	34.4	61	50.8	33	35.2

6. On the Filters shelf, you can add several Dimensions and Measures from the DATA menu.

   This enables you to dynamically control the data input, and discover the data at a more granular level.

   - For example, from Dimensions, drag 'un_subregion' to the Filters shelf.

     Note: The filter selection wizard appears each time you add a field to the Filters shelf. You can specify filter values at this time, or click CANCEL and select them later.

     In this example, the filter value is set to 'Middle Africa'.
   - To limit the time interval, add the 'year' field to the Filters shelf.

     In the Filter for year modal window, click the Range tab, select the last 30 years of the dataset, 1981 through 2010, and lick APPLY.

7. To improve the appearance of the visual, alias the 'country' field as 'Country' and the 'life_expectancy' field as 'Life Expectancy'.

   a. On the shelf of the visual, click the caret-right icon to the right of the field.
   b. In the FIELD PROPERTIES menu, click the caret-right icon next to Alias.
   c. In the text box below Alias, enter the alias name of the column, as it should appear in the visual.
   d. Click the x icon at the top of the FIELD PROPERTIES menu to close it.

   The shelves now show the columns with their alias names.

8. Adjust the display format for the 'Life Expectancy' field to use a comma thousands separator, and to carry zero decimal places.
   a. On the Measures shelf of the visual, click the caret-right icon to the right of the field.
   b. In the FIELD PROPERTIES menu, click Display Format.
   c. In the Display Format modal window, select the Category 'Real Number'.
   d. In the Decimal Places text box, set '0'.
   e. Select the Use 1000 separator option, and click SAVE.
   f. Click x icon at the top of the FIELD PROPERTIES menu to close it.
9. To provide more detailed information, add some fields to the Tooltips shelf.

   For example, drag the Dimensions 'un_region' and 'un_subregion', and the Measures 'population' and 'gdp_per_capita' to the Tooltips shelf.

10. Click REFRESH VISUAL.

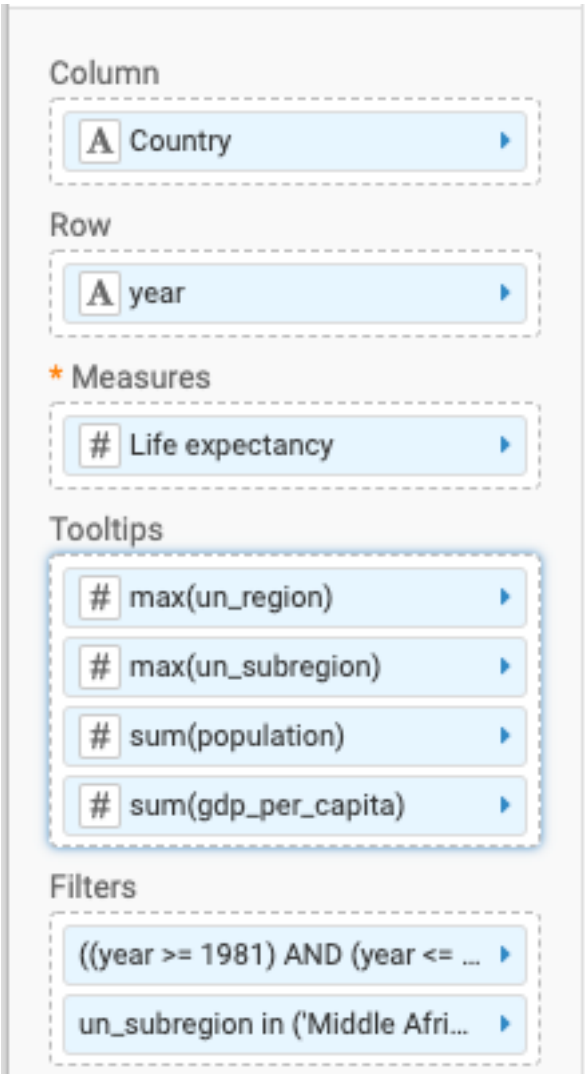

The updated Cross Tabulation visual appears. It demonstrates the relative size of populations by assigning a progressively more saturated color to represent increasingly larger values.

World Life Expectancy - Cross Tabulation

enter subtitle...

Antigua and Barbuda	Argentina	Armenia	Aruba	Australia	Austria	Azerbaijan	Bahamas	Bahrain	Bangladesh	Barbados	Belarus	Belgium	Belize	Benin	Bermuda	Bhutan	Bosnia an
34	37	36	34	50	42	31	35	30	22	32	36	47	27	31	0	29	
34	37	36	34	51	42	31	35	30	22	32	37	49	27	31	0	29	
34	38	36	34	51	41	32	35	30	22	32	37	49	27	31	0	29	
34	39	37	34	52	40	32	35	30	22	32	38	49	27	31	0	29	
34	39	37	34	52	41	32	35	30	22	32	39	49	27	31	0	29	
34	40	37	34	53	41	32	35	30	22	32	39	50	27	31	0	29	
34	41	37	34	53	42	33	35	30	22	32	40	50	27	31	0	29	
34	41	38	34	54	43	33	35	30	22	32	41	51	27	31	0	29	
34	42	38	34	54	43	33	35	30	22	32	41	50	27	31	0	29	
34	43	38	34	55	44	33	35	30	22	32	42	51	27	31	0	29	
34	44	38	34	55	45	34	35	30	22	32	42	51	27	31	0	29	
34	44	41	34	56	45	37	35	30	22	32	43	49	27	31	0	29	
34	45	42	34	56	46	37	35	30	23	32	44	52	28	31	0	29	
34	45	40	34	57	46	35	35	30	23	32	44	53	28	31	0	29	
34	46	39	34	57	47	34	35	30	23	32	43	42	29	31	0	29	
34	46	39	34	58	47	34	35	30	23	32	37	40	29	31	0	29	
34	47	39	34	58	48	34	35	30	23	32	37	42	30	31	0	29	
34	48	36	34	59	49	31	35	30	23	32	35	43	30	31	0	29	
22	42	27	22	55	32	22	23	25	13	21	31	35	27	12	0	19	
34	49	37	34	60	50	32	35	30	24	32	33	50	32	31	0	29	
34	50	28	34	61	50	30	35	30	24	32	30	54	32	31	0	29	
34	51	38	34	61	51	33	35	30	24	32	34	55	33	31	0	29	

11. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

In this example, the title is changed to 'World Life Expectancy - Cross Tabulation'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

12. At the top left corner of the Visual Builder, click SAVE.

Related Information

Shelves for cross tabulation

Creating a visual

Changing field aggregation

Displaying totals with data expansion

When performing granular data analysis in Cross Tabulation visuals, Cloudera Data Visualization enables you to display both subtotals and combined total. These summations operate on the dimensions in the Row shelf. When you enable expansion, you can expand rows to view details of some of the dimensions.

Check out the instructions below on how to enable expansion and display subtotals and totals in Cross Tabulation visuals.

For more information about enabling expansion totals, see *Showing expansion totals above expansion rows*.

Related Information

Showing expansion totals above expansion rows

Configuring a cross tabulation visual for expansion

Follow these steps to enable expansion for a Cross Tabulation visual.

About this task

This example shows how to enable expansion on the year and state fields.

Procedure

1. Start a new Cross Tabulation visual based on the US State Populations Over Time dataset.

   For instructions, see *Creating a visual*.
2. Populate the shelves from the available Fields (Dimensions, Measures) in the DATA menu.
   a) Add year and state to the Row shelf.
   b) Add population to the Measures shelf. Ensure that the default aggregation is sum(population).
   c) Add year to the Filters shelf, set it to 1990 and 2000, and click APPLY.
   d) Add state to the Filters shelf, select values AK, AL, AR, and AZ, then click APPLY.
3. Click REFRESH VISUAL.

   When the Cross Tabulation visual appears, it does not have subtotals and the combined total in the rows and columns.

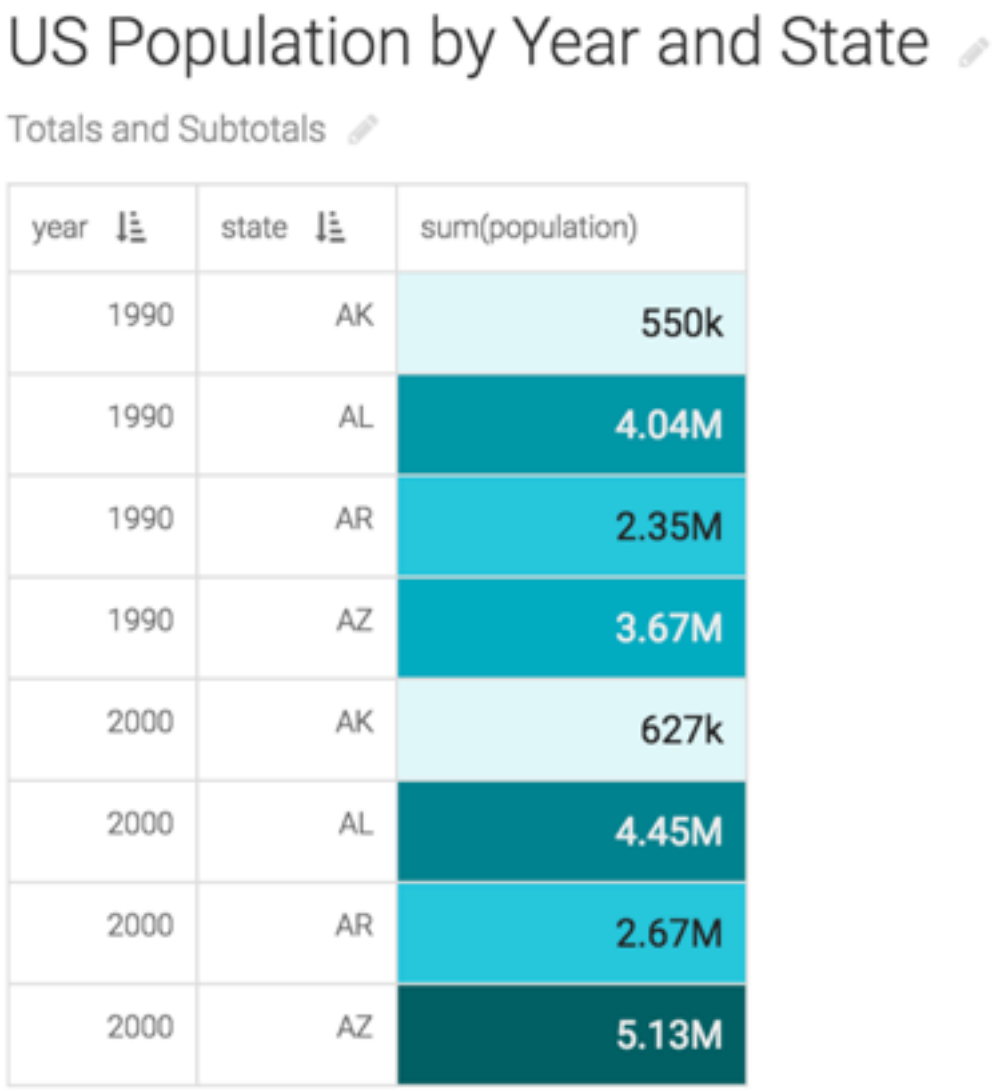

year	state	sum(population)
1990	AK	550k
1990	AL	4.04M
1990	AR	2.35M
1990	AZ	3.67M
2000	AK	627k
2000	AL	4.45M
2000	AR	2.67M
2000	AZ	5.13M

4. On the Row shelf, click year.

5. In the FIELD PROPERTIES menu, click Enable Expansion.

   A green tick appears to the left of the option and a green dot appears to the right of the option.

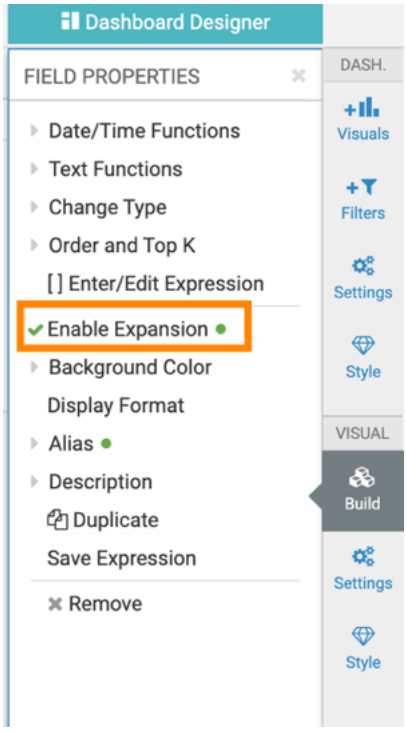

6. Click state on the Row shelf and enable expansion on that field.
7. Click REFRESH VISUAL to see the changes.

   The visual appears with a + icon in the year and state columns. This indicates that you can expand the data.

   The sum(population) column shows the combined population for both years.

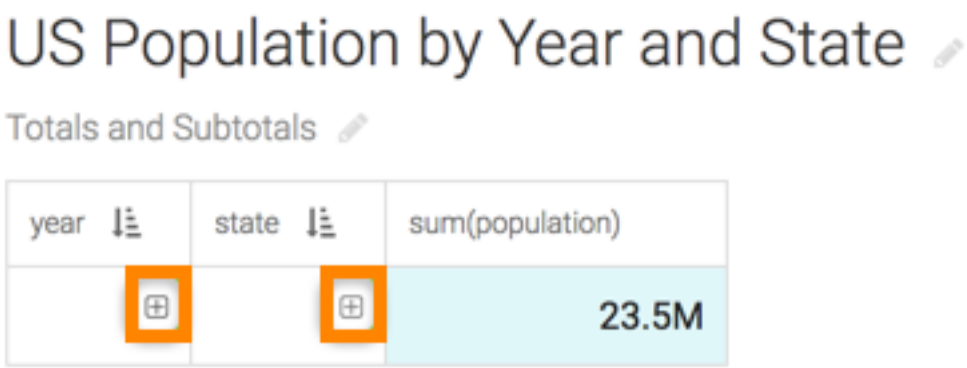

Displaying subtotals and totals

You can expand and shrink the data rows, and display only subtotals and combined total in a Cross Tabulation visual.

Procedure

1. Click the + icon in the year column.

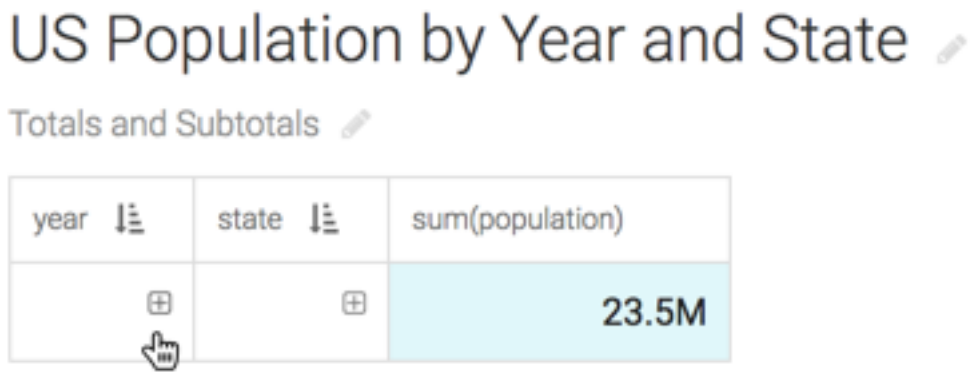

The visual expands the data to show subtotals of each year, and combined total for both years.

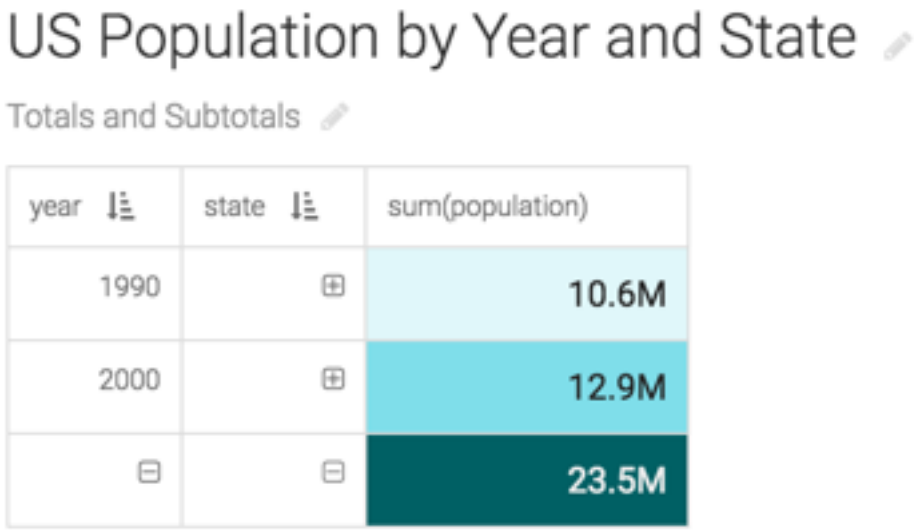

2. Click the two + (plus) icons in the state column.

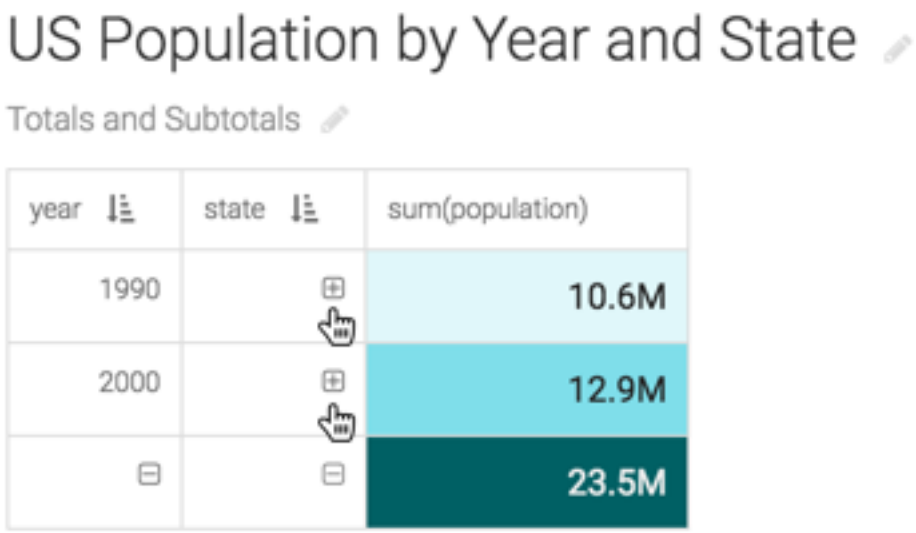

The visual expands the data to show population in each state, and the subtotals and combined population for both years. The subtotals and combined total appear at the bottom of expanded rows. To move the subtotals and totals over the rows, see *Showing expansion totals above expansion rows*.

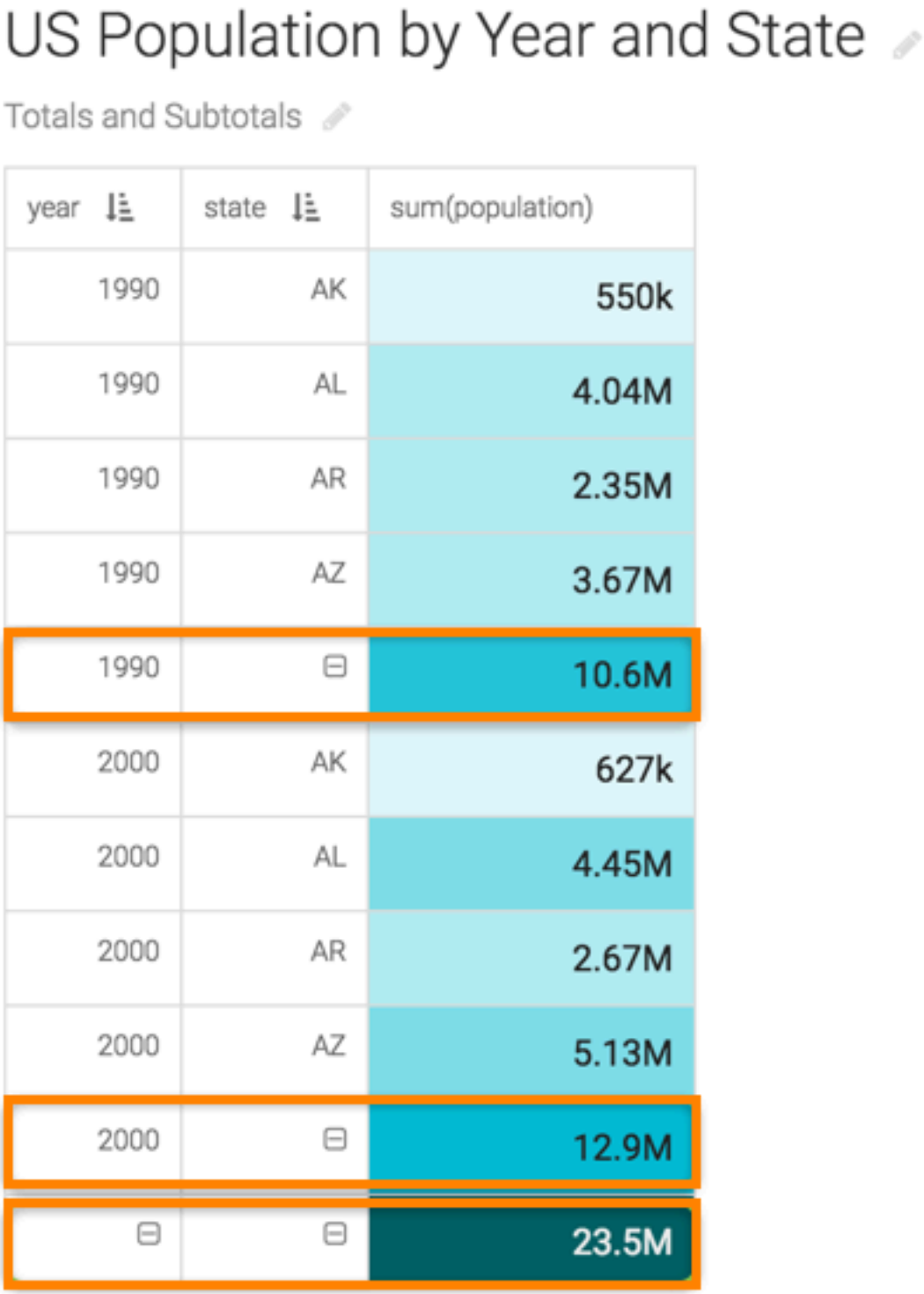

What to do next

To disable subtotals and combined totals, make changes in the FIELD PROPERTIES menu by de-selecting the Enable Expansion option.

Hiding data and subtotal rows

To hide the expanded rows and subtotals, follow these steps.

Procedure

1. Click the - (minus) icon in the year and state columns.

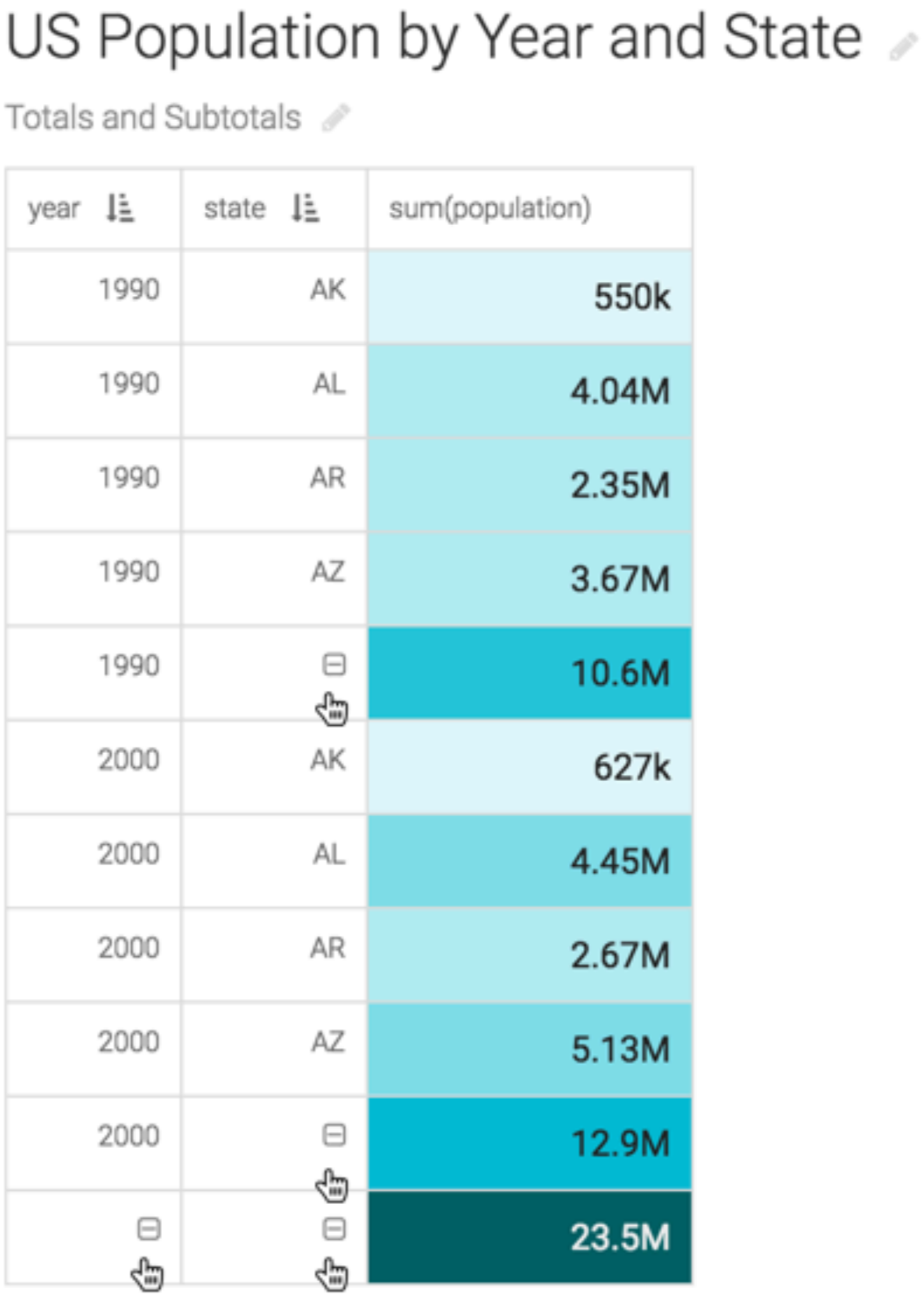

2. The expanded rows and subtotals no longer appear in the visual. Note that a (plus) icon appears again in the year and state columns.

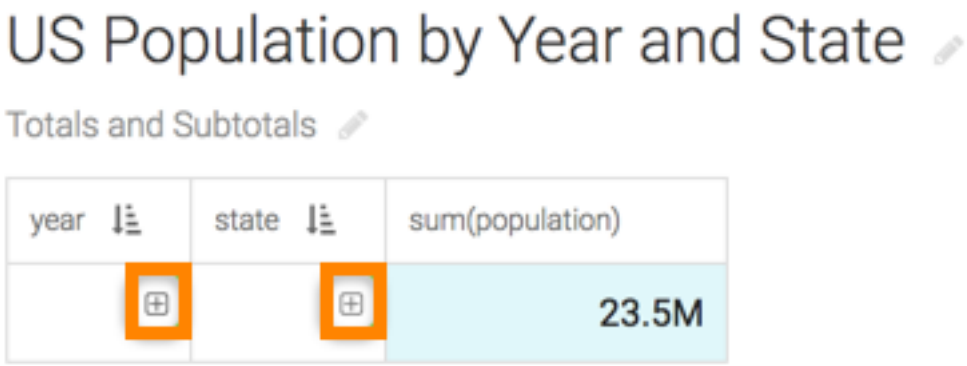

Shelves for cross tabulation

Overview of shelves for Cloudera Data Visualization Cross Tabulation visuals.

A Cross Tabulation visual has the following shelves:

Column

Add a Dimension field from the DATA menu to break aggregations into columns.

This shelf accepts multiple fields.

Row

Add a Dimension field from the DATA menu to break aggregations into rows.

This shelf accepts multiple fields.

Measures

Add a Measure field from the DATA menu to specify the values aggregated across columns and rows.

This shelf accepts multiple fields.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Bars

In Cloudera Data Visualization, bar charts use either vertical or horizontal bars to show comparisons among categories. One axis shows specific categories (typically, dimensions), while the other axis represents discrete values (measures).

About this task

Bar charts are good for avoiding clutter when data labels are long, or when comparing a large number of items, typically more than ten. Bar charts can also display negative numbers.

Note: This visual type supports smooth rendering of streaming data.

The following steps demonstrate how to create a new Bars visual on the World Life Expectancy dataset.

For more information, see *Adding data* and *Creating datasets*. For an overview of shelves that specify this visual, see *Shelves for bars visuals*.

Procedure

1. Start a new visual based on the World Life Expectancy dataset. For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Bars.

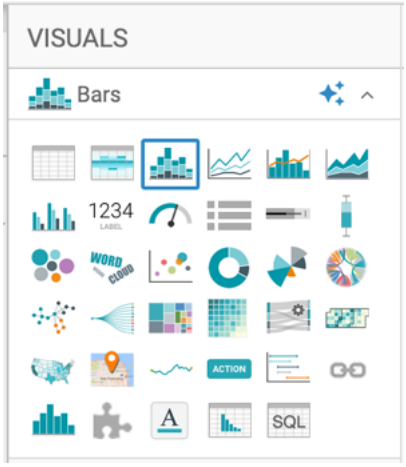

The shelves of the visual changed. They are now X Axis, Y Axis, Colors, Tooltips, Drill, Labels, and Filters.

The mandatory shelves are X Axis and Y Axis. Note also that the fields placed on these two shelves may be easily swapped by switching X and Y.

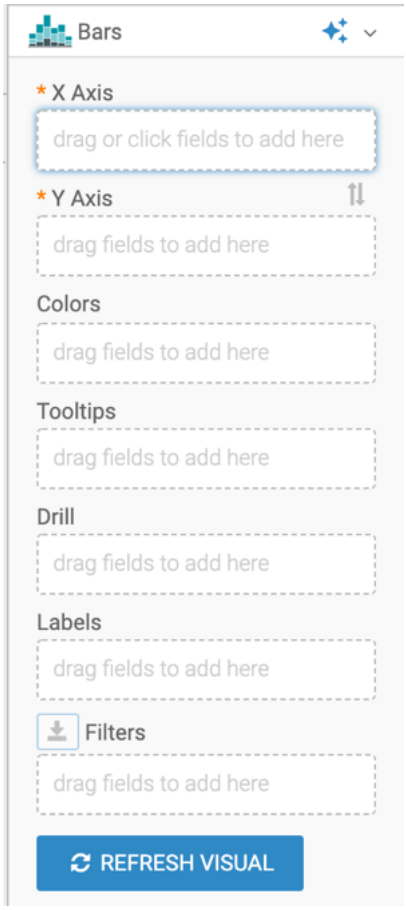

3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) Under Dimensions, select country and drag it to the X Axis shelf .
   b) Under Measures, select life_expectancy and drag it to the Y Axis shelf .
   c) Change the aggregate of the life_expectancy field from sum(life_expectancy) to avg(life_expectancy):
      - Click the field on the shelf of the visual to open the FIELD PROPERTIES menu.
      - Open the Aggregates menu and select Average from the list of aggregate functions.

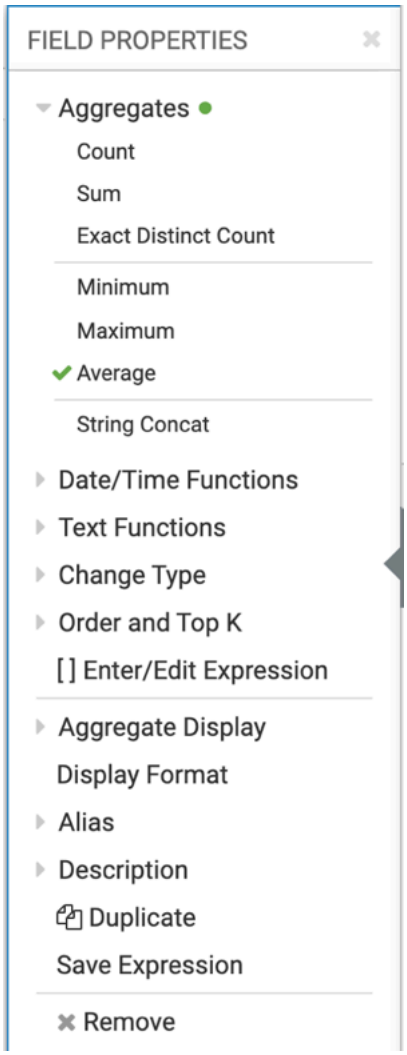

      - Click the x icon at the top of the FIELD PROPERTIES menu to close it.

        The shelf now contains the modified field with avg() aggregation function.

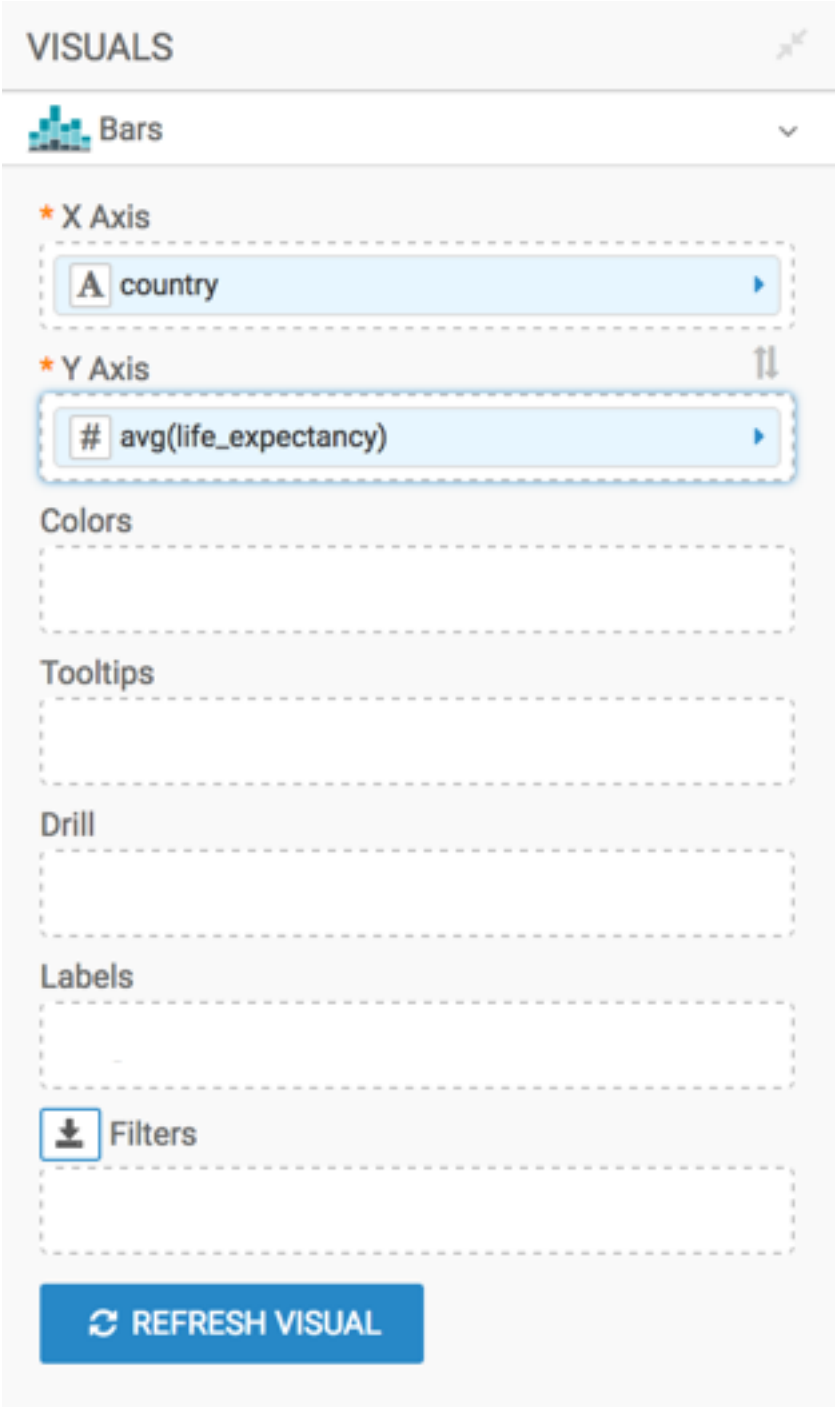

4. Alias the country field as Country, and the life_expectancy field as Life Expectancy:
   - Click the field on the shelf of the visual to open the FIELD PROPERTIES menu.
   - Open the Alias menu and enter the alias name of column in the textbox as it should appear in the visual.
   - Click the x icon at the top of the FIELD PROPERTIES menu to close it.

     Note that the shelf now shows the column with its alias name.

5. Click REFRESH VISUAL.

   The Bars visual appears.

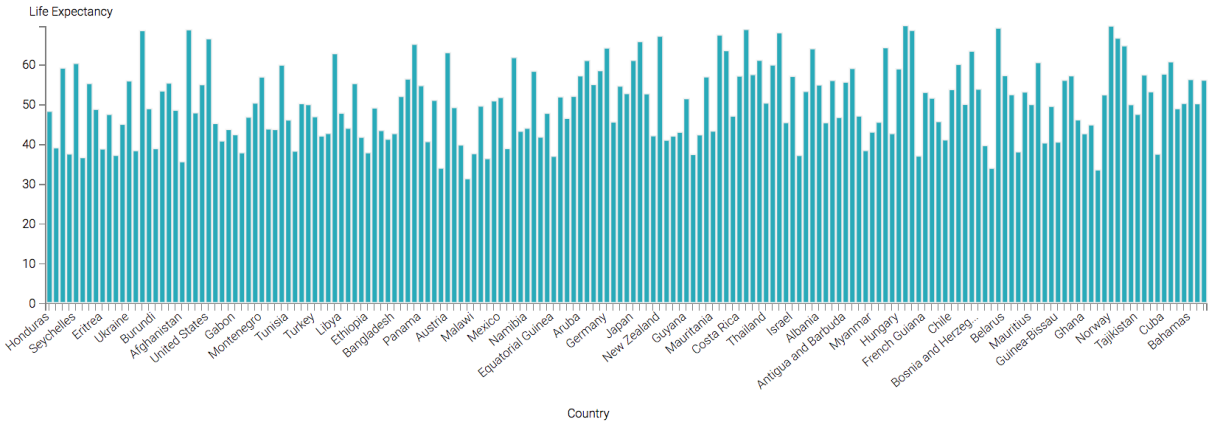

6. To see the bars as different colors, you can place a dimension to the Colors shelf.

   Drag the un_region field to the Colors shelf and click REFRESH VISUAL to see the changes.

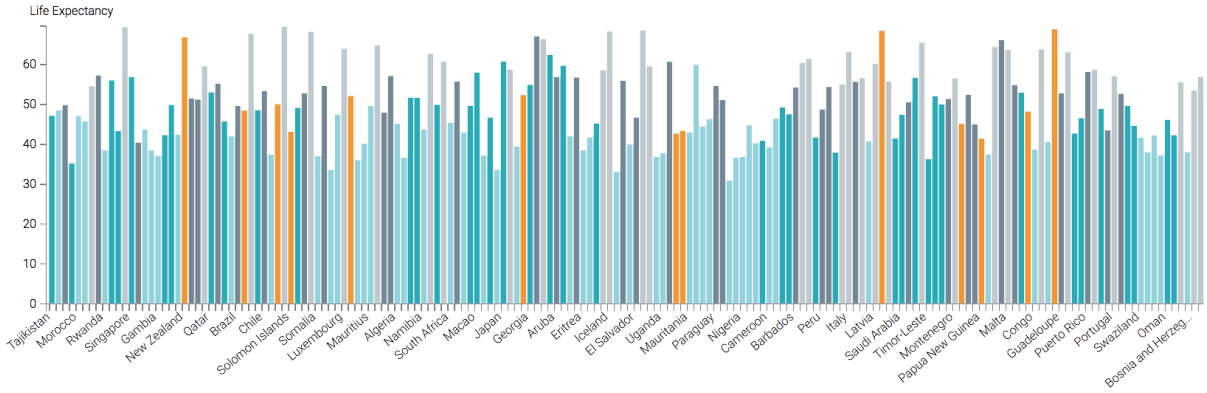

7. To group all countries of UN Regions together, change the order for the un_region field on the Colors shelf:
   - Click the field on the shelf of the visual to open the FIELD PROPERTIES menu.
   - Open the Order and Top K menu and select Ascending from the list of options.
   - 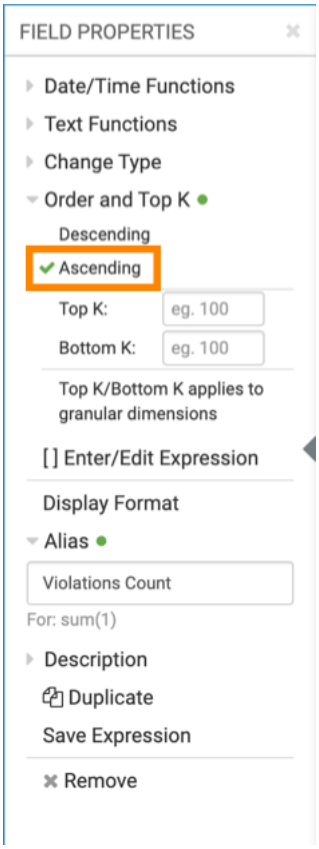

   - Click the x icon at the top of the FIELD PROPERTIES menu to close it.
8. Click REFRESH VISUAL to see changes.

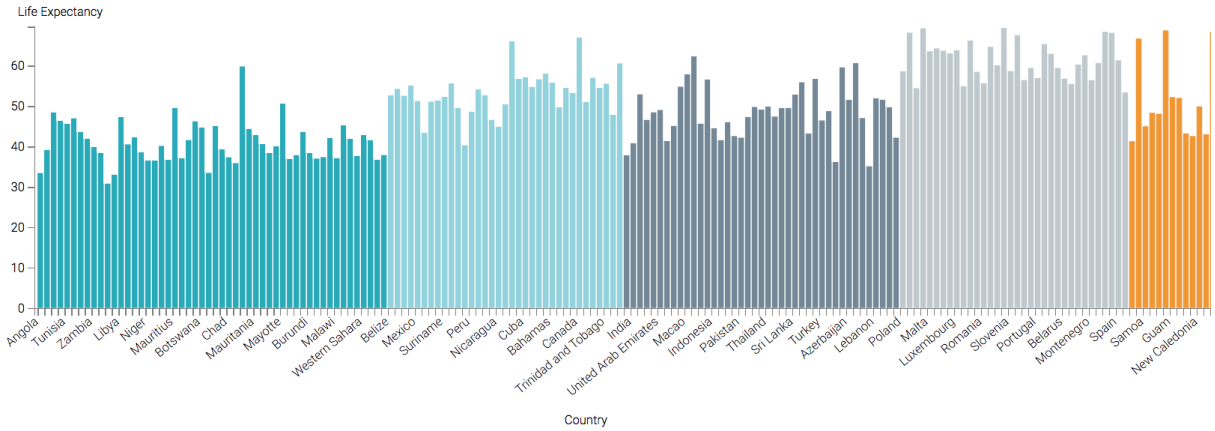

9. On the Filters shelf, add several Dimensions and Measures from the DATA menu.

   This enables you to dynamically control the data input, and discover the data at a more granular level.

   For example, from Dimensions, drag country, un_subregion, un_region, and year onto the Filters shelf. The filter selection wizard appears each time you place a field on a Filters shelf. You can specify filter values at this time, or click CANCEL and select them later.

10. Change the title to 'World Population - Bars'.

- Click (pencil) icon next to the title of the visualization to edit it and enter the new name.
- [Optional] Click (pencil) icon below the title of the visualization to add a brief description of the visual.

11. At the top left corner of the Dashboard Designer, click SAVE.

12. At the top left corner of the Dashboard Designer, click CLOSE.

What to do next

After learning how to create a basic bar chart, you can explore the following additional uses.

Related Information

Changing the min/max of the aggregate scale

Filtering 1: Bar chart by year

When working with a very large set of data, it is necessary to quickly access and examine a specific subset of information. Cloudera Data Visualization gives you the ability to filter information on all fields of the dataset.

About this task

This example shows how to use such a filter to limit the result set and visualization to one year. It uses the visual previously created in Bars. Clone it and make the changes to the duplicate visual.

To use this visual to see the bar chart for a specific year such as 2010, follow these steps:

Procedure

1. Set the filter value for year to 2010.
   - Click the field on the shelf of the visual to open the FIELD PROPERTIES menu.
   - Click Select values.

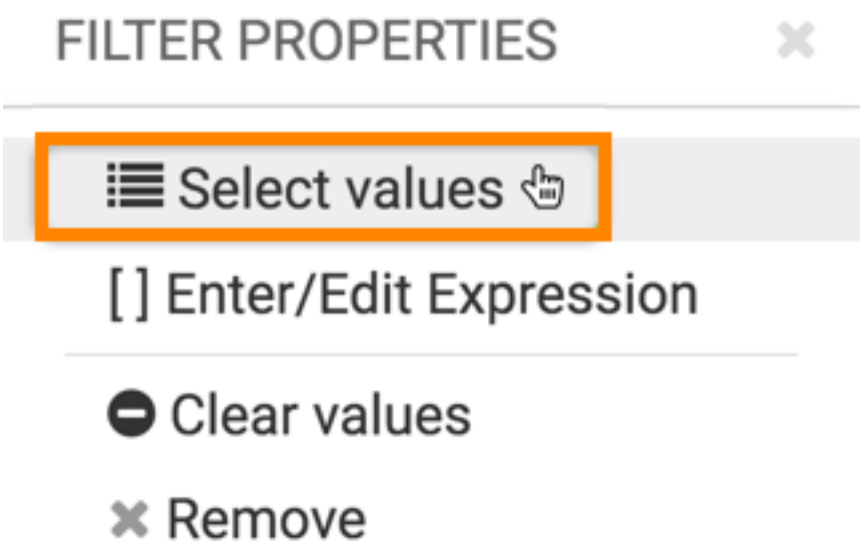

   - In the Filter for year modal window, select 2010 and click APPLY.

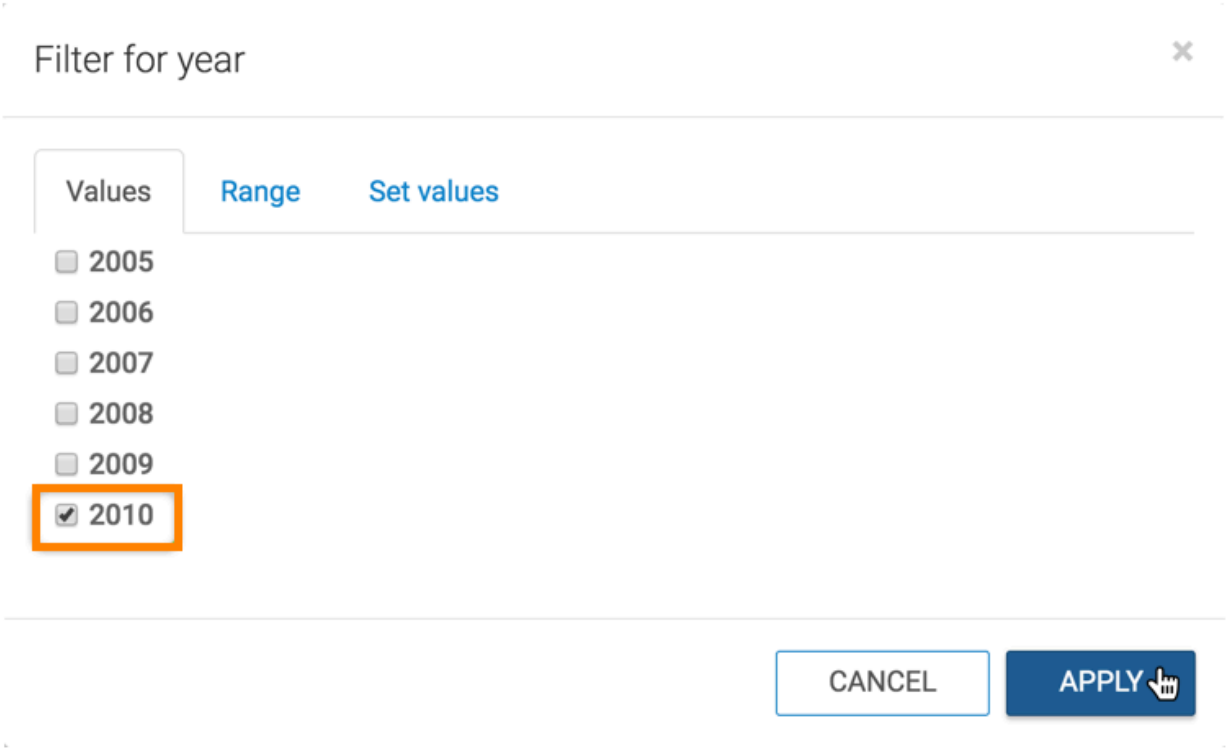

2. Click REFRESH VISUAL.

   The Bars visual for year 2010 has changed compared to the previous display, which was averaged over the 1900 - 2010 time span.

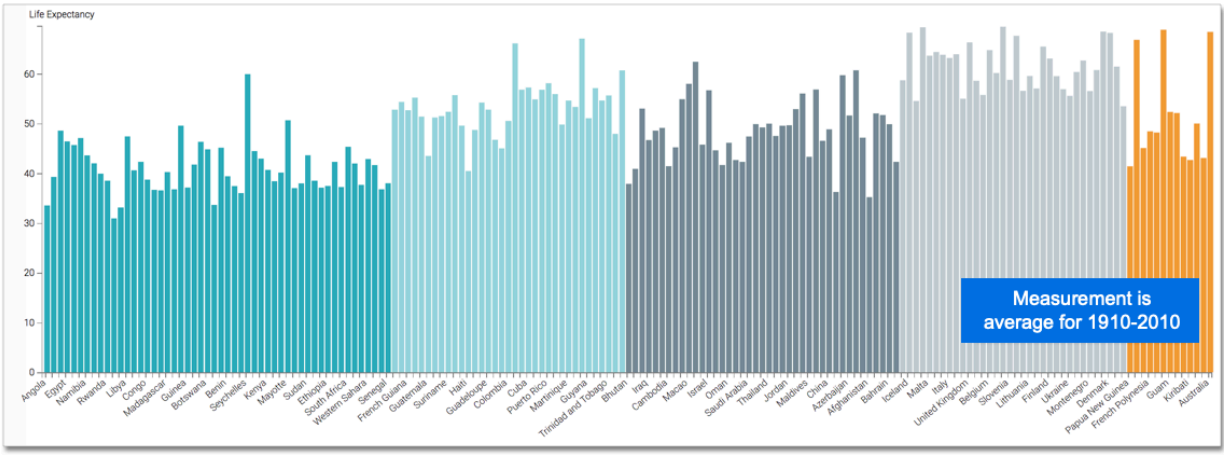

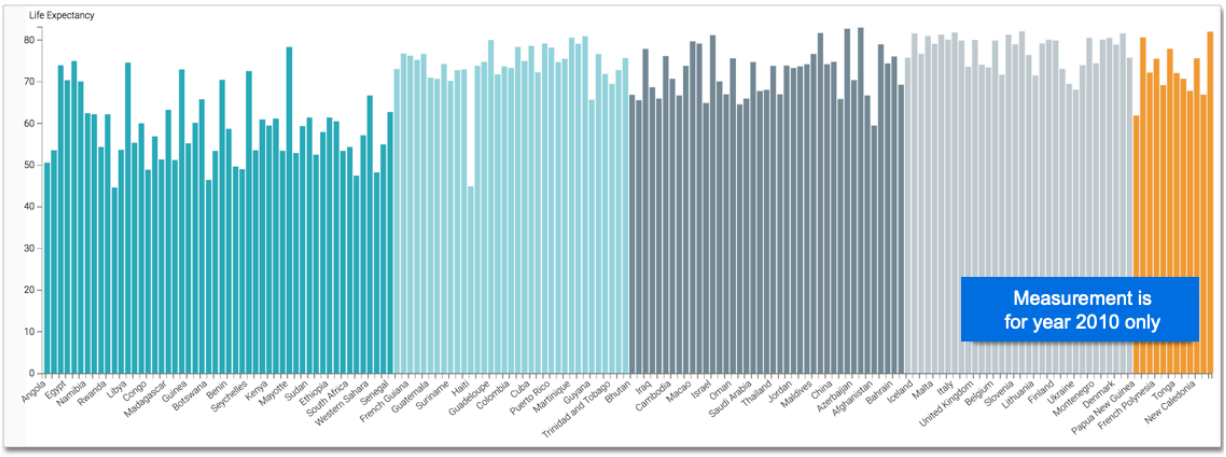

3. Rename the visual.
4. At the top left corner of the Dashboard Designer, click SAVE.

Filtering 2: bar chart by region

When working with a very large set of data, it is necessary to quickly access and examine a specific subset of information. Cloudera Data Visualization gives you the ability to filter information on all fields of the dataset.

About this task

This example shows how to use such a filter to limit the result set and visualization to a particular geographic region. It uses the visual previously created in Bars. Clone it and make the changes to the duplicate visual.

To use this visual to see a bar chart for a region, such as Oceania, follow these steps:

Procedure

1. Add the field un-region to the Filters shelf, and set its value to Oceania.
   a) Click the field on the shelf of the visual to open the FIELD PROPERTIES menu.
   b) Click Select values.

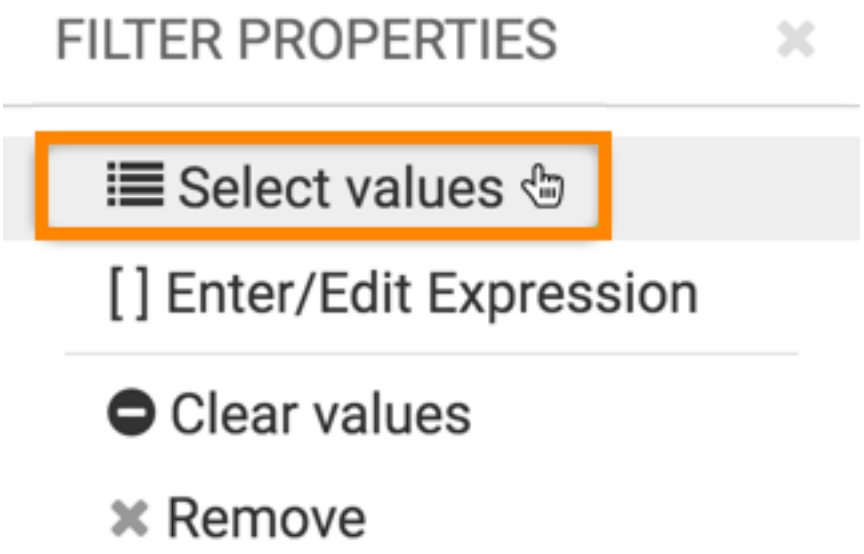

   c) In the Filter for un_region modal window, select Oceania and click APPLY.

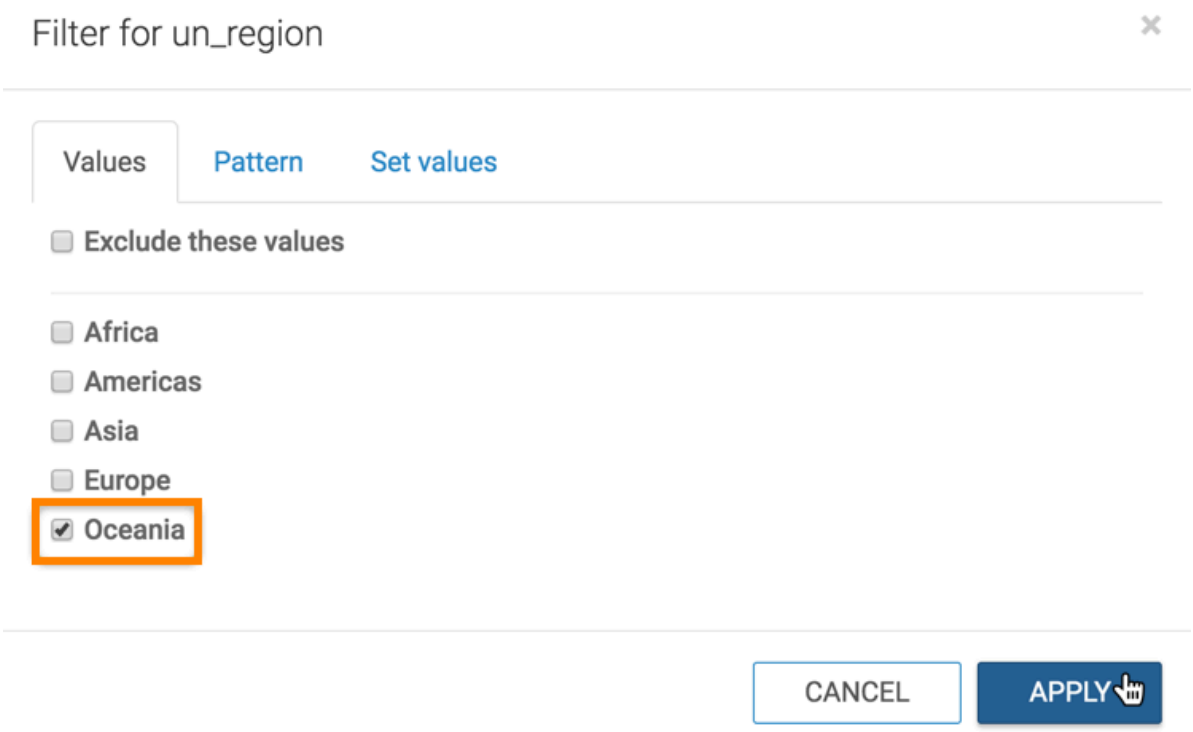

2. Click REFRESH VISUAL.
   The Bars visual changed to show only the countries in the Oceania UN Region.

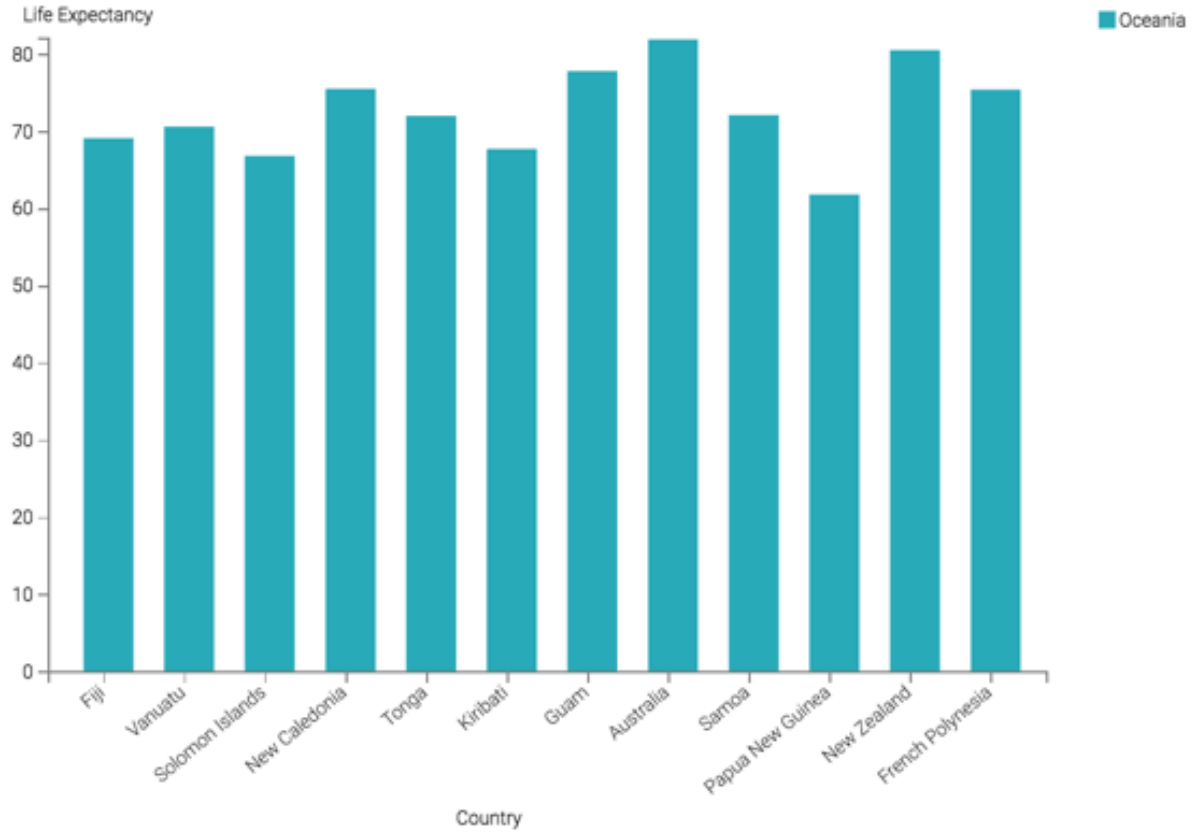

3. Remove un_region from the Colors shelf, and replace it with the country field.

4. Click REFRESH VISUAL.

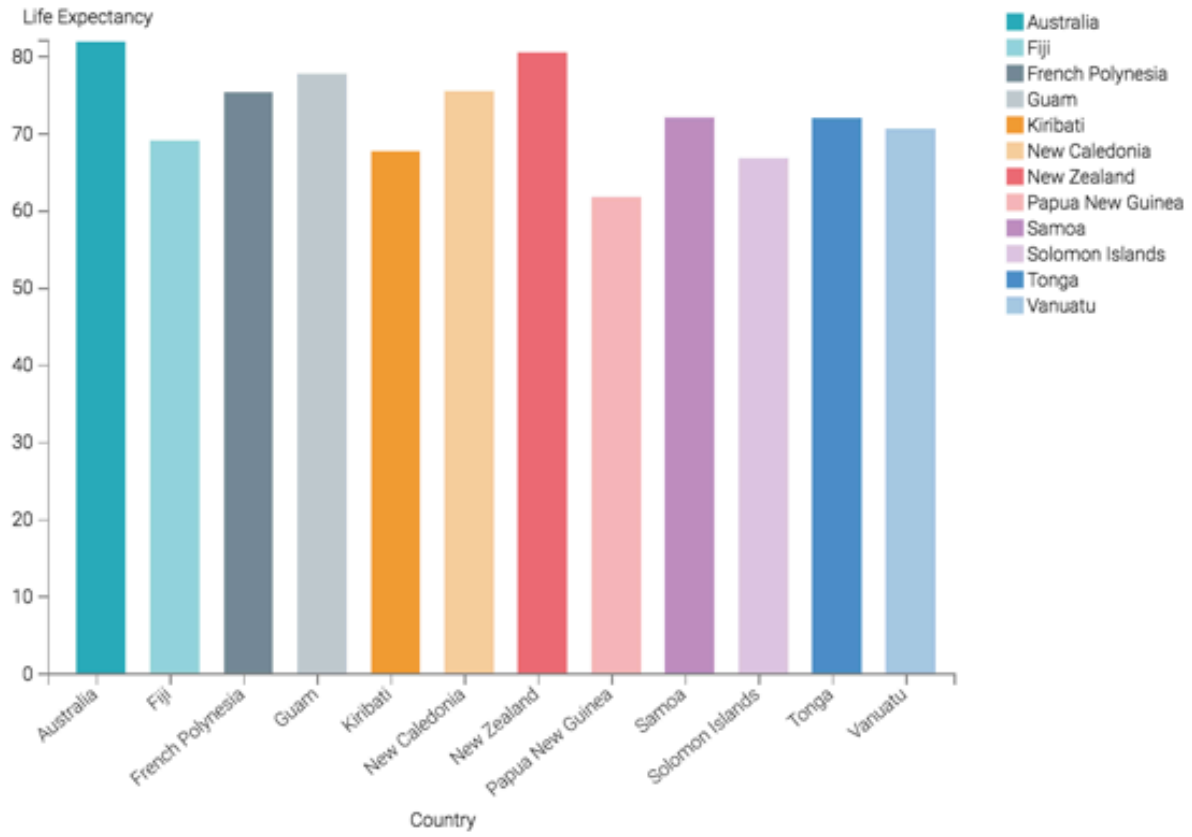

5. Rename the visual.
6. At the top left corner of the Dashboard Designer, click SAVE.

Horizontal bars

In general, dimensions of the dataset represent categories that are placed on the horizontal axis (and therefore the X shelf). The result set is rendered as a vertical bar chart. If your business requirements include the use of a horizontal bar chart, Cloudera Data Visualization makes it very easy to switch from an existing vertical bar chart to a horizontal one.

About this task

This example uses the visual previously created in Bars. Clone it, and make the changes to the duplicate visual.

To use this visual to create a horizontal bar chart, follow these steps.

Procedure

1. Click the (switch) icon between the X Axis and Y Axis shelves.

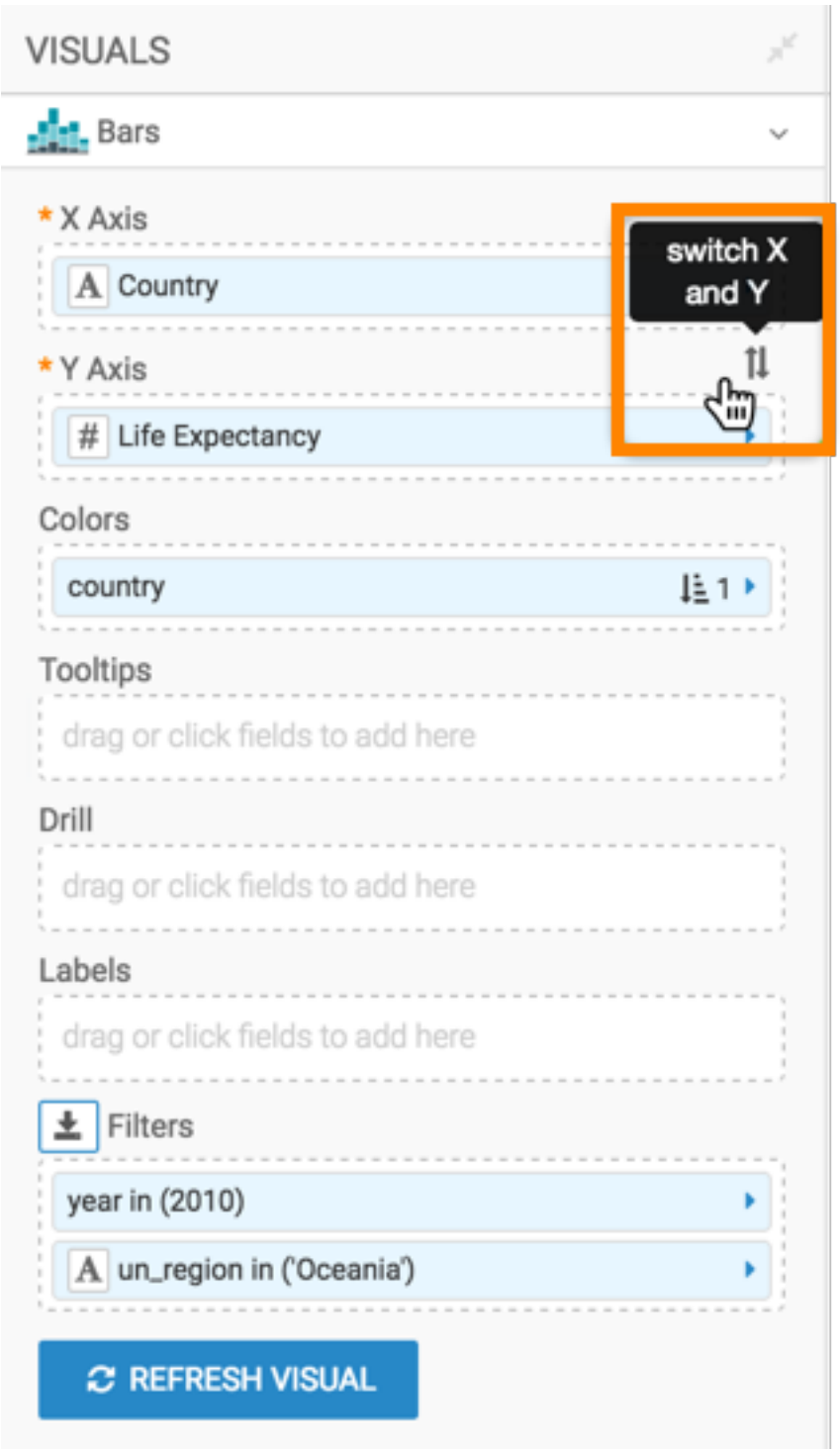

2. Click REFRESH VISUAL.

   The graph has changed, with country categories on the Y axis and horizontal bars.

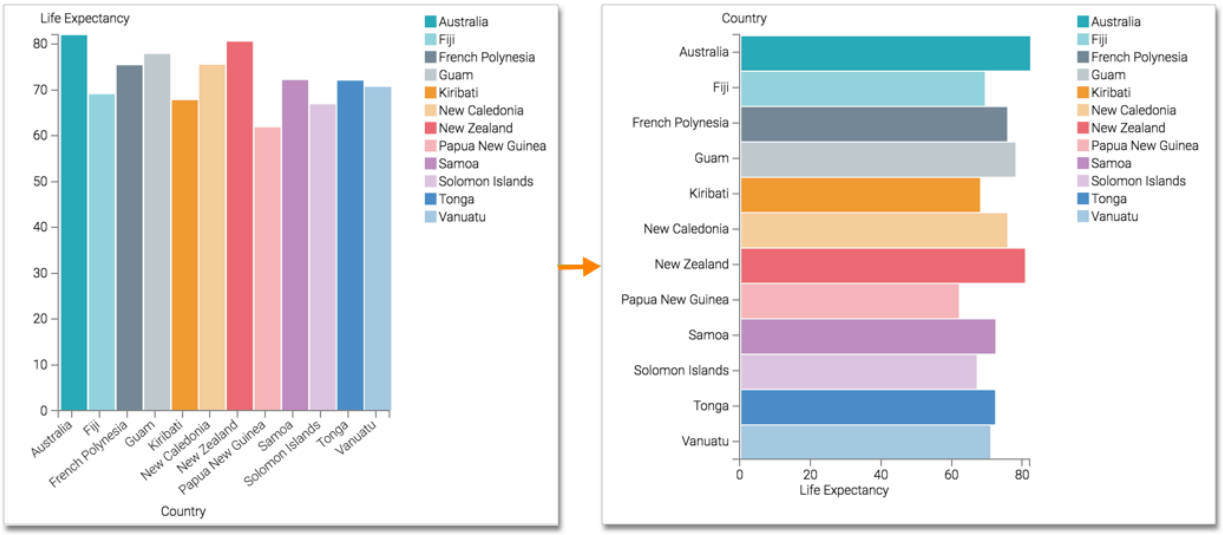

3. Rename the visual.
4. At the top left corner of the Dashboard Designer, click SAVE.

Stacked bars

While bar charts show comparisons among categories, Cloudera Data Visualization stacked bar charts show the bars divided into sub-parts. This demonstrates the cumulative effect of the component parts of a particular category.

About this task

This example uses the visual previously created in Bars. Clone it and make the changes to the duplicate visual.

To develop a stacked bar representation of European population divided by its subregions, follow these steps.

Procedure

1. Remove Life Expectancy from the Y Axis shelf and un_region from the Colors shelf.
2. From Dimensions in the DATA menu, drag un_subregion to the X Axis shelf.
3. Under Measures, select population and drop it to the Y Axis shelf.
4. Pull Country from the X Axis shelf to the Colors shelf.
5. On the Filters shelf, set year to 2010 and un_region to Europe.
6. Add all fields of interest to the Tooltips shelf.

   For example, add life_expectancy and gdp_per_capita.
7. Adjust Display Format for the values on the main shelves of the visual and in the Tooltip shelf.

   For example, set the gdp_per_capita to Currency, with $ symbol.
8. Alias the fields on the shelves as UN Subregion, Population, GDP per Capita, and so on:
   a) Click the field on the shelf of the visual to open the FIELD PROPERTIES menu.
   b) Open the Alias menu and enter the alias name of column in the textbox as it should appear in the visual.
   c) Click the x icon at the top of the FIELD PROPERTIES menu to close it.

      The shelf now shows the column with its alias name.

9. Click REFRESH VISUAL.

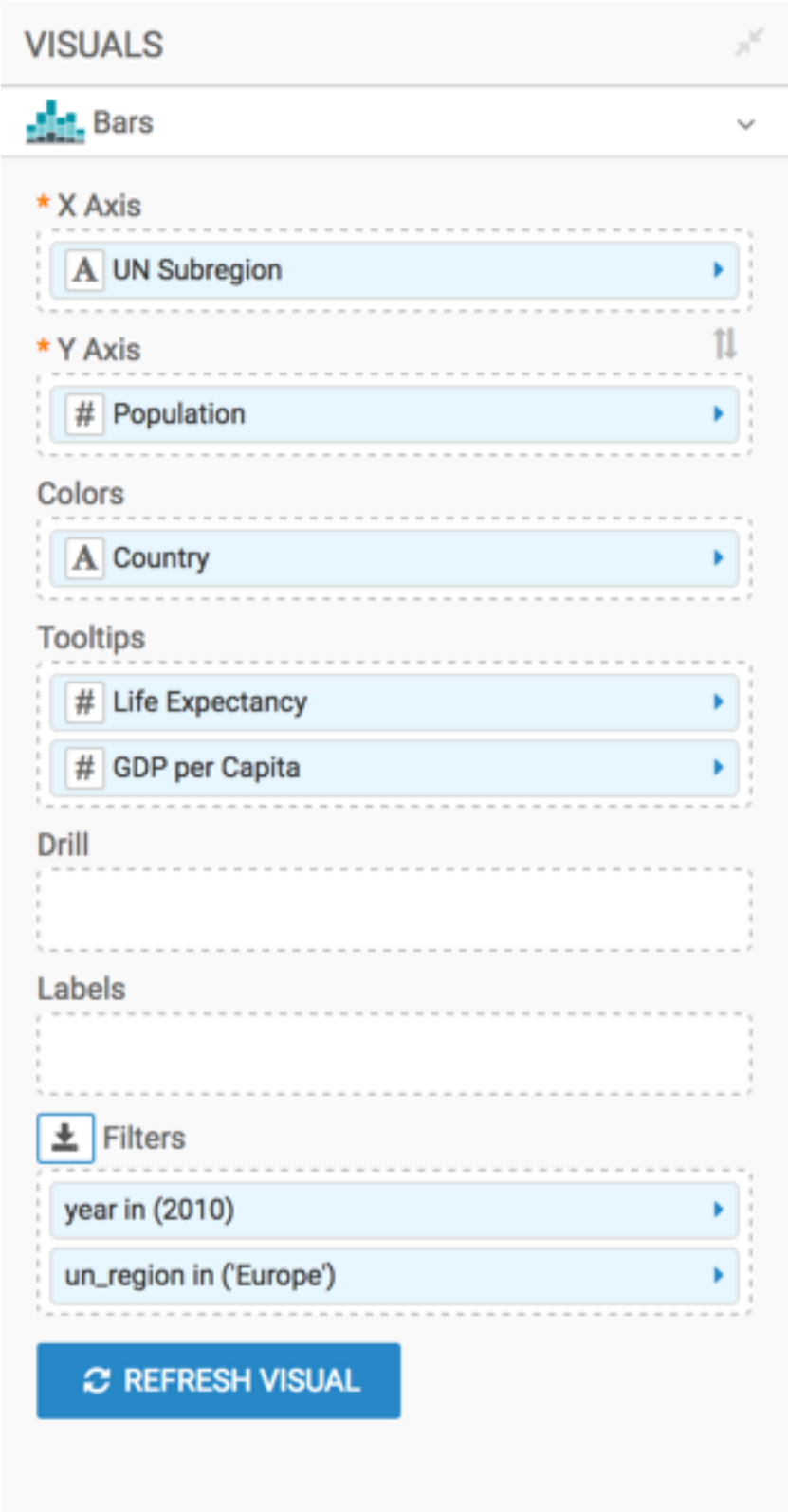

The visual has changed. You can see the relative size of total populations for the subregions, and the component countries for all subregions.

10. If you hover the mouse over the graph, you can see the tooltip text for each country.

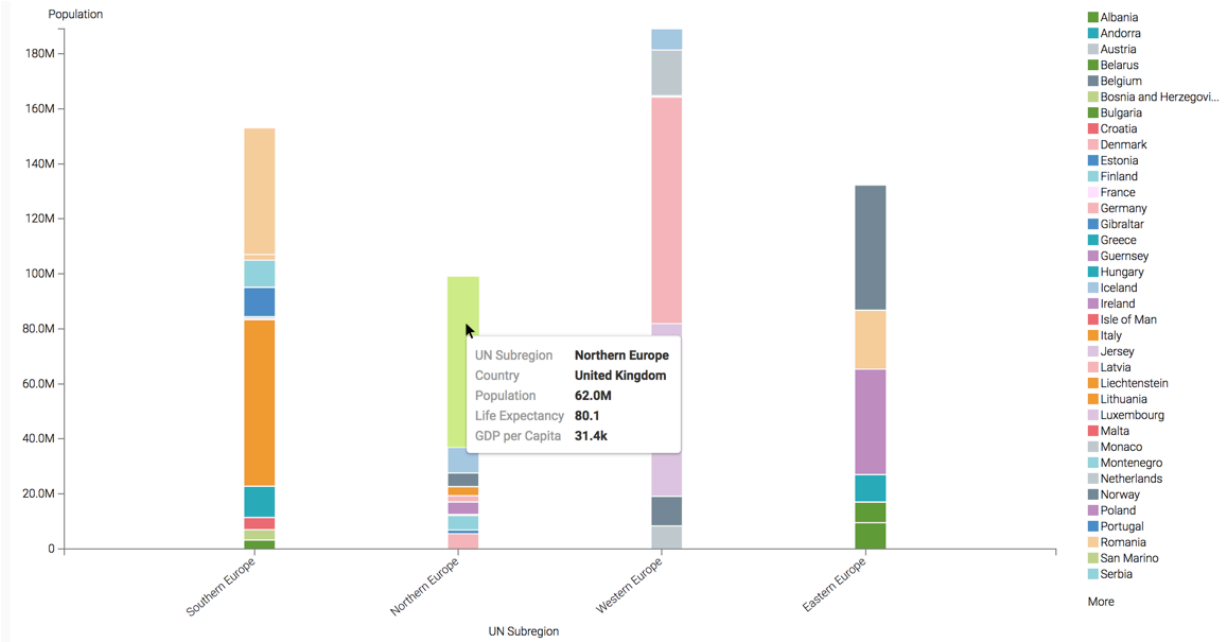

11. Rename the visual.

12. At the top left corner of the Dashboard Designer, click SAVE.

Labelled bars

In Cloudera Data Visualization, you can use the Labels shelf to display the measurement value. This way you can improve the comparison of relative measures on Bar visuals.

About this task

This example uses the visual previously created in Stacked bars. Clone it and make the changes to the duplicate visual.

To add labels to bar charts, follow these steps.

Procedure

1. Click inside the Labels shelf.
2. In the DATA menu, select the field for the label.

   This example uses the population field.
3. Set the Filter shelf to hold the following fields:
   - year in (2010)
   - un_subregion in ('Eastern Europe')

4. Increase the width of the bars as described in *Bar size range*.

   Your shelf setup should look something like the following:

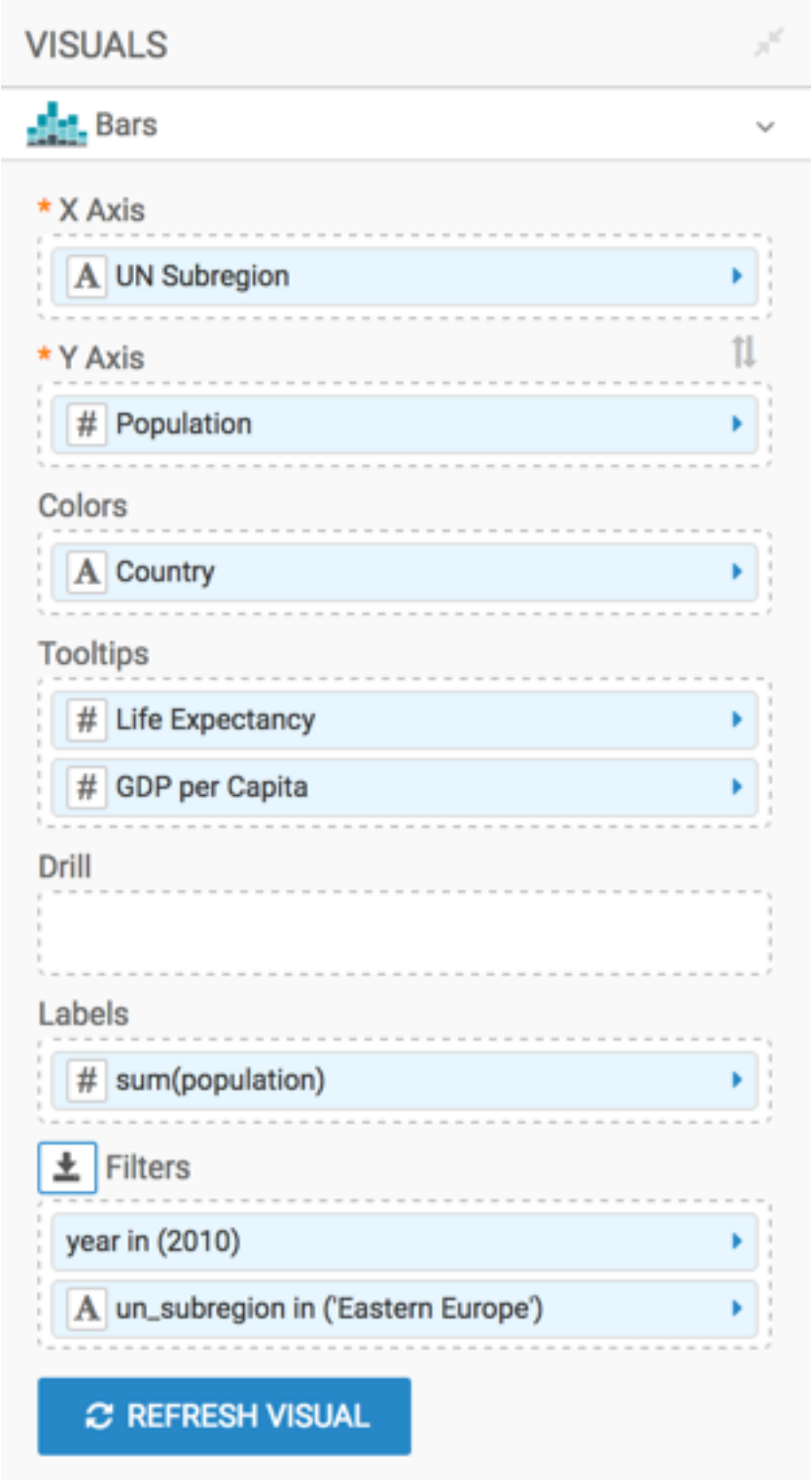

5. Click REFRESH VISUAL.

   The visual changed. You can see the relative size of total populations for the UN Subregion on one bar, and read off the value.

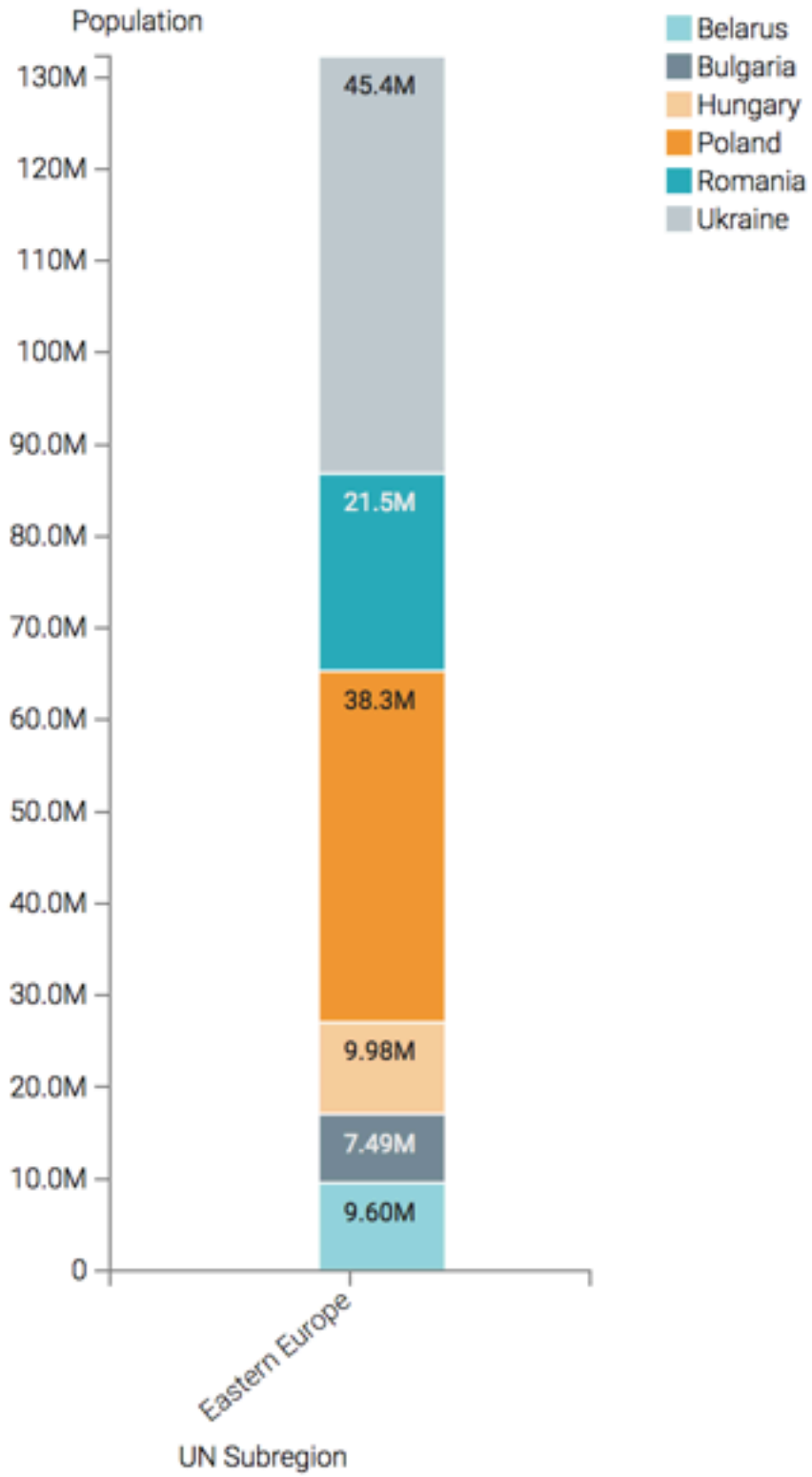

6. Rename the visual.

7. At the top left corner of the Dashboard Designer, click SAVE.

Shelves for bars

Overview of shelves for Cloudera Data Visualization Bars visuals.

A Bars visual has the following shelves:

X Axis

This is conceptually the X axis. Use a Dimension for horizontal bars or a Measure for vertical bars.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in columns.

Y Axis

This is conceptually the Y axis. Use a Measure for horizontal bars or a Dimension for vertical bars.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in rows.

Colors

To stack bars by category, add a Dimension field to this shelf.

This shelf displays only one field on the graph. Additional fields appear only in the Tooltip.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

Drill

Add Dimension fields from the DATA menu to this shelf.

When Drill Into context menu is enabled but no fields are specified, you can drill into any dimension. Specifying the fields on this shelf limits drilling to these dimensions.

This shelf accepts multiple fields.

Labels

Additional data labeling.

This shelf accepts at most one field.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Lines

In Cloudera Data Visualization, a line chart displays information as a series of data points, or markers, connected by straight line segments. It reveals trends or change over time. Line charts show relationships within a continuous data set, and can be applied to a wide variety of categories. While line charts are similar to scatter plots, the line chart orders the measurements along the primary axis and it is often used to visualize a time series.

About this task

Note:

This visual type supports smooth rendering of streaming data.

The following steps demonstrate how to create a new Lines visual on the World Life Expectancy dataset.

For an overview of shelves that specify this visual, see *Shelves for lines visuals*. For customization options, see *Settings for lines visuals*.

Procedure

1. Start a new visual based on dataset World Life Expectancy.

   For instructions, see *Creating a visual*.
2. In the VISUALS menu, find and click Lines.

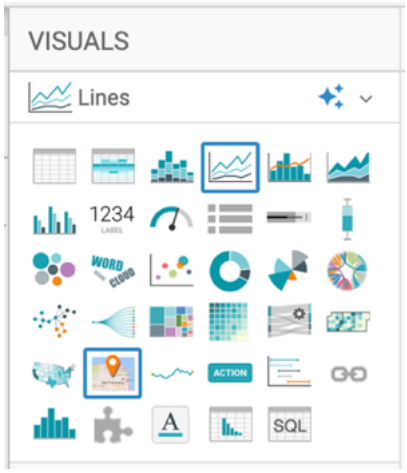

   The shelves of the visual changed. The mandatory shelves are X Axis and Y Axis.

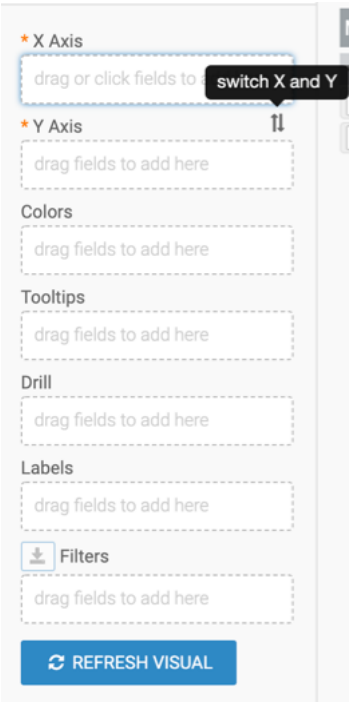

   The fields placed on these two shelves may be easily swapped by switching X and Y.
3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   - Under Dimensions, select year and drag it to the X Axis shelf .
   - Click the Y Axis shelf and select the life_expectancy field from the DATA menu by clicking it. This adds the field to the shelf.

4. Click the field on the X Axis shelf, and in the FIELD PROPERTIES menu, select Order and Top K Ascending .
5. If the field on the Y Axis appears as a sum aggregate, change the aggregate to avg(life_expectancy):
   a) Click the field on the Y Axis shelf to open the FIELD PROPERTIES menu.
   b) Select Aggregates Average .

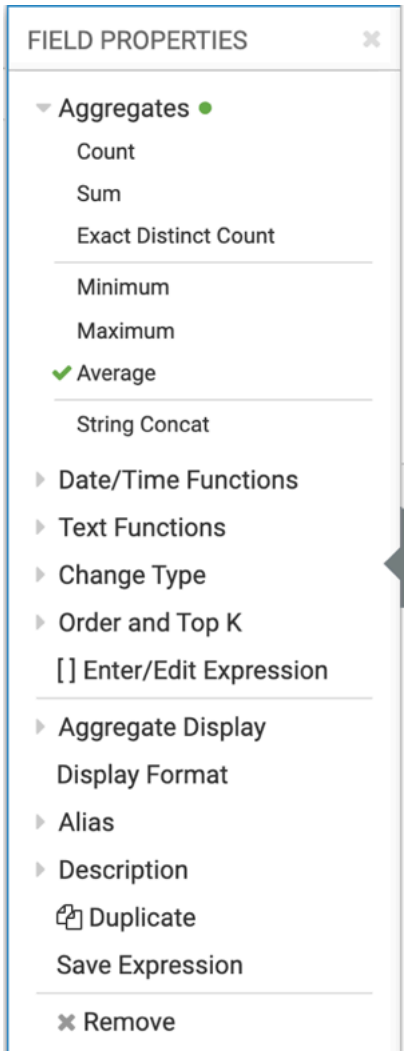

   c) Click the x icon at the top of the FIELD PROPERTIES menu to close it.

   The shelf now contains the modified field with avg() aggregation function.

   Alternatively, you can change the aggregation at the level of the dataset, as described in *Changing field aggregtion*.

6. Click REFRESH VISUAL.

   The Lines visual appears.

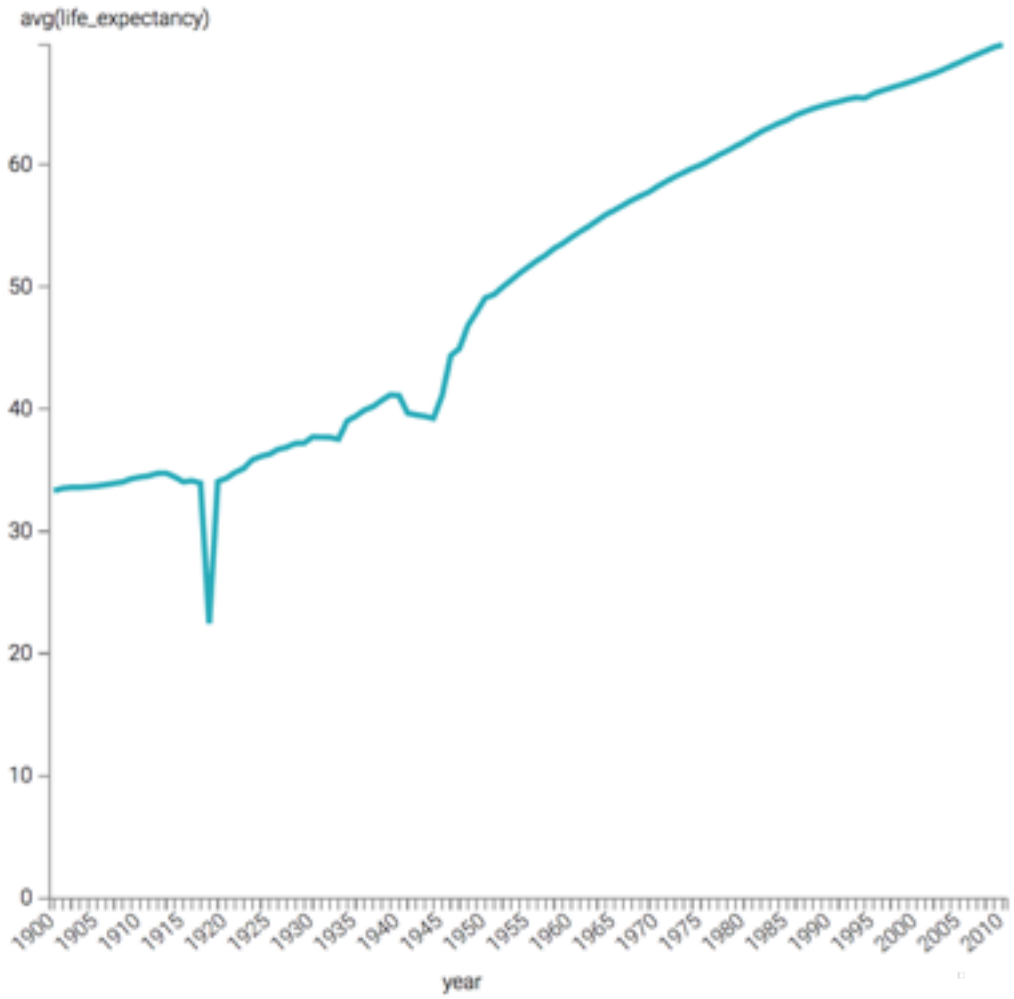

7. Alias the year field as Year, and the life_expectancy field as Life Expectancy:
   a) Click the field to open the FIELD PROPERTIES menu.
   b) Select Alias and enter the alias name of the column in the text box as it should appear in the visual.
   c) Click the x icon at the top of the FIELD PROPERTIES menu to close it.

   The shelf now shows the column with its alias name.
8. To see the individual countries in different colors, place the country field on the Colors shelf.

   To make the visual easier to understand, you can apply an *Alias* to it.

9. Click REFRESH VISUAL again.

The single line is broken out into country components, and Cloudera Data Visualization automatically provides a color legend.

10. On the Tooltips shelf, add several fields from Dimensions and Measures.

This enables you to see specific descriptive information in your visuals, such as input values, segment affiliation, and calculations.

11. On the Filters shelf, add several Dimensions and Measures from the DATA menu.

This enables you to dynamically control the data input and discover the data at a more granular level.

For example, from Dimensions, drag country, un_subregion, un_region, year, and population onto the Filters shelf.

12. On the Filters shelf:

a) Click the population field to open the FIELD PROPERTIES menu.
b) Select []Enter/Edit Expression.
c) Enter [population]>10000000.
d) Click VALIDATE & SAVE.

13. Click REFRESH VISUAL.

The updated Lines visual appears. Some of the lines do not originate at the beginning of the time series. This is because their population were lower than the 10,000,000 threshold that you set in the population filter.

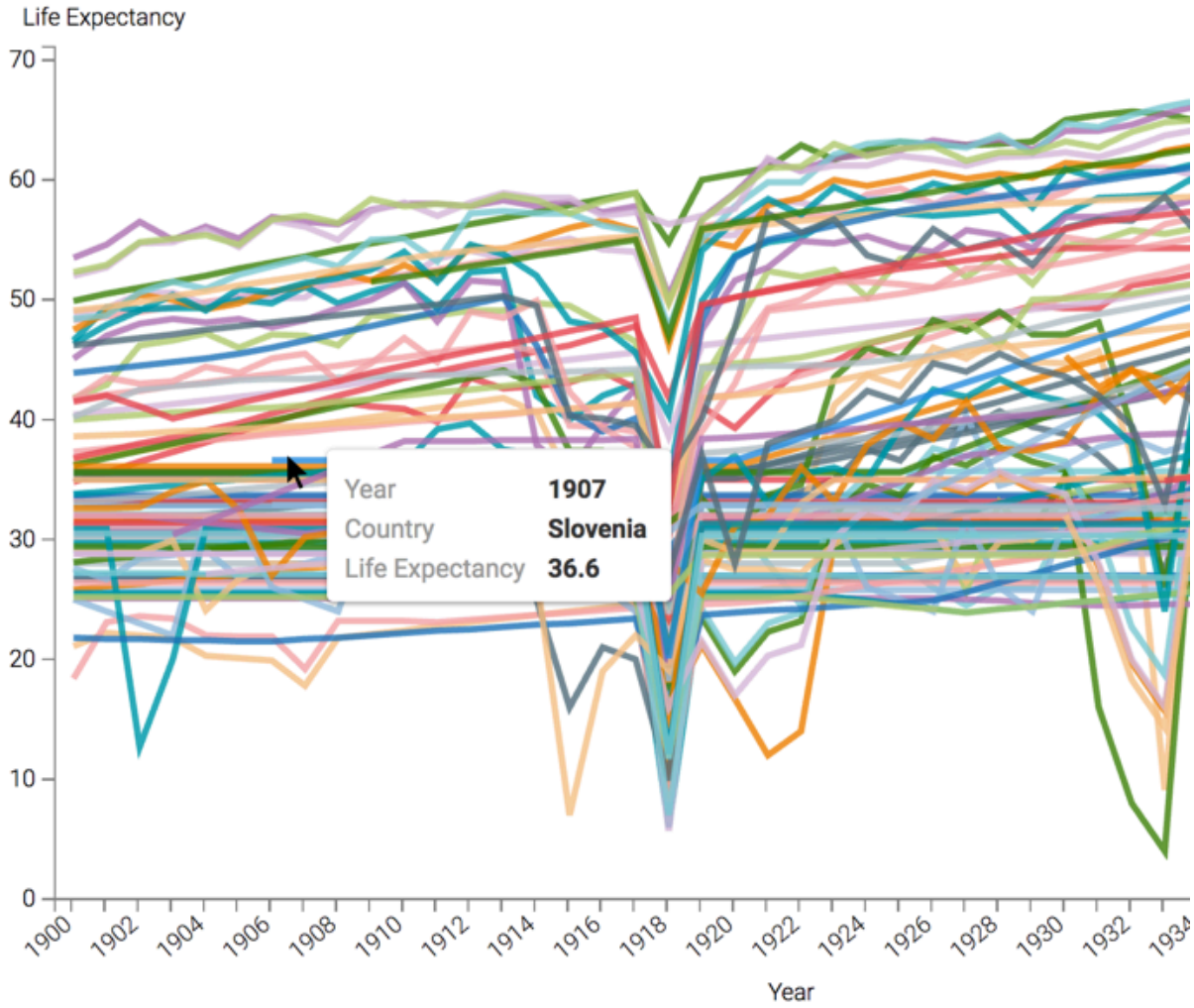

14. To limit the dimensions available to the Drill Into feature (it appears on dashboards that include the visual), specify the fields that can remain visible by placing them on the Drill shelf.

15. Click the (pencil) icon next to the title of the visual to change the title to 'World Population - Lines'.

16. At the top left corner of the Dashboard Designer, click SAVE.

Related Information

Changing the min/max of the aggregate scale

Lines with dual axes

It is often enlightening to see related dimensions on the same line chart. It clearly shows both correlation and pattern. However, values may be measuring completely different things, or the range of values (scale) is too different to show both lines clearly. Cloudera Data Visualization offers a dual axis option for situations like these.

About this task

The following steps demonstrate how to create a Lines visual that uses a double axis. The example is based on the World Life Expectancy dataset.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.

   For instructions, see *Creating a visual*.
2. In the VISUALS menu, select the Lines visual type.
3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) Under Dimensions, select year and place it on the X Axis shelf. Specify ascending order and alias the field as 'Year'.
   b) Under Measures, select life_expectancy and place it on the Y Axis shelf, and add an alias to the field.
   c) Under Measures, select Population and place it on the Colors shelf, and add an alias to the field.
4. Click REFRESH VISAL.

   The Lines visual appears. You can see the linear plot of population and life expectancy, on the same axis.

   It is impossible to see the values on the life expectancy line because the population line sets the scale for the Y axis.

   In this case, the label does not correspond to the scale shown, because this information comes from the Y axis shelf.

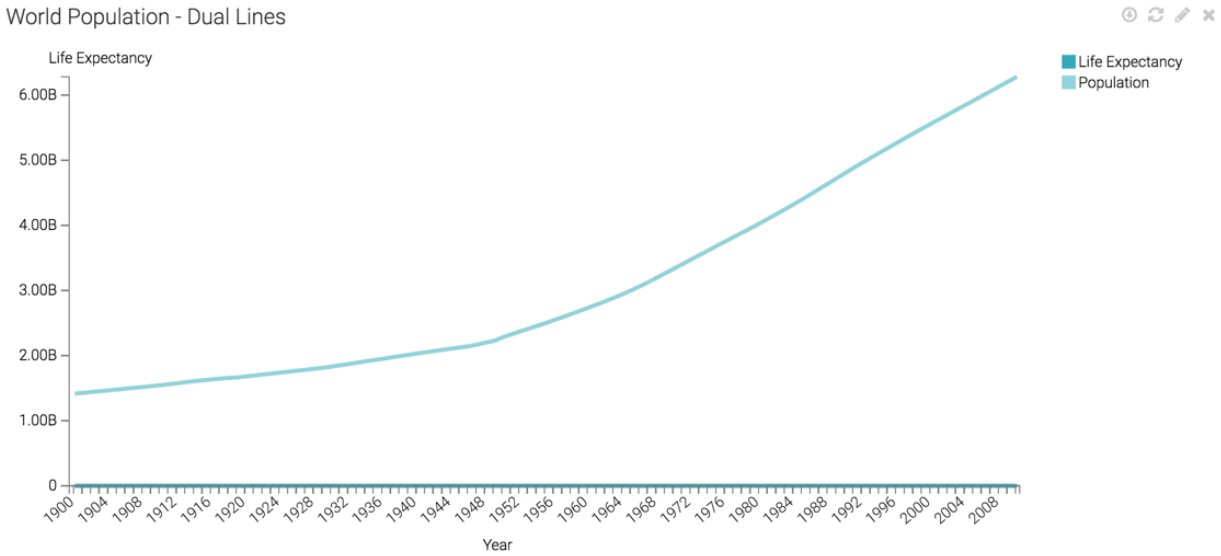

5. On the Colors shelf, click the sum(population) field to open the FIELD PROPERTIES menu.

6. Select Axis Secondary Axis .

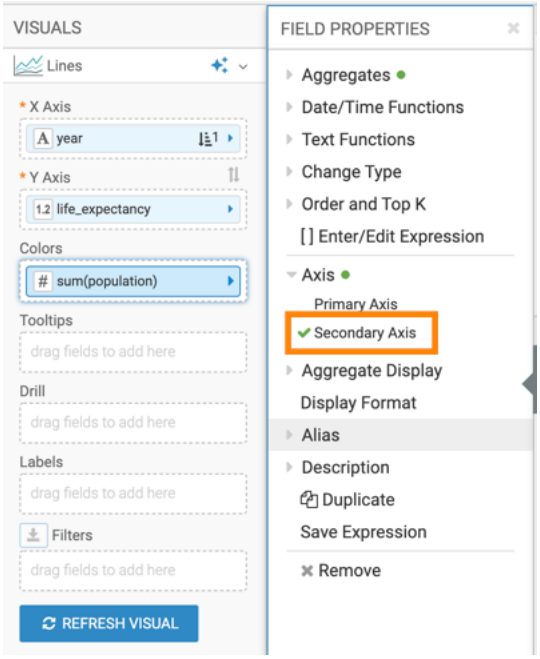

7. Click REFRESH VISUAL.

   The two lines can now be seen clearly. There is a vertical axis both on the left and the right side of the visual. The linear plot of population and life expectancy are visible each on its own axis.

   For convenience, the colors of the axis labels match the line.

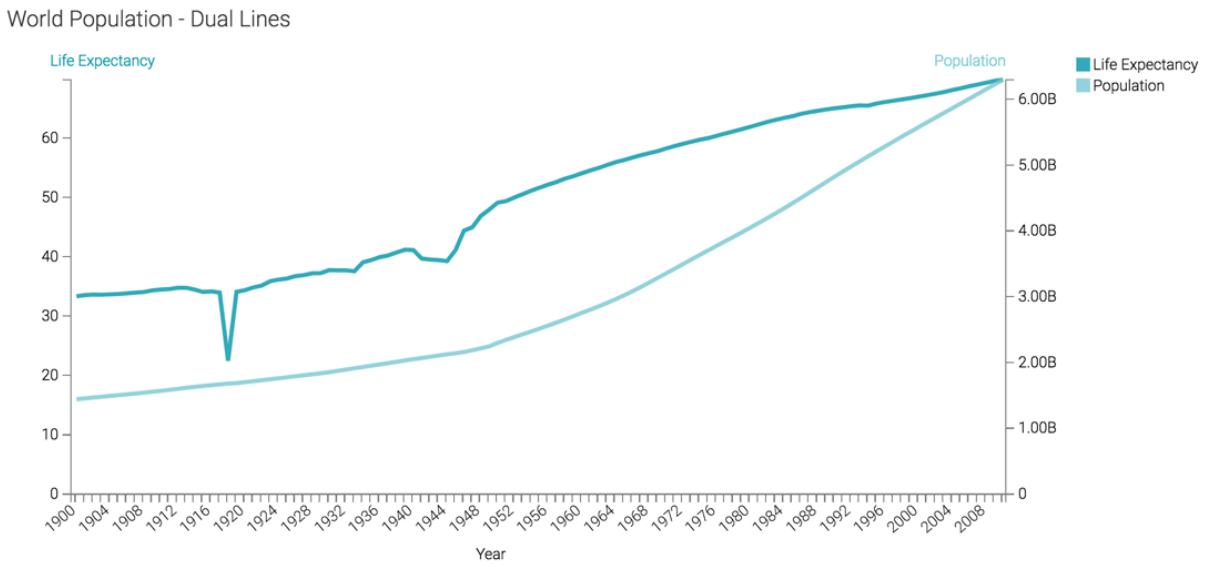

8. Save the visual as World Population - Dual Lines.

What to do next

Compare this approach with plotting multiple measures as trellised charts described in *Trellis on measures*.

Shelves for lines

Overview of shelves for Cloudera Data Visualization Line visuals.

A Lines visual has the following shelves:

X Axis

This is conceptually the X axis. Use a Dimension for a horizontal line or a Measure for a vertical line.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in columns.

Y Axis

This is conceptually the Y axis. Use a Measure for a horizontal line or a Dimension for a vertical line.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in rows.

Colors

To stack lines by category, add a Dimension field to this shelf.

This shelf displays only one field on the graph. Additional fields appear only in the Tooltip.

Tooltips

Add Measure fields from the DATA menu for additional information in the graph's Tooltip.

This shelf accepts multiple fields.

Drill

Add Dimension fields from the DATA menu to this shelf.

When Drill Into context menu is enabled but no fields are specified, you can drill into any dimension. Specifying the fields on this shelf limits drilling to these dimensions.

This shelf accepts multiple fields.

Labels

Additional data labeling.

This shelf accepts at most one field.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Combined bar/line visuals

In Cloudera Data Visualization, Combined Bar/Line (Combo) visuals unite bar and line plots to enable easy comparison of trends and measures across the same dimensions.

Note: This visual type supports smooth rendering of streaming data.

The bars and the line can share the same vertical axis, or use a secondary axis. For more information, see Creating basic combo visuals and Assigning a measure to a secondary axis.

You can optionally render the line portion of the visual as an area. For more details, see Change the line to an area.

Related Information

Changing the min/max of the aggregate scale

Creating basic combos

Cloudera Data Visualization enables you to create Combined Bar/Line (Combo) visuals.

About this task

The following steps demonstrate how to create a new combined bar and line chart .

For an overview of shelves that specify this visual, see *Shelves for combos*.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.
   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Combined Bar/Line.

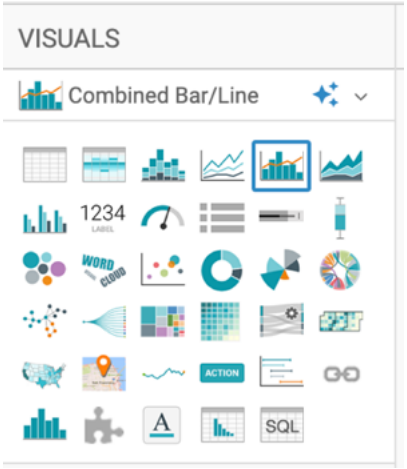

The shelves of the visual changed.

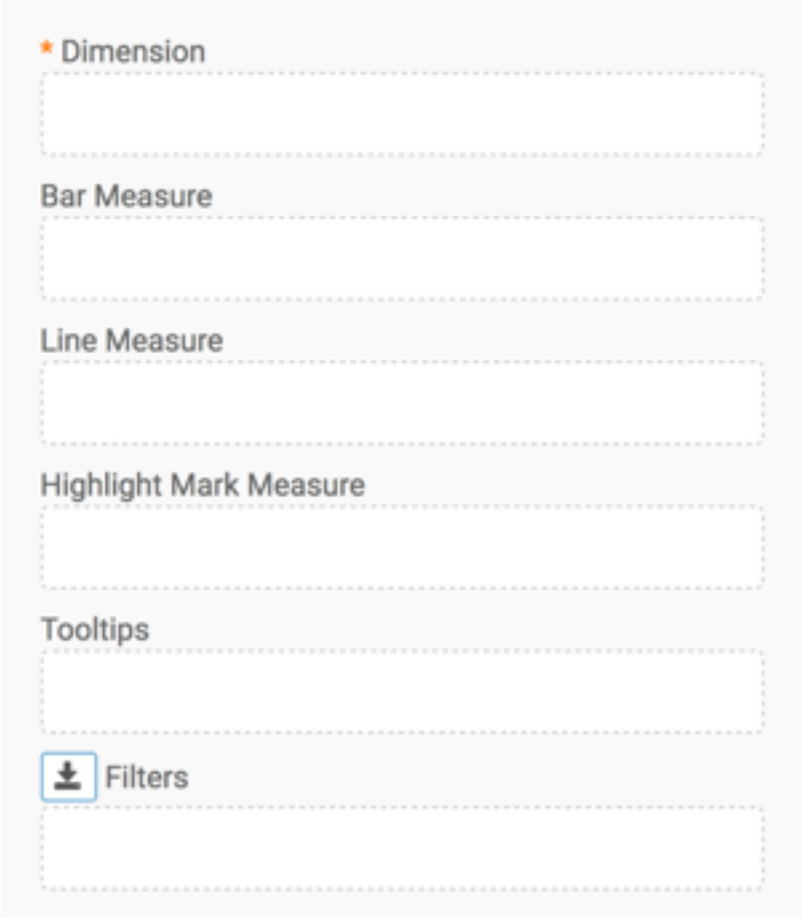

The mandatory shelf for Combined Bar/Line visuals is Dimension that maps to the X-axis.

Bar Measure and Line Measure both map to the Y-axis.

3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) Under Dimensions, select year and drag it to the Dimension shelf.
   b) Under Measures, select life_expectancy and drag it to the Bar Measures shelf.
   c) Under Measures, select gdp_per_capita and drag it to the Line Measures shelf.

4. Change the aggregate of the life_expectancy field from sum(life_expectancy) to avg(life_expectancy):
   a) Click the life_expectancy field to open the FIELD PROPERTIES menu.
   b) Select Aggregates Average .

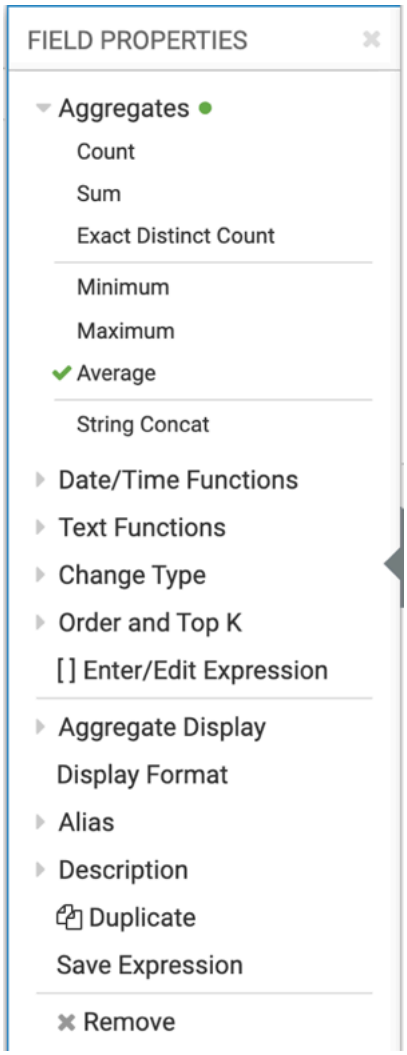

   c) Click the x icon at the top of the FIELD PROPERTIES menu to close it.

5. Similarly, change the aggregate of the gdp_per_capita field from sum(gdp_per_capita) to avg(gdp_per_capita):
   a) Click the gdp_per_capita field to open the FIELD PROPERTIES menu.
   b) Select Aggregates Average .
   c) Click the x icon at the top of the FIELD PROPERTIES menu to close it.

   The shelves of your visual should now look similar to the following image:

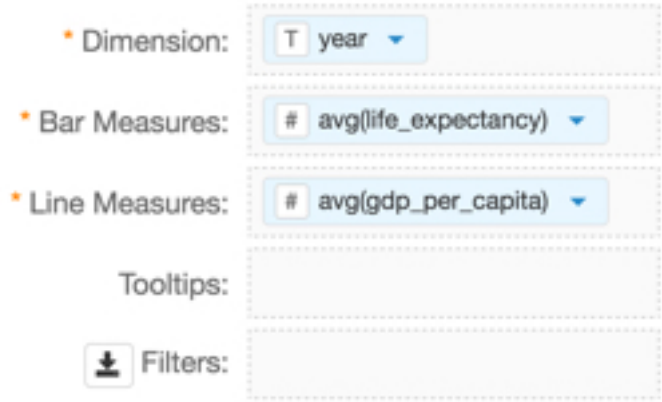

6. Click REFRESH VISUAL.

   The combined bar and line visual appears.

   The two legends for the Y axis match the color of the plot: the orange line represents sum(gdp_per_capita) and the blue bars represent avg(life_expectancy). However, you may not be able to see the bars because the values of that measure are much smaller than the values for the line measure. To see the bars, *assign a measure to a secondary axis*.

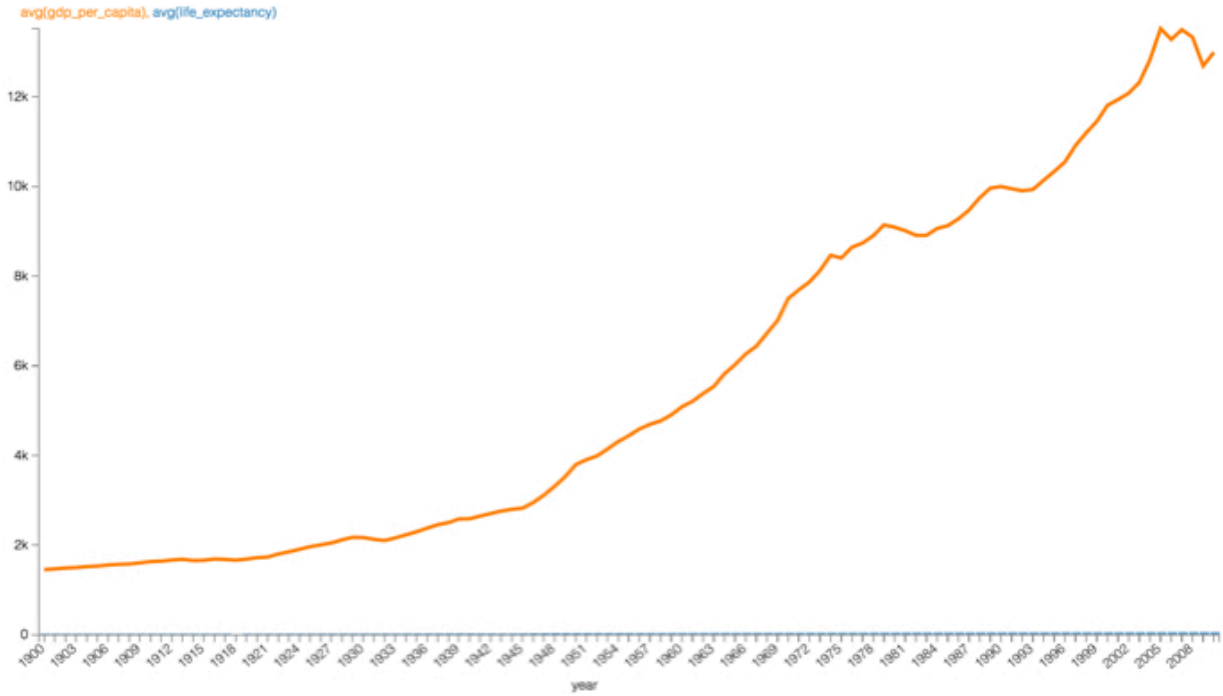

7. Alias the life_expectancy field as Life Expectancy, and the gdp_per-capita field as GDP Per Capita:
   a) Click the fields to open the FIELD PROPERTIES menu.
   b) Select Alias and enter the alias name of the column in the text box as it should appear in the visual.
   c) Click the x icon at the top of the FIELD PROPERTIES menu to close it.
8. Click REFRESH VISUAL again.

Assigning a measure to a secondary axis

Sometimes plotting two measures on the same axis does not show the bars because of their relative values, or because of their incompatible units of measurement. To be able to clearly see both a line and the bars in a Combined Bar/Line (Combo) visual, Cloudera Data Visualization enables you to specify a separate axis for each of the measures in the combined visual.

Before you begin

You have a already created a basic Combo visual based on the World Life Expectancy dataset.

About this task

Follow these steps to specify a different axis for each of the measures in your visual:

Procedure

1. Assign the 'GDP per Capita' field to the secondary axis:
   a) Click the field to open the FIELD PROPERTIES menu.
   b) Select Axis Secondary Axis .

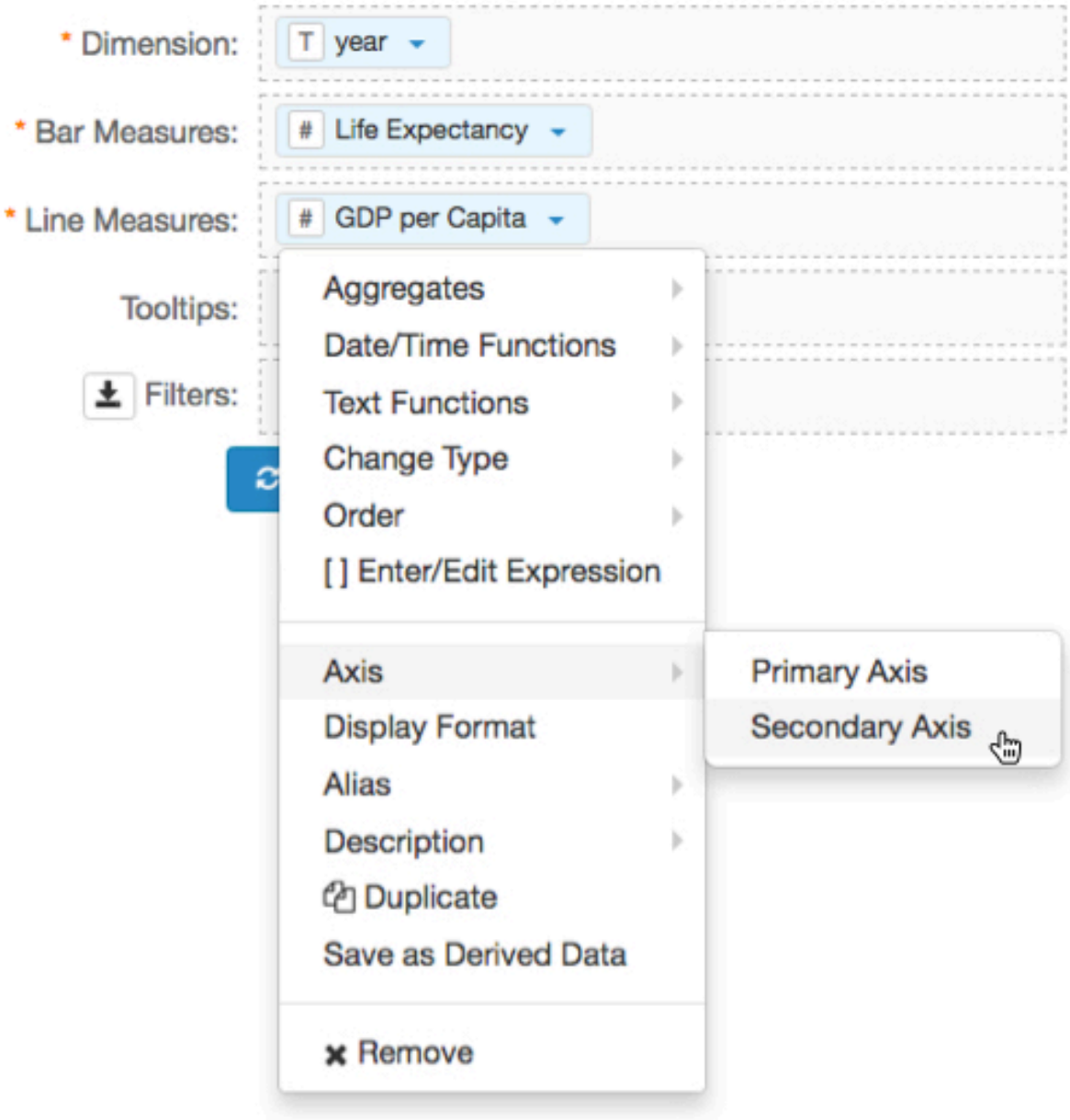

2. Click REFRESH VISUAL.

   Both the blue bars and the orange line are now clearly visible.

   The graph lists Life Expectancy on the left-hand vertical axis, in the range of 0 to 70, and the GDP per Capita on the right-hand vertical axis, in the range of 0 to 14k.

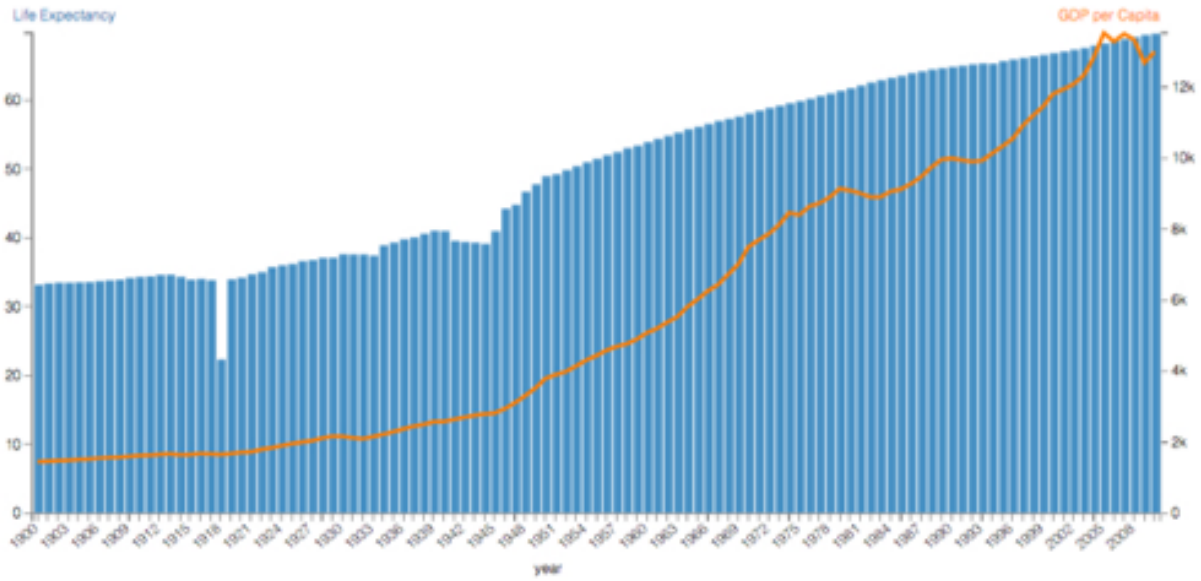

3. Click the (pencil) icon next to the title of the visual to edit it, and change the title to 'World Population - Dual Bar and Line'.
4. At the top left corner of the Dashboard Designer, click SAVE.

Change line to area

Cloudera Data Visualization allows you to convert the line to a semi-transparent area in Combined Bar/Line (Combo) visuals for better representation of data.

The default set-up is a combination of a bar measurement and a line measurement. To convert the line to a semi-transparent area that allows you to see the bar chart too, use the VISUAL SETTINGS interface, as described in *Converting line to area visual*.

Shelves for combos

Overview of shelves for Cloudera Data Visualizationn Combined Bar/Line (Combo) visuals.

A Combo visual has the following shelves:

Dimension

This is conceptually the X axis. Use a Dimension field to create bars and lines/areas for aggregate values in this dimension.

This shelf accepts a single field.

Bar Measure

This is conceptually the Y axis. Use a Measure field from the DATA menu to display aggregate values as bars.

This shelf accepts a single field.

Line Measure

This is conceptually the Y axis. Use a Measure field from the DATA menu to display aggregate values as lines or areas.

This shelf accepts a single field.

Highlight Mark Measure

This is conceptually the Y axis. Use a Measure field from the DATA menu to highlight aggregates of this measure.

This shelf accepts a single field.

Tooltips

Add Measure fields from the DATA menu for additional information in the graph's Tooltip.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Areas

In Cloudera Data Visualization, Areas visuals are a good choice for representing related values over time because they emphasize the relative proportions of totals or percentage relationships. With volume stacked beneath the line, the chart clearly demonstrates both the total of the fields and their relative size.

About this task

Note: This visual type supports smooth rendering of streaming data.

The following steps demonstrate how to create a new Areas visual on the World Life Expectancy dataset. For an overview of shelves that specify this visual, see *Shelves for areas*.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.

   For instructions, see *Creating a visual*.
2. In the VISUALS menu, find and click Areas.

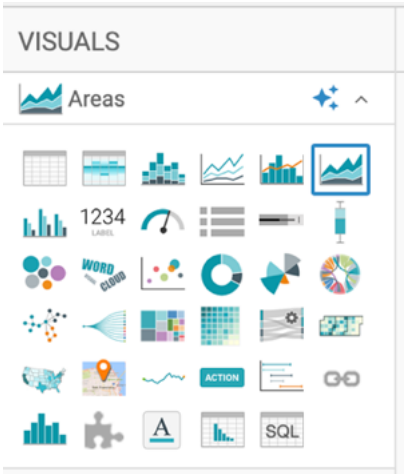

   The shelves of the visual changed. The mandatory shelves are X Axis and Y Axis. The fields placed on these two shelves may be easily swapped by switching X and Y.

3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) Under Dimensions, select year and drag it to the X shelf.
   b) Under Measures, select population and drag it to the Y shelf.

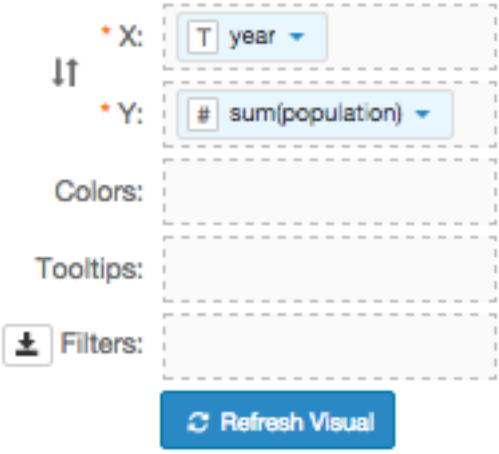

4. Change the aggregate of the population field from sum(population) to avg(population):

a) Click the field to open the FIELD PROPERTIES menu.

b) Select Aggregates Average .

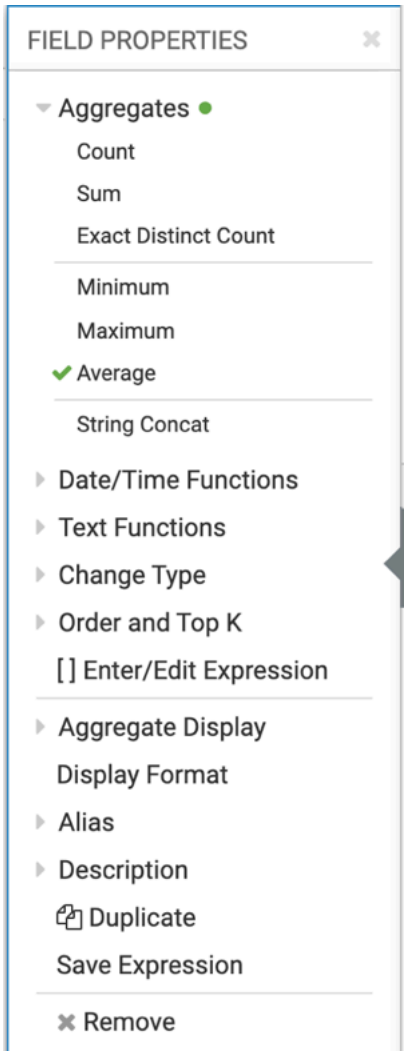

c) Click the x icon at the top of the FIELD PROPERTIES menu to close it.

d) Click REFRESH VISUAL.

The Areas visual appears.

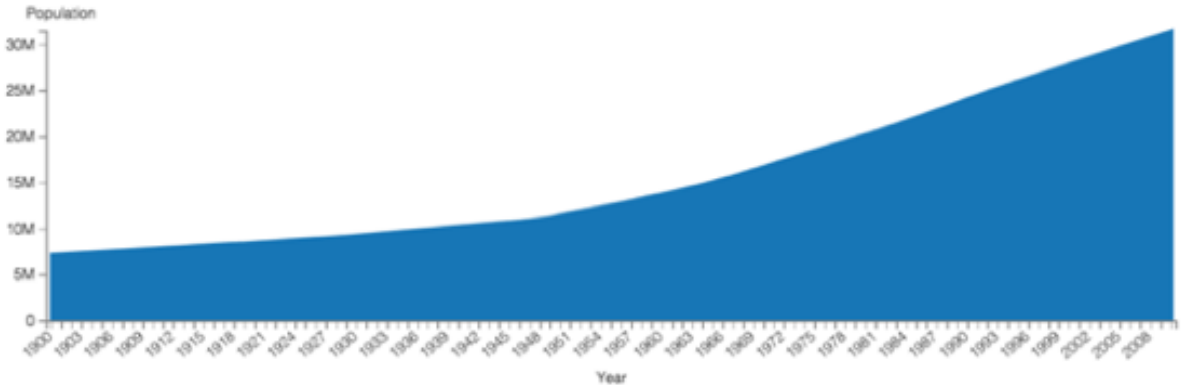

5. Alias the year field as Year, and the population field as Population:
   a) Click the field to open the FIELD PROPERTIES menu.
   b) Select Alias and enter the alias name of the column in the text box as it should appear in the visual.
   c) Click the x icon at the top of the FIELD PROPERTIES menu to close it.

   The shelf now shows the fields with their alias names.
6. To see the individual countries, select the country field from the list of Dimensions and drag it to the Colors shelf.
7. On the Tooltips shelf, add several fields from Dimensions and Measures.

   This enables you to see specific descriptive information in your visuals, such as input values, segment affiliation, and calculations.

   For example:

   a) From Measures, drag gdp_per_capita and life_expectancy to the Tooltip shelf.
   b) From Dimensions, drag country, un_subregion, and un_region to the Tooltip shelf.
   c) For all items on the Tooltip shelf, click the (down arrow) icon and select Aggregates Remove Aggregate .
   d) For all numeric items on the Y shelf and the Tooltip shelf, click the (down arrow) icon, select Display Format, and enter ,.2f.

      These fields are typically from the Measures menu. In this example include gdp_per_capita and life_expecta ncy.
8. On the Filters shelf, add several Dimensions and Measures from the DATA menu.

   This enables you to dynamically control the data input and discover the data at a more granular level.

   For example, from Dimensions, drag country, un_subregion, un_region, and year to the Filters shelf.
9. On the Filters shelf, click the (down arrow) icon on the un_region field.
10. Select Pick values from a list.
11. In the Pick values for un_region modal window, select Africa and click Save.

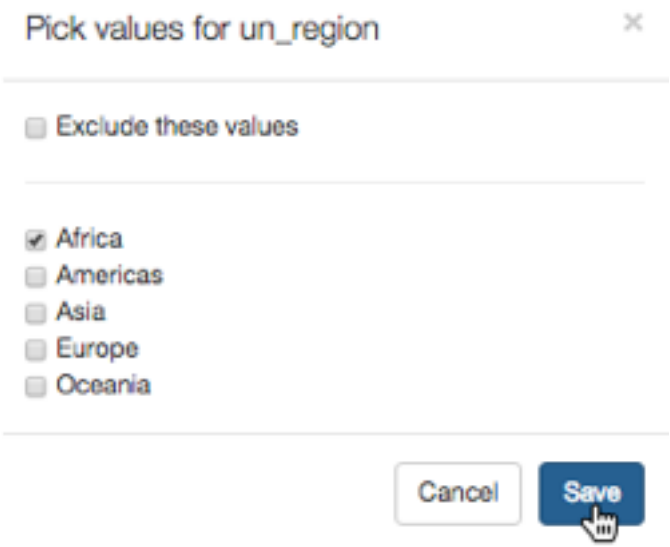

12. Click REFRESH VISUAL.

The updated Areas visual appears.

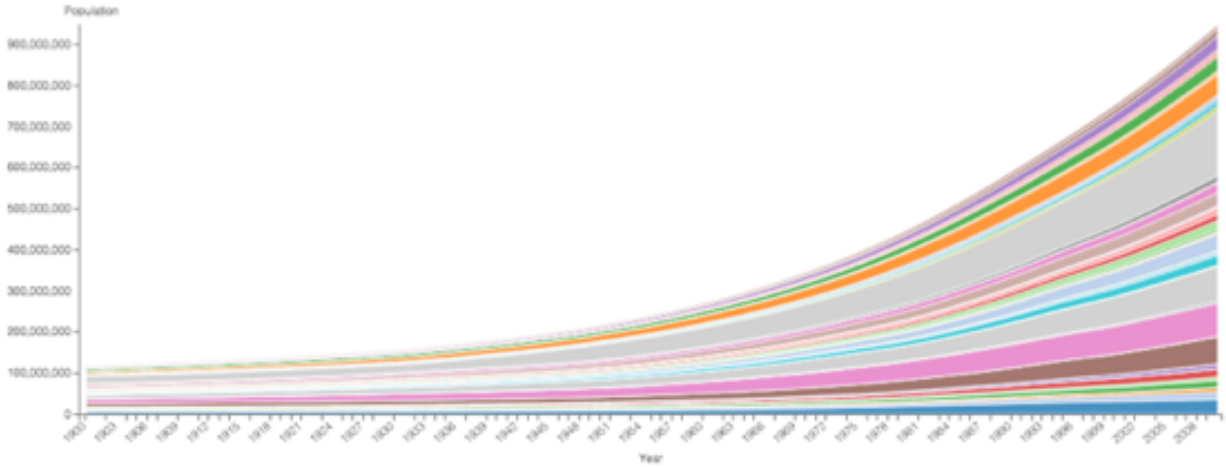

13. Click the (pencil) icon next to the title of the visual to edit it, and change the title to World Population - Area.

14. At the top left corner of the Dashboard Designer, click SAVE.

Related Information

Changing the min/max of the aggregate scale

Using the trellis option with areas

In Areas visuals, Cloudera Data Visualization enables you to create Trellis charts that show different partitions (segments) of the dataset alongside one other.

Before you begin

Create the World Population - Area visual as described in *Areas*.

About this task

This example uses the visual World Population - Area from *Areas* to create trellis charts.

The following steps demonstrate how to create a new trellised Area visual on the World Life Expectancy dataset.

Procedure

1. Open the World Population - Area visual in Edit mode.
2. Clone the visual as described in *Cloning a visual*.

3. Under Dimensions, select un_region and drag it to the X shelf. If it is not the first (left-most) field on the shelf, drag it to the left of the field Year.

   The first element on the shelf controls the category of the trellis.

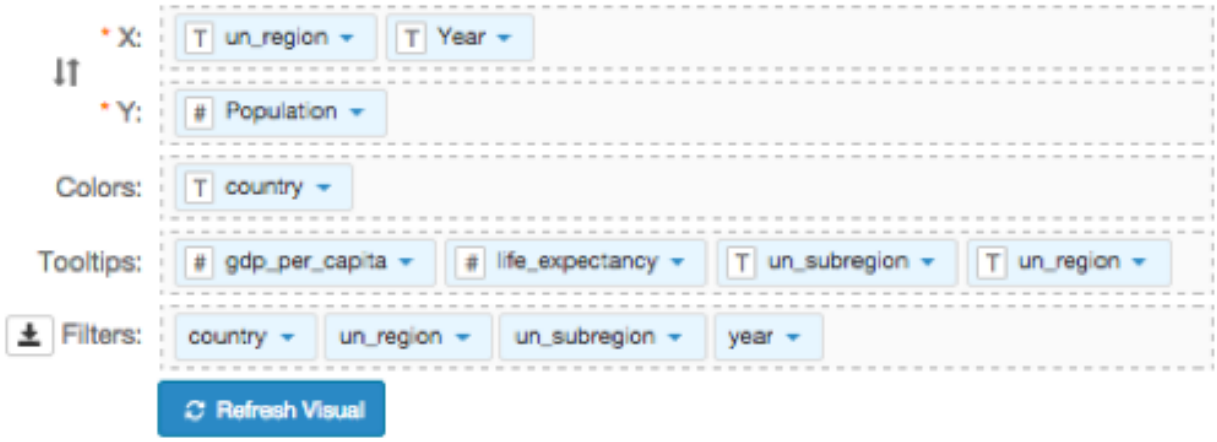

4. On the Filters shelf, clear any existing settings.
5. Click REFRESH VISUAL.

   The new trellised area chart appears.

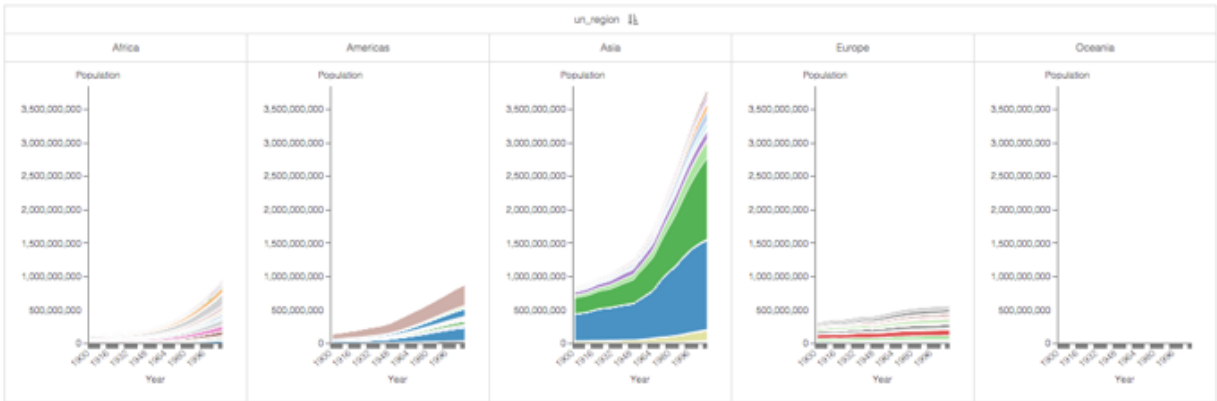

6. Click the (pencil) icon next to the title of the visual to edit it, and change the title to World Population - Area Trellis.
7. At the top left corner of the Dashboard Designer, click SAVE.

Using the trellis option with independent scale

Cloudera Data Visualization enables you to create a new trellised Areas visual with independent aggregate axes.

Before you begin

- Create the World Population - Area visual as described in *Areas*.
- Create the World Population - Area Trellis as described in *Enabling the trellis option*.

About this task

Trellis charts are very powerful visualization tools. However, they do not work well when one chart contains values that are several orders of magnitude larger than in the other charts. The charts that represent segments with smaller values are very difficult to interpret. You can mitigate this problem by enabling independent axis.

This example uses the visual World Population - Area Trellis from *Enabling the trellis option*, to create trellis charts with independent scale.

The following steps show you how to create a new trellised area chart with independent aggregate axes on the World Life Expectancy dataset.

Procedure

1. Open the World Population - Area Trellis visual in Edit mode.
2. Clone the visual as described in *Cloning a visual*.

   While the chart for Asia shows clear growth, the one of Oceania does not show any color. This is because by default, trellis charts are generated to the same scale. You should set Oceania to its own scale to be visible.

   Cloudera Data Visualization provides an override to this default setting. You have an option to set the trellis charts to their own axes by customizing the axes, and selecting Independent Aggregate Scale option, as described in *Customizing axes*.
3. Click REFRESH VISUAL.

   The updated trellised area chart appears.

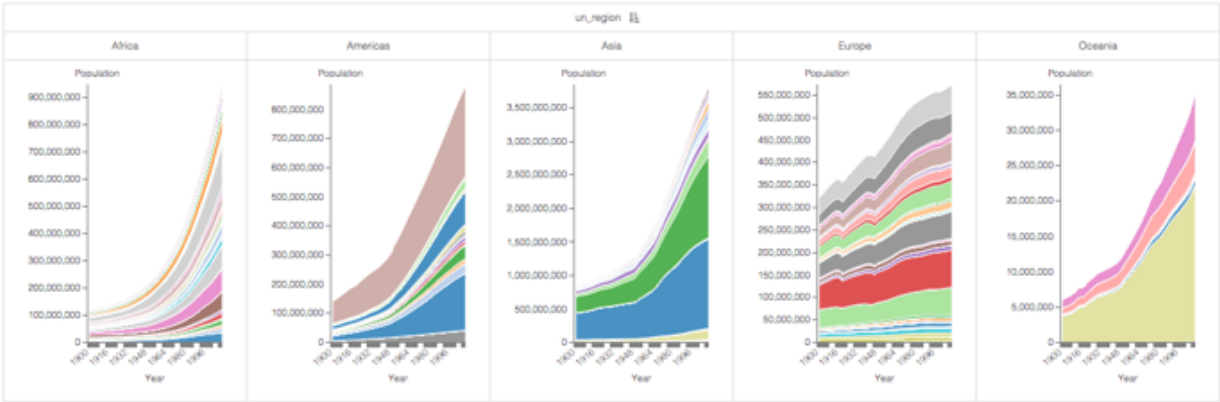

4. Click the (pencil) icon next to the title of the visualization to edit it, and change the title to World Population - Area Trellis, Independent Axes.
5. At the top left corner of the Dashboard Designer, click SAVE.

Shelves for areas

Overview of shelves for Cloudera Data Visualization Areas visuals.

An Areas visual has the following shelves:

X Axis

This is conceptually the X axis. Use a Dimension for a horizontaly-oriented area graph or a Measure for a vertically-oriented area graph.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in columns.

Y Axis

This is conceptually the Y axis. Use a Measure for a horizontally-oriented area graphor a Dimension for a vertically-oriented area graph.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in rows.

Colors

To stack areas by category, add a Dimension field to this shelf.

This shelf displays only one field on the graph. Additional fields appear only in the Tooltip.

Tooltips

Add Measure fields from the DATA menu for additional information in the graph's Tooltip.

This shelf accepts multiple fields.

Drill

Add Dimension fields from the DATA menu to this shelf.

When Drill Into context menu is enabled but no fields are specified, you can drill into any dimension. Specifying the fields on this shelf limits drilling to these dimensions.

This shelf accepts multiple fields.

Labels

Additional data labeling.

This shelf accepts at most one field.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Grouped bars

Cloudera Data Visualization enables you to create Grouped Bar visuals, which show clustered groups of related measurements as bars, enhancing comparisons among categories. While bar charts show comparisons among categories, grouped bar charts show clustered groups of related measurements as bars.

About this task

The following steps demonstrate how to create a Grouped Bars visual representation of ethnic distribution of population in some US counties. The example is based on the US Counties dataset. For more information, see *Adding data* and *Creating a dataset*.

For an overview of shelves that specify this visual, see *Shelves for grouped bars*.

Procedure

1. Create a new visual based on the US Counties dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Grouped Bars.

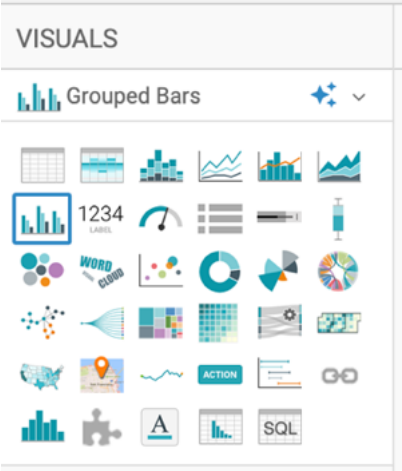

You can see that the shelves of the visual changed.

The mandatory shelves are X Axis and Y Axis. Fields placed on these shelves can be swapped to produce a horizontal chart.

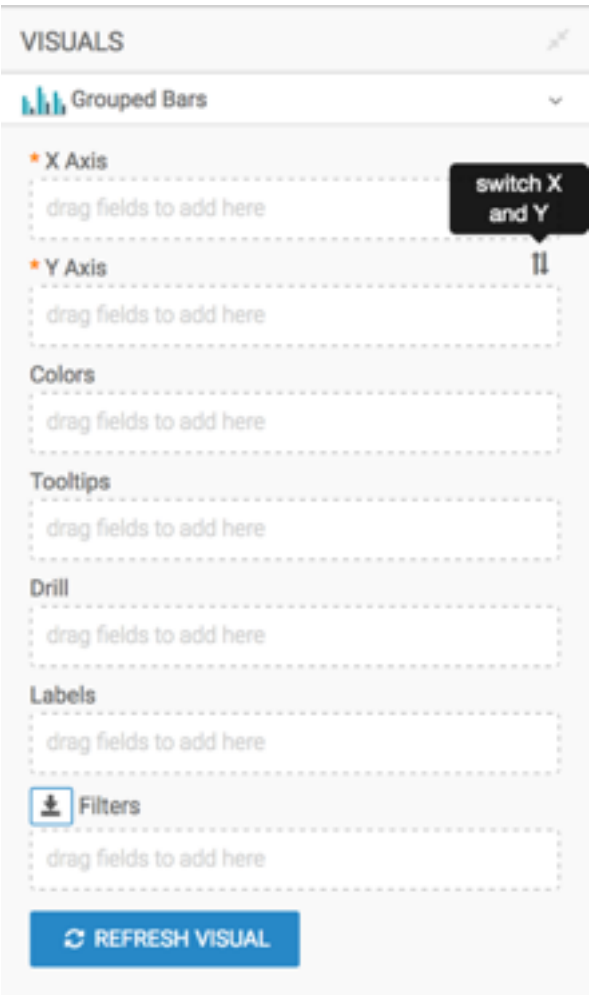

3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) Under Dimensions, select the Ctyname field and drag it to the X Axis shelf.

      At this point, it is recommended that you alias the field as County. For more information, see *Alias*.
   b) Select the Wa male field and drag it to the Y Axis shelf.

      This first measurement provides the axis definition.
   c) Add the other measurements to the Colors shelf.

      Together with the measurement on the Y Axis, this creates the 'group' Cloudera Data Visualization uses to draw grouped bars.

      For this example visual, add Ba Male, Ia Male, Aa Male, Na Male, and Tom Male.
   d) On the Y Axis shelf, click the field, and from the FIELD PROPERTIES menu, select [ ] Enter/Edit Expression.
   e) In the Enter/Edit Expression modal window, edit the expression to the following:

      ```
      sum([Wa Male]) + sum([Wa Female])
      ```

      Click VALIDATE EXPRESSION, and then click SAVE.

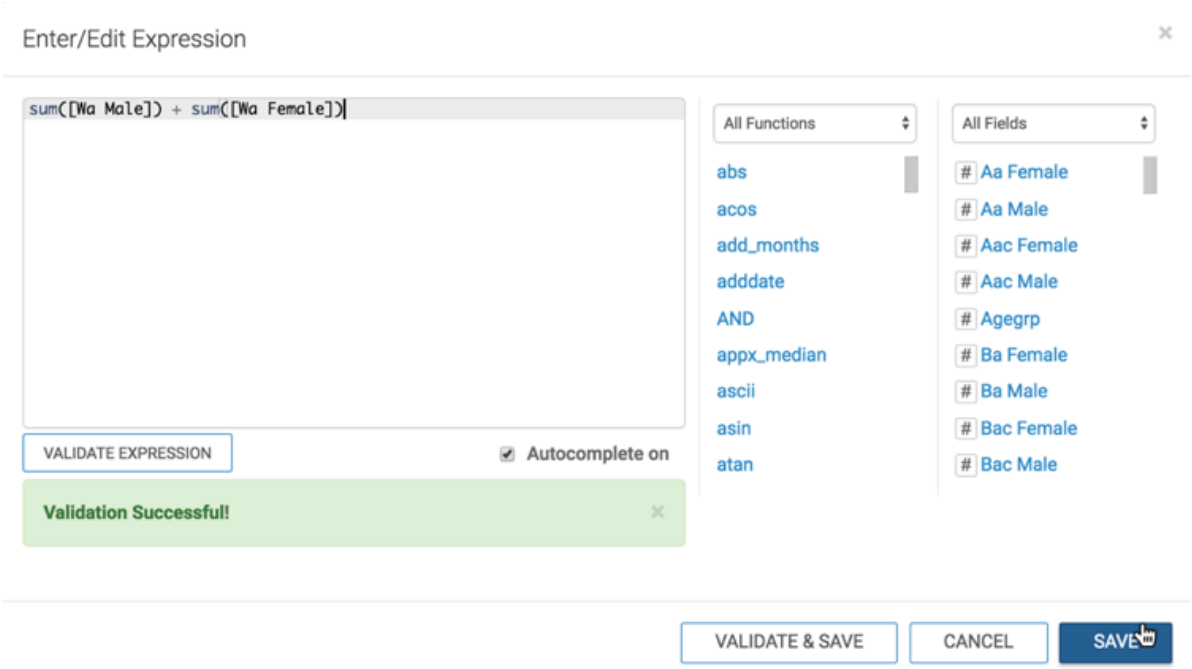

   f) Repeat the change in the expression on all the fields placed on the color shelves.

      ```
      sum([Ba Male]) + sum([Ba Female])
      sum([Ia Male]) + sum([Ia Female])
      sum([Aa Male]) + sum([Aa Female])
      sum([Na Male]) + sum([Na Female])
      ```

```
sum([Tom Male]) + sum([Tom Female])
```

g) Alias the field on the Y Axis shelf as 'White', and alias the fields on the Colors shelf as 'Black or African American', 'American Indian and Alaska Native', 'Asian American', 'Native Hawaiian and Other Pacific Islander', and 'Two or More Races'.

h) Add the field Stname to the Filters shelf and select the value 'Alaska'.

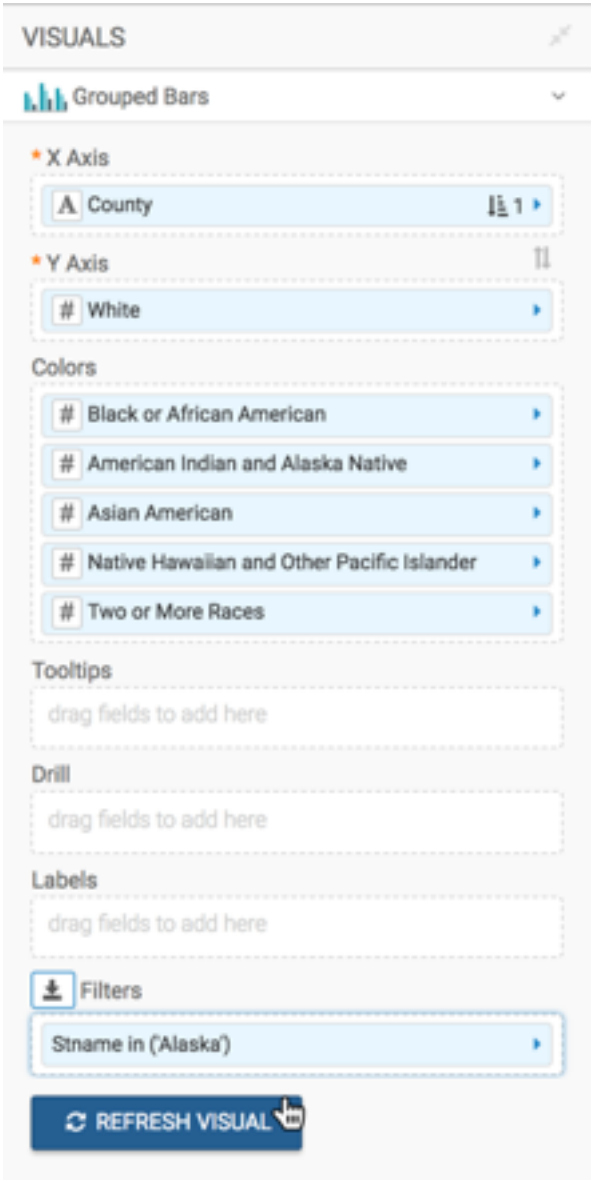

4. Click REFRESH VISUAL to generate the Grouped Bars visual.

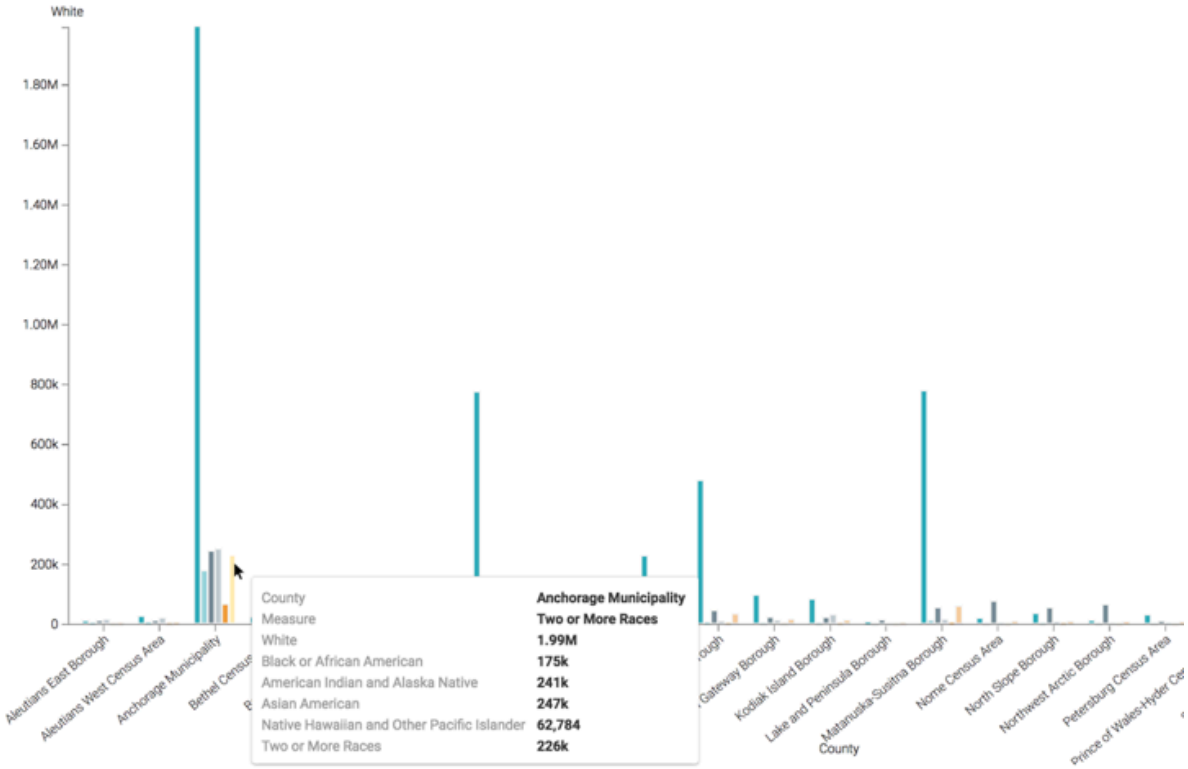

5. Click the (pencil) icon next to the title of the visualization to edit it, and change the title to World Population - G rouped Bars.
6. At the top left corner of the Dashboard Designer, click SAVE.

Shelves for grouped bars

Overview of shelves for Cloudera Data Visualization Grouped Bars visuals.

A Grouped Bars visual has the following shelves:

X Axis

This is conceptually the X axis. Use a Dimension for horizontal bars or a Measure for vertical bars.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in columns.

Y Axis

This is conceptually the Y axis. Use a Measure for horizontal bars or a Dimension for vertical bars.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in rows.

Colors

To stack bars by category, add a Dimension field to this shelf.

This shelf displays only one field on the graph. Additional fields appear only in the Tooltip.

Tooltips

Add Measure fields from the DATA menu for additional information in the graph's Tooltip.

This shelf accepts multiple fields.

Drill

Add Dimension fields from the DATA menu to this shelf.

When Drill Into context menu is enabled but no fields are specified, you can drill into any dimension. Specifying the fields on this shelf limits drilling to these dimensions.

This shelf accepts multiple fields.

Labels

Additional data labeling.

This shelf accepts at most one field.

Filters

Add Dimension or Measure fields from the Data menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

KPI

Cloudera Data Visualization enables you to create Key Performance Indicator (KPI) visuals. KPI visuals return values to emphasize particular areas of dashboards or applications. They can be used to provide section titles, totals for segmented data, draw attention to aggregation results, and so on.

About this task

The following steps demonstrate how to create a new KPI visual representation on the Customer Value Analysis dataset. This dataset is based on sample data previously imported into Cloudera Data Visualization.

For information on customization options, see *KPI settings*. For an overview of shelves that specify this visual, see *Shelves for KPI visuals*.

Procedure

1. Start a new visual based on the Customer Value Analysis dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click KPI.

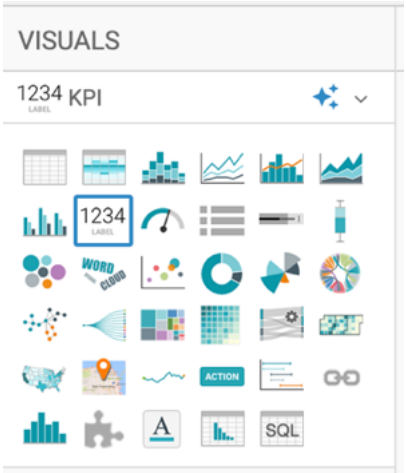

The shelves of the visual changed. There are no mandatory shelves for KPI visuals. Instead, the primary measure appears either on the Dimension or the Measure shelves.

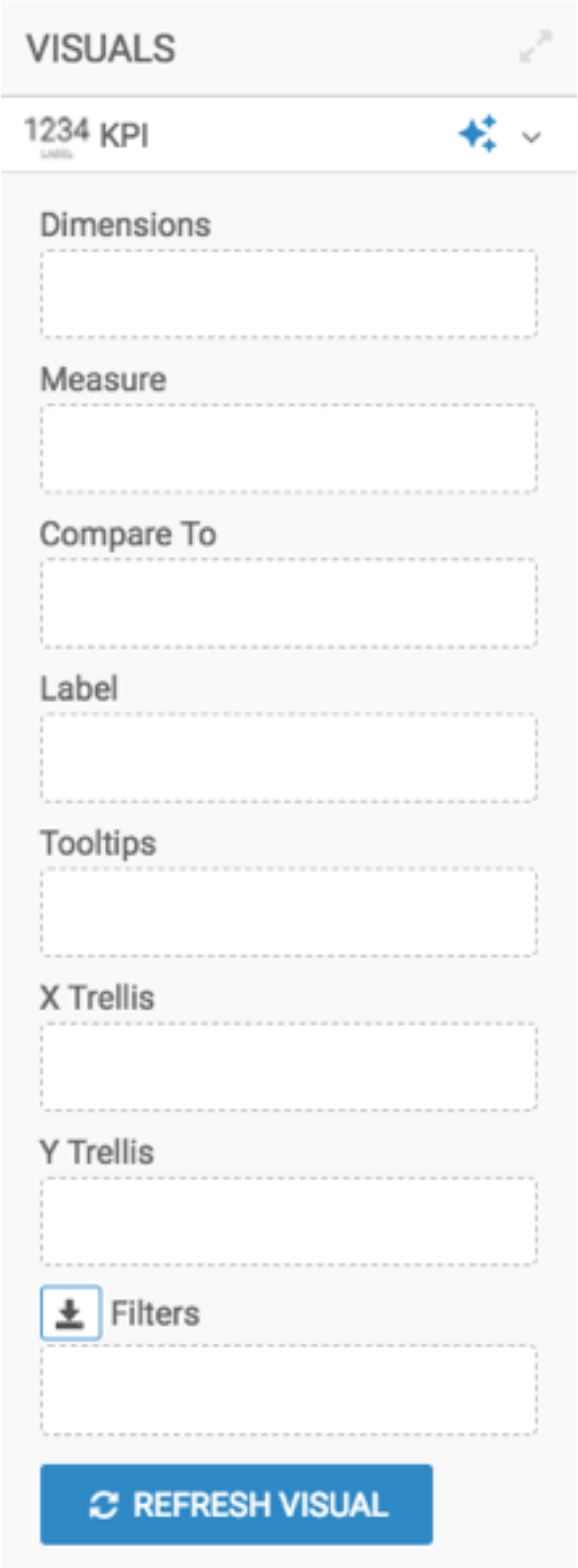
VISUALS
1234 KPI
Dimensions
Measure
Compare To
Label
Tooltips
X Trellis
Y Trellis
Filters
REFRESH VISUAL

3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) From Measures, select Customer Lifetime Value and place it on the Measure shelf. Ensure that the aggregate is set to Average.
   b) From Measures, select Total Claim Amount and place it on the Compare To shelf. Ensure that the aggregate is set to Average.
   c) From Measures, select Record Count and place it on the Label shelf.
   d) Change the expression on the shelf to the following calculation:

      ```
      concat("Records: ", cast(sum(1) as string)) as 'Record Count'
      ```

      The exact expression may be different, depending on the target data repository.
   e) Place the fields Gender and Marital Status (both under Dimensions) on the X Trellis and Y Trellis shelves.

4. Click REFRESH VISUAL.

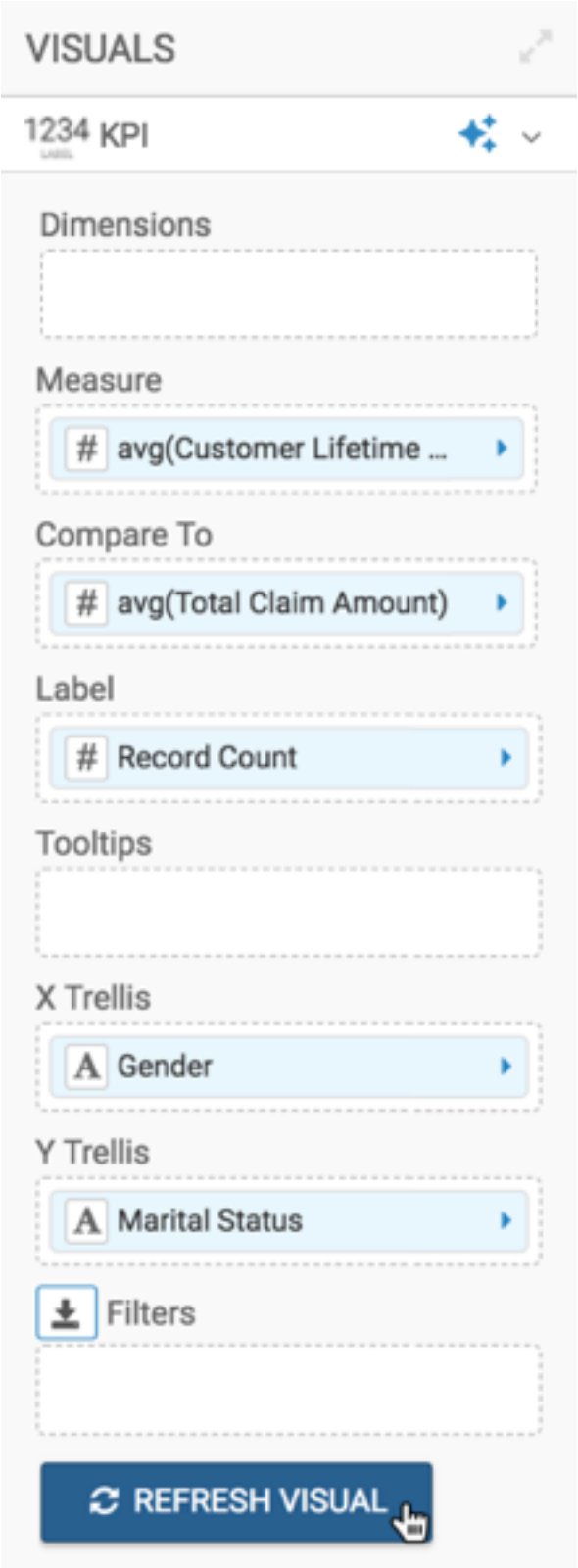

The KPI visual appears.

	Gender	
Marital Status	F	M
Divorced	8,309.09 398.54 Records: 709	8,168.35 409.25 Records: 660
Married	8,094.65 367.46 Records: 2779	8,061.67 403.33 Records: 2519
Single	7,972.48 529.36 Records: 1170	7,482.42 582.71 Records: 1297

5. Click the (pencil) icon next to the title of the visualization to edit it, and change the title to KPI by Gender and Marital Status.
6. At the top left corner of the Dashboard Designer, click SAVE.

Shelves for KPIs

Overview of shelves for Cloudera Data Visualization KPI visuals.

A KPI visual has the following shelves:

Dimensions

Add a Dimension field from the DATA menu to display a specific dimension value.

This shelf accepts multiple fields, but the visual displays only the first dimension or measure. Some additional information about the shelf:

- If something is placed on this shelf while the Measure shelf is blank, the visual displays the first value in the dimension as the main indicator.
- If you order this shelf, it displays the first value based on the order parameters.
- If you want to display a dimension sorted by another dimension, place a second dimension on the shelf and sort by it. The visual displays only the first dimension value on the dimension shelf, but the sorted specification on the second dimension determines the order.
- If you want to display a measure value sorted by a dimension (or a dimension sorted by a measure), place the fields on the matching shelves, and sort according to your business case.

For more information, see *Displaying first measure as main indicator*.

Measure

Add a Measure field from the DATA menu to display an aggregate.

This shelf accepts a single field.

The measure shelf either provides the main indicator, or it can be used for sorting. If a measure is used, the display format on this shelf dictates the appearance of the main indicator.

Compare To

Add a Measure field from the DATA menu to show a comparison aggregate as a target value.

This shelf accepts a single field.

If the Dimensions/Measure and Compare To shelves hold numeric values, this shelf specifies both the small value to the right side of the main measure (default placement), and an arrow.

Some additional information about the arrow:

- If the primary measure indicator is higher than the Compare To value, the arrow points up and it has the color specified for the up arrow in the color settings of the visual.
- If the compare value is higher, than the arrow points down, in the color specified for the down arrow in the color settings of the visual.
- If the two values are identical, the arrow is 'flat'. It means that it is horizontal and points to the right.

It is possible to switch the comparison direction by *flipping the comparison* setting.

Label

Add one field to this optional shelf to provide a label under the main indicator (default placement). You can *change label position*.

This shelf accepts 0 or 1 fields.

Tooltips

Add Measure fields from the DATA menu for additional information in the graph's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use fields from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use fields from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Gauge

In Cloudera Data Visualization, Gauge visuals are powerful tools for communicating performance against set targets, projected trends, and a qualitative assessment of the measurement.

About this task

Gauges show minimum and maximum values on an evenly distributed scale, and qualitative ranges for classifying the primary measurement as different colors, which is very useful for visually distinguishing them from one another.

Gauges are semi-circular by default, and include many setting adjustments.

You can review the following three examples of gauge charts to see their structure:

Example 1: Measure < Projection < Compare To

In this scenario, both the measurement and its projected future value are less than the target value.

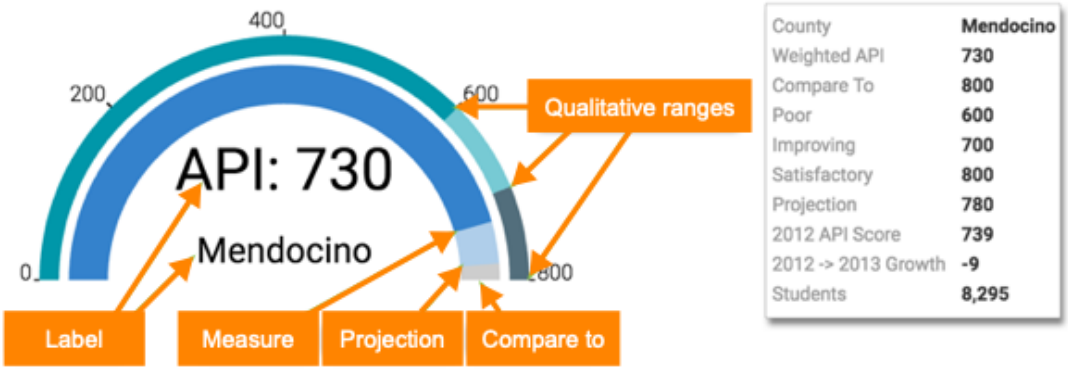

Example 2: Measure < Compare To < Projection

In this scenario, the measurement is less than the target value, but its projection is expected to exceed the target.

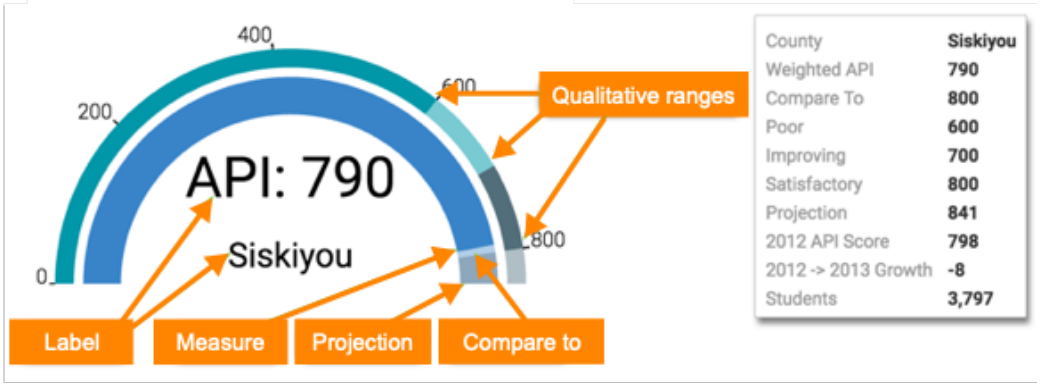

Example 3: Compare To < Measure < Projection

In this scenario, the measurement has already exceeded the target value, and the projection is even greater.

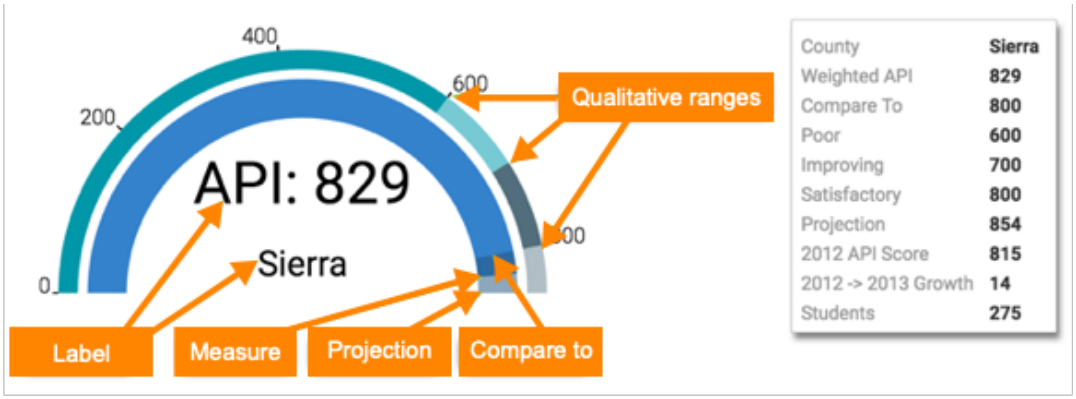

- The long dark inner arc of the visual represents the value on the Measure shelf. It is the feature measurement of the visual. In Example 3, the last section appears in dark blue because the value of Measure is greater than the Compare To value.
- The extension of the same arc is the expected value of the measure at a future date, supplied by the Projection shelf of the visual. The overlay of colors and their opacity leads to different effects depending on the relative values of Measure, Projection, and Compare To shelves:
  - In Example 1, this appears in pale blue and does not terminate the arc. This is because the value on the Projection shelf is less than the Compare To value.
  - In Example 2, Projection appears in darker grey as extension beyond the lower Compare To value (in pale blue).
  - In Example 3, Projection again appears in darker grey, but this time as an extension beyond the darker blue arc that shows by how much the Measure exceeds the lower Compare To value.
- The Compare To shelf shows the target value for the measurement. In these examples, it has a value of 800 for all gauges.
- The secondary outer arc shows the Qualitative Ranges that form the upper limits of the 'goodness' for the current measurement. From the origin to the upper value, they specify relative levels of performance success. You may define anywhere from 0 to 4 qualitative ranges.
- The Label shelf overrides default labeling taken from the Measure shelf. This is an opportunity to clarify what is being measured, as well as the units of measurement. This shelf accepts a maximum of two values.

Like other Cloudera Data Visualization visuals, the Gauge visual may be trellised using X and Y shelves, have additional fields on the Tooltips shelf, and use Filters. For more information, see *Shelves for gauge visuals*.

We are working with the District Performance dataset, built on a datafile that contains the California School District APIs from 2012 and 2013. The range of possible scores varies from a low of 200 to a high of 1000. However, the statewide API performance target is 800.

Procedure

1. Start a new visual based on dataset District Performance.

   For more information, see *Adding data*, *Creating datasets*, and *Creating a visual*.

2. In the visuals menu, find and click Gauge.

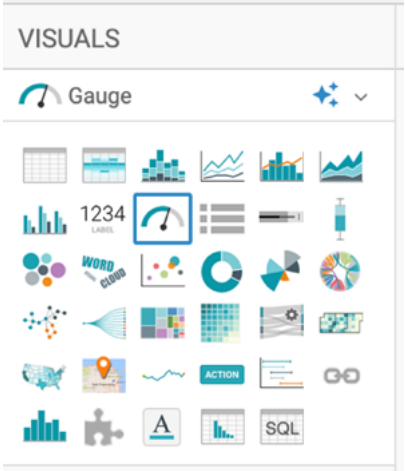

The shelves of the visual changed. The Measures shelf is mandatory.

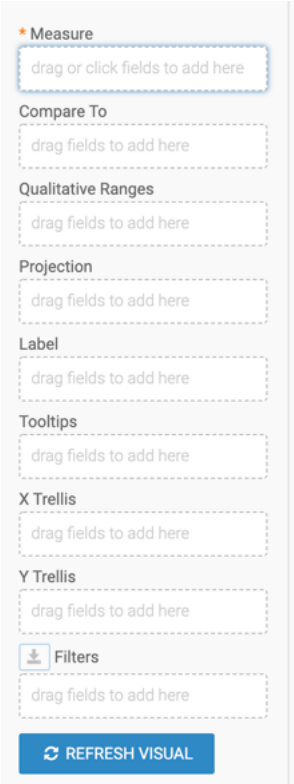

3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) For the Measure shelf, add the field Api13.
   b) Customize the expression to calculated the waited average of the field, based on the number of students tested in each district (field Valid).
      1. Click the (right-arrow) icon to the left of the field, and select the [ ] Enter/Edit Expression option.
      2. In the Enter/Edit Expression modal window, change the expression to the following formula:

         ```
         floor(sum([Api13]*[Valid])/sum([Valid]))
         ```

      3. Click VALIDATE & SAVE to ensure that everything works.
   c) For the Compare To shelf, add the field Api13.
   d) Customize the expression to change it to the state-wide target of API=800.
      1. Click the (right-arrow) icon to the left of the field, and select the [ ] Enter/Edit Expression option.
      2. In the Enter/Edit Expression modal window, change the expression to the scalar value of 800.
      3. Click VALIDATE & SAVE to ensure that everything works.
   e) For the Qualitative Ranges shelf, add the field Api13.
   f) Specify the equation that returns the upper limit of the first qualitative range. For this visual, the example uses simple scalar values.
      1. Click the (right-arrow) icon to the left of the field, and select the [ ] Enter/Edit Expression option.
      2. In the Enter/Edit Expression modal window, change the expression to the scalar value of 600.
      3. Click VALIDATE & SAVE to ensure that everything works.
   g) To create other ranges:
      1. Click the (right-arrow) icon to the left of the field, and select the Duplicate option.
      2. Open the Enter/Edit Expression modal window for the duplicate and change the value.
      3. Click VALIDATE & SAVE.

      In this example, we defined three ranges: at 600, 700. and 800, and we aliased the qualitative ranges as Poor, Improving, and Satisfactory, in that order.

      An alternative to specifying qualitative ranges through the shelf is to calculate them as percentages of the Compare To field. To do that, follow the instructions in.*Specifying qualitative range as percentages*.
   h) For the Projection shelf, add the field Api13.
   i) Specify the equation that projects the current measurement to some future date by clicking the (right-arrow) icon to the left of the field, selecting the [ ] Enter/Edit Expression option, and changing the expression to the following SQL clause:

      ```
      if(  sum([Api13]*[Valid])/sum([Valid])>800,
           sum([Api13]*[Valid])/sum([Valid])+25,
      ```

```
      sum([Api13]*[Valid])/sum([Valid]) + 50)
```

j) For the Label shelf, add the field Api13, and convert the expression to the following SQL clause:

```
concat("API: ", cast(floor(sum([Api13]*[Valid])/sum([Valid])) as str
ing))
```

You can add the field Cname under the first field and change its aggregation to max([Cname]).

k) For the Tooltips shelf, add more fields here to provide additional information.

In this example, the following tooltips are used:

- floor(sum([Api12]*[Valid])/sum([Valid]))
- floor(sum([Api13]*[Valid])/sum([Valid])) - floor(sum([Api12]*[Valid])/sum([Valid]))
- sum([Valid])

It is good practice to apply an alias to these fields. In this example, the following aliases are used: 2012 API Score, 2012 -> 2013 Growth, and Students.

l) For the Filters shelf, add the filters youo need.

In this example the following filters are used: Cname for name of the County, selecting the values Alameda, Kings, Marin, Mendocino, Sierra, and Siskiyou.

m) For the Y shelf, add the field Cname (county name) to trellis the visual, and lias it as County.

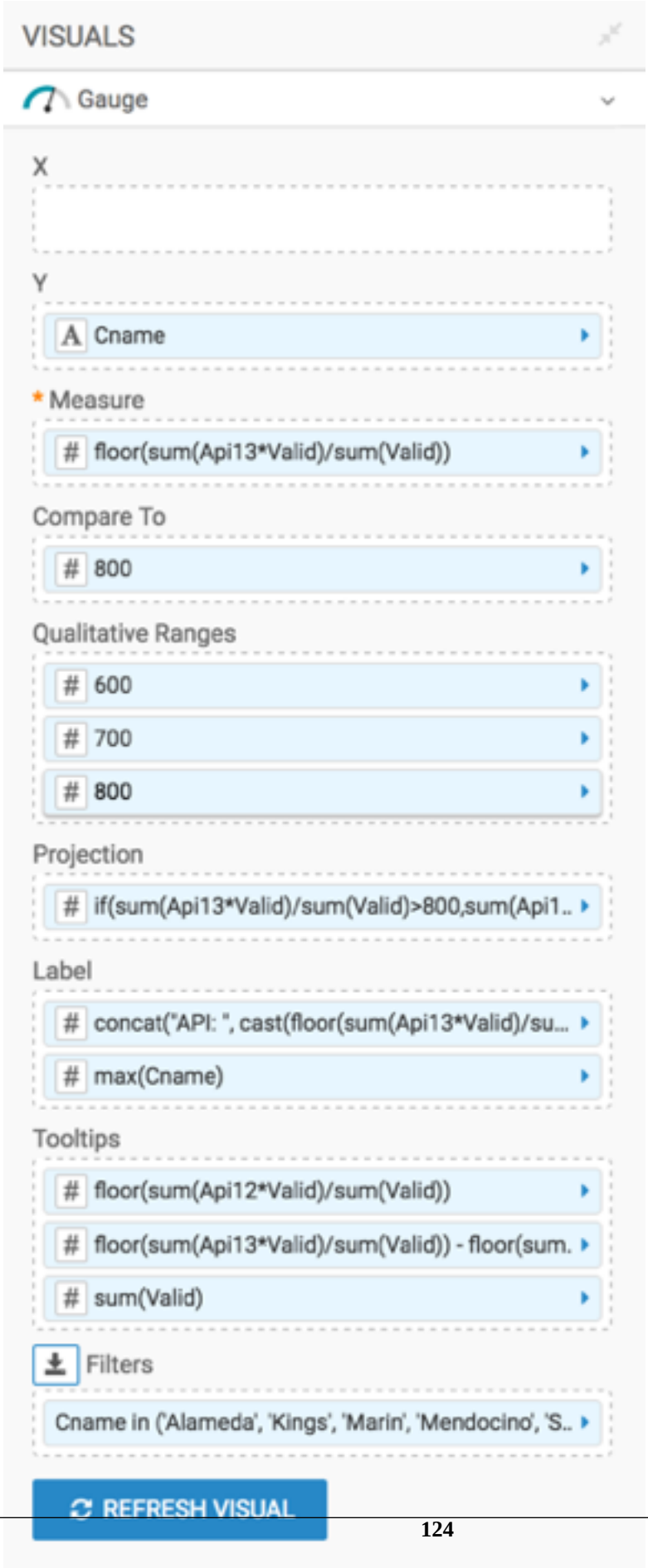
VISUALS
Gauge
X
Y
Cname
* Measure
floor(sum(Api13*Valid)/sum(Valid))
Compare To
800
Qualitative Ranges
600
700
800
Projection
if(sum(Api13*Valid)/sum(Valid)>800,sum(Api1..
Label
concat("API: ", cast(floor(sum(Api13*Valid)/su...
max(Cname)
Tooltips
floor(sum(Api12*Valid)/sum(Valid))
floor(sum(Api13*Valid)/sum(Valid)) - floor(sum.
sum(Valid)
Filters
Cname in ('Alameda', 'Kings', 'Marin', 'Mendocino', 'S..
REFRESH VISUAL

4. Click the Settings menu, select Trellis, and de-select the Show trellis borders option.
5. Click the Settings menu, select Marks, and select the Include a gauge line option. For more information, see *Displaying the gauge needle*.
6. Click the Colors menu, and select different options for Foreground Color, Pointer Color, and Background Color from the color palette menu. You may change the Projection Opacity and Background Opacity options too. In this example, they haven't been changed.
7. Alias the fields. In this example, the names of the shelves are used for each alias.

8. Click REFRESH VISUAL.

The new Gauge visual appears.

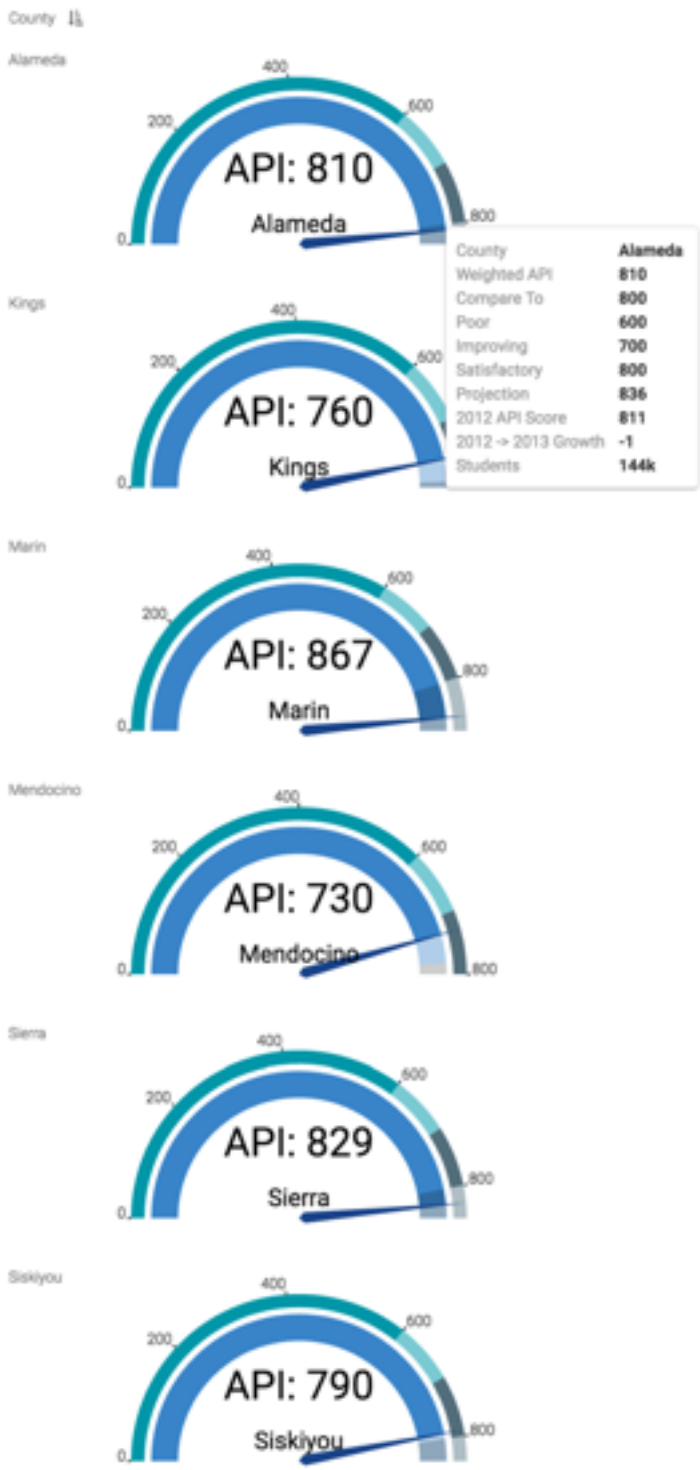

9. Change the title to District Performance, by County - Gauge.

- Click (pencil icon) next to the title of the visualization to edit it, and enter the new name.
- You can click (pencil icon) below the title of the visualization to add a brief description of the visual.

10. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

To adjust the gauge display, check all the available settings for this visual.

Shelves for gauges

Overview of shelves for Cloudera Data Visualization Gauge visuals.

A Gauge visual has the following shelves:

Measure

Add a Measure field from the Data menu to display an aggregate.

This is the feature measurement, such as the current value.

This shelf accepts a single field.

Compare To

Add a Measure field from the Data menu to show a comparison aggregate as a target value.

This shelf accepts a single field.

Qualitative Ranges

Add Measure fields from the Data menu to specify a range of values, such as poor, satisfactory, good, and excellent.

This shelf accepts multiple fields.

Projection

Add a Measure field from the Data menu to show a projected value based on current trends.

This shelf accepts a single field.

Label

Add as many as two Measure or Dimension fields from the Data menu to display label(s) for each graph. This overrides the default value provided by the field on the Measure shelf.

This shelf accepts 0 to 2 fields.

Tooltips

Add Measure fields from the Data menu for additional information in the graph's Tooltip.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the Data menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

X Trellis

Use a Dimension field from the Data menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the Data menu to trellis the visual into rows.

This shelf accepts multiple fields.

Row listing

Cloudera Data Visualization allows you to create Row Listing visuals. Row Listing is useful when you want to display raw data in a structured format.

About this task

The following steps demonstrate how to create a Row Listing visual based on the Cereals dataset. For an overview of the shelves that specify this visual, see *Shelves for row listing*.

Procedure

1. Start a new visual based on the Cereals dataset.

   By default, a table visual is displayed that shows all fields in the dataset.

   For instructions, see *Creating a visual*.
2. In the VISUALS menu, find and click the Row Listing icon.

   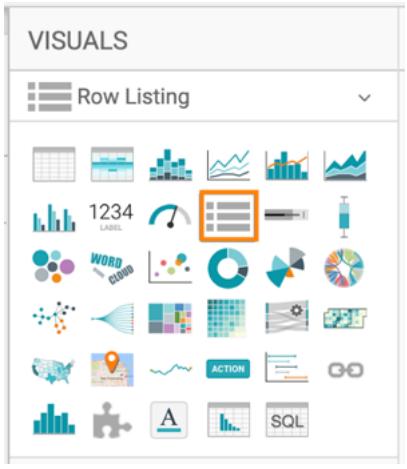

   The shelves of the visual change.
3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.

   Make the following selections to recreate the example Row Listing visual:

   - Under Dimensions, select ‘cereal_name’ and ‘manufacturer’ and add them to the Dimensions shelf.
   - Under Measures, select ‘calories’ and add it to the Measures shelf.

   For further information about the shelves of the visual, see *Shelves for row listing*.
4. Click REFRESH VISUAL.

   The updated Row Listing visual appears.
5. At the top left corner of the Visual Builder, click SAVE.

Example

In this example, you can see a list of different cereals together with their manufacturer and calorie information.

Cereals by manufacturer - Calories

enter subtitle...

100%_Bran

Nabisco

70

100%_Natural_Bran

Quaker Oats

120

All-Bran

Kelloggs

70

All-Bran_with_Extra_Fiber

Kelloggs

50

Almond_Delight

Ralston Purina

110

Apple_Cinnamon_Cheerios

General Mills

110

Apple_Jacks

Kelloggs

110

Basic_4

General Mills

130

Bran_Chex

Ralston Purina

90

Bran_Flakes

Post

Tip: If no default display format is specified, comma separation will be applied. See, Customizing display format.

Related Information

Creating a visual

Customizing display format

Shelves for row listing visuals

Cloudera Data Visualization Row Listing visuals have the following shelves.

Dimensions

This shelf is required and it accepts multiple fields.

You can add a dimension field from the DATA menu to display a specific dimension value. The visual displays all the added dimensions. If several fields are added, the visual displays the first value in the dimension as the main indicator. If you want to display a dimension sorted by another dimension, place a second dimension on the shelf and sort by it.

Measures

This shelf is optional and it accepts multiple fields.

You can add a measure field from the DATA menu to display an aggregate.

The measure shelf either provides the main indicator, or it can be used for sorting. To display a measure value sorted by a dimension (or a dimension sorted by a measure), place the fields on the matching shelves, and sort according to your business case.

Tooltips

This shelf is optional and it accepts multiple fields.

You can add measure fields from the DATA menu to provide additional information in the visual's tooltip.

Filters

This shelf is optional and it accepts multiple fields.

You can add dimension or measure fields from the DATA menu to include or exclude categorical or value criteria.

Bullet

Cloudera Data Visualization enables you to create Bullet visuals.

About this task

Bullet visuals are a variation of bar charts, customized for showing progress towards a predefined goal. They are developed specifically for financial data, but may be used in many other situations. They are an alternative to gauges.

This is an example of a Bullet visual for a high schools. It illustrates the various elements of the bullet chart.

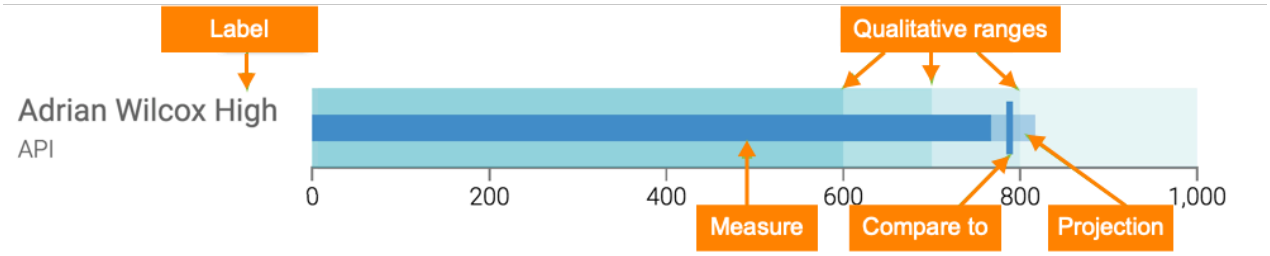

- The dark bar in the middle of the visual represents the value on the Measure shelf. It is the feature measurement of the visual.
- The lighter-shaded extension of the same bar is the expected value of the measure at a future date. This value is on the Projection shelf of the visual.
- The vertical bar is the target value of the measurement. Place this calculation on the Compare To shelf.
- Calculations on the Qualitative Ranges shelf determine the upper limits of the 'goodness' for the current measurement. From the origin to the upper value, they specify poor, better, good, and excellent performance. You may define anywhere from 0 to 4 qualitative ranges.
- The Label shelf overrides default labeling taken from the Measure shelf. This is an opportunity to clarify what is being measured, as well as the units of measurement. This shelf accepts a maximum of two values.

Just as other Cloudera Data Visualization visuals, the Bullet visual may be trellised using X and Y shelves, have additional fields on the Tooltips shelf, and you can use Filters with it. It also includes additional setting adjustments. For more information, see *Shelves for bullet visuals*.

The following example uses the 2012 and 2013 API Scores for California Schools. The School Performance dataset is used, which was built on a datafile that contains the California School APIs from 2012 and 2013. The range of possible scores varies from a low of 200 to a high of 1000. The statewide API performance target for all schools is 800. However, each school that has not achieved that target may have a more realistic score set for it by the Department of Eduction, based on a number of factors we will not discuss here.

Procedure

1. Start a new visual based on the School Performance dataset.

   For instructions, see *Adding data*, *Creating datasets*, and *Creating a visual*.

2. In the VISUALS menu, find and click Bullet.

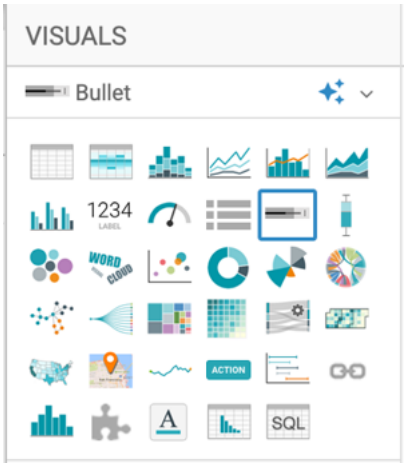

The shelves of the visual changed. They are now Measure (mandatory), Compare To, Qualitative Ranges, Projection, Label, Tooltips, X Trellis, Y Trellis, and Filters.

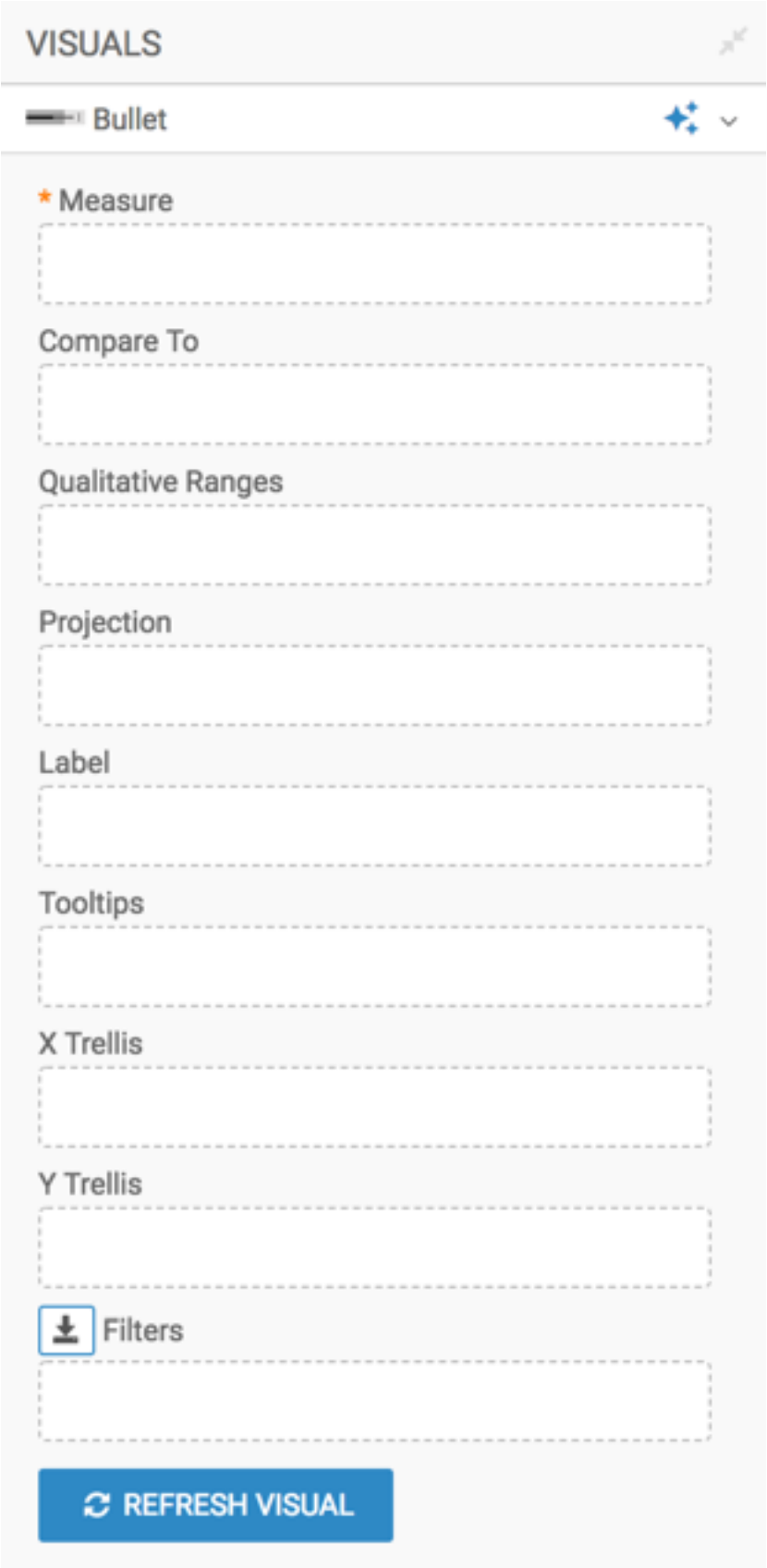
VISUALS
Bullet
* Measure
Compare To
Qualitative Ranges
Projection
Label
Tooltips
X Trellis
Y Trellis
Filters
REFRESH VISUAL

3. Populate the shelves from the available fields (in Dimensions and Measures) in the DATA menu.
   a) Measure - Add the field Api13, and change the aggregation, so that the shelf contains max([Api13]).

      [Optional] Limit the output to the top 10 performers:

      1. Click the (right-arrow) icon to the left of the field, and select the Order and Top K option.
      2. After the menu expands, enter 10 in the Top K option.

      [Optional] Alias the field as the name of the measurement. In this case, use API.

      Note: The Measure shelf supports only positive values.

   b) Compare To - Add the field Api12. Then customize the expression to reflect whether the school reached the state-wide target of API=800.

      1. Click the (right-arrow) icon to the left of the field, and select the [ ] Enter/Edit Expression option.
      2. In the Enter/Edit Expression modal window, change the expression to the following formula:

         ```
         Max(if([Api12] < 800, [Api12] + [Target], 800))
         ```

      3. Click VALIDATE EXPRESSION to ensure that everything works.
      4. Click SAVE.

      [Optional] Alias the field as Compare To.

   c) Qualitative Ranges - Add the field Api12. Then, specify the equation that returns the upper limit of the first qualitative range. For this visual, simple scalar values are used.

      1. Click the (right-arrow) icon to the left of the field, and select the [ ] Enter/Edit Expression option.
      2. In the Enter/Edit Expression modal window, change the expression to the scalar value 600, and click SAVE.
      3. To create other ranges, simply click the (right-arrow) icon to the left of this field, and select the Duplicate option. Then open the Enter/Edit Expression modal window for the duplicate, change the value, and click SAVE.

         In this example, four ranges are defined: at 600, 700. 800, and 1000.
      4. [Optional] Alias the ranges as Range 1, Range 2, Range 3, and Range 4.

   d) Projection - Add the field Api13. Then, specify the equation that projects the current measurement to some future date.

      1. Click the (right-arrow) icon to the left of the field, and select the [ ] Enter/Edit Expression option in the FIELD PROPERTIES menu.
      2. In the Enter/Edit Expression modal window, change the expression to the following formula and click SAVE:

         ```
         if(max([Api13]) > 800, max([Api13]), max([Api13]) + 2 * max([Target]
         ))
         ```

      3. [Optional] Alias the field as Projection.

   e) Label - Add the field Api13, and convert the aggregation to maximum, for max([Api13]).

      [Optional] Duplicate the field, and change its definition to max('API'), using the Enter/Edit Expression option in the FIELD PROPERTIES menu.. Move this field over the other field.

   f) Tooltips - Add more fields here, such as Api12, Target, Dname, Sname, and so on. You can apply an Alias to these fields.

   g) Y Trellis - Add the field Sname, and Alias it as School.

   h) Filters - Add the filters here. Cname is used in this example for name of the County (selecting Santa Clara), and Stype for school type (choosing H for high school).

   i) Y shelf - Ad the field Sname (school name) to trellis the visual.

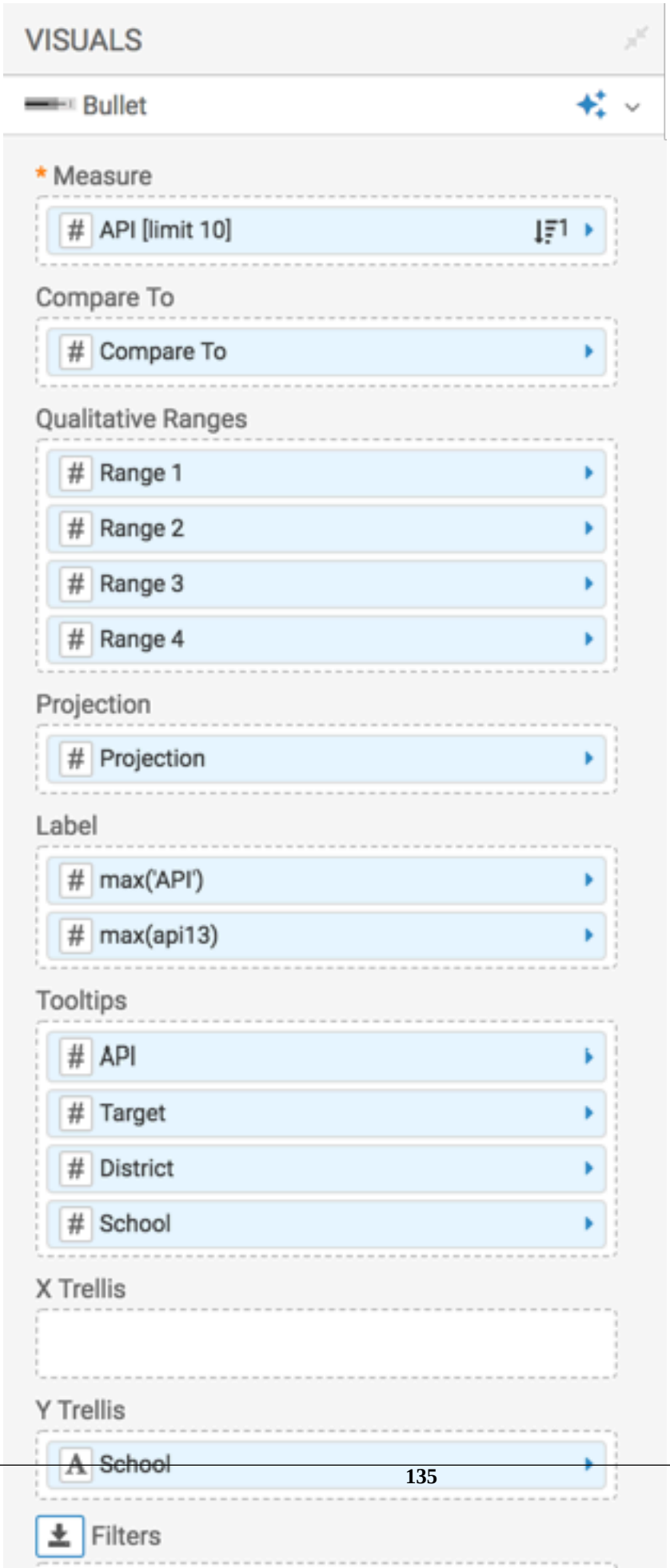
VISUALS
Bullet
* Measure
API [limit 10]
Compare To
Compare To
Qualitative Ranges
Range 1
Range 2
Range 3
Range 4
Projection
Projection
Label
max('API')
max(api13)
Tooltips
API
Target
District
School
X Trellis
Y Trellis
School
Filters

4. [Optional] Click the Settings menu, select Trellis, and de-select the Show trellis borders option.
5. [Optional] Click the Colors menu, and select different options for Background Color and Foreground Color from the color palette menu. You may change the Projection Opacity too.

6. Click REFRESH VISUAL.

   The new Bullet visual appears. Note that it is trellised, which is why it is sorted on the Sname field (school name), not on the measure Api13.

   Additionally, the Projection value is not visible; this is because it is relatively low for high-performing schools.

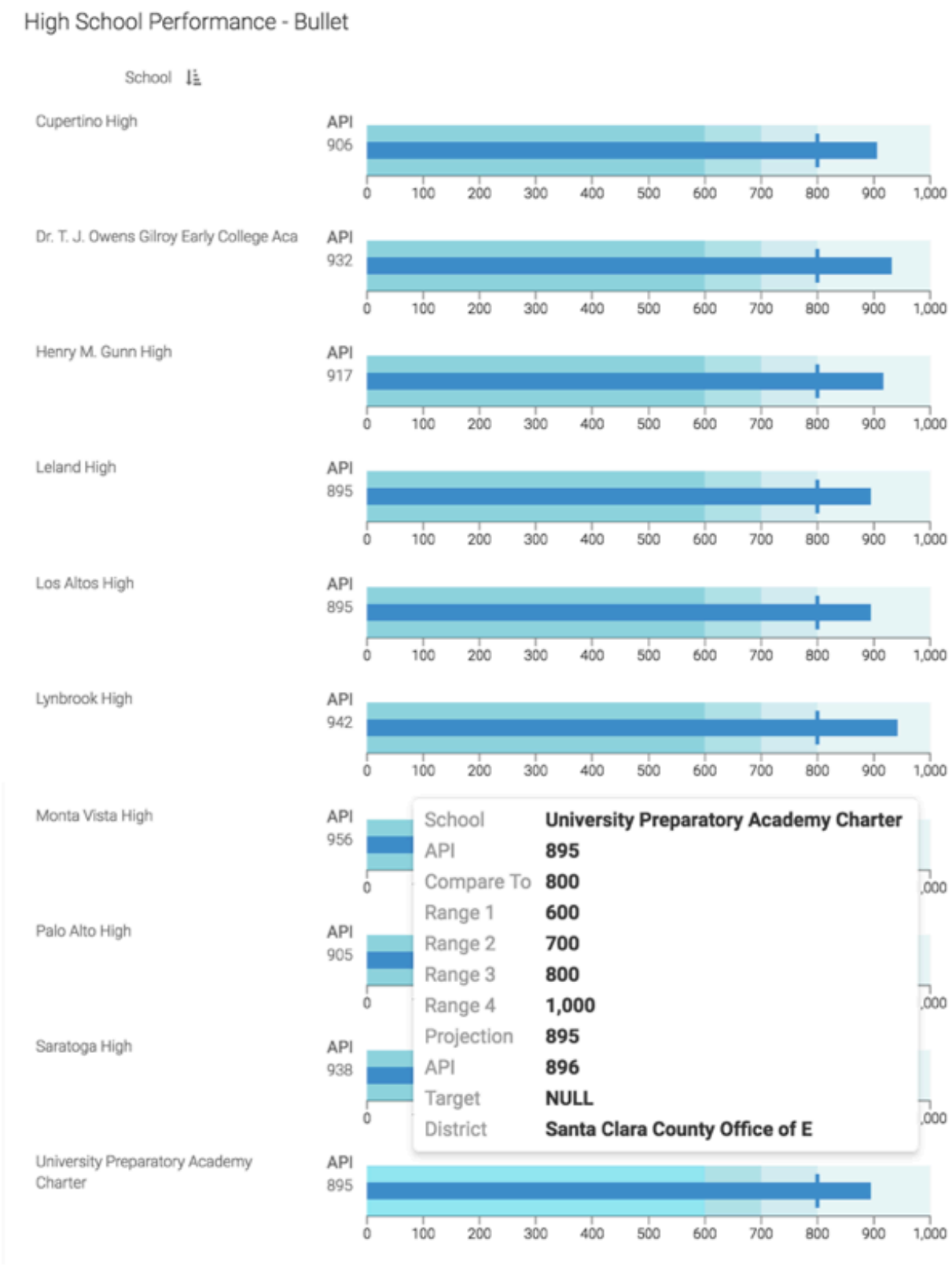

7. Change the order property of the Measure shelf by selecting the Bottom K option with a value of 10, and click REFRESH VISUAL.

   Note that in all cases here, the projection value is visible because it is relatively large compared to the projected improvement for schools that already reached the 800 API threshold.

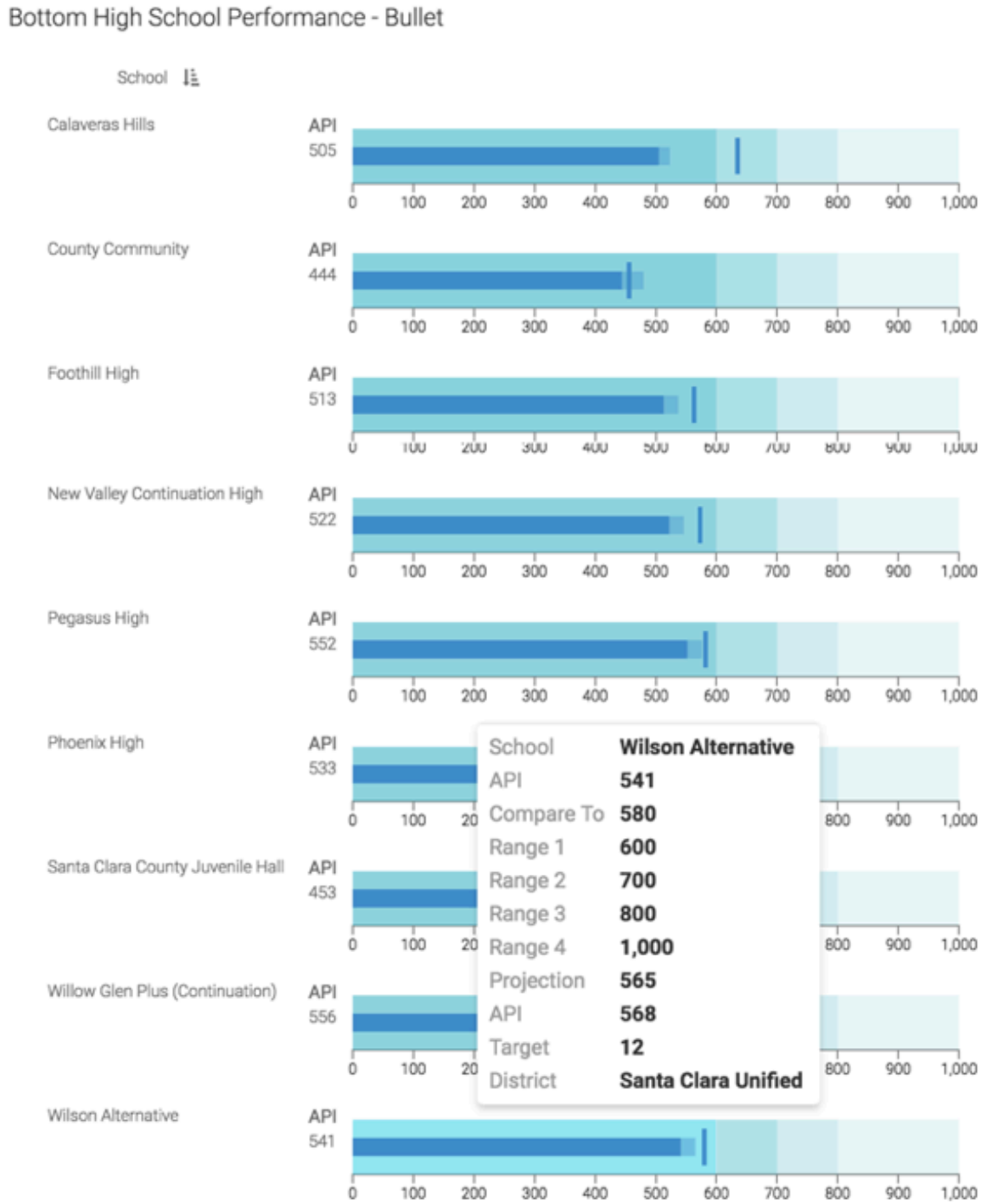

8. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'High School Performance - Bullet. You can also add a brief description of the visual as a subtitle below the title of the visualization.

9. At the top left corner of the Dashboard Designer, click SAVE.

10. At the top left corner of the Dashboard Designer, click CLOSE.

What to do next

To adjust the Bullet visual, check all the available settings for this visual.

Shelves for bullets

Overview of shelves for Cloudera Data Visualization Bullet visuals.

A Bullet visual has the following shelves:

Measure

Add a Measure field from the DATA menu to display an aggregate.

This is the feature measurement, such as the current value.

This shelf accepts a single field.

Compare To

Add a Measure field from the DATA menu to show a comparison aggregate as a target value.

This shelf accepts a single field.

Qualitative Ranges

Add Measure fields from the DATA menu to specify a range of values, such as poor, satisfactory, and good. This determines the background shading intensity.

This shelf accepts multiple fields.

Projection

Add a Measure field from the DATA menu to show a projected value based on current trends.

This shelf accepts a single field.

Label

Add as many as two Measure or Dimension fields from the DATA menu to display label(s) for each graph. This overrides the default value provided by the field on the Measure shelf.

This shelf accepts 0 to 2 fields.

Tooltips

Add Measure fields from the DATA menu for additional information in the graph's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimension field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Box plot

Cloudera Data Visualization enables you to create Box Plot visuals.

About this task

A Box Plot visual displays the distribution of data values. The box represents the bulk of the data (25th percentile through 75th percentile), and whisker lines represent statistical minimum and maximum values.

The following example uses the 2012 and 2013 API Scores for California Schools. The School Performance dataset is used, which is built on a datafile that contains the California School APIs from 2012 and 2013. The range of possible scores varies from a low of 200 to a high of 1000. The statewide API performance target for all schools is 800.

For an overview of shelves that specify this visual, see *Shelves for box plots*.

Procedure

1. Start a new visual based on the NYC Taxi Data dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Box Plot.

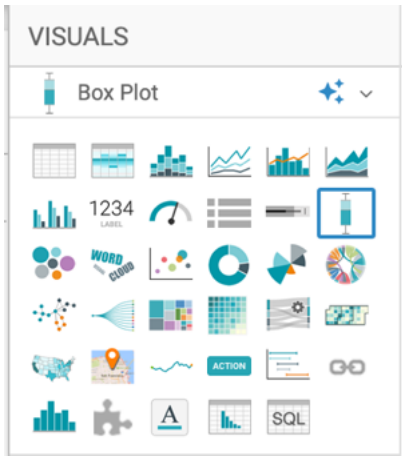

The shelves of the visual changed. They are now Dimension, Measure, X Trellis, Y Trellis, and Filters.

The mandatory shelves are:

- Dimension: you can specify one dimension for grouping the box plot
- Measure: you can specify the measurement for box plot calculations

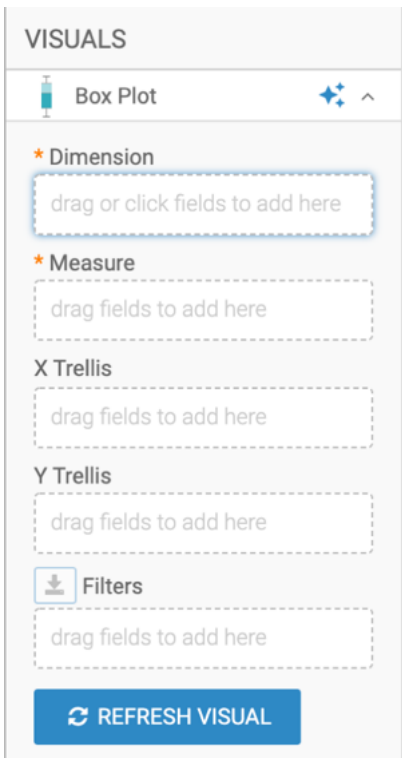

3. Populate the shelves:
   a) Add the Dname (district name) field to the Dimension shelf.
   b) Add the Api13 (API obtained in 2013)field to the Measure shelf.
   c) Add the Cname (county name) field to the Filters shelf, and select a single value. In this example, Santa Clara is used.
   d) Add the Stype (school type) field to the Filters shelf, and select the value H, for Highschool.
4. Alias the shelves.

5. Click REFRESH VISUAL.

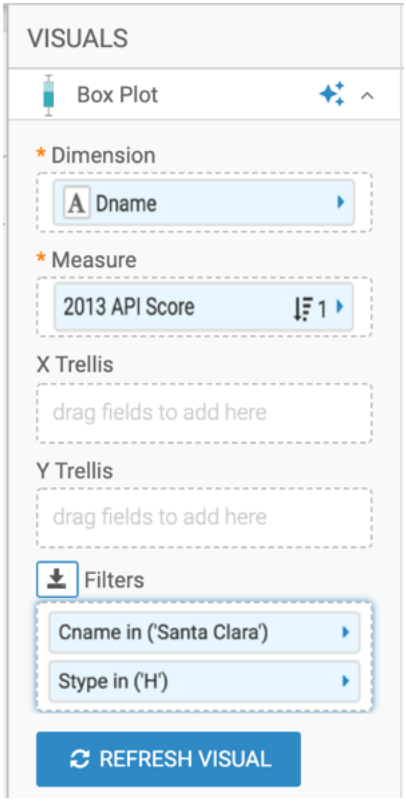

The new Box Plot visual appears.

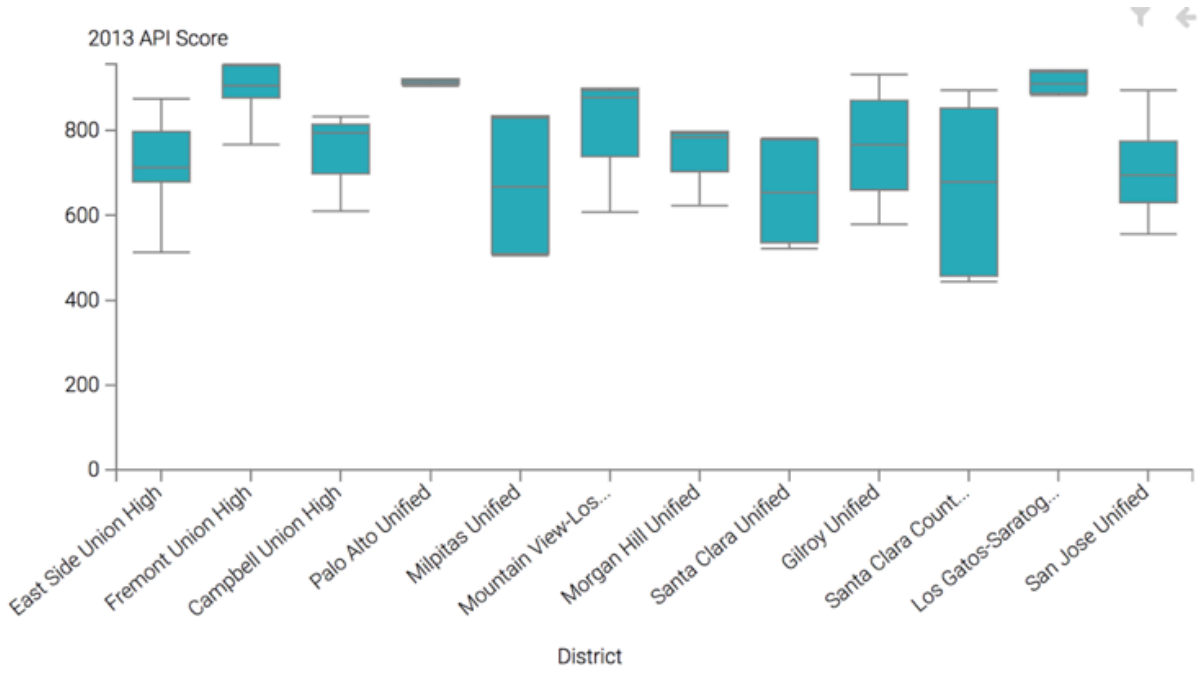

6. Hovering over a particular box plot distribution rectangle opens a Tooltip modal that contains the statistical measures of that grouping, including the minimum, the first quartile, the median/second quartile, the third quartile, and the maximum.

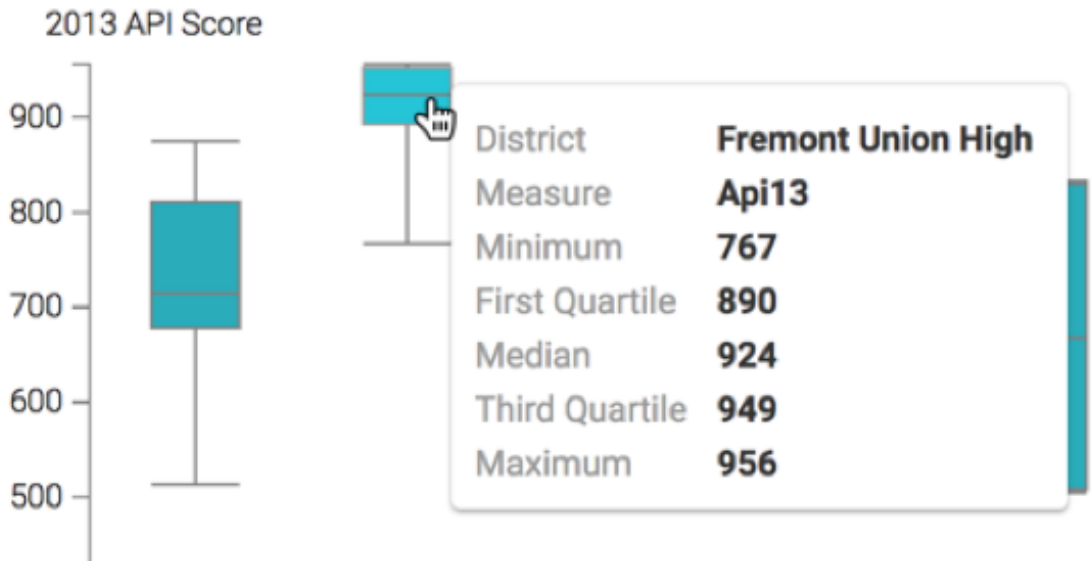

7. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'High School Performance - Box Plot'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
8. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

If you wish to generate a horizontal Box Plot visual, see the instructions in *Making box plots horizontal*.

To adjust the Box Plot display, check all the available settings for this visual.

Related Information

Creating a visual
Making box plots horizontal

Shelves for box plots

Overview of shelves for Cloudera Data Visualization Box Plot visuals.

A Box Plot visual has the following shelves:

Dimension

Add a Dimension field from the DATA menu to specify the grouping of each box plot.

This is the feature measurement, such as the current value.

This shelf accepts a single field.

Measure

Add a Measure field from the DATA menu; this is the measurement that determines the shape of the box plot.

We optimized Box plot statistics for Impala, Hive, and Spark SQL connections.

This shelf accepts a single field.

X Trellis

Use a Dimension field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Packed bubbles

Cloudera Data Visualization enables you to create Packed Bubbles visuals.

About this task

Packed Bubbles visuals compare the relative values in close physical proximity. Two main series of data are used: the field on the Dimensions shelf names the bubble and the field on the Measures shelf determines the relative area of the bubble.

This is different from the traditional approach, where bubbles represent three dimensions of data — one each for horizontal and vertical axes, and one for the area of bubble. This type of visualization is part of flexibility built into Scatter visuals available in Cloudera Data Visualization.

Note:

This visual type supports smooth rendering of streaming data.

The following steps demonstrate how to create a new Packed Bubbles visual on the World Life Expectancy dataset [data source samples.world_life_expectancy]. For an overview of shelves that specify this visual, see *Shelves for packed bubbles visuals*.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Packed Bubbles.

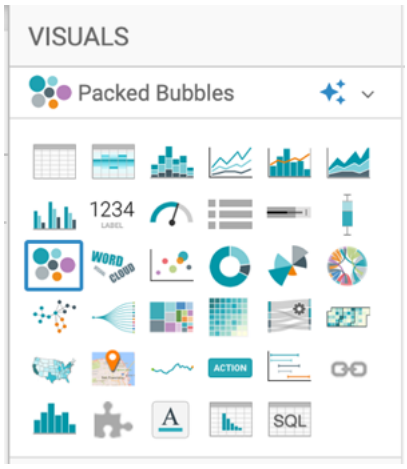

The shelves of the visual changed.

They are now Dimension, Measure, Color, Tooltips, X Trellis, Y Trellis, and Filters.

Both Dimension and Measure are mandatory.

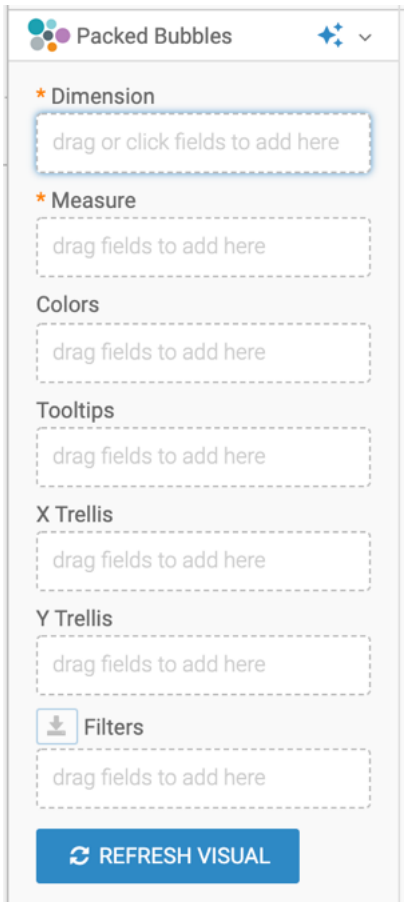

3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) Under Dimensions, select country and drag it to the Dimension shelf.
   b) Under Measures, select population and drag it to the Measure shelf.

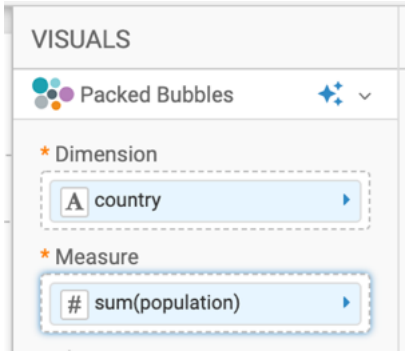

4. Change the aggregate of the population field from sum(population) to avg(population):
   a) On the shelf of the visual, click the caret-right icon to the right of the field.
   b) In the FIELD PROPERTIES menu, click Aggregates Average .

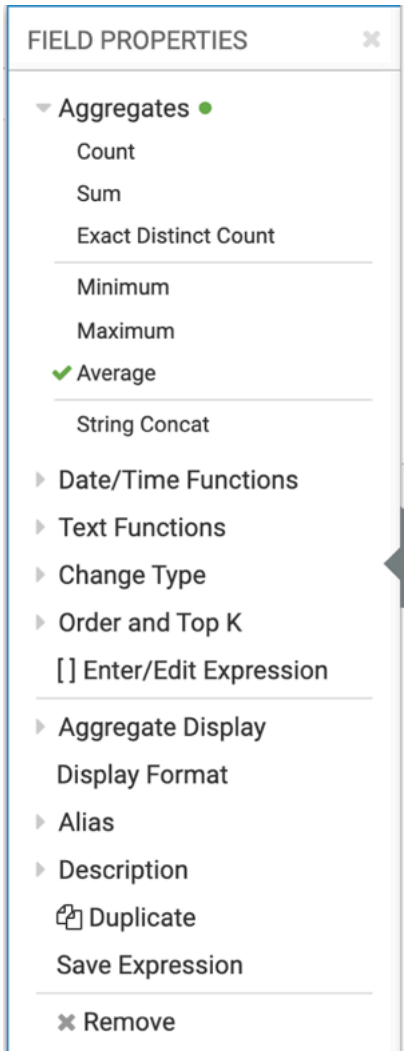

   c) Click the x icon at the top of the Column Properties menu to close it.

5. Click REFRESH VISUAL.

   The Packed Bubbles visual appears.

6. On the Filters shelf, add further Dimensions and Measures from the DATA menu.

   This enables you to dynamically control the data input, and discover the data at a more granular level.

   For example, from Dimensions, drag year to the Filters shelf.
7. On the Filter shelf, click the (down arrow) icon on the year field, and select Pick values from a list.

   The Filter for year modal window appears.
8. In the Values tab, select 2010, and click APPLY.

9. On the Measures shelf, click the (down arrow) icon on the avg(population) field, select Order and Top K, and select Ascending.

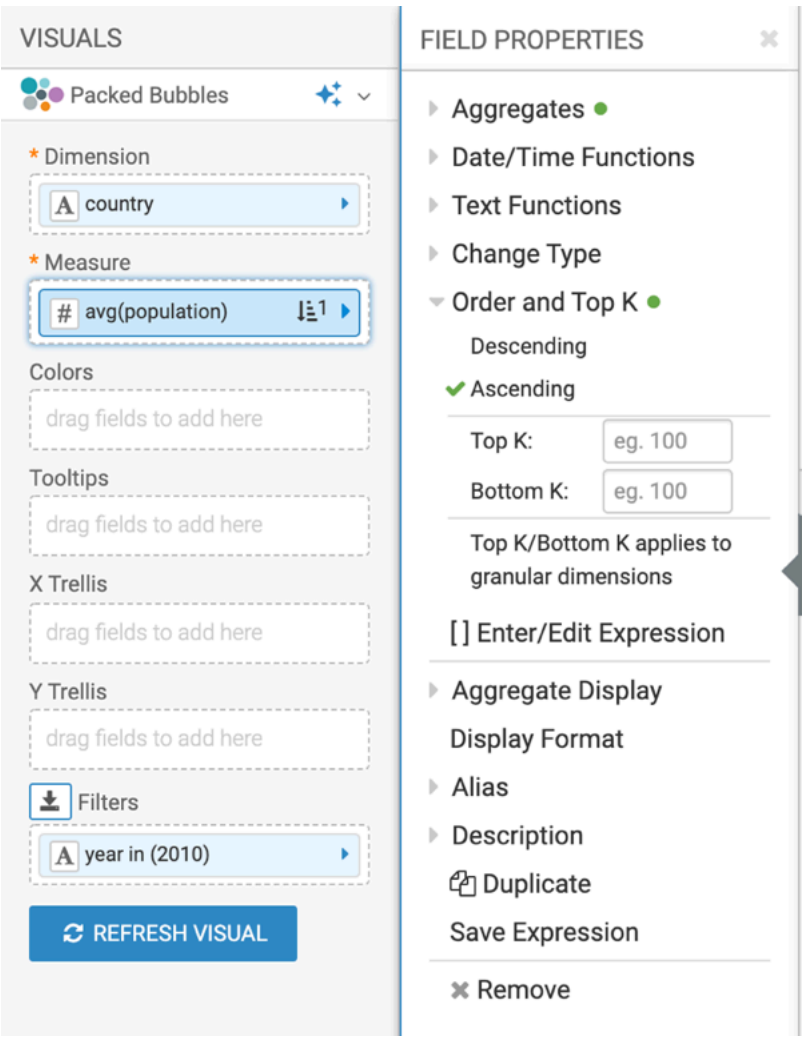

10. Click REFRESH VISUAL to see the updated Packed Bubbles chart.

11. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

    In this example, the title is changed to 'World Population - Packed Bubbles'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

12. At the top left corner of the Dashboard Designer, click SAVE.

Shelves for packed bubbles

Overview of shelves for Cloudera Data Visualization Packed Bubbles visuals.

A Packed Bubbles visual has the following shelves:

Dimensions

Add a Dimension field from the DATA menu to determine the number of bubbles to plot, and their names.

This is a mandatory shelf, and it accepts a single field.

Measures

Add a Measure field from the DATA menu to determine the size of the individual bubbles.

This is a mandatory shelf.

This shelf displays only one field on the visual. Additional fields appear only in the Tooltip.

Colors

To visually differentiate bubbles by category, add a Measure field to this shelf.

This shelf accepts at most one field.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimension field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Word cloud

Cloudera Data Visualization enables you to create Word Cloud visuals.

About this task

Word Cloud visuals aggregate text data, and display the words with higher frequency of occurrence. You can use them to emphasize prominent phrases or words in textual data. This visual type includes many manual setting adjustments.

In the following example, the Shakespeare's Sonnets dataset is used. For an overview of shelves that specify this visual, see *Shelves for word cloud visuals*.

Procedure

1. Start a new visual based on the Shakespeare's Sonnets dataset.

   For instructions, see *Creating datasets* and *Creating a visual*.

2. In the VISUALS menu, find and click Word Cloud.

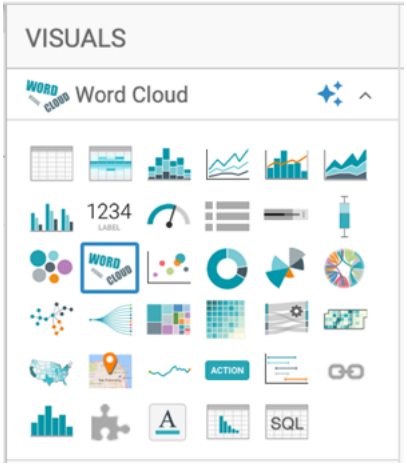

The shelves of the visual changed. They are now Dimension, Measure, Colors, Tooltips, X Trellis, Y Trellis, and Filters. The mandatory shelves are Dimension and Measure.

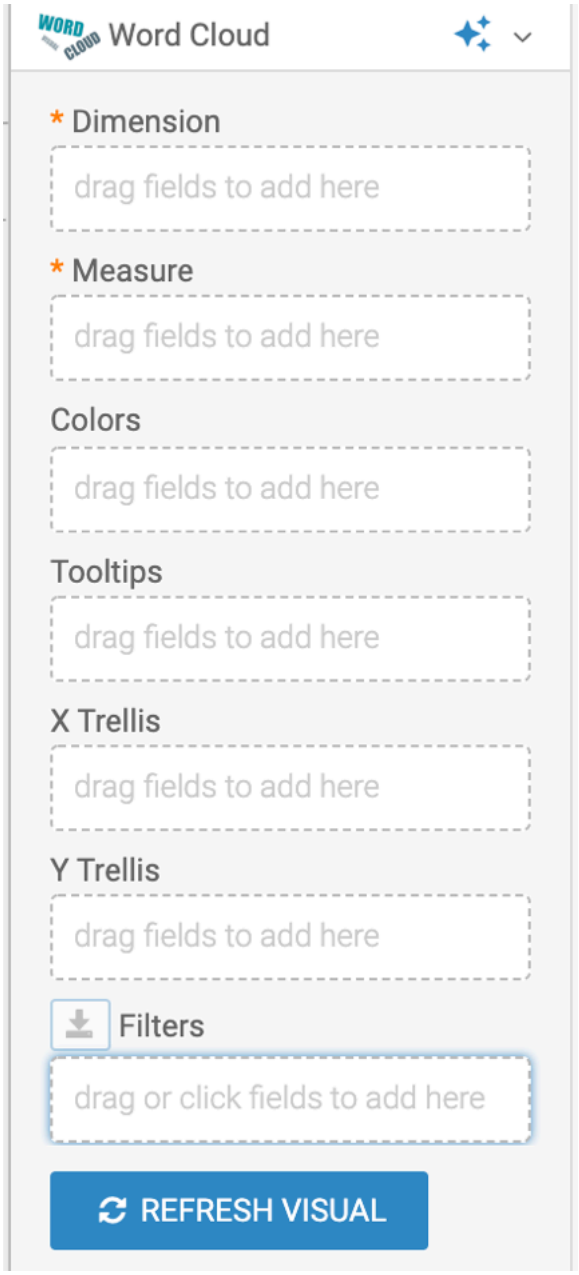

3. Populate the shelves from the available fields:
   a) Add the field Text to the Dimension shelf.
   b) Add the field Record Count to the Measure shelf.
4. Click REFRESH VISUAL.
5. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'Shakespeare Sonnets - Word Cloud'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
6. At the top left corner of the Dashboard Designer, click SAVE.

Results

The visual should look something like this:

What to do next

To adjust how the Word Cloud is displayed, check all the available settings for this visual.

It is interesting to compare different texts by examining their different word clouds. For more information, see *Trellised word clouds*.

Trellised word clouds

Cloudera Data Visualization enables you to create trellised Word Cloud visuals.

About this task

In the following example, you can compare text profiles using word cloud visuals, by considering the text of the collected plays of William Shakespeare.

The Shakespeare's Plays dataset is used, which was created from the shakespeare_play_text.csv datafile.

Procedure

1. In the VISUALS menu, find and click Word Cloud.

   For instructions, see *Creating datasets* and *Creating a visual*.
2. Populate the shelves from the available fields:
   a) Add the field Text to the Dimension shelf.
   b) Add the field Record Count to the Measure shelf.
   c) Add the field Play to the X Trellis shelf to trellis the visual.
   d) Add the field Play to the Filters shelf, and select some of the plays. In this example the visual is filtered on Anthony and Cleopatra, Hamlet, Julius Caesar, Othello, and Troilus and Cressida.
3. Remove the trellis borders, as described in *Displaying trellis borders*.
4. Click REFRESH VISUAL.
5. Click SAVE.

Results

You can see that the Word Cloud visuals for these plays feature the names of leading characters prominently.

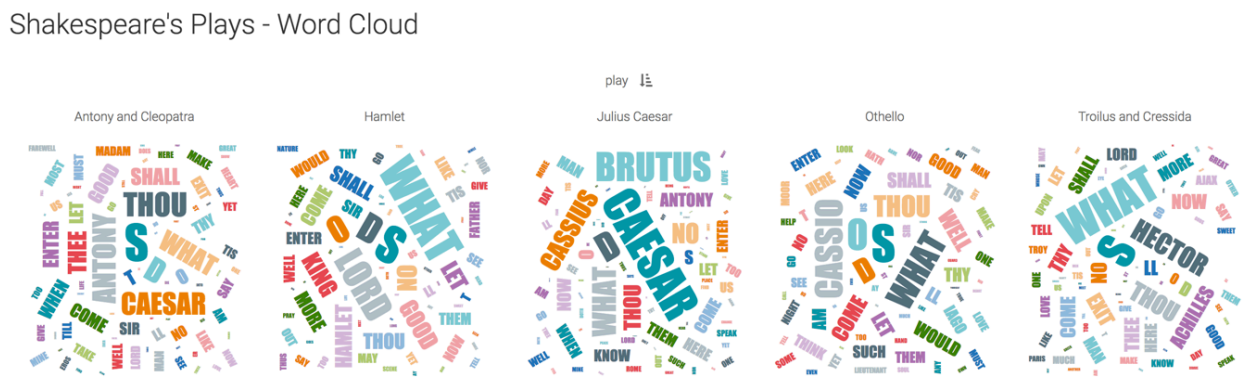

Shelves for word clouds

Overview of shelves for Cloudera Data Visualization Word Cloud visuals.

A Word Cloud visual has the following shelves:

Dimension

Add a Dimensions field from the DATA menu to determine the number of individual words in the word bubble graph.

This shelf accepts a single field.

Measure

Add a Measures field from the DATA menu to determine the size of the individual words.

This shelf accepts a single field.

Colors

To visually differentiate words by color gradient based on the field value, add a Measures field to this shelf.

This shelf accepts at most one field.

Tooltips

Add Measures fields from the DATA menu for additional information in the graph's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimensions field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimensions field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimensions or Measures fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Scatter

Cloudera Data Visualization enables you to create Scatter visuals. Scatter charts (or scatter plots) have points that show the relationship between two sets of data. They are great for detecting groupings or patterns in a large set of data points.

Scatter visuals have the following additional features in Cloudera Data Visualization:

- You can use color as a third 'dimension'. For more information, see *Creating basic scatter visuals*.
- You can control the size of the marks that represent individual data points, as shown in *Adding size to scatter visuals*.
- You can add a regular increment, such as time, to create an animated representation of data. For more information, see *Adding transition animation to scatter visuals*.
- You can divide the data along natural categories to create side-by-side charts to see individual data points more clearly. For instructions, see *Adding trellises to scatter visuals*.

Note: This visual type supports smooth rendering of streaming data.

For an overview of shelves that specify this visual, see *Shelves for scatter visuals*.

Creating a basic scatter visual

Cloudera Data Visualization enables you to create a basic Scatter visual.

About this task

The following steps demonstrate how to create a new Scatter visual representation, showing the trend of life-expectancy and GDP per capita for selected countries.

Procedure

1. Start a new visual based on the World Life Expectancy dataset. The data source for this is samples.world_life_expectancy.

   For instructions, see *Creating a visual*.

2. In the visuals menu, find and click Scatter.

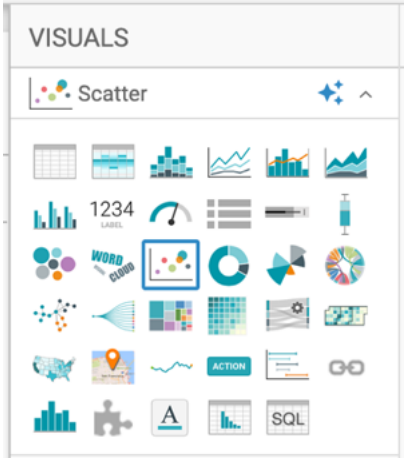

The shelves of the visual changed. They are now X Axis, Y Axis, Colors, Size, Transition, Tooltips, and Filters. The mandatory shelves for Scatter visuals are X Axis and Y Axis. The fields placed on these two shelves may be easily swapped by switching X Axis and Y Axis.

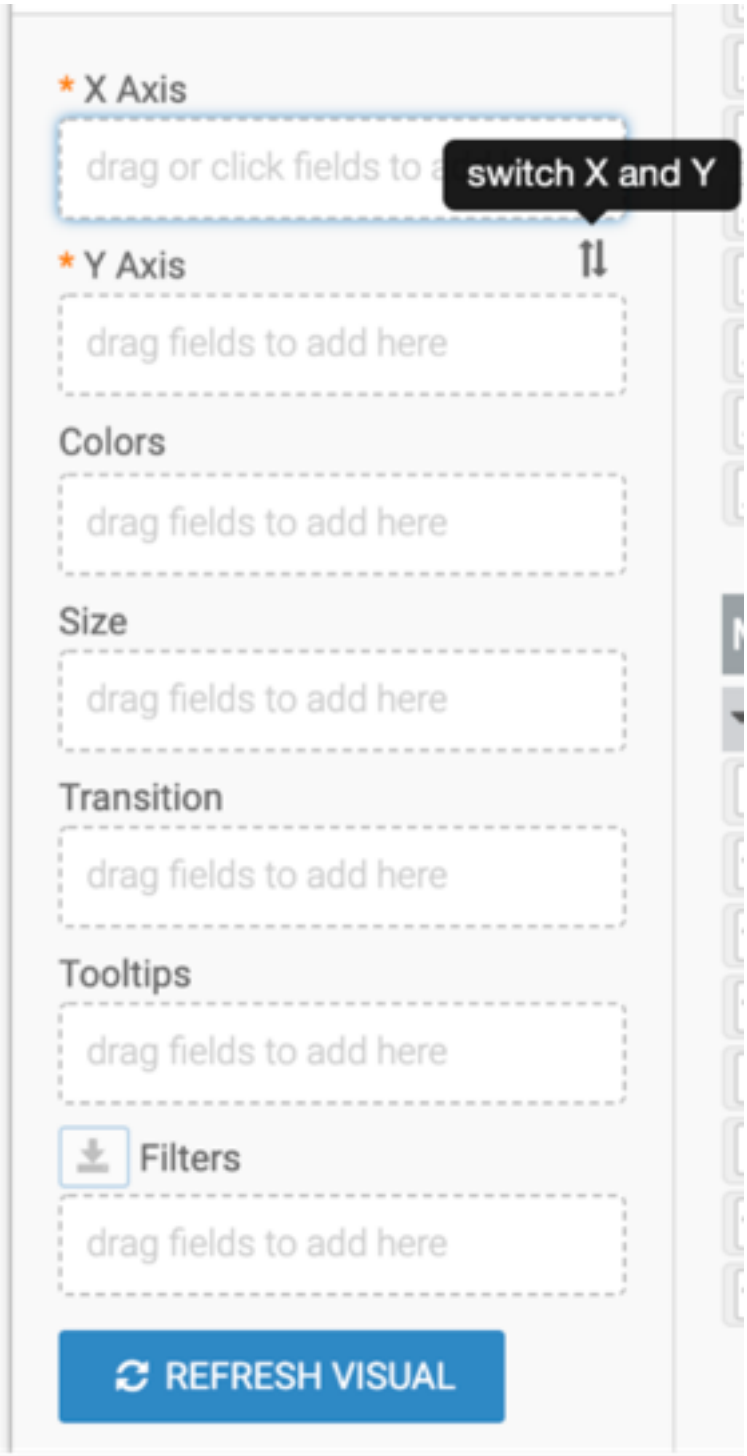
* X Axis
drag or click fields to a
switch X and Y
* Y Axis
drag fields to add here
Colors
drag fields to add here
Size
drag fields to add here
Transition
drag fields to add here
Tooltips
drag fields to add here
Filters
drag fields to add here
REFRESH VISUAL

3. Populate the shelves from the available fields of Dimensions and Measures in the DATA menu.
   a) Under Measures, select gdp_per_capita and drag it to the X shelf.
   b) Under Measures, select life_expectancy and drag it to the Y shelf.

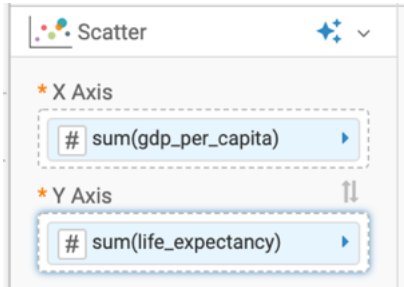

4. In both cases, remove the aggregate to see individual data points.
   a) On the shelf, on the sum(gdp_per_capita) field, click the (right arrow) icon to open the FIELD PROPERTIES menu.
   b) Expand Aggregates and click the check mark next to Sum to remove the aggregate.
   c) Repeate it for the sum(life_expectancy) field

   The shelves now contain the modified fields.

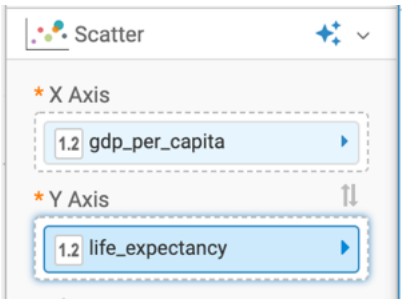

5. Click REFRESH VISUAL.

   The Scatter visual appears.

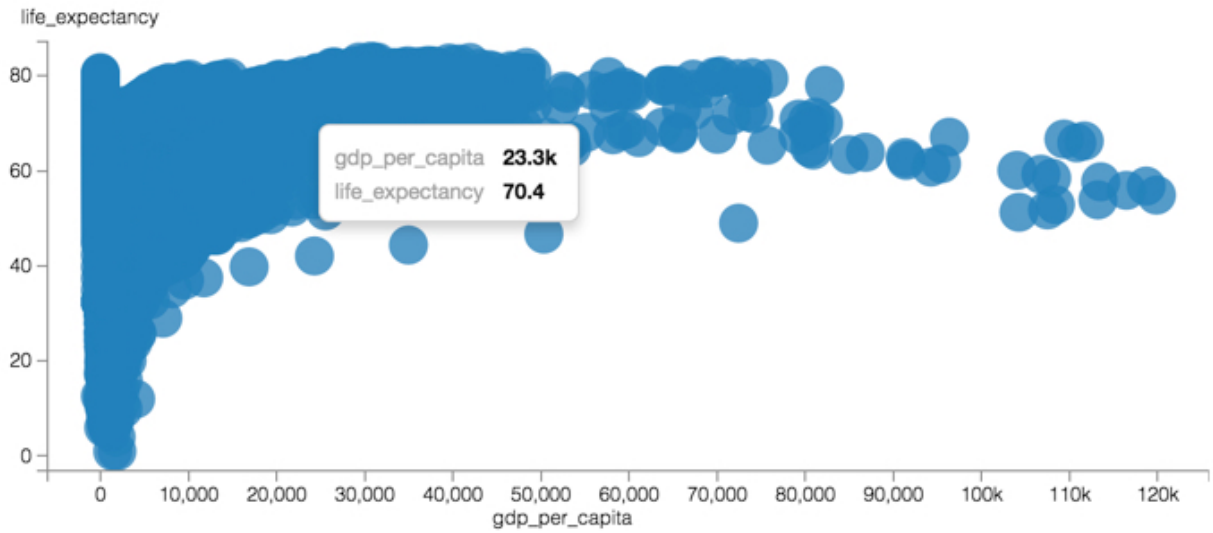

   While you can see the general shape of data and a few outliers, there is too little distinguishing information to help us understand the trends.

6. Use the Colors shelf to see if a pattern emerges.

   Under Dimensions, select country and drag it to the Colors shelf .

7. Click REFRESH VISUAL and examine the resulting graph.

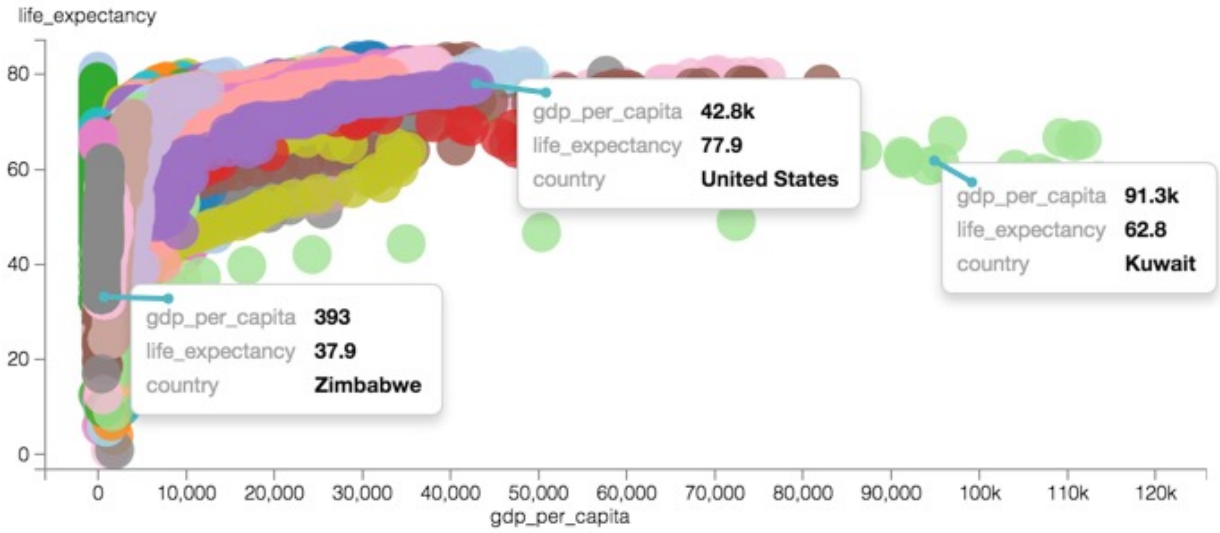

You can now see the trend for some of the countries very clearly, over time. The visualization is less than ideal, yet obvious details show:

- Zimbabwe, in gray, shows an appreciable increase in life expectancy, but relative to other countries, very little economic improvement (measured as GDP per capita).
- United States, in purple, shows the expected improvement in both longevity and in GDP.
- Kuwait, in light green, is a true outlier that shows phenomenal increase in GDP per capita, but suffers a noticeable decrease in life expectancy from about the mid-century to present day.

8. Add aggregation back to the measures on the X Axis and Y Axis shelves. This time, you want to see the average of the dimensions.
   a) On the shelf, on the gdp_per_capita field, click the (right arrow) icon to open the FIELD PROPERTIES menu.
   b) Expand Aggregates and select Average.
   c) Repeate it for the life_expectancy field

   The shelves now contain the modified fields.

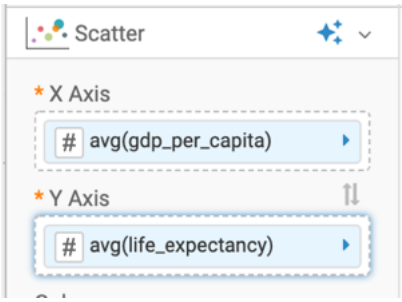

9. Turn on the *Changing legend style and removing the legend* feature.

10. Click REFRESH VISUAL.

The Scatter visual appears.

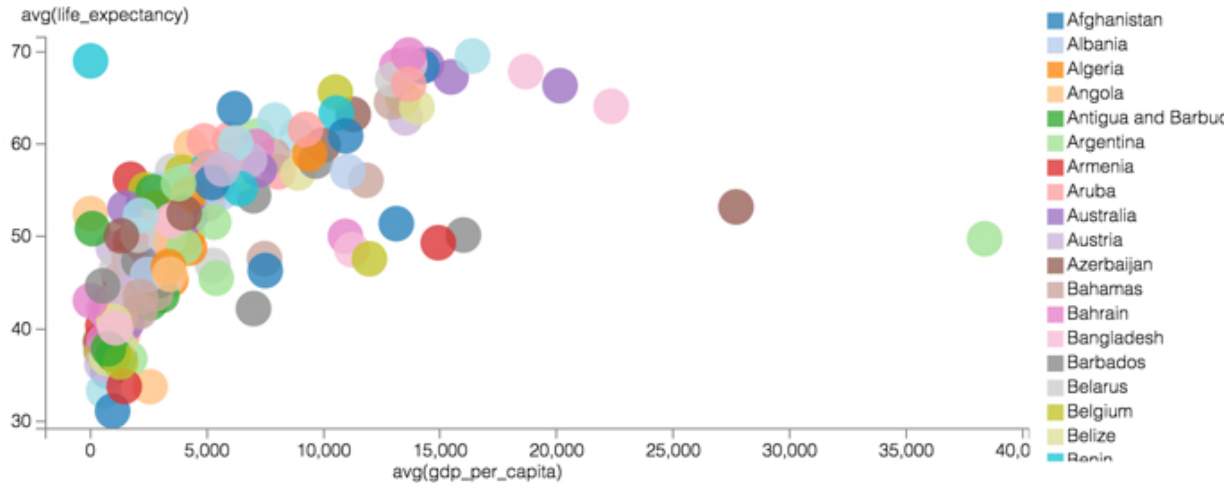

The distribution is an average of ALL years covered by the dataset, from 1990 through 2010. Still, some outliers are already clearly visible: Qatar and Kuwait for exceptionally high GDP per capita, Tokelau for high life expectancy at very low GDP per capita, and Sierra Leone, with the lowest life expectancy in the world.

Figure 1: Contrasting Average Life Expectancy and GDP Per Capita, World-Wide

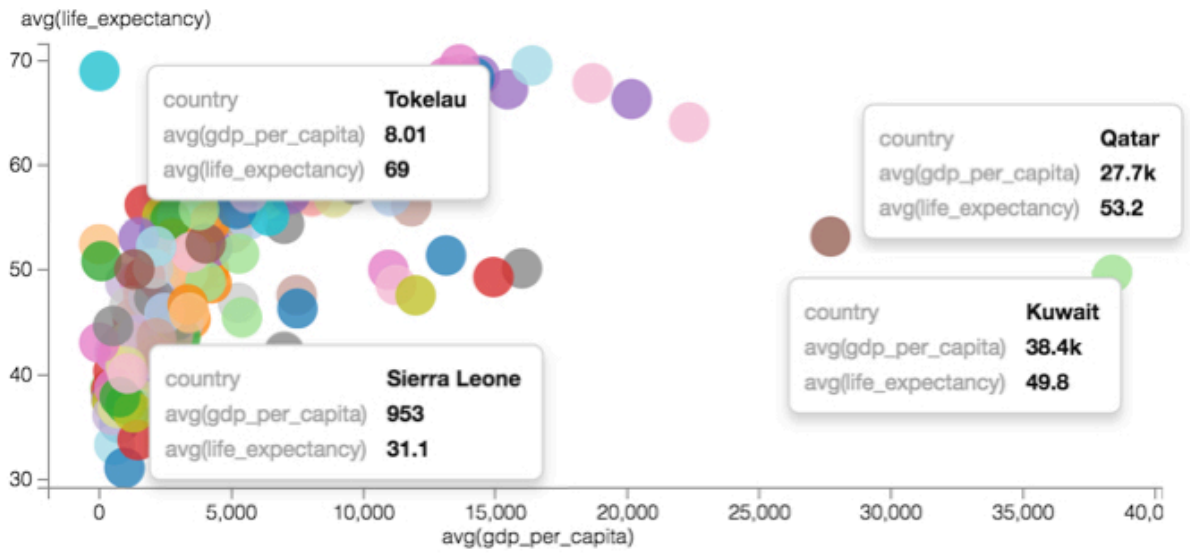

11. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

In this example, the title is changed to 'World Population - Scatter'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

12. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

It is useful to remember at this point that GDP per capita is actually influenced by the population of the country.

As a next step, look at how you can show the population variation on this graph, by *Adding size to a scatter visual*.

Adding size to scatter visuals

In Scatter visuals, you can control the size of the individual bubbles to show variation of a measure more clearly.

About this task

The World Life Expectancy dataset has several other dimensions and measurements that you can use to enrich the scatter visual. Consider how using the Size shelf, which controls the size of the individual bubbles, changes our immediate understanding of the dataset and represents the relative population size of each country more clearly.

The starting point of the following example is the 'World Population - Scatter' visual developed in *Creating basic scatter visuals*.

Procedure

1. Click Clone to create a new visual that has the same properties as the original.

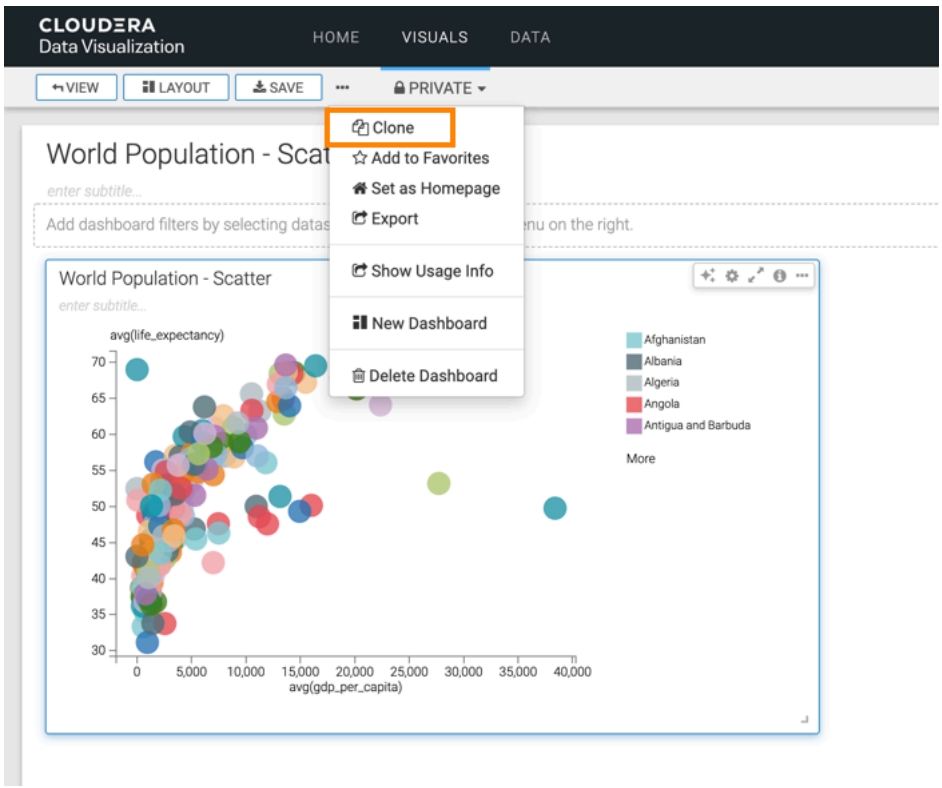

2. Under Measures, select population and drag it to the Size shelf.
3. On the sum(population) field, click the (right arrow) icon to open the FIELD PROPERTIES menu.
4. Expand Aggregates and select Average.

5. Click REFRESH VISUAL.

   The Scatter visual appears with its own visible population size 'outliers': China, India, and the United States.

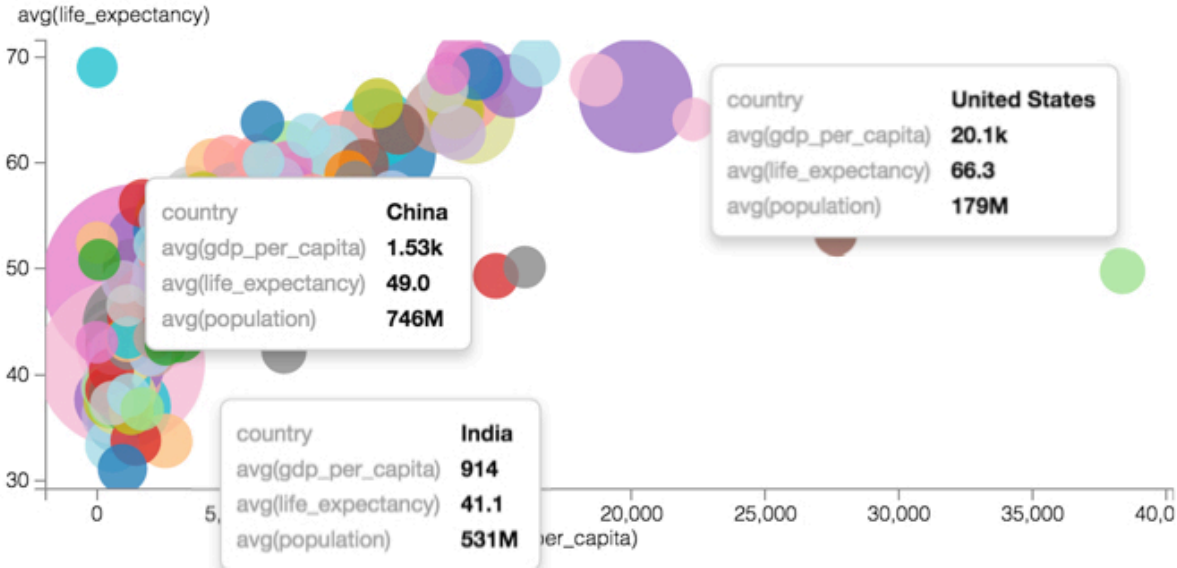

6. It is important to realize that the relative size of the bubbles is not linearly proportional to their values.

   At the lower range of population levels, the visual shows the same size marks for countries with very different average populations, such as Tokelau, where avg(population)=1.23K, Greenland, with avg(population)=34.5K, Luxembourg, with avg(population)=318K, and Kuwait, with avg(population)=751K. This is because by default,

the size of marks ranges from 10 to 50 pixels. Divided equally across the range of values, this squeezes too many disparate measurements on the lower end to use the same mark size 'bucket'.

Figure 2: World Population Figures with Mark Size Range of 10 - 50 Pixels

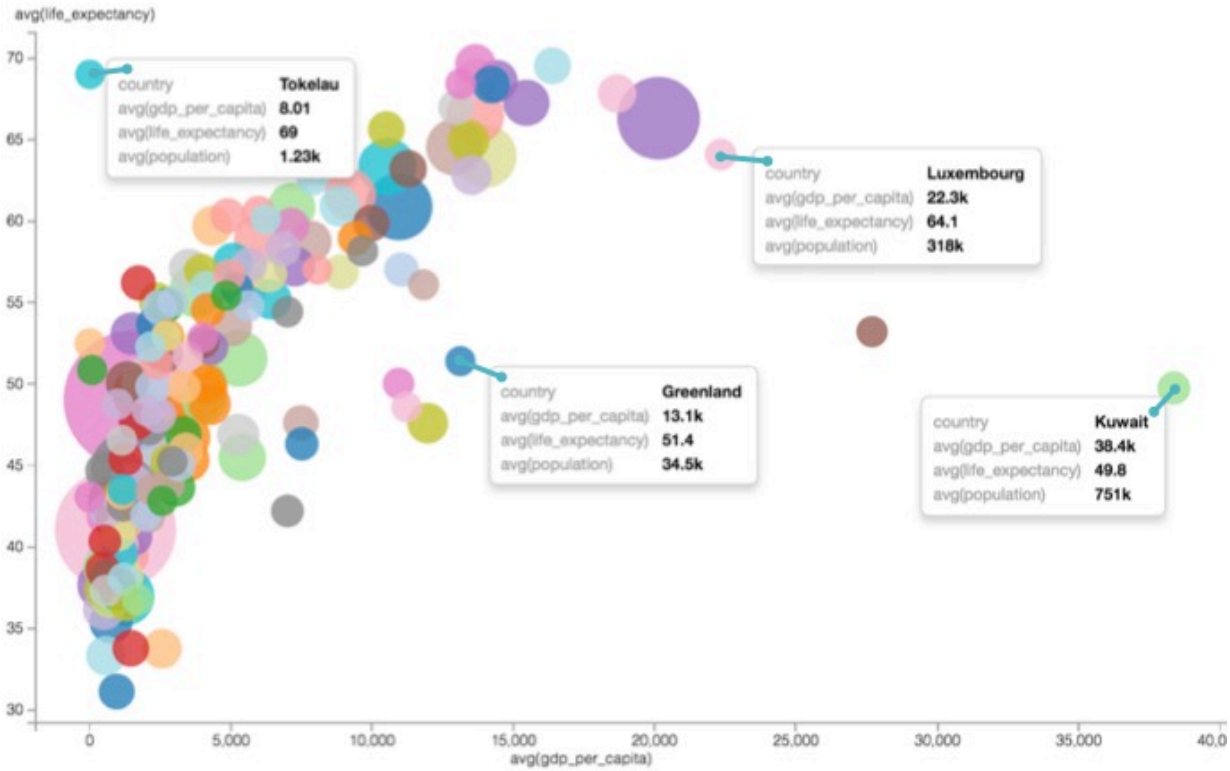

It is easy to somewhat alleviate this problem by changing the range of sizes of the bubbles, as described in *Changing the mark size range*. This graph shows the size range of 1-50.

Figure 3: World Population Figures with Mark Size Range of 1 - 50 Pixels

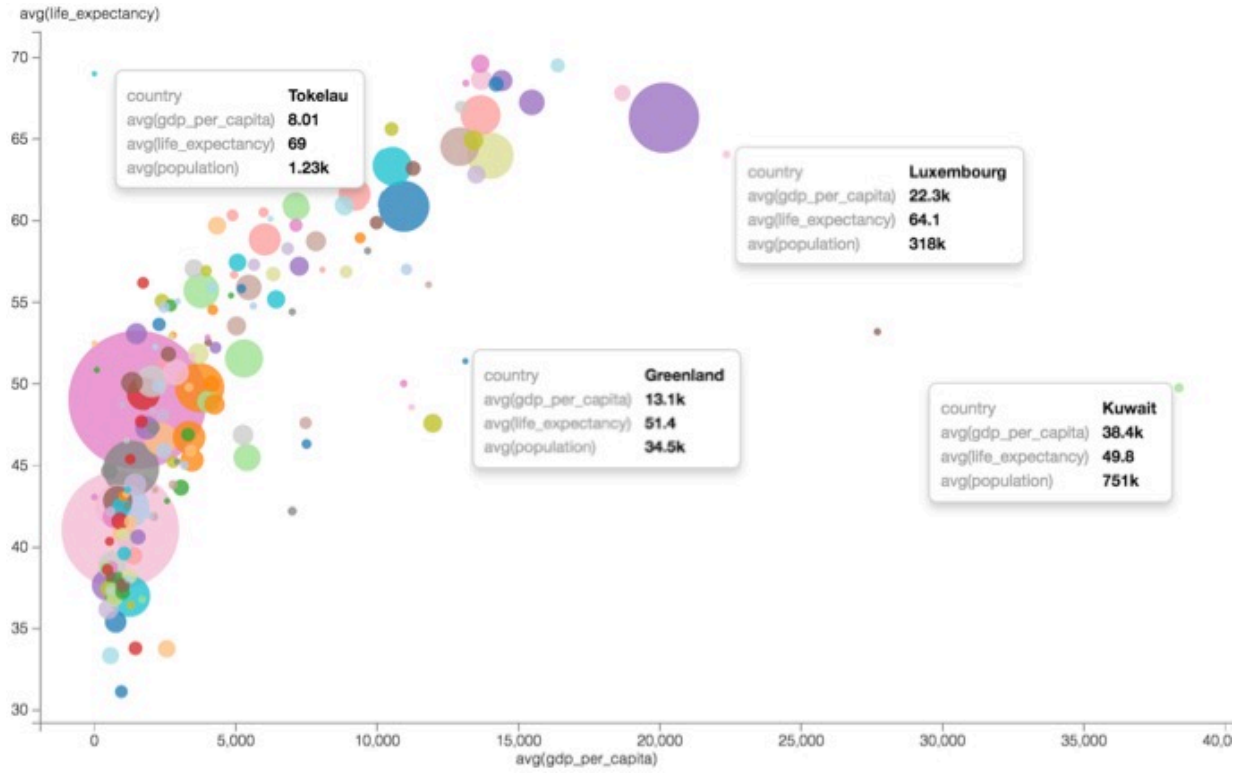

Incidentally, and obviously, the increased size range generates a more realistic representation of relative population size.

7. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'World Population - Scatter with Size'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

8. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

While you can see the average data for individual countries more clearly, these measurements do not help you to see the changes that occurred over the entire time domain of the dataset.

As a next step, look at how you can add the concept of time back into the visual by animating the scatter plot, as described in *Adding transition animation to scatters*.

Adding transition animation to scatter visuals

Scatter visuals have a powerful feature called transition animation.

About this task

In the World Life Expectancy dataset, the time is used as the transition 'axis' to show how GDP per capita, life expectancy, and population size changes over time, from year 1990 through year 2010.

The starting point of the following example is the 'World Population - Scatter with Size' visual developed in *Adding size to scatters*.

Procedure

1. Click Clone to create a new visual that has the same properties as the original.

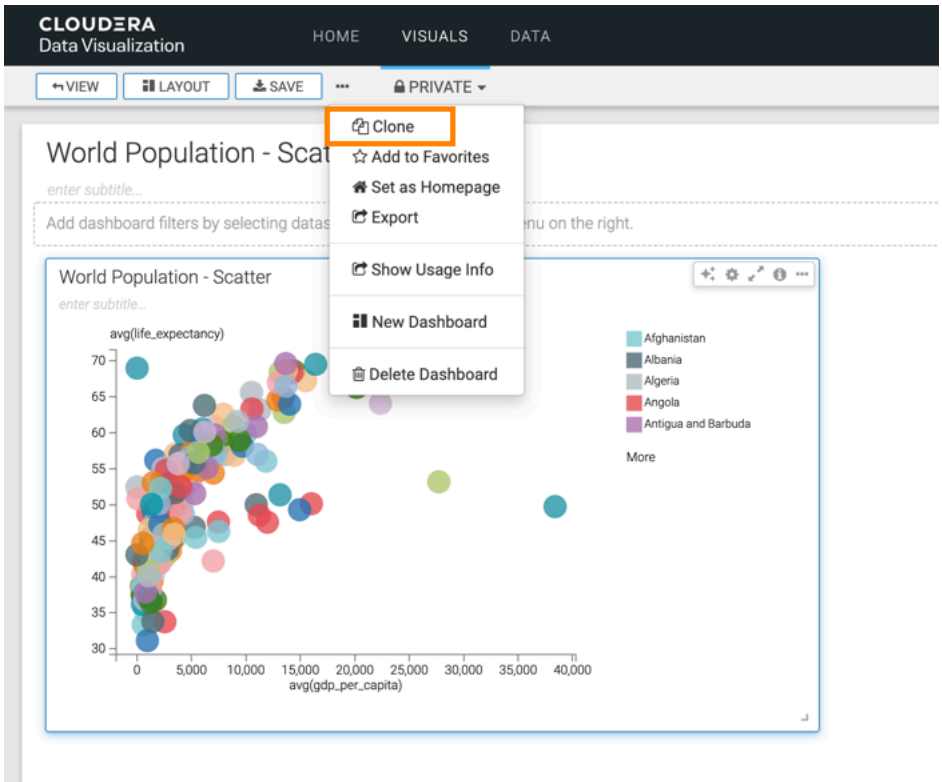

2. Under Dimensions, select year and drag it to the Transition shelf.
3. On the sum(population) field, click the (right arrow) icon to open the FIELD PROPERTIES menu.
4. Expand Aggregates and select Average.
5. Click REFRESH VISUAL.

   The Scatter visual refreshes, but this time you can see that it has animation controls at the very top, over the vertical axis legend. The controls common to all video animations: 'play' (which changes to 'pause' when the animation is active), 'back to start', 'back-up', 'advance', and 'to end'.

   You can also notice that both axes extend far beyond the data points for the initial conditions, year=1990. This is because the cover the entire range of measurements for all data.

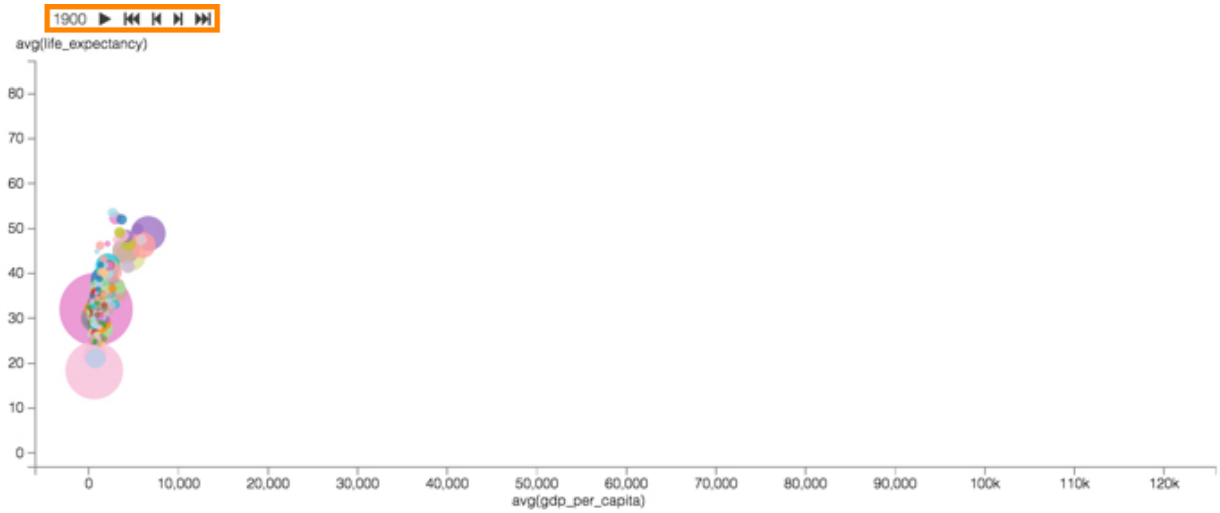

6. Turn on the gridline option, as described in *Enabling gridlines*.
7. Adjust the speed of animation, as described in *Changing transition delay*.
8. Press play and see the changes in data.

   Here are some highlights:

   - The World-Wide downward bounce in life expectancy centered on 1918, due to the Spanish Flu pandemic.

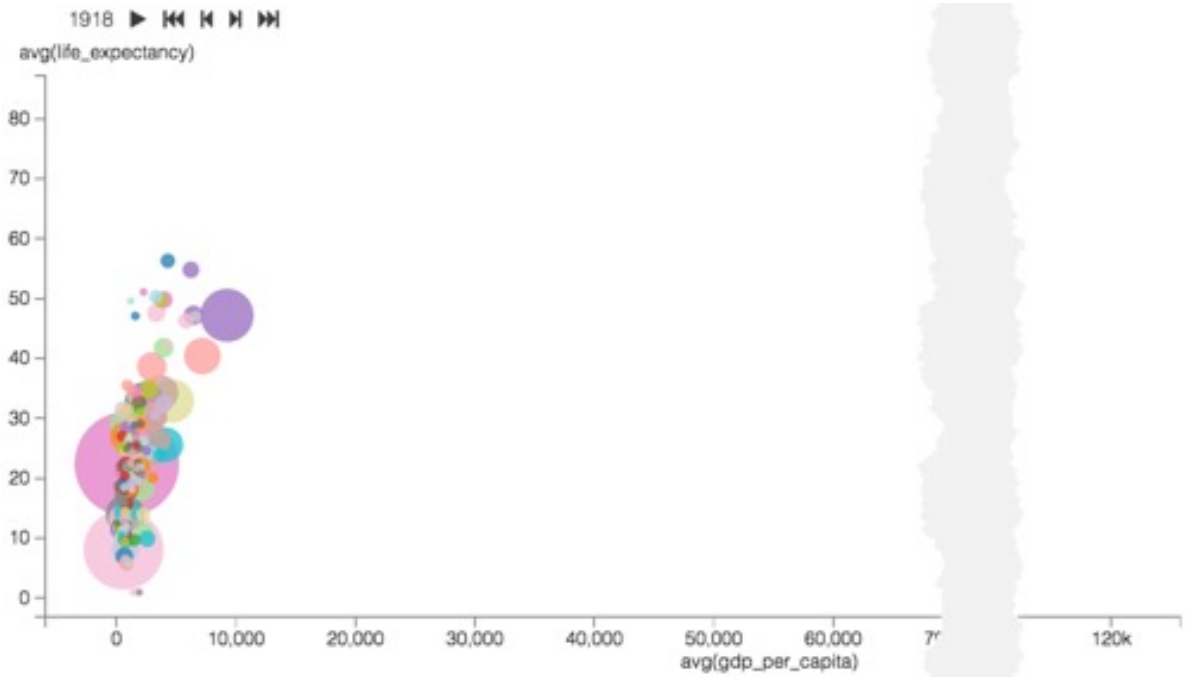

   - The Soviet Union secularization efforts and agricultural policies in its Asian territories resulted in war and famine that claimed many lives in Uzbekistan, Kazakhstan, Tajikistan, Kyrgyzstan, and Turkmenistan. This is reflected in the precipitous decrease in life expectancy for these countries in 1919-1923, with heaviest losses in 1920.

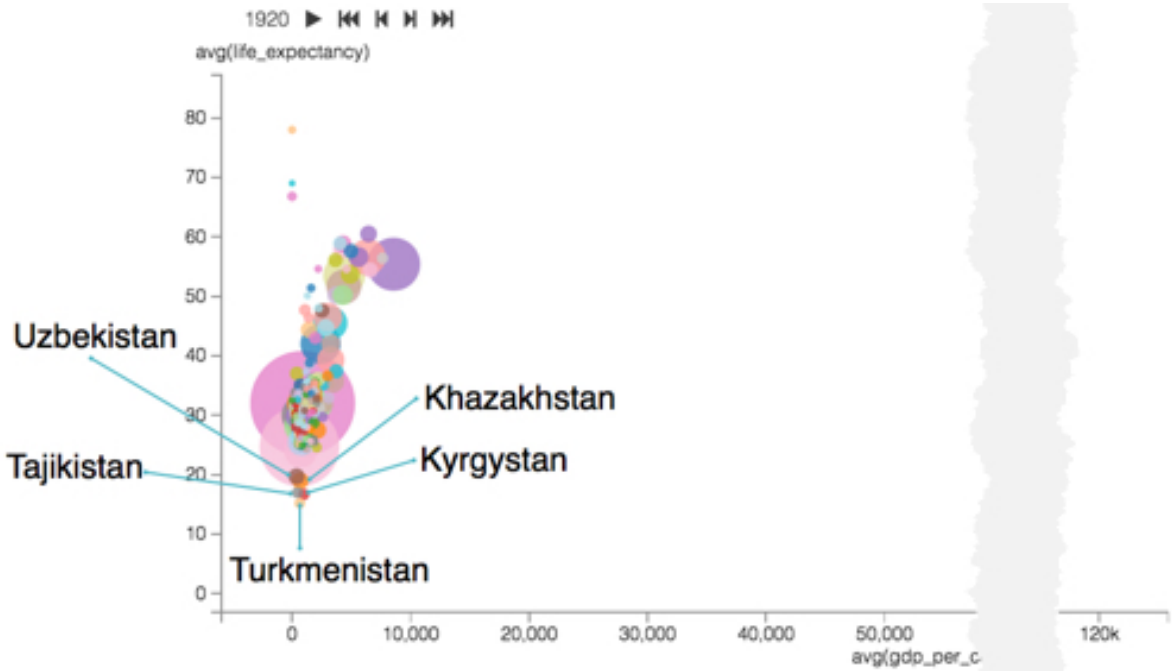

   - There was another significant decrease in life expectancy due to the loss of as many as 8 million lives in the Soviet famine of 1932–33 and more specifically, the Hlodomor. Soviet Union farm collectivization in the

Russia and Ukraine withheld grain from farmers and collectivized farms until unrealistic production quotas were be met.

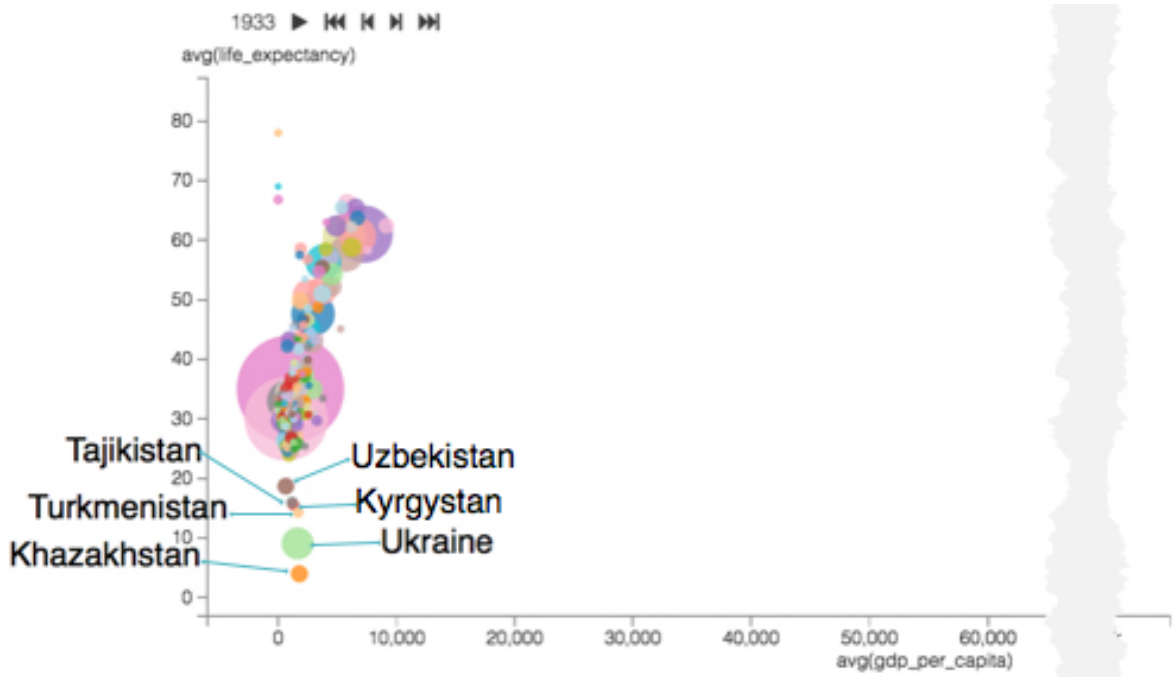

- The United States economy grew rapidly from mid-1930s through mid-1940s, because New Deal's economic relief and work creation policies stabilized the country after the Great Depression, while the World War II material support of the Allies and its own involvement in Europe, Africa, and the Pacific meant even greater production of materials, armaments, and two new naval fleets. You can see that GDP per capita increased nearly 3-fold, while the population increase and life expectancy were relatively modest.

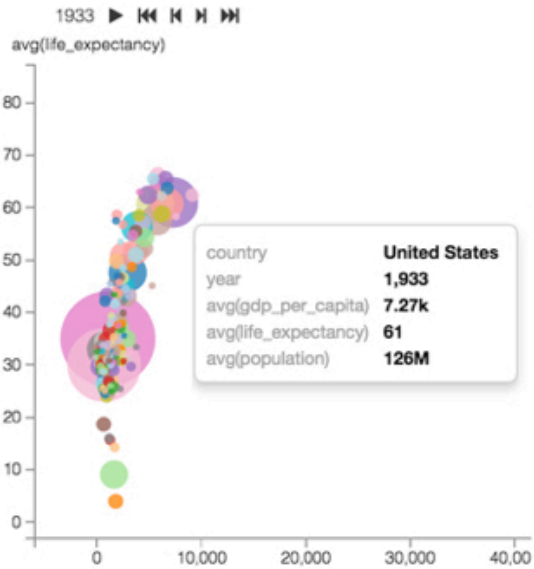

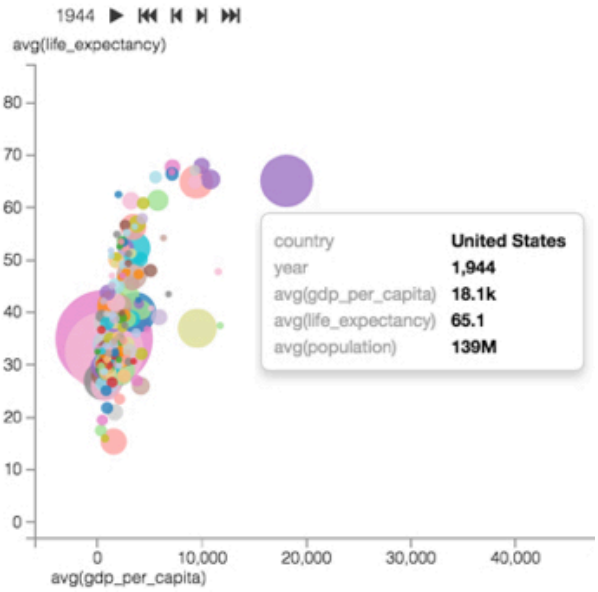

- The exceptional increase in GDP in a very modest country of Kuwait after the discovery of the world's sixth largest oil deposits in 1938, through 1956 peak levels of GDP per capita growth, and large scale modernization from 1946 through 1982. After a period of international instability in 1980s, the 1990 invasion

by Iraq destroyed much of the infrastructure. After 1991 US intervention, the country has been rebuilding its infrastructure.

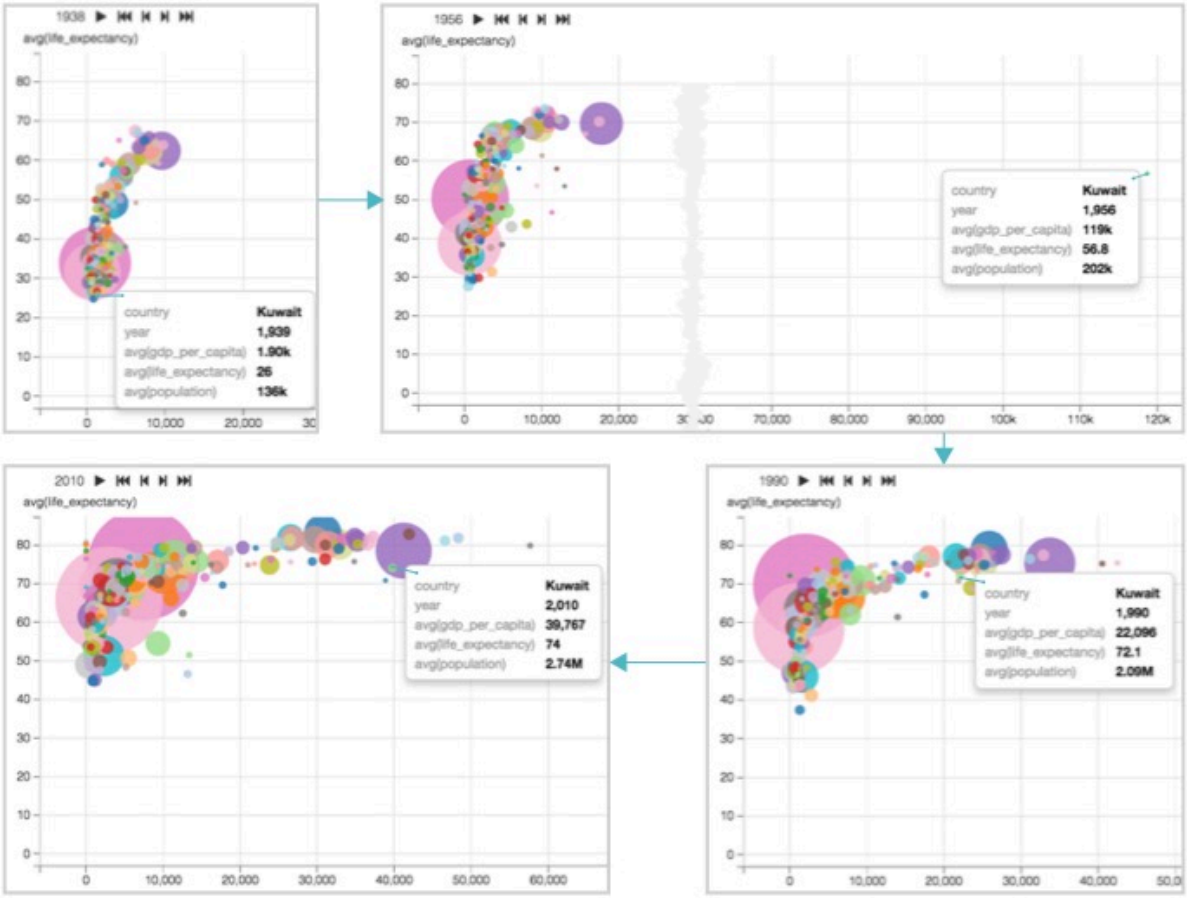

9. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'World Population - Scatter with Transition'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

10. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

While we can see the data for individual countries more clearly, it is often useful to contrast well-defined logical segments of data.

As a next step,you can compare and contrast GDP per capita and life expectancy for different UN Regions in *Adding trellises to scatter visuals*.

Adding trellis to scatters

In Scatter visuals, you can add trellises for comparing and contrasting data. To discover variation within the dataset, to see both similarities and differences among categorical subsets of data, it is very useful to see visuals for these categories of data side by side. In Cloudera Data Visualization, this is very easy to do with the trellis option, which is essentially a doubling down on one or both of the axes.

About this task

In the World Life Expectancy dataset, you can use un_region to compare how life expectancy and GDP per capita vary in different geographic zones. For an even greater degree of detail, you could use un_subregion.

The starting point of the following example is the 'World Population - Scatter with Transition' visual developed in *Adding transition animation to scatters*.

Procedure

1. Click Clone to create a new visual that has the same properties as the original.

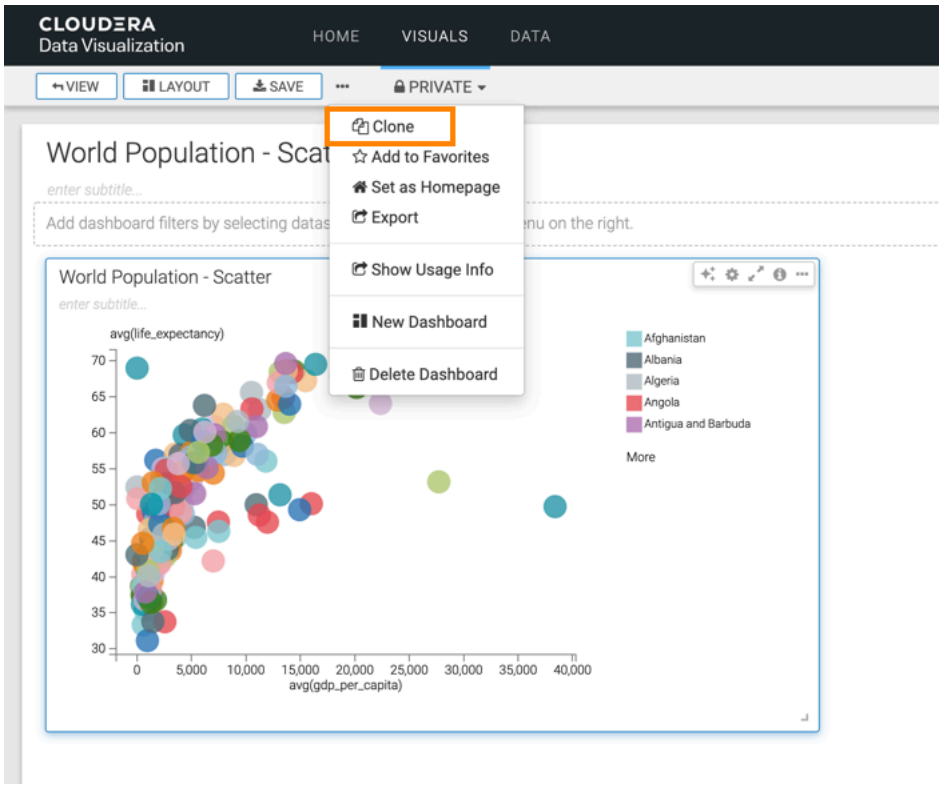

2. Under Dimensions, select un_region and drag it to the X Axis shelf.
3. Click REFRESH VISUAL.

   The Scatter visual refreshes, but this time you can see that it is divided into five separate graphs, that correspond to the identified UN Regions: Africa, America, Asia, Europe, and Oceania. There is a single animation control at the top, so when you click Play, all UN Regions animate simultaneously.

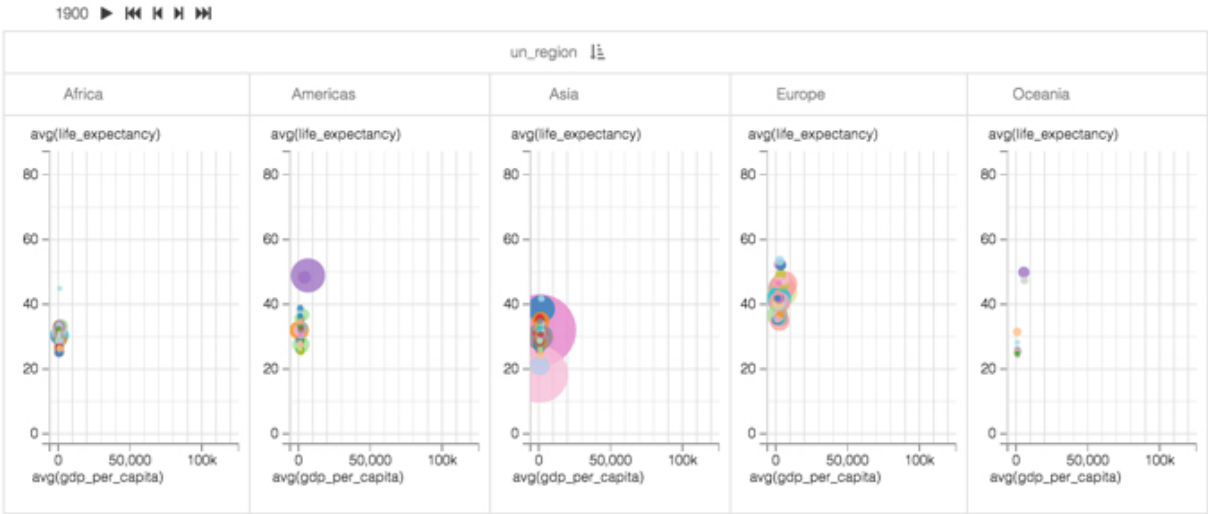

After running the animation, you will notice that the horizontal axes extend far beyond the necessary space for some of the areas. In particular, the GDP per Capita scale seems too large for Africa.

4. To have each graph use its own optimized scale for the horizontal axis, enable the independent x-axis option, as described in *Enabling independent axis scale*, and click REFRESH VISUAL.

   You can see that each area has its horizontal scale.

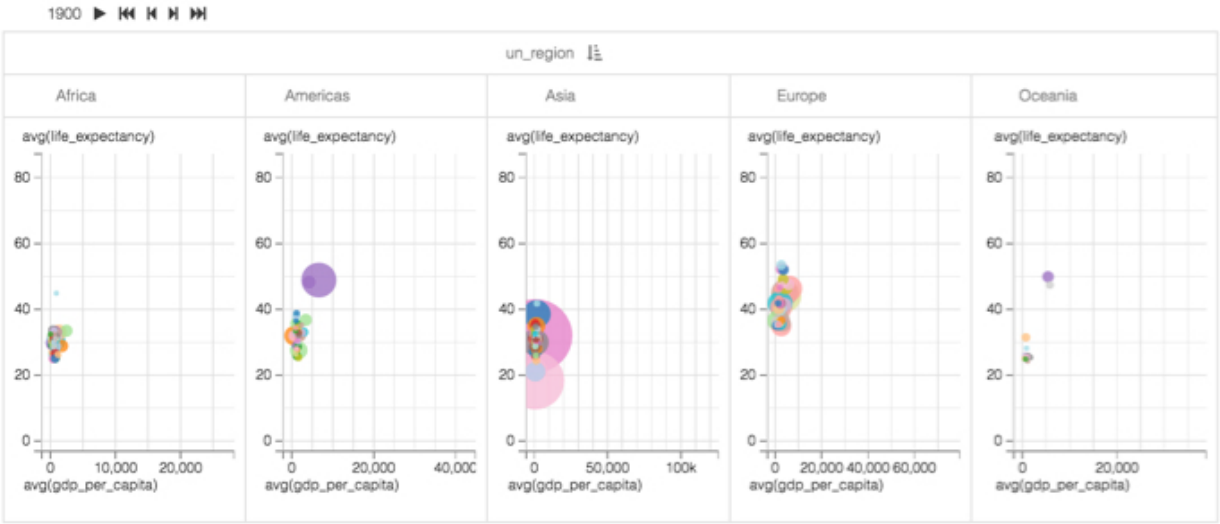

5. Similarly, choose the independent Y axis option. The differences in vertical scaling are slight, but still noticeable.

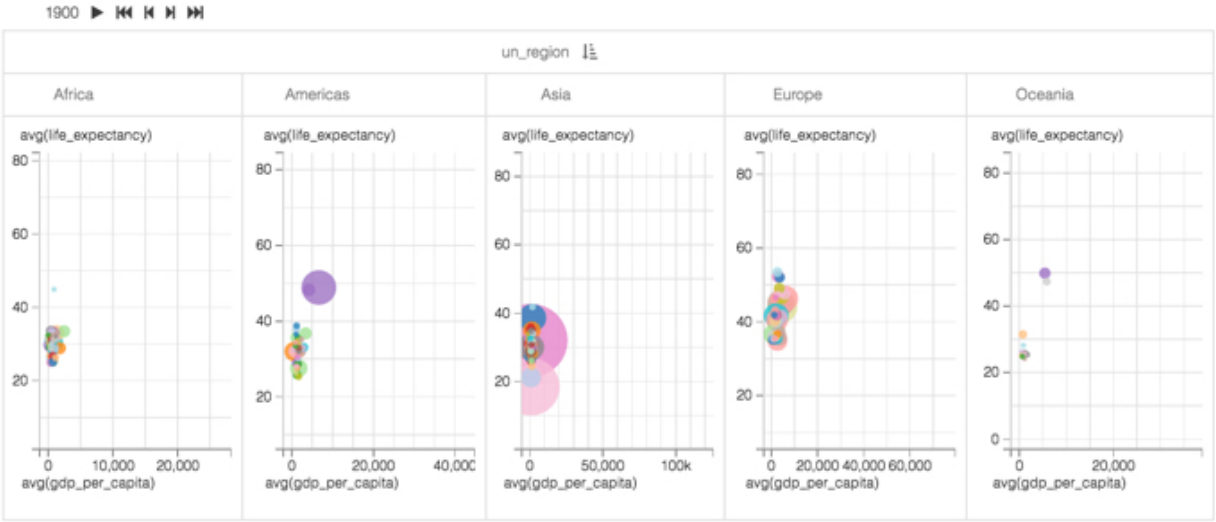

6. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'World Population - Scatter with Transition and Trellis'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

7. At the top left corner of the Dashboard Designer, click SAVE.

Shelves for scatters

Overview of shelves for Cloudera Data Visualization Scatter visuals.

A Scatter visual has the following shelves:

X Axis

This is conceptually the X axis. Use a Dimensions field from the DATA menu to plot the scatter horizontally, or a Measures field to plot the scatter vertically.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in columns.

Y Axis

This is conceptually the Y axis. Use a Measures field from the DATA menu to plot the scatter horizontally, or a Dimensions field to plot the scatter vertically.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in rows.

Colors

To visually differentiate bubbles by category, add a Dimensions field to this shelf.

This shelf accepts at most one field.

Size

Add a Measures field from the DATA menu to show an additional measurement as a relative size of the bubbles on the scatter graph.

This shelf accepts a single field.

Transition

Add Dimensions or Measures fields from the DATA menu to animate the graph based on the sequence defined by this field.

A good example would be time fields, such as timestamp, day, month, and so on. Cloudera recommends that you apply a sequential sort to ensure a natural transition.

This shelf accepts a single field.

Tooltips

Add Measures fields from the DATA menu for additional information in the visual's tooltip.

This shelf accepts multiple fields.

Filters

Add Dimensions or Measures fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Pie

Cloudera Data Visualization enables you to create Pie charts.

About this task

Pie charts are circular statistical graphics that show numerical proportion through relative size of slices; the arc length of each slice is proportional to the quantity it represents. In Cloudera Data Visualization, pie charts use a default 'doughnut' representation. However, you can change them to the more traditional closed pie configuration by controlling the arc width of the circle, as described in *Changing arc width*.

The following steps demonstrate how to create a new Pie visual on the Capital Expenditure by Industry dataset. You must first create this dataset, following the steps described in *Creating datasets*.

For an overview of shelves that specify this visual, see *Shelves for pie visuals*.

Note: This visual type supports smooth rendering of streaming data.

Procedure

1. Start a new visual based on the Capital Expenditure by Industry dataset.
   For instructions, see *Creating a visual*.
2. In the VISUALS menu, find and click Pie.

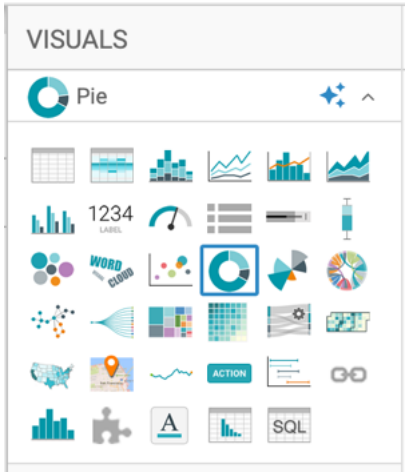

The shelves of the visual changed. They are now Dimensions, Measures, Tooltips, X Trellis, Y Trellis, and Filters. Both Dimensions and Measures are mandatory.

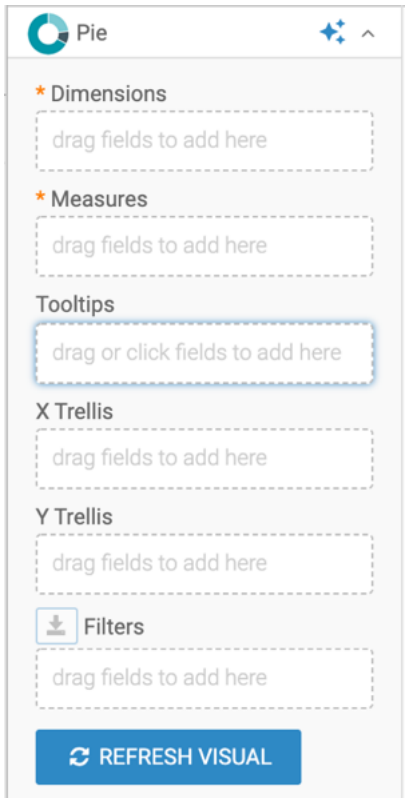

3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Under Dimensions, select industry and drag it to the Dimensions shelf.
   b) Under Measures, select expenditure and drag it to the Measures shelf.

4. Click REFRESH VISUAL.

   The Pie visual appears.

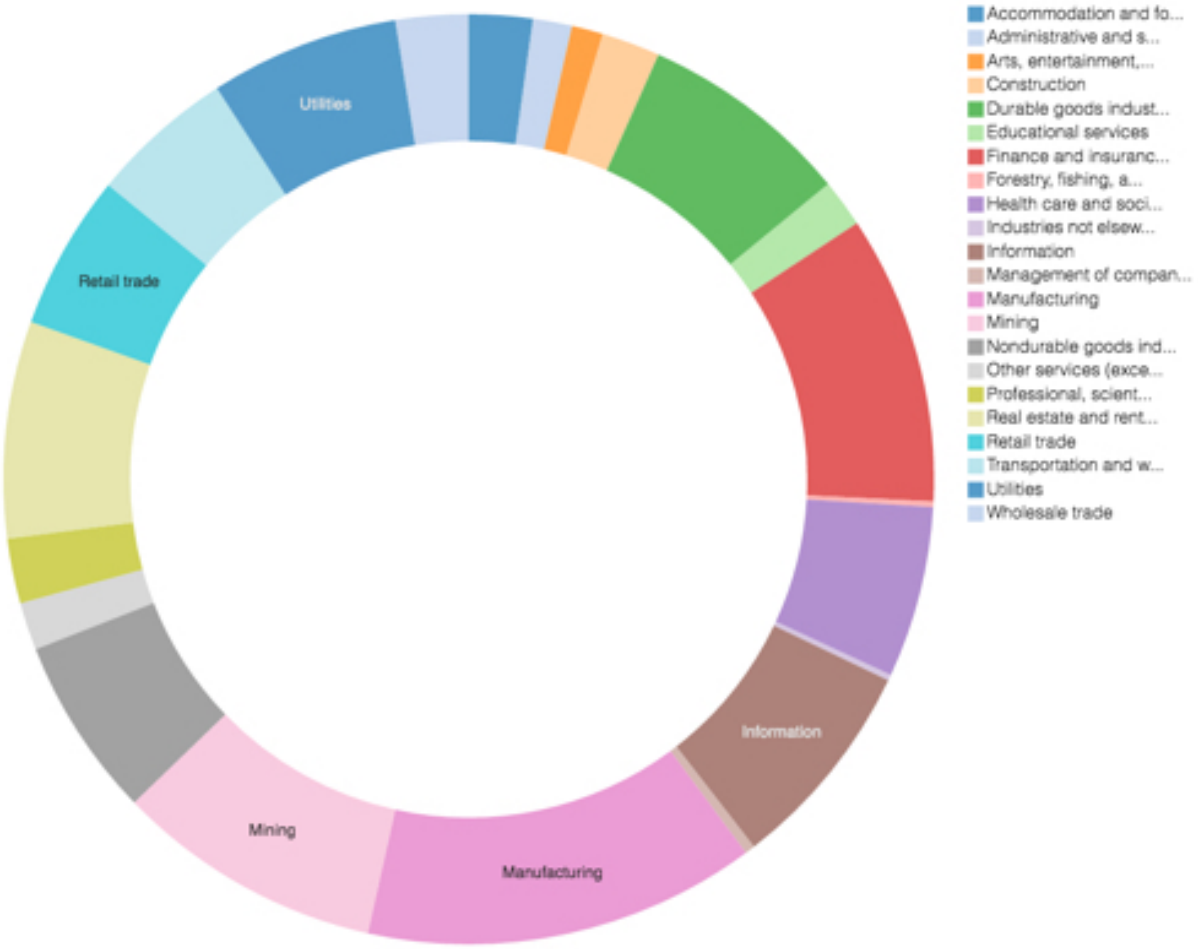

5. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'Capital Expenditures - Pie'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

6. At the top left corner of the Dashboard Designer, click SAVE.

7. Adjust the arc width to close the hole in the middle of the 'doughnut'.
   a) From the visual Settings menu, select the Marks tab, and set Arc Width to 0.

   The default arc width is 80.

Figure 4: Closing the Hole; Changing Arc Width from 80 to 0

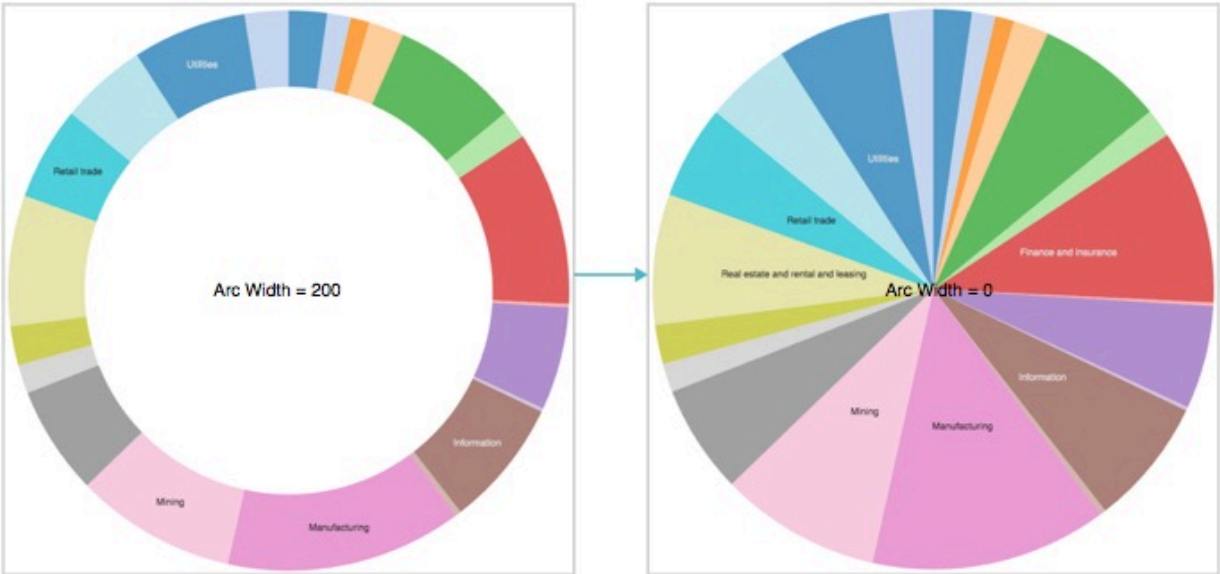

You can also experiment with changing the thickness of the 'doughnut'. See *Changing arc width* for details on how to adjust the thickness of the 'doughnut'.

Figure 5: Contrasting Arc Widths: 20 and 200

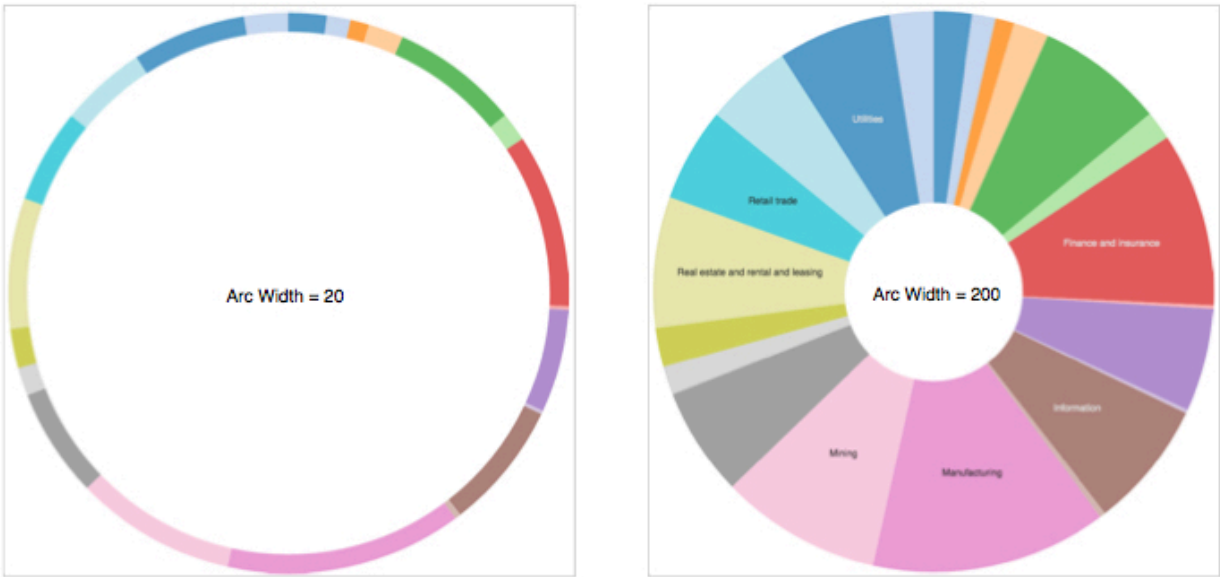

8. You can adjust the default percentage reporting inside the Tooltip.
   a) From the visual Settings menu, select the Tooltip.
   b) Change the Label for Percentage to 'Percent of Total' and the Decimals included in percentage to '3'.

Figure 6: Changing Tooltip Percentage Reporting Options

To remove the default percentage reporting, unselect the default Include percentage of total in tooltip.

Figure 7: Removing Percentage Reporting from Tooltip

For more information, see *Customizing tooltip*.

Shelves for pies

Overview of shelves for Cloudera Data Visualization Pie visuals.

A Pie visual has the following shelves:

Dimensions

Add a Dimension field from the DATA menu to create Pie chart segments for each value in that dimension.

This shelf accepts a single field.

Measures

Add a Measure field from the DATA menu to aggregate this measurement for each segment of the Pie chart.

This shelf displays only one field on the visual. Additional fields appear only in the Tooltip.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimension field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Radial

In Cloudera Data Visualization, Radial charts are a good approach for comparing the relative values of several measures. These measures are assigned an equal angle of rotation, and the relative values are represented through the areas of each wedge.

About this task

The following steps demonstrate how to create a new Radial visual representation on the US County Population dataset, for which the data source is samples.us_counties]. For an overview of shelves that specify this visual, see *Shelves for radials*.

Procedure

1. Start a new visual based on the US County Population dataset (data source samples.us_counties).

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Radial.

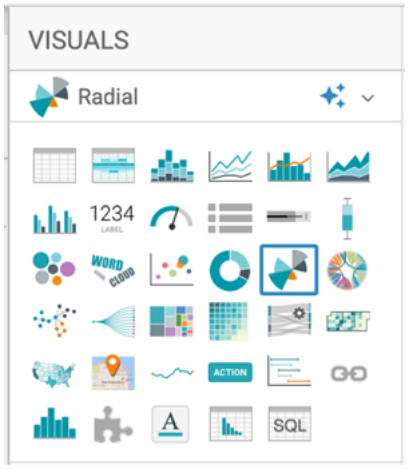

The shelves of the visual changed. They are now Dimensions, Measures, Tooltips, X Trellis, Y Trellis, and Filters. Both Dimensions and Measures are mandatory.

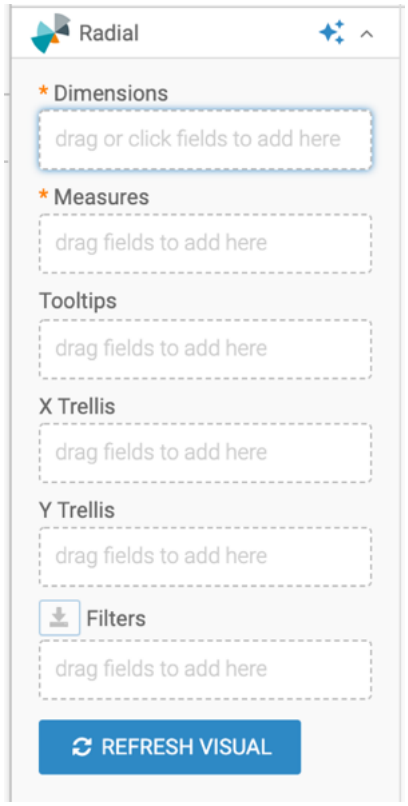

3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Under Dimensions, select ctyname, and add it to the Dimensions shelf.
   b) Under Measures, select the following and add them all to the Measures shelf: wa_male, ba_male, ia_male, aa_male, na_male and tom_male.

   Now all fields on the Measures shelf show the sum of male populations.

4. To show both male and female (total) in the group, change the calculations for the fields: change the expression sum([wa_male] to (sum([wa_male])+sum([wa_female])), and so on.

   The following steps demonstrate how to use the Expression Editor:

   a) On the shelf of the visual, click the (caret-right) icon to the right of the field.
   b) In the FIELD PROPERTIES menu, click [ ] Enter/Edit Expression.
   c) In the Enter-Edit Expression modal window, edit the text manually or pull in the appropriate functions and fields.
   d) Click VALIDATE EXPRESSION to ensure that the computation works, and then click SAVE.

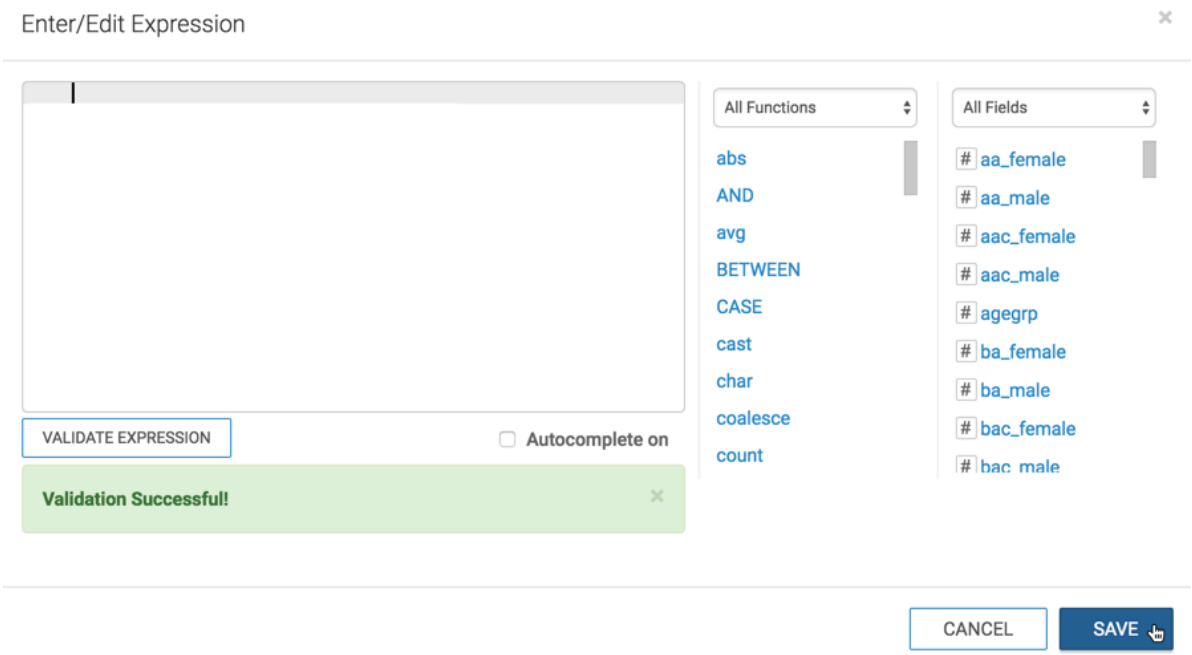

   e) Click the x icon at the top of the FIELD PROPERTIES menu to close it.

   The column in the shelf now appears to hold the expression.

5. To improve the appearance of the visual, alias ctyname as County on the Dimensions shelf, and all the fields on the Measures shelf according to the following:

   - (sum([wa_male])+sum([wa_female])) as White
   - (sum([ba_male])+sum([ba_female])) as Black or African American
   - (sum([ia_male])+sum([ia_female])) as American Indian and Alaska Native
   - (sum([aa_male])+sum([aa_female])) as Asian
   - (sum([na_male])+sum([na_female])) as Native Hawaiian and Other Pacific Islander
   - (sum([tom_male])+sum([tom_female])) as Two or More Races

   Follow the steps to create an alias:

   a) On the shelf of the visual, click the caret-right icon to the right of the field.
   b) In the FIELD PROPERTIES menu, click the caret-right icon next to Alias.
   c) In the text box below Alias, enter the alias name of column, as it should appear in the visual.
   d) Click the x icon at the top of the FIELD PROPERTIES menu to close it.

   The shelf now shows the column with its alias name.

6. Click REFRESH VISUAL.

7. Note all the information provided by the visual, for each county. Also, note how long the visual is, as it lists every county in the nation.

8. Filter on the stname field to see a smaller visual, or trellis the visual on the stname field.

   For instructions, see *Trellised radial*.

9. Adjust the axes depending on what your visual demonstrates: comparison of measures within same dimension, relative values of measure compared to same measure across all dimensions, or relative values of measures compared to all measures across all dimensions.

   For instructions, see *Changing the axes scale for radial graphs*.

10. You can replace the default 'wedge' representation of the radial chart with complete circles by enabling the overlap option on the Marks menu.

    For instructions, see *Changing display to overlap*.

11. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

In this example, the title is changed to 'Populations, by County - Radial'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

12. At the top left corner of the Dashboard Designer, click SAVE.

Trellised radial

In Radial visuals, Cloudera Data Visualization enables you to add trellises for comparing and contrasting data.

In the following example in the US County Population dataset, the field stname (aliased as State) is used as a trellising partition. Placed on the X Trellis shelf , it creates a trellised version of a radial visual you developed in *Radials*.

This is the shelves set-up for the trellised visual:

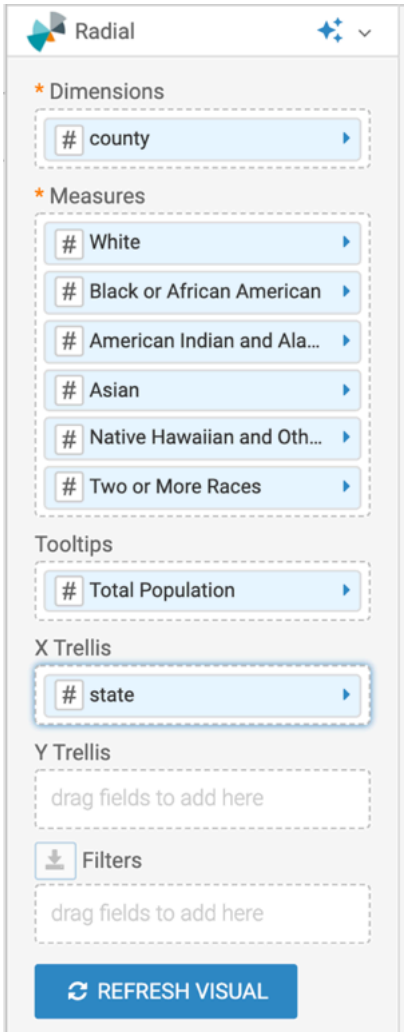

The trellising organizes the individual radial charts into columns, by State. Within each column, the charts appear by alphabetical name of the County.

Figure 8: Trellised Radial Visual

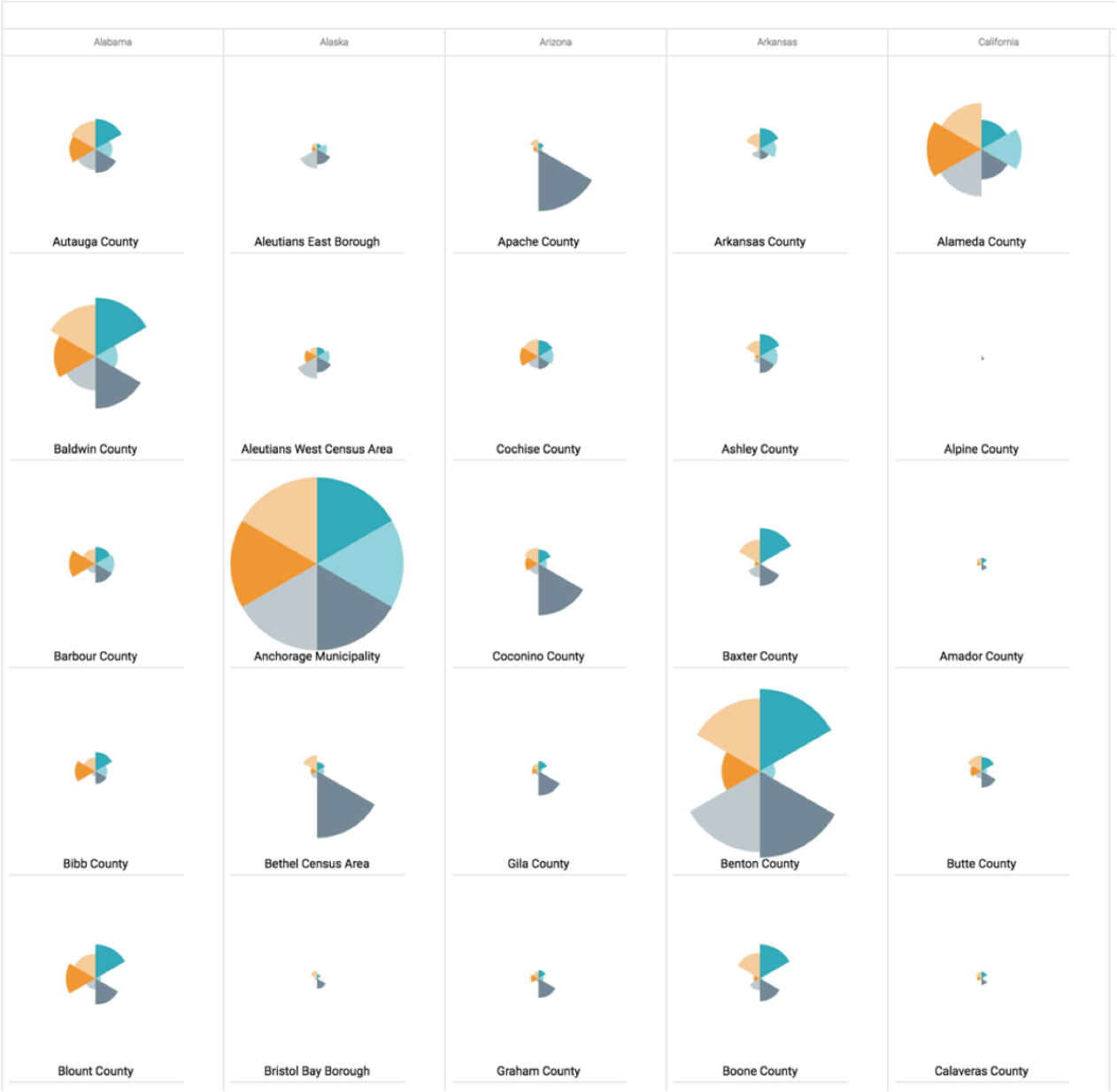

Shelves for radials

Overview of shelves for Cloudera Data Visualization Radial visuals.

A Radial visual has the following shelves:

Dimensions

Add a Dimensions field from the DATA menu to create radials for each value in that dimension.

This shelf accepts a single field.

This is a mandatory shelf.

Measures

Add a Measures field from the DATA menu to aggregate this measurement for each radial of the visual.

This shelf displays only one field on the visual. Additional fields appear only in the Tooltip.

This is a mandatory shelf.

Tooltips

Add Measures fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimensions field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimensions field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimensions or Measures fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Chord

Cloudera Data Visualization enables you to create Chord visuals. These visuals expose the inter-relationships between entities, making them an ideal choice for comparing the similarities within a dataset, or between different groups of data. The nodes/dimensions are on the circumference of a circle, and the relationships between them are the arcs inside the circle. The size of the node is proportional to the measurement. Cloudera Data Visualization uses color to help distinguish the overlapping relationship arcs.

About this task

Chord visuals are similar to Correlation Heatmaps because they both compare exactly two dimensions. Chords, however, plot dimensions that cover the same value domain, and are better at pointing out the presence of a relationship between nodes than the relative magnitude of that relationship.

The following steps demonstrate how to create a new chord visual on the SF Cabrides dataset. This dataset is based on data previously imported into Cloudera Data Visualization from a datafile. Each distinct line segment between two dimensions reports the correlation data. Because the values for pick-up and drop-off areas, srcarea and tgtarea , are in the same domain, it makes the dataset ideal for demonstrating chord visuals.

For an overview of shelves that specify this visual, see *Shelves for chord visuals*.

Procedure

1. Start a new visual based on the SF Cabrides dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Chord.

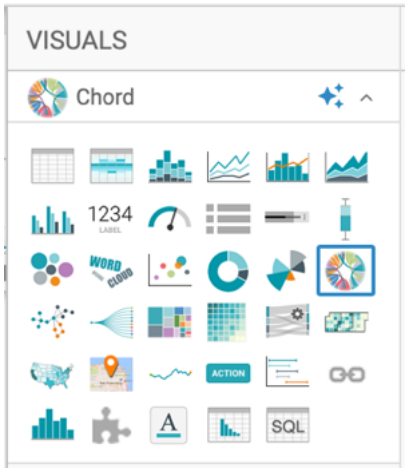

The shelves of the visual changed. They are now Dimensions, Measure, Tooltips, X Trellis, Y Trellis and Filters. Both Dimensions and Measure are mandatory.

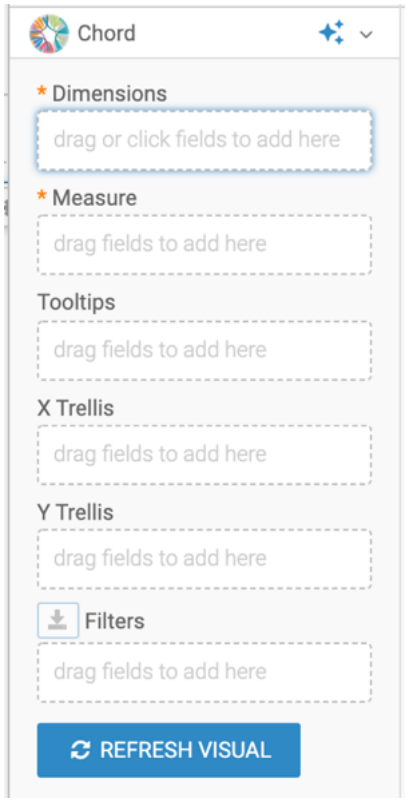

3. Populate the shelves from the fields available in the DATA menu:
   a) From Dimensions, select srcarea and trgarea, and drag them to the Dimensions shelf.
   b) From Measures, select rides and place it to the Measure shelf.

4. Click REFRESH VISUAL.

   The Chord visual appears.

   Figure 9: Chord Visual

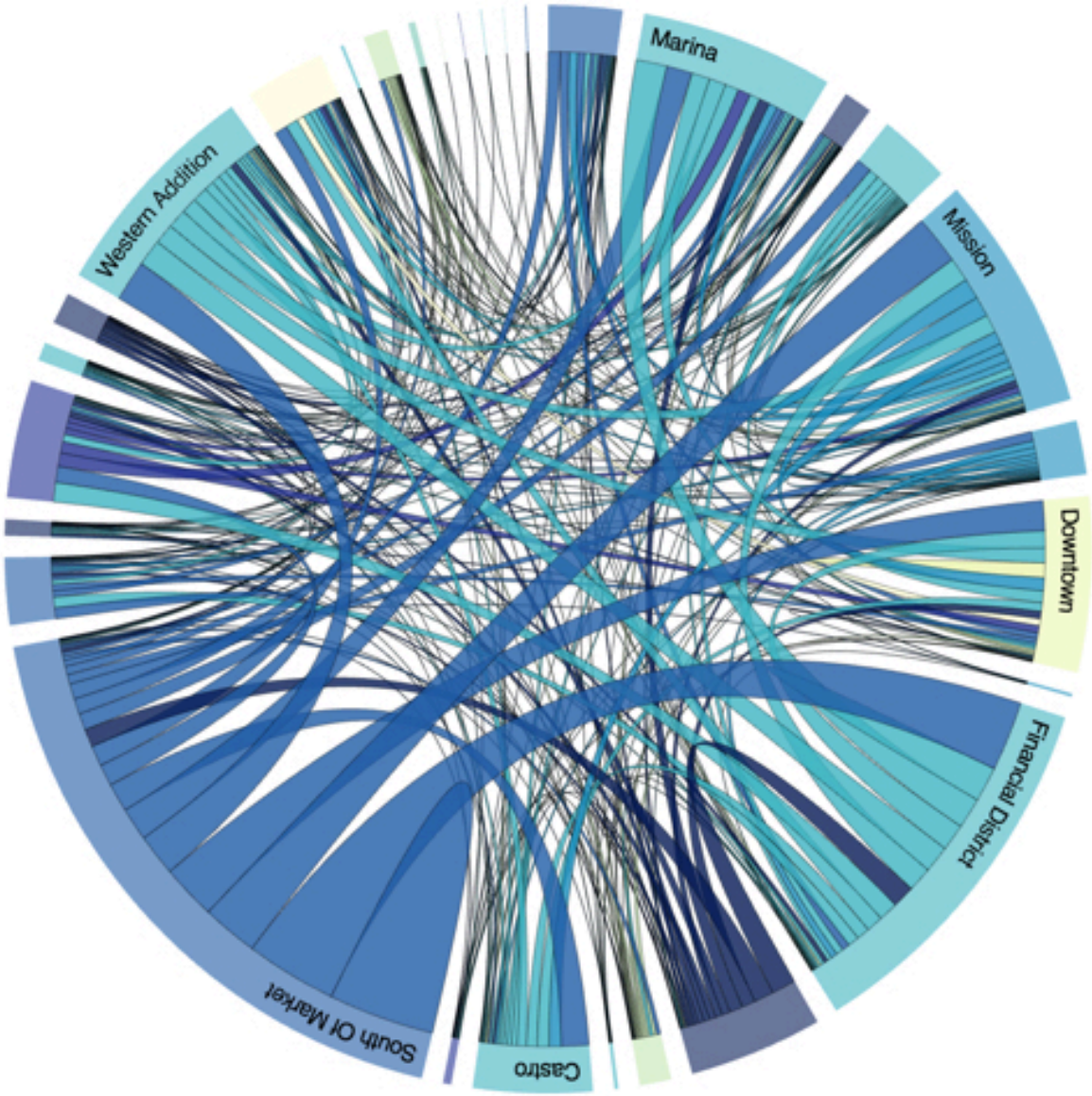

5. Check the details of the individual dimensions:
   a) Hover the pointer over one of the nodes.

      The visual shows the connections that originate or terminate with this specific value, as well as the tooltip. In the following image, you can see all connections for 'South of Market'.
   b) Hover the pointer over one of the connections.

      The tooltip shows the detail for that connection: starting and ending notes, and the measurement.

Figure 10: Viewing Details of a Chord Visual

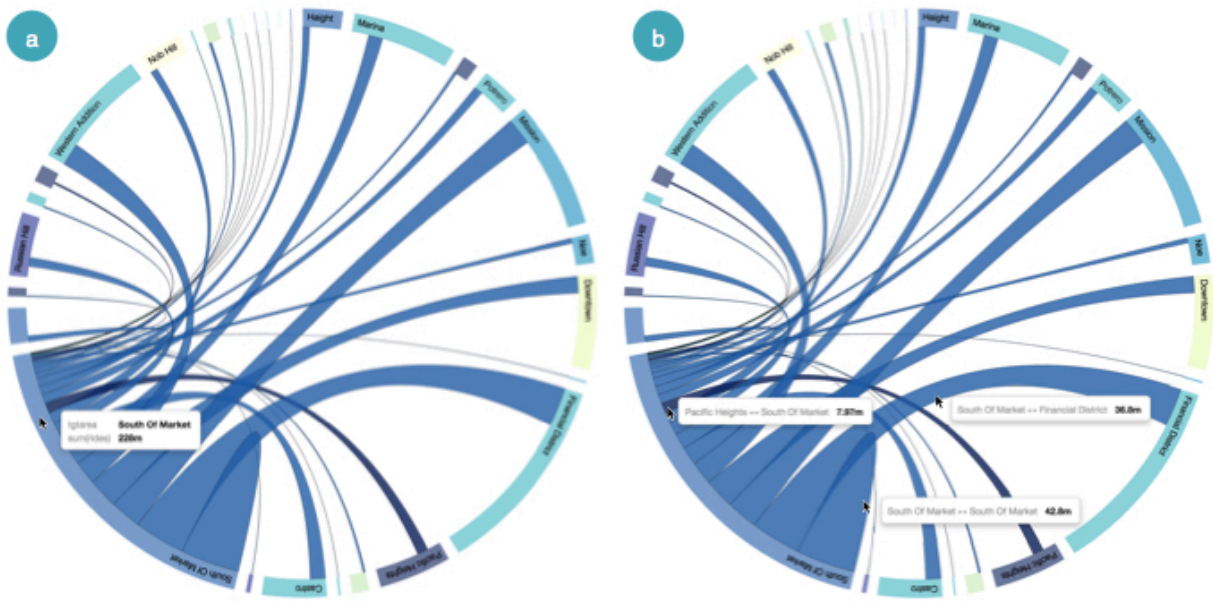

6. Click EDIT next to the title of the visualization, and change the title to 'San Francisco Cab Rides - Chord'.
7. At the top left corner of the Dashboard Designer, click SAVE.

Shelves for chords

Overview of shelves for Cloudera Data Visualization Chord visuals.

A Chord visual has the following shelves:

Dimensions

Add two Dimensions fields from the DATA menu to define the start and end points of each connection in the plot.

This shelf accepts exactly two fields.

This is a mandatory shelf.

Measure

Add a Measures field from the DATA menu to aggregate this measurement for each chord of the chart. The aggregate measurement determines the thickness of the chord connector between each pair of dimensions.

This shelf accepts a single field.

This is a mandatory shelf.

Tooltips

Add Measures fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimensions field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimensions or Measures fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Network

In Cloudera Data Visualization, Network visuals can be used to represent many systems and structures. The implementation is dynamic, with ongoing optimization adjustments that implement a force layout based on the Barnes–Hut approximation. It also includes many manual setting adjustments.

About this task

In the following example, the NCAA Basketball dataset is used for years 1985 through 2018.

For an overview of shelves that specify this visual, see *Shelves for networks*.

Procedure

1. Start a new visual based on the NCAA Basketball dataset.

   For instructions, see *Creating a visual*.
2. In the VISUALS menu, find and click Network.

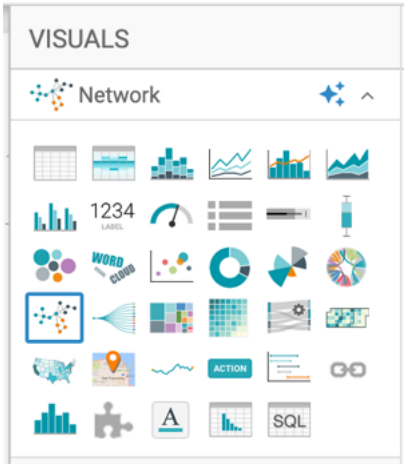

The shelves of the visual changed. They are now Nodes, Link Width, Link Color, Source Node Size, Target Node Size, Source Node Color, Target Node Color, Tooltips, X Trellis, Y Trellis, and Filters. There is only one

mandatory shelf: Nodes, which adds two dimensions from the data menu to specify the start and end of each network connection in the plot.

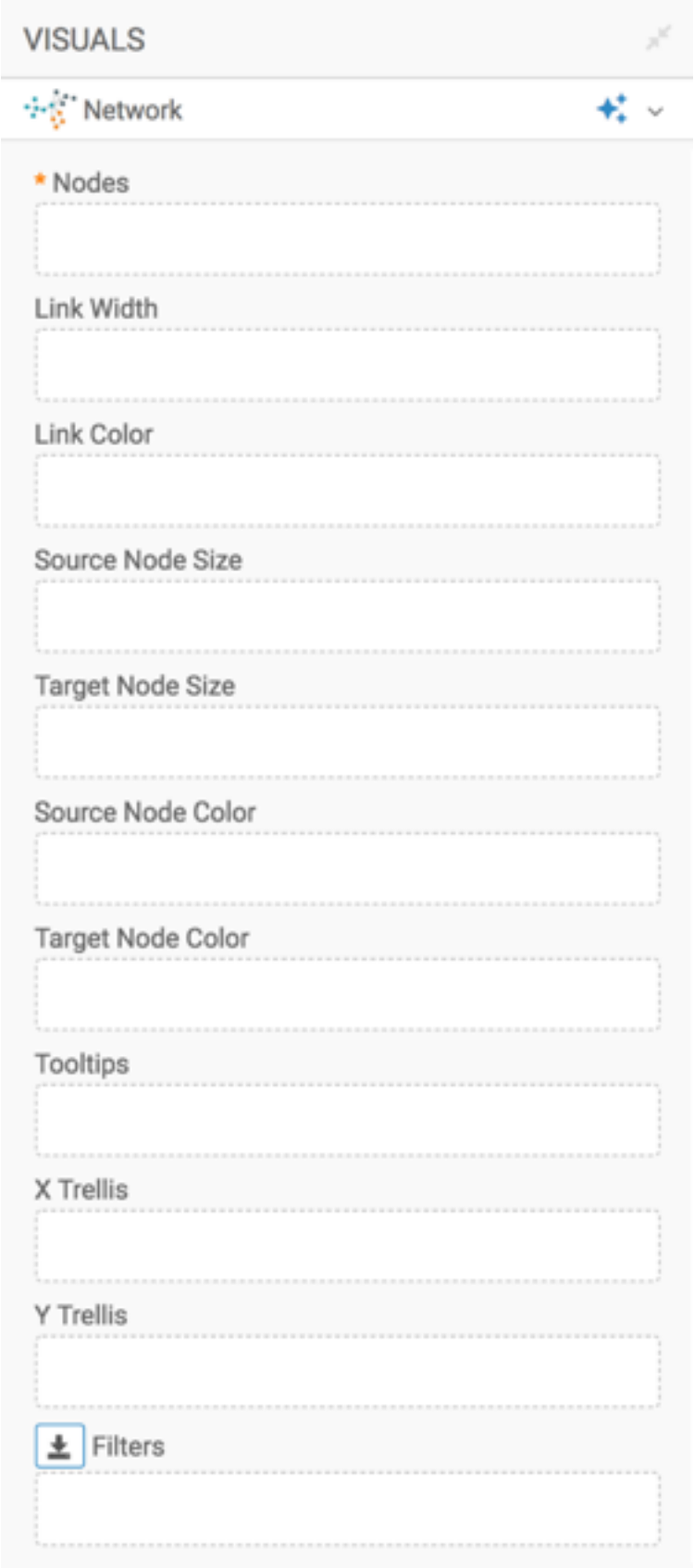

To learn more about the shelves of this visual, see *Shelves for networks*.

3. Populate the shelves from the fields available in the DATA menu:

a) Under Dimensions, select Team 1 and Team 2, and add them to the Nodes shelf.

b) Under Measures, select Year and add it to the Filters shelf. Select the years 2015, 2016, and 2017.

c) Add Round to the Filters shelf, and select 1.

4. Click REFRESH VISUAL.

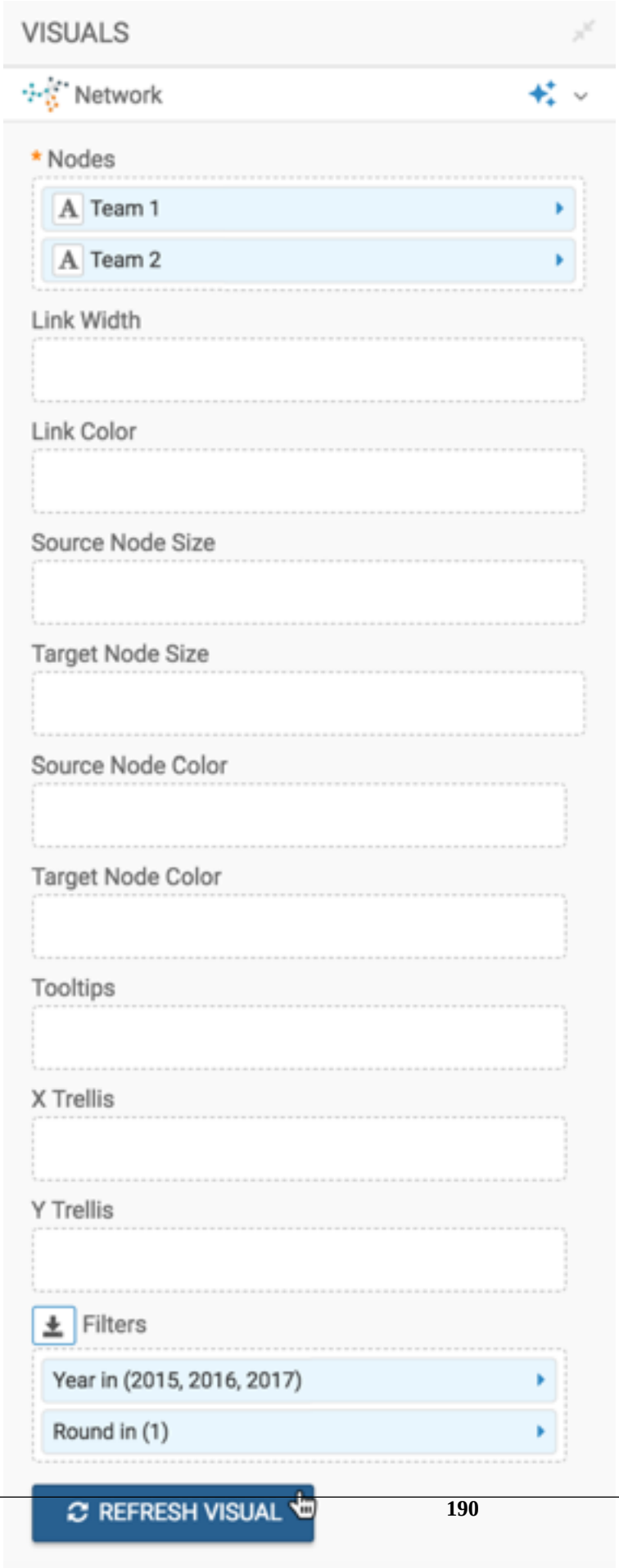
VISUALS
Network
* Nodes
Team 1
Team 2
Link Width
Link Color
Source Node Size
Target Node Size
Source Node Color
Target Node Color
Tooltips
X Trellis
Y Trellis
Filters
Year in (2015, 2016, 2017)
Round in (1)
REFRESH VISUAL

The new network visual appears. You can see that this visual shows 1:1 pairing, because round 1 of play represents 1:1 elimination.

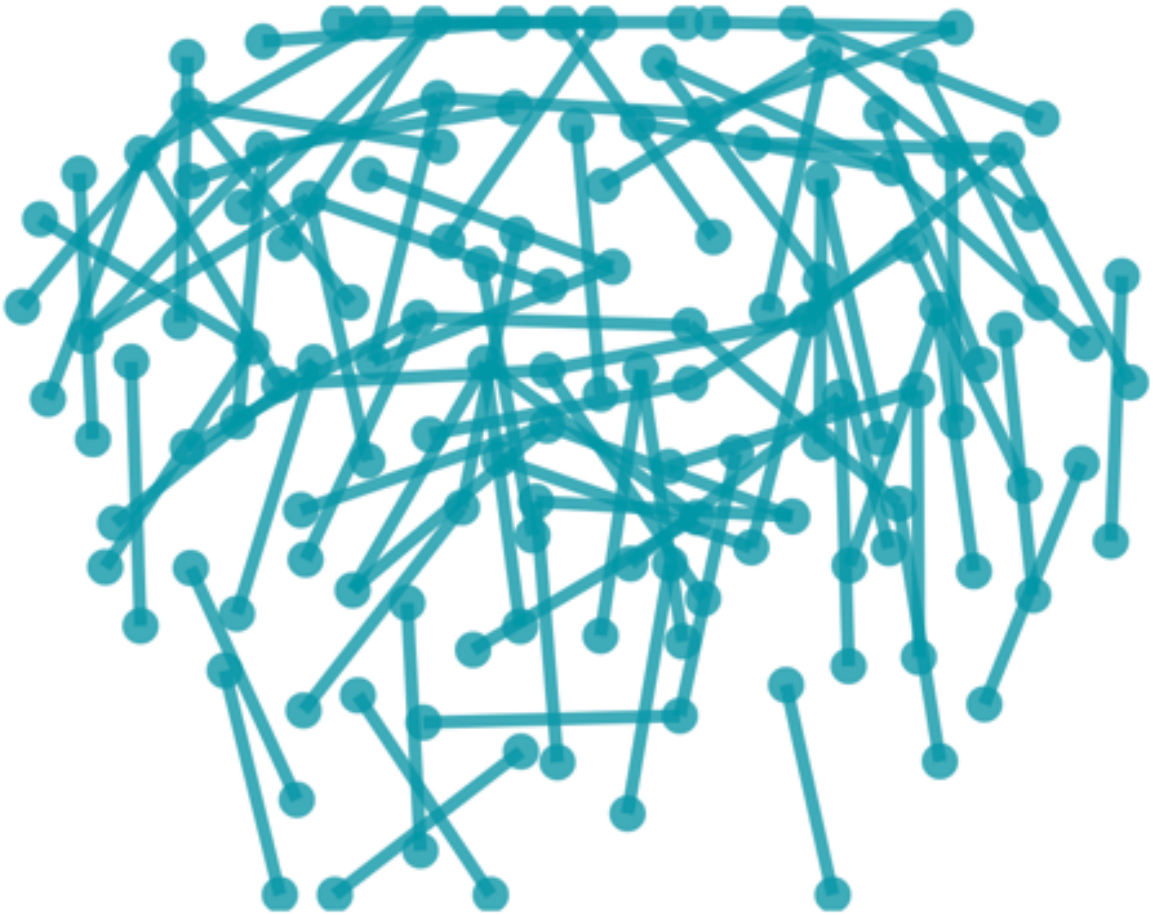

5. To show the years separately, place the Year field on the Y Trellis shelf and click REFRESH VISUAL.
   The trellised version of the network visual appears.

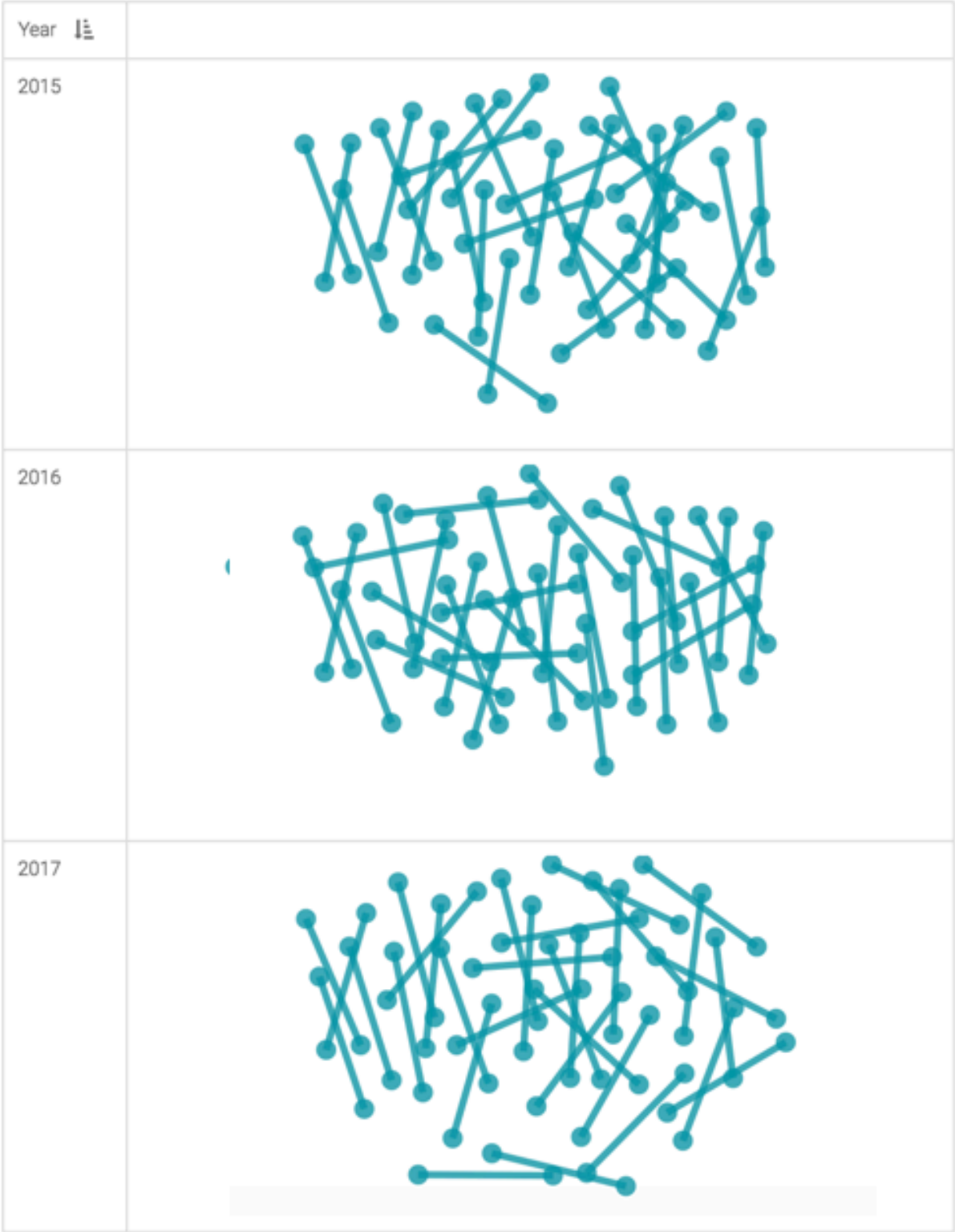

6. Change the title to Round 1, 2015-2017.
   - Click (pencil icon) next to the title of the visualization to edit it, and enter the new name.
   - [Optional] Click (pencil icon) below the title of the visualization to add a brief description of the visual.

7. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'Round 1, 2015-2017'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
8. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

To adjust the network display, check all the available settings for this visual.

For more information, see *Network visuals: same source and target data values* and *Advanced options for network visuals*. These topics demonstrate some of the many customizations that make the network visual an extremely versatile analysis tool.

Shelves for networks

Overview of shelves for Cloudera Data Visualization Network visuals.

A Network visual has the following shelves:

Nodes

Add two Dimensions fields from the DATA menu to define the start- and end-points of each connection in the plot.

This shelf accepts exactly two fields.

This is a mandatory shelf.

Link Width

Add a Measures field from the DATA menu to aggregate this measurement for each connection of the chart. The aggregate measurement determines the thickness of the connector between each pair of dimensions.

This shelf accepts a single field.

Link Color

Add a Measures or Dimensions field from the DATA menu to color the links of the plot.

This shelf accepts a single field.

Source Node Size

Add a Measures field from the DATA menu to show an additional measurement as a relative size of the source node.

This shelf accepts a single field.

Target Node Size

Add a Measures field from the DATA menu to show an additional measurement as a relative size of the target node.

This shelf accepts a single field.

Source Node Color

Add a Measures or Dimensions field from the Data menu to color the source node of the plot.

This shelf accepts a single field.

Target Node Color

Add a Measures or Dimensions field from the DATA menu to color the target node of the plot.

This shelf accepts a single field.

Tooltips

Add Measures fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimensions field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimensions field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimensions or Measures fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Advanced options for networks

Network visuals enable you to show a large amount of supplemental information directly by using the optional shelves of the visual, and the various settings for Marks, Nodes, and Links.

Networks: same source and target data values

Network visuals can show the same node once, or several times on the same chart, depending on node settings.

About this task

In the following example the NCAA Basketball dataset is used, and the visual is based on a visual developed in *Networks*. The original visual shows only Round 1 play, or 1:1 match-ups.

Procedure

1. Clone the visual, and save it as All Rounds, 2015-2018.

   For instructions, see *Cloning a visual*.
2. Add Region Name to the X Trellis shelf.

   This shows all regions separately, and makes it easier to pick out the nodes that have multiple links.
3. Because this dataset defines the 'Final 4' and 'Championship' play as regions, add Region Name to the Filters shelf, and exclude these values.
4. To choose all rounds of play, remove the Round field from the Filters shelf.
5. Click Settings Nodes and select Source & target node data come from the same base values.

   For more information, see *Showing the same source and target values*.

6. Click REFRESH VISUAL.

   You can see the difference between the two visuals: the one showing all rounds is much denser, and shows multi-branching from some of the same nodes.

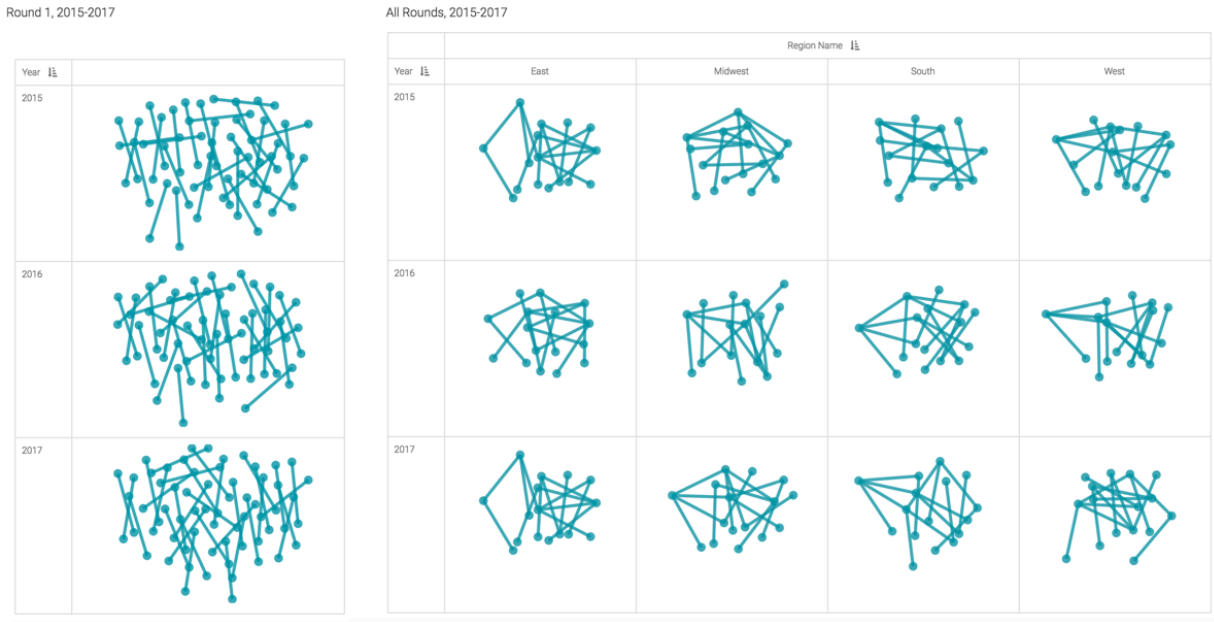

Networks: link color menu

Network visuals have many color palette options for links.

About this task

To get to the link color options, follow these steps:

Procedure

1. On the right side of the Visual Designer, click the Style menu.
2. Open the Link Colors menu and choose a color palette.
   - You can select Reverse color palette of visuals if you want to change the order of the colors.
   - You can click on an individual color swatch in the palette to change the first color within palette.
   - You can also change the color of the Direction Arrow.

Example

In this example the links palette is changed so links are more clearly distinguished from the nodes. For this visual, all rounds and all regions of the 2017 season are shown:

Figure 11: Changing color palette for links in network visual

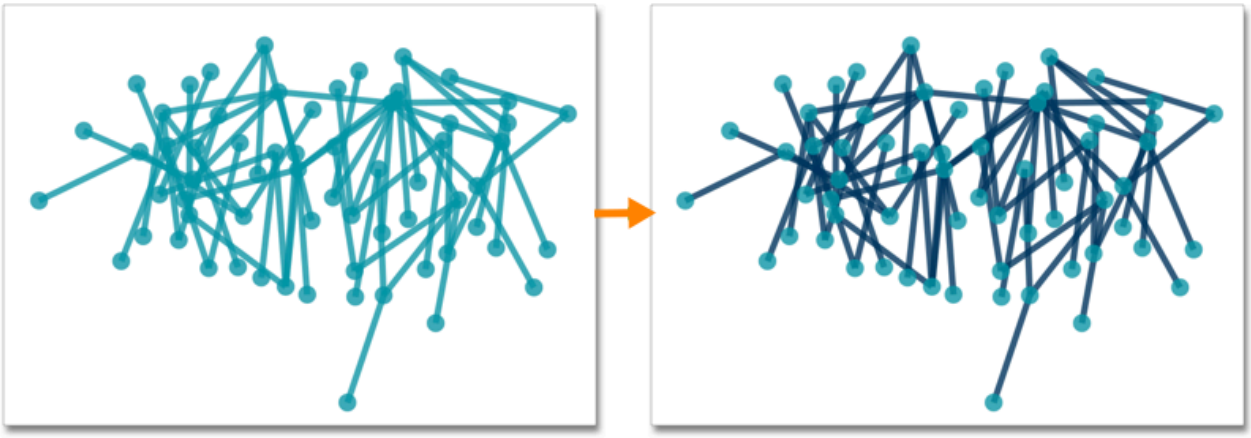

Networks: link color shelf

By default, the links of the network visual have the same thickness and length. Network visual has a data shelf called Link color that uses a specified palette to plot different link colors.

About this task

To specify the link color categories, follow these steps:

Procedure

1. Choose the appropriate palette, as described in *Networks: link color menu*.
2. Add the round field to the Link Color shelf.
3. Click the round field on the shelf to see the FIELD PROPERTIES menu.
4. In the FIELD PROPERTIES menu, select [ ] Enter/Edit Expression.
5. In the expression editor, specify the max() aggregation function, with an alias:

```
max([Round]) as 'Round'
```

6. Click VALIDATE & SAVE.
7. Click REFRESH VISUAL.

   You can see the change to the appearance of the visual:

   Figure 12: Specifying the link color shelf in a network visual

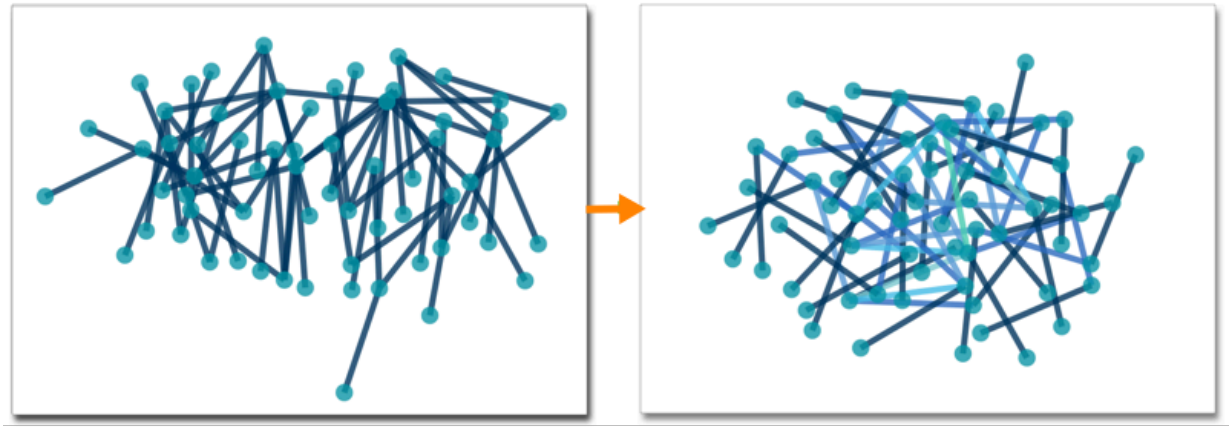

8. If you hover over a specific link, you can see the detail information. For example, the dark-colored links represent Round 1 games, and the lightest link in the middle is the final game, Round 6.

Figure 13: Tooltips on a network visual with active link color shelf

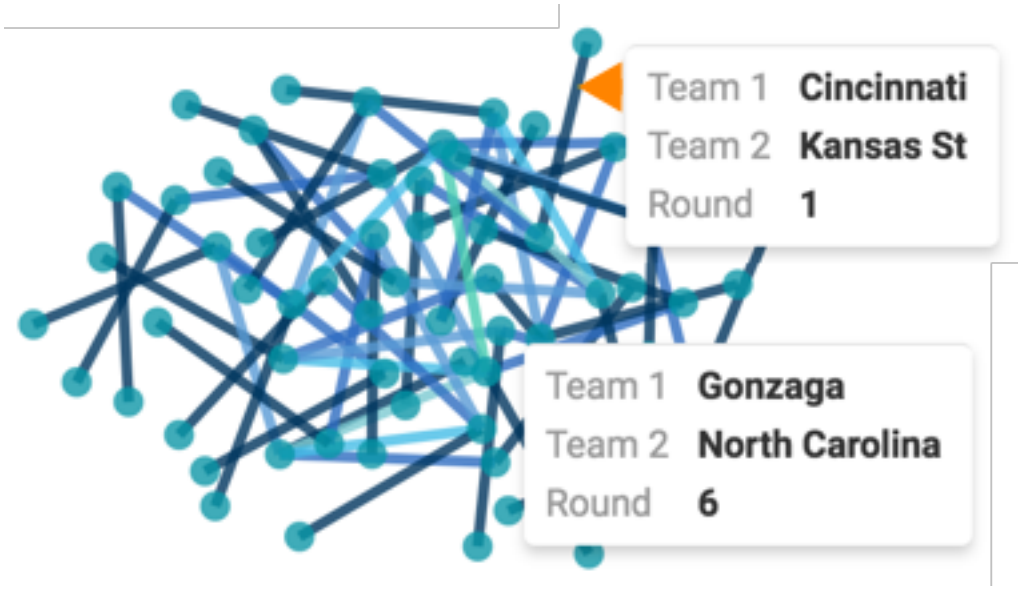

Networks: link width shelf

By default, the links of the network visual have the same thickness and length. Network visual has a data shelf called Link width that produces a chart with link thickness that depends on specified values.

About this task

To specify the width of the link, follow these steps:

Procedure

1. Add the Score 1 field to the Link Width shelf.
2. Click the Score 1 field on the shelf, to see the FIELD PROPERTIES menu.
3. In the FIELD PROPERTIES menu, select [ ] Enter/Edit Expression.
4. In the expression editor, specify a new calculation as a max() aggregation function, with an alias:

```
max(abs([Score 1] - [Score 2])) as 'Score Differential'
```

5. Click VALIDATE & SAVE.

6. Click REFRESH VISUAL.

   You can see the change to the appearance of the visual:

   Figure 14: Specifying the link width shelf in a network visual

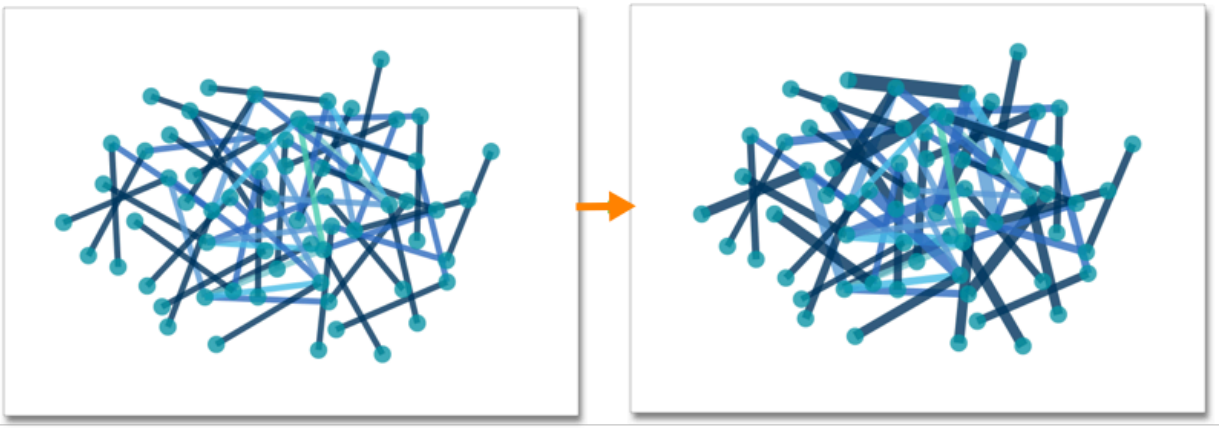

7. If you hover over a specific link, you can see the detail information.

   For example, compare the following two Round 1 games:

   - Kansas vs. UC Davis that has a 38 point score differential, and the thickest link in the chart
   - Cincinnati vs. Kansas State that has a 14 point score differential, and a comparatively thinner link

   Figure 15: Tooltips on a network visual with active link color shelf

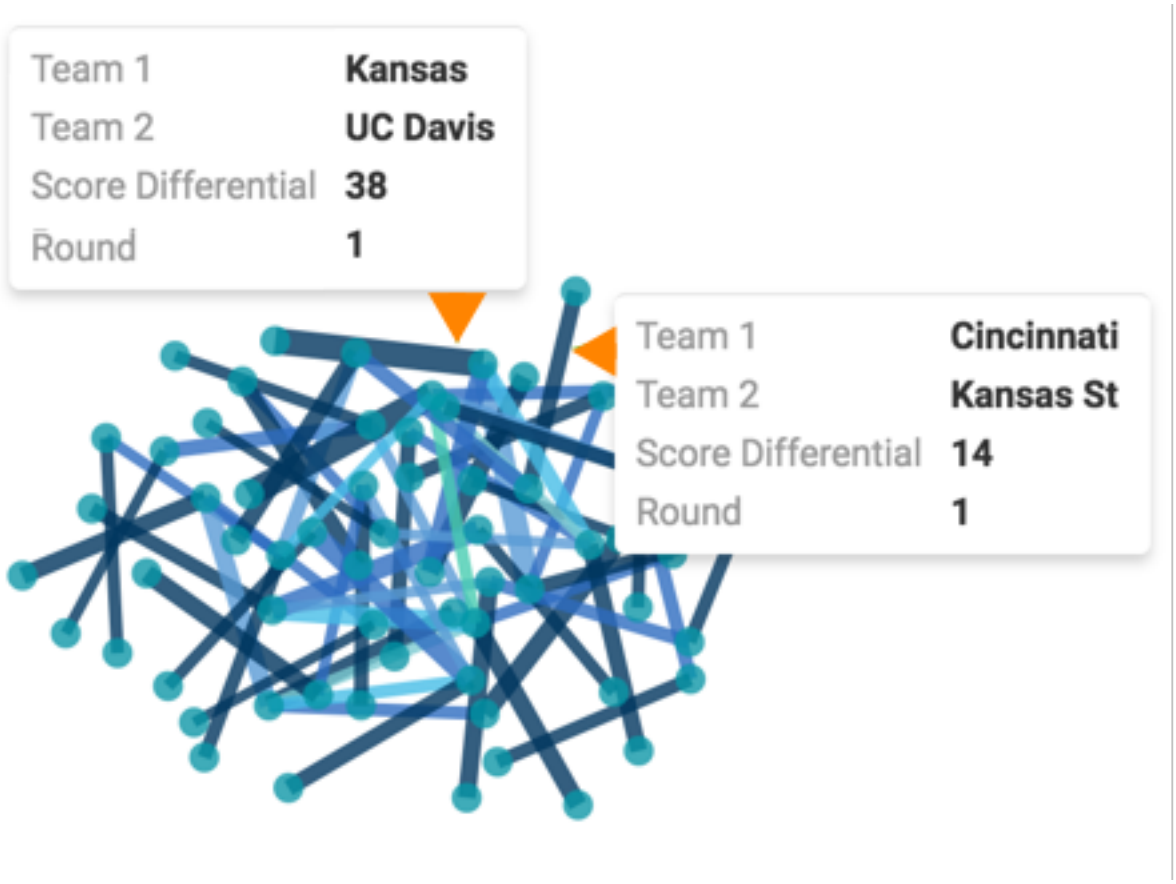

Networks: link legend

In network visuals, the legend for link colors helps you to visually identify the correct values.

To enable the link legend in network visuals, see the instructions in *Enabling the link legend*.

In the following example, a link legend is added to the visual developed in *Networks: link width shelf*.

Figure 16: Adding a legend for links in a network visual

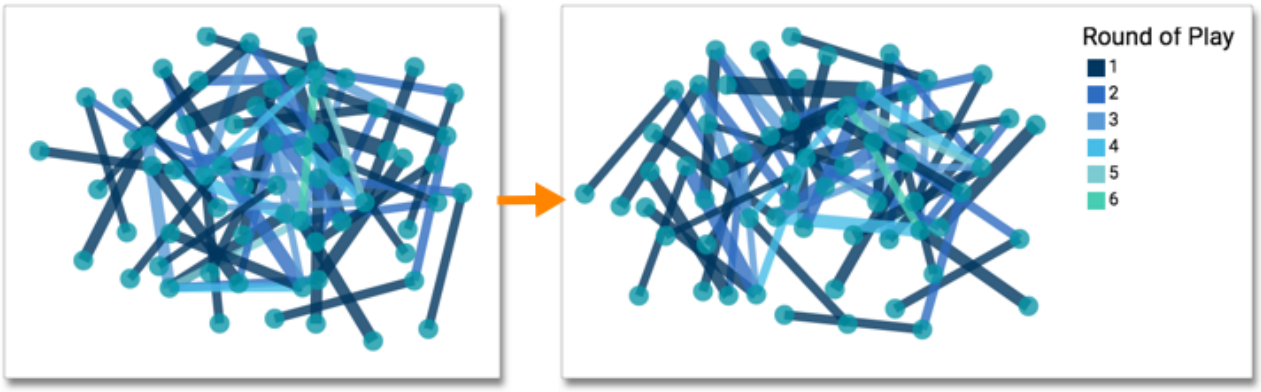

Networks: node size shelves

Network visuals enable you to change the size of the nodes.

About this task

By default, the nodes of a network visual have the same size. This visual has two data shelves, Source node size and Target node size, that produce a chart with nodes that represent values in the dataset.

To specify the size of the nodes, follow these steps:

Procedure

1. Add the Seed 1 field to the Source Node Size shelf.
2. Click the Score 1 field on the shelf, to see the FIELD PROPERTIES menu.
3. In the FIELD PROPERTIES menu, select [ ] Enter/Edit Expression.
4. In the expression editor, specify a new calculation as a max() aggregation function, with an alias:

```
max([Seed 1]) as 'Seed'
```

5. Click VALIDATE & SAVE.
6. Add the Seed 2 field to the Target Node Size shelf.
7. Click the Score 2 field on the shelf, to see the FIELD PROPERTIES menu.
8. In the FIELD PROPERTIES menu, select [ ] Enter/Edit Expression.
9. In the expression editor, specify a new calculation as a max() aggregation function, with an alias:

```
max([Seed 2]) as 'Seed'
```

10. Click VALIDATE & SAVE.

11. Click REFRESH VISUAL.

You can see the change to the appearance of the visual: the tooltips for the visual to point out the size difference between Baylor (seed 3) and Xavier (seed 11):

Figure 17: Specifying node size on a network visual

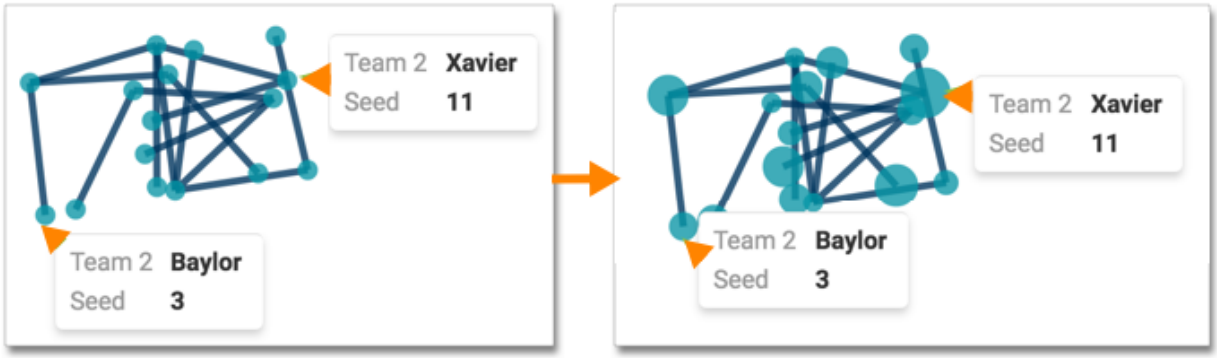

Networks: nodes color menu

Cloudera Data Visualization provides many color palette options for nodes in a Network visual.

About this task

To get to the options in the Colors menu, follow these steps:

Procedure

1. On the right side of the Visual Designer, click the Style menu.

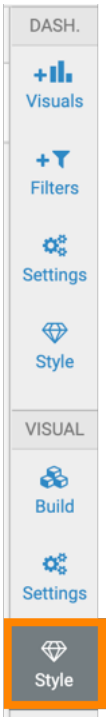

2. Open the Node Colors menu and choose a color palette. select one of the following options:
   - You can select Reverse color palette of visuals if you want to change the order of the colors.
   - You can click on an individual color swatch in the palette to change the first color within palette.
   - You can select Fix colors for specified dimension values.
   - You can add dimension values.

Networks: node color shelves

By default, the nodes of a network visual have the same color. This visual has two data shelves, Source node color and Target node color, that produce a where all nodes for a category have the same color value.

About this task

To specify the color of the nodes, follow these steps:

Procedure

1. Add the Region Name field to the Source Node Color shelf.
2. Add the Region Name field to the Target Node Color shelf.
3. For both shelves, ensure that the aggregation function is max().
4. Alias the fields on the shelves.

   In this example, the fields are named as Region Name to hide the necessary aggregation information.
5. Remove the round field from the Filter shelf.

   In this example, only one filter (Year in (2017)) is left in place.
6. Click VALIDATE & SAVE.
7. Click REFRESH VISUAL.

   You can see how much easier it is to distinguish between the nodes that represent different regions. The following example shows the tooltips for a South and West region:

   Figure 18: Specifying node color on a network visual

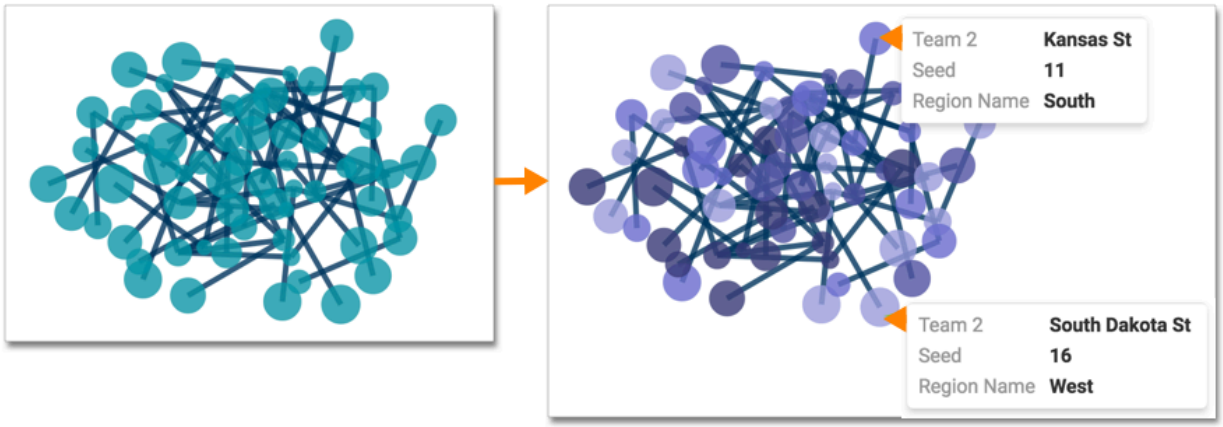

Networks: node legend

In network visuals, the legend for node colors helps you to visually identify the correct values.

To enable the node legend in network visuals, see the instructions in *Enabling the nodes legend*.

In this example, the node legend is added to the visual developed in *Networks: node color shelves*.

Figure 19: Displaying a Legend for Nodes in Network Visual

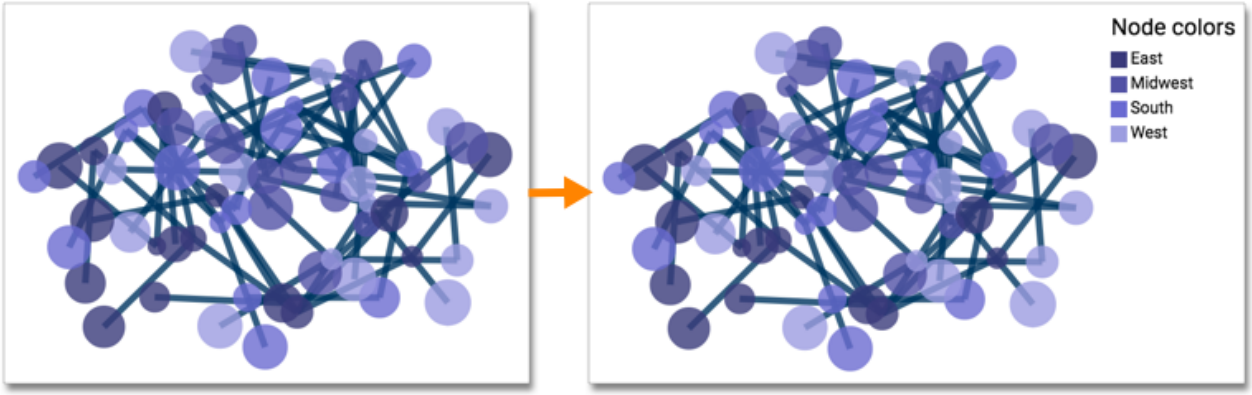

Dendrograms

Cloudera Data Visualization enables you to create Dendrogram visuals. Dendrograms are good at expressing hierarchical relationships in data. As the name derivation suggests (Greek 'dendro' for 'tree' and 'gramma' for 'drawing'), dendrograms are tree diagrams.

About this task

In the following example the NYC Taxi Data for year 2013 dataset is used. You can download a subset of this data for trips made within *Staten Island boro*, upload it, and create the dataset NYC Taxi Data.

For an overview of shelves that specify this visual, see *Shelves for dendogram visuals*.

Procedure

1. Start a new visual based on dataset NYC Taxi Data.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Dendrogram.

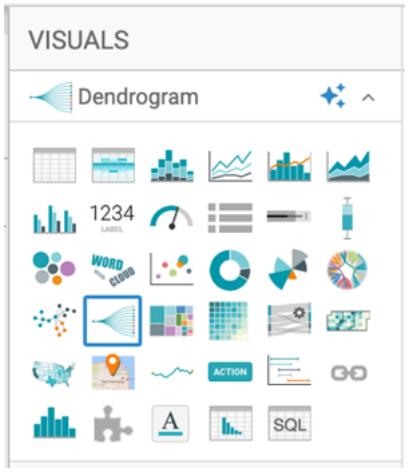

The shelves of the visual changed. They are now Dimensions, Measure, Tooltips, X Trellis, Y Trellis and Filters. The mandatory shelves are:

- Dimensions: specify two dimensions, because there are at least two nodes on each branch
- Measure; specify one measurement plotted on the branch between nodes

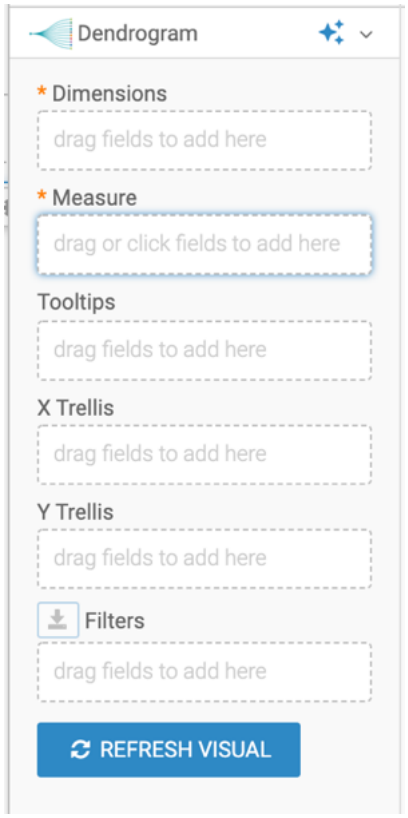

3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Under Dimensions, select Pickup Neighborhood and Dropoff Neighborhood, and add them to the Dimensions shelf.
   b) Under Measures, select Record Count and add it to the Measure shelf.

4. To examine only the rides within Staten Island, add Pickup Boro and Dropoff Boro to the Filters shelf, and set them to a single value 'Staten Island'.

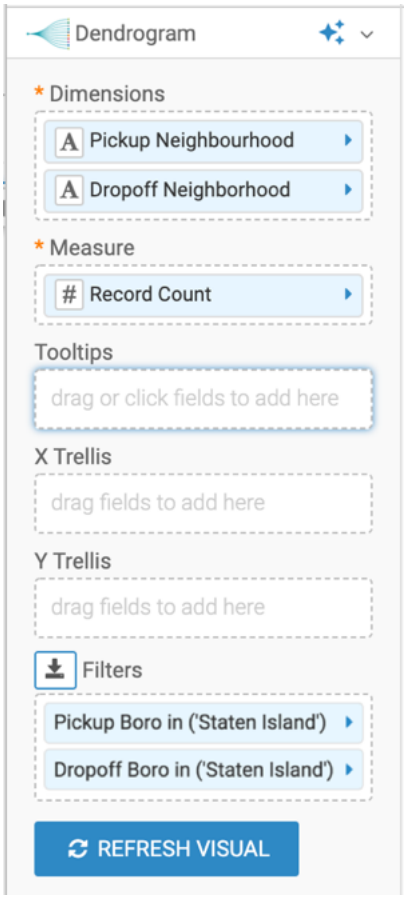

5. Click REFRESH VISUAL.
   The new dendrogram visual appears.

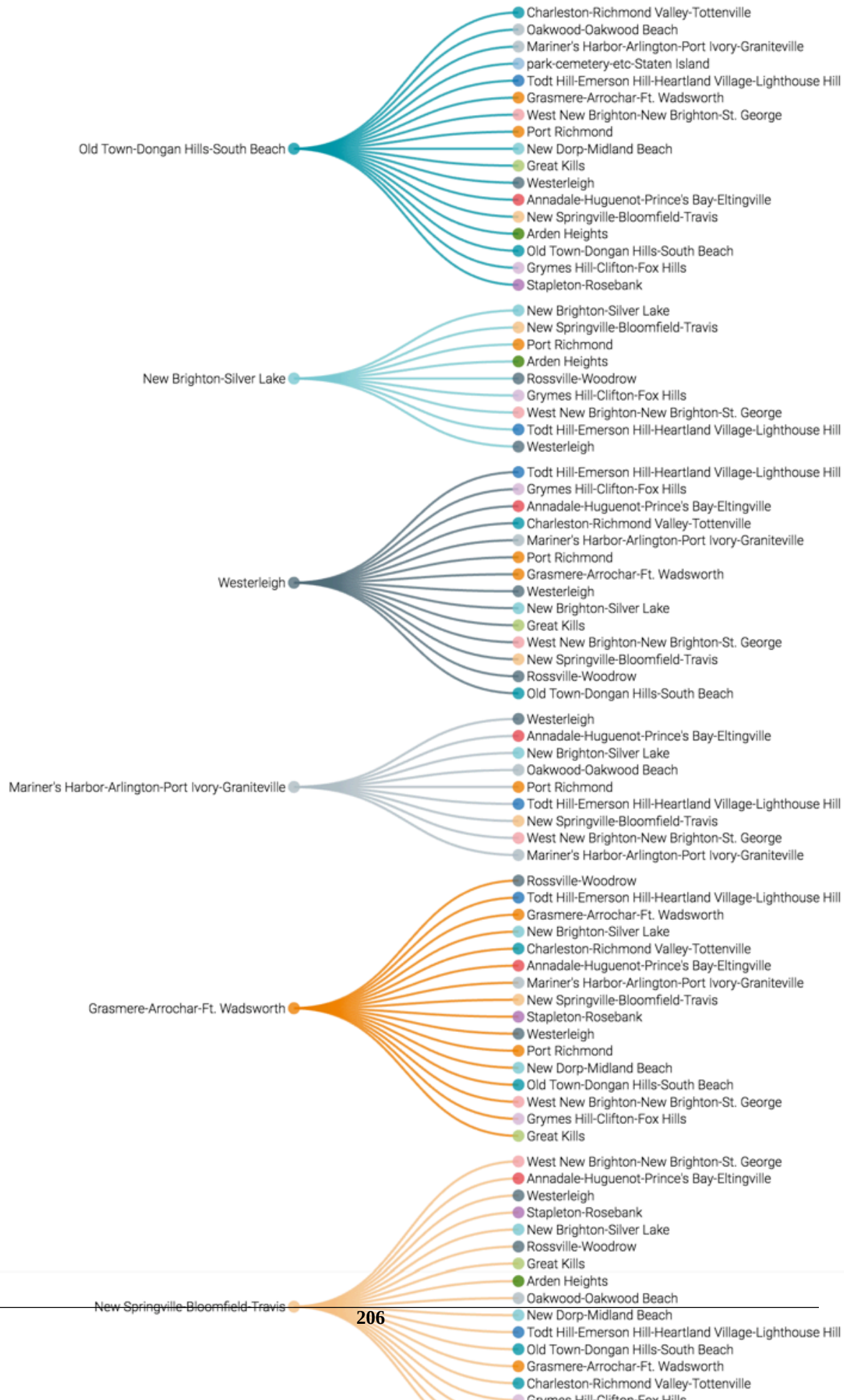
Old Town-Dongan Hills-South Beach
Charleston-Richmond Valley-Tottenville
Oakwood-Oakwood Beach
Mariner's Harbor-Arlington-Port Ivory-Graniteville
park-cemetery-etc-Staten Island
Todt Hill-Emerson Hill-Heartland Village-Lighthouse Hill
Grasmere-Arrochar-Ft. Wadsworth
West New Brighton-New Brighton-St. George
Port Richmond
New Dorp-Midland Beach
Great Kills
Westerleigh
Annadale-Huguenot-Prince's Bay-Eltingville
New Springville-Bloomfield-Travis
Arden Heights
Old Town-Dongan Hills-South Beach
Grymes Hill-Clifton-Fox Hills
Stapleton-Rosebank
New Brighton-Silver Lake
New Brighton-Silver Lake
New Springville-Bloomfield-Travis
Port Richmond
Arden Heights
Rossville-Woodrow
Grymes Hill-Clifton-Fox Hills
West New Brighton-New Brighton-St. George
Todt Hill-Emerson Hill-Heartland Village-Lighthouse Hill
Westerleigh
Westerleigh
Todt Hill-Emerson Hill-Heartland Village-Lighthouse Hill
Grymes Hill-Clifton-Fox Hills
Annadale-Huguenot-Prince's Bay-Eltingville
Charleston-Richmond Valley-Tottenville
Mariner's Harbor-Arlington-Port Ivory-Graniteville
Port Richmond
Grasmere-Arrochar-Ft. Wadsworth
Westerleigh
New Brighton-Silver Lake
Great Kills
West New Brighton-New Brighton-St. George
New Springville-Bloomfield-Travis
Rossville-Woodrow
Old Town-Dongan Hills-South Beach
Mariner's Harbor-Arlington-Port Ivory-Graniteville
Westerleigh
Annadale-Huguenot-Prince's Bay-Eltingville
New Brighton-Silver Lake
Oakwood-Oakwood Beach
Port Richmond
Todt Hill-Emerson Hill-Heartland Village-Lighthouse Hill
New Springville-Bloomfield-Travis
West New Brighton-New Brighton-St. George
Mariner's Harbor-Arlington-Port Ivory-Graniteville
Grasmere-Arrochar-Ft. Wadsworth
Rossville-Woodrow
Todt Hill-Emerson Hill-Heartland Village-Lighthouse Hill
Grasmere-Arrochar-Ft. Wadsworth
New Brighton-Silver Lake
Charleston-Richmond Valley-Tottenville
Annadale-Huguenot-Prince's Bay-Eltingville
Mariner's Harbor-Arlington-Port Ivory-Graniteville
New Springville-Bloomfield-Travis
Stapleton-Rosebank
Westerleigh
Port Richmond
New Dorp-Midland Beach
Old Town-Dongan Hills-South Beach
West New Brighton-New Brighton-St. George
Grymes Hill-Clifton-Fox Hills
Great Kills
New Springville-Bloomfield-Travis
West New Brighton-New Brighton-St. George
Annadale-Huguenot-Prince's Bay-Eltingville
Westerleigh
Stapleton-Rosebank
New Brighton-Silver Lake
Rossville-Woodrow
Great Kills
Arden Heights
Oakwood-Oakwood Beach
New Dorp-Midland Beach
Todt Hill-Emerson Hill-Heartland Village-Lighthouse Hill
Old Town-Dongan Hills-South Beach
Grasmere-Arrochar-Ft. Wadsworth
Charleston-Richmond Valley-Tottenville

6. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'NYC Taxi - Dendrogram'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
7. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

To adjust the dendrogram display, check all the available settings for this visual.

Shelves for dendrograms

Overview of shelves for Cloudera Data Visualization Dendogram visuals.

A Dendrogram visual has the following shelves:

Dimensions

Add two Dimensions fields from the DATA menu to define the start and end points of each connection in the plot.

This shelf accepts exactly two fields.

This is a mandatory shelf

Measure

Add a Measure field from the DATA menu. The aggregate determines the size of the connection.

This shelf accepts a single field.

This is a mandatory shelf.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimension field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Treemap

Cloudera Data Visualization enables you to create Treemap visuals. Treemaps display hierarchical data as a set of nested rectangles. Each 'branch' of the tree is a rectangle that corresponds to a particular dimension, with smaller rectangles that represent secondary dimensions. All rectangle areas are proportional to the aggregate measure that corresponds to the dimension.

About this task

Using color and size dimensions in the tree structure often brings to light new insights and correspondence patterns. Because of the highly efficient use of space, treemaps legibly display very large number of items simultaneously.

In the following example the NYC Taxi Trips dataset is used, which is based on the data in the ny-trips.csv file.

For an overview of shelves that specify this visual, see *Shelves for treemaps*.

Procedure

1. Start a new visual based on the NYC Taxi Trips dataset.
   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Treemap.

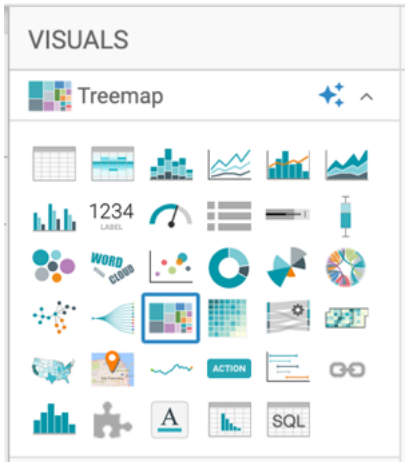

The shelves of the visual changed. They are now Dimensions, Measure, Tooltips, X Trellis, Y Trellis, and Filters. The mandatory shelves are:

- Dimensions: specify one or two dimensions to define the branches and leaf nodes of the treemap.
- Measure: specifies one measurement that determines the area of the leaf node.

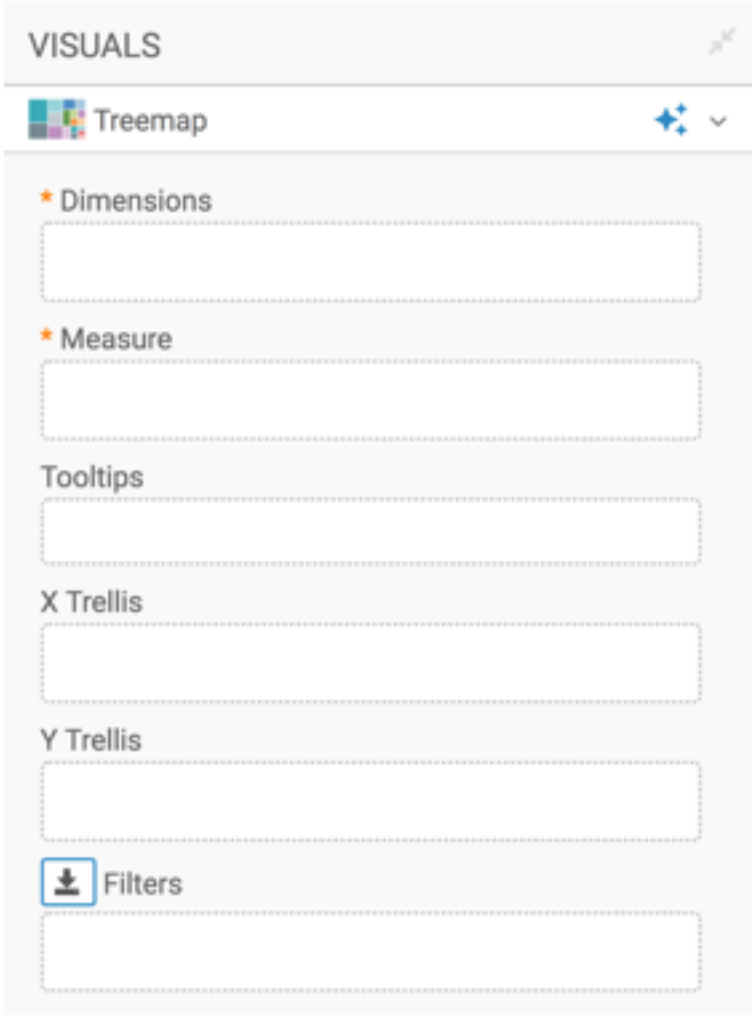

3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Under Dimensions, select Pickup Boro and Pickup Neighborhood, and add them to the Dimensions shelf, in that order.
   a) Under Measures, select Record Count and add it to the Measure shelf.

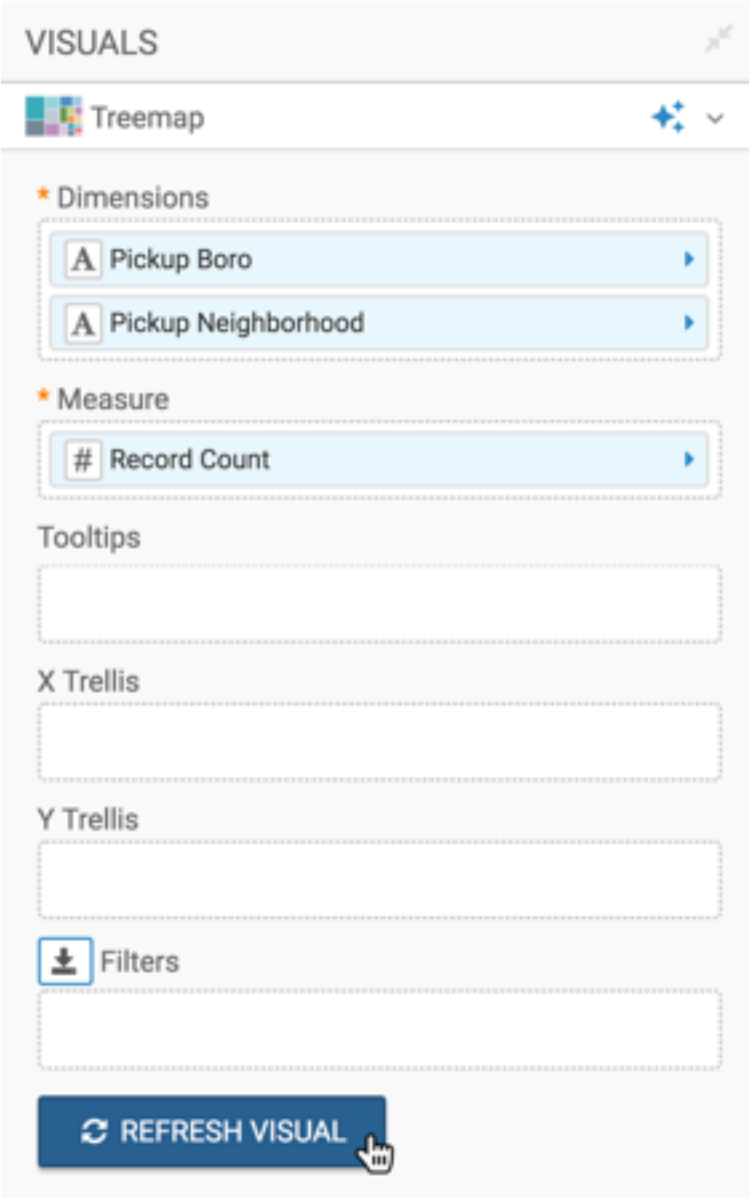

4. Click REFRESH VISUAL.

   The new treemap visual appears.

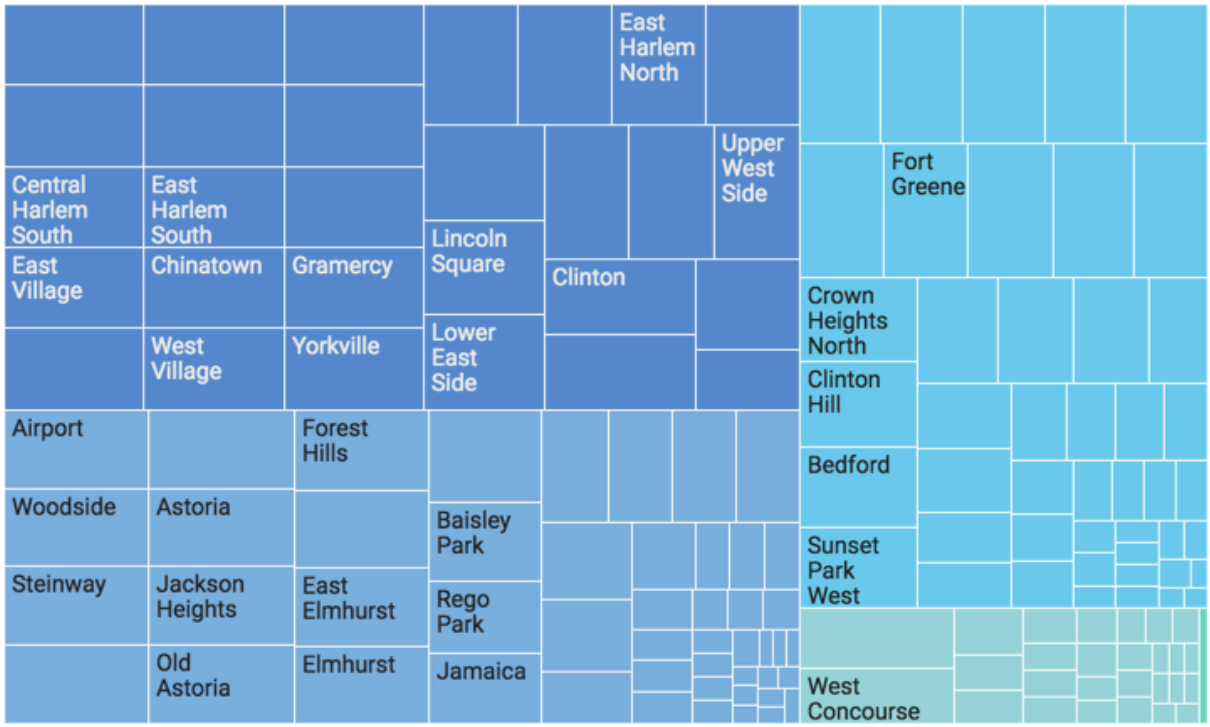

5. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'NYC Taxi - Treemap'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

6. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

To adjust the treemap display, check all the available settings for this visual.

Shelves for treemaps

Overview of shelves for Cloudera Data Visualization Treemap visuals.

A Treemap visual has the following shelves:

Dimensions

Add two Dimension fields from the DATA menu to define the branches of the treemap (primary rectangular divisions) and leaves of the treemap (the smaller nested divisions within each larger rectangular area).

This shelf accepts exactly two fields.

This is a mandatory shelf.

Measure

Add a Measure field from the DATA menu. The aggregate determines the size of the rectangular areas.

This shelf accepts a single field.

This is a mandatory shelf.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimension field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Correlation heatmap

Cloudera Data Visualization enables you to create Correlation Heatmap visuals. Correlation Heatmaps use colored cells, typically in a monochromatic scale, to show a 2D correlation matrix (table) between two discrete dimensions or event types. The values of the first dimensions appear as rows of the table, while the values of the second dimension are represented by the columns of the table. The color value of the cells is proportional to the number of measurements that match the dimensional values. This enables you to quickly identify incidence patterns, and to recognize anomalies.

About this task

Correlation Heatmap visuals are similar to Chords because they both compare exactly two dimensions. Correlation heatmaps are ideal for comparing the measurement for each pair of dimension values.

The following steps demonstrate how to create a new correlation visual on the SFPD Incidents dataset, which is based on data previously imported into Cloudera Data Visualization from the sfpd_incidents.csv datafile [data source default.sfpd_incidents].

For an overview of shelves that specify this visual, see *Shelves for correlation heatmaps*.

Note: This visual type supports smooth rendering of streaming data.

Procedure

1. Start a new visual based on the SFPD Incidents dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Correlation Heatmap.

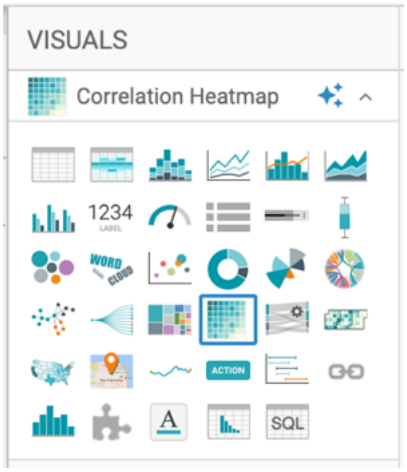

The shelves of the visual changed. They are now Dimensions, Measures, Tooltips, X Trellis, Y Trellis, and Filters. Both Dimensions and Measures are mandatory.

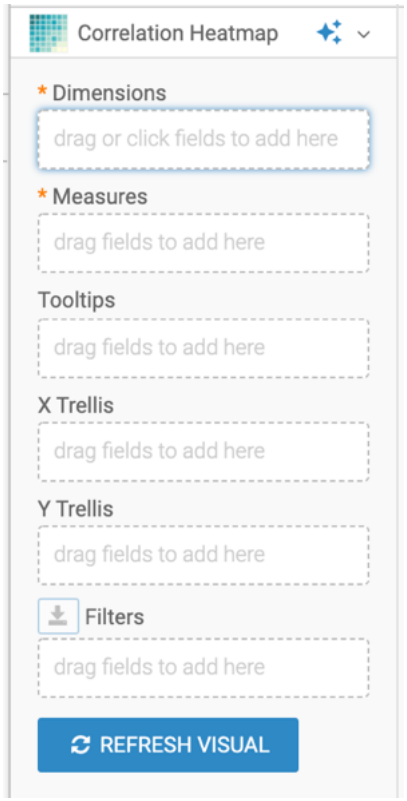

3. To show specific items, populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Under Dimensions, select pddistrict and drag it to the Dimensions shelf.
   b) Under Dimensions, select descript and drag it to the Dimensions shelf.
   c) Under Measures, select Record Count and drag it to the Measures shelf.

      Record Count is defined as a sum of events. If you hover over it with your mouse, you can see a black detail bubble with sum(1) contents.

4. Click REFRESH VISUAL.

   The default correlation heatmap visual appears.

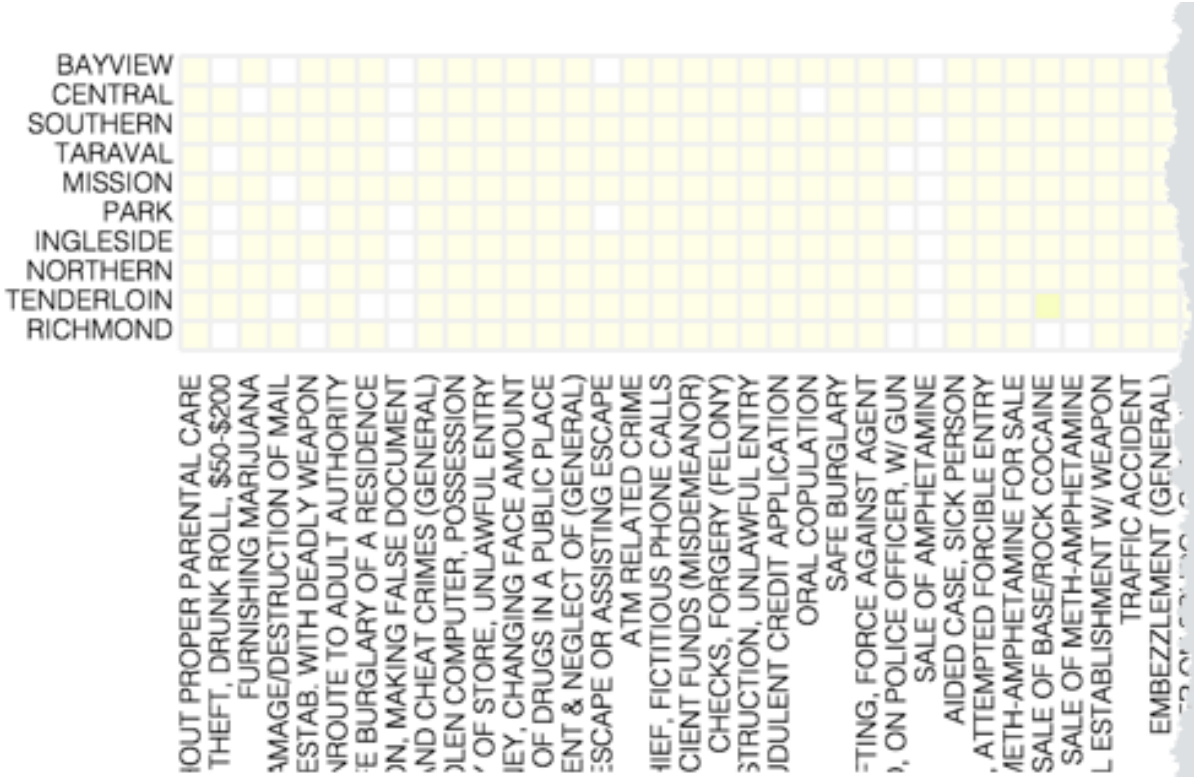

   You can see that this dataset has a very large number of possible values that represent the columns of the table. If you scroll to the right, you will see some cells rendered in dark shades of green.

5. To examine a shorter list of categories, you can add some filtering to the visual.
   a) Under Dimensions, select datetime and drag it to the Filters shelf .
   b) Repeat it with category and descript.
   c) On the Filters shelf, click the (down arrow) on the descript field, and then click Pick values from a list.
   d) Select a number of values.

      In this example, 7 distinct options were picked.

6. Click REFRESH VISUAL.

   You can see that this smaller matrix also shows the entire range of color values.

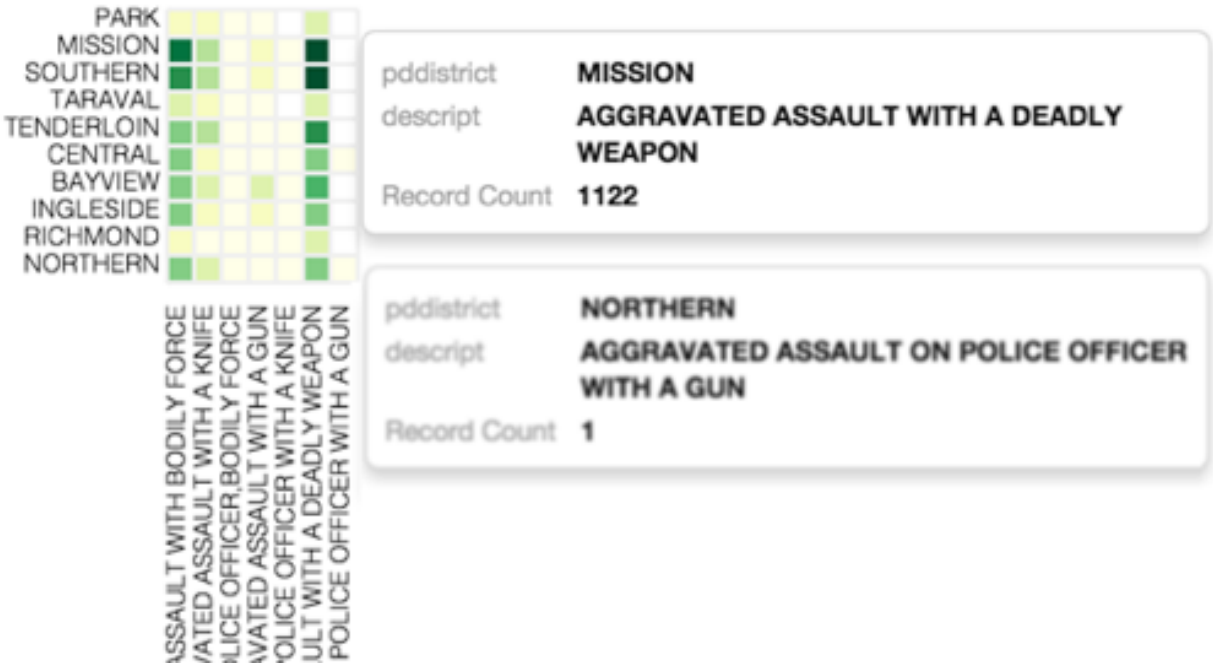

7. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'SFPD Incidents - Correlation Heat Map'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

8. At the top left corner of the Dashboard Designer, click SAVE.

Shelves for correlation heatmaps

Overview of shelves for Cloudera Data Visualization Correlation Heatmap visuals.

A Correlation Heatmap visual has the following shelves:

Dimensions

Add two Dimension fields from the DATA menu. The first dimension defines the rows and the second dimension defines the columns of the plot.

This shelf accepts exactly two fields.

This is a mandatory shelf.

Measures

Add a Measure field from the DATA menu to specify the values aggregated across columns and rows.

This shelf accepts a single field.

This is a mandatory shelf.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimension field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Correlation flow

Cloudera Data Visualization enables you to create Correlation Flow visuals. Correlation Flows enable quick visual analysis of co-occurrence of two or more distinct dimensions. In simple form, a Correlation Flow visual is a linear representations of a two-dimensional Correlation Heatmap matrix. Where a Correlation Heatmap represents correlation values through color intensity, the Correlation Flow plots thicker lines and taller dimension boxes to represent greater correlation.

About this task

The following steps demonstrate how to create a new Correlation Flow visual on the Customer Value Analysis dataset. This dataset is based on data previously imported into Cloudera Data Visualization from the customer-value-analysis.csv datafile.

For an overview of shelves that specify this visual, see *Shelves for correlation flows*.

Note: Correlation flow is not limited to two dimensions. You can model a large number of correlation 'dimensions'. Ordering of the dimensions on the shelves of the visual provides different insights into the dataset.

Procedure

1. Start a new visual based on the Customer Value Analysis dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Correlation Flow.

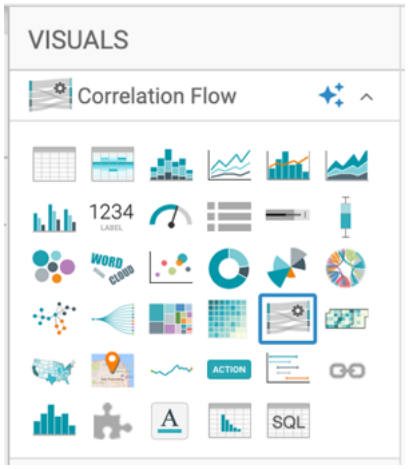

The shelves of the visual changed. They are now Dimensions, Measures, Tooltips, and Filters. Both Dimensions and Measures are mandatory.

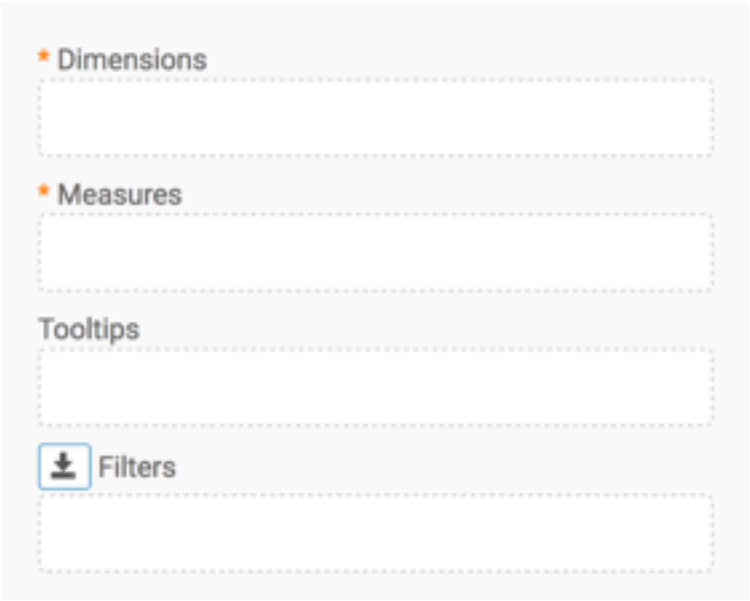

3. Populate the shelves from the available fields:
   a) Under Dimensions, select Vehicle Class and drag it to the Dimension shelf.
   b) Repeat it for Vehicle Size, Employment Status, Policy Type, and Policy.
   c) Under Measures, select Record Count and drag it to the Measures shelf.

4. Click REFRESH VISUAL.

   The correlation flow visual appears.

   Figure 20: Correlation Flow Visual

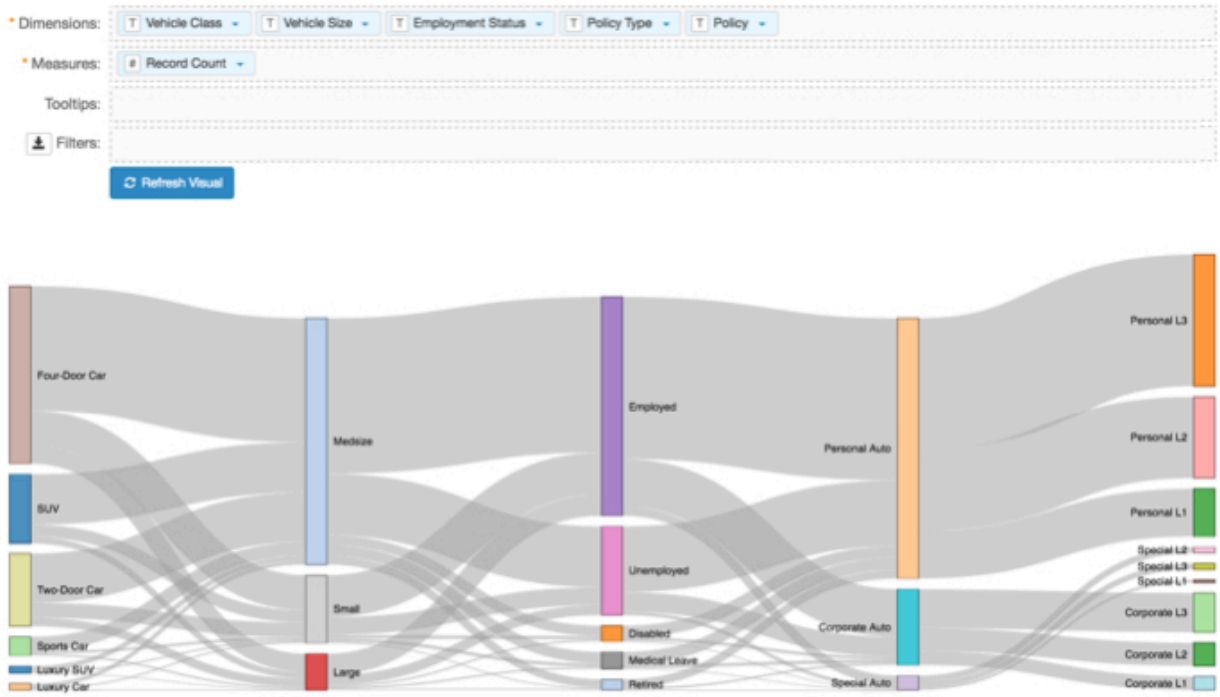

You can see that each distinct line segment between two dimensional events reports its own correlation data, as demonstrated by the Tooltips in the following chart.

- Tooltip 1 demonstrates the correlation between Four-Door and Medsize vehicles. The dataset has 3,237 Four-Door cars, and they represent 36% of the total. Additionally, 70% of these cars are considered Medsize. Viewing this from the other direction of the correlation relationship, 50% of all Medsize cars are four-door cars.
- Tooltip 2 demonstrates the correlation between Medsize vehicles and Unemployed owners. It shows that 1,597 people are in both categories, and this segment represents 17% of all records. 25% of Medsize cars are owned by people who are unemployed, while 69% of unemployed people own a medsize vehicle.
- Tooltip 3 demonstrates the correlation between Employed owners and the ones who have a Corporate Auto vehicle policy. It shows that 1,238 people are in these two categories, which is 14% of the total. 22% of

employed owners have a corporate auto policy, while 63% of people with a corporate auto policy are actively employed.

Figure 21: Correlation Flow Visual

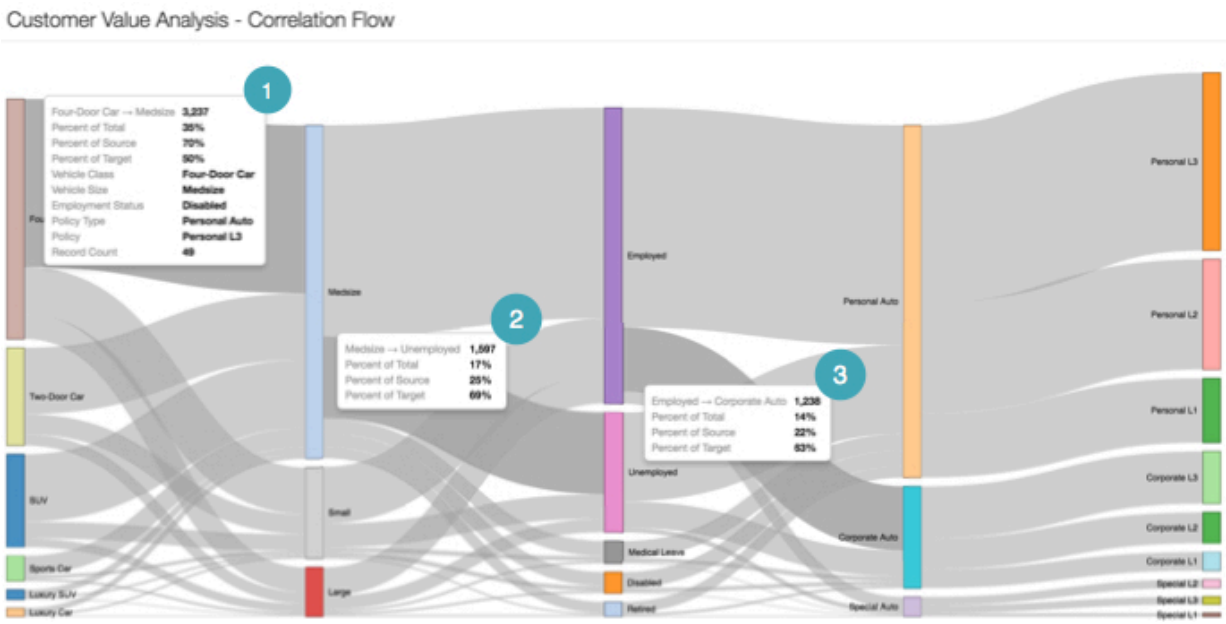

5. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'Customer Value Analysis - Correlation Flow'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

6. At the top left corner of the Dashboard Designer, click SAVE.

Shelves for correlation flows

Overview of shelves for Cloudera Data Visualization Correlation Flow visuals.

A Correlation Flow visual has the following shelves:

Dimensions

Add two Dimension fields from the DATA menu to define the start and end of each connection of the plot.

This shelf accepts exactly two fields.

This is a mandatory visual

Measures

Add a Measure field from the DATA menu. The aggregate measurement determines the thickness of the connector between each pair of dimensions.

This shelf accepts a single field.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Calendar heatmaps

Cloudera Data Visualization enables you to create Calendar Heatmap visuals. A Calendar Heatmap uses colored cells, typically in a monochromatic scale, to show relative number of events for each day in a calendar view. Days are arranged into columns by week and grouped by month and years. This enables you to quickly identify daily and weekly patterns, and to recognize anomalies.

About this task

The following steps demonstrate how to create a new calendar visual on the SFPD Incidents dataset, based on data previously imported into Cloudera Data Visualization from the sfpd_incidents.csv.zip datafile.

For an overview of shelves that specify this visual, see *Shelves for calendar heat maps*.

Note: This visual type supports smooth rendering of streaming data.

Procedure

1. Start a new visual based on the SFPD Incidents dataset.

   For instructions, see *Creating a visual*.

2. Select Calendar Heatmap in the VISUALS menu.

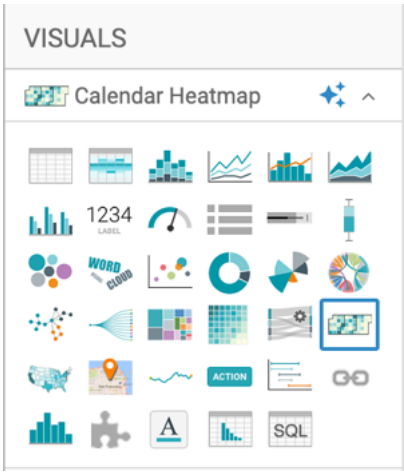

The shelves of the visual changed. They are now Date (mandatory), Measures (mandatory), Tooltips, X Trellis, Y Trellis, and Filters.

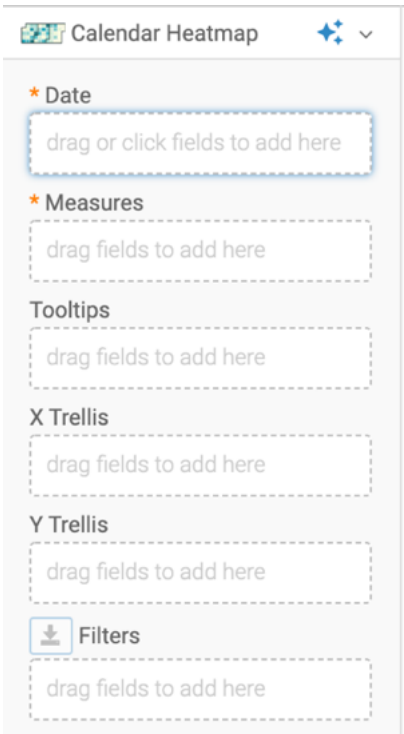

3. To show specific items, populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) Under Dimensions, select date and drag it to the Date shelf.
   b) Under Measures, select Record Count and drag it to the Measures shelf.

   Note: Record Count is defined as a sum of events. If you hover over it with your mouse, you can see a black detail bubble with sum(1) contents.

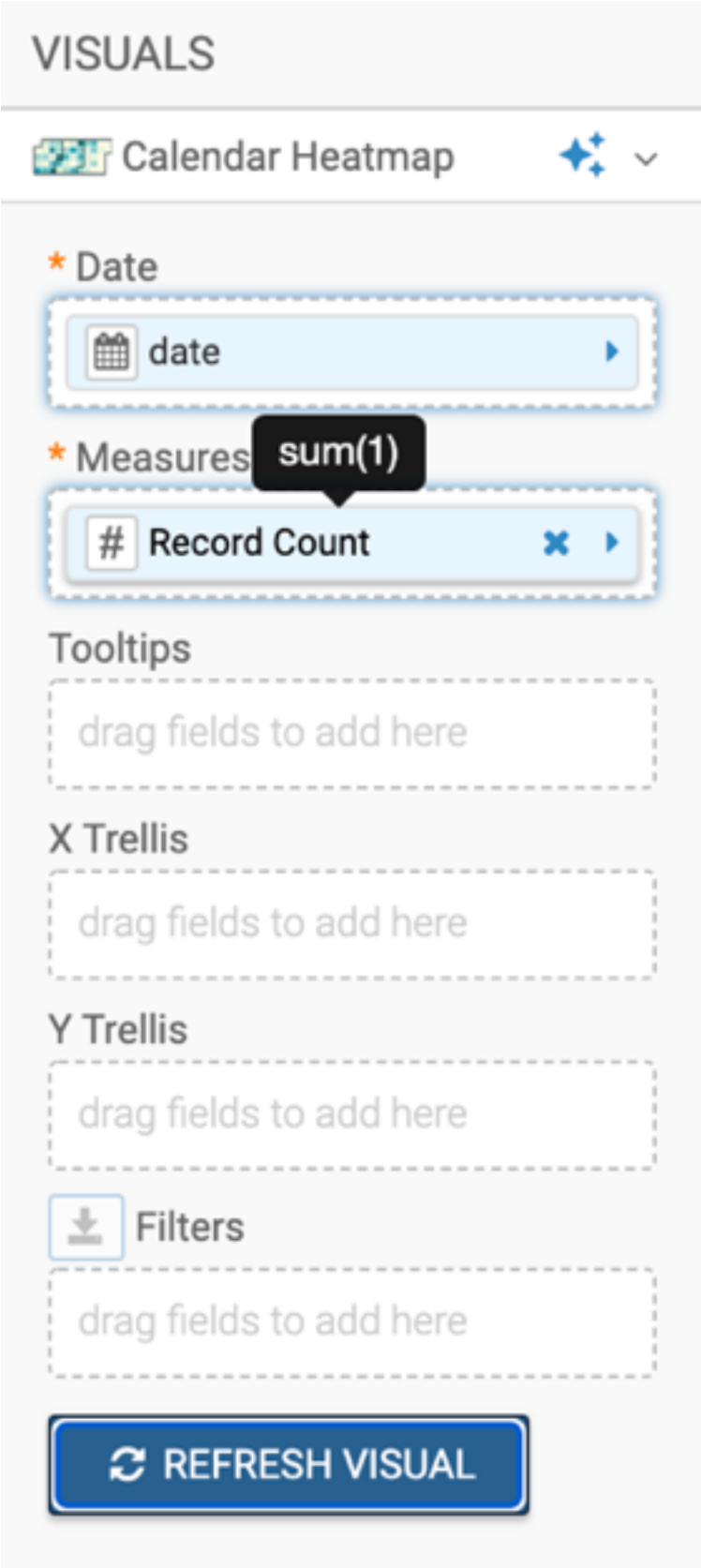
VISUALS
Calendar Heatmap
* Date
date
* Measures
sum(1)
Record Count
Tooltips
drag fields to add here
X Trellis
drag fields to add here
Y Trellis
drag fields to add here
Filters
drag fields to add here
REFRESH VISUAL

4. The field on the Date shelf should have a calendar icon to indicate that it is in the Date/Time category of data types. If the field is not automatically recognized as a date data type, convert it.

   On the Date shelf of the visual, click the date field, select Date/Time Function Date .

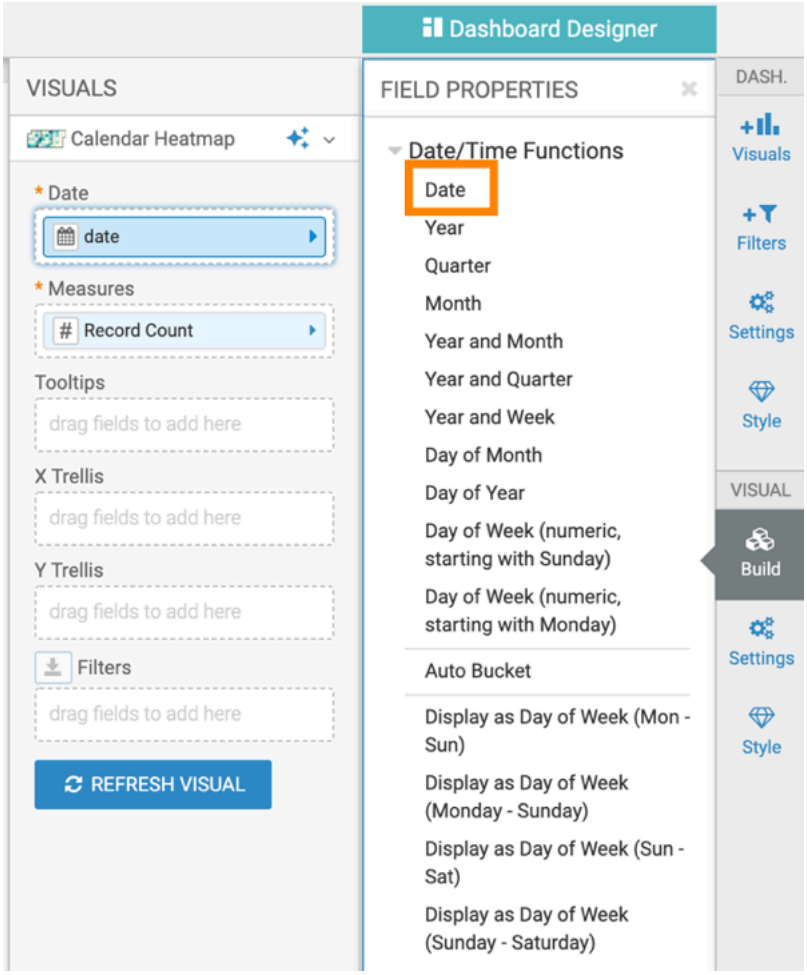

5. Click REFRESH VISUAL.

   The default calendar heat map visual appears.

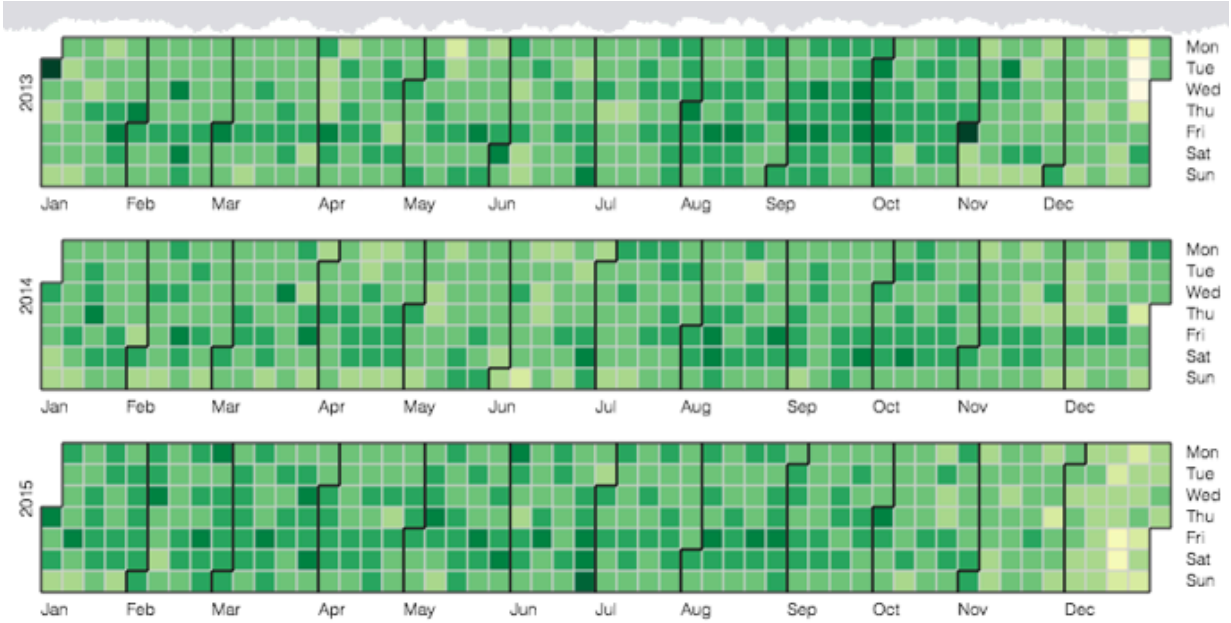

6. Change the title of the visual to SFPD Incidents - Calendar Heat Map.
7. Click SAVE at the top left corner of the Dashboard Designer.

Shelves for calendar heatmaps

Overview of shelves for Cloudera Data Visualization Calendar Heatmap visuals.

A Calendar Heatmap visual has the following shelves:

Date

Add a date field from the DATA menu to determine the range of the calendar.

This shelf accepts a single field.

This is mandatory shelf.

Measures

Add a Measure field from the DATA menu to specify the values aggregated across the dates of the calendar.

This shelf displays only one field on the visual. Additional fields appear only in the Tooltip.

This is mandatory shelf.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimension field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Maps

Cloudera Data Visualization enables you to create Map visuals.

A geomap of a country, continent, or region superimposes measurement values onto a geographic map.

Cloudera Data Visualization has two basic geomap representation options:

- A filled or choropleth map (Greek #### for area or region and ###### for multitude) shows geographical areas that are shaded in proportion to the value of the measurement that is displayed on the map.

  Tip: Consider the following when building choropleth maps:

  - Use choropleth maps when your data are attached to enumeration units, such as counties, countries, zipcodes, and so on.
  - Standardize data to show rates and ratios instead of raw data counts. For example, instead of plotting gross GDP, plot GDP per Capita.
  - Use choropleth maps when you have data with continuous statistical surface. For example, Population is a count and representing it in colored map may be misleading. Population density is a ratio with a continuous statistical surface (even where 0 represents unpopulated areas); so it can be used in choropleth maps.
- A geomap with symbols or marks, such as bubbles, superimposes these symbols onto a geographical map. By default, the area of the marks corresponds to the relative size of the measurement.

Cloudera Data Visualization supports the following geographical specifications by automatically recognizing them as qualified GEO dimensions inside the dataset:

- country
- state or province for United States and Canada, respectively
- county, for United States only
- zipcode or postal code for United States and Canada, respectively
- longitude and latitude
- DMA, for Nielsen Designated Market Area Regions, in United States

If the relevant field in your dataset is not named correctly, be sure to adjust it using the Change Type Geo Option, described in *Change type*. Note also that you must use Geo Type fields when limiting the Drill Into options. For more information, see *Limiting drill into options*.

To build the two types of map visuals with specified geographical units, follow the instructions for each option. Additionally, see the description on how to build a *Trellised map*.

For an overview of shelves that specify this visual, see *Shelves for maps*.

Choropleth country map

Map visuals can show relative measurement values for countries as colored areas. A world map can be used to show a comparison of measurement values among countries. With data that meets ratio and statistical continuity guidelines, a choropleth map is a good chart choice.

About this task

The following steps demonstrate how to create a new map visual on the dataset World Life Expectancy [data source samples.world_life_expectancy]. It produces colored regions for the measurements that correspond to the country field.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Map.

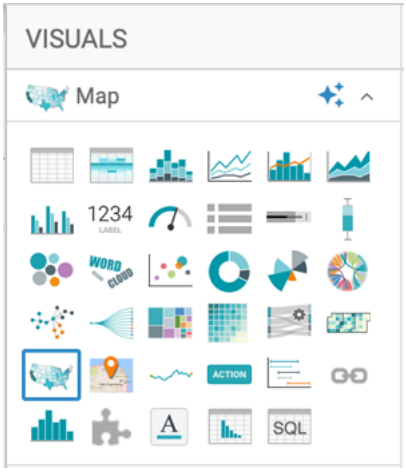

The shelves of the visual changed. They are now Geo, Color, Size, Drill, Tooltips, X Trellis, Y Trellis, and Filters. The only mandatory shelf for map visuals is Geo.

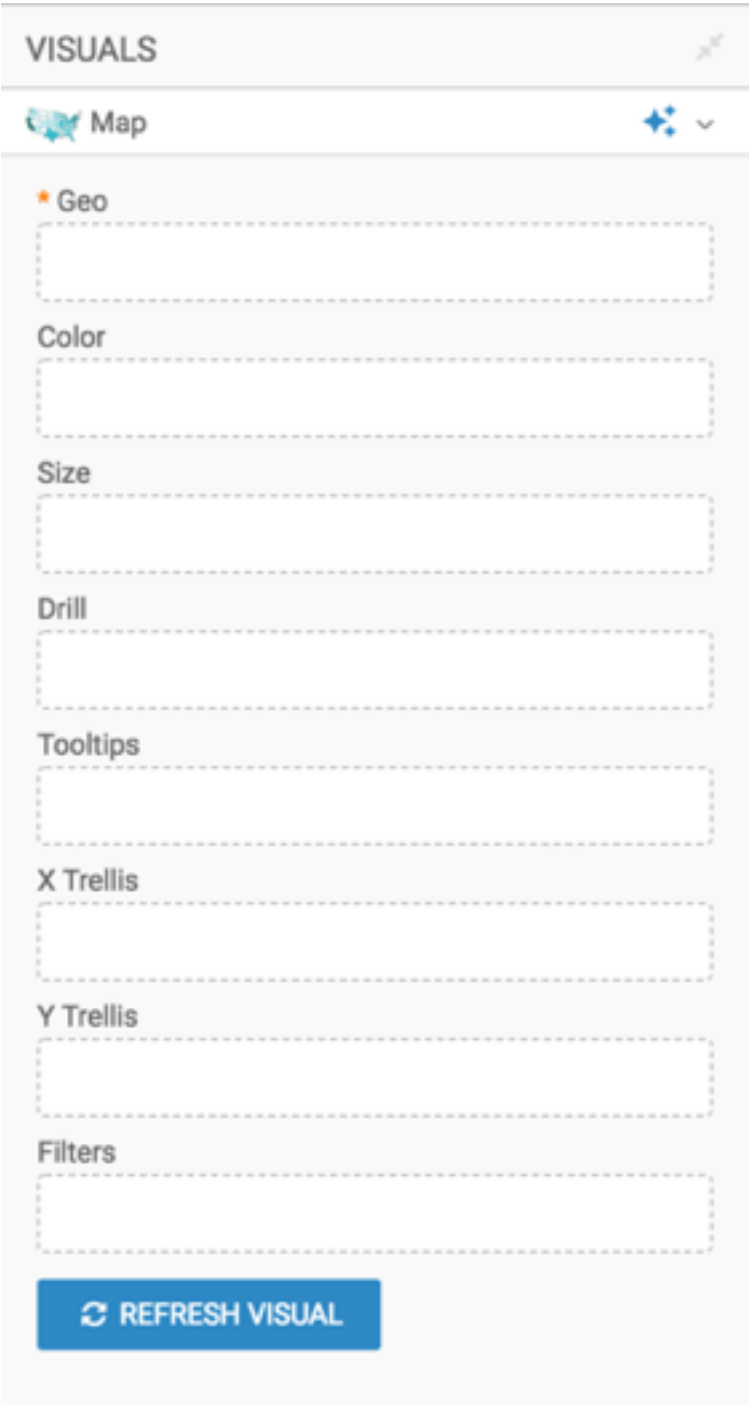
VISUALS
Map
* Geo
Color
Size
Drill
Tooltips
X Trellis
Y Trellis
Filters
REFRESH VISUAL

3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Under Dimensions, select country and add it to the Geo shelf.
   b) Under Measures, select gdp_per_capita and add it to the Color shelf.
   c) Under Measures, select year and add it to the Filters shelf. When the Filter for year window modal appears, select 2010 in the Values tab, and click Apply.

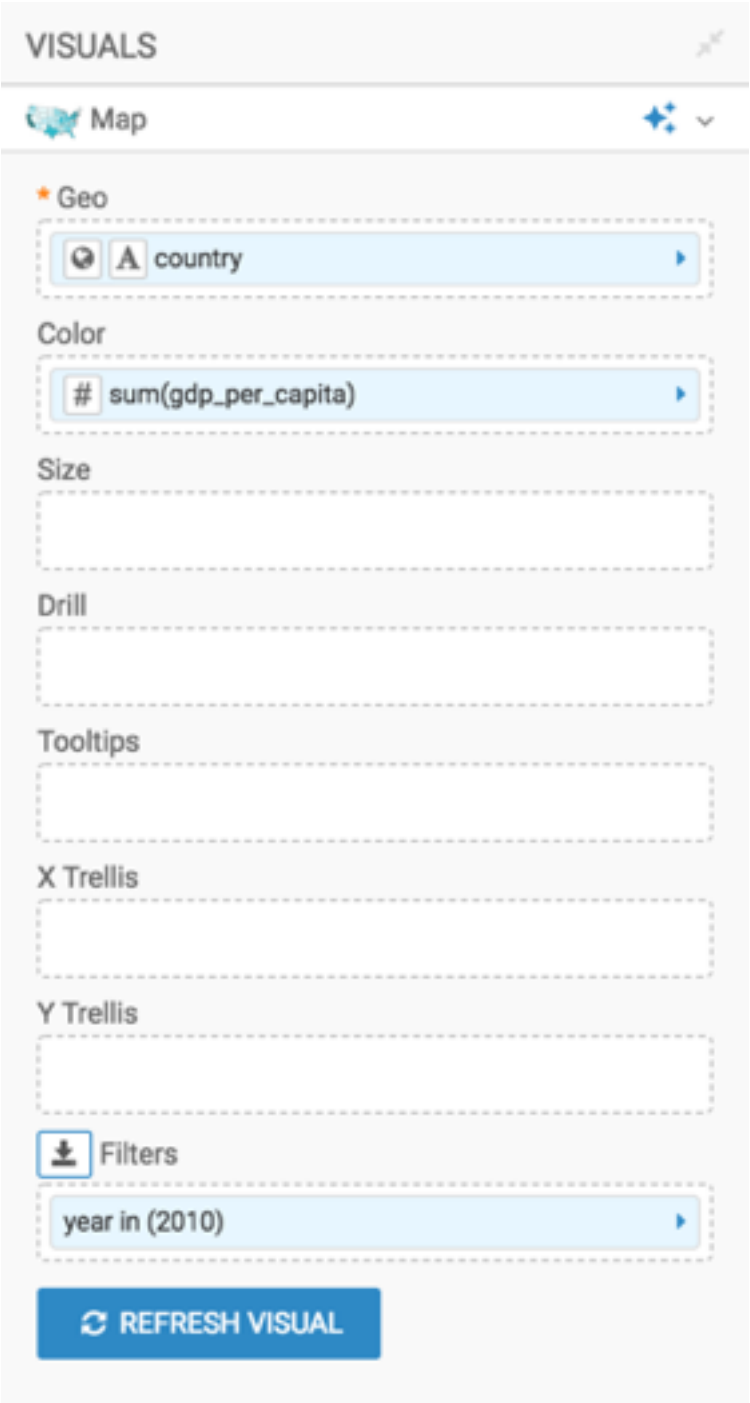

4. Click REFRESH VISUAL.

   The map visual graph appears.

5. On the Tooltips shelf, add several fields from Dimensions and Measures.

   This enables you to see specific descriptive information in your visuals, such as input values, segment affiliation, and calculations.

   For example, from Dimensions, add un_subregion, un_region, and year onto the Tooltip shelf.

6. Alias the fields on the shelves to improve the readability of the visual and its tooltip. See *Alias*.
   - gdp_per_capita as GDP per Capita
   - un_subregion as UN Subregion
   - un_region as UN Region
   - year as Year
7. Format the number reported by the GDP per Capita field to include a thousands separator, and format Year not to use the thousands separator.

   For more information, see *Real number display format* and *Integer display format*.

8. [Recommended] Enable panning and zooming option, as described in *Customizing zoom*.
9. Change the color palette to blue, as described in *Change color palette of visuals*.

10. Click REFRESH VISUAL.

The map visual appears. You can move the visual (pan), zoom in and out, and see the details in the Tooltips text.

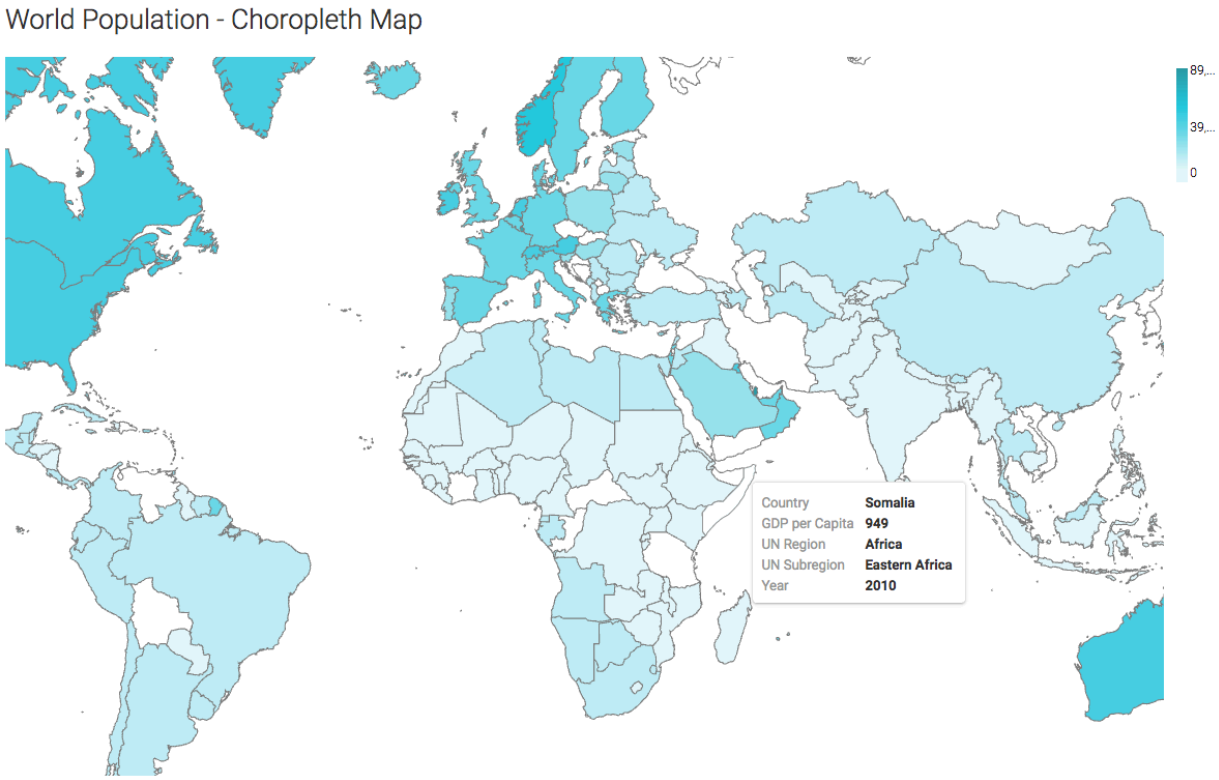

11. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

In this example, the title is changed to 'World Population - Choropleth Map'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

12. At the top left corner of the Dashboard Designer, click SAVE.

Country map with bubbles

Cloudera Data Visualization Map visuals can use bubbles to show relative measurement values for countries. You can display two measurements at the same time: the first measurement as the color of the bubble, and the second one as the relative size of the bubble.

About this task

The following steps demonstrate how to create a new map visual on the dataset World Life Expectancy [data source samples.world_life_expectancy]. It produces bubbles with area that corresponds to the relative population in each country.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.

For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Map.

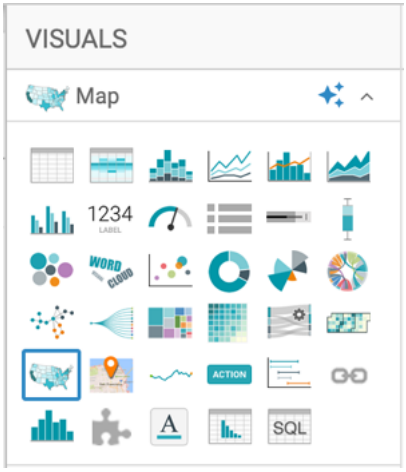

The shelves of the visual changed. They are now Geo, Color, Size, Drill, Tooltips, X Trellis, Y Trellis, and Filters. The only mandatory shelf for map visuals is Geo.

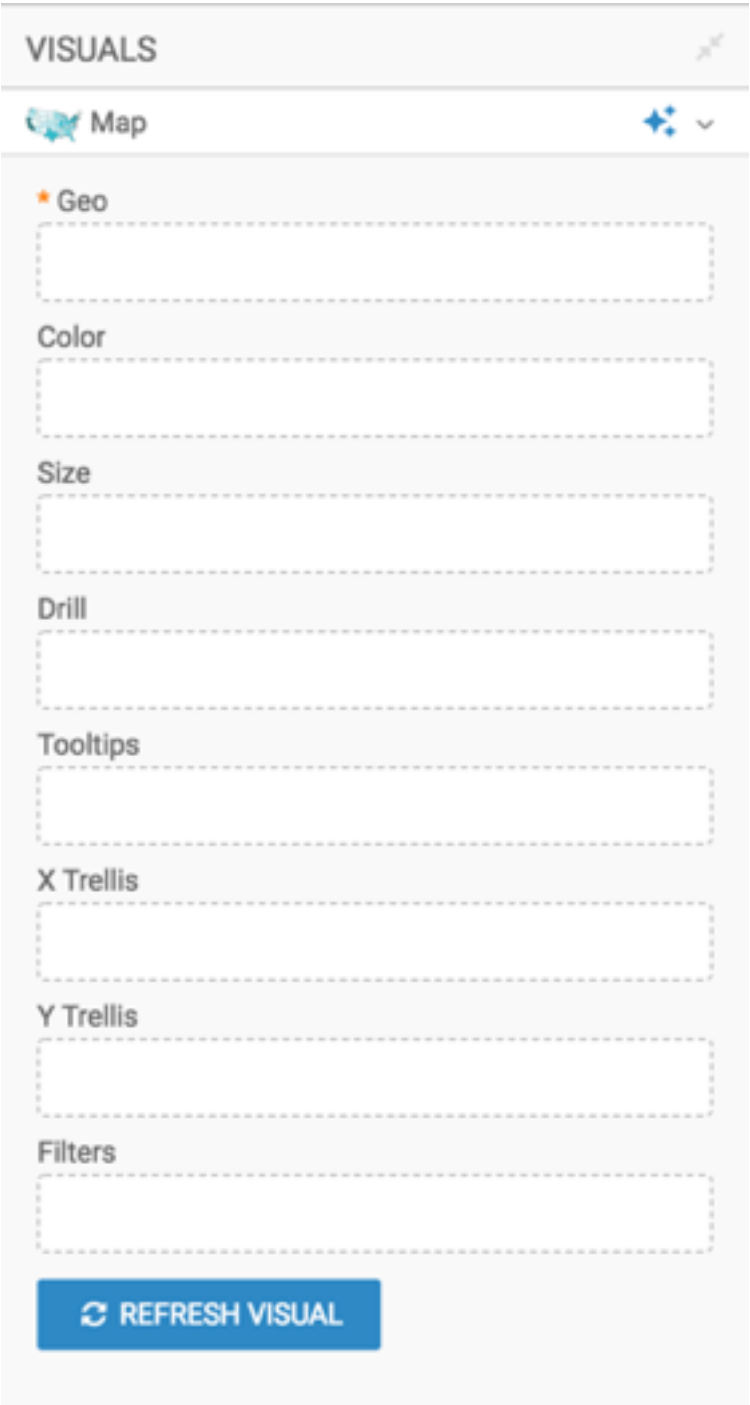
VISUALS
Map
* Geo
Color
Size
Drill
Tooltips
X Trellis
Y Trellis
Filters
REFRESH VISUAL

3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Under Dimensions, select country and add it to the Geo shelf.
   b) Under Measures, select population and add it to the Color shelf.
   c) Under Measures, select gdp_per_capita and add it to the Size shelf.
   d) Under Measures, select year and add it to the Filters shelf. When the Filter for year window modal appears, select 2010 in the Values tab, and click APPLY.

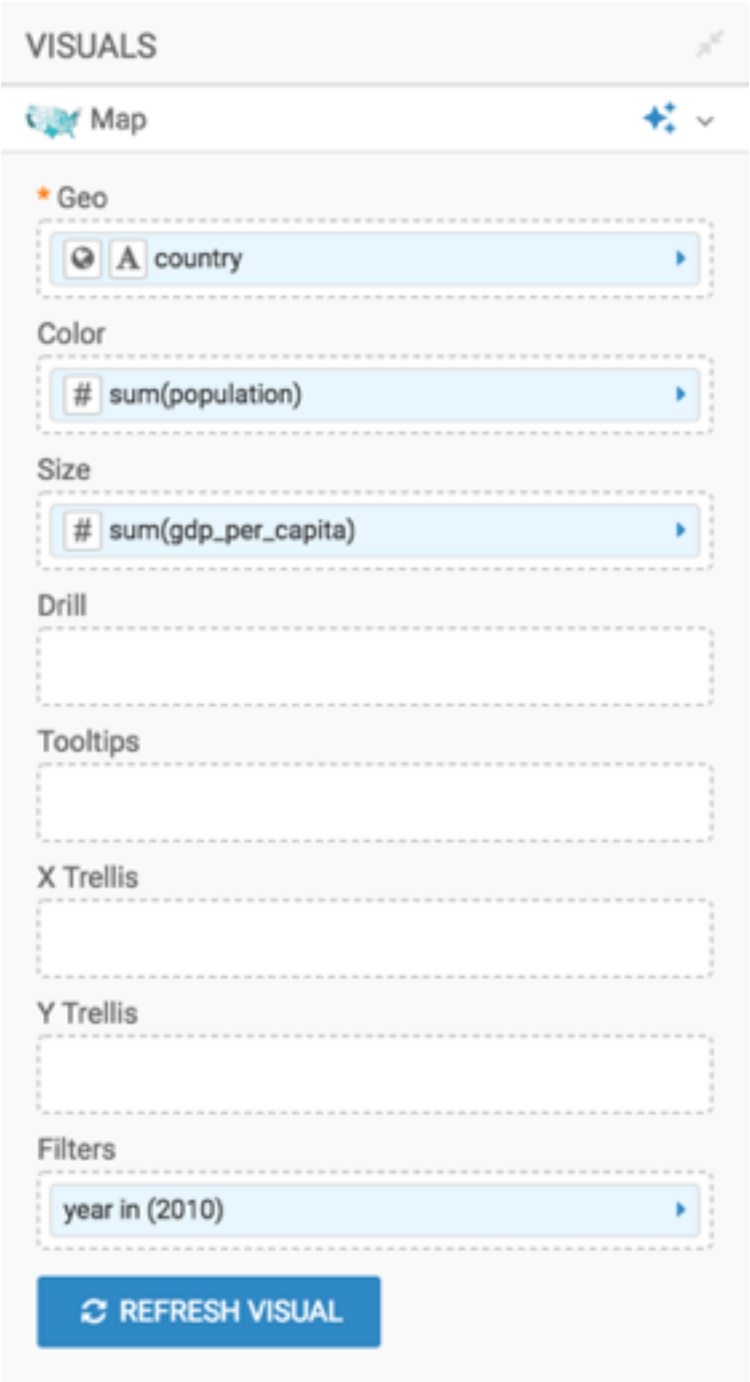

4. Change the map from choropleth to bubble marks, as described in *Displaying bubbles on maps*.
5. Adjust the size range of the marks to 1-10, as described in *Changing the mark size range*.
6. Click REFRESH VISUAL.

   The map visual graph appears.

7. On the Tooltips shelf, add several fields from Dimensions and Measures.

   This enables you to see specific descriptive information in your visuals, such as input values, segment affiliation, and calculations.

   For example, from Dimensions, drag un_subregion, un_region, and year to the Tooltips shelf.
8. Alias the fields on the shelves to improve the readability of the visual and its tooltip.
   - population as Population
   - un_subregion as UN Subregion
   - un_region as UN Region
   - year as Year

   For instructions, see *Alias*.
9. Format the number reported by the GDP per Capita field to include a thousands separator, and format Year not to use the thousands separator.

   For more information, see *Real number display format* and *Integer display format*.
10. [Recommended] Enable panning and zooming option, as described in *Customizing zoom*.
11. Change the color palette to blue, as described in *Change color palette visuals*.
12. Click REFRESH VISUAL.

    You can move the visual (pan), zoom in and out, and see the details in the Tooltips text.

13. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

    In this example, the title is changed to 'World Population - Map with Bubbles'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

14. At the top left corner of the Dashboard Designer, click SAVE.

Choropleth US state map

Cloudera Data Visualization Map visuals let you plot measurements, as areas, over US State regions.

About this task

A map of the United States shows a comparison of measurement values among US States. The following steps demonstrate how to create a new map visual on the dataset US County Population [data source samples.us_counties]. It produces colored regions for the measurements that correspond to the state field.

Procedure

1. Start a new visual based on the US County Population dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Map.

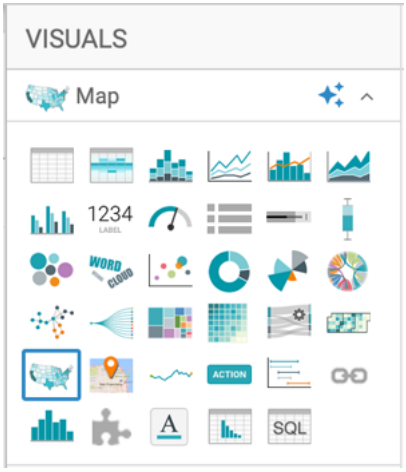

The shelves of the visual changed. They are now Geo, Color, Size, Drill, Tooltips, X Trellis, Y Trellis, and Filters. The only mandatory shelf for map visuals is Geo.

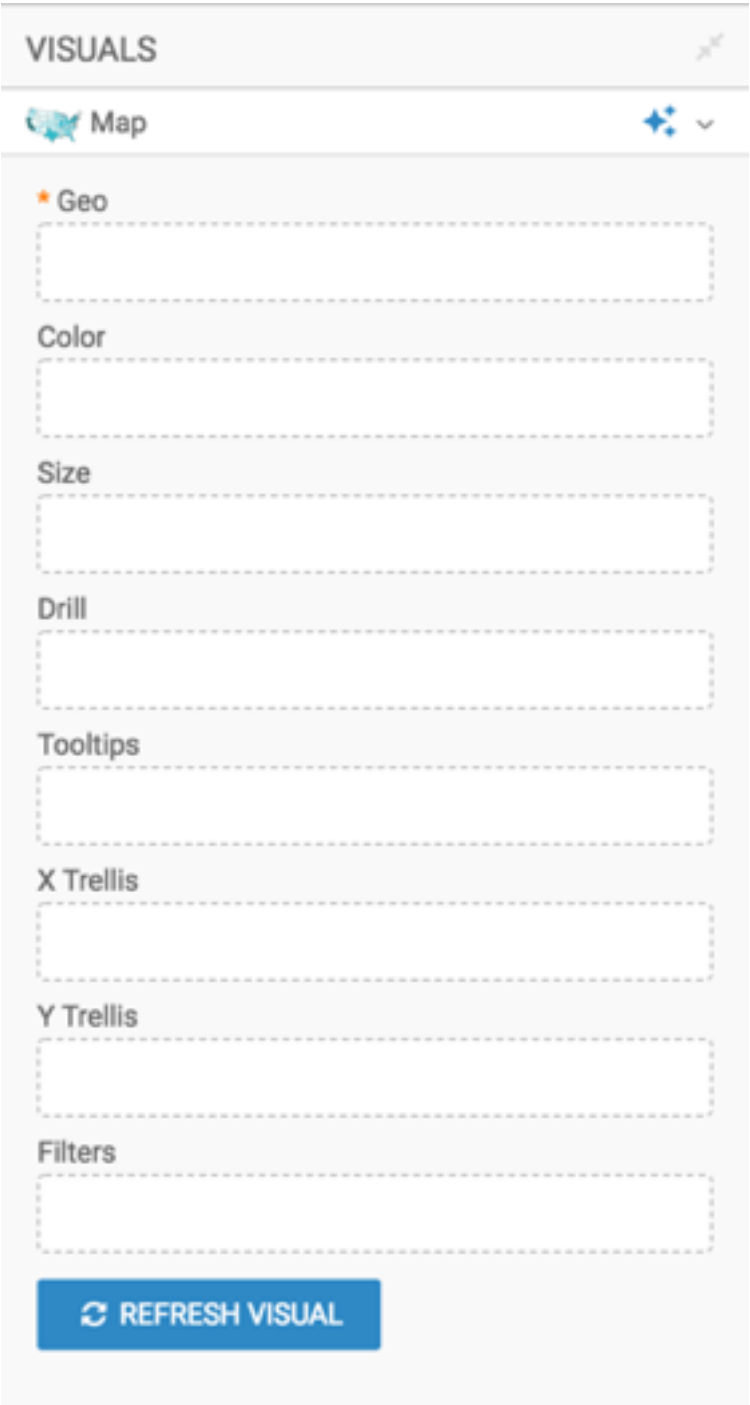
VISUALS
Map
* Geo
Color
Size
Drill
Tooltips
X Trellis
Y Trellis
Filters
REFRESH VISUAL

3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) Under Dimensions, select stname and add it to the Geo shelf.
   b) Click the field to open the FIELD PROPERTIES menu, expand the Change Type option, and select the Geo Type State.
   c) Under Measures, select tot_pop and add it to the Colors shelf.
   d) Click the field to open the Field Properties menu, select Display Format,Category Integer, and select the thousands separator option. Click Save.

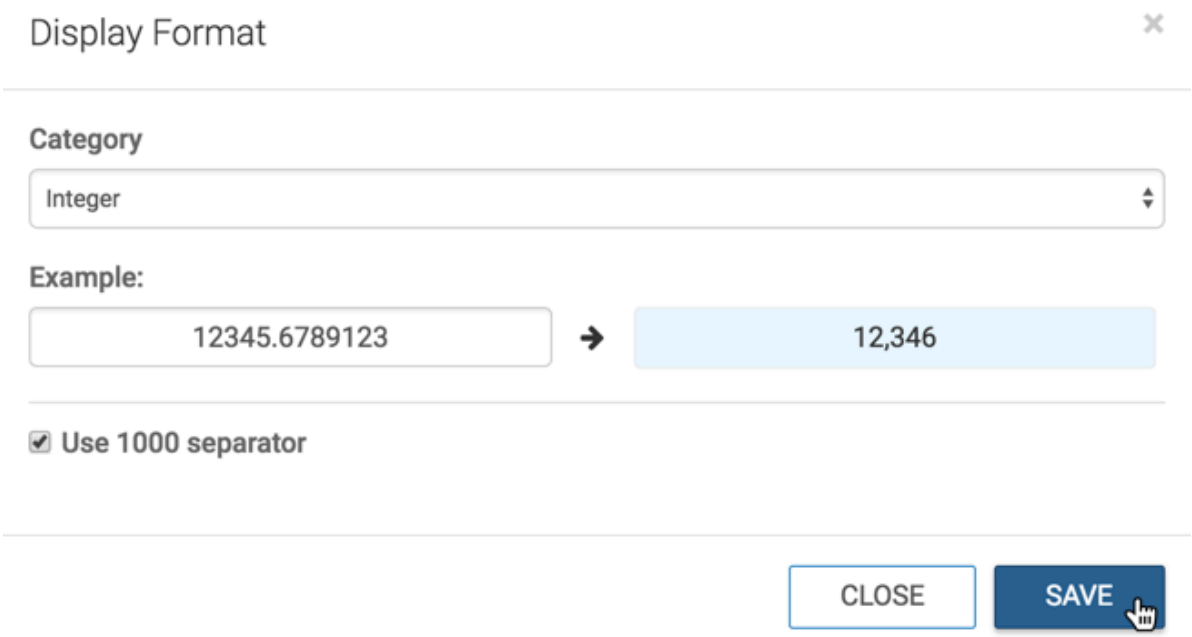

4. Alias the field stname as State, and sum(tot_pop) as Population; see *Alias*.
5. Enable panning and zooming option, as described in *Customizing zoom*.
6. Change the color palette, as described in *Change color palette of visuals*.
7. Click REFRESH VISUAL.

   The map visual appears.

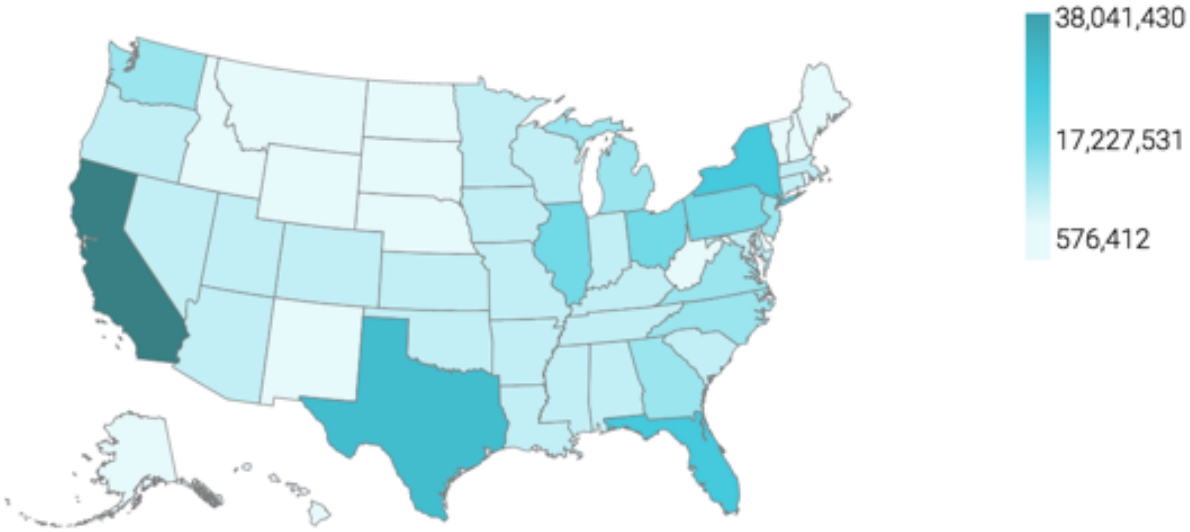

8. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'US State Population - Choropleth Map'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
9. At the top left corner of the Dashboard Designer, click SAVE.

US state map with bubbles

Cloudera Data Visualization Map visuals let you plot measurements, as bubbles, over US State regions. Bubble maps display up to two measurements simultaneously: as the color of the bubble, and as the relative size of the bubble.

About this task

A map of the United States shows a comparison of measurement values among US States. The following steps demonstrate how to create a new map visual on the dataset US County Population [data source samples.us_counties]. This produces a map with bubble marks that correspond to the total population of each state.

Procedure

1. Clone the visual that you developed in *State map, choropleth*.

   For instructions, see *Cloning a visual*.
2. Add a measure to the Size shelf.

   For example, to show the relative size of the Hispanic population across the country, use the following expression in the Enter/Edit Expression editor:

   ```
   sum([h_female]+[h_male])
   ```

   For more information, see *Enter or edit expressions*.
3. Change the map from choropleth to bubble marks, as described in *Displaying bubbles on maps*.
4. Adjust the size range of the marks to 1-10, as described in *Changing the mark size range*.
5. Alias all the fields.

   For instructions, see *Alias*.
6. Change the color palette or simply reverse it, as described in *Change color palette of visuals* and *Reverse color palette of visuals*.
7. Enable panning and zooming option, as described in *Customizing zoom*.

8. Click REFRESH VISUAL.

   The map visual appears.

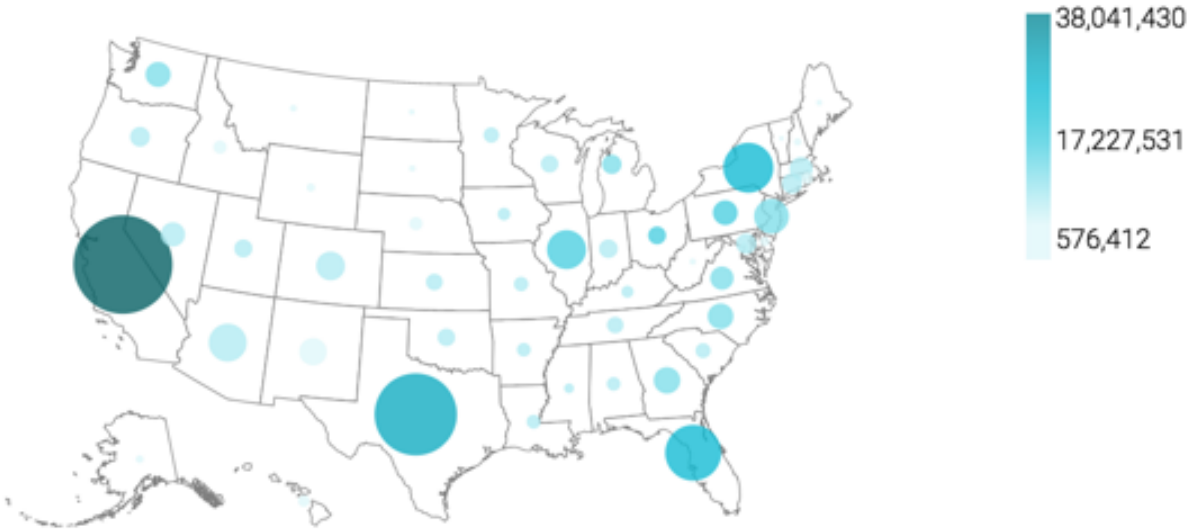

9. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'US State Population - Bubbles'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

10. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

You can move the visual (pan), zoom in and out, and see the details in the Tooltips text.

Choropleth US county map

In Cloudera Data Visualization Map visuals, you can plot measurements as areas over US County regions.

About this task

A county map of the United States shows a comparison of measurement values across US States. The following steps extend the work completed in *State map, choropleth*, and demonstrate how to create a map visual on the dataset US County Populations [data source samples.us_counties]. It produces colored regions for the measurements that correspond to the county field.

Procedure

1. Clone the US State Population - Choropleth Map visual that you developed in *State map, choropleth*.
2. Under Dimensions, select ctyname and add it to the Geo shelf.
3. Click the field to open the FIELD PROPERTIES menu, expand the Change Type option, and select the Geo Type County.
4. Alias the field ctyname as County.

   For instructions, see *Alias*.

5. Click REFRESH VISUAL.
   The map visual appears.

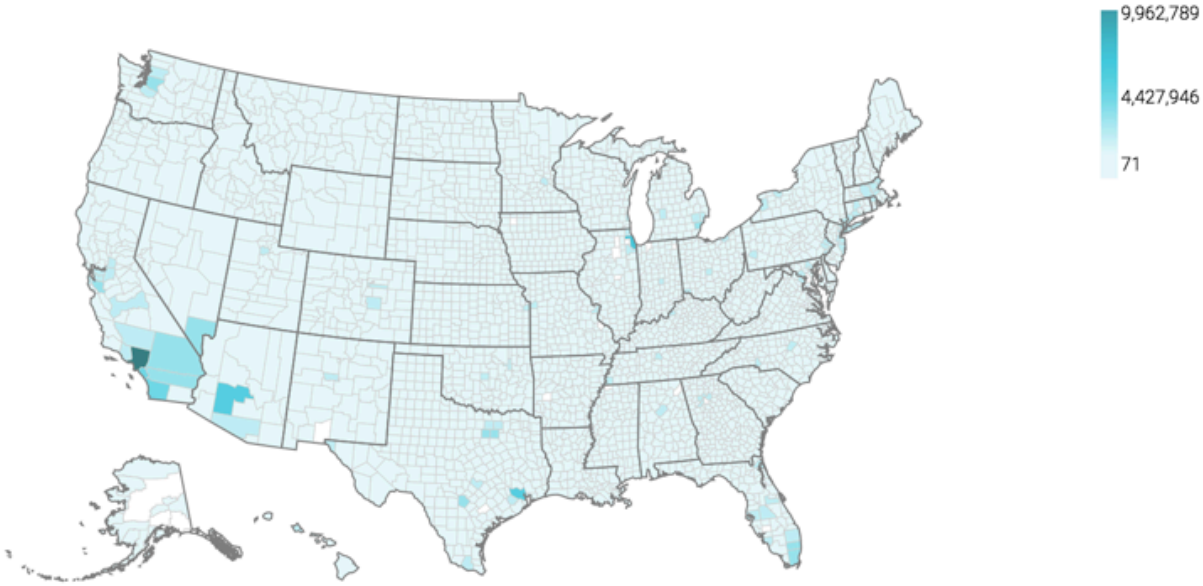

6. If not already selected, select the Enable pan and Zoom and Zoom to active area options, as described in *Customizing zoom*.
7. Under Dimensions, select stname and add it to the Filters shelf. When the Filter for stname window modal appears, select California in the Values tab, and click APPLY.

8. Click REFRESH VISUAL.

   The map automatically zooms in to show detail of California.

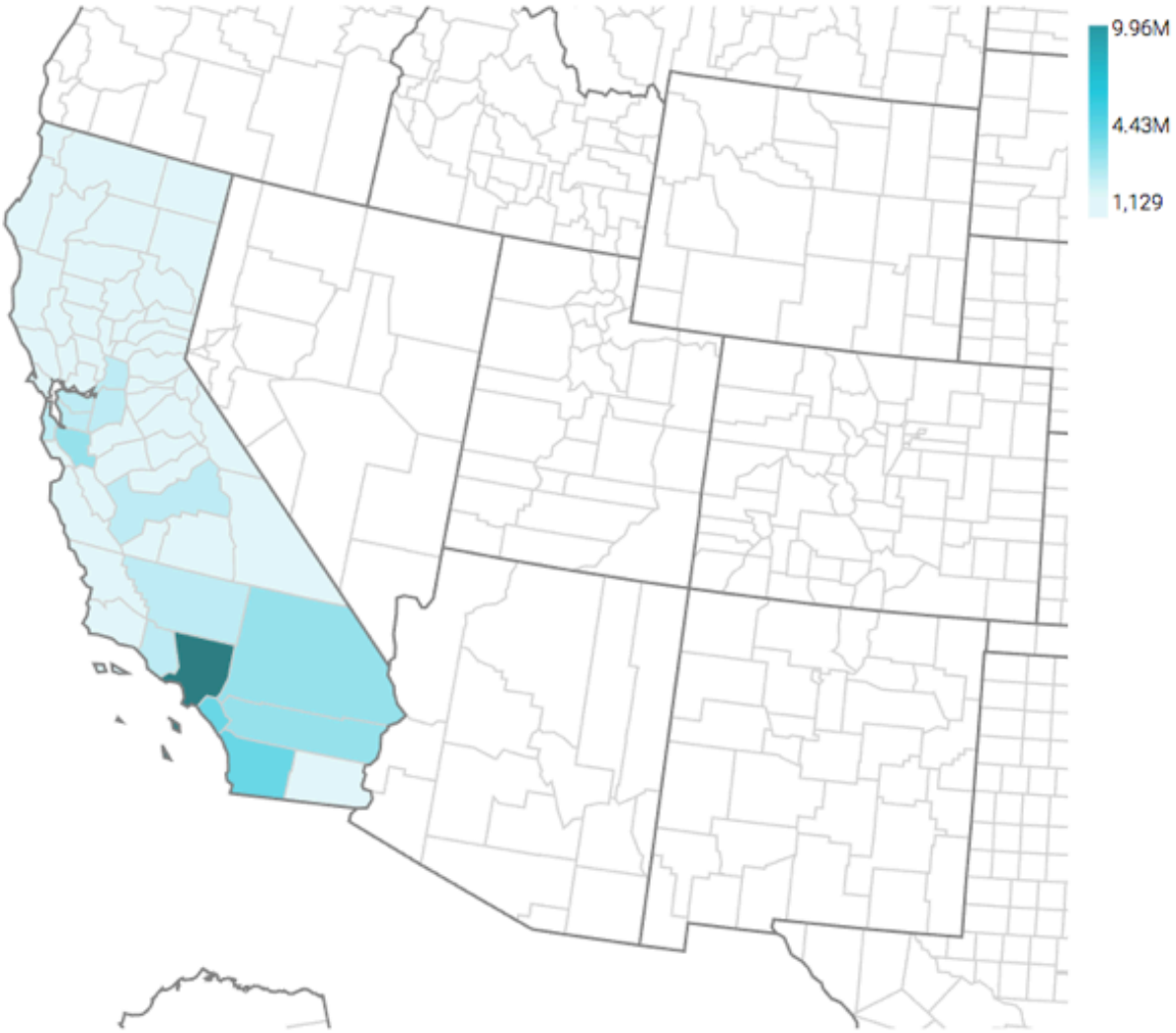

9. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'US County Population - Choropleth Map'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

10. At the top left corner of the Dashboard Designer, click SAVE.

US county map with bubbles

Cloudera Data Visualization Map visuals let you plot measurements, as bubbles, over US County regions. Bubble maps display up to two measurements simultaneously: as the color of the bubble, and as the relative size of the bubble.

About this task

A county map of the United States shows a comparison of measurement values across US States. The following steps extend the work completed in *State map with bubbles* and demonstrate how to create a map visual on the dataset US County Populations [data source samples.us_counties]. This produces a map with bubble marks that correspond to the total population of each county.

Procedure

1. Clone the visual that you developed in *Country map, choropleth*.

   For instructions, see *Cloning a visual*.
2. Under Dimensions, select ctyname and add it to the Geo shelf.
3. Click the field to open the Field Properties menu, expand the Change Type option, and select the Geo Type County.
4. Alias the field ctyname as County.

   For instructions, see *Alias*.
5. If the Size shelf is empty, add a measure to it.

   In this example the goal is to show the relative size of the Hispanic population across the country, so the following expression is used in the Enter/Edit Expression editor:

   ```
   sum([h_female]+[h_male])
   ```

   For more information, see *Enter or edit expressions*.
6. Alias all the fields.

   For instructions, see *Alias*.
7. Change the color palette or simply reverse it, as described in *Change color palette of visuals* and *Reverse color palette of visuals*.

8. Click REFRESH VISUAL.

   The map visual appears. Note that if you have Zoom enabled, the map automatically zooms in to show detail of California. For more information, see *Customizing zoom*.

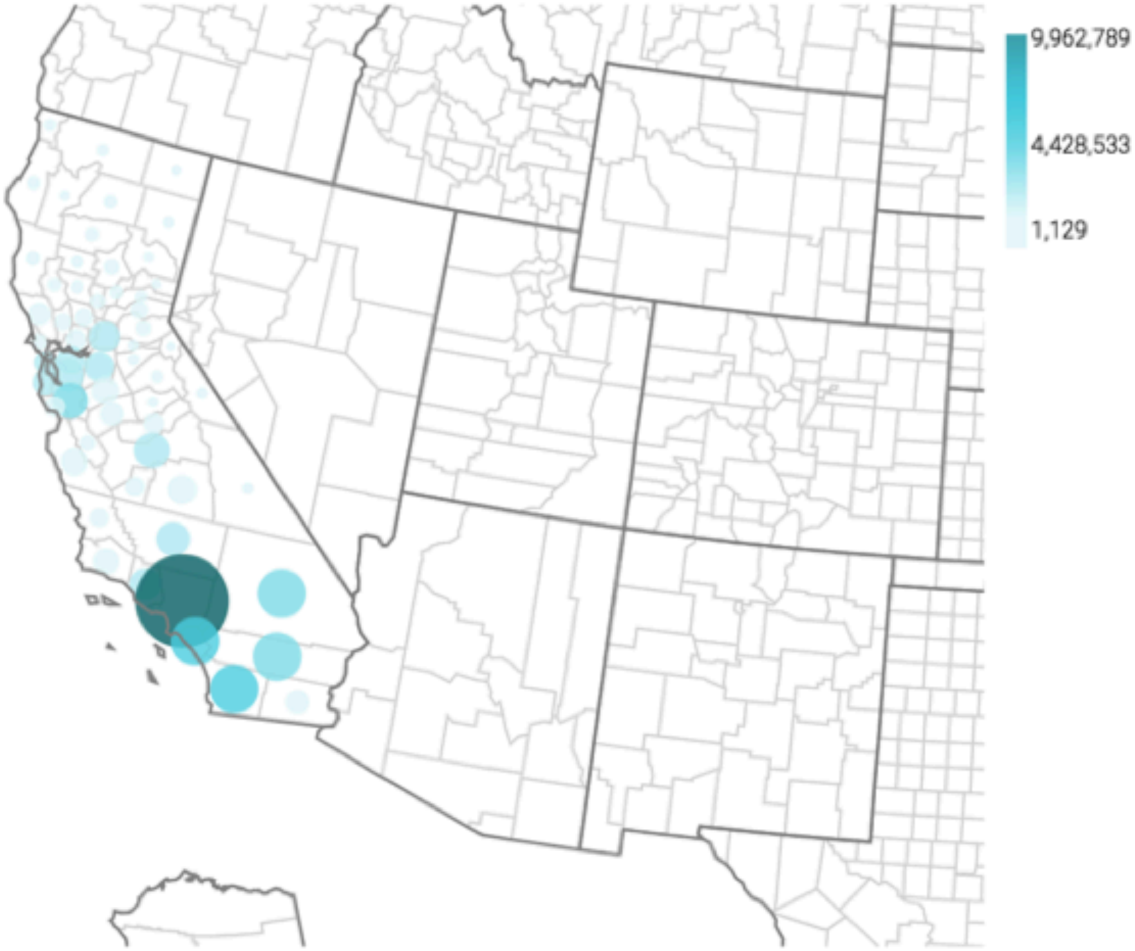

9. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'US County Population - Bubbles'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

10. At the top left corner of the Dashboard Designer, click SAVE.

US zipcode map with bubbles

In Cloudera Data Visualization Map visuals, you can plot measurements based on zipcode data. Bubble maps display up to two measurements simultaneously: as the color of the bubble, and as the relative size of the bubble.

About this task

A zip code map of the United States shows a comparison of measurement values across US States. Zipcode maps only work with bubble options. The Income Tax Returns dataset is used in this example, which is built on the tax_returns.csv datafile that contains statistical information about Personal Income Tax Returns for 2016 (zip_tax_returns_2016.csv), summarized by zipcode.

Procedure

1. Start a new visual based on the Income Tax Returns dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Map.

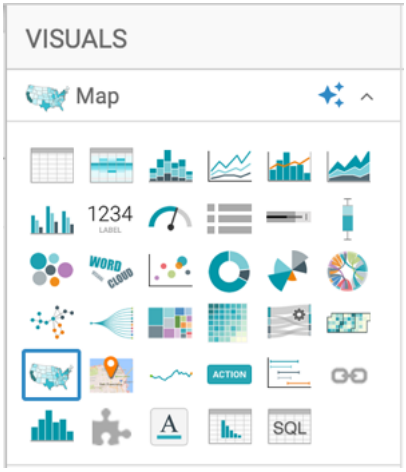

The shelves of the visual changed. They are now Geo, Color, Size, Drill, Tooltips, X Trellis, Y Trellis, and Filters. The only mandatory shelf for map visuals is Geo.

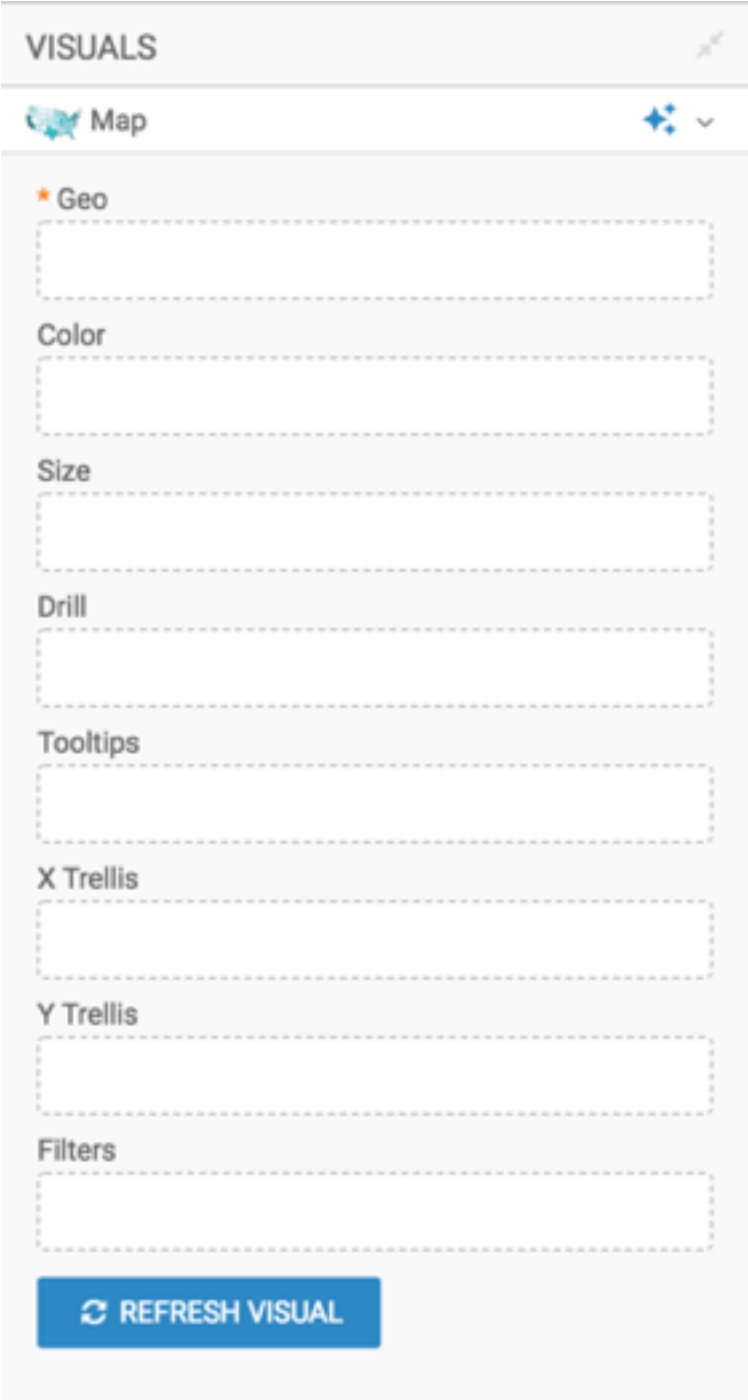

3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Select Zipcode and add it to the Geo shelf.
      The field is automatically accepted as a Geo Type.
   b) Apply the display format for integers, removing the 1000s separator.
      For instructions, see *Integer display format*.
   c) In the DATA menu, enter the term 'schf' in the Search box, and add the field to the Size shelf.
   d) Alias the field as Farm Returns.
   e) Select State and add it to the Filters shelf. When the Filter for State window modal appears, select the South Atlantic region states under the Values tab: DC (District of Columbia), DE (Delaware), FL (Florida), GA

(Georgia), MD (Maryland), NC (North Carolina), SC (South Carolina), VA (Virginia), and WV (West Virginia). Click Apply.

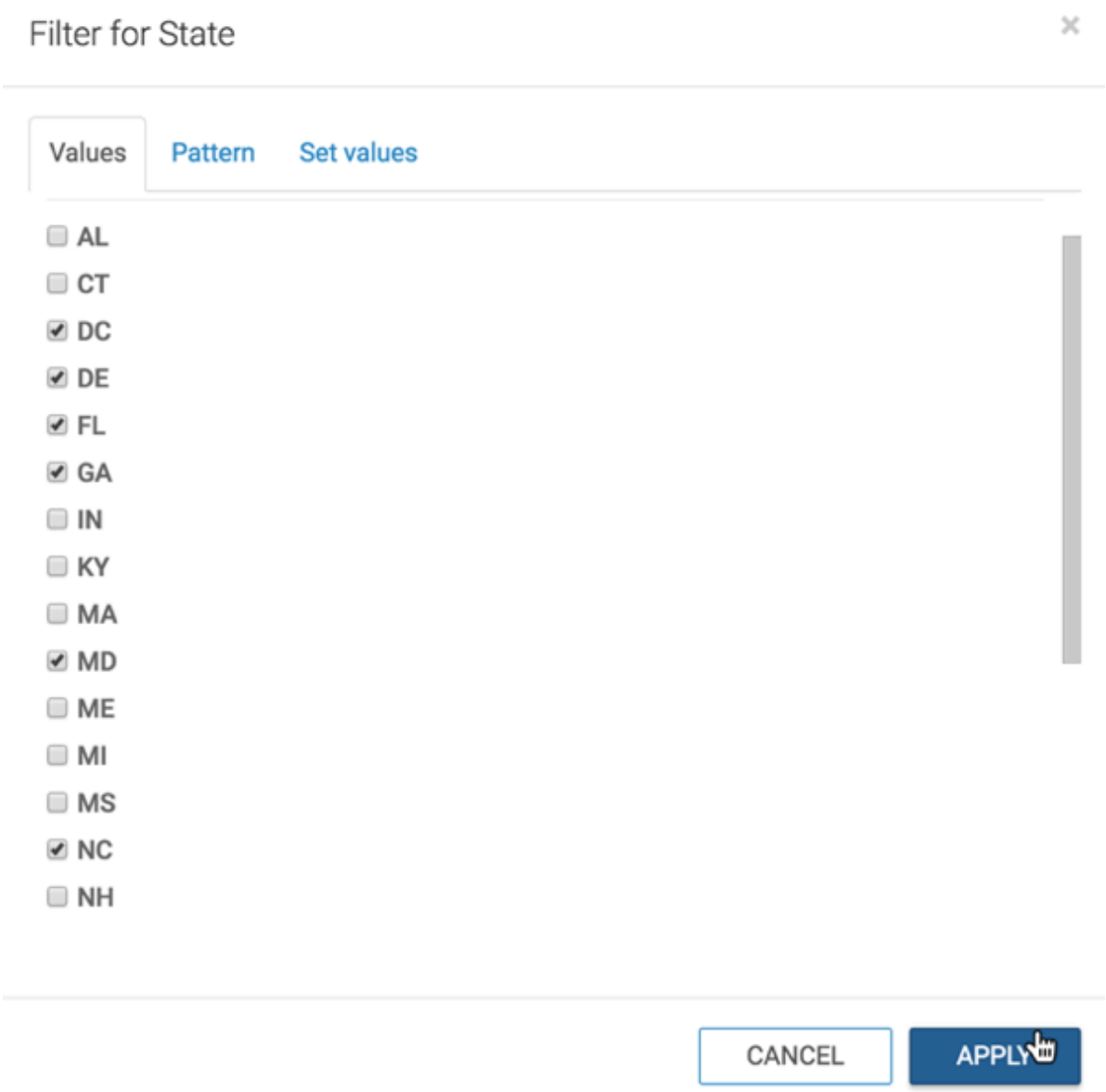

4. Adjust the size range of the marks to 1-3, as described in *Changing the mark size range*.
5. Change the color palette or simple reverse it, as described in *Change color palette visuals* and *Reverse color palette visuals*.
6. Enable panning and zooming option, as described in *Customizing zoom*.

7. Click REFRESH VISUAL.

   The map visual appears. You can adjust the size and focus of the visual to show the area of interest.

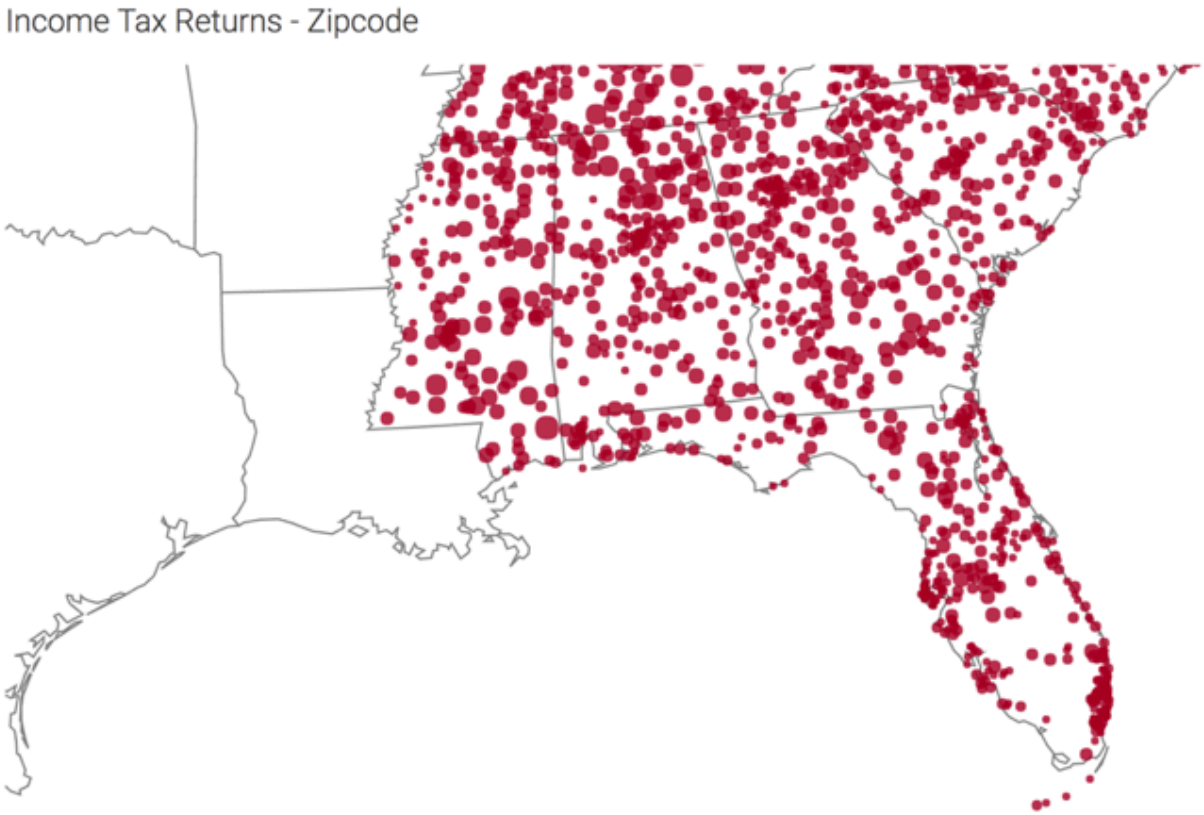

8. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'Income Tax Returns - Zipcode'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

9. At the top left corner of the Dashboard Designer, click SAVE.

Latitude and longitude map with bubbles

Cloudera Data Visualization Map visuals support measurement plots over a geographic map, based on latitude and longitude. Bubble maps display up to two measurements simultaneously: as the color of the bubble, and as the relative size of the bubble.

About this task

A geographic coordinate map of the United States, based on degrees of latitude and longitude, shows a comparison of measurement values across US States. The following steps extend the work completed in *US zipcode map with bubbles*, and demonstrate how to create a map visual on the Income Tax Returns dataset. It produces maps with bubble marks that correspond to the relative size of the measurement at each latitude and longitude pair.

Before starting this exercise, supplement the Income Tax Returns dataset by adding data from zipcode-lat-long.csv into your database, and making a simple joins between zip_tax_returns_2016.zipcode and zipcode_lat_long.zipcode. For instructions, see *Data model: joins*.

Procedure

1. Clone the Income Tax Returns - Zipcode visual.

2. Change the fields on the shelves:
   a) Remove all the fields from the Filters shelf.
   b) Remove the field Zipcode from the Geo shelf.
   c) In the DATA menu, enter the term 'latitude' in the Search box, and add the Latitude field to the Geo shelf.
   d) Search for term 'longitude', and add the Longitude field to the Geo shelf.
3. Open the Settings menu, expand the Map menu, and select the United Sates Map View option.

   For more information, see *Map view*.
4. Click REFRESH VISUAL.

   The map visual appears. You can move the visual (pan), zoom in and out, and see the details in the Tooltips text.

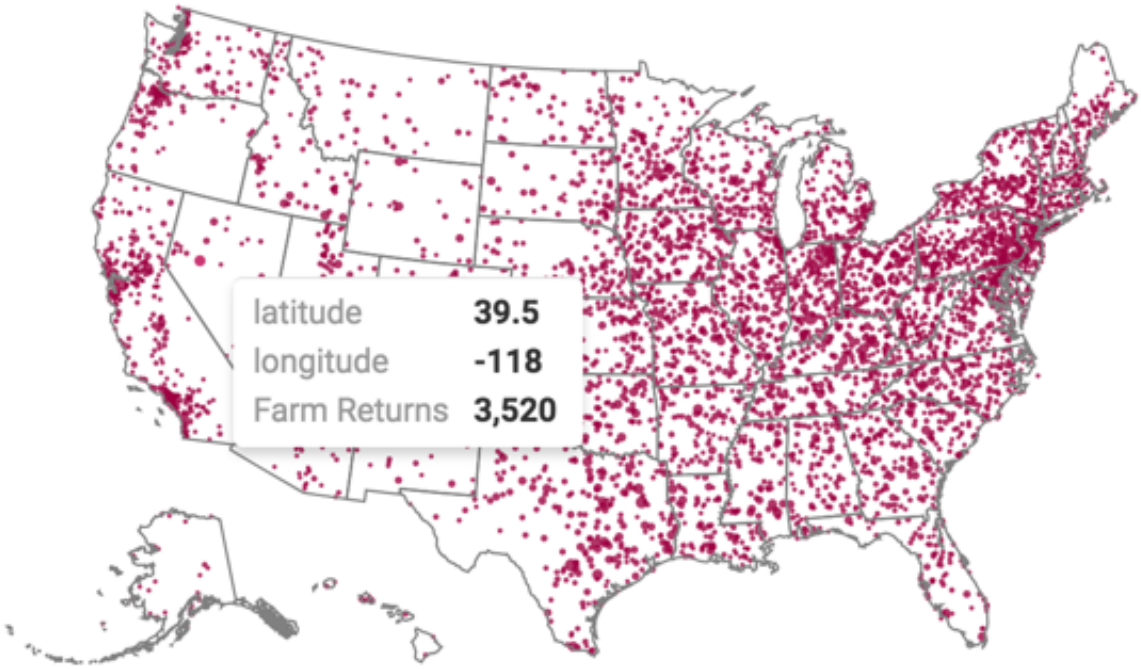

5. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'Income Tax Returns - Lat and Long'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
6. At the top left corner of the Dashboard Designer, click SAVE.

Choropleth DMA map

In Cloudera Data Visualization Map visual, you can visualize measurements on a map of DMA regions. A DMA map shows various measurements plotted relative to Nielsen Designated Market Area Regions.

About this task

In this example, the DMA TV Home Viewing is used, which is built on the dma-tv-home-viewing.csv datafile.

Procedure

1. Start a new visual based on the DMA TV Home Viewing dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Map.

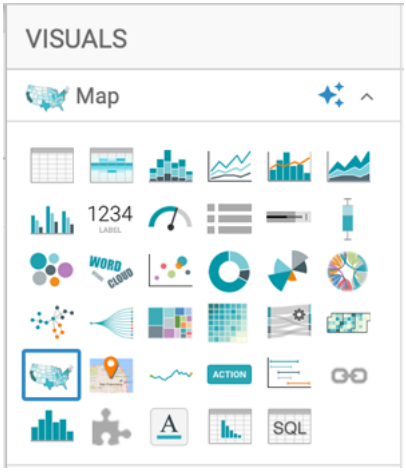

The shelves of the visual changed. They are now Geo, Color, Size, Drill, Tooltips, X Trellis, Y Trellis, and Filters. The only mandatory shelf for map visuals is Geo.

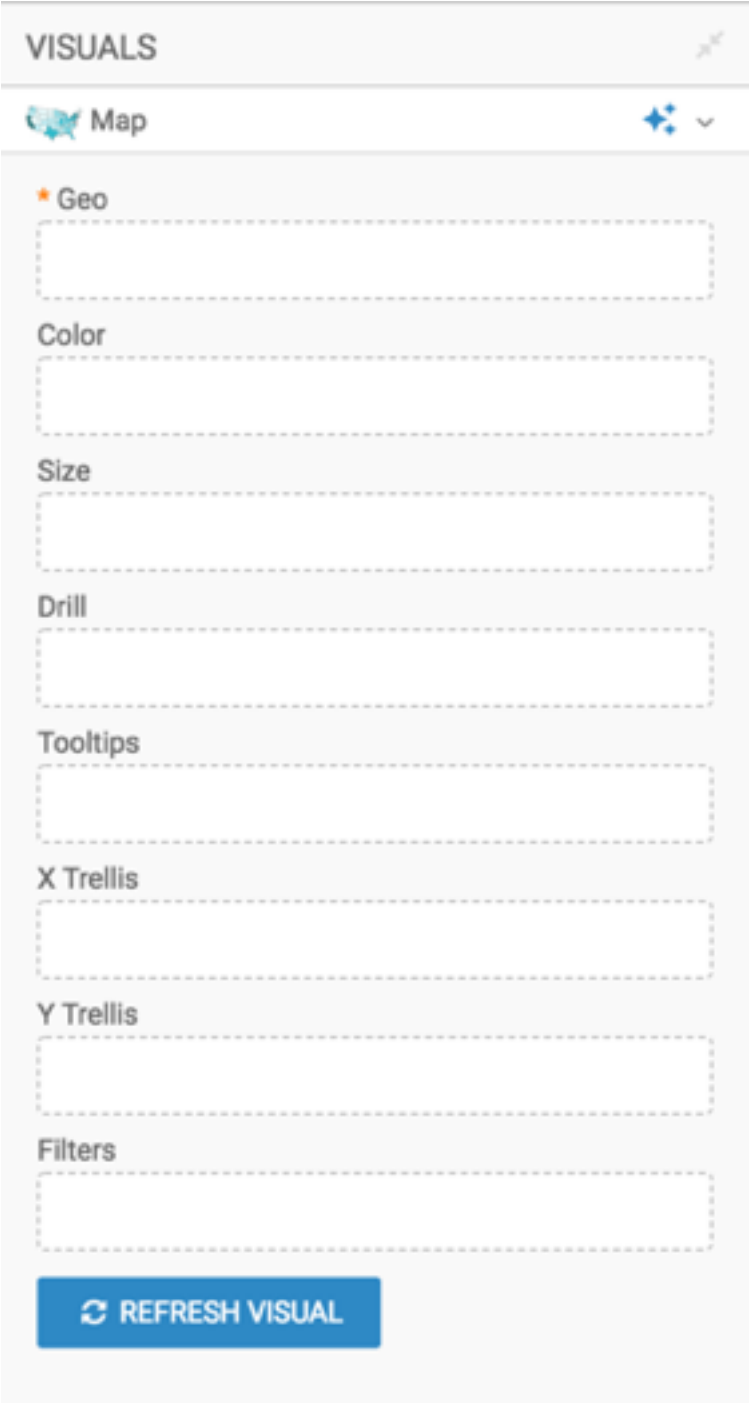

3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   - Under Dimensions, select dma and add it to the Geo shelf.
   - Under Measures, select homes and add it to the Color shelf.
4. Alias the field dma as DMA, and sum(homes) as Number of Homes.
   For instructions, see *Alias*.
5. Add the dimension dma_name to the Tooltips shelf, and then alias it as DMA Name.
6. Enable panning and zooming option, as described in *Customizing zoom*.
7. Change the color palette, as described in *Change color palette visuals*.

8. Click REFRESH VISUAL.

   The map visual appears. You can move the visual (pan), zoom in and out, and see the details in the Tooltips text.

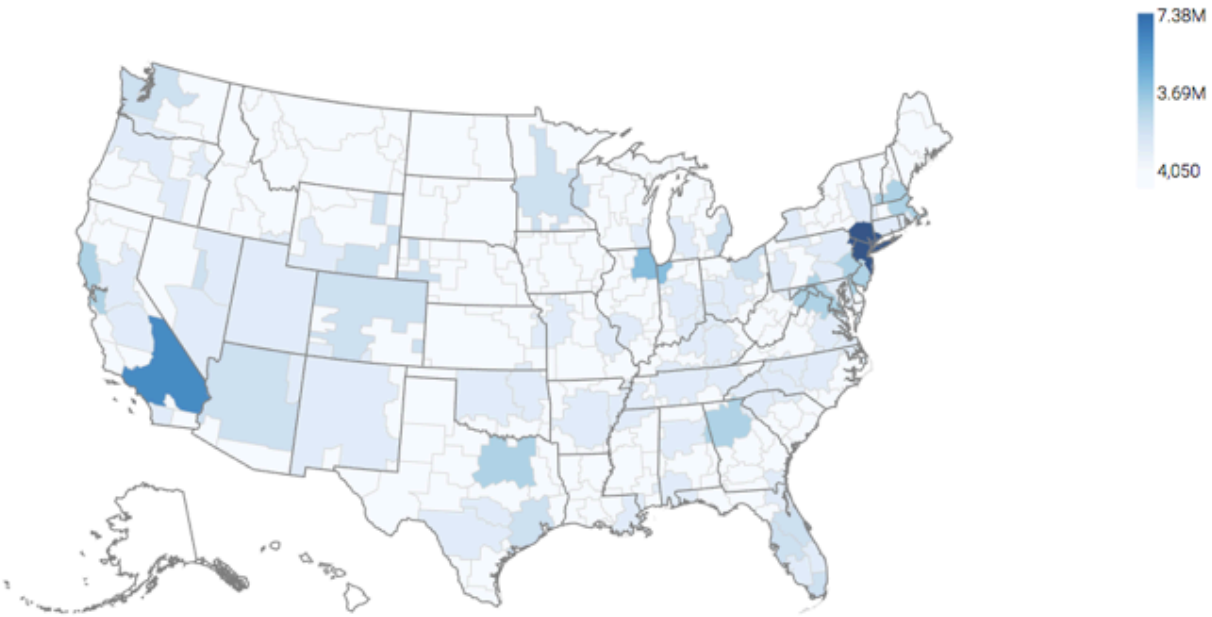

9. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'US DMA TV Viewing Map'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

10. At the top left corner of the Dashboard Designer, click SAVE.

Choropleth Canadian province map

Cloudera Data Visualization Map visuals support choropleth mapping of measurements over a map of Canadian Provinces and Territories.

About this task

A map of Canada shows a comparison of measurement values among Canadian Provinces and Territories. The following steps demonstrate how to create a new choropleth map of Canada. This example uses the Canadian Census dataset, constructed from the canada_census_population_dwellings.csv and ca-postal-codes.csvfiles, and joined of the fields fsa and Postal Code, respectively. For more information, see *Canadian census*, *Adding data*, and *Creating datasets*.

Tip: Remember that to map province information, the dimension must be recognized as a Geo Type. For more information, see *Specifying geographic fields*.

Procedure

1. Start a new visual based on the Canadian Census dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Map.

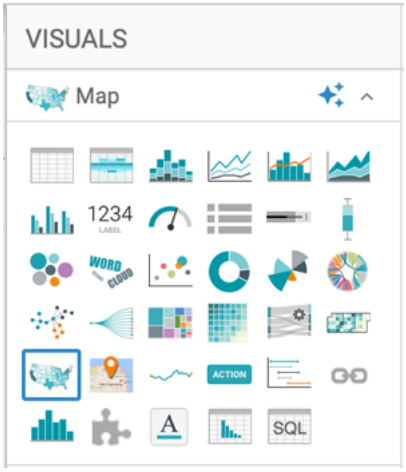

The shelves of the visual changed. They are now Geo, Color, Size, Drill, Tooltips, X Trellis, Y Trellis, and Filters. The only mandatory shelf for map visuals is Geo.

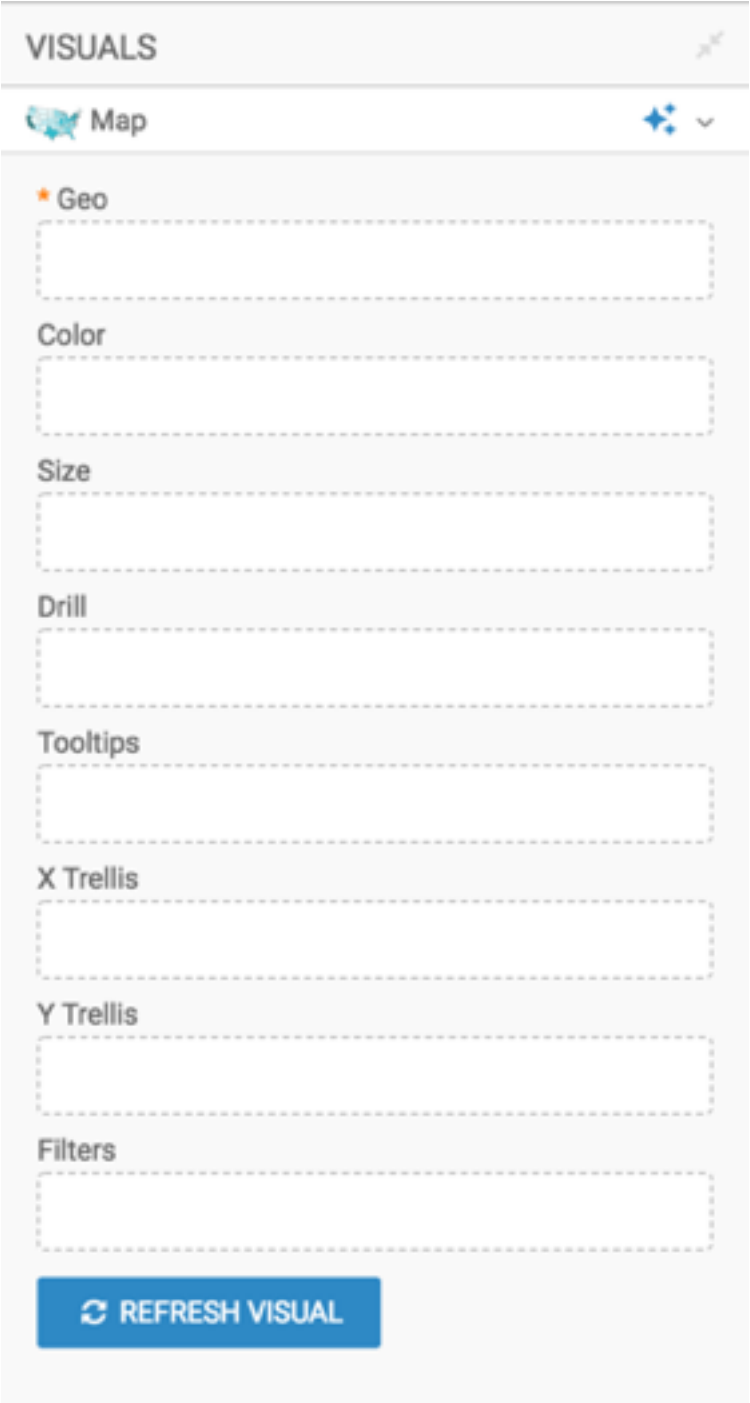

3. Populate the shelves from the DATA menu:
   a) Under Dimensions, select Province from the table ca_postal_codes and add it to the Geo shelf.
   b) Under Measures, select Population 2011 from table canada_census_population_dwellings and add it to the Colors shelf.
4. Specify the use of Canada Map View in the Settings Map interface.

   For instructions, see *Map view*.
5. Click REFRESH VISUAL.
6. Alias the fields as necessary; see *Alias*.
7. Enable panning and zooming option, as described in *Customizing zoom*.
8. Change the color palette, as described in *Changing color palette of visuals*.

9. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'Canadian 2011 Population - Province Choropleth'. You can also add a brief description of the visual as a subtitle below the title of the visualization.

10. At the top left corner of the Dashboard Designer, click SAVE.

Results

Your visual should look something like this:

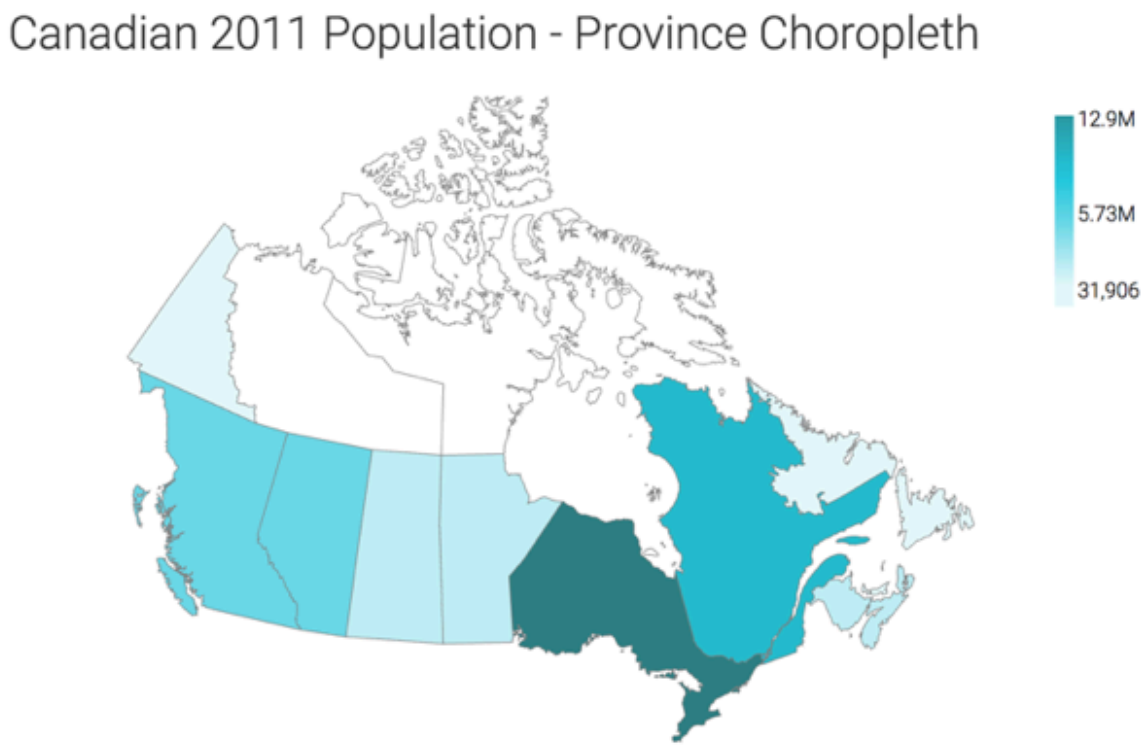

Canadian province map, bubbles

Cloudera Data Visualization Map visuals enable you to show measurements, as bubbles, over a map of Canadian Provinces and Territories. Bubble maps display up to two measurements simultaneously: as the color of the bubble, and as the relative size of the bubble.

About this task

A map of Canada shows a comparison of measurement values among Canadian Provinces and Territories. The following steps demonstrate how to create a new choropleth map of Canada. This example uses the Canadian Census dataset, constructed from the canada_census_population_dwellings.csv and ca-postal-codes.csvfiles, and joined of the fields fsa and Postal Code, respectively. For more information, see *Canadian census*, *Adding data*, and *Creating datasets*.

Tip: Remember that to map province information, the dimension must be recognized as a Geo Type. See *Specifying geographic fields*.

Procedure

1. Start a new visual based on the Canadian Census dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Map.

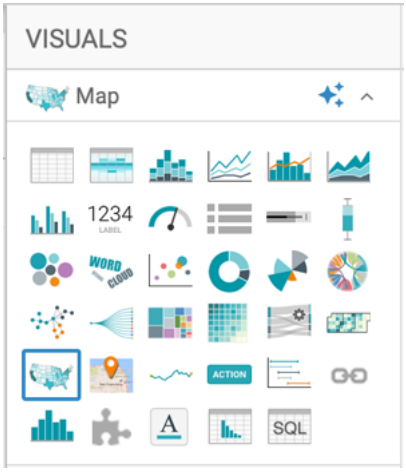

The shelves of the visual changed. They are now Geo, Color, Size, Drill, Tooltips, X Trellis, Y Trellis, and Filters. The only mandatory shelf for map visuals is Geo.

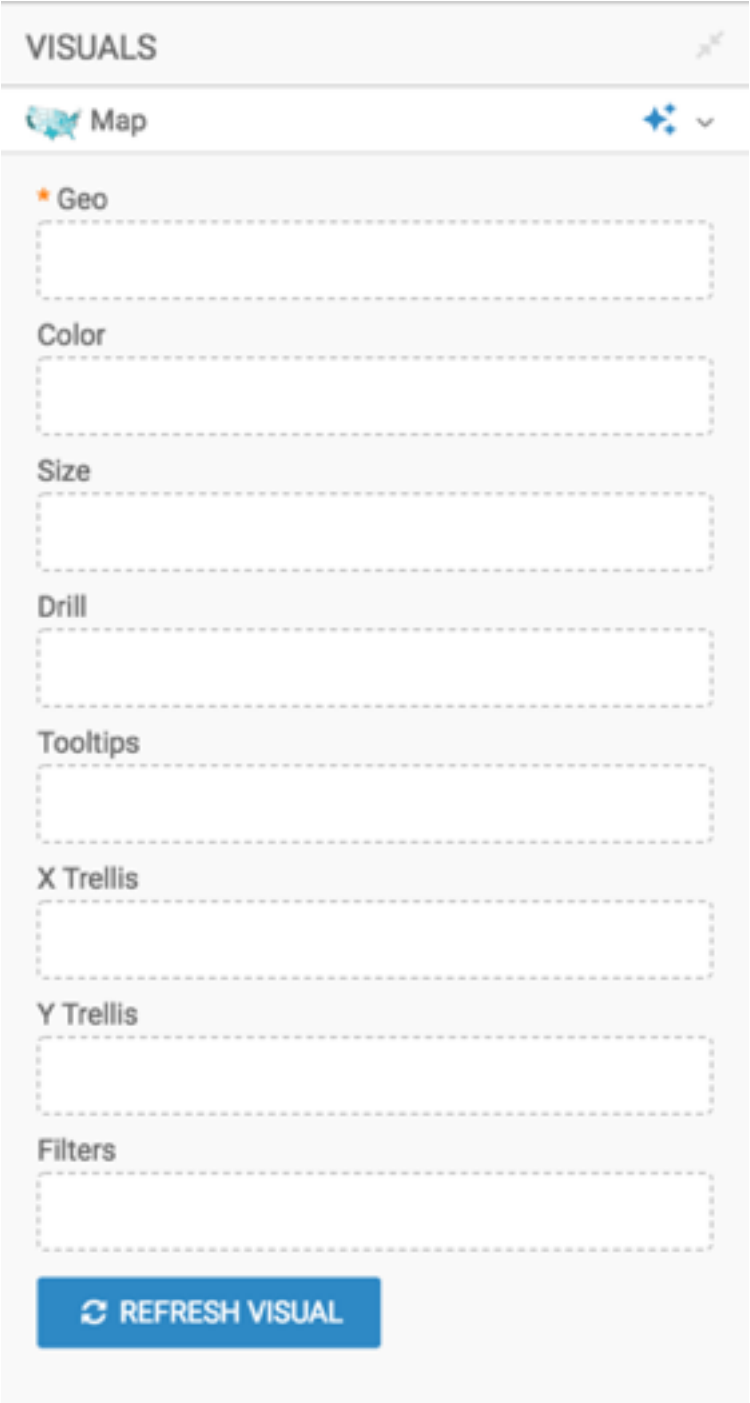

3. Populate the shelves from the DATA menu:
   a) Under Dimensions, select Province from the table ca_postal_codes, and add it to the Geo shelf.
   b) Under Measures, select Population 2011 from table canada_census_population_dwellings, and add it to the Measures shelf.
4. Specify the use of Canada Map View in the Settings Map interface.

   For instructions, see *Map view*.
5. Click REFRESH VISUAL.
6. Change the display of marks to bubbles as described in *Displaying bubbles on maps*.
7. Change the mark size range to 1-30 as described in *Changing the mark size range*.

8. You can change or simply reverse the color palette to suit your needs.

   For instructions, see *Change color palette of visuals* and *Reverse color palette of visuals*.
9. Alias the fields as necessary.

   For instructions, see *Alias*.
10. Enable panning and zooming option, as described in *Customizing zoom*.
11. Change the color palette, as described in *Change color palette of visuals*.
12. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

    In this example, the title is changed to 'Canadian 2011 Population - Province Bubbles'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
13. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

Your visual should look something like this:

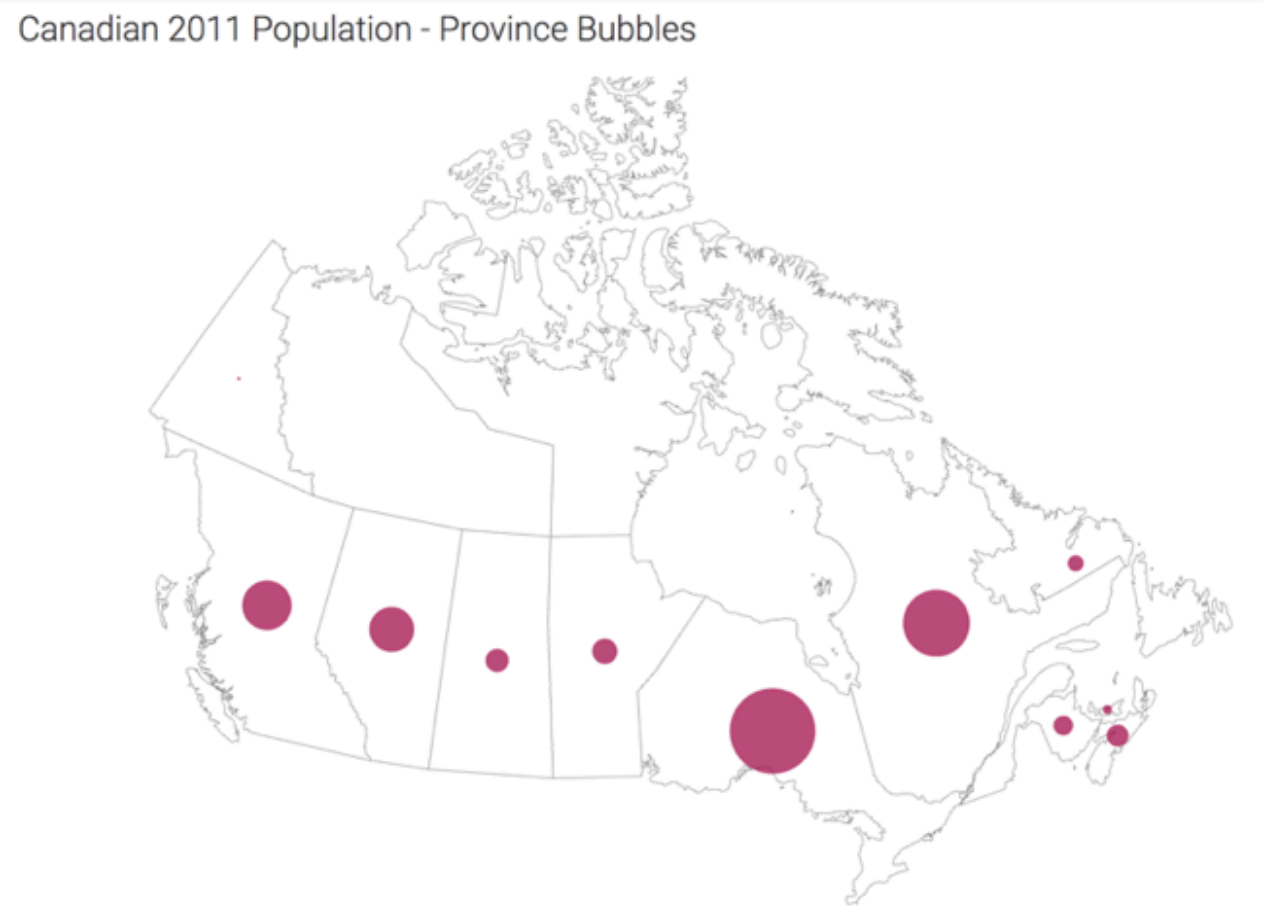

Choropleth Canadian postal code map

Cloudera Data Visualization Map visuals support choropleth mapping of measurements over a map of Canadian FSA and Postal Codes.

About this task

A map of Canada shows a comparison of population density, at the level of the FSA (1st 3 digits of the Postal Code). The following steps demonstrate how to create a new choropleth map of Canada. This example uses the Canadian Census dataset, constructed from the canada_census_population_dwellings.csv and ca-postal-codes.csvfiles, and

joined of the fields fsa and Postal Code, respectively. For more information, see *Canadian census*, *Adding data*, and *Creating datasets*.

Tip: Remember that to map province information, the dimension must be recognized as a Geo Type. See *Specifying geographic fields*.

Procedure

1. Start a new visual based on the Canadian Census dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Map.

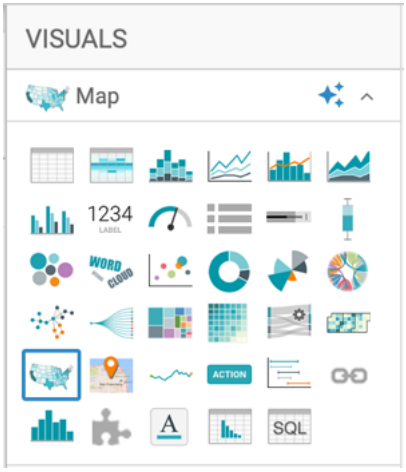

The shelves of the visual changed. They are now Geo, Color, Size, Drill, Tooltips, X Trellis, Y Trellis, and Filters. The only mandatory shelf for map visuals is Geo.

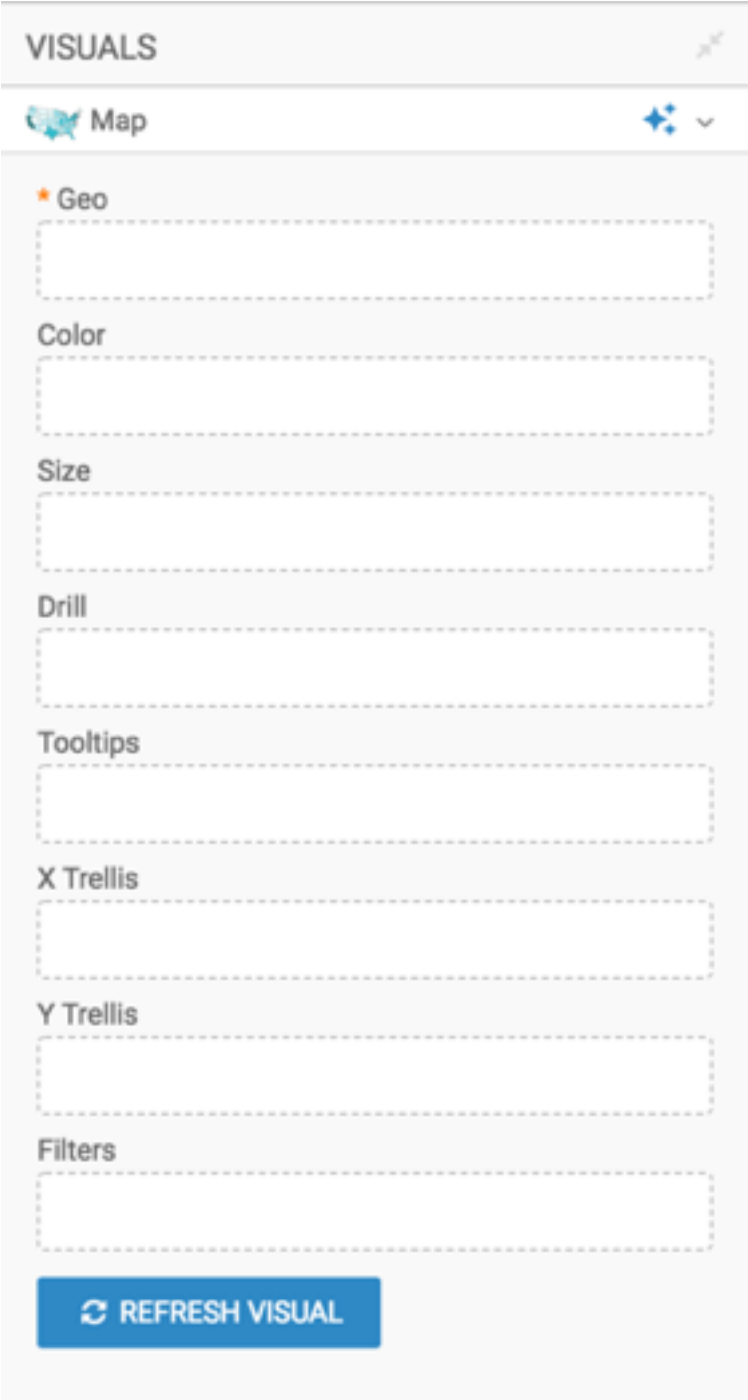

3. Populate the shelves from the DATA menu:
   a) Under Dimensions, select Postal Code from the table ca_postal_codes, and add it to the Geo shelf.
   b) Under Measures, select Population 2011 from table canada_census_population_dwellings, and add it to the Color shelf.
4. Specify the use of Canada Map View in the Settings Map interface.

   For instructions, see *Map view*.
5. Click REFRESH VISUAL.
6. Alias the fields as necessary.

   For instructions, see *Alias*.
7. Enable panning and zooming option, as described in *Customizing zoom*.

8. Change the color palette, as described in *Change color palette of visuals*.
9. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'Canadian 2011 Population - Postal Code Choropleth'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
10. At the top left corner of the Dashboard Designer, click SAVE.

Results

Your visual should look something like this:

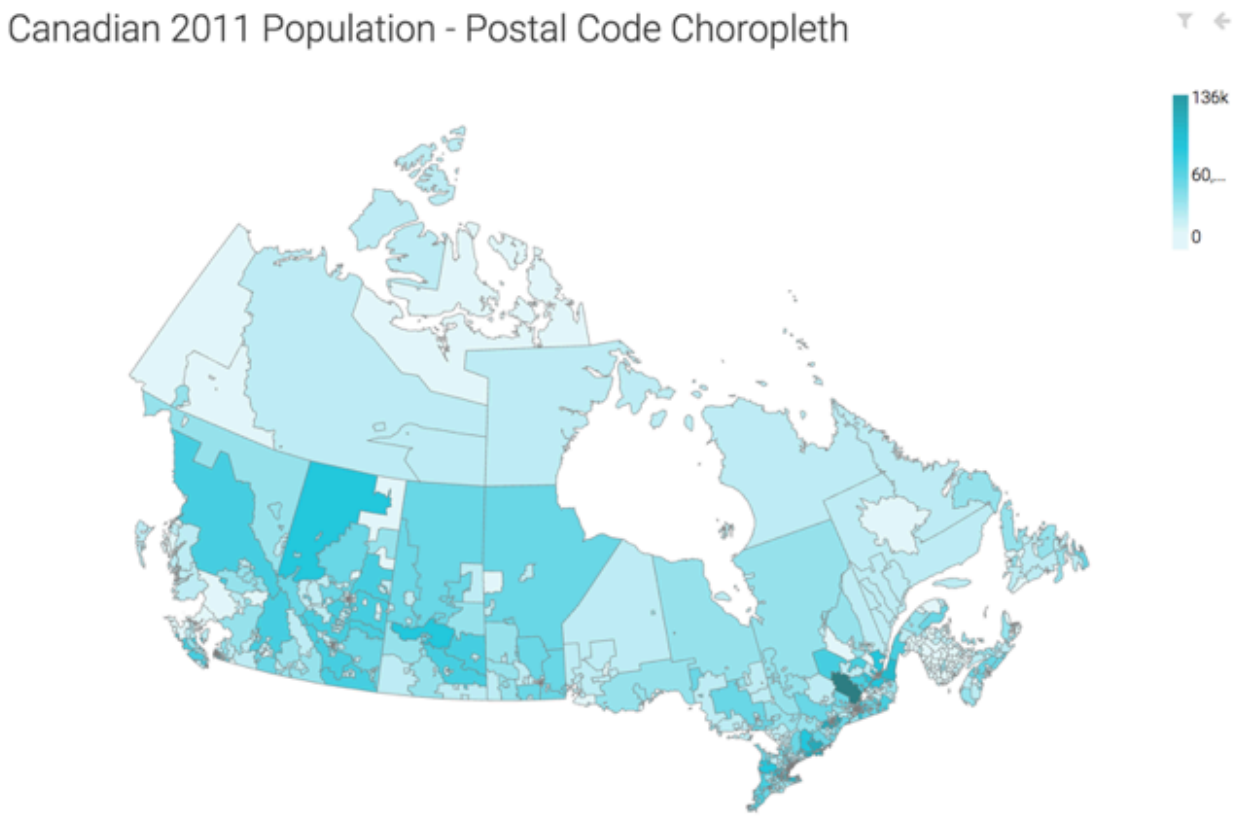

Canadian postal code map, bubbles

Cloudera Data Visualization Map visuals support choropleth mapping of measurements over a map of Canadian FSA and Postal Codes. Map visual provides support for mapping measurements, as bubbles, over a map of Canadian FSA and Postal Codes. Bubble maps display up to two measurements simultaneously: as the color of the bubble, and as the relative size of the bubble.

About this task

A map of Canada shows a comparison of population density, at the level of the FSA (1st 3 digits of the Postal Code). The following steps demonstrate how to create a new choropleth map of Canada. This example uses the Canadian Census dataset, constructed from the canada_census_population_dwellings.csv and ca-postal-codes.csvfiles, and joined of the fields fsa and Postal Code, respectively. For more information, see *Canadian census*, *Adding data*, and *Creating datasets*.

Tip: Remember that to map province information, the dimension must be recognized as a Geo Type. See *Specifying geographic fields*.

Procedure

1. In the VISUALS menu, find and click Map.

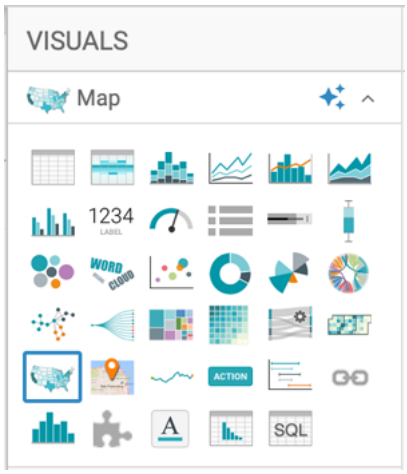

The shelves of the visual changed. They are now Geo, Color, Size, Drill, Tooltips, X Trellis, Y Trellis, and Filters. The only mandatory shelf for map visuals is Geo.

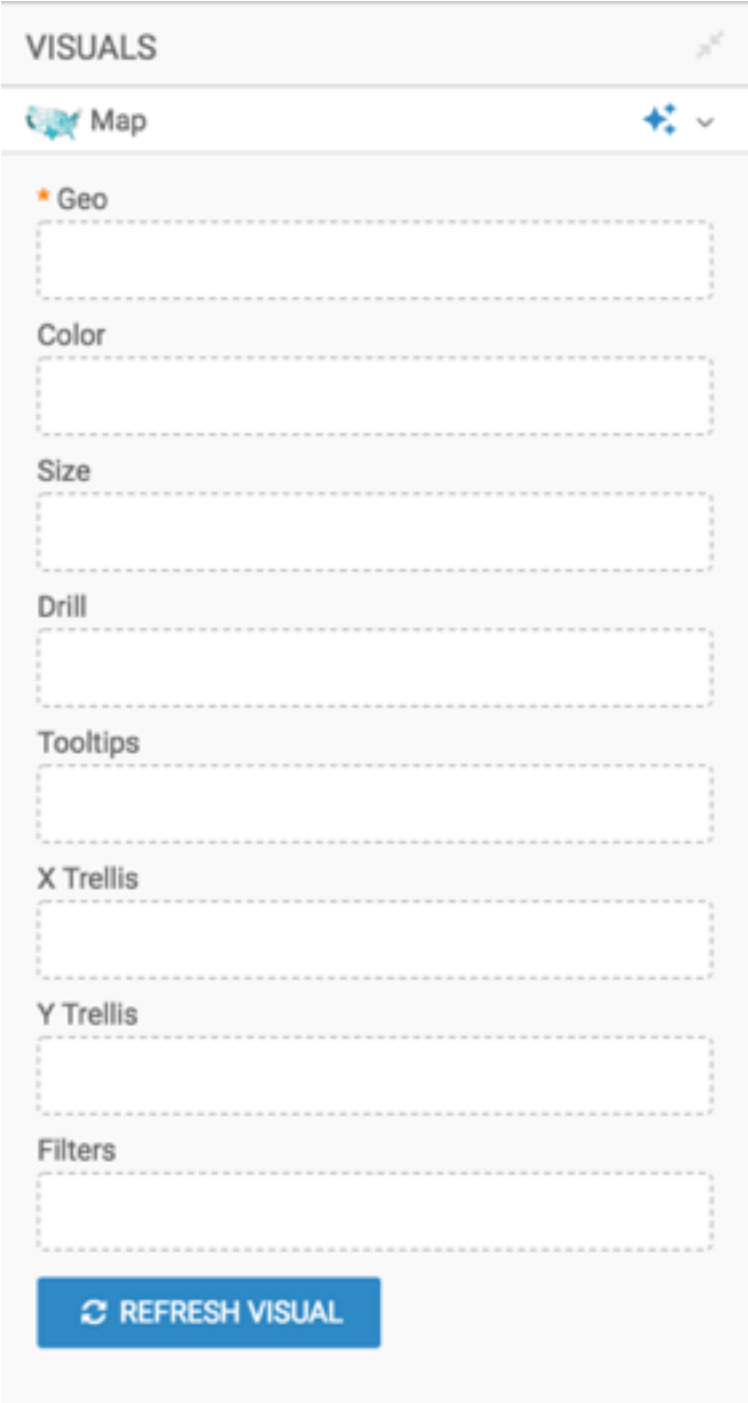

2. Populate the shelves from the DATA menu:
   a) Under Dimensions, select Postal Code from the table ca_postal_codes, and add it to the Geo shelf.
   b) Under Measures, select Population 2011 from table canada_census_population_dwellings, and add it to the Color shelf.
3. Specify the use of Canada Map View in the Settings Map interface.

   For instructions, see *Map view*.
4. Click REFRESH VISUAL.
5. Change the display of marks to bubbles as described in *Displaying bubbles on maps*.
6. Change the mark size range to 1-10 as described in *Changing the mark size range*.

7. You can change or simply reverse the color palette to suit your needs.

   For instructions, see *Change color palette of visuals* and *Reverse color palette of visuals*.
8. Alias the fields as necessary.

   For instructions, see *Alias*.
9. Enable panning and zooming option, as described in *Customizing zoom*.
10. Change the color palette, as described in *Change color palette of visuals*.
11. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

    In this example, the title is changed to 'Canadian 2011 Population - Postal Code Bubbles'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
12. At the top left corner of the Dashboard Designer, click SAVE.

Results

Your visual should look something like this:

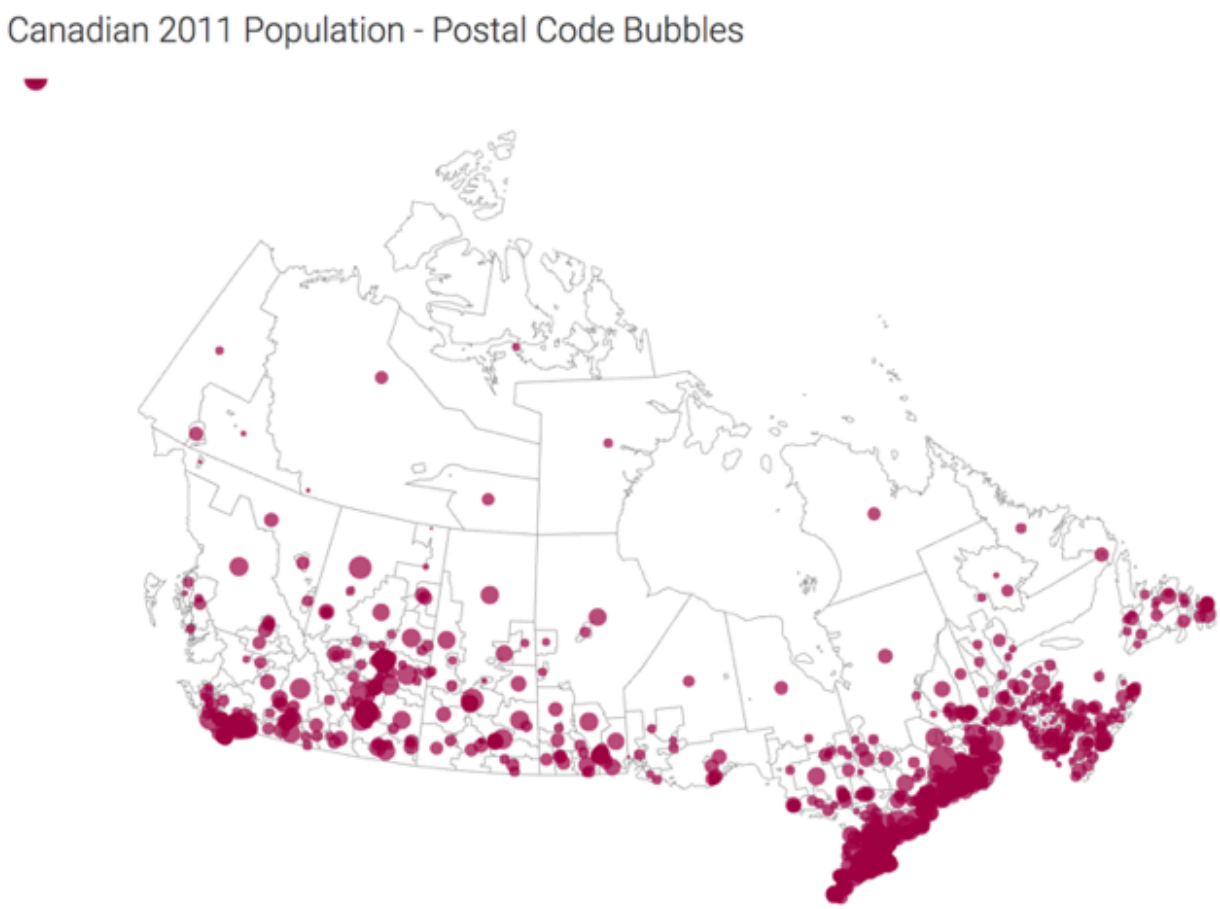

Shelves for maps

Overview of shelves for Cloudera Data Visualization Map visuals.

A Map visual has the following shelves:

Geo

Add a Geo-type field from the DATA menu to determine the location on the map.

Geo types include country, state, county, DMA, zipcode or postal code, or latitude and longitude. If the field is not automatically recognized as a qualified Geo type, select Change Type from the FIELD PROPERTIES menu.

This shelf accepts multiple fields, but plots a map with the highest possible resolution.

This is mandatory shelf.

Color

Add a Measure field from the DATA menu to aggregate values across locations.

This shelf displays only one field on the graph. Additional fields appear only in the Tooltip.

Size

Add a Measure field from the DATAmenu to aggregate values across locations.

This shelf displays only one field on the graph. Additional fields appear only in the Tooltip.

Drill

Add Dimension fields from the DATA menu to this shelf; use Geo-type fields only.

When Drill Into context menu is enabled but no fields are specified, you can drill into any dimension. Specifying the fields on this shelf limits drilling to these dimensions.

This shelf accepts multiple fields.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

X Trellis

Use a Dimension field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Interactive maps

An interactive map paints geographic details over a basic map tile layer provided by a third-party provider. Cloudera Data Visualization offers several overlay layers for data display. It is an excellent choice for displaying large amounts of geo-based information in relevant detail.

- The base Map Servers include either Google Maps or Mapbox.
- The overlay layers that display data include the Heatmap, Cluster, Circles, and Routes and pins options.

Note:

- To use this feature, an Admin user must first specify, through the Site Settings interface, the API keys granted to your organization. This is described in *Keys for interactive visuals*.
- This visual type supports zero, one, or two field aggregations on the measures shelf. If no measure is supplied, the map will paint with an assumed value of 1. With one measure, all options use that measure. With two measures, the Heatmap, Cluster and Routes & Pins use the first measure only, while the Circles display both measures.
- Remember that to place information onto a map or interactive map visual, the records must contain geographical information, such as valid latitude and longitude pairs, zipcodes, states, and so on. To learn how to cast dataset fields into usable type, see *Change type* and *Geo type*.

To learn how to use the interactive maps in their various forms, read the following topics.

Basic interactive map

Cloudera Data Visualization enables you to create a basic Interactive Map visual.

About this task

The following steps demonstrate how to create an interactive map visual on the National Geographic Features dataset. The data for this dataset comes from the United States Board on Geographic Names based on the AllStates zip file. Additionally, you can use Feature Class Definitions and State Abbreviations to supplement the dataset.

For an overview of shelves that specify this visual, see *Shelves for interactive visuals*.

Procedure

1. Start a new visual based on the National Geographic Features dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Interactive Map.

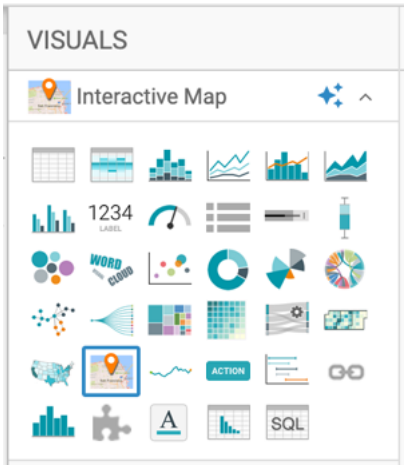

The shelves of the visual changed. They are now Geo, Measure, Colors, Tooltips, and Filters. The only mandatory shelf for map visuals is Geo.

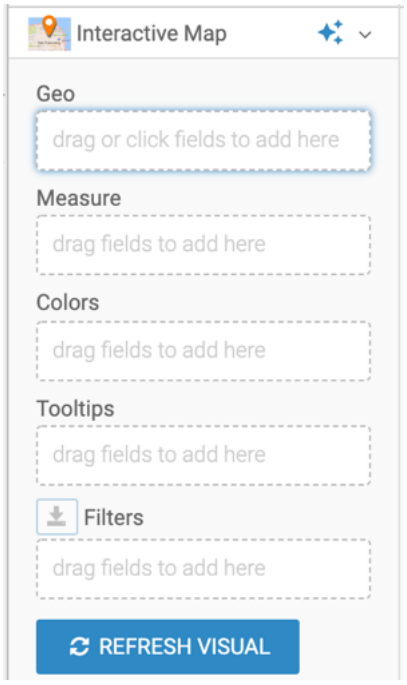

3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Under Measures, select prim-lat-dec and drag it to the Geo shelf.
   b) Cast this to the appropriate Geo Type: click the filed on the shelf, and in the FIELD PROPERTIES menu and select Change Type Latitude .

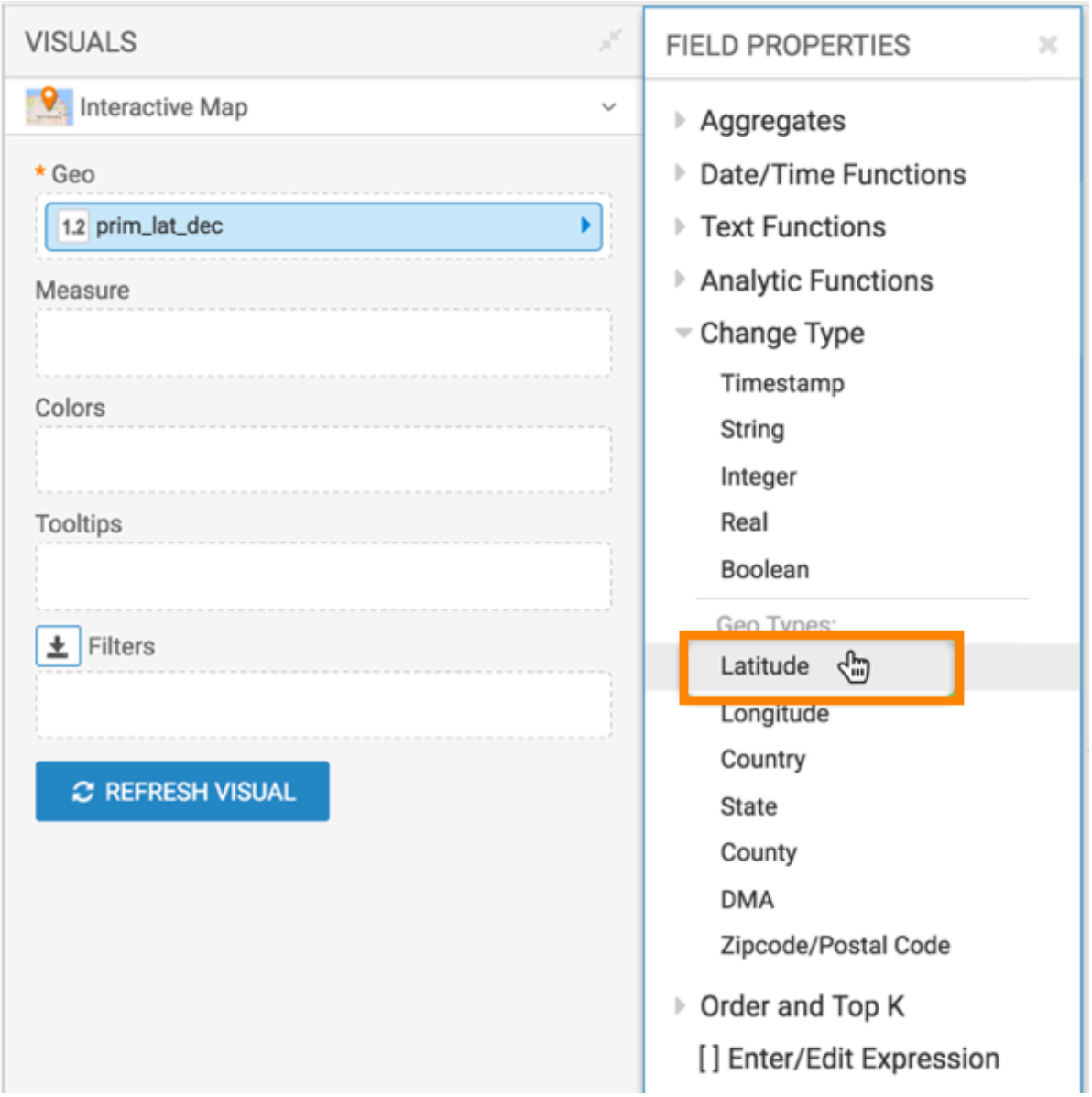

   c) Under Measures, select prim_long_dec, and drag it to the Geo shelf on the main part of the screen.
   d) Cast this to the appropriate Geo Type: click the filed on the shelf, and in the FIELD PROPERTIES menu and select Change Type Longitude .

   The two converted measurements have a (globe) icon tag. This indicates that these fields can be 'understood' as geographic fields. For more information, see *Change type* and *Geo data type*.

4. Under Dimensions, select feature_id, and drag it to the Measures shelf.

   Note: If you leave the Measures shelf empty, the system uses the value of 1, and does not require an aggregate.

5. Change the aggregation from sum(feature Id) to count(Feature Id), by selecting Aggregates Count.

6. Add the following fields to the Filters shelf:
   - Field feature_class, set to Woods
   - Field sate_alpha, set to CA

7. Click REFRESH VISUAL.

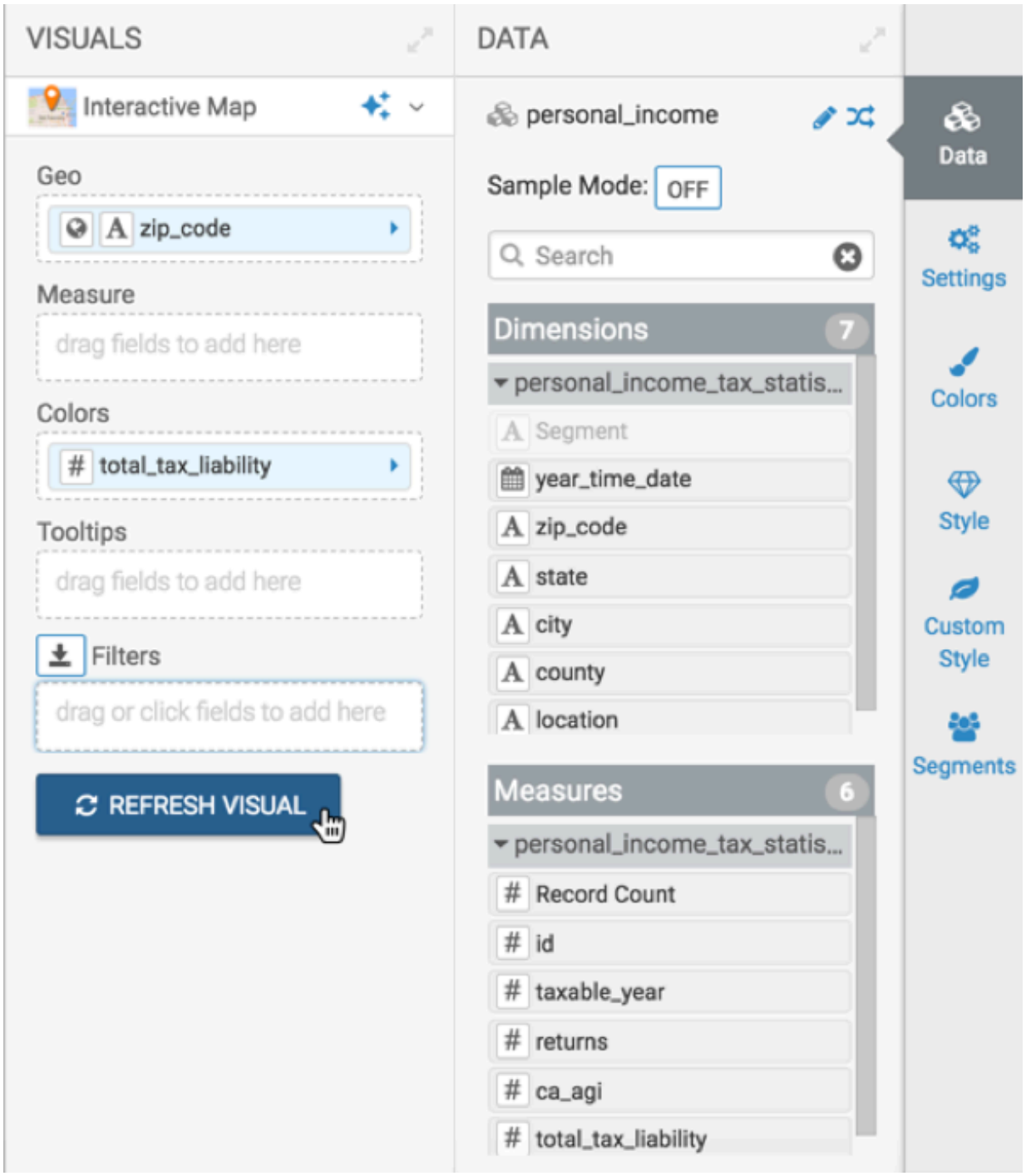

The visual appears. In this example, you can see a Google INEGI map.

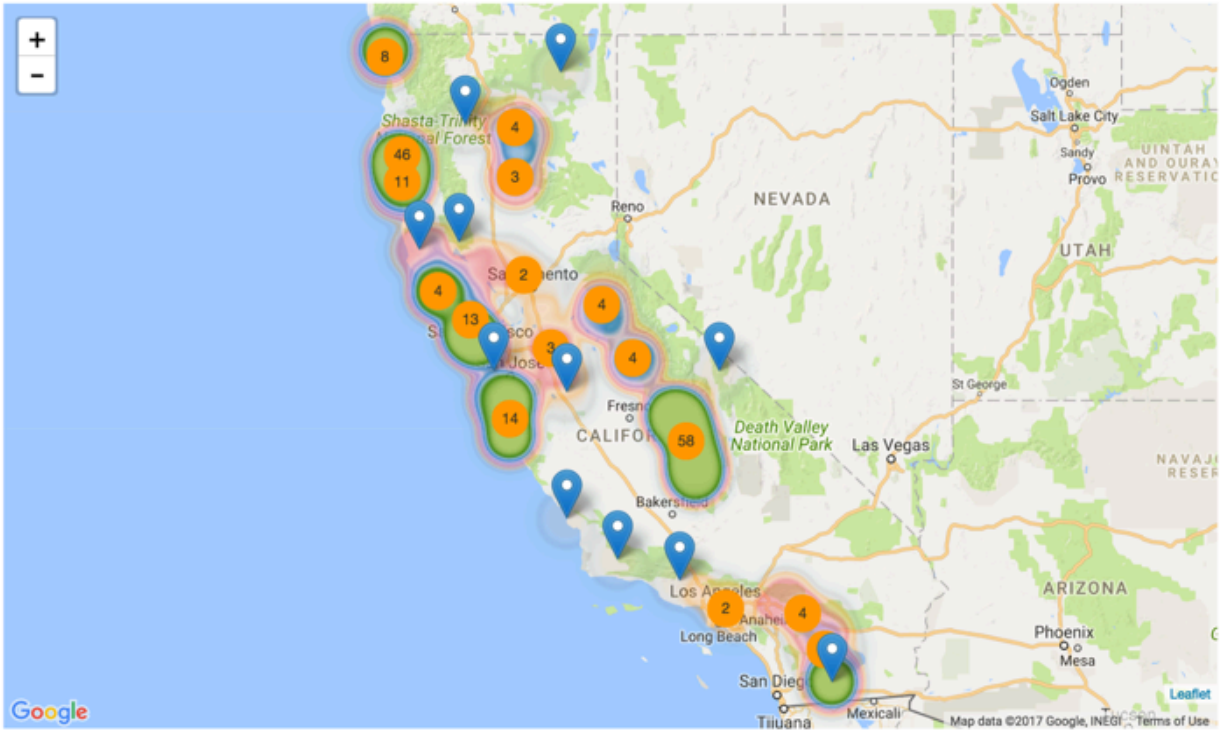

8. In this representation, there is noticeable overlap on the Heatmap. However, you can see the boundaries of each region by hovering the pointer over a section of the map.

   The total count of measurements in each region is in the circle inside the region. This comes from the Cluster. Both Heatmap and Cluster are on my default.

9. To see more detail, manipulate the map through magnification changes. You can also click on the map and manually 'pan' it to the area of interest.

   The following image shows that the regions automatically adjust into smaller sections, and you can see both greater detail and more individual 'totals'.

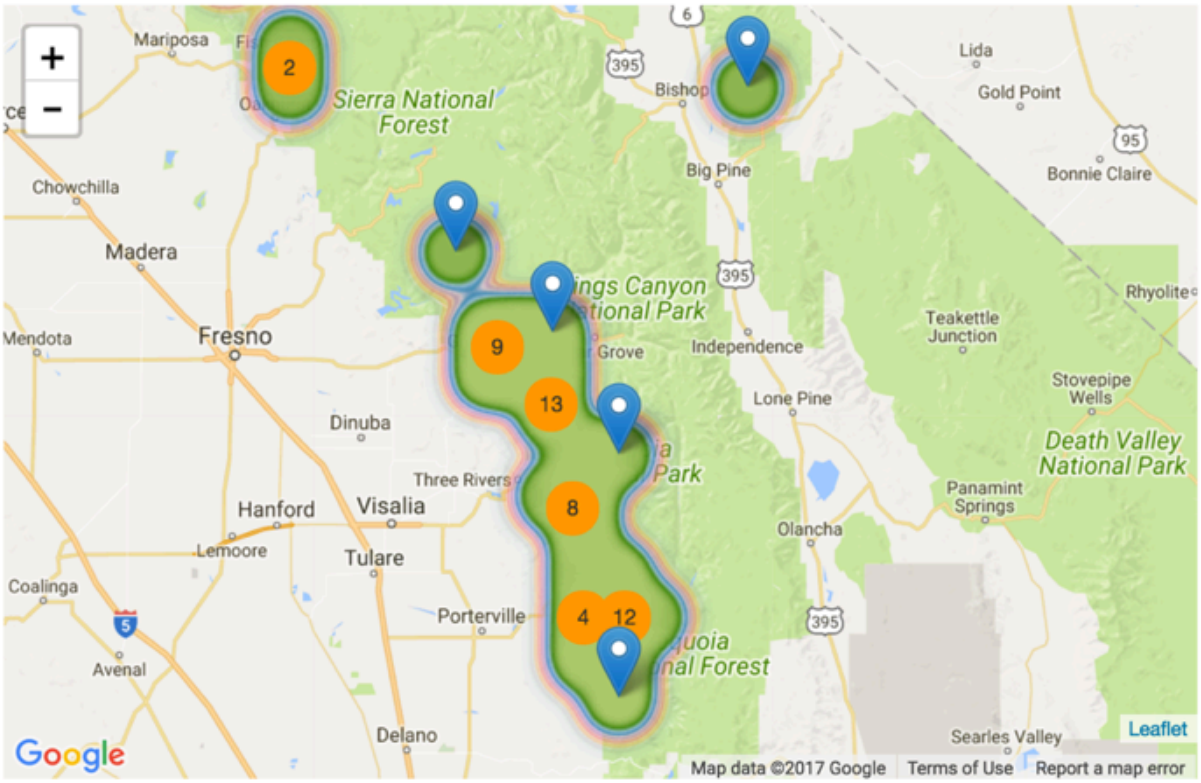

10. The 'interactive' part of this visual is that after clicking on a region to zoom-in and reposition, to the level of an individual marker, you can enable click behavior.

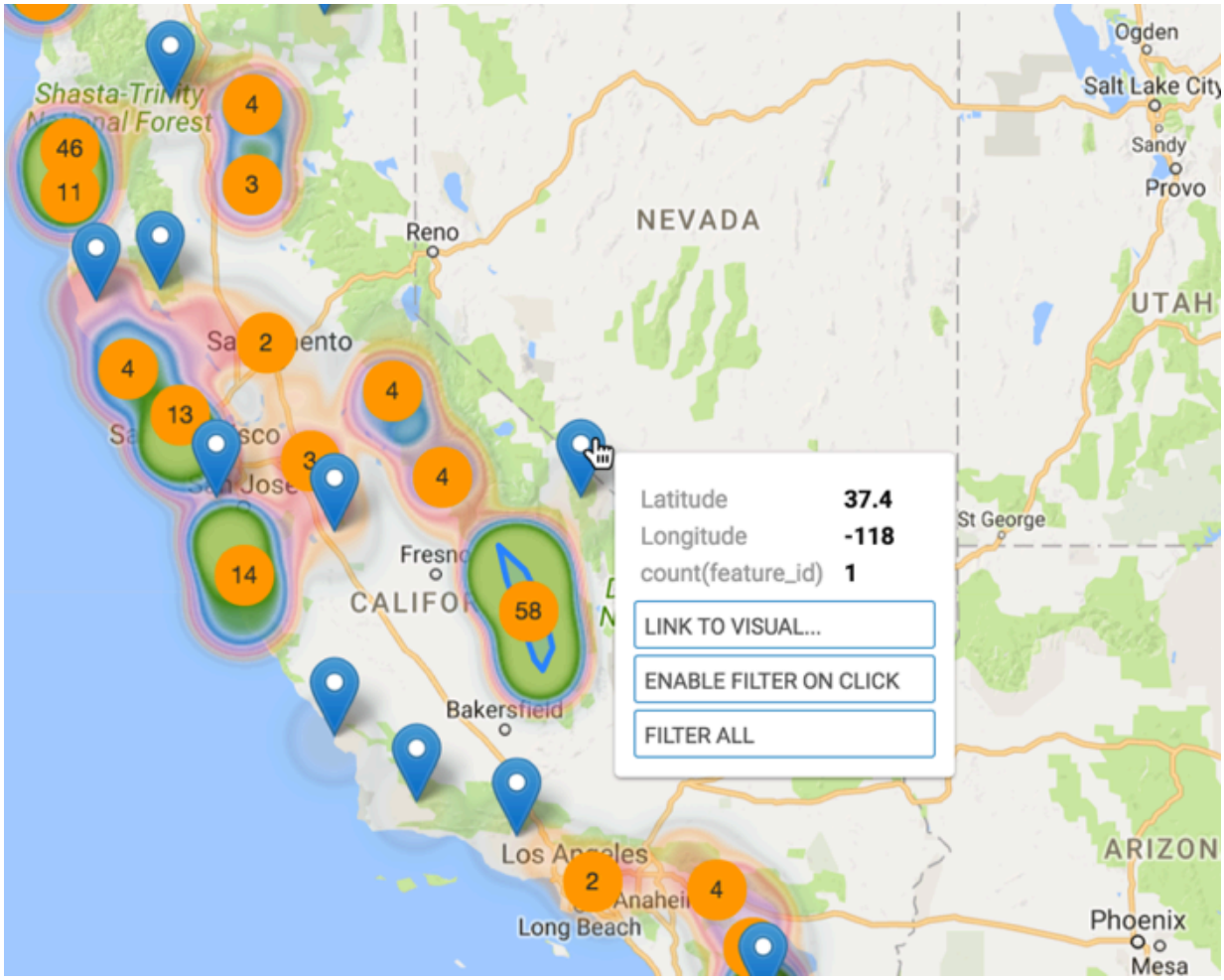

11. Change the title to California Woods.
12. At the top left corner of the Dashboard Designer, click SAVE.

What to do next

There are many interesting visualization options. To change the map server, or the base map from Google to Mapbox, see *Changing the map server to interactive maps*. To configure the various options, see these topics:

- *Map server*
- *Map style*
- *Choropleth*
- *Heatmap*
- *Cluster*
- *Circles*
- *Routes and pins*

Choropleth interactive maps

In an Interactive Map visual, Cloudera Data Visualization enables you to create choropleth maps. A choropleth map shows geographical areas that are shaded in proportion to the value of the measurement that is displayed on the map.

About this task

Tip: Consider the following when building choropleth maps:

- Use choropleth maps when your data is attached to enumeration units, such as counties, countries or zip codes.
- Standardize data to show rates and ratios instead of raw data counts. For example, instead of plotting gross GDP, plot GDP per capita.
- Use choropleth maps when you have data with continuous statistical surface. For example, Population is a count and representing it in colored map may be misleading. Population density is a ratio with a continuous statistical surface (even where 0 represents unpopulated areas); so it can be used in choropleth maps.

In this example, you will create a choropleth map that colors the average tax liability in California by zip code. The choropleth Interactive Map visual is based on the Personal Income Tax Statistics by Zip Code dataset. For an overview of shelves that specify this visual, see *Shelves for interactive maps*.

Before you begin

To create a choropleth map, you must specify the following for each visual:

- Custom mapbox style, which has a choropleth shape file in it. For more information, see *Adding a custom mapbox style*.
- Layer ID, which is the name of a unique layer within the Custom Mapbox style that identifies the shapes that you would like to color by a measure in your data. For more information, see *Specifying a choropleth layeer ID*.
- Geo-type as Shape ID: The dataset must have at least one field in the Geo shelf to determine the location on the map. You must change the Geo-type to Shape ID. A Shape ID is a key that joins the shapes in the Custom mapbox style and the field in the Cloudera Data Visualization dataset. This helps in identifying the measure that maps to the shape. This Geo field must be in the dataset that was used to create the custom mapbox style.

Procedure

1. Start a new visual based on the National Geographic Features dataset.

   For instructions, see *Creating a visual*.

2. In the VISUALS menu, find and click Interactive Map.

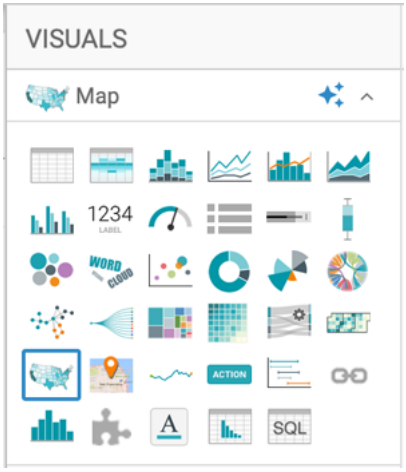

The shelves of the visual changed. They are now Geo, Measure, Colors, Tooltips, and Filters. The only mandatory shelf for map visuals is Geo.

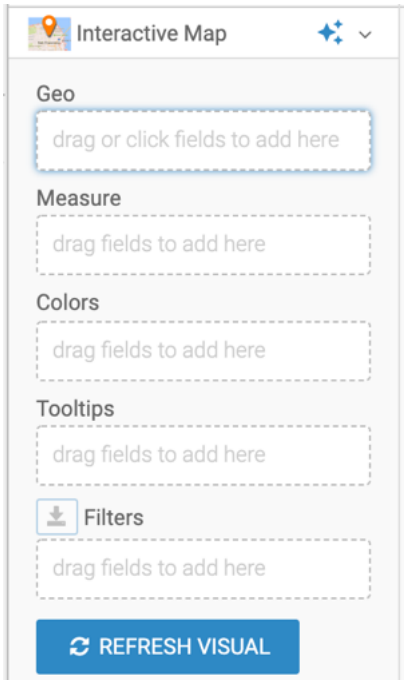

3. Populate the shelves from the available fields in Dimensions and Measures in the DATA menu.

Note: To create a choropleth map, the dataset must have at least one field in the Geo shelf with Geo-type Shape ID.

a) Select the zip_code field and drag it to the Geo shelf to identify the shapes by zip code.
b) Cast this to the appropriate Geo Type: click the field on the shelf, and in the Field Properties menu, select Change Type, and select Shape ID. This helps in identifying the measure that maps to the shape.

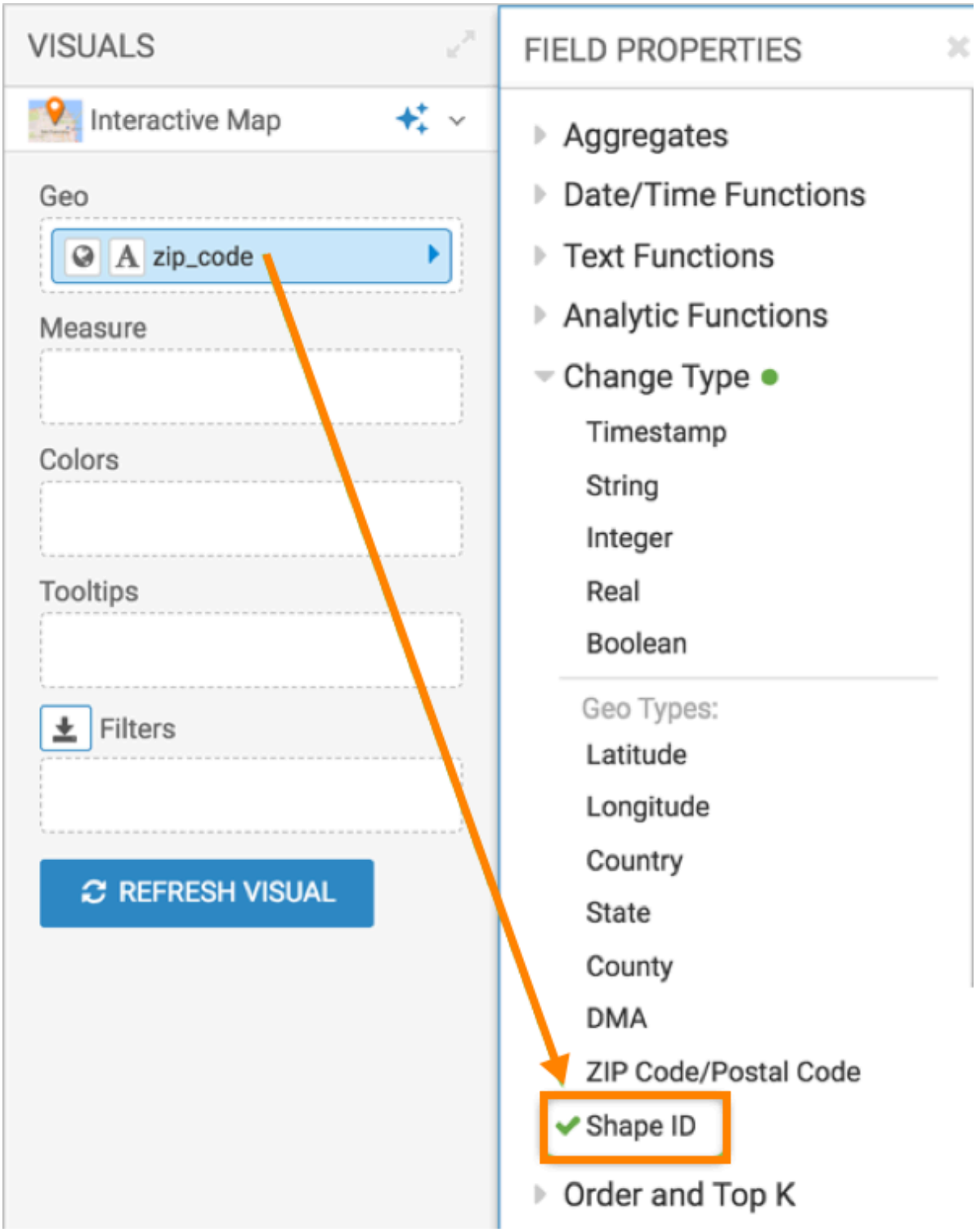

c) Under Measures, select the avg(total_tax_liability) field, and drag it to the Colors shelf.

d) Change the aggregation from total_tax_liability, by selecting Aggregates Average

This colors the average tax liability in California by zip code.

4. Navigate to the Map Style menu under Settings, and enter a Custom mapbox style that you have obtained from Mapbox. Make sure that this style has a choropleth shape file in it.

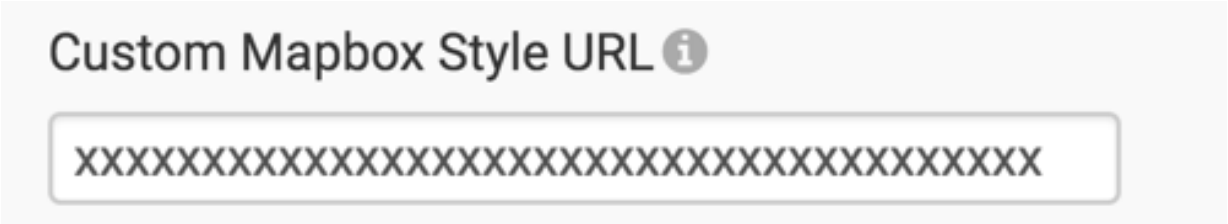

5. Navigate to the Choropleth menu under Settings, and enter a Layer ID.

Choropleth Layer ID identifies the shapes that you would like to color by a measure in your data.

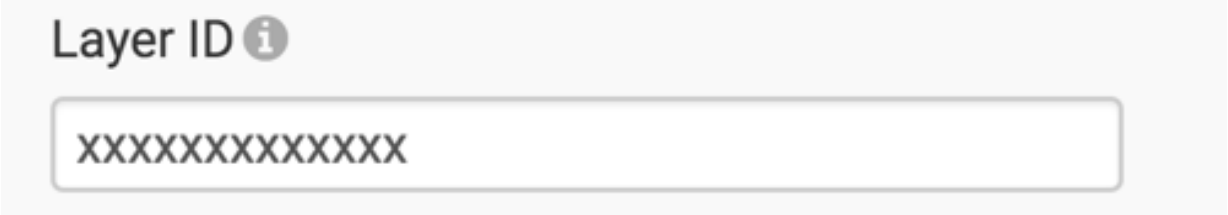

6. Click REFRESH VISUAL.

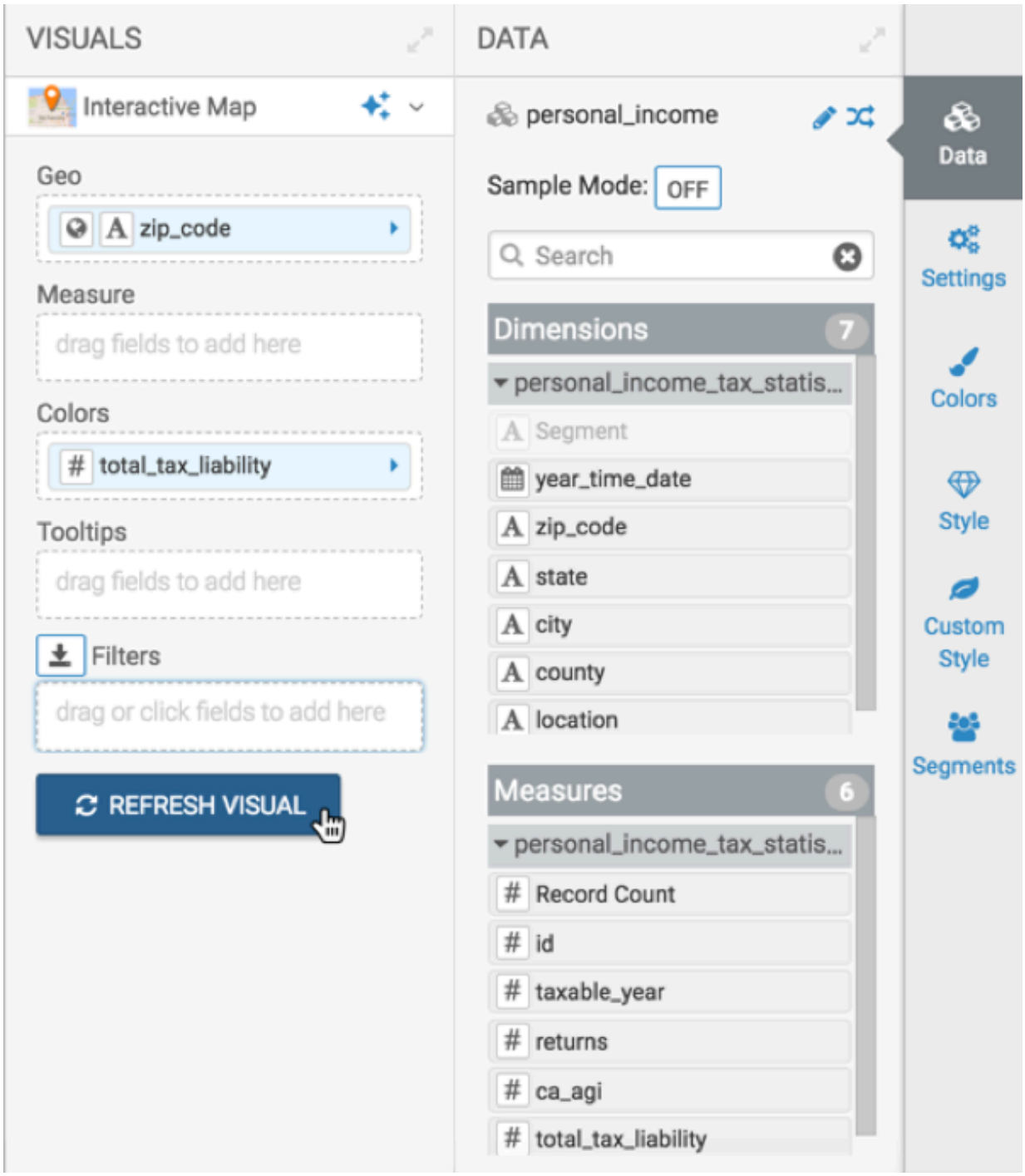

The choropleth map visual appears.

7. If the visual does not display a choropleth map, check that the Geo-type of the zip_code field is set to Shape ID.

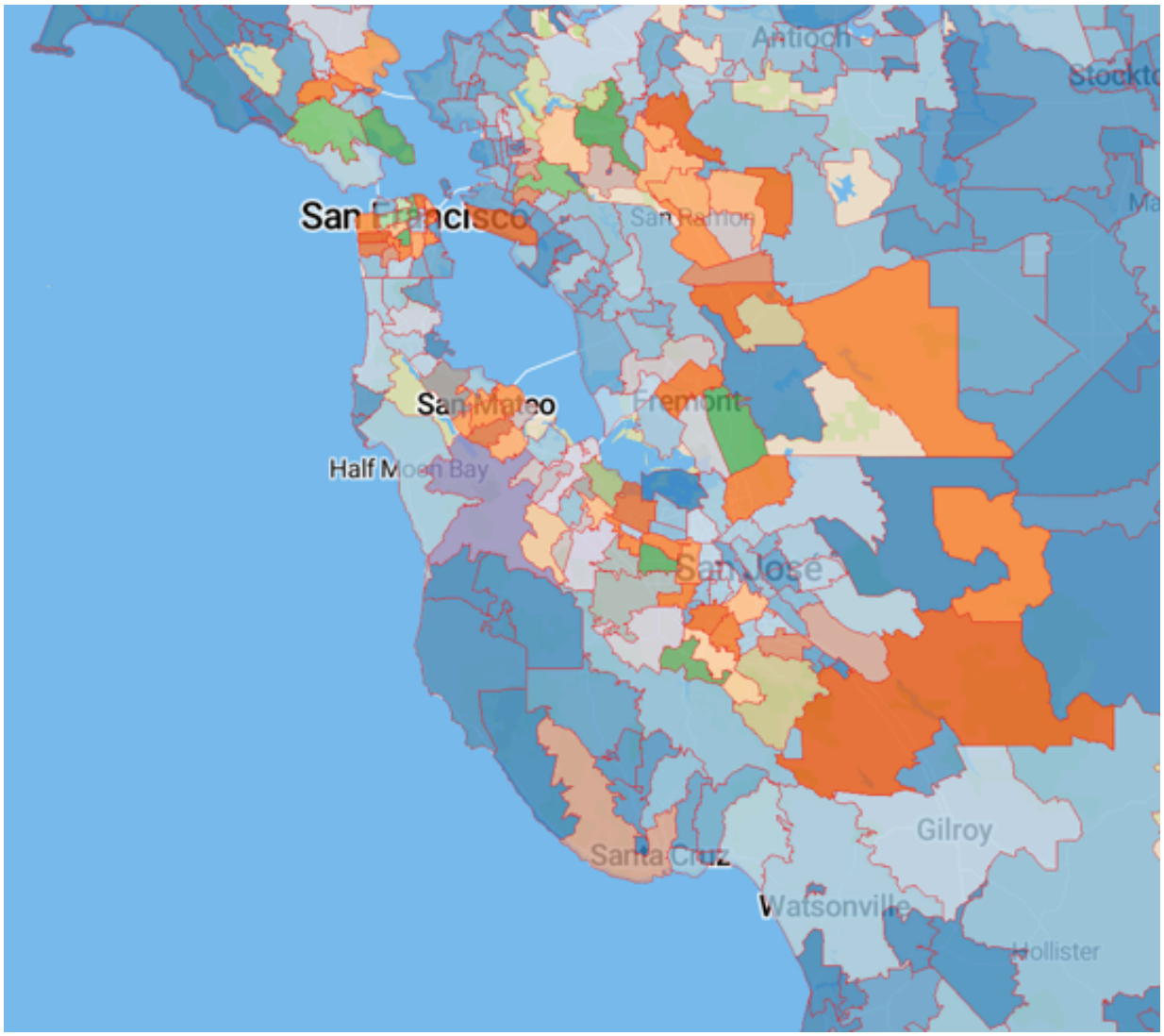

What to do next

Proceed to *Customizing* the choropleth map.

Interactive map with multiple dimensions

Cloudera Data Visualization enables you to display multiple dimensions in Interactive Map visuals.

About this task

In Interactive Map visuals you can display two distinct dimensions in the Circle Layer by placing the secondary dimension on the Colors shelf. The following steps extend the work completed in *Interactive maps*, and demonstrate how to visualize two separate measures on the same map.

Note: To use this feature, an Admin user must specify, through the Site Settings interface, the API keys granted to your organization. This is described in *Enabling API keys for interactive maps*.

Procedure

1. Clone the visual California Woods, previously built in *Interactive maps*.

   For instructions on how to clone visuals, see *Cloning a visual*.
2. Select another dimension to add to the visual, and add it to the Colors shelf.

   In this example, feature_class is selected, and the Alias Features is applied.
3. Change the definition of the feature_class filter (already on the Filters shelf) to add the Reserve and Woods values.
4. In the Settings menu, turn off the Heatmap and the Cluster.
5. In the Settings menu, enable the Circles, and turn on both the Area legend and the Color legend.
6. Click REFRESH VISUAL to see two measures on the same map.

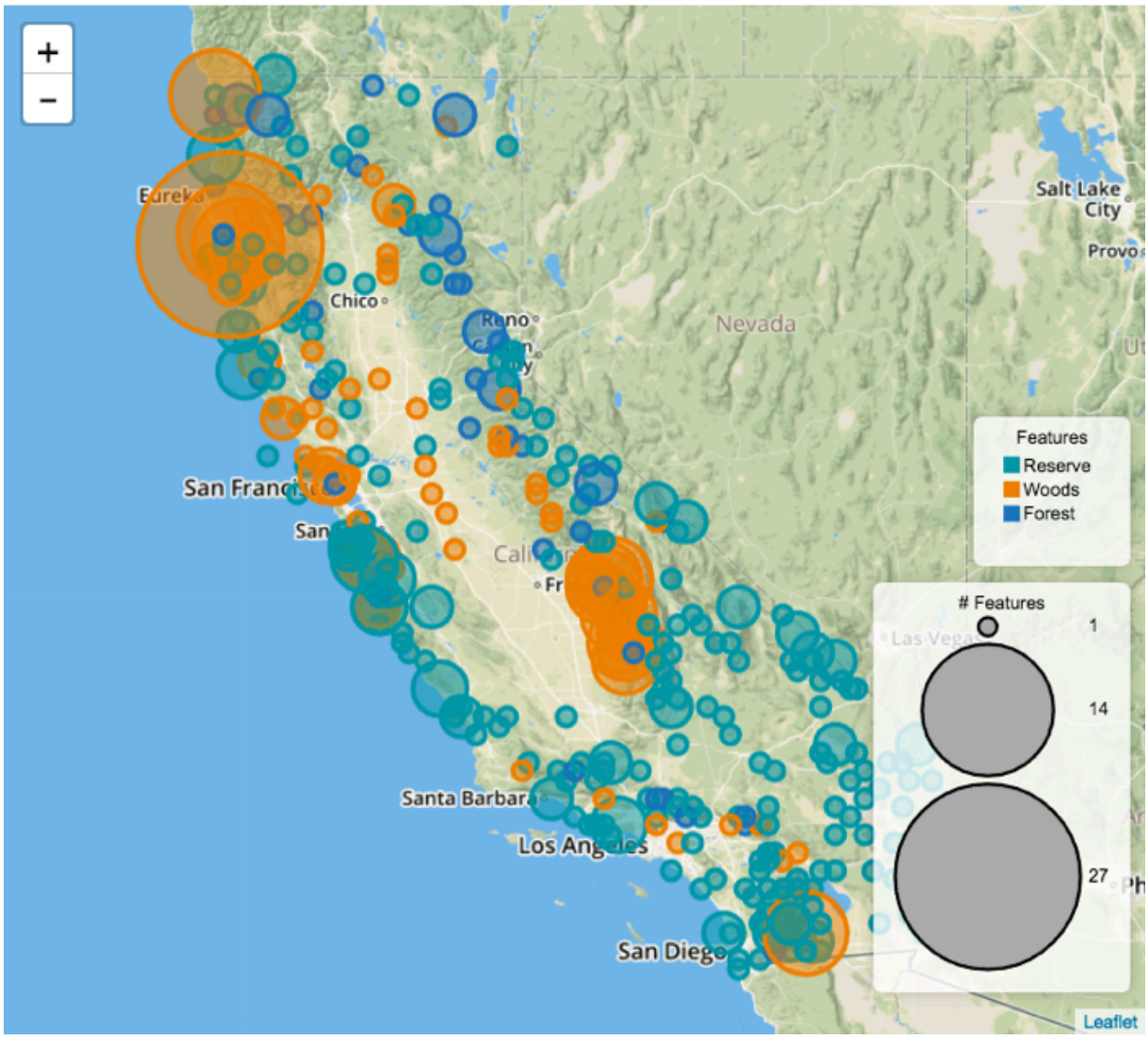

7. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'California Woods - Mapbox'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
8. At the top left corner of the Dashboard Designer, click SAVE.

Interactive map with multiple measures

Cloudera Data Visualization enables you to display multiple measures in Interactive Map visuals.

About this task

You can display two distinct measurements in the Circle Layer by placing the secondary measurement on the Colors shelf. The following steps extend the work completed in *Interactive maps*, and demonstrate how to visualize two separate measures on the same map.

Note: To use this feature, an Admin user must specify, through the Site Settings interface, the API keys granted to your organization. This is described in *Enabling API keys for interactive maps*.

Procedure

1. Clone the California Woods visual, previously built in *Interactive maps*.

   For instructions, see *Cloning visuals*.
2. Select another measurement to add to the visual, and add it to the Colors shelf.

   In this example, elev_in_m is selected, the aggregation function is changed to max(elev_in_m), and the Elevation alias has been applied
3. In the Settings menu, turn off Heatmap, Cluster, and Routes.
4. In the Settings menu, enable Circles, and turn on both the Area legend and the Color legend options.
5. Click REFRESH VISUAL to see two measures on the same map.

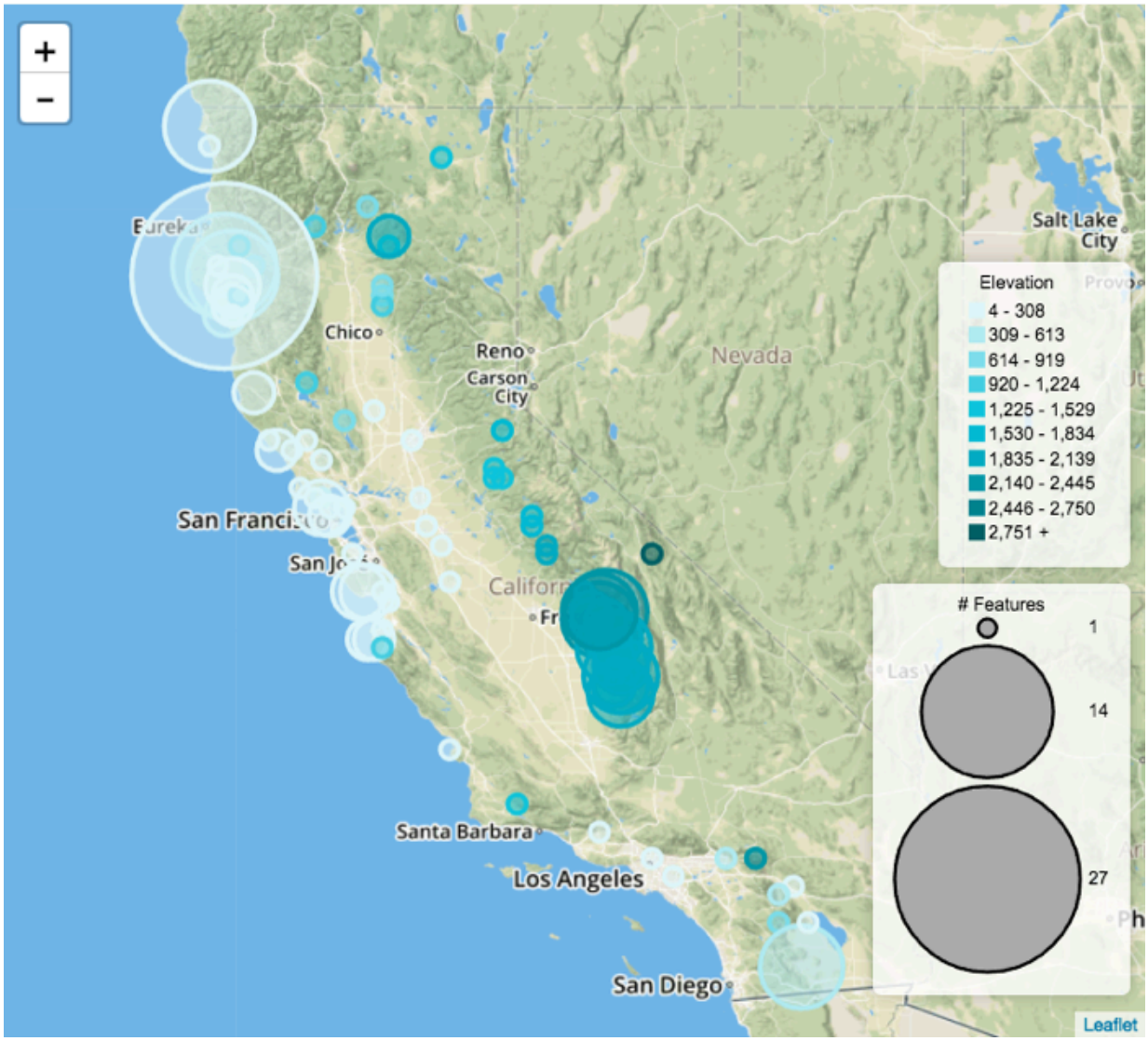

6. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'California Woods - Two Dimensions'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
7. At the top left corner of the Dashboard Designer, click SAVE.

Changing the map server for interactive maps

Cloudera Data Visualization uses two third-party options for its Interactive Map visuals: Google and Mapbox.

About this task

The Map Server setting is used to specify the mapping choice.

Note: To use this feature, an Admin user must specify, through the Site Settings interface, the API keys granted to your organization. This is described in *Enabling API keys for interactive maps*.

The following steps demonstrate how to switch between Google and Mapbox interactive map representation.

Procedure

1. Clone the visual previously built in *Interactive maps*.

   For instructions, see *Cloning visuals*.
2. Follow the instructions in *Changing the map server* to switch between Google and Mapbox map styles.
3. Click REFRESH VISUAL to see the map rendered with Mapbox option.

   See the *Mapbox* documentation for more details.

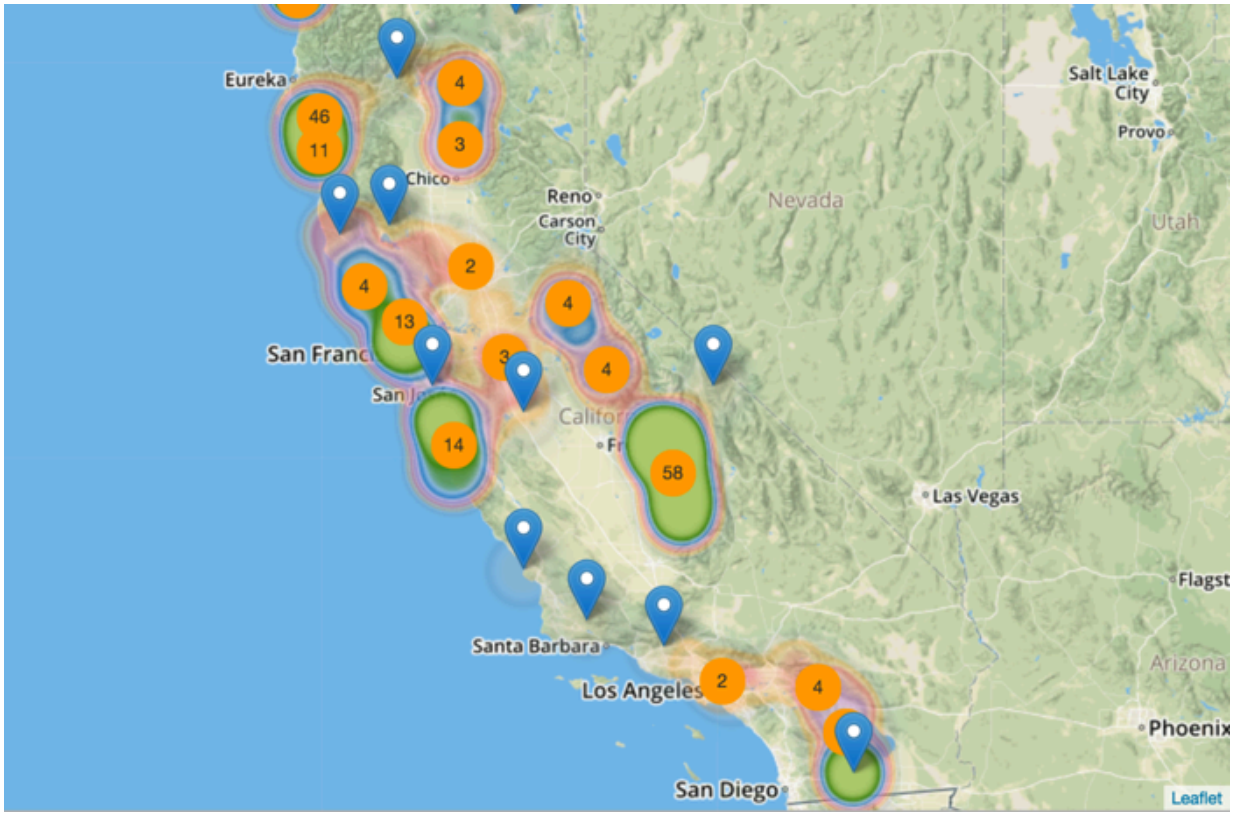

4. Compare with the similar map rendered on a Google INEGI map.

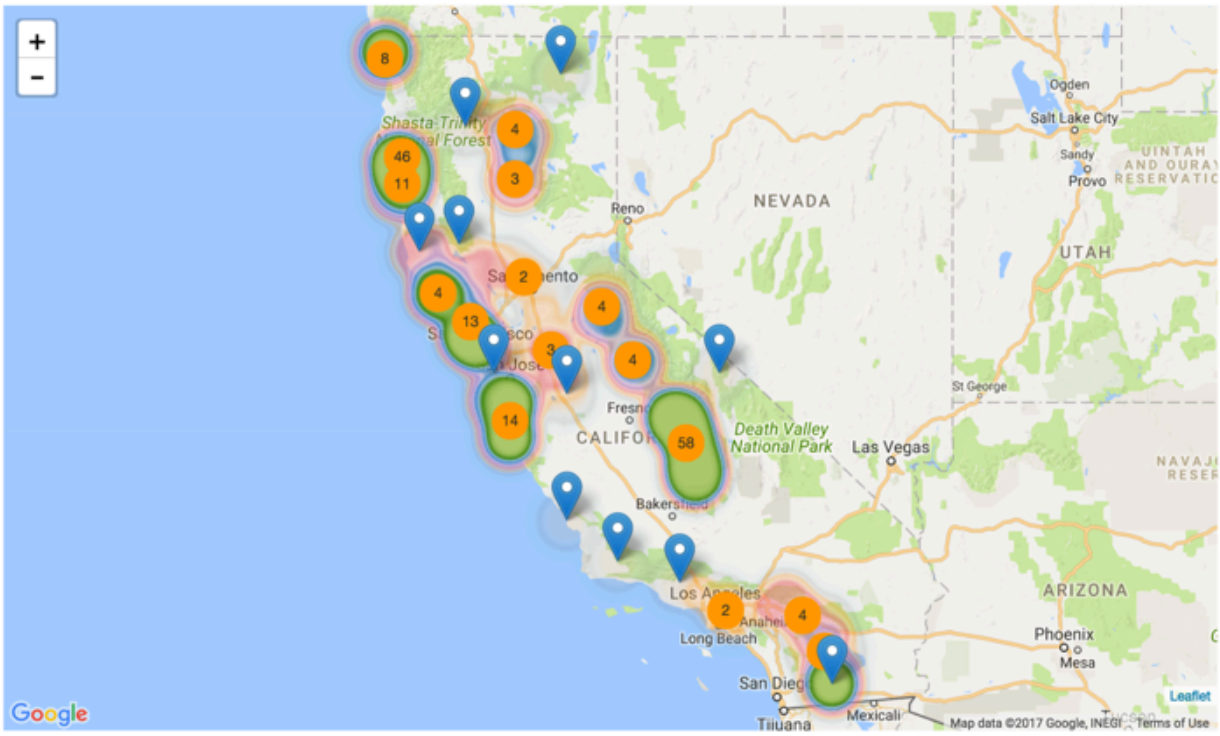

5. Click the pencil/edit icon next to the title of the visualization to enter a name for the visual.

   In this example, the title is changed to 'California Woods - Mapbox'. You can also add a brief description of the visual as a subtitle below the title of the visualization.
6. At the top left corner of the Dashboard Designer, click SAVE.

Changing layer options for interactive maps

Cloudera Data Visualization enables you to make adjustments to all layers of an Interactive Map visual.

About this task

By default, interactive maps render with a simple heatmap and a cluster. However, Cloudera Data Visualization has several adjustments to all the layering options: Heatmap, Cluster, and Circles.

Note: To use this feature, an Admin user must specify, through the Site Settings interface, the API keys granted to your organization. This is described in *Enabling API keys for interactive maps*.

The following steps demonstrate how to make changes to layer specifications in interactive maps.

Procedure

1. Clone the visual previously built in *Interactive maps*.

   For instructions, see *Cloning visuals*.
2. In the Settings interface, select the appropriate option for making customizations to interactive visuals:
   - Heatmap
   - Cluster
   - Circles
3. Make the necessary changes, and click REFRESH VISUAL.
4. Change the title of the visual, and SAVE it.

Plotting routes on interactive maps

Cloudera Data Visualization enables you to plot routes over an Interactive Map.

About this task

You can plot routes over a map by specifying distinct IDs on the Color shelf, a time series on the Tooltips shelf, and enabling routes as described in *Enabling routes*.

The following steps demonstrate how to plot routes on an interactive map. In the following example, truck movement data is used.

Procedure

1. Select the appropriate dataset, and create a new dashboard.
2. In the dashboard, open the default visual and change the visual type to Interactive Map.
3. Enable the Routes, as described in *Enabling routes*
   a) Click the Settings menu.
   b) Select Routes/Pins Enable Routes .
4. Disable heatmap and cluster. It is recommended that you leave the circles in place, so it shows the individual elements of the route. If you choose to hide the circles, the graph only shows the route connectors.
5. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Geo Shelf - Under Measures, select latitude and longitude, and add them to the Geo shelf.

      If your longitude and latitude measurements are not automatically recognized (appear with a (globe) icon on the Geo shelf), cast them to the appropriate Geo Type.

      1. Click the field on the shelf to open the FIELD PROPERTIES menu.
      2. Select Change Type and choose Latitude or Longitude as appropriate.

      For more information, see *Change type* and *Geo data type*.
   b) Colors Shelf - Under Measures, select truckid, and place it on the Colors shelf.

      You may need to remove the aggregation from the field on the shelf.
   c) Tooltips Shelf - Under Dimensions, find the eventtime field, and add it to the shelf.

      1. Click the field on the shelf to open the FIELD PROPERTIES menu.
      2. Click Order and Top K.
      3. Choose the Ascending order.
   d) Filters Shelf - [Optional] To restrict the number of routes, you can choose the filter on the routename field. To restrict the number of trucks displayed, you may choose to restrict on truckid field, and so on.
   e) Measure Shelf - [Optional] You can display a measure here.
6. Change the color palette to ensure that the routes stand out sufficiently over the regular features of the map: click Colors and select an appropriate palette.

7. Click REFRESH VISUAL.

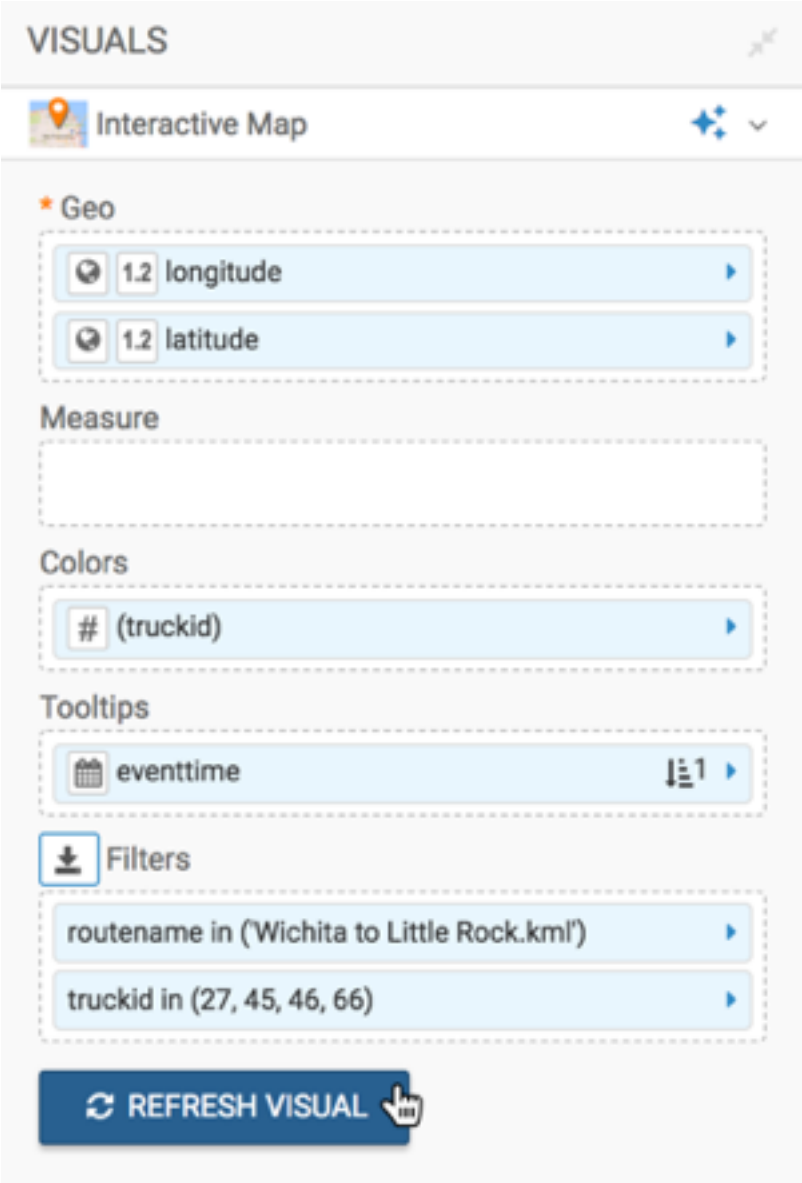

The visual is displayed.

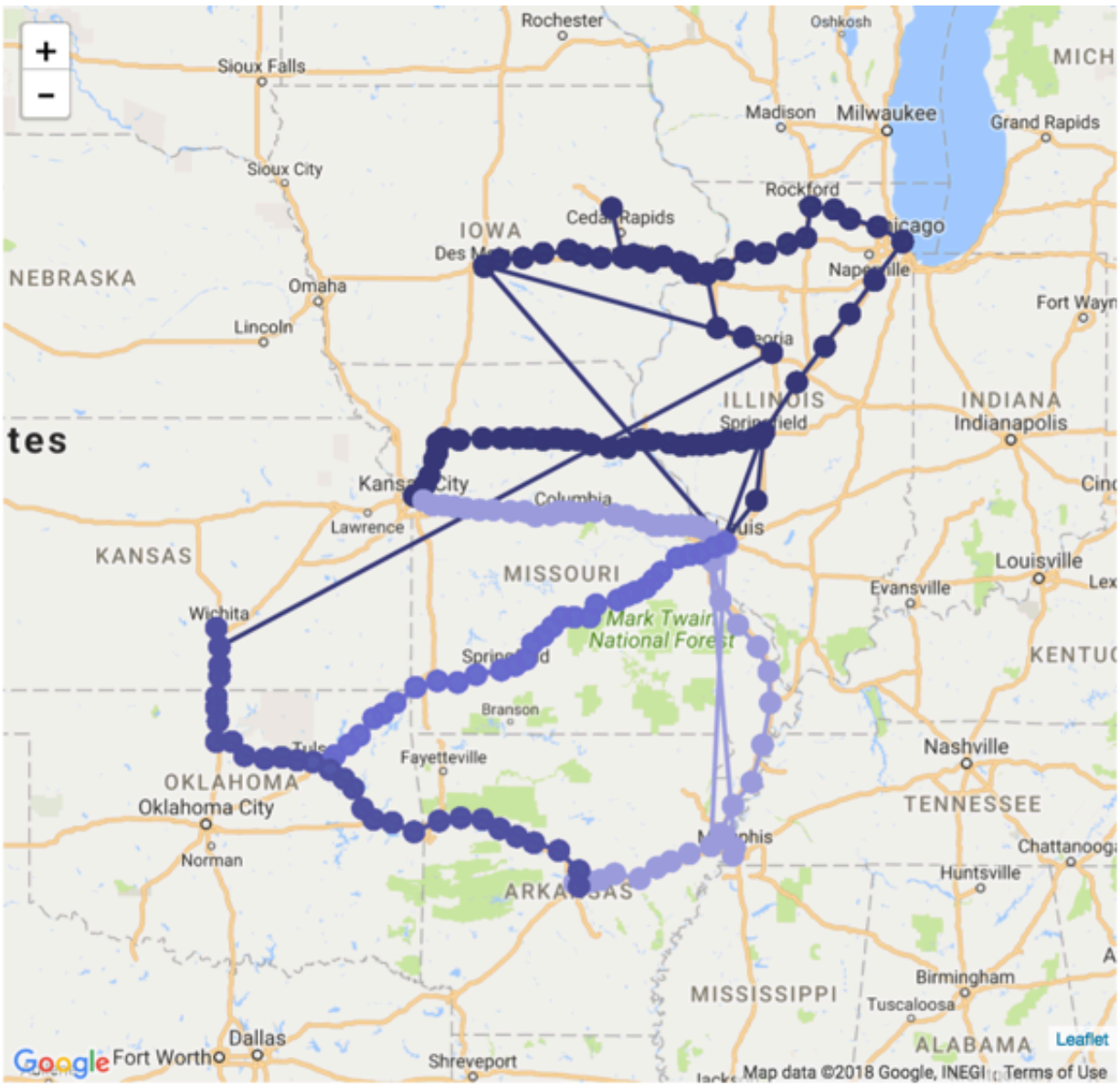

8. Change the title of the visual, and SAVE it.

Using alphabetic values in interactive maps

Cloudera Data Visualization Interactive Map visuals support both numeric and non-numeric values on the Color shelf. As a result, aspects of named categories can be viewed simultaneously on the same visual as a distinct series.

About this task

In the following example, truck movement data is used to demonstrate how to plot incident counts for several drivers, and differentiate them by the color.

Procedure

1. Select the appropriate dataset, and create a new dashboard.
2. In the dashboard, open the default visual and change the visual type to Interactive Map.

3. Enable the Circles, as described in *Displaying circles*.
   a) Click the Settings menu.
   b) Select Circles Enable Circles .
4. Enable both the area and color legends of the Circles option by selecting the Add Circle Area Legend and Add Circle Color Legend options.
5. Disable all the other layers: heatmap, cluster, marker, and route.
6. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Geo Shelf - Under Measures, select latitude and longitude, and add them to the Geo shelf.

      If your longitude and latitude measurements are not automatically recognized (appear with a (globe) icon on the Geo shelf), cast them to the appropriate Geo Type.

      1. Click the field on the shelf to open the FIELD PROPERTIES menu.
      2. Select Change Type and choose Latitude or Longitude as appropriate.

      For more information, see *Change type* and *Geo data type*.
   b) Measures Shelf - Under Measures, select eventtype and place it on the measures shelf.

      1. Change the aggregation function to count().
      2. Alias the field as # Incidents.
   c) Colors Shelf - Under Measures, select drivername, and place it on the Colors shelf.

      You may need to remove the default field aggregation from the field on the shelf. Alias the field as Driver.
   d) Filters Shelf - Add the field eventtype to this shelf, and select the single value 'Overspeed'.

      1. Add the field drivername to the filters shelf.
      2. Select a small number of drivers to see details.
7. Change the color palette to ensure that the routes stand out sufficiently over the regular features of the map: Click Colors, and select an appropriate palette.

8. Click REFRESH VISUAL.

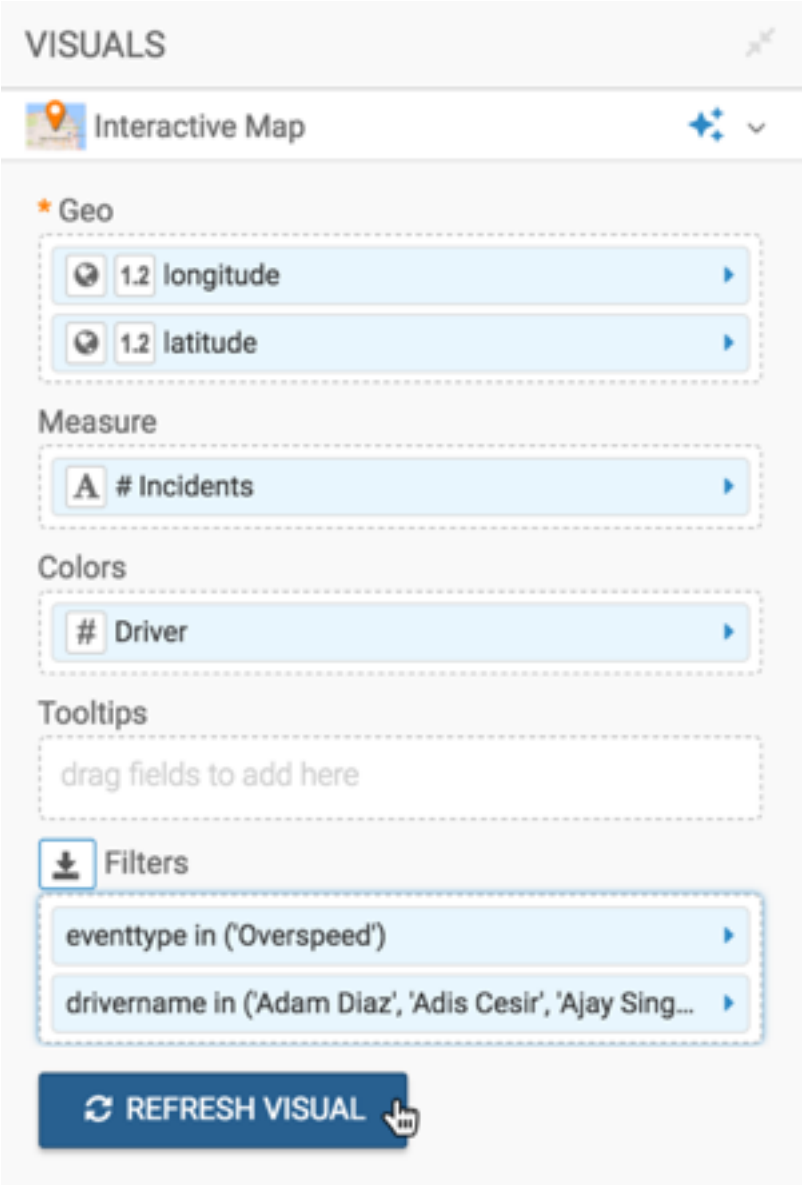

The visual is displayed. The colors are defined by the names of the drivers, which are non-numeric.

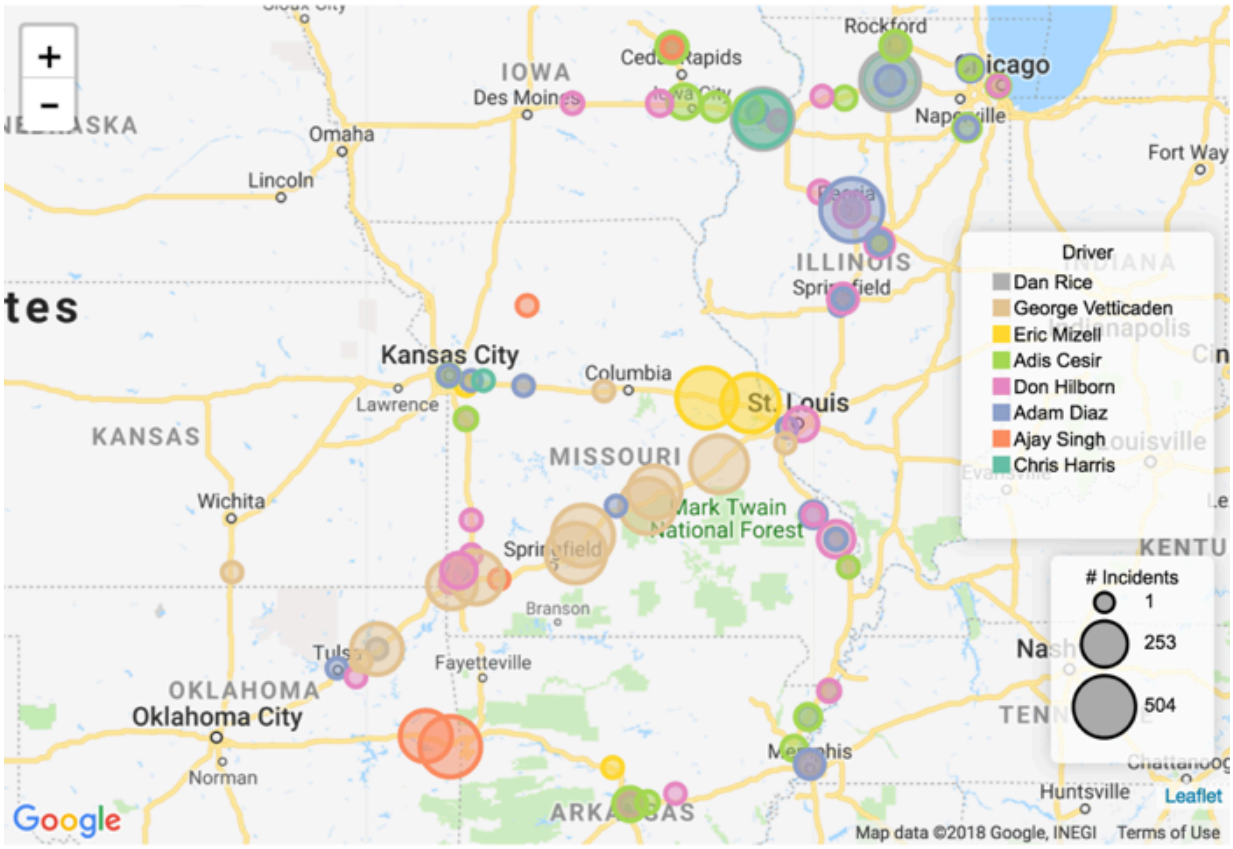

9. Change the title of the visual, and SAVE it.

Segmenting data qualitatively in interactive maps

Cloudera Data Visualization enables you to plot qualitative ranges of data by specifying the data segmentation on the Color shelf of an Interactive Map visual.

About this task

The following steps demonstrate how to plot incident counts on an Interactive Map. Truck movement data is used in the example.

Procedure

1. Select the appropriate dataset, and create a new dashboard.
2. In the dashboard, open the default visual and change the visual type to Interactive Map.
3. Enable Circles, as described in *Displaying circles*.
   a) Click the Settings menu.
   b) Select Circles Enable Circles .
4. Enable both the area and color legends of Circles by selecting the Add Circles Area Legend and Add Circles Color Legend options.
5. Disable all the other layers: heatmap, cluster, marker, and route.

6. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.
   a) Geo Shelf - Under Measures, select latitude and longitude, and add them to the Geo shelf.

      If your longitude and latitude measurements are not automatically recognized (appear with a (globe) icon on the Geo shelf), cast them to the appropriate Geo Type.

      1. Click the field on the shelf to open the FIELD PROPERTIES menu.
      2. Select Change Type and choose Latitude or Longitude as appropriate.

      For more information, see *Change type* and *Geo data type*.
   b) Measures Shelf - Under Measures, select eventtype and place it on the measures shelf.

      1. Change the aggregation function to count().
      2. Alias the field as # Incidents.
   c) Colors Shelf - Under Measures, select eventtype and place it on the Colors shelf.

      1. Click the field on the shelf, to open the FIELD PROPERTIES menu.
      2. Select the [] Enter/Edit Expression option.
      3. Change the field expression to split the data into two qualitative ranges, High and Low:

         ```
         case when count([eventtype]) > 20 then 'High' else 'Low (20 or less)'
          end
         ```

      4. Change the field Alias to # Lane Departures.
   d) Filters Shelf - Add the field eventtype to this shelf, and select the single value 'Lane Departures'.

      [Optional] Add the field drivername to the filters shelf, and select a small number of drivers to see details.
7. Change the color palette to ensure that the colors are distinct: Click Colors, and select an appropriate palette.

   In this example, because the data is divided into two distinct groups, a two-color palette is chosen.

8. Click REFRESH VISUAL.

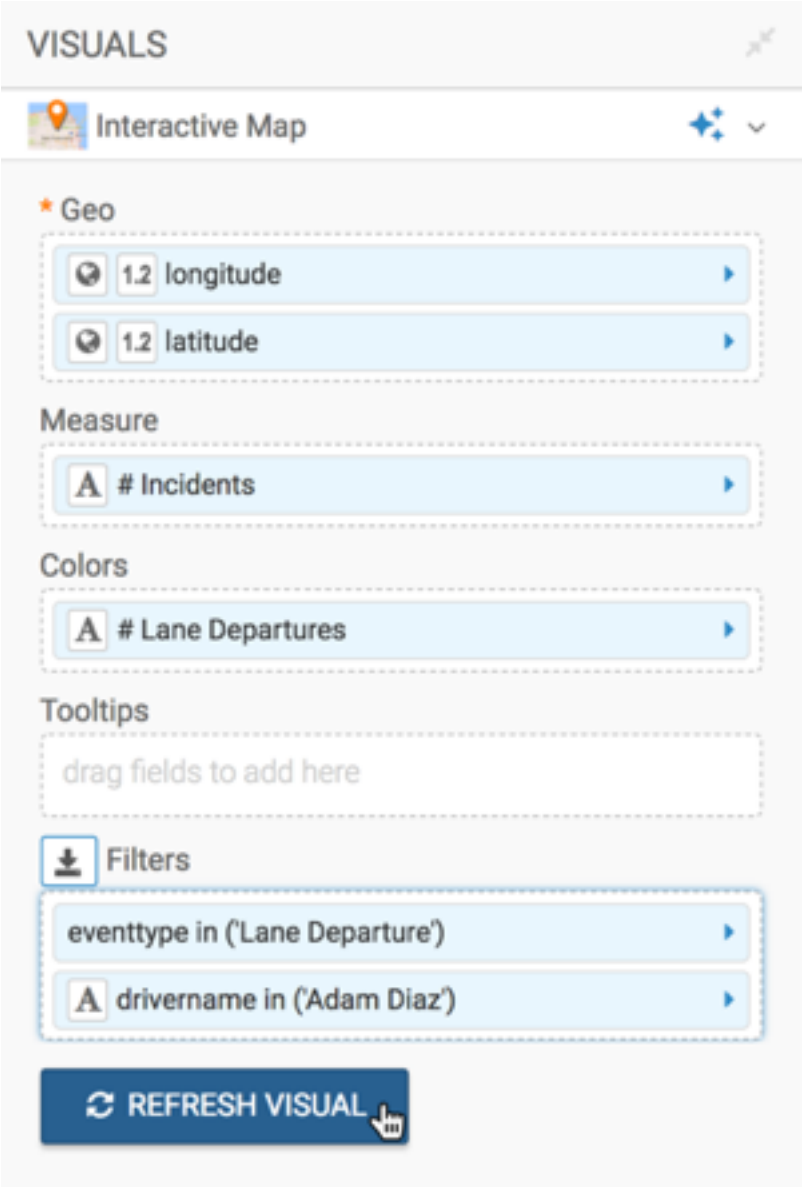

The visual is displayed.

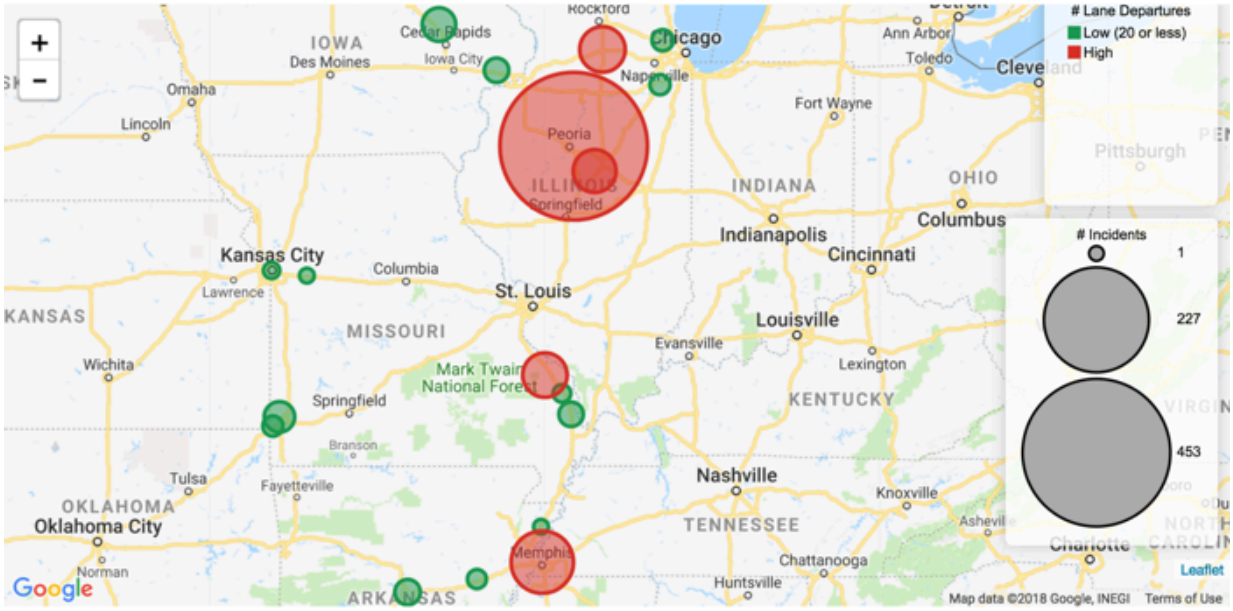

9. Change the title of the visual, and SAVE it.

Shelves for interactive maps

Overview of shelves for Cloudera Data Visualization Interactive Map visuals.

An Interactive Map visual has the following shelves:

Geo

Add a Geo-type field from the DATA menu to determine the location on the map.

This visual supports zipcode or latitude/longitude Geo types . If the field is not automatically recognized as a qualified Geo type, select Change Type from the FIELD PROPERTIES menu.

This shelf accepts multiple fields, but plots a map with the highest possible resolution.

This is a mandatory shelf.

Measure

Add a Measure field from the DATA menu to compute heatmap intensity across locations, values of cluster aggregations, and the size of circle markers.

This shelf accepts a single field, or uses the value of 1 if no measurement is specified.

Colors

Add a Dimension or Measure field from the DATA menu to specify the color of the circle markers, or distinct routes by common identifier.

This shelf accepts a single field, and uses both numeric and non-numeric values.

Tooltips

Add Measure fields from the DATA menu for additional information in the visual's Tooltip.

This shelf accepts multiple fields.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Sparklines

Cloudera Data Visualization enables you to create Sparklines visuals. A Sparklines chart shows the general shape of the variation in measurement (usually over time) in a simple, condensed, and memorable manner. Sparklines are small and succinct, and can be easily located with the text where they are discussed. You can contrast Sparklines with typical X/Y charts that are designed to show as much data as possible.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.
2. Select Sparklines in the VISUALS menu.

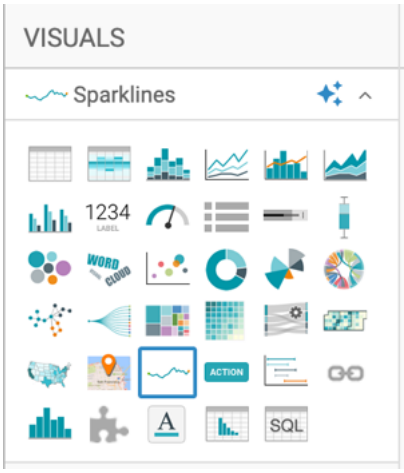

The shelves of the visual changed. They are now X Axis (mandatory), Y Axis (mandatory), and Filters.

3. Populate the shelves from the available fields (Dimensions, Measures) in the DATA menu.
   a) Under Dimensions, select year and drag it to the X Axis.
   b) Under Measures, select gdp_per_capita and drag it to the Y Axis.

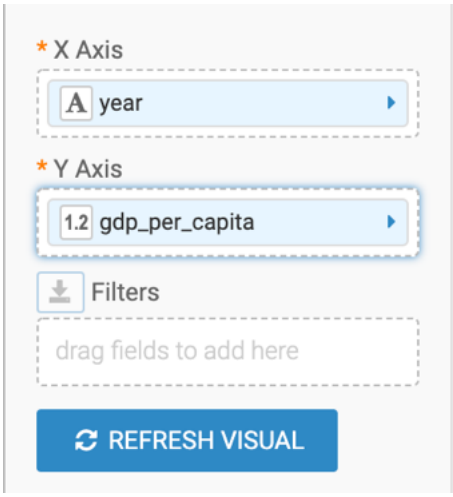

4. On the gdp_per_capita field of the Y Axis shelf, click the (arrow) icon and select Aggregates Sum .
   This is necessary because the Sparklines visual type uses aggregated measurements.

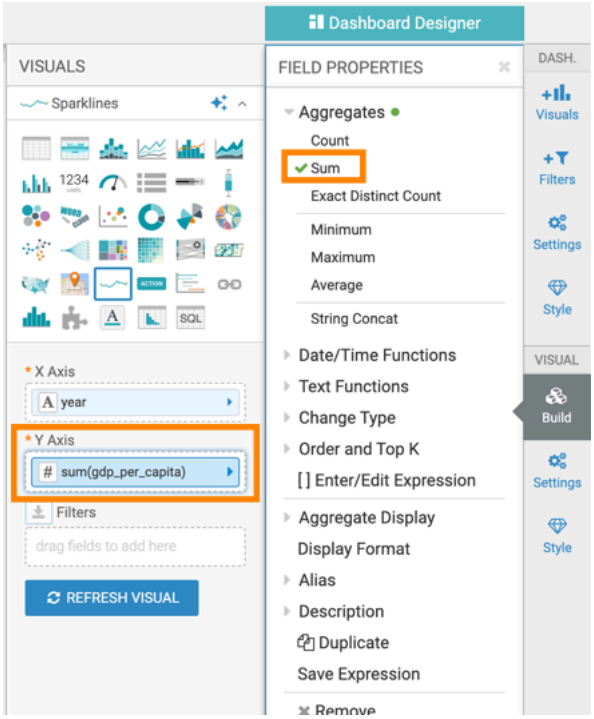

   The Y Axis shelf now contains the modified field sum(gdp_per_capita).
5. Click REFRESH VISUAL.
   The Sparklines visual graph appears.

6. On the Filters shelf, add several Dimensions and Measures from the DATA menu.
   This enables you to dynamically control the data input and to discover the data at a more granular level.

   For example, from Dimensions, drag country, un_subregion, and un_region to the Filters shelf.
7. Click the (pencil) icon next to the title of the visualization to edit it, and enter World Population - Sparklines as the new name.
8. Click SAVE at the top left corner of the Dashboard Designer.

Using sparklines

Learn about using Sparkline visuals.

Procedure

- Follow these steps to create a Sparklines chart for a specific country:
  a) On the Filters shelf, click country and click Select values.

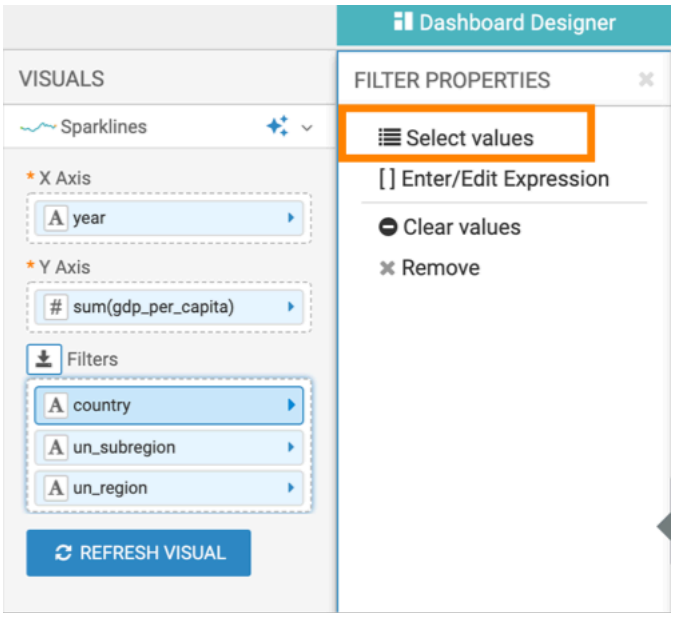

  b) In the Filter for country modal window, select Haiti and click APPLY.

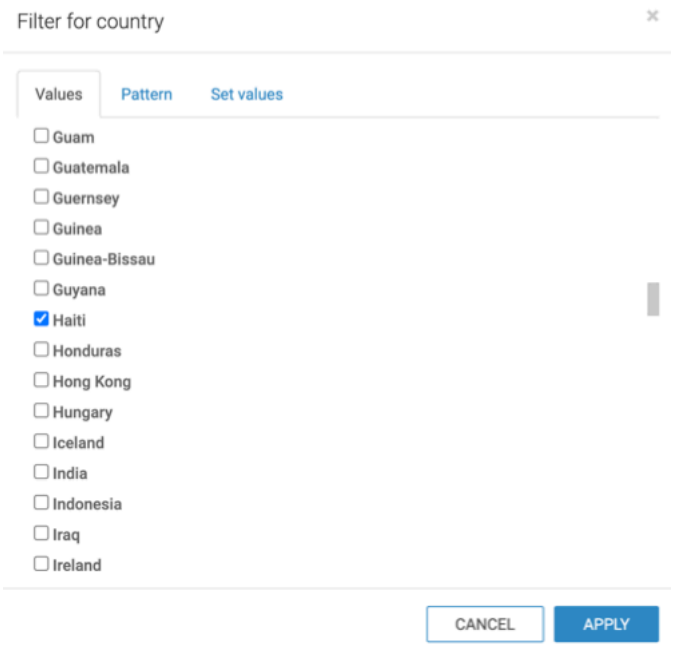

  c) Click REFRESH VISUAL.
  You can see how the line has changed compared to the previous display (world-wide).

- Follow these steps to create a Sparklines for a region, such as Europe:
  a) Clear the selection in the existing filters.
  b) From Dimensions, drag un_region to the Filters shelf.
  c) In the Filter for un_region modal window, select Europe and click APPLY.

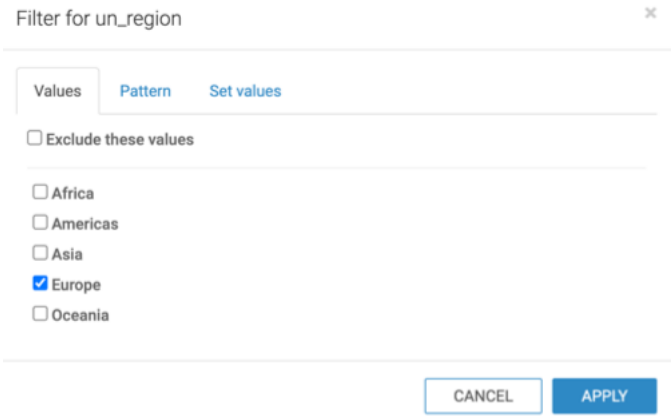

  d) Click REFRESH VISUAL.
  You can see how the line has changed.

Shelves for sparklines

Overview of shelves for Cloudera Data Visualization Sparklines visuals.

A Sparklines visual has the following shelves:

X Axis

This is conceptually the X axis. Use a Dimension for a horizontally-oriented plot or a Measure for a vertically-oriented plot.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in columns.

Y Axis

This is conceptually the Y axis. Use a Measure for a horizontally-oriented plot or a Dimension for a vertically-oriented plot.

When using multiple fields, the last field defines the axis, while the preceding fields trellis the graph in rows.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

AI visual [Technical Preview]

Cloudera Data Visualization enables you to embed interactive and actionable elements within your dashboards and applications. The AI visual introduces a natural language interface for interacting with dashboards and datasets. It leverages text input and speech detection with interactive data visualization to provide an intuitive user experience. The AI visual is a powerful tool for enhancing your dashboards. It helps you to obtain data insights through interactive conversations about the information stored in a dataset.

About this task

How does it work?

When added to a dashboard, the AI visual connects to a vector database (Solr 9 or SQLite). It queries this database based on user-entered text, providing augmented chat support. Results resembling your query are sent to the specified large language completion model defined in the site settings. The model processes the information and displays the results within the visual as a text response.

How to use it?

You can simply ask a question and the AI visual provides a textual response. After configuring and adding the visual to your dahsboard, you can interact with it by entering your question into the text box. You can also use the [microphone] icon to ask your question. The system will detect your speech, convert it to text, and enter it to the text box. The AI visual will provide a response based on the underlying data. You can ask further questions or refine your queries through voice or text, and the system will offer additional insights by updating the visualizations accordingly.

To explore the data behind the AI visual's response, you can click the [info] button to reveal the underlying data. The information displayed is based on the settings defined for Tooltips when the visual was created. Additionally, you can explore other components of the dashboard to validate your insights and enhance your analysis.

The following steps demonstrate how to create an AI visual. For an overview of the shelves that specify this visual, see *Shelves for AI visual*.

Before you begin

Before using the AI visual in Cloudera Data Visualization, ensure the following prerequisites are met:

- Enable the AI visual feature in Site Settings AI Settings General Enable AI visual (Tech Preview) . For more information, see Managing AI settings.
- Select the AI engine and configure the necessary settings for the selected engine. For more information, see Managing AI settings.
- Ensure the data you intend to use with the AI visual has been vectorized.
- Ensure that a data connection has been configured for a data source that supports columns of vector type.
  - If using Solr as the vector database, note that Solr vector database integration in Cloudera Data Visualization requires Solr 9 and does not work with Solr 8.
- You need to enable the data profiling feature in Site Settings Data Enable Data Profiling . For more information, see Managing data related site settings.

Procedure

1. Upload a CSV file that contains the required data to a supported vector database connection type.

   For data upload instructions, see *Importing data in CSV format*.

   Note: Make sure the CSV file includes a vector column with pre-created embeddings.

2. Create a dataset from the uploaded data.

   For instructions, see *Creating a dataset*.

3. Create a new dashboard.
   For instructions, see *Creating a dashboard*.
4. Add the AI visual to the dashboard.
   a) In the VISUALS menu, click the icon of the AI visual.

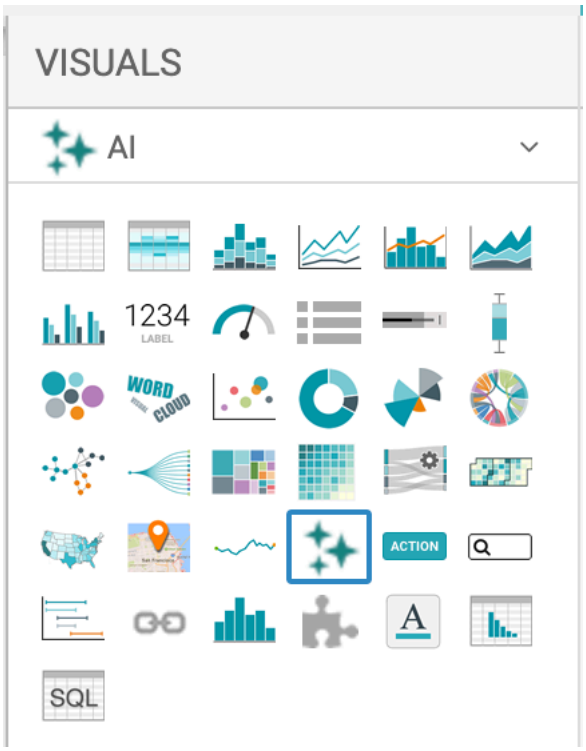

   b) Populate the shelves from the fields available in the DATA pane.
      1. **Embeddings**: Add vector fields containing embeddings.
      2. **Embedding Context**: Add fields that you want to include in the chat prompt.
      3. **Tooltip**: Define the source information to be included in the result. This information appears when the user hovers over the response's Info icon, but it is not sent to the completion service.
      4. **Limit**: Define the number of data rows from the vector database that are processed by the visual.

      For more information, see *Shelves for AI visual* .

5. Configure the visual settings by clicking Settings in the right-side VISUAL menu.
   a) Display
      1. Customize the chat welcome message, which is displayed as the first chat message from the AI visual.
      2. Set the display format for the conversation.

         You have two options: plain text or markdown
   b) Embeddings
      1. Maximum tokens: Set the maximum number of tokens. You can provide a zero or negative value to disable this setting.
      2. Selected profile: Displays the default profile, which you can change if needed.
   c) Completion
      1. Selected profile: Displays the default profile, which you can change if needed.
      2. Maximum tokens: Set a limit for the tokens. If exceeded, the request is not sent to the service and an error message is displayed.
      3. Context Overflow Policy: Choose how to handle situations where the length of generated tokens exceeds the context window size:
         - Throw an error: Raises an error when the conversation length exceeds the defined context window.
         - Remove old conversation first: Automatically manages conversation length by removing old tokens.
         - Truncate embedding context first: Truncates the embedding context, retaining the latest tokens.
      4. Context prompt: Specify the first prompt message for the chat.
      5. Question prompt: Specify the formatting of the user question.

Related Information

Importing data in CSV format
Creating a dataset
Creating a dashboard

Shelves for AI visual

Overview of shelves for Cloudera Data Visualization AI visual.

An AI visual has the following shelves:

Embeddings

This shelf is used for the vector database comparisons. The user input is vectorized and compared to the defined embeddings to get the most similar rows that contain the embeddings. The first field that contains results is returned.

Add a Dimension or Measures vector field that contains embeddings.

This is a mandatory shelf that accepts multiple fields.

Embedding Context

Add fields that you want to send as part of the chat prompt as a formatted version of all of the embedding context data.

This is a mandatory shelf that accepts multiple fields.

Tooltip

This setting enables the addition of a field for source information to be included in the displayed results.

Note: The Tooltip shelf is limited to one field for the AI visual type.

Limit

This setting defines the number of data rows that are processed by the visual.

Action

Cloudera Data Visualization enables you to include interactive, actionable elements in your dashboards and applications. You can use the Action visual to provide access to external sources of information or to internal search within your dataset.

About this task

The three main use cases of the Action visual are:

- **Linking**

  You can use Action to navigate to an external website or another application, and to move between different parts of your dashboard.
- **Search**

  You can provide a shortcut to the search pane of your current dataset(s).

  Note: Before using search in Cloudera Data Visualization, you must enable the feature in the Site Settings. For instructions, see *Enabling NLS in Site Settings*.
- **Email**

  You can share your visuals through email by sending out your dashboard in an email.

  Note: Before using this function, you must define your email settings in the Site Settings to allow users to share visuals through email. For instructions, see *Email settings*.

The following steps demonstrate how to create an Action visual based on the Restaurant Inspection SF dataset.

Procedure

1. Start a new visual based on the Restaurant Inspection SF dataset.

   For instructions, see *Creating a visual*.

   By default, a table visual is displayed that shows all fields in the dataset.

2. In the VISUALS menu, find and click the Action icon.

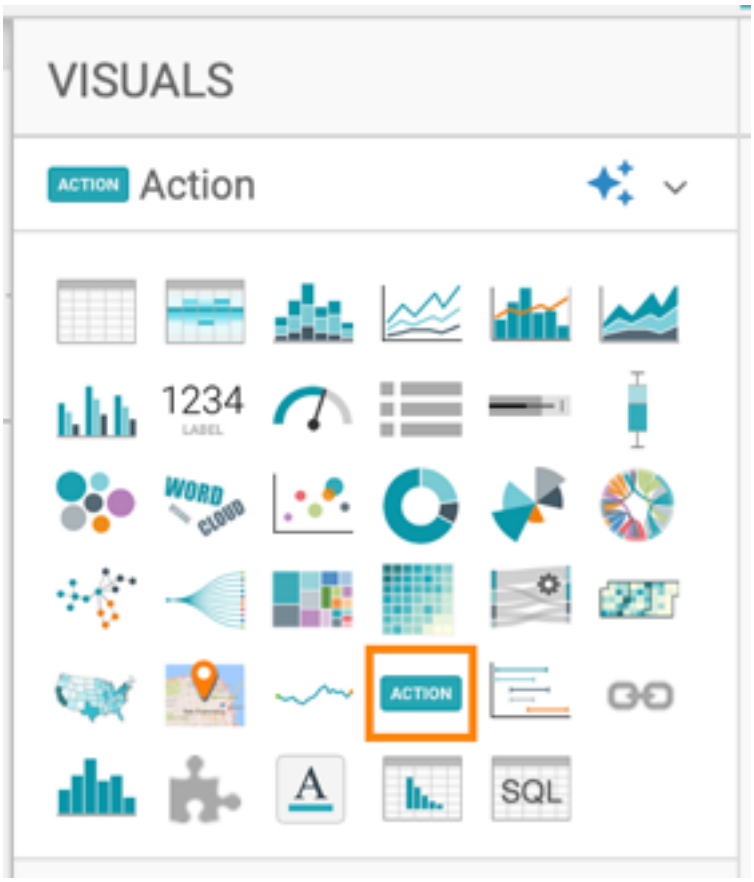

The shelves of the visual change.

3. From the Type drop-down list, choose the action element you want to add to your dashboard.

   You can select Button or Image.

   - If you choose Button, the remaining shelves are the following:

     Label

     Add the text you want to show on the button. The default value is ‘Click me!’.

     Action

     Select the action you want to invoke with the button. The options available in the drop-down list are Open URL, Search Dataset, and Send email.

     - If you choose Open URL, you can add the address of the webpage in the URL to open field.
     - If you choose Search Dataset, you can select an additional checkbox whether you want to include all datasets on the sheet.
     - If you choose Send email, you do not have to fill in additional fields.

     Button Type

     Select the button type that you want to add. The options available in the drop-down list have different designs. The default value is ‘Primary’.

     Button Size

     Select the size of the button from the drop-down list. The default value is ‘Normal’.

   - If you choose Image, the remaining shelves are the following:

     Image URL

     Add the URL of the image file that you want to add to your Action visual.

     Action

     Select the action you want to invoke with the button. The options available in the drop-down list are Open URL, Search Dataset, and Send email.

     - If you choose Open URL, you can add the address of the webpage in the URL to open field.
     - If you choose Search Dataset, you can select an additional checkbox whether you want to include all datasets on the sheet.
     - If you choose Send email, you do not have to fill in additional fields.

4. Click REFRESH VISUAL.

   The updated Action visual appears.

5. At the top left corner of the Dashboard Designer, click SAVE.

Example

In this example, you can see a dashboard with three action visuals and a table visual:

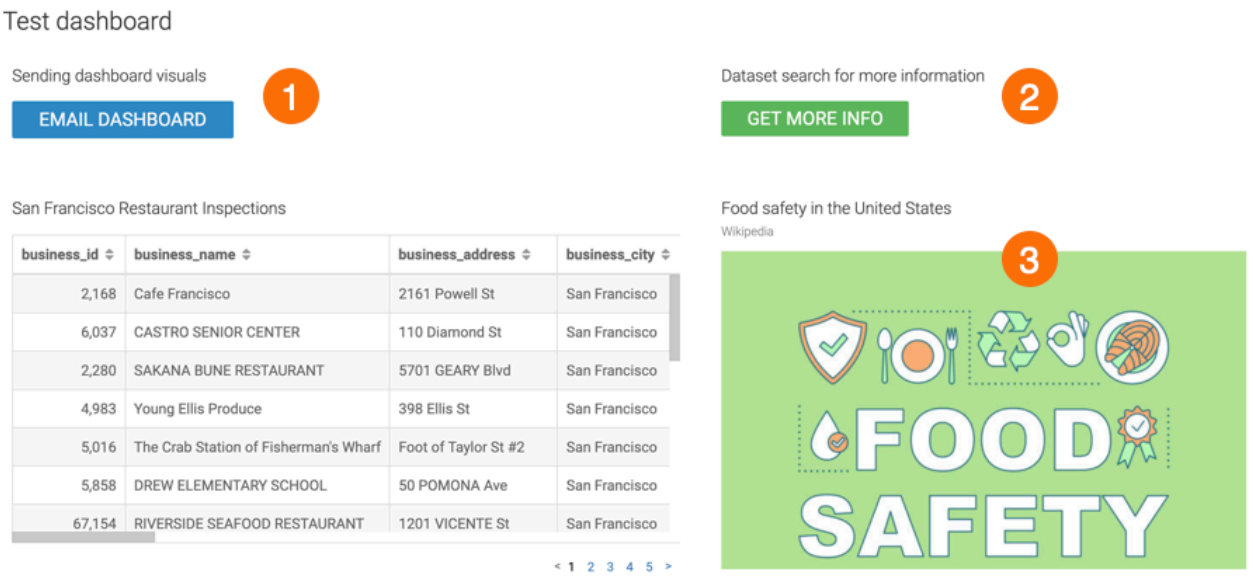
Test dashboard
Sending dashboard visuals
EMAIL DASHBOARD
1
Dataset search for more information
GET MORE INFO
2
San Francisco Restaurant Inspections
business_id
business_name
business_address
business_city
2,168 Cafe Francisco 2161 Powell St San Francisco
6,037 CASTRO SENIOR CENTER 110 Diamond St San Francisco
2,280 SAKANA BUNE RESTAURANT 5701 GEARY Blvd San Francisco
4,983 Young Ellis Produce 398 Ellis St San Francisco
5,016 The Crab Station of Fisherman's Wharf Foot of Taylor St #2 San Francisco
5,858 DREW ELEMENTARY SCHOOL 50 POMONA Ave San Francisco
67,154 RIVERSIDE SEAFOOD RESTAURANT 1201 VICENTE St San Francisco
< 1 2 3 4 5 >
Food safety in the United States
Wikipedia
3
FOOD
SAFETY

1. Action button for email

   The blue Action button is a large Primary button that you can use for sending the dashboard to an email address:

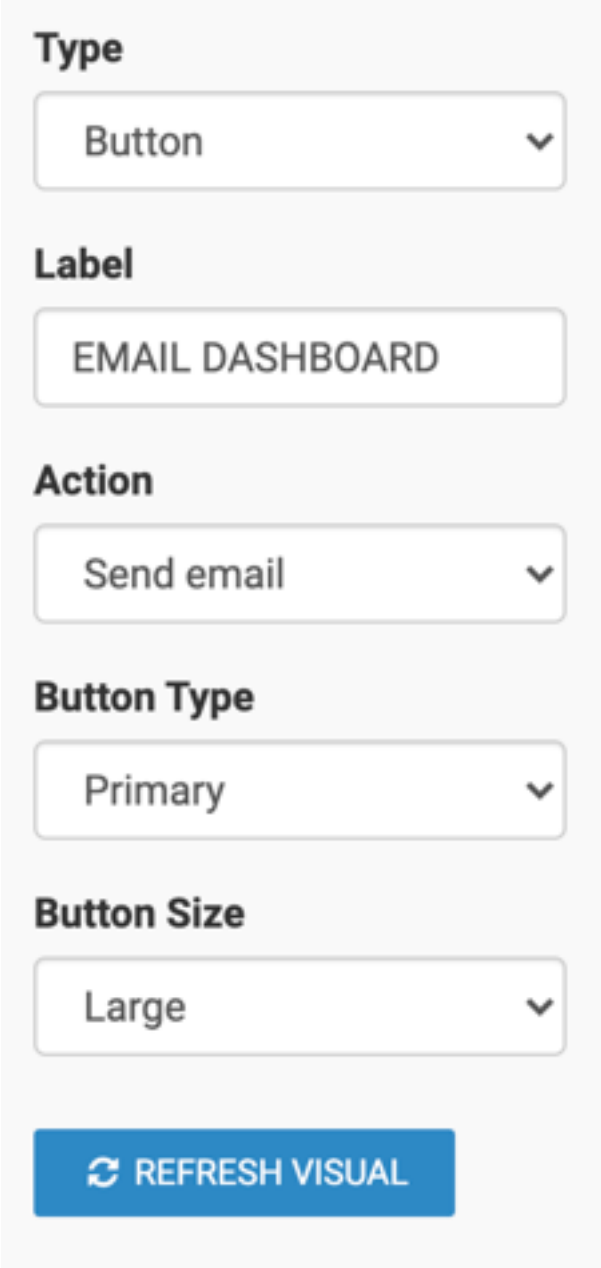

2. Action button for search

   The green Action button is a large Info button that you can use for searching the dataset:

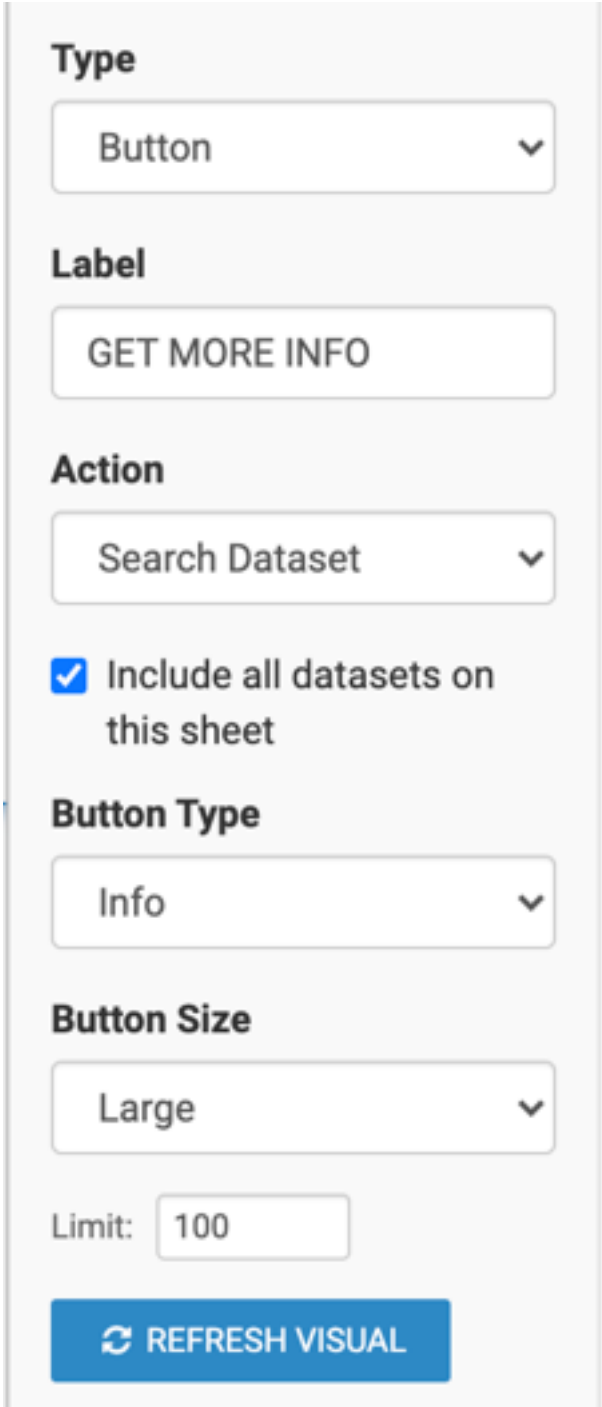

3. Action clickable image with link

   The Action image is clickable and it points to a Wikipedia page about food safety in the US:

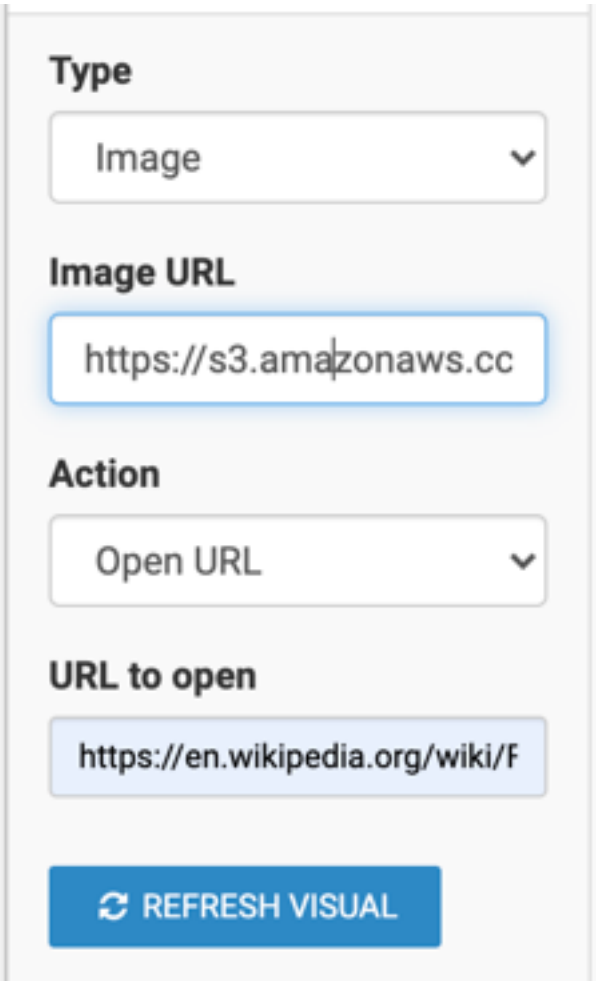

Related Information

Enabling NLS in Site Settings
Email settings
Creating a visual

Search

The purpose of the Search visual is to include search functionality in embedded dashboards. It enables you to place a data search button directly on the dashboard.

For instructions on how to create a Search visual, see *Creating a search visual*.

For more information, see *Using the search visual in a dashboard*.

Related Information

Creating a search visual
Using the search visual in a dashboard

Timeline

Cloudera Data Visualization allows you to create Timeline visuals. Timelines are graphical representations of activities/events displayed against time.

About this task

In a Timeline visual, each activity/event is represented by a bar of a time period along a suitable time scale. You need two different timestamps to define the projection of the time period. The starting point of the bar reflects the start date, the end point reflects the end date, and the length of the bar corresponds to the duration of the activity.

The following steps demonstrate how to create a Timeline visual based on the Earthquake Data January 2019 dataset. You can create simple linear representations or more complex visualizations. For an overview of the shelves that specify this visual, see *Shelves for Timelines*.

Procedure

1. Start a new visual based on the Earthquake Data January 2019 dataset.

   For instructions, see *Creating a visual*.

   By default, a table visual is displayed that shows all fields in the dataset.
2. In the VISUALS menu, find and click the Timeline icon.

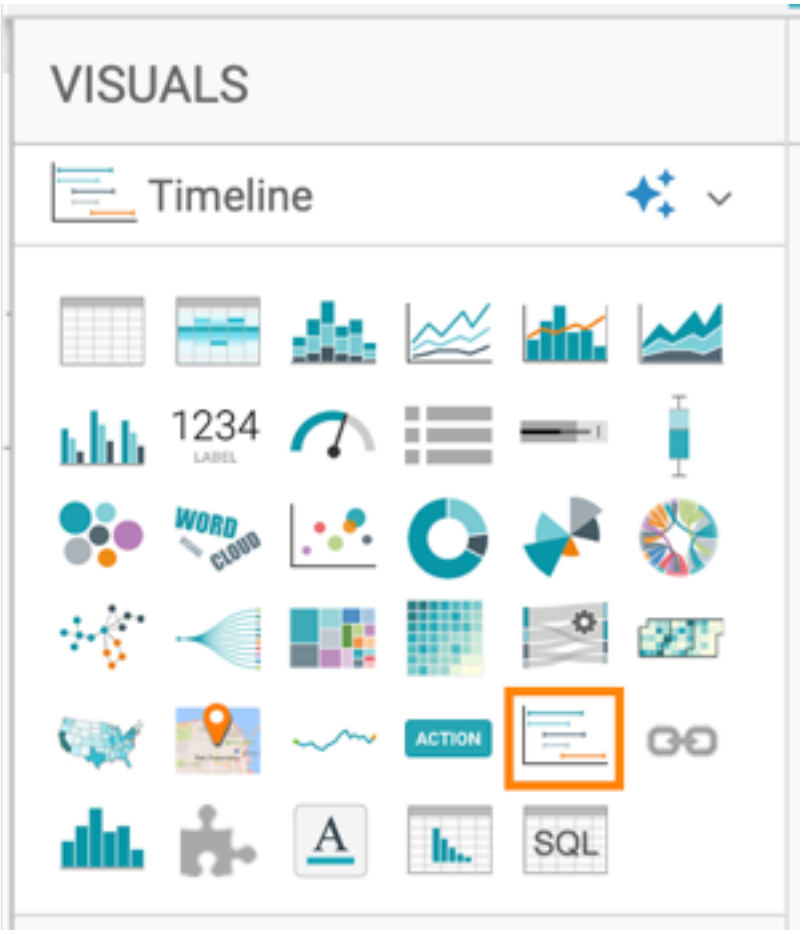

The shelves of the visual change.

3. Populate the shelves from the available fields (Dimensions and Measures) in the DATA menu.

   Make the following selections to recreate the example Timeline visual:

   - Add ‘time’ to the Start Time shelf.
   - Add ‘updated’ to the End Time shelf.
   - Add ‘Region’ to the Y axis.
   - Add ‘latitude’ to the Color shelf.
   - Specify ‘place’ as a label on the Labels shelf.
   - Define a filter for ‘type’. Select ‘earthquake’.
   - Define a filter for ‘longitude’. Only include earthquakes between -60° and 0°.

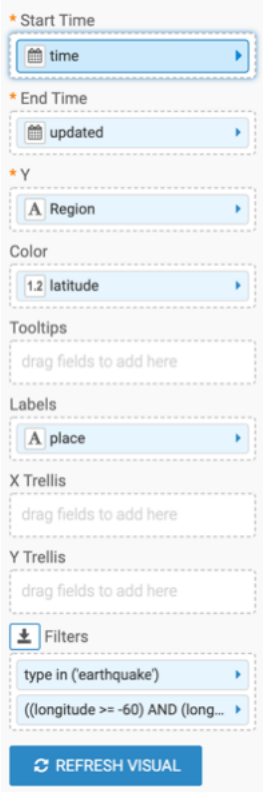

   For further information about the shelves of the visual, see *Shelves for Timelines*.

4. Click REFRESH VISUAL.

   The updated Timeline visual appears.

   Legend is on by default, and it appears on the right side of the visuals. If you want to change the legend style or remove it from the visual, see Adding and removing legend.

5. Add the title Earthquakes between -60 and 0 longtitude.
6. At the top left corner of the Dashboard Builder, click SAVE.

Results

Your visual should look like the following:

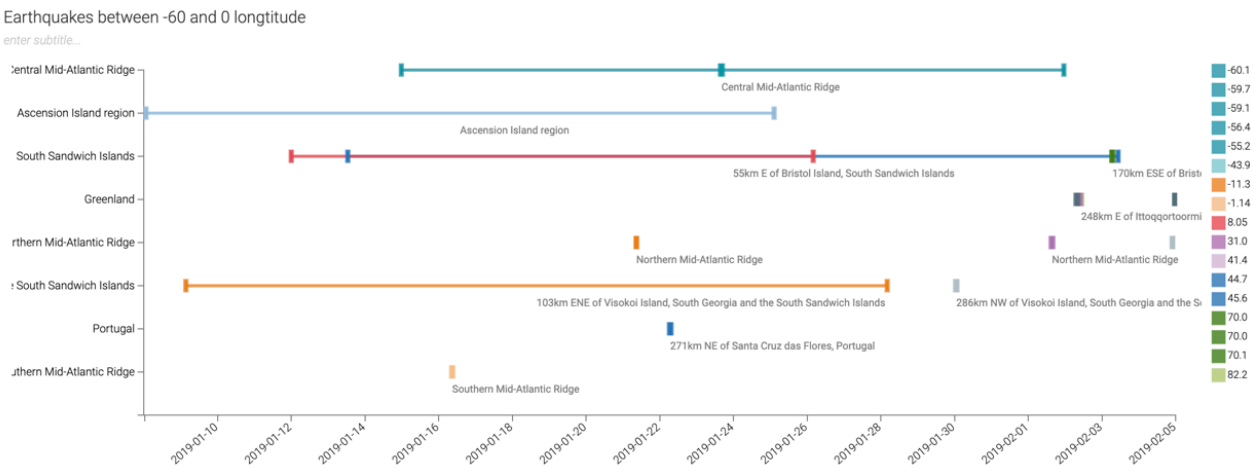

Details of a particular timeline are shown in tooltips if you hoove over:

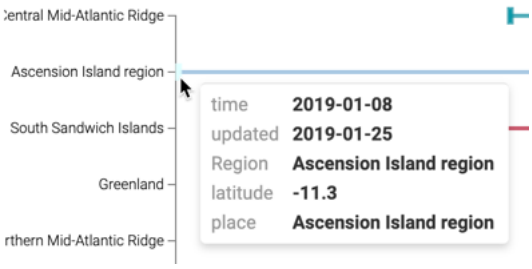

Related Information

Creating a visual

Adding and removing legend

Shelves for timelines

Cloudera Data Visualization Timeline visuals have the following shelves.

Start Time

This shelf is required and it is limited to one field.

You can add a time dimension from the DATA menu to specify the starting point of the analyzed timescale.

Note: The Display Format field property defined for the Start Time shelf will be used for the End Time shelf as well.

End Time

This shelf is required and it is limited to one field.

You can add a time dimension from the DATA menu to specify the ending point of the analyzed timescale.

Y

This shelf is required and it accepts multiple fields.

You can add one or more dimensions that uniquely identify the elements on the Y axis.

Color

This shelf is optional and it is limited to one field.

You can define how colors are used in the visual.

Tooltips

This shelf is optional and it accepts multiple fields.

You can add measures from the DATA menu for additional information in the graph's tooltips.

Labels

This shelf accepts multiple fields.

You can define what information should appear in the graph as data labeling.

X Trellis

This shelf is optional and it accepts multiple fields.

You can use a dimension field from the DATA menu to trellis the visual into columns.

Y Trellis

This shelf is optional and it accepts multiple fields.

You can use fields from the DATA menu to trellis the visual into rows.

Filters

This shelf is optional and it accepts multiple fields.

You can add measures and dimensions from the DATA menu to set exclude or include criteria for the visual.

External link visuals

Cloudera Data Visualization enables you to create External Link visuals.

About this task

A link visual enhances your applications by provding access to external sources of information. These sources may either be static URL addresses, or they can be dynamic serach results to which you pass a name of a variable used in the dataset.

The following steps demonstrate how to create a new External Link visual for the World Life Expectancy dataset. In this exercise, the variable country is passed to the Wikipedia site.

For an overview of shelves that specify this visual, see *Shelves for external links*.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.

   For instructions, see *Creating a visual*.

2. Select Link in the VISUALS menu.

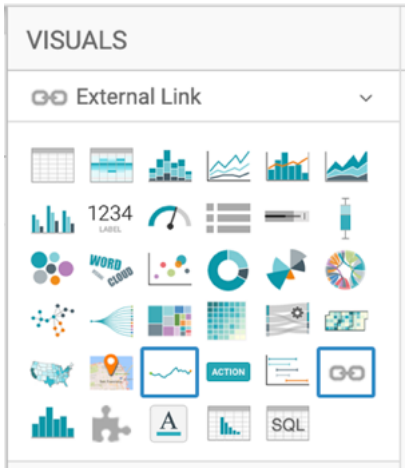

The shelves of the visual changed. The only shelf is URL.

3. In the URL field, enter the following text:

```
https://en.wikipedia.org/wiki/Afghanistan
```

4. Click REFRESH VISUAL.

The page specified in the URL appears.

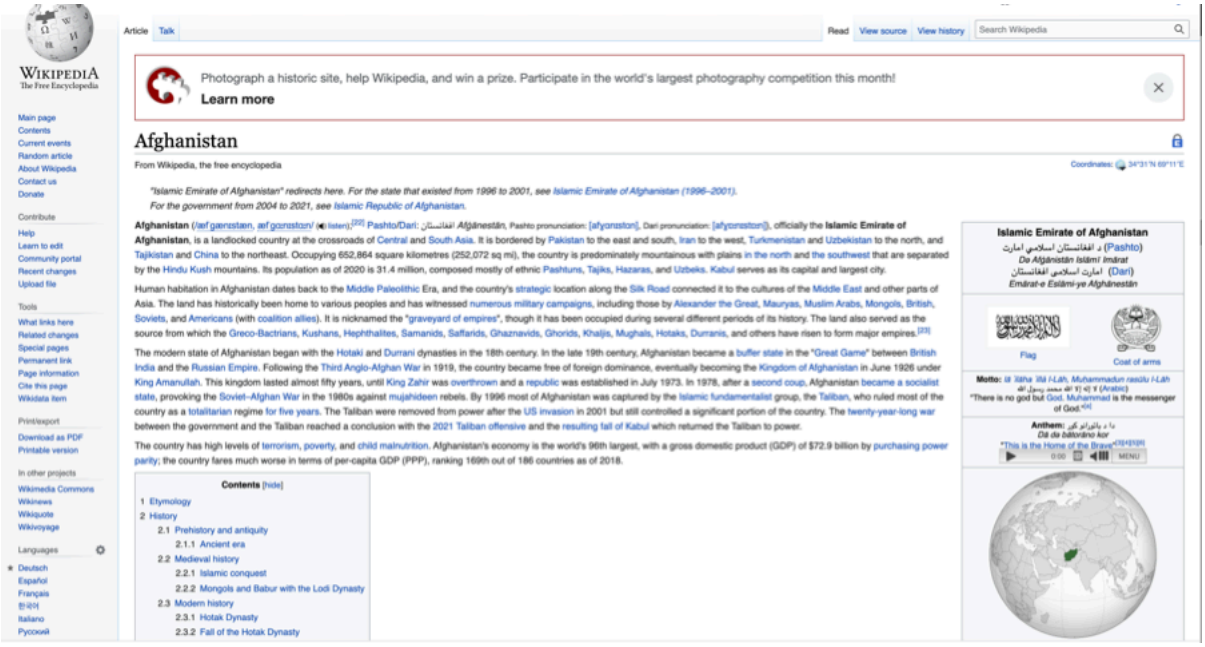

5. Click the (pencil icon) next to the title of the visualization to edit it, and change the title to World Population - C ountry UTL.
6. At the top left corner of the Dashboard Designer, click SAVE.

Setting up dynamic external links

You need to perform extra set-up steps if you want to create a visual with dynamic links.

About this task

To use a visual to see the Wikipedia page for a specific country, follow these steps:

Procedure

1. Create a new application for the World Life Expectancy dataset.
2. Add the World Population - Bars Bars visual that you created in *Bars*.
3. Add World Population - Country UTL as the title of the application.
4. Add World Population as the title of the active menu tab.
5. At the top left corner of the App Designer, click SAVE.
6. In the application, click the Bars visual to open it in a new tab.
7. Click EDIT to open the visual in Edit mode.

8. Click the visual and select Modify click behavior.

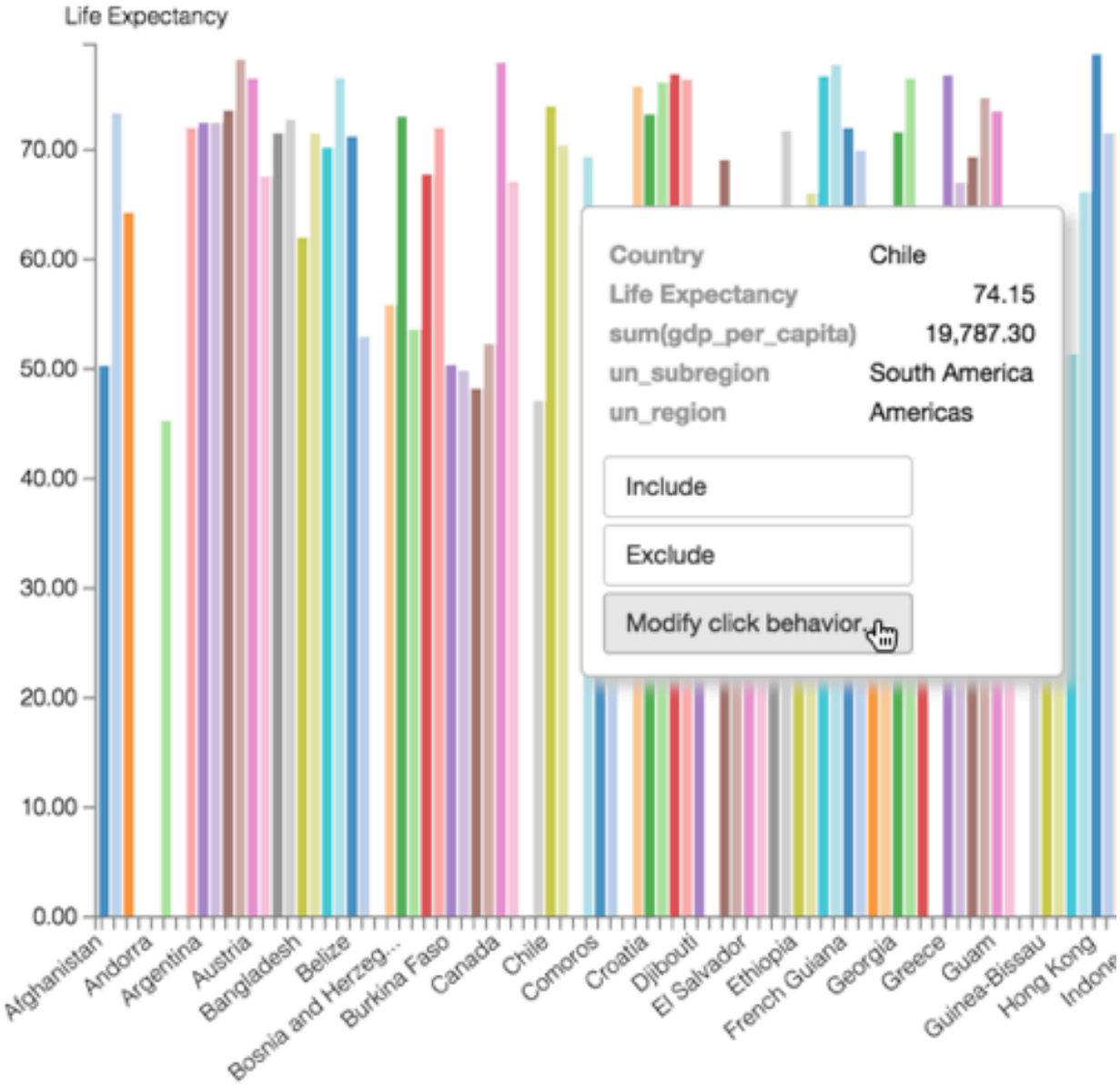

9. In the Pick an app modal window, under To set this selection, pick an app below, select World Life Expectancy.

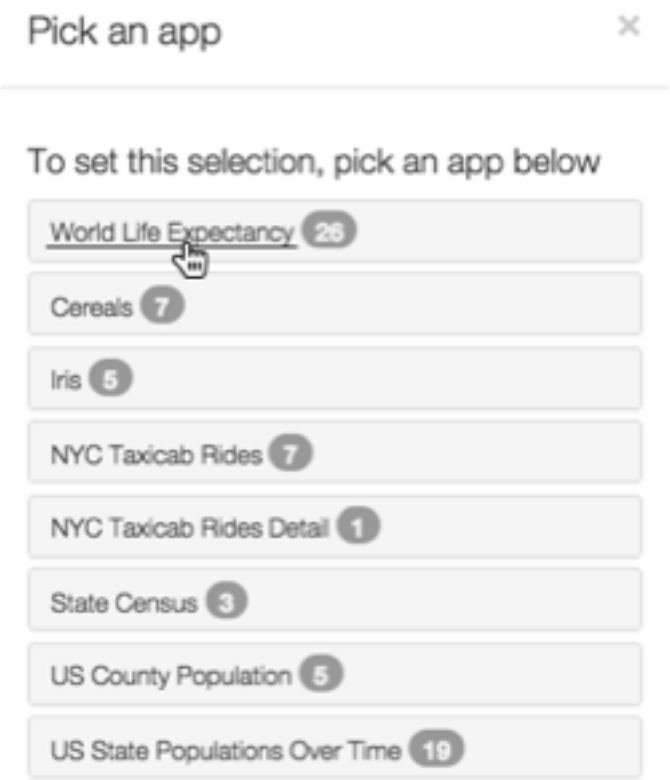

10. After the modal window changes, select World Population - Composite.

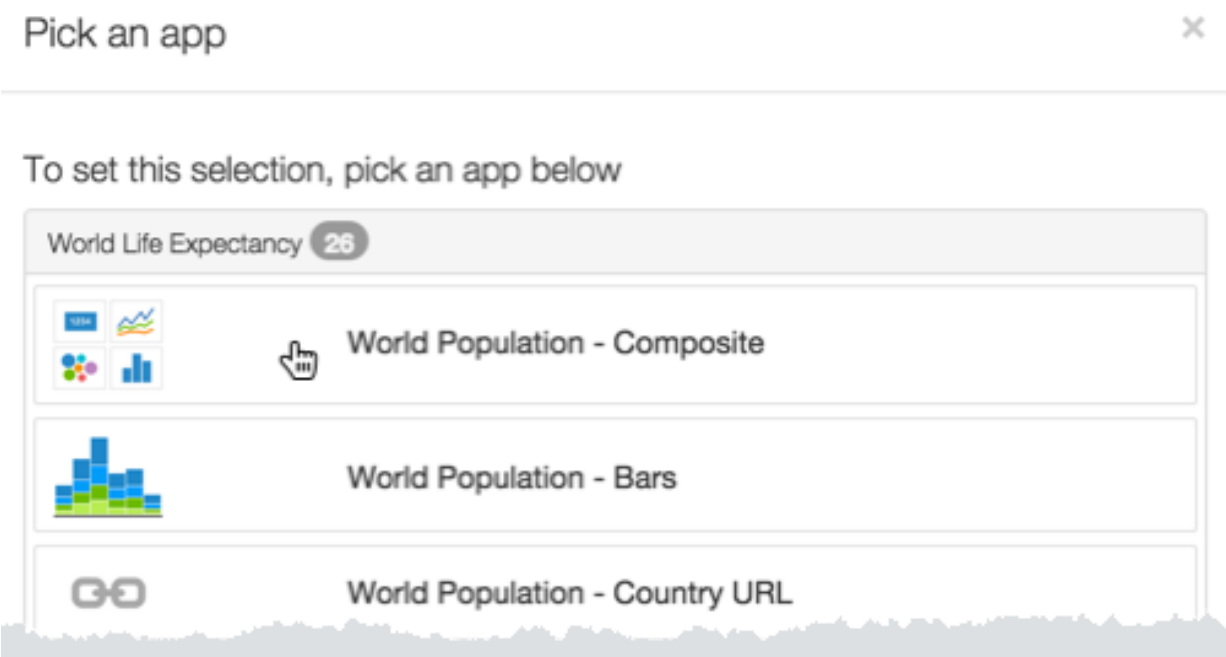

11. At the top left corner of the Dashboard Designer, click SAVE.

12. While in Edit mode, ensure that the settings work:

a) Left-click the visual.

b) Select Test click behavior.

Results

In the World Population - Composite app, when you click on the specific bars that represent a country, the Wikipedia page changes to match the country.

Shelves for external links

Overview of shelves for Cloudera Data Visualization External Link visuals.

An External Link has only one shelf:

URL

Add a URL address of an external site.

This shelf accepts multiple fields, but plots a map with the highest possible resolution.

Histograms

You can create Histogram visuals in Cloudera Data Visualization. Histograms enable the quick visual analysis of numerical data distribution. They provide an estimate of the frequency distribution of a continuous quantitative variable, accomplished by splitting it into consecutive, non-overlapping intervals, referred to as "buckets".

About this task

Note: Histograms can only be plotted for bare metrics, not aggregates. For example, you can plot a histogram for population, not for avg(population).

To work around this limitation, first save the calculated metric as derived data, and then use this field as a measure in a histogram. See *Derived data*.

When you change the number of buckets (the default is 10), Cloudera Data Visualizationn determines the data range and splits it equally among the buckets, groups the metric into these buckets, and plots vertical bars to represent it.

Histograms support both normalized and cumulative forms, and may be used in a trellis formation.

The following steps demonstrate how to create a new histogram visual on the World Life Expectancy dataset.

For an overview of shelves that specify this visual, see *Shelves for histograms*.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.

For instructions, see *Creating a visual*.

2. In the VISUALS menu, select Histogram.

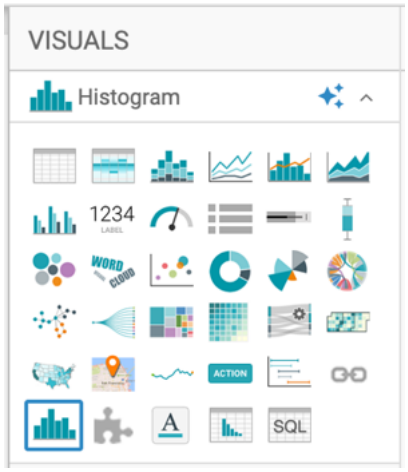

The shelves of the visual changed. You can see: Measure (mandatory), X Trellis, Y Trellis, and Filters.

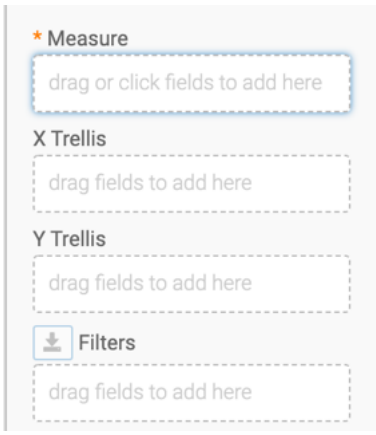

3. To show specific items, populate the shelves from the available fields (X Trellis, Y Trellis) in the DATA menu.

   Under Measures, select population and drag it to the Measures shelf.

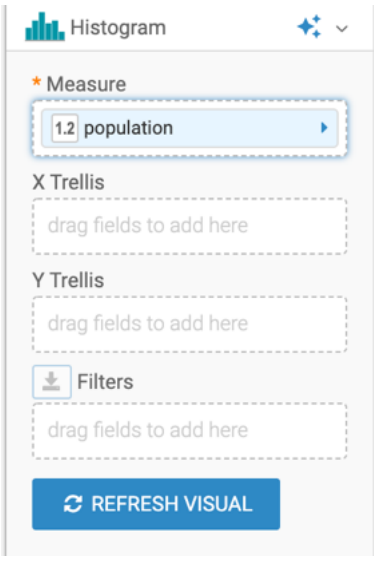

4. Click REFRESH VISUAL.

   The default histogram visual appears, dividing the data into 10 buckets. As you can see, most the data is in the first bucket, and this visual is not very enlightening.

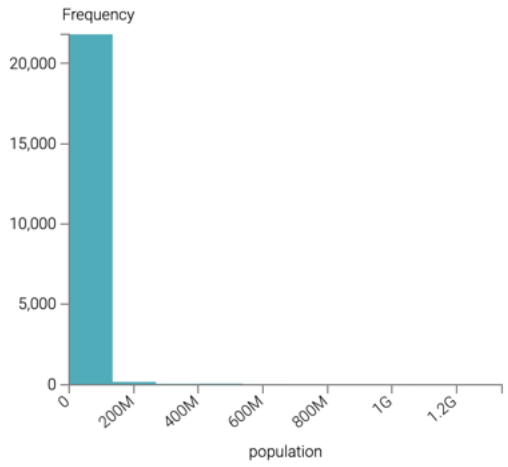

5. On the Filters shelf, add several Dimensions and Measures from the DATA menu.

   This enables you to dynamically control the data input and discover the data at a more granular level.

   For example, from Dimensions, drag year to the Filters shelf and select the year 2010.

   You can also add un_region to the Filters shelf and select Africa.

6. Click REFRESH VISUAL.

   The shape of the diagram has changed. Hovering the mouse over the histogram shows the data ranges and frequency for each bucket of the histogram.

   You can see that the filtering options significantly reduced result set.

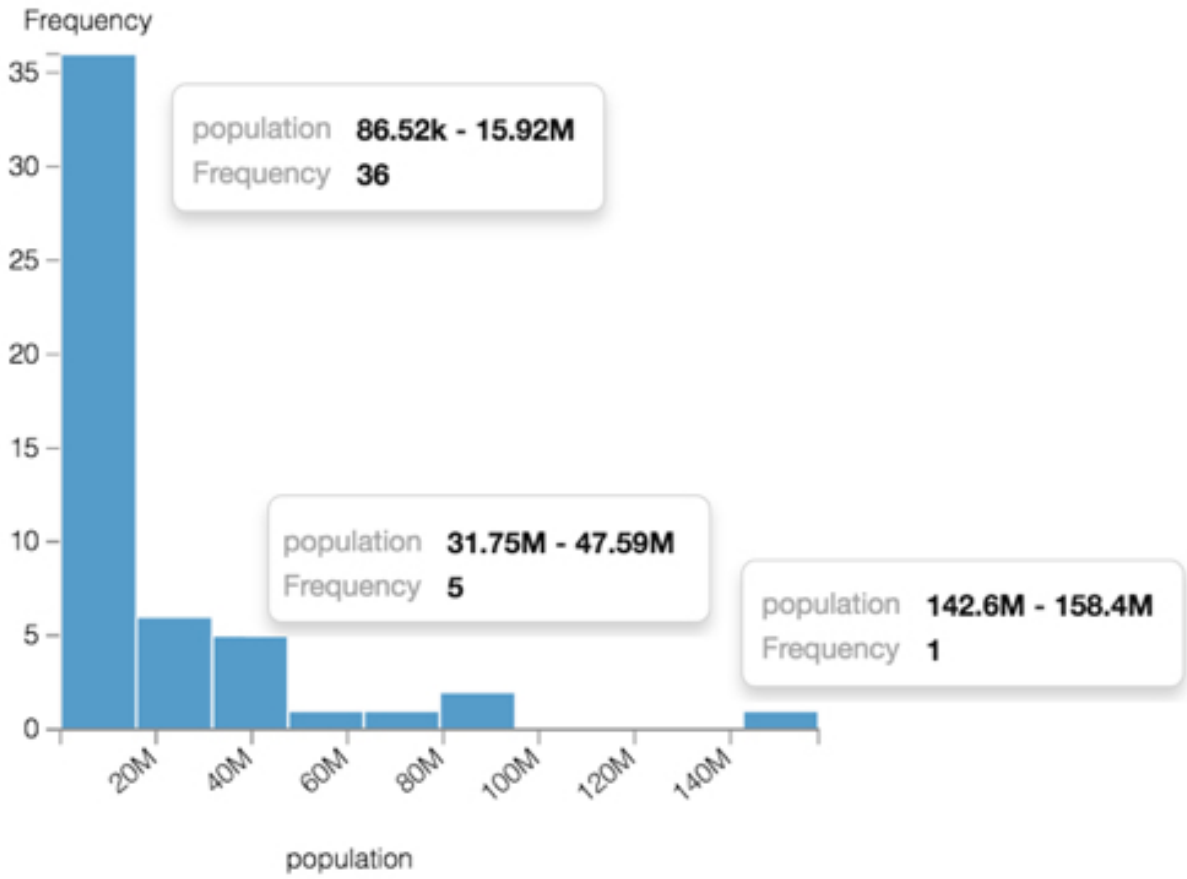

7. Change the title to World Population - Histogram.

   [Optional] You can also add a brief description of the visual as a subtitle below the title of the visualization.

8. Click SAVE at the top left corner of the Dashboard Designer.

Specifying the bucket count

The default histogram displays data in 10 buckets. You can easily adjust this setting to control the granularity of your data display.

Procedure

1. On the right side of the Dashboard Designer, go to the VISUAL menu and click Settings.
2. In the Settings menu, click Basic to expand its options.

3. Find the Bucket count field and use the selector to increase or decrease the number of buckets as desired.

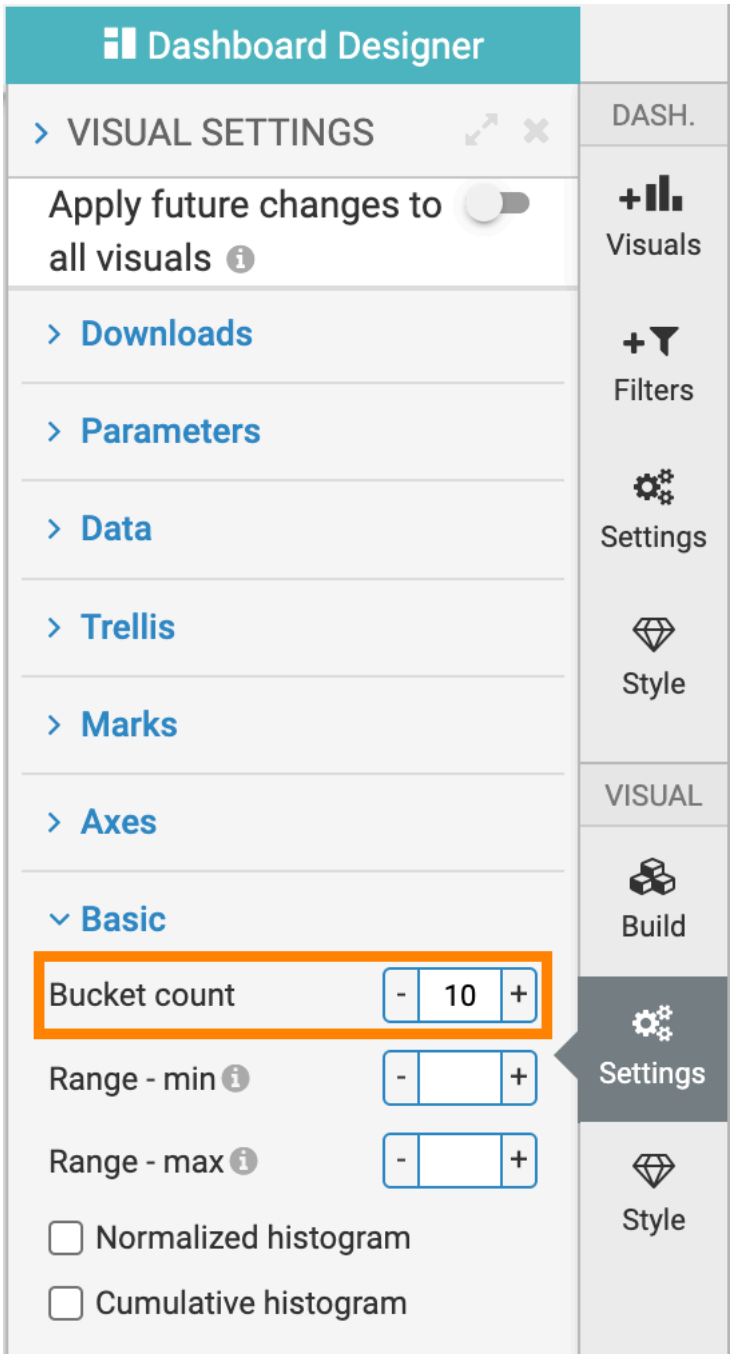

Example

Increasing the bucket count results in narrower buckets, meaning each bucket represents a smaller range of values. This can provide a more detailed view of the data distribution. Conversely, reducing the bucket count widens each bucket, simplifying the visualization and giving a broader overview.

The following example shows a histogram with 10 buckets compared to a histogram with 20 buckets. You can see how doubling the number of buckets reduces the range of values in each bucket by half, providing a more granular look at data distribution.

Specifying data range

You can specify the exact portion of your dataset to visualize, allowing you to focus on specific data intervals. This is especially useful for narrowing in on relevant subsets without including outliers or unnecessary values.

Procedure

1. On the right side of the Dashboard Designer, go to the VISUAL menu and click Settings.
2. In the Settings menu, click Basic to expand its options.

3. Define the range of values to focus on for your histogram.
   a) In the Range - min field, enter the minimum value for your range.
      This will be the starting point for the histogram's x-axis.
   b) In the Range - max field, enter the maximum value for your range.
   This will define the endpoint for the histogram's x-axis.

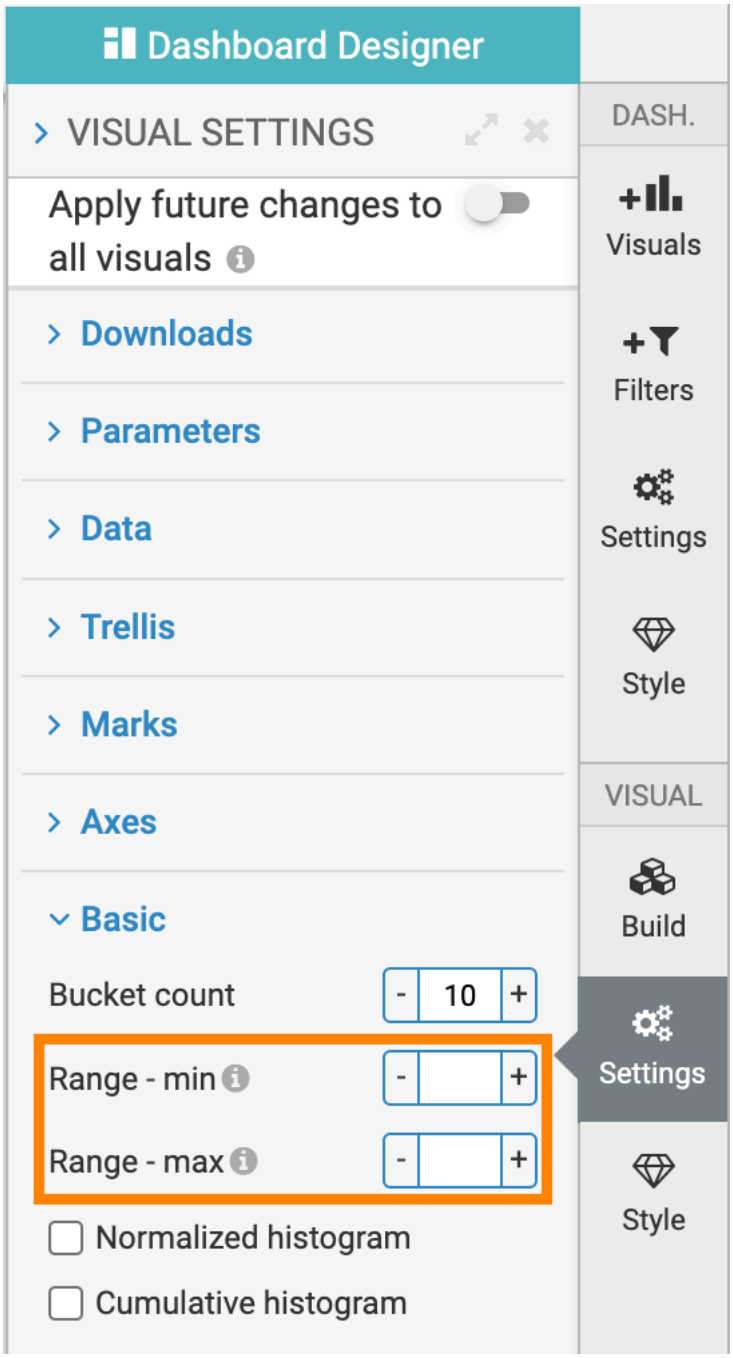

After setting these values, the histogram will automatically update to display only the data within the specified range. You can adjust the values as needed to explore different parts of your dataset.

Example

Using the range setting, you can easily control the focus of your histogram, allowing you to isolate and analyze specific intervals of your data.

The following example shows a histogram with a wider range compared to a histogram with narrower range. You can see how adjusting the range changes the histogram's appearance by displaying more or fewer data points.

Normalizing the histogram

By default, the vertical axis of a histogram represents the count of values that map to a particular bucket. To represent these values as percentages of the total, you can normalize the histogram. This adjustment changes the histogram counts so they sum to 1, with each bar representing the proportionate frequency.

Procedure

1. On the right side of the Dashboard Designer, go to the VISUAL menu and click Settings.
2. In the Settings menu, click Basic to expand its options.
3. Select the Normalized histogram option.

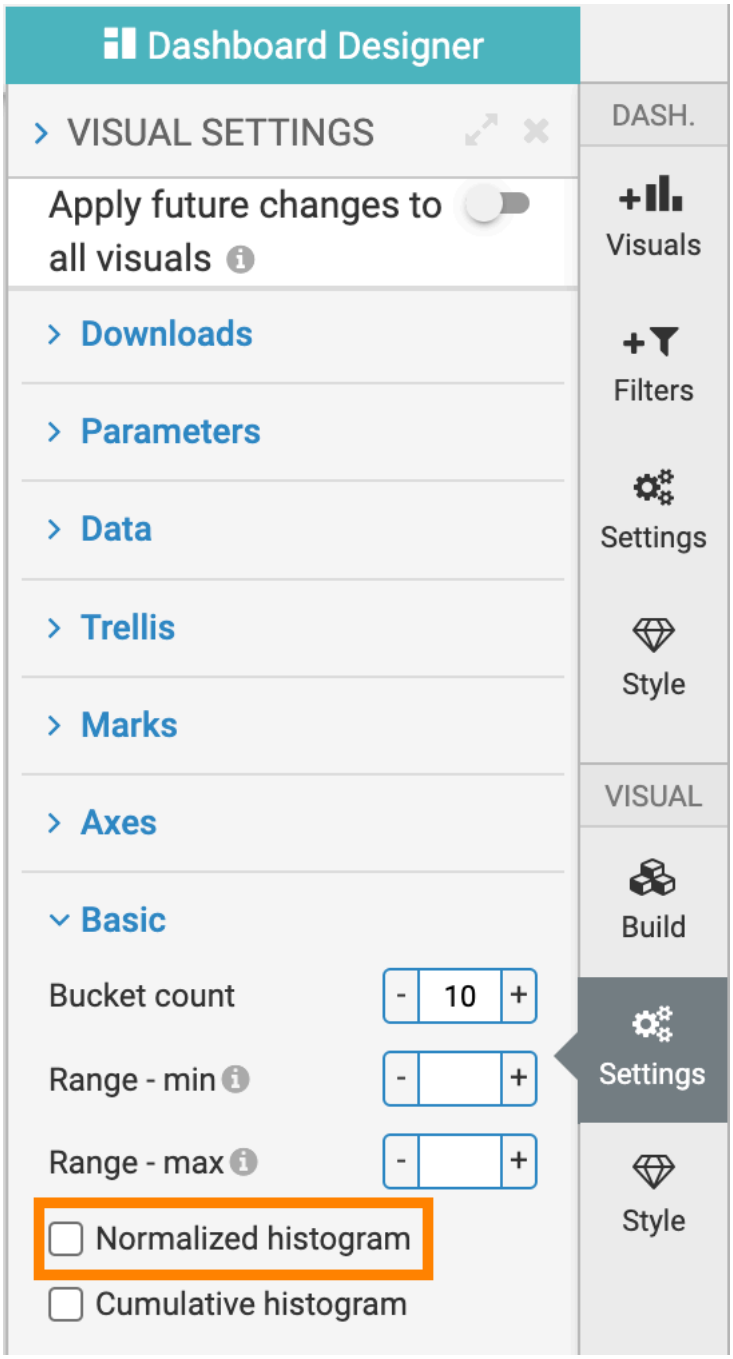

Example

The following image compares two versions of a histogram. In the standard histogram (top), the vertical axis and tooltips display raw counts. In the normalized histogram (bottom), the vertical axis and tooltips show values as percentages, indicating the relative frequency of each bucket.

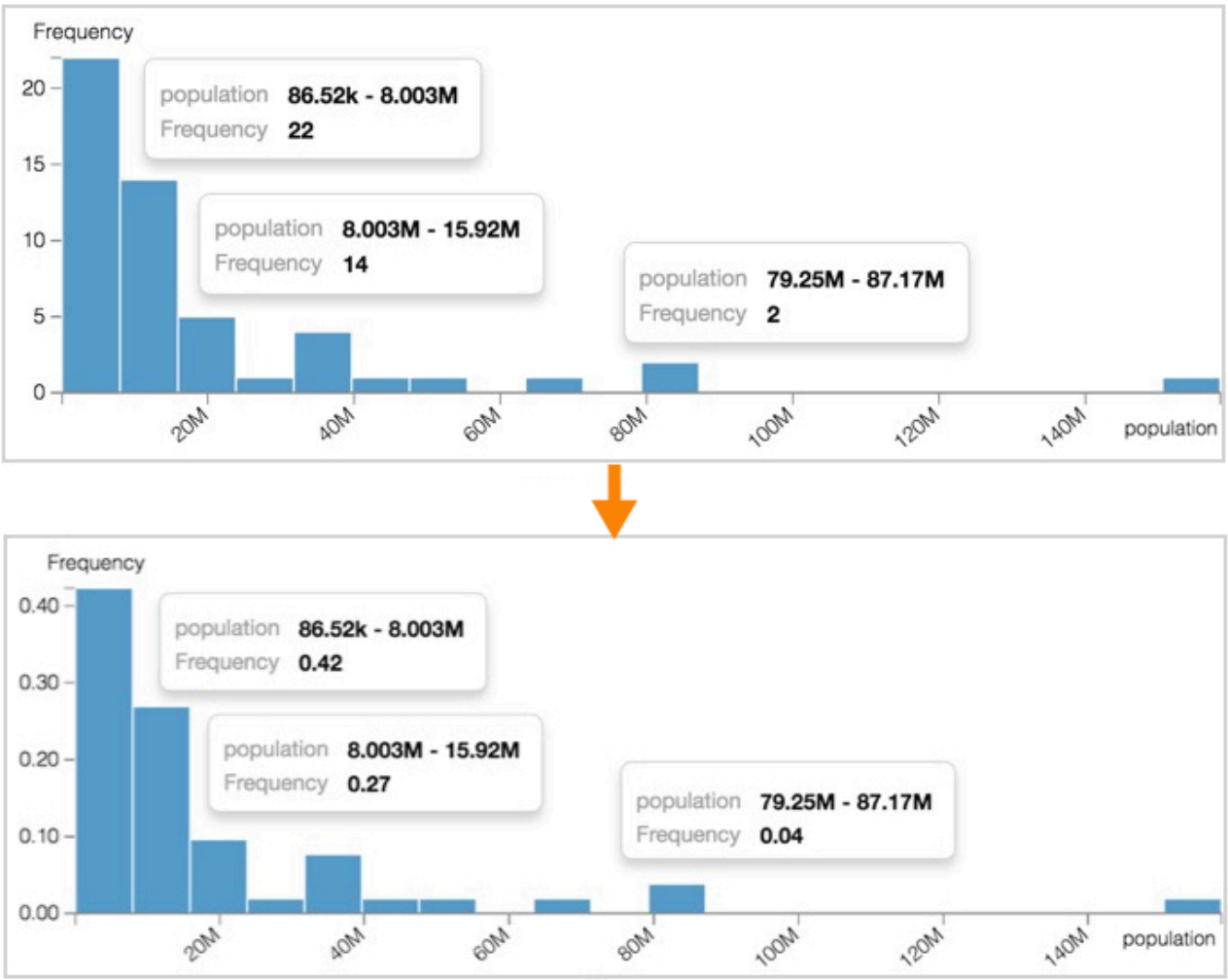

Using the cumulative option

By default, a histogram displays the count or frequency for each bucket individually. Enabling the cumulative option adjusts the histogram so that each bucket's count or frequency is added to the previous total. As a result, the right-most bucket shows the total count or 1 (100%) if the histogram is normalized.

Procedure

1. On the right side of the Dashboard Designer, go to the VISUAL menu and click Settings.
2. In the Settings menu, click Basic to expand its options.

3. Select the Cumulative histogram option.

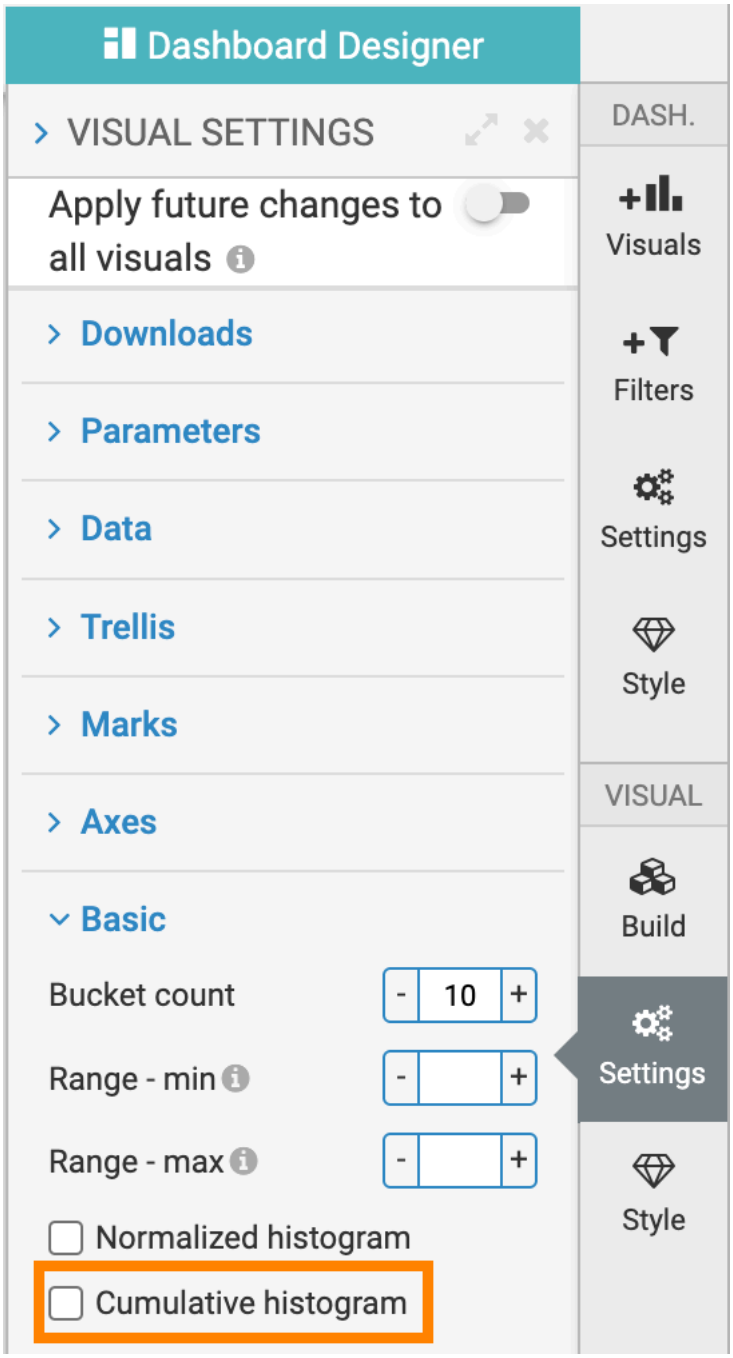

Example

The following image shows the difference in the vertical axes. In the lower image, the cumulative values build up towards 1 (or 100%) on the right side of the graph, reflecting the running total.

Shelves for histograms

Overview of shelves for Cloudera Data Visualization Histogram visuals.

A Histogram visual has the following shelves:

X Trellis

Use a Dimension field from the DATA menu to trellis the visual into columns.

This shelf accepts multiple fields.

Y Trellis

Use a Dimension field from the DATA menu to trellis the visual into rows.

This shelf accepts multiple fields.

Measure

Add a Measure field from the DATA menu to generate a histogram of its values.

This shelf accepts a single field.

Filters

Add Dimension or Measure fields from the DATA menu to include or exclude categorical or value criteria.

This shelf accepts multiple fields.

Extensions

In Cloudera Data Visualization, you can use extension visuals to add supporting textual, image, or behavioral aspects to your dashboard.

About this task

The following steps demonstrate how to create a simple extension visual with text content and minor HTML formatting.

Note: You must have system-level Manage styles and settings privilege to create Extension visuals. See *Setting system privileges* for details.

Procedure

1. Start a new visual based on the US State Populations Over Time dataset.

   For instructions, see *Creating a visual*.
2. Select Extension in the VISUALS menu.

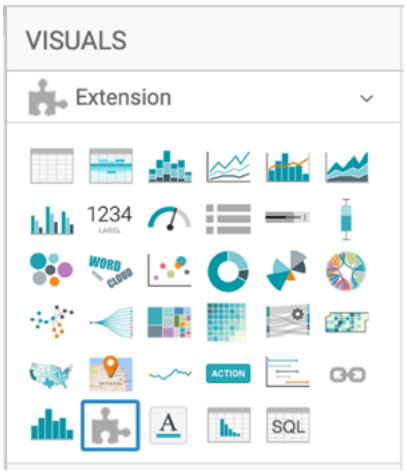

   The shelves of the have visual changed. This visual type has a single text box for entering and editing text or markup.

3. Click the (expand) icon or the link in the Dashboard Designer to open the HTML editor.

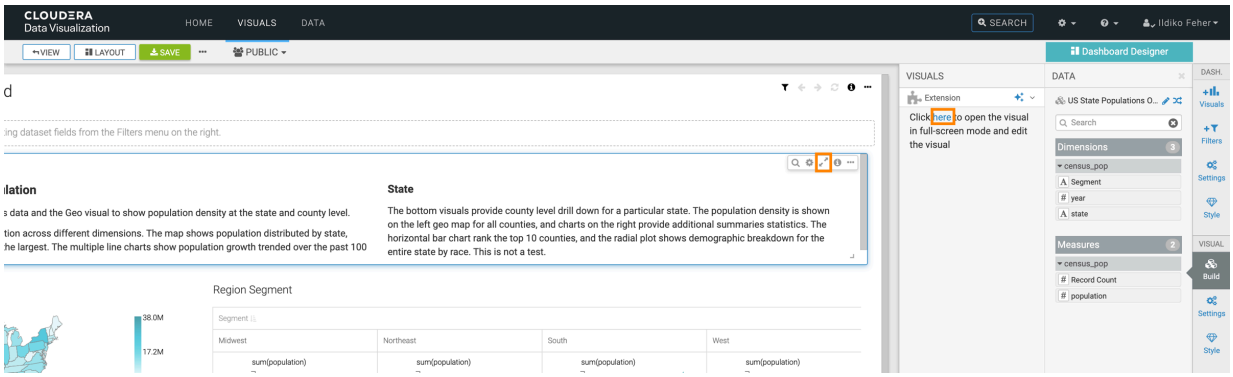

The editor modal window has two columns: one for code and the other one for preview.

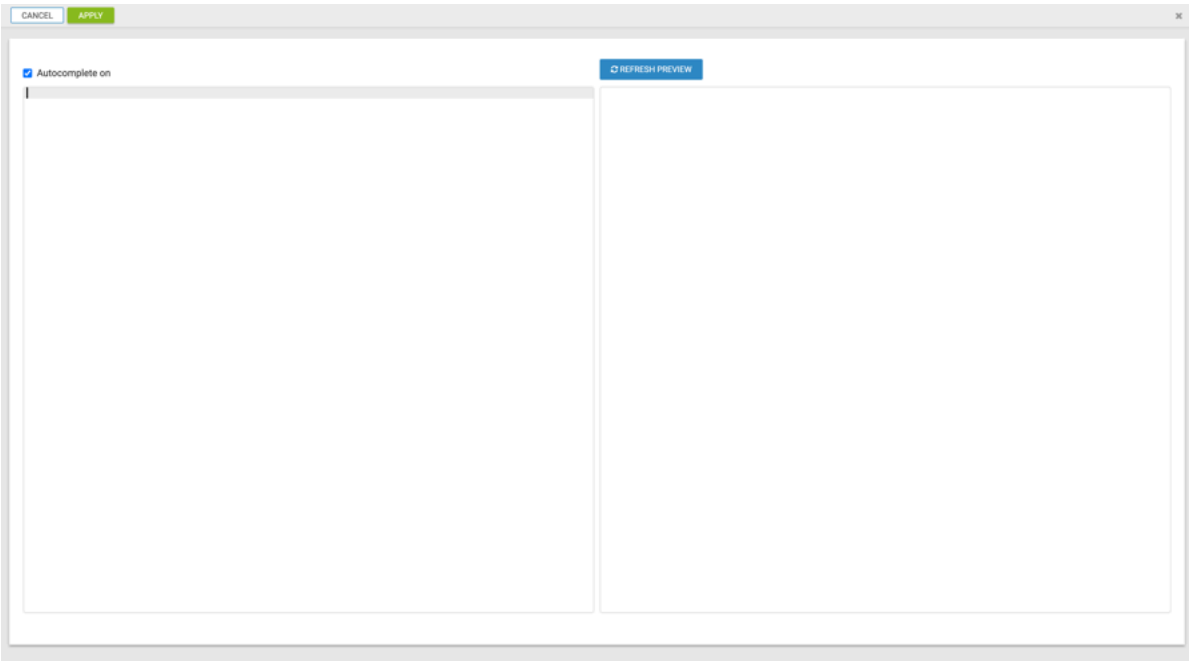

4. Paste a code block in the text box for code.

   In this example, the following code block is used:

```
<h1>2010 United States Census</h1>
<p>From Wikipedia, the free encyclopedia</p>
<h2>General information</h2>
<ul>
<li>Country: United States</li>
<li>Date taken: April 1, 2010</li>
<li>Total population: 308,745,538</li>
<li>Percent change: Increase 9.7%</li>
<li>Most populous state: California (37,253,956)</li>
<li>Least populous state: Wyoming (563,626)</li>
</ul>
<p>The 2010 United States Census, known as "Census 2010", is the twenty-
third
and most recent United States national census. National Census Day, the r
eference
day used for the census, was April 1, 2010. As part of a drive to increase
 the
count's accuracy, 635,000 temporary enumerators were hired. The population
 of
```

```
the United States was counted as 308,745,538, a 9.7% increase from the
2000 Census.</p>
```

5. Click REFRESH PREVIEW to view the changes in the preview column.

   The formatted text appears in the preview area of the HTML editor. You can make further edits to the HTML code and check the changes by refreshing the preview again.

   If you do not need the autocomplete function, uncheck the Autocomplete on option.

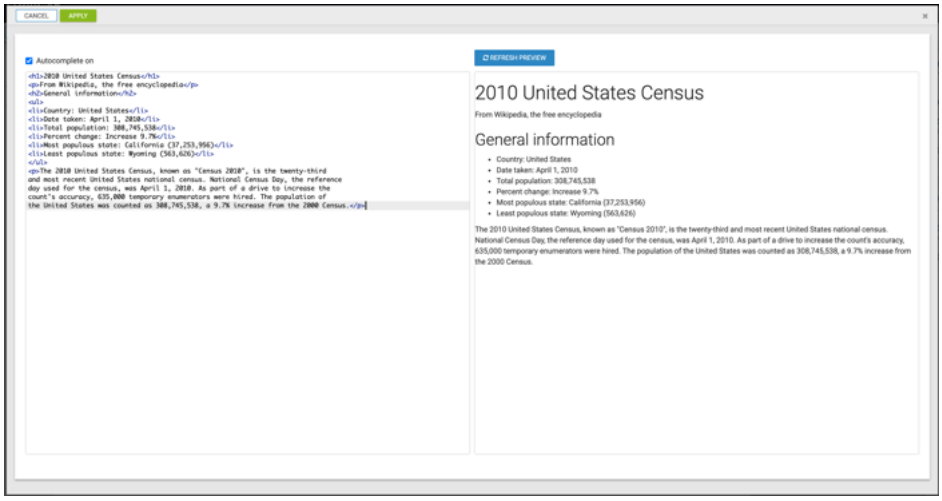

6. Click APPLY to save the changes.

   If you do not want to apply your changes, click CANCEL or just close the modal window.

7. Change the title to 2010 Census Information.

   [Optional] You can also add a brief description of the visual as a subtitle below the title of the visualization.

8. Click SAVE at the top left corner of the Dashboard Designer.

Results

Your visual should look something like this:

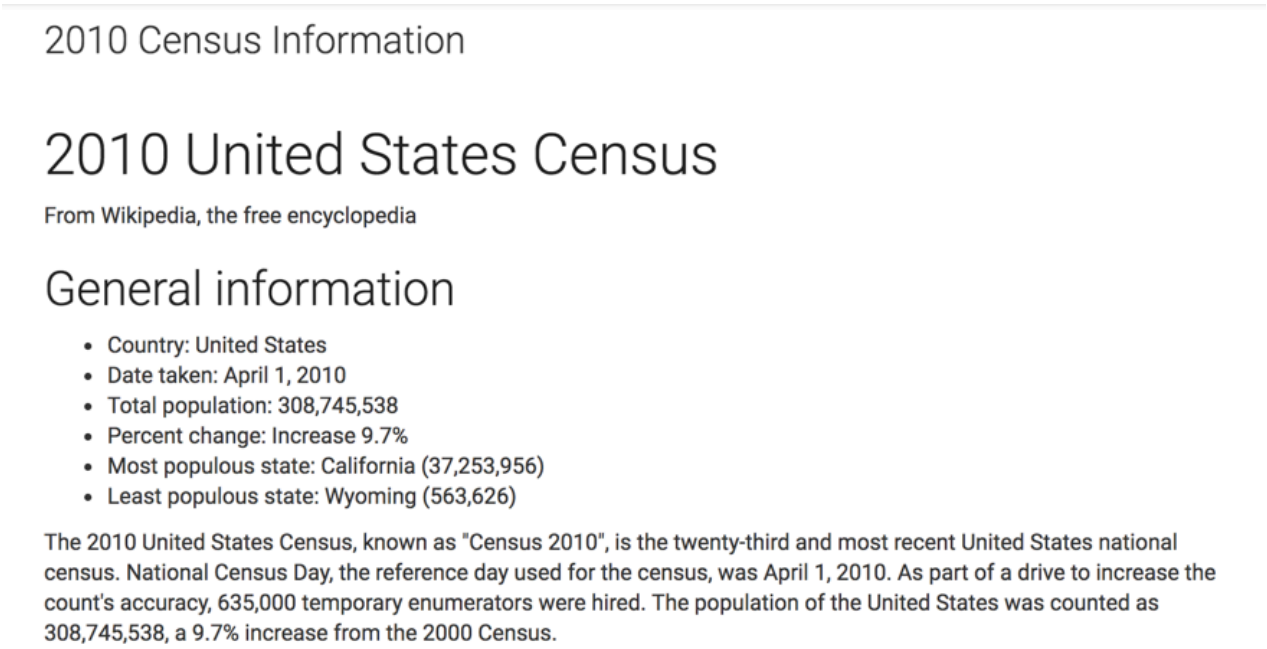

What to do next

For more sophisticated uses of the Extension visual, see *Examples of extensions*.

Related Information

Setting system privileges
Creating a visual

Examples of extensions

See additional examples for Extension visuals.

In the following examples, the definitions for *Print dashboard* and *Download dashboard as PDF* show how to perform these actions through an Extension visual. They have been implemented together, creating a visual that serves as an action panel within the dashboard.

Print dashboard

The following extension definition builds a (print) button with an icon. You can use this button to print the dashboard in View mode.

```
<div id="extension-print-btn">
  <button class="btn btn-xs btn-primary"
  onclick="window.print();return false;">
  <span class="fa fa-print"></span> Print</button>
</div>
```

Download dashboard as PDF

The following extension definition builds a (download PDF) button with an icon. You can use this button to save the dashboard in PDF format from View mode.

```
<div id="extension-pdf-btn">
  <button class="pdf-download btn btn-xs btn-primary">
  <span class="fa fa-download"></span> Download PDF</button>
  <div class="image-dropdown-submenu"/>
</div>
```

Print dashboard and download PDF

The following definition creates two buttons in an Extension visual:

- Print button with an icon
- Download PDF button with an icon

```
<div id="extension-print-and-pdf-btn">
  <button class="btn btn-xs btn-primary"
  onclick="window.print();return false;">
  <span class="fa fa-print"></span> Print</button>

  <button class="pdf-download btn btn-xs btn-primary">
  <span class="fa fa-download"></span> Download PDF</button>
  <div class="image-dropdown-submenu"/>
</div>
```

When you add this Extension visual to a dashboard, the dashboard will display print and download buttons:

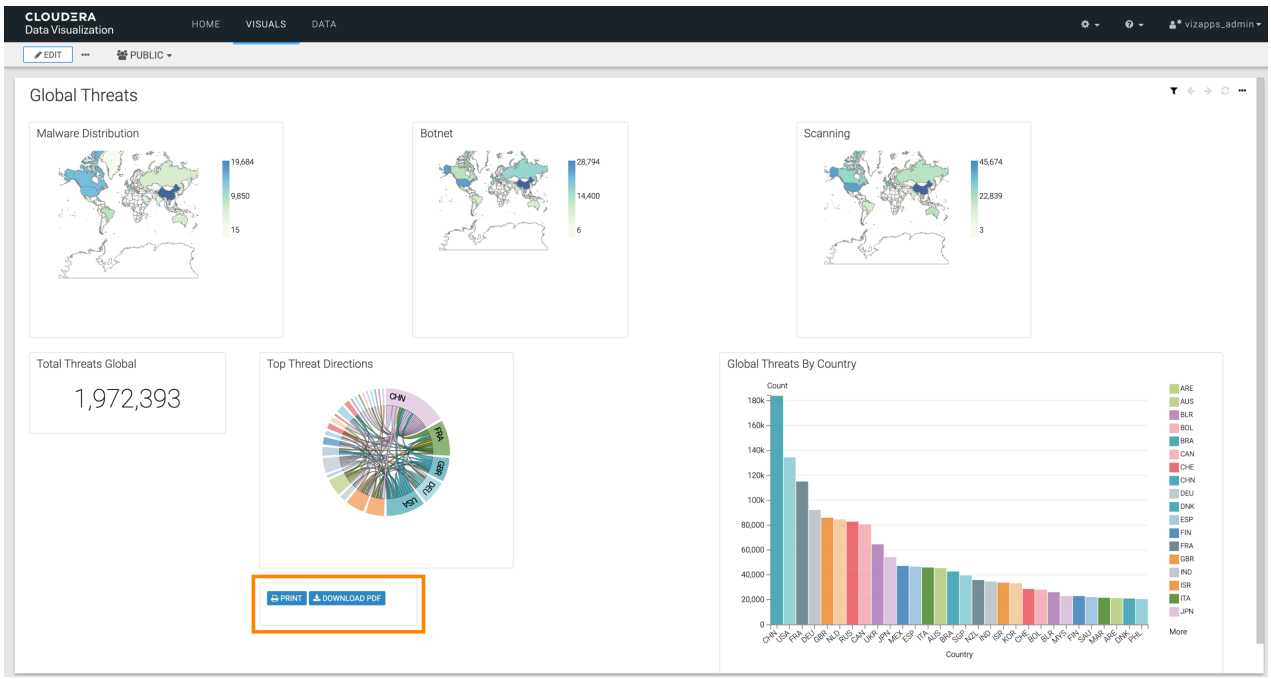

Rich texts

Cloudera Data Visualization enables you to create Rich Text visuals.

About this task

You can use markup code to add supporting textual or visual information to your dashboard. You can insert styled paragraphs, hyperlinks, tables, images, and other elements. These insterted elements do not use any information from the dataset.

Note: You must have system-level Manage styles and settings privilege to create Rich text visuals. For more information, see *Setting system privileges*.

The following steps demonstrate how to create a visual with rich text formatting.

Procedure

1. Start a new visual based on the World Life Expectancy dataset.

   For instructions, see *Creating a visual*.

2. Select Rich Text in the VISUALS menu.

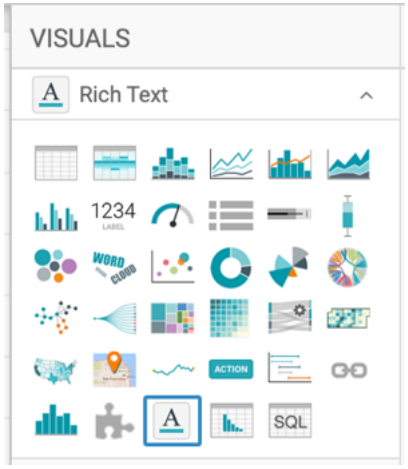

The shelves of the have visual changed. This visual type does not have any shelves associated with it. Instead, it has a single text box for entering and editing text or markup.

3. Click the (expand) icon to open the Rich Text Editor.

The editor modal window contains a large text box and an editing tool bar.

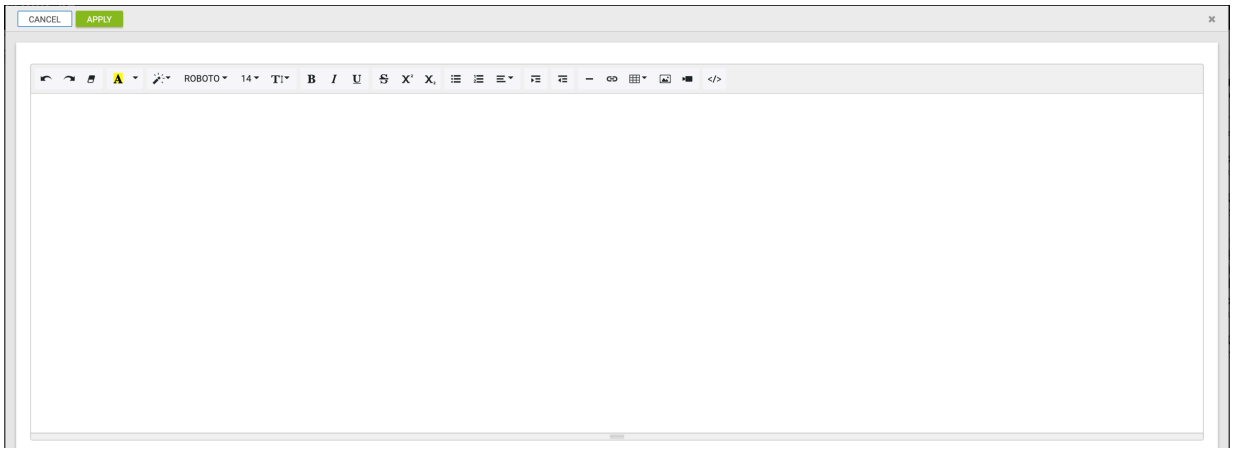

4. Paste the following text in the text box.

```
In the early 1800's, life expectancy at birth began to increase signific
antly in the countries that industrialized early, but it remained low in
 the rest of the world. Industrialization levels correlated strongly wit
h life expectancy, and lead to notable inequalities in global health dis
tribution. Good health was prevalent in the rich countries, where improv
ements in sanitation, infant/maternal care, and food quantity lead to lo
nger life. Meanwhile, poor health persisted in poorer countries.

This global inequality decreased over the last half of the 20th century. C
ountries that relatively recently suffered from bad health are catching up
 rapidly. Since 1900, the global average life expectancy more than doubled
, increasing from 32.6 years for all reporting countries to 70.0 years.
```

5. You can format the text that you have added to the visual. Enhance the basic text according to your requirements.

   Formatting controls include the following options (from left to right on the editing toolbar):

   - Undo and Redo
   - Remove Font Style
   - Recent Color and More Color selector for Background Color and Text Color
   - Style selector for Normal, Blockquote, Code, Header 1 - Header 6, Font Family menu, Font Size menu, and Line Height menu
   - Character formatting: Bold, Italic, Underline, Strikethrough, Superscript, and Subscript
   - Paragraph formatting: Unordered list, Ordered list, and Paragraph alignment: left, center, right, justified, outdent, and indent
   - Indentation options: Indent and Outdent

   Embedding controls include the following options:

   - Horizontal Rule
   - Link
   - Table
   - Picture
   - Video
6. You can also edit your visual in Code view.

   Rich text visuals support the Code View option, as a simple toggle between Text view and Code view.

   For example, you can override the width settings on a previously inserted image. Editing in Code view enables you to specify the width in terms of percentages:

   ```
   <img src="/arc/reports/staticasset/raw/5.png" style="width: 50%;">
   ```

   or in terms of pixels:

   ```
   <img src="/arc/reports/staticasset/raw/5.png" style="width: 350px;">
   ```

   You can also remove table borders or format custom email notification templates.
7. Click APPLY to save your changes and return to the dashboard.

   If you do not want to apply your changes, click CANCEL or just close the modal window.
8. Change the title to World Population - Rich Text.

   [Optional] You can also add a brief description of the visual as a subtitle below the title of the visualization.
9. Click SAVE at the top left corner of the Dashboard Designer.

Results

In this example, the following changes have been implemented: two tables with images in the left column and text in the right column, 2 levels of headings, and links for sources. Your visual should look something like this:

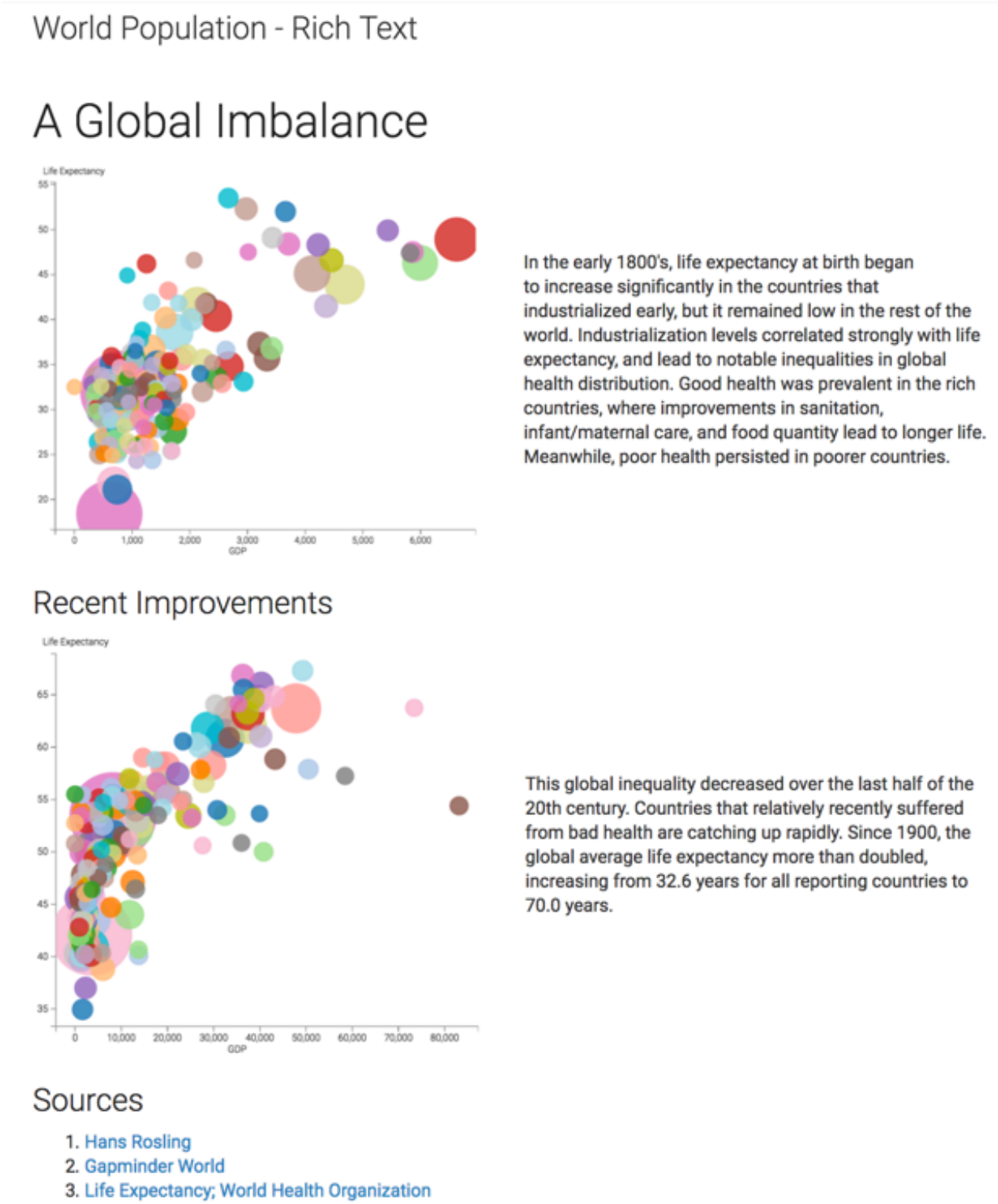

Related Information

Setting system privileges

Creating a visual

Inserting URL links

Cloudera Data Visualization enables you to insert URL links in Rich text visuals.

Procedure

1. Click the Link icon in the Rich Text Editor.

This opens the Insert Link modal window.

2. Make the following specifications:
   a) In the Text to display field, add the display text.
      In this example, 'Gapminder World' has been used as display text.
   b) In the To what URL should this link go? field, add the URL address.
      In this example, https://www.gapminder.org has been used.
   c) Select the Open in new window option.

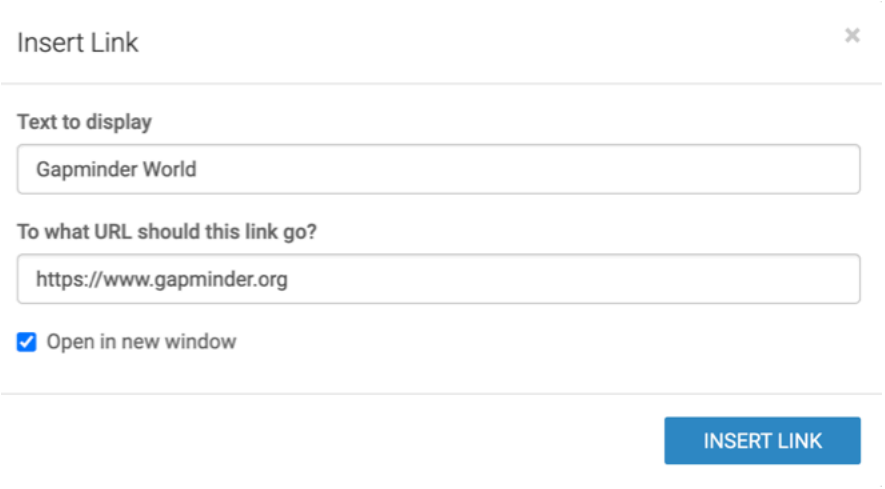

3. Click INSERT LINK to add the link to your visual.

Inserting tables

Cloudera Data Visualization enables you to insert tables in Rich text visuals.

In the Rich-Text Editor, click the Table icon and select the number of rows/columns for the new table.

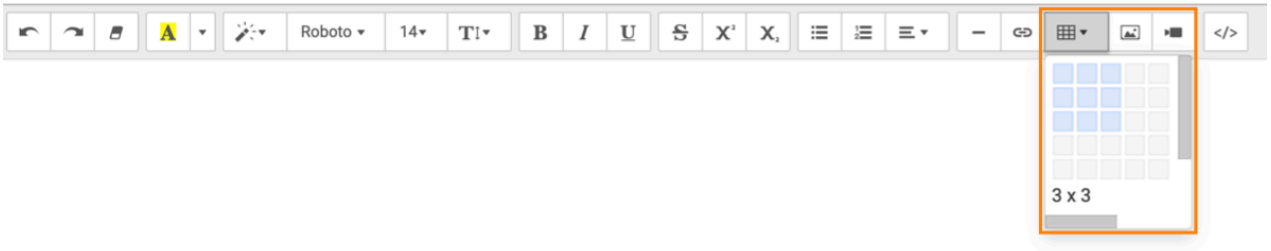

The table is automatically added to your visual.

Inserting images from local machine

Cloudera Data Visualization enables you to insert an image into a Rich text visual from a local directory.

Procedure

1. Click the Picture icon in the Rich Text Editor.

This opens the Insert Image modal window.

2. Under Select from files, click Choose Files and select an image file from your file system.

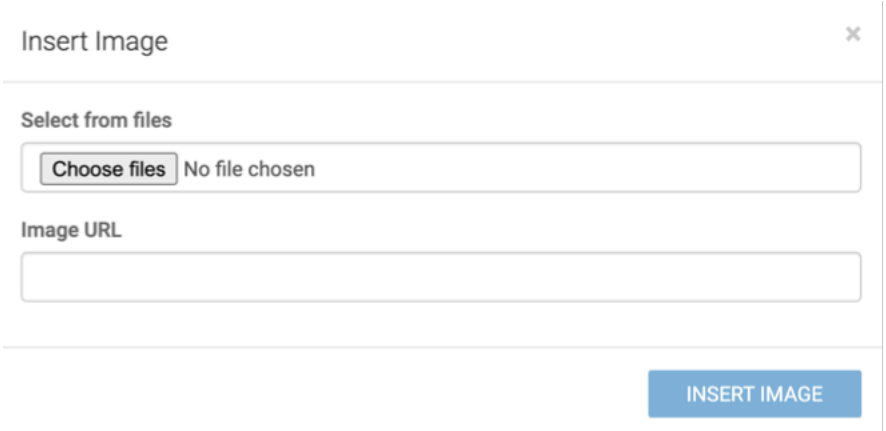

3. Click INSERT IMAGE to add the picture to your visual.

Note: Cloudera Data Visualization automatically converts the image into a static asset and adds it to the set of static assets it manages.

If you click the Code icon, it shows a link to the image inside the staticasset database.

Example

In this example, the link appears as asset_id.image_type, where asset_id is assigned automatically based on the order of upload, and image_type is consistent with the format of the uploaded file, such as png.

```
<img src="/arc/reports/staticasset/raw/4.png" style="width: 223px;">
```

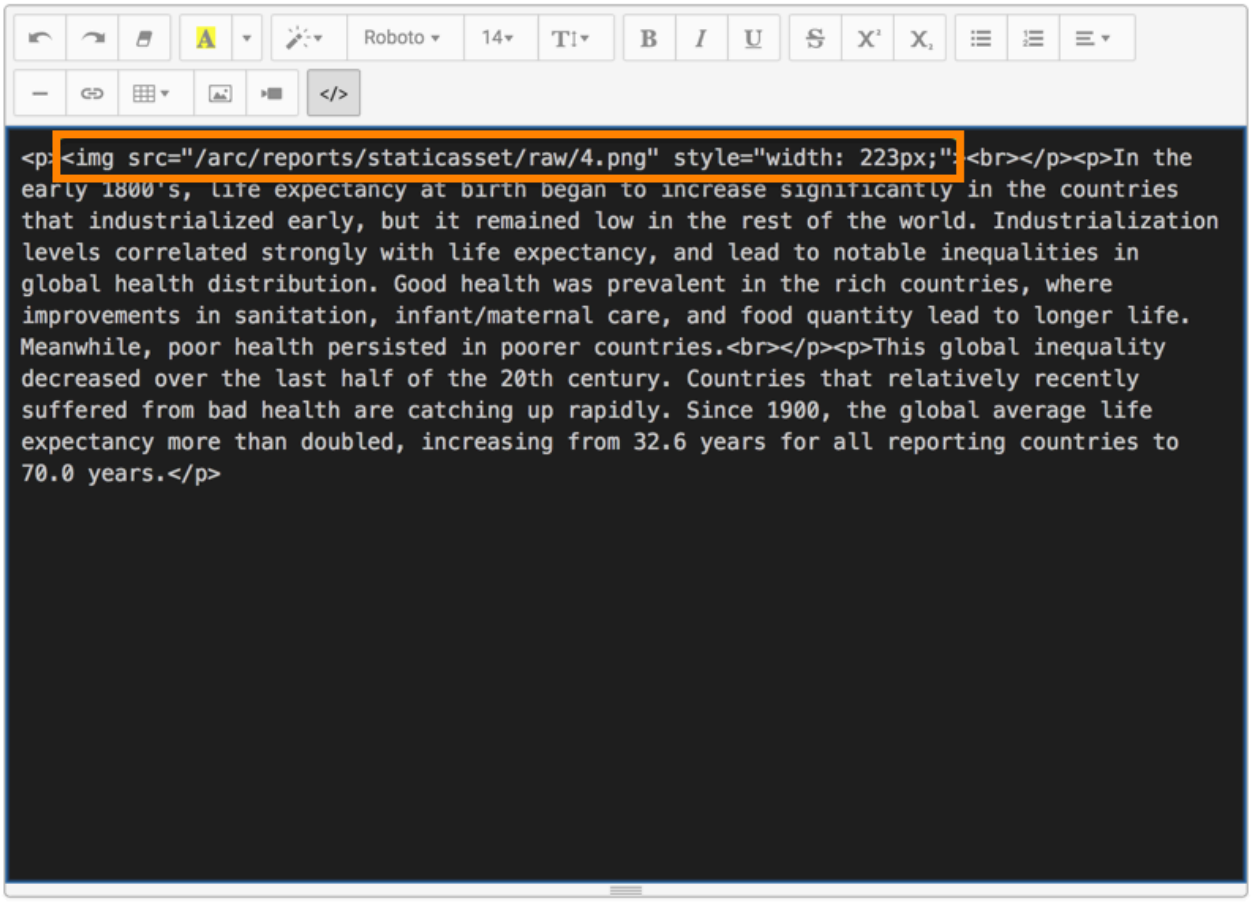

Inserting images from static assets

Cloudera Data Visualization enables you to insert an image that has been previously uploaded to the system as a static asset.

Procedure

1. Click the Picture icon in the Rich-Text Editor.

   This opens the Insert Image modal window.
2. Provide the address of the image in the Image URL field.

   The format for inserting existing static assets is the following:

   ```
   ip-address:port-address/arc/reports/staticassets/raw/asset-id.asset-type
   ```

   Where asset_id is the ID of an asset already imported into the system, and asset-type is the file extension.
3. Click INSERT IMAGE.

Example

In this example, asset 1.png is loaded.

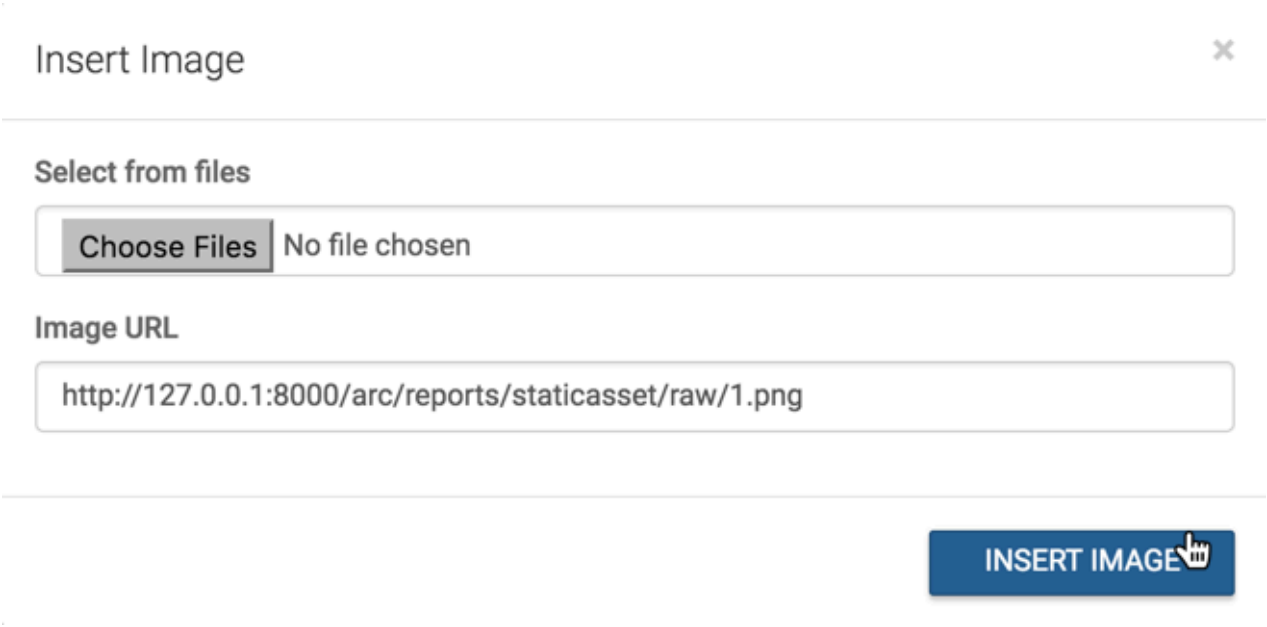

Resizing images

Cloudera Data Visualization enables you to resize an image inside a rich text visual.

Procedure

1. Open the Rich text visual in Edit mode.
2. Click the image and select one of the magnification controls: 100%, 50%, or 25%.
   You also have the option to delete the image from your visual.

Inserting videos

Cloudera Data Visualization enables you to insert videos in Rich text visuals.

Procedure

1. Click the Video icon in the Rich Text Editor.

   This opens the Insert Video modal window.

2. Enter the URL address of the video.

   In this example, the following video is used: https://youtu.be/cv58ODIXdRw.

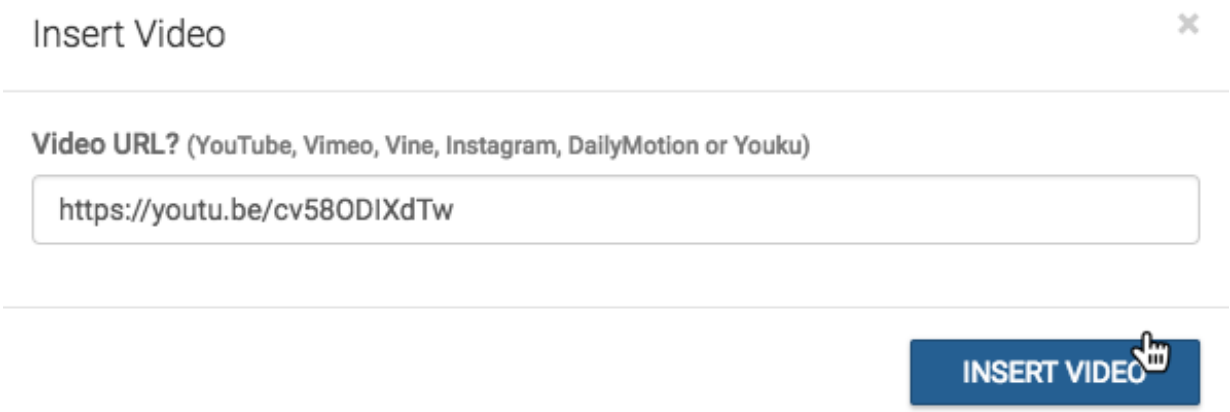

3. Click INSERT VIDEO.

   Cloudera Data Visualization automatically converts the URL into an embedded object. When you examine the visual in Code view, the video appears in an <iframe> element:

```
<iframe frameborder="0" src="//www.youtube.com/embed/cv58ODIXdTw" width=
"640" height="360" class="note-video-clip"></iframe>
```

   In View mode, the visual contains the embedded video object, together with the controls specific to the hosting site.

Field statistics

Cloudera Data Visualization enables you to create Field statistics visuals. Field statistics visuals describe the dataset in statistical terms. They work across all data connections, and on datasets that contain table joins.

Procedure

1. Start a new visual based on the US County Population dataset.

   For instructions, see *Creating a visual*.
2. Select Field Statistics in the VISUALS menu.

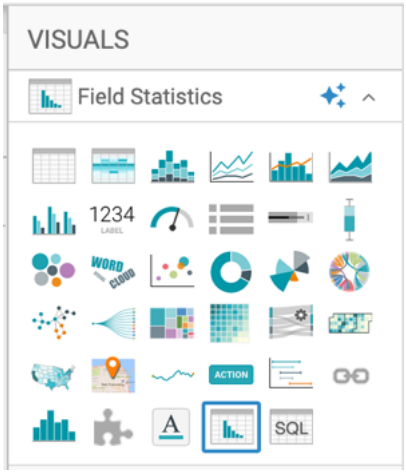

The visual appears as a table that shows the following statistical information for each field in the dataset.

- Actions column contains some widgets for making changes to the appearance of the table.
- Field column indicates the name of the dataset field, and provides an option to include only a limited number of top values, when applicable.
- Type column indicates the data type of the dataset field.
- Distinct Values column reports the number of distinct values for this field. This is typically a good indication of the field's suitability as a dimension.
- Minimum column reports the smallest (lowest) value in the dataset field. For strings, this would be the 'least' in alphabetical sort.

  In this example, for field stname (state name), this would be Alabama, and for field ctyname (county name), this would be Abbeville County.
- Maximum column reports the largest (highest) value in the dataset field. For strings, this would be the 'greatest' in alphabetical sort.

  In this example, for field stname (state name), this would be Wyoming, and for field ctyname (county name), this would be Zieback County.
- Detail column holds a graphic that is either a bar chart of top K values, or a distribution histogram of these values, for each column.

3. You can make changes to the visual through the following actions:
   a) In the Actions column, click the (double up) icon to move the field to the very top of the table.
   b) In the Actions column, click the (up) icon to move the field up one row in the table.
   c) In the Actions column, click the (down) icon to move the field down one row in the table.
   d) In the Actions column, click the (trash) icon to remove the field from the table.
   e) In the Field column, toggle between:
      - Change to top k to change the Detail column to show a bar representation of the top 20 values (default) for the dataset field
      - Change to histogram to change the Detail column to show a distribution histogram for the dataset field

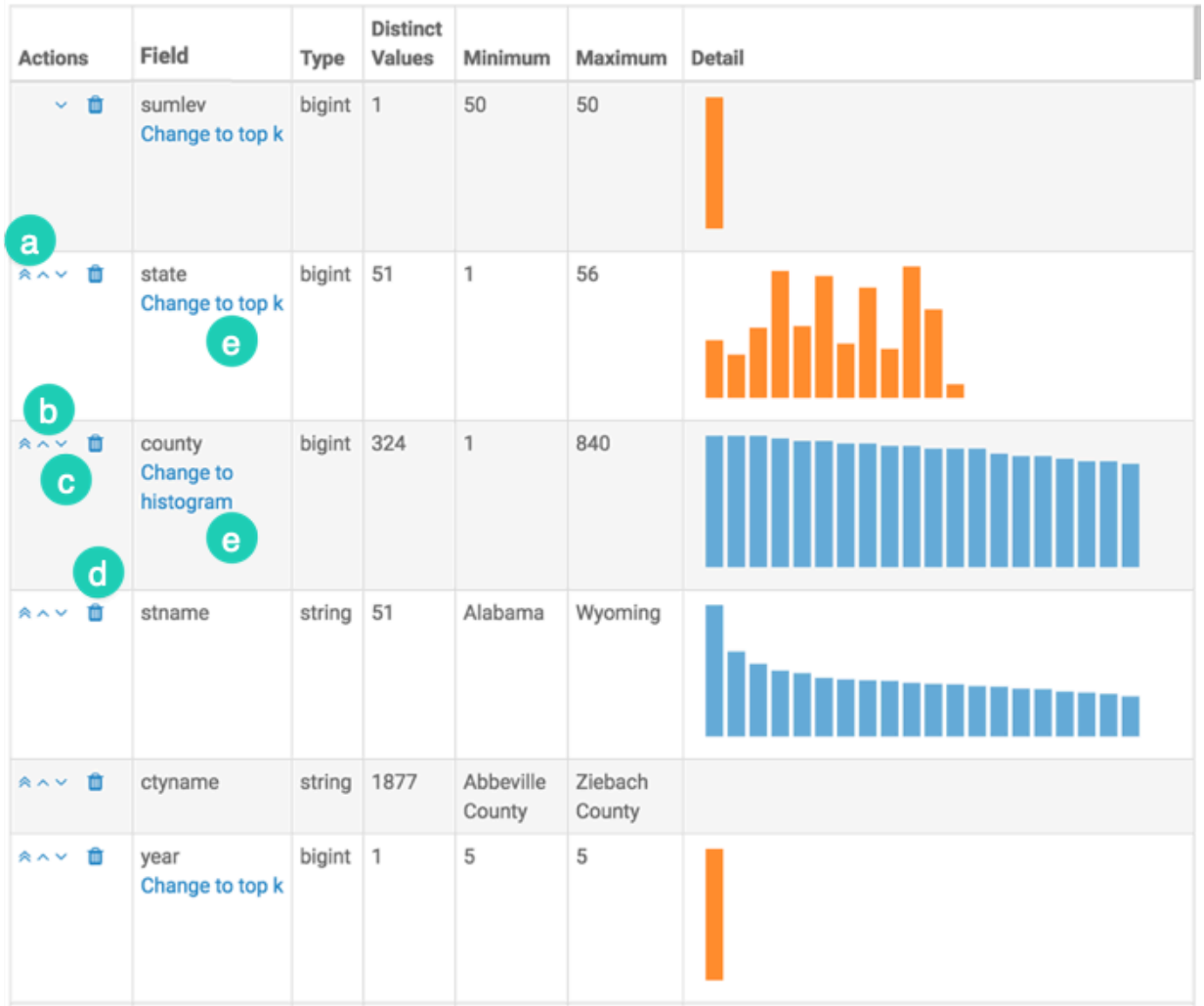

4. Change the title to US County Populations - Field Statistics.
   [Optional] You can also add a brief description of the visual as a subtitle below the title of the visualization.
5. Click SAVE at the top left corner of the Dashboard Designer.

Results

In View mode, the Field statistics visual does not have actions, or it does not contain the Actions column.

Field	Type	Distinct Values	Minimum	Maximum	Detail
sumlev	bigint	1	50	50	
state	bigint	51	1	56	
county	bigint	324	1	840	
stname	string	51	Alabama	Wyoming	Top 20 stname **Indiana** Records **92**
ctyname	string	1877	Abbeville County	Ziebach County	
year	bigint	1	5	5	
agegrp	bigint	1	0	0	

SQL queries

In Cloudera Data Visualization, SQL query visuals enable you to process SQL language queries using an enhanced visual editor for SQL authoring, including autocomplete.

Procedure

1. Start a new visual based on the US County Population dataset.

   For instructions, see *Creating a visual*.

2. Select SQL Queries in the VISUALS menu.

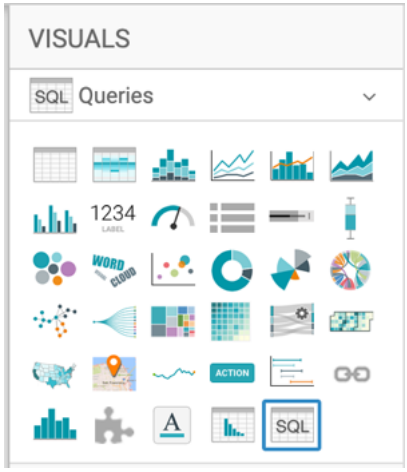

The shelves of the visual changed. You can find a link in the shelf area that leads you to the visual in full-screen mode, where you can view/edit the SQL statement in the SQL editor.

3. Enter the following text in the Enter/Edit SQL Expression modal window:

```
SELECT
    TA_0.`stname` as `State`,
    TA_0.`ctyname` as `County`,
    (sum((TA_0.`wa_male`))+sum((TA_0.`wa_female`))) as `White`,
    (sum((TA_0.`ba_male`))+sum((TA_0.`ba_female`))) as `Black or African
 American`,
    (sum((TA_0.`ia_male`))+sum((TA_0.`ia_female`))) as `American Indian an
d Alaska Native`,
    (sum((TA_0.`aa_male`))+sum((TA_0.`aa_female`))) as `Asian`,
    (sum((TA_0.`na_male`))+sum((TA_0.`na_female`))) as `Native Hawaiian
and Other Pacific Islander`,
    (sum((TA_0.`tom_male`))+sum((TA_0.`tom_female`))) as `Two or More Race
s`,
    sum((TA_0.`tot_pop`)) as `Total Population`
  FROM `samples`.`us_counties` TA_0
  GROUP BY 1, 2
  LIMIT 5000
```

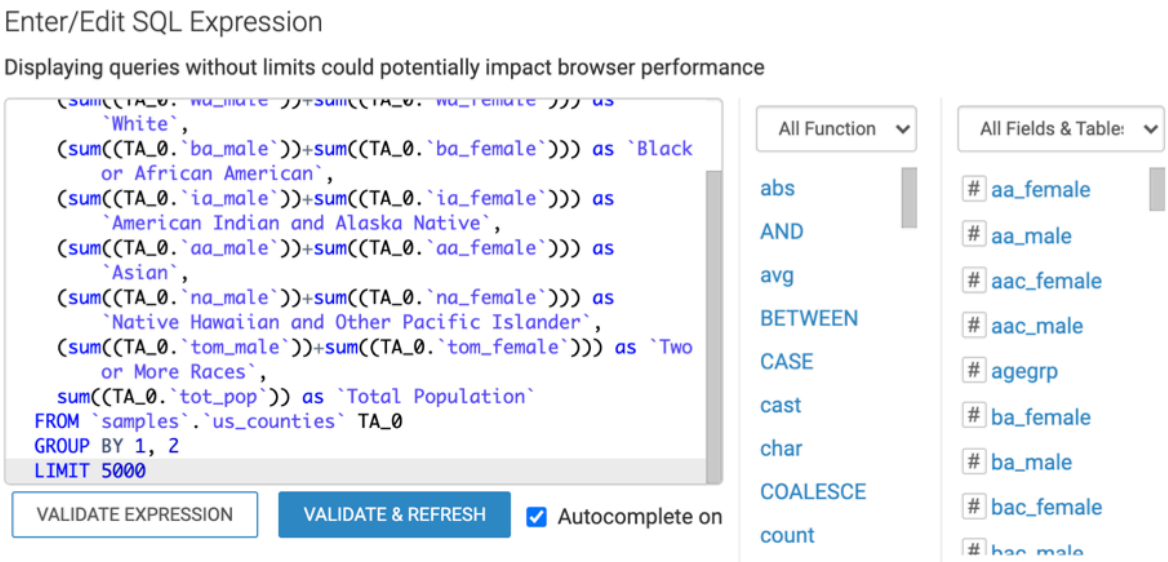

4. Click VALIDATE & REFRESH.

   The visual is updated in the preview area. You can make further edits to the SQL code and check the changes by validating and refreshing the expression again.

   If you do not need the autocomplete function, uncheck the Autocomplete on option.
5. Click APPLY to save the changes.

   If you do not want to apply your changes, click CANCEL or just close the modal window.
6. Change the title to County Populations, by State /County - Query.

   [Optional] You can also add a brief description of the visual below the title of the visualization.
7. Click SAVE at the top left corner of the Dashboard Designer.

Results

The table created from the query should look similar to this:

County Populations, by State /County - Query

State	County	White	Black or African American	American Indian and Alaska Native	Asian	Native Hawaiian and Other Pacific Islander	Two or More Races	Total Population
Alabama	Autauga County	43,568	10,211	289	566	48	832	55,514
Alabama	Baldwin County	166,616	18,283	1,399	1,548	125	2,819	190,790
Alabama	Barbour County	13,735	12,884	181	122	50	229	27,201
Alabama	Bibb County	17,229	5,030	82	31	30	195	22,597
Alabama	Blount County	55,627	994	348	155	64	638	57,826
Alabama	Bullock County	2,848	7,304	98	22	89	113	10,474
Alabama	Butler County	11,053	8,843	71	195	8	137	20,307
Alabama	Calhoun County	89,177	24,597	574	950	111	1,887	117,296
Alabama	Chambers County	20,076	13,319	72	194	22	381	34,064
Alabama	Cherokee County	24,239	1,207	135	66	8	366	26,021
Alabama	Chilton County	38,422	4,451	219	171	83	473	43,819
Alabama	Choctaw County	7,625	5,887	23	33	0	65	13,633
Alabama	Clarke County	13,662	11,117	111	86	4	181	25,161
Alabama	Clay County	11,166	1,956	72	29	1	211	13,435
Alabama	Cleburne County	14,033	546	55	26	10	162	14,832
Alabama	Coffee County	39,371	8,947	736	709	208	1,281	51,252
Alabama	Colbert County	44,096	8,920	300	273	23	834	54,446
Alabama	Conecuh County	6,741	5,999	47	18	3	173	12,981
Alabama	Coosa County	7,358	3,378	66	19	17	128	10,966
Alabama	Covington County	32,128	4,828	239	196	1	563	37,955

< 1 2 3 ... 158 >

As an example of using this visual, sort on the column Native Hawaiian and Pacific Islander to see where most people of that ethnicity live. You can notice that there is a very large population in Los Angeles, San Diego, and in the Salt Lake area.

County Populations, by State /County - Query

State	County	White	Black or African American	American Indian and Alaska Native	Asian	Native Hawaiian and Other Pacific Islander	Two or More Races	Total Population
Hawaii	Honolulu County	219,187	26,894	3,271	422,319	93,362	211,339	976,372
California	Los Angeles County	7,132,409	923,555	149,382	1,440,604	37,507	279,332	9,962,789
Hawaii	Hawaii County	64,697	1,465	1,089	42,527	23,682	55,731	189,191
California	San Diego County	2,436,930	177,261	42,434	369,491	18,898	132,049	3,177,063
Utah	Salt Lake County	947,370	20,445	13,746	38,414	17,312	26,555	1,063,842
California	Sacramento County	947,022	157,728	22,687	221,624	16,804	84,256	1,450,121
Hawaii	Maui County	56,817	1,182	815	46,075	16,688	36,649	158,226
Washington	King County	1,431,673	131,316	20,944	311,876	16,371	95,260	2,007,440
Nevada	Clark County	1,464,211	227,061	23,540	188,017	16,260	81,670	2,000,759
California	Alameda County	813,510	198,719	18,177	429,695	15,030	79,589	1,554,720
California	Orange County	2,300,708	63,299	33,746	582,911	11,916	97,552	3,090,132
California	San Mateo County	471,664	22,949	6,547	194,917	11,796	31,438	739,311
Washington	Pierce County	623,549	58,458	13,326	51,358	11,488	53,502	811,681
Arizona	Maricopa County	3,351,239	218,894	104,816	154,297	10,681	102,242	3,942,169
California	Riverside County	1,833,979	158,919	42,432	149,237	9,517	74,699	2,268,783
California	San Bernardino County	1,615,386	200,072	42,224	145,245	9,476	68,910	2,081,313
California	Santa Clara County	1,058,855	53,405	25,396	618,537	8,927	72,384	1,837,504
Alaska	Anchorage Municipality	199,967	18,441	24,300	25,918	6,670	23,314	298,610
Hawaii	Kauai County	22,818	364	332	21,671	6,336	16,913	68,434
California	Contra Costa County	736,866	104,138	10,680	168,815	6,163	52,935	1,079,597

< 1 2 3 ... 158 >

Cloudera Data Visualization 7.2.9

Workspaces

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a workspace.....4

Editing a workspace.....6

Deleting a workspace.....7

Creating a workspace

In addition to the standard public workspace that is shared by all users, and the single private workspace that each user has, you can create custom workspaces in Cloudera Data Visualization. Custom workspaces can provide you a designated area to collaborate and organize visual elements. Workspaces can be tailored to specific user groups or business lines for efficient dashboard development and viewing.

About this task

Workspaces may be shared with specific users and user groups to develop and view dashboards that are inherently useful to their line of business. Within each of these workspaces, each user has a defined access level: View Only, Edit, or Manage.

Only users with Create Workspace system-level privilege can perform this task.

Note: Administrators have full privileges. Users without a defined access level cannot see the workspace or any of its artifacts.

For information about the different types of workspaces, see *Workspaces in Cloudera Data Visualization*.

Procedure

1. On the main navigation bar, click VISUALS.
2. On the left-side menu, click the Plus icon next to the Search Workspace box.

   Note: This icon is only available to users who have the Create Workspace privilege.

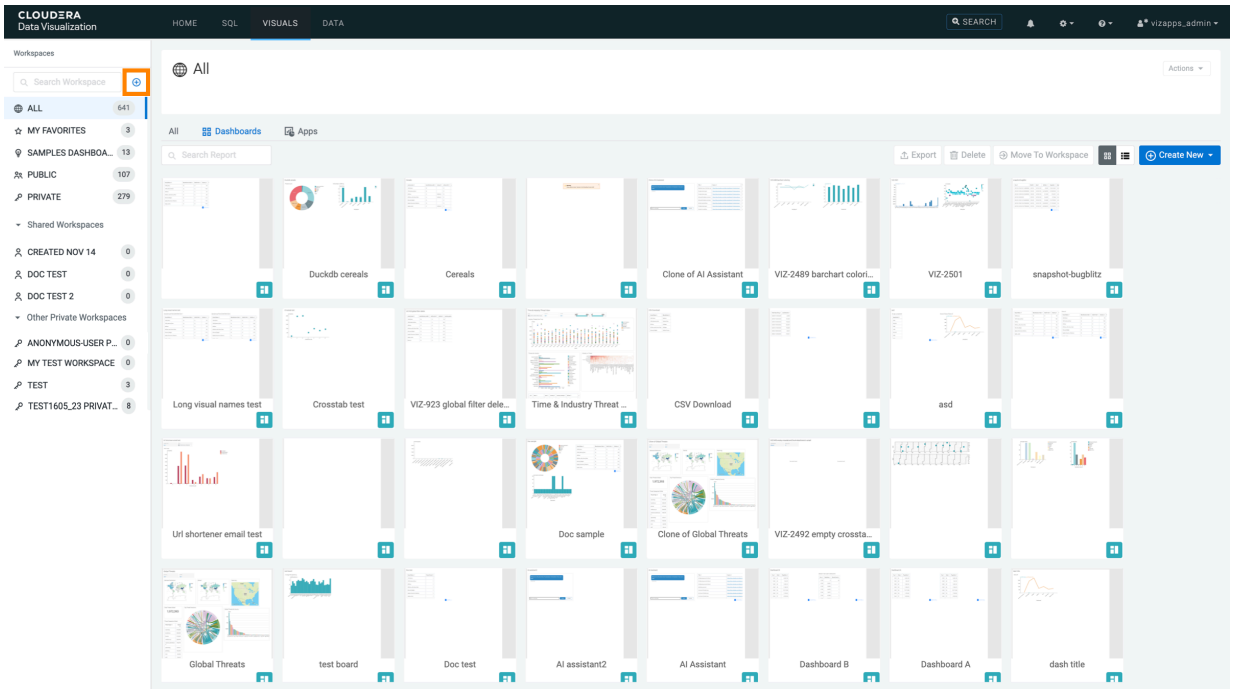

3. In the Add Workspace modal window, enter the name of the new workspace.

   Optionally, you can also provide a short description for the workspace.

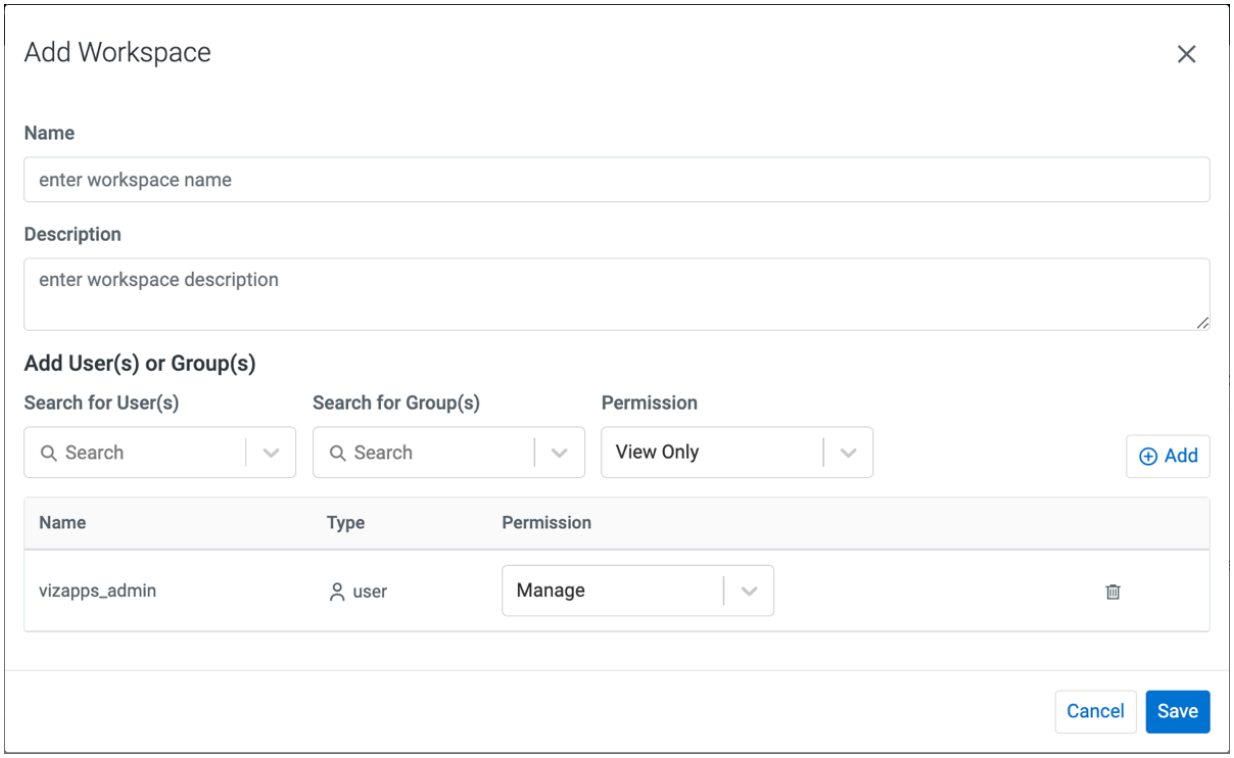

4. You can add users or user groups to the workspace.

   Note:

   If a user has a permission either through their username and group membership, or through multiple groups, that user has the broadest possible permissions. For example, if the user is part of "visual viewer" group and "dashboard editor" group (which has manage permission), that user has the manage permission.

   As you enter a user or group name, the autocomplete functionality makes it easier to find an existing user or group.

5. Click Add.
6. Choose the access level for each user or group: View Only, Edit, or Manage.
   - View Only

     Users can see the contents of the workspace in View mode but cannot make changes.
   - Edit

     Users can move dashboards in and out of the workspace. With adequate dataset permissions, they can also edit, delete, or clone any visuals, dashboards, or apps within this workspace.
   - Manage

     Users can change the name of the workspace, delete it when it is empty, add and remove users or user groups. This also includes all Edit privileges.

   Note: The Admin user is part of the access matrix, as the creator of this new workspace. If the creator is removed, and the creator is not an administrator, they lose access to this workspace.

7. Click Add.

   The workspace appears in the list of workspaces. This new workspace does not have any dashboards or visuals assigned to it at this time.

Related Information

Workspaces in Cloudera Data Visualization

Editing a workspace

In Cloudera Data Visualization, you have the flexibility to modify the visibility and accessibility of workspaces. This process allows users to tailor workspace properties based on evolving collaboration needs.

About this task

All users can edit the Public workspace by default. Administrators have the capability to manage the Public workspace and edit permissions for all users, excluding the changing the name and description of the workspace, or deleting it.

For information about the different types of workspaces, see *Workspaces in Cloudera Data Visualization*.

Procedure

1. On the main navigation bar, click VISUALS.
2. Choose the workspace to be deleted from the Workspaces section of the left-side menu.
3. Click Actions Delete Workspace in the top-right corner of the screen.

   The Edit Workspace modal window appears.

   Alternatively, you can hover over the name of the workspace in the left menu until the (pencil) icon appears and click it to open the modal window.

4. Make the required changes to the workspace properties.

   For example, add a new user or a user group. You can also remove users or groups from the access list.

   Note:

   Remember to set the correct access level for the newly added users.

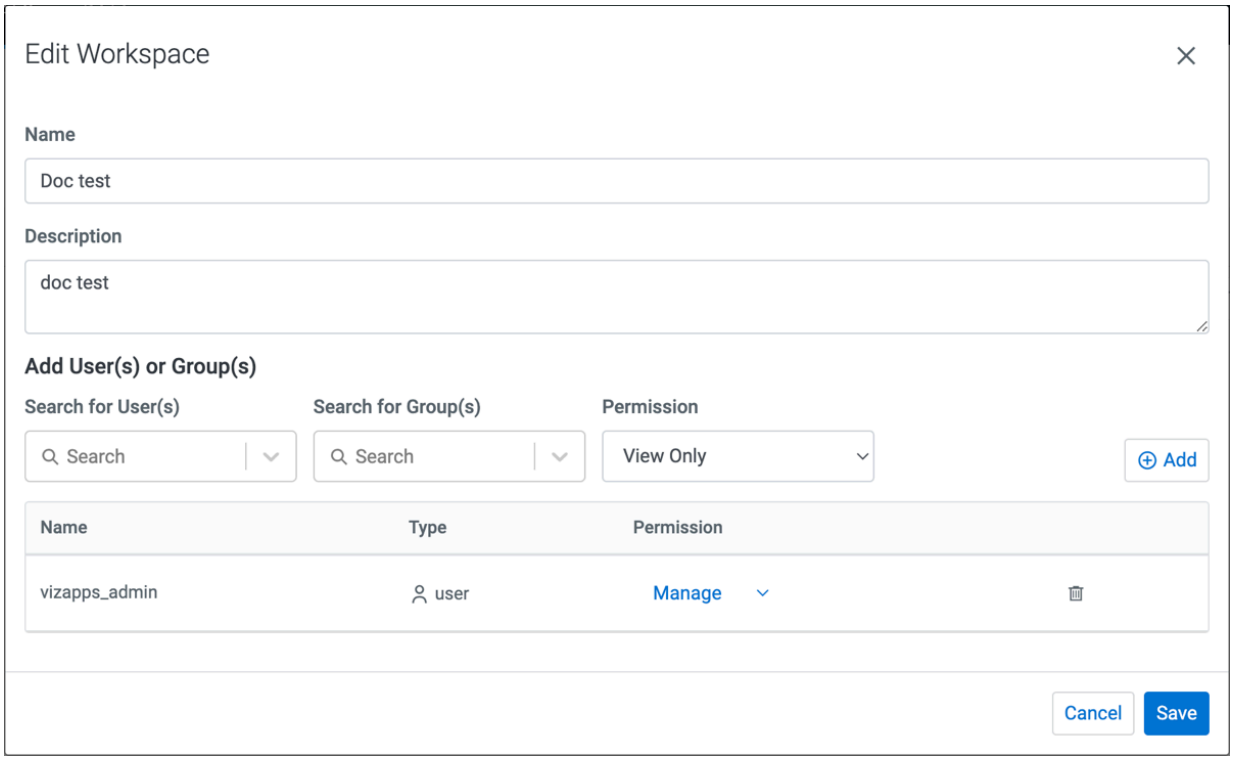

5. Click Save to confirm the changes.

Related Information

Workspaces in Cloudera Data Visualization

Deleting a workspace

You can delete workspaces in Cloudera Data Visualization.

About this task

To delete workspaces, you must have Manage privilege for the workspace.

For information about the different types of workspaces, see *Workspaces in Cloudera Data Visualization*.

Procedure

1. On the main navigation bar, click VISUALS.
2. Choose the workspace to be deleted from the Workspaces section of the left-side menu.
3. Click Actions Delete Workspace in the top-right corner of the screen.

   The Delete Workspace modal window appears.
4. Type DELETE and then click Delete to confirm the action.

Results

The workspace no longer appears in the list of workspaces.

Related Information

Workspaces in Cloudera Data Visualization

Cloudera Data Visualization 7.2.9

Job logs

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Jobs 4

Job runs 5

Scheduled jobs 6

Schedule intervals 7

Job details 8

Jobs

In Cloudera Data Visualization, a job in general refers to an action performed based on predefined parameters. The job outlines the basic details of the task, which can be performed either ad hoc or repeatedly, according to a specified interval or trigger. When a job is initiated, it generates a new running instance, and these runs are recorded in the job log.

Jobs can be classified based on their action type:

Email

The job generates a specific email.

Data Extract

The job performs a data extraction task. Additionally, an email notification can be sent, for example, if the extraction succeeds or fails.

Jobs can run in the following ways:

Adhoc

The action is not pre-scheduled or automated. The job is initiated manually by the user to respond to immediate needs.

Scheduled

The action is performed automatically at specific intervals, following the schedule defined by the schedule interval associated with the job.

Threshold

The action is triggered when specific conditions or thresholds are met. The schedule interval defines how often the system checks for these criteria.

You can review the information available about your jobs on the Jobs interface, which is accessible by clicking Job Status from the Administration menu.

The page has three tabs: Runs (Latest), Scheduled Jobs, and Intervals.

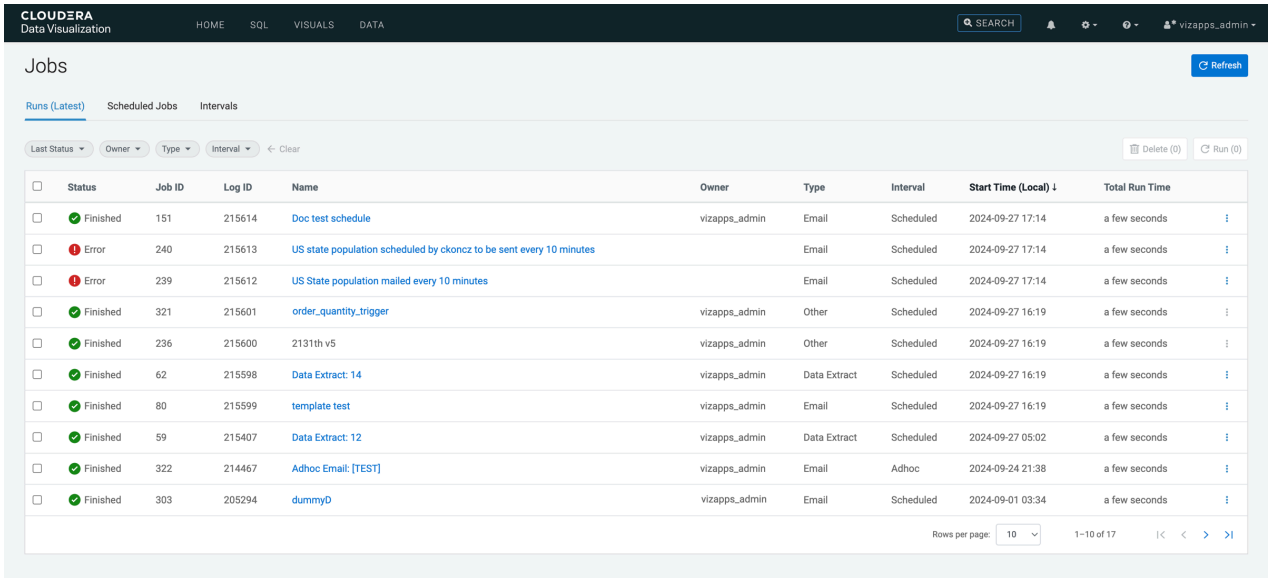

Status	Job ID	Log ID	Name	Owner	Type	Interval	Start Time (Local)	Total Run Time
Finished	151	215614	Doc test schedule	vizapps_admin	Email	Scheduled	2024-09-27 17:14	a few seconds
Error	240	215613	US state population scheduled by ckoncz to be sent every 10 minutes		Email	Scheduled	2024-09-27 17:14	a few seconds
Error	239	215612	US State population mailed every 10 minutes		Email	Scheduled	2024-09-27 17:14	a few seconds
Finished	321	215601	order_quantity_trigger	vizapps_admin	Other	Scheduled	2024-09-27 16:19	a few seconds
Finished	236	215600	2131th v5	vizapps_admin	Other	Scheduled	2024-09-27 16:19	a few seconds
Finished	62	215598	Data Extract: 14	vizapps_admin	Data Extract	Scheduled	2024-09-27 16:19	a few seconds
Finished	80	215599	template test	vizapps_admin	Email	Scheduled	2024-09-27 16:19	a few seconds
Finished	59	215407	Data Extract: 12	vizapps_admin	Data Extract	Scheduled	2024-09-27 05:02	a few seconds
Finished	322	214467	Adhoc Email: [TEST]	vizapps_admin	Email	Adhoc	2024-09-24 21:38	a few seconds
Finished	303	205294	dummyD	vizapps_admin	Email	Scheduled	2024-09-01 03:34	a few seconds

Job runs

Job runs in Cloudera Data Visualization contain essential information about the completed runs, including start time, duration of operation, result, and any error logs encountered during the process.

You can open the Jobs interface by clicking Job Status from the Administration menu. The interface opens on the Runs (Latest) tab by default. This tab provides information about the most recent instance of the latest 100 jobs. For more comprehensive information about a job or to view its additional previous runs, navigate to the Job Details page by clicking the name of the job.

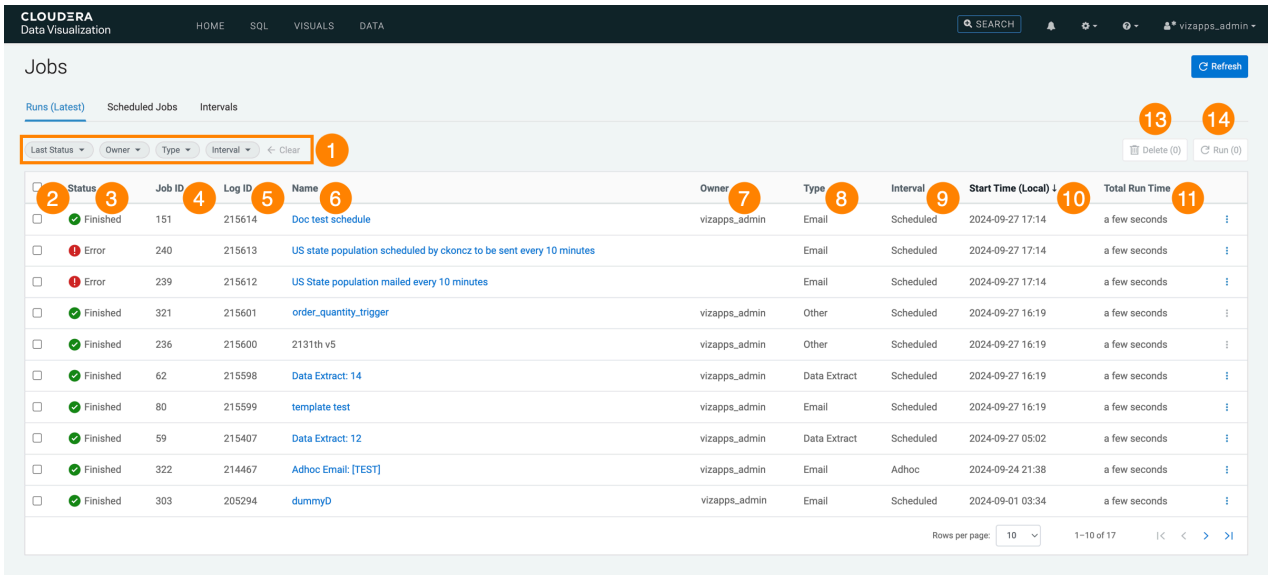

1. Filters are available for job runs based on Last Status, Owner, Type, and Interval. These filters allow for efficient navigation through the logs.
2. Selection for bulk actions: There is a checkbox at the beginning of each job run entry, facilitating bulk deletion or rerun operations. Additionally, the checkbox on the header row allows for selecting all job runs for an action.
3. The Status of a job can be one of the following: Queued, Running, Finished, Error, or Cancelled.
4. Job ID is a unique identifier for each job, shared between the Scheduled Jobs and Runs (Latest) tabs for easy identification and debugging.
5. Log ID is a unique identifier for each job log entry. Unlike the Job ID, which remains the same for every instance of the same job, the Log ID is unique for each run.
6. Name is the name of the job. You can click the name to open the Job Details page. For more information, see *Job details*.
7. Owner identifies the user who created the job. For more information, see *Changing the owner of a job*.
8. Type specifies whether the job target is email or data extract.

   Note: Legacy jobs are classified into two categories: Refresh and Other. These jobs cannot be edited through the UI and do not have a dedicated detail page. While rerunning these jobs is not supported, they remain visible for reference.
9. Interval provides information on whether the job is adhoc (can be initiated manually) or scheduled (running automatically at specified intervals).
10. Start Time is the timestamp indicating when a job commenced, displayed in local time.
11. Total Run Time shows the approximate duration of the job run.

12. Available actions:
    - Details opens a modal window that displays comprehensive information about the specific job run.
    - Run Now initiates a rerun of the job.
    - Delete deletes the the job run.
13. DELETE performs a delete on all selected job runs.
14. RUN reruns all selected jobs.

Related Information

Changing the owner of a job

Scheduled jobs

In Cloudera Data Visualization, a scheduled job is a predefined task or action that is set to occur automatically repeated according to a specified interval.

Scheduled jobs are designed to streamline repetitive processes, such as data extraction or report generation by automating their running according to a predetermined schedule. When creating a scheduled job, you can configure the details of the task, such as its frequency, timing, and parameters.

The Scheduled Jobs tab presents a comprehensive list of jobs that follow a running schedule, and shows the options and actions available for managing all your jobs running according to a schedule.

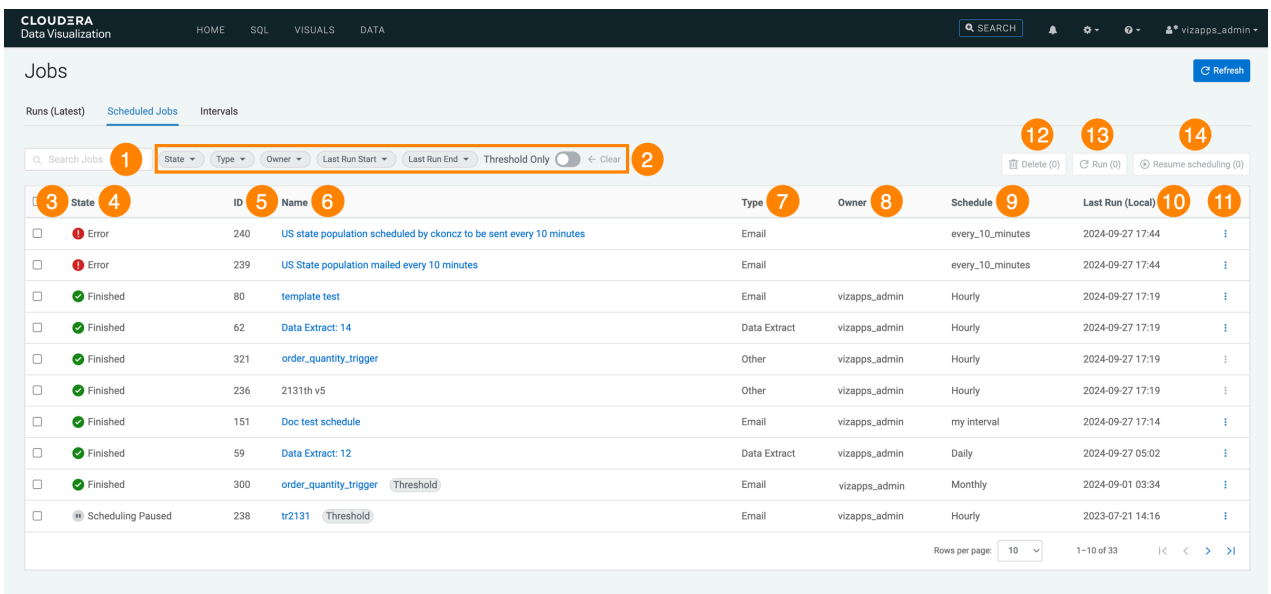

1. Search
2. Filters for scheduled job are based on State, Type, Owner, Last Run Start, Last Run End and Threshold Only.
3. Bulk actions: There is a checkbox at the beginning of each job entry, facilitating bulk deletion, rerun, or resume scheduling. Additionally, the checkbox on the header row allows for selecting all jobs for an action.
4. The State of a job can be one of the following: Running, Finished, Error, Scheduling Paused, Never Run, or Unknown.
5. ID is a unique identifier for each job, shared between the Scheduled Jobs and Job Log tabs for easy debugging.
6. Name is the name of the scheduled job.
7. Type specifies whether the job target is email or data extract.
8. Owner identifies the user who created the scheduled job. For more information, see *Changing the owner of a job*.
9. Schedule indicates the frequency of running the job, either standard types (weekly, hourly) or a custom interval.
10. Last Run is the timestamp of the job's last run displayed in local time.

11. Actions:

- Details open the information module of the job log.
- Run Now initiates a rerun of the job.
- Pause Scheduling suspends a scheduled or triggered job indefinitely, while preserving its definition. This allows you to temporarily stop the scheduling of the job without affecting its underlying configuration. Resume Scheduling beomes available and you can resume the paused job.

  Note: Pausing a job does not stop a running instance. If you want to stop a job in progress, use the cancel function instead.

- Edit opens the Edit Job Details modal window and allows you to edit the details of the scheduled job.

  Note: The Edit action is not available for data extract jobs. Editing data extracts is only possible on the Data Extracts page.

- Change Owner opens the Change Job Owner modal window and allows you to change the owner of the scheduled job.
- Delete rallows you to delete the scheduled job

12. Delete performs a delete on all selected jobs.

13. Run reruns all selected jobs.

14. Resume scheduling restarts all selected jobs that have been paused.

Note: This bulk action only resumes jobs that are paused, and will skip any jobs in the selection that are already active.

Related Information

Changing the owner of a job

Schedule intervals

Intervals in Cloudera Data Visualization refer to the time frames or patterns at which scheduled jobs are set to occur, so these intervals dictate the frequency and timing of the scheduled jobs.

You can specify the schedule intervals on the Intervals tab of the Jobs page. The tab tab provides a comprehensive list of schedule intervals along with their associated features and actions.

Common examples of schedule intervals include hourly, daily, weekly, or monthly occurrences, as well as custom intervals tailored to specific business needs. By defining schedule intervals, you can effectively automate repetitive tasks and ensure that jobs run consistently and on time.

Note: This feature is only available to users with administrative privileges.

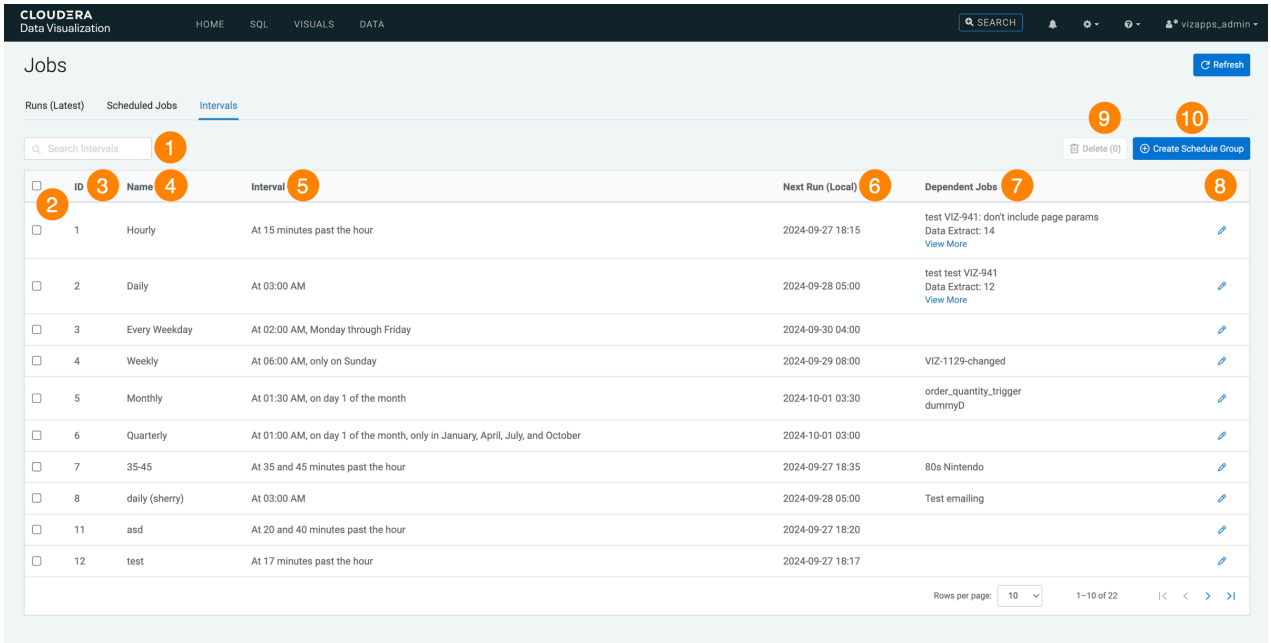

1. Search
2. Bulk actions: There is a checkbox at the beginning of each schedule interval entry, facilitating bulk deletion. Additionally, the checkbox on the header row allows for selecting all job logs for the action.
3. ID is a unique identifier assigned to each schedule interval for internal reference.
4. Name is a descriptive title assigned to the schedule interval for easy identification and organization.
5. Interval describes the parameters of the interval, such as Every day at 3:00, Every hour at 20 minutes past the hour, or Every month on 1 at 1:30.
6. Next Run (local) displays the expected time for the next deployment of the schedule in local time format.
7. Dependent Jobs list any scheduled jobs that use the specified schedule.
8. The icon enables you to change the properties of existing schedules. For more information, see *Changing schedule intervals*.
9. Delete performs a delete on all selected intervals.
10. Create Schedule Group enables you to specify a custom schedule interval. For more information, see *Creating new schedules*.

Related Information

Creating new schedules

Changing schedule intervals

Job details

The Job Details page provides comprehensive insights into a selected job, helping you understand and manage job operations effectively. You can access this page by clicking the name of the job on either the Runs (Latest) or Scheduled Jobs tab.

Job Details

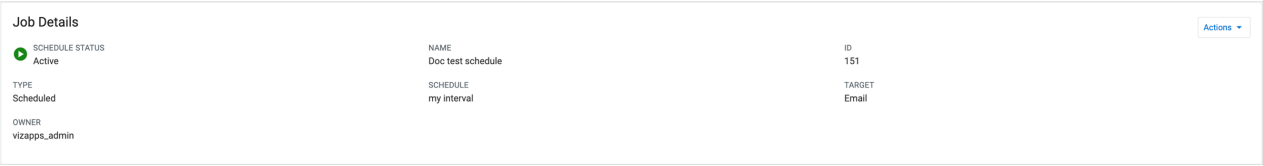

In this section, you can review basic information about the job and its scheduling parameters, as well as perform the following actions:

- Edit opens the Edit Job Schedule Details modal window, allowing you to modify schedule details.

  Note: The Edit action is not available for data extract jobs. For more information on changing data extract details, see *Editing a data extract*.

- Change Owner opens the Change Job Owner modal window, allowing you to assign a new owner to the job.

  Note: When the job owner is changed, the ownership of the related logs is also updated accordingly.

- Delete allows you to remove the job if it is no longer needed.

  Important: Deleting a scheduled job deletes the job itself along with all associated job logs.

- Run Now allows you to start the job immediately.
- Pause Scheduling allows you to suspend a scheduled or triggered job indefinitely while preserving its definition. This action temporarily stops the job scheduling without affecting its underlying configuration. You can resume a paused job.

  Note: Pausing a job does not stop a running instance. If you want to stop a job in progress, use the cancel function instead.

- Resume Scheduling allows you to restart a previously paused scheduled or triggered job.

Depending on the type of the job you are checking, you can also review data extract or email details on this page.

Email Details

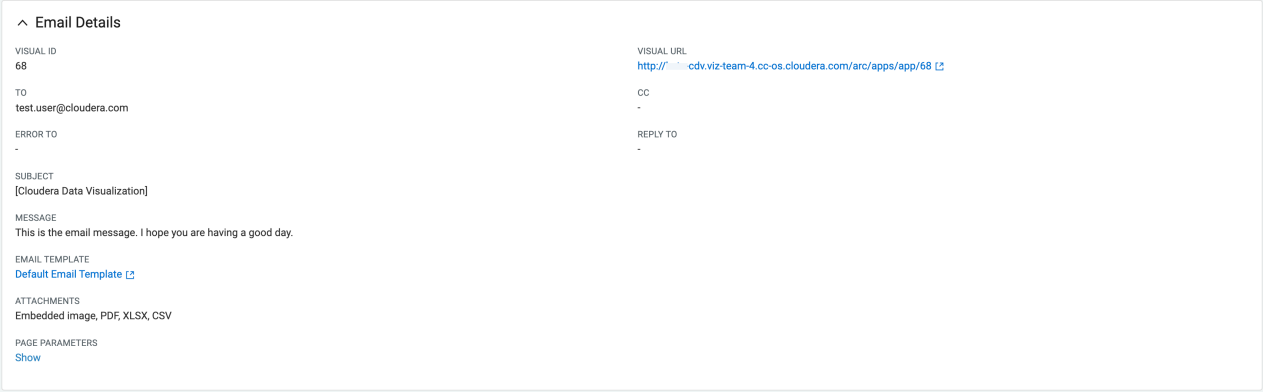

In this section, you can review information about email jobs including subject, message content, email template, attachments, and other relevant details.

Data Extract Details

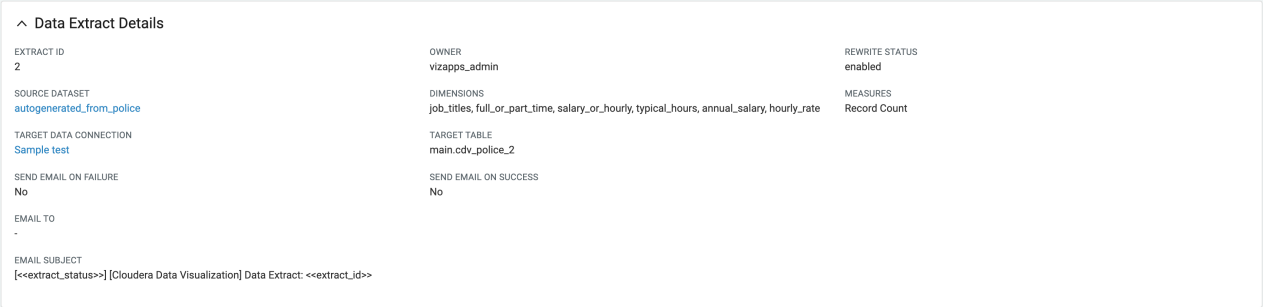

In this section, you can explore specifics like extract ID, source dataset, target data connection, and other details related to the data extraction process.

Previous Runs

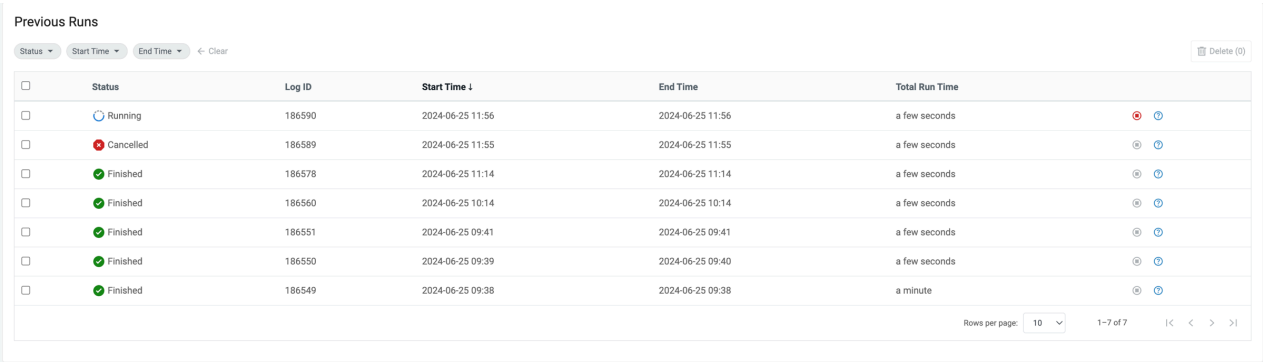

In this section, you can see a comprehensive list of past job operations, which can help in diagnosing and troubleshooting errors.

Filters

You can adjust the list of displayed previous runs using the following filters:

- Status – This filter is based on job status.
- Start Time – This filter is based on the start time of job runs. The selected value will be the minimum start time of the displayed runs.
- End Time – This filter is based on the end time of job runs. The selected value will be the maximum end time of the displayed runs.

Actions

You can perform the following actions:

- allows you to delete logs from the list of previous runs. To select logs for deletion, click the checkbox at the start of each log row. To select all logs, click the checkbox on the left side of the list header.

  Note: This action will only delete the job logs, not the job itself.

- allows you to stop a running job. This option is only active for jobs currently in progress.

- opens the Run Details modal, which includes information about the tasks performed in connection with the job instance and any errors encountered.

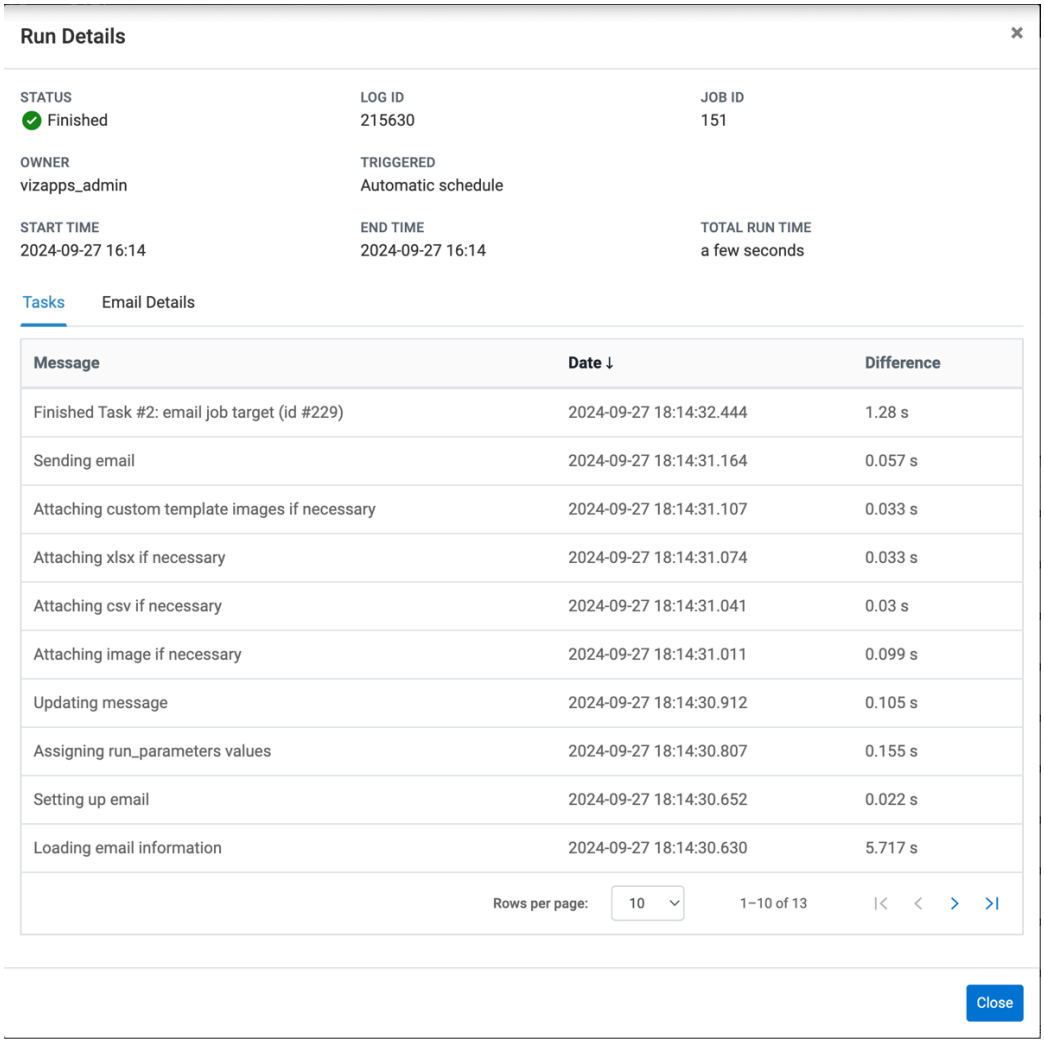

Related Information

Editing a data extract

Cloudera Data Visualization 7.2.9

Visual Artifact Migration

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Exporting and importing visual artifacts.....4

Supported visual artifacts.....5

Data compatibility during visual artifact migration.....5

Exporting and importing visual artifacts

Cloudera Data Visualization allows you to export and import visual artifacts. This migration utility addresses various use cases. For example, in an enterprise environment, it is a common practice to develop and test applications in one environment before making them available in a client-facing production environment. You can also create back-ups of visual artifacts, or you can share your artifacts in various communities.

Cloudera Data Visualization provides an easy UI method to export visual artifacts (visuals, dashboards, and apps, plus their dependencies) by saving them to a JSON file. This file can be used to import the visual artifacts to a new system, where they can be deployed, if the new environment has identical data connections.

Visual artifacts do not store the actual raw data, or results of queries. They contain information about the layout and settings, the associated dataset, style customizations, and the query. When saving artifact information for export, Cloudera Data Visualization also captures all these dependencies. For example, when you export a visual that uses a custom style, the export includes that custom style, as well as the dataset. As a result, when you import the visual into a new system, it looks exactly the same as it did at the source.

Manual export

You can export dashboards, linked visuals, or complete apps. The export generates a JSON file and saves it into your download directory. It can then be shared, and imported into multiple systems.

Note: You need the Manage visuals and dashboards privilege to export visual artifacts.

For more information, see *Exporting a visual*, *Exporting a dashboard*, and *Exporting an application*.

Manual import

You can import visuals and dashboards, or complete apps, using the file generated by the export functionality. This JSON file must be present on the client machine. When you import artifacts, Cloudera Data Visualization recognizes if they have been previously imported. In such cases, you only update the existing artifact instead of creating a new one. This ensures that the artifacts are correctly synchronized between development and production platforms. Cloudera Data Visualization verifies whether the artifacts already exist on the target system. By default, it examines all dashboards and visuals for a UUID match. Selecting the Disregard artifacts that exist in other workspaces option on import limits this search to the target workspace.

The necessary import privileges depend on the operation. These are some examples:

- On initial import, you must have these privileges: Create datasets, Explore tables, Manage visuals and dashboards. If custom styles are part of the backup file, you also need also Manage custom styles.
- If the import is only updating an existing dataset (everything else is already on the system), you only need the Manage dataset privilege.
- If import is only updating visuals and dashboards, you need the Manage visuals and dashboards privilege. If custom styles are part of the backup file, you also need the Manage custom styles privilege.

For more information, see *Importing a visual*, *Importing a dashboard*, and *Importing an application*.

Migration with REST APIs

For scenarios where migrations are regularly occuring tasks, Cloudera Data Visualization provides a set of REST APIs that can be used in scheduled scripts. This approach involves authentications with data API keys. For more information, see *Migrating visual artifacts using REST APIs*.

Related Information

Exporting a visual

Importing a visual

Exporting a dashboard
Importing a dashboard
Exporting an application
Importing an application

Supported visual artifacts

Cloudera Data Visualization allows you to export and import various visual artifacts.

Supported visual artifacts include datasets, linked visuals, dashboards, custom styles, custom colors (palettes), segments, and static assets.

The Export utility automatically exports any existing thumbnails. When you want to import visual artifacts, you have the option of importing the thumbnails into the new installation, generating new thumbnails at import, or skipping thumbnail processing.

Data compatibility during visual artifact migration

The migration utility of Cloudera Data Visualization allows you to export visual artifacts and then import them to a new system. It is important to pay attention to data compatibility between the two systems to make sure that migration works properly.

The destination system (which imports the visual artifacts) has to have a valid connection to the data used in the visual artifacts, and that data must be in a table that matches the name. When you import a visual artifact file with the Check data table compatibility option enabled, the import stops when it cannot find the needed table on the connection.

Some environments use multi-connection architecture for table access. In such cases, you must disable the Check data table compatibility option to successfully complete the import into a single connection. After the import completes, you can re-route the data connections of datasets as necessary.

Cloudera Data Visualization 7.2.9

Monitoring Dashboard Performance

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Showing usage information..... 4

Showing usage information

Cloudera Data Visualization helps you debug your dashboard performance by providing detailed statistics on the rendering and querying operations of the visuals.

Before you begin

You have enabled the following settings in Site Settings Debugging :

- [ ] Verbose Error Reporting. Display Javascript page errors on screen in app view/edit pages
- [x] Enable the "Show Usage Info" and Ctrl-Shift-G shortcut to view page profiles within all pages
- [x] Additionally enable the Ctrl-Shift-G shortcut to view page profiles within dashboard/app view pages

For more information, see *Debugging and Logs* in Site Settings.

Note: The Site Settings menu is only available to users with administrative privileges.

Procedure

1. Open an existing dashboard.

2. In the Dashboard Designer top menu bar, click the (ellipsis) icon and select Show Usage Info. Alternatively, you can use the keyboard shortcut Shift + Control + G to open the modal.

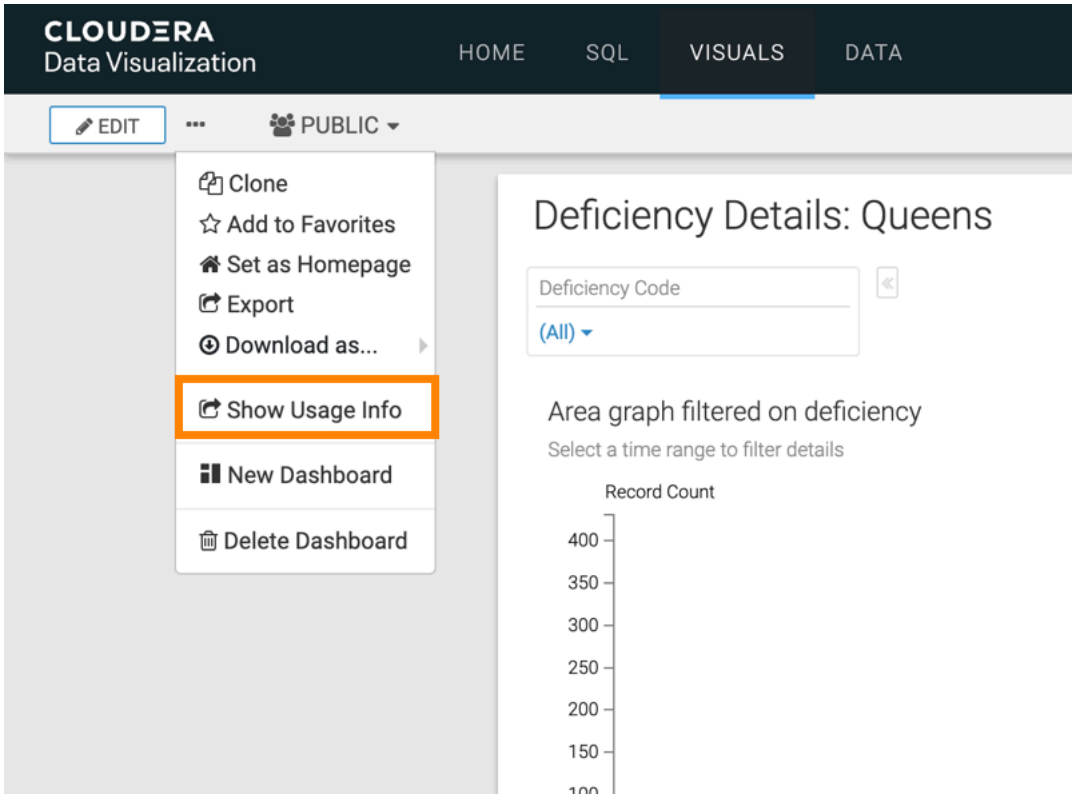

The Profile Data modal window is displayed.

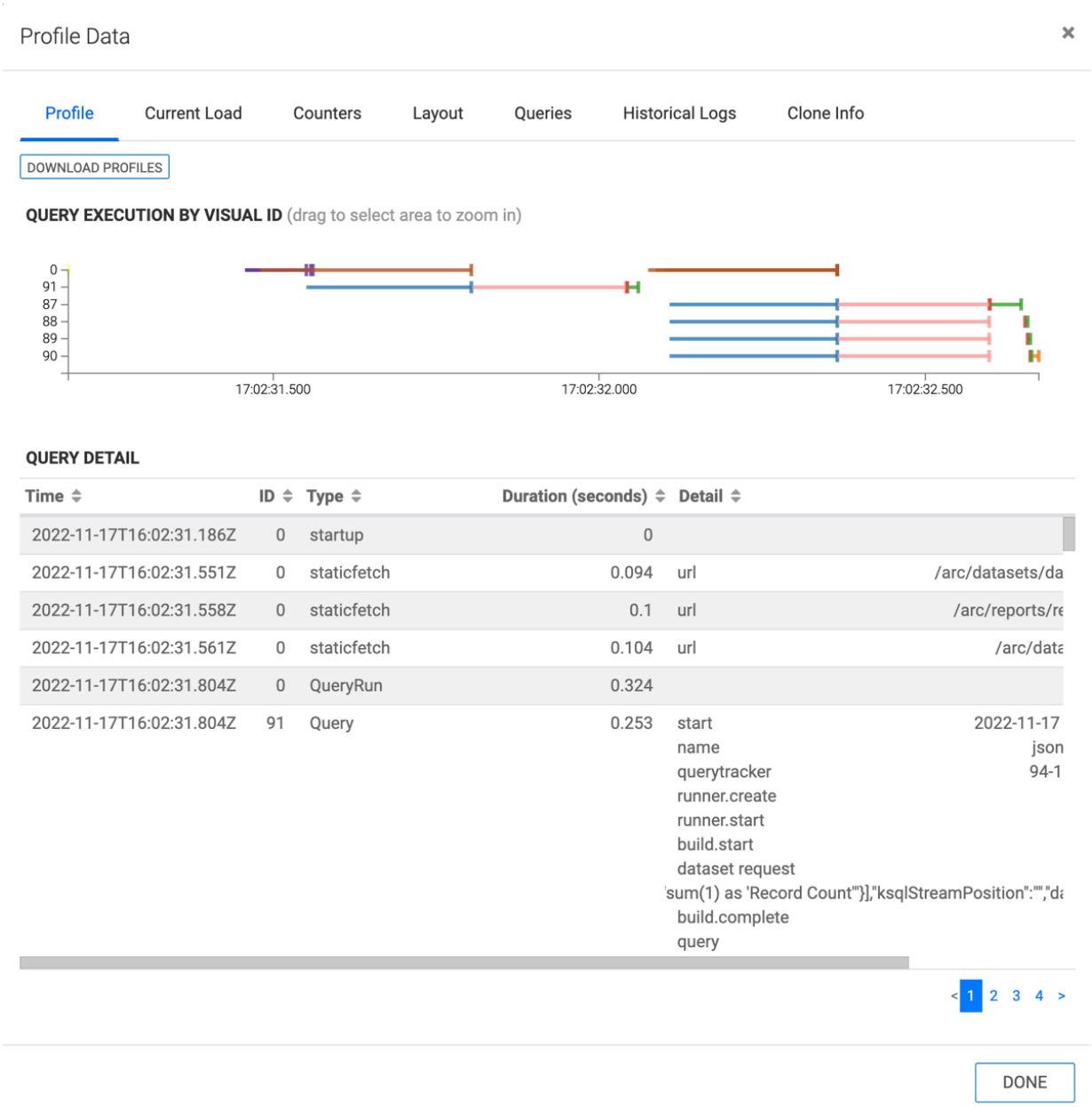
Profile Data
Profile
Current Load
Counters
Layout
Queries
Historical Logs
Clone Info
DOWNLOAD PROFILES
QUERY EXECUTION BY VISUAL ID (drag to select area to zoom in)
0
91
87
88
89
90
17:02:31.500
17:02:32.000
17:02:32.500
QUERY DETAIL
Time
ID
Type
Duration (seconds)
Detail
2022-11-17T16:02:31.186Z 0 startup 0
2022-11-17T16:02:31.551Z 0 staticfetch 0.094 url /arc/datasets/da
2022-11-17T16:02:31.558Z 0 staticfetch 0.1 url /arc/reports/re
2022-11-17T16:02:31.561Z 0 staticfetch 0.104 url /arc/data
2022-11-17T16:02:31.804Z 0 QueryRun 0.324
2022-11-17T16:02:31.804Z 91 Query 0.253 start 2022-11-17
name
querytracker
runner.create
runner.start
build.start
dataset request
sum(1) as 'Record Count'"}],"ksqlStreamPosition":"","da
build.complete
query
1 2 3 4
DONE

3. You can view the following information on the Profile Data interface:

Profile tab

- Download profiles – Click it to download the performance profile of your dashboard as a text file and save it to your local system.
- Query execution by visual ID – You can use this visual to check the timeline of each operation.

  Hover over one of the time segment lines that represent either the visual, or additional tasks. The tooltip shows the Duration, ID, Type, Detail, and Error statistics.

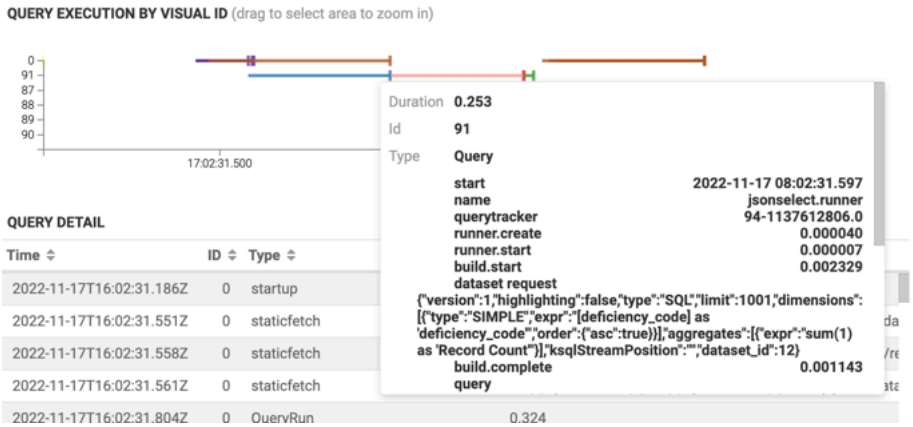

  - The ID of the visual is ‘91’ in the above image. This is the stand-in ID for ancillary tasks Cloudera Data Visualization performs when building a dashboard.
  - You can hover over a line that represents a visual ID (in the above example ‘91’), and view the duration, type, and details of the query performed on that visual.
  - To zoom in on a particular time segment, drag to select the area. You can now drill down to the details of this time selection. Click Clear Time Selection to return to the original view. In the following example 17:02:32.500 is selected.

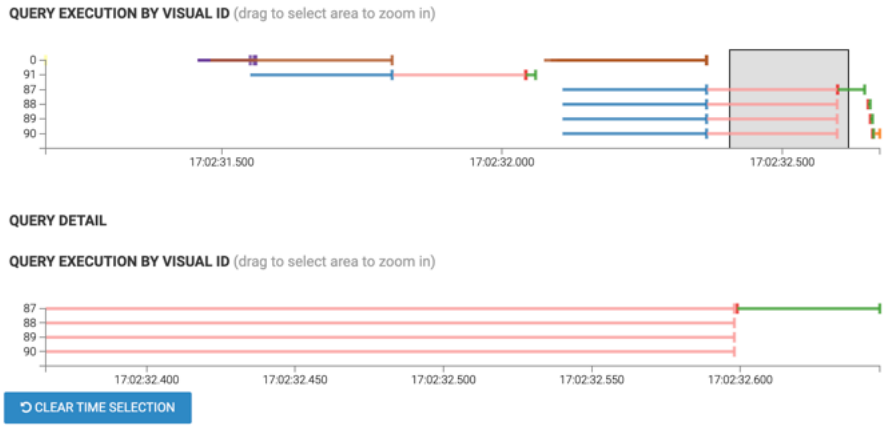

- Query detail – This section shows statistics of all the visuals and queries in a table.
  - Time shows the time when the query operation started.
  - ID shows the ID of the visual.
  - Type shows the query type: startup, set data result, draw visual, or table rows processed.
  - Duration shows the query execution time, in seconds.
  - Detail shows the syntax of the query.
  - Error shows any errors that occurred during query execution.

Current Load tab

You can see a summary of active and pending queries. You can also manage your queries: refresh the queries, cancel all your queries, or download the query data.

Counters tab

- Rendering Engine – The table shows real-time statistics for rendering operations on all visuals of the dashboard:
  - Average Render Time
  - Total Render Time
  - Render Count
  - Maximum Render Time
- Query Engine – The table shows real-time statistics on the query operations:
  - Average Query Time
  - Total Query Time
  - Query Count
  - Maximum Query Time
  - Total Result Rows

Layout tab

- Datasets – It shows you the ID of the datasets used on the dashboard.
- Page layout – The widgets show you how the visuals are arranged on the dashboard and which datasets are used.
- Top left color squares represent datasets.

Queries tab

You can see a list of queries. You can download the query data as a CSV file and save it to your local system.

This section shows the following for each query that was initiated from the dashboard:

- Time shows the time when the query operation started.
- Dataset shows the ID of the dataset.
- Visual shows the ID of the visual.
- Runtime shows the query execution time in seconds.
- Query shows the syntax of the query. Hovering on the information icon will show this data in a formatted manner where it can be copied to clipboard.

Historical Logs tab

Historical Logs show the list view for activities on this dashboard. For more information, see *Activity Log*.

Clone Info tab

Clone Info provides information about the clones of the dashboard. The table on the left shows the ancestor chain, and the table on the right shows the immediate children of the dashboard.

4. Click DONE to exit the performance Profile Data interface and return to your dashboard.

Related Information

Debugging and Logs in Site Settings

Activity Log

Cloudera Data Visualization 7.2.9

Natural Language Search

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Natural language search UI 4

Search results 5
- Order of search results 6

Search bookmarks 6

Search history 8

Search types 8
- Hit highlighting 8
- Match field names 8
- Matching suggested questions 9
- Bookmarked searches 9
- Search history 10
- View more searches 10

Global search 11
- Global search supported keywords 12

Natural language search UI

Cloudera Data Visualization enables you to find relevant information in the dataset through natural language statements, and renders the result in a graphical format. Using this type of search you can type in a sentence, using regular, spoken language, that describes the information you are looking for.

To enable Natural Language Search (NLS) modal globally, see Enabling NLS in Site Settings. For the required configurations, see Enabling dataset for NLS.

Note: To use NLS after enabling search in the Menu Bar and configuring datasets for searching, click SEARCH in the top right corner or the icon next to searchable datasets.

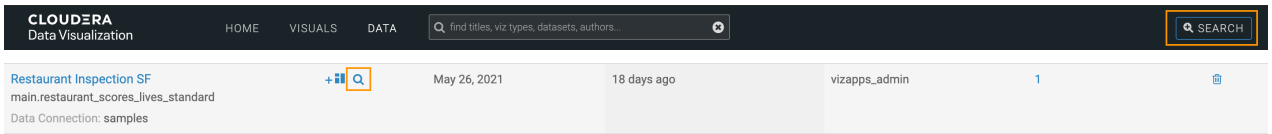

When you click SEARCH, the search modal window opens. It is shown in the figure below:

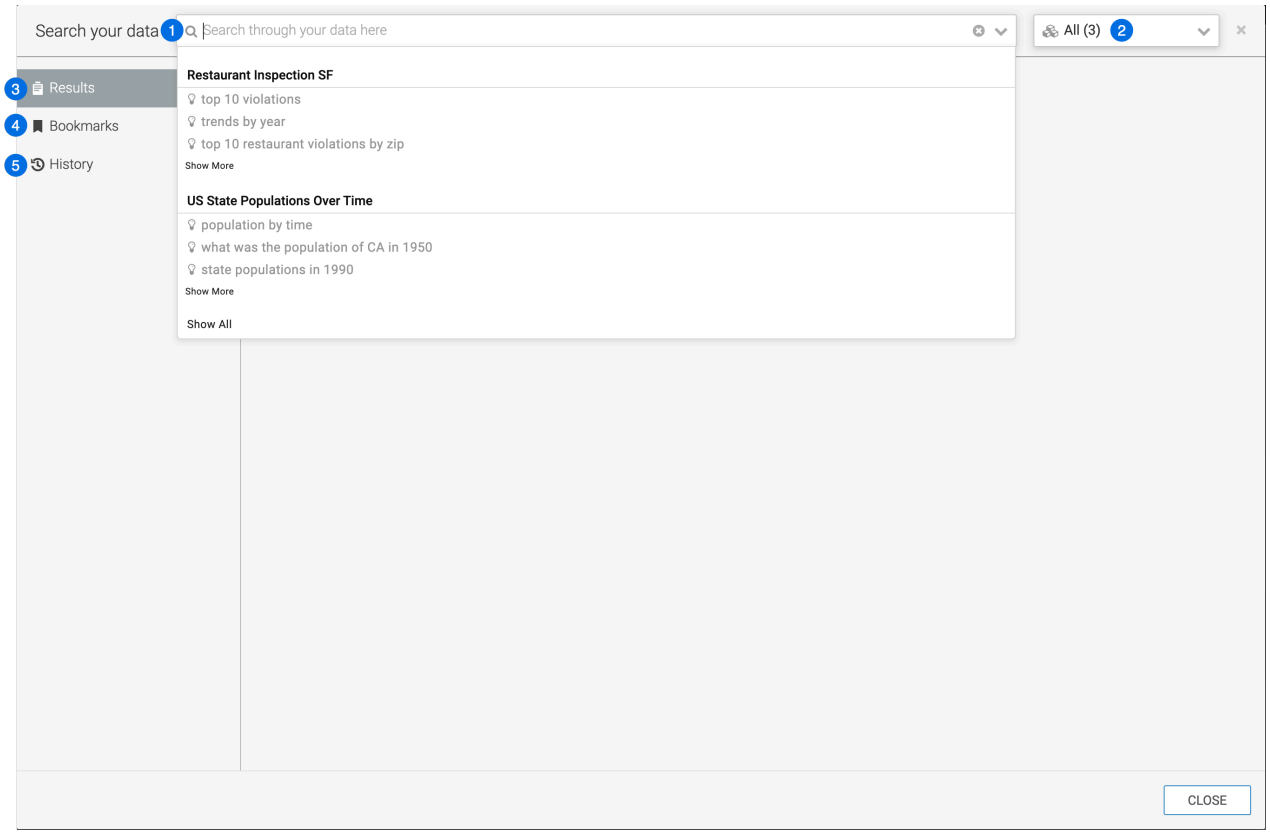

1. Search text box - You can enter search terms here. For more information on search suggestions in the drop-down list, see Search types.

   Note: The drop-down list initially shows a history of queries, organized by dataset. As you type, the search window changes to deliver intermediate results and shows many additional features of the search.

2. Dataset selector - By default the value is 'All' unless you access the modal from a search-enabled dataset page. You can narrow your search to a specific dataset, or a list of datasets - See Changing the dataset search domain.
3. Results tab - See Search results.
4. Bookmarks tab - See Search bookmarks.

5. History tab - See Search history.

Related Information

Enabling NLS in Site Settings
Enabling dataset for NLS
Search types
Changing dataset search domain
Search results
Search bookmarks
Search history

Search results

Unless the search phrase explicitly specifies the type of visual representation, Cloudera Data Visualization displays Natural Language Search results using a type that best matches the number and type of fields included in the search, for every searched dataset.

Cloudera Data Visualization displays the results of a Natural Language search directly in the search modal window.

The main features of the search results are shown in the figure below:

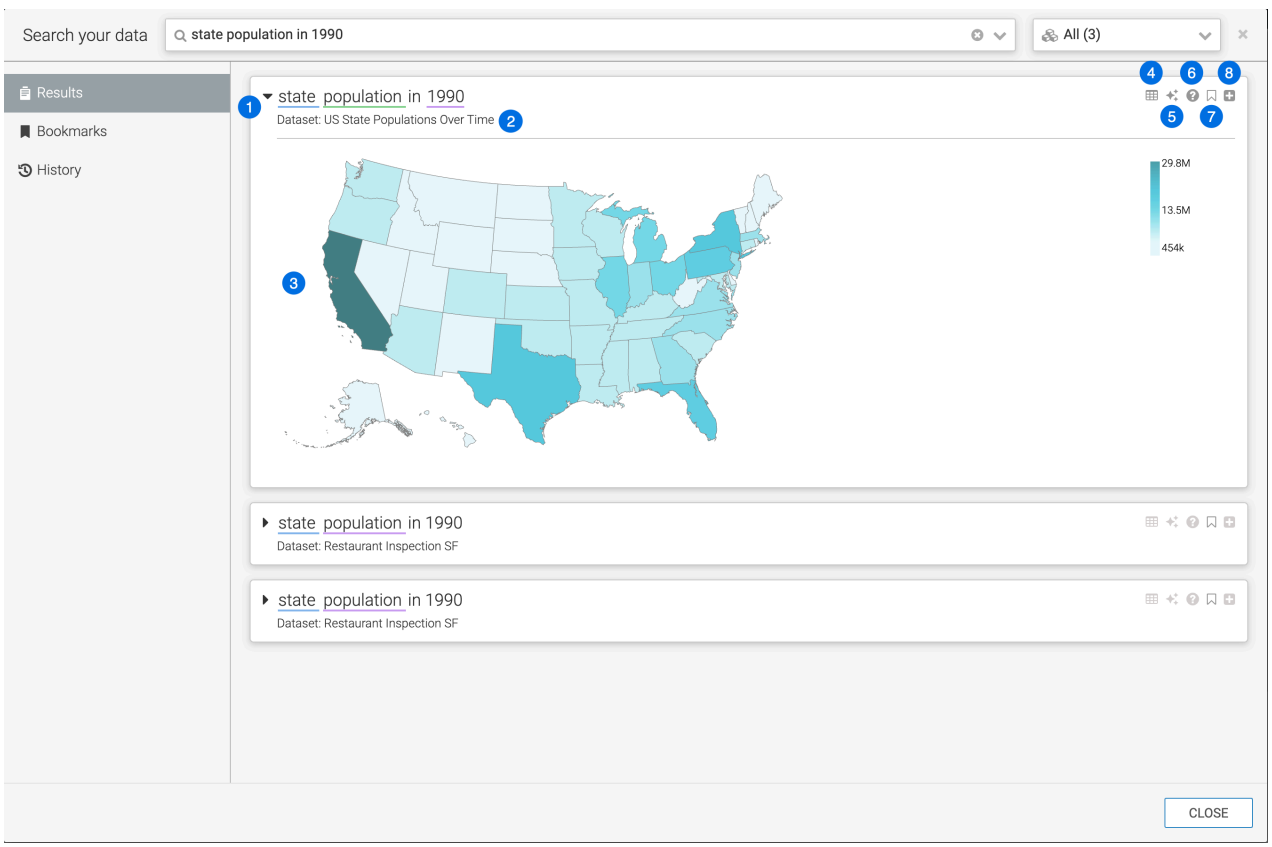

1. Search criteria

   This is identical to your original search. The various components of the search statement show colored highlights. If you move your mouse over these parts of the statement, the tooltip displays the details of the component.

2. Result dataset

   The dataset where a particular search result is found. Note that each search result may represent a different dataset.

3. Result visual

   The search result itself is a visual. Note that Cloudera Data Visualization displays results in a default visual type that best represents the number and type of fields in the search phrase.
4. View and download data

   This will show the data that is being graphed with an option to download as a CSV file.
5. Explore

   Clicking this icon enables you to choose an alternate visual. For more information, see Changing result visual type through explore.
6. Explain

   Hovering over this icon shows the dataset and connection informations, the SQL query and shelf information for the visual.
7. Bookmark

   Clicking this icon adds the visual to the bookmarked results. Clicking it again to removes it.
8. Dashboard actions

   Clicking this icon adds the visual to a dashboard.

Related Information

Changing result visual type through explore

Order of search results

Cloudera Data Visualization automatically ranks the results of Natural Language Search.

When search produces multiple results from different datasets, you can see the most relevant visual at the top of the list. The ranking is based on the highest number and accuracy of matched search terms.

Search bookmarks

Cloudera Data Visualization enables you to bookmark the results of Natural Language Search, and to easily manage these bookmarks. You can revisit the results using bookmarks without having to locate or search each time.

The Bookmarks tab of the Search modal window has the features displayed in the figure below:

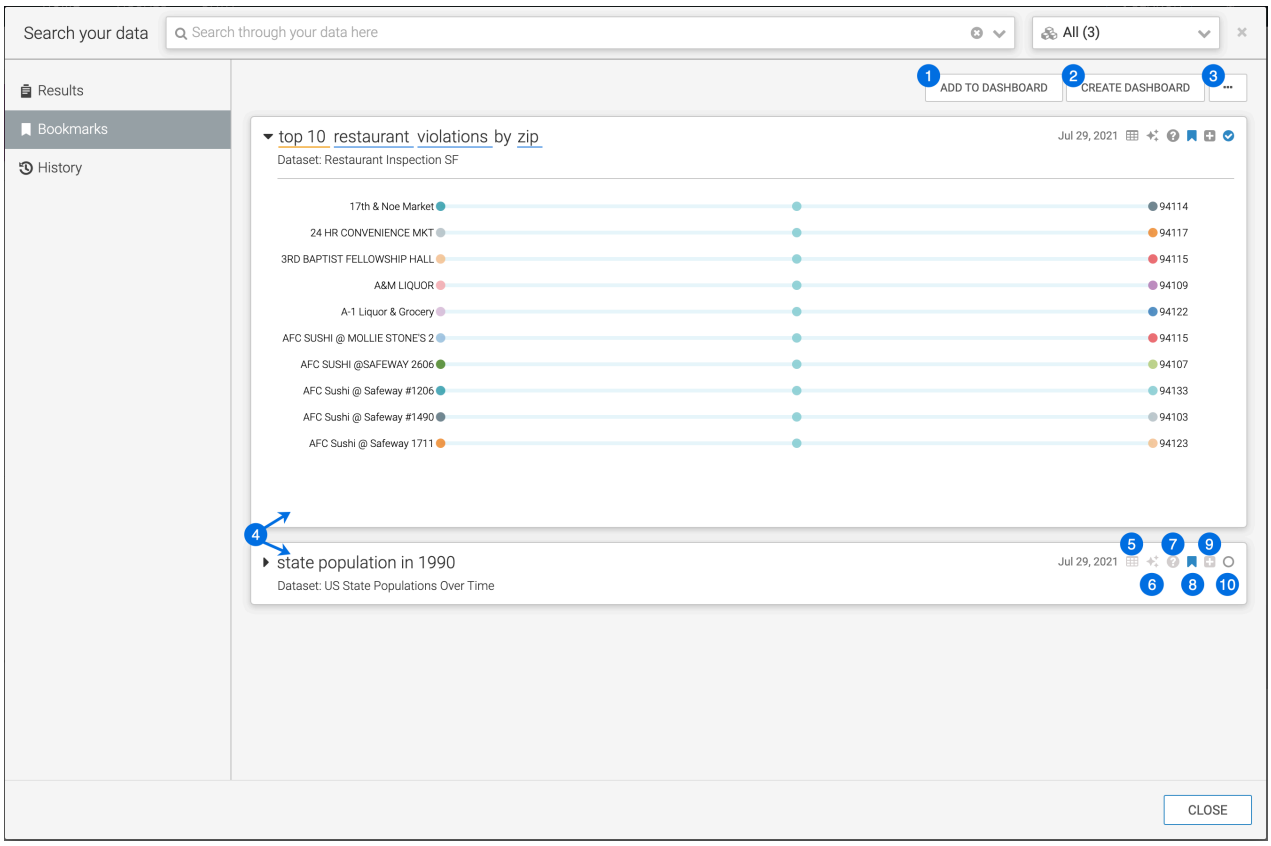

1. ADD TO DASHBOARD

   Selecting this option adds selected visuals to an existing dashboard. Note that this is a bulk operation.
2. CREATE DASHBOARD

   Selecting this option creates a new dashboard that contains selected visuals. Note that this is a bulk operation.
3. Supplemental menu

   Supplemental options enable you to make bulk changes to selected results.

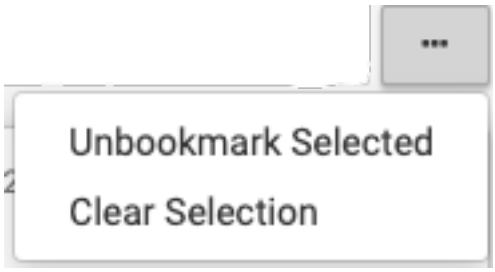

4. Bookmarked results

   The main area of the Bookmarks tab shows a list of bookmarked results. By default, they are in collapsed mode.
5. View and download data

   This shows the data that is being graphed with an option to download as a .csv file.
6. Explore

   Clicking this icon enables you to choose an alternate visual. See Changing result visual type through explore.
7. Explain

   Hovering over this icon shows the dataset and connection information, the SQL query and shelf information for the visual.

8. Unbookmark

   This enables you to remove visuals from the bookmarks.
9. Dashboard options

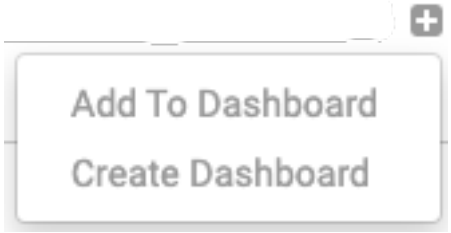

10. Selector

    You can select or deselect visuals to make bulk actions on them.

Related Information

Changing result visual type through explore

Search history

You can browse the history of Natural Language Search in Cloudera Data Visualization, and select the results listed there.

The History tab of the Search page consists of a list of searches made previously, and the date of the search. Cloudera Data Visualization groups this list into three categories:

- Last 7 Days
- Last 30 Days
- Older Searches

Search types

For faster search results, Cloudera Data Visualization enables you to choose from different Natural Language Search types.

In illustrating these topics, the search-enabled, sample datasets Cereals and World Life Expectancy are used.

Hit highlighting

To easily differentiate the visuals in the Natural Language Search results, Cloudera Data Visualization highlights the visuals that match the text in the search text box.

When the user enters text in the search text box, the dropdown list displays all the strings that match the characters entered in the search box. To easily identify and select one of the options, the typed characters are highlighted in a different color.

Match field names

For faster search results, Cloudera Data Visualization enables you to search visuals based on the field names entered in the Natural Language Search text box.

When you enter a field name in the search text box, the dropdown list highlights all the field names that match the characters entered in the search box. For example, if you type 'po' then the dropdown list highlights the fields

names, 'tot_pop', 'potassium_mg', and so on. The dropdown list also matches within the strings, so in our example, 'dropoff_longitude' field name is also highlighted.

In the following example, 'dropoff_longitude' and 'dropoff_latitude' field names are also highlighted.

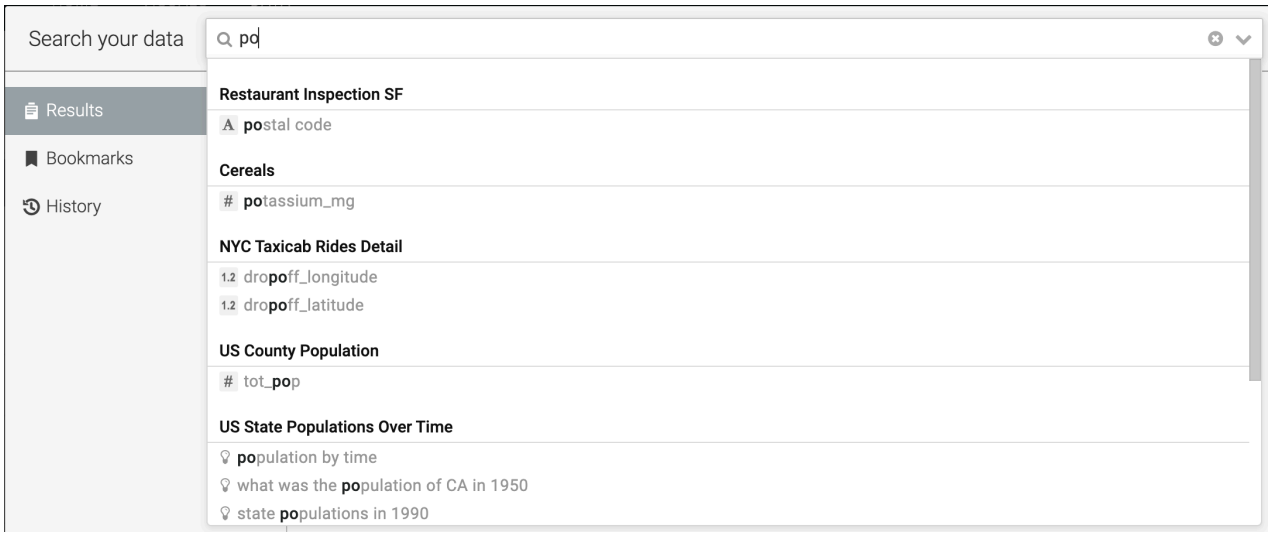

Matching suggested questions

For easy and faster search, Cloudera Data Visualization suggests predefined questions in the Natural Language Search drop-down list.

When you start to enter text in the search text box, the drop-down list displays predefined suggested questions that match the characters entered in the search box. In the below example, when the characters 'highest' are entered, the drop-down list highlights a suggestion with the characters ‘highest’ in the string.

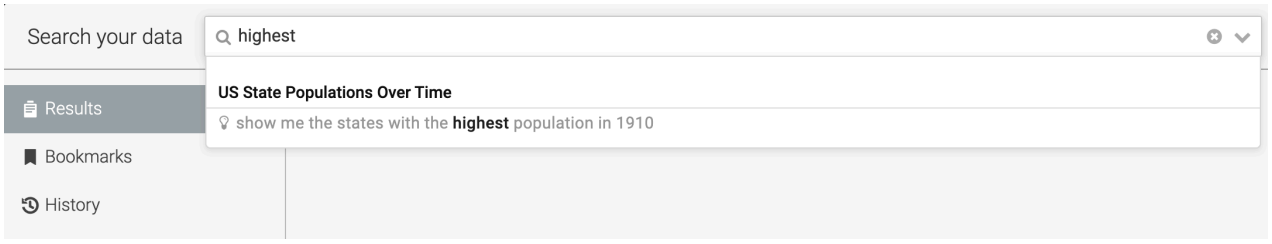

Bookmarked searches

For easier and faster results, Cloudera Data Visualization enables you to search from a list of favorites that are saved as bookmarks.

When you first click the search box and before you start typing, your most recent bookmarked searches appear in the drop-down list.

After you start typing, if the bookmarked options are still relevant for your current search, you can select one of the bookmarked options, as they are still shown in the drop-down list.

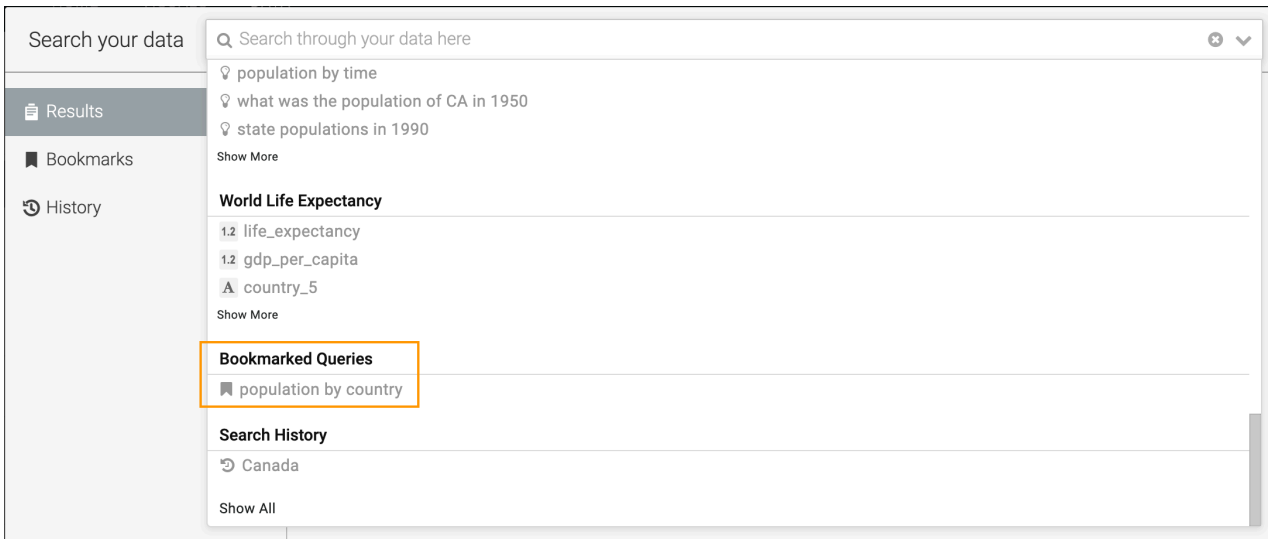

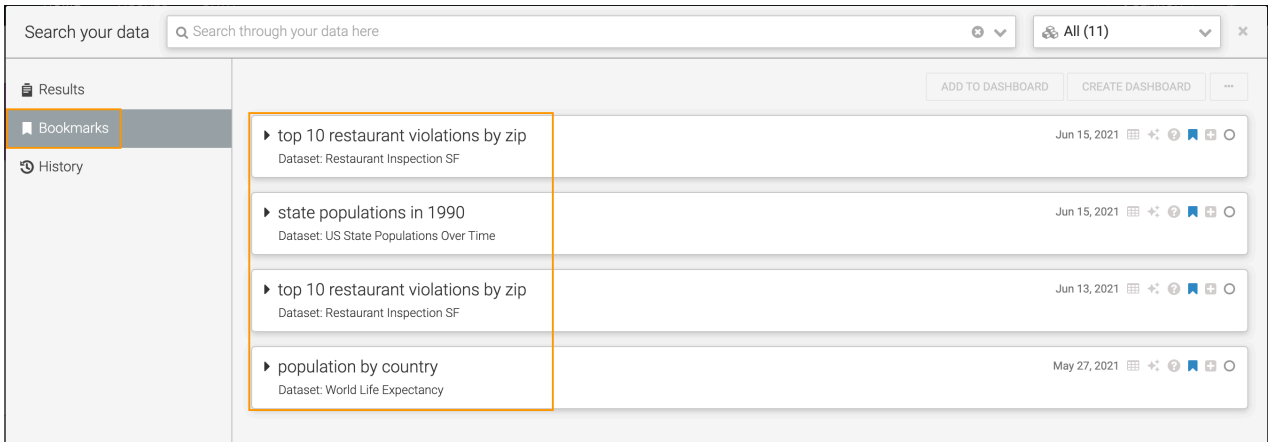

Search history

For easier and faster search results, Cloudera Data Visualization enables you to choose from the most recent Natural Language Searches.

When you first click the search box and before you start typing, your most recent searches appear in the drop-down list.

After you start typing, if your previous search is still relevant for your current search, you can select one of the previous search options, as they continue to appear in the drop-down list.

View more searches

To declutter the Natural Language Search drop-down window, Cloudera Data Visualization displays only a few matching values for any particular suggestion type.

If you click Show More in the search drop-down list under a particular dataset, all search results will be displayed for that dataset.

If you click Show All at the very end of the drop down list, all available searches will be displayed.

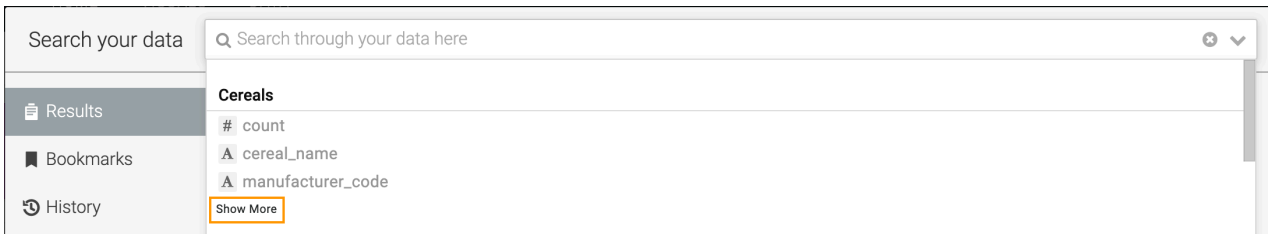

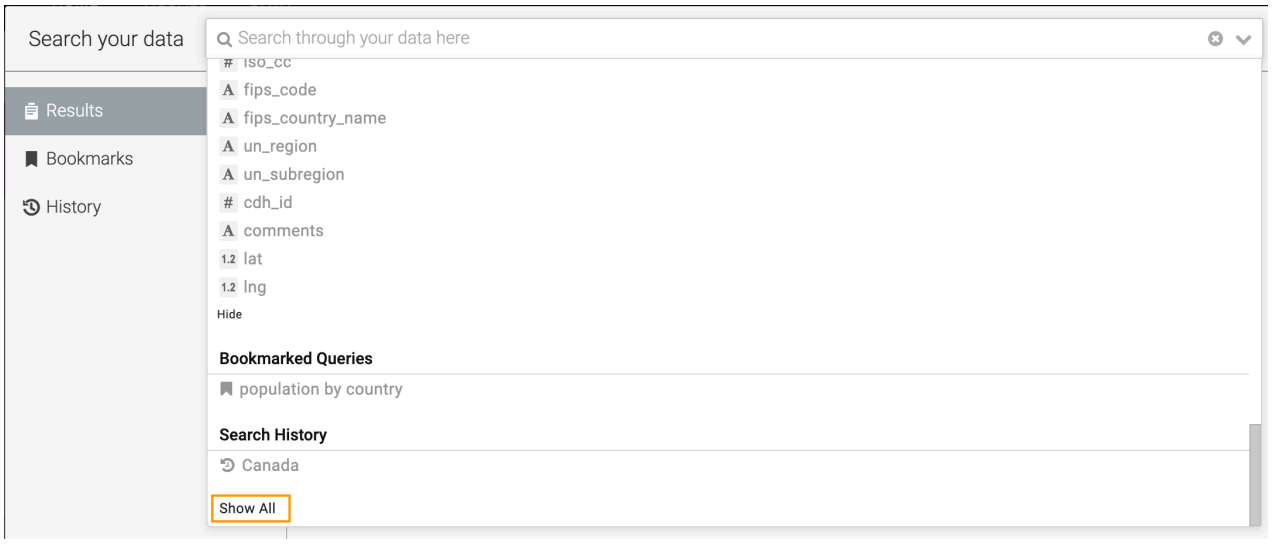

Global search

Global search quick access to various functions and assets within Cloudera Data Visualization, saving you time navigating through menus.

Key uses of global search:

- Create or find assets: Quickly locate existing data assets or create new ones.
- Perform actions: Initiate certain tasks directly from the search results.
- Search through documentation: Easily find relevant help topics or support articles.

On the HOME interface (main of page Cloudera Data Visualization), you can access the global search text box from the Help menu.

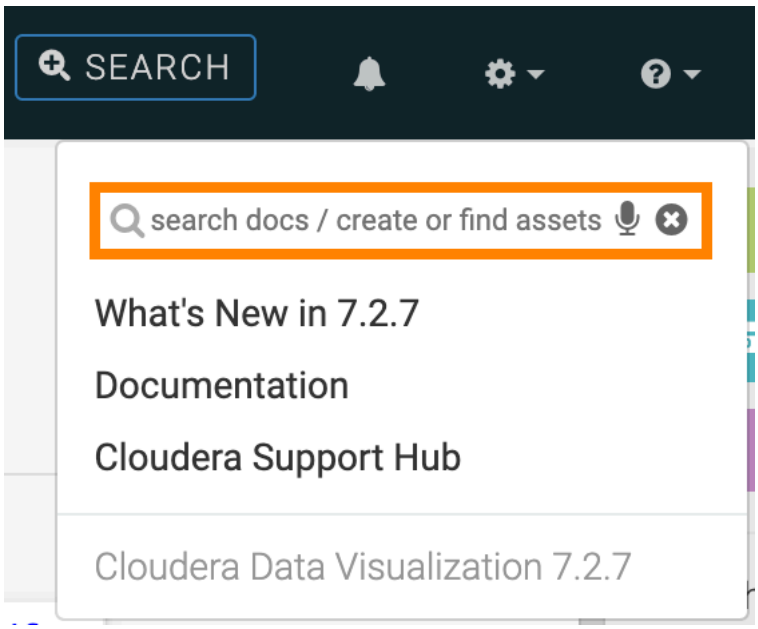

If you switch to another interface, for example to DATA, you can find the search box on the main navigation bar for easy access.

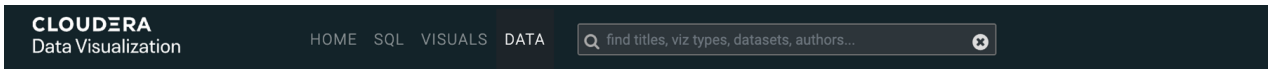

Global search supported keywords

The keywords currently supported by global search in Cloudera Data Visualization.

Create

Create app or Create application

Creates a new, empty App and opens the App Designer page.

Create dashboard

Creates a new, empty Dashboard and opens the Dashboard Designer page.

Create data connection or Create connection

Opens the Data page and the Create New Data Connection modal.

Create dataset

Opens the New dataset modal from the last viewed data connection.

Run SQL

Opens the SQL page.

Find

Find data or Find dataset

Opens the data page and inputs remaining text as search input.

Find app or Find application

Opens the visuals page, Apps tab selected, and input remaining text as search input.

Find dashboard

Opens the visuals page, Dashboards tab selected, and inputs remaining text as search input.

None keywords

Everything not listed becomes a search term to be used for searching through the documentation. The search opens in a seperate window. For and about keywords are removed from the beggining if included.

Cloudera Data Visualization 7.2.9

Natural Language Search References

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Icons for search types 4

NLS terms for basic filters 4

NLS terms for ranking 5

NLS terms for aggregation 6

NLS terms for sorting 6

NLS terms for comparing data points 7

NLS terms for visual types 7

NLS terms for dimension field matching 8

NLS terms for time series 8

Icons for search types

To identify the type of search, Cloudera Data Visualization displays the following icons in the Natural language Search (NLS) results.

Field name: string

Field name: integer

Field name: decimal

Field name: geo

Field name: timestamp

Field value

Field name: boolean

Suggested question

Suggested field

Bookmarked search

Historical search

NLS terms for basic filters

Cloudera Data Visualization provides Natural Language Search (NLS) support for basic filter operations.

To enable search using the terms such as ‘including’, ‘like’, ‘excluding’, ‘matching’, and so on, refer to the following functions:

Function	Example
all	Count all employees with sales < 10000

Function	Example
between; and	Revenue between 0 and 1000 cereal_name by manufacturer where calories between 50 and 100
vs; versus	Revenue east versus west
greater than; >	Sum sale amount, by visitor and product, for last year sale amount > 2000 cereal_names where calories greater than 100
less than; <	Unique visitor count, by product and store, for sale amount < 20 cereal_name by manufacturer where calories less than100
>=	Count calls by last name of the employee >= m cereal_name with calories >= 100
<=	Count shipments by latitude of the city <= 0
=	Unique visitor count by store purchased products = 3 for last 5 days
!=	Sum of sale amount by region != Canada and region != Mexico
begins with	Product name begins with "pro"
contains	Product name contains "alpine" and description contains "snow shoe"
ends with	Product name ends with "deluxe"
not begins with	Product name not begins with "tom's"
not contains	Product color not contains "tan" and product color not contains "red"
not ends with	Product name not ends with "trial"
similar to	Course name similar to "hand"
not similar to	Course name not similar to "hand"

NLS terms for ranking

Cloudera Data Visualization provides Natural Language Search (NLS) support for ranking operations.

The ranking operations are the following functions:

Function	Example
top	Top sales representative with sales > 10000 Sales representative by average revenue of top regions
bottom	Bottom average revenue

Function	Example
	Bottom revenue by state Customer by revenue, for each bottom sales representative
n (Top Numbers)	Top 10 sales representative by revenue
n (Bottom Numbers)	Bottom 25 customers by revenue, for each sales representative

NLS terms for aggregation

Cloudera Data Visualization provides Natural Language Search (NLS) support for aggregation concepts.

The aggregate support includes the following functions:

Function	Example
sum	Sum revenue
average	average revenue by store
count	Visitor count by site
max	maximum sales by visitor and by site
min	minimum revenue by store, by campaign, for cost > 5000
standard deviation	standard deviation revenue by product, by month, for date after 10/31/2010
unique count	unique count visitors last week, by product page
variance	variance sale amount last year, by visitor and by product

NLS terms for sorting

Cloudera Data Visualization provides Natural Language Search (NLS) support for sorting.

You can sort the search results by using the following functions:

Function	Example
order by	Viewership by channel order by viewership descending Viewership by channel order by channel
sort by	Sales order_date sort by order_date ascending

NLS terms for comparing data points

Cloudera Data Visualization provides Natural Language Search (NLS) support for comparing data points.

The user can request search results that compare two values, such as actual versus target.

Function	Example
versus; vs. (Metric Comparison)	Compare sales 'vs.' headcount
versus; vs. (Field Value Comparison)	Compare sales for region A 'vs.' region B
versus; vs. (Time Value Comparison)	Compare sales this week 'vs.' sales same week last year
versus; vs. (Contribution, or Window, Comparison)	Compare sales for region A 'vs.' all regions

NLS terms for visual types

In Cloudera Data Visualization, you can specify a visual type that displays the results of Natural Language Search (NLS).

The support for visual types includes the following function:

Funtion	Example
as	Revenue and quantity sold by product name as scatter

The Natural Language Search supports the following visual types:

Visual Type	Keyword
Table	table, data
Cross Tabulation	crosstab
Bar	bar
Lines	line
Combo	combo, combined
Areas	area
Grouped Bars	grouped bar, groupedbar
KPI	kpi
Gauge	gauge
Bullet	bullet

Visual Type	Keyword
Packed Bubbles	packed bubble, bubble
Scatter	scatter
Pie	pie
Radial	radial
Chord	chord
Network	network
Dendogram	dendogram
Treemap	tree map, treemap
Correlation Heatmap	correlation heatmap, heatmap
Map	map
Interactive Map	interactive map, leaflet
Sparklines	sparkline

NLS terms for dimension field matching

Cloudera Data Visualization provides Natural Language Search (NLS) support for matching a field in the dimension shelf.

The user can request search results to match a field (base or derived), in the Dimension shelf, defined on any search-enabled dataset.

Function	Example
net	Net Profit
by	By Customer
for	For Products

NLS terms for time series

Cloudera Data Visualization provides Natural Language Search (NLS) support for time series concepts.

When the user enters a time-series term in the search box, such as Trends or Over Time, Data Visualization automatically recognizes that the user is requesting a time-series based visual, and renders a visual with time field on the X-axis.

Function	Example
trends	Sales 'trends'
over time	headcount 'over time'

Cloudera Data Visualization 7.2.9

Using Natural Language Search

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Search in embedded mode **4**

Working with search visuals **4**

Creating a search visual 4

Using the search visual in a dashboard 6

Changing dataset search domain **8**

Changing result visual type through explore **11**

Using Search in embedded mode

To use Cloudera Data Visualization in embedded mode, a search visual has to be added to the dashboards, which provides the same functionality. For information on how to configure a search visual, see Working with Search visuals.

Related Information

Working with search visuals

Working with search visuals

The purpose of the search visual is to include search functionality in embedded dashboards.

Search visuals enable you to place a data search button directly on your dashboard. This is particularly useful in embedded cases. The Enable Search Visual, Enable Search in Menu Bar, and Enable Search Visual settings need to be enabled in Site Settings to turn on the search feature and the datasets need to be enabled separately to add them to the search results. See *Enabling NLS in Site Settings* and *Enabling dataset for NLS*.

Related Information

Enabling NLS in site settings

Enabling dataset for NLS

Creating a search visual

The purpose of the Search visual is to include search functionality in embedded dashboards.

Procedure

1. Start a new visual.

   For instructions, see Creating a visual.

2. Click Search in the VISUALS menu.

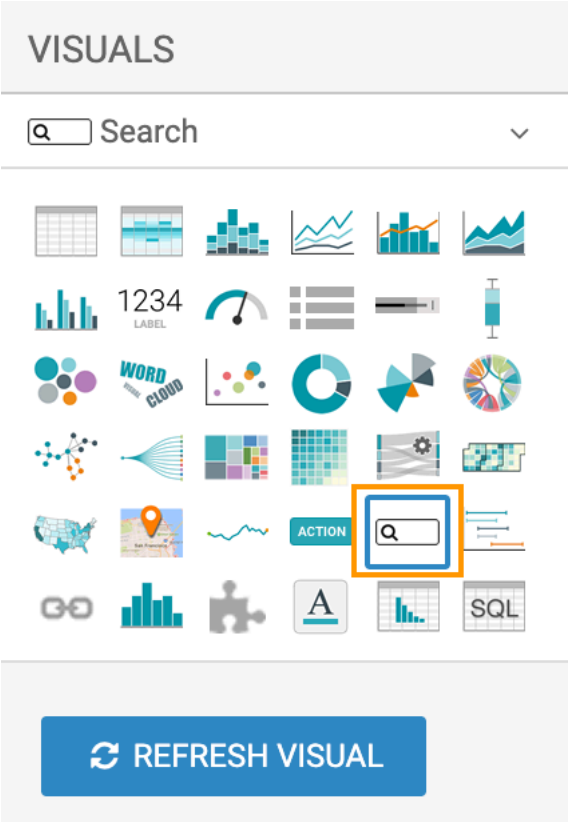

Note: This visual type does not use shelves.

3. Configure the Search visual.

   There are two drop-down lists in the main area of the Dashboard Designer.

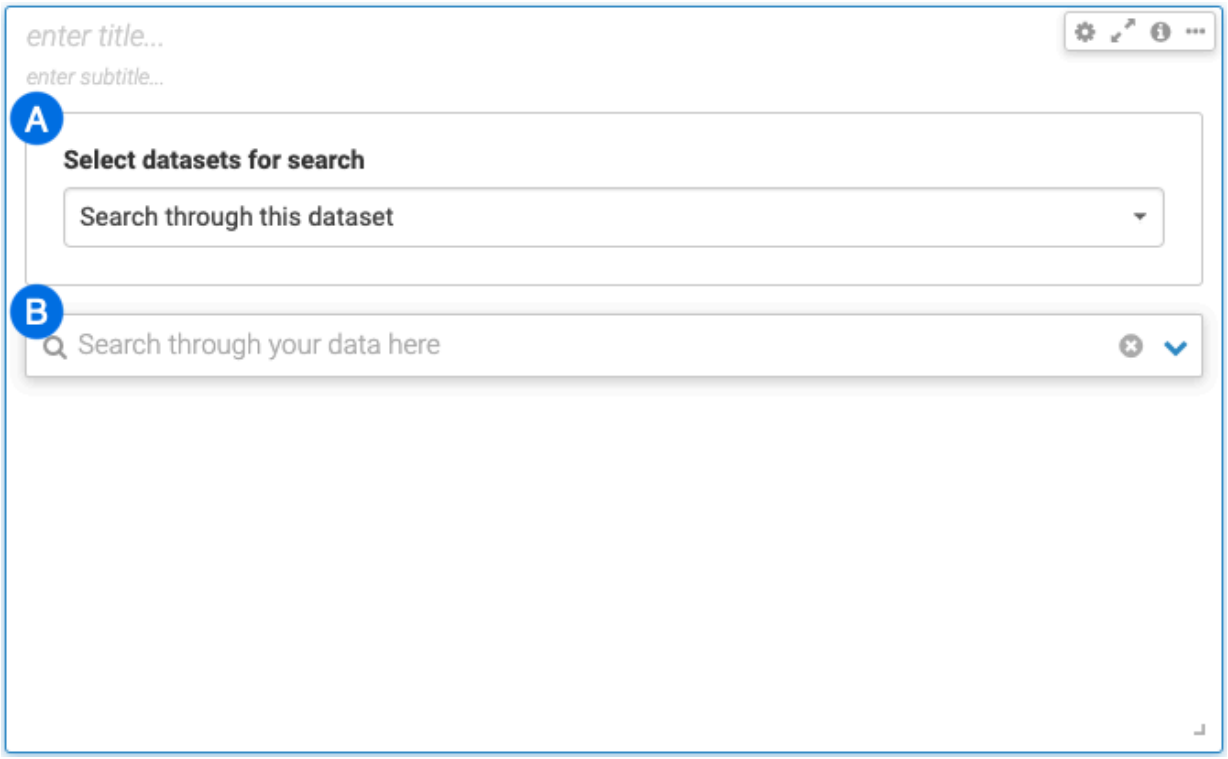

   a. Select datasets for search: The dataset selector menu lets you restrict the search to specific datasets with the following options:

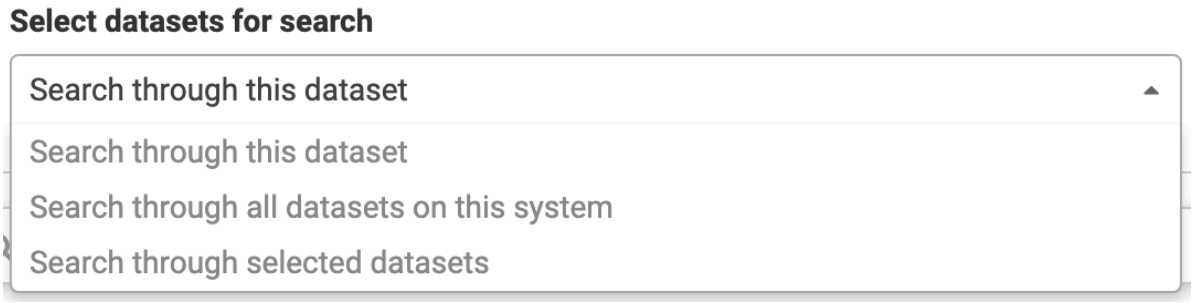

   Note: The current dataset does not have to be search enabled so the Search through this dataset option works. The Search through all datasets and the Search through selected datasets options only show search enabled datasets. See Enabling dataset for NLS.

   b. Search box: This is the area where you enter the search phrase for visualizing your data.

4. Enter a title for the visual.
5. Click SAVE in the top left corner.

Related Information

Creating a visual

Enabling dataset for NLS

Using the search visual in a dashboard

After creating a Search visual, it is available for use from the Dashboard page like any other visual.

Before you begin

Before performing this task, you need to create a Search visual. For instructions, see Creating a search visual.

Procedure

1. Open the dashboard from the Dashboard page.

   The Search Box visual appears in the dashboard. Its only element is the search box itself.

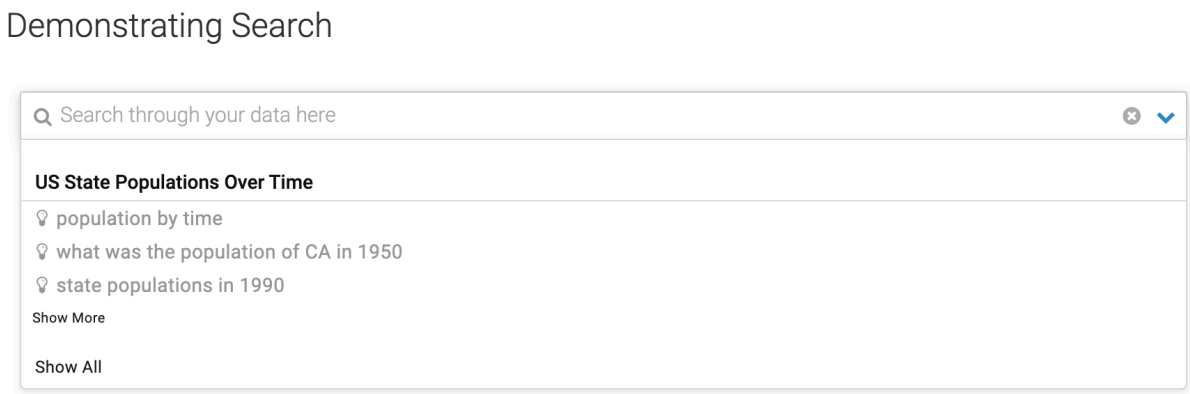

2. Click the search text box and choose one of the following search alternatives:
   - Type in a new search statement.
   - Select one of the searches from the drop-down list. They are grouped into the following categories:
     - Grouped by Dataset
     - Favorites
     - Search History

3. Select a search.

   The visual changes. It shows both the search box with the specified search statement and the visual that reports the results of that statement. In the following example, ‘population by time’ is selected.

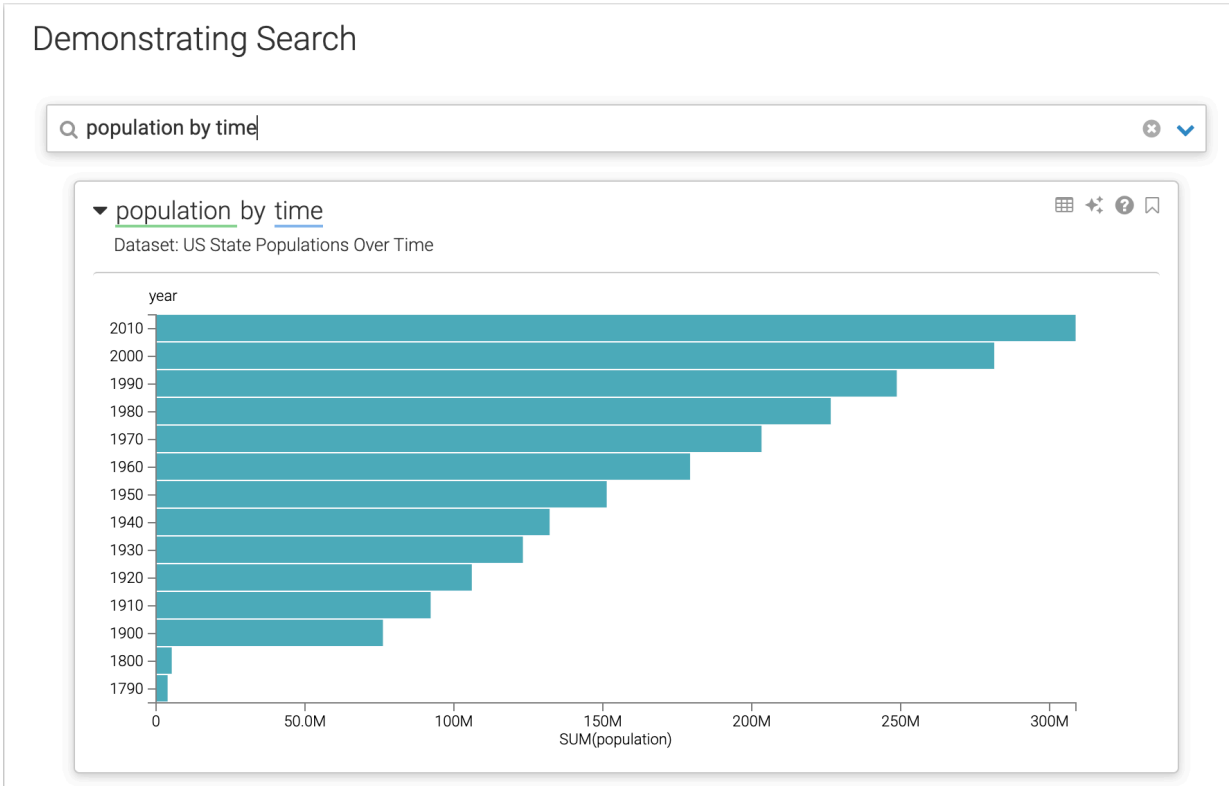

   For more information on the available actions, see Search results.

Related Information

Search results

Creating a search visual

Changing dataset search domain

In Cloudera Data Visualization, you can perform Natural Language Search (NLS) across all enabled datasets, or restrict your search to selected datasets.

About this task

When searching across all search-enabled sample datasets, which is the default setting, the search population by state returns results from five datasets: US State Populations Over Time, Restaurant Inspection SF, Food Stores Inspection in NYC, US County Population, and World Life Expectancy.

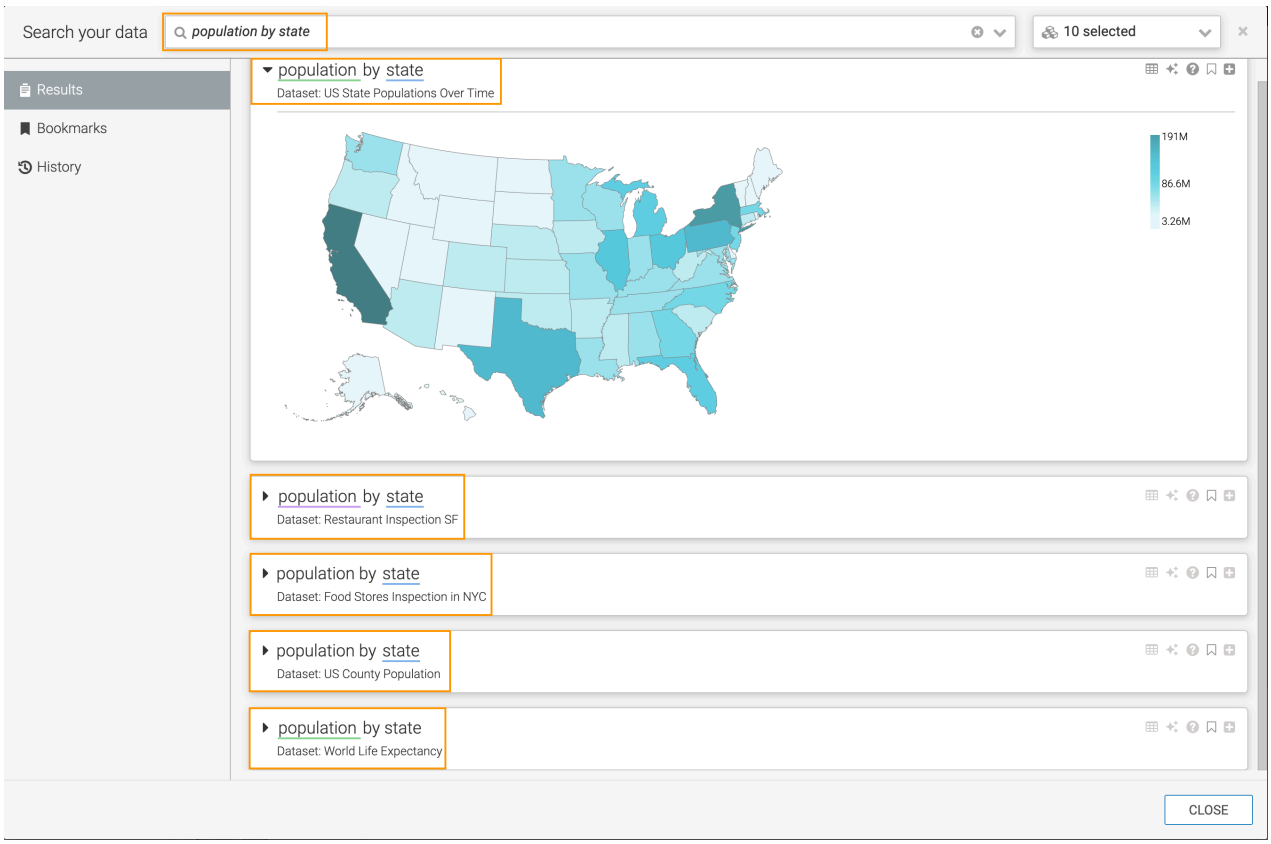

To limit the search:

Procedure

1. Select the Datasets menu in the top right corner.
2. Deselect all the datasets that you do not want to include in the search.

   This restricts the search to relevant data.

   Note: You may choose to deselect all datasets, and then selectively add some of them back.

3. Perform the search again.

Example

In the figure below, the US County Population and US State Populations Over Time datasets are selected, so the results based on the other datasets no longer appear in the search results.

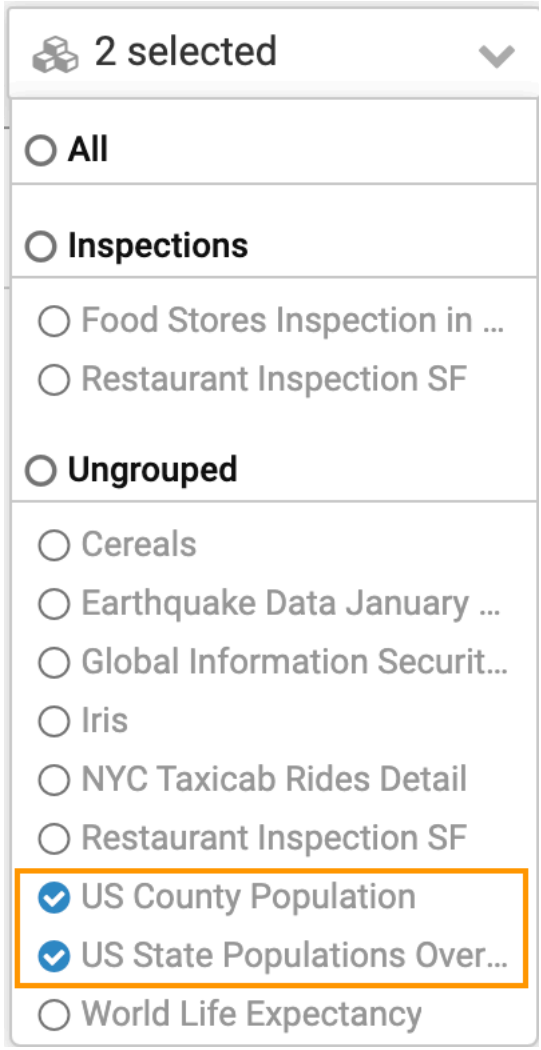
2 selected
All
Inspections
Food Stores Inspection in ...
Restaurant Inspection SF
Ungrouped
Cereals
Earthquake Data January ...
Global Information Securit...
Iris
NYC Taxicab Rides Detail
Restaurant Inspection SF
US County Population
US State Populations Over...
World Life Expectancy

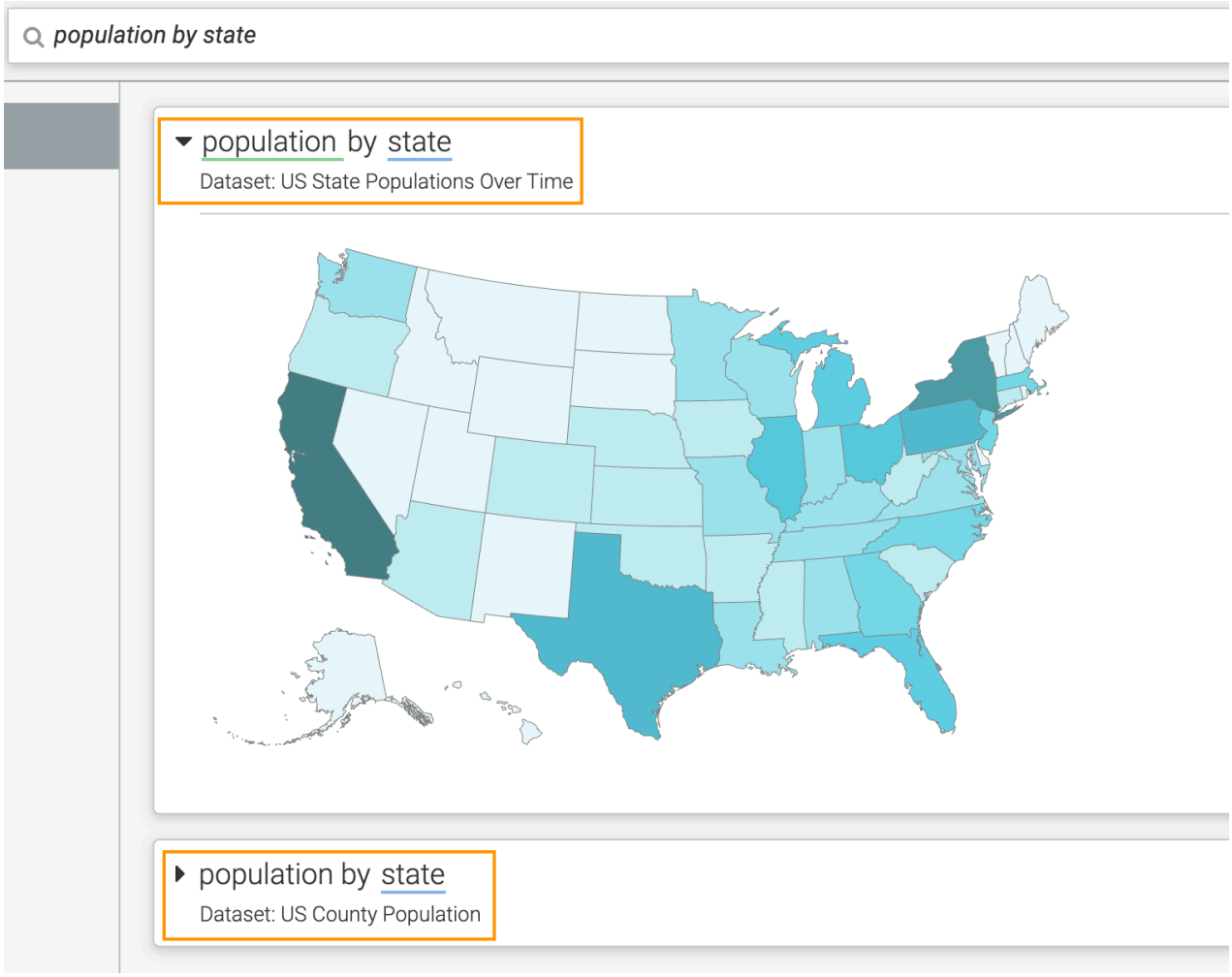

Changing result visual type through explore

Procedure

1. In the Search Results menu, select the visual you would like to change, and click the Explore icon.

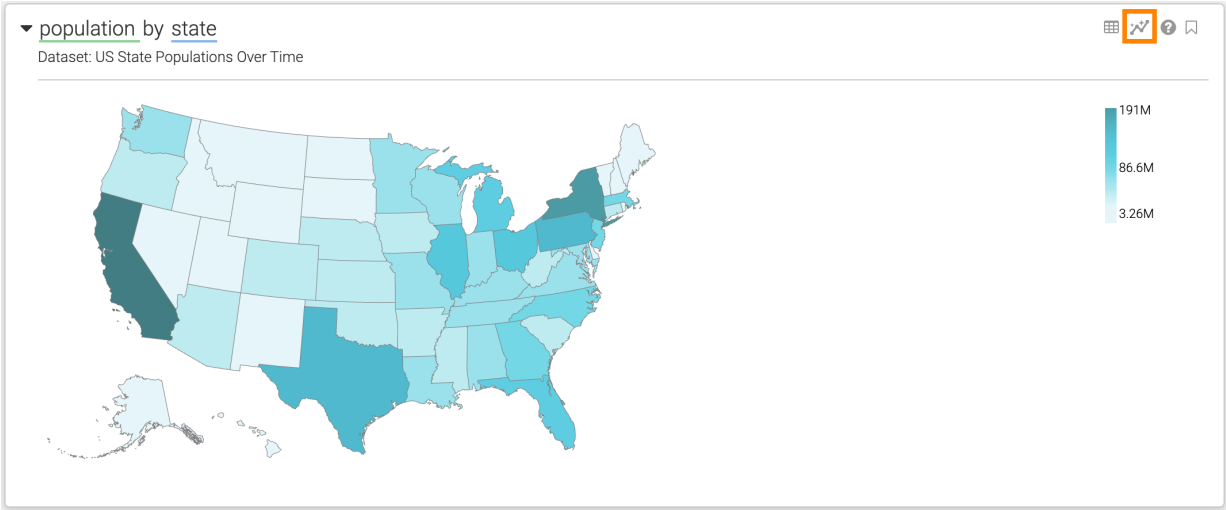

A modal window opens, where you can preview the different visuals. The top left corner of each tile indicates the visual type.

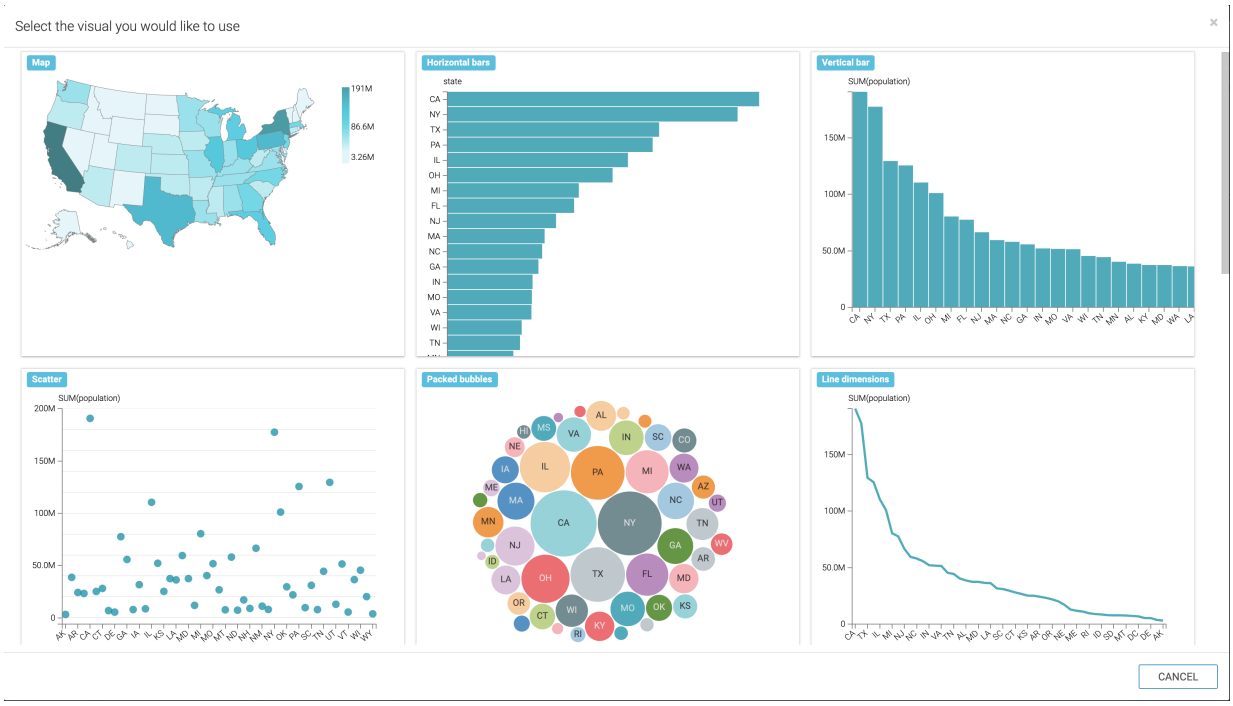

2. Select one of the visuals.

Results

The new visual type replaces the one originally generated by the search.

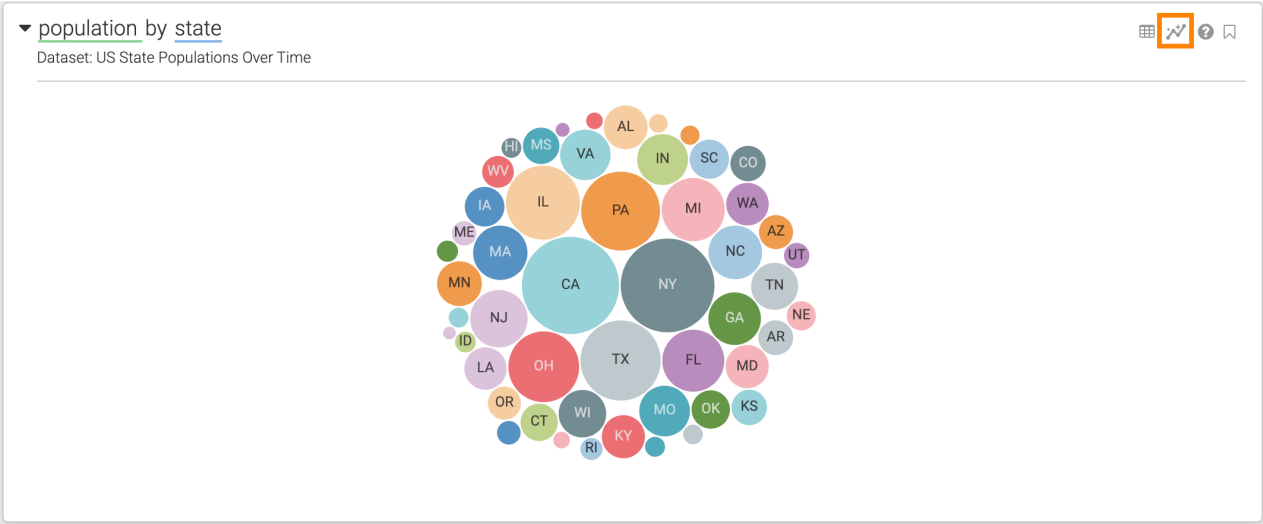
population by state
Dataset: US State Populations Over Time
AL
HI
MS
VA
IN
SC
CO
WV
IL
PA
MI
WA
IA
AZ
ME
MA
NC
UT
MN
CA
NY
TN
NJ
GA
NE
ID
AR
LA
OH
TX
FL
MD
OR
CT
WI
MO
OK
KS
KY
RI

Cloudera Data Visualization 7.2.9

Notifications

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Notification center.....4

Notification center

The notification system of Cloudera Data Visualization is designed to help you stay informed and manage important system events. Notifications are messages that provide information about errors, system failures, or security issues.

They help you to stay informed without interrupting your ongoing work. You can access and manage your notifications from any page using the Notifications icon located on the main navigation bar. An orange dot on the bell icon indicates that there are unread notifications. When you click the icon, the Notification Center opens in a new modal window, showing the system log entries in a single chronological list. The messages are arranged in order of recency, with the most recent item appearing at the top of the list.

You can refresh the list of notifications by clicking the button.

You can click read more or read less to expand or collapse the full notification text.

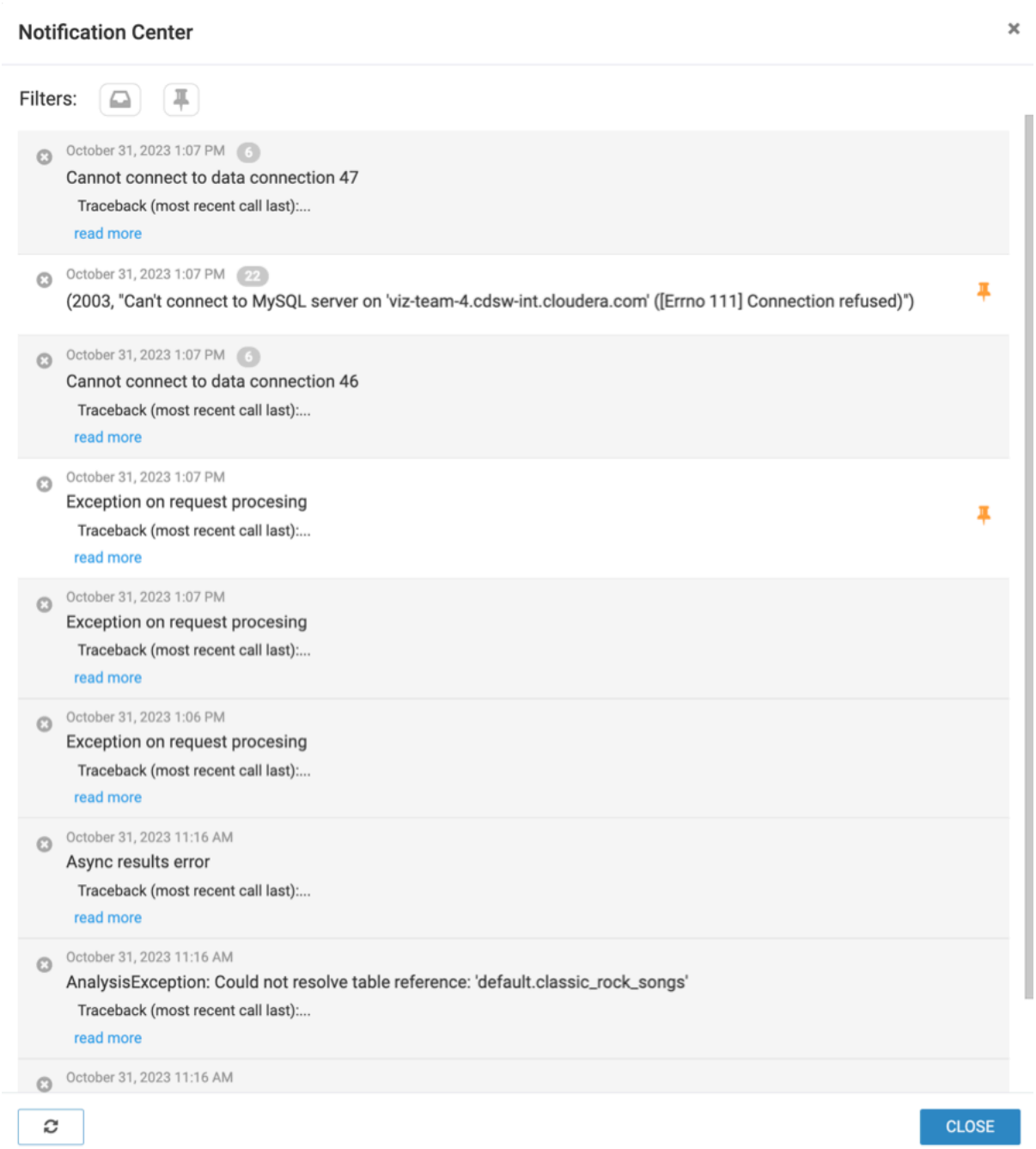

Muting notifications

Muting notifications prevents you from receiving notification messages from Cloudera Data Visualization. Muted notifications are stored as read and are automatically deleted at the end of each day. Administrators can apply application-wide notification muting for Cloudera Data Visualization.

For more information on enabling the Mute Notifications option, see *Site Configuration*.

Managing read notifications

Once you close the Notification Center modal, notifications are marked as read. Read notifications are retained for one day, with pinned messages as an exception, which remain preserved even after clearing the read notifications.

Administrators can adjust the length of the message retention period. For more information, see *Advanced Settings*.

Managing duplicate notifications

Duplicate notifications are stacked to prevent clutter and redundancy in error reporting. This feature also helps in diagnosing issues by highlighting high-frequency errors. A counter appearing next to the timestamp displays the number of identical errors. Clicking this badge displays a list of timestamps for each occurrence of the error.

Pinning notifications

If you want to prevent clearing a notification from the list, you can pin it. Pinned notifications have a white background and are marked by an orange pin.

Filtering notifications

You can filter notifications to streamline your view.

Show unread/all notifications

This filter narrows the list to show only the messages that have not been read or acknowledged. This allows you to focus on managing new or unopened items without the distraction of previously viewed messages.

Show pinned/all notifications

This filter narrows the list to show only the pinned notification messages, providing quick access to essential information.

Related Information

Site Configurations

Advanced Settings

Cloudera Data Visualization 7.2.9

Data Visualization Overview

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Data Visualization capabilties **4**

Cloudera Data Visualization user interface **4**

Cloudera Data Visualization homepage 5
Cloudera Data Visualization, SQL interface 10
Cloudera Data Visualization, Visuals interface 12
Cloudera Data Visualization, Data interface 16

Cloudera Data Visualization capabilties

Cloudera Data Visualization enables you to explore data and communicate insights across the whole data lifecycle by using visual objects. The fast and easy self-service data visualization streamlines collaboration in data analytics through the common language of visuals.

Cloudera Data Visualization is integrated with Cloudera AI and Cloudera Data Warehouse workflows in Cloudera Public Cloud. You can also use it on premises with Cloudera Data Science Workbench workflows. You can use the same visualization tool for structured, unstructured/text, and machine learning analytics, which means deeper insights and more advanced dashboard applications. The user interface is highly intuitive and enables you to build customized data applications with minimum time investment.

Advanced data analysis helps you to find data insights that answer your business problems. You can connect to your data in Cloudera, create rich visualizations on top of your datasets, build informative dashboards and applications, and publish them anywhere across the data lifecycle. This provides you with a dynamic and data-driven insight in any line of business, and lets you and other stakeholders discover, explore, and curate visualizations.

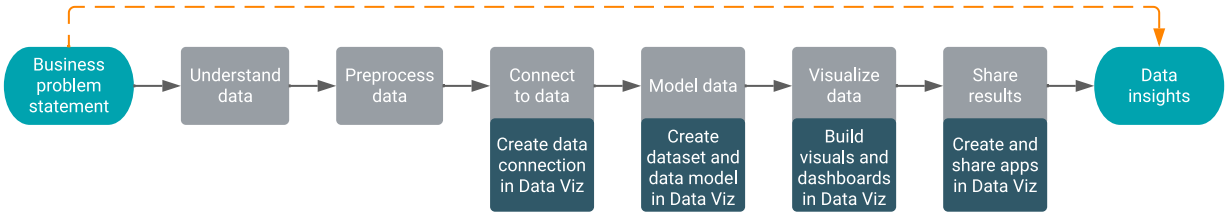

Self-service data visualization for the full data lifecycle

You can quickly and easily explore data, collaborate, and share insights across the data lifecycle. Cloudera Data Visualization provides a consistent and easy to use data visualization experience with intuitive and accessible drag-and-drop dashboards and custom application creation.

Intelligent and automated reporting

You can automate analytics dashboards for updates and consumption. You can set rules for scheduled updates, email reports, and dynamic alerts to ensure everyone has access to the latest insights.

Accelerating collaboration

You can build and publish custom dashboards and analytic applications in minutes and you can collaborate across teams to reach stakeholders and decision makers more effectively. Cloudera Data Visualization enables collaboration and transparency between data, analytics, and business teams without moving data to third party platforms.

Secure and integrated visualizations

Cloudera Data Visualization provides out of the box functionality without additional integration efforts, moving data, or creating security issues. From a security and governance perspective, native integration with Cloudera means SSO for authentication and seamless integration with Cloudera Shared Data Experience to manage user access and governance.

Cloudera Data Visualization user interface

The web interface of Cloudera Data Visualization has four views: HOME, SQL, VISUALS, and DATA. When you log in, you land on the homepage (HOME view) where the menus that you can find on the Main navigation bar enable you to access the other two interfaces.

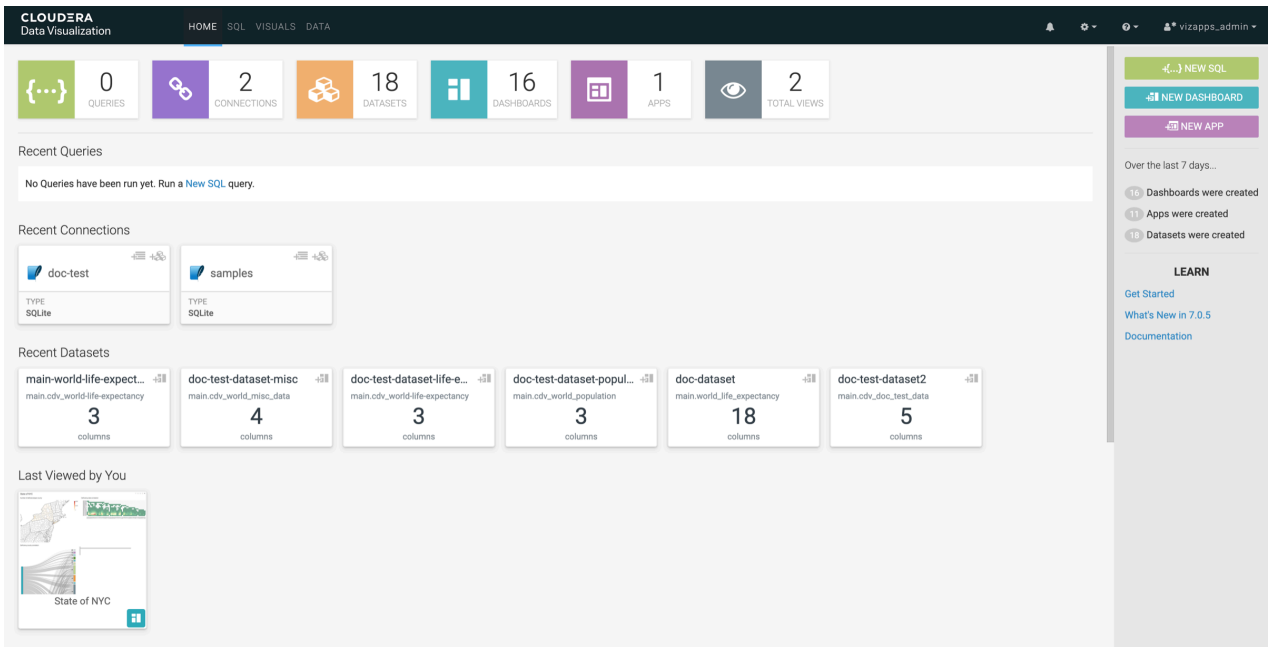

What users see and can access in the UI is determined by their role and permissions.

Cloudera Data Visualization homepage

Cloudera Data Visualization enables you to explore data and communicate insights across the whole data lifecycle by using visual objects.

When you log into Cloudera Data Visualization, you are taken to the HOME screen, which is the starting point to your data visualization journey. It displays recent views, favorites, and the application's most-viewed content. You can start exploring sample dashboards or the in-tool Get Started guide.

Clicking HOME on the main navigation bar always takes you to the homepage.

The default Cloudera Data Visualization homepage has the following main parts:

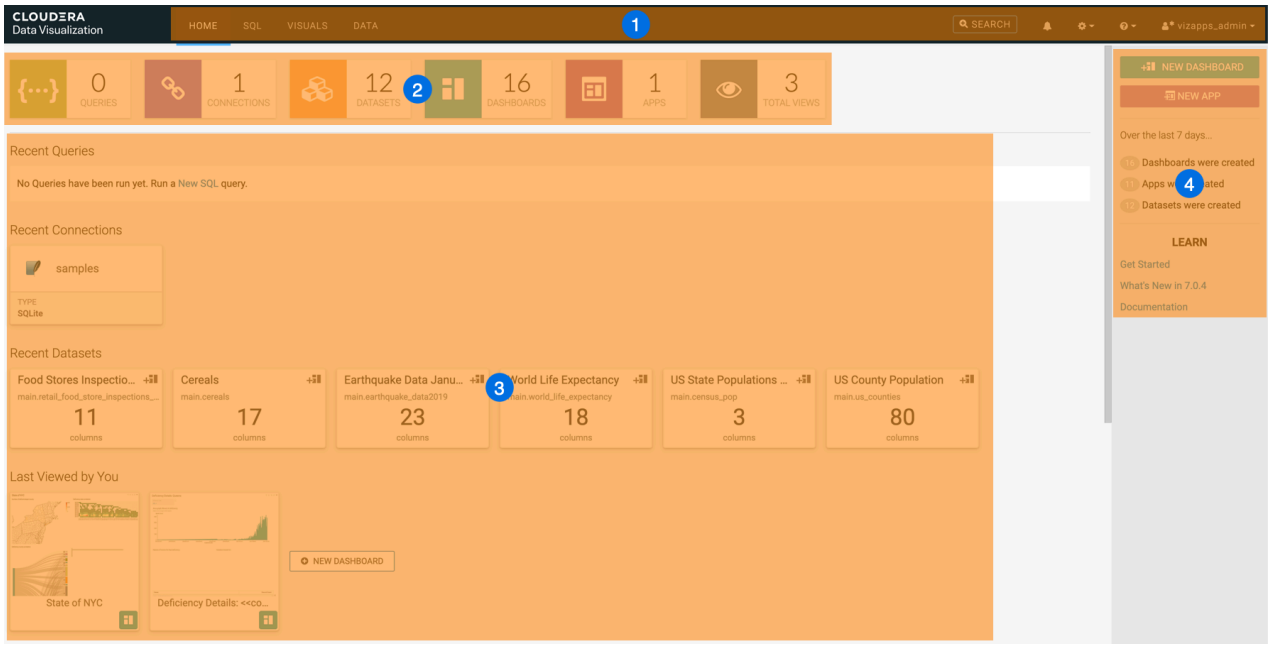

1. Main navigation bar
2. Statistics banner
3. Preview area
4. Homepage side menu bar

For information on how to customize the homepage, see *Custom Homepages*.

Main navigation bar

The main navigation bar enables you to directly access the following interfaces and functions:

1. The HOME interface is described in this topic.
2. The SQL interface to manually build queries.
3. The VISUALS interface is for building visuals, dashboards, and applications. For more information, see *Visuals interface*.
4. The DATA interface for access to datasets, connections, and the Connection Explorer. For more information, see *Data interface*.
5. The search bar for natural language search. It is only visible if search is enabled in the Site Settings.
6. The notification center that contains application, server and site settings errors and warnings.

   All users can see browser-level errors that might be affecting Cloudera Data Visualization. Additionaly, for admin users and users with permission to view activity logs, webserver errors (such issues with applying advanced site settings) are also available in the notification center.
7. The Settings menu provides access to various setting options:

   - With Set Homepage, you can define a custom homepage.
   - With Job Status, you can check job statuses.

   If you have administrative privileges, you can use Site Administration to make changes to the following functionalities:

   - Activity Log displays all recorded events generated by the data processing requests.
   - Users & Groups helps you to manage accounts of all users and user groups, and grant them varying privileges and levels of access.
   - Manage Roles helps you to define roles as a set of privileges, and assign them to users or user groups.
   - Manage API Keys helps you to authenticate with the system to perform some of the administrative tasks.
   - Email Templates helps you to create separate email templates for communicating with various stakeholders, departments, or users who have clearly defined roles in the organization.
   - Custom Styles helps you to apply styling options for the appearance of visuals and applications.
   - Custom Colors helps you to define custom color palettes for styling visuals and applications.
   - Custom Dates helps you to define custom sets of date ranges and anchor them to a specific date and time.
   - Static Assets helps you to save and manage images, CSS files, and JS files as static assets and use them across all reports to standardize the appearance and behavior of visuals and dashboards.
   - Site Settings helps you to configure site-wide settings and settings at the level of individual visuals, dashboards, and filters.
8. Help provides you with the following links:
   What's New in

   - a summary of the major new features and improvements of the latest release

   Documentation

   - the official Cloudera Data Visualization documentation

   Cloudera Support Hub

   - access to the support and professional services portal on MyCloudera

   It also displays the release number of running Cloudera Data Visualization instances.

9. The user management menu includes password change and logout options.

It also provides information on your last login date. The username displayed identifies the current user. The user icon can include small, additional icons to show if the current user is an admin or an LDAP/SSO user:

Admin user	SSO autheticated user	Admin and SSO autheticated user	Regular user

Statistics banner

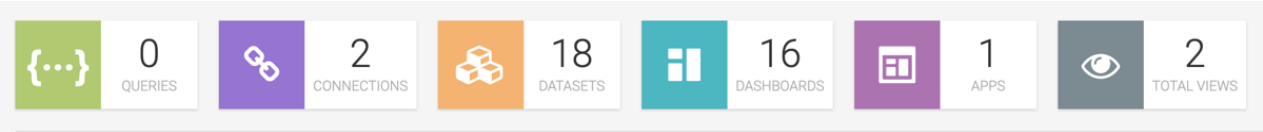

This pane shows the number of QUERIES, CONNECTIONS, DATASETS, DASHBOARDS, and APPS that you can access in your Cloudera Data Visualization instance. It also shows you the number of TOTAL VIEWS. You can click each tile to explore further details. By clicking the different tiles you can quickly access recent queries, data connections, and datasets, and can also open your their dashboards and applications.

If you click TOTAL VIEWS, you are directed to the Daily Usage tab of the Activity Log interface, where you can check the list of all activities, users, and dashboards in your Cloudera Data Visualization instance. You can adjust the time interval you want to review (the default is 30 days), and you can download the data in CSV format.

Preview area

The preview area provides quick access to the following items:

- Recent queries
- Recent connections
- Recent datasets
- Favorites
- Items you last viewed
- Items you recently created
- Overall most popular items
- Sample dashboards

You can click any of the tiles to explore the selected connection, dataset, visual, or dashboard.

The preview on connections displays tiles for the connections you recently worked with. The icons in the top-right corner of the tiles provide shortcuts to the Add Data modal and the New Dataset modal giving you an easy option to add data to a data connection or create datasets directly from the homepage.

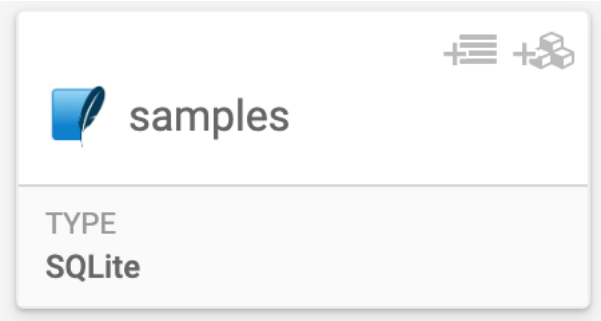

The preview on datasets displays tiles for the datasets you recently worked with. The icon in the top-right corner of the tiles provides a shortcut to creating a new dashboard on the selected dataset.

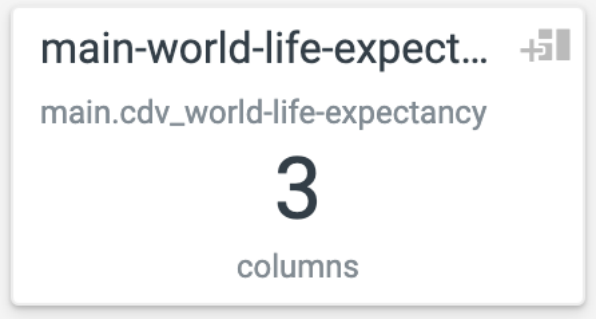

For instructions on how to add a dashboard to your favorites, see *Adding a dashboard to favorites*. Alternatively, you can access your bookmarked dashboards by clicking VISUALS on the main navigation bar and then clicking My Favorites in the side menu bar.

Homepage side menu bar

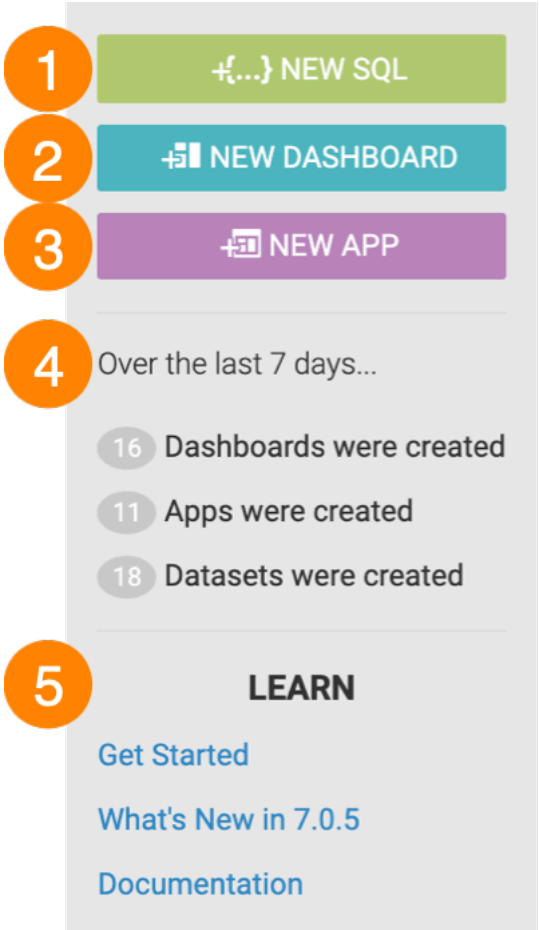

1. The NEW SQL button takes you to the Data Connection interface, where you can compose a SQL query.
2. The NEW DASHBOARD button takes you to the Dashboard Designer interface, where you can create new dashboards and visuals.
3. The NEW APP button takes you to the App Designer interface, where you can build and style custom BI applications from existing dashboards and visuals.
4. Over the last 7 days... includes statistics on how many dashboards, apps, and datasets have been created.
5. In the LEARN section, you can find the following information:
   - The Get Started link points to help content embedded in the tool.
   - The What's New in link opens a modal window showcasing new features.
   - The Documentation link opens this library.

Related Information

Visuals interface

Data interface

Cloudera Data Visualization, SQL interface

The SQL interface provides you with a SQL data query option. From the SQL tab, you can create a new dataset or dashboard from a query with a single click.

The SQL interface has the following main elements:

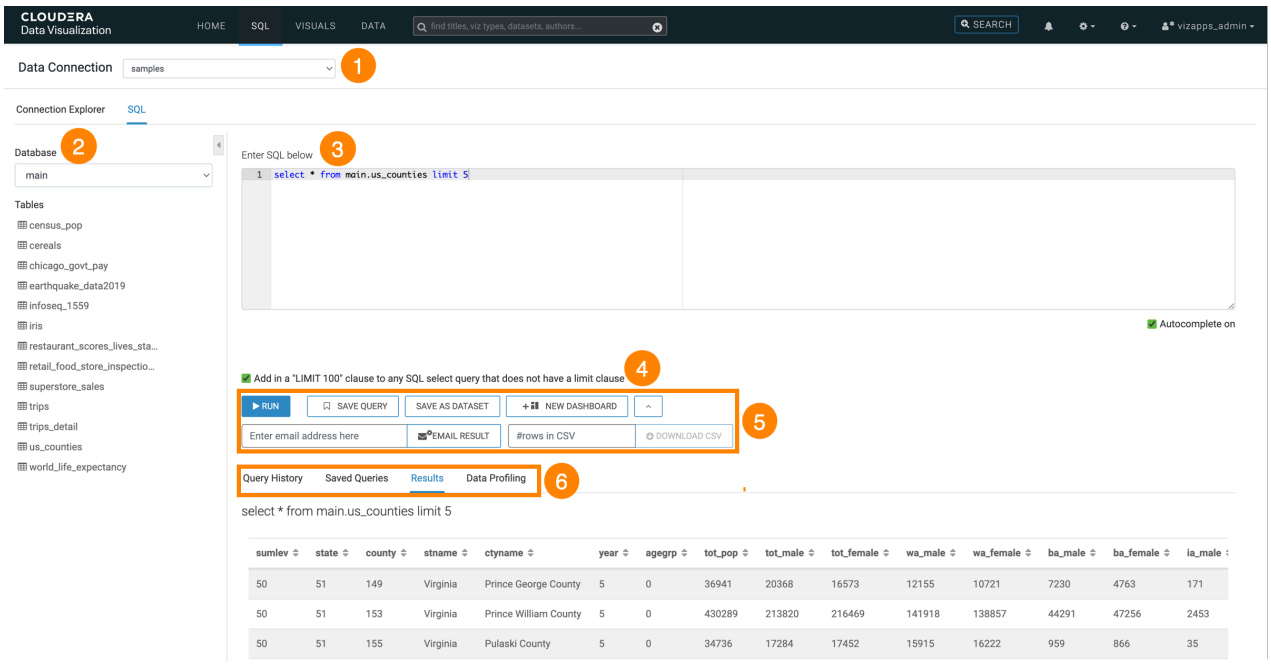

1. Data Connection – You can select any data source that is available in your project.
2. Database – On the SQL tab, you can select any database that is available on the selected data connection. When you select a database, the corresponding tables are displayed.
3. Textbox for SQL expression – You can add your SQL query here to interact with the selected dataset.
4. LIMIT 100 clause – You can automate the addition of a LIMIT 100 clause to your SQL SELECT queries when a limit is not specified in the query to retrieve a maximum of 100 rows from the database.
5. Action buttons
   - RUN – Click this button to initiate the SQL query that you have added. It allows you to quickly test and view query results. When clicked, it fetches the requested data from the dataset and displays the result in the Results tab.
   - SAVE QUERY – Click this button to save the current SQL query to the Saved Queries tab, where it can be easily accessed for future use. Saved queries are especially useful when you need to repeatedly run the same query or when collaborating with others who may need to refer to the query later.
   - SAVE AS DATASET – Click this button to save the output of the current query as a new dataset. Instead of only saving the SQL statement, this option preserves the resulting data, creating a customized dataset. This way you can preserve a specific subset of the data that can be analyzed, profiled, or visualized in the platform without needing to rerun the original query.
   - NEW DASHBOARD – Click this button to create a dashboard based on the current query output. A dashboard may contain visualizations, charts, or summary statistics that provide insights into the data.
   - Enter email address here – Enter one or more email addresses. It is used in conjunction with the EMAIL RESULT button, enabling users to send query results directly to specified email addresses. You can share query outputs with colleagues or stakeholders without having them access the interface directly.
   - EMAIL RESULT – Click this button after entering an email address to send the results of the query to the specified address.
   - #rows in CSV – Add a number to this field to specify the number of rows to be included in the CSV file when downloading the results. You can limit the data exported, which is useful when dealing with large datasets. For example, you might want to export only the first 1000 rows instead of the full dataset to reduce file size or focus on a sample.
   - DOWNLOAD CSV – Click this button to download the query results as a CSV file. The number of rows in the CSV will correspond to the value specified in the #rows in CSV field.

6. Display tabs
   - Query History – Shows a list and details of past queries.
   - Saved Queries – Shows the queries saved for repeated use.
   - Results – Shows the data returned by the query in a tabular format.
   - Data Profiling – Provides statistical summaries and insights into the dataset fields. For more information, see *Working with data profiling* .

Related Information

Working with data profiling

Cloudera Data Visualization, Visuals interface

The VISUALS interface provides you access to visualization features. You can initiate a number of operations from here when working with visuals.

The interface has the following main parts:

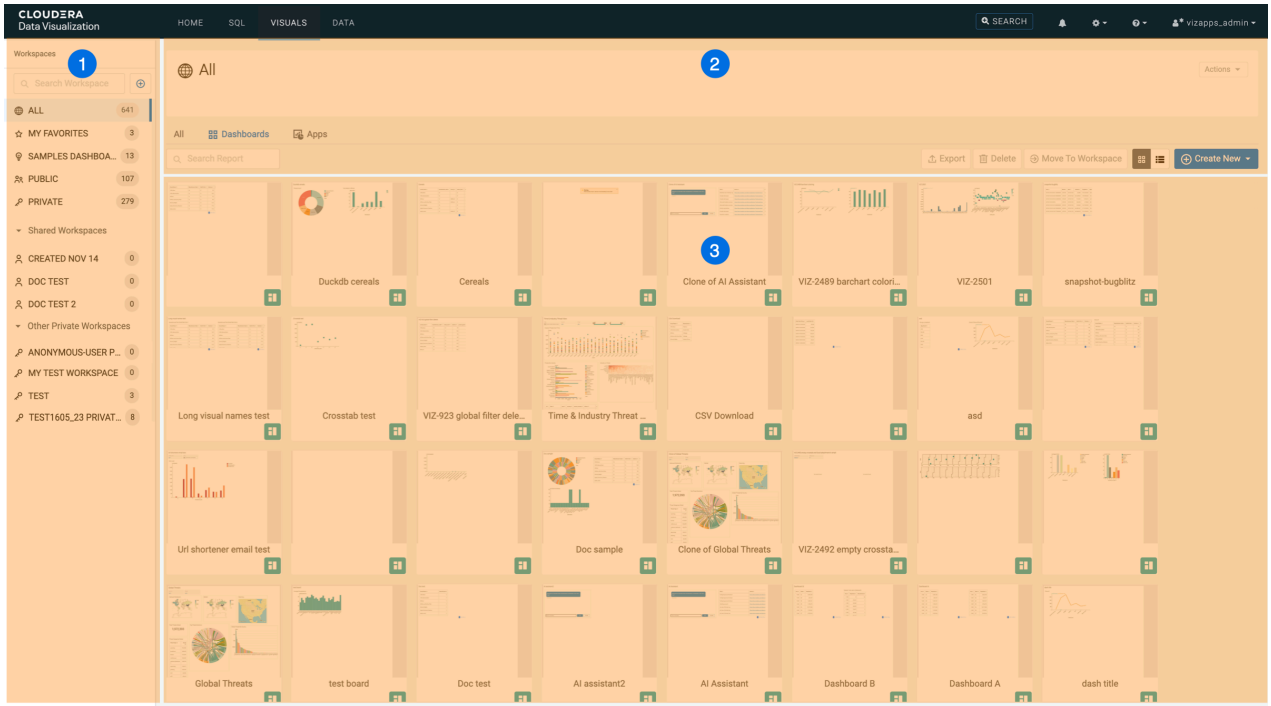

1. VISUALS side menu bar

   The sidebar displays the list of workspaces available in the environment. By default, Private and Public workspaces are available. You have the option to create custom workspaces, if needed. Once created, these

custom workspaces are automatically added to this side menu bar. The number next to each workspace shows you how many dashboards are available in it.

Shared and Private workspaces are sorted and the lists can be collapsed for easier navigation.

- You can search among the workspaces using the search bar.
- You can create a workspace by clicking the plus sign next to Search Workspace.
- You can edit an existing workspace by clicking located next to the name of the workspace. The icon appears when you hover over the workspace name.

  Note: You can see only if you have permission to edit the workspace.

  For more information, see *Editing a workspace*.
- Using the side menu bar, you can access standard groupings of visual artifacts by clicking the following options:
  - All
  - My Favorites
  - Sample Dashboards
  - Public
  - Private

  Note: Sample Dashboards provides access to a variety of example dashboards available within Cloudera Data Visualization. Get started by exploring these samples to see how they use a combination of visuals and filters to provide business insights.

2. VISUALS actions
   - The Actions button in the top-right corner of the page allows you to edit or delete the workspace you are viewing.

     Note: The button is only active if you have permissions to edit and/or delete the workspace.

     For more information, see *Editing a workspace* and *Deleting a workspace*.
   - The All, Dashboards, and Apps tabs allow you to switch between views displaying all visual artifacts, only dashboards, or only apps in the preview area.
   - Search Report allows you to search among the artifacts of the workspace.
   - The Export button allows you to export visual artifacts to your default download folder.

     Note: The button is only active if you select at least one artifact in the preview area.

     For more information about this function, see *Exporting and importing visual artifacts*.
   - The Delete button deletes the selected items.

     Note: The button is only active if you select at least one artifact in the preview area.
   - The Move To Workspace button helps you to move the selected items to a new workspace.

     Note: The button is only active if you select at least one artifact in the preview area.

     For more information, see *Moving dashboards between workspaces* and *Moving apps between workspaces*.
   - The two icons allow you to switch between Grid View and List View. Grid View shows thumbnail previews of your visuals, dashboards, and applications, while List View presents the same visual artifacts vertically arranged in a single, multi-line list.
   - The Create New button allows you to start a new dashboard or app.
3. VISUALS preview area

   All visuals artifacts of the selected workspace are displayed here in a grid or list view.

VISUALS interface – Grid view

Grid view shows you thumbnail previews of your visuals, dashboards, and applications.

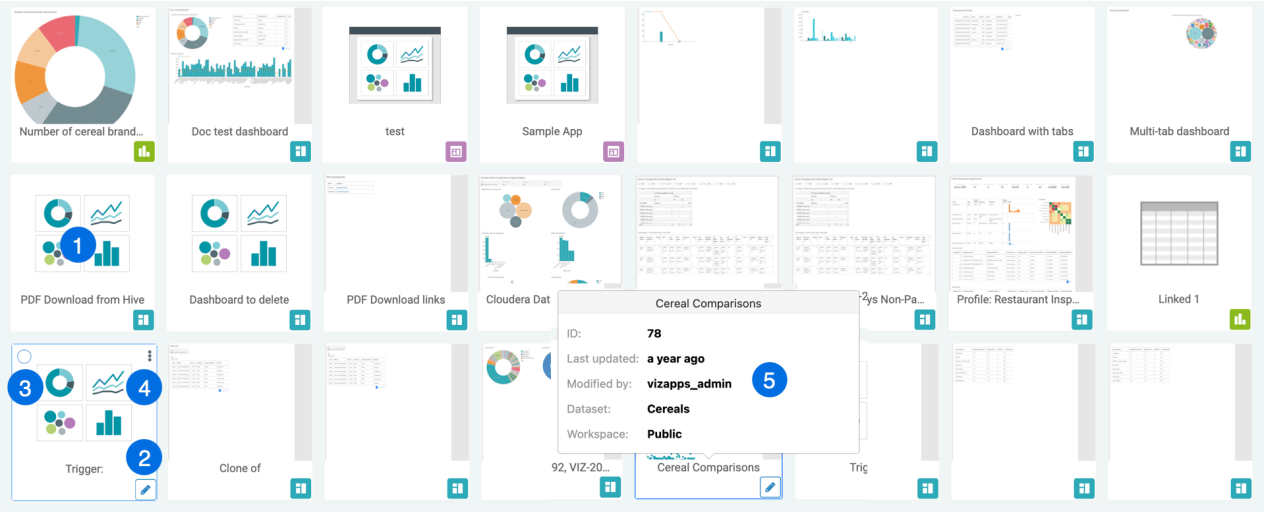

1. You can open a visual artifact in View mode by clicking the tile.

   Note: The icon in the bottom right corner of the tile shows you if the visual artifact is a dashboard (teal), a visual (green), or an app (purple).

2. You can open a visual artifact in Edit mode by hovering over the tile and clicking in the bottom right corner.
3. You can select visuals, dashboards, or apps, if you hover over a tile and click the radio button in the the top left corner. You can deselect the selected artifact by clicking the circle again.

   Note: When you select one or more items, the action buttons available in the top right corner of the preview area become active.

4. You can access the Supplemental menu of a visual artifact by hovering over the tile and clicking . You can use this menu to export the visual artifact, move it to a different workspace, or delete it.
5. You can access more information about the selected visual artifact if you hover over the name displayed on the tile.

VISUALS interface – List view

List view presents the visual artifacts of a workspace arranged vertically in a single, multi-line list. It shows the title and the ID of the visuals, dashboards, and apps. It also shows the number of related dashboards/linked visuals and descendant dashboards.

It provides further metadata about each visual artifact including creation date, date of last update, name of the modifier, total views and the corresponding dataset and workspace.

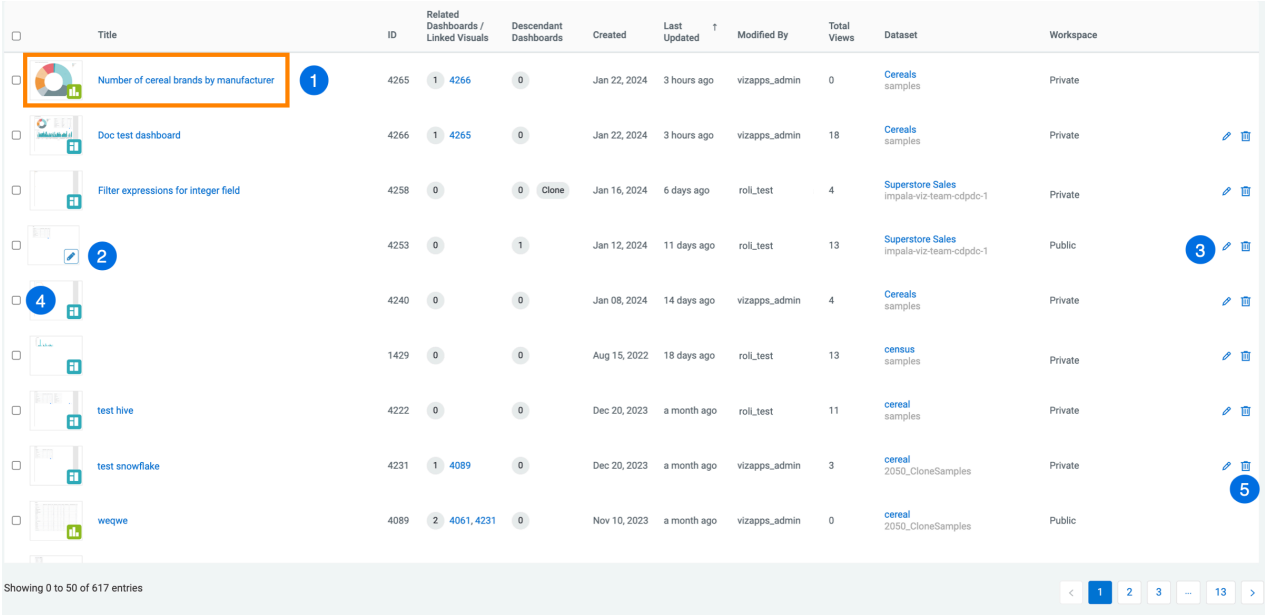

1. You can open a visual artifact in View mode if you click the reduced size image or the title of the visual artifact.
2. You can open a visual artifact in Edit mode if you hover over the reduced size image of the artifact and click in the bottom right corner.
3. You can also click at the end of the row if you want to edit a visual artifact.
4. You can select a visual artifact if you mark the check box of the item. You can deselect the selected artifact by clicking the checkbox again.

   Note: When you select one or more items, the action buttonss available in the top right corner of the preview area become active.

5. You can delete a single visual artifact by clicking at the end of its row.

Related Information

Editing a workspace
Deleting a workspace
Exporting and importing visual artifacts
Moving dashboards between workspaces
Moving apps between workspaces
Homepage
Data interface

Cloudera Data Visualization, Data interface

The information and functionalities available on the DATA interface of Cloudera Data Visualization enable you to manage the data you need for your visualizations and to initiate a number of operations when working with datasets and data extracts.

The DATA interface shows your existing data connections. You can also add new connections to data sources, create new datasets, and work with your data tables and extracts. The interface has the following main parts:

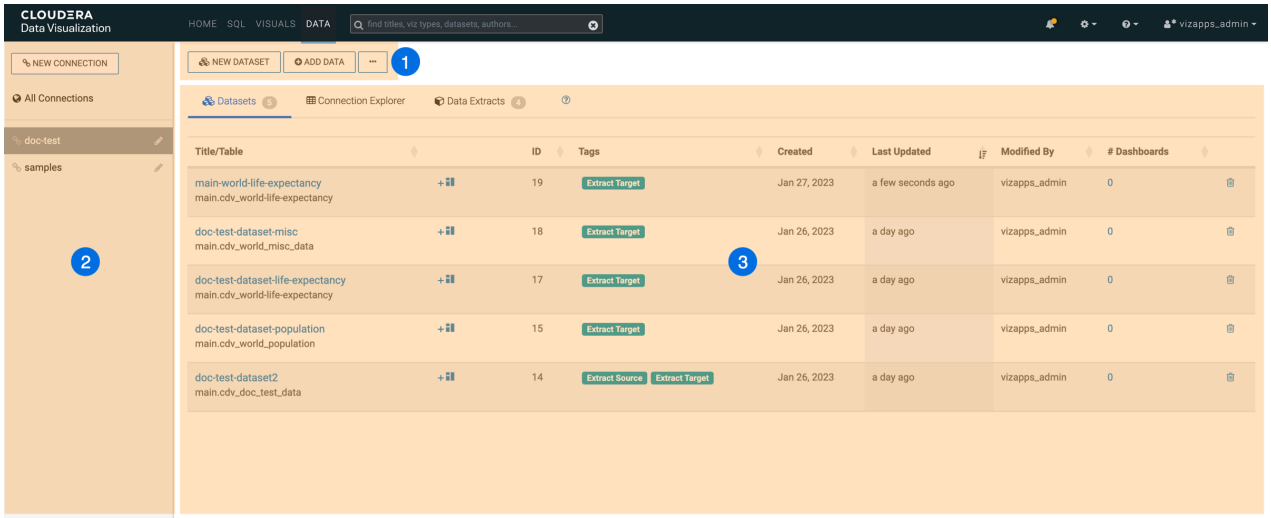

1. DATA interface top menu bar
2. DATA interface side menu
3. DATA interface preview area

DATA top menu bar

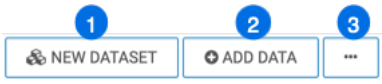

1. The NEW DATASET button is for creating datasets. Datasets are needed for developing dashboards and visuals. You can also start a new dataset from a specified table.
2. For SQLite connections, the ADD DATA button enables you to introduce data from outside sources to enrich your datasets.

3. The Supplemental menu provides access to additional functions:
   - Clear result cache forces a reload of the full table definition.
   - Import Visual Artifacts enables you to restore or import visual artifacts from a backup JSON file.
   - Direct Access lets you access data on the connections directly and build datasets from specific SQL queries, as opposed to starting with an existing table.

DATA side menu

1. The NEW CONNECTION button is for connecting to a new data source.
2. You can list all your datasets by clicking All Connections.
3. samples is the SQLite connection that contains the example datasets in Cloudera Data Visualization.

DATA preview area

The data preview area has three tabs: Datasets, Connection Explorer, and Data Extracts.

In the Datasets view, you can see all existing datasets on a connection available for creating visuals.

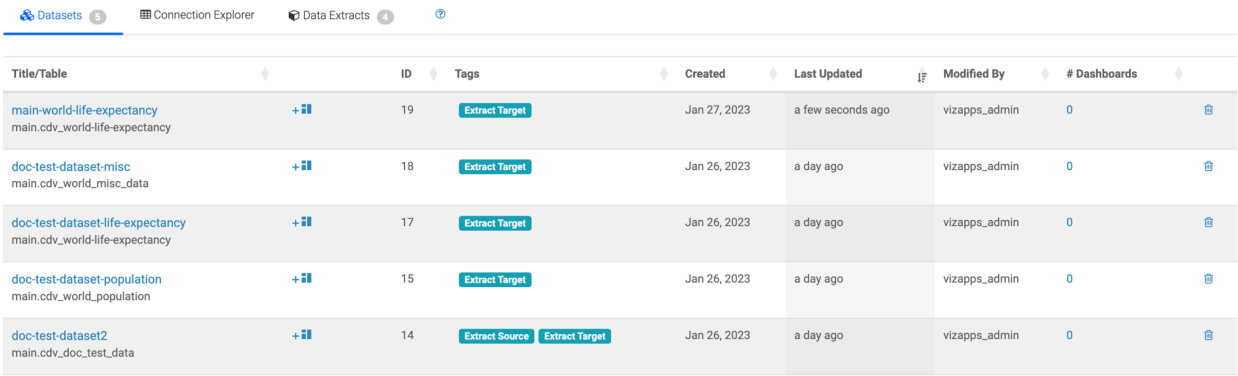

In the Connection Explorer view, you can manage functions related to the data on a connection. You can check the databases and preview table details in the data connection.

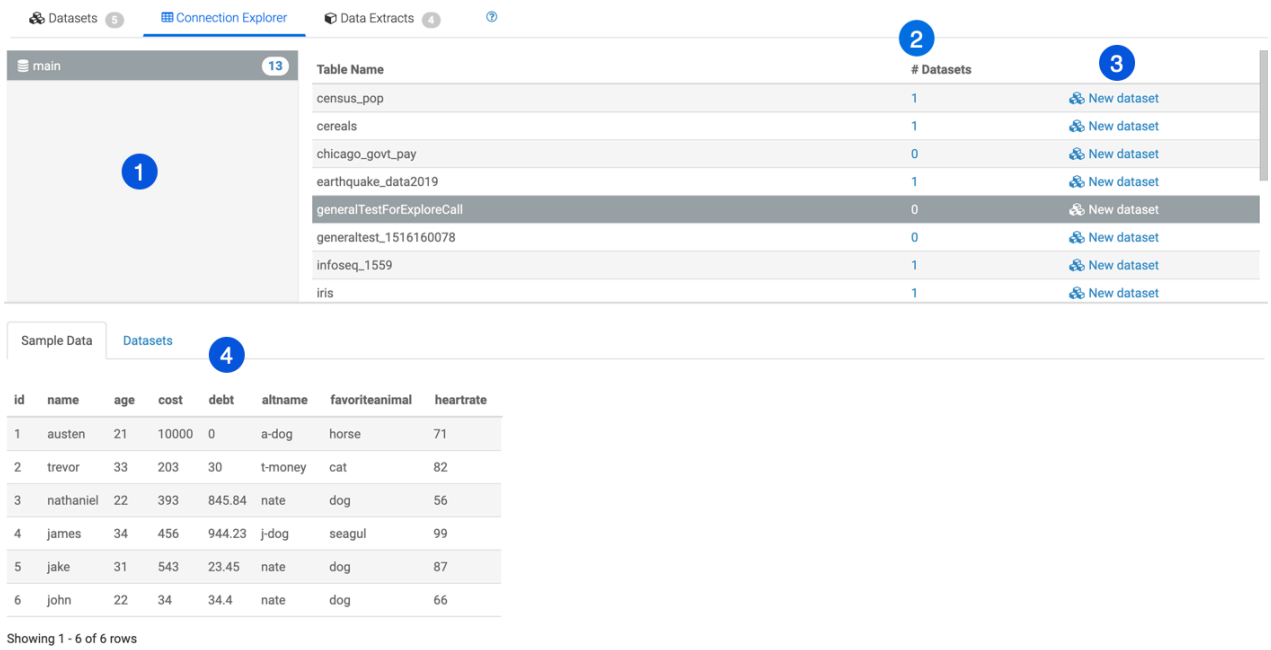

1. The database area shows all databases that you can access through the current connection. Selecting a database displays its tables.
2. The Datasets column lists the number of datasets that use the particular table as their initial definition.
3. You can click the New dataset link to create a dataset on a specific table.
4. To see more information about data tables, you can click the row of each table. The additional information is displayed under two tabs: Sample Data and Datasets.

In the Data Extracts view, you can see all existing data extracts on a connection.

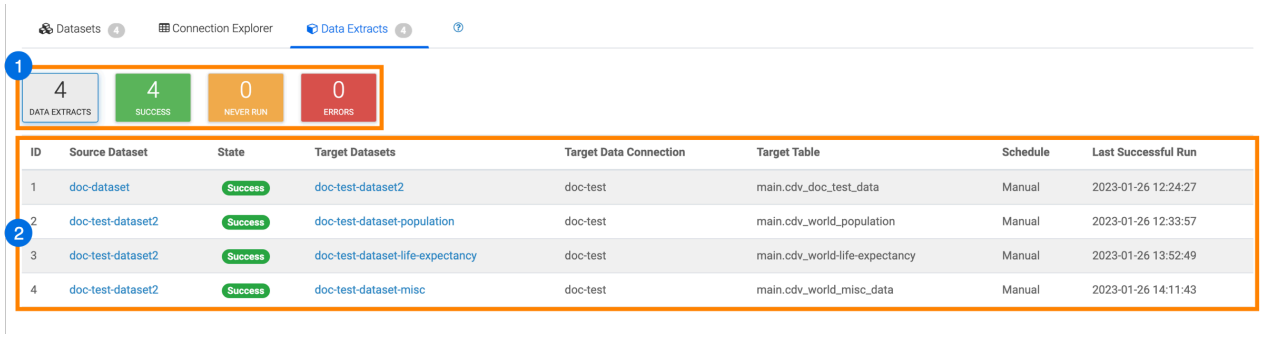

1. Overall statistics
2. List of Data Extracts sourced from the current data connection

Related Information

Custom homepages

Homepage

Visuals interface

Adding a dashboard to favorites

Cloudera Data Visualization 7.2.9

Publishing Datasets

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring dataset recipient roles....4

Publishing datasets.... 5

Configuring the dataset publisher role.... 8

Defining groups for teams and access levels....13

Assigning groups to roles....14

Assigning users to groups....17

Configuring dataset recipient roles

This document outlines the steps for setting up Dataset Recipient roles, providing members with access to datasets at various levels.

About this task

Possible levels are:

- Manage dataset
- Manage dashboards
- View dashboards

For more information on possible permissions, see *RBAC permissions.*

Within this context, three distinct dataset roles—Data Admin, Analyst, and Visual Consumer—are established for each of the three organization groups: Marketing, Sales, and Operations.

> **Restriction:**
>
> This task is for users with Manage roles and users privileges.

The steps for creating the role Marketing Data Admin:

Procedure

1. On the main navigation bar, click the Gear icon, and select Manage Roles.
2. In the Roles interface, start the new role by clicking New Role.
3. In the Role definition interface, enter the following information:
   - Name: Marketing Data Admin
   - Description: Creates datasets, can grant access to their datasets.

   Note: Privileges and membership will be defined later.

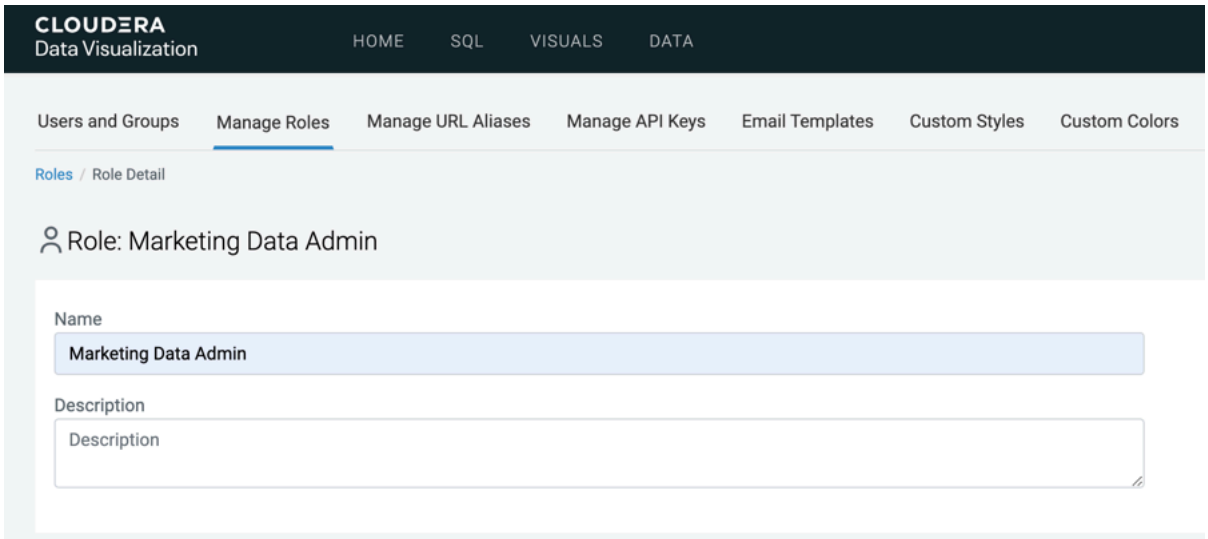

4. Click SAVE.
5. Cloudera Data Visualization generates a success message.

6. Repeat the previous steps to generate a total of 9 (nine) new roles, with the following names and descriptions:

Table 1: Roles that receive access to new datasets

Role Name	Description
Marketing Data Admin	Creates datasets, can grant access to their datasets.
Marketing Analyst	Can create dashboards.
Marketing Visual Consumer	Can view dashboards.
Sales Data Admin	Creates datasets, can grant access to their datasets.
Sales Analyst	Can create dashboards.
Sales Visual Consumer	Can view dashboards.
Operations Data Admin	Creates datasets, can grant access to their datasets.
Operations Analyst	Can create dashboards.
Operations Visual Consumer	Can view dashboards.

Results

After completing these steps, the list of roles should appear similar to this:

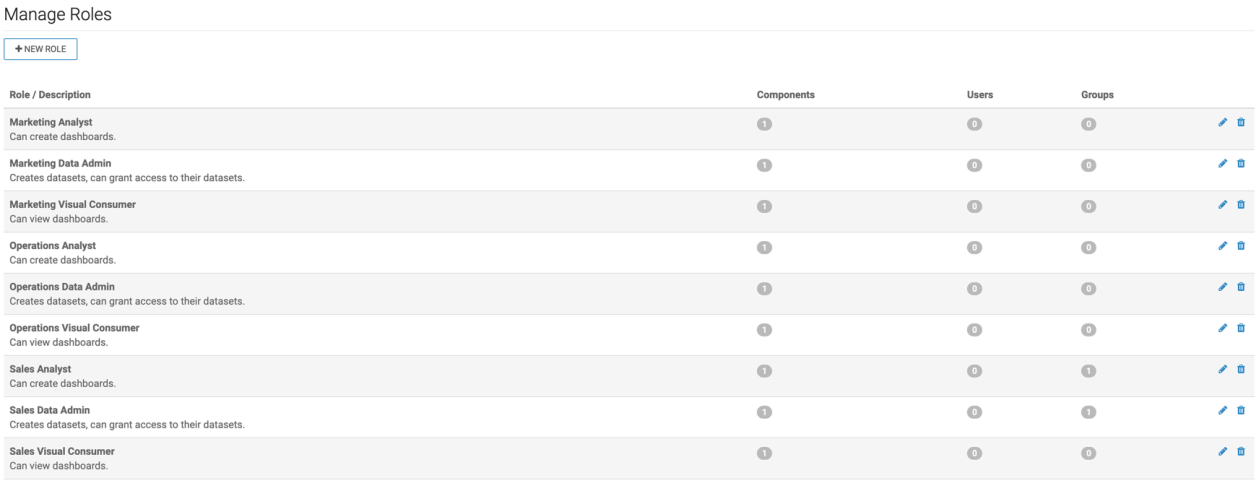

Related Information

RBAC permissions

Publishing datasets

When a user creates a new dataset, that dataset is private by default. To share, user must explicitly grant others access to the new dataset.

About this task

In principal, anyone with Manage roles and users privileges can grant dataset access to others. However, it is better to choose to restrict this privilege to a small number of people in the organization, typically system administrators. This is because role editing is very powerful and role editors have full access to the entire system and can update any role to grant any permission to any user.

As a result, most dataset creators do not ordinarily have the Manage roles and users permission, and must place a service request so that system administrators add new datasets to the desired roles. In large organizations with

multiple teams that consist of users with different responsibilities (dataset creators, dashboard creators, and dashboard viewers), access management becomes a laborious task.

Our role-based access control (RBAC) security approach gives dataset creators an opportunity to 'publish' or share their datasets through a set of grants based on roles. They can grant access to specific roles though the Grant manage dataset, Grant manage dashboards, and Grant view dashboards permissions.

The following workflow is established to ensure that dataset creators can grant access to their datasets without having to rely on system administrators.

Note: Only users with Manage roles and users role privileges, typically system administrators, can set up the roles and permissions, and define users and user groups.

Before a dataset creator, Data Admin, can publish a dataset, they must have appropriate RBAC permissions, and the roles have to be assigned correctly to the members of each team. To understand how to configure the required permissions, roles, groups, and users, read RBAC setup for dataset publishing, which contains the following topics:

- *Setting the dataset recipient roles*
- *Setting the dataset publisher role*
- *Defining groups for teams and access levels*
- *Assigning groups to roles*
- *Assign users to groups*

After making all necessary adjustments to the dataset, a user with a Data Admin role can publish the dataset.

Restriction:

This workflow is intended for users with Data Admin role privileges.

This feature is only available after completing RBAC setup for dataset publishing.

For more information on possible permissions, see *RBAC permissions.*

Follow these steps to publish a dataset to the intended users:

Procedure

1. Log in as a user with Data Admin privileges. In this example, you log in as user John_Marketing.
2. Navigate to the dataset you plan to publish. In this example, you plan to publish the Modified Iris Data dataset.
3. Click the Permissions option in the left navigation.

   At this stage, the Permissions interface for the Modified Iris Data dataset appears. In this example, it contains the default matrix for roles and privileges for the three Data Admin roles for All datasets (Manage dataset, Manage dashboards, and View dashboards).

   You can publish the dataset through one of the following actions:

Option	
Publish	Makes the dataset publicly visible to the already enabled roles. You can stop here.
Edit Permissions	Allows the user to grant additional access permissions on the dataset. This is only available to users with Role permissions. If you choose this option, continue to the next step.

4. Click Edit Permissions.

   The full Permissions matrix appears.

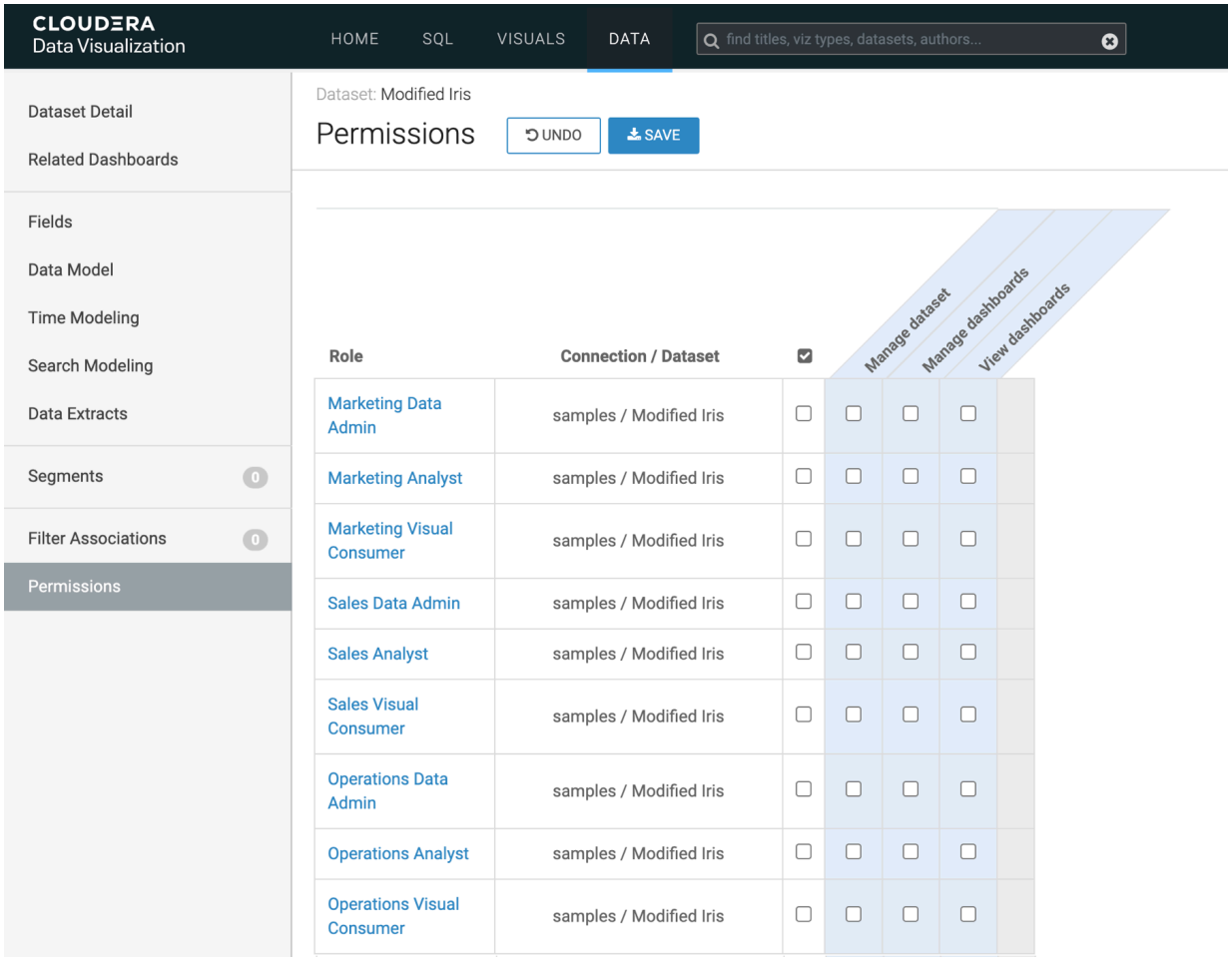

5. To grant the available privileges to the Modified Iris Data to the Marketing group, click the check boxes for the appropriate grants for the Data Admin, Analyst, and Visual Consumer roles defined in the Marketing group.

   In this example, you granted all possible privileges. However, it is feasible that you may initially select only the Data Admin role, to involve others who have that role in testing the dataset and building preliminary dashboards. You can subsequently use this interface to grant access to other roles and users, such as Analysts who build more sophisticated dashboards, and postpone Visual Consumer access to a later time.

6. Click SAVE.

   This action makes the dataset available to all specified roles.

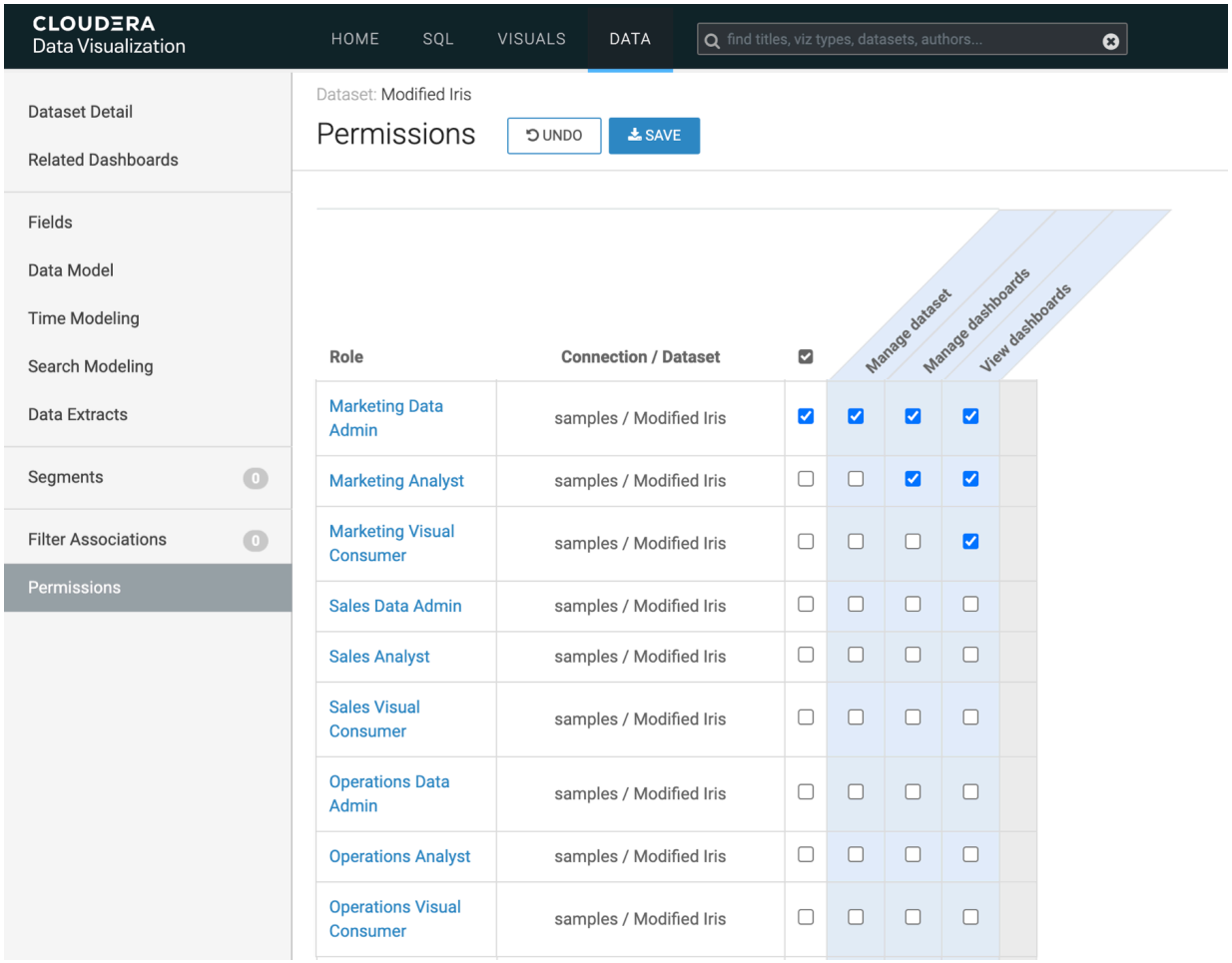

Related Information

RBAC permissions
RBAC setup for dataset publishing
Configuring dataset recipient roles
Configuring the dataset publisher role
Defining groups for teams and access levels
Assigning groups to roles
Assigning users to groups

Configuring the dataset publisher role

This guide walks through the setup process for the Dataset Publisher role.

About this task

In this example, the members assigned to this role possess the ability to grant dataset access to other users, which can be categorized into three levels:

- Grant manage dataset
- Grant manage dashboards

- Grant view dashboards

The granter must have the Grant manage dataset permissions on the target dataset.

In an earlier section, three (3) dataset publishing roles (data admins) in *Setting the dataset recipient roles*, one each for the three organization groups: Marketing, Sales, and Operations were established. These are Marketing Data Admin, Sales Data Admin, and Operations Data Admin, respectively. The roles for the recipients of the grants, the analysts and the visual consumers were also created.

For more information on possible permissions, see *RBAC permissions.*

Follow these steps to specify the role Marketing Data Admin:

Procedure

1. In the Roles interface, click either the name of the role, Marketing Data Admin, or the Edit icon on its role.
2. In the Role Detail interface, click ADD ROLES.

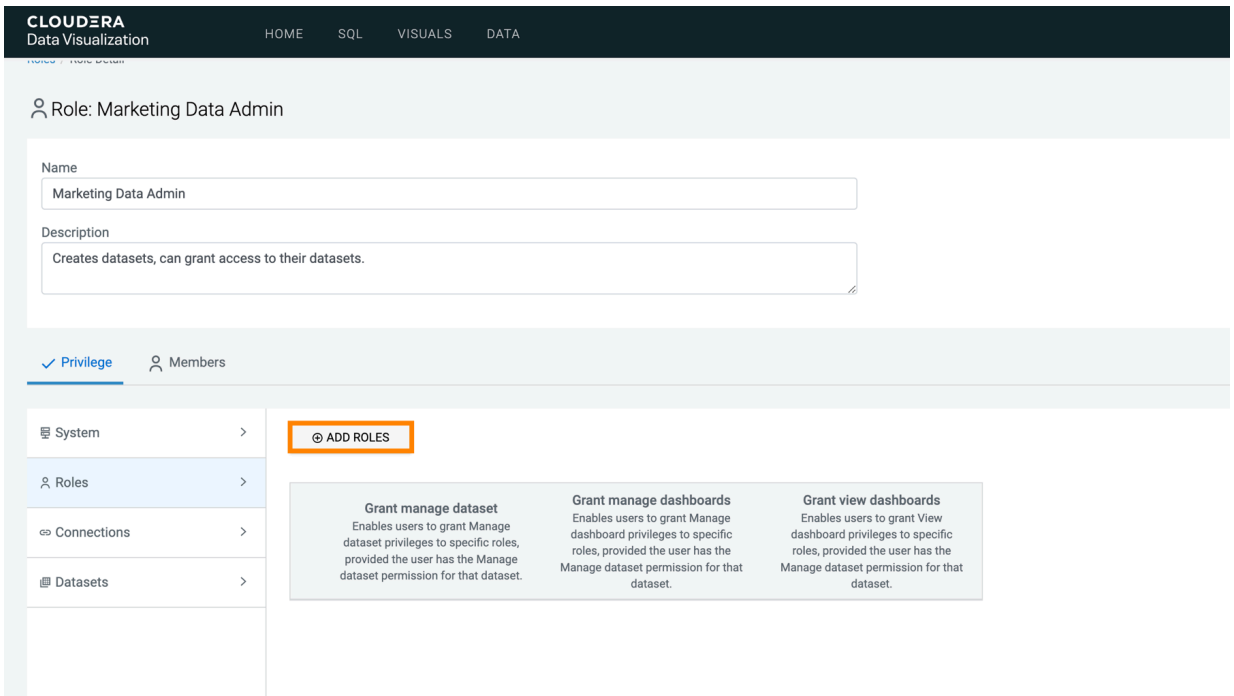

   The Add Role Privilege modal window appears.

3. From Roles, select Marketing Data Admin, a role defined previously.

   Note that this enables individuals with the Marketing Data Admin role to grant privileges to others with the same role.

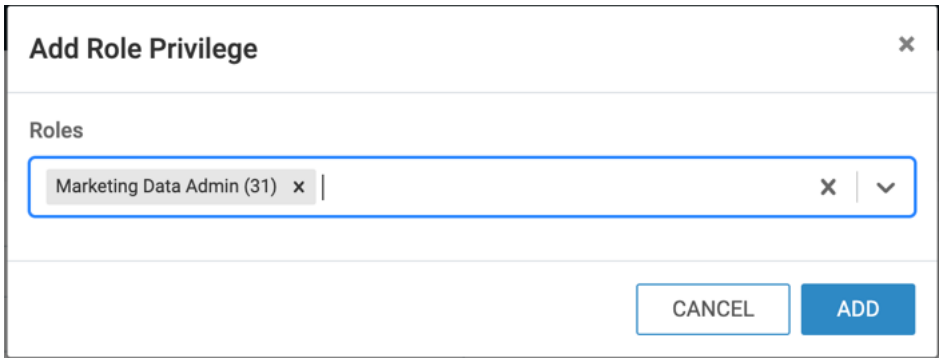

4. Click ADD.

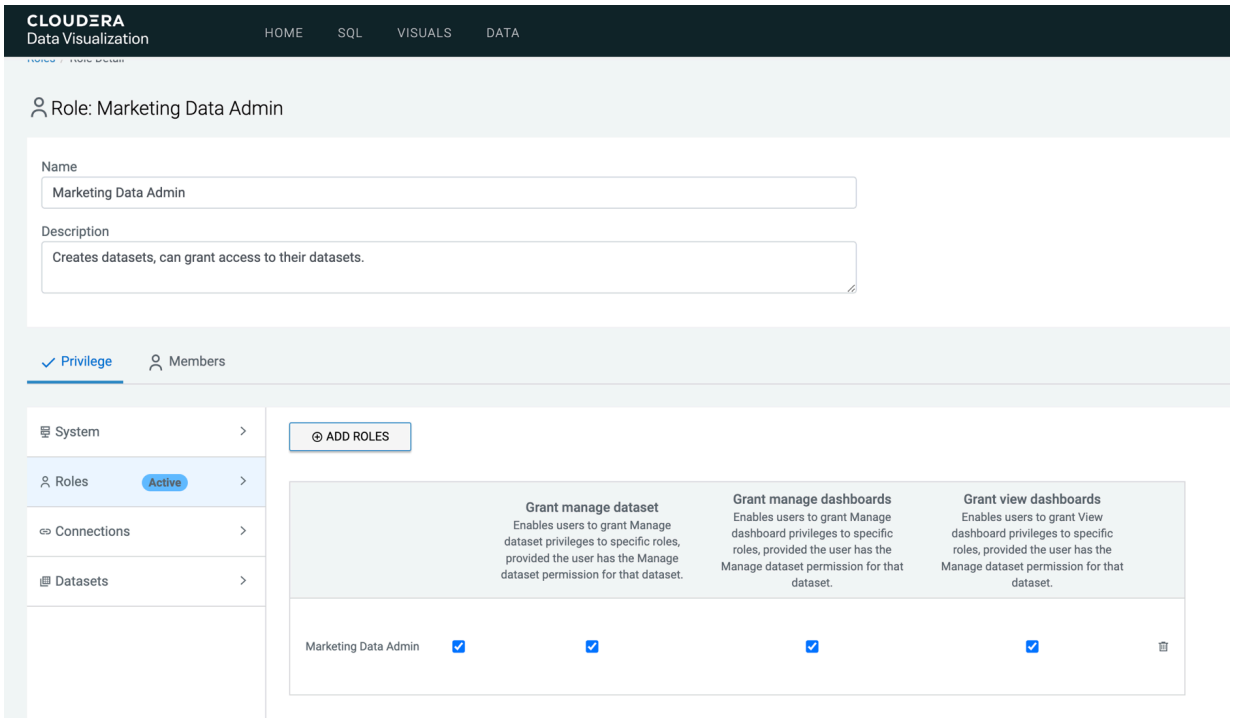

5. Leave the added privilege with all options on, and click ADD ROLES again to add the next privilege.
6. From Roles, select Marketing Analyst, a role defined previously.
7. Click ADD.
8. Deselect the Grant manage dataset option for this privilege.

   This is because analysts should not be altering the dataset.

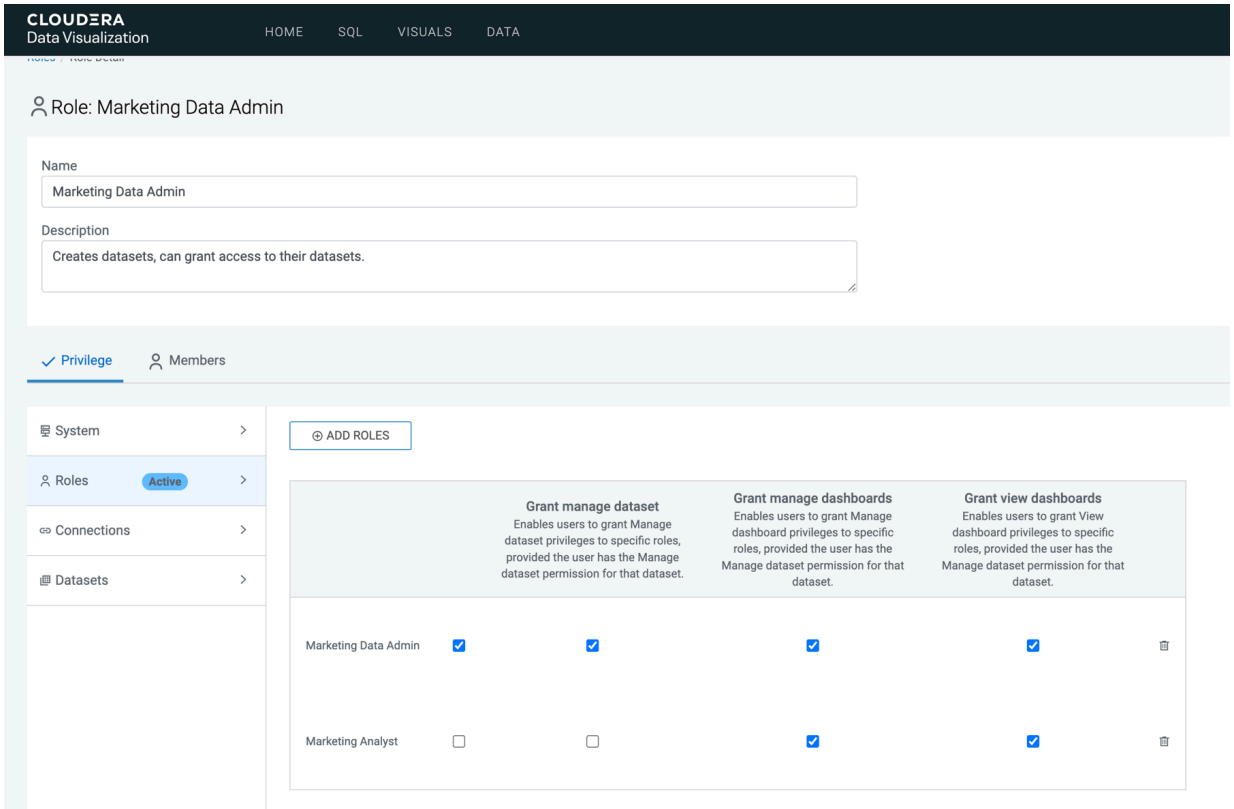

9. Click ADD ROLES to add the next privilege.
10. From Roles, select Marketing Visual Consumer, a role defined previously, then click ADD.
11. Unselect the Grant manage dataset and Grant manage dashboards options for this privilege. This is because visual consumers can only view dashboards, and should not be altering the dataset or creating new dashboards. Therefore, this line contains only the Grant view dashboards option.
12. Click the Connections tab.
13. Click ADD CONNECTIONS to add the next privilege to create the connection-level privileges necessary for the Marketing Data Admin user. This particular privilege enables access to all data available through the specified connection.
14. From Connections, select impala.
15. Click ADD.

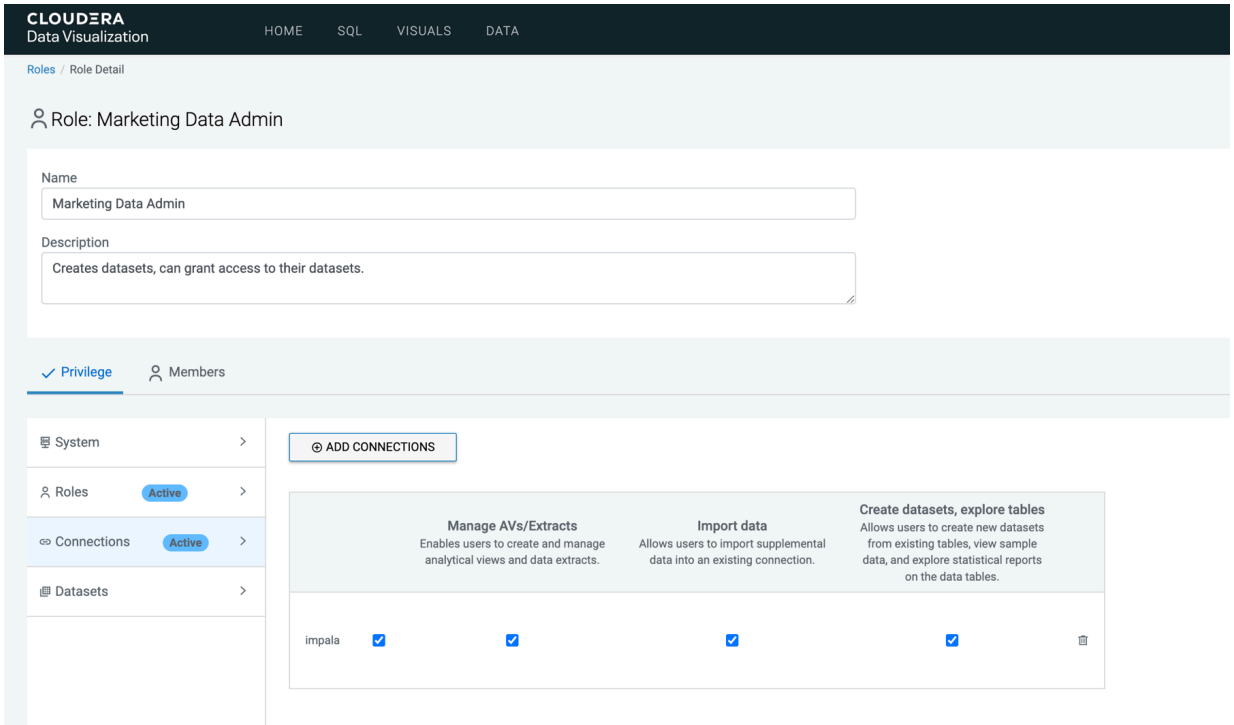

16. Leave this privilege with all options on (Manage AVs/Extracts, Import data, and Create datasets, explore tables).
17. Click the Datasets tab.
18. Click ADD DATASETS to add the next privilege.

    Finally, you declare dataset-level privileges. You can choose previously declared connections and limit access to a few of its datasets, or select a new connection, or choose all datasets on a connection. Here, you simply let members of the Marketing Data Admin role access all existing datasets on the Impala connection.
19. From Connections, select impala.
20. From Datasets, select All datasets.

21. Click ADD.

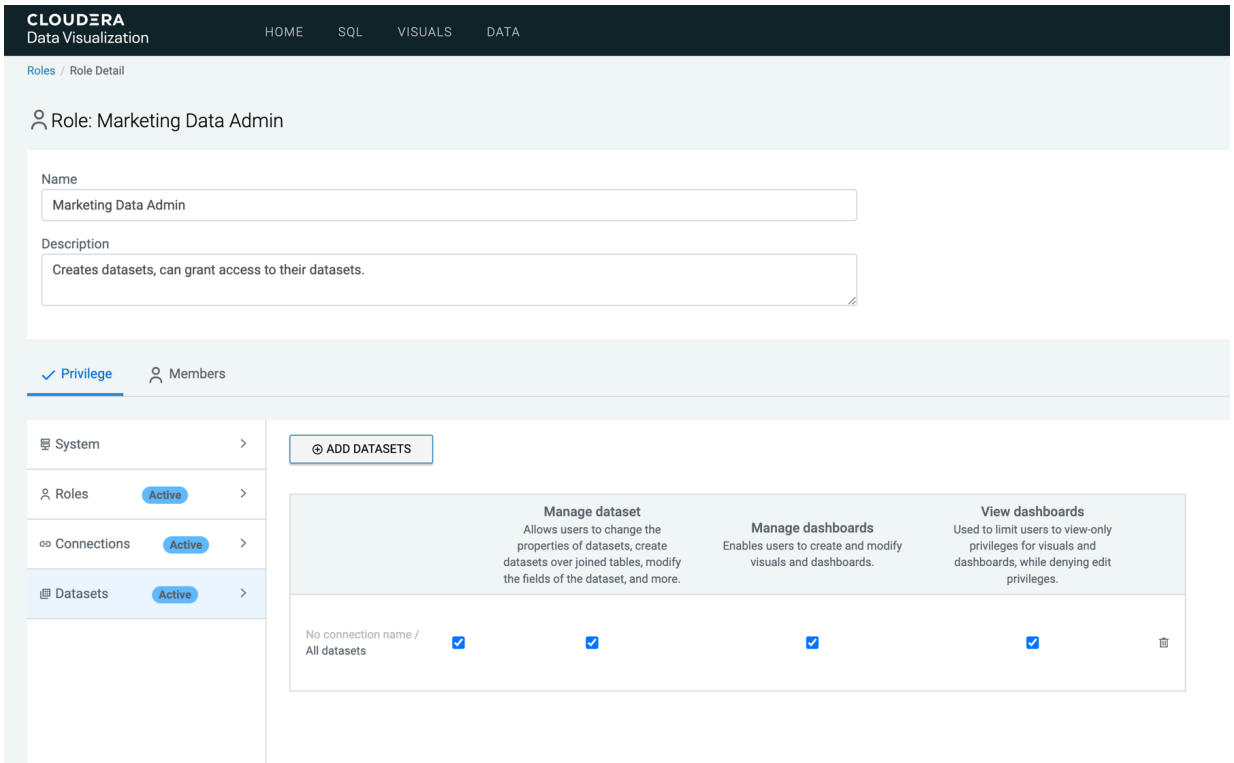

22. Leave this privilege with all options on (Manage dataset, Manage dashboards, and View dashboards)

23. Click APPLY CHANGES at the bottom of the page to save the new definitions for Marketing Data Admin.

24. Repeat the previous steps to add privileges to the roles of the Sales Data Admin and Operations Data Admin. Each of these roles should have a privilege matrix that is similar to privileges assigned to the Marketing Data Admin role.

Table 2: Privileges assigned to the Data Admin roles

Role Name	Component	Permission
Marketing Data Admin	Role: Marketing Data Admin	• Grant manage dataset • Grant manage dashboards • Grant view dashboards
	Role: Marketing Analyst	• Grant manage dashboards • Grant view dashboards
	Role: Marketing Visual Consumer	• Grant view dashboards
	Connection: Arcadia Enterprise	• Manage analytical views • Import data • Create datasets, explore tables
	Dataset: Arcadia Enterprise, All Datasets	• Manage dataset • Manage dashboards • View dashboards
Sales Data Admin	Role: Sales Data Admin	• Grant manage dataset • Grant manage dashboards • Grant view dashboards
	Role: Sales Analyst	• Grant manage dashboards • Grant view dashboards

Role Name	Component	Permission
	Role: Sales Visual Consumer	• Grant view dashboards
	Connection: Arcadia Enterprise	• Manage analytical views • Import data • Create datasets, explore tables
	Dataset: Arcadia Enterprise, All Datasets	• Manage dataset • Manage dashboards • View dashboards
Operations Data Admin	Role: Operations Data Admin	• Grant manage dataset • Grant manage dashboards • Grant view dashboards
	Role: Operations Analyst	• Grant manage dashboards • Grant view dashboards
	Role: Operations Visual Consumer	• Grant view dashboards
	Connection: Arcadia Enterprise	• Manage analytical views • Import data • Create datasets, explore tables
	Dataset: Arcadia Enterprise, All Datasets	• Manage dataset • Manage dashboards • View dashboards

Related Information

Configuring dataset recipient roles
RBAC permissions

Defining groups for teams and access levels

Procedure

1. Create the groups of users by organizational team, and by level of required access.
   - There are three distinct access levels across the organization: Data Admins, Analysts, and Visual Consumers.
   - There are three teams: Marketing, Sales, and Operations.

   The following steps demonstrate how to create the necessary user groups in Cloudera Data Visualization.

   Note: Depending on your architecture, this definition may happen at the LDAP layer, with the groups visible to Cloudera Data Visualization.

2. On the main navigation bar, click the Gear icon and select Users & Groups.
3. In the Manage Users & Groups interface, click New Group.
4. In the New Group window modal, create a new group Data_Admins.
5. You may choose to add users to the group at this point, or do this at a later stage.
6. Click SAVE.
7. Repeat these steps to create additional new user groups: Analysts, Visual_Consumers, Marketing, Sales, and Operations.

Assigning groups to roles

About this task

- There are three distinct access levels across the organization: Data Admins, Analysts, and Visual Consumers.
- There are three teams: Marketing, Sales, and Operations.

The following steps demonstrate how to create the necessary user groups in Cloudera Data Visualization. Note that depending on your architecture, this definition may happen at the LDAP layer, with groups visible to Cloudera Data Visualization.

Procedure

1. On the main navigation bar, click the Gear icon, and select Manage Roles.
2. In the Roles interface, click either the name of the role, Marketing Data Admin, or the Edit icon on its row.
3. In the Role Detail interface, select the Members tab.
4. In Groups section, click EDIT GROUP(S).

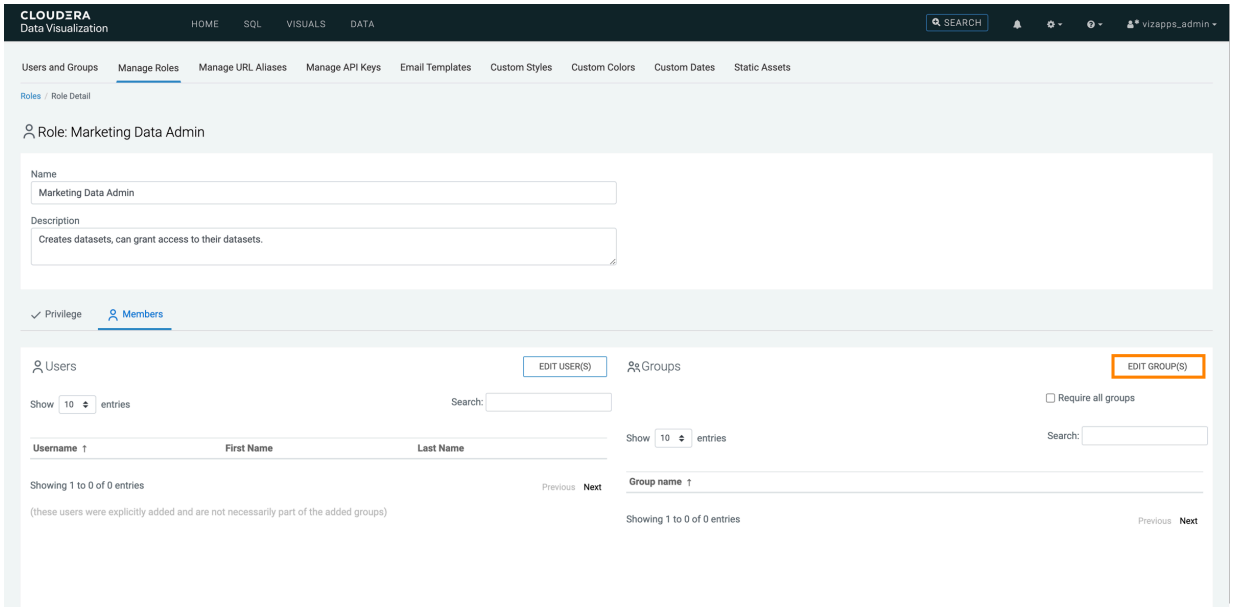

The Role Assignment modal window appears.

5. Select the groups Data_Admins and Marketing.

6. Click ADD.

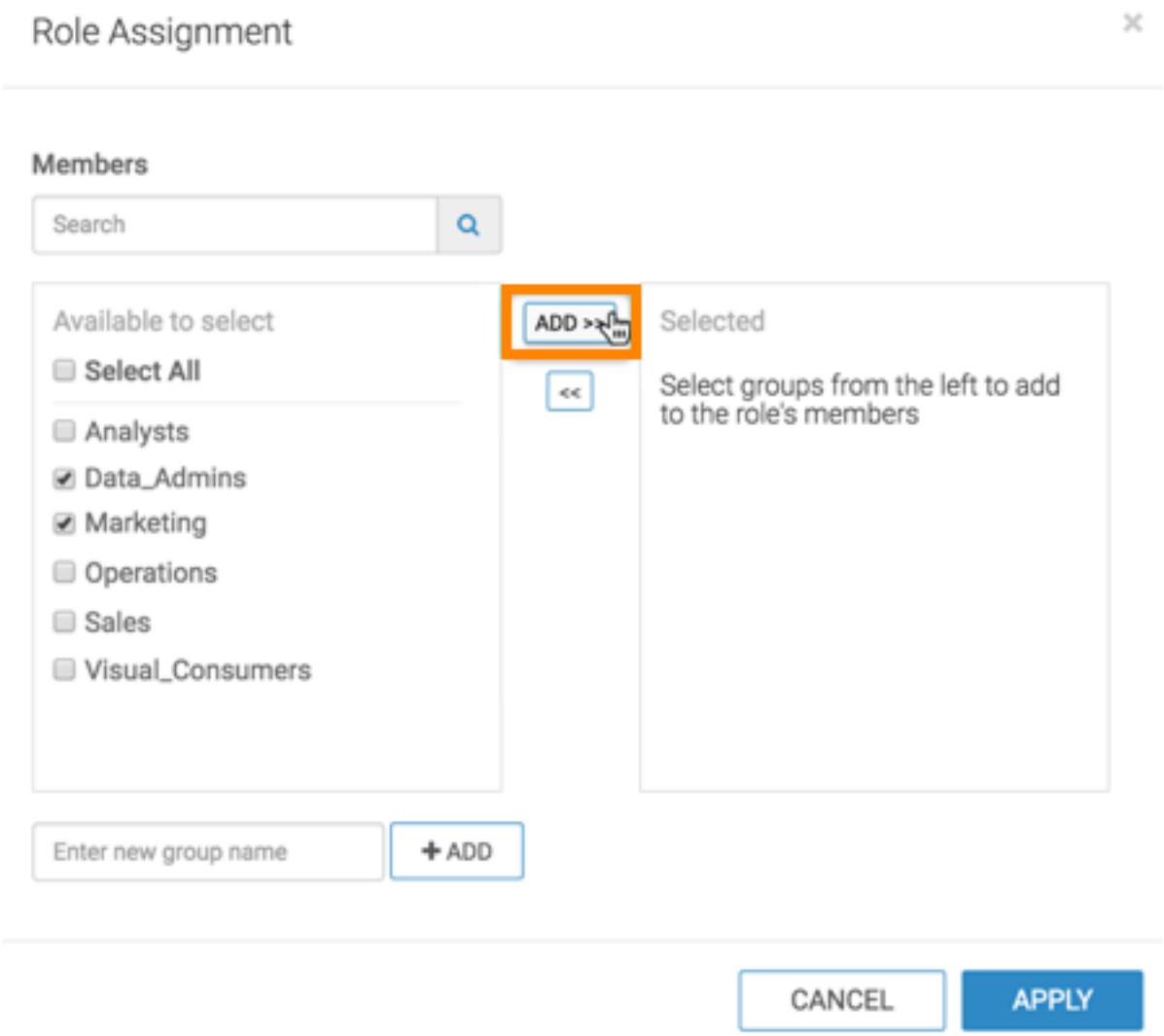

7. Click APPLY.

   The Groups list now contains Data_Admins and Marketing.

8. Select the Require all groups option.

9. Click SAVE.

A confirmation message appears briefly.

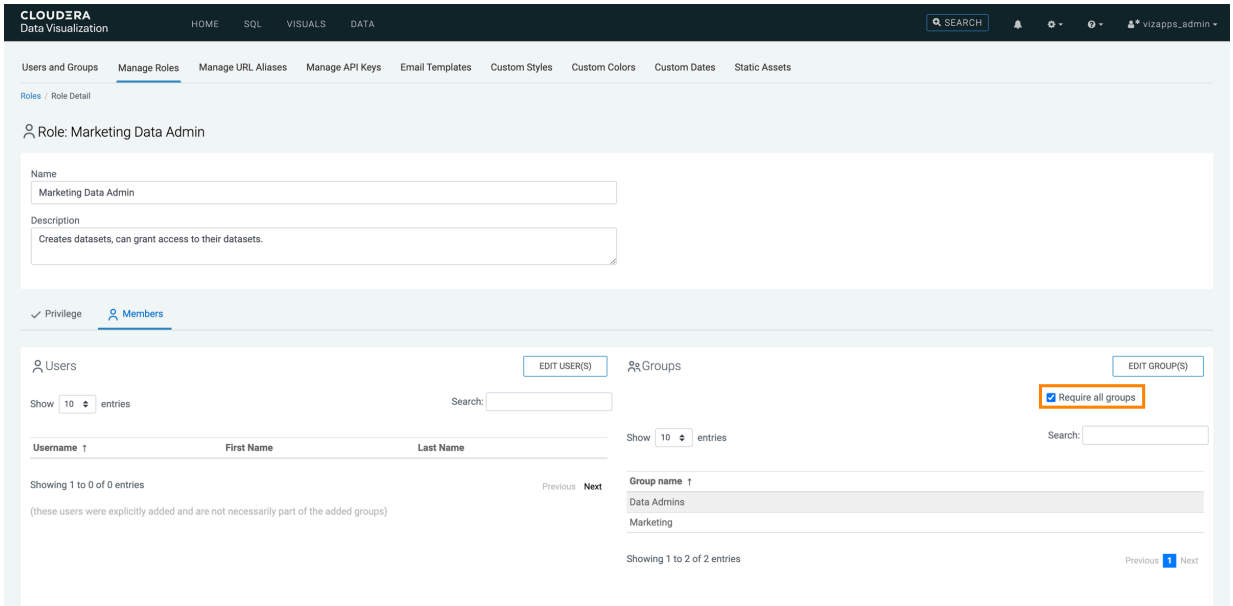

10. Repeat the previous steps to assign groups to the rest of the roles, according to the following information. Remember that you must use the Require all groups option.

Table 3: Roles and their group assignments

Role Name	Assigned groups
Marketing Data Admin	• Data_Admins • Marketing
Marketing Analyst	• Analysts • Marketing
Marketing Visual Consumer	• Visual_Consumers • Marketing
Sales Data Admin	• Data_Admins • Sales
Sales Analyst	• Analysts • Sales
Sales Visual Consumer	• Visual_Consumers • Sales
Operations Data Admin	• Data_Admins • Operations
Operations Analyst	• Analysts • Operations
Operations Visual Consumer	• Visual_Consumers • Operations

Assigning users to groups

About this task

Depending on your installation/migration scenario and security configuration, you may already have the required users, and they may already be assigned to the correct groups.

To continue our example, you can see how to create new users and assign them to the groups. Appropriate group membership automatically endows the user with the required role.

The following steps demonstrate how to create the necessary users and assign them to user groups in Cloudera Data Visualization.

Procedure

1. On the main navigation bar, click Gear the icon.
2. Select Users & Groups
3. In the Manage Users & Groups interface, click New User.

4. In the New User modal window, enter the following information:
   - Name: John_Marketing
   - Password: password of your choice.
   - Permissions: Select Normal user.
   - Under Groups tab, select Data Admin and Marketing.

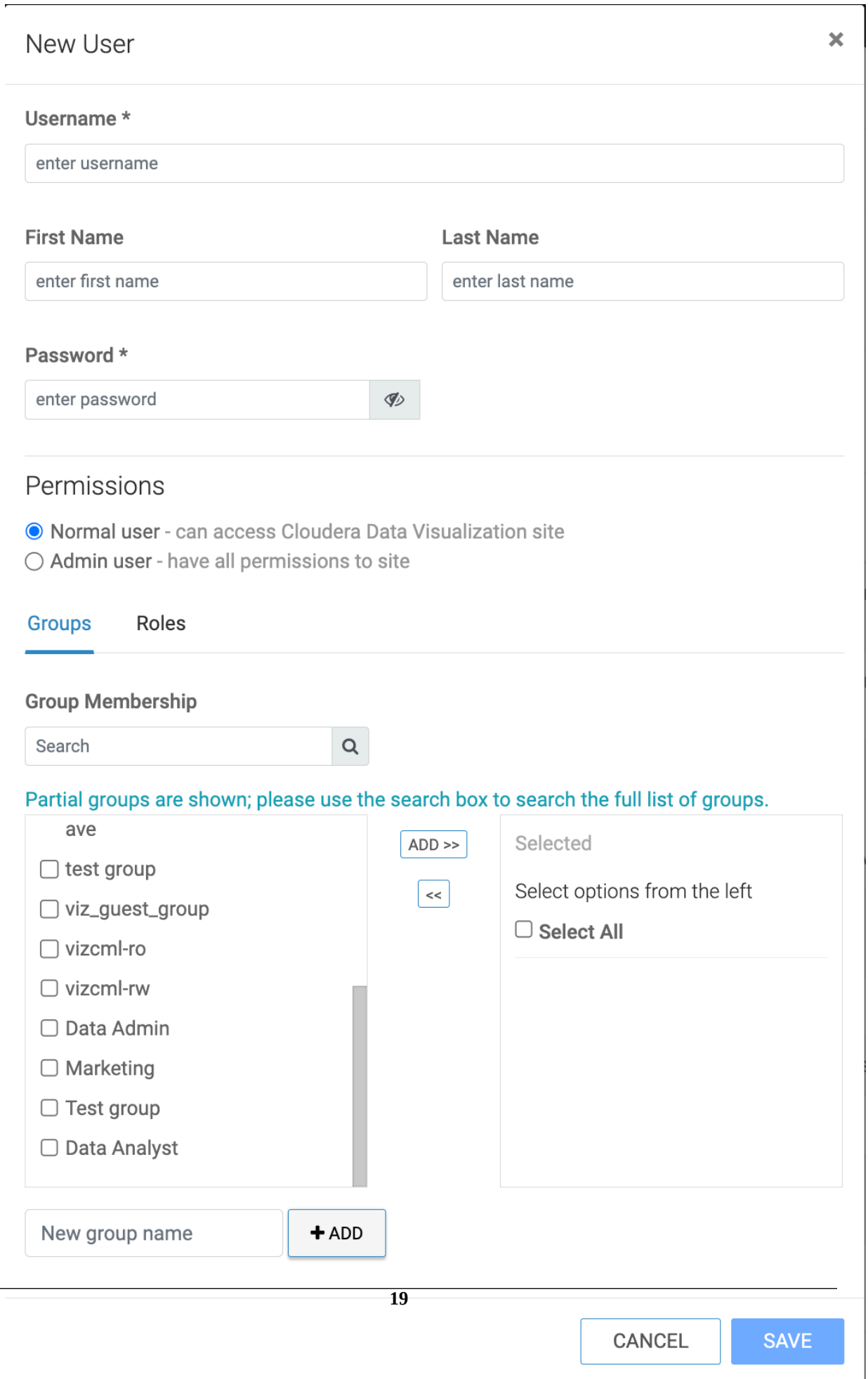
New User
Username *
enter username
First Name
enter first name
Last Name
enter last name
Password *
enter password
Permissions
Normal user - can access Cloudera Data Visualization site
Admin user - have all permissions to site
Groups
Roles
Group Membership
Search
Partial groups are shown; please use the search box to search the full list of groups.
ave
test group
viz_guest_group
vizcml-ro
vizcml-rw
Data Admin
Marketing
Test group
Data Analyst
ADD >>
<<
Selected
Select options from the left
Select All
New group name
ADD
CANCEL
SAVE

5. Click ADD to move these groups to the right column, and then click SAVE.

   In this process, you will not assign users to roles directly, but through the group membership only.

6. Repeat the previous steps to create new users and assign groups to them, according to the following information. This is a minimal set of users that 'populates' each role. You can have multiple users assigned to the same groups.

Table 4: Users and their group assignments

Role Name	Assigned groups
John_Marketing	• Data_Admins • Marketing
Kate_Marketing	• Analysts • Marketing
Lauren_Marketing	• Visual_Consumers • Marketing
Adele_Sales	• Data_Admins • Sales
Beyonce_Sales	• Analysts • Sales
Cher_Sales	• Visual_Consumers • Sales
Xuong_Operations	• Data_Admins • Operations
Yoneda_Operations	• Analysts • Operations
Zariski_Operations	• Visual_Consumers • Operations

What to do next

After completing these steps, review the list of users to verify the assigned roles.

Related Information

Assigning groups to roles

Cloudera Data Visualization 7.2.9

Cloudera Data Visualization Quickstart

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Understand the Cloudera Data Visualization user interface..... 4

Set up your data..... 5

Create a data connection..... 5

Creating a dataset..... 7

Create a visual..... 9

Create a dashboard..... 12

Create an app.....18

Understand the Cloudera Data Visualization user interface

Cloudera Data Visualization enables you to explore data and communicate insights across the whole data lifecycle by using visual objects. The web-based user interface provides user-friendly, clear and intuitive navigation in Cloudera Data Visualization.

The default Cloudera Data Visualization homepage contains the following main items:

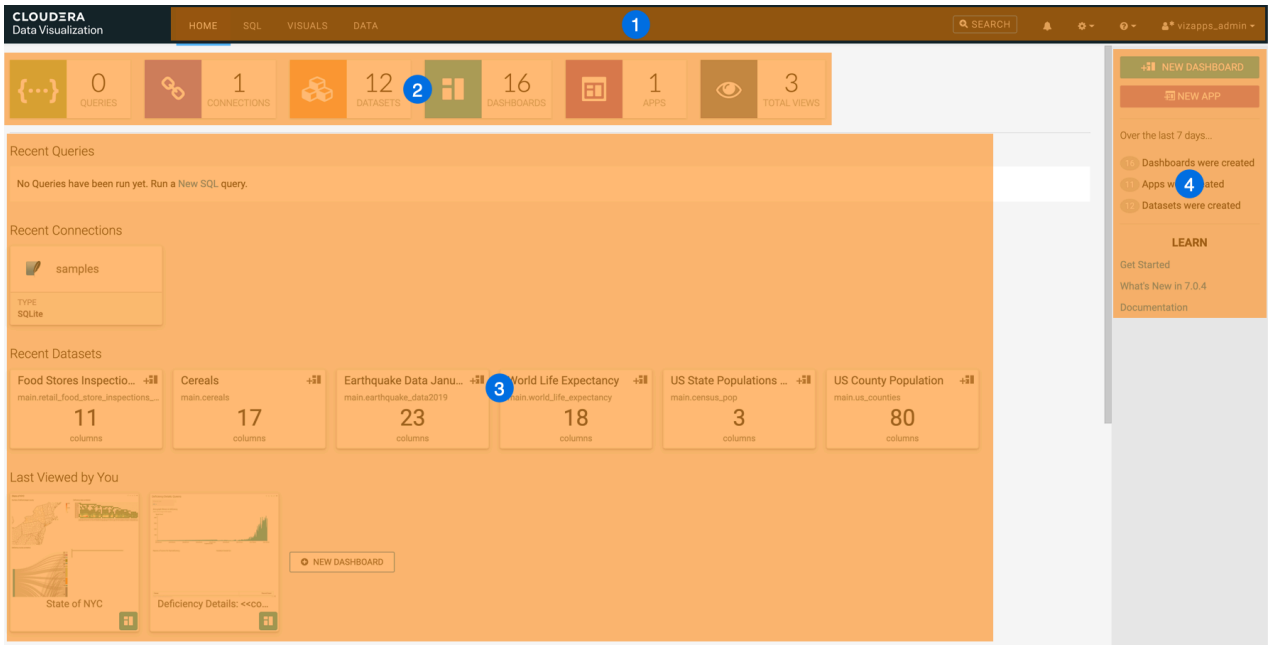

1. The main navigation bar offers you direct access to the following interfaces and actions:
   - Home
   - SQL
   - Visuals
   - Data
   - Search
   - Notifications center
   - Settings
   - Help
   - User management
2. The statistics banner shows the number of queries, connections, datasets, dashboards, apps, and total views in your Cloudera Data Visualization instance.
3. The preview area provides quick access to the following items:
   - Recent queries
   - Recent connections
   - Recent datasets
   - Favorites
   - Items you last viewed
   - Items you recently created
   - Overall most popular items
   - Sample dashboards

4. The homepage side menu bar offers you access to the following functions:
   - NEW SQL takes you to the Data Connection interface, where you can compose a SQL query.
   - NEW DASHBOARD takes you to the Dashboard Designer interface, where you can create new dashboards and visuals.
   - NEW APP takes you to the App Designer interface, where you can build and style custom applications from existing dashboards and visuals.
   - Over the last 7 days... shows statistics on how many dashboards, apps, and datasets were created.
   - In the LEARN section, you can find the following information:
     - Get Started points to help content embedded in the tool.
     - What's New in opens a modal window showcasing new features.
     - Documentation opens this product documentation library.

If you need more information about the UI, see *Cloudera Data Visualization homepage*.

Related Information

Cloudera Data Visualization homepage

Set up your data

Cloudera Data Visualization supports a large number of data connections.

Related Information

Data connections in Cloudera Data Visualization

Create a data connection

You must connect to your data before you can start using Cloudera Data Visualization for modeling and visualizing the data. You can define connections to various source systems. Learn how to create a simple data connection.

About this task

In Cloudera Data Warehouse, the connection to the database catalog is automatically set up when you enable Cloudera Data Visualization in a Virtual Data Warehouse.

Note: You can create your own connections to other data warehouses, but it is not supported.

Database catalogs and virtual data warehouses automatically inherit the same security restrictions that are applicable to your Cloudera environment. There is no need to specify the security setup again for each database catalog or virtual warehouse. If you cannot see the data in the connected database catalog after you log in, check and adjust data access permissions or your environment and data warehouse user permissions.

In Cloudera AI, you can set up several connection types. For example, you can connect Cloudera Data Visualization to an Impala or Hive data warehouse. For more information on connection types, see *Data connections in Cloudera Data Visualization*.

Important: Only users with Manage data connections privilege can create and manage connections in Cloudera Data Visualization. If you create a connection, you automatically have the privileges to create and manage datasets on this connection, and also build dashboards and visuals in these datasets. For more information on user privileges, see *RBAC permissions*.

Procedure

1. On the main navigation bar, click DATA.

   You get to the DATA interface, open on the Datasets tab.

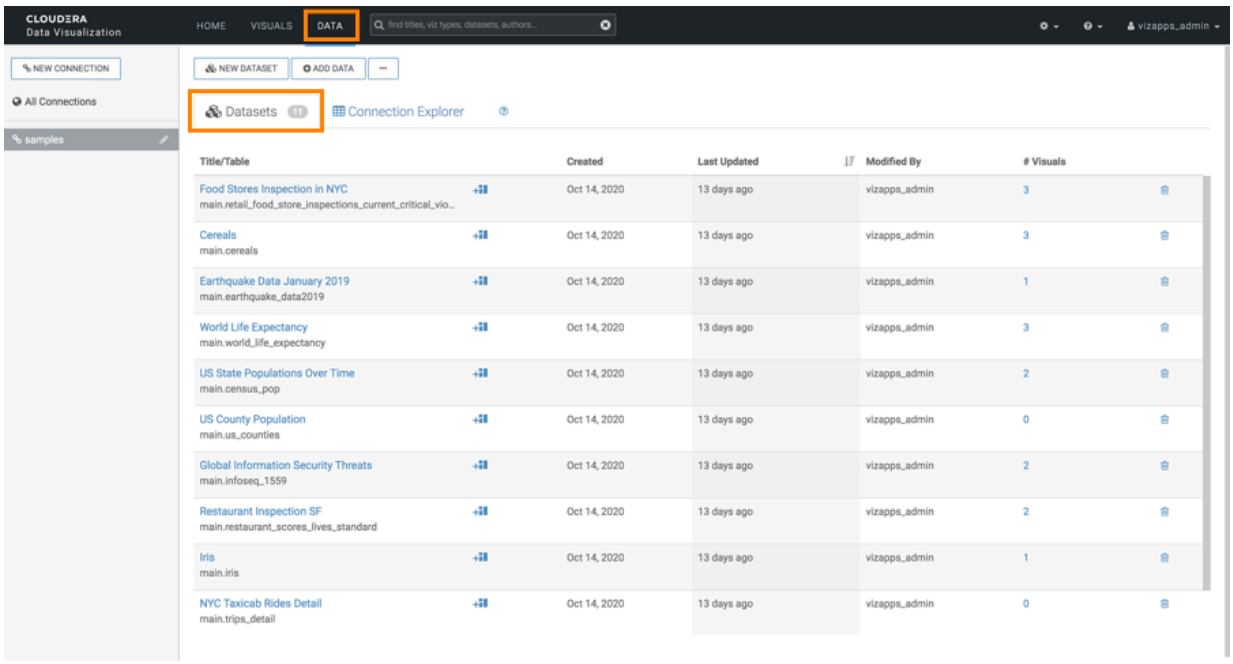

2. In the side menu bar, click NEW CONNECTION.

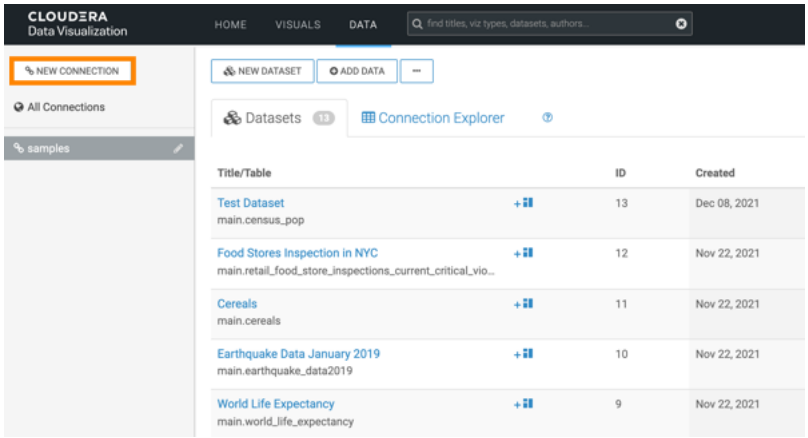

   The Create New Data Connection modal window appears.

3. Specify the details of the new connection.

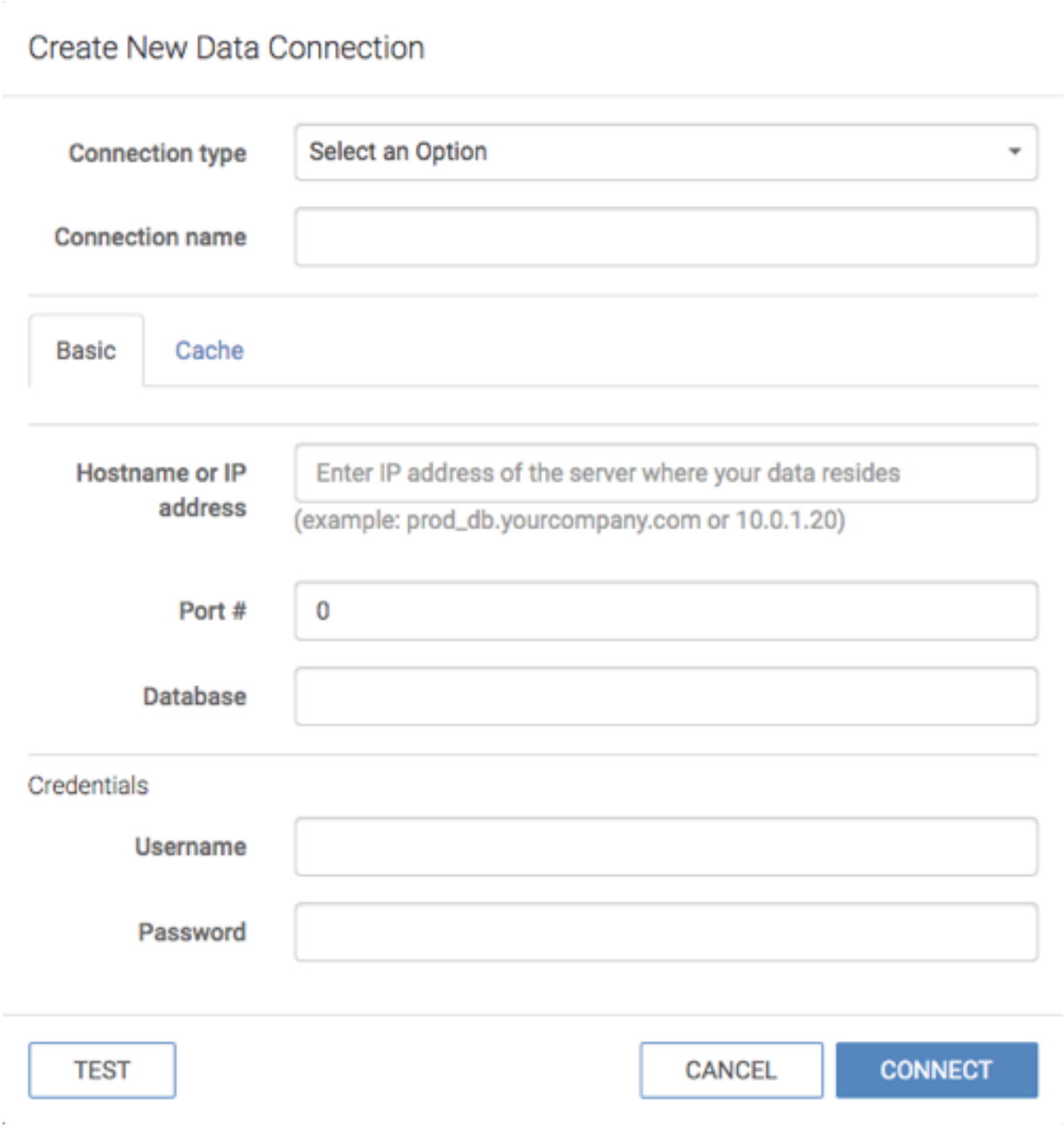

4. Click TEST.
   If the connection is valid, the system returns a Connection Verified message.
5. Click CONNECT.

Results

If this operation succeeds, the name of the new connection appears on the side menu bar.

Creating a dataset

You can define datasets as a semantic layer on top of your data tables and views in the data store. You need to create a dataset before you can create dashboards or apps. Learn how to create a dataset in Cloudera Data Visualization.

Procedure

1. On the main navigation bar, click DATA.
2. Select a connection from the side menu bar.
3. Click NEW DATASET near the top of the screen.

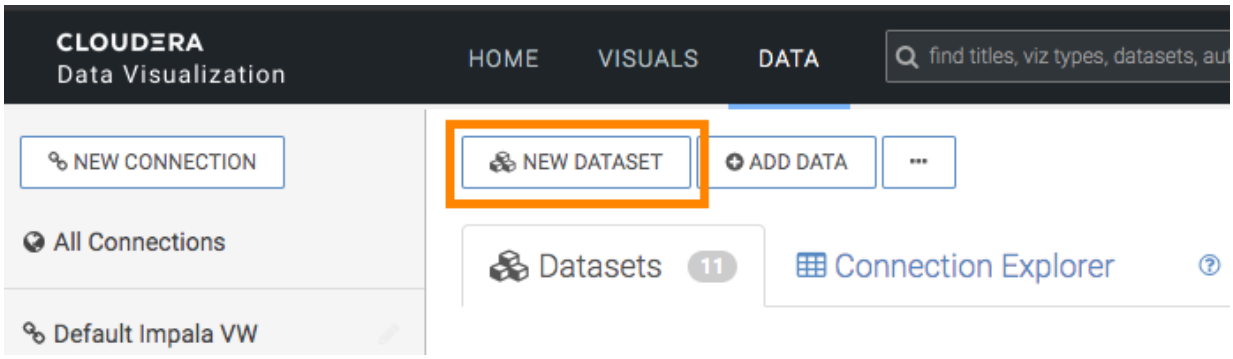

The New Dataset modal window appears.

4. Specify the details of the new dataset.

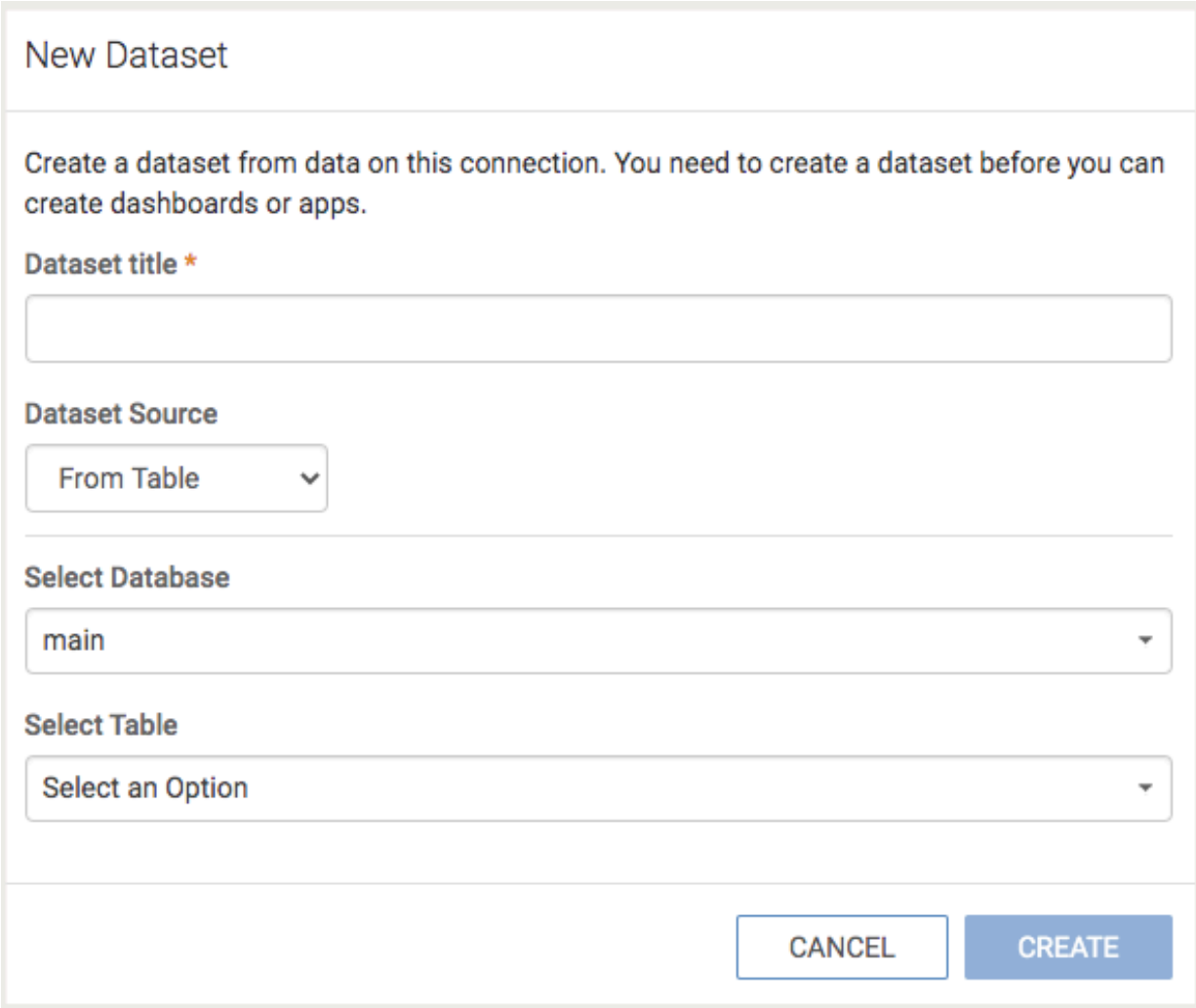

5. Click CREATE.

Results

You can see the newly created dataset under the Datasets tab in the DATA interface.

Tip: To find the dataset in the future, you can scroll through the list of datasets on the connection, or use Search at the top of the page.

What to do next

You can now start creating visualizations and organize your visual artifacts into dashboards and applications.

Create a visual

Learn how you can create a visual in Cloudera Data Visualization.

Procedure

1. On the main navigation bar, click VISUALS.
2. Click NEW DASHBOARD to open a new dashboard in Edit mode.

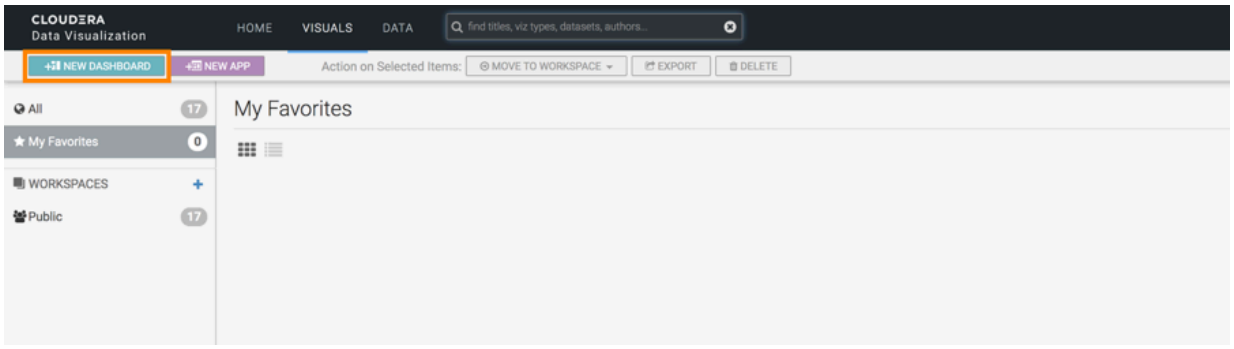

   Alternatively, click an existing dashboard, and then click Edit to make changes to it.
3. In the Dashboard Designer menu, click the Visuals icon to open the ADD VISUALS interface.

4. Click NEW VISUAL.

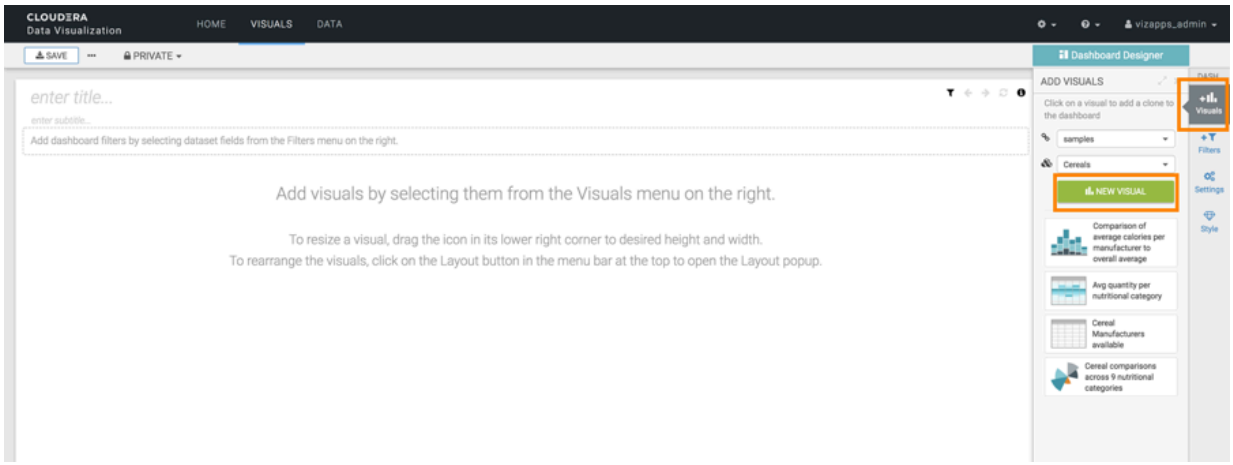

The visual designer interface opens with a default table representation of the dataset you want to visualize.

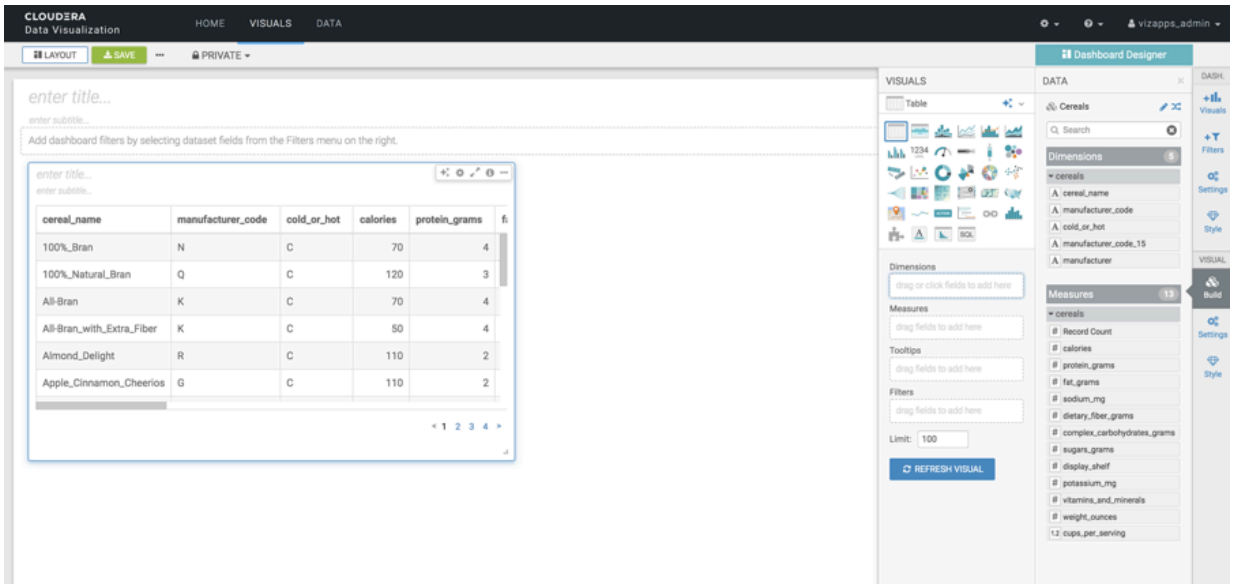

5. Choose a visual type from the VISUALS catalog, which shows the standard visual types. You can switch between visual types by selecting a different type in the catalog.

Tip:

When creating a new visual for a dataset, you can get visualization suggestions for your data. For details, see *Exploring visual options*.

If Explore Options Visual Types is not available on the UI when creating a new visual, enable it in the Site Settings interface. For details, see *Enabling exploring visual options*.

6. Populate the shelves from the available Dimensions and Measure fields.

   The shelves of a visual specify the fields and segments that provide the data for visualization. Different visual types require specific information, and have somewhat different shelves.

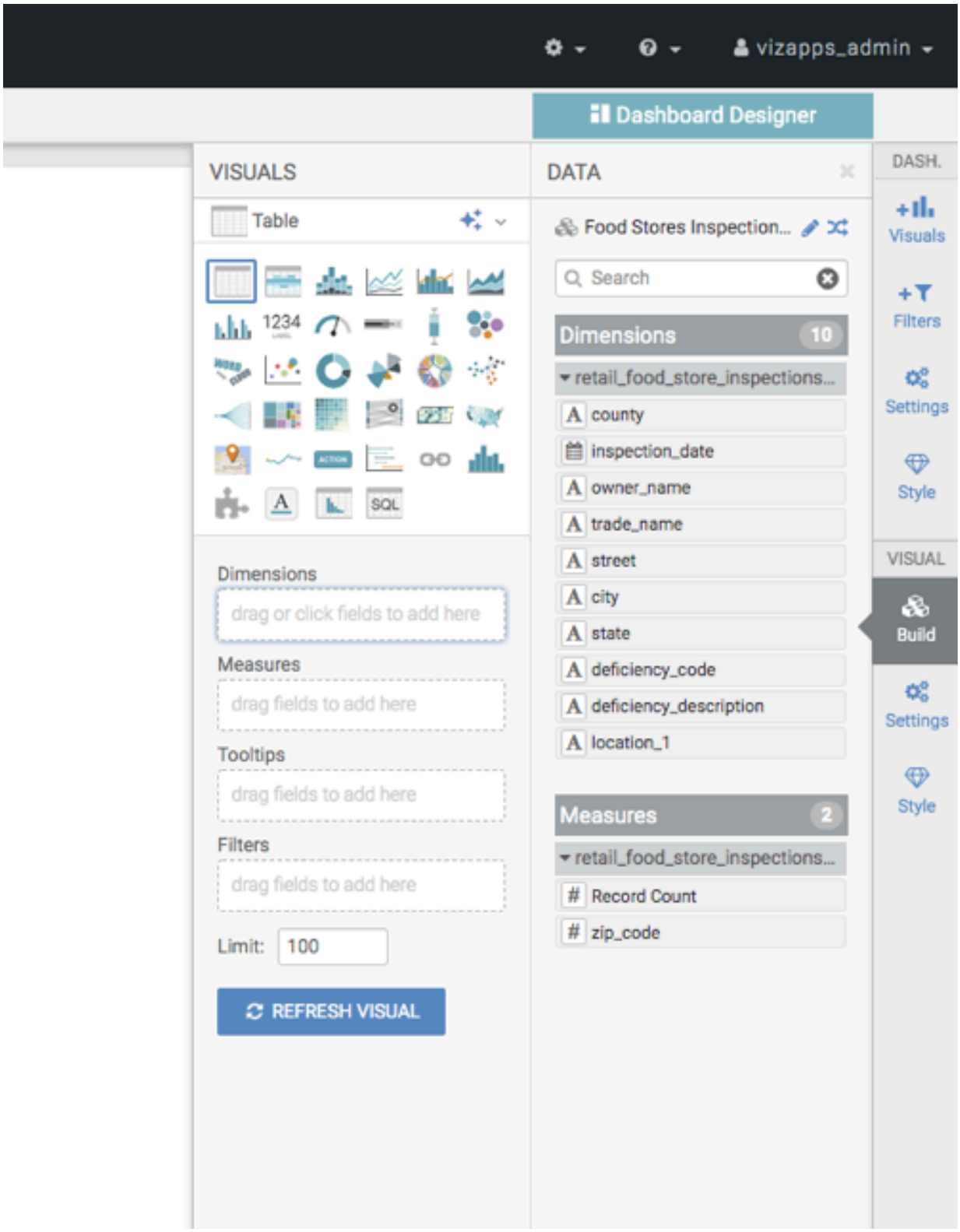

7. Click REFRESH VISUAL.
8. Add a title for the visual by clicking into the enter title... field to edit it.

9. [Optional] Add a brief description of the visual by clicking into the enter subtitle... field under the title of the visual.
10. At the top left corner of the visual designer, click SAVE.

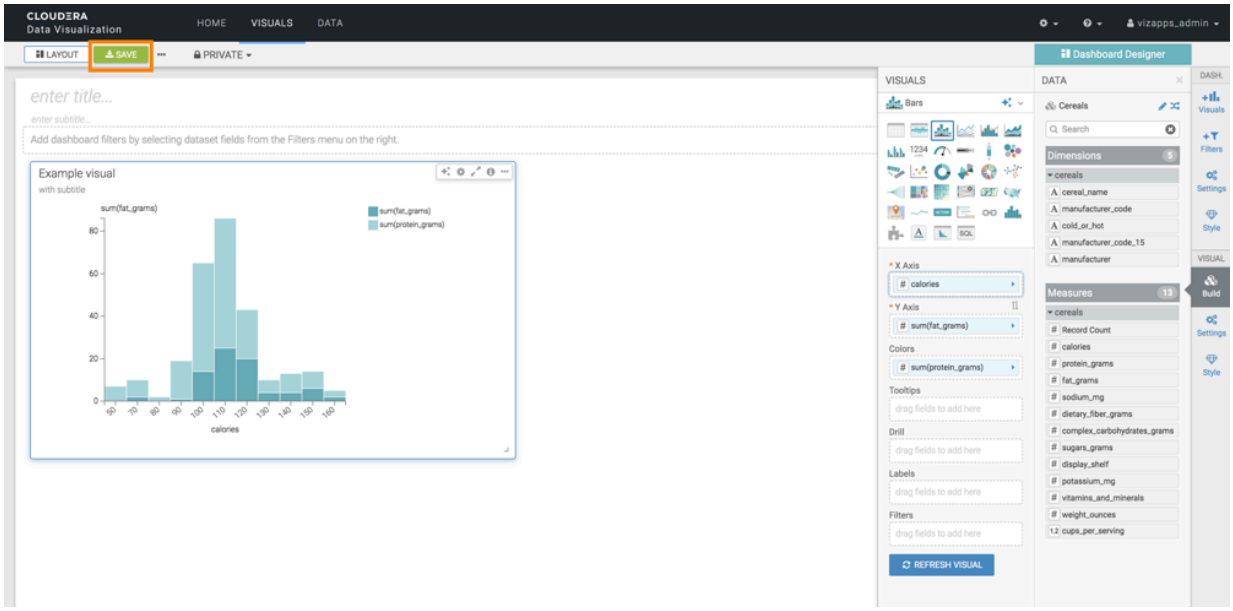

Related Information

Exploring visual options

Enabling exploring visual options

Create a dashboard

Learn how you can create a dashboard in Cloudera Data Visualization.

Procedure

1. There are several possible starting points for creating new dashboards:
   - Click NEW DASHBOARD on the main HOME interface, in the top right corner.
   - Click Create New Dashboard in the top right corner.
   - On the DATA interface, click the icon next to the name of a particular dataset.
   - On the Dataset Detail view for a particular dataset, click NEW DASHBOARD in the top right corner.

   The Dashboard Designer opens on an empty dashboard.

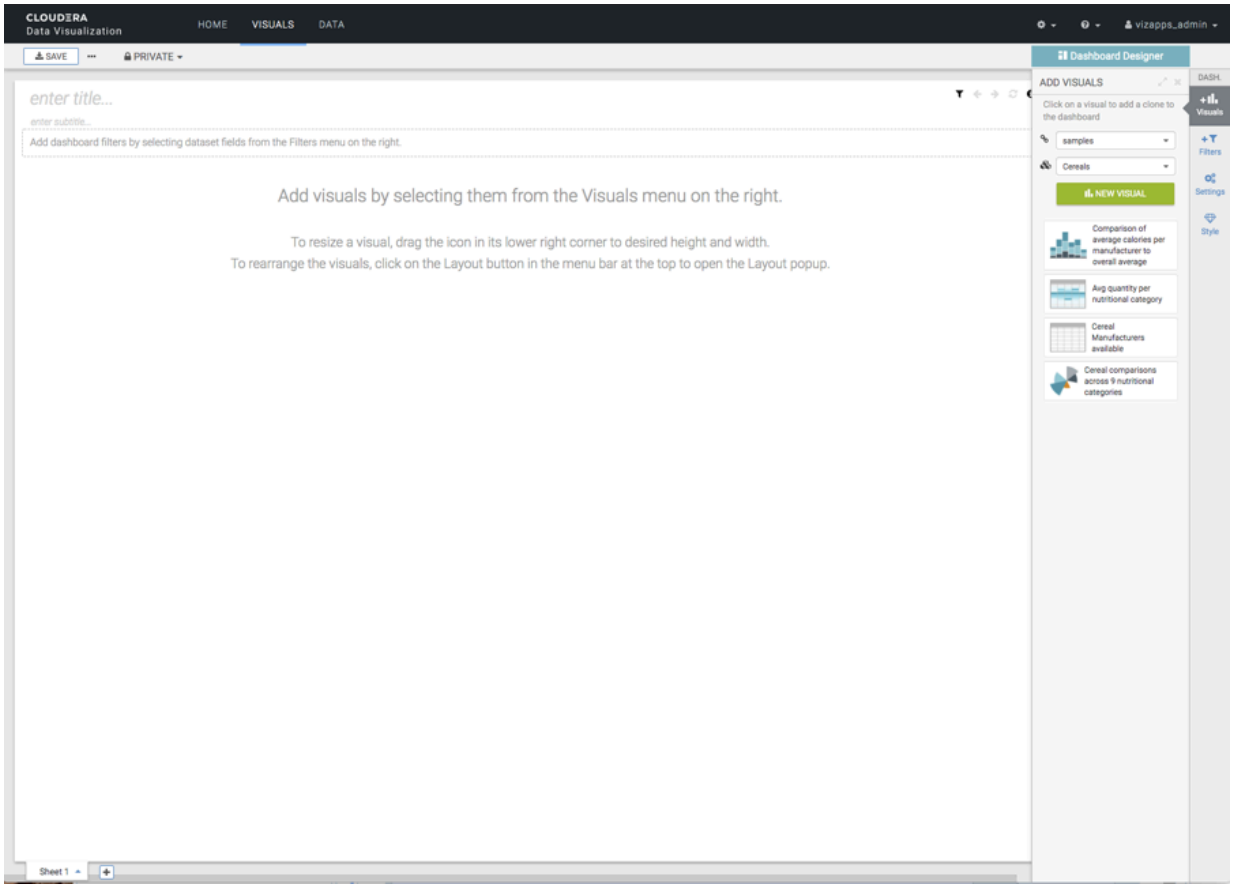

2. Check the dataset of the dashboard In the Visuals menu of the Dashboard Designer, and change it if needed.

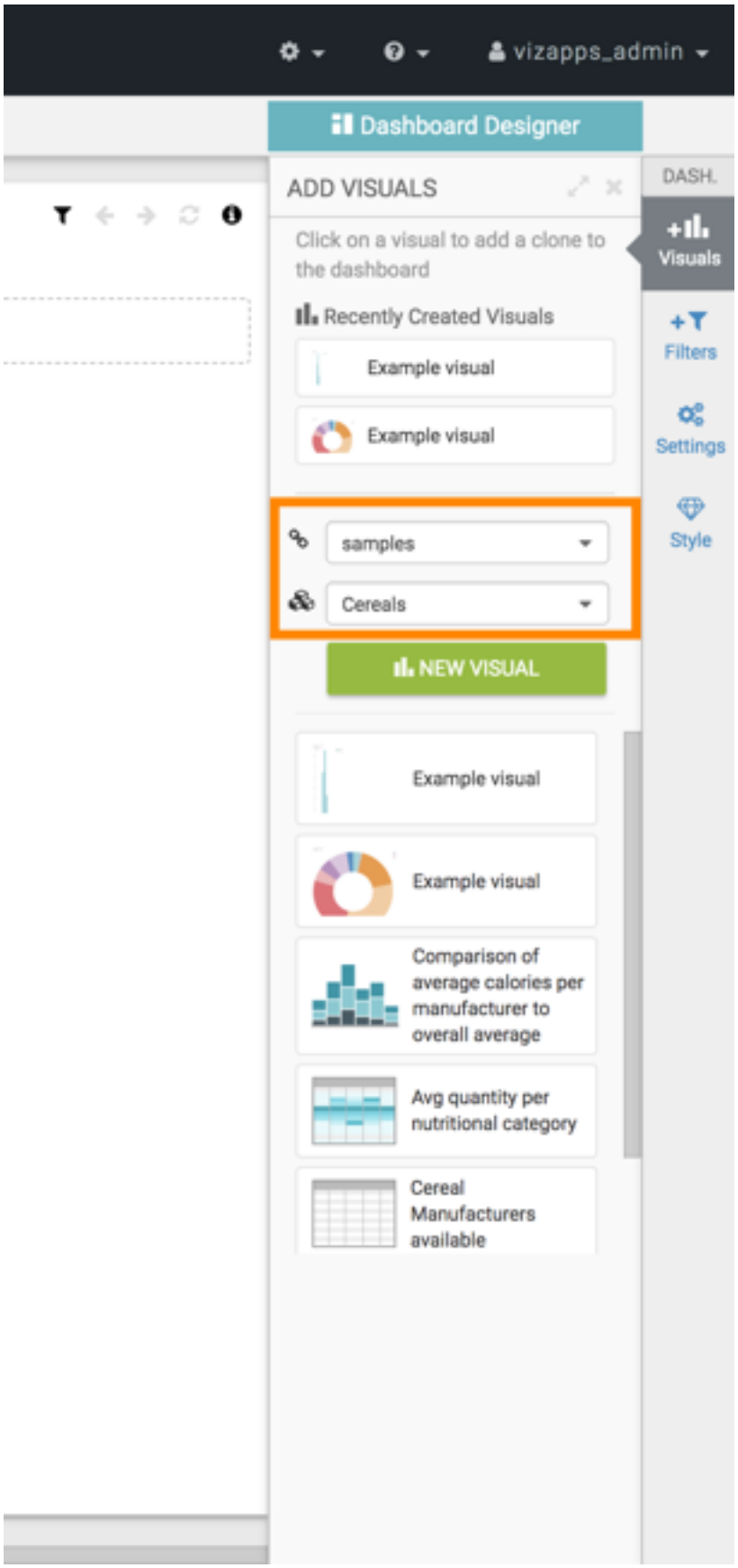

3. In the Visuals menu, click NEW VISUAL to add a new visual to the dashboard.

   Important: You cannot save an empty dashboard. You must add at least one visual to a sheet before saving the dashboard.

4. Add a title for the dashboard. Optionally, you can also add a subtitle.

5. [Optional] You can customize Settings, Styles, and Custom Styles for the dashboard in the Dashboard Designer side menu bar.

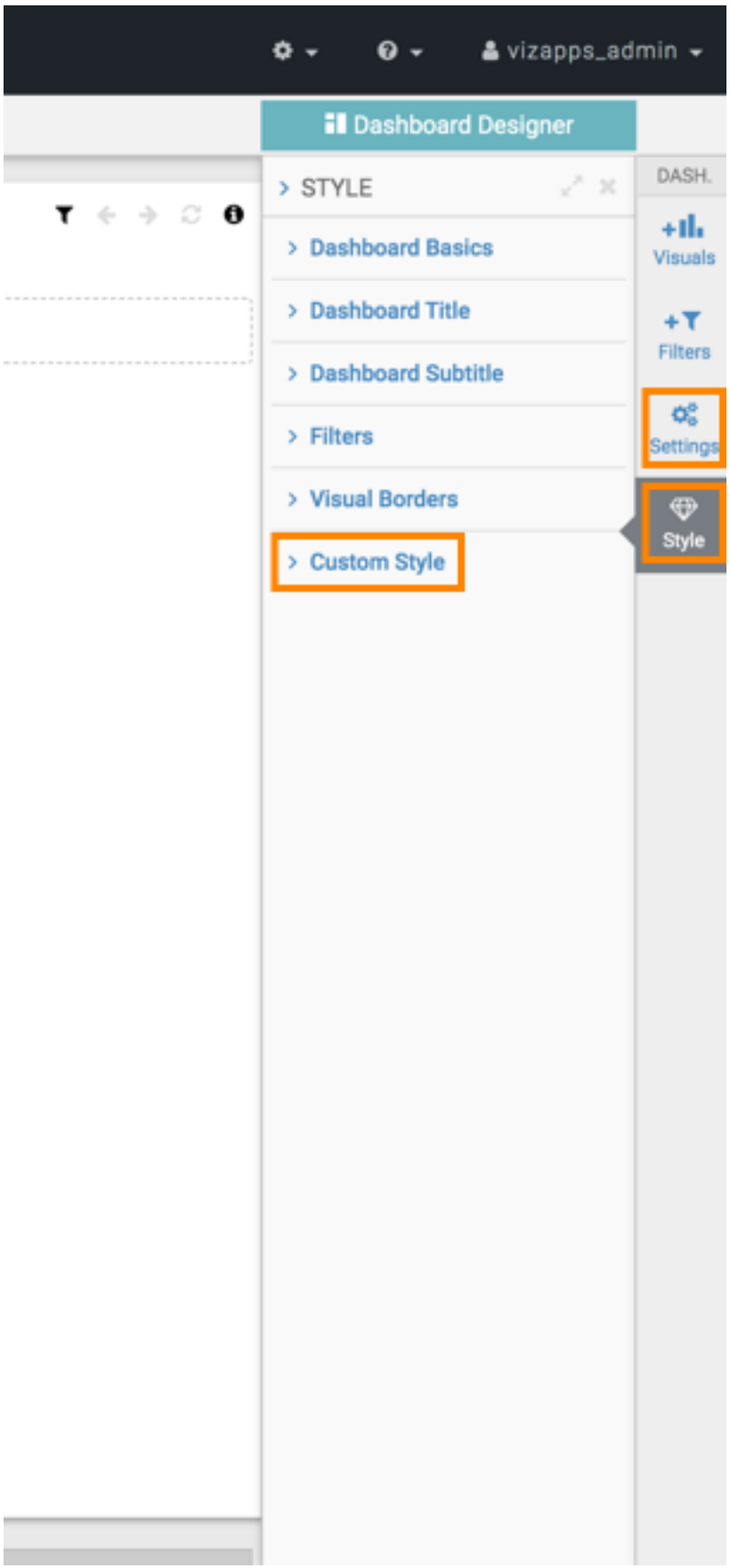
vizapps_admin
Dashboard Designer
STYLE
Dashboard Basics
Dashboard Title
Dashboard Subtitle
Filters
Visual Borders
Custom Style
DASH.
Visuals
Filters
Settings
Style

6. Click SAVE in the top left corner.

Create an app

Learn how you can create an app in Cloudera Data Visualization.

Procedure

1. Click NEW APP on the HOME interface, in the top right corner on the side panel.
   Alternatively, click Create New App in the top right corner.

   The App Designer interface is displayed. It opens on the App Navigation tab, and it shows a default menu tab tile.

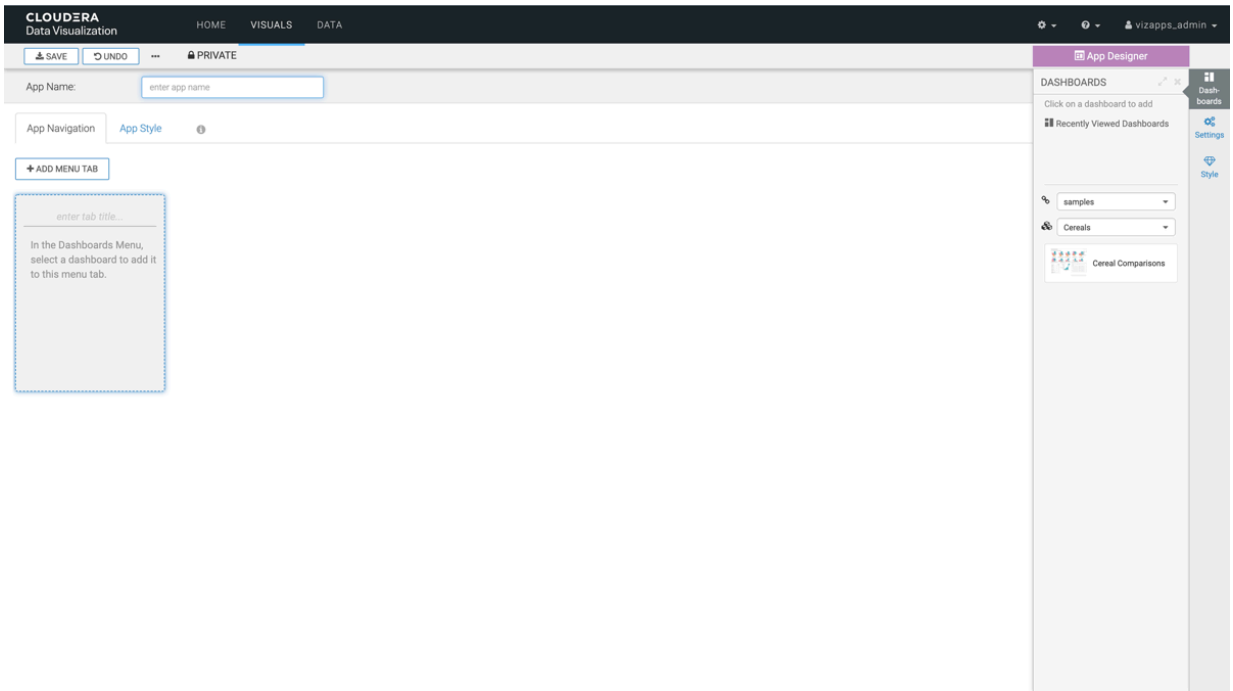

2. In the DASHBOARDS menu of the App Designer, select a dataset.

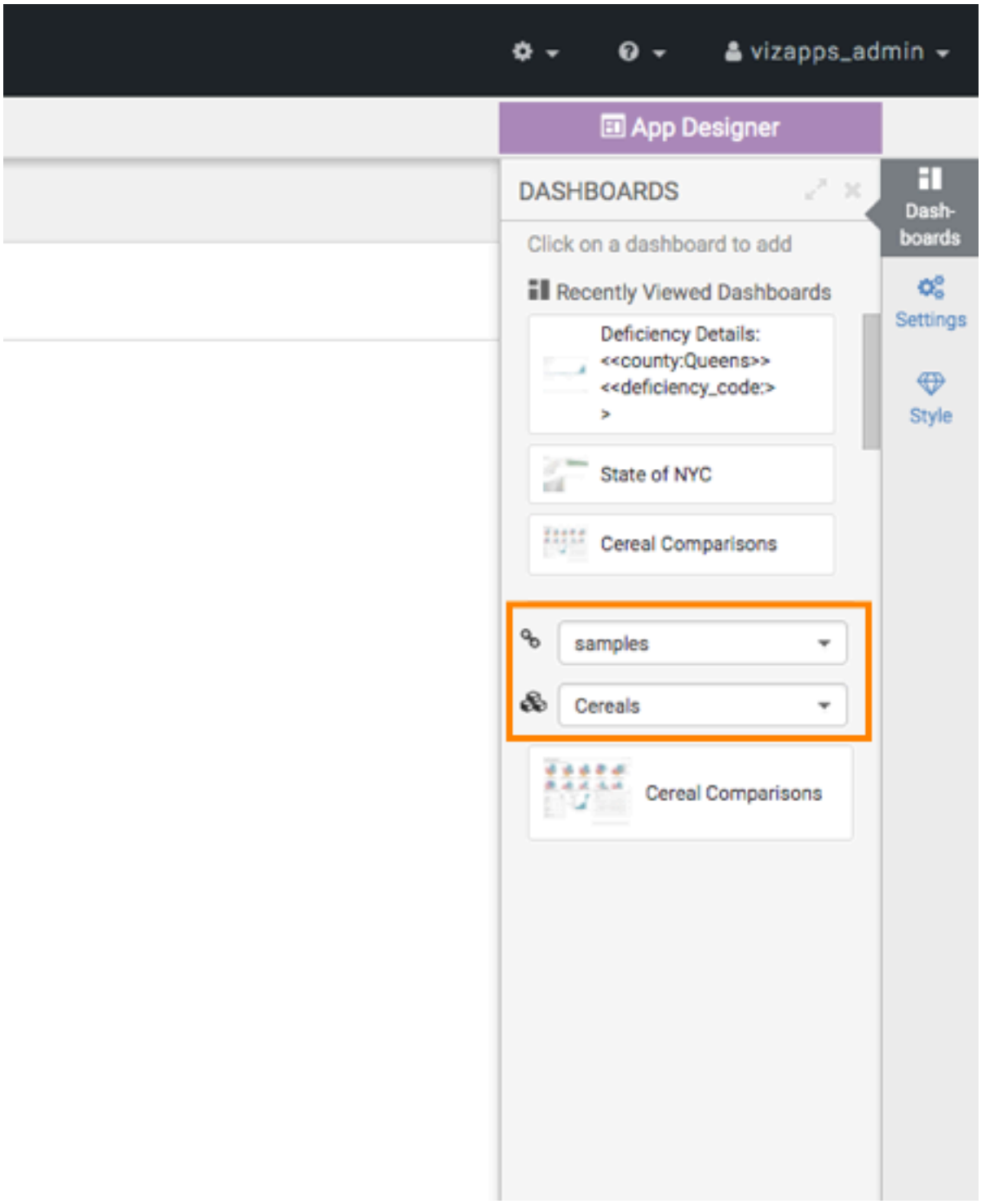

3. Enter a name for your app in the App Name text box.

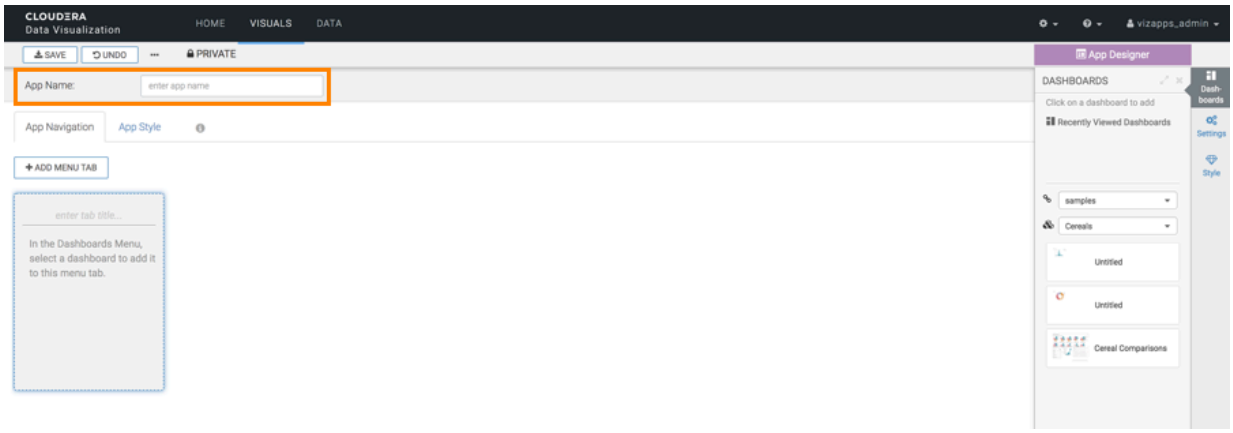

4. In the highlighted empty tab, enter the tab title.

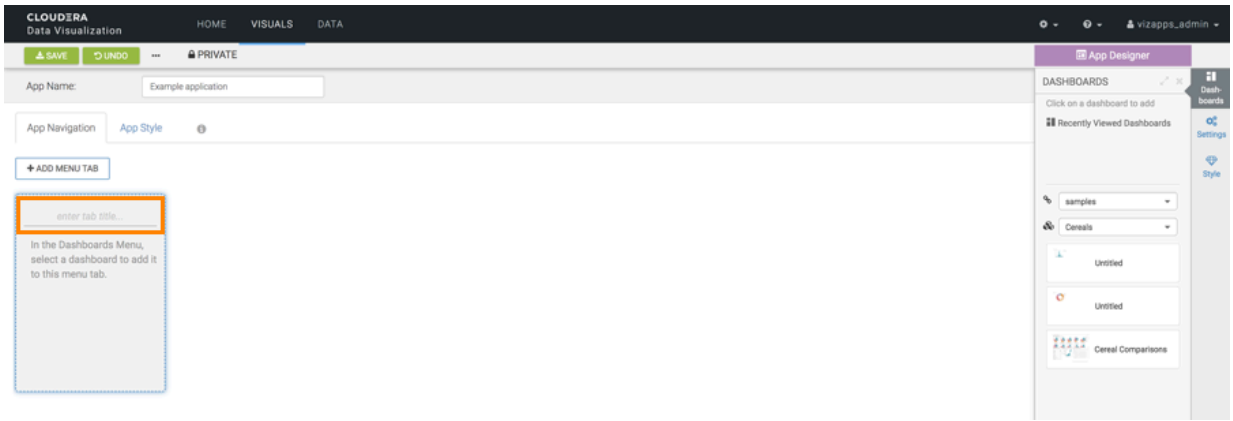

5. In the DASHBOARDS menu, select a dashboard to add it to this tab.

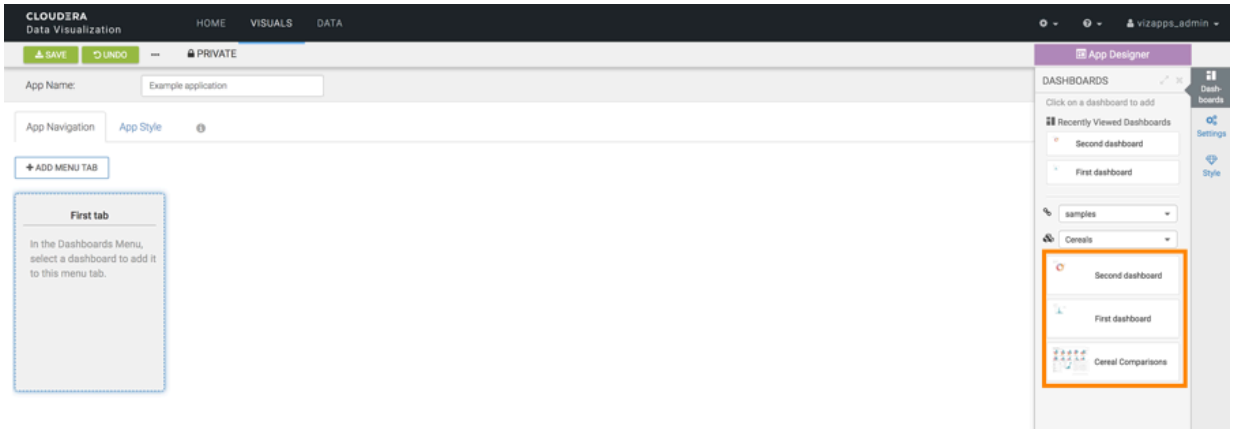

6. Switch to the App Style tab.

7. Design the appearance of your new app using the various style menu options on the side panel.

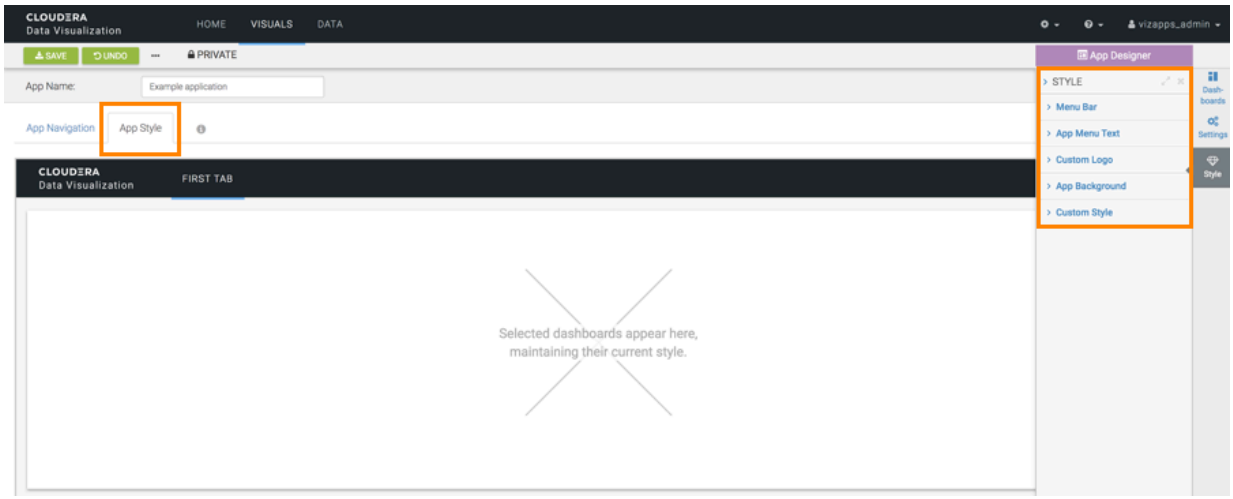

8. Click SAVE.

Results

You have created a new Cloudera Data Visualization app. To the right of the App Name, you can see the ID of the app, its update time information, and the last user who modified it.

Cloudera Data Visualization 7.2.9

Receiving Parameters in Dashboards

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating visuals with optional dimensions.....4

Creating filters to control optional dimensions.....6

Creating visuals with optional measures.....8

Creating filters to control optional measures.....10

Creating visuals with variable dimensions.....12

Creating filters to control variable dimensions.....15

Creating visuals with variable measures.....18

Creating filters to control variable measures.....20

Creating visuals with optional dimensions

Procedure

1. Open a new dashboard.
2. Click New Visual.
3. Select the World Life Expectancy dataset in the Data menu.
4. Select the Table visual type in the Visuals menu.
5. Populate the shelves of the visual:
   - From Dimension, select and move un_region and un_subregion fields onto the Dimension shelf.
   - From Measures, select and move population field onto the Measures shelf.
   - From Dimensions, select and move year field onto the Filters shelf.
   - On the Filters shelf, select year field, choose Pick values from a list, select 2000, and click SAVE.

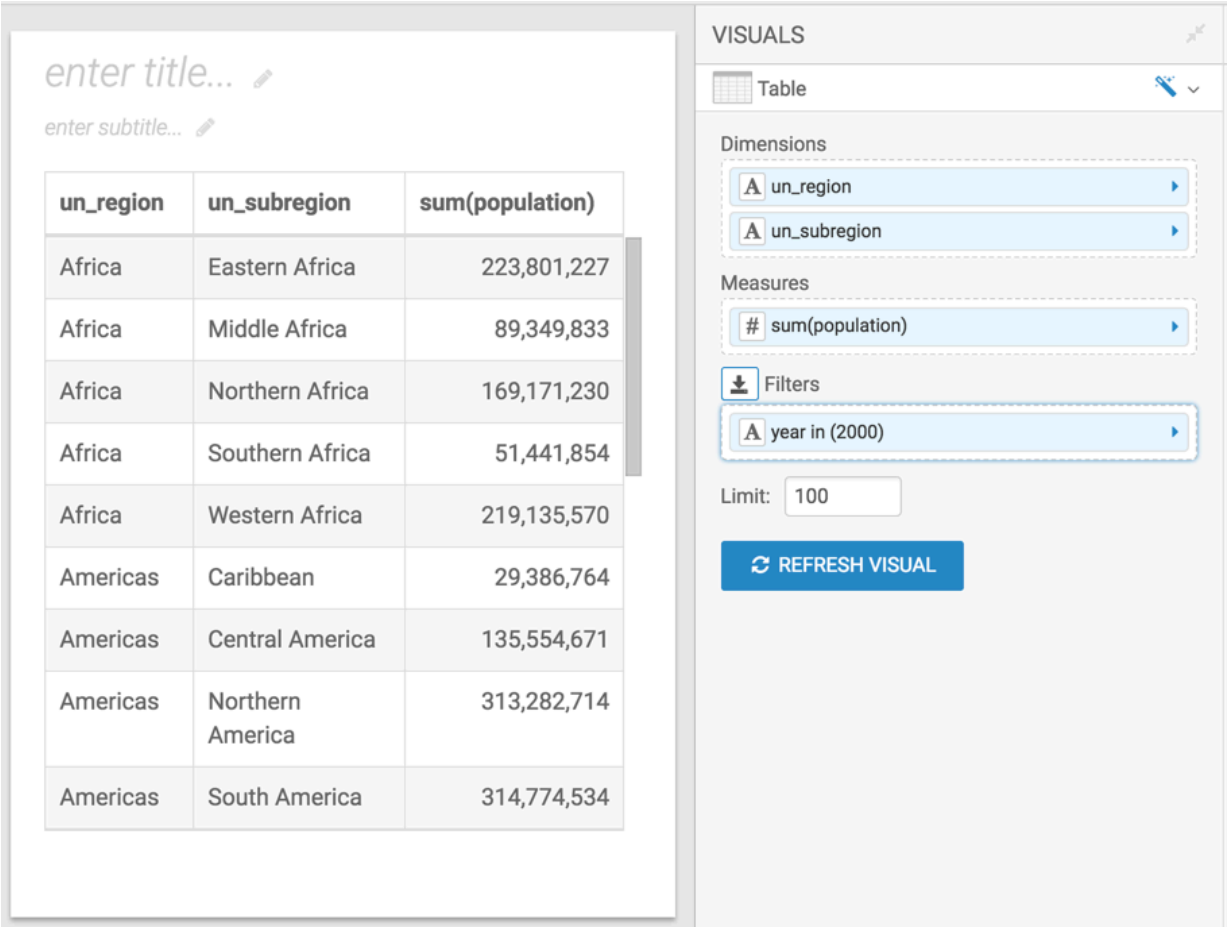

6. From Dimensions, select and move the country field to the Dimension shelf.
7. On the Dimensions shelf, click country field.

8. In the FIELD PROPERTIES menu, select [ ] Enter/Edit Expression.

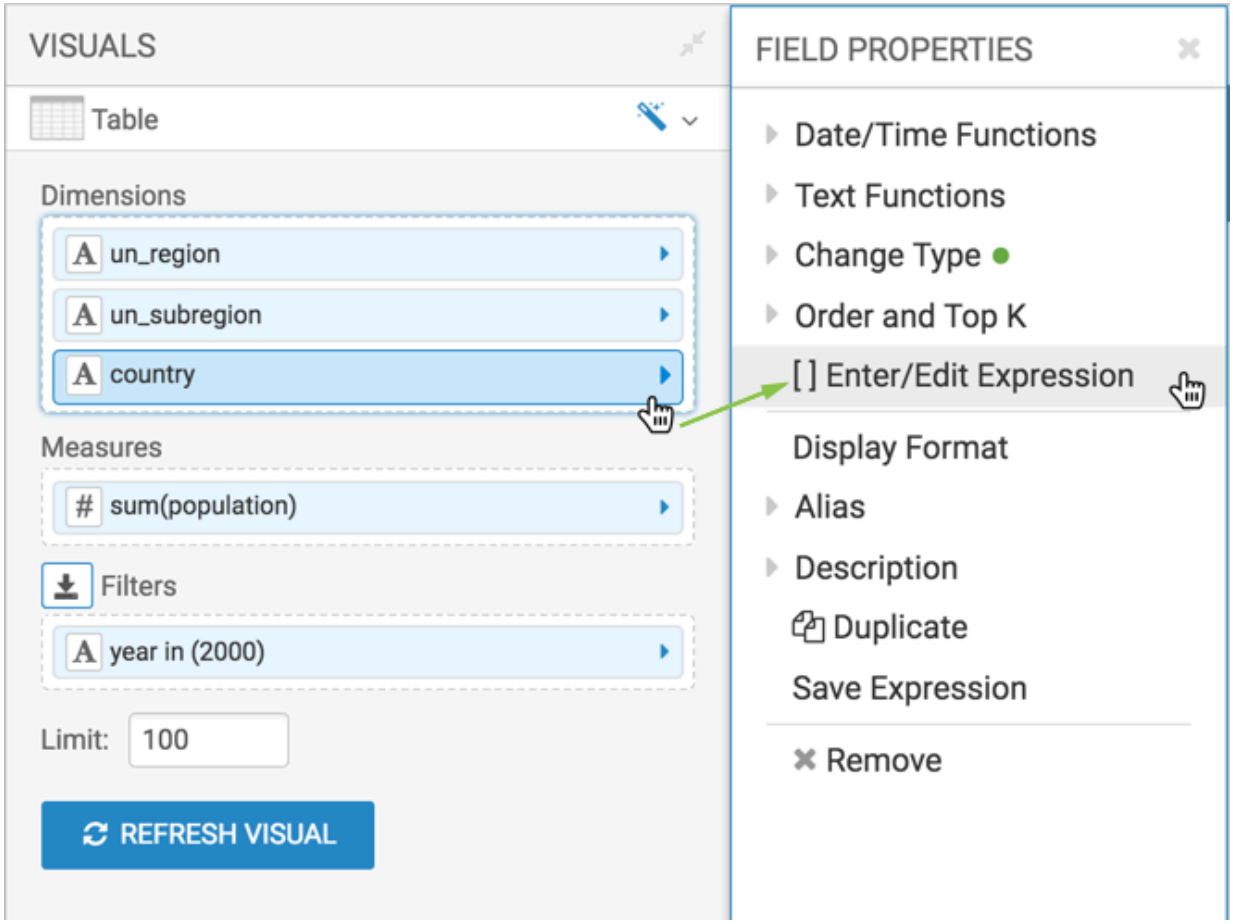

9. In the Enter/Edit Expression modal window, change the text to the following expression: [<<dim:>>]

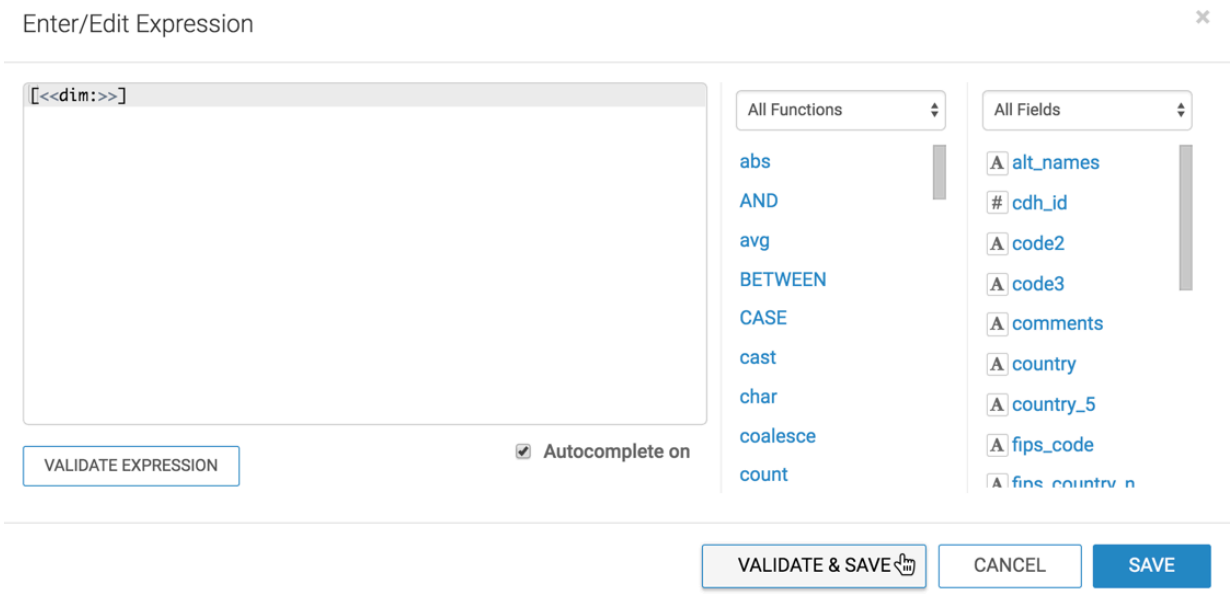

10. Click VALIDATE & SAVE.

11. Click REFRESH VISUAL.

12. Change the name of the visual to Regional Populations.

13. Click SAVE.

Creating filters to control optional dimensions

Before you begin

Before starting this work flow, complete the steps in Creating visuals with optional dimensions on page 4.

Procedure

1. In the dashboard, click the Filters tab.
2. Click CREATE CUSTOM FILTER.

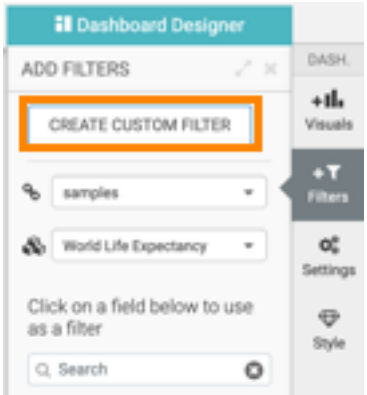

This creates a new filter in the filter area of the application, and opens the Settings modal window for that filter.

3. Click the Values tab and enter the following:
   - Under Title, enter Display Country Details.
   - Under Output Parameter, enter dim. Note that this is the parameter from Creating visuals with optional dimensions on page 4.
   - Under Specified values, enter Value: country, and Label: Country.

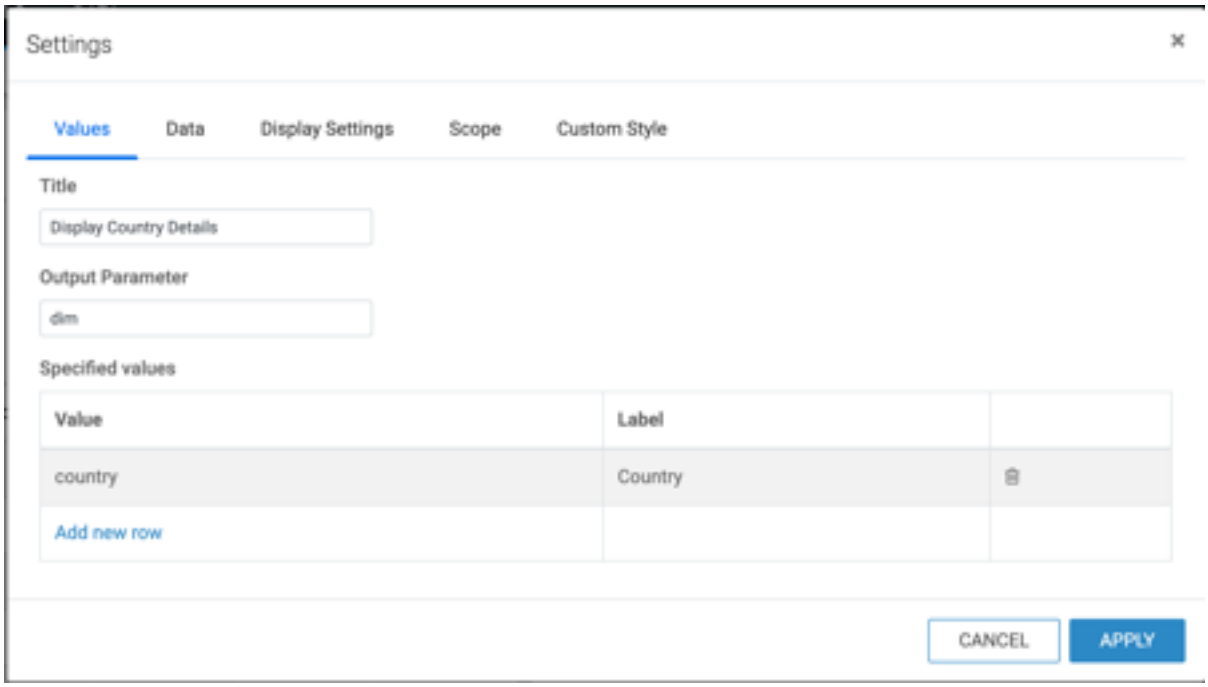

4. Switch to the Display Settings tab, select Allow only one item to be selected at a time and then select Include an option for 'All'.
5. Click APPLY.
6. Name and save the dashboard.
7. Switch to dashboard View mode and select World Regional Populations.
8. In the Display Country Details filter, select Country.

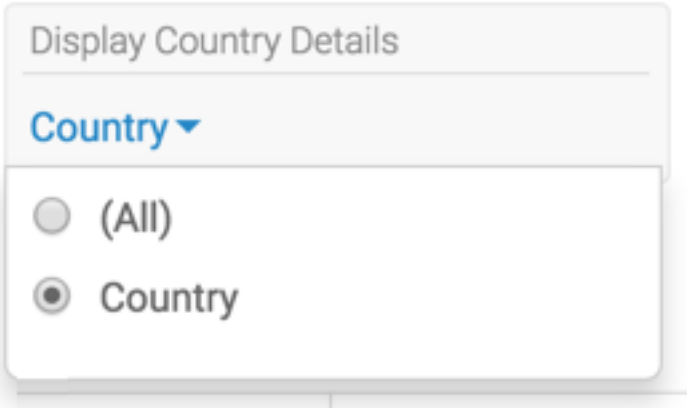

Results

The table now has a new column, country.

Creating visuals with optional measures

About this task

You may choose to duplicate the dashboard that you created earlier according to the instructions in Creating visuals with optional dimensions on page 4. In this case, open the visual, and jump to Step 6 in the following workflow.

Procedure

1. Open a new dashboard.
2. Click New Visual.
3. Select the World Life Expectancy dataset in the Data menu.
4. Select the Table visual type in the Visuals menu.
5. Populate the shelves of the visual:
   - From Dimension, select and move un_region and un_subregion fields to the Dimension shelf.
   - From Measures, select and move population field to the Measures shelf.
   - From Dimensions, select and move year field to the Filters shelf.
   - On the Filters shelf, select year field, choose Pick values from a list, select 2000, and click SAVE.

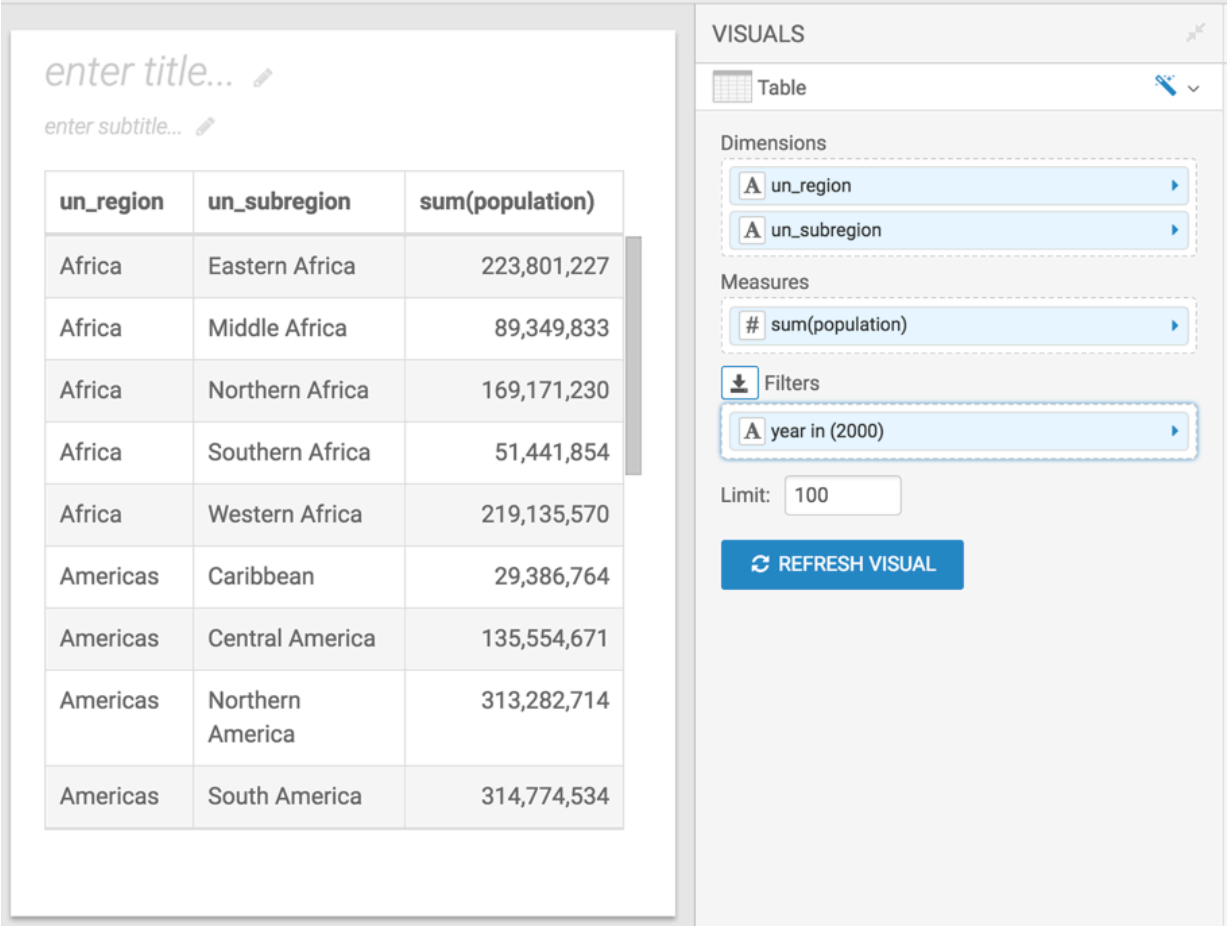

6. From Measures, select and move the life_expectancy field to the Measures shelf.
7. On the Measures shelf, click the life_expectancy field.

8. In the FIELD PROPERTIES menu, select [ ] Enter/Edit Expression.

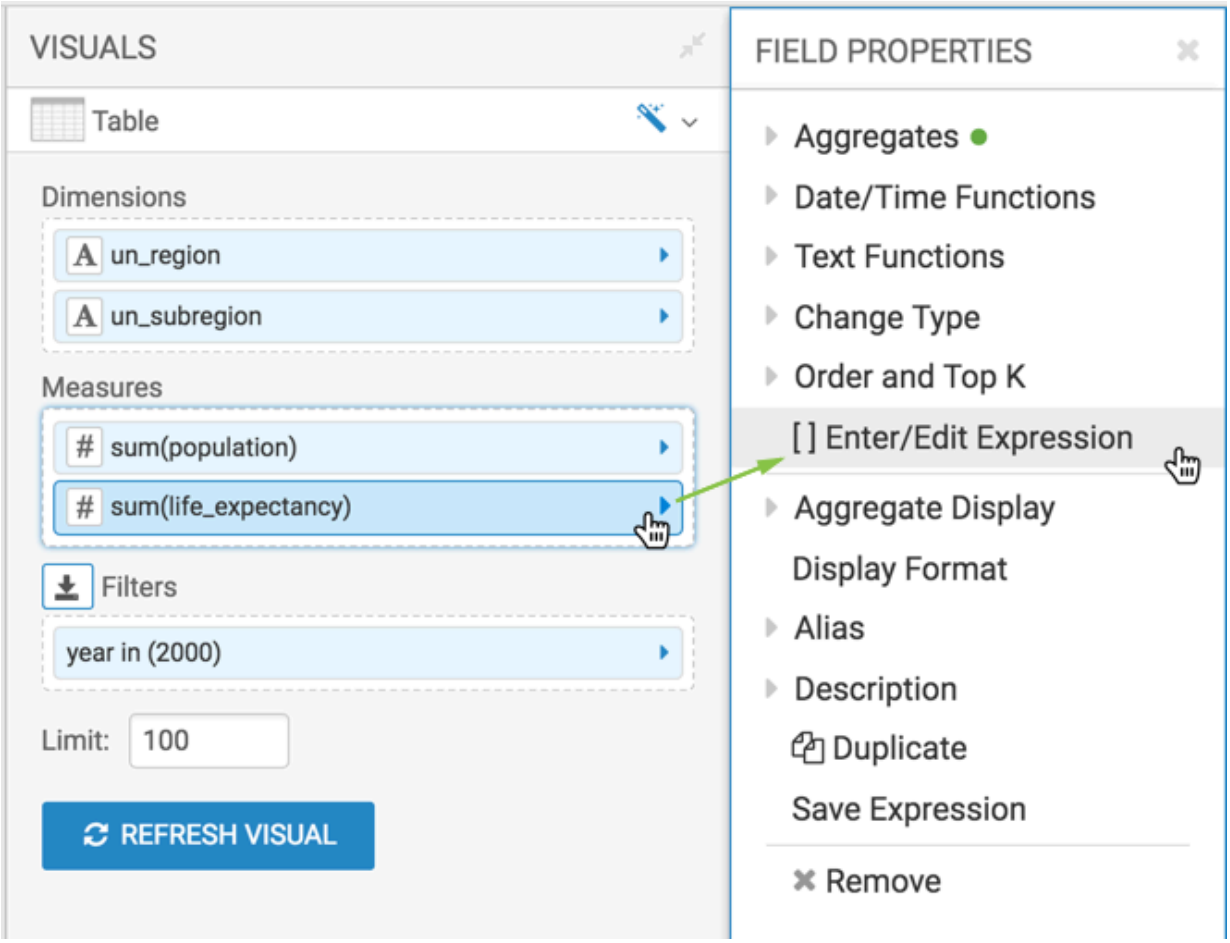

9. In the Enter/Edit Expression modal window, change the text to the following expression: [<<mes:>>].
10. Click VALIDATE & SAVE.

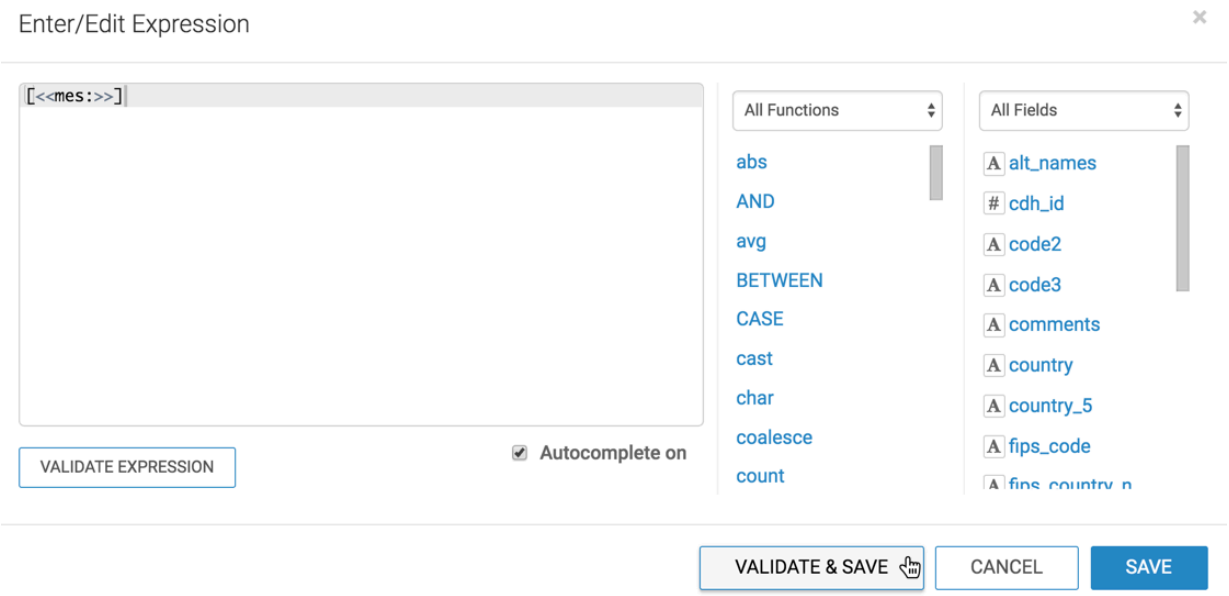

11. Click REFRESH VISUAL.
12. Change the name of the visual to Regional Populations.

13. Click SAVE.

Creating filters to control optional measures

Before you begin

Before starting this work flow, complete the steps in Creating visuals with optional measures on page 8.

Procedure

1. In the dashboard, click the Filters tab.
2. Click CREATE CUSTOM FILTER to create a second dashboard filter.

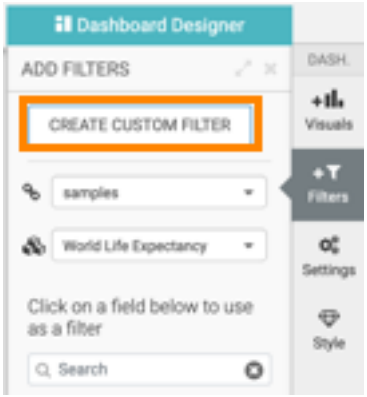

This creates a new filter in the filter area of the application, and opens the Settings modal window for that filter.

3. Click the Values tab and enter the following:
   - Under Title, enter Display Measures.
   - Under Output Parameter, enter mes. Note that this is the parameter from Creating visuals with optional measures on page 8.
   - Under Specified values, enter the following two rows:
     - Value: life_expectancy, Label: Life Expectancy
     - Value: gdp_per_capita, Label: GDP per Capita

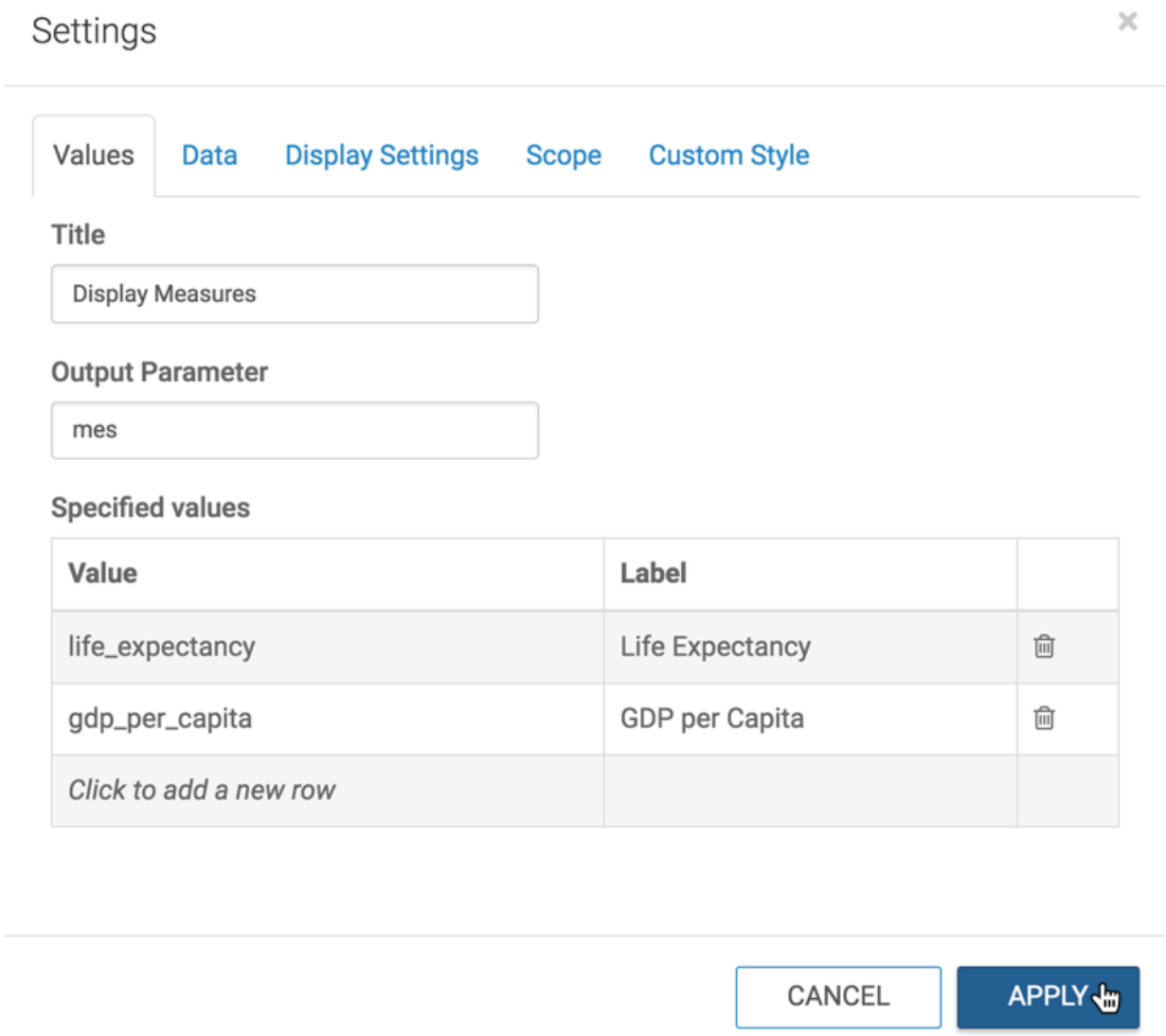

4. In the Settings modal window, switch to the Display Settings tab.
5. Select Allow only one item to be selected at a time
6. Select Include an option for 'All'.
7. Click APPLY.
8. Name and save the dashboard.
9. Switch to application View mode and select World Regional Populations.
10. In the Display Country Details filter, select Country.

11. In the Display Measures filter, select Life Expectancy.

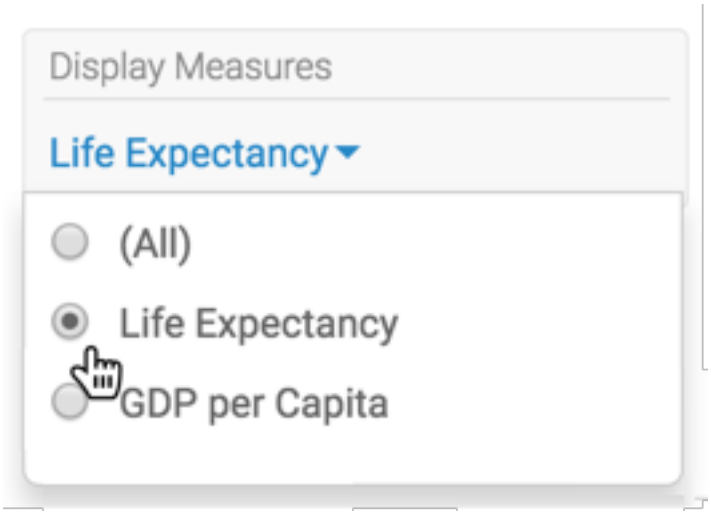

Note that the table now has a new column, life_expectancy.

12. To change the measure that appears in the visual, select GDP per Capita in the Display Measures filter.

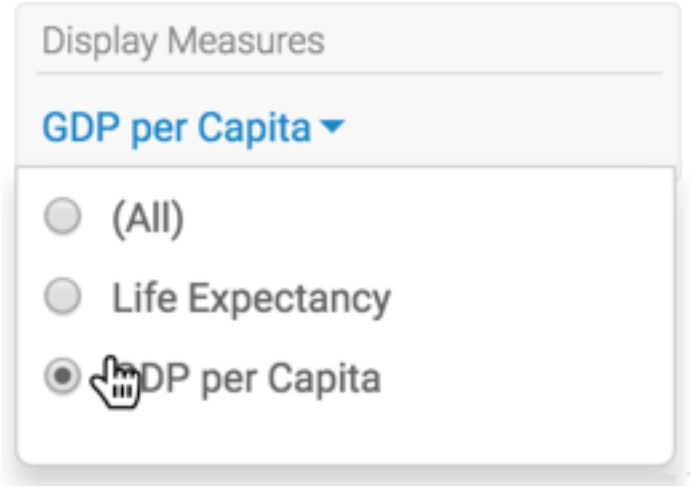

Note that the additional column is now titled gdp_per_capita, not life_expectancy.

To check the parameters of the dashboard, hover the pointer over the Filter icon at the top right corner. They are:

- dim: country
- dim.alias: Country
- mes: gdp_per_capita

You can scroll down to see the mes.alias: GDP per Capita parameter.

13. You can easily navigate between the permutations of filter outputs you create by using filter navigation controls at the top right corner.

Creating visuals with variable dimensions

Procedure

1. Open a new dashboard.

2. Click New Visual.
3. Select the US County Population dataset in the Data menu.
4. Select the Bar Chart visual type in the Visuals menu.
5. Populate the shelves of the visual:
   - From Dimension, select and move Stname field to the X Axis shelf.
   - From Measures, select and move Tot Pop field to the Y Axis shelf.
   - On the Y Axis shelf, change the aggregation of the Tot Pop field from sum(Tot Pop) to avg(Tot Pop): select Tot Pop field, chose the Aggregates menu, and change the aggregate from Sum to Average.
   - On the Y Axis shelf, click Tot Pop, and under the FIELD PROPERTIES menu select Order, and choose Ascending.

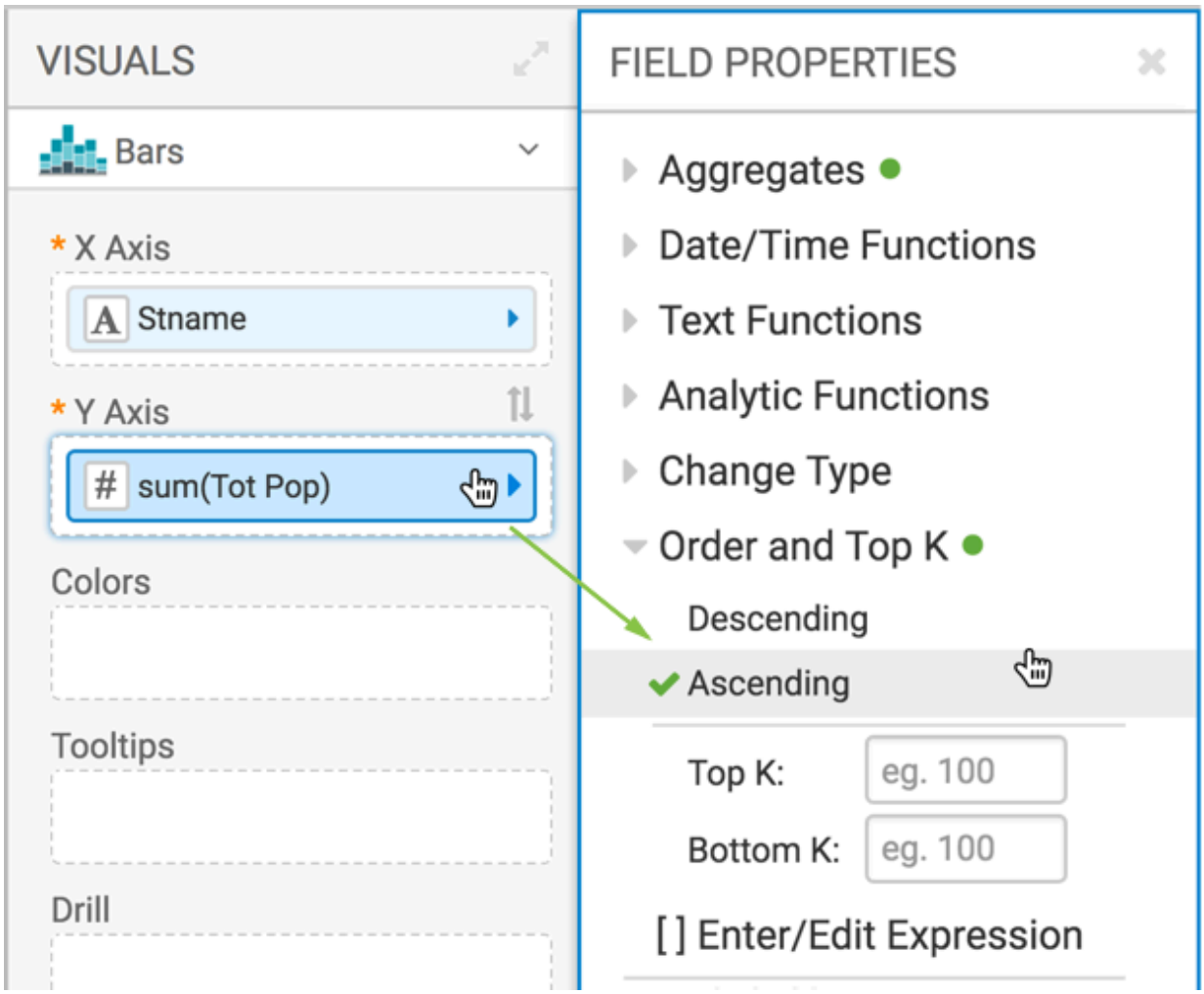

6. Click REFRESH VISUAL to see the basic set up of the bar chart.

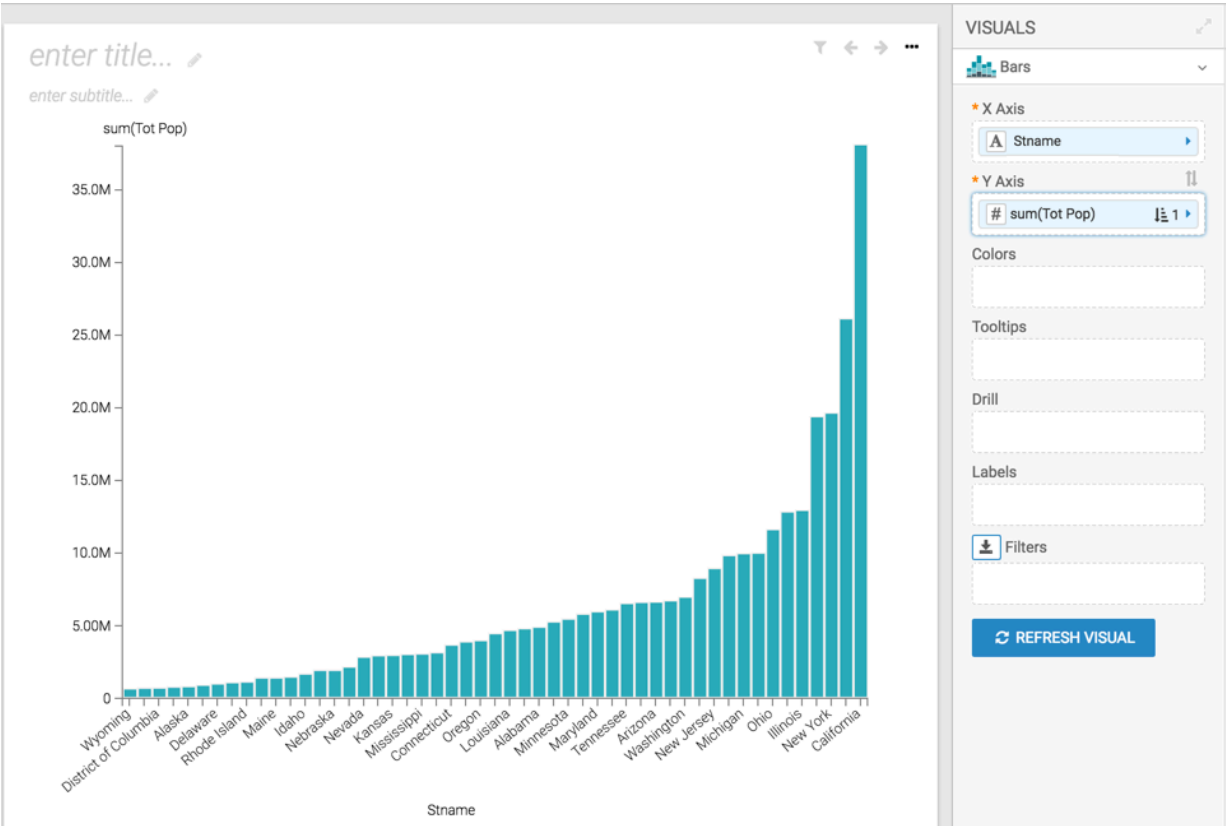

7. On the X Axis shelf, click Stname field.
8. In the FIELD PROPERTIES menu, select [ ] Enter/Edit Expression.

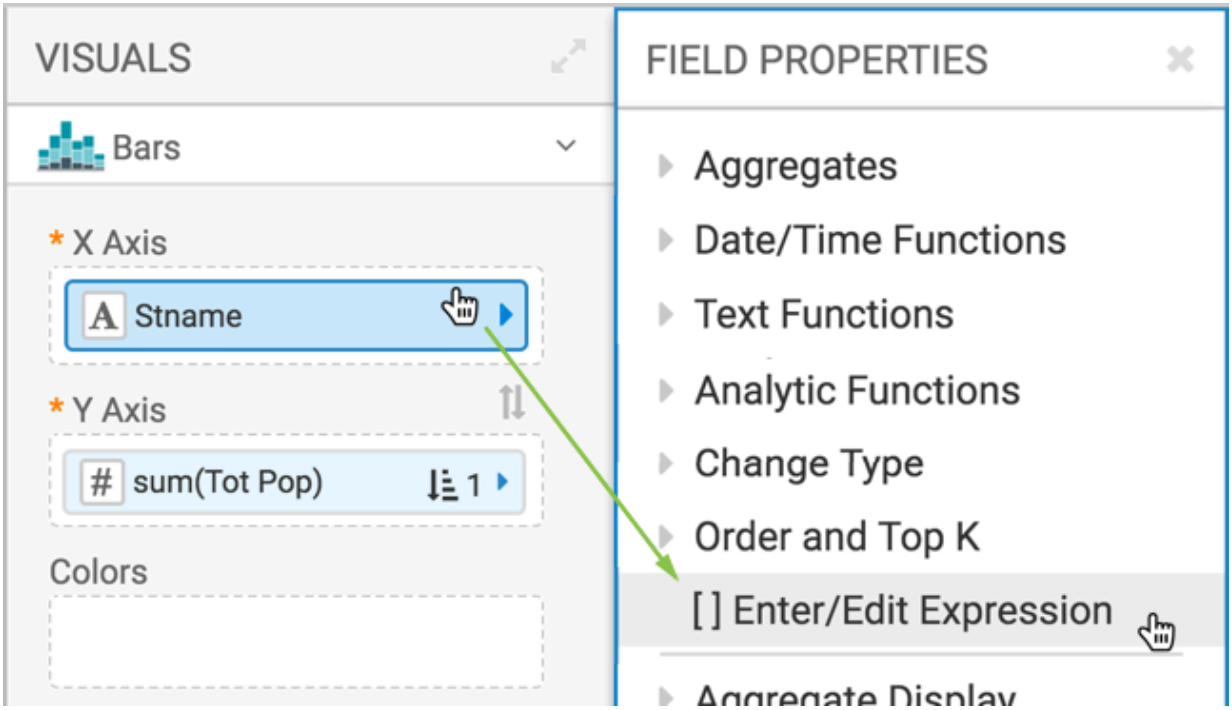

9. In the Enter/Edit Expression modal window, change the text to the following expression: <<dim:[Stname]>>.

10. Click VALIDATE & SAVE.

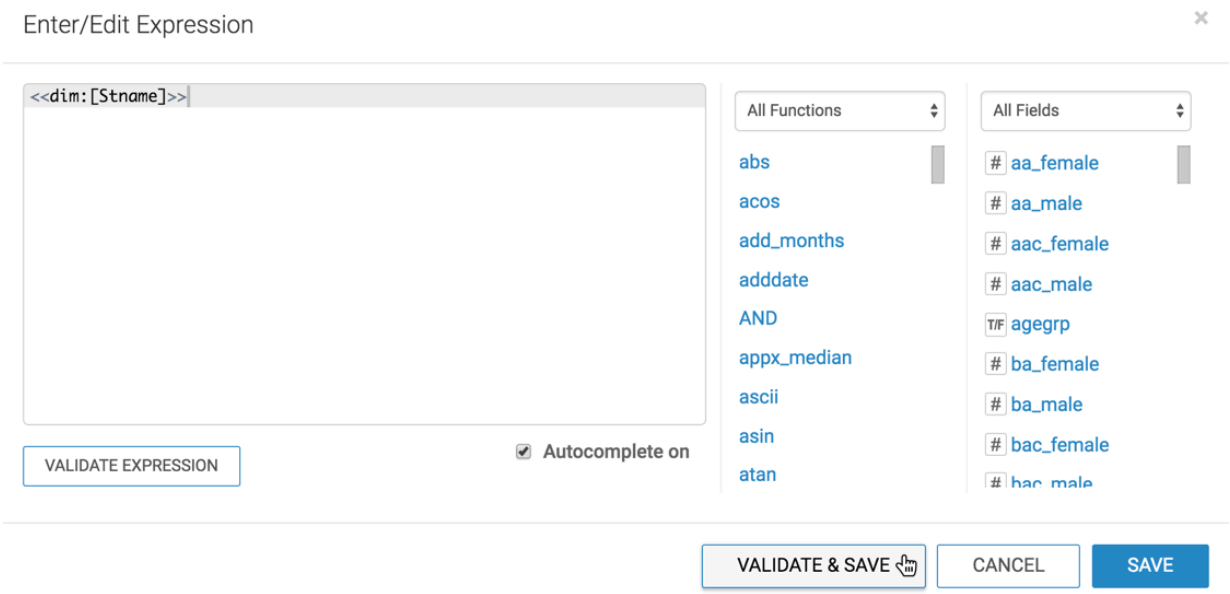

11. Change the name of the visual to Population by <<dim>>.

To have an informative title for the visual, you may add the parameter placeholders to it. The filter configured in Creating filters to control variable dimensions on page 15 supplies the required value.

12. Click SAVE.

Creating filters to control variable dimensions

Before you begin

Before starting this work flow, complete the steps in Creating visuals with variable dimensions on page 12.

Procedure

1. In the dashboard, click the Filters tab.
2. Click CREATE CUSTOM FILTER.

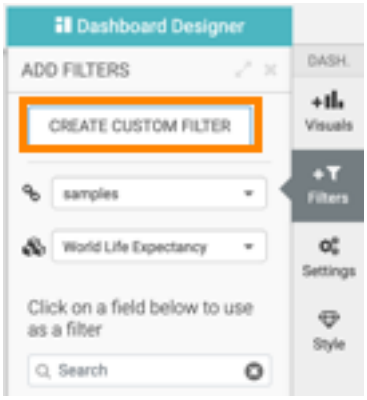

This creates a new filter in the filter area of the application, and opens the Settings modal window for that filter.

3. Switch to the Values tab and enter the following:
   - Under Title, enter Dimension Level.
   - Under Output Parameter, enter dim. Note that this is the parameter from Creating visuals with variable dimensions on page 12.
   - Under Specified values, enter the following two rows:
     - Value: stname, Label: State
     - Value: ctyname, Label: County

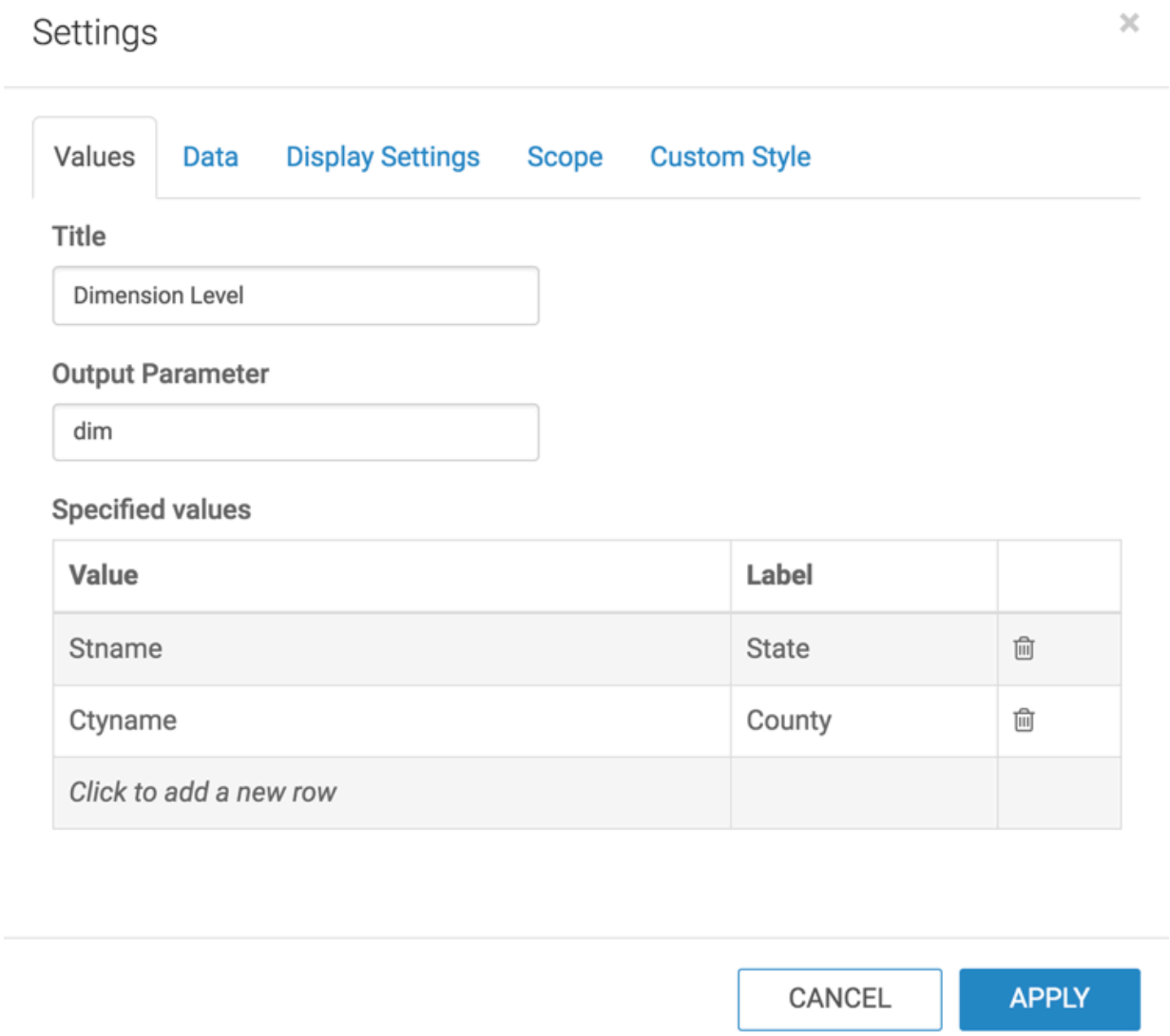

4. Switch to the Display Settings tab and select the Allow only one item to be selected option.

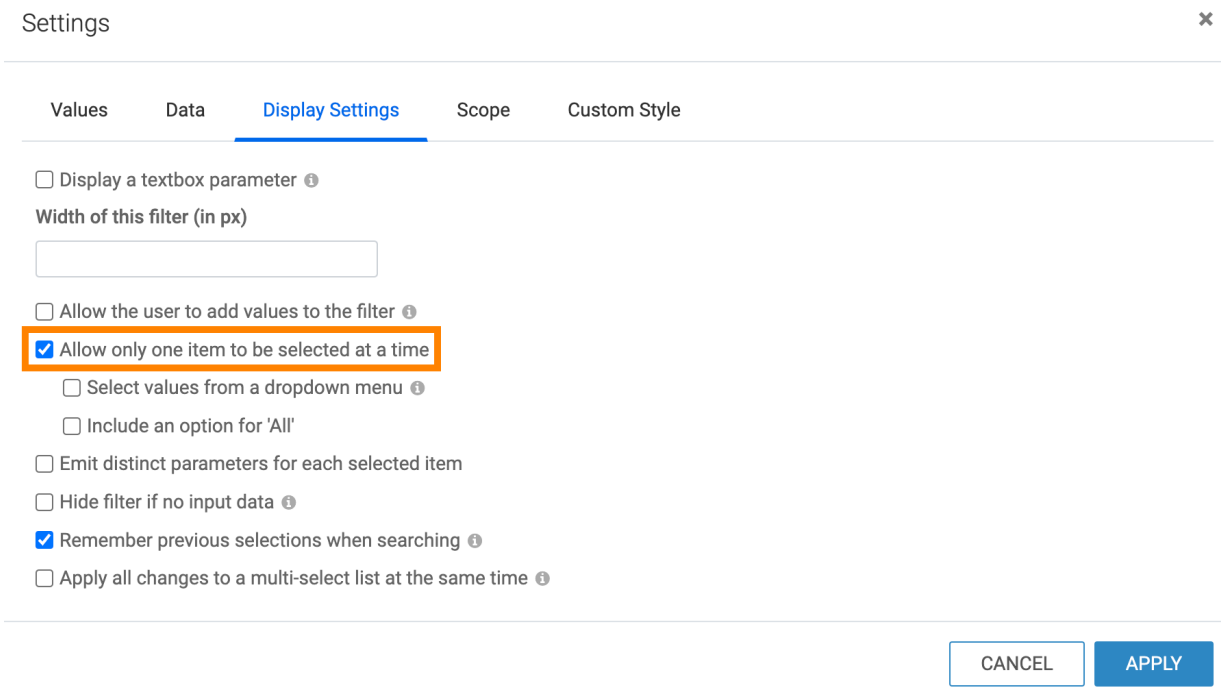

5. Click APPLY.
6. Name and save the dashboard.
7. Switch to application View mode.

   Note that the default choice, Stname, displays both on the horizontal axis, and in the title of the visual.

   To check the parameters of the dashboard, hover the pointer over the Filter icon at the top right corner. They are dim: Stname and dim.alias: State.
8. In the Dimension Level filter, select County.

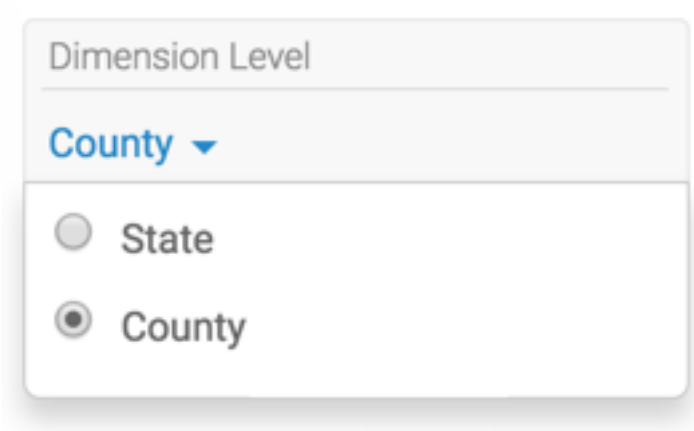

   Note that now the title of the graph and the axis changed to use Ctyname.

   You may also notice that your graph does not appear to have any bars. In this particular dataset, there are large differences among populations of various counties; a great majority has populations under one million, and a select few represent urban areas with extremely dense population.

   In the application, scroll to the extreme right of the visual to see the graph.

   If you want to check the status of parameters on this dashboard, hover the pointer over the Filter icon at the top right corner, and notice that they changed to dim: Ctyname and dim.alias: County.

Creating visuals with variable measures

About this task

You may choose to duplicate the dashboard that you created earlier according to the instructions in Creating visuals with variable dimensions on page 12. In this case, open the visual, and jump to Step 7 in this workflow.

Procedure

1. In the dashboard, click New Visual.
2. Select the US County Population dataset in the Data menu.
3. Select the Bar Chart visual type.n the Visuals menu
4. Populate the shelves of the visual:
   - From Dimension, select and move Stname field to the X Axis shelf.
   - From Measures, select and move Tot Pop field to the Y Axis shelf.
   - On the Y Axis shelf, change the aggregation of the Tot Pop field from sum(Tot Pop) to avg(Tot Pop): select Tot Pop field, chose the Aggregates menu, and change the aggregate from Sum to Average.
   - On the Y Axis shelf, click Tot Pop, and under the FIELD PROPERTIES menu select Order, and choose Ascending.

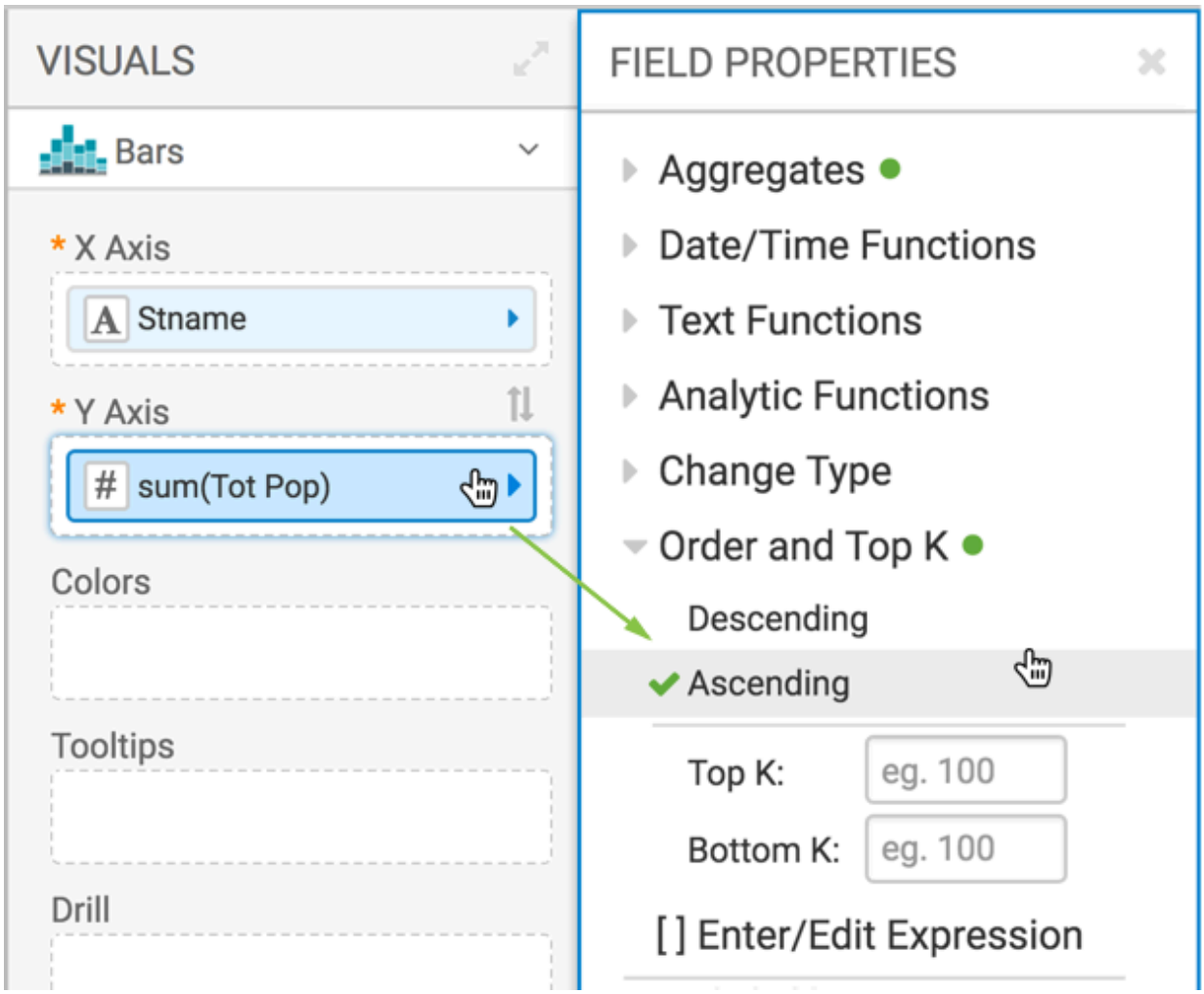

5. Click REFRESH VISUAL to see the basic set up of the bar chart.

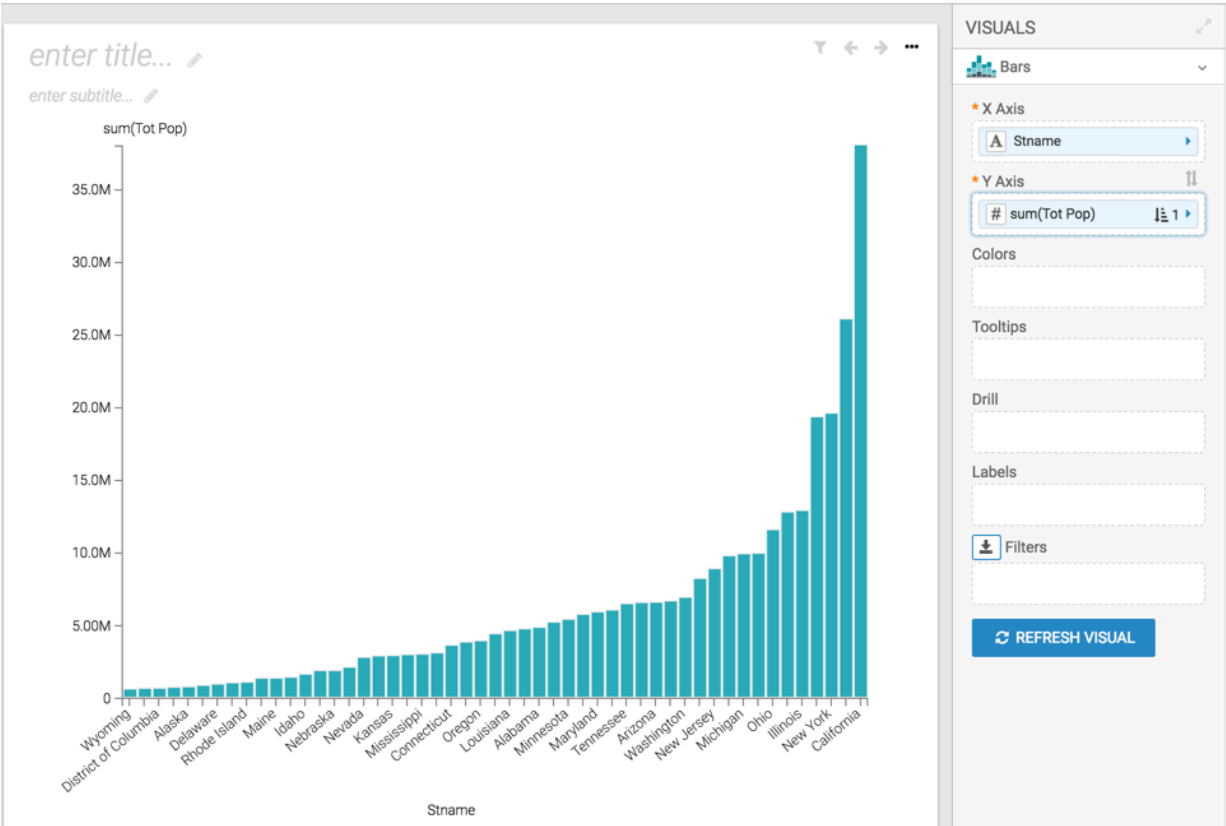

6. On the Y Axis shelf, click the sum(Tot Pop) field.
7. In the FIELD PROPERTIES menu, select [ ] Enter/Edit Expression.

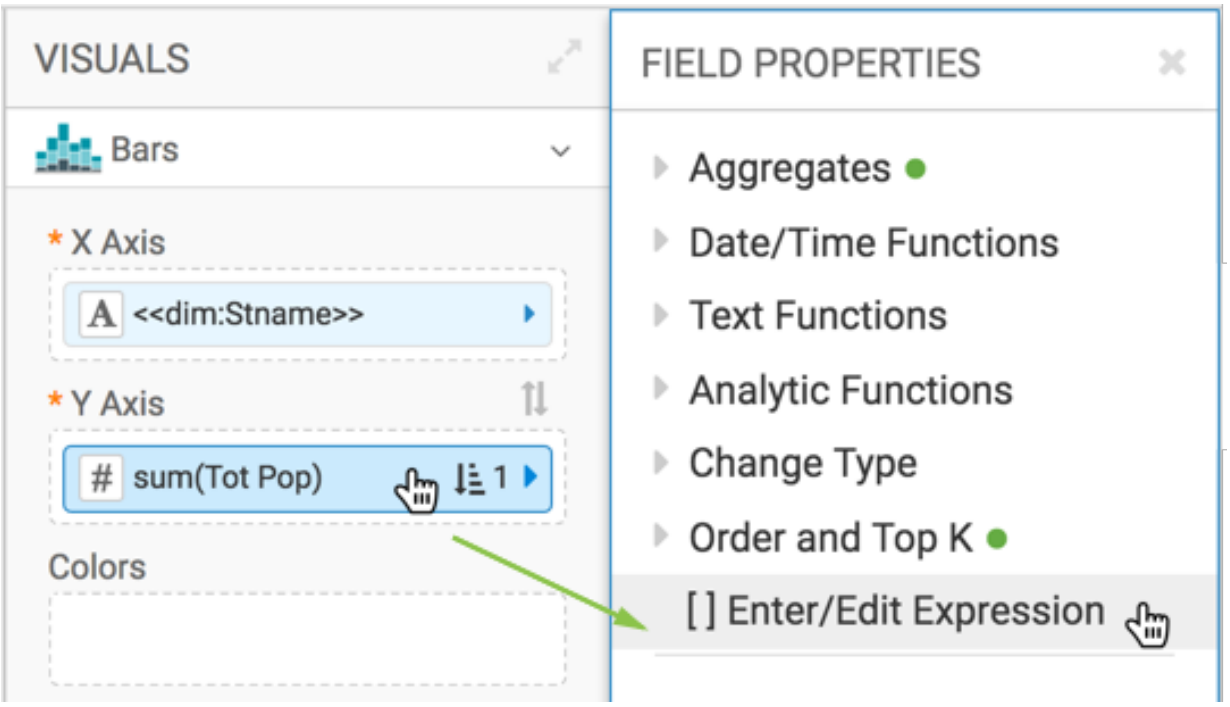

8. In the Enter/Edit Expression modal window, change the text to the following expression: <<agg:sum([Tot Pop])>>.

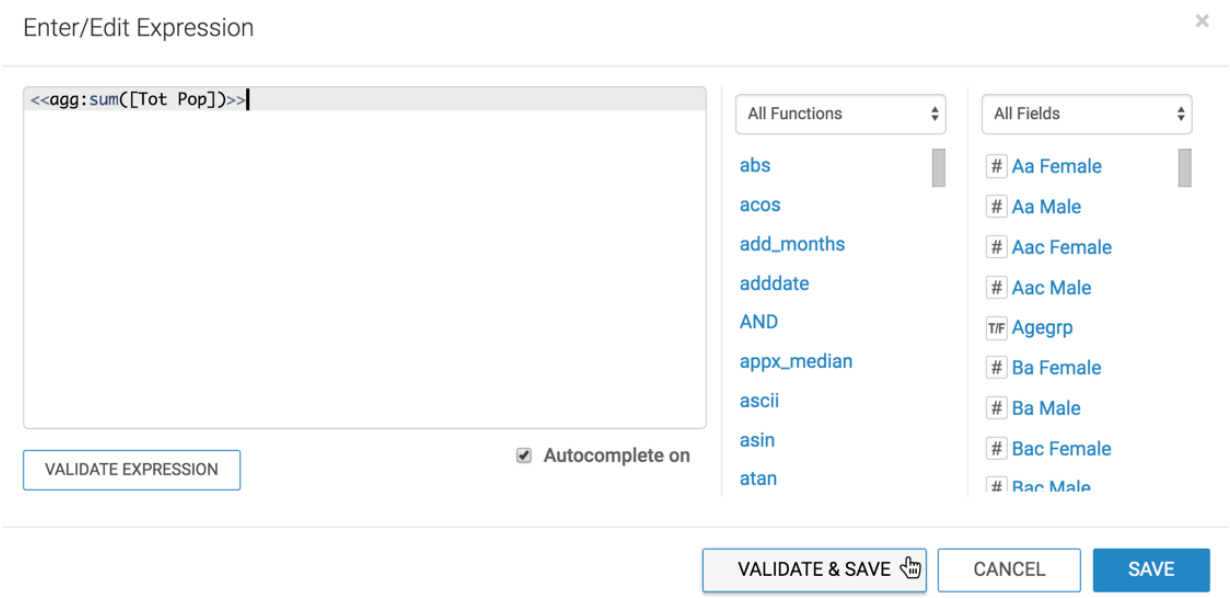

9. Click VALIDATE & SAVE.
10. Change the name of the visual to Population by <<dim>> and <<agg>>.

    To have an informative title for the visual, you may add the parameter placeholders to it. The filters configured in Creating filters to control variable dimensions on page 15 and Creating filters to control variable measures on page 20 supply the required values for <<dim>> and <<agg>>.
11. Click SAVE.

Creating filters to control variable measures

Before you begin

Before starting this work flow, complete the steps in Creating visuals with variable measures on page 18.

Procedure

1. In the dashboard, click the Filters tab.
2. Click CREATE CUSTOM FILTER.

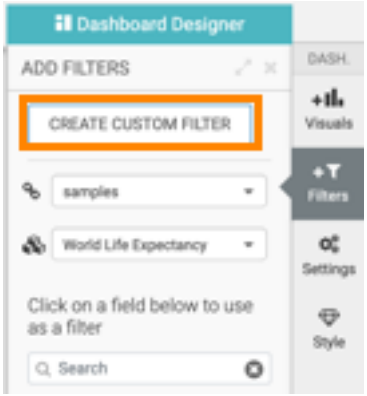

3. This creates a new filter in the filter area of the application, and opens the Settings modal window for that filter.

4. Switch to Values tab and enter the following:
   - Under Title, enter Population Type.
   - Under Output Parameter, enter agg. Note that this is the parameter from Creating visuals with variable measures on page 18.
   - Under Specified values, enter the following two rows:
     - Value: sum(tot_pop), Label: Total Population
     - Value: sum(tot_male), Label: Total Male Population
     - Value: sum(tot_female), Label: Total Female Population

     Note that these are the original field names in the source table.

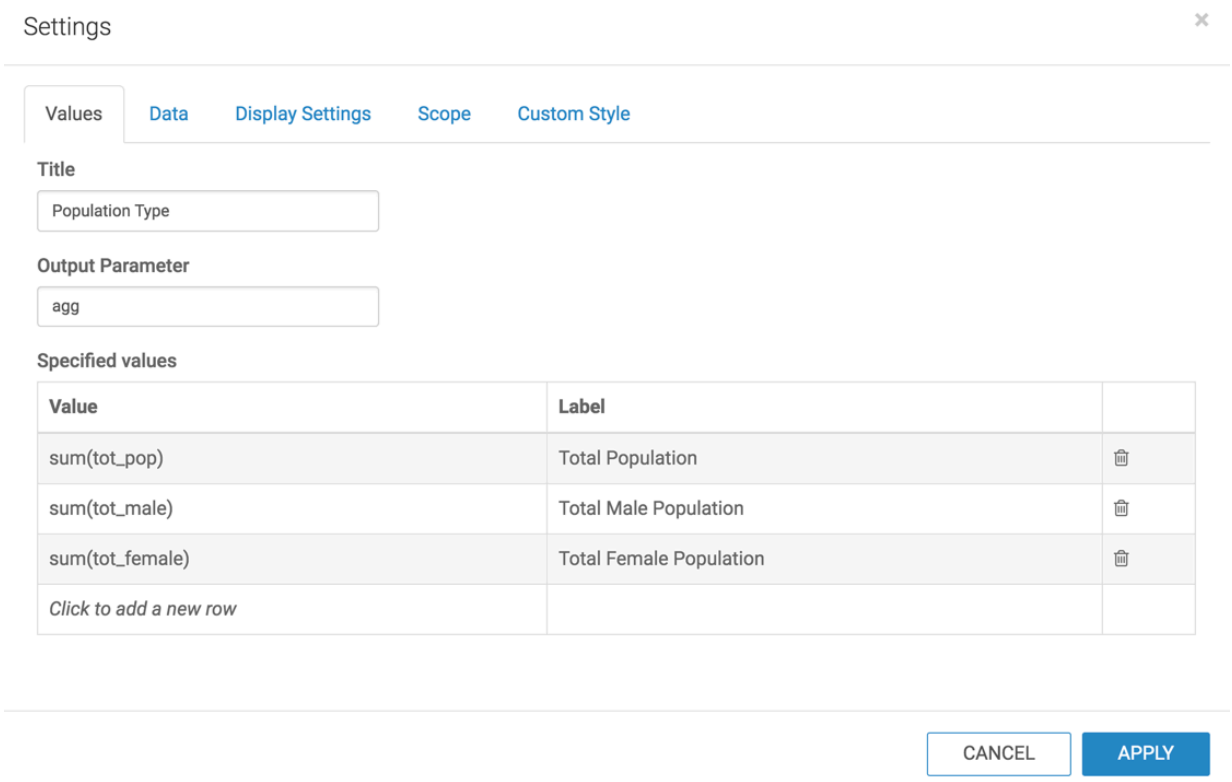

5. Switch to the Display Settings tab and select the options Allow only one item to be selected at a time.
6. Click APPLY.
7. Save the dashboard.
8. Switch to dashboard View mode.

9. In the Population Type filter, select Total Female Population.

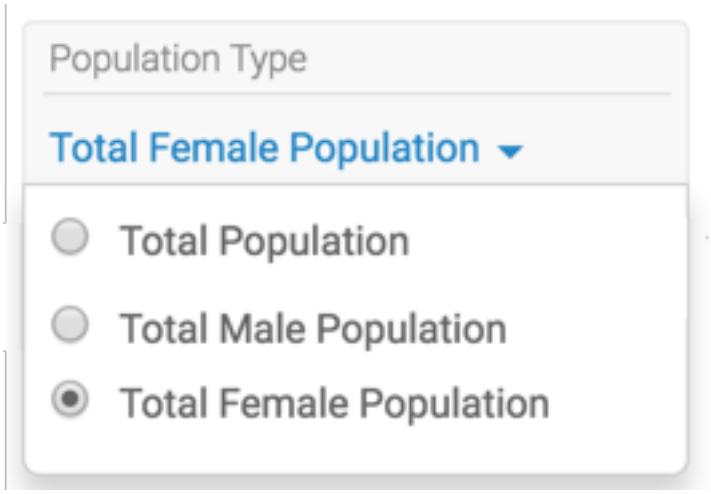

Note that the title of the graph and the vertical axis changed to include sum(tot_female).

10. To check the parameters of the dashboard, hover the pointer over the Filter icon at the top right corner.

They are agg: sum(tot_female), agg.alias: Total Female Population and dim: Stname. You can scroll down to see the dim.alias: State parameter.

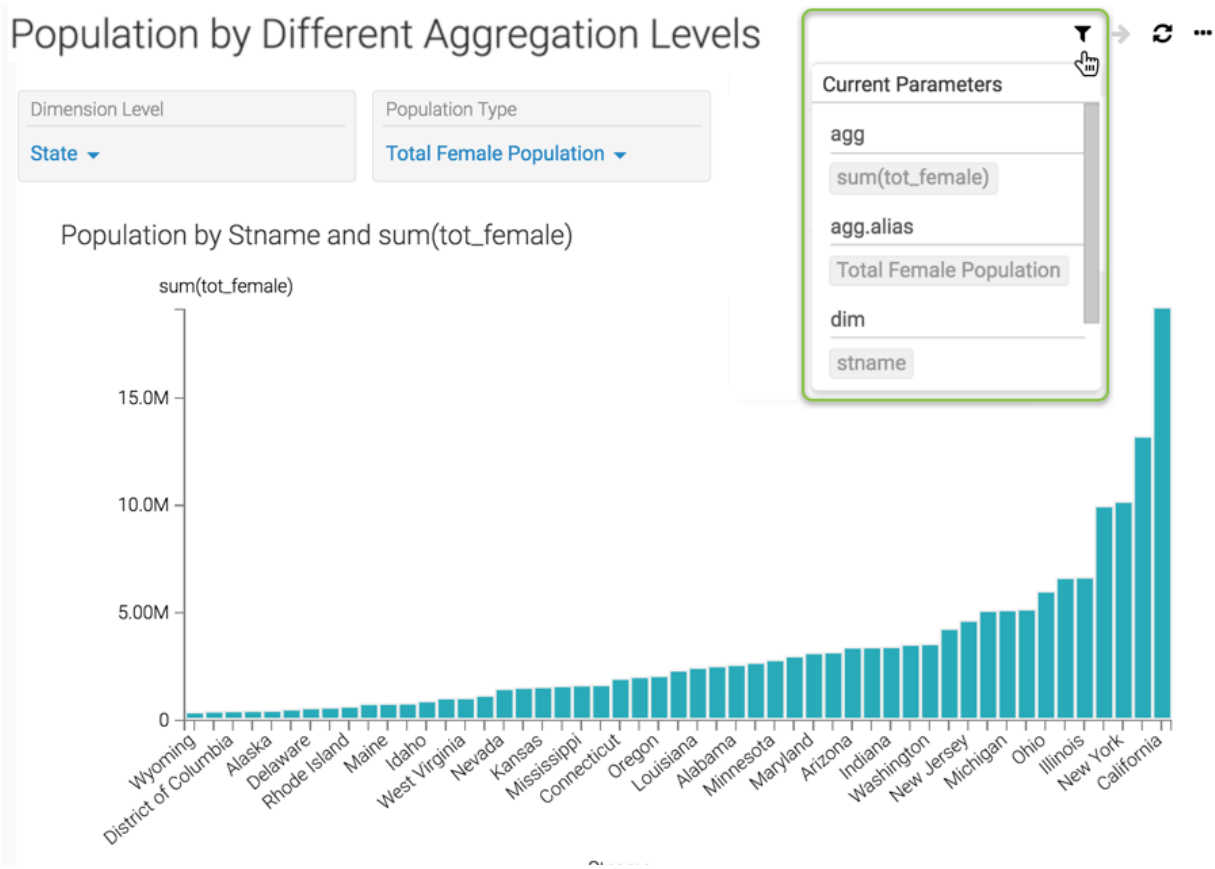

Results

You can now operate the two filters, Dimension Level and Population Type, independently.

You can also navigate between the permutations of filter outputs you create by using filter navigation controls at the top right corner.

Cloudera Data Visualization 7.2.9

Admin API

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Admin API 4

Admin API syntax and general usage 4

Admin API syntax parameters 5

Python Admin API data format and response 6

CURL data format and API key examples 7

Admin API Demo 8

Admin API demo examples 8

Data type details 12

Users 13

Groups 14

Roles 14

Segments 17

Filter associations API 17

Workspaces 18

Admin API

Cloudera Data Visualization provides a URL-based API, the Admin API, which enables access to Cloudera Data Visualization objects like users, groups, roles, datasets, segments, filter associations, connections, and visuals. This allows you to automate deployment operations, such as creating and managing roles, without logging into the Cloudera Data Visualization server console for access to command-line utilities.

Permissions

When using the Admin API , Cloudera Data Visualization enforces the same security policies as when using the graphical user interface. For example, you must have Manage roles and users system-level permission to create new users or update existing users.

Admin API is CRUD

The Admin API supports Create, Read, Update, and Delete (CRUD) operations for basic object access. It does not support complex actions, such as adding tables to datasets.

Related Information

Admin API syntax and general usage
Admin API syntax parameters
Python Admin API data format and response
CURL data format and API key examples
Admin API Demo
Admin API demo examples
Data type details
Users
Groups
Roles
Segments
Filter associations API
Workspaces

Admin API syntax and general usage

The Cloudera Data Visualization Admin API has a consistent pattern for each data type.

The Admin API has the following basic access syntax:

```
[http | https]:/host:port/arc/adminapi/version/data_type[/object_id][?
options]
```

The parameters of this line are in Admin API syntax parameters.

HTTP operations

The HTTP method specifies the following operation types:

GET

List of an item identified through the object_id or object_name, or all items, with default summary information. The URL option 'detail=true' returns all data details.

POST

Update: The fields in the request data update the item with matching object _id.

Create: If the URL or request data does not specify the object_id, ArcViz creates a new data item.

Validate: To verify that the connection is successful, issue the POST command two times:

1. Issue the POST command with the validate flag set to true.

```
[
  {
   "id":18,
   "name": "ArcEngine Dev",
   "type": "arcengine",
   "validate": "true",
   "info": {
   "PARAMS": {
   "HOST": "localhost",
   "PORT": "21051",
   "USERNAME": "admin",
   "SETTINGS": {
   "ANALYTICAL_VIEW_MAX_REFRESH_THREADS": "1",
   "MAX_PARTITIONS_FOR_REFRESH_INSERT": "1"
                }
.....
.....
             }
          }
  }
]
```

2. On success, issue the same POST command without the validate flag. This step saves the data.

DELETE

Delete the specified item.

HTTP access through Python

While you can use all standard HTTP access methods, we recommend the python request modules approach for HTTP operations. Note the following common setup:

```
import json
import requests
api_url = [http|htttps]://host:port/arc/adminapi/version
login_url = [http|https]:/host:port/arc/apps/login
```

Related Information

Admin API syntax parameters

Admin API syntax parameters

The Cloudera Data Visualization Admin API has a consistent pattern for each data type.

The Admin API has the following basic access syntax:

```
[http | https]:/host:port/arc/adminapi/version/data_type[/object_id][?
options]
```

host

The host of the Cloudera Data Visualization instance.

port

The port of the Cloudera Data Visualization instance.

version

The current API version is v1. This increments if the item data format changes. Whenever possible, we intend to support older versions for backward compatibility.

data_type

One of the Cloudera Data Visualization artifacts: users, groups, roles, datasets, connections, visuals, segments, filter associations, or workspaces.

object_id

The id of the individual object, such as a specific user, visual, or a specific dataset. You can either use the object_id or the object_name in the syntax, not both.

object_name

The name of the individual object, such as a specific user, visual, or a specific dataset. You can either use the object_id or the object_name in the syntax, not both.

options

Further request options, such as level of information details when 'detail=true'.

api_url

The address of the API management system, in the form [http|htttps]://host:port/arc/adminapi/version.

login_url

To authenticate the end user, address of the login URL, in the form [http|https]:/host:port/arc/apps/login.

Python Admin API data format and response

Cloudera Data Visualization provides examples of python Admin API data format and response.

The response data for GET operations is a list of JSON items. For POST operations, such as UPDATE and CREATE, the input format is a structure with a data field that contains the JSON list of items. The UPDATE and CREATE operations process one item at a time, so the list is exactly one entry long.

The response data for POST operations is the updated item, matching a GET with detail=1 for the item, as demonstrated in Example 1.

For item updates, it is only necessary to specify the fields you are updating. Cloudera Data Visualization merges the supplied fields in the input data with the existing item's data, as demonstrated in Example 2.

Example 1: Setting the name for role ID=1 to 'System Admin'

```
payload = {'name':'System Admin'}
session.post(api_url + '/roles/1', data={'data':json.dumps([payload])})
```

Note that the API URL has the following form:

```
[http|htttps]://host:port/arc/adminapi/version
```

For syntax of other parameters, see Admin API syntax parameters.

Example 2: Checking role ID=2; updating by adding a new user

```
response = session.get(api_url + '/roles/2?detail=1')
```

```
role = response.json()[0]
if 'new_user' not in role['users']:
    payload = {'users':role['users'] + ['new_user']}
    session.post(api_url + '/roles/2', data={'data':json.dumps([payload])})
```

For the definition of fields for each data type, see Data type details.

Related Information

Admin API syntax parameters

Data type details

CURL data format and API key examples

Cloudera Data Visualization provides examples of API Key in CURL data format.

When you add the APIKey to the request header and avoid explicit login, all interactions become simpler. The examples in this article use an APIKey obtained through the Manage API Keys interface, on the host:port/arc/apps/apikeys browser page of the DataViz installation The actual APIKey and the method of retrieving the key depends on the user system.

See Example 1 to learn how to get all roles information and Example 2 to change the description of a role.

Example 1: Getting all roles

To use CURL to obtain a full dump of all roles, use the following command.

Note that the output is piped to the standard python JSON dumper for easier reading; it is not necessary for CURL access.

```
curl -s \
 -X GET \
 -H "Authorization: apikey ApiKey" \
 api_url/roles?detail=1> | python -m json.tool
```

Note that the login URL has the form [http|https]:/host:port/arc/apps/login. For syntax of other parameters, see *Admin API syntax parameters*.

Example 2: Changing the role description

To change the description for role ID 3, use the following command.

For CURL to supply data through the POST method, use the application/x-www-form-urlencoded content type.

```
curl -s \
 -X POST \
 -H "Content-Type: application/x-www-form-urlencoded" \
 -H "Authorization: apikey ApiKey" \
 -d 'data=[{"desc":"Updated description again"}]' \
 api_url/roles/3
```

For the definition of fields for each data type, see Data type details. For syntax of other parameters, see *Admin API syntax parameters*.

Related Information

Admin API syntax parameters

Data type details

Admin API Demo

This is a simple GUI demo to demonstrate how the Cloudera Data Visualization URL-based Admin API can easily display the data format for each item type.

Note:

- Use the demo with some caution: the system does not confirm DELETE operations, and it is not meant for implementing system configuration changes. It is simply a working example of the Admin API.
- The demo is not enabled by default. It must be enabled, much as the actual Admin API URL support must be enabled, for each item.

Use the following statement to enable the demo:

```
ADMIN_API_DEMO_LIST  = ['visuals', 'datasets', 'connections', 'users', 'grou
ps',
  'roles', 'segments', 'filterassociations']
```

Alternatively, use the wild card to specify all options:

```
ADMIN_API_DEMO_LIST  =  ['*']
```

To fully enable all APIs and all demo tabs, use the wild card character in the settings entries for both ADMIN_AP I_DEMO_LIST and ADMIN_API_URL_LIST:

```
ADMIN_API_DEMO_LIST = ['*']
ADMIN_API_URL_LIST  = ['*']
```

After enabling the demo, you can access it through the following URL:

```
host_path/arc/apps/apidemo
```

Admin API demo examples

Cloudera Data Visualization's URL-based Admin API provides a straightforward way to display the data format for each item type.

The demo has a tab for each enabled item type, and supports all the API functions: Fetch (one), Fetch All, Create, Update, and Delete.

The following examples demonstrate some of the functions available through the demo.

Fetching all user information

To get information on all users, click Fetch. Cloudera Data Visualization returns the list of registered users, as shown in the following screenshot.

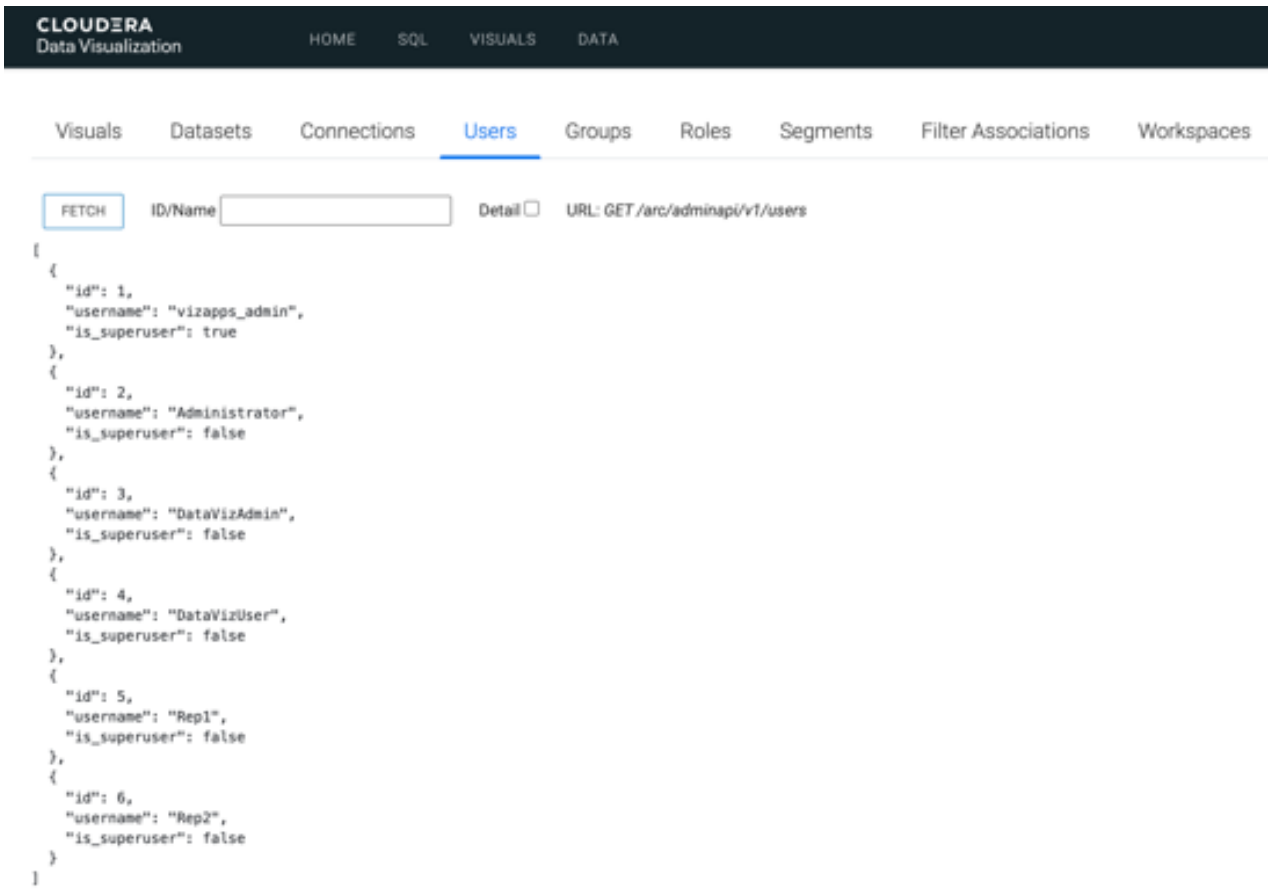

Fetching single user information, with detail

To extract information on a single item using its ID or name, follow these steps:

1. Under the Users tab, enter the ID.
2. Select the Detail option to get the full data for that item, in this case user ID 1.
3. Click Fetch.

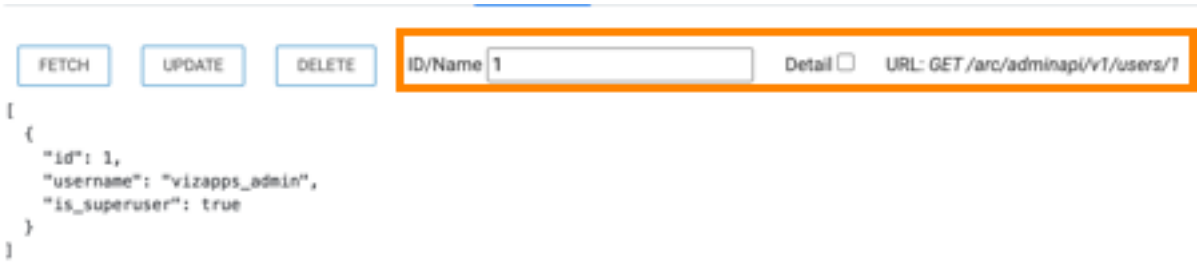

Cloning an item

When you clone an item, the resulting screen shows a duplicate of the item but clears the ID. field. For more information, see Creating a new item on page 10.

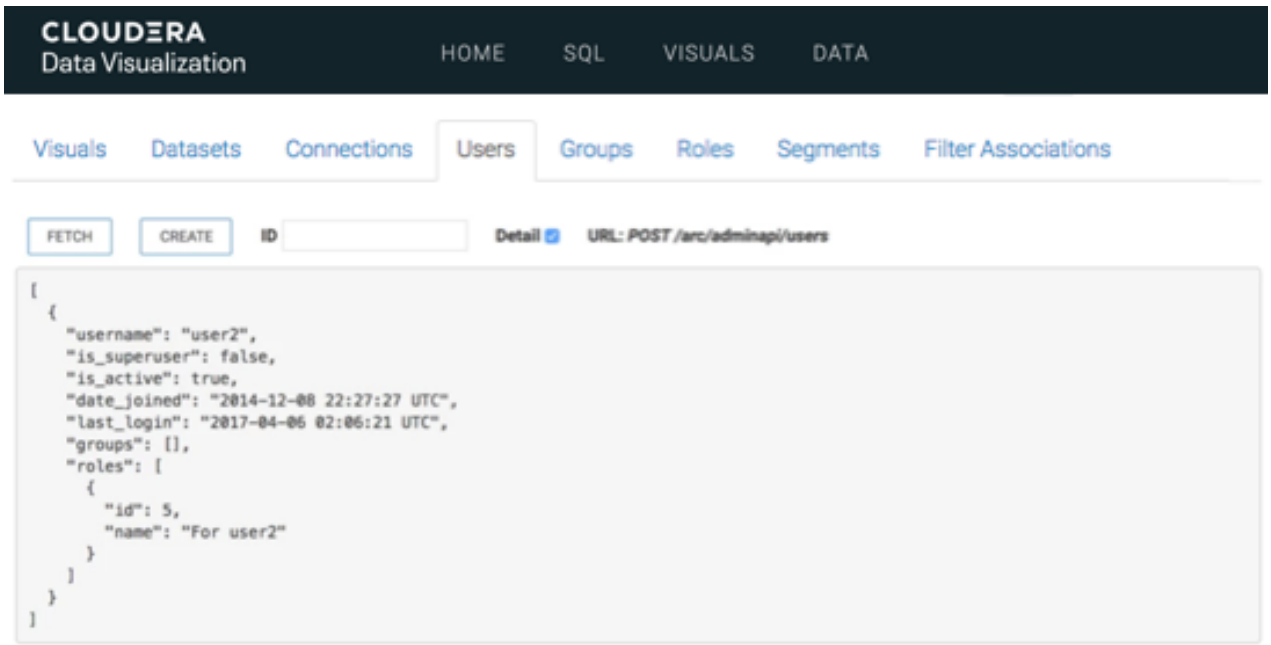

Creating a new item

If you click CREATE immediately after cloning an item, Cloudera Data Visualization returns an error because a user with this name already exists.

To create a new user, change the username and provide an initial password. Here is an example in which a new user was created by changing the username and adding an extra line that specifies the password:

```
"username": "user2-copy",
"password": "initial-pw",
```

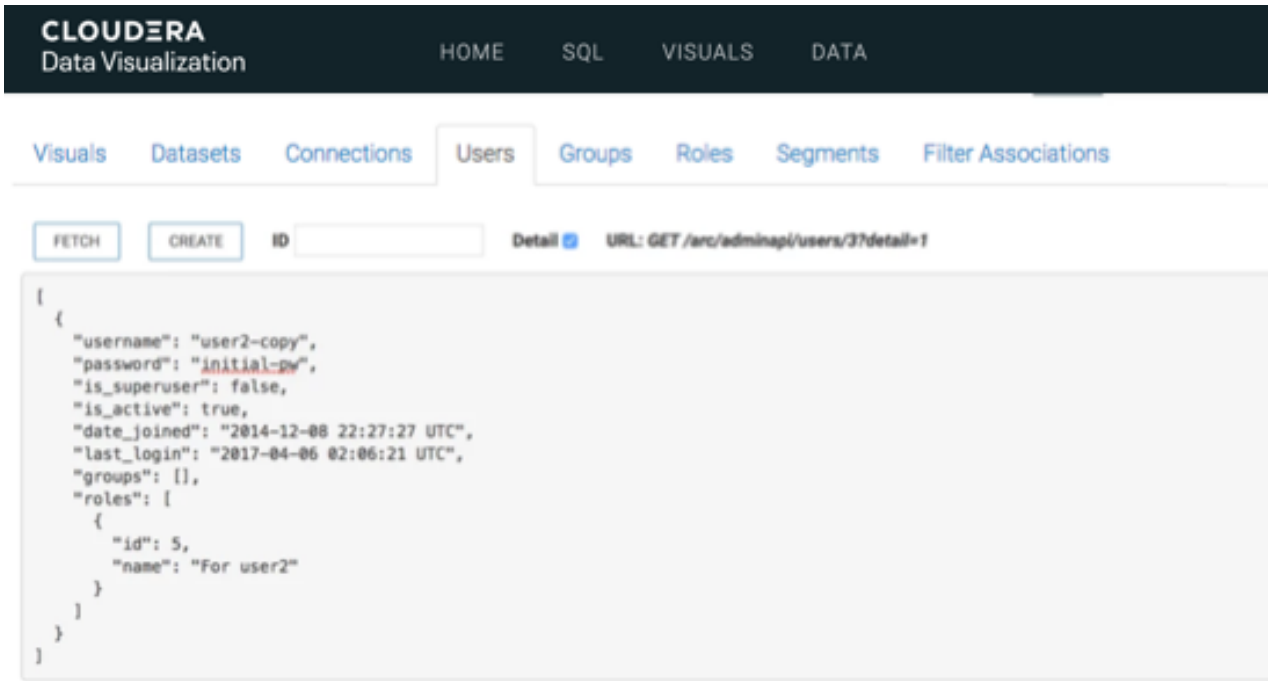

Click Create again. Cloudera Data Visualization notifies you that the update is successful and refreshes the interface to show the results, consistent with fetching detailed information for a specific user, as described in Fetching single user information, with detail on page 9.

Note: For security reasons, the password is not included in the fetched user details.

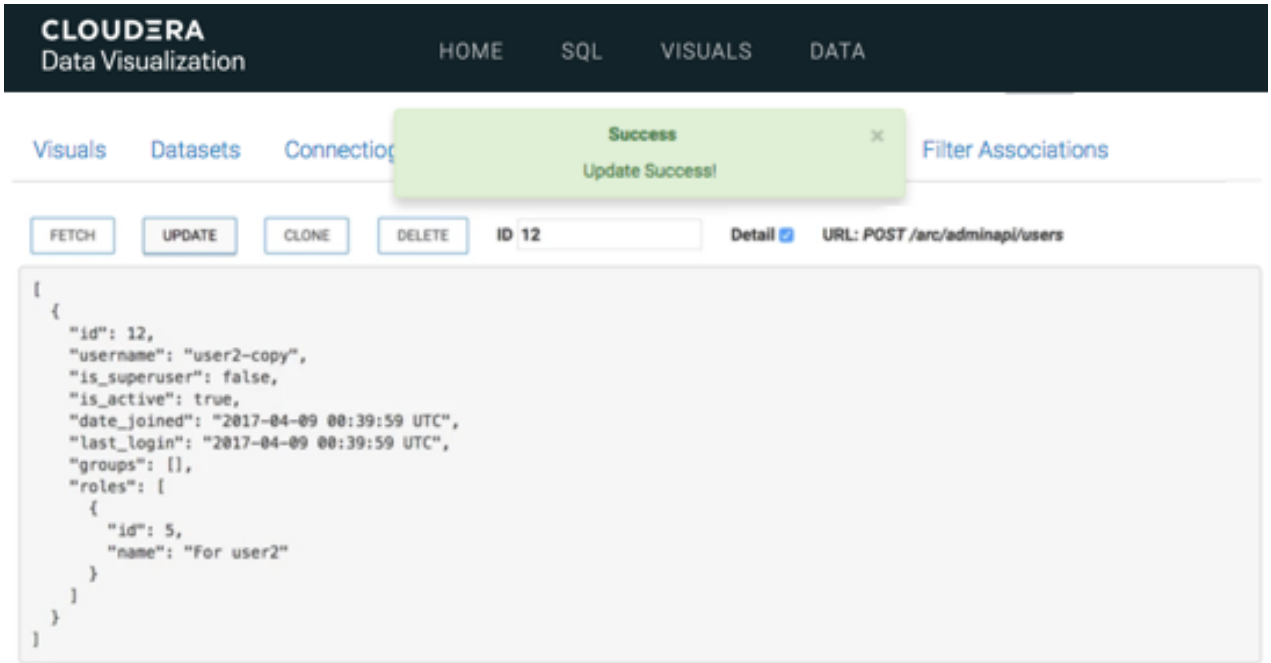

Changing passwords

To change a user's password, you must supply both the current password (password) and the new password (new_ password).

Edit the user detail by adding the following lines of code and click UPDATE.

```
"password": "initial-pw",
"new_password": "updated-pw",
```

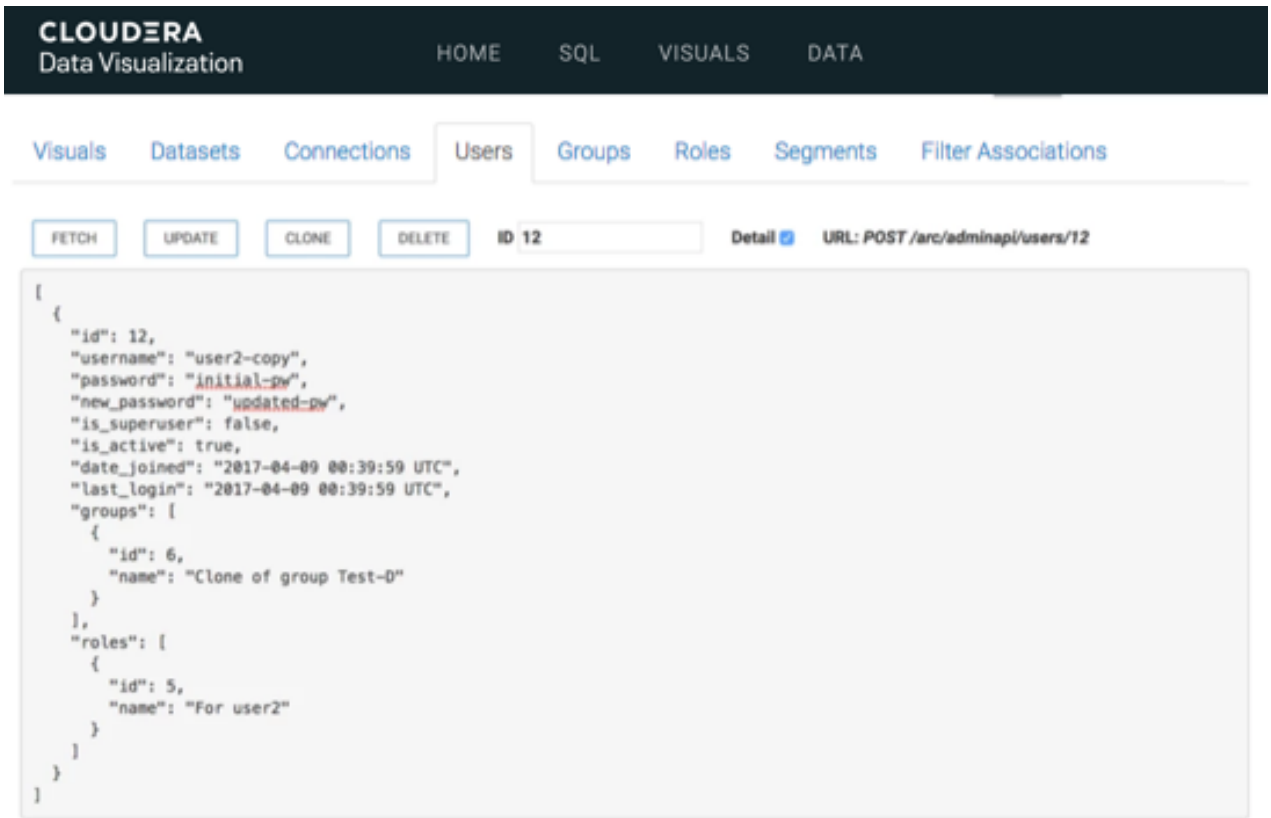

Data type details

Cloudera Data Visualization provides URL access to the Cloudera Data Visualization server objects.

The Admin API uses specific JSON definitions for each data type:

- Users
- Groups
- Roles
- Segments
- Filter associations API
- Workspaces

Note that we deliberately chose not to document the details of creating datasets, connections, and visuals. They are all highly complex structures, and should be created directly in the Cloudera Data Visualization application, through the graphical user interface.

For the GET requests, many returned fields are only informational, and cannot be updated through subsequent POST requests.

Some fields are themselves complex structures that contain sub-fields. The update logic that merges the supplied input fields with the existing data applies to the top-level fields only. For example, the role type contains a privs field, which is a list of individual privilege records. To update the privs field, you must supply the entire list, not merely the individual list element.

Related Information

Users

Groups

Roles

Segments
Filter associations API
Workspaces

Users

Cloudera Data Visualization provides URL access to the Cloudera Data Visualization server object, users.

When creating a new user, you must supply the password field. You must also supply the password field when updating a user's password. Like in the GUI, a regular user (non-admin) can only change their own password, and must supply the current password in the old_password field.

Supplying None (or null in the API demo page, the javascript version of None) for the password makes the user account unusable for login.

For the list of groups and roles, the name is supplied for information only. When updating the users's groups or roles, only the ID fields are necessary.

The examples in this article use an API Key obtained through the Manage API Keys interface, on the host:port/arc/apps/apikeys browser page of the DataViz installation. The actual API Key and the method of retrieving the key depends on the user system.

Here is a CURL example for setting the roles for user ID=3 to IDs 5, 7, and 8.

```
curl -s \
 -X POST \
 -H "Content-Type: application/x-www-form-urlencoded" \
 -H "Authorization: apikey api_key" \
 -d 'data=[{"roles": [{"id":5}, {"id":7}, {"id":8}] }]' \
 api_url + '/users/3'
```

Note that the API URL has the form [http|htttps]://host:port/arc/adminapi/version.

The JSON fields for user's data type are defined as follows:

Table 1: JSON Fields for Users Data Type

Field	Detail Only	Updatable	Description
id	No	No	User ID
username	No	Yes	Username, limited to alphanumeric, period, underscore, and dash
is_superuser	No	No	Indicates the admin userm, who has full permissions
is_active	Yes	No	TRUE: user can long through the UI FALSE: user cannot login through the UI, but trusted authentication works
date_joined	Yes	No	Shows the creation date for this user's metadata entry
last_login	Yes	No	Shows the last login date for this user
groups	Yes	Yes	List of groups to which this user belongs; each entry shows group ID and group name
roles	Yes	Yes	List of roles to which this user belongs; each entry shows role ID and role name

Groups

Cloudera Data Visualization provides URL access to the Cloudera Data Visualization server object, groups.

Like the users data type, the names for list of groups and roles are supplied for information only. When updating the users's groups or roles, only the ID fields are necessary.

The JSON fields for group's data type are defined as follows:

Table 2: JSON Fields for Groups Data Type

Field	Detail Only	Updatable	Description
id	No	No	Group ID
name	No	Yes	Group name
users	Yes	Yes	List of users in this group; each entry shows the user id and username
roles	Yes	Yes	List of roles to which this group belongs; each entry shows role ID and role name

Roles

Cloudera Data Visualization provides URL access to the Cloudera Data Visualization server object called roles.

To support installations that store the users and groups information outside the Cloudera Data Visualization environment (such as LDAP), the role membership lists for users and groups only store names. During role update and create operations, Cloudera Data Visualization accepts user and group names 'as is', without validating them.

Each entry in the privs list corresponds to a single privilege row on the Cloudera Data Visualization role edit page. Each row contains fields for the privilege type (ptype), an identifier section, and a list of permissions (perms) for the identified objects, such as datasets or data connections. Each privilege type has a specific identifier, and set of possible permissions. Cloudera Data Visualization stores the dataset IDs and connection IDs within the identifier sections as a STRING and uses the special value -1 to indicate 'All dataset' or 'All connections.

Roles data type

The JSON fields for the roles' data type are defined as follows:

Table 3: JSON fields for the roles data type

Field	Detail Only	Updatable	Description
id	No	No	Role ID
name	No	Yes	Role name
desc	No	Yes	Role description
users	No	Yes	List of usernames that belong to this role
groups	No	Yes	List of groups that belong to this role
privs	Yes	Yes	List of privilege structures for this role, as described in Privileges types on page 14

Privileges types

The Cloudera Data Visualization Role-Based Access Control system supports the following permission types:

- *ptype: "system"*
  Identifier

  None

  Permissions

Permission name	Description
sys_editperm	Manage roles and users
sys_styles	Manage styles and settings
sys_viewlogs	View query logs
sys_editconn	Manage data connections

- *ptype: "dataconn"*
  Identifier

Field name	Description	Example
dclist	List of data connection IDs, or -1 for 'All data connections'	"dclist" : ["-1"]

  Permissions

Permission name	Description
dc_aviews	Manage analytical views
dc_upload	Import data
dc_expore	Create datasets and explore tables

- *ptype: "dataset"*
  Identifier

Field name	Description	Example
dcid	Data connection ID for this privilege, or -1 for 'All'	"dcid" : "-1"
dslist	List of dataset IDs for this privilege	"dslist" : ["1", "2", "3"]

  Permissions

Permission name	Description
dc_aviews	Manage analytical views
dc_upload	Import data
dc_expore	Create datasets and explore tables

Creating roles

The following code creates a new role with groups dataconn_managers and arcviz_admins. The role has system-level permissions to view logs, and to create new datasets. It also has full permissions on all connections and all datasets.

The actual API Key and the method of retrieving the key depends on the user system.

```
curl -s \
 -X POST \
 -H "Content-Type: application/x-www-form-urlencoded" \
 -H "Authorization: apikey api_key" \
 -d 'data=[{
      "name": "Connection manager",
      "desc": "Data connection management",
      "groups": ["dataconn_managers", "arcviz_admins"],
```

```
      "privs": [
          {"ptype": "system",
           "perms": ["sys_viewlogs", "sys_editconn"]
          },
          {"ptype": "dataconn",
           "dclist": ["-1"],
           "perms": ["dc_aviews", "dc_upload", "dc_explore"]
          },
          {"ptype": "dataset",
           "dcid": "-1",
           "dslist": ["-1"],
           "perms": ["ds_manage", "ds_appedit", "ds_appview"]
          }
        ]
    }]' \
 127.0.0.1:7999/arc/adminapi/roles
```

When viewing this role through the Cloudera Data Visualization user interface, it appears on the edit page like this:

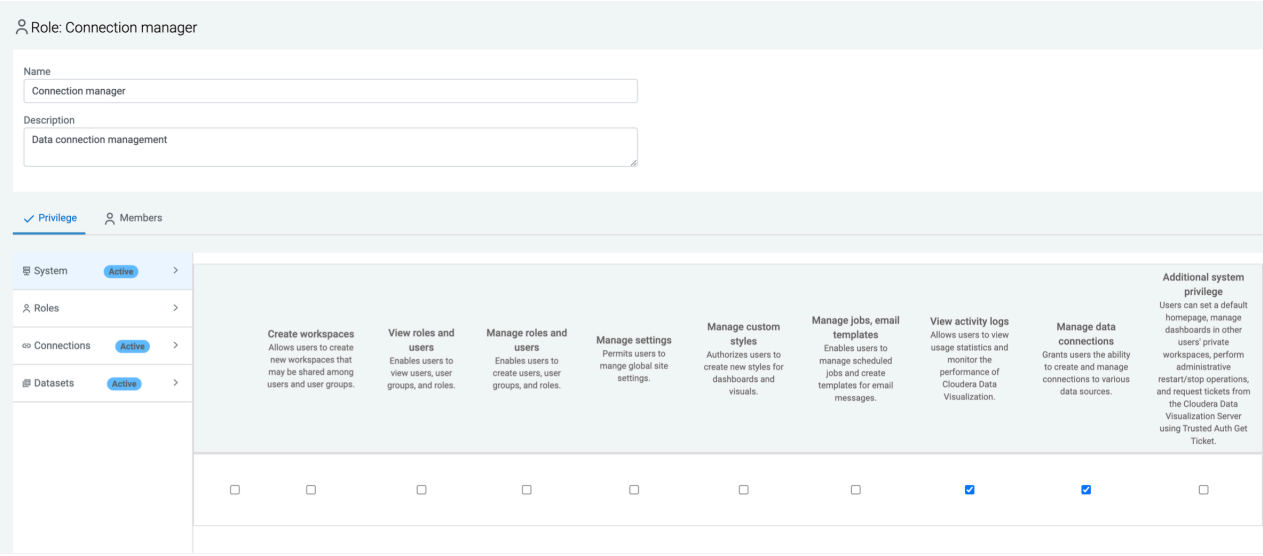

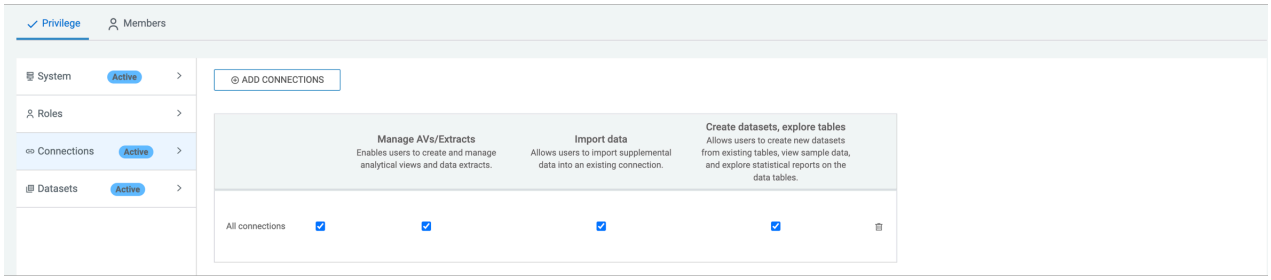

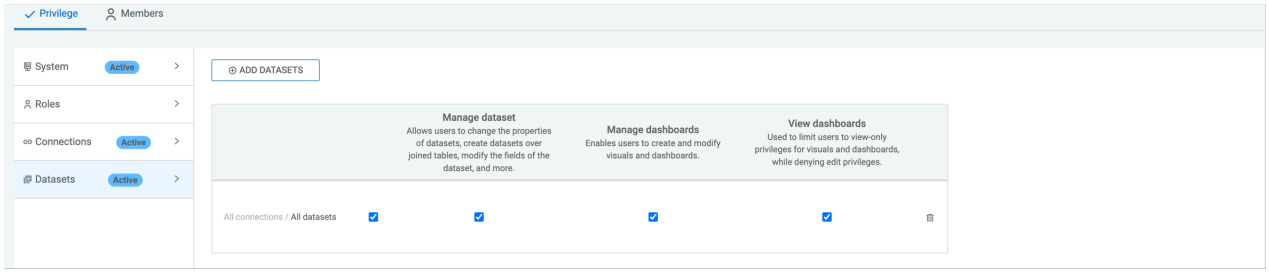

Segments

Cloudera Data Visualization provides URL access to the Cloudera Data Visualization server object, segment.

The segment data field is a complex structure that matches the UI Segment edit screen.

This article includes the following topics:

- Segment Data Type on page 17
- Data Field Detail on page 17

Segment Data Type

The JSON fields for role's data type are defined as follows:

Table 4: JSON Fields for Segment Data Type

Field	Detail Only	Updatable	Description
id	No	No	Segment ID
name	No	Yes	Segment name
dataset_id	No	No	Dataset ID for the segment
created	Yes	No	Shows the creation date for this segment
created_by	Yes	No	Username of the segment creator
updated	Yes	No	Shows the most recent update for this segment
updated_by	Yes	No	Username of the segment updater
data	Yes	Yes	Segment definition data, as described in Data Field Detail on page 17

Data Field Detail

The Cloudera Data Visualization segment data field is a complex structure with the following specification:

Field	Description
entities	List of dimension expressions emitted by the entity-type segments
group	Name of the group to which this segment belongs
filters	List of filter expressions that define this segment
applyToNewVisuals	Specify if new visuals on the dataset should start with filters defined in this segment

Filter associations API

Cloudera Data Visualization provides URL access to the Cloudera Data Visualization server object, filter association.

The filter association data field is a complex structure that matches the Filter Association edit screen.

This article includes the following topics:

- Segment Data Types on page 18
- Data Field Detail on page 18

Segment Data Types

The JSON fields for role's data type are defined as follows:

Table 5: JSON Fields for Filter Association Data Type

Field	Detail Only	Updatable	Description
id	No	No	Filter association ID
name	No	Yes	Filter association name
dataset_id	No	No	Dataset ID for the filter association
created	Yes	No	Shows the creation date for this filter association
created_by	Yes	No	Username of the filter association creator
updated	Yes	No	Shows the most recent update for this filter association
updated_by	Yes	No	Username of the filter association updater
users	Yes	Yes	List of user IDs to which the filter association applies
groups	Yes	Yes	List of group IDs to which this filter association applies
data	Yes	Yes	List of segments that make up this filter association, as described in Data Field Detail on page 18

Data Field Detail

The Cloudera Data Visualization filter association data field is a complex structure with the following specification:

Field	Description
id	ID of segment applied to filter association
group	Name of the group to which the identified segment belongs
negate	Indicates that the filter association defines the rows NOT IN the segment, rather than IN the segment

Workspaces

Cloudera Data Visualization provides URL access to the ArcViz server object, workspaces.

In addition to the standard public workspace that all users share, and the single private workspace that each user has, Cloudera Data Visualization users with Create Workspace privilege can create custom workspaces. These workspaces may be shared by specific users and user groups, so they can develop and view dashboards that are inherently useful to their line of business. Within each of these workspaces, each user has a defined access level: View Only, Edit, or Manage.

This article includes the following topics:

- Workspace data type on page 18
- Access control list in workspaces on page 19
- Creating workspaces on page 19

Workspace data type

The JSON fields for workspace data type are defined as follows:

Table 6: JSON fields for workspace data type

Field	Detail Only	Updatable	Description
id	No	No	Workspace ID
name	No	Yes	Workspace name
desc	No	Yes	Workspace description
editable	Yes	No	Permission to update only a non system-managed workspace (Custom workspace). The system managed workspaces are Public and Private workspaces.
private_user_id	Yes	No	ID of the user for private workspaces
acl	Yes	Yes	Access Control List (ACL) for a workspace

Access control list in workspaces

The workspace ACL is a list of privilege entries. Each entry contains three items. The following acl syntax shows two entries:

```
"acl": [
  [entry_type, access_level, group_name],
  [entry_type, access_level, user_name]]
```

Entry	Encoding
entry_type	1 = User, 2 = Group
access_level	1 = View, 2 = Edit, 3 = Manage
user_name/group_name	User or group name of the entry_type

Creating workspaces

The following code creates a new workspace Test workspace and provides View access to a special group Everyone and Manage access to user admin.

The actual APIKey and the method of retrieving the key depends on the user system.

```
curl -s \
 -X POST \
 -H "Content-Type: application/x-www-form-urlencoded" \
 -H "Authorization: apikey apikey" \
 -d 'data=[{
      "name": "Test workspace",
      "desc": "Workspace created via admin api",
      "acl": [[2, 1, "Everyone"], [1, 3, "admin"]]
   }]' \
 127.0.0.1:7999/arc/adminapi/workspaces
```

When viewing this workspace through the Cloudera Data Visualization UI, it appears on the workspace edit modal window like this:

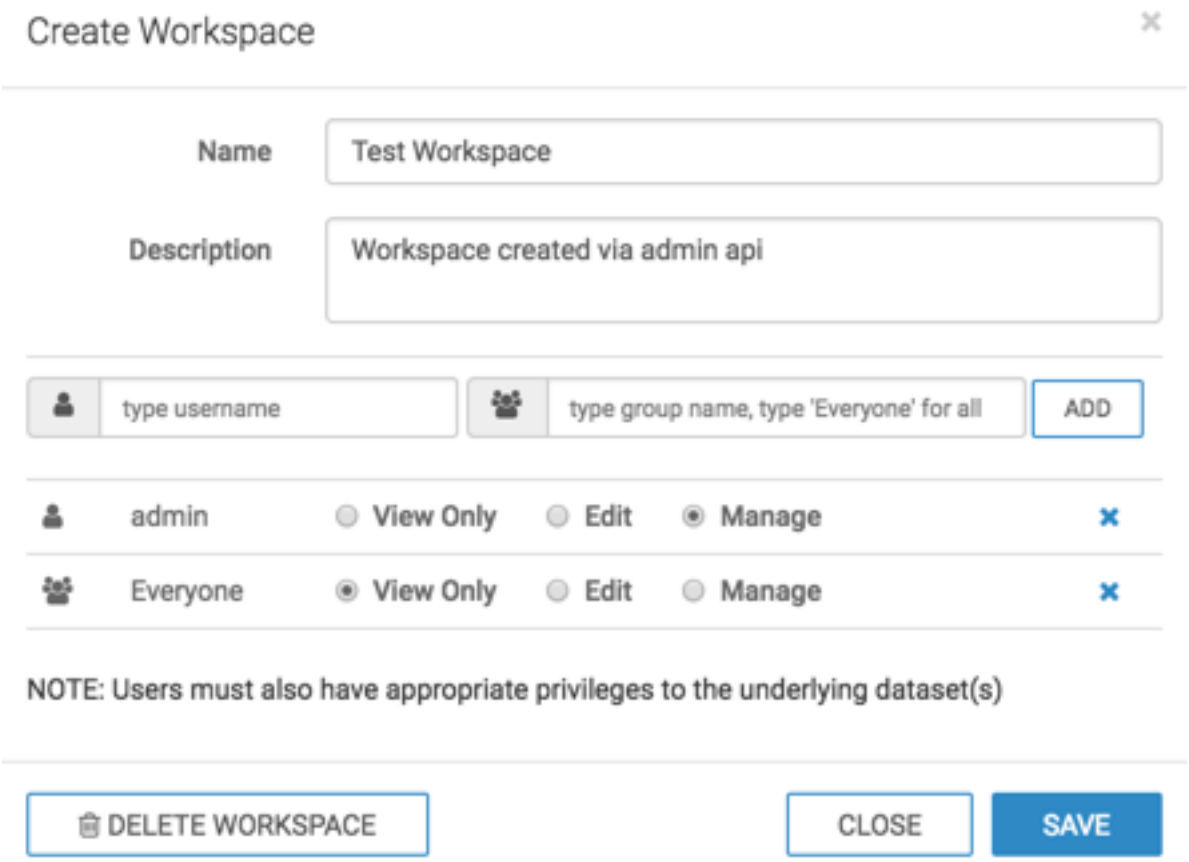
Create Workspace
Name
Test Workspace
Description
Workspace created via admin api
type username
type group name, type 'Everyone' for all
ADD
admin
View Only
Edit
Manage
Everyone
View Only
Edit
Manage
NOTE: Users must also have appropriate privileges to the underlying dataset(s)
DELETE WORKSPACE
CLOSE
SAVE

Cloudera Data Visualization 7.2.9

Certifications

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Certifications 4

Supported metastores 4

Mobile devices 4

Internet browsers 5

Certifications

Cloudera Data Visualization certifies compatibility with the following component categories

Related Information

Supported metastores

Mobile devices

Internet browsers

Supported metastores

Cloudera Data Visualization is certified with the following metadata servers:

- *SQLite*

  This option does not support multi-process Cloudera Data Visualization.

  Arcadia Instant is regularly certified with SQLite MetaStore.
- *MySql*

  Supports multi-process Cloudera Data Visualization in CDH, with 3 processes by default.
- *Oracle*

  Supports multi-process Cloudera Data Visualization in CDH, with 3 processes by default.
- *PostgreSQL*

  Supports multi-process Cloudera Data Visualization in CDH, with 3 processes by default.

Mobile devices

In addition to desktop use, you may access Cloudera Data Visualization (in View mode only) from many mobile devices.

This allows you to provide access to insightful information at a summary level, and quickly informs the user if there is an issue that needs deeper investigation.

The following devices has been certified to display visuals and dashboards:

- Android OS smartphones
- iOS smartphones
- iOS tablets

While you cannot manipulate datasets, or edit visuals and dashboards, you can access all existing visuals for which you have view permissions in View mode, and use existing filters and drill-downs. The URL address syntax is of the following type:

```
[ip_address]/arc/apps/app/[app_id]
```

Tip:

Cloudera recommends that dashboards designed for mobile viewing be simple and intuitive, and minimize the need for scrolling. The visuals should use a small number of dimensions so as not to clutter a small screen, and filters should be minimal.

Internet browsers

We certify Cloudera Data Visualization to run on the latest versions Chrome, and Microsoft Edge browsers.

Cloudera Data Visualization 7.2.9

Custom Style Extensions for Javascript and CSS

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Extension JS API endpoints **4**

Extension JS API arcapi functions **6**

Extension CSS APIs **10**

Extension JS API endpoints

JavaScript module returns an object that may implement one of the endpoints: version() on page 4, supported() on page 4, settings() on page 5, disableDraw() on page 6, beforeDraw() on page 6, afterDraw() on page 6, and afterDataRequest() on page 6.

version()

This must exist, and reports on the version of the API that this object implements. Currently supported version strings is "1". This must be a string, not a number.

supported()

If this member exists, it is expected to be an object that contains a member called visualType. This member must point to a string with the type of visual supported, or an array of strings that contain the types of visuals supported.

For example,

```
supported: {
  visualType: "trellis-bars"
},
```

Table 1: Supported visual types

Visual Type	Object Name
Table	table
Cross Tabulation	crosstab
Bars	trellis-bars
Lines	trellis-lines
Combo	combo
Areas	trellis-areas
Grouped Bars	trellis-groupedbars
KPI	kpi
Packed Bubbles	packed-bubbles
Scatter	scatter
Flow	flow
Funnel	funnel
Pie	pie
Radial	radial
Chord	chord
Correlation Heatmap	correlation
Correlation Flow	correlation-flow
Calendar Heatmap	calendar-heatmap
Map	map
Interactive Map	leaflet
Sparklines	sparklines

Visual Type	Object Name
External Link	link
Histogram	histogram
Extension	extension
Field Statistics	column-stats
Queries	rawtable
Rich Text	html
Network	network
Dendrogram	dendrogram
Treemap	treemap
Bullet	bullet
Gauge	gauge
Word Cloud	wordcloud
Box Plot	boxplot
Timeline	timeline

settings()

If a member with this name exists, it is expected to be a function that returns an array. Each array element is then expected to be an object with the following members:

```
settings: function() {
  return [
    {
      id: "Column Name"
    },
    {
      id: "Max bar width",
      defaultValue: "50"
    },
    {
      id: "Color",
      defaultValue: "steelblue"
    }
  ];
},
```

Table 2: Settings Specifications

Member	Name	Description	Default Value	Mandatory
id	Column Name	The ID of this setting.	None	Yes
displayName	Column Name	Name used to display this setting.	Same as ID	No
type	String	The data type of the variable. Supported types include String, and bool ean.	String	No
defaultValue	""	The default value for this setting.	null	No

disableDraw()

If a member with this name exists, and it is a function that returns a boolean variable set to true, we disable the drawing of the visual.

beforeDraw()

If a member with this name exists, it must be a function that executes after receiving data but before the draw event on the visual.

afterDraw()

If a member with this name exists, it must be a function that executes after the draw event on the underlying visual.

afterDataRequest()

If a member with this name exists, it must be a function that executes after creating the request object, but before the data request call. This function accepts the current request object in the arguments, adds your modifications, and returns an updated valid request object. If the function does not return a valid request object, the request is not sent, and the system returns an error.

Extension JS API arcapi functions

Note: There are additional functions currently available on the arcapi object but not listed here. These functions were developed for internal use only, and we do not warrant them for use by our clients. Cloudera cautions against using them in your customizations because of expected forward compatibility issues.

The extension JS module has access to an arcapi object. Cloudera Data Visualization supports the following functions fully:

Table 3: arcapi functions

Function	Description
arcapi.addScripts()	Adds script elements.
arcapi.addStyles()	Adds style elements.
arcapi.chartId()	Returns the id DOM element.
arcapi.dataResult()	Returns the data result object and gives access to the raw data for plotting the visual.
arcapi.getParameters()	Returns information on all current parameters.
arcapi.getSetting()	Returns the value of the specified setting.
arcapi.sendParameters()	Forwards parameter definitions to other visuals in the app.
arcapi.settings()	Returns information on all current settings.

arcapi.addScripts(filepaths, callback)

Adds script elements for the additional styles specified in the filepaths array. The function callback is invoked after load completes.

Syntax

```
arcapi.addScripts(
  filepaths,
  callback);
```

Parameters

- *filepaths* An array that specifies the additional JS files to load.
- *callback* Call-back function that is invoked after the load is completed.

Examples

In program code, you can use this API in the following manner:

```
arcapi.addScripts(
  ['https://cdnjs.cloudflare.com/ajax/libs/highcharts/7.0.1/highc
harts.js'],
  function() {
    console.log('highcharts.js has been loaded');
    }
  );
```

This code embeds the following <script> as the last element of the <head> tag in the HTML file:

```
<script
  type="text/javascript"
  src="https://cdnjs.cloudflare.com/ajax/libs/highcharts/7.0.1/hi
ghcharts.js"
  data-loaded="true">
</script>
```

arcapi.addStyles(filepaths, callback)

Adds style elements for the additional styles specified in the filepaths array. The function callback is invoked after load completes.

Syntax

```
arcapi.addStyles(
  filepaths,
  callback);
```

Parameters

- *filepaths* An array that specifies the additional CSS files to load.
- *callback* Callback function that is invoked after the load is completed.

Examples

In program code, you can use this API in the following manner:

```
arcapi.addStyles(
  ['https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4
.2.1/css/bootstrap.css'],
  function() {
    console.log('Twitter Bootstrap has been loaded');
    }
  );
```

This code embeds the following <link> as the last element of the <head> tag in the HTML file:

```
<link
  type="text/css"
  rel="stylesheet"
  href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstr
ap/4.2.1/css/bootstrap.css"
  data-loaded="true"
```

```
>
```

arcapi.chartId()

Returns the id attribute of the DOM element used to draw the visual.

Syntax

```
arcapi.chartId();
```

Usage

```
$("#" + arcapi.chartId())
```

Returns

A jquery selector to the DOM element.

arcapi.dataResult()

Returns the dataResult object that gives the extension access to the raw data used to plot the visual. This object supports the arc.data.result interface.

Syntax

```
arcapi.dataResult();
```

This example shows the result of the arcapi.dataResult() call:

```
f (){return"arc.data.result"}
```

arcapi.getParameters()

Returns a hash object that contains key-value mappings for all current parameters. The key is the parameter name.

Syntax

```
arcapi.getParameters();
```

Example

The following is a typical result of the arcapi.getParameters() command:

```
{
  "productCategory": "Analytics",
  "year": "2019",
  "companies.data": "'Apple', 'Google', 'Arcadia'",
  "companies.exclude": "in"
}
```

See Also

arcapi.sendParameters(params_hash) on page 9

arcapi.getSetting(settingName)

Returns the value of the specified setting of the current JS extension.

Syntax

```
arcapi.getSetting(settingName);
```

Parameters

- *settingName* Name of setting.

Example

The following command retrieves the "Color" setting:

```
arcapi.getSetting("Color");
```

Our example's result follows:

```
"red"
```

See Also

arcapi.settings() on page 9

arcapi.sendParameters(params_hash)

Accepts a hash definition, and sends out the keys and values of the hash as parameters to other visuals in the app.

Syntax

```
arcapi.sendParameters(params_hash);
```

Parameters

- *params_hash* Hash definition.

Example

To pass a parameter with name 'company' and value 'Cloudera' to all visuals and filters in the dashboard, issue the following command:

```
arcapi.sendParameters({"company": "'Cloudera'"});
```

See Also

arcapi.getParameters() on page 8

arcapi.settings()

Returns information on all current settings for the custom style: an array of objects, with id of a setting, and value of that setting.

The JS extension's settings() function specifies the order of the array.

Syntax

```
arcapi.settings();
```

Example

A typical example output after running the arcapi.settings() function looks similar to this:

```
 [
    {
      id: "Column Name",
      value: "",
    },
    {
      id: "Max bar width",
      defaultValue: "50",
      value: "50",
```

```
    },
    {
      id: "Color",
      defaultValue: "steelblue",
      value: "red",
    }
  ];
```

Extension CSS APIs

Best practice guidelines for CSS customizations:

- Always use a custom selector at the beginning of every rule.
- Can use any class name.
- Can use native DOM element selections.
- Avoid using !important to override existing rules.

Table 4: CSS Classes

Class	Description
arc-dashboard-title-container	Container for the title of the dashboard.
arc-dashboard-title	The title of the dashboard.
arc-dashboard-subtitle-container	Container for the subtitle of the dashboard.
arc-dashboard-subtitle	The subtitle of the dashboard.
arc-app-container	Container class for dashboards. One exists on the page when rendering the dashboard.
arc-viz-container	Container class for visuals. One exists on the page for each visual.
arc-viz-type-visualtype	Container class for a specific visual type. For example, tables have an arc-viz-type -table class.
arc-viz-title-container	Container class for a div that surrounds the title of the visual. Commonly used to center the title within this container.
arc-viz-subtitle-container	Container class for a div that surrounds the subtitle of the visual. Commonly used to center the subtitle within this container.
arc-viz-title	Class applied to the title of the visual.
arc-viz-subtitle	Class applied to the subtitle of the visual.
arc-viz-axis	Class applied to the axis of the visual.
arc-viz-legend	Class applied to the color or mark legend of the visual.
arc-viz-axis-label	Class applied to the axis label of the visual.

Cloudera Data Visualization 7.2.9

Data API

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Data API **4**

Data API request payload **4**

Data API response payload **6**

Example of data API usage **7**

Data API

Cloudera Data Visualization provides an interface for applications that fetch data through a REST interface.

Typically, users access Cloudera Data Visualization's web server through a web browser, and the interface is entirely visual. This visual interface interacts directly with Cloudera Data Visualization's data layer, which in turn provides seamless access to modeled data, integrated security controls, fast access to data in connected systems, and access to advanced analytics.

Sometimes, operational data applications do not need access to Cloudera Data Visualization's visual layer, but require direct access to the data layer. For these purposes, Cloudera Data Visualization supports a Data API that provides operational data applications with direct access to data.

Related Information

Example of data API usage

Data API request payload

Data API response payload

Data API request payload

Before invoking the Cloudera Data Visualization Data API, you must familiarize yourself with the syntax of the request payload.

Syntax of a Request Payload

The Data API has a consistent pattern for each payload, with the following basic access syntax:

```
{
        "version":version_number,
        "type":"SQL",
        "limit":number_of_rows,
        "dimensions":
          [{
             "type":"SIMPLE/SEGMENT",
             "expr":"[dimension_1] as 'dimension_value_1',
             "expr":"[dimension_2] as 'dimension_value_2',
             "expr":"[dimension_n] as 'dimension_value_n',
             "order": {"asc": True/False, "pri": order_sequence}
          }],
        "aggregates":
          [{
             "expr":["aggregate_1] as 'aggregate_value_1',
             "expr":["aggregate_2] as 'aggregate_value_2',
             "expr":["aggregate_n] as 'aggregate_value_n',
          }],
        "filters":
           [
             "[filter_1] in ('filter_value_1')",
             "[filter_2] in ('filter_value_2')",
             "[filter_n] in ('filter_value_n')"
           ],
        "having":
           [
            "[aggregate_1]) > aggregate_value_1"
           ],
        "dataset_id":dataset_id_number
```

```
}
```

Parameters of a Request Payload

The request payload parameters are defined as follows:

version

Specifies the version of the API. This is used for backward compatibility in case the API changes in the future. This is a mandatory parameter. For example, version 1.

type

Specifies the type of query. This is a mandatory parameter. For example, SQL.

limit

Specifies the number of rows to return. This is a mandatory parameter.

dimensions

List of zero or more dimensions. Specifying no dimensions or aggregates returns all columns from the dataset. Each item in the list specifies a dimension that is requested, and has the following structure of key value pairs:

- *type*

  Either SIMPLE or SEGMENT. Default is SIMPLE. Use field descriptions for expr and order with SIMPLE type.
- *expr*

  Either a dimension or an aggregate with the following format: "expr":"[dimension_1] as 'dimension_value'. This is a mandatory parameter.
- *order*

  Maximum of two key value pairs: ‘asc’, which is set to True or False. This indicates the order and priority, which is set to an integer and specifies the sequence in which the order is to be applied.

aggregates

List of zero or more aggregates. Specifying no aggregates returns all columns from the dataset. Each item in the list specifies an aggregate that is requested, and has the following structure of key value pairs:

- *expr*

  Specifies one aggregate with the following format:

  ```
  [{"expr":["aggregate_1] as 'aggregate_value_1'}]
  ```

dataset_id

Specifies the ID of the dataset object to run the data query. This is a mandatory parameter.

filters

Specifies a comma-separated list of filter expressions that apply to the dataset using the ‘WHERE’ clause. All the filters specified are combined with an ‘AND’ operation. These filters are applied as the ‘WHERE clause in the generated SQL. This is an optional parameter.

having

Specifies expressions to filter the dataset using the HAVING clause. Therefore, this parameter has an aggregation comparison. This is an optional parameter.

Example of a Request Payload

Here is an example of a Data API request payload that interfaces with the Cereals dataset, which ships as a sample with most Cloudera Data Visualization installations.

```
{
        "version":1,
        "type":"SQL",
        "limit":100,
        "dimensions":
          [{
             "type":"SIMPLE",
             "expr":"[manufacturer] as 'manufacturer'",
             "order": {"asc":False, "pri":1
          }],
        "aggregates":
          [{
             "expr":["avg([fat_grams]) as 'avg(fat_grams)'",
             "expr":["sum(1) as 'Record Count'"
          }],
        "filters":
           [
             ["[cold_or_hot] in ('C')"]
           ],
        "having":
           [
            ["[avg([fat_grams]) < 5"]
           ],
        "dataset_id":11
```

Data API response payload

After a successful data request payload execution, the system returns an HTTP response of type text/json.

Syntax of a Response Payload

The Data API has a consistent pattern for an output payload, with the following basic access syntax:

```
{
  "info": [
    "Query:SELECT query
  ],
  "coltypes": [
    "type_of_single_column",
    "BIGINT"
  ],

  "colnames":[
    "column_name_1",
    "column_name_2",
    "column_name_n"
  ],

  "rows": [
    [
     "row_information_1",
     "row_information_2",
     "row_information_n"
    ]
```

```
  ]
}
```

Parameters of a Response Payload

The response payload parameters are defined as follows:

info

String of the raw SQL query that is executed.

coltypes

Array of strings, where each string specifies the type of a single column returned by the data request.

colnames

Array of strings, where each string specifies the name of a single column returned by the data request. This contains the alias for the columns as specified in the initial request.

rows

Array of arrays, where each inner array represents a single row of information returned by the data request.

Example of a Response Payload

Here is an example of a Data API response payload:

```
{
  "info": [
    "Query:SELECT TA_0.`manufacturer` as `manufacturer`, sum(1) as `Record
 Count`\n
       FROM `main`.`cereals`  TA_0\n  WHERE TA_0.`cold_or_hot` in ('C')\n  G
ROUP BY 1\n  LIMIT 100"
  ],
  "coltypes": [
    "STRING",
    "BIGINT"
  ],
  "colnames": [
    "manufacturer",
    "Record Count"
  ],
  "rows": [
    [
      "General Mills",
      22
    ],
    [
      "Kelloggs",
      23
    ]
  ]
}
```

Example of data API usage

To use the Data API interface in Cloudera Data Visualization, you must enable the Data API, obtain an API key, and then invoke the Data API.

The API provides no discovery interfaces to understand the structure and format of the data. The invoker of the interface must be familiar with the target dataset, and what dimensions, aggregates, and filters are appropriate for each use case.

To construct the request payload, see Data API request payload.

The following example of a Data API python code interfaces with the Cereals dataset that ships as a sample within most Cloudera Data Visualization installations.

Data API Programmatic Access

In this example, we are supplying the API Key that authorizes the user to access the dataset.

The host and port variables specify the running instance of the DataViz, and the api_key string has been obtained earlier, as described in Enabling Data API.

Note that you must replace the sample data request, dsreq, with your custom request. The dataset ID in this example is 11; it may be different on your system. Therefore, edit the last line of the dsreq specification to use the dataset ID of your target dataset.

```
import requests
import json
url = 'http://host:port/arc/api/data'

def _fetch_data(dsreq):
  headers = {
   'Authorization': 'apikey api_key'
  }

  params = {
    'version': 1,
    'dsreq': dsreq,
  }

  r = requests.post(url, headers=headers, data=params)
  if r.status_code != 200:
    print('Error', r.status_code, r.content)
    return

  raw = r.content
  d = json.loads(raw)
  print('\nData Request being sent is:\n', \
        json.dumps(json.loads(dsreq), indent=2))
  print('\nData Response returned is:\n', json.dumps(d, indent=2))

def main():

 # CHANGE the following dsreq to a data request of your choice.

  dsreq =
"""{"version":1,"type":"SQL","limit":100,
"dimensions":[{"type":"SIMPLE","expr":"[manufacturer] as 'manufacturer'"}],
"aggregates":[{"expr":"sum([sodium_mg]) as 'sum(sodium_mg)'"},
{"expr":"avg([fat_grams]) as 'avg(fat_grams)'"},
{"expr":"sum(1) as 'Record Count'"}],
"filters":["[cold_or_hot] in ('C')"],
"dataset_id":11}"""

  _fetch_data(dsreq)

if __name__ == '__main__':
  main()
```

Related Information

Data API request payload

Cloudera Data Visualization 7.2.9

Display Format Reference

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Display format reference 4

Standard format specifier 4

Format style table 7

Quantity prefixes 8

Additional format examples 9

Date/time formats 9

Display format reference

Cloudera Data Visualization implements standard number formatting through Python's format specification mini-language. The general format of the specifier is:

```
[[fill]align][sign][symbol][0][width][,][.precision][type]
```

Related Information

Standard format specifier
Format style table
Quantity prefixes
Additional format examples
Date/time formats

Standard format specifier

Format of the specifier follows:

```
[[fill]align][sign][symbol][0][width][,][.precision][type]
```

where:

- fill is a character other than { or }.

  The fill works with one of the alignment options. If the alignment character is invalid, both fill and alignment are ignored.
- align is one of the following:
  - < forces the field to be left-aligned within the available space; this is the default setting.
  - > forces the field to be right-aligned within available space.
  - = forces padding after the sign, if any, but before the digits; used for printing numeric fields in the form +000 000120.
  - ^ forces the field to be centered within the available space.
- sign is one of the following:
  - + means that a sign must be used for both positive and negative numbers.
  - - means that a sign must be used for negative numbers only; this is the default setting.
  - " " (space) means that a leading space must be used on positive numbers, and a minus sign on negative numbers.
- symbol is one of the following:
  - $, to represent currency
  - #, which is valid only for integers of specific output: 0b for binary, 0o for octal, and 0x for hexadecimal options.
- 0 enables zero-padding.
- , (comma) signals to use a comma for a thousands separator.
- width is a decimal integer that defines the minimum field width; if not specified, the content determines the width.

  If preceded by 0, the field will be zero-padded. This produces the same result as alignment of = and fill of 0.

- precision is a decimal that specifies how many digits to display after the decimal point of a floating point value that is formatted with type f, F, or %, or before and after the decimal point of a floating point value that is formatted with type g, r, and p.

  For non-number types, it indicates the maximum fields size, or how many characters to use in field content.

  The precision field does not support integer values.

- type is one of the following:
  - % is for percentage. Multiplies the number by 100, and displays the number in fixed format, f, followed by percent sign.
  - b is for binary format, and outputs numbers in base 2.
  - c is for character; it converts integers to corresponding Unicode characters.
  - d is for decimal integer; it outputs numbers in base 10.

    Use Number.toString() method.
  - e is for exponent notation for floating point and decimal numbers; it prints the number in scientific notation, using e to indicate exponent.

    For example, it would print 345 as 3.45e2.

    Use Number.toExponential() method.
  - E is for exponent notation for floating point and decimal numbers; it prints the number in scientific notation, using E to indicate exponent.

    For example, it would print 345 as 3.45E2.

    Use Number.toExponential() method.
  - f is for fixed floating point. Displays the number as a fixed-point number.

    Use Number.toFixed() method.
  - F is for fixed floating point, same as f; also converts nan to NAN, and inf to INF. Displays the number as a fixed-point number.

    Use Number.toFixed() method.
  - g is general format for floating point and decimal values.

    For a precision p>=1, it rounds the number to p significant digits, and formats the result depending on magnitude, either in fixed-point format, or in scientific notation.

    Both insignificant trailing zeros and decimal point are dropped if unnecessary.

    A precision of 0 is treated the same as precision of 1.

    Regardless of precision, positive infinity is rendered as inf, negative infinity as -inf, positive zero as 0, negative zero as -0, and NaN as nan.

    Use Number.toPrecision() method.
  - G is general format for floating point and decimal values, same as g. Also switches to E notation if the number is too large. Represents infinity and Nan as uppercase: INF, -INF, NAN.

    For a precision p>=1, it rounds the number to p significant digits, and formats the result depending on magnitude, either in fixed-point format, or in scientific notation.

    Both insignificant trailing zeros and decimal point are dropped if unnecessary.

    A precision of 0 is treated the same as precision of 1.

    Regardless of precision, positive infinity is rendered as inf, negative infinity as -inf, positive zero as 0, negative zero as -0, and NaN as nan.

    Use Number.toPrecision() method.
  - n is general format for floating point and decimal number, same as g. However, it uses current locale settings to insert the appropriate number separator characters.

    For example, one thousand one hundred and one-tenth would be rendered as 1,100.1 in United States and as 1.100,1 in France.
  - o is for octal format; it outputs numbers in base 8.
  - p is rounded percentage; like r type, but multiplied by 100 and with suffix %.
  - r is rounded to precision significant digits, padded with zeros whenever necessary, same as for f type. If prec ision is not specified, behaves like g type.

- s is for string format; it is the default type for string, and does not have to be specified.

  If used for floating point or decimal numbers, it is the metric prefix, SI for the International System of Units. It would render micro (0.000001 or 10-6) as 1.00µ, and it would render tera (1,000,000,000,000 or 1012) as 1.00T.
- x is for hexadecimal format; it outputs numbers in base 16, using lower-case letters for digits over 9.

  a for 10, b for 11, c for 12, d for 13, e for 14, and f for 15.
- X is for hexadecimal format; it outputs numbers in base 16, using upper-case letters for digits over 9.

  A for 10, B for 11, C for 12, D for 13, E for 14, and F for 15.
- None is the same as s for strings, and d for integers. It is similar to g for floating point and decimal values, with at least one digit past the decimal point, and a default precision of 12.

Format style table

The following table shows the format style options, and demonstrates how these options would change the appearance of a sample input number, 1235.00.

Table 1: Format style usage and examples

Format Style	Values	Description	Examples	
			Mask	Output
Fill	All characters	Accepts all characters except curly brackets: { and }		
Align	<	Forces left-alignment within the specified space.	*<6	1235**
	>	Forces right-alignment within the specified space (default).	*>6	**1235
	^	Forces central alignment within the specified space.	*^6	*1235*
Sign	Plus +	Uses a sign for both positive and negative numbers.	+	+1235
	Minus -	Uses a sign for only negative numbers (default).	-	-1235
	Space	Uses a leading space on positive numbers, and a minus sign on negative numbers.	' '	1235
Symbol	$	Prefixes the $ (Dollar) currency symbol.	$	$1235
	£	Prefixes the £ (Pounds Sterling) currency symbol.	£	£1235
	¥	Prefixes the ¥ (Japanese Yen) currency symbol.	¥	¥1235
	#	Prefixes the # (Indian Rupee) currency symbol.	#	#1235
	€	Prefixes the € (Euro) currency symbol.	€	€1235
0	0	Enables zero-padding.	06	001235
Width	width	Defines the minimum field width. If not specified, the width is determined by the content.	6	1235
,	,	Enables the use of a comma to separate every third digit.	,	1,235
Precision	.precision	Indicates how many digits must be displayed after the decimal point, for a value formatted with types e, f and %, or before and after the decimal point for values formatted with types g, r, and p.	See f, g, r, p, and % types for examples	
Types	b	Binary: Outputs the number in base 2.	b	10011010 011
	c	Character: Converts the integer to the corresponding Unicode character.	c	ä
	d	Integer: Outputs the number in base 10. Ignores non-integer values.	d	1235

Format Style	Values	Description	Examples	
			Mask	Output
	e	Exponent notation: Displays the number in scientific notation, using the letter e to indicate the exponent.	e	1.235e+3
			.1e	1.2e+3
	f	Fixed point: Displays the number as a fixed-point number.	f	1235
			.1f	1235.0
	g	General: For a given precision p >= 1, this rounds the number to p significant digits and then formats the result in either fixed-point format or in scientific notation, depending on its magnitude.	g	1235
			.2g	1.2e+3
			.5g	1235.0
	n	Formatted Number: This is the same as d, except that it uses the current locale setting to insert the appropriate number separator characters.	n	1,235
			.2n	1.2e+3
			.5n	1,235.0
	o	Octal: Outputs the number in base 8.	o	2323
	r	Rounded: Similar to general format, but does not use scientific notation.	r	1235
			.2r	1200
			.5r	1235.0
	s	String: Rounded and using scientific notation, but with a unit suffixed. See, Name and Symbol in Quantity prefixes. • Renders micro (0.000001 or 10-6) as 1.00µ. • Renders tera (1,000,000,000,000 or 1012) as 1.00T.	s	1.235k
			.2s	1.2k
			.5s	1.2350k
	S	String: Rounded and using regular currency unit suffixes. See, Quantity prefixes. • Renders 1,000,000,000 as 1B, for billions. • Renders 1,000,000,000,000 as 1T, for trillions.	$S	$1235
			$S,	$1,235
	x	Hex: Outputs the number in base 16, using the lower-case letters for the digits greater than 9. a for 10, b for 11, c for 12, d for 13, e for 14, and f for 15.	x	4d3
	X	Hex: Outputs the number in base 16, using the lower- case letters for the digits greater than 9. A for 10, B for 11, C for 12, D for 13, E for 14, and F for 15.	X	4D3
	%	Percentage: Multiplies the number by 100 and displays it in fixed f format, followed by a percent sign.	%	123500%
			.1%	123500.0%

Related Information

Quantity prefixes

Quantity prefixes

Table 2: Quantity prefixes, relative values, symbols, and scientific notation

Word	Decimal	10n	Name (Scientific)	Symbol (Sientific)	Symbol (Currency)
septillion	1 000 000 000 000 000 000 000 000	1024	yotta	Y	Y

Word	Decimal	10n	Name (Scientific)	Symbol (Sientific)	Symbol (Currency)
sextillion	1 000 000 000 000 000 000 000	1021	zetta	Z	Z
quintillion	1 000 000 000 000 000 000	1018	exa	E	E
quadrillion	1 000 000 000 000 000	1015	peta	P	P
trillion	1 000 000 000 000	1012	tera	T	T
billion	1 000 000 000	109	giga	G	B
million	1 000 000	106	mega	M	M
thousand	1 000	103	kilo	k	k
hundred	100	102	hecto	h	-
ten	10	101	deca	da	-
one	1	100			-
tenth	0.1	10-1	deci	d	-
hundredth	0.01	10-2	centi	c	-
thousandth	0.001	10-3	milli	m	-
millionth	0.000 001	10-6	micro	µ	-
billionth	0.000 000 000 001	10-9	nano	n	-
trillionth	0.000 000 000 000 001	10-12	pico	p	-
quadrillionth	0.000 000 000 000 000 001	10-15	femto	f	-
quintillionth	0.000 000 000 000 000 000 001	10-18	atto	a	-
sextillionth	0.000 000 000 000 000 000 000 001	10-21	zepto	z	-
septillionth	0.000 000 000 000 000 000 000 000 001	10-24	yocto	y	-

Additional format examples

Table 3: Number formatting examples

Input	Action	Syntax	Output
1234	Separate thousands by commas.	,	1,234
1234	Show $ value with commas.	$,	$1,234
0.123	Show % value with 2 decimal points of precision.	.2%	12.30%
12345	Display SI syntax.	.2s	12k
123	Show precision to 2 decimal places.	.2f	123.00
123	Include zero-padding for fixed width of 7.	07.2f	0123.00
123	Set sign for positive or negative values with precision to 2 decimal points.	+.2f	+123.00

Date/time formats

Date and time may be represented by a wide variety of tokens. This is a quick reference adapted from Moment.js interface documentation.

Table 4: Tokens for date and time representation with examples

Token	Input	Example	Description
Y	YYYY	2014	4- or 2-digit year
	YY	14	2-digit year
	Y	-25	Year with any number of digits and a sign. Used for years that are not in 4-digit form, and for years Before the Common Era.
Q	Q	1..4	Quarter of year. Sets the month to the first month in quarter.
M	M MM	1..12	Month number
	MMM MMMM	Jan..December	Month name in locale set by moment.locale()
D and d	D DD	1..31	Day of month
	Do	1st..31st	Day of month with ordinal
	DDD DDDD	1..365	Day of year
	ddd dddd	Mon...Sunday	Day name in locale set by moment.locale()
X and x	X	1410715640.579	Unix timestamp
	x	1410715640579	Unix ms timestamp
G and g	GGGG	2014	ISO 4-digit week year
	GG	14	ISO 2-digit week year
	gggg	2014	Locale 4-digit week year
	gg	14	Locale 2-digit week year
W and w	W WW	1..53	ISO week of year
	w ww	1..53	Locale week of year
E and e	E	1..7	ISO day of week
	e	0..6	Locale day of week
H and h	H HH	0..23	24-hour time
	h hh	1..12	12-hour time used with a and A
A and a	a A	am pm	Post- or ante-meridian. Note that one character a or p are also valid.
m	m mm	0..59	Minutes
s and S	s ss	0..59	Seconds
	S SS SSS	0.999	Fractional seconds
Z	Z ZZ	+12:00	Offset from UTC, as +-HH:mm, +-HHmm, or Z.

Cloudera Data Visualization 7.2.9

Email Development Recommendations

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Email design best practices..... 4

Email development best practices.....4

Email design best practices

When creating emails or email campaigns, consider the following recommendations to improve readability, accessibility, and user experience. By following these guidelines, you will improve your email's usability across different devices and platforms, ensuring your message reaches your audience effectively.

- Keep email width between 600-800 pixels to ensure compatibility across most email clients and to display well in preview panes.
- Use a grid-based structure and avoid complex elements that require advanced float or positioning settings. Simple layouts are more likely to render correctly.
- Most email clients block images by default, so minimize their use and always include descriptive alt text.
- Ensure that images that you add by URL can be accessed by all intended email recipients. Keep in mind that emails may be forwarded beyond the initial distribution list.
- Avoid using images as the sole means to convey critical information, as background images may fail to load in some clients.
- Do not rely on images, or compositions of smaller images into a large one, to communicate your message. While large graphic elements may look very attractive in preview, they perform poorly.
- Use basic, cross-platform fonts such as Roboto, Arial, Verdana, Georgia, and Times New Roman. Custom fonts may not render correctly across all clients.
- Avoid elements that require Flash or JavaScript, because they may not be supported by the recipient's mail client. For animations, use GIF files as a compatible alternative.
- Optimize for mobile users:
  - Design emails to be legible on small screens without zooming.
  - Optimize images to prevent slow load times on mobile.
  - Ensure links are easy to tap on touchscreen devices.

Email development best practices

When developing HTML emails, follow these recommendations to improve compatibility, readability, and responsiveness. Following these guidelines helps ensure your emails look consistent, function correctly, and are easily accessible on various platforms

- Structure your email layout using <table> elements, nesting tables as needed for more complex layouts. Tables help ensure consistent rendering across different email clients.
- Set table dimensions and layout using element attributes (for example: cellpadding, valign, width) to enforce a reliable box-model structure.
- Avoid complex code styles:
  - Compound style declarations: font:#000 12px Arial, Helvetica, sans-serif;
  - Shorthand code: #000 instead of #000000
  - CSS layout properties: slot, position, clear, visibility, and so on
  - Complex selectors: descendant, child or sibling selectors, and pseudo-elements
- Inline all CSS styles to enhance compatibility, as many email clients strip out embedded CSS.
- Use only absolute URLs for images and host them on a reliable server to avoid broken images.
- Do not code with JavaScript or Flash. Most email clients do not support these technologies.
- Optimize for mobile users:
  - Make emails responsive using media queries to adjust font sizes and layout for smaller screens.
  - Ensure clickable elements are at least 46x46 pixels for easy tapping on touchscreens.
- Test email design by sending it to various email accounts to check compatibility across different services and devices.

Cloudera Data Visualization 7.2.9

Email Job Parameters

Date published: 2020-10-30
Date modified: 2024-12-18

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Parameters for jobs.....4

Parameters for jobs

Cloudera Data Visualization supports the following job parameters that you can include in emails and email templates.

All jobs

Current Username

<<cloudera_user_id>>

Example:

Current Sheet ID

<<app_id>>

Current Sheet Name

<<app_name>>

Current Sheet Link

<<app_url>>

Relative Link

<<relative_url>>

Job Start Time

<<start_time>>

Page Parameters

<<params>>

Total Job Run Time (as of time emailed)

<<run_time>>

Scheduled and trigger jobs

Next Run Time

<<next_run_time>>

Interval Name

<<run_interval>>

Interval Cron

<<run_interval_cron>>

Triggers jobs

Full Trigger Conditional

<<trigger_fullconditional>>

Data When Triggered

<<trigger_data_value>>

When Trigger Assessed

<<triggered_time>>

Trigger ID

<<trigger_id>>

Trigger Name

<<trigger_name>>

Email Limit

<<minimum_time_interval>>

Triggered jobs based on "Value changed since last assessment"

Change in Data When Triggered

<<trigger_difference>>

Cloudera Data Visualization 7.2.9

Enumerated Values

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enumerated values..... 4

Enumerated values

Cloudera Data Visualization uses standard enumerated types to represent objects, actions, and states, such as ComponentType on page 4, ActionType on page 4, and QueryState on page 4.

ComponentType

- *APP*
- *DATASET*
- *DATACONNECTION*
- *USER*
- *APPGROUP*
- *CUSTOMSTYLE*
- *CUSTOMCOLOR*
- *SNAPSHOT*
- *THUMBNAIL*
- *STATICASSET*
- *ROLE*
- *CUSTOMEMAIL*
- *CUSTOMDATERANGE*
- *WORKSPACE*
- *SEARCH*
- *RAW_QUERY*

ActionType

- *CREATE*
- *READ*
- *UPDATE*
- *DELETE*
- *UNKNOWN*

QueryState

- *STARTED*
- *COMPLETE*
- *QUEUED*
- *CANCELED*
- *ABANDONED*

Cloudera Data Visualization 7.2.9

RBAC Permissions

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Permission levels for role-based access..... 4

Permission levels for role-based access

Cloudera Data Visualization uses Role-Based Access Control (RBAC) permissions to regulate access to different components and functionalities of the system. Administrators configure these permissions when setting up roles, which are then assigned to relevant users or user groups.

System-level permissions

Create workspaces

Allows users to create and share workspaces among users and user groups.

View roles and users

Enables users to view users, user groups, and roles.

Manage roles and users

Grants users the ability to create users, user groups, and roles.

By default, this includes managing Filter Associations on the dataset management interface. Alternatively, you can configure Filter Associations as part of dataset management during individual dataset permission configuration. For more information on how to manage filter association configuration for a dataset, see the *Manage dataset* permission below.

Manage settings

Grants users the ability to manage global site settings.

Manage custom styles

Authorizes users to create new styles for dashboards and visuals.

Important: The Manage custom styles system-level permission allows users to upload external files, including custom styles, JavaScript, HTML, and rich text visuals. This access carries significant security risks, such as introducing malicious code, unverified scripts, or harmful content. To mitigate these risks, restrict this permission to trusted users, such as those with the default System Administrator role or designated admin users. Ensure proper validation and sanitization measures are in place before permitting uploads.

Manage jobs, email templates

Grants users the ability to handle scheduled jobs and create email templates.

View activity logs

Allows users to monitor Cloudera Data Visualization' usage statistics and performance.

Manage data connections

Grants users the ability to create and manage connections to various data sources.

Additional system privilege

Enables users to perform the following actions:

- Set a default homepage for all users. For more information, see *Setting a default homepage for all users*.
- Clone, delete, or edit dashboards in another user's private workspace.
- Perform administrative restart/stop work operations.
- Use Trusted Auth Get Ticket to request a ticket from the Cloudera Data Visualization Server. For more information, see *Embedding apps with trusted authentication*.

Role-level permissions

Grant manage dataset

Enables users to assign Manage dataset privileges to specific roles, provided the user has Manage dataset permission for that dataset.

Grant manage dashboards

Enables users to assign Manage dashboard privileges to specific roles, provided the user has Manage dataset permission for that dataset.

Grant view dashboards

Enables users to assign View dashboard privileges to specific roles, provided the user has Manage dataset permission for that dataset.

Connection-level permissions

Manage AVs/Extracts

Enables users to create and manage analytical views.

Import data

Allows users to import supplemental data into an existing connection.

Create datasets, explore tables

Allows users to create new datasets from existing tables, view sample data, and explore statistical reports on the data tables.

Dataset-level permissions

Manage dataset

Allows users to modify dataset properties, create datasets from joined tables, modify the fields of the dataset, and more.

To enable Filter Association (FA) configuration based on Manage dataset permission, add the the following line under Site Settings Advanced Settings Advanced Site Settings :

```
MANAGE_DS_FA = True
```

If you get the Manage filter associations permission using the MANAGE_DS_FA flag, you at least have to have View roles and users permission to be able to work with filter associations.

Manage dashboards

Enables users to create and modify visuals and dashboards.

View dashboards

Limits users to view-only privileges for visuals and dashboards, without edit privileges.

Cloudera Data Visualization 7.2.9

Release Notes

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What's new in Cloudera Data Visualization?.....5

What's new in Cloudera Data Visualization 7.2.9.....5
What's new in Cloudera Data Visualization 7.2.8.....5
What's new in Cloudera Data Visualization 7.2.7.....7
Older releases.....8
What's new in Cloudera Data Visualization 7.2.6.....8
What's new in Cloudera Data Visualization 7.2.5.....9
What's new in Cloudera Data Visualization 7.2.4.....10
What's new in Data Visualization 7.2.3.....10
What's new in Data Visualization 7.2.2.....11
What's new in Data Visualization 7.2.1.....12
What's new in Data Visualization 7.2.0.....14
What's new in Data Visualization 7.1.9.....16
What's new in Data Visualization 7.1.8.....16
What's new in Data Visualization 7.1.7.....17
What's new in Data Visualization 7.1.6.....19
What's new in Data Visualization 7.1.5.....19
What's new in Data Visualization 7.1.4.....20
What's new in Data Visualization 7.1.3.....21
What's new in Data Visualization 7.1.2.....22
What's new in Data Visualization 7.1.1.....23
What's new in Data Visualization 7.1.0.....23
What's new in Data Visualization 7.0.5.....25
What's new in Data Visualization 7.0.4.....26
What's new in Data Visualization 7.0.3.....28
What's new in Data Visualization 7.0.2.....30
What's new in Data Visualization 7.0.1.....31
What's new in Data Visualization 7.0.0.....32
December 17, 2021 - CVE-2021-44228.....33
December 10, 2021.....33
November 8, 2021.....34
September 30, 2021.....35
August 24, 2021.....37
July 30, 2021.....38
June 4, 2021.....39
April 29, 2021.....40
April 7, 2021.....41
February 26, 2021.....42
February 2, 2021.....43
December 15, 2020.....44
December 3, 2020.....45
October 30, 2020.....45

Known issues and limitations in Cloudera Data Visualization.....46

Known issues and limitations in Cloudera Data Visualization 7.2.9.....46
Known issues and limitations in Cloudera Data Visualization 7.2.8.....48
Known issues and limitations in Cloudera Data Visualization 7.2.7.....51
Older releases.....54

Known issues and limitations in Cloudera Data Visualization 7.2.6....54
Known issues and limitations in Cloudera Data Visualization 7.2.5....58
Known issues and limitations in Cloudera Data Visualization 7.2.4....61
Known issues and limitations in Data Visualization 7.2.3....64
Known issues and limitations in Data Visualization 7.2.2....67
Known issues and limitations in Data Visualization 7.2.1....70
Known issues and limitations in Data Visualization 7.2.0....74
Known issues and limitations in Data Visualization 7.1.9....77
Known issues and limitations in Data Visualization 7.1.8....81
Known issues and limitations in Data Visualization 7.1.7....85
Known issues and limitations in Data Visualization 7.1.6....88
Known issues and limitations in Data Visualization 7.1.5....92
Known issues and limitations in Data Visualization 7.1.4....96
Known issues and limitations in Data Visualization 7.1.3....100
Known issues and limitations in Data Visualization 7.1.2....104
Known issues and limitations in Data Visualization 7.1.1....107
Known issues and limitations in Data Visualization 7.1.0....111
Known issues and limitations in Data Visualization 7.0.5....114
Known issues and limitations in Data Visualization 7.0.4....118
Known issues and limitations in Data Visualization 7.0.3....122
Known issues and limitations in Data Visualization 7.0.2....127
Known issues and limitations in Data Visualization 7.0.1....132
Known issues and limitations in Data Visualization 7.0.0....135

What's new in Cloudera Data Visualization?

This section provides details on the new features and major updates introduced in the recent versions of Cloudera Data Visualization. Explore how these changes can improve functionality and usability in your workflows.

What's new in Cloudera Data Visualization 7.2.9

Cloudera Data Visualization 7.2.9 was released on December 18, 2024 introducing the following changes.

New features and improvements

VIZ-2946, VIZ-3017, VIZ-3038

AI visual (Technical Preview) improvements including support for authenticated Cloudera AI models, new functionality to test models, and enhanced logging and visibility of the AI terms and conditions when enabling the AI visual.

VIZ-2917

The sample application is now shown on the HOME and VISUALS pages as part of 'Samples'.

Fixed issues

VIZ-2650, VIZ-2717, VIZ-2805, VIZ-3019, VIZ-3035, VIZ-3020, VIZ-3024, VIZ-3025, VIZ-3026, VIZ-3027, VIZ-3030

Various security and performance enhancements.

VIZ-2754

The 'Everyone' option is now available under Groups in the Add Workspace and Edit Workspace modals.

VIZ-2972

Improved CSV/Excel download at scale.

VIZ-2944

Improved log history for data extract jobs.

VIZ-3006

Constraints now prevent duplicate private workspaces.

VIZ-3042

Improved partitioned data extract creation to better handle when partitions are dropped.

What's new in Cloudera Data Visualization 7.2.8

Cloudera Data Visualization 7.2.8 was released on October 30, 2024 introducing the following changes.

New features and improvements

VIZ-825

You can analyze and summarize data characteristics in a dataset with the new Data Profiling feature. This tool helps you to understand data quality, structure, and identify patterns and trends for more informed decision making. For more information, see Working with data profiling.

VIZ-2585, VIZ-2842

UI and performance improvements for Activity Log and Historical Log tables.

VIZ-2782, VIZ-2915

Clone and Delete actions are now available from the Dataset Detail page for users with appropriate permissions.

VIZ-2801

Discontinued support for older versions of several external database metastores to ensure compatibility with the latest security, performance, and feature updates introduced in Django 4.2.

Affected database versions:

- PostgreSQL 11 and lower
- MariaDB 10.3 and lower
- MySQL 5.7 and lower
- Oracle Database Server versions lower than 19c
- SQLite versions lower than 3.21.0

Users running any of the above database versions will need to upgrade their external database metastores to maintain full functionality with Cloudera Data Visualization.

Supported database versions:

- PostgreSQL 12+
- MariaDB 10.4+
- MySQL 8+
- Oracle Database Server 19c+
- SQLite 3.21.0+

To ensure compatibility and full feature support, upgrade to one of the supported database versions. Failure to upgrade to a supported version may lead to database connection issues, data integrity risks, or unexpected behavior in Cloudera Data Visualization features.

VIZ-2804

Added an option to create a new schedule interval in the Edit Job modal.

VIZ-2808, VIZ-2941

Introduced two new visual settings for histogram visuals. You can now set minimum and maximum values under Settings Basic , and separate columns using the checkbox available under Settings Marks .

Fixed issues

VIZ-987

Fixed an issue where generation of new thumbnails while importing visual artifacts failed with an error message.

VIZ-2324

Removed the Explore visuals icon from the Dashboard Builder when the setting is disabled.

VIZ-2720, VIZ-2784, VIZ-2916

Additional performance and security improvements.

VIZ-2819, VIZ-2897, VIZ-2898, VIZ-2899, VIZ-2900, VIZ-2901, VIZ-2903, VIZ-2904, VIZ-2905

Fixed critical and high-severity package vulnerability issues identified in Cloudera Data Visualization 7.2.7.

VIZ-2869, VIZ-2923

Fixed an issue where dashboards with large numbers of visuals and filters would freeze when switching sheets.

VIZ-2870

Filters with app-level filter visibility now work as expected when cloning and importing visual artifacts.

VIZ-2871

Partitioned data extract refresh now works as expected with Impala tables running in Cloudera Private Cloud Base 7.1.9.

VIZ-2880, VIZ-2945

Fixed an issue where CSV/Excel content downloads occasionally saved the HTML version of the page content instead of the requested data.

VIZ-2889

You can no longer reuse an existing data connection name when creating a new connection. When cloning a data connection, the system will automatically append a version number (for example: "_1") to the name of the new connection. This update resolves an issue where duplicate data connection names previously caused conflicts during visual artifact imports.

VIZ-2919

Searching within a filter now properly limits results based on columns specified as filter fields in the Filter Settings.

VIZ-2925, VIZ-2962

Visual artifact import and export for artifacts containing segments now work as expected.

What's new in Cloudera Data Visualization 7.2.7

Cloudera Data Visualization 7.2.7 was released on September 30, 2024 introducing the following changes.

New features and improvements

VIZ-1090, VIZ-2758, VIZ-2803, VIZ-2812

The Schedule Jobs and Runs tabs have been updated with improved viewing and filtering capabilities for job logs.

VIZ-2813

You can now set values of less than a full day for the following Site Settings:

- Site Configuration > Notification maximum retention and Maximum days for expiring usage data
- Debugging and Logs > Check Interval and Maximum Retention Days
- Version Control > Unnamed Dataset Version Deletion Interval and Unnamed Dataset Version Maximum Retention Days

VIZ-2821

Search functionality has been added to the Manage Roles page to locate specific roles.

VIZ-2846

Support has bee added for header and content alignment, along with column auto-sizing in the cross tab visual.

VIZ-2865, VIZ-2866

Improvements have been implemented for Daily Usage and Activity Log field summaries.

Fixed issues

VIZ-1074

Creating a new schedule interval with the same name as a previously deleted schedule interval now works as expected.

VIZ-2006

Field conversion now functions properly for MySQL data connections.

VIZ-2499

Datasets from SQL queries can now be created even if a query ends with a semicolon.

VIZ-2737

Linked visuals no longer appear twice in the Delete Dataset modal.

VIZ-2759

Paragraph styles in the rich text visual are now responsive.

VIZ-2787, VIZ-2841, VIZ-2844, VIZ-2858

Fixed critical and high-severity package vulnerability issues identified in Cloudera Data Visualization 7.2.6.

VIZ-2829

For Cloudera Data Visualization applications running in the Cloudera Machine Learning Private Cloud Data Service, application initialization now continues even if the data connection syncing fails.

VIZ-2837, VIZ-2857

Other security and performance improvements.

VIZ-2840

Extract references are no longer shown on hover when viewing a dashboard's information card. Extract details can be found on the Data tab or when viewing the dataset directly.

VIZ-2868

Cloudera Data Visualization instances backed by PostgreSQL metadata can now be seamlessly upgraded from 7.2.5 to later versions.

Older releases

This section provides an overview of the new features, enhancements, and changes introduced in previous versions of Cloudera Data Visualization.

What's new in Cloudera Data Visualization 7.2.6

Cloudera Data Visualization 7.2.6 was released on September 2, 2024 introducing the following changes.

New features and improvements

VIZ-2644 – Support for multiple completion and embedding models

Added support for saving multiple completion and embedding models within AI Settings [Technical Preview]. For more information on saving AI Engine profiles, see Managing AI settings.

VIZ-2490, VIZ-2789 – Dataset versioning

Added support for dataset versioning [Technical Preview]. Using dataset versioning allows you to manage different versions of your datasets, which is especially useful when handling frequent updates or iterative changes. Versioning helps you to track changes and roll back to an earlier version if an error is introduced or an earlier version is preferred, ensuring data consistency and reliability. It also adds a layer of security and control, ensuring that data integrity is maintained even

as datasets evolve. See Managing version control site settings and Tracking dataset versions for more information.

VIZ-2746, VIZ-2748 – Sentry group visibility in role management

Sentry Groups now appear when editing membership of a role from the Role Details page. Membership for roles imported from Sentry cannot be changed on this page and must be managed through the Sentry system.

VIZ-2662, VIZ-2663 – Updated icons

The icons for Exploring Visual Options and the AI Visual in the Visual Type Catalog have been updated.

VIZ-2739 – Improved column alignment in CSV files

Column alignment in CSV files attached to emails now mirrors that of downloaded CSV files.

Fixed issues

VIZ-1632, VIZ-2035, VIZ-2648, VIZ-2649, VIZ-2684, VIZ-2732, VIZ-2767, VIZ-2793, VIZ-2795, VIZ-2796

Implemented various security, performance, and user experience improvements across Cloudera Data Visualization.

VIZ-2151, VIZ-2152, VIZ-2731, VIZ-2749

Implemented fixes and improvements for Solr data connections [Technical Preview].

VIZ-2172

Kerberos login error messages are now properly surfaced when they occur.

VIZ-2429

Fixed an issue where the date range selector could become unresponsive when entering a custom range selection.

VIZ-2643, VIZ-2724, VIZ-2735, VIZ-2750, VIZ-2752, VIZ-2777

Implemented fixes and improvements for the recently redesigned Manage Users and Groups and Job Details interfaces.

VIZ-2738, VIZ-2765, VIZ-2771, VIZ-2772, VIZ-2773, VIZ-2791

Fixed critical and high-severity package vulnerability issues identified in Cloudera Data Visualization 7.2.5.

What's new in Cloudera Data Visualization 7.2.5

Cloudera Data Visualization 7.2.5 was released on June 28, 2024 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.2.5

VIZ-2678, VIZ-2685, VIZ-2688, VIZ-2705

UI and performance improvements for the Manage Roles and Role Detail interfaces.

Fixed issues in Cloudera Data Visualization 7.2.5

VIZ-2267

Fixed an issue where certain users without the Manage Custom Styles privilege were incorrectly able to clone and modify Rich Text Visuals.

VIZ-2557, VIZ-2638, VIZ-2639, VIZ-2641, VIZ-2672

Bug fixes and improvements for editing email jobs and viewing details of previous job runs.

VIZ-2602 & VIZ-2712

Fixed critical and high package vulnerability issues identified in Cloudera Data Visualization 7.2.4.

VIZ-2671, VIZ-2683, VIZ-2684, VIZ-2689, VIZ-2693, VIZ-2723

Bug fixes and performance improvements for loading, editing and deleting datasets using the Data page.

VIZ-2707

Fixed an issue where editing dataset fields for a newly created Impala column would cause a syntax error unless the dataset was saved after refreshing the table information.

VIZ-2721

Fixed an issue where CSV files sent as email attachments with lines exceeding 1000 characters could be truncated by email servers.

What's new in Cloudera Data Visualization 7.2.4

Cloudera Data Visualization 7.2.4 was released on May 30, 2024 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.2.4

VIZ-1272, VIZ-2673

The SQL editor window is now resizeable when editing a query on the Dataset page or SQL tab.

VIZ-2448

UI improvements for the Dataset deletion modal.

Fixed issues in Cloudera Data Visualization 7.2.4

VIZ-462, VIZ-2668

Bug fixes and UI improvements for the Role Details page.

VIZ-1522, VIZ-2636, VIZ-2637, VIZ-2642, VIZ-2658

Bug fixes and improvements for the Job Details page introduced in Cloudera Data Visualization. 7.2.3.

VIZ-1855

Removed SparkSQL as a data extract source or target connection.

VIZ-2193, VIZ-2621, VIZ-2655, VIZ-2680

Bug fixes and improvements for the Job Status page.

VIZ-2352

Email jobs without associated attachments can now be scheduled and edited as expected.

VIZ-2450

Email configuration with SMTP servers without client authentication or with an unspecified password field now works as expected.

VIZ-2607

Fixed a bug where an error occurred while hovering on legend items in Area visuals.

VIZ-2628, VIZ-2660

Fixed a bug where dashboards that contained references to removed visuals could prevent deletion of datasets.

VIZ-2661

Sorting a table visual with multiple columns now works as expected.

VIZ-2666, VIZ-2667

Fixed critical and high package vulnerability issues identified in Cloudera Data Visualization 7.2.3.

What's new in Data Visualization 7.2.3

Cloudera Data Visualization (CDV) 7.2.3 was released on April 29, 2024 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.2.3

VIZ-1975, VIZ-2520, VIZ-2540, VIZ-2608, VIZ-2612, VIZ-2625,

The Job status and management interface has been enhanced with performance improvements and a new More Info option to view job details alongside logs of previous runs.

VIZ-2392

The Manage Roles Role Details interface has been redesigned for improved user experience and performance. Detailed descriptions of each privilege are now included.

VIZ-2389

Joins can now be aliased from within the Data Model interface.

Fixed issues in Cloudera Data Visualization 7.2.3

VIZ-1776 & VIZ-2622

Fixed a bug where SQL Query visuals mistakenly inherited limits from table visuals.

VIZ-1825

Resolved an issue where Search visuals could become stuck in an infinite loop in Edit mode.

VIZ-2527

The following changes were implemented for audit logging:

- Extract-related SQL queries are logged with type Data Connection instead of Dashboard.
- Username and IP address are logged when there is a direct initiator (such as most editing/viewing actions, ad-hoc email jobs, and data extract runs).
- IP addresses are not logged for scheduled events such as emails and data extracts. The IP address is localhost (127.0.0.1) for query executions related to snapshot creation.

VIZ-2562

Fixed a bug where table join expression validation failed in scenarios where at least three tables were joined and one of the tables contained a derived field that referenced a field of another table.

VIZ-2563

Time axis autobucketing now works as expected when building Bar visuals.

VIZ-2564

Histogram visuals now use shelf expressions correctly when composing data queries.

VIZ-2580

Connection explorer is now shown as expected when viewing a Solr data connection from the SQL tab.

VIZ-2586

Datasets containing only a single table are now correctly typed, even if the dataset previously contained multiple tables.

VIZ-2587

Join pairs are now updated when updating an existing join using the Data Model page.

VIZ-2596

Multi-sheet dashboards are now correctly rendered in the Dashboard Builder.

What's new in Data Visualization 7.2.2

Cloudera Data Visualization (CDV) 7.2.2 was released on March 28, 2024 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.2.2

VIZ-2258

Active Queries Detail now shows username for queued HTTP requests.

VIZ-2285

Implemented UI improvements for Manage URL Alias page and when generating a new Dashboard URL.

VIZ-2495, VIZ-2522, VIZ-2558, VIZ-2560

Implemented application-wide performance improvements.

Fixed issues in Cloudera Data Visualization 7.2.2

VIZ-1195

Fixed a bug where a ghost join to a null table was created if the Edit Join process was abandoned.

VIZ-1212, VIZ-1707, VIZ-2462, VIZ-2548

Implemented UI improvements and bug fixes on Activity Log page and when showing Query details in-dashboard.

VIZ-1816

Temporary folders created during thumbnail creation are now removed after Chromium is shut down.

VIZ-2479

Fixed a bug which prevented deletion of sheets other than the current active sheet.

VIZ-2513, VIZ-2566, VIZ-2567, VIZ-2569

Fixed critical and high package vulnerability issues identified in CDV 7.2.1.

VIZ-2550

Viewing data load details from the “This dashboard is still loading data” notification now works as expected.

VIZ-2571

Fixed a bug where an error could occur when cloning a dataset.

VIZ-2577

Fixed a bug where some users experienced autofill issues on the Site Settings page.

VIZ-2578

Data is now properly plotted in X-axis of bar and line visuals for date range field types.

What's new in Data Visualization 7.2.1

Cloudera Data Visualization (CDV) 7.2.1 was released on Febrruary 29, 2024 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.2.1

VIZ-2344, VIZ-2502

Popup and inline alerts for warnings and errors have been redesigned application-wide for improved readability, clarity, and accessibility.

VIZ-2482, VIZ-2483, VIZ-2486, VIZ-2491, VIZ-2516

Implemented application-wide performance improvements.

VIZ-2488, VIZ-2515

Functionality, UI, and performance improvements have been made to the Artificial Intelligence (AI) Assistant feature. [Technical Preview]

VIZ-2496, VIZ-2538

Improvements and fixes to the redesigned Visuals page, including the following:

- Shared and Private workspaces are now sorted and these lists can now be collapsed for easier navigation.
- The positions of the Public and Samples workspaces in the workspace list now reflect user permissions.
- Selected reports are now remembered when you switch between workspaces.

VIZ-2506

Dataset fields can now be marked as 'Sensitive' using Dataset Field Editor . Sensitive fields cannot be moved to visual shelves marked as sensitive, such as the 'Embedding Context' shelf of the Artificial Intelligence (AI) Assistant visual [Technical Preview].

Fixed issues in Cloudera Data Visualization 7.2.1

VIZ-1868, VIZ-2498, VIZ-2501

Custom fixed dimensional color ranges have been disabled for combo visuals. Custom color for measure value ranges is only enabled for visuals that support it: scatter, table, map, and heatmap. To use this feature on these visuals, ensure that the Color shelf contains a field with numerical values.

VIZ-2267

Users without Manage Custom Styles privilege are no longer able to clone and modify Rich Text visuals.

VIZ-2328, VIZ-2489

Fixed a bug where enabling visual transitions for Bar visuals with fields on the color shelf resulted in an error.

VIZ-2408, VIZ-2454, VIZ-2348, VIZ-2512, VIZ-2525, VIZ-2526

Fixed critical and high package vulnerability issues identified in CDV 7.2.0.

VIZ-2475

When downloading the result of a query, the desired row limit for the CSV can now be set using the UI as expected.

VIZ-2484

Snapshot browser session lifetime values have been consolidated to align with manually configured snapshot timeouts. Additionally, user authentication errors encountered during snapshot creation are now properly surfaced.

VIZ-2492

Excel email attachment creation for crosstab visuals without data results now works as expected.

VIZ-2493

Fixed an issue where exporting a dashboard from the Visuals page resulted in corrupted JSON.

VIZ-2497

Duplicate visual titles on the same dashboard are now correctly parsed when generating Excel downloads for email jobs.

VIZ-2499

Dataset creation from SQL query now works as expected when the query closes with a semicolon.

VIZ-2519

The ‘Show Data’ preview in Dataset Data Modal is now limited to 100 rows.

VIZ-2535

Additional controls have been added to Additional System Permissions privilege, ensuring that correct report totals by workspace are presented to those users.

What's new in Data Visualization 7.2.0

Cloudera Data Visualization (CDV) 7.2.0 was released on January 29, 2024 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.2.0

VIZ-2310

New visual type: Artificial Intelligence (AI) Assistant [Technical Preview]

The AI Assistant introduces a natural language interface for interacting with dashboards and datasets. The new visual type is available for SQLite and Solr data connections that contain vectorized column information that you would like to use for creating a Retrieval-Augmented Generation (RAG) chatbot. Data upload is now also available for Solr data connections, with support for vectorized fields in Solr 9.0+. For more information, see the AI Assistant documentation.

VIZ-838

A new system-level privilege option called Additional system privileges has been added. Granting this privilege to a role enables users to perform the following actions:

- Set the homepage for all users
- Clone, delete, or edit dashboards in another user's private workspace
- Perform administrative restart/stop work operations
- Use Trusted Auth Get Ticket to request a ticket from the CDV Server.

For more information, see Additional system privileges in the system-level permissions documentation.

VIZ-843

Administrators or users with Additional system privilege can now easily move dashboards and apps to any workspace, including private ones.

VIZ-945, VIZ-2048, VIZ-2293, VIZ-2308, VIZ-2386, VIZ-2426, VIZ-2463, VIZ-2478

Application-wide performance improvements, including enhancements at application startup, during dashboard editing, and to loading the Data and Job Status interfaces.

VIZ-1424, VIZ-2432

The Visuals page has been redesigned for quicker loading and enhanced usability.

VIZ-1596, VIZ-2480

The Reorder sheet IDs to match positions option has been added to dashboards with multiple sheets. You can use this setting to update sheet numbers if a sheet is moved or deleted. For more information, see Reordering sheet IDs.

VIZ-1903, VIZ-1937

Users can more granularly add aggregation to fields when creating data extracts. It is also possible to specify the same field with different aggregations multiple times in the same data extract.

VIZ-1993, VIZ-2461

Improved logging for errors that occur during data import.

VIZ-2449

Caddy access log format customization is now possible using the ACCESS_LOG_FORMAT environment variable.

VIZ-2468

Health checking HTTP requests no longer result in a new row being added to the CDV metadata table for Django sessions if the HTTP user agent is identified as one used by a health check procedure.

Fixed issues in Cloudera Data Visualization 7.2.0

VIZ-1194

Fixed a bug where editing or canceling a second join within an existing dataset's data model could display details of a previously created join.

VIZ-1503

Canceling a running query from the SQL tab is now explicitly possible using a new 'Cancel' option.

VIZ-1662

When downloading CSV or Excel files, the queries' results are now saved in the cache.

VIZ-1677

"Query Timeout" and 'Queue Depth' parameters in the Advanced tab on applicable data connections have now been clarified.

VIZ-1853

Fixed a bug where filter dropdowns had unexpected overlap if stacked in a dashboard's filter bar.

VIZ-2153

The 'Add Data' icon is no longer displayed on the Recent Connections homepage tiles for read-only data connections.

VIZ-2277, VIZ-2348, VIZ-2407

Fixed critical and high package vulnerability issues identified in CDV 7.1.9.

VIZ-2325

Google Client Key can now be configured in Site Settings as required for Interactive Map visuals.

VIZ-2384

Fixed a bug where unexpected HTML elements could be included when downloading Query History from a dashboard's profile data modal.

VIZ-2385, VIZ-2383

SQL Query popups are now shown as expected for all queries within the Show Usage Info and Activity Log tables. Additionally, fixed a bug where the full query text popup could extend outside of the browser viewport for lengthy queries.

VIZ-2423

Connection syncing options for applications running in CML are now only shown when the underlying CML version permits connection syncing.

VIZ-2427

Setting a filter pattern using textbox input now works as expected.

VIZ-2442

Saved and Recent Queries launched from the Home interface are now displayed in the query editor as expected.

VIZ-2457

Fixed a bug where expressions with certain aggregations would return an error when placed on measure shelves during visual creation.

What's new in Data Visualization 7.1.9

Cloudera Data Visualization (CDV) 7.1.9 was released on December 22, 2023 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.1.9

VIZ-1078

Upgraded libraries now permit adding custom Mapbox styles to interactive map visuals.

Fixed issues in Cloudera Data Visualization 7.1.9

VIZ-2388

Improved error handling when an empty User or Admin Group is associated with a Data Visualization instance.

VIZ-2396

Data Visualization sessions are now cleared when User or Admin Groups. associated with the instance are updated from the Cloudera Data Warehouse service.

VIZ-2400

Saving a dashboard no longer generates a snapshotting error related to Pillow libraries.

VIZ-2402

Expected missing environmental variables are now reported with INFO level logging rather than ERROR level.

VIZ-2404

Resolved bugs in the Usage Analytics (Technical Preview) feature.

VIZ-2405

Fixed critical and high package vulnerability issues identified in CDV 7.1.5.

VIZ-2417

Query tracking IDs are now unique across connections to avoid collision on dashboards containing visuals from multiple data sources.

VIZ-2419

Fixed a bug where deleting a visual from an unsaved dashboard could trigger an error message about a non-existent report ID.

VIZ-2425

Resolved an issue where dataset creation on a data connection with connection-level caching could trigger an error message about a non-existent query.

VIZ-2431

Resolved bugs related to moving artifacts between workspaces.

What's new in Data Visualization 7.1.8

Cloudera Data Visualization (CDV) 7.1.8 was released on December 4, 2023 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.1.8

VIZ-1024, VIZ-2290, VIZ-2291, VIZ-2353

Application wide performance improvements, including enhancements for the Visuals and Data tabs.

VIZ-1191

Formatted SQL is now available when inspecting a query from historical and activity logs.

Fixed issues in Cloudera Data Visualization 7.1.8

VIZ-1607

Fixed a bug where the Select All option on a dashboard filter was not updated with a new selection if the parent filter field is changed.

VIZ-1806

Adding a custom color for a measure value range after changing color opacity value now correctly takes into account the selected opacity.

VIZ-2256

Fixed a bug where an error message could appear after selecting the Show Delta checkbox on the Logs and Usage WebServer Stats page.

VIZ-2268, VIZ-2272

Fixed a bug where the Jobs Overview page did not refresh after updating or editing an existing email job.

VIZ-2305

Resolved issue with browser autofill collision when entering data connection username and password.

What's new in Data Visualization 7.1.7

Cloudera Data Visualization (CDV) 7.1.7 was released on October 30, 2023 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.1.7

VIZ-2029, VIZ-2071

New functionalities have been added to the redesigned Notification Center.

- Notifications are automatically marked as read.
- Notifications can be pinned to prevent being cleared.
- Duplicate notifications are stacked to reduce clutter.
- Administrators can mute notifications across the entire application using a new site setting.

For more information, see Notification center.

VIZ-1673, VIZ-1729

Creating or editing a dataset from SQL is now easier with autocompletion and syntax highlighting available in the dataset modal.

VIZ-2238

The default file size limit for static asset upload has been increased to 100 MB. You can configure this limit using the DATA_UPLOAD_MAX_MEMORY_SIZE flag in Site Settings Advanced Settings . Updated limits are now correctly displayed in the UI.

VIZ-2067

The default row upload limit is now shown when importing data using CSV. For performance reasons, it is recommended to upload files of 50 MB or less.

VIZ-1858

Job logs can now be filtered by date range from the Job Status page.

VIZ-2007

When inputting custom colors for measure value ranges, it is now possible to include decimals for increased precision.

VIZ-2250

The ability to generate a custom or shortened URL alias when sharing a dashboard and to manage existing URL aliases is now available to all users by default. The Enable URL Shortener site setting used during the technical preview of the feature has been removed.

VIZ-2292

The number of HTTP worker threads available to the CDV server is now configurable using the HTTP_THREADS environment variable. A higher number of HTTP worker threads improves performance when serving a high number of concurrent requests.

VIZ-2302

Retrieving and changing the log level is now possible using the Admin API.

Fixed issues in Cloudera Data Visualization 7.1.7

VIZ-938, VIZ-1298, VIZ-1317, VIZ-2221, VIZ-2086, VIZ-2203

Several bugs related to filters have been fixed.

- Numeric Range filters now reset as expected.
- Filter parameters passed using URL now override the default or saved values for the filters.
- Forward and back buttons that are available in View mode from the dashboard now advance or undo the filter states as expected.

VIZ-1025

The Edit and Refresh options now consistently appear in the Dashboard Builder for the SQL Queries visual type.

VIZ-2096, VIZ-2215

Performance improvements have been implemented for the Data and the Logs and Usage WebServer Stats pages.

VIZ-2111

Fixed a bug where tooltips on several visual types could display the text Missing instead of the desired field.

VIZ-2113

Fixed a bug which prevented anchoring a custom date range to a desired start date other than the current date.

VIZ-2223

The dashboard deletion confirmation message now appears after the action.

VIZ-2260

A configuration change has been made to allow the correct end-user IPs to be shown in activity logs for CDV applications running in CDSW or CML.

VIZ-2270

Multiple rows of email addresses in the To field when configuring an email job now display as expected.

VIZ-2275

Automatic field renaming during dataset creation now works as expected.

What's new in Data Visualization 7.1.6

Cloudera Data Visualization (CDV) 7.1.6 was released on September 19, 2023 introducing the following changes.

Security fixes and improvements in Cloudera Data Visualization 7.1.6

VIZ-1808, VIZ-2219

Fixed critical and high package vulnerability issues found in CDV 7.1.5.

VIZ-2269

CDW Impala and Hive Warehouse impersonation settings now set properly. Fixed in CDV 7.1.6.2-3.

What's new in Data Visualization 7.1.5

Cloudera Data Visualization (CDV) 7.1.5 was released on August 31, 2023 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.1.5

VIZ-1475

Better visualization for dimensions with mixed negative and positive values on a stacked bar chart is now available.

VIZ-2244

When configuring a data extract, you can now search for a column by name, as well as view the field types and complex data types available.

VIZ-819, VIZ-1695

The Cloudera Data Visualization login page has been updated with a new design, and the currently logged in username is shown when prompting for a password change.

Fixed issues in Cloudera Data Visualization 7.1.5

VIZ-1076, VIZ-1661

The settings options shown in the Dashboard Builder for the Combined Bar/Line (Combo) visual now match the capabilities of the visual.

VIZ-2085

Fixed a bug where Extension visuals caused an error when performing API metadata updates.

VIZ-2104

Fixed a bug where special characters in a visual title resulted in a download file name with the visual title repeated.

VIZ-2126, VIZ-2227, VIZ-2230

Fixed critical and high package vulnerability issues found in CDV 7.1.4.

VIZ-2189

Fixed dropdown behavior for navigation bar on dataset pages.

VIZ-2232

Updated display format and default documentation links.

VIZ-2239

Max Retention Days settings for job logs are now properly respected in all time zones.

VIZ-2247

Snowflake data connections from CDV instances running in CDSW now work as expected.

VIZ-2249

Derived Data measures can now be used in box plot visuals.

What's new in Data Visualization 7.1.4

Cloudera Data Visualization (CDV) 7.1.4 was released on July 27, 2023 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.1.4

VIZ-842

Existing job owners and system admins can now change a job's owner on the Job Status page. Jobs from deleted users will have unassigned owners, and will not be able to run until the owner is changed.

VIZ-1711

Filters now have an information pop-over that provides useful information such as the dataset and data connection the filter was created from. The filter ID is also included for easier performance and query debugging.

VIZ-1814

Site Administration Site Settings have been redesigned to make finding, using, and editing site-wide application settings easier. Users with the "Manage Site Settings" permission will see the new category view as well as a search functionality to make finding a specific control faster.

VIZ-2087 & VIZ 2099

The Manage Dashboard URL Aliases page now allows improved editing of aliases, including changing the redirect to point to a different dashboard or application.

VIZ-2095

CML connections and user permissions now sync automatically for new CDV applications running in CML. You can sync your data connections, similar to CML Data tab deployments of CDV. Additionally, CML project owners, admin, contributor, and site admin will automatically be assigned the system admin role in CDV. All other project collaborators will be visual consumers.

VIZ-2102

You can now search for a user by first or last name from the Manage Users & Groups page.

VIZ-2128

Performance improvements have been implemented to allow for faster application startup.

VIZ-2184 & VIZ-2185

Significant improvements have been made to the Activity Log Current Load and Activity Log WebServer Stats tabs, providing better and more understandable monitoring of active queries and counters.

VIZ-2191

Design changes have been implemented in the Notification Center to improve readability and fix an issue where a lengthy list of new notifications could appear longer than the modal.

Fixed issues in Cloudera Data Visualization 7.1.4

VIZ-958

The greater-than and less-than selectors within the Custom Colors for Measure Value Ranges settings should now display properly on all screen sizes.

VIZ-1820

Configurations for data connections to Solr instances using Knox are now available.

VIZ-1904, VIZ-1907, VIZ-1910, VIZ-2125

Fixed critical and high package vulnerability issues found in CDV 7.1.3.

VIZ-1946

The Explore Visual Types wizard for visuals with a color shelf now works as expected.

VIZ-2047

Fixed dashboard and dataset association migration.

VIZ-2058

Fixed a bug where minimum and maximum aggregates were incorrectly passed within the Histogram visual type, resulting in an unexpected error when building this visual.

VIZ-2112

Dimension color sort order is now consistently applied across all bars of a stacked bar visual, even when some columns are missing specific values.

VIZ-2131

Editing a job from the Job Status page should now work as expected.

VIZ-2160

Fixed a bug where the Dashboard Designer Filters sidebar would occasionally display columns from the wrong dataset upon page refresh.

VIZ-2206

Success and warning alerts should now close properly.

What's new in Data Visualization 7.1.3

Cloudera Data Visualization (CDV) 7.1.3 was released on May 31, 2023 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.1.3

VIZ-775

Unsaved changes are preserved when cloning a dashboard.

VIZ-1811, VIZ-2075

Added API endpoints for creating API keys and editing the user profile.

VIZ-1962, VIZ-2045

Improved URL aliases with a new admin management page and better information in the email modal.

VIZ-2025

Improved performance of SQL tab.

VIZ-2054, VIZ-2060

Improved PNG/PDF download for multi-sheet dashboards and added support for East Asian languages.

VIZ-2063

Added setting adjustment and retry for handling common Hive connection errors.

Fixed issues in Cloudera Data Visualization 7.1.3

VIZ-1919, VIZ-2053

Added fixes for handling partitions, including icon for partitioned field in the dataset page.

VIZ-1934

Fixed display of threshold values when editing an existing job.

VIZ-2027

Added data connection name and type into the Historical logs tab of the Dashboard Profile data.

VIZ-2043, VIZ-2070, VIZ-2084

Various CSV/Excel download fixes including preserving capitalization in Impala, better handling of NAN/INF numeric values, and fixed formatting errors for NULL and incorrect date values.

VIZ-2050

Fixed issue so visual artifacts are always imported with the “Disregard artifacts that exist in other workspaces” option selected.

What's new in Data Visualization 7.1.2

Cloudera Data Visualization (CDV) 7.1.2 was released on April 28, 2023 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.1.2

VIZ-334, VIZ-1963

Short URLs of a specific sheet, dashboard, and set of parameters can be created when getting the URL for a dashboard or application. These short URLs can be easily shared with collaborators.

VIZ-596

A warning is automatically shown to any users accessing CDV in Internet Explorer. As a reminder, for best results Cloudera recommends using Google Chrome or Microsoft Edge browser. Some features may not function properly with other browsers.

VIZ-1211, VIZ-1268, VIZ-1617

Null values can be excluded for all filter types. This ability can be disabled in the 'Basic' filter settings using the 'Permit selection of NULL values' setting.

VIZ-1230

Favorited dashboards are now shown on a user’s default homepage.

VIZ-1368

Improvements for setting the default homepage for all users, per-user specific dashboards as homepages, and resetting the homepage to default.

VIZ-1730

When importing new tables into a data connection, there are now options for immediately creating a dataset or dashboard from the imported table.

VIZ-1990

When deleting a data extract, the option to drop the extract's target table from the database is now available.

Fixed issues in Cloudera Data Visualization 7.1.2

VIZ-980, VIZ-2009

Bug fixes and improvements for the cviz_rest function.

VIZ-1735, VIZ-1915, VIZ-1971, VIZ-1981, VIZ-2012, VIZ-2015

Bug fixes and performance improvements for data extracts.

VIZ-1758

Changing the sort priority on a trellised visual now works as expected.

VIZ-1881

If a query fails due to a parseexception, the failed query will be displayed in the query logs.

VIZ-1917

Improved sorting indicator for tables used in the product.

VIZ-1918

Broken image links on two sample dashboards have been updated.

VIZ-1959

Wrap column contents settings for table visual now works as expected.

VIZ-1997, VIZ-1998

The format editor for the rich text visual and email template builder now works as expected.

VIZ-2011

Using the 'Segment' dimension in a visual's filters shelf now works as expected.

VIZ-2024

Fixed a bug where importing visual artifacts containing dataset segments showed a 'Cannot parse segments' error.

VIZ-2028

Launching the Explore Visuals modal from the Visual Builder now works as expected.

What's new in Data Visualization 7.1.1

Cloudera Data Visualization (CDV) 7.1.1 was released on March 9, 2023 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.1.1

VIZ-1062

Added new settings for hiding the trellis labels for the KPI visual.

VIZ-1420, VIZ-1555

Impala and Hive connections now stream CSV/XLS downloads. This means downloads are no longer constrained by the 'Maximum data rows for visuals' site setting.

VIZ-1631

Improved performance of the 'Users & Groups' page.

VIZ-1683, VIZ-1684

Improved process for detecting errors in advanced site settings. This has also resulted in deprecation of redundant environmental variables for disabling advanced site settings.

VIZ-1781

Made the instructions on 'Setting Workload Password' easier to copy.

VIZ-1873

Job logs older than a year are now deleted every week. Deletion settings are configurable for admins in site settings.

Fixed issues in Cloudera Data Visualization 7.1.1

VIZ-1279, VIZ-1863, VIZ-1877, VIZ-1922, VIZ-1928, VIZ-1952, VIZ-1961

Improved data extracts. This includes enabling extracts on multi-table datasets with partitions, updating extract target and source tags, and other data extract creation improvements.

VIZ-1923, VIZ-1950

Fixed an error that prevented users from editing fields not present in data extracts.

VIZ-1927

Enabled non-admin users with 'View Activity Logs' permissions to view the 'WebServer Stats'.

VIZ-1967

Added 'Switch Dataset' option to range filter and properly saved dataset changes.

What's new in Data Visualization 7.1.0

Cloudera Data Visualization (CDV) 7.1.0 was released on February 3, 2023 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.1.0

VIZ-499

Data Extracts are now available for accelerated visualization. Data Extracts are a tool for managing the analytical capabilities, performance, concurrency and security of data access across data connections. placeholder

VIZ-600

A new notification center allows you to quickly view application errors. For admin users and users with the “View activity logs" privilege, webserver errors (such as issues with applying advanced site settings) are available in this new notification center. All users can view browser-level errors that might be affecting CDV.

VIZ-1728

You can now create datasets or add data to a data connection directly from the homepage.

VIZ-1670

A new "View Users and Roles" privilege can now be set to allow users with Create Workspace permissions to search the user database when adding new members to a workspace.

VIZ-1723

Percentage range can be specified as a custom color for measure value ranges.

VIZ-1859

Functionality added to filter Job Logs by job creator as well as job type and job status.

VIZ-1772

Performance improvement for dashboard loading in View mode.

VIZ-1397

UI improvements to the default CDV Homepage.

Fixed issues in Cloudera Data Visualization 7.1.0

VIZ-792

Dataset-defined color settings should now work as expected when adding a field to the Colors shelf in the Visual Builder interface.

VIZ-1269

URL parameters save and clear as expected when navigating between dashboards and tabs within an application.

VIZ-1457

Temporary dashboard errors related to query cancellation are now reset after requests complete.

VIZ-1505

Duplicate query rows no longer appear in the Show Usage Info > Queries modal.

VIZ-1512

Changing the SQL for a SQL-based dataset refreshes the result cache for the dataset.

VIZ-1540, VIZ-1664, VIZ-1665, VIZ-1754, VIZ-1826, VIZ-1833

Fixed critical and high package vulnerability findings in CDV 7.0.5.

VIZ-1710

Fixed a bug where applying a dataset switch to all visuals on the sheet did not work as expected.

VIZ-1726

Fix for linked visuals to avoid visual cloning.

VIZ-1799

Data Connection types that support data refresh from Connection Explorer now see refresh alongside other connection options.

VIZ-1810

Fix for setting query timeout maximum in cases where customized values were previously set in advanced site settings.

VIZ-1817

Fixed a bug where the bars in a trellised bar chart could overlap each other when multiple filters were selected.

VIZ-1845

Fixed issue where null binary characters could cause email jobs with XSLX file attachments to fail with a snapshot generation error.

VIZ-1847

Visual title edits on dashboards with multiple sheets should now save as expected.

VIZ-1861

Changing column position in a table visual now works as expected.

VIZ-1876, VIZ-1896

Fixed a bug where a filter on a column name that contained a space or special character could return a syntax error in some instances. Fixed in CDV 7.1.0-b3.

What's new in Data Visualization 7.0.5

Cloudera Data Visualization (CDV) 7.0.5 was released on December 7, 2022 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.0.5

VIZ-1685, VIZ-971

Application server errors, including issues applying advanced site settings, will now be displayed in the new notification center.

VIZ-1179

Improved page loading performance with better asset bundling.

VIZ-119

Dashboard filters can now be added by dragging and dropping into the dashboard.

VIZ-1567

Added data loading indication on the homepage.

VIZ-1568

New script was added to clean up duplicate permissions in dataset and data connection roles.

VIZ-1001

Improved title casing logic for datasets, especially when changing data model through additional joins.

VIZ-1671

Connection concurrency by connection or user, as well as query timeout maximum and default values, can be customized in advanced site settings.

VIZ-1765, VIZ-1577

Made changes to prevent duplicate artifacts when importing visuals. Existing artifacts with the same uuid will now be updated.

VIZ-1796

The visuals page search now allows users to search for dashboard IDs associated with apps, or visual IDs associated with dashboards.

VIZ-1416

Added options for vertical in addition to horizontal gridlines in bar, line, area, boxplot, and scatter visuals.

Fixed issues in Cloudera Data Visualization 7.0.5

VIZ-1803

Fixed “Edit Dataset SQL” button in dashboard editor for visuals using SQL-based datasets.

VIZ-1615

Default to “binary” connection mode for Impala connections.

VIZ-1612

Fixed issue where emails ignored custom email templates.

VIZ-1724

Table and crosstab expansion now works properly for quoted values across all connection types.

VIZ-1256

Improve accessibility for resizing columns when editing cross tab visuals.

VIZ-739

Fixed issue with trellised bar graphs that have independent dimensions.

VIZ-1469

Fixed setting for reversing color palette.

VIZ-1807

The query timeout value when editing a data connection is now properly displayed.

VIZ-1669

Fixed issue where edited text in HTML or raw SQL visuals are discarded upon screen resize.

VIZ-1790, VIZ-1516, VIZ-1515, VIZ-1514

Filter and table settings are disabled now if they have no effect due to another setting’s value.

VIZ-1720

Fixed alignment for headline table layout.

VIZ-726, VIZ-1692

Fixed styling for the “Drill Into” and Natural Language Search dropdowns.

VIZ-1718

Fixed issue where global filter changes were saved even after canceling.

VIZ-1716, VIZ-1713, VIZ-1706

Made various style improvements to application navbar, sheet name editing, and issue icon dashboard.

VIZ-1525

Improved activity log column header alignment on large screens.

VIZ-1454

Added indication when there are no CDW Impala or Hive warehouse options.

What's new in Data Visualization 7.0.4

Cloudera Data Visualization (CDV) 7.0.4 was released on October 28, 2022 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.0.4

VIZ-1627

Save selections made in edit mode setting is now available for all filter types. When unselected, filter values selected by the dashboard editor will not be the default in edit mode.

VIZ-1107

Improved the Expansion feature for cross tabulation visuals to more intuitively aggregate across row dimensions.

VIZ-1689

Cross Tabulation visual headers are now more readable, with tooltips for titles that are too long to display.

VIZ-1589

When selecting values for a visual filter, there is now a Select All for bulk selection of the shown filter values.

VIZ-1668

The build tab in Dashboard Designer can now be expanded to allow for greater readability.

Fixed issues in Cloudera Data Visualization 7.0.4

VIZ-938

Range filters should now reset as expected when using the reset icon on a dashboard.

VIZ-1060, VIZ-1143

Fixed bugs related to color selection on custom fields and when adding custom background colors to dimension and measures in a crosstab or table visual.

VIZ-872, VIZ-1518, VIZ-1639, VIZ-1638

CSV and Excel download improvements and bug fixes.

VIZ-1275, VIZ-1641, VIZ-1007

Optimized application menu and dashboard sheet rendering for smaller screens.

VIZ-1357

The export button on the Visuals page now behaves as expected.

VIZ-1676

Remember previous selections picklist setting has been renamed to “Remember previous selections when searching” for specificity.

VIZ-1500

Dataset search should now work as expected.

VIZ-1573

Field statistics histogram should now display frequency values in tooltips as expected.

VIZ-1608

Adding users or groups via Edit Workspace modal should now work as expected.

VIZ-1640, VIZ-1719

Bug fixes for creating and editing scheduled jobs

VIZ-1633

Special characters that are not possible to display should now show an icon indicating as such, rather than throwing an error.

VIZ-1634

Users that do not have access to one or more datasets used in a dashboard will now see an error message warning them of such, rather than a blank dashboard.

VIZ-1702

Fixed a bug where line labels on a combined bar/line visual for which display formatting has been applied did not appear correctly.

VIZ-1644

Fixed a bug where editing the title of a picklist filter did not save changes correctly.

VIZ-1637

Pagination for SQL page now appears as expected.

VIZ-1702

JS error when line labels are enabled with display formatting for combined visuals.

Fixed in CDV 7.0.4-b41

VIZ-1733

Switching dashboard sheets in view mode results in JS error.

Fixed in CDV 7.0.4-b41

VIZ-1742

Trigger Jobs fail with "name 'username' is not defined" NameError.

Fixed in CDV 7.0.4-b41

VIZ-1743

Emailing jobs fail when SSL and TLS errors are disabled.

Fixed in CDV 7.0.4-b41

What's new in Data Visualization 7.0.3

Cloudera Data Visualization (CDV) 7.0.3 was released on September 30, 2022 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.0.3

VIZ-1480

Daterange filters can be set to default to the max date rather than a fixed date.

VIZ-1519

CSV and Excel downloads show additional information when download or view fails because of global download row or size limit.

VIZ-1543

Dataset field names no longer allow parenthesis to avoid potential issues if used in filters.

VIZ-1463

Added settings to fine-tune the appearance of the alert when a query is taking an unusually long time, and enable/disable alerts for this purpose in View mode.

VIZ-1560

Query canceling is now immediately visible for each visual as it loads.

VIZ-1341

CDW Impala and CDW Hive connections now include Client Identifier and Application Name with a default of ‘viz’. This is configurable from connection settings.

VIZ-1452

It is now possible to filter for only the rows that have NULL for a particular field value.

VIZ-801

Improvements for the Packed Bubbles visual type, including better handling of dimensions placed on the Colors shelf and enablement of legend.

VIZ-863, VIZ-1053, VIZ-1642

Improvements for resizing visuals and dashboard width in Edit mode.

VIZ-1290, VIZ-1359, VIZ-1360

Improvements to the Crosstab visual type, including aligning the totals row, showing adjacent duplicate values, and allowing text selection in View mode.

Fixed issues in Cloudera Data Visualization 7.0.3

VIZ-1532, VIZ-1533, VIZ-1534, VIZ-1539

Fixed critical and high package vulnerability findings in CDV 7.0.2.

VIZ-1188, VIZ-1485, VIZ-1600, VIZ-1601, VIZ-1602, VIZ-1663

Improved the dataset selector for visuals and filters in the dashboard builder:

- The selected dataset in the filter tab now shows the correct fields and the data connections selector is always enabled.
- The data connections on the filters tab are now alphabetically sorted.
- All edited visuals are now listed as recent visuals in the visuals tab.

VIZ-1414, VIZ-1636

Fixed errors when downloading data as excel:

- Excel downloads that use results from cache no longer return just the headers and one column of data.
- Downloading fields with date display formats no longer throws an error.

VIZ-1526

Fixed a bug where renaming a column in the dataset breaks previously created dashboards using that column as a filter.

VIZ-1622

Fixed errors with editing and creating scheduled intervals in the Email form.

VIZ-787

Fixed a bug where filtering using an in-grid and app-scope filter did not apply on the target dashboard as expected.

VIZ-1565

Search bar in “Pick a Dashboard” modal during Sheet import now works as expected.

VIZ-1595

Fixed a bug where some users reported seeing the “What’s New” modal upon each login, even after clearing the modal.

VIZ-1591

Improved auditing of visual deletion when deleting dashboards and datasets.

VIZ-1562, VIZ-1651

Cloning a visual within a dashboard now works as expected, and metadata for cloned visuals now correctly updates create_user and update_user.

VIZ-1587, VIZ-1618

Fixed a bug where editing and viewing different sheets on a dashboard resulted in unexpected sheets being loaded if the sheets had been rearranged from their initial positions.

VIZ-1388, VIZ-1593

Bug fixes for Bar and Line visual types, including fixing the appearance of labels.

VIZ-1474, VIZ-1561

Bug fixes for display formats configured within a dataset.

VIZ-1409, VIZ-1435, VIZ-1605, VIZ-1610

Bug fixes and improvements for emailing dashboards and configuring email jobs.

VIZ-918, VIZ-1294, VIZ-1619, VIZ-1624

Bug fixes and improvements for the Data Extracts technical preview feature.

VIZ-1652

Non-legacy Impala and Arcengine connections will show comment prefixes indicating the visual, dashboard, and dataset that triggered the query

VIZ-1605

If the user hasn't dismissed the "What's New" modal using the "Got It" button, it can appear in emailed dashboards.

What's new in Data Visualization 7.0.2

Cloudera Data Visualization (CDV) 7.0.2 was released on August 9, 2022 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.0.2

VIZ-1364

It is now possible to search for dashboards, applications, datasets, or documentation from anywhere in the application.

VIZ-935, VIZ-1401, VIZ-1349, VIZ-1003

Several improvements to filters and filtering:

- Improved display of the user-added values in filters.
- Relative date filters are now included in URL parameters and email jobs.
- NULL values can be included or excluded in date filters.
- Filter width settings can now be adjusted as expected.

VIZ-1338

New site setting to allow administrators to disable manual user creation from within the application. This helps preventing username collisions between SSO users and manually created users.

VIZ-1177

The workspace description is now shown on hover when viewing the list of workspaces from the Visuals tab.

VIZ-1231

A "Copy to Clipboard" button was added to the Get URL modal to make sharing parameterized links easier.

Fixed issues in Cloudera Data Visualization 7.0.2

VIZ-1361, VIZ-1300, VIZ-1363

Fixed critical and high package vulnerability findings in CDV 7.0.1.

VIZ-1568

Fixed a known issue where Impala connections using kerberos resulted in a connection error that was discovered in the 7.0.2-b35 version of 7.0.2.

VIZ-1472

Fixed a known issue where updating CDV from 6.4.1 to 7.0.1 in CDSW or CML required an additional environmental variable to be set in the project.

VIZ-829

Fixed a bug where clicking the CANCEL button in the Edit User or New User modals could display an unexpected validation error instead of closing the modal.

VIZ-1486

Fixed a bug where emailing a dashboard failed with a security protocol error.

VIZ-890

Non-admin users with “Manage roles and users” permission can now create, edit, and delete roles as expected.

VIZ-601

Bug fixes and improvements for the Data Extracts technical preview feature.

VIZ-1412

Direct Access SQL now applies CSV download limits as expected.

VIZ-1478

Fixed a bug where users would see an application navigation error when a dashboard within the application contained an in-grid filter.

VIZ-809

Data type icons now appear as expected when adding fields to filter shelves.

VIZ-1419

Fixed a bug where direct access could show "no available connection" message unexpectedly.

VIZ-1499

Datasets created from SQL will no longer show a truncated query in some instances.

VIZ-1459

Axis scale disabling now works as expected with multiple aggregates on the color shelf.

VIZ-1358

Fixed a bug when editing a dashboard where the context menu on a visual required an additional click to open.

VIZ-1483

Fixed a bug where the Left, Center and Right Align icons didn’t appear next to their respective radio buttons in the Dashboard Settings pane.

What's new in Data Visualization 7.0.1

Cloudera Data Visualization (CDV) 7.0.1 was released on June 29, 2022 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.0.1

VIZ-1396

[Tech Preview] Data connections to Snowflake are now available as a technical preview feature.

VIZ-1315, VIZ-1427, and VIZ-1441

Introduced a per-user concurrency setting in data connections that allow users to limit the number of queries a single user can occupy. This prevents one user from blocking connection access for an entire connection. Added a Current Load panel to the Profile Data modal that allows users to view and manage ongoing queries.

VIZ-1336 and VIZ-1149

Added dataset switching to filters, so users can quickly change the filter across datasets and connections. Users can now also quickly change the dataset for all filters and visuals in a sheet.

VIZ-672

The dashboard layout interface now includes names and images of relevant visuals for easier visual identification when moving and resizing visuals.

VIZ-1351

When editing a Dashboard's style in the Dashboard Designer modal, it is now possible to reset all styles to original.

Fixed issues in Cloudera Data Visualization 7.0.1

VIZ-274, VIZ-284, VIZ-802, VIZ-1432, VIZ-1433, VIZ-1434, VIZ-1444

Security fixes and usability improvements.

VIZ-540 and VIZ-1295

Fixed a bug with importing visual artifacts with segments or time wrap or where the dashboard being imported had been cloned within another CDV instance.

VIZ-645

Dashboard and visual background color styles now interact and reset as expected.

VIZ-652

Improved timeout process for chrome processes initiated by thumbnail generation.

VIZ-740

Error messages when importing data from a CSV file should now appear and reset as expected.

VIZ-837

Users logging in for the first time will no longer see a message about their last login occuring on an invalid date.

VIZ-923

Fixed a bug where deleting app scope filters could leave an empty filter widget in Dashboard sheets.

VIZ-1196

Email protocol settings are now selectable as expected.

VIZ-1311

HTML visuals and custom filters now appear in emailed dashboards as expected.

VIZ-1337

SQLite data connection creation is now disabled in CDV instances running in CDV, as locally created SQLite files are transient in that environment.

VIZ-1345

Editing Scheduled Email jobs should no longer throw an error on the schedule interval.

VIZ-1370

Fixed a bug where a user could see a reference error when building a visual with a secondary axis.

VIZ-1390

Fixed a bug where a dashboard's filter parameter popup could become stuck if clicked.

VIZ-1392

Dashboard and visual info now shows upon hover when adding a recently created visual to a dashboard using Dashboard Builder.

VIZ-1406

Fixed a bug with the Table Browser that could prevent joining new tables in an existing Dataset.

VIZ-1445

Direct access functionality will now appear only on supported connection types.

What's new in Data Visualization 7.0.0

Cloudera Data Visualization (CDV) 7.0.0 was released on May 17, 2022 introducing the following changes.

New features and improvements in Cloudera Data Visualization 7.0.0

VIZ-20

Some major updates have been made to the Bootstrap and jQuery libraries. Bootstrap was upgraded from 3.3.7 to 4.6.0 and jQuery was upgraded from 2.2.4 to 3.6.0. This change has a potential impact to customers who have created Custom Styles in earlier versions of CDV, since they would potentially contain code that references legacy Bootstrap classes or jQuery methods in their CSS and/or JS respectively. For more information, see the *Custom Styles Migration Guide*.

VIZ-1115 & VIZ-1217

You can query your data directly through a new SQL tab in the top navigation bar, and a new homepage layout has been introduced so you can quickly add and access recent queries, data connections, and datasets alongside their dashboards and applications. From the SQL tab, you can now create a new dataset or dashboard from a query with a single click. Additionally, when editing a dashboard, updating a SQL-based dataset is even easier with the new Edit Dataset SQL option from the in-visual options menu.

VIZ-1328

[Tech Preview] Data Extracts are now available as a technical preview feature. Data Extracts help with managing the analytical capabilities, performance, concurrency, and security of data access across data connections. For more information, contact your Cloudera representative.

Fixed issues in Cloudera Data Visualization 7.0.0

VIZ-1220

Fixed a bug where a dashboard could become uneditable after disabling Custom Styles.

VIZ-1270

Fixed a bug where roles were not deletable from the Manage Roles interface.

VIZ-1313

Additional bug fixes and usability improvements.

VIZ-1333

Dashboards now appear in a searchable list in the Set Homepage interface.

VIZ-1339

Show/Hide Password toggles in the Change My Password modal now work as expected.

VIZ-1367

Visuals without a secondary axis should no longer display settings options for the secondary axis.

December 17, 2021 - CVE-2021-44228

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.3.5-b5

Cloudera released a new version of CDP Data Visualization (CDV). This new version addresses CVE-2021-44228 which affects products using Apache Log4j2 versions 2.0 through 2.14.1.

CDV is not written in Java and therefore is not vulnerable. However, as the custom CDP Data Visualization engine is built upon a CML engine, it has been updated to the newly released version of the CML engine that removes the vulnerability reported in CVE-2021-44228. For more information about the new CML engine, see CVE-2021-44228 Remediation for CML Data Service.

If you use CDV with Cloudera Machine Learning, update your CML engine to the latest version, which includes fixes for the log4j2 security vulnerability in CML. After updating the engine to the latest version, running CDV instances (applications) should be restarted.

If you use CDV with Cloudera Data Science Workbench, no action is required.

December 10, 2021

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.3.4-b13

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.3.4-b13

New features and improvements

VIZ-810

When adding a filter shelf to a visual, it is now possible to add and edit a custom filter expression directly from the filter modal.

VIZ-896

Cross Tabulation, Row Listing, and Table visuals now use a default number format with comma separators if a display format is not specified.

VIZ-917

Dashboard snapshot timeouts are now configurable by using an advanced site setting flag.

VIZ-925

The Dataset ID is now shown for each dataset listed on the Dataset overview page.

Fixed issues

VIZ-436

Dashboard-level settings for showing detail data in context menu now work as expected.

VIZ-915

Fixed a bug where dashboard titles could appear editable in view mode.

VIZ-916

Dashboard reset after drill-into now refreshes the dashboard as expected.

VIZ-952

Fixed a bug where setting the default homepage for all users as an admin user failed to save.

VIZ-986

Fixed a bug where importing a dashboard and running a data table compatibility check resulted in an unexpected error.

November 8, 2021

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.3.3-b43

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.3.3-b43

New features and improvements

VIZ-475

Checking dataset compatibility when importing a dataset from SQL is now possible.

VIZ-612

Including or excluding a data point from a visual now shows a confirmation message prior to visual refresh.

VIZ-680, VIZ-745, VIZ-877 and VIZ-878

Several improvements have been implemented for the Timeline visual type, including adding default x-axis formatting, easier display format selection, and the option to add a legend.

VIZ-721

You can now use the EDIT button on the dataset field tile to streamline the dataset editing process.

VIZ-750

When configuring custom site settings, errors encountered in the startup process are now logged and bypassed instead of blocking the startup, to allow for easier debugging.

VIZ-847

Configuring a visual's click behavior is now easier with new context menu options available in build mode.

VIZ-908

Job logs can now be filtered by job type and status.

Fixed issues

VIZ-385

Fixed a bug where only one column was shown for large tables in Sample Data inside the Connection Explorer.

VIZ-414, VIZ-862

Table fields with varchar, large, or irregular characters in comments are now parsed properly during dataset creation.

VIZ-696, VIZ-901

Other bug fixes and UI improvements.

VIZ-732

Fixed a bug where the menu for an on-dashboard filter could appear behind the visual.

VIZ-859, VIZ-860

Fixed data connection modal bugs where multiple empty parameter rows could be added, or non-numeric values could be saved in fields where they could cause errors.

VIZ-876

Downloaded Detail Data from a visual is now filtered as expected.

VIZ-942

Fixed a bug where CDV instances using a MySQL metadata store could see an error while upgrading.

September 30, 2021

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.3.2-b29

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.3.2-b29

New features and improvements

VIZ-398

A new style setting allows the user to change the font weight of row and column summaries. Data summary settings have also been moved into their own subsection for better discoverability.

VIZ-473

Dashboards created for the purpose of threshold-triggered jobs are now editable.

VIZ-713

It is now possible to chain multiple conditions for threshold based emails.

VIZ-689. VIZ-848, VIZ-861, VIZ-880, VIZ-897

Additional UI and usability improvements.

VIZ-234, VIZ-641, VIZ-835

Other security fixes and improvements.

Fixed issues

VIZ-99 & VIZ-304

Users with first names or last names longer than 30 characters can now be created as expected.

VIZ-316

Downloads of SQL visual types now inherit global limit settings from Site Settings as expected.

VIZ-384 & VIZ-886

Fixed a bug where CSV or Excel download could apply an unwanted default display format to some columns, or fail to show a display format that had been added.

VIZ-418

Fixed a bug where infinity or NaN values in Impala caused an error.

VIZ-545

PDF or PNG downloads for slow-loading dashboards now display fully rendered visuals rather than progress bar icons.

VIZ-693

Fixed a bug where a label on a scatter visual would be hidden while hovering.

VIZ-714

The create threshold functionality in the email jobs modal now works as expected.

VIZ-722

Highlighting elements by hovering over legend items now works as expected.

VIZ-724

The visual setting option to hover over a dimension and show each point with that dimensional value functionality now works as expected.

VIZ-747

Using an image in an Action visual now works as expected.

VIZ-757

Fixed a bug where time control in a dashboard's filter bar was not removable once added to dashboard.

VIZ-784 & VIZ 808

Display formats should now appear as expected in Grouped Bar and Row Listing visuals.

VIZ-790

Dimension color palettes now accept typed-in choices when configuring background colors for Table and Cross Tabulation visuals.

VIZ-804

Fixed a bug where rearranging elements in Heatmap visual shelves was not saved as expected.

VIZ-832

Email jobs with send limits greater than once per day should now be triggered as expected.

VIZ-857

Default date formatting on x-axis of bar visuals should now work as expected.

VIZ-885

Setting CSV download locale in the dashboard and visual settings now works as expected.

VIZ-891

Fixed a bug where LDAP Mirror Groups did not work as expected in version 6.3.1.

August 24, 2021

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.3.1-b27

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.3.1-b27

New features and improvements

VIZ-383

Display formats are now applied as expected on CSV and XLS downloads when the data connection type is Impala or Arcadia.

VIZ-434

You can drill-into your data from the context menu to discover more granular information for a particular data point by examining other dimensions of your data.

VIZ-647

New icons have been added to the username display based on user type — an asterisk appears for admin users and a checkmark appears for LDAP/SSO-signed users.

VIZ-700

Usability improvements have been implemented to the dashboard builder sidebar.

VIZ-719

The full dataset name can now be viewed on hover in dashboard builder menus.

VIZ-736

You can now save a segment from a filter definition of a visual.

VIZ-770

Creating and editing HTML extension visuals is now easier with syntax highlighting, autocompletion, and side-by-side preview.

VIZ-773

Additional details on job parameters, including trigger conditionals and active states can now be viewed from the Job Status page.

Fixed issues

VIZ-666

Fixed a bug where a filter's 'Maximum number of rows to fetch' setting unexpectedly affected the filter limit.

VIZ-667

Filters with a slide range selector now allow for precise selection of range values.

VIZ-691

Dashboards with multiple filters that are linked and given app scope now properly apply across all sheets.

VIZ-759

Fixed a bug where the relative dates timestamp filter reset upon switching between sheets of a dashboard.

VIZ-796

Fixed a bug where the Direct Access SQL editor applied the default limit even when the user specified a different value.

VIZ-820

Fixed a bug where changes in the Edit User modal could not be saved without retyping the current user password.

July 30, 2021

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.3.0-b53

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.3.0-b53

New features

VIZ-367, VIZ-715 - Data Visualization as Machine Learning Runtime

Data Visualization is available as a Machine Learning Runtime for Cloudera Machine Learning (CML) on CDP. For more information, see the ML runtime documentation.

VIZ-619 - Natural Language Search

Natural Language Search (NLS) is now generally available. You can speed up visual creation and data analysis insights by enabling search-based data exploration. For more information, see the NLS documentation.

VIZ-639 - SSB data connection [Tech Preview]

A new data connection type is available in technical preview: you can connect to SQL Stream Builder (SSB), an integrated job management interface. SSB is a part of Cloudera Streaming Analytics powered by Apache Flink. For more information on SSB, see the Introduction to SQL Stream Builder.

Improvements

VIZ-278

Improvements have been implemented for table expansion and sorting within an expanded table column.

VIZ-359

You can export all columns in CSV without selecting or forcing maximum column display to fit the entire column list.

VIZ-452

If you use the default credentials to login to Data Visualization, you will be prompted to change your password at the first login.

VIZ-454

Editing an HTML Extension visual is now easier with a larger modal.

VIZ-491

If you use Oracle metadata connections, you can now configure to connect using a Service Name.

VIZ-549

If you have dashboard editing permissions, you can view the clone history of a dashboard in the Usage Info menu.

VIZ-621, VIZ-623, VIZ-687

Multiple improvements have been implemented for email jobs, including the ability to configure a job to send only on the first trigger, XLS attachment options, and the ability to include a message along an embedded image sent via email.

VIZ-640

Views are shown along with tables on PostgreSQL connections in the Connection Explorer.

VIZ-668

If you are an admin user, you can access the Daily Usage metrics page by clicking through the view count on the Data Visualization homepage.

VIZ-683

You can select any specific color from a color palette as the default or starting color for a visual.

VIZ-710

Improvements have been implemented for migrations on mySQL metadata connections.

Fixed issues

VIZ-343

Fixed a bug where the table expansion tooltip displayed HTML elements when you selected 'Send all fields as parameters'.

VIZ-361

Refreshing an External Link visual now works as expected.

VIZ-495

Enabling highlighting on table visuals using a Solr connection now works as expected.

VIZ-528

Fixed a bug where the 'Download As' options were hidden under the neighboring menu.

VIZ-548

Fixed a bug where the Direct Access SQL editor may run queries as lowercase.

VIZ-650

Date and Time Functions on Hive connections now work as expected.

VIZ-651

Fixed a bug where casting a field resulted in any custom expression getting overwritten.

VIZ-662

Users may now copy the job status error text by clicking on the error icon.

VIZ-664

Fixed a bug where crosstab download limits did not apply as expected for CSV downloads.

VIZ-673, VIZ-674

The Custom Styles page now shows dashboards and applications that use custom CSS, not just visuals.

VIZ-690

The 'Set as Homepage' option is now visible from dashboards but not individual visuals.

VIZ-717

Fixed a bug where sorting on a Y-trellis did not work as expected.

VIZ-753

Fixed a bug where Custom Styles failed to load on the Manage Custom Styles page as expected.

June 4, 2021

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.2.7-b48

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.2.7-b48

New features and improvements

VIZ-84

CDP Data Visualization applications running in CML can be configured with unauthenticated access.

VIZ-433

When configuring email jobs, you can specify a reply-to email address that is different from the sender's email address.

VIZ-464

Users with manage users and roles permissions can now assign roles as expected.

VIZ-484

You can toggle between filter types when creating a filter for a date or for a numeric field.

VIZ-552

Field comments are shown by default when viewing a dataset.

Fixed issues

VIZ-147

The Solr connection type no longer permits expression creation that may cause visual failure.

VIZ-358

Fixed a bug where the use of advanced site settings along with a non-default metastore resulted in an unexpected error.

VIZ-465

Disabled legacy connection types by default, which had bugs that could resulted in an unexpected connection error.

VIZ-469

Smart acceleration now works as expected on the Arcadia connection type when the environmental variable ENABLE_LEGACY_DATACONN is set to true.

VIZ-542

Fixed a bug where visual titles did not substitute the correct parameter in edit mode.

VIZ-563

In-product help text is now shown as expected when building a list visual.

VIZ-605

Fixed a bug where the scrollbar in an application's navigation bar did not appear as expected on narrow screens.

VIZ-606

The Search button in an application's navigation bar now appears as expected on narrow screens.

VIZ-632

Area graphs that contain a date range now draw as expected.

VIZ-636

Fixed a bug where editing a shelf element's expression may have resulted in an unexpected error.

VIZ-637

Editing an email job without an attachment now works as expected.

April 29, 2021

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.2.6-b23

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.2.6-b23

New feature

VIZ-152 – Single sign-on in CDW

Single sign-on (SSO) is now enabled for Cloudera Data Visualization (CDV) applications running in Cloudera Data Warehouse (CDW). For more information, see Start Data Visualization in CDW.

Improvements

VIZ-389

Numerous bug fixes and improvements for Natural Language Search [Tech Preview] functionality.

VIZ-424

Timestamp columns are now auto-recognized in Druid data connections.

VIZ-461, VIZ-532

Improvements for threshold job creation.

VIZ-472

Improvements to sample dashboards that ship with CDV by default.

VIZ-597

Improvements for displaying dashboards on smaller screens.

Fixed issues

VIZ-353

For Hive and Impala connections, CDV can now pass down an application ID in opened sessions for better visibility and auditing into query workloads.

VIZ-422, VIZ-457

Error messages now appear as expected if a CSV/Excel download fails to complete.

VIZ-474

Fixed a bug where importing a dataset from SQL could result in an unexpected error.

VIZ-483

Fixed a bug where the Ignore missing dates setting did not work as expected on line and area charts.

VIZ-529

The Refresh button now appears only on data connections where this query is supported.

VIZ-541, VIZ-543

Fixed a regression from CDV 6.2.5 where visual context menu items did not react as expected.

VIZ-557

Admins can now update their own user profiles through Site Settings as expected.

VIZ-562

Fixed a bug where the PostgreSQL data connection settings did not save as expected.

April 7, 2021

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.2.5-b25

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.2.5-b25

New features

VIZ-388 – Natural language search

[Tech Preview] Use natural language search to find relevant information in your datasets through natural language statements, and render it in graphical format.

VIZ-468 – Account lockout

You can enable lockout after too many failed login attempts by adding `AXES_ENABLED` to the advanced site settings in CML or CDSW. This feature is disabled by default.

Improvements

VIZ-145

Improved Solr aggregation and faceting.

Fixed issues

VIZ-362, VIZ-420, VIZ-533

Fixed bugs for CSV and Excel downloads.

VIZ-387

Error message popovers now close when clicking elsewhere on the page as expected.

VIZ-408

Expansion on crosstab visuals now works as expected.

VIZ-415

Changing values on a custom picklist filter in dashboard mode now works as expected.

VIZ-419

Fixed a bug where filters on an Arcengine connection did not reset after unselecting.

VIZ-437

Enable "Download as Image/PDF" switch in dashboard settings now works as expected.

VIZ-486

Dashboards created from Arcengine connections can now be accelerated as expected.

VIZ-498

Changing colors in a scatter visual now redraws the visual as expected.

February 26, 2021

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.2.4-b16

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.2.4-b16

New feature

VIZ-72 – Job scheduling and customization

Email jobs can now be customized or changed post-creation, making it easy to change the send schedule, add additional recipients, or update the messaging to accompany a dashboard.

Usability improvements

VIZ-350, VIZ-406

Options that are not available on particular data connections are now hidden as expected.

VIZ-380

New Apply Display Format checkbox in the Data Model tab applies selected display formats to the shown data.

VIZ-407

Improvement to Oracle metadata database support.

VIZ-413

Autocomplete options in expression editor now includes ML Model API.

Fixed issues

VIZ-180

Importing access to Private workspaces when Public workspace access is disabled now works as expected.

VIZ-316

SQL Visual limits are now inherited from global limits and table limits as expected.

VIZ-323

Fixed a bug where PDF download of a dashboard that had not loaded fully resulted in impartial PDF download.

VIZ_335

PDF downloads of dashboards with multiple sheets now work as expected.

VIZ_344

Combo charts now allow for more granular control of aggregate axis size.

VIZ-378

Large Excel or CSV downloads that fail to complete are removed from temporary files.

VIZ-379

SQL Visuals downloaded as Excel should now work as expected.

VIZ-412

Fixed a bug where Show Selected and Show All in filter list selection did not work as expected.

VIZ-425

Fixed a bug where setting visual_max_dataresults in Site Settings could result in query failure.

VIZ-432

Improved horizontal scrolling on dashboards with multiple sheets.

February 2, 2021

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.2.3-b18

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.2.3-b18

New features and improvements

VIZ-317

Applications can now be configured to always display the dashboard titles within a tab.

VIZ-330

Additional in-product documentation has been included in the New Data Connection and Edit Data Connection modals.

VIZ-349

Refreshing a dataset cache now correctly logs the user and timestamp for when the cache was refreshed.

VIZ-358

Added support for Oracle metadata databases in CDSW Data Visualization instances.

VIZ-376

Make it easy to configure external metadata store with env variables for CDV in CDSW & CML.

VIZ-382

Added support for connecting to legacy ArcEngine databases from Data Visualization.

Fixed issues

VIZ-122

Fixed a bug where using custom aggregates resulted in an error.

VIZ-245

Display formats are now applied as expected in csv/xls downloads.

VIZ-263

Fixed a bug where a timestamp value could display differently on a dashboard than in an Excel download.

VIZ-351

Aggregates and functions in Impala and Hive connection types now work as expected.

VIZ-363

Sample mode is now hidden for unsupported connections.

VIZ-377

Fixed a bug where migration to Data Visualization from legacy product could result in a workspace error.

VIZ-386, VIZ-400

Searching within a filter and making multiple selections now works as expected.

VIZ-411

Threshold-based jobs now work as expected.

December 15, 2020

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.2.2-b383

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.2.2-b383

Fixed issues

VIZ-229

Frequent CML/CDSW ping requests are now filtered out from Webserver logs to increase log readability.

VIZ-239, VIZ-337

Apps without a thumbnail image now display a placeholder as expected.

VIZ-296

Advanced settings are now disabled by default in CDW versions of CDV.

VIZ-341

Fixed a bug that prevented attaching CSV files to emailed dashboards.

VIZ-342

Fixed a bug where crosstab sort buttons failed after scrolling on a visual.

VIZ-346

Linked visuals must now be edited on their original dashboards, to prevent inadvertent breakage on a dashboard.

December 3, 2020

New CML engine: docker.repository.cloudera.com/cloudera/cdv/cmldataviz:6.2.1-b12

New CDSW engine: docker.repository.cloudera.com/cloudera/cdv/cdswdataviz:6.2.1-b12

New features and improvements

CDP Data Visualization applications running in Cloudera Machine Learning released on November 23, 2020 or later can permit SSO authentication in admin settings.

Fixed issues

VIZ-322

Changing cache settings no longer causes the data connection type to unset itself.

VIZ-321

Fixed a bug that prevented some users from downloading a visual as a csv or Excel.

VIZ-329

Fixed a bug that displayed an error message when importing visual artifacts to a private workspace.

VIZ-328

Data Compatibility check when importing visual artifacts now works as expected.

VIZ-286, VIZ-295, VIZ-298

The correct version number and product name now display in all Data Visualization applications.

VIZ-137

Fixed a bug that caused expansion field values to throw an error when clicking Show Detailed Data or when including/excluding the value as a filter on click.

VIZ-320

Dashboard filter parameters now load as intended when sharing a dashboard via URL.

VIZ-310

SQL Visual Download CSV options now work as intended.

VIZ-326

Save as Table or Dataset functionality no longer displays an error message on Impala connections.

VIZ-270

Provided further in-product instructions for admin settings that can accommodate multiple inputs.

October 30, 2020

This marks the General Availability (GA) of the first CDP Data Visualization release. Data Visualization, the latest Cloudera feature built into the CDP platform can be accessed through Cloudera Data Warehouse (CDW) and Cloudera Machine Learning (CML) in the Public Cloud. It can also be accessed on-prem in CDSW.

It allows data engineers, data scientists, business analysts, and business users to quickly and easily explore data, collaborate, act on, and communicate explainable insights across the data lifecycle.

Fully integrated with CDW, CML, and CDSW, CDP Data Visualization (CDV) provides self-service and drag-and-drop visualization from both enterprise data and predictive models natively in CDP.

CDV enables:

- Fast, intelligent reporting – Rapid, out-of-the-box dashboarding and application building with built-in visual recommendations
- Intuitive workflows, secured by SDX – Easy-to-use visual UI for fast data exploration and instant sharing anywhere without moving data or creating silos
- Integrated data lifecycle collaboration – Accelerate insight sharing with a consistent, integrated data visualization experience across all data and business teams

Known issues and limitations in Cloudera Data Visualization

You can find here the list of known issues and limitations that you might experience while using Cloudera Data Visualization.

Known issues and limitations in Cloudera Data Visualization 7.2.9

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

AI redacting does not remove contextual information

AI visual payloads are included in historic query logs and webserver logs, regardless of the site setting.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-3061

Known issues and limitations in Cloudera Data Visualization 7.2.8

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

AI redacting does not remove contextual information

AI visual payloads are included in historic query logs and webserver logs, regardless of the site setting.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-3061

Known issues and limitations in Cloudera Data Visualization 7.2.7

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Older releases

You can find here the list of known issues and limitations that you might experience while using earlier versions of Cloudera Data Visualization.

Known issues and limitations in Cloudera Data Visualization 7.2.6

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Cloudera Data Visualization upgrade from 7.2.5 to 7.2.6 broken with Postgres metastore

Upgrading Cloudera Data Visualization instances with a PostgreSQL metadata store from version 7.2.5 to 7.2.6 may fail due to a compatibility issue with the PostgreSQL metastore..

Workaround:

To avoid this problem, upgrade directly to version 7.2.7 or a later version where the issue has been resolved.

Versions affected:

Cloudera Data Visualization 7.2.5-7.2.6

Fixed in version:

N/A

Cloudera issue:

VIZ-2868

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Cloudera Data Visualization 7.2.5

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Cloudera Data Visualization upgrade from 7.2.5 to 7.2.6 broken with Postgres metastore

Upgrading Cloudera Data Visualization instances with a PostgreSQL metadata store from version 7.2.5 to 7.2.6 may fail due to a compatibility issue with the PostgreSQL metastore..

Workaround:

To avoid this problem, upgrade directly to version 7.2.7 or a later version where the issue has been resolved.

Versions affected:

Cloudera Data Visualization 7.2.5-7.2.6

Fixed in version:

N/A

Cloudera issue:

VIZ-2868

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Cloudera Data Visualization 7.2.4

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.2.3

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.2.2

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.2.1

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.2.0

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

Dashboard export from Visuals page results in corrupted JSON

Exporting a dashboard from the Visuals page results in corrupted JSON, impacting the successful export and import of dashboards from the page. The issue manifests during the export process, resulting in a JSON structure that only contains workspace details.

Workaround:

To resolve this issue, consider the following workarounds:

- **Single report export**

For exporting a single report, use the export action within the Dashboard Builder. This method avoids the JSON corruption encountered on the Visuals page.

- **Multiple reports export**

For exporting multiple reports, access the Visuals page with the query parameter ?mode=old. This opens the old Visuals page, where the export functionality works correctly, ensuring the integrity of the JSON structure.

Versions affected:

Cloudera Data Visualization 7.1.9 and 7.2.0

Fixed in version:

Cloudera Data Visualization 7.2.1

Cloudera issue:

VIZ-2493

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.1.9

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

Dashboard export from Visuals page results in corrupted JSON

Exporting a dashboard from the Visuals page results in corrupted JSON, impacting the successful export and import of dashboards from the page. The issue manifests during the export process, resulting in a JSON structure that only contains workspace details.

Workaround:

To resolve this issue, consider the following workarounds:

- **Single report export**

For exporting a single report, use the export action within the Dashboard Builder. This method avoids the JSON corruption encountered on the Visuals page.

- **Multiple reports export**

For exporting multiple reports, access the Visuals page with the query parameter ?mode=old. This opens the old Visuals page, where the export functionality works correctly, ensuring the integrity of the JSON structure.

Versions affected:

Cloudera Data Visualization 7.1.9 and 7.2.0

Fixed in version:

Cloudera Data Visualization 7.2.1

Cloudera issue:

VIZ-2493

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.1.8

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

User group and administration designation delay in updates

User groups and administration designation is not updated until a user is authenticated. This means that CDW and CML users who have had user group or permission changes will not see the modifications if they remain logged in from before the change was made.

Workaround:

To resolve this issue, log out and reload Cloudera Data Visualization.

Versions affected:

Cloudera Data Visualization 7.1.8

Fixed in version:

Cloudera Data Visualization 7.1.9

Cloudera issue:

VIZ-2396

Error when creating dataset on connection with result caching

Creating a new dataset on a data connection with connection-level result caching enabled can return an “error: DataSet matching query does not exist” error.

Workaround:

To resolve this issue, disable Result Cache on the affected data connection.

Versions affected:

Cloudera Data Visualization 7.1.7 - 7.1.8

Fixed in version:

Cloudera Data Visualization 7.1.9

Cloudera issue:

VIZ-2425

'Everyone' group option has to be manually specified in workspace permissions

The 'Everyone' group is not shown as an available user group that can be applied when specifying visual workspace permissions.

Workaround:

Type 'Everyone' in the Groups field and press tab to apply it.

Versions affected:

Cloudera Data Visualization 7.1.8- 7.2.8

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-2754

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.1.7

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

Error when creating dataset on connection with result caching

Creating a new dataset on a data connection with connection-level result caching enabled can return an “error: DataSet matching query does not exist” error.

Workaround:

To resolve this issue, disable Result Cache on the affected data connection.

Versions affected:

Cloudera Data Visualization 7.1.7 - 7.1.8

Fixed in version:

Cloudera Data Visualization 7.1.9

Cloudera issue:

VIZ-2425

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.1.6

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

Issue with impersonation in CDW Impala connections

You may see the following error while creating or editing the data connection to a CDW Impala or Hive Virtual Warehouse and using the connection details auto-populated from the CDW Warehouse drop-down:

User <username> is not authorized to delegate to <username>.

This happens because the Impersonation and Trusted Impersonation options are both enabled.

Workaround:

If the message appears when creating a new data connection, try refreshing the page. This will reset the Impersonation and Trusted Impersonation options.

If the message appears on the Edit Data Connection modal, copy the hostname, username, HTTP path, and other values present on the Advanced tab, and manually edit the existing CDW Hive connections. A manual edit will typically not trigger this bug.

Versions affected:

Cloudera Data Visualization 7.1.1-7.1.5, CDV 7.1.6-b8

Fixed in version:

Cloudera Data Visualization 7.1.6.2-3

Cloudera issue:

VIZ-2269

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.1.5

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

Issue with impersonation in CDW Impala connections

You may see the following error while creating or editing the data connection to a CDW Impala or Hive Virtual Warehouse and using the connection details auto-populated from the CDW Warehouse drop-down:

User <username> is not authorized to delegate to <username>.

This happens because the Impersonation and Trusted Impersonation options are both enabled.

Workaround:

If the message appears when creating a new data connection, try refreshing the page. This will reset the Impersonation and Trusted Impersonation options.

If the message appears on the Edit Data Connection modal, copy the hostname, username, HTTP path, and other values present on the Advanced tab, and manually edit the existing CDW Hive connections. A manual edit will typically not trigger this bug.

Versions affected:

Cloudera Data Visualization 7.1.1-7.1.5, CDV 7.1.6-b8

Fixed in version:

Cloudera Data Visualization 7.1.6.2-3

Cloudera issue:

VIZ-2269

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.1.4

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Migration errors due to use of django model functions in data model

Upgrading from a version lower than 6.4.3 to 7.0.3 or above, there are errors in the migration due to use of django model functions in data model.

Workaround:

To resolve this issue, migrate to version 7.1.4+. If you are migrating to a version before 7.1.4, first migrate to 6.4.3.

Note: Applications migrated in this fashion might show incorrect dashboard associations on the dataset page.

Versions affected:

Cloudera Data Visualization 7.0.3 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b47

Cloudera issue:

VIZ-2047

Histogram visual fails with an execution error “e.map is not a function”

Workaround:

N/A

Versions affected:

Cloudera Data Visualization 6.2.6 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b74

Cloudera issue:

VIZ-2058

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

Snowflake connection shows query errors in CDSW 1.10.3 or lower

Workaround:

To resolve this issue, upgrade CDSW to version 1.10.4 or higher.

Versions affected:

Cloudera Data Visualization 7.1.4 or lower using CDSW 1.10.3 or lower

Fixed in version:

Cloudera Data Visualization 7.1.5

Cloudera issue:

VIZ-2247

Issue with impersonation in CDW Impala connections

You may see the following error while creating or editing the data connection to a CDW Impala or Hive Virtual Warehouse and using the connection details auto-populated from the CDW Warehouse drop-down:

User <username> is not authorized to delegate to <username>.

This happens because the Impersonation and Trusted Impersonation options are both enabled.

Workaround:

If the message appears when creating a new data connection, try refreshing the page. This will reset the Impersonation and Trusted Impersonation options.

If the message appears on the Edit Data Connection modal, copy the hostname, username, HTTP path, and other values present on the Advanced tab, and manually edit the existing CDW Hive connections. A manual edit will typically not trigger this bug.

Versions affected:

Cloudera Data Visualization 7.1.1-7.1.5, CDV 7.1.6-b8

Fixed in version:

Cloudera Data Visualization 7.1.6.2-3

Cloudera issue:

VIZ-2269

Cloudera Data Visualization startup failure in Cloudera Machine Learning Private Cloud Data Service

Cloudera Data Visualization startup in Cloudera Machine Learning Private Cloud Data Service may fail. This issue has been present in the data service since version 1.3.4.

Workaround:

Set the environment variable CDV_MODE to cml before starting the application.

Versions affected:

Cloudera Data Visualization 7.1.4- 7.2.6

Fixed in version:

Cloudera Data Visualization 7.2.7

Cloudera issue:

VIZ-2829

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.1.3

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Edit and Copy to clipboard buttons do not work on the Manage URL Aliases page

Workaround:

There is no workaround for the 'Edit' function. For 'Copy to clipboard', open the URL alias in a new tab and copy it from the browser's address bar.

Versions affected:

Cloudera Data Visualization 7.1.3

Fixed in version:

Cloudera Data Visualization 7.1.4

Cloudera issue:

N/A

Change password screen shown on page refresh for autologin user

If an unauthenticated/autologin user accesses a public workspace dashboard on an Incognito tab and then refreshes or tries to reload the page, it can unexpectedly trigger the display of the Change Password modal.

Workaround:

To resolve this issue, change the autologin user's password once using the Change Password modal. After changing the password, the issue should no longer occur and the autologin functionality will work as expected.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2122

Selecting NULL values can cause ParseException syntax error from Cloudera Data Visualization 7.1.3

After upgrading to Cloudera Data Visualization 7.1.3 or later, selecting "NULL" values in filters may lead to a ParseException syntax error.

Workaround:

To resolve this issue, deselect the “NULL” value in the picklist filter.

Versions affected:

Cloudera Data Visualization 7.1.3 - 7.2.9

Fixed in version:

N/A

Cloudera issue:

VIZ-2224

Issue with impersonation in CDW Impala connections

You may see the following error while creating or editing the data connection to a CDW Impala or Hive Virtual Warehouse and using the connection details auto-populated from the CDW Warehouse drop-down:

User <username> is not authorized to delegate to <username>.

This happens because the Impersonation and Trusted Impersonation options are both enabled.

Workaround:

If the message appears when creating a new data connection, try refreshing the page. This will reset the Impersonation and Trusted Impersonation options.

If the message appears on the Edit Data Connection modal, copy the hostname, username, HTTP path, and other values present on the Advanced tab, and manually edit the existing CDW Hive connections. A manual edit will typically not trigger this bug.

Versions affected:

Cloudera Data Visualization 7.1.1-7.1.5, CDV 7.1.6-b8

Fixed in version:

Cloudera Data Visualization 7.1.6.2-3

Cloudera issue:

VIZ-2269

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.1.2

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Importing visual artifacts can fail in some systems, even with the “Disregard artifacts that exist in other workspaces” option selected

Manually edit the UUID of objects in the exported JSON file with a suffix like ‘-import’.

Workaround:

To resolve this issue, manually edit the data connection to use impersonation if trusted impersonation is enabled.

Versions affected:

Cloudera Data Visualization 7.0.5 - 7.1.2

Fixed in version:

Cloudera Data Visualization 7.1.3

Cloudera issue:

VIZ-2050

Histogram visual fails with an execution error “e.map is not a function”

Workaround:

N/A

Versions affected:

Cloudera Data Visualization 6.2.6 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b74

Cloudera issue:

VIZ-2058

Trusted impersonation does not work for Impala in Cloudera Datawarehouse on CDP Private Cloud

Impersonation should be the default setting used for these connections.

Workaround:

To resolve this issue, manually edit the data connection to use impersonation if trusted impersonation is enabled.

Versions affected:

Cloudera Data Visualization 7.1.1 - 7.1.2

Fixed in version:

Cloudera Data Visualization 7.1.3

Cloudera issue:

VIZ-2088

Issue with impersonation in CDW Impala connections

You may see the following error while creating or editing the data connection to a CDW Impala or Hive Virtual Warehouse and using the connection details auto-populated from the CDW Warehouse drop-down:

User <username> is not authorized to delegate to <username>.

This happens because the Impersonation and Trusted Impersonation options are both enabled.

Workaround:

If the message appears when creating a new data connection, try refreshing the page. This will reset the Impersonation and Trusted Impersonation options.

If the message appears on the Edit Data Connection modal, copy the hostname, username, HTTP path, and other values present on the Advanced tab, and manually edit the existing CDW Hive connections. A manual edit will typically not trigger this bug.

Versions affected:

Cloudera Data Visualization 7.1.1-7.1.5, CDV 7.1.6-b8

Fixed in version:

Cloudera Data Visualization 7.1.6.2-3

Cloudera issue:

VIZ-2269

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.1.1

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Filters breaking on spaces and special characters

Filters with column names that contain spaces or special characters return a syntax error when added to the dashboard or to a visual's shelves.

Workaround:

To resolve this issue, bracket the column names affected in the filter's expression editor.

Versions affected:

Cloudera Data Visualization 7.0.5 - 7.1.1

Fixed in version:

Cloudera Data Visualization 7.1.0-b3

Cloudera issue:

VIZ-1876, VIZ-1896

Importing visual artifacts can fail in some systems, even with the "Disregard artifacts that exist in other workspaces" option selected

Manually edit the UUID of objects in the exported JSON file with a suffix like '-import'.

Workaround:

To resolve this issue, manually edit the data connection to use impersonation if trusted impersonation is enabled.

Versions affected:

Cloudera Data Visualization 7.0.5 - 7.1.2

Fixed in version:

Cloudera Data Visualization 7.1.3

Cloudera issue:

VIZ-2050

Histogram visual fails with an execution error "e.map is not a function"

Workaround:

N/A

Versions affected:

Cloudera Data Visualization 6.2.6 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b74

Cloudera issue:

VIZ-2058

Trusted impersonation does not work for Impala in Cloudera Datawarehouse on CDP Private Cloud

Impersonation should be the default setting used for these connections.

Workaround:

To resolve this issue, manually edit the data connection to use impersonation if trusted impersonation is enabled.

Versions affected:

Cloudera Data Visualization 7.1.1 - 7.1.2

Fixed in version:

Cloudera Data Visualization 7.1.3

Cloudera issue:

VIZ-2088

Issue with impersonation in CDW Impala connections

You may see the following error while creating or editing the data connection to a CDW Impala or Hive Virtual Warehouse and using the connection details auto-populated from the CDW Warehouse drop-down:

User <username> is not authorized to delegate to <username>.

This happens because the Impersonation and Trusted Impersonation options are both enabled.

Workaround:

If the message appears when creating a new data connection, try refreshing the page. This will reset the Impersonation and Trusted Impersonation options.

If the message appears on the Edit Data Connection modal, copy the hostname, username, HTTP path, and other values present on the Advanced tab, and manually edit the existing CDW Hive connections. A manual edit will typically not trigger this bug.

Versions affected:

Cloudera Data Visualization 7.1.1-7.1.5, CDV 7.1.6-b8

Fixed in version:

Cloudera Data Visualization 7.1.6.2-3

Cloudera issue:

VIZ-2269

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.1.0

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

When downloading CSV and Excel files using legacy Arcengine, Impala, or Hive connections only headers are visible

Workaround:

To resolve this issue, switch to non-legacy connections.

Versions affected:

CDV 7.0.3 - 7.1.0

Fixed in version:

Cloudera Data Visualization 7.1.1

Cloudera issue:

VIZ-1555

Filters breaking on spaces and special characters

Filters with column names that contain spaces or special characters return a syntax error when added to the dashboard or to a visual's shelves.

Workaround:

To resolve this issue, bracket the column names affected in the filter's expression editor.

Versions affected:

Cloudera Data Visualization 7.0.5 - 7.1.1

Fixed in version:

Cloudera Data Visualization 7.1.0-b3

Cloudera issue:

VIZ-1876, VIZ-1896

Histogram visual fails with an execution error "e.map is not a function"

Workaround:

N/A

Versions affected:

Cloudera Data Visualization 6.2.6 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b74

Cloudera issue:

VIZ-2058

Trusted impersonation does not work for Impala in Cloudera Datawarehouse on CDP Private Cloud

Impersonation should be the default setting used for these connections.

Workaround:

To resolve this issue, manually edit the data connection to use impersonation if trusted impersonation is enabled.

Versions affected:

Cloudera Data Visualization 7.1.1 - 7.1.2

Fixed in version:

Cloudera Data Visualization 7.1.3

Cloudera issue:

VIZ-2088

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.0.5

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Console errors when queries are automatically cancelled

Cancelled queries may result in intermittent dashboard errors or console errors unexpectedly.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.5

Fixed in version:

Cloudera Data Visualization 7.1.0

Cloudera issue:

VIZ-1457

When downloading CSV and Excel files using legacy Arcengine, Impala, or Hive connections only headers are visible

Workaround:

To resolve this issue, switch to non-legacy connections.

Versions affected:

CDV 7.0.3 - 7.1.0

Fixed in version:

Cloudera Data Visualization 7.1.1

Cloudera issue:

VIZ-1555

"Refresh" button not visible in Connection Explorer for Arcadia connections

The Refresh button is not available in the Connection Explorer. The visibility of this button is meant to be controlled by the Hide "Refresh" button from Connection Explorer in Site Settings, but changing the setting has no impact for the arcengine connection.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.4 - 7.0.5

Fixed in version:

Cloudera Data Visualization 7.1.0

Cloudera issue:

VIZ-1799

Filters breaking on spaces and special characters

Filters with column names that contain spaces or special characters return a syntax error when added to the dashboard or to a visual's shelves.

Workaround:

To resolve this issue, bracket the column names affected in the filter's expression editor.

Versions affected:

Cloudera Data Visualization 7.0.5 - 7.1.1

Fixed in version:

Cloudera Data Visualization 7.1.0-b3

Cloudera issue:

VIZ-1876, VIZ-1896

Histogram visual fails with an execution error "e.map is not a function"

Workaround:

N/A

Versions affected:

Cloudera Data Visualization 6.2.6 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b74

Cloudera issue:

VIZ-2058

Trusted impersonation does not work for Impala in Cloudera Datawarehouse on CDP Private Cloud

Impersonation should be the default setting used for these connections.

Workaround:

To resolve this issue, manually edit the data connection to use impersonation if trusted impersonation is enabled.

Versions affected:

Cloudera Data Visualization 7.1.1 - 7.1.2

Fixed in version:

Cloudera Data Visualization 7.1.3

Cloudera issue:

VIZ-2088

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.0.4

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Disable CDW warehouse input if there are no options

In CDW, if there are no associated data warehouses in the same workspace, the dropdown appears, but remains empty. It will be disabled in future.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version

7.0.5

Cloudera issue

VIZ-1454

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Console errors when queries are automatically cancelled

Cancelled queries may result in intermittent dashboard errors or console errors unexpectedly.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.5

Fixed in version:

Cloudera Data Visualization 7.1.0

Cloudera issue:

VIZ-1457

When downloading CSV and Excel files using legacy Arcengine, Impala, or Hive connections only headers are visible

Workaround:

To resolve this issue, switch to non-legacy connections.

Versions affected:

CDV 7.0.3 - 7.1.0

Fixed in version:

Cloudera Data Visualization 7.1.1

Cloudera issue:

VIZ-1555

Line labels are enabled with display formatting for combined visuals

Workaround:

To resolve this issue, uncheck Label measure values next to points in combined visuals with line graphs.

Versions affected:

Cloudera Data Visualization 7.0.3 - 7.0.4

Fixed in version:

Cloudera Data Visualization 7.0.4-b41

Cloudera issue:

VIZ-1702

Switching dashboard sheets in view mode results in JS error

Workaround:

To resolve this issue, edit a visual in a dashboard.

Versions affected:

Cloudera Data Visualization 7.0.3 - 7.0.4

Fixed in version:

7.0.4-b41

Cloudera issue:

VIZ-1733

Trigger Jobs fail with "name 'username' is not defined" NameError

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version:

7.0.4-b41

Cloudera issue:

VIZ-1742

Emailing jobs fail when SSL and TLS errors are disabled

Workaround:

To resolve this issue, switch to an email server that supports TLS or SSL.

Versions affected:

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version:

7.0.4-b41

Cloudera issue:

VIZ-1743

"Refresh" button not visible in Connection Explorer for Arcadia connections

The Refresh button is not available in the Connection Explorer. The visibility of this button is meant to be controlled by the Hide "Refresh" button from Connection Explorer in Site Settings, but changing the setting has no impact for the arcengine connection.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.4 - 7.0.5

Fixed in version:

Cloudera Data Visualization 7.1.0

Cloudera issue:

VIZ-1799

Histogram visual fails with an execution error “e.map is not a function”

Workaround:

N/A

Versions affected:

Cloudera Data Visualization 6.2.6 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b74

Cloudera issue:

VIZ-2058

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.0.3

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Disable CDW warehouse input if there are no options

In CDW, if there are no associated data warehouses in the same workspace, the dropdown appears, but remains empty. It will be disabled in future.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version

7.0.5

Cloudera issue

VIZ-1454

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Console errors when queries are automatically cancelled

Cancelled queries may result in intermittent dashboard errors or console errors unexpectedly.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.5

Fixed in version:

Cloudera Data Visualization 7.1.0

Cloudera issue:

VIZ-1457

When downloading CSV and Excel files using legacy Arcengine, Impala, or Hive connections only headers are visible

Workaround:

To resolve this issue, switch to non-legacy connections.

Versions affected:

CDV 7.0.3 - 7.1.0

Fixed in version:

Cloudera Data Visualization 7.1.1

Cloudera issue:

VIZ-1555

Custom background color assignment on dimensions or measures not possible

Cannot fix colors for specific dimension values or numeric ranges when adding background colors to tables or crosstabs.

Workaround:

To resolve this issue, add such basic custom styles into the application.

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.3

Fixed in version:

7.0.4

Cloudera issue:

VIZ-1580 & VIZ-1060

Trigger Jobs fail with "name 'username' is not defined" NameError

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version:

7.0.4-b41

Cloudera issue:

VIZ-1742

Line labels are enabled with display formatting for combined visuals

Workaround:

To resolve this issue, uncheck Label measure values next to points in combined visuals with line graphs.

Versions affected:

Cloudera Data Visualization 7.0.3 - 7.0.4

Fixed in version:

Cloudera Data Visualization 7.0.4-b41

Cloudera issue:

VIZ-1702

Switching dashboard sheets in view mode results in JS error

Workaround:

To resolve this issue, edit a visual in a dashboard.

Versions affected:

Cloudera Data Visualization 7.0.3 - 7.0.4

Fixed in version:

7.0.4-b41

Cloudera issue:

VIZ-1733

Emailing jobs fail when SSL and TLS errors are disabled

Workaround:

To resolve this issue, switch to an email server that supports TLS or SSL.

Versions affected:

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version:

7.0.4-b41

Cloudera issue:

VIZ-1743

Histogram visual fails with an execution error “e.map is not a function”

Workaround:

N/A

Versions affected:

Cloudera Data Visualization 6.2.6 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b74

Cloudera issue:

VIZ-2058

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.0.2

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

Picklist in complex data types Show Selected unselects previous selections

Show selected in a filter list selection may not show all selections depending on the data type. A fix is expected in a future release.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-456

Inconsistent support for non-roman or character-based alphabets across visual types

Non-romanized or character-based alphabets are not consistently supported across various visual types. If not supported, the affected visual is not properly rendered in PDF/PNG downloads. CSV/Excel downloads and general dashboard viewing is available for all languages.

The following languages are not supported when using a table visual.

In CML:	In CDW:
• Amharic (Ethiopia) • Arabic (multiple countries) • Bengali (Bangladesh) • Berber (multiple countries) • Burmese (Myanmar) • Dari Pashto (Afghanistan) • Dzongkha (Bhutan) • Fijian (Fiji) • Georgian • Hebrew (Israel) • Khmer (Cambodia) • Lao • Malay (Brunei) • Many Indian languages • Nepalese (Nepal) • Persian (Iran) • Tamil (Malaysia, India, Singapore) • Thai (Thailand) • Urdu (Pakistan)	• Amharic (Ethiopia) • Bengali (Bangladesh) • Burmese (Myanmar) • Dzongkha (Bhutan) • Fijian (Fiji) • Khmer (Cambodia) • Many Indian languages • Nepalese (Nepal) • Tamil (Malaysia, India, Singapore) • Thai (Thailand)

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-984

Saved value does not show properly when arcapi is present

After the fix included for VIZ-1070, a known issue is still present when using arcapi.sendParameters with app-scope filters.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.2 - 7.2.9

Fixed in version

N/A

Cloudera issue

VIZ-1181

Filtering for null values does not work

It is possible to create filters for fields whose set contains NULL values. However, it is not possible to display only the rows that have NULL for the filtered field value.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.1 - 7.0.2

Fixed in version

7.0.3

Cloudera issue

VIZ-1452

Disable CDW warehouse input if there are no options

In CDW, if there are no associated data warehouses in the same workspace, the dropdown appears, but remains empty. It will be disabled in future.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version

7.0.5

Cloudera issue

VIZ-1454

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Console errors when queries are automatically cancelled

Cancelled queries may result in intermittent dashboard errors or console errors unexpectedly.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.5

Fixed in version:

Cloudera Data Visualization 7.1.0

Cloudera issue:

VIZ-1457

Issue with old currency display format handling

Workaround

None

Versions affected:

Cloudera Data Visualization 7.0.2

Fixed in version:

7.0.3

Cloudera issue:

VIZ-1561

Unable to clone visuals in some cases

Workaround

None

Versions affected:

Cloudera Data Visualization 7.0.2

Fixed in version:

7.0.3

Cloudera issue:

VIZ-1562

Impala connection failing

While trying to connect Cloudera Data Visualization on CDSW, the connection test returns an error. This is due to a known THRIFT bug.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.2

Fixed in version:

7.0.2-b35

Cloudera issue:

VIZ-1568

Custom background color assignment on dimensions or measures not possible

Cannot fix colors for specific dimension values or numeric ranges when adding background colors to tables or crosstabs.

Workaround:

To resolve this issue, add such basic custom styles into the application.

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.3

Fixed in version:

7.0.4

Cloudera issue:

VIZ-1580 & VIZ-1060

If the user has not dismissed the “What’s New” modal using the “Got It” button, it can appear in emailed dashboards

Workaround:

To resolve this issue, click Got It in the What’s New modal.

Versions affected:

Cloudera Data Visualization 7.0.2

Fixed in version:

Cloudera Data Visualization 7.0.3

Cloudera issue:

VIZ-1605

Trigger Jobs fail with "name 'username' is not defined" NameError

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version:

7.0.4-b41

Cloudera issue:

VIZ-1742

Emailing jobs fail when SSL and TLS errors are disabled

Workaround:

To resolve this issue, switch to an email server that supports TLS or SSL.

Versions affected:

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version:

7.0.4-b41

Cloudera issue:

VIZ-1743

Histogram visual fails with an execution error “e.map is not a function”

Workaround:

N/A

Versions affected:

Cloudera Data Visualization 6.2.6 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b74

Cloudera issue:

VIZ-2058

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.0.1

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

CANCEL button on Edit User or New User modal needs to be clicked twice

Clicking the CANCEL button in the Edit User or New User modals may display an unexpected validation error instead of closing the modal.

Workaround:

The error can be bypassed by clicking CANCEL a second time.

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.1

Fixed in version:

Cloudera Data Visualization 7.0.2

Cloudera issue:

VIZ-829

Filtering for null values does not work

It is possible to create filters for fields whose set contains NULL values. However, it is not possible to display only the rows that have NULL for the filtered field value.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.1 - 7.0.2

Fixed in version

7.0.3

Cloudera issue

VIZ-1452

Disable CDW warehouse input if there are no options

In CDW, if there are no associated data warehouses in the same workspace, the dropdown appears, but remains empty. It will be disabled in future.

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version

7.0.5

Cloudera issue

VIZ-1454

Upgrade snowflake-connector-python when the corresponding issue is fixed

For some characters, snowflake queries return "Failed to convert current row, cause: 'utf-8' codec can't decode byte" error (SNOW-182387: UTF-8 decoding exception when querying table in SNOWFLAKE_SAMPLE_DATA).

Workaround

None

Versions affected

Cloudera Data Visualization 7.0.0 - 7.2.9

Fixed in version

Cloudera Data Visualization 7.2.2

Cloudera issue

VIZ-1455

Console errors when queries are automatically cancelled

Cancelled queries may result in intermittent dashboard errors or console errors unexpectedly.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.5

Fixed in version:

Cloudera Data Visualization 7.1.0

Cloudera issue:

VIZ-1457

Email fails with error

TLS and SSL can both be enabled even if they are mutually exclusive. This results in an error and sending the email fails.

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.1

Fixed in version:

Cloudera Data Visualization 7.0.2

Cloudera issue:

VIZ-1486

Custom background color assignment on dimensions or measures not possible

Cannot fix colors for specific dimension values or numeric ranges when adding background colors to tables or crosstabs.

Workaround:

To resolve this issue, add such basic custom styles into the application.

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.3

Fixed in version:

7.0.4

Cloudera issue:

VIZ-1580 & VIZ-1060

Trigger Jobs fail with "name 'username' is not defined" NameError

Workaround:

None

Versions affected:

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version:

7.0.4-b41

Cloudera issue:

VIZ-1742

Emailing jobs fail when SSL and TLS errors are disabled

Workaround:

To resolve this issue, switch to an email server that supports TLS or SSL.

Versions affected:

Cloudera Data Visualization 7.0.1 - 7.0.4

Fixed in version:

7.0.4-b41

Cloudera issue:

VIZ-1743

Histogram visual fails with an execution error “e.map is not a function”

Workaround:

N/A

Versions affected:

Cloudera Data Visualization 6.2.6 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b74

Cloudera issue:

VIZ-2058

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Known issues and limitations in Data Visualization 7.0.0

Learn about the known issues and limitations that you might experience while using Cloudera Data Visualization. You can find information about the areas of impact, affected product versions, and possible workarounds.

CANCEL button on Edit User or New User modal needs to be clicked twice

Clicking the CANCEL button in the Edit User or New User modals may display an unexpected validation error instead of closing the modal.

Workaround:

The error can be bypassed by clicking CANCEL a second time.

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.1

Fixed in version:

Cloudera Data Visualization 7.0.2

Cloudera issue:

VIZ-829

Excel or streaming CSV download error for Boolean data types

When the Number of rows downloaded during CSV/Excel export and Detailed Data setting is applied for Excel downloads (all connections) or CSV downloads (Impala/Hive connections), Boolean data types will cause download error.

Workaround:

To resolve this issue, cast the Boolean data as a string or number.

Versions affected:

Cloudera Data Visualization 7.0.0

Fixed in version:

Cloudera Data Visualization 7.0.1

Cloudera issue:

VIZ-1380

Custom background color assignment on dimensions or measures not possible

Cannot fix colors for specific dimension values or numeric ranges when adding background colors to tables or crosstabs.

Workaround:

To resolve this issue, add such basic custom styles into the application.

Versions affected:

Cloudera Data Visualization 7.0.0 - 7.0.3

Fixed in version:

7.0.4

Cloudera issue:

VIZ-1580 & VIZ-1060

Histogram visual fails with an execution error “e.map is not a function”

Workaround:

N/A

Versions affected:

Cloudera Data Visualization 6.2.6 - 7.1.4

Fixed in version:

Cloudera Data Visualization 7.1.4-b74

Cloudera issue:

VIZ-2058

Partitioned Cloudera Impala on premise extract tables cannot be updated

Tables in Cloudera Impala that are both partitioned and set as extract targets cannot be updated directly. This prevents operations such as dropping or adding new partitions.

Workaround:

Move the data to a new table that supports altering commands. Then replace the original extract table by renaming the newly created table. Optionally, you can create a backup of the original table before replacing it.

A sample SQL workflow is provided below. Customize the components marked in blue, while the remaining parts of the query are standard.

1. Create a temporary table (database_name.new_table_name) that supports altering.

```
create table DATABASE_NAME.CDV_TABLE_NAME_NEW
 (TABLE_DEFINITION) partitioned by (PARTITION_FIELD
 STRING) tblproperties('transactional'='false');
```

2. Copy existing data from the original extract target.

```
insert into DATABASE_NAME.CDV_TABLE_NAME_NEW
 partition(PARTITION_FIELD) select * f
rom DATABASE_NAME.CDV_TABLE_NAME;
```

3. [Optional] Save the original extract target as a backup.

```
alter table DATABASE_NAME.CDV_TABLE_NAME rename
to DATABASE_NAME.CDV_TABLE_NAME_BAK;
```

4. Make the temporary table the current extract target.

```
alter table DATABASE_NAME.CDV_TABLE_NAME_NEW rename
to DATABASE_NAME.CDV_TABLE_NAME;
```

Versions affected:

Cloudera Data Visualization 7.2.8 and all earlier versions

Fixed in version:

Cloudera Data Visualization 7.2.9

Cloudera issue:

VIZ-3042

Cloudera Data Visualization 7.2.9

Configuring Restricting Row-Level Access

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Setting segments 4

Creating users and user groups 4

Setting roles 5

Setting filter associations 7

Verifying access restriction on a visual 11

Setting segments

About this task

To specify segments in the dataset, you can save information from the Filters shelf of a visual as a segment. Follow the steps outlined in our article *Creating Segments from Filter Definitions*.

Alternatively, you can specify the filters in the Dataset Segments interface, as described in *Creating Segments*.

Procedure

Define the following segments:

Table 1: Definitions of Segments on the Dataset US County Population

Group	Name	Filters
Geographic Regions	California	([State] in ('California'))
Geographic Regions	Mountain	([State] in ('Arizona', 'Nevada'))
Geographic Regions	North West	([State] in ('Washington', 'Oregon'))

After you define these segments in your dataset, your Dataset Segments interface should look like this:

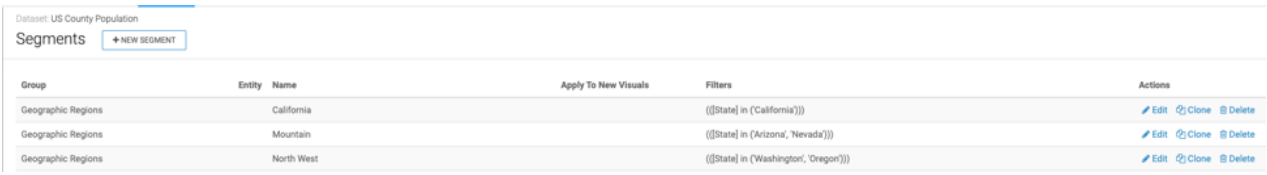

Related Information

Creating segments from filter definitions

Creating segments

Creating users and user groups

Procedure

1. Create a user group for all sales representatives. In the Manage Users & Groups interface, create a user group named Sales Reps.

   See, *Creating new user groups*.

   After you define this user group, your Manage Users & Groups interface should look like this:

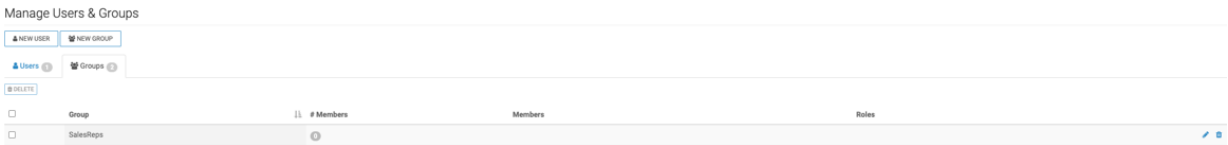

2. Define the following users, with normal user permissions, and assign them to the SalesReps user group:
   - AZrep1
   - CArep1
   - CArep2
   - NVrep1
   - ORrep1
   - WArep1

   See, *Creating new user accounts* and *Adding users to user groups*.

   After you create these users, your Manage Users & Groups interface should look like this:

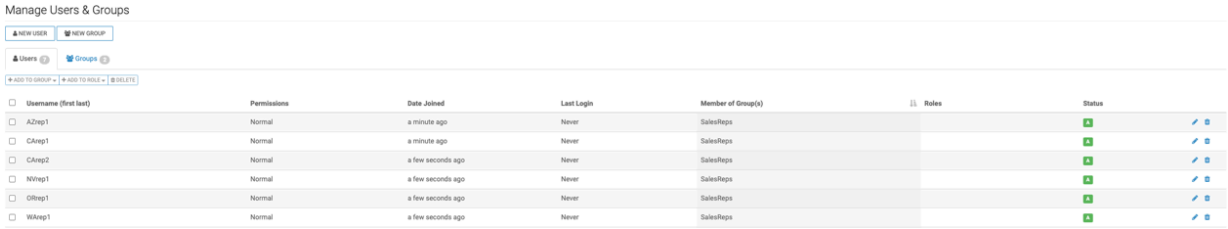

Related Information

Creating new user groups

Creating new user accounts

Adding users to user groups

Setting roles

About this task

To enforce row-level access restrictions, it is important to define roles with appropriate privileges and assign them to individual users or user groups.

Procedure

1. Follow the instructions covered in *Creating segments from filter definitions* and *Editing role assignments*.

   For more information about permission sets, see *Role privileges*.

2. Define a role with the following specifications:

   Name

   See_Customer_Data

   Description

   Sales Reps can see customer data

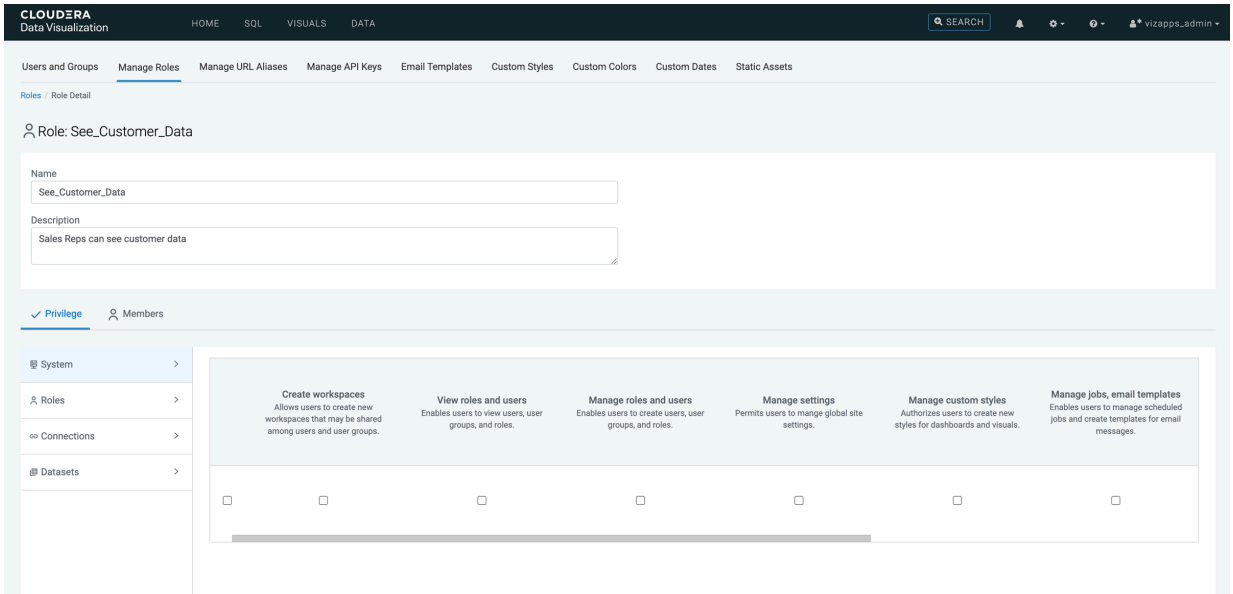

3. Configure the roles's privileges on the Privileges tab.

   System Privileges

   None

   Dataset Privileges

   Customer Value Analysis dataset, on the sample connection (this may vary depending on where you loaded the data).

   Permission

   View visuals and apps

4. Add the SalesReps group to the role on the Members tab.

   Users

   None

   Groups

   SalesReps

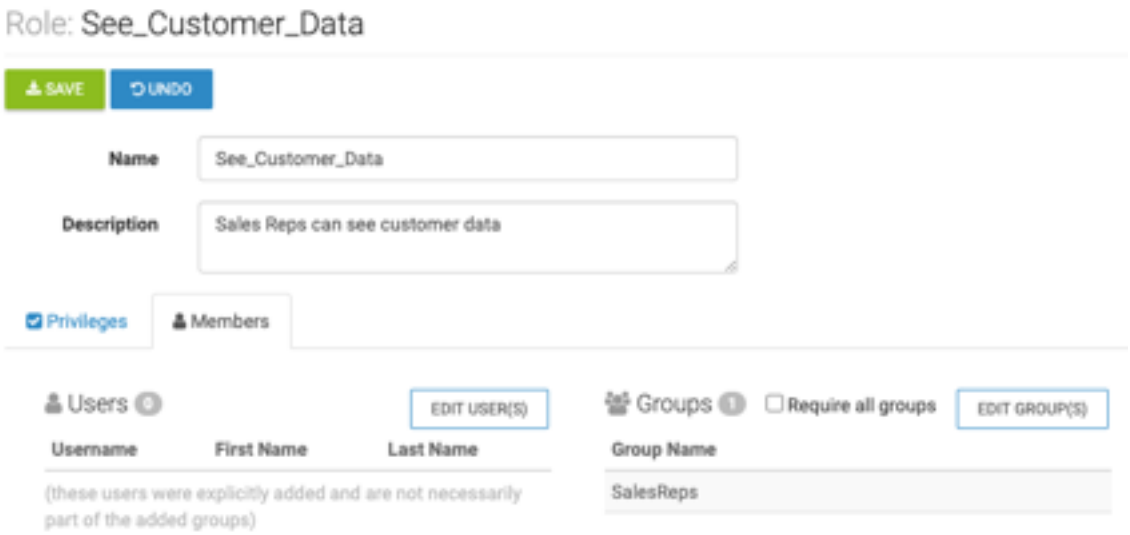

Related Information

Editing role assignments
Role privileges
Creating segments from filter definitions

Setting filter associations

About this task

The final task in restricting user or user group access is to specify filter associations.

Before you begin

Make sure that either of the listed requirements is fulfilled:

- You have Manage roles and users permission.
- You have View roles and users permission and MANAGE_DS_FA = True has been added to the Advanced Site Settings.

Procedure

1. On the main navigation bar, click DATA.

   The Data view appears, open on the Datasets tab.
2. In the left navigation menu, select your data connection.
3. In the Datasets area, click the dataset that you want to use.

   The Dataset Detail page appears.

4. In the left navigation, select Filter Associations and click NEW FILTER ASSOCIATION.

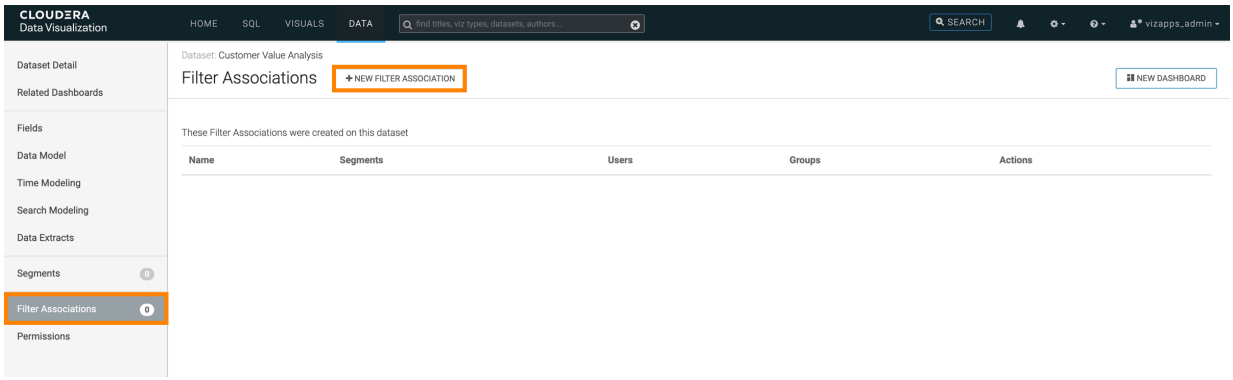

5. In the Add Filter Association modal window, perform the following actions:
   a. Under Filter Association Name, enter CA SalesReps.
   b. Under Applied Segments, select is California from Geographic Regions.
   c. On the Users tab, select users CArep1 and CArep2, and add them to the filter association.
   d. Click CREATE.

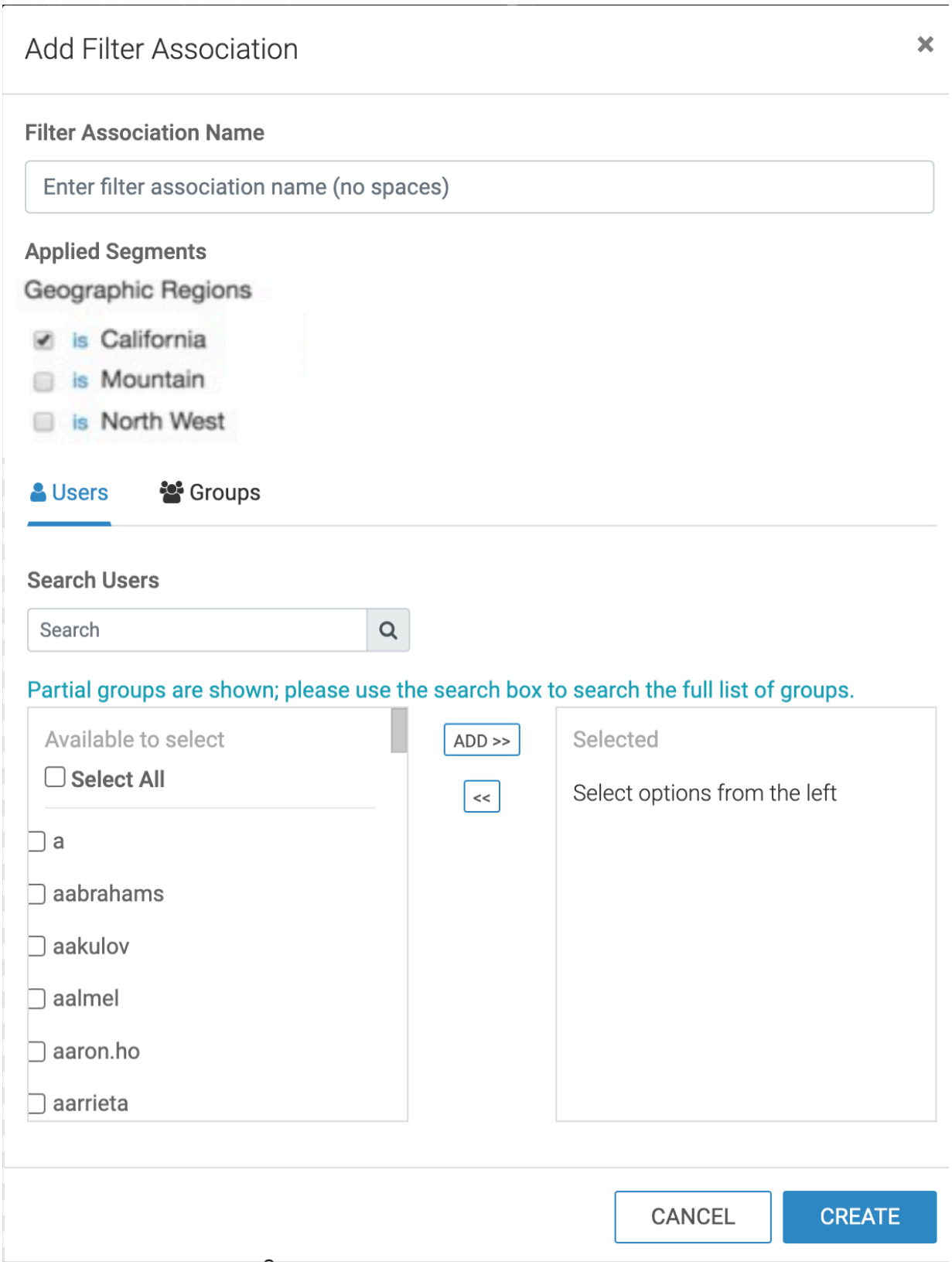

The new filter association now appears on the Filter Associations interface.

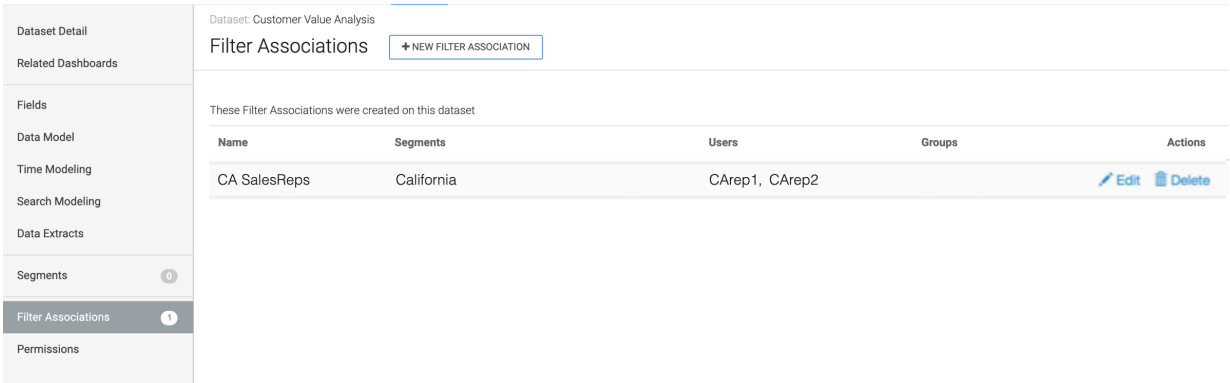

6. Repeat the steps above to add two more filter associations as described in the following table:

Table 2: Definitions of Segments on the Dataset Customer Value Analysis

Filter Association Name	Applied Segment	Users to Include
Mountain SalesReps	Mountain	NVrep1, AZrep1
North West SalesReps	North West	ORrep1, WArep1

After you define these filter associations in your dataset, your Filter Associations interface should look like this:

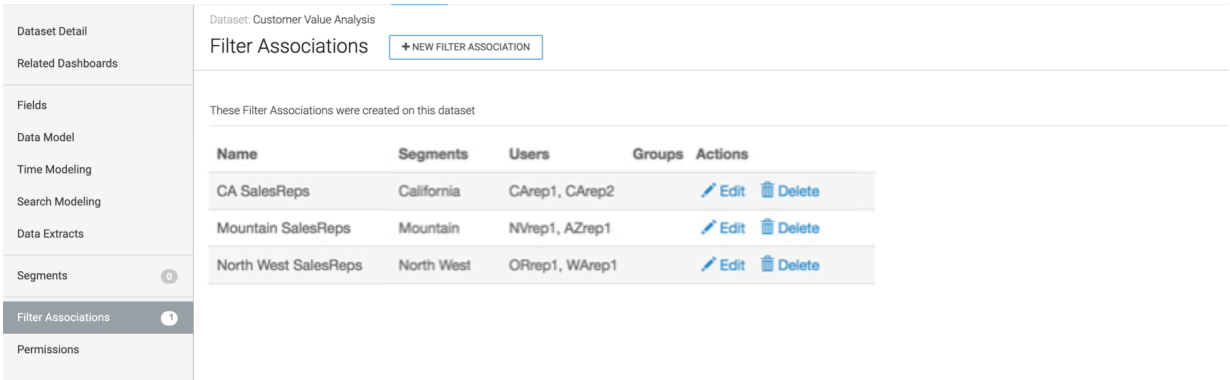

Verifying access restriction on a visual

About this task

To ensure that we specified the access restrictions correctly, let us view the same visual as an Administrative user, and as a user who is linked to each of the three filter associations declared in the previous section.

Procedure

1. Log in as a user with Administrative privileges, and navigate to the application. Notice that the entire dataset appears in the visual; for example, it shows 4,621 Four-Door Cars.

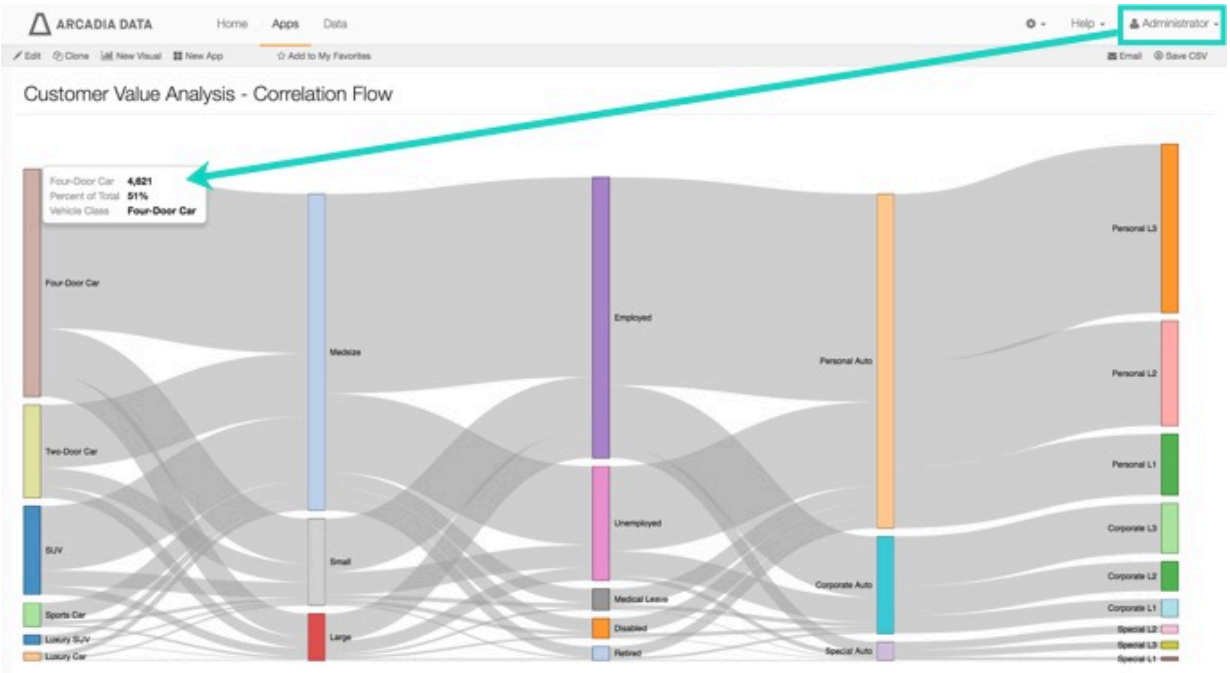

2. Log in as a user associated with the dataset through the CA SalesReps filter association. Such a user, in this case CArep1, can only see records available for the state of California. For example, when you navigate to the application, you can only see 1,593 records for Four-Door Cars.

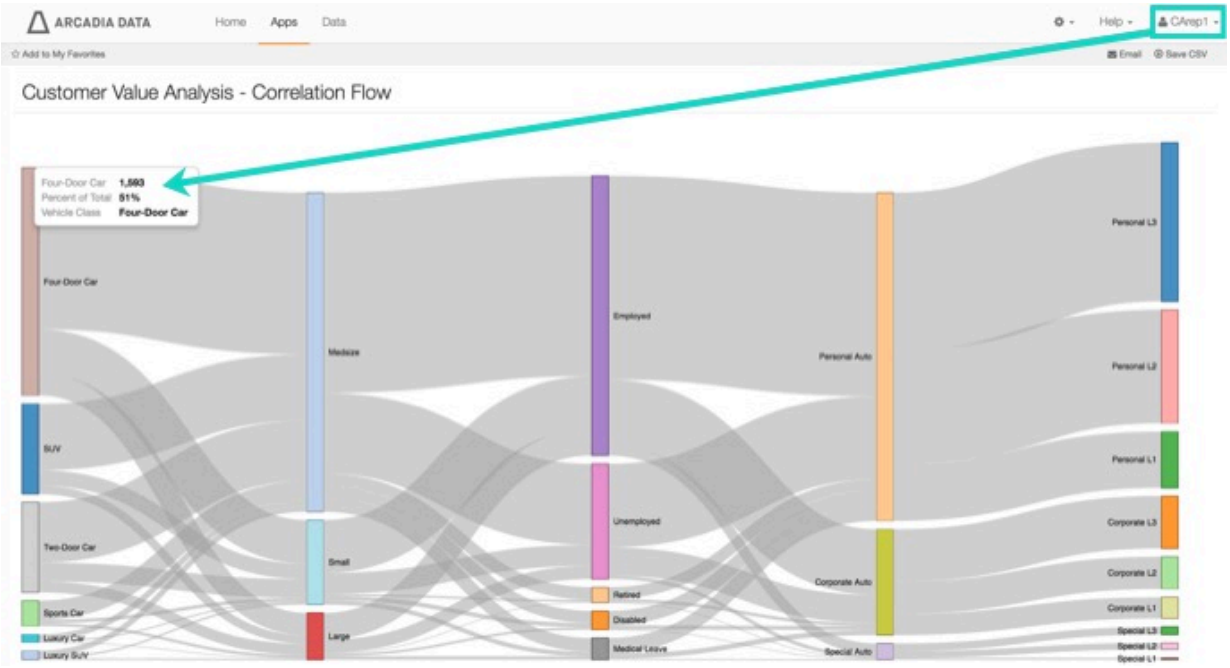

3. If you log in as AZrep1, a user associated with the dataset through the Mountain SalesReps filter association, you could only see records available for the states of Arizona and Nevada. For example, when you login as AZrep1 and navigate to the application, you can see only 1,325 records for Four-Door Cars.

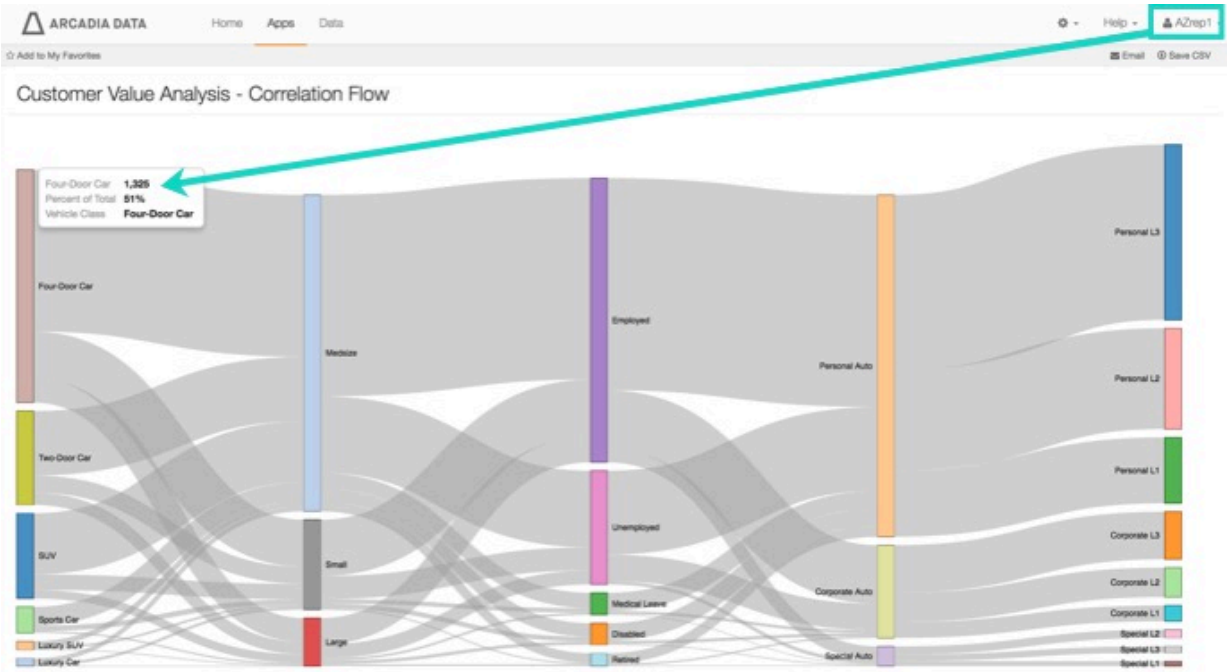

4. Finally, if you log in as WArep1, a user associated with the dataset through the North West SalesReps filter association, you would see records available for the states of Oregon and Washington, but none of the others. For example, when you login as WArep1 and navigate to the application, you see 1,703 records for Four-Door Cars.

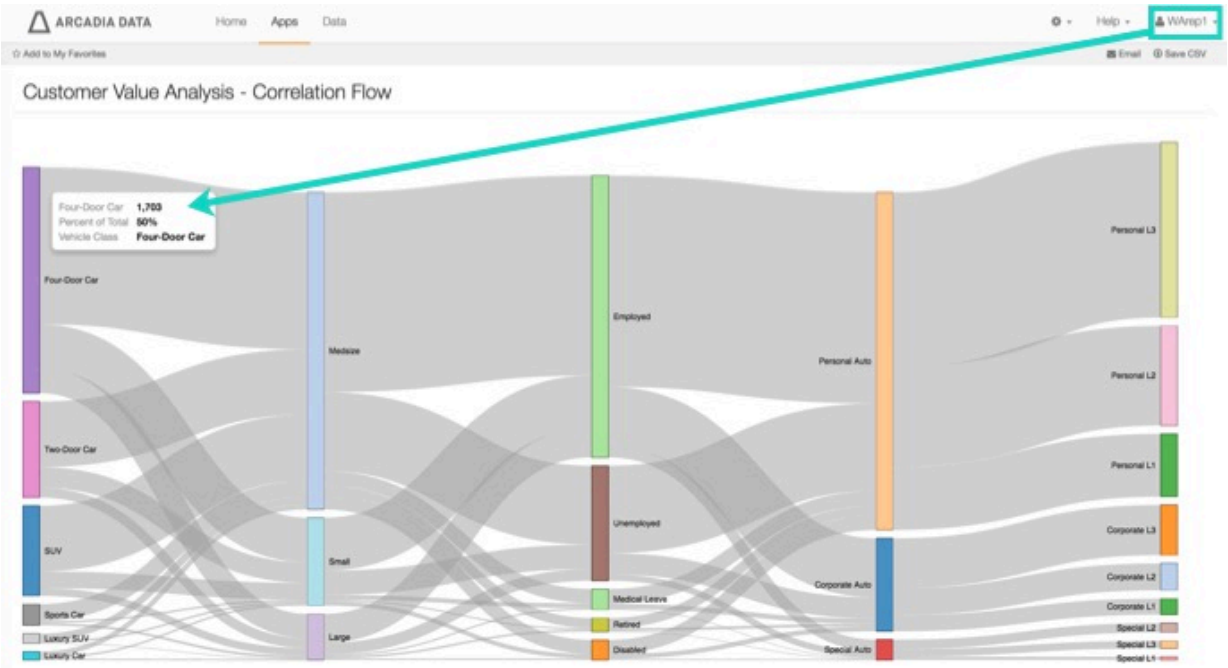

Cloudera Data Visualization 7.2.9

Security

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Security model 4

Role-based access control 4

Permissions 5

Role privileges 6

All groups requirement 8

RBAC setup for dataset publishing 9

Security model

Security in Cloudera Data Visualization means robust authentication and granular permission control. This comprehensive security approach involves various components, including role-based access control, which enables you to have precise control over data and feature access.

Administrative configuration

As an administrator, you have the authority to configure roles, privileges, and members, providing a foundation for robust security controls. For more information, see *Working with user roles*.

Role-based access control (RBAC)

You can manage privileges and members of a role witth the help of RBAC. Utilize RBAC to effectively manage privileges and members associated with specific roles, ensuring a tailored access control environment. For more information, see the *Role-based access control* documentation.

User roles for dataset control

You can leverage user roles to facilitate dataset creators and managers in sharing curated datasets within their organization. This includes the implementation of three levels of access control. For more information, see *Publishing datasets*.

Access control for defined data segments

You can restrict a user's access to defined data segments, ensuring a granular control mechanism. For more information, see *Setting segments*.

Advanced configuration options

You can further enhance security by configuring LDAP authentication. For more information, see *Enabling LDAP authentication*.

Related Information

Role-based access control
Working with user roles
Publishing datasets
Setting segments
Enabling LDAP authentication

Role-based access control

Role-Based Access Control (RBAC) is a functional system in Cloudera Data Visualization that helps to manage user access rights. By setting permissions and privileges, RBAC ensures that only authorized users can access specific functionalities, defined by their roles in the system.

Note: This feature is only available to users with administrative privileges.

RBAC provides administrators with precise control over data and feature access, based on different user roles. Roles come with distinct sets of rights, creating a structured and secure access system. For more information, see *Role privileges*.

Components of RBAC:

Permissions

Permissions regulate access to visuals, datasets, data connections, and system-level functions. They fall into four categories: system, role, connection, and dataset. For more information, see *Permissions*.

Privileges

Privileges are sets of permissions of a specific type, tied to associated components. For example, a privilege may contain the permission to View visuals and dashboards on the component specifier Connection default / All datasets.

Members

Members are a users and user groups assigned to a particular role.

Roles

Roles serve as collections of privileges, each with its associated members possessing these privileges.

Related Information

Role privileges

Permissions

Permissions define access to visuals, datasets, data connections, and system-level functions. There are four categories of permissions: system, role, connection, and dataset.

When defining privileges, the following default permissions exist at each level:

System-level permissions: site-level capabilities	Role-level permissions: defined separately for each role	Connection-level permissions: defined separately for each data connection	Dataset-level permissions: defined separately for each dataset
• Create workspaces • View roles and users • Manage roles and users • Manage site settings • Manage custom styles • Manage jobs, email templates • Manage data connections • View activity logs • Additional system privileges	• Grant manage dataset • Grant manage dashboards • Grant view dashboards	• Manage analytical views • Import data • Create datasets, explore tables	• Manage dataset • Manage dashboards • View dashboards

DATA interface interpretations

To connect the permissions to what options are available in the DATA interface of Cloudera Data Visualization, consider the following:

1. **System-level Manage data connections permission**

   Necessary to access the NEW CONNECTION button for creating new connections and the Pencil icon for editing existing connections.

2. **Dataset-level View visuals and dashboards permission**

   Required for a specific dataset to appear in the list of datasets for the selected connection.

3. **Connection-level Create datasets, explore tables permission**

   Necessary to access the:

   - NEW DATASET button over the list of datasets
   - Connection Explorer tab
   - Delete icon on the dataset row

4. **Connection-level Manage data connections and dataset-level Manage dataset permissions**

   Essential for the visibility of the Clear result cache option in the Supplemental menu.

5. **System-level Manage styles and settings, connection-level Create datasets, explore tables, and dataset-level Manage dataset and Manage visuals and dashboards permissions**

   May all be required for the visibility of the Import Visual Artifacts option in the Supplemental menu, depending on the type of import.

6. **Dataset-level Manage visuals and dashboards permission**

   Necessary to access the New Dashboard and New Visual icons on the specified dataset rows.

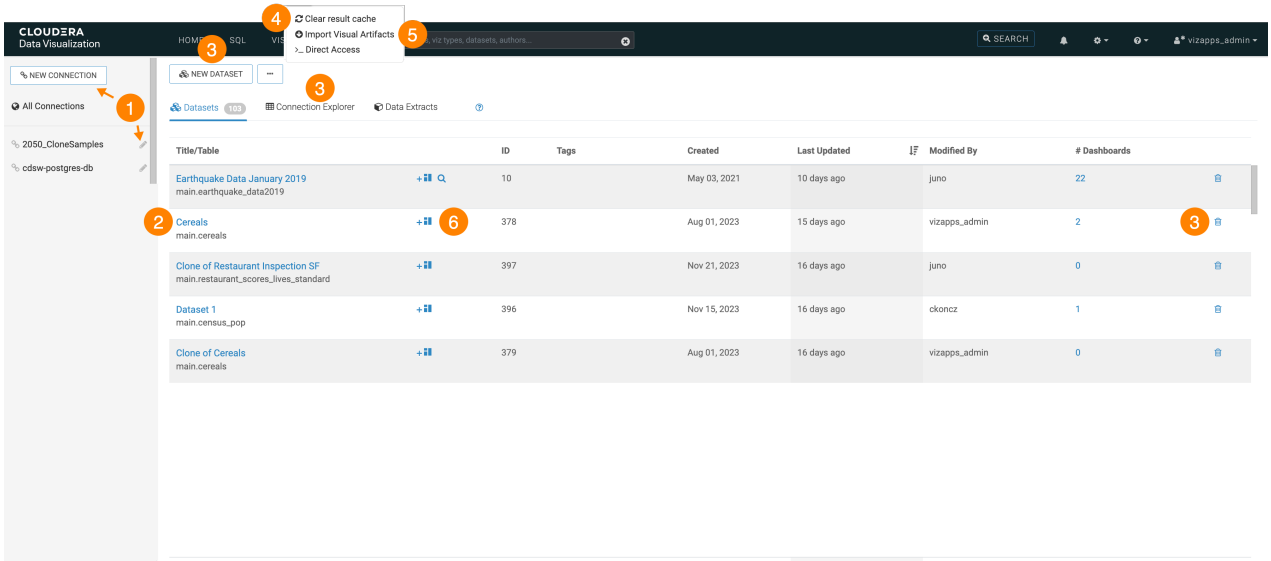

Role privileges

Privileges are sets of permissions of a particular type and the associated components on which the permissions are granted. Administrators can define role privileges at various levels within Cloudera Data Visualization. This allows precise control over system access for different roles.

Note: This feature is only available to users with administrative privileges.

Privileges for a role may be defined on one of the following levels:

- System privileges
- Role privileges
- Connection privileges
- Dataset privileges

The Role Detail interface shows a table matrix of privilege components and specific permissions granted to a role for each component. Each privilege type (system, roles, connections, or datasets) tab shows the relevant permissions.

System

You can see which system-level permissions are enabled.

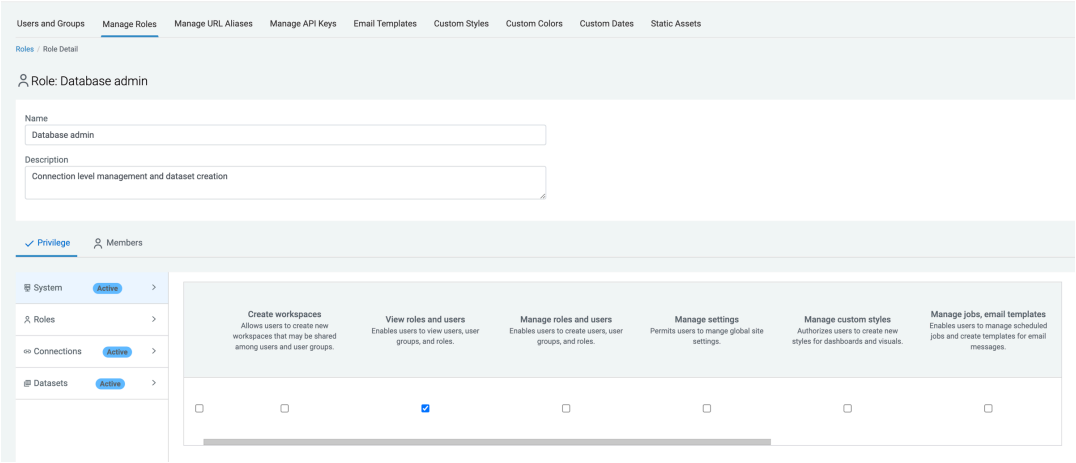

For more information on System privileges, see Create workspaces, View roles and users, Manage roles and users, Manage settings, Manage custom styles, Manage jobs, email templates, View activity logs, Manage data connections, and Additional system privilege.

Roles

You can see which role-based permissions are enabled.

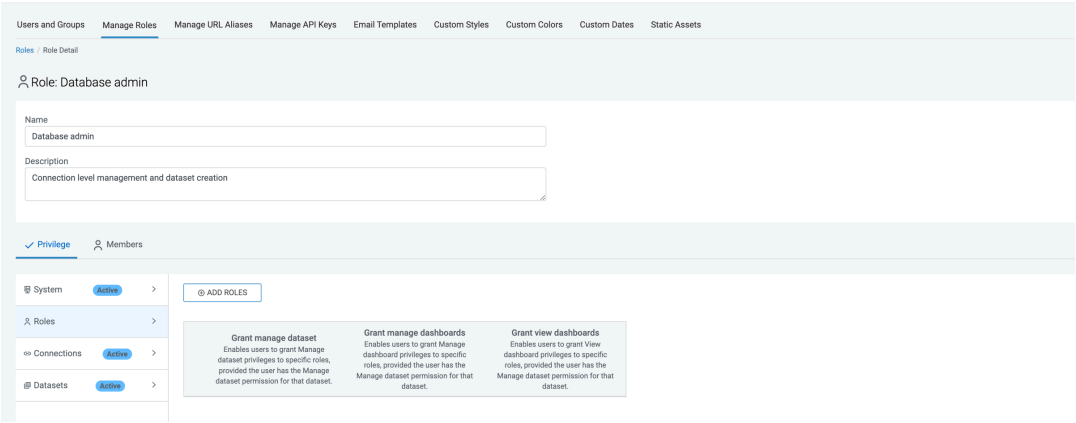

Actions available:

- ADD ROLES to add a new role privilege

For more information on Role privileges, see Grant manage dataset, Grant manage dashboards, and Grant view dashboards.

Connections

You can see what permissions are enabled for which connections.

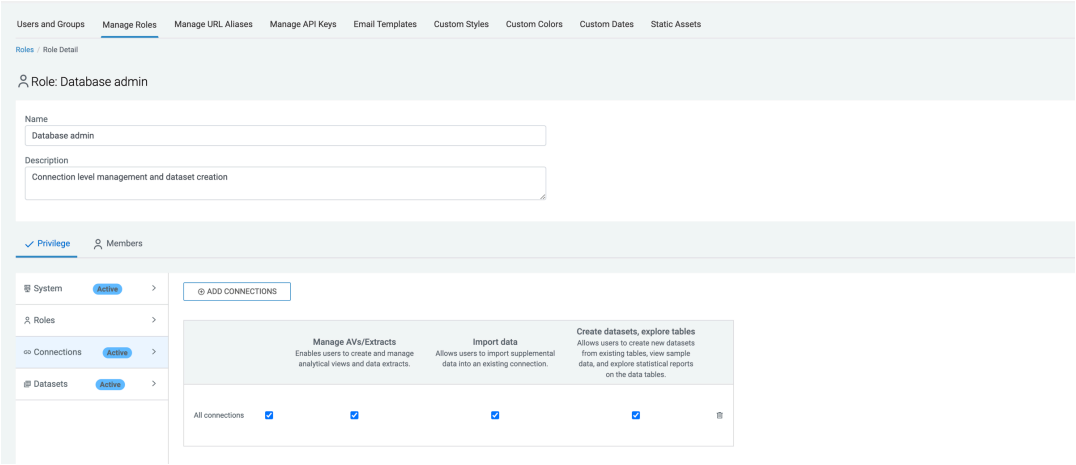

Actions available:

- ADD CONNECTIONS to add a new data connection role privilege
- Delete to remove a connection component

For more information on Connection privileges, see Manage AVs/Extracts, Import data, and Create datasets, explore tables.

Datasets

You can see what permissions are enabled for which datasets.

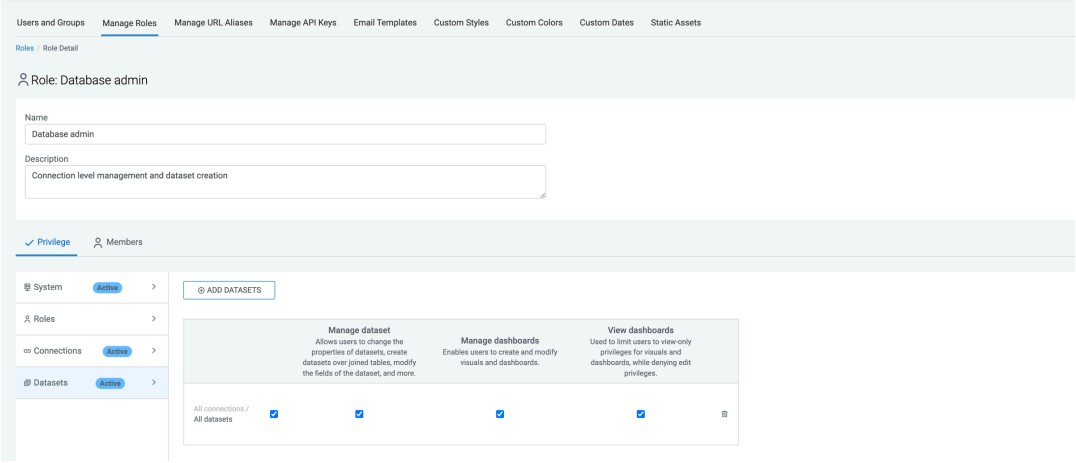

Actions available:

- ADD DATASETS to add a new dataset role privilege
- Delete to remove a dataset component

For more information on Dataset privileges, see Manage dataset, Manage dashboards, and View dashboards.

All groups requirement

The Require all groups option ensures that only members of ALL groups listed in the role membership fields have the role's defined access.

In this example, the role Administrators Only is shared by members of both Administrators and CDV users user groups. If you do not select the Require all groups option, all members of either group get the privileges of the role.

However, if you check the Require all groups options, only users who are members of BOTH Administrators and CDV users user groups get the privileges of the role.

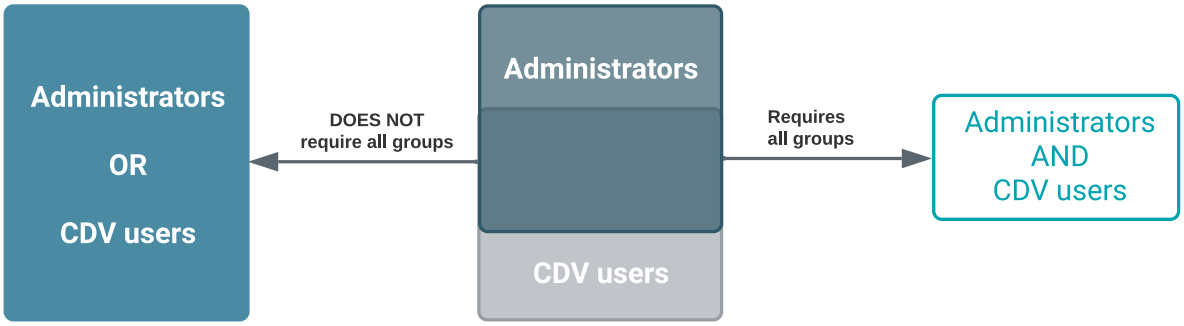

There are two other ways a user can be a member of the role, even when the Require all groups option is on:

- If the user is named specifically in the Users section of the membership page.
- For roles that are imported, if the Groups section is empty, and the user is a member of ANY imported group.

RBAC setup for dataset publishing

Role-Based Access Control (RBAC) empowers dataset creators to share or 'publish' their datasets securely by leveraging role-based privileges. The dataset creators can grant specific roles access using permissions like Grant manage dataset, Grant manage dashboards, and Grant view dashboards.

Note:

- Only users with Manage roles and users role privilege (typically system administrators) can set up roles and permissions and define users and user groups.

Consider the following practical scenario to demonstrate how dataset publishing works:

- **Teams**
  - Marketing
  - Sales
  - Operations
- **Access levels in each team**
  - Data admins
  - Analysts
  - Visual consumers

To set up the required permissions, roles, groups, and users, see the documentation listed in the Related information section.

After you have completed the RBAC setup steps, Data Admins can proceed to publish their datasets, following the instructions in Publishing datasets documentation.

Related Information

Setting the dataset recipient roles

Setting the dataset publisher role

Define groups for teams and access levels

Assign groups to roles

Assign users to groups

Publishing datasets

Cloudera Data Visualization 7.2.9

Segments

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating segments 4

Cloning segments 8

Editing segments 12

Creating entity segments 14

Deleting segments 19

Creating segments from filter definitions 20

Using segments in visuals 31

Creating segments

Cloudera Data Visualization lets you easily create a new segment definition.

Procedure

1. On the main navigation bar, click DATA.

   The Data view appears.
2. Open the Datasets tab.
3. Find the dataset in the list of datasets, either by scrolling or by using search, and click it.

   Dataset side navigation appears, open at Dataset Detail view.

   In this example, the dataset Flight Delays is used, which is based on data previously imported into Cloudera Data Visualization from a data file. Some additional data files are also used.
4. In the side navigation menu, click Segments.

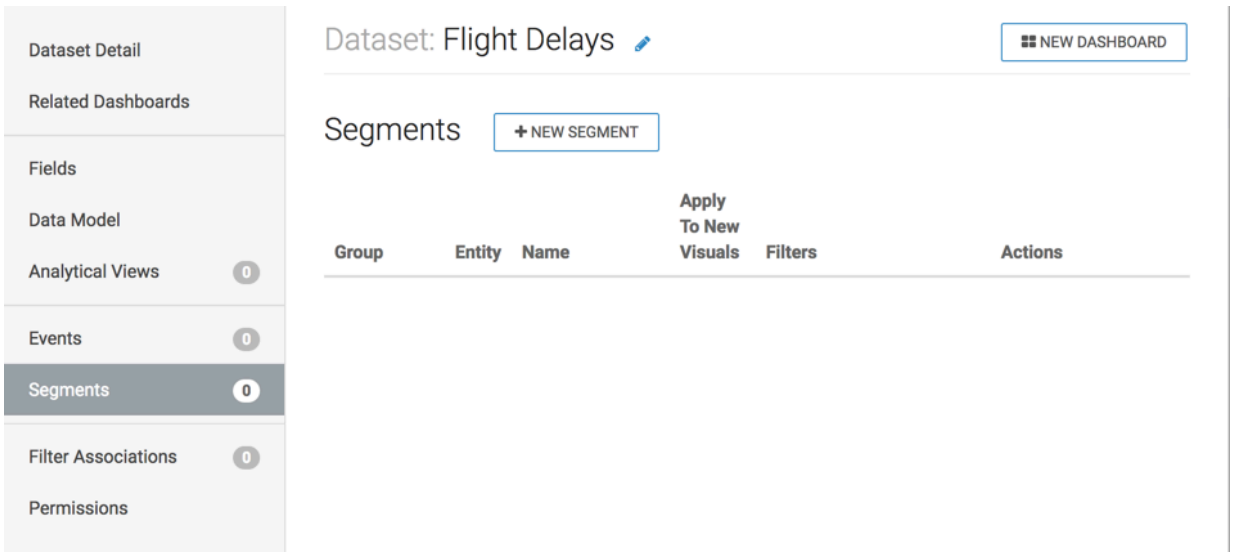

   The Segments view appears.
5. In the Segments interface, click New Segment.

   The Add Segment modal window appears.

6. Make the following entries:
   - Under Segment Name, enter Delayed Flights.
   - Under Segment Group, enter Delay Status.
   - Under Filters, click the text box to open the Filter Expression modal window.

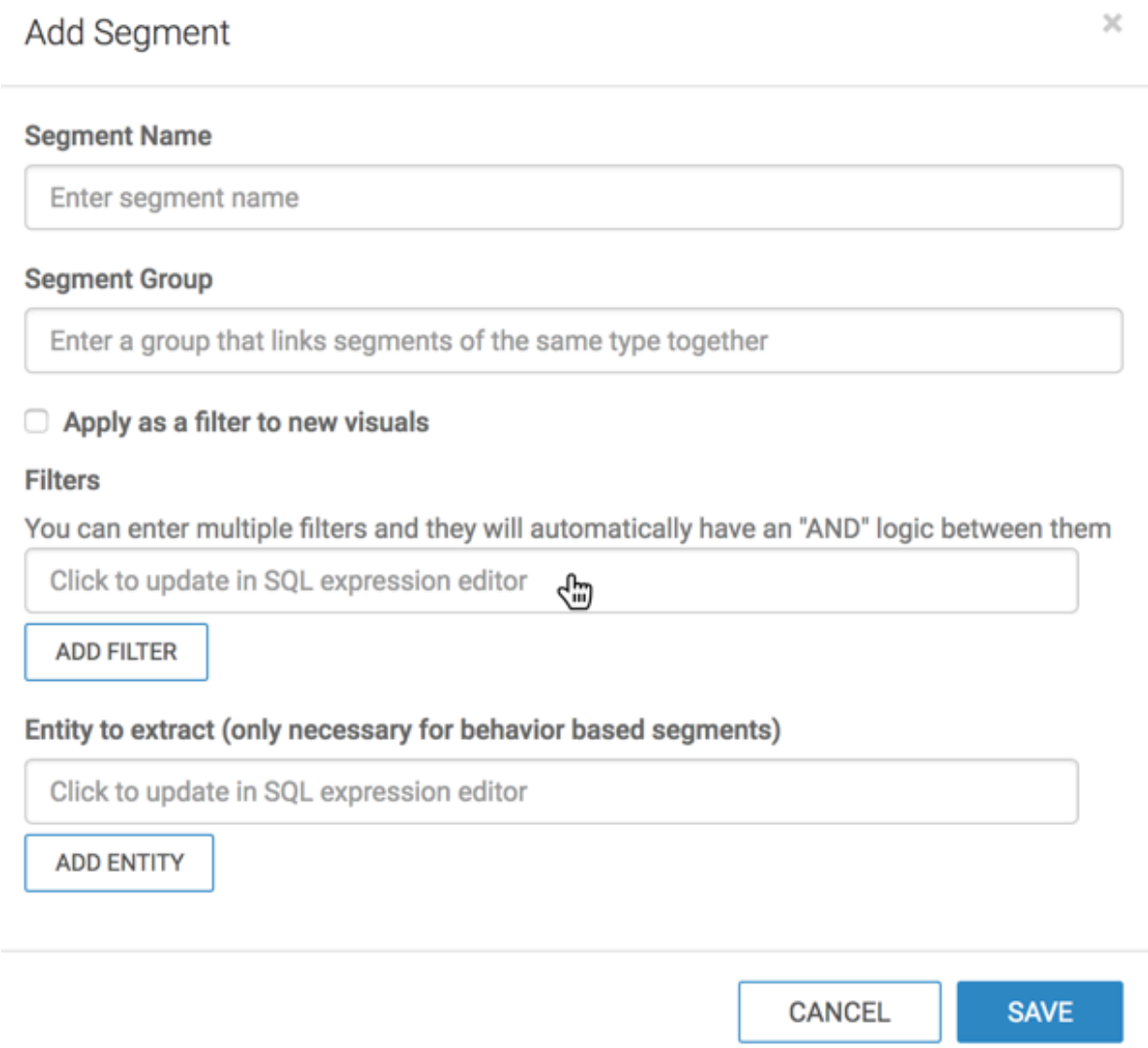

7. In the Filter Expression modal window, enter the following expression to show flights that are delayed.

```
[airtime] IS NOT NULL AND [depdelay] > 0
```

8. Click Validate Expression to verify correctness.

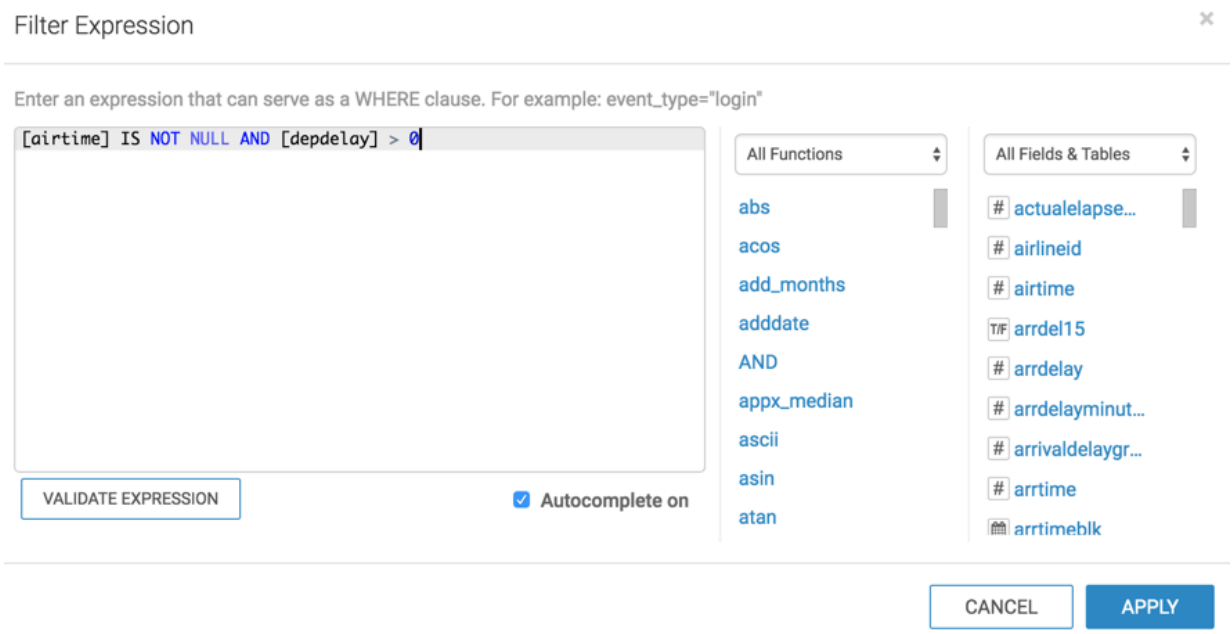

9. Click APPLY to save the filter and return to the Add Segment modal window.
The new filter appears under Filters.

10. Click SAVE.

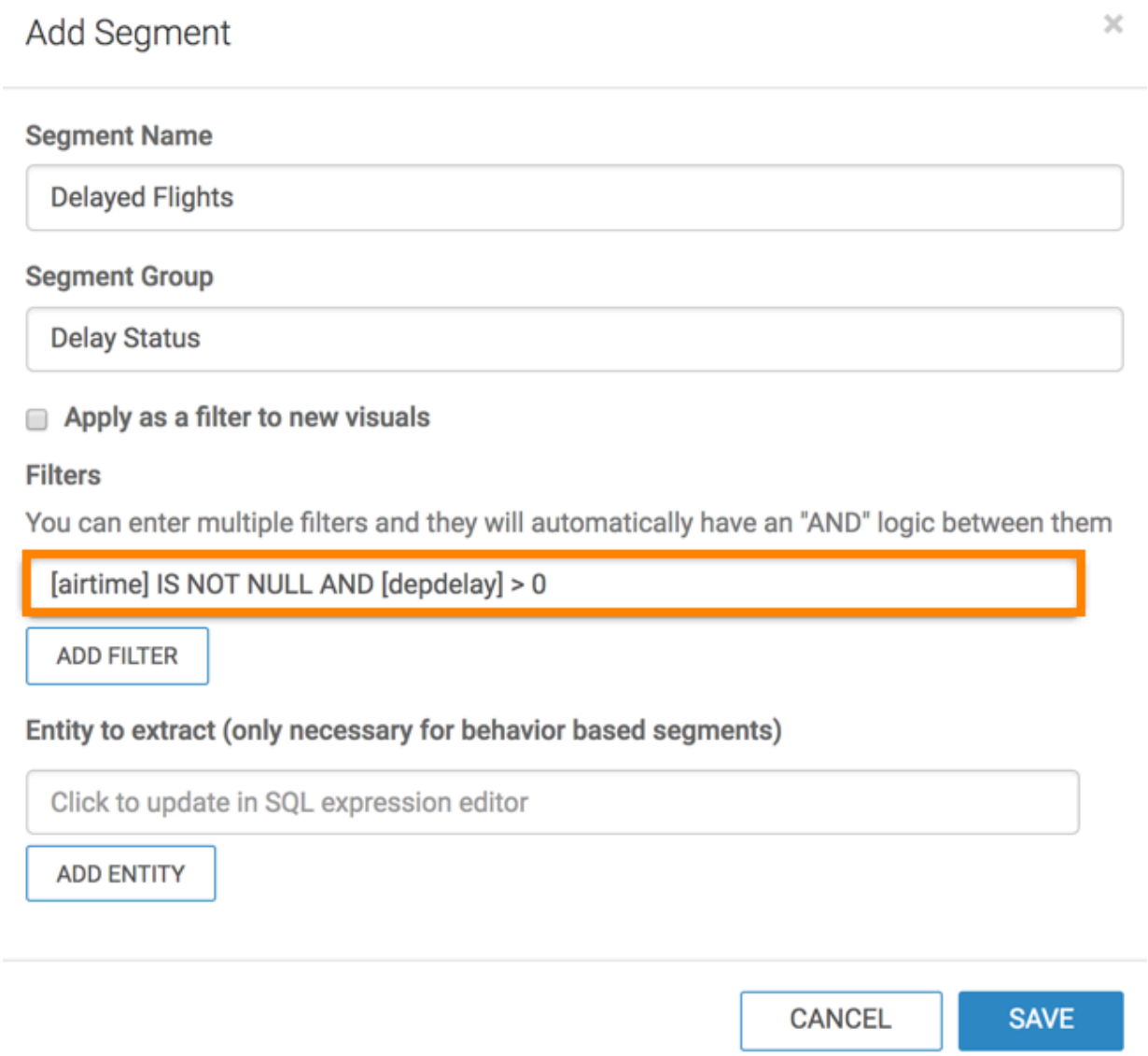

The new segment definition appears in the Segments interface.

Note: The actions available for this segment: Edit, Clone, and Delete.

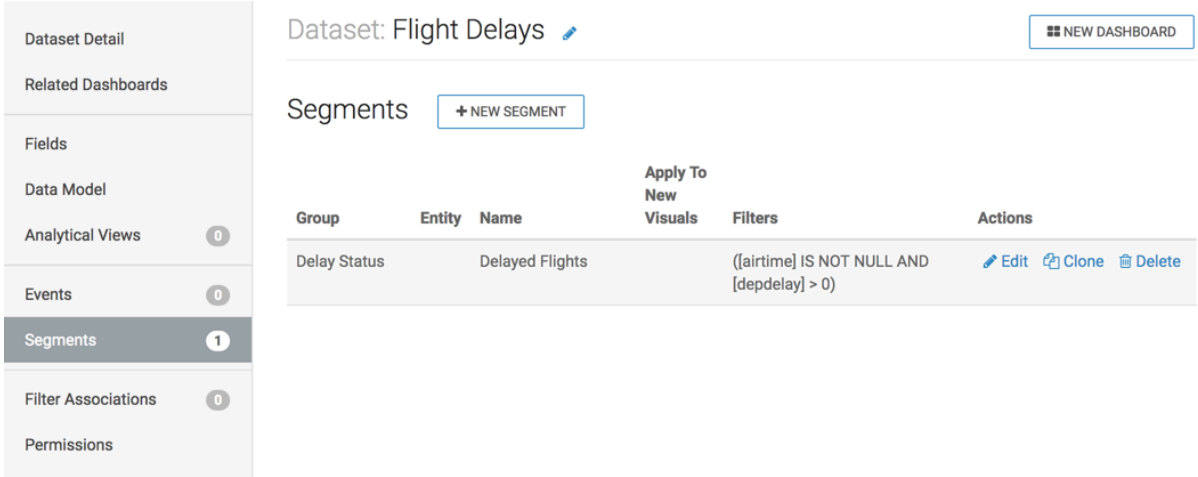

To view how segments work in a visual, see Using Segments in Visuals.

Related Information

Using segments in visuals

Cloning segments

Cloudera Data Visualization lets you clone an existing segment definition.

Procedure

1. On the main navigation bar, click DATA.
2. Select Flight Delays dataset.

   Dataset side navigation appears, open at Dataset Detail view.
3. In the side navigation menu, click Segments.
4. In the Segments list, find the segment to clone, and click Clone.

   In this example, Delayed Flights will be cloned to create a new segment On-Time Departure.

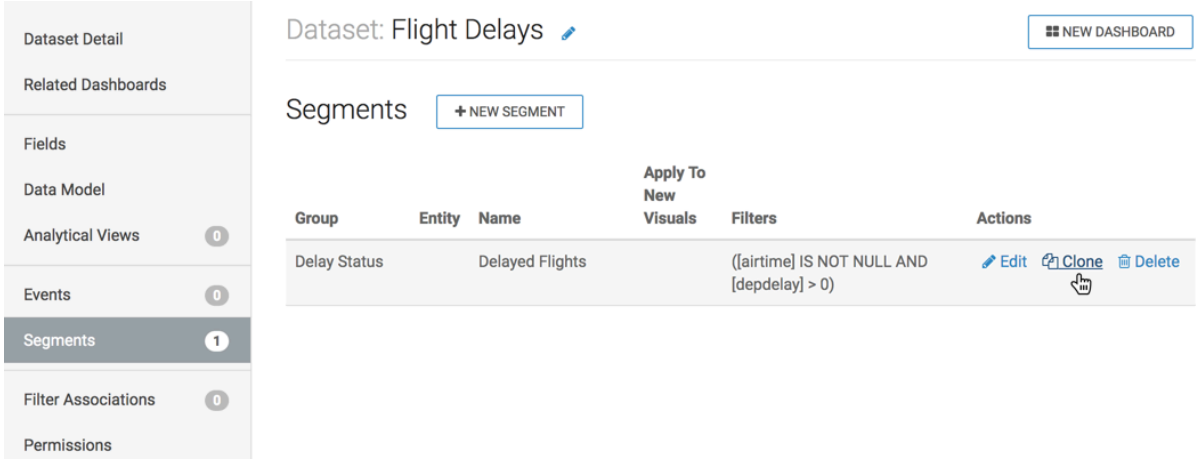

A pre-populated Add Segment modal window appears named Copy of Delayed Flights.

5. Edit the Add Segment modal window by changing its entries.
   - Change Segment Name, to On-Time Departure.
   - Leave Segment Group as is.
   - Under Filters, click the text box to open the Filter Expression modal window.

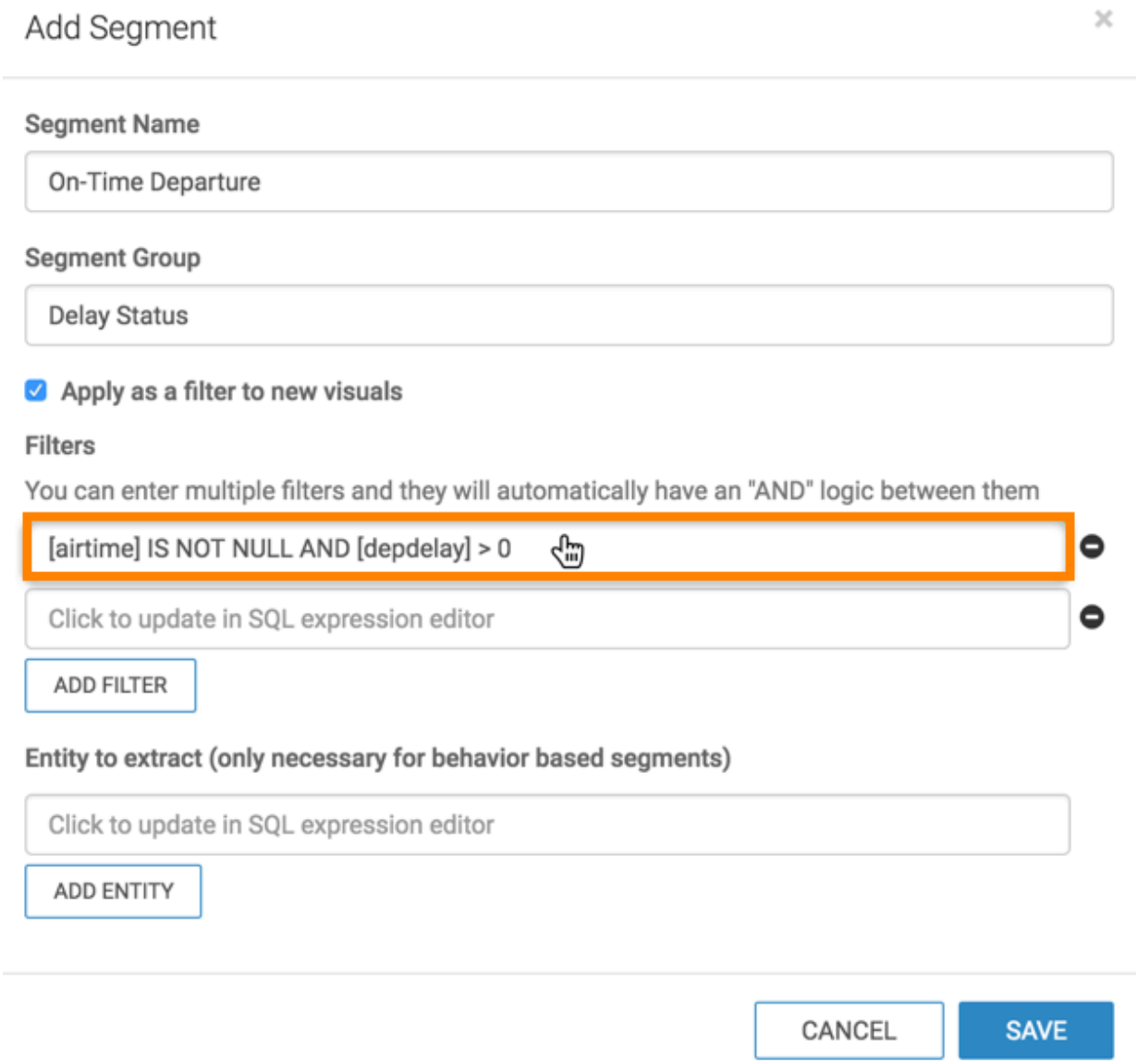

6. Replace the filter expression. In our example, we are adding the following expression:

```
[airtime] IS NOT NULL AND [depdelay] < = 0
```

7. Click Validate Expression to verify correctness.

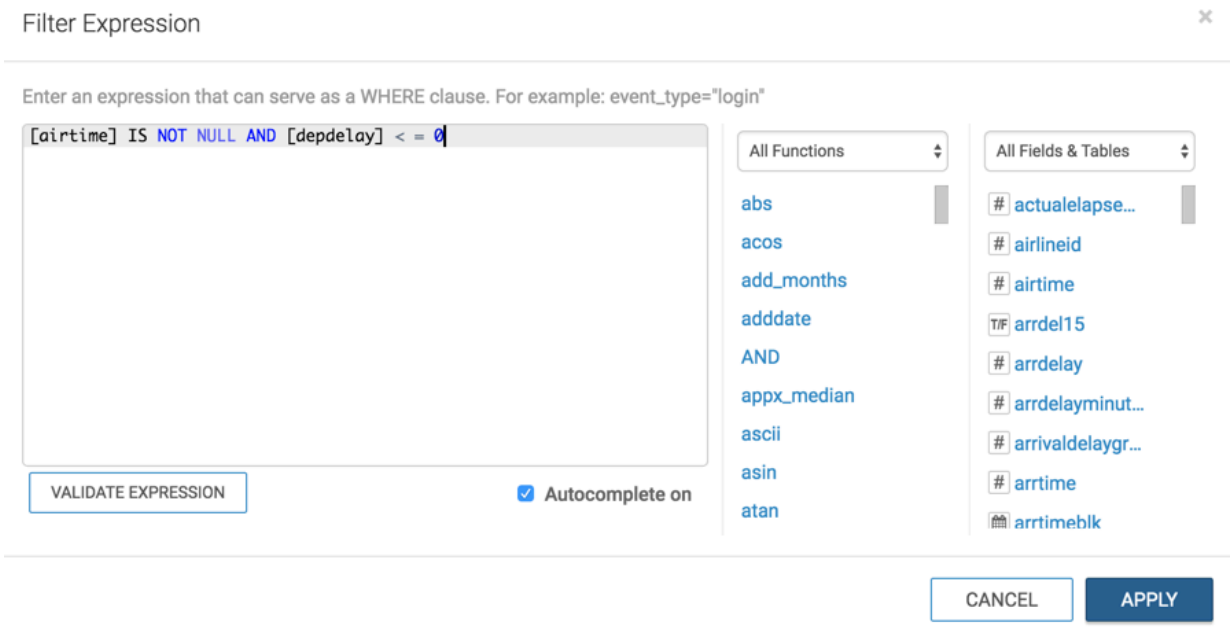

8. Click Apply to save the filter and return to the Add Segment modal window.
9. Under Filters, notice the new filter expression.

10. Click Save.

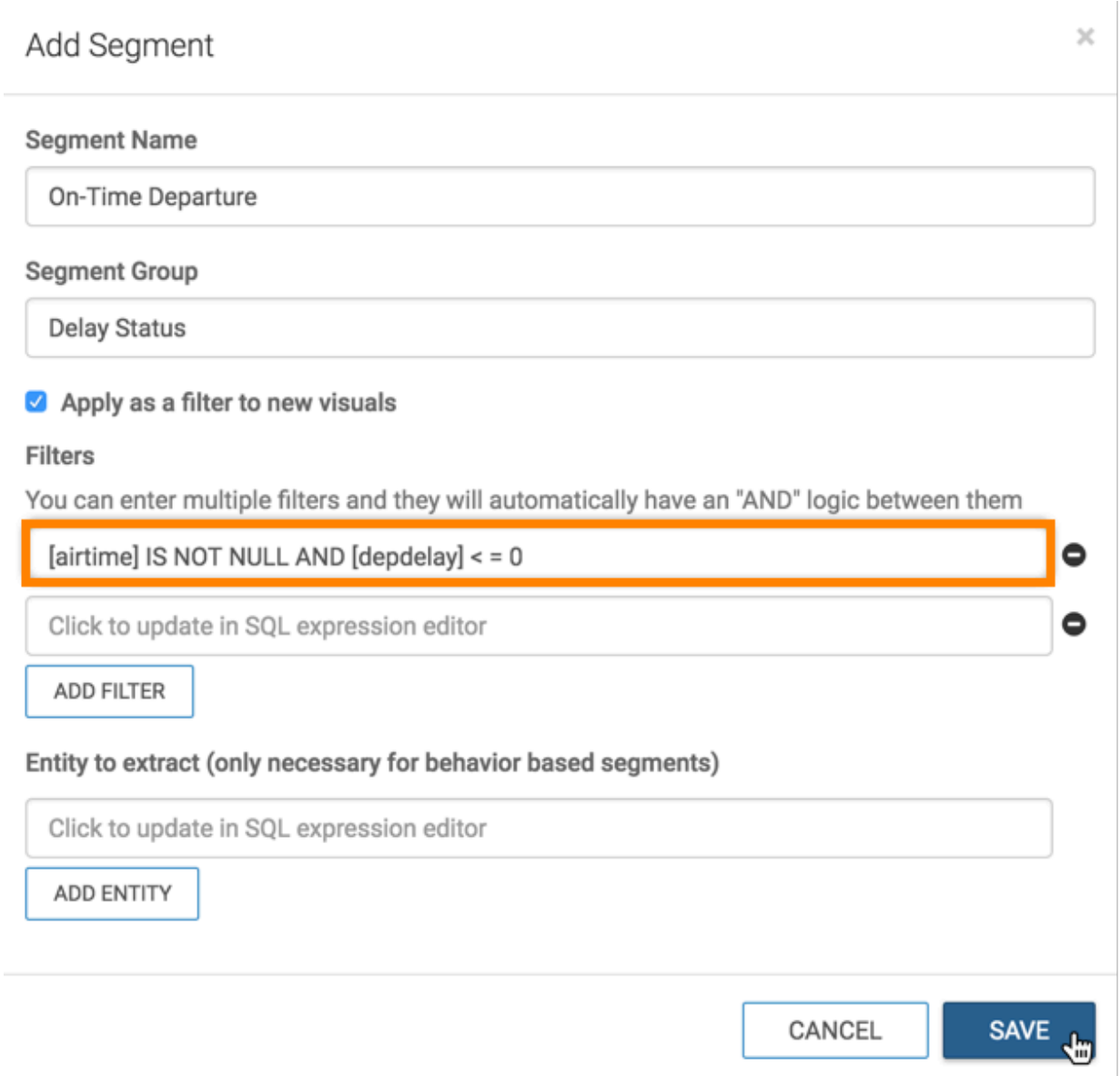

11. The new segment definition appears in the Segments interface.

Note the similarity in the definition of the two filters, Delayed Flights and On-Time Departure.

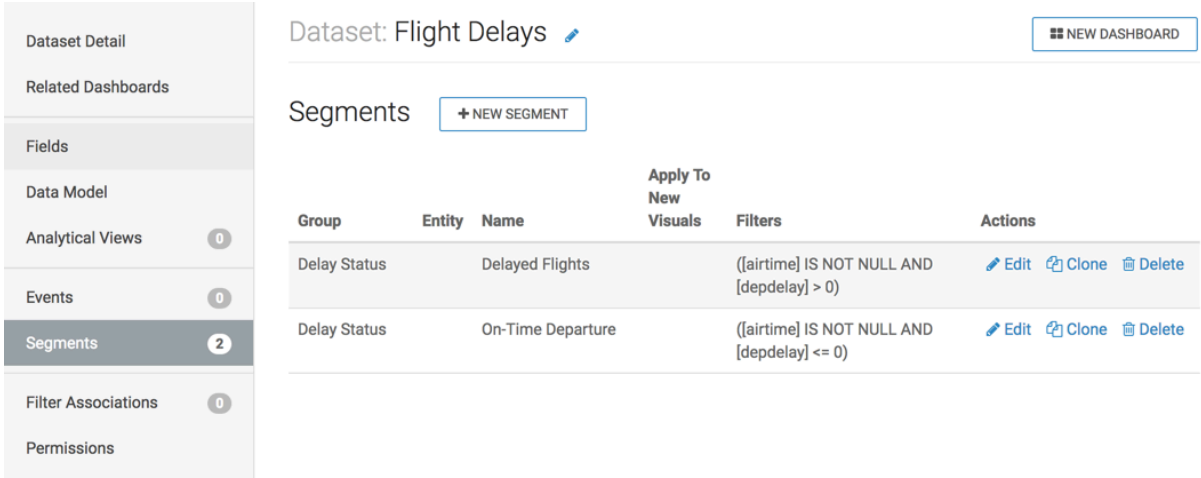

Editing segments

Cloudera Data Visualization allows you to edit an existing segment definition.

Procedure

1. On the main navigation bar, click DATA.
2. Select Flight Delays dataset.

   Dataset side navigation appears, open at Dataset Detail view.
3. In the side navigation menu, click Segments.
4. In the Segments list, find the segment to edit, and click Edit.

   In our example, let's edit the segment Delayed Flights to change the segment name from Delayed Flights to Delayed Departure.

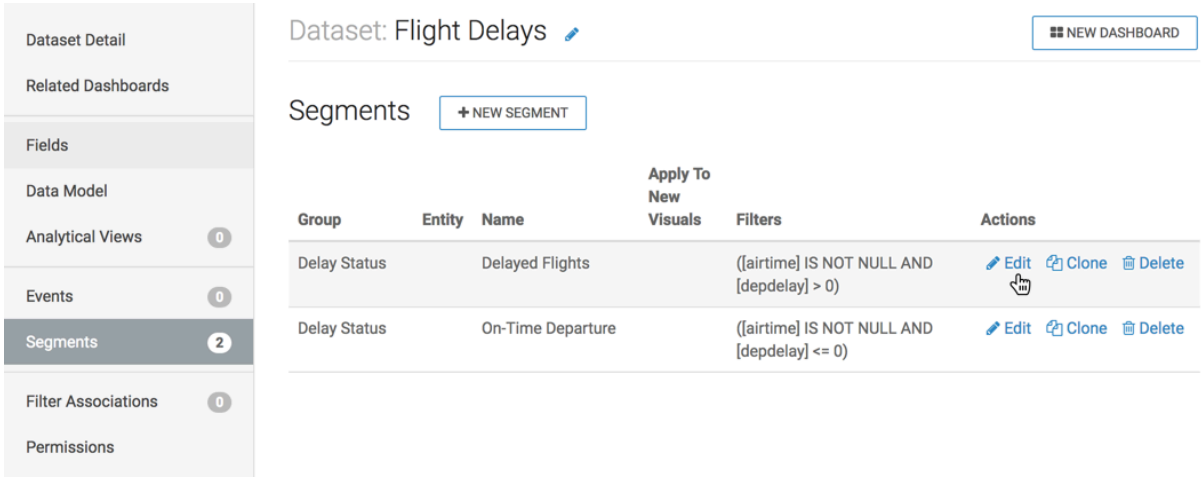

A pre-populated Edit Segment modal window appears.

5. Change the entries in Edit Segment modal window.
   - Change Segment Name from Delayed Flights to Delayed Departure.
   - Leave all other fields 'as is'.

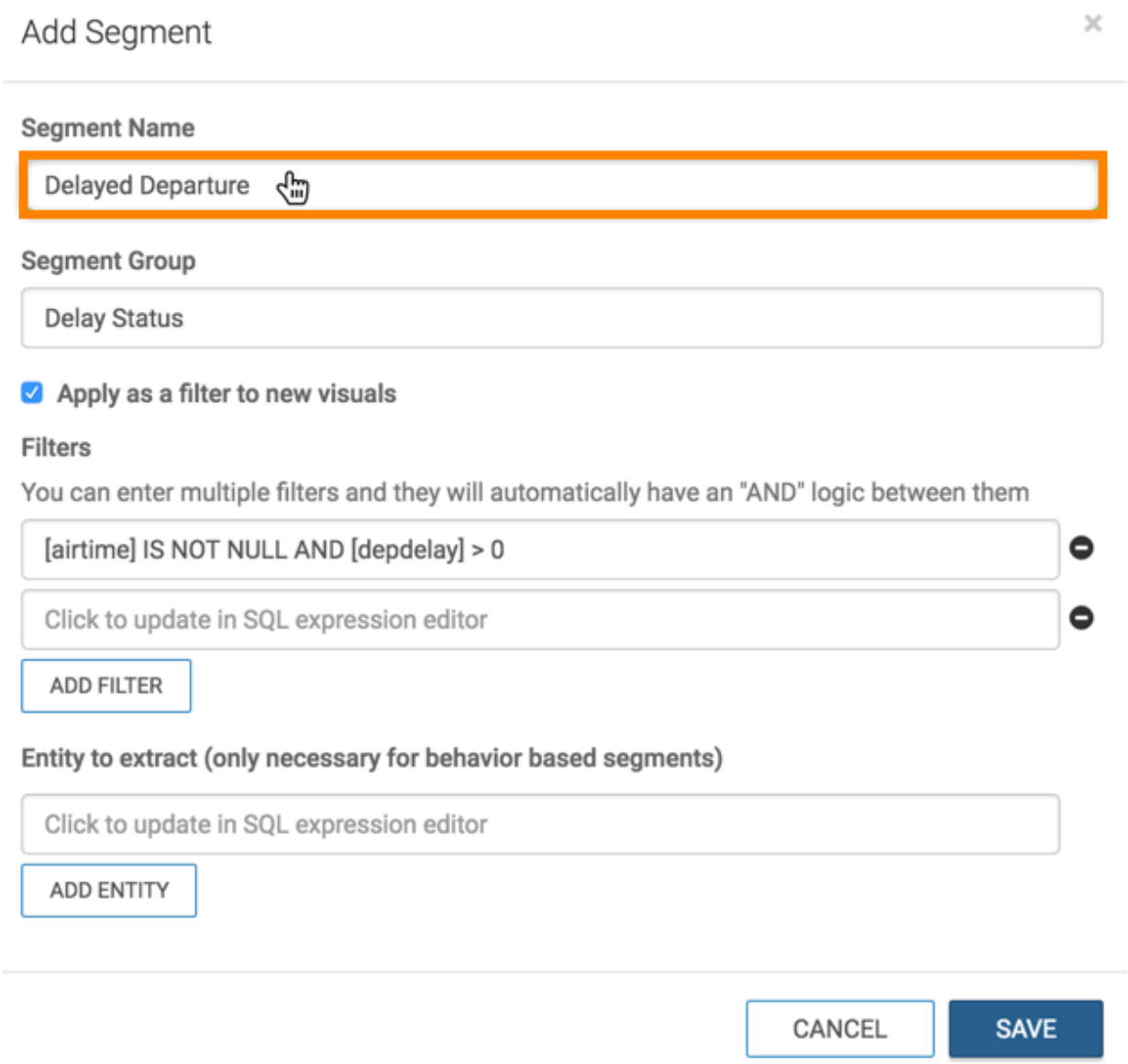

6. Click SAVE.

   The renamed/edited segment appears in the Segments interface.

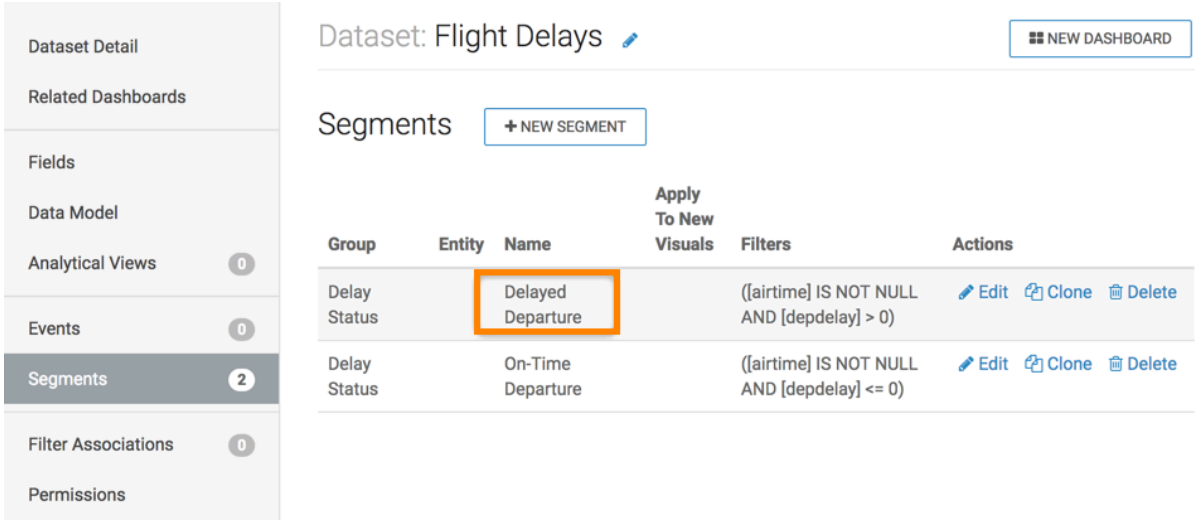

Creating entity segments

Entity segments are a more advanced use of segments, where the extraction of an entity enables cohort analysis.

Procedure

1. On the main navigation bar, click DATA.
2. Select Flight Delays dataset.

   Dataset side navigation appears, open at Dataset Detail view.
3. In the side navigation menu, click Segments.

4. In the Segments list, find the segment to clone, and click Clone.

   In this example, to make our task simpler, the Delayed Departure segment that was created in Cloning Segments will be cloned.

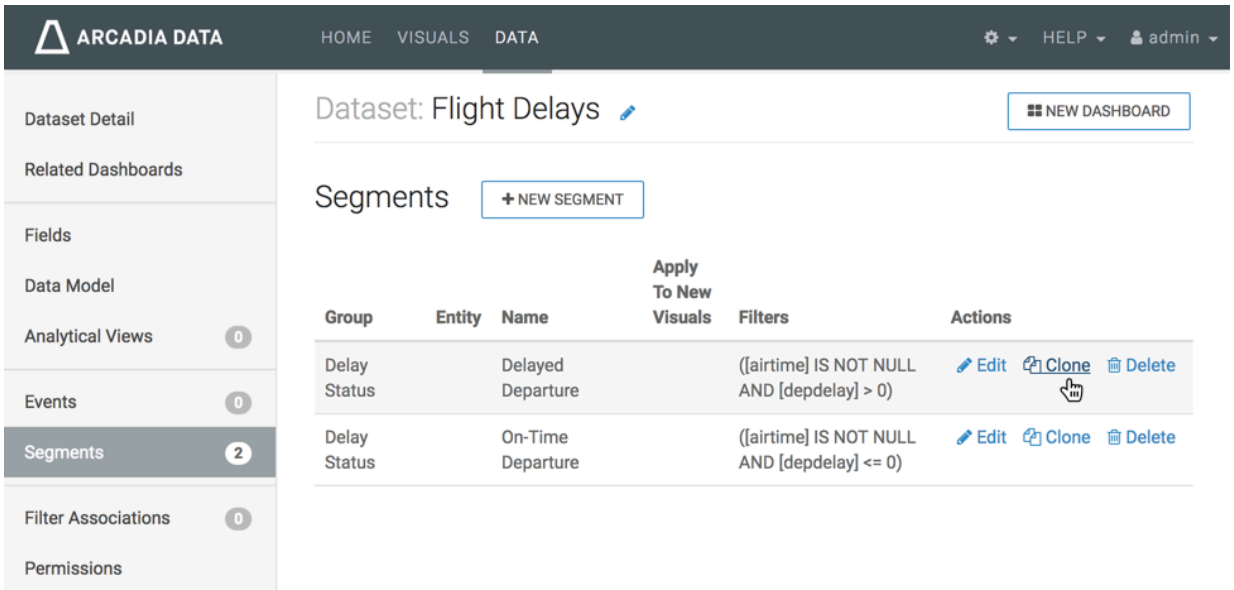

   A pre-populated Add Segment modal window appears.

5. Edit the Add Segment modal window by changing its entries.
   - Change Segment Name to Delayed Airlines.
   - Leave other fields 'as is'.
   - Under Entity to extract, click the text box to open the Entity Expression modal window.

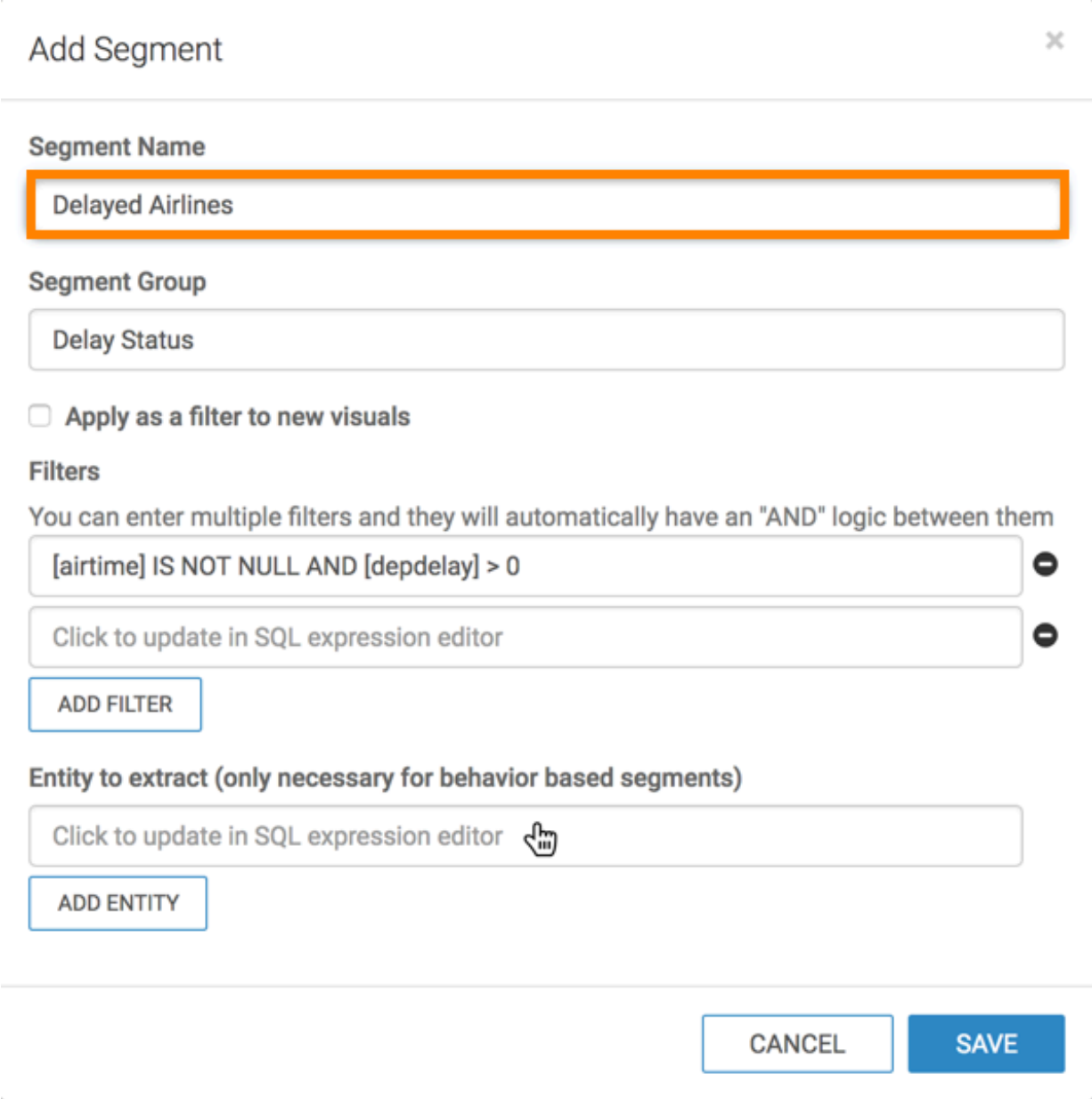

6. In the Entity Expression modal window, enter the following expression.

```
[airlineid]
```

7. Click Validate Expression to verify correctness.

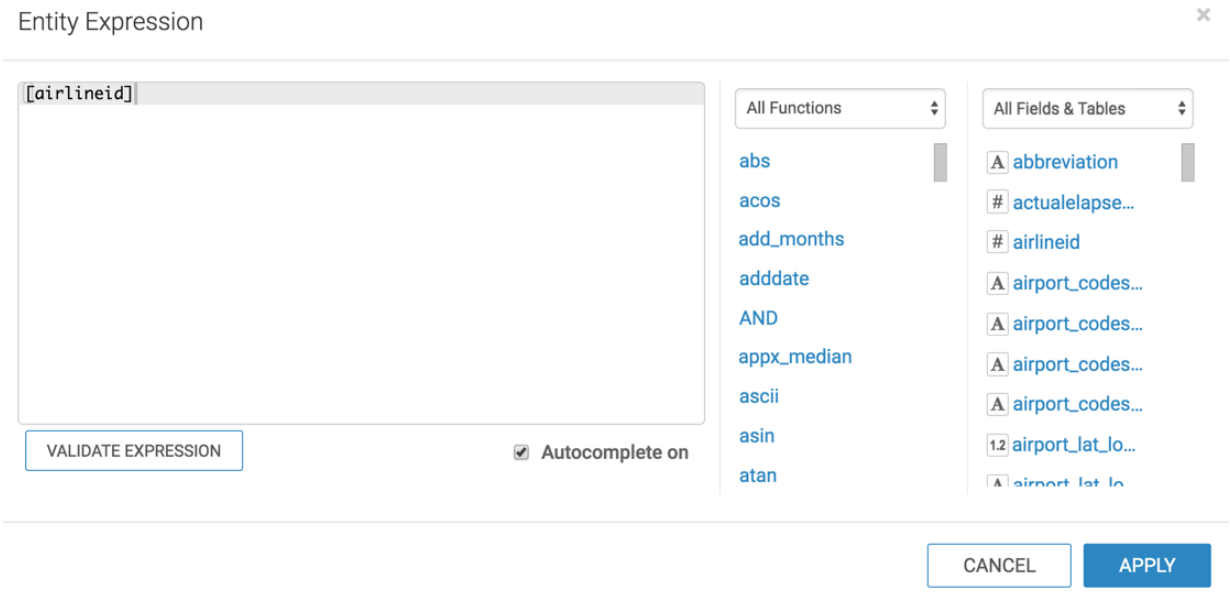

8. Click APPLY to save the expression and return to the Add Segment modal window.
9. In the Add Segment modal window, notice the new expression in the Entity to extract field.

10. Click SAVE.

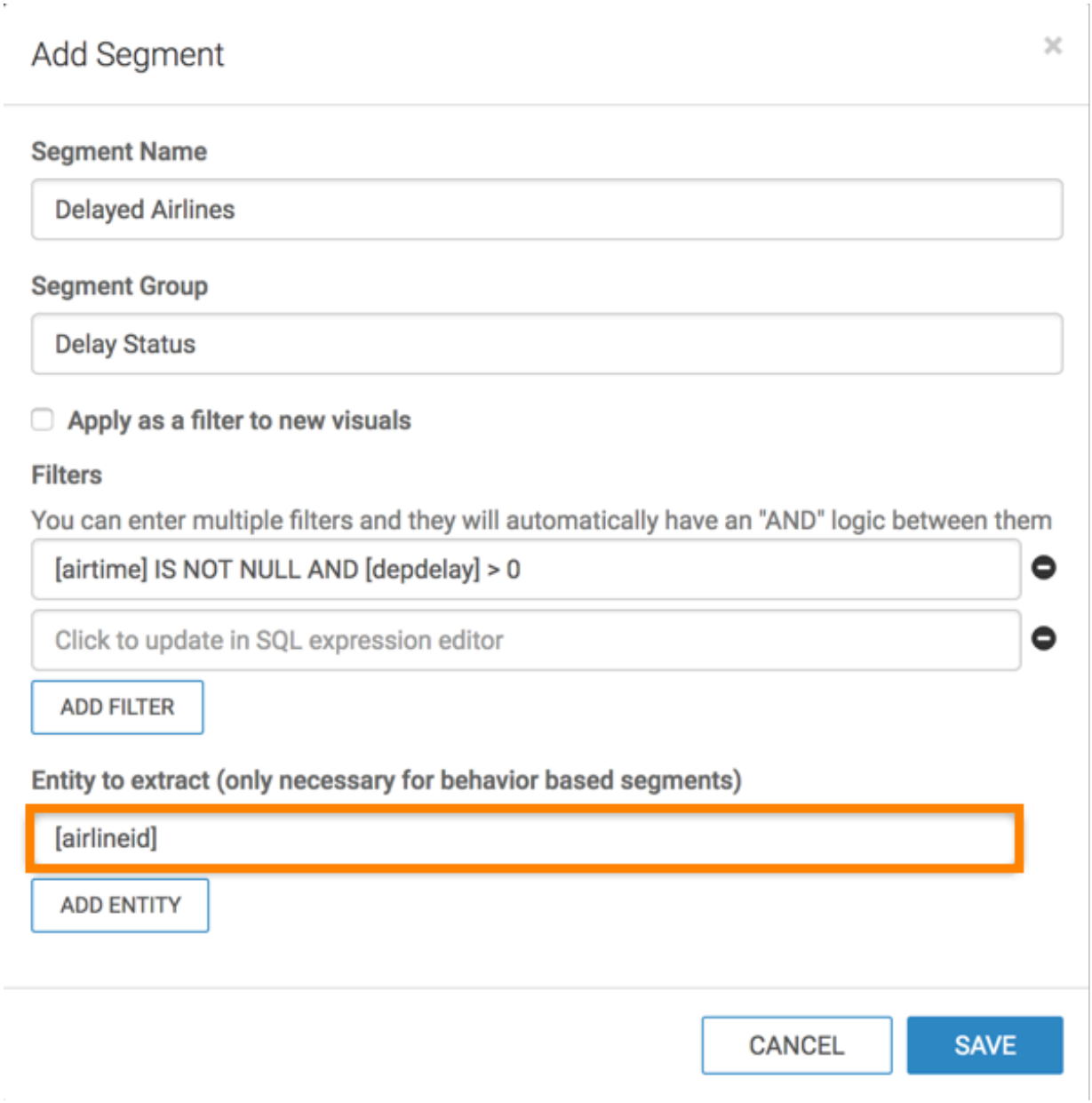

The new segment definition appears in the Segments interface. The segment Delayed Airlines has an entry in the Entity column, airlineid.

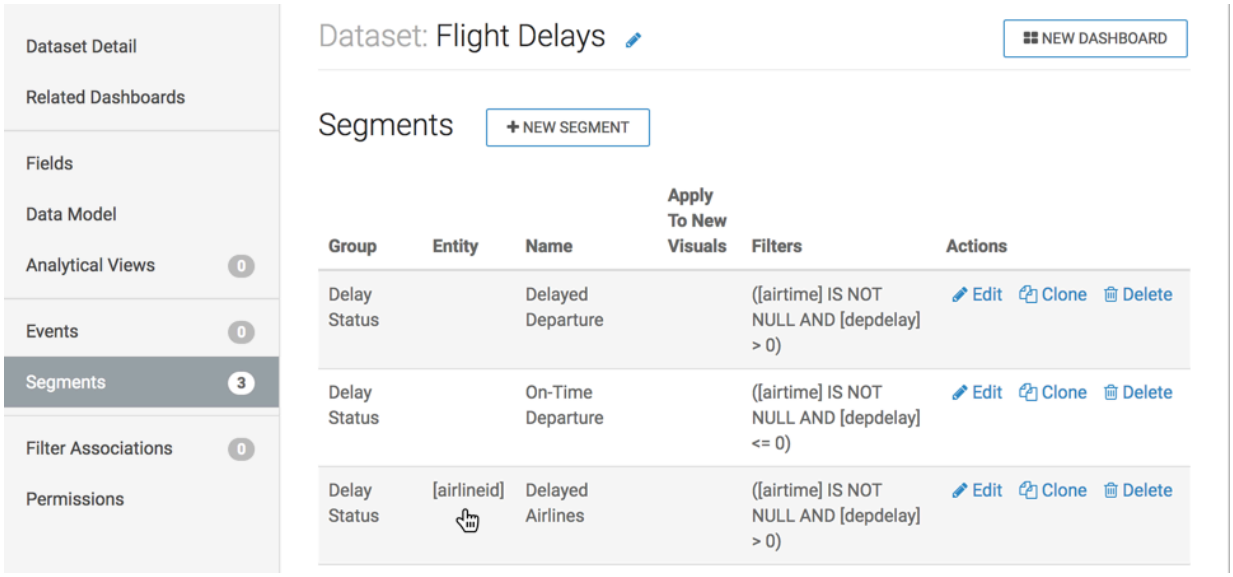

To view how segments work in a visual, see Using Segments in Visuals .

Related Information

Cloning segments

Using segments in visuals

Deleting segments

Cloudera Data Visualization lets you easily delete dataset segment definition.

Procedure

1. On the main navigation bar, click DATA.
2. Select Flight Delays dataset.

   Dataset side navigation appears, open at Dataset Detail view.
3. In the side navigation menu, click Segments.
4. In the Segments list, find the segment to delete, and click Delete.

   In this example, we are deleting Short Flights.

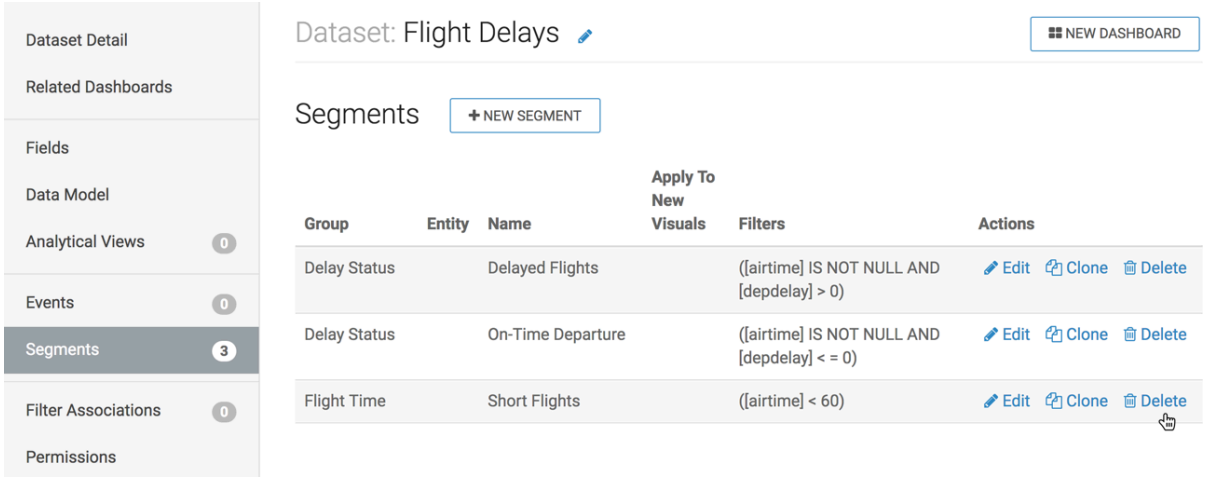

5. Click DELETE in the confirmation modal window.

Results

The segment Short Flights no longer appears in the Segments interface.

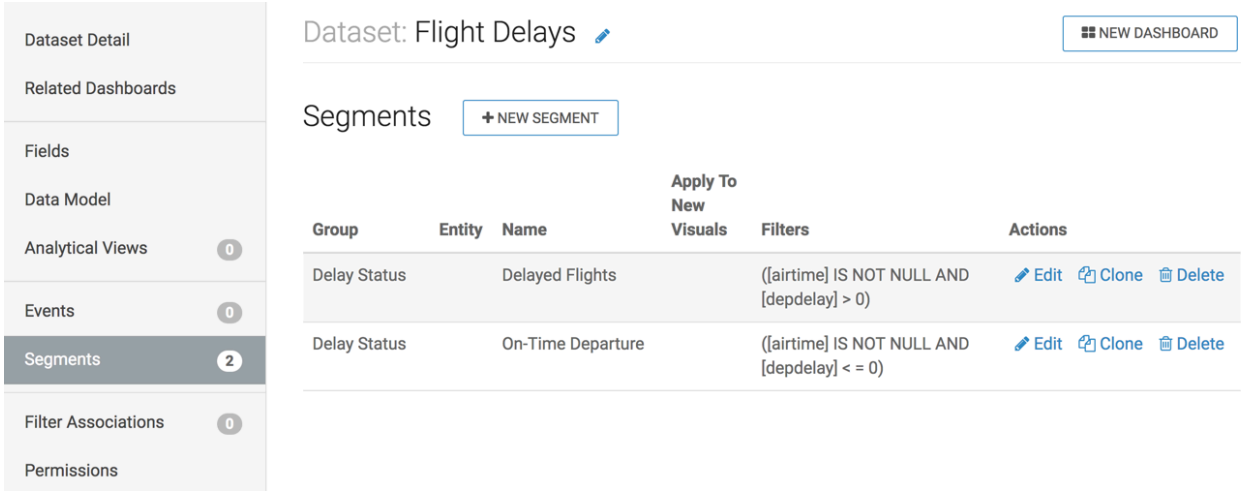

Creating segments from filter definitions

Besides creating segments in the dataset detail view, Cloudera Data Visualization also lets you create segments from a filter definition of a visual.

About this task

See the following topics to create a new segment from a filter in a visual, then edit the filter to add another segment, and finally update an existing segment.

- Creating the first segment
- Adding the second segment
- Viewing the defined segments
- Adding values to an existing segment

Follow these steps to create a new data segment, Mountain from a filter definition.

Procedure

Creating the first segment

1. Start a new visual based on dataset Flight Delays, based on data previously imported into Cloudera Data Visualization from a data file.

   For more information, see Creating a visual.

2. Add originstate to the Filters shelf.

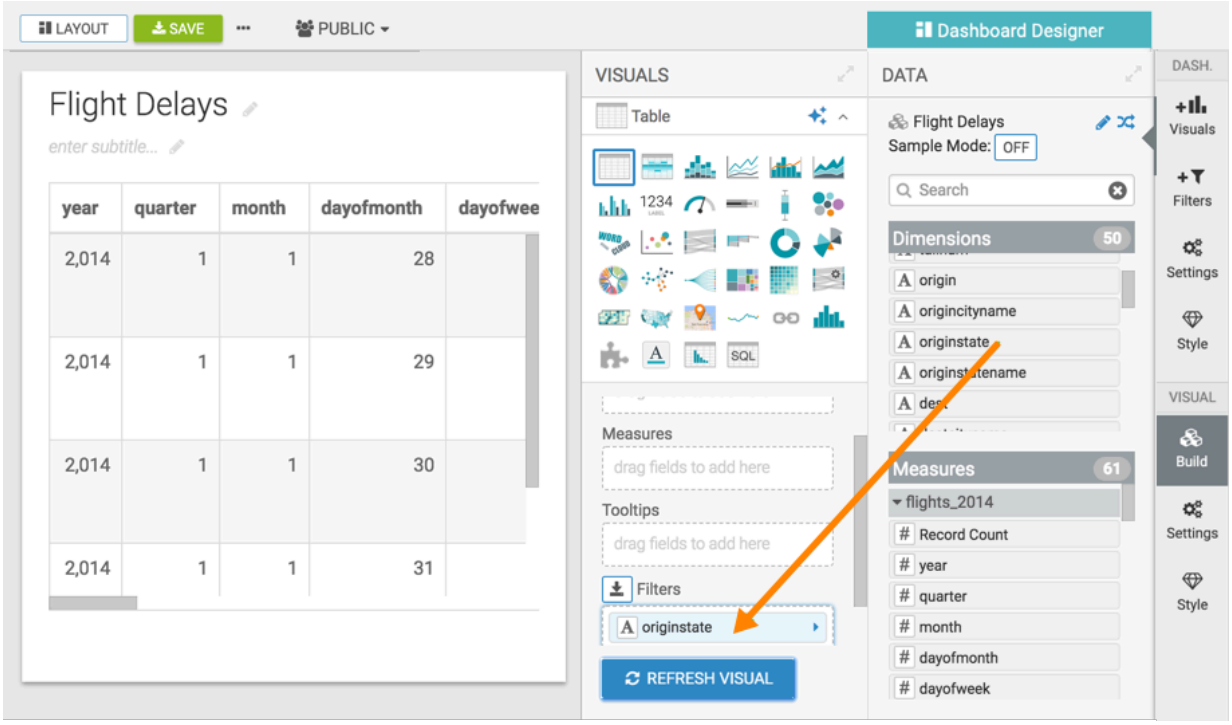

3. In the Filter for originstate window modal, select CO and WY, and click APPLY.

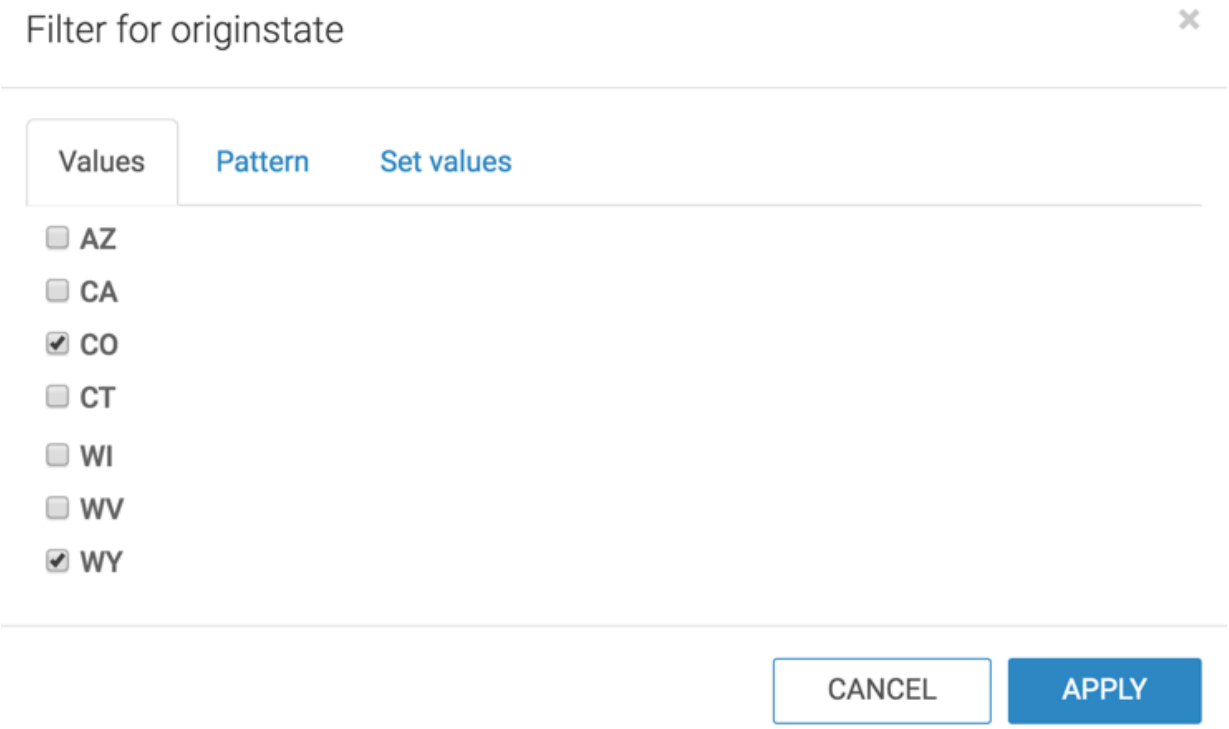

4. Click the Save Down icon to the left of the Filters shelf.

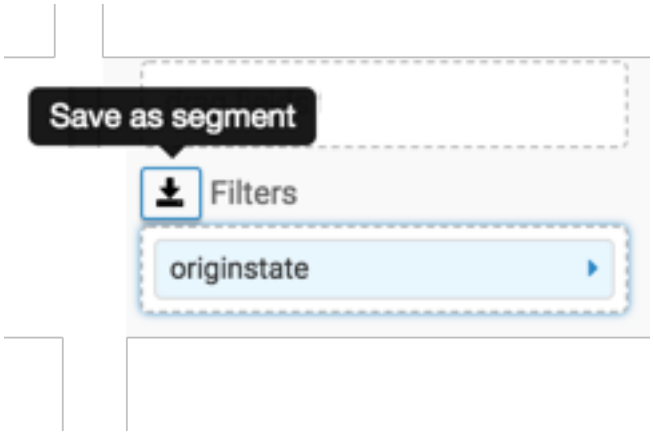

5. From the menu, select Create New Segment.

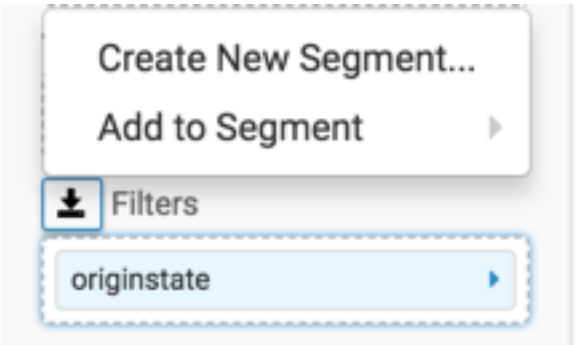

6. In the Add Segment window modal, specify the following:
   - Under Segment Name, enter Mountain.
   - Under Segment Group, enter Geographic Region.
   - Under Filters, notice the new filter expression, [originstate] in ('CO', 'WY').
   - Leave Entity to Extract field empty.

7. Click SAVE.

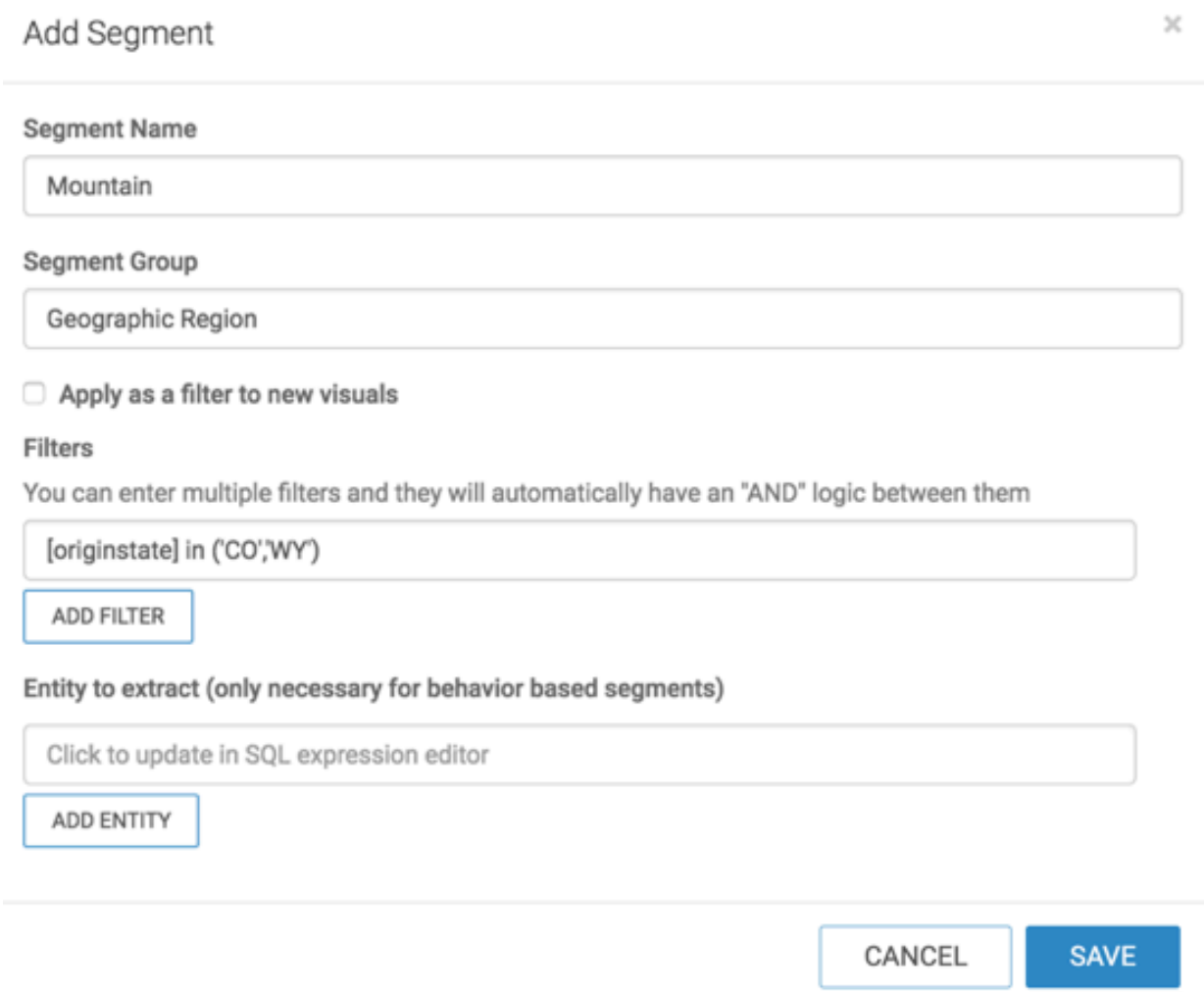

The new segment definition appears in Segments section on the right navigation menu.

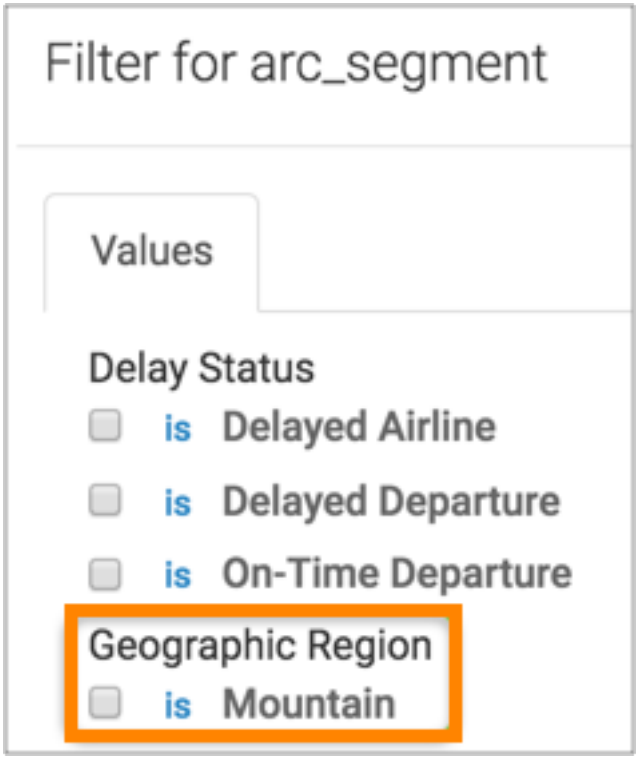

Adding the second segment

Let's edit the filter defined in the first segment and create a second segment.

8. On the Filters shelf, at the right of the originstate filter, click the Down icon to open the Filter Properties menu.

9. From the Filter Properties menu, select Clear values.

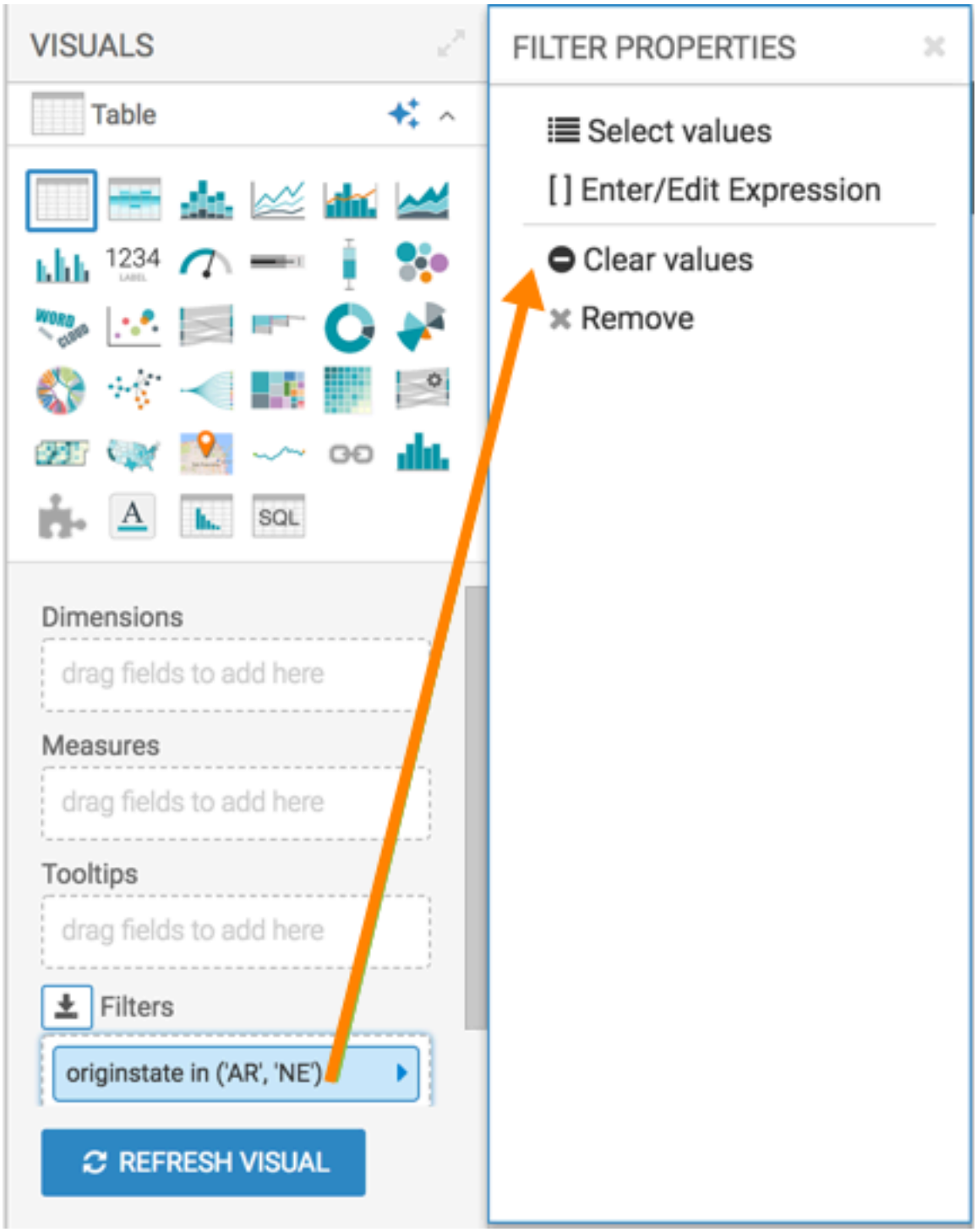

10. Click the Down icon to the right of the originstate filter to open the Filter Properties menu.

11. From the Filter Properties menu, select Select values.

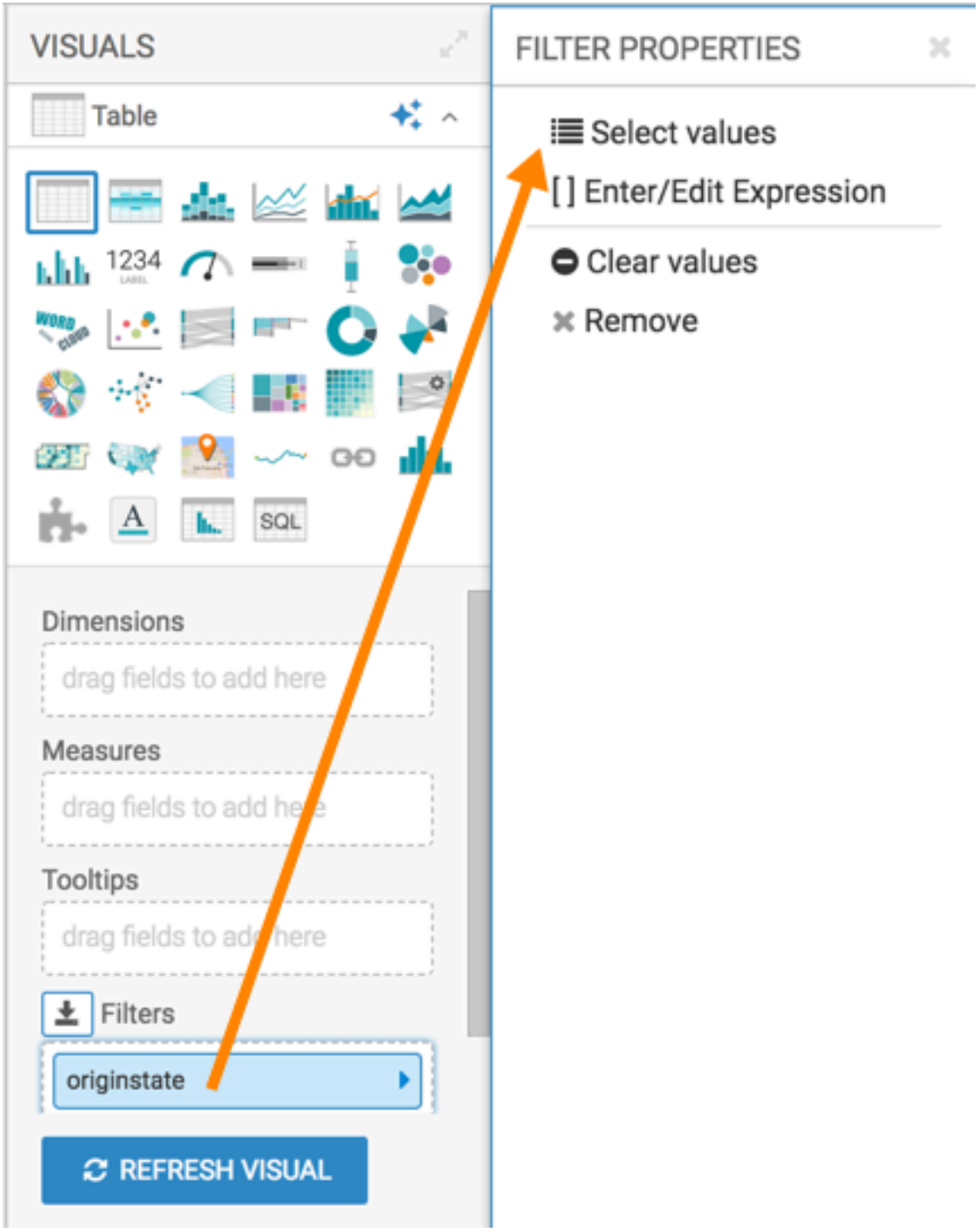

12. In the Filter for originstate window modal, select WA.

13. Click APPLY.

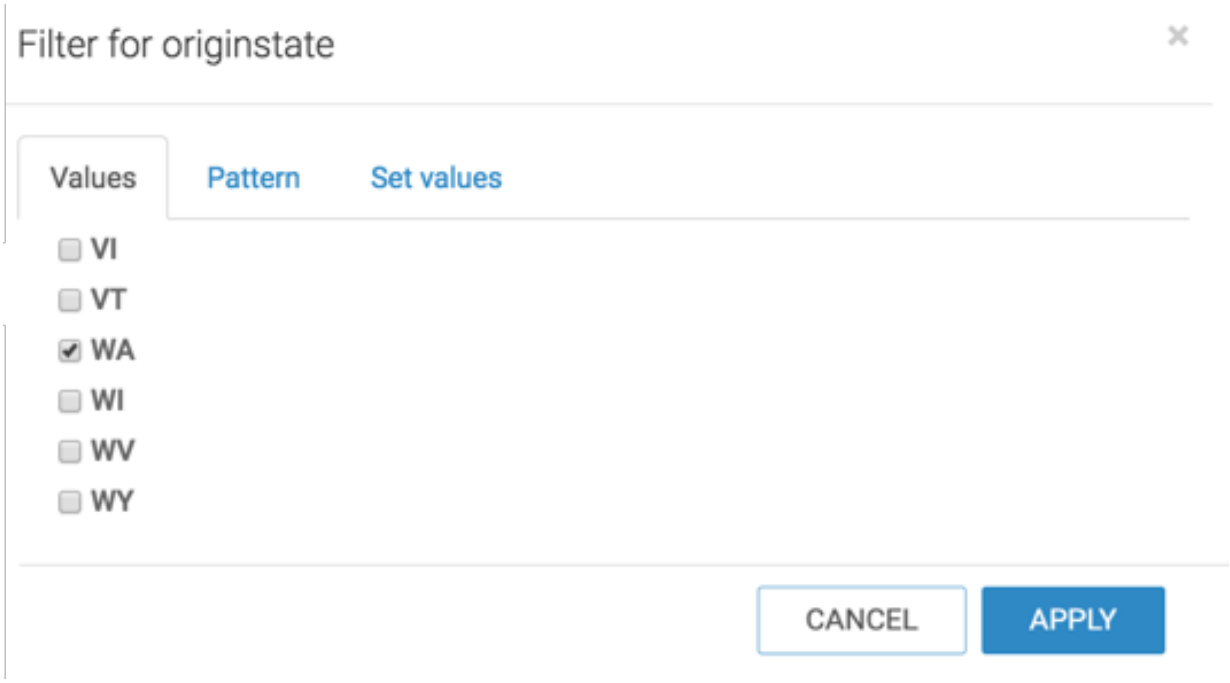

14. To define the next segment, select Create New Segment from the Save as Segment menu.

15. In the Add Segment modal window, specify the following:

- Under Segment Name, enter North West.
- Under Segment Group, enter Geographic Region that you used earlier.
- Under Filters, notice the new filter expression, [originstate] in ('WA').
- Leave Entity to Extract field empty.

16. Click Save.

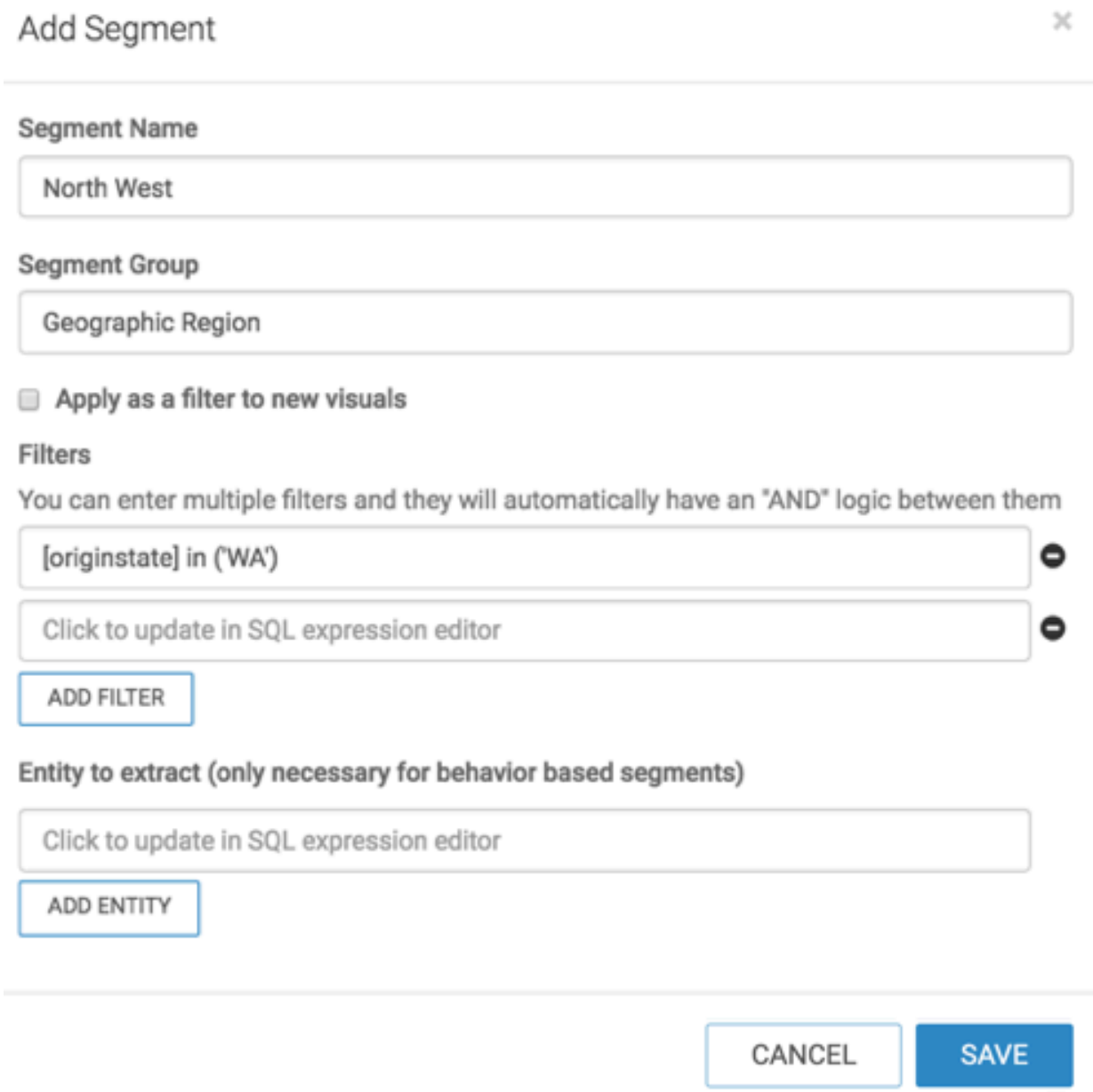

Viewing the defined segments

17. In the side navigation bar of the visual interface, click Segments.

Notice that the two segments (and the segment group) you just defined appear in the Segments menu.

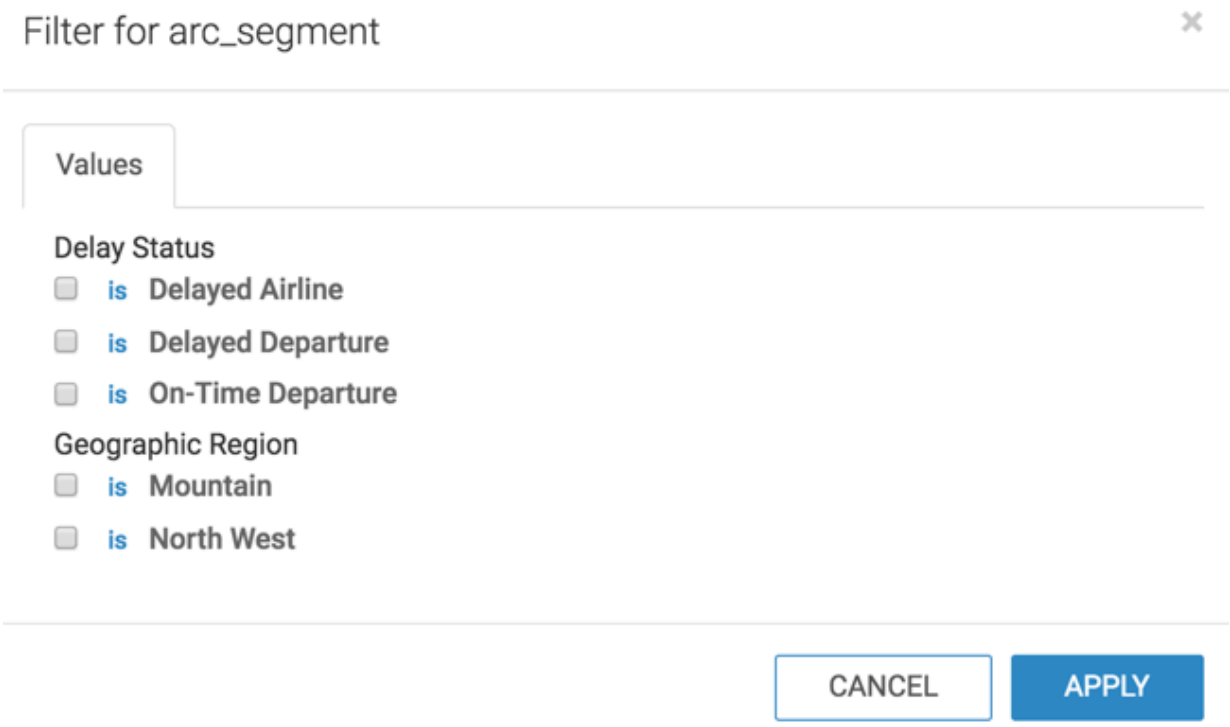

Adding values to an existing segment

Let's add a new state, OR, to our existing segment North West.

18. Click inside the Filter shelf to open the Field Properties menu.

19. Click Clear values.

20. Click Select values.

21. Select OR in the Filter for originstate window modal.

22. Click APPLY.

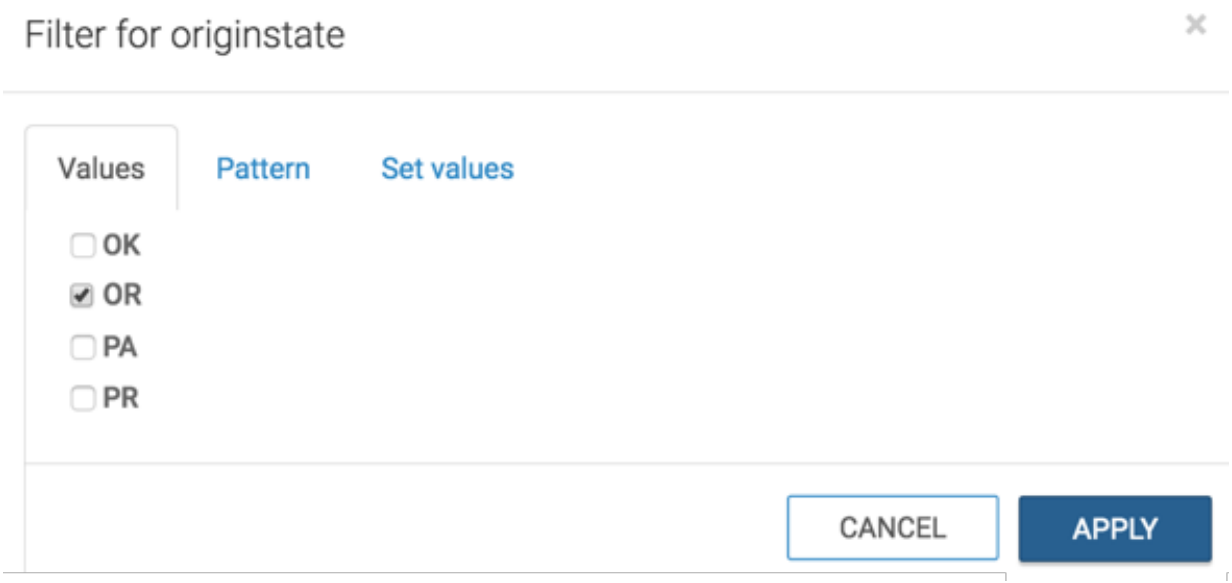

23. To add the new selection to an existing segment, do the following:

- Click Filters.
- In the menu, select Add to Segment from the Save as Segment menu.
- In the secondary menu, under Geographic Regions, select North West.

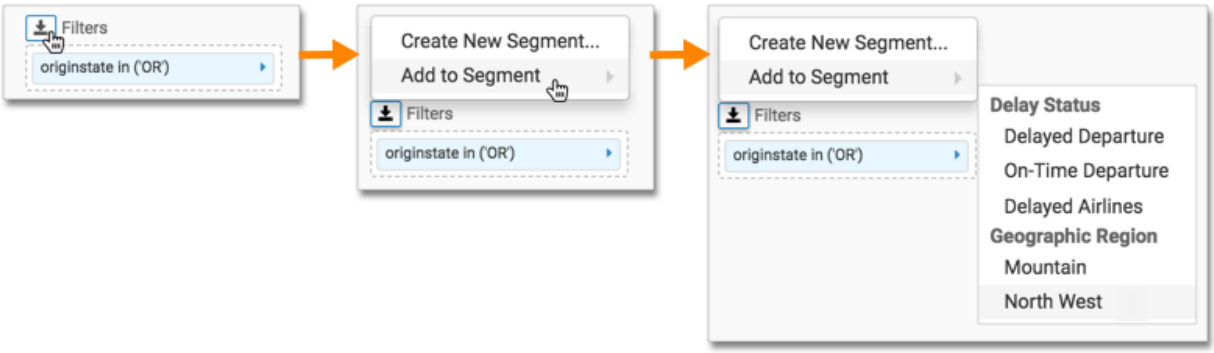

Notice that when the Edit Segment modal window appears, it shows the existing configuration for the segment North West.

However, its Filters specification now includes Oregon: [originstate] in ('WA', 'OR').

24. Click Save to finish adding the specification to the segment.

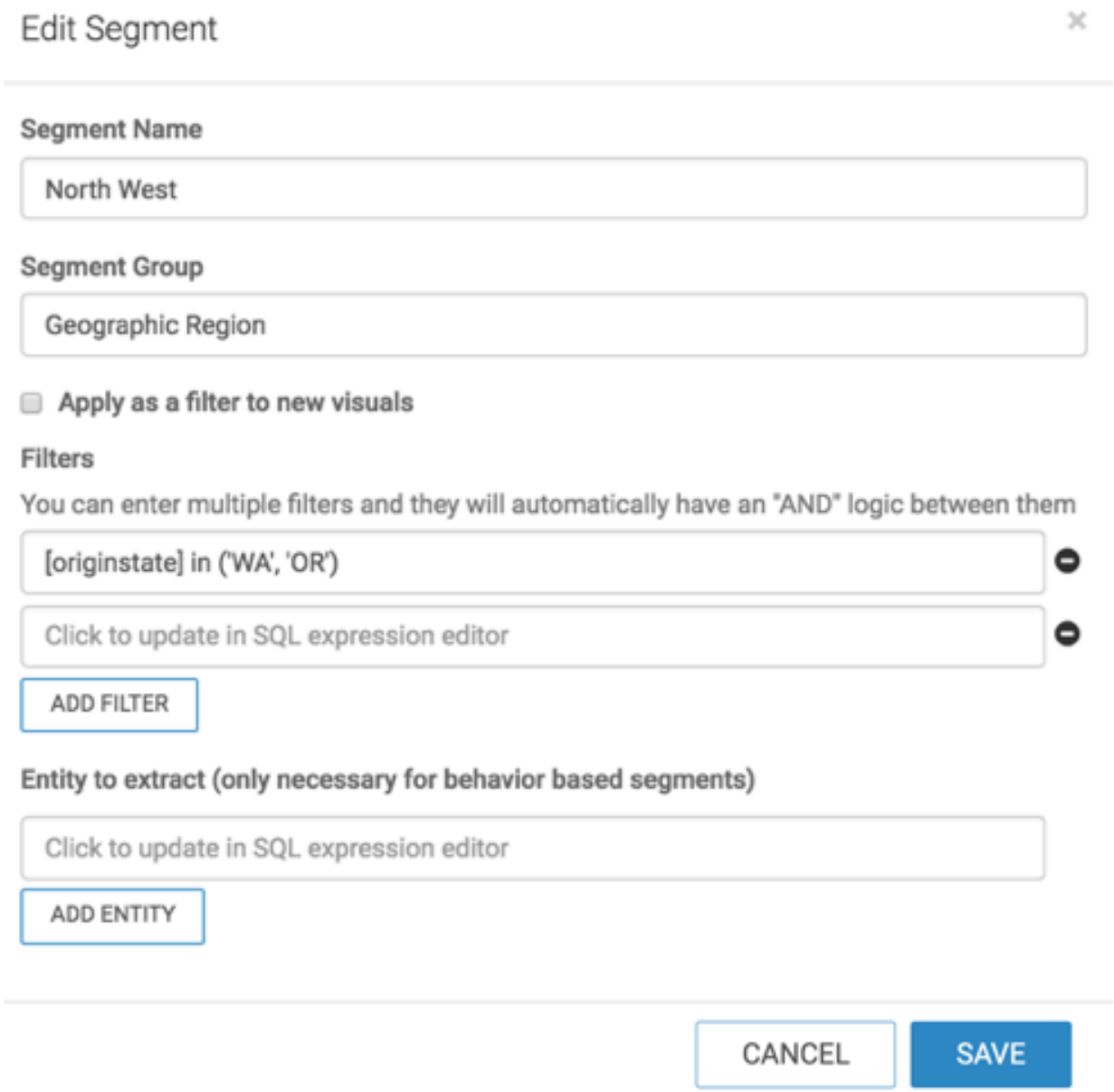

What to do next

To view how segments work in a visual, see *Using segments in visuals*.

Related Information

Using segments in visuals
Creating a visual

Using segments in visuals

This article demonstrates how segments work in a visual.

About this task

After creating segments in the Dataset Segments user interface, or creating segments from filter definitions of a visual, you can view this segmented data in a visual.

Procedure

1. Click Data in the side navigation bar of the visual interface.

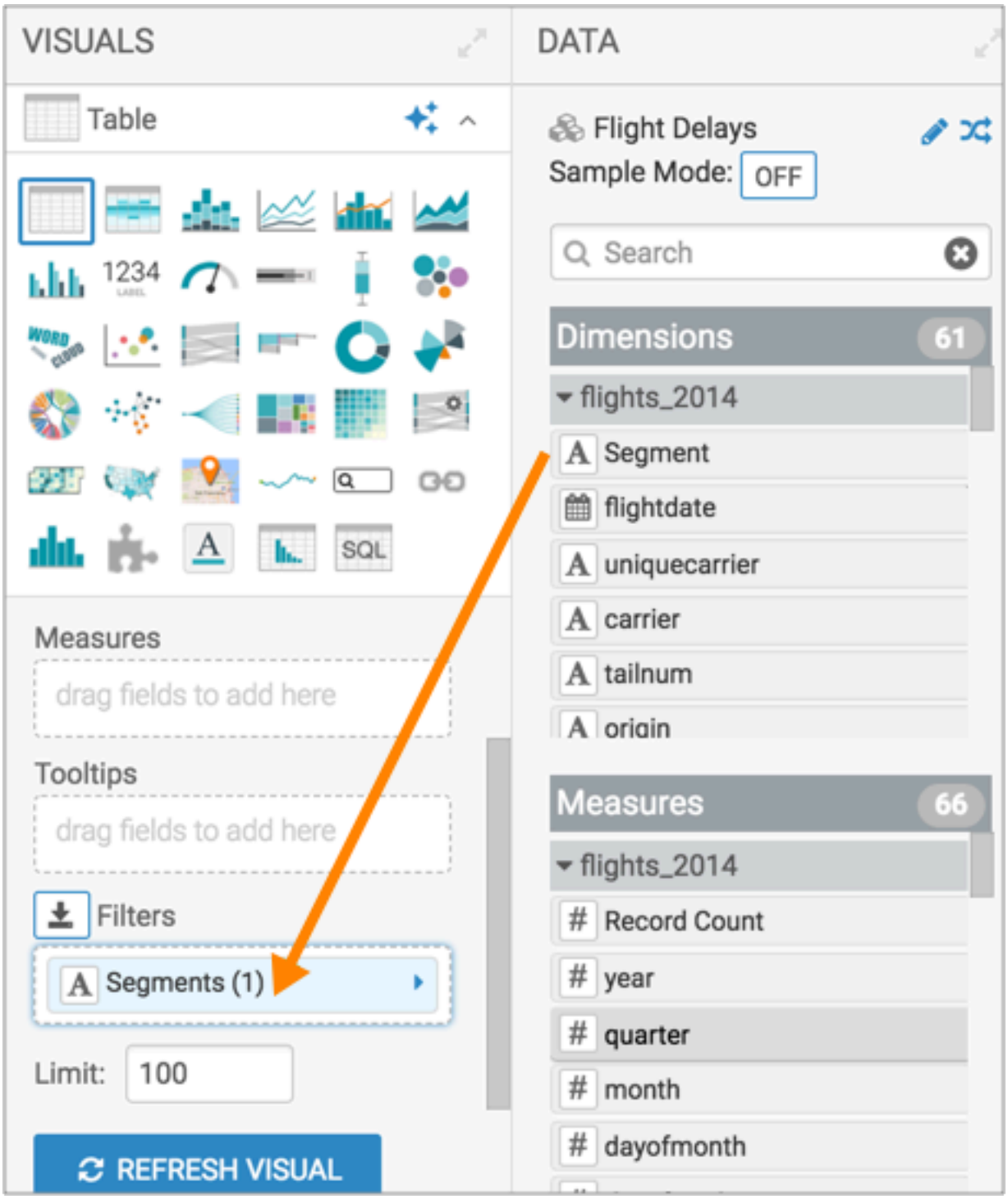

The segments (and the segment group) that you defined earlier appear in the Segments menu.

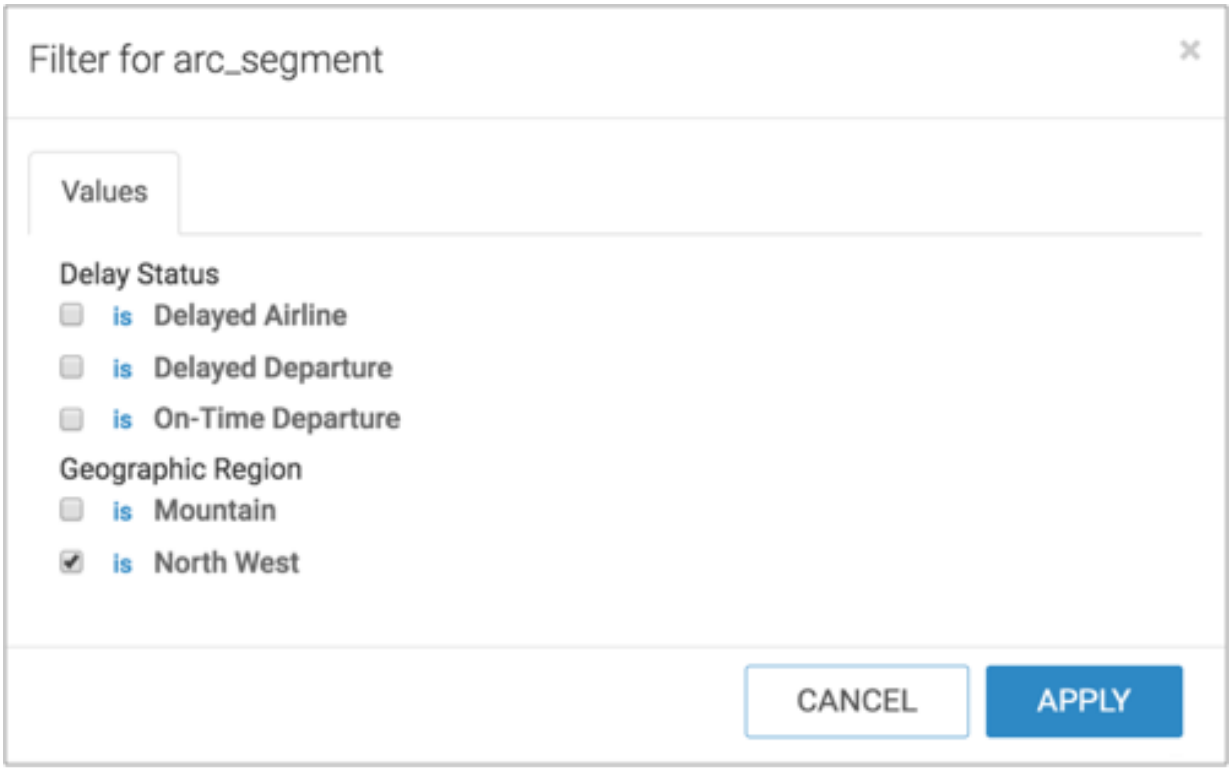
Filter for arc_segment
Values
Delay Status
is Delayed Airline
is Delayed Departure
is On-Time Departure
Geographic Region
is Mountain
is North West
CANCEL
APPLY

2. Select one of the segments, and click REFRESH VISUAL.

   In this example, North West has been selected.

   The visual only shows the rows that match the segment criteria. In this example, the states WA and OR.

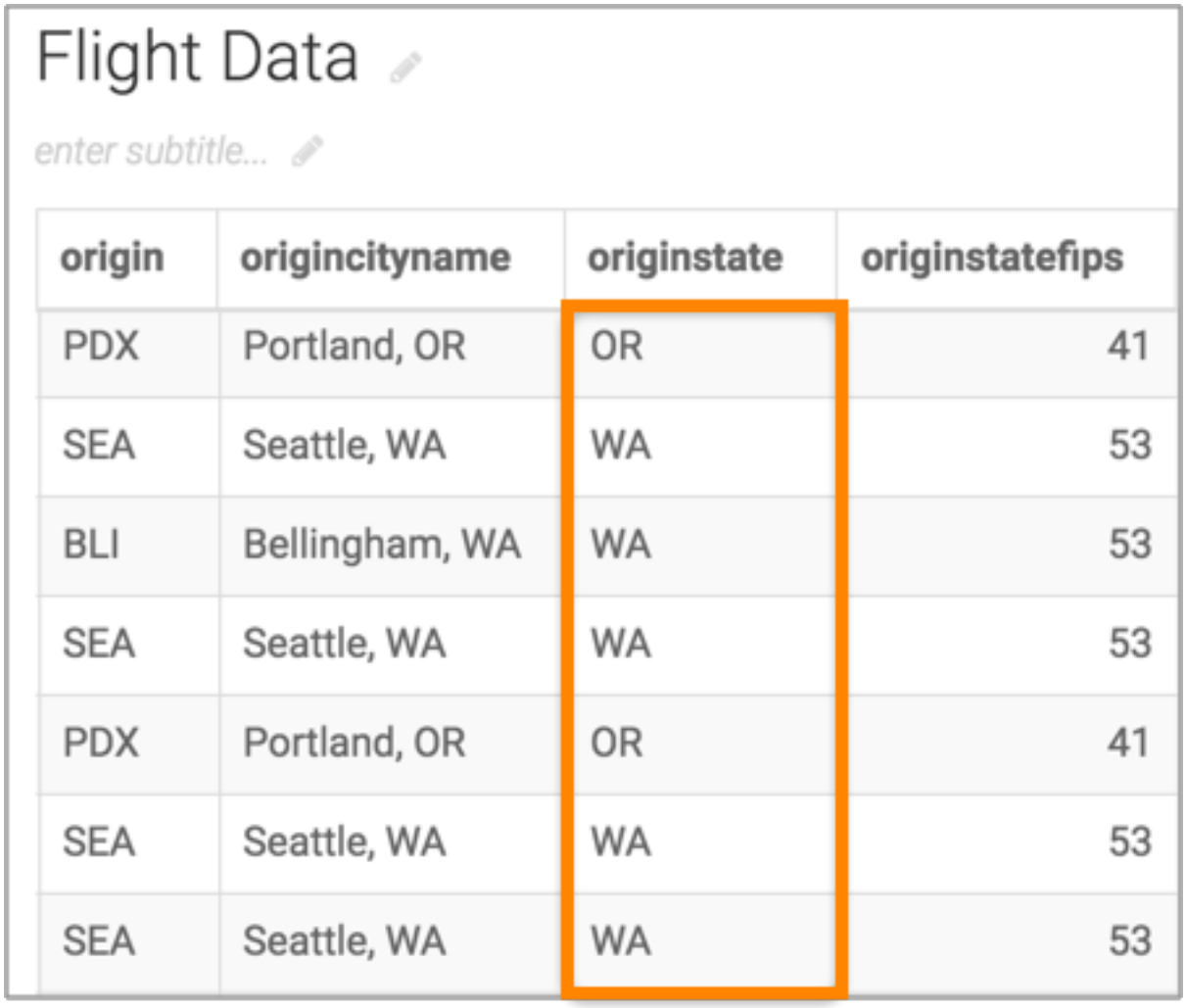

origin	origincityname	originstate	originstatefips
PDX	Portland, OR	OR	41
SEA	Seattle, WA	WA	53
BLI	Bellingham, WA	WA	53
SEA	Seattle, WA	WA	53
PDX	Portland, OR	OR	41
SEA	Seattle, WA	WA	53
SEA	Seattle, WA	WA	53

Related Information

Creating segments

Creating segments from filter definitions

Cloudera Data Visualization 7.2.9

Starting Cloudera Data Visualization in Cloudera Data Science Workbench

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Deploying a Cloudera Data Visualization application in Cloudera Data Science Workbench.....4

Creating a Cloudera Data Science Workbench project with Cloudera Data Visualization Runtime.....4
Creating a Cloudera Data Visualization application in Cloudera Data Science Workbench.....4
Starting Cloudera Data Visualization in Cloudera Data Science Workbench.....5

Using Cloudera Data Visualization in air-gapped Cloudera Data Science Workbench deployment..... 6

Deploying a Cloudera Data Visualization application in Cloudera Data Science Workbench

Learn how to deploy Cloudera Data Visualization in Cloudera Data Science Workbench.

Creating a Cloudera Data Science Workbench project with Cloudera Data Visualization Runtime

Learn how to create a Cloudera Data Science Workbench project with a Cloudera Data Visualization Runtime as the default runtime.

Procedure

1. Click Projects on the left sidebar of Cloudera Data Science Workbench.
2. Click New Project on the Projects page.
3. Enter a project name.
4. Select the visibility of the project.
5. Under Initial Setup, you can either create a blank project, or select a source for your project files.
6. Click Create Project.

   The created project uses Cloudera ML Runtimes that are managed in the Runtime Catalog.

What to do next

After the project is created, you can start creating an application. You can select which runtime edition you want to use for the application.

Creating a Cloudera Data Visualization application in Cloudera Data Science Workbench

Learn how to create a Cloudera Data Visualization application in Cloudera Data Science Workbench to help you visualize and interact with your data insights. This integration allows for seamless data visualization, exploration, and reporting within the same platform.

About this task

The following steps will guide you on how to create a Cloudera Data Visualization application within Cloudera Data Science Workbench.

Important:

Each Cloudera Data Visualization project can host only one standalone Cloudera Data Visualization application. Since applications share the same metadata and logs, any changes made by one application can overwrite those made by another.

Procedure

1. Navigate to the Overview page of your Cloudera Data Science Workbench project.
2. On the left sidebar, click Applications.
3. Click New Application.

4. Provide the following details for your new application:
   - Name – Enter a name for the application.
   - Subdomain – Enter a subdomain that will be used to construct the URL for the web application. Esure it only contains URL-friendly characters.
   - Description – Add a description for the application.
   - Enable Unauthenticated Access – Mark the checkbox if you want to allow unauthenticated access to your application. You can also update this setting later on the Settings page of the application.

     Note: To create public applications on an ML workspace, an Admin user needs to turn on the feature flag in Admin Security by selecting Allow applications to be configured with unauthenticated access. For more information on working with public applications, see *Application Limitations* in the Cloudera Data Science Workbench documentation.
   - Script – Select the path to the startup script.

     Note: Use the script located at: /opt/vizapps/tools/arcviz/startup_app.py
   - Runtime
     - Editor – Workbench
     - Kernel – Select Cloudera Data Visualization for the kernel supported by the Runtime variant of the Cloudera Data Science Workbench project.
     - Edition – Select the edition of the Runtime variant you want to use for your application.

       Note: The selected edition determines the version of the Runtime variant.

     All available runtimes are listed in the Cloudera Data Science Workbench Runtime Catalog. For more information about ML Runtimes, see Managing ML Runtimes and Using Runtime Catalog.

     Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.
5. Click Create Application.

Results

After a few minutes, the application status will change from Starting to Running on the Applications page. Your CDV application is now fully operational.

You can restart, stop, or delete a Cloudera Data Science Workbench application from the supplemental menu of the application. If you want to make changes to the application, navigate to Application Details Settings .

What to do next

Start Cloudera Data Visualization in Cloudera Data Science Workbench.

Related Information

Application Limitations

Starting Cloudera Data Visualization in Cloudera Data Science Workbench

Learn how to start the Cloudera Data Visualization application you have created in Cloudera Data Science Workbench.

Procedure

1. On the Applications page, click the name of your Cloudera Data Visualization application to access the login interface.

2. Log in to Cloudera Data Visualization by entering your username and password. Use your workload credentials.

   If you want to log in as an administrator, you can use the following default credentials:

   - username: vizapps_admin
   - password: vizapps_admin

   When using the default credentials to log in to Cloudera Data Visualization, you are prompted to change your password at the first login.

   Important: If you use the default credentials, security issues may arise. Cloudera recommends that you change the default username and password.

   SSO authentication is disabled by default. See Authentication for information on how to permit user authentication with the Cloudera Data Science Workbench login credentials and log users in automatically.

   After logging in, you land on the homepage view of Cloudera Data Visualization. Here you can explore some sample dashboards or access the in-tool Get Started guide for help.

Related Information

User interface overview

Cloudera Data Visualization quickstart

Using Cloudera Data Visualization in air-gapped Cloudera Data Science Workbench deployment

New Cloudera Data Visualization Runtime releases are automatically added to your deployment when an internet connection is available. However, in air-gapped Cloudera Data Science Workbench deployments, if you want to use Cloudera Data Visualization, you need to manually load the specific Cloudera Data Visualization Runtime in the cluster.

Before you begin

Ensure you have the following:

- Cloudera Data Science Workbench 1.9.0 or higher to support runtimes
- Root access to all cluster nodes
- Cloudera Data Science Workbench installed as a parcel
- Admin access on the Cloudera Data Science Workbench cluster
- Proficiency in Docker, SQL, and Linux

Procedure

1. Download the repo-assembly.json file from the ‘artifacts’ directory of the latest Cloudera Data Visualization version.
2. Download the Docker image to an Internet-connected node or your local machine.

   ```
   image_identifier=$(jq -r '.runtimedataviz[0].image_identifier' repo-asse
   mbly.json)
   docker pull "${image_identifier}"
   ```

3. Load the Docker image to all cluster nodes (master and all workers) using the `docker save` and `docker load` commands.

   ```
   docker save -o runtimedataviz.tar <image_name>
   docker load -i runtimedataviz:image_identifier
   ```

4. Verify that the Docker image is available on all nodes and has its original name and tag using the docker images command.

   You will get a summary view of the Docker images, showing details such as the repository name, tag, image ID, creation date, and size.

5. Add the Cloudera Data Visualization image as a custom runtime, using the original Docker image name.

   For example: docker.repository.cloudera.com/cloudera/cdv/runtimedataviz:7.1.2-b53

   For detailed instructions, see Adding New ML Runtimes.

Cloudera Data Visualization 7.2.9

Using Cloudera Data Visualization in Cloudera Data Warehouse

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Deploying Cloudera Data Visualization in Cloudera Data Warehouse.....4

Editing and deleting Cloudera Data Visualization instances in Cloudera Data Warehouse.....7

Upgrading Cloudera Data Visualization in Cloudera Data Warehouse.....9

Backing up and restoring Cloudera Data Visualization applications in Cloudera Data Warehouse.....11

Deploying Cloudera Data Visualization in Cloudera Data Warehouse

Learn how to create and launch a Cloudera Data Visualization instance within the Cloudera Data Warehouse data service. You can use Cloudera Data Visualization with Cloudera Data Warehouse in both Cloudera Public Cloud and Cloudera Private Cloud to explore and visualize data stored in database catalogs providing data-driven insights throughout the entire data lifecycle.

About this task

If you want to create visuals based data stored in Cloudera Data Warehouse, you have to create a Cloudera Data Visualization instance and connect it to Hive or Impala Virtual Warehouse(s).

Cloudera Data Visualization is not tied to a particular Virtual Warehouse (VW). You can access data for your visualizations from multiple Data Catalogs using multiple Hive or Impala Virtual Warehouses in various environments. With multiple Cloudera Data Visualization instances attached to an environment, you can connect to different data sets, create dashboards for different groups and share your visualizations with different users.

Note: When you delete a Virtual Warehouse, your visual artifacts remain intact as long as the Cloudera Data Visualization instance is not deleted.

Before you begin

- You are logged into the Cloudera web interface and you have opened the Cloudera Data Warehouse service.
- You have DWAdmin role in Cloudera Data Warehouse.
- You have activated your environment.
- You have a Hive/Impala warehouse in running state.
- If you are using Cloudera Data Visualization with Cloudera Data Warehouse in Cloudera Private Cloud:

  You have an admin group created in Management Console User Management . To log in using LDAP, select the Sync Groups on Login option from Management Console Administration Authentication , so that the associated groups can be imported when you log in to Cloudera Data Visualization.

Procedure

1. In Cloudera Data Warehouse, click Data Visualization in the left navigation panel.

   A list of existing Cloudera Data Visualization instances appears, if there are any.
2. Click ADD NEW to create a new instance.

   If you are creating the first Cloudera Data Visualization instance in Cloudera Data Warehouse, click CREATE.

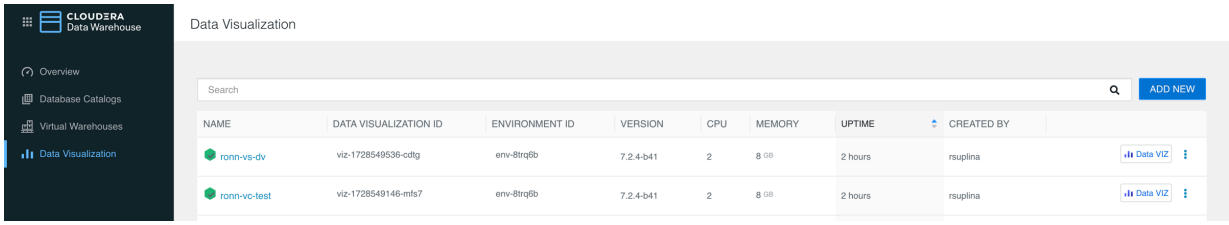

3. Provide the following information in the **New Data Visualization** modal window:

   Mandatory fields are marked with an asterisk.

   a) Name* – Specify a name for the instance.
   b) Environments* – Select the environment that you want Cloudera Data Visualization to be connected to.
   c) User Groups – Add user groups to allow user access to Cloudera Data Visualization for selected groups. If no group is added, all Cloudera users will have non-admin access.
   d) Admin Groups* – Add admin groups to allow configuration access to Cloudera Data Visualization for selected groups.

      For more information on Cloudera Data Visualization permission granularity, see the *Security model*.
   e) Tagging – Enter keys and values to apply tags to your resources for organizing them into a taxonomy.

      Note: This field is not available in Cloudera Data Warehouse on Cloudera Private Cloud.

   f) Resource Template – Select the Cloudera Data Warehouse resource template from the drop-down menu.
      - Default resources
      - Medium resources
      - Large resources

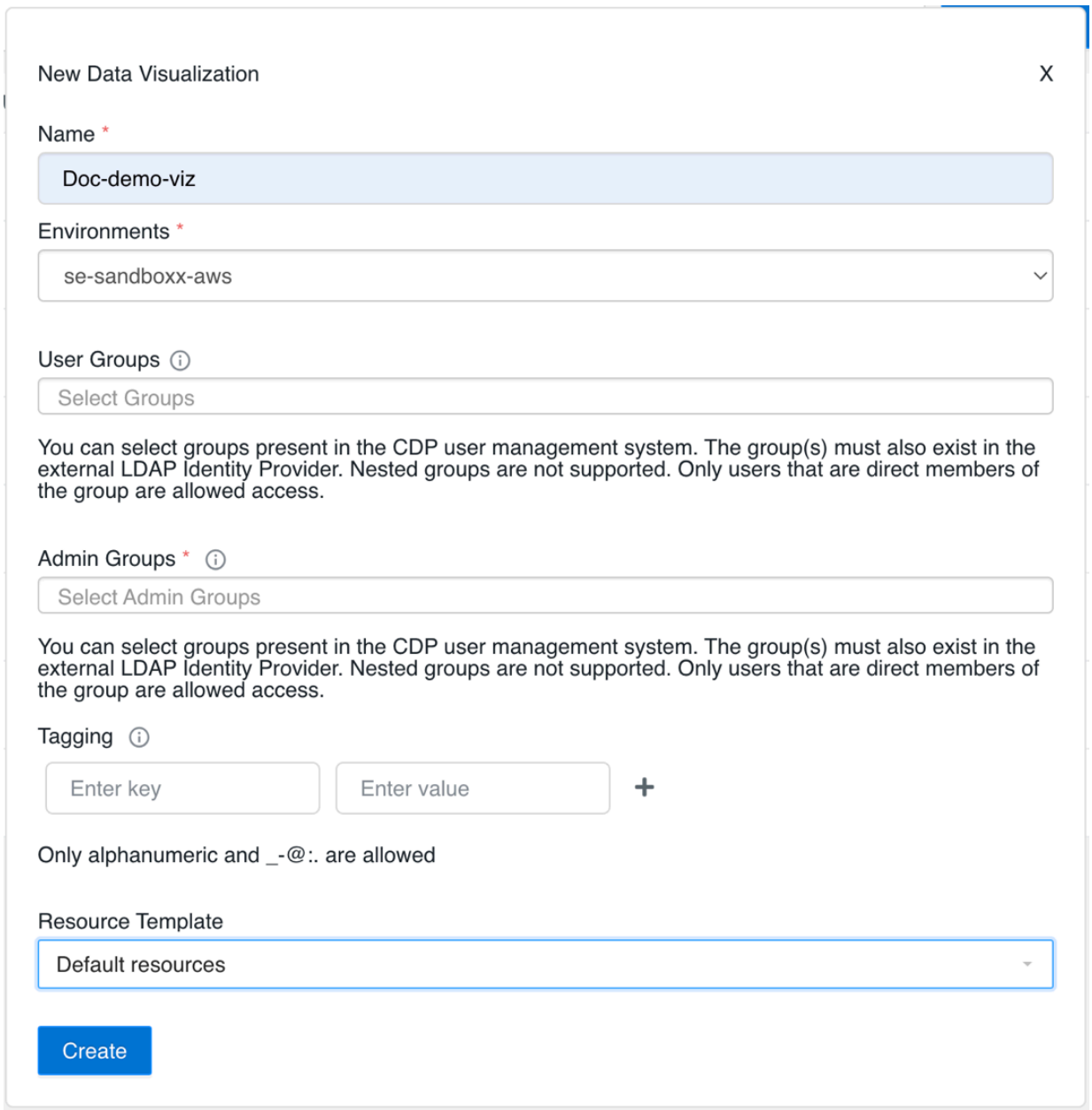

4. Click CREATE.

   Instance creation starts. Wait until the Cloudera Data Visualization instance is in running state.

5. You can find the list of Cloudera Data Visualization instances and environments appears under the Data Visualization menu that you can open from the left navigation panel.

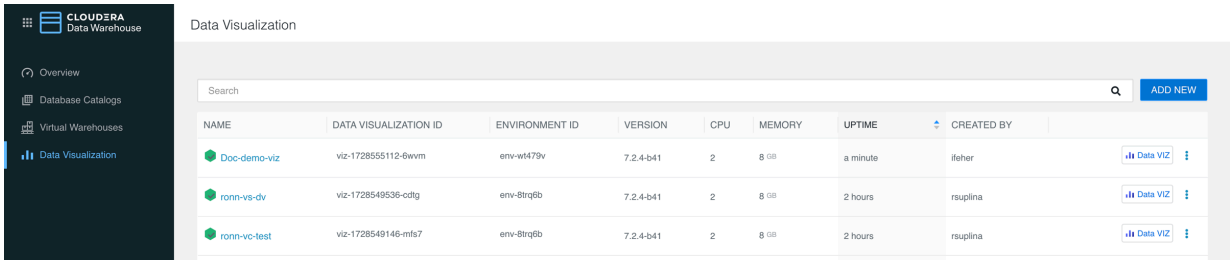

6. Select one from the list of running Cloudera Data Visualization instances and click

to start Cloudera Data Visualization.

   Alternatively, you can click the launch/options menu (three dots) at the end of the row and click Open Data Visualization in the pop-up menu.

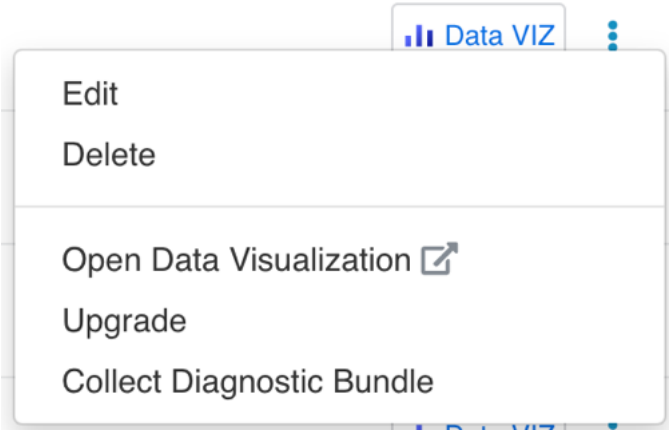

   SSO authentication is enabled by default both in Cloudera Public Cloud and Cloudera Private Cloud. If you are logged in to the Control Plane, you will be logged into Cloudera Data Visualization automatically. If not, log into the Control Plane using your LDAP credentials. Cloudera Data Visualization opens in a new browser tab and you land on the Cloudera Data Visualization homepage, where you can explore sample dashboards and access the in-tool *Get Started guide* for help.

   New users logging in to Cloudera Data Visualization are automatically added to the viz_guest_group group. You can assign the System Admin role to this group to grant new users the permission to create new connections. For more information, see *Assigning roles to users*.

Related Information

Security model

Editing and deleting Cloudera Data Visualization instances in Cloudera Data Warehouse

Learn how to edit and delete Cloudera Data Visualization instances in Cloudera Data Warehouse data service.

Before you begin

- You are logged into Cloudera Data Warehouse
- You have DWAdmin role in Cloudera Data Warehouse

Procedure

1. In Cloudera Data Warehouse, click Data Visualization in the left navigation panel.

   A list of available Cloudera Data Visualization instances appears.
2. In one of the Cloudera Data Visualization instance rows, click the launch/options menu (three dots).

3. To change instance details, click Edit.

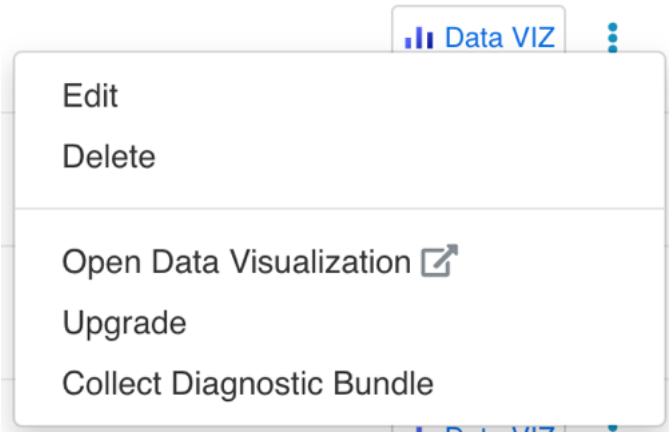

The instance overview page opens and you can make changes to general details and other properties.

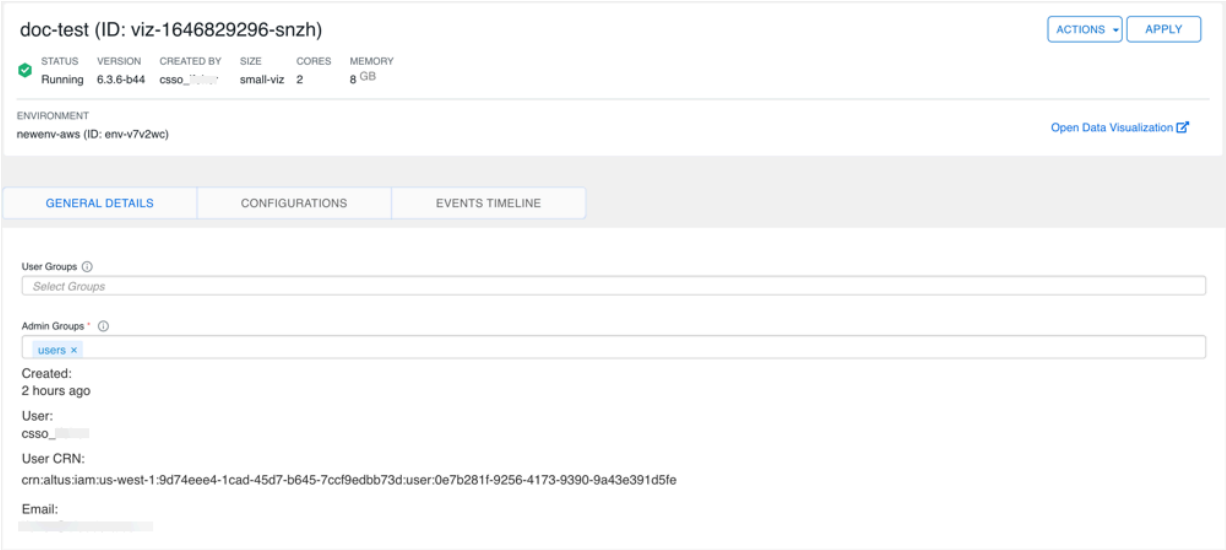

Click APPLY to save the changes.

4. To delete a Cloudera Data Visualization instance, click ACTIONS Delete .

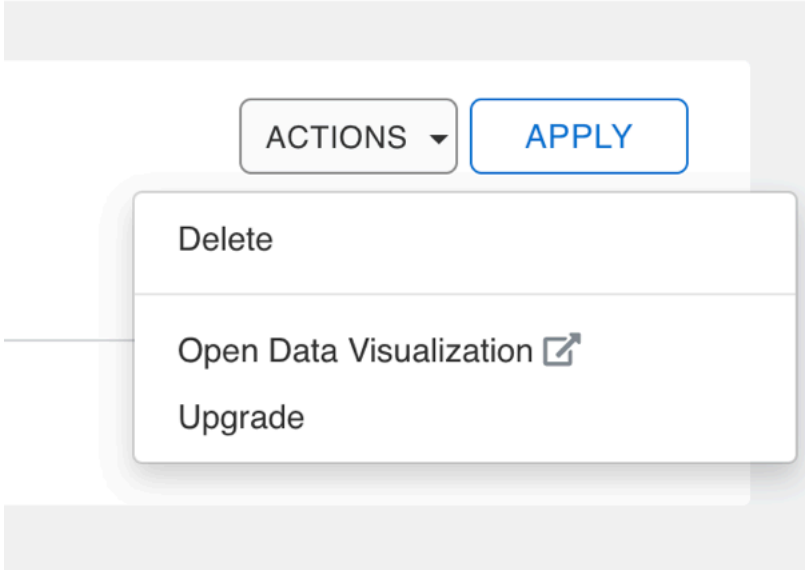

Alternatively, you can click the launch/options menu (three dots) at the end of a Cloudera Data Visualization instance row and select Delete.

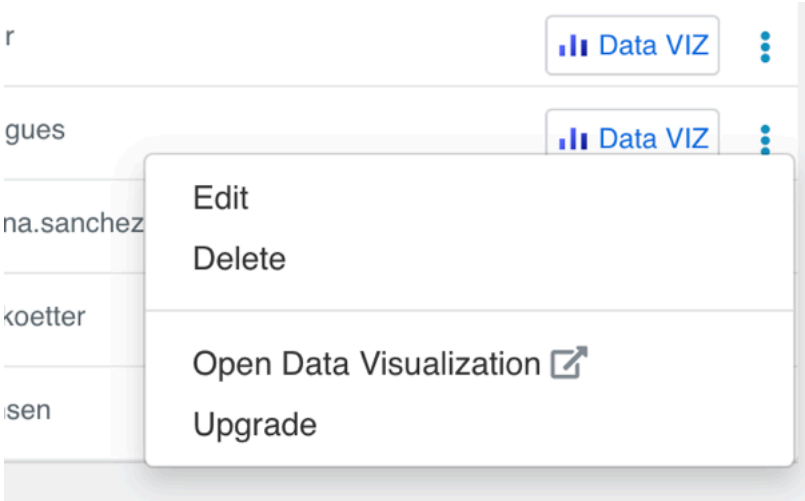

Upgrading Cloudera Data Visualization in Cloudera Data Warehouse

Learn how to upgrade your Cloudera Data Visualization instance if a later version is available in Cloudera Data Warehouse.

About this task

Cloudera Data Visualization upgrades in Cloudera Data Warehouse are centrally managed and can be initiated through the Management Console. This ensures a streamlined and controlled process for keeping your Cloudera Data Visualization deployments up to date.

You can start the upgrade manually when a new version becomes available. Once you start the upgrade, it gets completed automatically, and roll back to a previous Cloudera Data Visualization version is not supported. If you encounter issues after an upgrade, you must revert to a backed-up version of the database. Ensure you have created a

reliable backup before initiating the upgrade to safeguard against potential issues. For more information, see *Backing up and restoring Cloudera Data Visualization applications in Cloudera Data Warehouse*.

Before you begin

Ensure that:

- You are logged into Cloudera
- You can access the environment of the Cloudera Data Visualization to upgrade
- You have DWAdmin role

Procedure

1. In Cloudera Data Warehouse, click Data Visualization in the left navigation panel.

   A list of Cloudera Data Visualization instances appears. In Cloudera Public Cloud, if a newer version of Cloudera Data Visualization is available for any listed instance, the Upgrade option is shown in the relevant row.

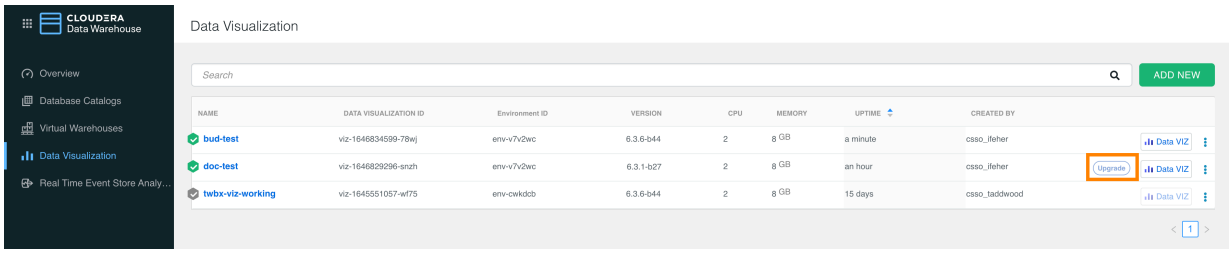

2. Find the instance from the list and click Upgrade to start the upgrade.

   Alternatively, you can click the Options menu (three dots) in the row of the Cloudera Data Visualization instance that you want to upgrade and click Upgrade in the pop-up menu.

   Note: In Cloudera Private Cloud, the Upgrade button is not available, so use the Options menu to start the upgrade process.

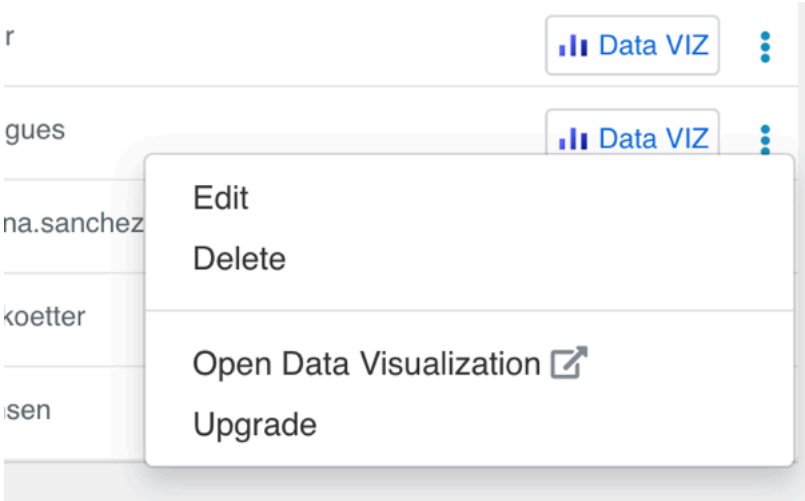

Related Information

Backing up and restoring Cloudera Data Visualization applications in Cloudera Data Warehouse

Backing up and restoring Cloudera Data Visualization applications in Cloudera Data Warehouse

You can ensure the safety of your Cloudera Data Visualization applications in Cloudera Data Warehouse by implementing a proactive backup strategy. The Cloudera Data Warehouse backup/restore feature saves your environment parameters, enabling the recreation of your environment with the same settings, URL, and connection strings used previously.

For more information on the backup/restore feature, see Backing up and restoring Cloudera Data Warehouse.

Backup procedure

A backup is a preventive measure to create a copy of your data to ensure that if data loss or system failure occurs, you can recover it by using the backup copy. This way you can mitigate risks associated with upgrades and return to a known state if unintended consequences arise.

To save the charts and dashboards you have created, it is important to perform data backup. You can use Kubernetes command-line tools, such as k9s or kubectl, to create backups for your Cloudera Data Visualization applications in an AWS environment in Cloudera Data Warehouse.

For detailed instructions, see Backing up Cloudera Data Visualization applications.

Restore procedure

You can use the restore procedure when you encounter issues that cannot be resolved, and you need to bring the system back to a stable state. You can only restore the system, if you have a previously backed up Cloudera Data Visualization instance.

In Cloudera Data Warehouse, you can use the environment-level automated process to restore the environment, the deployed Database Catalog, Virtual Warehouse, and Cloudera Data Visualization entities. If a Cloudera Data Visualization deployment is not present on the cluster, but the backup file contains it, Cloudera Data Visualization will be restored to the cluster. If such an entity is already deployed, no data restoration or configuration updates will take place. For more information, see Cloudera Data Visualization automatic restoration.

For detailed instructions, see Restoring Cloudera Data Visualization.

Cloudera Data Visualization 7.2.9

Starting Cloudera Data Visualization in Cloudera AI

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Deploying a Cloudera Data Visualization application in Cloudera AI.....4
Creating a Cloudera AI project with Cloudera Data Visualization Runtime.....4
Adding Cloudera Data Visualization Runtime to a Cloudera AI project.....4
Creating a Cloudera Data Visualization application in Cloudera AI.....5

Using Data Discovery and Visualization in Cloudera AI.....6

Upgrading Cloudera Data Visualization in Cloudera AI/Cloudera Data Science Workbench.....7

Restoring a Cloudera Data Visualization version in Cloudera AI/Cloudera Data Science Workbench.....8

Deploying a Cloudera Data Visualization application in Cloudera AI

Learn how to deploy Cloudera Data Visualization in Cloudera AI.

Creating a Cloudera AI project with Cloudera Data Visualization Runtime

Learn how to create a Cloudera AI project with Cloudera Data Visualization Runtime as the default runtime.

About this task

If you know that your project is going to be running Cloudera Data Visualization, you can add the Cloudera Data Visualization Runtime when setting up the project.

Procedure

1. Click Projects on the left sidebar of Cloudera AI data service.
2. Click New Project.
3. Enter a project name.
4. You can add a description for your project.
5. You can also select the visibility of the project.
6. Under Initial Setup, you can either create a blank project, or select a source for your project files.
7. Set a Cloudera Data Visualization Runtime for your project in the Runtime setup section.

   You can use the Basic view to select the Cloudera Data Visualization Runtime and add all of its Standard Edition ML Runtimes for that kernel. In the Advanced view, you can add ML Runtimes based on more detailed Editor, Kernel, Edition, and Version criteria.
8. Click Create Project.

What to do next

After the project is created, you can start creating your application. If you added Cloudera Data Visualization Runtime as the only runtime during project set-up, it will be the only available runtime when creating your application. You can select which edition you want to use for the application.

Adding Cloudera Data Visualization Runtime to a Cloudera AI project

Learn how to add a Cloudera Data Visualization Runtime to an existing Cloudera AI project.

About this task

You need to add a Cloudera Data Visualization Runtime to your project if the workspace of your project is not set to use this Cloudera Data Visualization Runtime by default.

Procedure

1. Click Projects on the left sidebar of the Cloudera AI data service.
2. Click the project where you want to add Cloudera Data Visualization Runtime.
3. Open Project Settings from the left navigation bar and click the Runtime/Engine tab.
4. Click Add Runtime.

   The Add new runtime to project modal window opens.
5. Select Cloudera Data Visualization from the Kernel drop-down list.

6. Select the edition of the Cloudera Data Visualization you want to add to your project.

   The selected edition automatically defines the version of the runtime variant.

   Note: New Cloudera Data Visualization Runtime versions are automatically added to the list of available editions, if internet connection is available.

7. Click Submit.

Results

You should now be able to select the Cloudera Data Visualization Runtime when creating a Cloudera Data Visualization application.

What to do next

Create a Cloudera Data Visualization application.

Creating a Cloudera Data Visualization application in Cloudera AI

Learn how to create a Cloudera Data Visualization application in Cloudera AI to help you visualize and interact with your data insights. This integration allows for seamless visualization of ML model outputs, data exploration, and reporting within the same platform.

About this task

The following steps will guide you on how to create a Cloudera Data Visualization application in a Cloudera AI project.

Important:

Each Cloudera AI project can host only one standalone Cloudera Data Visualization application. Since applications share the same metadata and logs, any changes made by one application can overwrite those made by another.

Before you begin

Ensure that a Cloudera Data Visualization Runtime is available in the Cloudera AI project where you want to create the Cloudera Data Visualization application.

For more information about ML Runtimes, see Managing ML Runtimes and Using Runtime Catalog.

Procedure

1. Navigate to the Overview page of your Cloudera AI project.
2. On the left sidebar, click Applications.
3. Click New Application.

4. Provide the following details for your new application:
   - Name – Enter a name for the application.
   - Subdomain – Enter a subdomain that will be used to construct the URL for the web application. Ensure it only contains URL-friendly characters.
   - Description – Add a description for the application.
   - Script – Select the path to the startup script.

     Note: Use the script located at: /opt/vizapps/tools/arcviz/startup_app.py

   - Runtime – Configure the Runtime.
     - Editor – Workbench
     - Kernel – Select Cloudera Data Visualization for the kernel supported by the Runtime variant of the Cloudera AI project.
     - Edition – Select the edition of the Runtime variant you want to use for your application.

       Note: The selected edition determines the version of the Runtime variant.
5. Click Create Application.

Results

After a few minutes, the application status will change from Starting to Running on the Applications page. Your Cloudera Data Visualization application is now fully operational.

You can restart, stop, or delete a Cloudera AI application from the supplemental menu. If you want to make changes to the application, navigate to Application Details Settings .

What to do next

Start Cloudera Data Visualization.

Using Data Discovery and Visualization in Cloudera AI

Exploratory data science and visualization makes it simple for data scientists to get started on a data science project. With the Data Discovery and Visualization application, you can automatically discover the data sources available to you from within the standard Cloudera AI user interface and provide various features for exploratory data visualization and analysis.

From the Data tab in Cloudera AI you can:

- Connect to data sources that are available in your project
- Explore the data with SQL to understand its basic shape and characteristics
- Create named datasets that you can reference later
- Create visualizations of the data to understand its properties
- Create dashboards that you can share with your team

For prerequisites and more information on how to work with data discovery and visualization in Cloudera AI, see the Exploratory Data Science and Visualization documentation

Upgrading Cloudera Data Visualization in Cloudera AI/Cloudera Data Science Workbench

Cloudera recommends following the Cloudera Data Visualization release cadence by upgrading to the latest version as it becomes available. This practice ensures that your Cloudera Data Visualization instances benefit from the latest new features, security enhancements, and bug fixes.

About this task

This guide outlines the available options performing an upgrade and the steps of the upgrade process. If you need to downgrade, follow the Restoring Cloudera Data Visualization in Cloudera AI/Cloudera Data Science Workbench procedure using the backup you created earlier.

Before you begin

- Before initiating the upgrade, review the Known issues and limitations in Cloudera Data Visualization in the Release Notes for both the currently running version and the version to which you plan to upgrade to. This helps you identify potential challenges, limitations, or specific actions required for a successful upgrade.
- If you are using an external database like MySQL, MariaDB, Postgres, or Oracle, ensure you back up the database and restore it as a new database that the new Cloudera Data Visualization instance can connect to.

Procedure

1. Stop the application.

   On the Applications page in Cloudera AI/CDSW, click Actions Stop to stop the selected Cloudera Data Visualization application.
2. Back up the SQLite metadata database.
   a) Navigate to your project's Overview page and click New Session.
   b) When your session is running, click >_ Terminal Access above the session log pane to open a terminal window.

      The default working directory is /home/cdsw, where all your project files reside.
   c) Navigate to /home/cdsw/.arc.

      This directory contains the Cloudera Data Visualization logs, files, settings, and the arcviz.db database.
   d) Run the following command to create a copy of your current Cloudera Data Visualization metadata.

      sqlite3 arcviz.db “.backup arcviz.db-BACKUP-*[***VERSION***]*”

      You can use this backup in case you need to revert to a previous Cloudera Data Visualization version.
3. Navigate to the Application Settings page.
4. Locate the runtime configuration option, where you can select the desired runtime for your application.
5. Select the runtime you want to set for your application from the available options.
6. Click Update Application to save the change.

   The application will restart automatically with the newly selected runtime.

   Note: Allow some time for the restart process to complete.

Restoring a Cloudera Data Visualization version in Cloudera AI/Cloudera Data Science Workbench

Restoring a previous version means reverting your Cloudera Data Visualization application to an earlier version. You can restore your system to a known stable state, especially when issues arise during an upgrade or if the new version introduces major problems that impact functionality, and you need to return to the previous Cloudera Data Visualization version to maintain system functionality. If you decide to downgrade, follow the steps outlined below.

Before you begin

To restore a previous version, you should have either connected your system to an external database (for example: MySQL, MariaDB, Postgres, Oracle) or created a backup of the SQLite metadata database.

Note:

If you are using an external database like MySQL, MariaDB, Postgres, or Oracle, you must back up the database before the upgrade and restore it as a new database for the Cloudera Data Visualization instance to connect to after the downgrade.

Procedure

1. Stop the application.

   On the Applications page in Cloudera AI/Cloudera Data Science Workbench, click Actions Stop to stop the selected Cloudera Data Visualization application.
2. Back up the SQLite metadata database.
   a) Navigate to your project's Overview page and click New Session.
   b) When your session is running, click >_ Terminal Access above the session log pane to open a terminal window.

      The default working directory is /home/cdsw, where all your project files reside.
   c) Navigate to /home/cdsw/.arc.

      This directory contains the Cloudera Data Visualization logs, files, settings, and the arcviz.db database.
   d) Run the following command to create a copy of your current Cloudera Data Visualization metadata.

      sqlite3 arcviz.db “.backup arcviz.db-BACKUP-*[***VERSION***]*”

      You can use this backup when reverting to a previous Cloudera Data Visualization version.
3. Create a new Cloudera Data Visualization application in your Cloudera AI/Cloudera Data Science Workbench project.
4. Copy the backed-up SQLite database to /home/cdsw/.arc/ in the new project.
5. Once the backup is restored, start the new Cloudera Data Visualization application.

Cloudera Data Visualization 7.2.9

CDP Data Visualization Top Tasks

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Deploying Cloudera Data Visualization.....4
Deploying Cloudera Data Visualization in Cloudera Data Warehouse.....4
Deploying a Cloudera Data Visualization application in Cloudera AI.....8
Creating a Cloudera AI project with Cloudera Data Visualization Runtime.....8
Adding Cloudera Data Visualization Runtime to a Cloudera AI project.....9
Creating a Cloudera Data Visualization application in Cloudera AI.....9
Deploying a Cloudera Data Visualization application in Cloudera Data Science Workbench.....10
Creating a Cloudera Data Science Workbench project with Cloudera Data Visualization Runtime.....10
Creating a Cloudera Data Visualization application in Cloudera Data Science Workbench.....11
Starting Cloudera Data Visualization in Cloudera Data Science Workbench.....12

Connecting to a data source in Cloudera Data Visualization.....13
Creating a Cloudera AI data connection to Impala.....13
Creating a Cloudera AI data connection to a Hive data warehouse.....20
Creating a Cloudera Data Warehouse data connection in Cloudera Data Visualization.....26
Creating a Cloudera Data Science Workbench data connection to a data warehouse.....29

Creating a visual.....31

Deploying Cloudera Data Visualization

The first step to start using Cloudera Data Visualization is to deploy it on your chosen Cloudera platform. Cloudera Data Visualization provides consistent visualization capabilities across various Cloudera deployments and services. Once your Cloudera Data Visualization instance is up and running, you can start creating visuals, build customized dashboards, and develop analytical applications to enable users to explore data across the whole data lifecycle.

You can log into Cloudera Data Visualization using your Cloudera credentials and start analyzing data that you have access to. In Cloudera Public Cloud, Cloudera Data Visualization is included as a core capability in Cloudera Data Warehouse and Cloudera AI both for Azure and AWS. Cloudera Data Visualization is also available in Cloudera Private Cloud for Cloudera Data Warehouse and Cloudera AI, and it can be used in Cloudera Data Science Workbench, certified with CDH, HDP and Cloudera Private Cloud Base.

Deploying Cloudera Data Visualization in Cloudera Data Warehouse

Learn how to create and launch a Cloudera Data Visualization instance within the Cloudera Data Warehouse data service. You can use Cloudera Data Visualization with Cloudera Data Warehouse in both Cloudera Public Cloud and Cloudera Private Cloud to explore and visualize data stored in database catalogs providing data-driven insights throughout the entire data lifecycle.

About this task

If you want to create visuals based data stored in Cloudera Data Warehouse, you have to create a Cloudera Data Visualization instance and connect it to Hive or Impala Virtual Warehouse(s).

Cloudera Data Visualization is not tied to a particular Virtual Warehouse (VW). You can access data for your visualizations from multiple Data Catalogs using multiple Hive or Impala Virtual Warehouses in various environments. With multiple Cloudera Data Visualization instances attached to an environment, you can connect to different data sets, create dashboards for different groups and share your visualizations with different users.

Note: When you delete a Virtual Warehouse, your visual artifacts remain intact as long as the Cloudera Data Visualization instance is not deleted.

Before you begin

- You are logged into the Cloudera web interface and you have opened the Cloudera Data Warehouse service.
- You have DWAdmin role in Cloudera Data Warehouse.
- You have activated your environment.
- You have a Hive/Impala warehouse in running state.
- If you are using Cloudera Data Visualization with Cloudera Data Warehouse in Cloudera Private Cloud:

  You have an admin group created in Management Console User Management . To log in using LDAP, select the Sync Groups on Login option from Management Console Administration Authentication , so that the associated groups can be imported when you log in to Cloudera Data Visualization.

Procedure

1. In Cloudera Data Warehouse, click Data Visualization in the left navigation panel.

   A list of existing Cloudera Data Visualization instances appears, if there are any.

2. Click ADD NEW to create a new instance.

   If you are creating the first Cloudera Data Visualization instance in Cloudera Data Warehouse, click CREATE.

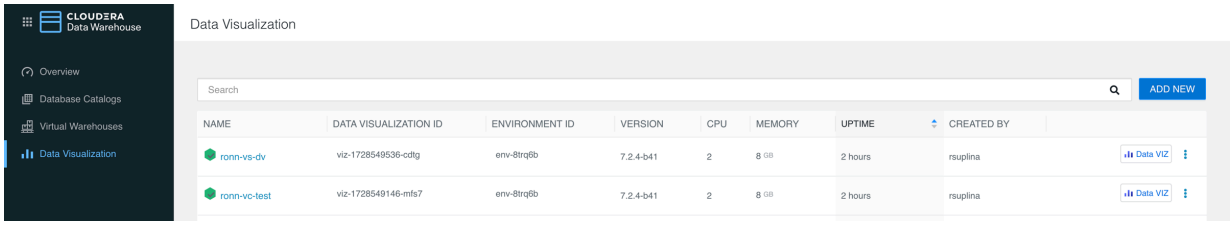

3. Provide the following information in the **New Data Visualization** modal window:

   Mandatory fields are marked with an asterisk.

   a) Name* – Specify a name for the instance.
   b) Environments* – Select the environment that you want Cloudera Data Visualization to be connected to.
   c) User Groups – Add user groups to allow user access to Cloudera Data Visualization for selected groups. If no group is added, all Cloudera users will have non-admin access.
   d) Admin Groups* – Add admin groups to allow configuration access to Cloudera Data Visualization for selected groups.

      For more information on Cloudera Data Visualization permission granularity, see the *Security model*.
   e) Tagging – Enter keys and values to apply tags to your resources for organizing them into a taxonomy.

      Note: This field is not available in Cloudera Data Warehouse on Cloudera Private Cloud.

   f) Resource Template – Select the Cloudera Data Warehouse resource template from the drop-down menu.
      - Default resources
      - Medium resources
      - Large resources

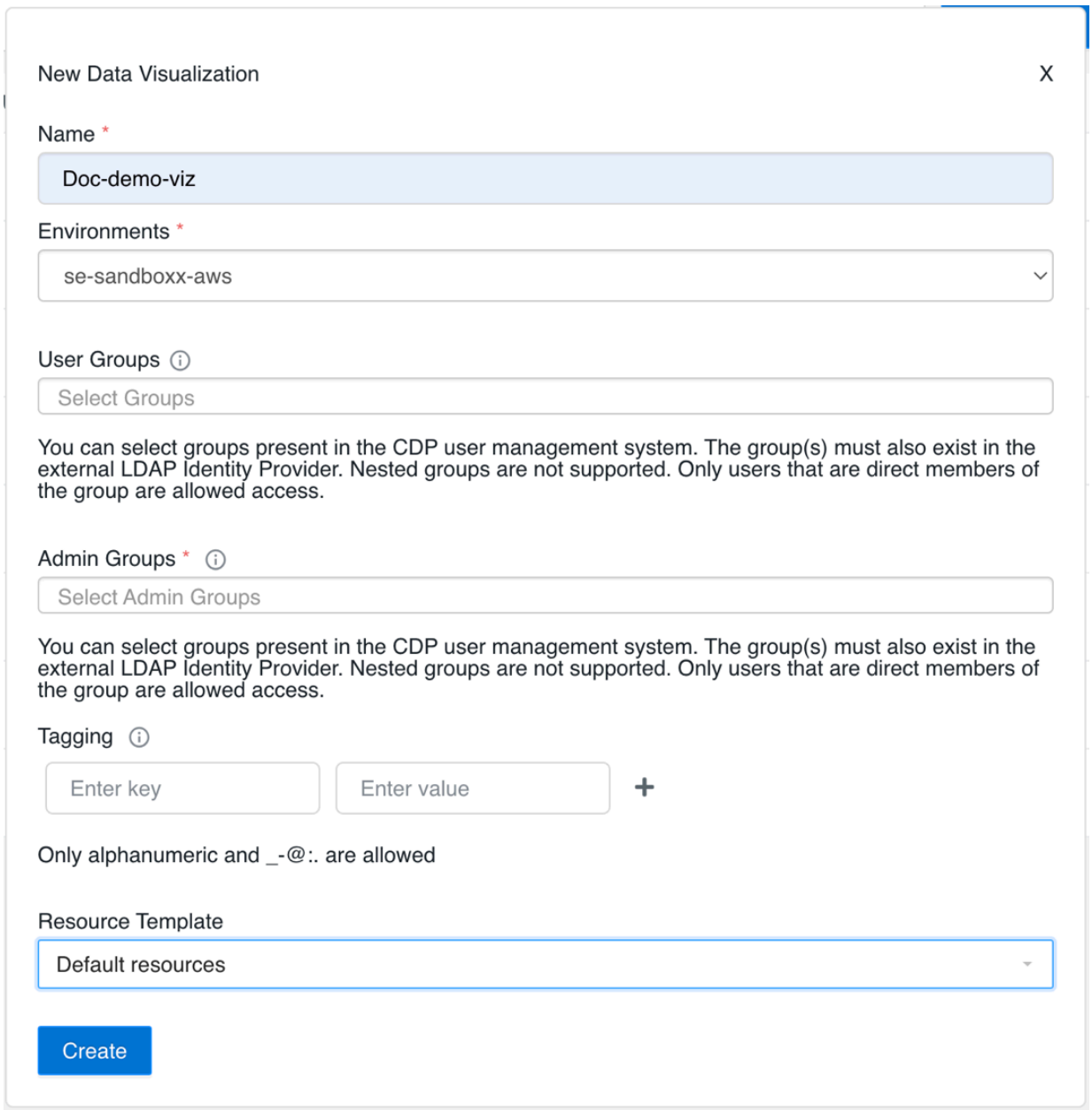

4. Click CREATE.

   Instance creation starts. Wait until the Cloudera Data Visualization instance is in running state.

5. You can find the list of Cloudera Data Visualization instances and environments appears under the Data Visualization menu that you can open from the left navigation panel.

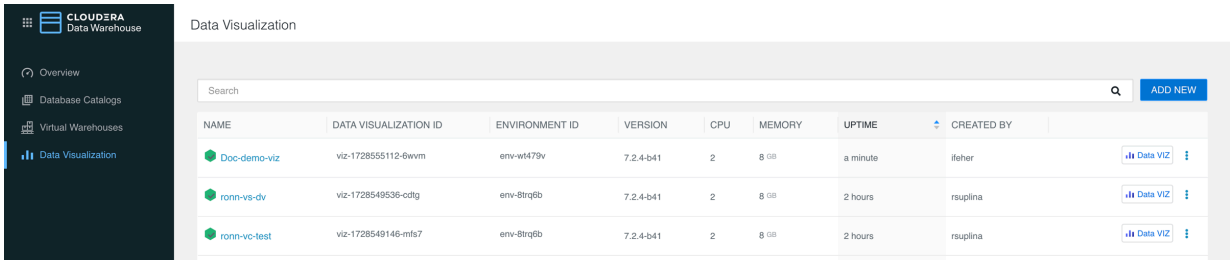

6. Select one from the list of running Cloudera Data Visualization instances and click

to start Cloudera Data Visualization.

   Alternatively, you can click the launch/options menu (three dots) at the end of the row and click Open Data Visualization in the pop-up menu.

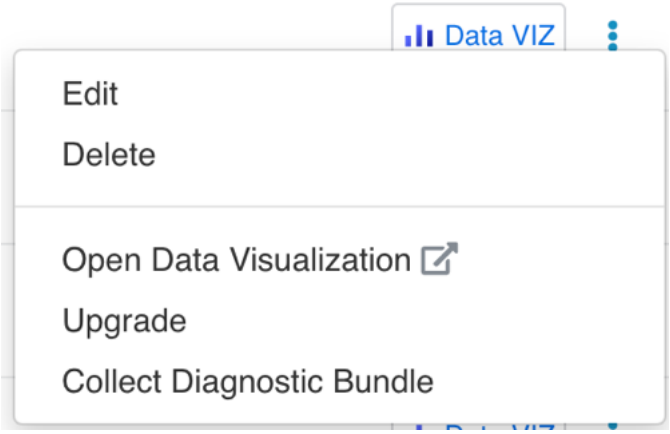

   SSO authentication is enabled by default both in Cloudera Public Cloud and Cloudera Private Cloud. If you are logged in to the Control Plane, you will be logged into Cloudera Data Visualization automatically. If not, log into the Control Plane using your LDAP credentials. Cloudera Data Visualization opens in a new browser tab and you land on the Cloudera Data Visualization homepage, where you can explore sample dashboards and access the in-tool *Get Started guide* for help.

   New users logging in to Cloudera Data Visualization are automatically added to the viz_guest_group group. You can assign the System Admin role to this group to grant new users the permission to create new connections. For more information, see *Assigning roles to users*.

Deploying a Cloudera Data Visualization application in Cloudera AI

Learn how to deploy Cloudera Data Visualization in Cloudera AI.

Creating a Cloudera AI project with Cloudera Data Visualization Runtime

Learn how to create a Cloudera AI project with Cloudera Data Visualization Runtime as the default runtime.

About this task

If you know that your project is going to be running Cloudera Data Visualization, you can add the Cloudera Data Visualization Runtime when setting up the project.

Procedure

1. Click Projects on the left sidebar of Cloudera AI data service.
2. Click New Project.
3. Enter a project name.
4. You can add a description for your project.
5. You can also select the visibility of the project.
6. Under Initial Setup, you can either create a blank project, or select a source for your project files.
7. Set a Cloudera Data Visualization Runtime for your project in the Runtime setup section.

   You can use the Basic view to select the Cloudera Data Visualization Runtime and add all of its Standard Edition ML Runtimes for that kernel. In the Advanced view, you can add ML Runtimes based on more detailed Editor, Kernel, Edition, and Version criteria.

8. Click Create Project.

What to do next

After the project is created, you can start creating your application. If you added Cloudera Data Visualization Runtime as the only runtime during project set-up, it will be the only available runtime when creating your application. You can select which edition you want to use for the application.

Adding Cloudera Data Visualization Runtime to a Cloudera AI project

Learn how to add a Cloudera Data Visualization Runtime to an existing Cloudera AI project.

About this task

You need to add a Cloudera Data Visualization Runtime to your project if the workspace of your project is not set to use this Cloudera Data Visualization Runtime by default.

Procedure

1. Click Projects on the left sidebar of the Cloudera AI data service.
2. Click the project where you want to add Cloudera Data Visualization Runtime.
3. Open Project Settings from the left navigation bar and click the Runtime/Engine tab.
4. Click Add Runtime.

   The Add new runtime to project modal window opens.
5. Select Cloudera Data Visualization from the Kernel drop-down list.
6. Select the edition of the Cloudera Data Visualization you want to add to your project.

   The selected edition automatically defines the version of the runtime variant.

   Note: New Cloudera Data Visualization Runtime versions are automatically added to the list of available editions, if internet connection is available.
7. Click Submit.

Results

You should now be able to select the Cloudera Data Visualization Runtime when creating a Cloudera Data Visualization application.

What to do next

Create a Cloudera Data Visualization application.

Creating a Cloudera Data Visualization application in Cloudera AI

Learn how to create a Cloudera Data Visualization application in Cloudera AI to help you visualize and interact with your data insights. This integration allows for seamless visualization of ML model outputs, data exploration, and reporting within the same platform.

About this task

The following steps will guide you on how to create a Cloudera Data Visualization application in a Cloudera AI project.

Important:

Each Cloudera AI project can host only one standalone Cloudera Data Visualization application. Since applications share the same metadata and logs, any changes made by one application can overwrite those made by another.

Before you begin

Ensure that a Cloudera Data Visualization Runtime is available in the Cloudera AI project where you want to create the Cloudera Data Visualization application.

For more information about ML Runtimes, see Managing ML Runtimes and Using Runtime Catalog.

Procedure

1. Navigate to the Overview page of your Cloudera AI project.
2. On the left sidebar, click Applications.
3. Click New Application.
4. Provide the following details for your new application:
   - Name – Enter a name for the application.
   - Subdomain – Enter a subdomain that will be used to construct the URL for the web application. Ensure it only contains URL-friendly characters.
   - Description – Add a description for the application.
   - Script – Select the path to the startup script.

     Note: Use the script located at: /opt/vizapps/tools/arcviz/startup_app.py

   - Runtime – Configure the Runtime.
     - Editor – Workbench
     - Kernel – Select Cloudera Data Visualization for the kernel supported by the Runtime variant of the Cloudera AI project.
     - Edition – Select the edition of the Runtime variant you want to use for your application.

       Note: The selected edition determines the version of the Runtime variant.
5. Click Create Application.

Results

After a few minutes, the application status will change from Starting to Running on the Applications page. Your Cloudera Data Visualization application is now fully operational.

You can restart, stop, or delete a Cloudera AI application from the supplemental menu. If you want to make changes to the application, navigate to Application Details Settings .

What to do next

Start Cloudera Data Visualization.

Deploying a Cloudera Data Visualization application in Cloudera Data Science Workbench

Learn how to deploy Cloudera Data Visualization in Cloudera Data Science Workbench.

Creating a Cloudera Data Science Workbench project with Cloudera Data Visualization Runtime

Learn how to create a Cloudera Data Science Workbench project with a Cloudera Data Visualization Runtime as the default runtime.

Procedure

1. Click Projects on the left sidebar of Cloudera Data Science Workbench.
2. Click New Project on the Projects page.
3. Enter a project name.
4. Select the visibility of the project.
5. Under Initial Setup, you can either create a blank project, or select a source for your project files.
6. Click Create Project.

   The created project uses Cloudera ML Runtimes that are managed in the Runtime Catalog.

What to do next

After the project is created, you can start creating an application. You can select which runtime edition you want to use for the application.

Creating a Cloudera Data Visualization application in Cloudera Data Science Workbench

Learn how to create a Cloudera Data Visualization application in Cloudera Data Science Workbench to help you visualize and interact with your data insights. This integration allows for seamless data visualization, exploration, and reporting within the same platform.

About this task

The following steps will guide you on how to create a Cloudera Data Visualization application within Cloudera Data Science Workbench.

Important:

Each Cloudera Data Visualization project can host only one standalone Cloudera Data Visualization application. Since applications share the same metadata and logs, any changes made by one application can overwrite those made by another.

Procedure

1. Navigate to the Overview page of your Cloudera Data Science Workbench project.
2. On the left sidebar, click Applications.
3. Click New Application.

4. Provide the following details for your new application:
   - Name – Enter a name for the application.
   - Subdomain – Enter a subdomain that will be used to construct the URL for the web application. Esure it only contains URL-friendly characters.
   - Description – Add a description for the application.
   - Enable Unauthenticated Access – Mark the checkbox if you want to allow unauthenticated access to your application. You can also update this setting later on the Settings page of the application.

     Note: To create public applications on an ML workspace, an Admin user needs to turn on the feature flag in Admin Security by selecting Allow applications to be configured with unauthenticated access. For more information on working with public applications, see *Application Limitations* in the Cloudera Data Science Workbench documentation.
   - Script – Select the path to the startup script.

     Note: Use the script located at: /opt/vizapps/tools/arcviz/startup_app.py
   - Runtime
     - Editor – Workbench
     - Kernel – Select Cloudera Data Visualization for the kernel supported by the Runtime variant of the Cloudera Data Science Workbench project.
     - Edition – Select the edition of the Runtime variant you want to use for your application.

       Note: The selected edition determines the version of the Runtime variant.

     All available runtimes are listed in the Cloudera Data Science Workbench Runtime Catalog. For more information about ML Runtimes, see Managing ML Runtimes and Using Runtime Catalog.

     Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.
5. Click Create Application.

Results

After a few minutes, the application status will change from Starting to Running on the Applications page. Your CDV application is now fully operational.

You can restart, stop, or delete a Cloudera Data Science Workbench application from the supplemental menu of the application. If you want to make changes to the application, navigate to Application Details Settings .

What to do next

Start Cloudera Data Visualization in Cloudera Data Science Workbench.

Starting Cloudera Data Visualization in Cloudera Data Science Workbench

Learn how to start the Cloudera Data Visualization application you have created in Cloudera Data Science Workbench.

Procedure

1. On the Applications page, click the name of your Cloudera Data Visualization application to access the login interface.

2. Log in to Cloudera Data Visualization by entering your username and password. Use your workload credentials.

   If you want to log in as an administrator, you can use the following default credentials:

   - username: vizapps_admin
   - password: vizapps_admin

   When using the default credentials to log in to Cloudera Data Visualization, you are prompted to change your password at the first login.

   Important: If you use the default credentials, security issues may arise. Cloudera recommends that you change the default username and password.

   SSO authentication is disabled by default. See Authentication for information on how to permit user authentication with the Cloudera Data Science Workbench login credentials and log users in automatically.

   After logging in, you land on the homepage view of Cloudera Data Visualization. Here you can explore some sample dashboards or access the in-tool Get Started guide for help.

Connecting to a data source in Cloudera Data Visualization

Cloudera Data Visualization allows you to create connections to many types of external data sources, enhancing your data analysis and visualization capabilities.

Cloudera Data Visualization currently supports the following connection types:

- Hive
- Impala
- MariaDB
- MySQL
- PostgreSQL
- Solr [Technical Preview]
- Spark SQL
- SQLite (not supported in CDW)
- Snowflake [Technical Preview]

This range of supported connection types provides flexibility and versatility in connecting to various data sources for comprehensive data analysis.

When using Cloudera Data Visualization with Cloudera Data Warehouse, the data connection is automatically set up, but you can connect to various other data sources as well to suit your specific requirements.

In Cloudera AI, you can connect to an Impala or a Hive data warehouse, or tie in data from predictive Cloudera AI models.

Creating a Cloudera AI data connection to Impala

Learn how to connect natively to data stored in Impala when using Cloudera Data Visualization in Cloudera AI.

About this task

Before you start using data modeling and visualization functions, you must connect to your data. The following steps show you how to create a new Cloudera AI data connection in Cloudera Data Visualization to an Impala data warehouse.

Note:

You must have the Manage data connections privilege or be an admin to be able to create and manage connections in Cloudera Data Visualization.

Setting user privileges requires administrator-level access. You can log in as an administrator using the default admin account with the following credentials:

- Username: vizapps_admin
- Password: vizapps_admin

When you create a connection, you automatically get privileges to create and manage the associated datasets. You can also build dashboards and visuals within these datasets.

- For more information on the Manage data connections privilege, see *RBAC permissions*.
- For instructions on how to define privileges for a specific role, see *Setting role privileges*.
- For instructions on how to assign the administrator role to a user, see *Promoting a user to administrator*.

Before you begin

If you are using a Cloudera Private Cloud Base cluster running Impala with Kerberos for authentication, make sure that Kerberos credentials are configured in Cloudera AI before creating a Cloudera AI data connection to the Impala data warehouse. This ensures seamless integration and authentication between Cloudera Data Visualization and the Impala cluster. If you add Kerberos credentials after launching the Cloudera Data Visualization app, you need to restart the app for the changes to take effect.

For more information on using Kerberos for authentication in Cloudera AI, see *Hadoop Authentication for AI Workspaces*.

Procedure

1. On the main navigation bar, click DATA.

   The DATA interface opens, displaying the Datasets tab.

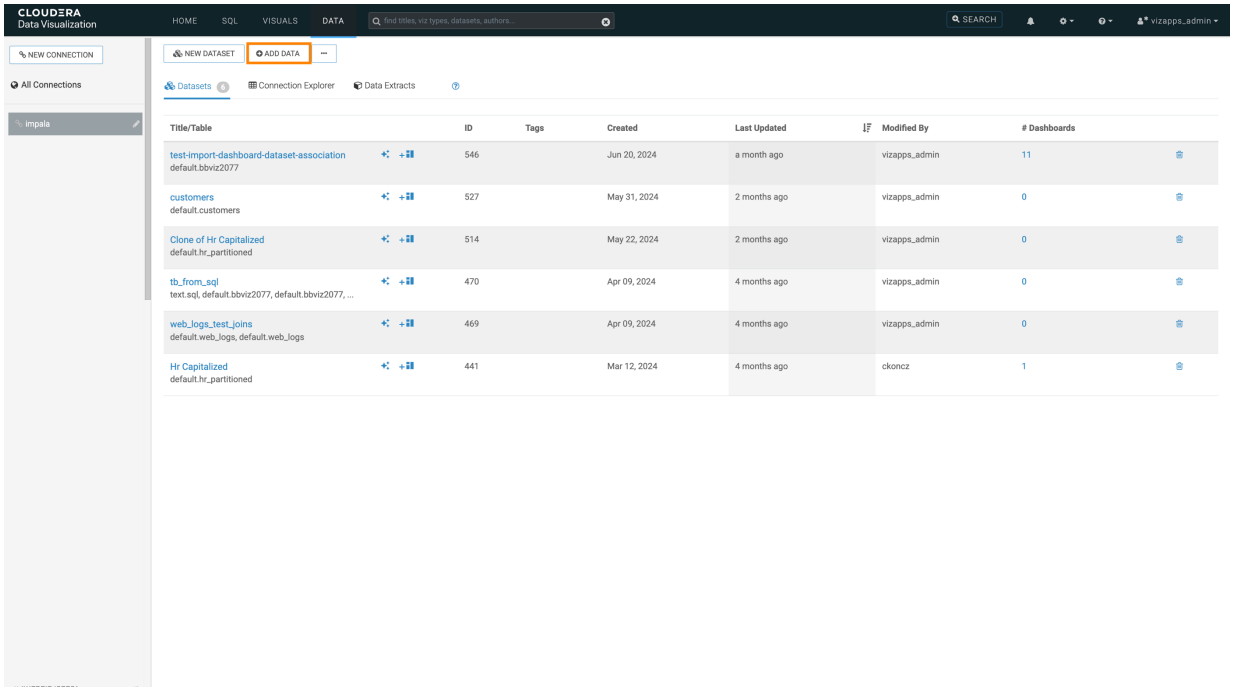

2. On the side menu bar, click NEW CONNECTION.

Note: Only users with Manage data connections privilege or administrators can access the NEW CONNECTION button.

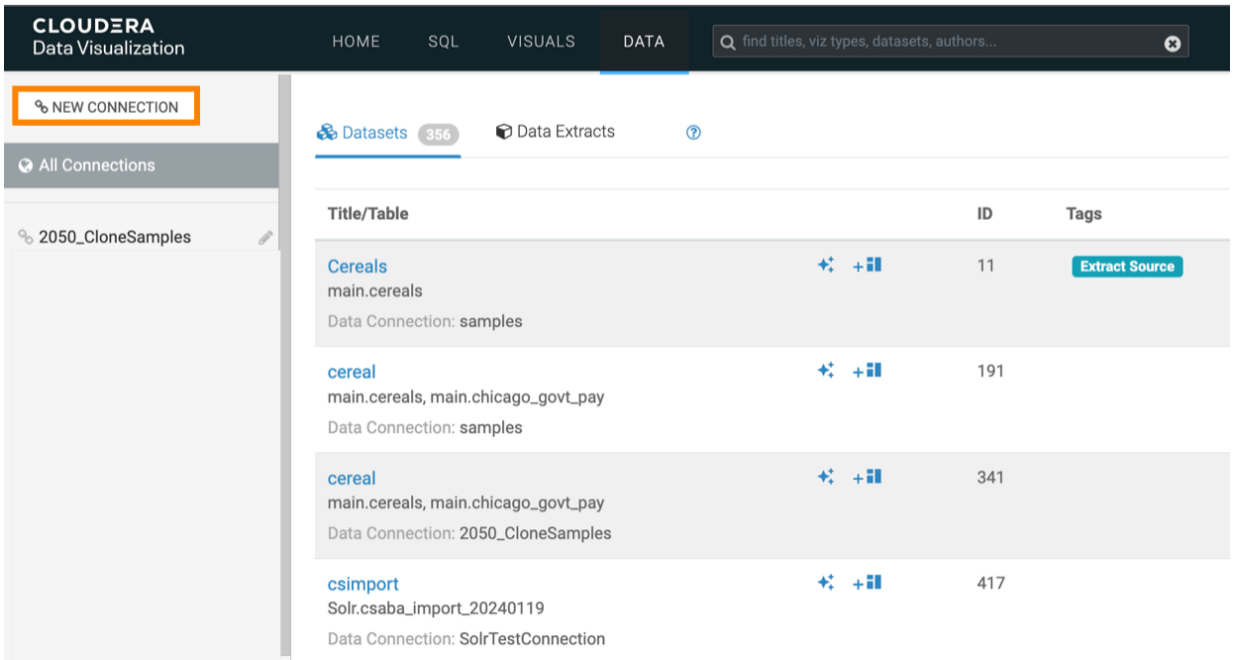

The Create New Data Connection modal window appears.

3. Choose Impala from the Connection type drop-down list and assign a name to your connection.

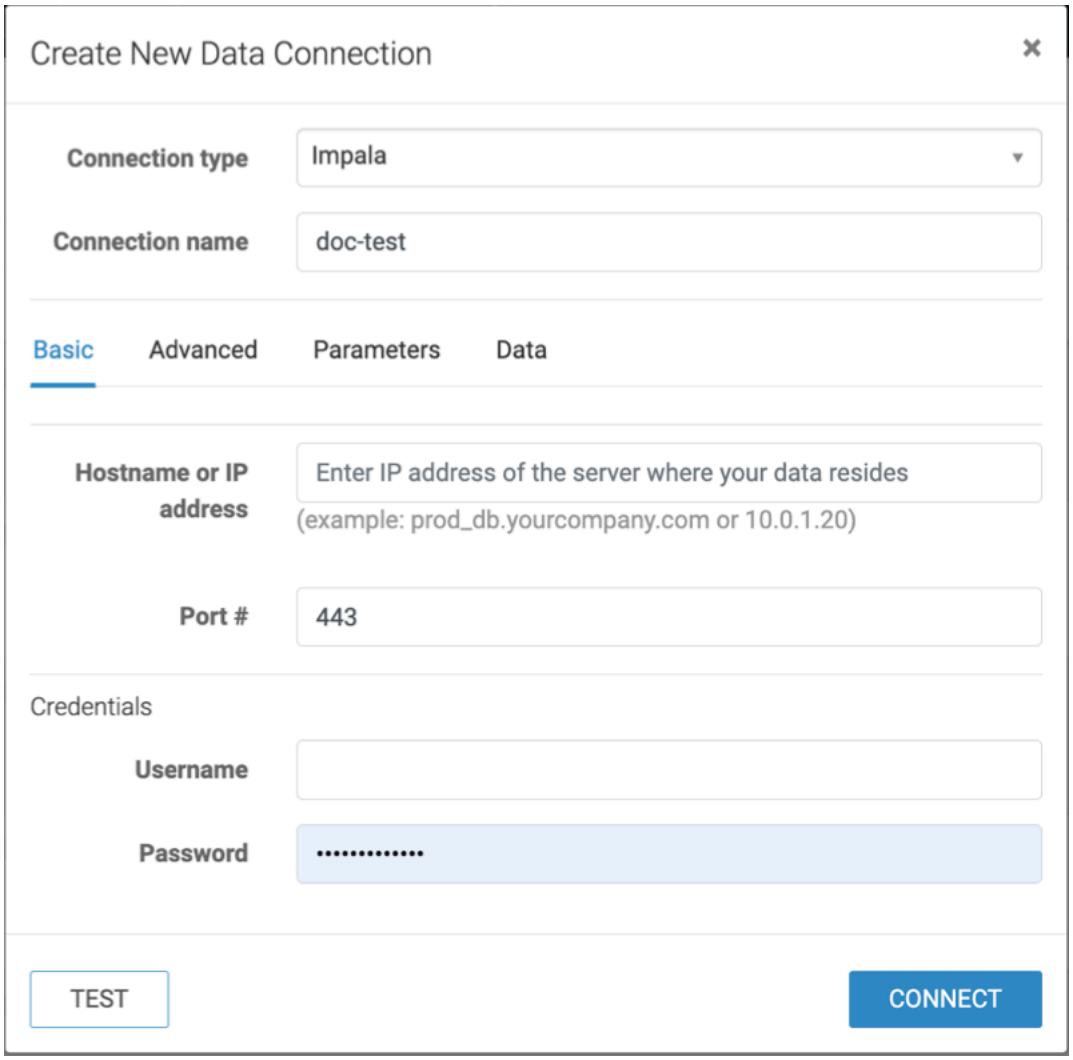

In this example, the Impala connection is made through Knox. Knox always uses TLS encryption and port 443 is the default HTTPS port.

4. Enter the hostname or IP address of the running coordinator.

   You can retrieve this information from the JDBC URL of the Impala DW.

5. Add 443 in the Port # field.
6. Enter your workload username and password as credentials.

7. Click the Advanced tab to configure additional details.

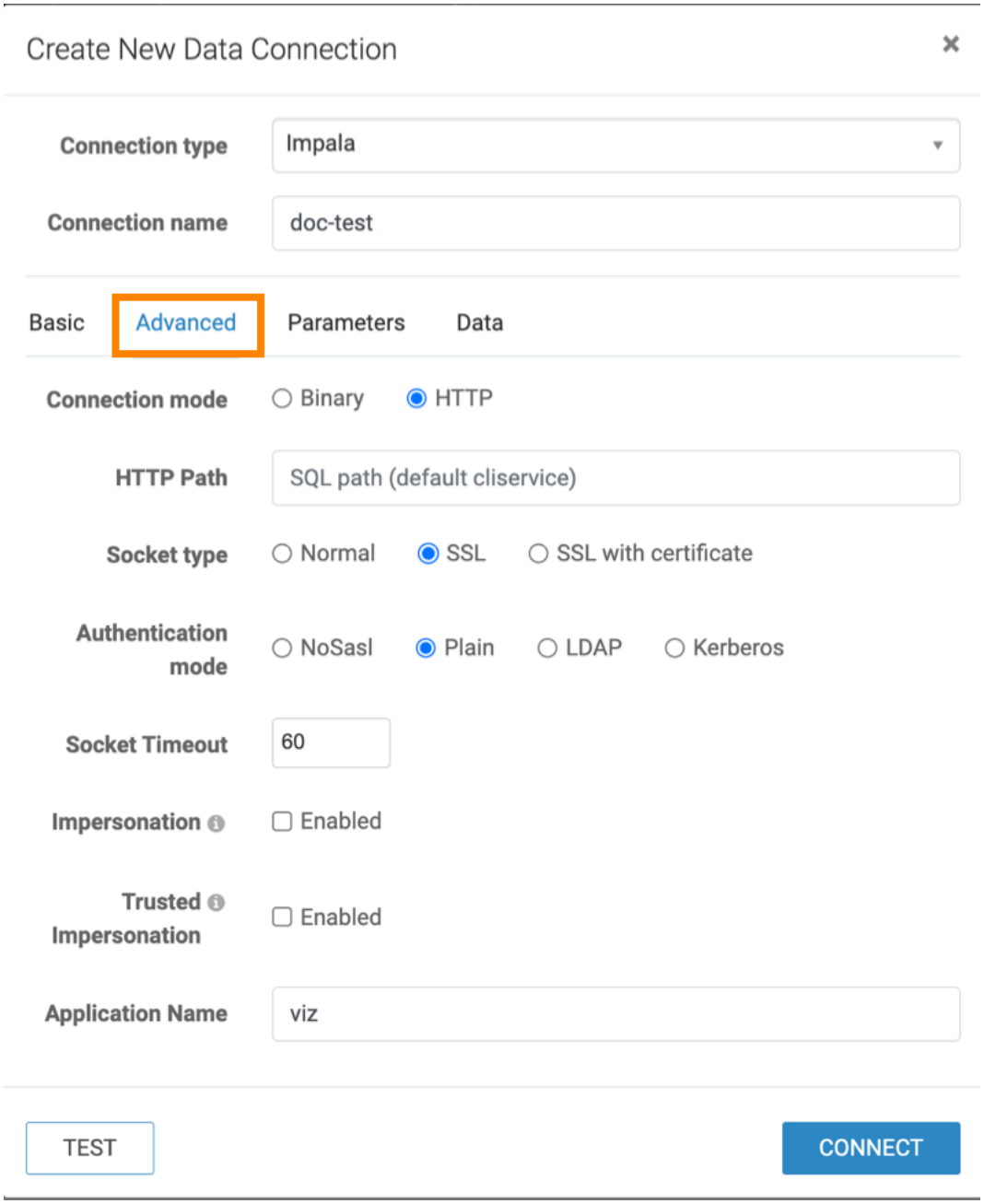

a) For HTTP connection mode, locate the Impala Endpoint for the Data Hub.

b) Copy and paste it into the HTTP Path field.

c) Set any additional details as required.

8. Check the Parameters and Data tabs for more configuration options.

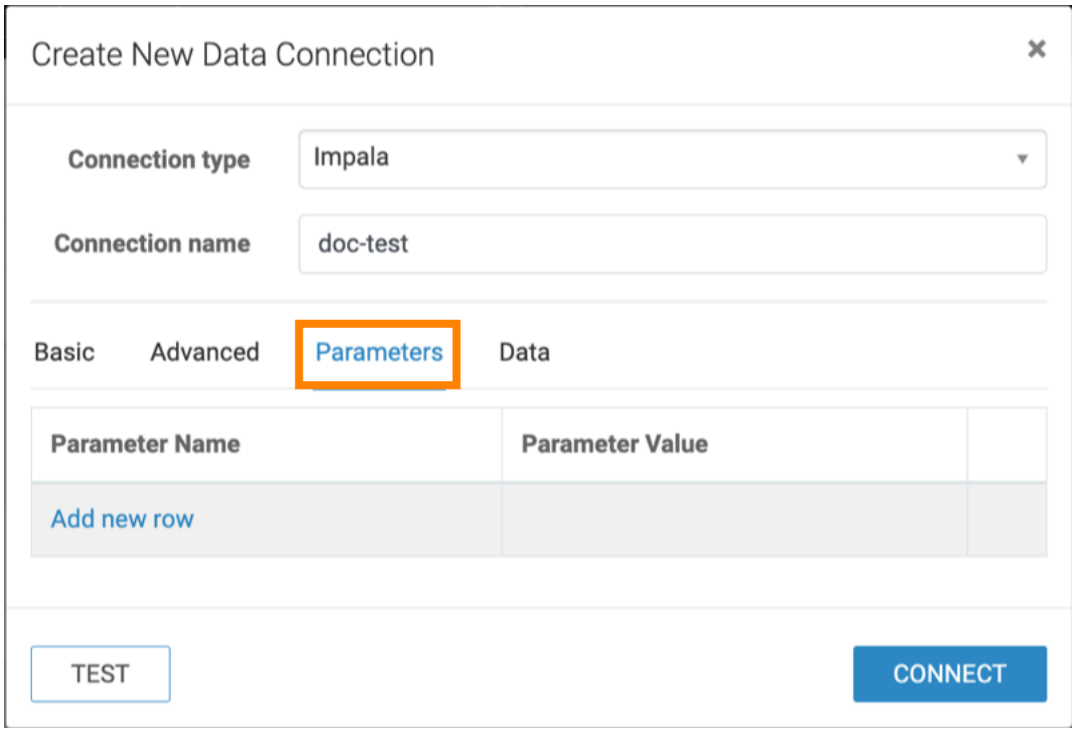

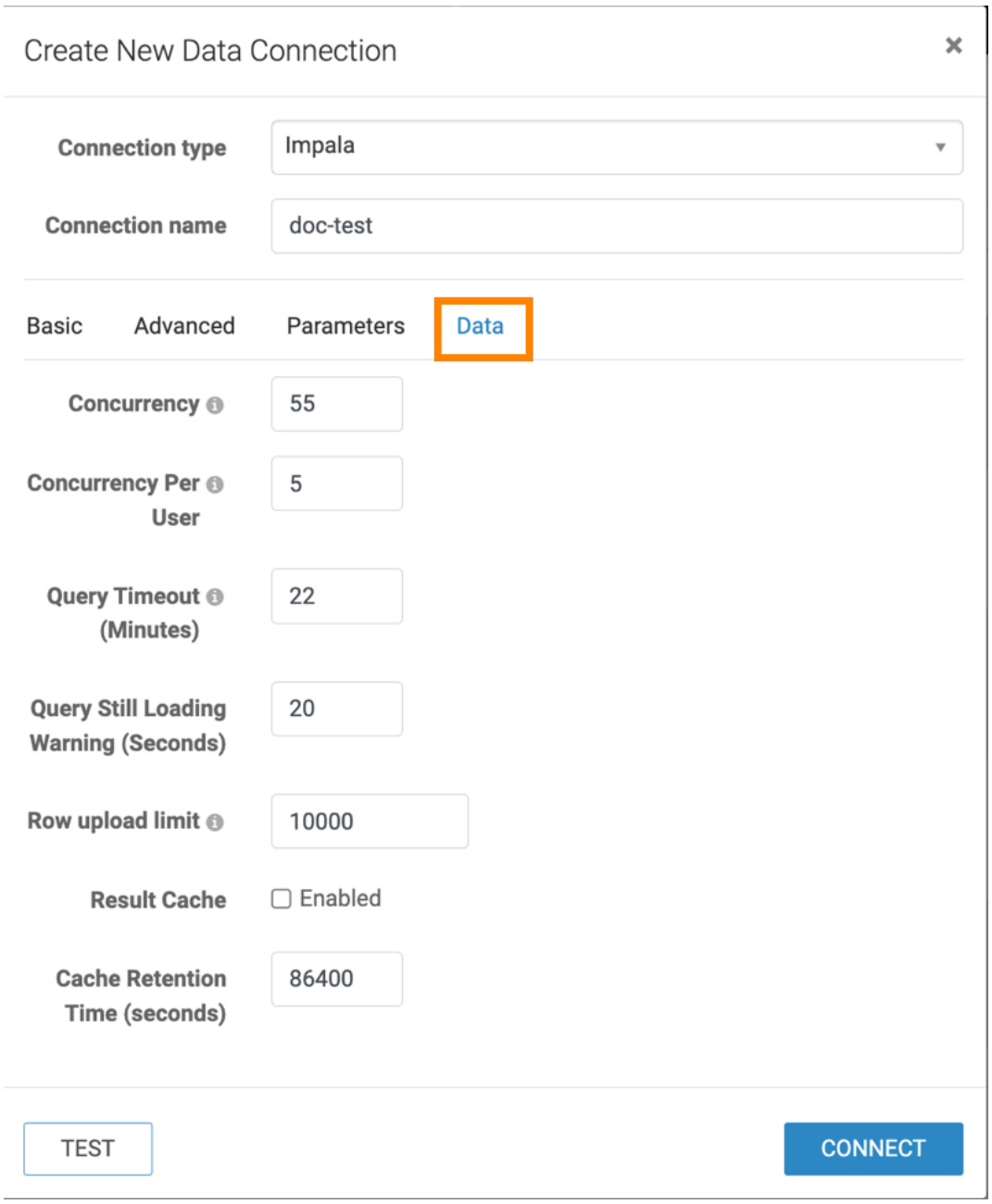

9. Once you finish configuring the settings, click TEST to check the connection.
10. Click CONNECT to establish the connection.

Results

You have successfully set up a connection to a running Impala DW.

Related Information

RBAC permissions
Setting role privileges
Promoting a user to administrator

Creating a Cloudera AI data connection to a Hive data warehouse

Learn how to connect natively to data stored in Hive when using Cloudera Data Visualization in Cloudera AI.

About this task

Before you start using data modeling and visualization functions, you must connect to your data. The following steps show you how to create a new Cloudera AI data connection in Cloudera Data Visualization to a Hive data warehouse.

Note:

You must have the Manage data connections privilege or be an admin to be able to create and manage connections in Cloudera Data Visualization.

Setting user privileges requires administrator-level access. You can log in as an administrator, using the default admin account with the following credentials:

- Username: vizapps_admin
- Password: vizapps_admin

When you create a connection, you automatically get privileges to create and manage the associated datasets. You can also build dashboards and visuals within these datasets.

- For more information on the Manage data connections privilege, see *RBAC permissions*.
- For instructions on how to define privileges for a specific role, see *Setting role privileges*.
- For instructions on how to assign the administrator role to a user, see *Promoting a user to administrator*.

Before you begin

If you are using a Cloudera Private Cloud Base cluster running Hive with Kerberos for authentication, make sure that Kerberos credentials are configured in Cloudera AI before creating a Cloudera AI data connection to the Hive data warehouse. This ensures seamless integration and authentication between Cloudera Data Visualization and the Hive cluster. If you add Kerberos credentials after launching the Cloudera Data Visualization app, you need to restart the app for the changes to take effect.

Procedure

1. On the main navigation bar, click DATA.

   The DATA opens, displaying the Datasets tab.

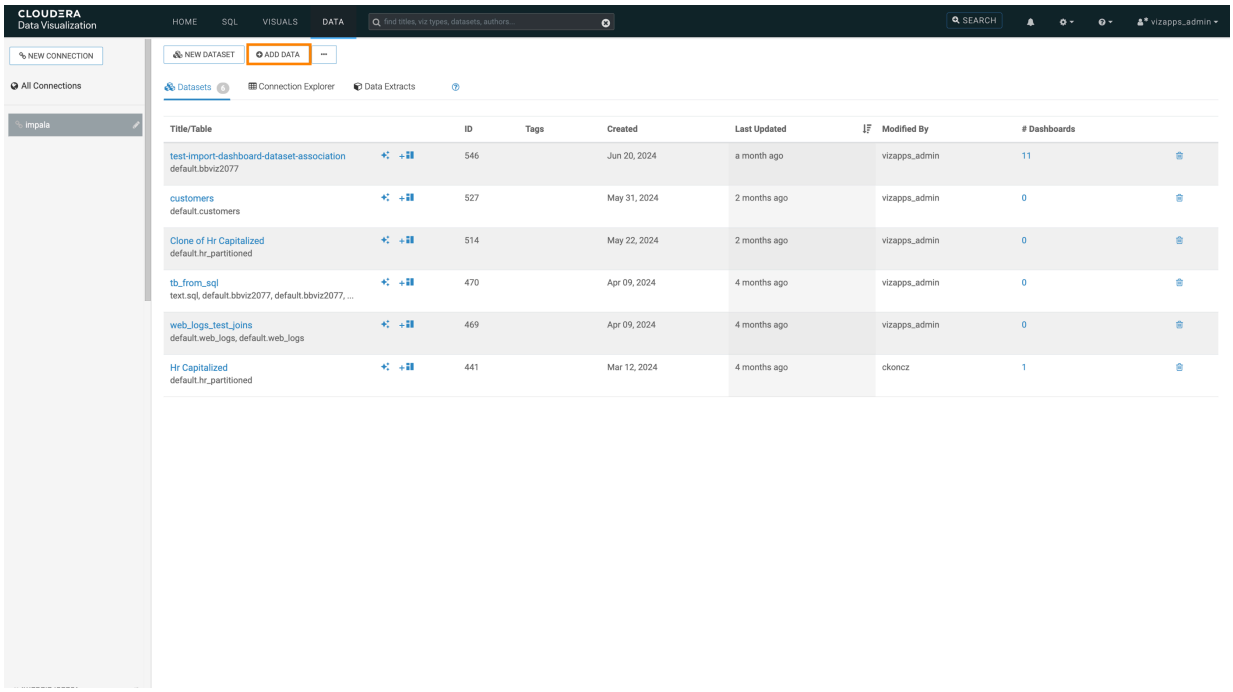

2. On the side menu bar, click NEW CONNECTION.

Note: The NEW CONNECTION button is only accessible to users assigned to roles with the Manage data connections privilege and to administrators.

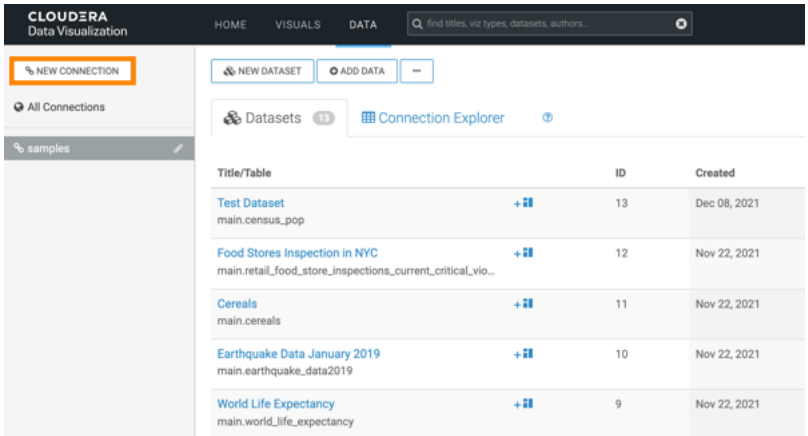

The Create New Data Connection modal window appears.

3. Choose Hive from the Connection type drop-down list and assign a name to your connection.

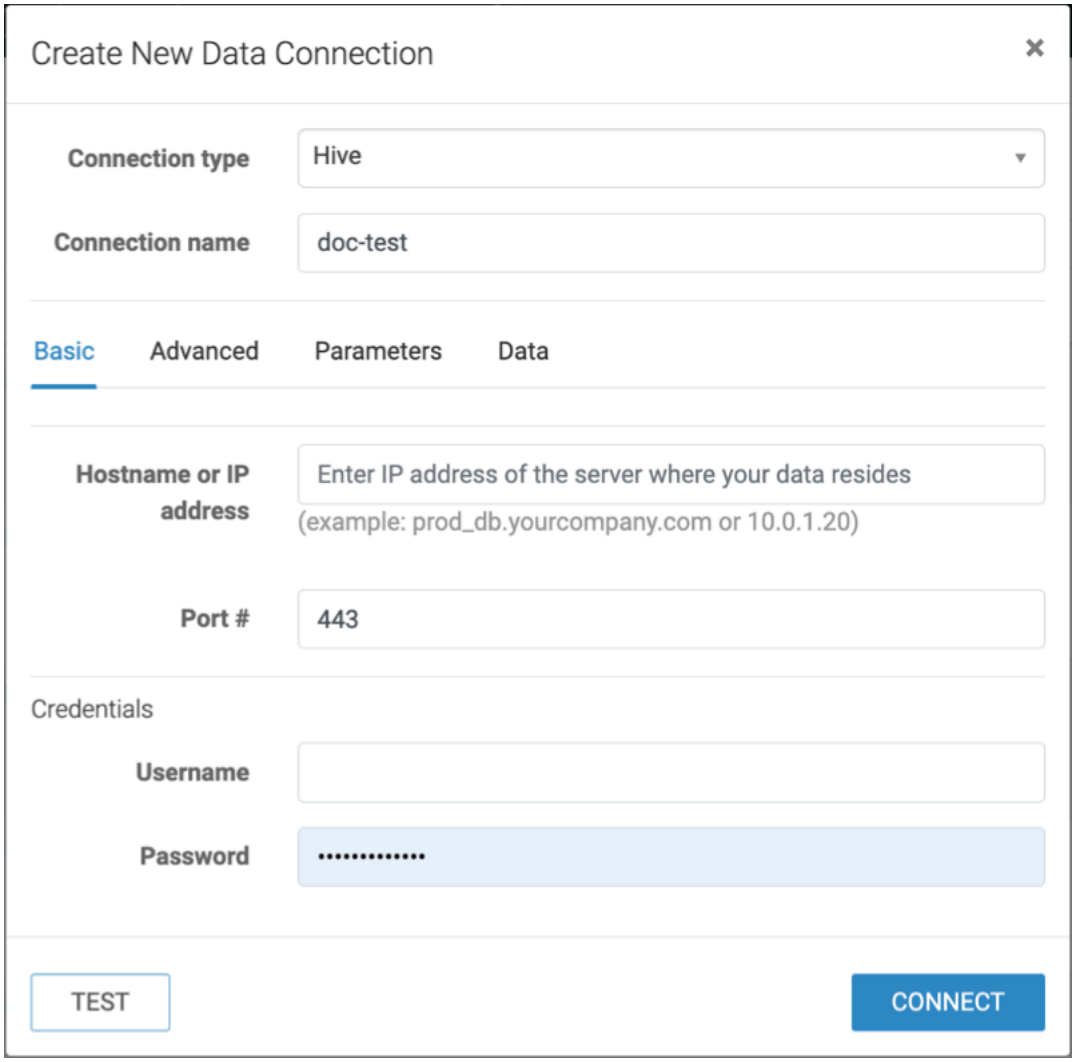

4. Enter the hostname or IP address of the running coordinator.
   You can get the coordinator hostname from the JDBC URL of the Hive DW.
5. Use port 443.
6. Enter your workload username and password as credentials.
7. Click the Advanced tab to configure the additional details.

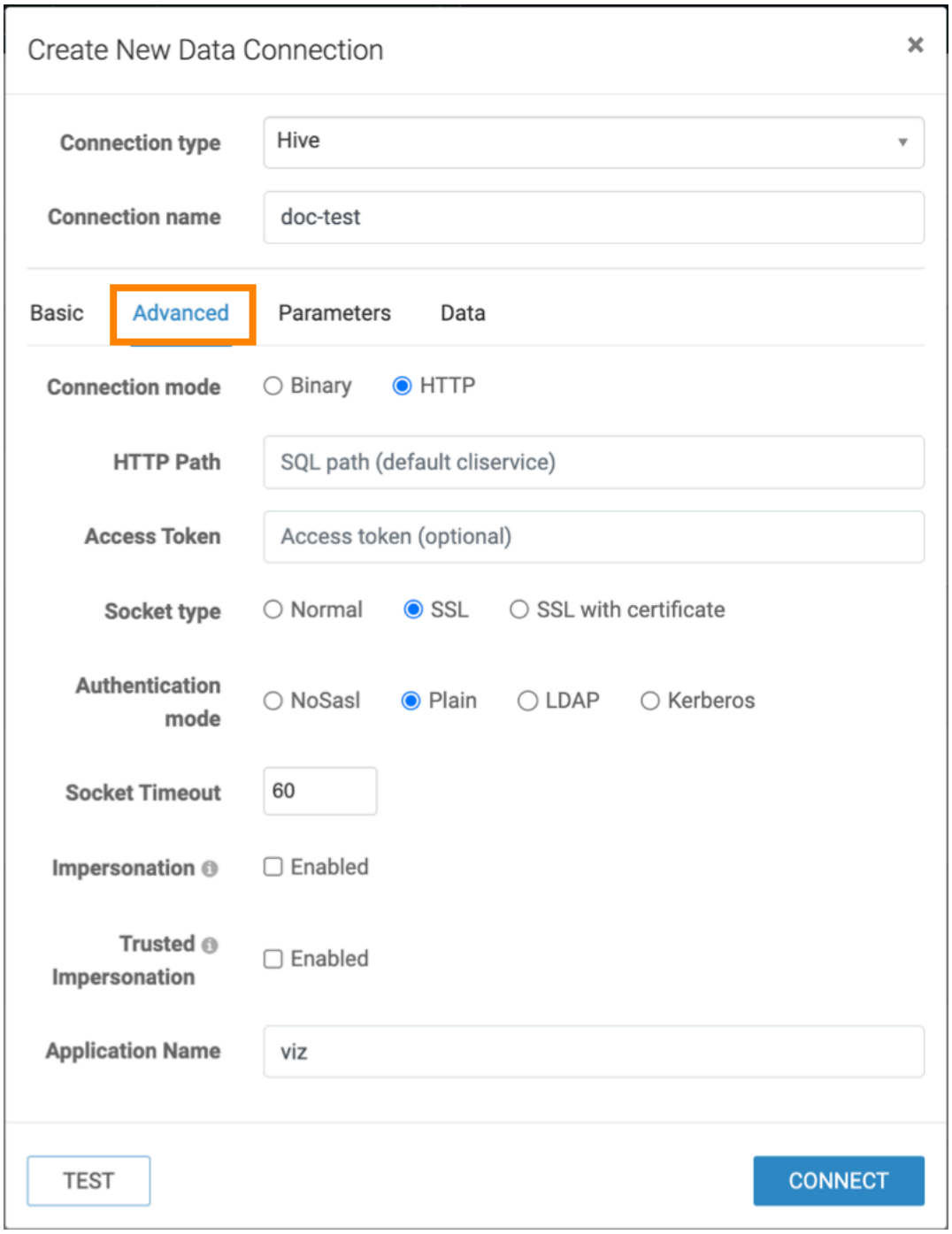

8. Click the Parameters tab and set the hive.server2.async.exec.async.compile parameter to false.

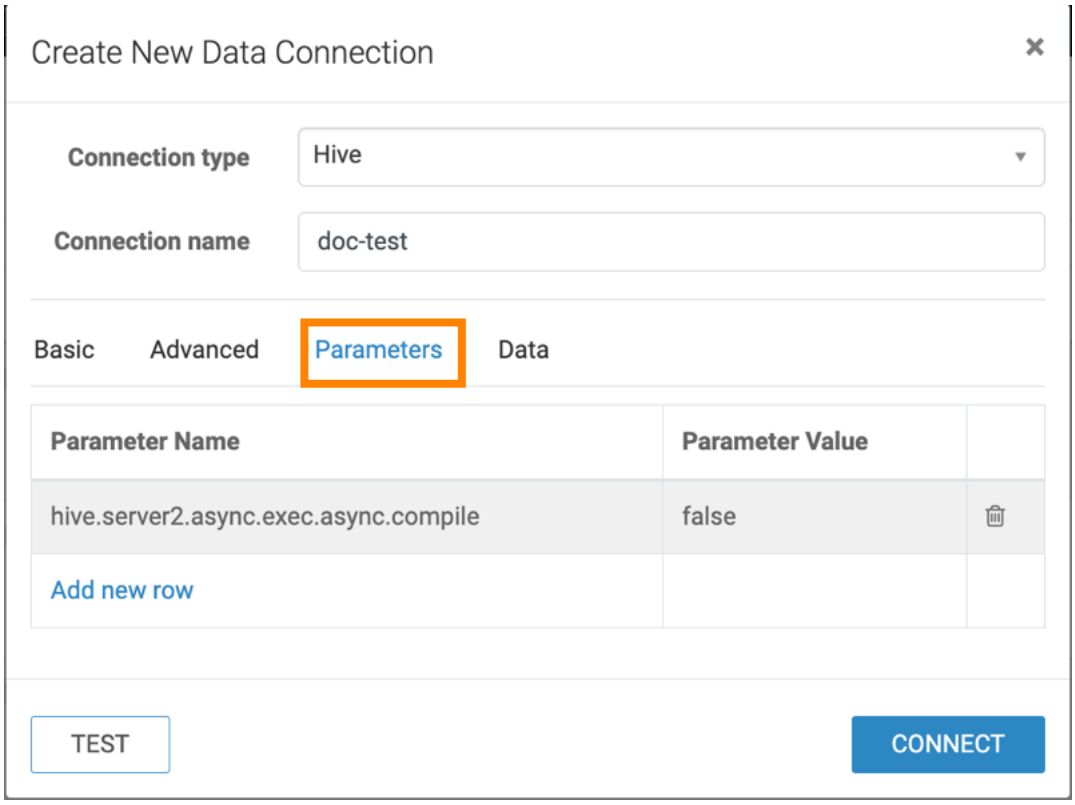

9. Check the Data tab for more configuration options.

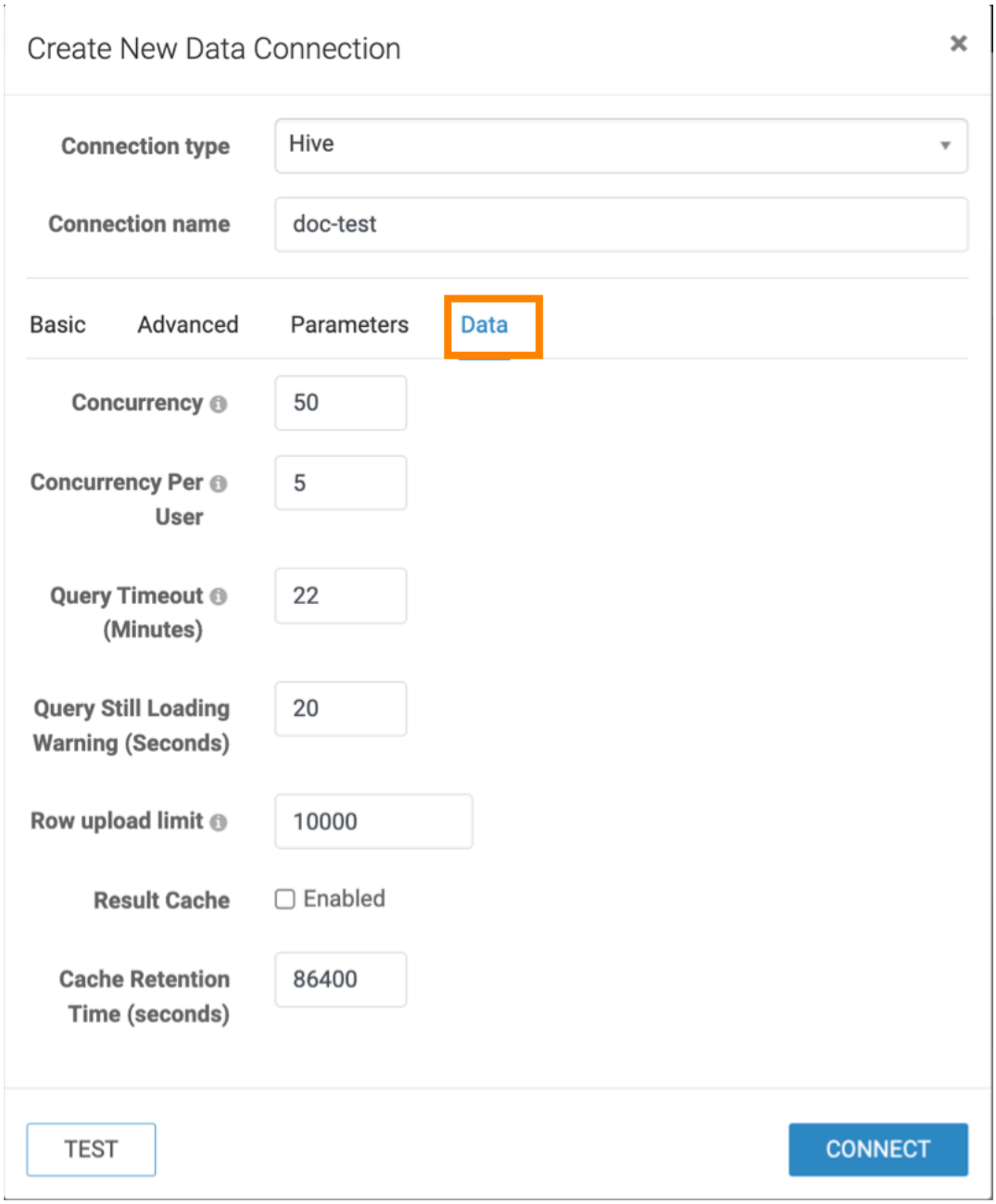

10. Once you finish configuring the settings, click TEST to test the connection.

11. Click CONNECT to establish the connection.

Results

You have successfully set up a connection to a running Hive DW.

Related Information

RBAC permissions

Setting role privileges

Promoting a user to administrator

Creating a Cloudera Data Warehouse data connection in Cloudera Data Visualization

Learn how to connect to data when using Cloudera Data Visualization in Cloudera Data Warehouse data service. You can connect Cloudera Data Visualization to a Virtual Warehouse to visualize your data. Similar to using a BI client, you can configure and connect to Virtual Warehouses from different clusters.

About this task

You must connect to your data prior to using the data modeling and visualization functions. You make the connection to the Virtual Warehouse when you select your warehouse in the steps below. The Cloudera Data Warehouse URL has the same compute instance ID as your Virtual Warehouse.

Note:

To create and manage connections in Cloudera Data Visualization, you must have the Manage data connections privilege or hold administrative privileges. In Cloudera Data Warehouse, these are the members of the Admin Groups associated with the Cloudera Data Visualization instance.

When you create a connection, you automatically gain privileges to create and manage datasets associated with this connection, and to build dashboards and visuals within these datasets.

- For more information on the Manage data connections privilege, see *RBAC permissions*.
- For instructions on how to define privileges for a specific role, see *Setting role privileges*.
- For instructions on how to assign the administrator role to a user, see *Promoting a user to administrator*.

When you are creating a Hive or Impala data connection within the same cluster, the connection is considered secure and trusted, and the connection details can be auto-populated with a default authentication user.

Procedure

1. Start Cloudera Data Visualization from the left navigation panel in Cloudera Data Warehouse.
2. On the main navigation bar, click DATA.

   The DATA interface appears, open on the Datasets tab.

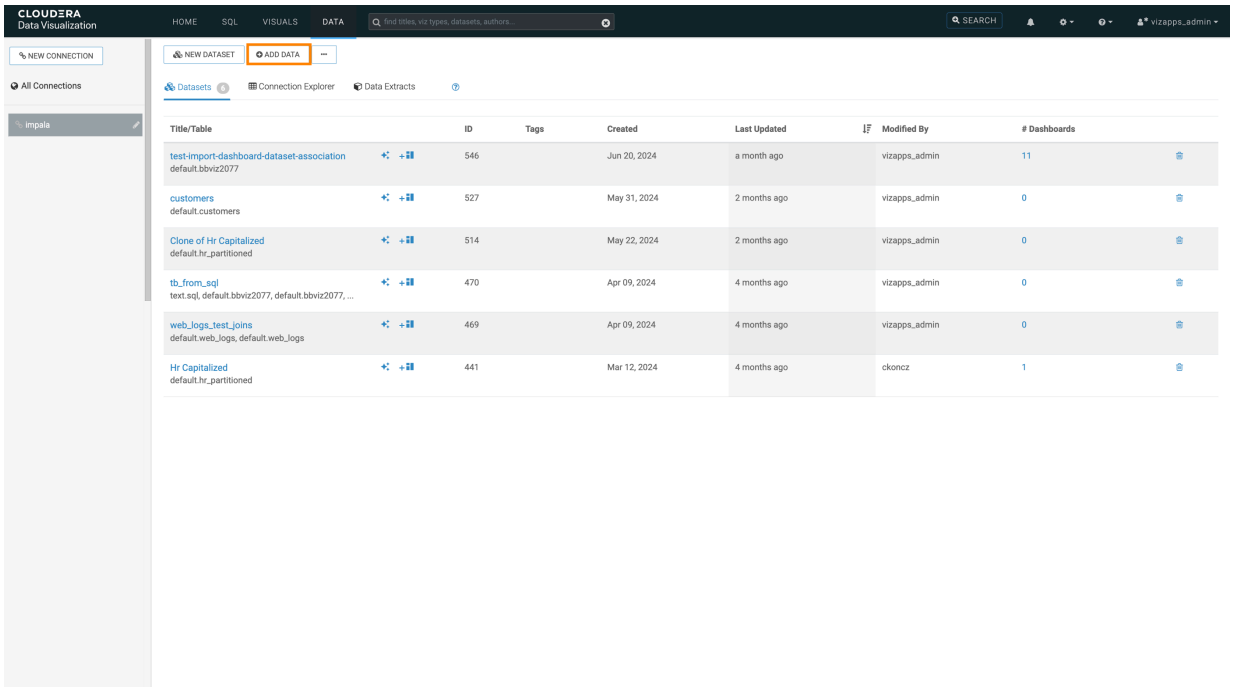

3. In the side menu bar of DATA, click NEW CONNECTION.

Note: The NEW CONNECTION button is only accessible to users assigned to roles with the Manage data connections privilege and to administrators.

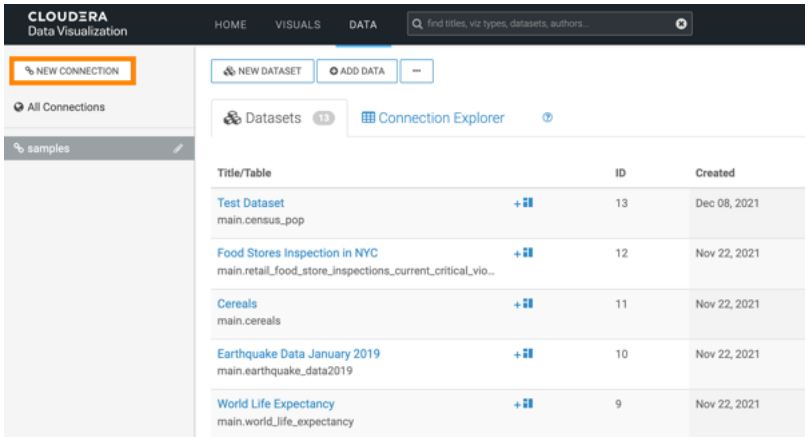

The **Create New Data Connection** modal window appears.

4. In Connection type, select CDW Hive or CDW Impala, and provide a name for your connection.

   Note: SQLite connection is not supported in Cloudera Data Warehouse, and the option is disabled in the connection list.

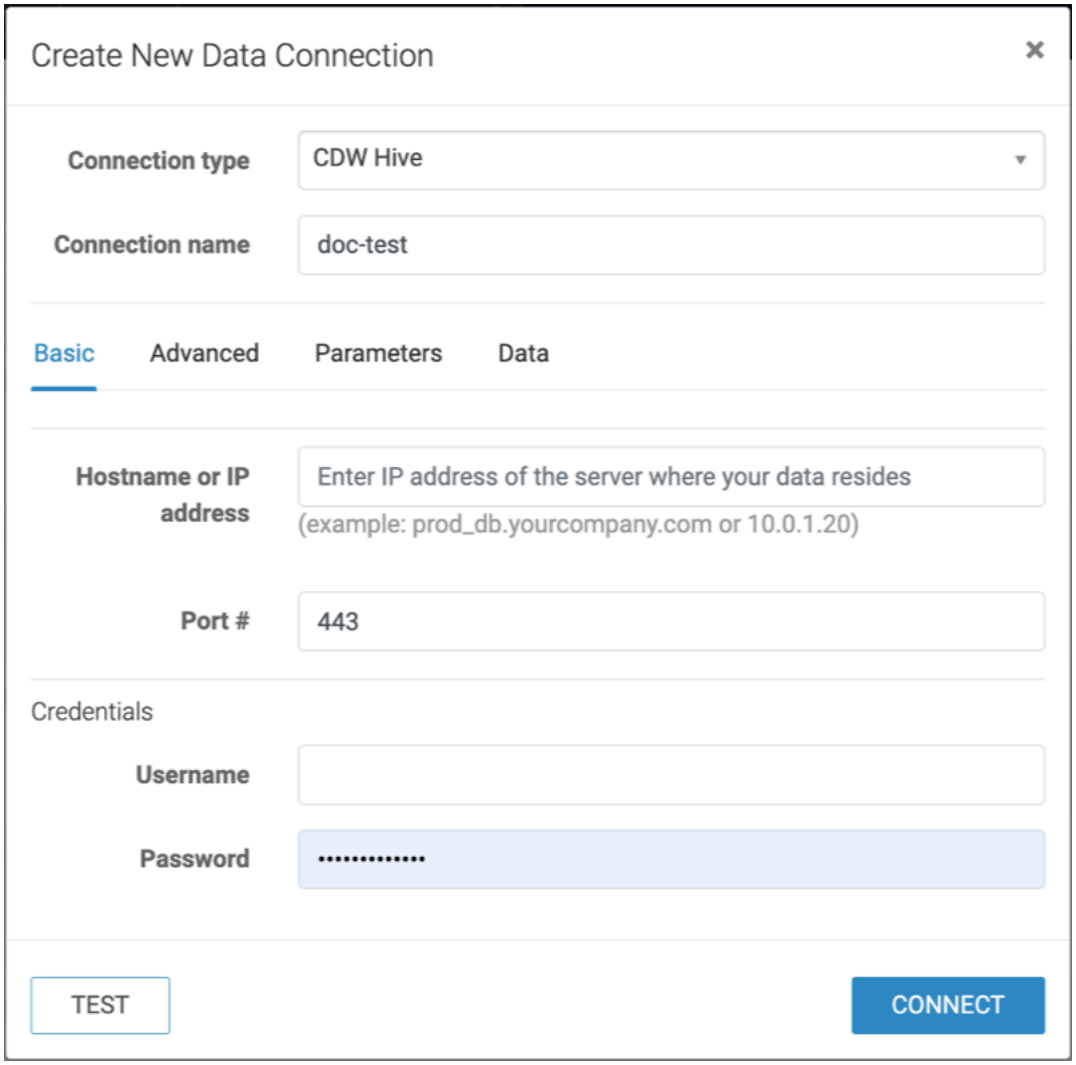

5. Select a Cloudera Data Warehouse warehouse to connect to.

   For Data connection within the same cluster

   The following fields are auto-populated:

   - Hostname or IP address
   - Port #
   - Username

   For Data connection outside the cluster

   Enter the following information:

   - Hostname or IP address
   - Port #
   - Username
   - Password

6. Click the Advanced tab and configure the additional details.

   Important: Depending on the type of connection you are creating, there can be additional tabs in the Create New Data Connection modal window where you have to adjust further settings.

7. Click TEST.

   If the connection is valid, the system returns a Connection Verified message.

8. Click CONNECT.

What to do next

You can create a data set, and then start creating visuals, dashboards, and applications. For more information, see *Creating datasets* and *Creating a visual*.

Related Information

RBAC permissions

Setting role privileges

Promoting a user to administrator

Creating datasets

Creating a visual

Creating a Cloudera Data Science Workbench data connection to a data warehouse

Learn how to connect natively to data stored in a data warehouse when using Cloudera Data Visualization in Cloudera Data Science Workbench.

About this task

You must connect to your data prior to using the data modeling and visualization functionalities. The following steps show you how to create a new Cloudera Data Science Workbench data connection to a running Impala system.

Note:

To create and manage connections in Cloudera Data Visualization, you must have the Manage data connections privilege or hold administrative privileges.

Setting user privileges requires administrator-level access. To log in as an administrator, you can use the default admin account with the following credentials:

- Username: vizapps_admin
- Password: vizapps_admin

When you create a connection, you automatically gain privileges to create and manage datasets associated with this connection, and to build dashboards and visuals within these datasets.

- For more information on the Manage data connections privilege, see *RBAC permissions*.
- For instructions on how to define privileges for a specific role, see *Setting role privileges*.
- For instructions on how to assign the administrator role to a user, see *Promoting a user to administrator*.

Procedure

1. On the main navigation bar, click DATA.

   The DATA interface appears, open on the Datasets tab.

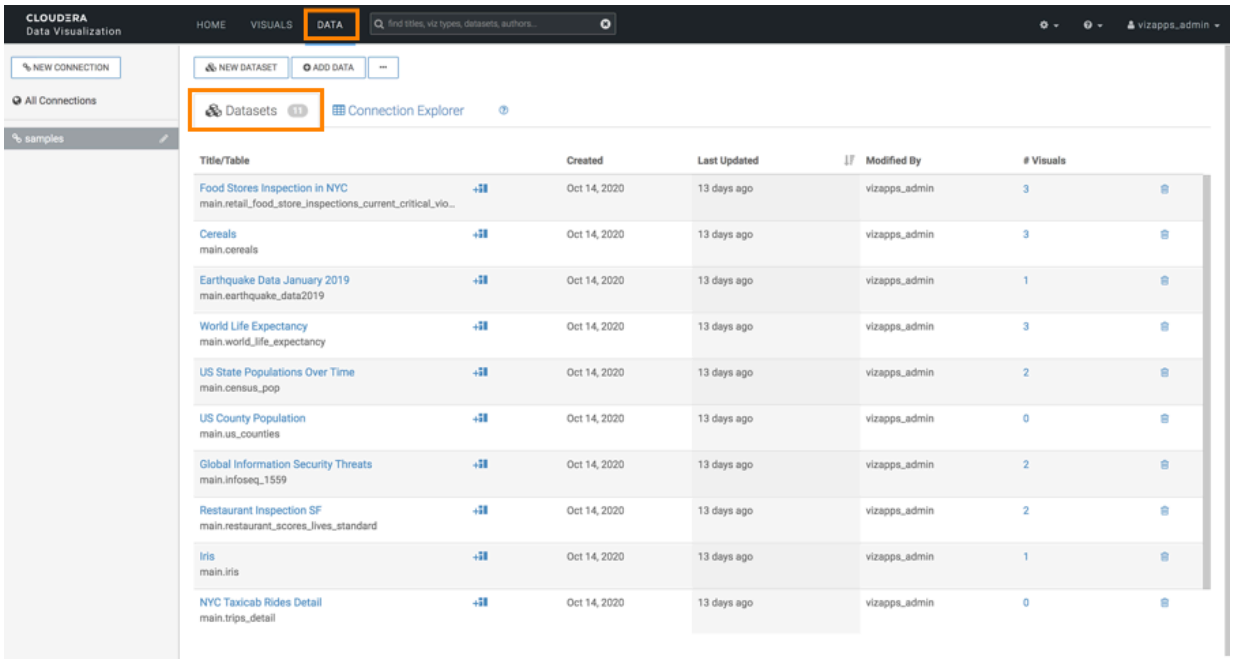

2. In the Data side menu bar, click NEW CONNECTION.

   Note: The NEW CONNECTION button is only accessible to users assigned to roles with the Manage data connections privilege and to administrators.

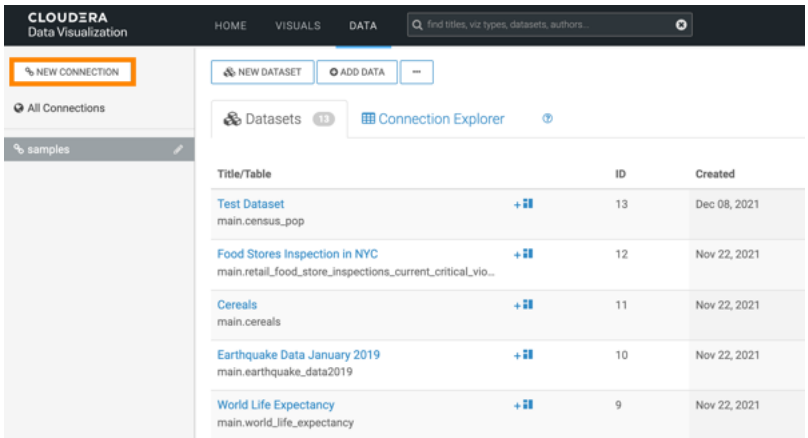

   The Create New Data Connection modal window appears.

3. Select a Connection type from the drop-down list, and provide a name for the connection.
4. Enter the hostname or IP address of the running coordinator.
5. Under Port #, enter the port number.
6. Use your workload username and password as credentials.
7. Click the Advanced tab and make the appropriate selections.

   Important: Depending on the type of connection you are creating, there can be additional tabs in the Create New Data Connection modal window where you have to adjust further settings.

8. Click TEST.

   If the connection is valid, the system returns a Connection Verified message.

9. Click CONNECT.

Results

You have set up a connection to a running data warehouse.

Related Information

RBAC permissions

Setting role privileges

Promoting a user to administrator

Creating a visual

You can easily create new visual representations for your data in Cloudera Data Visualization.

About this task

The following steps demonstrate how to create a new visual.

Procedure

1. On the main navigation bar, click VISUALS.
2. Click Create New Dashboard to open a new, blank dashboard.

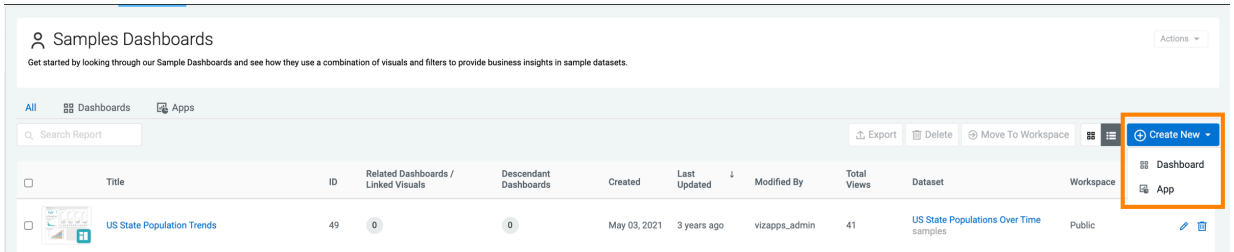

If you already have an existing dashboard and you just want to add a new visual, open the dashboard and click EDIT to make changes to it.

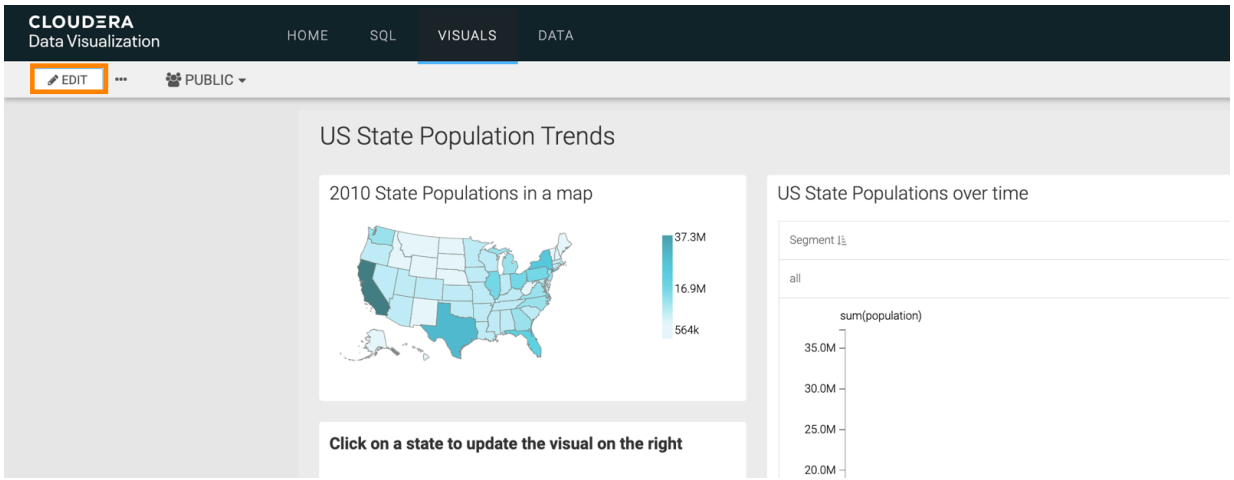

3. In the Dashboard Designer interface, open the Visuals menu from the side menu bar and click NEW VISUAL.

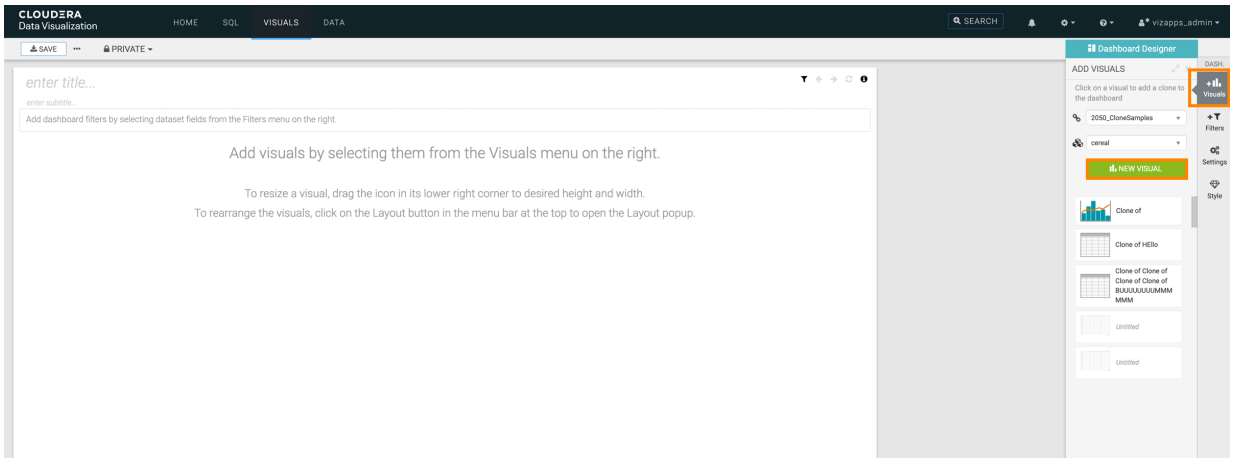

4. The Visual Builder appears.

   By default, a table visual is selected. For more information, see the *Visual Builder* overview.

5. Choose the visual type you want to build.
   In this example, the Bars visual has been selected.

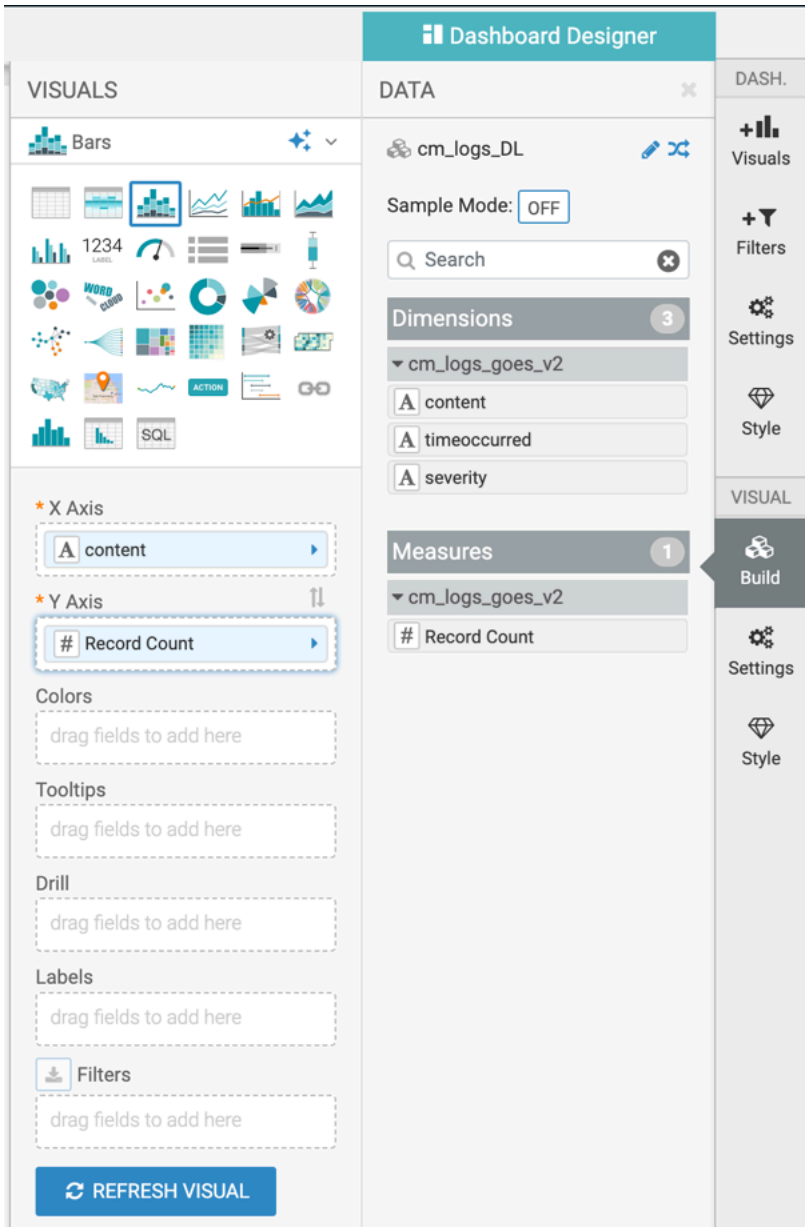

6. You can switch on Sample Mode for the visual you are building.

   With sample mode, you create the visual on a subset of your data to test it. You can define this subset as a percentage of the original data.

   Note: This feature is only available on Impala, Hive, and Spark connections.

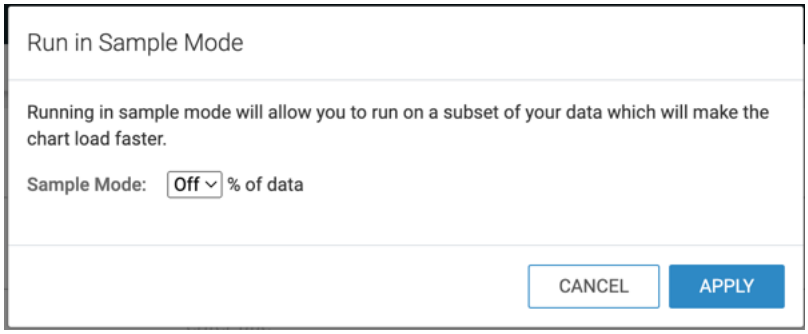

7. Populate the shelves of the visual from the Dimensions and Measures fields:

   In this example, the following fields have been added:

   - state dimension on the X Axis shelf
   - population measure on the Y Axis shelf

     Note: The population measure appears with its default aggregation, as sum(population).

   - year measure on the Colors shelf
   - year measure on the Filters shelf

     Note: The Filter for year module opens automatically. Years 1970, 1980, 1990, 2000, and 2010 have been selected.

8. Click REFRESH VISUAL.

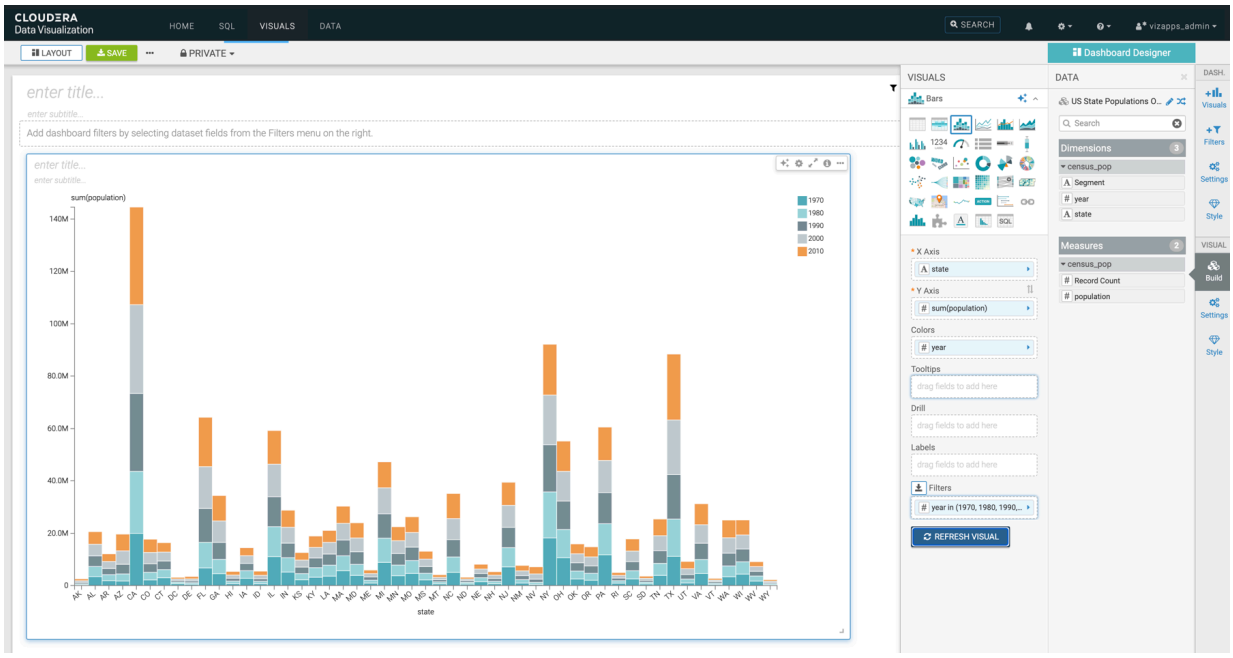

9. Click the enter title... field and enter a name for the visual.
   - In this example the title 'Recent US Census Results by State' has been added.
   - You can also click the enter subtitle... field below the title of the visualization to add a brief description of the visual.
10. Click SAVE at the top left corner of the Dashboard Designer.

Cloudera Data Visualization 7.2.9

Using Analytic Functions

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Supported connections 4

Using analytic functions - basic steps 4

Setting up a basic visual for aggregates 7

Setting up a basic visual for single values 8

Previous value analytic function 9

Difference from previous value analytic function 12

Percentage difference from previous value analytic function 15

Next value analytic function 17

Difference from next value analytic function 20

Percentage difference from next value analytic function 23

Row number function 26

Rank analytic function 29

Dense rank analytic function 31

Moving average analytic function 33

Percent of group analytic function 36

Running total analytic function 38

Supported connections

Cloudera Data Visualization supports several analytic functions that examine overlapping groupings of data.

Analytic functions are similar to aggregate functions because both use the contents of multiple data input rows to calculate the result. Analytic functions use flexible conditions that are specified by the OVER(...) clause to order and group input so that specific rows may be part of the calculation for several output values.

Note: Analytic functions do not validate in the Enter/Edit Expression editor. Refresh the visual to see if the function generates any errors.

Syntax for analytic functions is slightly different depending on the type of data connection used.

Supported Data Connections include the following:

- Impala
- Hive
- PostgreSQL
- SparkSQL
- RedShift

The Analytic Functions field properties are not available on MariaDB, MySQL, non-Oracle SQLite, Apache Drill, and Apache Solr connections.

Note:

- Syntax for analytic functions is slightly different depending on the type of data connection used. Analytic functions are not available for connections to MySQL, SQLite, Drill, MS Sql Server, Teradata, Solr, KSql, and MariaDB.
- In the query execution order, analytic functions follow the WHERE and GROUP BY clauses. Therefore, the function excludes the rows that are filtered out by these mechanisms, and they never become part of the analytic function data subset.
- When using both analytic functions and ordering, the available ordering options include all fields that are on the shelves, less the fields that are on the Filters shelf. To sort a visual on the results of an analytic functions, place the field used in the analytic function onto the Tooltips shelf. For more information, see *Customizing order and top K*.
- Use the Customizing enter/edit expressions option for running analytic functions that are not automated within Cloudera Data Visualization.

Related Information

Customizing order and top K

Customizing enter/edit expressions

Using analytic functions - basic steps

Procedure

1. Open the visual where you want to specify an analytic function, in Edit mode.
2. On a measurement shelf of a visual, click the field you plan to modify to open the Field Properties menu.

   In this examples, the population field is used on the X Axis shelf.

3. In the FIELD PROPERTIES menu, click to expand the Analytic Functions menu.

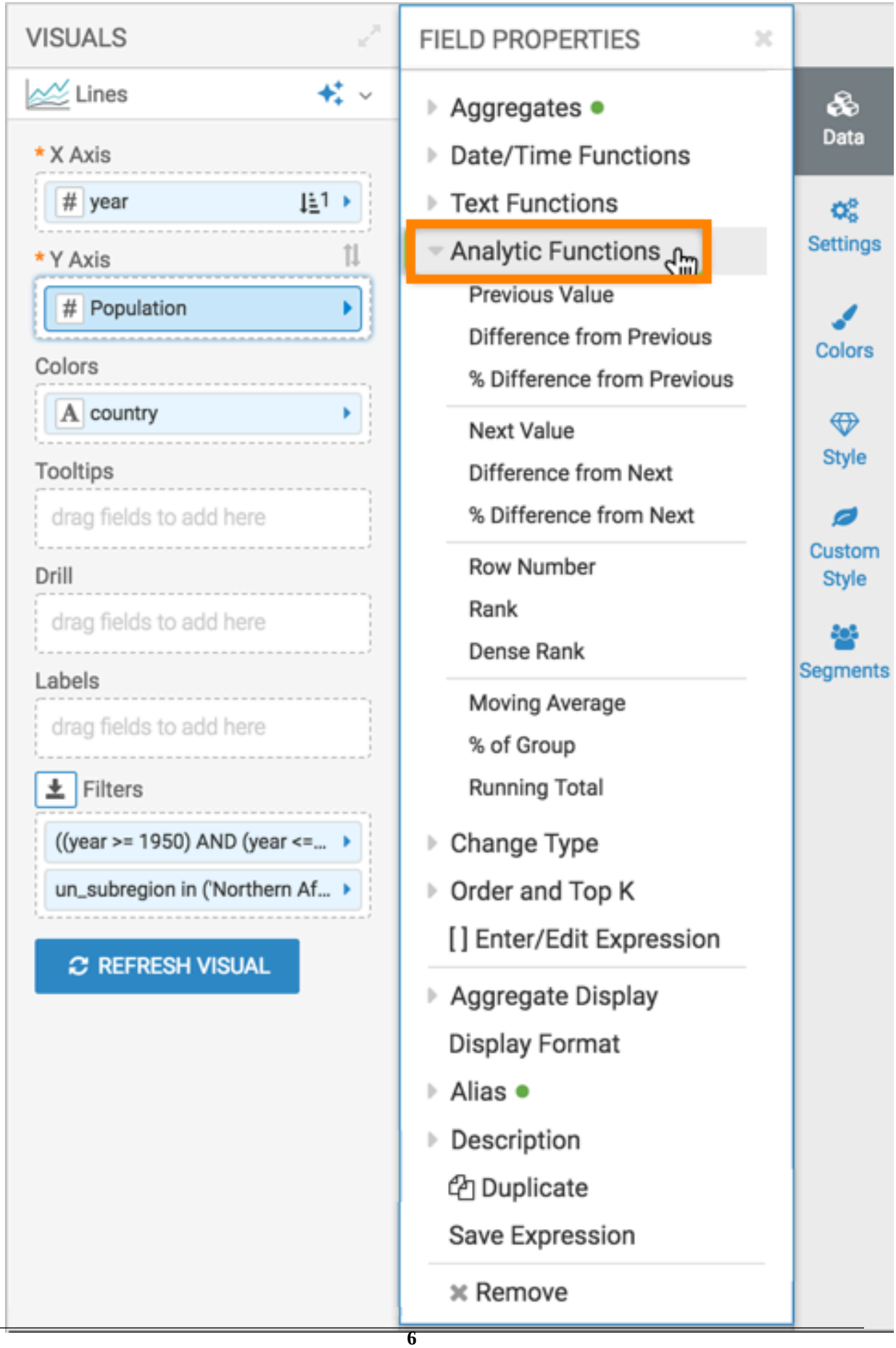
VISUALS
Lines
* X Axis
year
* Y Axis
Population
Colors
country
Tooltips
drag fields to add here
Drill
drag fields to add here
Labels
drag fields to add here
Filters
((year >= 1950) AND (year <=...
un_subregion in ('Northern Af...
REFRESH VISUAL
FIELD PROPERTIES
Aggregates
Date/Time Functions
Text Functions
Analytic Functions
Previous Value
Difference from Previous
% Difference from Previous
Next Value
Difference from Next
% Difference from Next
Row Number
Rank
Dense Rank
Moving Average
% of Group
Running Total
Change Type
Order and Top K
[ ] Enter/Edit Expression
Aggregate Display
Display Format
Alias
Description
Duplicate
Save Expression
Remove
Data
Settings
Colors
Style
Custom Style
Segments

4. Select one of the following analytic functions, directly supported by Cloudera Data Visualization.
5. In addition to these, you may use the expression builder to specify other analytic functions.

   For more information, see *Customizing enter/edit expressions*.

Related Information

Customizing enter/edit expressions

Setting up a basic visual for aggregates

About this task

In this example, a basic line visual is created on the World Life Expectancy dataset. In subsequent topics, this visual is used to demonstrate how the various analytic functions for aggregates work.

Procedure

1. Open a new visual.
2. In the VISUALS menu, select the Lines visual type.
3. Populate the shelves of the visual from the fields listed in the Data menu:
   - *X Axis*

     Add the field year. Order it in ascending order.
   - *Y Axis*

     Add the field population.
   - *Colors*

     Add the filed country. Order it in ascending order.
   - *Filters*

     Add the field year, and set it to the interval of 1950 through 2010.

     Add the field un_subregion, and set it to Northern Africa.

4. Click REFRESH VISUAL to see the new line visual.

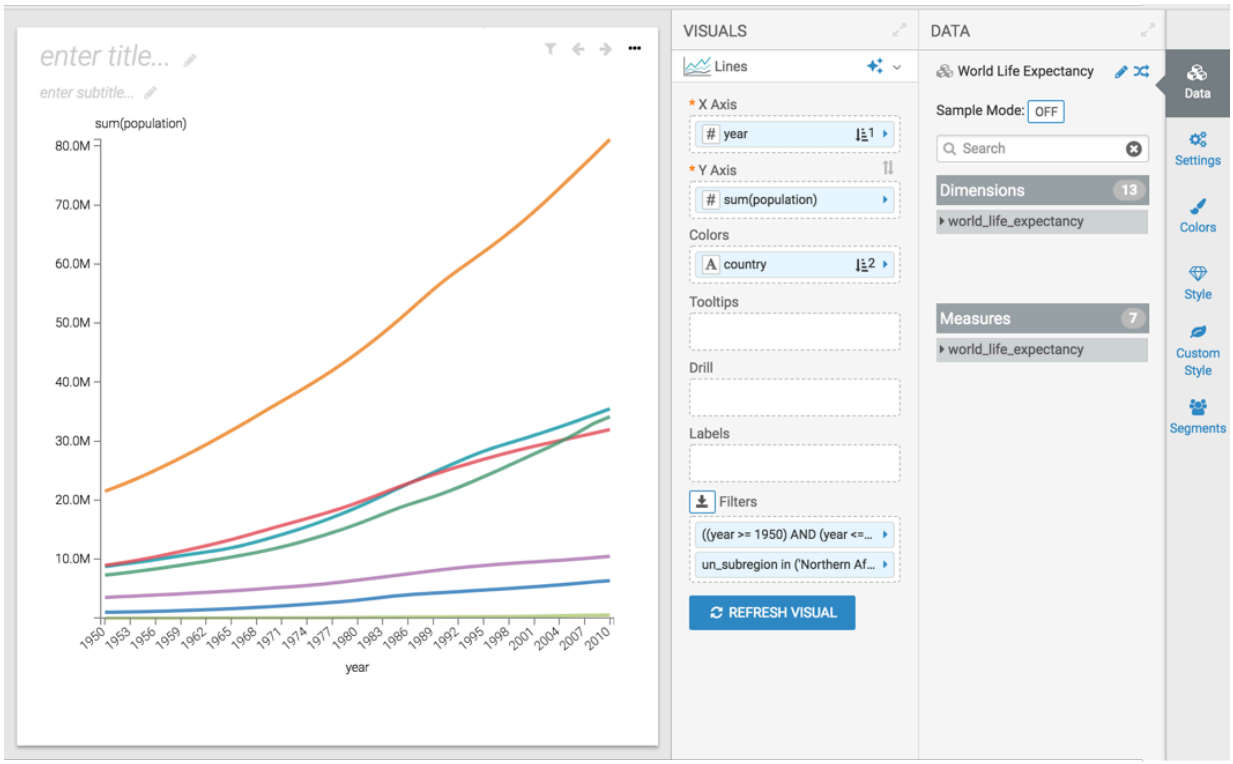

5. Name the visual Basic Lines.
6. Click SAVE.

Setting up a basic visual for single values

Procedure

1. Open a new visual.
2. In the Visuals menu, select the Table visual type.
3. Populate the shelves of the visual from the fields listed in the Data menu:
   - *Dimensions*

     Add the fields country, life_expectancy, gdp_per_capita, and population.
   - *Filters*

     Add the field year, and select the value 2010.

     Add the field un_subregion, and set it to Africa.
4. Specify descending order on life_expectancy.
5. In the Enter/Edit Expression editor, change the gdp_per_capita calculation on the shelf, and rename it:[gdp_per_capita]*[population] as 'GDP'
6. Change the Display Format options for the fields on the shelves to remove extra decimals, specify currency, and so on.

7. Change the Alias for the fields on the shelves. Click Refresh Visual to update the table visual.

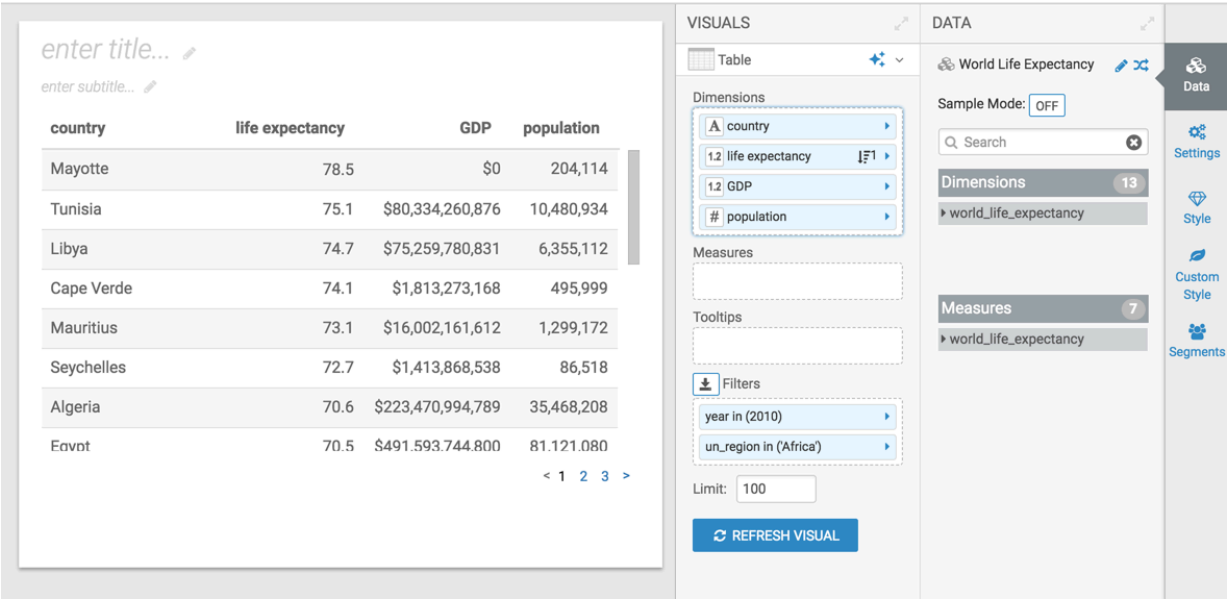

8. Name the visual Basic Table.
9. Click Save.

Previous value analytic function

This article describes how to use the Previous Value analytic function in Cloudera Data Visualization.

Procedure

1. Click the field on the measurement shelf.
   In this example, the sum(population) field on the Y Axis shelf.

2. In the FIELD PROPERTIES menu, expand Analytic Functions, and select Previous Value.

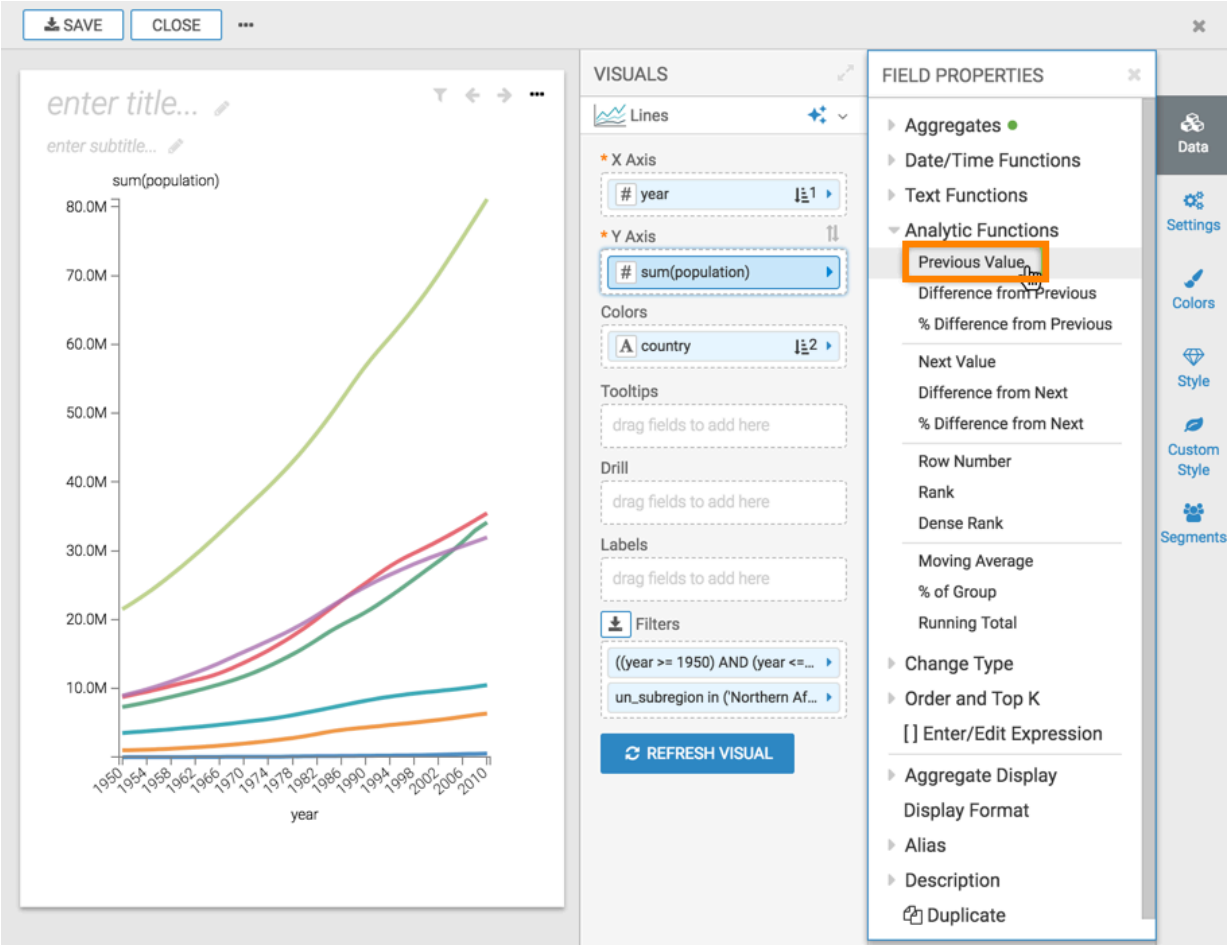

3. In the Analytic Function: Previous Value modal window:
   a. In Select aggregate for this measure, leave the default Sum. The other aggregation options are Count, Minimum, Maximum, and Average. In addition, some connection types also support Approximate Distinct Count and Exact Distinct Count.
   b. Under Select entities over which to tale previous value, select state. This is the grouping that applies when calculating previous values.
   c. Under Select the sort order, click the Plus icon, select year, and then set it to ascending sort order. This order specifies what determines what the previous row is, and the row where the value is taken.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

   d. Under Offset, specify the number of previous rows over which the function executes.

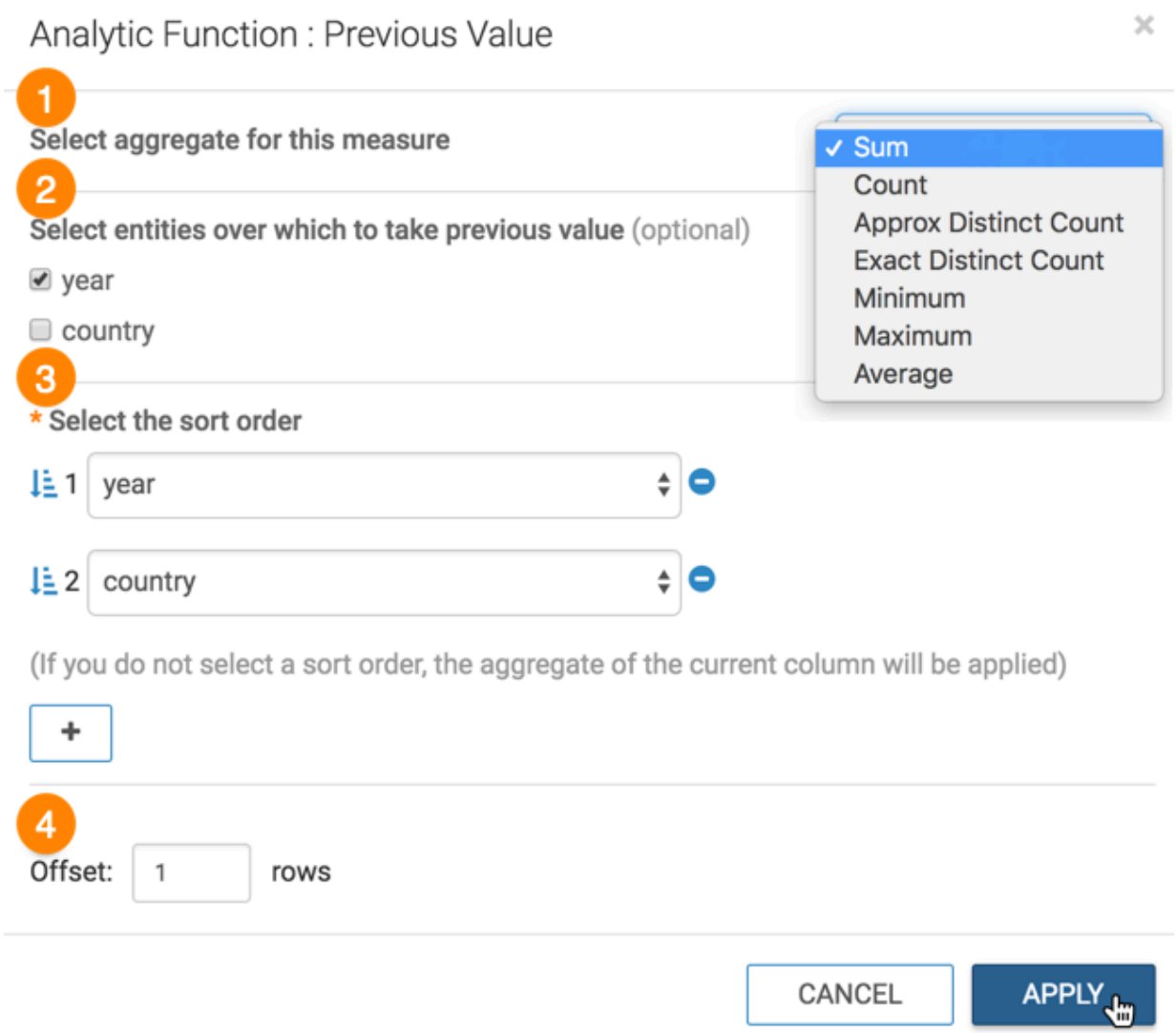

4. Click APPLY.
5. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form: LAG(sum([population]), 1) over (partition by [year] order by [year], [country]) as 'previous value(population)'

6. Click REFRESH VISUAL.

   You can see the change to the appearance of the visual, and the change in values that appear on the tooltip.

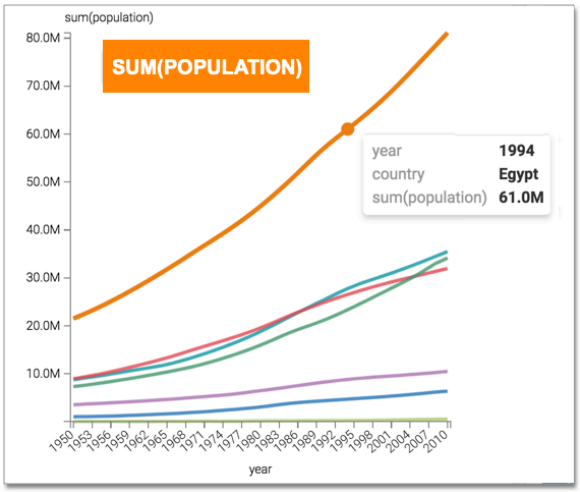

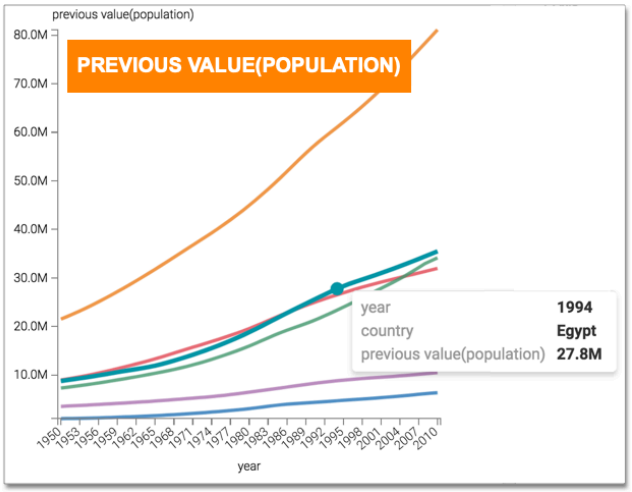

Difference from previous value analytic function

This article describes how to use the Difference from Previous Value analytic function in Cloudera Data Visualization.

Procedure

1. Click the field on the measurement shelf.

   In this example, the sum(population) field on the Y Axis shelf.

2. In the FIELD PROPERTIES menu, expand Analytic Functions, and select Difference from Previous.

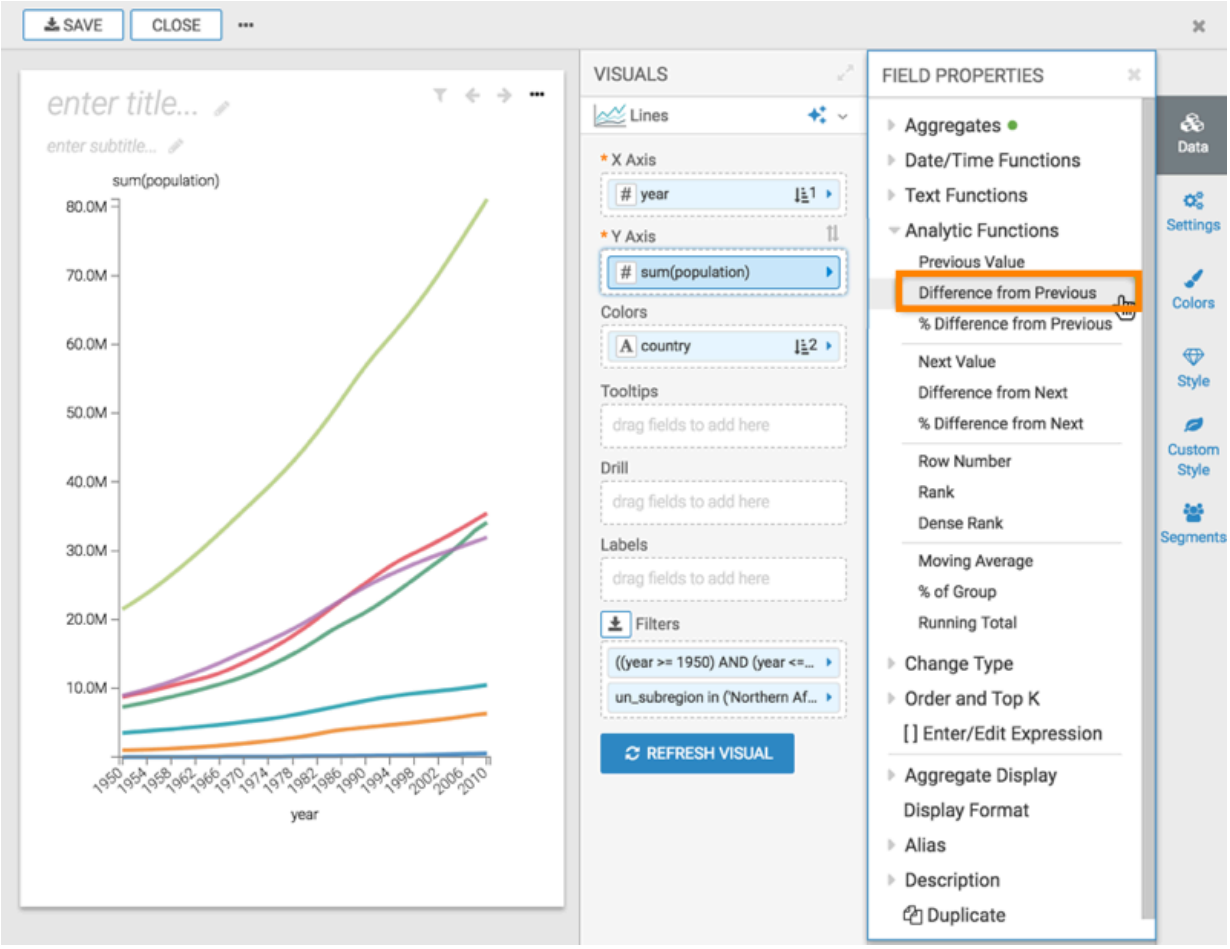

3. In the Analytic Function: Difference from Previous Value modal window:
   a. In Select aggregate for this measure, leave the default Sum. The other aggregation options are Count, Minimum, Maximum, and Average. In addition, some connection types also support Approximate Distinct Count and Exact Distinct Count.
   b. Under Select entities over which to take difference from previous value, select year. This is the grouping that applies when calculating the difference from previous values.
   c. Under Select the sort order, you can specify the order of sorting. This order specifies what the previous row is, and the row where the value is taken. If your visual already specifies sorting order, these values appear in the modal.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

      In this example, leave the default order: year ascending, followed by country ascending.
   d. Under Offset, specify the number of previous rows over which the function executes.

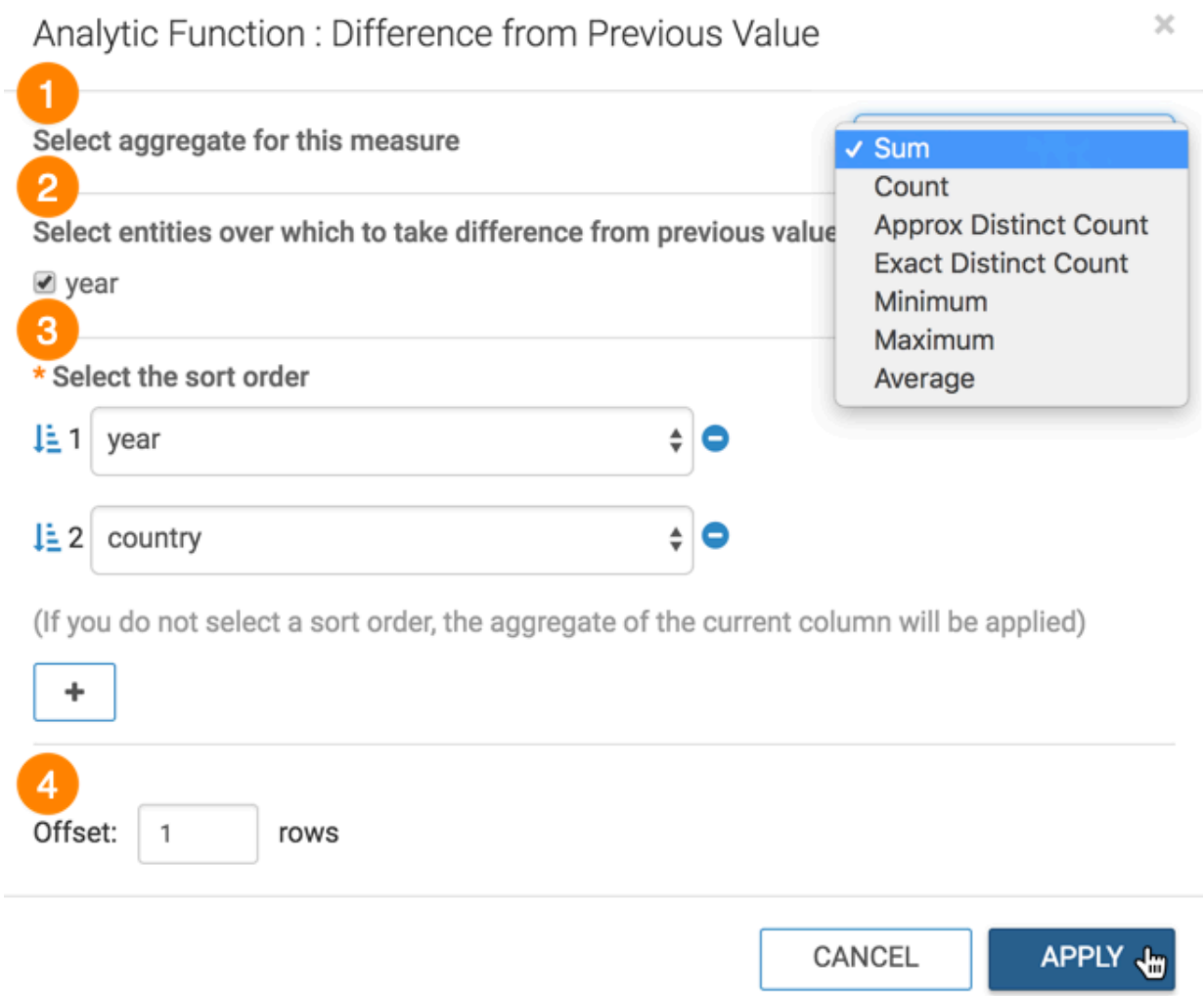

4. Click APPLY.
5. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form: sum([population]) - LAG(sum([population]), 1) over (partition by [year] order by [year], [country]) as ' difference from previous value(population)'

6. Click REFRESH VISUAL.

   You can see the change to the appearance of the visual, and the change in values that appear on the tooltip.

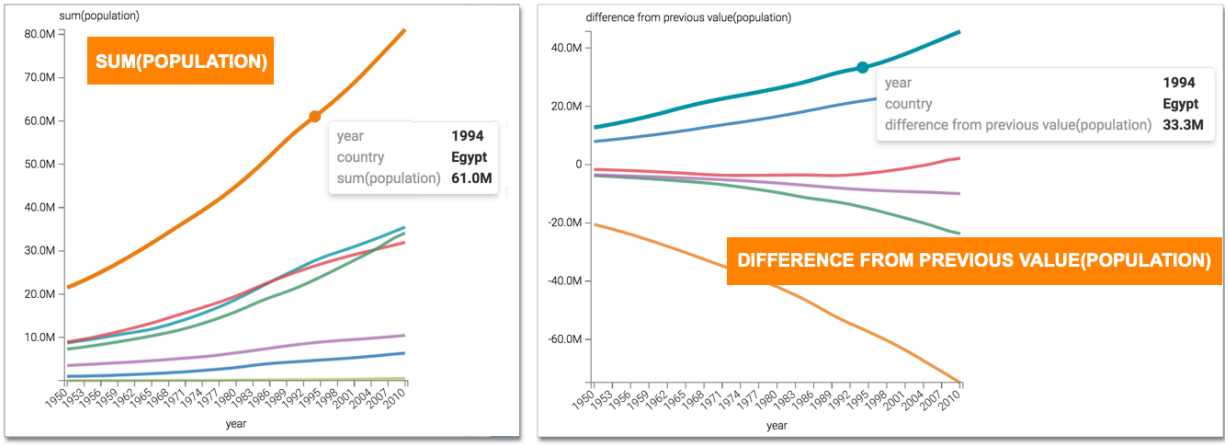

Percentage difference from previous value analytic function

This article describes how to use the Percent Difference from Previous Value analytic function in Cloudera Data Visualization.

Procedure

1. Click the field on the measurement shelf.

   In this example, the sum(population) field on the Y Axis shelf.

2. In the FIELD PROPERTIES menu, expand Analytic Functions, and select % Difference from Previous.

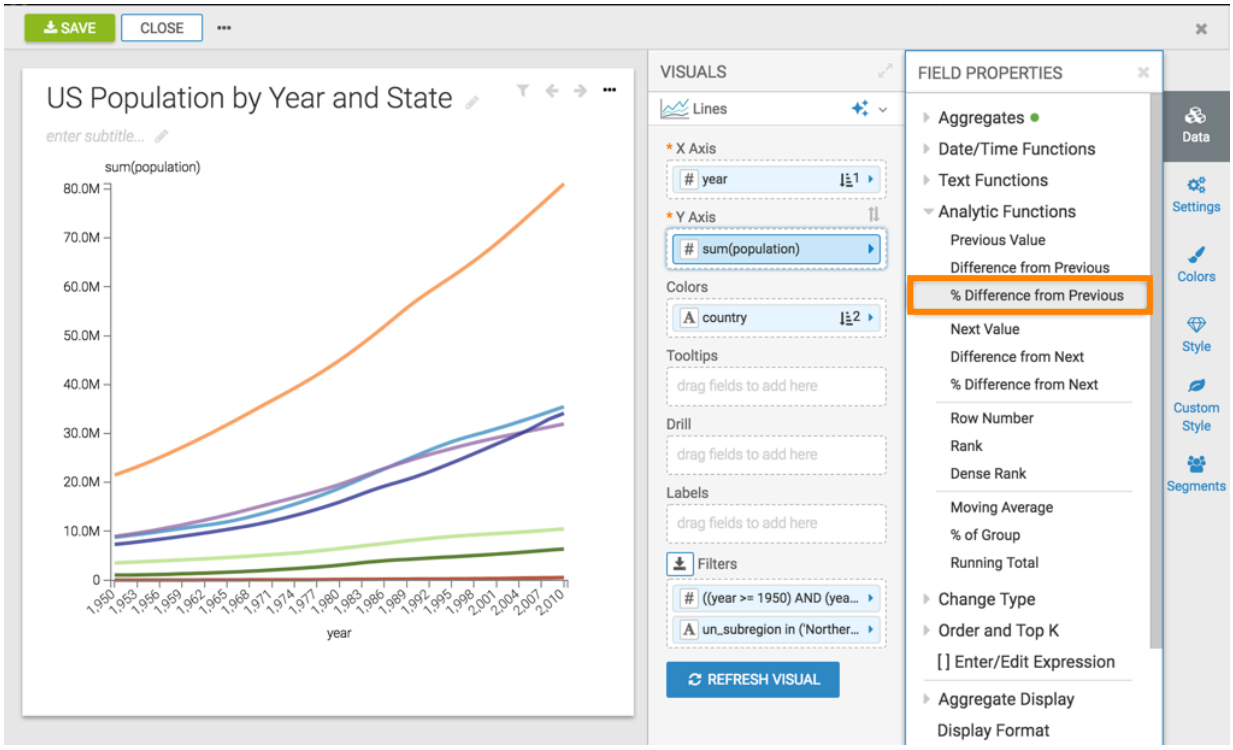

3. In the Analytic Function: Percentage Difference from Previous Value modal window:
   a. In Select aggregate for this measure, leave the default Sum. The other aggregation options are Count, Minimum, Maximum, and Average. In addition, some connection types also support Approximate Distinct Count and Exact Distinct Count.
   b. Under Select entities over which to take % difference from previous value, select country. This is the grouping that applies when calculating the % difference from previous values.
   c. Under Select the sort order, you can specify the order of sorting. This order specifies what the previous row is, and the row where the value is taken. If your visual already specifies sorting order, these values appear in the modal.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

      In this example, edit the sort order to be: country ascending, followed by year ascending.
   d. Under Offset, specify the number of previous rows over which the function executes.

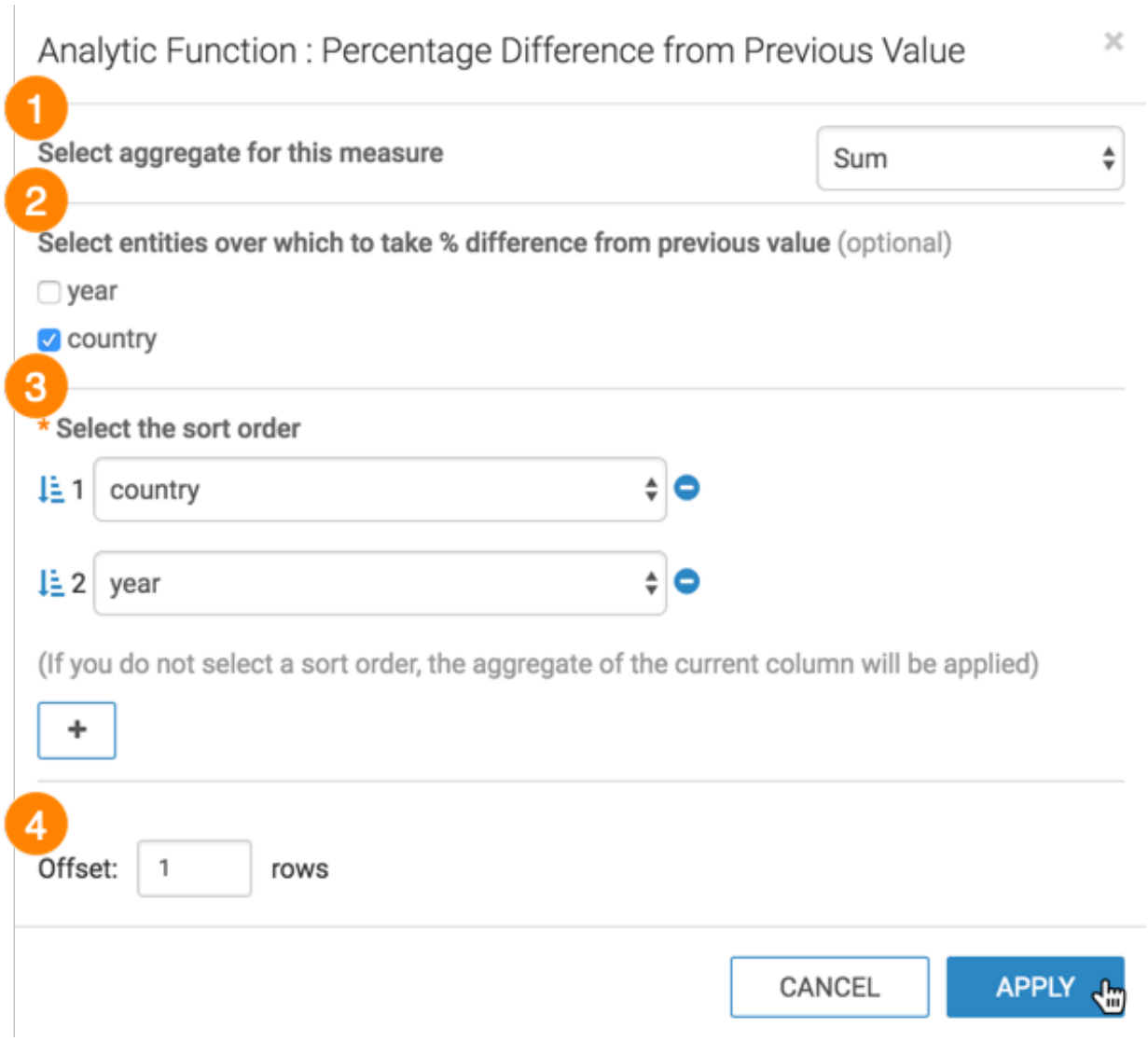

4. Click APPLY.
5. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form:

```
(sum([population]) - LAG(sum([population]), 1)
```

```
over (partition by [country] order by [country], [year])) / LAG(sum([pop
ulation]), 1)
over (partition by [country] order by [country], [year])
as 'pct diff from prev(population)'
```

6. Click REFRESH VISUAL.

   You can see the change in the appearance of the Y axis and the tooltip, from displaying the sum(population) in millions to population as a percentage difference from previous value.

   Hover over any country to view the percentage difference of population from the previous year. In the following example, the tooltip shows an increase in the population of Libya from previous year 1964 to the current year 1965 by 3.92%.

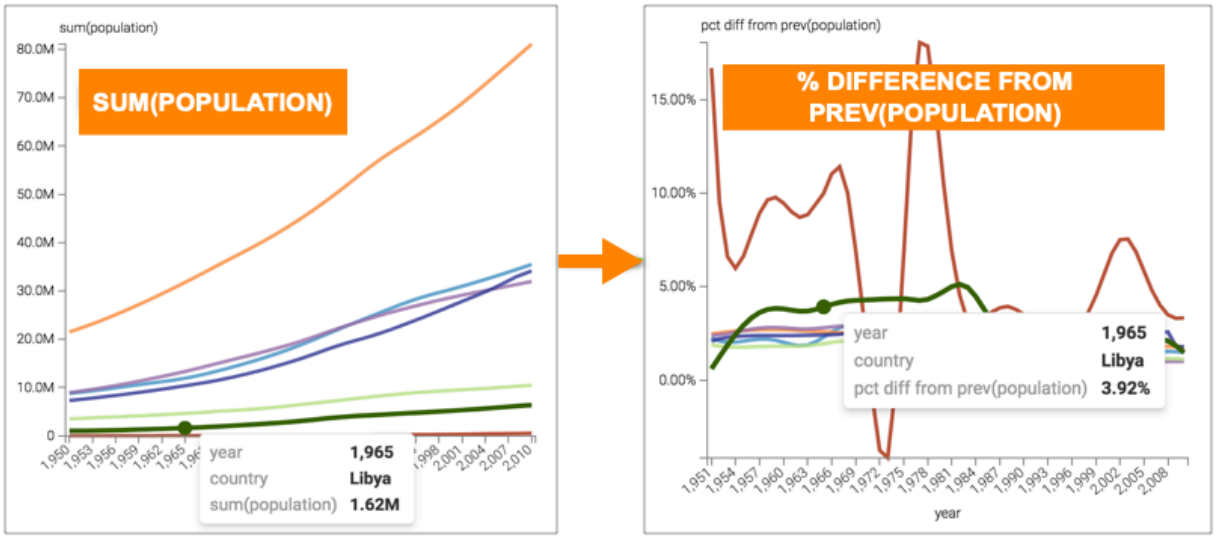

Next value analytic function

This article describes how to use the Next Value analytic function in Cloudera Data Visualization.

Procedure

1. Click the field on the measurement shelf.

   In this example, the sum(population) field on the Y Axis shelf.

2. In the FIELD PROPERTIES menu, expand Analytic Functions, and select Next Value.

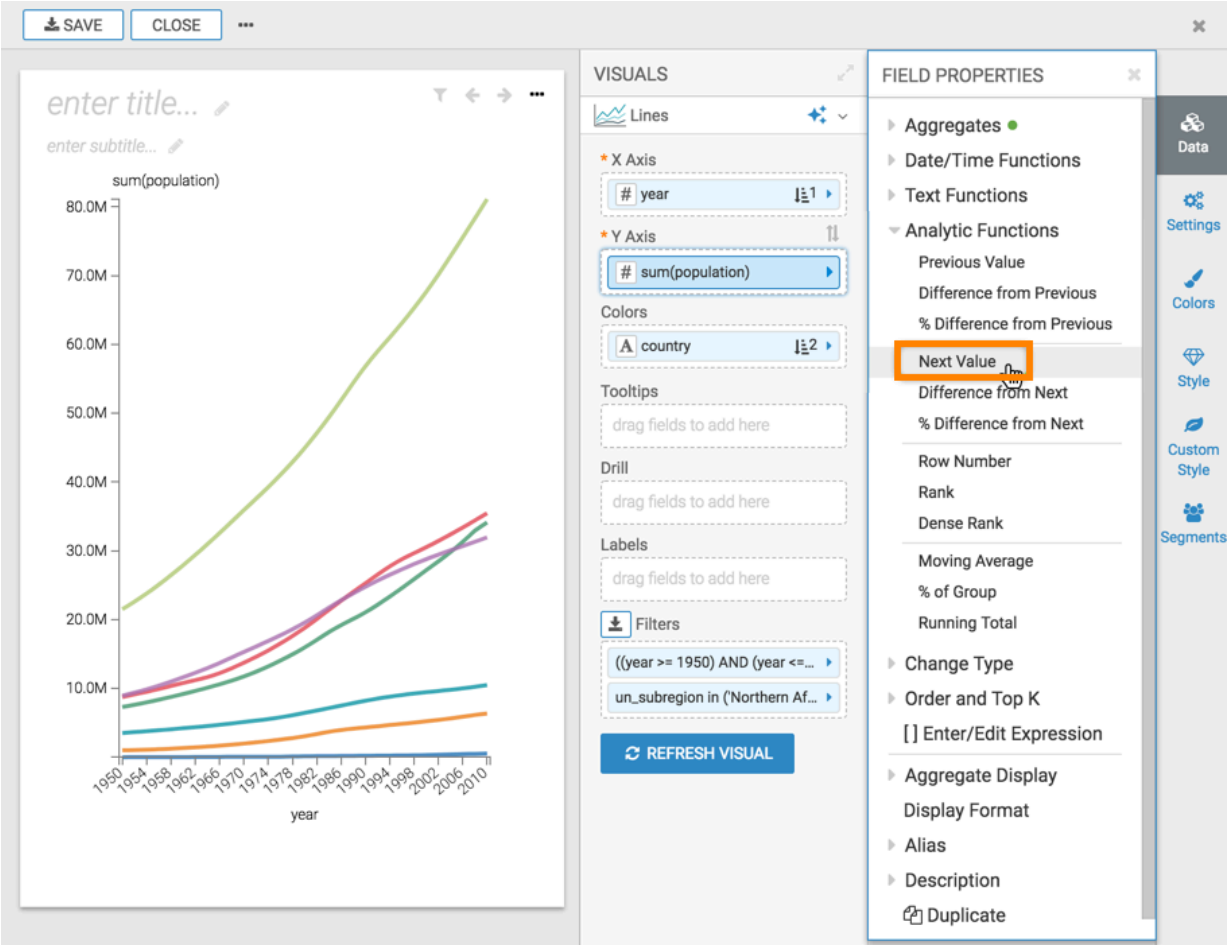

3. In the Analytic Function: Next Value modal window:
   a. In Select aggregate for this measure, leave the default Sum. The other aggregation options are Count, Minimum, Maximum, and Average. In addition, some connection types also support Approximate Distinct Count and Exact Distinct Count.
   b. Under Select entities over which to tale next value, select state. This is the grouping that applies when calculating next values.
   c. Under Select the sort order, click the (Plus) icon, select year, and then set it to ascending sort order. This order specifies what determines what the next row is, and the row where the value is taken.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

   d. Under Offset, specify the number of next rows over which the function executes.

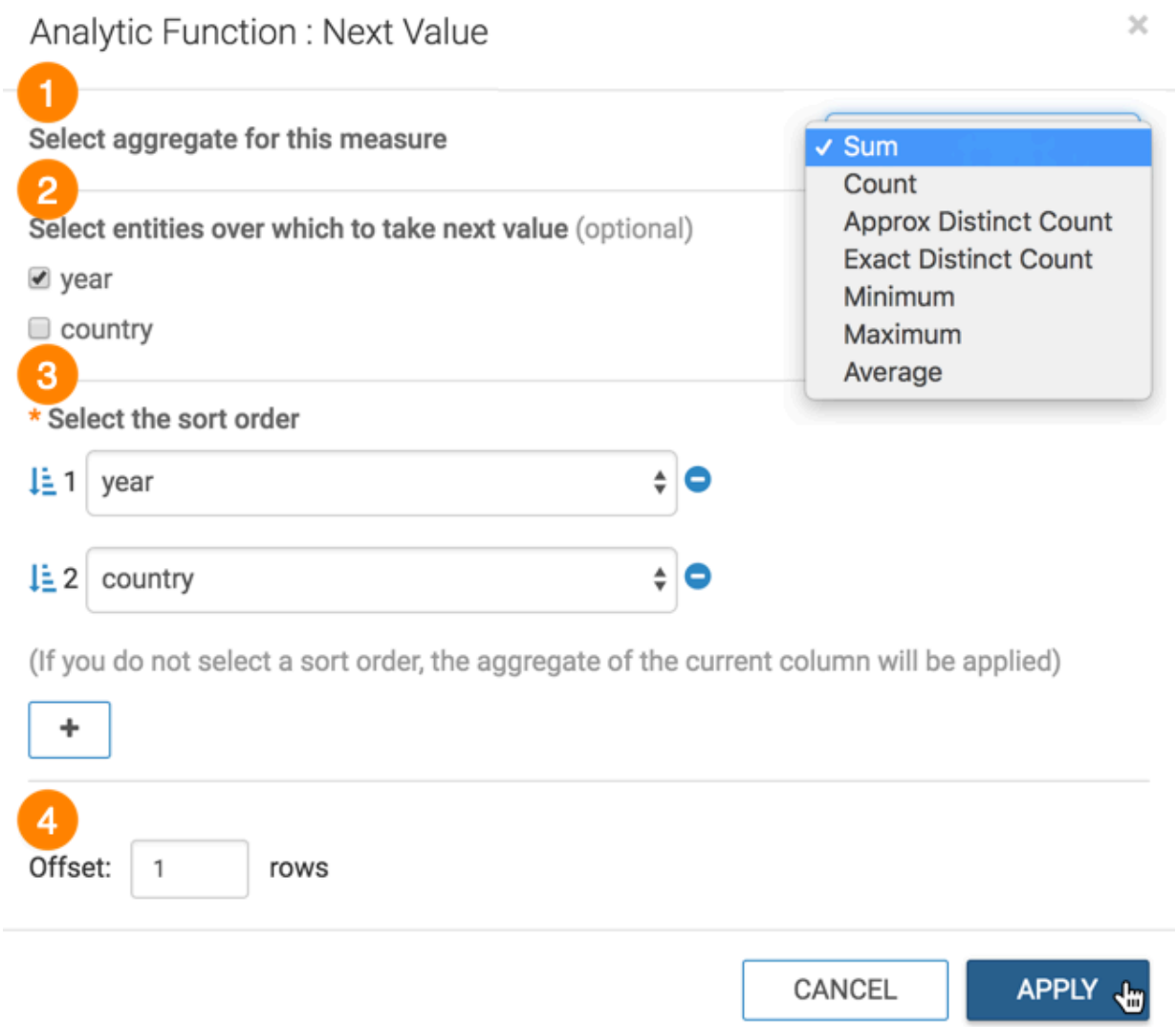

4. Click APPLY.
5. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form:

```
LAG(sum([population]), 1) over (partition by [year] order by [year], [co
untry])
            as 'next value(population)'
```

6. Click REFRESH VISUAL.

   You can see the change to the appearance of the visual, and the change in values that appear on the tooltip.

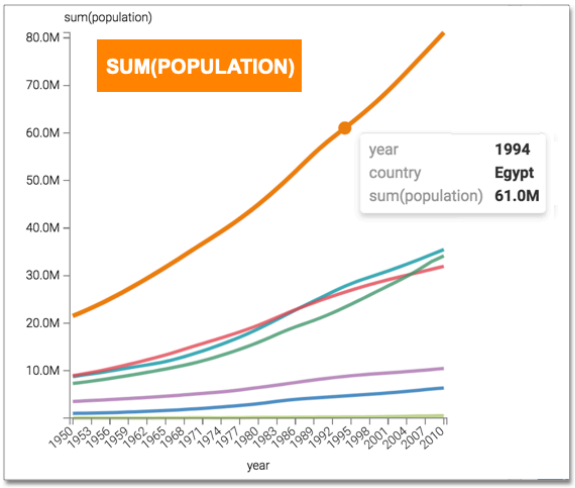

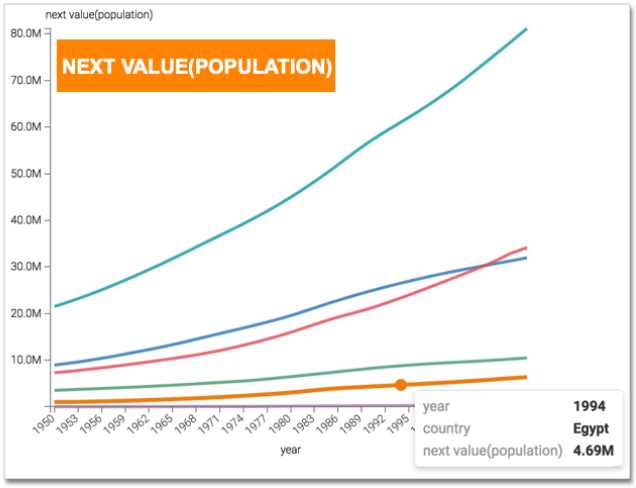

Difference from next value analytic function

This article describes how to use the Difference from Next Value analytic function in Cloudera Data Visualization.

Procedure

1. Click the field on the measurement shelf.

   In this example, the sum(population) field on the Y Axis shelf.

2. In the FIELD PROPERTIES menu, expand Analytic Functions, and select Difference from Next.

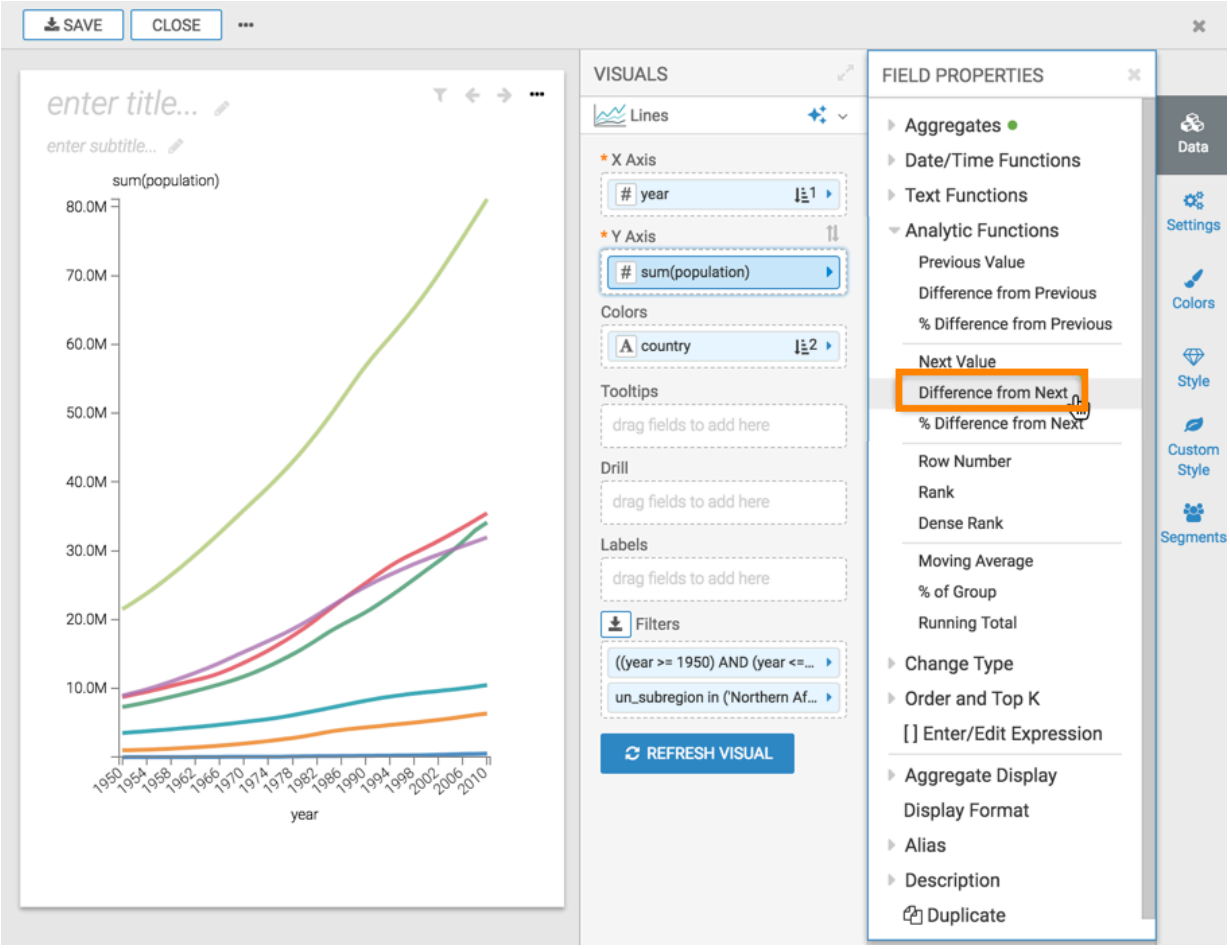

3. In the Analytic Function: Difference from Next Value modal window,
   a. In Select aggregate for this measure, leave the default Sum. The other aggregation options are Count, Minimum, Maximum, and Average. In addition, some connection types also support Approximate Distinct Count and Exact Distinct Count.
   b. Under Select entities over which to take difference from next value, select year. This is the grouping that applies when calculating the difference from next values.
   c. Under Select the sort order, you can specify the order of sorting. This order specifies what the next row is, and the row where the value is taken. If your visual already specifies sorting order, these values appear in the modal.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

      In this example, leave the default order: year ascending, followed by country ascending.
   d. Under Offset, specify the number of next rows over which the function executes.

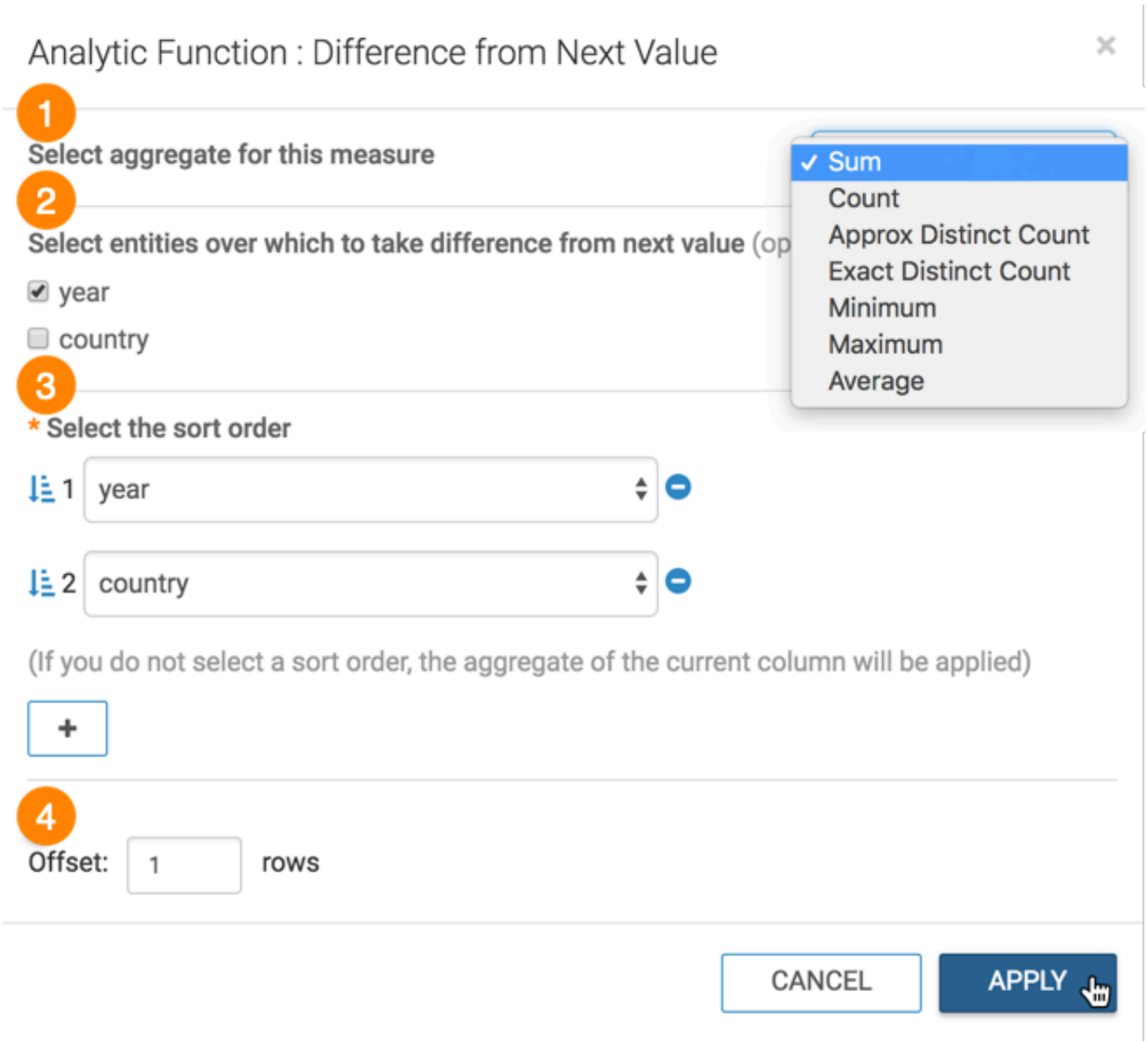

4. Click APPLY.
5. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form:

```
sum([population]) - LEAD(sum([population]), 1) over (partition by [year]
```

```
                order by [year], [country])
                as 'diff from next(population)'
```

6. Click REFRESH VISUAL.

   You can see the change to the appearance of the visual, and the change in values that appear on the tooltip.

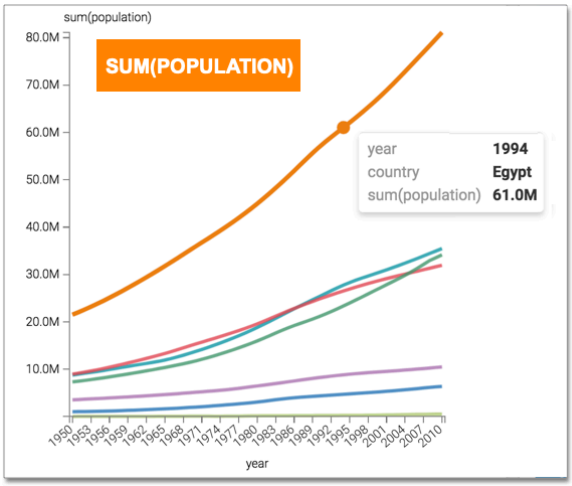

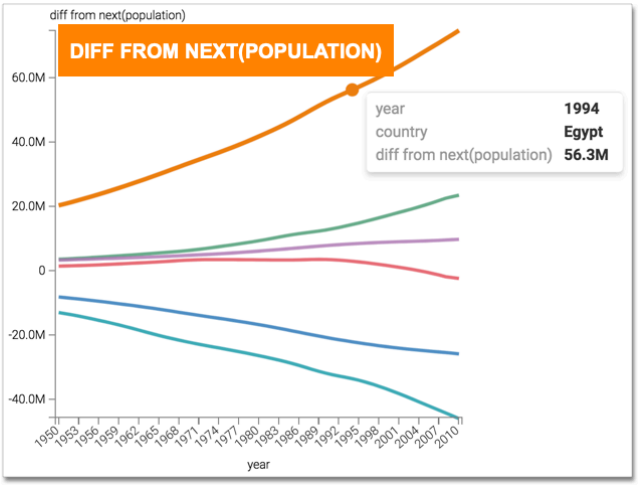

Percentage difference from next value analytic function

This article describes how to use the Percent Difference from Next Value analytic function in Cloudera Data Visualization.

Procedure

1. Click the field on the measurement shelf.

   In this example, the sum(population) field on the Y Axis shelf.

2. In the FIELD PROPERTIES menu, expand Analytic Functions, and select % Difference from Next.

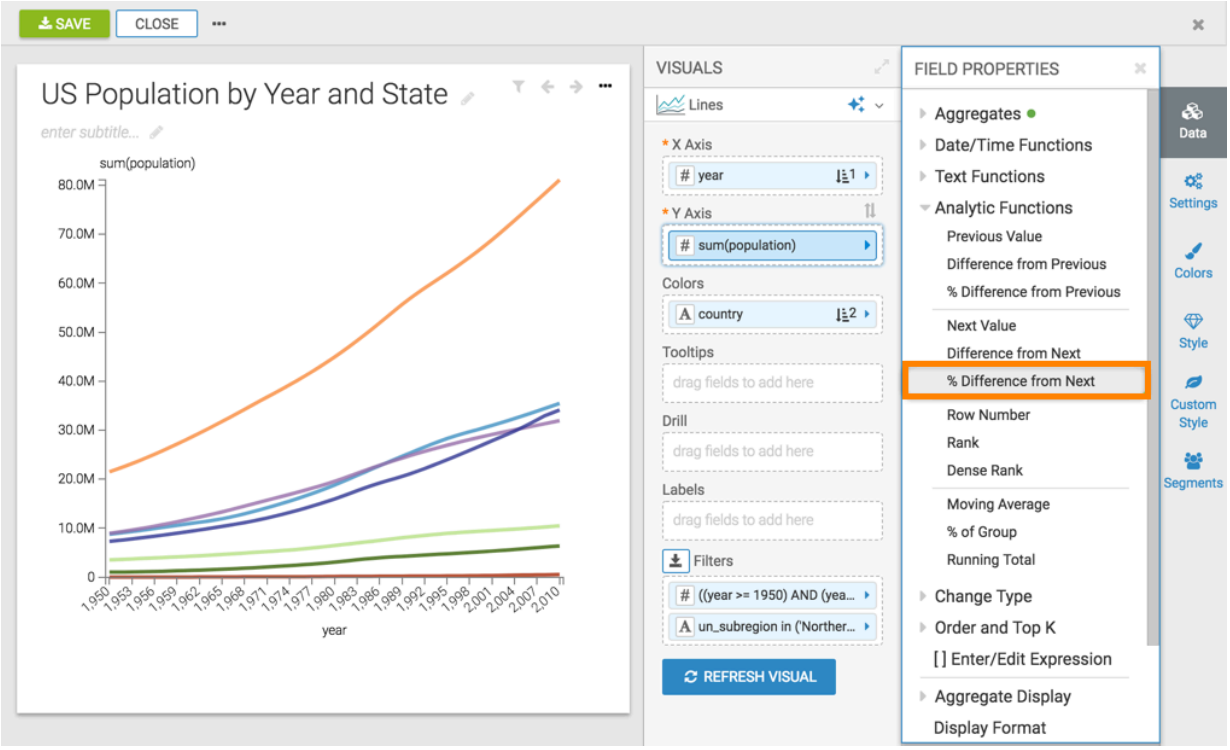

3. In the Analytic Function: Percentage Difference from Next Value modal window:
   a. In Select aggregate for this measure, leave the default Sum. The other aggregation options are Count, Minimum, Maximum, and Average. In addition, some connection types also support Approximate Distinct Count and Exact Distinct Count.
   b. Under Select entities over which to take % difference from next value, select country. This is the grouping that applies when calculating the % difference from next values.
   c. Under Select the sort order, you can specify the order of sorting. This order specifies what the next row is, and the row where the value is taken. If your visual already specifies sorting order, these values appear in the modal.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

      In this example, edit the sort order to be: country ascending, followed by year ascending.
   d. Under Offset, specify the number of next rows over which the function executes.

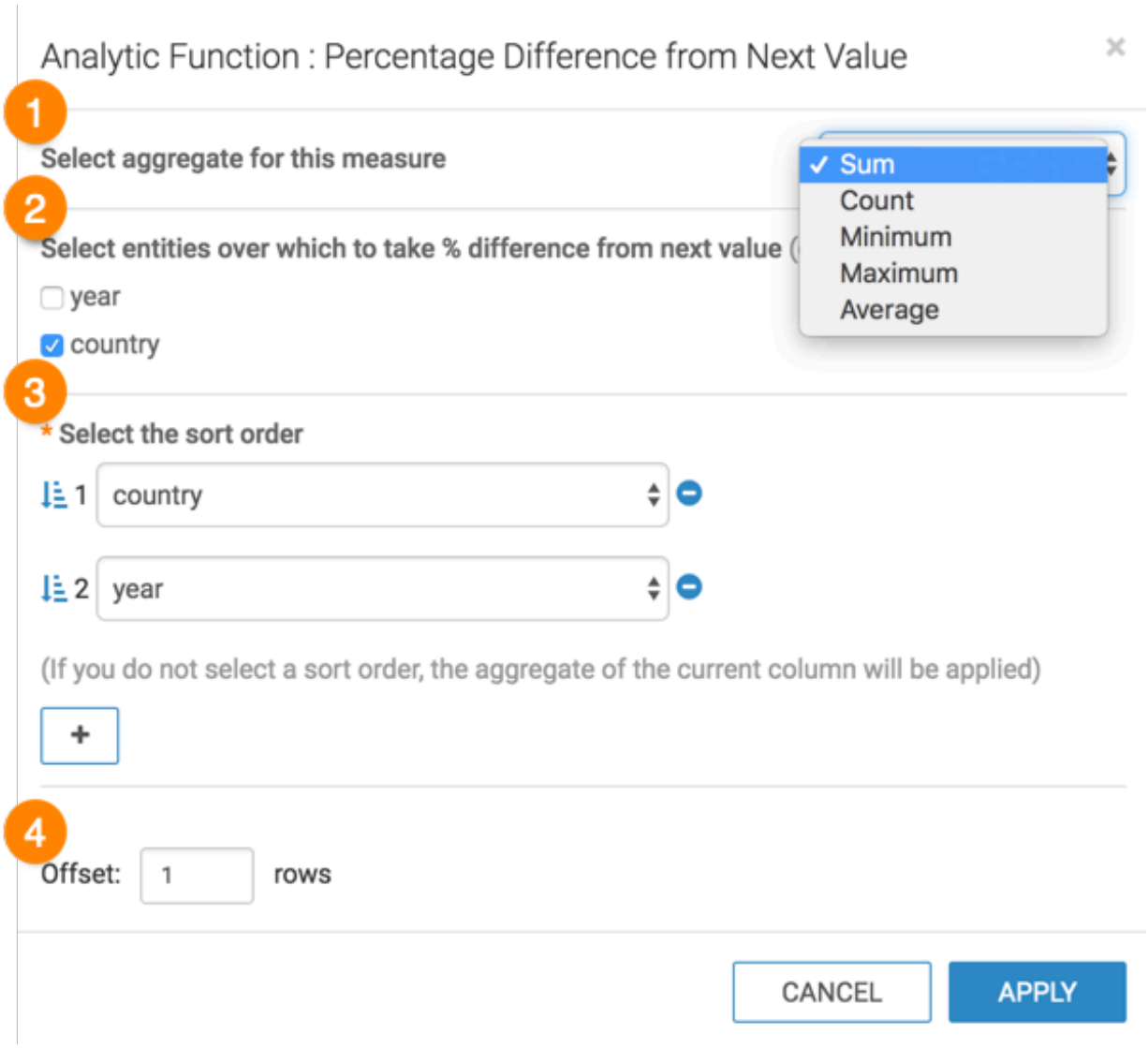

4. Click APPLY.
5. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form:

```
(sum([population]) - LAG(sum([population]), 1)
```

```
            over (partition by [country] order by [country], [year])) /
LAG(sum([population]), 1)
            over (partition by [country] order by [country], [year])
            as 'pct diff from next(population)'
```

6. Click REFRESH VISUAL.

   You can see the change in the Y axis and tooltips, from displaying the sum(population) in millions to population as a percentage difference from next value.

   Hover over any country to view the percentage difference of population from the next year. In the following example, the tooltip shows a decrease in the population of Libya from 1965 to next year 1966 by 3.90%.

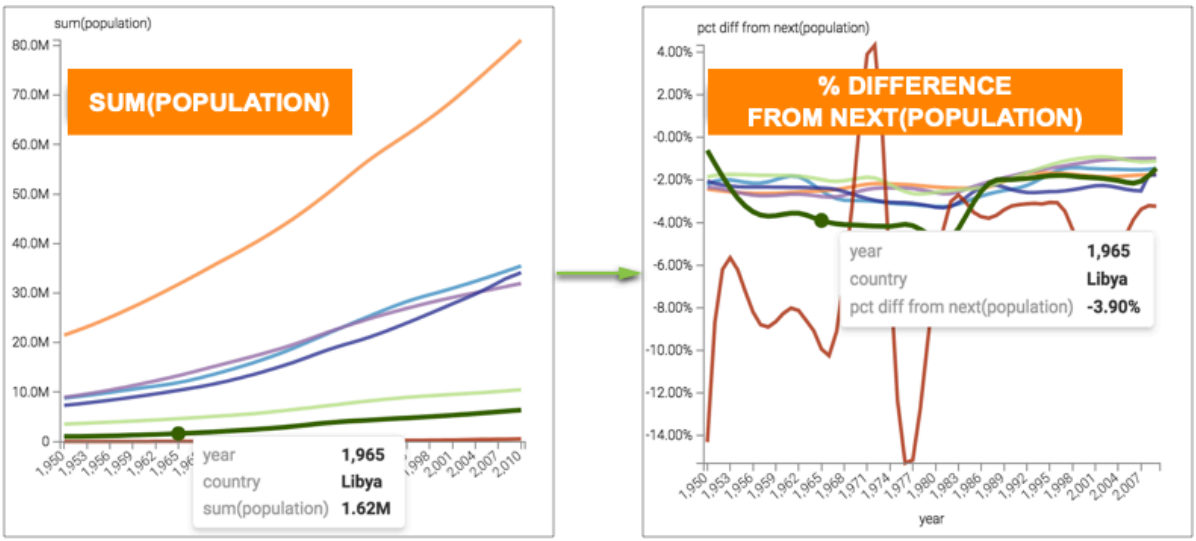

Row number function

This article describes how to use the Row Number analytic function in Cloudera Data Visualization.

Procedure

1. Click a field on the dimension shelf. In this case, choose the population field.
2. In the FIELD PROPERTIES menu, select Duplicate.
3. Click the second population field.

4. In FIELD PROPERTIES, expand Analytic Functions, and select Row Number.

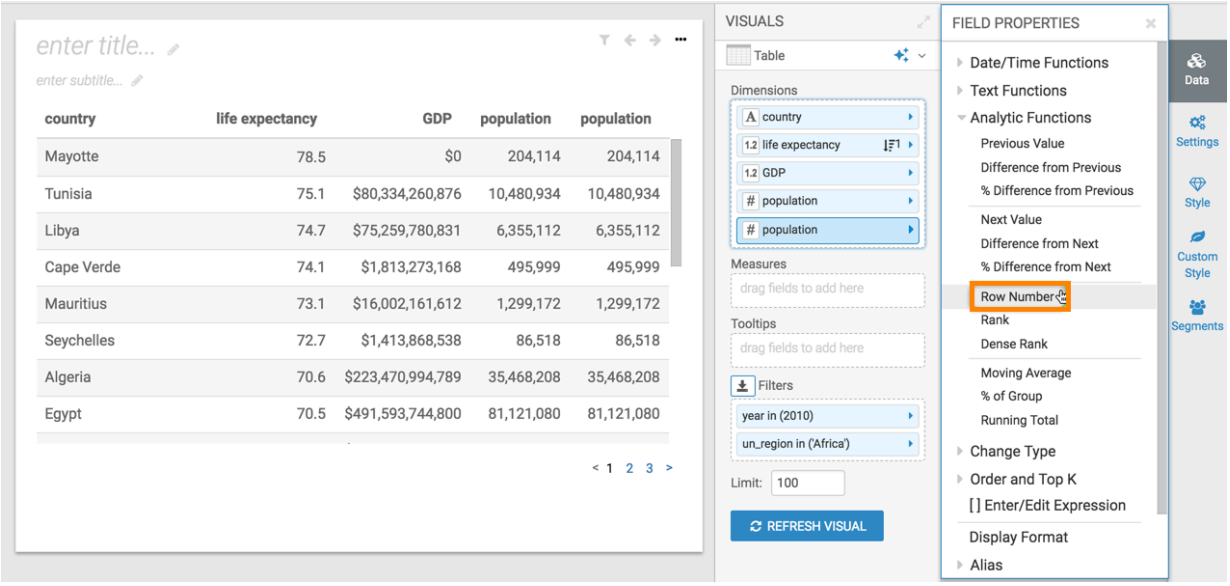

5. In the Analytic Function: Row Number modal window:
   a. Under Select entities over which to calculate row number, you can select any of the fields: country, life expectancy, or GDP. This is the grouping that applies when calculating the row number.
   b. Under Select the sort order, you can specify the order of sorting. This order specifies what the row is, and the row where the value is taken. If your visual already specifies sorting order, these values appear in the modal.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

      In this example, leave the default order: life expectancy descending, followed by country and GDP, both ascending.

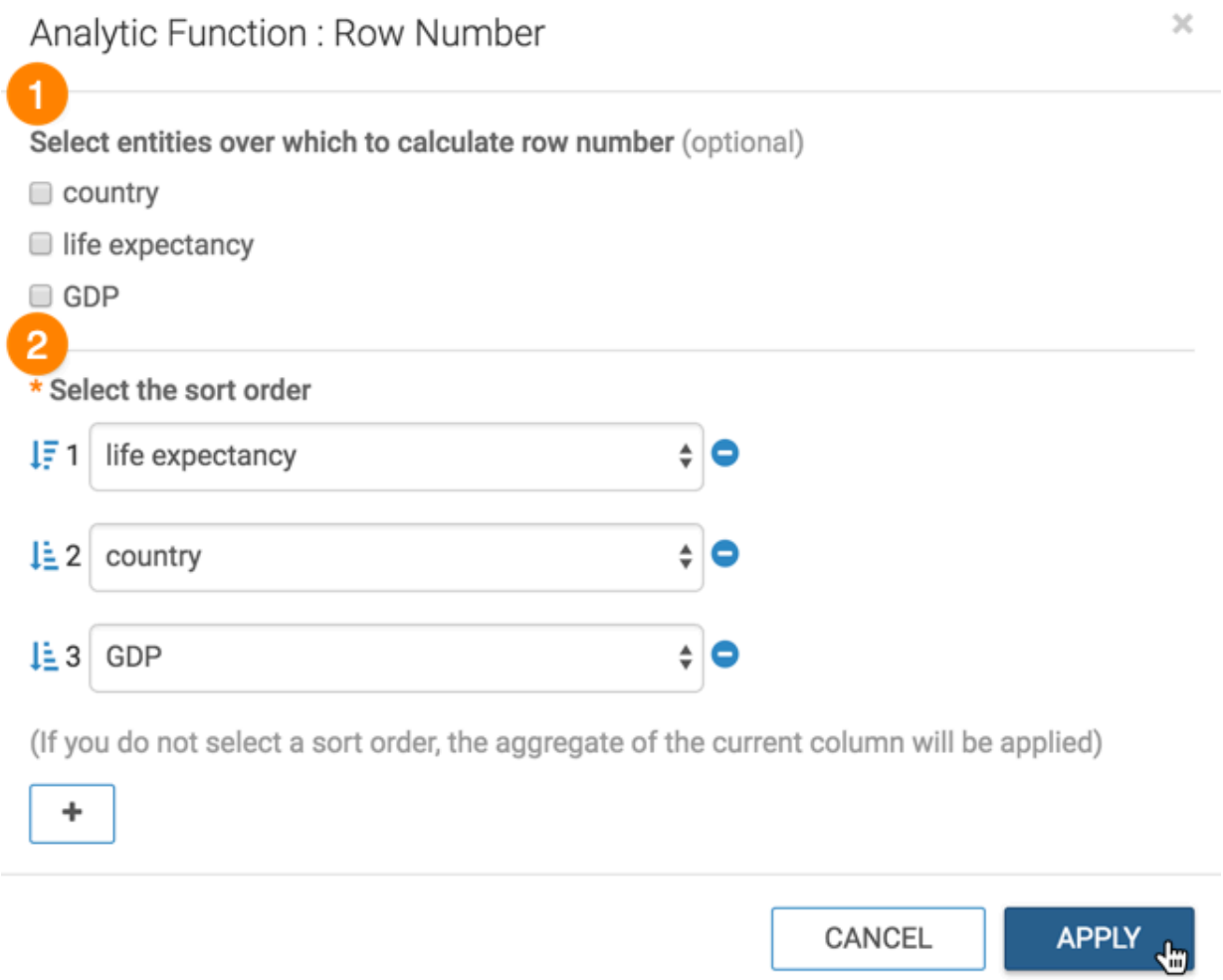

6. Click APPLY.
7. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form:

```
ROW_NUMBER() over ( order by [life_expectancy], [country], [gdp_per_capi
ta]*[population]) as 'row number'
```

8. Click REFRESH VISUAL.

   You can see that initially, the row number numbers are descending, following the order of the primary sorting column, life expectancy. If you sort the table on another column, such as population, the row number numbers appear disordered.

country	life expectancy	GDP	population	row number
Mayotte	78.5	$0	204,114	52
Tunisia	75.1	$80,334,260,876	10,480,934	51
Libya	74.7	$75,259,780,831	6,355,112	50
Cape Verde	74.1	$1,813,273,168	495,999	49
Mauritius	73.1	$16,002,161,612	1,299,172	48
Seychelles	72.7	$1,413,868,538	86,518	47
Algeria	70.6	$223,470,994,789	35,468,208	46
Egypt	70.5	$491,593,744,800	81,121,080	45

country	life expectancy	GDP	population	row number
Nigeria	51.3	$363,945,588,339	158,423,184	9
Ethiopia	61.5	$72,315,411,447	82,949,544	36
Egypt	70.5	$491,593,744,800	81,121,080	45
Congo	49.0	$24,644,821,788	65,965,796	5
South Africa	54.5	$467,348,147,107	50,132,816	19
Kenya	59.6	$59,606,308,062	40,512,680	29
Algeria	70.6	$223,470,994,789	35,468,208	46
Sudan	61.5	$88,655,959,426	34,136,520	35

Rank analytic function

This article describes how to use the Rank analytic function in Cloudera Data Visualization.

Procedure

1. Click a field on the dimension shelf. In this case, choose the population field.
2. In the FIELD PROPERTIES menu, select Duplicate.
3. Click the second population field.
4. In FIELD PROPERTIES, expand Analytic Functions, and select Rank.

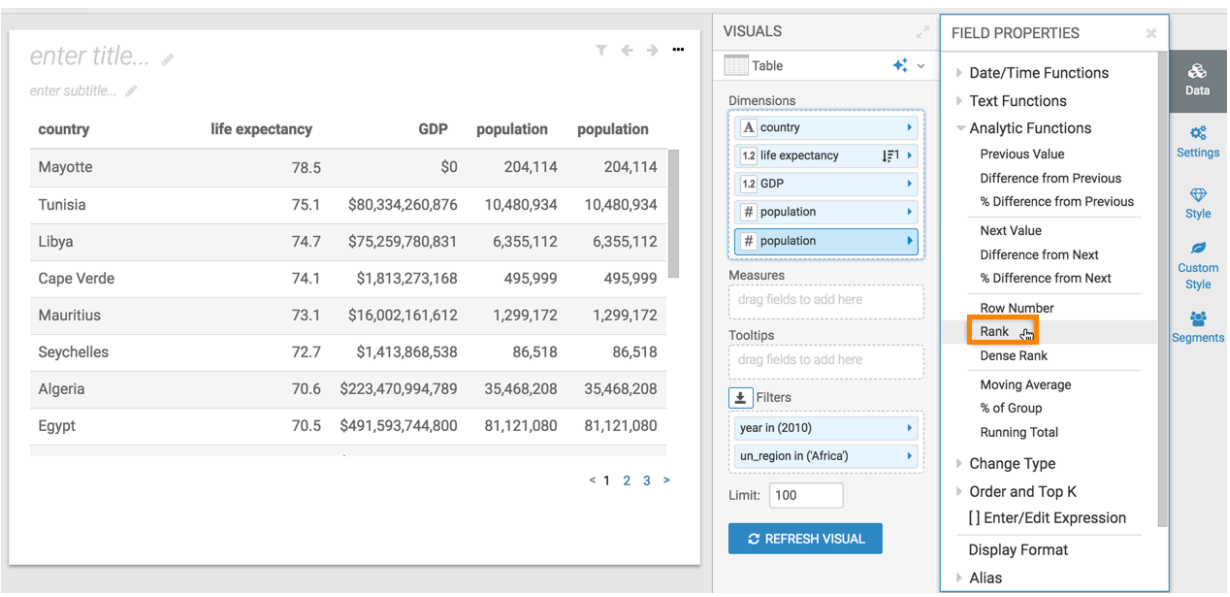

5. In the Analytic Function: Rank modal window:
   a. Under Select entities over which to calculate rank, you can select any of the fields: country, life expectancy, or GDP. This is the grouping that applies when calculating the rank.
   b. Under Select the sort order, you can specify the order of sorting. This order specifies what the row is, and the row where the value is taken. If your visual already specifies sorting order, these values appear in the modal.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

      In this example, leave the default order: life expectancy descending, followed by country and GDP, both ascending.

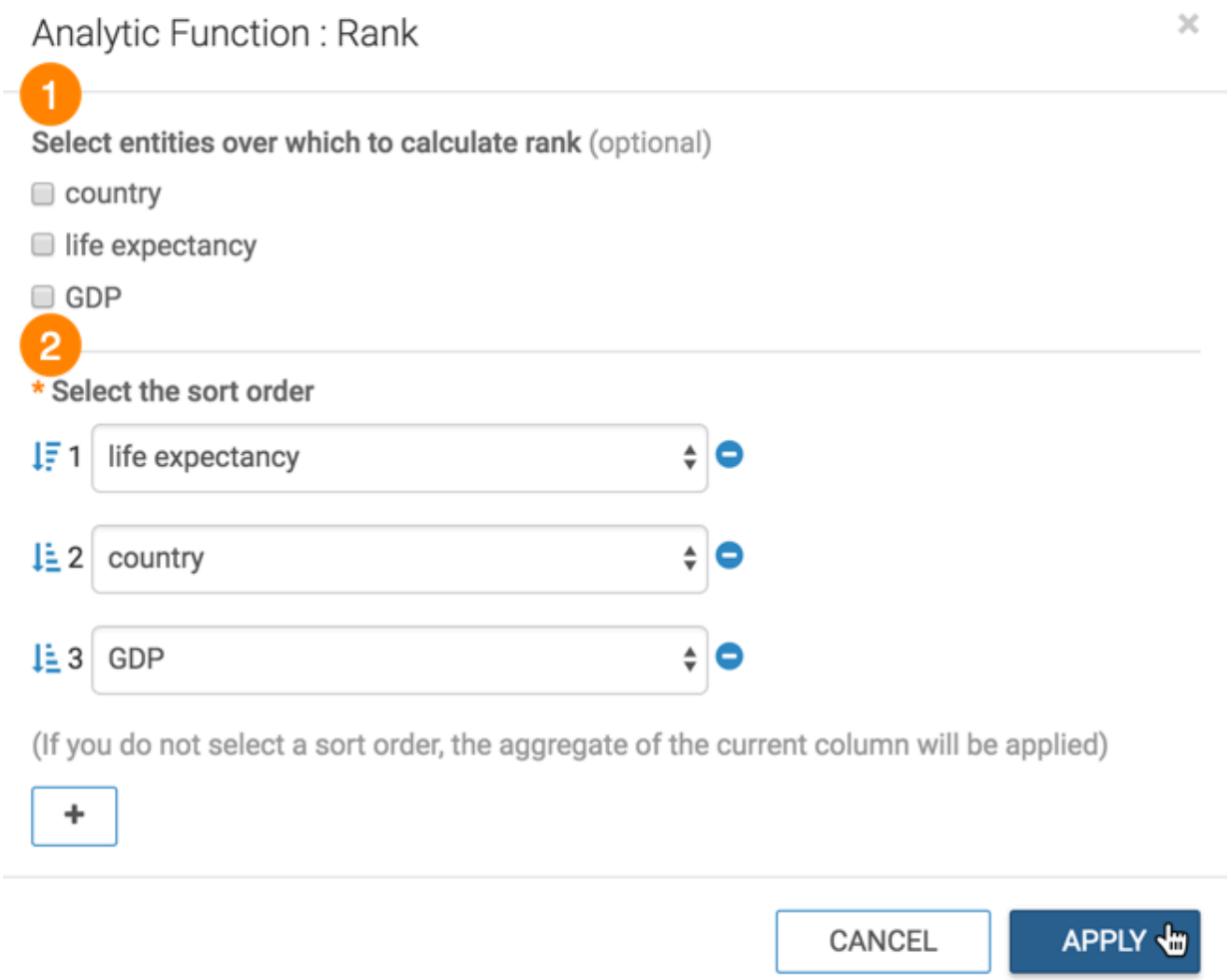

6. Click APPLY.
7. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form:

```
RANK() over ( order by [life_expectancy], [country], [gdp_per_capita]*[p
opulation]) as 'rank'
```

8. Click REFRESH VISUAL.

   You can see that initially, the rank numbers are descending, following the order of the primary sorting column, life expectancy. If you sort the table on another column, such as GDP, the rank numbers appear disordered.

country	life expectancy	GDP	population	rank
Mayotte	78.5	$0	204,114	52
Tunisia	75.1	$80,334,260,876	10,480,934	51
Libya	74.7	$75,259,780,831	6,355,112	50
Cape Verde	74.1	$1,813,273,168	495,999	49
Mauritius	73.1	$16,002,161,612	1,299,172	48
Seychelles	72.7	$1,413,868,538	86,518	47
Algeria	70.6	$223,470,994,789	35,468,208	46
Egypt	70.5	$491,593,744,800	81,121,080	45

< 1 2 3 >

country	life expectancy	GDP	population	rank
Egypt	70.5	$491,593,744,800	81,121,080	45
South Africa	54.5	$467,348,147,107	50,132,816	19
Nigeria	51.3	$363,945,588,339	158,423,184	9
Algeria	70.6	$223,470,994,789	35,468,208	46
Morocco	70.2	$136,215,265,879	31,951,412	44
Angola	50.7	$104,904,721,275	19,081,912	8
Sudan	61.5	$88,655,959,426	34,136,520	35
Tunisia	75.1	$80,334,260,876	10,480,934	51

< 1 2 3 >

Dense rank analytic function

This article describes how to use the Dense Rank analytic function in Cloudera Data Visualization.

Procedure

1. Click a field on the dimension shelf.

   In this example, choose the population field.
2. In the FIELD PROPERTIES menu, select Duplicate.
3. Click the second population field.
4. In FIELD PROPERTIES, expand Analytic Functions, and select Dense Rank.

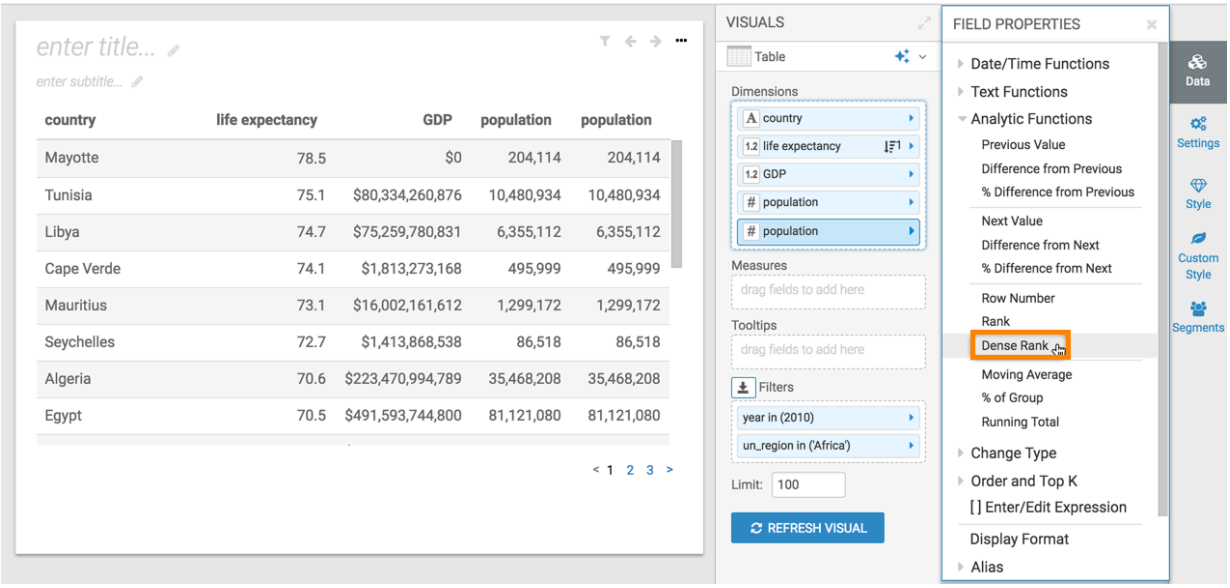

5. In the Analytic Function: Dense Rank modal window:
   a. Under Select entities over which to calculate rank, you can select any of the fields: country, life expectancy, or GDP. This is the grouping that applies when calculating the dense rank.
   b. Under Select the sort order, you can specify the order of sorting. This order specifies what the row is, and the row where the value is taken. If your visual already specifies sorting order, these values appear in the modal.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

      In this example, leave the default order: life expectancy descending, followed by country and GDP, both ascending.

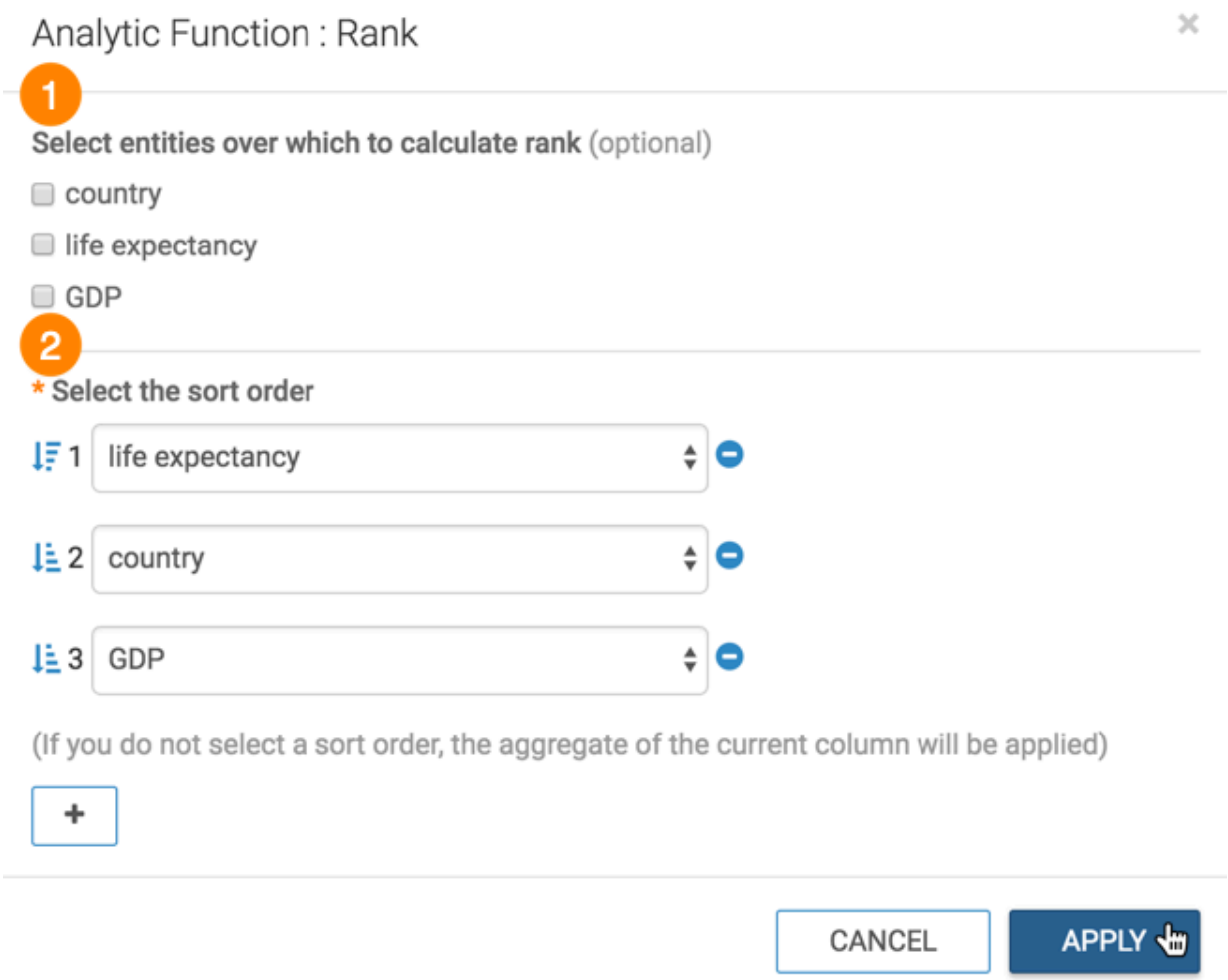

6. Click APPLY.
7. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form:

```
DENSE_RANK() over ( order by [life_expectancy], [country], [gdp_per_capi
ta]*[population]) as 'dense rank'
```

8. Click REFRESH VISUAL.

   You can see that initially, the dense rank numbers are descending, following the order of the primary sorting column, life expectancy. If you sort the table on another column, such as population, the dense rank numbers appear disordered.

country	life expectancy	GDP	population	dense rank
Mayotte	78.5	$0	204,114	52
Tunisia	75.1	$80,334,260,876	10,480,934	51
Libya	74.7	$75,259,780,831	6,355,112	50
Cape Verde	74.1	$1,813,273,168	495,999	49
Mauritius	73.1	$16,002,161,612	1,299,172	48
Seychelles	72.7	$1,413,868,538	86,518	47
Algeria	70.6	$223,470,994,789	35,468,208	46
Egypt	70.5	$491,593,744,800	81,121,080	45

country	life expectancy	GDP	population	dense rank
Seychelles	72.7	$1,413,868,538	86,518	47
Sao Tome and Principe	65.9	$303,586,194	165,397	42
Mayotte	78.5	$0	204,114	52
Cape Verde	74.1	$1,813,273,168	495,999	49
Western Sahara	66.8	$0	530,500	43
Equatorial Guinea	51.5	$9,374,026,767	700,401	10
Comoros	60.2	$751,355,332	734,750	30
Djibouti	60.3	$1,924,781,156	888,716	31

Moving average analytic function

This article describes how to use the Moving Average analytic function in Cloudera Data Visualization.

Procedure

1. Click the field on the measurement shelf.

   In this example, the sum(population) field on the Y Axis shelf.

2. In the FIELD PROPERTIES menu, expand Analytic Functions, and select Moving Average.

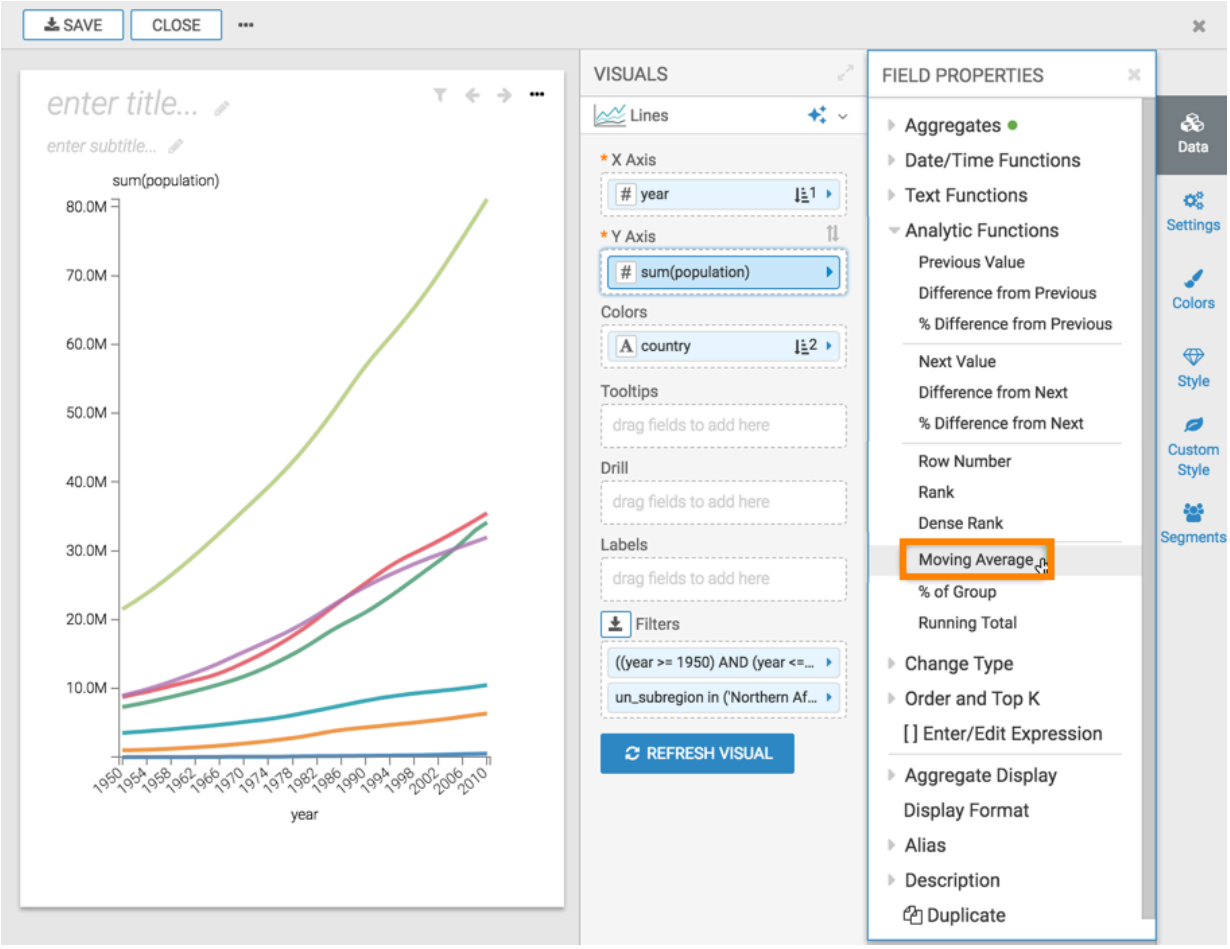

3. In the Analytic Function: Moving Average modal window:
   a. In Select aggregate for this measure, leave the default Sum. The other aggregation options are Count, Minimum, Maximum, and Average. In addition, some connection types also support Approximate Distinct Count and Exact Distinct Count.
   b. Under Select entities over which to calculate moving averages, select year. This is the grouping that applies when calculating the moving average.
   c. Under Select the sort order, you can specify the order of sorting. This order specifies what the row is, and the row where the value is taken. If your visual already specifies sorting order, these values appear in the modal.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

      In this example, leave the default order: year ascending, followed by country ascending.
   d. Under Rows to include in the average (relative to current row), the options for leading rows are: All previous rows, Current row, or Previous row(s) (default). Select the last option, then specify the number of preceding rows for the calculation; the default is 5, but we changed it to 2.

      Similarly, the options for following rows are: All forward rows, Current row (default), or Forward row(s). Select the last option; the default is 5, but we changed it to 2.

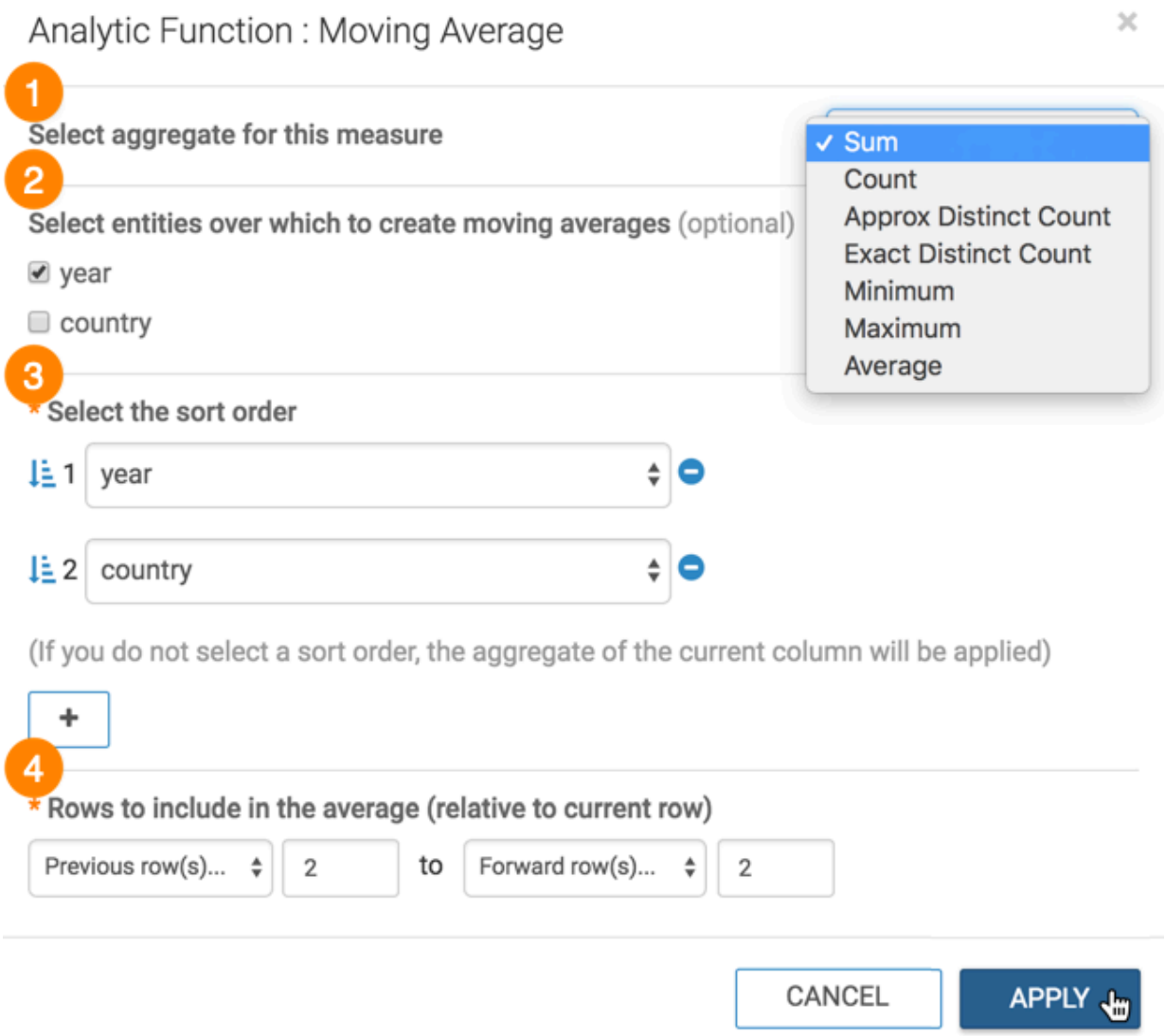

4. Click APPLY.

5. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form:

```
avg(sum([population])) over (partition by [year] order by [year], [count
ry]
             rows between 2 preceding and 2 following) as 'moving averag
e(population)'
```

6. Click REFRESH VISUAL.

   You can see the change to the appearance of the visual, and the change in values that appear on the tooltip.

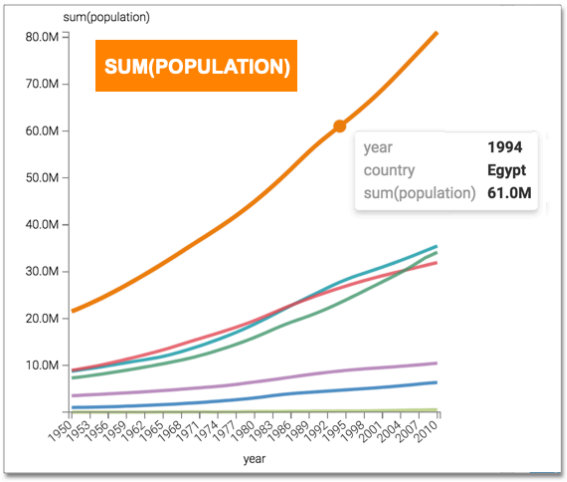

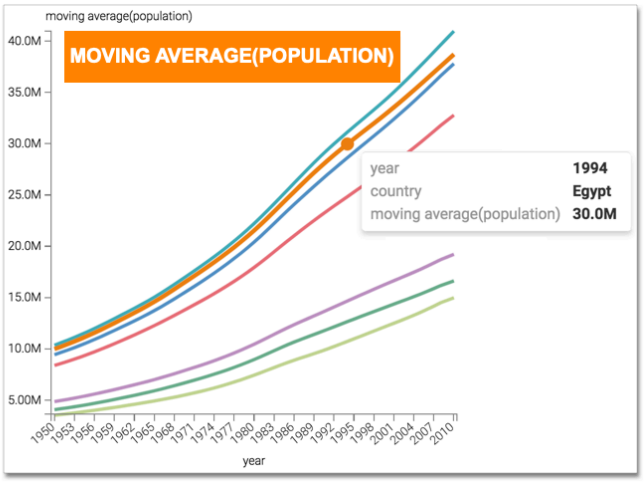

Percent of group analytic function

This article describes how to use the Percent of Group analytic function in Cloudera Data Visualization.

Procedure

1. Click the field on the measurement shelf.

   In this example, the sum(population) field on the Y Axis shelf.

2. In the FIELD PROPERTIES menu, expand Analytic Functions, and select % of Group.

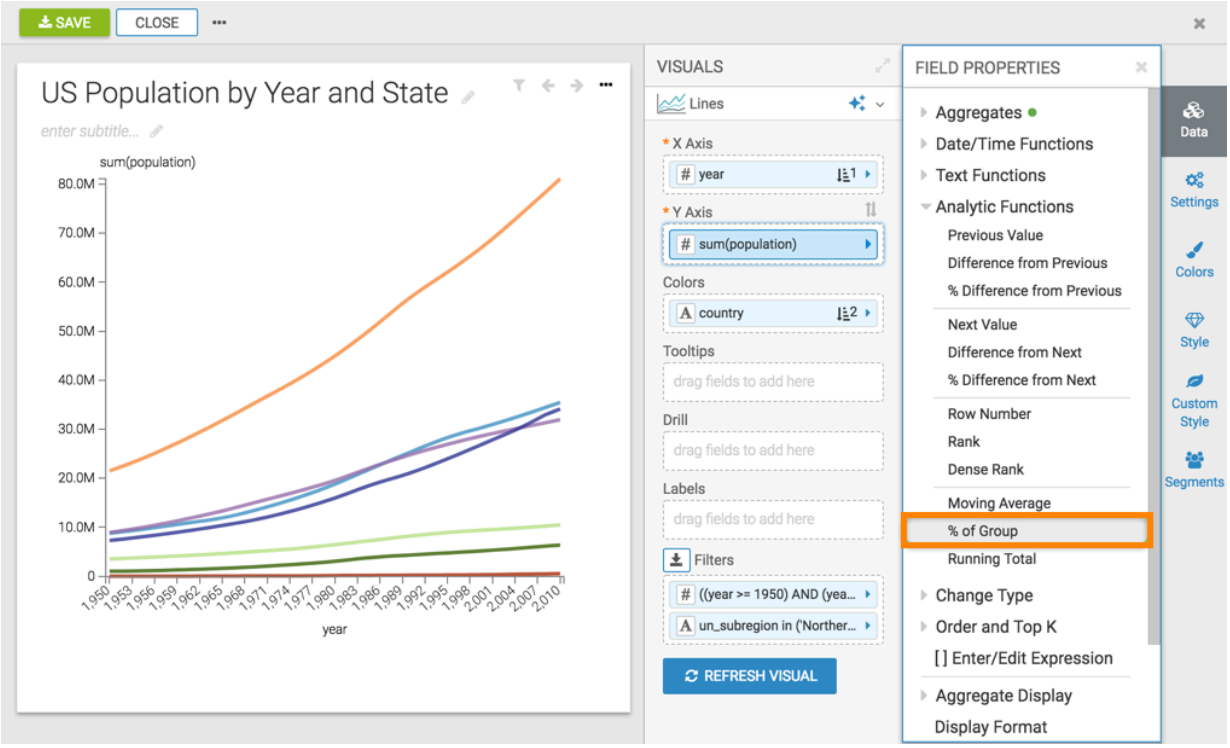

3. In the Analytic Function: Percent of Group modal window:

   a. In Select aggregate for this measure, leave the default Sum. The other aggregation options are Count, Minimum, Maximum, and Average. In addition, some connection types also support Approximate Distinct Count and Exact Distinct Count.

   b. Under Select entities over which to calculate percentage, select year. This is the grouping that applies when calculating the percent of group.

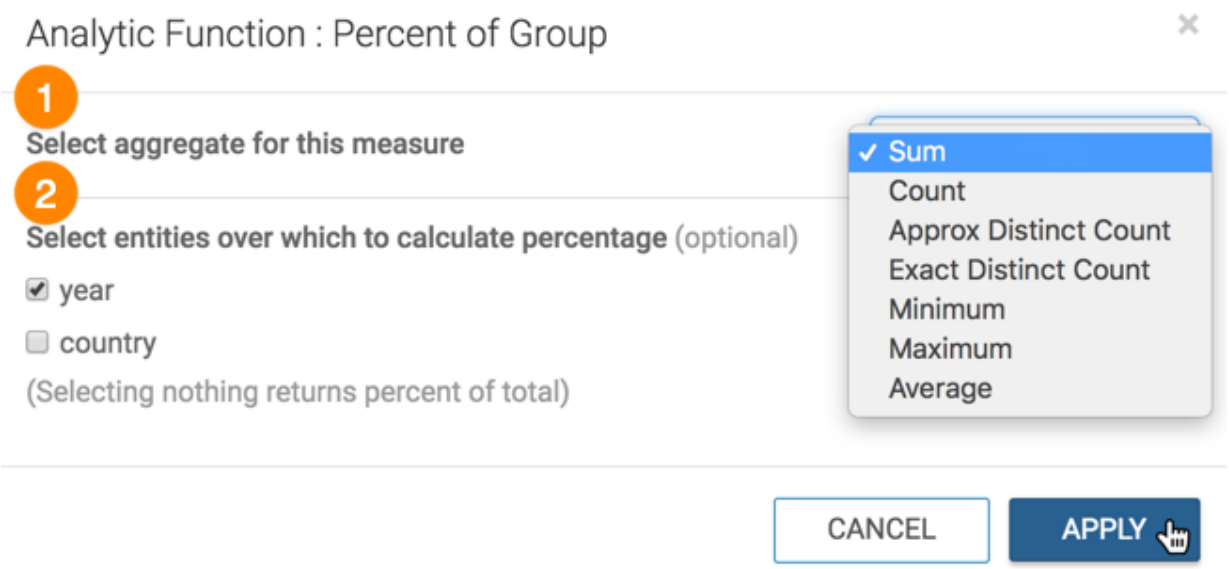

4. Click APPLY.

5. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form:

```
(sum([population]) / sum(sum([population])) over (partition by [year])) as
 'percent(population)'
```

6. Click REFRESH VISUAL.

   You can see the change in the Y axis and tooltips, from displaying the sum(population) in millions to population as a percent.

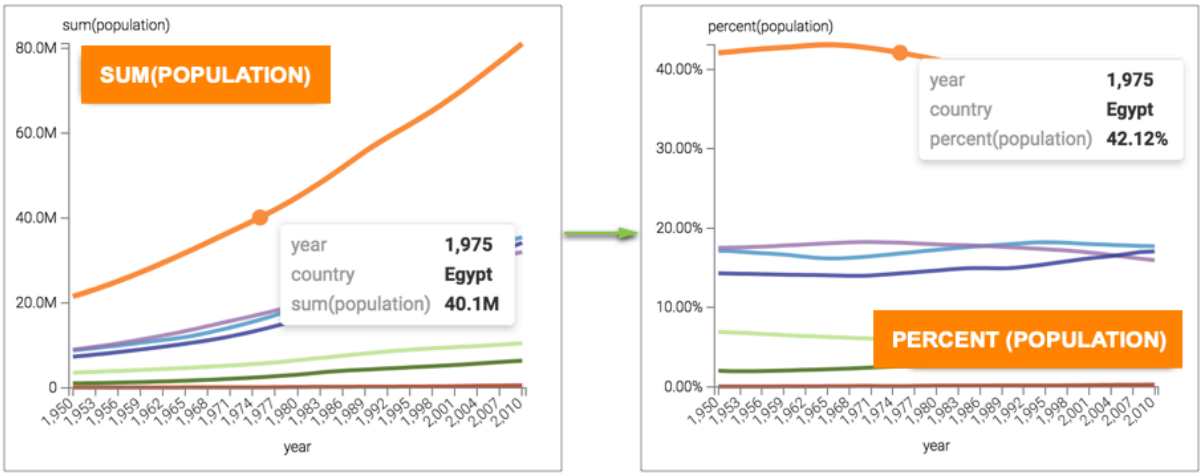

Running total analytic function

This article describes how to use the Running Total analytic function in Cloudera Data Visualization.

Procedure

1. Click the field on the measurement shelf.

   In this example, the sum(population) field on the Y Axis shelf.

2. In the FIELD PROPERTIES menu, expand Analytic Functions, and select Running Total.

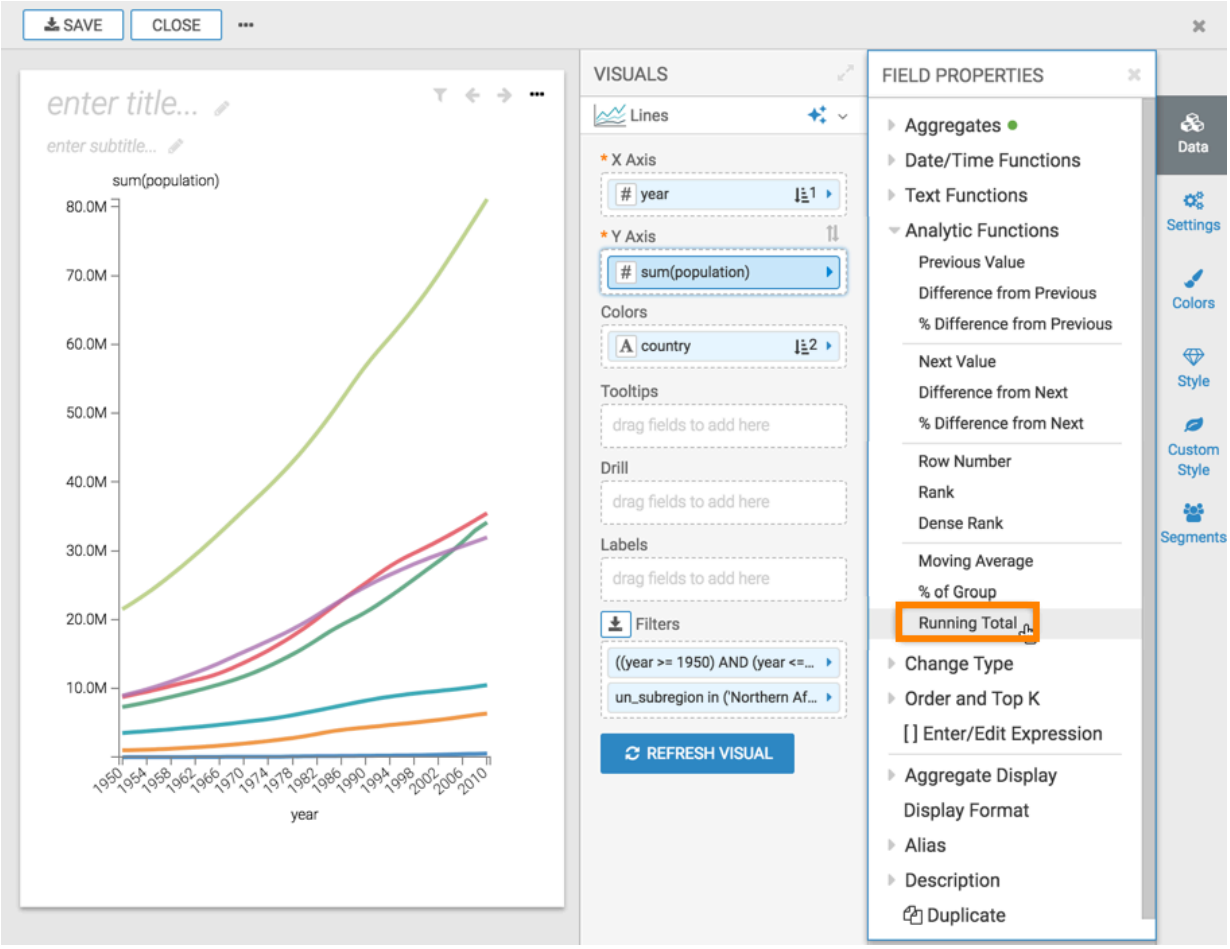

3. In the Analytic Function: Running Total modal window:
   a. In Select aggregate for this measure, leave the default Sum. The other aggregation options are Count, Minimum, Maximum, and Average. In addition, some connection types also support Approximate Distinct Count and Exact Distinct Count.
   b. Under Select entities over which to calculate running totals, select year. This is the grouping that applies when calculating the running total.
   c. Under Select the sort order, you can specify the order of sorting. This order specifies what the row is, and the row where the value is taken. If your visual already specifies sorting order, these values appear in the modal.

      Note: You can add additional sort parameters, by clicking the (Plus) icon.

      In this example, leave the default order: year ascending, followed by country ascending.

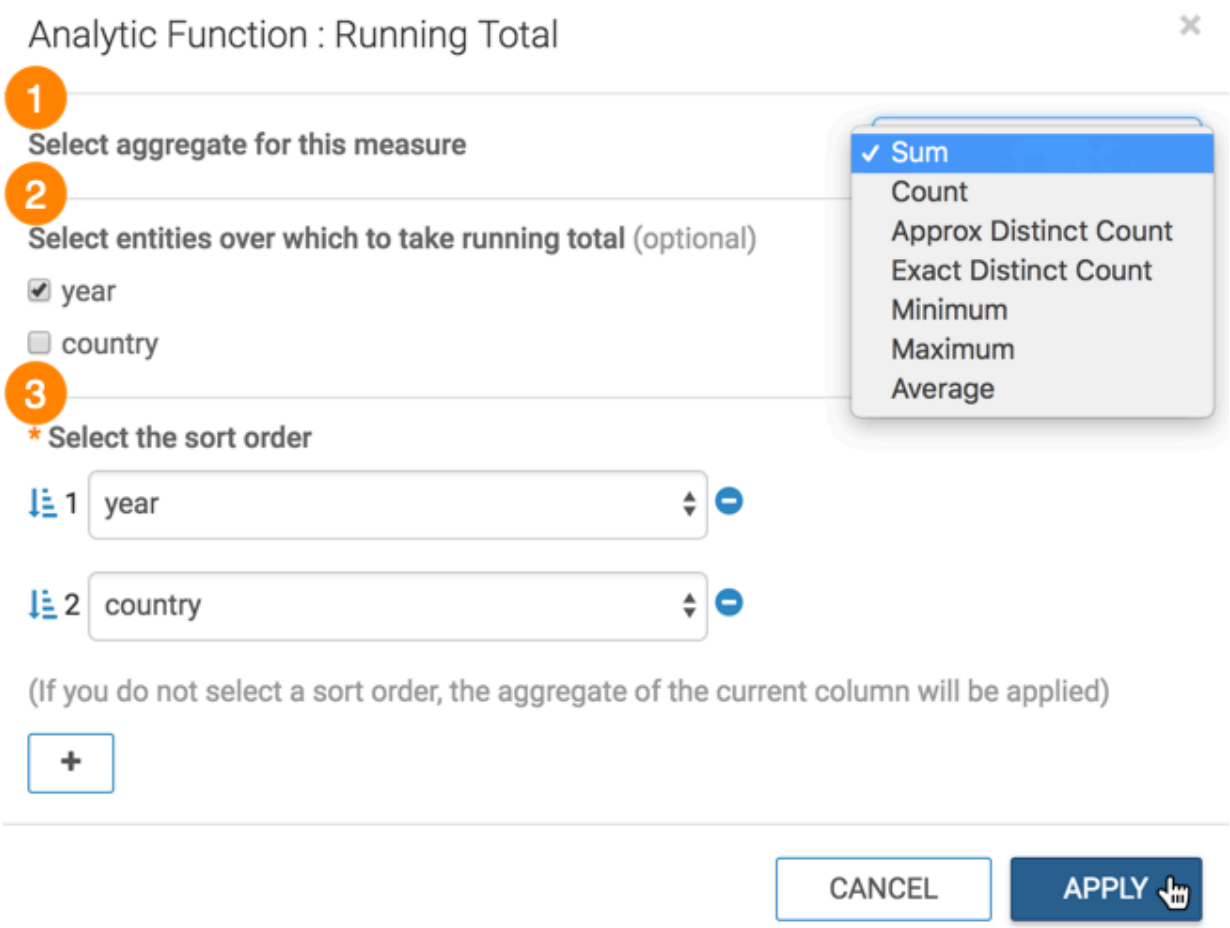

4. Click APPLY.
5. If you examine the expression in the Enter/Edit Expression interface, it shows that the function has the following form:

```
sum(sum([population])) over (partition by [year] order by [year], [count
ry]
            rows between unbounded preceding and current row) as 'running
 total(population)'
```

6. Click REFRESH VISUAL.

You can see the change to the appearance of the visual, and the change in values that appear on the tooltip.

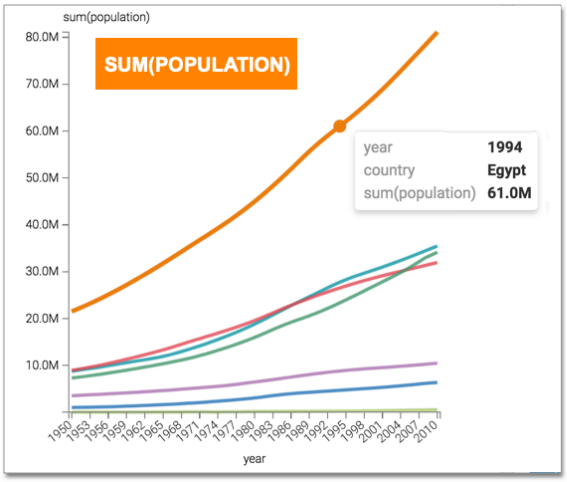

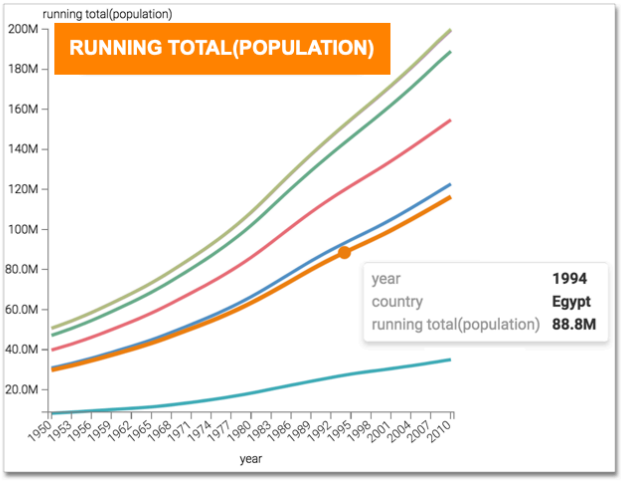

Cloudera Data Visualization 7.2.9

Visuals

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What is a visual? **4**

Visual Builder **4**

What is a visual?

A visual is a single piece of visualized data, for example, a pie chart, a histogram, or a heatmap. It translates large data sets and metrics into a visual representations, making it easier to identify insights about the information represented in the data.

Cloudera Data Visualization has a rich offering of different types of visualization to assist you in analyzing your data.

For a complete list of the available visual types, see the *Visual Type Catalog*.

Note: The following visual types support smooth rendering of streaming data: Lines, Bar, Areas, Stacked Bars, Combo, Calendar Heatmap, Correlation Heatmap, Packed Bubbles, Pie, and Scatter.

Related Information

Creating a visual

Visual Type Catalog

Visual Builder

The Visual Builder of Cloudera Data Visualization is an intuitive user interface for creating data visualizations. You can access the interface within the Dashboard Designer.

The Visual Builder page has the following main parts:

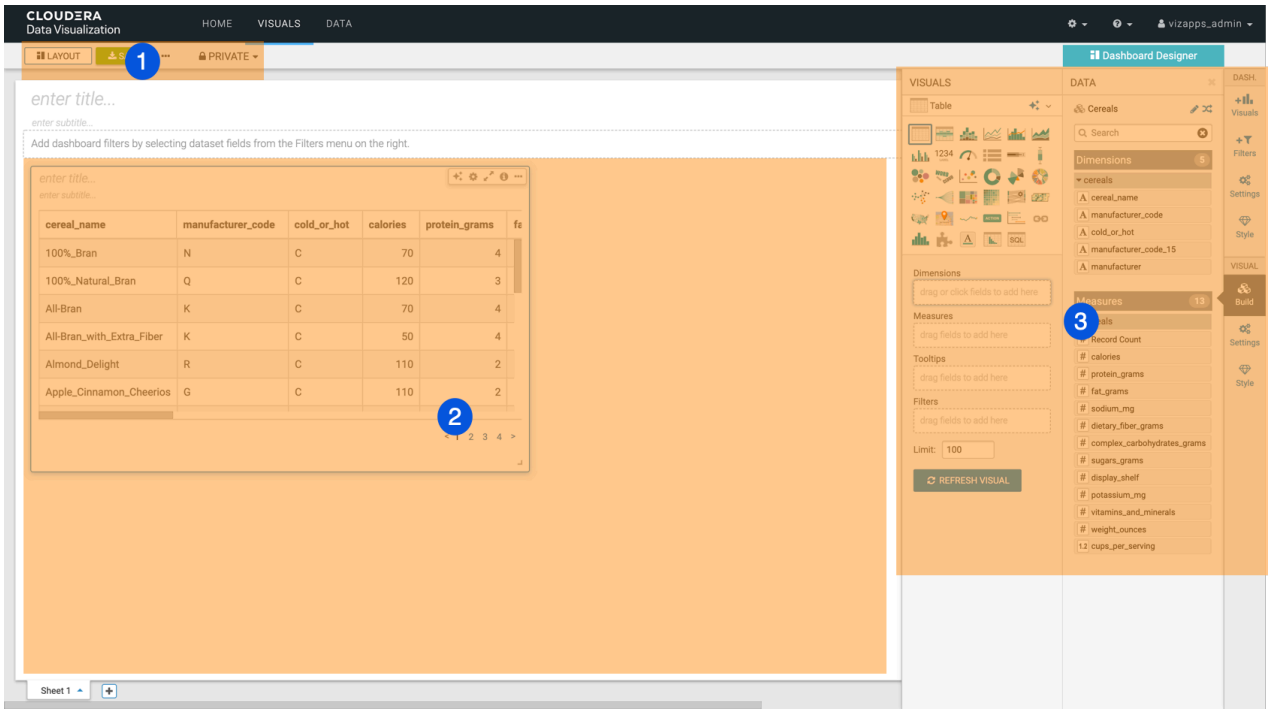

1. Visual Builder top menu bar
2. Visual preview area
3. Visual Builder side menu bar

Visual Builder top menu bar

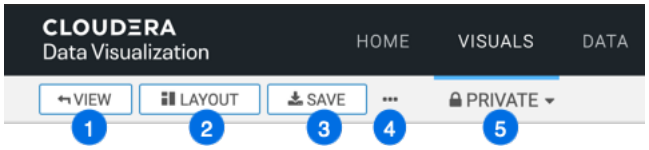

1. The VIEW button allows you to preview the dashboard at runtime. It is only displayed on the menu bar if you already saved your dashboard.
2. The LAYOUT button allows you to edit the layout of dashboard visuals. It is only displayed on the menu bar if you already saved your dashboard.
3. The SAVE button saves all changes to the dashboard.
4. The Supplemental menu provides access to additional functions:

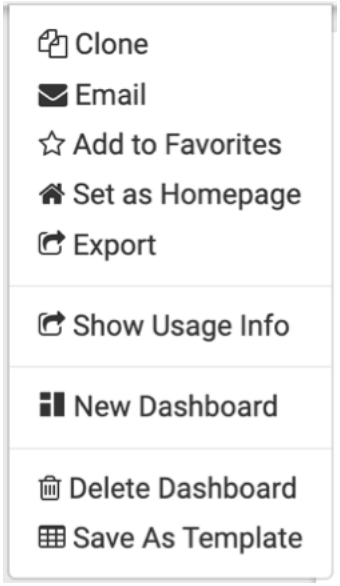

   - Clone lets you clone the current dashboard.
   - Add to Favorites lets you mark commonly-used dashboards as favorites and quickly find them in a subsequent work session. This menu item is only available after you first saved your dashboard.
   - Set as Homepage lets you change your home screen to the current dashboard. This menu item is only available after you saved your dashboard.
   - Export allows you to export all of the dashboard elements. This menu item is only available after you saved your dashboard.
   - Download as... provides you with options to save your dashboards as images (in png format) and as PDF files.
   - Show Usage Info provides you with profile data.
   - New Dashboard lets you create a new dashboard.
   - Delete Dashboard lets you remove the current dashboard.
5. The workspace selector allows you to assign your dashboard to a different workspace. Dashboards can belong to public, private, or a custom workspace, but they can only exist in one workspace at any time.

Visual preview area

The visual preview area shows the actual visual you have added to your dashboard. When you start a new dashboard and add your first visual, the default table type is shown listing all fields in the selected dataset. The preview area also contains the title and subtitle of the visual.

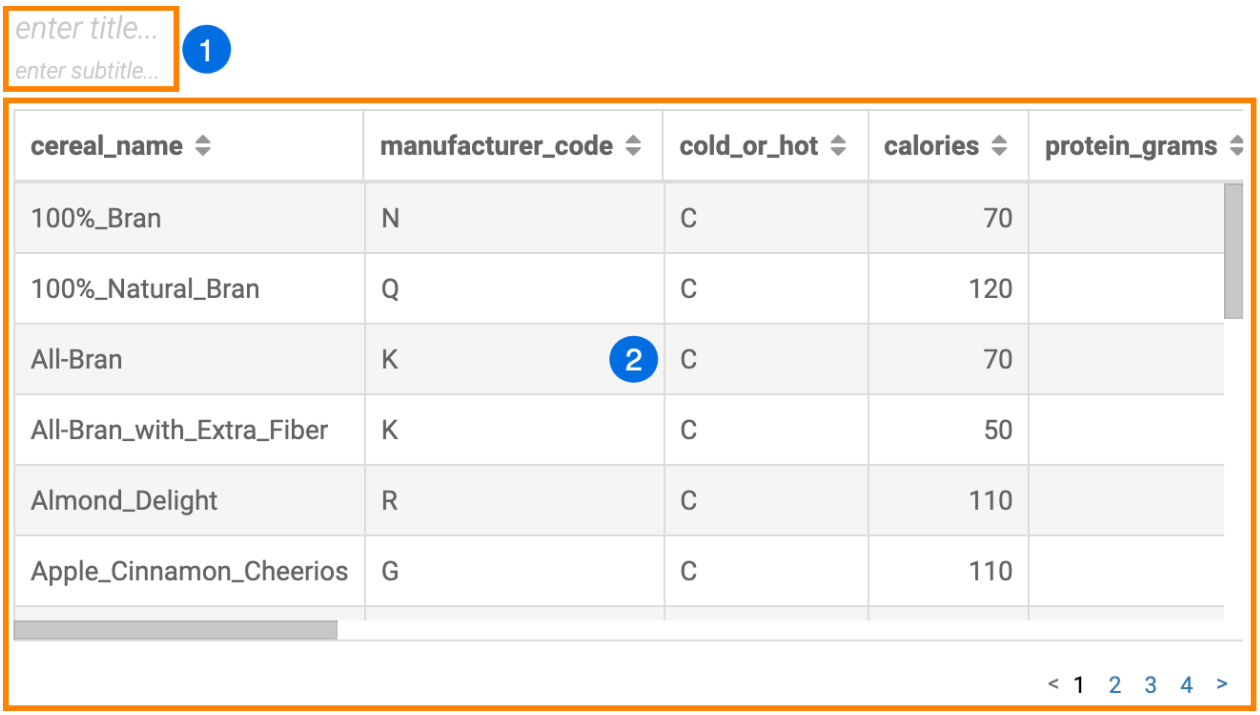

1. The title area contains two editable fields where you can add the title and subtitle of your choice. Both are empty when you create a new dashboard.
   - enter title...
   - enter subtitle...
2. The default table visual is based on the dataset selected.

Visual Builder side menu bar

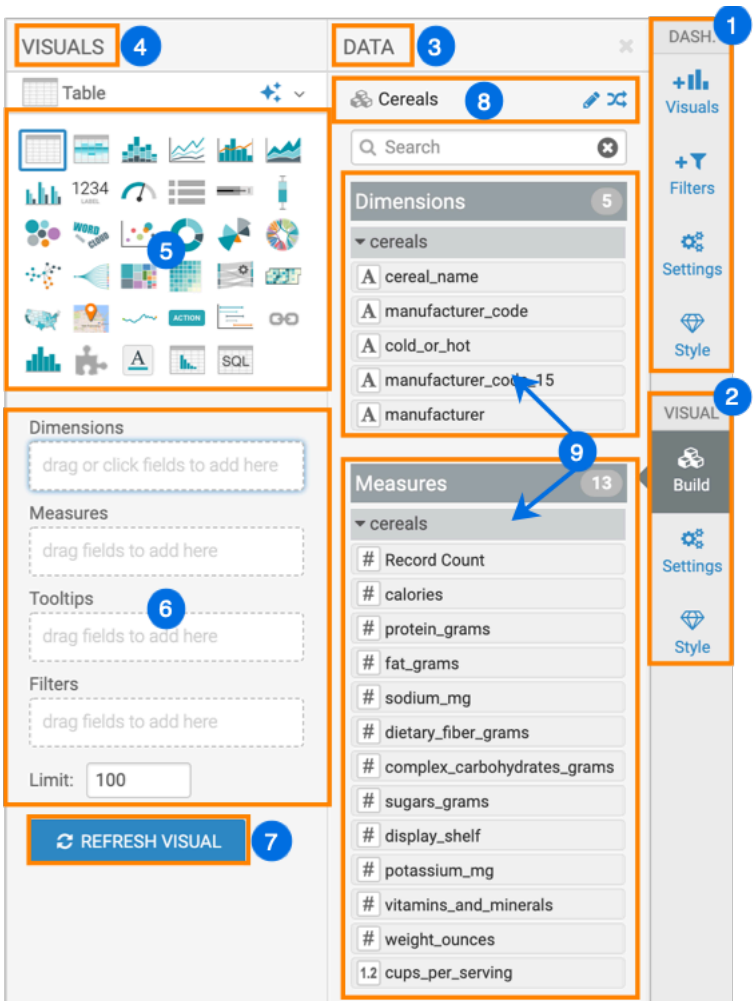

1. The dashboard menu selector enables you to choose between different menus within the Dashboard Designer.
   - Visuals
   - Filters
   - Settings
   - Style

   For more information about the dashboard menu options, see the *Dashboard Designer*.
2. The visual menu selector enables you to choose between different menus within the Visual Builder. You can minimize this side menu to see the full dashboard preview area by clicking the menu icon again.
   - Build
   - Settings
   - Style
3. The DATA menu provides you access to the selected dataset and its fields.
4. The VISUALS menu provides you with options to define what data is appears in your visual and how it is displayed.
5. The Visual Type Catalog menu shows the standard visual types. You can switch between visual types by selecting a different visual type in this menu. For more information, see the *Visual Type Catalog*.

6. Shelves of the visual specify the fields and segments that provide the data to the visual. Different visual types require specific information and have somewhat different shelves.

   The table visual has the following shelves:

   - Dimensions
   - Measures
   - Filters
   - Tooltips
   - Limit

   The shelves of the bars visual are:

   - X Axis,
   - Y Axis
   - Colors
   - Tooltips
   - Drill
   - Labels
   - Filters

   You can populate the shelves by dragging Dimensions and Measures fields from the DATA menu. When a shelf is active (selected), you can add multiple fields simply by clicking each of them in the DATA menu.
7. If you click the REFRESH VISUAL button after changing the visual type or the contents of the shelves, the visual is re-drawn.
8. The name of the Dataset is shown at the top of the DATA menu. You can click the [arrows] icon to change the dataset of the visual, or click the [pencil] icon to open the dataset definition in a new browser tab and make any necessary changes that are immediately reflected in the options for the visual.
9. Dimensions and Measures list all fields of the selected dataset. You can use these to populate the shelves of the visuals. If the dataset models a join of two or more tables, the fields are grouped with their source data table.

   Note: You can also use the Search text box to look for a specific dataset field.

Related Information

Visual Type Catalog

Dashboard Designer

Cloudera Data Visualization 7.2.9

Working with Legends on Visuals

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Highlighting elements by hovering over legend..... 4

Filtering on an element..... 4

Highlighting elements by hovering over legend

A visual may become very crowded, making it hard to understand what graphical elements represent. Cloudera Data Visualization makes it easy to find a particular series by making legends not only explanatory, but also interactive.

About this task

You can find details of a particular element in the legend by hovering over it.

Procedure

1. Open any visual inside a dashboard.
2. Make sure that the visual displays the legend.
   For more information, see *Adding and removing legend*.
3. Hover on one of the options in the legend to highlight only the data for that species.

   This example shows versicolor.

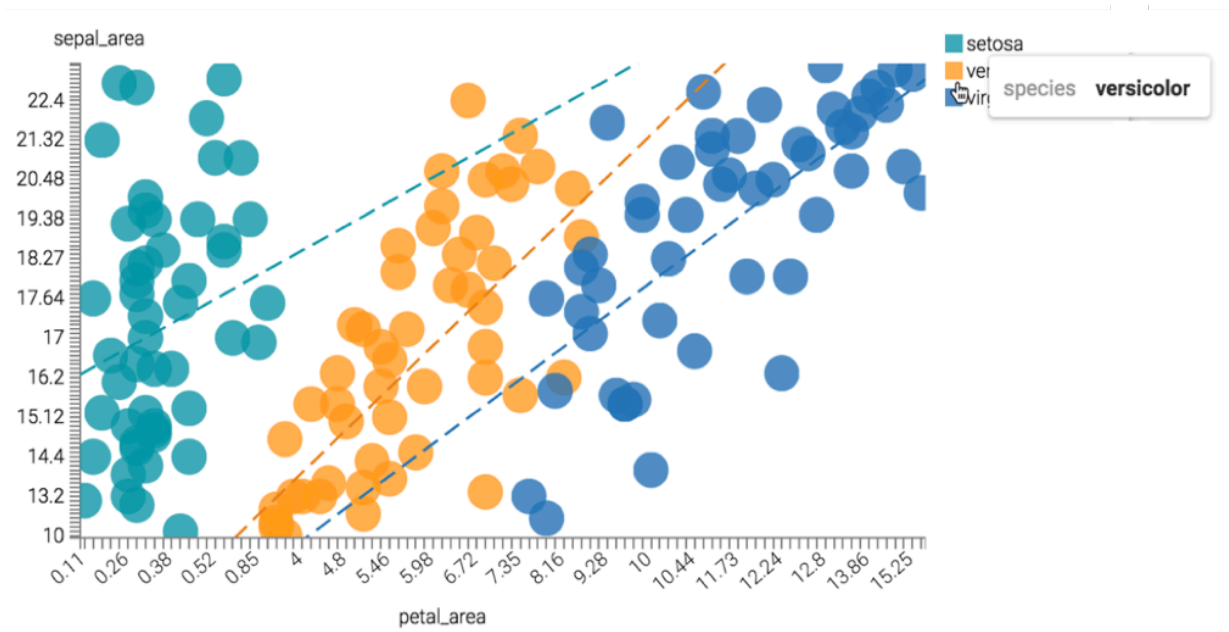

This graph is also interactive in Edit mode, inside a dashboard.

Related Information

Change legend style

Filtering on an element

About this task

A line chart display can become very crowded, making it difficult to discern which line models a particular series. For example, the World Life Expectancy dataset models data for over 200 countries. In View mode, you can click on a particular element of a legend, and filter data for just that element.

Procedure

1. Find the World Population - Lines visual.
2. Make sure that the visual displays the legend.

   If the legend is not displayed, follow the steps outlined in *Adding and removing legend* to add it.
3. Switch to Dashboard View mode.
4. Click one of the options in the legend for the visual, to see only the data for that country.

   In this example, Albania is shown.

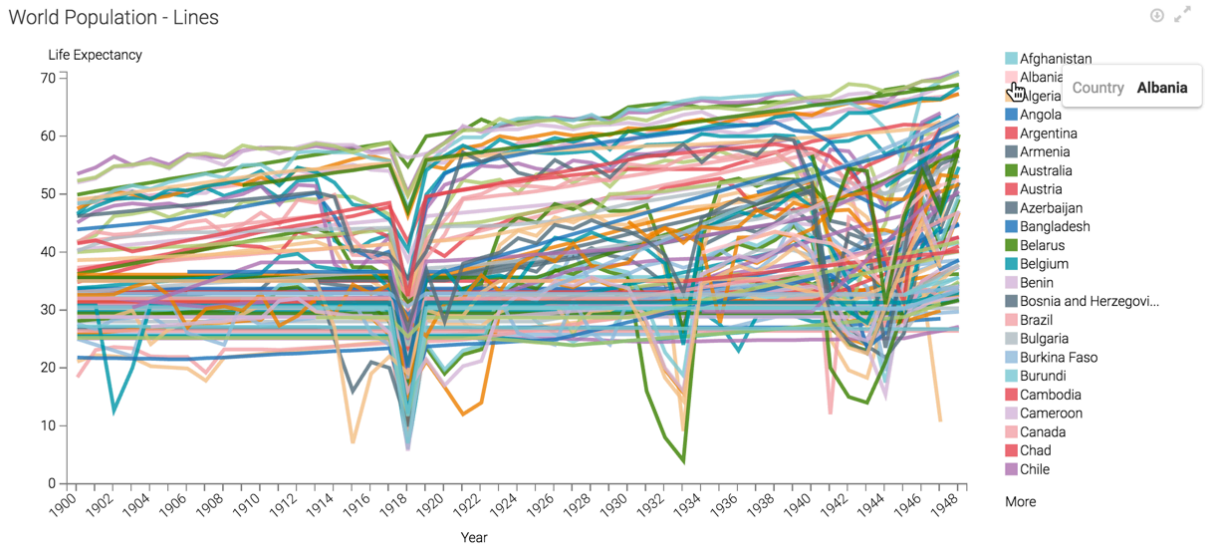

The new chart displays the line for Albania only. Additionally, the horizontal axis adjusts for the dimension range. Albania reports data starting in year 1934.

Cloudera Data Visualization lists all applicable filters at the top of the vertical axis. Hovering over them shows the values of the filters.

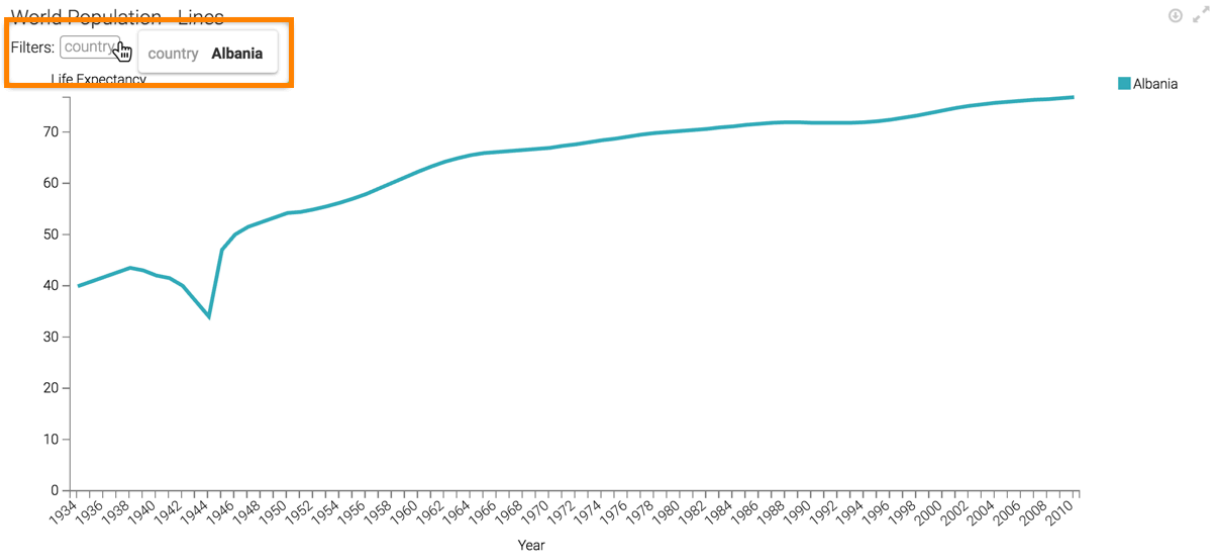

Related Information

Change legend style

Cloudera Data Visualization 7.2.9

Working with User Roles

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating new roles 4

Adding privileges 6

Setting system privileges 8

Setting role privileges 10

Setting connection privileges 13

Setting dataset privileges 16

Editing role assignments 19

Assigning roles to users 21

Assigning roles to user groups 23

Assigning multiple roles to multiple users 26

Deleting roles 29

Creating new roles

About this task

Administrators with Manage roles and users privilege can introduce new roles into the Role-Based Access Control (RBAC) system.

Follow these steps demonstrate to create a new role.

Procedure

1. On the main navigation bar, click the (gear) icon.
2. In the drop-down menu, click Manage Roles.

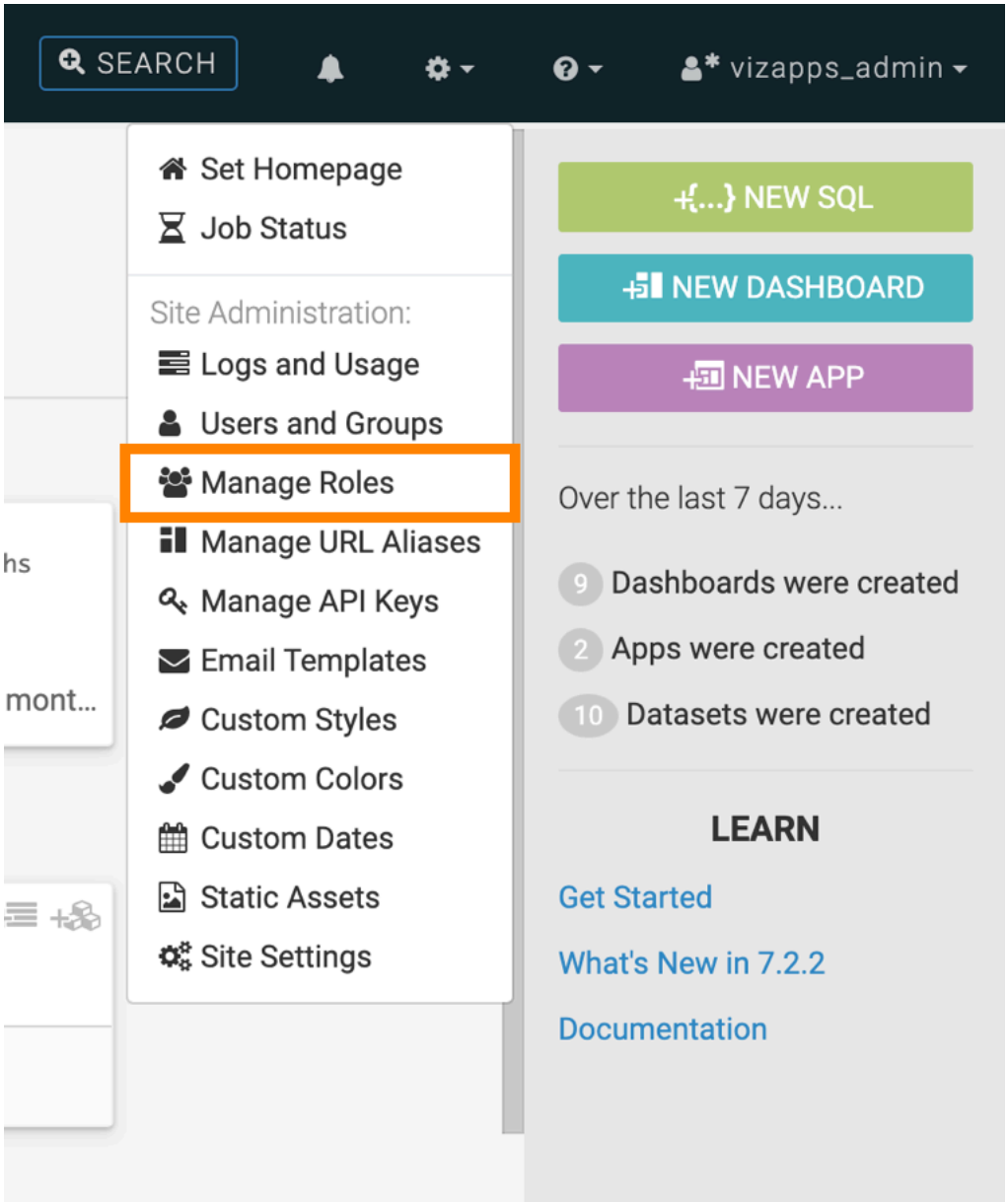

3. Click NEW ROLE.

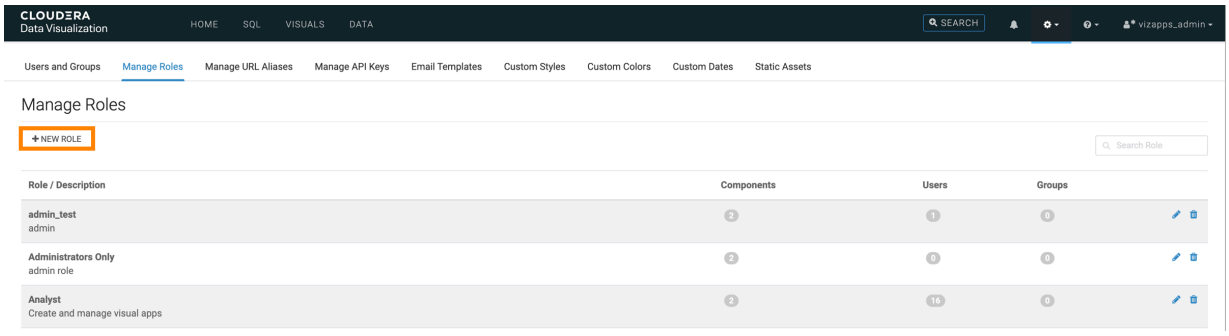

The Role Detail interface appears, open on the Privileges tab.

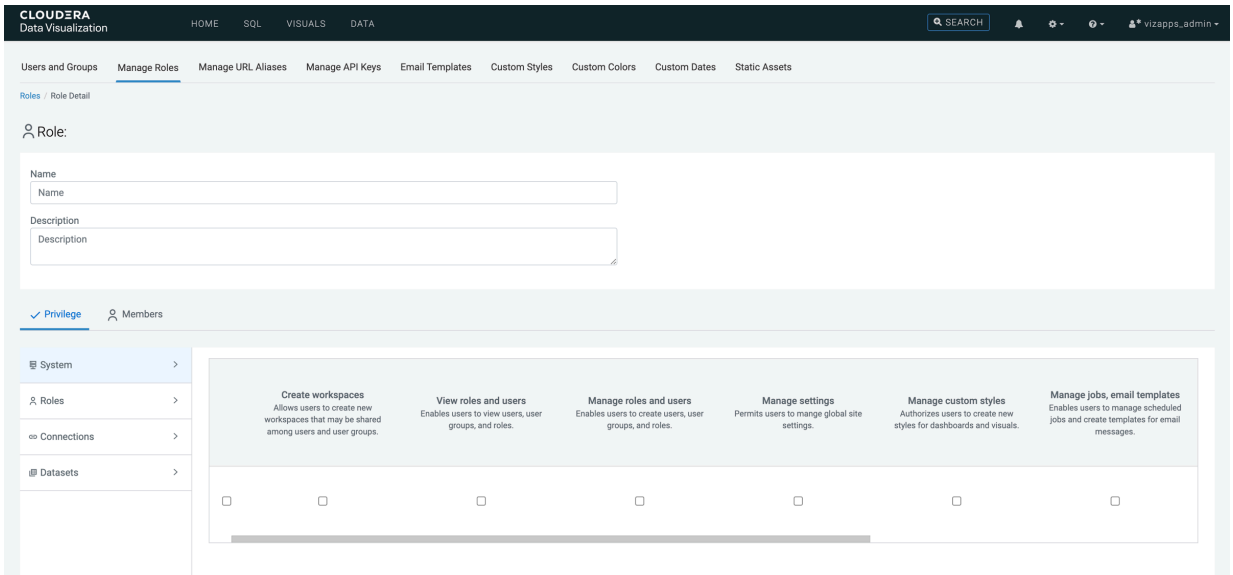

4. Define role details.
   a) Provide a distinctive name for the new role.
   b) In the Description field, briefly explain the purpose or context of the new role.
   c) Click APPLY CHANGES to save the role.
5. To ensure the role was successfully created, click Roles in the top left corner of the page.

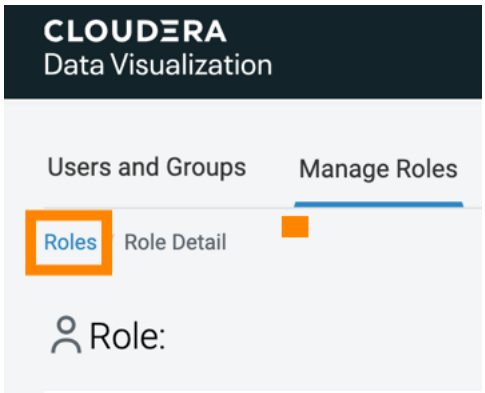

The newly created role should now be visible in the role list.

What to do next

Proceed by defining *role privileges* and *editing role assignments* as needed for the newly created role.

Related Information

Editing role assignments
Role privileges

Adding privileges

Once you have defined a new role in the Role-Based Access Control (RBAC) system of Cloudera Data Visualization, you can start adding privileges to that role.

About this task

The following steps show you how to add or modify privileges associated with a specific role.

Procedure

1. On the main navigation bar, click the (gear) icon.

2. In the drop-down menu, click Manage Roles.

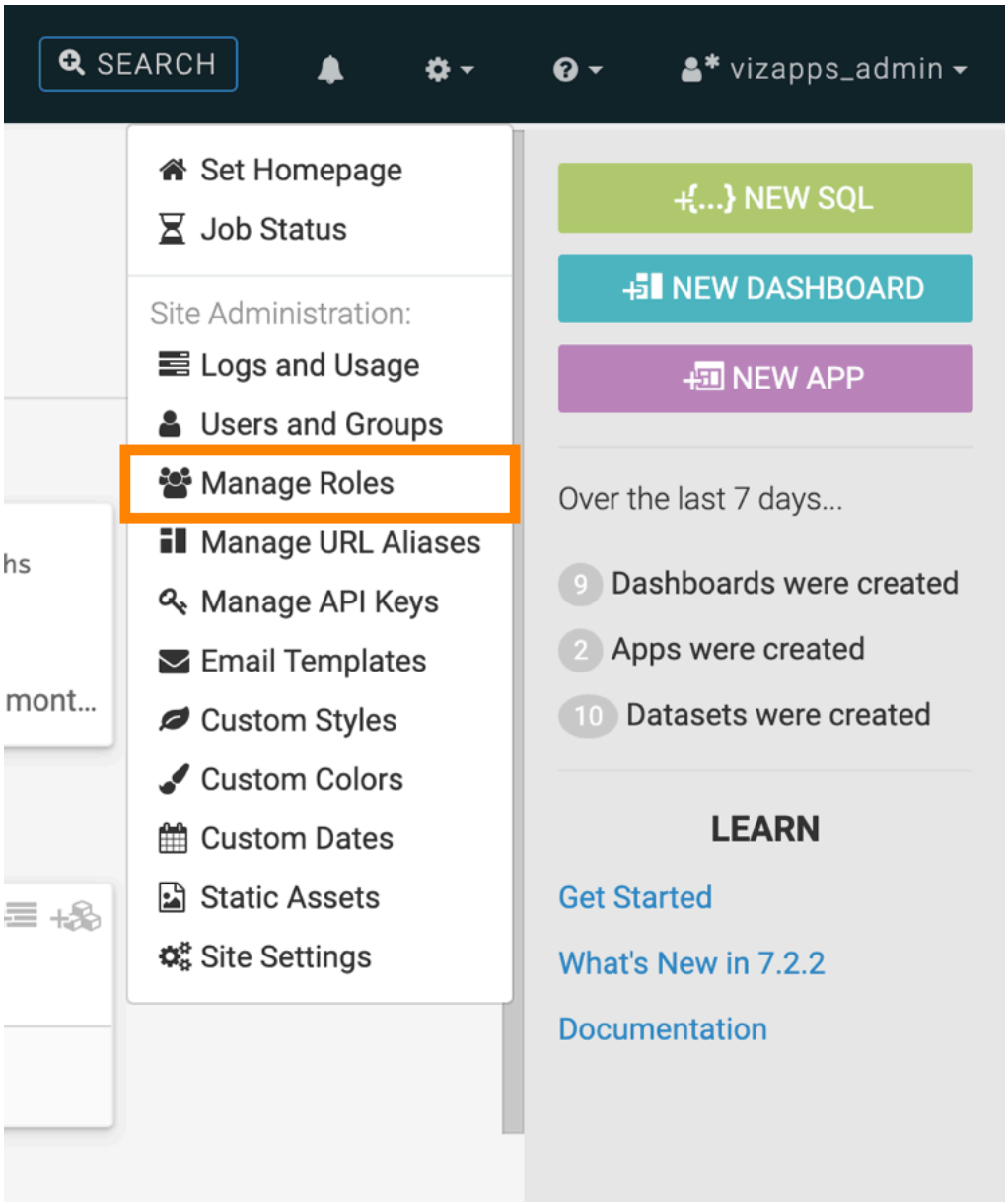

3. In the Manage Roles interface, locate the role you wish to edit and click on the (edit) icon next to the respective role.

   You can use the Search Role box to find the role you want to edit.

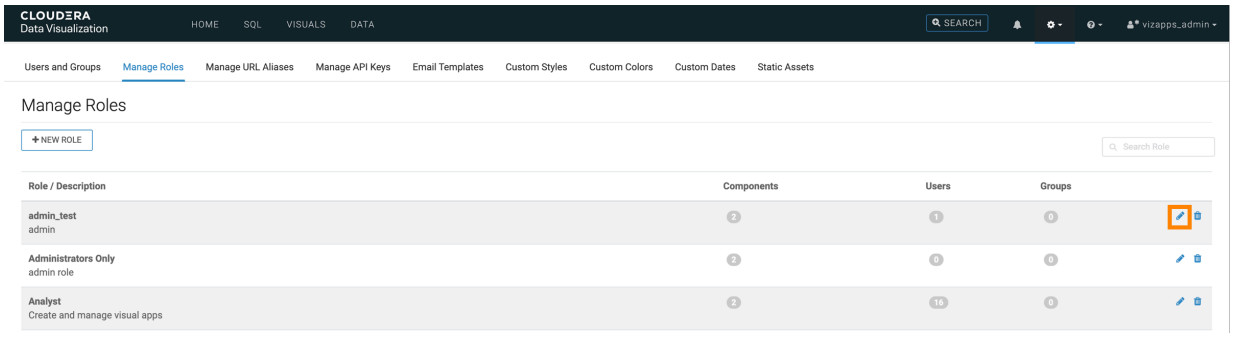

4. On the Role Detail interface, you can set the following groups of privileges:
   - *System privileges*
   - *Role privileges*
   - *Connection privileges*
   - *Dataset privileges*

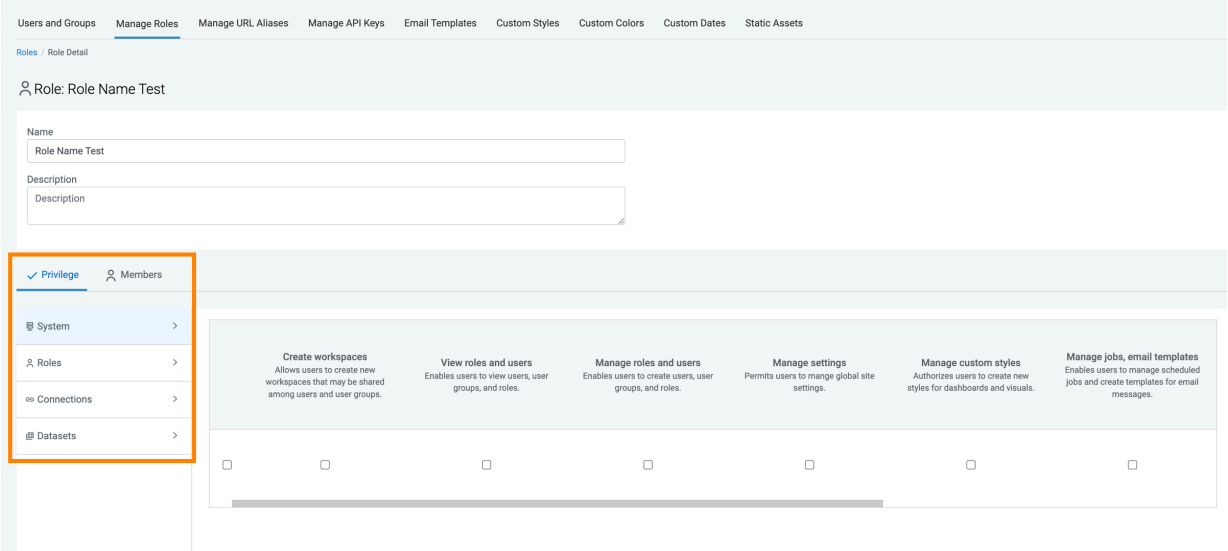

5. After configuring the desired privileges for the role, click APPLY CHANGES to save the new privilege assignments.

Related Information

Creating new roles
Setting system privileges
Setting role privileges
Setting connection privileges
Setting dataset privileges

Setting system privileges

System-level privileges are key components of the Role-Based Access Control (RBAC) system in Cloudera Data Visualization.

About this task

To configure system-level privileges for a role, follow the steps outlined below.

Procedure

1. On the main navigation bar, click the (gear) icon.

2. In the drop-down menu, click Manage Roles.

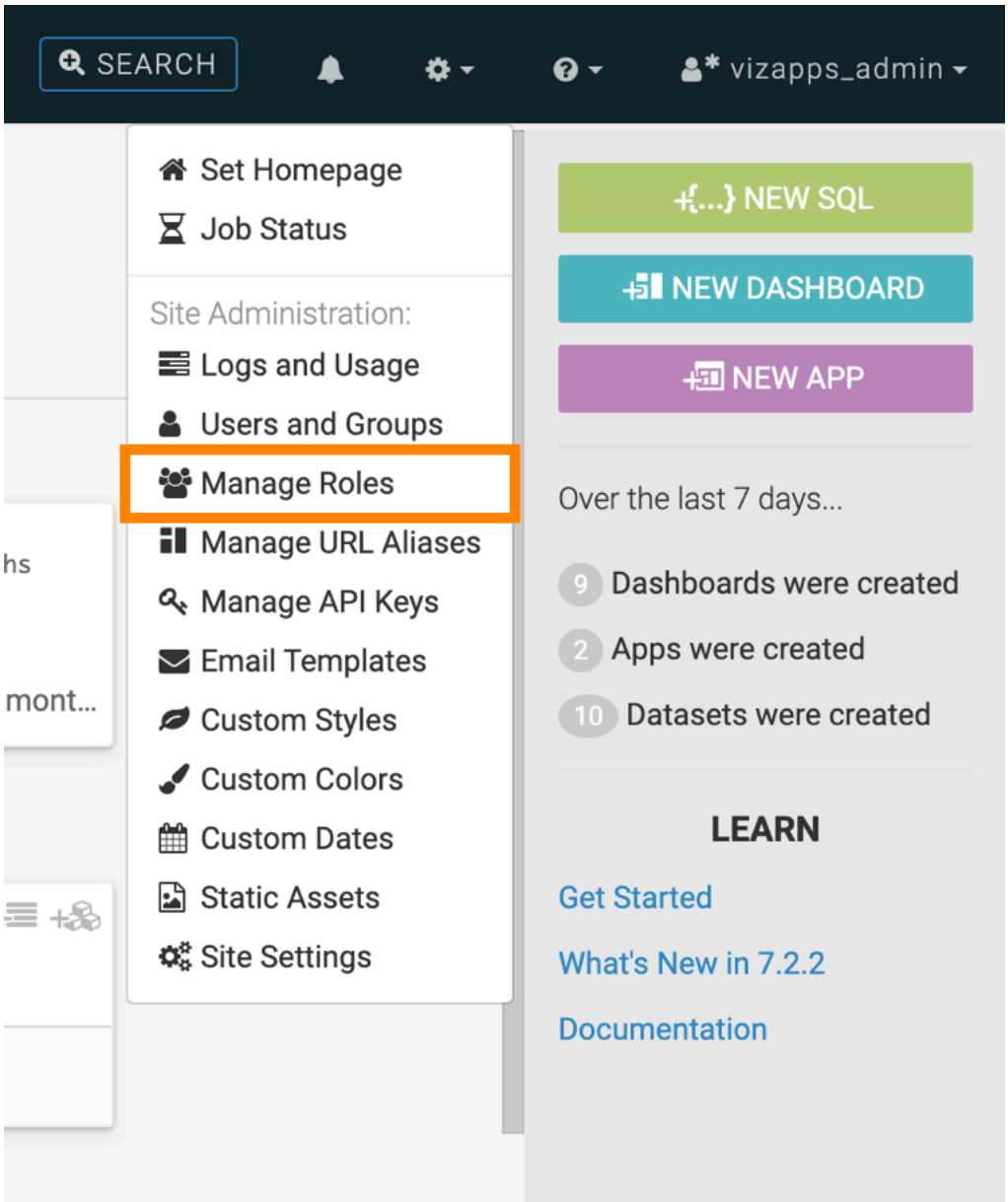

3. In the Manage Roles interface, find the role you wish to edit and click on the (edit) icon next to the respective role. You can use the Search Role box to find the role you want to edit.

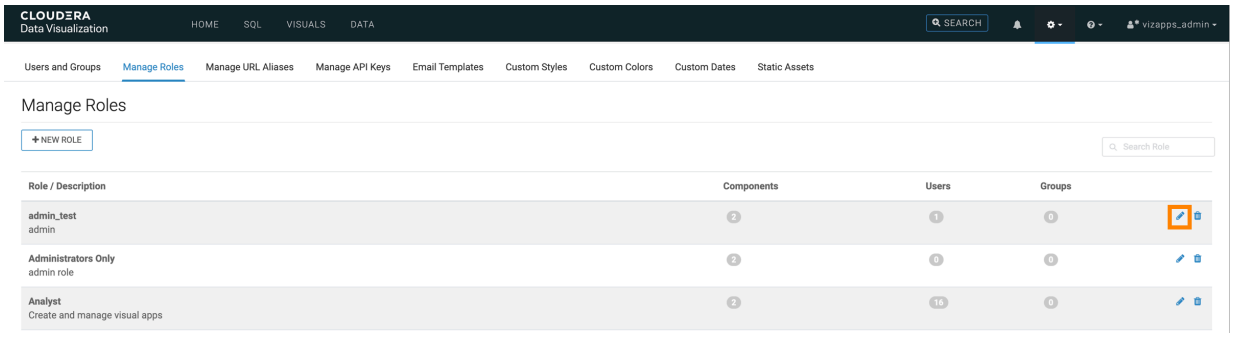

The Role Detail interface is displayed.

4. Locate the System category on the Privilege tab and select the checkboxes for the actions you want to permit for the role.

   Click the Select all checkbox, next to the System category name if you want to activate all permissions.

   - Create workspaces
   - View roles and users
   - Manage roles and users
   - Manage settings
   - Manage custom styles
   - Manage jobs, email templates
   - View activity logs
   - Manage data connections
   - Additional system privilege

   Note:

   Privilege dependency information:

   The View roles and users permission is mandatory if the privilege includes the Manage roles and users permission and cannot be removed. The checkboxes for these permissions are pre-filled and fixed. If Manage roles and users is deselected, the View roles and users permission becomes optional and can also be deselected.

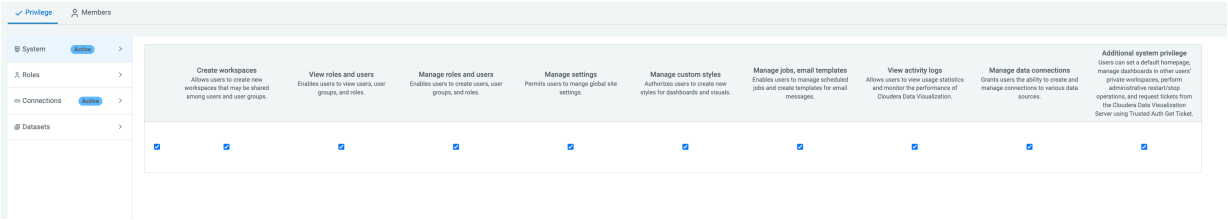

5. Click APPLY CHANGES to save the changes made to the role.

What to do next

Once system privileges are configured, proceed to *setting role privileges*.

For more information on available permissions, see *RBAC permissions*.

Related Information

Setting role privileges

RBAC permissions

Setting role privileges

Role privileges are integral components of the Role-Based Access Control (RBAC) system in Cloudera Data Visualization. They define the level of dataset access that a role is allowed to grant to the members of a specified role.

About this task

To add new role privileges to a role, follow the steps outlined below.

Procedure

1. On the main navigation bar, click the (gear) icon.

2. In the drop-down menu, click Manage Roles.

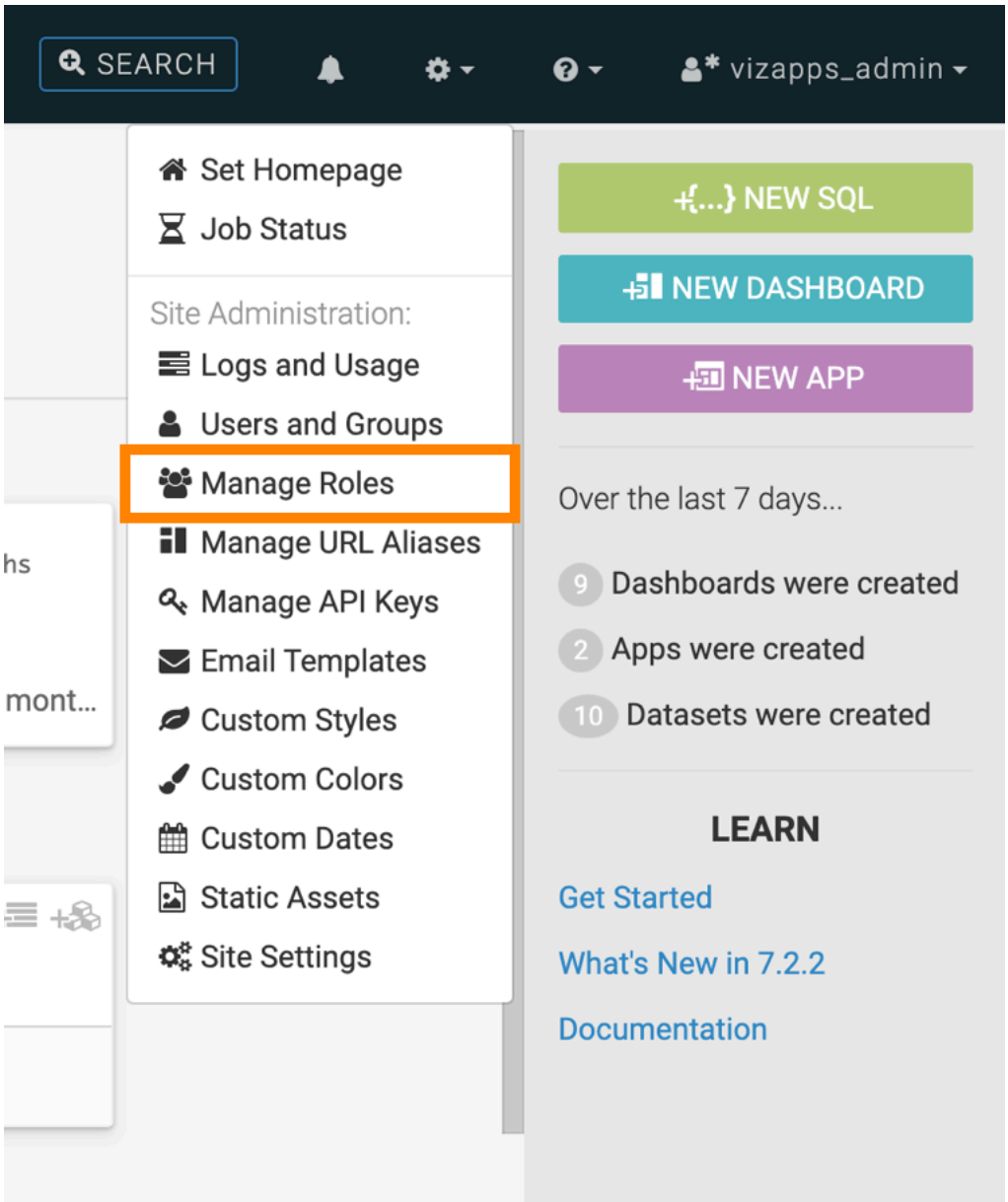

3. In the Manage Roles interface, find the role you wish to edit and click the (edit) icon next to the respective role. You can use the Search Role box to find the role you want to edit.

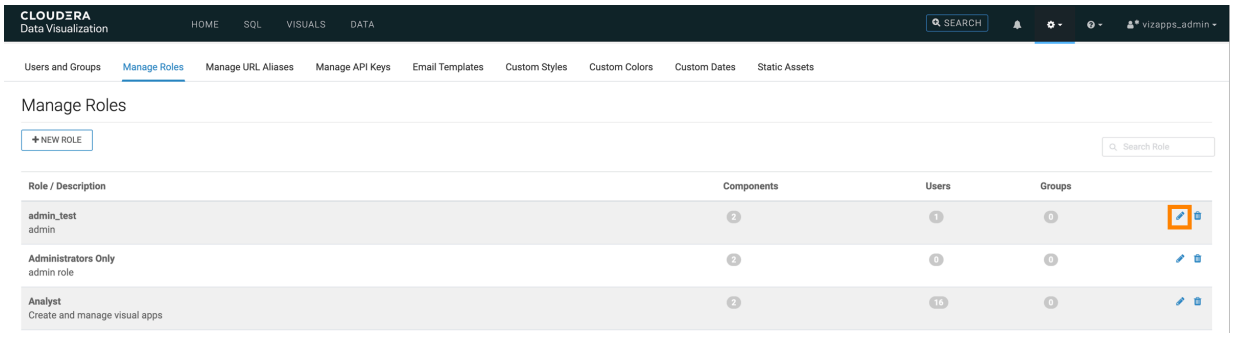

The Role Detail interface is displayed.

4. Switch to the Roles category on the Privilege tab.

5. Add a new role by clicking ADD ROLES.

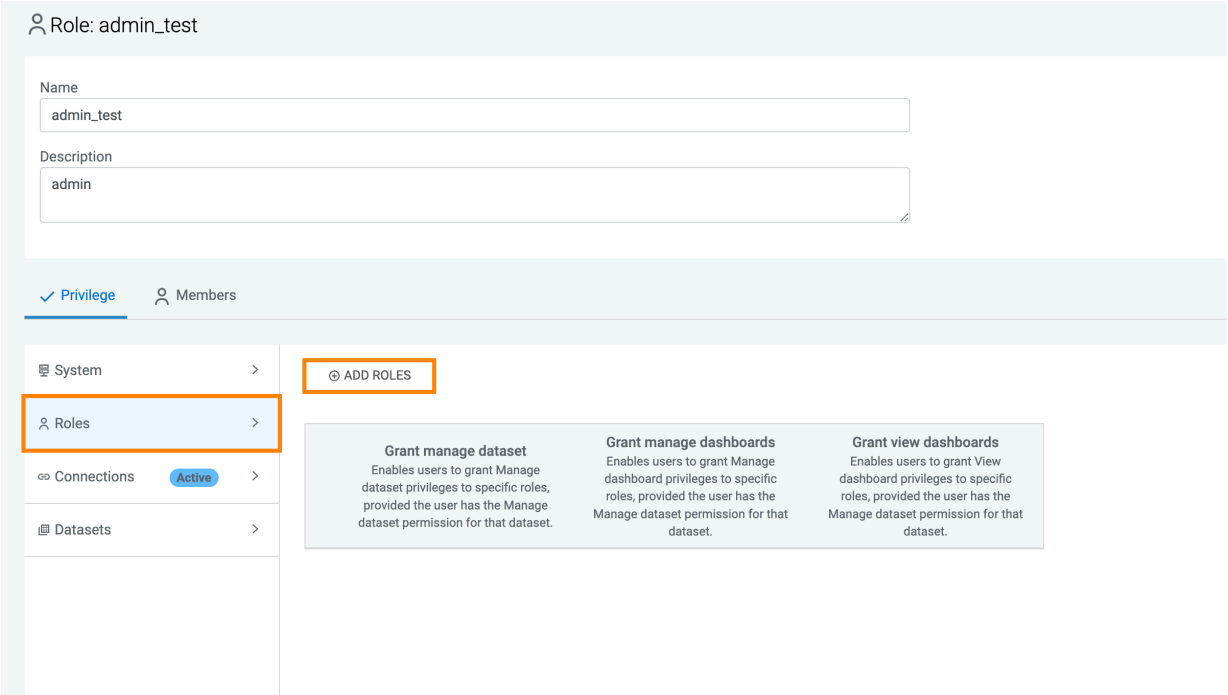

The Add Role Privilege modal window appears.

6. Select the required role from the Roles drop-down list and add it.

   When the role is added, by default, all possible privileges are marked for it.

   You can repeat this step to add more roles to the list, if needed.

   You can remove a role from the list by clicking the Delete icon.

7. Select the checkboxes for the actions that you want to permit for the role(s) you have added.

   - Grant manage dataset
   - Grant manage dashboards
   - Grant view dashboards

   Note:

   Privilege dependency information:

   The Grant manage dashboards and Grant view dashboards permissions are mandatory if the privilege includes the Grant manage dataset permission and cannot be removed. The checkboxes for these permissions are pre-filled and fixed.

   If Grant manage dataset is deselected, Grant manage dashboards becomes optional and can also be deselected. If Grant manage dashboards is deselected, Grant view dashboards becomes optional and can also be deselected.

8. After adding roles and selecting the appropriate role privileges, click APPLY CHANGES.

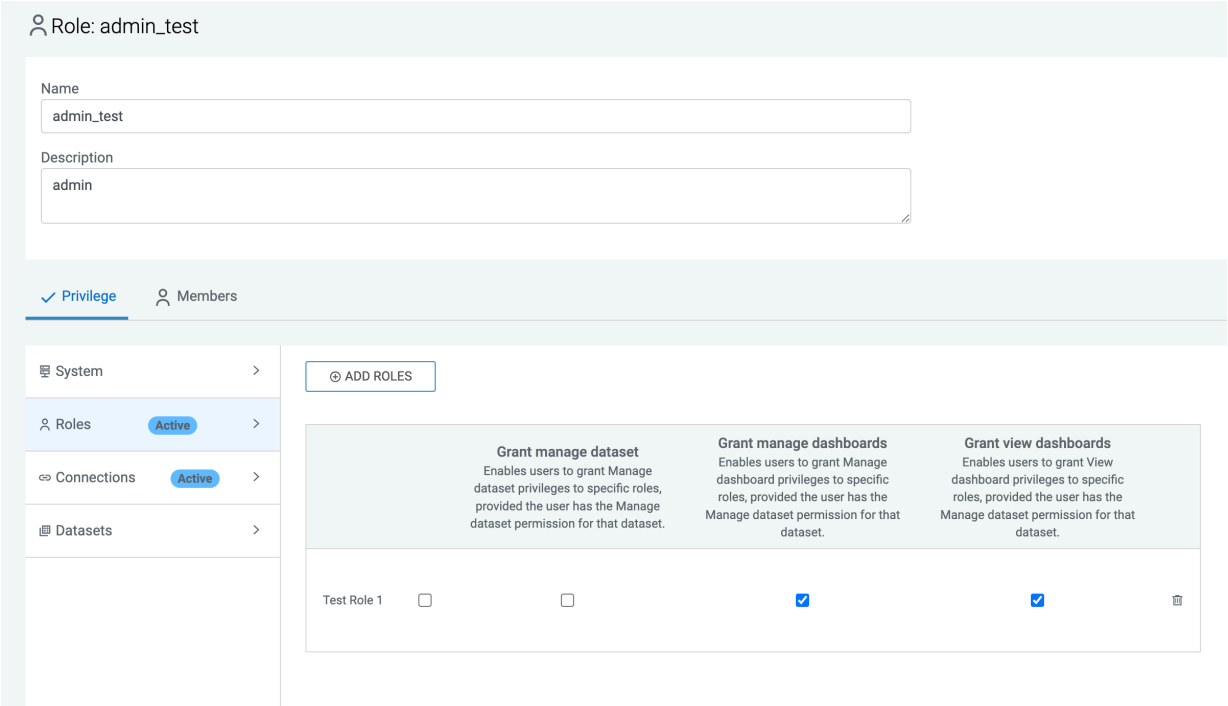

Results

- The members assigned to the selected role can now grant dataset access based on the role-based privilege rows.
- The selections in the rows indicate the level of privileges each role receives. For example, Test Role 1 can grant Manage and View privileges to users.
- The dataset access permissions are granted to the roles defined on the component line.

What to do next

Proceed to *Setting connection privileges*.

For more information on possible permissions, see *RBAC permissions*.

Setting connection privileges

Connection privileges are integral components of the Role-Based Access Control (RBAC) system in Cloudera Data Visualizationn. These privileges can be configured uniformly across multiple connections or customized for each connection, based on specific business requirements.

About this task

Follow the below steps to add connection-level privileges to a role, using "Test Role 1" as an example.

Procedure

1. On the main navigation bar, click the (gear) icon.

2. In the drop-down menu, click Manage Roles.

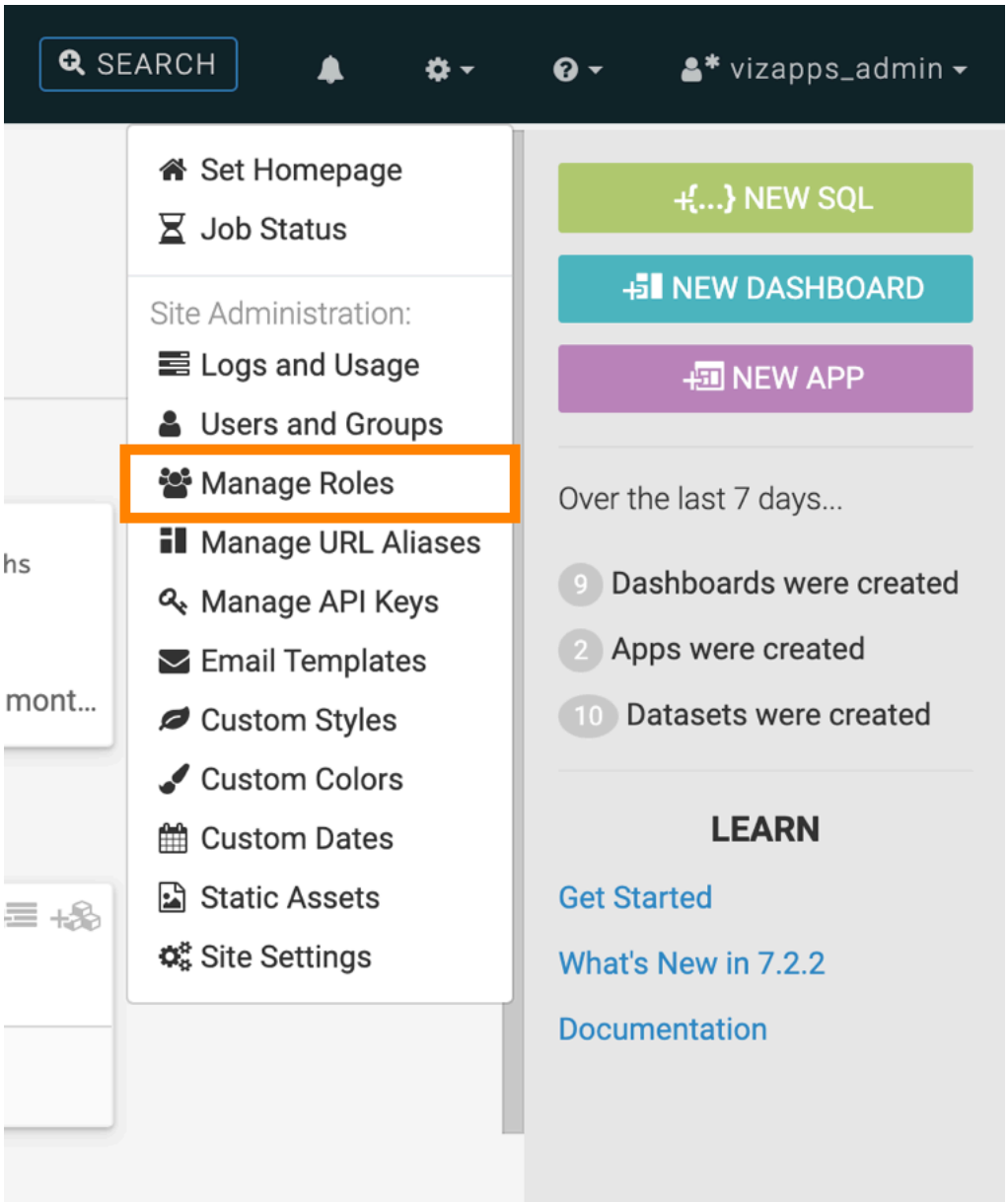

3. In the Manage Roles interface, find the role you wish to edit and click the (edit) icon next to the respective role. You can use the Search Role box to find the role you want to edit.

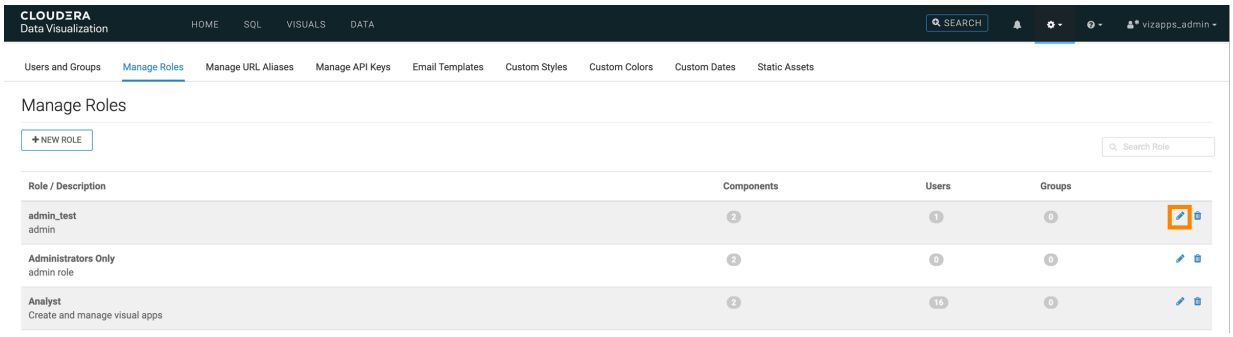

The Role Detail interface is displayed.

4. Switch to the Connections category on the Privilege tab.

5. Add a new connection by clicking ADD CONNECTIONS.

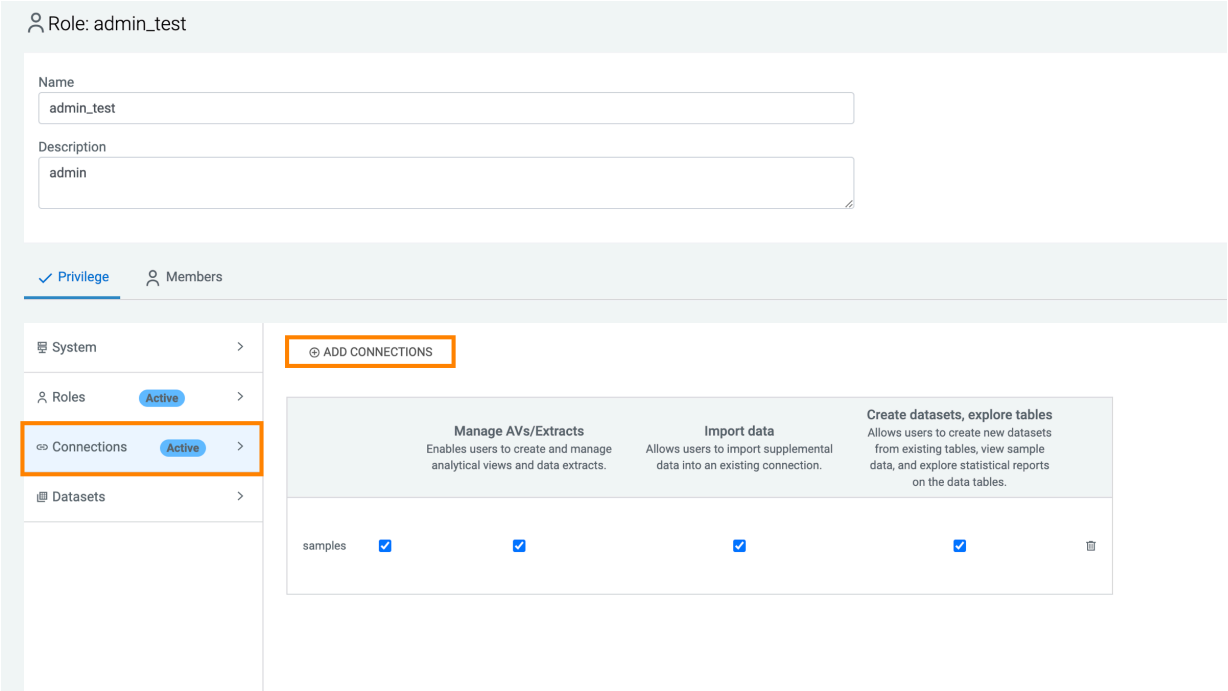

The Add Data connection Privilege modal window appears.

6. From the Connection(s) drop-down menu, select either All connections or one of the individual connections.

   When the connection is added, by default, all possible privileges are marked for it.

   You can repeat this step to include additional connections if necessary.

   You can remove a connection from the list by clicking the Delete icon.

7. Select the checkboxes for the actions that you want to permit for the connection(s) you have added.

   - Manage AVs/Extracts
   - Import data
   - Create datasets, explore tables

   Note:

   Privilege dependency information:

   Manage AVs/Extracts and Import data are independent privileges, both including the Create datasets, explore tables permission. The Create datasets, explore tables is mandatory if the privilege contains the Manage AVs/Extracts or Import data Connections permission and cannot be removed. The checkboxes for these are pre-filled in and fixed.

   If Manage AVs/Extracts is deselected, Create datasets, explore tables becomes optional and can also be deselected. Similarly, if Import data is deselected, it removes any connection-level access components, Create datasets, explore tables becomes optional and can also be deselected.

8. After adding roles and selecting the appropriate role privileges, click APPLY CHANGES.

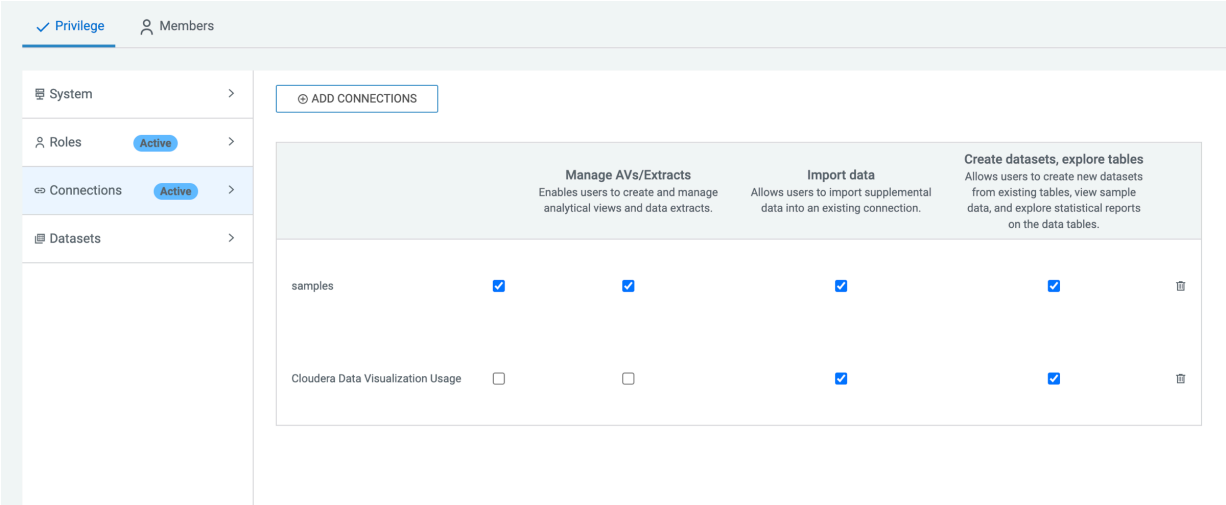

What to do next

Proceed to *Setting dataset privileges*.

For more information on possible permissions, see *RBAC permissions*.

Related Information

Setting dataset privileges
RBAC permissions

Setting dataset privileges

Dataset privileges are integral components of the Role-Based Access Control (RBAC) system in Cloudera Data Visualization. These privileges enable fine-grained control over access to datasets, catering to specific business needs. They can be applied uniformly across multiple datasets or customized individually for each dataset.

About this task

Follow these steps to add dataset-level privileges to a role using "Test Role 1" as an example.

Procedure

1. On the main navigation bar, click the (gear) icon.

2. In the drop-down menu, click Manage Roles.

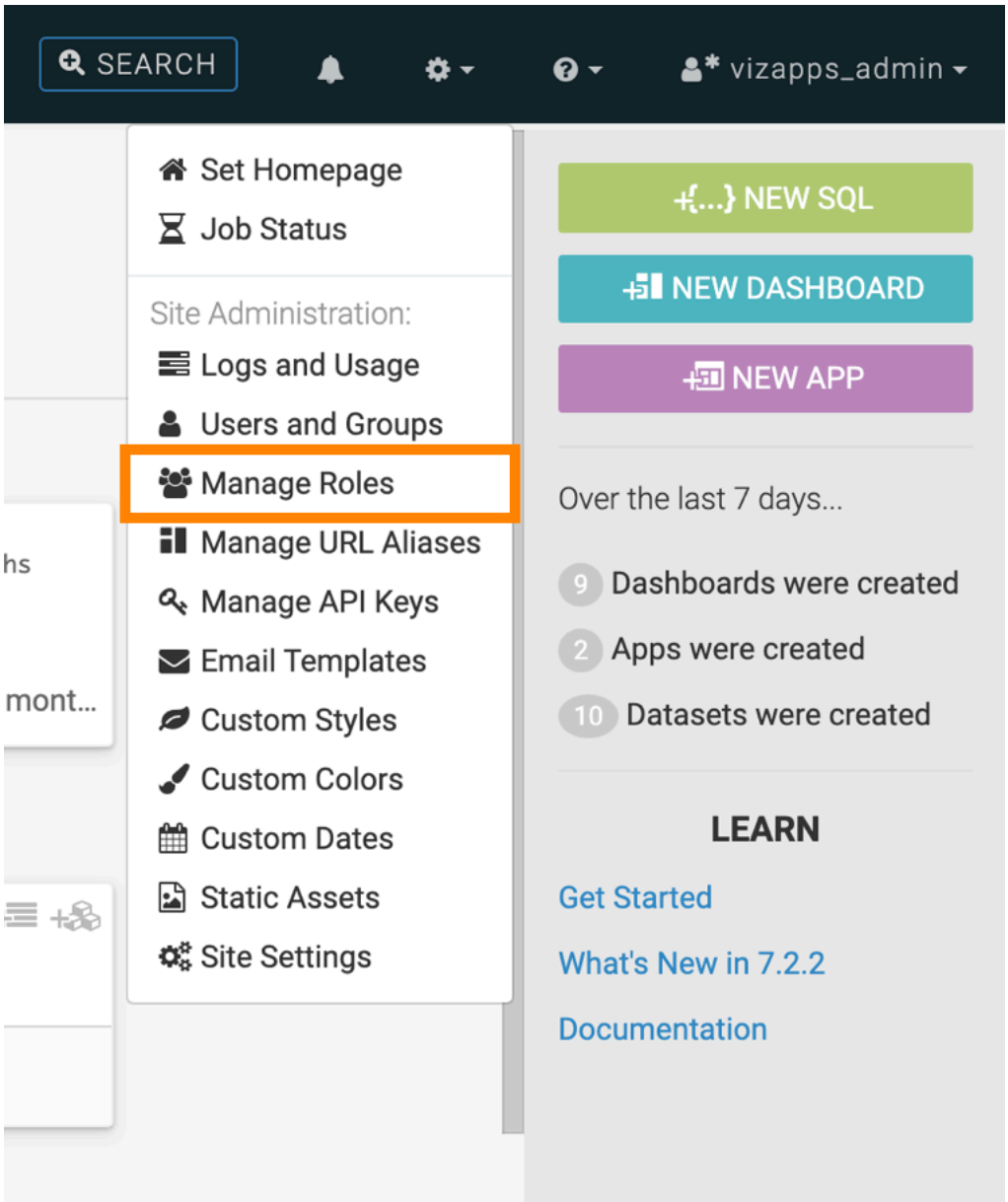

3. In the Manage Roles interface, find the role you wish to edit and click the (edit) icon next to the respective role. You can use the Search Role box to find the role you want to edit.

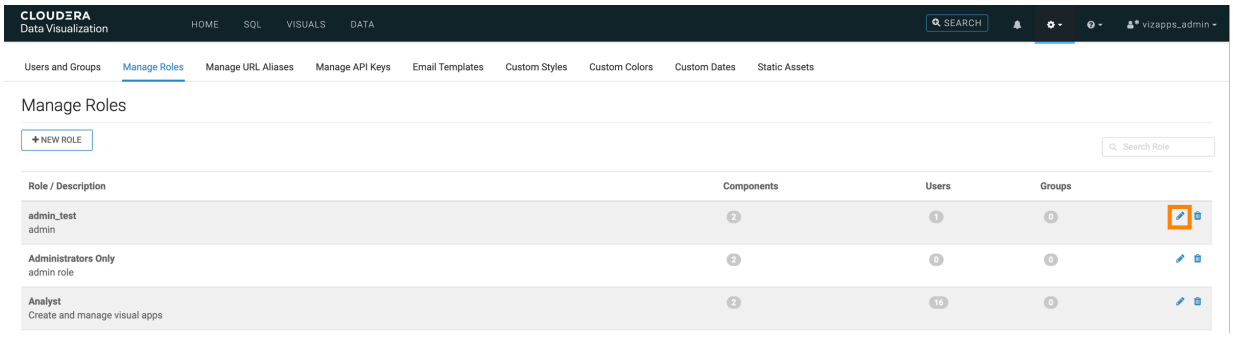

The Role Detail interface is displayed.

4. Switch to the Datasets category on the Privilege tab.

5. Add a new dataset privilege by clicking ADD DATASETS.

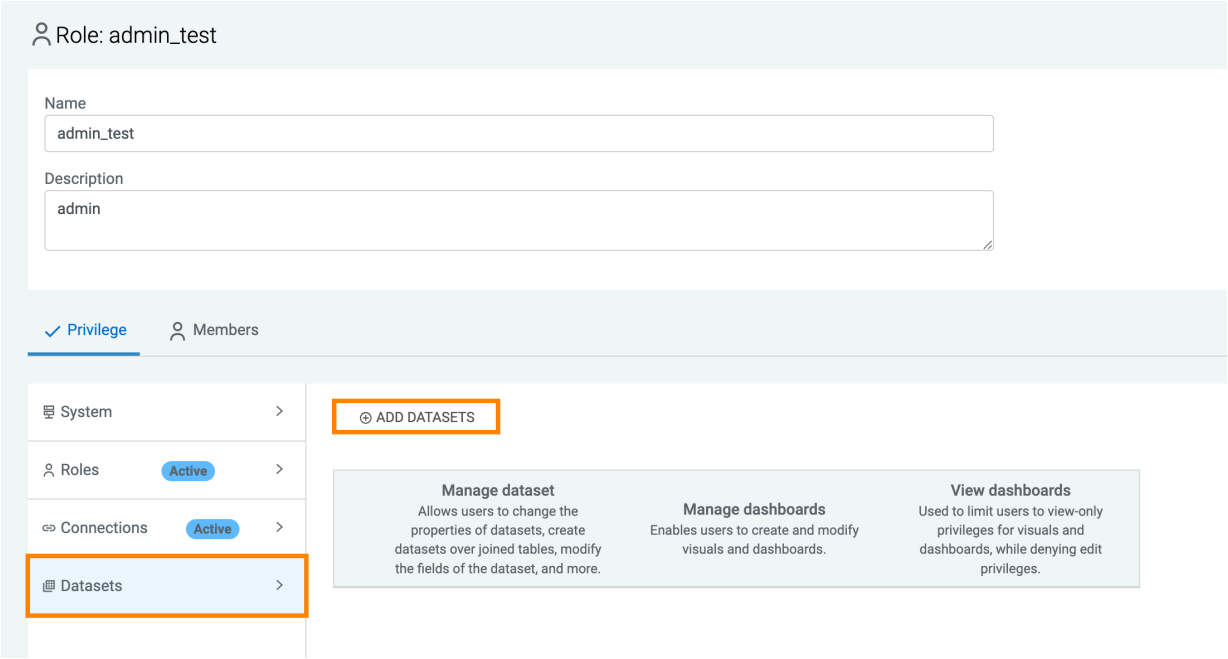

The Add Dataset Privilege modal window appears.

6. From the Connection(s) drop-down menu, select either All connections or one of the individual connections.
7. From the Datasets(s) drop-down menu, select either All datasets or specific datasets.

   When the dataset is added, by default, all possible privileges are marked for it.

   You can repeat this step to include additional connections if necessary.

   You can remove a connection from the list by clicking the Delete icon.
8. Select the checkboxes for the actions that you want to permit for the connection(s) you have added.
   - Manage dataset
   - Manage dashboards
   - View dashboards

   Note:

   Privilege dependency information:

   The Manage dashboards and View dashboards permissions are mandatory if the privilege includes the Manage dataset permission and cannot be removed. The checkboxes for these permissions are pre-filled and fixed.

   If Manage dataset is deselected, Manage dashboards becomes optional and can also be deselected. If Manage dashboards is deselected, View dashboards becomes optional and can also be deselected.

9. After adding roles and selecting the appropriate role privileges, click APPLY CHANGES.

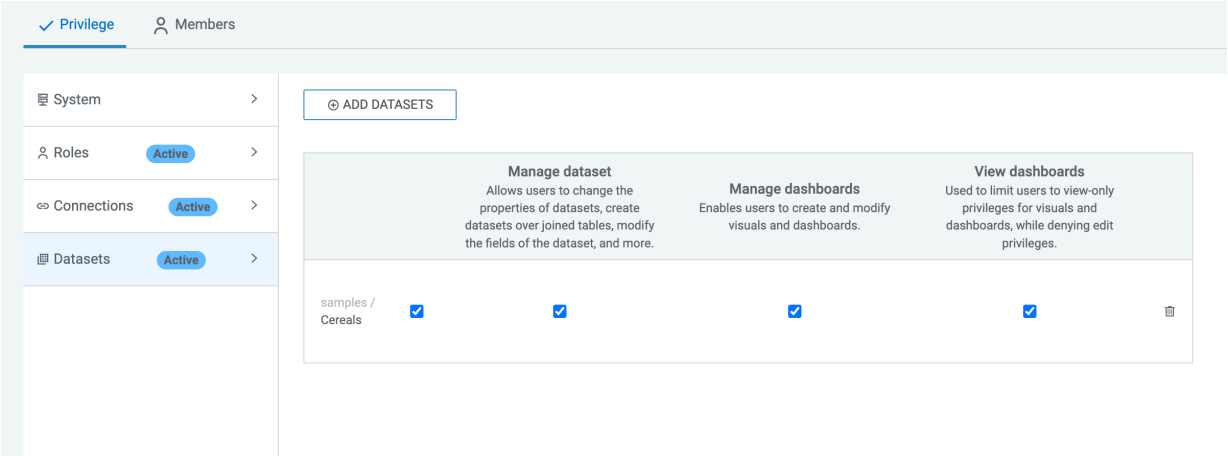

Results

For more information on possible permissions, see *RBAC permissions*.

Related Information

RBAC permissions

Editing role assignments

About this task

The following steps demonstrate how to edit role assignments for a specific role.

In this example, Test Role 1 is used, which was previously defined in *Creating new roles*.

Procedure

1. On the main navigation bar, click the (gear) icon.

2. In the drop-down menu, click Manage Roles.

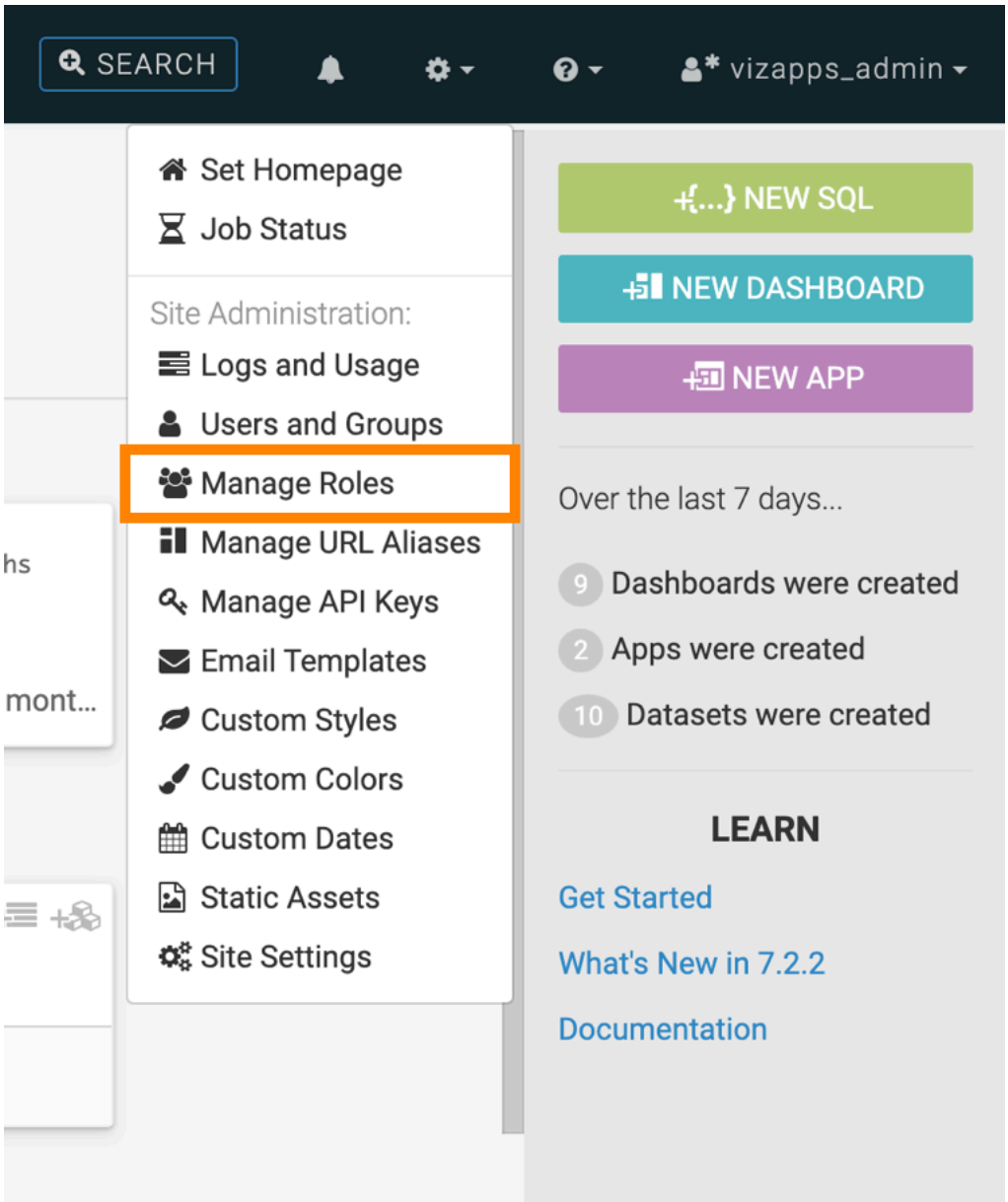

3. In the Manage Roles interface, find the role you wish to edit and click the (edit) icon next to the respective role. You can use the Search Role box to find the role you want to edit.

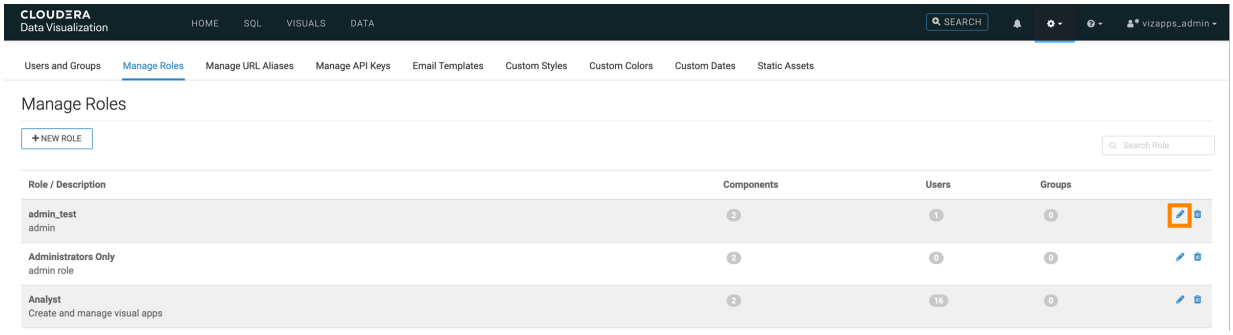

The Role Detail interface is displayed.

Note: If you do not have any roles defined, see the instructions in *Creating new roles*.

4. Click the Members tab and proceed to editing member roles through one of the following procedures:
   - *Assigning roles to a single user*
   - *Assigning roles to user groups*
   - *Assigning multiple roles to multiple users*

Related Information

Creating new roles
Assigning roles to a single user
Assigning roles to user groups
Assigning multiple roles to multiple users

Assigning roles to users

About this task

The following steps demonstrate how to add users to a role.

Procedure

1. Navigate to the Members tab of the Role Detail interface.

2. Click EDIT USER(S).

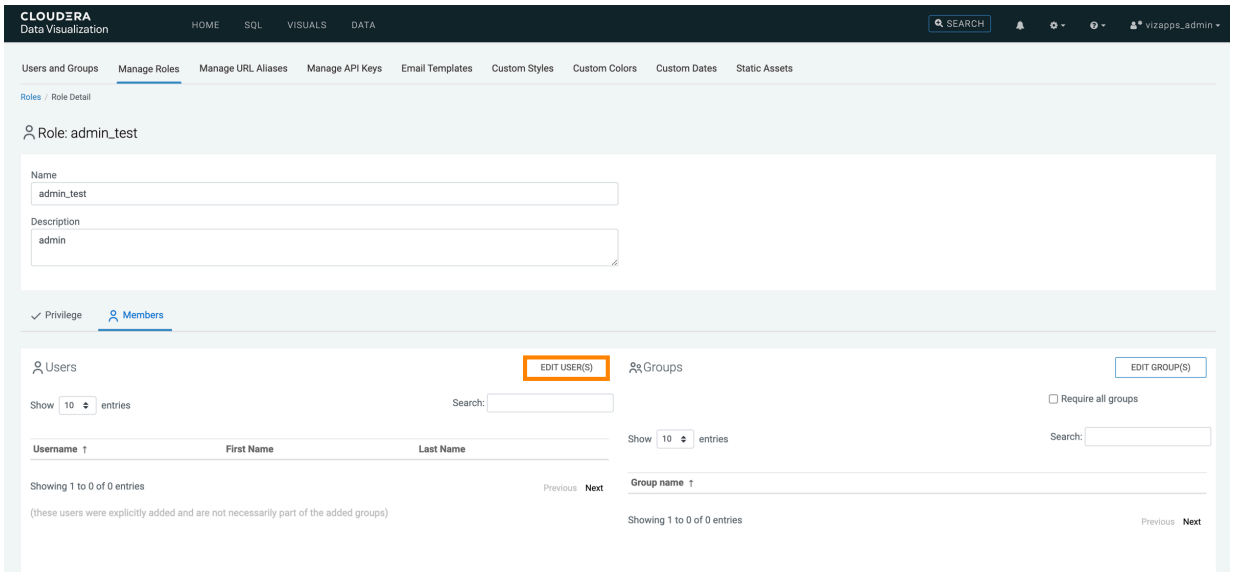

The Role Assignment modal window is displayed.

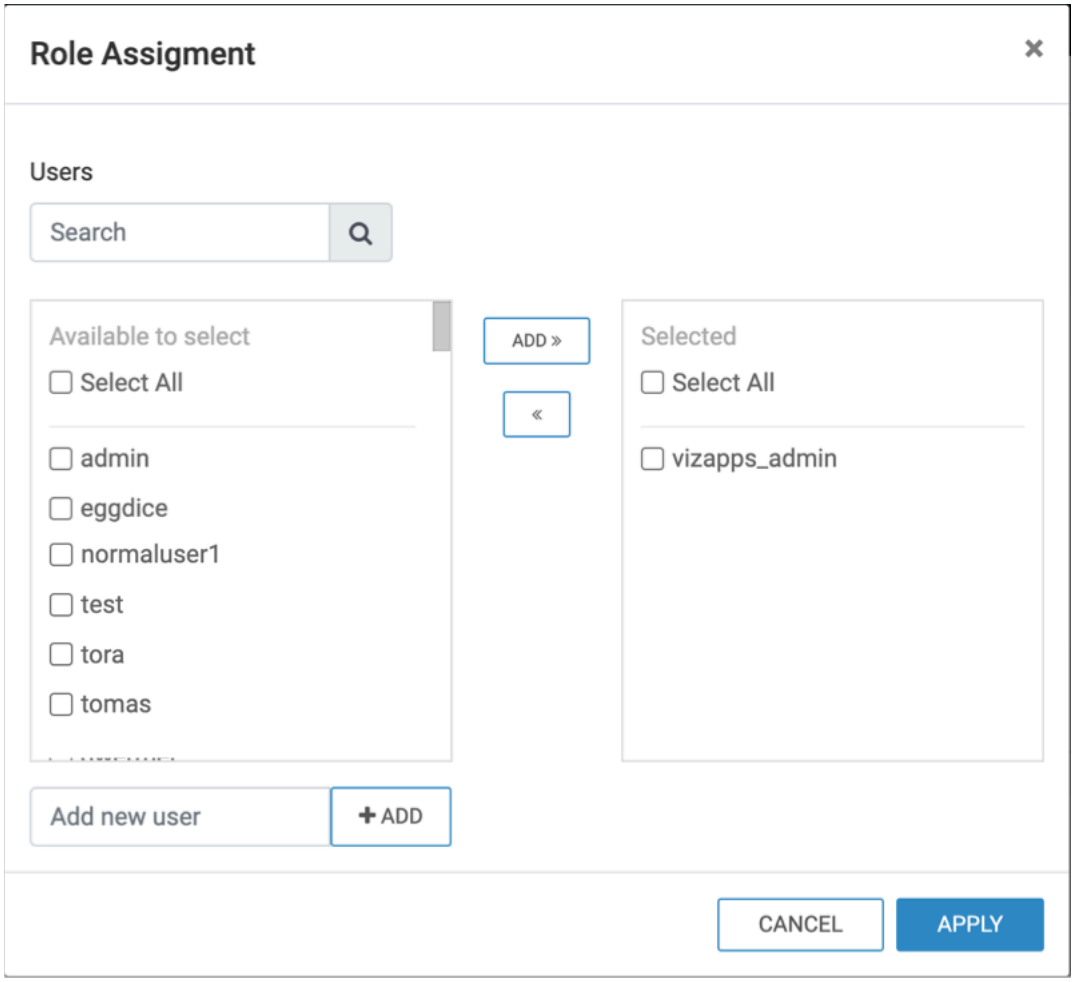

3. There are several options to add a user as a member:
   - Search

     If you have a long list of users in the Users section, use the Search box to find specific names, select them from the sub-list, and click ADD>> to move them to the right side of the modal window.

     When you are ready, click APPLY.
   - Select

     In the Users section, select the users to assign to the role and click ADD>> to move them to the right side of the modal window.

     When you are ready, click APPLY.
   - Select All

     To assign all users to Users, mark Select All and click ADD>> to move them to the right side of the modal window.

     When you are ready, click APPLY.
   - Adding externally defined users (LDAP authentication)

     The Role Assignment modal supports adding usernames that are not stored locally. For example, you can add known usernames available through LDAP authentication using this method.

     Enter the new user name and click ADD>> to add externally defined users. After the new user name appears in the Users section, select it and click ADD>> to move the new user to the right side of the modal window.

     When you are ready, click APPLY.
   - Remove

     To remove users from the Users list, select them from the right side panel and click <<.

     When you are ready, click APPLY.
4. Click APPLY CHANGES.

   The Users list is updated.

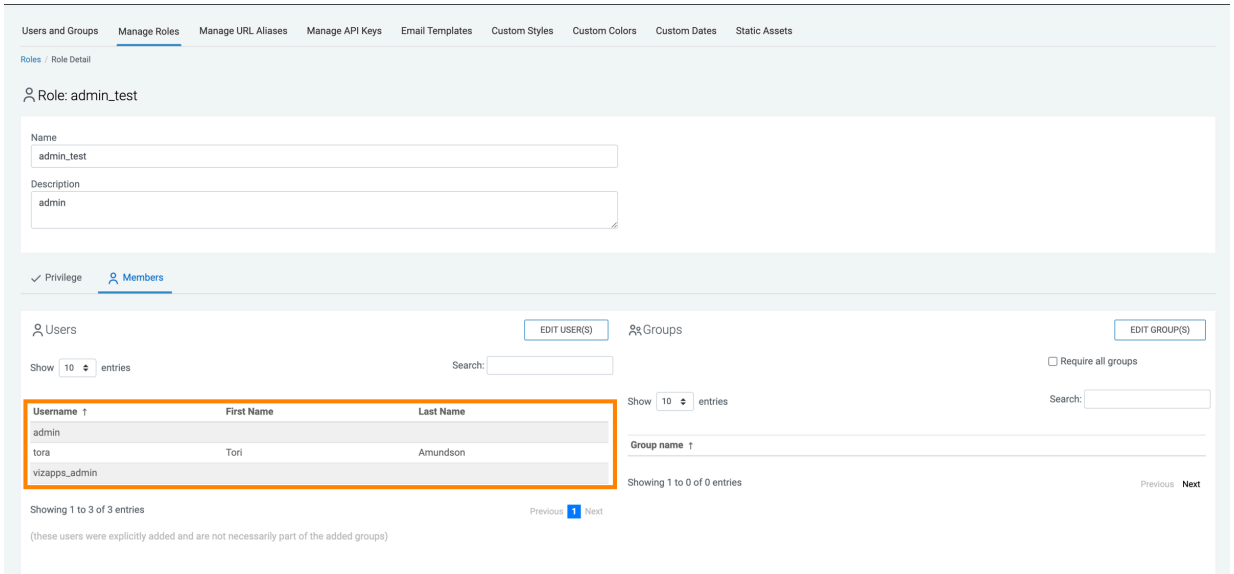

Assigning roles to user groups

About this task

The following steps demonstrate how to add user groups to a role.

Procedure

1. Navigate to the Members tab of the Role Detail interface.

   The Require all groups option ensures that only members of ALL groups listed in the role membership fields have the role's defined access. It determines whether members must belong to all listed groups or any one of them to access the role's privileges.

   Note:

   - If Require all groups is checked, only users belonging to both groups will receive the role's privileges.
   - Exceptions to group membership include explicitly named users and users imported through Ranger groups.

2. Click EDIT GROUP(S).

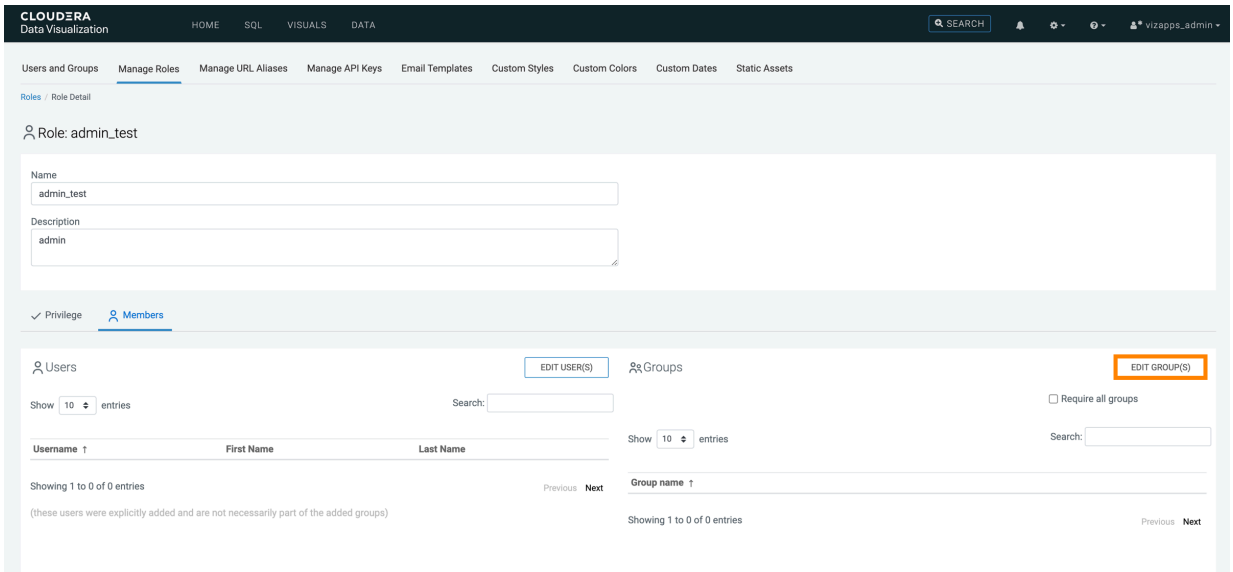

The Role Assignment modal window is displayed.

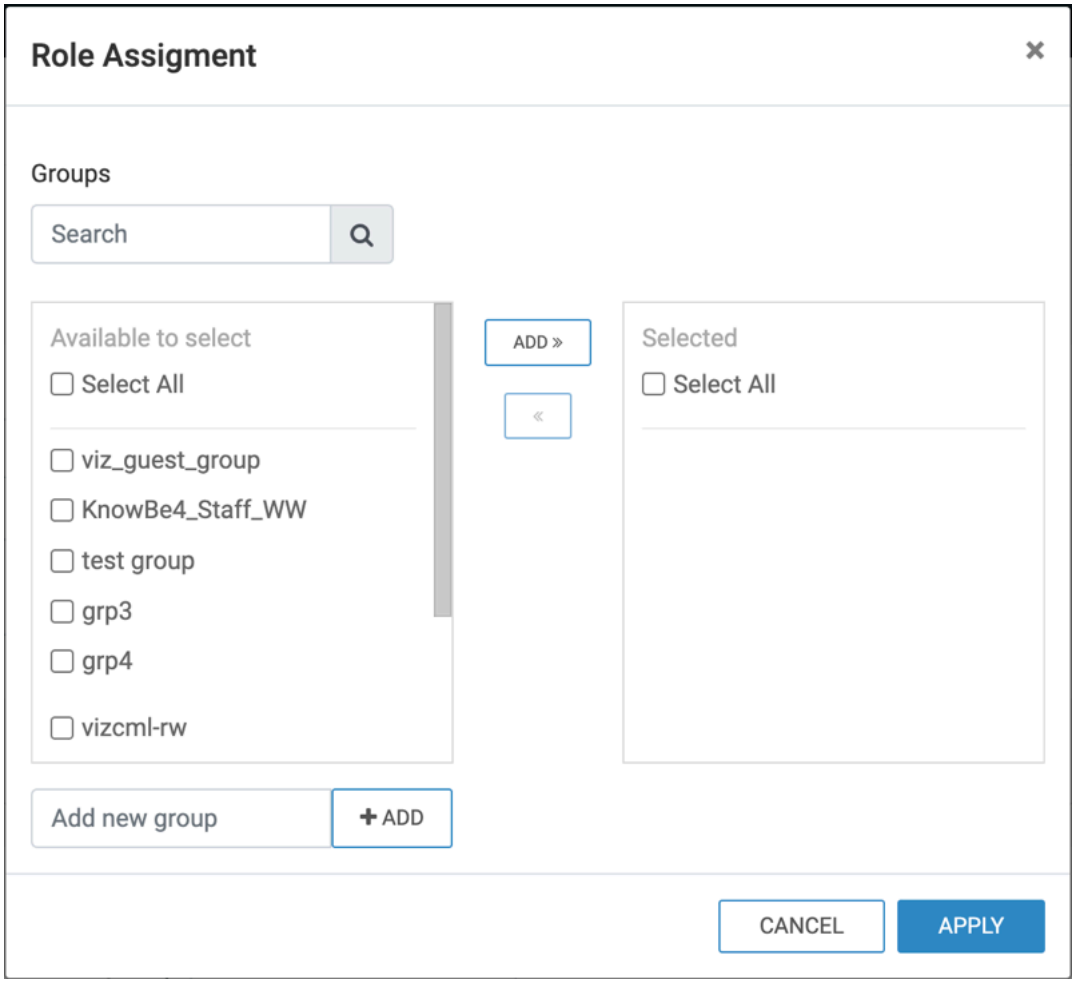

3. There are several options to add groups to role membership:
   - Search

     If you have a long list of groups in the Groups section, use the Search box to find specific group names, select them from the sub-list, and click ADD>> to move them to the right side of the modal window.

     When you are ready, click APPLY.
   - Select

     In the Groups section, select the groups to assign to the role and click ADD>> to move them to the right side of the modal window.

     When you are ready, click APPLY.
   - Select All

     To assign all groups to Groups, mark Select All and click ADD>> to move them to the right side of the modal window.

     When you are ready, click APPLY.
   - Adding externally defined users (LDAP authentication)

     The Role Assignment modal supports adding user groups to the list of assignees that are not stored locally. For example, you can add known groups available through LDAP authentication using this method.

     Enter the new group name and click ADD>>. After the new group name appears in the Members section, select it and click ADD>> to move the new group to the right side of the modal window.

     When you are ready, click APPLY.
   - Remove

     To remove users from the Members list, select them on the right side panel and click <<.

     When you are ready, click APPLY.
4. Click APPLY CHANGES.

   The Users list is updated.

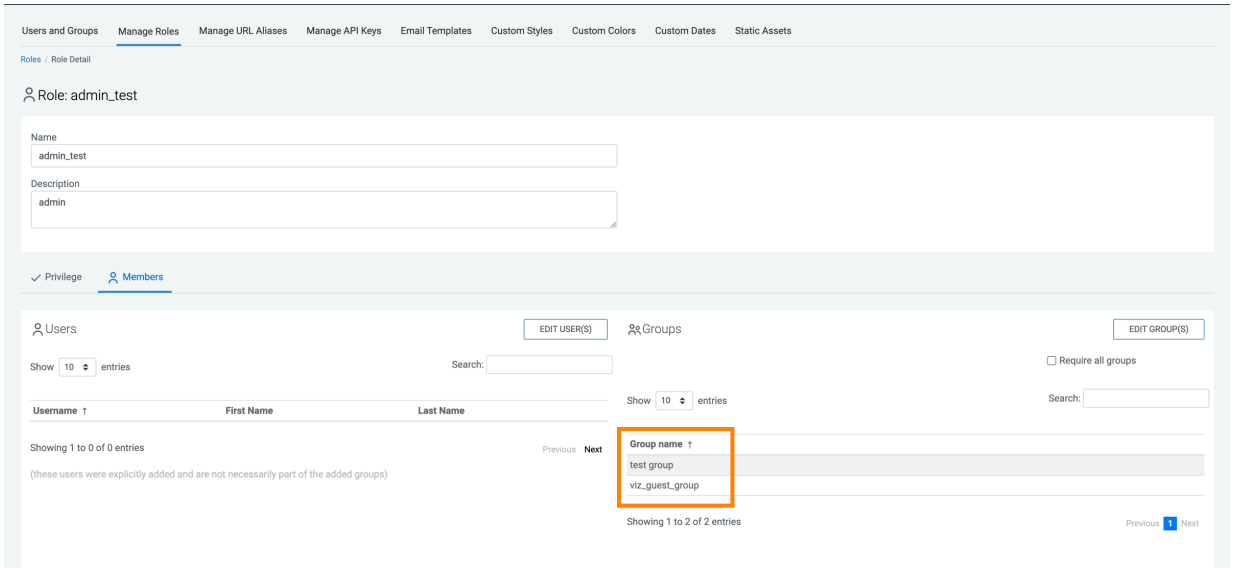

Assigning multiple roles to multiple users

About this task

The following steps demonstrate how to add multiple users to existing roles in the Users & Groups interface of Cloudera Data Visualization.

Procedure

1. Click the Gear icon on the main navigation bar.
2. In the drop-down menu, select Users & Groups.

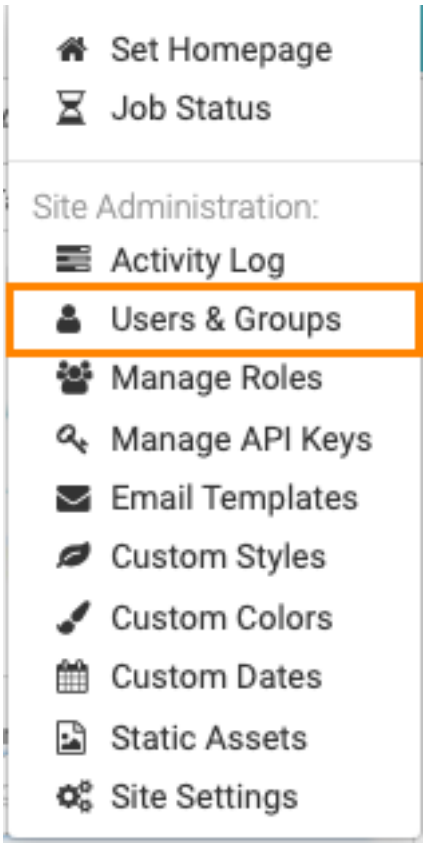

The Manage Users & Groups interface is displayed, open on the Users tab.

3. Choose the users you wish to assign to roles by selecting their respective checkboxes.
   You can use the Search box to find the users you want to edit.

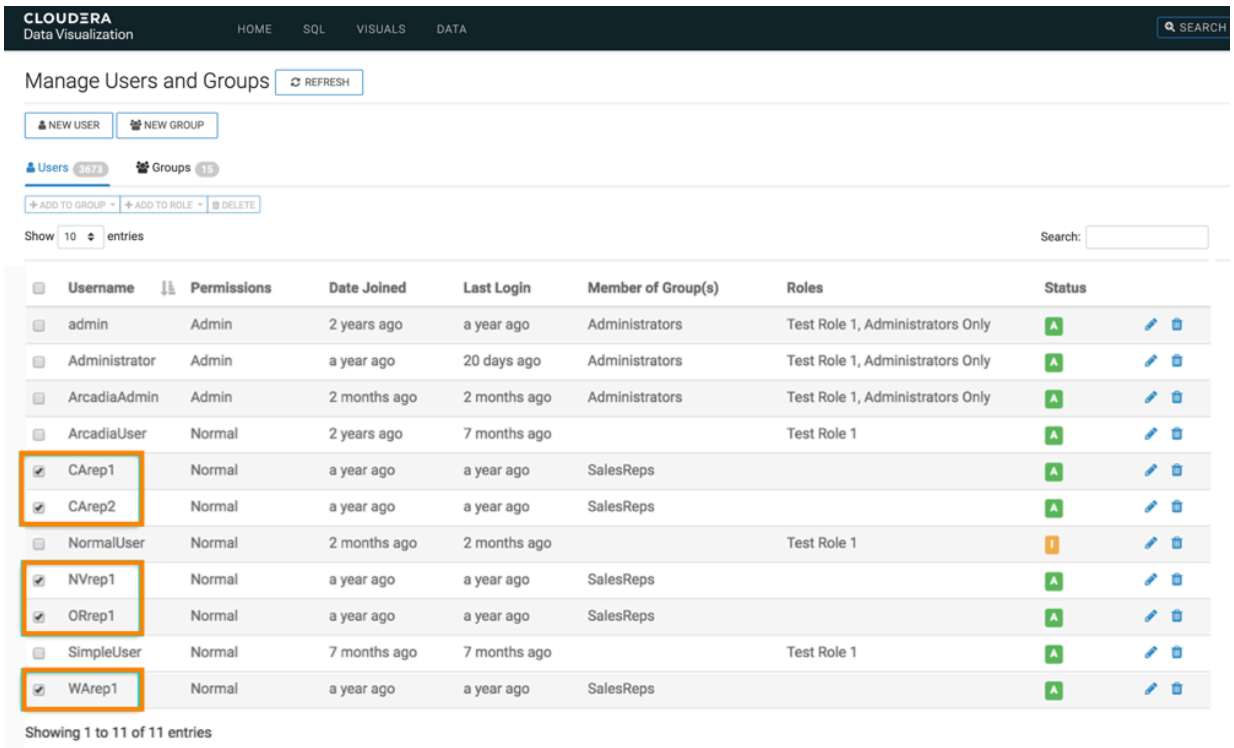

4. Click ADD TO ROLE.

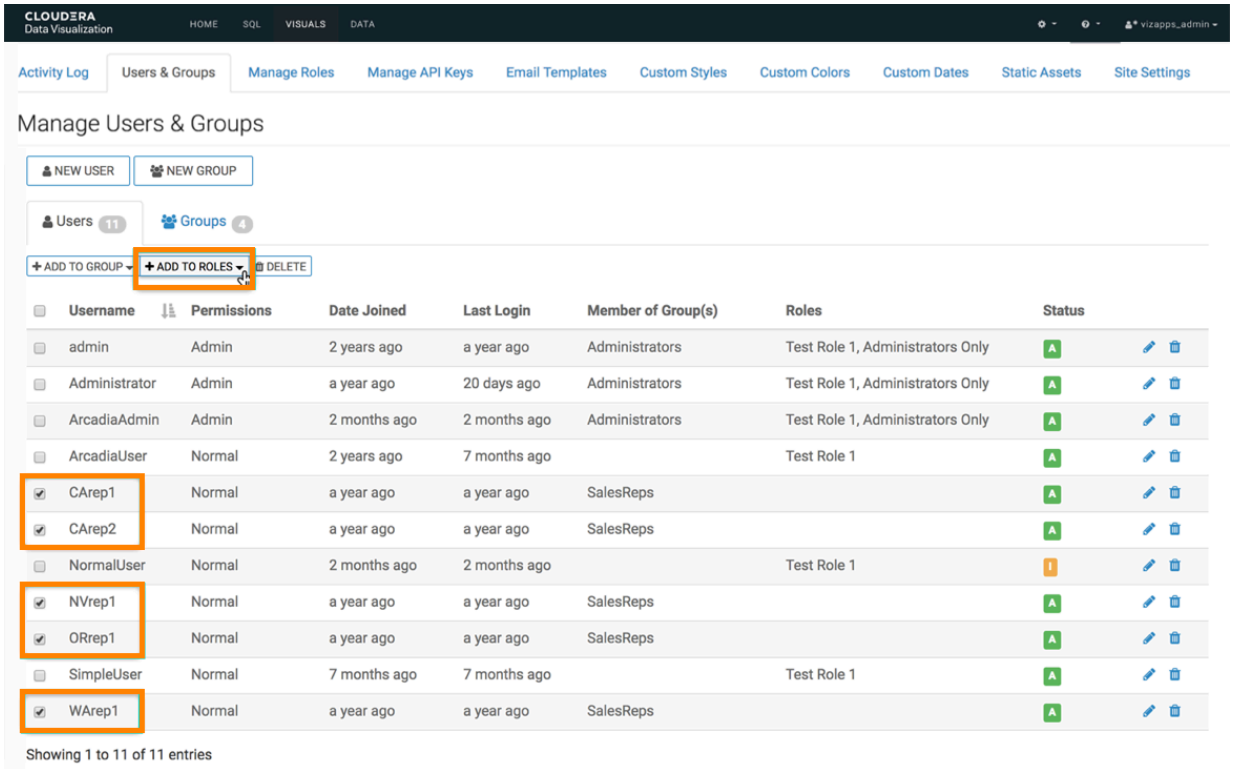

A drop-down menu is displayed, showing a list of available roles.

5. Check the checkboxes next to the roles you want to assign to the selected users and click Save.

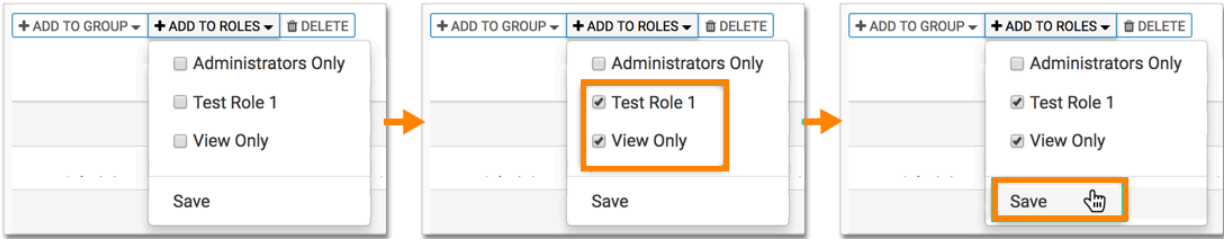

Results

You can see the changes to the information in the Users & Groups interface.

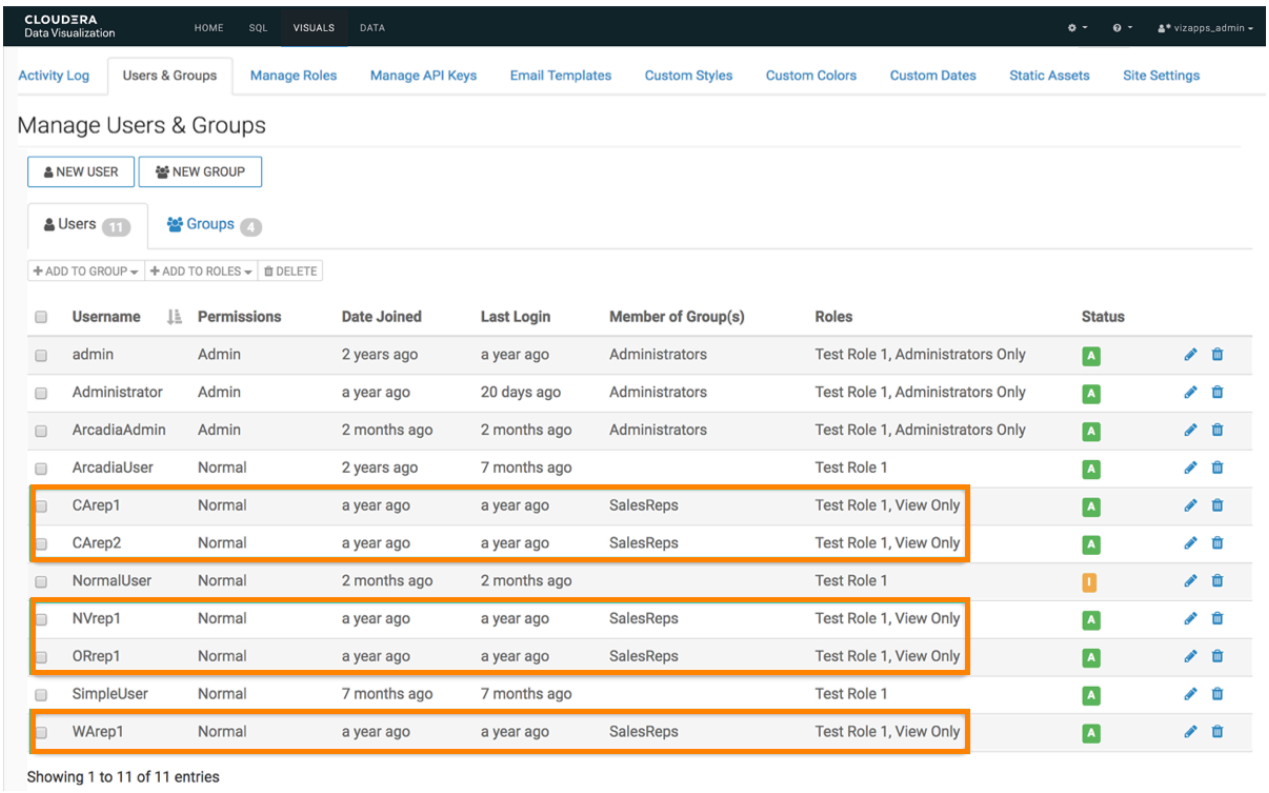

Username	Permissions	Date Joined	Last Login	Member of Group(s)	Roles	Status
admin	Admin	2 years ago	a year ago	Administrators	Test Role 1, Administrators Only	A
Administrator	Admin	a year ago	20 days ago	Administrators	Test Role 1, Administrators Only	A
ArcadiaAdmin	Admin	2 months ago	2 months ago	Administrators	Test Role 1, Administrators Only	A
ArcadiaUser	Normal	2 years ago	7 months ago		Test Role 1	A
CArep1	Normal	a year ago	a year ago	SalesReps	Test Role 1, View Only	A
CArep2	Normal	a year ago	a year ago	SalesReps	Test Role 1, View Only	A
NormalUser	Normal	2 months ago	2 months ago		Test Role 1	I
NVrep1	Normal	a year ago	a year ago	SalesReps	Test Role 1, View Only	A
ORrep1	Normal	a year ago	a year ago	SalesReps	Test Role 1, View Only	A
SimpleUser	Normal	7 months ago	7 months ago		Test Role 1	A
WArep1	Normal	a year ago	a year ago	SalesReps	Test Role 1, View Only	A

Deleting roles

Deleting a role in the role-based access control (RBAC) system is a straightforward process that helps maintain access control efficiency.

About this task

Note: This feature is only available to users with Manage roles and users privileges.

The following steps demonstrate how to add or change privileges defined for a specific role.

Procedure

1. In the Manage Roles interface, identify the role you wish to delete from the list of existing roles.
   You can use the Search Role box to find the role you want to delete.

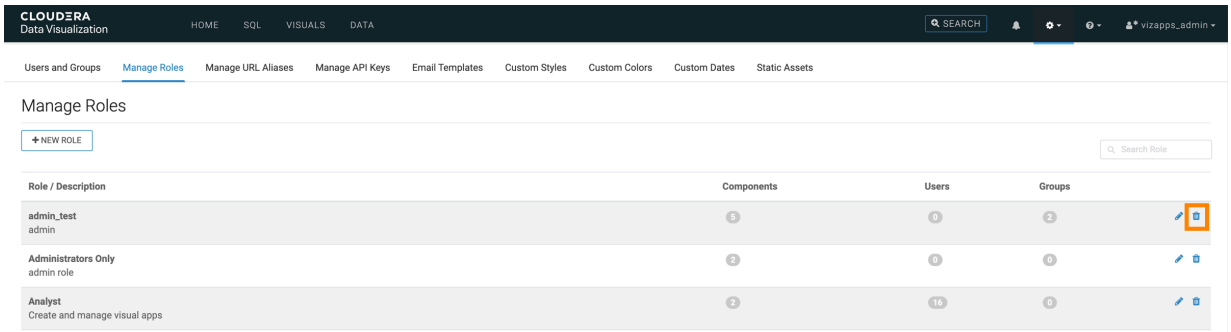

Note: Ensure that the role you are deleting is no longer required and does not impact any ongoing access control configurations.

2. Click the Trash can icon located next to the role.
   A Delete Confirmation modal window will appear, requesting confirmation for the deletion.
3. Click DELETE.

Related Information

Creating new roles

Cloudera Data Visualization 7.2.9

Workspaces

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Workspaces in Cloudera Data Visualization 4

Private workspace 6

Public workspace 6

Custom workspace 7

Workspaces in Cloudera Data Visualization

Cloudera Data Visualization uses workspaces to manage security and workflow aspects of data visualization within enterprise organizations. To manage and share your visuals in in the most effective way, it is crucial to understand the different workspace types available in the system.

When you work with Cloudera Data Visualization, all of your visuals are saved within a workspace. There are three main types of workspaces: private, public, and custom.

Private workspace

- Visuals created are automatically saved here and are exclusively accessible to you and administrators.
- Within this workspace, you can create dashboards and add visuals and filters.
- Dashboards and visuals exist in only one workspace at a time. To share a visual artifact, move it from your private workspace to a custom or public workspace.

Custom workspace

- Users with appropriate privileges can create custom workspaces for collaborative efforts.
- You can share visuals or dashboards with specific users or user groups by moving them to a custom workspace, configuring access rights as needed.

Public workspace

- Visuals within this workspace are visible to any user with minimal viewing permissions for the relevant dataset.

Workflow insights

To better understand how visuals are organized within workspaces, see the following relationship diagram.

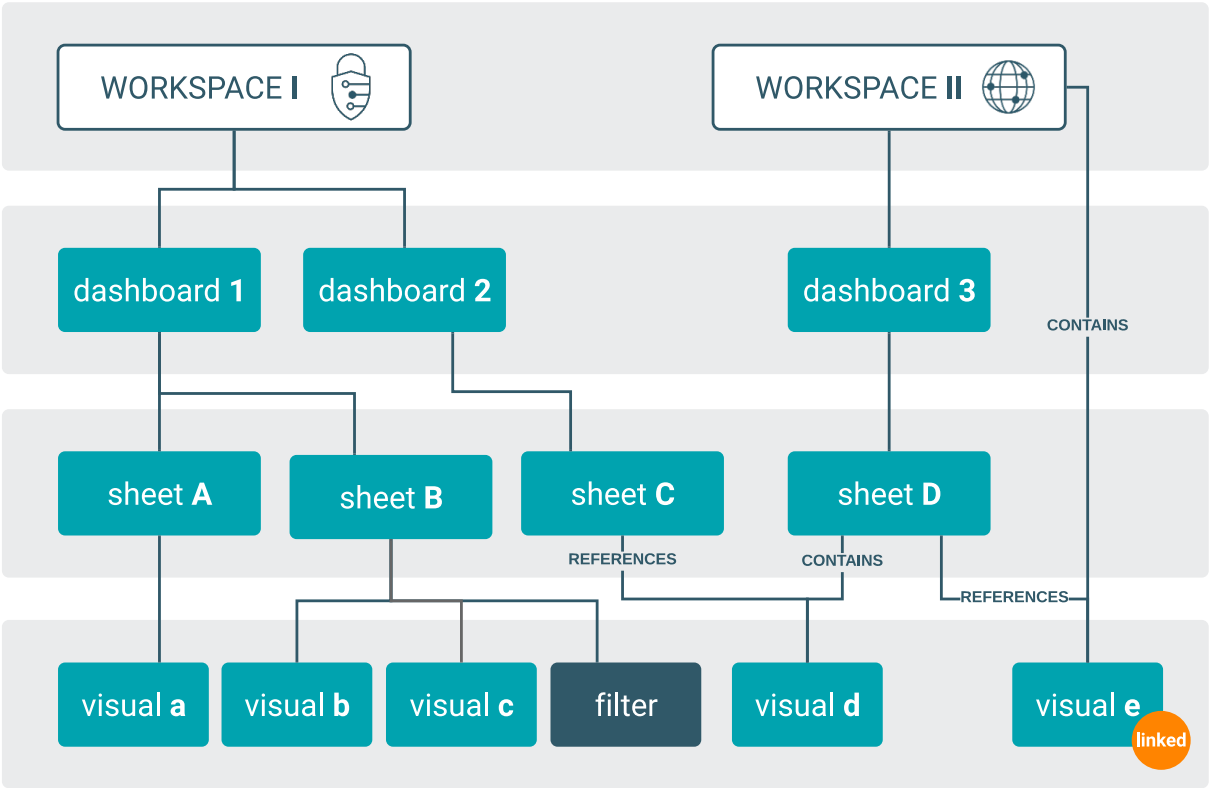

workspace I - PRIVATE

- Only its owner and the administrators have access to this workspace.
- Additional permissions cannot be applied.
- It contains dashboard 1.
  - Dasboard 1 contains two sheets: sheet A and sheet B.
  - Sheet A contains a single image called visual a.
  - Sheet B contains visual b, visual c, and a filter.
- It contains dashboard 2.
  - Dashboard 2 contains a single sheet C, which references visual d.

workspace II - PUBLIC

- Artifacts in this workspace are visible to any user with minimal viewing permissions to the relevant dataset.
- It contains dashboard 3.
  - Dashboard 3 contains a single sheet D.
  - Sheet D contains visual d and references visual e.

Related Information

Public workspace
Private workspace
Custom workspace
Creating a workspace

Private workspace

In Cloudera Data Visualization, you always work and save your visuals within a workspace. There are three types of workspaces: private, public, and custom.

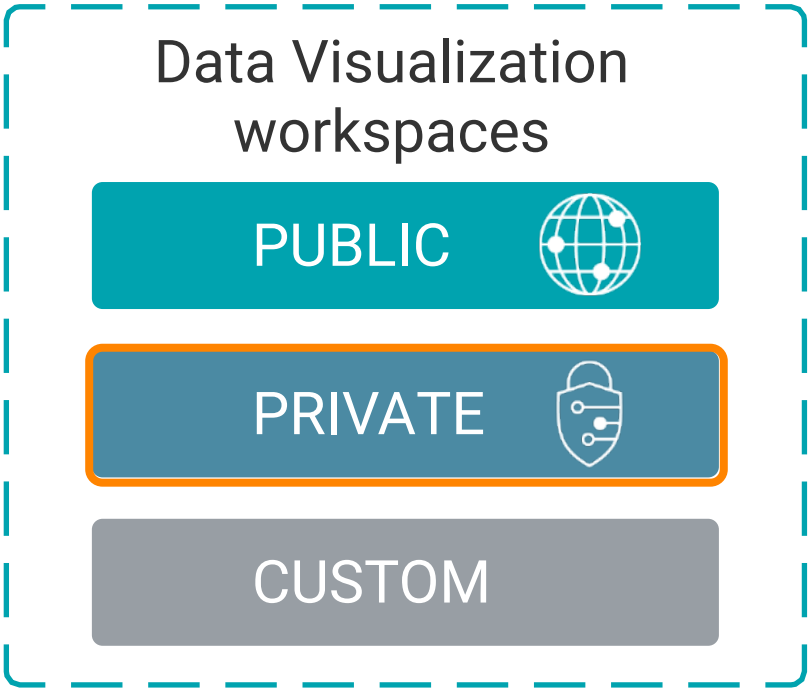

The private workspace is your exclusive working area in the system. All visual artifacts that you create (apps, dashboards, and visuals) are saved in your private workspace by default.

- Every user is automatically assigned a private workspace that cannot be deleted while the user is active in the system.
- Administrators have visibility into private workspaces, but they cannot change the permissions.
- If a user is deleted from the system, their private workspace converts to a custom workspace that is only visible to the administrators.

If you want to collaborate with others in designing dashboards, you can create a custom workspace. If you want to make a dashboard public (accessible to everyone), you can do it by moving the dashboard from your private workspace to the public workspace.

Related Information

Custom workspace

Public workspace

Creating a workspace

Moving dashboards between workspaces

Public workspace

In Cloudera Data Visualization, you always work and save your visuals within a workspace. There are three types of workspaces: private, public, and custom.

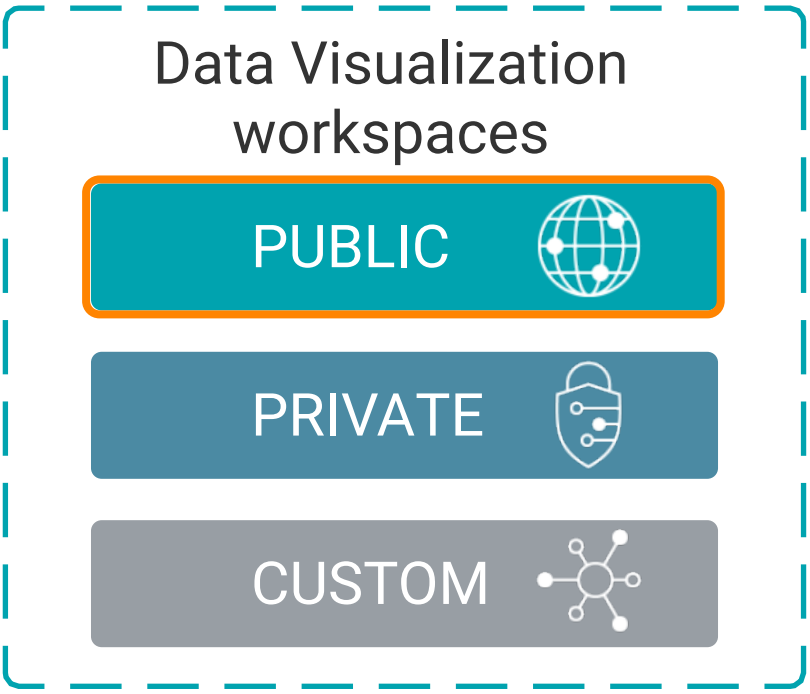

The public workspace contanins visual artifacts that are accessible to any user with minimal viewing permissions for the relevant dataset.

- Each Cloudera Data Visualization instance includes a single, non-deletable public workspace.
- Moving dashboards from private or custom workspaces to the public workspace is equivalent to publishing them across the entire organization.
- Only administrators have the ability to change permissions for the public workspace.

Related Information

Custom workspace

Private workspace

Creating a workspace

Moving dashboards between workspaces

Custom workspace

In Cloudera Data Visualization, you always work and save your visuals within a workspace. There are three types of workspaces: private, public, and custom.

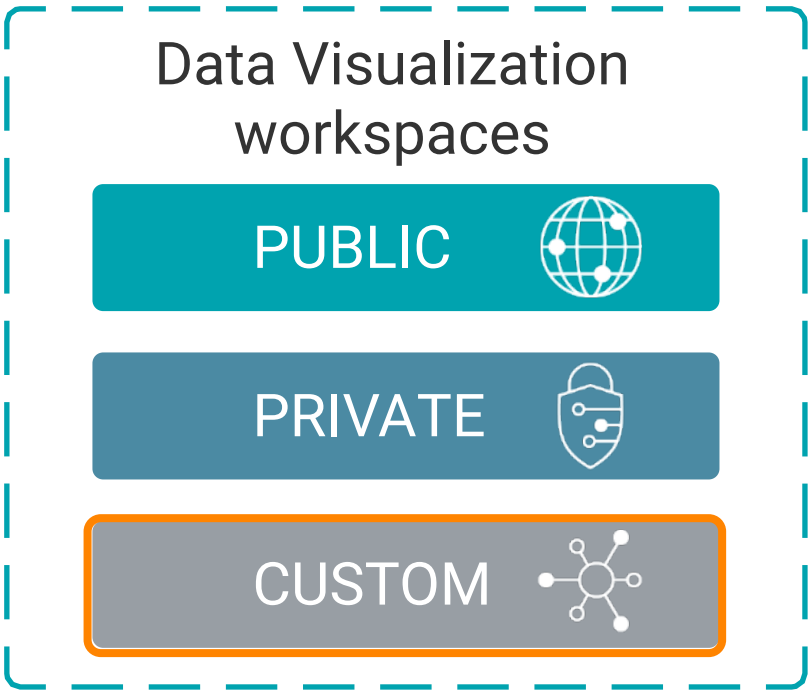

Custom workspaces are also known as shared workspaces and they act as collaborative folders designed for collective and team-oriented work.

- Administrators or regular users equipped with the Create workspaces system-level permission can create custom workspaces.
- Custom workspaces are governed by Access Control Lists (ACLs), enabling precise control over user access and permissions.
- Access to custom workspaces can be configured with three distinct permission levels: View Only, Edit, and Manage. This flexibility ensures tailored access for each user or user group.

For example, if you want to share a dashboard with specific groups of users, you can set up a custom workspace that contains all information that is relevant to those users. You may have a workspace dedicated entirely to the security metrics of your online marketplace, one to the marketing branch of your organization, another one to the sales performance and inventory monitoring, and yet another one to revenues versus expenses analysis for the entire organization.

Related Information

Public workspace

Private workspace

Creating a workspace

Moving dashboards between workspaces

Cloudera Data Visualization 7.2.9

Working with Data

Date published: 2020-10-30
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Working with datasets in Cloudera Data Visualization..... 5

Creating a dataset..... 5
Creating a dataset from a table..... 5
Creating a dataset from a query..... 7
Finding a dataset..... 9
Cloning a dataset..... 10
Deleting a dataset..... 10
Exploring dataset details..... 13
Tables..... 16
Connection..... 16
Description..... 17
Join elimination..... 17
Result cache..... 17
SQL for first table..... 18
Information on creation date and updates..... 18
Checking related dashboards..... 18
Tracking dataset versions [Technical Preview]..... 19
Changing dataset fields..... 20
Editing dataset fields..... 20
Hiding dataset fields from applications..... 24
Changing field aggregation..... 28
Creating calculated fields..... 32
Changing the field display format..... 38
Changing data type..... 48
Specifying geographic fields..... 49
Adding field comments..... 51
Automatically renaming dataset fields..... 55
Custom renaming dataset fields..... 57
Refreshing a dataset..... 59
Editing datasets based on a SQL query..... 61
Restricting columns in datasets based on SQL query..... 62
Restricting rows in datasets based on SQL query..... 65

Working with data models in Cloudera Data Visualization..... 67

Creating a join..... 68
Changing join types..... 72
Editing join details..... 74
Deleting a join..... 78
Applying field display format on sample data..... 79

Working with data profiling..... 81

Working with data extracts in Cloudera Data Visualization..... 84

Creating a data extract..... 84
Exploring data extract details..... 88
Running a data extract..... 88

Creating a dataset from a data extract..... 90
Editing a data extract.....90
Deleting a data extract.....92

Working with datasets in Cloudera Data Visualization

Datasets are defined on the connections to your data, and provide access to the specific tables in the data store.

In Cloudera Data Visualization, visualizations are built from datasets. These datasets provide access to data annd enhance data access and usage.

Related Information

Datasets

Creating a dataset

Cloudera Data Visualization allows you to create datasets based on your data available through a data connection.

About this task

There are two options to create a new dataset:

- Creating a dataset from a table on page 5
- Creating a dataset from a query on page 7

Creating a dataset from a table

Cloudera Data Visualization allows you to define a new dataset based on an exisiting table.

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.
2. Click NEW DATASET near the top of the screen.

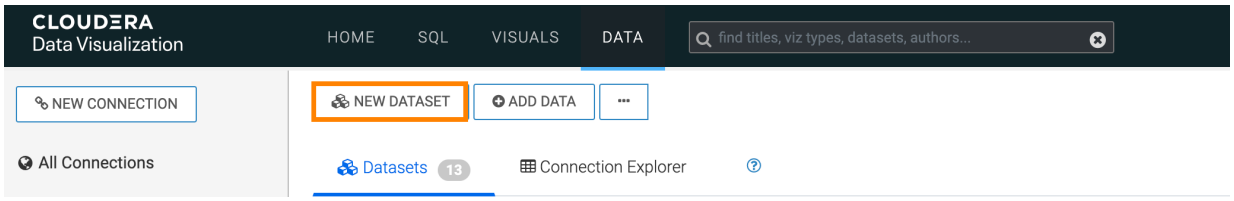

The New Dataset modal window opens.

Note: You can also open this modal window from the Connection Explorer, by clicking New dataset next to the table's name.

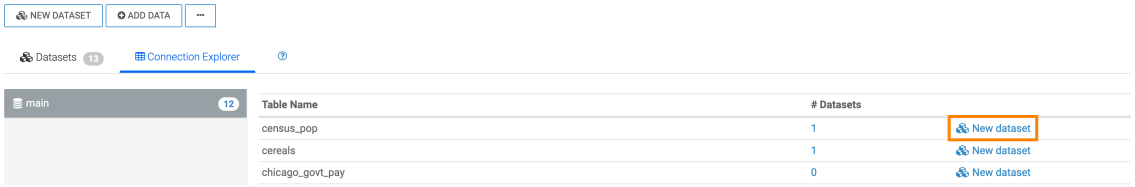

3. In the New Dataset modal window, specify the following values:

   Dataset title

   Provide a name for the new dataset.

   Dataset Source

   Open the menu and select the From Table option.

   Select Database

   Scroll down the list of connected databases to select the correct database.

   Select Table

   Scroll down the list of tables to select the correct table.

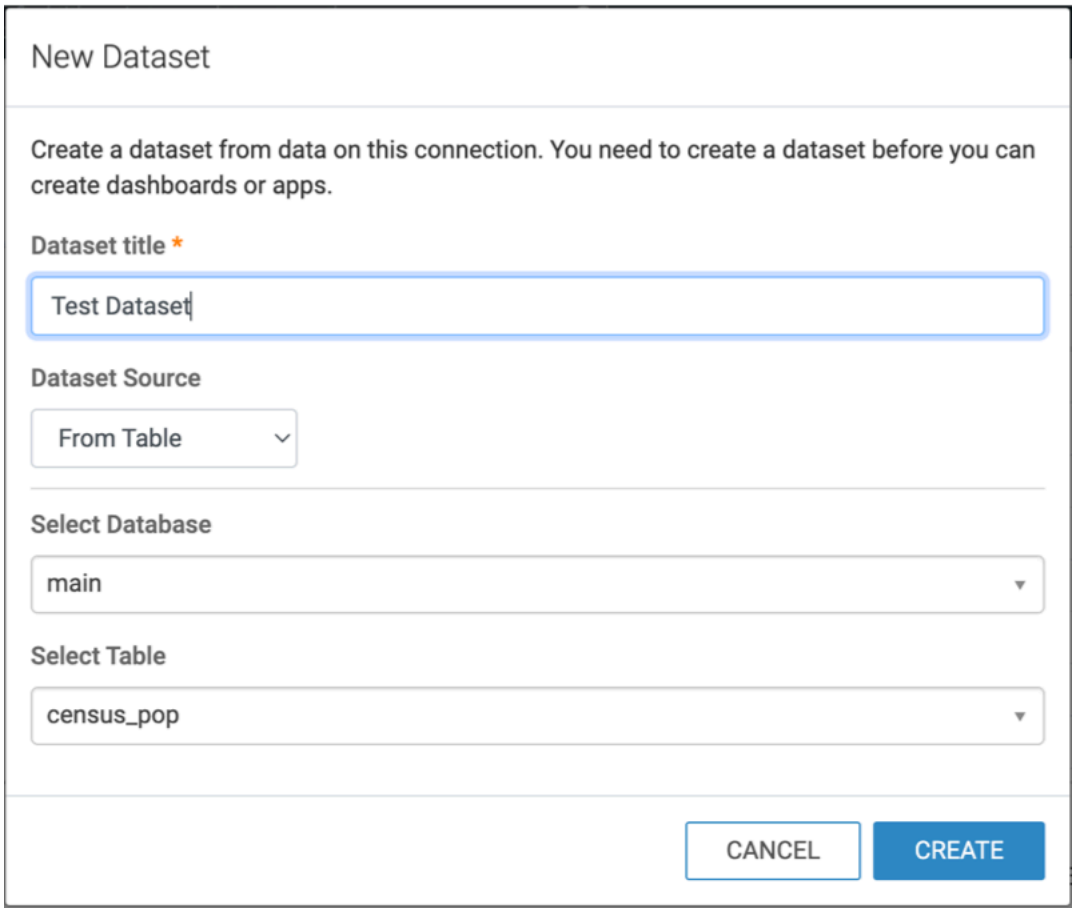

4. Click CREATE.

   You can now see the new dataset on the Datasets tab.

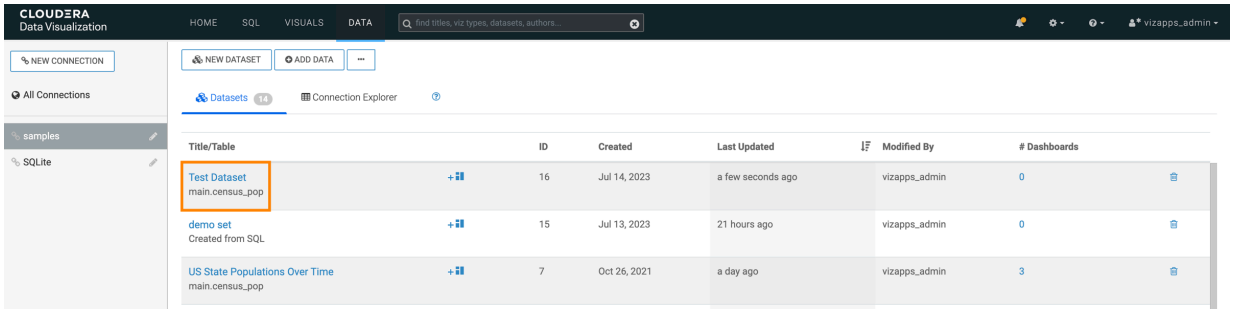

Tip: To find the dataset in the future, you can scroll through the list of datasets available on the connection, or use Search at the top of the page.

Creating a dataset from a query

Cloudera Data Visualization allows you to define a new dataset using a SQL query. This feature makes it easy to restrict access to specific table columns or rows for all users, either for security or for relevancy reasons. It also enables you to specify complex joins and analytic functions at the dataset level.

Using the DATA tab

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.

2. Click NEW DATASET near the top of the screen.

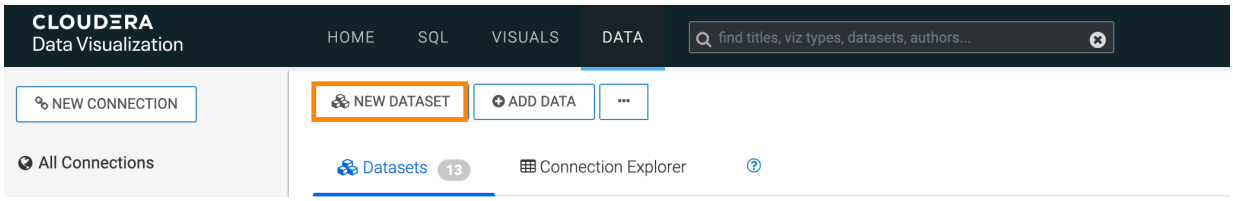

   The New Dataset modal window opens.

   Note: You can also open this modal window from the Connection Explorer, by clicking New dataset next to the table's name.

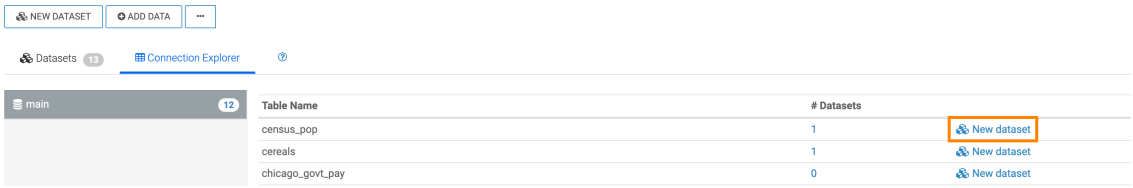

3. In the New Dataset modal window, specify the following values:

   Dataset title

   Provide a name for the new dataset.

   Dataset Source

   Open the menu and select the From SQL option.

   Enter SQL below

   a. Enter the SQL query you want to use to define the new dataset.
   b. Run the query.

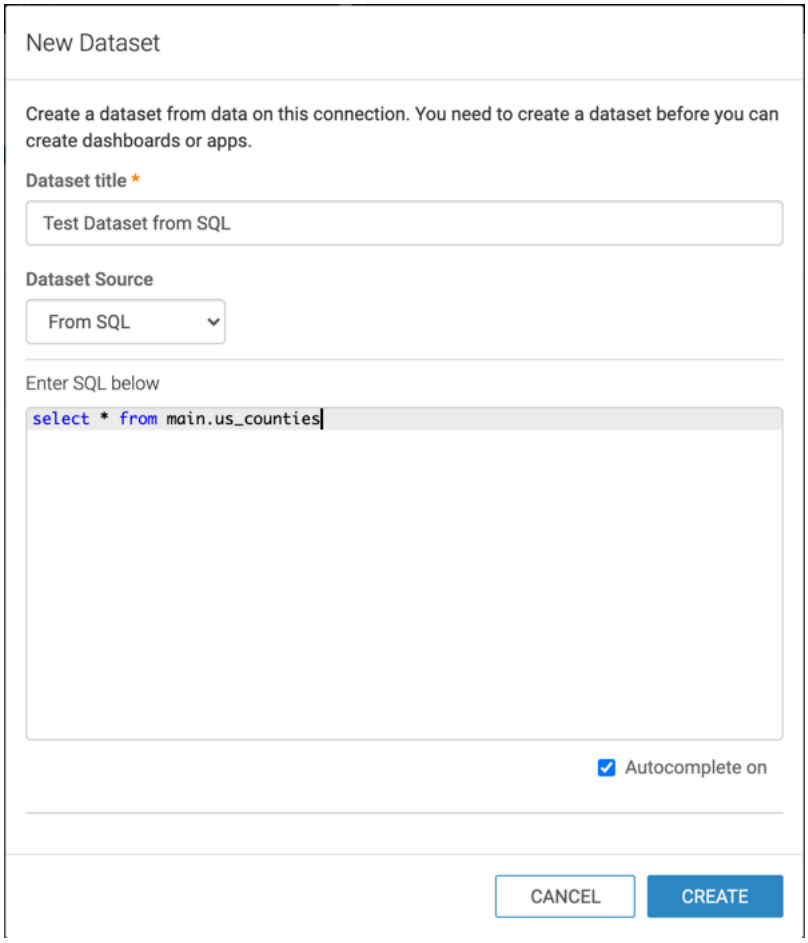

4. Click CREATE.

   You can now see the new dataset on the Datasets tab.

   Tip: To find the dataset in the future, you can scroll through the list of datasets available on the connection, or use Search at the top of the page.

Using the SQL tab

Procedure

1. On the main navigation bar, click SQL.

   The SQL view opens.
2. Select the database and the table.
3. Add the SQL query to the SQL text field.

4. Click RUN.

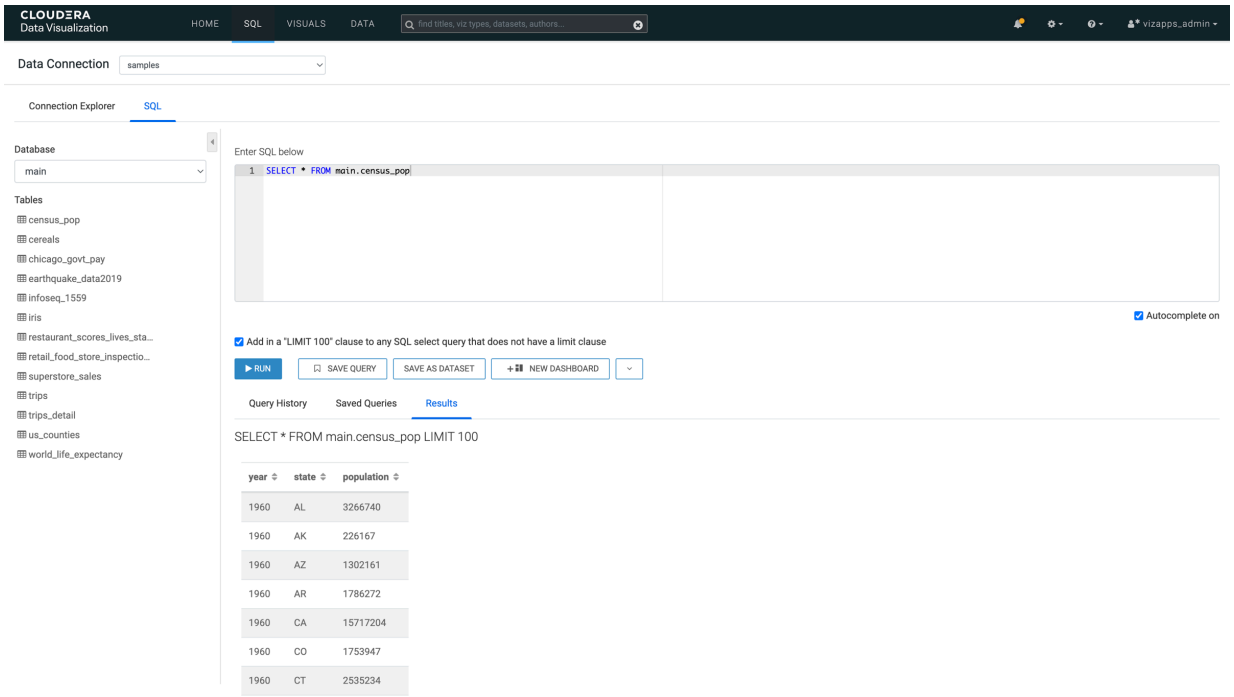

5. Click SAVE AS DATASET.

   The New Dataset modal window opens.

6. Add a title for the dataset.
7. Click CREATE.

   You can now see the new dataset on the Datasets tab.

   Tip: To find the dataset in the future, you can scroll through the list of datasets available on the connection, or use Search at the top of the page.

Finding a dataset

Cloudera Data Visualization makes it easy to find your datasets. You can browse the list of Datasets, check out the Connection Explorer, or use Search.

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.

2. On the left-side panel, select the connection on which the dataset is defined.

3. Use Search on the main navigation bar. Type the search string that matches all or part of the dataset you are looking for.

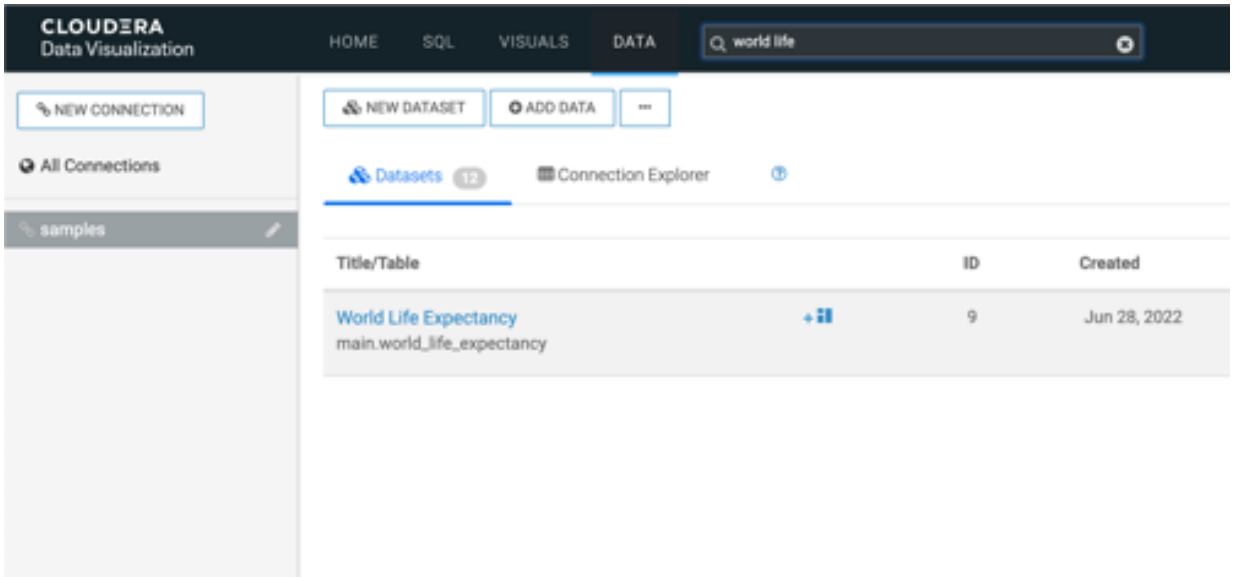

Some examples of search criteria are for example data connection names, data table names, dataset names, visualization names, or application names.

4. Select the correct dataset from the abbreviated list of datasets.

Cloning a dataset

Cloning a dataset in Cloudera Data Visualization enables you to quickly create a duplicate dataset to use as a foundation for new analyses, testing, or experimentation. This is especially useful for exploring different transformations, filters, or display options without affecting the original dataset.

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.
2. Find the dataset you want to clone, by either browsing through the connection and the dataset lists, or by using the Search function.
3. Click the name of the dataset.

   The dataset side navigation panel appears, open on the Dataset Detail view.
4. Click Actions Clone Dataset in the top-right corner.

   The original dataset is cloned. The cloned dataset will appear with the name "Clone of [original dataset name]" and will have a new ID, inheriting all other settings and attributes of the original dataset.

Deleting a dataset

In Cloudera Data Visualization, you can delete a dataset without removing the underlying data in the database.

About this task

You can delete a dataset from either the Datasets view or the Dataset Detail view.

Note: Deleting a dataset removes all associated dashboards and visuals, both linked and unlinked. However, it does not delete the data from the database, and any analytical views defined on the dataset will be preserved.

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.
2. Find the dataset that you want to delete by either scrolling through the list or using the search function.

3. On the row of the dataset, click the icon to delete the dataset.

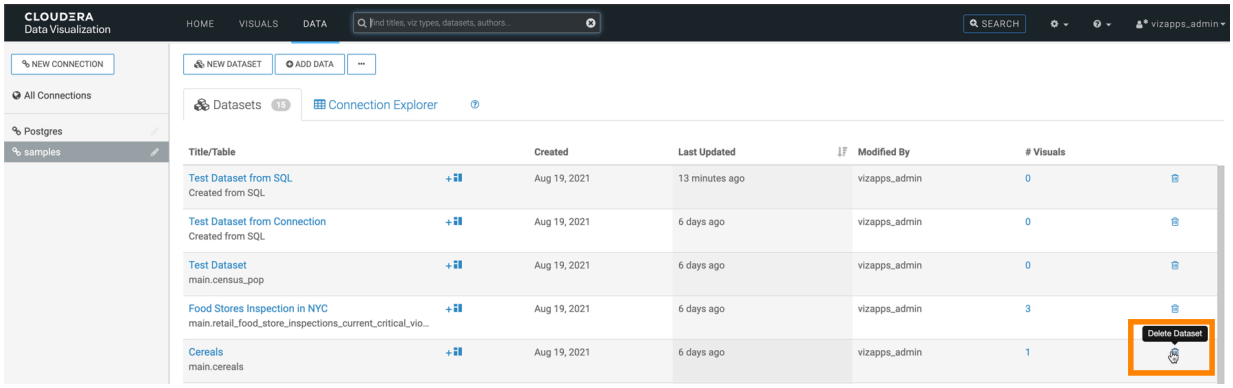

Alternatively, you can click the dataset to open the Dataset Detail view, then click Actions Delete .

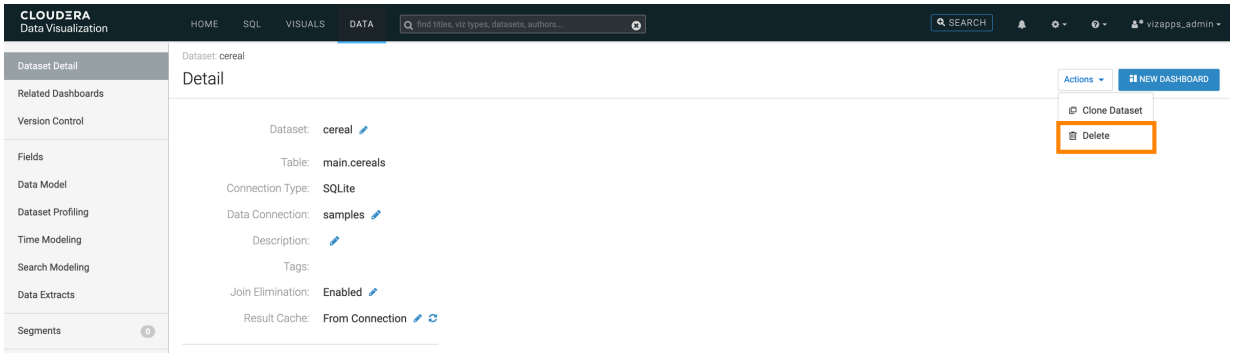

The Delete Dataset modal window opens displaying information about the dashboards and visuals linked to the dataset that will also be deleted with the dataset.

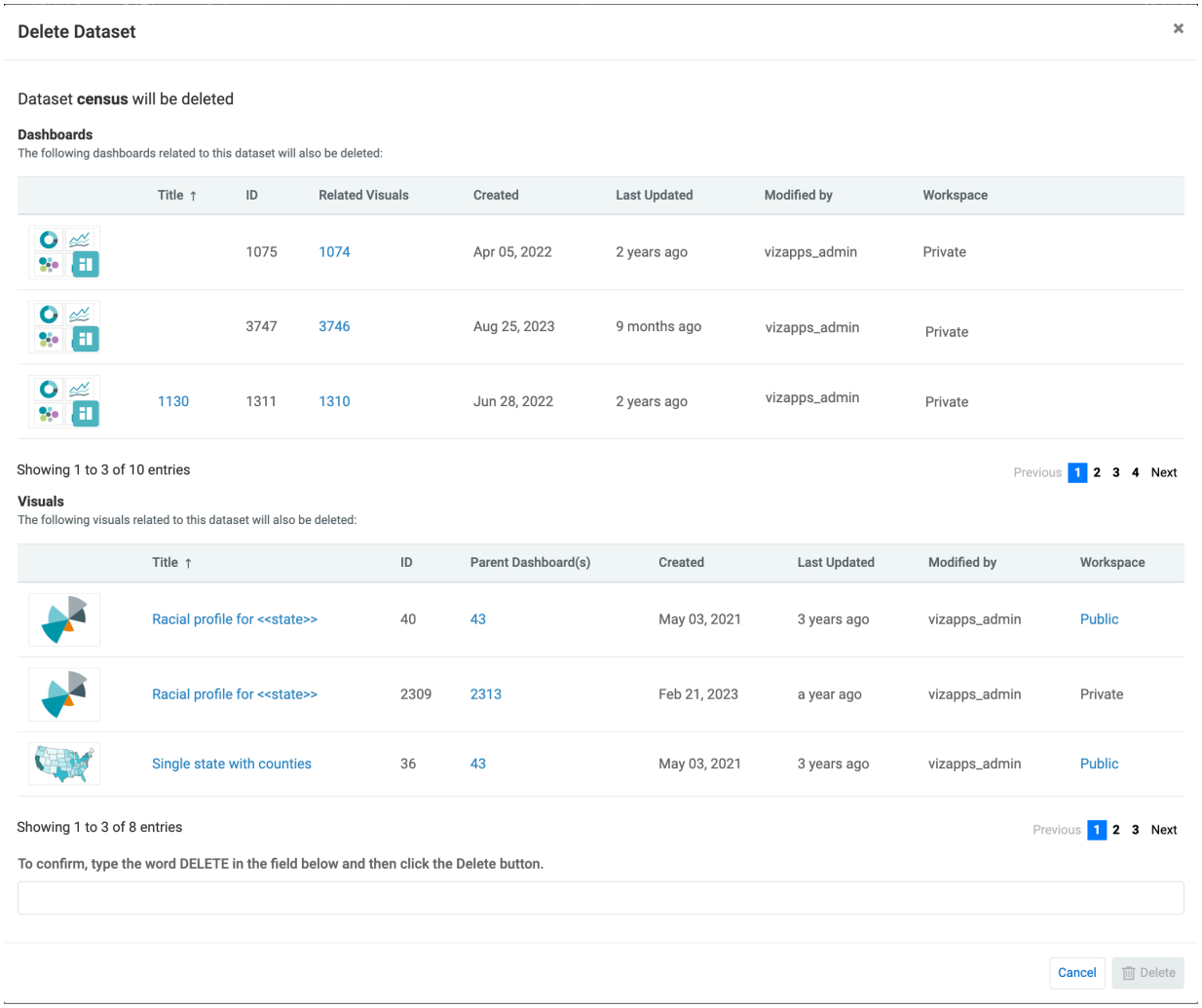

- *Snapshot* shows an icon indicating whether the artifact is a visual or a dashboard.
- *Title* shows the name of the artifact that will also be deleted.
- *ID* shows the identification number of the artifact that will also be deleted.
- *Related Visuals* / *Parent Dashboard(s)* list the number of related artifacts and their IDs.
  - For dashboards, *Related Visuals* include the IDs of visuals contained within the dashboard.
  - For visuals, *Parent Dashboard(s)* include the IDs of dashboards where the visuals appear.
- *Created* shows the creation date of the artifact.
- *Last Updated* shows the time of the last modification.
- *Modified By* shows the username of the last person who modified the artifact.
- *Workspace* indicates where the artifacte is located.

4. After you reviewed the listed dashboards and visuals, type DELETE in the confirmation text field and click Delete to permanently remove the dataset and its associated artifacts.

Exploring dataset details

Cloudera Data Visualization makes it easy to check out and examine the information available about your datasets.

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.
2. Find the dataset you are interested in, either by browsing to a known connection and scrolling, or by using Search.

3. Click the dataset name.

   The dataset side navigation panel appears, open on Dataset Detail view.

   Depending on the dataset definition approach, there are two alternatives:

   - Defined on Table

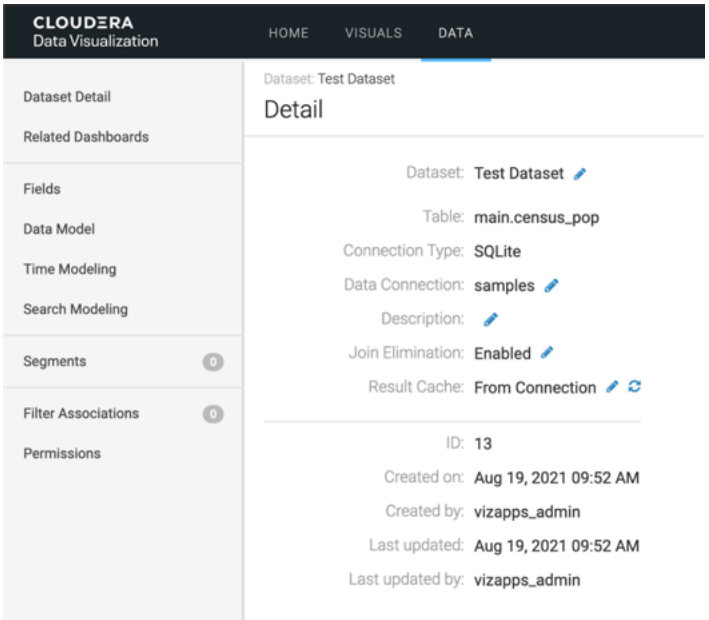

   - Defined on SQL

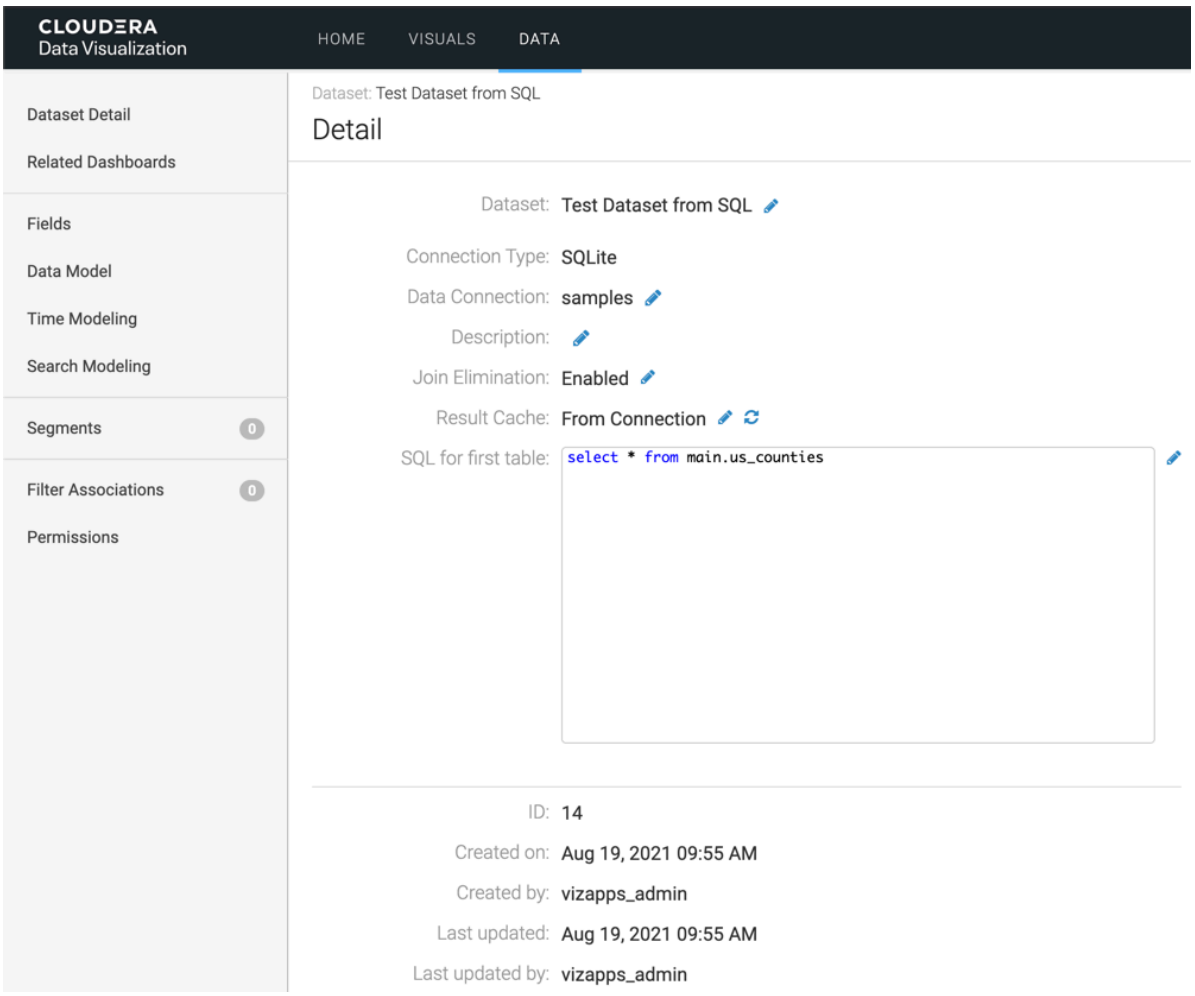

The following information is available on Dataset Detail view, under Detail:

Tables

This is the qualified name of the data source. It appears in the form DatabaseName.DatabaseTable.

Connection

This is the name of the database that hosts the data that appears in the form DataConnection.This feature is ideal for enterprise environments, with dashboards developed on test clusters, and then deployed to a production environment.

About this task

This assumes that the new connection has, at the minimum, the relevant base tables with metadata definitions that match those on the original connection. Users with appropriate permissions can switch the data connection of a dataset by following these steps.

Procedure

1. Click the icon.
2. Choose a different data connection from the menu.
3. Click Save.

Description

This is an optional field. You can add a description of the dataset in the available textbox.

Procedure

1. Click the icon.
2. Enter the descirption in the text box.
3. Click Save.

Join elimination

Join elimination improves query execution and visual rendering in Cloudera Data Visualization.

About this task

Join elimination is available both for left outer and inner joins. It is turned on by default.

When a visual uses fields and expressions that reference only a subset of the joined tables that form the dataset, this feature eliminates the unnecessary joins and access only the necessary subset of the joined tables. This improves query execution speeds, and renders the visuals faster.

Procedure

1. Click the Edit (pencil) icon.
2. You can eanble or disable join elimination:
   - Disabling: select the Disabled option.
   - Enabling: select the Enabled option.
3. Click Save.

Result cache

Each dataset inherits the result caching preferences configured for its connection. The value for this field is From Connection by default, but this may be changed at the level of the dataset.

Procedure

1. Click the icon.
2. You can enable or disable the result cache:
   - Disabling: select the Disabled option.
   - Enabling: select the Enabled option, and specify the Retention Time, in seconds.
3. Click SAVE.

4. To clear the result cache, click the icon.

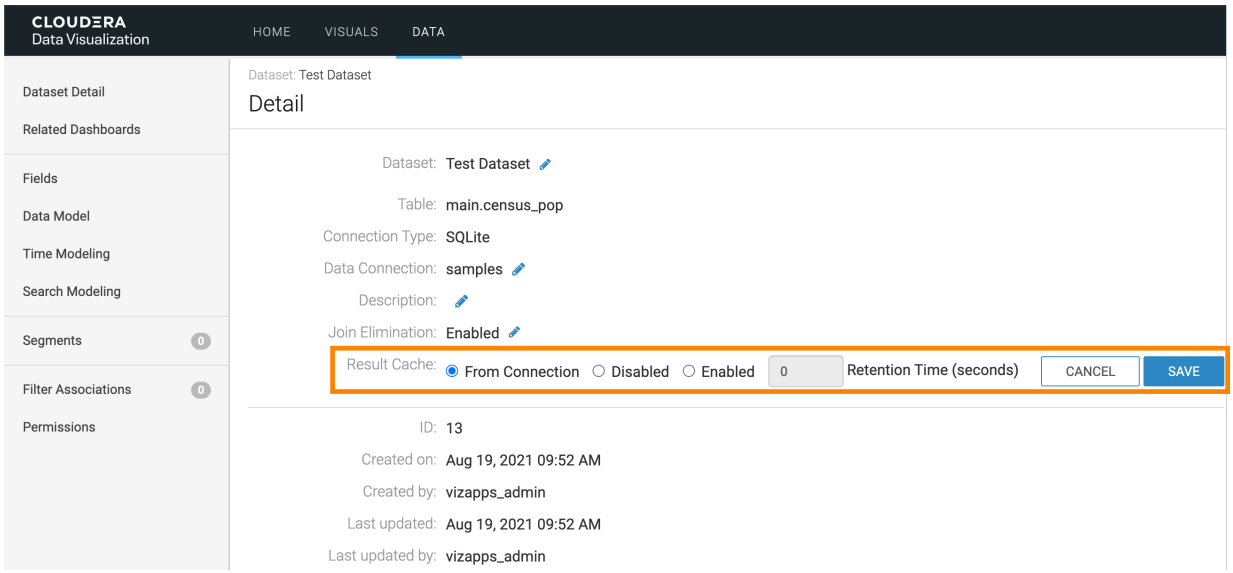

SQL for first table

For datasets created from a SQL query, you can modify the query at any time to adjust fields, update the WHERE clause, change the ORDER BY sequence, and more.

Procedure

1. Click the icon.
2. Make the desired changes to the SQL statement.
3. Click SAVE to apply the changes.

   To test your updates, click RUN before saving the updated SQL query.

Information on creation date and updates

The Dataset Detail interface provides information about the dataset's creation and its most recent updates.

ID

The unique identification number assigned to the dataset.

Created on

The exact date and time (in timestamp format) when the dataset was created.

Created by

The username of the user who initially created the dataset.

Last updated

The date and time (in timestamp format) of the most recent update made to the dataset.

Last updated by

The username of the user who last modified or updated the dataset.

Checking related dashboards

In Cloudera Data Visualization, you can easily determine which visuals use a particular dataset.

Procedure

1. On the main navigation bar, click Data.

   The Data view opens, displaying the Datasets tab.
2. Find the dataset that you want to examine in the list of datasets, either by scrolling or by using search.
3. Click the dataset.

   The dataset side navigation panel appears, open on the Dataset Detail view.
4. In the side navigation menu, click Related Dasboards.

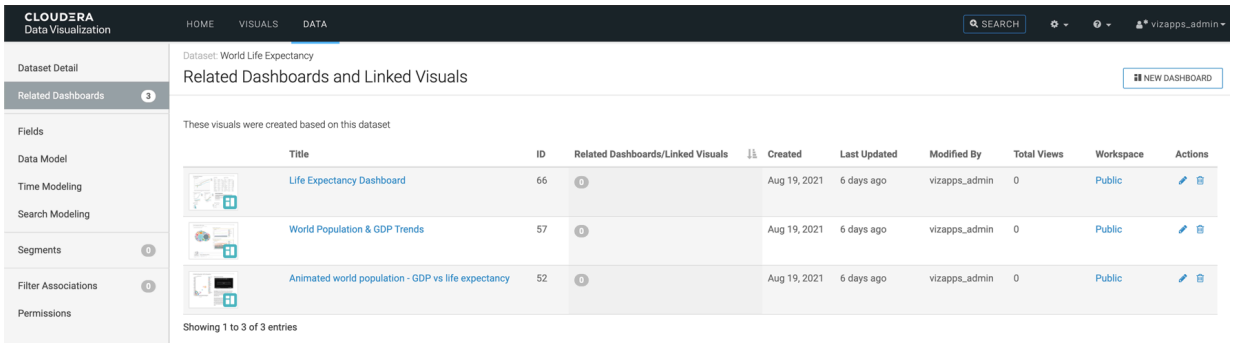

   The Related Dashboards view opens, showing a list of visuals that use this dataset.

   The following information is available for each app in this list, under Related Dashboards:

   - Type icon represents the style of the dashboard, or a snapshot icon of the visual (if this feature is on).
   - Title is the name of the visual.
   - ID
   - Related Dashboards/Linked Visuals
   - Created is the date when the visual was created.
   - Last Updated is the time interval after the last update of the visual. It is expressed in minutes, hours, days, or months, as appropriate.
   - Modified by is the name of the user who modified the app most recently.
   - Total Views is the number of times the app was viewed.
   - Workspace
   - Actions are the permissions available to you They may include the following:
     - Clicking Edit (pencil) icon edits the visual.
     - Clicking Delete (trash) icon deletes the visual.

Tracking dataset versions [Technical Preview]

Cloudera Data Visualization supports dataset versioning, an important feature for tracking changes and ensuring the integrity of your datasets over time.

About this task

Dataset versioning allows you to manage multiple versions of your datasets, which is particularly useful when dealing with frequent updates or iterative changes. Every time a dataset is modified, a new version is automatically created, allowing you to track the evolution of your data structure and content over time. If an error is introduced or an earlier version is preferred, dataset versioning allows you to revert to a previous version. This feature also supports a collaborative environment by ensuring that changes made by different team members are versioned, and previous states can be restored if necessary.

Note: Dataset versioning is currently available as a technical preview in Cloudera Data Visualization.

Before you begin

To use this feature, you must first enable dataset version control in the site settings. For instructions on how to enable and configure dataset version control, see Managing version control site settings.

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.
2. Find the dataset that you want to review, by scrolling through the list or using the search function.
3. Click the dataset name to open its details.

   The dataset side navigation pane opens for the selected dataset, displaying the Dataset Detail page.
4. Click Version Control in the side navigation to view version history.

   The Version Control page shows the details of the current dataset version along with any previous versions, if available.

   Current version: The active version of the dataset, which can either be the most recent saved version after modifications or a previous version that has been reinstated as the active version. This version reflects the dataset's state, including any changes or configurations applied up to the point of its selection. The first version of a dataset is always the active or current version, and it remains the current version, even if unnamed, until it is replaced by a new version.

   Previous versions: Each dataset modification (for example editing a dataset field, the data model or the time model, or adding new segments) triggers the creation of a new version of the dataset. A snapshot of the dataset’s state prior to the change is saved as a previous version, and the new modification becomes the current version.

   Note: Modifying filter associations or changing dataset permissions does not create a new dataset version.

   Named and unnamed versions: By default, new dataset versions are assigned a timestamp as their name and are considered unnamed versions. To retain any version, you must assign it a name. Unnamed versions are deleted based on the settings configured in the version control site settings. You can rename a dataset version by clicking the edit icon next to its name.
5. To change the current dataset version, identify the version on the Version Control page that you want to reinstate as the current version and click Change Version.

   The chosen version will now become the active version, replacing the previous current version.

Changing dataset fields

It is easy to make changes to the fields of a dataset in Cloudera Data Visualization.

Editing dataset fields

Editing dataset fields allows you to tailor your data to better suit specific analytical needs and visualization requirements. By modifying field parameters, you can enhance the accuracy, clarity, and relevance of your datasets.

Procedure

1. Navigate to the Data view by clicking DATA on the main navigation bar.

   The Data view opens, displaying the Datasets tab.
2. Select a connection from the left navigation menu.
3. In the Datasets area, select the dataset you want to modify.

4. In the Dataset Detail menu, select Fields.

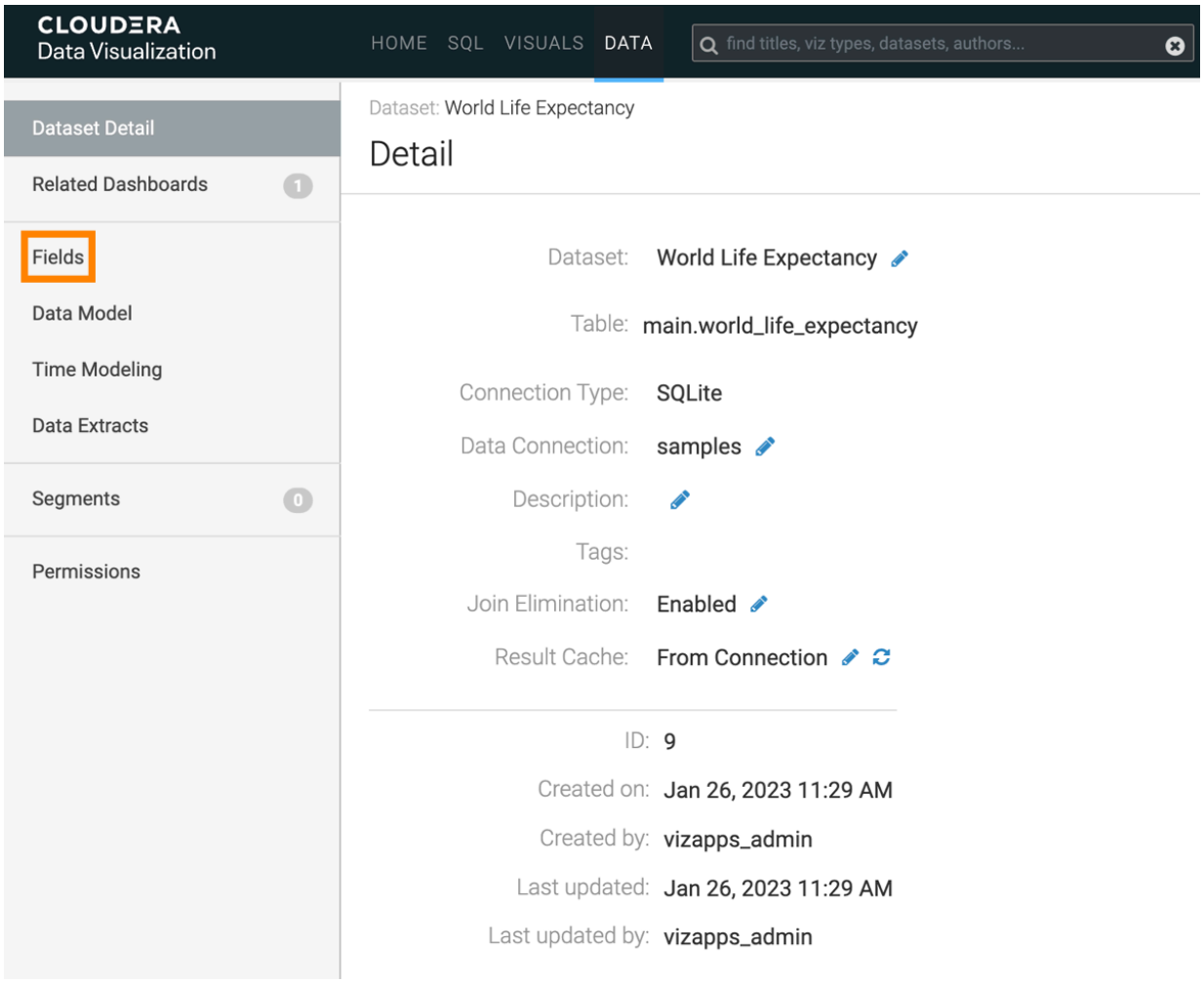

5. Click EDIT FIELDS to access the editing interface.

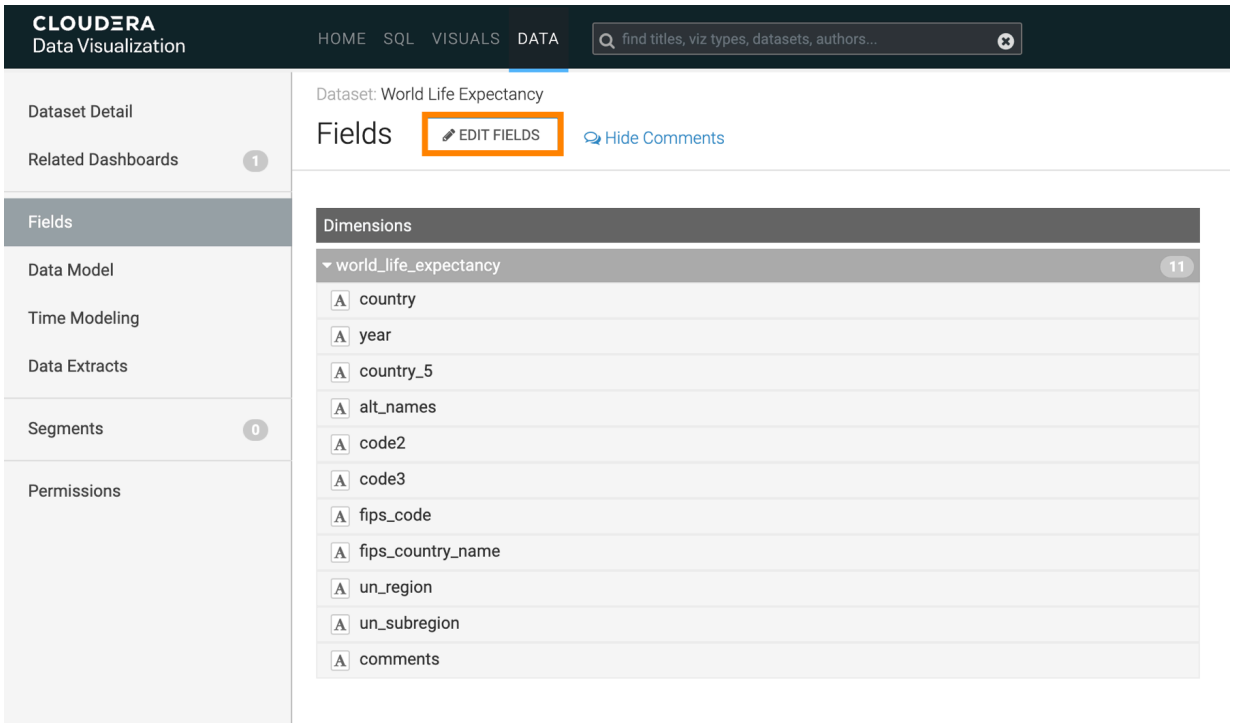

This action adds an edit button (pencil) to each dataset field row.

6. Click the edit button for the specific field you want to update.

   The Edit Field Parameters modal window opens.

7. Make your changes as needed.
   - You can adjust basic field settings such as name, default aggregation, or geo type, and add comments to the field.
   - You can mark a field as 'Sensitive' to restrict its usage in specific visual shelves that are designated as 'Sensitive'. For example the Embedding Context shelf of the Artificial Intelligence (AI) Assistant visual is a sensitive shelf.

     When you mark a field as 'Sensitive', the icon is displayed next to it.
   - You can switch to the Display Format tab to assign a category to the field or to the Color tab to enable custom colors for the field.

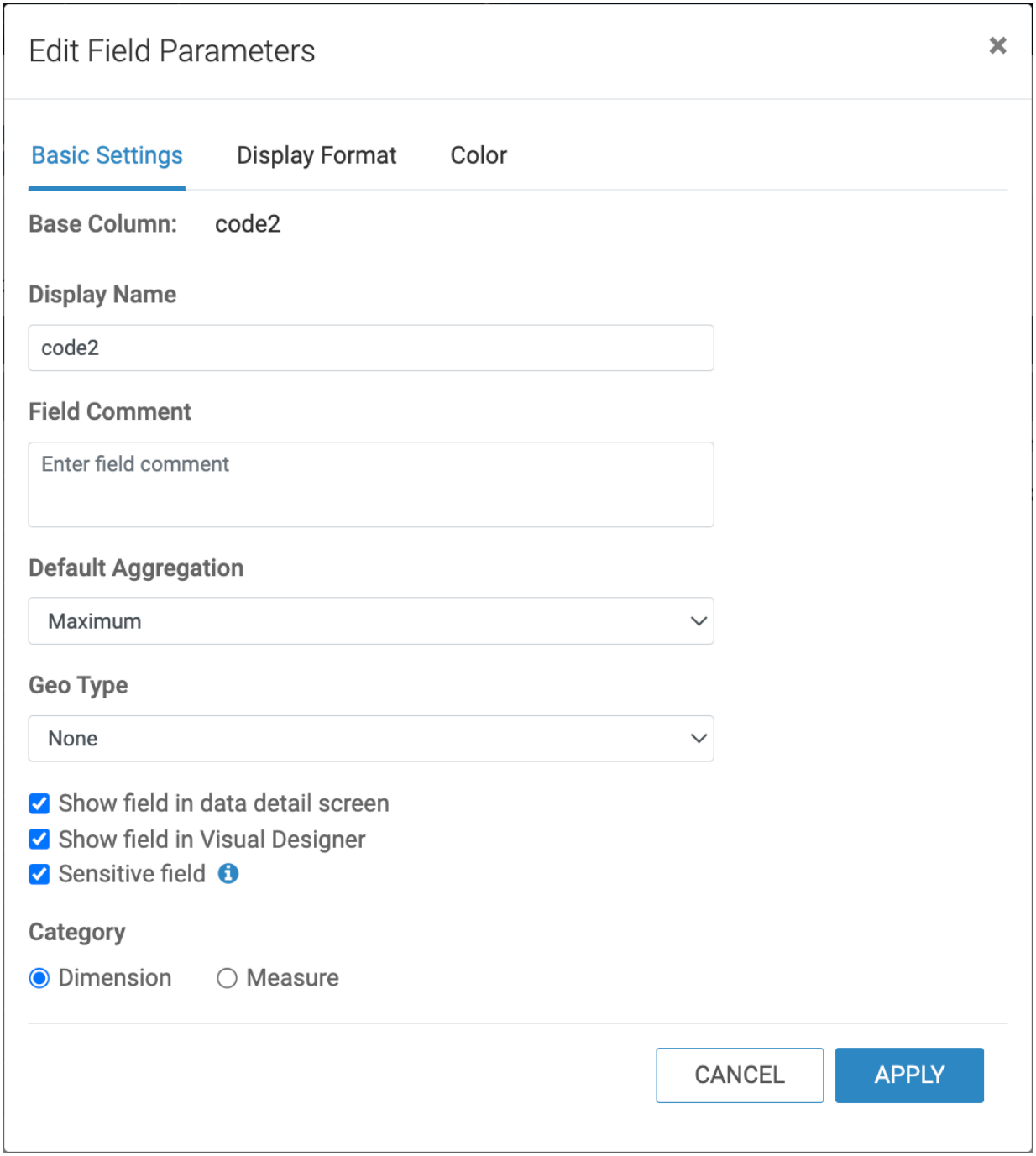

8. Click APPLY.

Hiding dataset fields from applications

You may find it useful to hide dataset fields that are not typically used for visualizations to prevent unintended bias in BI and analytics, or even to obscure confidential data. In Cloudera Data Visualization, you can do this by turning off the default visibility option of a particular dataset field.

About this task

The following steps demonstrate how to prevent data fields from appearing in visualizations and applications of dataset World Life Expectancy [data source samples.world_life_expectancy]. The fields comments, lat, and lng are empty, so they are good candidates for this operation.

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.
2. In the left navigation menu, click Samples.
3. In the Datasets area, select World Life Expectancy (main.world_life_expectancy).
4. In the Dataset Detail menu, select Fields.

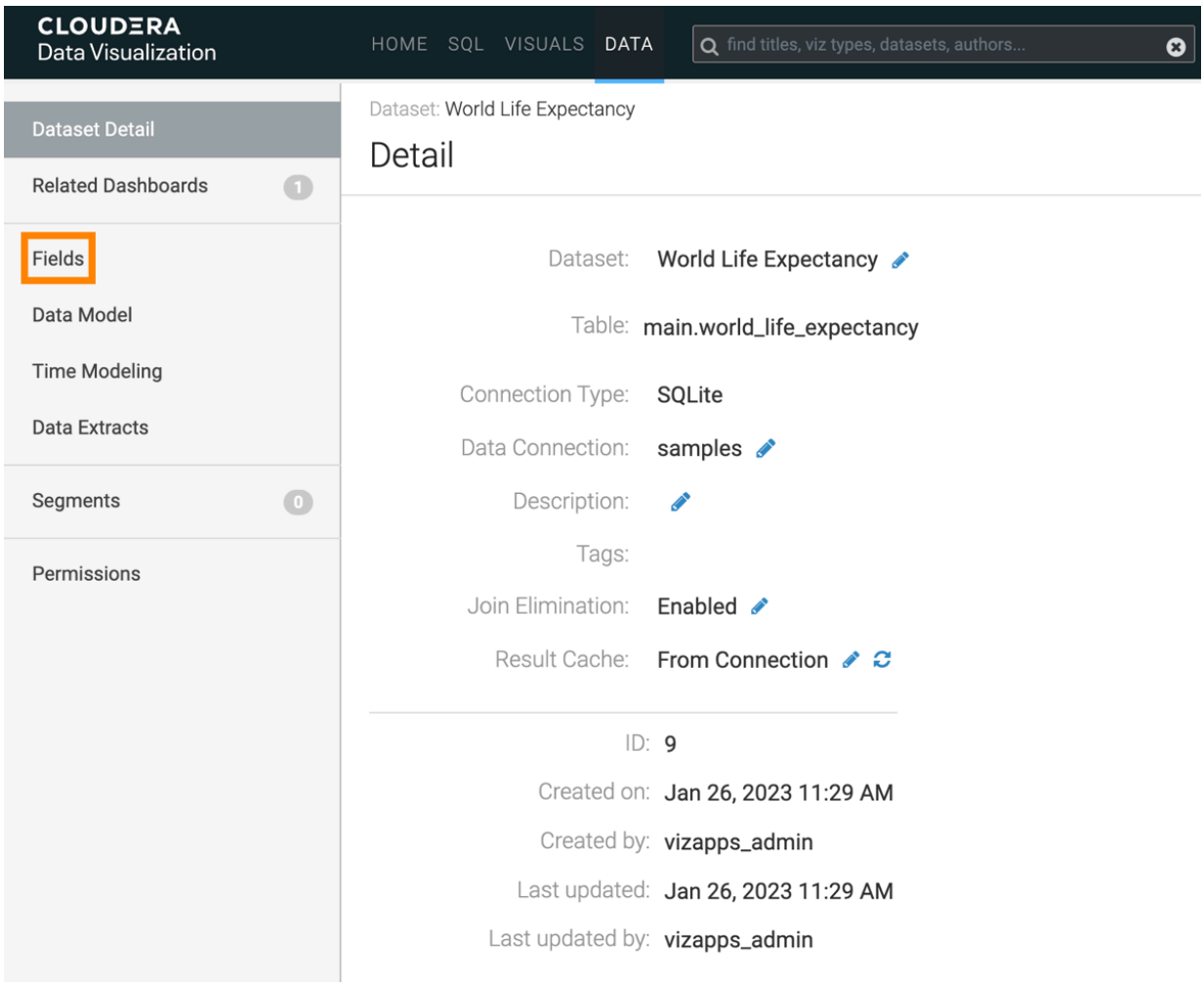

5. In the Fields interface, select EDIT FIELDS.

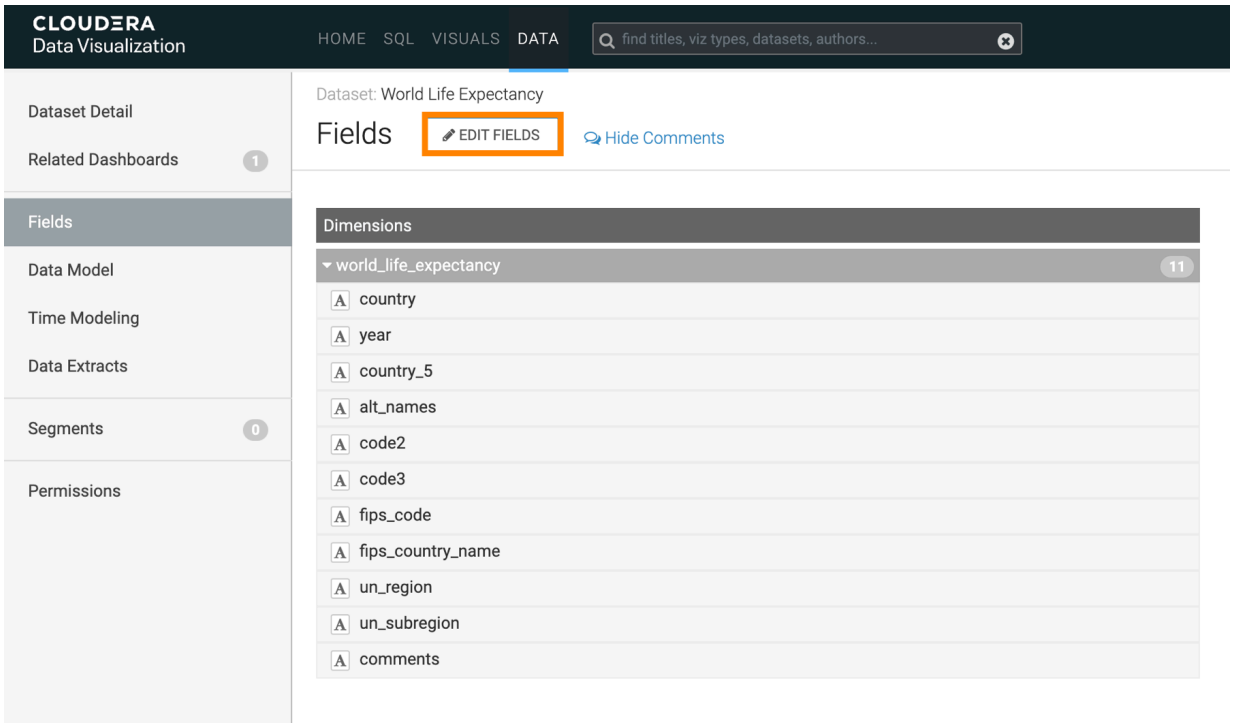

6. Under Dimensions, find the field comments.

7. Click the (eye) icon on the comments line.

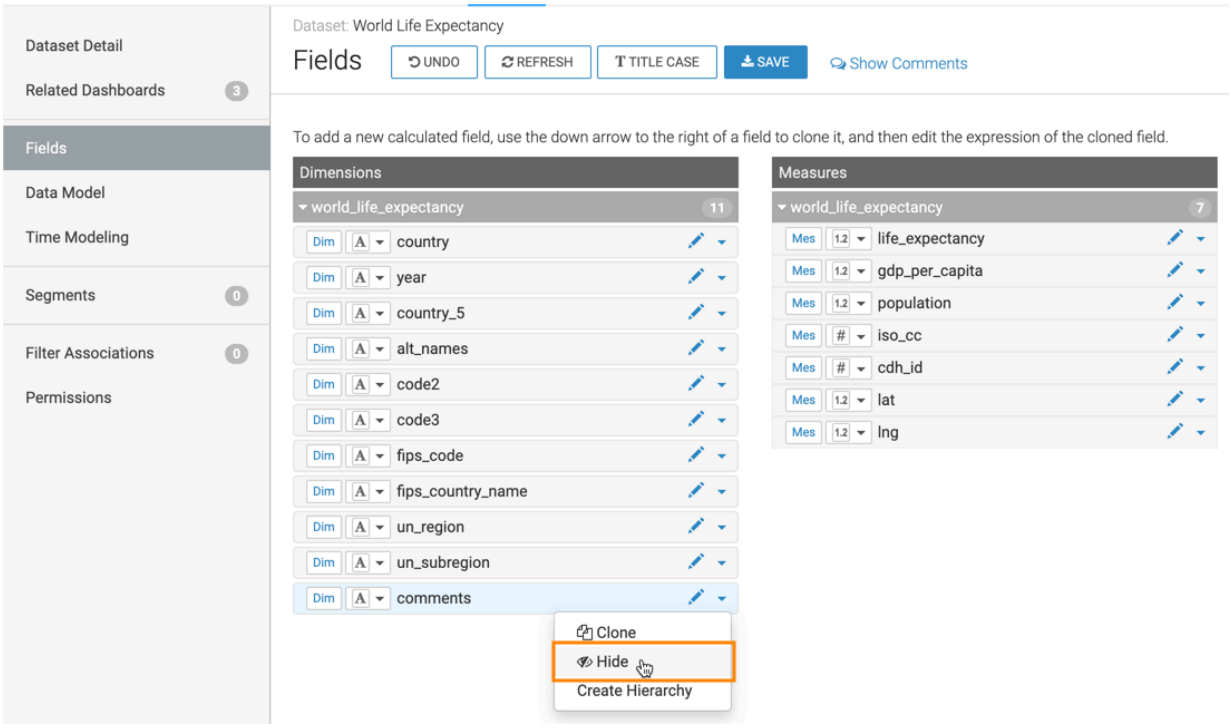

The icon next to the comment field changes to (slashed eye).

8. Under Measures, find the fields lat and lng, and hide them.

9. Click Save.

Results

In the updated Fields interface, Dimensions table has a total of 11 fields and Measures table lists a total of 7 fields, as before.

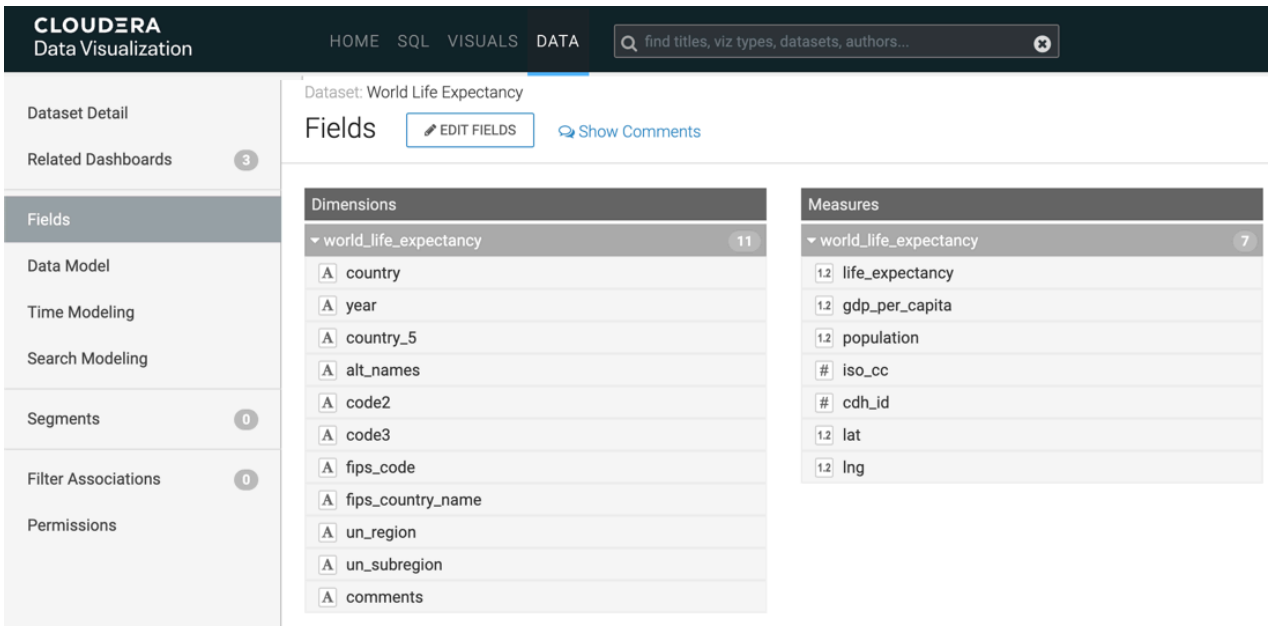

- The number of Dimensions is 11. This is calculated as All Dimensions (11) - Hidden Dimensions (1) + Segment (1).
- The number of Measures is 6. This is calculated as All Measures (7) - Hidden Measures (2) + Record Count (1).

However, when using Visual Designer, the hidden fields do not show.

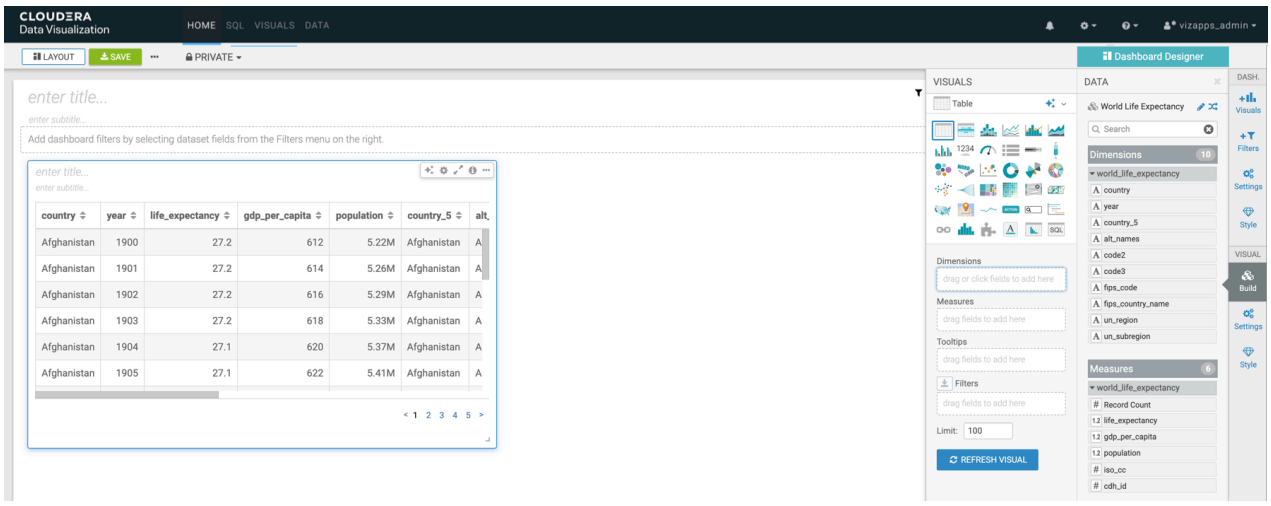

Tip:

The field visibility can be changed in the Edit Field Parameters window modal.

Changing field aggregation

You can change the basic fields defaults in your dataset.

About this task

The following steps demonstrate how to change the default aggregation function from Sum to Average for the field life_expectancy in the dataset World Life Expectancy [data source: main.world_life_expectancy].

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.
2. In the left navigation menu, click samples.
3. In the Datasets area, select World Life Expectancy (samples.world_life_expectancy).

4. In the Dataset Detail menu, select Fields.

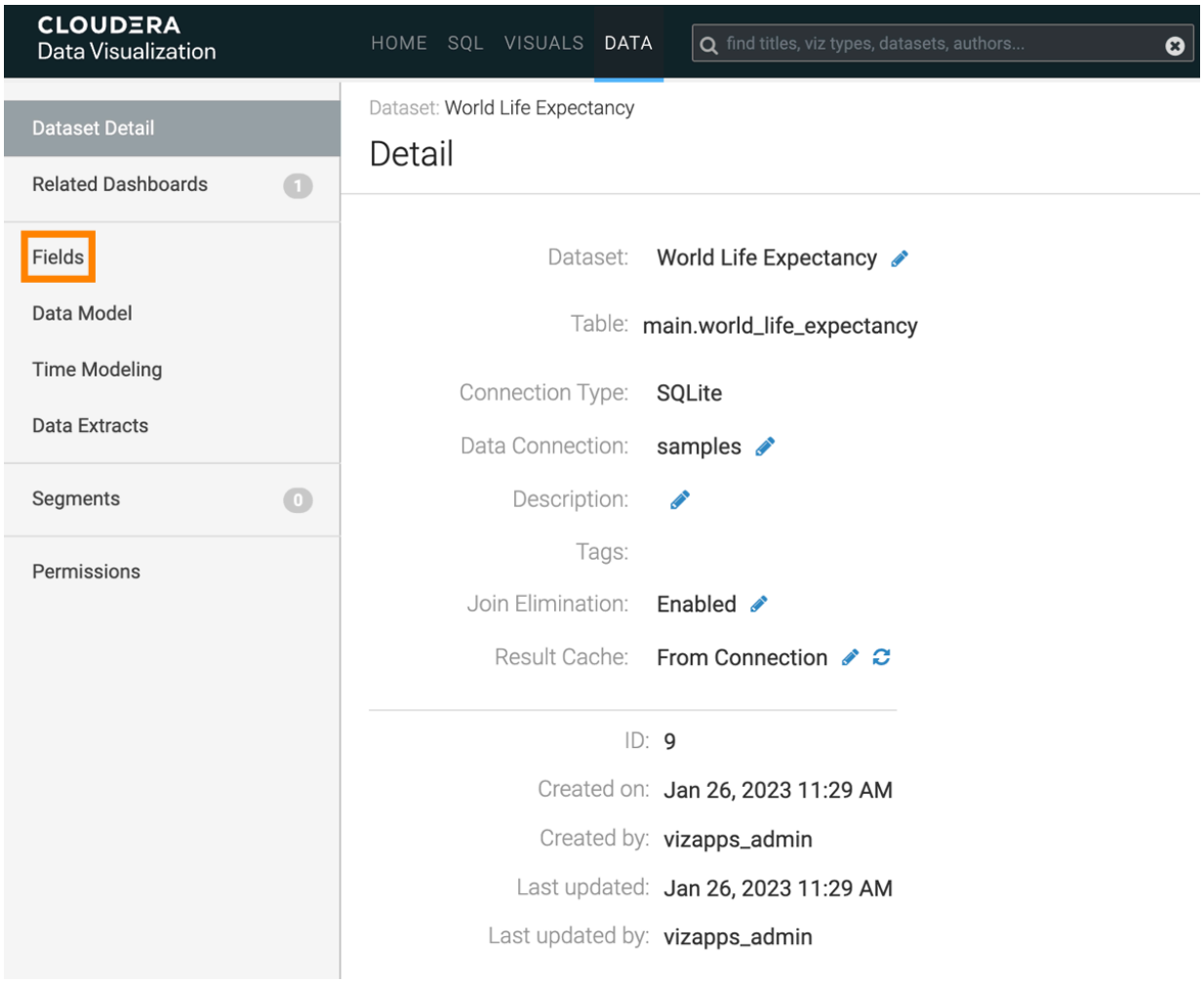

5. In the Fields interface, select EDIT FIELDS.

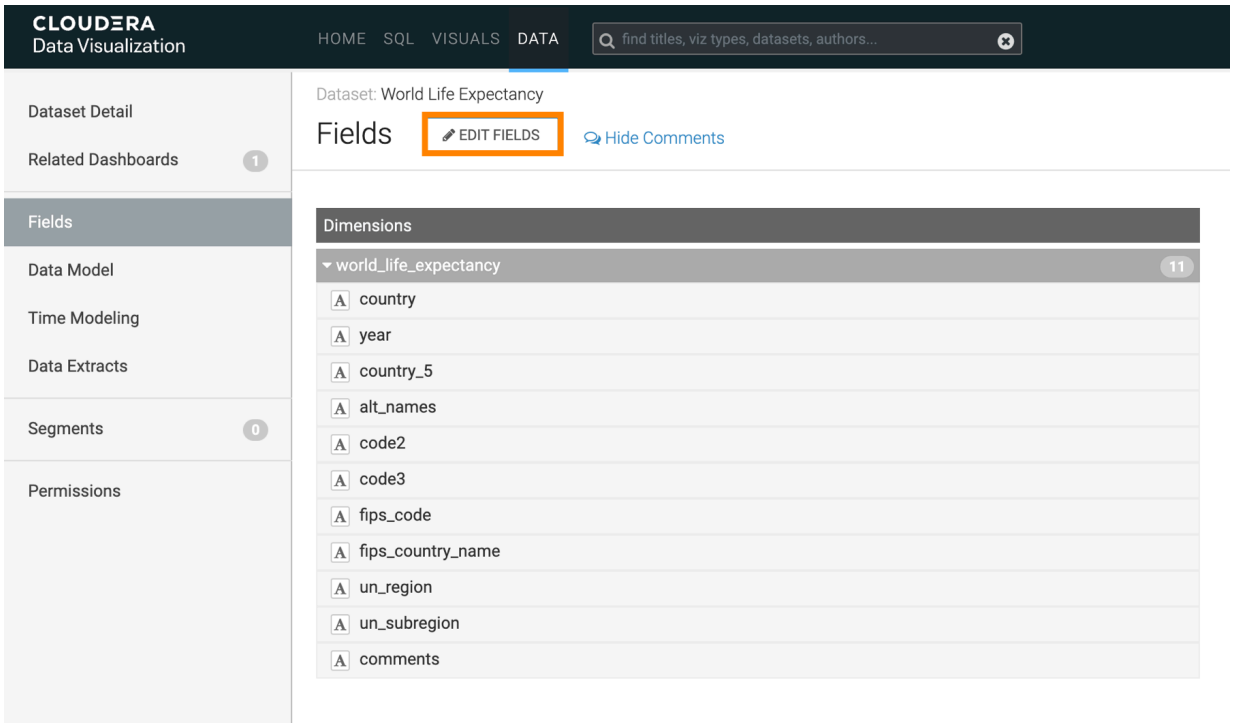

6. Under Measures, find the field life_expectancy and click the (down arrow) icon on its right side.
7. Click the Edit (pencil) icon.

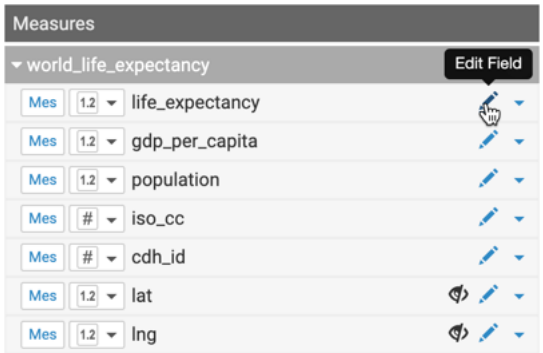

The Edit Field Parameters window modal opens.

8. Change Default Aggregation from Sum to Average, and click APPLY.

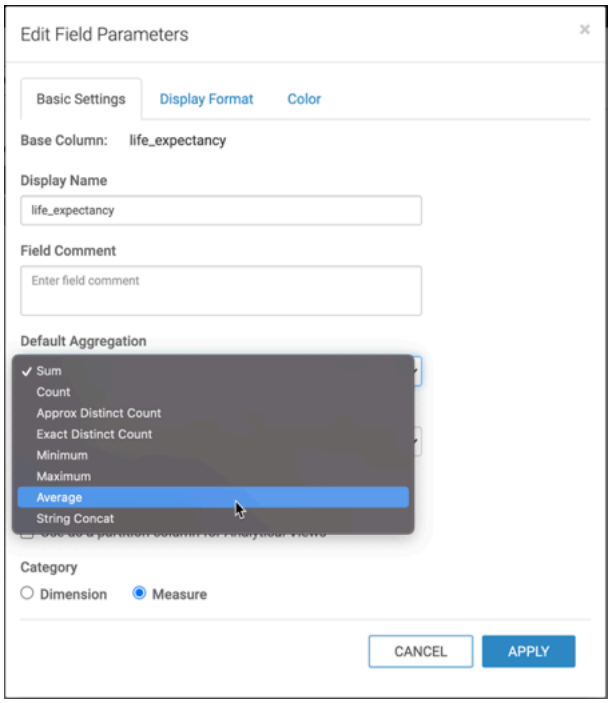

9. Under Dataset: World Life Expectancy, click SAVE.

Results

As a result of this change, all new visuals created from this dataset will automatically use the new aggregation.

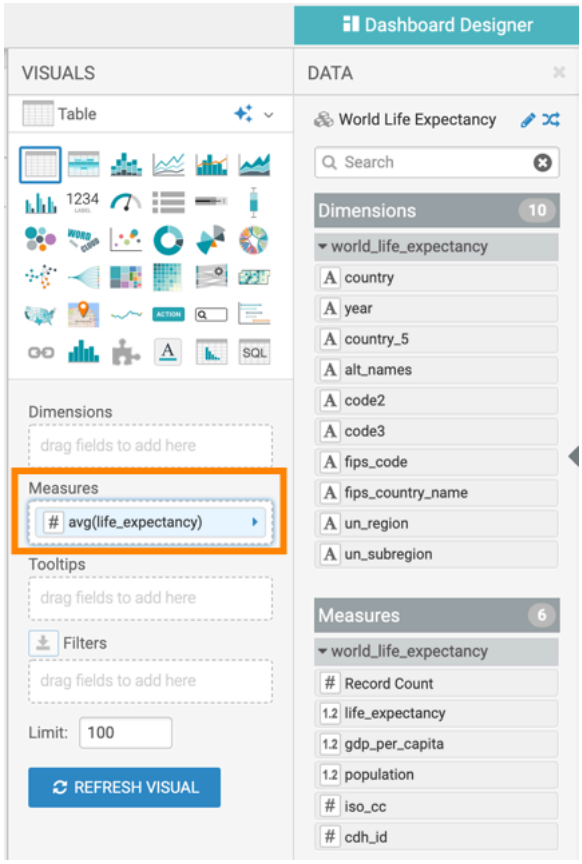

Creating calculated fields

In Cloudera Data Visualization, you can easily create a new calculated field in the dataset, and subsequently use it in dashboards and visuals.

About this task

Sometimes the data in the base tables cannot be used directly, and you must use an expression to change or "correct" it. For other use cases, you can create a calculation based on one or more fields. Instead of adding these expression for every visual, you can create a new calculated field in the dataset.

The following steps demonstrate how to create a new field gdp (gross domenstic product) in the dataset World Life Expectancy [data source main.world_life_expectancy]. We define it by the following equation:

```
gdp = gdp_per_capita x population
```

Cloudera Data Visualization supports three primary methods of editing fields at the dataset level: Basic, Expression, and Display Format. In this example, we make changes both on the Basic and Expression tabs. For innformationn on how to use Display Format options, see *Changing the Field Display Format.*.

Procedure

1. On the main navigation bar, click DATA.
2. In the left navigation menu, click samples.
3. In the Datasets area, select World Life Expectancy (main.world_life_expectancy).

4. In the Dataset Detail menu, select Fields.

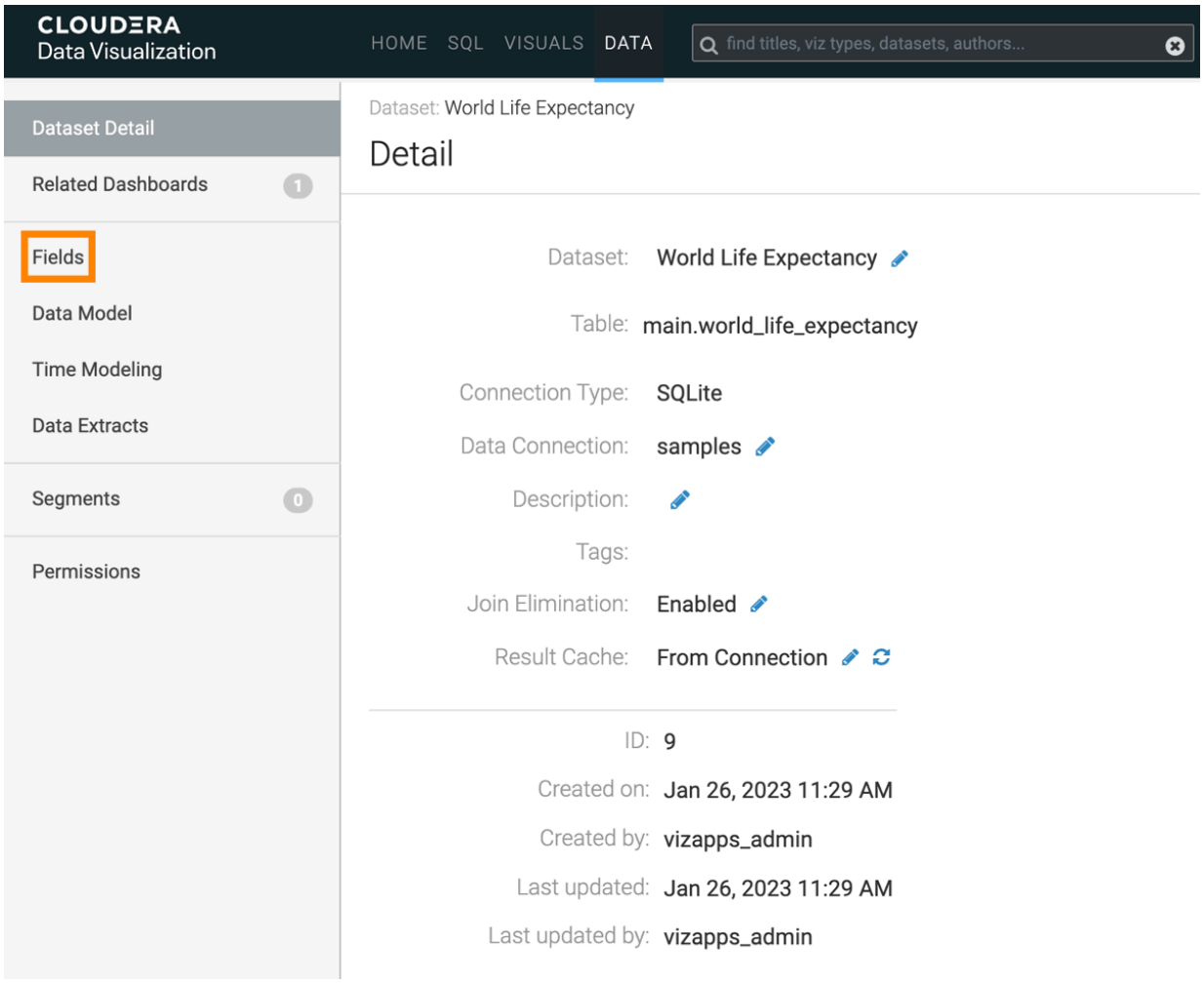

5. In the Fields interface, select EDIT FIELDS.

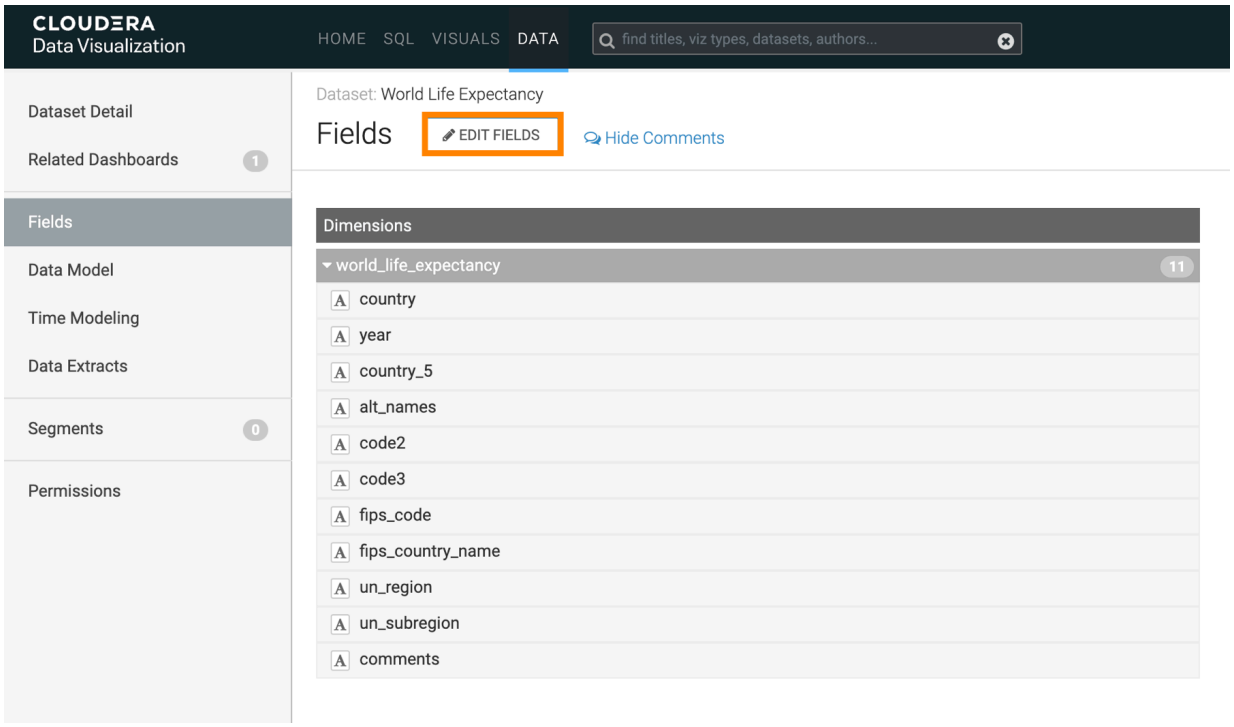

6. Under Measures, find the field gdp_per_capita, and click the (down arrow) icon on its right side.
7. From the menu, select Clone.

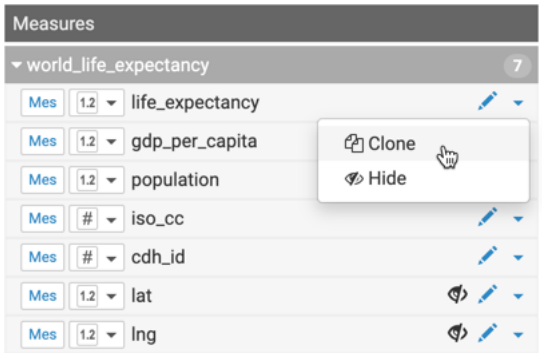

8. Under Measures, find the new cloned field Copy of gdp_per_capita, click the Edit (pencil) icon on its right side.

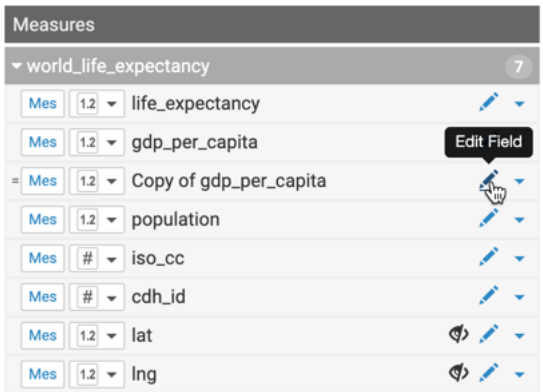

The Edit Field Parameters modal window opens, which supports three primary methods of editing fields. They match the three tabs of the modal: Basic, Expression, and Display Format.

9. Make the following changes on the Basic tab:

a) Change Display Name to gdp.

b) Add Field Comment gdp_per_capita * population.

c) Ensure that the Default Aggregation is Sum.

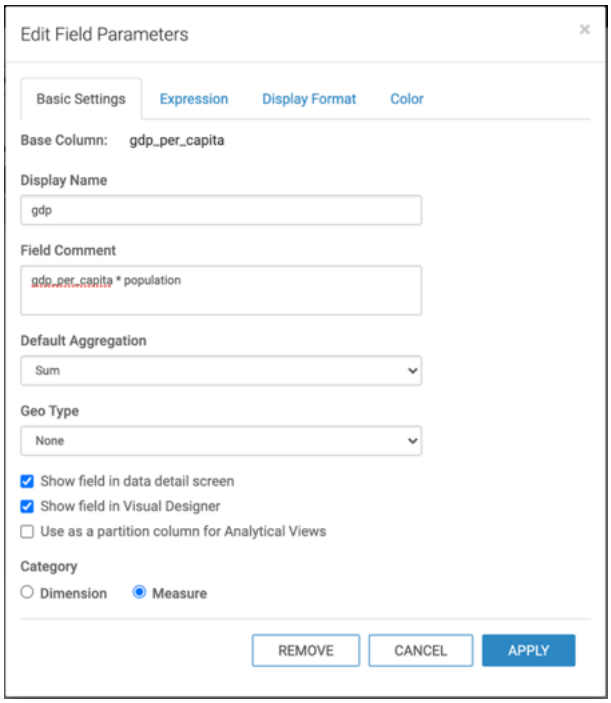

10. Click the Expression tab and make the following changes:

a) Change Expression to [gdp_per_capita] * [population].

b) Click VALIDATE EXPRESSION to ensure that the calculation works.

c) When the Validation Successful message appears on the modal, click APPLY.

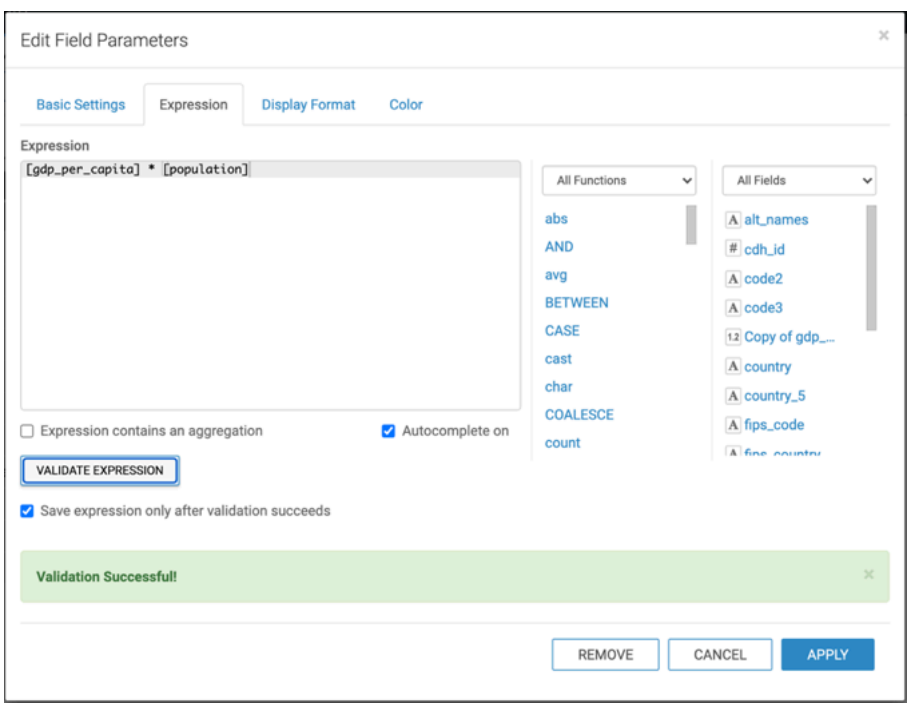

The new calculated field has an equal sign (=) notation.

11. Under Dataset: World Life Expectancy, click SAVE.

Testing calculated fields

In Cloudera Data Visualization, you can easily test whether a newly calculated field works correctly or not.

Procedure

1. Click New Dashboard in the top right corner of this interface.
2. Select the Combined Bar/Line visual type.

3. Populate the shelves from the available fields in the following way:
   a) Under Dimensions, select country, and place it on the Dimensions shelf.
   b) Under Dimensions, select year, and place it on the Filters shelf.
   c) In the Filter for year modal window, under the Value tab, select the year 2010.
   d) Under Dimensions, select un_region, and place it on the Filters shelf.
   e) In the Filter for un_region modal window, under the Value tab, select Europe.
   f) Under Measures, select gdp_per_capita, and place it on the Bar Measure shelf.
   g) Under Measures, select gdp, and place it on the Line Measure shelf.
   h) Click the arrow on gdp field.
   i) In the Field Properties under Axis, select Secondary Axis.

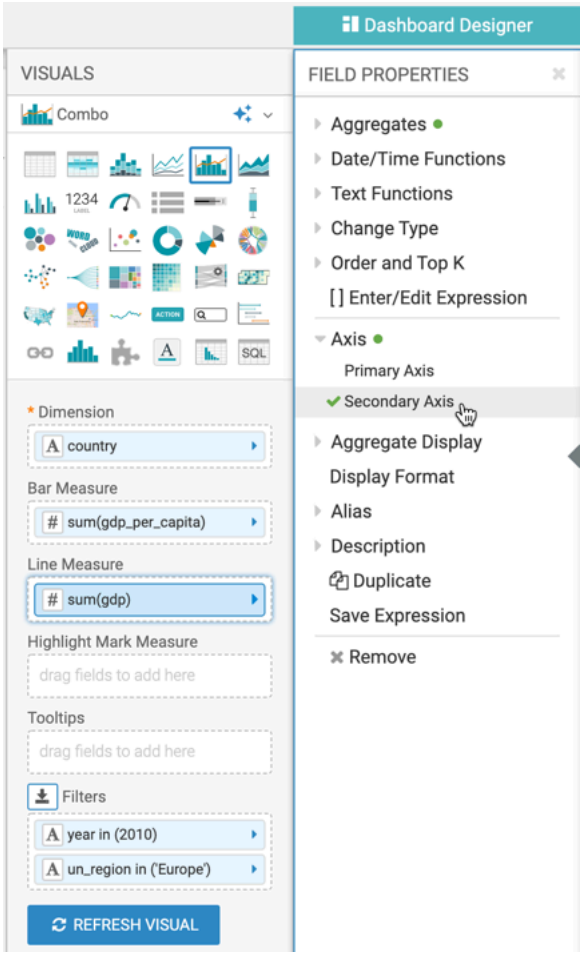

4. Click REFRESH VISUAL.

Results

The two measurements appear on the graph, superimposed on each other: the original gdp_per_capita represented by the bars, and the calculated gdp, represented by the line.

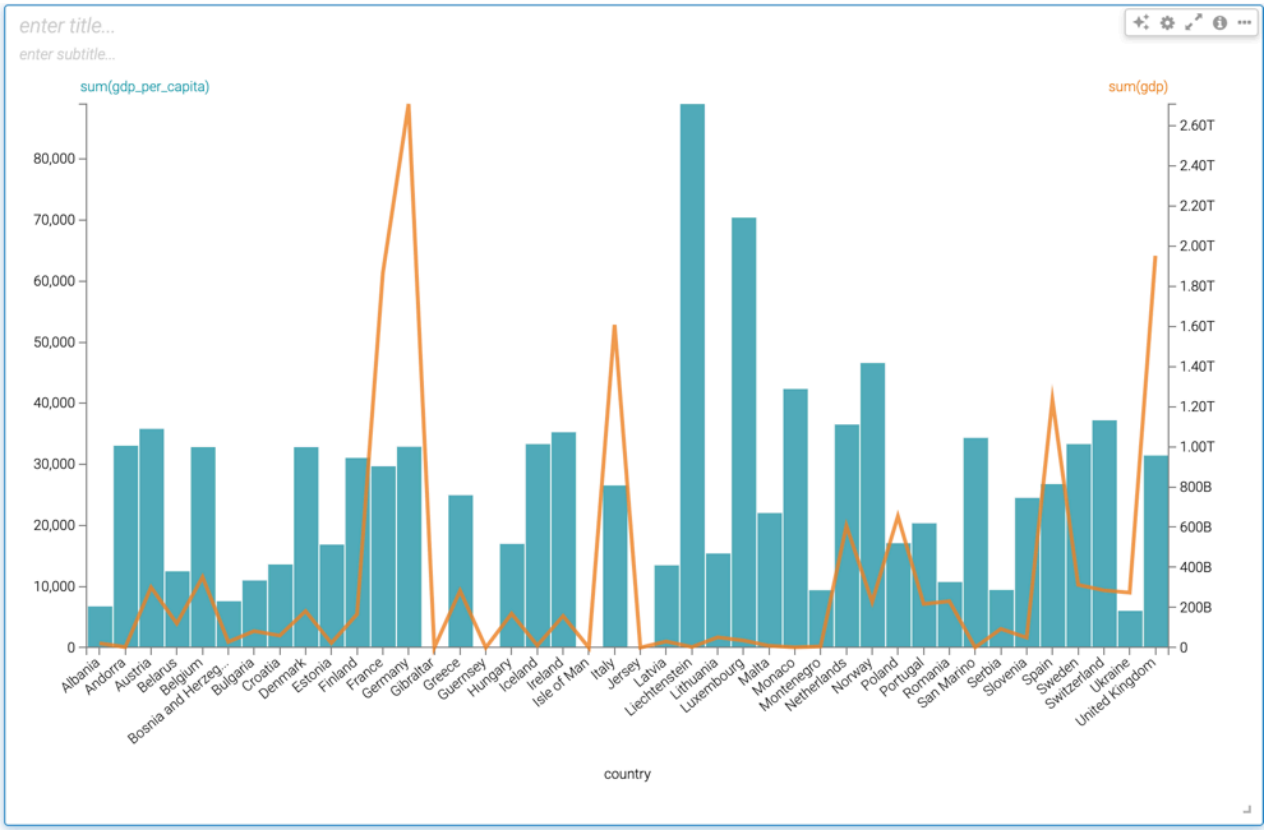

Changing the field display format

Cloudera Data Visualization lets you configure the display format of each field at the dataset level.

Procedure

1. In the World Life Expectancy dataset, click EDIT FIELDS.

2. Under Measures, find the field gdp_per_capita and click the (pencil) icon on its right side.

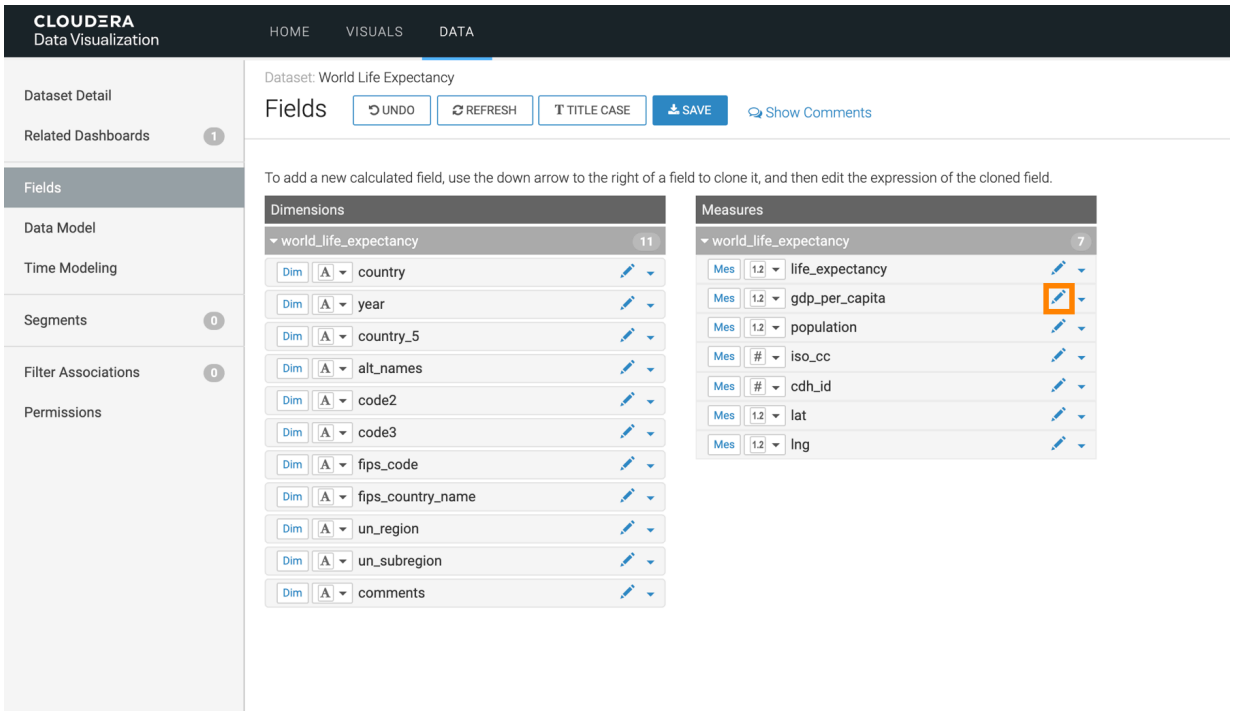

The Edit Field Parameters modal window opens.

3. Click the Display Format tab.

Under Display Format, you have several options in the Category menu:

- None
- Real Number
- Integer
- Percentage
- Scientific
- Currency
- Date/Time
- Custom Format
- Custom Javascript

For more information, see:

Changing currency field display format

In Cloudera Data Visualization, you can set currency display options for numerical fields across all visuals of a dataset.

About this task

Follow these steps to continue configuring a field at the dataset level for currency format. See *Changing the field display format* for the initial navigation steps.

Procedure

1. In the Edit Field Parameters modal window, under the Display Format tab, select Currency from the Category menu.

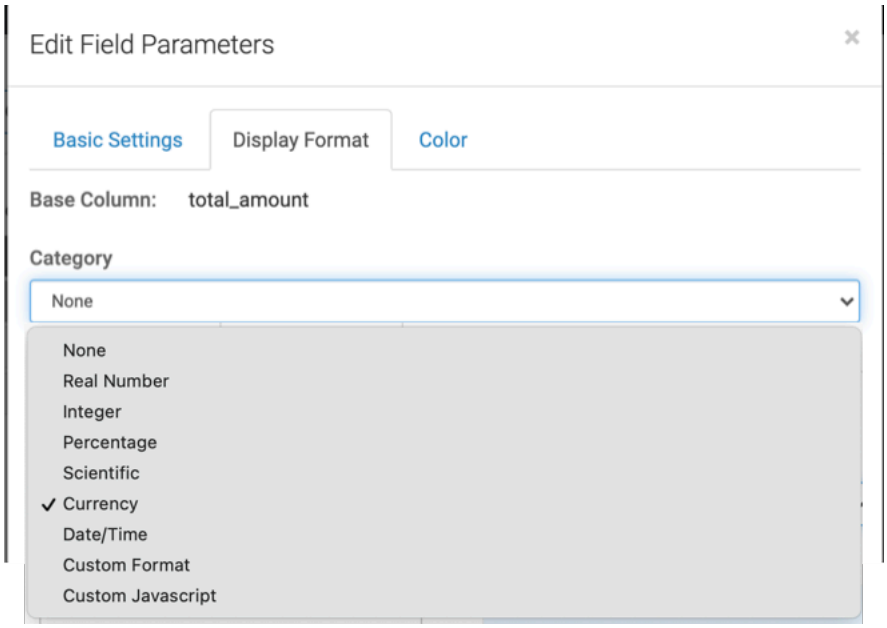

2. In the Currency Symbols menu, select the appropriate currency symbol: $ (Dollar), £ (Pound), ¥ (Yen/Yuan), # (Rupee), € (Euro), or # (Cedi).

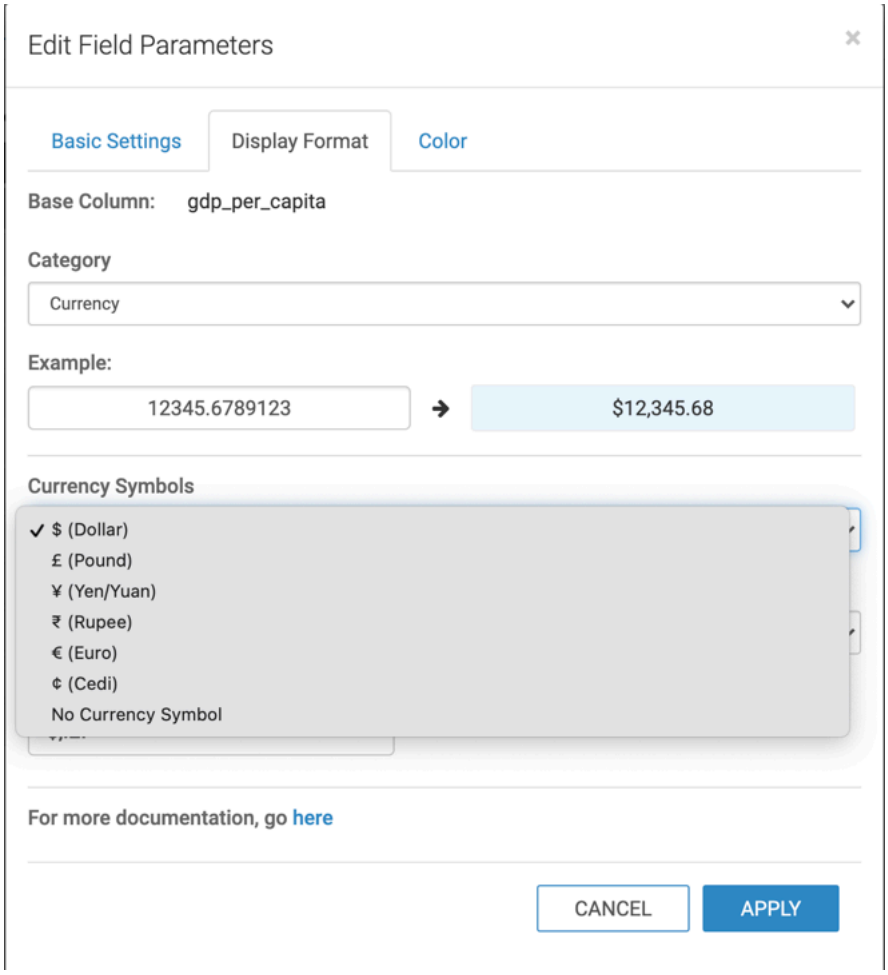

3. Select the Basic Format for your records.

   You can also define and apply a custom format. Enter a valid format mask in the Customize Currency text box. For a full list of options, see *Display Format Reference*.

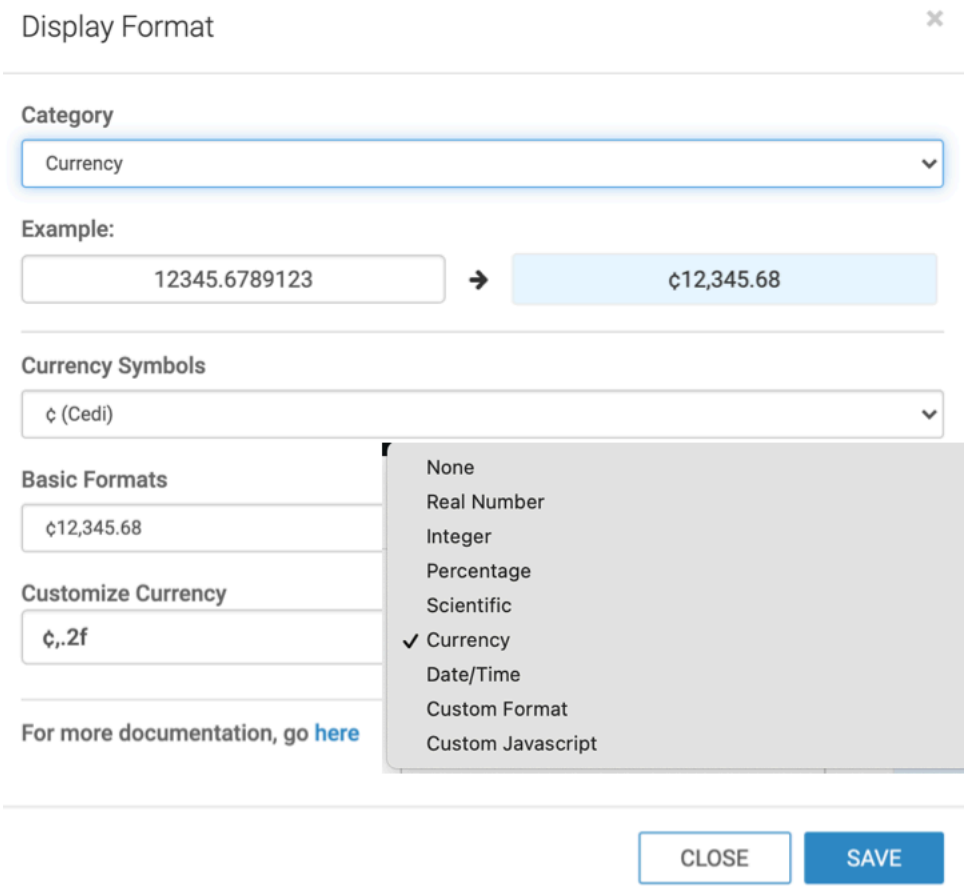

4. Click APPLY.
5. Click SAVE to save the changes to the dataset.
6. To verify that the format applies to all new visuals that use the field, create a new visual by repeating the steps in *Testing the Calculated Field*.

Results

The new visual displays the vertical axis numbers in currency format.

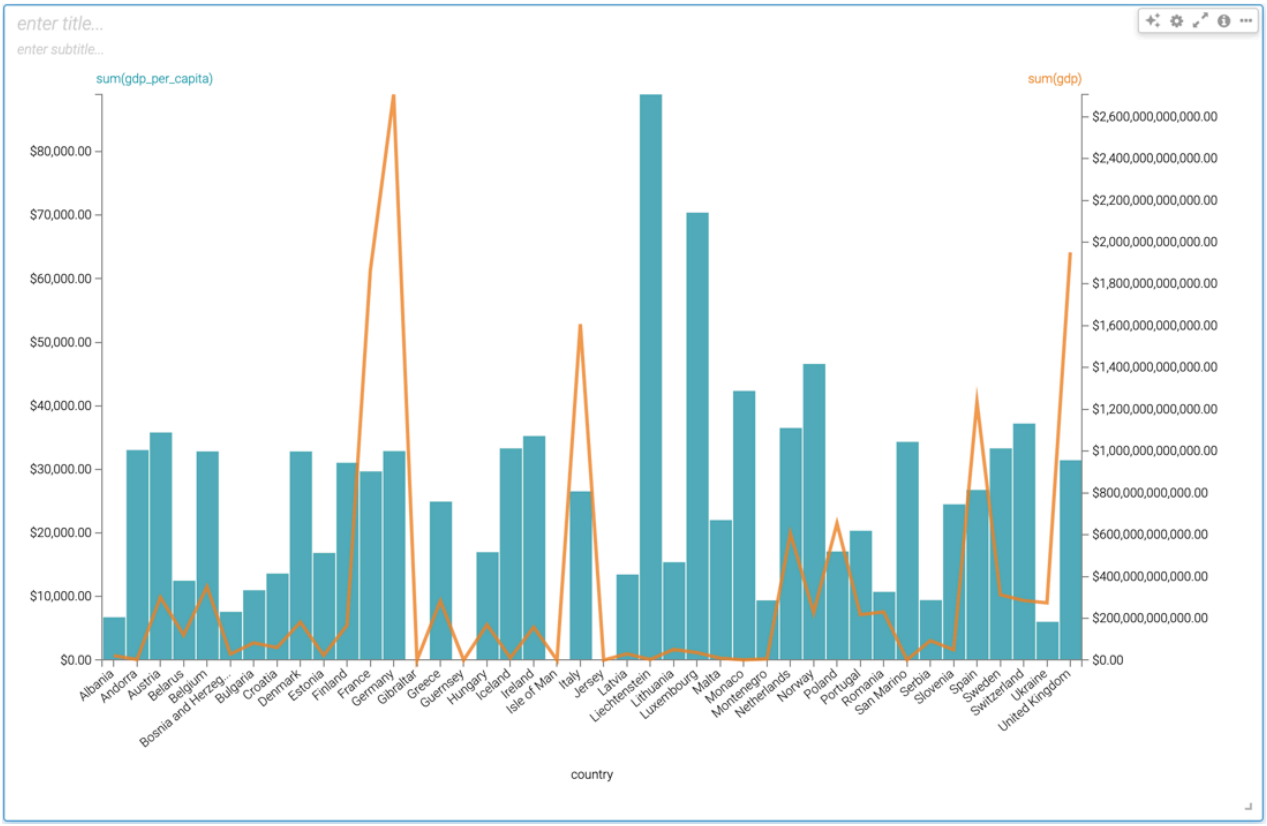

Related Information

Changing the field display format

Display format reference

Changing custom field display format

In Cloudera Data Visualization, you can Set currency display options for numerical fields across all visuals of a dataset.

About this task

Follow these steps to continue configuring a field at the dataset level for custom format. See *Changing the field display format* for the initial navigation steps.

Procedure

1. In the Edit Field Parameters window modal, under the Display Format tab, select Custom Format from the Category menu.

2. Under Customize Format, enter $S.
   - $ appends the dollar currency symbol to the left of the number.
   - S simplifies the number by minimizing the significant numbers (which on the axes appear with a large number of trailing zeros) and appending the appropriate non-scientific (currency) suffix to the right of the number.

   Note:

   Display Format Examples demonstrate some of the available options for defining custom format.

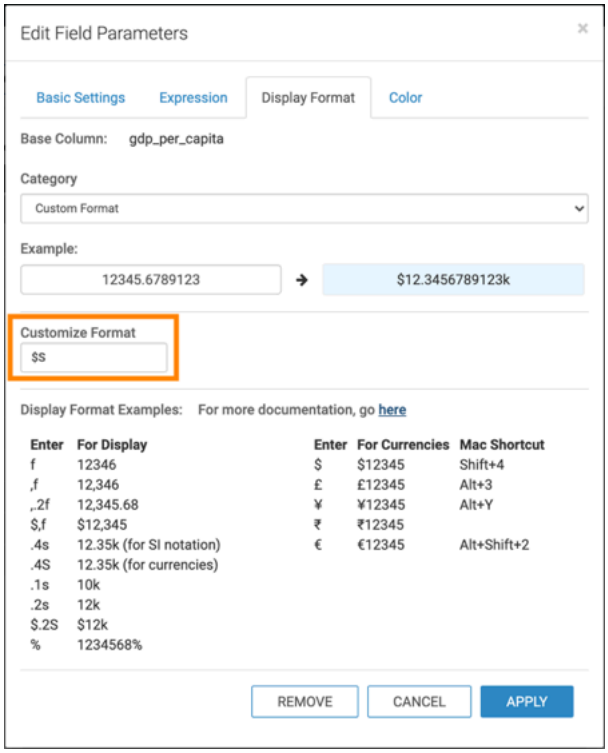

3. Click APPLY.
4. Save the changes to the dataset.
5. [Optional] To verify that the format applies to all new visuals that use the field, create a new visual by repeating the steps in *Testing the calculated field.*.

Results

When we use the $S custom format on the gdp field, the visual still uses the dollar currency sign, but also simplifies/abbreviates the number and shows the corresponding currency suffix. In the case of this visual, T for trillion; this custom format transforms the representation $2,500,000,000,000.00 into $2.5T.

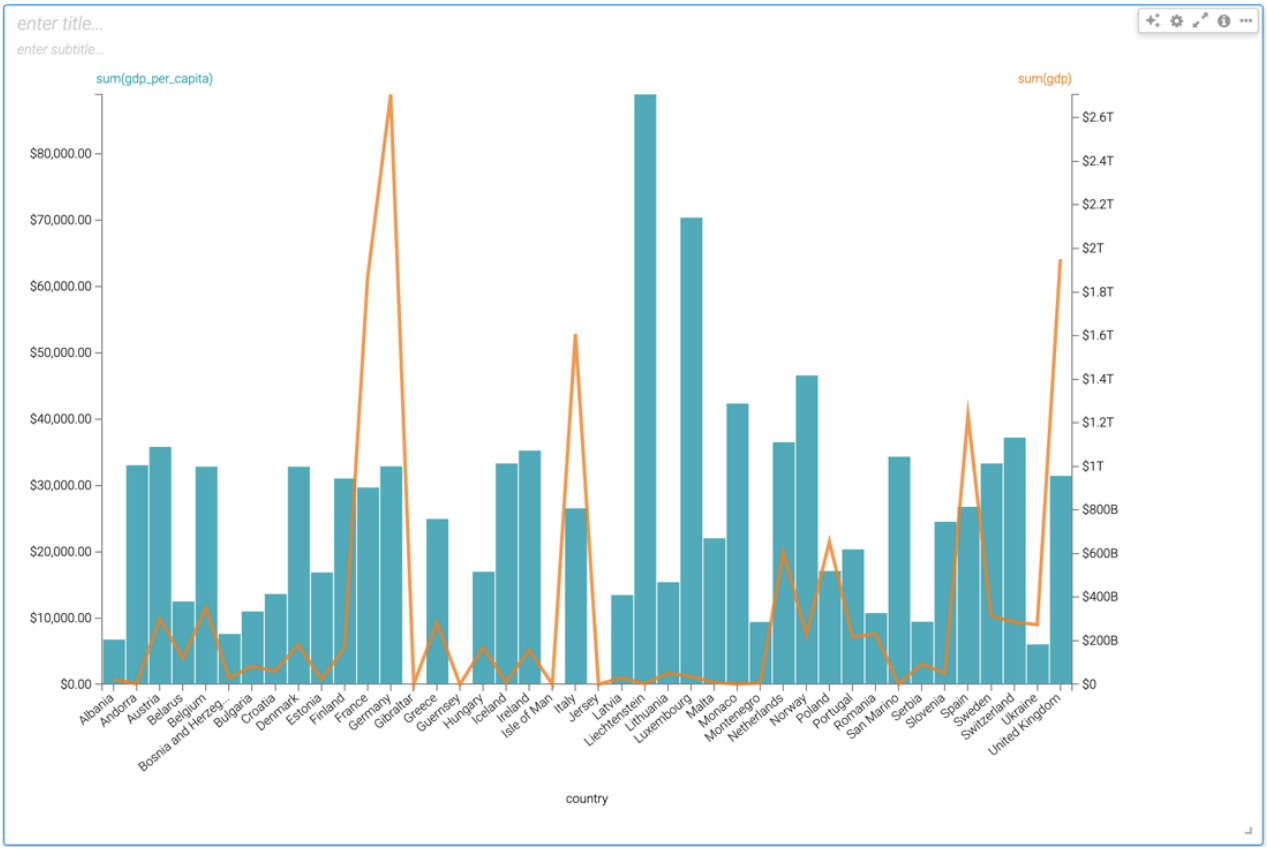

Related Information

Changing the field display format

Changing custom Javascript field display format

About this task

Follow these steps to continue configuring a field at the dataset level using Javascript (js). See *Changing the field display format* for initial navigation steps.

Procedure

1. In the Edit Field Parameters window modal, under the Display Format tab, select Custom Javascript from the Category menu.
2. Under Custom JS Format Function, enter the following js code:

```
function myFunc(value) {
  // Show the number in trillions with a dollar sign return
  '$${value/1000000000000}';
```

```
}
```

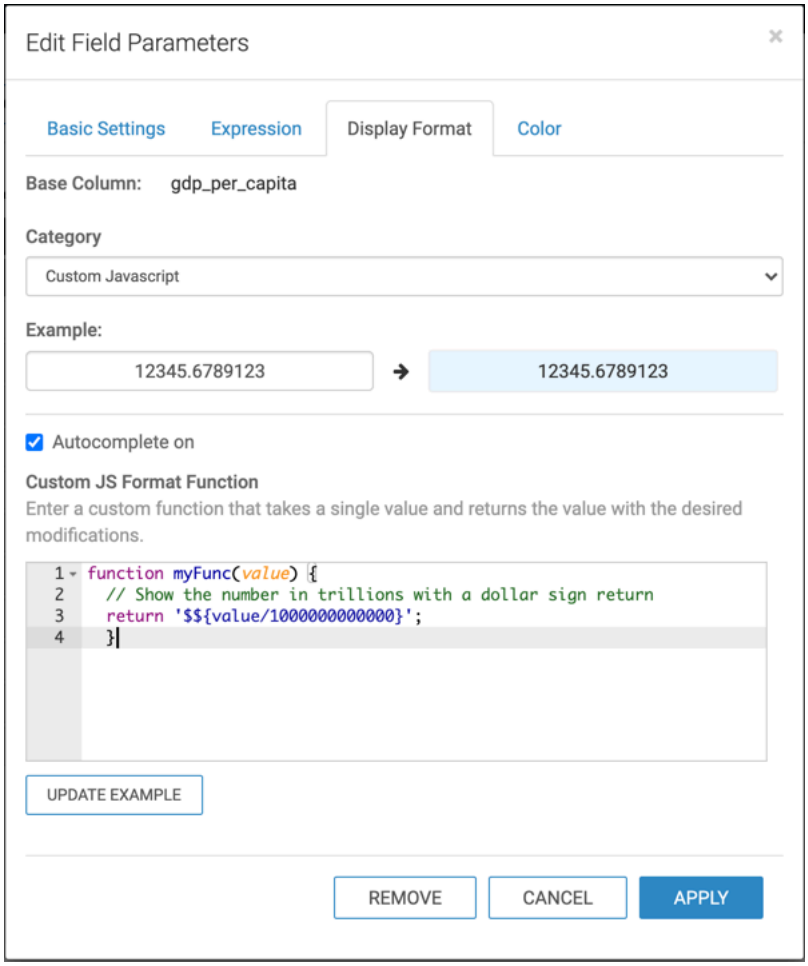

3. Click APPLY.
4. Save the changes to the dataset.

   In this example, under Dataset: World Life Expectancy, click Save.
5. To verify that the format applies to all new visuals that use the field, repeat the steps in *Testing the calculated field.*.

Results

Let's compare the visuals before and after we apply the js format on the gdp field.

- Before we apply the js format, the visual displays the currency with many trailing zeros.
- After applying the custom js format, notice that the trailing zeros no longer appear.

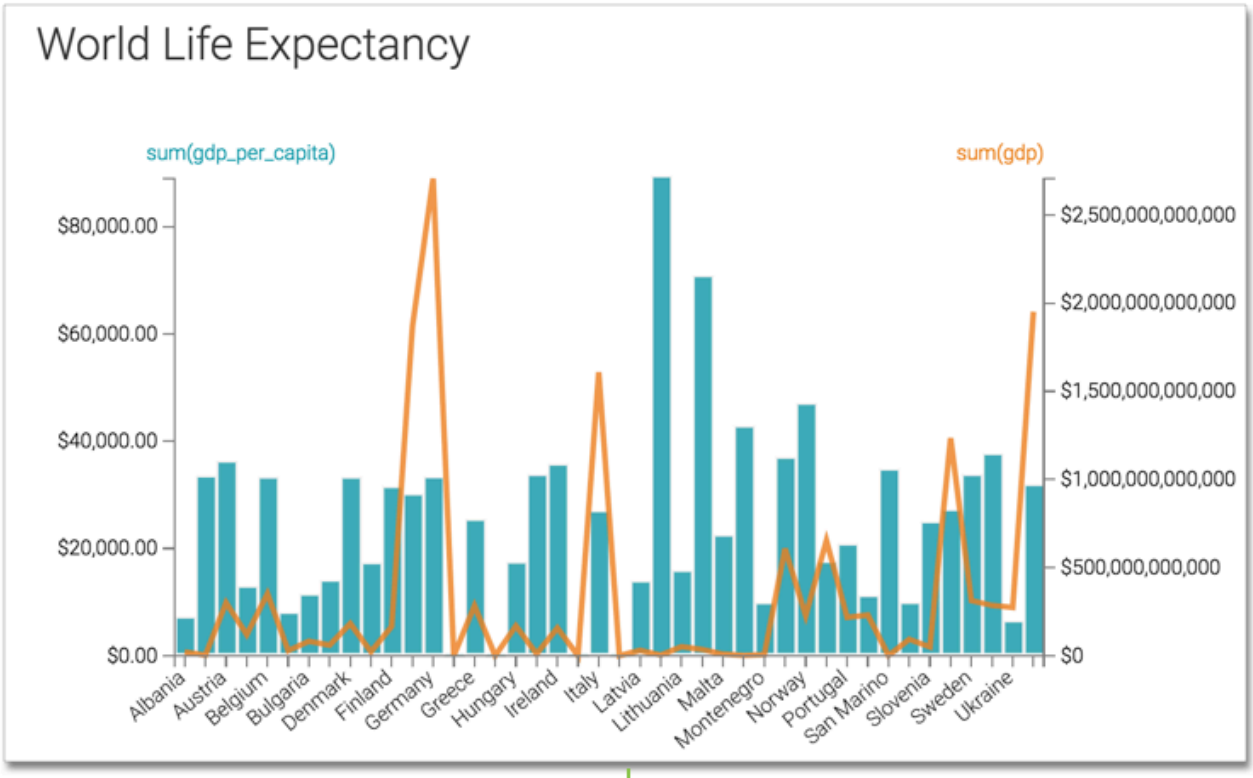

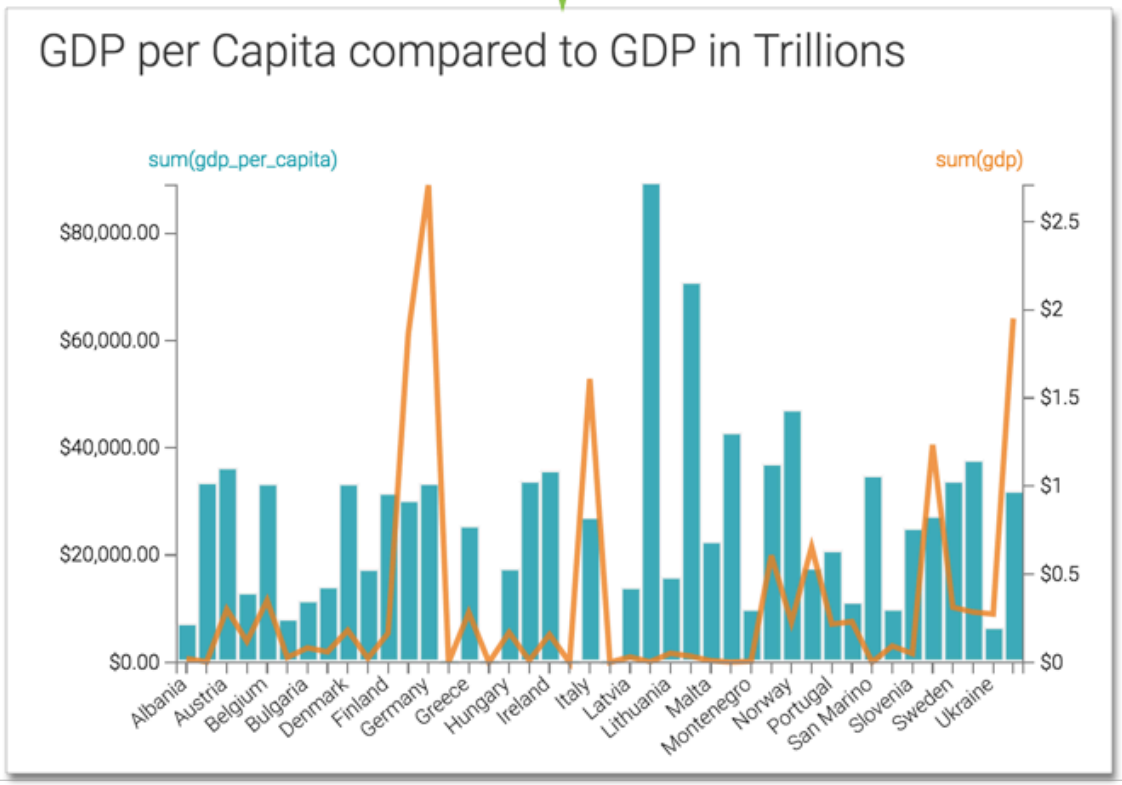

Related Information

Changing the field display format

Changing data type

Cloudera Data Visualization allows you to change the effective data type of a column in the dataset model without changing the source data. This is useful in many business environments to ensure correct processing of numerical codes, catalog numbers, event IDs, dates, and so on.

About this task

The following steps demonstrate how to change the type of a column. The column iso_cc (the ISO-compliant country code) in the dataset World Life Expectancy [data source samples.world_life_expectancy] is used.

Procedure

1. On the main navigation bar, click DATA.
2. In the left navigation menu, click samples.
3. In the Datasets area, select World Life Expectancy (samples.world_life_expectancy).
4. In the Dataset Detail menu, select Fields.

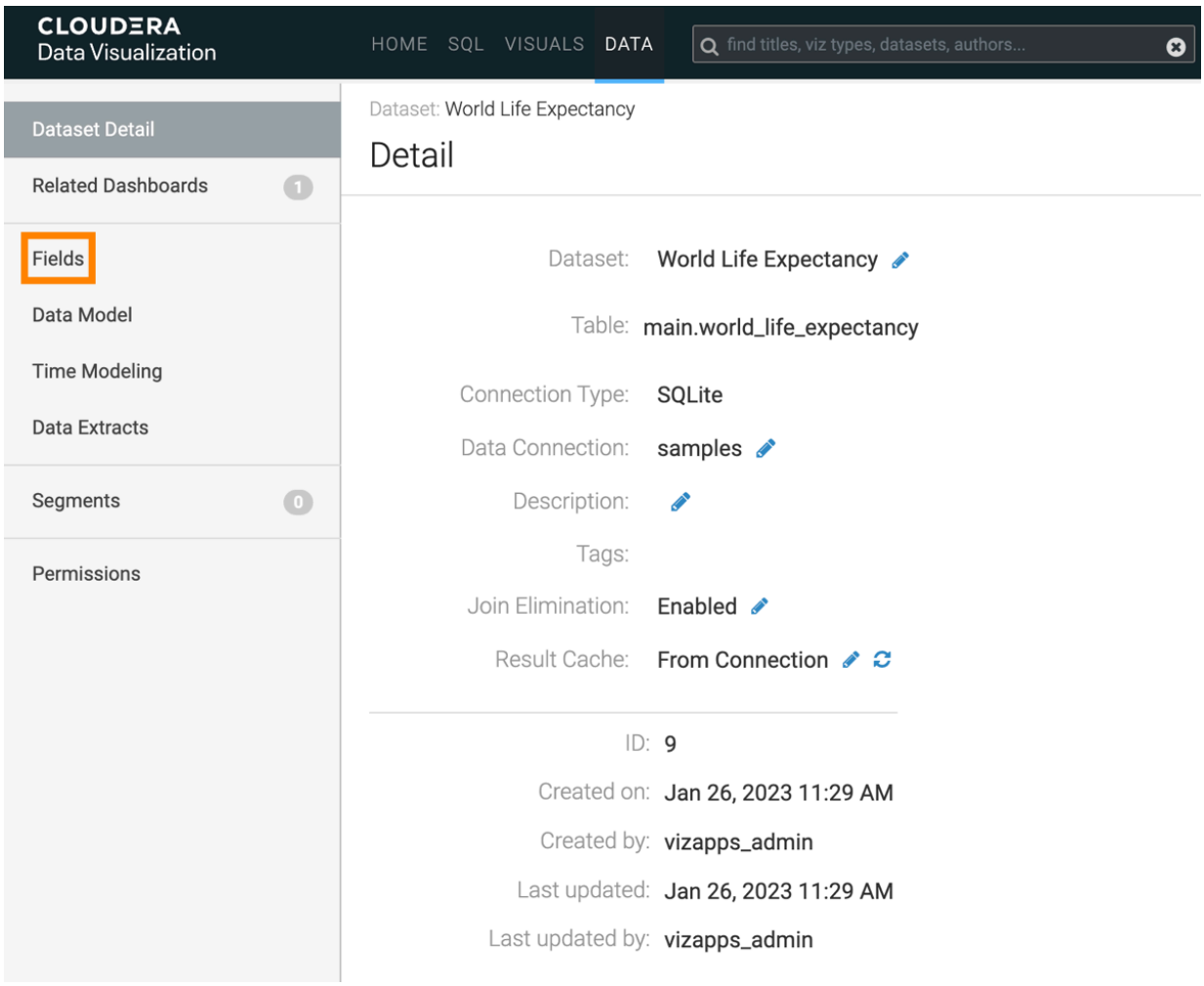

5. In the Fields interface, select EDIT FIELDS.

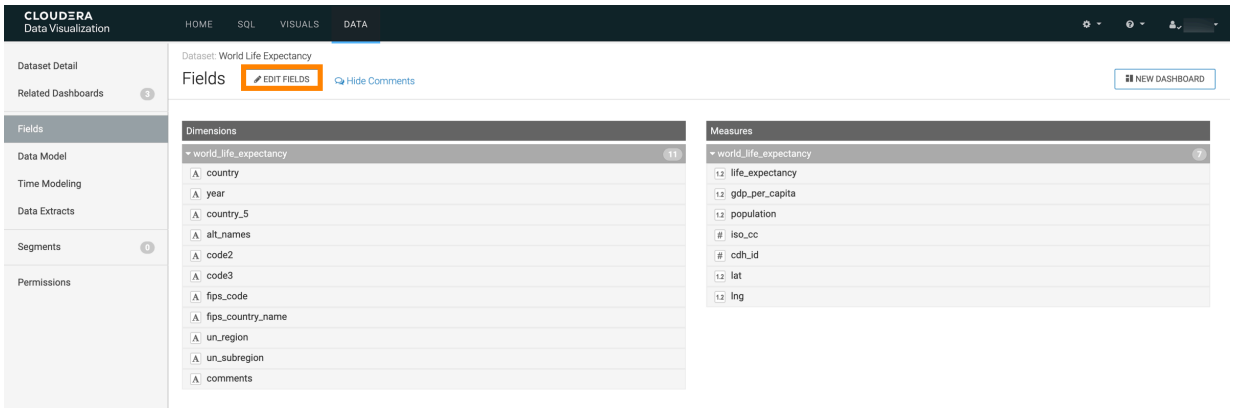

6. Under Dimensions, find the field fips_code, and click the down arrow icon, immediately following the # icon.
7. In the menu, select String.

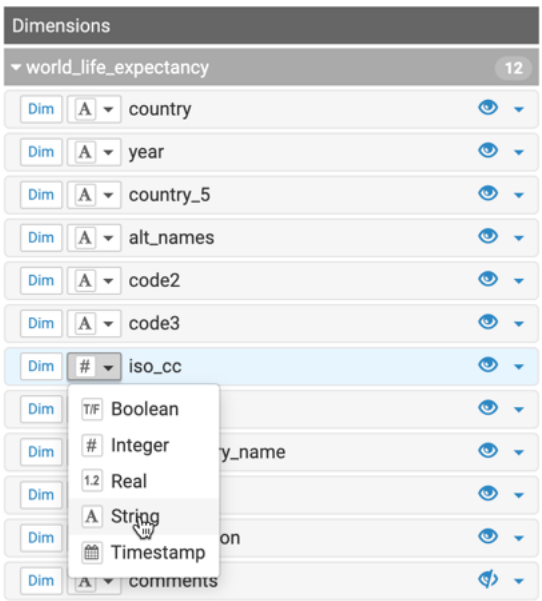

8. Under Dataset: World Life Expectancy, click Save.

Results

All new visuals created from this dataset will automatically use the new type.

Note: Use this functionality carefully, as it may break the visuals that already use the column in aggregations or custom expressions.

Specifying geographic fields

In Cloudera Data Visualization, you can explicitly specify a dataset field as one of the many supported geographical types.

About this task

In this example the dataset Canadian Census is used, constructed from example datasets, and joined of the fields fsa and Postal Code, respectively.

The following steps demonstrate how to assign Geo Types to a dataset field. We are using the two fields from the join of the Canadian Census dataset: fsa from canada_census_population_dwellings, and Postal Code from ca_postal_codes.

Note: Before creating visuals that use geographic data, specify the appropriate information as Geo Types.

Procedure

1. Navigate to the dataset that you must modify. In this example, the Food Stores Inspection in NYC dataset is used.
2. In the Dataset Detail menu, select Fields.
3. In the Fields interface, select EDIT FIELDS.

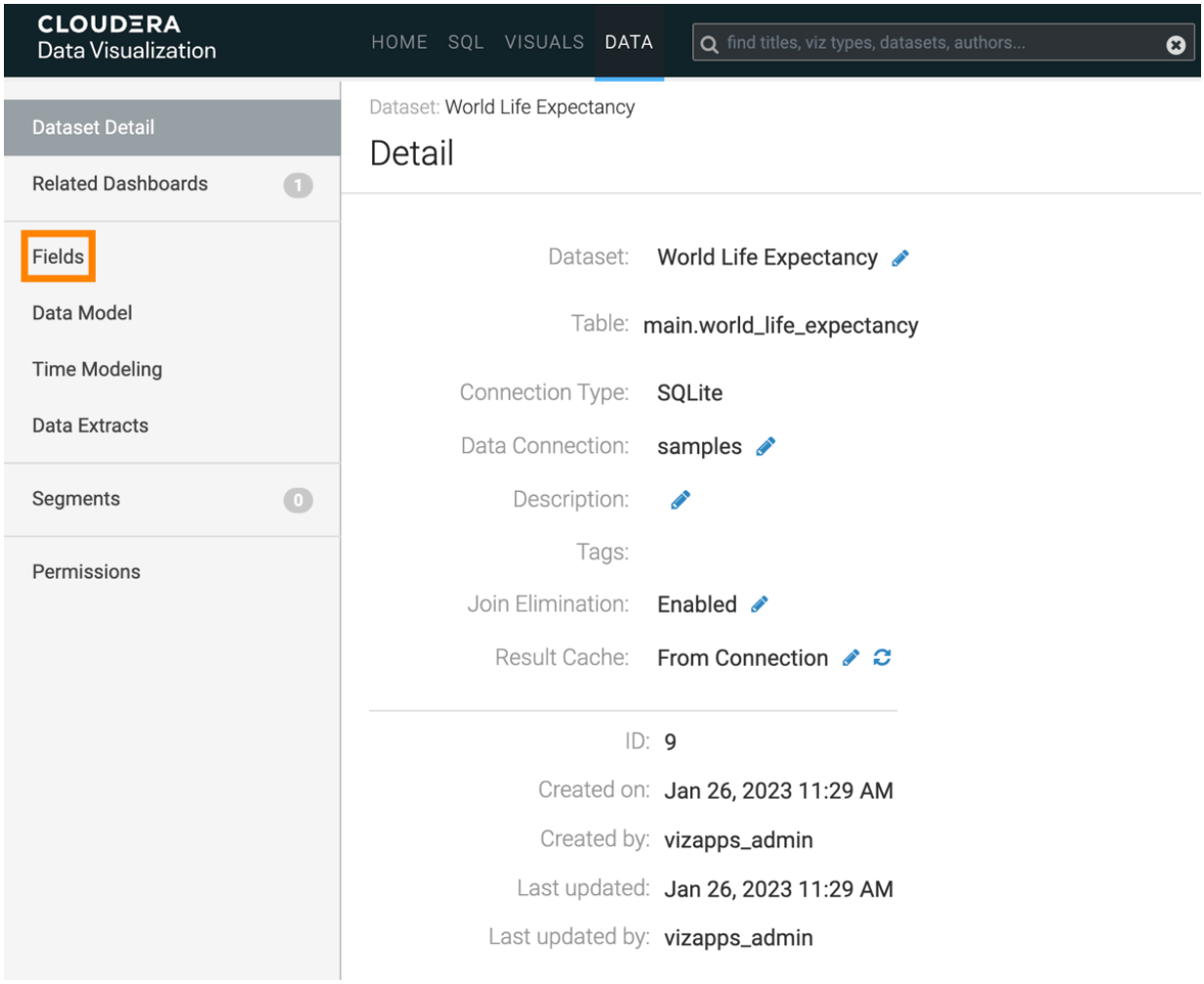

4. Click the Edit (pencil) icon on the right side of the field.

5. In the Edit Field Parameters window modal, under Geo Type, select the appropriate option from the menu.

6. Click APPLY. For zip_code, we also changed the Display Name to Postal Code in the example.

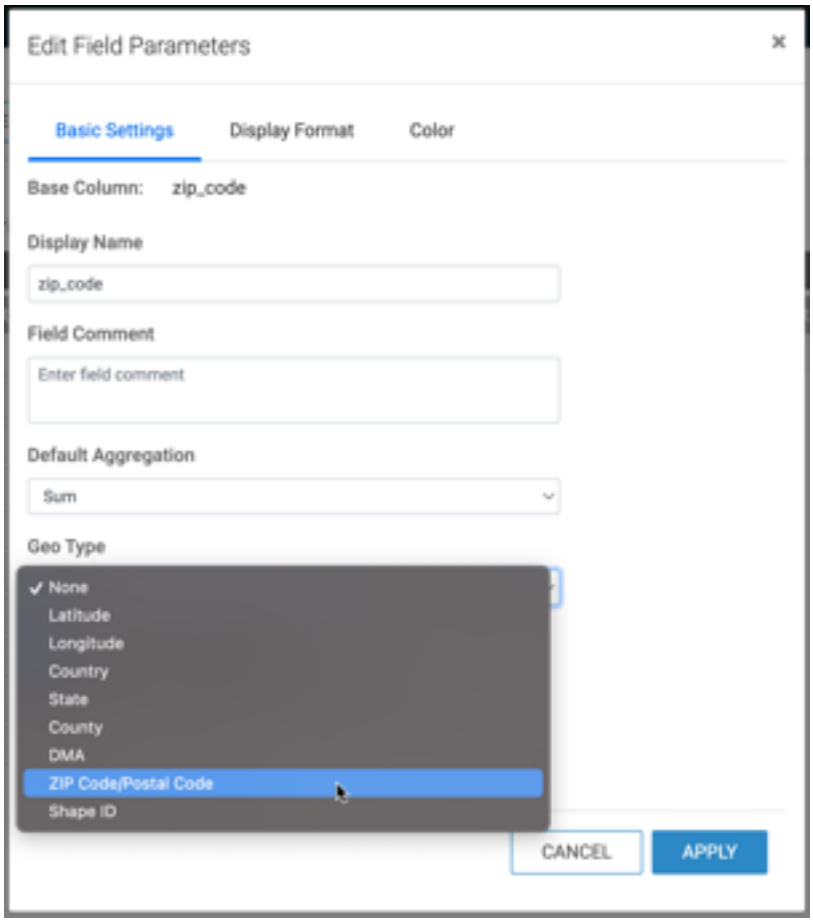

7. Under Dataset: Food Stores Inspection in NYC.
8. Click SAVE.

Results

The dataset can now be successfully used for map and interactive map visuals, without further adjustments at the level of the visual.

Adding field comments

When working with big data, it can be very helpful to have access to comprehensive field-level descriptions. In Cloudera Data Visualization, you can use field comments to provide the context and meaning of each dataset field.

Adding field comments in dataset

When working with large datasets data, it can be useful to have access to comprehensive field-level descriptions. In Cloudera Data Visualization, you can use field comments to provide the context and meaning of each dataset field.

About this task

The following steps demonstrate how to add description to a column of a dataset, as a 'comment'. In this example, the column iso_cc (the ISO-compliant country code) is used from the dataset World Life Expectancy [data source samp les.world_life_expectancy].

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.
2. In the left navigation menu, click samples.
3. In the Datasets area, select World Life Expectancy (samples.world_life_expectancy).
4. In the Dataset Detail menu, select Fields.

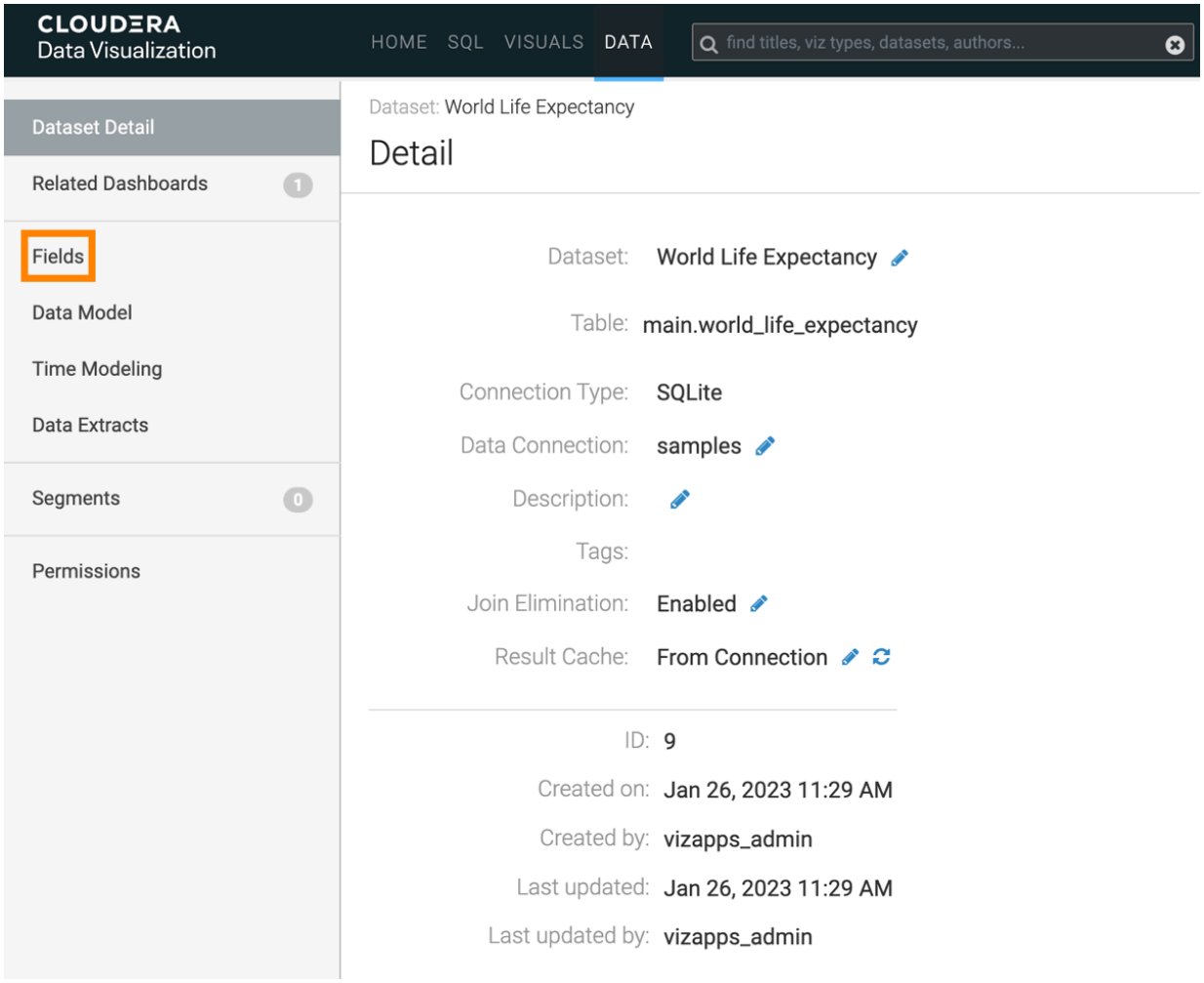

5. In the Fields interface, select EDIT FIELDS.

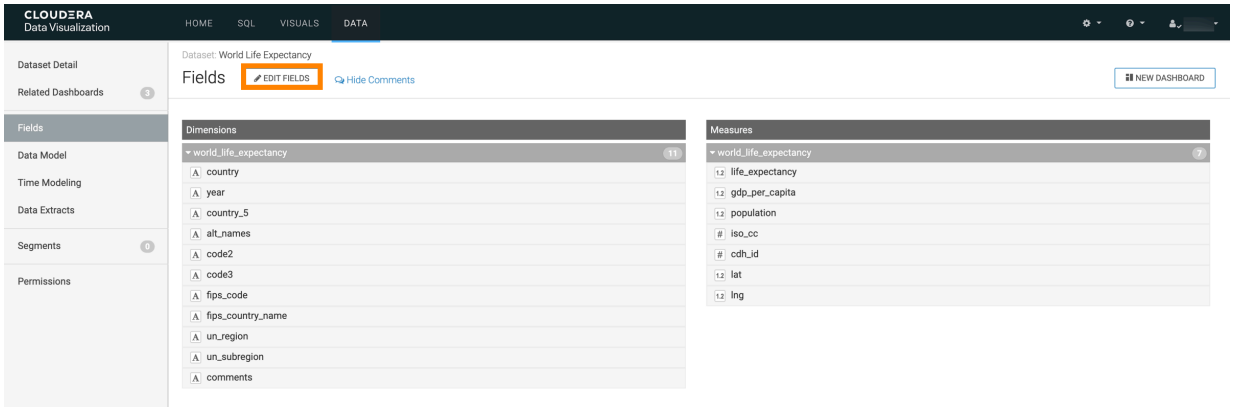

6. Under Dimensions, find the field iso_cc, and click the down arrow icon on the right.

7. Click the Edit (pencil) icon.

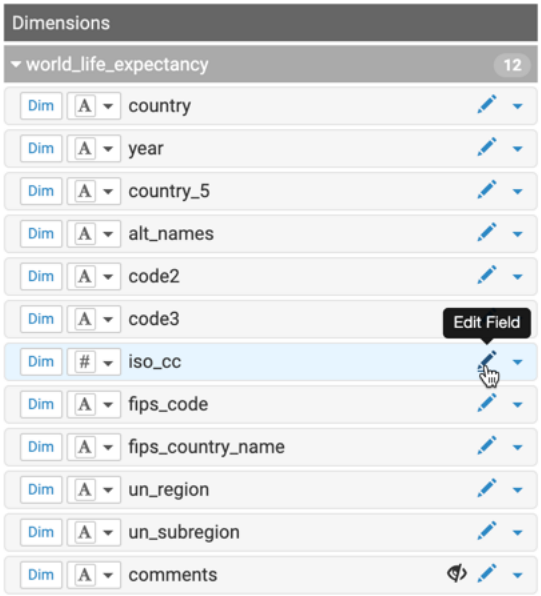

The Edit Field Parameters modal window opens.

8. Under Field Comment, add the following text:

   ISO 3166 Country Code Standard, Numeric, described in http://www.iso.org/iso/country_codes. Mappings are at https://www.iso.org/obp/ui/#search.

9. Click APPLY.

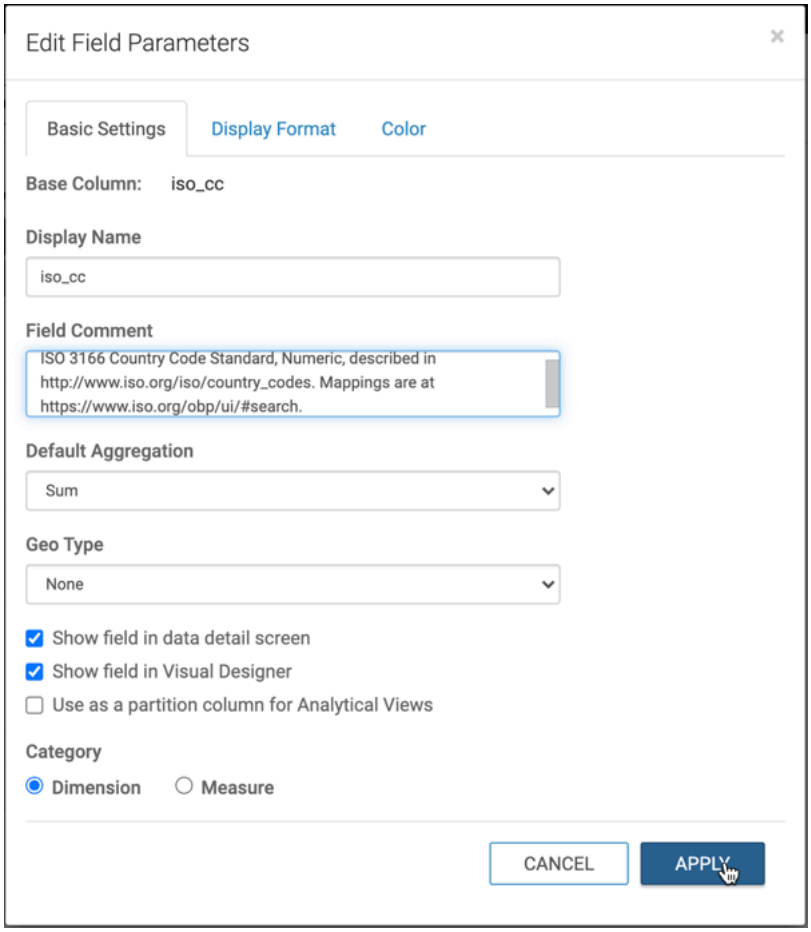

10. Click SAVE in the top row.

11. Click Show Comments to display the field comment.

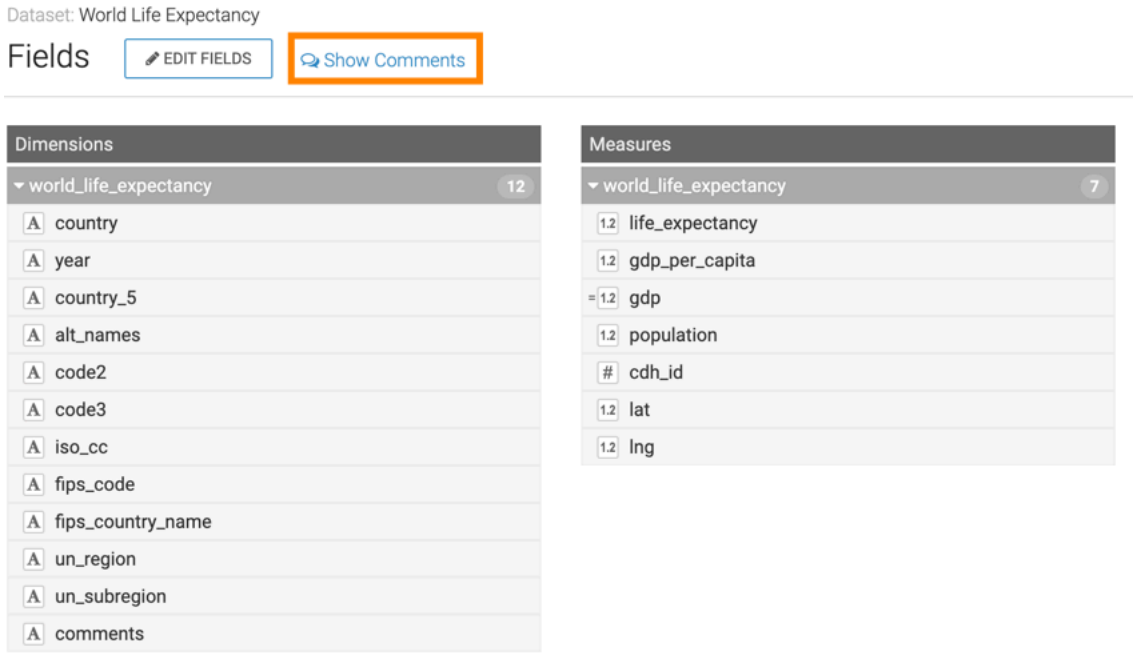

12. To hide field comments, click Hide Comments.

Viewing field comments in Visual Builder

When working with big data, it can be very helpful to have access to comprehensive field-level descriptions. In Cloudera Data Visualization, you can use field comments to provide the context and meaning of each dataset field.

To view field comments in the Visual Builder, create a new visual using the World Life Expectancy dataset [data source samples.world_life_expectancy]. For more information, see *Creating visuals*.

The fields iso-cc and gdp-per-capita display a gray triangle on the top right corner of the field. Hover over the triangle to view the field comments defined in the dataset.

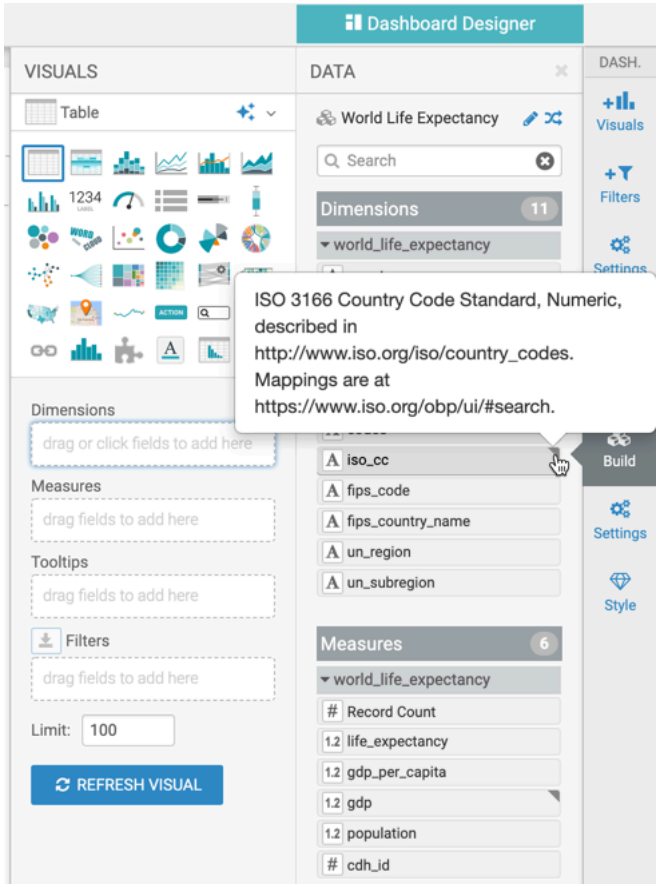

Automatically renaming dataset fields

Quite often, the column names of the base data tables are not very human-friendly. Cloudera Data Visualization gives you the option to automatically adjust field names at the level of the dataset.

About this task

The following steps demonstrate how to prevent data fields from appearing in visualizations and applications of dataset World Life Expectancy [data source samples.world_life_expectancy]. The fields comments, lat, and lng are empty, so they are good candidates for this operation.

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.
2. In the left navigation menu, click samples.
3. In the Datasets area, select World Life Expectancy (samples.world_life_expectancy).

4. In the Dataset Detail menu, select Fields.

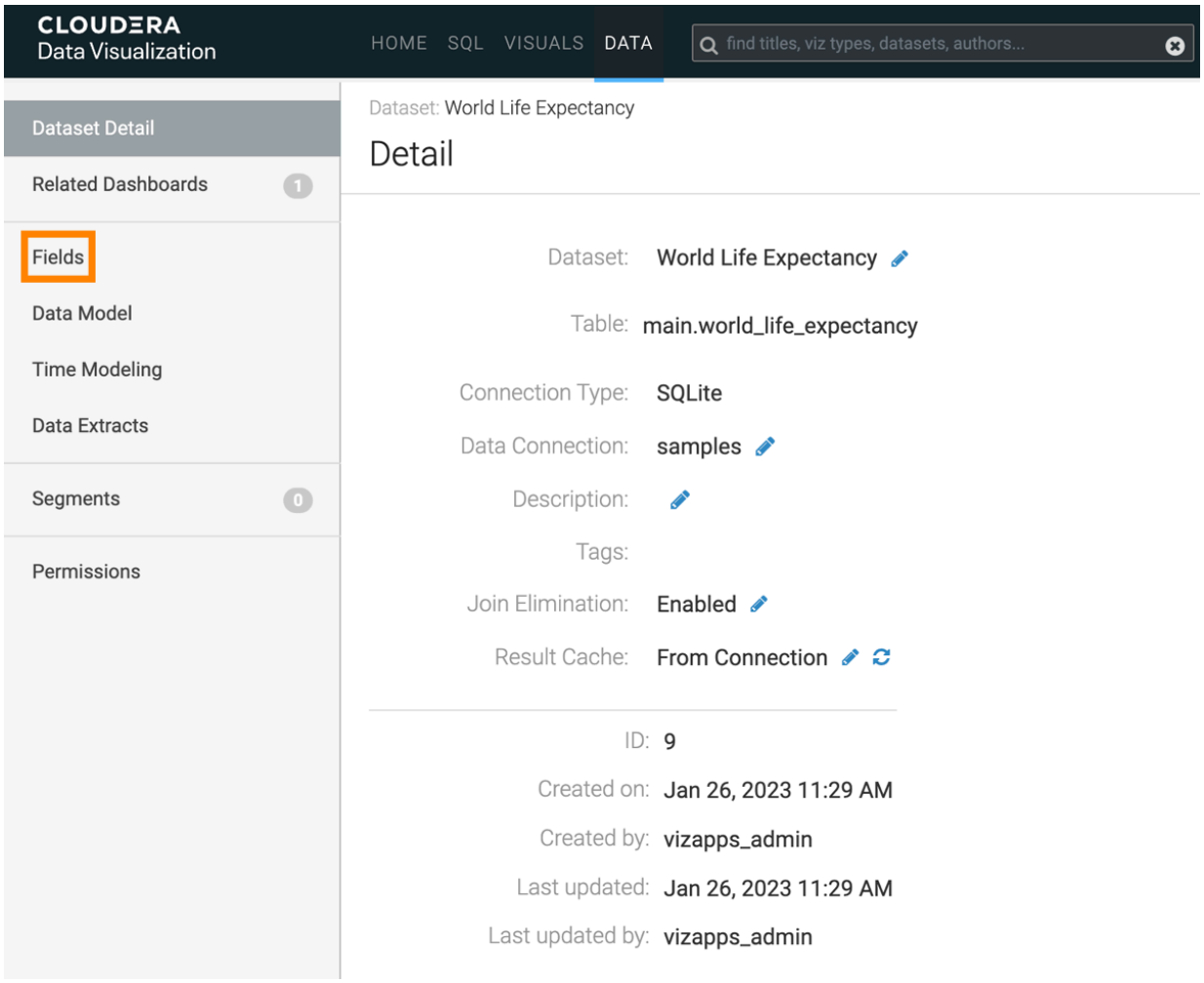

5. In the Fields interface, select EDIT FIELDS.

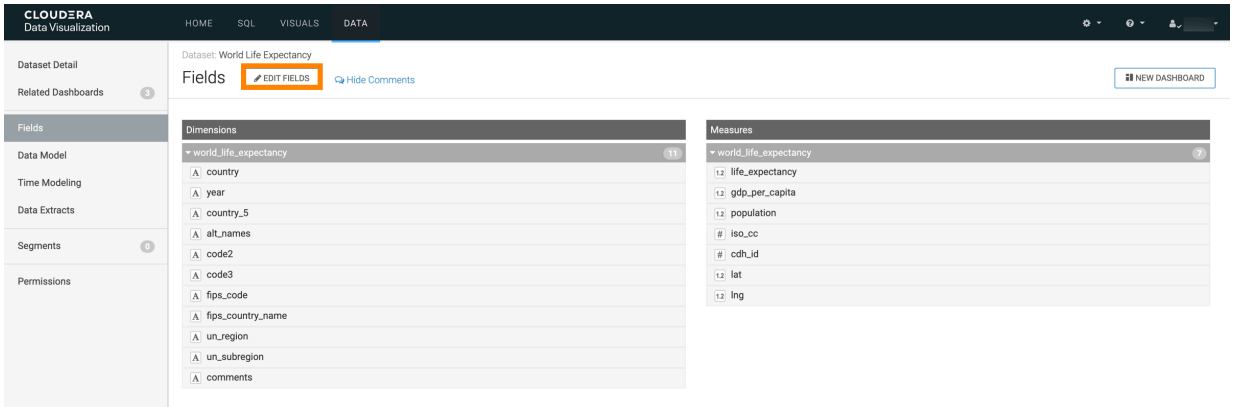

6. Near the top of the page, click TITLE CASE.

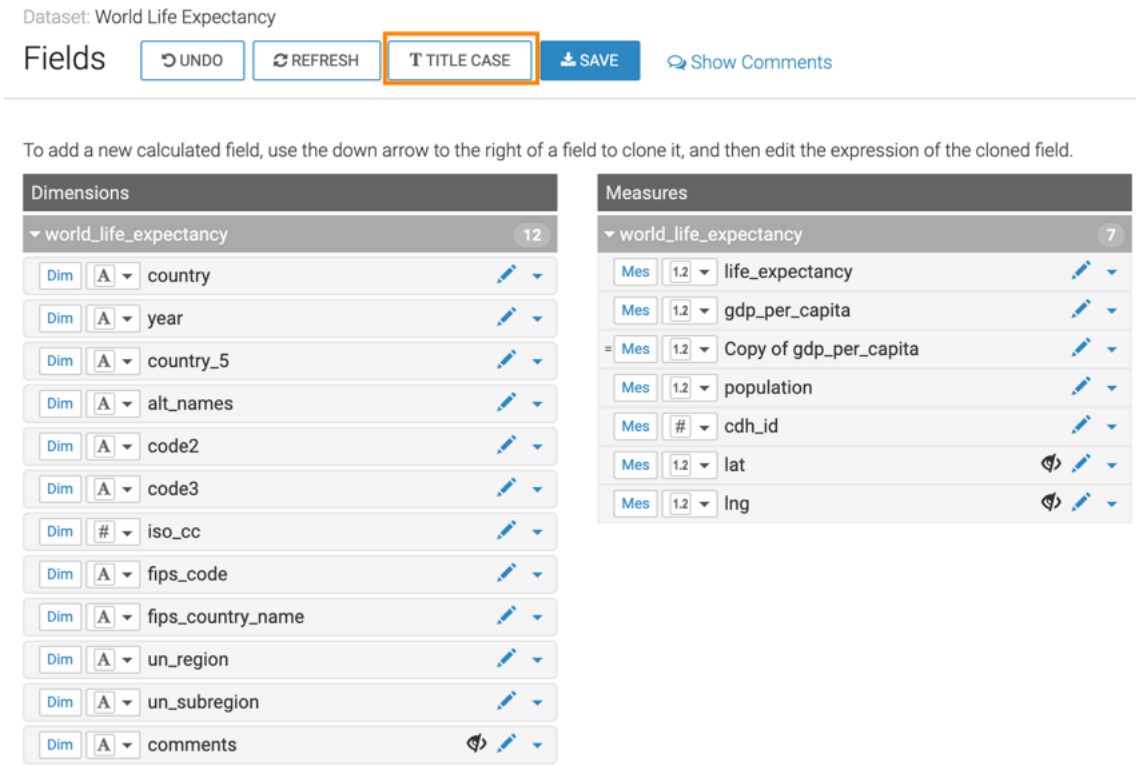

The field titles change. For example, the Measure gdp_per_captia appears as GDP Per Capita.

Note: The Title Case option does not affect fields that are calculated over table fields, such as gdp.

7. Click SAVE.

Custom renaming dataset fields

Often, we find it useful to rename a frequently-used field directly in the dataset, instead of using an alias in each visual. Cloudera Visualization makes it very easy to change the display name of a dataset field.

About this task

The following steps demonstrate how to rename a field. In this example, the column iso_cc (the ISO-compliant country code) is used from the dataset World Life Expectancy [data source: samples.world_life_expectancy].

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.
2. In the left navigation menu, click samples.
3. In the Datasets area, select World Life Expectancy (samples.world_life_expectancy).

4. In the Dataset Detail menu, select Fields.

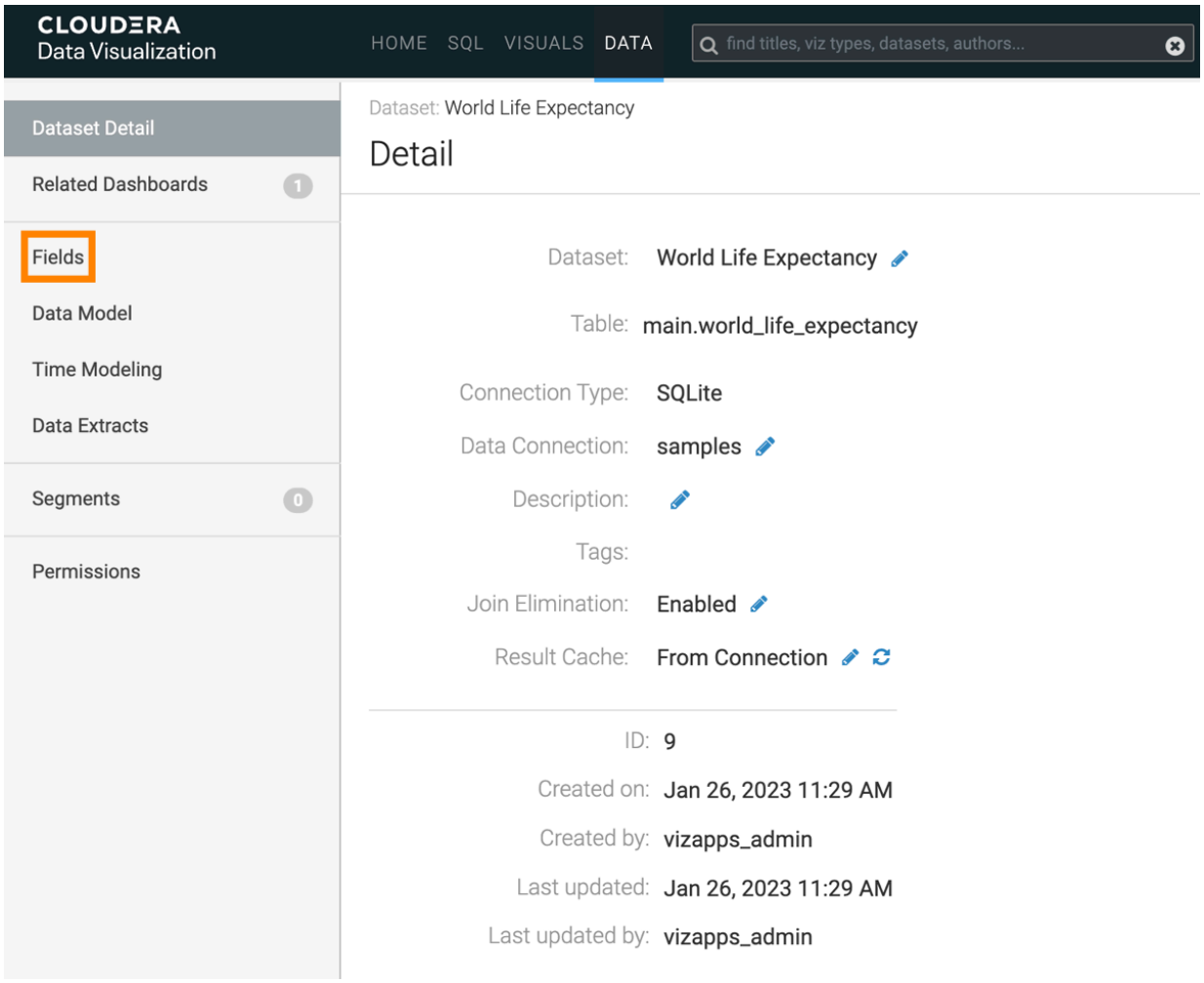

5. In the Fields interface, select EDIT FIELDS.

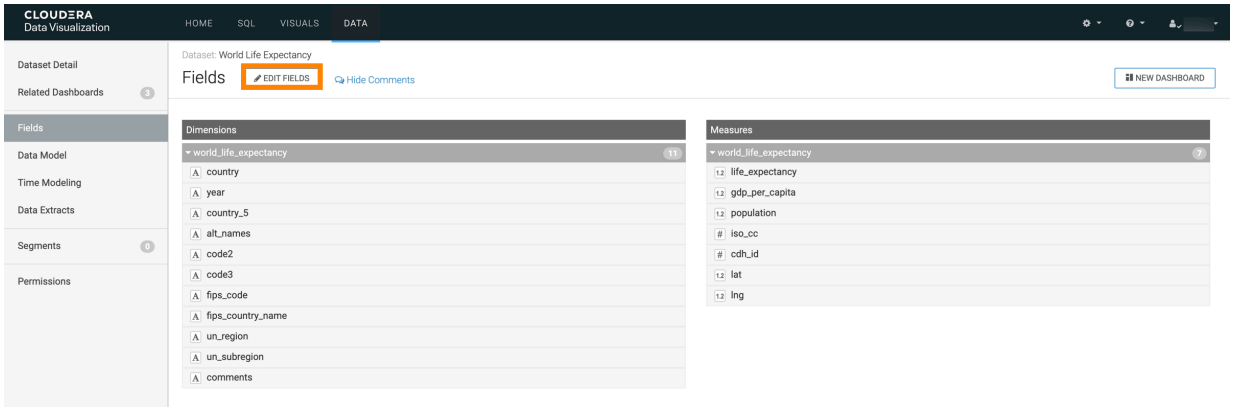

6. Under Dimensions, find the field iso_cc, and click the (down arrow) icon on the right.

7. Click the Edit (pencil) icon.

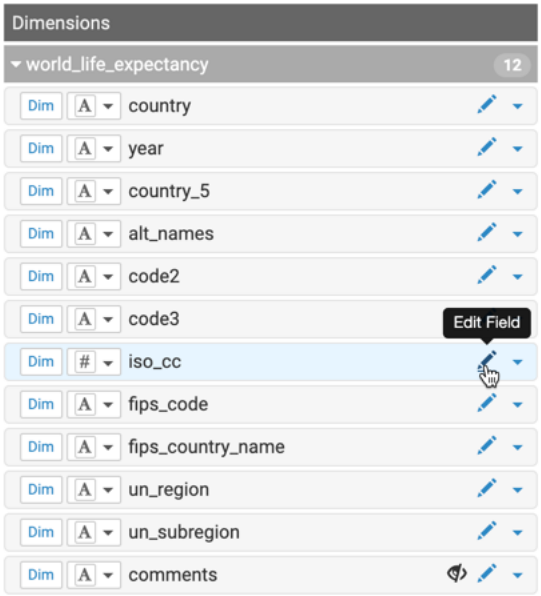

The Edit Field Parameters modal window opens.

Note: The Base Column name cannot be edited, but you can change the Display Name of the column.

8. Change the Display Name from iso_cc to ISO Country Code.
9. Click APPLY.
10. Under Dataset: World Life Expectancy, click SAVE.

Results

All new visuals created from this dataset use the new name automatically.

Refreshing a dataset

You can refresh an existing dataset if the underlying tables change or if you update the associated extract with new or changed fields.

Procedure

1. Click DATA on the main navigation bar.

   The Data view opens, displaying the Datasets tab.
2. Click the dataset you want to refresh.

   The Detail view opens.

3. Select Fields in the Dataset Detail menu.

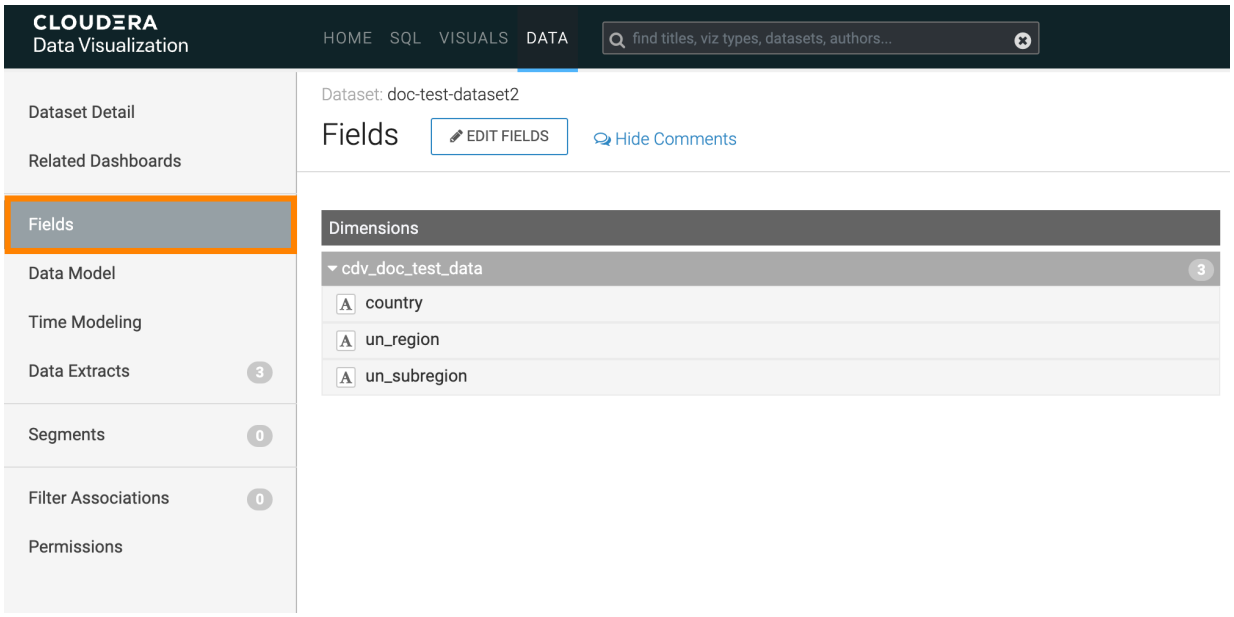

4. In the Fields interface, click EDIT FIELDS REFRESH

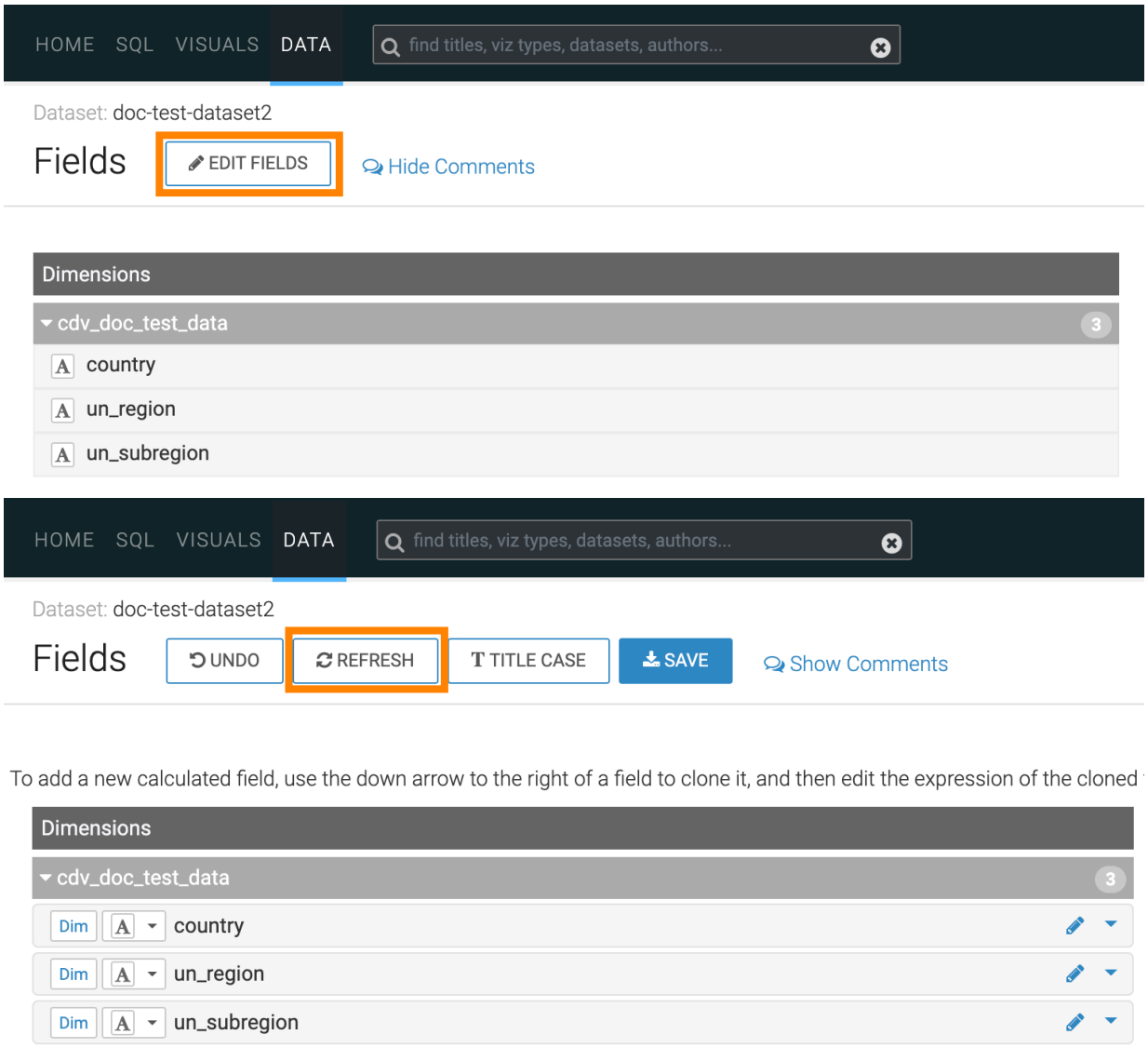

Cloudera Data Visualization scans the table for changed fields. If no changes are detected, no refresh is done and a notification is displayed. If changes are detected, the table columns are updated and a notification is displayed about the change.

5. Close the notification.

Editing datasets based on a SQL query

One of the major advantages Cloudera Data Visualization provides is the option to edit the data selection that defines the dataset.

Procedure

1. Navigate to a dataset that you created based on a SQL query.

2. Click to open the dataset, on the Dataset Detail view.

   There is a SQL text window, which you can edit.

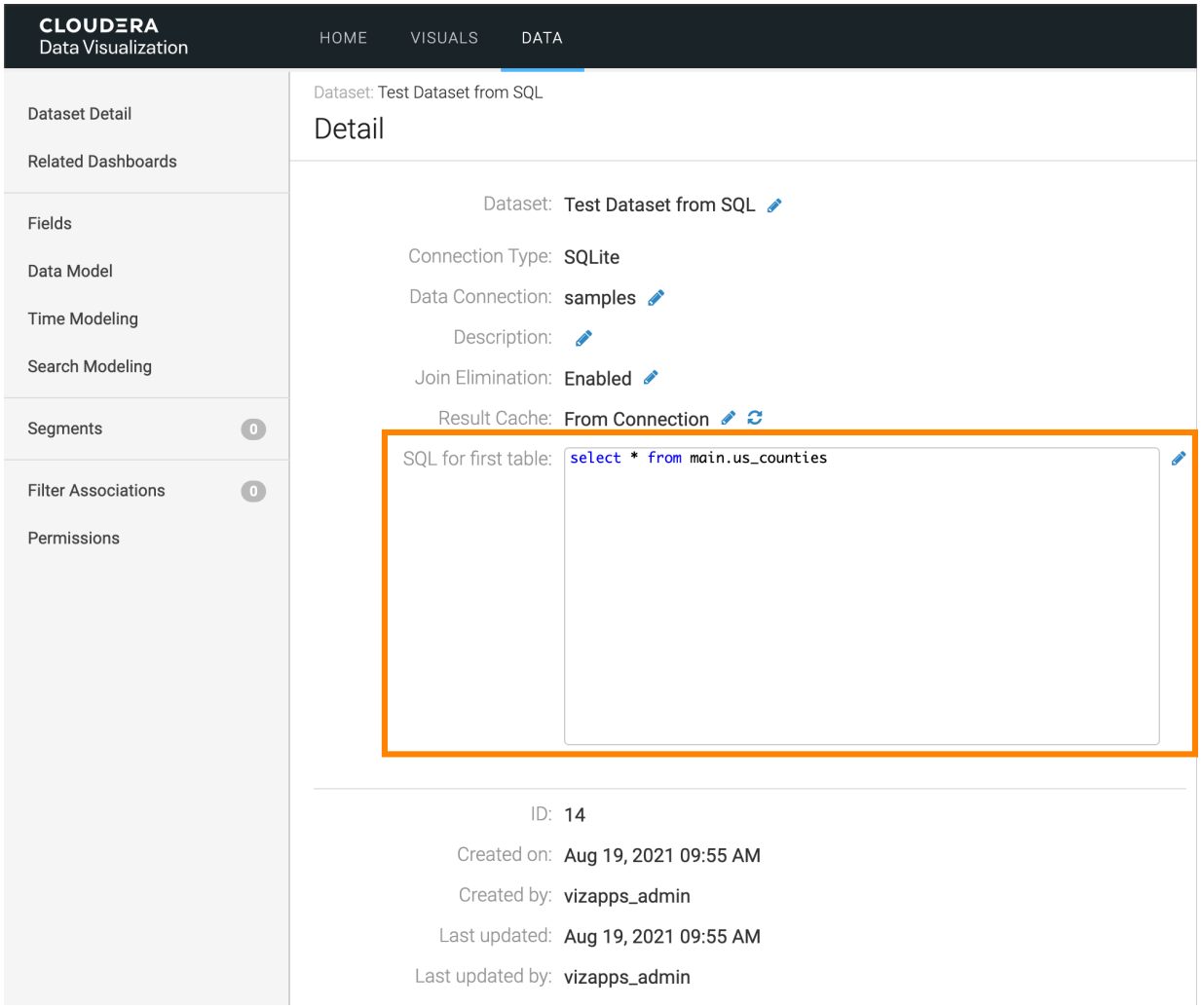

3. You can restrict rows and columns depending on what you need.

Restricting columns in datasets based on SQL query

In Cloudera Data Visualization, you can easily restrict the table columns in the dataset by changing the SQL definition of that dataset. SQL-defined datasets make it easy to limit their content to specific columns.

Procedure

1. Switch to Data Model interface, and click Show Data.

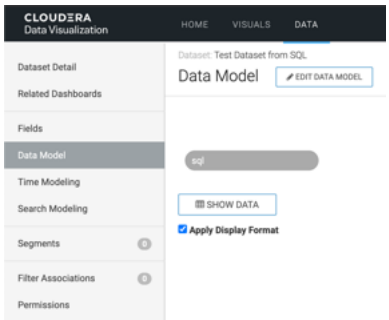

There is a large number of columns in the query result, and many of them are not necessary when it comes to answering most common questions.

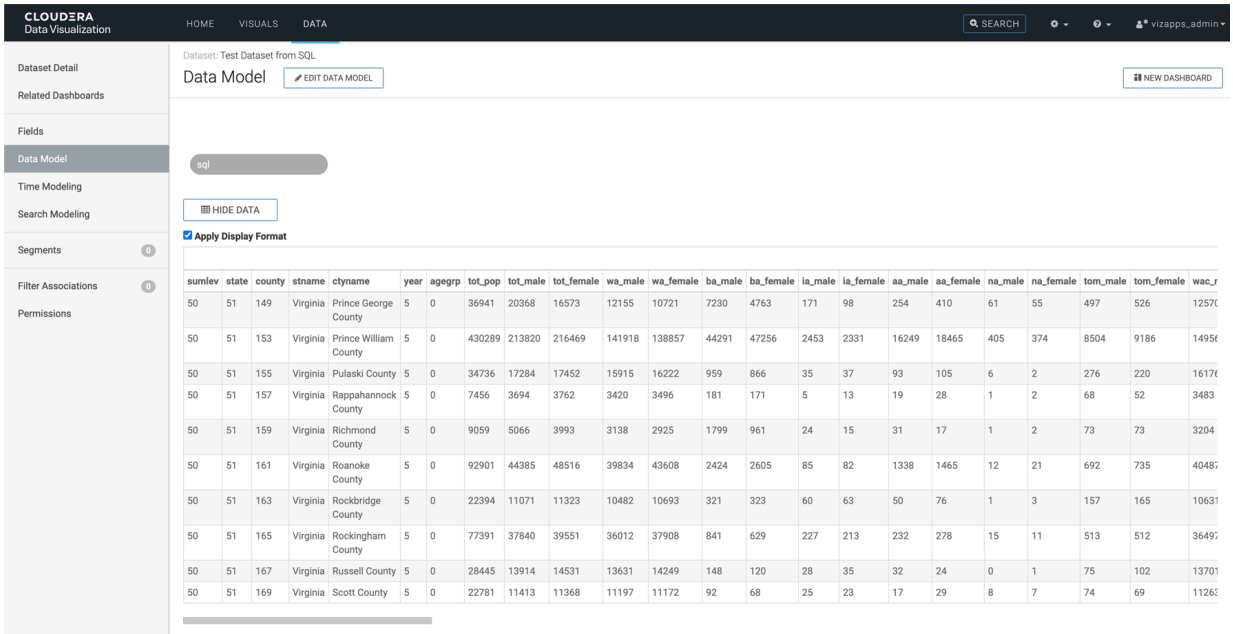

sumlev	state	county	stname	ctyname	year	agegrp	tot_pop	tot_male	tot_female	wa_male	wa_female	ba_male	ba_female	ia_male	ia_female	aa_male	aa_female	na_male	na_female	tom_male	tom_female	wac_r
50	51	149	Virginia	Prince George County	5	0	36941	20368	16573	12155	10721	7230	4763	171	98	254	410	61	55	497	526	1257
50	51	153	Virginia	Prince William County	5	0	430289	213820	216469	141918	138857	44291	47256	2453	2331	16249	18465	405	374	8504	9186	1495
50	51	155	Virginia	Pulaski County	5	0	34736	17284	17452	15915	16222	959	866	35	37	93	105	6	2	276	220	1617
50	51	157	Virginia	Rappahannock County	5	0	7456	3694	3762	3420	3496	181	171	5	13	19	28	1	2	68	52	3483
50	51	159	Virginia	Richmond County	5	0	9059	5066	3993	3138	2925	1799	961	24	15	31	17	1	2	73	73	3204
50	51	161	Virginia	Roanoke County	5	0	92901	44385	48516	39834	43608	2424	2605	85	82	1338	1465	12	21	692	735	4048
50	51	163	Virginia	Rockbridge County	5	0	22394	11071	11323	10482	10693	321	323	60	63	50	76	1	3	157	165	1063
50	51	165	Virginia	Rockingham County	5	0	77391	37840	39551	36012	37908	841	629	227	213	232	278	15	11	513	512	3649
50	51	167	Virginia	Russell County	5	0	28445	13914	14531	13631	14249	148	120	28	35	32	24	0	1	75	102	1370
50	51	169	Virginia	Scott County	5	0	22781	11413	11368	11197	11172	92	68	25	23	17	29	8	7	74	69	1126

2. Find the fields that you would like to keep in the dataset definition.
3. Switch back to Dataset Detail interface, and edit SQL text window by applying the following statement:

```
select county, stname, ctyname, tot_pop, tot_male, tot_female from main.u
s_counties
```

In this example we keep the columns county, stname, ctyname, tot_pop, tot_male, and tot_female.

4. Click Save.

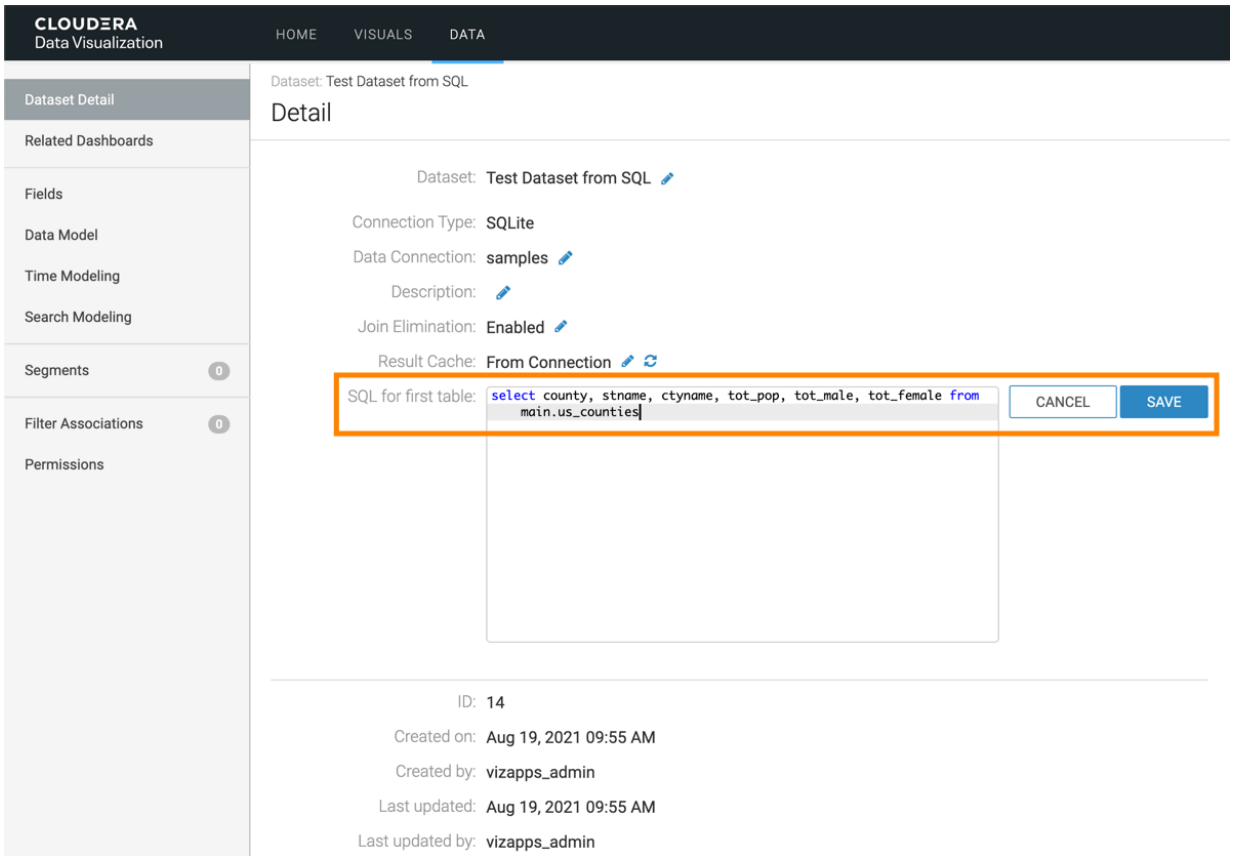

5. In the Refresh dataset table column information modal window, click Close.

6. Switch back to the Data Model interface, click Show Data, and check that the dataset only has the explicitly specified columns:

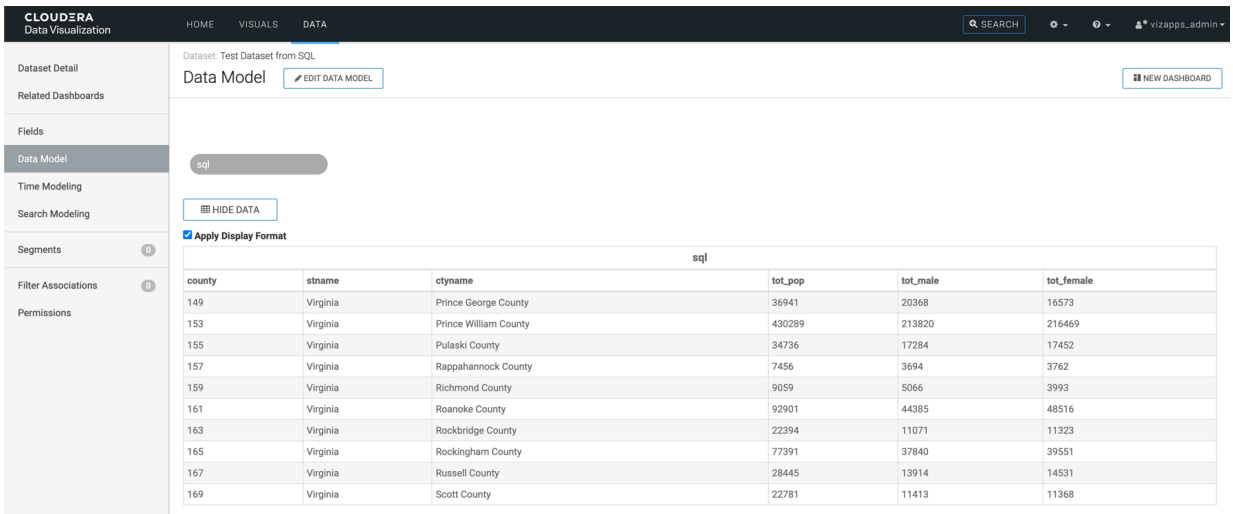

sql

county	stname	ctyname	tot_pop	tot_male	tot_female
149	Virginia	Prince George County	36941	20368	16573
153	Virginia	Prince William County	430289	213820	216469
155	Virginia	Pulaski County	34736	17284	17452
157	Virginia	Rappahannock County	7456	3694	3762
159	Virginia	Richmond County	9059	5066	3993
161	Virginia	Roanoke County	92901	44385	48516
163	Virginia	Rockbridge County	22394	11071	11323
165	Virginia	Rockingham County	77391	37840	39551
167	Virginia	Russell County	28445	13914	14531
169	Virginia	Scott County	22781	11413	11368

In this example we have kept the columns county, stname, ctyname, tot_pop, tot_male, and tot_female.

Restricting rows in datasets based on SQL query

In Cloudera Data Visualization, you can easily restrict the table rows in the dataset by changing the SQL definition of that dataset. SQL-defined datasets make it easy to limit their content to specific rows.

Procedure

1. Switch to Dataset Detail interface, and edit SQL text window by applying the following statement:

```
select county, stname, ctyname, tot_pop, tot_male, tot_female from main.
us_counties
  where stname in ('Arizona','New Mexico',
 'California','Nevada','Colorado','Utah')
```

2. Click Save.

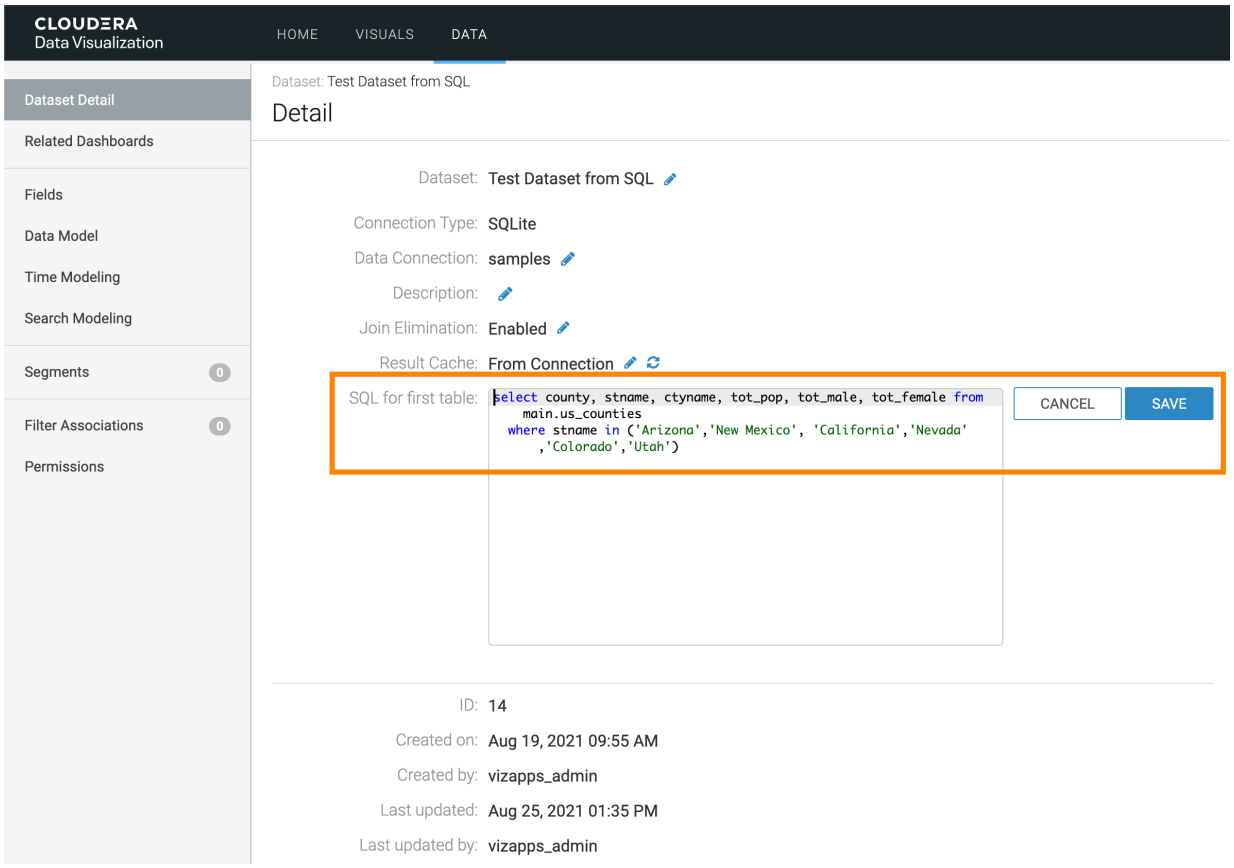

3. In the Refresh dataset table column information modal window, click Close.

4. Switch back to the Data Model interface, click Show Data, and notice that the dataset is limited to the states specified in the SQL statement.

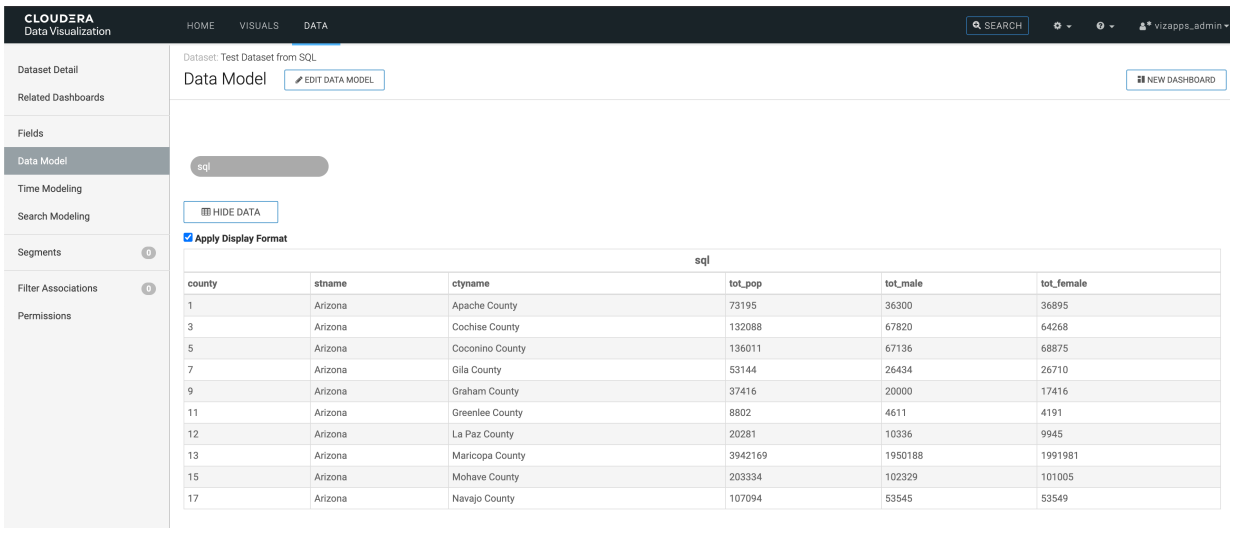

sql					
county	stname	ctyname	tot_pop	tot_male	tot_female
1	Arizona	Apache County	73195	36300	36895
3	Arizona	Cochise County	132088	67820	64268
5	Arizona	Coconino County	136011	67136	68875
7	Arizona	Gila County	53144	26434	26710
9	Arizona	Graham County	37416	20000	17416
11	Arizona	Greenlee County	8802	4611	4191
12	Arizona	La Paz County	20281	10336	9945
13	Arizona	Maricopa County	3942169	1950188	1991981
15	Arizona	Mohave County	203334	102329	101005
17	Arizona	Navajo County	107094	53545	53549

5. If you were to test it by creating a simple map visual on the dataset, it would look something like this:

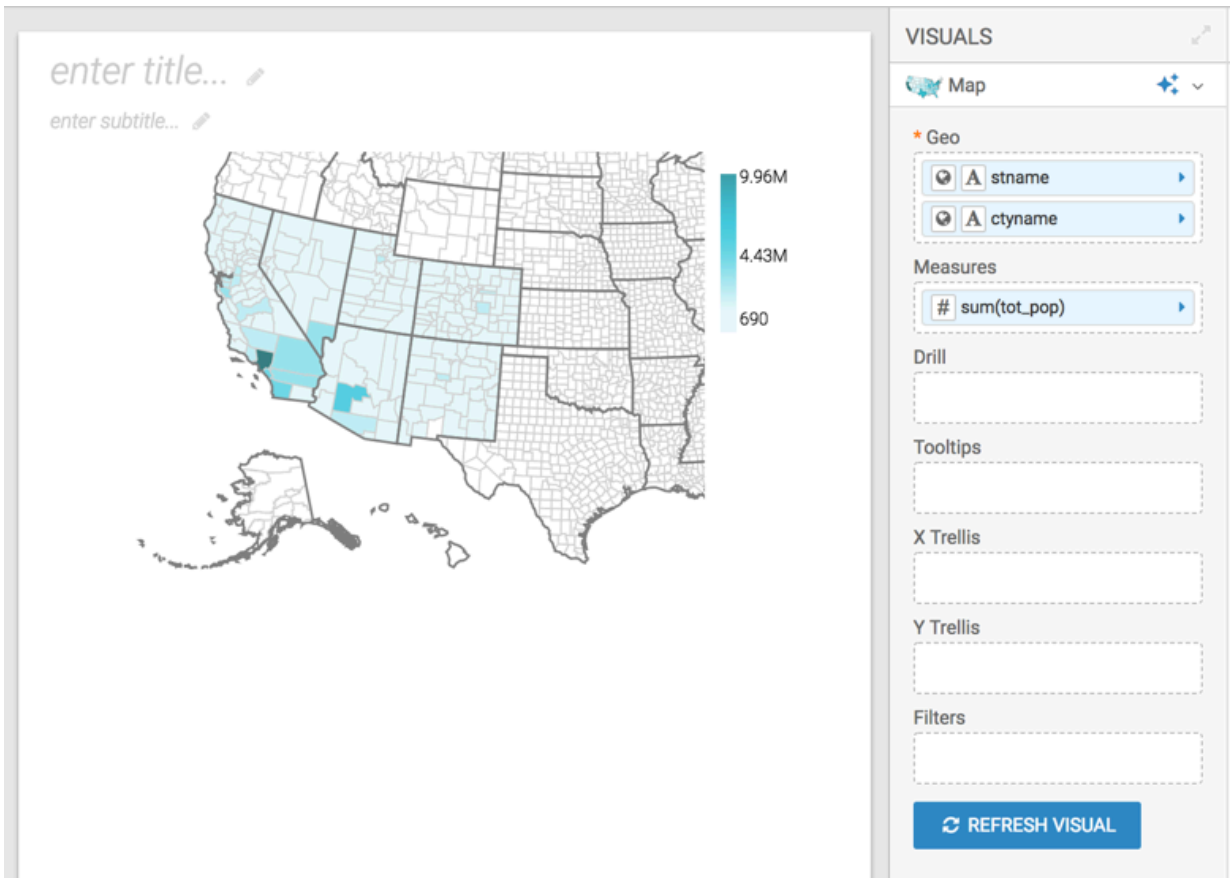

Working with data models in Cloudera Data Visualization

You can expand the basic one-table dataset by creating joins with other relevant tables from the same or other data stores. Combining data across multiple tables enriches the dataset considerably. It enables you to conduct much more meaningful research and produce insightful visual analytics.

There are two distinct approaches for creating data joins for the purpose of visualization:

- Defining in UI is ideal for datasets that include star-type schemas.
- Defining on back end ETL is preferable for fact-fact joins, so they may be pre-materialized.

Related Information

Data modeling

Creating a join

Learn how you can create new data joins in a dataset in Cloudera Data Visualization.

About this task

This example shows you how to create new data joins using the Flight Delays dataset.

Procedure

1. On the main navigation bar, click DATA.

   The Data view opens, displaying the Datasets tab.

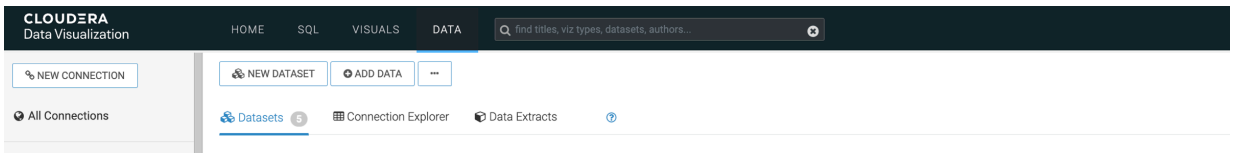

2. Create a new dataset based on the sample datafile.
3. Find the dataset in the list of datasets, either by scrolling or by using search, and click it.

   The dataset side navigation panel appears, open on Dataset Detail view.
4. In the side navigation menu, click Data Model.

   The Data Model view appears, and shows the name of the only table in the dataset. You may click SHOW DATA to display the data of that table.

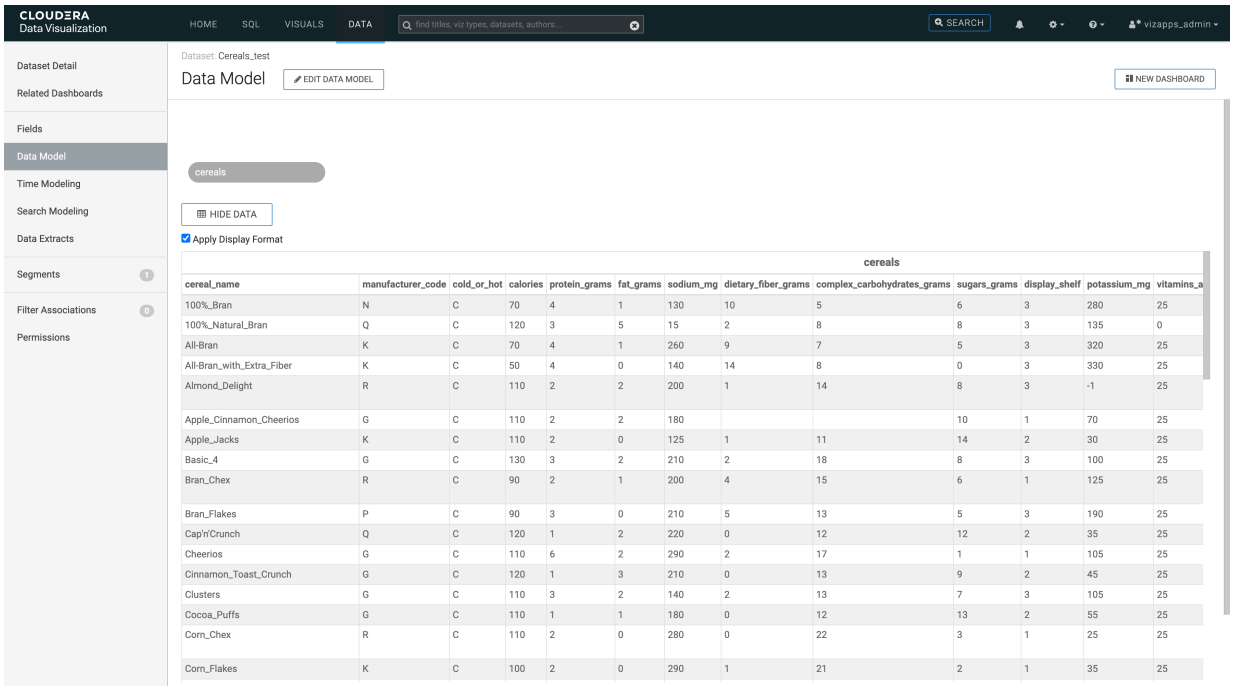

cereals

cereal_name	manufacturer_code	cold_or_hot	calories	protein_grams	fat_grams	sodium_mg	dietary_fiber_grams	complex_carbohydrates_grams	sugars_grams	display_shelf	potassium_mg	vitamins_a
100%_Bran	N	C	70	4	1	130	10	5	6	3	280	25
100%_Natural_Bran	Q	C	120	3	5	15	2	8	8	3	135	0
All-Bran	K	C	70	4	1	260	9	7	5	3	320	25
All-Bran_with_Extra_Fiber	K	C	50	4	0	140	14	8	0	3	330	25
Almond_Delight	R	C	110	2	2	200	1	14	8	3	-1	25
Apple_Cinnamon_Cheerios	G	C	110	2	2	180			10	1	70	25
Apple_Jacks	K	C	110	2	0	125	1	11	14	2	30	25
Basic_4	G	C	130	3	2	210	2	18	8	3	100	25
Bran_Chex	R	C	90	2	1	200	4	15	6	1	125	25
Bran_Flakes	P	C	90	3	0	210	5	13	5	3	190	25
Cap'n'Crunch	Q	C	120	1	2	220	0	12	12	2	35	25
Cheerios	G	C	110	6	2	290	2	17	1	1	105	25
Cinnamon_Toast_Crunch	G	C	120	1	3	210	0	13	9	2	45	25
Clusters	G	C	110	3	2	140	2	13	7	3	105	25
Cocoa_Puffs	G	C	110	1	1	180	0	12	13	2	55	25
Corn_Chex	R	C	110	2	0	280	0	22	3	1	25	25
Corn_Flakes	K	C	100	2	0	290	1	21	2	1	35	25

5. Click EDIT DATA MODEL.

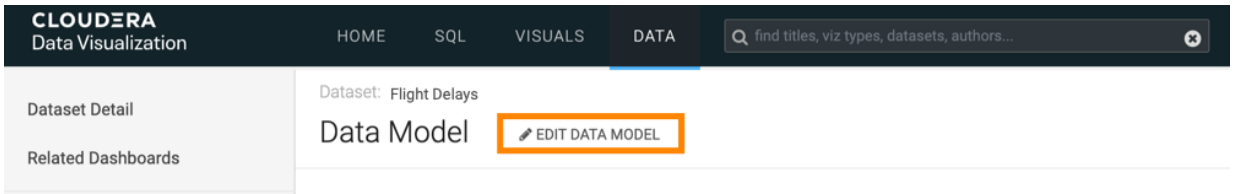

6. Click the plus sign on the table representation.

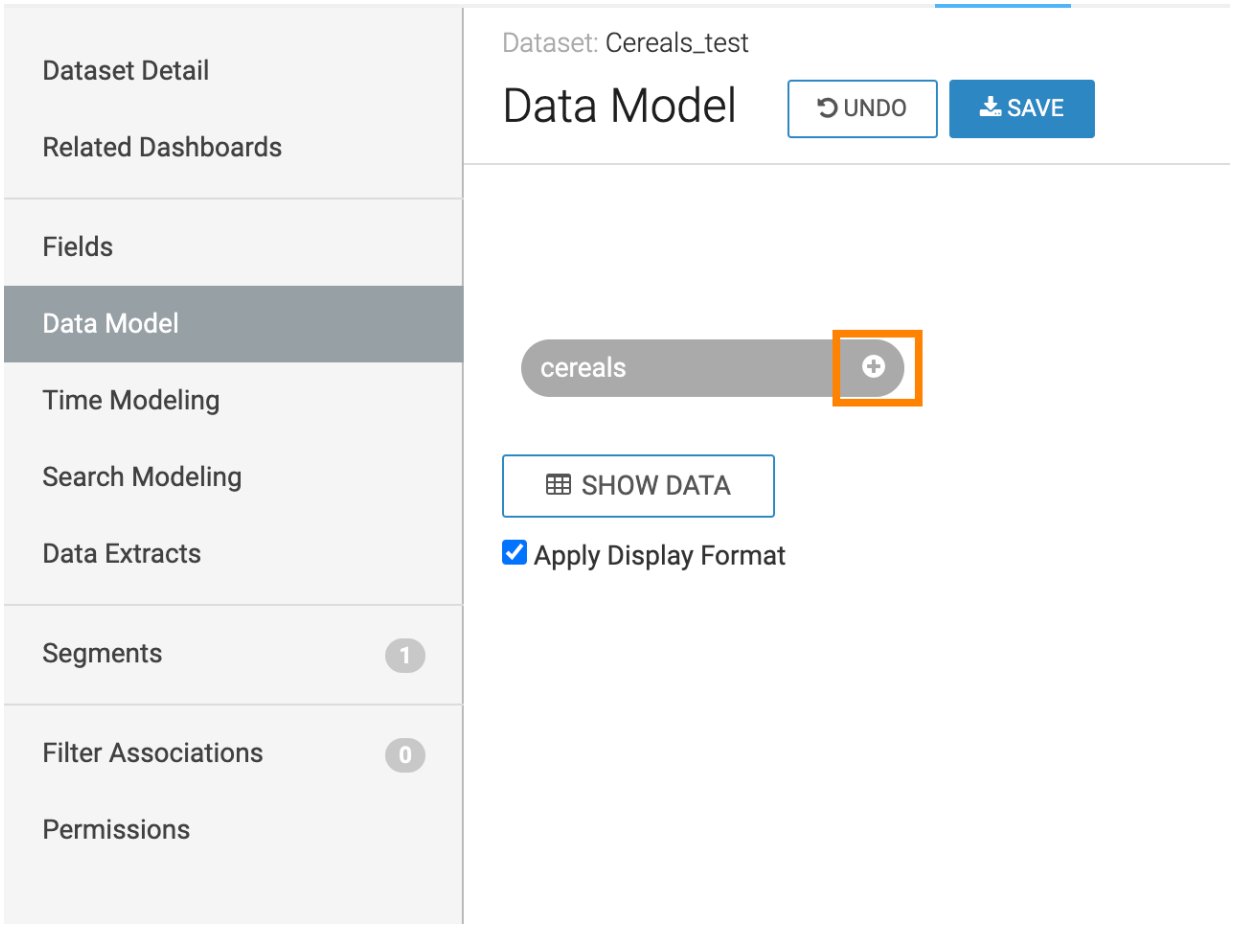

The Table Browser modal window opens.

7. In the Table Browser modal window, make the following selections:
   - In the Database Name selector, choose the database documentation.

     Note: You can join tables from different databases. This value is pre-populated to match the dataset's existing table, but it may be changed.
   - In the Table Name selector, choose the table name airline_id.

     This value is pre-populated to match the existing table of the dataset, but it may be changed.
   - Click SELECT.

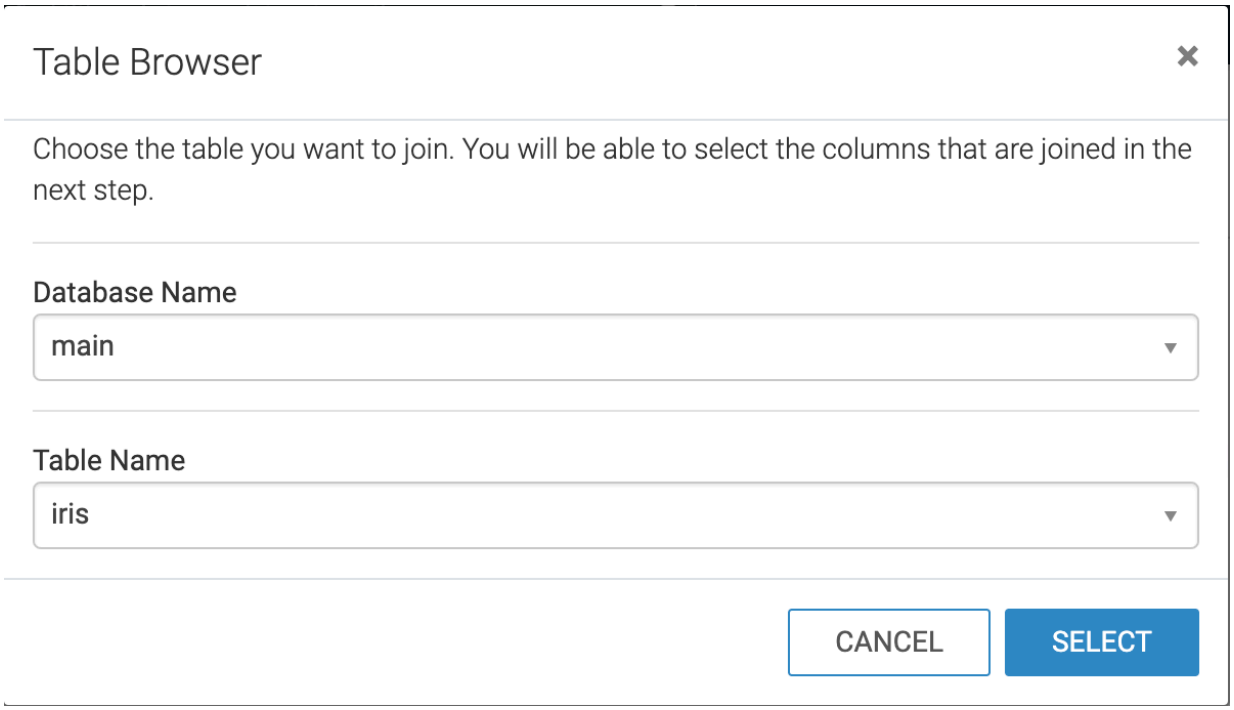

The Edit Join modal window opens.

8. In the Edit Join modal window, the following options are available:
   a. [Optional] Click Clear Fields to clear all already defined joins between the two tables.
   b. [Optional] Click sample data to preview the data. Click again to hide sample data.
   c. [Optional] Click Add Join Pair to add another column connection between the same two tables.
   d. [Optional] Click Add Join Expression to add a join between the two tables based on a custom SQL expression.
   e. [Optional] Click icon (minus) to remove an existing join pair or an existing join expression.
   f. [Optional] Under Join Expressions, click the text box to open the Join Expression interface and specify or update a custom SQL expression that defines the join conditions.
   g. Click APPLY to save the changes.

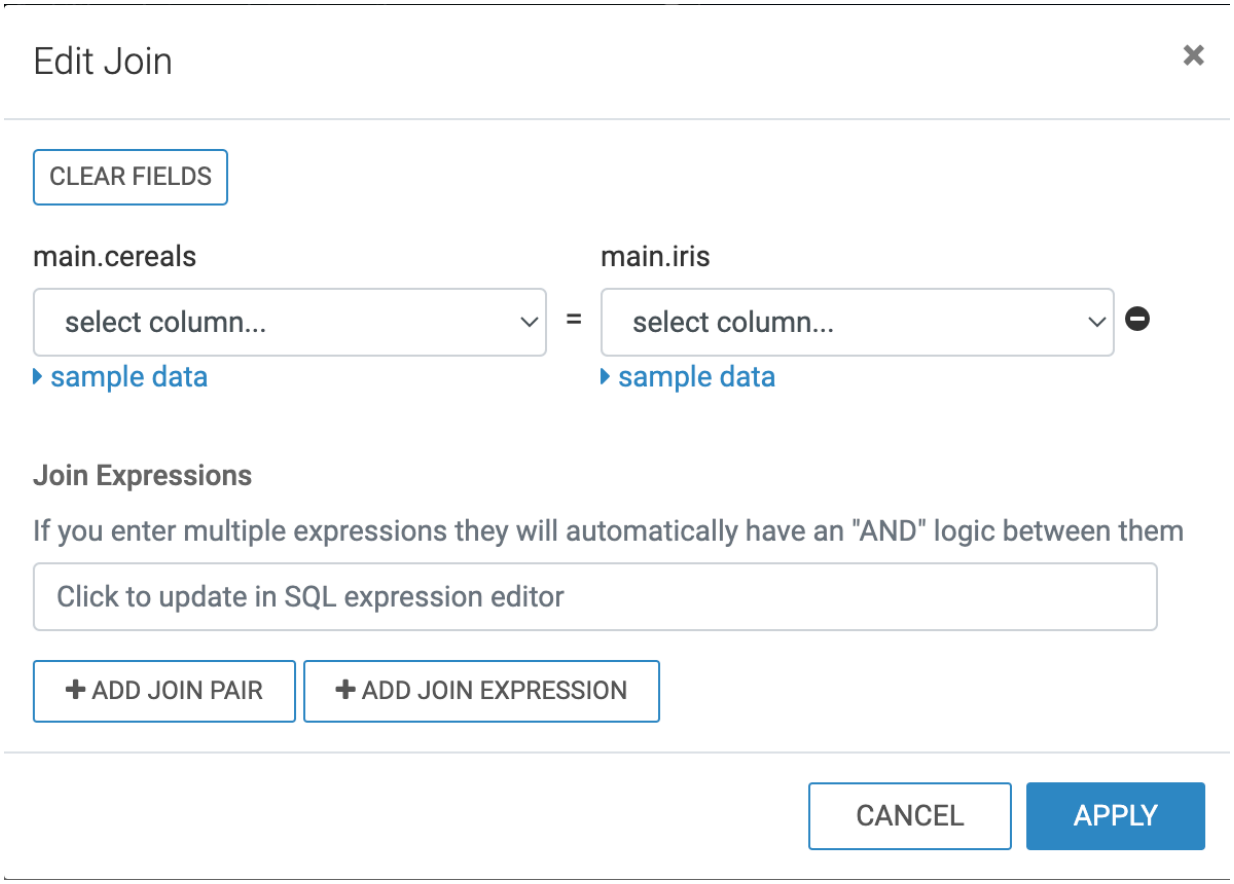

9. In the Edit Join modal window, do the following:
   - Select the matching columns for both tables. On the left side, select the field airlineid. On the right side, select the field code.
   - Click Sample Data to view some data in both columns, and verify that the join makes sense. Click again to hide sample data.
10. Click APPLY.

11. Repeat the previous two steps to create seven more joins as follows:

- The table airport_codes has two joins to the main table, and you must create each join separately as follows:
  - Left column origin = right column code.
  - Left column dest = right column code.
- The table cancellation_code has a join for left column cancellationcode = right column code.
- The table airport_lat_long has two joins to the main table, and you must create each join separately as follows:
  - Left column origin = right column locationid.
  - Left column dest = right column locationid.
- The table state_abbreviations has two joins to the main table, and you must create each join separately:
  - Left column deststate = right column abbreviation.
  - Left column originstate = right column abbreviation.

  This step is optional, and depends on whether your flights_* table has fully extended state names.

12. Click SAVE.

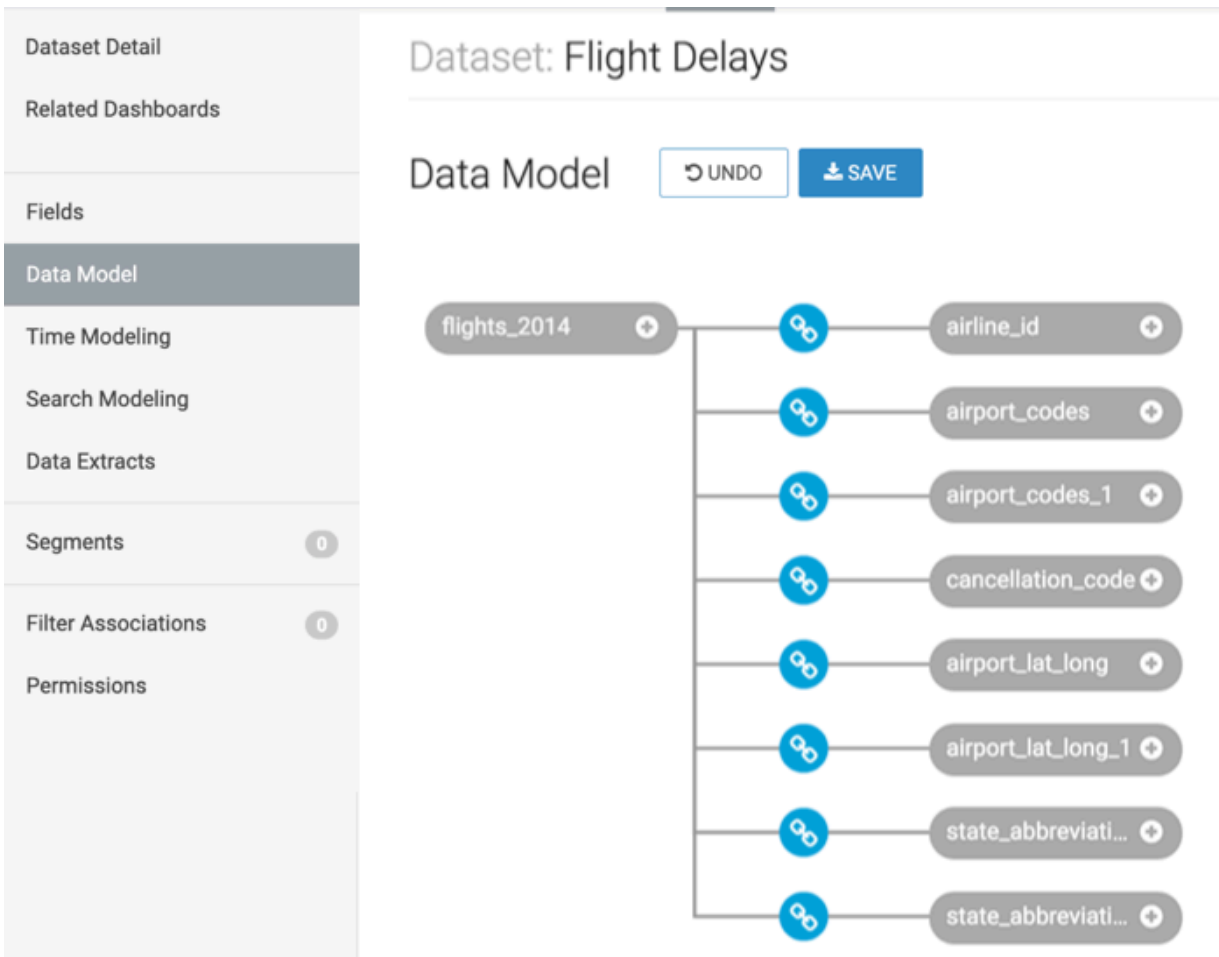

13. Click the (link) icon to edit joins or to change join type.

Changing join types

Learn how you can change the join type in a table in Cloudera Data Visualization.

About this task

The following steps demonstrate how to change the join type in the airport_codes table from the default Left join to the Right join.

Procedure

1. Navigate to the Data Model page of the dataset.
2. Click Edit Data Model to edit the data model.

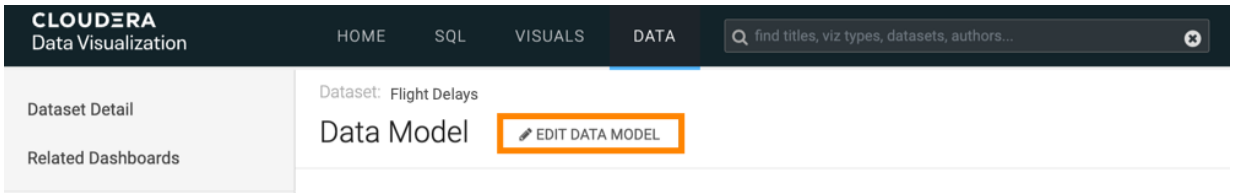

3. Click the (link) icon that represents the connection that must be changed.

   In our example, we clicked the join with the table airport_codes.

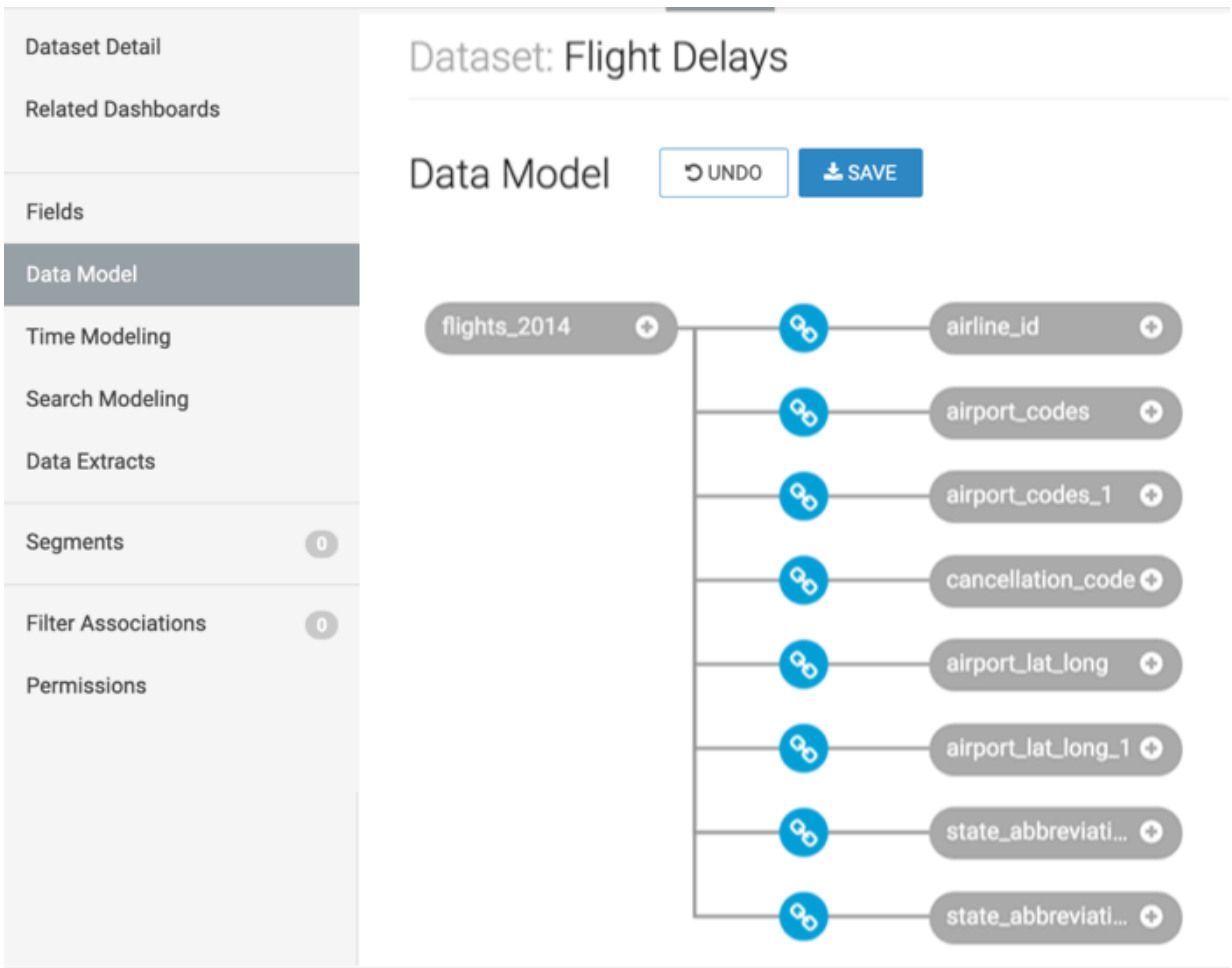

The Join Details modal window opens.

4. In the Join Details modal window, select an alternate join type.

   For example, instead of the default Left join, select Right join.

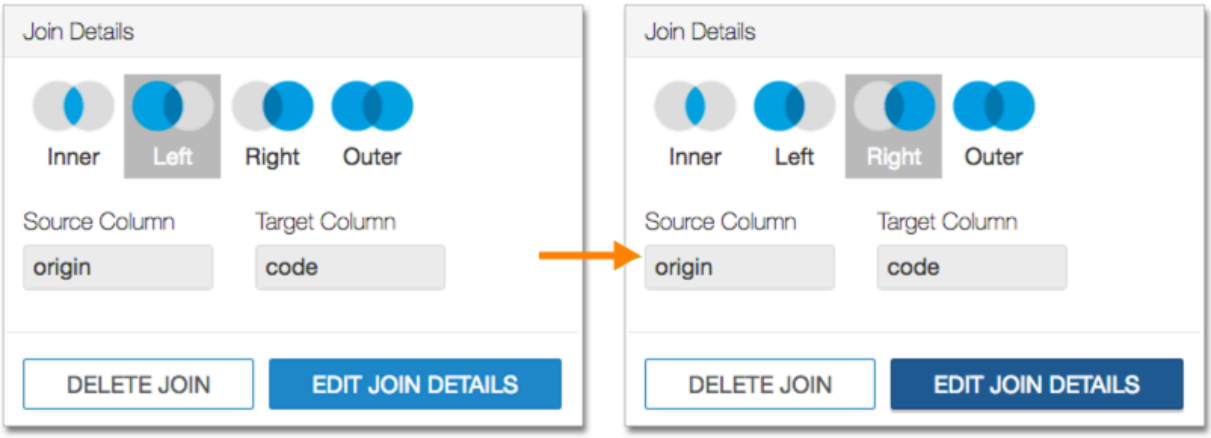

5. Click outside the Join Details modal window, or click Edit Join Details.
6. Click Save.

Editing join details

Learn how you can change the specifications of existing table joins in Cloudera Data Visualization.

About this task

To demonstrate how to create new data joins, we used dataset Flight Delays, based on data previously imported from a sample datafile

Procedure

1. Navigate to the Data Model page of the dataset.
2. Click EDIT DATA MODEL to edit the model.

3. Click the (link) icon beside the connection you want to edit.

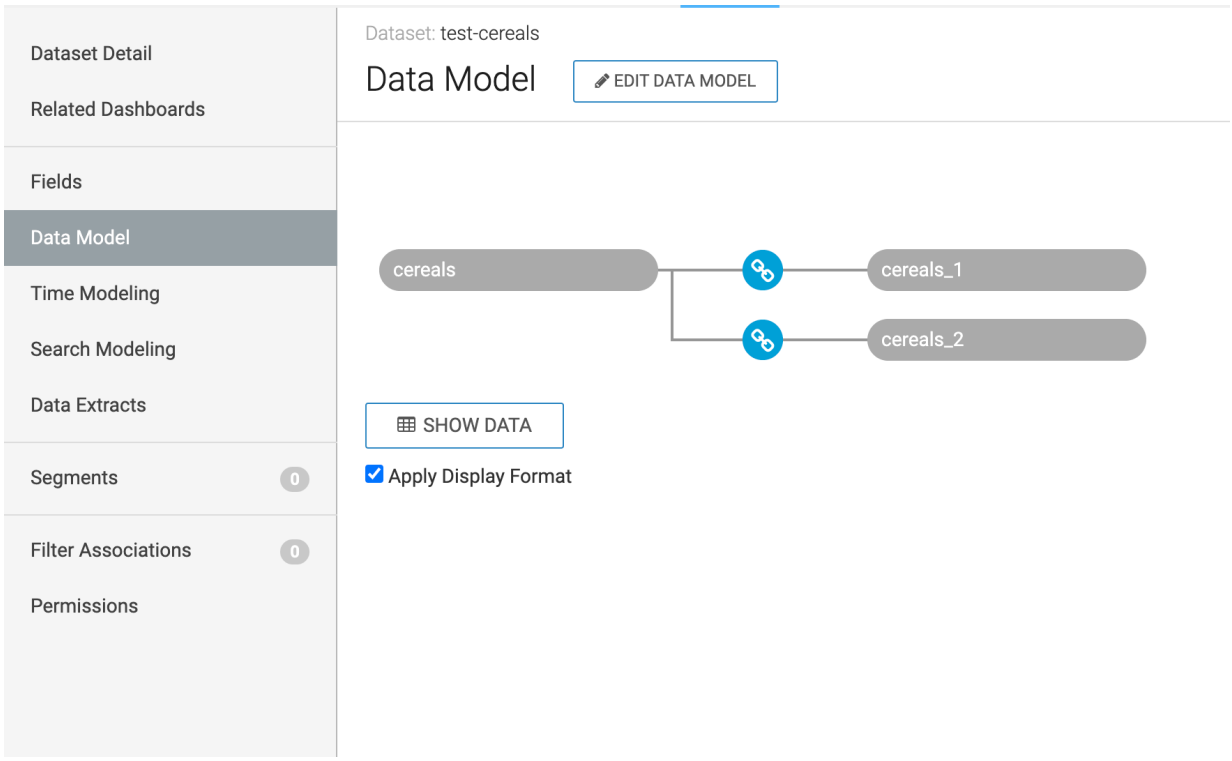

The Join Details modal window opens.

4. Click EDIT JOIN.

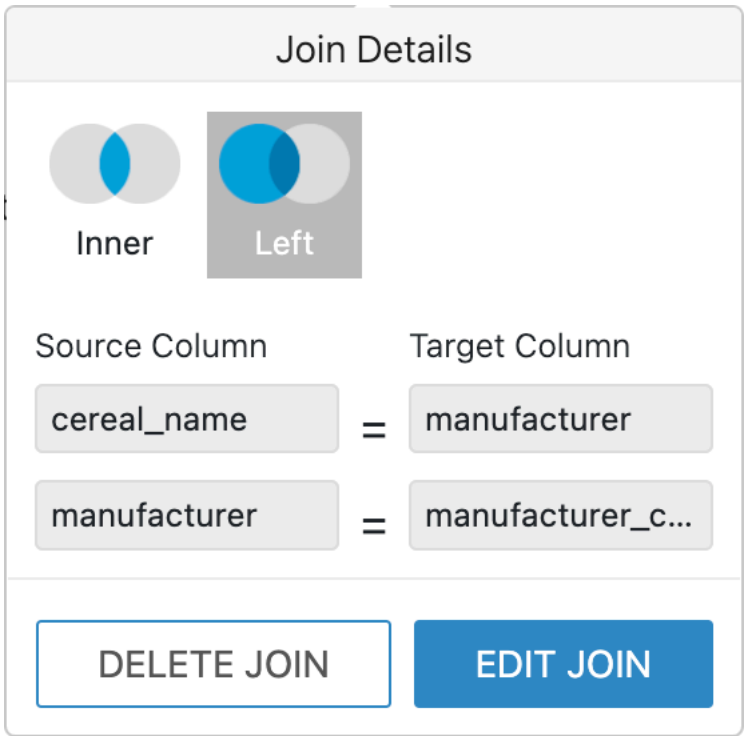

5. In the Edit Join modal window, the following options are available:

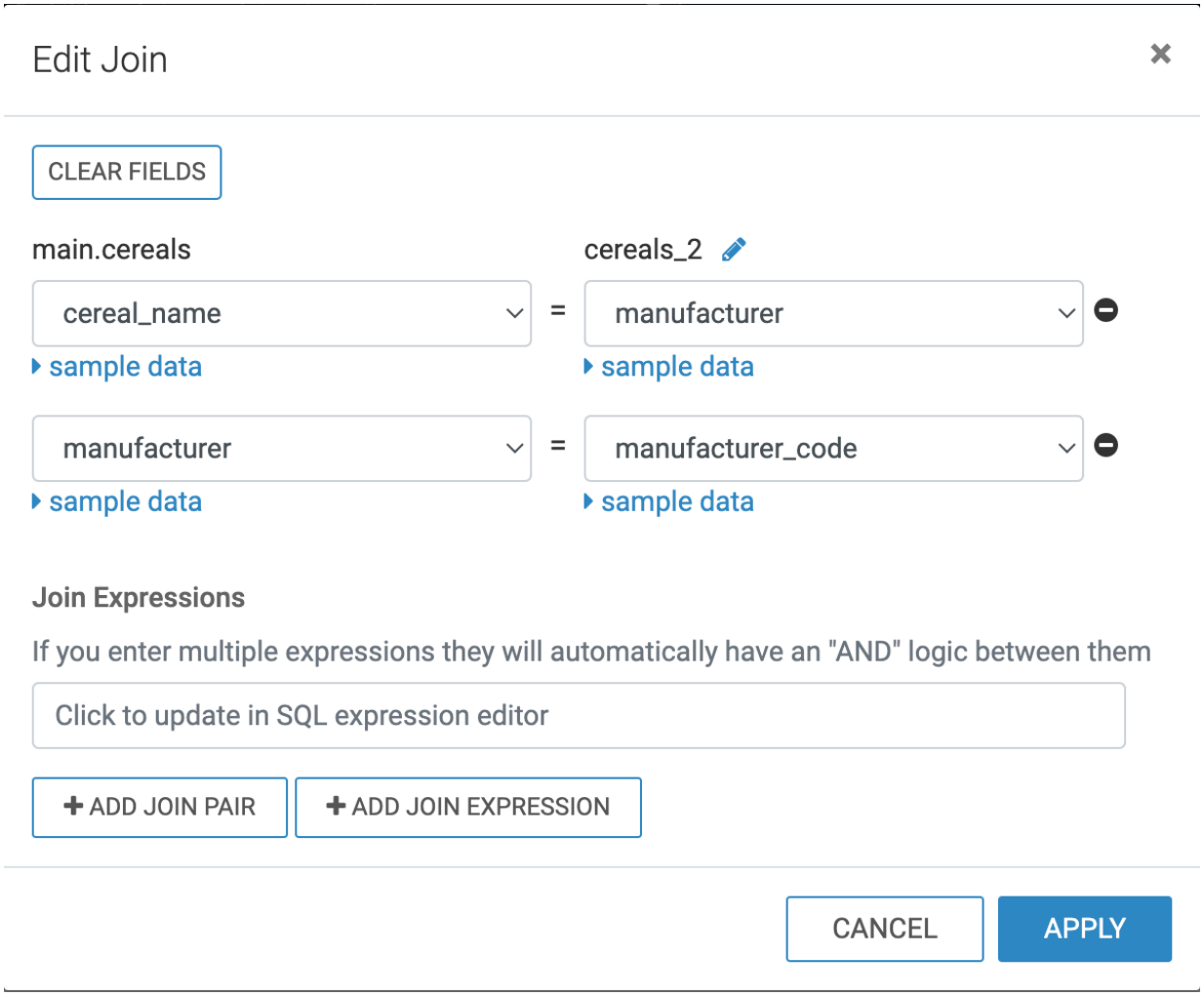

a. CLEAR FIELDS allows you to clear all already defined joins between the two tables.

b. The (pencil) icon allows you to edit the name of the joined table.

> **Important:** This action is only available when a join is created to the same table. In these cases, the table name is duplicated with a number appended to distinguish it. This alias is applied to both dimension and measure names within the joined dataset.

If you wish to modify this duplicated alias for clarity or consistency, you can edit the duplicated table name. When you provide a new alias, it is applied to the corresponding dimension and measure names of the dataset.

c. sample data allows you to preview the data from the tables. Click it again to hide the sample data.

d. The (minus) icon allows you to remove an existing join pair or an existing join expression.

e. Clicking the Join Expressions text box opens the Join Expression interface where you can specify or update a custom SQL expression that defines the join conditions.

f. ADD JOIN PAIR can be used to add another column connection between the same two tables.

g. ADD JOIN EXPRESSION can be used to add a join between the two tables based on a custom SQL expression.

Click APPLY to save the changes.

6. To add a a join expression and replace the original field:field join, perform the following steps in the Edit Join modal window:
   a. Remove the initial join between the two columns by clicking the (minus) icon.
   b. Under Join Expressions, click the text box to open the Join Expression interface, where you can specify or update the custom SQL expression that defines the join conditions.

      For example, you can enter the following expression to only show data where the amount of complex carbohydrates equals the amount of sugars, and the calorie content is greater than 250.

```
[Complex Carbohydrates Grams]=[Sugars Grams] AND [cereals_1 Calories] >
250
```

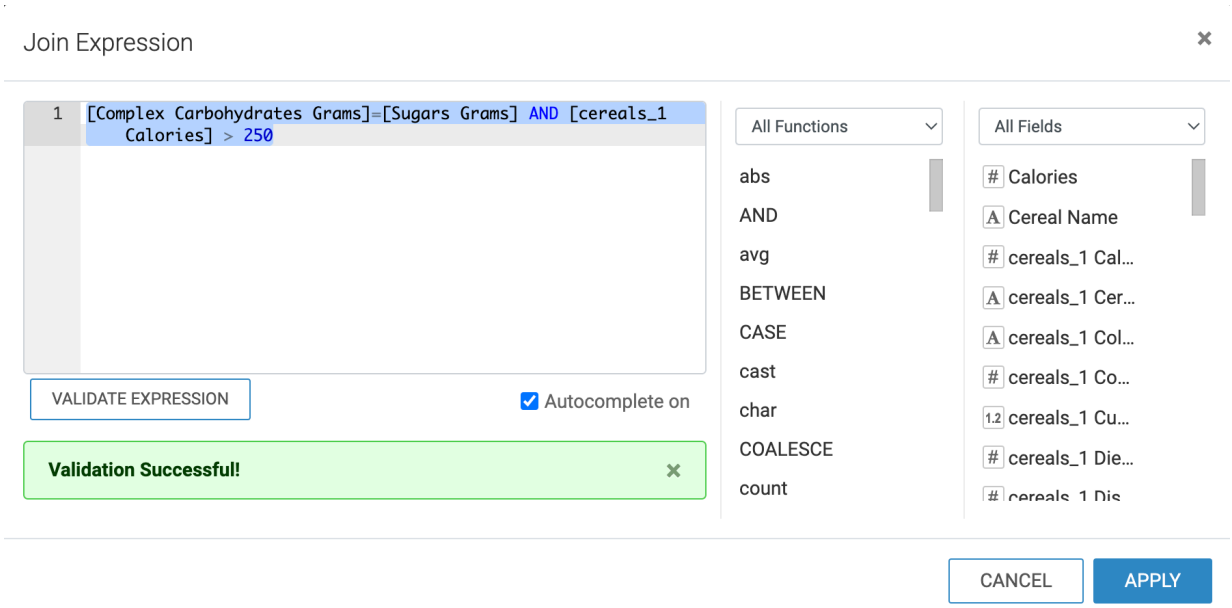

7. Click APPLY to save the expression and return to the Edit Join modal window.

8. In the Edit Join modal window, verify that the new join expression appears under Join Expressions and click APPLY.

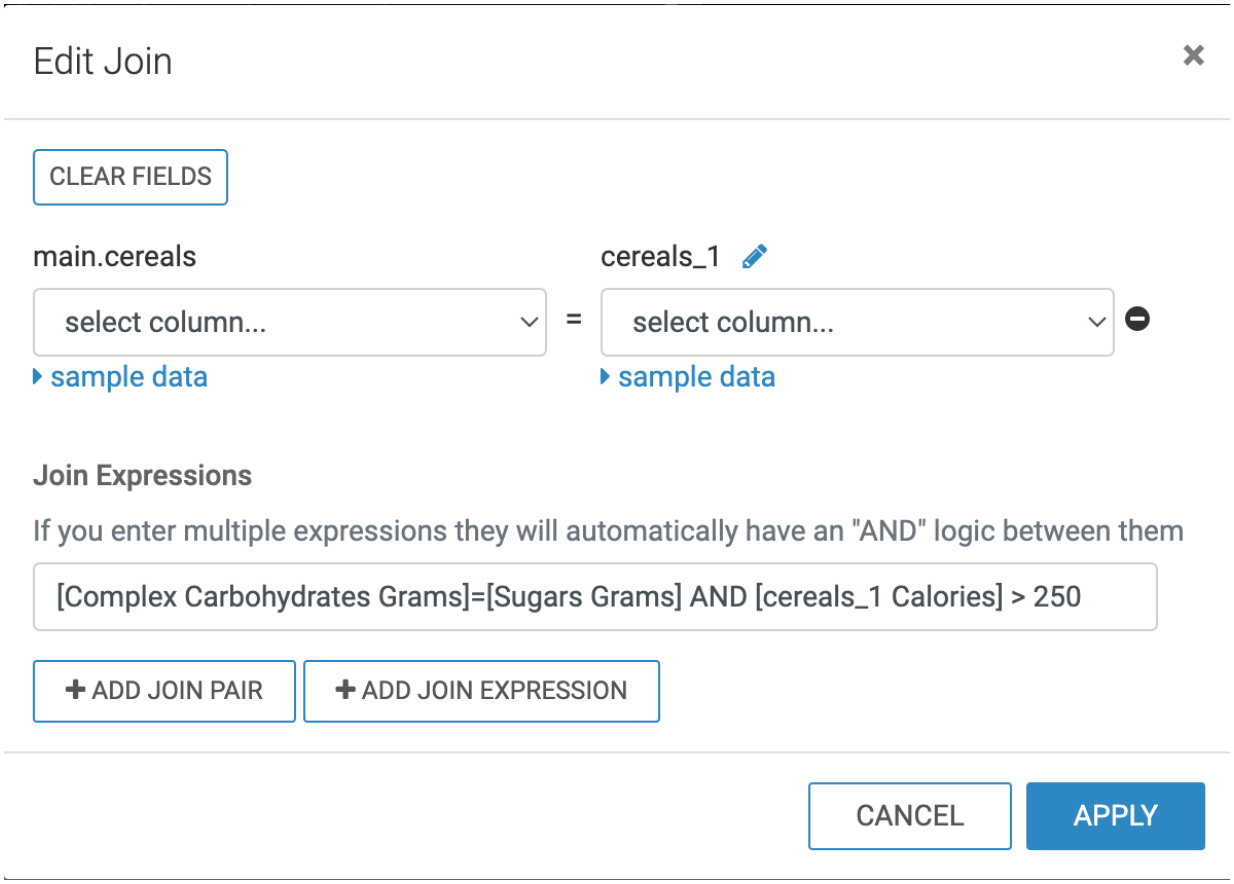

9. To revert this change prior to saving, click UNDO.
10. Click SAVE.

Deleting a join

Learn how you can remove a join that exists between tables in your dataset.

Procedure

1. Navigate to the **Data Model** page of the dataset.
2. Click EDIT DATA MODEL to enable editing of the data model.

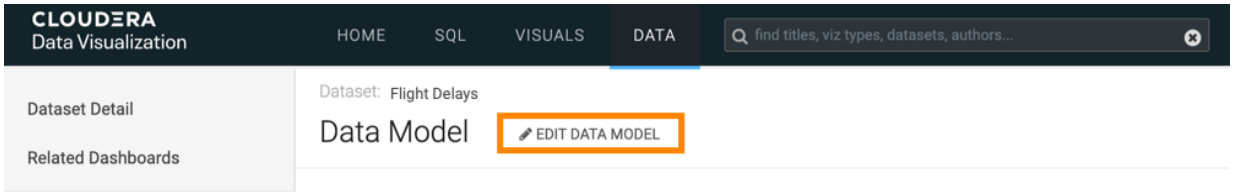

3. Click the (link) icon representing the connection you want to delete.
   The **Join Details** modal window opens.

4. Click DELETE JOIN.

   For example, if you delete the cancellation_code connection, you can see that after the deletion, this table no longer appears in the Data Model.

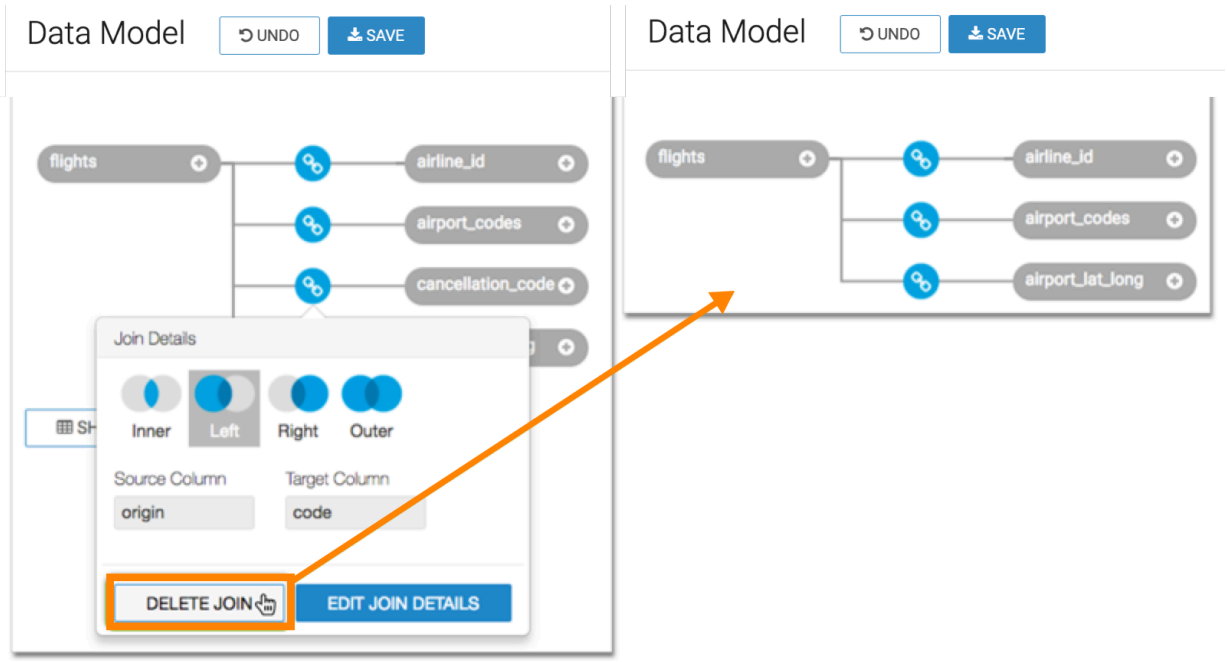

5. Click UNDO if you want to revert the change before saving.
6. Click SAVE to apply your changes.

Applying field display format on sample data

Learn how you can test field display formats you have configured.

Procedure

1. Navigate to the Data Model page of the dataset.

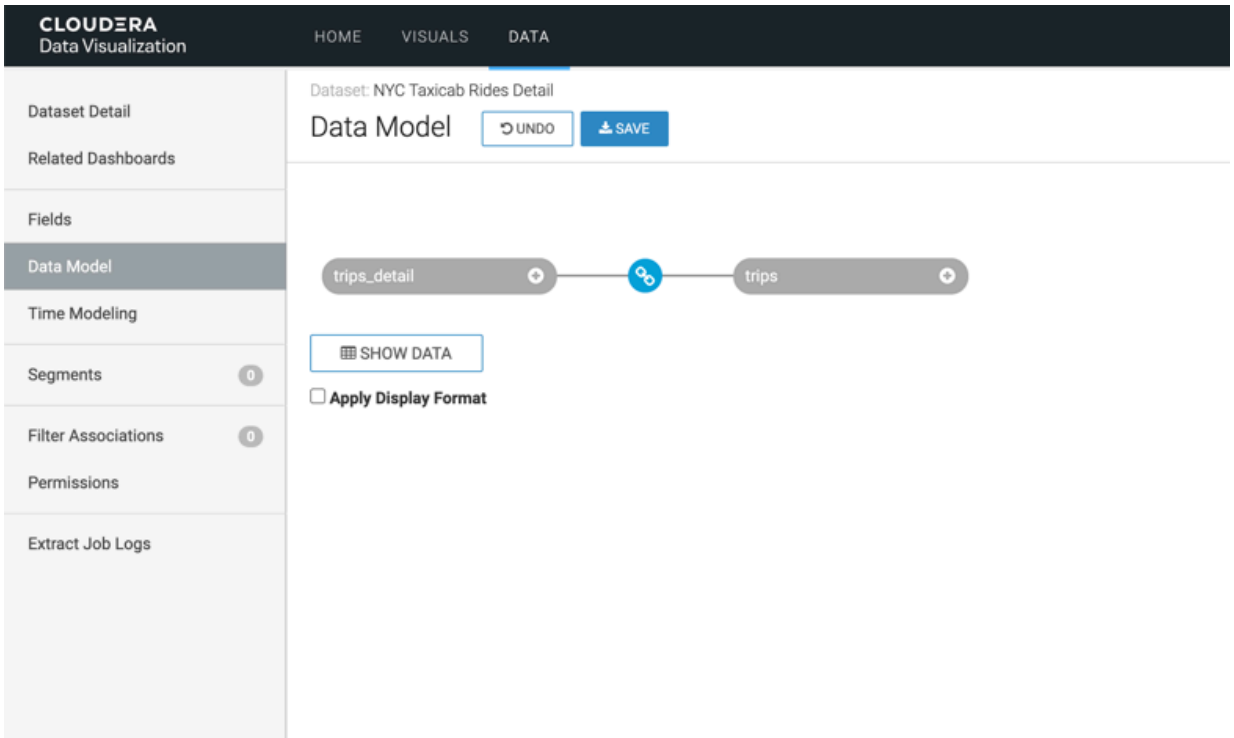

2. Select Apply Display Format.
   Selecting/deselecting the checkbox applies or removes the formatting without refetching the data.

   Note: The display format checkbox setting will be remembered.

   For information on how to configure field display at the dataset level, see *Changing the field display format*.

3. You can click SHOW DATA to view a sample of your data model.

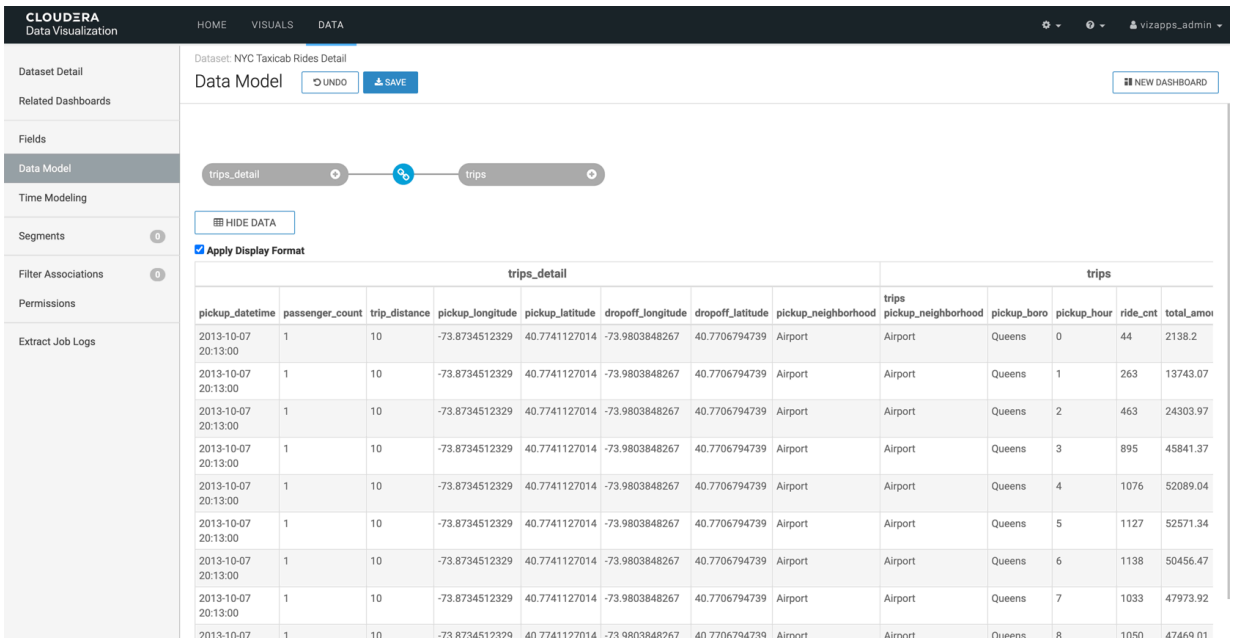

Related Information

Changing the field display format

Working with data profiling

Learn how you can profile your datasets. Dataset profiling provides a high-level overview of your data, offering insights into structure, content, and quality. This feature allows you to quickly assess the suitability of your data for analysis, and helps you identify the appropriate visualizations and any necessary data transformations.

About this task

With dataset profiling, you can access detailed statistics for dimension or measure in your dataset, including data type distributions, missing values, and unique values, as well as more in-depth information such as column distribution charts and statistics. These insights help you better understand the quality and characteristics of your data, ensuring it is suitable for the intended analysis or visualization.

Before you begin

- You need to enable the data profiling feature in Site Settings Data Enable Data Profiling . For more information, see Managing data related site settings.

Procedure

1. Click DATA on the main navigation bar.

   The Data view appears, open on the Datasets tab.
2. Find the dataset that you want to profile, either by scrolling through the list or using the search function.
3. Click the dataset you want to examine.

   The dataset side navigation pane opens for the selected dataset, displaying the Dataset Detail page.
4. Click Dataset Profiling in the left-side navigation panel.

   The Dataset Profiling view appears, with the option to select fields from the dimensions and measures of the dataset.

5. Select the fields that you want to examine and click Explore.

   Cards with various statistics and visual representations of the selected fields are displayed.

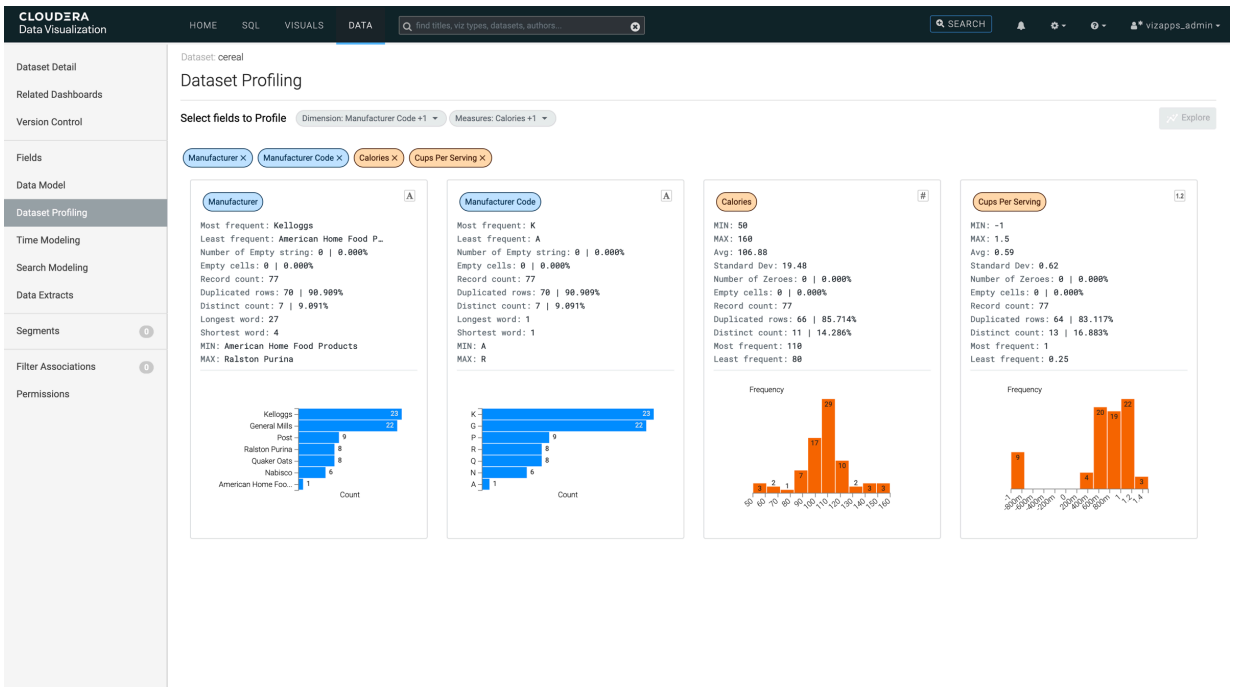

Tip:

- Click any selected data field to navigate directly to the corresponding card.
- To remove a card, click ✕ located in the field label.
- The data type used in the specific dimension or measure is shown in the top-right corner of the card.

The following information is shown for dimension values:

Most frequent

The string that appears the most often in the dataset, helping to identify the most common value.

Least frequent

The string that appears the least often in the dataset, helping to identify rare or unusual values.

Number of Empty string

The count of cells that contain an empty string, for example cells that are not null but contain no characters.

Empty cells

The number of cells that are completely empty, including both null values and empty strings.

Record count

The total number of records in the dataset.

Duplicated rows

The number of rows that are exactly identical, helping to identify redundant data. A high number of duplicates often signifies low variability in the dataset.

Distinct count

The number of unique values in the dataset, showing how varied the data is.

Longest word

The string with the highest number of characters in the dataset, providing insight into the potential size of values.

Shortest word

The string with the fewest number of characters, indicating the shortest data entry.

MIN

The minimum value in the dataset based on alphabetical order, useful for sorting or range analysis.

MAX

The maximum value in the dataset based on alphabetical order, helping to define the upper range of the data.

The count histogram for dimensions shows how many times each category appears, that is the number of occurrences (counts) of each category (dimension). This visualization is useful for understanding the quality, distribution, and structure of your dataset.

The following information is shown for measure values:

MIN

The smallest numeric value in the dataset, showing the lower bound of your data.

MAX

The largest numeric value in the dataset, showing the upper bound of your data.

Avg (Average)

The sum of all numeric values divided by the total number of records, giving an overall sense of the central value.

Standard Dev

Standard deviation that shows how spread out the numbers are in the dataset. A higher value indicates more variability in the data and a lower value indicates that the data points are closer to the average.

Number of Zeroes

The count of records where the value is exactly zero, indicating gaps or null-equivalents in the data.

Empty cells

The number of cells that are completely empty, including both null values and missing values.

Record count

The total number of records in the dataset.

Duplicated rows

The number of rows where all values are identical, helping to identify redundant data. A high number of duplicates often signifies low variability in the dataset.

Distinct count

The number of unique values in the dataset, showing how varied the data is.

Most frequent

The numeric value that appears the most often in the dataset, helping to identify the most common value.

Least frequent

The numeric value that appears the least often in the dataset, helping to identify rare or unusual values.

The frequency histogram for measures shows the distribution of numeric values (measures) within specific ranges (buckets), giving insight into their spread and common ranges.

Working with data extracts in Cloudera Data Visualization

Data extracts are saved subsets of data that you can use for data discovery and analytics.

You create an extract to reduce the total amount of data you work with by selecting certain dimensions and measures. With the help of data extracts you can manage the analytical capabilities, performance, concurrency and security of data access in your system.

The following diagram shows you the workflow of building visualizations on top of data extracts:

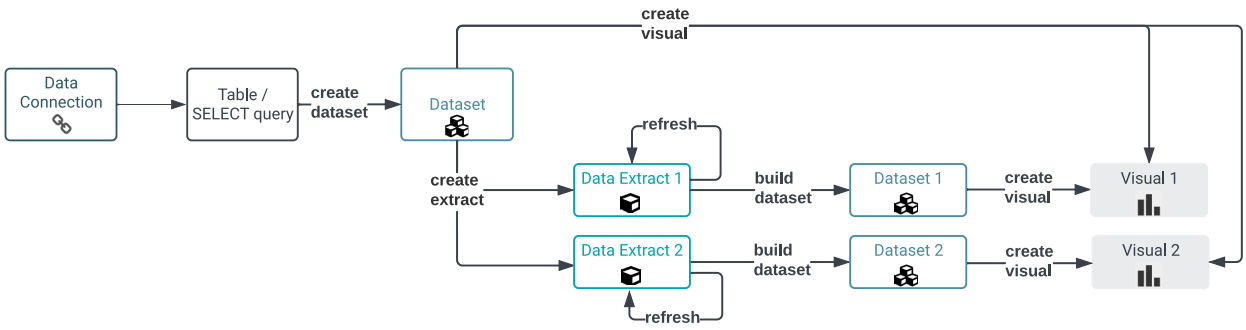

Creating a data extract

Learn how you can extract data from a dataset to a table in the same or a different data connection.

Before you begin

- You need to enable the data extract feature in Site Settings Data Enable Data Extracts . For more information, see Managing data related site settings.
- You need the following privileges to work with data extracts:
  - Manage dataset right for the source dataset
  - Manage AVs/extracts right for source and target data connections.

About this task

Follow these steps to create a data extract:

Procedure

1. Click DATA on the main navigation bar.

   The Data view opens, displaying the Datasets tab.
2. In the left navigation panel, select the connection on which your source dataset is defined.
3. Select the dataset you want to use as a source for the data extract.
4. Click Data Extracts in the left navigation panel.

   In the Data Extracts view, you can create data extracts and manage extracts that exist from this dataset.

5. Click CREATE DATA EXTRACT.

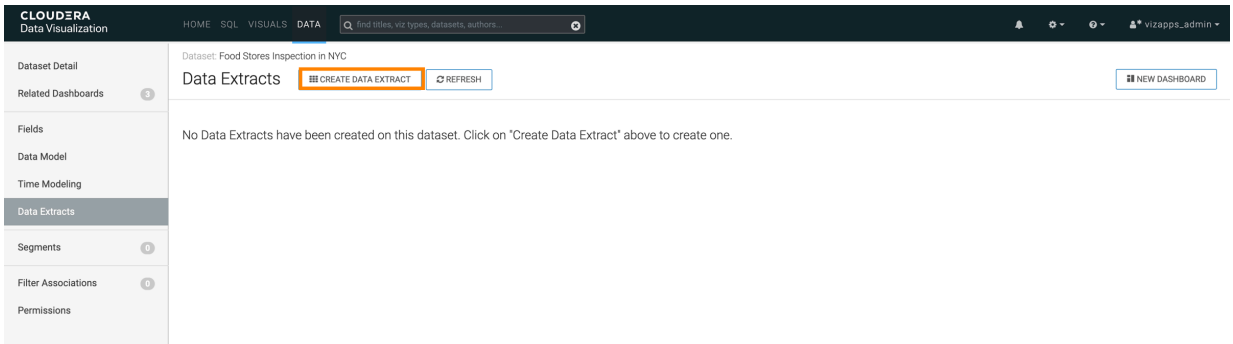

The CREATE DATA EXTRACT modal window opens.

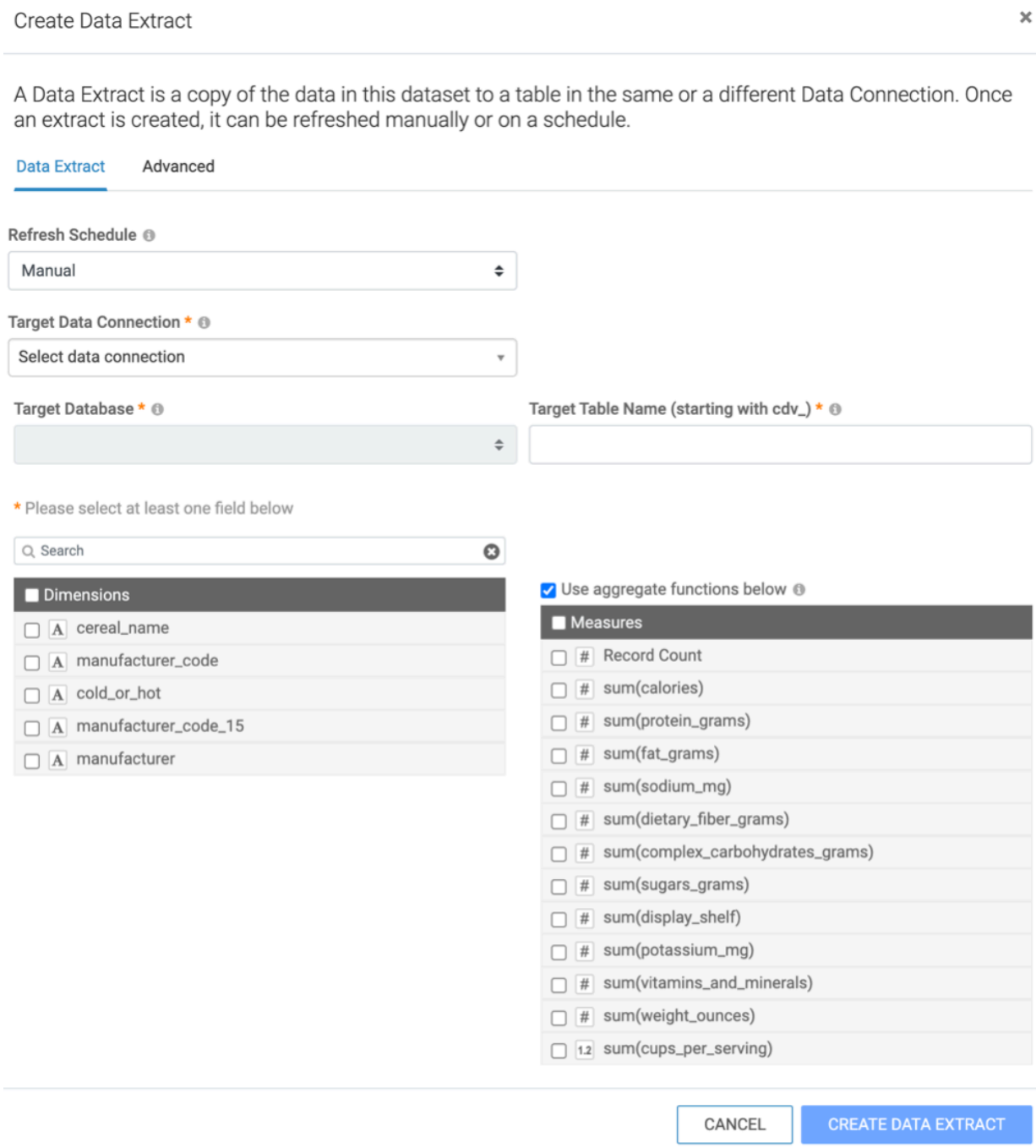

6. Select the frequency for the Data Extract refreshes.

   The default is Manual, but you can create custom schedules. For more information, see *Creating new schedules*.

7. Select the data connection where the Data Extract will be stored.

   It can be the same connection where the source dataset is or you can also choose a different connection.

   Note: In Cloudera Data Warehouse you cannot choose SQLite data connections as targets because those connections are read-only.

8. Select the database on the previously selected connection where you want to store the Data Extract.

9. Enter a target table name, starting with cdv_.

   Note: cdv_ is the default prefix for target table names. If you have admin privileges, you can change the prefix in the Advanced Site Settings by changing the value in DATA_EXTRACTS_TARGET_TABLENAME_PREFIX = "cdv_".

10. Select the dimension and measure fields depending on what data you want to store in your extract.

    If you want to use the selected measures as they are and you do not want the columns to be aggregated, deselect the Use aggregate functions below option.

    Note: You must select at least one dimension or measure field.

    When configuring a data extract, you can search for a column by name, as well as view the field types and complex data types available.

11. If you want to send an email confirmation about the extract's status, switch to the Advanced tab of the Create Data Extract modal window.

    Important: The Advanced tab is only visible if you enabled sending visuals and receiving extract refresh status messages through email in Site Settings.

    a) Select what event(s) you want to be notified about - success, failure or both.
    b) Add one or more email addresses as recipients.
    c) You can edit the default subject line of the notification email.

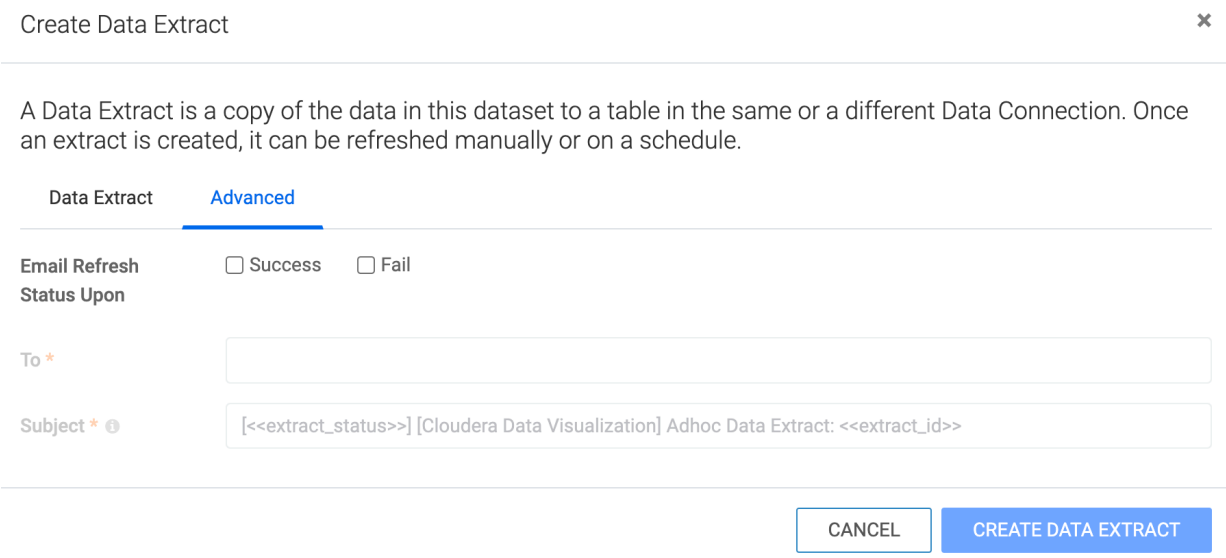

12. Click CREATE DATA EXTRACT after all required fields are completed.

    When the extract is created, you can refresh it manually or it will be refreshed on the schedule you defined in Step 5.

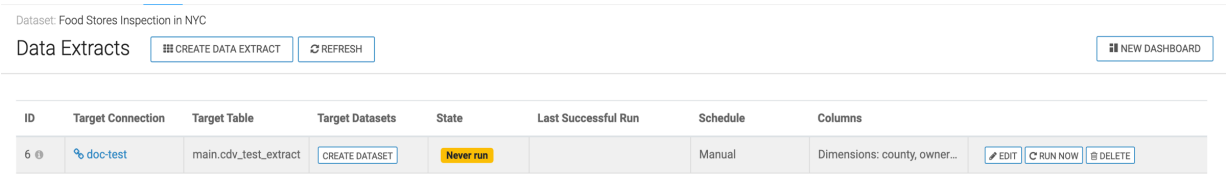

Note: After the source dataset becomes associated with the data extract, a new tag appears in the Datasets list showing Extract Source.

Results

The extract is created, and its state shows as Never run.

What to do next

You must run the extract to populate the target table and allow building a dataset or dashboard on the target table.

Related Information

Creating new schedules

Exploring data extract details

Learn how you can examine the available information about your data extracts.

Procedure

1. Click DATA on the main navigation bar.
   The Data view opens, displaying the Datasets tab.
2. In the left navigation panel, select the connection where your source dataset is defined.
3. Select the dataset of your data extract.
4. Click Data Extracts in the left navigation panel to display the Data Extracts view.
   You can see the list of available data extracts with the following information shown:
   - ID
   - Target Connection
   - Target Table
   - Target Datasets
   - State
   - Last Successful Run
   - Schedule
   - Columns

   You can also access the following action buttons:
   - Edit
   - Run now
   - Delete

Running a data extract

Learn how to run or refresh a data extract.

About this task

After creating a data extract, you must run it as an initial 'refresh' to populate the target table and allow building a dataset or dashboard on the target table. When the original data changes, becomes stale or invalid, you can manually refresh the extracted data by running your data extract.

Procedure

1. Click DATA on the main navigation bar.
   The Data view opens, displaying the Datasets tab.
2. In the left navigation panel, select the connection where your source dataset is defined.
3. Select the dataset of your extract.

4. Click Data Extracts in the left navigation panel.
5. Select the data extract you want to run/refresh.
6. Click RUN NOW.

   A confirmation modal window opens where you have to confirm the action.
7. Click RUN EXTRACT.

   A green success message is displayed and while the refresh is in progress, the state of the extract shows as Running. When the refresh is successfully completed, the state of the extract changes to Success.

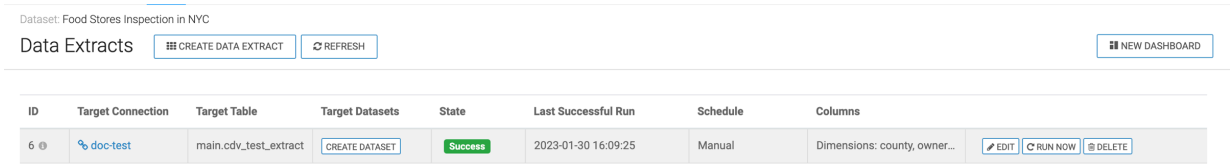

You can check out the extract refresh job logs on the Job Logs tab of the Jobs page. For more information, see *Job log*.

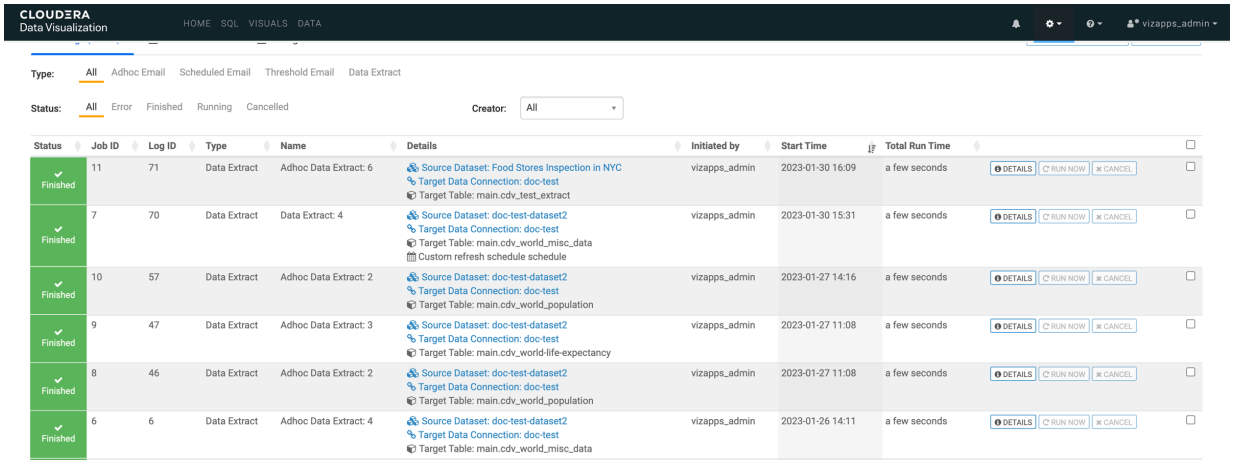

You can configure extracts to be automatically refreshed, and you can define a schedule for the automatic refreshes to update the data included in the extract.

If you set a refresh schedule for the extract to run, you can see the job logs on the Scheduled Jobs tab of the Jobs page. For more information, see *Scheduled jobs*.

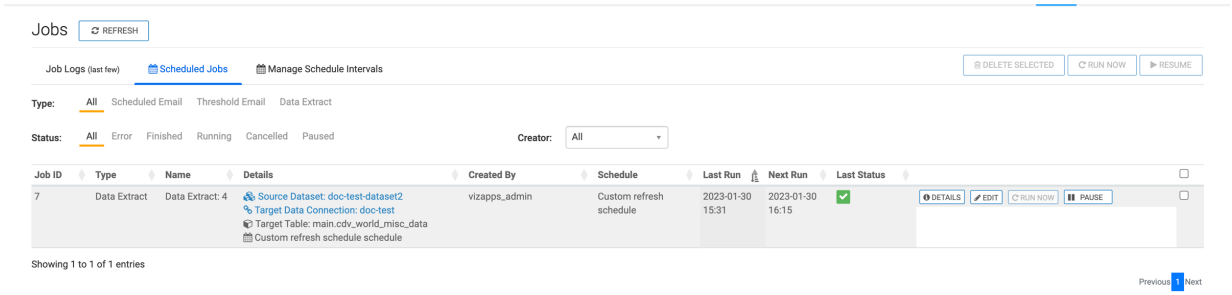

You can edit an existing schedule in the Edit Data Extract modal window from the Data Extracts view. For more information, see *Editing a data extract*. Or you can also update an existing schedule in the Manage Schedule Intervals interface. For more information, see *Changing schedule intervals*.

Related Information

Job log

Scheduled jobs

Editing a data extract
Changing schedule intervals

Creating a dataset from a data extract

Learn how to create a second dataset built on the extracted data, which is used as the dataset for building visuals.

Procedure

1. Click DATA on the main navigation bar.
   The Data view opens, displaying the Datasets tab.
2. Click Data Extracts.
3. Find the data extract that you want to use for the new dataset.
4. Click CREATE DATASET.

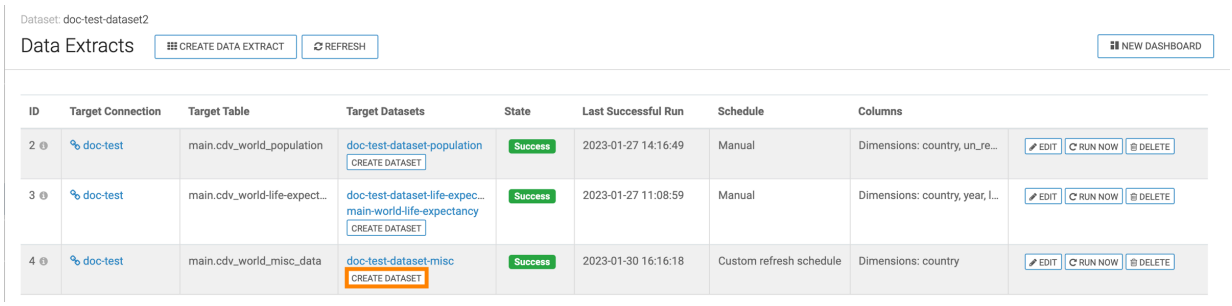

5. Add a name in the Dataset title field and click CREATE.

Note: After the dataset becomes associated with a data extract, a new tag appears in the Datasets list showing Extract Target.

Editing a data extract

Learn about the option to edit the data extract after it has been created.

Procedure

1. Click DATA on the main navigation bar.
   The Data view opens, displaying the Datasets tab.
2. Click Data Extracts.

3. Select the data extract you want to edit and click EDIT.
   The Edit Data Extract modal window opens.

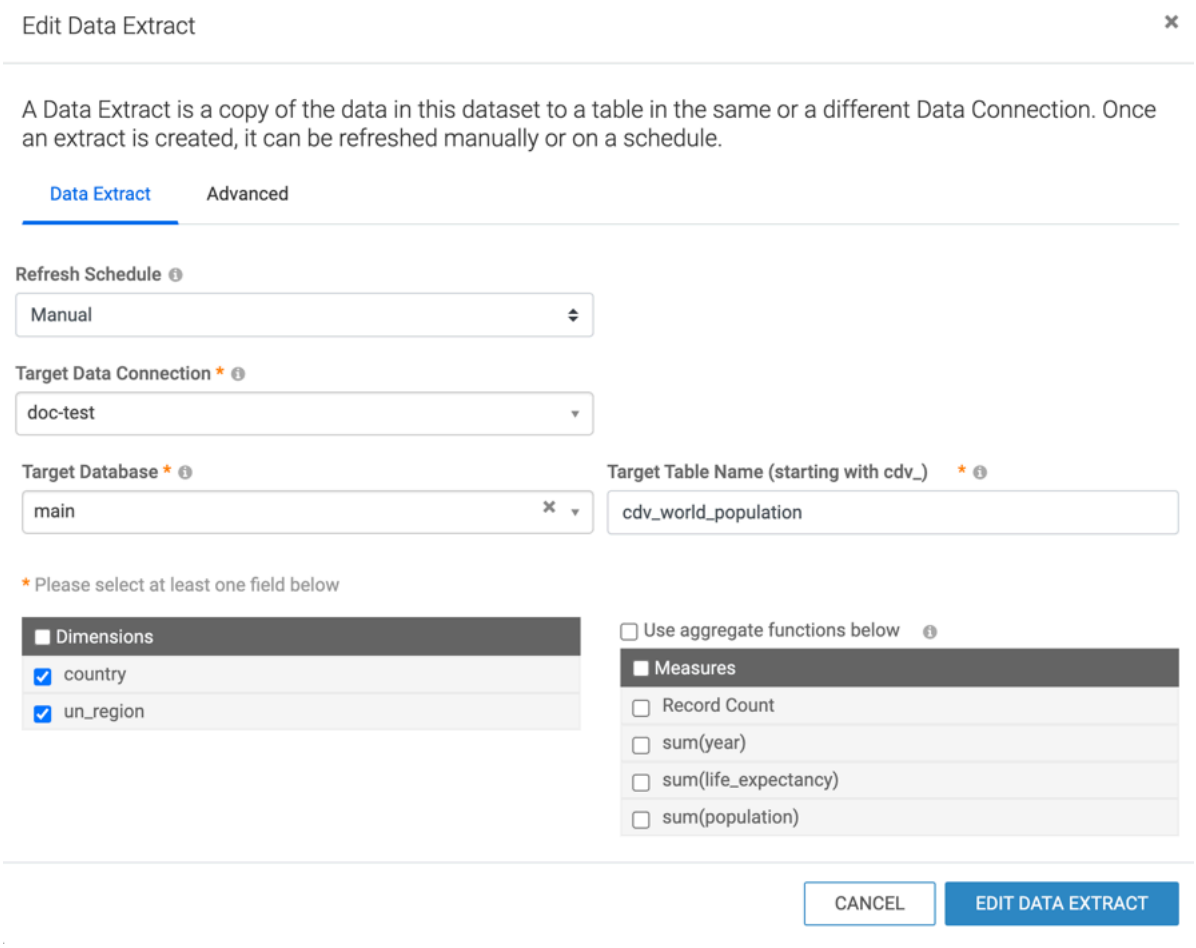

4. Make changes in the available fields.
   - You can set a Data Extract refresh schedule.

     Note: Editing the refresh schedule of a data extract does not delete previous jobs or logs. Instead, the schedule for the existing job is simply updated, ensuring that all historical data remains intact.
   - You can change the target data connection and database.
   - You can update the name of the target table.
   - You can add or remove dimension and measure columns.

     Important: If you change the target data connection, target database, target table name or target table fields the target table will be deleted and a new one will be created.
5. If you want to send an email confirmation about the extract's status, switch to the Advanced tab of the Edit Data Extract modal window.

   Important: The Advanced tab is only visible if you enabled sending visuals and receiving extract refresh status messages through email in Site Settings.

   a) Select what event(s) you want to be notified about - success, failure or both.
   b) Add one or more email addresses as recipients.
   c) You can edit the default subject line of the notification email.
6. Click EDIT DATA EXTRACT to save your changes.

Deleting a data extract

Learn how you can delete a data extract.

About this task

For various business reasons or because the extract has become invalid, you may choose to delete an existing data extract.

Procedure

1. Click DATA on the main navigation bar.

   The Data view opens, displaying the Datasets tab.
2. Click Data Extracts.
3. Select the data extract you want to edit and click DELETE.

   A confirmation modal window opens where you have to confirm the action.

   Important: Deleting an extract does not automatically remove the associated target table. The confirmation modal offers you an option to drop the target table when deleting the extract. If you want to delete the target table, mark the checkbox.

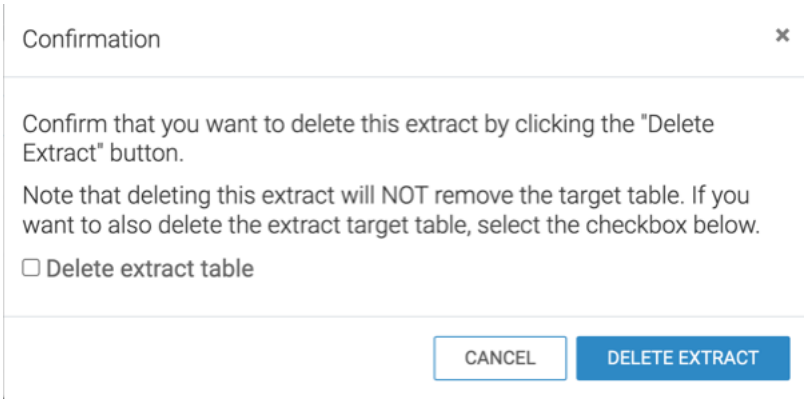

4. Click DELETE EXTRACT.

   Note: Deleting an extract does not delete the target table or any datasets that were created on the extract target.

Cloudera Data Warehouse Public Cloud

Cloudera Data Warehouse Auto-scaling

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Virtual Warehouse auto-scaling....4
Hive VW concurrency auto-scaling....4
Hive VW isolation auto-scaling....7
Impala VW auto-scaling....8

Workload Aware Auto-Scaling in Impala.... 10
How workload-aware autoscaling works....12
Workload Aware Auto-Scaling tuning....13

About the Impala Autoscaling Dashboard....14
Accessing the Impala Autoscaling Dashboard....15

Virtual Warehouse auto-scaling

The way Hive Virtual Warehouse performs auto-scaling is different from the Impala Virtual Warehouse. Learning how the auto-scaling works helps you choose the type of Virtual Warehouse and correctly configure the auto-scaling to handle your jobs.

Key features describes auto-scaling in Cloudera Data Warehouse and its benefits. You need to know specifics about how auto-scaling works in the type of Virtual Warehouse you choose. The number of queries running concurrently and the query complexity determine how Hive Virtual Warehouse auto-scaling works. A BI-type query does not require the same resources and scaling process as an ETL-type query. A complex ETL-type query requires intensive data scanning. You can configure the auto-scaling method used by the Hive Virtual Warehouse based on the number of concurrently running queries or the size of data scanned for a query. There are two methods of Hive Virtual Warehouse autoscaling:

- Concurrency for BI-type queries
- Isolation for ETL-type queries

Impala Virtual Warehouse auto-scaling is designed for concurrently running BI queries to make resources available for queued queries. Like Hive auto-scaling methods, coordinators and executors control Impala auto-scaling.

Workload Aware Auto-Scaling (WAAS) is available for Impala Virtual Warehouses.

Understanding executors

In AWS environments, an executor is an Elastic Compute Cloud (EC2) r5d.4xlarge instance for Hive and Impala compute nodes. For more information, see "AWS resources created for Data Warehouse". In Azure environments, an executor is an Azure E16 v3 instance. For more information, see "Virtual Warehouse sizing requirements for public cloud environments".

The default number of executors per executor group is based on the number of executor nodes contained by the original size of the Virtual Warehouse you configured during creation. For example, if you created the Virtual Warehouse and configured the MEDIUM size, which has 20 executor nodes, then each executor group contains 20 executors.

When you stop running queries, resources scale down, and executors are released. When all executor groups are scaled back, executors are idle, and after a configured period of idle time (AutoSuspend Timeout) elapses, the Virtual Warehouse is suspended. If a query is executed on a suspended Virtual Warehouse, the coordinator queues the query. When the auto-scaler detects a queued query, it immediately adds an executor group that can run the query on the Virtual Warehouse.

Hive VW concurrency auto-scaling

The Hive and Impala Virtual Warehouses use concurrency auto-scaling under certain conditions, but the auto-scaling implementation is different. You learn about the Hive Virtual Warehouse concurrency auto-scaling, which helps you better configure a Hive Virtual Warehouse for running BI queries.

Concurrency auto-scaling is designed for running BI queries. When the query load increases, auto-scaling increases. The query load grows proportionally with the number of concurrent queries and query complexity. When you run the BI query, HiveServer generates a preliminary query execution plan.

HiveServer locates an available query coordinator in the Virtual Warehouse to handle the query. The query coordinator generates the final query plan that distributes query tasks across available executors for execution. Each query coordinator can send query tasks to all query executors in the executor group, as shown in the following diagram of an XSmall Virtual Warehouse, for example:

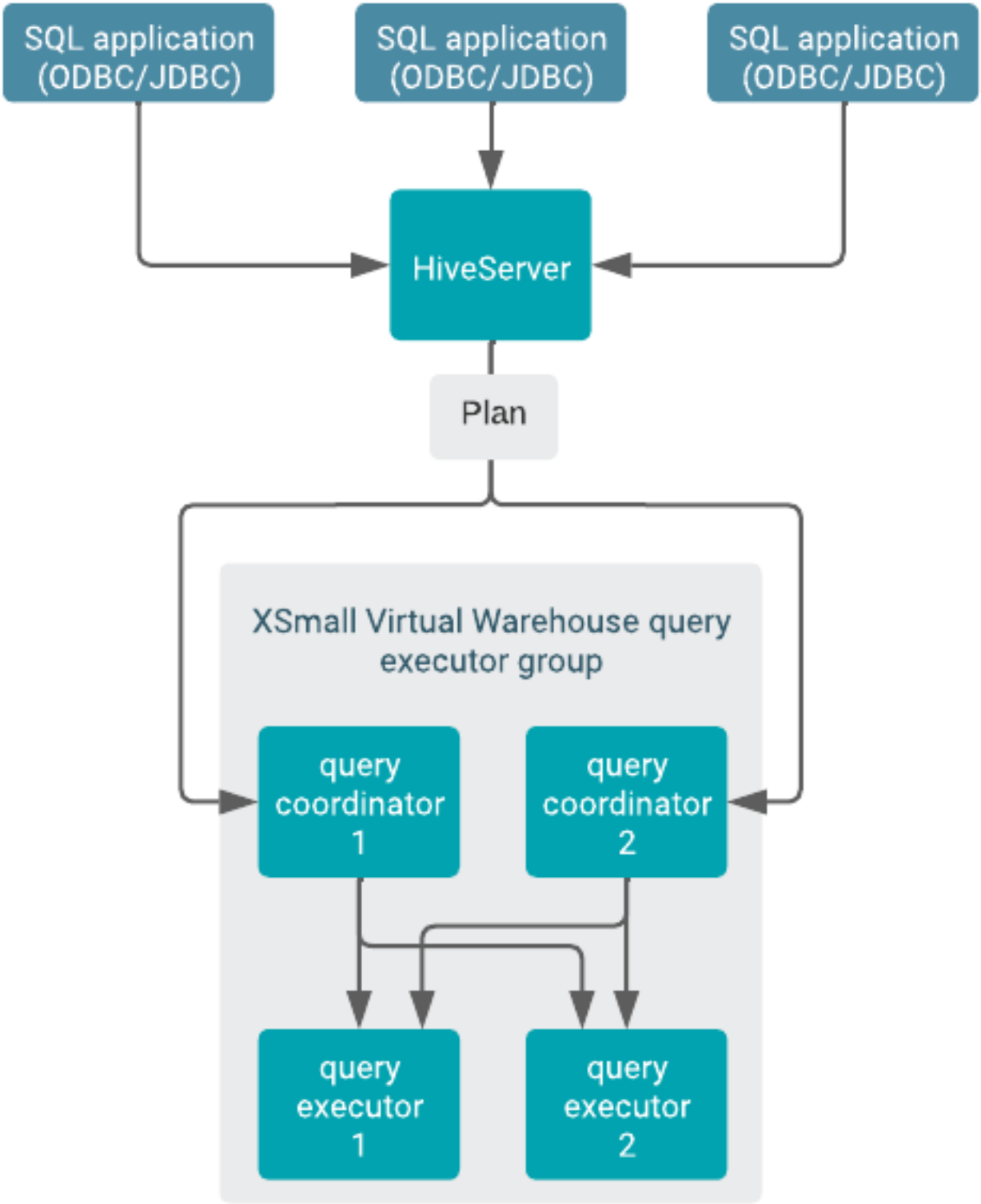

A single query on an idle Virtual Warehouse does not cause auto-scaling. The query uses the resources available at the time of the query submission. Only subsequent queries can cause auto-scaling.

Auto-scaling occurs when either the headroom, or wait time, exceeds the threshold you configure. Additional query executor groups are added to do the scaling, as shown in the following diagram.

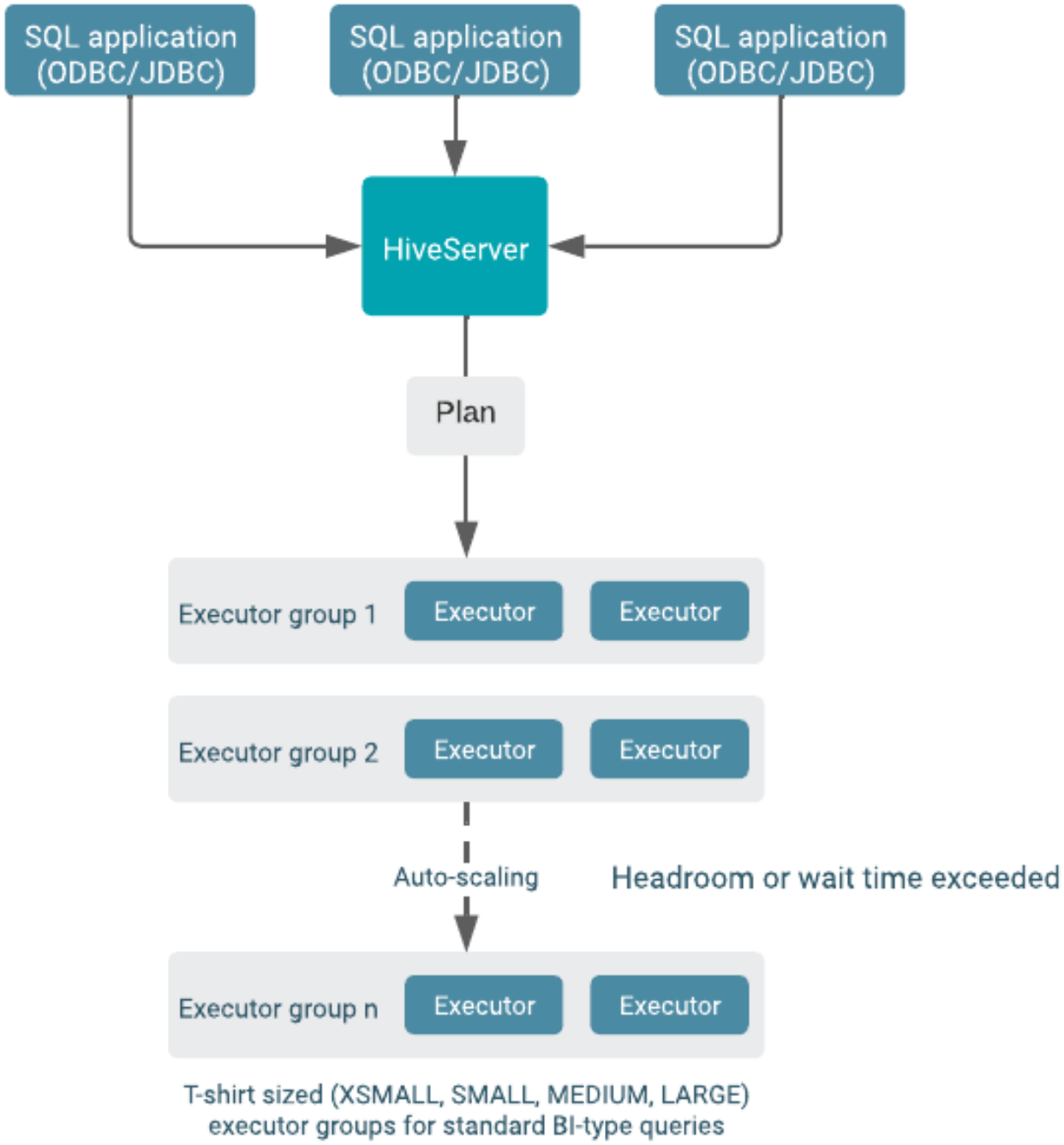

The number of simultaneously running queries are equal to the number of query coordinators. A query executor can run 12 query fragments at the same time. The size of executor groups is determined by the size you choose when you create the Virtual Warehouse (XSMALL, SMALL, MEDIUM, or LARGE). A single query is always contained within a single executor group and never spans multiple executor groups. The throughput for an individual query is determined by the original size of the warehouse.

The following diagram shows the same basic auto-scaling scenario as the diagram above but from a different perspective. In this diagram the query coordinator perspective, instead of the executor group, is shown.

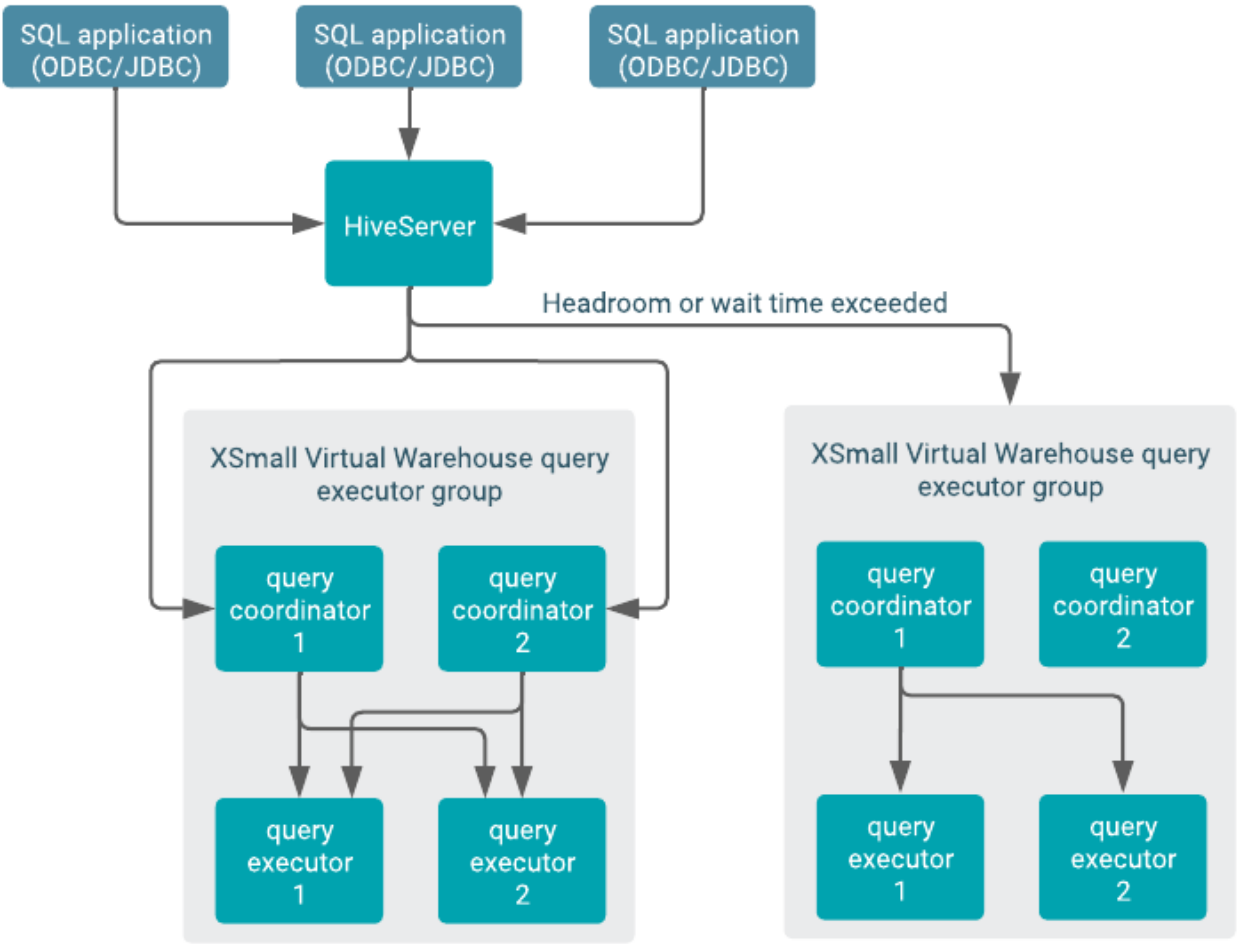

There are as many query coordinators as executors, but there is not a one-to-one correspondence. Instead, each query coordinator can interact with all executors. However, one query coordinator can only orchestrate the execution of tasks for one query at a time, so the number of query coordinators determines the query parallelism limit.

Related Information

Tuning Hive LLAP Virtual Warehouses
Virtual Warehouse sizing requirements
Virtual Warehouse IP address and cloud resource requirements

Hive VW isolation auto-scaling

The Hive Virtual Warehouse in Cloudera Data Warehouse Public Cloud bases its auto-scaling on the size of data scanned for a query. Queries that scan amounts of data that exceed the hive.query.isolation.scan.size.threshold setting use the query isolation method of auto-scaling.

To run ETL-type queries, you turn on query isolation when you configure the Hive Virtual Warehouse. Scaling occurs as follows:

- If the query does not exceed the hive.query.isolation.scan.size.threshold, concurrency scaling occurs.
- If the query does exceed the hive.query.isolation.scan.size.threshold, query isolation scaling occurs.

HiveServer generates a preliminary query execution plan.

The Hive Virtual Warehouse adds a new dedicated executor group with the right number of executors to handle the query instead of using the number associated with the T-shirt size you configured. This isolated executor group is used only under the following conditions:

- You run a single ETL-type query.
- The query exceeds the hive.query.isolation.scan.size.threshold.

Otherwise, the concurrency method of auto-scaling is used.

The following diagram shows the isolation method of auto-scaling:

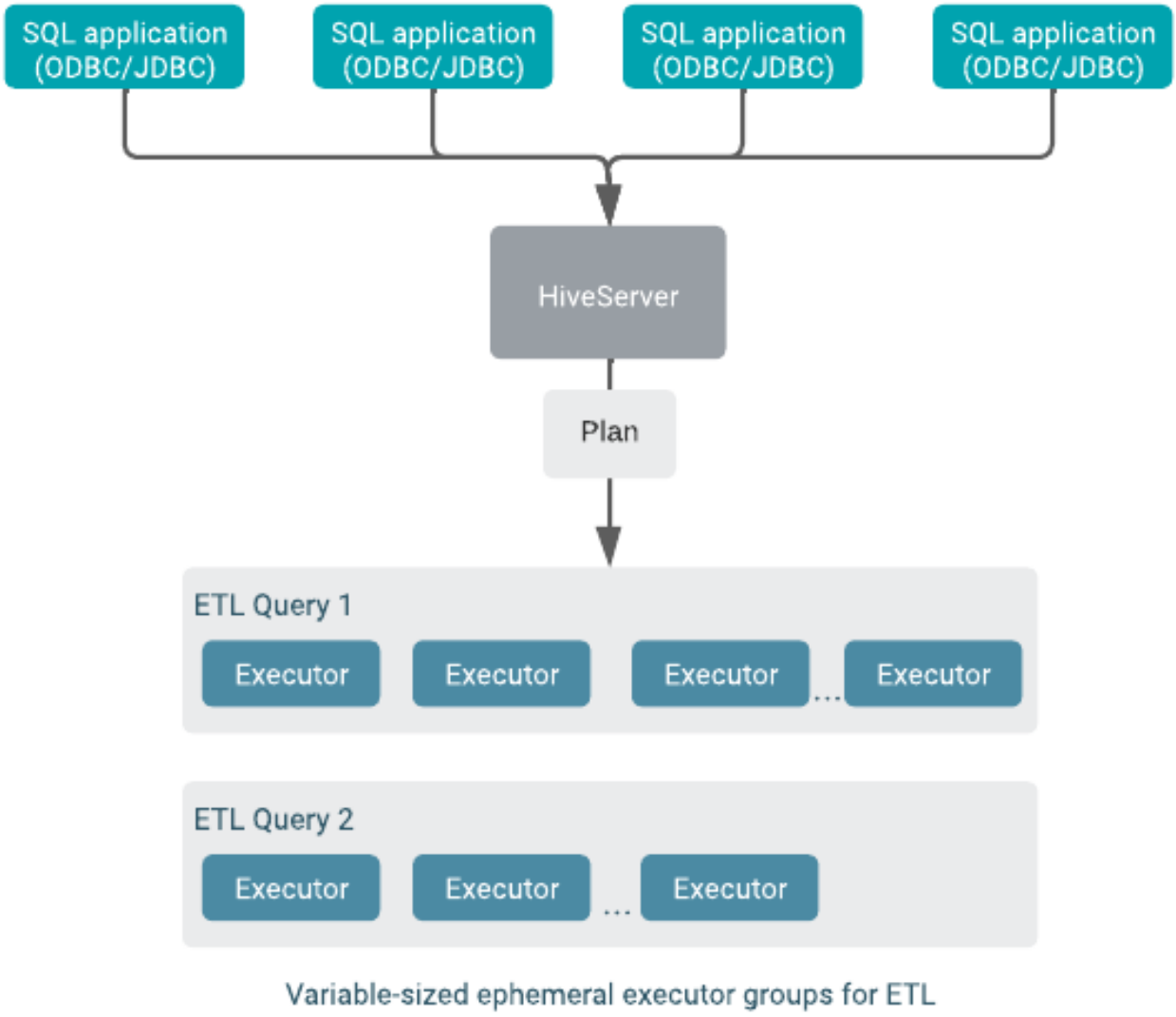

The variable-sized, ephemeral executor groups for ETL terminate after the query completes.

Impala VW auto-scaling

The Impala Virtual Warehouse, like the Hive Virtual Warehouse does concurrency auto-scaling, but in a different way. Learning how the auto-scaling works helps you choose the type of Virtual Warehouse and correctly configure the auto-scaling to handle your jobs.

The Impala Virtual Warehouse auto-scaling works with coordinators and executors to make resources available for queued queries, as shown in the following diagram:

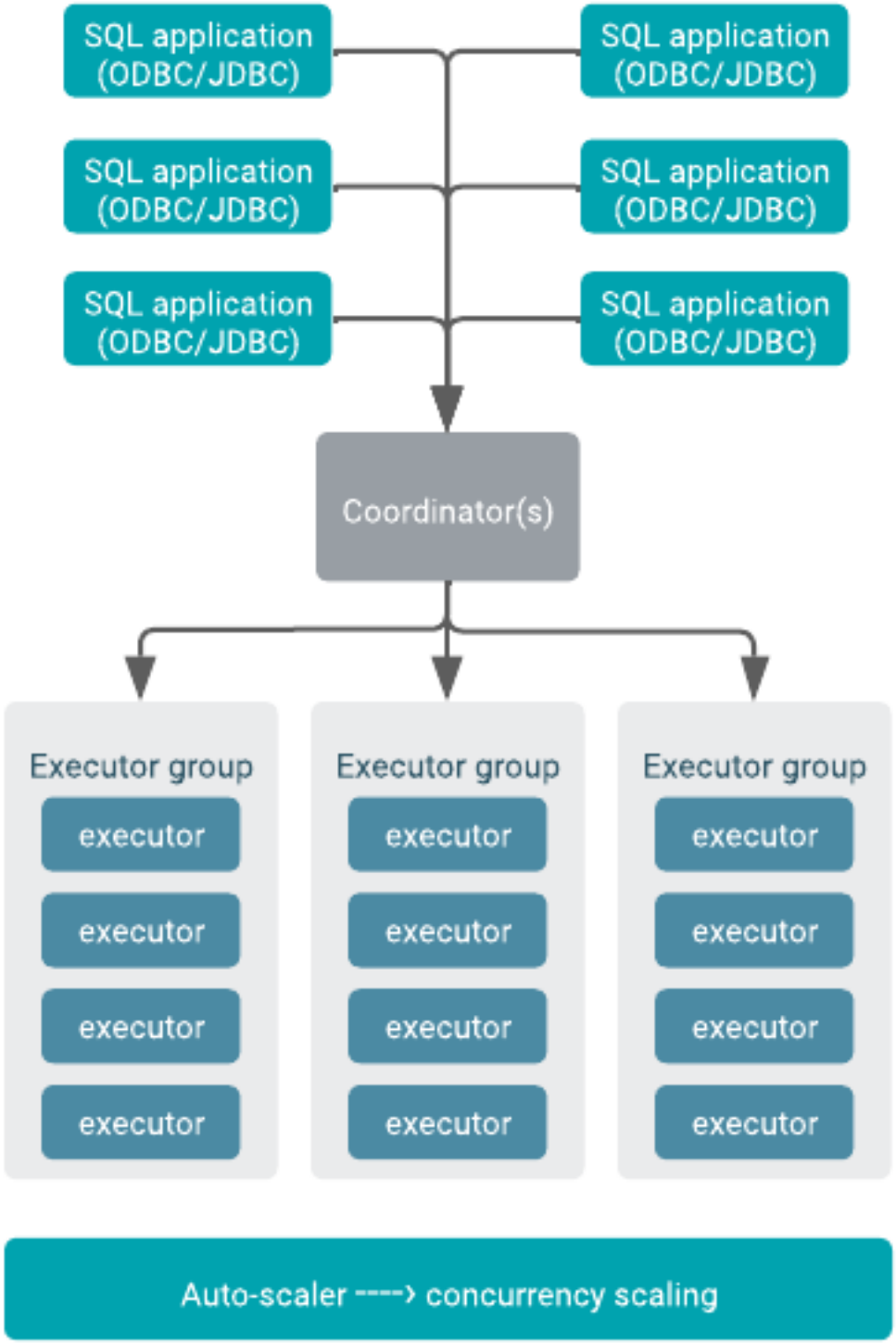

An auto-scaler process (a Kubernetes pod), shown at the bottom of the diagram, monitors Impala running in the Virtual Warehouse to determine the necessary resources. When the auto-scaler detects an imbalance in resources, it sends a request to Kubernetes to increase or decrease the number of executor groups in the Virtual Warehouse.

Executor groups execute queries. By default Impala Virtual Warehouses can run 3 large queries per executor group. For most read-only queries the default setting of 3 queries per executor group is sufficient. Executors can handle more queries that are simpler and that do not utilize concurrency on the executor.

Executor processes execute query fragments. Executor processes run on executor nodes, the unit of sizing for Virtual Warehouses. Each executor node runs one executor.

Related Information

Tuning Impala Virtual Warehouses
Virtual Warehouse sizing requirements
Virtual Warehouse IP address and cloud resource requirements

Workload Aware Auto-Scaling in Impala

Workload Aware Auto-Scaling (WAAS) allocates Impala Virtual Warehouse resources based on the workload that is running.

WAAS improves auto-scaling for the following types of workloads:

- Mixed or unknown workloads
- On-premises workloads you are migrating to Cloudera Data Warehouse
- Workloads running on a big cluster
- Workloads subject to queries with varying resource requirements
- Workloads you cannot easily segregate into different Virtual Warehouses
- Workloads clients must access from a single static endpoint

If you have already successfully segregated your workloads using multiple Virtual Warehouses, the benefits of enabling WAAS may be limited. The multiple Virtual Warehouse approach typically works well in this case.

Without WAAS, if you have mixed workloads, you size your Virtual Warehouse for the most resource-intensive queries, so the Virtual Warehouse can handle all your workloads.

With WAAS, if you have mixed workloads, you define constraints for the overall warehouse size. You can add multiple sizes of executor group sets to spin up if needed by the workload. Within the executor group set, executor groups spin up or shut down as warranted by the workload. Each executor group set is mapped to an admission control resource pool, discussed later.

An executor group set is analogous to a range of nodes assigned to a Virtual Warehouse of a particular size. Dynamic provisioning of mixed workloads uses two, or more, executor group sets instead of a one-size-fits-all executor group.

The following diagram depicts two executor groups sets in a WAAS Virtual Warehouse:

- group-set-small configured with an executor group size of 2 (small)
- group-set-large configured with an executor group size of 8 (large)

Within any executor group set are executor groups, all of the same size.

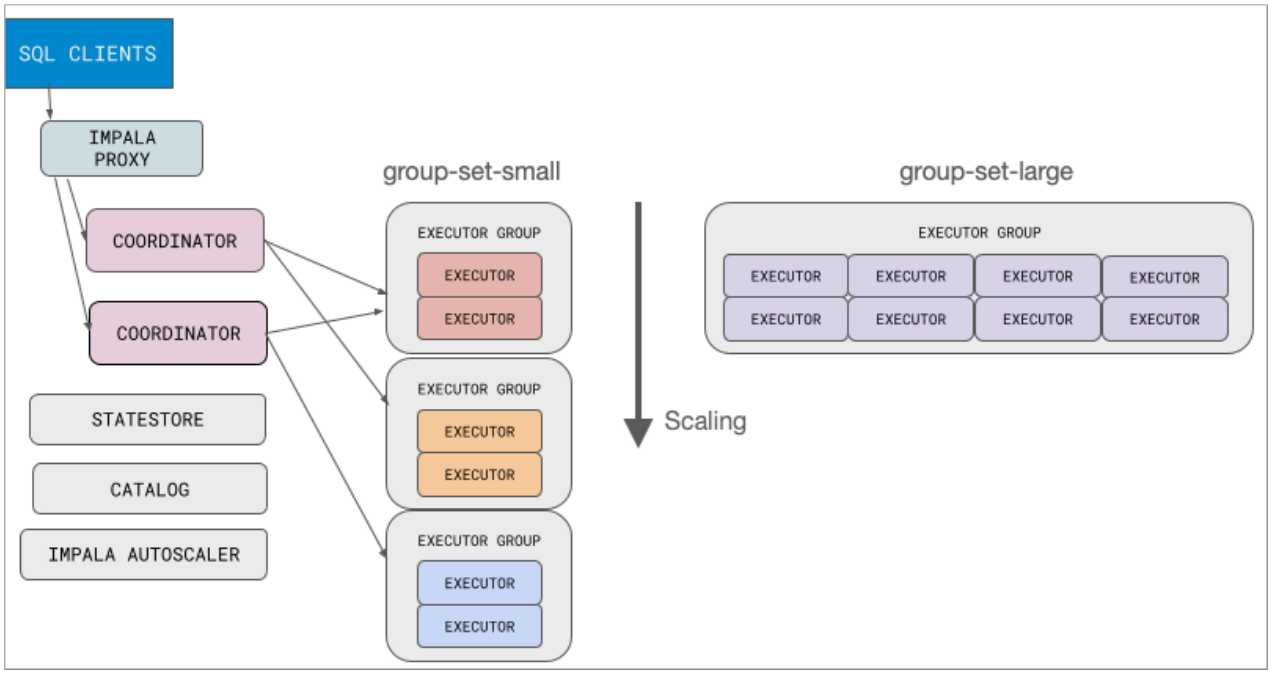

The following graphic depicts scaling that occurs when you run a large query and more capacity is needed. The executor group set is chosen based on the workload. The group-set-large spins up an executor group of size 8:

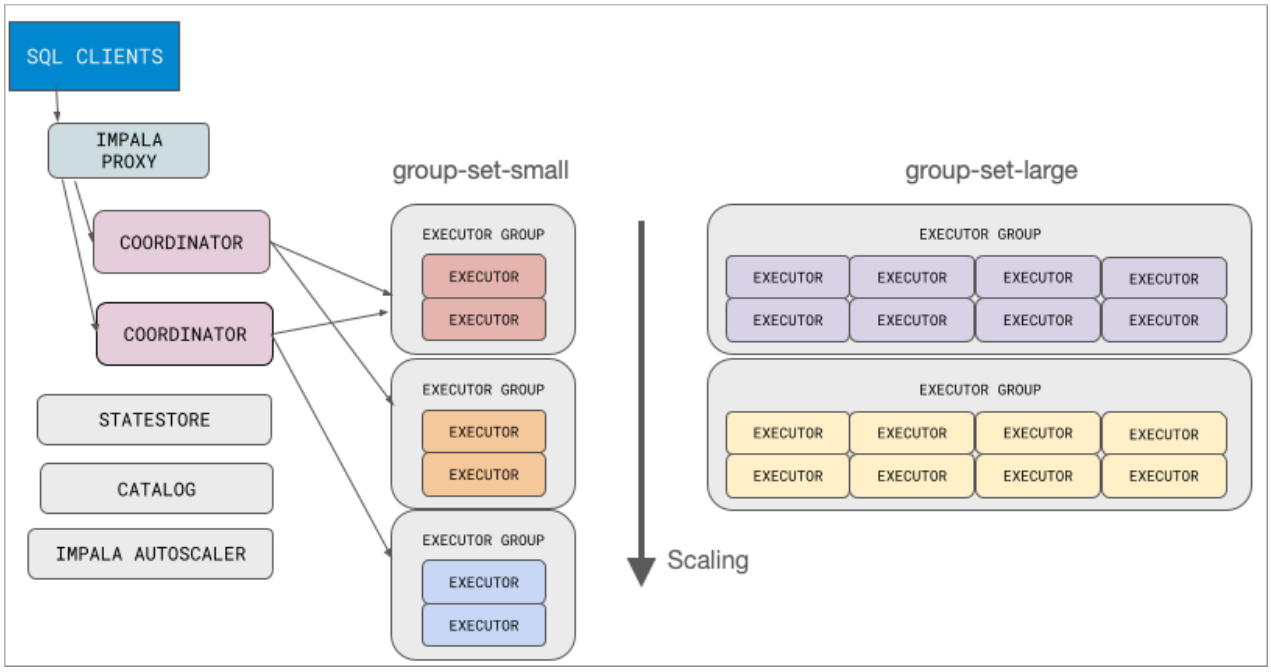

Using the UI shown below, you follow step-by-step instructions to configure a Small and Large executor group set.

You can expect the Small executor group set to handle the majority of cases, similar to a small (tee-shirt sized) Virtual Warehouse. This group keeps minimum resources for services. The Large executor group set can scale up greatly to service analytic queries. In Min No. of Groups, you configure a minimum number of groups to be allocated for processing queries before Impala auto-suspends. You configure a maximum number of groups based on your budget.

When migrating from a bare metal installation where your workload would run on hundreds of nodes, or more, consider also configuring an intermediate size using a Custom Set. You can add up to three Custom Sets for a total of five executor group sets. Cloudera recommends starting with no custom sets, establishing benchmarks, and adding one custom set at a time.

The following comparison highlights the differences in auto-scaling with and without WAAS.

Without WAAS	With WAAS
Executors provisioned when query runs for one-size-fits-all Virtual Warehouse	Executor group sets provisioned dynamically within range limits
Scaling occurs in multiples of the one-size-fits-all Virtual Warehouse	Scaling occurs within configured limits of two or more executor group sets
Small queries can run on the larger than necessary Virtual Warehouse	Small queries run on a smaller executor group set
Unused CPU expenses hard to control	Unused CPU expenses manageable
Large queries cannot run on a small Virtual Warehouse	Large queries run on an executor group of the appropriate size
SLA hard to manage	SLA manageable

WAAS uses replanning to select an executor group set. See *How workload-aware autoscaling works*.

Related Information

How workload-aware autoscaling works

How workload-aware autoscaling works

The Impala planner traverses successively larger executor groups iteratively to find one that can run the query efficiently. Traversal stops when the planner finds an acceptable executor group set. In Workload Aware Auto-Scaling, this process is called replanning.

The decision regarding whether an executor group set can run the query is based on memory and CPU estimates. If the query will not fit into the estimated memory, or if the query needs more cores to run efficiently, the planner considers the next largest executor group set. If none is found, the query is queued to the largest pool.

The following diagram shows an example of the replan:

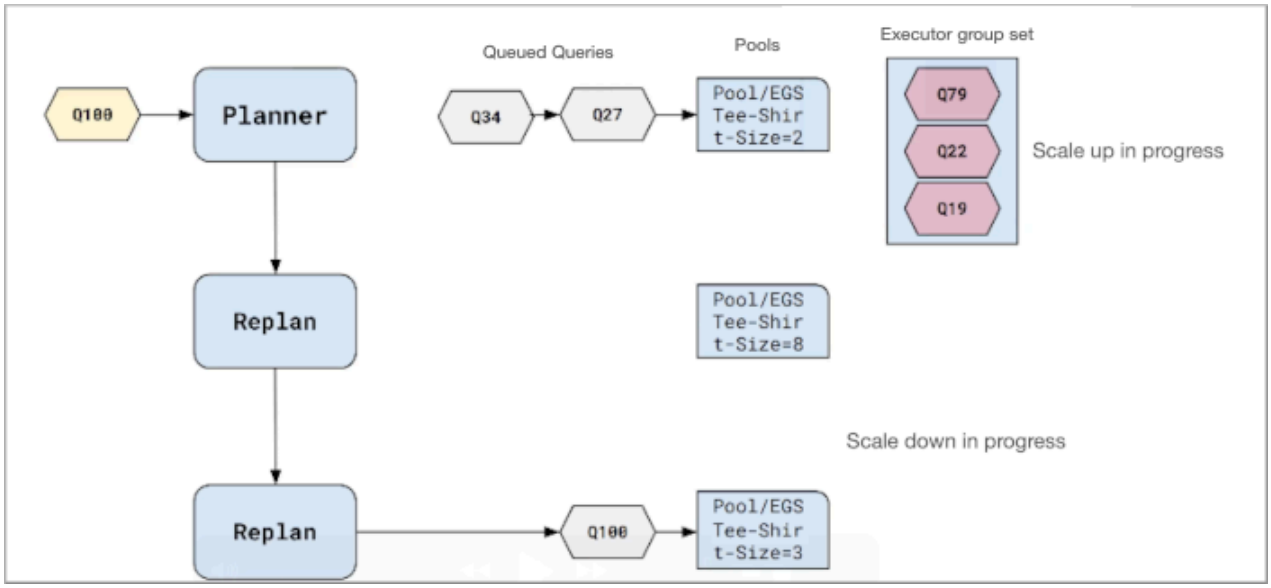

After replanning and establishing an executor group set able to run the query, the query enters the admission queue for a pool. The query will get admitted when the resources are available in the pool. The query may or may not succeed, depending on the resource settings and size of the executor group.

The size of the request pool and corresponding executor group set affects what queries can run there. No extra tuning is required. You can perform complex tuning of request pools using admission control by editing the xml configuration files. For more information, see the following documentation:

- Data Hub: Dynamic Resource Pool Settings
- Cloudera Data Warehouse: Configuring Admission Control and Configuring Resource Pools and Admission Control

Workload Aware Auto-Scaling tuning

You can use an Impala query option to control query routing. You can also set a parameter to restrict queries from running in parallel under certain circumstances.

With Workload Aware Auto-Scaling (WAAS), Impala internally uses the REQUEST_POOL query option to control which executor group set is used for the query. WAAS is sophisticated, but if a query is being sent to the wrong pool, you can use the REQUEST_POOL query option to override the executor group set to be used.

Request pool query option

You can override Impala using the REQUEST_POOL query option to select a particular executor group set if you want. For example, high query latency might indicate the query is not being routed to the correct pool. You might get better performance from a larger pool than the REQUEST_POOL. Your large queries can be targeted at a specific pool using a request pool.

The following example overrides Impala using the REQUEST_POOL query option to route large queries to group-set-large:

set request_pool="group-set-large"

Parallelism query option

WAAS includes the capability in the Impala planner to reduce query parallelism under certain conditions.

The MT_DOP multi-threaded planning for queries in Cloudera Data Warehouse was previously static in nature. A query would use 12 threads per node and aggressively use all cores. In some cases, using all cores is unnecessary. A new feature can reduce query parallelism to more effectively use resources. Sophisticated planner functionality makes

passes over the plan to determine how many cores are optimal for the query. Intelligence is gathered about fragments, how much work is involved in each fragment, and blocking operations in the plan.

Traversing fragments of the query plan tree, the planner propagates estimates of the max core count. Many details are gathered to determine the optimal parallelism for a pool. If parallelism is greater than the threshold (admission-controller.pool-max-query-cpu-core-per-node-limit), traverse to the next larger pool. The information is analyzed and the planner restricts parallelism for fragments that require few threads. Costs are reduced.

To enable this feature, you restrict parallelism by setting the COMPUTE_PROCESSING_COST=1 query option.

About the Impala Autoscaling Dashboard

The Impala Autoscaling Dashboard helps you to visualize and understand how the Impala Virtual Warehouse autoscales, how the queries are routed to an executor group set, how the queries affect the provisioning of the executors, and resource utilization over a specified time window. You can use this dashboard to monitor the regular as well as workload-aware autoscaling.

Impala Autoscaling Dashboard: Key insights

The Impala Autoscaling Dashboard enables you to obtain insight about the inner workings of how an Impala Virtual Warehouse autoscales. It contains the following graphs for each executor group set:

Query Execution Timeline

Enables you to analyze to which executor group sets the Impala queries are routed and why

- You can visualize which executor group sets are assigned what queries. This mainly depends on the resources available to the executor group sets.
- You can also view the timeline of each query's execution: how long was the query queued, when was it admitted, how long did it take to run, and so on.

Compute Metrics

Enables you to analyze how the queries that are queued and run affect the provisioning of the executors

- You can view the number of queued or running queries at any given time.
- You can also view the number of executors that are running, or have scaled up or scaled down over a specified time window.

Utilization Metrics

Enables you to analyze resource utilization of the running the queries

- You can view the percentage of available CPU, memory, scratch, and cache that is being used.

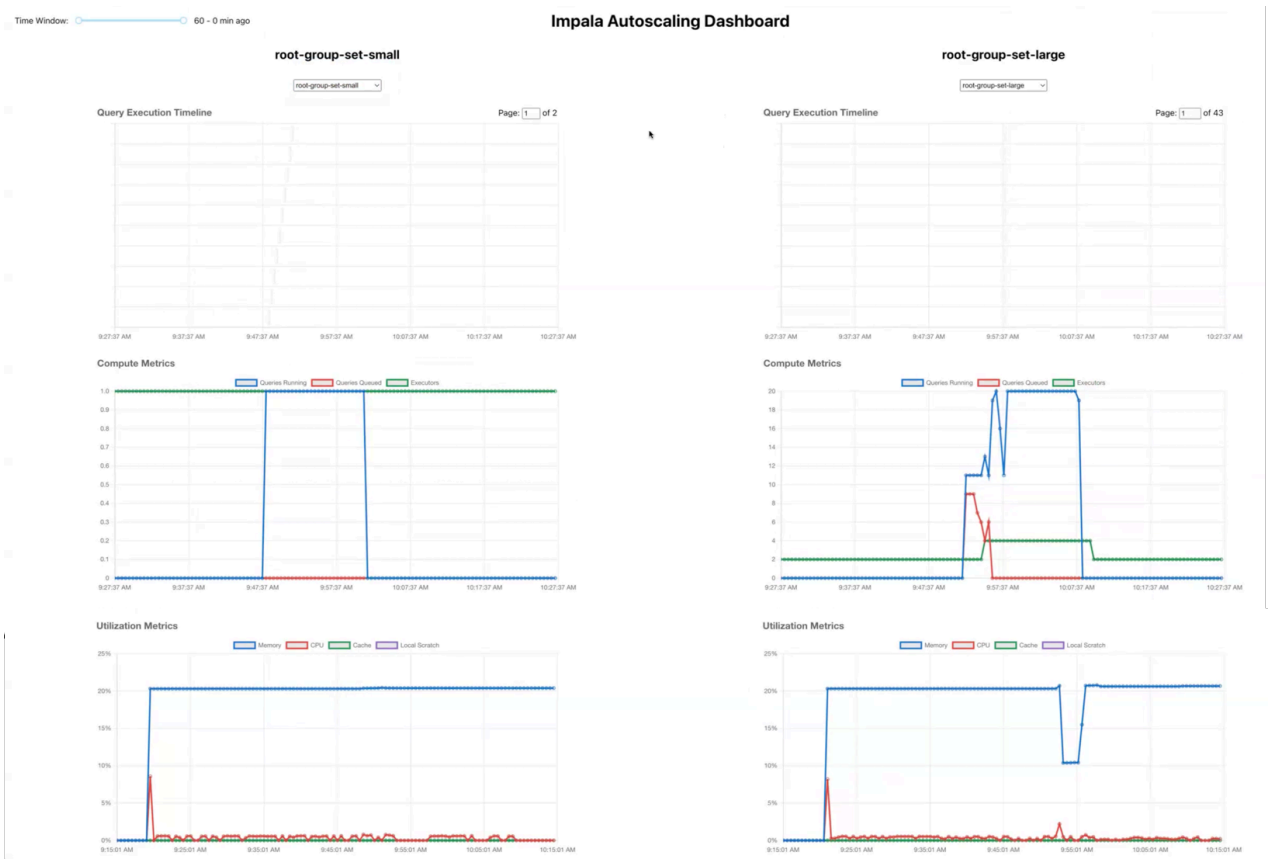

Impala Autoscaling Dashboard: Layout and controls

Each column on the Impala Autoscaling Dashboard is dedicated to an executor group set or a resource pool. For each executor group set, you get interactive charts to analyze query execution timeline, compute metrics, and utilization metrics. Columns for new executor group sets are added if you have selected to use additional executor group sets. If you have not enabled workload-aware autoscaling on your Impala Virtual Warehouse, then only one executor group set is deployed, and you see only one column on the dashboard.

You can use the Time Window slider to zoom into a specific time period.

Under each executor group set, there is a drop-down menu that enables you to select executor groups with a set. It enables you to view and analyze the metrics for a specific executor group in detail.

Each query execution timeline page is dedicated to a set of queries that has run in a recent time period. The first page shows the most recently run queries. You can navigate through older queries by increasing the Page number.

On the Query Execution Timeline graph, you can view all the phases of the query's lifecycle. To view details such as query ID, start and end time of a phase, the reason why this query was assigned to this executor group set, you can hover over a phase. You can view the full query profile by clicking on the query's phase bar.

Accessing the Impala Autoscaling Dashboard

You access the Impala Autoscaling Dashboard from the Web UI tab of the Virtual Warehouse Details page in Cloudera Data Warehouse.

Procedure

1. Log in to the Cloudera Data Warehouse service as an EnvironmentAdmin.

2. Go the Virtual Warehouses tab and click ⋮ Edit corresponding to your Impala Virtual Warehouse.
   The **Virtual Warehouse Details** page is displayed.
3. Go to the Web UI tab and click Impala Autoscaler Web UI.
   The Impala Autoscaling Dashboard opens in a new tab.
   Specify your workload credentials to authenticate.

Cloudera Data Warehouse Public Cloud

AWS environments

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

AWS environments overview 5
AWS environment requirements checklist 6
Activating an AWS environment from Cloudera Data Warehouse 9
Load balancers for AWS environments 10
AWS restricted policy 11
Attaching a managed policy ARN 13
Activating your environment in reduced permissions mode 15
Minimum set of IAM permissions required for reduced permissions mode 18
Reduced permissions mode JSON IAM permissions policy template 20
Required tags for CloudFormation stacks created with reduced permissions mode 20
Setting up cloud resources for reduced permissions mode 21
Deactivating AWS environments created with reduced permissions mode 23
Retaining PostgreSQL backups in AWS environments 24
Viewing and editing AWS environment details 24
Deactivating an AWS environment 27
Cloud storage buckets 27
Bucket encryption 29
Accessing S3 buckets 29
Accessing buckets in the same AWS account 30
Accessing buckets using a custom key 30
Accessing buckets in a different AWS account 35
Accessing S3 buckets in a managed policy environment 37
Accessing buckets in the same AWS account under a managed policy 38
Accessing buckets in a different AWS account under a managed policy 40
Accessing S3 buckets in a RAZ environment 43
Accessing buckets in the same AWS account under RAZ 44
Accessing buckets in a different AWS account under RAZ 44
Enabling RAZ manually 47
Remote access 49
Granting remote access to Kubernetes clusters on Amazon EKS 50
Revoking remote access to Kubernetes clusters on Amazon EKs 51
Restricting access to endpoints in AWS environments 51
Editing the IP CIDRs in the trusted list for endpoints in AWS environments 53
Networking 53
Overlay networks for AWS environments in Cloudera Data Warehouse service 53
Enabling overlay networks in AWS environments 55
Use a non-transparent proxy with Cloudera Data Warehouse on AWS environments 55
Configure non-transparent proxies for Cloudera Data Warehouse on AWS environments 56
Setting up private networking in AWS environments 57
Supported deployment modes for private networking in AWS 57
Prerequisites for private networking in AWS environments 58
Activating an AWS environment with private subnet support 59
Architecture for Private Load Balancer, Private Worker Nodes deployment on AWS 60
Custom tags in AWS environments 61
Upgrades 62
Upgrading PostgreSQL 9.6 before EOL 62
Validate the upgrade to PostgreSQL 10.16 64
Upgrade to PostgreSQL 11.12 65

Upgrading Amazon Kubernetes Service (EKS)....65
Upgrading using your own AMI or reduced permissions.... 66
Dynamically updating the Amazon Machine Image.... 67
Managed storage access....68
Setting up managed storage access....70
Creating the Cloudera environment and IAM roles....70
Creating a UMS group and machine users....72
Creating a new Database Catalog.... 74
Creating a tenant-specific Virtual Warehouse.... 74
Tenant IAM role policy....74
Using AWS Vault to manage credentials.... 75
Identifying the spill location for Impala temporary data....76
Configuring an existing Impala Virtual Warehouse to spill to S3....77
Setting the scratch space limit for spilling Impala queries....78

AWS environments overview

Learn about environments on AWS in Cloudera Public Cloud, including requirements you must meet before activating your environment in Cloudera Data Warehouse.

The Cloudera environment is closely related to the virtual private network in your cloud provider account. Registering an environment with Cloudera Management Console provides Cloudera with access to your cloud provider account and identifies resources in your account that Cloudera services can access, including Cloudera Data Warehouse. A single environment is contained within a single cloud provider region, so all resources deployed by Cloudera are deployed within that region within one specific virtual network. After you have registered an environment with Cloudera Management Console, you can activate the environment in Cloudera Data Warehouse. You create Virtual Warehouses in Cloudera Data Warehouse.

By default, Ranger Authorization is enabled in AWS environments. For an introduction to enabling RAZ, see the Cloudera Management Console documentation.

The following diagram shows the components of an AWS environment:

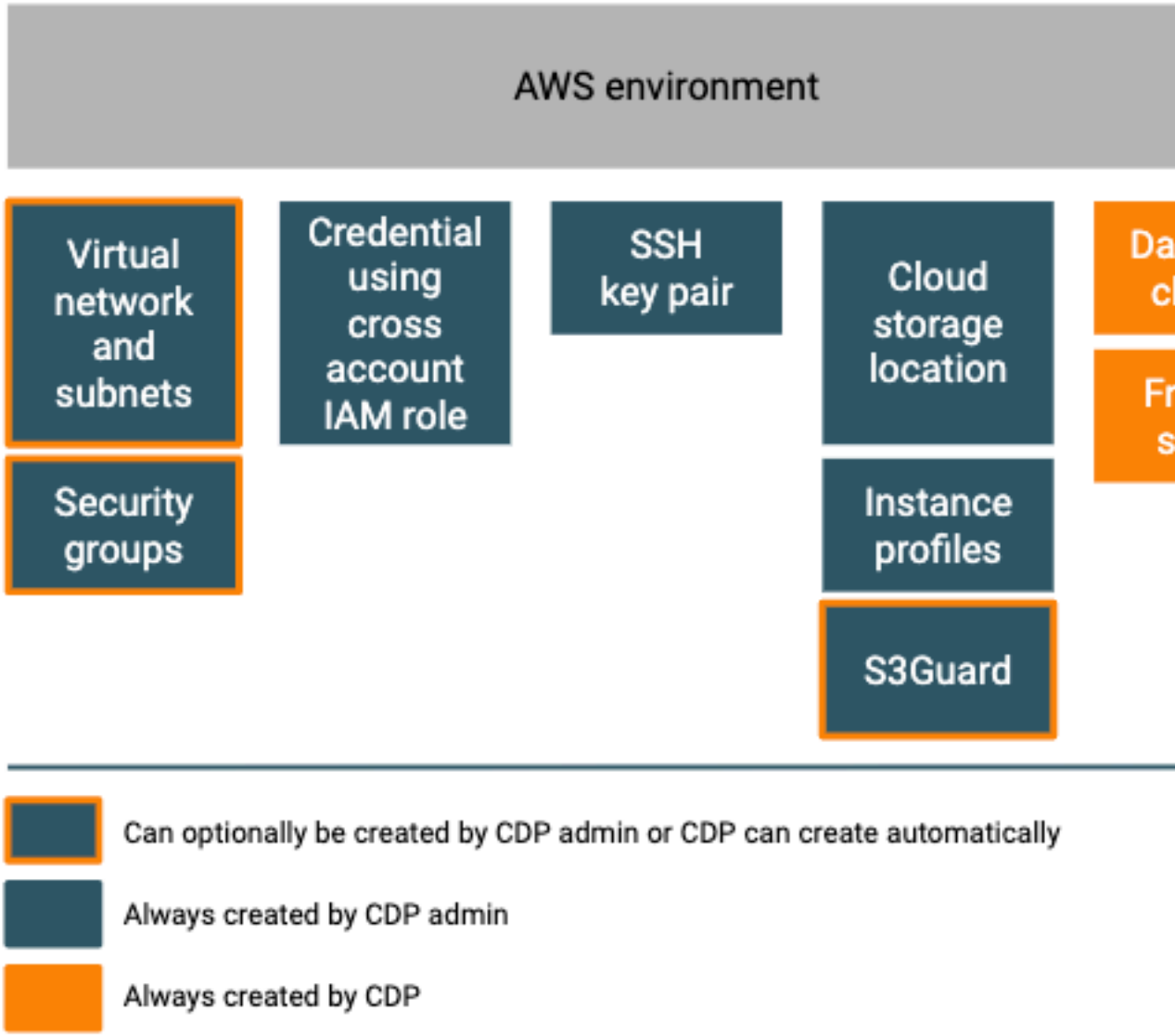

AWS environment requirements checklist

To successfully activate environments that have been registered with Cloudera on AWS VPCs with Cloudera Data Warehouse service, your AWS VPC must meet these requirements.

1. VPC has DNS resolution and DNS hostnames enabled

Ensure that your AWS VPC has DNS Resolution and DNS Hostnames enabled. For example, in the VPC Dashboard, click Your VPCs in the left navigation menu, and select the VPC you want to use for your Cloudera Data Warehouse service environment on Cloudera. View configuration details to make sure DNS resolution and DNS hostnames are Enabled. Thw AWS screen looks something like this:

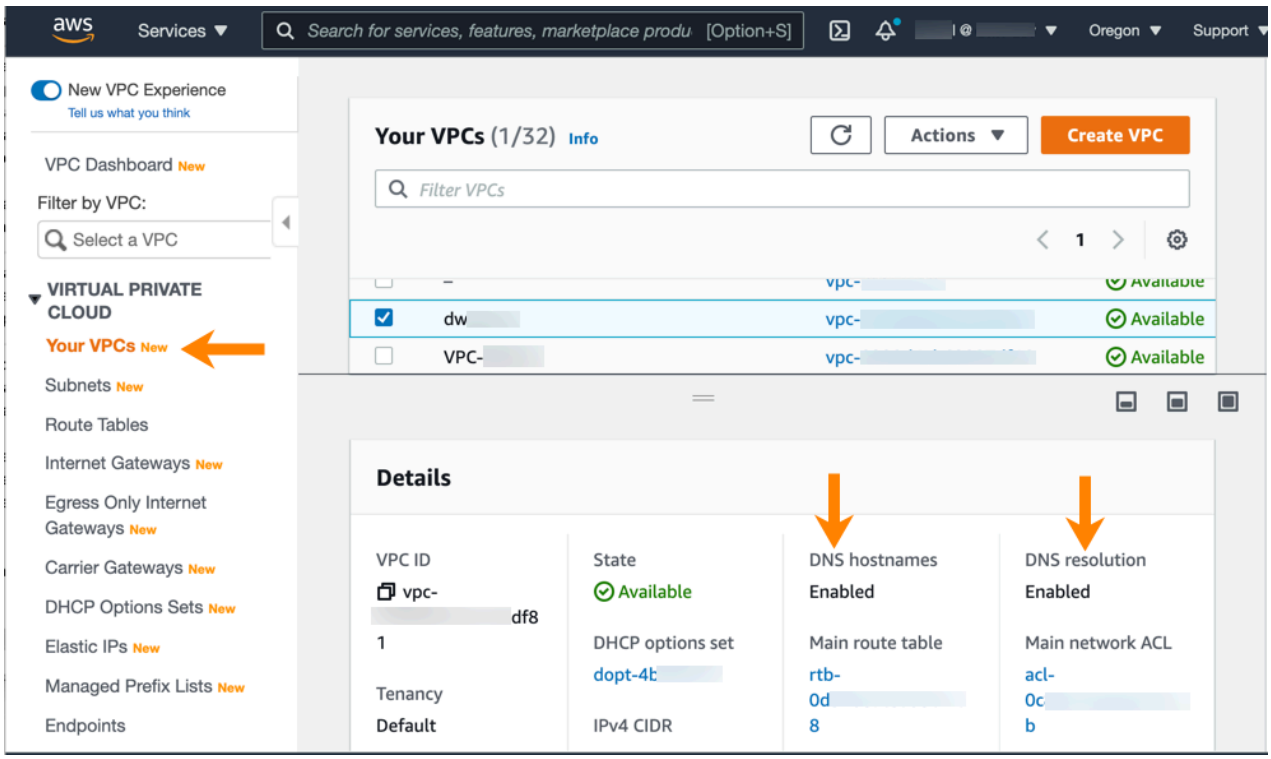

2. DHCP option set uses default domain name with one domain

When you create your VPC to use for the Cloudera Data Warehouse service, ensure that the DHCP option set attached to the VPC uses only one domain and use the default domain name:

domain-name = <region>.compute.internal;

Important:

If your VPC is in the us-east-1 region [U.S. East (N. Virginia)], the default domain name is ec2.internal:

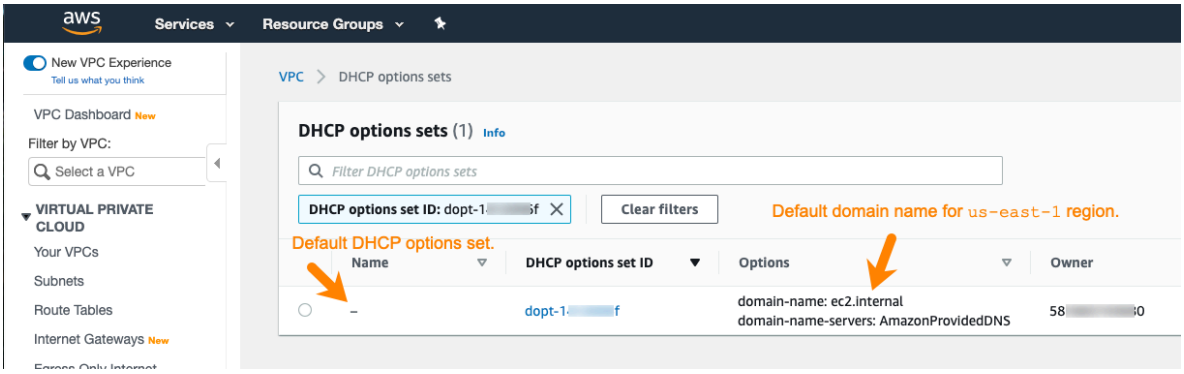

You can verify the setting in the VPC Dashboard of the AWS Console. Click the DHCP options set ID of the default DHCP options set (always named "-" by AWS) to view details, including the associated domain:

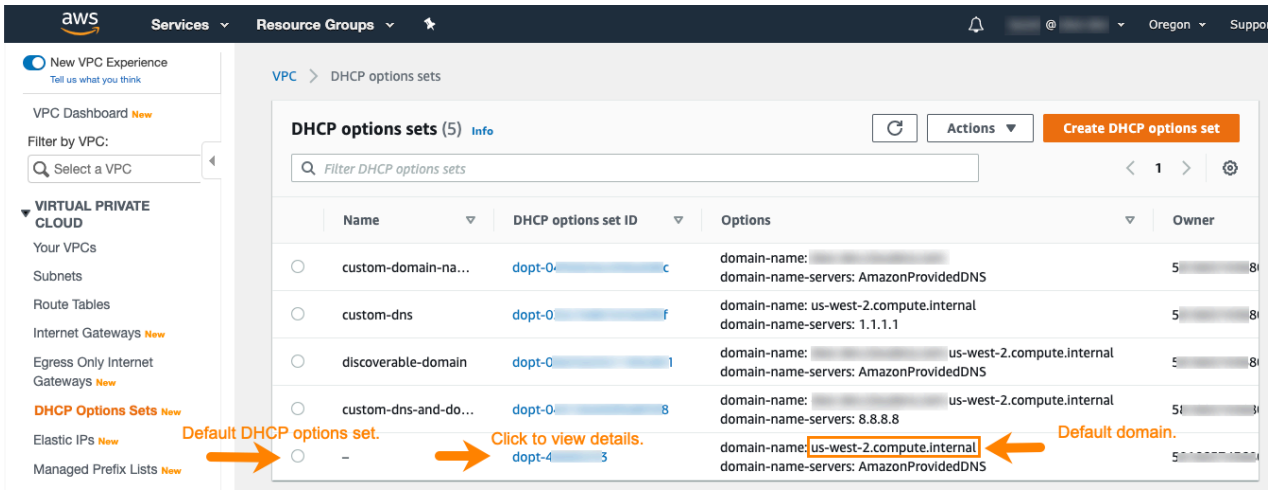

A details page appears:

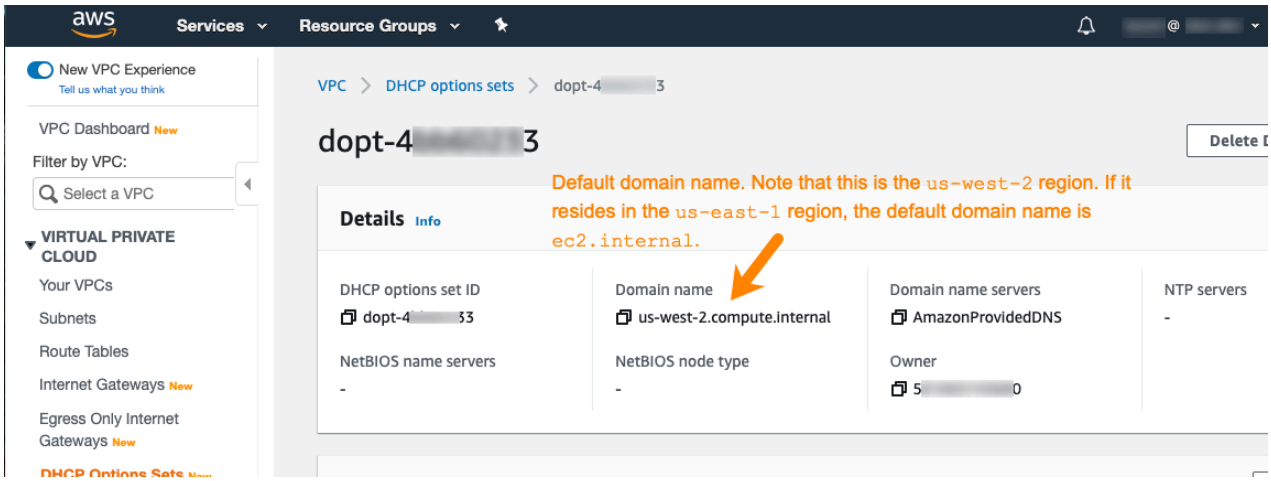

3. DHCP option set uses AmazonProvidedDNS

When you create the VPC for the Cloudera Data Warehouse service, AWS automatically creates a set of DHCP options and associates them with the VPC. This set of options specifies the Amazon DNS Server as the default domain name server:

domain-name-server = AmazonProvidedDNS;

Use this setting for VPCs for the Cloudera Data Warehouse service shown in the AWS Console VPC Dashboard above.

4. Ensure the correct subnets in VPC are specified

When you activate an environment for the Cloudera Data Warehouse service, ensure that the subnets are correct. If there are more than three private subnets in the VPC only the top three are selected. However, they may not be the subnets you intend to use for the Cloudera Data Warehouse service.

5. Ensure private subnets have outbound internet connectivity

Your private subnets must have outbound internet connectivity. Check the route tables of private subnets to verify the internet routing. Worker nodes must be able to download Docker images for Kubernetes, billing and metering information, and to perform API server registration.

Note: If you have used only Cloudera Data Hub in your Cloudera environment, you must check and possibly configure connections to additional outbound destinations for Cloudera Data Warehouse.

For more information, see AWS Outbound Network Access Destinations.

6. Ensure the Amazon Security Token Service (STS) is activated

To successfully activate an environment in the Cloudera Data Warehouse service, you must ensure the Amazon STS is activated in your AWS VPC:

1. In the AWS Cloudera Management Console home page, select IAM under Security, Identity, & Compliance.
2. In the Identity and Access Management (IAM) dashboard, select Account settings in the left navigation menu.
3. On the Account settings page, scroll down to the section for Security Token Service (STS).
4. In the Endpoints section, locate the region in which your environment is located and make sure that the STS service is activated.

Prerequisite for enabling a private EKS API server (Preview)

By enabling a private EKS API server, you can ensure that the EKS cluster is setup with only private endpoint enabled, which restricts the public access to your EKS API server from the internet. To set up the Amazon Eleastic Kubernetes Service (EKS) cluster in private mode and to enable the private EKS, ensure that the DataLake cluster is created with Cluster Connectivity Manager version 2 enabled.

You must also run the following Cloudera CLI command:

```
cdp dw create-cluster --environment-crn crn:cdp:environments:us-west-1:XXXX
--use-private-load-balancer --aws-options enablePrivateEKS=true,workerSubnet
Ids=privatesubnet-1,privatesubnet-2,privatesubnet-3,lbSubnetIds=privatesubne
t-XXX --profile dev
```

Note: Using a private EKS API server is under technical preview and not recommended for production environments. Cloudera recommends you use this feature in test or development environments.

Related Information

DHCP Options Sets in the Amazon documentation

Managing AWS STS in an AWS Region in the Amazon documentation

Activating environments

Activating an AWS environment from Cloudera Data Warehouse

To use an AWS environment for Cloudera Data Warehouse Public Cloud you must first activate it.

About this task

When you activate an environment, Cloudera creates an EKS cluster to host Kubernetes-based resources. The underlying compute, network resources are managed by AWS:

- Resource group
- Load balancer(s)
- Public IP address(es)
- Network security group
- Disk(s)

In the Cloudera Data Warehouse environment, instances for shared service components are set up within a Kubernetes (K8s) cluster. The setup begins with three m5.2xlarge instances running the Cloudera Data Warehouse service, but the K8s cluster is capable of autoscaling, automatically adding more instances if necessary to handle increased demand. Additionally, an Amazon Relational Database Service (RDS) (db.r5.large) running PostgreSQL is created to store user metadata for Hue and Data Visualization services. In total, three shared db.r5.large nodes are used for this purpose. Always active, shared services.

Before you begin

- Obtain the DWAdmin role.
- Review the AWS environment requirements.

Procedure

1. In the Cloudera Data Warehouse service, in Environments, locate the environment that you want to activate.
2. Click Activate.
3. In Deployment Mode, select load balancers.

   For more information, see Load balancers for AWS environments.

   Note: Select Private Load Balancer, Private Executor if you want to enable using a private EKS API server. Using a private EKS API server is under technical preview and not recommended for production environments. Cloudera recommends you use this feature in test or development environments.

   To view or configure the public and private subnets that have been specified for your Cloudera environment, click Advanced Settings.

   - Private Subnets: Accept the selected subnets you configured during AWS environment registration for load balancer and workload pods, or deselect subnets. Cloudera recommends three subnets for each load balancer to enable high availability (HA).
   - Enable IP CIDR for Kubernetes cluster: Enter the IP Classless Inter-Domain Routing (CIDRs) from which the Kubernetes cluster should accept incoming connections. Connections from other IP ranges are dropped. Obtain your internal network's IP CIDR ranges of IP addresses that need access to endpoints on the Kubernetes cluster. For more information, see Restricting access to endpoints in AWS.
   - Enable IP CIDRs for the load balancer: Enter the IP CIDR(s) from which the load balancer should accept incoming connections. Connections from other IP ranges are dropped. Obtain your internal network's IP CIDR ranges of IP addresses that need access to endpoints that are load balanced. For more information, see Restricting access to endpoints in AWS.
   - Use Overlay Network: Overlay Networks for AWS environments can increase the number of available IP addresses for your deployments of Cloudera Data Warehouse if you have an existing Virtual Private Cloud

(VPC). Use this feature if your Virtual Private Cloud subnet has fewer than 1,024 IP addresses. Cloudera recommends that you do not configure more than 200 executor nodes for an overlay network to operate.

- Attach Managed Policy ARN to Node Role: If you do not want to provide PutRolePolicy permission in your cross account role, you can attach a managed policy ARN to a node role to provide the cross account role permissions. You must create a new NodeInstanceRole manually, and provide the ARN during activation of the environment from Cloudera Data Warehouse.
- Use Reduced Permissions Mode: If you cannot provide the standard set of IAM permissions required by Cloudera Data Warehouse for environment activation, you can use reduced permissions mode to activate an AWS environment with fewer than half of these permissions. To use this feature, a minimum set of IAM permissions are required.
- Enable CloudWatch Logs: Enable CloudWatch logs if you use Amazon CloudWatch. In your AWS account, you can then find the logs in /aws/eks/<cluster name>/cluster. Before enabling CloudWatch, you must add required permissions to your IAM policy to access CloudWatch logs; otherwise, you cannot activate the environment.

4. Click Activate.

Related Information

Supported deployment modes
Setting up private networking
Restricting access to endpoints
Overlay networks for AWS environments

Load balancers for AWS environments

When you activate an environment, you can select load balancers.

You can select public and private load balancers to evenly distribute the inbound traffic. Deployment Mode options that you can select are as follows:

Option	Brief Description	Requirements and Recommendations
Private Load Balancer, Private Executors	Cloudera Data Warehouse nginix based load balancer runs on private subnets and all workload pods that also run on private subnets.	Requires a jump host or AWS direct connect to access Cloudera Data Warehouse. Cloudera recommends that you use the Private Load Balancer, Private Executors deployment mode if possible for security reasons. Selecting this option is required if you want to use a private EKS API server. **Note:** Using a private EKS API server is under technical preview and not recommended for production environments. Cloudera recommends you use this feature in test or development environments.
Public Load Balancer, Private Executors	Cloudera Data Warehouse nginix based load balancer runs on public subnets and all workload pods that also run on private subnets.	Select this option to connect to Cloudera Data Warehouse from anywhere as long as the source CIDR filters allow the connection.
Public Load Balancer, Public Executors	Cloudera Data Warehouse nginix based load balancer runs on public subnets and all workload pods that also run on public subnets.	Select this option to connect to Cloudera Data Warehouse from anywhere as long as the source CIDR filters allow the connection. Cloudera does not recommend selecting this option for security reasons.

AWS restricted policy

As Administrator, you must include the AWS restricted policy in your IAM role to limit access to the environment. You must include this policy before you activate the environment in Cloudera Data Warehouse.

The AWS restricted policy associates a cross-account role with the environment. If you do not have a Ranger Authorized (RAZ)-enabled Data Lake, simply attach the AWS restricted policy to your cross-account role, and nothing more. If you do have a RAZ-enabled Data Lake, take the following actions:

- Attach the AWS restricted policy described below to your cross-account role.
- Add ARNs to the role or a new sid with appropriate permissions to the AWS restricted policy.

  For more information see "RAZ-enabled Data Lake restricted policy" below.

Attaching the policy to your cross-account role

The AWS restricted policy is split into two files because the content exceeds the AWS file size limit. To attach the policy to your cross-account role, you need to work with both files as described in the following steps:

1. In the AWS management console, find the Cross-account IAM role you created.
2. Go to the following Github links to get the restricted policy files without comments:
   - Restricted policy file 1
   - Restricted policy file 2

     For your information, you can get a commented version of each file that explains each line in the policy.

     - Restricted policy file 1 with comments
     - Restricted policy file 2 with comments
3. Attach the policies to your IAM role.
4. In the Restricted policy file1 without comments, replace ${DATALAKE_BUCKET} with the name of your S3 bucket. For example my-bucket.
5. Attach both restricted policy file 1 and restricted policy file 2 without comments to your IAM role.

   Do not attempt to attach the policy files with comments as this would cause an error.

RAZ-enabled Data Lake restricted policy

The AWS restricted policy associates a cross-account role with the environment, as mentioned above. If you have a Ranger Authorized (RAZ)-enabled Data Lake, one of the following additions to either the cross-account role or cross-account json policy are required:

- File 1 Append ARNs to role: To your cross-account restricted policy, in the existing sid "AttachRole ", append all policy ARNs attached to the RAZ role.
- File 2 Add a new sid: Add "AttachRAZPolicyToNodeInstance" to the cross-account json policy

File 1 append ARNs to role

Append all the cross-account policy ARNS attached to the RAZ role to the sid key "AttachRole" value in your cross-account restricted policy.

```
{
      "Sid": "AttachRole",
      "Effect": "Allow",
      "Action": "iam:AttachRolePolicy",
      "Resource": [
         "arn:aws:iam::*:role/env-*-dwx-stack-EKSServiceRole-*",
         "arn:aws:iam::*:role/env-*-dwx-stack-NodeInstanceRole-*"
      ],
      "Condition": {
         "ForAnyValue:ArnEqualsIfExists": {
```

```
            "iam:PolicyARN": [
               "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy",
               "arn:aws:iam::aws:policy/AmazonEKSServicePolicy",
               "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly",
               "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy",
               "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy",
               "arn:aws:iam::aws:policy/CloudWatchAgentAdminPolicy"
               ...

            ]
         }
      }
   }
```

Replace the ellipsis … placeholder above with all policy ARNs attached to the RAZ role as shown in the example below. See IAM policy definitions for more information about these policies.

```
arn:aws:iam::1234567:policy/aws-cdp-datalake-admin-s3-policy

arn:aws:iam::1234567:policy/aws-cdp-bucket-access-policy

arn:aws:iam::1234567:policy/aws-datalake-backup-policy

arn:aws:iam::1234567:policy/aws-datalake-restore-policy
```

File 2 Add sid to policy

Add the new sid "AttachRAZPolicyToNodeInstance" value to the cross-account json policy.

Make sure all the policies attached to RAZ/Data Lake Admin role are in the following regex pattern format:

```
${ANY_WILDCARD_REGEX}
```

Use the pattern as a value for the key "iam:PolicyARN" as shown in the example below:

Example policies attached to the RAZ/Data Lake Admin role

```
arn:aws:iam::1234567:policy/cdp-dev-datalake-admin-s3-policy
arn:aws:iam::1234567:policy/cdp-dev-datalake-bucket-access-policy
arn:aws:iam::1234567:policy/cdp-dev-datalake-backup-policy
arn:aws:iam::1234567:policy/cdp-dev-datalake-restore-policy
```

The regex is "arn:aws:iam::1234567:policy/cdp-dev-datalake*".

```
{
      "Sid": "AttachRAZPolicyToNodeInstance",
      "Effect": "Allow",
      "Action": "iam:AttachRolePolicy",
      "Resource": [
            "arn:aws:iam::*:role/env-*-dwx-stack-NodeInstanceRole-*"
      ],
      "Condition": {
            "ForAnyValue:ArnLikeIfExists": {
                    "iam:PolicyARN": "arn:aws:iam::<AWS_ACCOUNT_ID>:policy/
${ANY_WILDCARD_REGEX}"
            }
       }
 }
```

For more information about using RAZ, see fine-grained access control and Cloudera policies documentation.

Attaching a managed policy ARN

For security reasons, if you do not want to provide PutRolePolicy permission in your cross account role, which would be used later to add an inline policy to the Node instance role, you must create a managed policy. Failure to do so results in an authorization error. You learn how to create the managed policy in a few steps.

About this task

If you are using the restricted policy with managed policy ARN, you need to add the permission to the cross account role from this Github link.

Note: Enable DAS Link will not be supported, since *PutRolePolicy* permission is not available in your cross account role.

Procedure

1. Follow instructions in steps 1-4 to activate your environment in Cloudera Data Warehouse, as described in the "Activating an AWS environment" above.

2. Select Attach Managed policy ARN to Node Role, and pass the ARN.
   For example:

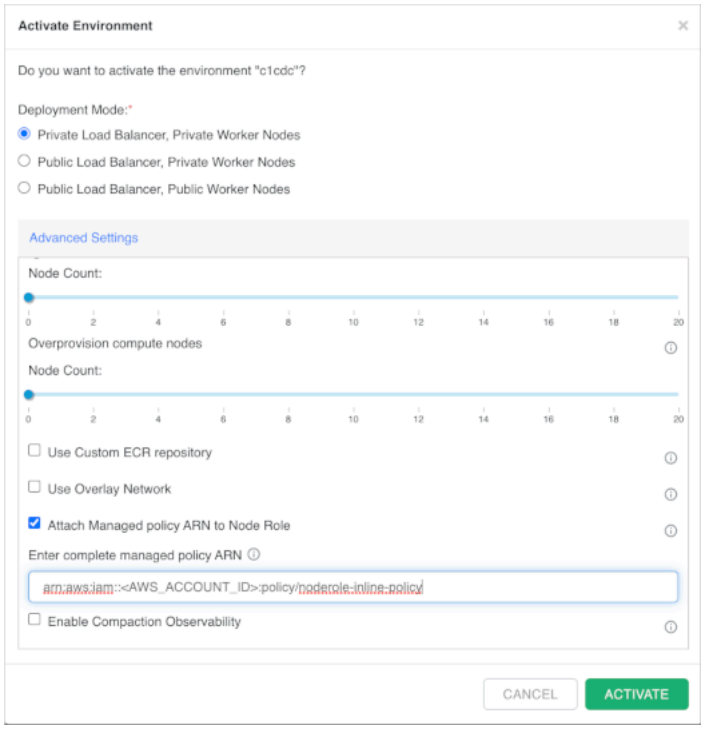

   When you remove the PutRolePolicy permission, which is one of the standard permissions needed, from your cross account role, the reduced permissions mode UI appears.

   For example:

   For more information about reduced permissions mode, see the topic below.

3. In Environmental Activations, optionally do not select Reduced Permissions mode.
4. Click Activate.
   The noderole-inline-policy.json is attached to the Node Instance role instead of a inline policy requiring the PutRolePolicy permission in your cross account role.

5. Make the following changes to the noderole-inline-policy.json file in your cross account role:
   - ${DATALAKE_BUCKET} - Replace this with the name of your S3 bucket. For example my-bucket.
   - ${STORAGE_LOCATION_BASE} - Replace this with the path to your Data Lake directory in the S3 bucket specified as ${DATALAKE_BUCKET}{}/SOME_PATH. For example my-bucket/my-dl.
   - ${LOGS_BUCKET} - Replace this with the name of your S3 bucket for logs. For example my-bucket. ${LOGS_LOCATION_BASE} - Replace this with the path to your S3 location for logs. For example my-bucket/my-dl.
   - ${BACKUP_LOCATION_BASE} - Replace this with the path to your S3 location for backups. This location is used for both FreeIPA and Data Lake backups. For example my-bucket/my-dl.
   - ${BACKUP_BUCKET} - Replace this with the name of your S3 bucket for backup. For example my-bucket.

Activating your environment in reduced permissions mode

IAM permissions are required by Cloudera Data Warehouse for environment activation. You can choose to provide a reduced set of IAM permissions for environment activation instead of the full set in the AWS restricted permissions policy.

About this task

You can activate an AWS environment for Cloudera Data Warehouse with fewer than half the set of required IAM permissions on your AWS cross-account IAM role. You can choose reduced permissions mode in two ways:

- Select the Use reduced permissions mode when you activate your environment from Cloudera Data Warehouse.
- Alternatively, let the system detect your account does not have the AWS restricted permissions policy.

  The following dialog appears. You can select Check to activate environment with reduced permissions mode.

In this task, you activate your environment from Cloudera Data Warehouse in reduced permissions mode. In steps 6 and 7, you manually create the stack and then remove it:

Procedure

1. In the Cloudera Data Warehouse service, in Environments, click the search icon and locate the environment that you want to activate.
2. Activate the environment.

3. In Activation Settings, if you do not have the standard set of required IAM permissions or a restricted policy in place, select Use Reduced Permissions Mode.
   For example:

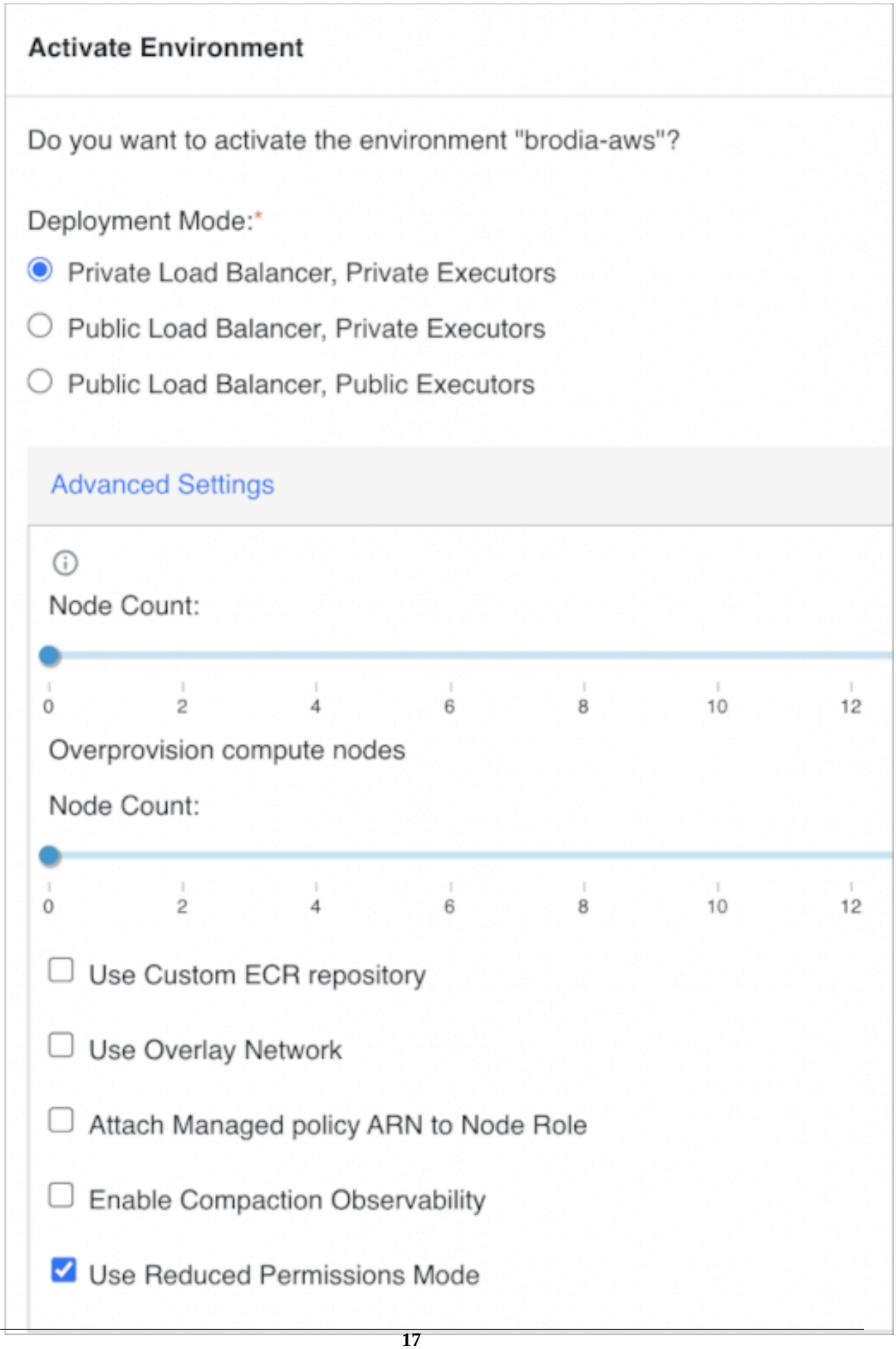
Activate Environment
Do you want to activate the environment "brodia-aws"?
Deployment Mode:*
Private Load Balancer, Private Executors
Public Load Balancer, Private Executors
Public Load Balancer, Public Executors
Advanced Settings
Node Count:
0 2 4 6 8 10 12
Overprovision compute nodes
Node Count:
0 2 4 6 8 10 12
Use Custom ECR repository
Use Overlay Network
Attach Managed policy ARN to Node Role
Enable Compaction Observability
Use Reduced Permissions Mode

Alternatively, if Environment Validations appears, select Check to activate environment with reduced permissions mode. Skip the next step and go to step 6.

4. If you do not want to activate the environment in reduced permissions mode, uncheck the option, and click Activate. Skip the rest of the steps in this procedure. Cloudera Data Warehouse automatically creates the cloud resources in your AWS account for you.
5. Manually create the cloud resources in your AWS account and tag them appropriately, as described in topic, "Setting up cloud resources for reduced permissions mode" below.

   Cloudera Data Warehouse pre-populates the required CloudFormation template for you within the AWS console, and you perform the manual steps to create the stack.
6. When you are finished using the stack, manually delete it in the AWS console.

Related Information

Minimum set of IAM permissions required for reduced permissions mode

Minimum set of IAM permissions required for reduced permissions mode

Review a list of the minimum IAM permissions required to activate AWS environments for Cloudera Data Warehouse in reduced permissions mode.

The following is a list of the minimum permissions that are required for your IAM policy to activate environments for Cloudera Data Warehouse in reduced permissions mode. In this mode you must manually create your CloudFormation stack from a template that Cloudera Data Warehouse pre-populates in the AWS console for you. When you are finished using the stack, you must manually delete its resources in your AWS account.

Table 1: Minimum set of IAM policy permissions required for environment activation in Cloudera Data Warehouse in reduced permissions mode

AWS service	"Allow" actions	Description
Certificate Manager (acm)	DescribeCertificate	Created by Cloud Formation to check the certification status during activation.
	ListCertificates	ACM validation adds DNS records.
CloudFormation (cloudformation)	DescribeStackEvents	Get Cloud Formation stack events, identify cause of failed CF stack creation failure
	DescribeStacks	Check the status of stack--error or completed, then install helm charts
	UpdateStack	Update Custom AMI, upgrade EKS
CloudWatch (logs)	CreateLogGroup	Create/name cloudwatch log group
	CreateLogStream	Create log stream of log group that originates from monitored application or resource
	DescribeLogStreams	List log streams for log groups
	PutLogEvents	Upload log events to log stream
	PutRetentionPolicy	Change number of days Cloudwatch retains
EC2 (ec2)	CreateKeyPair	Create ssh Public key pair, pass to ec2 instances. Not required if passed/set/reused via CloudBreak
	CreateTags	Tag subnets and eks security group. Amazon EKS security group requirements and considerations
	DeleteKeyPair	Delete keypair while deactivating Cloudera Data Warehouse // needed if CB env ssh is not reused
	DeleteTags	

AWS service	"Allow" actions	Description
	DescribeDhcpOptions	See points 2-3 in AWS Requirements Checklist
	DescribeKeyPairs	Validate CloudBreak env ssh key pair exists, not deleted inbetween; check for duplicate keypair in case of Cloudera Data Warehouse created keypair
	DescribeRouteTables	
	DescribeSubNets	See Point 4 in AWS Requirements Checklist
	DescribeVpcAttribute	Validate enableDnsHostnames and enableDnsSupport VPC attributes; see 1 and 3 points in Footnote 3 URL
	DescribeVpcs	Validate ID of set of DHCP options associated with the VPC
EC2 Auto Scaling (autoscaling)	DescribeAutoScalingGroups	Get shared services/compute ASGs, update as part of AZRebalance
	SuspendProcesses	Suspend AZRebalance for autoscaling group; include AZRebalance; cannot suspend AZRebalance in cloudformation; edit/update ASGs with AWS API to avoid AWS re-balancing nodes for AZ (most nodes run in stateful/critical pods)
	UpdateAutoScalingGroup	Calico overlaynetwork option requires no EKS nodes up on installation; with CF stack creation 3 nodes start up, autoscaling group updates desired capacity to Zero via AWS API; need latest SSH key from CloudBreak for EKS node updates; new Launch template passes SSH key and updates in ASG
EKS (eks)	DescribeCluster	Calico overlaynetwork option requires no EKS nodes up on installation; with CF stack creation 3 nodes start up, autoscaling group updates desired capacity to Zero via AWS API; need latest SSH key from CloudBreak for EKS node updates; new Launch template passes SSH key and updates in ASG
	DescribeUpdate	Check status of Updates--enable Private EKS and Cloudwatch on EKS
	TagResource	Tag eks cluster, e.g.: clusterId, envId, clustername, accountId...
	UpdateClusterConfig	Update EKScluster config Enable Private EKS and Cloudwatch on EKS
	UpdateClusterVersion	Updates an Amazon EKS cluster to the specified Kubernetes version
IAM (iam)	DeleteRolePolicy	
	GetRolePolicy	Delete inline policies like efs, ebs, cluster-autoscaler etc created/attached to Node instance role at deactivation
	ListAttachedRolePolicies*	List policies attached to Ranger RAZ role; attach to NodeInstanceRole for S3 access if RAZ enabled
	PutRolePolicy	Add inline policies like efs, ebs, cluster-autoscaler to Node Instance Role
	SimulatePrincipalPolicy	Simulate Cloud Formation stack formation policies

AWS service	"Allow" actions	Description
RDS (rds)	StartDBInstance	
	StopDBInstance	
	DescribeDBInstances	
S3 (s3)	GetBucketLocation	Needed for external bucket feature via UI, where we validate the VPC and bucket region are the same
	GetObject	Get Cloud Formatoin template while CF stack creation may not be needed for reduced mode
	ListBucket	
	PutObjectAcl	
	PutObject	Put Cloud Formation template in SDX bucket

*Needed only in a Ranger Authorization (RAZ) environment.

Reduced permissions mode JSON IAM permissions policy template

To activate an AWS environment for Cloudera Data Warehouse using reduced permissions mode, you can use this sample JSON template when you register an environment in Cloudera.

In this mode you must manually create your CloudFormation stack from a template that Cloudera Data Warehouse pre-populates in the AWS console for you. When you are finished using the stack, you must manually delete its resources in your AWS account.

Get the JSON reduced permissions mode policy from Github. Use the policy in Step 6 of the procedure to create your cross-account IAM role for Cloudera. Make the following substitutions in the policy:

- Replace ${ACCOUNT_ID} with your AWS account number.
- Replace ${DATALAKE_BUCKET} with your datalake bucket name.

Related Information

Create a cross-account IAM role

Required tags for CloudFormation stacks created with reduced permissions mode

This is a list of tags you must manually apply to AWS CloudFormation stack resources when you use the reduced permissions mode to activate environments for Cloudera Data Warehouse.

Table 2: Required tags for CloudFormation stacks created with reduced permissions mode in Cloudera Data Warehouse

Tag key	Tag value
clusterName	Name of the Cloudera environment registered with Cloudera Management Console
data-warehouse-env-owner	Email ID of the account of the user who owns the stack and is a user who has access to the Cloudera environment. This is the email ID of the email address that is listed for Email on the Users page of the User Management module of Cloudera Management Console.
stackName	Name of the Cloudera Data Warehouse CloudFormation stack. In the format: env-<*ENVIRONMENT-IDENTIFIER*>-dwx-stack For example: env-6g8dsf-dwx-stack

Tag key	Tag value
clusterId	Cloudera Data Warehouse environment ID that is displayed in the environment tile in the Cloudera Data Warehouse UI. For example: env-hmrt2z
Cloudera-Resource-Name	Cloudera environment ID (CRN [Cloudera Resource Name]). In the format: crn:cdp:environments:<*REGION*>:<*ACCOUNT-ID*>: environment :<*IDENTIFIER*> For example: crn:cdp:environments:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:environment:cc8ad776-4704-48f7-a243-97348939becd
actorCrn	CRN (Cloudera Resource Name) from the user's profile in the Cloudera Management Console module of Cloudera. For example: crn:altus:iam:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:user:de370d9f-ebb4-4b75-a89b-5d15306ae143
accountId	Tenant ID. In the above CRN example, the tenant ID is the GUID listed immediately after the AWS region: 9d74eee4-1cad-45d7-b645-7ccf9edbb73d

Setting up cloud resources for reduced permissions mode

Learn how to activate environments on AWS using the reduced permissions mode in Cloudera Data Warehouse. In this mode, you must manually create and delete the CloudFormation stack in the AWS Console.

About this task

Required role: EnvironmentAdmin or PowerUser

When you activate an AWS environment for Cloudera Data Warehouse, if you do not have the standard required IAM permissions, the following message displays in the environment tile of the Cloudera Data Warehouse UI, which provides a link to the AWS Console:

```
Click the link and perform the following listed steps to navigate
to the AWS Console and create the CloudFormation stack.
```

Before you begin

- Because you need to use the AWS Console to manually create your CloudFormation stack for Cloudera Data Warehouse environment activation, in another browser tab, log into your AWS account before you begin. Make sure that the IAM entity logged in has the two AWS restricted policies described in "AWS restricted policies".
- You must also have the AWS CLI and the kubectl CLI configured and available on your system to apply the kubeconfig that Cloudera Data Warehouse provides in Step 10 below.

Important:

- Make sure you have the time to complete the task of creating your CloudFormation stack in reduced permissions mode in one sitting. This can take up to 20 minutes. Otherwise, if the creation process outlined below is delayed in the AWS Console for more than an hour, the Cloudera Data Warehouse environment activation times out and the environment will go into an error state.
- Make sure that the IAM entity you use to log into the AWS Console has adequate permissions to create CloudFormation stacks and to run kubectl commands on your AWS environment.
- Make note of the name of the IAM entity you use because you must use it to log in using a terminal window again in Step 10c. of the following procedure.

Procedure

1. In the Cloudera Data Warehouse UI Overview page, go to the Environments tab.

2. Locate the environment you want to activate, and click Activate.
3. If the system detects that you do not have the standard required IAM permissions on your AWS account for automatic CloudFormation stack creation by Cloudera Data Warehouse, it displays the following message in the tile:

   ```
   Step 1 of 2: Insufficient permissions! Visit AWS Console to Create Stack
   and come back. Creating a stack will take up to 15 minutes.
   ```

   This message asks you to navigate to the AWS Console to manually create the CloudFormation stack.
4. Click the link Visit AWS Console to Create Stack and the AWS Console opens on the CloudFormationStacksCreate Stack page that is pre-populated with a template in another browser tab.
5. In the Create Stack page, click Next to advance to the Specify stack details page.

   Important: Do not change any configurations on the Specify stack details page, including the Stack Name.
6. In the Specify stack details page, click Next to advance to the Configure stack options page where you can specify the required tags for your stack resources. See "Required tags for CloudFormation stacks," which is linked to at the bottom of this page for a list of required tags.
7. After adding the required tags, do not set the remaining options on the page. Scroll down to the bottom of the page and click Next to advance to the Review *<ENV-STACK-NAME>* page.
8. On the Review *<ENV-STACK-NAME>* page, scroll down to the bottom, click the I acknowledge that AWS CloudFormation might create IAM resources check box, and then click Create Stack.

   Note: Stack creation can take up to 20 minutes depending on network traffic and load. The Cloudera Data Warehouse UI monitors AWS stack creation and displays a "Creating" message in the environment tile.
9. After stack creation has completed, a message displays in the Cloudera Data Warehouse UI environment tile. Click the Open Configurations link and a Configurations dialog box displays.
10. In the Configurations dialog box, perform the following steps:
    a. Copy the Kubeconfig text to your system clipboard and save it into a text file on your system.
    b. Copy the Aws Auth text to your system clipboard and save it into a text file on your system. The Aws Auth text provides the IAM cross-account role that is registered in Cloudera to access the EKS cluster on AWS after you perform the kubectl command in the next step.
    c. In a terminal window, verify that the AWS CLI is configured to use the same IAM entity that you used to create the CloudFormation stack in Step 8. Then, using the kubectl CLI, run the following commands to apply configurations from the two text files that you created in Step 10b:

       ```
       $> export KUBECONFIG=<PATH-TO-THE-KUBECONFIG-TEXT-FILE>
       ```

       ```
       $> kubectl apply -f <PATH-TO-THE-AWS-AUTH-TEXT-FILE>
       ```

    Look for shell output "configmap/aws-auth created" to confirm the configuration was applied correctly.

    If you deployed the CloudFormation stack with a federated user using the AWS console, you need to execute the commands mentioned in 10c (above) in the AWS CloudShell. You might need to manually install the kubectl command in the CloudShell.
11. Back in the Cloudera Data Warehouse UI Configurations dialog box, select Yes, Kubeconfig and AWS Auth configuration are applied checkbox, and then click Finish Activation.

Results

After clicking Finish Activation, the environment is activated and the tile displays a starting message.

Deactivating AWS environments created with reduced permissions mode

Learn how to deactivate an environment that has been activated for use in Cloudera Data Warehouse with the reduced permissions mode. When you deactivate an environment in Cloudera Data Warehouse, the environment registered with Cloudera remains available for use by other applications.

About this task

Required role: EnvironmentAdmin or PowerUser

If a Cloudera environment has been activated for Cloudera Data Warehouse with the reduced permissions mode, then if you deactivate the environment in the Cloudera Data Warehouse UI, you must manually delete the CloudFormation stack in AWS and its associated S3 buckets and DynamoDB table. After you click the deactivation icon in the Cloudera Data Warehouse environment tile, a link displays in the tile that you can use to navigate to the AWS Console to delete these cloud resources.

Procedure

1. In the Cloudera Data Warehouse UI, navigate to the tile for the environment you want to deactivate, and click Deactivate, which launches the Action dialog box.
2. (Optional) In the Action dialog box, you can select one of the following environment deactivation options if appropriate:
   - Choose Drop Data if you want to drop any data Cloudera Data Warehouse created outside of the Data Lake, but retain the Database Catalogs and Virtual Warehouses that are associated with the environment.
   - Choose Force Delete if you want to drop the data and also remove the Database Catalogs and Virtual Warehouses that are associated with the environment.
3. Click Visit AWS Console to Delete Stack, which displays in the Action dialog box. This opens the AWS Console in another browser tab.
4. In the Action dialog box of the Cloudera Data Warehouse UI, click OK, and then wait five minutes so Cloudera Data Warehouse can perform its deletion steps.
5. After waiting five minutes, in the AWS Console, click Delete to delete the CloudFormation stack in your AWS account.
6. In the AWS Console, perform the following tasks:
   - Navigate to the S3 service and delete the following S3 buckets that were used by the stack:
     - `<S3-BUCKET-NAME>-<LAST-4-DIGITS-OF-ENVIRONMENT-ID>-dwx-managed`
     - `<S3-BUCKET-NAME>-<LAST-4-DIGITS-OF-ENVIRONMENT-ID>-dwx-external`

     For example, if the bucket name is sales-east and the Cloudera Data Warehouse environment ID is ENV-CK8988, the buckets you should delete are:

     ```
     sales-east-8988-dwx-managed
     ```

     ```
     sales-east-8988-dwx-external
     ```
   - Navigate to the DynamoDB service and delete the associated DynamoDB table.

Results

After performing these steps, the resources in your Amazon account for the deactivated Cloudera Data Warehouse environment have been deleted.

Retaining PostgreSQL backups in AWS environments

When you create a Cloudera Data Warehouse cluster using the Cloudera CLI create-cluster command, any PostgreSQL backup retention period you set on your Cloud Provider side, is observed by Cloudera.

Procedure

1. In AWX, configure BackupRetentionPeriod.
2. Create a Data Warehouse cluster using the Cloudera CLI create-cluster command.
   The Data Warehouse cluster will retain the PostgreSQL backups according to your configuration.

Viewing and editing AWS environment details

This topic describes how to view and edit Cloudera AWS environment details in the Cloudera Data Warehouse service UI.

About this task

You can view Cloudera AWS environment details without leaving the Cloudera Data Warehouse service UI. Accessing the Environment Details page in the Cloudera Data Warehouse UI also enables you to edit the description of the environment and the allowed IP Classless Inter-Domain Routing (CIDRs) that control access to Kubernetes and load balancer service endpoints.

Before you begin

- You must activate an environment before you can view or edit its details. See "Activating AWS environments," which is linked to in the "Related information" section at the bottom of this page.
- Obtain the DWUser role for viewing environment details.
- Obtain the DWAdmin role for editing the environment details.

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment that you want to view.

3. Click ⋮ Edit .
In Environment Details, you can view information about the environment, like the Cloudera Data Warehouse release version in which you activated the environment, when you created and last updated the environment, and how many Database Catalogs and Virtual Warehouses use the environment.

In Configurations, you can also make the following changes:

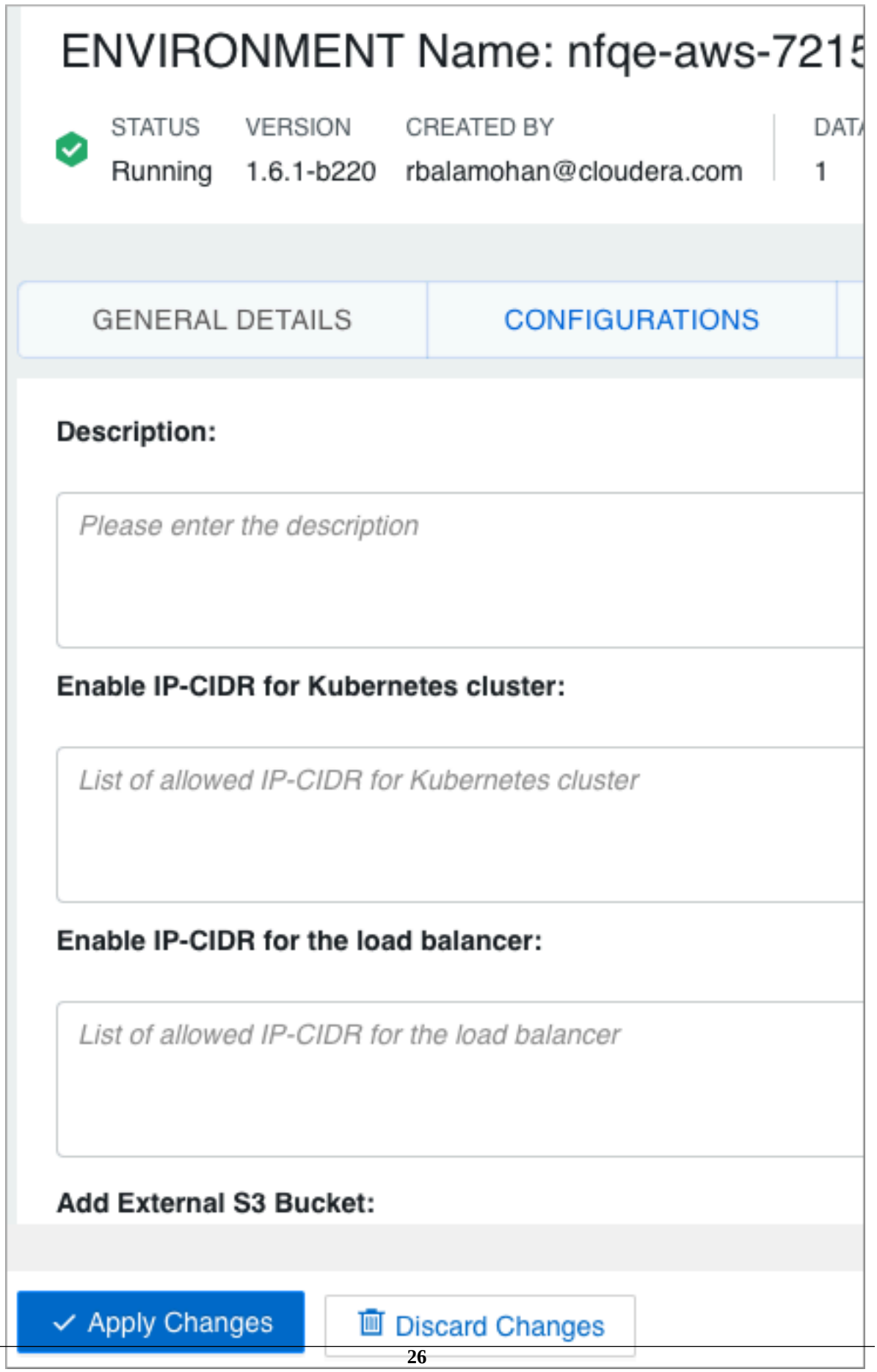
ENVIRONMENT Name: nfqe-aws-7215
STATUS
Running
VERSION
1.6.1-b220
CREATED BY
rbalamohan@cloudera.com
DATA
1
GENERAL DETAILS
CONFIGURATIONS
Description:
Please enter the description
Enable IP-CIDR for Kubernetes cluster:
List of allowed IP-CIDR for Kubernetes cluster
Enable IP-CIDR for the load balancer:
List of allowed IP-CIDR for the load balancer
Add External S3 Bucket:
Apply Changes
Discard Changes

- Add a description for the environment, that makes it easier to identify.
- Add or edit the list of IP CIDR(s) for Kubernetes Cluster to enable access from your internal network to the Kubernetes endpoints.
- Add or edit the list of IP CIDR(s) for the load balancer.
- Add an external S3 bucket to access data outside your Data Lake, such as a CSV File.
- Enable CloudWatch logs if you use Amazon CloudWatch. In your AWS account, you can then find the logs in /aws/eks/<cluster name>/cluster.

  Note: Before enabling CloudWatch, you must add required permissions to your IAM policy to access CloudWatch logs; otherwise, your cluster goes into an error state.

4. Click Apply Changes.

Related Information

Activating environments

Restricting access to endpoints

Deactivating an AWS environment

Learn how to deactivate an AWS environment for Cloudera Public Cloud.

Before you begin

Required role: EnvironmentAdmin or PowerUser

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment that you want to deactivate and click Deactivate.

   The **Action** modal is displayed.
3. On the Action modal, you can select environment deactivation options:
   - Choose Drop Data to remove the managed buckets that were created by Cloudera Data Warehouse during environment activation. The underlying data in the data lake is untouched. Only the default Database Catalog is retained because it resides in the data lake. Non-default Database Catalogs and Virtual Warehouses are deleted.

     Important: All metadata and the Ranger policies for non-default Database Catalogs are deleted.
   - Choose Force Delete to drop the data and to remove the Database Catalogs and Virtual Warehouses that are associated with the environment.
4. Click OK to deactivate the environment.

Cloud storage buckets

Cloudera Data Warehouse is integrated with the Data Lake Storage Cloud provider storage, such as AWS S3 or Azure Storage. During Data Lake creation, Cloudera creates storage locations for your data, logs, and backups.

For more information about AWS storage buckets, see S3 bucket and IAM roles and policies for logs, backup, and data storage. For information about which logs are stored in which directories, see Locations of Impala log files in S3.

Typically, during cluster creation, a managed policy is created automatically by Cloudera Data Warehouse and attached to a node instance role. Alternatively, if you need different permissions than those specified during cluster

creation, you can create or modify the managed policy and attach it to the node instance role. Whether you create the policy manually, or Cloudera Data Warehouse creates the policy automatically, the policy must specify the paths to the log, backup, and data buckets in the Resources array of the s3readwriteownbuckets object in the managed policy JSON.

```
"arn:aws:s3:::${LogBucket}/clusters",
"arn:aws:s3:::${LogBucket}/clusters/*",
"arn:aws:s3:::${LogBucket}/<Your configured log path>",
"arn:aws:s3:::${LogBucket}/<Your configured log path>/*",
"arn:aws:s3:::${BackupBucket}/<Your configured backup path>",
"arn:aws:s3:::${BackupBucket}/<Your configured backup path>/*",
"arn:aws:s3:::${DataBucket}/<Your configured data path>",
"arn:aws:s3:::${DataBucket}/<Your configured data path>/*"
"arn:aws:s3:::${DataBucket}/backup",
"arn:aws:s3:::${DataBucket}/backup/*",
```

You get the path and name of the bucket, which was specified during Data Lake creation. To get the paths and names of the buckets, navigate to Environments Data Lake . Click Summary.

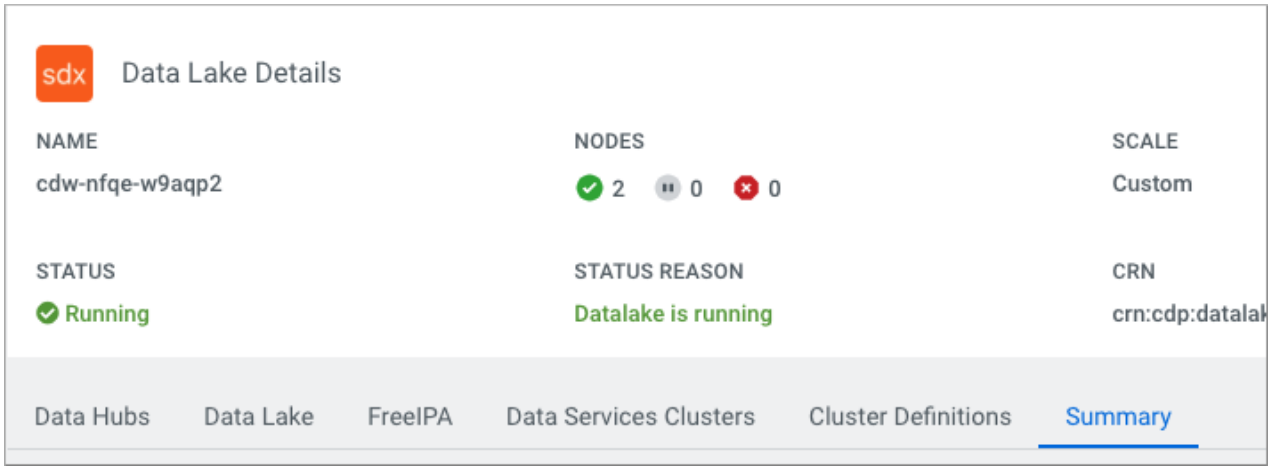

The name and paths of your logs and backup Data Lake buckets appear:

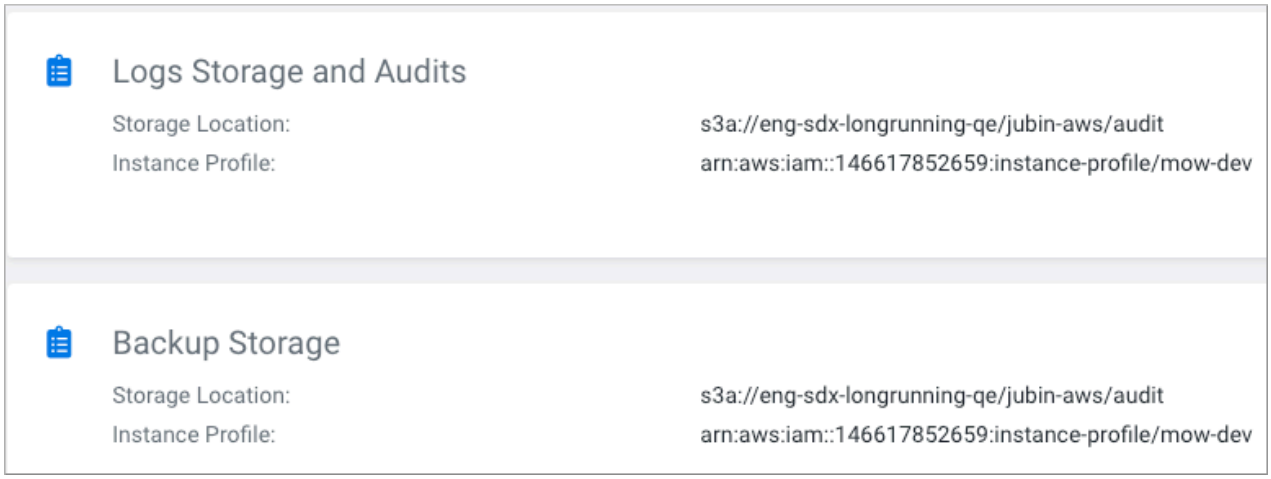

One S3 bucket with a sub-directory named after your data lake such as s3a://my-bucket/my-dl is created when your Data Lake is created. This bucket is called the Storage Location Base and is intended for your data.

Node instance roles need access to data, logs, and backup buckets in your Data Lake. To configure a node instance role to access these buckets, you attach the managed policy to the node instance role.

Bucket encryption

To allow encryption and decryption of your S3 bucket contents, the environment node instance role must access the KMS (Key Management System) key. Reading and writing to the bucket is impossible unless you provide this access.

By default, any resource under your account can use and manage keys. However, typically you want to restrict access to keys, authorizing only the node instance role to use the key. To accomplish this, you add code to your managed policy and attach the policy to the node instance role. The following example policy snippet provides minimal privileges to use the key.

```
{
     "Sid": "Allow use of the key",
     "Effect": "Allow",
     "Principal": {
         "AWS": "arn:aws:iam::555555555555:role/env-id-dwx-stack-NodeInsta
nceRole-xyz"
     },
     "Action": [
         "kms:Encrypt",
         "kms:Decrypt",
         "kms:GenerateDatakey",
         "kms:ReEncrypt*"
     ],
     "Resource": "*"
},
```

This code contains the following representations:

- 555555555555 is the example account id.
- env-id-dwx-stack-NodeInstanceRole-xyz is the example NodeInstanceRole for the environment.
  - env-id-dwx-stack is the example environment ID prefix.
  - xyz is a random string suffix

Accessing S3 buckets

In Cloudera Data Warehouse Public Cloudclusters running on AWS environments, you need to configure access to S3 buckets.

The subtopics of this document describe using the Cloudera Data Warehouse UI for configuring access to S3 buckets. This documentation does not apply if you are using Cloudera Data Warehouse as follows:

- In a RAZ (Ranger Authorized) environment

  See Accessing buckets in a RAZ environment.
- In a managed policy environment with a managed policy ARN

  See Accessing S3 buckets in a managed policy environment.
- In reduced permissions mode

In a RAZ or managed policy environment, or in reduced permissions mode, you use AWS instance profiles instead of the Cloudera Data Warehouse UI. The UI for configuring an external S3 bucket does not appear in these environments and modes.

Accessing buckets in the same AWS account

This topic explains how to configure read-only or read/write access using default encryption to external S3 buckets that reside in the same AWS account as the Cloudera Data WarehousePublic Cloud cluster.

About this task

Important: If you configure read-only access to an external S3 bucket, there is no need to restart Virtual Warehouses. However, if you configure read/write access to an external S3 bucket, you must restart Virtual Warehouses by suspending them and starting them again. Alternatively, you can create a new Virtual Warehouse to use the external S3 bucket with read/write access.

Required role: DWAdmin

Before you begin

- Identify and activate the environment you want to configure for access to an external bucket in the same AWS account.
- In the AWS Management Console, identify the external S3 bucket you want to configure access to.

Procedure

1. In the Cloudera Data Warehouse UI Overview, click Environments, choose the environment that is activated for the Virtual Warehouses you want to use with the external AWS bucket, and click ⋮ Edit .
2. In **Environment Details**, in the Enter s3 bucket name, type the name of the AWS bucket you want to configure access to.
3. Specify whether Read Only or Read Write access is needed.

   Note:

   For Read Write access only: if you do not need to use a custom encryption key, leave the ENCRYPTION SETTINGS text box blank. If you do not use a custom key, the system uses AES256 encryption by default. If you need to use a custom encryption key, see the next topic. Read Only access does not involve encryption.
4. Click Add Bucket to save the configuration. A success message displays at the top of the page.

What to do next

If you have configured Read Write access, you must restart the Virtual Warehouses that are associated with this environment for the configuration changes to take effect.

Accessing buckets using a custom key

If you want read/write access from Cloudera Data Warehouse Public Cloud on AWS to the external S3 bucket using your own custom encryption key, you must configure the encryption key.

About this task

Perform the steps described in this topic to use your own custom key.

Important: If you want to use a custom key, you must perform the configuration described in this topic whether the bucket is in the same AWS account as Cloudera Data Warehouse or if the bucket resides in a different AWS account.

Required role: DWAdmin

Before you begin

- Identify which environment you want to configure for access to an external bucket in another AWS account. In the Cloudera Data Warehouse UI, go to the Environments tab. This causes the Data Catalog and Virtual Warehouses that use this environment to be highlighted in the Cloudera Data Warehouse UI. Choose the environment that is activated for the Virtual Warehouses you want to use with the external AWS bucket.
- In the AWS Management Console for the different account, identify the external S3 bucket you want to configure access to.

Procedure

1. On the Cloudera Data Warehouse UI **Overview** page, go to the Environments tab and locate the environment for which you want to configure access to an external AWS bucket, and then click ⋮ Edit .

   This loads the **Environment Details** page.
2. Go to the Configuration tab and type the name of the AWS bucket you want to configure access to in the Add External S3 Bucket field.
3. (Optional) If you want to configure access to an S3 bucket in a different AWS account, select Bucket belongs to different AWS Account.
4. Select Read Write. This causes the ENCRYPTION SETTINGS text box and a key policy to display.
5. In the AWS Management Console for the account where the S3 bucket resides, navigate to the Key Management Service, and select Customer Managed Keys in the left navigation menu:

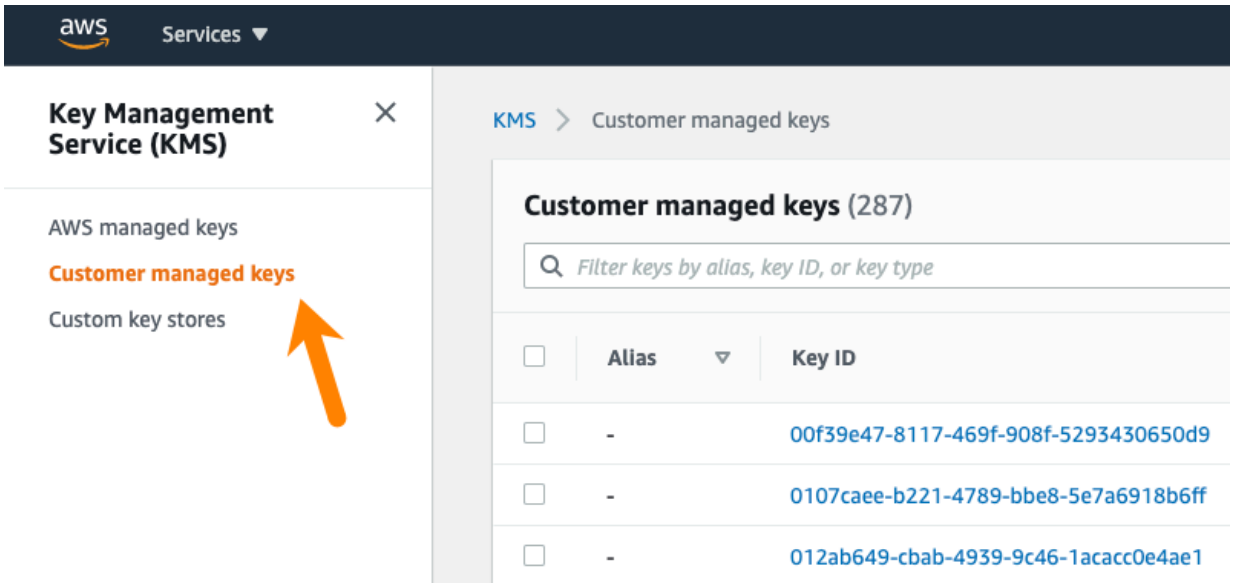

   On the Customer managed keys page, select the key you want to use.

6. On the key details page, select the Key policy tab in the center panel of the page:

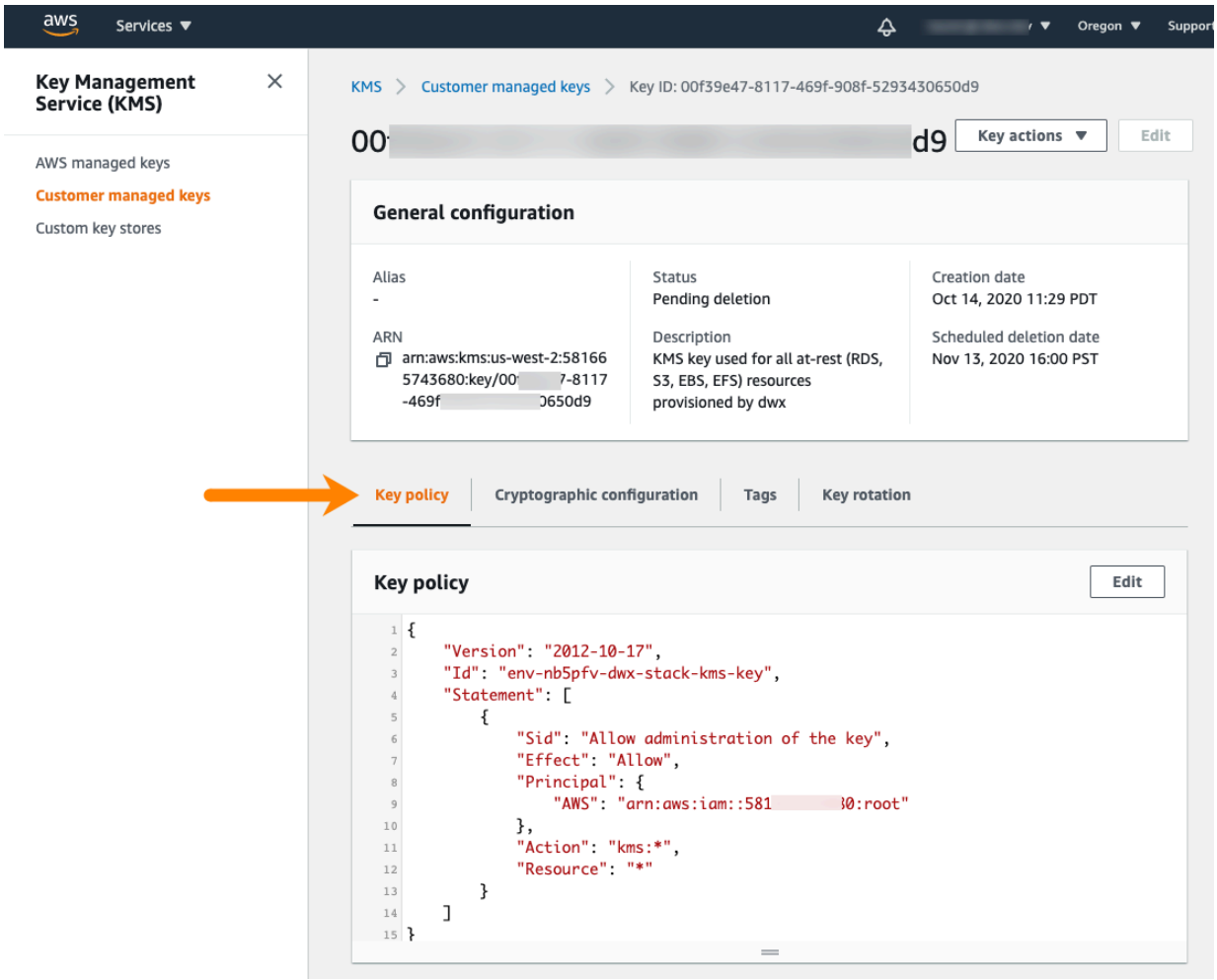

This displays the key policy for the customer managed key.

7. In the Cloudera Data Warehouse Environment Details page, copy the Amazon Resource Names (ARNs) associated with the environment that displays in the panel:

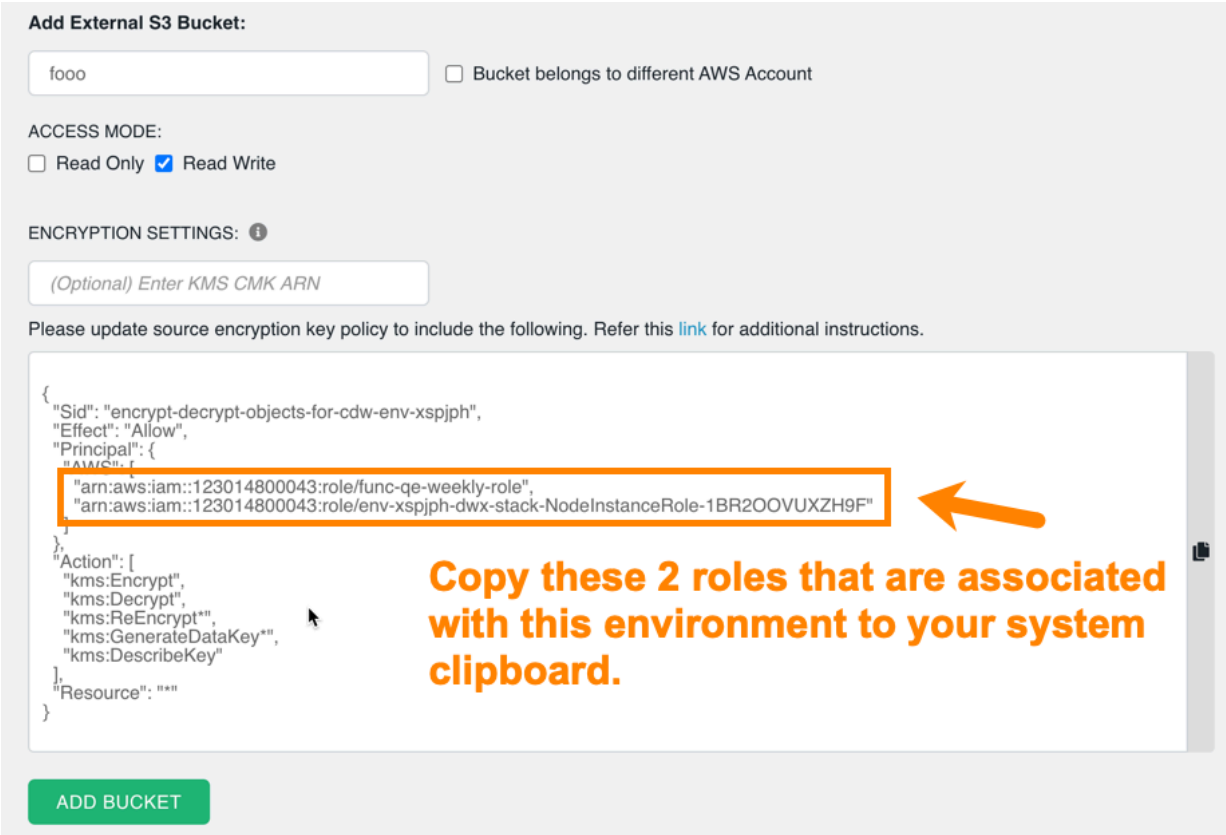

The actions listed in the above screen capture are the minimum set of actions needed by Cloudera Data Warehouse:

```
"Action:[
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
]
```

The key policy you use should allow at least these actions.

8. Return to the key details page in the AWS Management Console, click Edit in the upper right corner of the Key policy tab, paste the two ARNs to append them after the existing ARNs in the key policy, and then click Save changes:

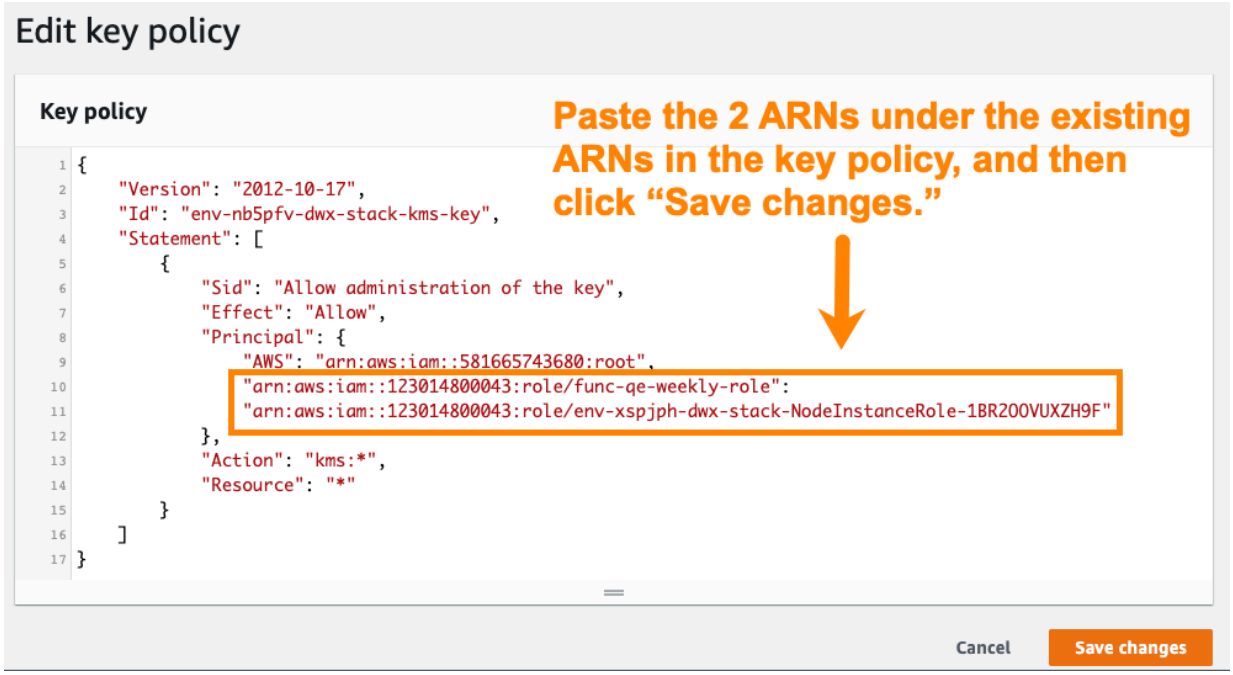

The key policy shown in the above screen capture lists "kms:*" in the "Action" section of the policy. This indicates that all actions are allowed. At minimum, Cloudera Data Warehouse requires the "Encrypt", "Decrypt", "ReEncrypt", "GenerateDataKey*", and the "DescribeKey" actions as shown on the screen capture in Step 7.

If there is no Key Policy tab of the key details page, copy and paste the entire key policy in the AWS Management Console from the Cloudera Data Warehouse UI.

9. After you save the changes to the key policy in the AWS Management Console, copy the ARN from the General configuration section of the key details page:

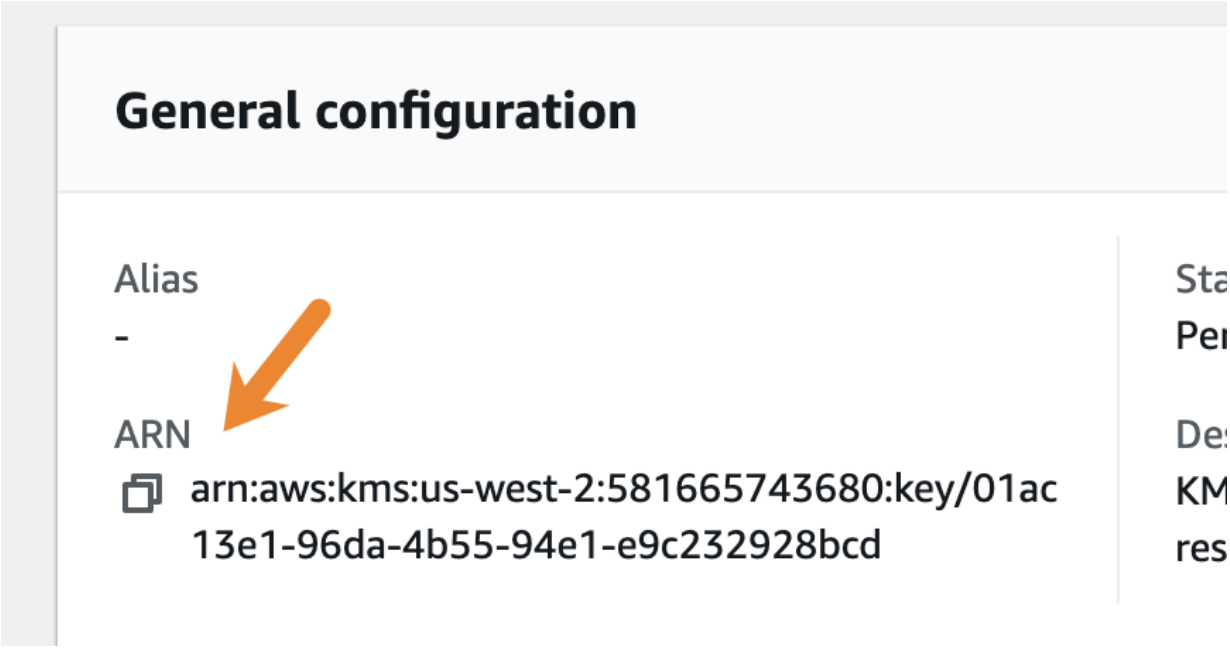

10. In the Cloudera Data Warehouse Environment Details page, add the ARN you copied in Step 9 to the ENCRYPTION SETTINGS text box:

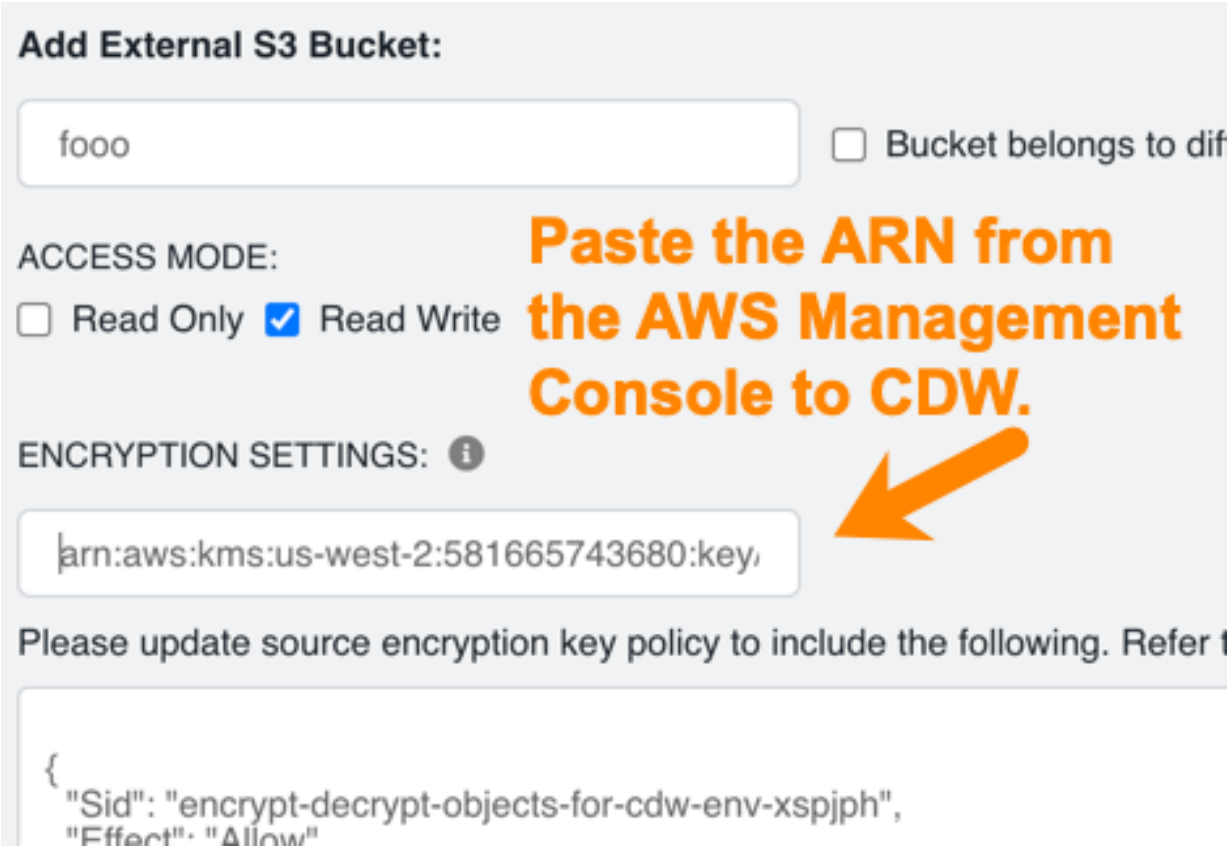

11. Click Add bucket to save the configuration. A success message displays at the top of the page.

What to do next

You must restart the Virtual Warehouses that are associated with this environment for the configuration changes to take effect.

Accessing buckets in a different AWS account

You must configure read-only or read/write access using default encryption to external S3 buckets in AWS accounts that are different from the Cloudera Data Warehouse cluster account.

About this task

Important: If you configure read-only access to an external S3 bucket, there is no need to restart Virtual Warehouses. However, if you configure read/write access to an external S3 bucket, you must restart Virtual Warehouses by suspending them and starting them again. Alternatively, you can create a new Virtual Warehouse to use the external S3 bucket with read/write access.

Required role: DWAdmin

Before you begin

- Identify and activate the environment you want to configure for access to an external bucket in a different AWS account.
- In the AWS Management Console, identify the external S3 bucket you want to configure access to.

Procedure

1. On the Cloudera Data Warehouse UI **Overview** page, go to the Environments tab and locate the environment for which you want to configure access to an external AWS bucket, and then click ⋮ Edit .
   This loads the **Environment Details** page.
2. Go to the Configuration tab and type the name of the AWS bucket you want to configure access to in the Add External S3 Bucket field.
3. Select Bucket belongs to different AWS Account. The Cloudera Data Warehouse bucket policy appears.
4. Click Copy .

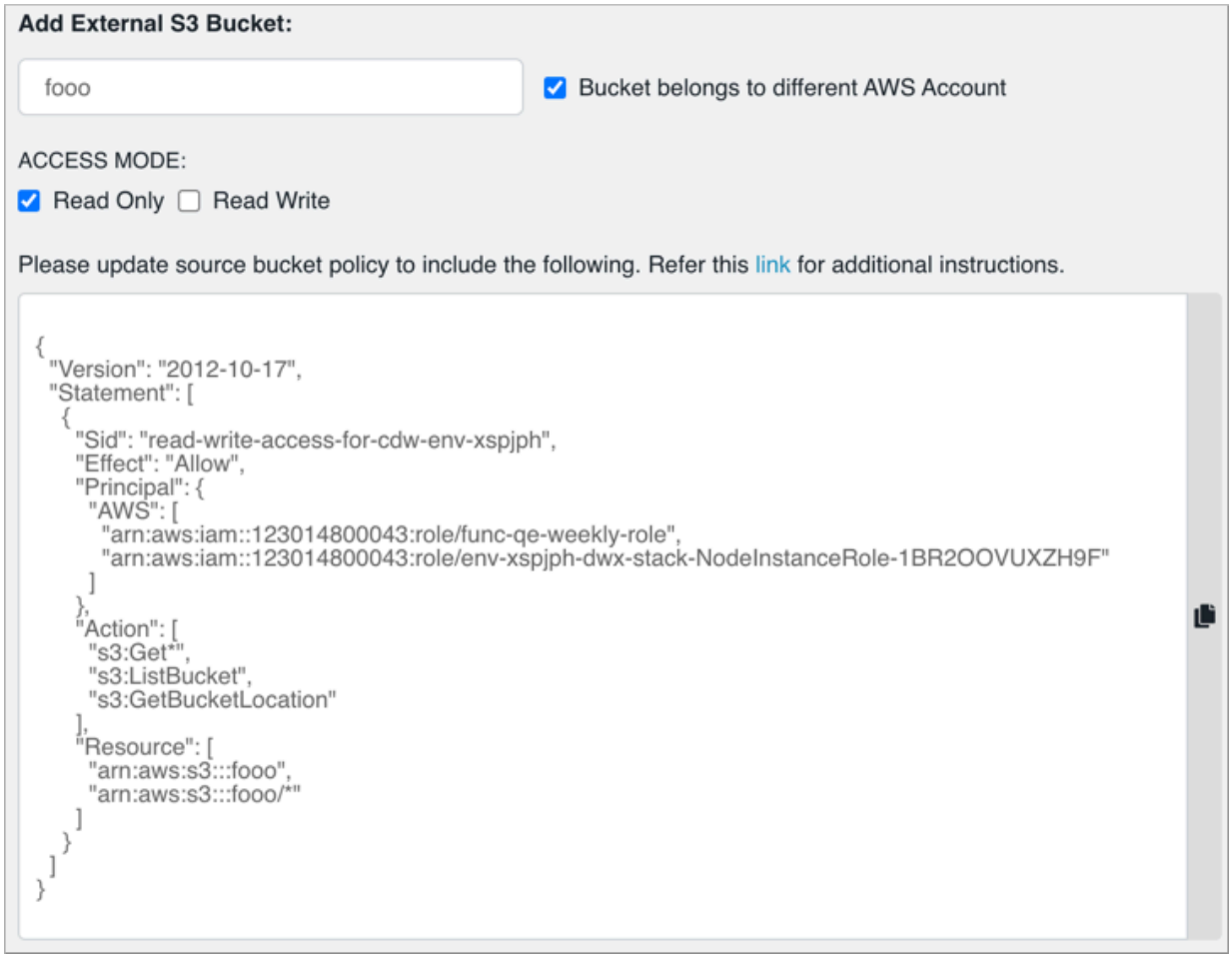

5. Open the AWS Management Console for the different account where the external bucket is located and navigate to the bucket to which you want to configure access.

6. On the bucket details page of AWS Management Console, click the Permissions tab, click Bucket Policy, paste the policy from Cloudera Data Warehouse, and click Save:

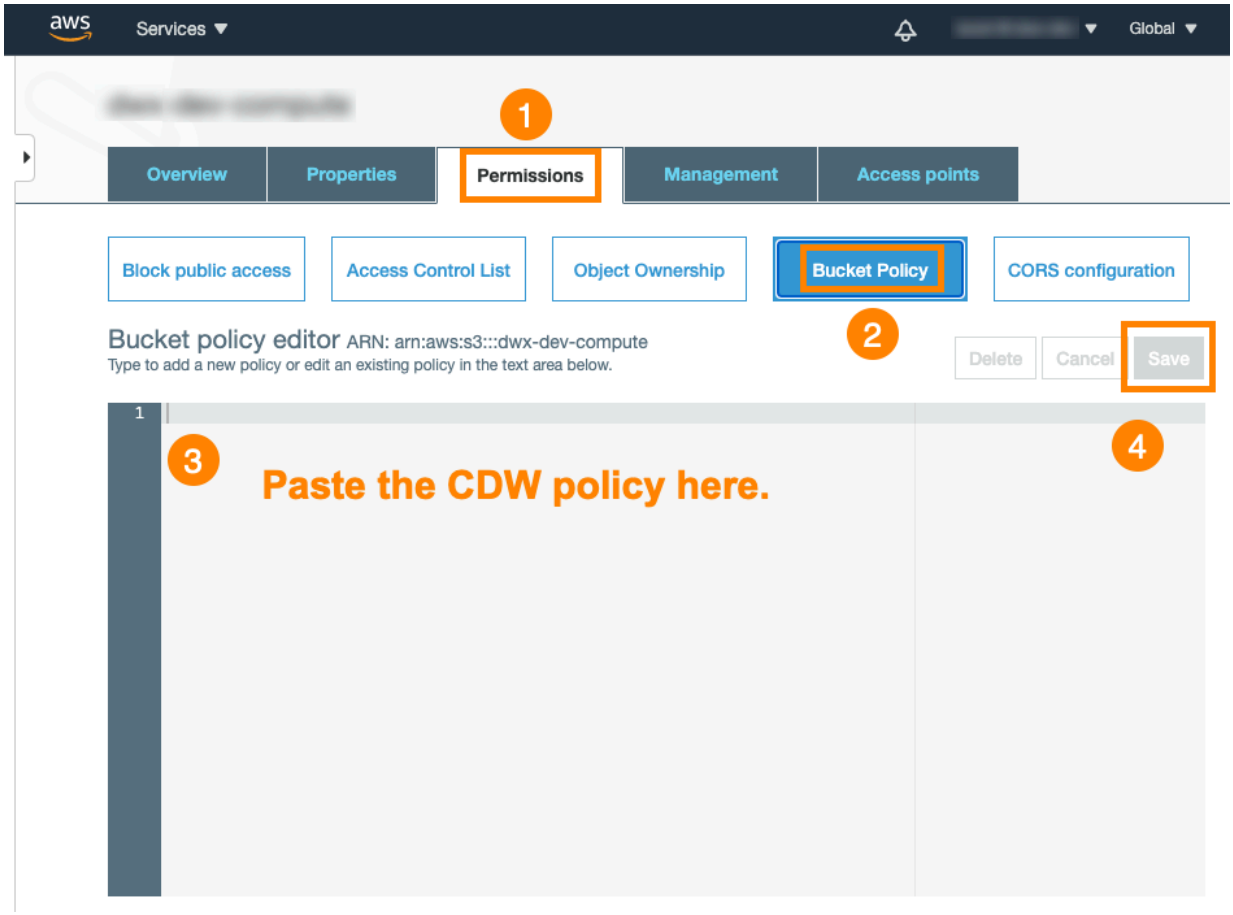

7. In the Cloudera Data Warehouse UI Environment Details page, specify either Read Only or Read Write access for the external bucket.

Note:

For Read Write access only: if you do not need to use a custom encryption key, leave the ENCRYPTION SETTINGS text box blank. If you do not use a custom key, the system uses AES256 encryption by default. If you need to use a custom encryption key, see the previous topic. Read Only access does not involve encryption.

8. Click Add Bucket to save the configuration. A success message displays at the top of the page.

What to do next

If you have configured Read Write access, you must restart the Virtual Warehouses that are associated with this environment for the configuration changes to take effect.

Accessing S3 buckets in a managed policy environment

You can add and query data in S3 external buckets you add to Cloudera Data Warehouse service clusters running on AWS environments.

This topic and the following subtopics are applicable if you are using managed policy ARN feature. If you are using RAZ enabled DW, see "Accessing buckets in a RAZ environment".

If you are not using a RAZ-enabled Cloudera Data Warehouse, access to S3 buckets is controlled by the managed policy attached to the AWS instance profiles. To access S3 buckets you add to your Cloudera Data Warehouse service

cluster, you must edit the instance profile to add read/write permissions to the additional buckets as described in the following subtopics.

Related Information

Accessing S3 buckets in a RAZ environment

Accessing buckets in the same AWS account under a managed policy

In certain scenarios, you might need to interact with data that resides outside of the data lake S3 buckets. You can add a bucket to S3, enable access to the bucket, and then, define external tables based on the data, such as a CSV file, you put into the bucket.

About this task

In this task, you see how to add access to an S3 bucket to create an external table based on CSV data using Hue. You need to enable read/write access to the external S3 bucket before creating the table. From the command line of your cluster, you can run HDFS CLI commands on the S3 bucket. You can also use the S3 bucket for uploading a UDF jar for registration, and then include UDFs in queries from your cluster.

Note: This topic might describe versions of the AWS console that have changed over time.

Before you begin

- The S3 bucket you add to hold the data outside your Data Lake must be in the same AWS account as your Cloudera Data Warehouse service cluster.
- Required role: DWAdmin

About this task

In this task, first you note the Managed Policy ARN attached to the Node Instance Role used while activating the cluster. Next, you edit the managed policy in the JSON file, for example noderole-inline-policy.json.

Procedure

1. In your managed policy, locate the sid "putgetmybucketpaths" for editing.

```
"Sid": "putgetmybucketpaths",
"Action": [
    "s3:Get*",
    "s3:Delete*",
    "s3:Put*",
    "s3:ListBucketMultipartUploads",
    "s3:AbortMultipartUpload"
],
"Resource": [
    "arn:aws:s3:::roohi-dwx-priv/clusters",
    "arn:aws:s3:::roohi-dwx-priv/clusters/*",
    "arn:aws:s3:::roohi-dl-bucket/clusters",
    "arn:aws:s3:::roohi-dl-bucket/clusters/*",
    "arn:aws:s3:::roohi-dwx-priv/logs",
    "arn:aws:s3:::roohi-dwx-priv/logs/*",
    "arn:aws:s3:::roohi-backup/backup",
    "arn:aws:s3:::roohi-backup/backup/*",
    "arn:aws:s3:::roohi-dl-bucket/data",
    "arn:aws:s3:::roohi-dl-bucket/data/*",
    "arn:aws:s3:::roohi-dl-bucket/backup",
    "arn:aws:s3:::roohi-dl-bucket/backup/*",
    "arn:aws:s3:::roohi-dl-bucket/tmp",
    "arn:aws:s3:::roohi-dl-bucket/tmp/*"
],
"Effect": "Allow"
```

2. Append resources to the resource section for the buckets you added.
   For example, you added a bucket more-sales-data. To enable access to the more-sales-data bucket, you append resources to the end of the "resource" section, as shown in the last two resource names:

```
"Resource":[
...
"arn:aws:s3:::roohi-dl-bucket/backup/*",
    "arn:aws:s3:::more-sales-data",
    "arn:aws:s3:::more-sales-data/*"
 ],
```

   Important: There is no comma after the last line of the "Resource" section.

3. Click Review policy in the lower right corner of the page, and then click Save changes.
   You can now access the more-sales-data bucket outside your data lake from Hue in your Cloudera Data Warehouse service cluster. For example, you can create external Hive tables that point to the bucket, and join those external tables with tables already in your data lake. You can govern Cloudera Data Warehouse user access to this external S3 bucket using Ranger Hadoop SQL Policies.

Accessing buckets in a different AWS account under a managed policy

You might need to know how to add read/write access to S3 buckets under AWS accounts that are different from the Cloudera Data Warehouse cluster account.

About this task

To enable Cloudera Data Warehouse service cluster access to a bucket you add to S3 under a different AWS account, you must configure the bucket in the different account to access the Cloudera Data Warehouse cluster account. Then, you can configure the Cloudera Data Warehouse service account to access the bucket you added. You perform both of these tasks in the AWS Management Console.

Required role: DWAdmin

Note: This topic might use both "new" and "old" versions of the AWS console.

Before you begin

To configure access to external S3 buckets for your Cloudera Data Warehouse cluster, you must edit the managed policy attached to the AWS instance profile.

You use this cluster ID in Step 5 below.

Procedure

1. In the AWS Console, navigate to AWS Management Console S3 , locate the bucket in the other AWS account you added, and then click the bucket name.
2. In the bucket details page, click the Permissions tab, and then click the Bucket Policy sub-tab.

3. In the Bucket Policy sub-tab page, in the Bucket policy editor, add the Cloudera Data Warehouse cluster Id and what permissions you want the Cloudera Data Warehouse service account to have for this bucket:

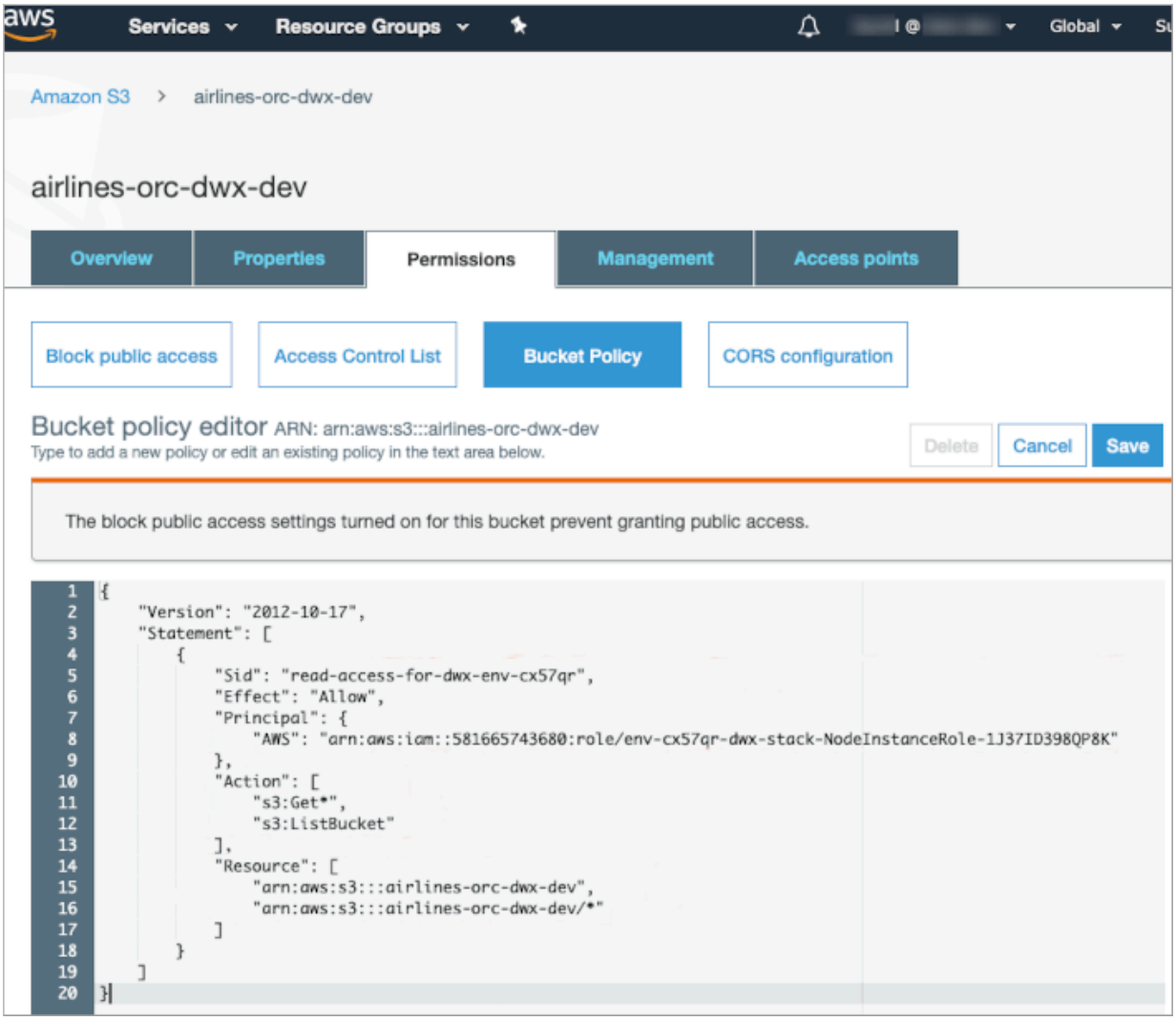

This example policy includes the following specifications:

- **Tip:** To get the ARN of your Cloudera Data Warehouse cluster account: In the AWS Console, navigate to CloudFormation and locate the stack for your Cloudera Data Warehouse cluster. Click the stack name. In the stack details page, click the Resources tab. In the Resources table, scroll down to the NodeInstanceRole, and then click the hyperlink just to the right of it. At the top of the Summary page, the Role ARN is listed. This is the ARN you must specify for the Principal in the bucket policy.
- The Action section specifies what actions the Principal can perform.
- The Resource section specifies the S3 bucket you added and want your Cloudera Data Warehouse cluster to be able to access.

For details about bucket policies, see Managing Access to Amazon S3 Buckets Using Bucket Policies in the AWS documentation.

4. Click Save.
5. Note the managed policy ARN attached to the Node Instance Role, used while activating the cluster.
6. Open the managed policy JSON file, for example noderole-inline-policy.json for editing

7. Locate the sid "putgetmybucketpaths" for editing.

```
"Sid": "putgetmybucketpaths",
"Action": [
    "s3:Get*",
    "s3:Delete*",
    "s3:Put*",
    "s3:ListBucketMultipartUploads",
    "s3:AbortMultipartUpload"
],
"Resource": [
    "arn:aws:s3:::roohi-dwx-priv/clusters",
    "arn:aws:s3:::roohi-dwx-priv/clusters/*",
    "arn:aws:s3:::roohi-dl-bucket/clusters",
    "arn:aws:s3:::roohi-dl-bucket/clusters/*",
    "arn:aws:s3:::roohi-dwx-priv/logs",
    "arn:aws:s3:::roohi-dwx-priv/logs/*",
    "arn:aws:s3:::roohi-backup/backup",
    "arn:aws:s3:::roohi-backup/backup/*",
    "arn:aws:s3:::roohi-dl-bucket/data",
    "arn:aws:s3:::roohi-dl-bucket/data/*",
    "arn:aws:s3:::roohi-dl-bucket/backup",
    "arn:aws:s3:::roohi-dl-bucket/backup/*",
    "arn:aws:s3:::roohi-dl-bucket/tmp",
    "arn:aws:s3:::roohi-dl-bucket/tmp/*"
],
"Effect": "Allow"
```

8. Append resources to the resource section for the buckets you added.
   For example, if you want to add access to the more-sales-data bucket, you append resources to the end of the "resource" section, as shown in the last two resource names:

```
"Resource":[
            ...
            "arn:aws:s3:::roohi-dl-bucket/backup/*",
            "arn:aws:s3:::more-sales-data",
            "arn:aws:s3:::more-sales-data/*"
            ],
```

   Important: There is no comma after the last line of the "Resource" section.

9. Click Review policy in the lower right corner of the page, and then click Save changes. You can access the new bucket from your Cloudera Data Warehouse service cluster now. For example, you can create external Hive tables that point to the bucket.

Accessing S3 buckets in a RAZ environment

In a RAZ (Ranger Authorized) environment, you must configure permissions to access an S3 bucket. The procedures for configuring the permissions differ depending on the AWS account that owns the bucket.

If you have enabled RAZ in your environment, policies attached to the Ranger RAZ Service role control access to external S3 buckets.

Prerequisites

You must meet the following prerequisites before adding access permissions to buckets to the RAZ environment within the same AWS account or in a different account:

- Obtain the DWAdmin role.
- Follow steps similar to the minimum setup for cloud storage to create the Ranger RAZ role.
- Register an environment with RAZ using the Cloudera web interface.

  In the web interface, in Fine-grained access control on S3, select Enable Ranger Authorization for AWS S3.

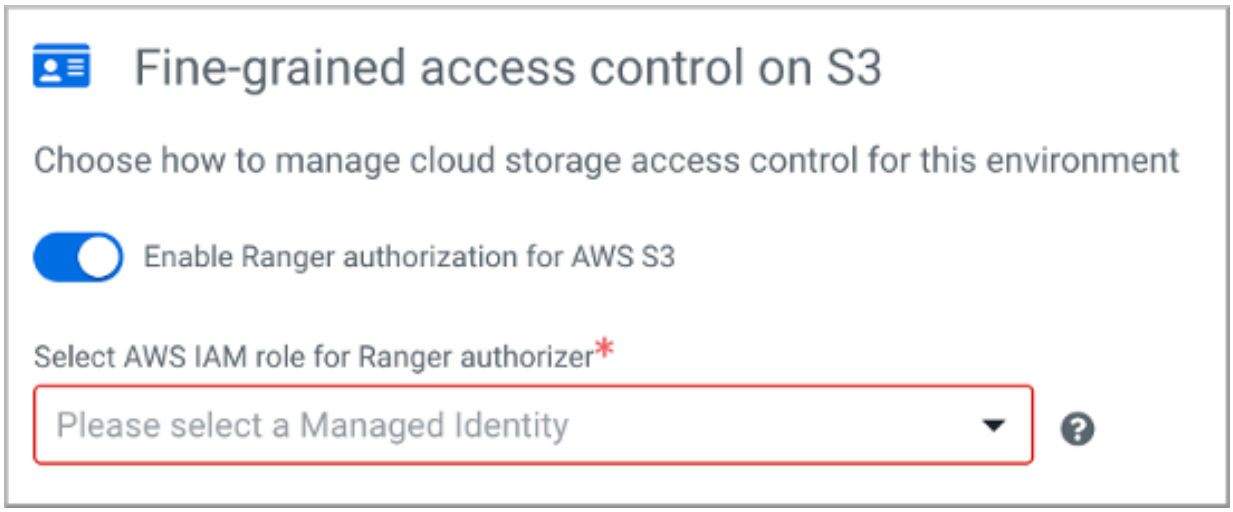

Accessing buckets in the same AWS account under RAZ

In a Ranger Authorized (RAZ) environment, to access an S3 external bucket in the same AWS account as the cluster, you must configure a policy that manages access to the Cloudera data lake. You modify a JSON file that defines the IAM policy to do this.

About this task

You update the following IAM policy definition for the minimal cloud storage setup: aws-cdp-datalake-admin-S3-policy.

Before you begin

You must meet the prerequisites mentioned in the topic above.

Procedure

Add the ARN of the external bucket to the "Resource" array of values in the JSON file for the aws-cdp-datalake-admin-S3-policy.

For example, the values in the Resource array give the Cloudera Data Warehouse cluster access to external bucket MY_EXTERNAL_BUCKET.

```
...
            "s3:ListBucketMultipartUploads",
            "s3:ListBucketVersions",
            "s3:ListMultipartUploadParts",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:${ARN_PARTITION}:s3:::${STORAGE_LOCATION_BASE}",
            "arn:${ARN_PARTITION}:s3:::${STORAGE_LOCATION_BASE}/*",
            "arn:${ARN_PARTITION}:s3:::MY_EXTERNAL_BUCKET",
            "arn:${ARN_PARTITION}:s3:::MY_EXTERNAL_BUCKET/*"
        ]
    }
  ]
}
```

Accessing buckets in a different AWS account under RAZ

In a Ranger Authorized (RAZ) environment, to access an S3 external bucket in a different AWS account from the Cloudera Data Warehouse cluster, you must configure the bucket policy of the other account. You use the AWS Management Console to do this.

About this task

For information about bucket policies, see Adding a bucket policy by using the Amazon S3 console in the AWS documentation.

Before you begin

You must meet the prerequisites mentioned in the topic above.

Procedure

1. Add the ARN of the external bucket to the "Resource" array of values in the JSON file for the aws-cdp-datalake-admin-s3-policy.
   For example, the values in the Resource array give the Cloudera Data Warehouse cluster access to external bucket MY_EXTERNAL_BUCKET.

```
...
            "s3:ListBucketMultipartUploads",
            "s3:ListBucketVersions",
            "s3:ListMultipartUploadParts",
            "s3:PutObject"
          ],
          "Resource": [
            "arn:${ARN_PARTITION}:s3:::${STORAGE_LOCATION_BASE}",
            "arn:${ARN_PARTITION}:s3:::${STORAGE_LOCATION_BASE}/*",
            "arn:${ARN_PARTITION}:s3:::MY_EXTERNAL_BUCKET",
            "arn:${ARN_PARTITION}:s3:::MY_EXTERNAL_BUCKET/*"
          ]
        }
      ]
    }
```

2. Get the cluster ID from the Environments tile in the Cloudera Data Warehouse service UI.
   For example:

   You use this cluster ID later.
3. In the AWS Console, navigate to AWS Management Console S3 , locate the bucket in the other AWS account you added, and then click the bucket name.
4. In the bucket details page, click Permissions, and then click Bucket Policy.

5. In Bucket Policy, in the Bucket policy editor, add the Cloudera Data Warehouse cluster Id and the permissions you want the Cloudera Data Warehouse service account to have to this bucket.

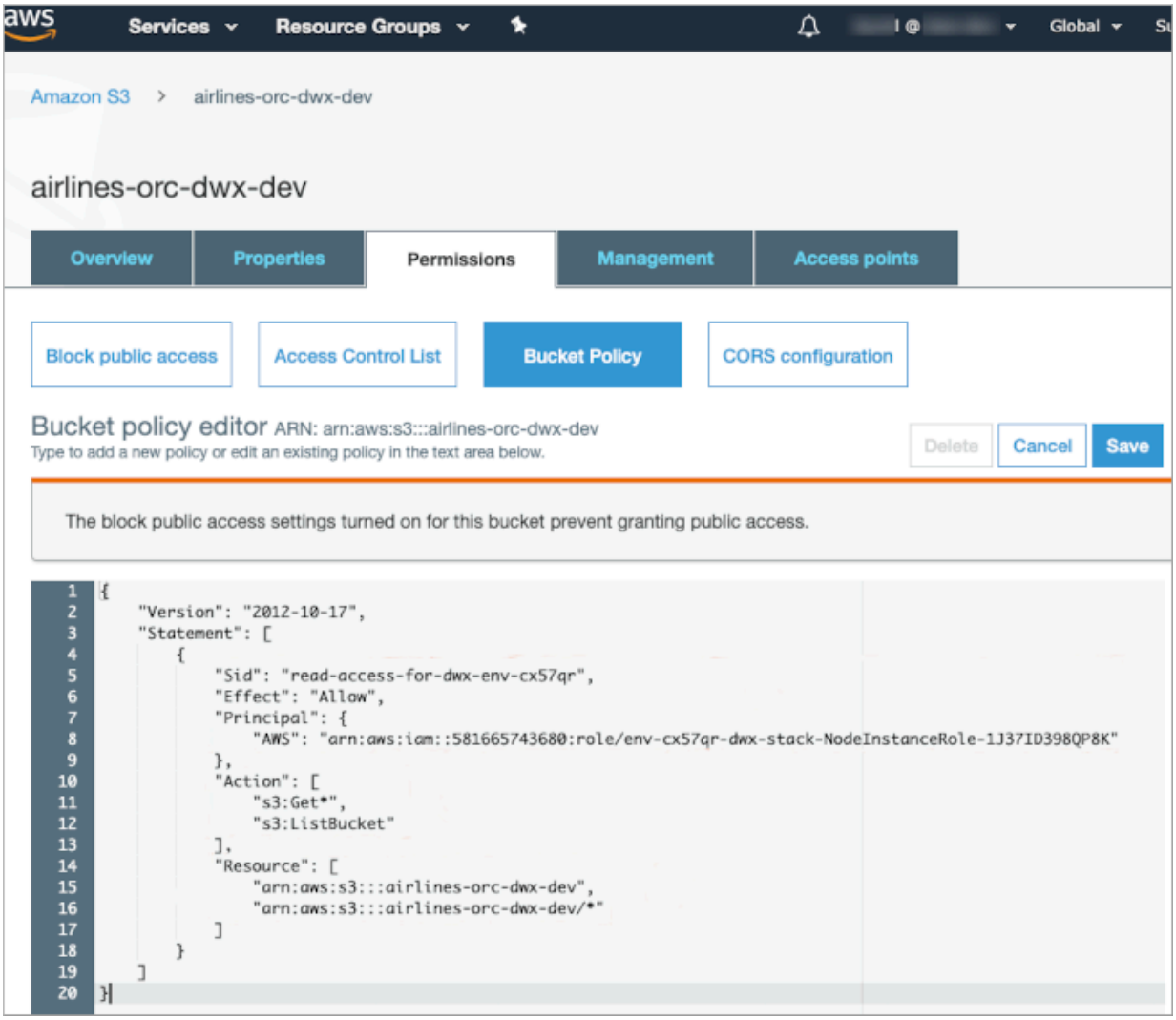

This example policy includes the following specifications:

- This first section includes the Sid, which is an optional identifier indicating what the policy does. The Effect specifies that this policy is allowing the Principal to do what is listed below in the Action section. The Prin cipal is where you specify the ARN of the instance role for your Cloudera Data Warehouse cluster account.

  Tip: To get the ARN of your Cloudera Data Warehouse cluster account:Follow steps 6-8 below, then specify the ARM for the Principal in the bucket policy.

- The Action section specifies what actions the Principal can perform.
- The Resource section specifies the S3 bucket you added that your Cloudera Data Warehouse cluster will access.

6. Navigate to AWS Management Console > CloudFormation and locate the stack corresponding to the cluster ID.

7. Click the CloudFormation stack name. This stack name is the one in this format: <cluster-ID>-dwx-stack. For example, if the cluster ID is env-6cwwgg, the CloudFormation stack name for this cluster is env-6cwwgg-dwx-stack.

   In CloudFormation stack details, in Resources, the NodeInstanceRole appears in the Logical ID column.

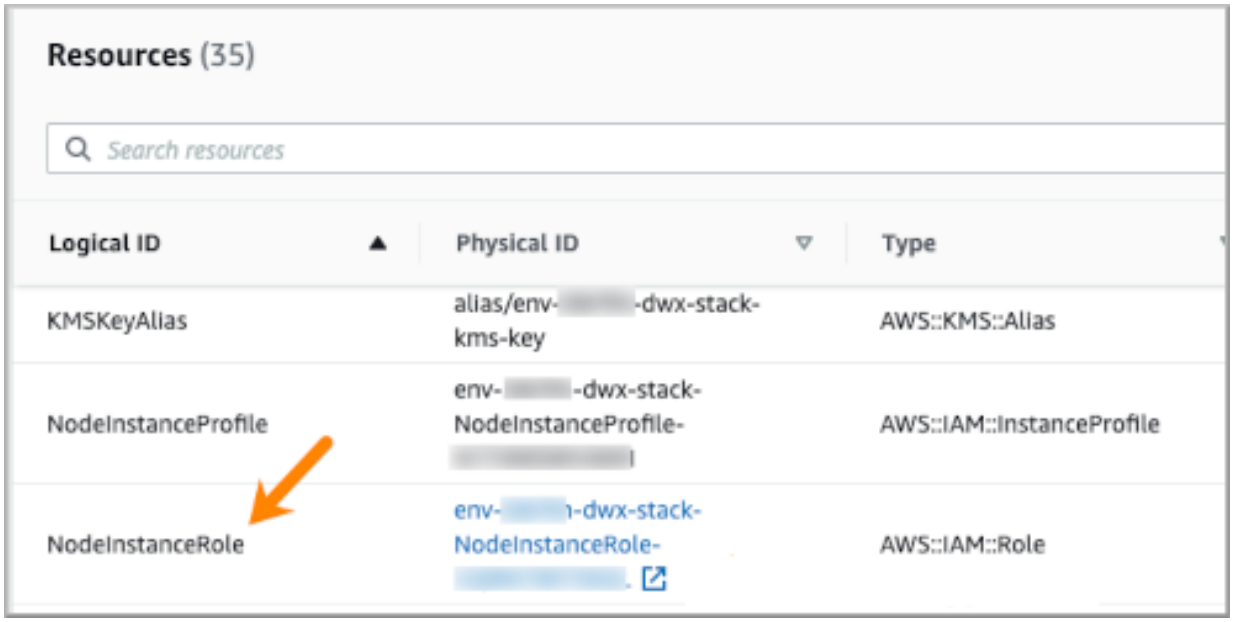

8. Click the hyperlink just to the right of it.
   At the top of the Summary page, the Role ARN is listed.
9. Specify the ARN for the Principal in the bucket policy.
10. Click Save.

Enabling RAZ manually

You might want to manually enable Ranger Authorization (RAZ) if you have a Cloudera Data Warehousethat predates the capability to enable RAZ for AWS S3. In this case, if you need Ranger authorization in Hue, perform this task.

About this task

For an introduction to enabling RAZ, see the Cloudera Management Console documentation.

When you activate your environment in 1-6.3-b319 (released April 5, 2023) or later, you configure S3 access using the Fine-grained access control on S3 dialog in the Cloudera Management Console UI. If your Data Lake is RAZ-enabled (Enable Ranger authorization for AWS S3), Cloudera Data Warehouse is RAZ-enabled by default, and cannot be turned off. Cloudera Data Warehouse will be RAZ-enabled to provide authorization mainly in Hue. There is nothing more you need to do.

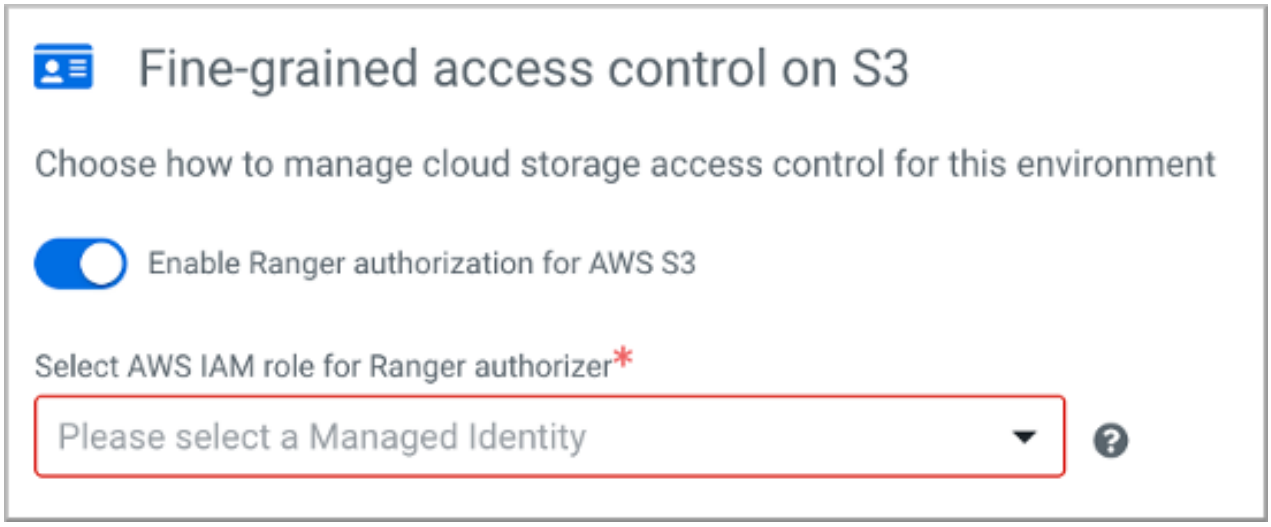

If you activated your Cloudera Data Warehouse environment in 1-6.2-b197 (released Feb 14, 2023) or earlier, your Cloudera Data Warehouse is not RAZ-enabled (Enable Ranger authorization for AWS S3 is not enabled). You can manually enable RAZ for Cloudera Data Warehouse only if the Data Lake is RAZ-enabled. You follow the steps below to manually enable RAZ for AWS S3 mainly for authorization of Hue users.

Before you begin

Your Data Lake must be RAZ-enabled before you can perform these steps.

Procedure

1. Obtain a list of the permissions policies related to the AWS IAM role for Ranger authorizer you created on AWS.

   In some cases, this role is interchangeable with the DATA_LAKE_ADMIN role for AWS S3. For information about this role and policies, see "Required IAM resources".

   Minimal permissions policies are:

   - aws-cdp-datalake-admin-s3-policy
   - aws-cdp-bucket-access-policy
   - aws-datalake-backup-policy
   - aws-datalake-restore-policy
2. Get the cluster ID from the Environments tile in the Cloudera Data Warehouse service UI.
3. Navigate to AWS Management Console CloudFormation and locate the stack corresponding to the cluster ID.

4. Click the CloudFormation stack name. This stack name is the one in this format: <cluster-ID>-dwx-stack. For example, if the cluster ID is env-6cwwgg, the CloudFormation stack name for this cluster is env-6cwwgg-dwx-stack.

   In CloudFormation stack details, in Resources, the NodeInstanceRole appears in the Logical ID column.

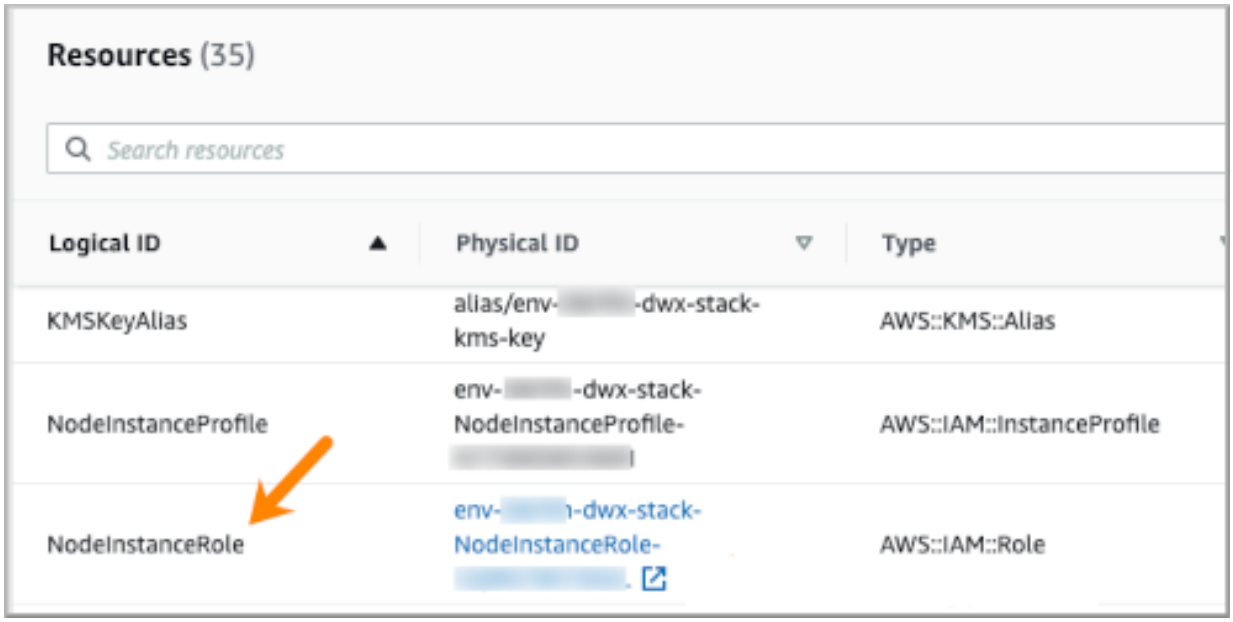

5. Click the NodeInstanceRole link.
   The Node instance Role page appears.
6. Click Add Permissions Attach Policies , and on the next page, in Permissions Policies, select the policies from step 1.

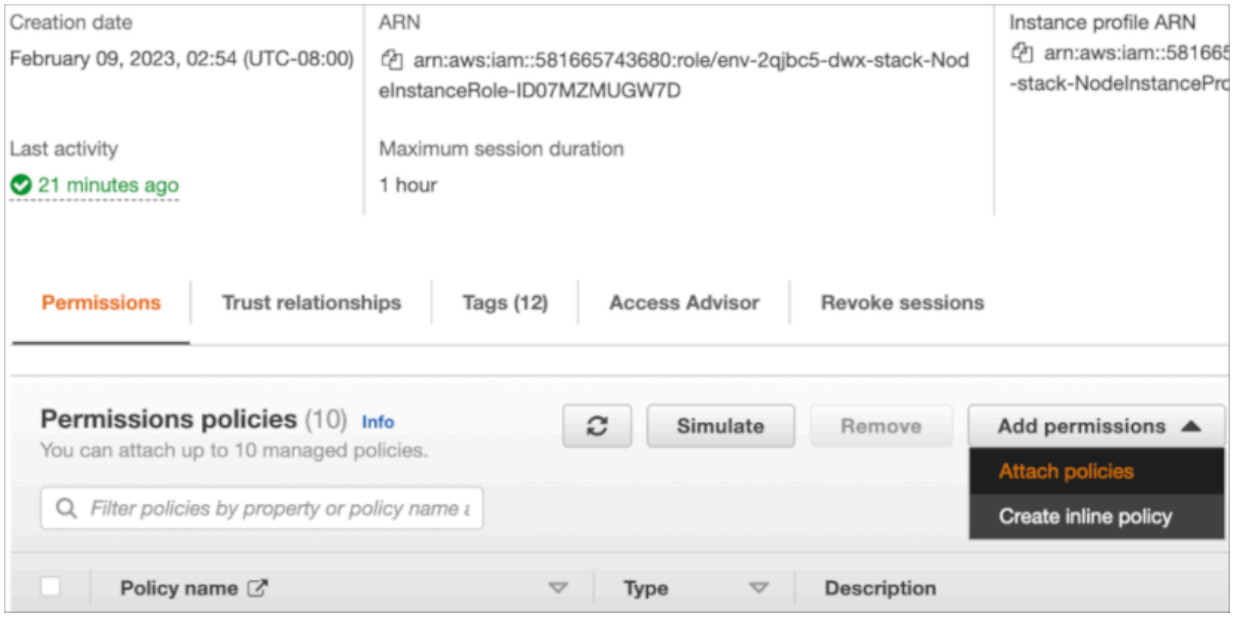

Remote access

You can follow a few steps to grant users access to Cloudera Data Warehouse service Kubernetes clusters on Amazon EKS. A step-by-step procedure explains how to restrict access to Kubernetes endpoints and service endpoints of the Kubernetes cluster.

Granting remote access to Kubernetes clusters on Amazon EKS

This topic describes how you can grant users access to Cloudera Data Warehouse service Kubernetes clusters on Amazon EKS.

About this task

To grant remote access to Kubernetes clusters on Amazon EKS, add the Amazon Resource Name identifiers (ARNs) to the list of trusted users.

Required role: DWAdmin

Before you begin

- You must activate an environment before you can grant users access to the Kubernetes cluster.
- Contact your AWS account administrator or the user who is requesting access to the Kubernetes cluster on AWS to get the ARN identifier for their Amazon account:

  Using the Amazon Management Console

  1. On the Amazon Management Console home page, enter IAM in Find Services, and then select IAM in the search results.
  2. On the Identity and Access Management home page, in the left navigation menu, select Users.
  3. On the User page, locate the user and click or tap their User name.
  4. On the Summary page for the user, their User ARN is listed at the top. Copy it to paste into the Data Warehouse service UI in Step 4 below.

  Using the AWS CLI

  As an alternative to copying the User ARN from the AWS Management Console, you can also ask the user who is requesting access to enter the following command in the AWS CLI:

  ```
  aws sts get-caller-identity
  ```

  This command evokes the AWS Security Token Service (sts) and returns the following type of information on the requesting user:

  ```
  #Sample output
  {
  "UserId": "ABCDE12345FGHIJKLMNO6789",
  "Account": "888888888888",
  "Arn": "arn:aws:iam::888888888888:user/<USERNAME>"
  }
  ```

  A link to the Amazon documentation on this command is available at the bottom of this page.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Environments tab from the **Overview** page.

3. Click ⋮ Edit GROUP ACCESS corresponding to the environment for which you want to grant access to ARNs.

4. Enter the ARN identifier under Add new group and click Grant Access.

Related Information

Activating environments

Amazon documentation on the AWS CLI command 'aws sts get-caller-identity'

Revoking remote access to Kubernetes clusters on Amazon EKs

This topic describes how to revoke remote access to Kubernetes clusters for Cloudera Data Warehouse Public Cloud.

About this task

Revoke users' access to Cloudera Data Warehouse service Kubernetes clusters on Amazon EKS by deleting their Amazon Resource Name identifiers (ARNs) from the list of trusted users.

Required role: DWAdmin

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment for which you want to revoke user's access and click ⋮ Edit GROUP ACCESS .

   The Amazon Resource Name (ARN) identifiers that have been granted access to the Kubernetes cluster are displayed.
3. Click Remove Access to revoke user access.
4. Click Apply Changes.

Restricting access to endpoints in AWS environments

This topic describes how to limit access to Hive, Impala, Data Analytics Studio, or Hue endpoints in Cloudera Data Warehouse Public Cloud.

About this task

You can restrict access to Kubernetes endpoints and load balancer endpoints of the Kubernetes cluster by specifying a list of IP Classless Inter-Domain Routing (CIDRs) that are allowed access. Kubernetes endpoints are used to control the deployment and maintenance of workload components, such as Virtual Warehouses and Database Catalogs. Load balancer endpoints are endpoints of services like Hive, Impala, or Hue. You can specify trusted IP addresses when you activate a Public Cloud environment to use in the Data Warehouse service or in the Environment Details page.

Otherwise, all external IP addresses can access these endpoints on the Kubernetes cluster that is being used in the Cloudera Data Warehouse service.

Required role: DWAdmin

Before you begin

Contact your network team to get your internal network's IP CIDR ranges of IP addresses that need access to Kubernetes and load balancer service endpoints. All Cloudera IP addresses that need access to these endpoints have already been allowed.

Procedure

1. In the Cloudera Data Warehouse service, in Environments, search for and locate the environment that you want to specify CIDRs.
2. Click Activate to activate the environment.
3. In Activation Settings Advanced Settings, in Enable IP CIDR for Kubernetes cluster specify a comma-separated list of IP CIDRs that you want to be able to access your Kubernetes endpoints:
4. In Activation Settings Advanced Settings, in Enable IP CIDR for the load balancer specify a comma-separated list of IP CIDRs that you want to be able to access your load balancer endpoints:

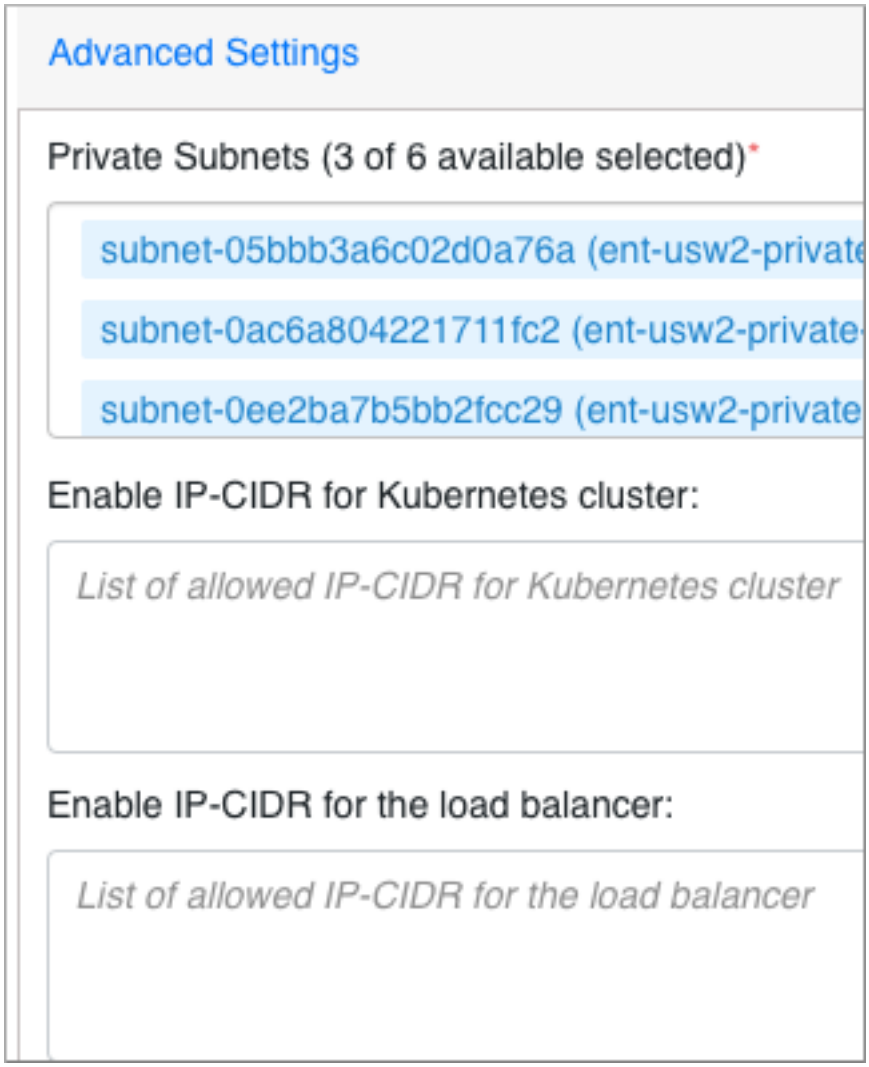

5. After specifying the IP CIDRs, click Activate.

Related Information

CIDR (Classless Inter-Domain Routing)

Editing the IP CIDRs in the trusted list for endpoints in AWS environments

This topic describes how to change access to Hive, Impala, or Hue service endpoints in Cloudera Data Warehouse Public Cloud.

About this task

After you have added IP CIDRs to the trusted list so they have access to Kubernetes endpoints and the load balancer endpoints of services such as Hive, Impala, or Hue, you can edit the lists of IP addresses to change access to the endpoints.

Required role: DWAdmin

Before you begin

You must activate an environment before you can view its details or edit the IP CIDRs that have been added to the trusted list. See "Activating environments" in the "Related information" section at the bottom of this page.

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment where you want to edit the IP CIDRs that have been specified as part of a trusted list and click ⋮ Edit .

   This launches the Environment Details page.
3. Go to the Configurations tab, edit the comma-separated list of IP CIDRs: Enable IP CIDR(s) for Kubernetes cluster and Enable IP CIDR(s) for the load balancer.
4. Click Apply Changes.

Related Information

Activating environments

Networking

Learn how to increase the available IP addresses for Cloudera Data Warehouse Public Cloud using overlay networks, use a network proxy in Cloudera Management Console, and set up private networking.

Overlay networks for AWS environments in Cloudera Data Warehouse service

This topic explains how you can use overlay networks to increase the number of available IP addresses for your deployments of Cloudera Data Warehouse Public Cloud.

If you have an existing Virtual Private Cloud (VPC) where you want to deploy a Virtual Warehouse, you must make sure that there is an adequate number of available IP addresses. Otherwise, the Cloudera Data Warehouse service cannot be used efficiently and some features like auto-scaling do not work because the number of IP addresses might become exhausted. For example, each executor node in a Virtual Warehouse uses 8 IP addresses. If you create a SMALL-sized Virtual Warehouse with 2 executor nodes, it uses 16 IP addresses. If you create a LARGE-sized Virtual Warehouse with 40 executor nodes, it uses 320 IP addresses. If your VPC is small, the available IP addresses can be consumed quickly. Using overlay networks solves this issue.

Important:

- Use this feature if your VPC subnet has fewer than 1,024 IP addresses.
- Overlay networks are configured to operate up to 200 executor nodes by default. Cloudera recommends that you do not exceed this limit or networking limitation might occur.

About overlay networks

An overlay network is a software-defined layer of network abstraction that is used to run multiple separate, discrete virtualized network layers over the VPC network. In the case of the Cloudera Data Warehouse service, a custom CNI (Container Network Interface) plugin is used to enable the overlay network. It creates two network spaces:

- A node network space, which derives per-node IP addresses from the VPC.
- A Kubernetes pod network space, which derives per-pod IP addresses from the CNI plugin's own network space.

The overlay network is bridged into the node network. As a result, one IP address is required per node instead of one IP address needed per pod. Consequently, there are more available IP addresses and you can use the Cloudera Data Warehouse service efficiently, auto-scaling Virtual Warehouses as needed to meet the demands of your workloads.

When to use overlay networks for Cloudera Data Warehouse service environments

By default, when you create a new network during environment registration with the Cloudera Management Console, Cloudera creates 3 subnets with 8,192 IP addresses per subnet, which means there are 24,576 available IP addresses:

Figure 1: Registering an environment with a new network in Cloudera Management Console

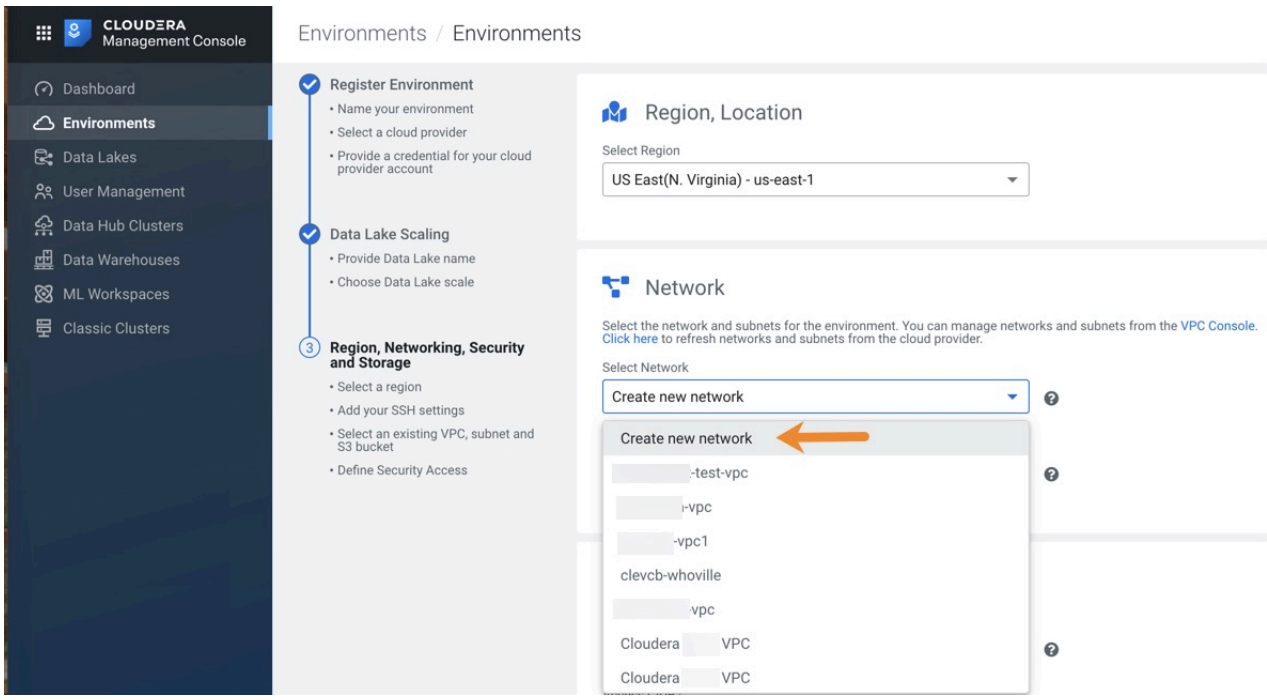

When Cloudera creates your environment, you most likely will have enough IP addresses to use and grow your Virtual Warehouses. However, if you decide to use an existing VPC network on your cloud provider, which might have a limited number of available IP addresses, configuring your environment to use overlay networks for the Cloudera Data Warehouse service avoids IP address exhaustion. Using overlay networks, which can be configured for an environment in the Cloudera Data Warehouse service, uses fewer IP addresses because it uses an "IP per host" model, so your Virtual Warehouse can be used efficiently.

Related Information

Register an AWS environment

Enabling overlay networks in AWS environments

This topic describes how to enable Cloudera environments on AWS to use overlay networks to increase the number of IP addresses that are available to Cloudera Data Warehouse Public Cloud.

About this task

Required role: EnvironmentAdmin or PowerUser

After you have registered your environment with Cloudera, navigate to the Cloudera Data Warehouse service and perform the following steps to configure your AWS environment to use overlay networks.

Procedure

1. In the Cloudera Data Warehouse service, in the Environments column, click the search icon and locate the environment you registered with Cloudera where you want to enable overlay networks.
2. Click Activate to activate the environment.
3. In Activation Settings, click Advanced Settings to see the options:

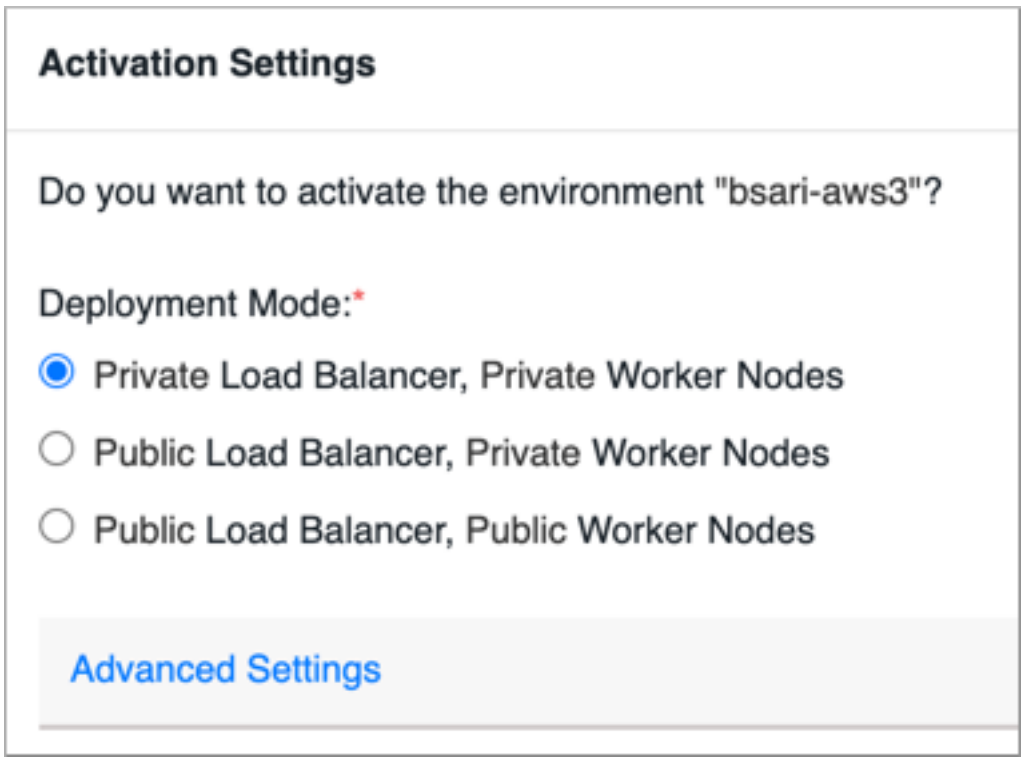

4. Select Use Overlay Network and click Activate:

Use a non-transparent proxy with Cloudera Data Warehouse on AWS environments

You can configure Cloudera environments to use a network proxy in Cloudera Management Console. Then when you activate the environment for Cloudera Data Warehouse, if you can only use non-transparent proxies with clients that must connect to the Virtual Warehouse that uses the environment, you can configure them.

Difference between transparent and non-transparent network proxies

Transparent proxies are unknown to clients and require no additional client configuration. Usually connections by way of transparent proxies are configured in route tables on your AWS VPC. However, clients are aware of non-transparent proxies and each client must be specifically configured to use the non-transparent proxy connection.

There are benefits to using non-transparent proxies because doing so permits you to pass connection or security information along with the connection request sent by clients. Some organizations' security policies require the use of non-transparent proxies and Cloudera Data Warehouse can support that requirement.

Configure non-transparent proxies for Cloudera Data Warehouse on AWS environments

You can configure an AWS environment to use non-transparent proxy connections when activating environments for Cloudera Data Warehouse.

About this task

This task explains how to configure non-transparent proxies when you activate AWS environments for Cloudera Data Warehouse.

Required role: DWAdmin

Before you begin

- Before you can configure non-transparent proxies during environment activation for Cloudera Data Warehouse, you must make sure that a proxy was configured for the environment when it was registered with Cloudera Management Console. For details about configuring a proxy when registering an environment with Cloudera Management Console, see Using a non-transparent proxy.

  Note: When you activate an environment to use with Cloudera Data Warehouse, the option to configure non-transparent proxies is only available if you have added a proxy during environment registration with Cloudera Management Console.
- Before activating the environment that uses a proxy for Cloudera Data Warehouse, set up the following VPC endpoints for your AWS account in the AWS Console:
  - sts.amazonaws.com
  - sts.<*REGION*>.amazonaws.com
  - .s3.<*REGION*>.amazonaws.com
  - .s3.amazonaws.com
  - s3.amazonaws.com
  - dynamodb.<*REGION*>.amazonaws.com
  - api.ecr.<*REGION*>.amazonaws.com
  - dkr.ecr.<*REGION*>.amazonaws.com
  - ec2.<*REGION*>.amazonaws.com
  - cloudformation.<*REGION*>.amazonaws.com
  - autoscaling.<*REGION*>.amazonaws.com
  - elasticfilesystem.<*REGION*>.amazonaws.com
  - elasticloadbalancing.<*REGION*>.amazonaws.com

  For information about creating VPC endpoints, see the Amazon documentation. If you cannot create a VPC endpoint for one of the required outbound destinations that are listed here, you must delete it from the Bypass proxy settings for these domains text box in Step 5 below, and add it to the proxy allowlist.
- Add other AWS specific outbound destinations to your proxy allowlist because creating VPC endpoints for them is not supported by AWS:
  - eks.<*REGION*>.amazonaws.com
  - rds.<*REGION*>.amazonaws.com
  - servicequotas.<*REGION*>.amazonaws.com
  - pricing.<*REGION*>.amazonaws.com

  For more information about the AWS specific outbound destinations used by Cloudera, see Outbound network access destinations for AWS in the Cloudera Management Console documentation.

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment you configured a non-transparent proxy for when you registered it with Cloudera Management Console.
3. When you locate the environment, click Activate to launch the Activation Settings dialog box where you can configure non-transparent proxies for the environment.
4. In the Activation Settings dialog box, click Advanced Settings to expand the dialog box options.
5. In the expanded dialog box, select Use default environment Proxy. When you select this option, the Bypass proxy settings for these domains option appears, and a text box listing all destinations that by-pass this proxy also appears. These are the destinations that you created VPC endpoints for in the above Before you begin section.

   Important: If you cannot create a VPC endpoint for one of the required outbound destinations that are listed in the "Before you begin" section, you must delete it from the Bypass proxy settings for these domains text box by clicking the "x" that is adjacent to it, and then add it to the proxy allowlist.
6. Click Activate to complete environment activation for Cloudera Data Warehouse.

Setting up private networking in AWS environments

The topics in this section describe how to enable private subnets in Cloudera Data Warehouse on AWS.

Required role: DWAdmin

In AWS, a *public subnet* is connected to an internet gateway which can send and receive traffic directly to and from the internet. *Private subnets* send outbound traffic from nodes to the internet by using a network address translation (NAT) gateway, and then forwards the traffic to an internet gateway. Private subnets receive no direct inbound connections from the internet. This provides private network connectivity for workload endpoints in Cloudera Data Warehouse service.

Related Information

Amazon documentation on public and private subnets

Supported deployment modes for private networking in AWS

This topic describes the different types of private networking that you can configure Cloudera Data Warehouse to use on AWS.

The Cloudera Data Warehouse service can be deployed in three different modes on AWS cloud resources. These supported deployment modes determine whether your cluster nodes can receive direct connections from the internet.

The supported deployment modes in AWS are:

1. Public Load Balancer, Public Worker Nodes

   Requires that you specify 3 public subnets in the virtual private cloud (VPC) that is registered with the Cloudera Management Console. In this mode, all AWS network components have a publicly visible IP address assigned to them. However, all traffic flows through the Kubernetes ingress controller so services and ports are not directly accessible.
2. Public Load Balancer, Private Worker Nodes

   Requires that you specify 3 public subnets and 3 private subnets in the VPC that is registered with the Cloudera Management Console. In this mode, the Amazon EKS nodes are not assigned public IP addresses. All traffic is routed from the load balancer (Amazon ELB), which is located in a public subnet, to the ingress controller and the private subnet containing the private worker nodes.
3. Private Load Balancer, Private Worker Nodes

   Requires that you specify 3 private subnets in the VPC that is registered with the Cloudera Management Console. This mode also requires that you set up network routing by way of a VPN between your on-premises network and

the VPC. Both the load balancer and worker nodes reside in private subnets so no Data Warehouse services have publicly visible IP addresses assigned to them.

Note: Selecting private load balancers is required if you want to use a private EKS API server.

Prerequisites for private networking in AWS environments

This topic lists the prerequisite requirements for configuring Cloudera Data Warehouse to use private networks on AWS.

To use the AWS private networking feature in the Cloudera Data Warehouse service, make sure you set up gateway virtual private cloud (VPC) endpoints and that you specify the correct number of public and private subnets for the Cloudera environment you plan to use for the service.

Set up gateway VPC endpoints to improve performance

For the Public Load Balancer/Private Worker Nodes and the Private Load Balancer/Private Worker Nodes deployment modes where worker nodes are running in private subnets, all outbound internet traffic passes through network address translation (NAT) gateways. For example, traffic created by activities such as downloading Docker images or accessing Cloudera services like billing or metering, S3 storage, and Amazon DynamoDB. To improve performance by reducing the number of network hops for accessing S3 and DynamoDB, set gateway VPC endpoints. A gateway endpoint is a gateway that you specify as a target for a route in your route table for traffic that is destined to either Amazon S3 or DynamoDB, the two supported AWS services.

If your VPC does not contain VPC endpoints that target S3 or DynamoDB, use the Amazon documentation, which is linked to at the bottom of the page, to set them up.

Specify the correct number of public and private subnets for your Cloudera environment

Depending on which deployment mode you choose, select the correct number of private and public subnets in the VPC environment that you register with the Cloudera Management Console. Check the documentation on Supported deployment modes to make sure what combination of public and private VPC subnets are needed before you register your environment.

Note: If you can only use private subnets in your AWS VPC environment, you can specify three private subnets when you register the environment with Cloudera Management Console. However, if you set up private networking with a Cloudera environment that includes three private subnets and no public subnets, network administrators must make sure there is outbound internet access from private subnets by way of the transit gateway or by another means.

General instructions for registering an environment are included in the Cloudera Management Console documentation. A link to this documentation is provided below. When you are on Step 6 where you specify information on the Region, Networking, Security and Storage page of Cloudera Management Console, specify the appropriate number of public or private subnets in your VPC for the Select Subnets prompt on the page:

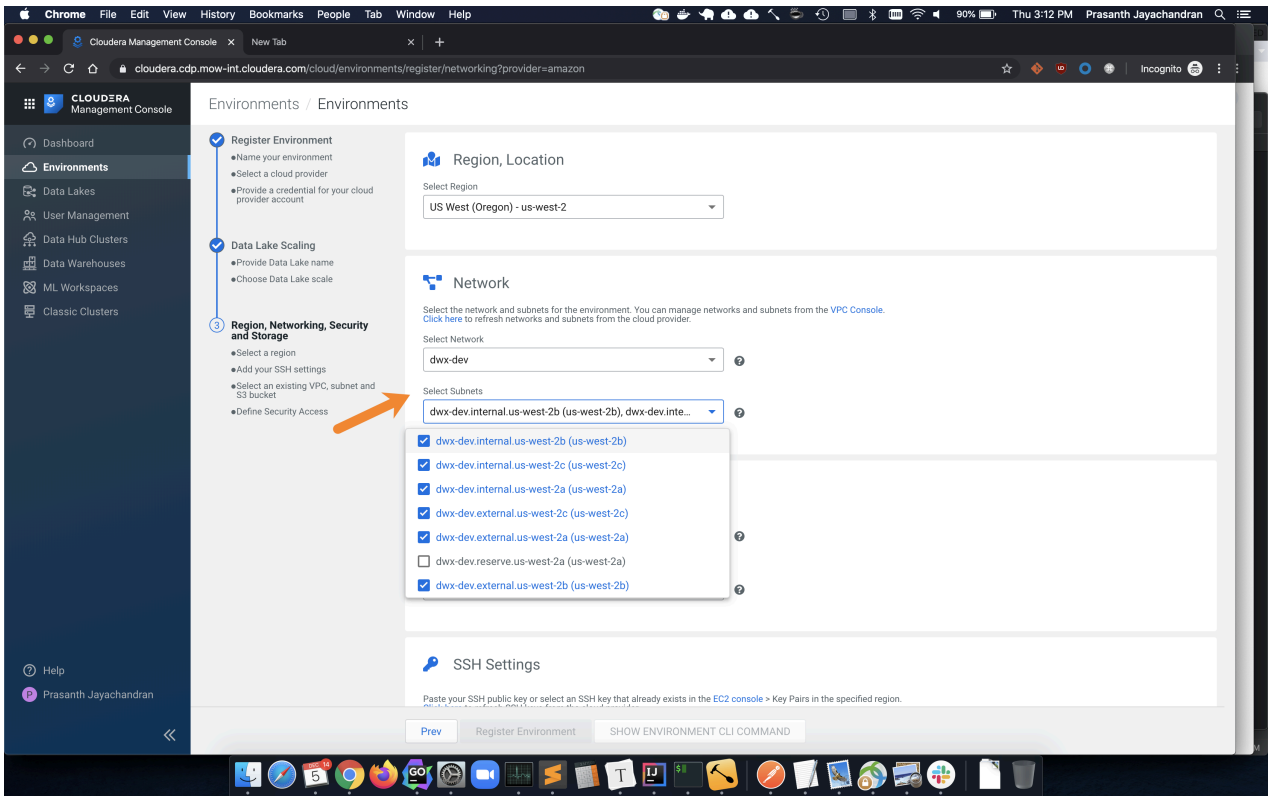

Then proceed through the instructions to complete registering your environment with Cloudera.

Related Information

Endpoints for Amazon S3

Endpoints for Amazon DynamoDB

Register an AWS environment with Management Console

Activating an AWS environment with private subnet support

This topic describes the procedure for activating an AWS environment to use private subnets in Cloudera Data Warehouse Public Cloud.

About this task

After you have registered your environment with Cloudera, navigate to Cloudera Data Warehouse service and perform the following steps to activate the environment.

Required role: DWAdmin

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment for which you want to configure private networking.
3. Click Activate to activate the environment.

4. Specify the Deployment Mode:

   Important: Cloudera recommends that you use the Private Load Balancer, Private Worker Nodes deployment mode if possible because it is the most secure.

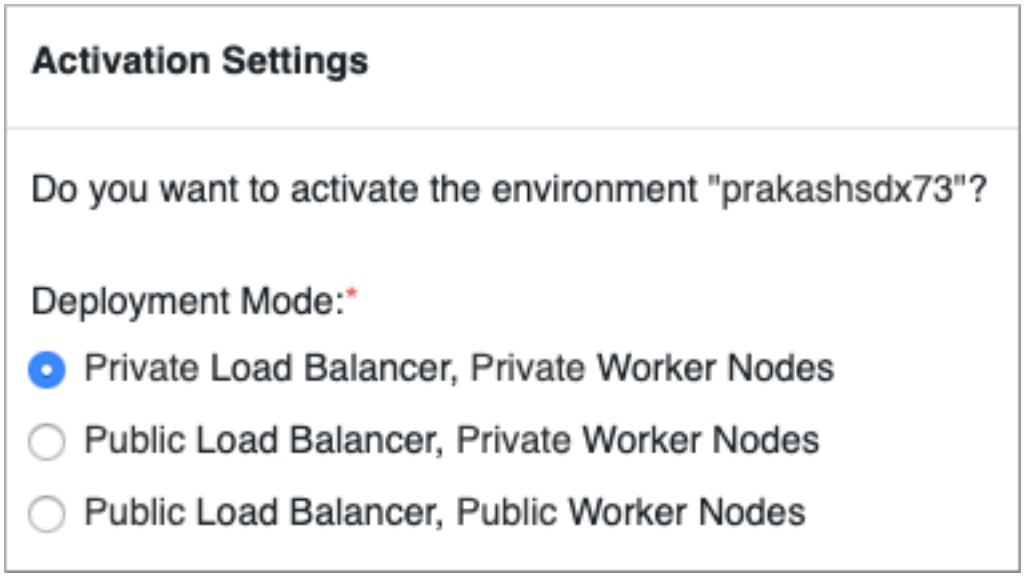

5. To view the public and private subnets specified for your Cloudera environment, click Advanced Settings.
6. Click Activate.

Related Information

Supported deployment modes

Architecture for Private Load Balancer, Private Worker Nodes deployment on AWS

This topic contains a network architecture diagram of Cloudera Data Warehouse configured to use private networking in an AWS environment.

The following diagram shows the architecture when you specify the Private Load Balancer, Private Worker Nodes deployment mode during environment activation in the Cloudera Data Warehouse service. In this deployment mode only the Network Address Translation (NAT) gateways reside in public subnets. Network administrators must make sure there is outbound internet access from private subnets by way of a transit gateway or by another means.

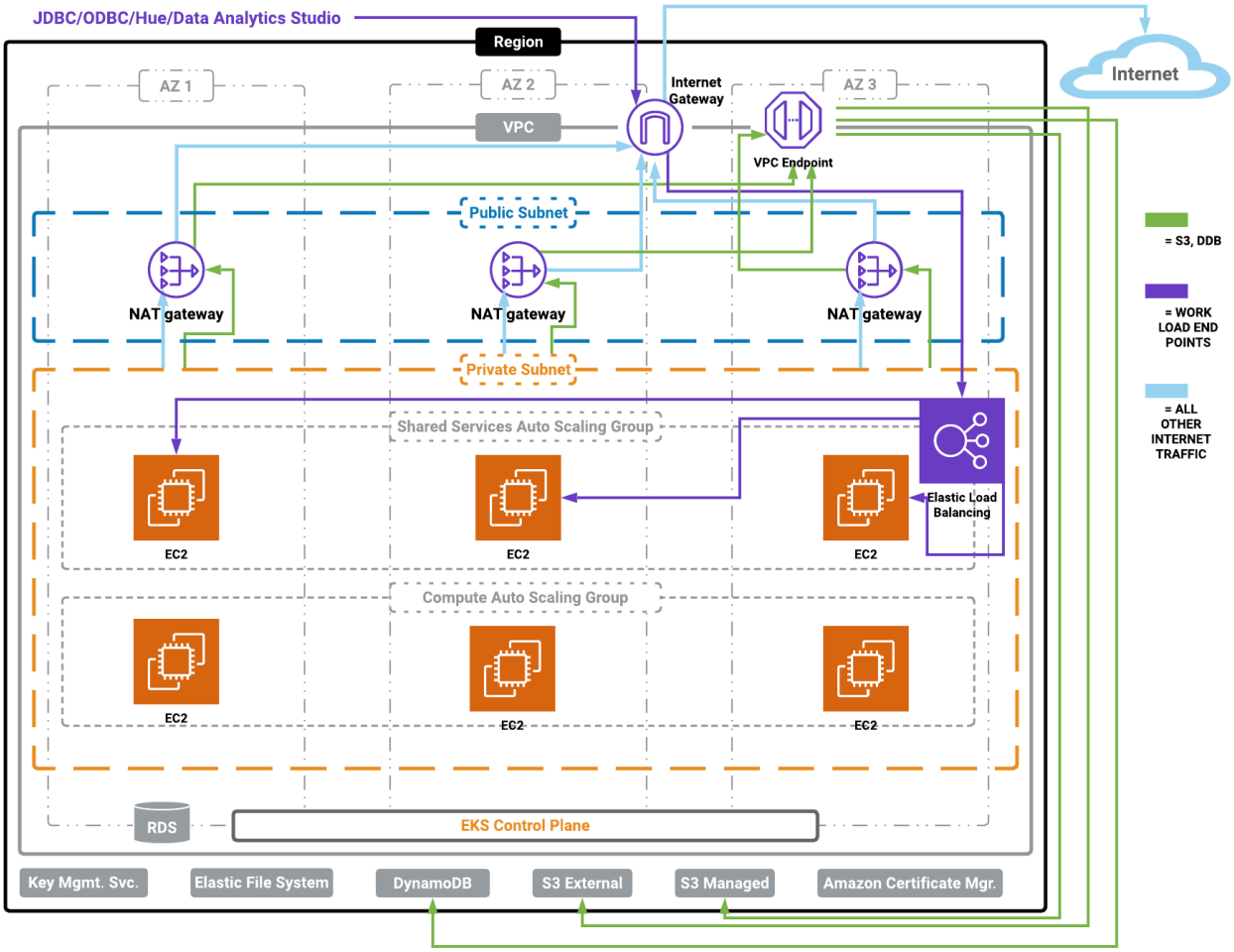

Custom tags in AWS environments

You can apply Virtual Warehouse-level tags and tenant-level custom tags to monitor, search, or filter on the key tag or key value in your AWS account.

Use the Virtual Warehouse-level tags to search and monitor the Virtual Warehouses within your environment. You can apply Virtual Warehouse-level tags even after creating the Cloudera environment. The tenant-level tags can only be applied while creating a new Cloudera environment. The Cloudera environment custom tags are propagated to the compute instances within Cloudera Data Warehouse.

You can identify and monitor the following resources within the Cloudera Data Warehouse service for AWS environments:

- Compute instances in Hive and Impala Virtual Warehouses with tenant-level and Virtual Warehouse tags
- Compute instances in Druid cluster with tenant-level tags
- AWS S3 buckets
- Amazon Elastic Block Store (EBS) volumes for EC2 instances used for compute pods with Virtual Warehouse tags. You cannot tag EBS volumes for EC2 instances used for shared service pods.

Upgrades

Upgrading Amazon Elastic Kubernetes Service (EKS) can prevent incompatibilities issues between Cloudera Data Warehouse and AWS resources.

Upgrading PostgreSQL 9.6 before EOL

You need to plan for the end-of-life (EOL) of PostgresSQL 9.6, announced by Amazon. As an AWS environment user, you need to check the version of PostgreSQL you use for Cloudera Data Warehouse environments. If your database version is still PostgreSQL 9.6, you need to upgrade to PostgreSQL 11.12.

About this task

AWS has published the following announcement regarding the imminent PostgreSQL upgrade.

> **Upgrade your Amazon RDS for PostgreSQL 9.6 databases before Jan 18, 2022**
>
> The RDS for PostgreSQL 9.6 end-of-life date is approaching. You have 1 database using RDS for PostgreSQL 9.6 that must be upgraded to 12 or higher as soon as possible.
>
> We plan to automatically upgrade RDS for PostgreSQL 9.6 databases to 12 starting January 18, 2022.
>
> For more information, see the RDS for PostgreSQL deprecation timeline in the Amazon RDS forum.

You can initiate an upgrade of your database instance — either immediately or during your next maintenance window — to the Cloudera-recommended version of PostgreSQL 11.12 using the AWS Management Console or the AWS Command Line Interface. Follow the procedure below to perform the upgrade.

The upgrade process shuts down the database instance, performs the upgrade, and restarts the database instance. The database instance may be restarted multiple times during the upgrade process. While major version upgrades typically complete within the standard maintenance window, the duration of the upgrade depends on the number of objects within the database. To estimate the time required, take a snapshot of your database and test the upgrade.

Before you begin

Procedure

1. Go to the AWS Management Console Amazon RDS Databases .

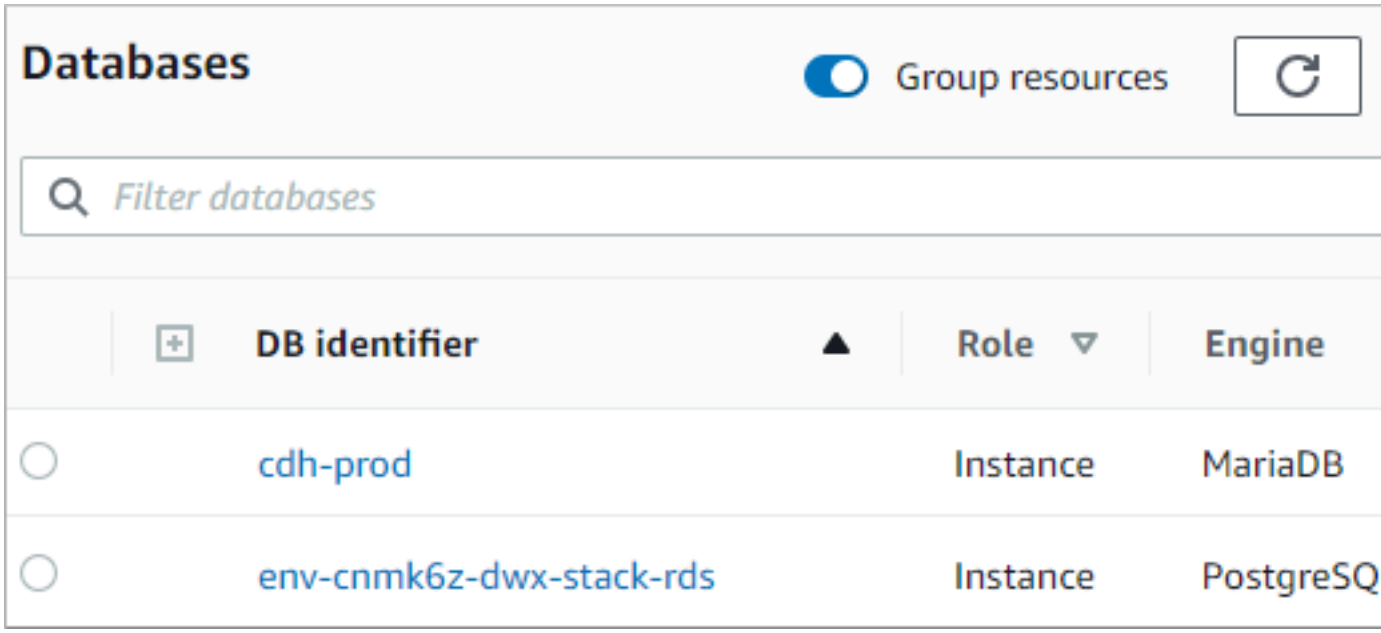

2. Check the version of PostgreSQL used in your environment, and if it is 9.6.6, go to the next step to start the upgrade process.
3. In Cloudera, go to your environment, and in a Database Catalog for that environment, create a Virtual Warehouse or use an existing one.
4. Run some basic queries in Hive or Impala to see if your PostgreSQL 9.6.6 is alive and well.

```
show tables;
use default;
show tables;
create table tbl1(col1 string, col2 string);
show tables;
describe tbl1;
create table tbl2 as (select * from tbl1);
describe tbl2;
insert into table tbl1 values ("Hello", "World");
select * from tbl1;
```

5. Go to the AWS Management Console Amazon RDS Databases . Select DB Engine version 10.16 to upgrade Amazon RDS to 10.16.

   Amazon prevents a direct upgrade to 11.x unless you are on PostgreSQL 9.6.20 or higher.
6. In Cloudera, rerun the basic queries in your Virtual Warehouse, and if all goes well, proceed to the next step.
7. Look for errors, such as those shown below, in the metastore log.
   Errors in the metastore might look something like this:

```
<14>1 2021-07-31T00:52:38.223Z metastore-0.metastore-service.warehouse-1
627669911-vl6x.svc.cluster.local metastore 1 0b245ec4-8419-4968-94bc-ee1
22960b1aa [mdc@18060
class="txn.TxnHandler" level="INFO" thread="pool-9-thread-200"] Non-ret
ryable error in enqueueLockWithRetry(LockRequest(component:[LockComponen
t(type:SHARED_READ, level:
DB, dbname:default, operationType:NO_TXN, isDynamicPartitionWrite:false)]
, txnid:79, user:hive, hostname:hiveserver2-0.hiveserver2-service.comput
e-1627670377-mbqx.svc.
```

```
cluster.local, agentInfo:hive_20210731005238_4aa14bf0-4e46-448d-b6c7-cdc3
ca4ec863, zeroWaitReadEnabled:false)) : Batch entry 0 INSERT INTO "HIVE_
LOCKS" ( "HL_LOCK_EXT_ID",
"HL_LOCK_INT_ID", "HL_TXNID", "HL_DB", "HL_TABLE", "HL_PARTITION", "HL_LO
CK_STATE", "HL_LOCK_TYPE", "HL_LAST_HEARTBEAT", "HL_USER", "HL_HOST", "H
L_AGENT_INFO") VALUES
(4561258935320160815, 1, 79, 'default', NULL, NULL, 'w', 'r', 0, 'hive', '
hiveserver2-0.hiveserver2-service.compute-1627670377-mbqx.svc.cluster.lo
cal',
'hive_20210731005238_4aa14bf0-4e46-448d-b6c7-cdc3ca4ec863') was aborted: E
RROR: index "hl_txnid_index" has wrong hash version^M  Hint: Please REIN
DEX it.  Call
getNextException to see other errors in the batch. (SQLState=XX002, ErrorC
ode=0)
```

8. Connect to the HiveServer pod or metastore pod, and using psql, connect to the RDS instance.
   For example,

   ```
   psql -h env-kg.us-west-2.rds.amazonaws.com --u hive --d postgres
   ```

   Login using the hostname from the AWS console. The postgres database password is stored in JCEKS file which is mounted using a secret volume inside the HiveServer pod or metastore pod. Note the namespace of pod and obtain the password.

   ```
   [hive@metastore-0 lib]$ psql -h env-kg.us-west-2.rds.amazonaws.com --u hive --d postgres
   Password for user hive:
   psql (9.2.24, server 9.6.6)
   WARNING: psql version 9.2, server version 9.6.
            Some psql features might not work.
   SSL connection (cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256)
   Type "help" for help.

   postgres=>
   ```

Related Information

Upgrading the PostgreSQL DB engine for Amazon RDS

Validate the upgrade to PostgreSQL 10.16

You need to connect to the postgres 10.16 database, and validate the upgrade.

Procedure

1. Connect to the PostgreSQL 10.16 database using the namespace of the pod and the password you obtained in the last procedure.

2. On the Postgres command line, look at all the databases.
   For example, type \l.

```
postgres=> \l
                                                  List of databases
                Name                  |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
--------------------------------------+----------+----------+-------------+-------------+-----------------------
 metastore                            | hive     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres                             | hive     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 rdsadmin                             | rdsadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 | rdsadmin=CTc/rdsadmin
 template0                            | rdsadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/rdsadmin          +
                                      |          |          |             |             | rdsadmin=CTc/rdsadmin
 template1                            | hive     | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/hive              +
                                      |          |          |             |             | hive=CTc/hive
 viz-1629222151-5xqq_vizdb            | hive     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 warehouse-1629193710-rmjh-das        | hive     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 warehouse-1629193710-rmjh-metastore  | hive     | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
```

3. Fix any problematic indexes. Go to a database in your metastore named something like
   For example, go to a database in your metastore named warehouse-1629221321-pf2h-metastore.

```
\c warehouse-1629221321-pf2h-metastore
```

```
postgres=> \c warehouse-1629221321-pf2h-metastore
psql (9.2.24, server 9.6.6)
WARNING: psql version 9.2, server version 9.6.
         Some psql features might not work.
SSL connection (cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256)
You are now connected to database "warehouse-1629221321-pf2h-metastore" as user "hive".
warehouse-1629221321-pf2h-metastore=>
```

4. Run the REINDEX command.

```
REINDEX INDEX hl_txnid_index;
```

5. In Cloudera, rerun the basic queries in your Virtual Warehouse, and if all goes well, proceed to the next step.
6. Upgrade the Amazon RDS version of PostgreSQL to 11.12, and rerun the basic queries.

Upgrade to PostgreSQL 11.12

Procedure

1. Go to the AWS Management Console Amazon RDS Databases , and upgrade the Amazon RDS version to PostgreSQL 11.12.
2. Rerun the basic queries in your Virtual Warehouse to validate the upgrade.

Upgrading Amazon Kubernetes Service (EKS)

Amazon Elastic Kubernetes Service (EKS) cluster requires regular updates to the Kubernetes versions. Using the latest EKS version supported by Cloudera avoids compatibility issues between Cloudera Data Warehouse and AWS resources.

Cloudera Data Warehouse automatically provisions the latest supported EKS version when you activate an environment in Cloudera Data Warehouse. EKS 1.30 is provisioned automatically when you activate your environment from Cloudera Data Warehouse using release 1.9.1-b166 (released December 5, 2024) and later.

To upgrade to the latest EKS, you must deactivate and reactivate your Cloudera Data Warehouse environment. To restore the state of your Cloudera Data Warehouse cluster after reactivation, Cloudera recommends using the backup-restore functionality. For more information, see Backing up and restoring Cloudera Data Warehouse.

Upgrading using your own AMI or reduced permissions

You learn the prerequisite for using your own AMI, the risks of using your own AMI, and the steps involved in upgrading.

About this task

You can upgrade the Amazon Elastic Kubernetes Service (EKS) using your own Amazon Machine Image (AMI) if you meet either one of the following conditions:

- You are in reduced permissions mode.
- You have obtained the CDW_CUSTOM_AMI entitlement.

The option to enter your own AMI name does not appear when you perform the upgrade steps in this document unless you meet one of these conditions.

Using your own AMI to upgrade worker executors is error-prone due to EKS version matching requirements, and therefore not recommended. Cloudera recommends that you upgrade to the recommended EKS instead of using your own AMI. If you decide to use your own AMI, you must choose the AMI version that matches the EKS version used by Cloudera Data Warehouse. Using your own AMI, you can upgrade only to the next minor EKS version, so more than one upgrade might be required to match the EKS version used by Cloudera Data Warehouse.

If you do not use a custom AMI in step 4 below, but use reduced permissions, you must upgrade three times from EKS v1.17 to get to v.1.20.

Before you begin

- Check that you are using the reduced permissions mode, or that you have obtained the CDW_CUSTOM_AMI entitlement.
- Perform this workaround described in the known issue DWX-8573: EKS upgrade from DWX UI to K8s v1.20 fails in reduced permissions mode. In AWS, add the following permissions to your Reduced permissions mode JSON IAM permissions policy right after "iam:PutRolePolicy":

```
{
 ...
                "cloudformation:GetTemplate",
                "cloudformation:GetTemplateSummary",
                "iam:GetRole",
                "eks:ListUpdates",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DescribeLaunchTemplates",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DeleteAutoScalingGroup",
                "ec2:RunInstances",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "autoscaling:DescribeScheduledActions",
                "autoscaling:SetDesiredCapacity",
                "iam:PassRole",
                "rds:DescribeDBInstances",
                "ec2:DescribeInstances"
...
}
```

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment on AWS for upgrading the Kubernetes version.
3. Click Upgrade.
   In reduced permissions mode, the option to upgrade using a custom AMI appears:

4. Select Use custom AMI.
   A text box appears for entering the your AMI version.

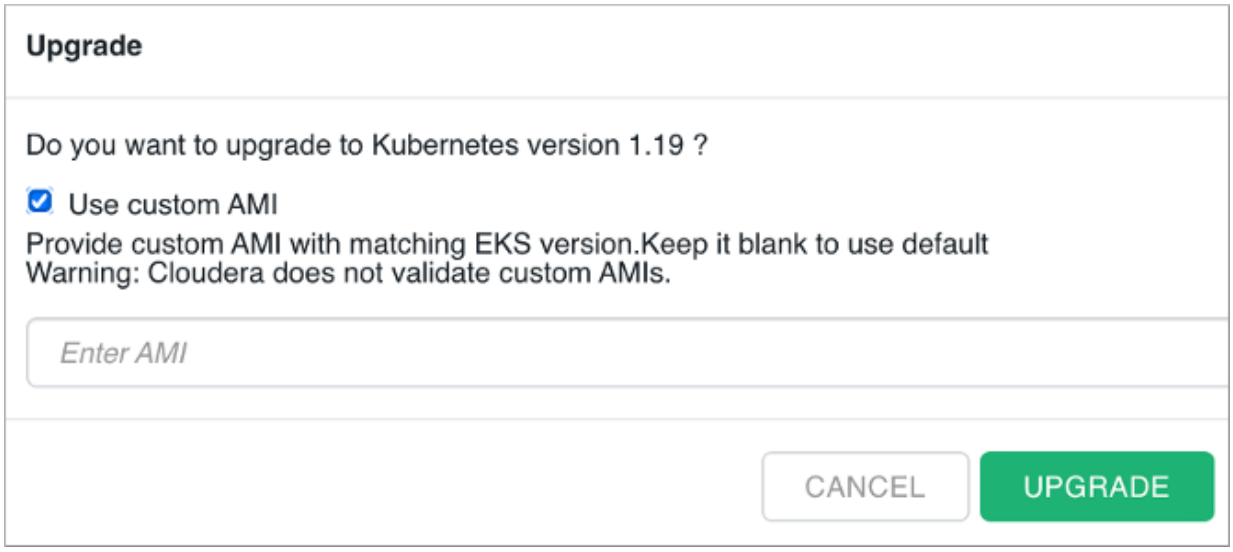

   You can upgrade only to the next minor version for each environment upgrade, for example from 1.18 to 1.19. For example, to upgrade to EKS 1.20, you must upgrade three times from EKS 1.17.
5. Enter an AMI version that is consistent with the Cloudera Data Warehouse upgrade to the EKS version and region of the activated cluster.
6. Click Upgrade to start the upgrade process.

Dynamically updating the Amazon Machine Image

You need the capability to update the Amazon Machine Image (AMI) to prevent potential problems running workloads on an old AMI. You can update the AMI of the Cloudformation stack while keeping the current Elastic Kubernetes Service (EKS) version.

Before you begin

You must meet the following prerequisites to get dynamic updates:

- You are running release 1.7.1 (released August, 2023) or later, which supports this feature.

- You must add the following IAM roles to your cross account role:

```
{
    "Sid": "AllowSsmParams",
    "Effect": "Allow",
    "Action": [
        "ssm:DescribeParameters",
        "ssm:GetParameter",
        "ssm:GetParameters",
        "ssm:GetParameterHistory",
        "ssm:GetParametersByPath"
    ],
    "Resource": [
        "arn:aws:ssm:*:*:parameter/aws/service/eks/optimized-ami/*"
    ]
}
```

  This action creates and upgrades DWX stacks with read permissions on AWS Systems Manager (SSM). The SSM parameter in the Cloudformation dwx-stack template obtains the latest AMI to use for your EKS version.
- If you use reduced permissions mode, you must update the following fields of the stack template while performing a Kubernetes version upgrade:
  - EksBaseVersion - The value of this parameter must be the next upgrade version of kubernetes.
  - EksImageSSMParam - The value of this parameter must be formatted as follows: /aws/service/eks/optimized-ami/<K8s-version>/amazon-linux-2/recommended/image_id

    Update only the K8s-version of EksImageSSMParam. For example:
    - Old value

      /aws/service/eks/optimized-ami/1.21/amazon-linux-2/recommended/image_id
    - New value

      /aws/service/eks/optimized-ami/1.22/amazon-linux-2/recommended/image_id

Procedure

1. Log into the AWS Management Console.
2. From the list of stacks, select the running CloudFormation stack.
3. In the stack details pane, click Update.
   A new AMI, if available, will be applied in the stack update.

Managed storage access

Understanding how Cloudera Data Warehouse stores data for multiple tenants and a high-level overview of the configuration tasks prepares you as DWAdmin to set up a RAZ-controlled warehouse.

The multitenant storage technique in Cloudera Data Warehouse offers increased security over the storage method used in earlier releases that based all storage access in Cloudera Data Warehouse on a single EC2 instance role. The old method listing all S3 buckets that need to be accessible by Hive or Impala made the impact of a leaked access token for a role unacceptably wide, even though the access token is normally very time-constrained. The new technique requires a separate Database Catalog plus at least one Virtual Warehouse per tenant.

You configure a dedicated IAM role to give only the tenant-specific default database catalog instance and its associated virtual warehouse data access to both of the following buckets.

- The tenant-specific bucket(s)
- Potentially shared bucket(s)
- The SDX Data Lake bucket

If the access token for a role is leaked, the data of others is not compromised; only one tenant's own, and shared, data is vulnerable. A custom identity broker IDBroker approach to accessing the Virtual Warehouse is shown in the following graphic:

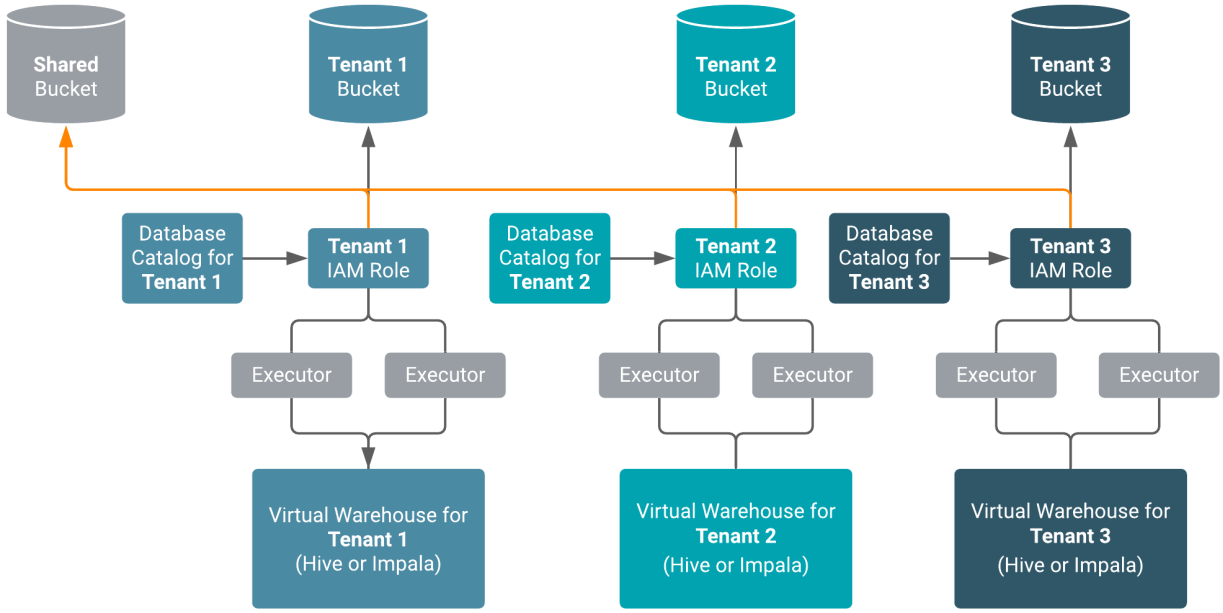

Using a Ranger Remote Authorization (RAZ)-enabled environment, you can control access to data based on user roles and classifications. As a Cloudera Admin, you can apply Ranger fine-grained access control policies to S3 buckets, directories, and files.

The following diagram shows the high-level steps for configuring RAZ-enabled storage:

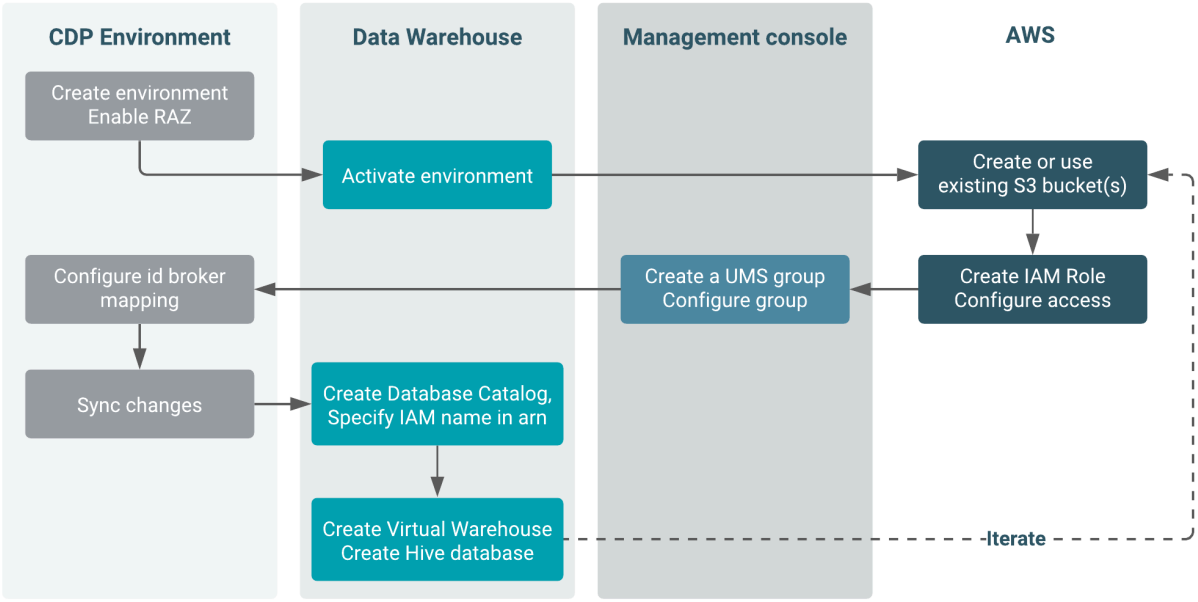

After you obtain the entitlements required for this feature CDW_ALLOW_MULTI_DEFAULT_DBC and CDW_STORAGE_ROLES, you can configure storage as shown in this diagram:

- In Cloudera, register an environment that enables RAZ and uses the SDX Data Lake.
- Activate the environment in Cloudera Data Warehouse.

- Manually create S3 bucket(s), or use existing ones, for access by the IAM tenant role.
- In AWS, create an IAM role for the tenant.
- Give the IAM role certain permissions.
- Configure a trust relationship with the IDBROKER_ROLE.
- Give the Instance Profile (idbroker assume role) the permission to assume the IAM role of the tenant.
- In Cloudera Management Console, create an UMS group.
- Add the UMS machine user (created when you activated the Cloudera Data Warehouse environment) to the UMS group.
- Add the id broker mapping to the environment.
- Sync user group changes for the RAZ-enabled environment with FreeIPA.
- Create a separate Database Catalog with a unique IAM role.
- Create a tenant-specific Virtual Warehouse based on the Database Catalog, SDX Data Lake, and RAZ-enabled environment.
- Create a tenant-specific Hive or Impala database that points to tenant-specific buckets.

  As the metadata is shared across all tenants, Ranger grants access to tenant data via a group at the database level.
- For each tenant, repeat the actions above, starting from the step after activating the environment for Cloudera Data Warehouse.

The IAM role accesses a tenant specific bucket s3-tenant-1 and SDX Data Lake bucket. The SDX Data Lake bucket is needed for shared data and other data used by Database Catalog and to integrate WXM. WXM writes Impala query data to the shared bucket. You need the UMS group to add the IDBroker mapping, as a single UMS machine user cannot have multiple ID broker mappings to different IAM roles.

The following topics describe step-by-step how to set up your environment, IAM roles, Database Catalog, and Virtual Warehouse for storing RAZ-enabled data.

Setting up managed storage access

You learn the consequences of setting up RAZ control, which cannot be removed, and requirements you must meet to enable managed storage access.

About this task

The RAZ-controlled warehouse supports only multiple Shared Data Experience (SDX) Database Catalogs instead of Cloudera Data Warehouse-specific, isolated database catalogs. In a RAZ-controlled environment, the Elastic Kubernetes Service (EKS) ec2 executor will not have read/write permissions to the S3 bucket. Consequently, after activating the Cloudera Data Warehouse environment, you cannot remove RAZ control. RAZ-control of Cloudera Data Warehouse continues during upgrades of the Database Catalog and Virtual Warehouse.

Before you begin

- Request activation of the following entitlements:
  - CDW_ALLOW_MULTI_DEFAULT_DBC
  - CDW_STORAGE_ROLES
- You meet the requirements described in the AWS requirements documentation.

Required role: PowerUser

Creating the Cloudera environment and IAM roles

You follows steps to register and activate a new environment to enable RAZ in Cloudera Data Warehouse.

About this task

You need to register and activate a new environment to enable RAZ in Cloudera Data Warehouse.

Procedure

1. Register an environment with RAZ using the Cloudera web interface.
   In the web interface, in Fine-grained access control on S3, select Enable Ranger Authorization for AWS S3.

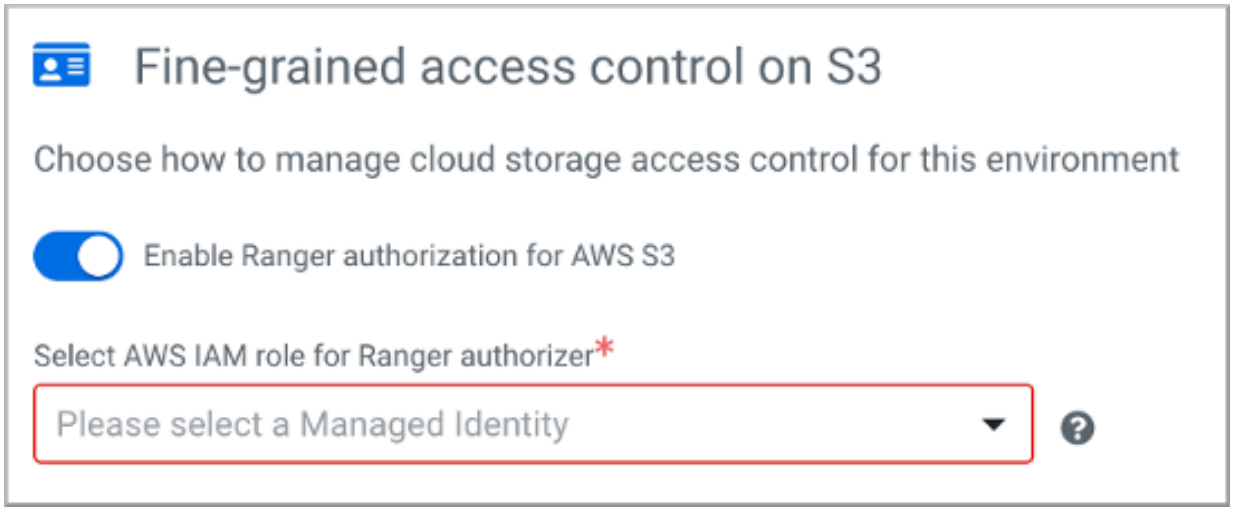

2. In Cloudera Data Warehouse Overview, locate and select the RAZ-enabled environment, and click Start to activate the environment for Cloudera Data Warehouse.
   This action disables the standard default Database Catalog that is automatically created after activation. The UMS machine user is created and attached to the environment when you activate the Cloudera Data Warehouse environment. Later, you see how to add this same UMS machine user to a different UMS group for each tenant.
3. In Activate Environment, select Enable Storage Roles.

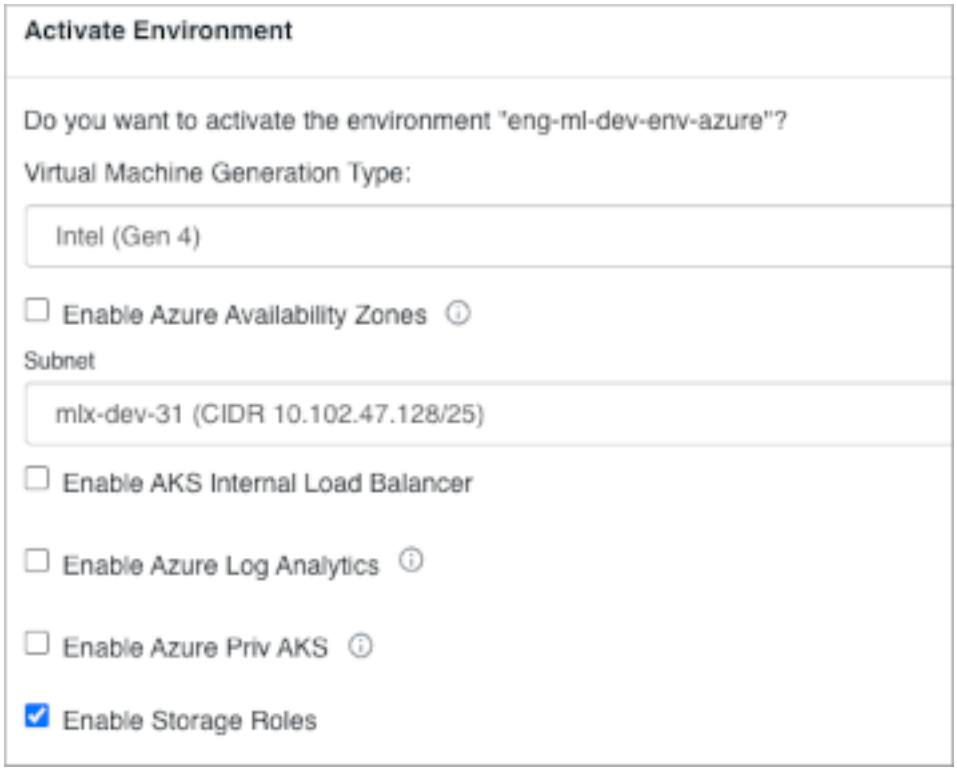

   Unchecking Enable Storage Roles disables the metadata proxy associated with managed storage access.

Repeat the following steps for each tenant:

4. In AWS, manually create one or more S3 buckets for the tenant, or use existing buckets. Use bucket-level encryption and a key that is accessible by the IAM role of the tenant.

5. In AWS, create an IAM role that has significant privileges to read/write to a tenant-specific buckets as well as across all tenants' shared SDX Data Lake bucket.
   For example, create a role named role-tenant-1 and a bucket called s3-tenant-1.
6. In the IAM role, configure a trust relationship with the IDBROKER_ROLE.
   For example:

```
{
                        "Version": "2012-10-17",
                        "Statement": [
                        {
                        "Effect": "Allow",
                        "Principal": {
                        "Service": [
                        "s3.amazonaws.com",
                        "ec2.amazonaws.com"
                        ],
                        "AWS": "arn:aws:iam::<AWS_ACCOUNT_ID>:IDBROKER_ROL
E"
                        },
                        "Action": "sts:AssumeRole"
                        }
                        ]
                        }
```

   The IDBROKER_ROLE needs the trust relationship to assume the role role-tenant-1.
7. Make a note of the IAM role ARN.
8. Attach the following AWS policies to the IAM role: AmazonEKSWorkerNodePolicy, AmazonEKS_CNI_Policy, AmazonEC2ContainerRegistryReadOnly.

   The AmazonEKSWorkerNodePolicy is shown in the next topic.
9. Give the AWS Instance Profile (idbroker assume role), the permission to assume role of role-tenant-1.

```
{
                        "Version": "2012-10-17",
                        "Statement": [
                        {
                        "Sid": "VisualEditor0",
                        "Effect": "Allow",
                        "Action": "sts:AssumeRole",
                        "Resource": [
                        "arn:aws:iam::<AWS_ACCOUNT_ID>:role/idbroker-data-
access-role",
                        "arn:aws:iam::<AWS_ACCOUNT_ID>:role/mock-idbroker-
admin-role",
                        "arn:aws:iam::<AWS_ACCOUNT_ID>:role/role-tenant-1"
                        ]
                        }
                        ]
                        }
```

Creating a UMS group and machine users

This procedure ensures that the Cloudera machine user gets permission to access the tenant bucket.

Procedure

Repeat the following steps for each tenant:

1. In Cloudera Management Console, User Management Groups , click CREATE GROUP and create a User Management Service (UMS) group, for example group-tenant-1.

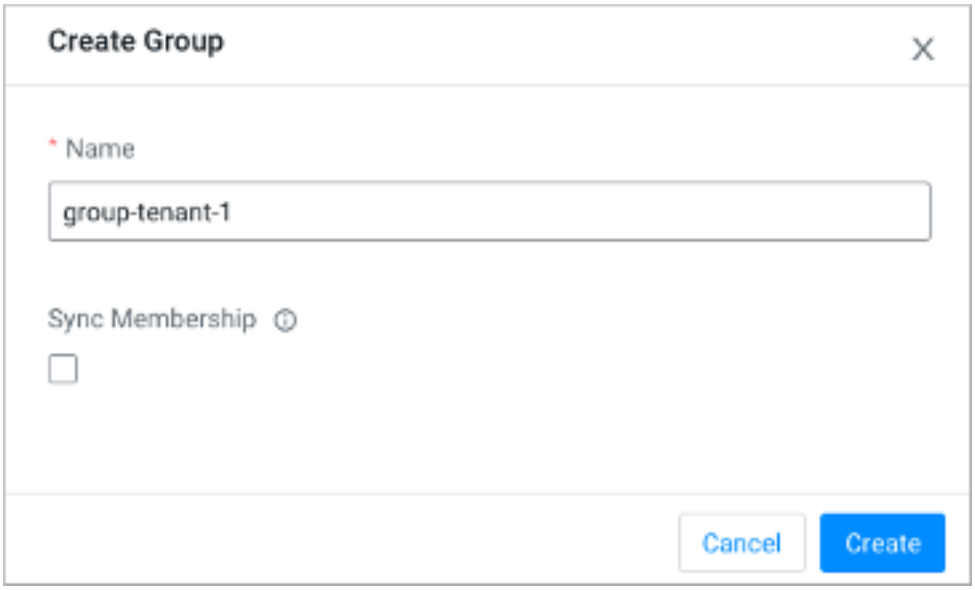

2. In Groups Members , search for and select your srv_machine_<env id>_storage_role to add this UMS machine user to group-tenant-1.

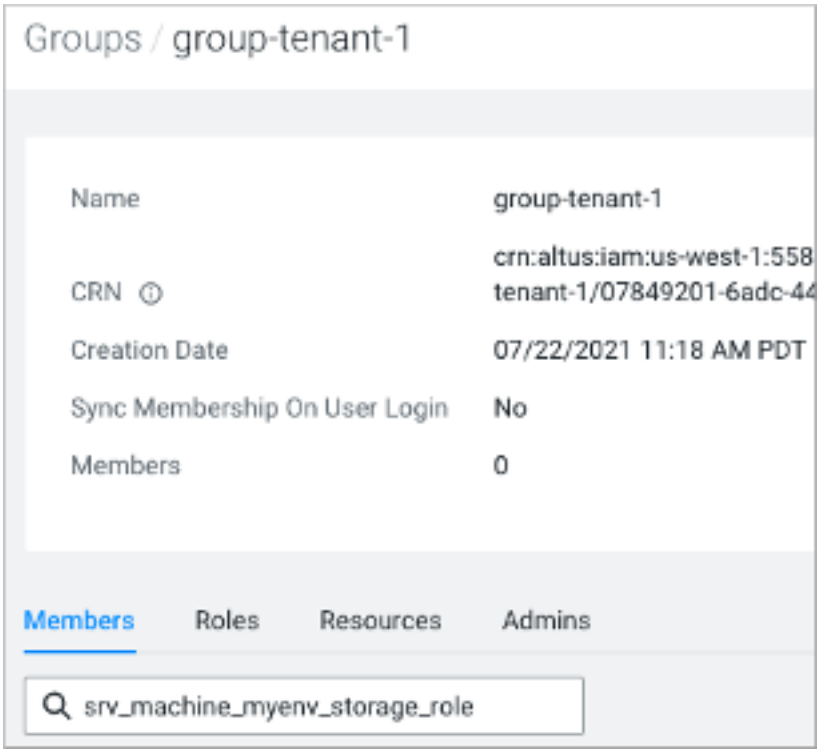

3. In Management Console Environments , select an environment , and click Actions Manage Access IDBroker Mappings Edit .
4. Click + to add a mapping, select the Group-tenant-1 and Role-tenant-1, and specify the role ARN (copied from the IAM role page on AWS).
5. Synchronize your group changes with FreeIPA by performing a user sync per environment: In the RAZ-enabled environment, click Actions Synchronize Users to FreeIPA.
   The UMS machine user gets the permission to access the tenant-specific container.

Creating a new Database Catalog

You repeat a step-by-step procedure to set up a Database Catalog for each tenant, taking care to correctly enter your IAM Role ARN.

Procedure

Repeat the following steps for each tenant.

1. Click Data Warehouse Database Catalog New Database Catalog .
2. In Tenant Storage Role, enter the Iam Role ARN you copied earlier.

   Take care to enter the ARN correctly because Cloudera Data Warehouse does not validate your ARN.
3. In Tenant Storage Location, enter the tenant bucket name, for example s3-tenant-1, and click CREATE.
4. Select the RAZ-enabled environment.

   In Data Lake, SDX is the required value. The backend Data Lake and Database Catalog database must be the same.

Creating a tenant-specific Virtual Warehouse

You follow a step-by-step procedure to create a Virtual Warehouse based on the Database Catalog and SDX Data Lake you created in the RAZ-enabled environment.

About this task

The Database Catalog and Data Lake point to the same backend database. In the tenant-specific Virtual Warehouse, you create a tenant-specific Hive or Impala database that points to tenant-specific buckets. As the metadata is shared across all tenants, Ranger grants access to tenant data at the table level. One or more tenant-specific databases alongside databases for shared data can run in the same HMS instance.

Procedure

Repeat the following steps for each tenant.

1. In the Cloudera Data Warehouse service, click Virtual Warehouses New Virtual Warehouse .
2. Specify a name, select either the Hive or Impala type, and select the Database Catalog you created for the tenant.
3. In Overview, find your Virtual Warehouse, click Hue.
4. Create a tenant-specific Hive or Impala database where the location for external and managed tables are pointing to the tenant-specific buckets.

   ```
   CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name
   [COMMENT database_comment]
   [LOCATION external_table_path]
   [MANAGEDLOCATION managed_table_directory_path]
   [WITH DBPROPERTIES (property_name=property_value, ...)];
   ```

   Do not set LOCATION and MANAGEDLOCATION to the same path. For more information, see Create a default directory for managed tables.
5. In Ranger, grant the tenant users access to the tenant-specific Hive or Impala database.

Tenant IAM role policy

The JSON code for the EKS worker node policy describes the policy. The policy is representative of one of the three policies you add to the IAM role.

```
{
   “Version”: “2012-10-17",
    “Statement”: [
    {
```

```
      “Sid”: “VisualEditor0”,
      “Effect”: “Allow”,
      “Action”: [
        “kms:Decrypt”,
        “s3:ListAllMyBuckets”,
        “kms:Encrypt”,
        “kms:ListAliases”,
        “kms:GenerateDataKey”,
        “kms:DescribeKey”
        ],
      “Resource”: “*”
    }
    ]
}
s3-read-write
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:Delete*",
                "s3:Put*",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::<SDX_BUCKET>",
                "arn:aws:s3:::<SDX_BUCKET>/*",
                "arn:aws:s3:::tenant-1-bucket-1",
                "arn:aws:s3:::tenant-1-bucket-1/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

In Cloudera Data Warehouse, you need full access to the SDX bucket, even if you used only folder level while creating the CB environment.

Using AWS Vault to manage credentials

Instead of using a jceks file to store credentials, or exposing your user name and password through some other means of credential management, you learn how to use AWS Secrets Service for storing credentials in the AWS Vault.

Procedure

1. Log into AWS.
2. From the IAM console Roles on AWS select your role.
3. Click your IAM role to navigate to its summary, and copy the Role ARN.
4. Create your AWS secret or look for it in your .aws/credentials directory.
5. In the AWS Secrets manager, modify secrets permissions, substituting your role arn and secret resource arn for the variables:

```
{
  "Version" : "2012-10-17",
```

```
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::NNNN:role/env-AAAAAA-dwx-stack-NodeInstanceRol
e-XXXXXX"
    },
    "Action" : [ "secretsmanager:GetSecretValue", "secretsmanager:Descr
ibeSecret" ],
    "Resource" : "arn:aws:secretsmanager:us-west-2:NNNN:secret:<secret-nam
e>-dsffd"
  } ]
}
```

6. Create a table that specifies the JDBC-URL, your role arn, and secrets arn in the table properties.
   For example:

```
ceate external table customer_address (
   column definitions ...
)
STORED BY 'org.apache.hive.storage.jdbc.JdbcStorageHandler'
TBLPROPERTIES (
    "hive.sql.database.type" = 'POSTGRES',
    "hive.sql.jdbc.driver" = 'com.mysql.jdbc.Driver',
    "hive.sql.jdbc.url" = '<jdbc-url>',
    "hive.sql.dbcp.username" = '<role-name>',
    "hive.sql.dbcp.password.uri" = 'aws-sm:///<secret-arn>',
    "hive.sql.table" = "<table-name>"
);
```

7. Create a Hive table that specifies the JDBC-URL, your database user name, and secrets arn in the table properties.

```
create external table customer_address (
   column definitions ...
)
STORED BY 'org.apache.hive.storage.jdbc.JdbcStorageHandler'
TBLPROPERTIES (
    "hive.sql.database.type" = 'POSTGRES',
    "hive.sql.jdbc.driver" = 'org.postgresql.Driver',
    "hive.sql.jdbc.url" = '<jdbc-url>',
    "hive.sql.dbcp.username" = '<database-user-name>',
    "hive.sql.dbcp.password.uri" = 'aws-sm:///<secret-arn>',
    "hive.sql.table" = "<table-name>"
);
```

   Only the password field is extracted from the secret.

Identifying the spill location for Impala temporary data

Impala writes temporary data to S3 to prevent a memory overflow. The result is successful completion of a query instead of an out-of-memory error. When you create an Impala Virtual Warehouse, a path to write temporary data to the Data Lake S3 bucket is configured automatically. You need to know the name of the bucket and then allow Impala to access it.

About this task

The Impala Virtual Warehouse attempts to write temporary data to the local Non-Volatile Memory Express (NVMe) SSD on compute instances before spilling to Data Lake S3 bucket. The attempt succeeds only if you configure Impala access to the default scratch location on DataLake S3 bucket.

Before you begin

- Obtain the DWAdmin role.
- Your Impala Virtual Warehouse spill-to-Data-Lake-bucket was configured automatically when you created a new Virtual Warehouse; you are not using an existing Virtual Warehouse that required a manual spill-to-S3 configuration.

Procedure

1. From the Cloudera Management Console or Cloudera landing page, navigate to Data Warehouses.
2. Go to the Impala Virtual Warehouse ⋮ Edit Configurations Impala coordinator and select flagfile from the Configuration files drop-down list.
3. Note the value of the scratch_dirs property.

   The first path segment of the value is the default scratch location in the Data Lake bucket. For example, qe-s3-bucket-weekly is the S3 bucket name of the default scratch location.

   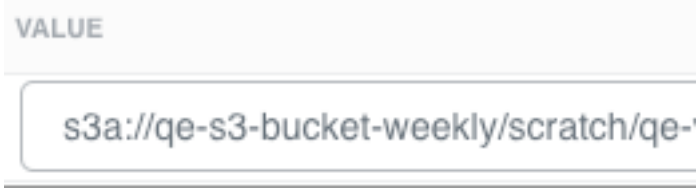

4. Go to the **Impala executor** tab and select flagfile from the Configuration files drop-down list.
5. Note the value of the scratch_dirs property.
6. Configure access to the default scratch location using the bucket name, for example qe-s3-bucket-weekly, as described in Accessing S3 buckets.

Related Information

Tuning Impala Virtual Warehouses on Public Cloud

Configuring an existing Impala Virtual Warehouse to spill to S3

A new Impala Virtual Warehouse requires no configuration to spill to S3. However, if you have an existing Impala Virtual Warehouse that you did not configure to spill to S3 when you created the Virtual Warehouse, configuration is required.

About this task

The Impala Virtual Warehouse on an AWS environment writes temporary data to S3 when you configure a spill to S3. This capability in an existing, but not a new, Virtual Warehouse, requires an entitlement.

If you have an existing Impala Virtual Warehouse, you need to take the following actions:

- Edit your existing Virtual Warehouse to specify an S3 URI to spill to S3.
  - After editing, you cannot change the S3 URI.
  - After editing, you cannot select additional storage in Scratch Space Limit per node. The default 300 instance storage is used.
- Ensure that Impala has read/write access to the configured scratch location on the Data Lake bucket using steps in Configuring a policy to spill Impala temporary data to S3.

Alternatively, instead of using the automatic default scratch location or to configuring the location, you can run the following Cloudera CLI command `create-vw` to configure a custom scratch location. Specify the spill location using the impala-options option using the spillToS3Uri field.

After you have created the Virtual Warehouse configured to spill to a specific S3 location, you cannot change the S3 URI. The field becomes uneditable.

Before you begin

- To use an external S3 bucket for spilled data, add an external S3 bucket to Cloudera Data Warehouse.
- Note the URI of the external S3 bucket you added. For example, S3://mybucket/scratch/path.

Procedure

1. From the Cloudera Management Console or Cloudera landing page, navigate to Data Warehouses.
2. Go to the Virtual Warehouses tab.
3. Click ⋮ Edit .
4. Set the spill to S3 location.
5. Click Save.
6. Configure read/write access to the configured scratch location on the Data Lake bucket using steps in Identifying the spill location for Impala temporary data.

Setting the scratch space limit for spilling Impala queries

Running out of space to store query data on the SSD causes query failure. To prevent these failures, you need to know how to configure scratch space limits for Impala Virtual Warehouse (Cloudera Data Warehouse) service.

About this task

Configuring scratch space on EBS volumes incurs additional cost and requires an entitlement specified below.
The compute nodes that are used for coordinator and executor Kubernetes pods in Impala Virtual Warehouses are equipped as follows:

- Attached solid state drive (SSD) storage, using NVMe (non-volatile memory express) protocol.
- Instance storage 2x300 GB allocated as follows:
  - Data cache (200 GiB default value)
  - Scratch space (200 GiB default value)

    Scratch space requires a base 2 storage increment, such as GibiBytes.

If you have spilling queries that require more scratch space than your compute nodes have, you need to configure additional scratch space for Impala Virtual Warehouses on AWS Elastic Block Store (EBS). You configure scratch space limits when you create an Impala Virtual Warehouse. After the Virtual Warehouse is created, you cannot add scratch space limits by editing the configuration. Configuring scratch space on EBS volumes incurs additional cost. See Amazon EBS Pricing for details.

Before you begin

You must obtain the CDW_IMPALA_EBS_SCRATCH_SPACE entitlement to use this feature.

Procedure

1. Create a Virtual Warehouse as described in Adding a new Virtual Warehouse, selecting the Impala engine type.

2. Select a database catalog, or accept the default.
3. Set User Groups that can access endpoints, keys and values for Tagging the Virtual Warehouse.
4. Select a size for the Virtual Warehouse.
5. In Scratch Space Limit per node (in GiBs), choose to only use the 300 GiB of instance storage for scratch space, or choose additional EBS volumes for spilling queries.

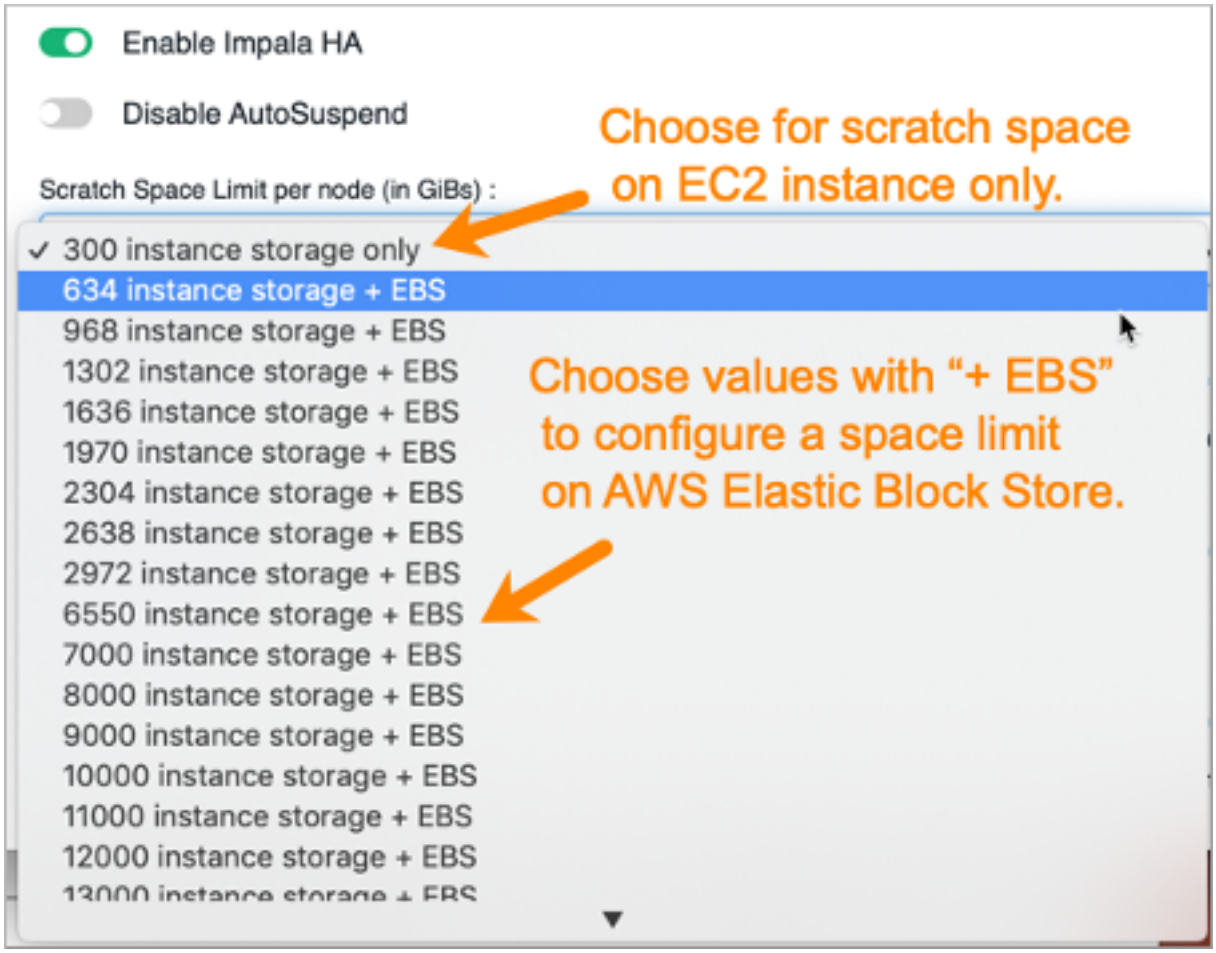

6. Click Create Virtual Warehouse.

Cloudera Data Warehouse Public Cloud

Azure environments

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Azure environments overview 5

Azure environments requirements checklist 5

Activating an Azure environment from Cloudera Data Warehouse 8
Retaining PostgreSQL backups in Azure Environments 9

Setting up minimum permissions 9

Viewing and editing Azure environment details 11

Enabling a private Cloudera Data Warehouse environment 11
Setting up the environment for private cluster deployment 12
Creating and registering the Azure environment 13
Configuring a custom private DNS zone 15
Configuration options for a private Cloudera Data Warehouse deployment 15

Overlay networking 16
Enabling CNI overlay networking 17

Using AKS monitoring with Cloudera Data Warehouse in Azure environments 18

Deactivating environments on Azure 18

Custom tags in Azure environments 19

Azure load balancers in Cloudera Data Warehouse 19
Enabling internal load balancer 21

Granting remote access to Kubernetes clusters on Azure Kubernetes Service 22
Revoking remote access to Kubernetes clusters on Azure Kubernetes Service 25

Managed storage access 25
Setting up managed storage access 27

Creating the Cloudera environment..... 27
Creating a UMS group and machine users..... 28
Creating a new Database Catalog..... 30
Creating a tenant-specific Virtual Warehouse..... 30

Azure Kubernetes Service upgrade..... 31

Setting scratch space limit for spilling Impala queries in Azure environments..... 31

Azure environments overview

Learn about environments on Azure in Cloudera Public Cloud, including requirements you must meet before activating your environment in Cloudera Data Warehouse.

The Cloudera environment is closely related to the virtual private network in your cloud provider account. Registering an environment with Cloudera Management Console provides Cloudera with access to your cloud provider account and identifies resources in your account that Cloudera services can access, including Cloudera Data Warehouse. A single environment is contained within a single cloud provider region, so all resources deployed by Cloudera are deployed within that region within one specific virtual network. After you have registered an environment with Cloudera Management Console you can activate the environment in Cloudera Data Warehouse. Next, you create Database Catalogs, which enables Cloudera Data Warehouse to access the associated Data Lake. Finally, you can create Virtual Warehouses in Cloudera Data Warehouse that use the Database Catalog and its underlying environment.

Ranger Authorization is not enabled by default in Azure environment. You must enable RAZ as described in the Introduction to RAZ on Azure environments.

The following diagram shows the components of an Azure environment:

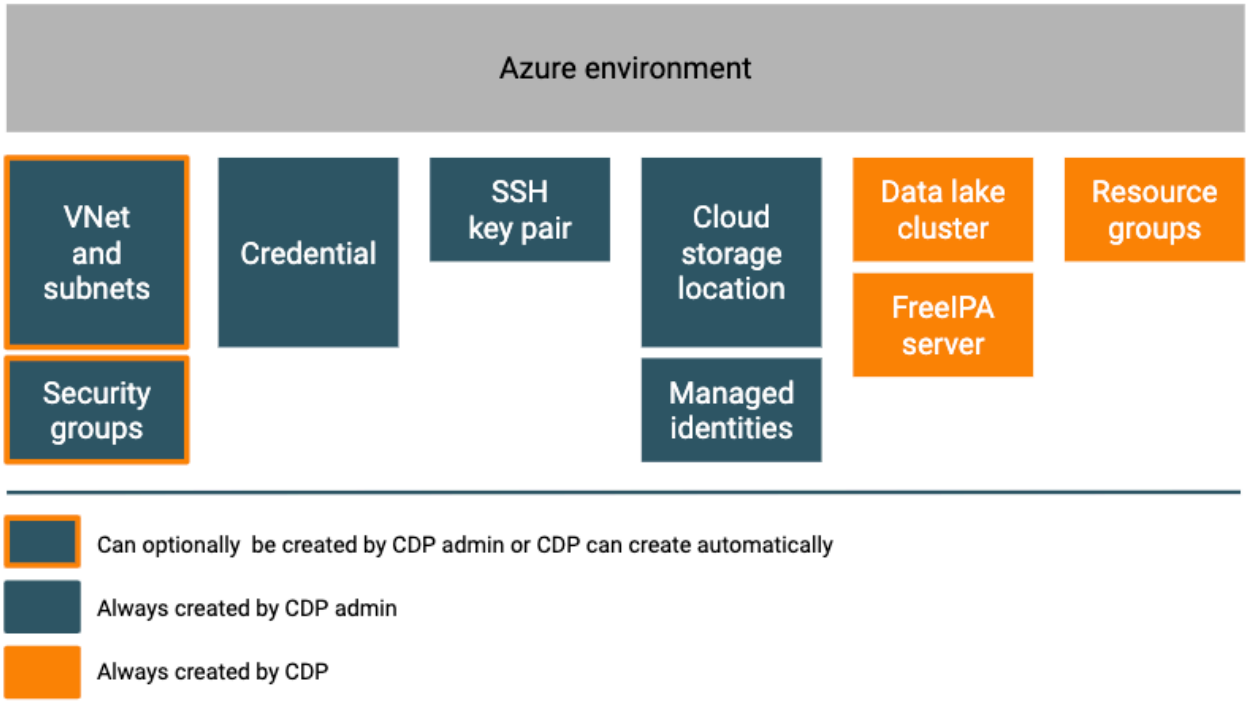

For more information about Azure environments in Cloudera, see the links in the "Related information" section at the bottom of this page.

Azure environments requirements checklist

To successfully activate Azure environments with Cloudera Data Warehouse service, make sure your environment meets the requirements listed in this topic.

1. Specific networking requirements

Important: Cloudera recommends activating Cloudera Data Warehouse service and Cloudera Machine Learning service workloads on different subnets to maximize performance.

- Make sure Azure VNet subnets are large enough to support the Cloudera Data Warehouse load

  When an Azure environment is activated for Cloudera Data Warehouse service, an Azure Kubernetes Service (AKS) cluster is provisioned in your subscription. The AKS cluster uses the Azure Container Networking Interface (CNI) plug-in for Kubernetes. This plug-in assigns IP addresses for every pod running inside the Kubernetes cluster. By default, the maximum number of pods per node is 30. This means that you need approximately 3,200 IP addresses for a 99-node cluster. If you activate an environment for Cloudera Data Warehouse service, make sure that the subnets are large enough on the Azure VNet for the Cloudera Data Warehouse load. Cloudera recommends using a CIDR/21 subnet or larger.

  The following IP address ranges are defined as part of the AKS cluster creation process:

  - 10.20.0.0/16
  - 172.17.0.1/16

  Make sure these ranges do not overlap with your VNET address ranges.

  If you want to activate Cloudera Data Warehouse on a very small subnet, overlay networking is recommended.
- Configure service endpoints on Cloudera Data Warehouse subnets

  You must configure service endpoints on the subnets used for the Cloudera Data Warehouse service. This ensures that the network traffic between Cloudera Data Warehouse components and Azure services remain on the Microsoft Azure backbone network. Microsoft.Storage and Microsoft.SQL must be registered; otherwise, the Cloudera Data Warehouse service does not start on existing Azure VNets.

  Note: If you have used only Data Hub in your Cloudera environment, you must check and possibly configure connections to additional outbound destinations for Cloudera Data Warehouse.

  For more information, see Azure Outbound Network Access Destinations and Virtual Network service endpoints in the Azure documentation.
- Firewall exceptions for Azure AKS

  If you need to restrict egress traffic in Azure, reserve a limited number of ports and addresses for cluster maintenance tasks including cluster provisioning. See Control egress traffic for cluster nodes in Azure Kubernetes Service (AKS) to prepare your Azure environment for AKS deployment.

  For maintenance, Cloudera recommends putting the Azure portal URLs on the allowlist of your firewall or proxy server. For more information, see Allow the Azure portal URLs on your firewall or proxy server.

2. Use only app-based credentials

For the Cloudera Data Warehouse service, you must use only an app-based credential, which requires the Contributor role to create a new service principal. For more information about creating an app-based credential for the environment you want to use for the Cloudera Data Warehouseservice, see Create an app-based credential. If you need to change your environment credential, see Change environment's credential.

3. Obtain permissions

For environments that you plan to use for the Cloudera Data Warehouse service, generally you need to ensure that the application you create in Azure has the built-in Contributor Azure role at the resource group level. For more information, see the description of app-based credentials in Credential options on Azure in the Cloudera Management Console documentation.

Important: Alternatively, you can use minimum permissions.

4. Created Azure app must have access to the storage account used during environment registration

Ensure that the application, which the Azure app-based credentials are attached to, have access to the ADLS Gen2 storage location you specify in Step 6 when you register an Azure environment topic. For information about storage accounts for Azure environments, see ADLS Gen2 and managed identities and Minimal setup for cloud storage.

5. AKS cluster credential

During the environment activation process, you must provide the resource ID of a user-assigned managed identity to create the Azure Kubernetes cluster resource. This crucial identity allows for impersonation of the AKS cluster. To ensure this functionality, you must assign the appropriate permissions to the identity resource. Cloudera strongly recommends assigning the same permissions as those assigned to the app-based credentials. Additionally, you must grant the Storage Blob Data Owner role to the AKS identity on your Data Lake storage account. This permission is required for activating an environment in Cloudera Data Warehouse.

To affirm that your resources, such as managed identities, virtual networks, and so on, are accessible, verify that you have assigned the appropriate permissions to any separate resource groups where they reside.

For more information about managed identities, see "AKS managed identities".

6. List of required resources for Azure environments

There is no cross-regional support for Cloudera Data Warehouse service. Azure environments used for the Cloudera Data Warehouse must have the following resources available in the specific Azure region where the environment is registered:

- Azure Kubernetes Service (AKS)
- Azure Database for PostgreSQL
- Azure Data Lake Storage Gen2
- Virtual machine scale sets
- Ev3 series VM
- Dsv3-series VM
- Availability zones for AKS (optional)

Important: To enable the Availability zones for AKS option when activating the environment for the Cloudera Data Warehouse service, make sure your region supports Azure Availability.

7. Azure subscription should be in a similar region as the resources

Ensure that your Azure subscription is in close proximity to the region where your resources are deployed and are governed by the same regulatory laws. The Azure region requirements says, "Cloudera requires that the ADLS Gen2 storage location provided during environment registration must be in the same region as the region selected for the environment." For more information, see Azure geographies.

8. Support for third party k8s components

To minimize the IP address usage of the AKS clusters, Cloudera currently creates the Kubernetes (k8s) compute node pools with a “maxpods” settings of 15. This setting specifies the maximum number of pods that can run on a node. By default, Azure deploys several key components to every k8s cluster to provide infrastructure services. The number of components can vary based on the enabled services in your Azure subscription. Generally it takes 6-8 pods to host these infrastructure services inside the AKS clusters. Additionally Cloudera deploys 1-2 other pods to host Cloudera compute resources.

If you want to deploy services to the AKS cluster, make sure you do not exceed the maxpods limits of the cluster.

Activating an Azure environment from Cloudera Data Warehouse

To use an Azure environment for Cloudera Data Warehouse Public Cloud you must first activate it.

About this task

When you activate an environment, Cloudera creates an AKS (Azure Kubernetes Services) cluster to host Kubernetes-based resources. The underlying compute, network resources are managed by Azure, including the following ones:

- Resource group
- Load balancer(s)
- Public IP address(es)
- Network security group
- Disk(s)

Three instances are added to the cluster as needed for shared services (always on components). Three shared nodes are dStandard_E2s_v3 MemoryOptimized using flexserver, for the Amazon Relational Database Service (RDS). These shared nodes are used for Hue and Data Visualization user metadata. For more information, see Always active, shared services.

Deploying a private AKS

If you want to deploy a Private AKS cluster, you must use the CDP CLI as described in AKS deployment configuration options.

Before you begin

- Obtain the DWAdmin role.
- Review the Azure environments requirements checklist.

Procedure

1. In the Cloudera Data Warehouse service, in Environments, locate the environment that you want to activate.
2. Click Activate.
3. In Activate Environment, configure the environment or accept the defaults:
   - Select a Subnet inside the virtual network (VNet) that you want to use for Cloudera Data Warehouse.

     The VNet that you select must have a sufficient number of free IP addresses.
   - Select Enable internal load balancer (ingress) to distribute traffic inside a virtual network.

     For information about load balancers, see "Azure load balancers".
   - Optionally, select Enable availability zones for AKS.
   - Specify a user-assigned, managed identity for the AKS cluster.
   - Select Enable AKS monitoring and then select the workspace from the adjacent drop-down list.
   - Add trusted endpoint IP CIDRs for your load balancer in a comma-separated list to the Enable IP CIDR for the load balancer text box.
   - Accept the default Use CNI overlay networking if IP address exhaustion is a concern for your deployment.
4. To use custom repository, select the Use Custom repository option.

   Depending in your custome repository type, input ecr, acr, or docker in the Registry Type field.

   Specify the URL of your custom repository. Fields for your user name and password appear. You do not need to enter these credentials.

5. Click ACTIVATE.

Related Information

Azure Standard_E16ds_v4 instances
Azure Standard_E16_v3 instances
Bring your own control plane managed identity
Overlay networking

Retaining PostgreSQL backups in Azure Environments

When you create a Cloudera Data Warehouse cluster using the CDP CLI create-cluster command, any PostgreSQL backup retention period you set on your Cloud Provider side, is observed by Cloudera.

Procedure

1. In Azure, configure backupRetentionDays.
2. Create a DW cluster using the CDP CLI create-cluster command.
   The DW cluster will retain the PostgreSQL backups according to your configuration.

Setting up minimum permissions

You learn how to configure the minimum permissions required for Cloudera Data Warehouse on Azure environments. The minimum permissions govern access control between Cloudera Data Warehouse, Azure resources, and the Azure storage account.

Before you begin

Obtain the permissions and roles as described in "Azure permissions".

Procedure

1. Write code for a custom role that contains the minimum permissions required for Cloudera Data Warehouse on Azure.
2. In the properties section of your subscription code, change variables into actual values.
   For example, make the following substitutions to the sample code shown below:
   - [YOUR-SUBSCRIPTION-ID]: Your subscription ID in use.
   - [YOUR-RESTRICTED-ROLE-NAME]: The custom role name which is assigned to the application.
   - [YOUR-SINGLE-RESOURCE-GROUP]: The original resource group name.
3. Insert the code for the custom role into the actions section of your subscription.
   The following sample code shows the properties and actions sections you need to add to your subscription:

```
{
   "properties": {
       "roleName": "[YOUR-RESTRICTED-ROLE-NAME]",
       "description": "Additional permissions to activate Cloudera Data
 Warehouse clusters.",
       "assignableScopes": [
    "/subscriptions/[YOUR-SUBSCRIPTION-ID]/resourceGroups/[YOUR-SINGLE-RES
OURCE-GROUP]"
       ],
       "permissions": [
           {
               "actions": [
                   "Microsoft.Resources/deployments/cancel/action",
                   "Microsoft.Resources/deployments/validate/action",
```

```
                    "Microsoft.ContainerService/managedClusters/write",
                    "Microsoft.ContainerService/managedClusters/agentPools/
write",
                    "Microsoft.ContainerService/managedClusters/read",
                    "Microsoft.ContainerService/managedClusters/agentPools/
read",
                    "Microsoft.ContainerService/managedClusters/accessProf
iles/listCredential/action",
                    "Microsoft.ContainerService/managedClusters/delete",
                    "Microsoft.ContainerService/managedClusters/rotateCl
usterCertificates/action",
                    "Microsoft.DBforPostgreSQL/flexibleServers/read",
                    "Microsoft.DBforPostgreSQL/flexibleServers/write",
                    "Microsoft.DBforPostgreSQL/flexibleServers/delete",
                    "Microsoft.DBforPostgreSQL/flexibleServers/firewallR
ules/write",
                    "Microsoft.DBforPostgreSQL/flexibleServers/firewallR
ules/read",
                    "Microsoft.DBforPostgreSQL/flexibleServers/firewallRu
les/delete",
                    "Microsoft.DBforPostgreSQL/flexibleServers/configura
tions/read",
                    "Microsoft.DBforPostgreSQL/flexibleServers/configura
tions/write",
                    "Microsoft.DBforPostgreSQL/flexibleServers/databases/re
ad",
                    "Microsoft.DBforPostgreSQL/flexibleServers/databases/
write",
                    "Microsoft.DBforPostgreSQL/flexibleServers/databases/
delete",
                    "Microsoft.DBforPostgreSQL/servers/virtualNetworkRules/
write",
                    "Microsoft.DBforPostgreSQL/servers/databases/write",
                    "Microsoft.Network/privateDnsZones/A/read",
                    "Microsoft.Network/privateDnsZones/A/write",
                    "Microsoft.Network/privateDnsZones/A/delete",
                    "Microsoft.Network/privateDnsZones/virtualNetworkLinks/
read",
                    "Microsoft.Network/virtualNetworks/subnets/joinViaSe
rviceEndpoint/action",
                    "Microsoft.Network/routeTables/read",
                    "Microsoft.Network/routeTables/write",
                    "Microsoft.Network/routeTables/routes/read",
                    "Microsoft.Network/routeTables/routes/write",
                    "Microsoft.Network/routeTables/join/action",
                    "Microsoft.Network/natGateways/join/action",
                    "Microsoft.Network/virtualNetworks/subnets/joinLoadBal
ancer/action",
                    "Microsoft.Network/privateDnsZones/write",
                    "Microsoft.Network/privateDnsZones/read",
                    "Microsoft.Network/privateDnsZones/virtualNetworkLinks/
write",
                    "Microsoft.Network/privateEndpoints/write",
                    "Microsoft.Network/privateEndpoints/read",
                    "Microsoft.Network/privateEndpoints/privateDnsZoneGr
oups/read",
                    "Microsoft.Network/privateEndpoints/privateDnsZoneGro
ups/write",
                    "Microsoft.Network/privateEndpoints/privateDnsZoneGro
ups/delete",
                    "Microsoft.Network/privateDnsZones/join/action"
                ],
                "notActions": [],
                "dataActions": [],
```

```
                "notDataActions": []
            }
        ]
    }
}
```

Viewing and editing Azure environment details

After activating an Azure environment for Cloudera Data Warehouse Public Cloud, you can change or view its details.

About this task

You can view Cloudera Azure environment details without leaving the Cloudera Data Warehouse service UI. You can make the following changes:

- Add a description for the environment, which makes it easier to identify.
- Add or edit the list of IP CIDR(s) for Kubernetes cluster, which enable access from your internal network to the Kubernetes endpoints.
- Add or edit the list of IP CIDR(s) for the load balancer, which enable access from your internal network to the load balancer endpoints of services such as Hive, Impala, or Hue.
- Renew the certificate for this environment.

  This option renews the encrypt certificates. Renewal happens automatically 30 days before the expiry date of the certificate.
- Refresh kubeconfig

  After making changes that alter the Kubernetes configuration, such as rotating certificates, as recommended by Microsoft, you must click Refresh kubeconfig.

Required role for viewing environment details: DWUser

Required role for editing environment details: DWAdmin

Before you begin

You must activate an environment before you can view or edit its details.

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment that you want to view and click ⋮ Edit .

   The **Environment Details** page is displayed.
3. Go to the Configurations tab.
4. Make changes described above, or just view information, such as when it was created and last updated, or how many Database Catalogs and Virtual Warehouses use the environment,
5. If you have made changes, click Apply Changes.

Enabling a private Cloudera Data Warehouse environment

An introduction to the terms and components involved in creating a private Cloudera Data Warehouse environment prepares you to successfully complete the required tasks. You see a network configuration diagram.

AKS simplifies container-based application deployment and management. When you create an AKS cluster, a control plane is automatically created and configured. The Azure platform configures the secure communication between the control plane and compute nodes. In a private cluster, the API server for the control plane has an internal IP address. The Azure nodes and the Kubernetes control plane components do not have publicly routable IP addresses. When you deploy a private cluster, the network traffic between your API server and your node pools remains only on the private network.

Cluster Connectivity Manager {version 2} enables the Cloudera Control Plane to communicate with the Kubernetes control plane using an inverting proxy solution. Cloudera Data Warehouse can communicate with the Kubernetes control plane and the other resources, such as virtual machines deployed in your network, by using a special established channel.

Large enterprises have a requirement to create private Cloudera Data Warehouse environments in the public cloud similar to the on-premises one in terms of security and networking. Generally, public access to your cloud environment, resources, and deployed applications is not allowed.

A private environment has the following criteria:

- Kubernetes API server for the control plane must have private API endpoints
- Hue, DAS, JDBC, or ODBC applications can only be accessed through private endpoints
- Egress Kubernetes traffic must be controlled using a transparent proxy
- Storage, SQL database, and other resources can only be accessed through private endpoints
- Cloudera Control Plane can only be accessed through private endpoints using Azure Private Link Service
- Ability to configure a DNS zone in your environment

The following layout shows a typical network for private environments:

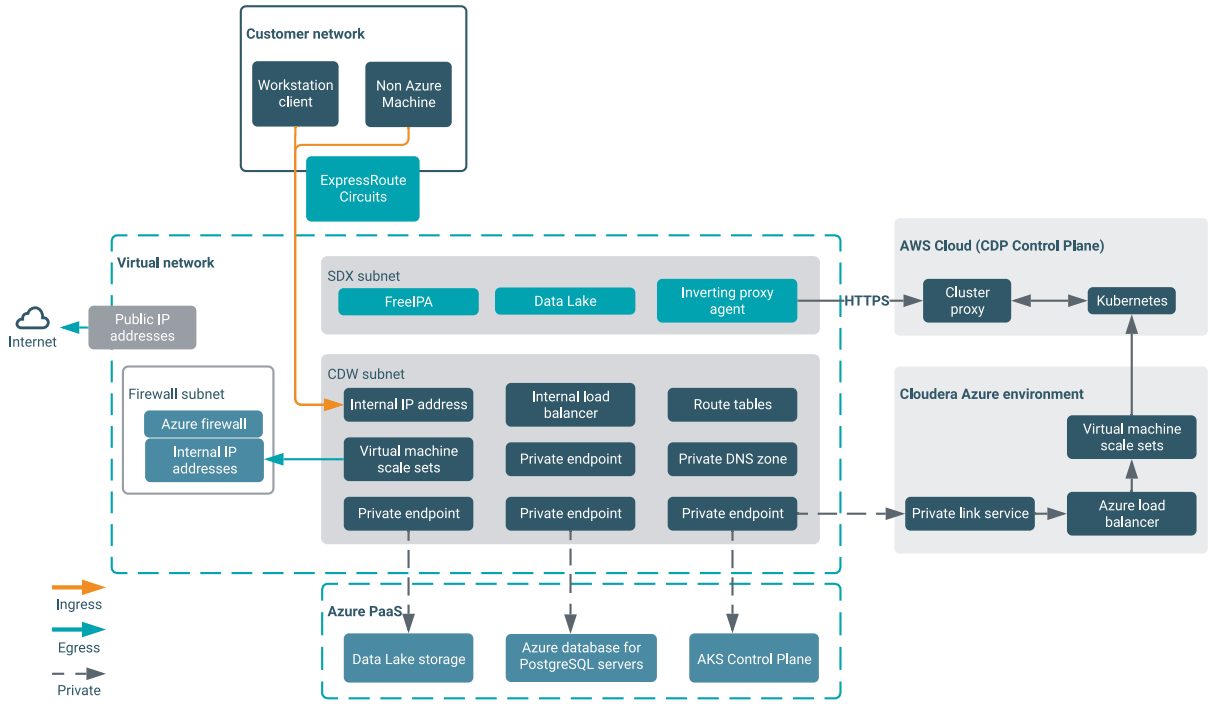

Setting up the environment for private cluster deployment

In Azure, you perform steps and follow links to Azure documentation to meet prerequisties for enabling a private Cloudera Data Warehouse environment.

Procedure

1. In the Microsoft Azure portal, create a resource group for Cloudera.

   The resource group provides a management layer that enables you to create, update, and delete resources in your Azure account.
2. (Optional)Create a private storage account and network access rules to block all internet traffic.

   The private endpoints allow clients on a virtual network (VNET) to securely access data over a Private Link.
3. Create a VNet and add a subnet.

   VNet is the fundamental building block for your private network in Azure. It enables Azure resources, such as Azure Virtual Machines, to securely communicate with each other and with the internet.
4. Create a delegated subnet for Azure Database for PostgreSQL Flexible Server.
5. (Optional) Configure the Cloudera Control Plane Private Link service.

   To ensure that all egress traffic including Cloudera Control Plane traffic travels only through private networks, you must configure Cluster Connectivity Manager (v2) and Cloudera Control Plane private endpoints properly for the selected VNet.

   Contact your Cloudera account representative to register the Cloudera Control Plane Private Link service endpoints in your Azure VNet.
6. Configure custom DNS on the VNet to resolve Azure Private DNS zones.

   To resolve private endpoint DNS records, the VNet DNS servers must be capable of resolving Azure DNS records.
7. Disable network endpoint policies for private endpoints and Azure Private Link Service.
8. Configure the following firewall exceptions for Cloudera Data Warehouse and AKS on the egress firewall, and for storage account endpoints, if the account is not Private-Link enable: Outbound network access destinations and Restrict egress traffic in AKS
9. Configure user-defined routing (UDR) on the VNet to forward all traffic to an egress firewall and link it to the subnet.

Creating and registering the Azure environment

In Cloudera, you perform a step-by-step procedure to create the environment, and select options for the database, virtual machines, and Cluster Connectivity Manager during registration.

About this task

Procedure

1. Create a Cloudera Azure environment in the VNET that you created earlier.
2. Select private environment options for the PostgreSQL database, virtual machines, and Cluster Connectivity Manager (v2) while registering the Azure environment in Cloudera.

   On the environment registration screen, the Enable Cluster Connectivity Manager option is enabled by default. It ensures that all traffic from Cloudera Control Plane to your cloud resources travels through a secured HTTPS tunnel. Cloudera Data Warehouse Private AKS only works with Cluster Connectivity Manager (v2).

   Do not create public IPs so that Azure VMs have private IP addresses only.
3. Enable the Create Private Endpoints option.

   By default, the PostgreSQL Azure database provisioned for your Data Lake is reachable through a service endpoint (public IP address). To increase security, you must select to have it reachable through a private endpoint

instead of a service endpoint. You can create a new private DNS zone for the database or you can use your own. Cloudera Data Warehouse will also use the zone specified here.

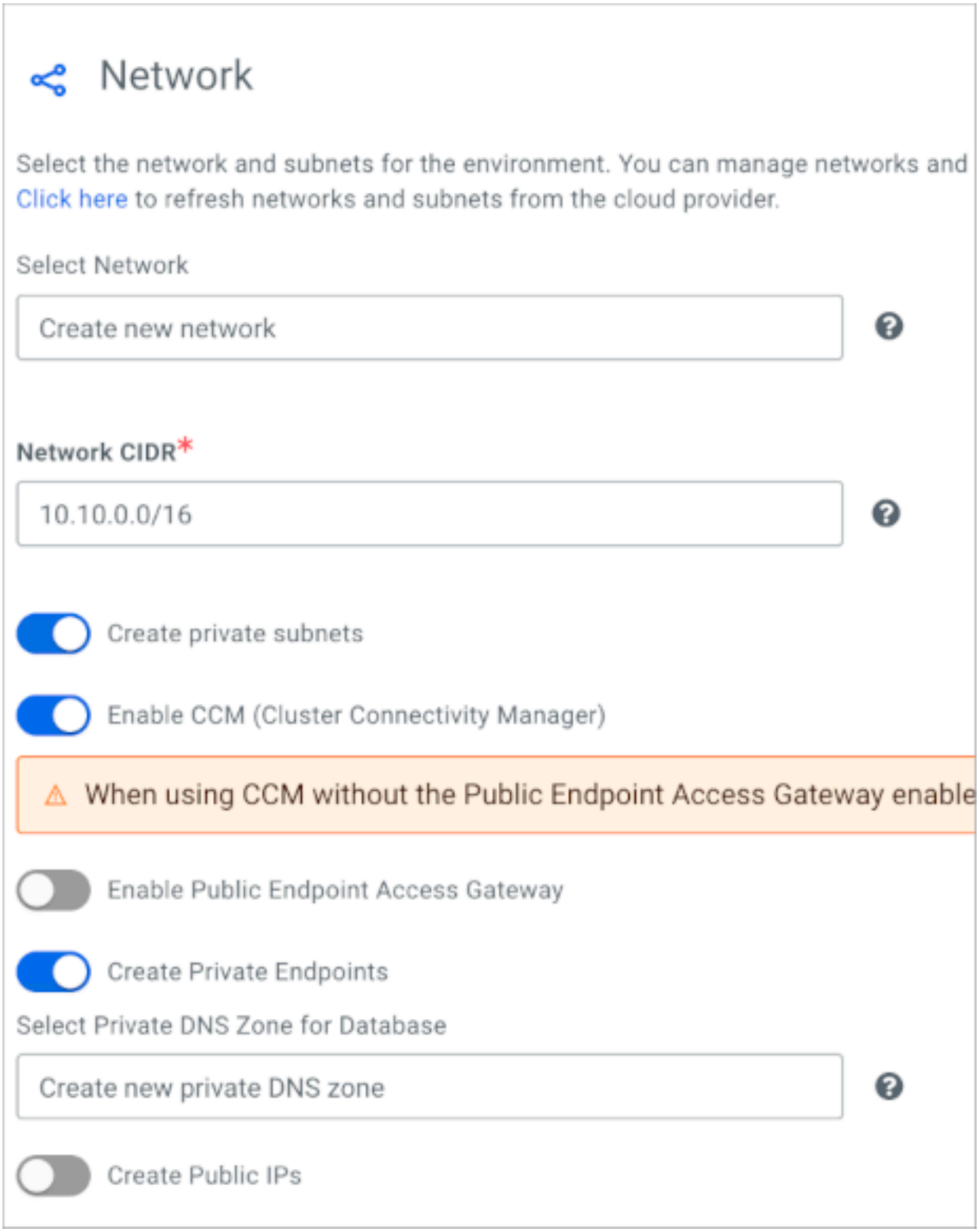

Related Information

Working with Azure environments

Register an Azure environment from Cloudera UI

Enabling private endpoint for PostgreSQL on Azure

Configuring a custom private DNS zone

You learn the requirements for configuring a DNS zone for AKS and Azure Database for PostgreSQL - Flexible Server. From the syntax information, you see how to construct a Cloudera CLI command that configures the DNS zone.

About this task

When you set up the environment for private cluster deployment, you configure custom DNS on the VNET to resolve Azure Private DNS zones. If this default AKS Private DNS zone creation behavior is not suitable for your needs, you can use the CDP CLI to further customize the zones. Activating Cloudera Data Warehouse clusters with the private Cloudera Data Warehouse option enables multiple different AKS features at once to support private Cloudera Data Warehouse deployments.

Before you begin

- Create the DNS zone privatelink.<region>.azmk8s.io.
- Grant at least the private dns zone contributor and vnet contributor roles to the service principal (Azure Enterprise Application) used for environment creation.
- If the Private DNS Zone is in a different subscription than the AKS cluster, you need to register Microsoft.ContainerServices in both the subscriptions.

Procedure

1. Configure a DNS zone for AKS using the CDP CLI.

   Use the following syntax:

   ```
   cdp dw create-cluster --<create_cluster_options> \
   --environment-crn <CLOUDERA_RESOURCE_NAME_OF_ENVIRONMENT> --use-private-l
   oad-balancer \
   --azure-options <property>=<value>,<property>=<value> ... ,\
   enablePrivateAks=true,\
   privateDNSZoneAKS=<PRIVATE_DNS_ZONE_RESOURCE_ID>
   ```

   where: Azure options might be userAssignedManagedIdentity={managed- identity},subnetId={subnet-id},outbo
   undType=UserDefinedRouting,enablePrivateSQL=true, and the PRIVATE_DNS_ZONE_RESOURCE_ID looks something like this:

   ```
   privateDNSZoneAKS=/subscriptions/{sub-id}/resourceGroups/{rg-name}/provi
   ders/Microsoft.Network/privateDnsZones/privatelink.{region}.azmk8s.io
   ```

   If you specify none for the privateDNSZoneAKS parameter, Azure defaults to public DNS which means AKS does not create a Private DNS Zone.

2. Create a DNS zone for Azure Database for PostgreSQL.

   When you set up the environment for private cluster deployment, you need to specify a custom private DNS zone. See the Use a Private DNS zone topic in Azure documentation to prepare your DNS infrastructure.

Related Information

Configure a private DNS Zone

Configuration options for a private Cloudera Data Warehouse deployment

The list of properties and explanations of how to use them, followed by an example, prepares you to configure a private Cloudera Data Warehouse deployment.

To customize the Cloudera Data Warehouse deployment, you can use the following properties and values after the --azure-options parameter:

outboundType

You can customize an AKS cluster with a unique outboundType depending on your network configuration for egress traffic.

Possible values: LoadBalancer/UserDefinedRouting/UserAssignedNATGateway

enablePrivateSQL=true/false

If you set this option to true, Cloudera Data Warehouse creates an Azure Database for PostgreSQL - Flexible Server with Private access (virtual network integration) enabled to bring the server inside your Virtual Network (VNET).

privateSQLSubnetName

You can specify the delegated subnet for Azure Database for PostgreSQL - Flexible Server.

privateDNSZoneSQL

You can specify the private DNS zone for Azure Database for PostgreSQL - Flexible Server.

enablePrivateAks=true/false

By using a private cluster, you can ensure network traffic between your API server and your node pools remains on the private network. Enabling this option instructs Cloudera Data Warehouse services to create an AKS cluster with a private API.

The --create-cluster options are:

- --use-private-load-balancer
- --no-use-private-load-balancer

An internal load balancer makes a Kubernetes service accessible only to applications running in the same virtual network as the Kubernetes cluster.

The following example shows how to configure a DNS zone for AKS:

```
cdp dw create-cluster --environment-crn --use-private-load-balancer \
--azure-options userAssignedManagedIdentity={managed-identity},subnetId={s
ubnet-id},enablePrivateSQL=true,\
enablePrivateAks=true, \
privateDNSZoneAKS=/subscriptions/subid/resourceGroups/rgname/providers/Mi
crosoft.Network/privateDnsZones/privatelink.{region}.azmk8s.io
```

Related Information

Customize user-defined routes (UDR) in Azure Kubernetes Service (AKS) - Azure Kubernetes Service

Private Link Azure Database for PostgreSQL Single server

Create a private Azure Kubernetes Service cluster - Azure Kubernetes Service

Create an internal load balancer - Azure Kubernetes Service

Create an AKS cluster with a user-assigned NAT Gateway

Overlay networking

By default, Cloudera Data Warehouse uses CNI overlay networking.

Default Cloudera Data Warehouse networking

Using Azure Container Networking Interface (CNI), every pod gets an IP address from the node subnet and is accessed directly. Each IP address must be unique across your network space, and you must plan for them in advance of deploying your Cloudera Data Warehouse cluster. Each node has a configuration parameter for the maximum

number of pods that it can support. The equivalent number of IP addresses per executor node is reserved up front for it. This requires advanced planning and it can often lead to IP address exhaustion. As an alternative, you must rebuild the cluster in large subnets so your cluster can meet your applications' demands. You can configure the maximum pods that are deployable to an executor node when you create the cluster or when you create new executor node pools. However, if you do not specify the maximum number of pods for the maxPods property when you create new executor node pools, by default each executor node gets 30 pods (with one IP address per pod).

About using overlay networking

To avoid IP address exhaustion, you can enable the overlay networking feature when you activate an Azure environment to use with Cloudera Data Warehouse. For a full description of CNI Overlay networking in AKS, see the Microsoft documentation.

Important: Third-party overlay networking solutions like Weaveworks Weave Net and Project Calico are not supported in Microsoft Azure. Cloudera strongly recommends that you do not deploy these third-party overlay networking solutions.

Enabling CNI overlay networking

If you are experiencing IP address exhaustion for your Azure environment in Cloudera Data Warehouse Public Cloud, you can enable a CNI overlay network for the environment.

About this task

Required role: EnvironmentAdmin or PowerUser

After you have registered your environment with Cloudera, navigate to the Cloudera Data Warehouse service and perform the following steps to configure your Azure environment to use overlay networking.

Procedure

1. In the Cloudera Data Warehouse, in Environments, search for and locate the environment that you want to configure.
2. Click Activate to activate the environment.
3. In Activation Settings, check Use CNI overlay networking.

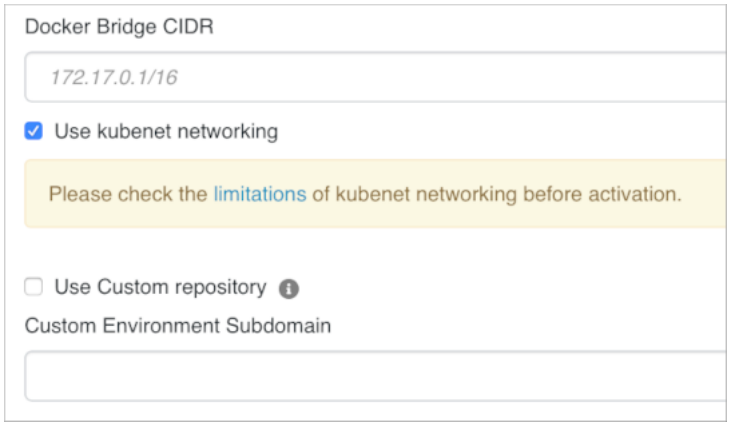

4. Click ACTIVATE.

Using AKS monitoring with Cloudera Data Warehouse in Azure environments

Cloudera Data Warehouseprovides an option to enable and leverage AKS monitoring in Azure environments. When you enable AKS monitoring from the Cloudera Data Warehouse UI and select a Log Analytics workspace, Cloudera Data Warehouse sends the log data to Azure Monitor Logs.

About this task

You can use the Log Analytics tool to edit and run log queries, and create dashboards and alerts from the log queries.

Required role: EnvironmentAdmin or PowerUser

Before you begin

Before you enable AKS monitoring on Cloudera Data Warehouse Public Cloud, you must configure a log analytics workspace in your Azure account. See the link to Microsoft Azure documentation for this task at the bottom of this page.

Procedure

1. In the Cloudera Data Warehouse UI, go to the Environments tab.
2. Locate the environment for which you want to configure the AKS monitoring workspace.
3. Click Activate to launch the Activation Settings dialog box where you can configure the environment to use an AKS monitoring workspace:
4. In the Activation Settings dialog box, select Enable AKS monitoring.
5. Click the drop-down list to select a workspace.
6. Click Activate:

Related Information

Create a Log Analytics workspace in the Azure portal

Deactivating environments on Azure

Learn how to deactivate an Azure environment for Cloudera Data Warehouse Public Cloud.

About this task

Required role: EnvironmentAdmin or PowerUser

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment that you want to deactivate and click Deactivate.

   The **Action** modal is displayed.
3. On the Action modal, you can select environment deactivation options:
   - Choose Drop Data to remove the managed buckets that were created by Cloudera Data Warehouse during environment activation. The underlying data in the data lake is untouched. Only the default Database Catalog

is retained because it resides in the data lake. Non-default Database Catalogs and Virtual Warehouses are deleted.

Important: All metadata and the Ranger policies for non-default Database Catalogs are deleted.

- Choose Force Delete to drop the data and to remove the Database Catalogs and Virtual Warehouses that are associated with the environment.

4. Click OK to deactivate the environment.

Custom tags in Azure environments

Adding tenant-level custom tags help you to search or filter resources the key tag or key value in your Azure account.

Tenant-level tags apply to Cloudera-created resources across your organization's entire cloud provider account. The tenant-level tags can only be applied while creating a new Cloudera environment. The Cloudera environment custom tags propagated to the compute instances within the Cloudera Data Warehouse service. You can use these enant-level tags to identify and monitor the following resources within the Cloudera Data Warehouse service for Azure environments:

- Compute instances in Hive and Impala Virtual Warehouses
- Compute instances in Druid cluster
- Azure virtual machine agent pools

Note: If you are activating a private Cloudera Data Warehouse environment, ensure you do not have more than 15 tags total; otherwise, use a custom DNS zone or public DNS for the AKS cluster. In Azure, the maximum number of tags for the private DNS zones are limited to 15.

Azure load balancers in Cloudera Data Warehouse

Azure provides a public and a private (internal) load balancer to evenly distribute the inbound traffic that arrives at the load balancer's front end to backend pool instances, such as Azure virtual machine scale sets. By default, Azure Kubernetes Service (AKS) cluster uses the Standard SKU for the load balancer.

Depending on your network configuration, you can choose to use the public or the private (internal) load balancer. If you have defined custom user-defined (static) routes (UDR) to route the traffic to a firewall appliance in Azure or on-premises resources, then you can enable the internal load balancer while activating an Azure environment for Cloudera Data Warehouse. Otherwise, Cloudera Data Warehouse uses the Standard public load balancer that is enabled by default when you provision an AKS cluster.

You enable the internal load balancer from the Cloudera Data Warehouse UI when you activate the environment.

Using the public load balancer

When you activate an Azure environment in Cloudera, you provision Azure cloud resources such as the PostgreSQL database server and AKS cluster in your Microsoft Azure subscription. AKS contains a number of Azure infrastructure resources, including virtual machine scale sets, virtual networks, and managed disks. An AKS deployment contains a Standard public load balancer by default, that provides inbound and outbound connections to the cluster executor nodes.

The public load balancer has two public IP addresses to handle inbound and outbound connections. You can access Hue, JDBC, or the Impala shell using the AKS endpoints from the public internet. However, you can restrict access to the endpoints by defining an IP trusted list while or after activating the environment.

Also, the outbound connections towards Cloudera Control Plane and the image registry flows through the public load balancer using the outbound public IP address.

The following diagram shows the ingress and egress traffic when a public load balancer is used:

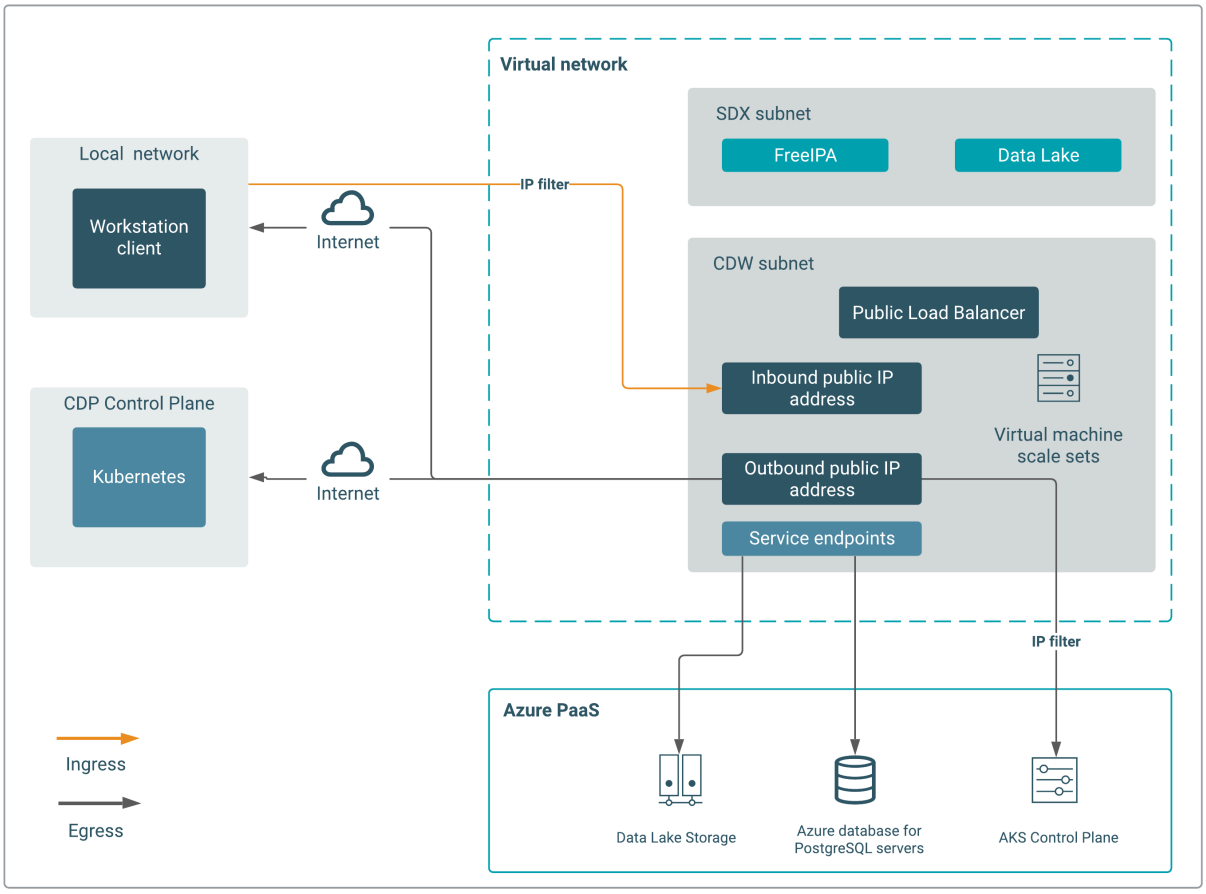

If you have defined custom UDRs to route the traffic to a firewall appliance in Azure or on-premises resources, then you cannot access the load balancer endpoints from the internet. Using a firewall with a UDR breaks the ingress setup due to asymmetric routing. In asymmetric routing, a packet takes one path to the destination and takes another path when returning to the source. The incoming load balancer traffic is received via its public IP address, but the return path goes through the firewall's private IP address. Using an internal load balancer solves the issue with asymmetric routing. The incoming packets arrive at the firewall's public IP address, get translated to the load balancer's private IP address, and then return to the firewall's private IP address using the same return path because the load balancer is deployed with a private frontend IP address.

Using the private (internal) load balancer

Internal load balancers distribute traffic inside a virtual network. Internal load balancers do not have a public IP address and cannot be reached from the public internet, even when you have configured UDRs. An internal load balancer makes a Kubernetes service accessible only to applications running in the same or peered Azure virtual network (VNET) as the Kubernetes cluster, or applications running in on-premises networks that are connected using Azure ExpressRoute circuits.

The following diagram shows ingress and egress traffic when a private (internal) load balancer is used:

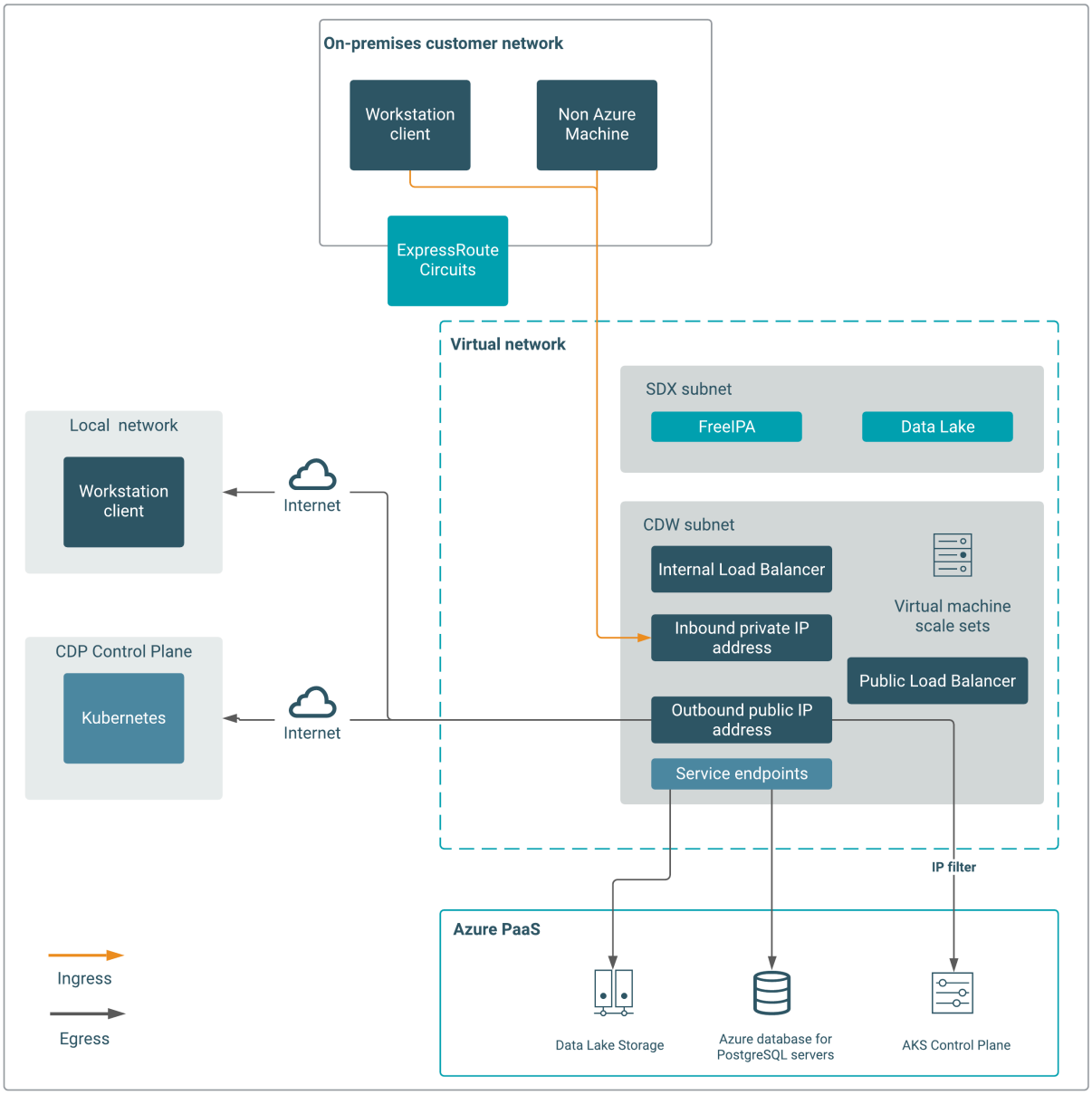

For outbound connections, a public load balancer having a public IP address is used. For inbound connections, an internal load balancer having a private IP address is used. You cannot access the AKS endpoints from the public internet. Even with firewall publishing rules, public requests may fail due to certificate/DNS issues.

Related Information

Enabling internal load balancer

Enabling internal load balancer

To use an internal load balancer for Cloudera Data Warehouse, you must select the option to enable an internal load balancer while activating the Azure environment from the Cloudera Data Warehouse UI. Otherwise, Cloudera Data Warehouse uses the Standard public load balancer that is enabled by default when you provision an AKS cluster.

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment that you want to activate and click Activate.

   The **Activation Settings** modal is displayed.
3. In the **Activation Settings** dialog box, select Enable internal load balancer (ingress).
4. Click Activate.

Granting remote access to Kubernetes clusters on Azure Kubernetes Service

You can remotely access Azure Kubernetes Service (AKS) clusters for troubleshooting, log collection, and maintenance purposes using SSH. To do that, you must add the Azure Active Directory (AD) object ID in the Cloudera Data Warehouse environment Kubeconfig.

About this task

Required Role: DWAdmin

Before you begin

- You must have an active Cloudera Data Warehouse environment to grant your users remote access to the Kubernetes cluster.
- Contact your Azure account administrator to obtain the Azure AD object ID.

Obtaining the Azure AD object ID using Azure CLI

You can obtain the object ID by running the following command:

```
az ad group list --filter "displayname eq '[***GROUP-NAME***]'" -o table
```

This lists all the existing groups within the directory.

Obtaining the Azure AD object ID from the Azure portal

1. Sign in to the Azure portal using a Global administrator account for the directory.
2. Search for and select Azure Active Directory.
3. On the **Active Directory** page, select Groups and click Overview.

4. Note the Object ID from the **Group Overview** page.

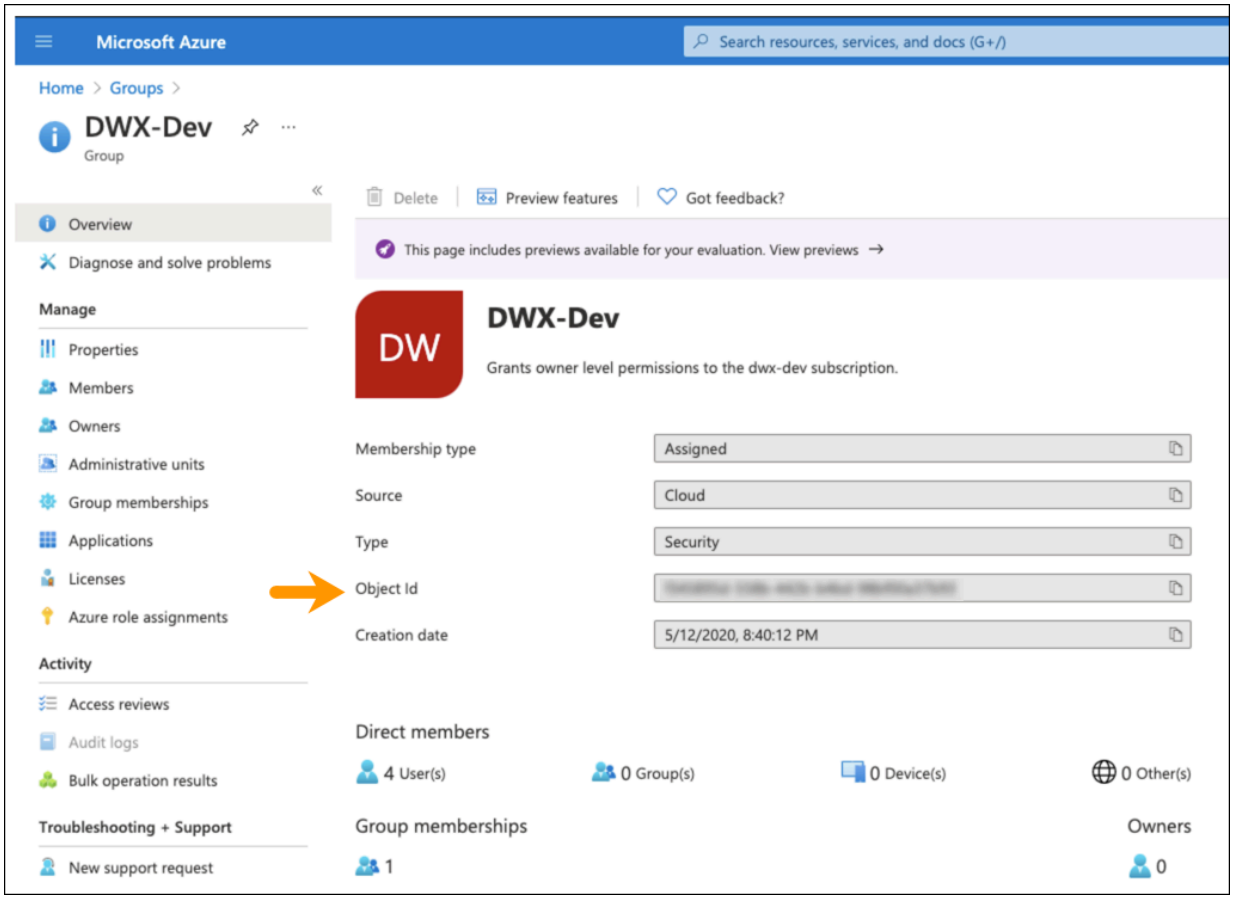

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment for which you want to grant access to AKS and click ⋮ Edit GROUP ACCESS .
3. Enter the Azure AD object ID in the Add new group text box.
4. Click Grant Access to save your changes.

What to do next

Access the Virtual Machines remotely using Azure CLI. When you enter a kubectl command, such as, kubectl get pods Azure CLI, you see the following message:

```
To sign in, use a web browser to open the page https://microsoft.com/
devicelogin and enter the code ****** to authenticate.
```

1. Open a web browser and go to https://microsoft.com/devicelogin.
2. Enter the code on the **Enter code** page and click Next.

3. On the **Pick an account** page, select the account for signing into the AKS AAD Client.

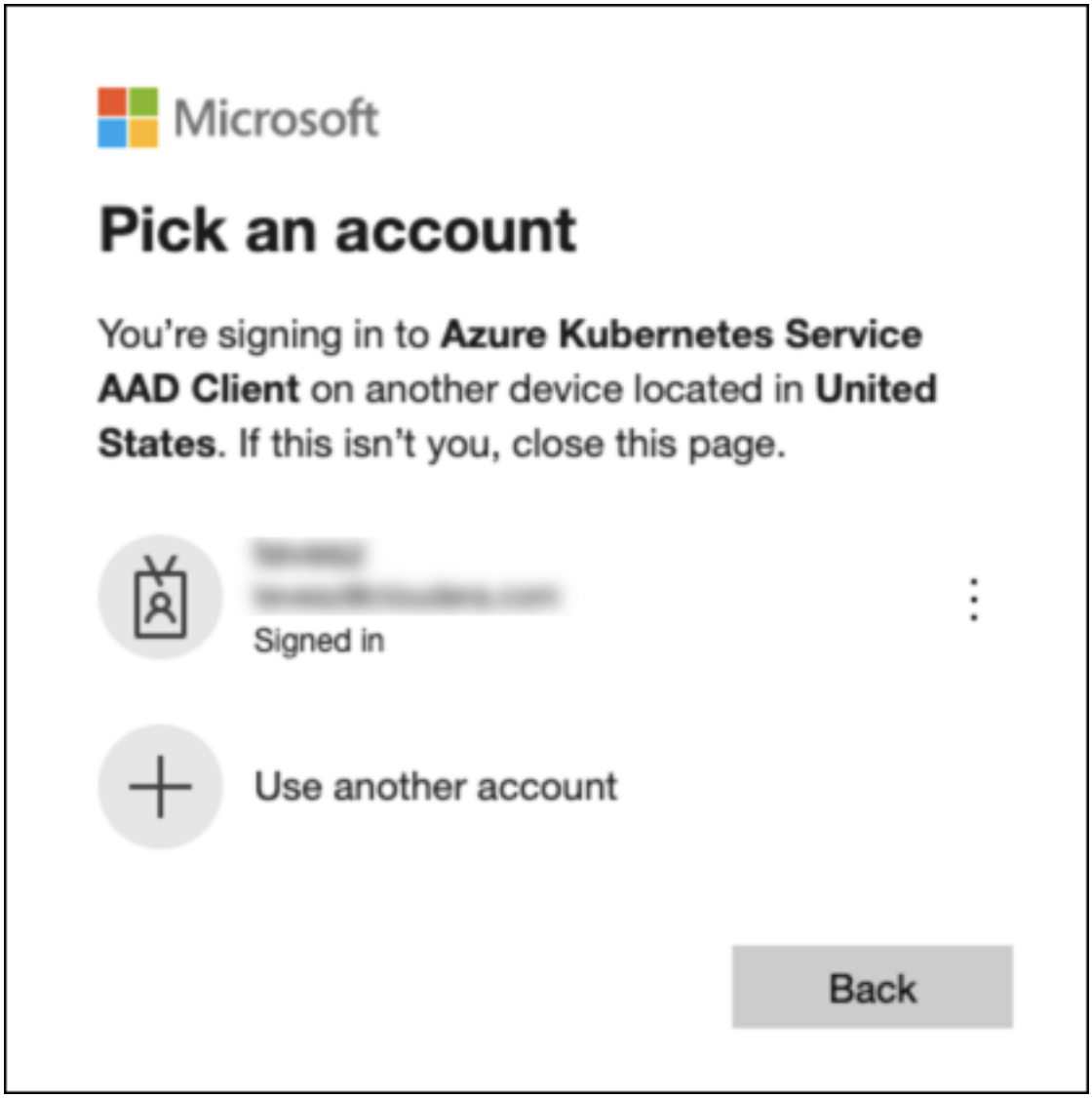

Upon successful login, you get the following message:

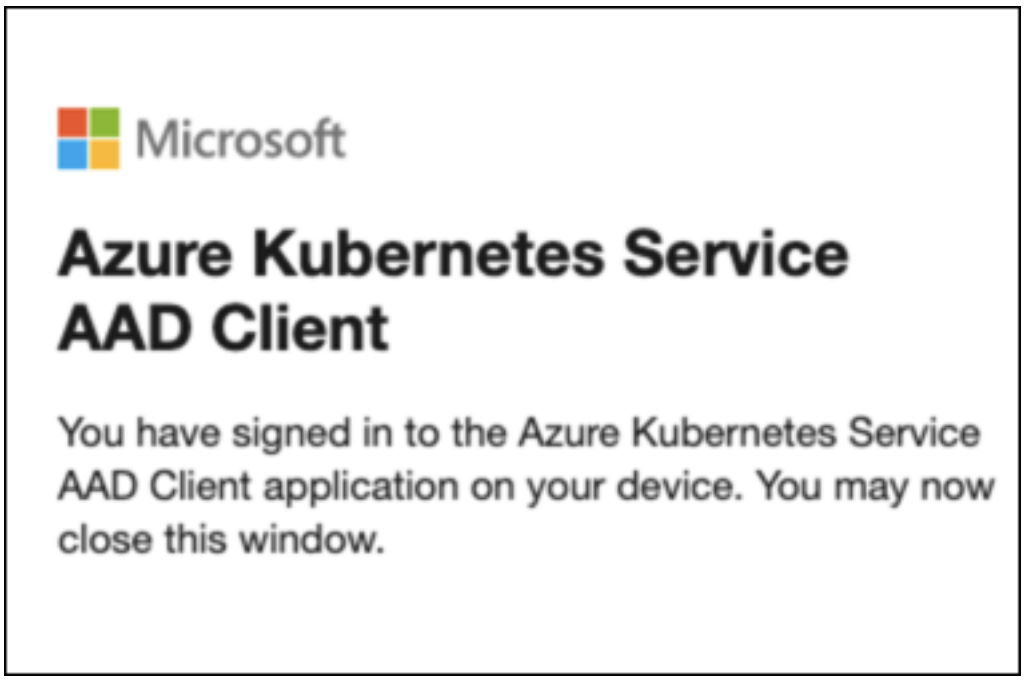

Revoking remote access to Kubernetes clusters on Azure Kubernetes Service

You can revoke remote access to Kubernetes clusters on Azure Kubernetes Service (AKS) by removing the Azure AD object ID from the Cloudera Data Warehouse environment Kubeconfig.

About this task

Required Role: DWAdmin

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment for which you want to revoke access.
3. Click the options menu and select Show Kubeconfig.
4. Click the delete icon under Action to revoke access for the user.
5. Click Hide to close the dialog box.

Managed storage access

Understanding how Cloudera Data Warehouse stores data for multiple tenants and a high-level overview of the configuration tasks prepares you as DWAdmin to set up a RAZ-controlled warehouse.

The multitenant storage technique in Cloudera Data Warehouse requires a separate Database Catalog plus at least one Virtual Warehouse per tenant.

You configure a dedicated tenant storage role to give only the tenant-specific default database catalog instance and its associated virtual warehouse data access to both of the following buckets.

- The tenant storage location
- Potentially shared Storage Location Base

- The SDX Data Lake bucket

If the access token for a role is leaked, the data of others is not compromised; only one tenant's own, and shared, data is vulnerable. A managed identity IDBroker approach to accessing the Virtual Warehouse is shown in the following graphic:

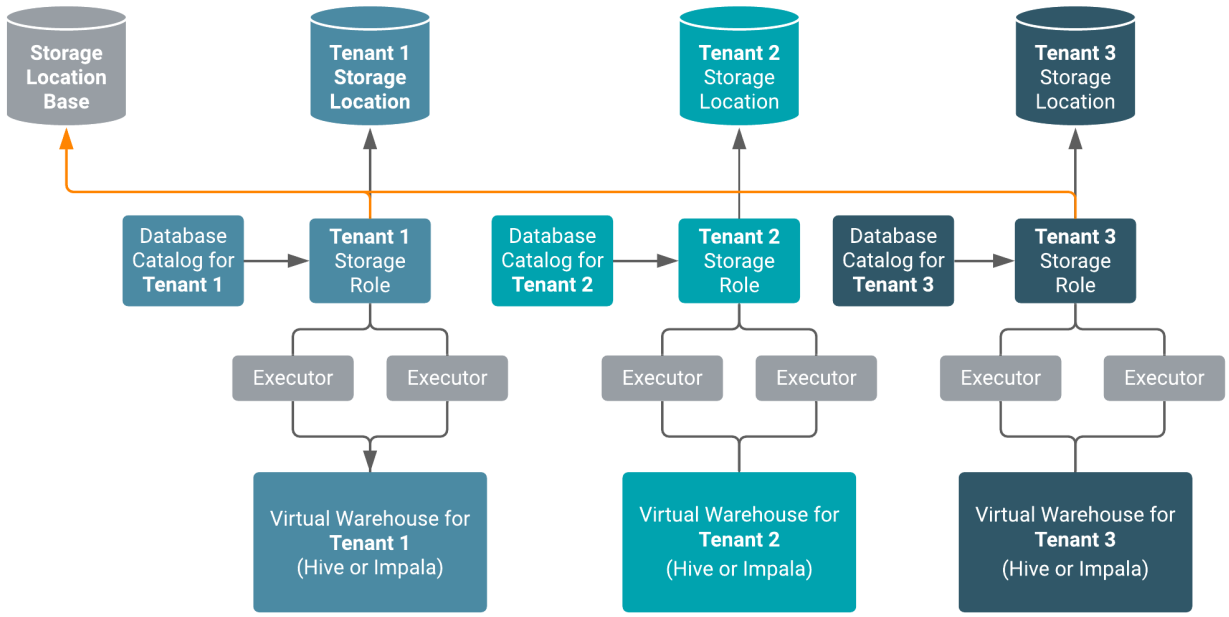

Using a Ranger Remote Authorization (RAZ)-enabled environment, you can control access to data based on user roles and classifications. As a Cloudera Admin, you can apply Ranger fine-grained access control policies to cloud storage.

The following diagram shows the high-level steps for configuring RAZ-enabled storage:

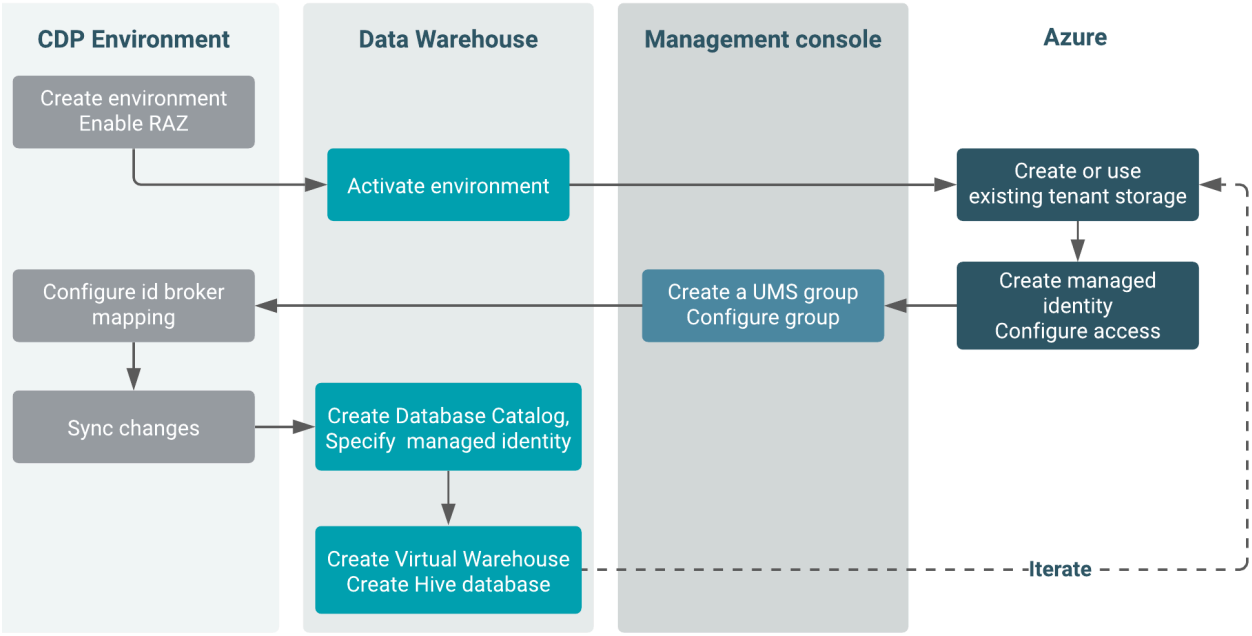

After you obtain the entitlements required for this feature CDW_ALLOW_MULTI_DEFAULT_DBC and CDW_STORAGE_ROLES, you can configure storage as shown in this diagram:

- In Cloudera, register an environment that enables RAZ and uses the SDX Data Lake.
- Activate the environment in Cloudera Data Warehouse.
- Manually create a tenant storage location(s), or use existing ones, for access by the tenant storage role.
- In Azure, create an Azure managed identity for the tenant.
- Assign the tenant specific Azure managed identity as a Storage Blob Data Owner role to the tenant-specific container; assign the tenant specific Azure managed identity as a Storage Blob Data Owner role to Storage Location Base.
- In Cloudera Management Console, create an UMS group for this tenant.
- Add the UMS machine user (created when you activated the Cloudera Data Warehouse environment) to the UMS group.
- Add the id broker mapping to the environment.
- Sync user group changes for the RAZ-enabled environment with FreeIPA.
- Create a separate Database Catalog with a unique managed identity created in the step above.
- Create a tenant-specific Virtual Warehouse based on the Database Catalog, SDX Data Lake, and RAZ-enabled environment.
- Create a tenant-specific Hive or Impala database that points to tenant storage locations.

  As the metadata is shared across all tenants, Ranger grants access to tenant data via a group at the database level.
- For each tenant, repeat the actions above, starting from the step after activating the environment for Cloudera Data Warehouse.

The managed identity accesses a tenant specific location and SDX Data Lake shared Storage Location Base. The Storage Location Base contains shared data, stores the Directed Acyclic Graph (DAG) data used by the Database Catalog, and provides integration with Cloudera Workload XM. WXM writes Impala query data to the shared Storage Location Base.

You need the UMS group to add the IDBroker mapping, as a single UMS machine user cannot have multiple ID broker mappings to different managed identities.

The following topics describe step-by-step how to set up your Database Catalog, and Virtual Warehouse for storing RAZ-enabled data.

Setting up managed storage access

You learn the consequences of setting up RAZ control, which cannot be removed, and requirements you must meet to enable managed storage access.

About this task

The RAZ-controlled warehouse supports only multiple Cloudera Shared Data Experience (SDX) Database Catalogs instead of Cloudera Data Warehouse-specific, isolated database catalogs. In a RAZ-controlled environment, the Elastic Kubernetes Service (EKS) ec2 executor will not have read/write permissions to the storage location. Consequently, after activating the Cloudera Data Warehouse environment, you cannot remove RAZ control. RAZ-control of Cloudera Data Warehouse continues during upgrades of the Database Catalog and Virtual Warehouse.

Before you begin

- Request activation of the following entitlements:
  - CDW_ALLOW_MULTI_DEFAULT_DBC
  - CDW_STORAGE_ROLES
- You meet the requirements described in the Azure requirements documentation.

Required role: PowerUser

Creating the Cloudera environment

You follows steps to register and activate a new environment to enable RAZ in Cloudera Data Warehouse.

Procedure

1. Register an environment with RAZ using the CDP web interface.
   In the web interface, in Fine-grained access control, select Enable Ranger Authorization.

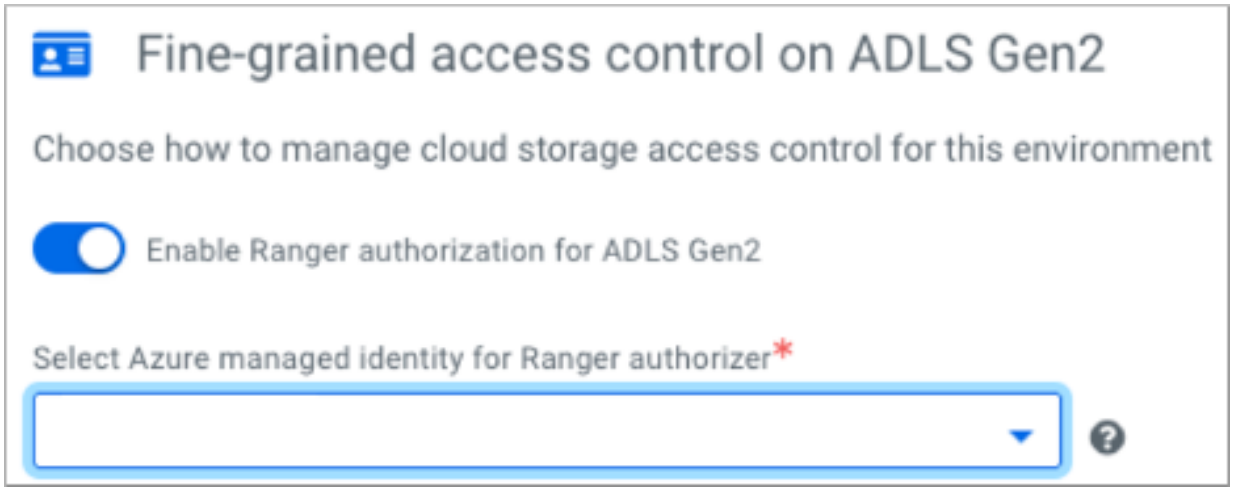

2. In Cloudera Data Warehouse Overview, locate the RAZ-enabled environment, and click Start to activate the environment..
   This action disables the standard default Database Catalog that is automatically created after activation. The UMS machine user is created and attached to the environment when you activate the Cloudera Data Warehouse environment. Later, you see how to add this same UMS machine user to a different UMS group for each tenant.
3. In Activate Environment, select Enable Storage Roles.

   Unchecking Enable Storage Roles disables the metadata proxy associated with managed storage access.

Repeat the following steps for each tenant:

4. In Azure, manually create a one or more storage locations for the tenant, or use existing storage locations.
5. In Azure, create a managed identity that has the role Storage Blob Data Owner to access tenant specific container and across all tenants shared, Storage Location Base in the SDX Data Lake.
   For example, create a managed identity named tenant-1 and a container called container-tenant-1.

Creating a UMS group and machine users

This procedure ensures that the Cloudera machine user gets permission to access the tenant bucket.

Procedure

Repeat the following steps for each tenant:

1. In Cloudera Management Console, User Management Groups , click CREATE GROUP and create a User Management Service (UMS) group, for example group-tenant-1.

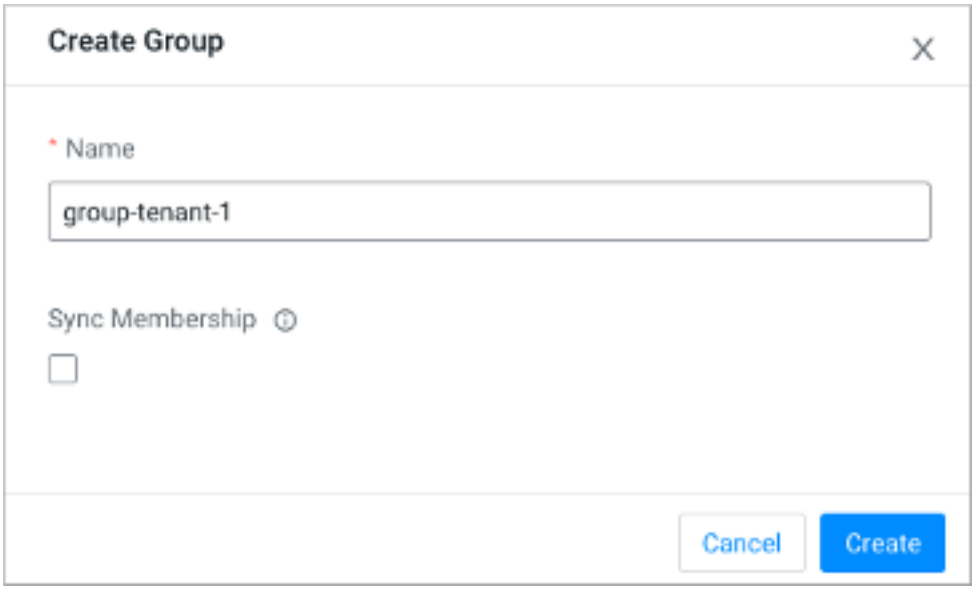

2. In Groups Members , search for and select your srv_machine_<env id>_storage_role to add this UMS machine user to group-tenant-1.

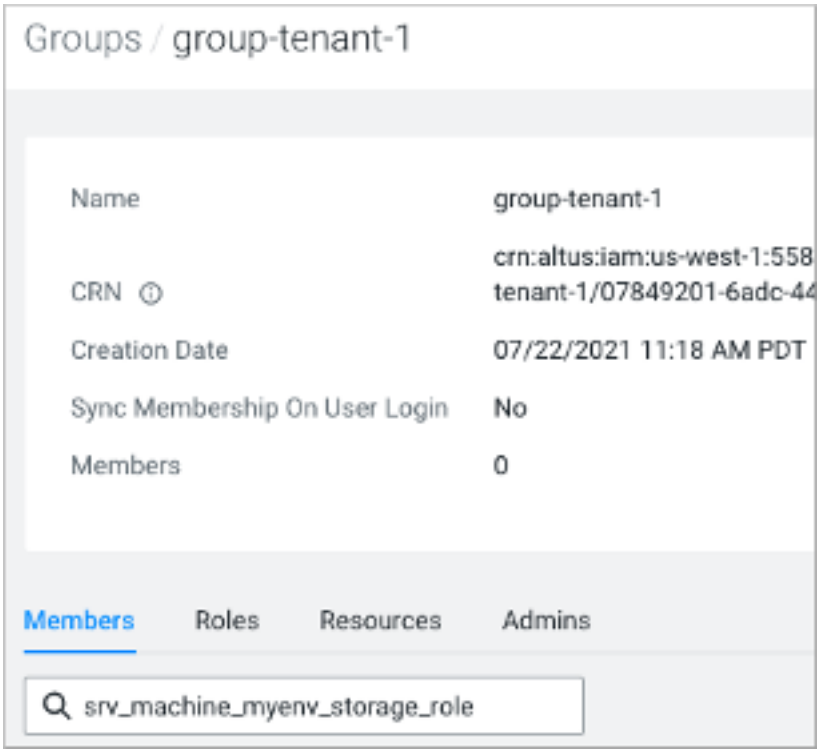

3. In Management Console Environments , select an environment, and click Actions Manage Access IDBroker Mappings Edit .
4. Click + to add a mapping, select the Group-tenant-1 and Role-tenant-1, and specify the complete managed identity. For example: /subscriptions/<subsciption_id>/resourcegroups/<resource_group>providers/Microsoft.ManagedIdentity/userAssignedIdentities/<managed-identity-name>
5. Synchronize your group changes with FreeIPA by performing a user sync per environment: In the RAZ-enabled environment, click Actions Synchronize Users to FreeIPA.
   The UMS machine user gets the permission to access the tenant-specific container.

Creating a new Database Catalog

You repeat a step-by-step procedure to set up a new Database Catalog for each tenant. You must correctly enter your managed identity regular expression. You see how to recognize a discrepancy in your entry and your actual managed identity.

About this task

When you create a new database catalog, you specify the managed identity you created earlier. In the event of an invalid managed identity specification, the following regular expression appears in the UI.

```
 '\/subscriptions\/(.+?)\/resourcegroups\/(.+?)\/providers\/Microsoft.Manage
dIdentity\/userAssignedIdentities\/(.+)', 'i'
```

An example that matches the regular expression is:

```
/subscriptions/<subsciption_id>/resourcegroups/<resource_group>providers/Mic
rosoft.ManagedIdentity/userAssignedIdentities/<managed-identity-name>
```

- subscriptions: Your subscription ID
- providers: Microsoft.ManagedIdentity (required value)
- resourcegroups: Your resource group
- userAssignedIdentities: Your managed identity name.

Procedure

Repeat the following steps for each tenant.

1. Click Data Warehouse Database Catalog Create Database Catalog .
2. In **New Database Catalog**, enter the complete description of the managed identity you created earlier.
3. Select the RAZ-enabled environment.

   In Data Lake, SDX is the required value. The backend Data Lake and Database Catalog database must be the same.

   The backend Data Lake and Database Catalog database must be the same.
4. In Tenant Storage Role, enter the complete managed identity you obtained earlier.
   For example:

   ```
   For example:
   /subscriptions/<subsciption_id>/resourcegroups/<resource_group>providers/
   Microsoft.ManagedIdentity/userAssignedIdentities/<managed-identity-name>
   ```

   Cloudera Data Warehouse attempts to validate your managed identity. If successful, proceed. If validation fails, you see the error message described above. Correct the problem, and try again.
5. In Tenant Storage Location, enter the tenant-specific container container-tenant-1, for example, and click CREATE.

Creating a tenant-specific Virtual Warehouse

You follow a step-by-step procedure to create a Virtual Warehouse based on the Database Catalog and SDX Data Lake you created in the RAZ-enabled environment.

About this task

The Database Catalog and Data Lake point to the same backend database. In the tenant-specific Virtual Warehouse, you create a tenant-specific Hive or Impala database that points to tenant-specific buckets. As the metadata is shared across all tenants, Ranger grants access to tenant data at the table level. One or more tenant-specific databases alongside databases for shared data can run in the same HMS instance.

Procedure

Repeat the following steps for each tenant.

1. In the Cloudera Data Warehouse service, click Virtual Warehouses New Virtual Warehouse .
2. Specify a name, select either the Hive or Impala type, and select the Database Catalog you created for the tenant.
3. In Overview, find your Virtual Warehouse, click Hue.
4. Create a tenant-specific Hive or Impala database where the location for external and managed tables are pointing to the tenant-specific buckets.

   ```
   CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name
   [COMMENT database_comment]
   [LOCATION external_table_path]
   [MANAGEDLOCATION managed_table_directory_path]
   [WITH DBPROPERTIES (property_name=property_value, ...)];
   ```

   Do not set LOCATION and MANAGEDLOCATION to the same path. For more information, see Create a default directory for managed tables.
5. In Ranger, grant the tenant users access to the tenant-specific Hive or Impala database.

Azure Kubernetes Service upgrade

Azure Kubernetes Service (AKS) cluster requires regular updates to the Kubernetes versions. Using the latest AKS version supported by Cloudera avoids compatibility issues between Cloudera Data Warehouse and Azure resources.

Using the Azure CLI or Azure portal to upgrade the AKS cluster is not supported. Cloudera Data Warehouse automatically provisions the latest supported AKS version when you activate an environment in Cloudera Data Warehouse. AKS 1.30 is provisioned automatically when you activate your environment from Cloudera Data Warehouse using release 1.9.3-b166 (released December 5, 2024) and onwards.

To upgrade to the latest AKS, you must deactivate and reactivate your Cloudera Data Warehouse environment. To restore the state of your Cloudera Data Warehouse cluster after reactivation, Cloudera recommends using the backup-restore functionality. For more information, see Backing up and restoring Cloudera Data Warehouse.

Setting scratch space limit for spilling Impala queries in Azure environments

Certain memory-intensive SQL operations write temporary data to disk (known as spilling to disk) when Impala is close to exceeding its memory limit on a particular host. Consequently, a query can complete successfully instead of failing with an out-of-memory error. You must set the scratch space limit carefully because you cannot change the limit later.

About this task

The compute nodes on Azure environments use the STANDARD_E16DS_v4 or STANDARD_E16DS_V5 memory-optimized Virtual Machines by default. These nodes have a temporary SSD storage of 2 * 300 GiB (600 GiB). This space is utilized for both cached data and scratch space. The scratch space available is insufficient for the spilling Impala queries.

Spilling queries to disk require a scratch space in multiples of several 100 GiBs or TiBs, without which the queries may fail. To prevent failure, you can configure the scratch space between 300 GiB and 16 TiB per executor node while creating an Impala Virtual Warehouse. In addition to the local SSD, you can use managed disks (Standard SSD and Premium disks) for extra space.

In Cloudera Data Warehouse, the range of options for scratch space using the combination of the local SSDs and managed disks has been tailored to optimize the cost-to-space ratio. The following table shows the pricing and throughput information for the available scratch space options:

Table 1:

Disks (SSD + managed disk)	Size (GiB)	Throughput (MB/s)	Price ($/hour)
SSD only	300	968 cached / 384 uncached	0
SSD + 3 * P10	684	300 (3 *100)	0.073 (3 * 0.024)
SSD + 2 * P15	812	250 (2 * 125)	0.094 (2 * 0.047)
SSD + 3 * P15	1068	375 (3 * 125)	0.141 (3 * 0.047)
SSD + 2 * P20	1324	300 (2 * 150)	0.182 (2 * 0.091)
SSD + 3 * P20	1836	450 (3 * 150)	0.273 (3 * 0.091)
SSD + P40	2348	250	0.322
SSD + 3 * P30	3372	600 (3 * 200)	0.504 (3 * 0.168)
SSD + P50	4396	250	0.617
SSD + E60	8492	400	0.841
SSD + E70	16684	600	1.683

You can select the scratch space limit only while creating a new Impala Virtual Warehouse.

Before you begin

- Obtain the CDW_IMPALA_EBS_SCRATCH_SPACE entitlement.
- Activate the environment in Cloudera Data Warehouse using the E16ds_v4 or E16ds_v5.
- Disable Availability Zones for AKS during the environment activation; otherwise, you cannot set the scratch space limit.

Procedure

1. Log in to Cloudera Data Warehouse.
2. In the Data Warehouse service, click Virtual Warehouses in the left navigation panel.
3. On the **Virtual Warehouses** page, click New Virtual Warehouse.
4. Specify a Name.
5. Select Virtual Warehouse type as IMPALA.
6. Select the Database Catalog that it queries.
7. Select a Size.

   Additional configuration options are displayed along with Scratch Space Limit per node (in GiBs).
8. Select the scratch space from the Scratch Space Limit per node (in GiBs) dropdown.
9. Click Create Virtual Warehouse.

Related Information

Azure Standard_E16ds_v4 instances
Azure Standard_E16_v3 instances
Adding a new Virtual Warehouse
Activating Azure environments

Cloudera Data Warehouse Public Cloud Backup and Restore

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Backing up and restoring Cloudera Data Warehouse.....4
Cluster URLs after reactivation.....5
Automatic backup and restore.....6
Automatically backing up the environment.....6
Decommissioning the existing environment.....9
Automatically restoring the environment.....10
Monitoring Hue and Data Visualization restoration.....14
Semi-automatic backup and restore.....15
Gathering AWS environment activation parameters.....16
Gathering Azure activation parameters.....17
Gathering observability configurations.....19
Backing up the environment and objects.....19
Monitoring Hue and Data Visualization database backup.....21
Decommissioning the existing environment.....22
Manually reactivating the environment.....23
Modifying configurations after activation.....24
Restoring the environment and objects.....24
Testing the restoration.....27
Monitoring Hue and Data Visualization restoration.....27
Manual backup and restore.....29
Manually backing up the environment.....29
Decommissioning the existing environment.....35
Manually restoring the environment.....36
Monitoring environment restoration.....43

Backing up and restoring Cloudera Data Warehouse

The backup and restore procedures for AWS and Azure replace the in-place upgrade Cloudera offered for AWS environments. To get the supported Kubernetes version, you back up your old AWS or Azure environment and start up a new environment using the restoration process.

Backup and restore does not replace upgrading your Virtual Warehouse. To pick up the latest engine changes in your Virtual Warehouse, simply upgrade your Database Catalog and Virtual Warehouse in Cloudera Data Warehouse. The backup and restore process saves your environment parameters, making it possible to recreate your environment with the same settings, URL, and connection strings you used in your previous environment.

You can back up and restore Cloudera Data Warehouse one of the following methods:

- automatic
- semi-automatic
- manual

Choosing the right method

You choose the automatic, semi-automatic, or manual method based on the following conditions:

Automatic

Use the automatic backup and restore under one of the following conditions:

- Your Cloudera Data Warehouse runs in an AWS environment using the AWS restricted policy.
- Your Cloudera Data Warehouse runs in an Azure environment using Cloudera Data Warehouse 1.6.3-b319 (released May 5, 2023) or later.
- Your private Cloudera Data Warehouse environment in Azure Kubernetes Service is using Cloudera Data Warehouse 1.9.1-b233 (released July 26, 2024) or later.

Semi-automatic backup and restore

Use the semi-automatic procedure under any one of the following conditions:

- Your Cloudera Data Warehouse runs in an AWS environment using reduced permissions mode.
- Your Cloudera Data Warehouse runs in an Azure environment using Cloudera Data Warehouse 1.6.2-b197 (released Feb 13, 2023) or earlier.
- Your private Cloudera Data Warehouse environment in Azure Kubernetes Service is using Cloudera Data Warehouse 1.8.7-b37 (released March 25, 2024) or later.

Manual backup and restore

Use the manual backup and restore under any one of the following conditions:

- You cannot meet conditions for using either the automatic backup and restore or the semi-automatic backup and restore.
- You have enabled the MULTI_DEFAULT_DBC entitlement.
- You have one, or more, custom (non-default) Database Catalogs.
- You experienced problems using the automatic or semi-automatic backup and restore.

Ensuring a successful restoration

To ensure a successful restoration:

- Create a fresh backup when you plan your upgrade.
- Ensure that the backup is of the same Cloudera Data Warehouse Control Plane version with which you plan to restore the files.

For example, if your Cloudera Data Warehouse environment version is 1.8.4-b33, and you wish to upgrade to 1.9.1-b233 (i.e. the Cloudera Data Warehouse Control Plane version is now 1.9.1-b233), then you must create a fresh backup with the Cloudera Data Warehouse Control Plane version of 1.9.1-b233 and use this backup to restore the files after reactivating the environment in Cloudera Data Warehouse. Cloudera recommends that you do not use a backup created with any other/older Cloudera Data Warehouse Control Plane version (1.6.2-b197, for example) during the restoration process.

Check and note the version of your Cloudera Data Warehouse Control Plane (on the bottom left side of the page) before performing these tasks. The Cloudera Data Warehouse Control Plane version is not the same as your environment, Database Catalog, or Virtual Warehouse versions.

Finding the version of your Cloudera Data Warehouse environment

In Cloudera Data Warehouse, select your environment, click Edit. The Environment Details include the version.

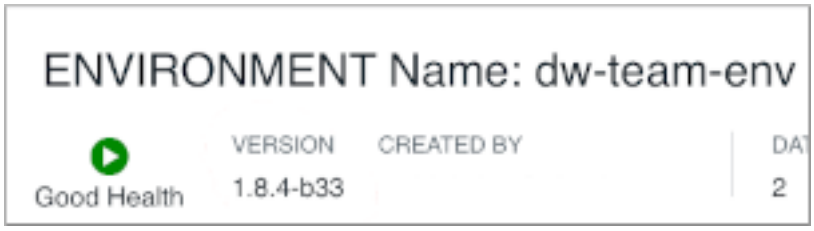

Importance of bringing down the cluster

Backing up and restoring Cloudera Data Warehouse requires bringing down the cluster to ensure successful cluster restoration. During downtime, Cloudera Data Warehouse, you must prevent end-users from accessing the cluster. If downtime is not feasible due to your operational model, you can use a workaround that disables end-user access instead of bringing down the cluster.

You lose any manual modification of the Kubernetes objects or configurations when you bring down the cluster. Modifications applied using the Cloudera Data Warehouse UI and settings defined during creation are preserved.

Cluster URLs after reactivation

It is important to understand the difference between cluster URLs before and after activation.

Current URL format

New Cloudera Data Warehouse environments deploy a URL in a new format that will be preserved even after environment reactivation. The new format is:

```
<vw name>.<dw environment name>.<tenant id>.cloudera.site
```

New format example:

```
Hue-cli-update-vw-config-hive.dw-dwx-qr2j9b.xcu2-8y8x.cloudera.site
```

The new URL consists of the following components:

- Virtual Warehouse name
- Cloudera Data Warehouse environment name
- Static tenant identifier

If this format is already in use, the current URLs will be preserved for both Hue and JDBC.

Old URL format

Old Cloudera Data Warehouse environments use a different format. Old format example:

```
Hue-cli-update-vw-config-hive.env-qwertyu.dw.xcu2-8y8x.cloudera.site
```

The old URL consists of the following components:

- Virtual Warehouse name
- A random generated environment identifier
- A .dw separator
- Tenant ID

URL actions and recommendations

If a new environment is being activated, the old format will change to the new format. It is highly recommended to move to the new format to simplify the environment backup and restore in the future.

This recommendation is because the old format has a dynamic environment id that changes upon reactivation which changes the URL endpoints. Whereas, the new format has a static environment ID that does not change upon reactivation and maintains the URL endpoints. Additionally, Cloudera may deprecate the old format in a future release.

Should there be a need to preserve the old URL format in old Cloudera Data Warehouse environments, a workaround is available and documented as an <OPTIONAL> step in the "Reactivating the environment" section below. To be able to preserve the old URL format, contact support to enable the CDW_CUSTOM_CLUSTER_ID entitlement.

Automatic backup and restore

You use Cloudera Data Warehouse CLI commands in these steps to back up and restore Cloudera Data Warehouse.

Prerequisites

- Cloudera Data Warehouse CLI 0.9.108 or later is installed and configured.
- You have Cluster Administrator privileges and can access the Cloudera Data Warehouse web UI.
- You must use the same Cloudera Data Warehouse version to restore files that you used to back up those files.

Automatically backing up the environment

Backing up the environment consists of capturing environment activation parameters, observability configurations, Virtual Warehouse parameters, the Hue data, and Data Visualization applications.

Some backup procedures automate the process. Some manual backup is also required.

Backing up the environment and objects

You can export environment configurations, which you use later to automatically restore the entire Cloudera Data Warehouse environment and all logical objects, such as Database Catalogs, Virtual Warehouses, and Data Visualization applications. The procedure preserves the data and configurations of the logical objects.

Before you begin

- You must temporarily deploy at least one Virtual Warehouse that runs 2023.0.14.0-15 or later to your environment as described in the steps below if you meet both of the following conditions:
  - You have not deployed Runtime version 2023.0.14.0-15 (released May 5, 2023) or later in any Virtual Warehouse in your cluster.
  - You have deployed only Runtime version 2023.0.13.0-20 (released Feb 7, 2023) or earlier in any Virtual Warehouse in your cluster.

    1. Create a Virtual Warehouse that runs 2023.0.14.0-15 or later.
    2. Delete the Virtual Warehouse you just created.

    The steps above resolve a Hue schema incompatibility issue before backing up and restoring Hue.
- Add bucket encryption to your managed policy and attach the policy to the node instance role.
- You must use the CDP CLI version 0.9.108 or later.

About this task

The procedure below backs up the environment and objects, which includes Virtual Warehouse parameters. Use the CDP CLI `dw backup-cluster` command to create the backup data.

Note: If you originally activated your cluster with the early Cloudera Data Warehouse version 1.1.2 and then used in-place upgrades to upgrade the underlying kubernetes cluster, consider creating a manual backup as well as automatically backing up the environment and objects.

Procedure

Use the CDP CLI `dw backup-cluster` command to create the backup data.

```
export CDP_PROFILE=<test / prod / etc>
export CLUSTER_ID=<the-id-of-the-cluster> # the current ID (original ID) of
 the cluster

cdp \
  --profile ${CDP_PROFILE} \
  dw backup-cluster \
  --cluster-id ${CLUSTER_ID} 1>dump_${CLUSTER_ID}.json
```

Example content of the dump_${CLUSTER_ID}.json file:

```
{
  "clusterId": "env-lqhwqs",
  "operationId": "94197da9-fff7-4414-8b56-a30446c75119",
  "timestamp": "2023-08-16T20:22:00+00:00",
  "data": "UEsDBBQACAAIAAAAAAAAAAAA....
  "md5": "5f427b11f01f5540fa961aba8ea232aa"
}
```

The cluster ID is the unique Cloudera Data Warehouse environment identifier. You can use the operation ID to query the backup execution details using the CLI. The data holds the object data and the configuration,. The md5 is a hash for this data. In case the data and this file is lost, the cluster objects cannot be restored automatically.

What to do next

- Monitor database backup jobs

  The backup will automatically start the Hue backup and Data Visualization database backup jobs that you can monitor. Make sure that the database backup jobs finish before destroying the cluster. If the cluster is deleted before the jobs are finished, you cannot recover the application contents.

- Alert settings

  The compaction observability alert settings are backed up. If the configuration has been modified, make a copy of the configurations, and apply them to the new cluster after restoration.

  Using one of the following ways, get the value of the Alert Manager settings:

  - Use the Cloudera Data Warehouse UI:

    Navigate to your environment tile, click Edit, and in Alert Settings, add the alert settings.
  - Use kubectl:

    kubectl get configmap -n istio-system alertmanager -o json
- Azure environments
  - Public Azure environments activated prior to 1.6.3-b319 (released May 5, 2023) support only manual environment backup. New activations require a managed identity for cluster creation. Old clusters do not have this setting available. Automatic recovery is not an option if your Azure was activated in 1.6.3-b319 (released May 5, 2023).
  - Private Azure environments activated prior to 1.9.1-b233 (released July 26, 2024) support only manual environment backup. In this case, automatic recovery is not an option. New activations require a managed identity and the subnet name of the private SQL for cluster creation. Old clusters do not have these setting available.

    Learn more about private Cloudera Data Warehouse deployment options for Azure.
- Grafana dashboards

  Any changes made to the Grafana dashboards will be lost. A new cluster will be provisioned, the data from the previous cluster won't be carried over to the new Grafana deployment.

Monitoring Hue and Data Visualization database backup

The automatic backup procedure saves the Data Visualization database contents to the configured logs or data folders based on availability.

Hue

During the manual or automatic Hue database backup operation it is critical to block any traffic to the running Hue services. If you cannot bring down the cluster, Cloudera recommends you disable end user access to the cluster endpoints. Failing to do so results in errors in addition to existing key constraints and other issues.

Automatic Hue backup

Automatic backup of Hue extracts the saved query and query history and loads them to the new cluster.

Monitoring Hue backup

The backup starts a job to load the database dump file, but does not wait for the job to complete. If you have a large database, the job can take up to an hour to complete. Ensure you allow enough time for the job to succeed.

To monitor Hue backup, log into the cluster and monitor the job status under the database catalog namespace.

```
$ kubectl get jobs -n <database catalog id>
```

The output that shows the hue-backup job looks something like this:

```
$ kubectl get jobs -n warehouse-1692037411-96hk
  NAME                                                COMPLETIONS   DURATION
 AGE
  hue-backup-ede2b8bd-1d53-4d23-a0f9-87d8ec658f74  1/1           11s
 113s
  hue-query-processor-db-create-job                 1/1           8s
 42h
```

Data Visualization

The automatic backup procedure saves the Data Visualization database contents to the configured logs or data folders based on availability.

Automatic backup

Automatic backup of Data Visualization extracts the dashboards, tables and connections. Make sure to wait for the job to finish before destroying the cluster.

Monitoring backup of Data Visualization

The backup starts a job to create the database dump file, but it does not wait for it to complete. In case your database size is large, it can take up to 20 minutes for the job to complete. Make sure to leave enough time for the job to succeed. To monitor Data Visualization backup, you can log into the cluster and see the job status under the viz namespace using the following command to extracts the dashboards, tables and connections:

```
$ kubectl get jobs -n <data visualization id>
```

The output looks something like this:

```
$ kubectl get jobs -n viz-1692216942-fc2g
  NAME                                                COMPLETIONS   DURATION
 AGE
  viz-backup-d874515a-be7e-4902-ac75-269c14f9580c   1/1           3m3s
  10m
  viz-webapp-vizdb-create-job                         1/1           57s
 99m
```

Decommissioning the existing environment

You follow procedures to first delete the Virtual Warehouses in the Cloudera Data Warehouse Environment and the Data Visualization resources (visuals). Next, you deactivate the environment.

The topic "Delete Data Visualization visuals" describes the resources that are deleted when you delete a Virtual Warehouse as described in the procedure below.

Deleting the Virtual Warehouses

You must delete the Virtual Warehouses in the Cloudera Data Warehouse Environment and the Data Visualization resources (visuals).

Procedure

1. Log in to the Cloudera web interface, navigate to Data Warehouse Overview , note the name of the Virtual Warehouse you want to modify, and note which Database Catalog it is configured to access.

2. In the Virtual Warehouse you want to delete, click SUSPEND to stop running the Virtual Warehouse.

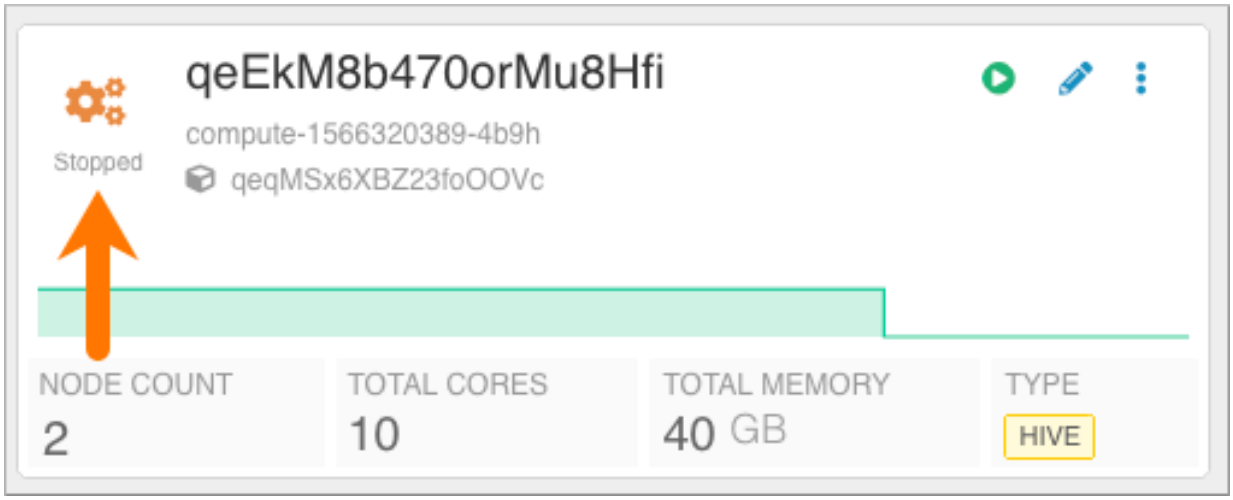

3. Click ⋮ Delete corresponding to the Virtual Warehouse you want to delete.

   Data Visualization visuals are deleted as described in "Delete Data Visualization visuals" when you delete a Virtual Warehouse.

Deactivating the environment

Assuming you have already deleted your Virtual Warehouses, you need to delete your non-default (custom) Database Catalog and Virtual Warehouses, and then deactivate the environment.

About this task

You cannot delete default Database Catalogs created during environment activation. Default catalogs are deleted when the environment is deactivated.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Database Catalogs tab, locate your Database Catalog, and click ⋮ Delete .
3. Go to the **Overview** page and click on the Environments tab.
4. Locate the environment that you want to deactivate and click Deactivate.

Automatically restoring the environment

You can automatically reactivate the entire environment using the CLI, which includes your cluster. Automatic restoration enables all settings of the environment that you backed up.

An environment-level automated restoration CLI option restores the environment, the deployed Database Catalog, Virtual Warehouse, and Data Visualization entities. If a Virtual Warehouse or a Data Visualization object is not present on the cluster, but the backup file contains it, the Virtual Warehouse or Data Visualization object will be restored to the cluster. If such an entity is already deployed, no changes or configuration updates will take place.

The CLI `dw restore-cluster` command can be used in the following ways:

- Passing the environment's Cloudera resource name (crn) will activate the cluster from the backup file and restore all the entities and database contents.
- Passing an activated environment identifier will restore all the entities and database contents to the running environment. This is useful when you need to change activation parameters, but requires manual reactivation.

Automatic restoration consists of the following operations in the order shown here:

- Activates the environment and waits for infrastructure creation
- Applies the cluster services and sets up the environment

- Creates the default Database Catalog
- Updates the Database Catalog configuration to apply customer configuration customizations
- Starts the Hue database restore job in the database catalog namespace asynchronously
- Deploys the Virtual Warehouse instances
- Deploys the Data Visualization instances
- Starts the Data Visualization restore job in the individual namespaces asynchronously

Before you begin the restoration of Hue, if you cannot bring down the cluster, use the recommended workaround to disable end user access to the cluster endpoints. The automatic restoration process does not wait for the database operations to be finished. You must monitor the status of the jobs using the operation id to make sure the process finishes. For more information, see Monitoring Hue and Data Visualization restoration and Monitoring environment restoration.

Details about the restore process

The restore process is designed to be an idempotent process, it can be restarted as many times as you want. If the environment is activated and healthy, you can run the restore operation multiple times to restore the Virtual Warehouse and Data Visualization objects. For every restore operation, the Hue database restore will run. This operation will overwrite the Hue database contents. If a Virtual Warehouse or a Data Visualization object is not present on the cluster, but the backup file contains it, it will be restored to the cluster. In case such an entity is already deployed, no changes or configuration updates will take place.

Restoring the environment and objects

You learn how to use the `dw restore-cluster` command, which you can use either to pass the environment's Cloudera resource name (crn) or to pass the identifier of an activated environment.

About this task

Passing the Cloudera resource name (crn) will activate the cluster from the backup file and restore all the entities and database contents.

Passing an activated environment resource name will restore all the entities and database contents to the running environment. Passing the environment identifier is useful when you need to change activation parameters, but requires manual reactivation.

In the steps below, use `dw restore-cluster` to pass the Cloudera resource name (crn) to activate the cluster.

Before you begin

- You have a public Azure cluster and your Azure cluster must run version 1.6.3-b319 (released May 5, 2023) or later.

  You cannot automatically activate a public Azure cluster that runs version 1.6.2-b197 (released Feb 13, 2023) or earlier.

- You have a private Azure cluster and your Azure cluster must run version 1.9.1-b233 (released July 26, 2024) or later.

  You cannot automatically activate a private Azure cluster that runs version 1.8.7-b37 (released March 26, 2024) or earlier.

  Learn more about private Cloudera Data Warehouse deployment options for Azure.

- You must use the same Cloudera Data Warehouse version to restore files that you used to back up those files.

  Using a backup file from older than the current version will not work.

Procedure

1. Get your environment resource name from the Cloudera portal by selecting the environment that is not activated, and clicking Manage.
   The environment properties open.

   Under the environment resource name the Cloudera resource name (crn) appears.

```
crn:cdp:environments:us-west-1:98765432-abcd-45d7-b645-7ccf9edbb73d:envi
ronment:00000000-7bf2-4aeb-af71-f2bf2c038588
```

2. Create a CLI skeleton file to serve the base file for the restore command.
   For example, replace your environment resource name placeholder <your cluster name> with the environment resource name of the newly activated cluster (for example env-npk886 shown step 3 of Reactivating the environment).

```
export CLUSTER_NAME="<your cluster name>"
cdp \
  dw restore-cluster \
  --generate-cli-skeleton 1>restore_${CLUSTER_NAME}_cli_input.json
```

3. Open restore_<CLUSTER_NAME>_cli_input.json for editing, and fill in the clusterId and the data fields.
   For example:

```
{
   "clusterId": "crn:cdp:environments:us-west-1:98765432-abcd-45d7-b645-7c
cf9edbb73d:environment:00000000-7bf2-4aeb-af71-f2bf2c038588",
   "data": "UEsDBBQ…AAAAAAAAAAABkYXRhUEsFBgAAAAABAAEAMgAAAKuBAQAAAA=="
}
```

4. Use the `dw restore-cluster` command, provide the same CLUSTER_NAME as you used in step 3 and use the CDP_PROFILE from your CLI configuration.

```
export CLUSTER_NAME="<your cluster name>"
export CDP_PROFILE="<your CDP CLI profile>"

cdp \
  --profile ${CDP_PROFILE} \
  dw restore-cluster \
    --cli-input-json file://restore_${CLUSTER_NAME}_cli_input.json
```

   Example output:

```
{
    "clusterId": "crn:cdp:environments:us-west-1:98765432-abcd-45d7-b645-7
ccf9edbb73d:environment:00000000-7bf2-4aeb-af71-f2bf2c038588",
    "operationId": "62408134-3d8c-46e8-a914-0f427fc3b1b1",
    "action": "Create",
    "message": "the cluster will be created",
    "dbcRestorePlans": [
        {
            "ref": "test-aws-dl-default",
            "id": "warehouse-1692719478-xrm4",
```

```
            "action": "Create",
            "message": "the SDX-type DB Catalog will be created based on
 the data referenced in the backup as test-aws-dl-default"
        }
    ],
    "hueRestorePlans": [
        {
            "ref": "test-aws-dl-default",
            "id": "warehouse-1692719478-xrm4",
            "action": "LoadOrOverwrite",
            "message": "Hue restore is started for warehouse-1692719478
-xrm4 DB Catalog, referenced in the backup data as test-aws-dl-default.
Restore will overwrite Hue database with the backup if it isn't empty."
        }
    ],
    "hiveRestorePlans": [
        {
            "ref": "test-hive",
            "action": "Create",
            "message": "the test-hive Hive Virtual Warehouse will be cr
eated and attached to the warehouse-1692719478-xrm4 DB Catalog"
        }
    ],
    "impalaRestorePlans": [
        {
            "ref": "test-impala",
            "action": "Create",
            "message": "the test-impala Impala Virtual Warehouse will be
 created and attached to the warehouse-1692719478-xrm4 DB Catalog"
        }
    ],
    "vizRestorePlans": [
        {
            "ref": "test-viz",
            "action": "Create",
            "message": "the test-viz Data Visualization will be created"
        }
    ]
    }
```

After several minutes the environment will be activated, the Virtual Warehouses will be created in the new cluster and attached to the Database Catalog. The Virtual Warehouse and Data Visualization ids will be changed.

The Data Visualization database will be recovered. However, because this is a new deployment, the recovered connections will be broken.

5. Monitor the environment restoration as described in Monitoring environment restoration.
6. Adjust the Data Visualization connection settings to point to the new Virtual Warehouse(s).

Testing the restoration

The environment and entity automated restore process can deploy objects to any given environment. You can validate all entities and their settings to gain confidence that the restore operation will succeed for the production environment.

If the data lake storage path for the restored environment if different from than the data lake storage path for the backed up environment, the database restore jobs will fail. The jobs will not be able to reach the database backup file paths; this is expected. The restoration will report the failure, but all entity deployment occurs normally. Customers with older environments might want to consider either testing the restore process first or making a manual backup of the cluster and its properties.

To test the restore process, define a test data lake environment crn in the restore file and follow the details from the "Automatically restoring the environment" page.

Monitoring Hue and Data Visualization restoration

The restore process is designed to be an idempotent process, it can be restarted as many times as you want. In case the environment is activated and healthy, the restore operation can be run multiple times to restore the Virtual Warehouse and Data Visualization objects.

Hue automatic restoration

Restoration of the Hue database occurs only if no Virtual Warehouses arevattached to the particular Database Catalog. In the CDP CLI response to "hueRestorePlans" section of the dw restore-cluster command is the "LoadOrOverwrite" action. This action denotes that the Hue restore operation will run. In any other case the response will be "Skip".

To retry a failed Hue restore operation using the CDP CLI dw restore-cluster command, you must first discommission Virtual Warehouses.

Cloudera can only ensure the data-correctness if there are no running Virtual Warehouses attached to the Database catalog. During the automated restore process the creation of the Virtual Warehouses is deferred until the Hue restore completes. Depending on the size of the Hue backup, this process can take up to 30 minutes. During this time the Cloudera Data Warehouse UI indicates that the Environment and the Database Catalog is created, but no Virtual Warehouses appear in the UI.

Learn more about how to monitor the Hue restoration process in the following section.

Monitoring Hue restoration

The restoration starts a job to load the database dump file, but does not wait for the job to complete. If you have a large database, the job can take up to an hour to complete. Ensure you allow enough time for the job to succeed.

To monitor Hue restoration, log into the cluster and monitor the job status under the database catalog namespace.

```
$ kubectl get jobs -n <database catalog id>
```

The output that shows the hue-restore job looks something like this:

```
$ kubectl get jobs -n warehouse-1692037411-96hk
     NAME                                                COMPLETIONS   DURATIO
N   AGE
     hue-restore-ede2b8bd-1d53-4d23-a0f9-87d8ec658f74   1/1           11s
     113s
     hue-query-processor-db-create-job                  1/1           8s
    42h
```

You can monitor the Hue restoration process using the CDP CLI dw list-events --operation-id <operation-id> command. Use the operation id acquired with the dw restore-cluster command to see events belonging to the long running restore process.

Monitor the "HueRestoreWait" events. For example:

```
{
            "operationId": "5cd90dab-5b15-4de9-b70f-ddab0dfa0c10",
            "event": "HueRestoreWait",
            "message": "{\"type\":\"info\",\"message\":\"Waiting Hue data re
store job to finish\",\"error\":null}",
            "timestamp": "2024-03-21T09:15:11+00:00"
        },
        {
            "operationId": "5cd90dab-5b15-4de9-b70f-ddab0dfa0c10",
            "event": "HueRestoreWait",
            "message": "{\"type\":\"info\",\"message\":\"Hue Data restore st
ill processing: job status, active: 1, failed: 0, succeeded: 0\",\"error\":n
ull}",
```

```
            "timestamp": "2024-03-21T09:15:26+00:00"
        },
```

As Hue is restored, the following HueRestoreWait message will appear: "active: 1, failed: 0, succeeded: 0"

Upon the restoration of Hue, the following message appears: "Hue data restore job finished"

Data Visualization automatic restoration

If a Data Visualization object is not present on the cluster, but the backup file contains it, it will be restored to the cluster. In case such an entity is already deployed, no changes or configuration updates will take place.

Automatic restoration of Data Visualization loads the dashboards, tables, and connections to the new applications. Make sure to wait for the job to finish before destroying the cluster.

To monitor restoration of Data Visualization, you can log into the cluster and see the job status under the viz namespace using the following command.

```
$ kubectl get jobs -n <data visualization id>
```

The output will be similar to this, the viz-restore job shows the status.

```
$ kubectl get jobs -n viz-1692216942-fc2g
     NAME                                                COMPLETIONS   DURAT
ION   AGE
     viz-restore-d874515a-be7e-4902-ac75-269c14f9580c   1/1           3m3s
     10m
     viz-webapp-vizdb-create-job                        1/1           57s
      99m
```

The job logs contain the upload path where the backup file has been downloaded from.

Automatic restoration of Data Visualization

Automatic backup and restore for Data Visualization extracts the dashboards, tables and connections. Make sure to wait for the job to finish before destroying the cluster. In the event of a restoration failure, try manually restoring Data Visualization.

Monitoring Data Visualization restoration

To monitor the restoration of Data Visualization, you can log into the cluster and see the job status under the viz namespace using the following command.

```
$ kubectl get jobs -n <data visualization id>
```

The output looks something like this:

```
$ kubectl get jobs -n viz-1692216942-fc2g
  NAME                                                COMPLETIONS   DURATION
 AGE
  viz-restore-d874515a-be7e-4902-ac75-269c14f9580c   1/1           3m3s
   10m
  viz-webapp-vizdb-create-job                        1/1           57s
 99m
```

Semi-automatic backup and restore

You use CDP CLI commands and manual reactivation in these steps.

Prerequisites

- CDP CLI 0.9.108 or later is installed and configured.
- You have Cluster Administrator privileges and can access the Cloudera Data Warehouse web UI.
- You must use the same Cloudera Data Warehouse version to restore files that you used to back up those files.

Gathering AWS environment activation parameters

You back up AWS environment activation parameters using the Cloudera Data Warehouse UI , AWS CLI, and kubectl.

About this task

AWS environment activation settings that need to be available in the new environment may include IP-CIDRs for the Kubernetes cluster and load balancer, the deployment mode setting, reduced permissions mode, and overlay networks. It is required that you gather and document these settings to have the environment behave the same after the back-up/restore process as before. The activation parameter values are available in the Cloudera Data Warehouse UI. Additionally, some parameters may be fetched using AWS CLI and kubectl.

The following steps gather and document the environment parameters.

Procedure

Deployment Mode (Network selection)

1. Get subnet information in one of the following ways:
   - Use the Cloudera Data Warehouse UI

     Navigate to Environment Details General Details , and note the settings for **Public Subnets** or **Private Subnets**.
   - Use the AWS CLI

     ```
     aws cloudformation describe-stacks --stack-name env-q4tzxd-dwx-stack --o
     utput json --query "Stacks[].{StackName:StackName, PublicSubnetIds:Outpu
     ts[?OutputKey=='PublicSubnetIds'].OutputValue, PrivateSubnetIds:Outputs[
     ?OutputKey=='PrivateSubnetIds'].OutputValue}"
     ```

IP-CIDR for kubernetes cluster and load balancer

2. In the Cloudera Data Warehouse UI, navigate to Environment Details Configurations , and note the settings of Enable IP-CIDR for Kubernetes cluster and Enable IP-CIDR for the load balancer.

Overprovision nodes

3. Document the value of Overprovision nodes.

   Stored behind the CDW_CLUSTER_OVERPROVISIONER entitlement.

Use Custom ECR repository

4. Document the value of Use Custom ECR repository.

   Stored behind the CDP_CUSTOM_REPO entitlement.

Use Overlay Network

5. Using kubectl, get the value of Use Overlay Network.

   ```
   kubectl get daemonsets -n kube-system
   ```

   Note: If the daemonSet aws-node is not present, the overlay network is enabled.

Attach Managed policy ARN to Node Role

6. Using the AWS CLI, fetch the nodeInstanceRole.

```
export nodeInstanceRole=$(aws iam list-roles --query "Roles[].{RoleName:
RoleName}" | grep "env-q4tzxd-dwx-stack-NodeInstanceRole-*" | tr -d '"'|
 cut -f 2 -d ':'| awk '{$1=$1};1')
```

7. List the policies attached to this nodeInstanceRole.

```
aws iam list-role-policies --role-name $nodeInstanceRole
```

8. Check for customized policies, which are not included in the following policies:

```
"PolicyNames": [
  "cluster-autoscaler",
  "dynamodb",
  "ebs",
  "efs",
  "kms",
  "limits-monitoring",
  "s3-list-all-buckets",
  "s3-read-only-buckets",
  "s3-read-write-own-buckets"
]
```

AMI ID

9. Using one of the following ways, get the value of the AMI ID only if you need to use a custom AMI:
   - Use the Cloudera Data Warehouse UI:

     Navigate to Environment Details Configurations , and note the setting of AMI ID.
   - Use AWS CLI:

     ```
     aws cloudformation describe-stacks --stack-name env-q4tzxd-dwx-stack --o
     utput json --query "Stacks[].{StackName:StackName, AMI:Parameters[?Param
     eterKey=='EksAmi'].ParameterValue}"
     ```

Reduced Permissions Mode

10. Using the Cloudera Data Warehouse UI, determine whether or not your current policy has standard activation permission.

    If the current policy does not have the standard activation permission, when you reactivate the environment you see a prompt to use the Reduced Permissions Mode.

    In Reduced Permissions Mode, you deploy the cluster manually. A Cloudformation template is generated and the you must deploy the cluster and apply some role-based access control roles to the Kubernetes cluster.

Enable CloudWatch logs

11. Using the Cloudera Data Warehouse UI, navigate to Environment Details Configurations Enable CloudWatch Logs to get the value of Enable CloudWatch logs..

    CloudWatch logs provide better visibility into cluster operations in addition to the diagnostic bundles within Cloudera Data Warehouse. Enabling this option will not impact the restore procedure, even if it was not previously configured in the old environment.

Additional External Buckets

12. Using the Cloudera Data Warehouse UI, gather values of the Additional External Buckets parameter. Navigate to Environment Details Configurations Bucket Name .

Gathering Azure activation parameters

You configure almost all (99%) of Azure cloud resources using environment activation parameters. These parameters are available by querying Azure resource providers in the old environment. You use these parameters, which you manually document, during the activation of the new environment.

About this task

The Azure environment activation settings you want to carry over to the new environment include the compute VM size (E16ds_v4 or E16_v3), any user-assigned, managed identity, subnets, private Cloudera Data Warehouse and IP CIDRS, overlay networking, and minimum permissions.

Procedure

1. Obtain a managed identity for Azure activations.

   The new, required managed identity parameter provides privileges to deploy the AKS cluster. For more information about required minimal privileges, see "Setting up minimum permissions".

2. Query the environment to get the Azure environment activation settings you want to carry over to the new environment.
   Query the environment to get the activation parameters.

   ```
   az aks show -n <AKS_CLUSTER_NAME>  -g <CDW_RESOURCE_GROUP> --query '{Age
   ntpools:agentPoolProfiles[].{Name:name, Version:orchestratorVersion, Sta
   te:provisioningState, AZ:availabilityZones, SKU:vmSize, VnetSubnet:vnetS
   ubnetId, PodSubnet:podSubnetId, CDW_Timestamp:tags.timestamp, PowerState
   : powerState}, Api:apiServerAccessProfile, NetworkType:networkProfile.ne
   tworkPluginMode, DockerCIDR:networkProfile.dockerBridgeCidr, outboundTyp
   e:networkProfile.outboundType, privateFQDN:privateFqdn, Identity:identity,
    FQDN:fqdn, AKSVersion:kubernetesVersion, Location:location, SKU: sku, OMS
   : addonProfiles.omsagent}' -o jsonc
   ```

   The query output maps to the following activation parameters:

   - Compute VM Size: Agentpoolc.SKU
   - Subnet: Agentpoolc.VnetSubnet
   - Private Cloudera Data Warehouse: api.enablePrivateCluster
   - Managed identity: Identity
   - Availability Zones: agentpoolss.AZ
   - AKS Monitoring: oms
   - K8s CIDR: api.authorizedIpRanges
   - Overlay networking: NetworkType
   - Docker CIDR: dockerCidr
   - AKS DNS Zone: api.privateDnsZone
   - OutboundType: outboundType

3. Get the internal load balancer settings in one of the following ways.

   ```
   az resource list -g  MC_<AKS_CLUSTER_NAME>_<REGION> --query "[?type == '
   Microsoft.Network/loadBalancers'].{Name: name, Type: type}" -o jsonc
   ```

   If the output lists an internal load balancer, the environment has been activated with the **Enable internal load balancers** option.

4. Get the PostgreSQL server setting using Azure CLI.

   ```
   az postgres flexible-server show -g <CDW_RESOURCE_GROUP> -n <SQL_SERVER_
   NAME> --query '{Name:name, FQDN: fullyQualifiedDomainName, Version: vers
   ion, Network:network}' -o jsonc
   ```

   Note the following activation parameters:

   - Private-sql-subnet-name: delegatedSubnetResourceId
   - Private-dns-zone-sql: privateDnsZoneArmResourceId

Gathering observability configurations

Environment configurations are used to monitor and observe an environment.

About this task

If you did not make changes to the environment for Observability or the Alert Manager, skip this step.

Procedure

Alert Manager setting

1. Using one of the following ways, get the value of the Alert Manager settings:
   - Use the Cloudera Data Warehouse UI:

     In Environments, search for and find your environment tile, and in Actions, click ⋮ and select Edit Alert Settings , add the alert settings.
   - Use kubectl:

     ```
     kubectl get configmap -n istio-system alertmanager -o json
     ```

Observability configurations

2. Using the Cloudera Data Warehouse UI, in Environments, search for and find your environment tile, and click ⋮ and select Edit Observability , and copy the configurations. Observability::json > ObservabilityConfig>

Backing up the environment and objects

You can export environment configurations, which you use later to automatically restore the entire Cloudera Data Warehouse environment and all logical objects, such as Database Catalogs, Virtual Warehouses, and Data Visualization applications. The procedure preserves the data and configurations of the logical objects.

Before you begin

- You must temporarily deploy at least one Virtual Warehouse that runs 2023.0.14.0-15 or later to your environment as described in the steps below if you meet both of the following conditions:
  - You have not deployed Runtime version 2023.0.14.0-15 (released May 5, 2023) or later in any Virtual Warehouse in your cluster.
  - You have deployed only Runtime version 2023.0.13.0-20 (released Feb 7, 2023) or earlier in any Virtual Warehouse in your cluster.

    1. Create a Virtual Warehouse that runs 2023.0.14.0-15 or later.
    2. Delete the Virtual Warehouse you just created.

    The steps above resolve a Hue schema incompatibility issue before backing up and restoring Hue.
- Add bucket encryption to your managed policy and attach the policy to the node instance role.
- You must use the CDP CLI version 0.9.108 or later.

About this task

The procedure below backs up the environment and objects, which includes Virtual Warehouse parameters. Use the CDP CLI `dw backup-cluster` command to create the backup data.

Note: If you originally activated your cluster with the early Cloudera Data Warehouse version 1.1.2 and then used in-place upgrades to upgrade the underlying kubernetes cluster, consider creating a manual backup as well as automatically backing up the environment and objects.

Procedure

Use the CDP CLI `dw backup-cluster` command to create the backup data.

```
export CDP_PROFILE=<test / prod / etc>
export CLUSTER_ID=<the-id-of-the-cluster> # the current ID (original ID) of
 the cluster

cdp \
  --profile ${CDP_PROFILE} \
  dw backup-cluster \
  --cluster-id ${CLUSTER_ID} 1>dump_${CLUSTER_ID}.json
```

Example content of the dump_${CLUSTER_ID}.json file:

```
{
  "clusterId": "env-lqhwqs",
  "operationId": "94197da9-fff7-4414-8b56-a30446c75119",
  "timestamp": "2023-08-16T20:22:00+00:00",
  "data": "UEsDBBQACAAIAAAAAAAAAAAA....
  "md5": "5f427b11f01f5540fa961aba8ea232aa"
}
```

The cluster ID is the unique Cloudera Data Warehouse environment identifier. You can use the operation ID to query the backup execution details using the CLI. The data holds the object data and the configuration,. The md5 is a hash for this data. In case the data and this file is lost, the cluster objects cannot be restored automatically.

What to do next

- Monitor database backup jobs

  The backup will automatically start the Hue backup and Data Visualization database backup jobs that you can monitor. Make sure that the database backup jobs finish before destroying the cluster. If the cluster is deleted before the jobs are finished, you cannot recover the application contents.
- Alert settings

  The compaction observability alert settings are backed up. If the configuration has been modified, make a copy of the configurations, and apply them to the new cluster after restoration.

  Using one of the following ways, get the value of the Alert Manager settings:

  - Use the Cloudera Data Warehouse UI:

    Navigate to your environment tile, click Edit, and in Alert Settings, add the alert settings.
  - Use kubectl:

    kubectl get configmap -n istio-system alertmanager -o json
- Azure environments

  - Public Azure environments activated prior to 1.6.3-b319 (released May 5, 2023) support only manual environment backup. New activations require a managed identity for cluster creation. Old clusters do not have this setting available. Automatic recovery is not an option if your Azure was activated in 1.6.3-b319 (released May 5, 2023).
  - Private Azure environments activated prior to 1.9.1-b233 (released July 26, 2024) support only manual environment backup. In this case, automatic recovery is not an option. New activations require a managed identity and the subnet name of the private SQL for cluster creation. Old clusters do not have these setting available.

    Learn more about private Cloudera Data Warehouse deployment options for Azure.

- Grafana dashboards

  Any changes made to the Grafana dashboards will be lost. A new cluster will be provisioned, the data from the previous cluster won't be carried over to the new Grafana deployment.

Monitoring Hue and Data Visualization database backup

The automatic backup procedure saves the Data Visualization database contents to the configured logs or data folders based on availability.

Hue

During the manual or automatic Hue database backup operation it is critical to block any traffic to the running Hue services. If you cannot bring down the cluster, Cloudera recommends you disable end user access to the cluster endpoints. Failing to do so results in errors in addition to existing key constraints and other issues.

Automatic Hue backup

Automatic backup of Hue extracts the saved query and query history and loads them to the new cluster.

Monitoring Hue backup

The backup starts a job to load the database dump file, but does not wait for the job to complete. If you have a large database, the job can take up to an hour to complete. Ensure you allow enough time for the job to succeed.

To monitor Hue backup, log into the cluster and monitor the job status under the database catalog namespace.

```
$ kubectl get jobs -n <database catalog id>
```

The output that shows the hue-backup job looks something like this:

```
$ kubectl get jobs -n warehouse-1692037411-96hk
  NAME                                                COMPLETIONS   DURATION
 AGE
  hue-backup-ede2b8bd-1d53-4d23-a0f9-87d8ec658f74  1/1           11s
 113s
  hue-query-processor-db-create-job                 1/1           8s
 42h
```

Data Visualization

The automatic backup procedure saves the Data Visualization database contents to the configured logs or data folders based on availability.

Automatic backup

Automatic backup of Data Visualization extracts the dashboards, tables and connections. Make sure to wait for the job to finish before destroying the cluster.

Monitoring backup of Data Visualization

The backup starts a job to create the database dump file, but it does not wait for it to complete. In case your database size is large, it can take up to 20 minutes for the job to complete. Make sure to leave enough time for the job to succeed. To monitor Data Visualization backup, you can log into the cluster and see the job status under the viz namespace using the following command to extracts the dashboards, tables and connections:

```
$ kubectl get jobs -n <data visualization id>
```

The output looks something like this:

```
$ kubectl get jobs -n viz-1692216942-fc2g
  NAME                                                COMPLETIONS   DURATION
 AGE
```

```
  viz-backup-d874515a-be7e-4902-ac75-269c14f9580c   1/1           3m3s
  10m
  viz-webapp-vizdb-create-job                       1/1           57s
 99m
```

Decommissioning the existing environment

You follow procedures to first delete the Virtual Warehouses in the Cloudera Data Warehouse Environment and the Data Visualization resources (visuals). Next, you deactivate the environment.

The topic "Delete Data Visualization visuals" describes the resources that are deleted when you delete a Virtual Warehouse as described in the procedure below.

Deleting the Virtual Warehouses

You must delete the Virtual Warehouses in the Cloudera Data Warehouse Environment and the Data Visualization resources (visuals).

About this task

Procedure

1. Log in to the Cloudera web interface, navigate to Data Warehouse Overview , note the name of the Virtual Warehouse you want to modify, and note which Database Catalog it is configured to access.
2. In the Virtual Warehouse you want to delete, click SUSPEND to stop running the Virtual Warehouse.

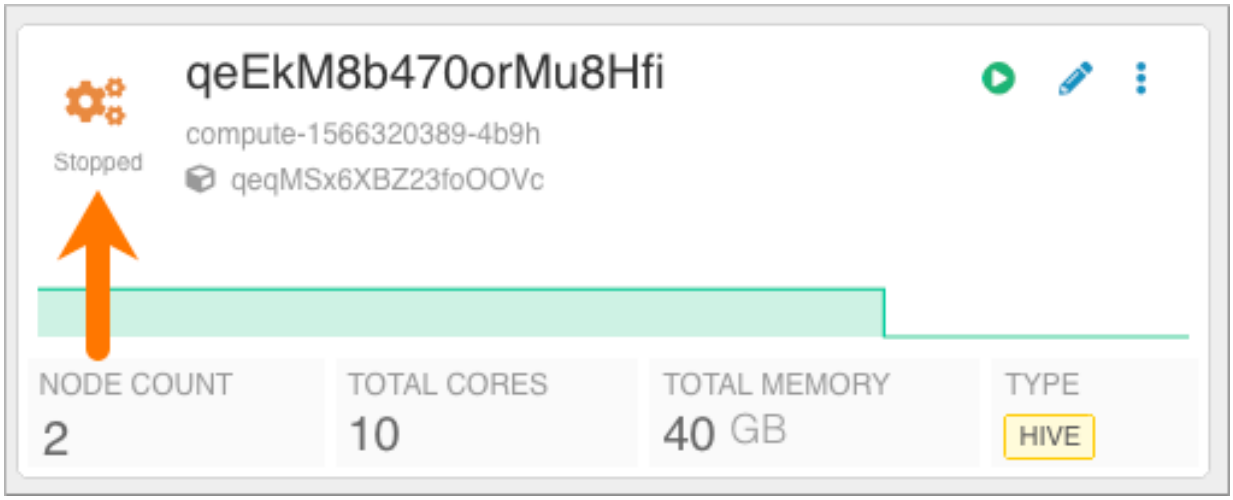

3. Click the options ⋮ of the Virtual Warehouse you want to delete, and select Delete.

   Data Visualization visuals are deleted as described in "Delete Data Visualization visuals" when you delete a Virtual Warehouse.

Deactivating the environment

Assuming you have already deleted your Virtual Warehouses, you need to delete your non-default (custom) Database Catalog and Virtual Warehouses, and then deactivate the environment.

About this task

You cannot delete default Database Catalogs created during environment activation. Default catalogs are deleted when the environment is deactivated.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.

2. Go to the Database Catalogs tab, locate your Database Catalog, and click ⋮ Delete .
3. Go to the **Overview** page and click on the Environments tab.
4. Locate the environment that you want to deactivate and click Deactivate.

Manually reactivating the environment

You learn how to reactivate the AWS or Azure environment.

About this task

Follow this procedure to reactivate the AWS environment or reactivate the Azure environment, and then ensure the reactivated environment is configured the same as the deactivated one. You must add the activated parameters that were backed-up and documented in the previous steps to the new reactivated environment.

Using the CDP CLI

You parameterize the CLI create-cluster command to activate the cluster, as described in the CLI documentation.

1. Activate the cluster by passing the options you retrieved backing up AWS or activating Azure.

   To see all options run the following command.

   ```
   cdp dw create-cluster –help
   ```

   Examples for shorthand and JSON syntax are available.

   For example, an Azure CLI activation option looks something like this:

   ```
   cdp dw create-cluster --environment-crn <crn:cdp:environments:us-west-1:
   abc:environment:123> \
   --use-overlay-network --no-use-private-load-balancer \
   --azure-options \
   userAssignedManagedIdentity="<full managed identity identifier>",subnetI
   d="<full subnet identifier>",enableSpotInstances=false,logAnalyticsWorks
   paceId="<full log analytics workspace identifier>" \
   --profile <customer profile>
   ```

   An AWS CLI activation option looks something like this:

   ```
   cdp dw create-cluster --environment-crn <crn:cdp:environments:us-west-1:
   abc:environment:123> \
   --use-overlay-network --use-private-load-balancer \
   --aws-options \
   lbSubnetIds=<list of subnet identifiers>,workerSubnetIds=<list of subnet
   identifiers>,enableSpotInstances=false --profile <customer profile>
   ```

2. (Optional) If you need to preserve the old environment URL, specify the custom subdomain in the `dw create-cluster` command.

   ```
   --custom-subdomain (string)
   ```

   For example:

   ```
   --custom-subdomain env-qwertyu.dw
   ```

For more information about cluster URLs after reactivation, see “Cluster URLS after reactivation”. For more information about the CLI, see CDP CLI documentation.

Using the Cloudera Data Warehouse UI

Procedure

1. In the Cloudera Data Warehouse service, in Environments, search for and locate the environment that you want to activate.
2. Click Activate to activate the environment.
   The default Database Catalog is created. When the Cloudera Data Warehouse cluster is restored, the IDs of the environment and Database Catalog change.
3. In the Activation Settings, configure the environment using the information you gathered when you backed up activation parameters.
4. (Optional) If you need to preserve the old environment URL, specify the custom subdomain.

   In **Custom Environment Subdomain**, if the environment is env-qwertyu.dw, for example, specify the custom subdomain in the following format:

   ```
   <old environment identifier>.dw
   ```

   Cloudera Data Warehouse UI example:

   For more information about cluster URLs after reactivation, see Cluster URLS after reactivation”.
5. Apply changes.

Modifying configurations after activation

From the Cloudera Data Warehouse UI, you configure alert and observability settings you had in the old environment.

Procedure

1. In your environment tile, click Edit, and in **Alert Settings**, and add the alert settings.
2. In your environment tile, click Edit, and in **Observability**, and add the observability settings.

Restoring the environment and objects

You learn how to use the `dw restore-cluster` command, which you can use either to pass the environment's Cloudera resource name (crn) or to pass the identifier of an activated environment.

About this task

Passing the Cloudera resource name (crn) will activate the cluster from the backup file and restore all the entities and database contents.

Passing an activated environment resource name will restore all the entities and database contents to the running environment. Passing the environment identifier is useful when you need to change activation parameters, but requires manual reactivation.

In the steps below, use `dw restore-cluster` to pass the Cloudera resource name (crn) to activate the cluster.

Before you begin

- You have a public Azure cluster and your Azure cluster must run version 1.6.3-b319 (released May 5, 2023) or later.

  You cannot automatically activate a public Azure cluster that runs version 1.6.2-b197 (released Feb 13, 2023) or earlier.

- You have a private Azure cluster and your Azure cluster must run version 1.9.1-b233 (released July 26, 2024) or later.

  You cannot automatically activate a private Azure cluster that runs version 1.8.7-b37 (released March 26, 2024) or earlier.

  Learn more about private Cloudera Data Warehouse deployment options for Azure.

- You must use the same Cloudera Data Warehouse version to restore files that you used to back up those files.

  Using a backup file from older than the current version will not work.

Procedure

1. Get your environment resource name from the Cloudera portal by selecting the environment that is not activated, and clicking Manage.
   The environment properties open.

   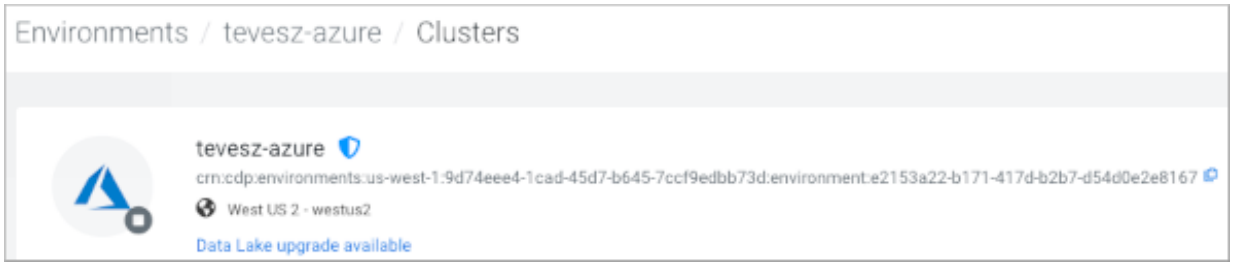

   Under the environment resource name the Cloudera resource name (crn) appears.

   ```
   crn:cdp:environments:us-west-1:98765432-abcd-45d7-b645-7ccf9edbb73d:envi
   ronment:00000000-7bf2-4aeb-af71-f2bf2c038588
   ```

2. Create a CLI skeleton file to serve the base file for the restore command.
   For example, replace your environment resource name placeholder <your cluster name> with the environment resource name of the newly activated cluster (for example env-npk886 shown step 3 of Reactivating the environment).

   ```
   export CLUSTER_NAME="<your cluster name>"
   cdp \
     dw restore-cluster \
     --generate-cli-skeleton 1>restore_${CLUSTER_NAME}_cli_input.json
   ```

3. Open restore_<CLUSTER_NAME>_cli_input.json for editing, and fill in the clusterId and the data fields.
   For example:

   ```
   {
      "clusterId": "crn:cdp:environments:us-west-1:98765432-abcd-45d7-b645-7c
   cf9edbb73d:environment:00000000-7bf2-4aeb-af71-f2bf2c038588",
      "data": "UEsDBBQ…AAAAAAAAAAABkYXRhUEsFBgAAAAABAAEAMgAAAKuBAQAAAA==",
   }
   ```

4. Use the `dw restore-cluster` command, provide the same CLUSTER_NAME as you used in step 3 and use the CDP_PROFILE from your CLI configuration.

   ```
   export CLUSTER_NAME="<your cluster name>"
   export CDP_PROFILE="<your CDP CLI profile>"
   ```

```
cdp \
  --profile ${CDP_PROFILE} \
  dw restore-cluster \
    --cli-input-json file://restore_${CLUSTER_NAME}_cli_input.json
```

Example output:

```
{
    "clusterId": "crn:cdp:environments:us-west-1:98765432-abcd-45d7-b645-7
ccf9edbb73d:environment:00000000-7bf2-4aeb-af71-f2bf2c038588",
    "operationId": "62408134-3d8c-46e8-a914-0f427fc3b1b1",
    "action": "Create",
    "message": "the cluster will be created",
    "dbcRestorePlans": [
        {
            "ref": "test-aws-dl-default",
            "id": "warehouse-1692719478-xrm4",
            "action": "Create",
            "message": "the SDX-type DB Catalog will be created based on
 the data referenced in the backup as test-aws-dl-default"
        }
    ],
    "hueRestorePlans": [
        {
            "ref": "test-aws-dl-default",
            "id": "warehouse-1692719478-xrm4",
            "action": "LoadOrOverwrite",
            "message": "Hue restore is started for warehouse-1692719478
-xrm4 DB Catalog, referenced in the backup data as test-aws-dl-default.
Restore will overwrite Hue database with the backup if it isn't empty."
        }
    ],
    "hiveRestorePlans": [
        {
            "ref": "test-hive",
            "action": "Create",
            "message": "the test-hive Hive Virtual Warehouse will be cr
eated and attached to the warehouse-1692719478-xrm4 DB Catalog"
        }
    ],
    "impalaRestorePlans": [
        {
            "ref": "test-impala",
            "action": "Create",
            "message": "the test-impala Impala Virtual Warehouse will be
 created and attached to the warehouse-1692719478-xrm4 DB Catalog"
        }
    ],
    "vizRestorePlans": [
        {
            "ref": "test-viz",
            "action": "Create",
            "message": "the test-viz Data Visualization will be created"
        }
    ]
    }
```

After several minutes the environment will be activated, the Virtual Warehouses will be created in the new cluster and attached to the Database Catalog. The Virtual Warehouse and Data Visualization ids will be changed.

The Data Visualization database will be recovered. However, because this is a new deployment, the recovered connections will be broken.

5. Monitor the environment restoration as described in Monitoring environment restoration.
6. Adjust the Data Visualization connection settings to point to the new Virtual Warehouse(s).

Testing the restoration

The environment and entity automated restore process can deploy objects to any given environment. You can validate all entities and their settings to gain confidence that the restore operation will succeed for the production environment.

If the data lake storage path for the restored environment if different from than the data lake storage path for the backed up environment, the database restore jobs will fail. The jobs will not be able to reach the database backup file paths; this is expected. The restoration will report the failure, but all entity deployment occurs normally. Customers with older environments might want to consider either testing the restore process first or making a manual backup of the cluster and its properties.

To test the restore process, define a test data lake environment crn in the restore file and follow the details from the "Automatically restoring the environment" page.

Monitoring Hue and Data Visualization restoration

The restore process is designed to be an idempotent process, it can be restarted as many times as you want. In case the environment is activated and healthy, the restore operation can be run multiple times to restore the Virtual Warehouse and Data Visualization objects.

Hue automatic restoration

Restoration of the Hue database occurs only if no Virtual Warehouses arevattached to the particular Database Catalog. In the CDP CLI response to "hueRestorePlans" section of the dw restore-cluster command is the "LoadOrOverwrite" action. This action denotes that the Hue restore operation will run. In any other case the response will be "Skip".

To retry a failed Hue restore operation using the CDP CLI dw restore-cluster command, you must first discommission Virtual Warehouses.

Cloudera can only ensure the data-correctness if there are no running Virtual Warehouses attached to the Database catalog. During the automated restore process the creation of the Virtual Warehouses is deferred until the Hue restore completes. Depending on the size of the Hue backup, this process can take up to 30 minutes. During this time the Cloudera Data Warehouse UI indicates that the Environment and the Database Catalog is created, but no Virtual Warehouses appear in the UI.

Learn more about how to monitor the Hue restoration process in the following section.

Monitoring Hue restoration

The restoration starts a job to load the database dump file, but does not wait for the job to complete. If you have a large database, the job can take up to an hour to complete. Ensure you allow enough time for the job to succeed.

To monitor Hue restoration, log into the cluster and monitor the job status under the database catalog namespace.

```
$ kubectl get jobs -n <database catalog id>
```

The output that shows the hue-restore job looks something like this:

```
$ kubectl get jobs -n warehouse-1692037411-96hk
     NAME                                                COMPLETIONS   DURATIO
N   AGE
     hue-restore-ede2b8bd-1d53-4d23-a0f9-87d8ec658f74   1/1           11s
     113s
     hue-query-processor-db-create-job                  1/1           8s
    42h
```

You can monitor the Hue restoration process using the CDP CLI dw list-events --operation-id <operation-id> command. Use the operation id acquired with the dw restore-cluster command to see events belonging to the long running restore process.

Monitor the "HueRestoreWait" events. For example:

```
{
            "operationId": "5cd90dab-5b15-4de9-b70f-ddab0dfa0c10",
            "event": "HueRestoreWait",
            "message": "{\"type\":\"info\",\"message\":\"Waiting Hue data re
store job to finish\",\"error\":null}",
            "timestamp": "2024-03-21T09:15:11+00:00"
        },
        {
            "operationId": "5cd90dab-5b15-4de9-b70f-ddab0dfa0c10",
            "event": "HueRestoreWait",
            "message": "{\"type\":\"info\",\"message\":\"Hue Data restore st
ill processing: job status, active: 1, failed: 0, succeeded: 0\",\"error\":n
ull}",
            "timestamp": "2024-03-21T09:15:26+00:00"
        },
```

As Hue is restored, the following HueRestoreWait message will appear: "active: 1, failed: 0, succeeded: 0"

Upon the restoration of Hue, the following message appears: "Hue data restore job finished"

Data Visualization automatic restoration

If a Data Visualization object is not present on the cluster, but the backup file contains it, it will be restored to the cluster. In case such an entity is already deployed, no changes or configuration updates will take place.

Automatic restoration of Data Visualization loads the dashboards, tables, and connections to the new applications. Make sure to wait for the job to finish before destroying the cluster.

To monitor restoration of Data Visualization, you can log into the cluster and see the job status under the viz namespace using the following command.

```
$ kubectl get jobs -n <data visualization id>
```

The output will be similar to this, the viz-restore job shows the status.

```
$ kubectl get jobs -n viz-1692216942-fc2g
     NAME                                                COMPLETIONS   DURAT
ION   AGE
     viz-restore-d874515a-be7e-4902-ac75-269c14f9580c   1/1           3m3s
     10m
     viz-webapp-vizdb-create-job                        1/1           57s
      99m
```

The job logs contain the upload path where the backup file has been downloaded from.

Automatic restoration of Data Visualization

Automatic backup and restore for Data Visualization extracts the dashboards, tables and connections. Make sure to wait for the job to finish before destroying the cluster. In the event of a restoration failure, try manually restoring Data Visualization.

Monitoring Data Visualization restoration

To monitor the restoration of Data Visualization, you can log into the cluster and see the job status under the viz namespace using the following command.

```
$ kubectl get jobs -n <data visualization id>
```

The output looks something like this:

```
$ kubectl get jobs -n viz-1692216942-fc2g
  NAME                                                COMPLETIONS   DURATION
 AGE
  viz-restore-d874515a-be7e-4902-ac75-269c14f9580c   1/1           3m3s
   10m
  viz-webapp-vizdb-create-job                        1/1           57s
 99m
```

Manual backup and restore

You perform back up and restoration of your cluster manually in these steps.

Prerequisites

- You have Cluster Administrator privileges and can access the Cloudera Data Warehouse web UI.
- You must use the same Cloudera Data Warehouse version to restore files that you used to back up those files.

Manually backing up the environment

Backing up the environment consists of capturing environment activation parameters, observability configurations, Virtual Warehouse parameters, the Hue data, and Data Visualization applications.

If you automatically backed up the environment following procedures above, do not perform procedures below for backing up the environment.

Gathering AWS environment activation parameters

You back up AWS environment activation parameters using the Cloudera Data Warehouse UI , AWS CLI, and kubectl.

About this task

AWS environment activation settings that need to be available in the new environment may include IP-CIDRs for the Kubernetes cluster and load balancer, the deployment mode setting, reduced permissions mode, and overlay networks. It is required that you gather and document these settings to have the environment behave the same after the back-up/restore process as before. The activation parameter values are available in the Cloudera Data Warehouse UI. Additionally, some parameters may be fetched using AWS CLI and kubectl.

The following steps gather and document the environment parameters.

Procedure

Deployment Mode (Network selection)

1. Get subnet information in one of the following ways:
   - Use the Cloudera Data Warehouse UI

     Navigate to Environment Details General Details , and note the settings for **Public Subnets** or **Private Subnets**.
   - Use the AWS CLI

     ```
     aws cloudformation describe-stacks --stack-name env-q4tzxd-dwx-stack --o
     utput json --query "Stacks[].{StackName:StackName, PublicSubnetIds:Outpu
     ```

```
ts[?OutputKey=='PublicSubnetIds'].OutputValue, PrivateSubnetIds:Outputs[
?OutputKey=='PrivateSubnetIds'].OutputValue}"
```

IP-CIDR for kubernetes cluster and load balancer

2. In the Cloudera Data Warehouse UI, navigate to Environment Details Configurations , and note the settings of Enable IP-CIDR for Kubernetes cluster and Enable IP-CIDR for the load balancer.

Overprovision nodes

3. Document the value of Overprovision nodes.

   Stored behind the CDW_CLUSTER_OVERPROVISIONER entitlement.

Use Custom ECR repository

4. Document the value of Use Custom ECR repository.

   Stored behind the CDP_CUSTOM_REPO entitlement.

Use Overlay Network

5. Using kubectl, get the value of Use Overlay Network.

```
kubectl get daemonsets -n kube-system
```

Note: If the daemonSet aws-node is not present, the overlay network is enabled.

Attach Managed policy ARN to Node Role

6. Using the AWS CLI, fetch the nodeInstanceRole.

```
export nodeInstanceRole=$(aws iam list-roles --query "Roles[].{RoleName:
RoleName}" | grep "env-q4tzxd-dwx-stack-NodeInstanceRole-*" | tr -d '"'|
 cut -f 2 -d ':'| awk '{$1=$1};1')
```

7. List the policies attached to this nodeInstanceRole.

```
aws iam list-role-policies --role-name $nodeInstanceRole
```

8. Check for customized policies, which are not included in the following policies:

```
"PolicyNames": [
  "cluster-autoscaler",
  "dynamodb",
  "ebs",
  "efs",
  "kms",
  "limits-monitoring",
  "s3-list-all-buckets",
  "s3-read-only-buckets",
  "s3-read-write-own-buckets"
]
```

AMI ID

9. Using one of the following ways, get the value of the AMI ID only if you need to use a custom AMI:

   - Use the Cloudera Data Warehouse UI:

     Navigate to Environment Details Configurations , and note the setting of AMI ID.
   - Use AWS CLI:

     ```
     aws cloudformation describe-stacks --stack-name env-q4tzxd-dwx-stack --o
     utput json --query "Stacks[].{StackName:StackName, AMI:Parameters[?Param
     eterKey=='EksAmi'].ParameterValue}"
     ```

Reduced Permissions Mode

10. Using the Cloudera Data Warehouse UI, determine whether or not your current policy has standard activation permission.

    If the current policy does not have the standard activation permission, when you reactivate the environment you see a prompt to use the Reduced Permissions Mode.

    In Reduced Permissions Mode, you deploy the cluster manually. A Cloudformation template is generated and the you must deploy the cluster and apply some role-based access control roles to the Kubernetes cluster.

Enable CloudWatch logs

11. Using the Cloudera Data Warehouse UI, navigate to Environment Details Configurations Enable CloudWatch Logs to get the value of Enable CloudWatch logs..

    CloudWatch logs provide better visibility into cluster operations in addition to the diagnostic bundles within Cloudera Data Warehouse. Enabling this option will not impact the restore procedure, even if it was not previously configured in the old environment.

Additional External Buckets

12. Using the Cloudera Data Warehouse UI, gather values of the Additional External Buckets parameter. Navigate to Environment Details Configurations Bucket Name .

Gathering Azure activation parameters

You configure almost all (99%) of Azure cloud resources using environment activation parameters. These parameters are available by querying Azure resource providers in the old environment. You use these parameters, which you manually document, during the activation of the new environment.

About this task

The Azure environment activation settings you want to carry over to the new environment include the compute VM size (E16ds_v4 or E16_v3), any user-assigned, managed identity, subnets, private Cloudera Data Warehouse and IP CIDRS, overlay networking, and minimum permissions.

Procedure

1. Obtain a managed identity for Azure activations.

   The new, required managed identity parameter provides privileges to deploy the AKS cluster. For more information about required minimal privileges, see "Setting up minimum permissions".

2. Query the environment to get the Azure environment activation settings you want to carry over to the new environment.
   Query the environment to get the activation parameters.

```
az aks show -n <AKS_CLUSTER_NAME>  -g <CDW_RESOURCE_GROUP> --query '{Age
ntpools:agentPoolProfiles[].{Name:name, Version:orchestratorVersion, Sta
te:provisioningState, AZ:availabilityZones, SKU:vmSize, VnetSubnet:vnetS
ubnetId, PodSubnet:podSubnetId, CDW_Timestamp:tags.timestamp, PowerState
: powerState}, Api:apiServerAccessProfile, NetworkType:networkProfile.ne
tworkPluginMode, DockerCIDR:networkProfile.dockerBridgeCidr, outboundTyp
e:networkProfile.outboundType, privateFQDN:privateFqdn, Identity:identity,
 FQDN:fqdn, AKSVersion:kubernetesVersion, Location:location, SKU: sku, OMS
: addonProfiles.omsagent}' -o jsonc
```

   The query output maps to the following activation parameters:

   - Compute VM Size: Agentpoolc.SKU
   - Subnet: Agentpoolc.VnetSubnet
   - Private Cloudera Data Warehouse: api.enablePrivateCluster
   - Managed identity: Identity
   - Availability Zones: agentpoolss.AZ
   - AKS Monitoring: oms
   - K8s CIDR: api.authorizedIpRanges

- Overlay networking: NetworkType
- Docker CIDR: dockerCidr
- AKS DNS Zone: api.privateDnsZone
- OutboundType: outboundType

3. Get the internal load balancer settings in one of the following ways.

```
az resource list -g  MC_<AKS_CLUSTER_NAME>_<REGION> --query "[?type == '
Microsoft.Network/loadBalancers'].{Name: name, Type: type}" -o jsonc
```

If the output lists an internal load balancer, the environment has been activated with the **Enable internal load balancers** option.

4. Get the PostgreSQL server setting using Azure CLI.

```
az postgres flexible-server show -g <CDW_RESOURCE_GROUP> -n <SQL_SERVER_
NAME> --query '{Name:name, FQDN: fullyQualifiedDomainName, Version: vers
ion, Network:network}' -o jsonc
```

Note the following activation parameters:

- Private-sql-subnet-name: delegatedSubnetResourceId
- Private-dns-zone-sql: privateDnsZoneArmResourceId

Gathering observability configurations

Environment configurations are used to monitor and observe an environment.

About this task

If you did not make changes to the environment for Observability or the Alert Manager, skip this step.

Procedure

Alert Manager setting

1. Using one of the following ways, get the value of the Alert Manager settings:
   - Use the Cloudera Data Warehouse UI:

     In Environments, search for and find your environment tile, and in Actions, click ⋮ and select Edit Alert Settings , add the alert settings.
   - Use kubectl:

     ```
     kubectl get configmap -n istio-system alertmanager -o json
     ```

Observability configurations

2. Using the Cloudera Data Warehouse UI, in Environments, search for and find your environment tile, and click ⋮ and select Edit Observability , and copy the configurations. Observability::json > ObservabilityConfig>

Backing up Virtual Warehouse parameters

You use the CDP CLI version 0.9.88 or later.

Procedure

Use the CDP CLI `dw backup-cluster` command to create the backup data.

```
export CDP_PROFILE=<test / prod / etc>
export CLUSTER_ID=<the-id-of-the-cluster> # the current ID (original ID) of
 the cluster
```

```
cdp \
  --profile ${CDP_PROFILE} \
  dw backup-cluster \
   --cluster-id ${CLUSTER_ID} 1>dump_${CLUSTER_ID}.json
```

Example content of the dump_${CLUSTER_ID}.json file:


```
{
   "clusterId": "env-qf5cj7",
   "timestamp": "2023-02-16T10:29:16+00:00",
   "data": "UEsDBBQ…AAAAAAAAAAABkYXRhUEsFBgAAAAABAAEAMgAAAKuBAQAAAA==",
   "md5": "5a18129ad01b75f315ae518a37004804"
   }
```


In this file, the “clusterId” field denotes the cluster from which the backup was taken. The “data” field contains the configuration backup data itself.

Backing up Hue

Backing up Hue is an automated process that saves the Hue database content. The process places the content in configured logs or data folders based on availability. If, for any reason, you want to manually back up the database, you can choose to do so.

Before you begin

- You must temporarily deploy at least one Virtual Warehouse that runs 2023.0.14.0-15 or later to your environment as described in the steps below if you meet both of the following conditions:
  - You have not deployed Runtime version 2023.0.14.0-15 (released May 5, 2023) or later in any Virtual Warehouse in your cluster.
  - You have deployed only Runtime version 2023.0.13.0-20 (released Feb 7, 2023) or earlier in any Virtual Warehouse in your cluster.

  1. Create a Virtual Warehouse that runs 2023.0.14.0-15 or later.
  2. Delete the Virtual Warehouse you just created.

     The steps above resolve a Hue schema incompatibility issue before backing up and restoring Hue.

Manual backup

If anything goes wrong with the automatic backup of Hue, or if you just prefer a manual process, you can back up Hue manually. You can choose to manually save and restore the Hue data to keep the Hue saved queries and query history for your Virtual Warehouses on the new cluster.

About this task

One Hue database is shared between all Virtual Warehouses, so you execute the following steps only once. To back up Hue, follow steps below to create a Postgres dump.

Procedure

1. Go to the Hue container.

   ```
   $ export VW=<virtual_warehouse_id>
   $ kubectl -n $VW exec -i -t huebackend-0 --container hue -- /bin/bash
   ```

2. Check the Hue database configuration.

   ```
   $ vi /etc/hue/conf/zhuei.ini
   ```

Example of content:

```
[[database]]
engine=postgres
host="postgres-service"
port=5432
user="hive"
password_script=/etc/hue/conf/altscript.sh hue_database_password
name="warehouse-1702498301-jjtw_huedb"
```

3. Get database name and password

```
$ IFS="=" read nameKey HUEDB <<< $(cat /etc/hue/conf/zhue.ini | grep huedb
 | sed 's/"//g')
$ /etc/hue/conf/altscript.sh hue_database_password; echo
```

4. Dump the database backup file and input password when prompted.

```
$ pg_dump -h postgres-service -U hive --schema-only -W $HUEDB > /tmp/hue
_database_backup_schema-only.psql
$ pg_dump -h postgres-service -U hive -W -F c $HUEDB > /tmp/hue_database_b
ackup.psql
```

5. Copy the database backup file to a local backup path.

```
$ kubectl cp $VW/huebackend-0:/tmp/hue_database_backup_schema-only.psql -c
 hue <local_backup_path>/hue_database_backup_schema-only.psql
$ kubectl cp $VW/huebackend-0:/tmp/hue_database_backup.psql -c hue <loca
l_backup_path>/hue_database_backup.psql
```

Backing up Data Visualization applications

You can use kubectl or k9s to back up Data Visualization (DataViz) applications in an AWS environment.

About this task

To keep the charts and dashboards that you created, you must save and restore Data Visualization data.

If you did not make changes to the environment for Data Visualization, skip these steps; otherwise, perform these steps on each of your Data Visualization applications because each Data Visualization uses a separate database.

Before you begin

- Data Visualization must use the DataLake Postgres instance for storing its metadata.

Procedure

1. Find the necessary information, such as database name, host, port, user, and password.

```
$ kubectl get secrets/pg-db-secret -o=jsonpath={.data.'\.pgpass'} -n viz
-1680129861-kbtr | base64 -D
```

```
postgres-service:5432:metastore:hive:ZufGC6Dmh03N1iW042uTosZtr4XvCtJIYPQ==
```

2. Use kubectl or k9s to access the Hue container in the targeted Cloudera Data Warehouse environment (need KUBECONFIG setup), and find the namespace for Hue and for DataViz.

```
$ kubectl get pods --all-namespaces --field-selector metadata.name=hueba
ckend-0
```

```
$ kubectl get pods --all-namespaces --field-selector metadata.name=viz-
webapp-0
```

If you are running multiple DataViz instances, make a note of the DataViz namespaces and their user-provided friendly names from the Cloudera Data Warehouse UI. On the new cluster, the namespace names will change, so it's important to know where to load the corresponding DataViz deployment.

3. Select one of the Hue namespaces and shell into the container.

```
$ kubectl exec -it huebackend-0 -n <virtual warehouse ID> -c hue -- /bin/
bash
```

4. Get the dump using the code below by providing a DataViz namespace. All DataViz databases can be backed up from this container. If you have multiple namespaces, make a note of the friendly DataViz name pairs.
   pg_dump -U hive -h postgres-service -W -F t <DataViz namespace>_vizdb > ./viz_pg_dump.tar

   If you have multiple DataViz instances running, backup all databases in this step. It is important to use a naming convention that will allow you to identify which database back-up contains the contents of the corresponding DataViz database. This is needed, as the namespace names will be different on the new cluster.
5. Copy the dump file to your local machine.
   For example:

```
kubectl cp <virtual warehouse ID>/huebackend-0:/opt/hive/viz_pg_dump.tar
 ~/Downloads/logs/viz_pg_dump.tar -c hue
```

Decommissioning the existing environment

You follow procedures to first delete the Virtual Warehouses in the Cloudera Data Warehouse Environment and the Data Visualization resources (visuals). Next, you deactivate the environment.

The topic "Delete Data Visualization visuals" describes the resources that are deleted when you delete a Virtual Warehouse as described in the procedure below.

Deleting the Virtual Warehouses

You must delete the Virtual Warehouses in the Cloudera Data Warehouse Environment and the Data Visualization resources (visuals).

Procedure

1. Log in to the Cloudera web interface, navigate to Data Warehouse Overview , note the name of the Virtual Warehouse you want to modify, and note which Database Catalog it is configured to access.
2. In the Virtual Warehouse you want to delete, click SUSPEND to stop running the Virtual Warehouse.

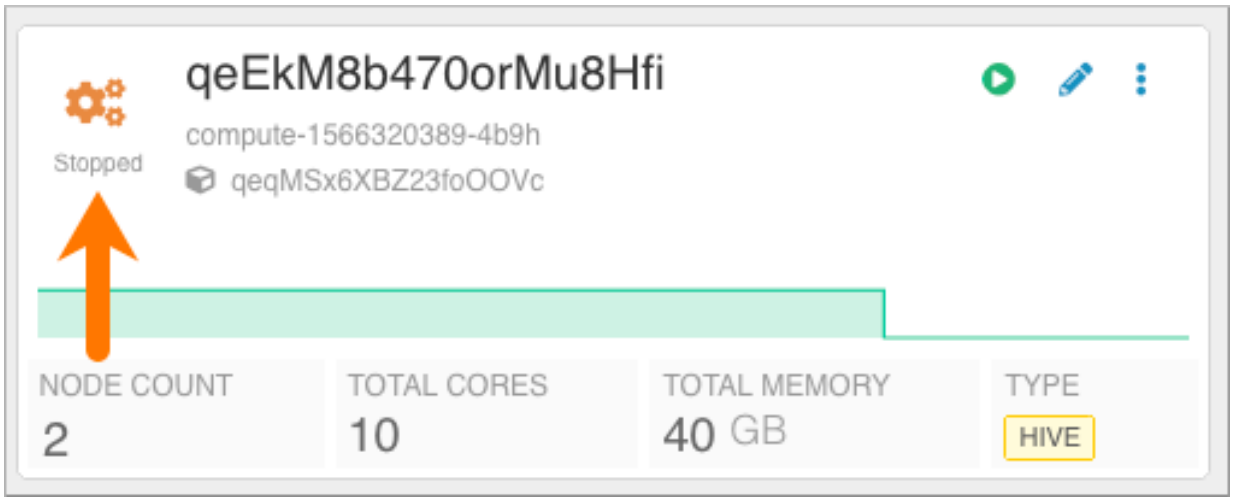

3. Click ⋮ Delete corresponding to the Virtual Warehouse you want to delete.

   Data Visualization visuals are deleted as described in "Delete Data Visualization visuals" when you delete a Virtual Warehouse.

Deactivating the environment

Assuming you have already deleted your Virtual Warehouses, you need to delete your non-default (custom) Database Catalog and Virtual Warehouses, and then deactivate the environment.

About this task

You cannot delete default Database Catalogs created during environment activation. Default catalogs are deleted when the environment is deactivated.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Database Catalogs tab, locate your Database Catalog, and click ⋮ Delete .
3. Go to the **Overview** page and click on the Environments tab.
4. Locate the environment that you want to deactivate and click Deactivate.

Manually restoring the environment

To manually reactivate the environment, you modify configurations, recreate Virtual Warehouses, and restore Hue and Data Visualization. You can automatically reactivate the environment using a number of procedures.

You must use the same Cloudera Data Warehouse version to restore files that you used to back up those files.

Manual restoration consists of the following operations in the order shown below:

Manually reactivating the environment

You learn how to reactivate the AWS or Azure environment.

About this task

Follow this procedure to reactivate the AWS environment or reactivate the Azure environment, and then ensure the reactivated environment is configured the same as the deactivated one. You must add the activated parameters that were backed-up and documented in the previous steps to the new reactivated environment.

Using the Cloudera CLI

You parameterize the CLI create-cluster command to activate the cluster, as described in the CLI documentation.

1. Activate the cluster by passing the options you retrieved backing up AWS or activating Azure.

   To see all options run the following command.

   ```
   cdp dw create-cluster –help
   ```

   Examples for shorthand and JSON syntax are available.

   For example, an Azure CLI activation option looks something like this:

   ```
   cdp dw create-cluster --environment-crn <crn:cdp:environments:us-west-1:
   abc:environment:123> \
   --use-overlay-network --no-use-private-load-balancer \
   --azure-options \
   userAssignedManagedIdentity="<full managed identity identifier>",subnetI
   d="<full subnet identifier>",enableSpotInstances=false,logAnalyticsWorks
   paceId="<full log analytics workspace identifier>" \
   ```

```
--profile <customer profile>
```

An AWS CLI activation option looks something like this:

```
cdp dw create-cluster --environment-crn <crn:cdp:environments:us-west-1:
abc:environment:123> \
--use-overlay-network --use-private-load-balancer \
--aws-options \
lbSubnetIds=<list of subnet identifiers>,workerSubnetIds=<list of subnet
identifiers>,enableSpotInstances=false --profile <customer profile>
```

2. (Optional) If you need to preserve the old environment URL, specify the custom subdomain in the `dw create-cluster` command.

```
--custom-subdomain (string)
```

For example:

```
--custom-subdomain env-qwertyu.dw
```

For more information about cluster URLs after reactivation, see "Cluster URLS after reactivation". For more information about the CLI, see Cloudera CLI documentation.

Using the Cloudera Data Warehouse UI

Procedure

1. In the Cloudera Data Warehouse service, in Environments, search for and locate the environment that you want to activate.
2. Click Activate to activate the environment.
   The default Database Catalog is created. When the Cloudera Data Warehouse cluster is restored, the IDs of the environment and Database Catalog change.
3. In the Activation Settings, configure the environment using the information you gathered when you backed up activation parameters.
4. (Optional) If you need to preserve the old environment URL, specify the custom subdomain.

   In **Custom Environment Subdomain**, if the environment is env-qwertyu.dw, for example, specify the custom subdomain in the following format:

```
<old environment identifier>.dw
```

   Cloudera Data Warehouse UI example:

   For more information about cluster URLs after reactivation, see "Cluster URLS after reactivation".
5. Apply changes.

Modifying configurations after activation

From the Cloudera Data Warehouse UI, you configure alert and observability settings you had in the old environment.

Procedure

1. In your environment tile, click Edit, and in **Alert Settings**, and add the alert settings.

2. In your environment tile, click Edit, and in **Observability**, and add the observability settings.

Disabling end user access

If your business cannot tolerate downtime, you can prevent end user access to your clusters by disabling end user access. Disabling end user access is recommended only when bringing down your clusters is not feasible.

About this task

This procedure is recommended for AWS or Azure clusters that you cannot bring down for some reason.

Procedure

1. In the Cloudera Data Warehouse service, click Overview, search for and locate an activated environment you want to disable.
2. Click the environment options ⋮ and select Edit.
3. Click **Configurations** and make a note of the values for setting **Enable-IP_CIDR** load balancer.
4. Obtain your IP address.
   For example, run `ifconfig` on Windows or `ipconfig` on Linux.
5. Set **Enable IP-CIDR** for the load balancer to your IP address.
   For example, replace x.x.x.x/32 with your IP address 201.123.45.100/32.

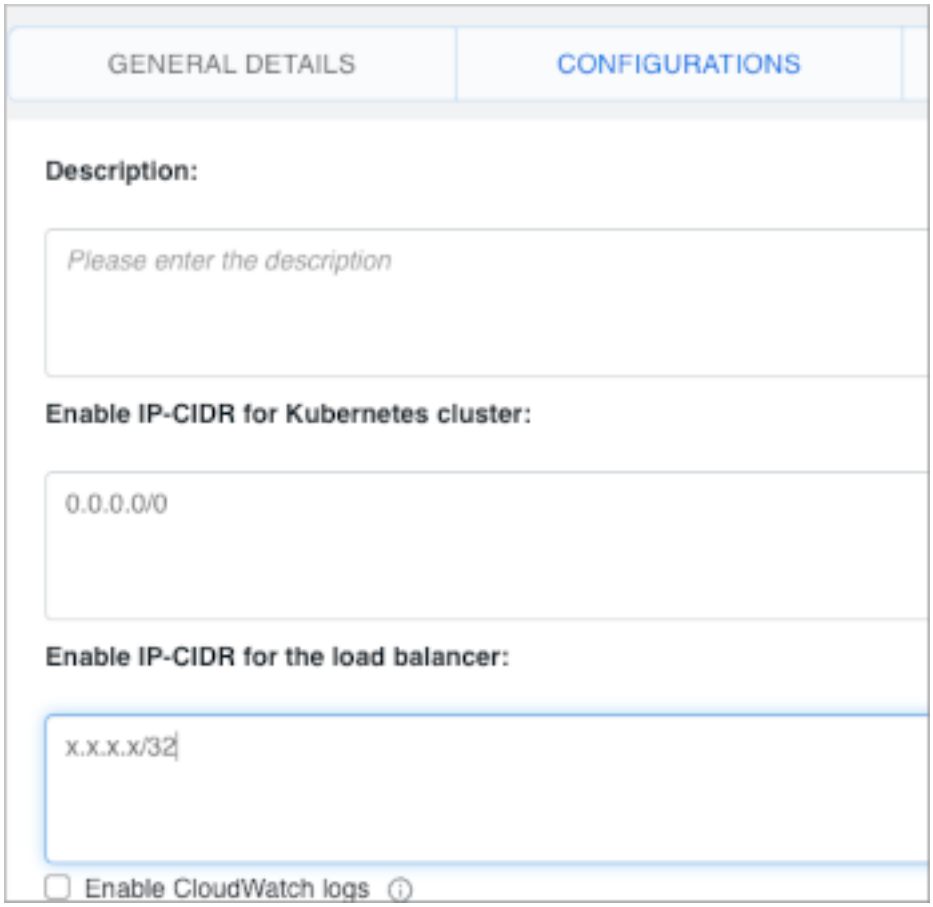

6. Apply changes.
   No user except you can access the Virtual Warehouses. Wait until the cluster updates itself and goes back to Running state to continue the process.

What to do next

You must enable end user access later.

Recreating the Virtual Warehouses

You recreate the Virtual Warehouses in a few steps.

About this task

To recreate the Virtual Warehouses, you use the dw restore-cluster command. This command also restores Data Visualization apps, but additional steps are required as described in section "Restoring Data Visualization".

Procedure

1. Create a CLI skeleton file to serve the base file for the restore command.
   For example, replace <your new cluster id> placeholder with the ID of the newly activated cluster (for example env-npk886 shown step 4 of Reactivating the environment).

```
export NEW_CLUSTER_ID="<your new cluster id>"
cdp \
  --profile ${CDP_PROFILE} \
  dw restore-cluster \
   --generate-cli-skeleton \ file://restore_${NEW_CLUSTER_ID}_cli_input.
json
```

2. Open restore_<new-cluster-id>_cli_input.json for editing, and fill in the *CLUSTERID* and the *DATA* fields.

   To accomplish this, copy contents of the dump_${CLUSTER_ID}.json file from the backup step to the data field in the restore_<new-cluster-id>_cli_input.json file.

   For example:

```
{
   "clusterId": "env-npk886",
   "data": "UEsDBBQ…AAAAAAAAAAABkYXRhUEsFBgAAAAABAAEAMgAAAKuBAQAAAA==",
}
```

3. Use the `dw restore-cluster` command with the CLI input JSON file created in the previous step.

```
cdp \
  --profile ${CDP_PROFILE} \
  dw restore-cluster \
   --cli-input-json file://restore_${NEW_CLUSTER_ID}_cli_input.json
```

   Example output:

```
{
    "clusterId": "env-npk886",
    "operationId": "e279bc25-6eb2-45d5-b2a5-ebdaf8d9c809",
    "dbcRestorePlans": [],
    "hiveRestorePlans": [
        {
            "ref": "org-master-prd-hive",
            "action": "Create",
            "message": "the org-master-prd-hive Hive Virtual Warehouse
will be crated and attached to the warehouse-1676473680-986m DB Catalog"
        }
    ],
    "impalaRestorePlans": [
        {
            "ref": "impala-master-prd-impala",
            "action": "Create",
            "message": "the impala-master-prd-impala Impala Virtual Wareh
ouse will be crated and attached to the warehouse-1676473680-986m DB Cat
alog"
        }
    ],
```

```
    "vizRestorePlans": [
        {
            "ref": "viz-analyst",
            "action": "Create",
            "message": "the viz-analyst Data Visualization will be creat
ed"
        },
        {
            "ref": "viz-scientist",
            "id": "viz-1678182673-jzxs",
            "action": "Skip",
            "message": "the Data Visualization viz-scientist exist with id
 viz-1678182673-jzxs, no change will be applied"
        }
    ]
}
```

After several minutes the Virtual Warehouses will be created in the given cluster and attached to the Database Catalog. The Virtual Warehouse and Data Visualization ids have changed.

4. Monitor the environment restoration as described in Monitoring environment restoration.

What to do next

You must use the new Virtual Warehouse and Data Visualization ids to restore the HUE or DataViz databases as described in the next section.

Restoring Hue

The backup procedure automatically saved the Hue database content and placed the content into the configured logs or data folders based on availability. Using the saved content, the restore process loads the data for the new Hue deployments.

About this task

In the manual backup of Hue, you followed steps to dump the entire Hue database. In the following procedure, you move the Hue backup file from the dump to the new Cloudera Data Warehouse environment.

Before you begin

- Ensure that at least one Hive Virtual Warehouse or Impala Virtual Warehouse exists. Create a Virtual Warehouse if necessary.
- Do not open the Hue web interface prior to completing the steps below.
- During the manual or automatic Hue database restore operation it is critical to block any traffic to the running Hue services. If you cannot bring down the cluster, use the recommended workaround to disable end user access to the cluster endpoints. Failing to do so results in errors in addition to existing key constraints and other issues.

Procedure

1. Copy the database backup files to the container.

```
$ export VW=<virtual_warehouse_id>
$ kubectl cp <local_backup_path>/hue_database_backup_schema-only.psql $VW/
huebackend-0:/tmp/hue_database_backup_schema-only.psql -c hue
$ kubectl cp <local_backup_path>/hue_database_backup.psql $VW/huebackend-
0:/tmp/hue_database_backup.psql -c hue
```

2. Go to Hue container.

```
$ kubectl -n $VW exec -i -t huebackend-0 --container hue -- /bin/bash
```

3. Check the Hue database configuration.

```
$ vi /etc/hue/conf/zhuei.ini
```

Example of content:

```
[[database]]
engine=postgres
host="postgres-service"
port=5432
user="hive"
password_script=/etc/hue/conf/altscript.sh hue_database_password
name="warehouse-1702498325-gqmk_huedb"
```

4. Get the Hue database name and password.

```
$ IFS="=" read nameKey HUEDB <<< $(cat /etc/hue/conf/zhue.ini | grep huedb
 | sed 's/"//g')
$ /etc/hue/conf/altscript.sh hue_database_password; echo
```

5. Resent the Hue database and recreate tables.

```
$ ./build/env/bin/hue reset_db -c --router=default
```

6. Restore the Hue database from the backup file.

```
$ psql -d $HUEDB -h postgres-service -U hive < /tmp/hue_database_backup_
schema-only.psql
$ pg_restore -d $HUEDB -h postgres-service /tmp/hue_database_backup.psql -
c -U hive
$ ./build/env/bin/hue migrate
```

Restoring Data Visualization

You can restore the Data Visualization instance that you backed up.

Before you begin

Do not open the Data Visualization web interface prior to applying the steps below.

Procedure

1. Find the necessary information like database name, host, port, user, and password. On the new cluster, this information will be different from the old cluster.

```
$ kubectl get secrets/pg-db-secret -o=jsonpath={.data.'\.pgpass'} -n <viz-
id> | base64 -D

postgres-service:<port>:<database name>:<user>:<password>
```

For example:

```
$ kubectl get secrets/pg-db-secret -o=jsonpath={.data.'\.pgpass'} -n viz
-1680129861-kbtr | base64 -D

postgres-service:5432:metastore:hive:ZufGC6Dmh03N1iW042uTosZtr4XvCtJIYPQ==
```

Using these properties you will be able to connect to the database and load the backup.

2. Use kubectl or k9s to access Hue container in the targeted Cloudera Data Warehouse environment (need KUBECONFIG setup), and find the namespace for Hue and for Data Visualization.

```
$ kubectl get pods --all-namespaces --field-selector metadata.name=hueba
ckend-0
$ kubectl get pods --all-namespaces --field-selector metadata.name=viz-
webapp-0
```

If you have multiple Data Visualization instances running, match the new Data Visualization namespace names to the user-friendly names on the Cloudera Data Warehouse UI.

3. Select one of the Hue namespaces and copy the dump file to your container.

```
$ kubectl cp ~/Downloads/logs/viz_pg_dump.tar <Hue container>/huebackend
-0:/opt/hive/viz_pg_dump.tar -c hue
```

For example:

```
$ kubectl cp ~/Downloads/logs/viz_pg_dump.tar impala-1679934278-6pgc/hue
backend-0:/opt/hive/viz_pg_dump.tar -c hue
```

4. Shell into the container.

```
$ kubectl exec -it huebackend-0 -n <Hue container> -c hue -- /bin/bash
```

5. Load the dump back to the database, taking care if you have multiple DataViz instances to load the contents back to the right database.

```
pg_restore -d <DataViz database ID> -h postgres-service ./viz_pg_dump.tar
 -c -U hive
```

For example:

```
pg_restore -d viz-1680137534-87s4_vizdb -h postgres-service ./viz_pg_dum
p.tar -c -U hive
```

The pg_restore command might output the following errors as it runs:

```
....
pg_restore: [archiver (db)] Error from TOC entry 258; 1259 302230 TABLE
 apps_apikey hive
pg_restore: [archiver (db)] could not execute query: ERROR:  table "apps_a
pikey" does not exist
    Command was: DROP TABLE public.apps_apikey;

WARNING: errors ignored on restore: 10
```

It is safe to continue restoring Data Visualization; just ignore the errors.

Enabling end user access

If you disabled end user access before recreating your Virtual Warehouse, you must enable access afterward.

About this task

Perform this task only if you disabled end user access.

Procedure

1. In the Cloudera Data Warehouse service, click Overview and in Environments, search for and locate the environment you disabled.

2. Click the environment options ⋮ and select Edit.
3. Click **Configurations** and set the value of **Enable-IP_CIDR** load balancer to the setting you noted when disabling end user access.
4. Apply changes.

Monitoring environment restoration

Restoring the Cloudera Data Warehouse environment is an unvaried process; no configuration update or settings change are applied to existing Virtual Warehouses or Data Visualization applications.

After restoration, a Virtual Warehouse has exactly the same settings as the original Virtual Warehouse according to the following principles.

- Configurations are not copied as is; rather a new configuration is created. All the changes made to the Virtual Warehouse configuration throughout its lifecycle are applied on top of the new configuration.
- Values that were undefined or not present in earlier versions are set to the default.
- Configurations are mapped 1-to-1 during restoration.

Track the progress of the cluster restoration using the operationID from the cluster-restore response. Due to the unvaried nature of the restore process, if any Virtual Warehouse is missing because you deleted it accidentally, or it failed to come up, you can choose to try again.

Steps

1. Check that there are no errors after running the cluster-restore command.

   Example: A cluster-restore command response free of errors looks like this:

```
{
  "clusterId": "env-npk886",
  "operationId": "acbe40f4-560b-485c-833c-451a64bb76c4",
  # truncated output
}
```

2. Use the Cloudera CLI dw list-events commands in conjunction of the operation-id (obtained in the first step) to see the restoration progress.

```
cdp \
   --profile ${CDP_PROFILE} \
   dw list-events \
   --operation-id acbe40f4-560b-485c-833c-451a64bb76c4
```

   The command returns the most recent events. You can use --asc switch to flip the ordering and see the first event. You can limit the output with the --limit switch.

   You might experience a slight delay before the event appears in the audit app.

The output looks something like this:

```
{
  "events": [
  {
    "operationId": "acbe40f4-560b-485c-833c-451a64bb76c4",
    "event": "RestoreCluster",
    "message": "{\"type\":\"info\",\"message\":\"restore cluster operation
 for env-m6mcfd has finished\",\"error\":null}",
    "timestamp": "2023-08-28T12:12:53+00:00"
  },
  {
    "operationId": "acbe40f4-560b-485c-833c-451a64bb76c4",
```

```
      "serviceId": "compute-1693224718-abcd",
      "event": "Completed",
      "message": "Started hive Virtual Warehouse",
      "timestamp": "2023-08-28T12:12:50+00:00"
  }
  ]
}
```

Cloudera Data Warehouse Public Cloud

Using BI tools with Cloudera Data Warehouse

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Built-in BI clients and drivers....4

Using dbt with Hive, Impala and Cloudera....4

Connecting to Impala Virtual Warehouse from Impala shell client....5

Using the Impala shell client....6

Download Beeline tarball....7

Connect to Hive from Tableau....9

Connect a Virtual Warehouse and Microsoft Power BI Desktop....12

Download JDBC JAR....14

Upload additional JARs....15

Built-in BI clients and drivers in Cloudera Data Warehouse

You can connect BI tools and SQL clients such as Beeline, impala-shell, impyla, Tableau, and so on to Cloudera Data Warehouse and use them to explore and query data in the Data Lakehouse.

Cloudera Data Warehouse provides built-in downloadables such as Hive JDBC driver for connecting JDBC-compliant tools to the Virtual Warehouses, Beeline CLI, Impala JDBC and ODBC drivers for connecting to Impala Virtual Warehouses, and a JDBC driver for using with Unified Analytics. You can download these from the Resources and Downloads tile present on the **Overview** page.

Using dbt with Hive, Impala and Cloudera

You can use the dbt adapters for Hive and Impala that enable you to use the dbt data management workflow with Cloudera. Learn how you can access the dbt adapters from the Cloudera Data Warehouse service.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In the **Overview** page of the Cloudera Data Warehouse service, click See More in the Resources and Downloads tile.

3. Click DBT Hive or DBT Impala to launch the required dbt adapter page.

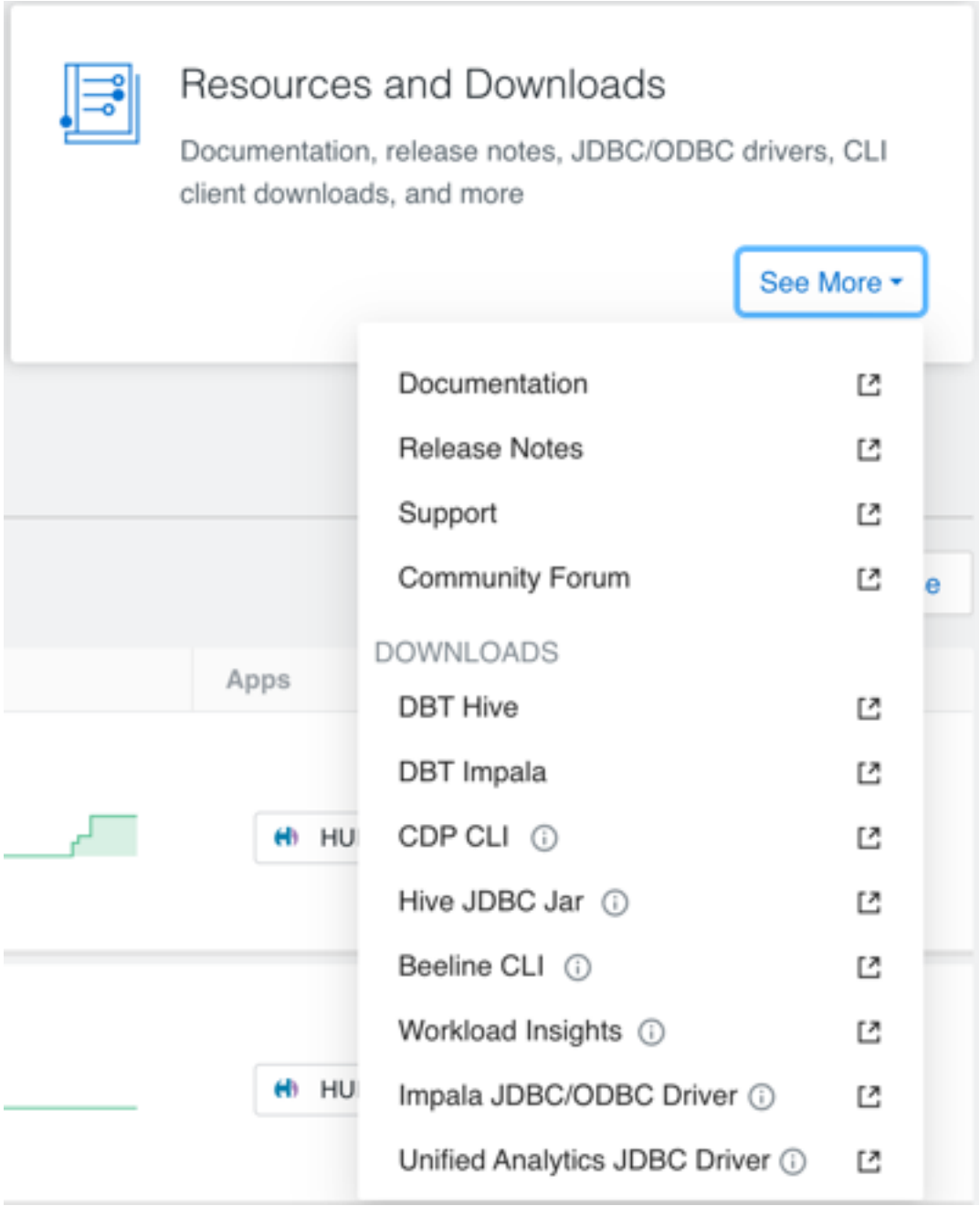

4. Follow the instructions in the dbt adapter page to install and use the adapter.

Connecting to Impala Virtual Warehouse from Impala shell client

You need to provide commands to your client users for installing and launching the Impala shell to connect to your Impala Virtual Warehouse. Client users can then query your tables. You learn how to obtain the command for installing the Impala shell on a client and other information to provide to clients.

About this task

To query an Impala Virtual Warehouse from the Impala Shell, clients need to connect to HiveServer (HS2), which is available in the same cluster as Impala. The JDBC URL for the HS2 endpoint uses the strict HS2 protocol option to access the Impala Virtual Warehouse from the Impala Shell.

Before you begin

- Obtain the DWUser role.
- You must run a Virtual Warehouse version DWX 1.1.2-b2008 or later.

Procedure

1. In the Impala Virtual Warehouse, click Options ⋮, and select Copy Impala shell Download command.

   The command for installing the Impala shell compatible with your Impala Virtual Warehouse is copied to the clipboard. The command looks something like this:

```
pip install impala-shell==4.1.0a1
```

   Using this command, your clients update impyla to the version compatible with Cloudera Data Warehouse, as listed in the Cloudera Data Warehouse Release Notes in section, "Version mapping".

2. Click Options ⋮ again, and select Copy Impala shell command.

   This action copies the command that launches the Impala shell and includes the connection string your client needs to connect to your Virtual Warehouse. For example:

```
impala-shell --protocol='hs2-http' --ssl -i 'coordinator-vw-impala.dw-dwx-
rzs556.xcu2-8y8x.dev.cldr.work:443' -u client_max -l
```

3. Provide the commands you copied to your client user.
4. Provide the instructions in the next topic Using the Impala Shell client to your client user.

Using the Impala shell client

You obtain the commands from the Impala Virtual Warehouse owner for installing and launching the Impala shell to connect to the Impala Virtual Warehouse. You learn how to install and launch the Impala shell that connects

to the Impala Virtual Warehouse UI. You then query tables in the Impala Virtual Warehouse, assuming you have authentication credentials. Windows clients are not supported.

Before you begin

- You must have the latest stable version of Python 2.7.
- You must obtain the pip installer compatible with the Python version.

Procedure

1. In a terminal window, run the command provided by the Impala Virtual Warehouse owner to update impyla to the version compatible with the Impala Virtual Warehouse.
   For example:

   ```
   pip install impyla==0.18a2
   ```

   Installing/updating to impyla 0.18a2, for example, is required before you install the Impala shell in the next step.
2. Run the Impala shell Download command provided by the Impala Virtual Warehouse owner to install the Impala shell.

   The command looks something like this:

   ```
   pip install impala-shell==4.1.0a1
   ```

   The Impala shell is installed.
3. Run the Help command to confirm a successful installation:

   ```
   impala-shell --help
   ```
4. Run the provided Impala shell command to launch the Impala shell.

   The command looks something like this:

   ```
   impala-shell --protocol='hs2-http' --ssl -i 'coordinator-vw-impala.dw-dwx-
   rzs556.xcu2-8y8x.dev.cldr.work:443' -u client_max -l
   ```

   The Impala shell is launched and you connect to the Impala Virtual Warehouse.
5. Run a SQL command to confirm that you are connected to the Impala Virtual Warehouse instance.

   ```
   SHOW DATABASES;
   ```

   The query returns a list of databases in the Impala Virtual Warehouse.

Downloading the Beeline CLI tarball

Download the Beeline CLI tarball from Cloudera Data Warehouse to your local system and use the Beeline client to connect to a Hive Virtual Warehouse and run queries. The archive file contains all the dependent JARs and libraries that are required to run the Beeline script.

Before you begin

From the Cloudera Management Console user profile, note the Workload User Name and Workload Password.

Attention: Ensure that Java is installed on the node on which you want to download and use the Beeline CLI, and you have set the JAVA_HOME environment variable correctly while installing the JDK.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In the **Overview** page of the Cloudera Data Warehouse, click See More in the Resources and Downloads tile.
3. Select Beeline CLI and click to download the file.
4. Save the apache-hive-beeline-x.x.xxxx.tar.gz file in your local system and extract the tarball.
5. In the Data Warehouse service **Overview** page, for the Virtual Warehouse you want to connect to the client, click and select Copy JDBC URL.
6. Paste the copied JDBC URL in a text file, to be used in later steps.

```
jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<dwx.company.co
m>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
```

7. Open a terminal window and go to the folder where the tarball is extracted to start Beeline.
   bin/beeline

   This starts an interactive Beeline shell where you can connect to Hive and run SQL queries.
8. Run the connect command to connect to Hive using the JDBC URL that you copied earlier.

```
beeline> !connect [***JDBC URL***]
```

```
Connecting to jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<
dwx.company.com>/default;transportMode=http;httpPath=cliservice;ssl=true
;retries=3
```

9. Enter the Workload User Name and Workload Password when you are prompted for the user credentials.

```
Enter username for jdbc:hive2://<your-virtual-warehouse>.<your-environme
nt>.<dwx.company.com>/default: [***WORKLOAD USERNAME***]
Enter password for jdbc:hive2://<your-virtual-warehouse>.<your-environm
ent>.<dwx.company.com>/default: [***WORKLOAD PASSWORD***]
Connected to: Apache Hive (version 3.1.2000.7.0.2.2-24)
Driver: Hive JDBC (version 3.1.2000.7.0.2.2-24)
Transaction isolation: TRANSACTION_REPEATABLE_READ
```

10. To verify if you are connected to HiveServer2 on the Virtual Warehouse, run the following SQL command:
    SHOW TABLES;

```
INFO : Compiling command(queryId=hive_20200214014428_182d2b63-a510-421f-
8bbc-65a4ae24d1d6): show tables
INFO : Semantic Analysis Completed (retrial = false)
INFO : Completed compiling command(queryId=hive_20200214014428_182d2b63-
a510-421f-8bbc-65a4ae24d1d6); Time taken: 0.054 seconds
INFO : Executing command(queryId=hive_20200214014428_182d2b63-a510-421f
-8bbc-65a4ae24d1d6): show tables
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hive_20200214014428_182d2b63-a
510-421f-8bbc-65a4ae24d1d6); Time taken: 0.018 seconds
INFO : OK
-----------

table_name
-----------
-----------
No rows selected (0.311 seconds)
```

Connecting to Hive Virtual Warehouses from Tableau

This topic describes how to connect to Tableau with Hive Virtual Warehouses on Cloudera Data Warehouse service.

About this task

Required role: DWUser

Before you begin

Before you can use Tableau with Hive Virtual Warehouses, you must have populated your Database Catalog with sample data when you create it. You must also create a Hive Virtual Warehouse, which is configured to connect to the Database Catalog that is populated with data.

Procedure

1. Download the latest version of the Hive ODBC driver from Cloudera Downloads page.
2. Install the driver on the local host where you intend to use Tableau Desktop.
3. Log in to the Cloudera Data Warehouse service as DWUser.
4. Go to the **Virtual Warehouses** tab, locate the Hive Virtual Warehouse you want to connect to, and click ⋮ Copy JDBC URL .

   This copies the JDBC URL to your system's clipboard.
5. Paste the copied JDBC URL into a text file. It should look similar to the following:

   ```
   jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<dwx.company.co
   m>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
   ```
6. From the text file where you just pasted the URL, copy the host name from the JDBC URL to your system's clipboard. For example, the host name in the URL is:

   ```
   <your-virtual-warehouse>.<your-environment>.<dwx.company.com>
   ```

7. Start Tableau and navigate to ConnectMore…Cloudera Hadoop :

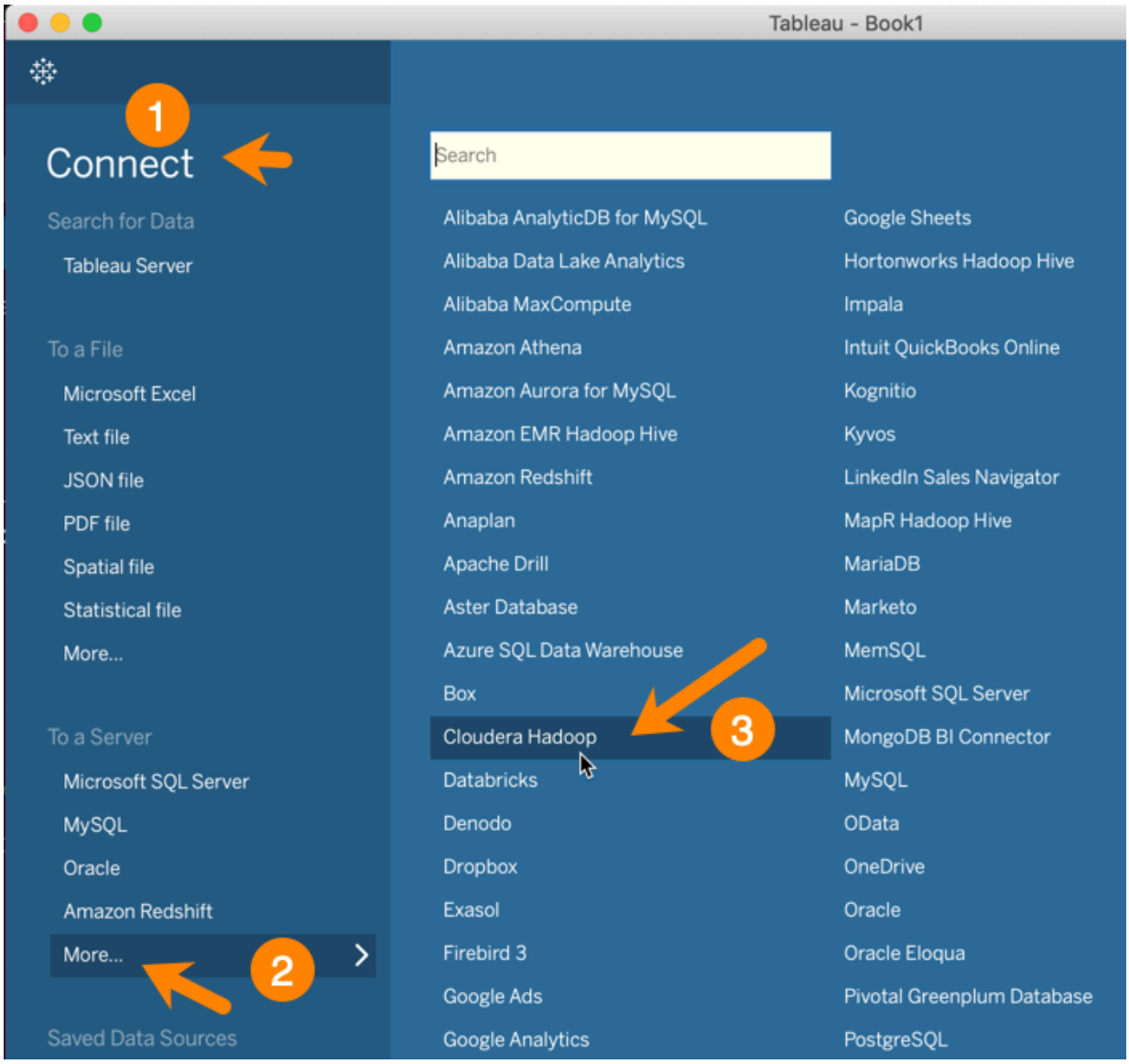

This launches the Cloudera Hadoop dialog box.

8. In the Tableau Cloudera Hadoop dialog box, paste the host name you copied to your clipboard in Step 7 into the Server field:

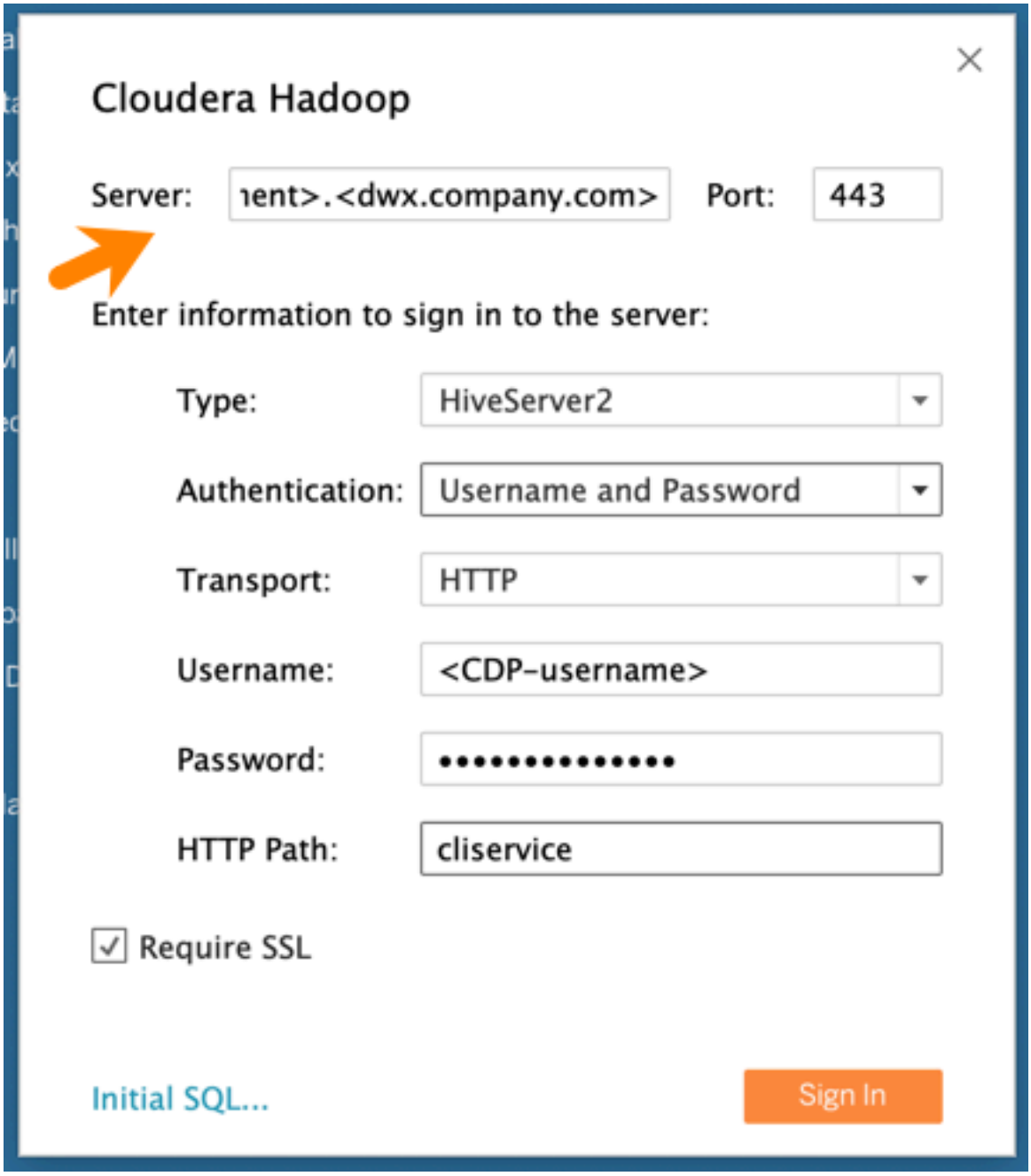

9. Then in the **Tableau Cloudera Hadoop** dialog box, set the following other options:
   - Port: 443
   - Type: HiveServer2
   - Authentication: Username and Password
   - Transport: HTTP
   - Username: Username you use to connect to the Cloudera Data Warehouse service.
   - Password: Password you use to connect to the Cloudera Data Warehouse service.
   - HTTP Path: cliservice
   - Require SSL: Make sure this is checked.
10. Click Sign In.

Related Information

Cloudera Hadoop connection option described in the Tableau documentation

Connect a Virtual Warehouse and Microsoft Power BI Desktop

You learn how to connect to a Hive or Impala Virtual Warehouse from Microsoft Power BI. You can then use Power BI to visualize the data in the Virtual Warehouse on Cloudera.

About this task

Although you configure Power BI using a Microsoft UI labeled Hive LLAP, you can use this procedure to connect to a Hive Virtual Warehouse or to an Impala Virtual Warehouse. After making the connection, you can use Power BI to create reports based on data in the Virtual Warehouse.

Warning: We have not tested or certified connecting a Virtual Warehouse and Microsoft Power BI Desktop.

Before you begin

- You have access to a Cloudera Environment and a Hive LLAP or Impala Virtual Warehouse that contains tables of data.
- You know the Cloudera workload user name and password you set in User Management in the Cloudera Management Console. You need to use your workload user name and its associated password to log into the Virtual Warehouse.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.
2. Go to the **Virtual Warehouses** tab, locate the Hive Virtual Warehouse you want to connect to, and click ⋮ Copy JDBC URL .

   This copies the JDBC URL to your system's clipboard.
3. Paste the copied JDBC URL into a text file. It should look similar to the following:

   ```
   jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<dwx.company.co
   m>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
   ```
4. From the text file where you just pasted the URL, copy the host name from the JDBC URL to your clipboard. For example, in the URL shown in the step above, the host name is:

   ```
   <your-virtual-warehouse>.<your-environment>.<dwx.company.com>
   ```

5. Open Power BI Desktop, select Get Data More .

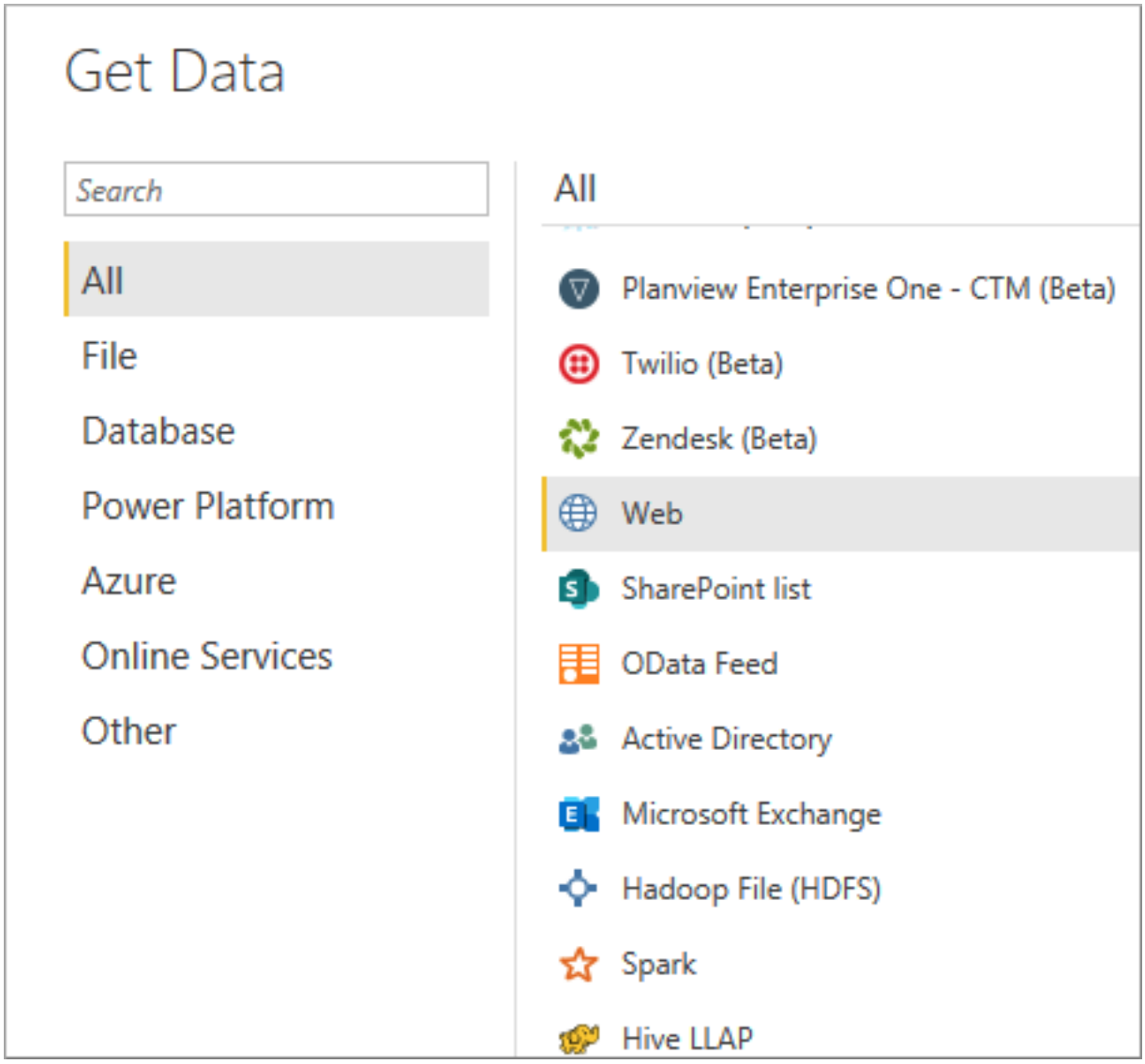

6. Select Hive LLAP.

7. In the Hive LLAP dialog, configure Power BI to connect to the Hive or Impala Virtual Warehouse in Cloudera Public Cloud as follows:
   - In Server, paste the contents of your clipboard from step 4, add a colon and 443.
   - In Database, enter the database name, for example default.
   - In Thrift Transport Protocol, select HTTP.
   - In Data Connectivity mode, select DirectQuery.

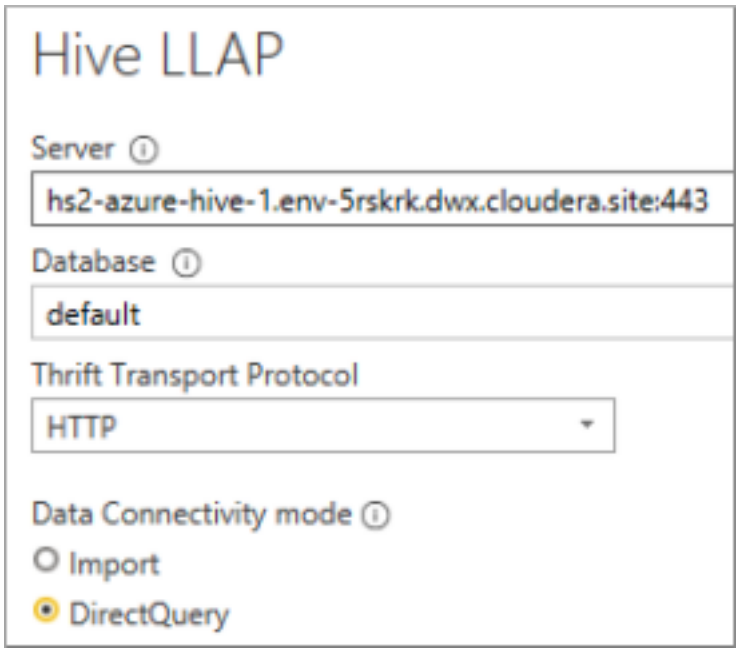

   Click OK.
8. Enter your Cloudera workload user name and password.
9. Click Connect.
   The PowerBI Navigator appears.
10. Select the tables you want to use in a report, and click Load.
    In Fields, the table appears and is now available as a data source for your Power BI report.

Downloading a JDBC driver from Cloudera Data Warehouse

To use third-party BI tools, your client users need a JDBC JAR to connect your BI tool and the service. You learn how to download the JDBC JAR to give to your client, and general instructions about how to use the JDBC JAR.

Before you begin

Before you can use your BI tool with the Cloudera Data Warehouse service:

- You created a Database Catalog.

  You have the option to populate your Database Catalog with sample data when you create it.
- You created a Virtual Warehouse and configured it to connect to the Database Catalog.

Of course, to query tables in the Virtual Warehouse from a client, you must have populated the Virtual Warehouse with some data.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.
2. In the **Overview** page of the Cloudera Data Warehouse service, click See More in the Resources and Downloads tile.
3. Click Hive JDBC Jar to download the Apache Hive JDBC JAR file.
4. Provide the JAR file you downloaded to your JDBC client.

   On most clients, add the JAR file under the Libraries folder. Refer to your client documentation for information on the location to add the JAR file.
5. In the Cloudera Data Warehouse service **Overview** page, for the Virtual Warehouse you want to connect to the client, click ⋮ and select Copy JDBC URL.

   A URL is copied to your system clipboard in the following format:

   ```
   jdbc:hive2://<your_virtual_warehouse>.<your_environment>.<dwx.company.co
   m>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
   ```
6. Paste the URL into a text editor and configure your client BI tool to connect to the Virtual Warehouse using the following portion of the URL, represents the server name of the Virtual Warehouse:

   ```
   <your_virtual_warehouse>.<your_environment>.<dwx.company.com>
   ```
7. In your JDBC client, set the following other options:

   Authentication: Username and Password

   Username: Username you use to connect to the Cloudera Data Warehouse service.

   Password: Password you use to connect to the Cloudera Data Warehouse service.

Uploading additional JARs to Cloudera Data Warehouse

You add additional Java Archive (JAR) files to the Cloudera Data Warehouse Hive classpath that might be required to support dependency JARs, third-party Serde, or any Hive extensions.

About this task

- The JARs are added to the end of the Hive classpath and do not override the Hive JARs.
- Cloudera recommends that you do not use this procedure to add User-Defined Function (UDF) JARs. If you do, then you must restart HiveServer2 or reload the UDF. For more information about reloading functions, see the Hive Data Definition Language (DDL) manual.

Before you begin

You have the EnvironmentAdmin role permissions to upload the JAR to your object storage.

Procedure

1. Build the archive file.

   The archive file can be either a .jar file or a tar.gz file. For a tar.gz archive file, only JARs present in the top level are considered.

   For example, if the tar.gz file contains these files — test1.jar, test2.jar, and deps/test3.jar, only test1.jar and test2.jar are considered; deps/test3.jar is excluded.

   Note: There is no defined priority to use a particular file. If there are multiple files with the same name or same class in multiple jars, any file can be in effect.

2. Upload the archive file to the Hive Virtual Warehouse on Cloudera Data Warehouse object storage, such as AWS S3 or Azure Data Lake Storage (ADLS).
3. Log in to the Cloudera Data Warehouse service and from the Overview page, locate the Hive Virtual Warehouse that uses the bucket or container where you placed the archive file, and click ⋮ and select Edit.
4. In the **Virtual Warehouse Details** page, click Configurations Hiveserver2 .
5. From the Configuration files drop-down list, select env.
6. Search for CDW_HIVE_AUX_JARS_PATH and add the archive file to the environment variable.

   Important: You can specify multiple locations in the CDW_HIVE_AUX_JARS_PATH environment variable separated by a colon (:). For example, s3a://examplebucket/path/to/file1.jar:s3a://examplebucket/path/to/file2.jar or abfs://storage@example/path/to/file1.jar:abfs://storage@example/path/to/file2.jar.

   Important: Cloudera Data Warehouse only supports the default S3 bucket or ADLS Gen 2 container associated with the environment.

   If you add a directory, the .jar or tar.gz files within the directory are copied and extracted. For a tar.gz file, only the JARs present in the top level are copied.

   Consider the following JAR path - /common-jars/common-jars.tar.gz:/common-jars/single-jar.jar:/serde-specific-jar/serde.jar. In this example, common-jars.tar.gz is extracted and single-jar.jar and serde.jar files are copied.
7. Repeat the previous step and add the archive file or directory for Query coordinator and Query executor.

   If the CDW_HIVE_AUX_JARS_PATH environment variable is not present, click + and add the following custom configuration:

   ```
   CDW_HIVE_AUX_JARS_PATH=[***VALUE***]
   ```
8. If you require the JARs for Hive metastore (HMS), go to the corresponding Database Catalog and add the environment variable in Configurations Metastore .

Results

On applying the configuration changes, Hive Virtual Warehouse restarts and the archive files are available and added to the end of the Hive classpath.

Related Information

Hive Data Definition Language manual

Cloudera Data Warehouse Public Cloud

Cloud Support

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Public Cloud provider support 4

Amazon Web Services support in Cloudera Data Warehouse Public Cloud 4

Microsoft Azure support in Cloudera Data Warehouse Public Cloud 6

Custom image repositories 7

Limitations 8
Downloading images 9
Setting up a custom repository 9
Handling Cloudera Data Warehouse upgrades 9
Disabling image verification 10

Public Cloud provider support

Cloudera Data Warehouse Public Cloud service supports Amazon Web Services (AWS) and Microsoft Azure. Basic configuration for environments that use these public cloud providers is performed during registration in Management Console. To configure specific Cloudera Data Warehouse features, you use the Cloudera Data Warehouse service UI.

Amazon Web Services support in Cloudera Data Warehouse Public Cloud

This topic provides an overview of Amazon Web Services (AWS) support in Cloudera Data Warehouse Public Cloud.

Important: Ensure your AWS environment satisfies the items listed in the AWS environments requirements checklist.

The "environment" concept in Cloudera is closely related to the virtual private network in your AWS account. Registering an environment provides Cloudera with access to your AWS account and identifies the resources there that Cloudera services can access or provision. A single environment is contained within a single AWS region, so all resources deployed by Cloudera are deployed within that region within one specific virtual network. Once you've registered an environment with Cloudera Management Console, you can start provisioning resources such as clusters, which run on the physical infrastructure in an AWS data center.

The following diagram shows the components of a Cloudera environment on AWS:

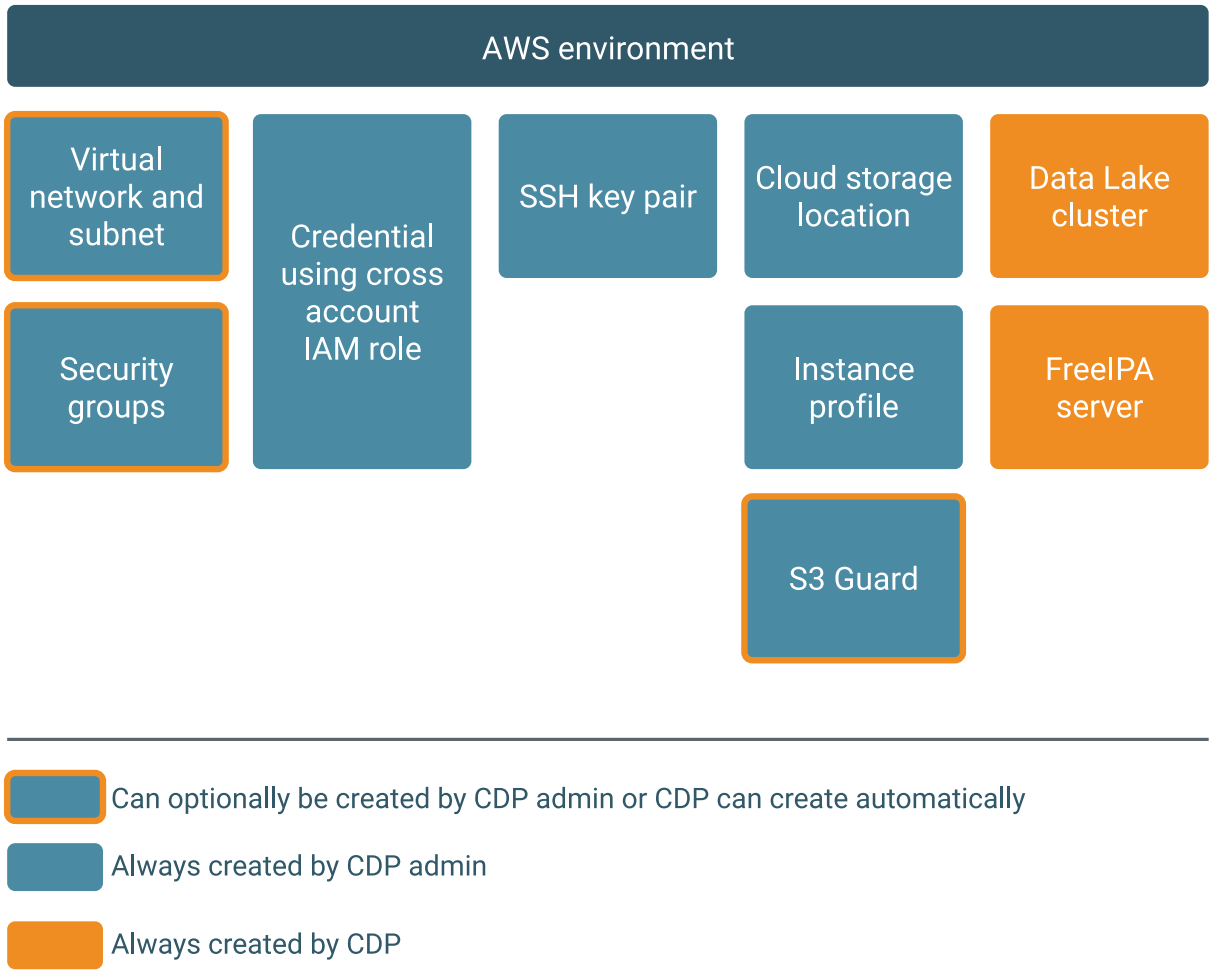

The diagram includes all major user-created and Cloudera-created components of an AWS environment:

- Items in dark blue boxes with orange outlines can be automatically provisioned by Cloudera in your AWS account. Or you can optionally pre-create them in your AWS account and then provide them when registering an environment in Cloudera.
- Items in dark blue boxes must be pre-created by your Cloudera administrator prior to environment registration and then provided during environment registration in Cloudera.
- Items in orange boxes are automatically provisioned on AWS by Cloudera as part of environment registration.

Note: Items that are user-created are not terminated when a Cloudera environment is deleted.

Information that describes how to register an AWS environment with Cloudera is linked to at the end of this page.

Cloudera Data Warehouse service features for AWS environments

Cloudera Data Warehouse offers the following additional features for AWS environments that are used for Database Catalogs and Virtual Warehouses:

- Restrict access to Kubernetes endpoints and to service endpoints of the Kubernetes cluster at the load balancer lever by specifying a list of IP CIDRs that are allowed access. For more information, see "Restricting endpoint access in AWS," which is linked to at the end of this page.
- Set up private networking in AWS environments, which uses AWS private subnets. Private subnets receive no direct inbound connections from the internet, providing private network connectivity for workload endpoints in

the Cloudera Data Warehouse service. For more information, see "Set up private networking in AWS," which is linked to at the end of this page.

These features require additional configuration in the Cloudera Data Warehouse service UI to use them.

Related Information

AWS account requirements
AWS credentials
How to register AWS environments
Restricting endpoint access in AWS
Set up private networking in AWS

Microsoft Azure support in Cloudera Data Warehouse Public Cloud

This topic provides an overview of Azure support in Cloudera Data Warehouse Public Cloud.

Important: Ensure your Azure environment satisfies the items listed in the Azure environments requirements checklist.

Like Amazon Web Services support, Cloudera environments are closely related to the virtual private network concept in your Azure account as well. Also registering an environment provides Cloudera access to resources in your Azure account and a single Cloudera environment is contained within a single region in Azure. All resources that are deployed by Cloudera are deployed within that region and within one specific virtual network. After you have registered an environment in Cloudera, you can start provisioning Cloudera resources such as clusters, which run on the physical infrastructure in an Azure data center.

The following diagram shows the components of a Cloudera environment on Azure:

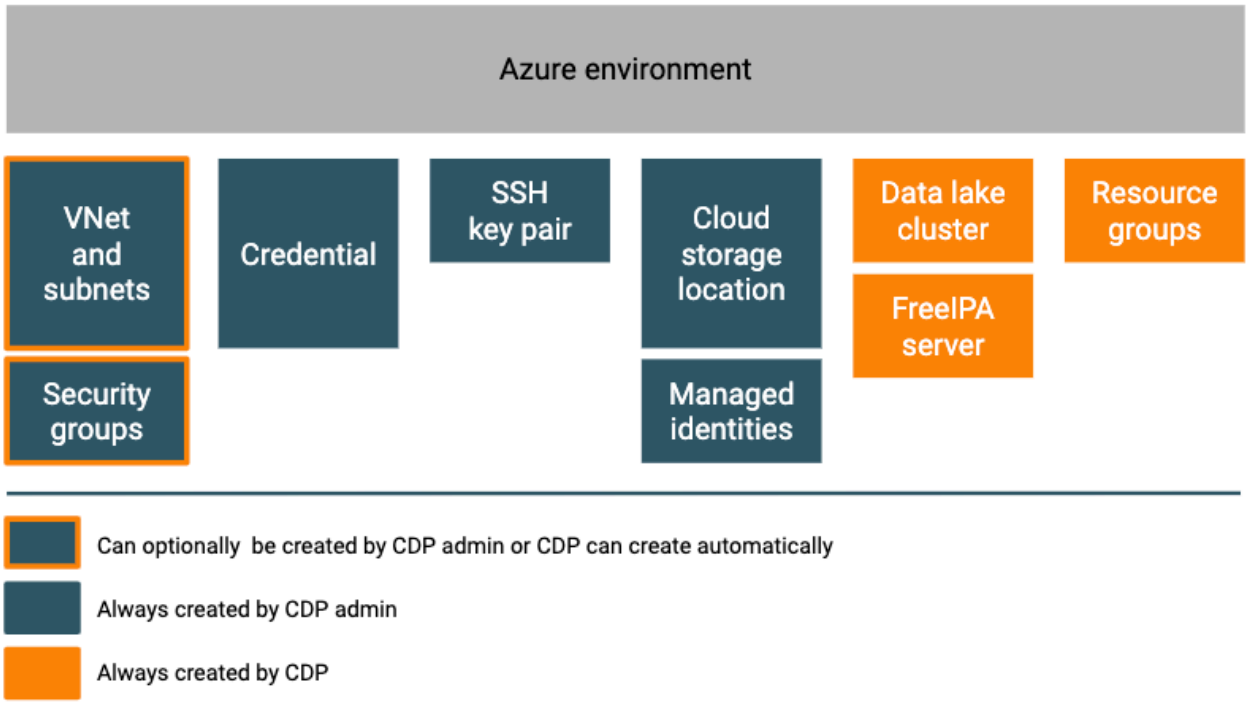

The diagram illustrates all major user-created and Cloudera-created components of an environment:

- The items in dark blue boxes with orange outlines can either be automatically provisioned by Cloudera on your Azure account, or you can optionally pre-create them and specify them when registering an environment.
- The items in dark blue boxes must be pre-created by your Cloudera administrator prior to environment registration and then specified when registering an environment.
- The items in orange boxes are automatically provisioned on Azure by Cloudera as part of environment provisioning.

Note: The items that are user-created are not terminated during environment deletion.

Information that describes how to register an Azure environment with Cloudera is linked to at the end of this page.

Related Information

Azure account requirements

Azure credentials

How to register Azure environments

Custom image repositories

If your organization must control the acquisition and provisioning of images in your cloud account custom image repositories are available. If your organization does not allow internet access, or restricts image repositories to only those within your virtual private network (VPC in AWS or VNet in Azure), you can bring your own repository to Cloudera.

Using your custom repository, you can host and scan Cloudera Data Warehouse images. You gain complete control over which images are provisioned in your cloud account and how you acquire the images. You can choose to verify, or not, that the image targeted by a query is present.

The following registries are supported for use as a custom image repository in Cloudera Data Warehouse:

- Azure Container Registry (ACR)
- Amazon Elastic Container Registry (ECR)
- Docker registry V2

To use this feature, you must obtain the CDP_CUSTOM_REPO entitlement.

Cloudera Data Warehouse architecture

The following diagram shows the Cloudera Data Warehouse architecture using a custom image repository based on ACR. The architecture is similar for ECR-based repositories.

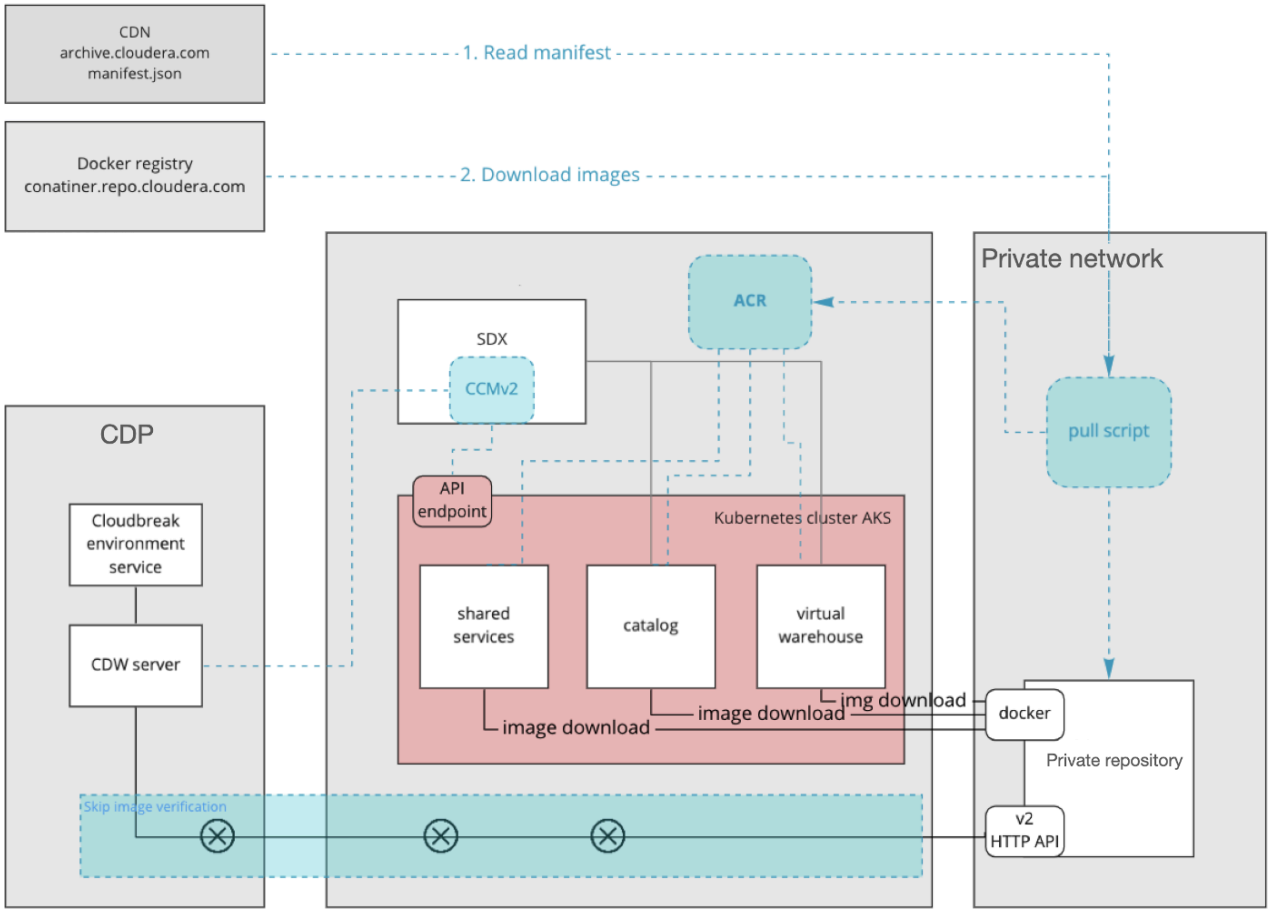

When you set up a custom repository, you must read the manifest.json file using your paywall credentials from Cloudera to obtain a list of images. Cloudera provides an image list with each release. You must pull images each time a new release of Cloudera Data Warehouse occurs. It is recommended that you create an automated way to pull these images.

You can set the Disable image version verification option. You might not set this option, thereby enabling verification, for networking or security reasons. You might set this option to disable verification if you need time to onboard new images. The option works as follows:

- If you do not set this option, Cloudera attempts to verify the images in your repository, and if the latest version of the images is not available in the registry, some filtering occurs, and old images are started. Cloudera Data Warehouse does not fully support starting older images, so pulling new versions of images promptly is recommended.
- If you do set this option, when the new releases occur, the server starts the new images without verification. If the images are not present in the registry, provisioning fails.

Limitations

- Image namespaces must not include arbitrary paths, such as .../mycompany/myhierarchy/hive:… Only the following namespace format is allowed:

  ```
  container.repo.cloudera.com/cloudera/hive:…
  ```

- CLI integration is not available.
- Azure private registry support is available, but image verification is disabled. In this release, Cloudera does not support a repository that is set up as a private registry from the networking perspective.

Downloading images

Learn how to download Cloudera Data Warehouse container images from the registry. You can run a security scan and approve (or block) each image before copying the image into your repository

Before you begin

Obtain the following entitlements:

- Salesforce entitlement
- CDP_CUSTOM_REPO entitlement

Procedure

1. Login to Cloudera container registry (https://container.repo.cloudera.com) using your Salesforce entitlement token and password.
   For example:

   ```
   $ docker login container.repo.cloudera.com -u 634ae2db-f926-4530-842c-95
   d3401f37e5
   ```

2. Download container images using the docker pull command. For example:

   ```
   docker pull container.repo.cloudera.com/cloudera/hive:7.2.3.0-22
   ```

   For example:

   ```
   Pulling from cloudera/hive Digest:sha256:29c635b8612c770c8089d4ab134c10e
   599dcf61553dfa4147975d6af33c835a9
   Status: Downloaded newer image for container.repo.cloudera.com/cloudera/hi
   ve:7.2.3.0-22container.repo.cloudera.com/cloudera/hive:7.2.3.0-22
   ```

Setting up a custom repository

When you activate the environment from Cloudera Data Warehouse, you can choose to use a custom repository.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Environments tab, locate the environment you want to activate, and click Activate.
   The **Activate Environment** modal is displayed.
3. Select the Use Custom repository option.
   Depending in your custome repository type, input ecr, acr, or docker in the Registry Type field.
4. Specify the URL of your custom repository.
5. Fields for your user name and password appear.
   You do not need to enter these credentials.
6. Click Activate.

Handling Cloudera Data Warehouse upgrades

About this task

You must update your custom repository to contain New Cloudera Data Warehouse images when available before you upgrade your Virtual Warehouse or Database Catalog. The Virtual Warehouse UI indicates when an upgrade

is available. If the new images are not available in your custom repository the upgrade fails. Currently there is no rollback option available.

If all required images for the new version of a Virtual Warehouse or Database catalog are not in the repository, Cloudera Data Warehouse falls back to a previous version. You can then create a Virtual Warehouse based on the earlier version of Cloudera Data Warehouse.

Disabling image verification

About this task

By default, when you authorize Cloudera Data Warehouse to run a query on your custom repository, Cloudera Data Warehouse performs the following verifications:

- Searches tags in repository to verify image availability.
- Populates the image registry secrets on the Kubernetes namespaces for registry type docker if credentials are required.
- Fills in the registry address in the Helm chart template.

Cloudera Data Warehouse Public Cloud

Getting Started with Public Cloud

Date published: 2024-01-01
Date modified: 2024-12-05

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Getting started in Cloudera Data Warehouse.....4
Prerequisites.....5
Granting Cloudera users and groups access to Cloudera Data Warehouse.....5
Creating your first Virtual Warehouse.....6
Set up SQL AI Assistant.....6
Prerequisites for configuring Hue SQL AI Assistant.....7
Configure SQL AI Assistant using Cloudera AI Workbench.....8
Configure SQL AI Assistant using the Cloudera AI Inference service.....9
Configure SQL AI Assistant using the Microsoft Azure OpenAI service.....9
Configure SQL AI Assistant using the Amazon Bedrock Service.....10
Configure SQL AI Assistant using the OpenAI platform.....10
Complete list of model-related configurations for setting up the Hue SQL AI Assistant.....11

Getting started in Cloudera Data Warehouse

In Cloudera Data Warehouse, you can deploy and administer cloud data warehousing. You can move workloads to the cloud securely and manage user access to warehouse data, or just parts of the data, through Apache Ranger fine-grained control. You learn how to get started in Cloudera Data Warehouse.

As Cloudera Data Warehouse administrator, you need privileges to access a Cloudera environment. As a Cloudera Data Warehouse user, you need privileges to access warehouse data as described in the subtopics, *Prerequisites* and *Granting Cloudera users and groups access*.

To get started, click Data Warehouse on the Cloudera home page. The **Overview** page is displayed:

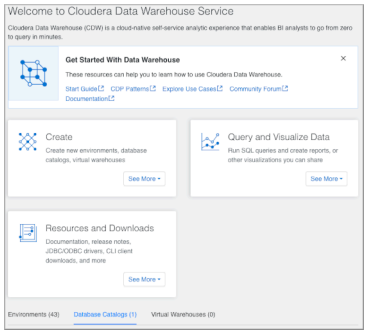

If you have permission to access an environment, you can create a Virtual Warehouse:

- A Hive Virtual Warehouse for Hive users
- An Impala Virtual Warehouse for Impala users
- Unified Analytics-enabled Virtual Warehouse.

  Unified Analytics offers Hive/Impala-equivalent SQL syntax for many SQL queries, plus optimizations and enhancements.

You provide users with the URL of your Virtual Warehouse, and they can access the data you authorize using tools you download from **Resources and Downloads** and provide to them from **Resources and Downloads**, you can install or download the following software or drivers:

- DBT Hive and DBT Impala

  Adapters for using dbt with Hive or Impala.
- CDP CLI

  Install a tool to help you manage your Service Manager-managed cluster instances. The CLI can be used for automating cluster creation and termination.
- Hive JDBC Jar

  Download the Hive JDBC driver for clients to connect to a Virtual Warehouse.
- Beeline CLI

  Download the Beeline CLI tarball for clients to install and connect to a Virtual Warehouse.
- Workload Insights

  Download this software for getting recommendations for creating materialized views.
- Impala JDBC/ODBC Driver

  Download the Impala JDBC/ODBC driver for clients to connect to a Virtual Warehouse.
- Unified Analytics JDBC Driver

  Download a JDBC driver for clients to connect to a Unified Analytics Virtual Warehouse.

Exiting Cloudera Data Warehouse

In the left navigation, click ⠿ as shown in the following image:

Select Management Console to leave Cloudera Data Warehouse to manage environments, users, and Cloudera services, or select a different menu item.

Prerequisites

To get started in Cloudera Data Warehouse, your data must conform to supported compression codecs, and you must obtain Cloudera resource roles to grant users access to an environment. Users can then get started on tasks, such as activating the environment from Cloudera Data Warehouse.

Unsupported compression

Cloudera Data Warehouse does not support LZO compression due to licensing of the LZO library. You cannot query tables having LZO compression in Virtual Warehouses, which use Cloudera Data Warehouse Impala or Hive LLAP engines.

Cloudera resource roles

You need to have the PowerUser role to change the following Cloudera resource roles that are associated with the Cloudera Data Warehouse service.

- DWAdmin: This role enables users or groups to grant a Cloudera user or group the ability to activate, terminate, launch, stop, or update services in Virtual Warehouses.
- DWUser: This role enables users or groups to view and use Cloudera Data Warehouse clusters (Virtual Warehouses) that are associated with specific environments.

A Cloudera PowerUser assigns these roles to users who require access to the Virtual Warehouses that are associated with specific environments. A default Database Catalog is associated with your environment. After granting these roles to users and groups, they then have Single Sign-On (SSO) access to the Virtual Warehouses.

Granting Cloudera users and groups access to Cloudera Data Warehouse

You assign the DWAdmin and DWUser roles to users and groups who must use or manage Virtual Warehouses in Cloudera Data Warehouse service on cloud environments. These roles are not for accessing Kubernetes endpoints in the cluster.

Procedure

1. Log in to the Cloudera web interface.
2. In the left navigation menu, click Environments.
3. Search for the environment you want to grant access to, and then click the environment name.
4. In the environment's Clusters page, click the Actions drop-down menu on the right, and select Manage Access.
5. In the Access page, in the Select group or user text box, type the name of the user or group you want to add, and then click that user or group's name.
6. In Update Resource Roles for…, select the DWAdmin or DWUser or both roles, and then click Update Roles. You should receive a success message.
7. Click the environment name at the top of the page to navigate back to the Clusters page.
8. In the Clusters page, click Actions, and select Synchronize Users to FreeIPA.

   Depending on how many users have access to the environment, this synchronization process can take a few seconds or a few minutes.

Results

The users and groups you have granted the DWAdmin or DWUser resource roles to can now use Cloudera Data Warehouse service with the environment.

Creating your first Virtual Warehouse

After meeting prerequisites, you learn how to activate the environment from Cloudera Data Warehouse. You then create a Virtual Warehouse.

Before you begin

Before you can create a Virtual Warehouse in the Cloudera Data Warehouse service, you must have completed the following tasks:

- Register an environment with Cloudera Management Console. This sets up your Data Lake. Depending on which cloud platform you are using, see Register an AWS environment or Register an Azure environment.
- Grant the DWAdmin role to the user or group that needs to create a Virtual Warehouse. This user or group determines which environment and Data Lake your Virtual Warehouse uses. See Granting Cloudera users access to Cloudera Data Warehouse service.

Procedure

1. Activate the environment that uses the Data Lake. See Activating AWS environments or Activating Azure environments.
   A default Database Catalog is created.
2. Add a new Virtual Warehouse. See Creating a Virtual Warehouse.

Results

You can start executing workloads in the new Virtual Warehouse.

About setting up the Hue SQL AI Assistant

Administrators are required to set up and enable the SQL AI Assistant before analysts can use it to generate, edit, optimize, explain, and fix queries using natural language in Hue.

First, you must obtain clearance from your organization’s infosec team to ensure it is safe to use the SQL AI Assistant because some of the table metadata and data, as mentioned in the previous section, is shared with the LLM.

Next, select and prepare one of the following AI services of your choice for hosting an LLM, and then configure the SQL AI Assistant in Hue:

- Cloudera AI Workbench
- Cloudera AI Inference service
- Microsoft Azure OpenAI service
- Amazon Bedrock service
- OpenAI platform

Prerequisites for configuring Hue SQL AI Assistant

To configure the SQL AI Assistant in Hue, you must pass the token required for connecting to the LLM service. Learn about the open and secure approaches to pass the tokens, and use the one that fits your organization policy.

(Recommended) Secure approach for passing a token

In this approach, you use Kubernetes' method of distributing secrets. You first encode the credentials and then add the encoded bit as a data item in the HUE_AI_INTERFACE_TOKEN property. The token becomes available in the Hue pod as an environment variable.

About this task

Note: Secrets are lost when you rebuild the Virtual Warehouse. You need to redo this step to continue using encoded credentials.

Procedure

1. Use a base64 encoding tool to convert your token to a base-64 representation by running the following command:

   ```
   echo -n '[***MY-TOKEN***]' | base64
   ```

   Replace *[***MY-TOKEN***]* with the token value you want to encode.
2. Open a terminal session and run the following command to add the encoded secret:

   ```
   kubectl edit secret hue-secret -n [***VIRTUAL-WAREHOUSE-NAMESPACE***]
   ```

   # Replace *[***VIRTUAL-WAREHOUSE-NAMESPACE***]* with the actual Virtual Warehouse ID (same as the namespace) in which you want to add the secret.
3. Add the encoded value returned for your token in the HUE_AI_INTERFACE_TOKEN property as follows:

   ```
   ...
   apiVersion: v1
   data:
     HADOOP_CREDSTORE_PASSWORD: [***ENCODED-HADOOP-CREDSTORE-PASSWORD***]
     HUE_AI_INTERFACE_TOKEN: [***ENCODED-TOKEN-VALUE***]
   kind: Secret
   ```

   Replace *[***ENCODED-TOKEN-VALUE***]* with the actual encoded value returned for your token.

Open approach for passing a token

In this approach, you specify the token value in the hue-safety-valve field in Cloudera Data Warehouse. The credentials are saved in a configuration file in the plain text format.

Note: Cloudera recommends that you use the open approach to pass tokens in test deployments, for proof of concept use cases. Use the Secure approach for passing a token in production deployments.

Here's a list of the open token values in the hue-safety-valve field to configure the SQL AI Assistant:

For Open token

Microsoft Azure OpenAI

```
[desktop]
[[ai_interface]]
    service='azure'
    model_name='[***DEPLOYMENT-NAME***]'
    base_url="https://[***RESOURCE***].openai.azure.com/"
    token="[***RESOURCE-KEY***]"
```

AWS

```
[aws]
 [[bedrock_account]]
  access_key_id='[***ACCESS-KEY***]'
  secret_access_key='[***SECRET-KEY***]'
  region='us-east-1'
[desktop]
 [[ai_interface]]
  service='bedrock'
  model='claude'
```

OpenAI

```
[desktop]
 [[ai_interface]]
  service='openai'
  token='[***API-KEY***]'
```

Configure SQL AI Assistant using Cloudera AI Workbench

This topic describes how to deploy and configure the SQL AI Assistant using the Cloudera AI Workbench. With the added support for Cloudera AI Workbench, you can securely deploy and run your own models within a virtual private cloud. This self-contained integration offers enhanced control and privacy within your environment.

Before you begin

To know more about creating and deploying models using Cloudera AI Workbench, see Create and deploy the model.

Procedure

1. Upon succesuccessful completion of model deployment, log in to the Cloudera Data Warehouse service.
2. Go to the Virtual Warehouses tab, locate the Virtual Warehouse on which you want to enable this feature, and click ⋮ Edit .
3. Go to Configurations Hue , select hue-safety-valve from the Configuration files drop-down menu, and add the following lines :

```
[desktop]
 [[ai_interface]]
    service='cml'
    model='llama'
    model_ref='[***Place model access key here***]'
    base_url='https://[***RESOURCE***].cloudera.site/model'
```

4. Click Apply Changes.

Results

You see Assistant on the Hue SQL editor, where the SQL AI Assistant utilizes the model hosted in the Cloudera AI Workbench.

Configure SQL AI Assistant using the Cloudera AI Inference service

This topic describes configuring the SQL AI Assistant using the Cloudera AI Inference service.

Before you begin

To know more about installing and setting up the Cloudera AI Inference service, see Prerequisites for setting up the Cloudera AI Inference service.

Procedure

1. Upon installing and setting up the Cloudera AI Inference service, log in to the Cloudera Data Warehouse service.
2. Go to the Virtual Warehouses tab, locate the Virtual Warehouse on which you want to enable this feature, and click Edit .
3. Go to Configurations Hue , select hue-safety-valve from the Configuration files drop-down menu, and add the following lines :

```
[[ai_interface]]
    service='caii'
    model_name='[***Place MODEL name here***]'
    base_url="https://[***RESOURCE***]/v1"
```

4. Click Apply Changes.

Results

You see Assistant on the Hue SQL editor, where the SQL AI Assistant utlizes the model hosted inCloudera AI Inference service.

Configure SQL AI Assistant using the Microsoft Azure OpenAI service

Microsoft Azure allows for dedicated deployments of OpenAI GPT models. You can use Azure's OpenAI service instead of the publicly hosted OpenAI APIs, as it enables data processing within your Azure Virtual Network (VNet) network. GPT models can also be integrated with the Hue SQL AI Assistant using Azure's OpenAI service.

Before you begin

Obtain a Microsoft Azure subscription by working with your organization's IT team and registering for access to the Azure OpenAI service. For more information, see Create and deploy an Azure OpenAI Service resource.

Procedure

1. Log in to the Cloudera Data Warehouse service
2. Go to the Virtual Warehouses tab, locate the Virtual Warehouse on which you want to enable this feature, and click Edit .
3. Go to Configurations Hue , select hue-safety-valve from the Configuration files drop-down menu, and add the following lines :

```
[desktop]
[[ai_interface]]
    service='azure'
```

```
    model_name='[***DEPLOYMENT-NAME***]'
    base_url="https://[***RESOURCE***].openai.azure.com/"
```

4. Click Apply Changes.

Results

You see ✨ Assistant on the Hue SQL editor, and the SQL AI Assistant will connect to the specified model on the Microsoft Azure OpenAI service.

Configure SQL AI Assistant using the Amazon Bedrock Service

This topic describes how to configure the SQL AI Assistant using the Amazon Bedrock Service.

Before you begin

You must have an AWS account with Bedrock access. For more information on accessing keys, see Amazon Bedrock.

Procedure

1. Log in to the Cloudera Data Warehouse service.
2. Go to the Virtual Warehouses tab, locate the Virtual Warehouse on which you want to enable this feature, and click ⋮ Edit .
3. Go to Configurations Hue , select hue-safety-valve from the Configuration files drop-down menu, and add the following lines :

```
[aws]
  [[bedrock_account]]
    access_key_id_script='echo $AWS_BEDROCK_ACCESS_KEY_ID'
    secret_access_key_script='echo $AWS_BEDROCK_SECRET_ACCESS_KEY'
    region='us-east-1'
[desktop]
  [[ai_interface]]
    service='bedrock'
    model='claude'
```

   AWS_BEDROCK_ACCESS_KEY_ID and AWS_BEDROCK_SECRET_ACCESS_KEY must be added as encoded values under hue-secret. For more information, see Secure approach for passing a token.
4. Click Apply Changes.

Results

You see ✨ Assistant on the Hue SQL editor, and the SQL AI Assistant will connect to the specified model in the Amazon Bedrock service.

Configure SQL AI Assistant using the OpenAI platform

This topic describes how to set up SQL AI Assistant and connect to a model on the OpenAI platform.

Before you begin

You must have created an account with the OpenAI platform.

Procedure

1. Log in to the Cloudera Data Warehouse service

2. Go to the Virtual Warehouses tab, locate the Virtual Warehouse on which you want to enable this feature, and click ⋮ Edit .
3. Go to Configurations Hue , select hue-safety-valve from the Configuration files drop-down menu, and add the following lines :

```
[desktop]
  [[ai_interface]]
    service='openai'
```

   You can specify the model_name (optional) and define the model. If no model is defined, the default model (gpt-3.5-turbo-16k) will be used.
4. Click Apply Changes.

Results

You see ✨ Assistant on the Hue SQL editor, and the SQL AI Assistant will connect to the specified model on the OpenAI platform.

Complete list of model-related configurations for setting up the Hue SQL AI Assistant

Review the list of service, model, and semantic search-related configurations used for custom configuring the AI services and models you want to use with the SQL AI Assistant and how to specify them in the Hue Advanced Configuration Snippet in the Cloudera Data Warehouse web interface.

List of service and model-related configurations

IN Cloudera Data Warehouse, you can configure by going to Virtual Wharehouse Configurations Hue hue-safety-valve . The following template shows the base structure of the configurations and adding the following lines:

```
[desktop]
 [[ai_interface]]
  [***CONFIG-KEY1***]='[***VALUE***]'
  [***CONFIG-KEY2***]='[***VALUE***]'
 [[semantic_search]]
  [***CONFIG-KEY1***]='[***VALUE***]'
  [***CONFIG-KEY2***]='[***VALUE***]'
```

AI interface-related configurations

Here is the complete list of configurations under [[ai_interface]], which allows you to specify the service and model to be used:

AI interface config key	Description
service	API service to be used for AI tasks. AI is disabled when a service is not configured. For exam Workbench and Cloudera AI Inference service are API services.
service_version	API service version to be used for AI tasks
trusted_service	Indicates whether the LLM is trusted or not. Turn on to disable the warning. The default value
model	The AI model you want to use for AI tasks. For example, gptand llama.
model_name	The fully qualified name of the model to be used. For example, gpt-3.5-turbo-16k.
model_ref	The `model_ref` is a placeholder for adding the access key of the specific model you want to u
base_url	Service API base URL.
add_table_data	When enabled, sample rows from the table are added to the prompt. The default value is True.
table_data_cache_size	Size of the LRU cache used for storing table sample data.

AI interface config key	Description
auto_fetch_table_meta_limit	Number of tables to load from a database, initially.
token	Service API secret token.
token_script	Provides a secure way to get the service API secret token.
enabled_sql_tasks	A comma-separated list of SQL-related AI tasks available in the Editor.

Semantic search-related configurations

Specify the semantic search-related configurations used for RAG under the [[semantic_search]] section, as listed in the following table:

Semantic search config key	Description
relevancy	The technology you want to use for semantic search. Acceptable values are vector_search or v
embedding_model	The model you want to use for data-embedding. This must be compatible with SentenceTransf
top_k	Number of top-ranking items returned by semantic search.
cache_size	Size of the LRU cache used for storing embedding.

Cloudera Data Warehouse Public Cloud

Managing Virtual Warehouses

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Virtual Warehouse IP address and cloud resource requirements for Public Cloud environments.....5

IP address and cloud resource requirements for Virtual Warehouses running on AWS environments..... 5
IP address and cloud resource requirements for Virtual Warehouses running on Azure environments..... 7

Creating a Virtual Warehouse.....9

Compute Instance Types..... 10
Virtual Warehouse sizing requirements for public cloud environments.....11
How to size a Virtual Warehouse.....14
Auto-suspend timeout in Cloudera Data Warehouse.....14
Configuring auto-scaling..... 15
Configuring Hive VW concurrency auto-scaling.....15
Configuring Hive query isolation auto-scaling.....16
Configuring Impala Virtual Warehouse auto-scaling..... 17
Configuring Workload Aware Autoscaling..... 18
Configuring Impala coordinator shutdown..... 20
Configuring Impala coordinator high availability.....22
Configuring Impala catalog high availability..... 25
Configuring Impala Statestore high availability..... 26
Impala JDBC/ODBC host names.....26

Creating a Virtual Warehouse on ARM instances.....27

Configuring a Virtual Warehouse..... 27

Tuning Virtual Warehouses.....28
Tuning Impala auto-scaling.....28
Tuning Impala Java tool options.....29
Disabling metadata synchronization.....29
Disabling Impala catalog high availability..... 30
Enabling SSO to a Virtual Warehouse..... 30
Configuring token-based authentication.....32
Generating a JWT token from the Cloudera side..... 32
Configuring the Impala Virtual Warehouse for authentication.....35
Configuring Impyla for authentication.....35
Configuring the Hive Virtual Warehouse for authentication.....36
Getting a JDBC URL for your Virtual Warehouse..... 37
Authenticating from a JDBC client.....37
Connecting Cloudera Data Warehouse and Cloudera AI..... 38
Configuring a time zone.....38
Configuring UDF JAR caching in Hive Virtual Warehouse.....39
Connecting Cloudera Data Warehouse and Kudu..... 40
Gathering Kudu master FQDNs.....40
Configuring a Cloudera Data Warehouse-Kudu connection.....42
Setting up a past release for a Cloudera Data Warehouse-Kudu connection..... 43
Configuring Iceberg manifest caching in Impala Virtual Warehouse..... 45
Managing high partition workloads..... 47

Configuring an Operational Database connection.... 48
Setting up Cloudera Data Warehouse to connect to Cloudera Operational Database....48

Resizing the Impala Virtual Warehouse size....49

Resizing the Hive Virtual Warehouse size....49

Compaction in Cloudera Data Warehouse.... 50
Compaction prerequisites....50
Change compactor configuration for Hive Virtual Warehouses on Cloudera Data Warehouse Public Cloud....51
Initiating automatic compaction.... 51
Configuring compaction.... 52
Automating metadata invalidation after compaction.... 52
How compaction works....53
Compactor processes.... 53
How compaction interacts with the Data Lake....53
Cloudera Data Warehouse Public Cloud Compaction Architecture.... 54

Compaction observability.... 54
Viewing a compaction alert using Grafana....55
Adding a compaction alert.... 56
Managing a compaction alert.... 57
Configuring alert notifications....58
Configuring alert notifications, no UI....59
Checking an alert notification configuration....61
Deleting a compaction alert.... 62

Data Visualization in Cloudera Data Warehouse.... 62

Rebuilding a Database Catalog....63
Backing up and restoring Hue documents.... 64

Rebuilding a Virtual Warehouse.... 65

Upgrading Database Catalogs and Virtual Warehouses in Cloudera Data Warehouse Public Cloud.... 66

Debugging with Impala Web UIs.... 68

Virtual Warehouse IP address and cloud resource requirements for Public Cloud environments

Learn about how many IP addresses and cloud resources are required to run Virtual Warehouses efficiently in public cloud environments for Cloudera Data Warehouse Public Cloud.

IP address and cloud resource requirements for Virtual Warehouses running on AWS environments

Learn about the estimated number of IP addresses and cloud resources required to run Virtual Warehouses on AWS environments for Cloudera Data Warehouse Public Cloud.

Important: These requirements are estimated. Your particular workloads and configurations can affect the number of IP addresses and cloud resources required to run Virtual Warehouses efficiently. In addition, these requirements are for AWS environments that use the default AWS VPC Container Networking Interface (CNI) plugin. To reduce the number of required IP addresses, you can enable the overlay networking feature for AWS environments in . For further details, see the link to "Overlay networks for AWS environments in " at the bottom of this page.

Virtual Warehouse requirements:

Each compute node in a Virtual Warehouse that runs on AWS environments requires 8 IP addresses. Each executor needs one compute node, so the size of your Virtual Warehouse contributes to the number of IP addresses required. To calculate the number of IP addresses required for custom sizes, multiply the number of executors by 8 and add for the shared services as specified in the following sections.

Shared services requirements for Database Catalogs:

Virtual Warehouses also require shared services for the Database Catalog.

Additional shared services requirements:

The usage for other shared services is different for Hive Virtual Warehouses and Impala Virtual Warehouses. Here are the different requirements for Hive versus Impala Virtual Warehouses:

- Hive Virtual Warehouses add 1 compute node for each executor and 1 shared services node for the HiveServer for each Virtual Warehouse.
- Impala Virtual Warehouses add 1 compute node for each executor, 1 or 2 compute nodes for the coordinator, depending on the HA configuration, and 1 shared services node per Virtual Warehouse for Impala catalogd.

The following tables summarize the approximate number of IP addresses and cloud resources you should plan for Virtual Warehouses on AWS environments.

Table 1: Hive Virtual Warehouses running on AWS environments

Size (# executors)	# Compute nodes for executors	# Shared services nodes Database Catalog	# Shared services nodes for HiveServer	Total IP addresses required
XSMALL (2)	2	3	1	Executor nodes: 2 nodes X 8 = 16 Shared services nodes: 3 + 1 = 4 nodes X 25 = 100 TOTAL = ~116 IP addresses

Size (# executors)	# Compute nodes for executors	# Shared services nodes Database Catalog	# Shared services nodes for HiveServer	Total IP addresses required
SMALL (10)	10	3	1	Executor nodes: 10 nodes X 8 = 80 Shared services nodes: 3 + 1 = 4 nodes X 25 = 100 TOTAL = 180 IP addresses
MED (20)	20	3	1	Executor nodes: 20 nodes X 8 = 160 Shared services nodes: 3 + 1 = 4 nodes X 25 = 100 TOTAL = 260 IP addresses
LARGE (40)	40	3	1	Executor nodes: 40 nodes X 8 = 320 Shared services nodes: 3 + 1 = 4 nodes X 25 = 100 TOTAL = 420 IP addresses

Table 2: Impala Virtual Warehouses running on AWS environments

Size (# executors)	# Compute nodes for executors	# Compute nodes for coordinator	# Shared services nodes for Impala catalogd	# Shared services nodes for Database Catalog	Total IP addresses required
XSMALL (2)	2	1-2	1	3	Executor/ coordinator nodes: 3-4 nodes X 8 = 24-32 Shared services nodes: 1 + 3 = 4 nodes X 25 = 100 TOTAL = 124 to 132 IP addresses
SMALL (10)	10	1-2	1	3	Executor/ coordinator nodes: 11-12 nodes X 8 = 88-96 Shared services nodes: 1 + 3 = 4 nodes X 25 = 100 TOTAL = 188 to 196 IP addresses
MED (20)	20	1-2	1	3	Executor/ coordinator nodes: 21-22 nodes X 8 = 168-176 Shared services nodes: 1 + 3 = 4 nodes X 25 = 100 TOTAL = 168 to 176 IP addresses
LARGE (40)	40	1-2	1	3	Executor/ coordinator nodes: 41-42 nodes X 8 = 328-336 Shared services nodes: 1 + 3 = 4 nodes X 25 = 100 TOTAL = 428 to 436 IP addresses

Related Information

Overlay networks for AWS environments in Cloudera Data Warehouse

IP address and cloud resource requirements for Virtual Warehouses running on Azure environments

Learn about the estimated number of IP addresses and cloud resources required to run Virtual Warehouses on Azure environments for Cloudera Data Warehouse Public Cloud.

Important: These requirements are estimated. Your particular workloads and configurations can affect the number of IP addresses and cloud resources required to run Virtual Warehouses efficiently. In addition, these requirements are for Azure environments that use the default Azure Container Networking Interface (CNI) plugin. To reduce the number of required IP addresses, you can enable the kubenet networking feature for Azure environments in .

Virtual Warehouse requirements:

Each compute node in a Virtual Warehouse that runs on Azure environments requires 16 IP addresses (for pods and for the node itself). Each executor needs one compute node, so the size of your Virtual Warehouse contributes to the number of IP addresses required. To calculate the number of IP addresses required for custom sizes, multiply the number of executors by 16 and add for the shared services as specified in the following sections.

Shared services requirements for Database Catalogs:

Virtual Warehouses also require shared services for Database Catalogs.

Additional shared services requirements:

The usage for other shared services is different for Hive Virtual Warehouses and Impala Virtual Warehouses. Here are the different requirements for Hive versus Impala Virtual Warehouses:

- Hive Virtual Warehouses add 1 compute node for each executor and 1 shared services node for the HiveServer for each Virtual Warehouse.
- Impala Virtual Warehouses add 1 compute node for each executor, 1 or 2 compute nodes for the coordinator, depending on the HA configuration, and 1 shared services node per Virtual Warehouse for Impala catalogd.

The following tables summarize the approximate number of IP addresses and cloud resources you should plan for Virtual Warehouses on Azure environments.

Table 3: Hive Virtual Warehouses running on Azure environments

Size (# executors)	# Compute nodes for executors	# Shared services nodes for Database Catalog	# Shared services nodes for HiveServer	Total IP addresses required
XSMALL (2)	2	3	1	Executor nodes: 2 nodes X 16 = 32 Shared services nodes: 3 + 1 = 4 nodes X 31 = 124 TOTAL = 156 IP addresses
SMALL (10)	10	3	1	Executor nodes: 10 nodes X 16 = 160 Shared services nodes: 3 + 1 = 4 nodes X 31 = 124 TOTAL = 284 IP addresses

Size (# executors)	# Compute nodes for executors	# Shared services nodes for Database Catalog	# Shared services nodes for HiveServer	Total IP addresses required
MED (20)	20	3	1	Executor nodes: 20 nodes X 16 = 320 Shared services nodes: 3 + 1 = 4 nodes X 31 = 124 TOTAL = 444 IP addresses
LARGE (40)	40	3	1	Executor nodes: 40 nodes X 16 = 640 Shared services nodes: 3 + 1 = 4 nodes X 31 = 124 TOTAL = 764 IP addresses

Table 4: Impala Virtual Warehouses running on Azure environments

Size (# executors)	# Compute nodes for executors	# Compute nodes for coordinator	# Shared services nodes for Impala catalogd	# Shared services nodes for Database Catalog	Total IP addresses required
XSMALL (2)	2	1-2	1	3	Executor/ coordinator nodes: 3-4 nodes X 16 = 48-64 Shared services nodes: 1 + 3 = 4 nodes X 31 = 124 TOTAL = 172 to 188 IP addresses
SMALL (10)	10	1-2	1	3	Executor/ coordinator nodes: 11-12 nodes X 16 = 176-192 Shared services nodes: 1 + 3 = 4 nodes X 31 = 124 TOTAL = 300 to 316 IP addresses
MED (20)	20	1-2	1	3	Executor/ coordinator nodes: 21-22 nodes X 16 = 336-352 Shared services nodes: 1 + 3 = 4 nodes X 31 = 124 TOTAL = 460 to 476 IP addresses
LARGE (40)	40	1-2	1	3	Executor/ coordinator nodes: 41-42 nodes X 16 = 656-672 Shared services nodes: 1 + 3 = 4 nodes X 31 = 124 TOTAL = 780 to 796 IP addresses

Related Information

Overlay networks for Azure environments in Cloudera Data Warehouse

Creating a Virtual Warehouse

A Virtual Warehouse is an instance of compute resources in Public Cloud that is equivalent to an on-prem cluster. You learn how to create a new Virtual Warehouse in Cloudera Data Warehouse Public Cloud.

About this task

A Virtual Warehouse provides access to the data in tables and views in the data lake your Database Catalog uses. A Virtual Warehouse can access only the Database Catalog you select during creation of the Virtual Warehouse.

In this task and subtasks, you configure Virtual Warehouse features, including performance-related features for production workloads, such as the Virtual Warehouse size and auto-scaling. These features are designed to manage huge workloads in production, so if you are evaluating Cloudera Data Warehouse, or just learning, simply accept the default values. This task covers the bare minimum configurations.

In subtopics, you see details about how to configure features for production workloads, such as Hive query isolation and Impala catalog high availability.

Before you begin

- You obtained permissions to access a running environment for creating a Virtual Warehouse.
- You obtained the DWAdmin role to perform Cloudera Data Warehouse tasks.
- You logged into the Cloudera web interface.
- Your activated the environment from Cloudera Data Warehouse.

For more information about meeting these prerequisites, see Getting started in Cloudera Data Warehouse.

Procedure

1. Navigate to Data Warehouses Virtual Warehouses Create Virtual Warehouse .
2. In New Virtual Warehouse, specify a Name.

   Note: The fully qualified domain name of your Virtual Warehouse, which includes the Virtual Warehouse name plus the environment name must not exceed 64 characters; otherwise, Hue cannot load.

3. Select the Hive or Impala type of Virtual Warehouse you want.

   Virtual Warehouses can use Hive or Impala as the underlying SQL execution engine. Typically, Hive is used to support complex reports and enterprise dashboards. Impala is used to support interactive, ad-hoc analysis.
4. Select the Environment and Database Catalog for this Virtual Warehouse.
5. In AWS environments only, accept the default availability zone, or select an availability zone, such as us-east-1c.

   The default behavior is to randomly select an availability zone from the list of configured availability zones for the associated environment. Generally, it is fine to accept the default. All compute resources will run in the selected zone.

   Note: Selection of the zone is not an option in Azure environments.

6. Select the Compute Instance Types for the Virtual Warehouse based on your workload.

   For more information, see *Supported Compute Instance Types*.
7. Select the Hive Image Version or the Impala Image Version version, and the Hue Image version you want to use, or accept the default version (latest) at the top of the drop-down menus.
8. Select the Size of the Virtual Warehouse as described in the next subtopic.

   The “AutoSuspend Timeout” and “Concurrency Autoscaling” controls appear.
9. Configure auto-scaling as described in the subsequent subtopic.

10. In Authentication, select Enable SSO to enable single sign-on to your Virtual Warehouse, and in User Groups, select a user group set up in advance to access endpoints. If you do not have a user group set up for SSO, do not select Enable SSO.

    In Management Console User Management you set up a user group, required for enabling SSO, that identifies the users authorized to access to this Virtual Warehouse.
11. Enter keys and values for Tagging the Virtual Warehouse.
12. Accept default values for other settings, or change the values to suit your use case, and click Create Virtual Warehouse to create the new Virtual Warehouse.

    Click the tooltip for information about settings.

    When you create a Virtual Warehouse, a cluster is created in your cloud provider account. This cluster has two buckets. One bucket is used for managed data and the other is used for external data.

Related Information

Supported compute instance types

Custom tags in AWS environments

Custom tags in Azure environments

Compute Instance Types

Learn about the supported AWS and Azure compute instance types that you can select while creating a Virtual Warehouse.

Supported AWS compute instance types

Cloudera Data Warehouse supports the following AWS compute instance types (Hive and Impala executors):

Instance type	Processor	Usage	Virtual Warehouse Support
r7gd.4xlarge	ARM	Compute	Impala
r6gd.4xlarge	ARM	Compute	Impala
r6id.4xlarge	Intel	Compute	Hive and Impala
r5d.4xlarge	Intel	Compute (default)	Hive and Impala
r5ad.4xlarge	AMD	Compute	Hive and Impala
r5dn.4xlarge	Intel	Compute	Hive and Impala
m5.2xlarge	Intel	Shared services	Hive and Impala

In the Cloudera Data Warehouse environment, instances for shared service components are set up within a Kubernetes (K8s) cluster. The setup begins with three m5.2xlarge instances running the Cloudera Data Warehouse service, but the K8s cluster is capable of autoscaling, automatically adding more instances if necessary to handle increased demand. Additionally, an Amazon Relational Database Service (RDS) (db.r5.large) running PostgreSQL is created to store user metadata for Hue and Data Visualization services. In total, three shared db.r5.large nodes are used for this purpose. For more information, see Always active, shared services.

Supported Azure compute instance types

Cloudera Data Warehouse supports the following Azure compute instance types (Hive and Impala executors):

Azure VM	Processor Type	Usage	Virtual Warehouse Support
Standard_E16pds_v5	ARM	Compute	Impala
Standard_E16_v3	Intel	Compute	Hive and Impala
Standard_E16ds_v4	Intel	Compute (default)	Hive and Impala

Azure VM	Processor Type	Usage	Virtual Warehouse Support
Standard_E16ads_v5	AMD	Compute	Hive and Impala
Standard_E16ds_v5	Intel	Compute	Hive and Impala
Standard_D8s_v4	intel	Shared services (default)	Hive and Impala
Standard_D8as_v5	Intel	Shared services, used with AMD compute instance Standard_E16ads_v5	Hive and Impala

Three instances are added to the cluster as needed for shared services (always on components). Three shared nodes are dStandard_E2s_v3 MemoryOptimized using flexserver, for the Amazon Relational Database Service (RDS). These shared nodes are used for Hue and Data Visualization user metadata. For more information, see Always active, shared services.

Virtual Warehouse sizing requirements for public cloud environments

You need to understand how to estimate size requirements for Cloudera Data Warehouse Public Cloud Virtual Warehouses to create a Virtual Warehouse that meets your needs.

Calculating requirements for on-premises data warehouse deployments

Selecting the correct size of environment before you migrate your workloads from CDH and HDP to is critical for preserving performance characteristics. Consider the following workload characteristics when you plan for capacity on your environment:

- Query memory requirements
- CPU utilizations
- Disk bandwidth
- Working set size
- Concurrent query execution requirements

As part of this calculation, it is important to understand the core hardware difference between and on-premises hosts as explained in the following table:

Table 5: Hardware differences between and on-premises hosts

Hardware component	On-premises	AWS R5D.4xlarge instance	Azure E16 v3 instance
CPU cores	20 - 80	16	16
Memory	128 GB minimum 256 GB+ recommended	128 GB	128 GB
Network	10 Gbps minimum 40 Gbps recommended	Up to 10 Gbps	Up to 8,000 Mbps
Instance storage	12 x 2 TB drives (1,000 MB/s sequential)	2 x 300 GB NVMe SSD (1,100 MB/s sequential) **Note:** Optimally EBS volumes can be added to scratch and spool space for Impala.	400 GiB SSD
Persistent storage performance	At least 500 MB/s per disk With 20 disks, 10 GB/s per node	1,156 MB/s per EC2 instance	Maximum IOPS: 24,000 Maximum Read: 375 MB/s Maximum Write: 187 MB/s

An AWS R5D.4xlarge instance closely matches the CPU, memory, and bandwidth specifications that are recommended for CDH clusters. AWS R5D.4xlarge instance specifications are the default "instance type" for . AWS EBS storage cannot be used as primary database storage because it is transient and lacks sufficient capacity. This core difference makes it necessary to use a different strategy for than you used for CDH to achieve good scan performance.

Supported AWS instances

You can choose an instance type other than the default AWS R5D.4xlarge , and also a fallback instance type, using the CDP CLI. To fetch a string of available instance types, use the following CLI command:

```
cdp dw describe-allowed-instance-types
```

Example output:

```
{
     "aws": {
     "default": [
     "r5d.4xlarge"
     ],
     "allowed": [
     "r5d.4xlarge",
     ...
```

sizing and scaling

Before you migrate data to , plan for scaling and concurrency. In the cloud, scaling and concurrency can elastically respond to workload demands, which enables the system to operate at a lower cost than you might expect. If you configure your environment to accommodate peak workloads as a constant default configuration, you may waste resources and money when system demand falls below that level.

In , the size of the Virtual Warehouse determines the number of executor instances for an individual cluster so it determines memory limits and performance capabilities of individual queries:

Table 6: Virtual Warehouse sizes

Size	Number of executors
X-Small	2
Small	10
Medium	20
Large	40

Warehouse size in combination with auto-scaling settings determine how many clusters are allocated to support concurrent query execution.

The Virtual Warehouse size must be at least large enough to support the memory used by the most data-intensive query. Usually, the Virtual Warehouse size does not need to be larger that the most data-intensive query. Better caching is provided if there is commonality between data sets accessed by queries. Increasing the Virtual Warehouse size can increase single-user and multi-user capacity because additional memory and resources allows larger datasets to be processed. Concurrent query execution is also supported by sharing resources. If too small a size is configured for the Virtual Warehouse, poor data caching and memory paging can result. If too large a size is configured, excessive public cloud costs are incurred due to idle executors.

The primary difference between and CDH on-premises deployments when choosing a warehouse size based on existing hardware:

- With all resources on an executor are dedicated to query processing.

- With CDH on-premises deployments resources support other operations in addition to query processing. For example, these on-premises resources are shared with other services, such as HDFS or other locally hosted file systems. In particular Spark, HBase, or MapReduce. These other services might consume significant resources.

Consequently, you might be able to choose a much smaller Virtual Warehouse size in because resources are isolated in their own pod in the environment.

In the case of Impala, it is useful to look at the per-process metrics to isolate the impalad backend and the Impala front-end Java processes that hold the Catalog cache. In , the Impala coordinator and executor roles are separated leaving the unused Catalog JVM memory free to support query execution. You should look at the memory utilization metrics for executor-only impalad nodes (those not also running the coordinator role) to estimate how much memory your current cluster of Impala executors requires.

Concurrency

Concurrency is the number of queries that can be run at the same time. Determine the size you need by considering the amount of resources your system needs to support peak concurrency.

By default, Impala Virtual Warehouses can run 3 large queries per executor group. Executors can handle more queries that are simpler and that do not utilize concurrency on the executor. When you enable legacy multithreading, the Virtual Warehouse can run 12 queries per executor group. For most read-only queries the default setting of 3 queries per executor group is sufficient. Occasional peaks are handled transparently by the auto-scaling feature. When auto-scaling is triggered an additional executor group is added thereby doubling query concurrency capacity. Scaling the Virtual Warehouse by adding more clusters enables additional concurrent queries to run, but does not improve single-user capacity or performance. Concurrently executed queries are routed to the different clusters and execute independently. The number of clusters can be changed to match concurrent usage by changing the auto-scaling parameters. For more details about auto-scaling settings, see the links at the bottom of this page.

Important: The number of very large queries that can be run might also be impacted by memory limits on the executors.

For Hive on LLAP Virtual Warehouses, each size setting indicates the number of concurrent queries that can be run. For example, an X-Small Hive on LLAP Virtual Warehouse can run 2 concurrent queries. A Small Virtual Warehouse can run 10 concurrent queries. To run 20 concurrent queries in a Hive on LLAP Virtual Warehouse choose Medium size.

Caching "Hot Datasets"

Frequently accessed data is sometimes referred to as a "hot dataset." CDH supports caching mechanisms on the compute nodes to cache the working set that is read from remote file systems, such as remote HDFS data nodes, S3, ABFS, or ADLS. This offsets the input/output performance difference.

In , frequently accessed data is cached in a storage layer on SSD so that it can be quickly retrieved for subsequent queries. This boosts performance. Each executor can have up to 200 GB of cache. For example, a Medium-sized Virtual Warehouse can keep 200 * 20 = 4 TB of data in its cache. For columnar formats, such as ORC, data in the cache is decompressed, but not decoded. If the expected size of the hot dataset is 6 TB, which requires about 30 executors, you can over-provision by choosing a Large-sized warehouse to ensure full cache coverage. A case can also be made to under-provision by choosing a Medium-sized warehouse to reduce costs at the expense of having a lower cache hit rate. To offset this, keep in mind that columnar formats allow optimizations such as column projection and predicate push-down, which can significantly reduce cache requirements. In these cases, under-provisioning the cache might have no negative performance effect.

Scanned dataset size

Scanning large datasets on Amazon S3 or Azure ADLS can be slow. A single R5D.4xlarge EC2 instance can only scan data at 1,156 MB/s maximum throughput according to standard S3 benchmarks. If a query must read 100 GB from S3, S3 scanning takes a minimum of 88 seconds on just one node. Depending on the number of files in the S3 directory, it might take more than 3 minutes. In this case, if a query needs to scan 100 GB of data, if you use 10 nodes, you can get the scan time down to approximately 20 seconds.

However, keep in mind that with columnar storage, minimum/maximum statistics in files, and other conditions, often the amount of data read is significantly less than the total size of the files for certain queries.

Related Tasks

Tuning Impala auto-scaling

Related Information

Tuning Virtual Warehouses

How to size a Virtual Warehouse

When you create a Virtual Warehouse, you need to carefully set the size of your Virtual Warehouse. The size of the Virtual Warehouse you select during Virtual Warehouse creation determines the number of executors and concurrent queries the Virtual Warehouse can run.

Before creating a Virtual Warehouse and setting the Virtual Warehouse size, learn about critical "Warehouse sizing requirements". If you misconfigure the size of your Virtual Warehouse, you can use one of the following methods, depending on the Impala or Hive type of Virtual Warehouse, to correct the size:

- Impala: Edit your Virtual Warehouse configuration to correct the size.
- Hive: Delete and recreate the Virtual Warehouse to correct the size.

When you create a Virtual Warehouse, you select one of the following Virtual Warehouse sizes:

Virtual Warehouse Size	Number of Executors
XSMALL	2
SMALL	10
MEDIUM	20
LARGE	40
Custom	Enter a value between '1' and '100'

If you are evaluating Cloudera Data Warehouse, or just learning, XSMALL is the recommended size. For production workloads, choose a size based on the following factors:

- The number of executors you typically use for clusters in an on-premises deployment.
- The complexity of your queries and the size of the data sets that they access.

Large warehouses with more executors can cache more data than small warehouses. Caching enhances performance.

Auto-suspend timeout in Cloudera Data Warehouse

Auto-suspend enables you to handle resources when the auto-scaler has scaled back to the last executor group. You can control the time that the original warehouse executor group idles after all other groups scale down and release their executors. The JDBC endpoint lives on to respond to queries from the result cache or statistics, but expensive executors no longer run.

When the auto-scaler increases or decreases the number of executor groups in the Virtual Warehouse, it can take several minutes before the scaling up or down takes effect. This slight delay is caused by the time required by your cloud provider to provision clusters.

When no queries are sent to an executor group, resources scale down and executors are released. When all executor groups are scaled back, when executors are idle, and after a period of idle time (Auto-suspend Timeout), the Virtual Warehouse is suspended.

You set an Auto-suspend Timeout to configure how long a Virtual Warehouse idles before shutting down.

Auto-suspend timeout is independent of the auto-scaling process and only applies to the original Virtual Warehouse and not to any additional warehouses that are created as a result of auto-scaling.

If a query is executed on a suspended Virtual Warehouse, then the query coordinator queues the query. When a queued query is detected, an executor group is immediately added (a scale-up occurs) to run the query on the Virtual Warehouse.

This feature can help lower cloud costs.

Auto-suspend differs from manually stopping a Virtual Warehouse. Sending a query to a stopped Virtual Warehouse returns an HTTP 503 "no healthy upstream" error, no scale-up occurs, and the query does not execute.

Auto-suspend is enabled by default.

While creating a Virtual Warehouse, set Auto-suspend Timeout to the number of seconds you want the Virtual Warehouse to idle before shutting down. For example, if you set the timeout to 300 seconds, the Virtual Warehouse suspends itself after 300 seconds to save compute resource expense. The suspended Virtual Warehouse restarts as soon as you run a query.

To disable auto-suspend, select the Disable Auto-suspend option. You can disable the auto-suspend option both while creating a Virtual Warehouse or after creating it.

Configuring auto-scaling

When you create a Virtual Warehouse, you set auto-scaling to increase and decrease resources according to demand. Auto-scaling relinquishes resources when demand decreases to limit unnecessary cloud expenses. Auto-scaling increases resources to speed performance.

Auto-scaling is designed for huge workloads in production, so if you are evaluating Cloudera Data Warehouse, or just learning, simply accept the default values. Later, you can edit the Virtual Warehouse to tune auto-scaling.

Before configuring or tuning auto-scaling, you need to understand the auto-scaling process to make good configuration decisions. As a Hive Virtual Warehouse user, choose the method of auto-scaling and see the relevant configuration subtopic (below):

- Concurrency for BI-type queries
- Isolation for ETL-type queries

To configure auto-scaling in a Hive or Impala Virtual Warehouse, see the topics below:

Configuring Hive VW concurrency auto-scaling

Configuring the Hive Virtual Warehouse to use concurrency auto-scaling is critical for controlling cloud expenses.

Before you begin

- You are familiar with the auto-scaling process.
- You are creating a Hive Virtual Warehouse for running BI-type queries.
- In Cloudera Data Warehouse, you added a Hive Virtual Warehouse, configured the size of the Hive Virtual Warehouse, and configured auto-suspend as described in previous topics.
- You obtained the DWAdmin role.

About this task

In this task, first you select the number of executors for your virtual cluster.

- Minimum executors

  The fewest executors you think you will need to provide sufficient resources for your queries. The number of executors needed for a cloud workload is analogous to the number of nodes needed for an on-premises workload.
- Maximum number

  The maximum number of executors you will need. This setting limits prevents running an infinite number of executors and runaway costs.

Consider the following factors when configuring the minimum and maximum number of executors:

- Number of concurrent queries
- Complexity of queries
- Amount of data scanned by the queries
- Number of queries

Next, you configure when your cluster should scale up and down based on one of the following factors:

- Headroom: The number of available coordinators that trigger auto-scaling. For example, if Desired Free Capacity is set to 1 on an XSMALL-sized Virtual Warehouse, which has 2 executors, when there is less than one free coordinator (2 queries are concurrently executing), the warehouse auto-scales up and an additional executor group is added.
- Wait Time: How long queries wait in the queue to execute. A query is queued if it arrives on HiveServer and no coordinator is available. For example, if WaitTime Seconds is set to 10, queries are waiting to execute in the queue for 10 seconds. The warehouse auto-scales up and adds an additional executor group.

Auto-scaling based on wait time is not as predictable as auto-scaling based on headroom. The scaling based on wait time might react to non-scalable factors to spin up clusters. For example, query wait times might increase due to inefficient queries, not query volume.

When headroom or wait time thresholds are exceeded, the Virtual Warehouse adds executor groups until the maximum setting for Minimum and Maximum has been reached.

Procedure

1. In Concurrency Autoscaling properties, limit auto-scaling by setting the minimum and maximum number of executor nodes that can be added.

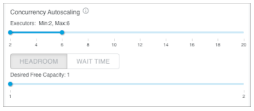

2. Choose when your cluster auto-scales up based on one of the following choices:
   - Headroom
   - Wait Time
3. Click Apply Changes.

Configuring Hive query isolation auto-scaling

You can configure your Hive Virtual Warehouse to add dedicated executors to run scan-heavy, data-intensive queries, also known as ETL queries. You learn how to enable a Virtual Warehouse auto-scaling feature and set a query isolation parameter in the Hive configuration.

Before you begin

- You are familiar with the auto-scaling process.
- You are creating a Hive Virtual Warehouse for running ETL-type queries.
- In Cloudera Data Warehouse, you added a Hive Virtual Warehouse, configured the size of the Hive Virtual Warehouse, and configured auto-suspend as described in previous topics.
- You obtained the DWAdmin role.

Note: You can enable query isolation only while creating a Virtual Warehouse. On existing warehouses, you can only disable the query isolation option.

About this task

In this task, first you configure the same auto-scaling properties as described in the previous topic Configuring Hive VW concurrency auto-scaling, and then you enable query isolation. Next, you set the hive.query.isolation.scan.size.threshold configuration parameter.

Procedure

1. In Concurrency Autoscaling properties, limit auto-scaling by setting the minimum and maximum number of executor nodes that can be added.

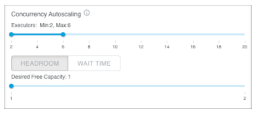

2. Choose when your cluster auto-scales up based on one of the following choices:
   - HEADROOM
   - WAIT TIME
3. In Concurrency Autoscaling, select Enable Query Isolation.

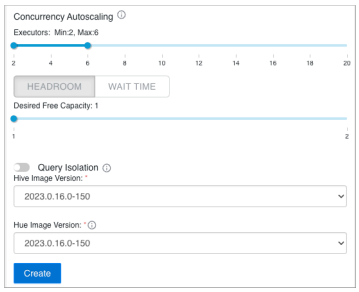

   Two configuration options appear: Max Concurrent Isolated Queries and Max Nodes Per Isolated Query.
4. In Max Concurrent Isolated Queries, set the maximum number of isolated queries that can run concurrently in their own dedicated executor nodes.

   Select this number based on the scan size of the data for your average scan-heavy, data-intensive query.
5. In Max Nodes Per Isolated Query, set how many executor nodes can be spawned for each isolated query.
6. Click Create Virtual Warehouse.
7. In the Cloudera Data Warehouse service **Overview** page, click Virtual Warehouses, and find your Virtual Warehouse. Click ⋮ Edit Configurations Hiveserver2 .
8. Select hive-site from the Configuration files drop-down menu and set hive.query.isolation.scan.size.threshold parameter to limit how much data can be scanned.

   For example, set 400GB.
9. Click Apply Changes.

Configuring Impala Virtual Warehouse auto-scaling

Configuring the Impala Virtual Warehouse to use concurrency auto-scaling is critical for controlling cloud expenses.

Before you begin

- You are familiar with the auto-scaling process.
- You are creating a Virtual Warehouse for running BI-type queries.
- In Cloudera Data Warehouse, you added an Impala Virtual Warehouse, configured the size of the Impala Virtual Warehouse, and configured auto-suspend as described in previous topics.
- You obtained the DWAdmin role.

About this task

In this task, you configure the following properties:

- Scale Up Delay: Sets the length of time in seconds to wait before adding more executors if queries wait in the queue.
- Scale Down Delay: Sets the length of time in seconds to wait before removing executors if executor groups are idle.

The time to auto-scale up or down is affected by the underlying Kubernetes configuration.

By default Impala Virtual Warehouses can run 3 large queries per executor group. Executors can handle more queries that are simpler and that do not utilize concurrency on the executor.

.

Procedure

1. In Scale Up Delay, set the seconds to wait before adding more executors if queries wait in the queue.

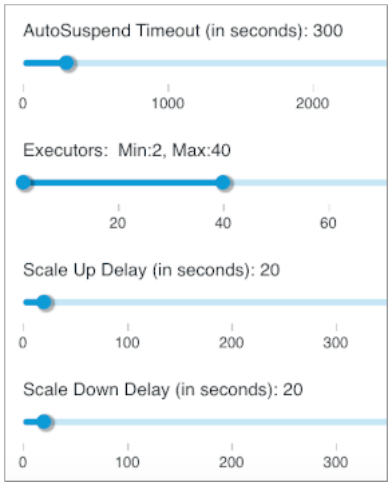

2. In Scale Down Delay, set the seconds to wait before removing executors when executor groups are idle.
3. Click Create Virtual Warehouse.

Configuring Workload Aware Autoscaling

You configure Workload Aware Autoscaling when you create a Virtual Warehouse following a step-by-step procedure using the Cloudera Data Warehouse UI.

About this task

You can also configure WAAS using the CLI.

Before you begin

- You are familiar with the auto-scaling process and Workload Aware Autoscaling.
- You are creating a Virtual Warehouse for running BI-type queries.
- You obtained the DWAdmin role.

About this task

You perform the initial configuration of Workload Aware Autoscaling when you create a Virtual Warehouse, and later, you can edit your configuration. Tune your configuration based on meeting an SLA or reducing cost.

Procedure

1. Create an Impala Virtual Warehouse completing steps in Creating a Virtual Warehouse up to, and including, naming and selecting a Database Catalog for the Virtual Warehouse.
2. In Size, select use workload aware autoscaling.

3. In Small, which is an executor group set, configure Executors Per Group.

   The number of executors per group is analogous to the number of nodes you would need on a bare metal system to handle your workload. A small executor group set helps improve cost and prevents larger groups from starting up.
4. In Large, which is your second executor group set, configure Executors Per Group.

   Choose a value that is a multiple of the value you set for the small Executors Per Group. The greater the spread between values of Executors Per Group the better the query routing.
5. In Min No. of Groups, set the initial number of groups to use with no scaling down unless the Virtual Warehouse is suspended.
6. In Max No. of Groups, set the upper limit for scaling, which is often based on budget or intolerance for query latency.

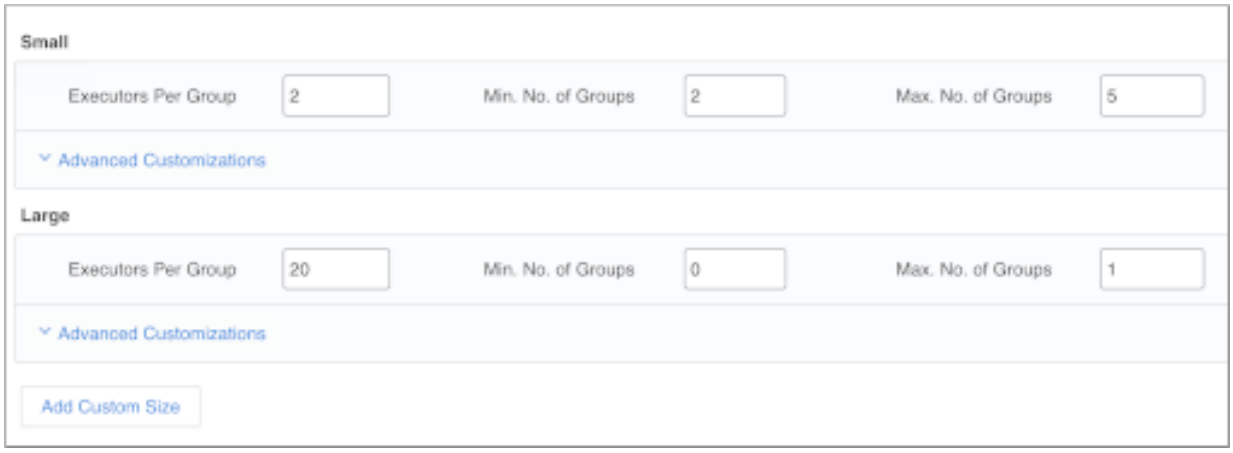

   An executor group set can contain up to 200 executors total. A group size (number of executors per group) must be within 1-100. The max no. of groups is limited by the group size . For example, a set with group size 2 can have up to 100 groups. A set with group size 100 can have up to 2 groups.
7. Optionally, click Advanced Customizations for an executor group to configure Disable AutoSuspend, AutoSuspend Timeout, Scale Up Delay, Scale Down Delay.
8. Optionally, click Add Custom Set to configure more scaling flexibility.
   First, try using Workload Aware Autoscaling without a custom group. Later, if warranted, add up to three custom sets for a total of five executor group sets. Only use more than two executor groups under the following conditions:

   - Your workload is well understood.
   - You have a large mixed workload with clear categories of queries.

   A row for configuring the Custom Set appears.

9. In Custom1, configure executors per group, minimum number of groups, and maximum number of groups.

   Executors per group for a custom set cannot be less than executors per group for the small set or greater than executors per group for the large set.

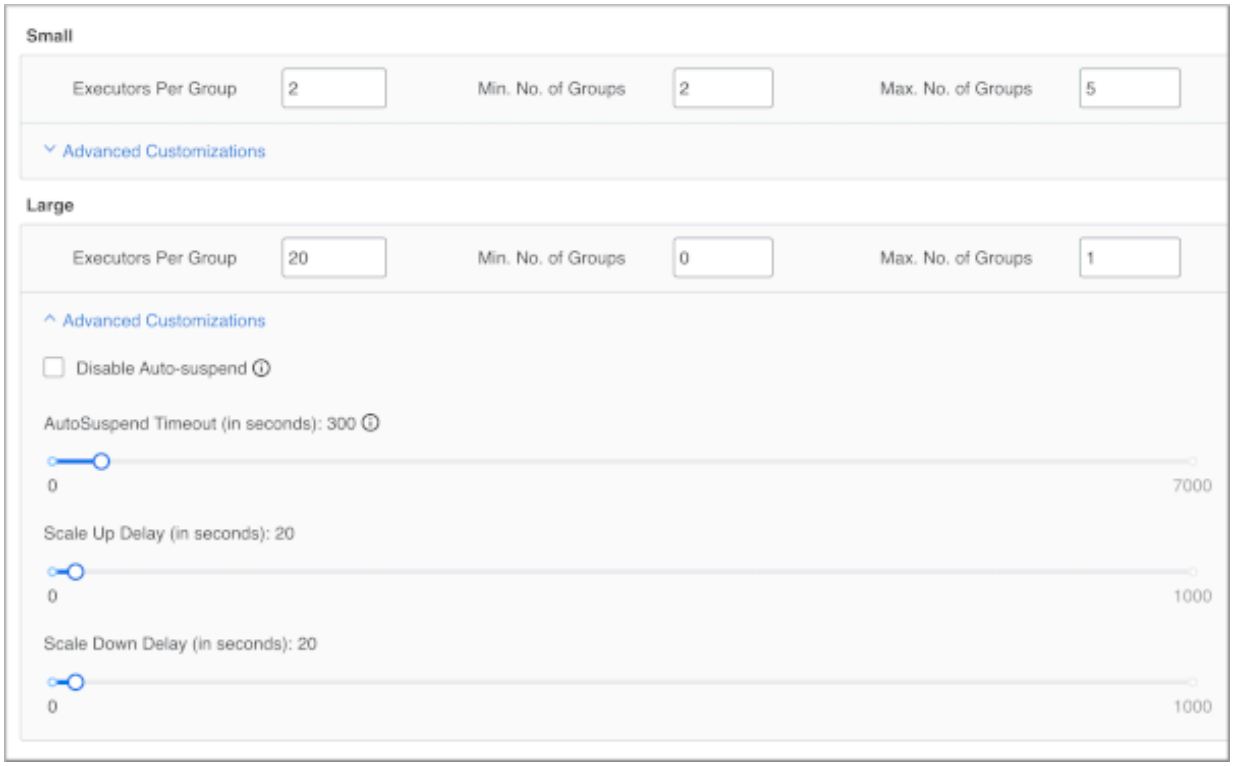

   Custom sized executor group sets are named group-set-custom-1, group-set-custom-2, and so on.

10. Run varying workloads, establish benchmarks for your queries, and add or delete custom sets.
11. Tune your configuration using the REQUEST_POOL query option or COMPUTE_PROCESSING_COST query option.
    Query pool option example:

```
set request_pool=group-set-custom-1;
```

    For more information, see Request query pool option

    Restrict parallelism example:

```
Set COMPUTE_PROCESSING_COST=1;
```

    For more information, see Parallelism query option.

Configuring Impala coordinator shutdown

To cut cloud expenses, you need to know how to configure Impala coordinators to automatically shutdown during idle periods. You need to know how to prevent unnecessary restarts. Monitoring programs that periodically connect to Impala can cause unnecessary restarts.

About this task

When you create a Virtual Warehouse, you can configure Impala coordinators to automatically shutdown during idle periods. The coordinator start up can last several minutes, so clients connected to the Virtual Warehouse can time out.

Before you begin

- Update impyla, jdbc, impala shell clients if used to connect to Impala.
- When you create a Virtual Warehouse of SQL engine type Impala, and then you enable Unified Analytics, you can configure Impala coordinator shutdown only if you also configure Active-Passive mode.

Procedure

1. Follow instructions for "Creating a Virtual Warehouse".
2. Select a size for the Virtual Warehouse.
3. Do not select the Disable AutoSuspend option.

   The Impala coordinator does not automatically shutdown unless the Impala executors are suspended.

4. Select the Allow Shutdown of Coordinator option.

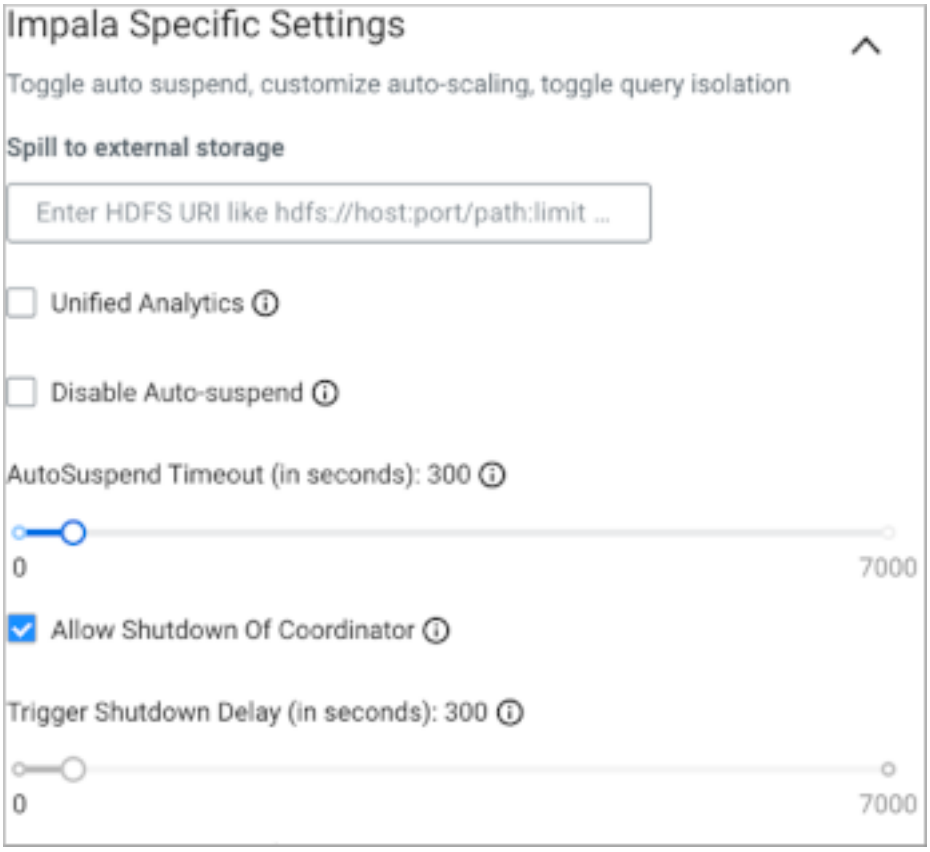

After Impala executors have been suspended, the Impala coordinator waits for the time period specified by the Trigger Shutdown Delay before shutting down.

For example, if AutoSuspend Timeout = 300 seconds and Trigger Shutdown Delay=150 seconds, after 300 seconds of inactivity Impala executors suspend, and then 150 seconds later, the Impala coordinator shuts down.

5. Accept default values for other settings, or change the values to suit your use case, and click Create Virtual Warehouse.

   Click the tooltip ⓘ for information about a setting.

Configuring Impala coordinator high availability

A single Impala coordinator might not handle the number of concurrent queries you want to run or provide the memory your queries require. You can configure multiple active coordinators to resolve or mitigate these problems. You can change the number of active coordinators later.

About this task

You can configure up to five active-active Impala coordinators to run in an Impala Virtual Warehouse. When you create an Impala Virtual Warehouse, Cloudera Data Warehouse provides you an option to configure Impala coordinator and Database Catalog high availability, described in the next topic. You can choose one of the following options:

Disabled

Disables Impala coordinator and Database Catalog high availability

Active-passive

Runs multiple coordinators (one active, one passive) and Database Catalogs (one active, one passive)

Active-active

Runs multiple coordinators (both active) and Database Catalogs (one active, one passive)

By using two coordinators in an active-passive mode, one coordinator is active at a time. If one coordinator goes down, the passive coordinator becomes active.

If you select the Impala coordinators to be in an active-active mode, the client software uses a cookie to keep a virtual connection to a particular coordinator. When a coordinator disappears for some reason, perhaps due to a coordinator shutting down, then the client software may print the error "Invalid session id" before it automatically reconnects to a new coordinator.

Using active-active coordinators, you can have up to five coordinators running concurrently in active-active mode with a cookie-based load-balancing.

An Impala Web UI is available for each coordinator which you can use for troubleshooting purposes.

Clients who connect to your Impala Virtual Warehouse using multiple coordinators must use the latest Impala shell. The following procedure covers these tasks.

Procedure

1. Follow instructions for "Creating Virtual Warehouse".
2. Select the number of executors you need from the Size dropdown menu.

   A number of additional options are displayed, including High availability (HA).

3. Select the Enabled (Active-Active) option from the High availability (HA) drop-down menu.
4. Select the number of coordinators you need from the Number of Active Coordinators drop-down menu ranging from 2 to 5.

   You can edit an existing Impala Virtual Warehouse to change the number of active coordinators.

   Important: Do not decrease the number of active-active coordinators you set up initially; otherwise, the Virtual Warehouse may shut down immediately. If clients are running queries on the Virtual Warehouse, the queries could fail.
5. Change values for other settings as needed, click Create Virtual Warehouse, and wait for the Impala Virtual Warehouse to be in the running state.

   Click ⓘ to learn more about the setting.

6. Go to Cloudera Data Warehouse Overview Impala Virtual Warehouse ⋮ Edit Web UI , and then click each Impala Coordinator Web UI *N* link to get information about the coordinator.

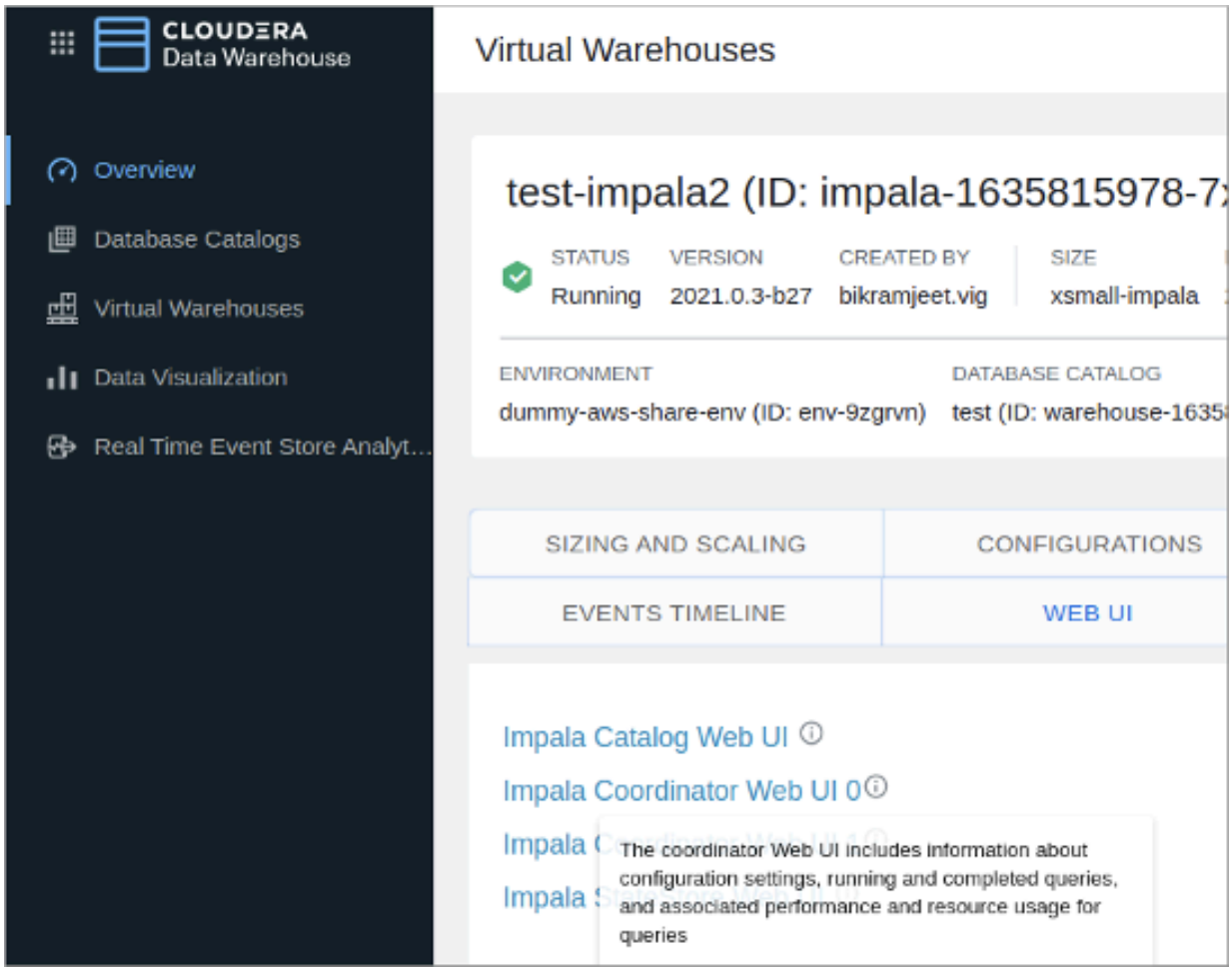

7. Go to Overview Impala Virtual Warehouse ⋮ and select the Copy Impala shell Download command option. The following command is copied to your clipboard:

```
pip install impala-shell==4.1.0
```

8. Provide the command to clients who want to connect to the Impala Virtual Warehouse with multiple coordinators using the Impala shell.
9. Instruct the client user to update impyla to version compatible with Cloudera Data Warehouse, as listed in Version information for Cloudera Data WarehousePublic Cloud components.

   For example, installing/updating impyla 0.18a2, is required to connect to your Virtual Warehouse active-active coordinators in Cloudera Data Warehouse 2021.0.3-b27.
10. Inform the client that to connect over ODBC to an HA-configured Impala Virtual Warehouse that uses active-active coordinators, you must append impala.session.id to the HTTPAuthCookies connector configuration option of the Cloudera ODBC driver.

Table 7: HTTPAuthCookies

Key Name	Value	Required
HTTPAuthCookies	impala.auth,JSESSIONID,KNOXSESSIONID,impala.session.id	Yes

Related Information

Debugging Impala Virtual Warehouses using Web UIs

Configuring Impala catalog high availability

By default, the Impala Virtual Warehouse runs a single instance of the catalog. The catalog stores databases, tables, resource usage information, configuration settings, and other objects managed by Impala. You can optionally configure running an additional Impala catalog instance. One catalog instance operates in active mode, the other in passive mode. The passive instance serves as a backup and takes over if the active instance goes down.

About this task

You enable catalog high availability when you create a New Virtual Warehouse. You cannot turn on, but you can turn off catalog high availability after creating an Impala Virtual Warehouse.

When you create an Impala Virtual Warehouse, you use the same UI dropdown to configure Impala coordinator, covered in the previous topic, and Database Catalog high availability. You can choose one of the following options:

- Disabled

  Disables Impala coordinator and Database Catalog high availability
- Active-passive

  Runs multiple coordinators (one active, one passive) and Database Catalogs (one active, one passive)
- Active-active coordinators

  Runs multiple coordinators (both active) and Database Catalogs (one active, one passive)

Before you begin

You must obtain the DWAdmin role.

Procedure

1. Follow instructions for Adding a new Virtual Warehouse.
2. In Size, select the number of executors, for example xsmall-2Executors.

   A number of additional options appear, including High availability.

3. In High availability (HA), accept the default Enabled (Active-Passive) or Enabled (Active-Active).

   Either option enables Database Catalog high availability in active-passive mode.
4. Accept default values for other settings, or change the values to suit your use case.

   Click the tooltip for information about settings.
5. Click Create Virtual Warehouse.

Configuring Impala Statestore high availability

You can configure High Availability (HA) for Impala Statestore pods in a Virtual Warehouse. One Statestore pod operates in active mode, the other in passive mode. The passive pod serves as a backup and takes over if the active pod goes down.

About this task

Note: You can enable Statestore HA only while creating a new Virtual Warehouse. Cloudera Data Warehouse provides you an option to turn off catalog and statestore HA on an existing Virtual Warehouse.

When you create an Impala Virtual Warehouse, you use the same UI dropdown to configure Impala coordinator, covered in the previous topic, and Database Catalog, and Statestore high availability. You can choose one of the following options:

Disabled

Disables Impala coordinator and Database Catalog, and Statestore high availability

Enabled (Active-passive)

Runs multiple coordinators (one active, one passive) and Database Catalogs, and Statestore (one active, one passive)

Enabled (Active-active)

Runs multiple coordinators (both active) and Catalog daemon (one active, one passive), and Statestore (one active, one passive)

Before you begin

You must obtain the DWAdmin role.

Procedure

1. Follow instructions for Adding a new Virtual Warehouse.
2. In Size, select the number of executors, for example, xsmall - 2 Executors.

   A number of additional options appear, including High availability.
3. In High availability (HA), accept the default Enabled (Active-Passive) or Enabled (Active-Active).

   Either option enables Statestore high availability in active-passive mode.
4. Accept default values for other settings, or change the values to suit your use case.

   Click the tooltip for information about settings.
5. Click Create Virtual Warehouse.

Impala JDBC/ODBC host names

The host name prefix in a JDBC / ODBC URL of an Impala Virtual Warehouse depends on Impala coordinator configurations.

The following table shows the configuration combinations and resulting host name prefix in the JDBC/ODBC URL:

Coordinator HA	Allow Shutdown of Coordinator	Host Name Prefix
Disabled or enabled/active-passive	false	coordinator-<virtual-warehouse-name>
	true	impala-proxy-<virtual-warehouse-name>
Enabled/active-active	true or false	impala-proxy-<virtual-warehouse-name>

Creating a Virtual Warehouse with ARM compute instance types using CDP CLI

In Cloudera Data Warehouse 1.9.1-b233 (released July 26, 2024), you can select a compute instance type while creating a Virtual Warehouse. However, this is supported only using the BETA CDP CLI. By following the steps in this topic, you can create a Virtual Warehouse with regular or ARM-based instance types.

About this task

The following ARM-based instance types are supported starting with the 1.9.1-b233 release of Cloudera Data Warehouse on cloud:

Cloud provider	ARM instance type	Virtual Warehouse type
AWS	r7gd.4xlarge r6gd.4xlarge	Impala
Azure	Standard_E16pds_v5	Impala

Before you begin

You must install the BETA CDP CLI on your Kubernetes cluster.

Procedure

1. Connect to the Kubernetes cluster on which you want to create a Virtual Warehouse.
2. Run the following command to list the supported instance types:

```
dw describe-allowed-instance-types
```

3. Run the command to create a Virtual Warehouse:

   For example:

```
dw create-vw --cluster-id <value> --dbc-id <value> --vw-type <value> --n
ame <value> --instance-type <value>
```

   Specify the instance type you want to use for your Virtual Warehouse using the --instance-type option.

   For the complete list of options, see the documentation for the create-dw command.

Configuring a Virtual Warehouse

After adding a Virtual Warehouse, you can change auto-scaling, enable single-sign-on (SSO) for your clients, configure a time zone, and speed up UDF queries by caching UDF JARs.

Tuning Virtual Warehouses

After creating a Virtual Warehouse, you can change the auto-scaling and some other configurations.

About this task

You set the auto-suspend timeout, the minimum and maximum number of executors for your virtual cluster, when your cluster should scale up and down, and other configurations.

Required role: DWAdmin

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, click Overview in the left navigation pane.

   Note: You can also tune your data warehouse on the Virtual Warehouse page, using the same steps.

3. In the **Overview** page under Virtual Warehouses, find your Virtual Warehouse, click ⋮, and select Edit.
4. In Sizing And Scaling, view the properties you can adjust to tune auto-scaling.
5. Configure auto-scaling as described in Creating a Virtual Warehouse.
6. Click Apply Changes in the upper right corner of the page to apply your changes.

Related Tasks

Tuning Impala auto-scaling

Related Information

Hive Virtual Warehouse auto-scaling

Hive query isolation for ETL jobs

Tuning Impala auto-scaling

About this task

When you tune Impala Virtual Warehouses, you can disable the auto-suspend feature, set the minimum and maximum executor nodes allocated for the warehouse and you can set the scale up and scale down delay which determines auto-scaling behavior.

Required role: DWAdmin

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In **Overview**, in Virtual Warehouses, edit the Impala Virtual Warehouse.
3. Set the auto-suspend behavior:
   - If you do not want your Virtual Warehouse to auto-suspend, click Disable AutoSuspend. If you enable this feature, your Virtual Warehouse does not suspend itself, even if it is idle and no workloads are being processed.
   - If you do not want to disable auto-suspend, set the AutoSuspend Timeout. This sets the time, in seconds, that it takes for the Virtual Warehouse to automatically suspend itself. A Virtual Warehouse auto-suspends itself when the auto-scaler has scaled back to the last executor group and those executors are idle.

4. Adjust the minimum or maximum number of executor nodes as needed:

   Setting and adjusting the minimum and maximum number of executor nodes per Virtual Warehouse is very similar to setting the number of nodes for on-premises clusters. Keep in mind the number of concurrent queries, the complexity of queries, and the volume of queries in your workloads to determine the appropriate number of executor nodes to set on each Virtual Warehouse instance.
5. Set the Scale Up Delay and the Scale Down Delay to fine-tune when the auto-scaler starts scaling up and the number of executor groups to meet workload demand.
   - Scale Up Delay: Sets the length of time in seconds that the system waits before adding more executors when it detects queries waiting in the queue to execute. The time to auto-scale up is affected by how the underlying Kubernetes system is configured.
   - Scale Down Delay: Sets the length of time in seconds that the system waits before it removes executors when it detects idle executor groups. As with the Scale Up Delay setting, the time to auto-scale down is affected by how the underlying Kubernetes system is configured.
6. Click Apply Changes.

Related Information

Impala auto-scaling overview

Tuning Virtual Warehouses

Tuning Impala Java tool options

After creating an Impala Virtual Warehouse, you can change the Java tool options. The options apply only to the Impala Catalog.

About this task

Tuning Impala Java tool options has the following limitations:

- The value of the JAVA_TOOL_OPTIONS represents JVM flags, but in this release, you can change only the Xmx and Xms values for CatalogD's JVM Heap.
- The Xmx flag value cannot be more than 18G.
- Floating point values for memory are not supported; only integers are supported.

Before you begin

Obtain the DWAdmin role.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, click Overview in the left navigation pane.
3. In the Overview page under Virtual Warehouses, select a Virtual Warehouse, click ⋮, and select Edit.
4. Click Configurations.
5. In Impala catalogd, in Configuration files, select env.
6. Adjust the value of environment variables JAVA_TOOL_OPTIONS for Impala CatalogD.
7. Click Apply Changes.

Disabling metadata synchronization

By default, metadata is synchronized when refreshed/invalided across multiple Impala Virtual Warehouses that share a Database Catalog. When you run a command to refresh a table or invalidate metadata in any one of these Impala Virtual Warehouses, metadata is refreshed/invalidated in parallel. You learn how to disable this feature if you do not want to synchronize metadata for some reason.

About this task

By default, running a command from an Impala Virtual Warehouse to refresh tables or invalidate metadata raises events in the Hive metastore. Catalog daemons process events synchronously across all Virtual Warehouses that share metadata. Metadata is refreshed/invalidated in parallel across all your Virtual Warehouses. The enable_reload_events configuration parameter enables or disables raising events for metadata synchronization, effectively enabling or disabling the synchronization:

- enable_reload_events = true (default)
- enable_reload_events = false

Metadata has to be synchronized when you make changes to the data that causes inconsistent file metadata. For example, external clients create, update, and delete partitioned files on the object store.

To synchronize the file metadata in the catalog cache, you run a refresh/invalidate command in a single virtual warehouse. The catalog cache across all the Virtual Warehouses that share a Database Catalog is synchronized and consistent.

In this task, you disable metadata synchronization.

Before you begin

- The DWAdmin role is required to configure Cloudera Data Warehouse.
- An Impala Virtual Warehouse running 1.6.3-b319 (released May 5, 2023) or later

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Virtual Warehouses tab and click ⋮ Edit .

   The **Virtual Warehouse Details** page is displayed.
3. Go to Configurations Impala Catalogd .
4. Select flagfile from the Configuration files drop-down menu, and configure the enable_reload_events key to the value false.
5. Click Apply Changes.

Disabling Impala catalog high availability

You enable atalog high availability when you create a New Virtual Warehouse. You cannot turn on, but you can turn off catalog high availability after creating an Impala Virtual Warehouse. You turn off this feature by editing your Virtual Warehouse.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Virtual Warehouses tab and click ⋮ Edit .

   The **Virtual Warehouse Details** page is displayed.
3. Go to the Sizing And Scaling tab and disable the Enable Impala Catalog Server HA option.
4. Click Apply Changes.

Enabling SSO to a Virtual Warehouse

Learn how to enable SSO (single sign-on) to your Virtual Warehouse from JDBC/ODBC clients. Your authorized clients can connect to the Virtual Warehouse using SSO. Find out how to recognize a connection string to connect to a Virtual Warehouse that is SSO-enabled.

About this task

You enable SSO connections to your Virtual Warehouse when you create a Virtual Warehouse. Authentication occurs through your browser and enterprise identity provider (IdP) provider. Your authorized clients can connect to the Virtual Warehouse using SSO.

When you configure an Impala Virtual Warehouse to use SSO, connections that use the JDBC URL or the ODBC connection string are SSO-enabled connections. Connections to the warehouse using the Impala shell will still use LDAP.

When you configure a Hive Virtual Warehouse to use SSO, all connections that use the JDBC URL are authenticated with the IdP provider that is configured in Cloudera Management Console.

If you are using ODBC to connect, ODBC connector (driver) documentation explains how to make the ODBC connection.

Before you begin

- You must configure an IdP in the User Management module of Cloudera Management Console compliant with Security Assertion Markup Language (SAML 2.0).
- In Management Console User Management you must set up a user group, required for enabling SSO, that identifies the users authorized to access to this Virtual Warehouse.
- You must obtain the DWAdmin role.

Procedure

1. Follow instructions for adding a new Virtual Warehouse.
2. In Size, select the number of executors, for example xsmall-2 Executors.
3. Under Authentication, select Enable SSO to enable single sign-on to your Virtual Warehouse, and in User Groups, select a user group set up in advance to access endpoints.
4. Accept default values for other settings, or change the values to suit your use case, and click Create Virtual Warehouse.
5. Optionally, find your Virtual Warehouses, click options ⋮, and select Copy JDBC URL.

   The JDBC URL is copied to your clipboard.
6. Paste the URL somewhere, and check that a Virtual Warehouse is configured for SSO connections by looking for the “auth=browser” clause in the JDBC URL.
   Hive example:

   ```
   jdbc:hive2://hs2-test.env-9qls4k.dw.xcu2-8y8x.dev.cldr.work/default;tran
   sportMode=http;httpPath=cliservice;ssl=true;auth=browser;
   ```

   Impala example:

   ```
   jdbc:impala://coordinator-test.env-mxxmcn.dw.xcu2-8y8x.dev.cldr.work:443/
   default;
   AuthMech=12;transportMode=http;httpPath=cliservice;ssl=1;auth=browser;
   ```
7. Give users the JDBC JAR and JDBC URL information for connecting to the Virtual Warehouse.
   Users configure an ODBC or JDBC connection to the Virtual Warehouse with a third-party BI tool. A browser window appears with the following success message:

   ```
   Successfully authenticated. You may close this window.
   ```

   Users can query tables they are authorized to access.

Configuring token-based authentication

Using a JSON web token (JWT), your Virtual Warehouse client user can sign on to your Virtual Warehouse for a period of time instead of entering single-sign on (SSO) credentials every time your user wants to run a query.

JWT authentication is a method of proving identity and is standardized through *IETF RFC 7519*. It uses a time limited token to provide identity instead of a username and password. JWT tokens are secrets that are generated by a third-party system using asymmetric key cryptography and used as a bearer token for accessing Virtual Warehouses. Virtual Warehouses only need the public portion of the third-party's asymmetric keys. Within Cloudera, Apache Knox serves as the trusted third-party JWT provider.

Following the procedure below, you acquire a token and set the lifespan of that token (cannot be revoked until expiration). Tokens are secrets that can be used multiple times until they expire.

If you created an Impala Virtual Warehouse for JWT authentication, your client user can choose the following ways to access your the Impala Virtual Warehouse:

- From Impyla

  This option requires your client to configure Impyla as described in "Configuring Impyla for authentication".
- From a JDBC client.

  If you set up a Hive Virtual Warehouse for authentication, you need to configure a few properties as described in "Configuring a Hive Virtual Warehouse for authentication". No configuration of the Impala Virtual Warehouse is required. Finally, you instruct your client to connect to your Hive or Impala Virtual Warehouse to use authentication from a JDBC client.

Related Information

IETF RFC 7519

Generating a JWT token from the Cloudera side

You can generate a JWT token in Cloudera and use the token as a Standard Authorization Bearer token in your client to access a Hive or Impala Virtual Warehouse.

About this task

You can navigate to the Apache Knox UI in Cloudera, or click ⋮ in the Hive or Impala Virtual Warehouse tile. From the Virtual Warehouse tile, skip steps 1 and 2 below, and just click JWT Auth Token Generator shown in the following screenshot.

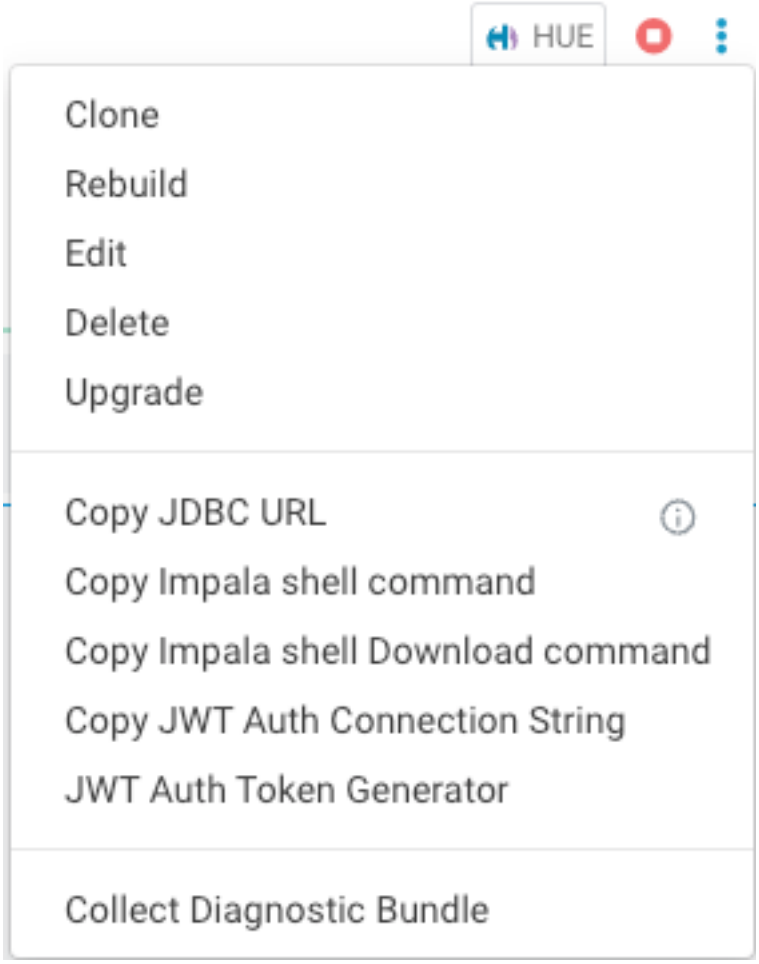

Apache Knox in Cloudera appears. Follow steps 3-5 below to generate the token.

To navigate to the Knox UI instead of using the Cloudera Data Warehouse UI, follow the steps below:

Procedure

1. Log into Cloudera, and in the Cloudera Management Console, select Environments Data Lake .

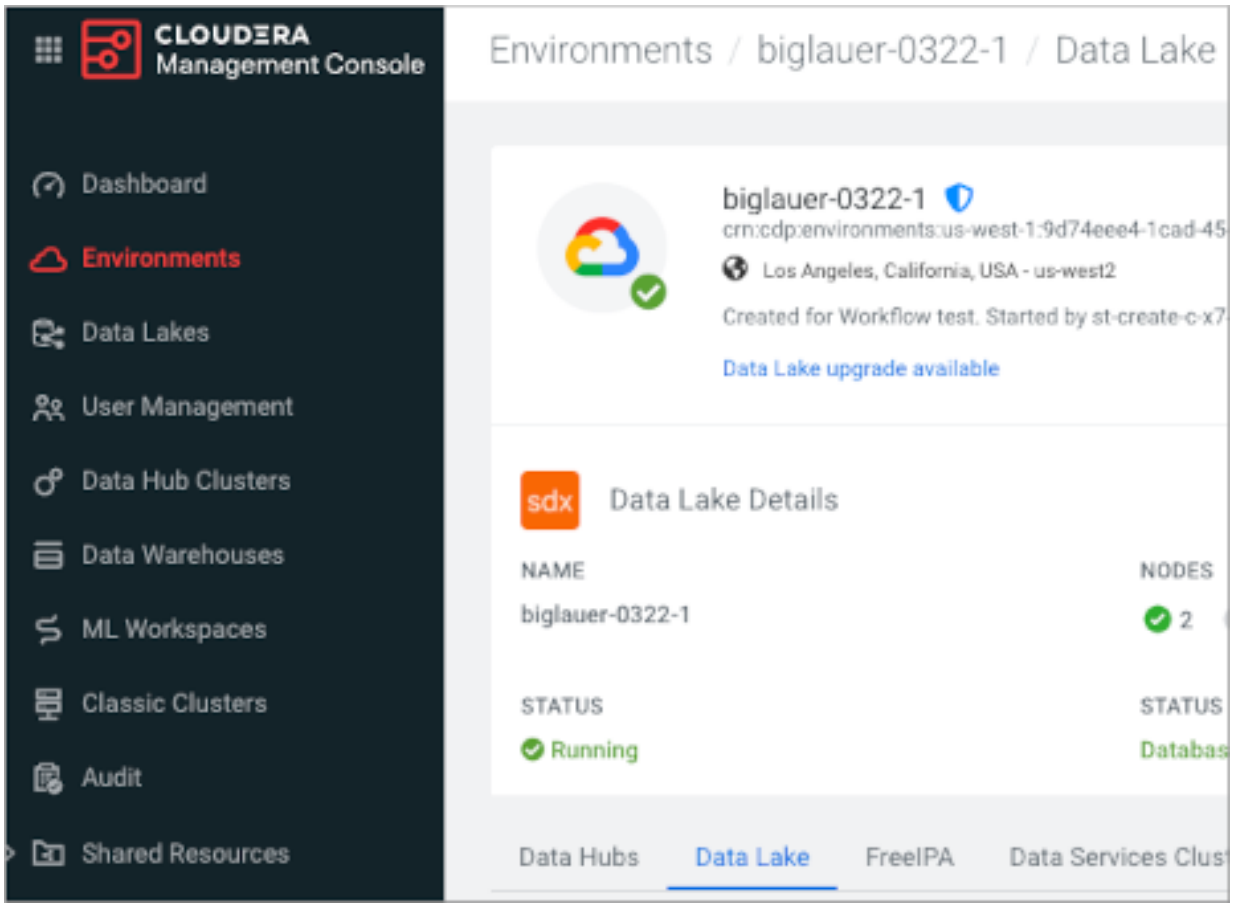

2. Scroll down to Services, and click Token Integration.

Apache Knox in Cloudera appears.

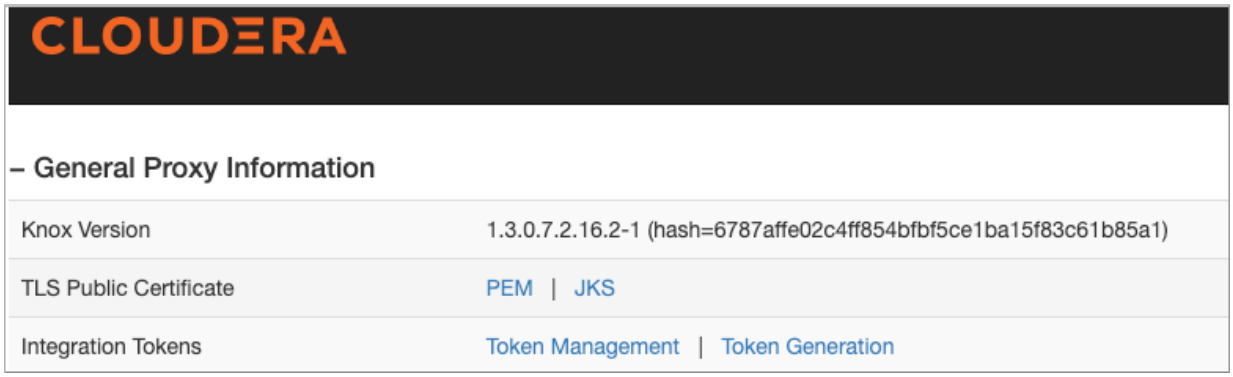

3. Click Token Generation.

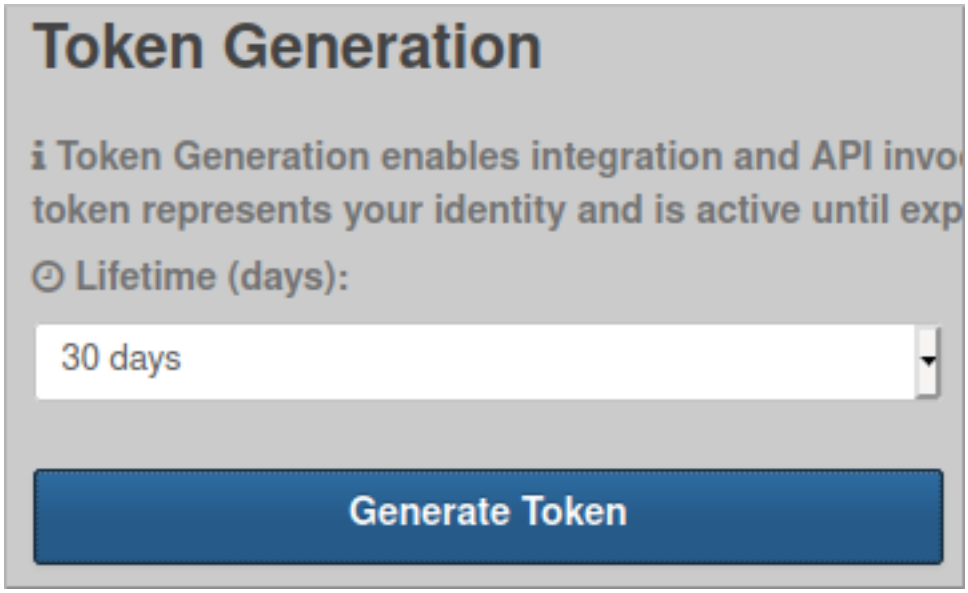

4. In **Token Generation**, choose the lifespan of the token, and click Generate Token.
5. Copy the JWT token that you generated and use the token on your Hive or Impala client to sign into your Virtual Warehouse.

   For security reasons, after you close Token Generation, you cannot see the token again.

Configuring the Impala Virtual Warehouse for authentication

You need to enable platform JWT authentication for your Impala Virtual Warehouse during creation or editing of the Virtual Warehouse. This action automatically configures the Impala Virtual Warehouse to support authentication using JWT tokens issued by the Cloudera JWT authentication provider.

About this task

In this task, you see how to enable platform JWT authentication when you create a new Virtual Warehouse.

Before you begin

Do not enable Unified Analytics if you want to configure JWT access to your Impala Virtual Warehouse for clients.

Procedure

1. Follow the instructions for creating a Virtual Warehouse.
2. Under Authentication, select Enable JWT Authentication.

Configuring Impyla for authentication

You can connect Impyla to the Impala Virtual Warehouse using the HTTP transport protocol and JWT as the authentication mechanism.

Procedure

1. In a terminal window, as a client, you run the command provided by the Impala Virtual Warehouse owner to update Impyla to the version compatible with the Impala Virtual Warehouse.

```
pip install impyla==0.18.0
```

2. Connect Impyla to the Impala Virtual Warehouse using a connection string as described below:
   - host=Name of your Virtual Warehouse host
   - port=Virtual Warehouse port
   - use ssl=true

     Cloudera Data Warehouse only provides a TLS endpoint, it does not provide an unencrypted endpoint. Also, not using TLS will expose the JWT in plain text.
   - http_transport=True

     The http transport is required by Impala for JWT authentication.
   - jwt=<JWT token>

     Substitute your JWT token, which you copied earlier, for <JWT Token>.
   - http_path="cliservice"

     Required.

```
self.connection = connect(<ENV>.host, <ENV>.http_port, use_http_transpor
t=True,
http_path="cliservice",
auth_mechanism="JWT",
timeout=5, jwt=<JWT Token>)
```

On the client side, you can now run queries from Impyla on the Impala Virtual Warehouse.

Configuring the Hive Virtual Warehouse for authentication

You need to enable platform JWT authentication for your Hive Virtual Warehouse during creation or editing of the Virtual Warehouse and configure some HiveServer (HS2) properties. This ensures that the Hive Virtual Warehouse supports authentication using JWT tokens.

Before you begin

You must have acquired the JWT token through your Identity Provider (IDP). In Cloudera Data Warehouse, you use the Knox Token Generation service to generate the token.

Procedure

1. Follow the instructions for creating a Virtual Warehouse.
2. Under Authentication, select Enable JWT Authentication.
3. From the **Overview** page, go to the Virtual Warehouses tab, click ⋮ Edit against the required Hive Virtual Warehouse.

   The **Virtual Warehouse Details** page is displayed.
4. Go to Configurations Hiveserver2 .
5. Select hive-site from the Configuration files drop-down list, search for hive.server2.authentication, and set the value to JWT along with existing authentication protocols that are specified.
   For example, 'LDAP,SAML,JWT'.
6. Search for hive.server2.authentication.jwt.jwks.skip.ssl.cert and set the value to true.
7. Search for hive.server2.transport.mode and set the value to http.

8. Search for hive.server2.authentication.jwt.jwks.url and, if you find it, set the value to the JSON Web Key Sets (JWKS) URL, which is used to sign the JWT token. For example, https://myadmin-host/myadmin/homepage/knoxtoken/api/v1/jwks.json.

   If you do not find hive.server2.authentication.jwt.jwks.url, click Add Custom Configuration and add it to hive-site as a custom property, and then set the value to the JSON Web Key Sets (JWKS) URL.

   You can fetch the JWKS URL by pasting the copied JWT token in the jwt.io website. Paste the JWT token in the Encoded text box of the Debugger and the decoded token header displays a jku parameter that contains the URL. You can provide a comma-separated list of jwks.json URL for this configuration.
9. If you modify the JWKS URL, restart HiveServer.
10. Click Apply Changes.

Getting a JDBC URL for your Virtual Warehouse

After generating a JWT token, if your clients access the Virtual Warehouse over a JDBC connection, you must get the URL to provide to the client.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Virtual Warehouses tab, locate the Virtual Warehouse you want to connect to, and click ⋮ Copy JDBC URL .

   The JDBC URL is copied to your clipboard.
3. Paste the copied JDBC URL into a text file, and provide it to your client.
4. Instruct your client about how to use the token and JDBC URL as described in *Authenticating from a JDBC client*.

Authenticating from a JDBC client

On the Hive or Impala client side, use the JWT token as a Standard Authorization Bearer token by either specifying it in a JWT environment variable or by specifying the token directly in the JDBC connection string.

Before you begin

Authenticating from a JDBC client requires one of the following prerequistes:

- Connect from your client to a Hive or Impala Virtual Warehouse.
- Connect from your client to a Unified Analytics Virtual Warehouse of the Impala type using the Unified Anaytics JDBC driver.

You can download the required driver from the Cloudera Data Warehouse Overview page. Click See More from the **Resources and Download** tile and select the required driver to download.

Note: Simba JDBC and ODBC drivers for Impala support JWT authentication in this release. Ensure that you download the required Database drivers from Cloudera Downloads.

Procedure

Specify the JWT token you copied in "Generating a JWT token from the Cloudera side" as a JWT environment variable, and then construct a Beeline command.
Hive example:

```
export JWT=[***JWT token***]

beeline -u "jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<dwx.
company.com>/default;transportMode=http;httpPath=cliservice;ssl=true;retries
=3;auth=jwt"
```

Impala example using the Unified Analytics JDBC driver and a Unified Analytics Impala Virtual Warehouse:

```
export JWT=[***JWT token***]

beeline -u "jdbc:impala://<your-virtual-warehouse>.<your-environment>.<dwx
.company.com>/default;transportMode=http;httpPath=cliservice;ssl=true;retrie
s=3;auth=jwt"
```

Alternatively, specify the JWT token directly in the JDBC URL connection string.

Hive example:

```
beeline -u "jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<dwx.co
mpany.com>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
;jwt=[***JWT token***]"
```

Impala example using the Unified Analytics JDBC driver and a Unified Analytics Impala Virtual Warehouse:

```
beeline -u "jdbc:impala://<your-virtual-warehouse>.<your-environment>.<dwx.c
ompany.com>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=
3;jwt=[***JWT token***]"
```

You can now query the Virtual Warehouse from the client side.

Connecting Cloudera Data Warehouse and Cloudera AI

A Cloudera user can connect to Cloudera Data Warehouse from Cloudera AI using a JWT Auth Connection String.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Virtual Warehouses tab, locate the Virtual Warehouse you want to connect to, and click ⋮ Copy JWT Auth Connection String .

   The JWT connection string is copied to your clipboard.
3. Paste the copied string into a text file, and provide it to your Cloudera AI client user.

   For more information about connecting Cloudera AI to Cloudera Data Warehouse Virtual Warehouses, see the Cloudera AI documentation.

Configuring a time zone

Cloudera Data Warehouse by default is configured for the UTC time zone. If you want time-related queries in your Virtual Warehouse to return results in your local time zone instead of UTC, you need learn how to change the configuration.

About this task

The way you configure the time zone in the Hive Virtual Warehouse versus the Impala Virtual Warehouse differs. In this task, you configure both.

- First you configure a Hive Virtual Warehouse to return Dubai time when you enter a time-related query, such as SELECT CURRENT_TIMESTAMP.
- Next, you configure an Impala Virtual Warehouse to return Budapest time when you enter a query such as SELECT NOW().

Note: Configuring a timezone in a Virtual Warehouse affects only the time-related queries and not the logs. The Cloudera Data Warehouse logs use the UTC timezone.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Virtual Warehouses tab and click ⋮ Edit against your Virtual Warehouse.

   The **Virtual Warehouse Details** page is displayed.

   a) (Hive) Go to Configurations HiveServer2 .
   b) Select hive-site from the Configuration files drop-down list, and search for hive.local.time.zone.
   c) If the hive.local.time.zone key does not appear, click Add Custom Configuration to add it.
   d) In Configuration Value, enter your local time zone.
   Refer to Timezonedb to find the name of your time zone. For example, in Configuration Value, specify Asia/Dubai.

   a) (Impala) Go to Configurations Impala Coordinator .
   b) Select flagfile from the Configuration files drop-down list, and to the default_query_options key, add your local time zone.
   For example, in VALUE, add timezone="Europe/Budapest",.

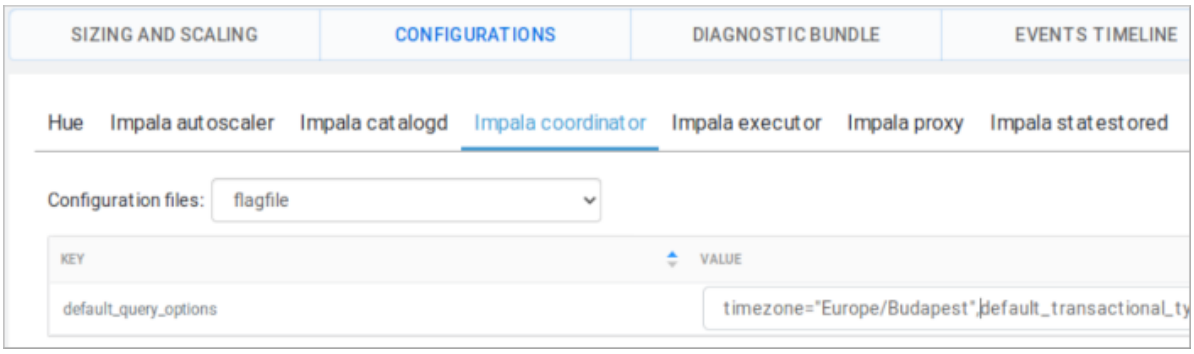

3. Click Apply Changes.

Configuring UDF JAR caching in Hive Virtual Warehouse

After you write and compile your User Defined Function (UDF) code into a Java Archive (JAR) file, you can configure a Hive Virtual Warehouse to cache the UDF JAR in HiveServer (HS2) in Cloudera Data Warehouse.

About this task

UDFs enable you to create custom functions to process records or groups of records. Although Hive provides a comprehensive library of functions, there are gaps for which UDFs are a good solution.

In this task, you configure the Hive Virtual Warehouse to cache the JAR file for quick access by the Virtual Warehouse. After configuring the Virtual Warehouse for caching, the UDF JAR is downloaded from the object store the first time it is called, and then cached. Subsequent calls to the JAR are answered from the cache.

Configuring caching significantly improves performance for queries that use the UDF. Without caching, loading a very large UDF of several hundred MBs can take up to several minutes for each query.

Before you begin

- Create a user-defined function.
- Write, compile, and export your UDF code to a JAR file.
- Upload the UDF JAR file to a bucket or container on AWS or Azure, respectively.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.

2. Go to the Virtual Warehouses tab, locate the Hive Virtual Warehouse that uses the bucket or container where you placed the UDF JAR file, and click ⋮ Edit .

   The **Virtual Warehouse Details** page is displayed.
3. Go to Configurations HiveServer2 .
4. Select hive-site from the Configuration files drop-down menu, and click Add Custom Configuration.

   The **Custom Configuration** modal is displayed.
5. Add the following configuration information, and then click Add:
   - Configuration Key: hive.server2.udf.cache.enabled
   - Configuration Value: true
6. Click Apply Changes.
7. Verify that the configuration property and setting have been added by searching for hive.server2.udf.cache.enabl ed in the search box. If the property has been added, the property name is displayed in the KEY column of the table.

Related Information

Creating a user-defined function

Connecting Cloudera Data Warehouse and Kudu

You can configure an Impala Virtual Warehouse to connect to Kudu in Data Hub and then create tables stored in Kudu from Impala. You learn prerequisites, gather information for the configuration, and see the steps for making the connection. There are several advantages of running Kudu queries from Cloudera Data Warehouse.

Running Kudu queries from an Impala Virtual Warehouse provides benefits, such as isolation from noisy neighbors, auto-scaling, and autosuspend. After configuring an Impala Virtual Warehouse to connect to Kudu, you can create Kudu tables using Hue or the Impala shell.

Prerequisites

You must meet the following requirements:

- Obtain the DWAdmin role.
- Get permission to access Cloudera Data Hub 7.2.15, or later, details to obtain Kudu cluster information.

  Cloudera Data Hub cluster 7.2.15, or later, is required
- Reactivate your AWS or Azure environment version to get 1.6.1-b258 with runtime 2023.0.13.20 (released Feb 7, 2023) and use an Impala Virtual Warehouse.

  Alternatively, run a past release 1.5.1-b110 with runtime 2022.0.12.0-90 or earlier, and perform additional Kubernetes configuration described in the next topic "Setting up a past release for a Cloudera Data Warehouse-Kudu connection" (below), and use an Impala Virtual Warehouse.

  Note: You can upgrade an existing Impala Virtual Warehouse with an older runtime version to 2023.0.13.20 or later. Set catalogd flagfile configuration enable_kudu_impala_hms_check to false if you upgrade.

Gathering Kudu master FQDNs

You need to get the fully qualified domain names (FQDNs) of the Kudu master hosts in your Cloudera Data Hub cluster. You follow steps to navigate to the Kudu Master service and find the FQDNs for configuring a Cloudera Data Warehouse-Kudu connection.

Before you begin

You must meet the prerequisites listed in the previous topic, "Connecting Cloudera Data Warehouseand Kudu".

Procedure

1. Login to Cloudera and click Cloudera Data Hub Clusters.

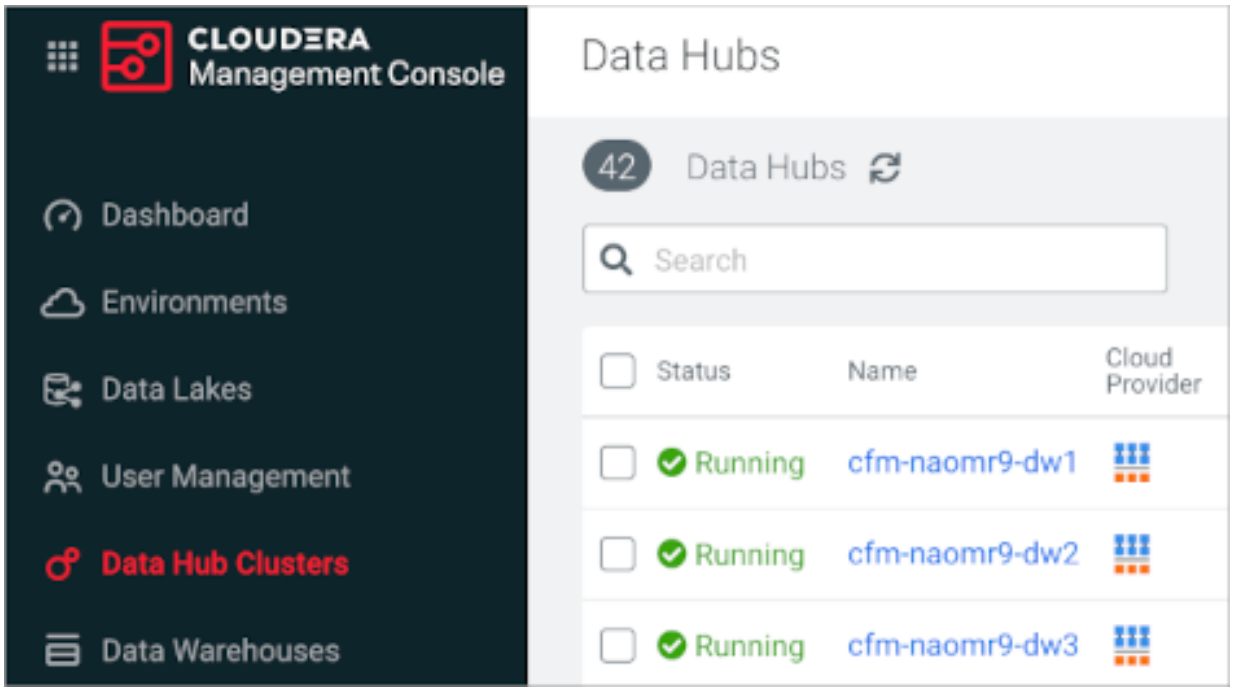

2. In Cloudera Data Hub, in Name, click a cluster name.
3. Scroll down to Services.

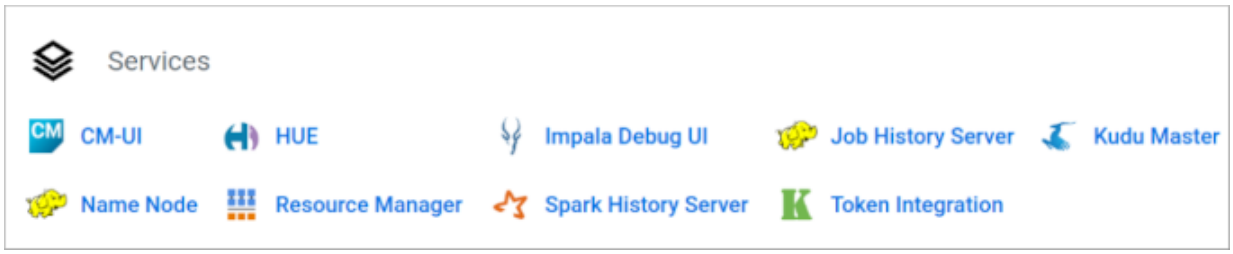

4. Click Kudu Master, and note the instance ID of a Kudu master host(s).
5. Scroll down to Cloudera Manager Info, and click Nodes.

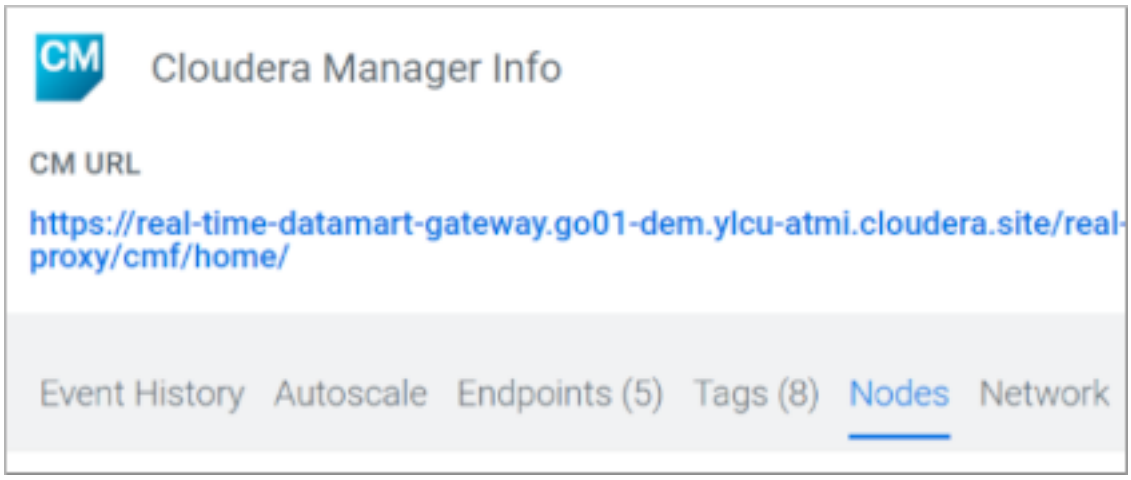

The list of nodes appear, including their fully qualified domain name (FQDN).

6. Find the Kudu cluster in the list, and note the FQDN of the instances(s) running Kudu Master services.
   For example:

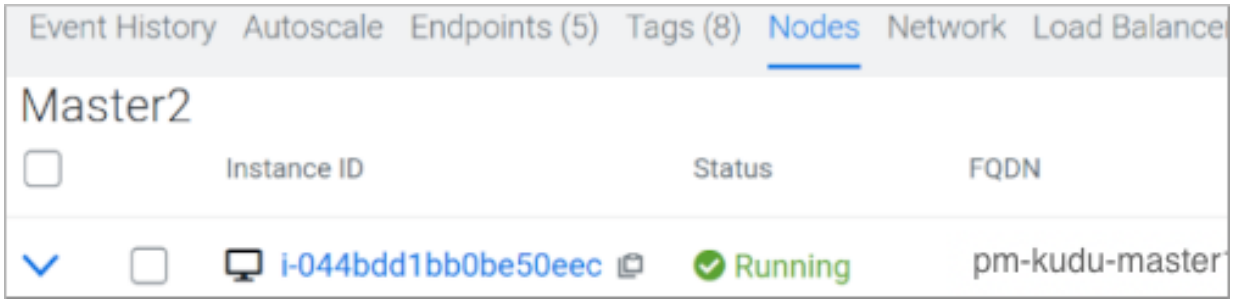

   Now, you have the information you need to configure a Cloudera Data Warehouse-Kudu connection as described in the next topic.

Configuring a Cloudera Data Warehouse-Kudu connection

The steps for making the connection involve setting a Impala coordinator property to the fully qualfied domain names (FQDNs) of the Kudu master hosts.

Before you begin

- You must meet the prerequisites listed in the "Connecting Cloudera Data Warehouse and Kudu".
- You need to have gathered the FQDNs of Kudu master hosts as described in the previous topic, "Gathering Kudu master FQDNs".

Procedure

1. From the Cloudera Management Console or Cloudera landing page, navigate to Data Warehouse.
2. In Virtual Warehouses, select your Impala Virtual Warehouse, and click ⋮ Edit .
3. In **Virtual Warehouse Details** page, click Configurations Impala Coordinator .

4. Select flagfile from the Configuration files drop-down list, and set the kudu_master_hosts key to a list of all Kudu master FQDNs.
   New Virtual Warehouse example:

```
kudu_master_hosts=kudu-master1.sandbox.a.cloudera.site:7051,kudu-master2
.sandbox.a.cloudera.site:7051,kudu-master3.sandbox.a.cloudera.site:7051
```

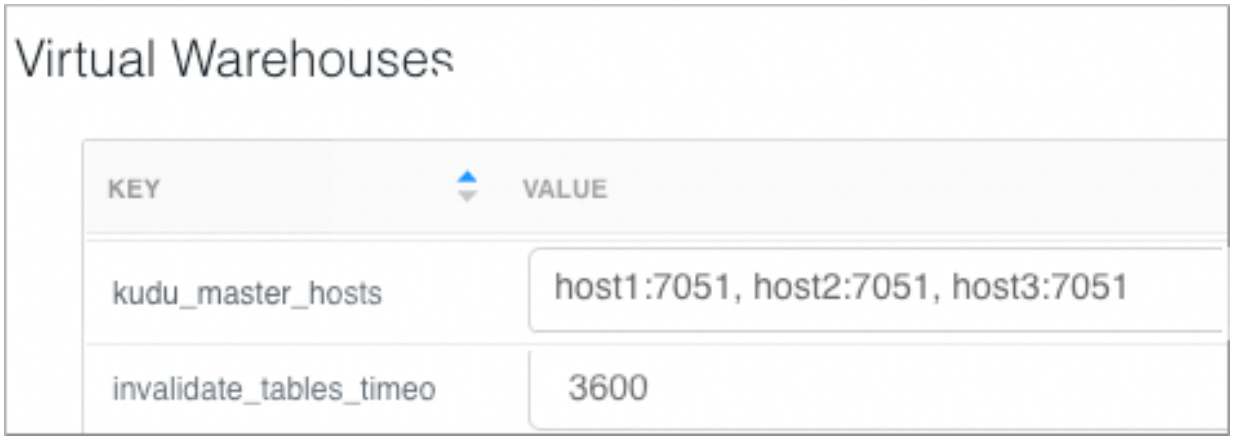

   This screenshot shows the three FQDNs, oversimplified for readability.

   Existing Virtual Warehouse:

   - Click Add Custom Configurations, set kudu_master_hosts to the FQDN of a single master host as shown follows:
     - Configuration Key: kudu_master_hosts
     - Configuration Value: host1:7051
   - Select flagfile from the Configuration files drop-down list, and add the FQDNs of other Kudu master hosts to the value of kudu_master_hosts.

5. Click Apply Changes.

   When the Virtual Warehouse finishes updating, you can query Kudu tables from Hue, an Impala shell, or an ODBC/JDBC client. For example:

```
--- First, create an example table
CREATE TABLE my_first_kudu_table
(
  id BIGINT,
  name STRING,
  PRIMARY KEY(id)
)
PARTITION BY HASH PARTITIONS 16
STORED AS KUDU;

--- Next, insert some example data
INSERT INTO my_first_kudu_table VALUES (101,'A'),(102,'B'),(103,'C');
--- Finally, select from the example table to show the above has worked
SELECT * FROM my_first_kudu_table;
```

Setting up a past release for a Cloudera Data Warehouse-Kudu connection

To connect Cloudera Data Warehouse to Kudu using a past release 1.5.1-b110 (released Nov 22, 2022) or earlier, you need to configure Kubernetes DNS resolution unless you create a new environment. Configuring the Kubernetes DNS resolution is required only in AWS environments.

About this task

If you create a new environment, you do not need to configure Kubernetes DNS resolution. After completing the steps in this task, you need to meet additional prerequisites and configure a Cloudera Data Warehouse-Kudu connection as described in the previous topic, “Configuring a Cloudera Data Warehouse-Kudu connection” .

Procedure

1. If you have an AWS environment, perform all the steps in this procedure. If you have an Azure environment, go to step 6 (upgrading).
2. Grant remote access to Kubernetes clusters on Amazon EKS.
3. On the command line of your cluster, open the Cloudera CoreDNS Updater file for configuration in a text editor. For example:

```
kubectl edit configmap cdp-coredns-updater-63cc3d36-283cd77b-template -n
cluster
```

The template will have a unique name, not cdp-coredns-updater-63cc3d36-283cd77b-template in your cluster.

The data section looks something like this:

```
data:
Corefile.template: |-
.:53 {
  errors
  health
  kubernetes cluster.local in-addr.arpa ip6.arpa {
  pods insecure
  fallthrough in-addr.arpa ip6.arpa
  ttl 30
}
prometheus :9153
forward . /etc/resolv.conf
cache 30
loop
reload
loadbalance
}
{{applications.FREEIPA.services.dns.config.domain}}:53 {
  errors
  cache 30
  reload
  forward . {{#applications.FREEIPA.services.dns.endpoints}}{{host}} {{/
applications.FREEIPA.services.dns.endpoints}}
}
```

4. Append a section to the data section above to enable reverse lookup for data lake IPs.

```
cluster.local in-addr.arpa ip6.arpa {
  kubernetes {
  pods insecure
  fallthrough in-addr.arpa ip6.arpa
  ttl 30
}
forward . {{#applications.FREEIPA.services.dns.endpoints}}{{host}} {{/ap
plications.FREEIPA.services.dns.endpoints}}
}
```

Your data section should now look like this:

```
data:
Corefile.template: |-
.:53 {
  errors
  health
  kubernetes cluster.local in-addr.arpa ip6.arpa {
  pods insecure
  fallthrough in-addr.arpa ip6.arpa
  ttl 30
}
prometheus :9153
forward . /etc/resolv.conf
cache 30
loop
reload
loadbalance
}
{{applications.FREEIPA.services.dns.config.domain}}:53 {
  errors
  cache 30
  reload
  forward . {{#applications.FREEIPA.services.dns.endpoints}}{{host}} {{/
applications.FREEIPA.services.dns.endpoints}}
}
cluster.local in-addr.arpa ip6.arpa {
  kubernetes {
  pods insecure
  fallthrough in-addr.arpa ip6.arpa
  ttl 30
}
forward . {{#applications.FREEIPA.services.dns.endpoints}}{{host}} {{/app
lications.FREEIPA.services.dns.endpoints}}
}
```

5. Restart the coredns updater pod to pick up the new configurations.

```
kubectl --kubeconfig ~/kubeconfig_int get pods -n cluster | grep -i 'cor
edns-updater'
cdp-coredns-updater-63cc3d36-283cd77b-67b8c9479d-bjtlh 3/3 Running 1 (2d20
h ago) 2d20h

kubectl --kubeconfig ~/kubeconfig_int delete pod cdp-coredns-updater-63cc3
d36-283cd77b-67b8c9479d-bjtlh -n cluster
```

6. If the Impala Virtual Warehouse runtime version is 2022.0.12.0-90 released December 13, 2022 or earlier, upgrade the Virtual Warehouse version to 2023.0.13.20 or later.
7. Proceed to configure a Cloudera Data Warehouse-Kudu connection as described in the previous topic, "Configuring a Cloudera Data Warehouse-Kudu connection".

Configuring Iceberg manifest caching in Impala Virtual Warehouse

Apache Iceberg provides a mechanism to cache the contents of Iceberg manifest files in memory. In Cloudera Data Warehouse, you can enable or disable Iceberg manifest caching for Impala Coordinators and Catalogd, and set a few other properties, in your Impala Virtual Warehouse.

About this task

The manifest caching feature helps to reduce repeated reads of small Iceberg manifest files from remote storage by Impala Coordinators and Catalogd.

The following default properties are set in Cloudera Data Warehouse for manifest caching:

```
iceberg.io.manifest.cache-enabled=true;
iceberg.io.manifest.cache.max-total-bytes=104857600;
iceberg.io.manifest.cache.expiration-interval-ms=3600000;
iceberg.io.manifest.cache.max-content-length=8388608;
```

The following list describes each property:

- iceberg.io.manifest.cache-enabled: enable/disable the manifest caching feature.
- iceberg.io.manifest.cache.max-total-bytes: maximum total amount of bytes to cache in the manifest cache. Must be a positive value.
- iceberg.io.manifest.cache.expiration-interval-ms: maximum duration for which an entry stays in the manifest cache. Must be a non-negative value. Setting zero means cache entries expire only if it gets evicted due to memory pressure from iceberg.io.manifest.cache.max-total-bytes.
- iceberg.io.manifest.cache.max-content-length: maximum length of a manifest file to be considered for caching in bytes. Manifest files with a length exceeding this property value will not be cached. Must be set with a positive value and lower than iceberg.io.manifest.cache.max-total-bytes.

Generally, you set a different value for the expiration interval in catalogd and the coordinator. The expiration time is later in catalogd, for example, 1 week. Catalogd needs caching for a longer period of time because the catalogd service serves table metadata.

Changing configuration parameters in Cloudera Data Warehouse is recommended only when following Cloudera instructions.

Before you begin

Obtain the DWAdmin role.

Procedure

1. In Cloudera Data Warehouse, click Overview, and find your Impala Virtual Warehouse.
2. Click ⋮ Edit Configurations Impala Catalogd
3. Search for the following string: iceberg.io.
   The following properties appear:

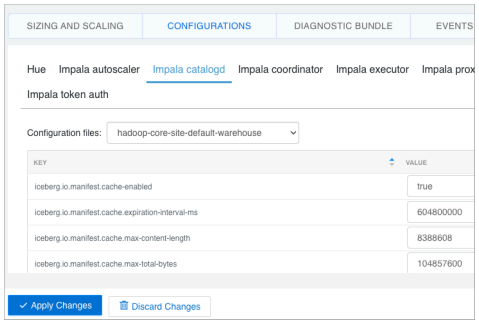

4. Change the value of one or more properties, and click Apply Changes.

5. Click Configurations Impala coordinator .
   The following properties appear:

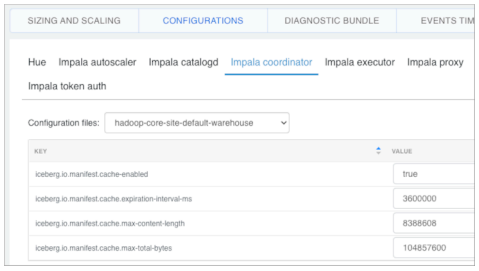

6. Change the value of one or more properties, and click Apply Changes.

Managing high partition workloads

You learn how to identify an error related to high partition workloads that require configuration of your Hive Virtual Warehouse to run successfully. You might need to configure your Virtual Warehouse to prevent these errors when inserting data into workloads with a large number of partitions and columns.

About this task

To prevent an error when inserting data into a high partition workload, such as a table having 5,000 partitions and 100 columns, configure the Hive Virtual Warehouse as described in the steps below.

The error message you might see when inserting data into a high partition workload that is not configured properly looks something like this:

```
ERROR : FAILED: Execution Error, return code 40000 from org.apache.hadoop.hi
ve.ql.exec.MoveTask. MetaException(message:One or more instances could not b
e made persistent)
```

Procedure

1. Try changing the size of your Hive Virtual Warehouse.
   For example, if you experience the high partition error in a small Virtual Warehouse, try using a medium Virtual Warehouse and set min/max nodes to 40.
2. In the Cloudera Data Warehouse UI, in Overview, go to the Virtual Warehouses tab, click ⋮ , and select Edit corresponding to your Virtual Warehouse.
3. Click Configurations HiveServer2 .
4. Tune HS2 parameters in the Virtual Warehouse.
   For example, search for and set the following properties, or if the property is not found, click Add Custom Configuration to add a custom configuration, and set it as follows:

```
set hive.optimize.sort.dynamic.partition.threshold=0;
set hive.thrift.client.max.message.size=2147483647;
set hive.metastore.client.skip.columns.for.partitions=true;
set hive.stats.autogather=false;
set hive.stats.column.autogather=false;
set hive.msck.repair.batch.size=[***TABLE SCHEMA SIZE***];
```

5. Go to the Database Catalogs tab, and select your Database Catalog.
6. Click Options ⋮ , and select Edit.

7. Tune Metastore parameters in the Database Catalog.
   For example, click Configurations Metastore , search for and set the following properties, or if the property is not found, click + to add a custom configuration, and set it as follows.

```
hive.metastore.direct.sql.batch.size=[***TABLE SCHEMA SIZE***]
hive.txn.timeout=3600
hive.metastore.try.direct.sql=true
hive.metastore.try.direct.sql.ddl=true
```

8. Click Apply Changes.

Configuring an Operational Database connection

Having a connection to Cloudera Operational Database, you can create tables from Hive and store data in HBase.

The tables you create from Hive and store in HBase have some metadata stored in the Hive metastore. The data ia stored in HBase.

You must meet the following requirements to configure a Cloudera Operational Database connection.

- Have an AWS-based environment.
- Have administrative access to the S3 object store associated with your Hive Virtual Warehouse.

Configuring a Cloudera Operational Database connection involves three top-level steps. First, you set up Ranger permissions to access Cloudera Operational Database from Hive. Next, you generate configuration files. Finally, you configure the Hive Virtual Warehouse to connect to Cloudera Operational Database.

Setting up Cloudera Data Warehouse to connect to Cloudera Operational Database

You configure your Virtual Warehouse to use configuration files you generated for connecting to Cloudera Operational Database.

Before you begin

- You generated configuration files hbase-site.xml and hbase-cdw.tar.gz as described in the previous topic.

Procedure

1. In the Cloudera Data Warehouse UI, in Overview, go to the Virtual Warehouses tab, select a Hive Virtual Warehouse, and click ⋮ Edit .
   The **Virtual Warehouse Details** page is displayed.
2. Go to Configurations Hiveserver2 .
3. In Configuration files, select hbase-site.xml, and copy the contents of the hbase-site.xml generated by the script you ran.
4. Create a directory or use an existing directory on S3.

```
hadoop fs -mkdir s3a://odx-weekly-test/cod--tddl6gutqgn9/jars
```

5. Copy the hbase-cdw.tar.gz, which you also obtained from the script you ran, to that directory.
6. In the Hive Virtual Warehouse Details Configurations Hiveserver2 , in Configuration files, select env and select the CDW_HIVE_AUX_JARS_PATH key.
7. Set the value of the CDW_HIVE_AUX_JARS_PATH key to the path of the jar file.
8. Go to the Query coordinator tab and select env from the Configuration files drop-down menu, and select the CDW_HIVE_AUX_JARS_PATH key.
9. Set the value of the CDW_HIVE_AUX_JARS_PATH key to the path to the jar file.
10. Click Save Changes and restart the Virtual Warehouse.

Resizing the Impala Virtual Warehouse size

After creating an Impala Virtual Warehouse, you can tune, or correct, the T-shirt size of executor groups that drive the Impala Virtual Warehouse. The size of the executor groups is critical for achieving cost and performance goals.

About this task

You do not have to recreate the Virtual Warehouse if you want to try a different size. For example, you can easily change the size of executor groups from xxsmall-impala to xsmall-impala after creating a the Virtual Warehouse by following the procedure below to edit the Impala Virtual Warehouse.

Before you begin

- You obtained the DWAdmin role.
- Understand "Warehouse sizing requirements".

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, click Overview in the left navigation pane.

   Note: You can also tune your data warehouse on the Virtual Warehouse page, using the same steps.

3. In **Overview** page, find your Virtual Warehouses, click ⋮, and select Edit.
4. In Sizing And Scaling, select a different size from the Size drop-down menu.
5. Click Apply Changes.

Resizing the Hive Virtual Warehouse size

The size of the Hive Virtual Warehouse you select during Virtual Warehouse creation determines the number of executors and concurrent queries the Virtual Warehouse can run. You need to know how to change the size of the Virtual Warehouse upward or downward to tune performance and manage cost.

About this task

You cannot change the size of a Hive Virtual Warehouse, but you can handle incorrect sizing in the following ways.

- You can delete the Virtual Warehouse, and then recreate it in a different size.
- You can change the auto-scaling thresholds to change the effective size of the Virtual Warehouse based on demand. The actual size does not change, but increases or decreases in resources occurs automatically.

This task assumes you have two Virtual Warehouses that you decide are incorrectly sized for some reason. You correct the sizing of one by deleting and recreating the Virtual Warehouse. You correct the effective sizing of the other by changing auto-scaling thresholds.

Before you begin

- You obtained the DWAdmin role.

Procedure

First Virtual Warehouse: Replace this Virtual Warehouse

1. Log in to the Cloudera web interface, navigate to Data Warehouse Overview , note the name of the Virtual Warehouse you want to modify, and note which Database Catalog it is configured to access.
2. Click the options ⋮ of the Virtual Warehouse you want to delete, and select Delete.
3. Go to the Virtual Warehouses tab and click New Virtual Warehouse.
4. Set up the new Virtual Warehouse:
   - Type the same Name for the new Virtual Warehouse as you used for the old Virtual Warehouse.

     Note: The fully qualified domain name of your Virtual Warehouse, which includes the Virtual Warehouse name plus the environment name must not exceed 64 characters; otherwise, Hue cannot load.
   - In Type, click the SQL engine you prefer: Hive or Impala.
   - Select your Database Catalog and User Group if you have been assigned a user group.
   - In Size, select the number of executors, for example xsmall-2Executors.
   - Accept default values for other settings, or change the values to suit your use case.
5. Click Create Virtual Warehouse.

Second Virtual Warehouse: Change the Auto-Scaling Thresholds

6. In Data Warehouse Overview , click the options ⋮ of the other Virtual Warehouse, a Hive Virtual Warehouse for example, to change auto-scaling thresholds, and select Edit.
7. Go to the Sizing And Scaling tab and in Concurrency Autoscaling, slide the control to change the Max number of executors.
8. Click Apply Changes.

Related Information

Creating a Virtual Warehouse

Configuring auto-scaling

Compaction in Cloudera Data Warehouse

You understand the importance of compaction and the consequences of neglecting to perform compaction. Compaction keeps your Data Warehouse healthy.

Over time tables belonging to a workload become fragmented due to operations performed on them by your workload users. These small, obsolete files might lead to performance degradation and query latency problems. Compaction plays a major role in improving response time to workload queries by reducing the number of underlying files for a table and eliminating the obsolete ones. Compaction runs periodically in the background to maintain the optimal state.

Running periodic compaction is a best practice for the performance for ACID transactions. ACID inserts and deletes generate the problematic files that you might need to monitor and manage. In Cloudera Data Warehouse, compaction is always performed by a Hive Virtual Warehouse.

Compaction prerequisites

To prevent data loss or an unsuccessful compaction, you must check for adequate resources in your first Hive Virtual Warehouse used for compaction.

Check for adequate resources

You must take into account the query workload for compaction when you create the first Hive Virtual Warehouse. You must make sure that the warehouse has adequate resources to handle the compaction workload in addition to any other workloads you might run in that warehouse.

Important: Impala Virtual Warehouses cannot be designated as the compactor Virtual Warehouse for a Database Catalog. Compaction tasks can only be assigned to a Hive Virtual Warehouse.

Change compactor configuration for Hive Virtual Warehouses on Cloudera Data Warehouse Public Cloud

To enhance performance, the compactor is a set of background processes that compact delta files, which are created as a by-product of data modifications. When it runs, it incurs additional load on the Hive Virtual Warehouse assigned as the compactor in Cloudera Data Warehouse Public Cloud. You can change which Hive warehouse performs compaction to load-balance this workload as necessary.

About this task

Required role: DWAdmin

Before you begin

One Hive Virtual Warehouse must be the compactor for all the Virtual Warehouses associated with a particular Database Catalog. This Hive Virtual Warehouse compactor runs all of the compaction queries for all Virtual Warehouses that use one particular Database Catalog, including Impala Virtual Warehouses. However, Impala Virtual Warehouses cannot be designated as the compactor Virtual Warehouse for a Database Catalog. Compaction tasks can only be assigned to a Hive Virtual Warehouse. The first Hive Virtual Warehouse you create against a Database Catalog is automatically set as the compactor. If you decide you do not want that particular warehouse to take on the compaction workload, you can set another Hive Virtual Warehouse to perform the compaction workload by following these steps:

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. On the Overview page, select the Hive Virtual Warehouse that you want to set as the compactor, and click the options menu ⋮ .
3. In the options menu, select Set Compactor.

Related Information

Virtual Warehouse sizing requirements for public cloud environments

Initiating automatic compaction

To start automatic compaction in Cloudera Data Warehouse, you set a property in the Database Catalog.

Procedure

1. Log into the Cloudera web interface and in Cloudera Home, click Cloudera Data Warehouse.
2. In Cloudera Data Warehouse, select your Database Catalog and click ⋮ Edit .
3. Click Configurations Metastore .
4. From the drop-down, select hive-site for the configuration category.
5. Set hive.compactor.initiator.on to true.
6. Click Apply Changes and restart the Database Catalog.

Configuring compaction

You configure some configure compaction properties from the Database Catalog and some from the Virtual Warehouse.

About this task

In this task, you configure the compactor worker threads in the Virtual Warehouse.

Procedure

1. Go to the Virtual Warehouses tab and select a Hive Virtual Warehouse.
2. Click ⋮ Edit Configurations Hiveserver2 .

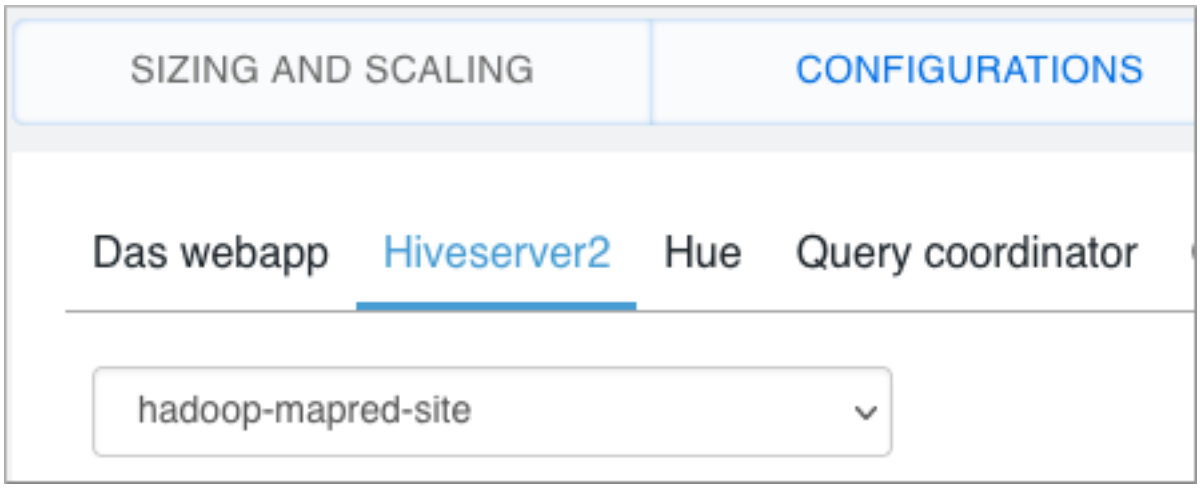

3. Select hive-site from the Configuration files drop-down menu.
4. View, increase, or decrease the number the hive.compactor.work.threads.
5. Click Apply Changes.

Automating metadata invalidation after compaction

As you insert or delete data from Hive ACID tables, Hive generates delta files. The compaction process consolidates the delta files and keeps the system healthy. If you query Hive ACID tables from an Impala Virtual Warehouse, you need to learn how to automatically invalidate metadata after compaction to prevent a possible query failure.

About this task

After compaction of ACID tables, metadata in Impala coordinator local caches might be stale. You might experience a failure (FileNotFoundException) when you query these tables. Invalidation refreshes the metadata and prevents the problem. You can configure automatic metadata invalidation as described in the following procedure.

Before you begin

You must upgrade your Database Catalog to the compatible version for DWX-1.1.2-b2008 or later to use this feature.

Procedure

1. In Cloudera Data Warehouse, click Overview.
2. In an Impala Virtual Warehouse, click ⋮ Edit .
3. Go to Configurations Impala Coordinator and select flagfile from the Configuration files drop-down menu.

4. Set auto_check_compaction to true, and click Apply Changes.

How compaction works

When data changes are made on Cloudera Data Warehouse with inserts, updates, and deletes, delta files are created. The more changes that are made, the more delta files are created. When a large number of delta files are created, query performance degrades. Compaction removes these delta files to enhance query performance.

There are two types of compaction:

Minor compaction

Compacts multiple delta files into a single delta file.

Major compaction

Compacts one or more delta files and the base file for the bucket and creates a single new base file per bucket.

The goal of compaction is to "self heal" tables in order to restore the baseline query performance. All compactions are done in the background and do not prevent concurrent reads and writes of the data. After compacting, the system waits for all readers of the old files to finish and then removes the old files.

Compactor processes

These background processes run inside the metastore and HiveServer2 in Cloudera Data Warehouse Public Cloud. They support the data modifications made as a result of ACID transactions.

Compactor process	Description
Initiator	This process runs in the metastore, which equates to the Database Catalog construct in the Cloudera Data Warehouse UI, and discovers which tables and partitions are due for compaction. By default, it runs every 5 minutes. To change this interval: **1.** Identify the Database Catalog for the Virtual Warehouse on which you want to change the compaction interval by selecting the Virtual Warehouse tile. The associated Database Catalog is highlighted. **2.** In the Database Catalog, click the edit icon in the tile to launch the Database Catalog details page. **3.** On the Database Catalog details page, make sure the CONFIGURATIONS tab is selected, and then select the Metastore subtab. **4.** On the Metastore tab, select hive-site from the drop-down list, and search for the hive.compactor.check .interval KEY. **5.** Add your preferred check interval in the associated VALUE field in seconds. **6.** Click Apply Changes. The services are automatically updated with the new configuration.
Worker	This process runs in HiveServer2, which equates to the Hive Virtual Warehouse construct in the Cloudera Data Warehouse UI. The worker process performs the actual compacting work. In Cloudera Data Warehouse, compaction runs an INSERT statement created from the output of a SELECT statement, thereby re-writing the data to new base or delta files.
Cleaner	This process runs in the metastore and deletes delta files after compaction and after it determines the files are no longer needed. By default, the cleaner runs every 5 seconds (5,000 milliseconds). The check occurs on the visibility ID/transaction ID, which is a global transaction identifier.

How compaction interacts with the Data Lake

In the Data Lake on Cloudera, the initiator and cleaner processes also run in the metastore as they do in Cloudera Data Warehouse Public Cloud. However, the worker process runs in HiveServer (HS2), which equates to a Hive Virtual Warehouse..

In Cloudera Data Warehouse, the initiator and cleaner processes run in the Database Catalog, which equates to the metastore. The Database Catalog maintains a connection with the Data Lake and compaction jobs run in parallel with it. The worker process in Hive Virtual Warehouses executes queries to perform compaction.

Cloudera Data Warehouse Public Cloud Compaction Architecture

This diagram illustrates how the components that perform compaction interact on Cloudera Data Warehouse Public Cloud.

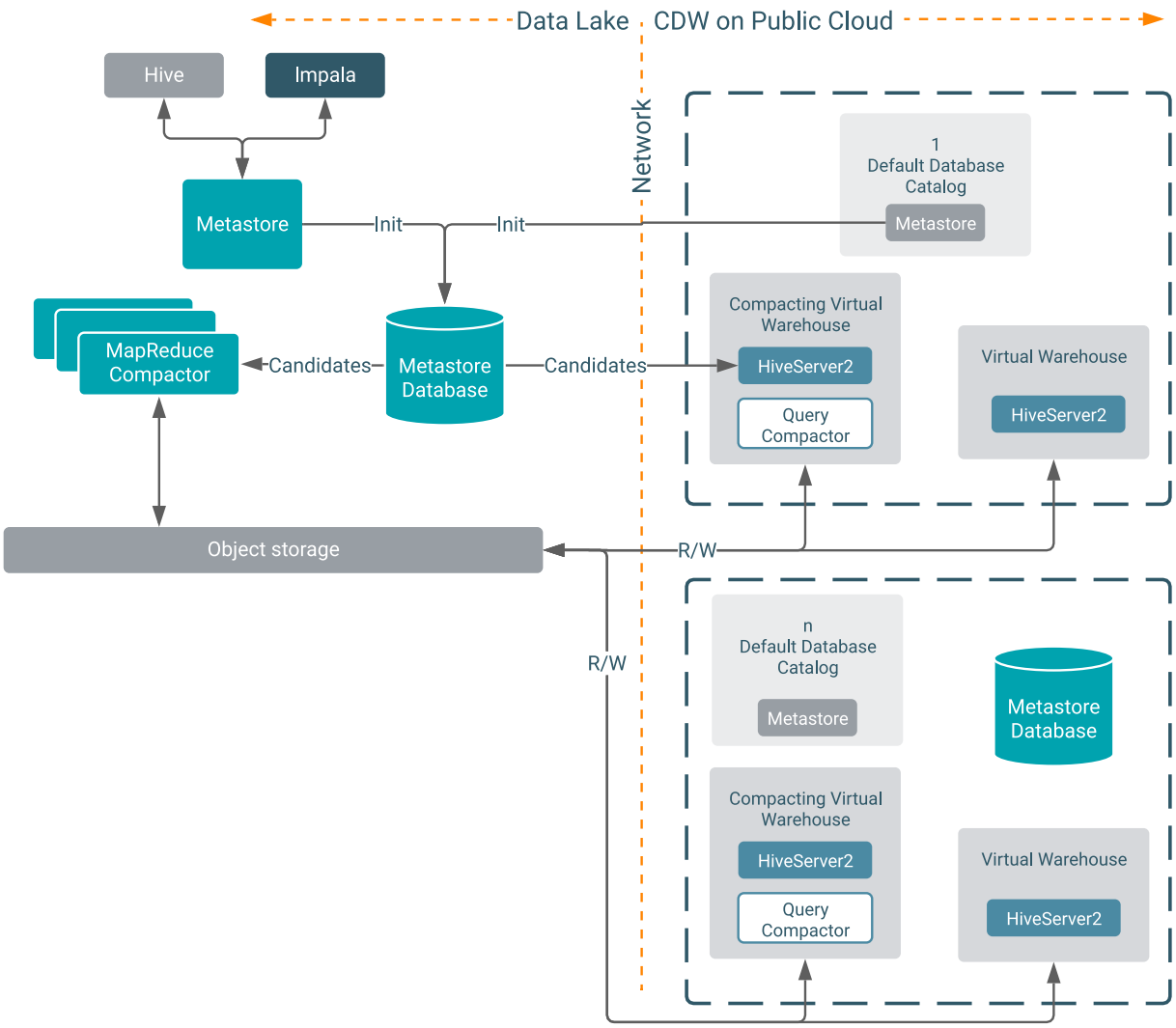

Compaction observability

Compaction observability is a notification and information system based on metrics about the health of the compaction process. A healthy compaction process is critical to query performance, availability, and uptime of your data warehouse. You learn how to use compaction observability to prevent serious problems from developing.

Compaction runs in the background. At regular intervals, Hive accesses the health of the compaction process and logs an error in the event of a problem. The assessment is based on metrics, such as the number of rows in the metadata table TXN_TO_WRITE_ID and the age of the longest running transaction (oldest_open_txn_age_in_sec). For example, if compaction isn't running the TXN_TO_WRITE_ID table in the HMS backend database becomes bloated and queries slow down. You use prebuilt Grafana dashboards to view alerts about compaction status, the issue, and recommended actions. The following list describes a few of more than 25 notifications:

- Oldest initiated compaction passed threshold
- Large number of compaction failures

- More than one host is initiating compaction

Compaction alerts use metrics to provide the following information to help you proactively address the problems before the problems become an emergency:

- Warnings and errors that suggest next steps
- Charted metrics
- Hive logging

Compaction observability does not attempt to do root cause analysis (RCA) and does not attempt to fix the underlying problem. Compaction observability helps you quickly respond to symptoms of compaction problems. Factors unrelated to compaction per se can look like a compaction problem. For example, an underlying problem related to renewing a Kerberos ticket problem can surface as a compaction problem. Configuring kerberos to add authorization, changing the running user, or increasing the queue size might solve the problem. Compaction observability provides troubleshooting information.

Compaction alerts are enabled by default in Cloudera Data Warehouse and the compaction health data is collected by default. Alerts place no load on Hive. The data about compaction health is not stored for very long, and is not stored in Hive. The data is emitted from Hive, and a backend thread, which is configurable to run as often as you want, collects metrics in Prometheus.

Create an environment in this release to pick up this feature.

Viewing a compaction alert using Grafana

Learn how to access Grafana dashboards from Cloudera Data Warehouse to view compaction alerts and take necessary actions to keep your cluster healthy.

Before you begin

You must be able to log into Grafana from Cloudera Data Warehouse as described in the *Getting started in Grafana* topic.

About this task

Perform the following steps to access Grafana from Cloudera Data Warehouse:

Procedure

1. In the Cloudera Data Warehouse service, click Overview and go to the Environments tab.
2. Locate the environment that you activated for Cloudera Data Warehouse, which has the default Database Catalog.
3. Click ⋮ Open Grafana .
4. In the Grafana web interface, click ⊞ and then select Dashboards.

   A list of dashboard groups is displayed.

5. Locate and click the Hive - Compaction observability dashboard group to view all the dashboards related to Hive compactions.

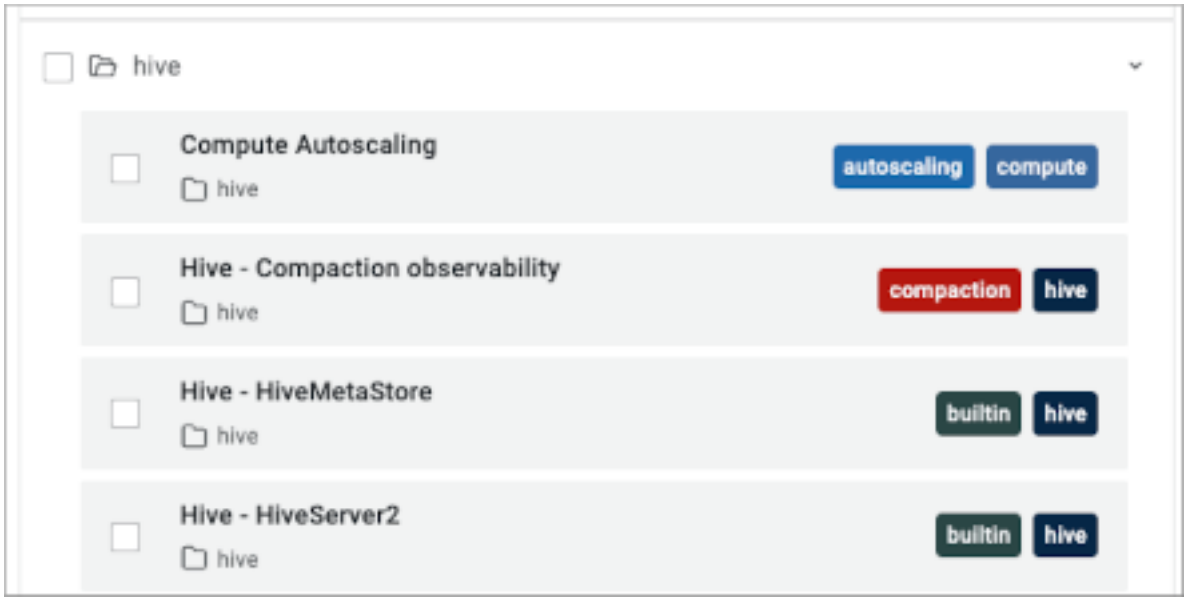

Related Information

Getting started in Grafana
Viewing prebuilt Grafana dashboards

Adding a compaction alert

You can define compaction alert rules for your Cloudera Data Warehouse deployment based on Prometheus expressions. The alerts are automatically triggered when specific events occur in your deployment.

About this task

You use PromQL expressions to express the conditions that trigger the alert, as shown in the following procedure.

Procedure

1. In the Cloudera Data Warehouse service, click Overview and go to the Environments tab.
2. Locate an environment that you activated for Cloudera Data Warehouse, which has the default Database Catalog.
3. Click ⋮ Edit .
4. In the **Environment Details** page, click the ALERT SETTINGS tab and then click Add New.
5. Enter the following information for the new alert:
   - Alert ID
   - Name of the alert
   - An error or warning message to display when the alert is triggered
   - The Prometheus expression to specify conditions that trigger the alert
   - Duration to specify how long the conditions last before the alert is triggered
   - Severity of the alert — Error or Warning
6. Click Create.

Related Information

Querying Prometheus

Managing a compaction alert

You can manage compaction alerts in Cloudera Data Warehouse by modifying alert details, such as the alert name, message, expression that defines threshold conditions for the alert, frequency, and severity of the alert.

Procedure

1. In the Cloudera Data Warehouse service, click Overview and go to the Environments tab.
2. Locate an environment that you activated for Cloudera Data Warehouse, which has the default Database Catalog.
3. Click ⋮ Edit .
4. In the **Environment Details** page, click the ALERT SETTINGS tab and then click ⋮ Edit for the alert that you want to modify.

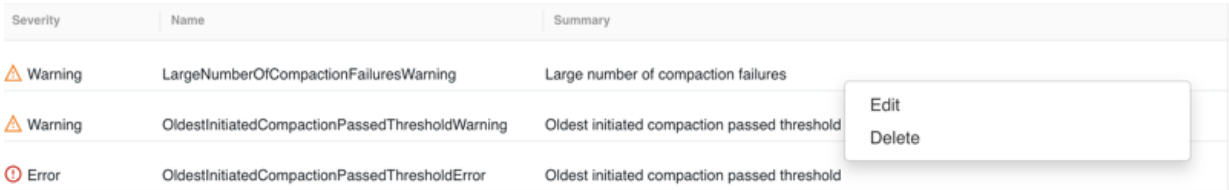

5. In the **Edit Alert Details** page, modify the required properties to redefine the alert.

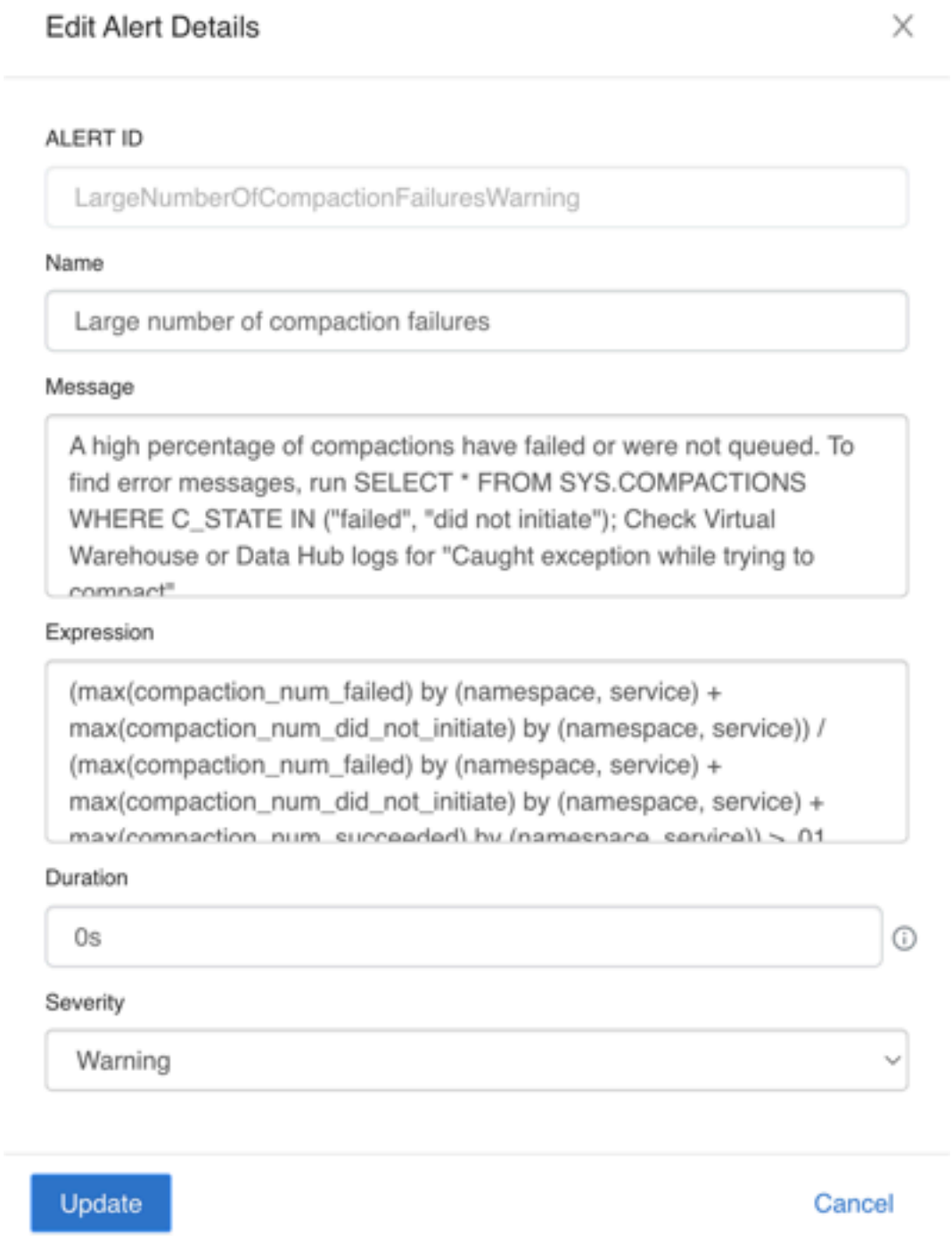

6. Click Update.

Configuring alert notifications

As Administrator, you need to learn how to configure alert notifications that appear in the user notification system. The notifications supplement alert information in charts on the Manage/Hive/Compaction Observability dashboard in Grafana. You learn to use the UI for this configuration, but alternatively, you can use the Kubernetes configuration method.

About this task

In this task, you see how to funnel alerts into a user notification system. In the next task, you see how to use the Kubernetes configuration method.

Procedure

1. In the Cloudera Data Warehouse service, click Overview and go to the Environments tab.
2. Locate an environment that you activated for Cloudera Data Warehouse, which has the default Database Catalog.
3. Click ⋮ Edit .
4. In the **Environment Details** page, click the ALERT SETTINGS tab.
5. In Alert Manager Configuration, configure the receiver object by following instructions in the Prometheus AlertManager v.0.20.0 documentation to configure the receiver object.
   For example, a simple WebHook configuration looks something like this:

```
global:
  resolve_timeout: 5m

route:
  group_by: ['alertname', 'namespace', 'service']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 3h
  receiver: notification_system

receivers:
  - name: 'notification_system'
    webhook_configs:
      - url: https://co-alerting.prod.mycompany.com
        send_resolved: true

inhibit_rules:
  - source_match:
      severity: 'error'
    target_match:
      severity: 'warning'
    equal: ['alertid', 'namespace', 'service']
```

   For information about integration possibilities, see the receiver section.
6. Click Save, and then Apply Changes.
   The alertmanager recognizes the change and automatically updates the configuration.

Configuring alert notifications, no UI

You learn to use the Kubernetes configuration method of configuring alert notifications that appear in the user notification system. Configuring notifications using this method has the same effect as using the UI method. The notifications supplement alert information in charts on the Manage/Hive/Compaction Observability dashboard in Grafana.

About this task

In this task, you see how to funnel alerts into a user notification system using a Kubernetes configuration available from your Database Catalog.

Procedure

1. In the Cloudera Data Warehouse service, click Overview and go to the Environments tab.
2. Locate an environment that you activated for Cloudera Data Warehouse, which has the default Database Catalog.

3. Click ⋮ Show Kubeconfig .

   The **Kubeconfig for environment** modal is displayed.
4. Click Copy Kubeconfig and paste the configuration into a file, and save the file as a YML file, for example, kube_env.yml.
5. On the command line of the cluster, set the KUBECONFIG environment variable to the path of the YML file.

```
$ export KUBECONFIG=<the-absolute-path-to>/kube_env.yml
```

6. Download the alert configuration using a command to fetch the configuration into the alertmanager.yml local file.

```
$ kubectl \
    -n istio-system \
    get configmap \
    alertmanager \
    -o "jsonpath={.data['alertmanager\.yml']}" > alertmanager.yml
```

```
global:
  resolve_timeout: 5m

route:
  group_by: ['alertname', 'namespace', 'service']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 3h
  receiver: notification_system

receivers:
  - name: 'notification_system'

inhibit_rules:
  - source_match:
      severity: 'error'
    target_match:
      severity: 'warning'
    equal: ['alertid', 'namespace', 'service']
```

7. Edit the alert configuration to configure the receiver object by following instructions in the Prometheus AlertManager v.0.20.0 documentation.

   For example, a simple WebHook configuration looks something like this:

```
global:
  resolve_timeout: 5m

route:
  group_by: ['alertname', 'namespace', 'service']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 3h
  receiver: notification_system

receivers:
  - name: 'notification_system'
    webhook_configs:
      - url: https://co-alerting.prod.mycompany.com
        send_resolved: true

inhibit_rules:
  - source_match:
      severity: 'error'
    target_match:
```

```
      severity: 'warning'
    equal: ['alertid', 'namespace', 'service']
```

For information about integration possibilities, see the receiver section.

8. Check that your configuration conforms to YAML syntax.
9. Update the related configmap.
   For example:

```
$ kubectl \
    -n istio-system \
    create configmap \
    alertmanager \
    --from-file=alertmanager.yml=./alertmanager.yml \
    --dry-run=client -o yaml \
 | kubectl apply -f -
```

The alertmanager recognizes the change and automatically updates the configuration.

Checking an alert notification configuration

You learn how to ensure your alert notification configuration is in effect. You fetch the configuration directly from the alertmanager and check that it is reachable.

Procedure

1. Port-forward the prometheus service.

```
$ kubectl \
    -n istio-system \
    port-forward \
    service/prometheus \
    9093:9093
```

2. From another terminal the configuration is reachable by using the following curl command.

```
$ curl http://localhost:9093/api/v2/status
```

3. Check that the output shows your configuration.

   Alternatively, to check the output, you can use the alertmanager WebUI by navigating to http://localhost:9093/#/status.

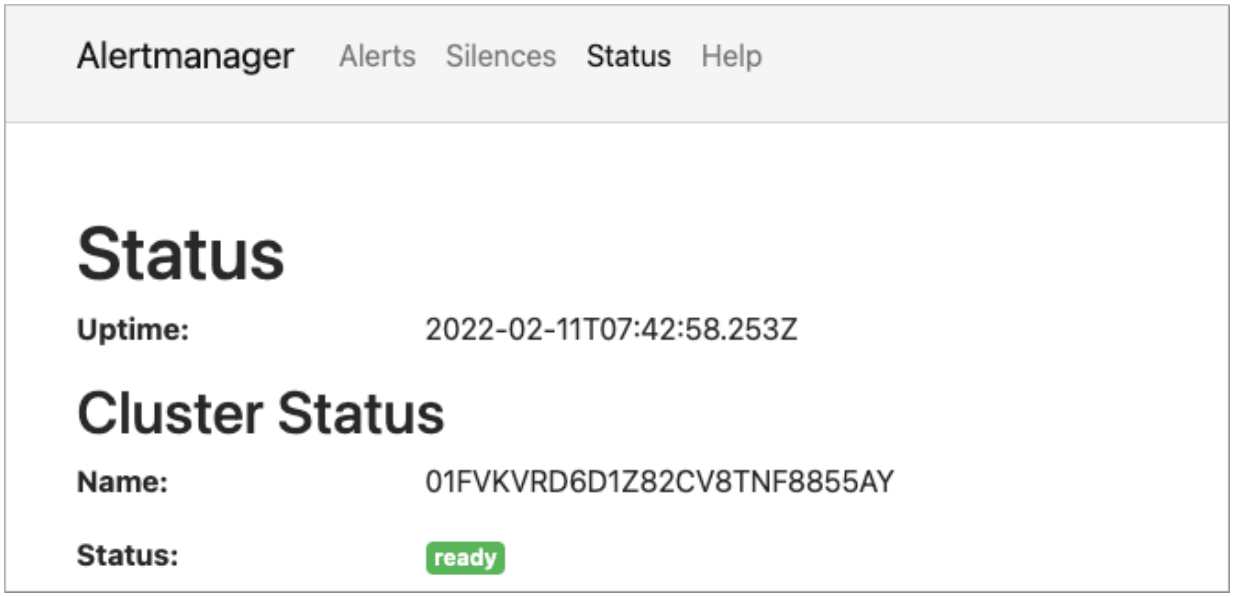

Deleting a compaction alert

Learn how to delete an user-defined or default compaction alert that you no longer require for your Cloudera Data Warehouse deployment.

Procedure

1. In the Cloudera Data Warehouse service, click Overview and go to the Environments tab.
2. Locate an environment that you activated for Cloudera Data Warehouse, which has the default Database Catalog.
3. Click ⋮ Edit .
4. In the **Environment Details** page, click the ALERT SETTINGS tab, and then click ⋮ Delete for the alert that you want to delete.
5. Click OK to confirm the delete action.

Data Visualization in Cloudera Data Warehouse

Cloudera Data Warehouse integrates Data Visualization for building graphic representations of data, dashboards, and visual applications based on Cloudera Data Warehouse data, or other data sources you connect to. You, and authorized users, can explore data across the entire Cloudera data lifecycle using graphics, such as pie charts and histograms. You arrange visuals on a dashboard for collaborative analysis.

You connect Data Visualization to a Virtual Warehouse as described in Starting Data Visualization integrated in Cloudera Data Warehouse. Similar to using a BI client, you can configure and connect to Virtual Warehouses from different clusters. You configure the connection in a familiar way, providing an IP address or host name. Data Visualization is not tied to a particular Virtual Warehouse (VW). You can access data for your visualization from multiple Data Catalogs using multiple Hive or Impala Virtual Warehouses and multiple environments.

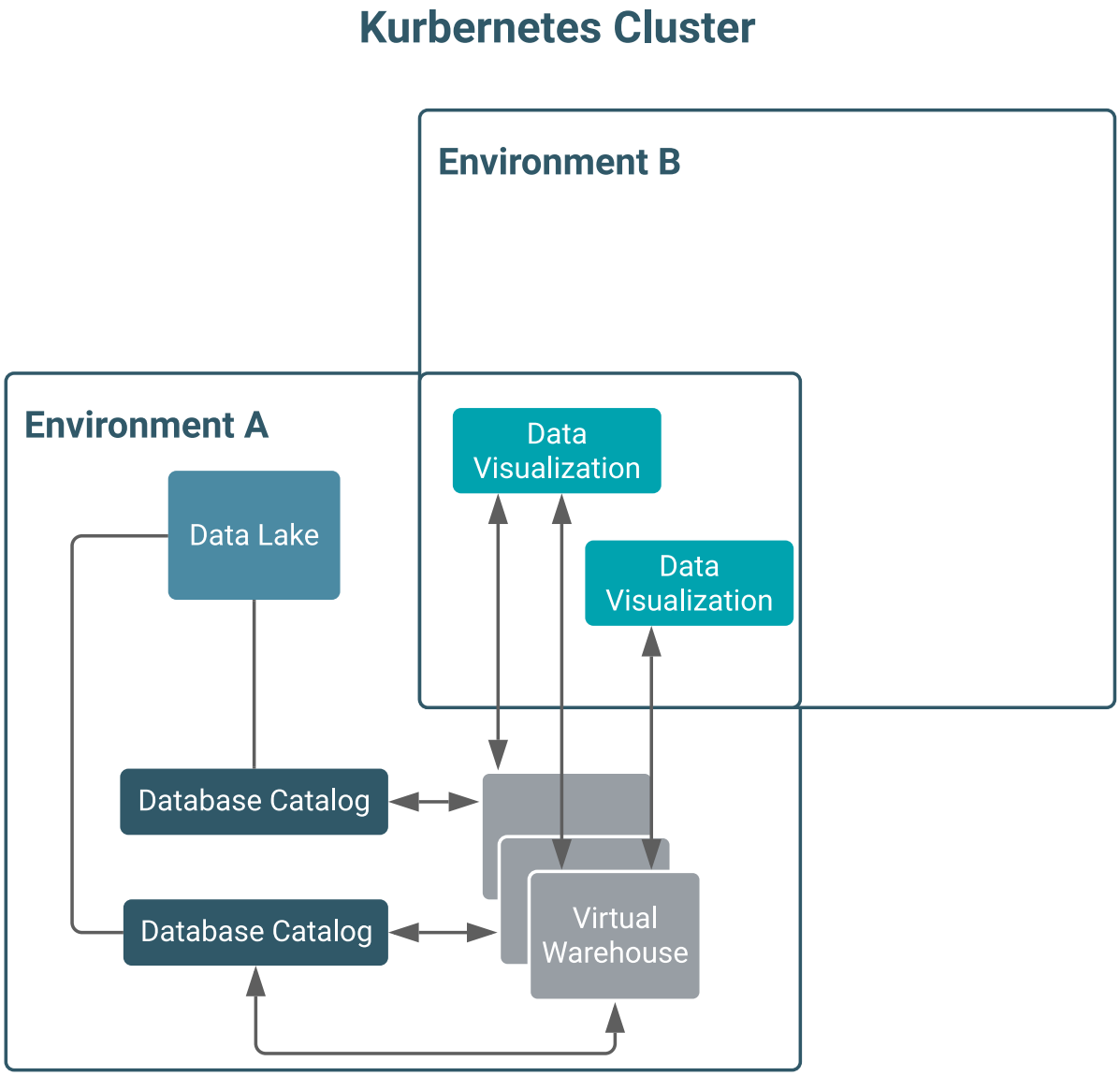

Having multiple Data Visualization instances attached to an environment, you can create dashboards for different groups. For example, Marketing and Sales can have their own private dashboards. When you delete a Virtual Warehouse, your visuals remain intact.

Related Information

Cloudera Data Visualization

Creating a visual

Working with datasets

Rebuilding a Database Catalog

You can clean up resources and redeploy your Database Catalog while preserving the image version. Rebuild your Database Catalog to perform housekeeping or troubleshooting a problem.

About this task

Rebuilding the Database Catalog redeploys resources in the cluster while keeping the configurations and other settings you made using either the Cloudera Data Warehouse UI or CDP CLI.

You can rebuild the Database Catalog in the following ways:

- Using the Beta version of Cloudera Data Warehouse CLI
- Using the Rebuild option in the Cloudera Data Warehouse UI

Limitations

- Rebuilding deprecated versions of the Database Catalog is not supported and will fail.
- Rebuilding does not preserve the changes you made directly, using kubectl for example, to cluster resources in the affected namespaces.

Before you begin

- You must obtain the DWAdmin role.
- You must be running the version 2022.0.6-b92 or later of the Database Catalog to rebuild it.
- To preserve Hue documents, such as saved queries, you need to save a backup of the documents as described in *Backing up and restoring Hue documents*.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Database Catalogs tab.
3. Locate the Database Catalog you want to rebuild and click ⋮ Rebuild .

   The **Rebuild Database Catalog** modal is displayed.
4. Click Rebuild Database Catalog.
5. Restore Hue documents as described *Backing up and restoring Hue documents*.

Results

The Database Catalog is rebuilt with the same image version as it had before.

What to do next

If you do not want to change JDBC/ODBC URLs, for example env-xdd4x6.dw, when rebuilding the Database Catalog, open a Support case to have the CDW_CUSTOM_CLUSTER_ID entitlement enabled.

Backing up and restoring Hue documents

You can back up and restore Hue data, such as a Hive or Impala query you saved in a JSON document, in the Hue database to prevent losing the documents after rebuilding the Database Catalog.

About this task

In Hue, you can view the JSON documents you created or imported in My documents.

Before you begin

- Avoid accessing or using the Hue web interface until the restore process is fully complete, as concurrent access may disrupt the operation and potentially lead to failure.

- During the Hue database restore operation it is critical to block any traffic to the running Hue services. If you cannot bring down the cluster, use the recommended workaround to disable end user access to the cluster endpoints. Failing to do so results in errors in addition to existing key constraints and other issues.

Procedure

1. Backup Hue documents you want to preserve from the command line.

```
# on hue pod
./build/env/bin/hue dumpdata -o /tmp/data.json
# on local machine
kubectl --kubeconfig ~/kconfig cp compute-1675582336-knmf/huebackend-0:/
tmp/data.json -c hue ./data.json
```

2. Rebuild the Database Catalog as described in the previous topic.
3. Restore the Hue documents in the rebuilt Database Catalog.

```
#on local machine
kubectl --kubeconfig ~/kconfig cp ./data.json compute-1675582336-knmf/hueb
ackend-0:/tmp/data.json -c hue

# on new hue pod
./build/env/bin/hue loaddata --ignorenonexistent /tmp/data.json
```

Rebuilding a Virtual Warehouse

You can clean up resources and redeploy your Virtual Warehouse while preserving its image version. Rebuild your Virtual Warehouse to perform housekeeping or troubleshooting a problem.

About this task

Rebuilding the Virtual Warehouse redeploys resources in the cluster while preserving the configurations and other settings you made using the Cloudera Data Warehouse UI or CDP CLI.

You can rebuild the Database Catalog in the following ways:

- Using the Beta version of Cloudera Data Warehouse CLI
- Using the Rebuild option in the Cloudera Data Warehouse UI

Limitations

- Rebuilding deprecated versions of the Virtual Warehouse is not supported and will fail.
- Rebuilding does not preserve the changes you made directly, using kubectl for example, to cluster resources in the affected namespaces.

Before you begin

- You must obtain the DWAdmin role.
- You must be running the version 2022.0.6-b92 or later of the Virtual Warehouse to rebuild it.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Virtual Warehouses tab.
3. Locate the Virtual Warehouse you want to rebuild and click ⋮ Rebuild .

   The **Rebuild Virtual Warehouse** modal is displayed.

4. Click Rebuild Virtual Warehouse.

Results

The Virtual Warehouse is rebuilt with the same image version and configurations as it had before.

Upgrading Database Catalogs and Virtual Warehouses in Cloudera Data Warehouse Public Cloud

You need to learn about the choices, requirements, and alerts for upgrading Database Catalogs and Virtual Warehouses in Cloudera Data Warehouse Public Cloud.

Before you begin the upgrade

- Obtain the DWAdmin role.
- Take a look at the release notes for the following information:
  - Version mapping
  - What's New
  - Known issues
  - Fixed issues
  - Behavior changes
  - Older releases

Text and JSON configuration files

After upgrading, text and JSON configuration files in your upgraded Database Catalog and Virtual Warehouse are not upgraded. The pre-upgrade configurations are carried over to the upgraded Database Catalog and Virtual Warehouse. For example, in the Database Catalog Configurations , the .yml file example below is not upgraded.

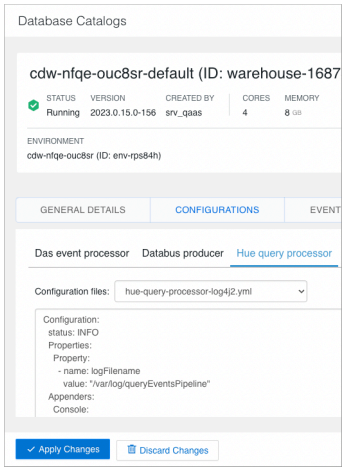

Configurations, such as key/value configurations, that are not in a text or JSON file are not affected by this limitation. These configurations are upgraded.

Upgrade available indicator

Database Catalogs and Virtual Warehouse must be compatible. The Cloudera Data Warehouse upgrade framework understands the interoperability constraints between them, and generates alerts. Alerts appear to indicate when upgrades are available for the Database Catalog and Virtual Warehouse as showin in the following image:

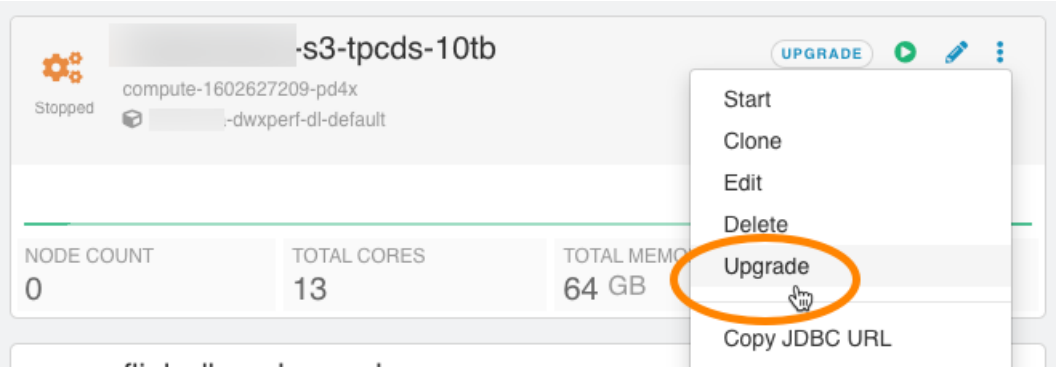

Latest releases versus compatible releases

A release is considered *compatible* with the current release if its dependents or dependencies do not need to be upgraded. Compatible Database Catalogs do not need to be upgraded to function effectively with associated Virtual Warehouses. Compatible Virtual Warehouses do not need to be upgraded to function effectively with the associated Database Catalog.

Some releases contain changes that require upgrading associated components. For example, a Hive Virtual Warehouse might use new APIs introduced in a new release of the Hive metastore/Database Catalog. In this case, you must upgrade the Database Catalog that is associated with this Hive Virtual Warehouse. On the other hand, if you upgrade a Database Catalog, you are not required to upgrade its associated Virtual Warehouse. The Latest releases indicator appears in this case.

Available upgrade choices

When you click Upgrade, you might see one of the following options:

- Already up-to-date, which means there are no new releases that you can upgrade to.
- Upgrade to the latest version:

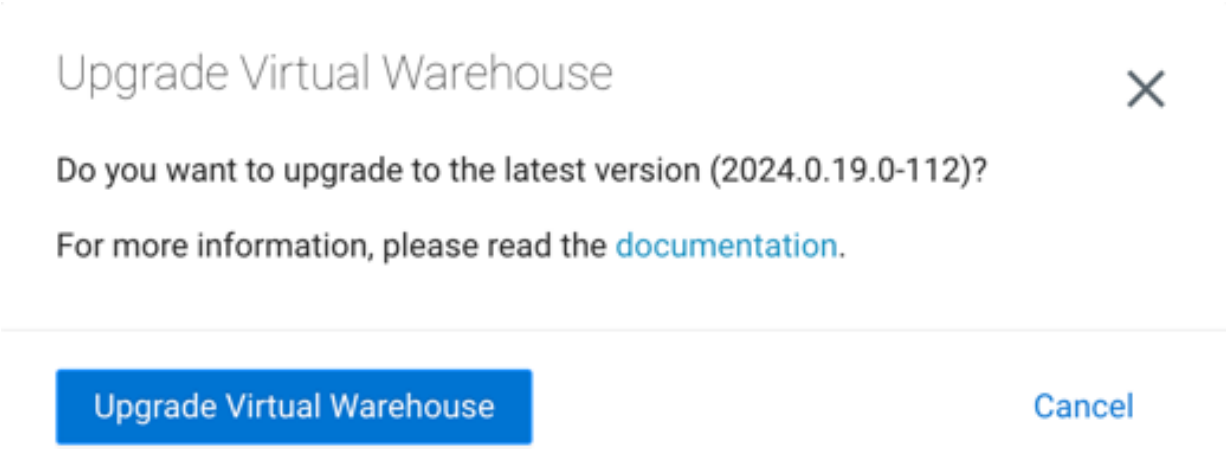

Click Upgrade Virtual Warehouse to start the upgrade process. The status of the upgrade appears: Upgrading, Starting, or Started.

Difference between upgrade and rebuild

Starting with the Cloudera Data Warehouse 1.8.5-b35 release, upgrade and rebuild operations perform different tasks. Upgrade operation updates the image version of the Cloudera Data Warehouse entity. Rebuild operation deletes and recreates the pods while preserving the image version and configutrations.

Debugging Impala Virtual Warehouses using Web UIs

You can use the Catalog Web UI, Coordinator Web UI, and the StateStore Web UI to debug Impala Virtual Warehouses in Cloudera Data Warehouse.

About this task

The Impala daemons (impalad, statestored, and catalogd) debug Web UIs, which can be used in by using , is also available in the service. In service, the following Web UIs are provided:

- Impala Catalog Web UI

  This UI provides the same type of information as the Catalog Server Web UI in . It includes information about the objects managed by the Impala Virtual Warehouse. For more information about this debug Web UI, see Debug Web UI for Catalog Server.
- Impala Coordinator Web UI

  This UI provides the same type of information as the Impala Daemon Web UI in . It includes information about configuration settings, running and completed queries, and associated performance and resource usage for queries. For information about this debug Web UI, see Debug Web UI for Impala Daemon.
- Impala StateStore Web UI

  This UI provides the same type of information as the StateStore Web UI in . It includes information about memory usage, configuration settings, and ongoing health checks that are performed by the Impala statestored daemon. For information about this debug Web UI, see Debug Web UI for StateStore.
- Impala Autoscaler Web UI

  This UI gives you insight into autoscaler operations (regular as well as workload-aware autoscaling), accessing log messages, and resetting the log level. The autoscaler Web UI includes information about the queries queued and running, executor groups, suspended calls, scale up/down calls, the autoscaler config, and the autoscaler logs.

Required role: EnvironmentAdmin

Before you begin

Make sure that you note your workload user name and have set a password for it in the User Management module of . You need to use your workload user name and its associated password to log into the debug Web UIs. For more information, see Setting the workload password in the documentation set.

Procedure

1. In the UI on the Overview page, locate the Impala Virtual Warehouse for which you want to view the debug UIs, and select ⋮ Edit .

   The **Virtual Warehouse Details** page is displayed.
2. In the **Virtual Warehouse Details** page, select the Web UI tab. The list of debug Web UI links are displayed as shown in the following image:

   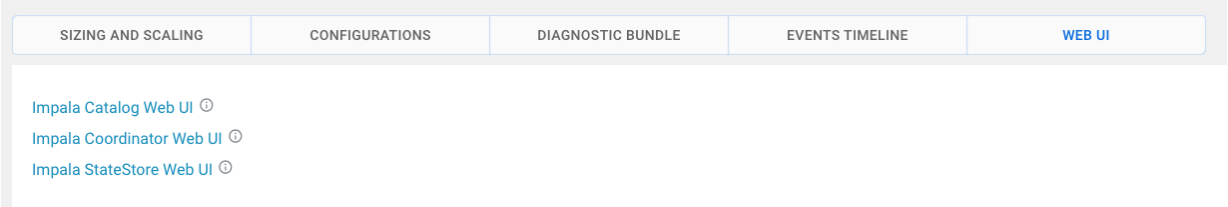

3. Click a Web UI link corresponding to an Impala daemon that you want to debug.

   You are prompted to enter your workload user name and password.

Results

After you are authenticated, you can view the debug Web UI and use the information to help you troubleshoot issues with your Impala Virtual Warehouse.

Cloudera Data Warehouse Public Cloud

Monitoring

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Monitoring resources with Grafana..... 4
Grafana in Cloudera Data Warehouse overview..... 4
Limitations of Grafana in Cloudera Data Warehouse..... 8
Getting started in Grafana..... 8
Viewing prebuilt dashboards..... 10
Monitoring HMS..... 14
Monitoring key Hive metrics..... 18
Monitoring Impala executors..... 21
Monitoring Impala catalog..... 24
Monitoring Impala statestore..... 26
Monitoring Impala admission control..... 27
Monitoring Impala coordinators..... 28
Monitoring nodes..... 29
Monitoring Kubernetes Services from Grafana..... 34
Updating a dashboard graph..... 38
Inspecting dashboard data and queries..... 42
Creating a custom dashboard..... 44
Meeting prerequisites to set up alerts..... 49
Creating alerts..... 57
Reviewing alerts and notifications..... 59

Forwarding Prometheus metrics from Cloudera Data Warehouse to an endpoint..... 62

Monitoring Kubernetes resources from K8s dashboard..... 63
Activating the K8S dashboard..... 63
Using the K8S dashboard..... 64

Forwarding logs to your observability system..... 68
Providing proxy CA certificates..... 73

Monitoring resources with Grafana

Grafana is visualization and analytics software for using dashboards to monitor metrics data. You learn how to access pre-built Grafana dashboards to monitor Virtual Warehouses and your compute cluster in Cloudera Data Warehouse. As a Cluster Operations professional, you visualize Kubernetes and Istio metrics from Grafana to monitor and maintain the cluster.

When you log into Grafana, the Welcome page contains links to documentation about Grafana basics. From the left navigation, you can get started by searching for a particular dashboard or hovering over the navigation icons to familiarize yourself with the UI functions.

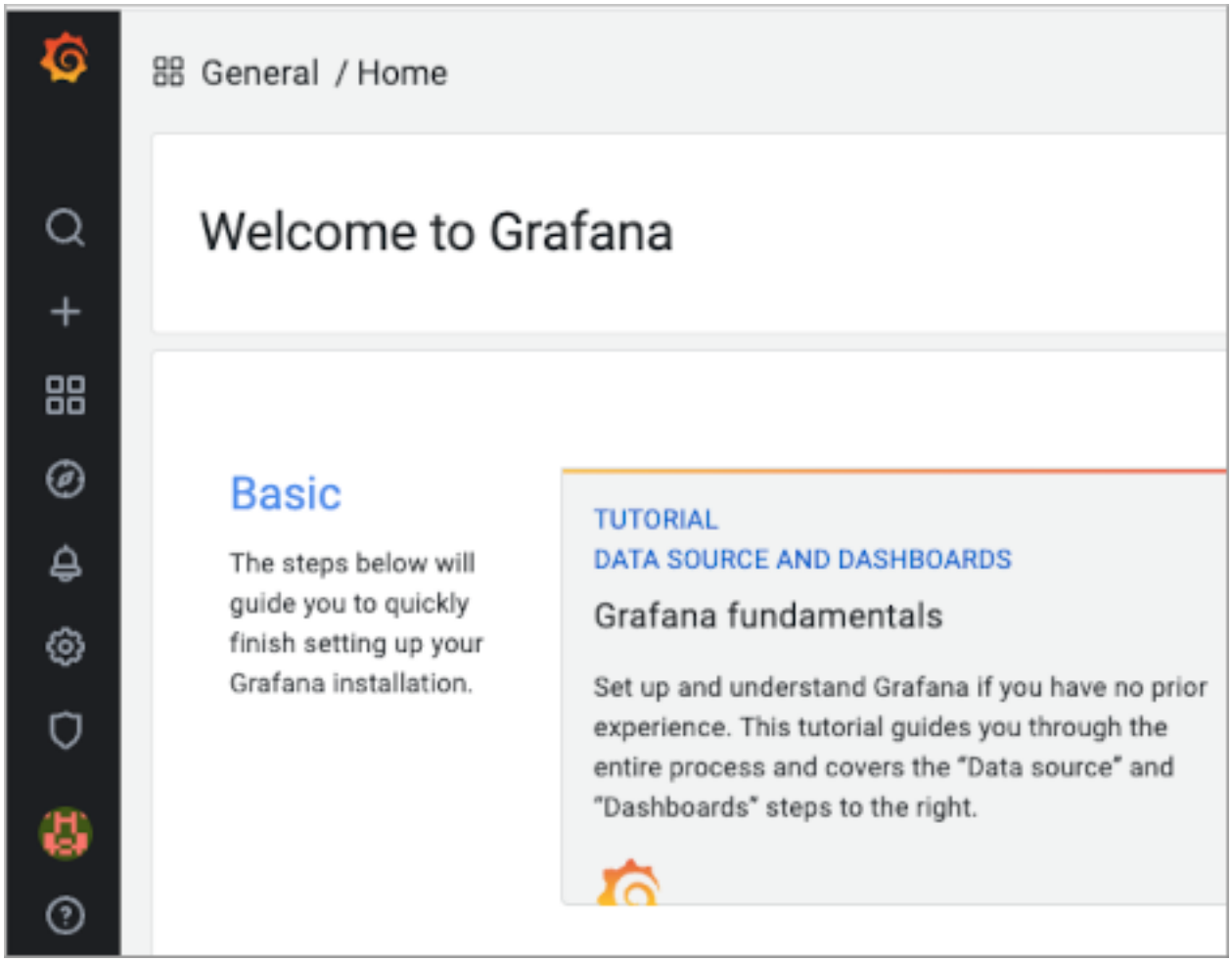

As a new user, to familiarize yourself with the pre-built dashboards that chart metrics about Hive, Impala, Hue, Druid, Kubernetes, and Istio in Cloudera Data Warehouse, follow steps in this documentation.

Related Information

Grafana documentation

Grafana in Cloudera Data Warehouse overview

You connect to prebuilt dashboards to view metrics of Cloudera Data Warehouse operations. Cloudera provides prebuilt Grafana dashboards for Hive, Impala, Hue, Druid, Kubernetes, and Istio dashboards of metrics data, charts, and other visuals.

Using Grafana, Cloudera metrics are centralized in a single spot, stored in the Prometheus database and monitored by Prometheus. Your workload databases are not involved in any way. You can immediately view pre-built dashboards described below. You can view dashboard metrics for different time periods by selecting the period of interest from the dropdown in the horizontal navigation:

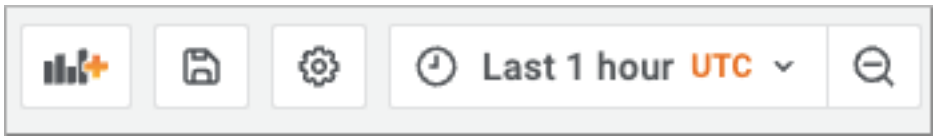

You can also create your own dashboards. "Get started with Grafana and Prometheus" describes how to create dashboards of Cloudera metrics. Describing all the details of how to use Grafana is beyond the scope of this documentation. Grafana described in this documentation is not the enterprise version.

Hive dashboards

The Hive dashboards cover the following operations of the Hive SQL engine in Cloudera Data Warehouse:

- Auto-scaling
- Hive metastore
- HiveServer
- The Hive service itself (Hive-Home)
- LLAP

For example:

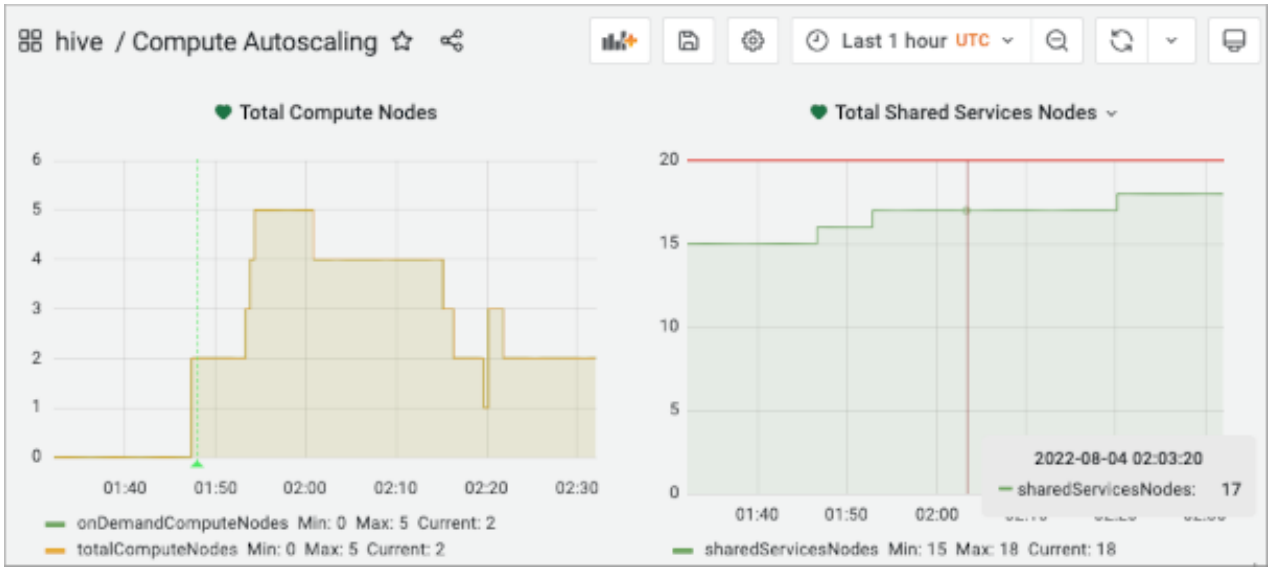

Impala dashboards

The Impala dashboards include the following operations of the Impala SQL engine in Cloudera Data Warehouse:

- Catalog server
- Coordinator
- Executor
- Statestore
- The Impala service itself

Kubernetes dashboards

You can get insight into the operations of your Cloudera clusters from Kubernetes dashboards. Kubernetes dashboards represent the following metrics:

- CoreDNS: requests and duration of responses
- App Metrics: number of Kubernetes pods, CPU usage relative to request or to limit, memory usage sliced and diced a number of ways
- Object Metrics: Deployment memory and CPU usage, in total, and by node.

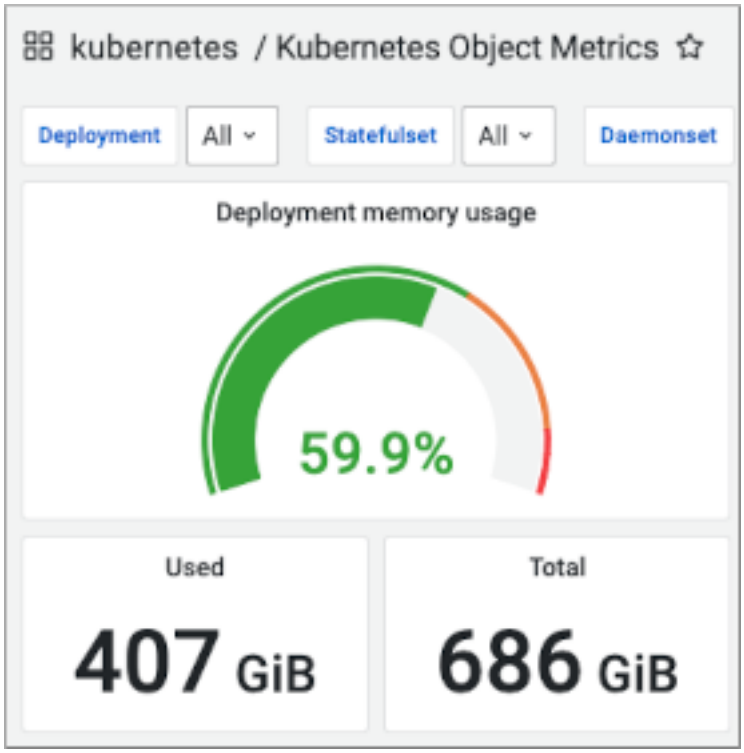

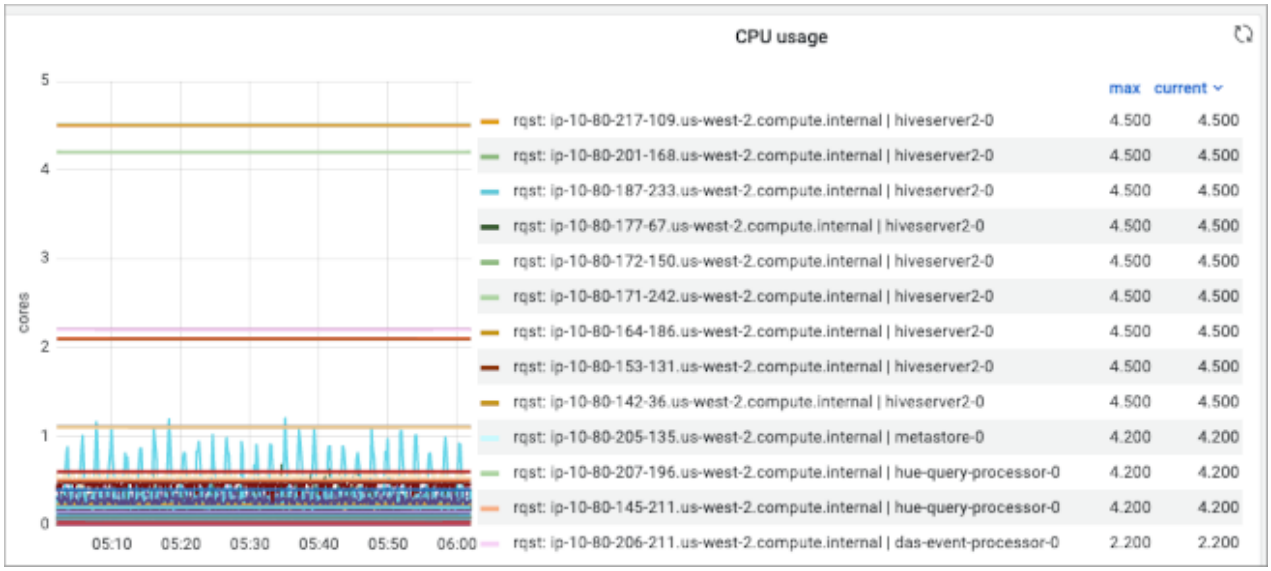

Istio dashboards

To work with Cloudera Support, you might use the Istio dashboards. Istio is an open platform that provides microservice security, connections, and monitoring. The Istio Mesh dashboards cover the following views of the service mesh network of microservices:

- Istio Mesh summary: describes the network of microservices by HTTP/gRPC and TCP workloads in the Mesh.
- Individual Cloudera services, such as HiveServer and Impala coordinator: Request and response metrics, such as latency, for each mesh service (HTTP/gRPC and TCP) and client and service workloads metrics.

- Individual workloads using the services: Request and response metrics for each workload in the Mesh (HTTP/gRPC and TCP) and inbound/outbound workload services.

The Istio performance dashboard presents visualizations of the following metrics:

- Cloudera Usage
- Memory Usage
- Data Rates
- Bytes transferred per second

The Istio control plane dashboard includes memory, CPU, and disk resource usage.

For example:

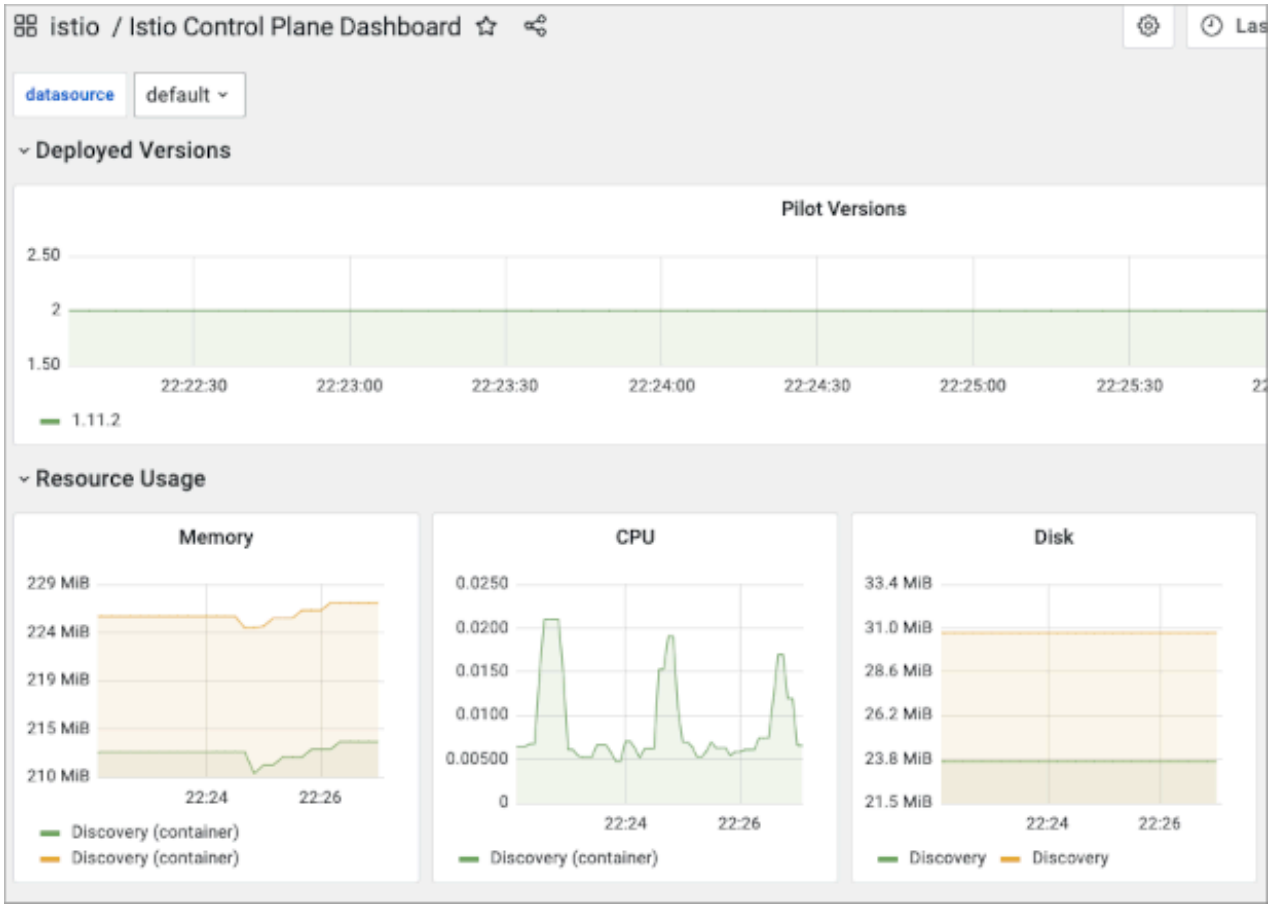

The sidecar proxy metrics in Istio dashboards reveals the interceptions of network communication between microservices.

The Istio Wasm Plugin extends the Istio proxy capabilities.

Nodes dashboards

The Prometheus Node Exporter is used to gather detailed metrics for the AWS and Azure Virtual Machines that host an environment's Kubernetes cluster. These metrics cover both machine and OS level metrics such as CPU, memory, network, processes, time synchronization, disk, and file system. All metric names start with node_.

Three prebuild dashboards show metrics for AWS/Azure Virtual Machines that host the Kubernetes cluster:

- Cluster Totals

  Shows CPU/Memory utilization and node counts (shared services/compute for the entire cluster.

- Node Details

  Shows very detailed metrics for a single node at a time.
- Node Trends

  Combines CPU, Memory, Disk, and Network metrics for all nodes for node-to-node comparisons.

Related Information

Grafana documentation
Get started with Grafana and Prometheus
Istio documentation
Istio in-depth documentation
Istio github site
Wasm Plugin

Limitations of Grafana in Cloudera Data Warehouse

You need to understand the Grafana capabilities in Cloudera Data Warehouse that Cloudera does not support. Grafana in Cloudera Data Warehouse is intended for use by cluster operations professionals who are familiar with monitoring tools, interpreting metrics, and performing maintenance.

Unsupported features

Storing metrics longer than 15 days or consuming more than 90GB of disk space is not supported. Metrics older than 15 days will be deleted. If the stored metrics consume more than 90GB of disk space, metrics will be deleted regardless of the number of days stored.

Custom dashboards you create in Grafana are lost upon restarting or updating the cluster. Only the default dashboards are supported.

Getting started in Grafana

Learn how to log into Grafana in Cloudera Data Warehouse Public Cloud.

Before you begin

- To access Grafana, your Cloudera administrator must use Cloudera Management Console to assign the environment to you as a resource.
- You must obtain the DWAdmin role.

Procedure

1. In Data Warehouse Overview , expand the Environment column.

2. Search for the environment that is associated with the Virtual Warehouse, click more options ⋮ in the environment tile, and select Open Grafana.

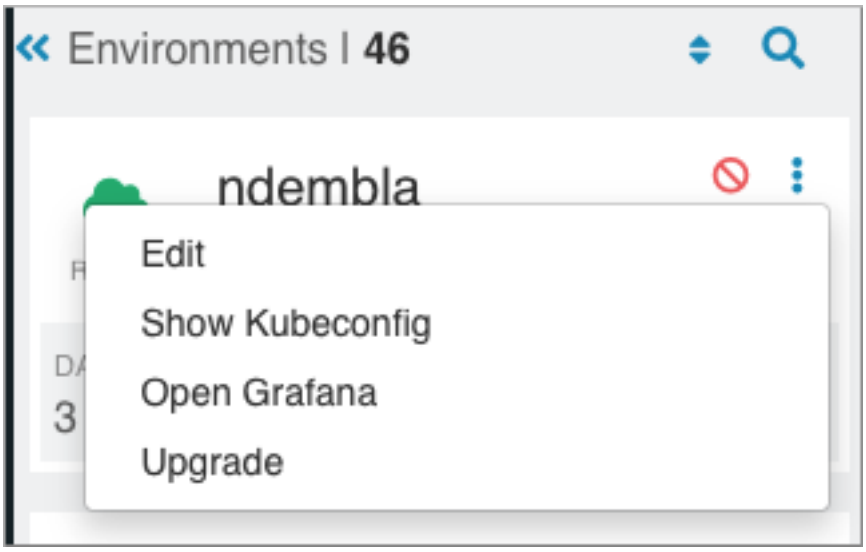

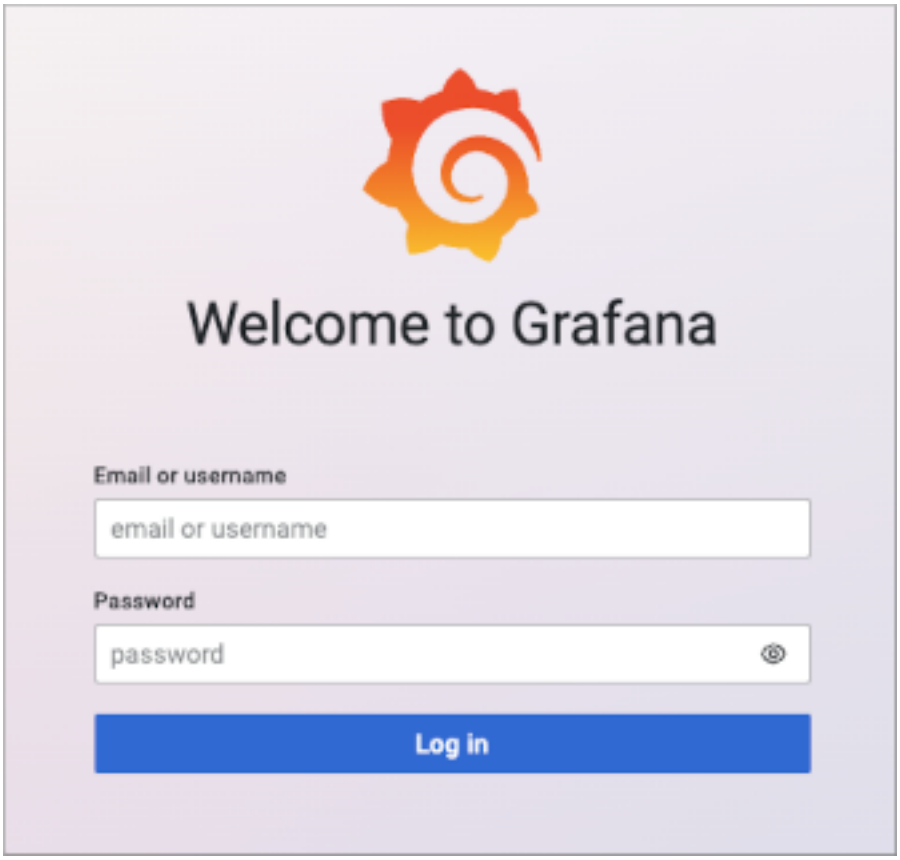

3. Enter your LDAP user name, or the name provided by your Administrator for logging into Grafana.
4. Go back to the environment tile, and select Copy Kubeconf.
5. Open a terminal window, paste the contents of your clipboard into a text file, and save the file by the name kube conf (no extension).
6. Export the KUBECONFIG variable equal to the file name.

```
export KUBECONFIG=kubeconf
```

7. Get the password for Grafana.
   For example, use the kubectl command as follows:

```
kubectl get secret grafana -n istio-system -o jsonpath="{.data.passphras
e}" | base64 -D | pbcopy
```

8. Go back to the Grafana login dialog, and paste the contents of your clipboard.
   The Grafana login dialog should look something like this:

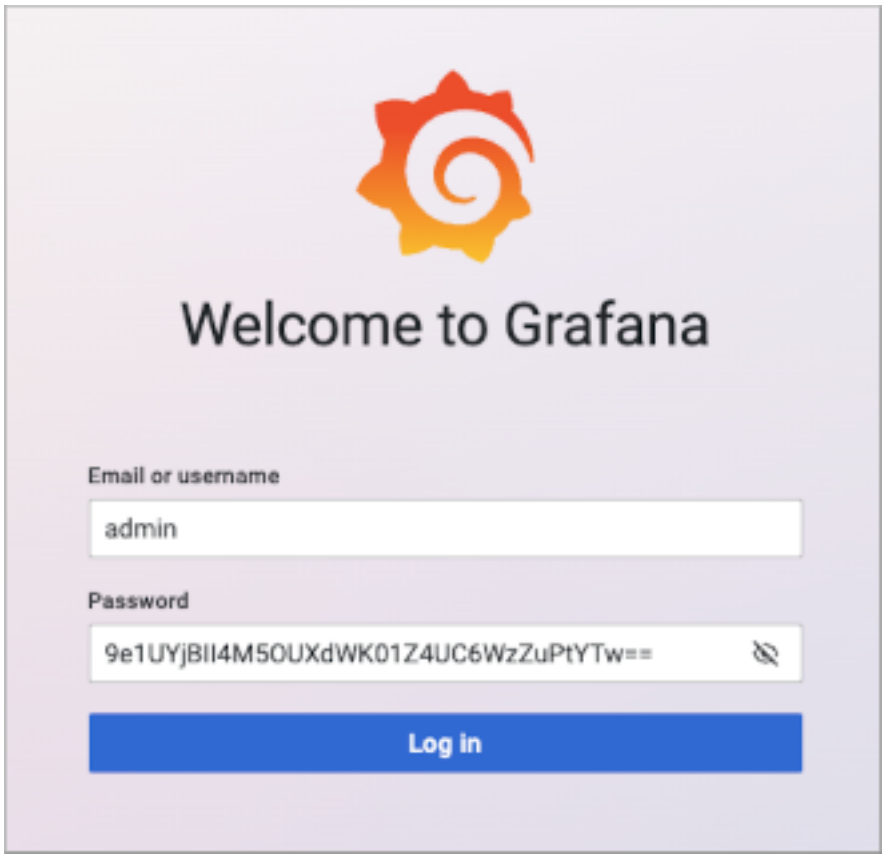

9. Click Log in.

Related Information

Viewing prebuilt dashboards

You see how to list dashboard groups, view the Hive dashboard, and see the actual metric data on a point in the X-axis. You can follow the same steps to work with other dashboards.

Procedure

1. Log into Grafana as described in the previous topic, "Getting Started in Grafana".

2.

In the Welcome screen, click grid , and then select Manage.
A list of dashboard groups appears:

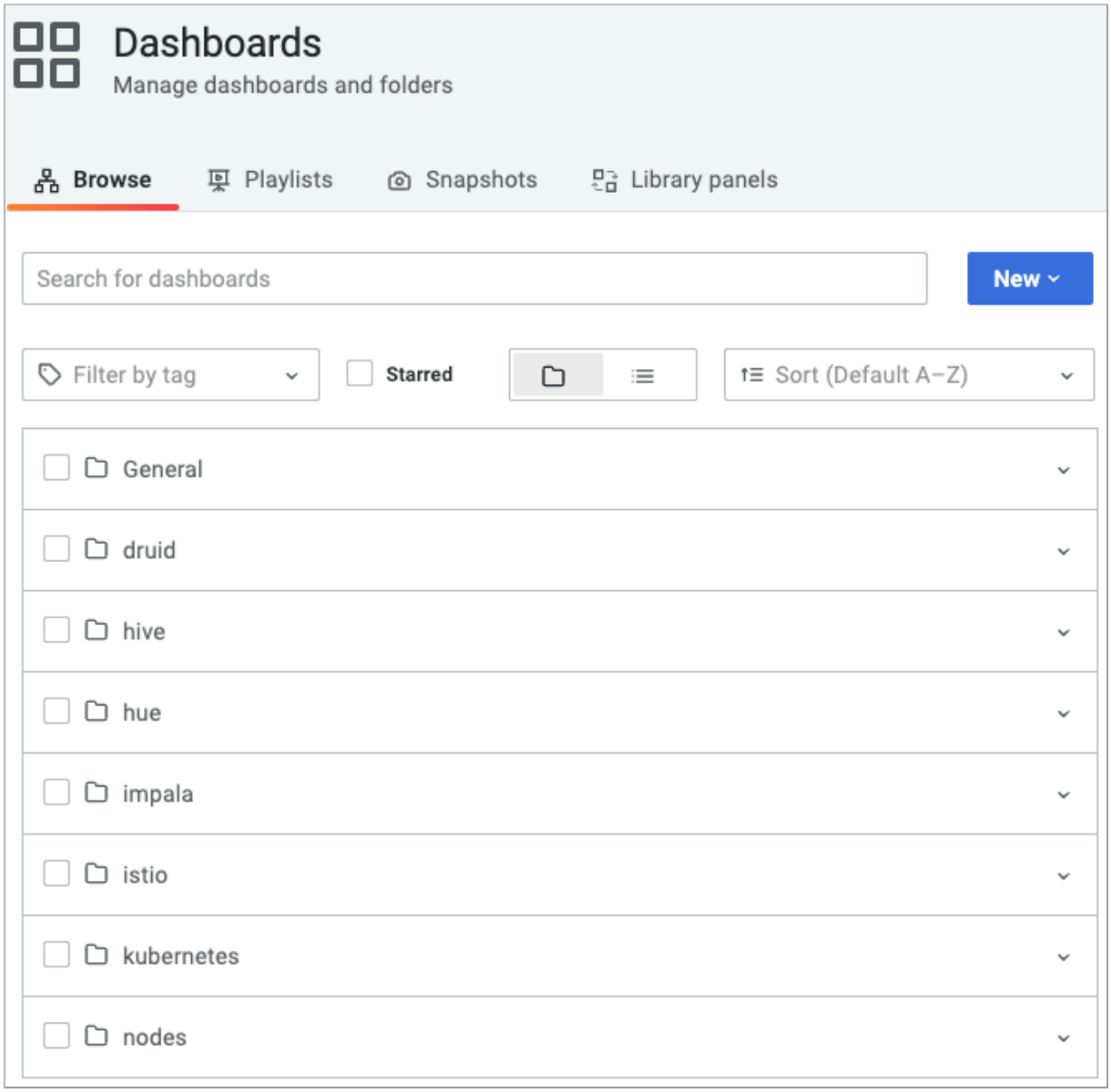

3. Click the name of a dashboard group.
   For example, click hive.
   Names of the Hive dashboards in the group appear, a few of which are shown below:

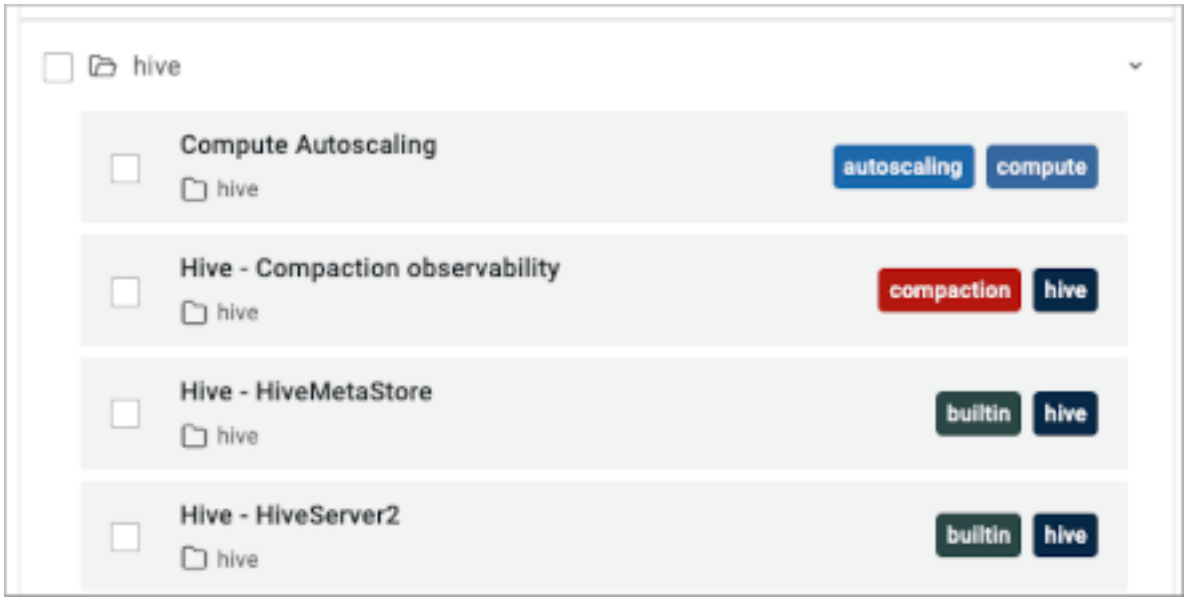

4. Click the name of a dashboard group.
   For example, click Compute Autoscaling.
   A number of dashboards appear. Only two are shown below:

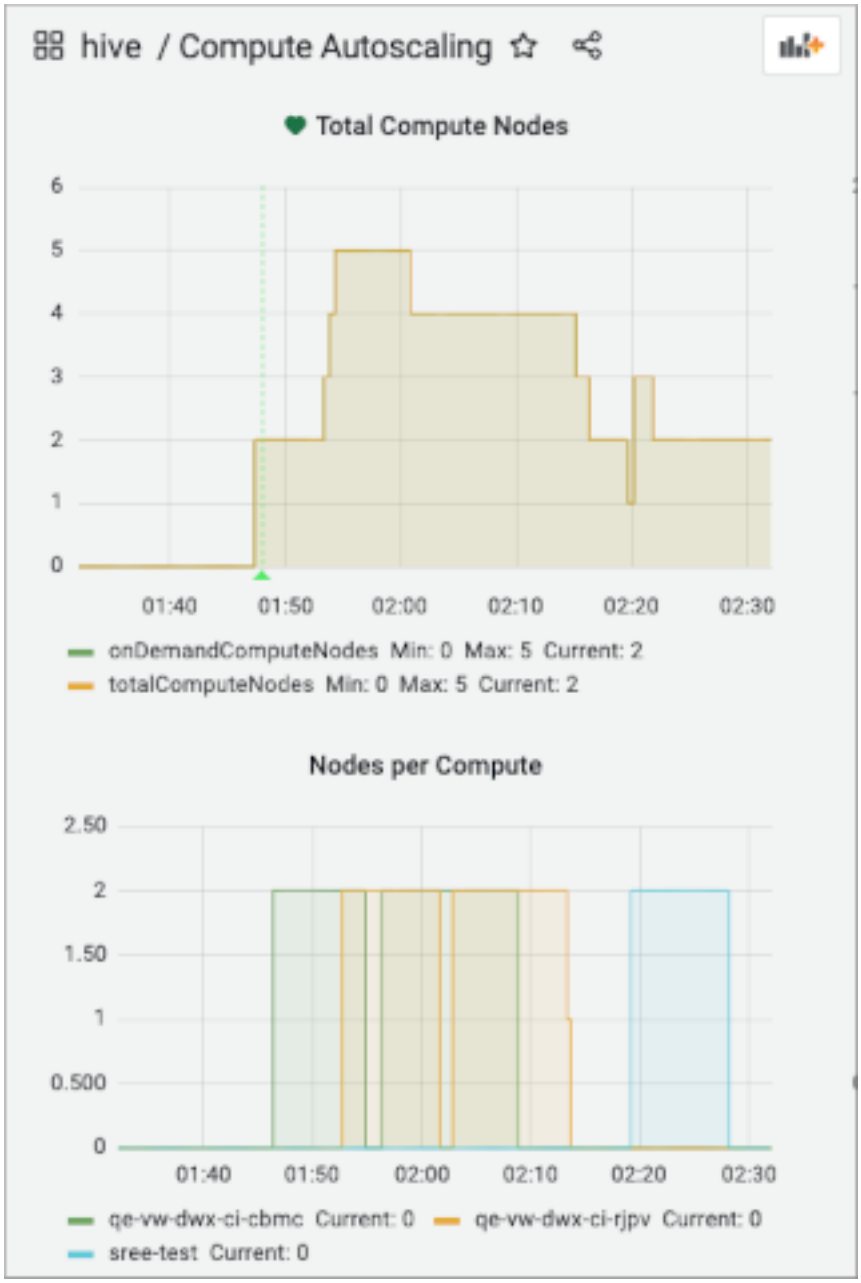

5. Select viewing of metrics over the last 24 hours.

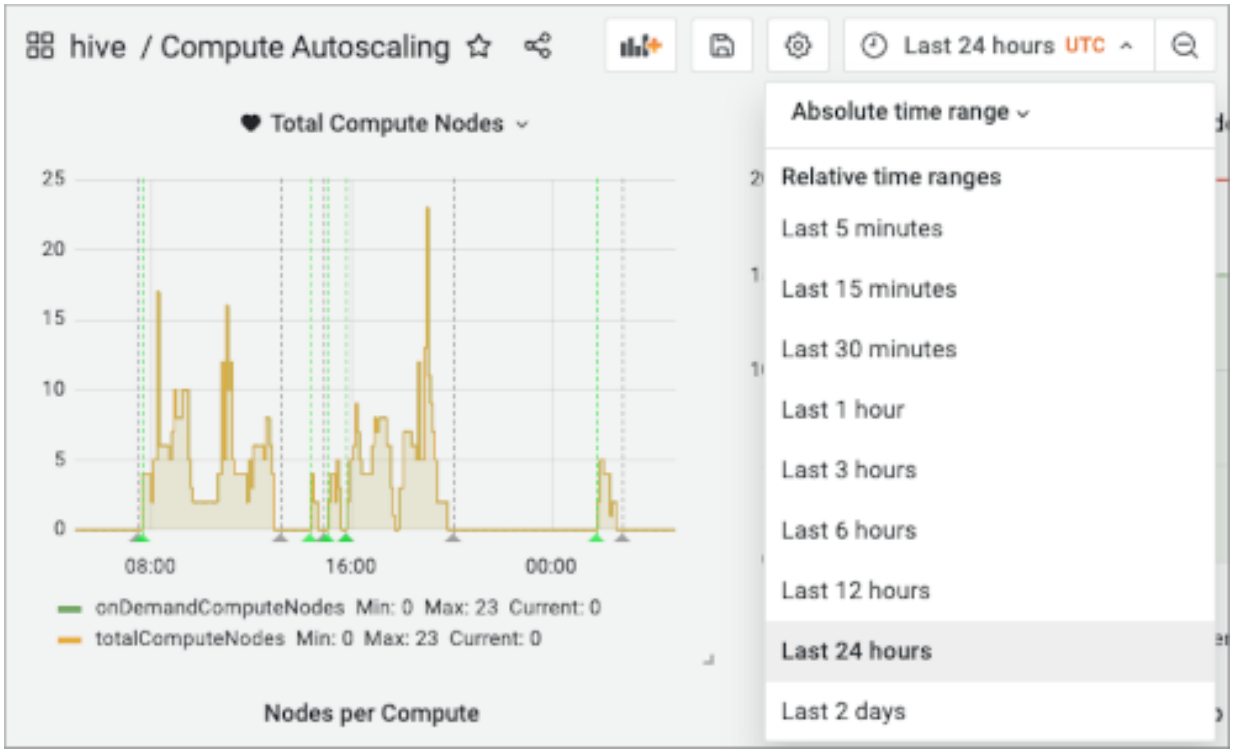

6. Hover over a graphic to reveal metric data for a particular point on the X-axis.

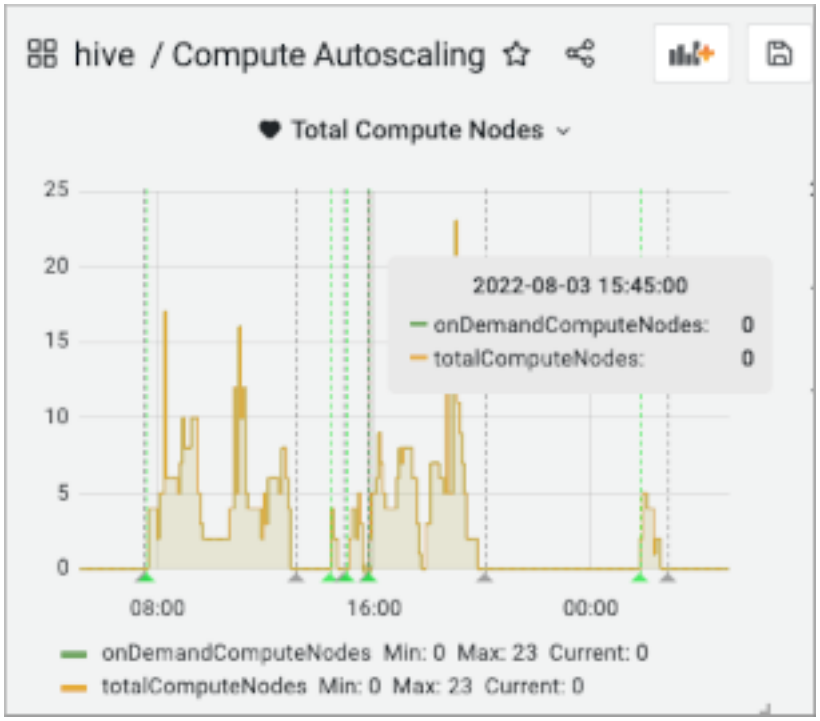

Related Information

Monitoring HMS

You can monitor Hive Metastore (HMS), heap usage, and key Hive metrics.

Procedure

1. In the Welcome screen, click grid , and then select Manage.
   A list of dashboard groups appears:

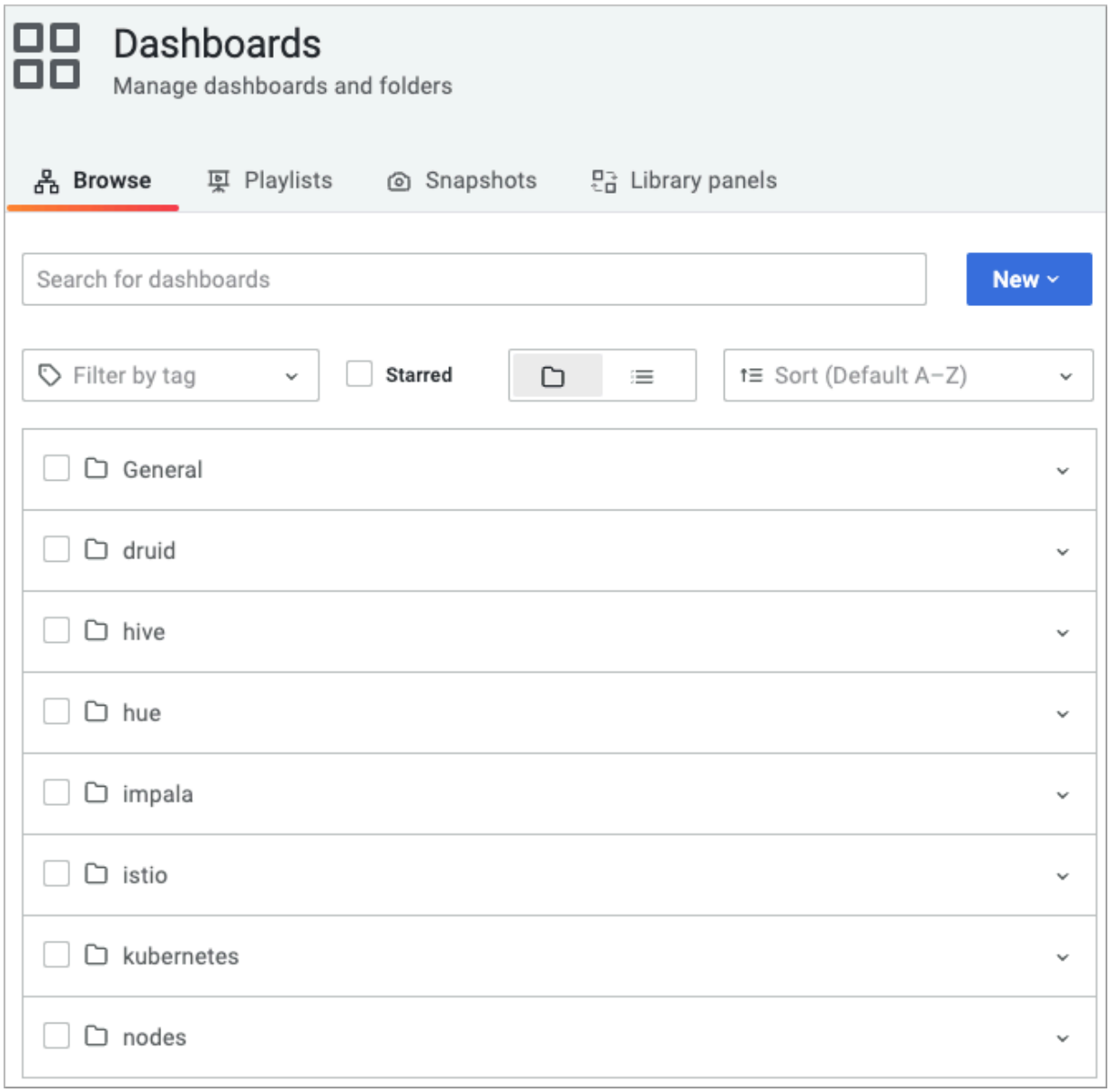

2. Select the hive dashboard.
   Names of the Hive dashboards in the group appears, a few of which are shown below:

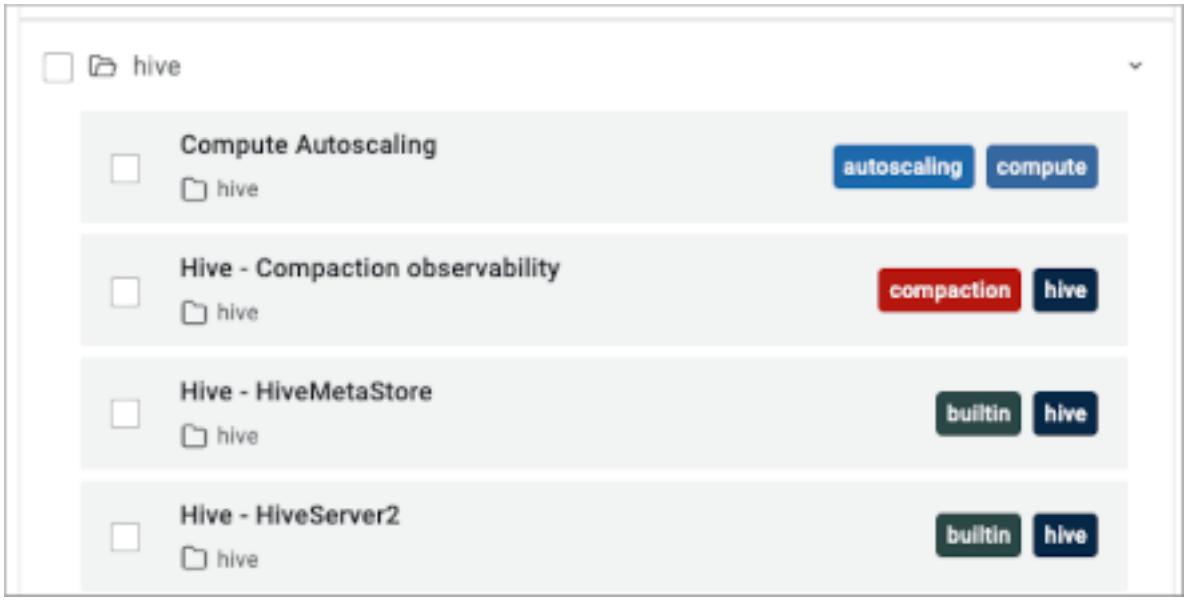

3. Select Hive - Compaction observability, and in Namesparce, select a namespace.
   In Alerts, the error indicator shows the number of errors and warnings.

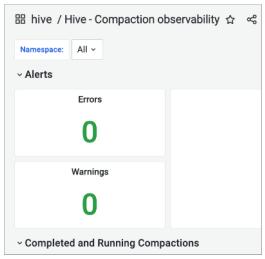

4. Expand Completed and Running Compactions.
   Graphs showing the status of compactions appears.

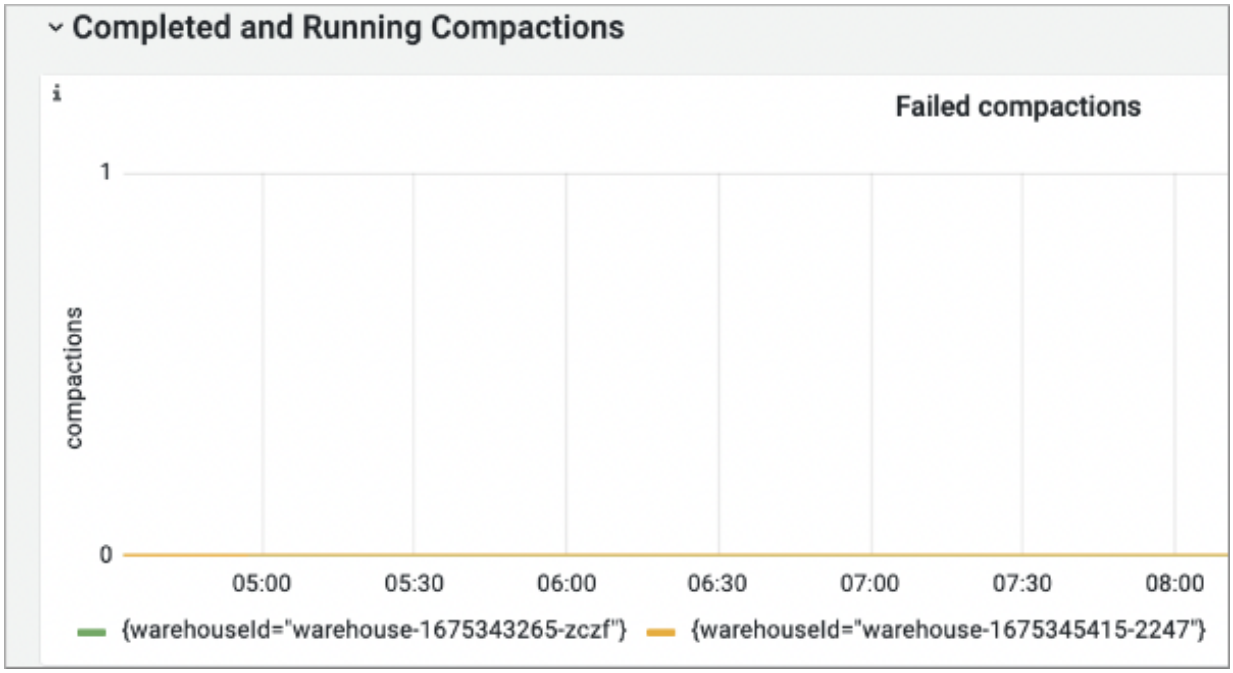

5. Go back to the list of Hive dashboards, and select Hive - HiveMetaStore.

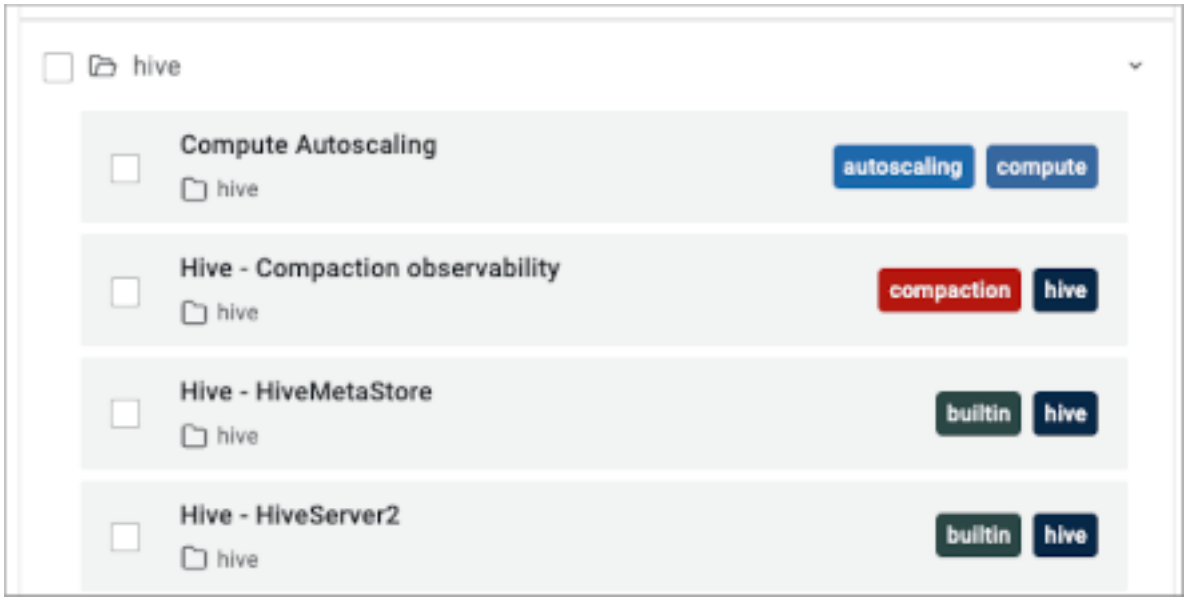

6. In Database Catalog, and then in Pod Name, accept the default All, or select a particular Database Catalog and pod.

In Memory Pressure, metrics about the overall status of HMS appear.

7. Click each row title to expand and look at various metrics: app stop times due to garbage collection and heap usage.

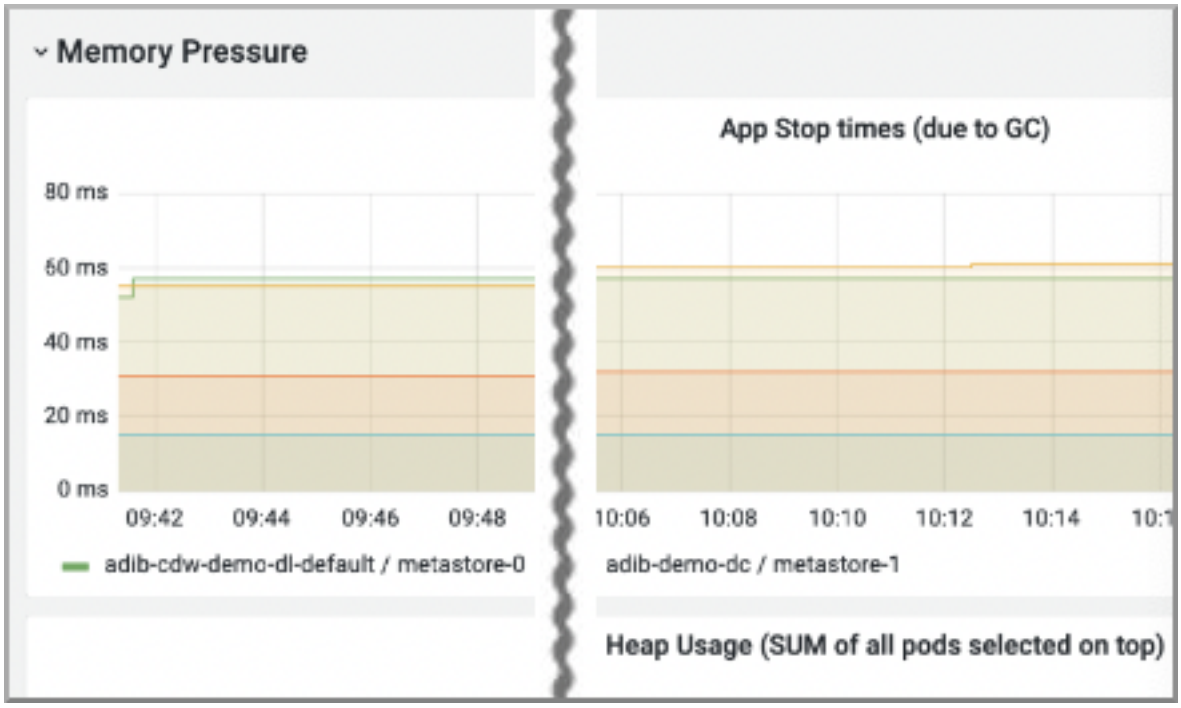

Monitoring key Hive metrics

You can get metrics about the status of HiveServer.

Procedure

1. Go back to the list of Hive dashboards, and select Hive - HiveServer2.

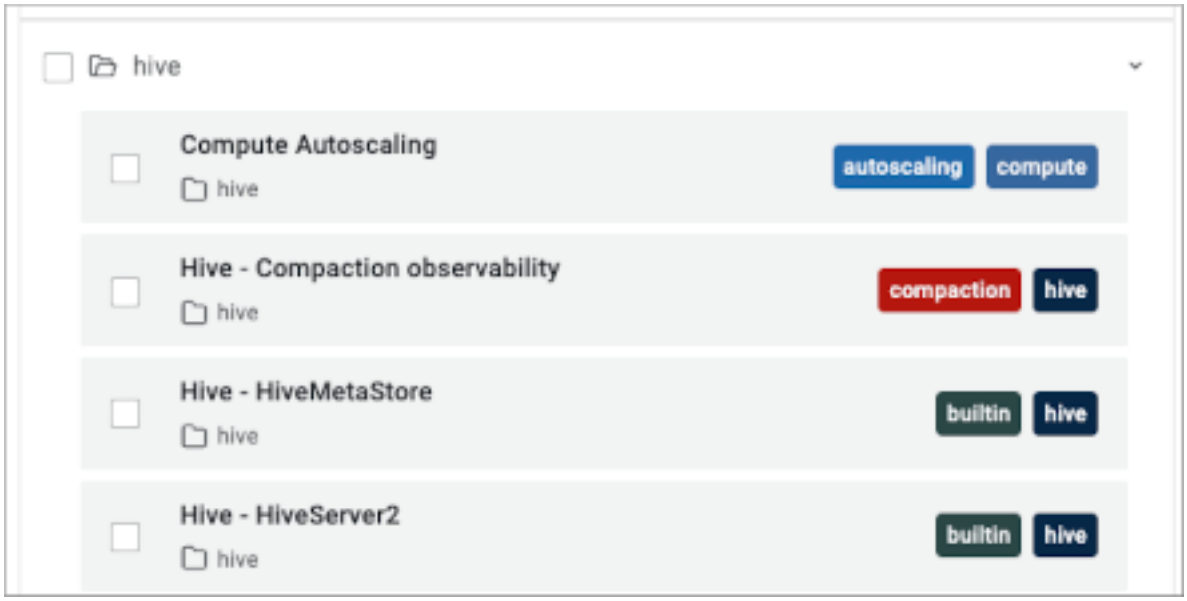

2. In Virtual Warehouse, and then in Pod Name, accept the default All, or select a particular Virtual Warehouse and pod.

In Memory Pressure, metrics about the overall status of HiveServer memory usage appear.

3. Click each row title to expand and look at various metrics: App stop times due to garbage collection, heap usage, active thread operation count, and completed thread operation states.

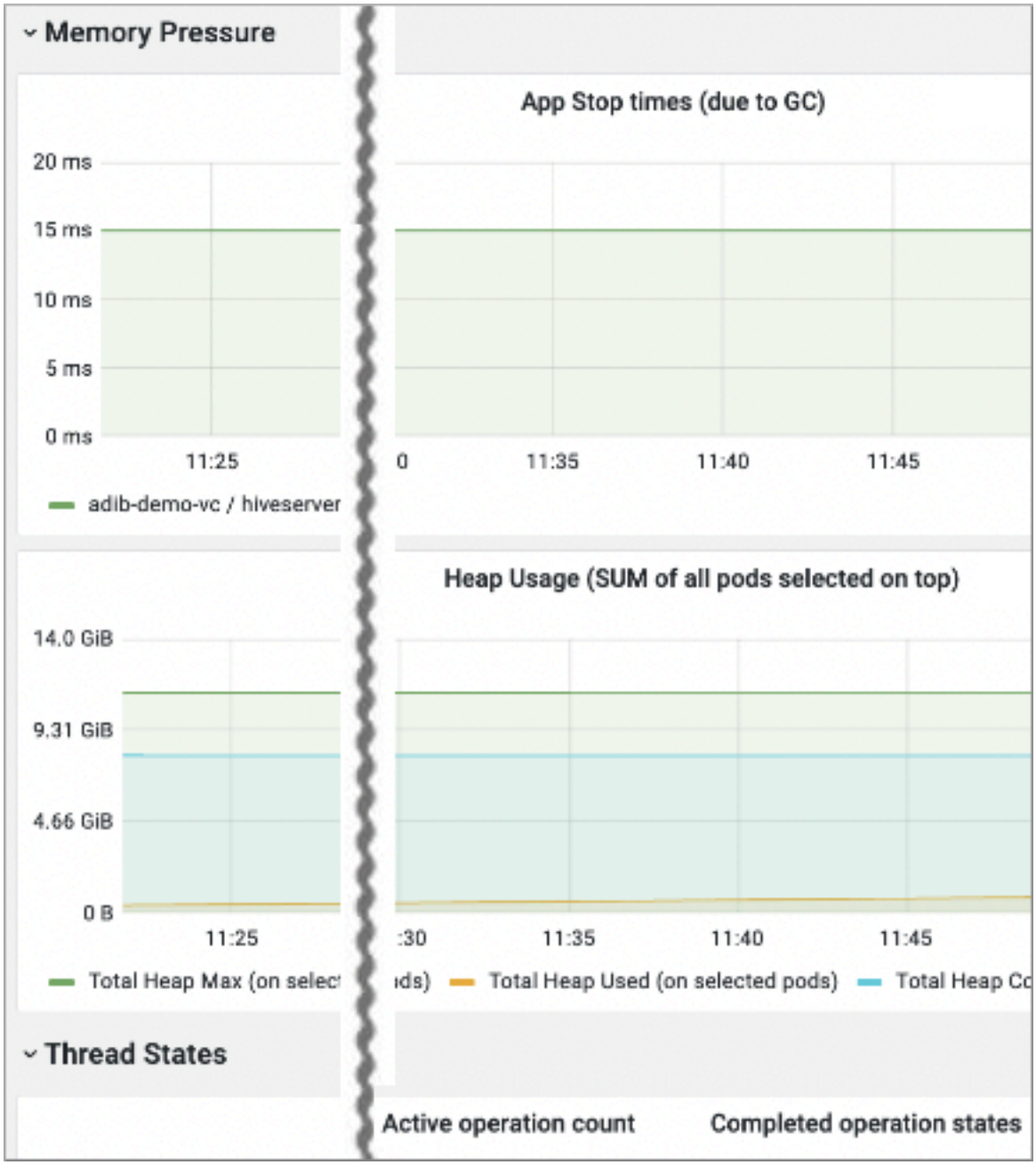

4. In hive / Hive - Home, you can monitor warehouse growth from metics on the tables and partitions created hourly.

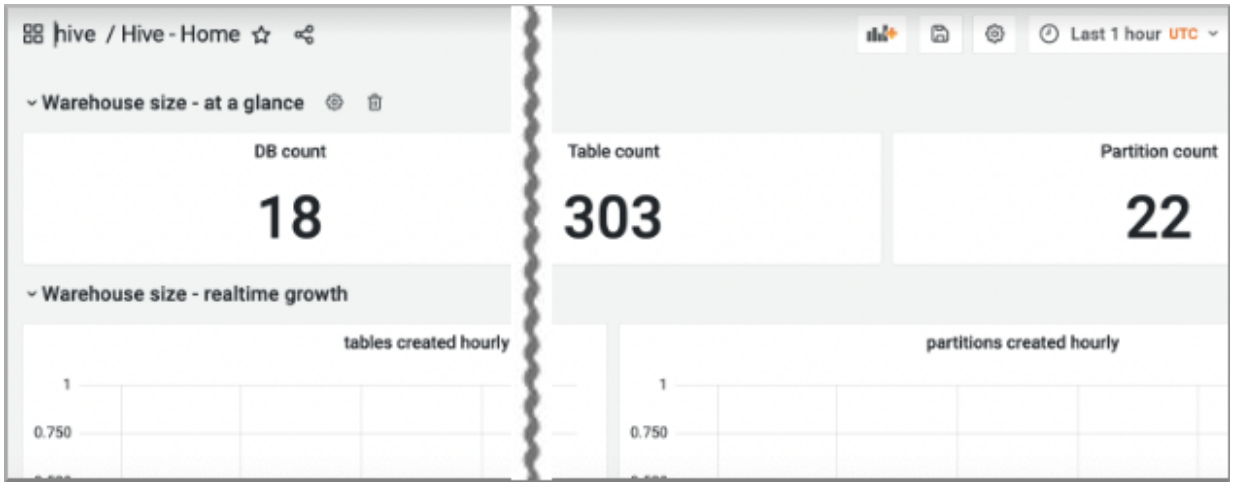

Monitoring Impala executors

You can monitor Impala executors and get related metrics.

Procedure

1. In the Welcome screen, click grid , and then select Manage.
   A list of dashboard groups appears:

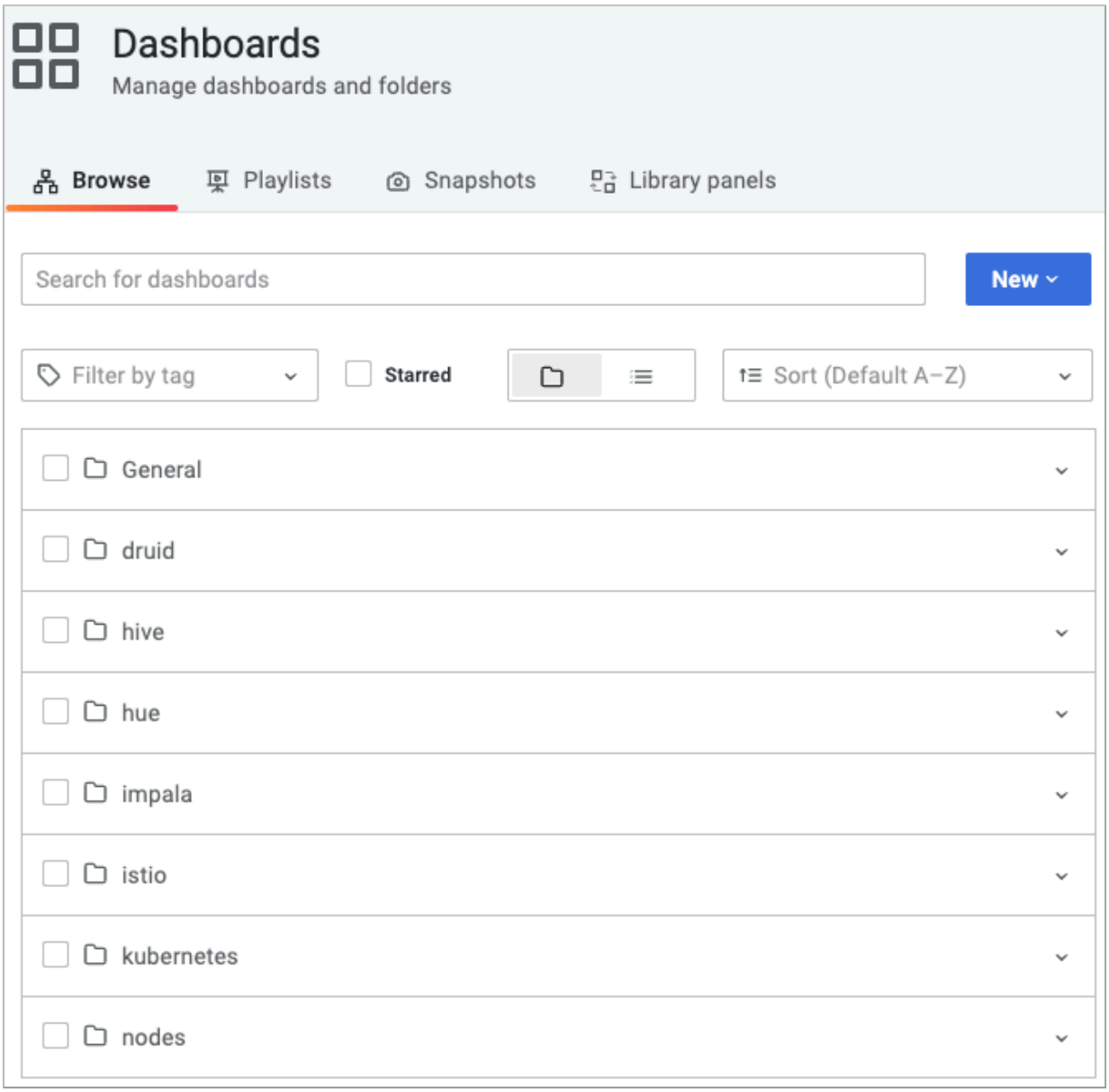

2. Click the impala dashboard group.

3. In the list of Impala dashboards, and select Impala - Executor.
   The number of Executor Groups and Executors appear.

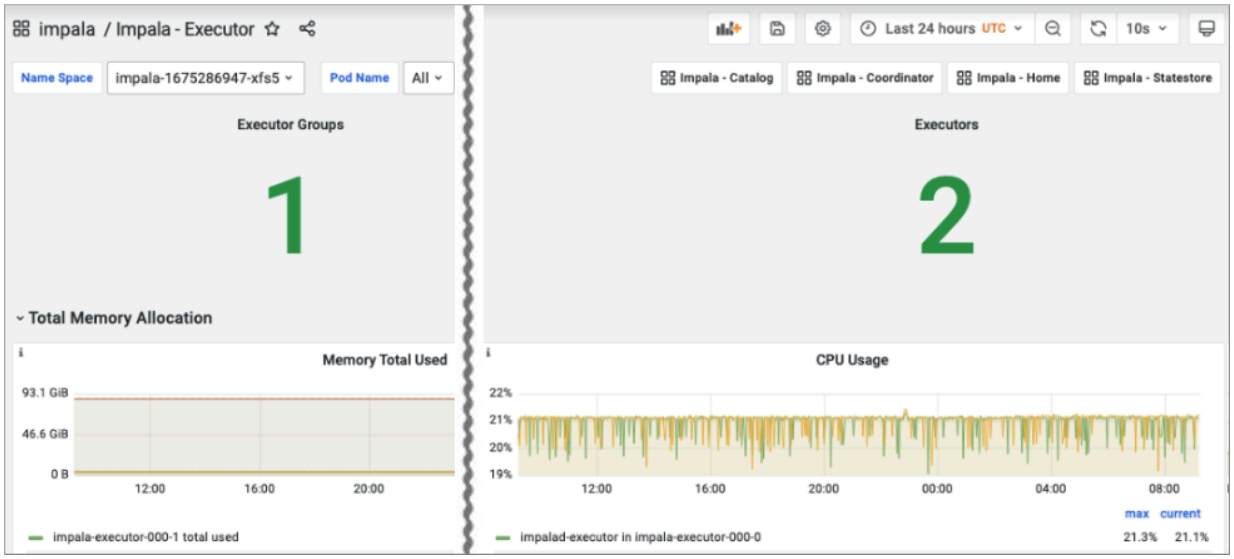

4. View graphs and metrics of memory usage.

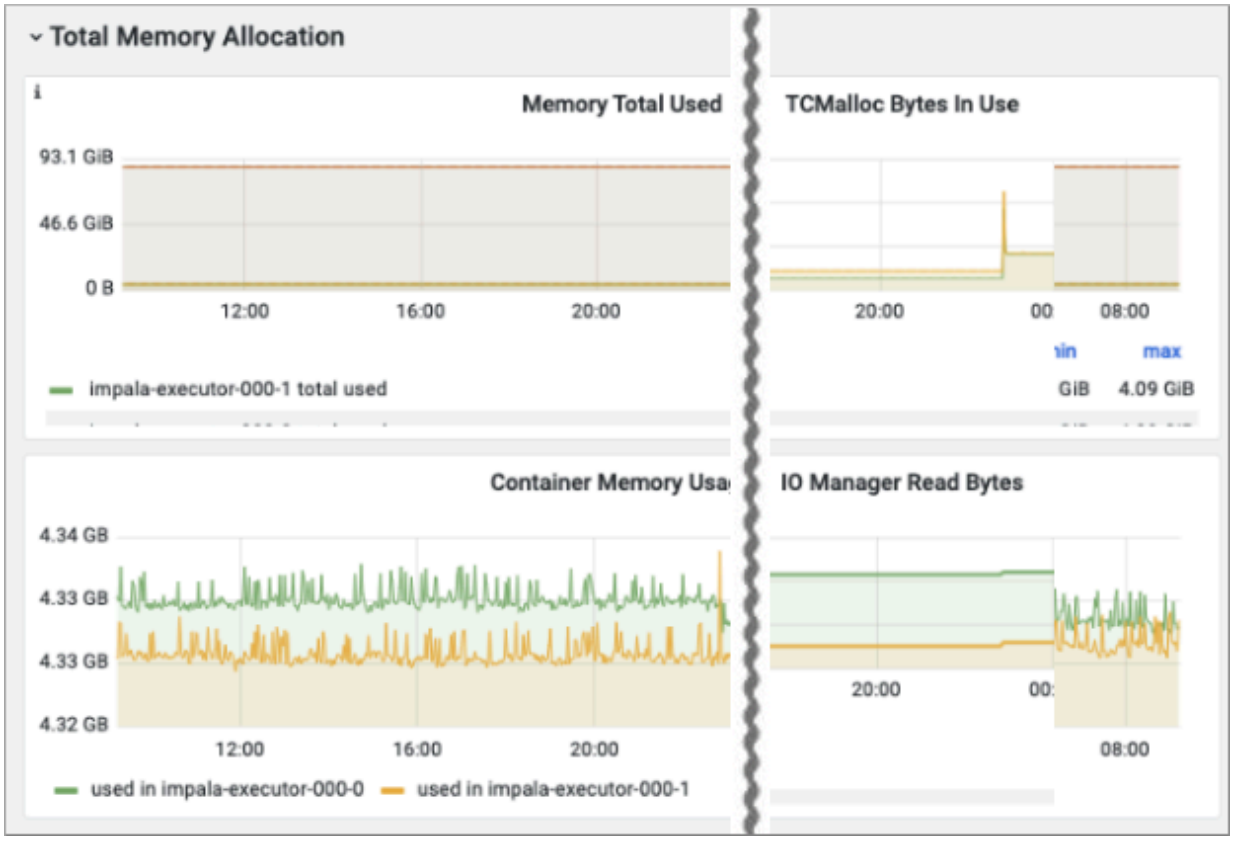

5. Get information about Executor node CPU usage.

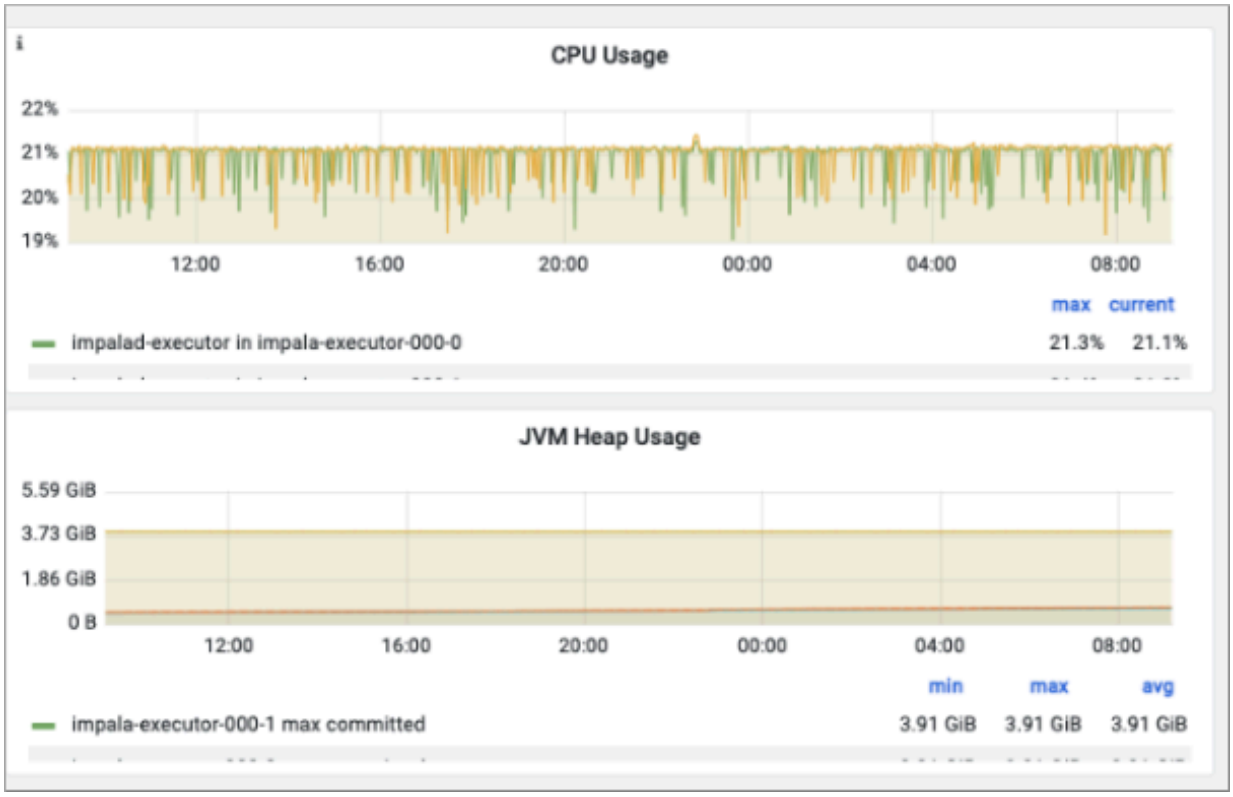

Monitoring Impala catalog

You can monitor Impala catalogd from Grafana.

Procedure

1. In the list of Impala dashboards, and select Impala - Catalog.
   The number of databases and tables appear.
2. Select a Name Space, and from the Last 24 hours dropdown, select an time interval to monitor.

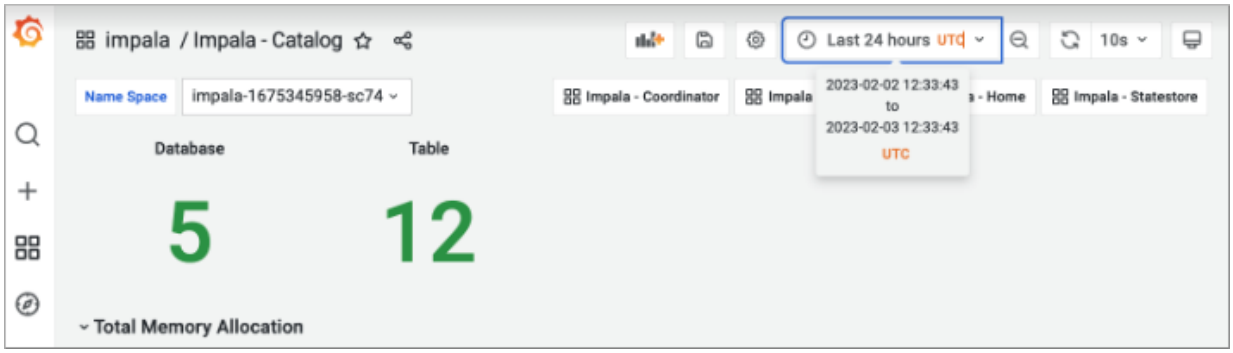

3. View catalog memory usage, container memory usage, cpu usage, JVM heap usage, and cumulative JVM garbage collection time.

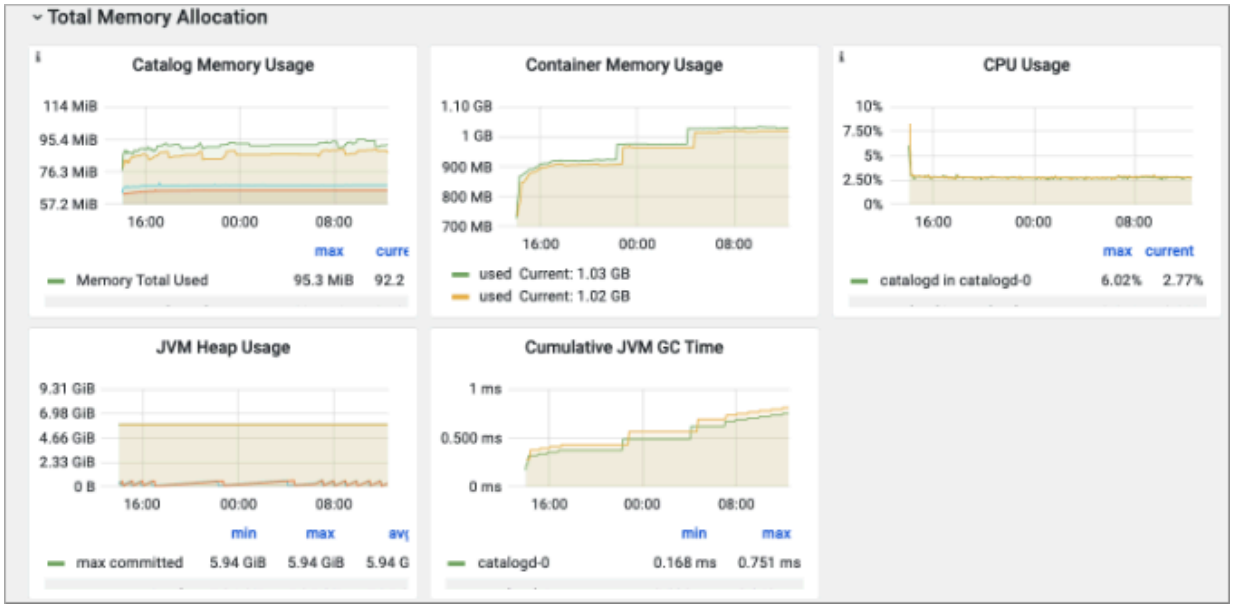

4. Scroll down, and view remote procedure call metrics.

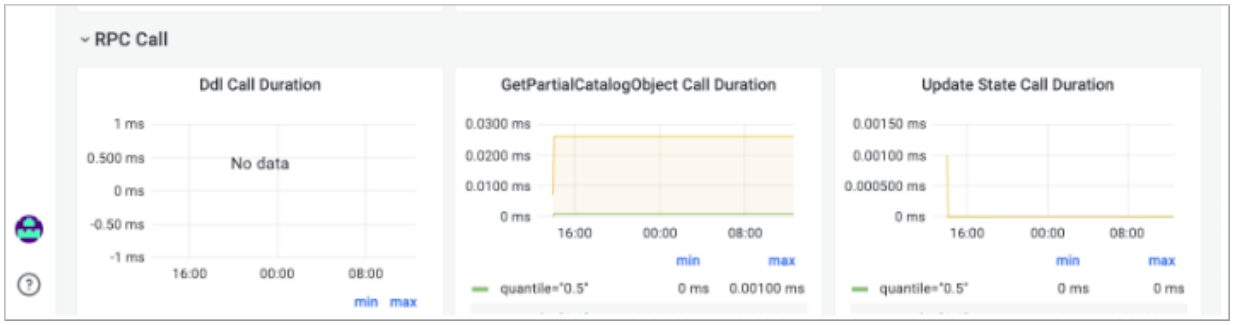

5. Scroll down, and view Impala statestore metrics.

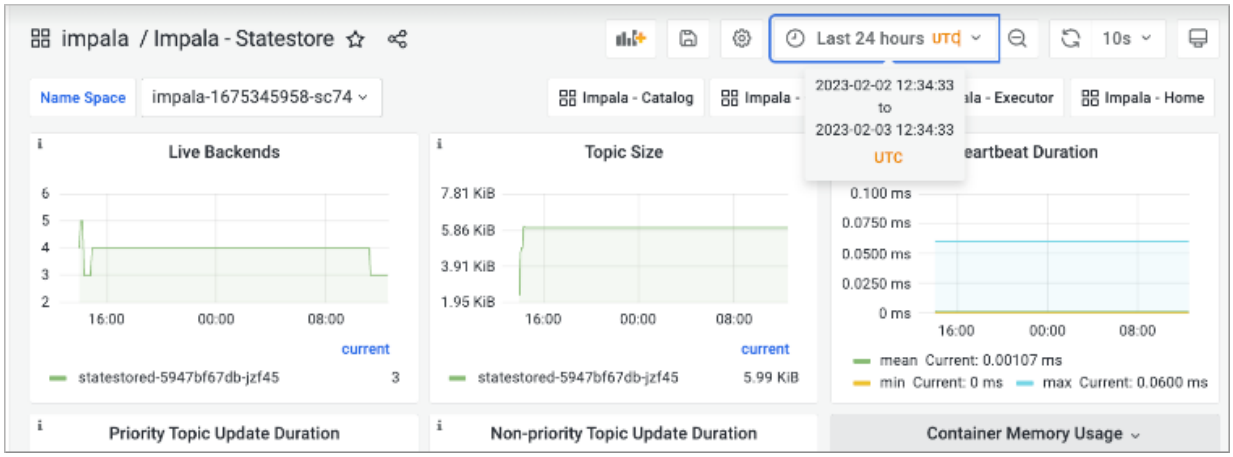

6. Scroll down, and view update, memory, and CPU metrics.

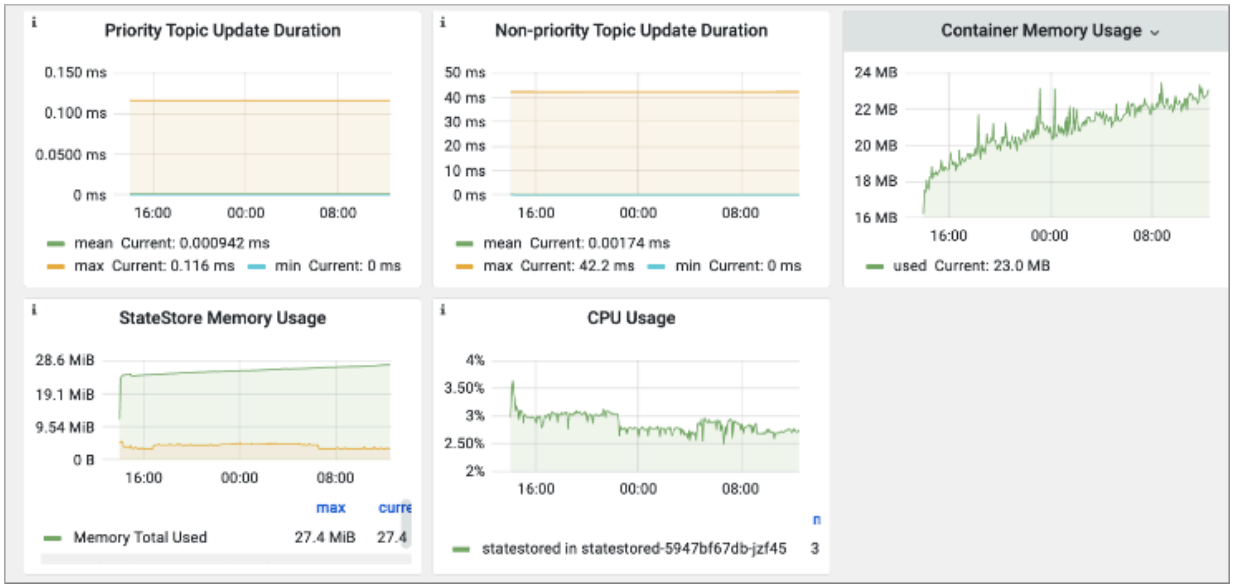

Monitoring Impala statestore

You can monitor Impala statestore from Grafana.

Procedure

1. In the list of Impala dashboards, and select Impala - Statestore.
2. Scroll down, and view live backends, topic size, and heartbeat duration.

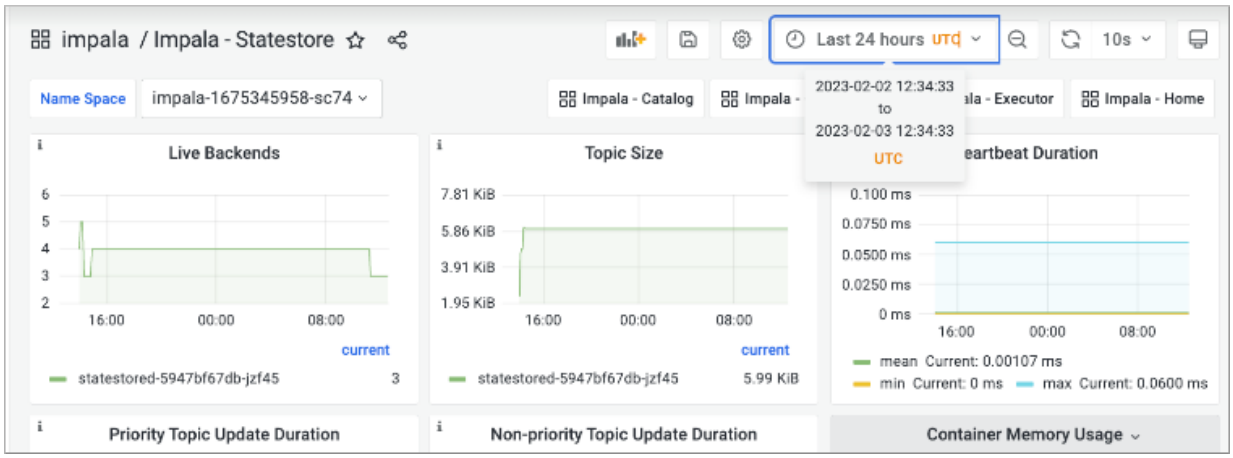

3. Scroll down, and view priority and non-priority topic update duration, container memory usage, statestore memory, and cpu usage.

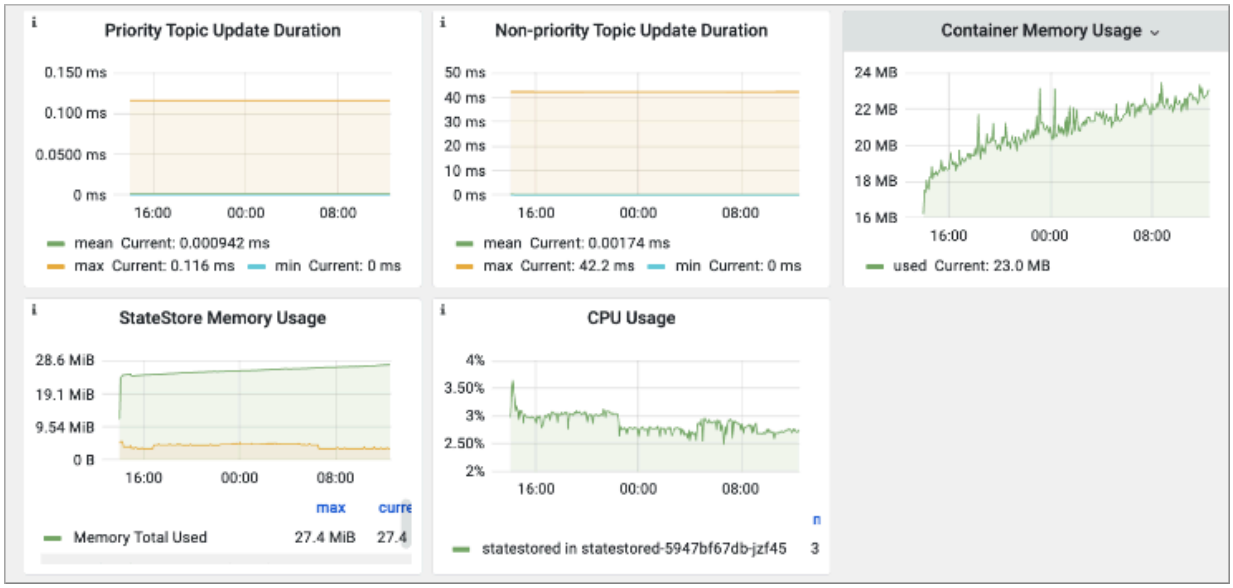

Monitoring Impala admission control

You can monitor Impala admission control from Grafana.

Procedure

1. In the list of Impala dashboards, and select Impala - Home.
   In Impala Overview, the number of of Impala databases, tables, the catalogd status, and other metrics appear.

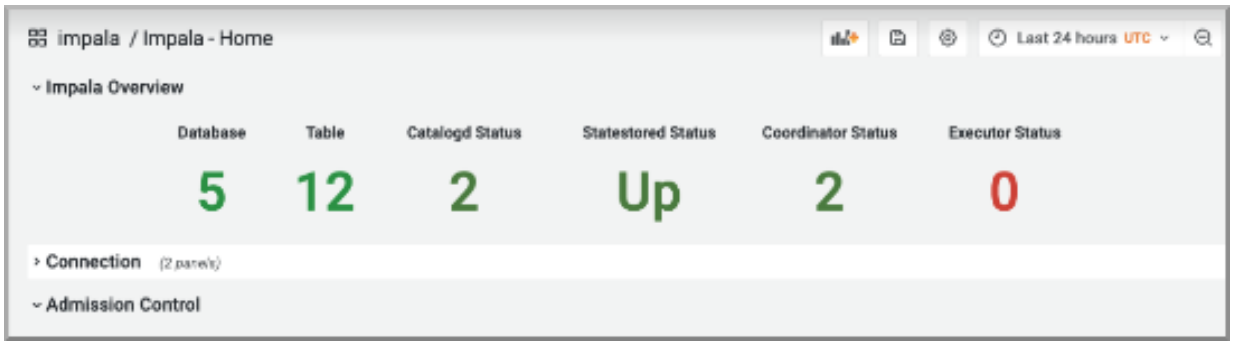

2. In Admission Control, view the Impala local memory in root default, the number of queries admitted that are running in root default, and the queries in root default (not shown).

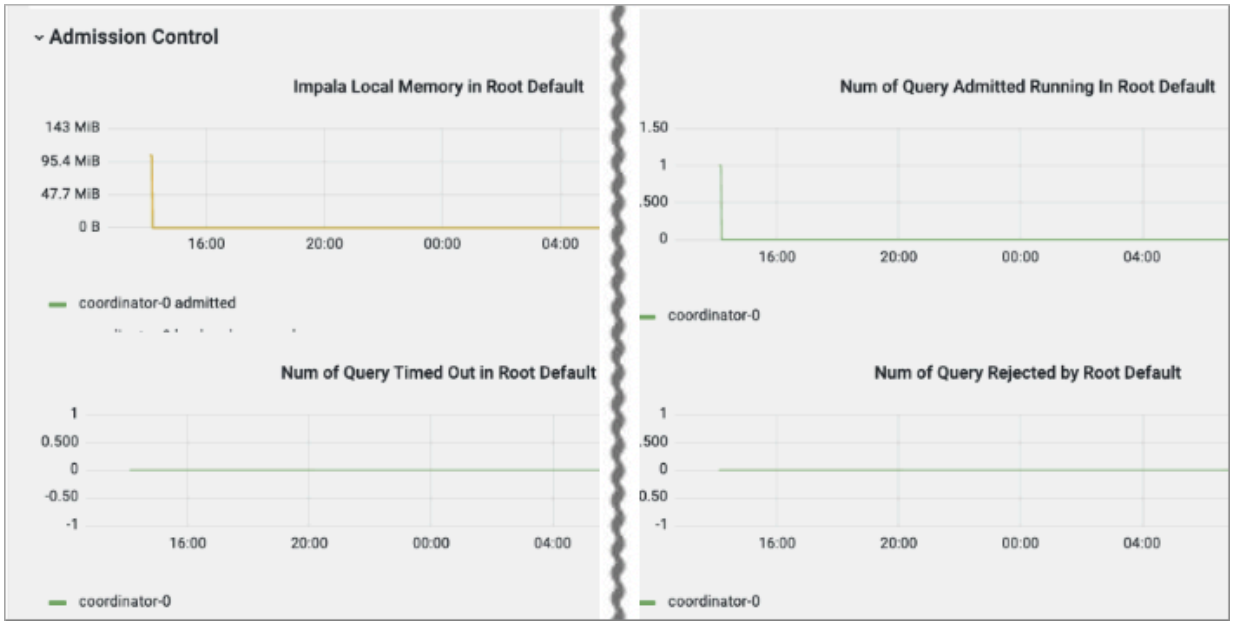

3. View the number of queries timed out and rejected in root default.
4. Scroll down and in Query view the number of registered queries, fragments in flight, number of open files (not shown), data manipulation language (DML) time distribution, and data definition language (DDL) time distribution.

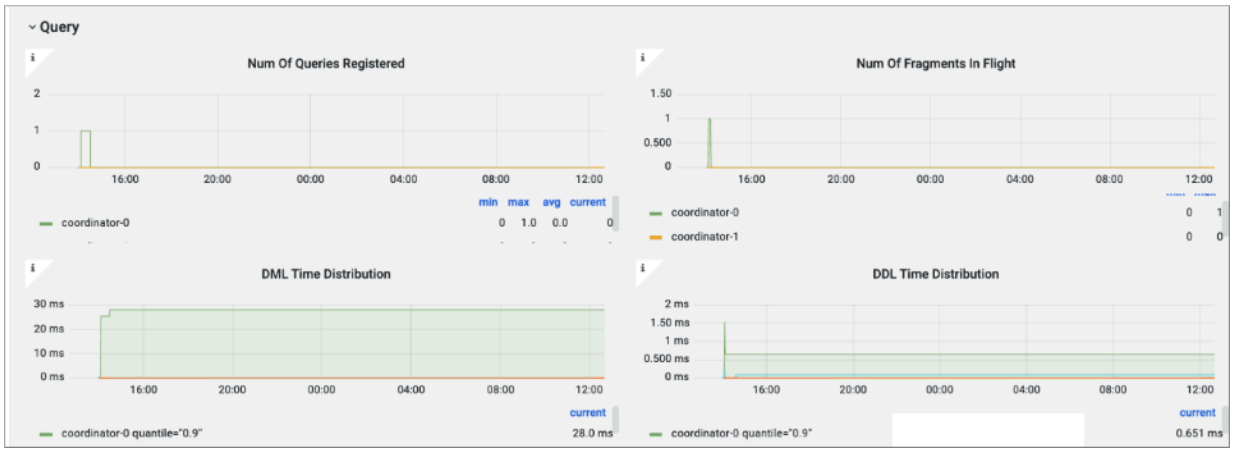

Monitoring Impala coordinators

You can monitor Impala coordinators caching, resource use, such as files handles hit, memory consumption, and more.

Procedure

1. In the list of Impala dashboards, click Impala - Coordinator.

2. Select a name space and pods, and view cache hit status, remote data cache miss bytes, catalog cache hit rate (not shown), file handles hit count, and file handles miss count.

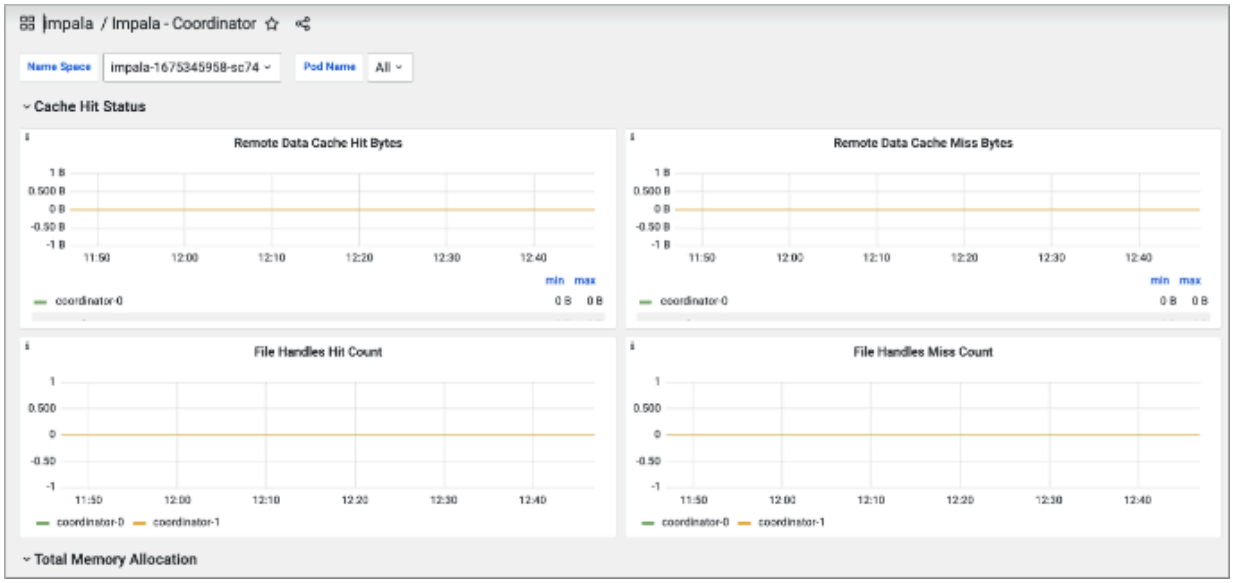

3. Scroll down to Total Memory Allocation and view memory used, TCMalloc bytes in use, total buffer memory allocated, JVM max heap, JVM heap current usage, and CPU usage.

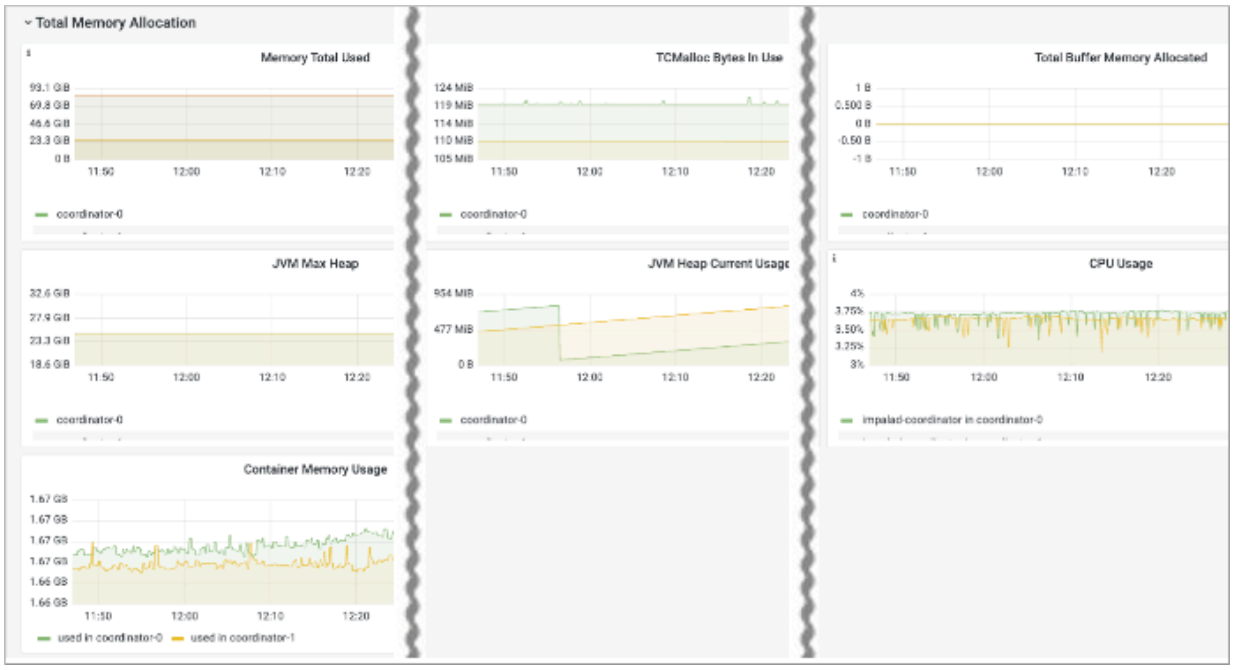

Monitoring nodes

You can get Kubernetes, cluster health, and node health information by monitoring Cloudera Data Warehouse nodes.

Procedure

1. In the Welcome screen, click grid , and then select Manage.
   A list of dashboard groups appears:

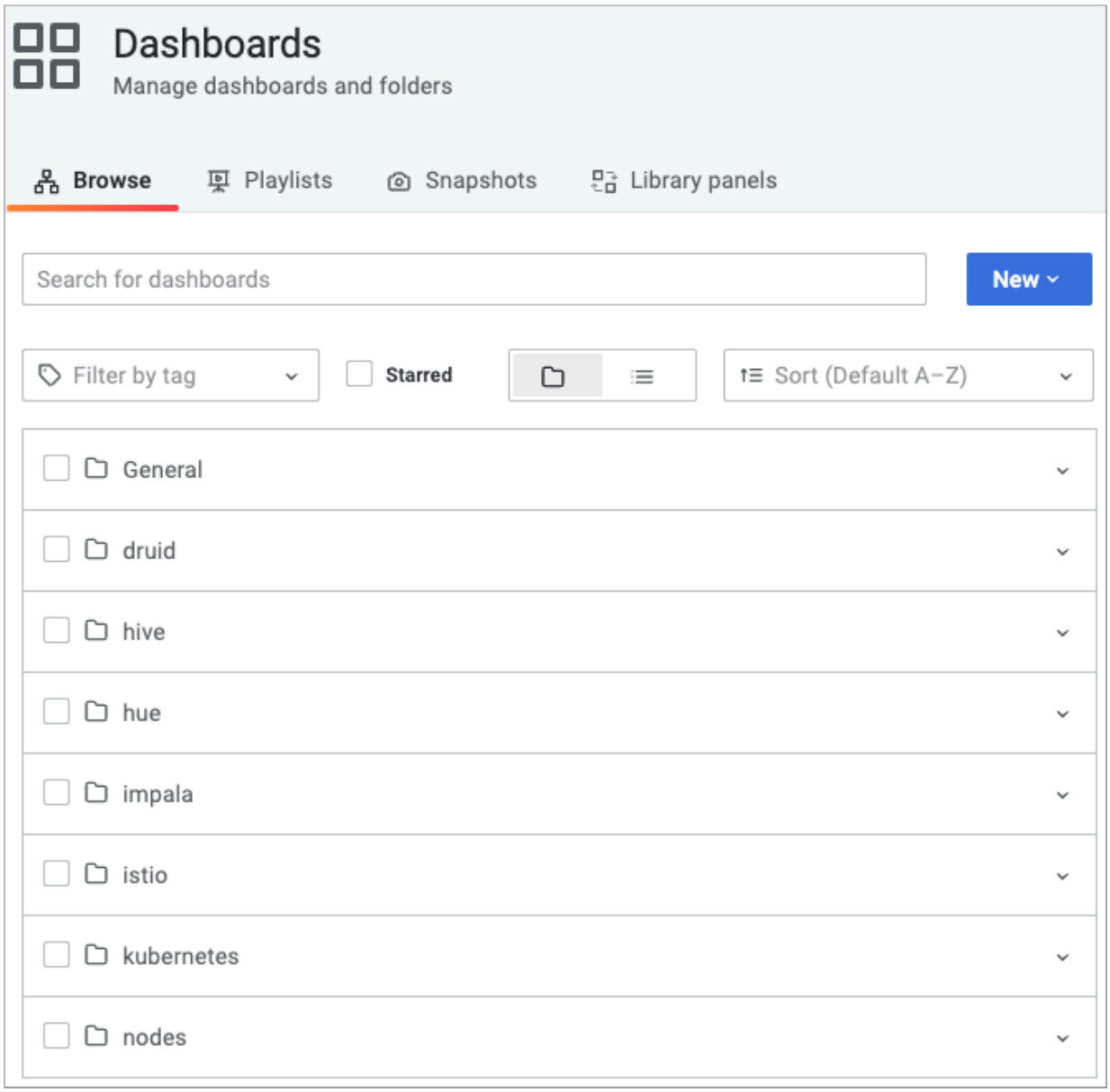

2. Click the nodes dashboard group.
   Names of the nodes dashboards in the group appear:

3. In the list of nodes dashboards, click Node Details, and then set the node of interest: In datasource, set default for example; in Job, set kubernetes-service-endpoints for example; in Host, set the IP address of a host in the Cloudera Data Warehouse cluster.

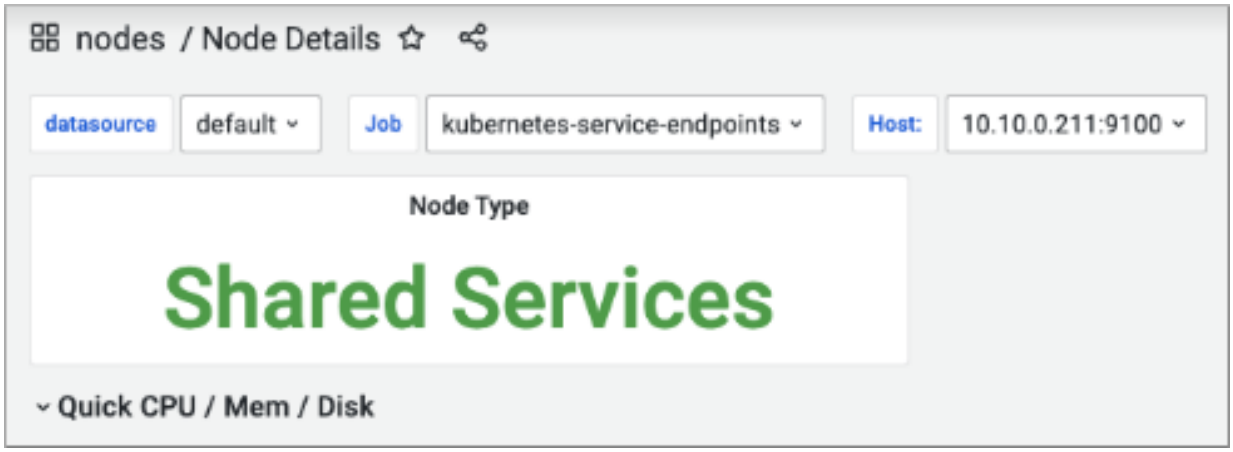

4. Expand Quick CPU/Mem/Disk, and view the percentage of busy CPU, the percentage of average load over the past 5 minutes and over the last 15 minutes, and RAM used.
5. On the right side of the dashboard, with Last 24 hours selected, for example, view the SWAP space used, root file system used, CPU cores used, and uptime of the node.

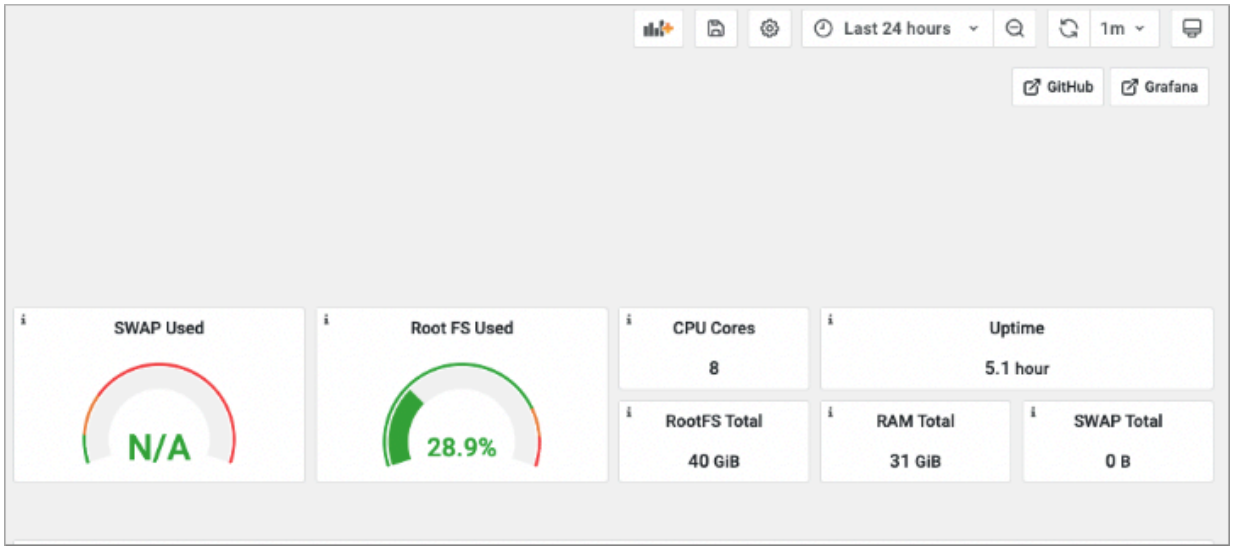

6. Scroll down, expand Basic CPU/Mem/Net/Disk, and see graphs on the left showing metrics about high activity from various souces, such as a busy user or busy interrupt requests (IRQ), and a graph of packets received and transmitted by network traffic.

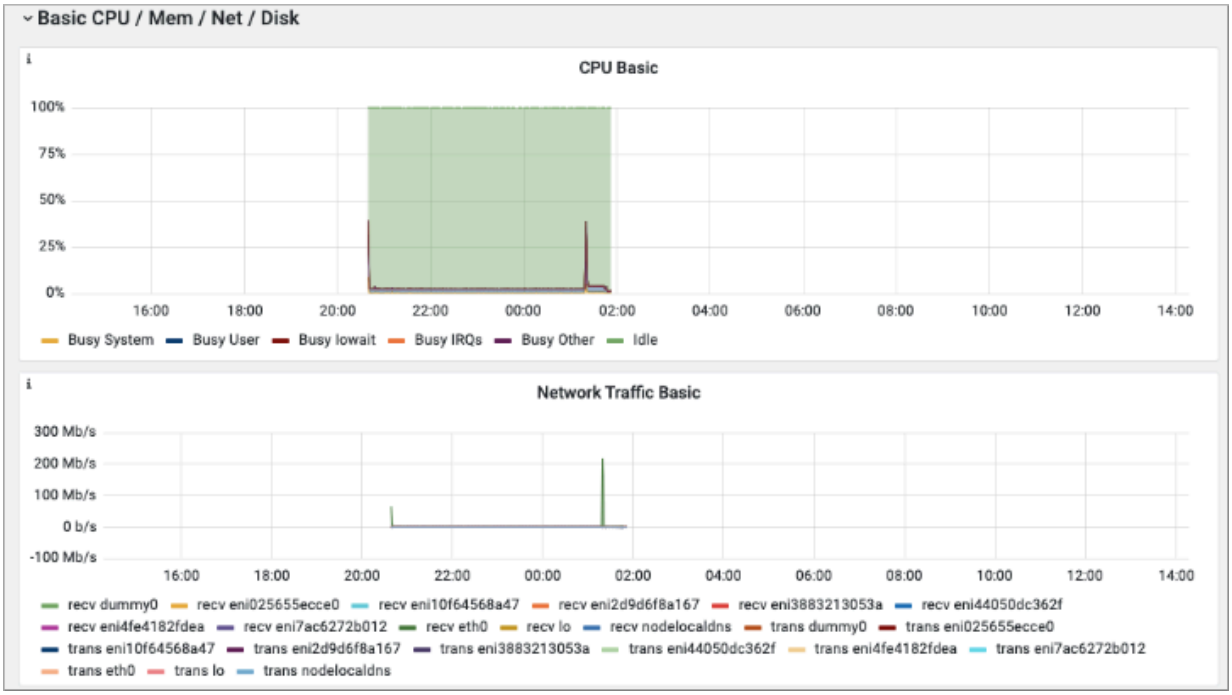

7. On, the right, see graphs of basic memory and disk space use.

8. In the list of nodes dashboards, click Cluster Totals to view cluster health in graphs and metrics over the past 24 hours, as indicated by Cluster Memory Utilization, Cluster CPU Utilization, Shared Services Node Count, and Compute Nodes Count.

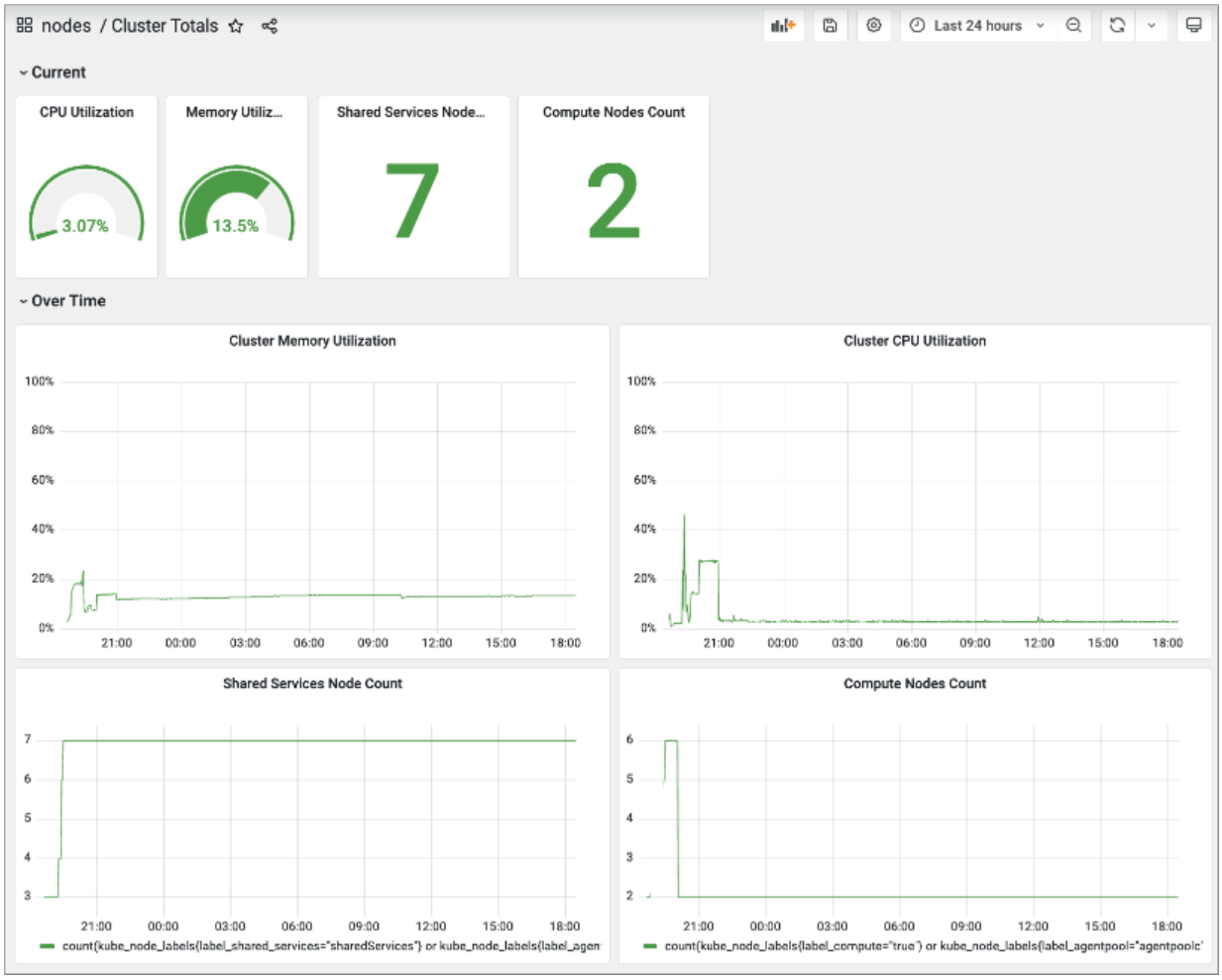

9. In the list of nodes dashboards, click Node Trends to view node health over the past 24 hours, as indicated by CPU Utilization, Load Average and CPU Count, Memory Utilization, and Disk Utilization.

Monitoring Kubernetes Services from Grafana

You can monitor the Amazon Elastic Kubernetes Service (EKS) or the Azure Kubernetes Service (AKS) used by your Cloudera Data Warehouse cluster from Grafana.

About this task

You can also monitor the Amazon Elastic Kubernetes Service (EKS) from the Cloudera K8s dashboard.

Procedure

1. In the Welcome screen, click grid , and then select Manage.
   A list of dashboard groups appears:

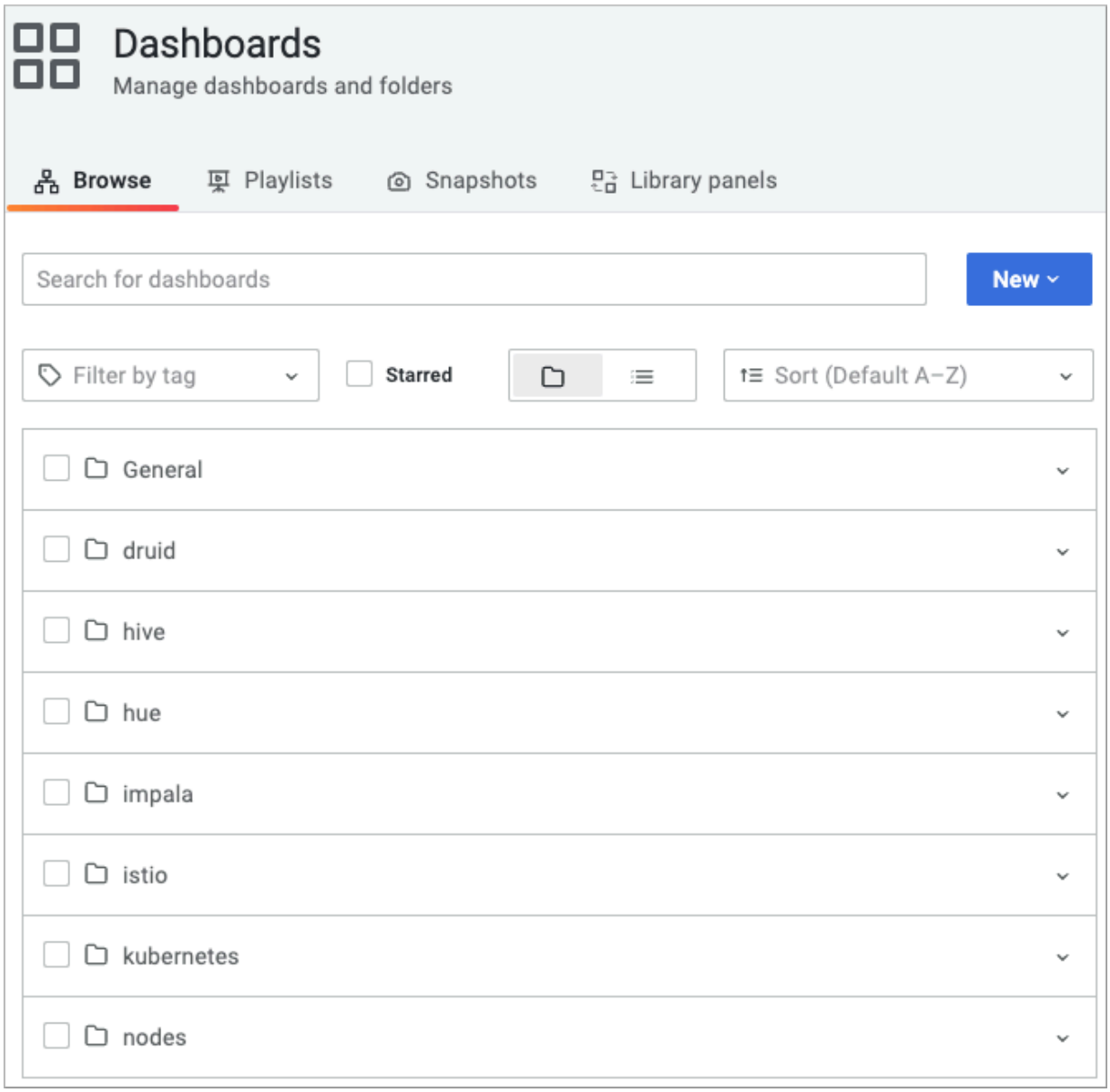

2. Click the kubernetes dashboard group.
   Names of the kubernetes dashboards in the group appear:

3. Click the Kubernetes App Metrics, and then select a namespace, for example cluster, and container, for example a container named cdp-coredns-updater.
   View a graph and metrics of the number of pods.

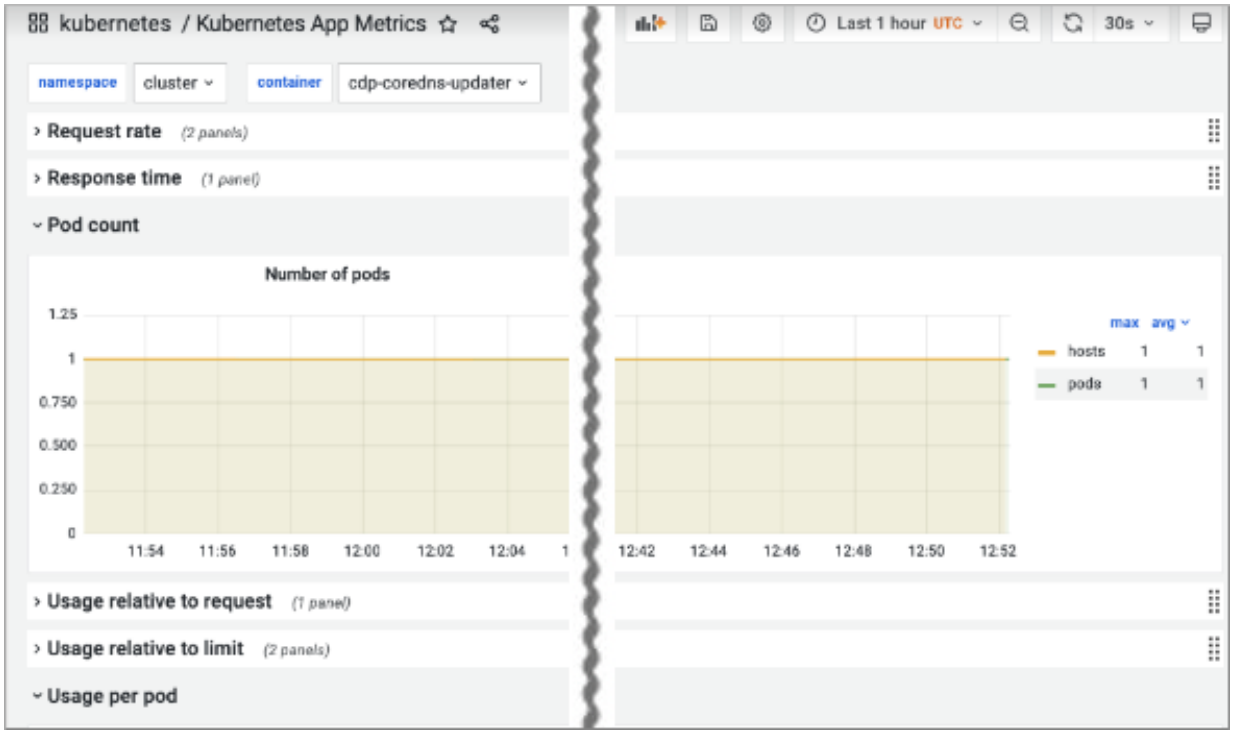

4. View the CPU usage per pod.

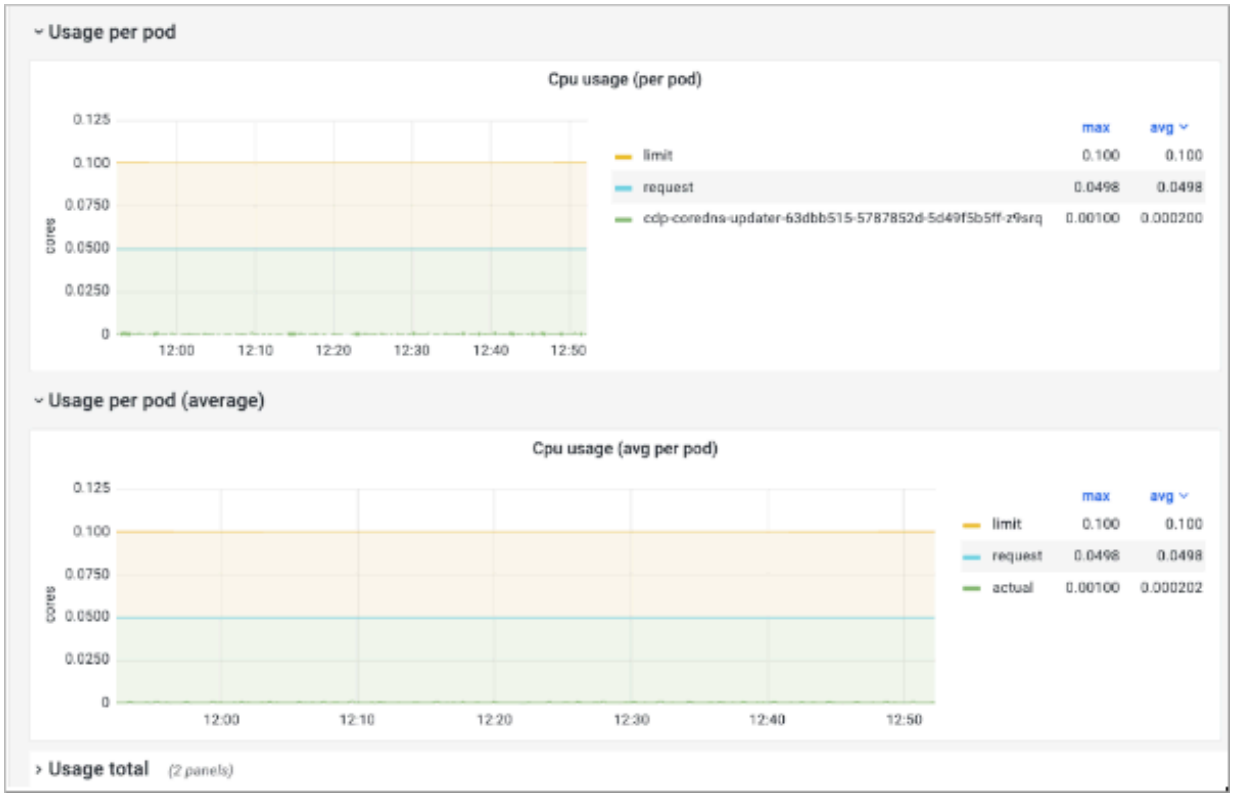

5. On the right, view the memory usage per pod.

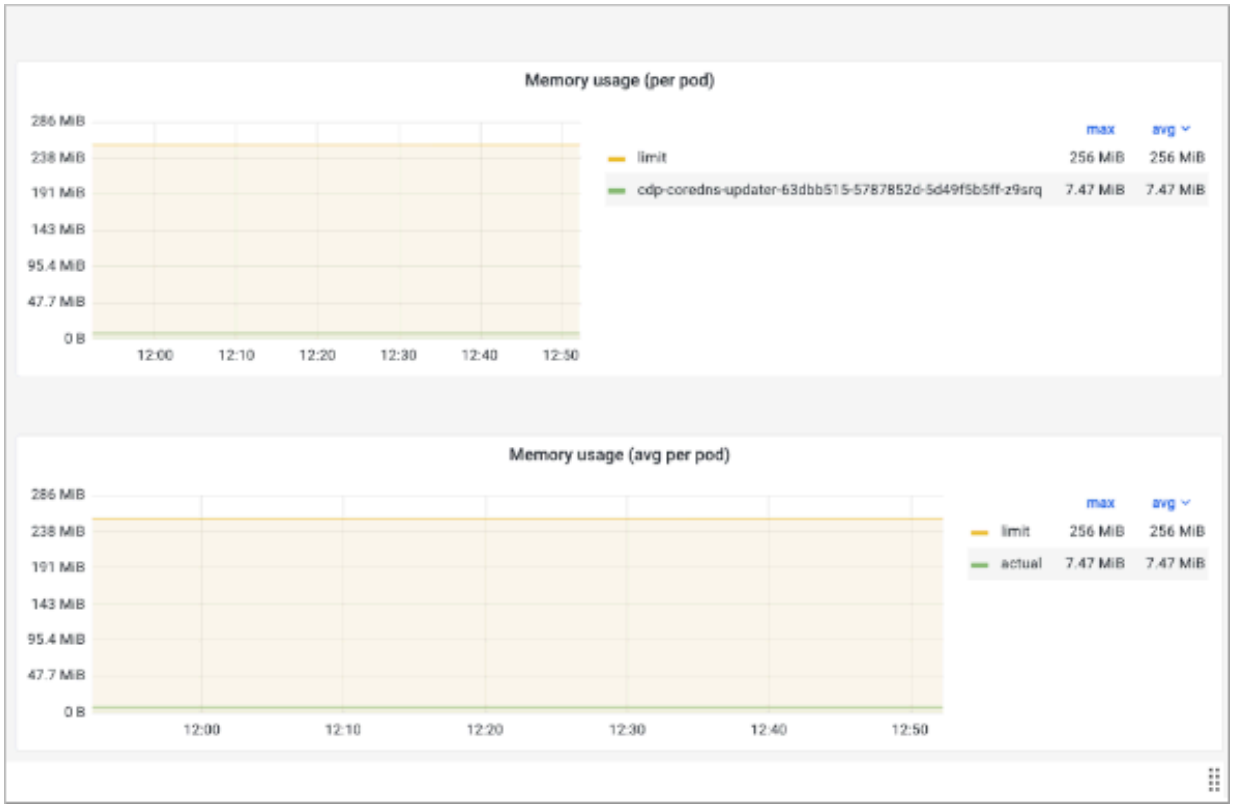

6. From the kubernetes dashboard group, click Usage Report Overview, and see the active nodes on the X axis of the Active Node Count graph.

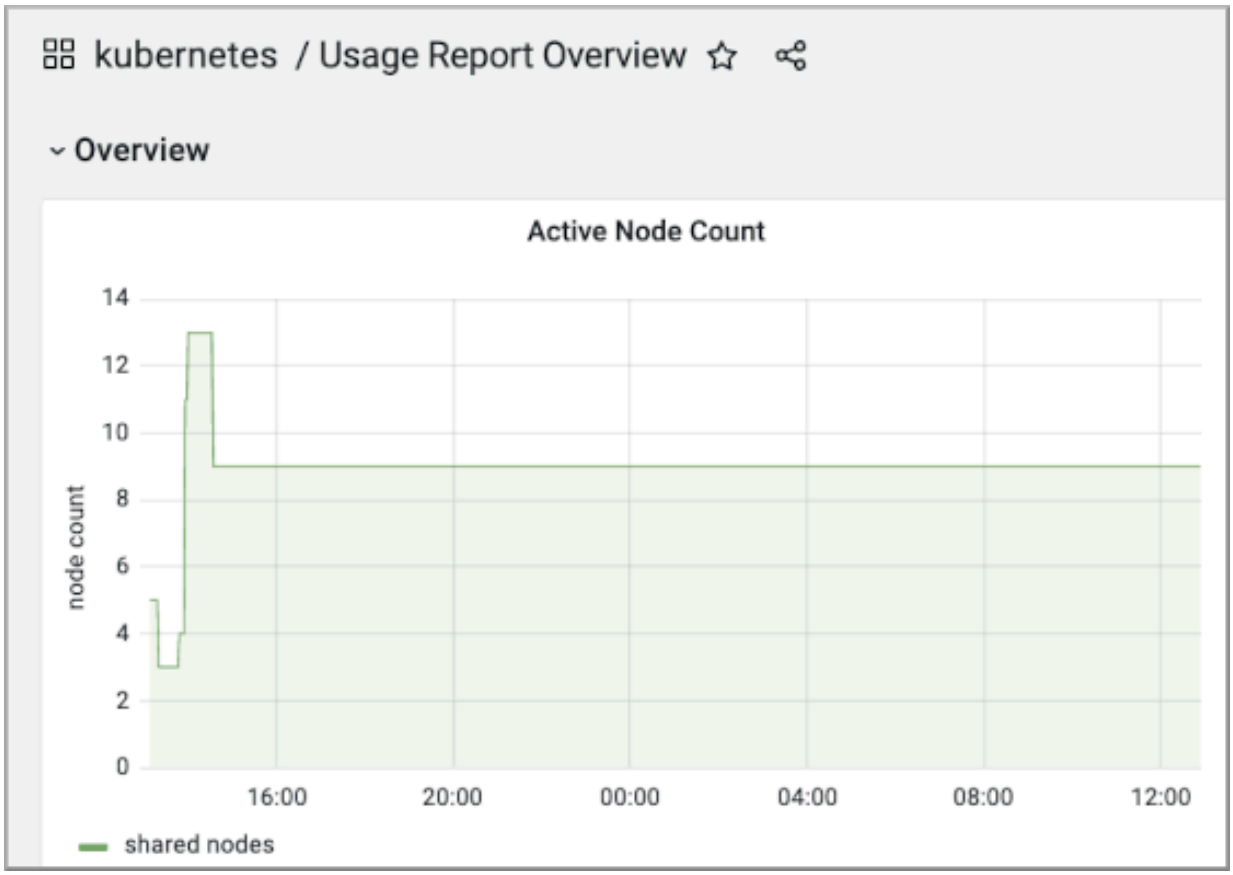

Updating a dashboard graph

You can view, edit, share, explore, or remove a dashboard graph, and more.

Procedure

1. Open a dashboard, for example the HiveServer2 dashboard, and click the title in a row, for example, the **App Stop times** title.

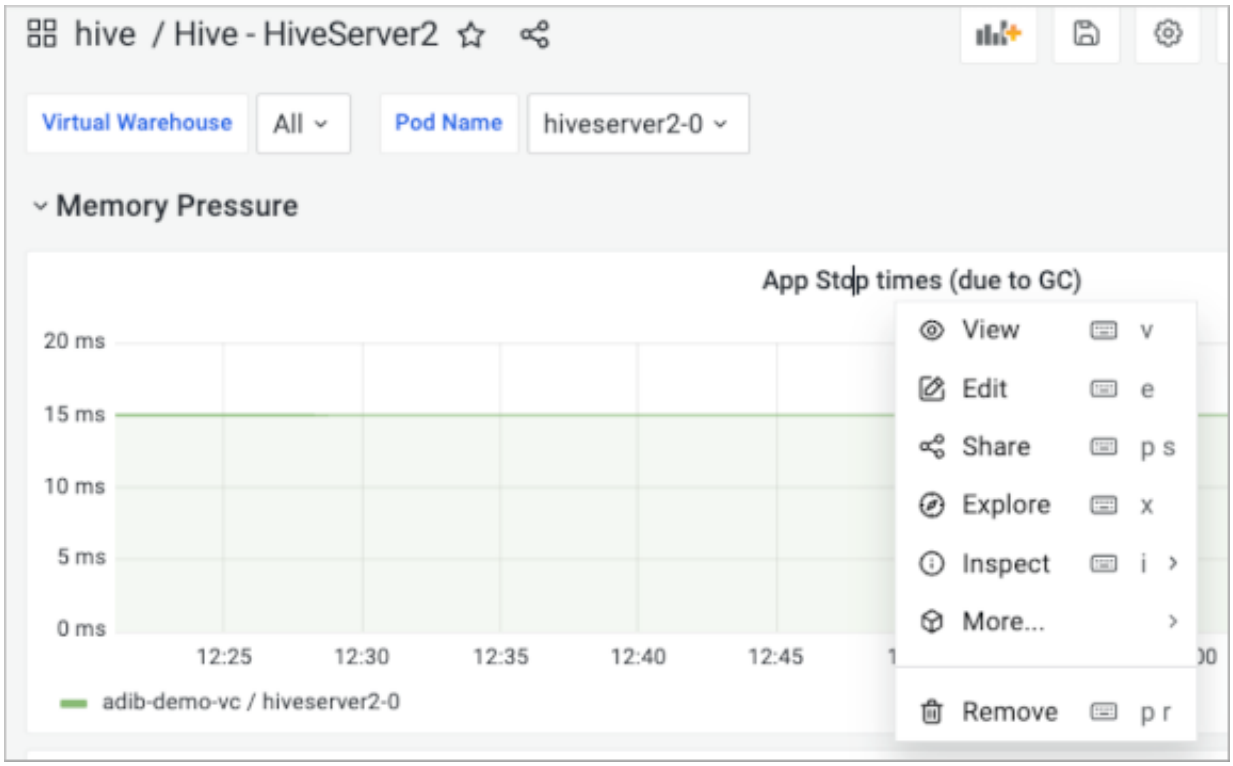

2. Click Edit.
   The edit panel appears.

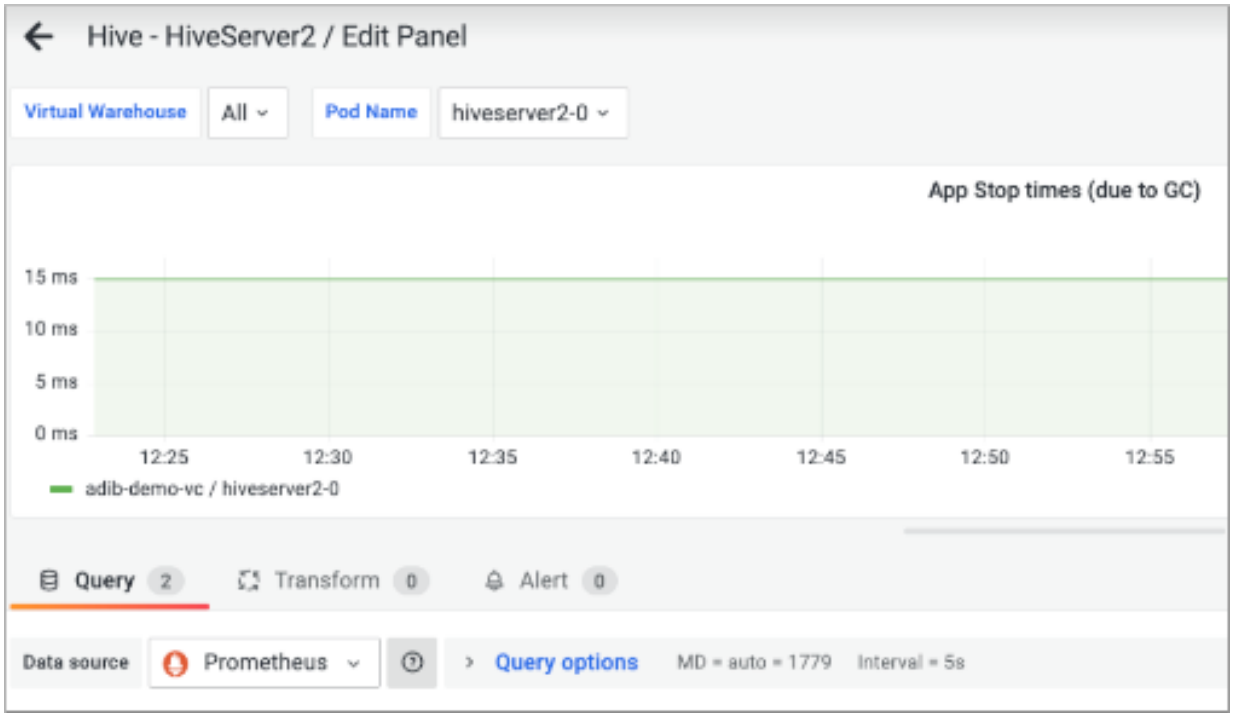

3. At the top left side of the edit panel, change the Virtual Warehouse and Pod on which the graph is based.
4. At the top right side of the edit panel, click ⚙ to add variables and permissions to the panel.

5. On the right side of the edit panel, select a graph type, such as Time Series, and specify panel options, such as the title of the graph, tooltip mode, legend, graph styles, and more.

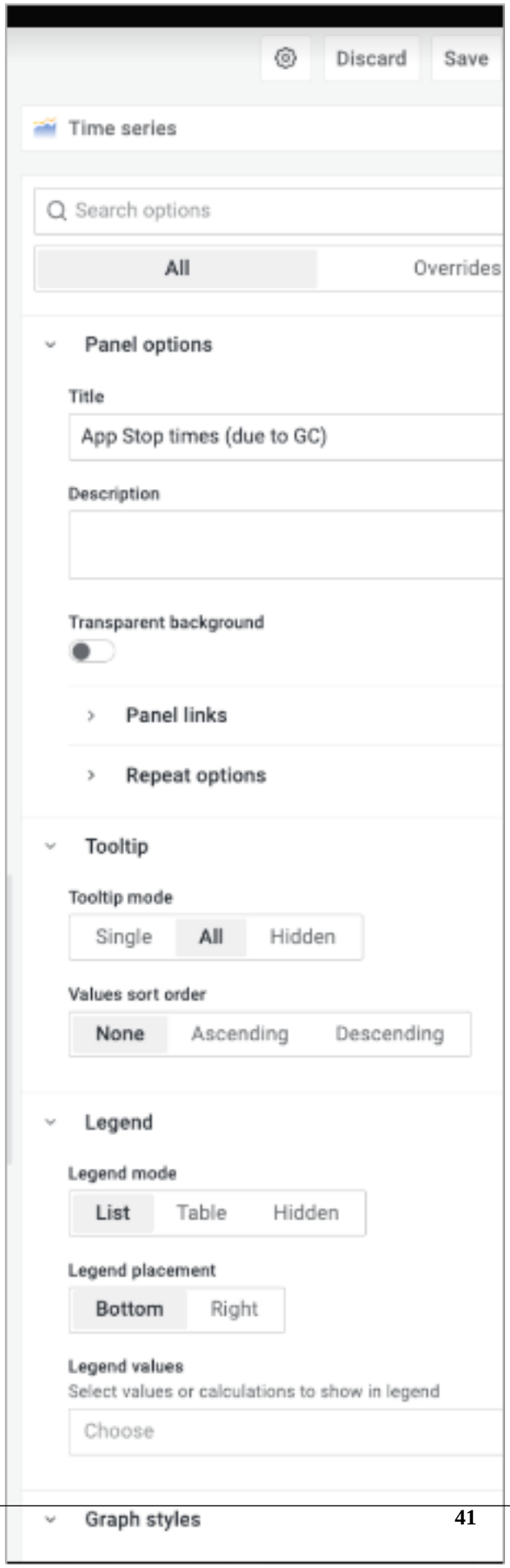
Discard
Save
Time series
Search options
All
Overrides
Panel options
Title
App Stop times (due to GC)
Description
Transparent background
Panel links
Repeat options
Tooltip
Tooltip mode
Single
All
Hidden
Values sort order
None
Ascending
Descending
Legend
Legend mode
List
Table
Hidden
Legend placement
Bottom
Right
Legend values
Select values or calculations to show in legend
Choose
Graph styles

6. Scroll down the edit panel, and change query options.
   For example, in Metrics browser, change the metrics to be graphed from jvm_pause_extrasleeptime.

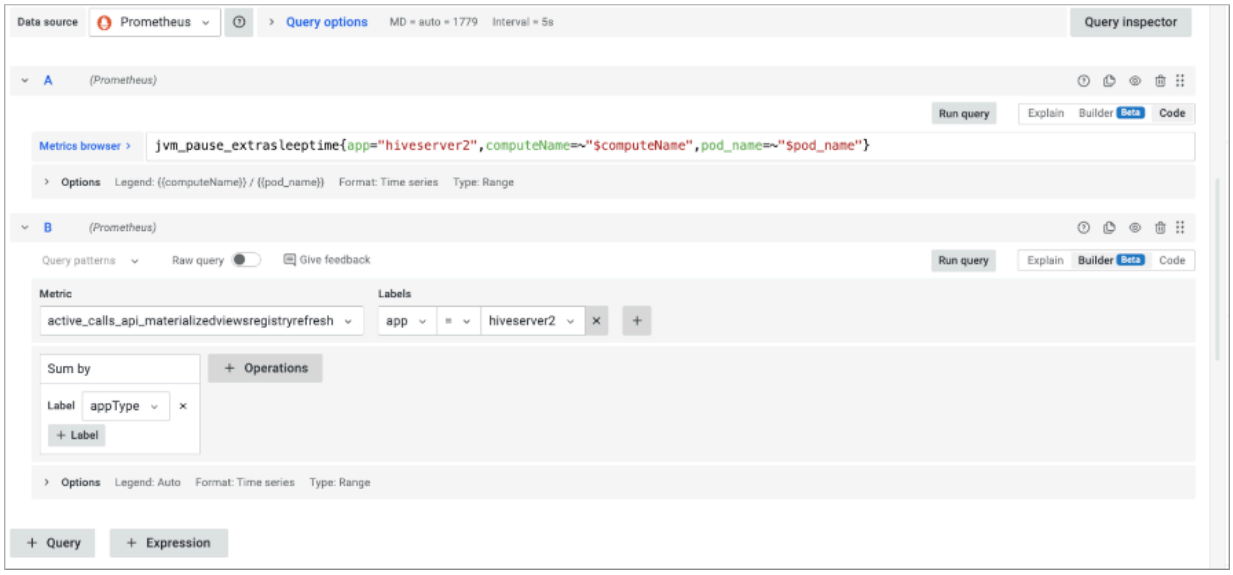

7. 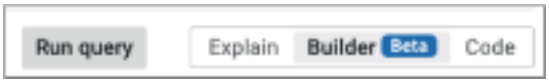

   On the right side of the edit panel, use the Builder code option to frame the query and Explain to understand the query.
8. At the bottom of the edit panel, click + Query to add a query.
9. At the top-right, click Apply to save changes.

Inspecting dashboard data and queries

You can drill down into details of resource usage of Hive or Impala, get statistics about usage, and run queries on the details.

Procedure

1. Open a dashboard, for example the HiveServer2 dashboard, and click the title in a row, for example, the **Heap usage** title.

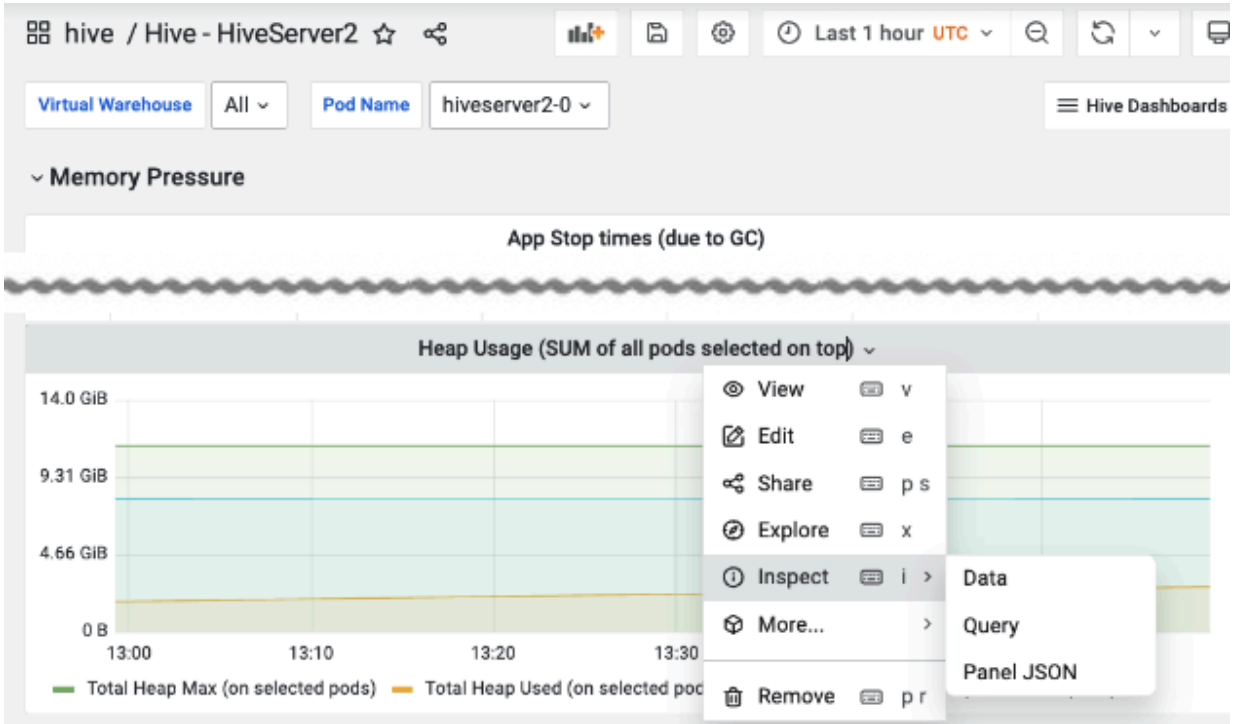

2. Click Inspect Data .
The total max heap usage over time appears.

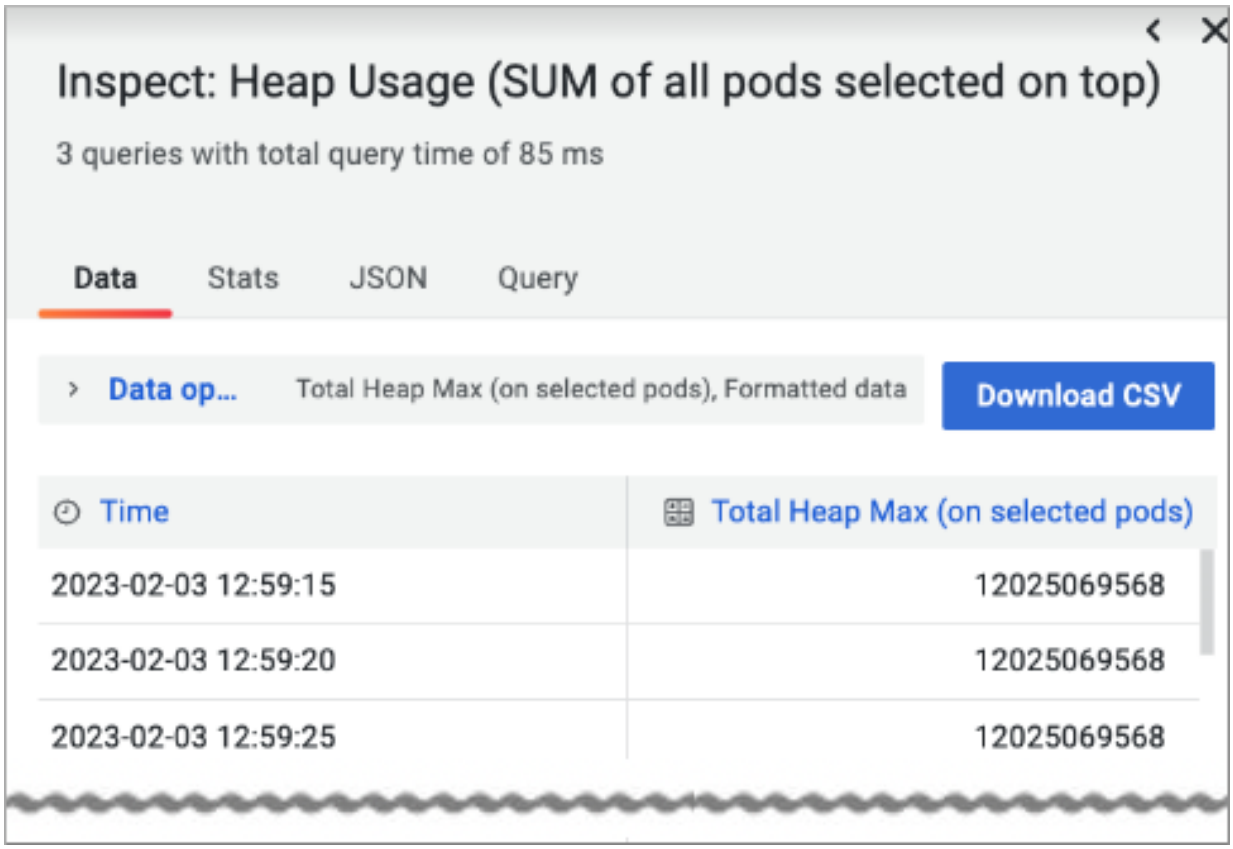

3. Click Stats or Query to view statistics or queries.
4. In the HiveServer2 dashboard, and click the Heap usage title More Duplicate panel content.

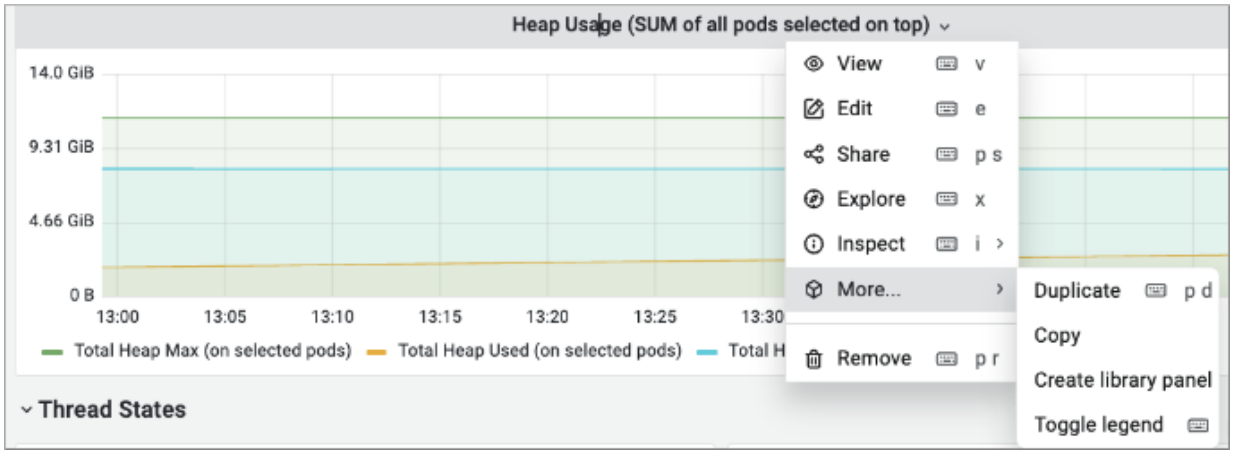

5. In the HiveServer2 dashboard, and click the Heap usage title More Copy panel content to another dashboard.

Creating a custom dashboard

You can create a custom dashboard in a few steps.

Procedure

1. Log into Grafana, and in the Welcome screen, click + Create

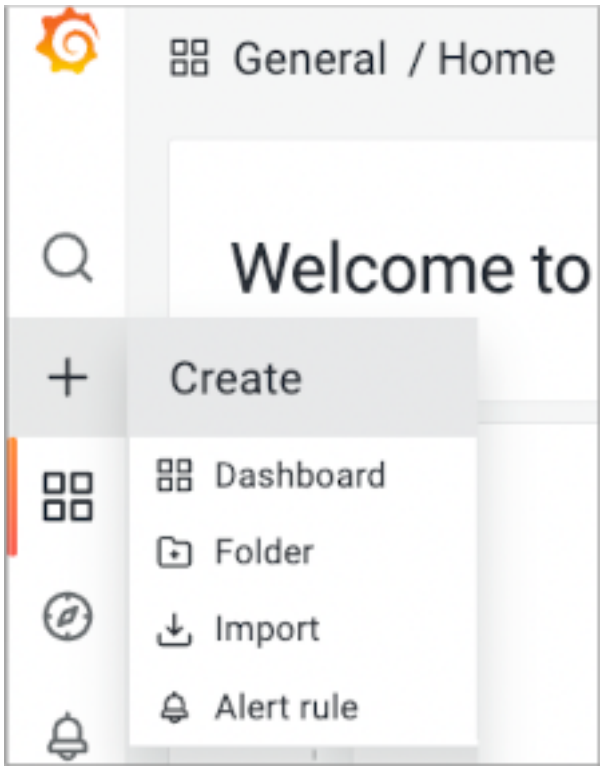

New dashboard appears.

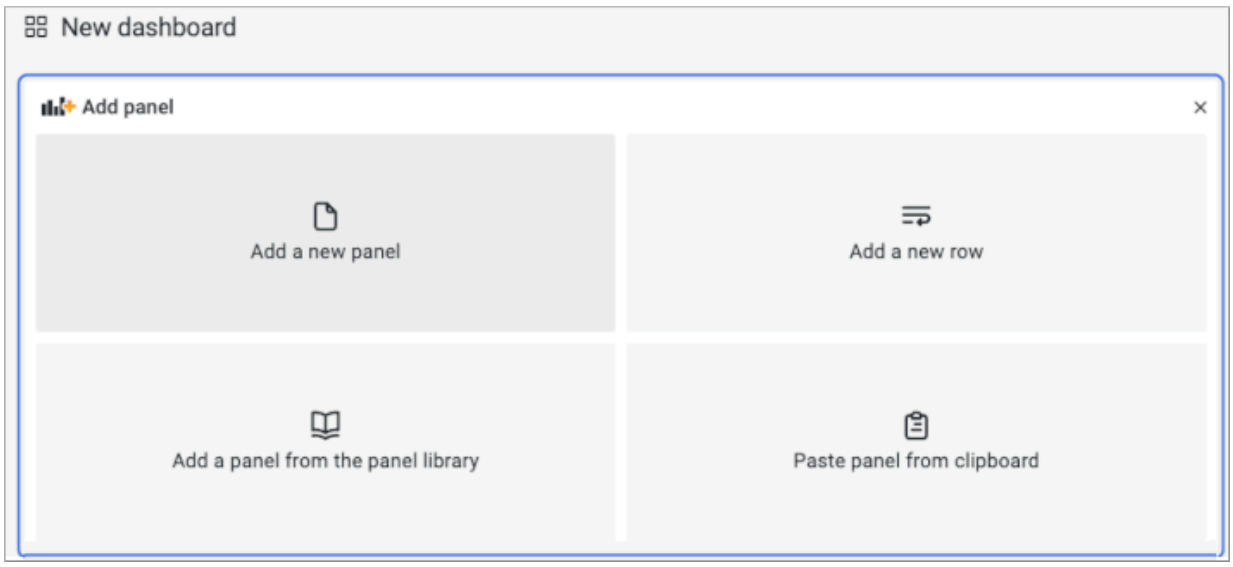

2. At the top right side of the edit panel, click ⚙ to add variables, permissions, and names to the new dashboard.

3. Click Add a new panel.

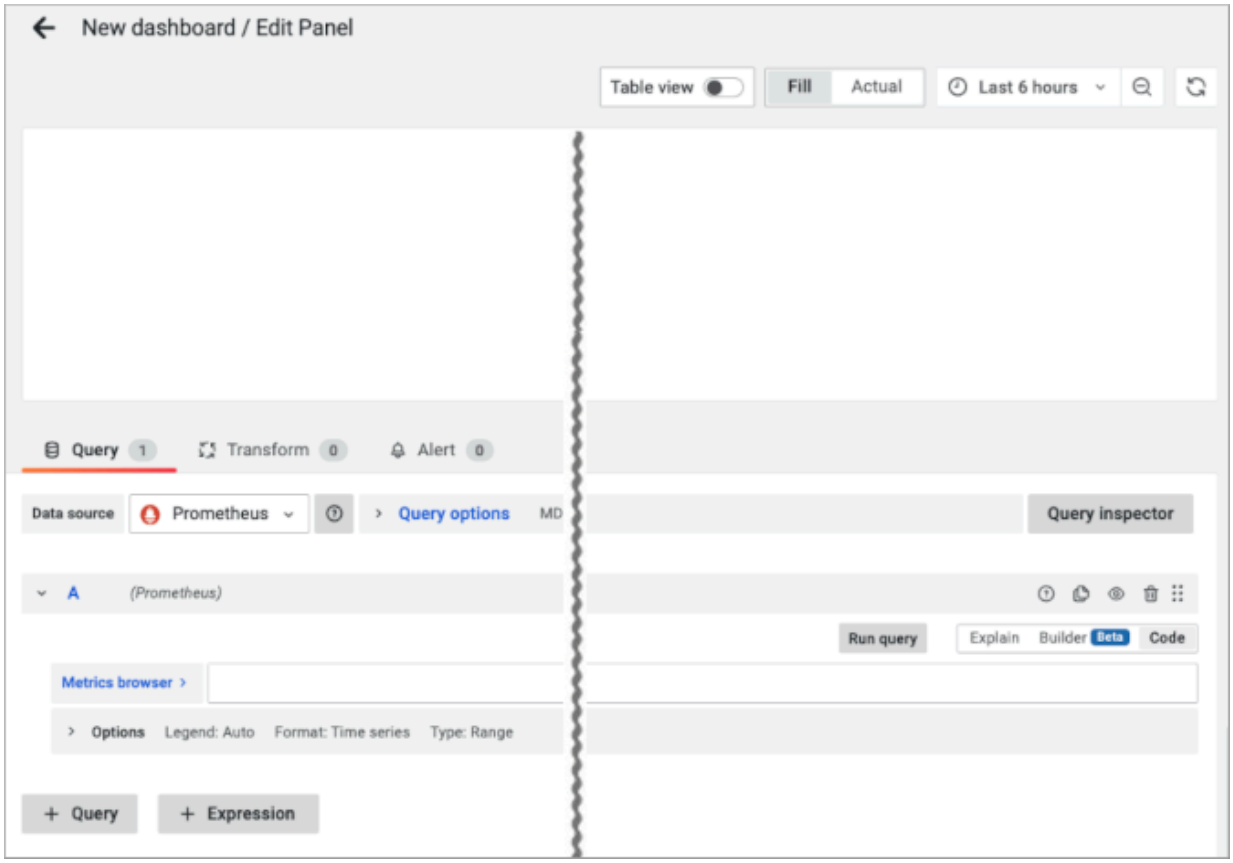

4. On the right side of the New edit panel, click ⚙ to add variables and permissions to the new dashboard.

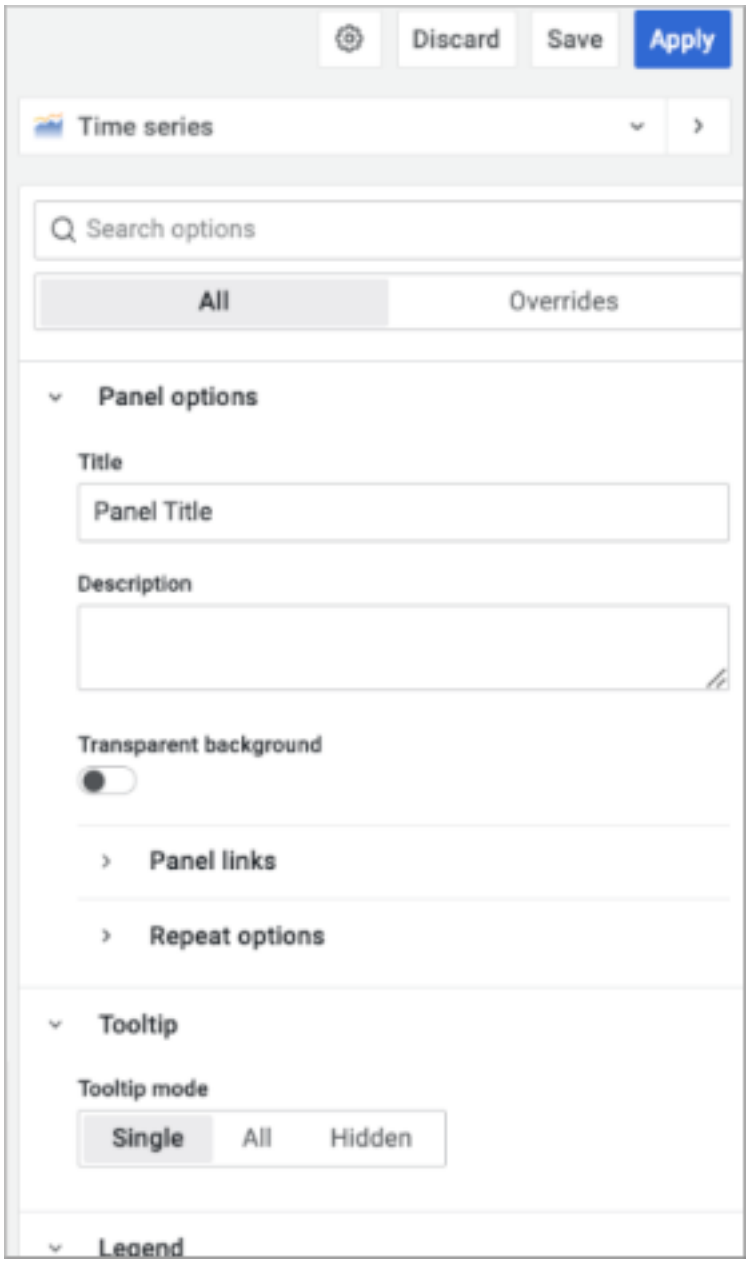

5. Edit the panel and rename it test1.

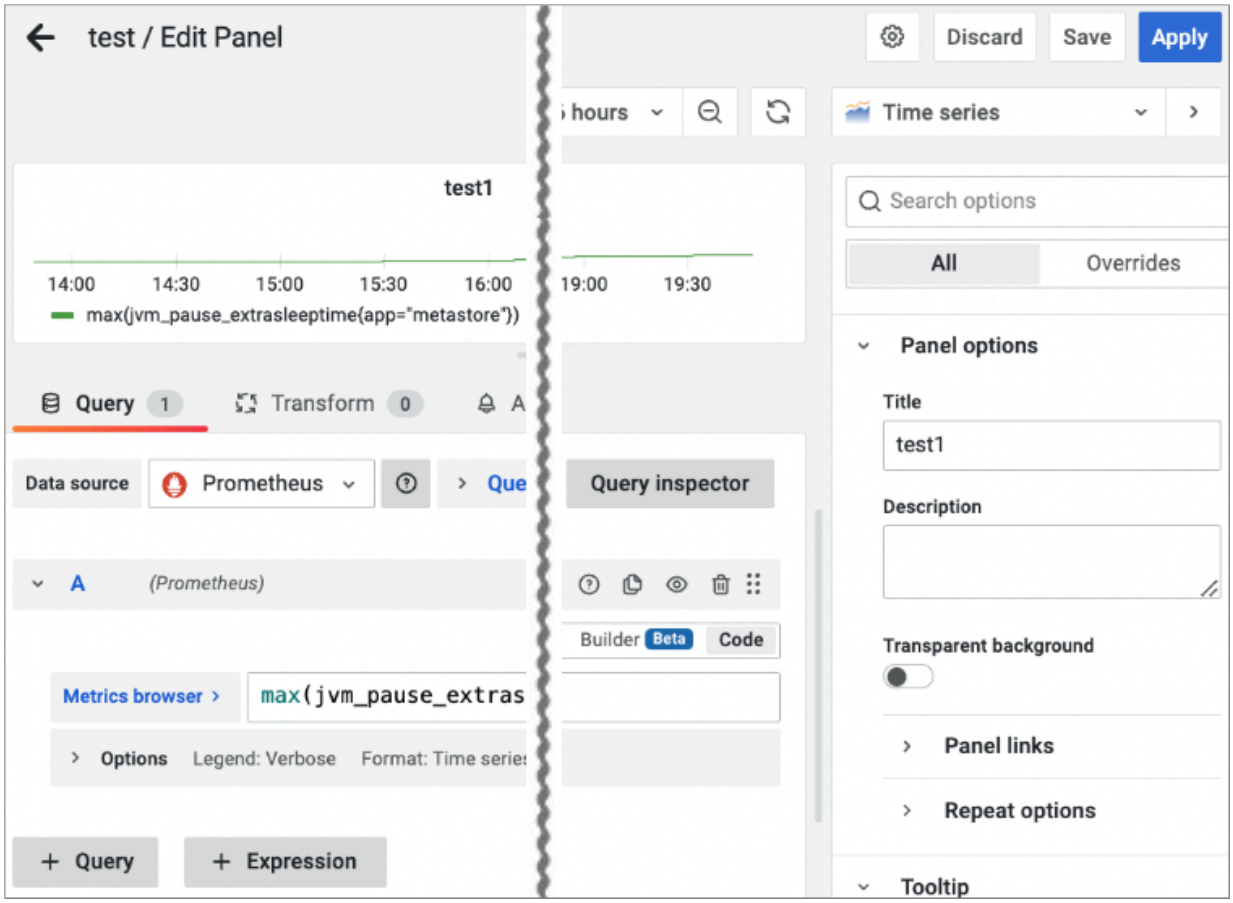

6. Click Apply.
   Test1 is added to the dashboard.

7. Edit the panel, rename it test2, and click Apply.
   Test2 is added to the dashboard with Test1.

Meeting prerequisites to set up alerts

Before you can set up an alert triggered by a dashboard event, you must configure an SMTP (Simple Mail Transfer Protocol) server, an alert recipient, and a notification policy.

Procedure

1. Configure grafana using kubectl.

```
kubectl edit configmap grafana -n istio-system
```

   For more information, see "Granting remote access to Kubernetes".

2. Update the grafana.ini to configure SMTP, setting user to the altert account email id and the password to the app password generated in your gmail account.

```
...
[smtp]
enabled = true
host = smtp.gmail.com:465
user = youremail@gmail.com
password = substitute_your_password
;cert_file=
;key_file=
skip_verify = true
from_address = alertgrafan@gmail.com
from_name = Grafana
* EHLO identity in SMTP dialog (defaults to instance_name)
;ehlo_identity = dashboard.example.com
```

```
;startTLS_policy = NoStartTLS
```

3. Refresh the grafana pod.

```
kubectl get pod -A grep -i grafana
kubectl delete pod grafana-9bcb5d5c-w8r6c -n istio-system
```

4. Log into Grafana, and click Alerting Contact points .

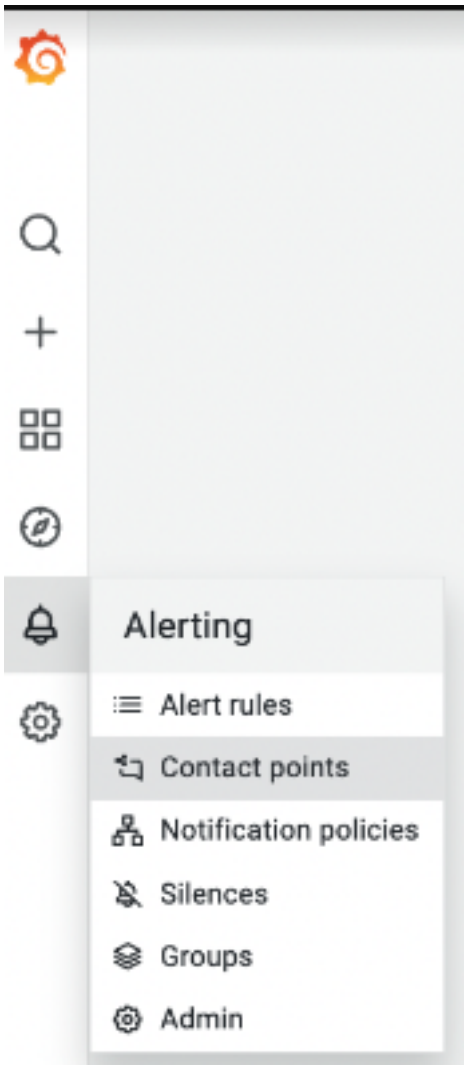

5. In Contact point type, select an alert receiver type.

6. In Addresses, enter an email address, and then in Message, compose an email message body.
   For example, enter an email message body test grafana email.

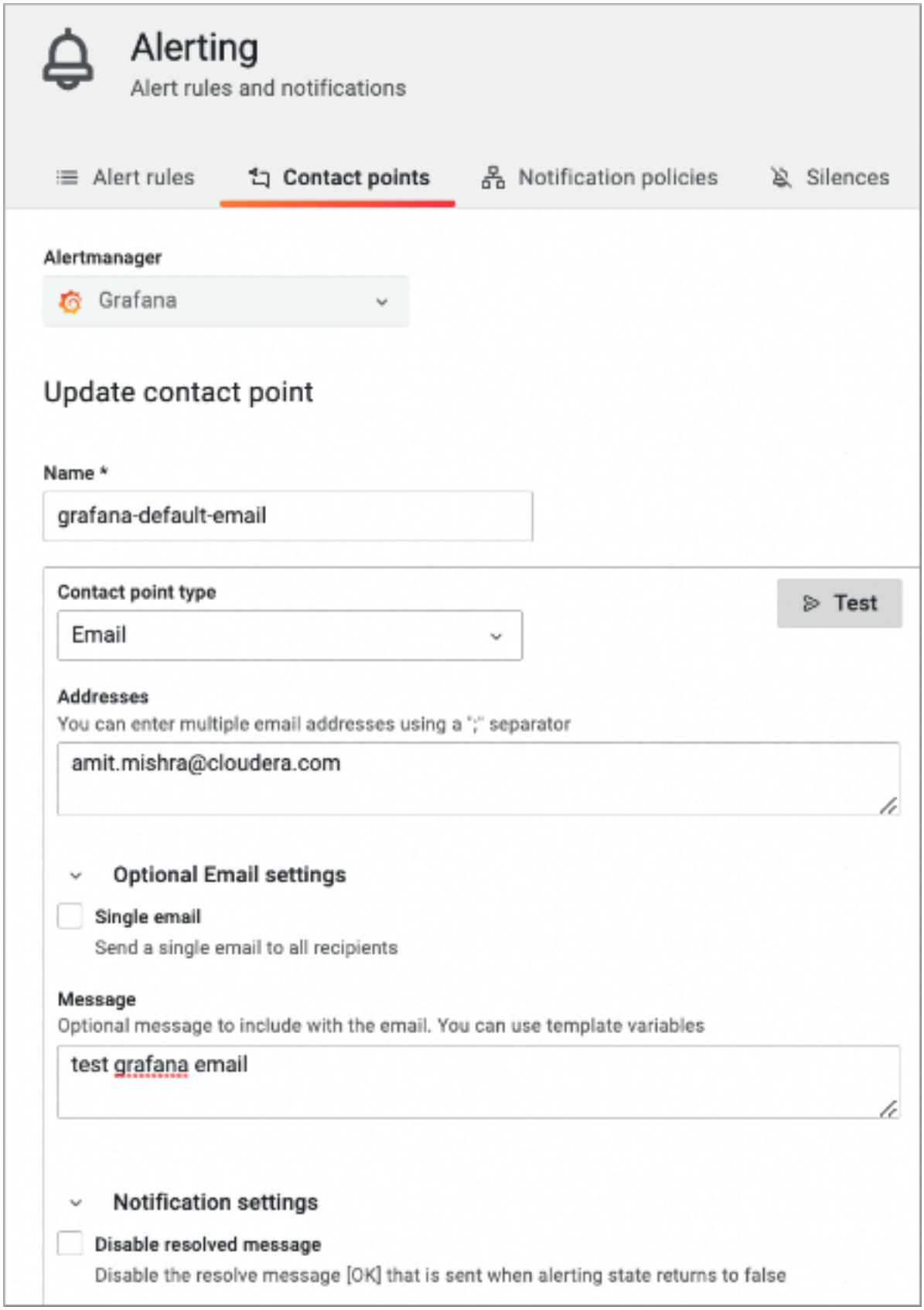

7. Click Test to verify that a test alert arrives in your inbox.

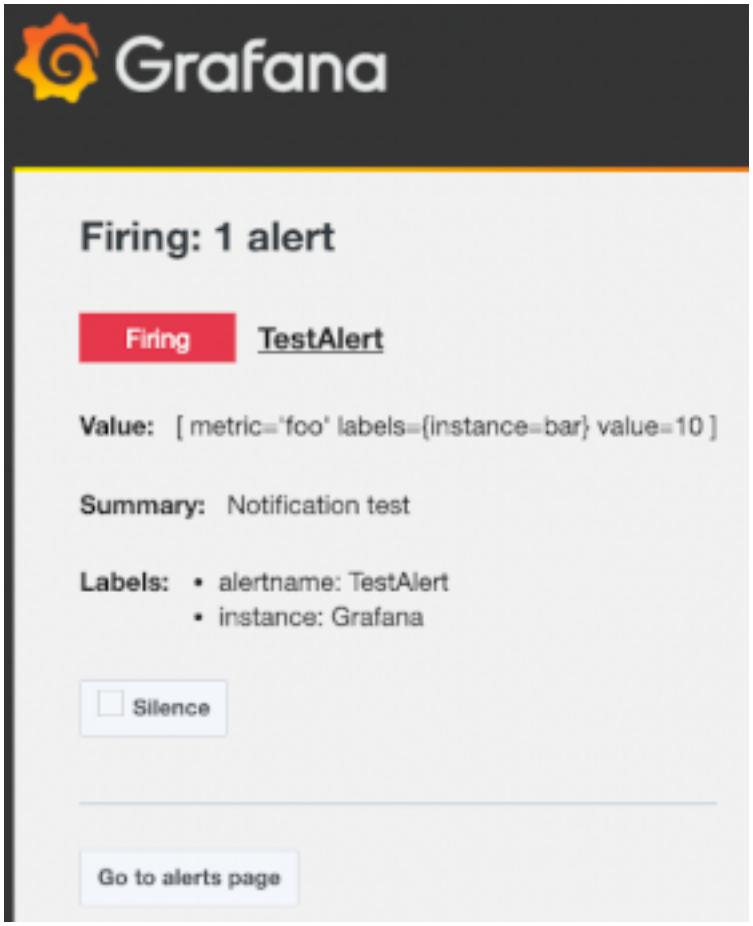

8. Scroll up and in Name, enter the contact point name.

9. Log into Grafana, and click Alerting Notification policies .

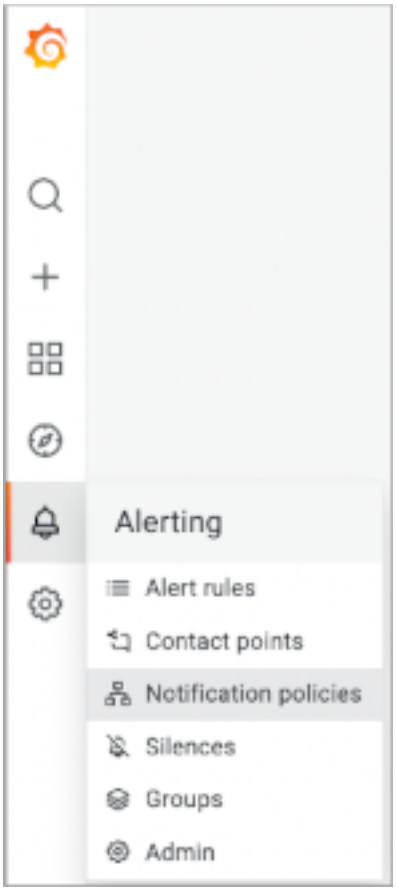

10. Click New specific policy.

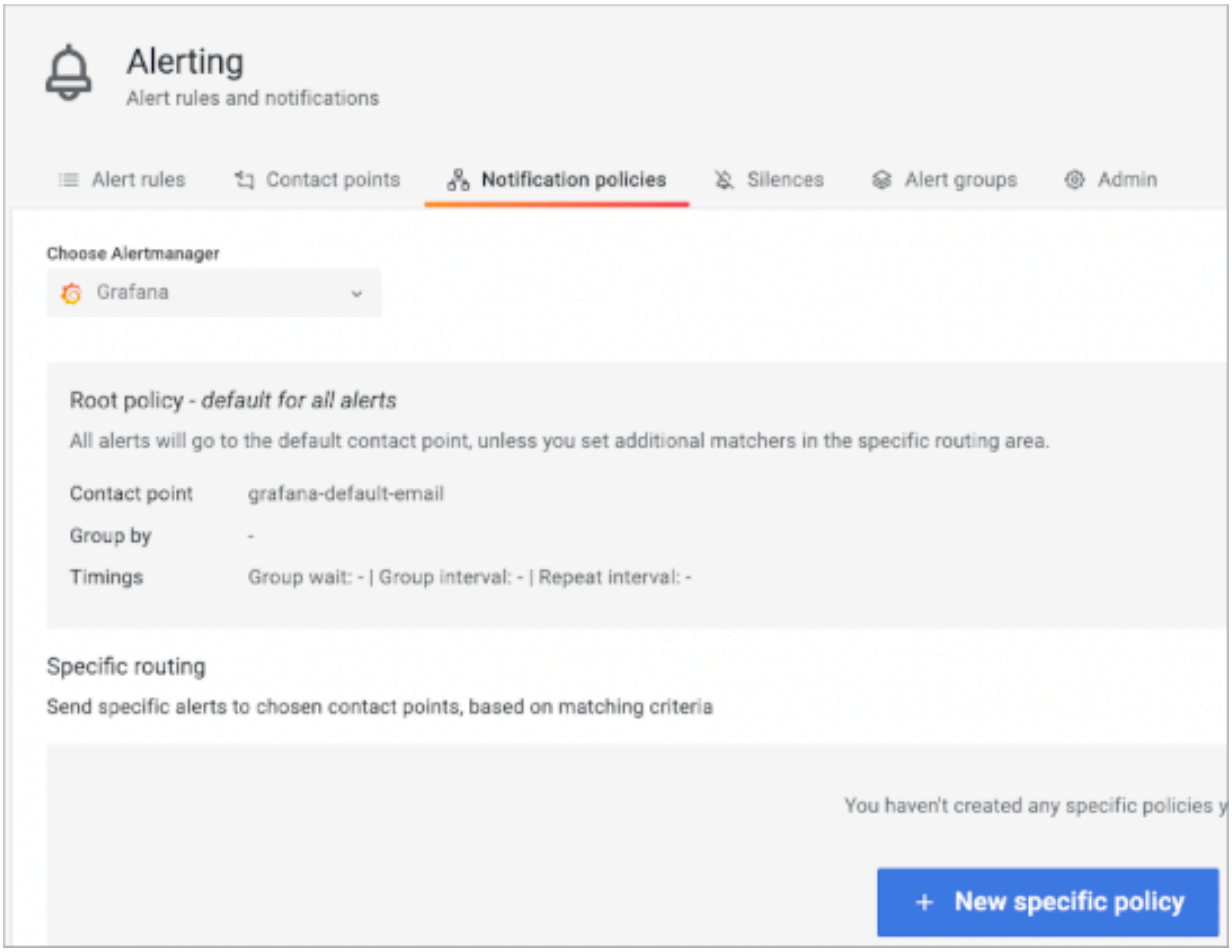

11. In Contact Point, select the contact point to receive notification, and save the policy.

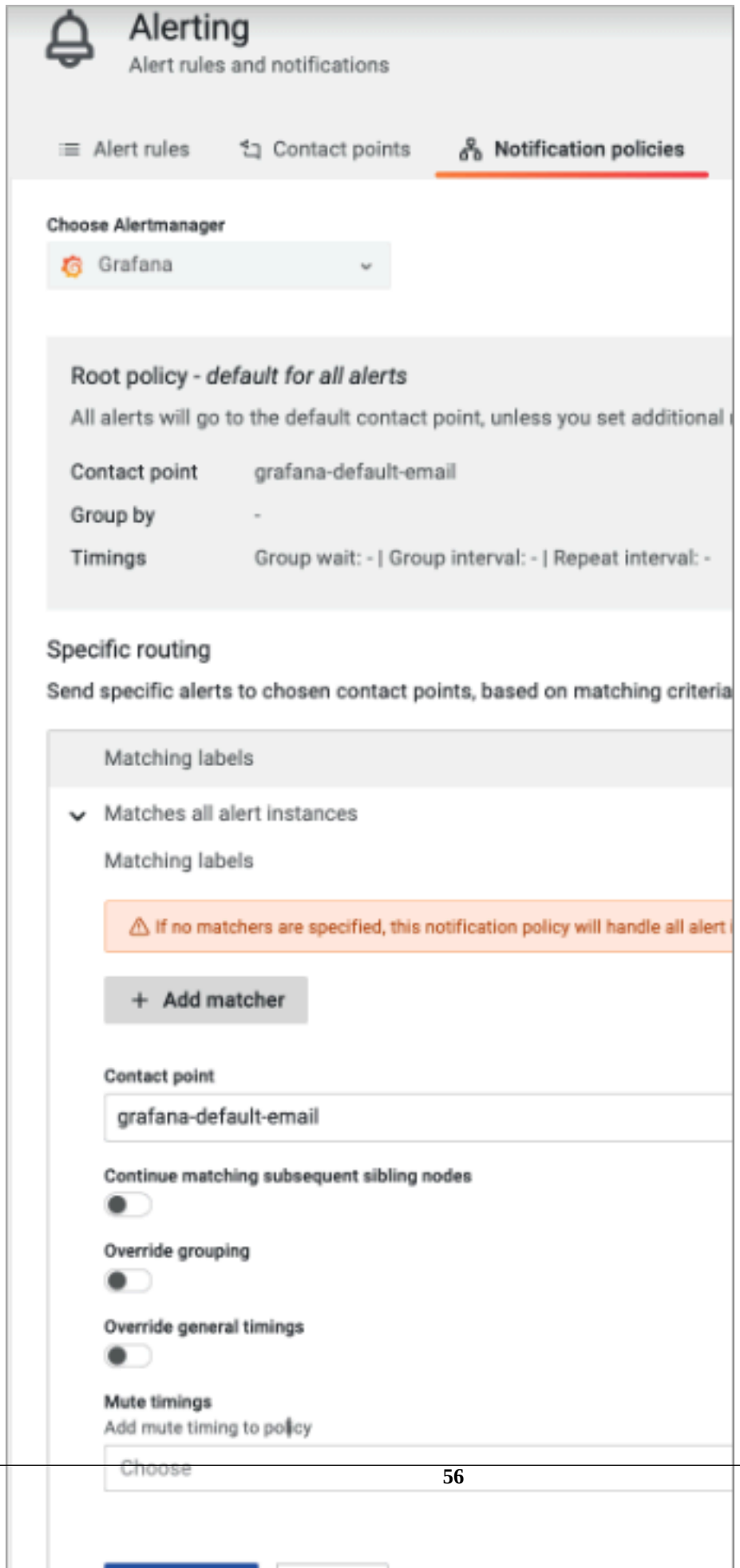
Alerting
Alert rules and notifications
Alert rules
Contact points
Notification policies
Choose Alertmanager
Grafana
Root policy - default for all alerts
All alerts will go to the default contact point, unless you set additional
Contact point
grafana-default-email
Group by
-
Timings
Group wait: - | Group interval: - | Repeat interval: -
Specific routing
Send specific alerts to chosen contact points, based on matching criteria
Matching labels
Matches all alert instances
Matching labels
If no matchers are specified, this notification policy will handle all alert
+ Add matcher
Contact point
grafana-default-email
Continue matching subsequent sibling nodes
Override grouping
Override general timings
Mute timings
Add mute timing to policy
Choose

Creating alerts

After meeting prerequisites to set up alerts, you can create an alert for an existing dashboard.

Procedure

1. Click the title of the graph that you want to fire an alert, and select Edit.

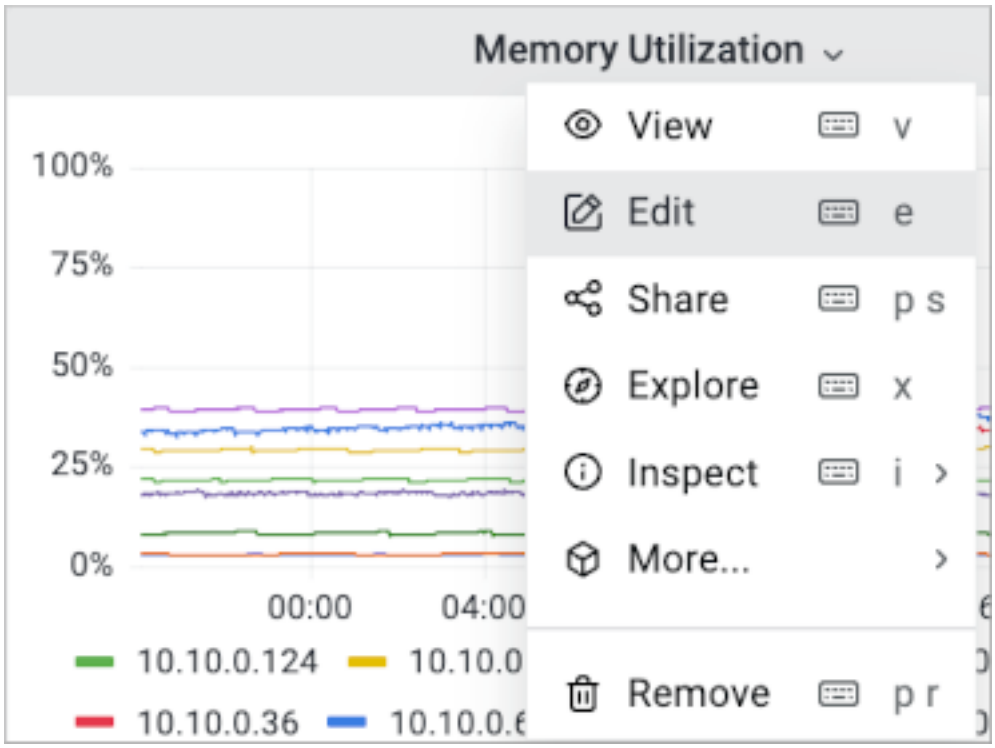

2. In the edit panel that appears, click Alert Create alert rule from this panel .

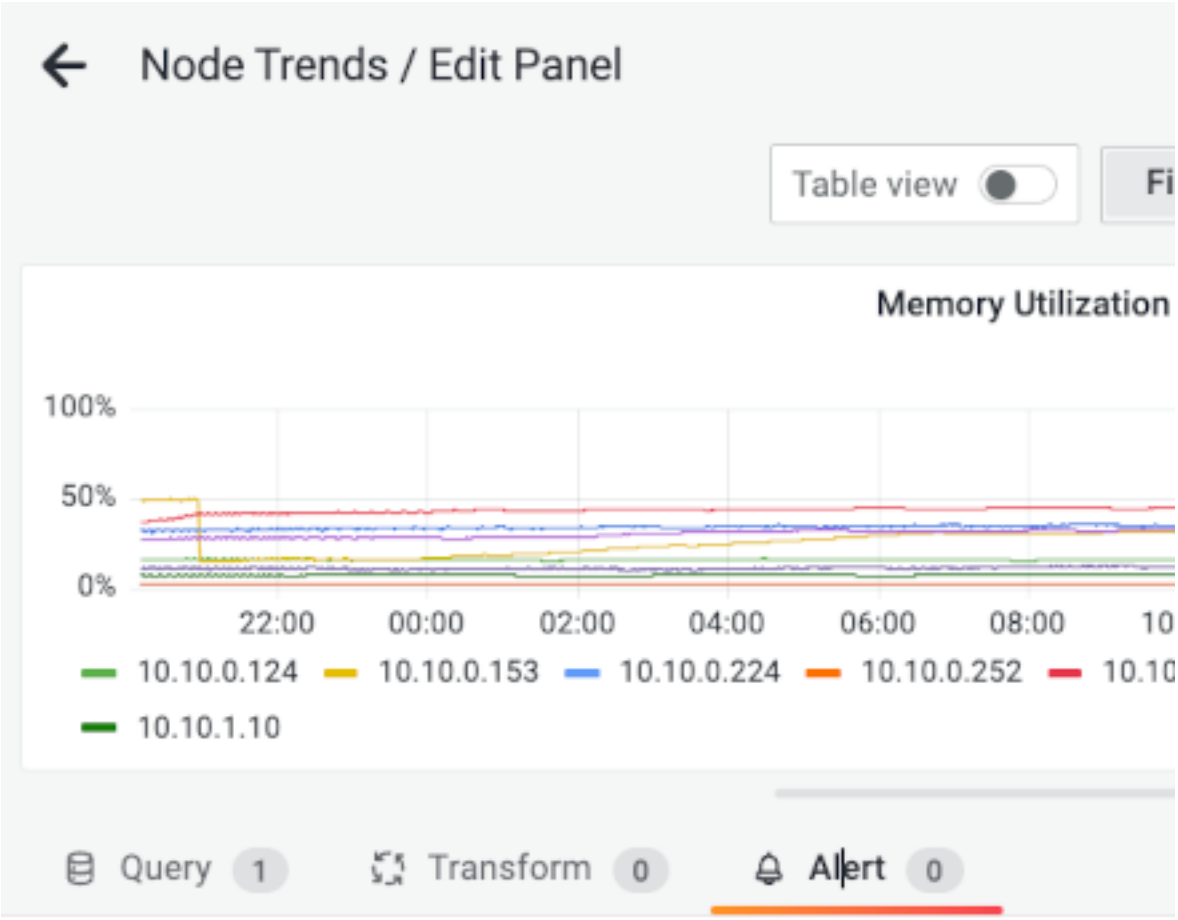

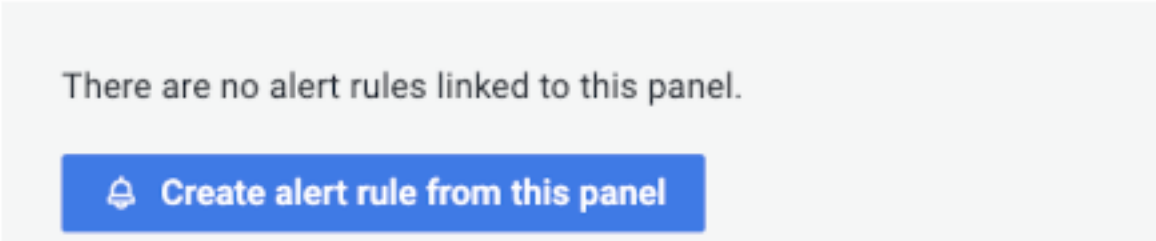

Create alert rule appears. The Rule name is set to Memory Utilization.

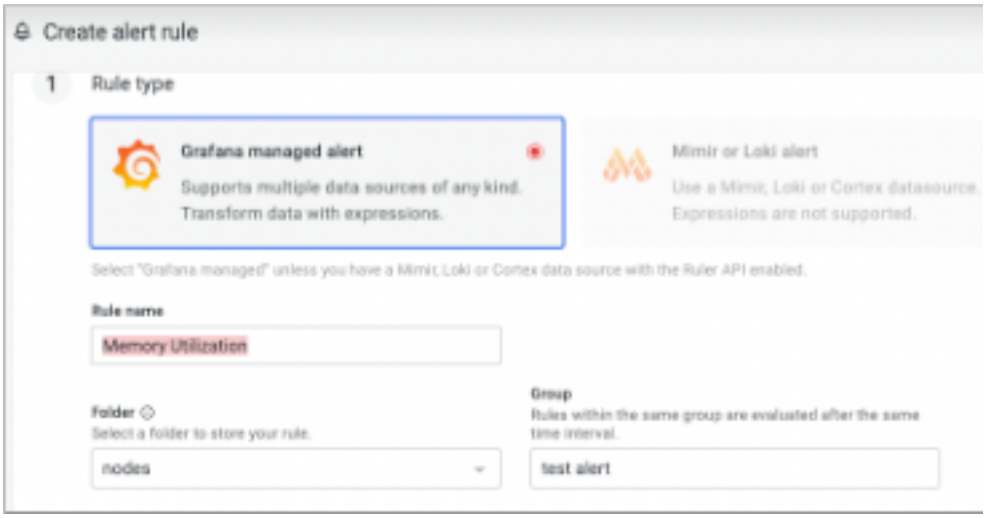

3. In **Create alert rule** scroll down, and in Group, enter a meaningful name for the alert group, for example, test alert..

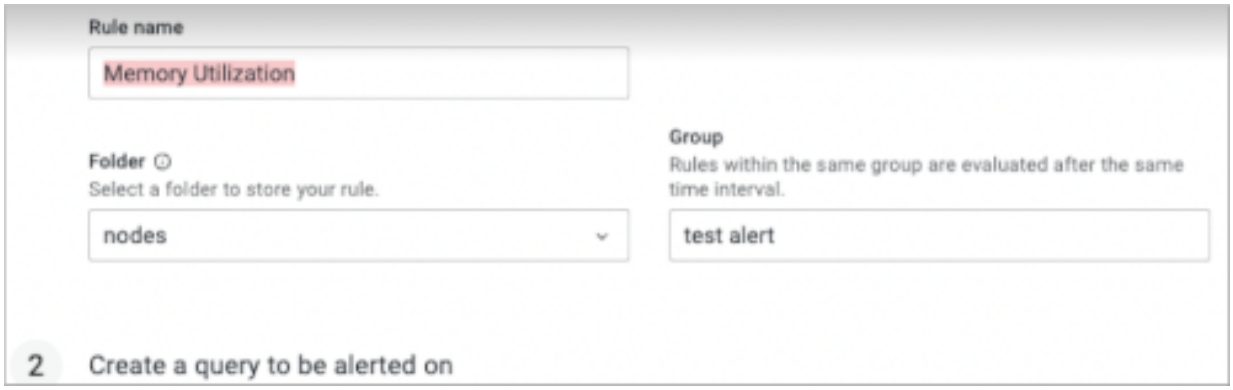

4. In **B**, define a classic condition to trigger the alert, for example when last() of A is above 20.

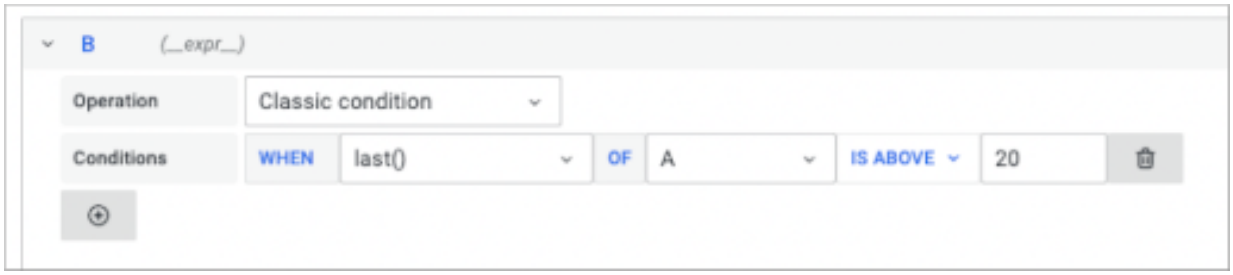

5. In **Define alert conditions**, select B for the expression that triggers the alert.

6. Click Preview alerts, and then scroll up to the top right of **Create alert rule**, and click Save.

Reviewing alerts and notifications

Procedure

1. Log into Grafana, and click Alerting Alert rules .

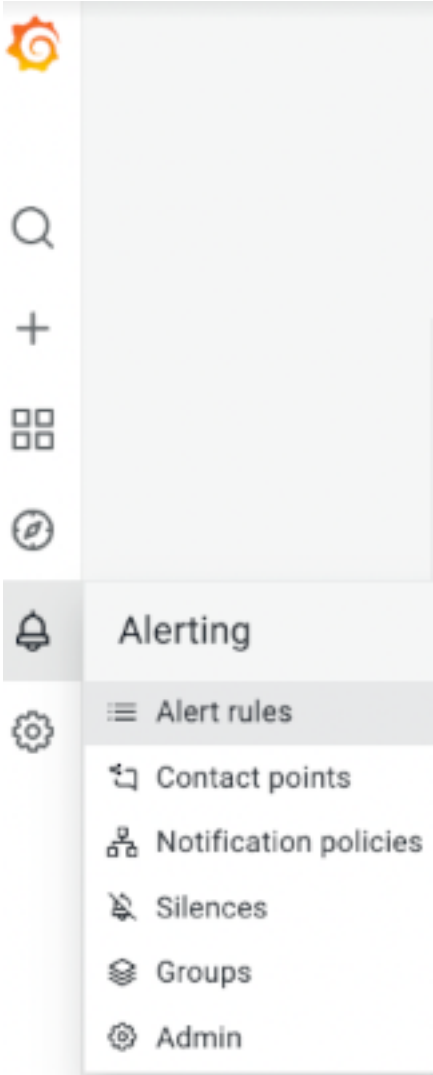

2. In Alert rules, in the **hive** directory, click Firing to review the alert in the UI.

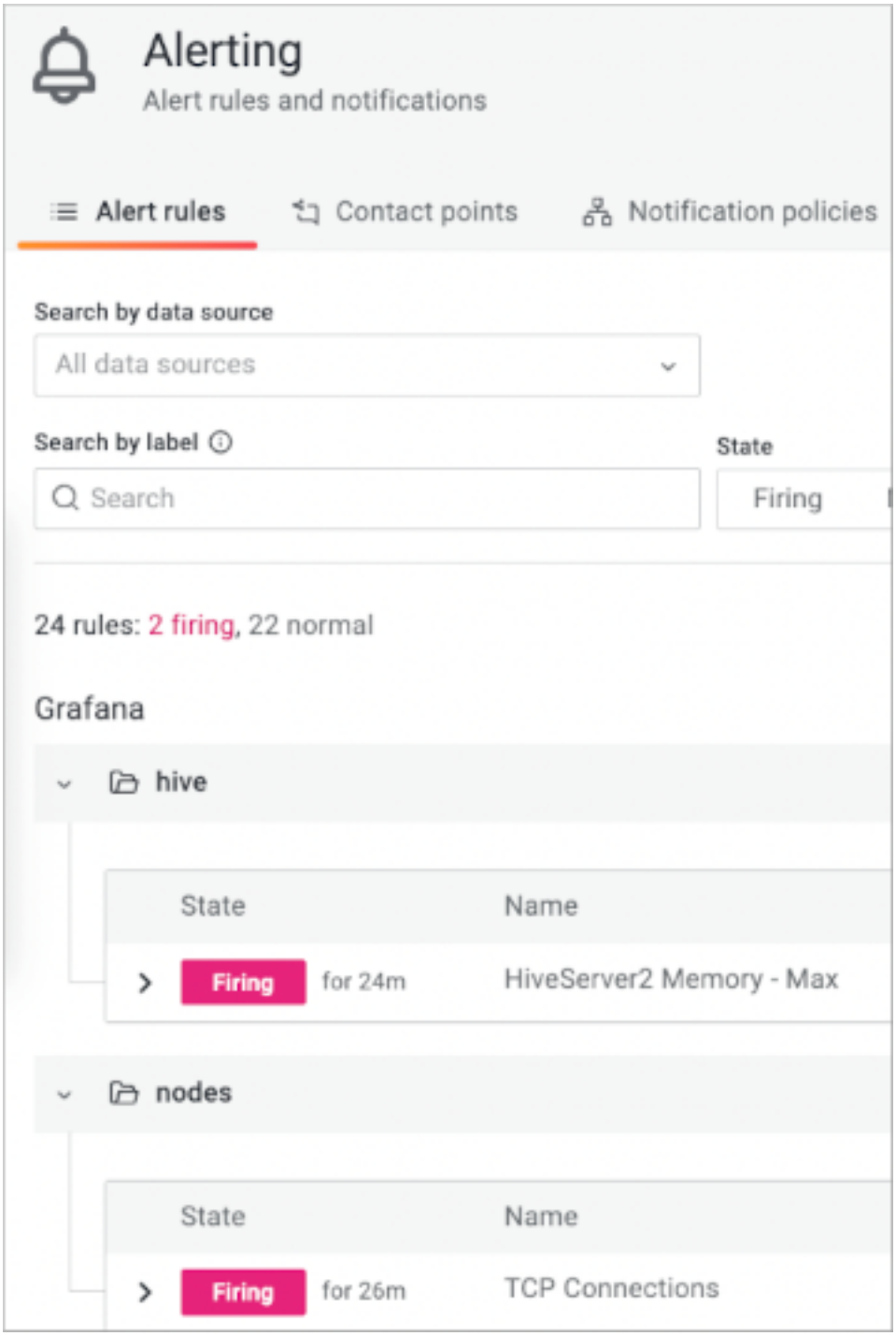

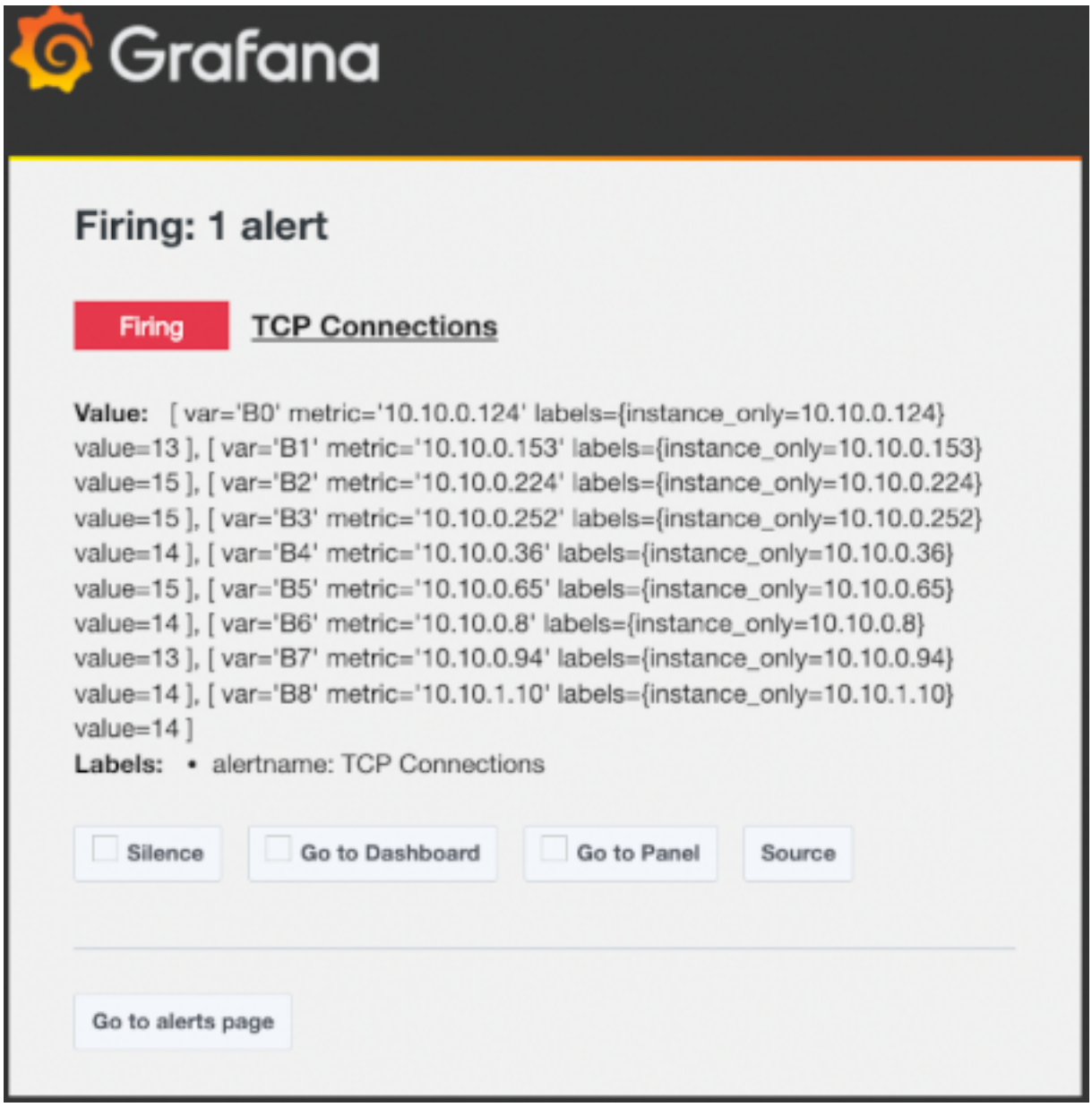

Forwarding Prometheus metrics from Cloudera Data Warehouse to an endpoint

You can configure Prometheus in Cloudera Data Warehouse to push its metrics to an external endpoint, such as Prometheus, Grafana, Thanos, or some other endpoint.

About this task

For information about other receivers, see Prometheus documentation: Compatible senders and receivers.

In this procedure you add a configuration snippet that triggers a Prometheus Remote Write. Prometheus then sends HTTP POST requests to the Remote Write endpoint to forward your metrics.

Before you begin

Cloudera Data Warehouse must run in an AWS or Azure environment.

Procedure

1. In **Environments**, locate your Cloudera Data Warehouse environment.
2. Click ⋮, and select Edit Observability .
3. In Metrics Forwarding Configuration, add the config and authentication methods for your endpoint in YAML format.
   For example:

```
remote_write:
  - url: https://aps-workspaces.us-west-2.amazonaws.com/workspaces/ws-3e8f
cedb-6727-4912-82aa-f05a8be25fe4/api/v1/remote_write
    sigv4:
      region: us-west-2
    name: prometheus_remote_write
    queue_config:
      max_samples_per_send: 1000
      max_shards: 200
      capacity: 2500
```

4. Click Apply Changes.

Results

This action updates the config map. The Prometheus pod restarts, so the new config takes effect.

Monitoring Kubernetes resources from K8s dashboard

As a Cloudera Data Warehouse administrator, you can monitor Kubernetes resources in your Cloudera Data Warehouse cluster. On the K8S dashboard, you can view the state of the resources, such as CPU and memory usage, see the status of pods, and download logs.

The dashboard can provide insights into the performance and health of a Cloudera Data Warehouse cluster. From the dashboard, if authorized, you can monitor the environments of any Cloudera Data Warehouse cluster efficiently. You do not need to copy/paste kubeconfig files to switch to monitoring another environment, Monitoring the dashboard can help keep your cluster running smoothly and efficiently.

By default, the dashboard is disabled. You follow instructions in the next topic to activate and use the dashboard. Using the dashboard incurs some cloud cost, and is designed to time out automatically after 4 hours to prevent wasting resources. Cloudera recommends deactivating the dashboard when not in use to reduce cloud expenses.

Limitations

Cloudera Data Warehouse does not support using the Kubernetes dashboard in environments with internal load balancer (Enable internal load balancer (ingress) option) on Azure, and with Private Load Balancer mode in AWS.

Prerequisites

- You have an AWS or Azure environment in Cloudera Data Warehouse Public Cloud.
- You have activated your environment.
- You obtained the Cloudera Data Warehouse Admin role.

Activating the K8S dashboard

From your environment, you can flip a switch to activate, or deactivate the K8S dashboard. You can then click a link to view the K8S dashboard in your web browser.

Before you begin

You must meet the prerequisites listed in the previous topic.

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate the environment that you want to view.
3. Click ⋮ Edit .

   The **Environment Details** page is displayed.
4. Toggle Activate Dashboard to on.
5. Wait for the K8S Dashboard Activated success message to appear.
   A link to the K8S dashboard appears in your environment tile.

What to do next

- Follow steps in the next topic to familiarize yourself with the K8S dashboard.
- Deactivate the dashboard when not in use because an activated dashboard incurs a cost.

Using the K8S dashboard

You see how to view the dashboard and get ideas about the types of K8S metrics, charts, and other visualizations that appear on the dashboard. You might use the dashboard to keep your cluster running efficiently and troubleshoot problems.

Before you begin

- You have met the prerequisites listed in Monitoring Kubernetes resources.
- You have activated the K8S dashboard.

Procedure

1. Log in to the Cloudera web interface, navigate to Data Warehouse Overview , and select your environment.
   Environment Details shows the K8S dashboard is activated.
2. In Environment Details, click Open.

   The K8S dashboard appears.

3. View the status of your containerized applications.
   For example:

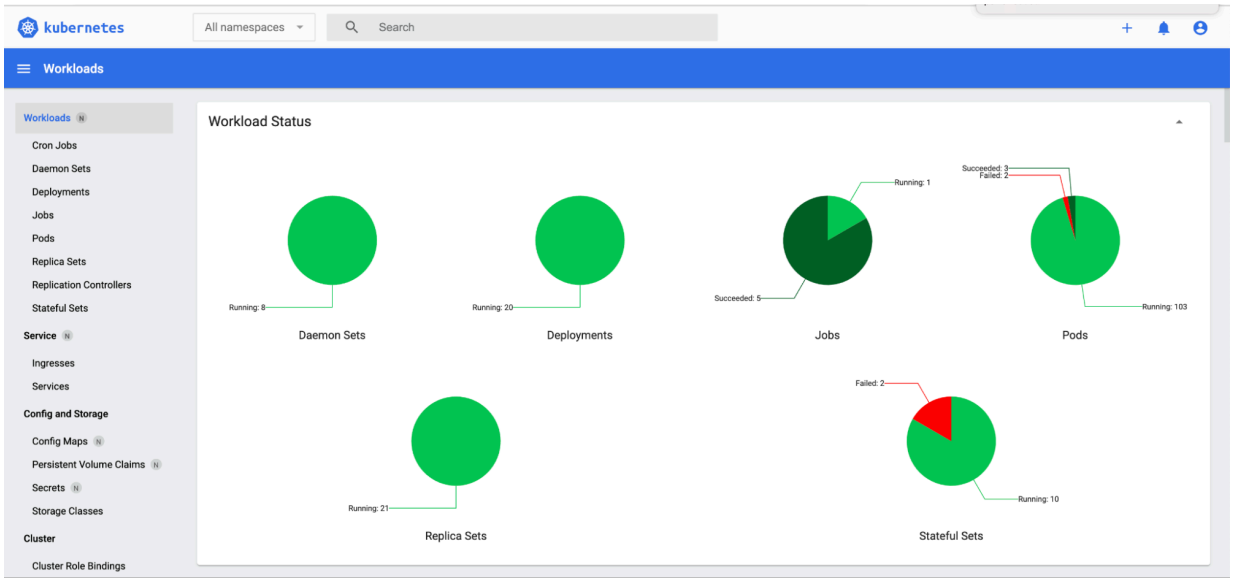

4. Click All Namespaces and select a specific Namespace.

   The workload status of the namespace, impala-167998, in this example appears.

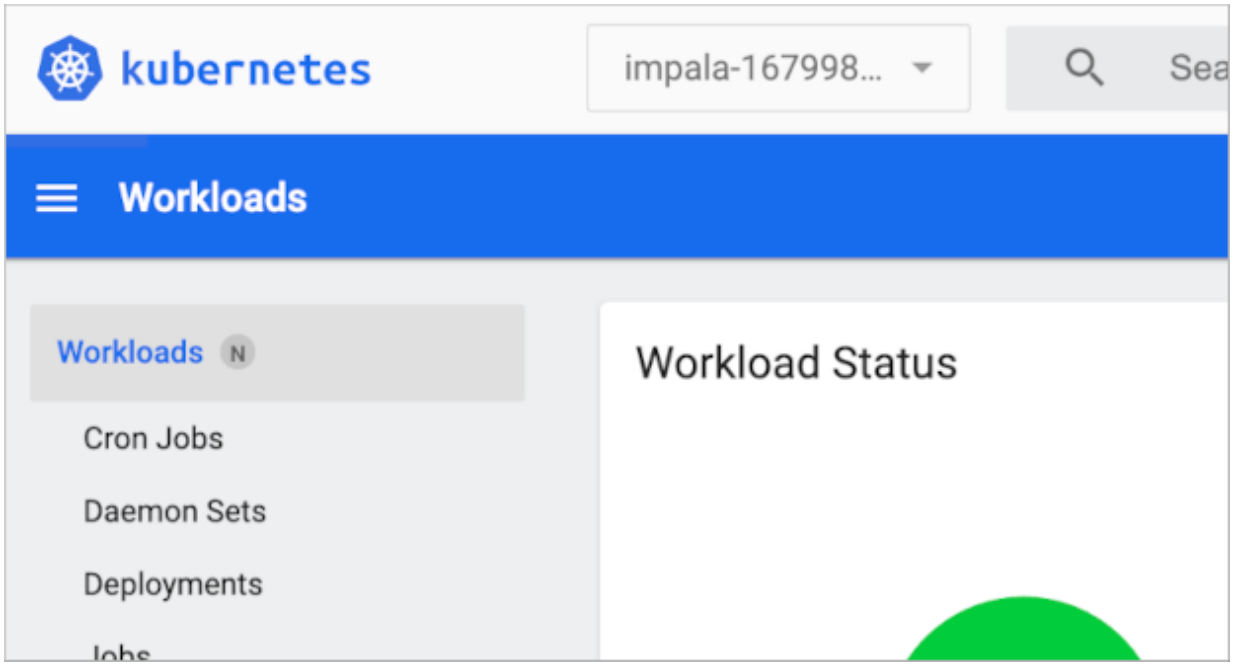

5. Kubernetes dashboard features are available and documented on the kubernetes site.

6. To download logs, click Pods.

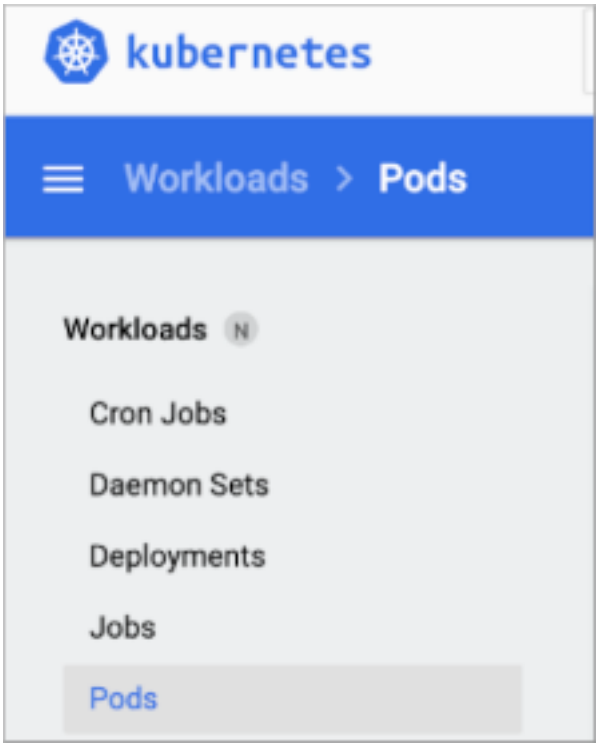

The dashboard of pods appears.

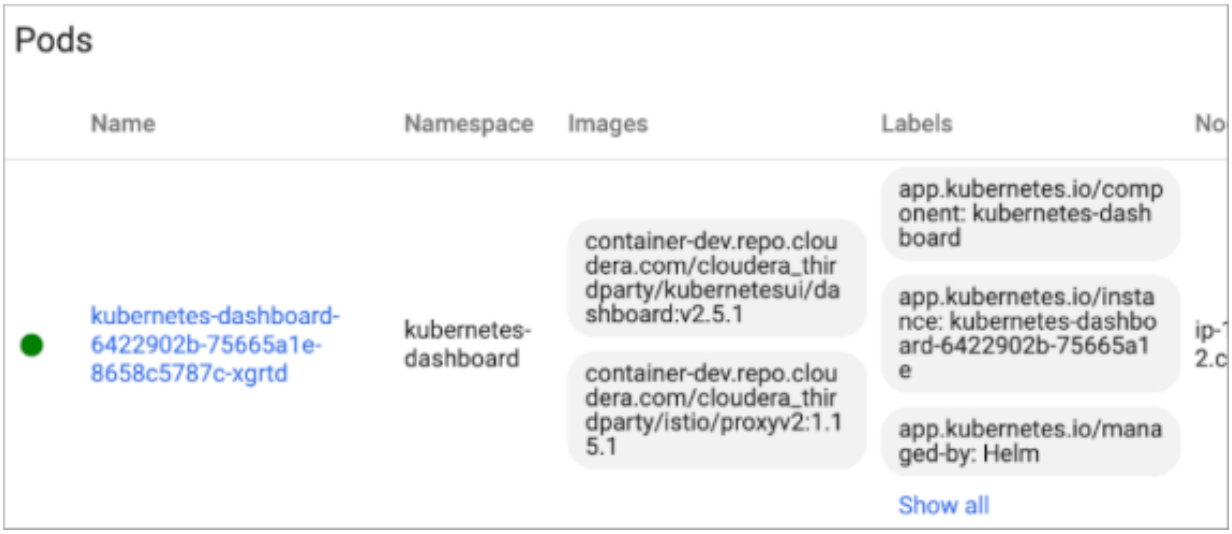

7.
Click ⋮, and select Logs.
For example:

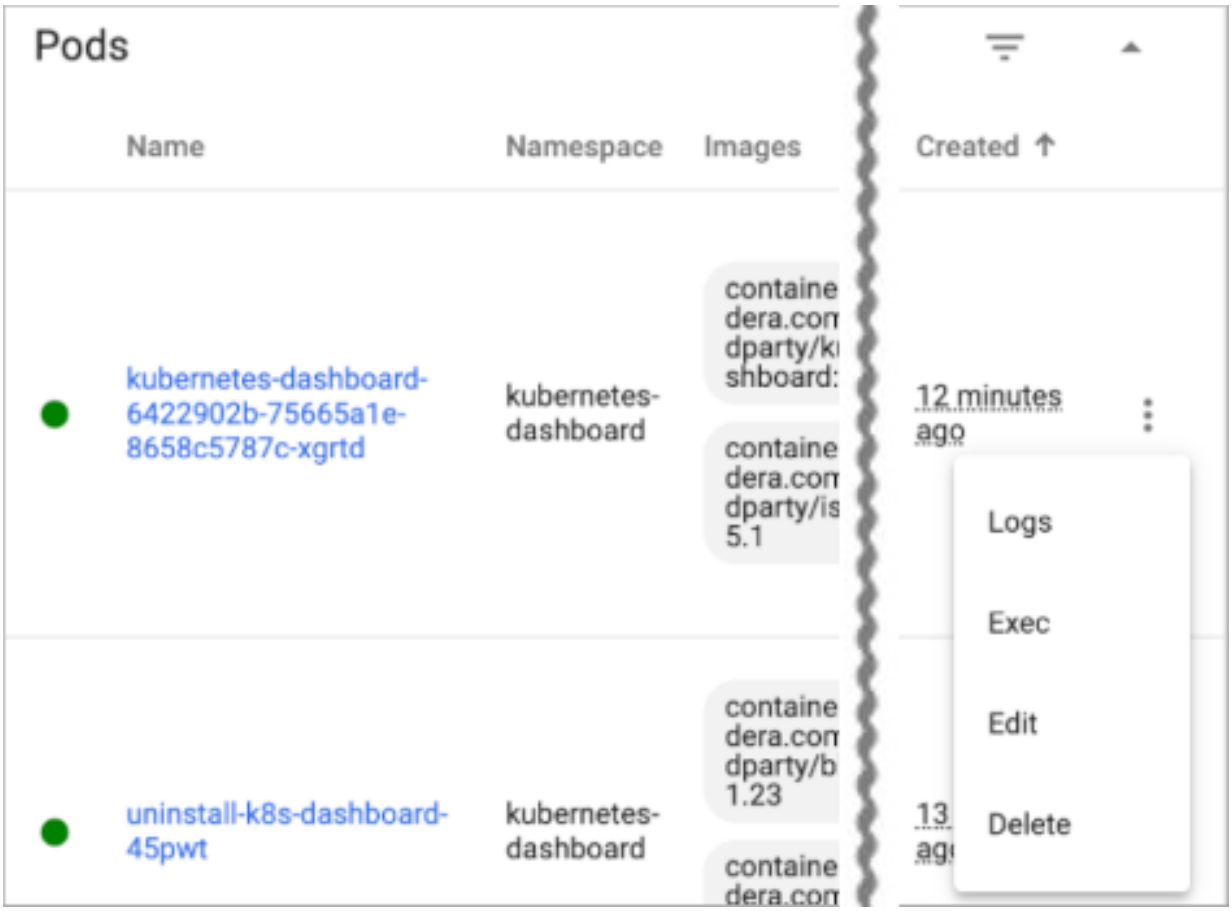

Logs appear.

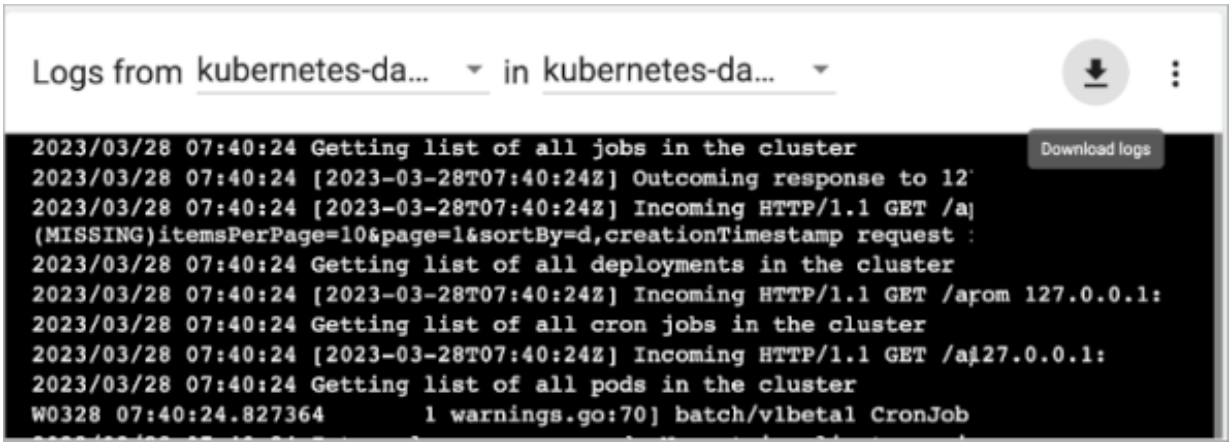

8.
Click Download logs .

Forwarding logs to your observability system

You can forward logs from environments activated in Cloudera Data Warehouse to observability and monitoring systems such as Datadog, New Relic, or Splunk. You learn how to configure a Cloudera Data Warehouse environment for these systems.

About this task

After configuring log forwarding as described in this task, logs flow from Cloudera Data Warehouse to your system automatically. You enjoy the convenience of sorting, searching, and viewing logs on your own system instead of grepping logs from diagnostic bundles on S3 or ABFS. In addition to configuring the log forwarding, you configure removal of debug logs and text strings from the logs. You can configure log forwarding to one of the following observability systems:

- Datadog — https://github.com/DataDog/fluent-plugin-datadog
- Honeycomb.io — https://docs.honeycomb.io/getting-data-in/logs/log-collectors/fluentd/
- New Relic — https://github.com/newrelic/newrelic-fluentd-output
- Splunk — https://github.com/splunk/fluent-plugin-splunk-hec (covers both Splunk-HEC and Splunk-SCS)

You create the log forwarding configuration in valid fluentd format. The configuration is inserted into a larger fluentd configuration. All fluentd events are copied and relabeled with the new label @cloudera_cdw. Your custom configuration is then inserted between <label> tags:

```
<label @cloudera_cdw>
```

customer config goes here

```
</label>
```

You can use any of the built-in fluentd filter, formatter, parser, or output plugins to build the custom config.

Before you begin

- Before configuring log forwarding you must activate an AWS environment or activate an Azure environment in Cloudera Data Warehouse.
- You must be familiar with fluentd and accept the responsibility of configuring log forwarding to your observability systems.

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate your environment, and click ⋮ Edit Observability .

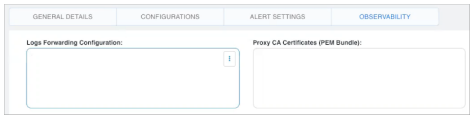

3. Decide how you want to create the fluentd config.
   - Write your own fluentd config from the ground up.
   - Use a Cloudera-provided snippet as a template to write your fluentd config.

4. In Log Forwarding Configuration, click ⋮ .

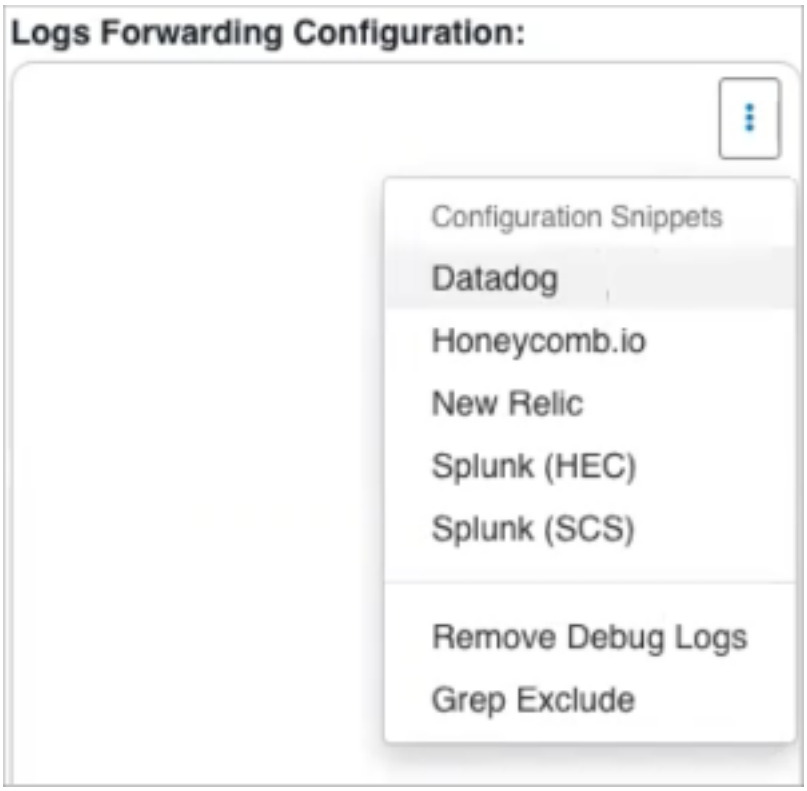

5. Select one of the systems, such as Datadog, to configure.
A fluentd snippet appears. For example, the Datadog snippet appears:

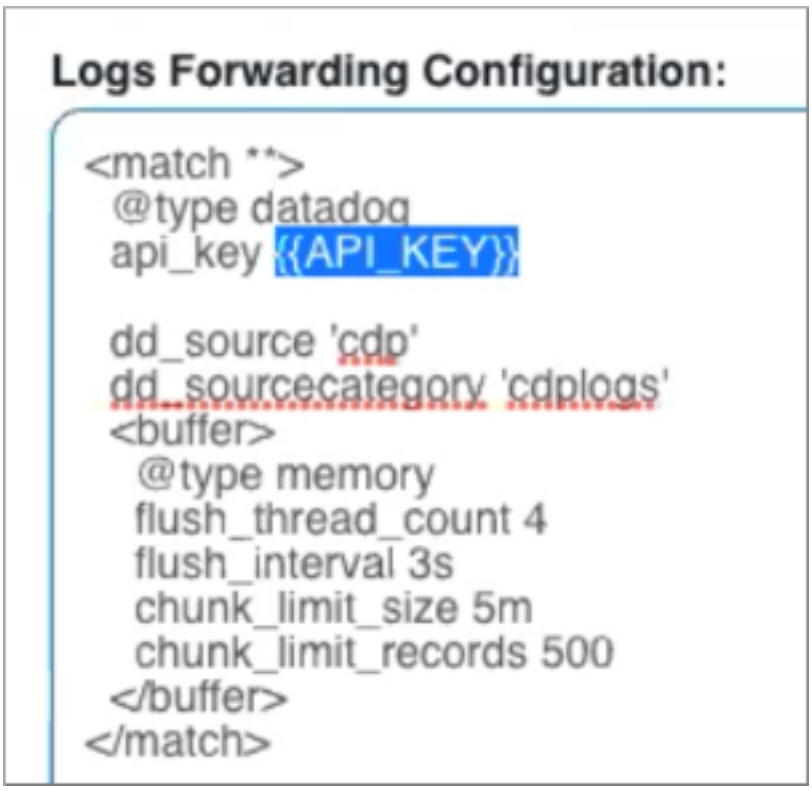

6. Replace the snippet with the fluentd config you wrote from the ground up, or customize the provided snippet. For example, to customize the provided snippet replace the placeholder {{API Key}} with the actual key.

Logs Forwarding Configuration:

```
<match **>
  @type datadog
  api_key 5674465

  dd_source 'cdp'
  dd_sourcecategory 'cdplogs'
  <buffer>
    @type memory
    flush_thread_count 4
    flush_interval 3s
    chunk_limit_size 5m
    chunk_limit_records 500
  </buffer>
</match>
```

7. (Optional) If debug level log messages are not desired, add a fluentd filter to remove them: In the environment, click ⋮, and select Remove Debug Logs.
The fluentd snippet appears for removing debug logs. For example:

Logs Forwarding Configuration:

```
<filter **>
  @type grep
  <exclude>
    key log
    pattern /debug/
  </exclude>
</filter>

<match **>
  @type datadog
  api_key 5674465

  dd_source 'cdp'
  dd_sourcecategory 'cdplogs'
  <buffer>
    @type memory
    flush_thread_count 4
    flush_interval 3s
    chunk_limit_size 5m
    chunk_limit_records 500
  </buffer>
</match>
```

No user customization is necessary to remove debug logs.

8. (Optional) If certain log messages do not provide value for you, remove them with a fluentd grep exclude filter:

   In the environment, click ⋮, select Grep Exclude, and replace {{PATTERN}} with the grep expression that matches the phrase you want to exclude.

   For more information about using Grep Exclude, see https://docs.fluentd.org/filter/grep.

9. If you use a proxy server for outbound traffic, provide the proxy server's CA certificates PEM bundle as described in the next task.

10. Click Apply Changes.
Cloudera Data Warehouse tests the log forwarding configuration and proxy CA certificates bundle, and saves the configuration if both are valid. An invalid log forwarding config error message appears in the event of a configuration problem. For example:

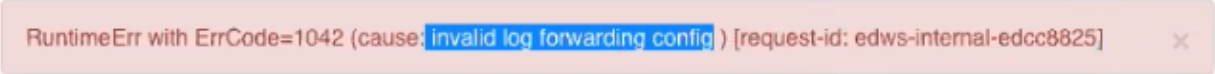

If your configuration is valid, Cloudera Data Warehouse initiates a restart of fluentd to apply the updated config. You see the following indicators of success:

- The environment Running indicator changes, blinks Updating, and then once again says Running.
- You see logs appearing in your observability system.

  Many factors affect how long it takes for forwarding to begin, but generally, the bigger your Cloudera Data Warehouse environment, the longer it takes.

Providing proxy CA certificates

If you use a TLS-terminating proxy server to inspect outbound internet traffic, you need to provide the proxy server's CA certificates bundle in PEM bundle format when you configure log forwarding.

About this task

You learn how to use the Observability tab in Cloudera Data Warehouse Environment Details to configure the Proxy CA Certificates (PEM Bundle) field.

Before you begin

Before you apply the proxy CA certificate to a configuration of log forwarding, you must provide a configuration in the Logs Forwarding Configuration section of the Observability tab.

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.

2. Locate your environment and click ⋮ Edit Observability .
Environment details include a UI for configuring log forwarding.

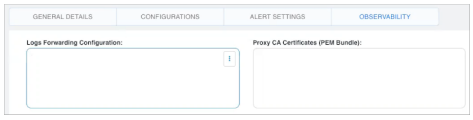

3. Obtain and copy your proxy server's CA certificates PEM bundle.

4. In Proxy CA Certificates (PEM Bundle), paste the copy of the PEM bundle.

5. Click Apply Changes.

If the certificate and log forwarding configuration are valid, log forwarding begins. If the certificates are invalid, an error message occurs.

The log forwarding configuration and certificates are not saved.

Cloudera Data Warehouse Public Cloud

Cloudera Data Warehouse Overview

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Service overview 4

Service architecture 4

Key features 7

About SQL AI Assistant 8

Cloudera Data Warehouse - Service overview and components

Cloudera Data Warehouse Data Service is a containerized application for creating highly performant, independent, self-service data warehouses in the cloud which can be scaled dynamically and upgraded independently. Learn more about the service architecture, and how Cloudera Data Warehouse enables data practitioners and IT administrators to achieve their goals.

Cloudera enables you to implement powerful modern data architectures such as Data Mesh, Data Fabric, and Data Lakehouse. Cloudera supports a Data Lakehouse architecture by pre-integrating and unifying the capabilities of Data Warehouses and Data Lakes, to support data engineering, business intelligence, and machine learning – all on a single platform. Cloudera's support for an open data lakehouse, centered on Cloudera Data Warehouse, brings high-performance, self-service reporting and analytics to your business – simplifying data management for both for data practitioners and administrators.

Cloudera Data Warehouse provides tight integration with the other Cloudera Data Services providing data ingestion, data engineering, machine learning, and data visualization.

Cloudera Data Warehouse leverages Apache Iceberg, Apache Impala, Hive ACID, and Hive table format to provide broad coverage, enabling the best optimized set of capabilities to each workload.

For data practitioners, Cloudera Data Warehouse provides consistent quick response times with high concurrency, easy data exploration and business intelligence on any enterprise data across the Data Lake. Cloudera Data Warehouse also supports streamlined application development with open standards, open file and table formats, and standard APIs. Through tenant isolation, Cloudera Data Warehouse can process workloads that do not interfere with each other, so everyone meets report timelines while controlling costs.

For administrators, Cloudera Data Warehouse simplifies administration by making multi-tenancy secure and manageable and reduces cloud costs. Virtual Warehouses can be provisioned on-demand, using self-service, and de-provisioned when idle. Administrators also benefit from the ability to independently upgrade the Virtual Warehouses and Database Catalogs. Additionally, Cloudera Data Warehouse allows the choice of the version for the Hive or Impala Virtual Warehouse, and Hue you want to use.

Related Information

Adding a new Virtual Warehouse

Cloudera identity management

Cloudera Data Warehouse service architecture

Administrators and IT teams can get a high-level view of the Cloudera Data Warehouse service components and how they are integrated within the Cloudera stack.

The Cloudera Data Warehouse service is composed of the Cloudera Data Warehouse Control Plane (CRUD application), Database Catalogs (storage prepared for use with a Virtual Warehouse) and Virtual Warehouses (compute environments that can access a Database Catalog) and they are decoupled by design. Multiple Virtual Warehouses of differing sizes and types can be configured to operate on the same Database Catalog, providing workload diversity and isolation on the same data at the same time. Cloudera Data Warehouse is also integrated with Cloudera Data Visualization for visualizing data and obtaining insights.

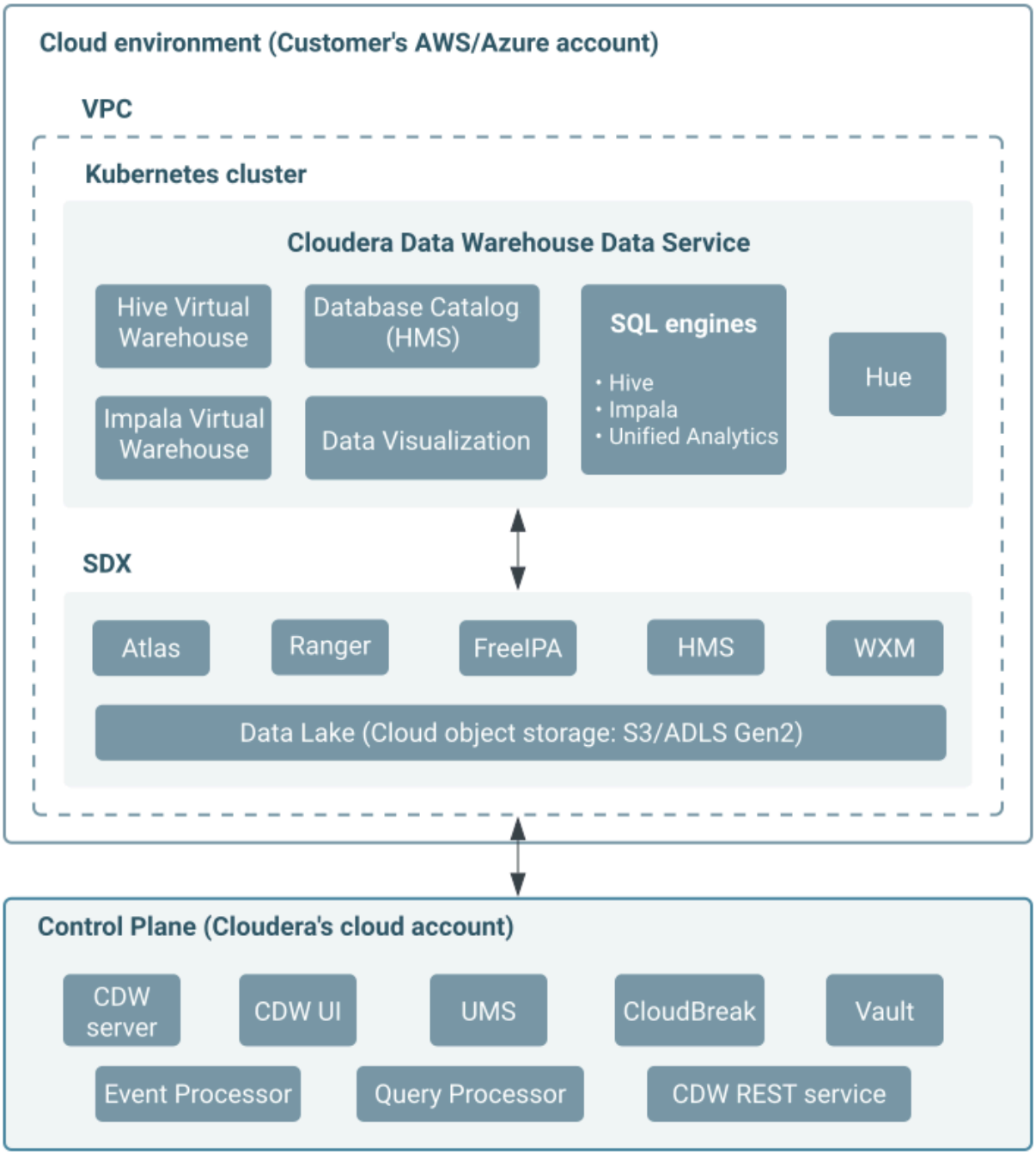

Cloudera Data Warehouse Control Plane

The Cloudera Data Warehouse Control Plane is a CRUD application that serves every operation that you can take on the Cloudera Data Warehouse UI or by using the Cloudera CLI. For example, editing a Cloudera Data Warehouse environment, deactivating the environment, upgrading a Virtual Warehouse, suspending (stopping), and resuming (starting or restarting) Cloudera Data Warehouse entities. The Cloudera Data Warehouse Control Plane version is displayed on the bottom left corner of the UI and automatically reflects the latest version when Cloudera publishes a release. For example 1.9.2-b657.

Cloudera Data Warehouse Environment

A Cloudera Data Warehouse Environment is a cluster that is deployed on your cloud infrastructure (AWS or Azure). After you register your account with Cloudera Management Console, the environment is made available in the Cloudera Data Warehouse service. You must then activate your environment in the Cloudera Data Warehouse service. The Cloudera Data Warehouse environment inherits the version of the Cloudera Data Warehouse Control Plane when it is created.

The Environment version is displayed on the **Environment Details** page on the Cloudera Data Warehouse UI. For example, 1.9.2-b657. The Cloudera Data Warehouse Environment version is not automatically upgraded. You must deactivate and reactivate the environment using the backup and restore process to upgrade to the latest version. Reactivating an environment also updates the Kubernetes (AKS or EKS) version.

Database Catalog

A Database Catalog is a logical collection of table and view metadata, security permissions, and other information. Behind each Database Catalog is a Hive metastore (HMS) that collects your definitions about data in cloud storage. An object store in a secure data lake contains all the actual data for your environment. A Database Catalog includes transient user and workload contexts from the Virtual Warehouse and governance artifacts that support functions such as auditing. Multiple Virtual Warehouses can query a Database Catalog. An environment can have multiple Database Catalogs.

When you activate an environment from the Cloudera Data Warehouse, a Database Catalog is created automatically and named after your environment. The environment shares a default HMS with services, such as Cloudera Data Engineering, Cloudera Data Warehouse, Cloudera AI to some extent, and Cloudera Data Hub templates, such as Data Mart. Consequently, the same objects and data sets are accessible from Cloudera Data Warehouse or any Cloudera Data Hub created in the environment by virtue of using the same HMS. Queries and query history saved in the Hue database are stored in the Database Catalog and not deleted when you delete a Virtual Warehouse.

The Database Catalog version is displayed on the Database Catalog Details page. For example, 2024.0.18.1-1. This is the Cloudera Runtime version of the HMS. The Database Catalogs are automatically upgraded when you deactivate and reactivate your Cloudera Data Warehouse Environment. You can also upgrade the Database Catalog independently without reactivating the Environment. Cloudera Data Warehouse displays an Upgrade Now option when a new version is available.

Virtual Warehouses

A Virtual Warehouse is an instance of compute resources running in Kubernetes to execute the queries. From a Virtual Warehouse, you access tables and views of your data in a Database Catalog's Data Lake. Virtual Warehouses bind compute and storage by executing authorized queries on tables and views through the Database Catalog. Virtual Warehouses can scale automatically, and ensure performance even with high concurrency. All JDBC/ODBC compliant tools connect to the virtual warehouse to run queries. Virtual Warehouses also expose HS2-compatible endpoints for CLI tools such as Beeline, Impala-Shell, and Impyla.

The Virtual Warehouse version is displayed on the **Virtual Warehouse Details** page. For example, 2024.0.18.1-1. This is the Cloudera Runtime version of the underlying SQL engines (Hive, Impala) and Iceberg service that is distributed with a particular release. The automatic or semi-automatic backup and restore procedure recreates the Virtual Warehouses with the latest version however this is not the recommended way to upgrade the Virtual Warehouses. You can upgrade the Virtual Warehouses independently without reactivating the Environment. Cloudera Data Warehouse displays an Upgrade Now option when a new version is available.

Note: At times, you might be required to upgrade your Database Catalogs when you upgrade the Virtual Warehouses. A warning indicating this is displayed on the Cloudera Data Warehouse UI.

Data Visualization

In addition to Database Catalogs and Virtual Warehouses that you use to access your data, Cloudera Runtime integrates Data Visualization for building graphic representations of data, dashboards, and visual applications based on Cloudera Runtime data, or other data sources you connect to. You, and authorized users, can explore data across

the entire Cloudera data lifecycle using graphics, such as pie charts and histograms. You can arrange visuals on a dashboard for collaborative analysis.

Key features of Cloudera Data Warehouse Public Cloud

The Cloudera Data Warehouse service provides data warehouses that can be automatically configured and isolated. Cloudera Data Warehouse optimizes existing workloads when you move to the cloud. You scale resources up and down to meet your business demands, and save costs by suspending and resuming resources automatically. Cloudera Data Warehouse comply with your Data Lake security requirements.

Automatically configured and isolated

Each data warehouse can be automatically configured for you by the Cloudera Data Warehouse service, but you can adjust certain settings to suit your needs. Individual warehouses are completely isolated, ensuring that the users have access to only their data and eliminating the problem of "noisy neighbors." *Noisy neighbors* are workloads that monopolize system resources and interfere with the queries from other tenants. With Cloudera Data Warehouse, you can easily offload noisy neighbor workloads to their own Virtual Warehouse instance so other tenants have access to enough compute resources for their workloads to complete and meet their SLAs.

This capability to isolate individual warehouses is equally useful for "VIP workloads." *VIP workloads* are crucial workloads that must have resources to complete immediately and as quickly as possible without waiting in a queue. You can run these VIP workloads in their own warehouse to ensure they get the resources they need to complete as soon as possible.

Optimized for your workloads

Cloudera Data Warehouse are automatically optimized for your workloads. This includes pre-configuring the software and creating the different caching layers, which means that you need not engage in complex capacity planning or tuning. You create a Virtual Warehouse that specifies a SQL engine:

- Hive for Cloudera that support complex reports and enterprise dashboards.
- Impala for Data Marts that support interactive, ad-hoc analysis.

You choose the Virtual Warehouse instance size and adjust thresholds for auto-scaling.

Auto-scaling

Auto-scaling enables both scaling up and scaling down of Virtual Warehouse instances so they can meet your varying workload demands and save costs on cloud resources when they are not needed.

Auto-scaling provides the following benefits:

- Service availability: Clusters are ready to accept queries “24 x 7”.
- Auto-scaling based on query wait-time: Queries start executing within the number of seconds that you specify and cluster resources are added or shut down to meet demand.
- Auto-scaling based on number of concurrent queries running on the system: "Infinite scaling" means that the number of concurrent queries can go from 10 to 100 in minutes.
- Cost guarantee: You can configure auto-scaling upper limits, which determine how large a compute cluster can grow. Since compute costs increase as cluster size increases, having a way to configure upper limits gives administrators a method to stay within a budget.

Auto-suspend and resume

You have the capability to set an AutoSuspend Timeout when creating a Virtual Warehouse. This sets the maximum time a Virtual Warehouse idles before shutting down. For example, if you set this to 60 seconds, then if the Virtual Warehouse is idle for 60 seconds, it suspends itself so you do not have to pay for unused compute resources. The first

time a new query is run against an auto-suspend Virtual Warehouse, it restarts. This feature helps you maintain a tight control on your cloud spend while ensuring availability to run your workloads.

Security compliance

Your Database Catalogs and Virtual Warehouses automatically inherit the same security restrictions that are applicable to your Cloudera environment. There is no need to specify the security setup again for each Database Catalog or Virtual Warehouse. For more information, see Cloudera identity management. It discusses integration with Apache Knox and your LDAP provider which uses FreeIPA Identity Management.

The following security controls are inherited from your Cloudera environment:

- Authentication: Ensures that all users have proven their identity before accessing the Cloudera Data Warehouse service or any created Database Catalogs or Virtual Warehouses.
- Authorization: Ensures that only users who have been granted adequate permissions are able to access the Cloudera Data Warehouse service and the data stored in the tables.
- Dynamic column masking: If rules are set up to mask certain columns when queries run, based on the user executing the query, then these rules also apply to queries executed in the Virtual Warehouses.
- Row-level filtering: If rules are set up to filter certain rows from being returned in the query results, based on the user executing the query, then these same rules also apply to queries executed in the Virtual Warehouses.

Tenant isolation

The multitenant storage technique in Cloudera Data Warehouse offers increased security over the storage method used in earlier releases. The earlier releases based all storage access in Cloudera Data Warehouse on a single EC2 instance role. Tenant isolation offers users independence on several levels:

- Isolation reduces contention
- Workloads do not interfere with each other
- Each tenant can choose a version of an independent deployment
- Independent upgrades limit the scope of changes

Related Information

Cloudera identity management

About the Hue SQL AI Assistant

Learn about the AI models and services that Hue uses to run the SQL AI Assistant and its limitations. Review what data is shared with the LLM models before you start using the SQL AI Assistant with Hue.

A SQL AI Assistant has been integrated into Hue with the capability to leverage the power of Large Language Models (LLMs) for various SQL tasks. It helps you to create, edit, optimize, fix, and succinctly summarize queries using natural language and makes SQL development faster, easier, and less error-prone. SQL AI Assistant is available with the Hue image version 2023.0.16.0 and higher on Cloudera Public Cloud. Both Hive and Impala dialects are supported.

AI models and services that Hue uses

The SQL AI Assistant supports various LLMs and hosting services. The models run on cloud infrastructure, and the AI Assistant can be configured to use them remotely. Cloudera has tested with GPT running in Open AI, Microsoft Azure, and Amazon Bedrock. The following service-model combinations are supported:

Service Provider	Model	Model Versions
OpenAI	OpenAI GPT	• gpt-3.5-turbo • gpt-3.5-turbo-16k Current GPT version is 3.5 turbo. You can configure GPT 4 for better results.
Microsoft Azure	OpenAI GPT	• gpt-3.5-turbo • gpt-3.5-turbo-16k
Amazon Bedrock	Anthropic Claude	• anthropic.claude-v1 • anthropic.claude-v2
Amazon Bedrock	Amazon Titan	• amazon.titan-text-express-v1

Note:

You must have access to the Hugging Face to download the required sentence transformer model, and ensure your system can connect to the internet, specifically, huggingface.co. Since these models aren't pre-bundled, they must be downloaded during setup. For more information, see Hugging Face

For better results, Cloudera recommends you to use the SQL AI Assistant with the Azure OpenAI service. This ensures that the models run in your VirtualPrivate Cloud network.

The SQL AI Assistant uses a Retrieval Augmented Generation (RAG)-based architecture for augmenting results. It uses the sentence-transformer library for semantic search, and Hue can be configured with any of the pre-trained models for better multi-lingual support. By default, "all-MiniLM-L6-v2" models are used.

Embedding Model	Language Support
all-MiniLM-L6-v2	English
distiluse-base-multilingual-cased-v1	Arabic, Chinese, Dutch, English, French, German, Italian, Korean, Polish, Portuguese, Russian, Spanish, and Turkish.

What data is shared with the LLM models

The following details are shared with the LLMs:

- Everything that a user inputs
- Dialect in use
- Table details such as table name, column names, column data types and related keys, partitions, and constraints that the logged-in user has access to.
- Three sample rows from the tables (as per the best practices specified in *Evaluating the Text-to-SQL Capabilities of Large Language Models*)

Limitations

Non-deterministic nature

LLMs are non-deterministic, which means you cannot guarantee the same output for the same input every time, and it can lead to different responses to similar queries.

Ambiguity

LLMs may struggle to handle ambiguous queries or contexts. SQL queries often rely on specific and unambiguous language, but LLMs can misinterpret or generate ambiguous SQL queries, leading to incorrect results.

Hallucinations

In the context of LLMs, hallucination refers to a phenomenon where these models generate text or responses that are incorrect, nonsensical, or fabricated. Occasionally you might see incorrect identifiers or literals in the response.

Related Information

About setting up the SQL AI Assistant in Cloudera Data Warehouse
Starting the SQL AI Assistant in Hue

Cloudera Data Warehouse Public Cloud

Resource Planning

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Virtual Warehouse sizing requirements for Public Cloud environments.....4

Virtual Warehouse IP address and cloud resource requirements for Public Cloud environments.....7

IP address and cloud resource requirements for Virtual Warehouses running on AWS environments..... 7

IP address and cloud resource requirements for Virtual Warehouses running on Azure environments..... 9

Managing costs in the Public Cloud environments for Cloudera Data Warehouse..... 11

Virtual Warehouse sizing requirements for Public Cloud environments

You need to understand how to estimate size requirements for Cloudera Data Warehouse Public Cloud Virtual Warehouses to create a Virtual Warehouse that meets your needs.

Calculating Public Cloud requirements for on-premises data warehouse deployments

Selecting the correct size of Public Cloud environment before you migrate your workloads from CDH and HDP to Cloudera Data Warehouse Public Cloud is critical for preserving performance characteristics. Consider the following workload characteristics when you plan for capacity on your Public Cloud environment:

- Query memory requirements
- CPU utilizations
- Disk bandwidth
- Working set size
- Concurrent query execution requirements

As part of this calculation, it is important to understand the core hardware difference between Public Cloud and on-premises hosts as explained in the following table:

Table 1: Hardware differences between Public Cloud and on-premises hosts

Hardware component	On-premises	AWS R5D.4xlarge instance	Azure E16 v3 instance
CPU cores	20 - 80	16	16
Memory	128 GB minimum 256 GB+ recommended	128 GB	128 GB
Network	10 Gbps minimum 40 Gbps recommended	Up to 10 Gbps	Up to 8,000 Mbps
Instance storage	12 x 2 TB drives (1,000 MB/s sequential)	2 x 300 GB NVMe SSD (1,100 MB/s sequential) **Note:** Optimally EBS volumes can be added to scratch and spool space for Impala.	400 GiB SSD
Persistent storage performance	At least 500 MB/s per disk With 20 disks, 10 GB/s per node	1,156 MB/s per EC2 instance	Maximum IOPS: 24,000 Maximum Read: 375 MB/s Maximum Write: 187 MB/s

An AWS R5D.4xlarge instance closely matches the CPU, memory, and bandwidth specifications that are recommended for CDH clusters. AWS R5D.4xlarge instance specifications are the default "instance type" for Cloudera. AWS EBS storage cannot be used as primary database storage because it is transient and lacks sufficient capacity. This core difference makes it necessary to use a different strategy for Cloudera than you used for CDH to achieve good scan performance.

Supported AWS instances

You can choose an instance type other than the default AWS R5D.4xlarge , and also a fallback instance type, using the UI or Cloudera CLI. Select the instance type using the UI when you activate the environment in Cloudera. To fetch a string of available instance types, use the following CLI command:

```
cdp dw describe-allowed-instance-types
```

Example output:

```
{
     "aws": {
     "default": [
     "r5d.4xlarge"
     ],
     "allowed": [
     "r5d.4xlarge",
     ...
```

For more information, see the DWX-1.5.1 November 22, 2022 release notes.

Public Cloud sizing and scaling

Before you migrate data to Cloudera Data Warehouse Public Cloud, plan for scaling and concurrency. In the cloud, scaling and concurrency can elastically respond to workload demands, which enables the system to operate at a lower cost than you might expect. If you configure your environment to accommodate peak workloads as a constant default configuration, you may waste resources and money when system demand falls below that level.

In Cloudera Data Warehouse, the size of the Virtual Warehouse determines the number of executor instances for an individual cluster so it determines memory limits and performance capabilities of individual queries:

Table 2: Virtual Warehouse sizes

Size	Number of executors
X-Small	2
Small	10
Medium	20
Large	40

Warehouse size in combination with auto-scaling settings determine how many clusters are allocated to support concurrent query execution.

The Virtual Warehouse size must be at least large enough to support the memory used by the most data-intensive query. Usually, the Virtual Warehouse size does not need to be larger that the most data-intensive query. Better caching is provided if there is commonality between data sets accessed by queries. Increasing the Virtual Warehouse size can increase single-user and multi-user capacity because additional memory and resources allows larger datasets to be processed. Concurrent query execution is also supported by sharing resources. If too small a size is configured for the Virtual Warehouse, poor data caching and memory paging can result. If too large a size is configured, excessive public cloud costs are incurred due to idle executors.

The primary difference between Cloudera Data Warehouse Public Cloud and CDH on-premises deployments when choosing a warehouse size based on existing hardware:

- With Cloudera Data Warehouse Public Cloud all resources on an executor are dedicated to query processing.
- With CDH on-premises deployments resources support other operations in addition to query processing. For example, these on-premises resources are shared with other services, such as HDFS or other locally hosted file systems. In particular Spark, HBase, or MapReduce. These other services might consume significant resources.

Consequently, you might be able to choose a much smaller Virtual Warehouse size in Cloudera Data Warehouse Public Cloud because resources are isolated in their own pod in the Cloudera Data WarehousePublic Cloud environment.

In the case of Impala, it is useful to look at the Cloudera Manager per-process metrics to isolate the impalad backend and the Impala front-end Java processes that hold the Catalog cache. In Cloudera Data Warehouse Public Cloud, the Impala coordinator and executor roles are separated leaving the unused Catalog JVM memory free to support query execution. You should look at the memory utilization metrics for executor-only impalad nodes (those not also running the coordinator role) to estimate how much memory your current cluster of Impala executors requires.

Concurrency

Concurrency is the number of queries that can be run at the same time. Determine the size you need by considering the amount of resources your system needs to support peak concurrency.

By default, Impala Virtual Warehouses can run 3 large queries per executor group. Executors can handle more queries that are simpler and that do not utilize concurrency on the executor. When you enable legacy multithreading, the Virtual Warehouse can run 12 queries per executor group. For most read-only queries the default setting of 3 queries per executor group is sufficient. Occasional peaks are handled transparently by the auto-scaling feature. When auto-scaling is triggered an additional executor group is added thereby doubling query concurrency capacity. Scaling the Virtual Warehouse by adding more clusters enables additional concurrent queries to run, but does not improve single-user capacity or performance. Concurrently executed queries are routed to the different clusters and execute independently. The number of clusters can be changed to match concurrent usage by changing the auto-scaling parameters. For more details about auto-scaling settings, see the links at the bottom of this page.

Important: The number of very large queries that can be run might also be impacted by memory limits on the executors.

For Hive on LLAP Virtual Warehouses, each size setting indicates the number of concurrent queries that can be run. For example, an X-Small Hive on LLAP Virtual Warehouse can run 2 concurrent queries. A Small Virtual Warehouse can run 10 concurrent queries. To run 20 concurrent queries in a Hive on LLAP Virtual Warehouse choose Medium size.

Caching "Hot Datasets"

Frequently accessed data is sometimes referred to as a "hot dataset." CDH supports caching mechanisms on the compute nodes to cache the working set that is read from remote file systems, such as remote HDFS data nodes, S3, ABFS, or ADLS. This offsets the input/output performance difference.

In Cloudera Data Warehouse Public Cloud, frequently accessed data is cached in a storage layer on SSD so that it can be quickly retrieved for subsequent queries. This boosts performance. Each executor can have up to 200 GB of cache. For example, a Medium-sized Virtual Warehouse can keep 200 * 20 = 4 TB of data in its cache. For columnar formats, such as ORC, data in the cache is decompressed, but not decoded. If the expected size of the hot dataset is 6 TB, which requires about 30 executors, you can over-provision by choosing a Large-sized warehouse to ensure full cache coverage. A case can also be made to under-provision by choosing a Medium-sized warehouse to reduce costs at the expense of having a lower cache hit rate. To offset this, keep in mind that columnar formats allow optimizations such as column projection and predicate push-down, which can significantly reduce cache requirements. In these cases, under-provisioning the cache might have no negative performance effect.

Scanned dataset size

Scanning large datasets on Amazon S3 or Azure ADLS can be slow. A single R5D.4xlarge EC2 instance can only scan data at 1,156 MB/s maximum throughput according to standard S3 benchmarks. If a query must read 100 GB from S3, S3 scanning takes a minimum of 88 seconds on just one node. Depending on the number of files in the S3 directory, it might take more than 3 minutes. In this case, if a query needs to scan 100 GB of data, if you use 10 nodes, you can get the scan time down to approximately 20 seconds.

However, keep in mind that with columnar storage, minimum/maximum statistics in files, and other conditions, often the amount of data read is significantly less than the total size of the files for certain queries.

Related Information

Tuning Hive Virtual Warehouses on cloud environments

Tuning Impala Virtual Warehouses on cloud environments

Virtual Warehouse IP address and cloud resource requirements for Public Cloud environments

Learn about how many IP addresses and cloud resources are required to run Virtual Warehouses efficiently in Public Cloud environments for Cloudera Data Warehouse.

IP address and cloud resource requirements for Virtual Warehouses running on AWS environments

Learn about the estimated number of IP addresses and cloud resources required to run Virtual Warehouses on AWS environments for Cloudera Data Warehouse Public Cloud.

Important: These requirements are estimated. Your particular workloads and configurations can affect the number of IP addresses and cloud resources required to run Cloudera Data Warehouse Virtual Warehouses efficiently. In addition, these requirements are for AWS environments that use the default AWS VPC Container Networking Interface (CNI) plugin. To reduce the number of required IP addresses, you can enable the overlay networking feature for AWS environments in Cloudera Data Warehouse. For further details, see the link to "Overlay networks for AWS environments in Cloudera Data Warehouse" at the bottom of this page.

Virtual Warehouse requirements:

Each compute node in a Virtual Warehouse that runs on AWS environments requires 8 IP addresses. Each executor needs one compute node, so the size of your Virtual Warehouse contributes to the number of IP addresses required. To calculate the number of IP addresses required for custom sizes, multiply the number of executors by 8 and add for the shared services as specified in the following sections.

Shared services requirements for Database Catalogs:

Virtual Warehouses also require shared services for the Database Catalog.

Additional shared services requirements:

The usage for other shared services is different for Hive Virtual Warehouses and Impala Virtual Warehouses. Here are the different requirements for Hive versus Impala Virtual Warehouses:

- Hive Virtual Warehouses add 1 compute node for each executor and 1 shared services node for the HiveServer for each Virtual Warehouse.
- Impala Virtual Warehouses add 1 compute node for each executor, 1 or 2 compute nodes for the coordinator, depending on the HA configuration, and 1 shared services node per Virtual Warehouse for Impala catalogd.

The following tables summarize the approximate number of IP addresses and cloud resources you should plan for Virtual Warehouses on AWS environments.

Table 3: Hive Virtual Warehouses running on AWS environments

Size (# executors)	# Compute nodes for executors	# Shared services nodes Database Catalog	# Shared services nodes for HiveServer	Total IP addresses required
XSMALL (2)	2	3	1	Executor nodes: 2 nodes X 8 = 16 Shared services nodes: 3 + 1 = 4 nodes X 25 = 100 TOTAL = ~116 IP addresses
SMALL (10)	10	3	1	Executor nodes: 10 nodes X 8 = 80 Shared services nodes: 3 + 1 = 4 nodes X 25 = 100 TOTAL = 180 IP addresses
MED (20)	20	3	1	Executor nodes: 20 nodes X 8 = 160 Shared services nodes: 3 + 1 = 4 nodes X 25 = 100 TOTAL = 260 IP addresses
LARGE (40)	40	3	1	Executor nodes: 40 nodes X 8 = 320 Shared services nodes: 3 + 1 = 4 nodes X 25 = 100 TOTAL = 420 IP addresses

Table 4: Impala Virtual Warehouses running on AWS environments

Size (# executors)	# Compute nodes for executors	# Compute nodes for coordinator	# Shared services nodes for Impala catalogd	# Shared services nodes for Database Catalog	Total IP addresses required
XSMALL (2)	2	1-2	1	3	Executor/ coordinator nodes: 3-4 nodes X 8 = 24-32 Shared services nodes: 1 + 3 = 4 nodes X 25 = 100 TOTAL = 124 to 132 IP addresses
SMALL (10)	10	1-2	1	3	Executor/ coordinator nodes: 11-12 nodes X 8 = 88-96 Shared services nodes: 1 + 3 = 4 nodes X 25 = 100 TOTAL = 188 to 196 IP addresses

Size (# executors)	# Compute nodes for executors	# Compute nodes for coordinator	# Shared services nodes for Impala catalogd	# Shared services nodes for Database Catalog	Total IP addresses required
MED (20)	20	1-2	1	3	Executor/ coordinator nodes: 21-22 nodes X 8 = 168-176 Shared services nodes: 1 + 3 = 4 nodes X 25 = 100 TOTAL = 168 to 176 IP addresses
LARGE (40)	40	1-2	1	3	Executor/ coordinator nodes: 41-42 nodes X 8 = 328-336 Shared services nodes: 1 + 3 = 4 nodes X 25 = 100 TOTAL = 428 to 436 IP addresses

Related Information

Overlay networks for AWS environments in Cloudera Data Warehouse

IP address and cloud resource requirements for Virtual Warehouses running on Azure environments

Learn about the estimated number of IP addresses and cloud resources required to run Virtual Warehouses on Azure environments for Cloudera Data Warehouse Public Cloud.

Important: These requirements are estimated. Your particular workloads and configurations can affect the number of IP addresses and cloud resources required to run Cloudera Data Warehouse Virtual Warehouses efficiently. In addition, these requirements are for Azure environments that use the default Azure Container Networking Interface (CNI) plugin. To reduce the number of required IP addresses, you can enable the kubenet networking feature for Azure environments in Cloudera Data Warehouse.

Virtual Warehouse requirements:

Each compute node in a Virtual Warehouse that runs on Azure environments requires 16 IP addresses (for pods and for the node itself). Each executor needs one compute node, so the size of your Virtual Warehouse contributes to the number of IP addresses required. To calculate the number of IP addresses required for custom sizes, multiply the number of executors by 16 and add for the shared services as specified in the following sections.

Shared services requirements for Database Catalogs:

Virtual Warehouses also require shared services for Database Catalogs.

Additional shared services requirements:

The usage for other shared services is different for Hive Virtual Warehouses and Impala Virtual Warehouses. Here are the different requirements for Hive versus Impala Virtual Warehouses:

- Hive Virtual Warehouses add 1 compute node for each executor and 1 shared services node for the HiveServer for each Virtual Warehouse.
- Impala Virtual Warehouses add 1 compute node for each executor, 1 or 2 compute nodes for the coordinator, depending on the HA configuration, and 1 shared services node per Virtual Warehouse for Impala catalogd.

The following tables summarize the approximate number of IP addresses and cloud resources you should plan for Virtual Warehouses on Azure environments.

Table 5: Hive Virtual Warehouses running on Azure environments

Size (# executors)	# Compute nodes for executors	# Shared services nodes for Database Catalog	# Shared services nodes for HiveServer	Total IP addresses required
XSMALL (2)	2	3	1	Executor nodes: 2 nodes X 16 = 32 Shared services nodes: 3 + 1 = 4 nodes X 31 = 124 TOTAL = 156 IP addresses
SMALL (10)	10	3	1	Executor nodes: 10 nodes X 16 = 160 Shared services nodes: 3 + 1 = 4 nodes X 31 = 124 TOTAL = 284 IP addresses
MED (20)	20	3	1	Executor nodes: 20 nodes X 16 = 320 Shared services nodes: 3 + 1 = 4 nodes X 31 = 124 TOTAL = 444 IP addresses
LARGE (40)	40	3	1	Executor nodes: 40 nodes X 16 = 640 Shared services nodes: 3 + 1 = 4 nodes X 31 = 124 TOTAL = 764 IP addresses

Table 6: Impala Virtual Warehouses running on Azure environments

Size (# executors)	# Compute nodes for executors	# Compute nodes for coordinator	# Shared services nodes for Impala catalogd	# Shared services nodes for Database Catalog	Total IP addresses required
XSMALL (2)	2	1-2	1	3	Executor/ coordinator nodes: 3-4 nodes X 16 = 48-64 Shared services nodes: 1 + 3 = 4 nodes X 31 = 124 TOTAL = 172 to 188 IP addresses
SMALL (10)	10	1-2	1	3	Executor/ coordinator nodes: 11-12 nodes X 16 = 176-192 Shared services nodes: 1 + 3 = 4 nodes X 31 = 124 TOTAL = 300 to 316 IP addresses

Size (# executors)	# Compute nodes for executors	# Compute nodes for coordinator	# Shared services nodes for Impala catalogd	# Shared services nodes for Database Catalog	Total IP addresses required
MED (20)	20	1-2	1	3	Executor/ coordinator nodes: 21-22 nodes X 16 = 336-352 Shared services nodes: 1 + 3 = 4 nodes X 31 = 124 TOTAL = 460 to 476 IP addresses
LARGE (40)	40	1-2	1	3	Executor/ coordinator nodes: 41-42 nodes X 16 = 656-672 Shared services nodes: 1 + 3 = 4 nodes X 31 = 124 TOTAL = 780 to 796 IP addresses

Related Information

Overlay networks for Azure environments in Cloudera Data Warehouse

Managing costs in the Public Cloud environments for Cloudera Data Warehouse

Cost optimization in cloud environments is top priority for enterprises. Cloudera Data Warehouse Public Cloud and your cloud provider give you ways to monitor and control the cloud resources you use.

Setting resource limits with Cloudera Data Warehouse service

Cloudera Data Warehouse service provides the following ways to manage your cloud costs:

- Choose Virtual Warehouse size: Virtual Warehouse size specifies the number of executor nodes used by the Virtual Warehouse, which translates to compute instances. Before you create a Virtual Warehouse, determine the number of concurrent queries or users your Virtual Warehouse must serve during peak periods. This information helps you determine what size of Virtual Warehouse you need. Choose the size based on the number of nodes you typically use for clusters in an on-premises deployment.
- Set auto-scaling thresholds: When you create a Virtual Warehouse, you can define auto-scaling, which sets limits on how many cloud resources can be consumed to meet workload demands. In addition, you can also set the maximum time a Virtual Warehouse idles before shutting down. Both settings ensure that you only use cloud resources that you need when you need them, helping you to manage your costs in the cloud.

Setting resource limits with your cloud provider

Another way to manage your cloud costs is by setting effective resource limits with your cloud provider. Cloud providers offer ways to set the overall limit on numbers of virtual machines or containers that can be used for your account. You can also save on cloud spend by shutting down resources when they are not in use. Cloudera has built-in functionality to shut down cloud resources when not in use. If necessary, you can also terminate resources to save on costs related to disk snapshots, reserved IP addresses, and so on. In addition, cloud providers might offer tools to help you save. For example, AWS offers AWS Trusted Advisor and CloudWatch. Microsoft Azure offers Azure Cost Management and Azure Advisor.

Always active, shared services

Several shared service nodes are always active in a Cloudera Data Warehouse environment. A number of infrastructure pods are required at all times, and sufficient nodes must be deployed for these pods. The following table lists actions and the impact of those actions on shared service nodes.

Action	Impact on Shared Services Nodes	Comments
Activating a Cloudera Data Warehouse environment	Yes	Starts with minimum 3 nodes
Creating Database Catalog	Maybe	It depends on the pod placement logic
Creating Impala/Hive Virtual Warehouse	Yes	This also adds more shared service pods to support compute nodes (weavenet, kube-proxy etc.)
Deleting Impala/Hive Virtual Warehouse	Maybe	It depends on the pod placement logic. “Cooldown” period also plays a role.
Compute autoscaling	Yes	Autoscaling compute nodes will add hue, hs2 pods.

The following scaling factors also affect shared services nodes.

- The number of Virtual Warehouses you run increases the required shared service resources.
- The T-shirt you set for Virtual Warehouses and autoscaling on executors has impact as shown in the following example:

  Hue instance (part of shared service) count = (query-executors/10) + 1

  => from 2 to 10 query executors then 1 Hue server Pod instance,

  => from 10 to 19 query executors then 2 Hue server Pod instances

  => from 20 to 29 query executors then 3 Hue server Pod instances

Related Information

AWS Trusted Advisor
AWS CloudWatch
Azure Cost Management
Azure Advisor
Adding a new Virtual Warehouse
Tuning auto-scaling for Virtual Warehouses

Cloudera Data Warehouse Public Cloud

Querying Data

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Querying data in Cloudera Data Warehouse 4

Submitting queries with Hue 4
- Configure custom properties 5

Creating tables by importing CSV files from AWS S3 in Cloudera Data Warehouse 6

Creating tables by importing CSV files from ABFS 10

View query history 11

Cloudera Data Engineering Airflow ETL workloads in Hive Virtual Warehouses 12

Simplify queries with User-defined functions 12
- Creating a Hive user-defined function 12
  - Setting up the development environment 12
  - Create the UDF class 14
  - Building the project and uploading the JAR 15
  - Configuring UDF JAR caching in Hive Virtual Warehouse 15
  - Registering the UDF 16
- User-defined functions (UDFs) 17
  - UDF concepts 35
  - Runtime environment for UDFs 39
  - Writing UDFs 39
  - Writing user-defined aggregate functions (UDAFs) 42
  - Building and deploying UDFs 43
  - Performance considerations for UDFs 45
  - Examples of creating and using UDFs 45
  - Security considerations for UDFs 51
  - Limitations and restrictions for Impala UDFs 52

Start SQL AI Assistant 52
- Generate SQL from NQL 53
- Edit query in natural language 54
- Explain query in natural language 55
- Optimize SQL query 57
- Fixing a query in Hue 58
- Generate comment for a SQL query 59

Querying data in Cloudera Data Warehouse

This topic describes how to query data in your Virtual Warehouse on Cloudera Data Warehouse.

About this task

The Cloudera Data Warehouse service includes the Hue SQL editor that you can use to submit queries to Virtual Warehouses. For example, you can use Hue to submit queries to an Impala Virtual Warehouse. For detailed information about using Hue, see Using HueUsing Hue.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.

   The **Overview** page is displayed.
2. Click Hue on the Virtual Warehouse tile.
3. Enter your query into the editor and submit it to the Virtual Warehouse.

Related Information

Submitting queries with Hue

Submitting queries with Hue

You can write and edit queries for Hive or Impala Virtual Warehouses in the Cloudera Data Warehouse service by using Hue.

About this task

For detailed information about using Hue, see Using Hue.

Before you begin

Hue uses your LDAP credentials that you have configured for the Cloudera cluster.

Procedure

1. Log into the Cloudera Data Warehouse service as DWUser.
2. Go to the **Virtual Warehouses** tab, locate the Virtual Warehouse using which you want to run queries, and click HUE.

   The Hue query editor opens in a new browser tab.
3. To run a query:
   a) Click a database to view the tables it contains.

      When you click a database, it sets it as the target of your query in the main query editor panel.
   b) Type a query in the editor panel and click to run the query.

      You can also run multiple queries by selecting them and clicking .

      Note: Use the language reference to get information about syntax in addition to the SQL auto-complete feature that is built in. To view the language reference, click the book icon to the right of the query editor panel.

Related Information

Hue overview
Using Hue

Configuring custom properties using safety valves

You can configure Hue properties that are not directly exposed through the Cloudera Data Warehouse web interface by specifying them in the Hue's Advanced Configuration Snippet called hue-safety-valve for a Virtual Warehouse. These configurations are stored in the hue.ini file.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Virtual Warehouses tab, locate the Virtual Warehouse associated with Hue, and click ⋮ Edit .
   The **Virtual Warehouse Details** page is displayed.
3. Go to the Configurations tab, select Hue, and select hue-safety-valve from the Configuration files drop-down menu.

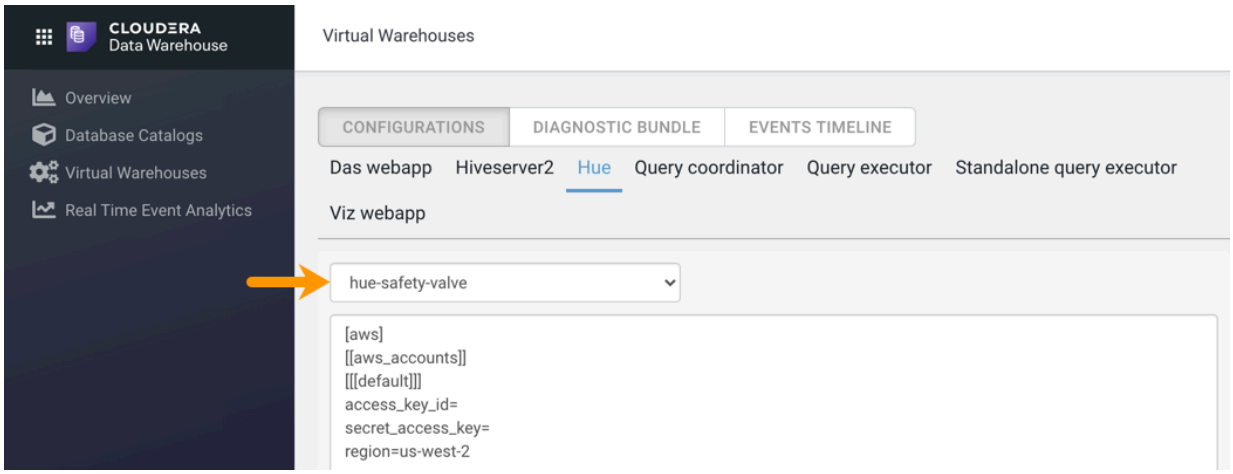

4. Specify the custom configuration properties in the space provided as shown in the following examples:
   For AWS:

```
[aws]
  [[aws_accounts]]
    [[[default]]]
      access_key_id=<access key id>
      secret_access_key=<secret access key>
      region=<aws region>
```

For Azure:

```
[azure]
  [[azure_accounts]]
    [[[default]]]
      client_id=<client id>
      client_secret=<client secret>
      tenant_id=<tenant id>
```

```
[azure]
  [[abfs_clusters]]
     [[[default]]]
```

```
        fs_defaultfs=abfs://<container name>@<storage account>.dfs.core.wi
ndows.net
        webhdfs_url=https://<storage account>.dfs.core.windows.net/
```

5. Click Apply Changes.

Creating tables by importing CSV files from AWS S3 in Cloudera Data Warehouse

You can create tables in Hue by importing CSV files stored in S3 buckets. Hue automatically detects the schema and the column types, thus helping you to create tables without using the CREATE TABLE syntax.

About this task

The maximum file size supported is three gigabytes.

(Non-RAZ deployment) Only Hue Superusers can access S3 buckets and import files to create tables. To create tables by importing files from S3, you must assign and authorize use of a specific bucket on S3 bucket for your environment. The bucket then appears like a home directory on the Hue web interface.

Procedure

1. Log in to the Cloudera Data Warehouse service.
2. On the **Overview** page, select the Virtual Warehouse in which you want to create the table and click on Hue.
3. From the left assist panel, click on Importer.
4. On the Importer screen, click .. at the end of the Path field:

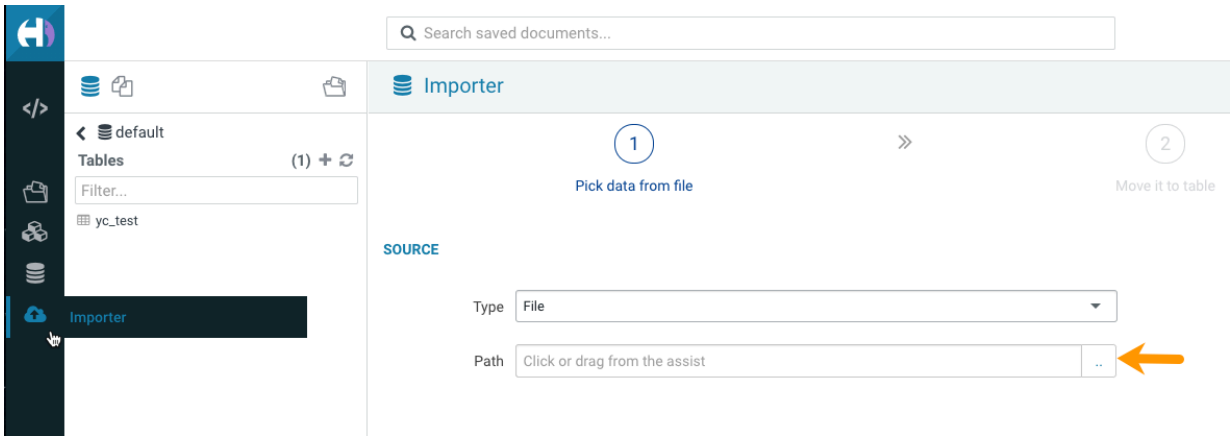

Choose a file pop-up is displayed.

5. (Non-RAZ deployment) Type s3a:// in the address text box and press enter.

   The S3 buckets associated with the Cloudera Data Warehouse environment are displayed. You can narrow down the list of results using the search option.

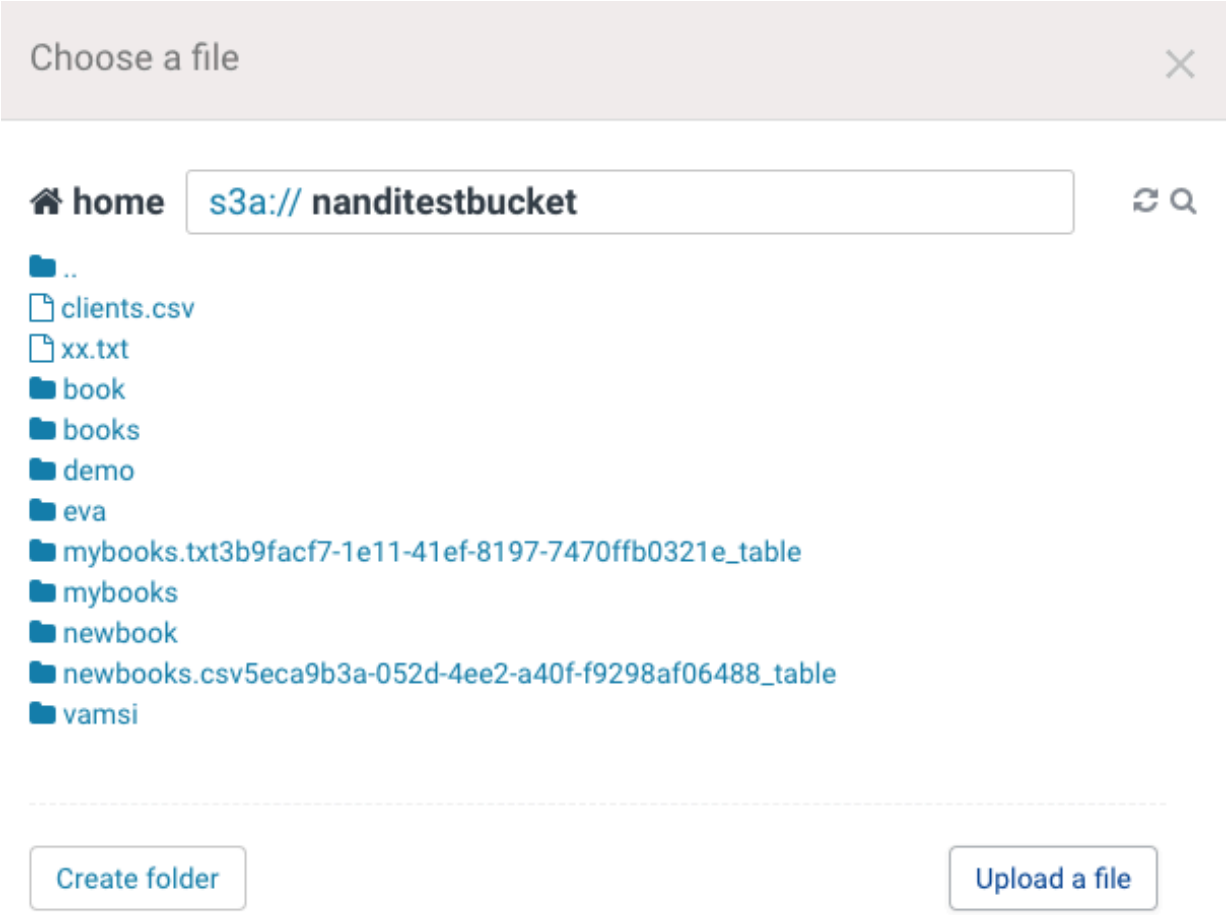

   If the file is present on your computer, then you can upload it to S3 by clicking Upload a file. To do this, you must have enabled read/write access to the S3 bucket from the Cloudera Data Warehouse environment.

6. Select the CSV file that you want to import into Hue.

Hue displays the preview of the table along with the format:

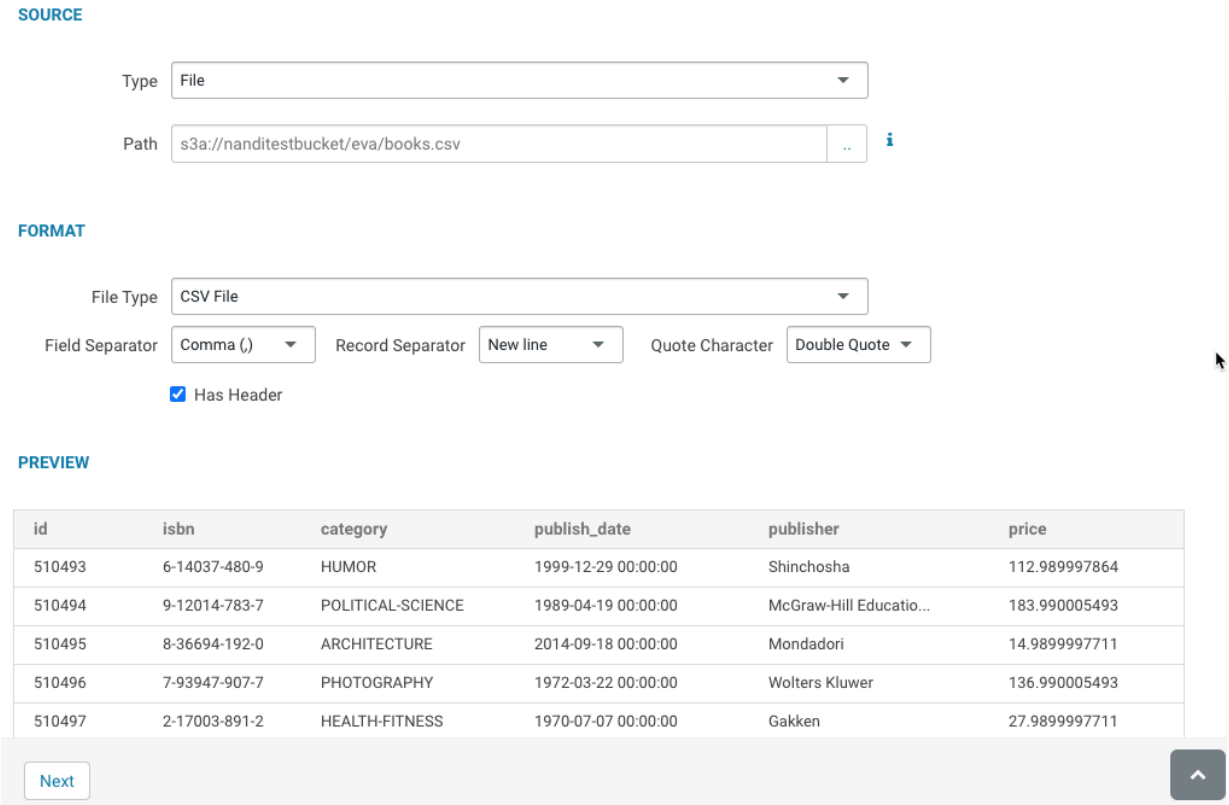

Hue automatically detects the field separator, record separator, and the quote character from the CSV file. If you want to override a specific setting, then you can change it by selecting a different value from the drop-down menu.

7. Click Next.

   On this page, you can set the table destination, partitions, and change the column data types.

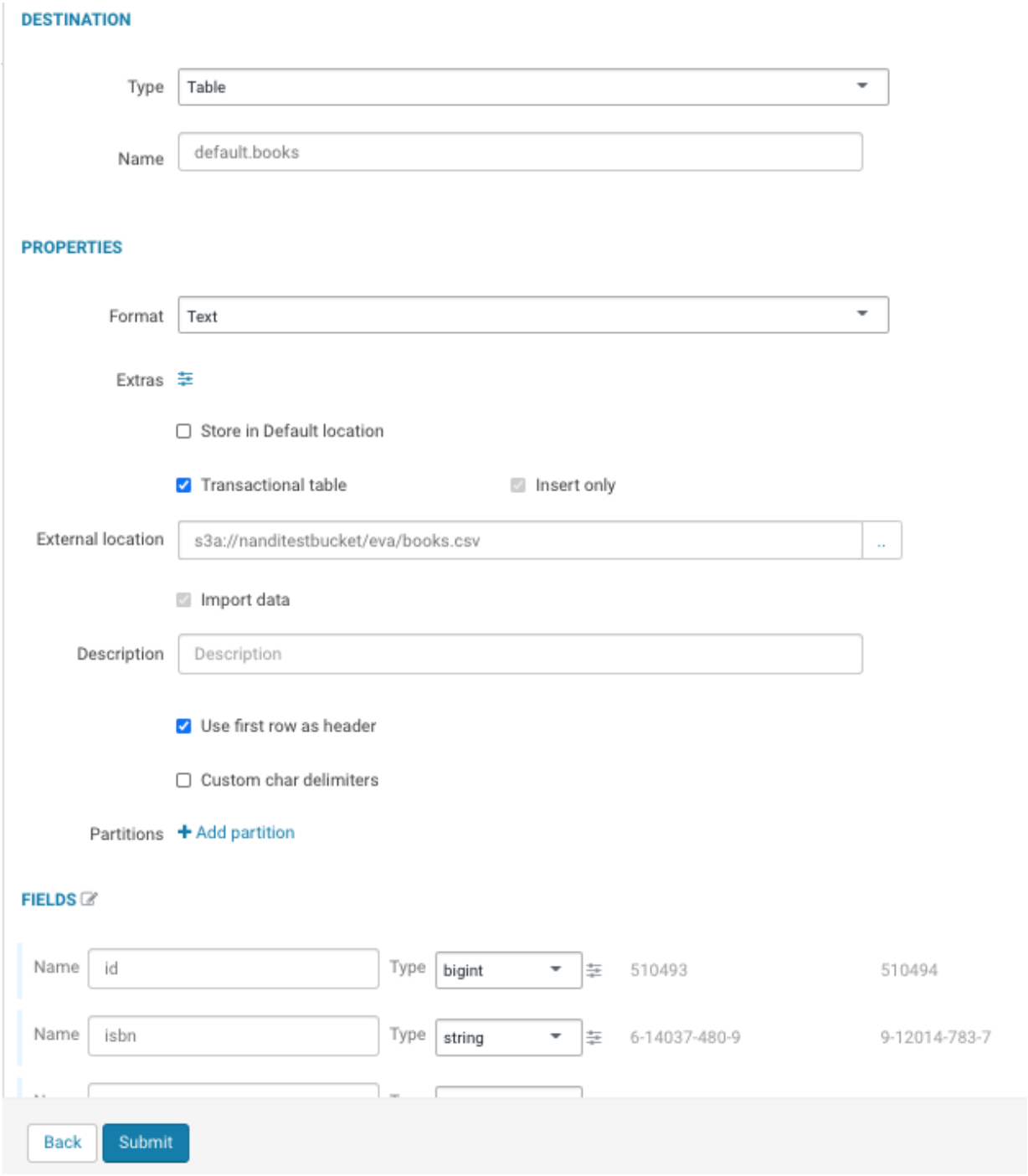

8. Verify the settings and click Submit to create the table.

   The CREATE TABLE query is triggered:

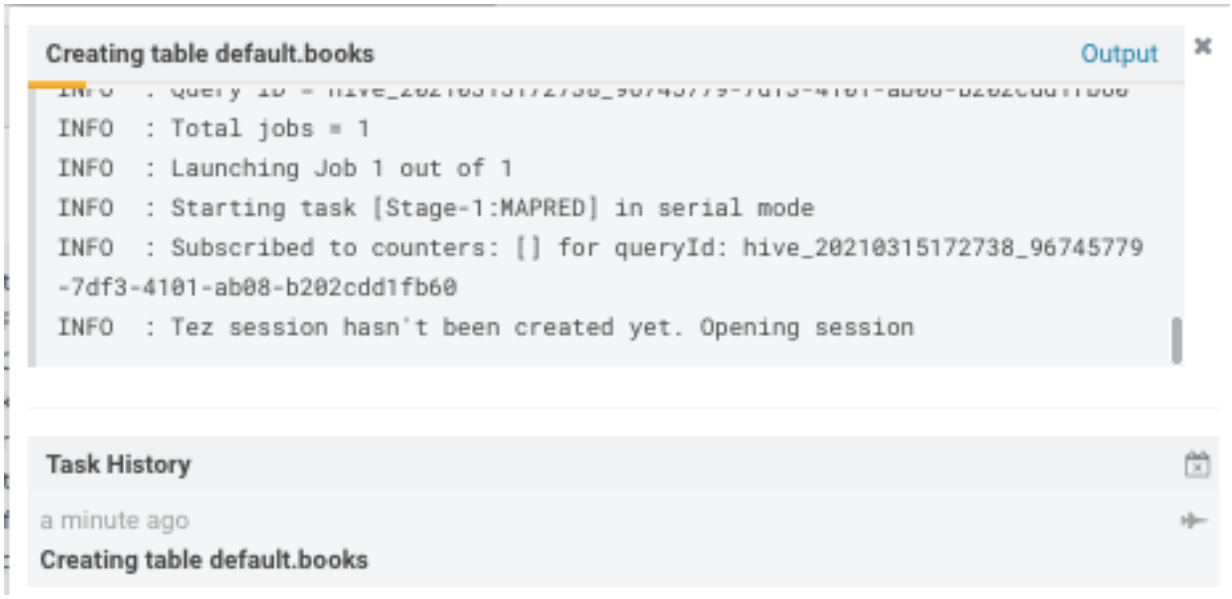

   Hue displays the logs and opens the **Table Browser** from which you can view the newly created table when the operation completes successfully.

Related Information

Assigning resources to users

Performing user sync

Adding access to external S3 buckets for Cloudera Data Warehouse clusters on AWS

Adding Cloudera Data Warehouse cluster access to external S3 buckets in the same AWS account

Creating tables by importing CSV files from ABFS

You can create tables in Hue by importing CSV files stored in ABFS. Hue automatically detects the schema and the column types, thus helping you to create tables without using the CREATE TABLE syntax.

About this task

The maximum file size supported is three gigabytes.

(Non-RAZ deployment) Only Hue Superusers can access ADLS Gen2 containers and import files to create tables. To create tables by importing files from ADLS, you must assign and authorize use of a specific bucket on ADLS Gen2 containers for your environment. The bucket then appears like a home directory on the Hue web interface.

Procedure

1. In the Cloudera Data Warehouse service **Overview** page, select the Virtual Warehouse in which you want to create the table, click the options menu in the upper right corner and click Open Hue.
2. From the left assist panel, click on Importer.

3. On the **Importer** screen, click .. at the end of the Path field:

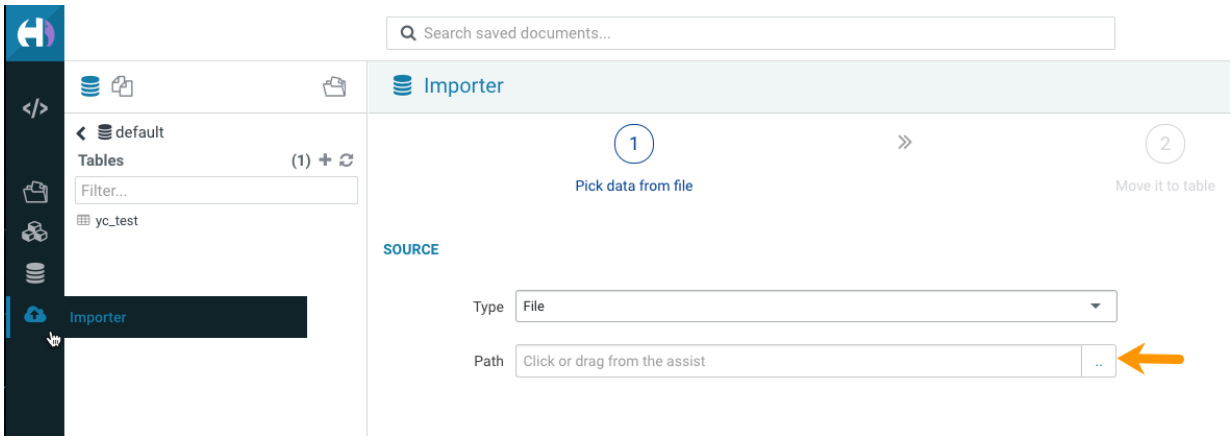

Choose a file pop-up is displayed.

4. (Non-RAZ deployment) Type abfs://[***CONTAINER-NAME***] in the address text box and press enter.

   The ABFS containers created under the Azure storage account are displayed.

   You can narrow down the list of results using the search option.

   If the file is present on your computer, then you can upload it to ABFS by clicking Upload a file.

5. Select the CSV file that you want to import into Hue.

   Hue displays the preview of the table along with the format.

   Hue automatically detects the field separator, record separator, and the quote character from the CSV file. If you want to override a specific setting, then you can change it by selecting a different value from the drop-down menu.

6. Click Next.

   On this page, you can set the table destination, partitions, and change the column data types.

7. Verify the settings and click Submit to create the table.

   The CREATE TABLE query is triggered.

   Hue displays the logs and opens the **Table Browser** from which you can view the newly created table when the operation completes successfully.

Viewing query history in Cloudera Data Warehouse

In Cloudera Data Warehouse, you can view all queries that were run against a Database Catalog from Hue, Beeline, Hive Warehouse Connector (HWC), Tableau, Impala-shell, Impyla, and so on.

About this task

You need to set up Query Processor administrators to view the list of all queries from all users, or to restrict viewing of queries.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, navigate to the **Overview** page.
3. From a Virtual Warehouse, launch Hue.
4. Click on the Jobs icon on the left-assist panel.

   The **Job Browser** page is displayed.

5. Go to the **Queries** tab to view query history and query details.

Related Information

Viewing Hive query details
Viewing Impala query details
Adding Query Processor Administrator users and groups in Cloudera Data Warehouse

Cloudera Data Engineering Airflow ETL workloads in Hive Virtual Warehouses

Cloudera Data Engineering can use Apache Airflow to create jobs that run ETL workloads on Hive Virtual Warehouses in Cloudera Data Warehouse.

Hive Virtual Warehouses, which use LLAP daemons (one per executor), are optimized to run interactive queries. However, when the HiveServer query planner receives queries, it examines the scan size, and if the scan size exceeds the value specified for the hive.query.isolation.scan.size.threshold parameter, the planner runs the query in isolation mode. This means that an isolated standalone executor group is spawned to run the data-intensive ETL-type query. This applies to query workloads that originate in Cloudera Data Warehouse. For more information about query isolation mode, see Hive query isolation.

When query workloads originate in Cloudera Data Engineering and are sent to Hive Virtual Warehouses in Cloudera Data Warehouse using automated Airflow data pipelines, the HiveServer query planner assumes these are ETL workloads and these workloads are automatically processed in query isolation mode.

To set up automated data pipelines using Airflow in Cloudera Data Engineering, you must copy the Hive Virtual Warehouse JDBC URL in the Cloudera Data Warehouse UI to a text editor. Then you must navigate to the Cloudera Data Engineering experience and configure the connection for the data pipeline using Airflow. For details about setting this up, see Automating data pipelines with Cloudera Data Engineering and Cloudera Data Warehouse using Apache Airflow.

Simplify queries with User-defined functions

Learn how to use built-in Hive and Impala functions or create custom user-defined functions (UDFs) for specific needs.

Register UDFs using Hive commands and incorporate them into queries. Impala supports UDFs, enabling custom logic for processing column values, complex calculations, and data transformations. These UDFs streamline query logic and enhance flexibility in data processing.

Creating a Hive user-defined function

You can call a built-in Hive function to run one of a wide-range of operations instead of performing multiple steps. You can create a user-defined function (UDF) when a built-in is not available to do what you need.

Assuming you have installed Java and a Java integrated development environment (IDE) tool, such as IntelliJ, you can create the UDF offline. For example, you can create the UDF on your laptop. You export a user-defined function (UDF) to a JAR from a Hadoop- and Hive-compatible Java project and store the JAR on the object store. Using Hive commands, you register the UDF based on the JAR, and call the UDF from a Hive query.

Setting up the development environment

You can create a Hive UDF in a development environment using IntelliJ, for example, and build the UDF. You define the Cloudera Maven Repository in your POM, which accesses necessary JARS hadoop-common-<version>.jar and hive-exec-<version>.jar.

Procedure

1. Open IntelliJ and create a new Maven-based project. Click Create New Project. Select Maven and the supported Java version as the Project SDK. Click Next.
2. Add archetype information.
   For example:
   - GroupId: com.mycompany.hiveudf
   - ArtifactId: hiveudf
3. Click Next and Finish.
   The generated pom.xml appears in sample-hiveudf.
4. To the pom.xml, add properties to facilitate versioning.
   For example:

```
<properties>
   <hadoop.version>TBD</hadoop.version>
   <hive.version>TBD</hive.version>
</properties>
```

5. In the pom.xml, define the repositories.

   Use internal repositories if you do not have internet access.

```
<repositories>
   <repository>
      <releases>
         <enabled>true</enabled>
         <updatePolicy>always</updatePolicy>
         <checksumPolicy>warn</checksumPolicy>
      </releases>
      <snapshots>
         <enabled>false</enabled>
         <updatePolicy>never</updatePolicy>
         <checksumPolicy>fail</checksumPolicy>
      </snapshots>
      <id>HDPReleases</id>
      <name>HDP Releases</name>
      <url>http://repo.hortonworks.com/content/repositories/releases/</u
rl>
      <layout>default</layout>
   </repository>
   <repository>
      <id>public.repo.hortonworks.com</id>
      <name>Public Hortonworks Maven Repo</name>
      <url>http://repo.hortonworks.com/content/groups/public/</url>
      <snapshots>
         <enabled>false</enabled>
      </snapshots>
   </repository>
   <repository>
         <id>repository.cloudera.com</id>
         <url>https://repository.cloudera.com/artifactory/cloudera-repos/<
/url>
   </repository>
 </repositories>
```

6. Define dependencies.
   For example:

```
<dependencies>
  <dependency>
       <groupId>org.apache.hive</groupId>
```

```
        <artifactId>hive-exec</artifactId>
        <version>${hive.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>${hadoop.version}</version>
    </dependency>
</dependencies>
```

Create the UDF class

You define the UDF logic in a new class that returns the data type of a selected column in a table.

Procedure

1. In IntelliJ, click the vertical project tab, and expand hiveudf: hiveudf src main . Select the java directory, and on the context menu, select New Java Class , and name the class, for example, TypeOf.
2. Extend the GenericUDF class to include the logic that identifies the data type of a column.
   For example:

```
package com.mycompany.hiveudf;

import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDF;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.primitive.\
PrimitiveObjectInspectorFactory;
import org.apache.hadoop.io.Text;
public class TypeOf extends GenericUDF {
  private final Text output = new Text();
@Override
  public ObjectInspector initialize(ObjectInspector[] arguments) throws U
DFArgumentException {
    checkArgsSize(arguments, 1, 1);
    checkArgPrimitive(arguments, 0);
    ObjectInspector outputOI = PrimitiveObjectInspectorFactory.writableSt
ringObjectInspector;
    return outputOI;
  }

@Override
  public Object evaluate(DeferredObject[] arguments) throws HiveException
 {
    Object obj;
    if ((obj = arguments[0].get()) == null) {
      String res = "Type: NULL";
      output.set(res);
      } else {
      String res = "Type: " + obj.getClass().getName();
      output.set(res);
    }
    return output;
  }

@Override
  public String getDisplayString(String[] children) {
    return getStandardDisplayString("TYPEOF", children, ",");
  }
}
```

Building the project and uploading the JAR

First, you compile the UDF code into a JAR, and then you add the JAR to Cloudera Data Warehouse object storage.

Before you begin

You have the EnvironmentAdmin role permissions to upload the JAR to your object storage.

Procedure

1. Build the IntelliJ project.

```
...
[INFO] Building jar: /Users/max/IdeaProjects/hiveudf/target/TypeOf-1.0-S
NAPSHOT.jar
[INFO] ------------------------------------------------------------------
-------
[INFO] BUILD SUCCESS
[INFO] -------------------------------------------------------------------
--------
[INFO] Total time: 14.820 s
[INFO] Finished at: 2019-04-03T16:53:04-07:00
[INFO] Final Memory: 26M/397M
[INFO] -------------------------------------------------------------------
--------

Process finished with exit code 0
```

2. In IntelliJ, navigate to the JAR in the /target directory of the project.
3. In Cloudera Data Warehouse, click Overview Database Catalogs , select your Database Catalog, and click options ⋮ , and then click Edit.
4. Upload the JAR to the Hive warehouse on Cloudera Data Warehouse object storage.
   - AWS object storage

     In AWS, upload the JAR file to a bucket on S3 that you can access, for example S3a://my-bucket/path. Add an external AWS S3 bucket if necessary.
   - Azure object storage

     In Azure, upload the JAR file to a default Azure Blob Storage (ABFS) location that you can access, for example abfs://my-storage/path.
5. In IntelliJ, click Save.
6. Click Actions Deploy Client Configuration .
7. Restart the Hive Virtual Warehouse.

Related Information

Adding access to external S3 buckets

Configuring UDF JAR caching in Hive Virtual Warehouse

After you write and compile your User Defined Function (UDF) code into a Java Archive (JAR) file, you can configure a Hive Virtual Warehouse to cache the UDF JAR in HiveServer (HS2) in Cloudera Data Warehouse.

About this task

UDFs enable you to create custom functions to process records or groups of records. Although Hive provides a comprehensive library of functions, there are gaps for which UDFs are a good solution.

In this task, you configure the Hive Virtual Warehouse to cache the JAR file for quick access by the Virtual Warehouse. After configuring the Virtual Warehouse for caching, the UDF JAR is downloaded from the object store the first time it is called, and then cached. Subsequent calls to the JAR are answered from the cache.

Configuring caching significantly improves performance for queries that use the UDF. Without caching, loading a very large UDF of several hundred MBs can take up to several minutes for each query.

Before you begin

- Create a user-defined function.
- Write, compile, and export your UDF code to a JAR file.
- Upload the UDF JAR file to a bucket or container on AWS or Azure, respectively.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Virtual Warehouses tab, locate the Hive Virtual Warehouse that uses the bucket or container where you placed the UDF JAR file, and click ⋮ Edit .

   The **Virtual Warehouse Details** page is displayed.
3. Go to Configurations HiveServer2 .
4. Select hive-site from the Configuration files drop-down menu, and click Add Custom Configuration.

   The **Custom Configuration** modal is displayed.
5. Add the following configuration information, and then click Add:
   - Configuration Key: hive.server2.udf.cache.enabled
   - Configuration Value: true
6. Click Apply Changes.
7. Verify that the configuration property and setting have been added by searching for hive.server2.udf.cache.enabl ed in the search box. If the property has been added, the property name is displayed in the KEY column of the table.

Registering the UDF

In Cloudera Data Warehouse, you run a command from a client interface to your Virtual Warehouse, such as Hue, to call the UDF from Hive queries. The UDF persists between HiveServer restarts.

Before you begin

You need to set up UDF access using a Ranger policy as follows:

- Log in to the Cloudera Data Warehouse service and open Ranger from the Database Catalog associated with your Hive Virtual Warehouse.
- On the **Service Manager** page, under the HADOOP SQL section, select the Database Catalog associated with the Hive Virtual Warehouse in which you want to run the UDFs.
- Select the all - database, udf policy and add the users needing access to Hue. To add all users, you can specify {USER}.

Procedure

1. Open Hue from the Hive Virtual Warehouse in Cloudera Data WarehouseDW.

2. Run the registration command by including the JAR location on your object store.

   For example, on AWS:

```
CREATE FUNCTION udftypeof AS 'com.mycompany.hiveudf.TypeOf01' USING JAR
's3a://mybucket/mypath/TypeOf01-1.0-SNAPSHOT.jar';
```

   On Azure:

```
CREATE FUNCTION udftypeof AS 'com.mycompany.hiveudf.TypeOf01' USING JAR
'abfs://mybucket/mypath/TypeOf01-1.0-SNAPSHOT.jar';
```

3. Restart the Virtual Warehouse.
4. Check that the UDF is registered.

```
SHOW FUNCTIONS;
```

   You scroll through the output and find default.typeof.

5. Run a query that calls the UDF.

```
SELECT students.name, udftypeof(students.name) AS type FROM students WHERE
 age=35;
```

User-defined functions (UDFs)

User-defined functions (frequently abbreviated as UDFs) let you code your own application logic for processing column values during an Impala query. For example, a UDF could perform calculations using an external math library, combine several column values into one, do geospatial calculations, or other kinds of tests and transformations that are outside the scope of the built-in SQL operators and functions.

You can use UDFs to simplify query logic when producing reports, or to transform data in flexible ways when copying from one table to another with the INSERT ... SELECT syntax.

You might be familiar with this feature from other database products, under names such as stored functions or stored routines.

Impala support for UDFs is available in Impala 1.2 and higher:

- In Impala 1.1, using UDFs in a query required using the Hive shell. (Because Impala and Hive share the same metastore database, you could switch to Hive to run just those queries requiring UDFs, then switch back to Impala.)
- Starting in Impala 1.2, Impala can run both high-performance native code UDFs written in C++, and Java-based Hive UDFs that you might already have written.
- Impala can run scalar UDFs that return a single value for each row of the result set, and user-defined aggregate functions (UDAFs) that return a value based on a set of rows. Currently, Impala does not support user-defined table functions (UDTFs) or window functions.

UDF concepts

Depending on your use case, you might write all-new functions, reuse Java UDFs that you have already written for Hive, or port Hive Java UDF code to higher-performance native Impala UDFs in C++. You can code either scalar functions for producing results one row at a time, or more complex aggregate functions for doing analysis across. The following sections discuss these different aspects of working with UDFs.

UDFs and UDAFs

Depending on your use case, the user-defined functions (UDFs) you write might accept or produce different numbers of input and output values:

- The most general kind of user-defined function (the one typically referred to by the abbreviation UDF) takes a single input value and produces a single output value. When used in a query, it is called once for each row in the result set. For example:

```
select customer_name, is_frequent_customer(customer_id) from
 customers;
select obfuscate(sensitive_column) from sensitive_data;
```

- A user-defined aggregate function (UDAF) accepts a group of values and returns a single value. You use UDAFs to summarize and condense sets of rows, in the same style as the built-in COUNT, MAX(), SUM(), and AVG() functions. When called in a query that uses the GROU P BY clause, the function is called once for each combination of GROUP BY values. For example:

```
-- Evaluates multiple rows but returns a single value.
select closest_restaurant(latitude, longitude) from places;

-- Evaluates batches of rows and returns a separate value for
 each batch.
select most_profitable_location(store_id, sales, expenses, tax
_rate, depreciation) from franchise_data group by year;
```

- Currently, Impala does not support other categories of user-defined functions, such as user-defined table functions (UDTFs) or window functions.

Native Impala UDFs

Impala supports UDFs written in C++, in addition to supporting existing Hive UDFs written in Java. Cloudera recommends using C++ UDFs because the compiled native code can yield higher performance, with UDF execution time often 10x faster for a C++ UDF than the equivalent Java UDF.

Using Hive UDFs with Impala

Impala can run Java-based user-defined functions (UDFs), originally written for Hive, with no changes, subject to the following conditions:

- The parameters and return value must all use scalar data types supported by Impala. For example, complex or nested types are not supported.
- Hive/Java UDFs must extend org.apache.hadoop.hive.ql.exec.UDF class.
- Currently, Hive UDFs that accept or return the TIMESTAMP type are not supported.
- Prior to Impala 2.5 the return type must be a “Writable” type such as Text or IntWritable, rather than a Java primitive type such as String or int. Otherwise, the UDF returns NULL. In Impala 2.5 and higher, this restriction is lifted, and both UDF arguments and return values can be Java primitive types.
- Hive UDAFs and UDTFs are not supported.
- Typically, a Java UDF will execute several times slower in Impala than the equivalent native UDF written in C++.
- In Impala 2.5 and higher, you can transparently call Hive Java UDFs through Impala, or call Impala Java UDFs through Hive. This feature does not apply to built-in Hive functions. Any Impala Java UDFs created with older versions must be re-created using new CREATE FUNCT ION syntax, without any signature for arguments or the return value.

To take full advantage of the Impala architecture and performance features, you can also write Impala-specific UDFs in C++.

For background about Java-based Hive UDFs, see the Hive documentation for UDF. For examples or tutorials for writing such UDFs, search the web for related blog posts.

The ideal way to understand how to reuse Java-based UDFs (originally written for Hive) with Impala is to take some of the Hive built-in functions (implemented as Java UDFs) and take the

applicable JAR files through the UDF deployment process for Impala, creating new UDFs with different names:

1. Take a copy of the Hive JAR file containing the Hive built-in functions.
2. Use jar tf *JAR_FILE* to see a list of the classes inside the JAR. You will see names like org/apache/hadoop/hive/ql/udf/UDFLower.class and org/apache/hadoop/hive/ql/udf/UDFOPNegative.class. Make a note of the names of the functions you want to experiment with. When you specify the entry points for the Impala CREATE FUNCTION statement, change the slash characters to dots and strip off the .class suffix, for example org.apache.hadoop.hive.ql.udf.UDFLower and org.apache.hadoop.hive.ql.udf.UDFOPNegative.
3. Copy that file to an HDFS location that Impala can read. (In the examples here, we renamed the file to hive-builtins.jar in HDFS for simplicity.)
4. For each Java-based UDF that you want to call through Impala, issue a CREATE FUNCTION statement, with a LOCATION clause containing the full HDFS path of the JAR file, and a SYMBOL clause with the fully qualified name of the class, using dots as separators and without the .class extension. Remember that user-defined functions are associated with a particular database, so issue a USE statement for the appropriate database first, or specify the SQL function name as *DB_NAME.FUNCTION_NAME*. Use completely new names for the SQL functions, because Impala UDFs cannot have the same name as Impala built-in functions.
5. Call the function from your queries, passing arguments of the correct type to match the function signature. These arguments could be references to columns, arithmetic or other kinds of expressions, the results of CAST functions to ensure correct data types, and so on.

Note:

In Impala 2.9 and higher, you can refresh the user-defined functions (UDFs) that Impala recognizes, at the database level, by running the REFRESH FUNCTIONS statement with the database name as an argument. Java-based UDFs can be added to the metastore database through Hive CREATE FUNCTION statements, and made visible to Impala by subsequently running REFRESH FUNCTIONS. For example:

```
CREATE DATABASE shared_udfs;
USE shared_udfs;
...use CREATE FUNCTION statements in Hive to create so
me Java-based UDFs
   that Impala is not initially aware of...
REFRESH FUNCTIONS shared_udfs;
SELECT udf_created_by_hive(c1) FROM ...
```

Java UDF example: Reusing lower() function

For example, the following `impala-shell` session creates an Impala UDF my_lower() that reuses the Java code for the Hive lower(): built-in function. We cannot call it lower() because Impala does not allow UDFs to have the same name as built-in functions. From SQL, we call the function in a basic way (in a query with no WHERE clause), directly on a column, and on the results of a string expression:

```
[localhost:21000] > create database udfs;
[localhost:21000] > use udfs;
localhost:21000] > create function lower(string) returns string
location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.hi
ve.ql.udf.UDFLower';
ERROR: AnalysisException: Function cannot have the same name as a
 builtin: lower
[localhost:21000] > create function my_lower(string) returns s
tring location '/user/hive/udfs/hive.jar' symbol='org.apache.had
oop.hive.ql.udf.UDFLower';
[localhost:21000] > select my_lower('Some String NOT ALREADY LOWE
RCASE');
+-----------------------------------------------------+
```

```
| udfs.my_lower('some string not already lowercase') |
+----------------------------------------------------+
| some string not already lowercase                  |
+----------------------------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > create table t2 (s string);
[localhost:21000] > insert into t2 values ('lower'),('UPPER'),('I
nit cap'),('CamelCase');
Inserted 4 rows in 2.28s
[localhost:21000] > select * from t2;
+-----------+
| s         |
+-----------+
| lower     |
| UPPER     |
| Init cap  |
| CamelCase |
+-----------+
Returned 4 row(s) in 0.47s
[localhost:21000] > select my_lower(s) from t2;
+------------------+
| udfs.my_lower(s) |
+------------------+
| lower            |
| upper            |
| init cap         |
| camelcase        |
+------------------+
Returned 4 row(s) in 0.54s
[localhost:21000] > select my_lower(concat('ABC ',s,' XYZ')) f
rom t2;
+-------------------------------------------+
| udfs.my_lower(concat('abc ', s, ' xyz')) |
+-------------------------------------------+
| abc lower xyz                             |
| abc upper xyz                             |
| abc init cap xyz                          |
| abc camelcase xyz                         |
+-------------------------------------------+
Returned 4 row(s) in 0.22s
```

Java UDF example: Reusing negative() function

Here is an example that reuses the Hive Java code for the negative() built-in function. This example demonstrates how the data types of the arguments must match precisely with the function signature. At first, we create an Impala SQL function that can only accept an integer argument. Impala cannot find a matching function when the query passes a floating-point argument, although we can call the integer version of the function by casting the argument. Then we overload the same function name to also accept a floating-point argument.

```
[localhost:21000] > create table t (x int);
[localhost:21000] > insert into t values (1), (2), (4), (100);
Inserted 4 rows in 1.43s
[localhost:21000] > create function my_neg(bigint) returns bigin
t location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.
hive.ql.udf.UDFOPNegative';
[localhost:21000] > select my_neg(4);
+----------------+
| udfs.my_neg(4) |
+----------------+
| -4             |
+----------------+
[localhost:21000] > select my_neg(x) from t;
```

```
+----------------+
| udfs.my_neg(x) |
+----------------+
| -2             |
| -4             |
| -100           |
+----------------+
Returned 3 row(s) in 0.60s
[localhost:21000] > select my_neg(4.0);
ERROR: AnalysisException: No matching function with signature:
 udfs.my_neg(FLOAT).
[localhost:21000] > select my_neg(cast(4.0 as int));
+-------------------------------+
| udfs.my_neg(cast(4.0 as int)) |
+-------------------------------+
| -4                            |
+-------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > create function my_neg(double) returns double
 location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.h
ive.ql.udf.UDFOPNegative';
[localhost:21000] > select my_neg(4.0);
+------------------+
| udfs.my_neg(4.0) |
+------------------+
| -4               |
+------------------+
Returned 1 row(s) in 0.11s
```

You can find the sample files mentioned here in the Impala github repo.

Runtime environment for UDFs

By default, Impala copies UDFs into /tmp, and you can configure this location through the --local_library_dir startup flag for the `impalad` daemon.

Installing the UDF development package

To develop UDFs for Impala, download and install the impala-udf-devel package (RHEL-based distributions). This package contains header files, sample source, and build configuration files.

1. Download the repo file.
2. Use the yum command to install the package. For the package name, specify impala-udf-devel (RHEL-based distributions).

Note: The UDF development code does not rely on Impala being installed on the same machine. You can write and compile UDFs on a minimal development system, then deploy them on a different one for use with Impala. If you develop UDFs on a server managed by Cloudera Manager through the parcel mechanism, you still install the UDF development kit through the package mechanism; this small standalone package does not interfere with the parcels containing the main Impala code.

When you are ready to start writing your own UDFs, download the sample code and build scripts from the Cloudera sample UDF github. Then see Writing UDFs on page 21 for how to code UDFs, and Examples of creating and using UDFs on page 28 for how to build and run UDFs.

Writing UDFs

Before starting UDF development, make sure to install the development package and download the UDF code samples. Install the Impala UDF development package as described in Installing the UDF development package on page 21.

When writing UDFs:

- Keep in mind the data type differences as you transfer values from the high-level SQL to your lower-level UDF code. For example, in the UDF code you might be much more aware of how many bytes different kinds of integers require.
- Use best practices for function-oriented programming: choose arguments carefully, avoid side effects, make each function do a single thing, and so on.

Getting started with UDF coding

To understand the layout and member variables and functions of the predefined UDF data types, examine the header file /usr/include/impala_udf/udf.h:

```
// This is the only Impala header required to develop UDFs and U
DAs. This header
// contains the types that need to be used and the FunctionCont
ext object. The context
// object serves as the interface object between the UDF/UDA and
 the impala process.
```

For the basic declarations needed to write a scalar UDF, see the header file udf-sample.h within the sample build environment, which defines a simple function named AddUdf():

```
#ifndef IMPALA_UDF_SAMPLE_UDF_H
#define IMPALA_UDF_SAMPLE_UDF_H
#include <impala_udf/udf.h>

using namespace impala_udf;

IntVal AddUdf(FunctionContext* context, const IntVal& arg1, const
 IntVal& arg2);
#endif
```

For sample C++ code for a simple function named AddUdf(), see the source file udf-sample.cc within the sample build environment:

```
#include "udf-sample.h"
// In this sample we are declaring a UDF that adds two ints and
 returns an int.
IntVal AddUdf(FunctionContext* context, const IntVal& arg1, const
 IntVal& arg2) {
  if (arg1.is_null || arg2.is_null) return IntVal::null();
  return IntVal(arg1.val + arg2.val);
}

// Multiple UDFs can be defined in the same file
```

Data types for function arguments and return values

Each value that a user-defined function can accept as an argument or return as a result value must map to a SQL data type that you could specify for a table column.

Currently, Impala UDFs cannot accept arguments or return values of the Impala complex types (STRUCT, ARRAY, or MAP).

Each data type has a corresponding structure defined in the C++ and Java header files, with two member fields and some predefined comparison operators and constructors:

- is_null indicates whether the value is NULL or not. val holds the actual argument or return value when it is non-NULL.
- Each struct also defines a null() member function that constructs an instance of the struct with the is_null flag set.
- The built-in SQL comparison operators and clauses such as <, >=, BETWEEN, and ORDER BY all work automatically based on the SQL return type of each UDF. For example, Impala knows

how to evaluate BETWEEN 1 AND udf_returning_int(col1) or ORDER BY udf_returning_string(col2) without you declaring any comparison operators within the UDF itself.

For convenience within your UDF code, each struct defines == and != operators for comparisons with other structs of the same type. These are for typical C++ comparisons within your own code, not necessarily reproducing SQL semantics. For example, if the is_null flag is set in both structs, they compare as equal. That behavior of null comparisons is different from SQL (where NULL == NULL is NULL rather than true), but more in line with typical C++ behavior.

- Each kind of struct has one or more constructors that define a filled-in instance of the struct, optionally with default values.
- Impala cannot process UDFs that accept the composite or nested types as arguments or return them as result values. This limitation applies both to Impala UDFs written in C++ and Java-based Hive UDFs.
- You can overload functions by creating multiple functions with the same SQL name but different argument types. For overloaded functions, you must use different C++ or Java entry point names in the underlying functions.

The data types defined on the C++ side (in /usr/include/impala_udf/udf.h) are:

- IntVal represents an INT column.
- BigIntVal represents a BIGINT column. Even if you do not need the full range of a BIGINT value, it can be useful to code your function arguments as BigIntVal to make it convenient to call the function with different kinds of integer columns and expressions as arguments. Impala automatically casts smaller integer types to larger ones when appropriate, but does not implicitly cast large integer types to smaller ones.
- SmallIntVal represents a SMALLINT column.
- TinyIntVal represents a TINYINT column.
- StringVal represents a STRING column. It has a len field representing the length of the string, and a ptr field pointing to the string data. It has constructors that create a new StringVal struct based on a null-terminated C-style string, or a pointer plus a length; these new structs still refer to the original string data rather than allocating a new buffer for the data. It also has a constructor that takes a pointer to a FunctionContext struct and a length, that does allocate space for a new copy of the string data, for use in UDFs that return string values.
- BooleanVal represents a BOOLEAN column.
- FloatVal represents a FLOAT column.
- DoubleVal represents a DOUBLE column.
- TimestampVal represents a TIMESTAMP column. It has a date field, a 32-bit integer representing the Gregorian date, that is, the days past the epoch date. It also has a time_of_day field, a 64-bit integer representing the current time of day in nanoseconds.

Variable-length argument lists

UDFs typically take a fixed number of arguments, with each one named explicitly in the signature of your C++ function. Your function can also accept additional optional arguments, all of the same type. For example, you can concatenate two strings, three strings, four strings, and so on. Or you can compare two numbers, three numbers, four numbers, and so on.

To accept a variable-length argument list, code the signature of your function like this:

```
StringVal Concat(FunctionContext* context, const StringVal& sepa
rator,
  int num_var_args, const StringVal* args);
```

In the CREATE FUNCTION statement, after the type of the first optional argument, include ... to indicate it could be followed by more arguments of the same type. For example, the following function accepts a STRING argument, followed by one or more additional STRING arguments:

```
[localhost:21000] > create function my_concat(string, string ...
) returns string location '/user/test_user/udfs/sample.so' symbo
l='Concat';
```

The call from the SQL query must pass at least one argument to the variable-length portion of the argument list.

When Impala calls the function, it fills in the initial set of required arguments, then passes the number of extra arguments and a pointer to the first of those optional arguments.

Handling NULL values

For correctness, performance, and reliability, it is important for each UDF to handle all situations where any NULL values are passed to your function. For example, when passed a NULL, UDFs typically also return NULL. In an aggregate function, which could be passed a combination of real and NULL values, you might make the final value into a NULL (as in CONCAT()), ignore the NULL value (as in AVG()), or treat it the same as a numeric zero or empty string.

Each parameter type, such as IntVal or StringVal, has an is_null Boolean member. Test this flag immediately for each argument to your function, and if it is set, do not refer to the val field of the argument structure. The val field is undefined when the argument is NULL, so your function could go into an infinite loop or produce incorrect results if you skip the special handling for NULL.

If your function returns NULL when passed a NULL value, or in other cases such as when a search string is not found, you can construct a null instance of the return type by using its null() member function.

Memory allocation for UDFs

By default, memory allocated within a UDF is deallocated when the function exits, which could be before the query is finished. The input arguments remain allocated for the lifetime of the function, so you can refer to them in the expressions for your return values. If you use temporary variables to construct all-new string values, use the StringVal() constructor that takes an initial FunctionCont ext* argument followed by a length, and copy the data into the newly allocated memory buffer.

Thread-safe work area for UDFs

One way to improve performance of UDFs is to specify the optional PREPARE_FN and CLOS E_FN clauses on the CREATE FUNCTION statement. The “prepare” function sets up a thread-safe data structure in memory that you can use as a work area. The “close” function deallocates that memory. Each subsequent call to the UDF within the same thread can access that same memory area. There might be several such memory areas allocated on the same host, as UDFs are parallelized using multiple threads.

Within this work area, you can set up predefined lookup tables, or record the results of complex operations on data types such as STRING or TIMESTAMP. Saving the results of previous computations rather than repeating the computation each time is an optimization known as Memoization. For example, if your UDF performs a regular expression match or date manipulation on a column that repeats the same value over and over, you could store the last-computed value or a hash table of already-computed values, and do a fast lookup to find the result for subsequent iterations of the UDF.

Each such function must have the signature:

```
void FUNCTION_NAME(impala_udf::FunctionContext*, impala_udf::F
unctionContext::FunctionScope)
```

Currently, only THREAD_SCOPE is implemented, not FRAGMENT_SCOPE. See udf.h for details about the scope values.

Error handling for UDFs

To handle errors in UDFs, you call functions that are members of the initial FunctionContext* argument passed to your function.

A UDF can record one or more warnings, for conditions that indicate minor, recoverable problems that do not cause the query to stop. The signature for this function is:

```
bool AddWarning(const char* warning_msg);
```

For a serious problem that requires cancelling the query, a UDF can set an error flag that prevents the query from returning any results. The signature for this function is:

```
void SetError(const char* error_msg);
```

Writing user-defined aggregate functions (UDAFs)

User-defined aggregate functions (UDAFs or UDAs) are a powerful and flexible category of user-defined functions. If a query processes N rows, calling a UDAF during the query condenses the result set, anywhere from a single value (such as with the SUM or MAX functions), or some number less than or equal to N (as in queries using the GROUP BY or HAVING clause).

The underlying functions for a UDA

A UDAF must maintain a state value across subsequent calls, so that it can accumulate a result across a set of calls, rather than derive it purely from one set of arguments. For that reason, a UDAF is represented by multiple underlying functions:

- An initialization function that sets any counters to zero, creates empty buffers, and does any other one-time setup for a query.
- An update function that processes the arguments for each row in the query result set and accumulates an intermediate result for each node. For example, this function might increment a counter, append to a string buffer, or set flags.
- A merge function that combines the intermediate results from two different nodes.
- A serialize function that flattens any intermediate values containing pointers, and frees any memory allocated during the init, update, and merge phases.
- A finalize function that either passes through the combined result unchanged, or does one final transformation.

In the SQL syntax, you create a UDAF by using the statement CREATE AGGREGATE FUNCTION. You specify the entry points of the underlying C++ functions using the clauses INIT_FN, UPDATE_FN, MERGE_FN, SERIALIZE_FN, and FINALIZE_FN.

For convenience, you can use a naming convention for the underlying functions and Impala automatically recognizes those entry points. Specify the UPDATE_FN clause, using an entry point name containing the string update or Update. When you omit the other _FN clauses from the SQL statement, Impala looks for entry points with names formed by substituting the update or Update portion of the specified name.

uda-sample.h:

See this file online at: uda-sample.h

uda-sample.cc:

See this file online at: uda-sample.cc

Intermediate results for UDAs

A user-defined aggregate function might produce and combine intermediate results during some phases of processing, using a different data type than the final return value. For example, if you implement a function similar to the built-in AVG() function, it must keep track of two values, the number of values counted and the sum of those values. Or, you might accumulate a string value over the course of a UDA, then in the end return a numeric or Boolean result.

In such a case, specify the data type of the intermediate results using the optional INTERMEDIATE *TYPE_NAME* clause of the CREATE AGGREGATE FUNCTION statement. If the intermediate data is a typeless byte array (for example, to represent a C++ struct or array), specify the type name as CHAR(*N*), with *N* representing the number of bytes in the intermediate result buffer.

For an example of this technique, see the trunc_sum() aggregate function, which accumulates intermediate results of type DOUBLE and returns BIGINT at the end. View the appropriate CREATE FUNCTION statement and the implementation of the underlying TruncSum*() functions on Github.

- test_udfs.py
- test-udas.cc

Building and deploying UDFs

This section explains the steps to compile Impala UDFs from C++ source code, and deploy the resulting libraries for use in Impala queries.

Impala UDF development package ships with a sample build environment for UDFs, that you can study, experiment with, and adapt for your own use.

To build the sample environment:

1. Install the Impala UDF development package as described in Installing the UDF development package on page 21.
2. Run the following commands:

```
cmake .
make
```

The `cmake` configuration command reads the file CMakeLists.txt and generates a Makefile customized for your particular directory paths. Then the `make` command runs the actual build steps based on the rules in the Makefile.

Impala loads the shared library from an HDFS location. After building a shared library containing one or more UDFs, use hdfs dfs or hadoop fs commands to copy the binary file to an HDFS location readable by Impala.

The final step in deployment is to issue a CREATE FUNCTION statement in the `impala-shell` interpreter to make Impala aware of the new function. Because each function is associated with a particular database, always issue a USE statement to the appropriate database before creating a function, or specify a fully qualified name, that is, CREATE FUNCTION *DB_NAME.FUNCTION_NAME*.

As you update the UDF code and redeploy updated versions of a shared library, use DROP FUNCTION and CREATE FUNCTION to let Impala pick up the latest version of the code.

Note:

In Impala 2.5 and higher, Impala UDFs and UDAs written in C++ are persisted in the metastore database. Java UDFs are also persisted, if they were created with the new CREATE FUNCTION syntax for Java UDFs, where the Java function argument and return types are omitted. Java-based UDFs created with the old CREATE FUNCTION syntax do not persist across restarts because they are held in the memory of the `catalogd` daemon. Until you re-create such Java UDFs using the new CREATE FUNCTION syntax, you must reload those Java-based UDFs by running the original CREATE FUNCTION statements again each time you restart the `catalogd` daemon. Prior to Impala 2.5 the requirement to reload functions after a restart applied to both C++ and Java functions.

See CREATE FUNCTION statement and DROP FUNCTION statement for the new syntax for the persistent Java UDFs.

Prerequisites for the build environment are:

1. Install the packages using the appropriate package installation command for your Linux distribution.

```
sudo yum install gcc-c++ cmake boost-devel
sudo yum install impala-udf-devel
# The package name on Ubuntu and Debian is impala-udf-dev.
```

2. Download the UDF sample code:

```
git clone https://github.com/cloudera/impala-udf-samples
cd impala-udf-samples &amp;&amp; cmake . &amp;&amp; make
```

3. Unpack the sample code in udf_samples.tar.gz and use that as a template to set up your build environment.

To build the original samples:

```
# Process CMakeLists.txt and set up appropriate Makefiles.
cmake .
# Generate shared libraries from UDF and UDAF sample code,
# udf_samples/libudfsample.so and udf_samples/libudasample.so
make
```

The sample code to examine, experiment with, and adapt is in these files:

- udf-sample.h: Header file that declares the signature for a scalar UDF (AddUDF).
- udf-sample.cc: Sample source for a simple UDF that adds two integers. Because Impala can reference multiple function entry points from the same shared library, you could add other UDF functions in this file and add their signatures to the corresponding header file.
- udf-sample-test.cc: Basic unit tests for the sample UDF.
- uda-sample.h: Header file that declares the signature for sample aggregate functions. The SQL functions will be called COUNT, AVG, and STRINGCONCAT. Because aggregate functions require more elaborate coding to handle the processing for multiple phases, there are several underlying C++ functions such as CountInit, AvgUpdate, and StringConcatFinalize.
- uda-sample.cc: Sample source for simple UDAFs that demonstrate how to manage the state transitions as the underlying functions are called during the different phases of query processing.
  - The UDAF that imitates the COUNT function keeps track of a single incrementing number; the merge functions combine the intermediate count values from each Impala node, and the combined number is returned verbatim by the finalize function.
  - The UDAF that imitates the AVG function keeps track of two numbers, a count of rows processed and the sum of values for a column. These numbers are updated and merged as with COUNT, then the finalize function divides them to produce and return the final average value.
  - The UDAF that concatenates string values into a comma-separated list demonstrates how to manage storage for a string that increases in length as the function is called for multiple rows.

- uda-sample-test.cc: basic unit tests for the sample UDAFs.

Performance considerations for UDFs

Because a UDF typically processes each row of a table, potentially being called billions of times, the performance of each UDF is a critical factor in the speed of the overall ETL or ELT pipeline. Tiny optimizations you can make within the function body can pay off in a big way when the function is called over and over when processing a huge result set.

Examples of creating and using UDFs

This section demonstrates how to create and use all kinds of user-defined functions (UDFs).

For downloadable examples that you can experiment with, adapt, and use as templates for your own functions, see the Cloudera sample UDF github. You must have already installed the appropriate header files, as explained in Installing the UDF development package on page 21.

Sample C++ UDFs: HasVowels, CountVowels, StripVowels

This example shows 3 separate UDFs that operate on strings and return different data types. In the C++ code, the functions are HasVowels() (checks if a string contains any vowels), CountVowels() (returns the number of vowels in a string), and StripVowels() (returns a new string with vowels removed).

First, we add the signatures for these functions to udf-sample.h in the demo build environment:

```
BooleanVal HasVowels(FunctionContext* context, const StringVal&
input);
IntVal CountVowels(FunctionContext* context, const StringVal& ar
g1);
StringVal StripVowels(FunctionContext* context, const StringVal&
 arg1);
```

Then, we add the bodies of these functions to udf-sample.cc:

```
BooleanVal HasVowels(FunctionContext* context, const StringVal&
input)
{
        if (input.is_null) return BooleanVal::null();

        int index;
        uint8_t *ptr;

        for (ptr = input.ptr, index = 0; index <= input.len; i
ndex++, ptr++)
        {
                uint8_t c = tolower(*ptr);
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o'
 || c == 'u')
                {
                        return BooleanVal(true);
                }
        }
        return BooleanVal(false);
}

IntVal CountVowels(FunctionContext* context, const StringVal&
arg1)
{
        if (arg1.is_null) return IntVal::null();

        int count;
        int index;
```

```
        uint8_t *ptr;

        for (ptr = arg1.ptr, count = 0, index = 0; index <= arg1.
len; index++, ptr++)
        {
                uint8_t c = tolower(*ptr);
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o'
 || c == 'u')
                {
                        count++;
                }
        }
        return IntVal(count);
}

StringVal StripVowels(FunctionContext* context, const StringVal&
 arg1)
{
        if (arg1.is_null) return StringVal::null();

        int index;
        std::string original((const char *)arg1.ptr,arg1.len);
        std::string shorter("");

        for (index = 0; index < original.length(); index++)
        {
                uint8_t c = original[index];
                uint8_t l = tolower(c);

                if (l == 'a' || l == 'e' || l == 'i' || l == 'o'
 || l == 'u')
                {
                        ;
                }
                else
                {
                    shorter.append(1, (char)c);
                }
        }
// The modified string is stored in 'shorter', which is destroyed
 when this function ends. We need to make a string val
// and copy the contents.
        StringVal result(context, shorter.size()); // Only the ve
rsion of the ctor that takes a context object allocates new memo
ry
        memcpy(result.ptr, shorter.c_str(), shorter.size());
        return result;
}
```

We build a shared library, libudfsample.so, and put the library file into HDFS where Impala can read it:

```
$ make
[  0%] Generating udf_samples/uda-sample.ll
[ 16%] Built target uda-sample-ir
[ 33%] Built target udasample
[ 50%] Built target uda-sample-test
[ 50%] Generating udf_samples/udf-sample.ll
[ 66%] Built target udf-sample-ir
Scanning dependencies of target udfsample
[ 83%] Building CXX object CMakeFiles/udfsample.dir/udf-sample.o
Linking CXX shared library udf_samples/libudfsample.so
[ 83%] Built target udfsample
```

```
Linking CXX executable udf_samples/udf-sample-test
[100%] Built target udf-sample-test
$ hdfs dfs -put ./udf_samples/libudfsample.so /user/hive/udfs/li
budfsample.so
```

Finally, we go into the `impala-shell` interpreter where we set up some sample data, issue CREATE FUNCTION statements to set up the SQL function names, and call the functions in some queries:

```
[localhost:21000] > create database udf_testing;
[localhost:21000] > use udf_testing;

[localhost:21000] > create function has_vowels (string) returns b
oolean location '/user/hive/udfs/libudfsample.so' symbol='HasVow
els';
[localhost:21000] > select has_vowels('abc');
+------------------------+
| udfs.has_vowels('abc') |
+------------------------+
| true                   |
+------------------------+
Returned 1 row(s) in 0.13s
[localhost:21000] > select has_vowels('zxcvbnm');
+----------------------------+
| udfs.has_vowels('zxcvbnm') |
+----------------------------+
| false                      |
+----------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select has_vowels(null);
+-----------------------+
| udfs.has_vowels(null) |
+-----------------------+
| NULL                  |
+-----------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > select s, has_vowels(s) from t2;
+-----------+--------------------+
| s         | udfs.has_vowels(s) |
+-----------+--------------------+
| lower     | true               |
| UPPER     | true               |
| Init cap  | true               |
| CamelCase | true               |
+-----------+--------------------+
Returned 4 row(s) in 0.24s
[localhost:21000] > create function count_vowels (string) retur
ns int location '/user/hive/udfs/libudfsample.so' symbol='CountV
owels';
[localhost:21000] > select count_vowels('cat in the hat');
+-------------------------------------+
| udfs.count_vowels('cat in the hat') |
+-------------------------------------+
| 4                                   |
+-------------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select s, count_vowels(s) from t2;
+-----------+----------------------+
| s         | udfs.count_vowels(s) |
+-----------+----------------------+
| lower     | 2                    |
| UPPER     | 2                    |
| Init cap  | 3                    |
```

```
| CamelCase | 4                    |
+-----------+----------------------+
Returned 4 row(s) in 0.23s
[localhost:21000] > select count_vowels(null);
+-------------------------+
| udfs.count_vowels(null) |
+-------------------------+
| NULL                    |
+-------------------------+
Returned 1 row(s) in 0.12s

[localhost:21000] > create function strip_vowels (string) returns
 string location '/user/hive/udfs/libudfsample.so' symbol='Strip
Vowels';
[localhost:21000] > select strip_vowels('abcdefg');
+------------------------------+
| udfs.strip_vowels('abcdefg') |
+------------------------------+
| bcdfg                        |
+------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > select strip_vowels('ABCDEFG');
+------------------------------+
| udfs.strip_vowels('abcdefg') |
+------------------------------+
| BCDFG                        |
+------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select strip_vowels(null);
+-------------------------+
| udfs.strip_vowels(null) |
+-------------------------+
| NULL                    |
+-------------------------+
Returned 1 row(s) in 0.16s
[localhost:21000] > select s, strip_vowels(s) from t2;
+-----------+----------------------+
| s         | udfs.strip_vowels(s) |
+-----------+----------------------+
| lower     | lwr                  |
| UPPER     | PPR                  |
| Init cap  | nt cp                |
| CamelCase | CmlCs                |
+-----------+----------------------+
Returned 4 row(s) in 0.24s
```

Sample C++ UDA: SumOfSquares

```
[localhost:21000] > insert overwrite sos values (1, 1), (2, 0),
(3, 1), (4, 0);
Inserted 4 rows in 1.24s

[localhost:21000] > -- Compute 1 squared + 3 squared, and 2 sq
uared + 4 squared;
[localhost:21000] > select y, sum_of_squares(x) from sos group by
 y;
+---+------------------------+
| y | udfs.sum_of_squares(x) |
+---+------------------------+
| 1 | 10                     |
| 0 | 20                     |
+---+------------------------+
Returned 2 row(s) in 0.43s
```

This example demonstrates a user-defined aggregate function (UDA) that produces the sum of the squares of its input values.

The coding for a UDA is a little more involved than a scalar UDF, because the processing is split into several phases, each implemented by a different function. Each phase is relatively straightforward: the “update” and “merge” phases, where most of the work is done, read an input value and combine it with some accumulated intermediate value.

As in our sample UDF from the previous example, we add function signatures to a header file (in this case, uda-sample.h). Because this is a math-oriented UDA, we make two versions of each function, one accepting an integer value and the other accepting a floating-point value.

```
void SumOfSquaresInit(FunctionContext* context, BigIntVal* val);
void SumOfSquaresInit(FunctionContext* context, DoubleVal* val);

void SumOfSquaresUpdate(FunctionContext* context, const BigIntVal
& input, BigIntVal* val);
void SumOfSquaresUpdate(FunctionContext* context, const Double
Val& input, DoubleVal* val);

void SumOfSquaresMerge(FunctionContext* context, const BigIntV
al& src, BigIntVal* dst);
void SumOfSquaresMerge(FunctionContext* context, const DoubleV
al& src, DoubleVal* dst);

BigIntVal SumOfSquaresFinalize(FunctionContext* context, const Bi
gIntVal& val);
DoubleVal SumOfSquaresFinalize(FunctionContext* context, const Do
ubleVal& val);
```

We add the function bodies to a C++ source file (in this case, uda-sample.cc):

```
void SumOfSquaresInit(FunctionContext* context, BigIntVal* val) {
  val->is_null = false;
  val->val = 0;
}
void SumOfSquaresInit(FunctionContext* context, DoubleVal* val) {
  val->is_null = false;
  val->val = 0.0;
}

void SumOfSquaresUpdate(FunctionContext* context, const BigIntVal
& input, BigIntVal* val) {
  if (input.is_null) return;
  val->val += input.val * input.val;
}
void SumOfSquaresUpdate(FunctionContext* context, const DoubleVal
& input, DoubleVal* val) {
  if (input.is_null) return;
  val->val += input.val * input.val;
}

void SumOfSquaresMerge(FunctionContext* context, const BigIntVal&
 src, BigIntVal* dst) {
  dst->val += src.val;
}
void SumOfSquaresMerge(FunctionContext* context, const DoubleV
al& src, DoubleVal* dst) {
  dst->val += src.val;
}
BigIntVal SumOfSquaresFinalize(FunctionContext* context, const
BigIntVal& val) {
  return val;
```

```
}
DoubleVal SumOfSquaresFinalize(FunctionContext* context, const
 DoubleVal& val) {
  return val;
}
```

As with the sample UDF, we build a shared library and put it into HDFS:

```
$ make
[  0%] Generating udf_samples/uda-sample.ll
[ 16%] Built target uda-sample-ir
Scanning dependencies of target udasample
[ 33%] Building CXX object CMakeFiles/udasample.dir/uda-sample.o
Linking CXX shared library udf_samples/libudasample.so
[ 33%] Built target udasample
Scanning dependencies of target uda-sample-test
[ 50%] Building CXX object CMakeFiles/uda-sample-test.dir/uda-s
ample-test.o
Linking CXX executable udf_samples/uda-sample-test
[ 50%] Built target uda-sample-test
[ 50%] Generating udf_samples/udf-sample.ll
[ 66%] Built target udf-sample-ir
[ 83%] Built target udfsample
[100%] Built target udf-sample-test
$ hdfs dfs -put ./udf_samples/libudasample.so /user/hive/udfs/li
budasample.so
```

To create the SQL function, we issue a CREATE AGGREGATE FUNCTION statement and specify the underlying C++ function names for the different phases:

```
[localhost:21000] > use udf_testing;

[localhost:21000] > create table sos (x bigint, y double);
[localhost:21000] > insert into sos values (1, 1.1), (2, 2.2),
(3, 3.3), (4, 4.4);
Inserted 4 rows in 1.10s

[localhost:21000] > create aggregate function sum_of_squares(b
igint) returns bigint
  > location '/user/hive/udfs/libudasample.so'
  > init_fn='SumOfSquaresInit'
  > update_fn='SumOfSquaresUpdate'
  > merge_fn='SumOfSquaresMerge'
  > finalize_fn='SumOfSquaresFinalize';
[localhost:21000] > -- Compute the same value using literals or
 the UDA;
[localhost:21000] > select 1*1 + 2*2 + 3*3 + 4*4;
+-------------------------------+
| 1 * 1 + 2 * 2 + 3 * 3 + 4 * 4 |
+-------------------------------+
| 30                            |
+-------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select sum_of_squares(x) from sos;
+------------------------+
| udfs.sum_of_squares(x) |
+------------------------+
| 30                     |
+------------------------+
Returned 1 row(s) in 0.35s
```

Until we create the overloaded version of the UDA, it can only handle a single data type. To allow it to handle DOUBLE as well as BIGINT, we issue another CREATE AGGREGATE FUNCTION statement:

```
[localhost:21000] > select sum_of_squares(y) from sos;
ERROR: AnalysisException: No matching function with signature: ud
fs.sum_of_squares(DOUBLE).

[localhost:21000] > create aggregate function sum_of_squares(dou
ble) returns double
  > location '/user/hive/udfs/libudasample.so'
  > init_fn='SumOfSquaresInit'
  > update_fn='SumOfSquaresUpdate'
  > merge_fn='SumOfSquaresMerge'
  > finalize_fn='SumOfSquaresFinalize';

[localhost:21000] > -- Compute the same value using literals or t
he UDA;
[localhost:21000] > select 1.1*1.1 + 2.2*2.2 + 3.3*3.3 + 4.4*4.4;
+-----------------------------------------------+
| 1.1 * 1.1 + 2.2 * 2.2 + 3.3 * 3.3 + 4.4 * 4.4 |
+-----------------------------------------------+
| 36.3                                          |
+-----------------------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select sum_of_squares(y) from sos;
+------------------------+
| udfs.sum_of_squares(y) |
+------------------------+
| 36.3                   |
+------------------------+
Returned 1 row(s) in 0.35s
```

Typically, you use a UDA in queries with GROUP BY clauses, to produce a result set with a separate aggregate value for each combination of values from the GROUP BY clause. Let's change our sample table to use 0 to indicate rows containing even values, and 1 to flag rows containing odd values. Then the GROUP BY query can return two values, the sum of the squares for the even values, and the sum of the squares for the odd values:

Security considerations for UDFs

When the Impala authorization feature is enabled:

- To call a UDF in a query, you must have the required read privilege for any databases and tables used in the query.
- The CREATE FUNCTION statement requires:
  - The CREATE privilege on the database.
  - The ALL privilege on two URIs where the URIs are:
    - The JAR file on the file system. For example:

      ```
      GRANT ALL ON URI 'file:///PATH_TO_MY.JAR' TO ROLE MY_ROLE;
      ```

    - The JAR on HDFS. For example:

      ```
      GRANT ALL ON URI 'hdfs:///PATH/TO/JAR' TO ROLE MY_ROLE
      ```

Limitations and restrictions for Impala UDFs

The following limitations and restrictions apply to Impala UDFs in the current release:

- Impala does not support Hive UDFs that accept or return composite or nested types, or other types not available in Impala tables.
- The Hive current_user() function cannot be called from a Java UDF through Impala.
- All Impala UDFs must be deterministic, that is, produce the same output each time when passed the same argument values. For example, an Impala UDF must not call functions such as rand() to produce different values for each invocation. It must not retrieve data from external sources, such as from disk or over the network.
- An Impala UDF must not spawn other threads or processes.
- Prior to Impala 2.5 when the `catalogd` process is restarted, all UDFs become undefined and must be reloaded. In Impala 2.5 and higher, this limitation only applies to older Java UDFs. Re-create those UDFs using the new CREATE FUNCTION syntax for Java UDFs, which excludes the function signature, to remove the limitation entirely.
- Impala currently does not support user-defined table functions (UDTFs).
- The CHAR and VARCHAR types cannot be used as input arguments or return values for UDFs.

UDF concepts

Depending on your use case, you might write all-new functions, reuse Java UDFs that you have already written for Hive, or port Hive Java UDF code to higher-performance native Impala UDFs in C++. You can code either scalar functions for producing results one row at a time, or more complex aggregate functions for doing analysis across. The following sections discuss these different aspects of working with UDFs.

UDFs and UDAFs

Depending on your use case, the user-defined functions (UDFs) you write might accept or produce different numbers of input and output values:

- The most general kind of user-defined function (the one typically referred to by the abbreviation UDF) takes a single input value and produces a single output value. When used in a query, it is called once for each row in the result set. For example:

```
select customer_name, is_frequent_customer(customer_id) from
 customers;
select obfuscate(sensitive_column) from sensitive_data;
```

- A user-defined aggregate function (UDAF) accepts a group of values and returns a single value. You use UDAFs to summarize and condense sets of rows, in the same style as the built-in COUNT, MAX(), SUM(), and AVG() functions. When called in a query that uses the GROU P BY clause, the function is called once for each combination of GROUP BY values. For example:

```
-- Evaluates multiple rows but returns a single value.
select closest_restaurant(latitude, longitude) from places;

-- Evaluates batches of rows and returns a separate value for
 each batch.
select most_profitable_location(store_id, sales, expenses, tax
_rate, depreciation) from franchise_data group by year;
```

- Currently, Impala does not support other categories of user-defined functions, such as user-defined table functions (UDTFs) or window functions.

Native Impala UDFs

Impala supports UDFs written in C++, in addition to supporting existing Hive UDFs written in Java. Cloudera recommends using C++ UDFs because the compiled native code can yield higher performance, with UDF execution time often 10x faster for a C++ UDF than the equivalent Java UDF.

Using Hive UDFs with Impala

Impala can run Java-based user-defined functions (UDFs), originally written for Hive, with no changes, subject to the following conditions:

- The parameters and return value must all use scalar data types supported by Impala. For example, complex or nested types are not supported.
- Hive/Java UDFs must extend org.apache.hadoop.hive.ql.exec.UDF class.
- Currently, Hive UDFs that accept or return the TIMESTAMP type are not supported.
- Prior to Impala 2.5 the return type must be a “Writable” type such as Text or IntWritable, rather than a Java primitive type such as String or int. Otherwise, the UDF returns NULL. In Impala 2.5 and higher, this restriction is lifted, and both UDF arguments and return values can be Java primitive types.
- Hive UDAFs and UDTFs are not supported.
- Typically, a Java UDF will run several times slower in Impala than the equivalent native UDF written in C++.
- In Impala 2.5 and higher, you can transparently call Hive Java UDFs through Impala, or call Impala Java UDFs through Hive. This feature does not apply to built-in Hive functions. Any Impala Java UDFs created with older versions must be re-created using new CREATE FUNCT ION syntax, without any signature for arguments or the return value.

To take full advantage of the Impala architecture and performance features, you can also write Impala-specific UDFs in C++.

For background about Java-based Hive UDFs, see the Hive documentation for UDF. For examples or tutorials for writing such UDFs, search the web for related blog posts.

The ideal way to understand how to reuse Java-based UDFs (originally written for Hive) with Impala is to take some of the Hive built-in functions (implemented as Java UDFs) and take the applicable JAR files through the UDF deployment process for Impala, creating new UDFs with different names:

1. Take a copy of the Hive JAR file containing the Hive built-in functions.
2. Use jar tf *JAR_FILE* to see a list of the classes inside the JAR. You will see names like org/apac he/hadoop/hive/ql/udf/UDFLower.class and org/apache/hadoop/hive/ql/udf/UDFOPNegative. class. Make a note of the names of the functions you want to experiment with. When you specify the entry points for the Impala CREATE FUNCTION statement, change the slash characters to dots and strip off the .class suffix, for example org.apache.hadoop.hive.ql.udf.UDFLower and org.apache.hadoop.hive.ql.udf.UDFOPNegative.
3. Copy that file to an HDFS location that Impala can read. (In the examples here, we renamed the file to hive-builtins.jar in HDFS for simplicity.)
4. For each Java-based UDF that you want to call through Impala, issue a CREATE FUNCT ION statement, with a LOCATION clause containing the full HDFS path of the JAR file, and a SYMBOL clause with the fully qualified name of the class, using dots as separators and without the .class extension. Remember that user-defined functions are associated with a particular database, so issue a USE statement for the appropriate database first, or specify the SQL function name as *DB_NAME.FUNCTION_NAME*. Use completely new names for the SQL functions, because Impala UDFs cannot have the same name as Impala built-in functions.
5. Call the function from your queries, passing arguments of the correct type to match the function signature. These arguments could be references to columns, arithmetic or other kinds of expressions, the results of CAST functions to ensure correct data types, and so on.

Note:

In Impala 2.9 and higher, you can refresh the user-defined functions (UDFs) that Impala recognizes, at the database level, by running the REFRESH FUNCTIONS statement with the database name as an argument. Java-based UDFs can be added to the metastore database through Hive CREATE FUNCTION statements, and made visible to Impala by subsequently running REFRESH FUNCTIONS. For example:

```
CREATE DATABASE shared_udfs;
USE shared_udfs;
...use CREATE FUNCTION statements in Hive to create so
me Java-based UDFs
   that Impala is not initially aware of...
REFRESH FUNCTIONS shared_udfs;
SELECT udf_created_by_hive(c1) FROM ...
```

Java UDF example: Reusing lower() function

For example, the following `impala-shell` session creates an Impala UDF my_lower() that reuses the Java code for the Hive lower(): built-in function. We cannot call it lower() because Impala does not allow UDFs to have the same name as built-in functions. From SQL, we call the function in a basic way (in a query with no WHERE clause), directly on a column, and on the results of a string expression:

```
[localhost:21000] > create database udfs;
[localhost:21000] > use udfs;
localhost:21000] > create function lower(string) returns string
location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.hi
ve.ql.udf.UDFLower';
ERROR: AnalysisException: Function cannot have the same name as a
 builtin: lower
[localhost:21000] > create function my_lower(string) returns s
tring location '/user/hive/udfs/hive.jar' symbol='org.apache.had
oop.hive.ql.udf.UDFLower';
[localhost:21000] > select my_lower('Some String NOT ALREADY LOWE
RCASE');
+----------------------------------------------------+
| udfs.my_lower('some string not already lowercase') |
+----------------------------------------------------+
| some string not already lowercase                  |
+----------------------------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > create table t2 (s string);
[localhost:21000] > insert into t2 values ('lower'),('UPPER'),('I
nit cap'),('CamelCase');
Inserted 4 rows in 2.28s
[localhost:21000] > select * from t2;
+-----------+
| s         |
+-----------+
| lower     |
| UPPER     |
| Init cap  |
| CamelCase |
+-----------+
Returned 4 row(s) in 0.47s
[localhost:21000] > select my_lower(s) from t2;
+------------------+
| udfs.my_lower(s) |
+------------------+
| lower            |
| upper            |
```

```
| init cap         |
| camelcase        |
+------------------+
Returned 4 row(s) in 0.54s
[localhost:21000] > select my_lower(concat('ABC ',s,' XYZ')) f
rom t2;
+-----------------------------------------+
| udfs.my_lower(concat('abc ', s, ' xyz')) |
+-----------------------------------------+
| abc lower xyz                           |
| abc upper xyz                           |
| abc init cap xyz                        |
| abc camelcase xyz                       |
+-----------------------------------------+
Returned 4 row(s) in 0.22s
```

Java UDF example: Reusing negative() function

Here is an example that reuses the Hive Java code for the negative() built-in function. This example demonstrates how the data types of the arguments must match precisely with the function signature. At first, we create an Impala SQL function that can only accept an integer argument. Impala cannot find a matching function when the query passes a floating-point argument, although we can call the integer version of the function by casting the argument. Then we overload the same function name to also accept a floating-point argument.

```
[localhost:21000] > create table t (x int);
[localhost:21000] > insert into t values (1), (2), (4), (100);
Inserted 4 rows in 1.43s
[localhost:21000] > create function my_neg(bigint) returns bigin
t location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.
hive.ql.udf.UDFOPNegative';
[localhost:21000] > select my_neg(4);
+----------------+
| udfs.my_neg(4) |
+----------------+
| -4             |
+----------------+
[localhost:21000] > select my_neg(x) from t;
+----------------+
| udfs.my_neg(x) |
+----------------+
| -2             |
| -4             |
| -100           |
+----------------+
Returned 3 row(s) in 0.60s
[localhost:21000] > select my_neg(4.0);
ERROR: AnalysisException: No matching function with signature:
 udfs.my_neg(FLOAT).
[localhost:21000] > select my_neg(cast(4.0 as int));
+--------------------------------+
| udfs.my_neg(cast(4.0 as int)) |
+--------------------------------+
| -4                             |
+--------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > create function my_neg(double) returns double
 location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.h
ive.ql.udf.UDFOPNegative';
[localhost:21000] > select my_neg(4.0);
+------------------+
| udfs.my_neg(4.0) |
+------------------+
```

```
| -4                |
+------------------+
Returned 1 row(s) in 0.11s
```

You can find the sample files mentioned here in the Impala github repo.

Runtime environment for UDFs

By default, Impala copies UDFs into /tmp, and you can configure this location through the --local_library_dir startup flag for the `impalad` daemon.

Writing UDFs

Before starting UDF development, make sure to install the development package and download the UDF code samples.

When writing UDFs:

- Keep in mind the data type differences as you transfer values from the high-level SQL to your lower-level UDF code. For example, in the UDF code you might be much more aware of how many bytes different kinds of integers require.
- Use best practices for function-oriented programming: choose arguments carefully, avoid side effects, make each function do a single thing, and so on.

Getting started with UDF coding

To understand the layout and member variables and functions of the predefined UDF data types, examine the header file /usr/include/impala_udf/udf.h:

```
// This is the only Impala header required to develop UDFs and U
DAs. This header
// contains the types that need to be used and the FunctionCont
ext object. The context
// object serves as the interface object between the UDF/UDA and
 the impala process.
```

For the basic declarations needed to write a scalar UDF, see the header file udf-sample.h within the sample build environment, which defines a simple function named AddUdf():

```
#ifndef IMPALA_UDF_SAMPLE_UDF_H
#define IMPALA_UDF_SAMPLE_UDF_H
#include <impala_udf/udf.h>

using namespace impala_udf;

IntVal AddUdf(FunctionContext* context, const IntVal& arg1, const
 IntVal& arg2);
#endif
```

For sample C++ code for a simple function named AddUdf(), see the source file udf-sample.cc within the sample build environment:

```
#include "udf-sample.h"
// In this sample we are declaring a UDF that adds two ints and
 returns an int.
IntVal AddUdf(FunctionContext* context, const IntVal& arg1, const
 IntVal& arg2) {
  if (arg1.is_null || arg2.is_null) return IntVal::null();
  return IntVal(arg1.val + arg2.val);
}
```

```
// Multiple UDFs can be defined in the same file
```

Data types for function arguments and return values

Each value that a user-defined function can accept as an argument or return as a result value must map to a SQL data type that you could specify for a table column.

Currently, Impala UDFs cannot accept arguments or return values of the Impala complex types (STRUCT, ARRAY, or MAP).

Each data type has a corresponding structure defined in the C++ and Java header files, with two member fields and some predefined comparison operators and constructors:

- is_null indicates whether the value is NULL or not. val holds the actual argument or return value when it is non-NULL.
- Each struct also defines a null() member function that constructs an instance of the struct with the is_null flag set.
- The built-in SQL comparison operators and clauses such as <, >=, BETWEEN, and ORDER BY all work automatically based on the SQL return type of each UDF. For example, Impala knows how to evaluate BETWEEN 1 AND udf_returning_int(col1) or ORDER BY udf_returning_string(col2) without you declaring any comparison operators within the UDF itself.

  For convenience within your UDF code, each struct defines == and != operators for comparisons with other structs of the same type. These are for typical C++ comparisons within your own code, not necessarily reproducing SQL semantics. For example, if the is_null flag is set in both structs, they compare as equal. That behavior of null comparisons is different from SQL (where NULL == NULL is NULL rather than true), but more in line with typical C++ behavior.
- Each kind of struct has one or more constructors that define a filled-in instance of the struct, optionally with default values.
- Impala cannot process UDFs that accept the composite or nested types as arguments or return them as result values. This limitation applies both to Impala UDFs written in C++ and Java-based Hive UDFs.
- You can overload functions by creating multiple functions with the same SQL name but different argument types. For overloaded functions, you must use different C++ or Java entry point names in the underlying functions.

The data types defined on the C++ side (in /usr/include/impala_udf/udf.h) are:

- IntVal represents an INT column.
- BigIntVal represents a BIGINT column. Even if you do not need the full range of a BIGINT value, it can be useful to code your function arguments as BigIntVal to make it convenient to call the function with different kinds of integer columns and expressions as arguments. Impala automatically casts smaller integer types to larger ones when appropriate, but does not implicitly cast large integer types to smaller ones.
- SmallIntVal represents a SMALLINT column.
- TinyIntVal represents a TINYINT column.
- StringVal represents a STRING column. It has a len field representing the length of the string, and a ptr field pointing to the string data. It has constructors that create a new StringVal struct based on a null-terminated C-style string, or a pointer plus a length; these new structs still refer to the original string data rather than allocating a new buffer for the data. It also has a constructor that takes a pointer to a FunctionContext struct and a length, that does allocate space for a new copy of the string data, for use in UDFs that return string values.
- BooleanVal represents a BOOLEAN column.
- FloatVal represents a FLOAT column.
- DoubleVal represents a DOUBLE column.
- TimestampVal represents a TIMESTAMP column. It has a date field, a 32-bit integer representing the Gregorian date, that is, the days past the epoch date. It also has a time_of_day field, a 64-bit integer representing the current time of day in nanoseconds.

Variable-length argument lists

UDFs typically take a fixed number of arguments, with each one named explicitly in the signature of your C++ function. Your function can also accept additional optional arguments, all of the same type. For example, you can concatenate two strings, three strings, four strings, and so on. Or you can compare two numbers, three numbers, four numbers, and so on.

To accept a variable-length argument list, code the signature of your function like this:

```
StringVal Concat(FunctionContext* context, const StringVal& sepa
rator,
  int num_var_args, const StringVal* args);
```

In the CREATE FUNCTION statement, after the type of the first optional argument, include ... to indicate it could be followed by more arguments of the same type. For example, the following function accepts a STRING argument, followed by one or more additional STRING arguments:

```
[localhost:21000] > create function my_concat(string, string ...
) returns string location '/user/test_user/udfs/sample.so' symbo
l='Concat';
```

The call from the SQL query must pass at least one argument to the variable-length portion of the argument list.

When Impala calls the function, it fills in the initial set of required arguments, then passes the number of extra arguments and a pointer to the first of those optional arguments.

Handling NULL values

For correctness, performance, and reliability, it is important for each UDF to handle all situations where any NULL values are passed to your function. For example, when passed a NULL, UDFs typically also return NULL. In an aggregate function, which could be passed a combination of real and NULL values, you might make the final value into a NULL (as in CONCAT()), ignore the NULL value (as in AVG()), or treat it the same as a numeric zero or empty string.

Each parameter type, such as IntVal or StringVal, has an is_null Boolean member. Test this flag immediately for each argument to your function, and if it is set, do not refer to the val field of the argument structure. The val field is undefined when the argument is NULL, so your function could go into an infinite loop or produce incorrect results if you skip the special handling for NULL.

If your function returns NULL when passed a NULL value, or in other cases such as when a search string is not found, you can construct a null instance of the return type by using its null() member function.

Memory allocation for UDFs

By default, memory allocated within a UDF is deallocated when the function exits, which could be before the query is finished. The input arguments remain allocated for the lifetime of the function, so you can refer to them in the expressions for your return values. If you use temporary variables to construct all-new string values, use the StringVal() constructor that takes an initial FunctionCont ext* argument followed by a length, and copy the data into the newly allocated memory buffer.

Thread-safe work area for UDFs

One way to improve performance of UDFs is to specify the optional PREPARE_FN and CLOS E_FN clauses on the CREATE FUNCTION statement. The “prepare” function sets up a thread-safe data structure in memory that you can use as a work area. The “close” function deallocates that memory. Each subsequent call to the UDF within the same thread can access that same memory area. There might be several such memory areas allocated on the same host, as UDFs are parallelized using multiple threads.

Within this work area, you can set up predefined lookup tables, or record the results of complex operations on data types such as STRING or TIMESTAMP. Saving the results of previous

computations rather than repeating the computation each time is an optimization known as Memoization. For example, if your UDF performs a regular expression match or date manipulation on a column that repeats the same value over and over, you could store the last-computed value or a hash table of already-computed values, and do a fast lookup to find the result for subsequent iterations of the UDF.

Each such function must have the signature:

```
void FUNCTION_NAME(impala_udf::FunctionContext*, impala_udf::F
unctionContext::FunctionScope)
```

Currently, only THREAD_SCOPE is implemented, not FRAGMENT_SCOPE. See udf.h for details about the scope values.

Error handling for UDFs

To handle errors in UDFs, you call functions that are members of the initial FunctionContext* argument passed to your function.

A UDF can record one or more warnings, for conditions that indicate minor, recoverable problems that do not cause the query to stop. The signature for this function is:

```
bool AddWarning(const char* warning_msg);
```

For a serious problem that requires cancelling the query, a UDF can set an error flag that prevents the query from returning any results. The signature for this function is:

```
void SetError(const char* error_msg);
```

Related Information

Building and deploying UDFs

Writing user-defined aggregate functions (UDAFs)

User-defined aggregate functions (UDAFs or UDAs) are a powerful and flexible category of user-defined functions. If a query processes N rows, calling a UDAF during the query condenses the result set, anywhere from a single value (such as with the SUM or MAX functions), or some number less than or equal to N (as in queries using the GROUP BY or HAVING clause).

The underlying functions for a UDA

A UDAF must maintain a state value across subsequent calls, so that it can accumulate a result across a set of calls, rather than derive it purely from one set of arguments. For that reason, a UDAF is represented by multiple underlying functions:

- An initialization function that sets any counters to zero, creates empty buffers, and does any other one-time setup for a query.
- An update function that processes the arguments for each row in the query result set and accumulates an intermediate result for each node. For example, this function might increment a counter, append to a string buffer, or set flags.
- A merge function that combines the intermediate results from two different nodes.
- A serialize function that flattens any intermediate values containing pointers, and frees any memory allocated during the init, update, and merge phases.
- A finalize function that either passes through the combined result unchanged, or does one final transformation.

In the SQL syntax, you create a UDAF by using the statement CREATE AGGREGATE FUN CTION. You specify the entry points of the underlying C++ functions using the clauses INIT_FN, UPDATE_FN, MERGE_FN, SERIALIZE_FN, and FINALIZE_FN.

For convenience, you can use a naming convention for the underlying functions and Impala automatically recognizes those entry points. Specify the UPDATE_FN clause, using an entry point name containing the string update or Update. When you omit the other _FN clauses from the SQL statement, Impala looks for entry points with names formed by substituting the update or Update portion of the specified name.

uda-sample.h:

See this file online at: uda-sample.h

uda-sample.cc:

See this file online at: uda-sample.cc

Intermediate results for UDAs

A user-defined aggregate function might produce and combine intermediate results during some phases of processing, using a different data type than the final return value. For example, if you implement a function similar to the built-in AVG() function, it must keep track of two values, the number of values counted and the sum of those values. Or, you might accumulate a string value over the course of a UDA, then in the end return a numeric or Boolean result.

In such a case, specify the data type of the intermediate results using the optional INTERMEDIATE *TYPE_NAME* clause of the CREATE AGGREGATE FUNCTION statement. If the intermediate data is a typeless byte array (for example, to represent a C++ struct or array), specify the type name as CHAR(*N*), with *N* representing the number of bytes in the intermediate result buffer.

For an example of this technique, see the trunc_sum() aggregate function, which accumulates intermediate results of type DOUBLE and returns BIGINT at the end. View the appropriate CREATE FUNCTION statement and the implementation of the underlying TruncSum*() functions on Github.

- test_udfs.py
- test-udas.cc

Building and deploying UDFs

This section explains the steps to compile Impala UDFs from C++ source code, and deploy the resulting libraries for use in Impala queries.

Impala UDF development package ships with a sample build environment for UDFs, that you can study, experiment with, and adapt for your own use.

To build the sample environment:

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. From the **Overview** page, go to **Resources and Downloads** and click See More Impala UDF SDK .
3. Download the impala-udf-devel zip file.
4. SSH into the host on which you want to install the install the UDF environment.
5. Run the following commands:

```
cmake .
make
```

The `cmake` configuration command reads the file CMakeLists.txt and generates a Makefile customized for your particular directory paths. Then the `make` command runs the actual build steps based on the rules in the Makefile.

Impala loads the shared library from an HDFS location. After building a shared library containing one or more UDFs, use hdfs dfs or hadoop fs commands to copy the binary file to an HDFS location readable by Impala.

The final step in deployment is to issue a CREATE FUNCTION statement in the `impala-shell` interpreter to make Impala aware of the new function. Because each function is associated with a particular database, always issue

a USE statement to the appropriate database before creating a function, or specify a fully qualified name, that is, CREATE FUNCTION *DB_NAME.FUNCTION_NAME*.

As you update the UDF code and redeploy updated versions of a shared library, use DROP FUNCTION and CREATE FUNCTION to let Impala pick up the latest version of the code.

Note:

In Impala 2.5 and higher, Impala UDFs and UDAs written in C++ are persisted in the metastore database. Java UDFs are also persisted, if they were created with the new CREATE FUNCTION syntax for Java UDFs, where the Java function argument and return types are omitted. Java-based UDFs created with the old CREATE FUNCTION syntax do not persist across restarts because they are held in the memory of the `catalogd` daemon. Until you re-create such Java UDFs using the new CREATE FUNCTION syntax, you must reload those Java-based UDFs by running the original CREATE FUNCTION statements again each time you restart the `catalogd` daemon. Prior to Impala 2.5 the requirement to reload functions after a restart applied to both C++ and Java functions.

See CREATE FUNCTION statementand DROP FUNCTION statement for the new syntax for the persistent Java UDFs.

Prerequisites for the build environment are:

1. Install the packages using the appropriate package installation command for your Linux distribution.

```
sudo yum install gcc-c++ cmake boost-devel
sudo yum install impala-udf-devel
# The package name on Ubuntu and Debian is impala-udf-dev.
```

2. Download the UDF sample code:

```
git clone https://github.com/cloudera/impala-udf-samples
cd impala-udf-samples &amp;&amp; cmake . &amp;&amp; make
```

3. Unpack the sample code in udf_samples.tar.gz and use that as a template to set up your build environment.

To build the original samples:

```
# Process CMakeLists.txt and set up appropriate Makefiles.
cmake .
# Generate shared libraries from UDF and UDAF sample code,
# udf_samples/libudfsample.so and udf_samples/libudasample.so
make
```

The sample code to examine, experiment with, and adapt is in these files:

- udf-sample.h: Header file that declares the signature for a scalar UDF (AddUDF).
- udf-sample.cc: Sample source for a simple UDF that adds two integers. Because Impala can reference multiple function entry points from the same shared library, you could add other UDF functions in this file and add their signatures to the corresponding header file.
- udf-sample-test.cc: Basic unit tests for the sample UDF.
- uda-sample.h: Header file that declares the signature for sample aggregate functions. The SQL functions will be called COUNT, AVG, and STRINGCONCAT. Because aggregate functions require more elaborate coding to handle the processing for multiple phases, there are several underlying C++ functions such as CountInit, AvgUpdate, and StringConcatFinalize.

- uda-sample.cc: Sample source for simple UDAFs that demonstrate how to manage the state transitions as the underlying functions are called during the different phases of query processing.
  - The UDAF that imitates the COUNT function keeps track of a single incrementing number; the merge functions combine the intermediate count values from each Impala node, and the combined number is returned verbatim by the finalize function.
  - The UDAF that imitates the AVG function keeps track of two numbers, a count of rows processed and the sum of values for a column. These numbers are updated and merged as with COUNT, then the finalize function divides them to produce and return the final average value.
  - The UDAF that concatenates string values into a comma-separated list demonstrates how to manage storage for a string that increases in length as the function is called for multiple rows.
- uda-sample-test.cc: basic unit tests for the sample UDAFs.

Performance considerations for UDFs

Because a UDF typically processes each row of a table, potentially being called billions of times, the performance of each UDF is a critical factor in the speed of the overall ETL or ELT pipeline. Tiny optimizations you can make within the function body can pay off in a big way when the function is called over and over when processing a huge result set.

Examples of creating and using UDFs

This section demonstrates how to create and use all kinds of user-defined functions (UDFs).

For downloadable examples that you can experiment with, adapt, and use as templates for your own functions, see the Cloudera sample UDF github. You must have already installed the appropriate header files, as explained in *Building and deploying UDFs*.

Sample C++ UDFs: HasVowels, CountVowels, StripVowels

This example shows 3 separate UDFs that operate on strings and return different data types. In the C++ code, the functions are HasVowels() (checks if a string contains any vowels), CountVowels() (returns the number of vowels in a string), and StripVowels() (returns a new string with vowels removed).

First, we add the signatures for these functions to udf-sample.h in the demo build environment:

```
BooleanVal HasVowels(FunctionContext* context, const StringVal&
input);
IntVal CountVowels(FunctionContext* context, const StringVal& ar
g1);
StringVal StripVowels(FunctionContext* context, const StringVal&
 arg1);
```

Then, we add the bodies of these functions to udf-sample.cc:

```
BooleanVal HasVowels(FunctionContext* context, const StringVal&
input)
{
        if (input.is_null) return BooleanVal::null();

        int index;
        uint8_t *ptr;

        for (ptr = input.ptr, index = 0; index <= input.len; i
ndex++, ptr++)
        {
                uint8_t c = tolower(*ptr);
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o'
 || c == 'u')
                {
```

```
                        return BooleanVal(true);
                }
        }
        return BooleanVal(false);
}

IntVal CountVowels(FunctionContext* context, const StringVal&
arg1)
{
        if (arg1.is_null) return IntVal::null();

        int count;
        int index;
        uint8_t *ptr;

        for (ptr = arg1.ptr, count = 0, index = 0; index <= arg1.
len; index++, ptr++)
        {
                uint8_t c = tolower(*ptr);
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o'
 || c == 'u')
                {
                        count++;
                }
        }
        return IntVal(count);
}

StringVal StripVowels(FunctionContext* context, const StringVal&
 arg1)
{
        if (arg1.is_null) return StringVal::null();

        int index;
        std::string original((const char *)arg1.ptr,arg1.len);
        std::string shorter("");

        for (index = 0; index < original.length(); index++)
        {
                uint8_t c = original[index];
                uint8_t l = tolower(c);

                if (l == 'a' || l == 'e' || l == 'i' || l == 'o'
 || l == 'u')
                {
                        ;
                }
                else
                {
                    shorter.append(1, (char)c);
                }
        }
// The modified string is stored in 'shorter', which is destroyed
 when this function ends. We need to make a string val
// and copy the contents.
        StringVal result(context, shorter.size()); // Only the ve
rsion of the ctor that takes a context object allocates new memo
ry
        memcpy(result.ptr, shorter.c_str(), shorter.size());
        return result;
}
```

We build a shared library, libudfsample.so, and put the library file into HDFS where Impala can read it:

```
$ make
[  0%] Generating udf_samples/uda-sample.ll
[ 16%] Built target uda-sample-ir
[ 33%] Built target udasample
[ 50%] Built target uda-sample-test
[ 50%] Generating udf_samples/udf-sample.ll
[ 66%] Built target udf-sample-ir
Scanning dependencies of target udfsample
[ 83%] Building CXX object CMakeFiles/udfsample.dir/udf-sample.o
Linking CXX shared library udf_samples/libudfsample.so
[ 83%] Built target udfsample
Linking CXX executable udf_samples/udf-sample-test
[100%] Built target udf-sample-test
$ hdfs dfs -put ./udf_samples/libudfsample.so /user/hive/udfs/li
budfsample.so
```

Finally, we go into the `impala-shell` interpreter where we set up some sample data, issue CREATE FUNCTION statements to set up the SQL function names, and call the functions in some queries:

```
[localhost:21000] > create database udf_testing;
[localhost:21000] > use udf_testing;

[localhost:21000] > create function has_vowels (string) returns b
oolean location '/user/hive/udfs/libudfsample.so' symbol='HasVow
els';
[localhost:21000] > select has_vowels('abc');
+------------------------+
| udfs.has_vowels('abc') |
+------------------------+
| true                   |
+------------------------+
Returned 1 row(s) in 0.13s
[localhost:21000] > select has_vowels('zxcvbnm');
+----------------------------+
| udfs.has_vowels('zxcvbnm') |
+----------------------------+
| false                      |
+----------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select has_vowels(null);
+-----------------------+
| udfs.has_vowels(null) |
+-----------------------+
| NULL                  |
+-----------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > select s, has_vowels(s) from t2;
+-----------+--------------------+
| s         | udfs.has_vowels(s) |
+-----------+--------------------+
| lower     | true               |
| UPPER     | true               |
| Init cap  | true               |
| CamelCase | true               |
+-----------+--------------------+
Returned 4 row(s) in 0.24s
```

```
[localhost:21000] > create function count_vowels (string) retur
ns int location '/user/hive/udfs/libudfsample.so' symbol='CountV
owels';
[localhost:21000] > select count_vowels('cat in the hat');
+-------------------------------------+
| udfs.count_vowels('cat in the hat') |
+-------------------------------------+
| 4                                   |
+-------------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select s, count_vowels(s) from t2;
+-----------+----------------------+
| s         | udfs.count_vowels(s) |
+-----------+----------------------+
| lower     | 2                    |
| UPPER     | 2                    |
| Init cap  | 3                    |
| CamelCase | 4                    |
+-----------+----------------------+
Returned 4 row(s) in 0.23s
[localhost:21000] > select count_vowels(null);
+-------------------------+
| udfs.count_vowels(null) |
+-------------------------+
| NULL                    |
+-------------------------+
Returned 1 row(s) in 0.12s

[localhost:21000] > create function strip_vowels (string) returns
 string location '/user/hive/udfs/libudfsample.so' symbol='Strip
Vowels';
[localhost:21000] > select strip_vowels('abcdefg');
+------------------------------+
| udfs.strip_vowels('abcdefg') |
+------------------------------+
| bcdfg                        |
+------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > select strip_vowels('ABCDEFG');
+------------------------------+
| udfs.strip_vowels('abcdefg') |
+------------------------------+
| BCDFG                        |
+------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select strip_vowels(null);
+-------------------------+
| udfs.strip_vowels(null) |
+-------------------------+
| NULL                    |
+-------------------------+
Returned 1 row(s) in 0.16s
[localhost:21000] > select s, strip_vowels(s) from t2;
+-----------+----------------------+
| s         | udfs.strip_vowels(s) |
+-----------+----------------------+
| lower     | lwr                  |
| UPPER     | PPR                  |
| Init cap  | nt cp                |
| CamelCase | CmlCs                |
+-----------+----------------------+
Returned 4 row(s) in 0.24s
```

Sample C++ UDA: SumOfSquares

```
[localhost:21000] > insert overwrite sos values (1, 1), (2, 0),
(3, 1), (4, 0);
Inserted 4 rows in 1.24s

[localhost:21000] > -- Compute 1 squared + 3 squared, and 2 sq
uared + 4 squared;
[localhost:21000] > select y, sum_of_squares(x) from sos group by
 y;
+---+------------------------+
| y | udfs.sum_of_squares(x) |
+---+------------------------+
| 1 | 10                     |
| 0 | 20                     |
+---+------------------------+
Returned 2 row(s) in 0.43s
```

This example demonstrates a user-defined aggregate function (UDA) that produces the sum of the squares of its input values.

The coding for a UDA is a little more involved than a scalar UDF, because the processing is split into several phases, each implemented by a different function. Each phase is relatively straightforward: the “update” and “merge” phases, where most of the work is done, read an input value and combine it with some accumulated intermediate value.

As in our sample UDF from the previous example, we add function signatures to a header file (in this case, uda-sample.h). Because this is a math-oriented UDA, we make two versions of each function, one accepting an integer value and the other accepting a floating-point value.

```
void SumOfSquaresInit(FunctionContext* context, BigIntVal* val);
void SumOfSquaresInit(FunctionContext* context, DoubleVal* val);

void SumOfSquaresUpdate(FunctionContext* context, const BigIntVal
& input, BigIntVal* val);
void SumOfSquaresUpdate(FunctionContext* context, const Double
Val& input, DoubleVal* val);

void SumOfSquaresMerge(FunctionContext* context, const BigIntV
al& src, BigIntVal* dst);
void SumOfSquaresMerge(FunctionContext* context, const DoubleV
al& src, DoubleVal* dst);

BigIntVal SumOfSquaresFinalize(FunctionContext* context, const Bi
gIntVal& val);
DoubleVal SumOfSquaresFinalize(FunctionContext* context, const Do
ubleVal& val);
```

We add the function bodies to a C++ source file (in this case, uda-sample.cc):

```
void SumOfSquaresInit(FunctionContext* context, BigIntVal* val) {
  val->is_null = false;
  val->val = 0;
}
void SumOfSquaresInit(FunctionContext* context, DoubleVal* val) {
  val->is_null = false;
  val->val = 0.0;
}

void SumOfSquaresUpdate(FunctionContext* context, const BigIntVal
& input, BigIntVal* val) {
  if (input.is_null) return;
```

```
  val->val += input.val * input.val;
}
void SumOfSquaresUpdate(FunctionContext* context, const DoubleVal
& input, DoubleVal* val) {
  if (input.is_null) return;
  val->val += input.val * input.val;
}

void SumOfSquaresMerge(FunctionContext* context, const BigIntVal&
 src, BigIntVal* dst) {
  dst->val += src.val;
}
void SumOfSquaresMerge(FunctionContext* context, const DoubleV
al& src, DoubleVal* dst) {
  dst->val += src.val;
}
BigIntVal SumOfSquaresFinalize(FunctionContext* context, const
BigIntVal& val) {
  return val;
}
DoubleVal SumOfSquaresFinalize(FunctionContext* context, const
 DoubleVal& val) {
  return val;
}
```

As with the sample UDF, we build a shared library and put it into HDFS:

```
$ make
[  0%] Generating udf_samples/uda-sample.ll
[ 16%] Built target uda-sample-ir
Scanning dependencies of target udasample
[ 33%] Building CXX object CMakeFiles/udasample.dir/uda-sample.o
Linking CXX shared library udf_samples/libudasample.so
[ 33%] Built target udasample
Scanning dependencies of target uda-sample-test
[ 50%] Building CXX object CMakeFiles/uda-sample-test.dir/uda-s
ample-test.o
Linking CXX executable udf_samples/uda-sample-test
[ 50%] Built target uda-sample-test
[ 50%] Generating udf_samples/udf-sample.ll
[ 66%] Built target udf-sample-ir
[ 83%] Built target udfsample
[100%] Built target udf-sample-test
$ hdfs dfs -put ./udf_samples/libudasample.so /user/hive/udfs/li
budasample.so
```

To create the SQL function, we issue a CREATE AGGREGATE FUNCTION statement and specify the underlying C++ function names for the different phases:

```
[localhost:21000] > use udf_testing;

[localhost:21000] > create table sos (x bigint, y double);
[localhost:21000] > insert into sos values (1, 1.1), (2, 2.2),
(3, 3.3), (4, 4.4);
Inserted 4 rows in 1.10s

[localhost:21000] > create aggregate function sum_of_squares(b
igint) returns bigint
  > location '/user/hive/udfs/libudasample.so'
  > init_fn='SumOfSquaresInit'
  > update_fn='SumOfSquaresUpdate'
  > merge_fn='SumOfSquaresMerge'
  > finalize_fn='SumOfSquaresFinalize';
```

```
[localhost:21000] > -- Compute the same value using literals or
 the UDA;
[localhost:21000] > select 1*1 + 2*2 + 3*3 + 4*4;
+-------------------------------+
| 1 * 1 + 2 * 2 + 3 * 3 + 4 * 4 |
+-------------------------------+
| 30                            |
+-------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select sum_of_squares(x) from sos;
+------------------------+
| udfs.sum_of_squares(x) |
+------------------------+
| 30                     |
+------------------------+
Returned 1 row(s) in 0.35s
```

Until we create the overloaded version of the UDA, it can only handle a single data type. To allow it to handle DOUBLE as well as BIGINT, we issue another CREATE AGGREGATE FUNCTION statement:

```
[localhost:21000] > select sum_of_squares(y) from sos;
ERROR: AnalysisException: No matching function with signature: ud
fs.sum_of_squares(DOUBLE).

[localhost:21000] > create aggregate function sum_of_squares(dou
ble) returns double
  > location '/user/hive/udfs/libudasample.so'
  > init_fn='SumOfSquaresInit'
  > update_fn='SumOfSquaresUpdate'
  > merge_fn='SumOfSquaresMerge'
  > finalize_fn='SumOfSquaresFinalize';

[localhost:21000] > -- Compute the same value using literals or t
he UDA;
[localhost:21000] > select 1.1*1.1 + 2.2*2.2 + 3.3*3.3 + 4.4*4.4;
+-----------------------------------------------+
| 1.1 * 1.1 + 2.2 * 2.2 + 3.3 * 3.3 + 4.4 * 4.4 |
+-----------------------------------------------+
| 36.3                                          |
+-----------------------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select sum_of_squares(y) from sos;
+------------------------+
| udfs.sum_of_squares(y) |
+------------------------+
| 36.3                   |
+------------------------+
Returned 1 row(s) in 0.35s
```

Typically, you use a UDA in queries with GROUP BY clauses, to produce a result set with a separate aggregate value for each combination of values from the GROUP BY clause. Let's change our sample table to use 0 to indicate rows containing even values, and 1 to flag rows containing odd values. Then the GROUP BY query can return two values, the sum of the squares for the even values, and the sum of the squares for the odd values:

Related Information

Building and deploying UDFs

Security considerations for UDFs

When the Impala authorization feature is enabled:

- To call a UDF in a query, you must have the required read privilege for any databases and tables used in the query.
- The CREATE FUNCTION statement requires:
  - The CREATE privilege on the database.
  - The ALL privilege on two URIs where the URIs are:
    - The JAR file on the file system. For example:

```
GRANT ALL ON URI 'file:///PATH_TO_MY.JAR' TO ROLE MY_ROLE;
```

    - The JAR on HDFS. For example:

```
GRANT ALL ON URI 'hdfs:///PATH/TO/JAR' TO ROLE MY_ROLE
```

Limitations and restrictions for Impala UDFs

The following limitations and restrictions apply to Impala UDFs in the current release.

Limited support for Hive Generic UDFs

Hive has 2 types of UDFs. This release contains limited support for the second generation UDFs called GenericUDFs. The main limitations are as follows:

- Decimal types are not supported
- Complex types are not supported
- Functions are not extracted from the jar file

GenericUDFs cannot be made permanent. They will need to be recreated every time the server is restarted.

Other limitations

- Impala does not support Hive UDFs that accept or return composite or nested types, or other types not available in Impala tables.
- The Hive current_user() function cannot be called from a Java UDF through Impala.
- All Impala UDFs must be deterministic, that is, produce the same output each time when passed the same argument values. For example, an Impala UDF must not call functions such as rand() to produce different values for each invocation. It must not retrieve data from external sources, such as from disk or over the network.
- An Impala UDF must not spawn other threads or processes.
- Prior to Impala 2.5 when the `catalogd` process is restarted, all UDFs become undefined and must be reloaded. In Impala 2.5 and higher, this limitation only applies to older Java UDFs. Re-create those UDFs using the new CREATE FUNCTION syntax for Java UDFs, which excludes the function signature, to remove the limitation entirely.
- Impala currently does not support user-defined table functions (UDTFs).
- The CHAR and VARCHAR types cannot be used as input arguments or return values for UDFs.

Starting the SQL AI Assistant in Hue

A SQL AI Assistant has been integrated into Hue with the capability to leverage the power of Large Language Models (LLMs) for various SQL tasks. It helps you to create, edit, optimize, fix, and succinctly summarize queries using natural language and makes SQL development faster, easier, and less error-prone. You can also generate comments and insert them into your queries to improve readability.

About this task

Attention: The SQL AI Assistant operates only on the database that you have selected in the Hue editor, and not necessarily on the one that is displayed on the left-assist bar.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant .

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.
2. Open Hue corresponding to your Virtual Warehouse.
3. Click Assistant on the Hue SQL editor:

Results

The following options are displayed:

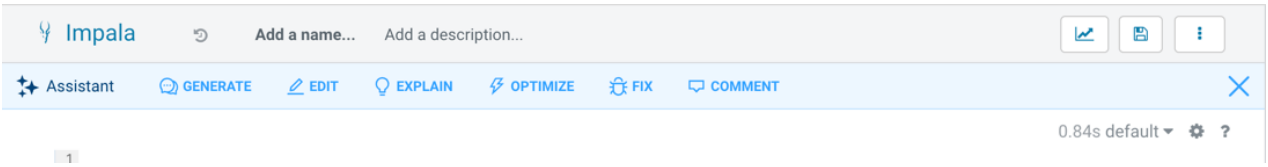

Related Information

About setting up the SQL AI Assistant in Cloudera Data Warehouse

Generating SQL from natural language in Hue

The SQL AI Assistant in Cloudera Data Warehouse helps you to generate SQL queries by entering a prompt in natural language. You can then insert the generated SQL in the Hue SQL editor and run it as usual.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant .

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.
2. Open Hue corresponding to your Virtual Warehouse.

3. Click Assistant on the Hue SQL editor:

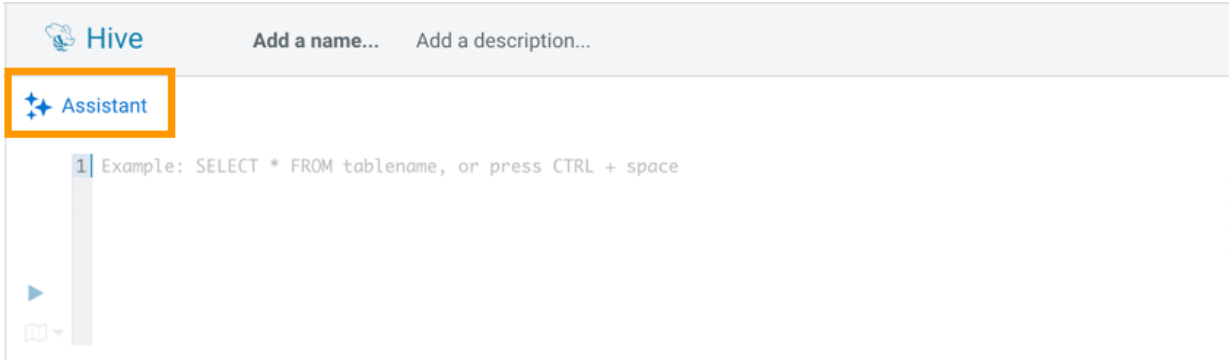

4. Click GENERATE.

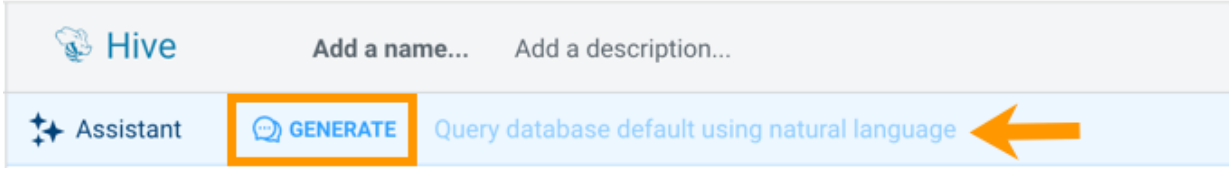

A SQL query is generated based on your input prompt. Click Insert to insert the query into the editor and run it.

Related Information

About setting up the SQL AI Assistant in Cloudera Data Warehouse

Editing the query in natural language in Hue

You can edit a query in natural language and generate it again to fine-tune your results using the Hue SQL AI Assistant. You can then insert the SQL query into the editor and run it as usual.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant .

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.
2. Open Hue corresponding to your Virtual Warehouse.
3. Click Assistant on the Hue SQL editor:

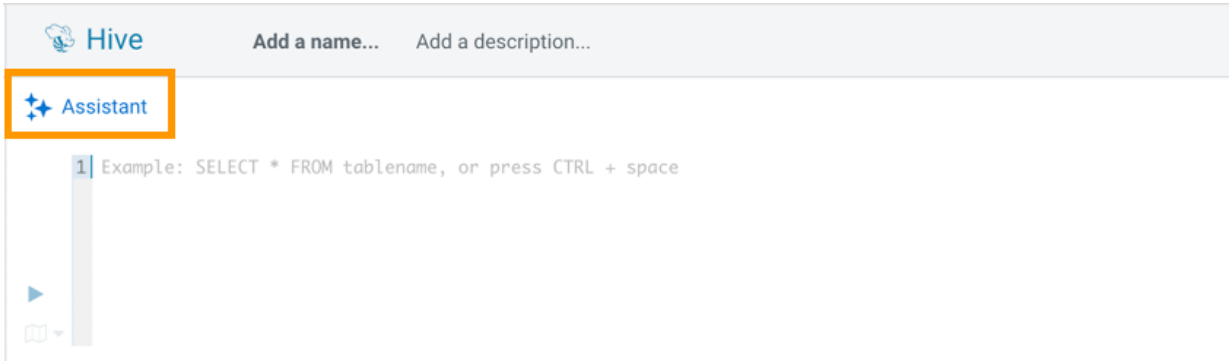

4. Click EDIT.

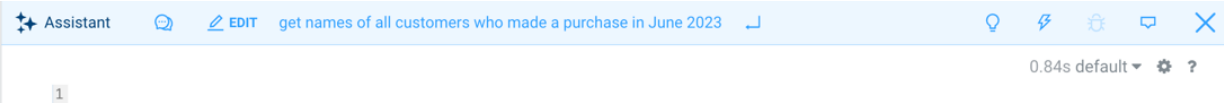

Select the query from the list and edit it as needed. Then press enter to regenerate the SQL.

Related Information

About setting up the SQL AI Assistant in Cloudera Data Warehouse

Getting an explanation of a SQL query in natural language in Hue

You can use the SQL AI Assistant to breakdown and understand a complex SQL query in natural language.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant .

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.
2. Open Hue corresponding to your Virtual Warehouse.
3. Insert a SQL query in the Hue editor that you would like to understand in natural language.
4. Click Assistant on the Hue SQL editor:

5. Click EXPLAIN.

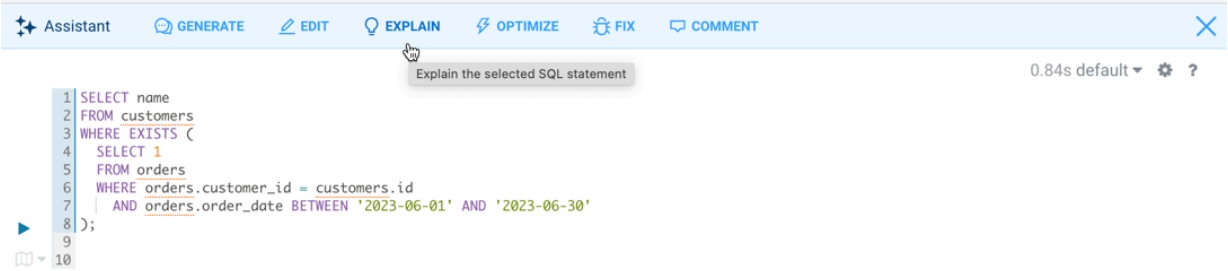

The LLM generates the explanation of the SQL query.

Related Information

About setting up the SQL AI Assistant in Cloudera Data Warehouse

Optimizing a query in Hue

You can use the SQL AI Assistant to optimize a SQL query. Hue identifies the issues in the source query, optimizes it, and provides the optimized version of the SQL query. Hue also summarizes the issues and how it optimized the query in natural language.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant .

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.
2. Open Hue corresponding to your Virtual Warehouse.
3. Insert a SQL query in the Hue editor that you would like to optimize.
4. Click Assistant on the Hue SQL editor:

5. Click OPTIMIZE.

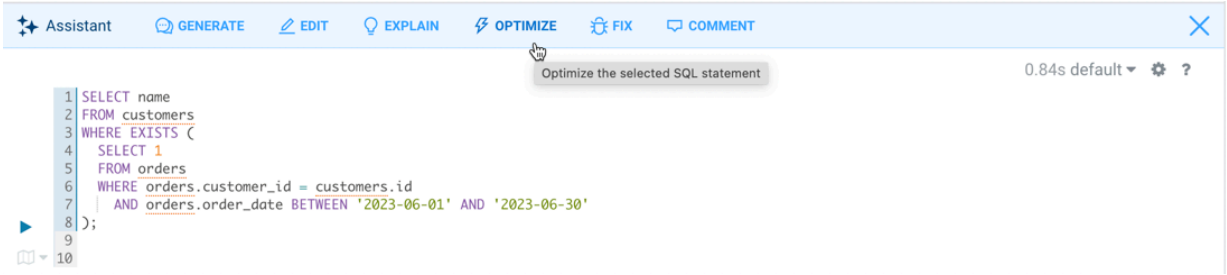

Hue displays the original and the optimized SQL query side-by-side. It also provides an explanation of the issues in the original query and how it was optimized.

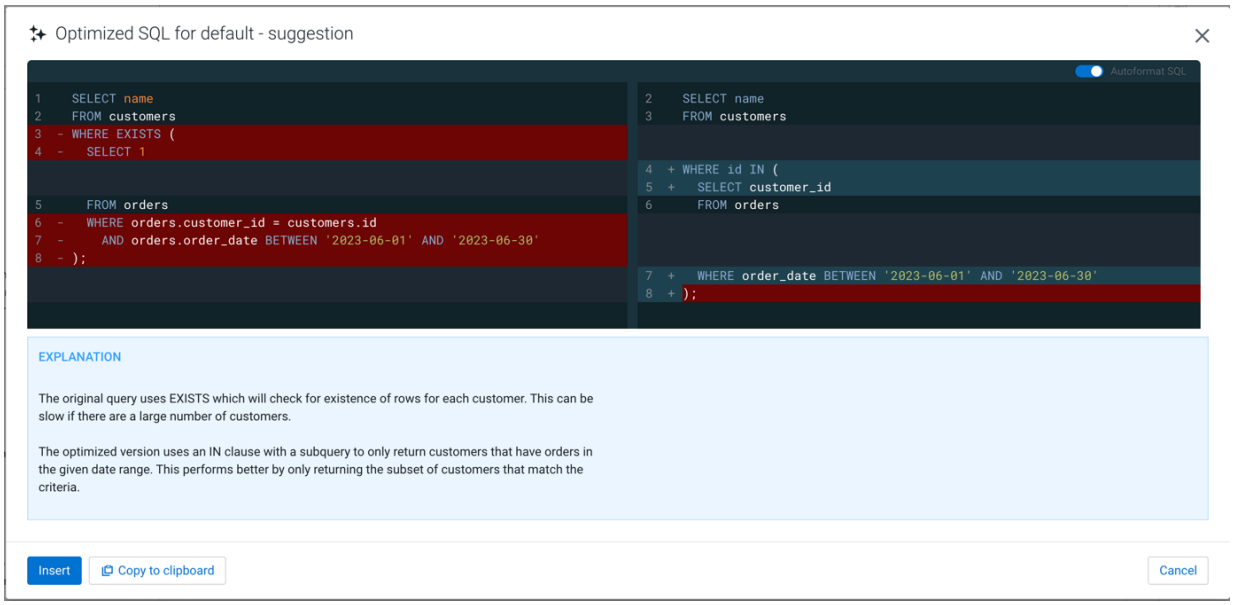

Related Information

About setting up the SQL AI Assistant in Cloudera Data Warehouse

Fixing a query in Hue

You can use the SQL AI Assistant to fix a broken SQL query. Hue identifies the issues in SQL syntax and provides the corrected version.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant .

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.
2. Open Hue corresponding to your Virtual Warehouse.
3. Insert a SQL query in the Hue editor that you would like to fix.

4. Click Assistant on the Hue SQL editor:

5. Click FIX.

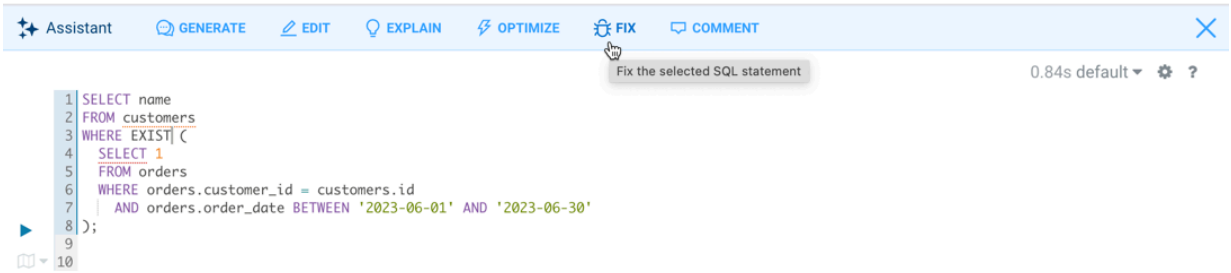

Hue displays the original and the fixed SQL query in a side-by-side comparison.

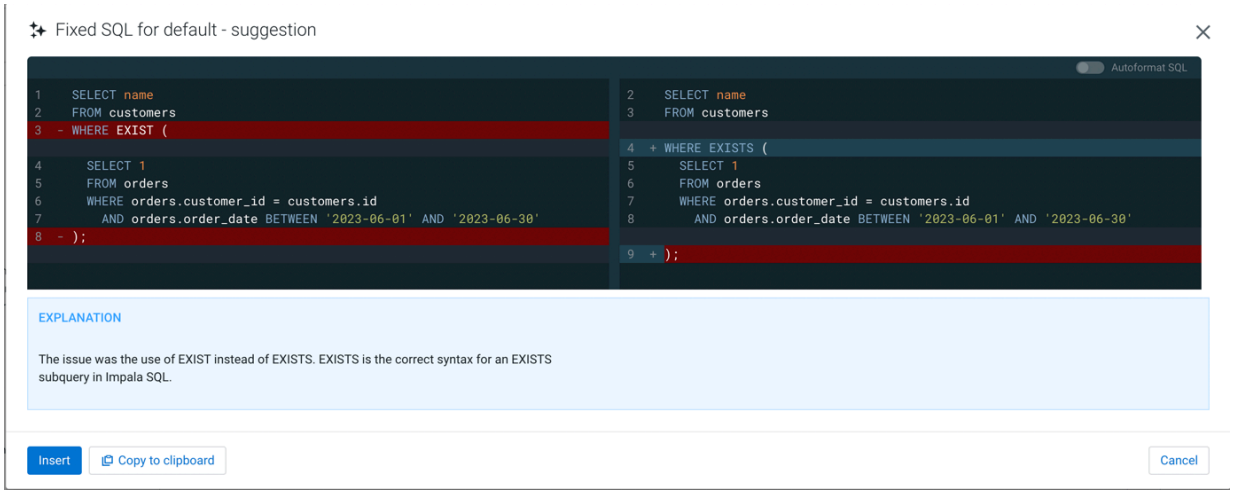

Click Insert to insert the fixed query in the Hue editor and run it.

Related Information

About setting up the SQL AI Assistant in Cloudera Data Warehouse

Generating a comment for a query in Hue

The SQL AI Assistant can generate a comment explaining what SQL query does. You can insert it into the query to improve readability.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant .

Procedure

1. Log in to the Cloudera Data Warehouse service as DWUser.
2. Open Hue corresponding to your Virtual Warehouse.
3. Insert a SQL query in the Hue editor for which you want to generate a comment.
4. Click ✨ Assistant on the Hue SQL editor:

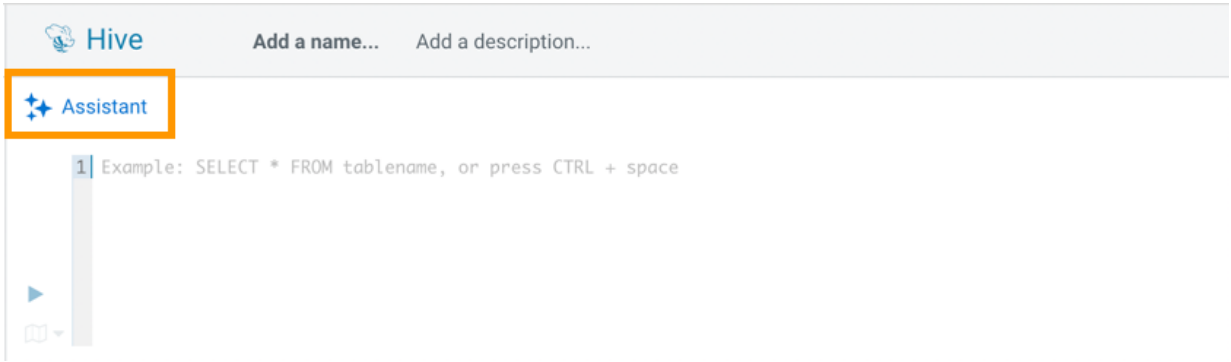

5. Click COMMENT.

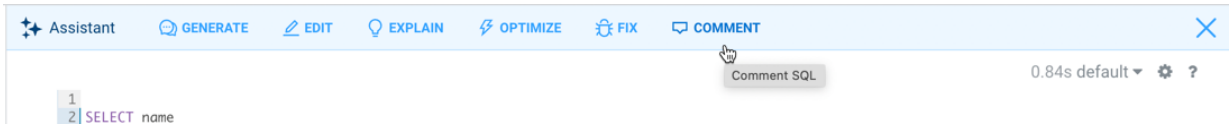

The SQL AI Assistant generates a detailed comment for the input SQL query.

Click Insert to insert the comment into the query.

Cloudera Data Warehouse Public Cloud

Querying Data in Unified Analytics

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Unified Analytics basics 5
- Setting up an Impala Virtual Warehouse 5
- Setting up a Hive Virtual Warehouse 6
- Running a query in Cloudera Data Warehouse 7
- Granting permissions to run SQL queries 7

Using DDL statements 8
- Escaping an invalid identifier 8

Client access to Unified Analytics 9
- Connecting a JDBC client 9
- Connecting an Impala shell client 10
- Using the Impala shell client 10

Materialized views 11
- Creating and using a materialized view 12
  - Creating the tables and view 12
  - Verifying the query rewrite 13
  - Query rewrite example 14
  - Managing query rewrites 15
  - Using optimizations from a subquery 16
  - Dropping a materialized view 17
  - Showing materialized views 17
  - Describing a materialized view 17
  - Periodically rebuilding a materialized view 19
  - Incremental rebuilds 20
  - Purposely using a stale materialized view 21
  - Creating and using a partitioned materialized view 22
- Materialized view commands 24
  - ALTER MATERIALIZED VIEW REBUILD 24
  - ALTER MATERIALIZED VIEW REWRITE 24
  - CREATE MATERIALIZED VIEW 24
  - DESCRIBE EXTENDED and DESCRIBE FORMATTED 26
  - DROP MATERIALIZED VIEW 27
  - SHOW MATERIALIZED VIEWS 28
- Materialized view recommendations introduction 28
- Using the recommender in local mode 29
  - Setting up a mini HiveServer 29
  - Getting wi script help 31
  - Running Recommender examples 33

Speeding up queries using BI mode 42
- Get the JDBC driver 43
- Connect to Unified Analytics with a BI tool 43
- Enable BI mode to rewrite queries automatically 46

Using roll ups and grouping sets **46**

Using set operations **47**

Using anti joins **48**

Creating a table from Parquet data **50**

Query correlated data **50**

Subquery restrictions 51
Comparing tables using ANY/SOME/ALL 51

Handling ETL jobs **52**

Inserting data into a table 52
Updating data in a table 53
Merging data in tables 53
Deleting data from a table 54

Unified Analytics basics

A brief description of Unified Analytics engines and query editors prepares you to run a query. You learn how to select the SQL engine that best suits your needs.

Before using Unified Analytics, understand the concepts behind it.

In Cloudera Data Warehouse, Unified Analytics integrates essential warehouse software:

- SQL engines: Hive and Impala
- Query editors: Hue and the Impala shell

As DW Admin, you use a one- or two-step procedure to create a Virtual Warehouse (VW) for querying your data, as shown in the following diagram:

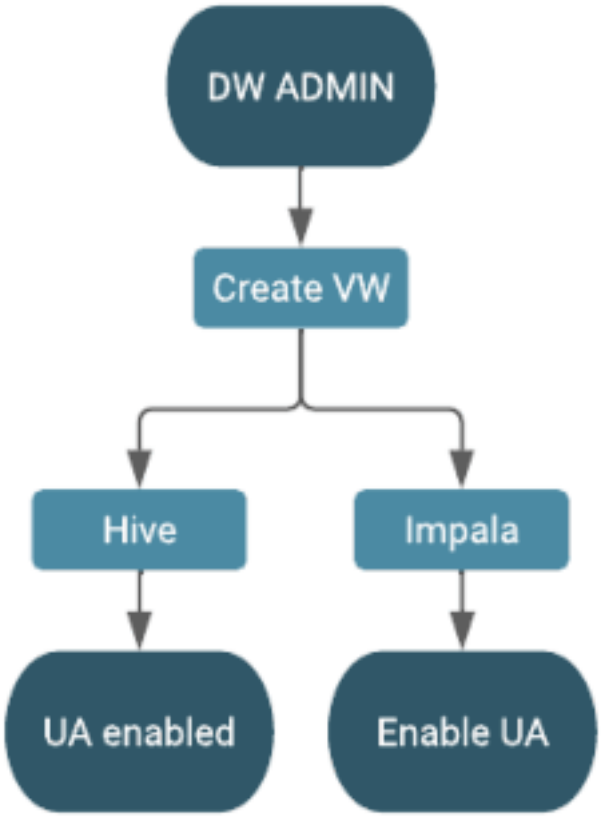

When you create a Hive Virtual Warehouse, Unified Analytics is enabled. When you create an Impala Virtual Warehouse, you need to perform an extra step to enable Unified Analytics.

Related Information

Activating AWS environments

Activating Azure environments

CDP CLI: Create a Database Catalog create-db

Setting up an Impala Virtual Warehouse

You learn how to set up a Unified Analytics Virtual Warehouse for Business Intelligence (BI), or ad-hoc querying. An Impala Virtual Warehouse can read full ACID tables in the Hive metastore, and can write insert-only ACID tables because you select the Impala engine type.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the **Virtual Warehouses** tab and click New Virtual Warehouse.

   The **Create Virtual Warehouse** modal is displayed.
3. Specify a name for your Virtual Warehouse and select an environment, database catalog, Impala, and Hue image versions.
4. Select Impala as the Virtual Warehouse Type.
5. Select a size for the Virtual Warehouse.
6. Select Unified Analytics.
7. Deselect Disable AutoSuspend if you intend to use Impala coordinator shutdown, which you enable in the next step.

   The Impala coordinator does not automatically shutdown unless the Impala executors are suspended.
8. Configure Impala coordinator shutdown by turning on Allow Shutdown Of Coordinator and accept the default value Enabled Active-Passive in High Availability (HA).

   For the benefits and other information about Coordinator shutdown, see configure Impala coordinator shutdown.
9. Set User Groups that can access endpoints, keys and values for Tagging the Virtual Warehouse.
10. Customize other options as necessary.
11. Click Create Virtual Warehouse.

Related Information

CDP CLI: Create a Virtual Warehouse create-vw

Creating a new Virtual Warehouse

Setting up a Hive Virtual Warehouse

You learn how to set up a Hive Unified Analytics Virtual Warehouse. If you have extract, transform, and load (ETL) jobs, you need to learn how to enable query isolation mode.

About this task

When you create a Hive Virtual Warehouse, Unified Analytics is automatically enabled. This type of Virtual Warehouse can read and write full ACID tables in the Hive metastore. ETL jobs, identified by their size, run in query isolation mode if enabled.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the **Virtual Warehouses** tab and click New Virtual Warehouse.

   The **Create Virtual Warehouse** modal is displayed.
3. Specify a name for your Virtual Warehouse and select an environment, database catalog, Impala, and Hue image versions.
4. Select Hive as the Virtual Warehouse Type.
5. Select a size for the Virtual Warehouse.
6. Select the Enable Query Isolation option.

   This action sets the internal configuration property hive.etl.execution.engine to Tez for query isolation.
7. Set User Groups that can access endpoints, keys and values for Tagging the Virtual Warehouse.
8. Click Create Virtual Warehouse.

Related Information

CDP CLI: Create a Virtual Warehouse create-vw

Creating a new Virtual Warehouse

Running a query in Cloudera Data Warehouse

You simply open Hue and submit your query. You do not need to manually start beeline or any other shell.

About this task

As a DWUser, you open Hue from a Virtual Warehouse that you set up, and run the query. The SQL engine reads from and writes to the same metastore, regardless of the type Virtual Warehouse.

Before you begin

- Required role: DW User
- You obtained permission to run SQL queries from the EnvironmentAdmin, who added you to a Hadoop SQL policy.

Procedure

1. Log into the Cloudera Data Warehouse service as DWUser.
2. Go to the **Virtual Warehouses** tab, locate the Virtual Warehouse using which you want to run queries, and click HUE.

   The Hue query editor opens in a new browser tab.
3. To run a query:
   a) Click a database to view the tables it contains.

      When you click a database, it sets it as the target of your query in the main query editor panel.

   b) Type a query in the editor panel and click to run the query.

Related Information

Submitting queries with Hue

Get permission to run SQL queries

Granting permissions to run SQL queries

You must grant Hadoop SQL Ranger permissions to your users so they can query tables.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Database Catalog tab, locate your Database Catalog, and click Open Ranger .

   The Ranger application opens in a new browser tab.
3. Click Hadoop SQL on the **Service Manager** page.
4. Click Add New Policy.
5. Specify the databases, tables, and columns on which you want to grant permissions to the users.
6. Under the **Allow Conditions** section, specify users or groups to whom you want to grant permissions.

   you can specify {USER} to grant permissions to all users.

7. Click Add Permissions and select permissions such as Create, Alter, Drop, Select, and so on.

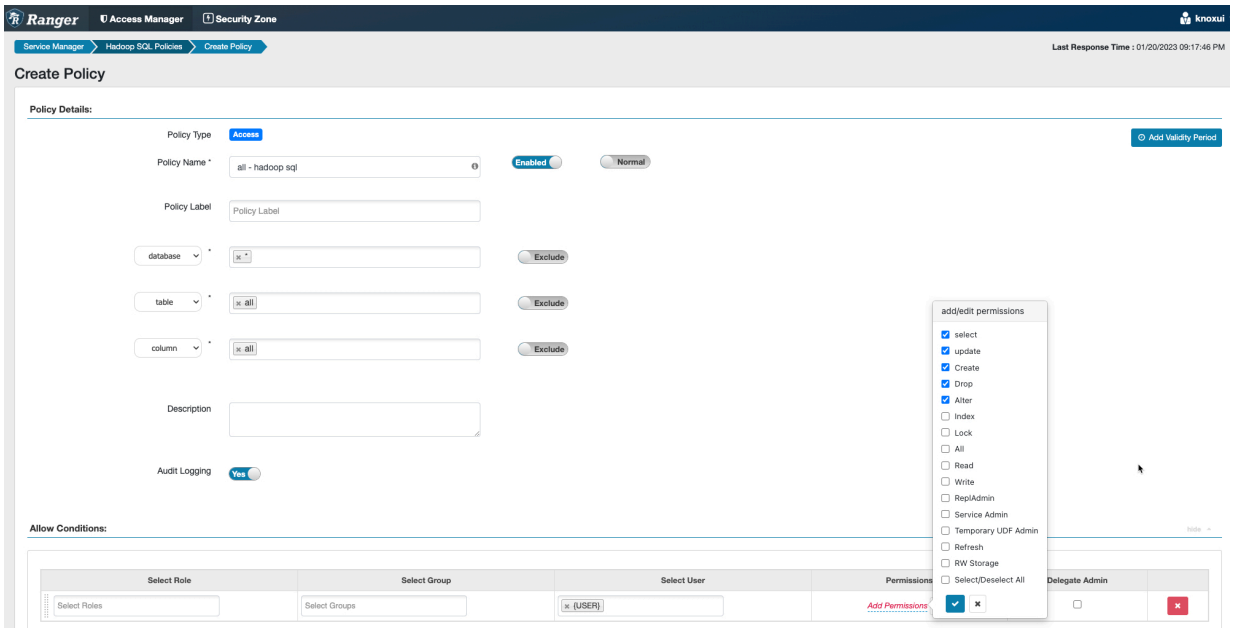

8. Click Add to save the settings.

Using DDL statements

Unified Analytics supports DDL statements to create a database schema and to define the type and structure of the data added to a database.

The following list contains the DDL statements that are currently supported through the Unified Analytics engine.

- Create/Drop/Alter/Use Database
- Create/Drop/Truncate Table
- Create Table Like Parquet
- Alter Table/Partition/Column
- Create/Drop/Alter View
- Create/Drop/Alter Materialized View
- Create/Drop/Alter Scheduled Query
- Create/Drop/Grant/Revoke Roles and Privileges
- Show
- Describe
- Analyze Compute / Drop Statistics
- Refresh Table

Note: For more information about these statements, refer to Hive documentation. Be sure to use Hive syntax when you run these statements on either Unified Analytics SQL engines.

Escaping an invalid identifier

When you need to use reserved words, special characters, or a space in a column or partition name, enclose it in backticks (`).

About this task

An identifier in SQL is a sequence of alphanumeric and underscore (_) characters enclosed in backtick characters. In Hive, these identifiers are called quoted identifiers and are case-insensitive. You can use the identifier instead of a column or table partition name.

Before you begin

You have set the following parameter to column in the hive-site.xml file to enable quoted identifiers:

Set the hive.support.quoted.identifiers configuration parameter to column in the hive-site.xml file to enable quoted identifiers in column names. Valid values are none and column. For example, in Hive execute the following command: SET hive.support.quoted.identifiers = column.

Procedure

1. Create a table named test that has two columns of strings specified by quoted identifiers:
   CREATE TABLE test (`x+y` String, `a?b` String);
2. Create a table that defines a partition using a quoted identifier and a region number:
   CREATE TABLE partition_date-1 (key string, value string) PARTITIONED BY (`dt+x` date, region int);
3. Create a table that defines clustering using a quoted identifier:
   CREATE TABLE bucket_test(`key?1` string, value string) CLUSTERED BY (`key?1`) into 5 buckets;

Client access to Unified Analytics

You learn which interfaces Unified Analytics supports for connecting a client. Procedures for using the JDBC and Impala shell interfaces include step-by-step instructions.

To connect to Unified Analytics you can download the JDBC driver from the Unified Analytics to give to your client. Also, provide the Unified Analytics endpoint to your client to establish the connection.

You can connect and submit requests using the following interfaces:

- JDBC/ODBC
- Beeline
- Impala shell
- Impyla Python client
- Hue web-based user interface

Connecting a JDBC client

You see how to establish the connection between your JDBC client and Unified Analytics.

About this task

In this task, you connect to a Unified Analytics Virtual Warehouse. First, you download the JDBC driver to give to your client. Next, you obtain the Unified Analytics endpoint for the connection. Finally, you provide necessary information to your client.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Resource and Downloads tile and click See More.
3. Click Download JDBC Jar or Download JDBC/ODBC Driver, depending on your Virtual Warehouse SQL engine type.
   This action downloads the JAR (driver) file to your local system.

4. Provide the JAR file you downloaded to your JDBC client. For most of the clients, the JAR file should be added under the Libraries folder. Refer to your client documentation for information about the correct location.
5. Go to the Virtual Warehouses tab, locate the Virtual Warehouse you to connect the JDBC client to, and click ⋮ Copy JDBC URL .
6. Paste the copied JDBC URL into a text file. The URL is in the following format:

```
jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<your-domain.co
m>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
```

7. Provide the Unified Analytics JDBC URL to your client.

```
<your-virtual-warehouse>.<your-environment>.<domain.company.com>
```

8. On the JDBC client end, check that the user can set the authentication credentials:

   Authentication: Username and Password

   Username: Username you use to connect to the Cloudera Data Warehouse service.

   Password: Password you use to connect to the Cloudera Data Warehouse service.

Connecting an Impala shell client

You need to provide the commands to your client users for installing and launching the Impala shell to connect to your Unified Analytics. Client users can then query your tables. You learn how to obtain the command for installing the Impala shell on a client and other information to provide to clients.

About this task

To access Unified Analytics from the Impala Shell, clients need to connect to HiveServer (HS2), which is available in the same Cloudera cluster as Impala. The JDBC URL for the HS2 endpoint uses the strict HS2 protocol option to access Unified Analytics from the Impala Shell. Windows clients are not supported.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Virtual Warehouses tab, locate the Unified Analytics Impala Virtual Warehouse to which you want to connect the Impala shell client, and click ⋮ Copy Unified Analytics Impala shell Download command
3. Click ⋮ again, and select Copy Unified Analytics Impala shell command.

   This action copies the command that launches the Impala shell and includes the connection string your client needs to connect to Unified Analytics in your Virtual Warehouse. For example:

```
impala-shell --protocol='hs2-http' --ssl -i 'coordinator-vw-impala.dw-dwx-
rzs556.xcu2-8y8x.dev.cldr.work:443' -u client_max -l
```

4. Provide the commands you copied to your client user.
5. Provide the instructions to your client user to use as described in *Using the Impala shell client*.

Using the Impala shell client

You obtain the commands from the Impala Virtual Warehouse owner for installing and launching the Impala shell to connect to the Impala Virtual Warehouse. You learn how to install and launch the Impala shell that connects to the Impala Virtual Warehouse UI. You then query tables in the Impala Virtual Warehouse, assuming you have authentication credentials. Windows clients are not supported.

Before you begin

- You must have the latest stable version of Python 2.7.
- You must obtain the pip installer compatible with the Python version.

Procedure

1. In a terminal window, run the command provided by the Impala Virtual Warehouse owner to update impyla to the version compatible with the Impala Virtual Warehouse.
   For example:

   ```
   pip install impyla==0.18a2
   ```

   Installing/updating to impyla 0.18a2, for example, is required before you install the Impala shell in the next step.
2. Run the Impala shell Download command provided by the Impala Virtual Warehouse owner to install the Impala shell.

   The command looks something like this:

   ```
   pip install impala-shell==4.1.0a1
   ```

   The Impala shell is installed.
3. Run the Help command to confirm a successful installation:

   ```
   impala-shell --help
   ```
4. Run the provided Impala shell command to launch the Impala shell.

   The command looks something like this:

   ```
   impala-shell --protocol='hs2-http' --ssl -i 'coordinator-vw-impala.dw-dwx-
   rzs556.xcu2-8y8x.dev.cldr.work:443' -u client_max -l
   ```

   The Impala shell is launched and you connect to the Impala Virtual Warehouse.
5. Run a SQL command to confirm that you are connected to the Impala Virtual Warehouse instance.

   ```
   SHOW DATABASES;
   ```

   The query returns a list of databases in the Impala Virtual Warehouse.

Materialized views

A materialized view is a Hive-managed database object that can be queried by Hive or Impala users. The materialized view holds a query result you can use to speed up the execution of a query workload. If your queries are repetitive, you can reduce latency and resource consumption by using materialized views. You create materialized views to optimize your queries automatically.

Using a materialized view, the optimizer can compare old and new tables, rewrite queries to accelerate processing, and manage maintenance of the materialized view when data updates occur. The optimizer can use a materialized view to fully or partially rewrite projections, filters, joins, and aggregations.

When you enable BI mode to use Apache DataSketches approximations, a materialized view automatically calls DataSketches for some types of operations.

You can perform the following materialized view operations:

- Create a materialized view of queries or subqueries
- Drop a materialized view

- Show materialized views
- Describe a materialized view
- Enable or disable query rewriting based on a materialized view
- Globally enable or disable rewriting based on any materialized view
- Use partitioning to improve the performance of materialized views

Creating and using a materialized view

You can create a materialized view of a query to calculate and store results of an expensive operation, such as a particular join, on a managed, ACID table that you repeatedly run. When you issue queries specified by that materialized view, the optimizer rewrites the query based on it. This action saves reprocessing. Query performance improves.

About this task

In the tasks that follow, first you create, or use an existing, Hive Virtual Warehouse.You create and populate example tables. The tables are managed tables. You cannot create a materialized view of an external table. You create a materialized view of a join of the tables. Subsequently, you run a query to join the tables, and the query plan takes advantage of the precomputed join to accelerate processing. These over-simplified tasks show the syntax and output of a materialized view, and do not demonstrate accelerated processing that occurs in a real-world task, processing a large amount of data.

Before you begin

- You have access to an existing Hive Virtual Warehouse, or you created a new one.
- You have a client connection, such as Beeline, to Hive in Cloudera.

Creating the tables and view

You see how to create simple tables, insert the data, and join the tables using a materialized view. You run the query, and the optimizer takes advantage of the precomputation performed by the materialized view to speed response time.

Procedure

1. Log into the Cloudera Data Warehouse service as DWUser.
2. Go to the **Virtual Warehouses** tab, locate the Virtual Warehouse using which you want to run queries, and click HUE.

   The Hue query editor opens in a new browser tab.
3. Select a database, for example default.
4. Create a managed table.
   For example, create this emps table in the Hue editor:

   ```
   CREATE TABLE emps (
   empid INT,
   deptno INT,
   name VARCHAR(256),
   salary FLOAT,
   hire_date TIMESTAMP);
   ```
5. Create another table.
   For example, create this depts table:

   ```
   CREATE TABLE depts (
   deptno INT,
   deptname VARCHAR(256),
   locationid INT);
   ```

6. Insert data into the tables by copy/pasting the following statements one-by-one and executing them one-by-one.

```
INSERT INTO TABLE emps VALUES (10001,101,CAST('jane doe' as VARCHAR(256)
),250000,'2018-01-10');
INSERT INTO TABLE emps VALUES (10002,100,CAST('somporn klailee' as VARCH
AR(256)),210000,'2017-12-25');
INSERT INTO TABLE emps VALUES (10003,200,CAST('jeiranan thongnopneua' as
VARCHAR(256)),175000,'2018-05-05');
INSERT INTO TABLE depts VALUES (100,CAST('HR' as VARCHAR(256)),10);
INSERT INTO TABLE depts VALUES (101,CAST('Eng' as VARCHAR(256)),11);
INSERT INTO TABLE depts VALUES (200,CAST('Sup' as VARCHAR(256)),20);
```

7. Create a materialized view to join the tables:

```
CREATE MATERIALIZED VIEW mv
AS SELECT empid, deptname, hire_date
FROM emps JOIN depts
ON (emps.deptno = depts.deptno)
WHERE hire_date >= '2017-01-01';
```

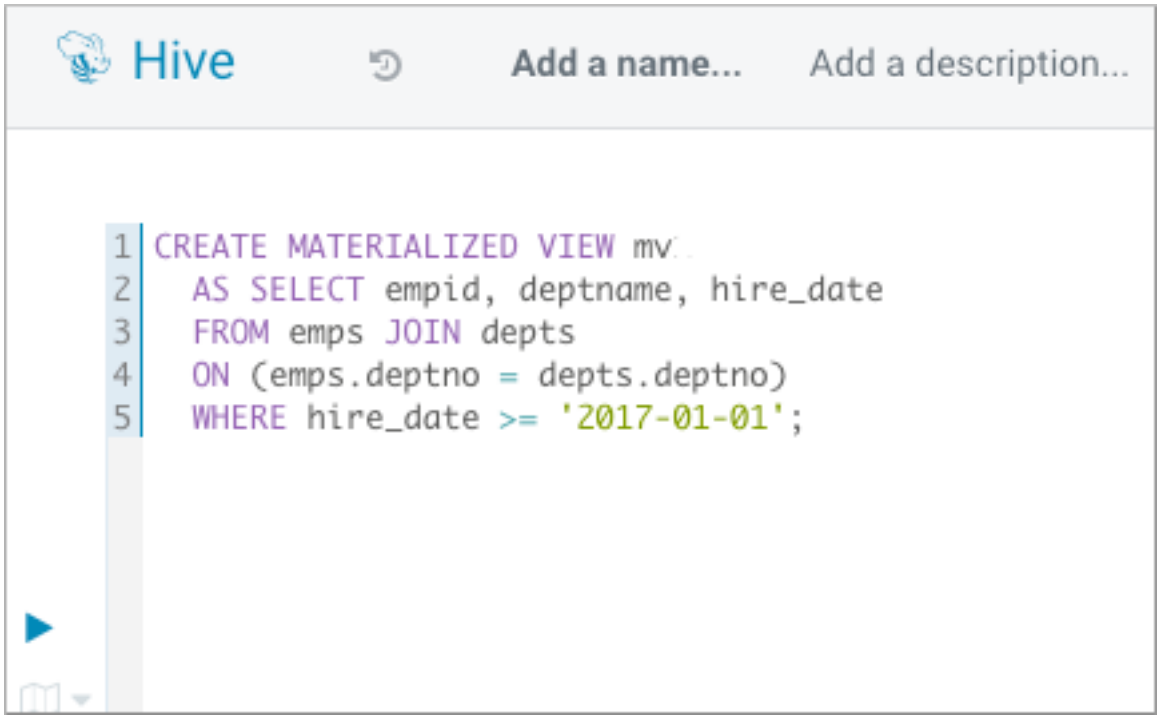

Verifying the query rewrite

You can check that a materialized view is used as a query rewrite.

Procedure

1. Log into the Cloudera Data Warehouse service as DWUser.
2. Go to the **Virtual Warehouses** tab, locate the Virtual Warehouse using which you want to run queries, and click HUE.

   The Hue query editor opens in a new browser tab.
3. In Hue, verify that the query rewrite used the materialized view by running an extended EXPLAIN statement:

```
EXPLAIN EXTENDED SELECT empid, deptname
FROM emps
JOIN depts
```

```
ON (emps.deptno = depts.deptno)
WHERE hire_date >= '2017-01-01'
AND hire_date <= '2019-01-01';
```

4. Export results.
   The output shows the alias default.mv1 for the materialized view in the Scan section of the plan.

```
Explain
OPTIMIZED SQL: SELECT `empid`, `deptname`
FROM `default`.`mv`
WHERE '2019-01-01' >= `hire_date`
STAGE DEPENDENCIES:
Stage-1 is a root stage
Stage-0 is a root stage
STAGE PLANS:
Stage: Stage-1
Impala
Impala Plan:
F01:PLAN FRAGMENT [UNPARTITIONED] hosts=1 instances=1
| Per-Instance Resources: mem-estimate=4.02MB mem-reservation=4.00MB threa
d-reservation=1
PLAN-ROOT SINK
| output exprs: default.mv.empid, default.mv.deptname
| mem-estimate=4.00MB mem-reservation=4.00MB spill-buffer=2.00MB thread-
reservation=0
|
01:EXCHANGE [UNPARTITIONED]
| mem-estimate=16.00KB mem-reservation=0B thread-reservation=0
| tuple-ids=0 row-size=32B cardinality=15
| in pipelines: 00(GETNEXT)
|
F00:PLAN FRAGMENT [RANDOM] hosts=1 instances=1
Per-Instance Resources: mem-estimate=16.00MB mem-reservation=24.00KB thre
ad-reservation=1
00:SCAN S3 [default.mv default.mv1, RANDOM]
S3 partitions=1/1 files=1 size=964B
predicates: casttotimestamp('2019-01-01') >= default.mv1.hire_date
stored statistics:
table: rows=unavailable size=964B
columns: unavailable
extrapolated-rows=disabled max-scan-range-rows=unavailable
parquet dictionary predicates: casttotimestamp('2019-01-01') >= default
.mv1.hire_date
mem-estimate=16.00MB mem-reservation=24.00KB thread-reservation=0
tuple-ids=0 row-size=32B cardinality=15
in pipelines: 00(GETNEXT)
Stage: Stage-0
Fetch Operator
limit: -1
Processor Tree:
ListSink
```

Query rewrite example

You examine a materialized view definition and see how an example query that matches a materialized view is rewritten transparently.

The following materialized view definition uses dynamic partitioning by year:

```
CREATE MATERIALIZED VIEW mat_view
PARTITIONED ON (d_year)
AS SELECT d_year, d_moy, d_dom, SUM(ss_sales_price) AS sum_sales
FROM store_sales, date_dim
```

```
WHERE ss_sold_date_sk = d_date_sk AND d_year > 2017
GROUP BY d_year, d_moy, d_dom;
```

With this materialized view in place, when the following query runs, transparent rewriting to speed up the execution occurs.

```
SELECT SUM(ss_sales_price) AS sum_sales
FROM store_sales, date_dim
WHERE ss_sold_date_sk = d_date_sk AND d_year = 2018 AND d_mon IN (1,2,3);
```

The query matches most of the constructs, but not the filter, in the materialized view. Calcite recognizes this query as a full, mapped query, minus the predicate. Calcite rewrites the query as a select from the materialized view. Calcite also recognizes that the materialized view is partitioned, and does partition elimination:

```
SELECT SUM(sum_sales) AS sum_sales
FROM mat_view
WHERE d_year = 2018 AND d_mon IN (1,2,3);
```

The final rewrite selects from the partition, and applies the predicate to the eliminated partition.

```
SELECT SUM(sum_sales) AS sum_sales
 FROM mat_view(d_year = 2018)
 WHERE d_mon IN (1,2,3);
```

Managing query rewrites

After changes to base tables, the data in a materialized view is stale. You need to know how to prevent the SQL optimizer from rewriting queries in this situation. If you want a query executed without regard to a materialized view, for example to measure the execution time difference, you can disable rewriting and then enable it again.

About this task

As administrator, you can globally enable or disable all query rewrites based on materialized views. By default, the optimizer rewrites a query based on a materialized view.

DWAdmin

Procedure

1. Disable rewriting of a query based on a materialized view named mv1 in the default database.

   ```
   ALTER MATERIALIZED VIEW default.mv1 DISABLE REWRITE;
   ```

2. Enable rewriting of a query based on materialized view mv1.

   ```
   ALTER MATERIALIZED VIEW default.mv1 ENABLE REWRITE;
   ```

3. Globally enable rewriting of queries based on materialized views by setting a global property.

   ```
   SET hive.materializedview.rewriting=true;
   ```

Purposely using a stale materialized view

About this task

When materialized view data becomes stale, or when the optimizer cannot determine the data freshness, which is the case if you use external tables, you can use the materialized view by performing these tasks:

- Schedule the materialized view for rebuilding. For example, schedule a rebuild to occur every x minutes.

- Adjust the rewriting time window to use stale or possibly stale data for a period of time. For example, schedule the window within which to use stale data for x + y minutes.

Procedure

1. Create a scheduled query to invoke the rebuild statement every 10 minutes.

```
CREATE SCHEDULED QUERY scheduled_rebuild
EVERY 10 MINUTES AS
ALTER MATERIALIZED VIEW mv_recently_hired REBUILD;
```

2. Define the window of time for using stale data.

```
SET hive.materializedview.rewriting.time.window=10min;
```

Using optimizations from a subquery

You can create a query having a subquery that the optimizer rewrites based on a materialized view. You create a materialized view, and then run a query that uses the materialized view.

About this task

In this task, you create a materialized view and use it in a subquery to return the number of destination-origin pairs. Suppose the data resides in a table named flights_data that has the following columns:

c_id	dest	origin
1	Chicago	Hyderabad
2	London	Moscow
...		

Procedure

1. Create a table schema definition named flights_data for destination and origin data.

```
CREATE TABLE flights_data(
  c_id INT,
  dest VARCHAR(256),
  origin VARCHAR(256));
```

2. Create a materialized view that counts destinations and origins.

```
CREATE MATERIALIZED VIEW mv1
AS
  SELECT dest, origin, count(*)
  FROM flights_data
  GROUP BY dest, origin;
```

3. Take advantage of the materialized view to speed your queries when you have to count destinations and origins again.

   For example, use a subquery to select the number of destination-origin pairs like the materialized view.

```
SELECT count(*)/2
FROM(
  SELECT dest, origin, count(*)
  FROM flights_data
  GROUP BY dest, origin
```

```
) AS t;
```

Transparently, the SQL engine uses the work already in place since creation of the materialized view instead of reprocessing.

Dropping a materialized view

You must understand when to drop a materialized view to successfully drop related tables.

About this task

Drop a materialized view before performing a DROP TABLE operation on a related table. You cannot drop a table that has a relationship with a materialized view.

In this task, you drop a materialized view named mv1 from the database named default.

Procedure

Drop a materialized view in my_database named mv1 .
DROP MATERIALIZED VIEW default.mv;

Showing materialized views

You can list all materialized views in the current database or in another database. You can filter a list of materialized views in a specified database using regular expression wildcards.

About this task

You can use regular expression wildcards to filter the list of materialized views you want to see. The following wildcards are supported:

- Asterisk (*)

  Represents one or more characters.
- Pipe symbol (|)

  Represents a choice.

For example, mv_q* and *mv|q1* match the materialized view mv_q1. Finding no match does not cause an error.

Procedure

1. List materialized views in the current database.
   SHOW MATERIALIZED VIEWS;
2. List materialized views in a particular database.
   SHOW MATERIALIZED VIEWS IN another_database;

Describing a materialized view

You can get summary, detailed, and formatted information about a materialized view.

About this task

This task builds on the task that creates a materialized view named mv1.

Procedure

1. Get summary information about the materialized view named mv1.

```
DESCRIBE mv1;
```

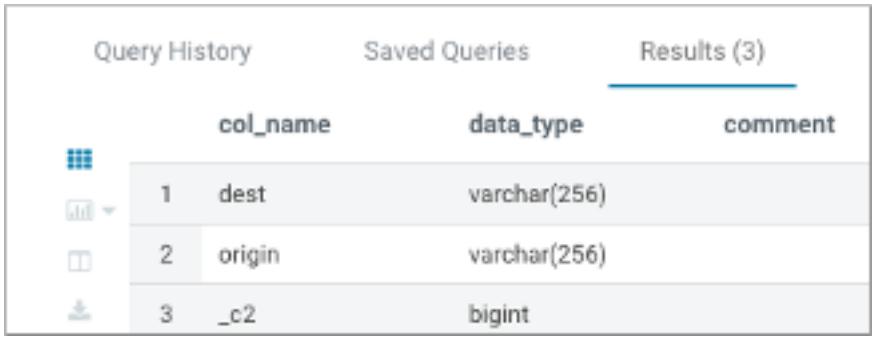

	col_name	data_type	comment
1	dest	varchar(256)	
2	origin	varchar(256)	
3	_c2	bigint	

2. Get detailed information about the materialized view named mv1.

```
DESCRIBE EXTENDED mv1;
```

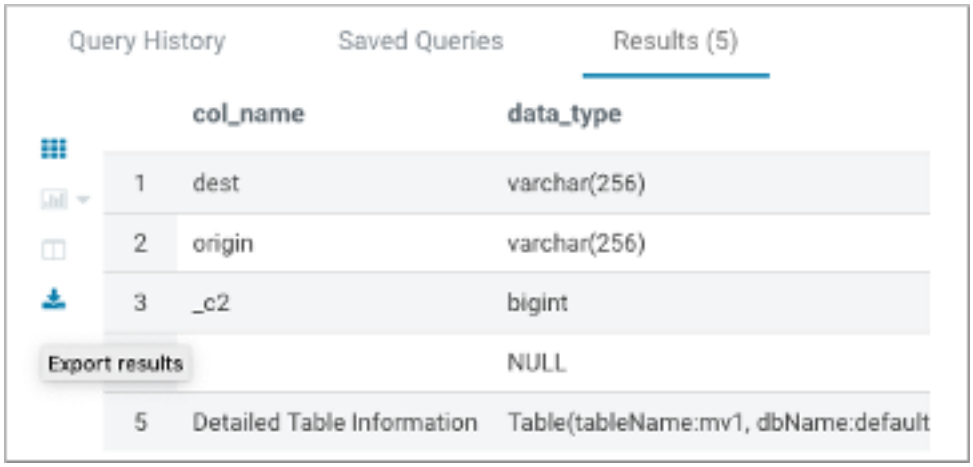

	col_name	data_type
1	dest	varchar(256)
2	origin	varchar(256)
3	_c2	bigint
		NULL
5	Detailed Table Information	Table(tableName:mv1, dbName:default

3. Click Export Results to copy the information.

4. Get formatting details about the materialized view named mv1.

```
DESCRIBE FORMATTED mv1;
```

For example, 14 of the 38 actual results are shown below:

Query History Saved Queries Results (38)

	col_name	data_type
1	dest	varchar(256)
2	origin	varchar(256)
3	_c2	bigint
4		NULL
5	# Detailed Table Information	NULL
6	Database:	default
7	OwnerType:	USER
8	Owner:	csso_khahn
9	CreateTime:	Sat May 22 23:57:47 UTC 2021
10	LastAccessTime:	UNKNOWN
11	Retention:	0
12	Location:	s3a://qe-s3-bucket-weekly-xtx4-
13	Table Type:	MATERIALIZED_VIEW
14	Table Parameters:	NULL

Periodically rebuilding a materialized view

Using materialized views can enhance query performance. You need to update materialized view contents when new data is added to the underlying table. Instead of rebuilding the materialized view manually, you can schedule this task. Automatic rebuilding then occurs periodically and transparently to users.

About this task

This task assumes you created the following schemas for storing employee and departmental information:

```
CREATE TABLE emps (
empid INTEGER,
deptno INTEGER,
name VARCHAR(256),
salary FLOAT,
hire_date TIMESTAMP);

CREATE TABLE depts (
deptno INTEGER,
deptname VARCHAR(256),
locationid INTEGER);
```

Imagine that you add data for a number of employees to the table. Assume many users of your database issue queries to access to data about the employees hired during last year including the department they belong to.

You perform the steps below to create a materialized view of the table to address these queries. Imagine new employees are hired and you add their records to the table. These changes render the materialized view contents outdated. You need to update its contents. You create a scheduled query to perform this task. The scheduled rebuilding will not occur unless there are changes to the input tables. You test the scheduled query by bypassing the schedule and executing the schedule immediately. Finally, you change the schedule to rebuild less often.

Procedure

1. To handle many queries to access recently hired employee and departmental data, create a materialized view.

```
CREATE MATERIALIZED VIEW mv_recently_hired AS
  SELECT empid, name, deptname, hire_date FROM emps
  JOIN depts ON (emps.deptno = depts.deptno)
  WHERE hire_date >= '2020-01-01 00:00:00';
```

2. Use the materialized view by querying the employee data.

```
SELECT empid, name FROM emps
JOIN depts  ON (emps.deptno = depts.deptno)
WHERE hire_date >= '2020-03-01 00:00:00' AND deptname = 'finance';
```

3. Assuming new hiring occurred and you added new records to the emps table, rebuild the materialized view.

```
ALTER MATERIALIZED VIEW mv_recently_hired REBUILD;
```

   The rebuilding updates the contents of the materialized view.

4. Create a scheduled query to invoke the rebuild statement every 10 minutes.

```
CREATE SCHEDULED QUERY scheduled_rebuild
EVERY 10 MINUTES AS
ALTER MATERIALIZED VIEW mv_recently_hired REBUILD;
```

   A rebuild executes every 10 minutes, assuming changes to the emp table occur within that period. If a materialized view can be rebuilt incrementally, the scheduled rebuild does not occur unless there are changes to the input tables.

5. To test the schedule, run a scheduled query immediately.

```
ALTER SCHEDULED QUERY scheduled_rebuild EXECUTE;
```

6. Change the frequency of the rebuilding.

```
ALTER SCHEDULED QUERY scheduled_rebuild EVERY 20 MINUTES;
```

Incremental rebuilds

The materialized view is a physical manifestation and summary of the original data in your table. If the source table, or multiple tables in the case of joins, changes, you likely want updates to the materialized view.

An update can occur as an incremental rebuild, or a full rebuild, of the materialized view. A full rebuild can be expensive. An incremental rebuild updates only the affected parts of the materialized view. An incremental rebuild improves the efficiency of maintaining the materialized view by decreasing rebuild step execution time. The incremental rebuild also relies on existing ACID guarantees of view tables.

The following types of incremental rebuilds occur automatically:

- Record-based

  The delta is generated based on a snapshot of source tables taken at view creation and at rebuild time. Source tables must be full ACID (transactional) tables, which only Hive can write.
- Partition-based

  The partitions that need rebuilding are identified, and then only those partitions are updated by an insert-overwrite. Source tables can be insert-only ACID (tranactional) tables, which Hive and Impala can write.

The materialized view must be transactional to qualify for an incremental rebuild. If the materialized view is not transactional, a full rebuild occurs.

Conditions for a record-based incremental rebuild

A record-based incremental rebuild occurs under the following conditions:

- The materialized view includes a count, sum, min, or max aggregation, and one of the source tables has an additional insert after the view is created.
- The materialized view includes a sum aggregation, and one of the source tables has an additional insert or delete after the view is created.
- You are using a Unified Analytics Virtual Warehouse with a Hive engine, which is the only Virtual Warehouse type that supports the incremental, records-based rebuild.
- The materialized view is a full ACID table.

Conditions for a partition-based incremental rebuild

If compaction did not run on the source table after the last rebuild, a partitions-based incremental rebuild occurs under the following conditions:

- The source table is an insert-only transactional tables.
- The materialized view is an insert-only transactional table.

  If the materialized view is a full acid, a partition-based incremental rebuild is not attempted. Instead, a record-based incremental rebuild is attempted.
- The materialized view definition includes any supported aggregation.
- You are using a Unified Analytics Virtual Warehouse with the Hive engine.

Purposely using a stale materialized view

When materialized view data becomes stale, or when the optimizer cannot determine the data freshness, which is the case if you use external tables, you might want to use the materialized view anyway:

About this task

You can use the materialized view by performing these tasks:

- Schedule the materialized view for rebuilding. For example, schedule a rebuild to occur every x minutes.
- Adjust the rewriting time window to use stale or possibly stale data for a period of time. For example, schedule the window within which to use stale data for x + y minutes.

Procedure

1. Create a scheduled query to invoke the rebuild statement every 10 minutes.

```
CREATE SCHEDULED QUERY scheduled_rebuild
EVERY 10 MINUTES AS
ALTER MATERIALIZED VIEW mv_recently_hired REBUILD;
```

2. Define the window of time for using stale data, 13 minutes.

```
SET hive.materializedview.rewriting.time.window=13min;
```

Creating and using a partitioned materialized view

When creating a materialized view, you can partition selected columns to improve performance. Partitioning separates the view of a table into parts, which often improves query rewrites of partition-wise joins of materialized views with tables or other materialized views.

About this task

This task assumes you created a materialized view of the emps and depts tables and assumes you created these tables. The emps table contains the following data:

empid	deptno	name	salary	hire_date
10001	101	jane doe	250000	2018-01-10
10005	100	somporn klailee	210000	2017-12-25
10006	200	jeiranan thongnopneua	175000	2018-05-05

The depts table contains the following data:

deptno	deptname	locationid
100	HR	10
101	Eng	11
200	Sup	20

In this task, you create two materialized views: one partitions data on department; the other partitions data on hire date. You select data, filtered by department,from the original table, not from either one of the materialized views. The explain plan shows that Hive rewrites your query for efficiency to select data from the materialized view that partitions data by department. In this task, you also see the effects of rebuilding a materialized view.

Procedure

1. Create a materialized view of the emps table that partitions data into departments.

```
CREATE MATERIALIZED VIEW partition_mv_1 PARTITIONED ON (deptno)
AS SELECT hire_date, deptno FROM emps WHERE deptno > 100 AND deptno < 200;
```

2. Create a second materialized view that partitions the data on the hire date instead of the department number.

```
CREATE MATERIALIZED VIEW partition_mv_2 PARTITIONED ON (hire_date)
  AS SELECT deptno, hire_date FROM emps where deptno > 100 AND deptno < 2
00;
```

3. Generate an extended explain plan by selecting data for department 101 directly from the emps table without using the materialized view.

```
EXPLAIN EXTENDED SELECT deptno, hire_date FROM emps_a where deptno = 101;
```

The explain plan shows that Hive rewrites your query for efficiency, using the better of the two materialized views for the job: partition_mv_1.

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| OPTIMIZED SQL: SELECT CAST(101 AS INTEGER) AS `deptno`, `hire_date` |
```

```
| FROM `default`.`partition_mv_1`                    |
| WHERE 101 = `deptno`                               |
| STAGE DEPENDENCIES:                                |
|   Stage-0 is a root stage
...
```

4. Correct Jane Doe's hire date to February 12, 2018, rebuild one of the materialized views, but not the other, and compare contents of both materialized views.

```
INSERT INTO emps VALUES (10001,101,'jane doe',250000,'2018-02-12');
ALTER MATERIALIZED VIEW partition_mv_1 REBUILD;
SELECT * FROM partition_mv_1 where deptno = 101;
SELECT * FROM partition_mv_2 where deptno = 101;
```

The output of selecting the rebuilt partition_mv_1 includes the original row and newly inserted row because INSERT does not perform in-place updates (overwrites).

```
+----------------------------+------------------------+
| partition_mv_1.hire_date  | partition_mv_1.deptno  |
+----------------------------+------------------------+
| 2018-01-10 00:00:00.0     | 101                    |
| 2018-02-12 00:00:00.0     | 101                    |
+----------------------------+------------------------+
```

The output from the other partition is stale because you did not rebuild it:

```
+------------------------+----------------------------+
| partition_mv_2.deptno  | partition_mv_2.hire_date  |
+------------------------+----------------------------+
| 101                    | 2018-01-10 00:00:00.0     |
+------------------------+----------------------------+
```

5. Create a second employees table and a materialized view of the tables joined on the department number.

```
CREATE TABLE emps2 TBLPROPERTIES AS SELECT * FROM emps;
CREATE MATERIALIZED VIEW partition_mv_3 PARTITIONED ON (deptno) AS
  SELECT emps.hire_date, emps.deptno FROM emps, emps2
  WHERE emps.deptno = emps2.deptno
  AND emps.deptno > 100 AND emps.deptno < 200;
```

6. Generate an explain plan that joins tables emps and emps2 on department number using a query that omits the partitioned materialized view.

```
EXPLAIN EXTENDED SELECT emps.hire_date, emps.deptno FROM emps, emps2
  WHERE emps.deptno = emps2.deptno
  AND emps.deptno > 100 AND emps.deptno < 200;
```

The output shows that Hive rewrites the query to use the partitioned materialized view partition_mv_3 even though your query omitted the materialized view.

7. Verify that the partition_mv_3 sets up the partition for deptno=101 for partition_mv_3.

```
SHOW PARTITIONS partition_mv_3;
```

Output is:

```
+-------------+
|  partition  |
+-------------+
| deptno=101  |
+-------------+
```

Materialized view commands

The syntax and examples of materialized view commands provide details for using materialized views in Unified Analytics.

ALTER MATERIALIZED VIEW REBUILD

You must rebuild the materialized view to keep it up-to-date when changes to the data occur.

Syntax

```
ALTER MATERIALIZED VIEW [db_name.]materialized_view_name REBUILD;
```

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by the name of the materialized view in dot notation.

Description

A rewrite of a query based on a stale materialized view does not occur automatically. If you want a rewrite of a stale or possibly stale materialized view, you can force a rewrite. For example, you might want to use the contents of a materialized view of a non-transactional table because the freshness of such a table is unknown. To enable rewriting of a query based on a stale materialized view, you can run the rebuild operation periodically and set the following property: hive.materializedview.rewriting.time.window. For example, SET hive.materializedview.rewriting.time.window=10min;

Example

ALTER MATERIALIZED VIEW mydb.mv1 REBUILD;

ALTER MATERIALIZED VIEW REWRITE

You can change the behavior of Hive to enable or disable the rewriting of queries based on a particular materialized view.

Syntax

```
ALTER MATERIALIZED VIEW [db_name.]materialized_view_name ENABLE|DISABLE REWR
ITE;
```

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by the name for the materialized view in dot notation.

Description

To optimize performance, by default, Hive rewrites a query based on materialized views. You can change this behavior to manage query planning and execution manually. By setting the hive.materializedview.rewriting global property, you can manage query rewriting based on materialized views for all queries.

Example

ALTER MATERIALIZED VIEW mydb.mv1 DISABLE REWRITE;

CREATE MATERIALIZED VIEW

If you are familiar with the CREATE TABLE AS SELECT (CTAS) statement, you can quickly master how to create a materialized view.

Syntax

```
CREATE MATERIALIZED VIEW [IF NOT EXISTS] [db_name.]materialized_view_name
   [DISABLE REWRITE]
   [COMMENT materialized_view_comment]
   [PARTITIONED ON (column_name, ...)]
   [
     [ROW FORMAT row_format]
     [STORED AS file_format]
        | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (ser
de_property_name=serde_property_value, ...)]
   ]
   [LOCATION hdfs_path]
   [TBLPROPERTIES (tbl_property_name=tbl_property_value, ...)]
   AS
   <query>;
```

Required Parameters

QUERY

The query to execute for results that populate the contents of the materialized view.

Optional Parameters

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by a name, unique among materialized view names, for the materialized view. The name must conform to Apache Hive specifications for a table name, including case-insensitive alphanumeric and underscore characters.

MATERIALIZED_VIEW_COMMENT

A string literal enclosed in single quotation marks.

COLUMN_NAME

A key that determines how to do the partitioning, which divides the view of the table into parts.

'STORAGE.HANDLER.CLASS.NAME'

The name of a storage handler, such as org.apache.hadoop.hive.druid.DruidStorageHandler, that conforms to the Apache Hive specifications for storage handlers in a table definition that uses the STORED BY clause. When not specified, Hive uses the default hive.materializedview.fileformat.

SERDE_PROPERTY_NAME

A property supported by SERDEPROPERTIES that you specify as part of the STORED BY clause. The property is passed to the serde provided by the storage handler. When not specifed, Hive uses the default hive.materializedview.serde.

SERDE_PROPERTY_VALUE

A value of the SERDEPROPERTIES property.

HDFS_PATH

The location on the file system for storing the materialized view.

TBL_PROPERTY_NAME

A key that conforms to the Apache Hive specification for TBLPROPERTIES keys in a table.

TBL_PROPERTY_VALUE

The value of a TBLPROPERTIES key.

Usage

The materialized view creation statement meets the criteria of being atomic: it does not return incomplete results. By default, the optimizer uses materialized views to rewrite the query. You can store a materialized view in an

external storage system using the STORED AS clause followed by a valid storage handler class name. You can set the DISABLE REWRITE option to alter automatic rewriting of the query at materialized view creation time.

Examples

```
CREATE MATERIALIZED VIEW druid_t
  STORED AS 'org.apache.hadoop.hive.druid.DruidStorageHandler'
  AS SELECT a, b, c
  FROM src;
```

```
CREATE MATERIALIZED VIEW mv4
  LOCATION '/user/csso_max'
  AS SELECT empid, deptname, hire_date
  FROM emps JOIN depts
  ON (emps.deptno = depts.deptno)
  WHERE hire_date >= '2017-01-01';
```

DESCRIBE EXTENDED and DESCRIBE FORMATTED

You can get extensive formatted and unformatted information about a materialized view.

Syntax

```
DESCRIBE [EXTENDED | FORMATTED] [db_name.]materialized_view_name;
```

DB_NAME

The database name.

MATERIALIZED_VIEW_NAME

The name of the materialized view.

Examples

Get summary, details, and formatted information about the materialized view in the default database and its partitions.

```
DESCRIBE FORMATTED default.partition_mv_1;
```

Example output is:

col_name	data_type	comment
# col_name	data_type	comment
name	varchar(256)	
	NULL	NULL
# Partition Information	NULL	NULL
# col_name	data_type	comment
deptno	int	
	NULL	NULL
# Detailed Table Information	NULL	NULL
Database:	default	NULL
OwnerType:	USER	NULL
Owner:	hive	NULL
CreateTime:	Wed Aug 22 19:46:08 UTC 2018	NULL

col_name	data_type	comment
LastAccessTime:	UNKNOWN	NULL
Retention:	0	NULL
Location:	hdfs://myserver:8020/warehouse/ tables pace/managed/hive/partition_mv_1	NULL
Table Type:	MATERIALIZED_VIEW	NULL
Table Parameters:	NULL	NULL
	COLUMN_STATS_ACCURATE	{\"BASIC_STATS\":\"true\"}
	bucketing_version	2
	numFiles	2
	numPartitions	2
	numRows	4
	rawDataSize	380
	totalSize	585
	transient_lastDdlTime	1534967168
	NULL	NULL
# Storage Information	NULL	NULL
SerDe Library:	org.apache.hadoop.hive.ql.io.orc.OrcSerde	NULL
InputFormat:	org.apache.hadoop.hive.ql.io.orc.OrcInputFor mat	NULL
OutputFormat:	org.apache.hadoop.hive.ql.io.orc.OrcOutputFo rmat	NULL
Compressed:	No	NULL
Num Buckets:	-1	NULL
Bucket Columns:	[]	NULL
Sort Columns:	[]	NULL
	NULL	NULL
# Materialized View Information	NULL	NULL
Original Query:	SELECT hire_date, deptno FROM emps W HERE deptno > 100 AND deptno < 200	NULL
Expanded Query:	SELECT `hire_date`, `deptno` FROM (S ELECT `emps`.`hire_date`, `emps`.`dept no` FROM `default`.`emps` WHERE `e mps`.`deptno` > 100 AND `emps`.`deptno ` < 200) `default.partition_mv_1`	NULL
Rewrite Enabled:	Yes	NULL
Outdated for Rewriting:	No	NULL

DROP MATERIALIZED VIEW

You can avoid making a table name unusable by dropping a dependent materialized view before dropping a table.

Syntax

```
DROP MATERIALIZED VIEW [db_name.]materialized_view_name;
```

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by a name for the materialized view in dot notation.

Description

Dropping a table that is used by a materialized view is not allowed and prevents you from creating another table of the same name. You must drop the materialized view before dropping the tables.

Example

DROP MATERIALIZED VIEW mydb.mv1;

SHOW MATERIALIZED VIEWS

You can list all materialized views in the current database or in another database. You can filter a list of materialized views in a specified database using regular expression wildcards.

Syntax

```
SHOW MATERIALIZED VIEWS [IN db_name];
```

DB_NAME

The database name.

'IDENTIFIER_WITH_WILDCARDS'

The name of the materialized view or a regular expression consisting of part of the name plus wildcards. The asterisk and pipe (* and |) wildcards are supported. Use single quotation marks to enclose the identifier.

Examples

SHOW MATERIALIZED VIEWS;

```
+-----------------+--------------------+-----------------+
|     mv_name     |  rewrite_enabled   |      mode       |
+-----------------+--------------------+-----------------+
| # MV Name       | Rewriting Enabled  | Mode            |
| partition_mv_1  | Yes                | Manual refresh  |
| partition_mv_2  | Yes                | Manual refresh  |
| partition_mv_3  | Yes                | Manual refresh  |
+-----------------+--------------------+-----------------+
```

SHOW MATERIALIZED VIEWS '*1';

```
+-----------------+--------------------+-----------------+
|     mv_name     |  rewrite_enabled   |      mode       |
+-----------------+--------------------+-----------------+
| # MV Name       | Rewriting Enabled  | Mode            |
| partition_mv_1  | Yes                | Manual refresh  |
|                 | NULL               | NULL            |
+-----------------+--------------------+-----------------+
```

Materialized view recommendations introduction

Materialized views can improve query performance greatly, but deciding which query results to materialize is difficult. You can use the materialized view recommender command-line interface to help you decide which query results to materialize.

The recommender design is based on *Agrawal et al. - Automated Selection of Materialized Views and Indexes for SQL Databases (VLDB 2000)*. The recommender takes a set of Hive queries as input and creates materialized views as output by exploring many table subsets and syntactical variants for each view candidate.

Typically, instead of considering a single, or just a few queries for materializing, you are considering an entire workload. Even the most experienced database experts have difficulty. There are many factors to consider, such as view maintenance and effectiveness. Making a bad decision about what to materialize can make performance worse.

You provide a set of queries or explain plans to the recommender, and then run the Workload Insights (wi) script to get recommendations. You can get recommendations online or offline, depending on the type of input you have:

- Online: input = queries only
- Offline: input = explain plans

If you have explain plans to provide as input to the Workload Insights (wi) script, you can run the script offline; otherwise, you have to provide the HiveServer endpoint for example, jdbc:hive2://myhost:10000, to the script. In lieu of explain plans, the recommender needs to gather information about the query, such as the number of rows of each query, how long the query ran, and other statistics, from HiveServer.

The following diagram shows the input and output of the recommender:

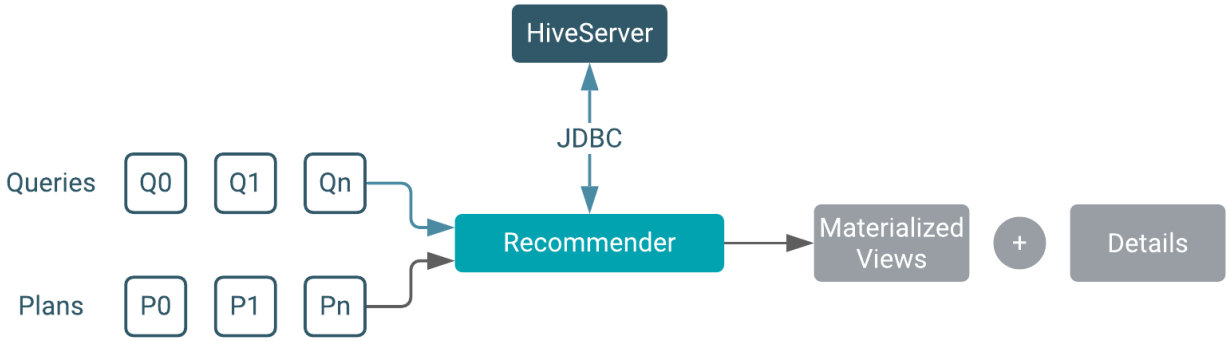

How much detail you receive in a recommendation depends on your configuration. Details include the number of recommendations and other information, such as why a particular view is recommended and how to exploit the view.

Workload Insights input requirements

You must have one of the following inputs to the Workload Insights (wi) script:

- One or more directories containing one or more select and aggregation queries in .sql files and multiple queries in files, separated by a semicolon (;).

  A .sql file name suffix is required.
- One or more directories containing an explain plan in .json files. A .json file name suffix is required.

Using the recommender in local mode

You can use the recommender on a local computer for learning and testing. You can use a Mac laptop, for example. First, from a Cloudera Data Warehouse Virtual Warehouse that enables Unified Analytics, you download software that runs a mini HiveServer in local mode. Next, you provide a set of queries or explain plans to the recommender. Finally, you execute the Workload Insights (wi) script to get pseudo-recommendations.

Setting up a mini HiveServer

You follow a step-by-step procedure to get a docker image of HiveServer (HS2) and example files to simulate working with HiveServer in a cluster. In local mode, you do not get actual recommendations, but you experiment with and understand recommendations using this mode without incurring any cloud costs.

About this task

You download the workload-insights archive and unpack it. You run one docker command to start HiveServer and another to load the TPC-DS schema. Example query files, based on the TPC-DS Benchmark DDL queries, are included in the package. No actual data is needed.

Before you begin

- Check that Java 8 or later is installed.
- Install Docker (https://docs.docker.com/get-docker/).
- Check that Docker is working: docker run hello-world

Procedure

1. On the **Overview** page under Virtual Warehouses, click options ⋮ in a Virtual Warehouse that enables Unified Analytics.

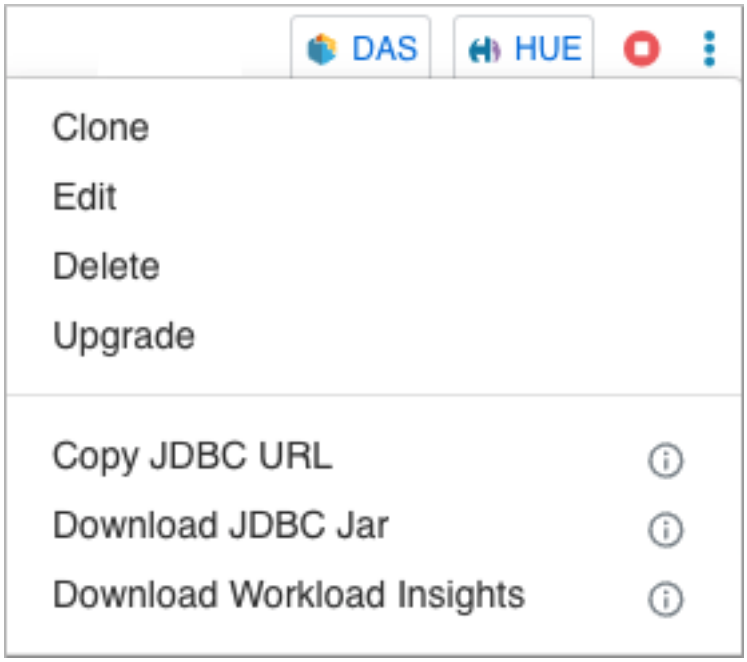

2. Click Download Workload Insights to download the workload-insights archive.
3. On your file system, unpack the archive.
4. Set environment variable WI_HOME="$PWD".
   For example, on the command line of a terminal:

```
export WI_HOME="$PWD"
```

5. Set environment variable *WI_CLASSPATH*. WI_CLASSPATH="*:./lib/*:$WI_HOME/target/*:$WI_HOME/target/lib/*"
6. Start up a HiveServer (HS2) container named mini-hs2 that listens on port 10000.

```
docker run -p "10000:10000" -d --name mini-hs2 zabetak/hs2-embedded:1.0.
2.7.2.3.0-220
```

7. Load the TPC-DS schema required for running examples.

```
docker exec mini-hs2 schema-load tpcds
```

   Ignore any log4j warning messages.

8. Run the recommender.

Getting wi script help

To get pseudo-recommendations for learning, or testing purposes, you need to know the Workload Insights (wi) script options. You run your first wi script to get online help.

Procedure

1. Go to the github directory where you downloaded and built the project.
2. Run the wi script with no options to get online help about script usage and options:

```
$ ./wi
```

Output looks something like this:

```
usage: wi -i <file_1,...,file_N> [-o <directory>] [-c <configfile>] [-u
<url>] [-n <username>] [-p
 <password>] [-t <threshold>] [-f <format>] [-r <recommendation>] [-l <le
vel>][-d]
```

-i,--input <file_1,...,file_N>	one or more .sql/.json files or a directory (version Y)
-o,--output <directory>	output directory
-c,--config <configfile>	path to configuration .properties) file
-u,--url <url>	JDBC url to connect to
-n,--username <username>	username to connect as
-p,--password <password>	password to connect as
-t,--tables-threshold <threshold>	the threshold controlling above which a table subset is considered interesting if it is estimated to contribute to total workload cost above this threshold (between 0 and 1). The default is 0.1 (10% of total workload cost)

-f,--format <format>	output format TEXT(Default)/JSON
-r,--recommendation <recommendation>	the kind of recommendations to i nclude in the report: - SPJ for SELECT-PROJECT-JOIN reco mmendations - SPJA for SELECT-PROJECT-JOIN-AGG REGATE recommendations - ALL for both SPJ and SPJA recomme ndations (DEFAULT)
-l,--level <level>	the level of detail to display in the report: - MINIMAL displays only the view recommendations - DEFAULT displays recommendations along with basic information ab out the query workload - FULL displays all information ava ilable in the report (useful for debugging purposes)
-d, --delete	suggest which materialized views, not used by a given workload could be deleted. Requires JDBC information to be provided. Available in vers ion Y.

```
usage: wi -i <directory> [-o <directory>] [-c <configfile>] [-u <url>] [-n
 <username>] [-p <password>] [-t <threshold>] [-f <format>] [-r <recommend
ation>] [-l <level>]
 -i,--input <directory>                 input directory with .sql/.json
 files
 -o,--output <directory>                output directory
 -c,--config <configfile>               path to configuration (.properti
es) file
 -u,--url <url>                         JDBC url to connect to
 -n,--username <username>               username to connect as
 -p,--password <password>               password to connect as
 -t,--tables-threshold <threshold>      the threshold controlling above wh
ich a table subset is considered interesting if it is estimated to contr
ibute to total workload cost above this threshold (between 0 and 1). The
 default is 0.1 (10% of total workload cost)
 -f,--format <format>                   output format TEXT(Default)/JSON
 -r,--recommendation <recommendation>   the kind of recommendations to inc
lude in the report:
                                        - SPJ for SELECT-PROJECT-JOIN re
commendations
                                        - SPJA for SELECT-PROJECT-JOIN-AGG
REGATE recommendations
                                        - ALL for both SPJ and SPJA recom
mendations (DEFAULT)
```

```
 -l,--level <level>                     the level of detail to display in
 the report:
                                        - MINIMAL displays only the view
recommendations
                                        - DEFAULT displays recommendations
 along with basic information about the query workload
                                        - FULL displays all information
 available in the report (useful for debugging purposes)
```

3. Try the recommender examples.
   A text report that has the following information appears:
   - View recommendations for the workload.
   - Queries in the workload, along with the recommendations for each query (if available), and other information.
4. Stop the mini-hs2.

```
docker stop mini-hs2
```

5. Remove the mini-hs2.

```
docker rm mini-hs2
```

Running Recommender examples

The workload-insights archive includes sample queries and explain plans that demonstrate the recommender. You gain an understanding of the recommender by running examples.

About this task

In local mode, you do not get actual recommendations, but you experiment with and understand recommendations using this mode without incurring any cloud costs.The recommender examples you can run use one or both of the following queries as input:

QueryA

```
SELECT sr_item_sk, sum(sr_net_loss) AS net_loss
FROM store_returns
INNER JOIN reason ON sr_reason_sk = r_reason_sk
WHERE r_reason_desc = 'Found a better price in a store'
GROUP BY sr_item_sk
```

Query A asks these questions:

- Which items were returned because the customer found a better price?
- How much money did the company lose on each returned item?

QueryB

```
SELECT sr_item_sk, sum(sr_net_loss) as net_loss, avg(ss_sales_price) as pric
e
FROM store_sales
INNER JOIN store_returns
  ON ss_item_sk=sr_item_sk AND ss_ticket_number=sr_ticket_number
INNER JOIN reason
  ON sr_reason_sk = r_reason_sk
WHERE r_reason_desc = 'Found a better price in a store'
GROUP BY sr_item_sk
ORDER BY net_loss DESC NULLS LAST, price DESC NULLS LAST
LIMIT 10
```

QueryB asks these questions:

- Which items were returned because the customer found a better price?
- How much money did the company lose on each returned item?
- What was the average selling price of return items?
- What are the top ten items causing the biggest loss?

Before you begin

- The mini-hs2 docker image is running.
- You downloaded and unpacked the workload-insights archive, and it includes the following sample queries that you need to provide as input to the Workload Insights (wi) script:
  - Directories containing select and aggregation queries in .sql files

    A .sql file name suffix is required.
  - Directories containing explain plans in .json files

    A .json file name suffix is required.

Running the Hello world example

You use the minimum set of required parameters, and get a result that is not interesting. From this example, you learn what to expect in the output when you use the minimum set of required parameters.

About this task

In the input directory workload-insights/demo/queries, a file named queryA.sql contains a single query.

Procedure

1. Go to the workload-insights directory

   The Workload Insights (wi) script is located in this directory.
2. Run the script using the -i option to specify file queryA.sql as the input to the recommender.

```
./wi -i queries/queryA.sql -u jdbc:hive2://localhost:10000
```

Output looks something like this:

```
2021-06-10 15:17:37,461 INFO c.c.i.a.m.t.JdbcWorkloadReader Computing plan
 for queryA from JDBC connection
...
Recommended views:
-- materialized_view_1:
CREATE MATERIALIZED VIEW `materialized_view_1` AS
SELECT `store_returns`.`sr_item_sk`, SUM(`store_returns`.`sr_net_loss`) AS
 `$f1`
FROM `default`.`reason`, `default`.`store_returns`
WHERE `store_returns`.`sr_reason_sk` = `reason`.`r_reason_sk` AND 'Found a
 better price in a store' = `reason`.`r_reason_desc`
GROUP BY `store_returns`.`sr_item_sk`;
-----------------------------------------------------
Query details:
-- queryA:SELECT `t0`.`sr_item_sk`, SUM(`t0`.`sr_net_loss`) AS `$f1` FROM
 (SELECT `sr_item_sk`, `sr_reason_sk`, `sr_net_loss` FROM `default`.`sto
re_returns` WHERE `sr_reason_sk` IS NOT NULL) AS `t0` INNER JOIN (SELECT
 `r_reason_sk` FROM `default`.`reason` WHERE `r_reason_desc` = 'Found a
better price in a store' AND `r_reason_sk` IS NOT NULL) AS `t2` ON `t0`.
`sr_reason_sk` = `t2`.`r_reason_sk` GROUP BY `t0`.`sr_item_sk`
 -- RECOMMENDATION : materialized_view_1
```

```
-----------------------------------------------------
```

The result is identical to the input query. The output contains a list of recommendations followed by the queries in the workload. By default, for each query, the materialized views matching the query appear.

Running the exclude aggregations example

If you are knowledgeable about your workload and want to exclude views with aggregations for maintenance reasons, you need to know about the SPJ (select-project-join) recommendation mode.

About this task

In the input directory workload-insights/demo/queries, a file named queryA.sql contains queryA. QueryA contains the GROUP BY aggregation clause, which you want the recommender to exclude from the view..

Procedure

1. Go to the workload-insights directory.

   The Workload Insights (wi) script is located in this directory.
2. Run the script using the -r SPJ option to exclude aggregations from recommended views.

```
./wi -i queries/queryA.sql -u jdbc:hive2://localhost:10000 -r SPJ
```

Output looks something like this:

```
...
Recommended views:
-- materialized_view_0:
CREATE MATERIALIZED VIEW `materialized_view_0` AS
SELECT `store_returns`.`sr_item_sk`, `store_returns`.`sr_reason_sk`, `s
tore_returns`.`sr_net_loss`, `store_returns`.`sr_reason_sk` AS `sr_reaso
n_sk0`
FROM `default`.`reason`,`default`.`store_returns`
WHERE `store_returns`.`sr_reason_sk` = `reason`.`r_reason_sk` AND 'Found
 a better price in a store' = `reason`.`r_reason_desc`;
-----------------------------------------------------
Query details:
-- queryA:SELECT `t0`.`sr_item_sk`, SUM(`t0`.`sr_net_loss`) AS `$f1` FROM
(SELECT `sr_item_sk`, `sr_reason_sk`, `sr_net_loss` FROM `default`.`stor
e_returns` WHERE `sr_reason_sk` IS NOT NULL) AS `t0` INNER JOIN (SELECT
`r_reason_sk` FROM `default`.`reason` WHERE `r_reason_desc` = 'Found a b
etter price in a store' AND `r_reason_sk` IS NOT NULL) AS `t2` ON `t0`.`
sr_reason_sk` = `t2`.`r_reason_sk` GROUP BY `t0`.`sr_item_sk`
-- RECOMMENDATION : materialized_view_0
-----------------------------------------------------
```

The recommender runs faster by excluding aggregations because the recommender explores a smaller search space. Similarly, the user can request to obtain only recommendations with aggregations using the SPJA mode. Using the default ALL option, you would get the same result as above. ALL includes SPJ for SELECT-PROJECT-JOIN recommendations and SPJA for SELECT-PROJECT-JOIN-AGGREGATE.

Running the level of detail examples

You see what the level of detail options, -l MINIMAL, -l DEFAULT, and -l FULL do byy running these examples. The recommended materialized view is identical to the input query regardless of the level of detail option you use.

About this task

Changing the level of detail does not affect the number or structure of recommendations but provides more insights for each query/recommendation in the case of FULL, and less in the case of MINIMAL.

In the input directory src/test/resources/demo/queries/scenario02, a file named queryA.sql contains queryA.

Procedure

1. Go to the workload-insights directory.

   The Workload Insights (wi) script is located in this directory.

2. Run the script to get a recommendation using the -l MINIMAL option.

```
./wi -i queries/queryA.sql -u jdbc:hive2://localhost:10000 -l MINIMAL
```

   Only recommendations appear. You can easily copy and paste the CREATE statements for use in another CLI, for example:

```
SUMMARY (END) ################
Recommended views:
CREATE MATERIALIZED VIEW `materialized_view_1` AS
SELECT `store_returns`.`sr_item_sk`, SUM(`store_returns`.`sr_net_loss`) AS
 `$f1`
FROM `default`.`reason`,`default`.`store_returns`
WHERE `store_returns`.`sr_reason_sk` = `reason`.`r_reason_sk` AND 'Found a
 better price in a store' = `reason`.`r_reason_desc`
GROUP BY `store_returns`.`sr_item_sk`;
-------------------------------------------------------
```

3. Run the script using the -l FULL option.

```
./wi -i queries/queryA.sql -u jdbc:hive2://localhost:10000 -l FULL
```

   Recommendations are:

```
…
Recommended views:
-- materialized_view_1:
CREATE MATERIALIZED VIEW `materialized_view_1` AS
SELECT `store_returns`.`sr_item_sk`, SUM(`store_returns`.`sr_net_loss`) AS
 `$f1`
FROM `default`.`reason`,`default`.`store_returns`
WHERE `store_returns`.`sr_reason_sk` = `reason`.`r_reason_sk` AND 'Found a
 better price in a store' = `reason`.`r_reason_desc`
GROUP BY `store_returns`.`sr_item_sk`;
-- QUERIES : queryA
-- POTENTIAL : 0.9902040816326532
-------------------------------------------------------
Total improvement: 0.9902040816326532
-------------------------------------------------------
```

   From the following verbose details in the output, you can understand why a recommendation was selected, or not, for a particular query. You might use these details to troubleshoot problems with the recommender.

```
Query details:
-- queryA:SELECT `t0`.`sr_item_sk`, SUM(`t0`.`sr_net_loss`) AS `$f1` FROM
 (SELECT `sr_item_sk`, `sr_reason_sk`, `sr_net_loss` FROM `default`.`sto
re_returns` WHERE `sr_reason_sk` IS NOT NULL) AS `t0` INNER JOIN (SELECT
 `r_reason_sk` FROM `default`.`reason` WHERE `r_reason_desc` = 'Found a
better price in a store' AND `r_reason_sk` IS NOT NULL) AS `t2` ON `t0`.
`sr_reason_sk` = `t2`.`r_reason_sk` GROUP BY `t0`.`sr_item_sk`
-- RECOMMENDATION : materialized_view_1
-- PLAN :
    DASAggregate(group=[{0}], agg#0=[sum($2)])
      DASJoin(condition=[=($1, $3)], joinType=[inner])
        DASProject(sr_item_sk=[$2], sr_reason_sk=[$8], sr_net_loss=[$19])
          DASFilter(condition=[IS NOT NULL($8)])
```

```
            DASTableScan(table=[[default, store_returns]])
        DASProject(r_reason_sk=[$0])
          DASFilter(condition=[AND(=($2, 'Found a better price in a store
'), IS NOT NULL($0))])
            DASTableScan(table=[[default, reason]])
 -- BT: default.store_returns is a base table
  -- Row Count: 1.0|Row size: 208.0
 -- BT: default.reason is a base table
  -- Row Count: 1.0|Row size: 332.0
 -- IMPROVEMENT : 0.9902040816326532
-----------------------------------------------------
Interesting subsets:
-- [[default, reason], [default, store_returns]]:1.0
  "recommended_views" : [ {
    "name" : "materialized_view_1",
    "sql" : "CREATE MATERIALIZED VIEW `materialized_view_1` AS\nSELECT
`store_returns`.`sr_item_sk`, SUM(`store_returns`.`sr_net_loss`) AS `$f1
`\nFROM `default`.`reason`,\n`default`.`store_returns`\nWHERE `store_ret
urns`.`sr_reason_sk` = `reason`.`r_reason_sk` AND 'Found a better price in
 a store' = `reason`.`r_reason_desc`\nGROUP BY `store_returns`.`sr_item_sk
`;"
  } ],
  "query_details" : [ {
    "name" : "queryA",
    "sql" : "SELECT `t0`.`sr_item_sk`, SUM(`t0`.`sr_net_loss`) AS `$f1`\nF
ROM (SELECT `sr_item_sk`, `sr_reason_sk`, `sr_net_loss`\nFROM `default`.
`store_returns`\nWHERE `sr_reason_sk` IS NOT NULL) AS `t0`\nINNER JOIN (
SELECT `r_reason_sk`\nFROM `default`.`reason`\nWHERE `r_reason_desc` = '
Found a better price in a store' AND `r_reason_sk` IS NOT NULL) AS `t2` ON
 `t0`.`sr_reason_sk` = `t2`.`r_reason_sk`\nGROUP BY `t0`.`sr_item_sk`",
    "recommendations" : [ "materialized_view_1" ]
  } ]
```

Running the JSON output example

You see the example JSON output when you run the Workload Insights command with the -f JSON option.

About this task

In the input directory src/test/resources/demo/queries, a file named queryA.sql contains queryA.

Procedure

1. Go to the github directory where you downloaded and built the project.

   The Worload Insights (wi) script is located in this directory.
2. Run the script to get a JSON-formatted recommendation.

```
./wi -i queries/queryA.sql -u jdbc:hive2://localhost:10000 -f JSON
```

   The output is:

```
...
{
  "recommended_views" : [ {
    "name" : "materialized_view_1",
    "sql" : "CREATE MATERIALIZED VIEW `materialized_view_1` AS\nSELECT `st
ore_returns`.`sr_item_sk`, SUM(`store_returns`.`sr_net_loss`) AS `$f1`\n
FROM `default`.`reason`,\n`default`.`store_returns`\nWHERE `store_return
s`.`sr_reason_sk` = `reason`.`r_reason_sk` AND 'Found a better price in
a store' = `reason`.`r_reason_desc`\nGROUP BY `store_returns`.`sr_item_s
k`;"
  } ],
```

```
  "query_details" : [ {
    "name" : "queryA",
    "sql" : "SELECT `t0`.`sr_item_sk`, SUM(`t0`.`sr_net_loss`) AS `$f1`\n
FROM (SELECT `sr_item_sk`, `sr_reason_sk`, `sr_net_loss`\nFROM `default`
.`store_returns`\nWHERE `sr_reason_sk` IS NOT NULL) AS `t0`\nINNER JOIN
(SELECT `r_reason_sk`\nFROM `default`.`reason`\nWHERE `r_reason_desc` =
'Found a better price in a store' AND `r_reason_sk` IS NOT NULL) AS `t2`
 ON `t0`.`sr_reason_sk` = `t2`.`r_reason_sk`\nGROUP BY `t0`.`sr_item_sk`",
    "recommendations" : [ "materialized_view_1" ]
  } ]
```

Running the multiple query input example

Typically, you want the recommender to base recommendations on multiple queries. You see how to run the wi script on a directory containing multiple queries and how to interact with the recommender.

About this task

Upon completing the procedure below, you see the recommendation you get for queryA is identical to the query itself. The recommendation you get for queryB closely resembles queryA. The recommender does not find any way to consolidate queries into a meaningful recommendation because queries are not similar in any significant way.

Notice that the recommendation for queryB does not contain the ORDER BY and LIMIT clauses. Recommendations cannot contain an ORDER BY and LIMIT. Doing so could significantly reduce the applicability of the view in many queries.

In the input directory demo/queries, one file named queryA.sql contains queryA, and another file named queryB.sql contains queryB.

Procedure

1. Go to the workload-insights directory.

   The Workload Insights (wi) script is located in this directory.
2. Run the script on the directory containing multiple .sql files.

```
./wi -i "queries/queryA.sql, queries/queryB.sql" -u jdbc:hive2://localho
st:10000
```

   Recommended views are:

```
...
Recommended views:
-- materialized_view_1:
CREATE MATERIALIZED VIEW `materialized_view_1` AS
SELECT `store_sales`.`ss_item_sk`, SUM(`store_returns`.`sr_net_loss`) AS `
$f1`, SUM(`store_sales`.`ss_sales_price`) AS `$f2`, COUNT(`store_sales`.`
ss_sales_price`) AS `$f3`
FROM `default`.`reason`,
 `default`.`store_returns`,
 `default`.`store_sales`
WHERE `store_sales`.`ss_ticket_number` = `store_returns`.`sr_ticket_numb
er` AND `store_returns`.`sr_reason_sk` = `reason`.`r_reason_sk` AND `sto
re_sales`.`ss_item_sk` = `store_returns`.`sr_item_sk` AND 'Found a better
 price in a store' = `reason`.`r_reason_desc`
GROUP BY `store_sales`.`ss_item_sk`;
-- materialized_view_4:
CREATE MATERIALIZED VIEW `materialized_view_4` AS
SELECT `store_returns`.`sr_item_sk`, SUM(`store_returns`.`sr_net_loss`) AS
 `$f1`
FROM `default`.`reason`,`default`.`store_returns`
WHERE `store_returns`.`sr_reason_sk` = `reason`.`r_reason_sk` AND 'Found
 a better price in a store' = `reason`.`r_reason_desc`
```

```
GROUP BY `store_returns`.`sr_item_sk`;
-----------------------------------------------------
```

Details in the output are:

```
Query details:
-- queryA:SELECT `t0`.`sr_item_sk`, SUM(`t0`.`sr_net_loss`) AS `$f1` FROM
 (SELECT `sr_item_sk`, `sr_reason_sk`, `sr_net_loss` FROM `default`.`sto
re_returns` WHERE `sr_reason_sk` IS NOT NULL) AS `t0` INNER JOIN (SELECT
 `r_reason_sk` FROM `default`.`reason` WHERE `r_reason_desc` = 'Found a
better price in a store' AND `r_reason_sk` IS NOT NULL) AS `t2` ON `t0`.
`sr_reason_sk` = `t2`.`r_reason_sk` GROUP BY `t0`.`sr_item_sk`
-- RECOMMENDATION : materialized_view_4
-- queryB:SELECT `t0`.`sr_item_sk` AS `$f0`, SUM(`t0`.`sr_net_loss`) AS `
$f1`, SUM(`t4`.`ss_sales_price`) / COUNT(`t4`.`ss_sales_price`) AS `$f2`
 FROM (SELECT `sr_item_sk`, `sr_reason_sk`, `sr_ticket_number`, `sr_net_
loss` FROM `default`.`store_returns` WHERE `sr_item_sk` IS NOT NULL AND
`sr_ticket_number` IS NOT NULL AND `sr_reason_sk` IS NOT NULL) AS `t0` I
NNER JOIN (SELECT `r_reason_sk` FROM `default`.`reason` WHERE `r_reason_
desc` = 'Found a better price in a store' AND `r_reason_sk` IS NOT NULL)
 AS `t2` ON `t0`.`sr_reason_sk` = `t2`.`r_reason_sk` INNER JOIN (SELECT
`ss_item_sk`, `ss_ticket_number`, `ss_sales_price` FROM `default`.`store
_sales` WHERE `ss_item_sk` IS NOT NULL AND `ss_ticket_number` IS NOT NULL)
 AS `t4` ON `t0`.`sr_item_sk` = `t4`.`ss_item_sk` AND `t0`.`sr_ticket_numb
er` = `t4`.`ss_ticket_number` GROUP BY `t0`.`sr_item_sk` ORDER BY SUM(`t
0`.`sr_net_loss`) DESC, SUM(`t4`.`ss_sales_price`) / COUNT(`t4`.`ss_sale
s_price`) DESC LIMIT 10
-- RECOMMENDATION : materialized_view_1
-----------------------------------------------------
```

3. In Hive, run SHOW MATERIALIZED VIEWS to list the recommended materialized views.
4. Run EXPLAIN CBO.

```
beeline -u jdbc:hive2//localhost:10000 -n admin -p mypassword -e "EXPLAIN
 CBO `cat <path>/queryB.sql`"
```

5. Take a look at the explain plan for one of the queries.
   The output contains the name of the materialized view used in the plan.
6. Run the queryB.sql again and look at the execution time.

   Using the materialized view, execution time is likely much shorter.
7. Disable a materialized view n, for example, from being used in rewrites.

```
beeline -u jdbc:hive2://localhost:10000 -n admin -p mypassword -e "ALTER
 MATERIALIZED VIEW materialized_view_n DISABLE REWRITE"
```

8. Run EXPLAIN CBO again.
   The materialized view does not appear because it has been disabled.

Running the repetitive queries example

You see how the recommender cost model takes repetitive queries into account when generating final recommendations.

About this task

The input directory workload-insights/demo/queries contains 21 .sql files, queryA.sql, queryA1.sql - queryA19.sql, and queryB.sql.

Procedure

1. Go to the workload-insights directory.

   The Workload Insights (wi) script is located in this directory.

2. Run the script on the directory that simulates the common, repetitive query situation.

```
./wi -i "queries/queryA.sql, queries/queryAx19.sql, queries/queryB.sql" -u
 jdbc:hive2://localhost:10000 -r SPJ
```

   Recommendations are:

```
...
Recommended views:
-- merged_59:
CREATE MATERIALIZED VIEW `merged_59` AS
SELECT `store_returns`.`sr_item_sk` AS `$f2`, SUM(`store_returns`.`sr_n
et_loss`) AS `$f1`
FROM `default`.`reason`,`default`.`store_returns`
WHERE `store_returns`.`sr_reason_sk` = `reason`.`r_reason_sk` AND 'Found
 a better price in a store' = `reason`.`r_reason_desc`
GROUP BY `store_returns`.`sr_item_sk`;
-- materialized_view_0:
CREATE MATERIALIZED VIEW `materialized_view_0` AS
SELECT `store_returns`.`sr_item_sk`, `store_returns`.`sr_reason_sk`, `s
tore_returns`.`sr_ticket_number`, `store_returns`.`sr_net_loss`, `store_
returns`.`sr_reason_sk` AS `sr_reason_sk0`
FROM `default`.`reason`,`default`.`store_returns`
WHERE `store_returns`.`sr_reason_sk` = `reason`.`r_reason_sk` AND 'Foun
d a better price in a store' = `reason`.`r_reason_desc` AND `store_retur
ns`.`sr_item_sk` IS NOT NULL AND `store_returns`.`sr_ticket_number` IS NOT
 NULL;
-----------------------------------------------------
```

   Details in the output look something like this:

```
Query details:
-- queryA:SELECT `t0`.`sr_item_sk`, SUM(`t0`.`sr_net_loss`) AS `$f1` FROM
 (SELECT `sr_item_sk`, `sr_reason_sk`, `sr_net_loss` FROM `default`.`sto
re_returns` WHERE `sr_reason_sk` IS NOT NULL) AS `t0` INNER JOIN (SELECT
 `r_reason_sk` FROM `default`.`reason` WHERE `r_reason_desc` = 'Found a
better price in a store' AND `r_reason_sk` IS NOT NULL) AS `t2` ON `t0`.
`sr_reason_sk` = `t2`.`r_reason_sk` GROUP BY `t0`.`sr_item_sk`
-- RECOMMENDATION : merged_59
-- queryA1:SELECT `t0`.`sr_item_sk`, SUM(`t0`.`sr_net_loss`) AS `$f1` F
ROM (SELECT `sr_item_sk`, `sr_reason_sk`, `sr_net_loss` FROM `default`.`
store_returns` WHERE `sr_reason_sk` IS NOT NULL) AS `t0` INNER JOIN (SEL
ECT `r_reason_sk` FROM `default`.`reason` WHERE `r_reason_desc` = 'Found a
 better price in a store' AND `r_reason_sk` IS NOT NULL) AS `t2` ON `t0`.`
sr_reason_sk` = `t2`.`r_reason_sk` GROUP BY `t0`.`sr_item_sk`

...
-----------------------------------------------------
```

Cleaning up materialized views

You can delete materialized views, not used by your workload. You provide JDBC information about the queries associated with the materialized views using Workload Insights (wi) script and then use command-line JSON processor jq to delete the unused materialized views.

About this task

You get information about unused materialized views associated with a single or multiple queries, as shown in the following command:

```
./wi -i “demo/queries/all/queryA.sql, demo/queries/all/queryB.sql” -u jdbc:h
ive2://localhost:10000 -n admin -p mypassword -d
```

In the following procedure, you delete unused materialized views. First, you get and store information about the unused materialized views in a record, next you use jq to read the record of deleted materialized views, and finally you drop the materialized views.

Procedure

1. Delete materialized views associated with query03.sql from being used by your workload and store the deletion record in a file.

   ```
   ./wi -i "queries/queryA.sql, queries/queryAx19.sql, queries/queryB.sql" -u
    jdbc:hive2://localhost:10000 -n admin -p mypassword -f JSON -o out_dir -d
   ```

2. Read the deletion record.

   ```
   jq ‘.unused_views.name’ < out_dir/DELETE.json
   ```

   Output is:

   ```
   “materialized_view_2”
   “materialized_view_9”
   ```

3. Read the commands for dropping the unused materialized views.
   Output is:

   ```
   "DROP MATERIALIZED VIEW materialized_view_2;"
   "DROP MATERIALIZED VIEW materialized_view_9;"
   ```

4. Drop the materialized views using the commands above.

   ```
   beeline -u jdbc:hive2://localhost:10000 -n admin -p mypassword -e "`jq -r
    ‘.unused_views.sql’ < out_dir/DELETE.json`"
   ```

Analyzing JSON records

You can use a jq command to extract information about recommendations in JSON. Then, you can rerun the query, take a look at the plan, and see which materialized view was used, and other information.

The jq command that you can use for analyzing JSON records looks something like this:

```
jq -r `.recommended_views | .name, .queries < out_dir/ALL-FULL.json`
```

In this example, output includes the original query submitted by the user. The output also includes the optimized sql that uses the materialized view. Output looks something like this:

```
    {
      "name": "query20",
      "sql": "SELECT i_item_id,\n
       sum(cs_ext_sales_price) AS itemr
 (PARTITION BY i_class) AS revenueratio
k\n  AND i_category IN ('Shoes',\n
ld_date_sk = d_date_sk\n  AND d_date BE
GROUP BY i_item_id,\n        i_item_de
 BY i_category,\n        i_class,\n
      "recommendations": [
        "materialized_view_9"
      ],
      "optimized_sql": "SELECT `i_item_
* 100 / (SUM(`$f5`) OVER (PARTITION BY
\nFROM `materialized_view_9`\nORDER BY
m_desc` NULLS LAST, `$f5` * 100 / (SUM(
UNDED FOLLOWING)) NULLS LAST\nLIMIT 100
      "plan": "DASSortLimit(sort0=[$2],
=[ASC], dir3=[ASC], dir4=[ASC], fetch=[
s=[$3], i_current_price=[$2], itemreven
Y $3 ROWS BETWEEN UNBOUNDED PRECEDING A
sum($3)])\n      DASJoin(condition=[=($
```

This optimized SQL above matches the information in the plan.

Speeding up queries using BI mode

You can use BI mode to automatically rewrite incoming queries to be answered approximately using Apache DataSketches. BI mode can be useful when using data visualization BI tools such as Tableau.

About this task

Queries of very large data sets with a number of distinct values often take too long to return results. If you can accept approximate results, using DataSketch algorithms can save significant time. BI mode integrated with DataSketches accelerates query execution while decreasing resource utilization. BI mode can rewrite a wide range of SQL analytical queries, including COUNT(DISTINCT), CUME_DIST, RANK, and NTILE.

When you enable BI mode to use Apache DataSketches approximations, a materialized view automatically calls DataSketches for some types of operations. You can pre-compute data sketches using materialized views, and then rely on Unified Analytics BI mode to rewrite algorithms to answer queries directly from those materialized views. This accelerates query execution by orders of magnitude without any change to your original queries.

Before you begin

- Choose an authorization model.
- Configure authenticated users for querying Hive through JDBC or ODBC driver. For example, set up a Ranger policy.

Get the JDBC driver

To query, analyze, and visualize data stored in Cloudera Data Warehouse Unified Analytics, you must download the Unified Analytics JDBC driver to your cluster and provide it to your client. You can use the latest version of drivers provided by Cloudera to connect Unified Analytics to Business Intelligence (BI) tools.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Go to the Resource and Downloads tile, click See More Unified Analytics JDBC Driver .
   This action downloads the JAR (driver) file to your local system.

Connect to Unified Analytics with a BI tool

After you downloaded the JDBC driver, you must set up a BI tool such as Tableau to connect with the Unified Analytics engine.

Before you begin

Before you can use Tableau with an Impala/Hive Virtual Warehouses, you must populate your Database Catalog with sample data when you create it. You must also create a Impala/Hive Virtual Warehouse and enable Unified Analytics, which is configured to connect to the Database Catalog that is populated with data.

Procedure

1. Click ⋮ on the Impala Virtual Warehouse tile.
2. Select Copy Unified Analytics JDBC URL.
3. Paste the copied JDBC URL into a text file. It should look similar to the following:

```
jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<dwx.company.co
m>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
```

4. From the text file where you just pasted the URL, copy the host name from the JDBC URL to your system's clipboard. For example, in the URL shown in Step 3, the host name is:

```
<your-virtual-warehouse>.<your-environment>.<yourdomain.com>
```

5. Start Tableau and navigate to ConnectMore…Cloudera Hadoop :

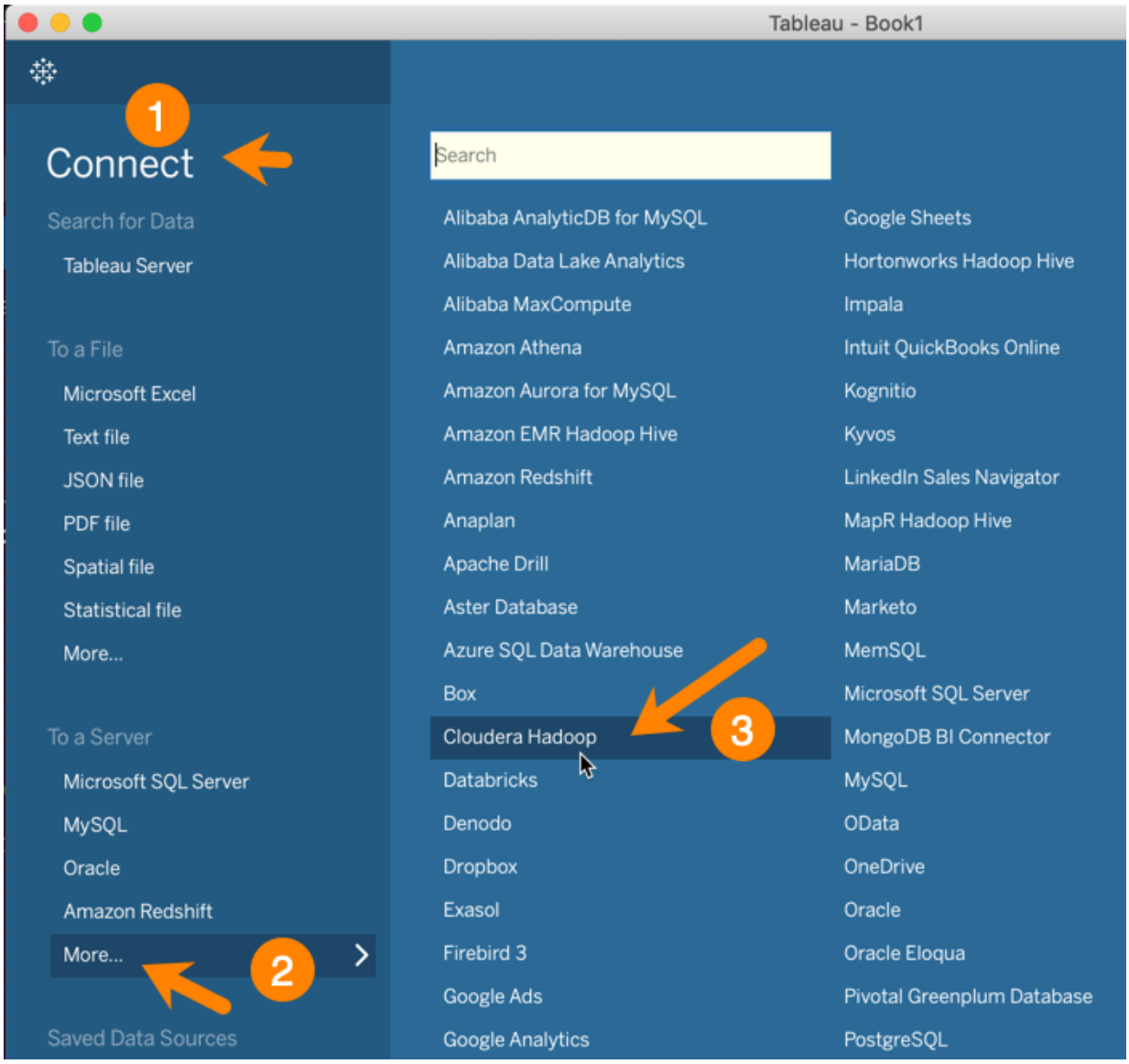

This launches the Cloudera Hadoop dialog box.

6. In the Tableau Cloudera Hadoop dialog box, paste the host name you copied to your clipboard in Step 4 into the Server field:

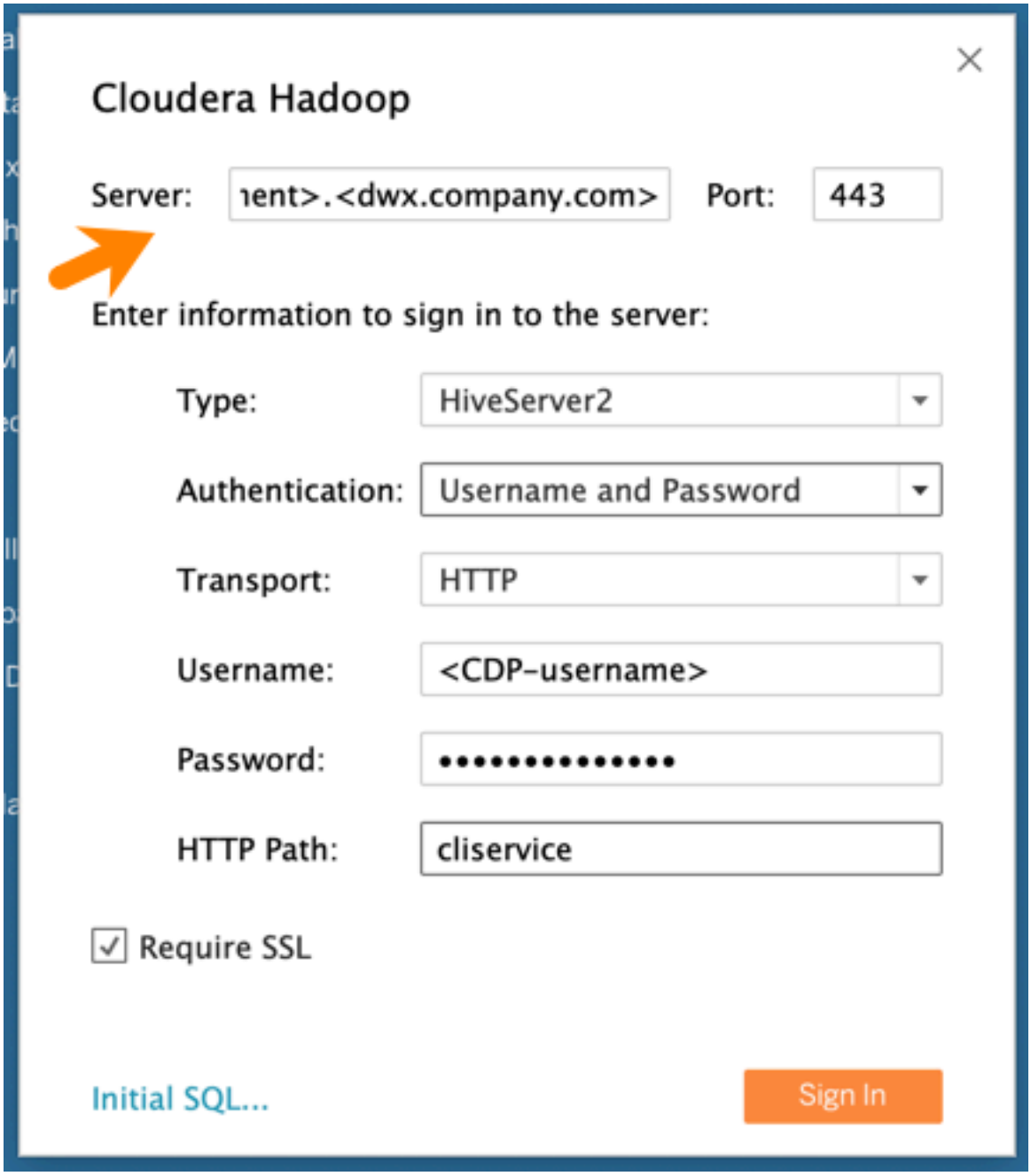

7. Then in the Tableau Cloudera Hadoop dialog box, set the following other options:
   - Port: 443
   - Type: HiveServer2
   - Authentication: Username and Password
   - Transport: HTTP
   - Username: Username you use to connect to the Cloudera Data Warehouse service.
   - Password: Password you use to connect to the Cloudera Data Warehouse service.
   - HTTP Path: cliservice
   - Require SSL: Make sure this is checked.
8. Click Sign In.

Enable BI mode to rewrite queries automatically

You must enable BI mode to combine the BI capability with the power of transparent materialized view rewritings.

About this task

If you want your incoming queries, at different granularity levels, to be answered directly from the materialized view rather than being computed from the source tables, you must enable BI mode. BI mode is disabled by default.

Procedure

1. To enable BI mode, execute the following statement from Hue or your preferred client. set hive.optimize.bi.enabled=true;
2. Create a materialized view that stores one of the supported DataSketches (for example, HLL or KLL) on the columns for which the aggregations will be computed during query processing.
   The following example shows creating a materialized view for computing the number of unique visitors per region for a given website.

```
CREATE MATERIALIZED VIEW AS
SELECT country, state, city, ds_hll_sketch(userid)
FROM visitors
GROUP BY country, state, city;
```

Results

With the BI mode enabled, any incoming queries will be sliced and diced at different granular levels to be answered directly from the materialized view.

```
SELECT country, count(distinct userid)
FROM visitors
GROUP BY country;

SELECT country, state, count(distinct userid)
FROM visitors
GROUP BY country, state;

SELECT country, state, count(distinct userid)
FROM visitors
WHERE country = ‘USA’
 GROUP BY country, state;
```

Using roll ups and grouping sets

If your work involves heavy OLAP queries, you need to know about grouping sets and roll ups. Unified Analytics supports grouping sets with rollups and grouping functions.

Before you begin

You created a Unified Analytics Virtual Warehouse based on a Database Catalog that loads the sample data to run queries of airline data.

Procedure

1. Log into the Cloudera Data Warehouse service as DWUser.

2. Go to the **Virtual Warehouses** tab, locate the Virtual Warehouse using which you want to run queries, and click HUE.

   The Hue query editor opens in a new browser tab.
3. Select a database.
4. In Hue, select either the Impala or Hive editor.
5. Run a query that determines the number of flights per country and state.

```
SELECT
  a.country,
  a.state,
  grouping(a.country)
  grouping(a.state)
  SUM(f.Distance),
  COUNT(*)
FROM
  flights f,
  airports a
WHERE
  f.origin = a.iata
GROUP BY
  a.country,
  a.state
WITH ROLLUP;
```

6. Run a query that defines custom grouping sets: country and country/state.

```
SELECT
  a.country,
  a.state,
  grouping(a.country)
  grouping(a.state)
  SUM(f.Distance),
  COUNT(*)
FROM
  flights f,
  airports a
WHERE
  f.origin = a.iata
GROUP BY
  a.country,
  a.state
GROUPING SETS (
  (a.country),
  (a.country, a.state)
);
```

Using set operations

When you want to combine separate query results into a single result, you need to know about the INTERSECT and EXCEPT clauses in addition to UNION.

About this task

In this task, you run a query to find the number of Boeing planes leaving from DFW in 2005. The query filters data using an intersect on the tail number, tailnum, with the unique identification of a plane, planes.manufacturer.

Before you begin

You created a Unified Analytics VW that loads the sample data to run queries of airline data.

Procedure

1. Log into the Cloudera Data Warehouse service as DWUser.
2. Go to the **Virtual Warehouses** tab, locate the Virtual Warehouse using which you want to run queries, and click HUE.

   The Hue query editor opens in a new browser tab.
3. Select a database.
4. In Hue, select either the Impala or Hive editor.
5. Enter a query that intersects the tail number and type of plane.

```
SELECT COUNT(*)
FROM
  SELECT tailnum
  FROM flights
  WHERE
    flights.origin = 'DFW' AND flights.`year`= 2005
  INTERSECT
  SELECT tailnum
  FROM planes
  WHERE
    planes.manufacturer = 'BOEING'
) subq;
```

6. Run a query that returns the planes not manufactured by Boeing leaving from DFW.

```
SELECT COUNT(*)
FROM
  SELECT tailnum
  FROM flights
  WHERE
    flights.origin = 'DFW' AND flights.`year`= 2005
  EXCEPT
  SELECT tailnum
  FROM planes
  WHERE
    planes.manufacturer = 'BOEING'
) subq;
```

Using anti joins

Unified Analytics supports anti join queries to find the rows in a table that do not match rows in another table. The performance of queries involving NULL filters, "not exists" clauses, and "not in" clauses" are improved using the anti join optimization.

About this task

In this task, run an anti join query. For example:

```
SELECT * FROM left_table t1 LEFT JOIN right_table t2 ON t1.id = t2.id WHERE
t2.id IS NULL;
```

Procedure

1. Log into the Cloudera Data Warehouse service as DWUser.
2. Go to the Virtual Warehouses tab, locate the Virtual Warehouse you want to edit, and click ⋮ Edit Configurations Hiveserver2 and select hive-site from the Configuration files drop-down menu.
3. Click Add Custom Configuration.

   The **Custom Configuration** is displayed.
4. Specify the following values in the Configuration Key: and Configuration Value: fields:
   - Configuration Key: hive.auto.convert.anti.join
   - Configuration Value: true
5. Open Hue, and create two tables, one having more rows than the other.

```
create table tt1 (order_num bigint, price decimal(5, 2));

create table tt2 (order_num bigint);
```

6. Insert several, say five, rows of values (order number, price) in the first table and fewer, say 2, rows of values (order number only) in the second table.

```
insert into tt1 values (42, 300.50), (141, 250.25), (150, 85.00), (150,
500.44), (166, 125.65);
insert into tt2 values (150), (166);
```

7. Take a look at the content of the first table.

```
select * from tt1;
```

```
tt1.order_num tt1.price
42             300.50
141             250.25
150             85.00
150             500.44
166             125.65
```

8. Take a look at the content of the second table.

```
select * from tt2;
```

```
tt2.order_num
150
166
```

9. Perform an equality left join on the order numbers in the tables.

```
select * from tt1 t1 LEFT JOIN tt2 t2 ON t1.order_num=t2.order_num;
```

```
t1.order_num t1.price t2.order_num
42           300.50    NULL
141          250.25    NULL
150          85.00     150
150          500.44    150
166          125.65    166
```

10. Add a null filter to the join query to anti join the tables.

```
select * from tt1 t1 LEFT JOIN tt2 t2 ON t1.order_num=t2.order_num where
 t2.order_num is NULL;
```

The anti join returns the rows having NULL in the price column.

```
t1.order_num t1.price t2.order_num
42           300.50   NULL
141          250.25   NULL
```

Creating a table from Parquet data

In Unified Analytics, you can base a new table on a schema in a Parquet file using the CREATE TABLE LIKE FILE PARQUET statement. You are likely familiar with this statement if you have been an Impala user. The Impala counterpart CREATE TABLE LIKE PARQUET lacks the keyword LIKE, but is functionally similar.

About this task

The column definitions in the new table are inferred from the Parquet data file when you create a table like Parquet in Unified Analytics. Set the following table property for creating the table:

```
hive.parquet.infer.binary.as = <value>
```

Where <value> is binary (the default) or string.

This property determines the interpretation of the unannotated Parquet binary type. Some systems expect binary to be interpreted as string.

Use the following syntax for creating a table like Parquet.

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name LIKE PARQUET FI
LE 'object_storage_path_of_parquet_file'
[PARTITIONED BY (col_name data_type [COMMENT 'col_comment'], ...)]
[SORT BY ([column [, column ...]])] [COMMENT 'table_comment'] [ROW FORMAT
row_format]
[WITH SERDEPROPERTIES ('key1'='value1', 'key2'='value2', ...)] [STORED AS f
ile_format]
[LOCATION 'object_storage_path']
[TBLPROPERTIES ('key1'='value1', 'key2'='value2', ...)]
```

In this task, you create a table like parquet that uses the default value binary for hive.parquet.infer.binary.as.

Procedure

Create a table based on a parquet table.

```
CREATE TABLE ua_table LIKE FILE PARQUET 's3a://testbucket/files/schema.p
arq';
```

Query correlated data

You can query one table relative to the data in another table.

About this task

A correlated query contains a query predicate with the equals (=) operator. One side of the operator must reference at least one column from the parent query and the other side must reference at least one column from the subquery. An uncorrelated query does not reference any columns in the parent query.

Procedure

Select all state and net_payments values from the transfer_payments table for years during which the value of the state column in the transfer_payments table matches the value of the state column in the us_census table.

```
SELECT state, net_payments
FROM transfer_payments
WHERE EXISTS
   (SELECT year
   FROM us_census
   WHERE transfer_payments.state = us_census.state);
```

This query is correlated because one side of the equals predicate operator in the subquery references the state column in the transfer_payments table in the parent query and the other side of the operator references the state column in the us_census table.

This statement includes a conjunct in the WHERE clause.

A conjunct is equivalent to the AND condition, while a disjunct is the equivalent of the OR condition The following subquery contains a conjunct:

... WHERE transfer_payments.year = "2018" AND us_census.state = "california"

The following subquery contains a disjunct:

... WHERE transfer_payments.year = "2018" OR us_census.state = "california"

Subquery restrictions

To construct queries efficiently, you must understand the restrictions of subqueries in WHERE clauses.

- Subqueries must appear on the right side of an expression.
- Nested subqueries are not supported.
- Subquery predicates must appear as top-level conjuncts.
- Subqueries support four logical operators in query predicates: IN, NOT IN, EXISTS, and NOT EXISTS.
- The IN and NOT IN logical operators may select only one column in a WHERE clause subquery.
- The EXISTS and NOT EXISTS operators must have at least one correlated predicate.
- The left side of a subquery must qualify all references to table columns.
- References to columns in the parent query are allowed only in the WHERE clause of the subquery.
- Subquery predicates that reference a column in a parent query must use the equals (=) predicate operator.
- Subquery predicates may refer only to columns in the parent query.
- Correlated subqueries with an implied GROUP BY statement may return only one row.
- All unqualified references to columns in a subquery must resolve to tables in the subquery.
- Correlated subqueries cannot contain windowing clauses.

Comparing tables using ANY/SOME/ALL

You learn how to use quantified comparison predicates (ANY/SOME/ALL) in non-correlated subqueries according to the SQL standard. SOME is any alias for ANY.

About this task

You can use one of the following operators with a comparison predicate:

- >
- <
- >=
- <=
- <>
- =

ALL:

- If the table is empty, or the comparison is true for every row in subquery table, the predicate is true for that predicand.
- If the comparison is false for at least one row, the predicate is false.

SOME or ANY:

- If the comparison is true for at least one row in the subquery table, the predicate is true for that predicand.
- If the table is empty or the comparison is false for each row in subquery table, the predicate is false.

If the comparison is neither true nor false, the result is undefined.

For example, you run the following query to match any value in c2 of tbl equal to any value in c1 from the same tbl:

```
select c1 from tbl where c1 = ANY (select c2 from tbl);
```

You run the following query to match all values in c1 of tbl not equal to any value in c2 from the same tbl.

```
select c1 from tbl where c1 <> ALL (select c2 from tbl);
```

Handling ETL jobs

Unified Analytics provides DML support for extract, transform, and load tasks. ETL jobs, identified by their size, run in query isolation mode if enabled.

About this task

Unified Analytics supports insert, update, merge, and delete statements for performing ETL tasks.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Create a Unified Analytics Virtual Warehouse (Hive), selecting the Enable Query Isolation option.

   Query isolation is available as a technical preview and requires an entitlement.
3. Click Create Virtual Warehouse.

What to do next

Insert, update, and delete data as described in subsequent topics.

Related Information

Automating data pipelines with Cloudera Data Engineering and Cloudera Data Warehouse using Apache Airflow

Inserting data into a table

To insert data into a table you use a familiar ANSI SQL statement. A simple example shows you have to accomplish this basic task.

About this task

To insert data into an ACID table, use the Optimized Row Columnar (ORC) storage format. To insert data into a non-ACID table, you can use other supported formats. You can specify partitioning as shown in the following syntax:

INSERT INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)] VALUES values_row [, values_row...]

where

values_row is (value [, value]) .

A value can be NULL or any SQL literal.

Procedure

1. Create an ACID table to contain student information.
   CREATE TABLE students (name VARCHAR(64), age INT, gpa DECIMAL(3,2));
2. Insert name, age, and gpa values for a few students into the table.
   INSERT INTO TABLE students VALUES ('fred flintstone', 35, 1.28), ('barney rubble', 32, 2.32);
3. Create a table called pageviews and assign null values to columns you do not want to assign a value.

```
CREATE TABLE pageviews (userid VARCHAR(64), link STRING, from STRING) PA
RTITIONED BY (datestamp STRING) CLUSTERED BY (userid) INTO 256 BUCKETS;
INSERT INTO TABLE pageviews PARTITION (datestamp = '2014-09-23') VALUES
('jsmith', 'mail.com', 'sports.com'), ('jdoe', 'mail.com', null);
INSERT INTO TABLE pageviews PARTITION (datestamp) VALUES ('tjohnson', 'sp
orts.com', 'finance.com', '2014-09-23'), ('tlee', 'finance.com', null, '
2014-09-21');
```

   The ACID data resides in the warehouse.

Updating data in a table

The syntax describes the UPDATE statement you use to modify data already stored in a table. An example shows how to apply the syntax.

About this task

You construct an UPDATE statement using the following syntax:

UPDATE tablename SET column = value [, column = value ...] [WHERE expression];

Depending on the condition specified in the optional WHERE clause, an UPDATE statement might affect every row in a table. The expression in the WHERE clause must be an expression supported by a SELECT clause. Subqueries are not allowed on the right side of the SET statement. Partition columns cannot be updated.

Before you begin

You must have SELECT and UPDATE privileges to use the UPDATE statement.

Procedure

Create a statement that changes the values in the name column of all rows where the gpa column has the value of 1.0.
UPDATE students SET name = null WHERE gpa <= 1.0;

Merging data in tables

A sample statement shows how you can conditionally insert existing data in Hive tables using the ACID MERGE statement. Additional merge operations are mentioned.

About this task

The MERGE statement is based on ANSI-standard SQL.

Procedure

1. Construct a query to update the customers' names and states in customer table to match the names and states of customers having the same IDs in the new_customer_stage table.
2. Enhance the query to insert data from new_customer_stage table into the customer table if none already exists.

   Update or delete data using MERGE in a similar manner.

```
MERGE INTO customer USING (SELECT * FROM new_customer_stage) sub ON sub.id
 = customer.id
WHEN MATCHED THEN UPDATE SET name = sub.name, state = sub.new_state
WHEN NOT MATCHED THEN INSERT VALUES (sub.id, sub.name, sub.state);
```

Deleting data from a table

You use the DELETE statement to delete data already written to an ACID table.

About this task

Use the following syntax to delete data from a Hive table. DELETE FROM tablename [WHERE expression];

Procedure

Delete any rows of data from the students table if the gpa column has a value of 1 or 0.
DELETE FROM students WHERE gpa <= 1,0;

Cloudera Data Warehouse Public Cloud

Cloudera Data Warehouse Release Notes

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Version mapping **6**
Version mapping for supported releases 6
Version mapping for unsupported releases 7
Impala shell and Impyla versions 8
February 3, 2025 - Hotfix **8**
What's new 8
Fixed issues 9
December 18, 2024 - Hotfix **9**
Fixed issues 9
December 5, 2024 - GA **10**
What's new 10
Known issues 11
Fixed issues 25
Deprecation notices 27
Behavior changes 27
October 4, 2024 - Hotfix **29**
Fixed issues 29
Known issues 29
August 15, 2024 - Hotfix **30**
Fixed issues 30
Known issues 31
Behavior changes 31
July 26, 2024 - GA **32**
What's new 32
Preview features in Cloudera Data Warehouse Public Cloud 35
Known issues in Cloudera Data Warehouse Public Cloud 36
Fixed issues in Cloudera Data Warehouse Public Cloud 50
Behavior changes in Cloudera Data Warehouse Public Cloud 51
Unsupported releases **52**
May 2, 2024 - Hotfix 52
Fixed issues 52
Behavior changes 53
March 26, 2024 - Hotfix 54
Fixed issues 54
Known issues 55
March 4, 2024 - Hotfix 71
Fixed issues 71
Known issues 71
Behavior changes 88
February 29, 2024 - GA 88
What's new in Cloudera Data Warehouse on Public Cloud 88
Preview features in Cloudera Data Warehouse on Public Cloud 92
Known issues in Cloudera Data Warehouse on Public Cloud 92
Fixed issues in Cloudera Data Warehouse on Public Cloud 110
Behavior changes in Cloudera Data Warehouse on Public Cloud 111
What's new (unsupported releases) 111
January 18, 2024 112
January 10, 2024 112
December 19, 2023 112
December 1, 2023 112

November 20, 2023 113
October 12, 2023 116
October 5, 2023 116
August 30, 2023 117
June 29, 2023 122
May 30, 2023 122
May 5, 2023 123
February 14, 2023 127
February 7, 2023 127
December 13, 2022 129
November 21, 2022 130
September 15, 2022 132
August 4, 2022 133
June 22, 2022 135
May 10, 2022 135
April 27, 2022 135
March 8, 2022 138
January 19, 2022 139
December 22, 2021 139
November 20, 2021 139
October 29, 2021 140
October 22, 2021 141
September 28, 2021 141
August 27, 2021 141
August 9, 2021 142
July 29, 2021 142
July 13, 2021 143
June 14, 2021 143
June 1, 2021 143
May 21, 2021 144
May 3, 2021 144
April 9, 2021 145
March 31, 2021 145
March 23, 2021 145
March 12, 2021 145
February 23, 2021 146
February 10, 2021 146
January 29, 2021 146
December 21, 2020 146
November 3, 2020 147
September 10, 2020 149
August 27, 2020 150
August 14, 2020 150
July 30, 2020 151
June 26, 2020 151
May 14, 2020 151
April 7, 2020 153
March 13, 2020 153
Known issues (unsupported releases) 154
January 18, 2024 154
January 10, 2024 170
December 1, 2023 186
December 19, 2023 201
November 20, 2023 217
October 16, 2023 233
October 12, 2023 248

October 5, 2023 262
August 30, 2023 277
June 29, 2023 291
May 30, 2023 306
May 5, 2023 321
February 7, 2023 336
February 14, 2023 348
December 13, 2022 348
November 21, 2022 361
September 15, 2022 374
August 4, 2022 386
June 22, 2022 397
General known issues on public clouds 406
Known issues in CDW running on AWS environments 407
Known issues in CDW running on Azure environments 410
Known issues in Data Analytics Studio on public clouds 410
Known issues in Hue on public clouds 411
Known issues in Database Catalog on public clouds 413
Known issues in Hive Virtual Warehouses on public clouds 413
Known issues in Impala Virtual Warehouses on public clouds 415
Behavior changes (unsupported releases) 417
November 17, 2023 417
August 30, 2023 419
February 7, 2023 419
Fixed issues (unsupported releases) 420
January 18, 2024 Fixed issues 420
December 19, 2023 Fixed issues 420
December 1, 2023 Fixed issues 422
November 20, 2023 Fixed issues 422
October 5, 2023 Fixed issues 423
August 30, 2023 Fixed issues 423
May 5, 2023 fixed issues 423
Feb 7, 2023 fixed issues 424
December 13, 2022 fixed issues 424
November 22, 2022 fixed issues 424
September 15, 2022 fixed issues 424
August 4, 2022 fixed issues 424
April 27, 2022 fixed issues 425
CVE-2021-44228 remediation for Cloudera Data Warehouse 425

Version information for Cloudera Data Warehouse Public Cloud

Cloudera Data Warehouse includes components covered in the Cloudera Runtime documentation. A mapping of the Cloudera Data Warehouse versions to its Hive, Impala, Hue versions and CDP CLI counterparts provides the path you follow to select the correct CLI functions for Cloudera Data Warehouse. Cloudera Data Warehouse Runtime version mapping to Apache Iceberg and to Impala client versions is also covered.

Review the Amazon EKS Kubernetes release calendar and AKS Kubernetes release calendar for support information.

Review the Support Lifecycle Policy page for information about Cloudera Data Warehouse releases that have reached end of support (unsupported releases).

Related Information

Cloudera Data Warehouse Runtime Changelog

CDP CLI Reference

Cloudera Control Plane API Changelog

Version mapping for supported releases

Review the release date, Cloudera Data Warehouse version, supported AKS and EKS versions, runtime versions (this includes Hive, Impala, Hue, and Iceberg), Cloudera Data Visualization version, and CDP CLI version for the supported releases.

Cloudera Data Warehouse component version information

Version mapping information is provided for the following Cloudera Data Warehouse components:

- Cloudera Data Warehouse: The Cloudera Data Warehouse Control Plane (CRUD application) version and its release date
- AKS/EKS: Azure Kubernetes Service or Amazon Elastic Kubernetes version of newly-activated environments
- Runtime: The Cloudera Data Warehouse Runtime version that includes Hive, Impala, Iceberg, and Hue
- Apache Iceberg: The Apache Iceberg version
- Cloudera Data Visualization: The version of Cloudera Data Warehouse
- CDP CLI: The Cloudera command line interface version

Release type	Release date	Cloudera Data Warehouse	AKS/EKS	Runtime	Apache Iceberg	Cloudera Data Visualization	CDP CLI
Hotfix	February 3, 2025	1.9.5-b10	1.31/1.31	2024.0.18.4-15	1.4.3	7.2.7-b48	0.9.127 Released on: 2024-12-06
Hotfix	December 18, 2024	1.9.4-b147	1.30/1.30	2024.0.18.4-15	1.4.3	7.2.7-b48	0.9.127 Released on: 2024-12-06
GA	December 5, 2024	1.9.3-b166	1.30/1.30	2024.0.18.4-15	1.4.3	7.2.7-b48	0.9.127 Released on: 2024-12-06

Release type	Release date	Cloudera Data Warehouse	AKS/EKS	Runtime	Apache Iceberg	Cloudera Data Visualization	CDP CLI
Hotfix	October 4, 2024	1.9.2-b657	1.29/1.29	2024.0.18.2-4	1.4.3	7.2.4-b41	0.9.119 Released on: 2024-08-02
Hotfix	August 15, 2024	1.9.2-b657	1.29/1.29	2024.0.18.1-1	1.4.3	7.2.4-b41	0.9.119 Released on: 2024-08-02
GA	July 26, 2024	1.9.1-b233	1.29/1.29	2024.0.18.0-206	1.4.3	7.2.4-b41	0.9.119 Released on: 2024-08-02

Version mapping for unsupported releases

Review the release date, Cloudera Data Warehouse version, supported AKS and EKS versions, runtime versions (this includes Hive, Impala, Hue, and Iceberg), Cloudera Data Visualization version, and CDP CLI version for the older, unsupported releases.

Release type	Release date	Cloudera Data Warehouse	AKS/EKS	Runtime	Apache Iceberg	Cloudera Data Visualization	CDP CLI
Hotfix	May 2, 2024	1.8.7-b37	1.27/1.28	2024.0.17.2-2	1.3.0	7.2.2-b33	0.9.108 Released on: 2024-03-13
Hotfix	Mar 26, 2024	1.8.7-b37	1.27/1.28	2024.0.17.1-3	1.3.0	7.1.9-b19	0.9.108 Released on: 2024-03-13
Hotfix	Mar 4, 2024	1.8.6-b264	1.27/1.28	2024.0.17.0-73	1.3.0	7.1.9-b19	0.9.108 Released on: 2024-03-13
GA	Feb 29, 2024	1.8.5-b35	1.27/1.28	2024.0.17.0-73	1.3.0	7.1.9-b19	0.9.108 Released on: 2024-03-13
Hotfix	Jan 18, 2024	1.8.4-b90	1.27/1.26	2023.0.16.3-2	1.3.0	7.1.6.2-3	0.9.102 Released on: 2023-10-11
Hotfix	Jan 10, 2024	1.8.3-b130	1.27/1.26	2023.0.16.3-2	1.3.0	7.1.6.2-3	0.9.102 Released on: 2023-10-11
Hotfix	Dec 19, 2023	1.8.2-b221	1.27/1.26	2023.0.16.2-1	1.3.0	7.1.6.2-3	0.9.102 Released on: 2023-10-11

Release type	Release date	Cloudera Data Warehouse	AKS/EKS	Runtime	Apache Iceberg	Cloudera Data Visualization	CDP CLI
Hotfix	Dec 1, 2023	1.8.1-b248	1.27/1.26	2023.0.16.1-2	1.3.0	7.1.6.2-3	0.9.102 Released on: 2023-10-11
GA	Nov 20, 2023	1.8.1-b248	1.27/1.26	2023.0.16.0-150	1.3.0	7.1.6.2-3	0.9.102 Released on: 2023-10-11
Hotfix	Oct 16, 2023	1.7.3-b12	1.26/1.26	2023.0.15.1-2	1.3.0	7.1.3-b36	0.9.99 Released on: 2023-10-11
GA	Oct 5, 2023	1.7.2-b321	1.26/1.26	2023.0.15.1-2	1.3.0	7.1.3-b36	0.9.99 Released on: 2023-10-11

Impala shell and Impyla versions

Review the minimum Impala shell and Impyla versions your client users need to connect to a particular version of Cloudera Data Warehouse.

Cloudera Data Warehouse Runtime Version	Impala shell version	Impyla version
2024.0.18.0-206	4.4.0.a1	-
2023.0.15.0-243 - 2023.0.16.0-150	4.3.0a3	0.18.0
2023.0.14.0-155	4.2.0	0.18.0
2021.0.5-b36 - 2023.0.13.0-20	4.1.0a1	0.18a2

February 3, 2025 - Hotfix

Review the new features and fixed issues in this hotfix release of Cloudera Data Warehouse Public Cloud

What's new in Cloudera Data Warehouse Public Cloud

Review the new features introduced in this release of Cloudera Data Warehouse service on Cloudera Public Cloud

- Cloudera Data Warehouse on Azure environments
- Cloudera Data Warehouse on AWS environments

What's new in Cloudera Data Warehouse on Azure environments

Azure AKS 1.31 upgrade

Cloudera supports the Azure Kubernetes Service (AKS) version 1.31. In 1.9.5-b10 (released Februrary 3, 2025), when you activate an Environment, Cloudera Data Warehouse automatically provisions AKS 1.31. To upgrade to AKS 1.31 from an earlier version of Cloudera Data Warehouse, you must backup and restore Cloudera Data Warehouse.

Note: Using the Azure CLI or Azure portal to upgrade the AKS cluster is not supported. Doing so can cause the cluster to become unusable and can cause downtime. For more information about upgrading, see Upgrading an Azure Kubernetes Service (AKS) cluster.

What's new in Cloudera Data Warehouse on AWS environments

AWS EKS 1.31 upgrade

Cloudera supports the AWS Elastic Kubernetes Service (EKS) version 1.31. In 1.9.5-b10 (released Februrary 3, 2025), when you activate an Environment, Cloudera Data Warehouse automatically provisions EKS 1.31. To upgrade to EKS 1.31 from an earlier version of Cloudera Data Warehouse, you must backup and restore Cloudera Data Warehouse.

Note: Using the AWS tools to upgrade the EKS cluster is not supported. Doing so can cause the cluster to become unusable and can cause downtime. For more information about upgrading, see Upgrading an Amazon Kubernetes Service (EKS) cluster

Fixed Issues

Review the fixed issues in this release of the Cloudera Data Warehouse service Public Cloud

DWX-20070: Incompatibility with custom subdomains using .dw format

Cloudera Data Warehouse environments could not be activated when specifying a custom subdomain in the older .dw format using the --custom-subdomain CLI flag. This resulted in certificate creation failures with the error:

```
error while obtaining certificate: context canceled Error Code:
undefined
```

This issue impacted customers with the entitlement CDW_CUSTOM_CLUSTER_ID using the old .dw domain format.

The fix includes updating the certificate creation process to support custom subdomains in the old .dw format. Cloudera Data Warehouse environments can now be activated seamlessly with custom subdomains in both legacy and updated formats however, Cloudera strongly recommends to migrate to the new format that was introduced in August, 2021, see Cloudera Data Warehouse Public Cloud endpoints change.

Note: Support for the old subdomain format is going to be removed in a future release.

December 18, 2024 - Hotfix

Review the fixed issues and changed behaviors in this hotfix release of Cloudera Data Warehouse Public Cloud.

Fixed issues

Review the fixed issues in this release of the Cloudera Data Warehouse service Public Cloud.

DWX-19898: Databus producer fails with multiple bucket configurations

After a fresh install or DBC upgrade, the databus-producer fails when separate buckets are configured for data and logs. The error occurs because it expects all paths to be in the primary data bucket, but logs are stored in a different bucket.

The issue was resolved by updating the databus-producer version to 1.1.0-b4 in the Cloudera Data Warehouse 1.9.4-b147 release (released December 18, 2024).

December 5, 2024 Release Notes

Review the new feaures, fixes, behavioral changes, and preview features in this release of Cloudera Data Warehouse Public Cloud.

What's new in Cloudera Data Warehouse Public Cloud

Review the new features introduced in this release of Cloudera Data Warehouse service on Cloudera Public Cloud.

- Cloudera Data Warehouse features
- Cloudera Data Warehouse on Azure environments
- Cloudera Data Warehouse on AWS environments
- Impala
- Hue

What's new in Cloudera Data Warehouse Public Cloud

Ability to select an instance type for Virtual Warehouses is GA

You can select AWS or Azure compute instance types, such as r6id.4xlarge or Standard_E16_v3, while creating a Virtual Warehouse, both using CDP CLI and UI. See Creating a Virtual Warehouse.

If you select a compute instance type both during environment activation and creating a Virtual Warehouse, then what you select for a Virtual Warehouse takes precedence.

Improvements to Impala Autoscaler Dashboard

The following inprovements were introduced for the Impala Autoscaler Dashboard:

- Ability to select the log-level configuration for the autoscaler and autoscaler metrics containers.
- A new “Understanding The Dashboard” page has been added which explains the metrics displayed on the UI and how they are calculated.
- Empty data points that manifest as gaps in the graphs are skipped. Zero values are accurately displayed.

Ability to view end-of-support information through UI and CDP CLI

Cloudera Data Warehouse releases reach the end of support every six months. The Cloudera Data Warehouse UI displays whether your deployment is nearing its end of support time or is unsupported, enabling you to plan an upgrade. You can also view the upgrade instructions on the UI. The end of support information is also displayed when you run the list-clusters and describe-clusters CDP CLI commands.

Streamlined option for downloading Cloudera Data Warehouse diagnostic bundles

Cloudera Data Warehouse users can now easily download diagnostic bundles with a direct Collect option that reduces the need for prior time interval and log selection adjustments. This update enables faster, more efficient access to relevant diagnostic data.

For more information, see Cloudera Data Warehouse Diagnostic Bundle Documentation, Downloading Cloudera Data Warehouse and Kubernetes diagnostic bundles and Diagnostic bundles for Cloudera Data Warehouse and Kubernetes.

What's new in Cloudera Data Warehouse on Azure environments

Azure AKS 1.30 upgrade

Cloudera supports the Azure Kubernetes Service (AKS) version 1.30. In 1.9.3-b166 (released December 5, 2024), when you activate an Environment, Cloudera Data Warehouse automatically provisions AKS 1.30. To upgrade to AKS 1.30 from an earlier version of Cloudera Data Warehouse, you must backup and restore Cloudera Data Warehouse.

Note: Using the Azure CLI or Azure portal to upgrade the AKS cluster is not supported. Doing so can cause the cluster to become unusable and can cause downtime. For more information about upgrading, see Upgrading an Azure Kubernetes Service (AKS) cluster.

What's new in Cloudera Data Warehouse on AWS environments

AWS EKS 1.30 upgrade

Cloudera supports the AWS Elastic Kubernetes Service (EKS) version 1.30. In 1.9.3-b166 (released December 5, 2024), when you activate an Environment, Cloudera Data Warehouse automatically provisions EKS 1.30. To upgrade to EKS 1.30 from an earlier version of Cloudera Data Warehouse, you must backup and restore Cloudera Data Warehouse. For more information about upgrading, see Upgrading an Amazon Kubernetes Service (EKS) cluster.

What's new in Impala on Cloudera Data Warehouse Public Cloud

Improved log storage location access for S3 and Azure in Cloudera Data Warehouse

Cloudera Data Warehouse now offers a simplified way to locate Impala diagnostic logs in S3 and Azure. With fewer required identifiers and clear directory paths, users can efficiently access the logs they need. Find details on the updated steps in S3 and Azure.

What's new in Hue on Cloudera Data Warehouse Public Cloud

Enhanced AI Integration in Hue SQL AI Assistant

The Hue SQL AI Assistant now supports Cloudera AI Workbench and Cloudera AI Inference service. These integrations enhance the Hue SQL AI Assistant by enabling the use of private models hosted within Cloudera-managed infrastructure. This ensures enhanced security and privacy while leveraging GenAI for the Hue SQL-related tasks.

- Cloudera AI Workbench: This enables you to securely deploy and run your own models within a virtual private cloud. This configuration enhances control and privacy within your environment. For more information, see Configure SQL AI Assistant using Cloudera AI Workbench.
- Cloudera AI Inference service: Helps in a production-grade serving environment for hosting predictive and generative AI models. This service simplifies model deployment and maintenance. For more information, see Configure SQL AI Assistant using Cloudera AI Inference service.

Known issues in Cloudera Data Warehouse Public Cloud

Review the known issues in this release of the Cloudera Data Warehouse service on Cloudera Public Cloud.

- Known issues identified in the December 5, 2024 release on page 11
- Known issues identified before the December 5, 2024 release on page 12
- Technical Service Bulletins on page 24

Known issues identified in the December 5, 2024 release

Impala Virtual Warehouse fails to start due to missing helm release

A failed helm release status prevents helm upgrade operations from executing successfully. This causes the Impala Virtual Warehouse to fail with the following error:

```
time="2024-07-04T13:32:46Z" level=info msg="helm release not fou
nd for input executor group: \"impala-executor-000\" in service
impala-executor"
```

```
time="2024-07-04T13:32:46Z" level=error msg="addExecutorGroups
failed with err: \"helm release not found for \\\"impala-executo
r-000\\\"\""
```

Perform one of the following tasks to address this issue:

- Recreate the Virtual Warehouse
- Manually roll back the Helm release using the Helm client.

Look through the release history and identify a release that was in a healthy deployed state. Roll back to the identified release version to bring back the Virtual Warehouse to a healthy state.

Database Catalog goes to an error state after a Data Lake resize or after an upgrade to Azure Flexible Server

After a Cloudera Data Lake resize or after an upgrade to Azure Flexible Server, the Cloudera Data Warehouse Database Catalog needs to be restarted to consider the change in Data Lake size. Users might restart the Database Catalog using the CDP CLI or by using the Stop and Start functionality from the Cloudera Data Warehouse UI.

Irrespective of how the Database Catalog is restarted, it ends up in an "Error" state.

After you resize the Cloudera Data Lake or upgrade from Azure Single Server to Flexible Server, start the Database Catalog from the Cloudera Data Warehouse UI and then rebuild it.

Note: The Database Catalog will fail when you start it, however, this step is required and needs to be performed before rebuilding the Database Catalog.

You must also ensure that you restart each Virtual Warehouse as recommended in the Data Lake resizing post-requisites or Post-requisites after upgrading to Azure Flexible Server.

Known issues identified before the December 5, 2024 release

DWX-19451: Cloudera Data Visualization restore job can fail with ignorable errors

After a successful Cloudera Data Visualization restoration job, the restore job could be in a failed state with the log displaying ignorable errors.

```
pg_restore: error: could not execute query: ERROR:  sequence "jo
bs_joblog_id_seq" does not exist
Command was: DROP SEQUENCE public.jobs_joblog_id_seq;
pg_restore: error: could not execute query: ERROR:  table "jobs_j
oblog" does not exist
Command was: DROP TABLE public.jobs_joblog;
pg_restore: error: could not execute query: ERROR:  sequence "
jobs_jobcontent_id_seq" does not exist
Command was: DROP SEQUENCE public.jobs_jobcontent_id_seq;
.......
.......
```

This issue occurs because the restore job issues commands to DROP all the objects that will be restored, and if any of these objects do not exist in the destination database, such ignorable errors are reported.

This has no functional impact on the restored Cloudera Data Visualization application. It is noticed that all the backed up queries, datasets, connections, and dashboards are restored successfully and Cloudera Data Visualization is available for new queries.

None.

DWX-19003: Unable to set t-shirt size for a Cloudera Data Visualization instance

You cannot set or configure a t-shirt size for a Cloudera Data Visualization instance while creating or editing it from Cloudera Data Warehouse. By default, any Cloudera Data Visualization instance created in Cloudera Data Warehouse release version 1.9.1 uses the “small” compute instance size.

None.

DWX-18950: Hue data restoration fails on Azure Cloudera Data Warehouse clusters having Cloudera Data Warehouse version 1.8.7 or earlier

If you backup your Cloudera Data Warehouse cluster running Cloudera Data Warehouse version 1.8.7 or earlier and if you perform the restore operation without reactivating the latest Cloudera Data Warehouse environment, you might encounter Hue restoration job failure.

None.

DWX-18843: Unable to read Iceberg table from Hive Virtual Warehouse

If you have used Apache Flink to insert data into an Iceberg table that is created from Hive, you cannot read the Iceberg table from the Hive Virtual Warehouse.

Add the engine.hive.enabled table property through the Hive beeline and set the value to "true". You can add this table property either while creating the Iceberg table or use the ALTER TABLE statement to add the table property.

DWX-18489: Hive compaction of Iceberg tables results in a failure

When Cloudera Data Warehouse and Cloudera Data Hub are deployed in the same environment and use the same Hive Metastore (HMS) instance, the Cloudera Data Hub compaction workers can inadvertently pick up Iceberg compaction tasks. Since Iceberg compaction is not yet supported in the latest Cloudera Data Hub version, the compaction tasks will fail when they are processed by the Cloudera Data Hub compaction workers.

In such a scenario where both Cloudera Data Warehouse and Cloudera Data Hub share the same HMS instance and there is a requirement to run both Hive ACID and Iceberg compaction jobs, it is recommended that you use the Cloudera Data Warehouse environment for these jobs. If you want to run only Hive ACID compaction tasks, you can choose to use either the Cloudera Data Warehouse or Cloudera Data Hub environments.

If you want to run the compaction jobs without changing the environment, it is recommended that you use Cloudera Data Warehouse. To avoid interference from Cloudera Data Hub, change the value of the hive.compactor.worker.threads Hive Server (HS2) property to '0'. This ensures that the compaction jobs are not processed by Cloudera Data Hub.

1. In Cloudera Manager, click Clusters Hive Configuration to navigate to the configuration page for HMS.
2. Search for hive.compactor.worker.threads and modify the value to '0'.
3. Save the changes and restart the Hive service.

DWX-18854: Compaction cleaner configuration

The compaction cleaner is turned off by default in the Cloudera Data Warehouse database catalog, potentially causing compaction job failures.

To enable the compactor on the Hive Metastore (HMS) instance of the Cloudera Data Warehouse database catalog, set the hive.compactor.cleaner.on property to "true".

DWX-12703: Hue connects to only one Impala coordinator in Active-Active mode

You may not see all Impala queries that have run on the Virtual Warehouse from the **Impala** tab on the Hue **Job Browser**. You encounter this on an Impala Virtual Warehouse that has Impala coordinator configured in an active-active mode. This happens because Hue fetches this information from only one Impala coordinator that is active.

None. You can view the query history from the **Impala Queries** tab on the **Job Browser** page, because the information is fetched from the Hue Query Processor.

Known limitation: Cloudera Data Warehouse does not support S3 Express One Zone buckets

Cloudera does not recommend deploying the Data Lake on S3 Express One Zone buckets. Cloudera Data Warehouse cannot read content present in the S3 Express One Zone buckets. The following limitations apply when using S3 Express buckets:

- You can only use S3 Express buckets with Cloudera Data Hubs running Runtime 7.2.18 or newer. Data services do not support it, currently.
- S3 Express buckets may not be used for logs and backups.

DWX-15112: Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

None.

DWX-17455: ODBC client using JWT authentication cannot connect to Impala Virtual Warehouse

If you are using the Cloudera ODBC Connector and JWT authentication to connect to a Cloudera Data Warehouse Impala Virtual Warehouse where the Impala coordinators are configured for high availability in an active-active mode, the connection results in a "401 Unauthorized" error. The error is not seen on Impala Virtual Warehouses that have an active-passive high availability or where high availability is disabled.

In the ODBC Data Source Name (DSN), make the following changes:

1. Ensure that SSL is enabled by setting SSL=1.
2. Remove authentication by setting AuthMech=0.
3. Remove the JWTString parameter.
4. Add the parameter http.header.Authorization and set it to the word "Bearer" followed by a space and the JWT string

DSN configuration before the workaround

```
SSL=1
AuthMech=8
JWTString=full_jwt_text
```

DSN configuration after the workaround

```
SSL=1
AuthMech=0
http.header.Authorization=Bearer full_jwt_text
```

HIVE-28055: Merging Iceberg branches requires a target table alias

Hive supports only one level of qualifier when referencing columns. In other words only one dot is accepted. For example, select table.col from ...; is allowed. select db.table.col is not allowed. Using the merge statement to merge Iceberg branches without a target or source table alias causes an exception:

```
org.apache.hadoop.hive.ql.parse.SemanticException: ... Invalid t
able alias or column reference ...
```

Use an alias, for example t, for the target table.

```
merge into mydb.target.branch_branch1 t using mydb.source.branch
_branch1 s on t.id = s.id when matched then update set value = '
matched';
```

DWX-17620: Folder having special characters in its name is not accessible in ABFS

In Cloudera Data Warehouse Public Cloud, from a Virtual Warehouse going to the Azure Blob Filesystem (ABFS), creating a folder, and then performing an action such as Move, causes an error as shown in the following example:

```
Cannot access: abfs://data-files/user/hrt_qa/~@$&()*!+'=;. 404 C
lient Error: The specified path does not exist.
```

None.

Branch FAST FORWARD does not work as expected

The Apache Iceberg spec indicates you can use either one or two arguments to fast forward a branch. The following example shows using two arguments:

```
ALTER TABLE <name> EXECUTE FAST-FORWARD 'x' 'y'
```

However, omitting the second branch name, does not work as documented by Apache Iceberg. The named branch is not fast-forwarded to the current branch. An exception occurs at the Iceberg level.

You must use two arguments to the EXECUTE FAST FORWARD feature to forward a branch.

DWX-17613: Generic error message is displayed when you click on the directory you don't have access to on a RAZ cluster

You see the following error message when you click on an ABFS directory to which you do not have read/write permission on the ABFS File Browser in Hue: There was a problem with your request. This message is generic and does not provide insight into the actual issue.

None.

DWX-17109: ABFS File Browser operations failing intermittently

You may encounter intermittent issues while performing typical operations on files and directories on the ABFS File Broswer, such as moving or renaming files.

None.

CDPD-27918: Hue does not automatically pick up RAZ HA configurations

On a Cloudera Public Cloud environment in which you have configured RAZ in High Availability mode, Hue in Cloudera Data Warehouse does not pick up all the RAZ host URLs automatically. Therefore, if a RAZ instance to which Hue is connected goes down, Hue becomes unavailable.

You must manually add comma-separated RAZ instances in the Hue Advanced Configuration Snippet.

1. Log in to the Cloudera Management Console as an Administrator.
2. Go to Environment Data Lake and open Cloudera Manager for your environment.
3. Go to Clusters Ranger RAZ service Instances RAZ server Processes and note the value of the fs.s3a.ext.raz.rest.host.url property from the core-site.xml file. You need this to specify the value of the api_url property in the Hue configuration.

   For Azure environments, note the value of the fs.azure.ext.raz.rest.host.url property.

   For AWS and GCS environments, note the value of the fs.s3a.ext.raz.rest.host.url property.
4. Go to Cloudera Data Warehouse Virtual Warehouse ⋮ Edit CONFIGURATIONS and select the hue-safety-valve from the Configuration files dropdown menu.
5. Add the following lines in the hue-safety-valve field:

   ```
   [desktop]
   [[raz]]
   is_enabled=true
   api_url=https://[***INSTANCE-1***]:6082/,https://[***INSTANCE-
   2***]:6082/
   ```

6. Click Apply Changes.

CDPD-66779: Partitioned Iceberg table not getting loaded with insert select query from Hive

If you create a partitioned table in Iceberg and then try to insert data from another table as shown below, an error occurs.

```
insert into table partition_transform_4 select t, ts from vector
tab10k;
```

Use the CLUSTER BY clause on the partitioned column to insert data. For example:

```
insert into table partition_transform_4 select t, ts from t1 clu
ster by ts;
```

DWX-17703: Non-HA Impala Virtual Warehouse on a private Azure Kubernetes Service (AKS) setup fails

When 'Refresh' and 'Stop' operations run in parallel, Impala might move into an error state. The Refresh operation might think that Impala is in an error state as the coordinator pod is missing.

Rebuild the Impala Virtual Warehouse or restart it using the CLI.

DWX-14923: After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

None.

DWX-15145: Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an environment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several different popups can occur, including the following one:

```
worker subnets should be selected from 3 different
availability zones. got: 0
```

None.

DWX-15144: Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

None.

DWX-13103: Cloudera Data Warehouse environment activation problem

When Cloudera Data Warehouse environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for Cloudera Data Warehouse Reduced Permissions Mode

If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing Cloudera Data Warehouse environments

You may see the following error while trying to apply tags to Virtual Warehouses in an existing Cloudera Data Warehouse environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.

4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Incorrect diagnostic bundle location

The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

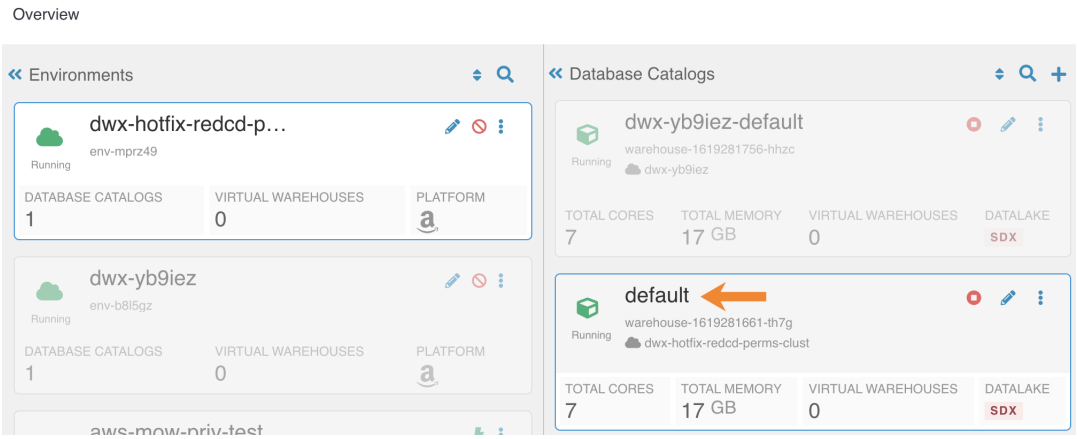

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

None.

DWX-15064: Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

None.

DWX-14452: Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in Cloudera Data Warehouse.

Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click CONFIGURATIONS Query executor and select the env configuration file.

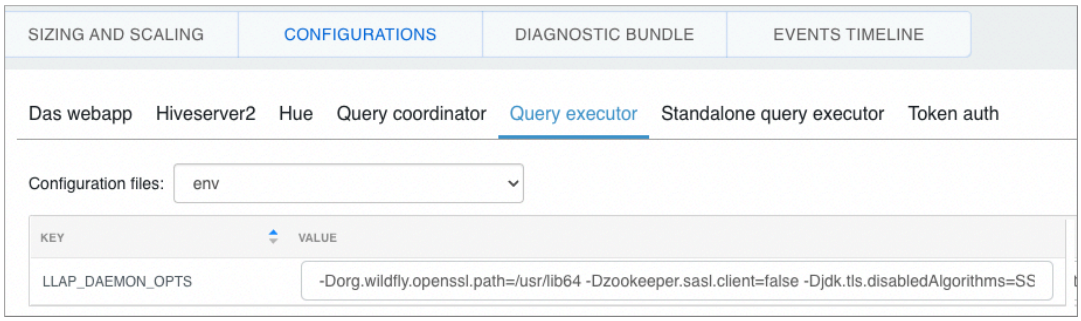

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

CDPD-40730: Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of Cloudera Data Warehouse, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

None.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Only use Beeline clients from Cloudera Runtime version 7.0.1.0 or later.

DWX-16899: Error while viewing Impala job status on the mini Job Browser

When you click on the application ID after submitting an Impala query on Hue that is running on the environment level, you may notice the following error: 401 Client Error: Unauthorized for url: http://coordinator.impala-xyz.svc.cluster.local:25000/queries?json=true Must authenticate with Basic authentication. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16895: Incorrect status of Hue pods when you edit the Hue instance properties

When you update a configuration of a Hue instance that is deployed at the environment level, such as increasing or decreasing the size of the Hue instance, you see a success message on the Cloudera Data Warehouse UI. After some time, the status of the Hue instance also changes from “Updating” to “Running”. However, when you list the Hue pods using kubectl, you see that not all backend pods are in the running state–a few of them are still in the init state.

None. The pods come up successfully eventually after a sufficient time has passed.

DWX-16863: The upgrade button is present on the Cloudera Data Warehouse UI, but Hue upgrades are not supported

You see the Upgrade button on the **Query Editor** page in the Cloudera Data Warehouse UI when Hue is deployed at the environment level. However, on Cloudera Data Warehouse version 1.8.1, upgrading the Hue instance that is deployed at the environment level is not supported.

None.

DWX-16893: A user can see all the queries in Job browser

In a Hue instance deployed at the environment level, by design, the Hue instances must not share the saved queries and query history with other Hue instances even for the same user. However, a logged in user is able to view all the queries executed by that user on all the Virtual Warehouses on a particular Database Catalog.

None.

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

None.

DWX-14927: Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

None.

DWX-14968: Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Refresh the page, or alternatively start a new session.

DWX-15115: Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

None.

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447: Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your Cloudera environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

DWX-17210, DWX-13733: Timeout issue querying Iceberg tables from Hive

When querying Iceberg tables from Hive, the queries can faile due to a timeout issue.

1. Add the following configurations to hadoop-core-site for the Database Catalog and the Virtual Warehouse.
   - fs.s3.maxConnections=1000
   - fs.s3a.connection.maximum=1000
2. Restart the Database Catalog and Virtual Warehouse.

DWX-15014: Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

None.

DWX-14163: Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in Cloudera. In addition to these, the DECIMAL type is not supported in this release.

None.

DEX-7946: Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13062: Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276: Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in Cloudera Data Warehouse Public Cloud) may encounter an exception

IMPALA-11045: Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

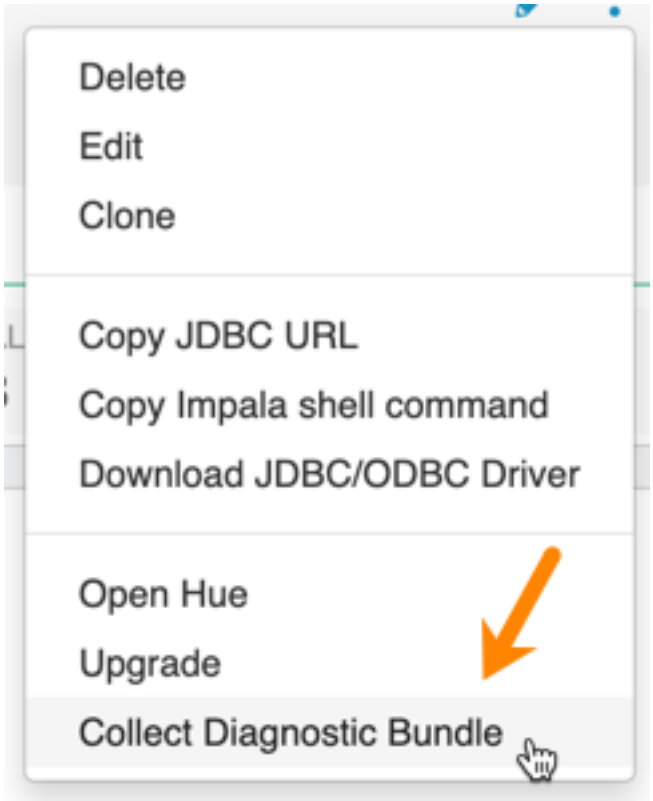

None.

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

None.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

None.

Technical Service Bulletins

TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete

Cloudera Data Warehouse customers using the Cloudera Data Warehouse Automated Backup/Restore feature will encounter an issue with the restoration versions of Cloudera Data Visualization

older than CDV 7.1.6.2-3 due to a schema change in this release. If the Backup was taken from an older Cloudera Data Warehouse environment that contained a version of Cloudera Data Visualization older than 7.1.6.2-3, the Restore procedure will succeed. Though once the user opens the Cloudera Data Visualization Queries tab, the user could encounter the error message: “column jobs_jobschedule.owner_id does not exist…”

None. For the latest update on this issue see the corresponding Knowledge article: TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete.

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in Cloudera Data Warehouse Public Cloud) may encounter an exception

The Cloudera Data Warehouse Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in Cloudera Data Warehouse Public Cloud) may encounter an exception

Fixed issues in Cloudera Data Warehouse Public Cloud

Review the issues fixed in this release of the Cloudera Data Warehouse Public Cloud service.

CDPD-74640: Improved query consistency and data writing for Beeline and Hive queries

In concurrent workflows using Beeline, queries occasionally returned incorrect results due to non-thread-safe file handling, especially when hive.query.result.cached.enabled was disabled. Additionally, INSERT OVERWRITE DIRECTORY operations failed to write data correctly to specified directories when query result caching was enabled.

The issue was addressed by implementing thread-safe file handling in HiveSequenceFileInputFormat and adjusting cache handling for directory overwrite queries, ensuring reliable query results and consistent data writes in concurrent workflows.

Apache Jira: HIVE-28530, HIVE-21386, HIVE-25907

CDPD-72985: Compatibility issue in HMS thrift struct for Hive column stats

Hive4 introduced a new field "engine" to differentiate the stats generated by different engines and this was a required field. This breaks the compatibility with clients that are already using Hive 3 or engines that are using customized thrift api like `TrinoDB`.

Made the "engine" field optional in the HMS API, restoring compatibility with Hive 3 clients for column stats retrieval.

Apache Jira: HIVE-27984

CDPD-71484: Improve LLAP performance by reusing FileSystem objects across tasks

Frequent closure of FileSystem objects disabled Hadoop's FileSystem cache, reducing LLAP efficiency.

Adjusted FileSystem handling to close objects once per query and daemon rather than per task, enhancing reuse and maintaining cache functionality.

Apache Jira: HIVE-27884

CDPD-74017: Schema resolution does not work for migrated partitioned Iceberg tables with complex data types

Schema resolution does not work correctly for migrated partitioned Iceberg tables that have complex data types. This fix addresses the field ID generation by taking the number of partitions into account. If none of the partition columns are included in the data file (common scenario), file-level field IDs are adjusted accordingly. You could also come across a scenario where all the partition columns are included in the data files.

However, if some partition columns are included in the data file while other partition columns are not, an error is generated.

Apache Jira: IMPALA-13364

DWX-18975: Configure statestored's memory based on executor group size for Impala Virtual Warehouse with workload-aware autoscaling

For smaller executor group sizes statestored's memory is configured to 1GB while for larger executor group sizes it is configured to 2GB to handle updates from more executors.

DWX-18701: Increasing Catalogd's JVM's Xmx does not increase container's memory limit

Increasing the JVM's Xmx for the Catalogd did not lead to a corresponding increase in the container's memory request and limit, leading to a mismatch between the configuration and actual memory usage. Adjusted the configuration process to ensure that when JVM's Xmx memory is increased, the container's memory request and limits are updated accordingly.

CDPD-73442: Resolution of potential deadlock

This fix addresses a deadlock issue in long-running sessions with an active idle_query_timeout, which caused new queries to hang and prevented existing queries from expiring.

Apache Jira: IMPALA-13313

CDPD-73187: Impala Ranger audit plugin fails to create audit logs

The fix ensures that the Ranger plug-in in Hive and Impala send audit events to the Solr service after upgrading Data Lake to a version that requires SSL for Ranger's audit events.

DWX-18050: Ranger audit logs show origin client's IP address for Impala Virtual Warehouse

Ranger audit logs now show the origin client's IP address for Impala Virtual Warehouse when Impala coordinator's flagfile config has use_xff_address_as_origin=true. This applies to all Impala clients such as impala-shell, impyla, jdbc and odbc clients.

Workaround for Hue: When using Hue client, enable_xff_for_hive_impala=true in hue-safety-valve configuration is additionally also required to get origin client's IP in ranger audit logs.

Note: This change does not fix the issue where Cloudera Data Visualization connections log an internal IP address and not the origin client's IP address in ranger audit.

DWX-19110: Executor deployment issue on workload-aware autoscaling VW creation

Workload-aware autoscaling Impala Virtual Warehouses deployed only one executor for the small group, ignoring the configured group size. The StatefulSet for this group also has a replica count of 1, which does not reflect the expected configuration.

The deployment now respects the configured small group size, ensuring the correct number of executors is deployed. This issue is now fixed.

DWX-19309: Prometheus pods fail due to unencrypted EBS volumes

Prometheus pods switched to EBS volumes failed to start because the default ebs-storageclass did not create encrypted volumes. This caused authorization errors, preventing the pods from running and impacting Grafana, Impala autoscaling, and Hue functionality.

The issue was resolved by ensuring EBS volumes used by Prometheus are encrypted, enabling successful pod deployment and restoring all affected services.

DWX-18448: Impala Virtual Warehouse size changes during updates

When using the update-vw command in the CDP CLI, the --template flag is required to specify the Virtual Warehouse size.

This issue is now fixed.

DWX-16875: Improved memory management for data publishing to Cloudera Observability dashboard

Data from Virtual Warehouses was intermittently not updating on the Cloudera Observability dashboard. Restarting the databus-producer deployment temporarily resolved the issue. The pod was being abruptly killed due to memory limits being exceeded, and the JVM settings were not optimized for dynamic resource allocation.

This issue is now fixed.

DWX-18932: Incorrect High Availability mode displayed for Impala Virtual Warehouses

When creating an Impala Virtual Warehouse (VW) in ACTIVE_PASSIVE High Availability (HA) mode with coordinator auto-shutdown enabled, the CLI incorrectly displayed the HA mode as ACTIVE_ACTIVE.

The logic for determining HA mode has been corrected. This ensures that ACTIVE_PASSIVE and ACTIVE_ACTIVE modes are correctly distinguished.

DWX-19172: Upgrade Cluster Autoscaler to version 1.30.2

The Kubernetes Cluster Autoscaler version is upgraded to 1.30.2 with the corresponding 9.37.0 chart version to ensure support for Amazon Elastic Kubernetes Service (EKS) 1.30.

Deprecation notices in Cloudera Data Warehouse Public Cloud

Review the features and functionalities that have been or will be removed or deprecated in this release of Cloudera Data Warehouse Public Cloud.

The ability to select a compute instance type during environment activation is deprecated

The ability to select a compute instance type during environment activation is deprecated and will be removed in a future release. In 1.9.3, you can select a compute instance type only when you use CDP CLI to activate an environment in Cloudera Data Warehouse. This option is no longer available on the UI.

The “CDW_CLUSTER_OVERPROVISIONER” entitlement is deprecated

The CDW_CLUSTER_OVERPROVISIONER entitlement allows you to specify the number of compute nodes available on standby in the underlying Kubernetes cluster in the Cloudera Data Warehouse web interface so that autoscaling and new Virtual Warehouse creation happen faster. This entitlement is deprecated and no longer functional.

Behavior changes

This release of the Cloudera Data Warehouse service on Cloudera Public Cloud has the following behavior changes:

Summary: Optimized ephemeral storage for Impala's Catalogd

Ephemeral storage for Catalogd is now optimized for shared nodes, automatically set to the lesser of 24GB or twice the Java heap (Xmx) plus a 1GB buffer.

Before this release: The storage limit of 512MB often caused container evictions due to insufficient space, especially during JVM heap dumps.

After this release: The ephemeral storage limit for Catalogd is now set dynamically based on the JVM heap size (Xmx), calculated as

```
2 * Xmx + Buffer space
```

This approach ensures adequate storage for heap dumps and minimizes the risk of container eviction, resulting in improved stability for Catalogd and other services sharing node resources.

Summary: Optimized memory management for databus-producer deployment

Memory management for the databus-producer deployment has been optimized to improve stability and ensure consistent data publishing to the Cloudera Observability dashboard.

Before this release:

- Data updates to the Cloudera Observability dashboard stopped intermittently.
- The pod could be abruptly terminated if memory usage exceeded the requested limit.
- JVM heap memory (-Xmx) was set to a hardcoded value, requiring manual adjustments during updates.

After this release:

- Memory requests and limits are now aligned to prevent abrupt pod termination.
- The JVM heap memory (-Xmx) is dynamically set to 70% of the memory resource limit.
- Enhanced JVM options allow automatic heap dumps in the event of out-of-memory errors, simplifying debugging.

Summary: Deterministic IDs for environments, Database Catalog, and Virtual Warehouses

Before this release: Every time you reactivated an Environment, the Environment ID changed and this led to a change in the path of the logs. The log folders and diagnostic bundle paths were written to the root folder of the storage instead of the configured paths.

After this release: Only the configured paths that are added during the Data Lake creation and prefixed by /tmp are used by the service. Also, the logs are prefixed with the Environment and Data Base Catalog name (wherever applicable) instead of the IDs to simplify manual searching through the logs. The base path of the log folder is also changed and users do not have to go to the external/sys.db/hive folders.

The various log files are still prefixed with the entity IDs in storage and will continue to remain in use to simplify the debugging process but navigating to the entity logs is now made easier.

Summary: Changes to the supported Azure instance types

Before this release: You could only select an instance while activating a Cloudera Data Warehouse Environment, and the Cloudera Data Warehouse Control Plane created the node pool from this specific instance. There was no option to select an instance type for a Virtual Warehouse.

After this release: You can select an instance during Environment activation (only through CDP CLI) and Virtual Warehouse creation (through UI and CDP CLI). The Cloudera Data Warehouse Control Plane requires the following instance types to be allowed, if they are available, in your Azure region to support instance type selection while creating a Virtual Warehouse:

- Standard_E16_v3
- Standard_E16ds_v4
- Standard_E16ads_v5
- Standard_E16ds_v5
- Standard_E16pds_v5

The Cloudera Data Warehouse Control Plane creates node pools using all these instance types. If you get an error such as the following while activating a Cloudera Data Warehouse Environment, then increase the quota of the problematic Azure instance type:

```
The VM size of Standard_E16ads_v5 is not allowed in your subscription in loc
ation 'eastus'.
```

You may request that your infrastructure team allow the Azure instance type by asking them to increase the resource quota. A minimal increase in the resource quota is sufficient for the instance to be allowed.

Note: You must understand the distinction between "available" instance types and "allowed" instance types. Cloudera Data Warehouse filters for regional instance type availability (available instance types) but cannot detect quota problems (allowed instance types).

Summary: Simplified Cloudera Data Warehouse Diagnostic Bundle Download Process

The diagnostic bundle download process in Cloudera Data Warehouse has been simplified for an improved user experience.

Before this release: Users had to select specific information types within time intervals or choose a custom time interval. Additionally, they needed to manually adjust options in "Collect For" to include or exclude types of logs for the bundle.

After this release: Users now directly access a simplified “Collect” option, eliminating the need for additional time interval and log selection adjustments.

October 4, 2024 - Hotfix

Review the fixed issues and changed behaviors in this hotfix release of Cloudera Data Warehouse Public Cloud and learn how to apply this hotfix.

Attention: To apply this hotfix and to upgrade the Cloudera Runtime version to 2024.0.18.2-4, navigate to the Virtual Warehouses page in the Cloudera Data Warehouse UI, identify the Virtual Warehouse that you want to upgrade, and click Upgrade for each Virtual Warehouse. It is not necessary to reactivate the Cloudera Data Warehouse Environment to apply this hotfix.

Fixed issues

This hotfix release of the Cloudera Data Warehouse service on Cloudera Public Cloud introduces a runtime fix.

DWX-18477: (Addendum) Merge query failures with reserved keywords as column names

If a reserved keyword is used as a column name in a query, you must escape the keyword by enclosing it within backticks (`). However, If you have merge queries involving Iceberg target tables that use reserved keywords as column names, the backticks are not retained when the merge queries are rewritten to a join query, and the query fails with a "SemanticException" error.

This issue is now fixed as part of an additional fix that was provided in HIVE-28282.

Known issues

Review the known issues in this release of the Cloudera Data Warehouse Public Cloud service.

DWX-19451: Cloudera Data Visualization restore job can fail with ignorable errors

After a successful Cloudera Data Visualization restoration job, the restore job could be in a failed state with the log displaying ignorable errors.

```
pg_restore: error: could not execute query: ERROR:  sequence "jo
bs_joblog_id_seq" does not exist
Command was: DROP SEQUENCE public.jobs_joblog_id_seq;
```

```
pg_restore: error: could not execute query: ERROR:  table "jobs_j
oblog" does not exist
Command was: DROP TABLE public.jobs_joblog;
pg_restore: error: could not execute query: ERROR:  sequence "
jobs_jobcontent_id_seq" does not exist
Command was: DROP SEQUENCE public.jobs_jobcontent_id_seq;
.......
.......
```

This issue occurs because the restore job issues commands to DROP all the objects that will be restored, and if any of these objects do not exist in the destination database, such ignorable errors are reported.

This has no functional impact on the restored Cloudera Data Visualization application. It is noticed that all the backed up queries, datasets, connections, and dashboards are restored successfully and Cloudera Data Visualization is available for new queries.

None.

DWX-19454: Default Database Catalog does not start in an Azure transparent proxy, private link, and private AKS setup

If you are running a transparent proxy with a private link and private AKS setup in your Azure environment, the Cloudera Data Warehouse server is unable to connect to the private AKS end point, and the default Database Catalog fails to start.

None.

August 15, 2024 - Hotfix

Review the fixed issues and changed behaviors in this hotfix release of Cloudera Data Warehouse Public Cloud.

Fixed issues

Review the issues fixed in this hotfix release of the Cloudera Data Warehouse service on Cloudera Public Cloud.

DWX-19003: Unable to set t-shirt size for a Cloudera Data Visualization instance

You could not set or configure a t-shirt size for a Cloudera Data Visualization instance while creating or editing it from Cloudera Data Warehouse. By default, any Cloudera Data Visualization instance created in Cloudera Data Warehouse release version 1.9.1 used the small t-shirt size. This issue has been resolved.

DWX-19034: Unable to create an Impala Virtual Warehouse with HA enabled on older runtime versions

Enabling HA on Impala Virtual Warehouses on runtime versions lower than 2024.0.18.0-206 caused errors such as the following on the catalogd pods:

```
I0805 13:33:15.674875     1 thrift-util.cc:198] TSocket::open()
 getaddrinfo() <Host: statestored-0 Port: 24000>Name or service
 not known
I0805 13:33:15.674933     1 thrift-client.cc:82] Couldn't open
 transport for statestored-0:24000 (Could not resolve host for
 client socket.)
```

This was because enabling HA on an Impala Virtual Warehouse enabled Statestore HA, too. Statestore HA is available starting with runtime 2024.0.18.0-206. This issue has been resolved.

DWX-19035: Service discovery does not work with Impala HA

If you created an Impala Virtual Warehouse with HA enabled in Cloudera Data Warehouse 1.9.1-b233, and then tried to link to it from Cloudera Data Visualization, you could not view the required Virtual Warehouse on the **Create New Data Connection** modal. This issue has been resolved.

IMPALA-13270: Addressing IllegalStateException in Complex Views post upgrade

When executing queries that generate runtime filters where the same column identifier appears repeatedly after upgrading to the 2024.0.18.0-206 runtime, you encountered the following error: IllegalStateException: null. This issue has been fixed.

IMPALA-13272: Stability Improvement for analytic functions on collections

The sorting process incorrectly included unnecessary elements, causing errors during array operations and leading to frequent query failures. This issue has been fixed to ensure only complete data entries are used in the sorting process. This prevents crashes and maintains stable execution of analytic functions on collections.

Known issues

Review the known issues in this release of the Cloudera Data Warehouse Public Cloud service.

DWX-19138: The option to enable Impala query logging is unavailable on the Cloudera Data Warehouse UI

You do not see the Log Impala queries option while creating or editing a Virtual Warehouse on the Cloudera Data Warehouse UI.

Use CDP CLI to create or edit a Virtual Warehouse and specify the --impala-query-log option to enable logging Impala queries.

DWX-19451: Cloudera Data Visualization restore job can fail with ignorable errors

After a successful Cloudera Data Visualization restoration job, the restore job could be in a failed state with the log displaying ignorable errors.

```
pg_restore: error: could not execute query: ERROR:  sequence "jo
bs_joblog_id_seq" does not exist
Command was: DROP SEQUENCE public.jobs_joblog_id_seq;
pg_restore: error: could not execute query: ERROR:  table "jobs_j
oblog" does not exist
Command was: DROP TABLE public.jobs_joblog;
pg_restore: error: could not execute query: ERROR:  sequence "
jobs_jobcontent_id_seq" does not exist
Command was: DROP SEQUENCE public.jobs_jobcontent_id_seq;
.......
.......
```

This issue occurs because the restore job issues commands to DROP all the objects that will be restored, and if any of these objects do not exist in the destination database, such ignorable errors are reported.

This has no functional impact on the restored Cloudera Data Visualization application. It is noticed that all the backed up queries, datasets, connections, and dashboards are restored successfully and Cloudera Data Visualization is available for new queries.

None.

Behavior changes

This release of the Cloudera Data Warehouse service on Cloudera Public Cloud has the following behavior changes:

Summary: Selecting a resource template replaces configuring Cloudera Data Visualization T-shirt sizes

Before this release: You could configure the following sizes for your Cloudera Data Visualization instance:

- Small (default), 8Gb
- Medium, 16Gb
- Large, 24Gb

After this release: You can configure the following Cloudera Data Warehouse resource template for your Cloudera Data Visualization instance:

- Default resources
- Medium resources
- Large resources

July 26, 2024 Release Notes

Review the new feaures, fixes, behavioral changes, and preview features in this release of Cloudera Data Warehouse Public Cloud.

What's new in Cloudera Data Warehouse Public Cloud

Review the new features introduced in this release of Cloudera Data Warehouse service on Cloudera Public Cloud.

- Cloudera Data Warehouse features
- Cloudera Data Warehouse on Azure environments
- Cloudera Data Warehouse on AWS environments
- Iceberg
- Hue
- Technical Preview features
- Behavior changes

What's new in Cloudera Data Warehouse Public Cloud

General availability of Virtual Warehouse and Database Catalog workload version selections

The Cloudera Data Warehouse UI now provides a list of workload versions that match your cluster from which you can select one during cluster installation. The Database Catalog list contains versions compatible with your Kubernetes version and your cluster environment (DWX version). The Virtual Warehouse list contains versions compatible with your Kubernetes version, your cluster environment (DWX version), and your Database Catalog version.

General availability of Impala workload-aware autoscaling

Workload-aware autoscaling allocates Impala Virtual Warehouse resources based on the workload that is running. You choose multiple executor group sets size based on your workload requirements, instead of the fixed executor group size of the previous auto-scaling implementation. This feature is now generally available. See Workload Aware Auto-Scaling in Impala.

Improved Impala Autoscaling Dashboard

You can now use the new Impala Autoscaling Dashboard to monitor Impala autoscaling in a warehouse that uses workload-aware autoscaling or the regular autoscaling. You can access the Impala Autoscaling Dashboard by going to the **Virtual Warehouse Details** page's Web UI tab, and clicking the Impala Autoscaler Web UI option. See About the Impala Autoscaling Dashboard.

Ability to forward Prometheus metrics from Cloudera Data Warehouse to an external endpoint

In this release, you can configure Prometheus in Cloudera Data Warehouse to push its metrics to an external endpoint, such as Prometheus, Grafana, Thanos, or some other endpoint. See Forwarding Prometheus metrics from Cloudera Data Warehouse to an endpoint.

Automatically backing up and restoring Cloudera Data Warehouse

This release adds more automation to back up and restore procedures for AWS and Azure environments and clarifies the documentation of the automatic, semi-automatic, and manual procedures.

To get the supported Kubernetes version for this release, you back up your old AWS or Azure environment and start up a new environment using the restoration process. The backup/restore feature saves your environment parameters, making it possible to recreate your environment with the same settings, URL, and connection strings you used in your previous environment.

Ability to configure Impala Statestore high availability

You can now configure high availability for Impala Statestore pods in a Virtual Warehouse, with active and passive modes ensuring continuity and reliability during failovers. See Configuring Impala Statestore high availability.

Downloading the UDF development package from Cloudera Data Warehouse UI

Introducing the ability to download the Impala UDF development package directly from the Cloudera Data Warehouse UI for enhanced convenience and integration, see Building and deploying UDFs.

PostgreSQL replaces SQLite database for Grafana in Cloudera Data Warehouse Public Cloud

The file-based SQLite database for Grafana has been replaced with PostgreSQL database, providing a more robust experience. You must deactivate and reactivate your environment in Cloudera Data Warehouse to use this feature.

What's new in Cloudera Data Warehouse on Azure environments

Azure AKS 1.29 upgrade

Cloudera supports the Azure Kubernetes Service (AKS) version 1.29. In 1.9.1-b233 (released July 26, 2024), when you activate an environment, Cloudera Data Warehouse automatically provisions AKS 1.29. To upgrade to AKS 1.29 from an earlier version of Cloudera Data Warehouse, you must backup and restore Cloudera Data Warehouse. To avoid compatibility issues between Cloudera Data Warehouse and AKS, upgrade to version 1.29.

Note: Using the Azure CLI or Azure portal to upgrade the AKS cluster is not supported. Doing so can cause the cluster to become unusable and can cause downtime. For more information about upgrading, see Upgrading an Azure Kubernetes Service cluster for Cloudera Data Warehouse.

Addition of new Azure instance types

This release offers the selection of the Standard_E16pds_v5 Azure Virtual Machine, an AKS Ampere® Altra® Arm-based instance type for an Impala Virtual Warehouse. For more information about using the instance type, see Activating an Azure environment from Cloudera Data Warehouse.

Cloudera Data Warehouse provisions Azure Database for PostgreSQL - Flexible Server

Starting with this release, Cloudera Data Warehouse provisions Azure Database for PostgreSQL - Flexible Server instead of Azure Database for PostgreSQL - Single Server. See Enabling a private Cloudera Data Warehouse environment.

What's new in Cloudera Data Warehouse on AWS environments

Amazon EKS 1.29 upgrade

Cloudera supports the Amazon Elastic Kubernetes Service (EKS) version 1.29. In 1.9.1-b233 (released July 26, 2024), when you activate an environment, Cloudera Data Warehouse automatically provisions EKS 1.29. To upgrade to EKS 1.29 from an earlier version of CDW, you

must backup and restore CDW. To avoid compatibility issues between Cloudera Data Warehouse and EKS, upgrade to version 1.29. See Upgrading Amazon Kubernetes Service (EKS).

Note about the impact of AWS RDS root certificate rotation in 2024

A Cloudera Data Warehouse Cluster RDS does not use certificate verification for connections to the Cloudera Data Warehouse. Therefore you are not directly impacted by certificate expiration for your Cloudera Data Warehouse Cluster RDS. You can either choose to clear the warnings or rotate the certificate.

To rotate the certificate for the Cloudera Data Warehouse Cluster RDS, follow the step outlined by AWS in Rotating your SSL/TLS certificate to update the certificate. There should be no impact on Cloudera Data Warehouse because the Cloudera Data Warehouse Cluster RDS should not be restarted, Postgres RDS has SupportsCertificateRotationWithoutRestart=true.

For the Datalake RDS, follow instructions shared by the Datalake account team to update the certificate. There maybe some impact to Cloudera Data Warehouse while restarting the Datalake, such as query failures or delays. This could happen because services such as Ranger, Knox, and FreeIPA might be unavailable during this period.

Addition of new AWS instance types

This release offers the selection of the r6gd.4xlarge and r7gd.4xlarge Arm-based instance types for an Impala Virtual Warehouse. For more information about using the instance type, see Activating an AWS environment from Cloudera Data Warehouse.

Ability to use envelope encryption for EKS secrets

Envelope encryption is now added for EKS Secrets through Cloudera Data Warehouse KMS Key by default. See Encrypt Kubernetes secrets with AWS KMS on existing clusters.

What's new in Iceberg on Cloudera Data Warehouse Public Cloud

Cloudera support for Iceberg version 1.4.3

The Apache Iceberg component has been upgraded from 1.3.0 to 1.4.3.

Support for Iceberg data compaction

You can compact Iceberg tables and optimize them for read operations from Hive and Impala. Compaction is an essential table maintenance activity that creates a new snapshot, which contains the table content in a compact form. See Iceberg data compaction.

SQL support for querying Iceberg metadata tables

Apache Iceberg stores extensive metadata for its tables. From Hive and Impala, you can query the metadata tables as you would query a regular table. For example, you can use projections, joins, filters, and so on. See Query metadata tables feature.

What's new in Hue on Cloudera Data Warehouse Public Cloud

General availability (GA) of the SQL AI Assistant

Hue leverages the power of Large Language Models (LLM) to help you generate SQL queries from natural language prompts and also provides options to optimize, explain, and fix queries, promoting efficient and accurate practices for accessing and manipulating data. You can use several AI services and models such as OpenAI's GPT service, Amazon Bedrock, and Azure's OpenAI service to run the Hue SQL AI assistant.

- To learn more about the supported models and services, limitations, and what data is shared with the LLMs, see About the SQL AI Assistant in Cloudera Data Warehouse.
- To set up and enable the SQL AI Assistant, see About setting up the SQL AI Assistant in Cloudera Data Warehouse.
- To see how to generate, edit, explain, optimize, and fix queries, see Starting the SQL AI Assistant in Hue.

Introduction of task server in Hue and significant improvement in the file upload functionality

A new **Task Server** page has been added to the Hue web interface. The Hue task server enables the following functionalities:

- It improves the file-upload experience, allowing you to upload multiple files up to 5 GB each in parallel.
- It helps you to schedule tasks to clean up Hue documents and the /tmp directory, improving cluster maintenance experience and performance.

See About the Hue task server in Cloudera Data Warehouse.

Related Information

Preview features in Cloudera Data Warehouse Public Cloud

Preview features in Cloudera Data Warehouse Public Cloud

This release of the Cloudera Data Warehouse service on Cloudera Public Cloud introduces this technical preview.

Note: Technical previews are considered under development. Do not use these features in production environments.

Enabling the Hive Virtual Warehouse to spill to an EBS volume (Preview)

To prevent failures when query data exceeds memory capacity, you spill data to an EBS volume. The data spills to the Amazon gp3 Elastic Block Store (EBS) volumes. You select the Additional LLAP Spill Disk (EBS) option when you create a Hive Virtual Warehouse. Cloudera automatically provisions the gp3 volume type for spilling Hive queries when you create or reactivate a Hive Virtual Warehouse on the latest Cloudera Data Warehouse environment. For more information about EBS volumes, see Amazon documentation. Using the EBS volume incurs cost.

Note: You cannot enable the option to spill data to EBS volume after creating a Virtual Warehouse.

Improvements to the shared Hue service (Preview)

- Name change from Query Editor to Shared Hue Service in the left navigation pane in the Cloudera Data Warehouse UI.
- Shared Hue service supports upgrade and rebuild operations similar to other Cloudera Data Warehouse components.
- Added a one-time option to copy saved queries and query history while creating a shared Hue service instance.

For more information, see Deploying shared Hue service in Cloudera Data Warehouse Public Cloud (Preview).

Ability to log and manage Impala workloads (Preview)

Cloudera Data Warehouse provides you the option to enable logging Impala queries on an existing Virtual Warehouse or while creating a new Impala Virtual Warehouse. By logging the Impala queries in Cloudera Data Warehouse, you gain increased observability of the workloads running on Impala, which you can use to improve the performance of your Impala Virtual Warehouses.

This feature represents a significant enhancement to query profiling capabilities. You can have Impala archive crucial data from each query's profile into dedicated database tables known as the query history table and live query table. These tables are part of the sys database and are designed to store valuable information that can later be queried using any Impala client, providing a consolidated view of reports from previously executed queries.

For more information, see Impala workload management in Cloudera Data Warehouse Public Cloud (Preview).

Introducing AI-enhanced UDF development package in Impala (Preview)

- A built-in AI function, ai_generate_text, enabling direct access to Large Language Models (LLMs) from SQL queries by inputting a prompt and retrieving the response.

- This integration into existing workflows simplifies the process, reducing complexity and enhancing the user experience, allowing for quicker setup and deployment of UDFs in Impala.

For more information, see Advantages and use cases of Impala AI functions (Preview).

Support for Impala external JDBC data sources (Preview)

Apache Impala now supports reading from external JDBC data sources. An external JDBC table represents a table or a view in a remote RDBMS database or another Impala cluster. Using external JDBC tables, you can connect Impala to a database, such as MySQL, PostgreSQL, or another Impala cluster and read the data in the remote tables.

For more information, see Using Impala to query external JDBC data sources (Preview).

AWS environment permissions support for the EKS start/stop feature (Preview)

AWS permissions have been expanded to support the Elastic Kubernetes Service (EKS) start/stop feature:

- rds:StartDBInstance
- rds:StopDBInstance
- rds:DescribeDBInstances
- autoscaling:DescribeAutoScalingGroups

Ability to select an instance type for Virtual Warehouses (Preview)

You can now specify AWS or Azure instance types, such as r6id.4xlarge or Standard_E16_v3, that you want to use for your Virtual Warehouse while creating a Virtual Warehouse. You are no longer confined to use the instance types that were specified while activating the environment in Cloudera Data Warehouse. See Creating a Virtual Warehouse with ARM compute instance types using CDP CLI.

Note:

- In Cloudera Data Warehouse 1.9.1-b233 (released July 26, 2024), instance type selection while creating a Virtual Warehouse is only supported using BETA CDP CLI. See CDP CLI (BETA) documentation. You must install and use BETA CDP CLI version 0.9.119.
- You can continue to select an instance type while activating an environment in Cloudera Data Warehouse using both the UI and CDP CLI. The ability to do this will be deprecated in a future release.

Impala support for reading Iceberg equality deletes for NiFi (Preview)

Cloudera supports row-level deletes, and starting with this release you can read equality deletes from Impala with suport added for Apache NiFi. See the Delete data feature.

Related Information

What's new in Cloudera Data Warehouse Public Cloud

Cloudera preview features documentation

Known issues in Cloudera Data Warehouse Public Cloud

Review the known issues in this release of the Cloudera Data Warehouse service on Cloudera Public Cloud.

- Known issues identified in the July 26, 2024 release on page 36
- Known issues identified before the July 26, 2024 release on page 38
- Technical Service Bulletins on page 49

Known issues identified in the July 26, 2024 release

DWX-19451: Cloudera Data Visualization restore job can fail with ignorable errors

After a successful Cloudera Data Visualization restoration job, the restore job could be in a failed state with the log displaying ignorable errors.

```
pg_restore: error: could not execute query: ERROR:  sequence "jo
bs_joblog_id_seq" does not exist
Command was: DROP SEQUENCE public.jobs_joblog_id_seq;
pg_restore: error: could not execute query: ERROR:  table "jobs_j
oblog" does not exist
Command was: DROP TABLE public.jobs_joblog;
pg_restore: error: could not execute query: ERROR:  sequence "
jobs_jobcontent_id_seq" does not exist
Command was: DROP SEQUENCE public.jobs_jobcontent_id_seq;
.......
.......
```

This issue occurs because the restore job issues commands to DROP all the objects that will be restored, and if any of these objects do not exist in the destination database, such ignorable errors are reported.

This has no functional impact on the restored Cloudera Data Visualization application. It is noticed that all the backed up queries, datasets, connections, and dashboards are restored successfully and Cloudera Data Visualization is available for new queries.

None.

DWX-19003: Unable to set t-shirt size for a Cloudera Data Visualization instance

You cannot set or configure a t-shirt size for a Cloudera Data Visualization instance while creating or editing it from Cloudera Data Warehouse. By default, any Cloudera Data Visualization instance created in Cloudera Data Warehouse release version 1.9.1 uses the “small” compute instance size.

None.

DWX-18950: Hue data restoration fails on Azure Cloudera Data Warehouse clusters having Cloudera Data Warehouse version 1.8.7 or earlier

If you backup your Cloudera Data Warehouse cluster running Cloudera Data Warehouse version 1.8.7 or earlier and if you perform the restore operation without reactivating the latest Cloudera Data Warehouse environment, you might encounter Hue restoration job failure.

None.

DWX-18843: Unable to read Iceberg table from Hive Virtual Warehouse

If you have used Apache Flink to insert data into an Iceberg table that is created from Hive, you cannot read the Iceberg table from the Hive Virtual Warehouse.

Add the engine.hive.enabled table property through the Hive beeline and set the value to "true". You can add this table property either while creating the Iceberg table or use the ALTER TABLE statement to add the table property.

DWX-18489: Hive compaction of Iceberg tables results in a failure

When Cloudera Data Warehouse and Cloudera Data Hub are deployed in the same environment and use the same Hive Metastore (HMS) instance, the Cloudera Data Hub compaction workers can inadvertently pick up Iceberg compaction tasks. Since Iceberg compaction is not yet supported in the latest Cloudera Data Hub version, the compaction tasks will fail when they are processed by the Cloudera Data Hub compaction workers.

In such a scenario where both Cloudera Data Warehouse and Cloudera Data Hub share the same HMS instance and there is a requirement to run both Hive ACID and Iceberg compaction jobs, it is recommended that you use the Cloudera Data Warehouse environment for these jobs. If you want to run only Hive ACID compaction tasks, you can choose to use either the Cloudera Data Warehouse or Cloudera Data Hub environments.

If you want to run the compaction jobs without changing the environment, it is recommended that you use Cloudera Data Warehouse. To avoid interference fromCloudera Data Hub, change the value of the hive.compactor.worker.threads Hive Server (HS2) property to '0'. This ensures that the compaction jobs are not processed by Cloudera Data Hub.

1. In Cloudera Manager, click Clusters Hive Configuration to navigate to the configuration page for HMS.
2. Search for hive.compactor.worker.threads and modify the value to '0'.
3. Save the changes and restart the Hive service.

DWX-18854: Compaction cleaner configuration

The compaction cleaner is turned off by default in the Cloudera Data Warehouse database catalog, potentially causing compaction job failures.

To enable the compactor on the Hive Metastore (HMS) instance of the Cloudera Data Warehouse database catalog, set the hive.compactor.cleaner.on property to "true".

DWX-12703: Hue connects to only one Impala coordinator in Active-Active mode

You may not see all Impala queries that have run on the Virtual Warehouse from the **Impala** tab on the Hue **Job Browser**. You encounter this on an Impala Virtual Warehouse that has Impala coordinator configured in an active-active mode. This happens because Hue fetches this information from only one Impala coordinator that is active.

None. You can view the query history from the **Impala Queries** tab on the **Job Browser** page, because the information is fetched from the Hue Query Processor.

Known limitation: Cloudera Data Warehouse does not support S3 Express One Zone buckets

Cloudera does not recommend deploying the Data Lake on S3 Express One Zone buckets. Cloudera Data Warehouse cannot read content present in the S3 Express One Zone buckets. The following limitations apply when using S3 Express buckets:

- You can only use S3 Express buckets with Cloudera Data Hub clusters running Runtime 7.2.18 or newer. Data services do not support it, currently.
- S3 Express buckets may not be used for logs and backups.

Known issues identified before the July 26, 2024 release

DWX-15112: Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

None.

DWX-17455: ODBC client using JWT authentication cannot connect to Impala Virtual Warehouse

If you are using the Cloudera ODBC Connector and JWT authentication to connect to a Cloudera Data Warehouse Impala Virtual Warehouse where the Impala coordinators are configured for high availability in an active-active mode, the connection results in a "401 Unauthorized" error. The error is not seen on Impala Virtual Warehouses that have an active-passive high availability or where high availability is disabled.

In the ODBC Data Source Name (DSN), make the following changes:

1. Ensure that SSL is enabled by setting SSL=1.
2. Remove authentication by setting AuthMech=0.
3. Remove the JWTString parameter.
4. Add the parameter http.header.Authorization and set it to the word "Bearer" followed by a space and the JWT string

DSN configuration before the workaround

```
SSL=1
AuthMech=8
JWTString=full_jwt_text
```

DSN configuration after the workaround

```
SSL=1
AuthMech=0
http.header.Authorization=Bearer full_jwt_text
```

HIVE-28055: Merging Iceberg branches requires a target table alias

Hive supports only one level of qualifier when referencing columns. In other words only one dot is accepted. For example, select table.col from ...; is allowed. select db.table.col is not allowed. Using the merge statement to merge Iceberg branches without a target or source table alias causes an exception:

```
org.apache.hadoop.hive.ql.parse.SemanticException: ... Invalid t
able alias or column reference ...
```

Use an alias, for example t, for the target table.

```
merge into mydb.target.branch_branch1 t using mydb.source.branch
_branch1 s on t.id = s.id when matched then update set value = '
matched';
```

DWX-17620: Folder having special characters in its name is not accessible in ABFS

In Cloudera Data Warehouse Public Cloud, from a Virtual Warehouse going to the Azure Blob Filesystem (ABFS), creating a folder, and then performing an action such as Move, causes an error as shown in the following example:

```
Cannot access: abfs://data-files/user/hrt_qa/~@$&()*!+'=;. 404 C
lient Error: The specified path does not exist.
```

None.

Branch FAST FORWARD does not work as expected

The Apache Iceberg spec indicates you can use either one or two arguments to fast forward a branch. The following example shows using two arguments:

```
ALTER TABLE <name> EXECUTE FAST-FORWARD 'x' 'y'
```

However, omitting the second branch name, does not work as documented by Apache Iceberg. The named branch is not fast-forwarded to the current branch. An exception occurs at the Iceberg level.

You must use two arguments to the EXECUTE FAST FORWARD feature to forward a branch.

DWX-17613: Generic error message is displayed when you click on the directory you don't have access to on a RAZ cluster

You see the following error message when you click on an ABFS directory to which you do not have read/write permission on the ABFS File Browser in Hue: There was a problem with your request. This message is generic and does not provide insight into the actual issue.

None.

DWX-17109: ABFS File Browser operations failing intermittently

You may encounter intermittent issues while performing typical operations on files and directories on the ABFS File Broswer, such as moving or renaming files.

None.

CDPD-27918: Hue does not automatically pick up RAZ HA configurations

On a Cloudera Public Cloud environment in which you have configured RAZ in High Availability mode, Hue in Cloudera Data Warehouse does not pick up all the RAZ host URLs automatically. Therefore, if a RAZ instance to which Hue is connected goes down, Hue becomes unavailable.

You must manually add comma-separated RAZ instances in the Hue Advanced Configuration Snippet.

1. Log in to the Cloudera Management Console as an Administrator.
2. Go to Environment Data Lake and open Cloudera Manager for your environment.
3. Go to Clusters Ranger RAZ service Instances RAZ server Processes and note the value of the fs.s3a.ext.raz.rest.host.url property from the core-site.xml file. You need this to specify the value of the api_url property in the Hue configuration.

   For Azure environments, note the value of the fs.azure.ext.raz.rest.host.url property.

   For AWS and GCS environments, note the value of the fs.s3a.ext.raz.rest.host.url property.
4. Go to Cloudera Data Warehouse Virtual Warehouse ⋮ Edit CONFIGURATIONS and select the hue-safety-valve from the Configuration files dropdown menu.
5. Add the following lines in the hue-safety-valve field:

```
[desktop]
[[raz]]
is_enabled=true
api_url=https://[***INSTANCE-1***]:6082/,https://[***INSTANCE-
2***]:6082/
```

6. Click Apply Changes.

CDPD-66779: Partitioned Iceberg table not getting loaded with insert select query from Hive

If you create a partitioned table in Iceberg and then try to insert data from another table as shown below, an error occurs.

```
insert into table partition_transform_4 select t, ts from vector
tab10k;
```

Use the CLUSTER BY clause on the partitioned column to insert data. For example:

```
insert into table partition_transform_4 select t, ts from t1 clu
ster by ts;
```

DWX-17703: Non-HA Impala Virtual Warehouse on a private Azure Kubernetes Service (AKS) setup fails

When 'Refresh' and 'Stop' operations run in parallel, Impala might move into an error state. The Refresh operation might think that Impala is in an error state as the coordinator pod is missing.

Rebuild the Impala Virtual Warehouse or restart it using the CLI.

DWX-14923: After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

None.

DWX-15145: Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an environment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several different popups can occur, including the following one:

worker subnets should be selected from 3 different availability zones. got: 0

None.

DWX-15144: Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

None.

DWX-13103: Cloudera Data Warehouse environment activation problem

When Cloudera Data Warehouse environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Cloudera Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for Cloudera Data Warehouse Reduced Permissions Mode

If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
```

```
        Type: String
        Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing Cloudera Data Warehouse environments

You may see the following error while trying to apply tags to Virtual Warehouses in an existing Cloudera Data Warehouse environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Incorrect diagnostic bundle location

The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

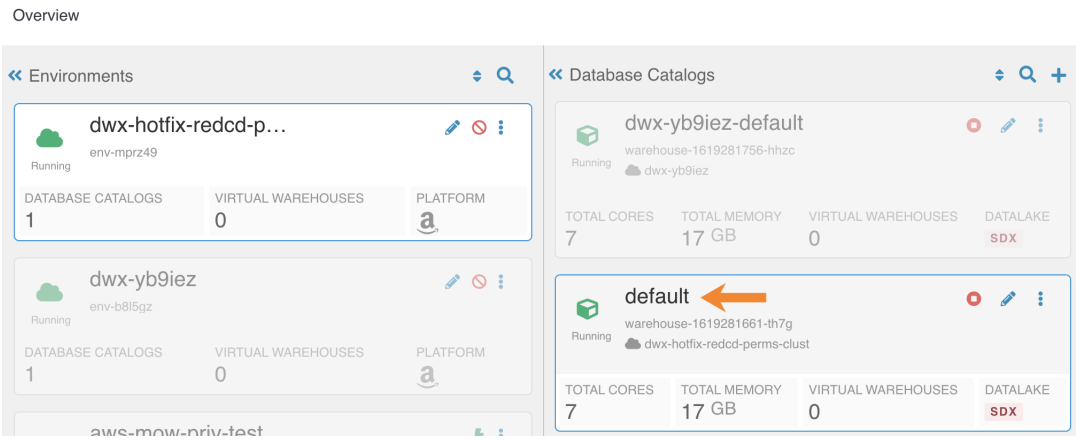

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

None.

DWX-15064: Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

None.

DWX-14452: Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click CONFIGURATIONS Query executor and select the env configuration file.

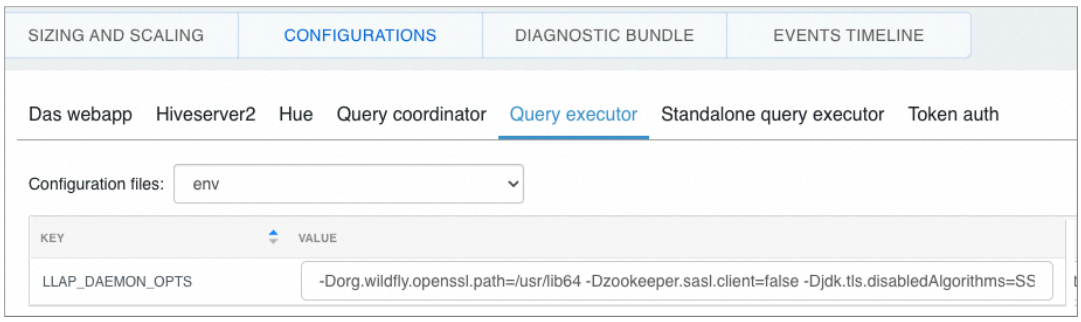

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

CDPD-40730: Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of Cloudera Data Warehouse, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

None.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Only use Beeline clients from Cloudera Runtime version 7.0.1.0 or later.

DWX-16899: Error while viewing Impala job status on the mini Job Browser

When you click on the application ID after submitting an Impala query on Hue that is running on the environment level, you may notice the following error: 401 Client Error: Unauthorized for url: http://coordinator.impala-xyz.svc.cluster.local:25000/queries?json=true Must authenticate with Basic authentication. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16895: Incorrect status of Hue pods when you edit the Hue instance properties

When you update a configuration of a Hue instance that is deployed at the environment level, such as increasing or decreasing the size of the Hue instance, you see a success message on the Cloudera Data Warehouse UI. After some time, the status of the Hue instance also changes from “Updating” to “Running”. However, when you list the Hue pods using kubectl, you see that not all backend pods are in the running state–a few of them are still in the init state.

None. The pods come up successfully eventually after a sufficient time has passed.

DWX-16863: The upgrade button is present on the Cloudera Data Warehouse UI, but Hue upgrades are not supported

You see the Upgrade button on the **Query Editor** page in the Cloudera Data Warehouse UI when Hue is deployed at the environment level. However, on Cloudera Data Warehouse version 1.8.1, upgrading the Hue instance that is deployed at the environment level is not supported.

None.

DWX-16893: A user can see all the queries in Job browser

In a Hue instance deployed at the environment level, by design, the Hue instances must not share the saved queries and query history with other Hue instances even for the same user. However, a

logged in user is able to view all the queries executed by that user on all the Virtual Warehouses on a particular Database Catalog.

None.

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

None.

DWX-14927: Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

None.

DWX-14968: Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Refresh the page, or alternatively start a new session.

DWX-15115: Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

None.

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447: Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your Cloudera environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

DWX-17210, DWX-13733: Timeout issue querying Iceberg tables from Hive

When querying Iceberg tables from Hive, the queries can faile due to a timeout issue.

1. Add the following configurations to hadoop-core-site for the Database Catalog and the Virtual Warehouse.
   - fs.s3.maxConnections=1000
   - fs.s3a.connection.maximum=1000
2. Restart the Database Catalog and Virtual Warehouse.

DWX-15014: Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

None.

DWX-14163: Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in Cloudera. In addition to these, the DECIMAL type is not supported in this release.

None.

DEX-7946: Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13062: Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276: Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in Cloudera Data Warehouse Public Cloud) may encounter an exception

IMPALA-11045: Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statementINVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statementINVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

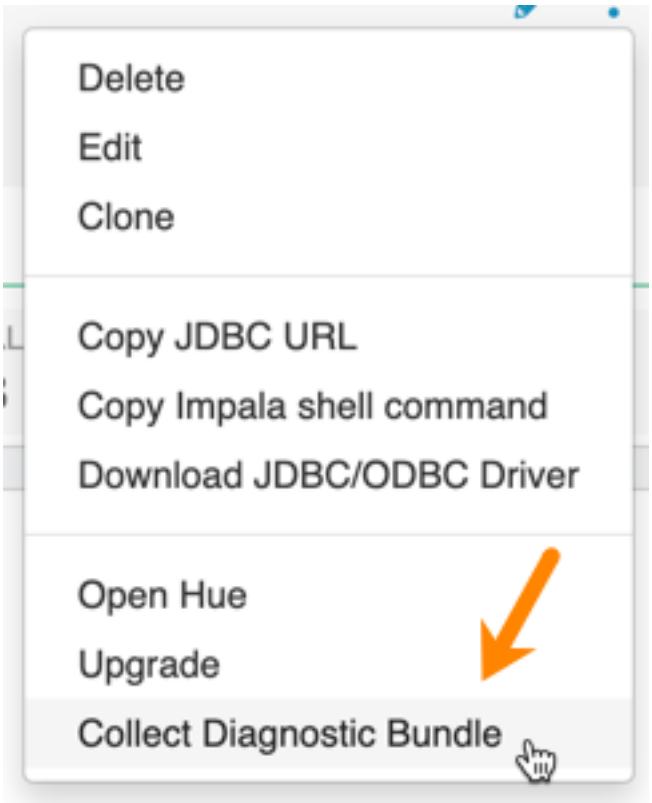

None.

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

None.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

None.

Technical Service Bulletins

TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete

Cloudera Data Warehouse customers using the Cloudera Data Warehouse Automated Backup/Restore feature will encounter an issue with the restoration versions of Cloudera Data Visualization older than 7.1.6.2-3 due to a schema change in this release. If the Backup was taken from an older Cloudera Data Warehouse environment that contained a version of Cloudera Data Visualization older than 7.1.6.2-3, the Restore procedure will succeed. Though once the user opens the Cloudera Data Visualization Queries tab, the user could encounter the error message: “column jobs_jobschedule.owner_id does not exist…”

None. For the latest update on this issue see the corresponding Knowledge article: TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete.

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in Cloudera Data Warehouse Public Cloud) may encounter an exception

The Cloudera Data Warehouse Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in Cloudera Data Warehouse Public Cloud) may encounter an exception

Fixed issues in Cloudera Data Warehouse Public Cloud

Review the issues fixed in this release of the Cloudera Data Warehouse service on Cloudera Public Cloud.

DWX-17794: Custom tags specified on a Virtual Warehouse do not work

Earlier, when you added tags while creating a Virtual Warehouse, the tags did not show on the AWS console. This issue has been fixed.

DWX-15859: Issue backing up and restoring Workload Aware Auto-Scaling (WAAS)

You could not use the Cloudera Data Warehouse backup and restore process to restore the WAAS configurations and you had to manually recreate your Virtual Warehouse by setting the right configurations. This issue has been fixed.

DWX-17620: The folder having special characters in its name is not accessible in ABFS

Earlier, creating a folder and then performing an action such as Move from a Virtual Warehouse going to the Azure Blob Filesystem (ABFS) caused an “404 Client Error: The specified path does not exist” error. This issue has been resolved.

HIVE-28006: Incremental rebuild accuracy for materialized views

Optimized the accuracy of representing incremental rebuild availability for materialized views containing aggregate functions in their definition query.

HIVE-27775: Fix for partition metadata fetch issue during DST shift

Fixes fetching partition metadata from Hive Metastore using Java Data Objects (JDO) by timestamp now does not provide incorrect results in daylight saving time (DST) shift in partition pruning.

HIVE-27924: Materialized view rebuild issue with delete operations

Addressed data correctness issues when rebuilding a materialized view incrementally in scenarios involving delete operations in one of its source tables. This is evident when the materialized view definition query includes joins and aggregates.

IMPALA-12809: Metadata query problem from Impala

When querying metadata from Impala on a cluster with more than a few nodes, the query might fail with a null pointer exception.

The fix addresses issues noticed while querying metadata from Impala on a cluster with more than a few nodes. The change forces Iceberg metadata scanner fragments to be scheduled on the coordinator. A new flag is introduced in the TPlanFragment thrift struct and if the value is 'true', then the fragment is always scheduled on the coordinator.

IMPALA-12742: DELETE/UPDATE Iceberg table partitioned by DATE fails

Fixed data correctness issue by ensuring that Impala CatalogD is able to corretcly parse identity-partitioned DATE values.

IMPALA-13138: Never smallify existing StringValue objects, only new ones during DeepCopy

This issue has been fixed.

Technical Service Bulletins

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion.

Behavior changes in Cloudera Data Warehouse Public Cloud

This release of the Cloudera Data Warehouse service on Cloudera Public Cloud has the following behavior changes:

Summary: Changes to the Azure Kubernetes Service (AKS) managed identities

Before this release: Hive inherited the Storage Blob Data Owner role from the Data Lake.

After this release: You must explicitly grant the Storage Blob Data Owner role to the AKS identity on your Data Lake storage account. This permission is required for activating an environment in Cloudera Data Warehouse.

Summary: Overlay Networking replaces kubenet in Azure environments

Before this release: To avoid IP address exhaustion, you enable the kubenet networking feature when you activate an Azure environment to use with Cloudera Data Warehouse.

After this release: To avoid IP address exhaustion, by default, Cloudera Data Warehouse uses CNI overlay networking and replaces kubenet networking. You use the CDP CLI to set CIDR (Classless Inter-Domain Routing) ranges. For more information, see Overlay networking.

Summary: Selecting a resource template replaces configuring Database Catalog T-shirt sizes

Before this release: You could configure the following sizes of the Java Heap for your Database Catalog workload:

- Small (default), 8Gb
- Medium, 16Gb
- Large, 24Gb

After this release: You can configure a Cloudera Data Warehouse resource template, which fulfills all requirements for a Database Catalog for the type of Virtual Warehouse you have, for example a Hive or Impala Virtual Warehouse:

- Default resources
- Medium resources
- Large resources

For more information, see Creating a Database Catalog.

Summary: Impala Autoscaling Dashboard replaces the older Impala Autoscaler Web UI

Before this release: The Impala Autoscaler Web UI provided the tools to monitor how Impala Virtual Warehouses autoscaled.

After this release: You can monitor Impala autoscaling in a warehouse that uses workload-aware autoscaling or the regular autoscaling using the new Impala Autoscaling Dashboard.

Summary: Apache Iceberg component version

Before this release: The supported Iceberg version was 1.3.0.

After this release: The supported Iceberg version is 1.4.3.

Summary: CDW_VERSIONED_DEPLOY entitlement

Before this release: To upgrade Azure Kubernetes Service for Cloudera Data Warehouse to certain workload versions, you needed the CDW_VERSIONED_DEPLOY entitlement.

After this release: The status of Virtual Warehouse and Database Catalog workload version selections is now General availability (GA). The entitlement is no longer needed. For more information, see What's New in this release.

Summary: Uploading files to object store using Hue

Before this release: The Hue server performed the file upload operations. The maximum file size you could upload to S3 buckets or ADLS Gen2 containers was 2 GB. When multiple users performed major upload operations, it impacted the performance of Hue since all the backend calls were synchronous. The Hue server became slow as the upload blocked Hue services. You may have worked around it by disabling the upload from the main Hue server and having another setup with upload enabled and redirecting the end users wanting to upload files to the second instance.

After this release: The files are uploaded using an asynchronous task queue or job queue in Hue. This improves performance and allows you to upload files as large as 5 GB.

Summary: Change in the option name to enable Hue at the environment level

Before this release: The option to enable the shared Hue service (Hue at environment level) from the **Environment Details** page was called Enable Query Editor.

After this release: The option has been renamed to Enable Shared Hue Service.

Summary: Upgrading and rebuilding the shared Hue service

Before this release: In Cloudera Data Warehouse version 1.8.1, you could not upgrade the Hue instances that are deployed at the environment level.

After this release: You can upgrade and rebuild the shared Hue service.

Unsupported releases

Overview of new features, enhancements, known issues, fixed issues, and changed behavior introduced in earlier, and now unsupported releases of Cloudera Data Warehouse.

May 2, 2024 - Hotfix

Review the fixed issues and changed behaviors in this hotfix release of Cloudera Data Warehouse Public Cloud.

Fixed issues

This release of the Cloudera Data Warehouse service on Cloudera Public Cloud introduces these changes.

Upgrade your Virtual Warehouse to get these hotfixes:

- HIVE-28051 A new housekeeping task is added to clean up local folders on Hive Virtual Warehouse startup and periodically after startup, resolving the issue with disk overflow.

  When a Hive LLAP daemon crashes, it can leave behind unnecessary files in the LLAP local directories. These files have to be cleaned periodically without which Hive queries can fail indicating an invalid disk error exception.

  The following properties are introduced to periodically delete the unnecessary files:

  hive.llap.local.dir.cleaner.cleanup.interval

  Specifies the time interval based on which the LocalDirCleaner service in LLAP daemon checks for stale or old files.

  Default value: 2 hours

  hive.llap.local.dir.cleaner.file.modify.time.threshold

  Specifies the threshold time for the LocalDirCLeaner service. If a file is older than the threshold time, the file is deleted.

  Default value: 24 hours
- IMPALA-12827 Fixes event processing errors when write IDs of an AbortTxnEvent are cleaned up by the HMS cleaner housekeeping threads.
- IMPALA-12831 Fix race condition when a table is being invalidated and updated concurrently.
- IMPALA-12832 Implicitly invalidates a table instead of resulting in an ERROR state during event processing.
- IMPALA-12835 Fix event processor, which is not synching file metadata for non-partitioned ACID tables when incremental refresh on transactional tables is turned off.

  Note:

  - This issue only occurs when hms_event_incremental_refresh_transactional_table is set to 'false'.
  - This issue occurs on non-partitioned tables. Partitioned tables are not affected.
- IMPALA-12851 Fix issue of txnId not being added to tableWriteIds mapping in Catalog.
- IMPALA-12855 Fixes the possibility of encountering a NullPointerException when refreshing a partition that has just been dropped.
- IMPALA-12356 Fix for incorrect identification on self events.
- IMPALA-12969 Fix conditional JVM heap leak in array allocation on deserialization failures.

Behavior changes

This release of the Cloudera Data Warehouse service on Cloudera Public Cloud has the following behavior changes:

Summary:

Change in value for the write.delete.mode property

Before this release:

The value for the write.delete.mode property was set to 'merge-on-read'.

After this release:

The default value for the write.delete.mode property is changed to 'copy-on-write'.

This change might result in issues while deleting Iceberg table records.

If you want to continue to use the 'merge-on-read' mode for new Iceberg tables, perform the following steps:

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. From the Cloudera Data Warehouse service, click Database Catalogs, locate your Database Catalog and then click ⋮ Edit .
3. In the **Database Catalogs detail** page, click Configurations Metastore and select the hive-site configuration file.

4. Click + and add the following configuration key and key value:

```
write.delete.mode=merge-on-read
```

5. Click Apply Changes.
6. Click Virtual Warehouses, locate your virtual warehouse and then click ⋮ Edit .
7. In the **Virtual Warehouse Details** page, click Configurations Hiveserver2 and select the hive-site configuration file.
8. Click + and add the following configuration key and key value:

```
write.delete.mode=merge-on-read
```

9. Click Apply Changes.

If you want to set 'merge-on-read' for older Iceberg tables that were created before upgrading to this Cloudera Data Warehouse version, perform the following steps:

1. Run the following queries to modify the table properties of the old Iceberg tables:

```
ALTER TABLE [***OLD TABLE NAME***] SET TBLPROPERTIES('write.
on.delete'='dummy');
ALTER TABLE [***OLD TABLE NAME***] SET TBLPROPERTIES('write
.on.delete'='merge-on-read');
```

March 26, 2024 - Hotfix

Review the known issues and fixed issues in this hotfix release of Cloudera Data Warehouse on Public Cloud.

Fixed issues

This hotfix release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

Note: Re-activate your Virtural Warehouse to get these hotfixes.

IMPALA-12787: Concurrently running UPDATE and DELETE operations can no longer revive deleted rows

This issue is fixed.

Technical Service Bulletins

TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

TSB 2024-750: CDW activation resulting in inaccessible workload endpoints on AWS

CDW activation no longer creates two cloudformation stacks instead of one. This might result in all the workloads being inaccessible depending on which cloudformation stack finishes first. For the latest update on this issue, see the corresponding Knowledge article: Cloudera Customer Advisory 2024-750: CDW activation through UI can result in inaccessible workload endpoints (AWS only)

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

Known issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.9.

Carried over from the previous release: Upgrade-related

TSB 2023-719: Cloudera Data Warehouse Backup/Restore of CDP Data Visualization incomplete

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature will encounter an issue with the restoration versions of Cloudera Data Visualization (CDV) older than CDV 7.1.6.2-3 due to a schema change in this release. If the Backup was taken from an older CDW environment that contained a version of CDV older than CDV 7.1.6.2-3, the Restore procedure will succeed. Though once the user opens the CDV Queries tab, the user could encounter the error message: “column jobs_jobschedule.owner_id does not exist…”

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

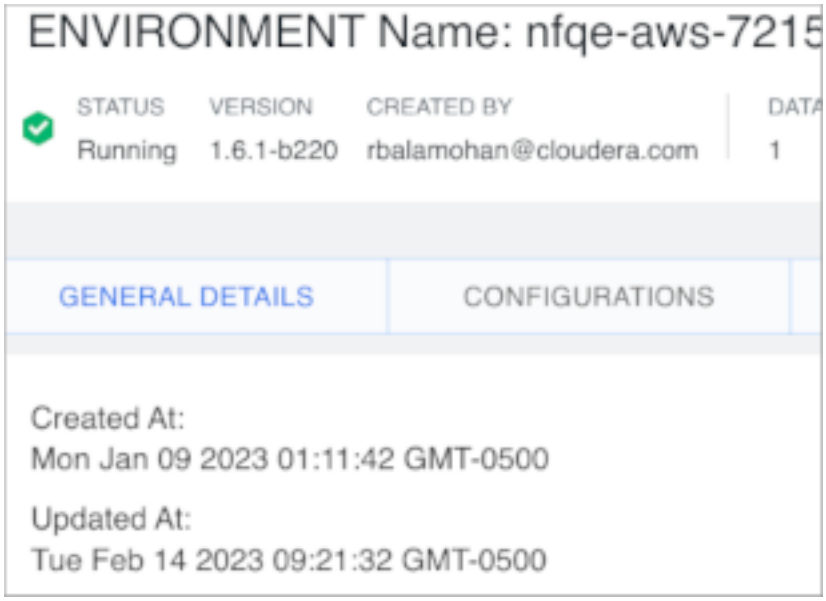

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an environment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several different popups can occur, including the following one:

worker subnets should be selected from 3 different availability zones. got: 0

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

DWX-17703: Non-HA Impala Virtual Warehouse on a private Azure Kubernetes Service (AKS) setup fails

When 'Refresh' and 'Stop' operations run in parallel, Impala might move into an error state. The Refresh operation might think that Impala is in an error state as the coordinator pod is missing.

Rebuild the Impala Virtual Warehouse or restart it using the CLI.

DWX-17109: ABFS File Browser operations failing intermittently

You may encounter intermittent issues while performing typical operations on files and directories on the ABFS File Broswer, such as moving or renaming files.

None.

DWX-17613: Generic error message is displayed when you click on the directory you don't have access to on a RAZ cluster

You see the following error message when you click on an ABFS directory to which you do not have read/write permission on the ABFS File Browser in Hue: There was a problem with your request. This message is generic and does not provide insight into the actual issue.

None.

DWX-17620: Folder having special characters in its name is not accessible in ABFS

In Cloudera Data Warehouse Public Cloud, from a Virtual Warehouse going to the Azure Blob Filesystem (ABFS), creating a folder, and then performing an action such as Move, causes an error as shown in the following example:

```
Cannot access: abfs://data-files/user/hrt_qa/~@$&()*!+'=;. 404 C
lient Error: The specified path does not exist.
```

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

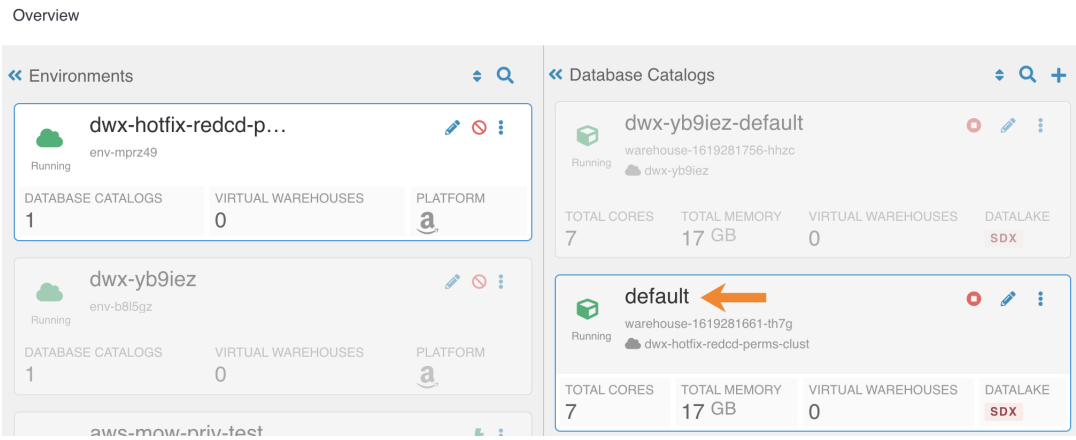

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Limited Hive image versions

Hive Virtual Warehouses you create in 1.8.1-b248 (released Nov 20, 2023) and later will run Istio 1.19.0. The new Istio version supports only new versions of Hive helm charts. If you have the CDW_VERSIONED_DEPLOY, only new Hive image versions appear in UI when you create a new Hive Virtual Warehouse.

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

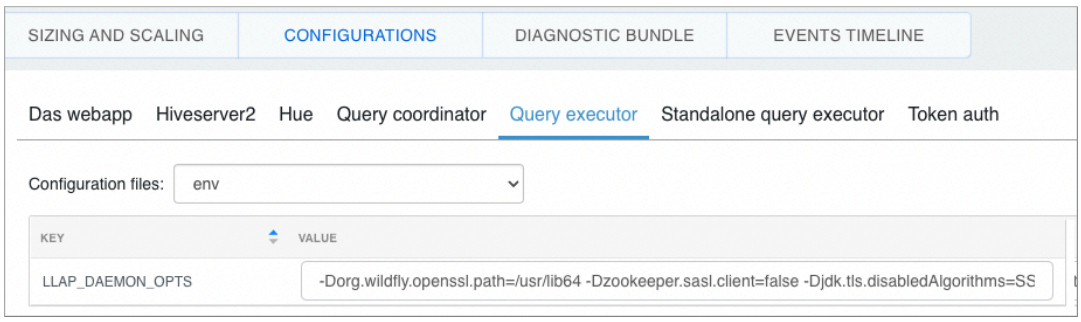

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
       - apps
       resources:
       - deployments
       - statefulsets
       verbs:
       - get
       - list
       - apiGroups:
       - "edws.cloudera.com"
       resources:
       - computes
       verbs:
```

```
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

CDPD-27918: Hue does not automatically pick up RAZ HA configurations

On a CDP Public Cloud environment in which you have configured RAZ in High Availability mode, Hue in CDW does not pick up all the RAZ host URLs automatically. Therefore, if a RAZ instance to which Hue is connected goes down, Hue becomes unavailable.

You must manually add comma-separated RAZ instances in the Hue Advanced Configuration Snippet.

1. Log in to the CDP Management Console as an Administrator.
2. Go to Environment Data Lake and open Cloudera Manager for your environment.
3. Go to Clusters Ranger RAZ service Instances RAZ server Processes and note the value of the fs.s3a.ext.raz.rest.host.url property from the core-site.xml file. You need this to specify the value of the api_url property in the Hue configuration.

   For Azure environments, note the value of the fs.azure.ext.raz.rest.host.url property.

   For AWS and GCS environments, note the value of the fs.s3a.ext.raz.rest.host.url property.
4. Go to CDW Virtual Warehouse ⋮ Edit CONFIGURATIONS and select the hue-safety-valve from the Configuration files dropdown menu.
5. Add the following lines in the hue-safety-valve field:

   ```
   [desktop]
   [[raz]]
   is_enabled=true
   api_url=https://[***INSTANCE-1***]:6082/,https://[***INSTANCE-
   2***]:6082/
   ```
6. Click Apply Changes.

DWX-16899: Error while viewing Impala job status on the mini Job Browser

When you click on the application ID after submitting an Impala query on Hue that is running on the environment level, you may notice the following error: 401 Client Error: Unauthorized for url: http://coordinator.impala-xyz.svc.cluster.local:25000/queries?json=true Must authenticate with Basic authentication. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16913: “Results have expired” message while running a CTAS query

You may see the following message on the Hue UI when you submit a CREATE TABLE AS SELECT (CTAS) query from a Hue instance that is deployed on the environment level: “Results have expired, rerun the query if needed”. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16917: Failed to validate proxy privileges error while running queries from Hue

You may see the following error intermittently in Hue's pod logs while running queries from a Hue instance that is deployed at the environment level: "Failed to validate proxy privileges of <username>".

None.

DWX-16895: Incorrect status of Hue pods when you edit the Hue instance properties

When you update a configuration of a Hue instance that is deployed at the environment level, such as increasing or decreasing the size of the Hue instance, you see a success message on the CDW UI. After some time, the status of the Hue instance also changes from "Updating" to "Running". However, when you list the Hue pods using kubectl, you see that not all backend pods are in the running state–a few of them are still in the init state.

None. The pods come up successfully eventually after a sufficient time has passed.

DWX-16863: The upgrade button is present on the CDW UI, but Hue upgrades are not supported

You see the Upgrade button on the **Query Editor** page in the CDW UI when Hue is deployed at the environment level. However, on CDW version 1.8.1, upgrading the Hue instance that is deployed at the environment level is not supported.

None.

DWX-16893: A user can see all the queries in Job browser

In a Hue instance deployed at the environment level, by design, the Hue instances must not share the saved queries and query history with other Hue instances even for the same user. However, a logged in user is able to view all the queries executed by that user on all the Virtual Warehouses on a particular Database Catalog.

None.

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

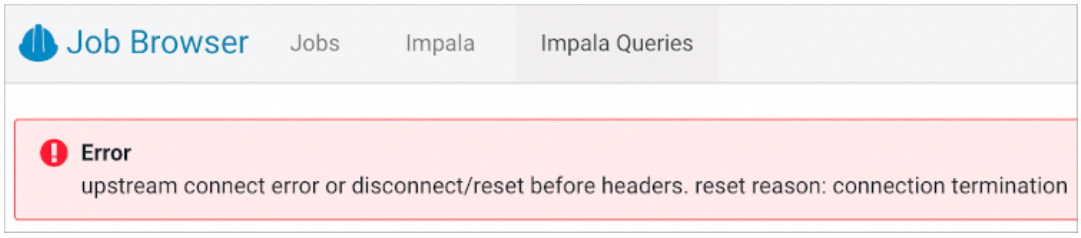

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the "403 - CSRF" error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Carried over from the previous release: Iceberg

IMPALA-12742 DELETE/UPDATE Iceberg table partitioned by DATE fails

You can partition Iceberg tables using identities, such as int and date. The file path contains the partition value you can read. When you partition by DATE, subsequently running queries to update or delete data from partitioned data creates an incorrect conversion of DATE. The Catalog cannot parse the data and throws an error.

CDPD-66779: Partitioned Iceberg table not getting loaded with insert select query from Hive

If you create a partitioned table in Iceberg and then try to insert data from another table as shown below, an error occurs.

```
insert into table partition_transform_4 select t, ts from vector
tab10k;
```

Use the CLUSTER BY clause on the partitioned column to insert data. For example:

```
insert into table partition_transform_4 select t, ts from t1 clu
ster by ts;
```

Branch FAST FORWARD does not work as expected

The Apache Iceberg spec indicates you can use either one or two arguments to fast forward a branch. The following example shows using two arguments:

```
ALTER TABLE <name> EXECUTE FAST-FORWARD 'x' 'y'
```

However, omitting the second branch name, does not work as documented by Apache Iceberg. The named branch is not fast-forwarded to the current branch. An exception occurs at the Iceberg level.

You must use two arguments to the EXECUTE FAST FORWARD feature to forward a branch.

HIVE-28055 Merging Iceberg branches requires a target table alias

Hive supports only one level of qualifier when referencing columns. In other words only one dot is accepted. For example, select table.col from ...; is allowed. select db.table.col is not allowed. Using the merge statement to merge Iceberg branches without a target or source table alias causes an exception:

```
org.apache.hadoop.hive.ql.parse.SemanticException: ... Invalid t
able alias or column reference ...
```

Use an alias, for example t, for the target table.

```
merge into mydb.target.branch_branch1 t using mydb.source.branch
_branch1 s on t.id = s.id when matched then update set value = '
matched';
```

DWX-17210: Timeout issue querying Iceberg tables from Hive

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

- fs.s3.maxConnections=1000
- fs.s3a.connection.maximum=1000

Restart the Database Catalog and Virtual Warehouse.

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

DWX-17455 ODBC client using JWT authentication cannot connect to Impala Virtual Warehouse

If you are using the Cloudera ODBC Connector and JWT authentication to connect to a CDW Impala Virtual Warehouse where the Impala coordinators are configured for high availability in an active-active mode, the connection results in a "401 Unauthorized" error. The error is not seen on Impala Virtual Warehouses that have an active-passive high availability or where high availability is disabled.

In the ODBC Data Source Name (DSN), make the following changes:

1. Ensure that SSL is enabled by setting SSL=1.
2. Remove authentication by setting AuthMech=0.
3. Remove the JWTString parameter.
4. Add the parameter http.header.Authorization and set it to the word "Bearer" followed by a space and the JWT string

DSN configuration before the workaround

```
SSL=1
AuthMech=8
JWTString=full_jwt_text
```

DSN configuration after the workaround

```
SSL=1
AuthMech=0
http.header.Authorization=Bearer full_jwt_text
```

IMPALA-12809 Metadata query problem from Impala

When querying metadata from Impala on a cluster with more than a few nodes, the query might fail with a null pointer exception. Reported problems occurred when running join queries on metadata, but other metadata queries might fail.

Limit the number of nodes that process the query by setting the NUM_NODES query option to 1:

```
SET NUM_NODES=1
```

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-17175: Impala Virtual Warehouse max executor limit issue

The Impala Virtual Warehouse has a limit of max 200 executors. The Executors slider has a max limit of 200 for creating or editing a Virtual Warehouse. If you select a bigger t-shirt size than the default, or set a custom t-shirt size, the default max limit set by the UI could be more than 200 in some cases. Exceeding the max limit causes an error during a future Edit/Upgrade/Rebuild operation.

Workaround: Manually set the max executors limit to a preferred value less than 200, but a multiple of t-shirt size, when creating or editing Impala Virtual Warehouse

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

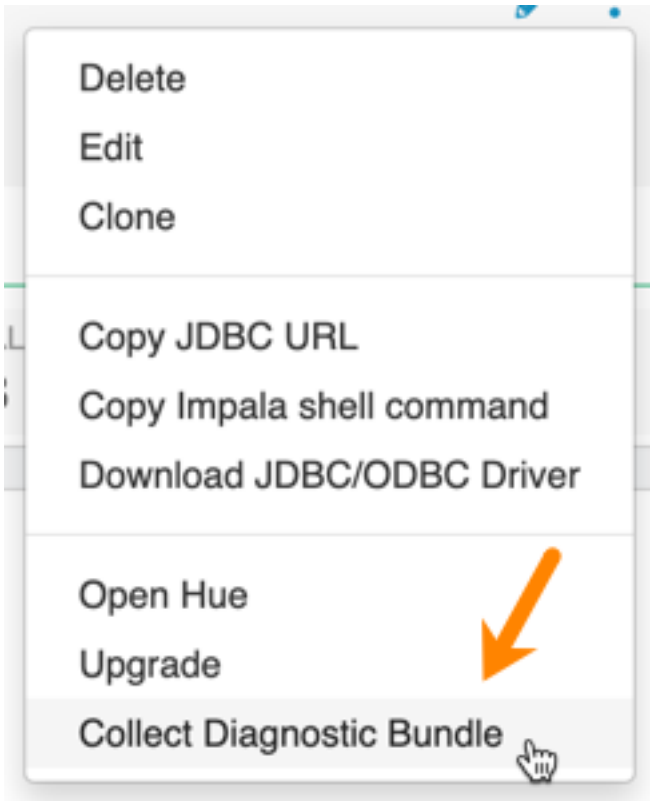

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

When working with Apache Hive (Hive) and Apache Iceberg (Iceberg) V2 tables, using the TRUNCATE statement may lead to unintended data deletion. This issue arises when the truncate command is applied to a branch of an Iceberg table. Instead of truncating the branch itself, the command affects the original (main) table, which results in unintended loss of data.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

March 4, 2024 - Hotfix

Review the known issues, fixed issues, and changed behaviors in this hotfix release of Cloudera Data Warehouse on Public Cloud.

Fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

Technical Service Bulletins

TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW.

Known issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.9.

Carried over from the previous release: Upgrade-related

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

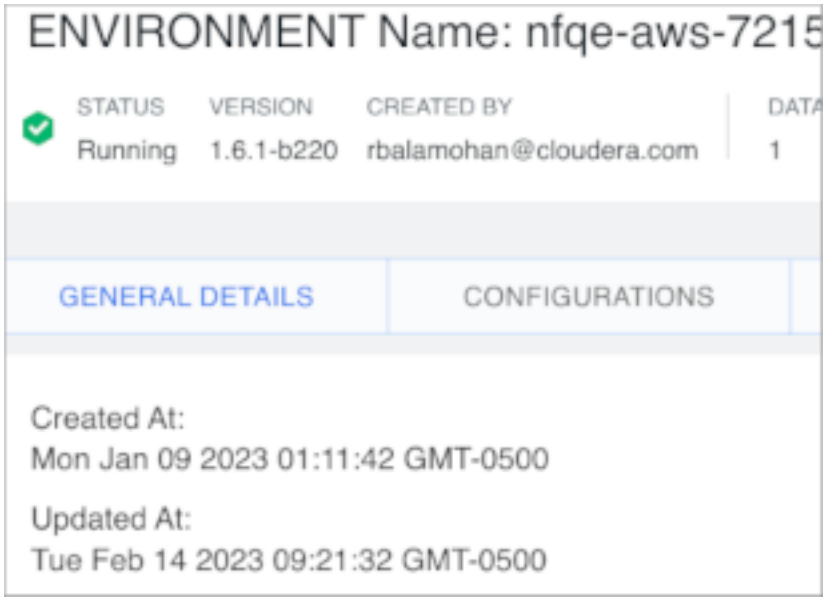

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an environment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several different popups can occur, including the following one:

```
worker subnets should be selected from 3 different
availability zones. got: 0
```

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the Cre ateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersio ns line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
```

```
            "ec2:DescribeLaunchTemplateVersions",
            "ec2:CreateTags"
            ],
```

7. Save changes.

Carried over from the previous release: Azure

DWX-17703: Non-HA Impala Virtual Warehouse on a private Azure Kubernetes Service (AKS) setup fails

When 'Refresh' and 'Stop' operations run in parallel, Impala might move into an error state. The Refresh operation might think that Impala is in an error state as the coordinator pod is missing.

Rebuild the Impala Virtual Warehouse or restart it using the CLI.

DWX-17109: ABFS File Browser operations failing intermittently

You may encounter intermittent issues while performing typical operations on files and directories on the ABFS File Broswer, such as moving or renaming files.

None.

DWX-17613: Generic error message is displayed when you click on the directory you don't have access to on a RAZ cluster

You see the following error message when you click on an ABFS directory to which you do not have read/write permission on the ABFS File Browser in Hue: There was a problem with your request. This message is generic and does not provide insight into the actual issue.

None.

DWX-17620: Folder having special characters in its name is not accessible in ABFS

In Cloudera Data Warehouse Public Cloud, from a Virtual Warehouse going to the Azure Blob Filesystem (ABFS), creating a folder, and then performing an action such as Move, causes an error as shown in the following example:

```
Cannot access: abfs://data-files/user/hrt_qa/~@$&()*!+'=;. 404 C
lient Error: The specified path does not exist.
```

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click Edit Diagnostic Bundle. Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

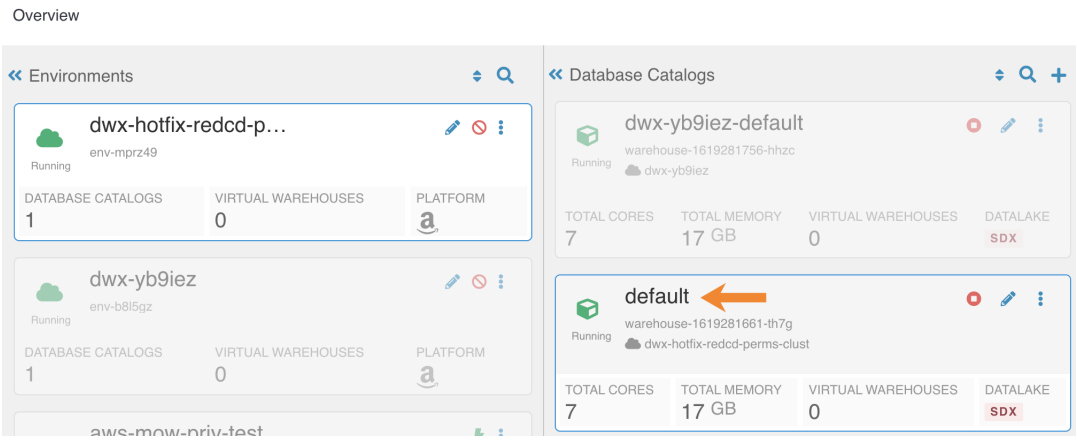

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Limited Hive image versions

Hive Virtual Warehouses you create in 1.8.1-b248 (released Nov 20, 2023) and later will run Istio 1.19.0. The new Istio version supports only new versions of Hive helm charts. If you have the CDW_VERSIONED_DEPLOY, only new Hive image versions appear in UI when you create a new Hive Virtual Warehouse.

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

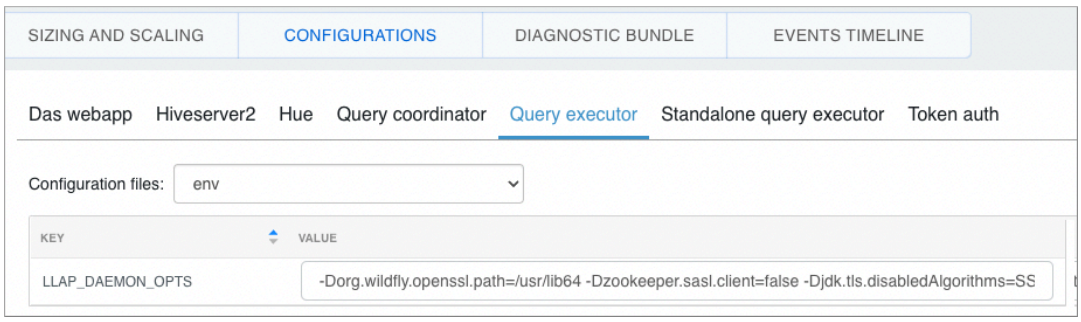

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
       - apps
       resources:
       - deployments
       - statefulsets
       verbs:
       - get
       - list
       - apiGroups:
       - "edws.cloudera.com"
       resources:
       - computes
       verbs:
```

```
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

CDPD-27918: Hue does not automatically pick up RAZ HA configurations

On a CDP Public Cloud environment in which you have configured RAZ in High Availability mode, Hue in CDW does not pick up all the RAZ host URLs automatically. Therefore, if a RAZ instance to which Hue is connected goes down, Hue becomes unavailable.

You must manually add comma-separated RAZ instances in the Hue Advanced Configuration Snippet.

1. Log in to the CDP Management Console as an Administrator.
2. Go to Environment Data Lake and open Cloudera Manager for your environment.
3. Go to Clusters Ranger RAZ service Instances RAZ server Processes and note the value of the fs.s3a.ext.raz.rest.host.url property from the core-site.xml file. You need this to specify the value of the api_url property in the Hue configuration.

   For Azure environments, note the value of the fs.azure.ext.raz.rest.host.url property.

   For AWS and GCS environments, note the value of the fs.s3a.ext.raz.rest.host.url property.
4. Go to CDW Virtual Warehouse ⋮ Edit CONFIGURATIONS and select the hue-safety-valve from the Configuration files dropdown menu.
5. Add the following lines in the hue-safety-valve field:

   ```
   [desktop]
   [[raz]]
   is_enabled=true
   api_url=https://[***INSTANCE-1***]:6082/,https://[***INSTANCE-
   2***]:6082/
   ```
6. Click Apply Changes.

DWX-16899: Error while viewing Impala job status on the mini Job Browser

When you click on the application ID after submitting an Impala query on Hue that is running on the environment level, you may notice the following error: 401 Client Error: Unauthorized for url: http://coordinator.impala-xyz.svc.cluster.local:25000/queries?json=true Must authenticate with Basic authentication. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16913: “Results have expired” message while running a CTAS query

You may see the following message on the Hue UI when you submit a CREATE TABLE AS SELECT (CTAS) query from a Hue instance that is deployed on the environment level: “Results have expired, rerun the query if needed”. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16917: Failed to validate proxy privileges error while running queries from Hue

You may see the following error intermittently in Hue's pod logs while running queries from a Hue instance that is deployed at the environment level: "Failed to validate proxy privileges of <username>".

None.

DWX-16895: Incorrect status of Hue pods when you edit the Hue instance properties

When you update a configuration of a Hue instance that is deployed at the environment level, such as increasing or decreasing the size of the Hue instance, you see a success message on the CDW UI. After some time, the status of the Hue instance also changes from "Updating" to "Running". However, when you list the Hue pods using kubectl, you see that not all backend pods are in the running state–a few of them are still in the init state.

None. The pods come up successfully eventually after a sufficient time has passed.

DWX-16863: The upgrade button is present on the CDW UI, but Hue upgrades are not supported

You see the Upgrade button on the **Query Editor** page in the CDW UI when Hue is deployed at the environment level. However, on CDW version 1.8.1, upgrading the Hue instance that is deployed at the environment level is not supported.

None.

DWX-16893: A user can see all the queries in Job browser

In a Hue instance deployed at the environment level, by design, the Hue instances must not share the saved queries and query history with other Hue instances even for the same user. However, a logged in user is able to view all the queries executed by that user on all the Virtual Warehouses on a particular Database Catalog.

None.

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

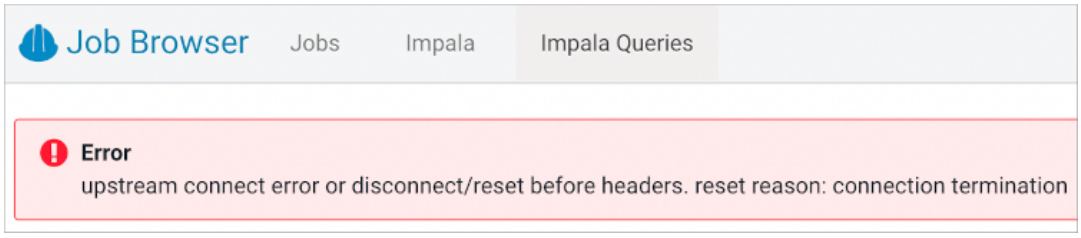

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the "403 - CSRF" error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Carried over from the previous release: Iceberg

IMPALA-12742 DELETE/UPDATE Iceberg table partitioned by DATE fails

You can partition Iceberg tables using identities, such as int and date. The file path contains the partition value you can read. When you partition by DATE, subsequently running queries to update or delete data from partitioned data creates an incorrect conversion of DATE. The Catalog cannot parse the data and throws an error.

CDPD-66779: Partitioned Iceberg table not getting loaded with insert select query from Hive

If you create a partitioned table in Iceberg and then try to insert data from another table as shown below, an error occurs.

```
insert into table partition_transform_4 select t, ts from vector
tab10k;
```

Use the CLUSTER BY clause on the partitioned column to insert data. For example:

```
insert into table partition_transform_4 select t, ts from t1 clu
ster by ts;
```

Branch FAST FORWARD does not work as expected

The Apache Iceberg spec indicates you can use either one or two arguments to fast forward a branch. The following example shows using two arguments:

```
ALTER TABLE <name> EXECUTE FAST-FORWARD 'x' 'y'
```

However, omitting the second branch name, does not work as documented by Apache Iceberg. The named branch is not fast-forwarded to the current branch. An exception occurs at the Iceberg level.

You must use two arguments to the EXECUTE FAST FORWARD feature to forward a branch.

HIVE-28055 Merging Iceberg branches requires a target table alias

Hive supports only one level of qualifier when referencing columns. In other words only one dot is accepted. For example, select table.col from ...; is allowed. select db.table.col is not allowed. Using the merge statement to merge Iceberg branches without a target or source table alias causes an exception:

```
org.apache.hadoop.hive.ql.parse.SemanticException: ... Invalid t
able alias or column reference ...
```

Use an alias, for example t, for the target table.

```
merge into mydb.target.branch_branch1 t using mydb.source.branch
_branch1 s on t.id = s.id when matched then update set value = '
matched';
```

DWX-17210: Timeout issue querying Iceberg tables from Hive

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

- fs.s3.maxConnections=1000
- fs.s3a.connection.maximum=1000

Restart the Database Catalog and Virtual Warehouse.

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

DWX-17455 ODBC client using JWT authentication cannot connect to Impala Virtual Warehouse

If you are using the Cloudera ODBC Connector and JWT authentication to connect to a CDW Impala Virtual Warehouse where the Impala coordinators are configured for high availability in an active-active mode, the connection results in a "401 Unauthorized" error. The error is not seen on Impala Virtual Warehouses that have an active-passive high availability or where high availability is disabled.

In the ODBC Data Source Name (DSN), make the following changes:

1. Ensure that SSL is enabled by setting SSL=1.
2. Remove authentication by setting AuthMech=0.
3. Remove the JWTString parameter.
4. Add the parameter http.header.Authorization and set it to the word "Bearer" followed by a space and the JWT string

DSN configuration before the workaround

```
SSL=1
AuthMech=8
JWTString=full_jwt_text
```

DSN configuration after the workaround

```
SSL=1
AuthMech=0
http.header.Authorization=Bearer full_jwt_text
```

IMPALA-12809 Metadata query problem from Impala

When querying metadata from Impala on a cluster with more than a few nodes, the query might fail with a null pointer exception. Reported problems occurred when running join queries on metadata, but other metadata queries might fail.

Limit the number of nodes that process the query by setting the NUM_NODES query option to 1:

```
SET NUM_NODES=1
```

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-17175: Impala Virtual Warehouse max executor limit issue

The Impala Virtual Warehouse has a limit of max 200 executors. The Executors slider has a max limit of 200 for creating or editing a Virtual Warehouse. If you select a bigger t-shirt size than the default, or set a custom t-shirt size, the default max limit set by the UI could be more than 200 in some cases. Exceeding the max limit causes an error during a future Edit/Upgrade/Rebuild operation.

Workaround: Manually set the max executors limit to a preferred value less than 200, but a multiple of t-shirt size, when creating or editing Impala Virtual Warehouse

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

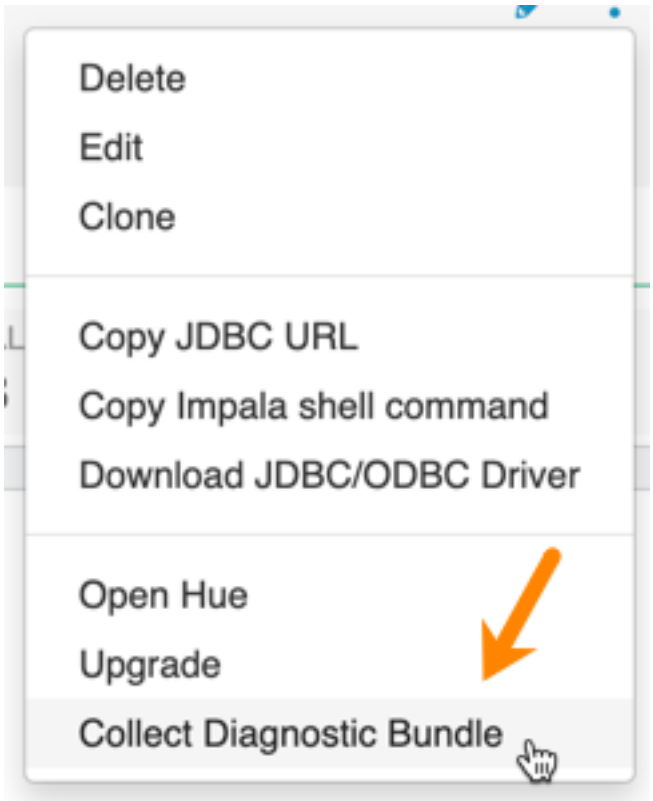

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2023-719: Cloudera Data Warehouse Backup/Restore of CDP Data Visualization incomplete

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature will encounter an issue with the restoration versions of Cloudera Data Visualization (CDV) older than CDV 7.1.6.2-3 due to a schema change in this release. If the Backup was taken from an older CDW environment that contained a version of CDV older than CDV 7.1.6.2-3, the Restore procedure will succeed. Though once the user opens the CDV Queries tab, the user could encounter the error message: “column jobs_jobschedule.owner_id does not exist…”

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete.

TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

Apache Hive (Hive) and Apache Impala (Impala) modify statements (DELETE/UPDATE/MERGE) on Apache Iceberg (Iceberg) V2 tables can corrupt the tables if there is a concurrent table compaction from Apache Spark. The issue happens when the compaction and modify statement run in parallel, and when the compaction job commits before the modify statement. In this case the position delete files of the modify statement still point to the old files. This means the following in case of

- DELETE statements
  - Deleting records pointing to old files have no effect
- UPDATE / MERGE statements
  - Deleting records pointing to old files have no effect
  - The table will also have the newly added data records
  - Rewritten records will still be active

This issue does not affect Apache NiFi (NiFi) and Apache Flink (Flink) as these components write equality delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables.

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

When working with Apache Hive (Hive) and Apache Iceberg (Iceberg) V2 tables, using the TRUNCATE statement may lead to unintended data deletion. This issue arises when the truncate command is applied to a branch of an Iceberg table. Instead of truncating the branch itself, the command affects the original (main) table, which results in unintended loss of data.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion.

Behavior changes

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

This release is a hot fix for a condition related to DWX server code and the Amazon Relational Database (RDS) that causes an inordinate number of events to be saved. Insufficient database connections can occur and Cloudera Data Warehouse can shut down.

February 29, 2024 Release Notes

Review the new feaures, fixes, behavioral changes, and preview features in this release of Cloudera Data Warehouse on Public Cloud.

What's new in Cloudera Data Warehouse on Public Cloud

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Cloudera Data Warehouse Public Cloud 1.8.5-b35 changes, described in more detail below:

- AWS Elastic Kubernetes Service 1.28 support on page 89
- Accessing S3 buckets in an AWS environment on page 89
- Database Catalog size configuration on page 89
- GA availability of a private CDW environment in Azure Kubernetes Service on page 89
- IAM policies moved from documentation to a public Github repository on page 89
- Metrics for monitoring and troubleshooting Impala Virtual Warehouses on page 89
- CDP CLI commands for creating and updating an AWS and Azure cluster on page 89
- CDP CLI commands for configuring Workload Aware Auto-Scaling on page 90

Cloudera Data Warehouse Runtime 2024.0.17.0-73 changed, described in more detail below:

Iceberg

- Iceberg drop partition feature on page 90
- Iceberg branching and tagging GA on page 90
- Enhanced security of Iceberg metadata on page 90
- Impala support for changing the Iceberg table metadata location on page 91
- Support for copy-on-write (COW) on page 91
- Query metadata tables feature on page 91
- Directed distribution mode on page 91

Impala

- SHOW VIEWS statement on page 91
- Planner changes to improve cardinality estimation on page 91
- Distribute runtime filter aggregation on page 91

- Improvement in catalog observability on page 91
- Caching codegen functions on page 92

AWS Elastic Kubernetes Service 1.28 support

New CDW clusters using AWS environments you activate in this release 1.8.4-b35 (released February, 2024) of CDW will use Amazon Kubernetes (EKS) version 1.28. For more information, see Upgrading Amazon Kubernetes Service.

Accessing S3 buckets in an AWS environment

You can use the CDW UI for configuring access to S3 buckets under certain conditions. You can also configure your own custom encryption key for read/write access from CDW Public Cloud on AWS to the external S3 bucket.

For more information, see Accessing S3 buckets.

Database Catalog size configuration

Using the following CDP CLI commands, you can configure the Java heap for your Database Catalog to small 8Gb (default), medium 16Gb, or large 24 Gb:

Create a default Database Catalog that configures, for example, a large Java Heap size.

```
dw create-dbc --cluster-id=env-sb42vs --name=SSlarge --memory-tshirt-size=la
rge
```

Create a Database Catalog that accepts the default Java Heap size, which is small, by not configuring the size.

```
dw create-dbc --cluster-id=env-sb42vs --name=SSdefault
```

List Database Catalog configuration information:

```
dw list-dbcs --cluster-id=env-sb42vs
```

Using the CDW UI as described in Creating a Database Catalog, you can configure the Database Catalog size.

To avoid unnecessary cloud expenses, do not increase the size unless you experience Java heap issues.

GA availability of a private CDW environment in Azure Kubernetes Service

You can now enable a private CDW environment in Azure Kubernetes Service (AKS). You use the Private CDW option in the CLI, and CDW deploys an Azure Kubernetes Cluster with only private endpoints enabled. The cluster can be accessed only from your Azure network.

IAM policies moved from documentation to a public Github repository

To meet customer requests for tracking changes to IAM policies for CDW, the policies now reside in Github. CDW documentation, such as "Attaching the policy to your cross-account role" provides links to the policies.

Metrics for monitoring and troubleshooting Impala Virtual Warehouses

Additional global metrics and table level event metrics are available for debugging an Impala Virtual Warehouse.

CDP CLI commands for creating and updating an AWS and Azure cluster

You can create and update a CDW cluster in Amazon and Azure environments using the following commands:

- create-aws-cluster --environment-crn <value> [options]
- create-azure-cluster --environment-crn <value> --user-assigned-managed-identity <value> [options]
- update-aws-cluster [options]

- update-azure-cluster [options]

For more information about using the commands, including options, see the CDP CLI reference.

CDP CLI commands for configuring Workload Aware Auto-Scaling

You can perform the following Workload Aware Auto-Scaling (WAAS) configurations as part of a Virtual Warehouse create request:

- Create a Virtual Warehouse with executor group sets
- Choose how many executor group sets to configure
- Configure each executor group set.

  Modify, add, or delete an executor group set as part of a Virtual Warehouse update request. To delete a group set, set the deleteGroupSet option to true for the group set.
- Update executor group sets of a Virtual Warehouse

For example:

```
cdp --profile ${CDP_PROFILE} \
  dw create-vw \
   --cluster-id ${CLUSTER_ID} \
   --dbc-id ${DBC_ID} \
   --vw-type impala \
   --name "impala-$(openssl rand -hex 6)" \
   --template xsmall \
   --impala-ha-settings highAvailabilityMode=ACTIVE_ACTIVE \
   --autoscaling "impalaExecutorGroupSets={small={execGroupSize=1,minExecu
torGroups=1,maxExecutorGroups=1,autoSuspendTimeoutSeconds=301,disableAutoSus
pend=true,triggerScaleUpDelay=21,triggerScaleDownDelay=21},custom1={execGrou
pSize=2,minExecutorGroups=0,maxExecutorGroups=1,autoSuspendTimeoutSeconds=30
2,disableAutoSuspend=true,triggerScaleUpDelay=22,triggerScaleDownDelay=22},c
ustom2={execGroupSize=3,minExecutorGroups=0,maxExecutorGroups=1,autoSuspendT
imeoutSeconds=303,disableAutoSuspend=true,triggerScaleUpDelay=23,triggerScal
eDownDelay=23},custom3={execGroupSize=4,minExecutorGroups=0,maxExecutorGroup
s=1,autoSuspendTimeoutSeconds=304,disableAutoSuspend=true,triggerScaleUpDela
y=24,triggerScaleDownDelay=24},large={execGroupSize=5,minExecutorGroups=0,ma
xExecutorGroups=1,autoSuspendTimeoutSeconds=305,disableAutoSuspend=true,trig
gerScaleUpDelay=25,triggerScaleDownDelay=25}}"
```

Iceberg drop partition feature

You can easily remove a partition from an Iceberg partition using an alter table statement from Impala. Removing a partition does not affect the table schema. The column is not removed from the schema. This feature is offered on a general availability (GA) basis.

Iceberg branching and tagging GA

In this release, branching and tagging is offered on a general availability (GA) basis. From Hive, you can manage the lifecycle of snapshots using the Iceberg branching and Iceberg tagging features. Branches are references to snapshots that have a lifecycle of their own. Tags identify snapshots you need for auditing and conforming to GDPR. Cloudera recommends that you use this feature in test and development environments. It is not recommended for production deployments.

Enhanced security of Iceberg metadata

In this release, you can prevent overrides of the Iceberg file metadata location by unauthorized users. You accept the default (true), in the Virtual Warehouse to use this feature:

- Hive

  Configure the property hive.iceberg.allow.datafiles.in.table.location.only.

- Impala

  Configure the catalogd property iceberg_restrict_data_file_location.

When set to true, all the data files being read must be within the table location; otherwise, an error occurs. When set to false, an unauthorized party who knows the underlying schema and file location outside the table location can rewrite the manifest files within one table location to point to the data files in another table location to read your data. For more information, see "Changing the table metadata location".

Impala support for changing the Iceberg table metadata location

In this release, you can change the Iceberg table metadata location from Impala as well as Hive.

Support for copy-on-write (COW)

Hive supports the copy-on-write (COW) as well as merge-on-read (MOR) for handling Iceberg row-level updates and deletes. You configure COW or MOR based on your use case and rate of data change.

Query metadata tables feature

From Hive and Impala, you can query Iceberg metadata tables as you would query a Hive table. For example, you can use projections, joins, filters, and so on.

Directed distribution mode

This release implements directed distribution mode. The scheduler collects information about which Iceberg data file is scheduled on which host. Since, the scan node for the data files are on the same host as the Iceberg join node, delete files are sent directly to that specific host. This mode can improve V2 table read performance.

SHOW VIEWS statement

This release introduces the SHOW VIEWS statement, which simplifies the task of listing all views within a specified schema or database. Using this command, you can quickly identify and review views, thereby enhancing performance by reducing metadata scan operations.

Planner changes to improve cardinality estimation

Significant changes have been made to the query planner to improve cardinality estimation, a critical component of workload-aware autoscaling.

In previous versions, Impala would generate a plan first and then search for runtime filters based on the entire plan. In this release, selective runtime filters have been integrated. These filters aim to reduce the cardinality estimates of scan nodes and specific join nodes located above them. This refinement occurs after the generation of runtime filters and before the computation of resource requirements.

Distribute runtime filter aggregation

Aggregating runtime filters during runtime can impose significant memory overhead on the coordinator. To address this issue, we initially introduced local aggregation of runtime filters within a single executor node, aiming to alleviate strain on the coordinator by transmitting filter updates only after local aggregation. However, as Impala clusters scale up, the limitations of local filter aggregation become evident, especially in scenarios involving numerous nodes. This places significant memory stress on the coordinator node. To mitigate this challenge, we have implemented a solution that distributes the runtime filter aggregation across specific Impala backends.

Improvement in catalog observability

This release introduces significant enhancements to the Impala Catalog Web UI, focusing on addressing performance issues associated with delays in processing Hive Metastore (HMS) events. These improvements aim to mitigate the risk of queries using outdated metadata.

Caching codegen functions

In Impala, "codegen" involves generating specialized machine code for each query based on query-specific information. When executing a standard query, the query optimizer generates an optimized query plan, which is then passed to the executor for processing. The codegen capability converts query-specific information into machine code, enhancing query performance through faster execution.

Support ORDER BY for collections of variable length types in SELECT list

This release introduces support for collections of variable length types in the sorting tuple. While it's now possible to include these collection columns in the SELECT list alongside other columns used for sorting, direct sorting by these collection columns is not supported. Additionally, collections of variable-length types can now serve as non-passthrough children of UNION ALL nodes.

It is important to note that structs containing collections, whether of variable or fixed length, are still not supported in the select list for ORDER BY queries.

Here are examples of supported queries:

```
select id, arr_string_1d from collection_tbl order by id;
select id, map_1d from collection_tbl order by id;
```

However, queries such as the following are not supported:

```
select id, struct_contains_map from collection_struct_mix order by id;
```

Attempting to execute such queries will result in the error message "AnalysisException: Sorting is not supported if the select list contains collection(s) nested in struct(s)."

Preview features in Cloudera Data Warehouse on Public Cloud

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces this technical preview.

Note: Technical previews are considered under development. Do not use these features in production environments.

Impala Virtual Warehouse EBS volume type provisioning (Preview)

Impala Virtual Warehouse when configured to spill to EBS volumes can now also spill to the Amazon gp3 Elastic Block Store volumes in addition to gp2 and st1 volumes. CDP automatically provisions the gp3 volume type when conditions exist to reduce cost. The gp3 volume has slightly reduced baseline cost compared to gp2 and other advantages. Provisioning might occur when you specify scratch space limits for spilling Impala queries. Also, when you create an Impala Virtual Warehouse on new CDW environment, CDP uses a gp3 volume. For more information, see Amazon documentation.

Impala and Hive support for Iceberg equality deletes (Preview)

Cloudera supports row-level deletes, and in this release you can read equality deletes from Impala as well as Hive. For more information, see the Delete data feature.

Known issues in Cloudera Data Warehouse on Public Cloud

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.9.

New Known Issues in this release

DWX-17455 ODBC client using JWT authentication cannot connect to Impala Virtual Warehouse

If you are using the Cloudera ODBC Connector and JWT authentication to connect to a CDW Impala Virtual Warehouse where the Impala coordinators are configured for high availability in an

active-active mode, the connection results in a "401 Unauthorized" error. The error is not seen on Impala Virtual Warehouses that have an active-passive high availability or where high availability is disabled.

In the ODBC Data Source Name (DSN), make the following changes:

1. Ensure that SSL is enabled by setting SSL=1.
2. Remove authentication by setting AuthMech=0.
3. Remove the JWTString parameter.
4. Add the parameter http.header.Authorization and set it to the word "Bearer" followed by a space and the JWT string

DSN configuration before the workaround

```
SSL=1
AuthMech=8
JWTString=full_jwt_text
```

DSN configuration after the workaround

```
SSL=1
AuthMech=0
http.header.Authorization=Bearer full_jwt_text
```

IMPALA-12809 Metadata query problem from Impala

When querying metadata from Impala on a cluster with more than a few nodes, the query might fail with a null pointer exception. Reported problems occurred when running join queries on metadata, but other metadata queries might fail.

Limit the number of nodes that process the query by setting the NUM_NODES query option to 1:

```
SET NUM_NODES=1
```

HIVE-28055 Merging Iceberg branches requires a target table alias

Hive supports only one level of qualifier when referencing columns. In other words only one dot is accepted. For example, select table.col from ...; is allowed. select db.table.col is not allowed. Using the merge statement to merge Iceberg branches without a target or source table alias causes an exception:

```
org.apache.hadoop.hive.ql.parse.SemanticException: ... Invalid t
able alias or column reference ...
```

Use an alias, for example t, for the target table.

```
merge into mydb.target.branch_branch1 t using mydb.source.branch
_branch1 s on t.id = s.id when matched then update set value = '
matched';
```

DWX-17620: Folder having special characters in its name is not accessible in ABFS

In Cloudera Data Warehouse Public Cloud, from a Virtual Warehouse going to the Azure Blob Filesystem (ABFS), creating a folder, and then performing an action such as Move, causes an error as shown in the following example:

```
Cannot access: abfs://data-files/user/hrt_qa/~@$&()*!+'=;. 404 C
lient Error: The specified path does not exist.
```

Branch FAST FORWARD does not work as expected

The Apache Iceberg spec indicates you can use either one or two arguments to fast forward a branch. The following example shows using two arguments:

```
ALTER TABLE <name> EXECUTE FAST-FORWARD 'x' 'y'
```

However, omitting the second branch name, does not work as documented by Apache Iceberg. The named branch is not fast-forwarded to the current branch. An exception occurs at the Iceberg level.

You must use two arguments to the EXECUTE FAST FORWARD feature to forward a branch.

DWX-17613: Generic error message is displayed when you click on the directory you don't have access to on a RAZ cluster

You see the following error message when you click on an ABFS directory to which you do not have read/write permission on the ABFS File Browser in Hue: There was a problem with your request. This message is generic and does not provide insight into the actual issue.

None.

DWX-17109: ABFS File Browser operations failing intermittently

You may encounter intermittent issues while performing typical operations on files and directories on the ABFS File Broswer, such as moving or renaming files.

None.

CDPD-27918: Hue does not automatically pick up RAZ HA configurations

On a CDP Public Cloud environment in which you have configured RAZ in High Availability mode, Hue in CDW does not pick up all the RAZ host URLs automatically. Therefore, if a RAZ instance to which Hue is connected goes down, Hue becomes unavailable.

You must manually add comma-separated RAZ instances in the Hue Advanced Configuration Snippet.

1. Log in to the CDP Management Console as an Administrator.
2. Go to Environment Data Lake and open Cloudera Manager for your environment.
3. Go to Clusters Ranger RAZ service Instances RAZ server Processes and note the value of the fs.s3a.ext.raz.rest.host.url property from the core-site.xml file. You need this to specify the value of the api_url property in the Hue configuration.

   For Azure environments, note the value of the fs.azure.ext.raz.rest.host.url property.

   For AWS and GCS environments, note the value of the fs.s3a.ext.raz.rest.host.url property.
4. Go to CDW Virtual Warehouse ⋮ Edit CONFIGURATIONS and select the hue-safety-valve from the Configuration files dropdown menu.
5. Add the following lines in the hue-safety-valve field:

   ```
   [desktop]
   [[raz]]
   is_enabled=true
   api_url=https://[***INSTANCE-1***]:6082/,https://[***INSTANCE-
   2***]:6082/
   ```
6. Click Apply Changes.

CDPD-66779: Partitioned Iceberg table not getting loaded with insert select query from Hive

If you create a partitioned table in Iceberg and then try to insert data from another table as shown below, an error occurs.

```
insert into table partition_transform_4 select t, ts from vector
tab10k;
```

Use the CLUSTER BY clause on the partitioned column to insert data. For example:

```
insert into table partition_transform_4 select t, ts from t1 clu
ster by ts;
```

DWX-17703: Non-HA Impala Virtual Warehouse on a private Azure Kubernetes Service (AKS) setup fails

When 'Refresh' and 'Stop' operations run in parallel, Impala might move into an error state. The Refresh operation might think that Impala is in an error state as the coordinator pod is missing.

Rebuild the Impala Virtual Warehouse or restart it using the CLI.

IMPALA-12742 DELETE/UPDATE Iceberg table partitioned by DATE fails

You can partition Iceberg tables using identities, such as int and date. The file path contains the partition value you can read. When you partition by DATE, subsequently running queries to update or delete data from partitioned data creates an incorrect conversion of DATE. The Catalog cannot parse the data and throws an error.

Carried over from the previous release: Upgrade-related

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

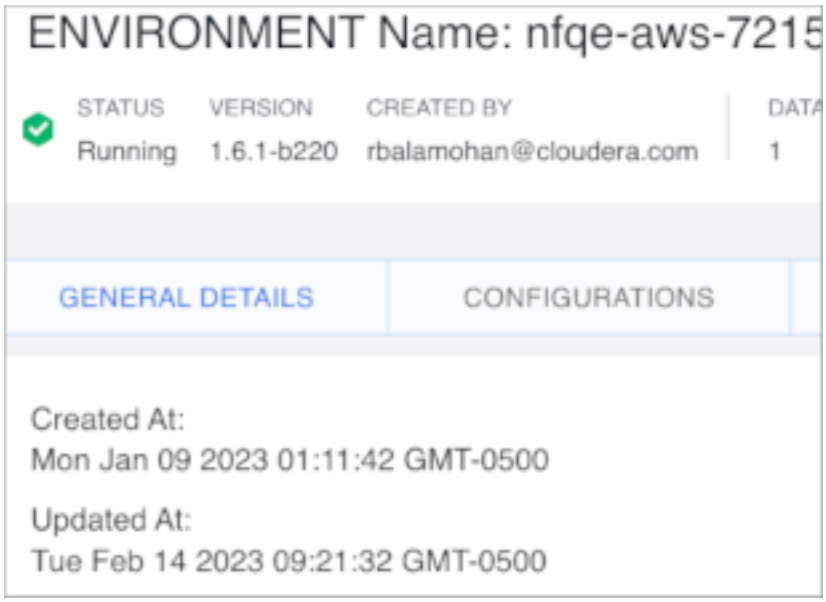

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an environment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several different popups can occur, including the following one:

```
worker subnets should be selected from 3 different
availability zones. got: 0
```

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
```

```
        operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle. Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

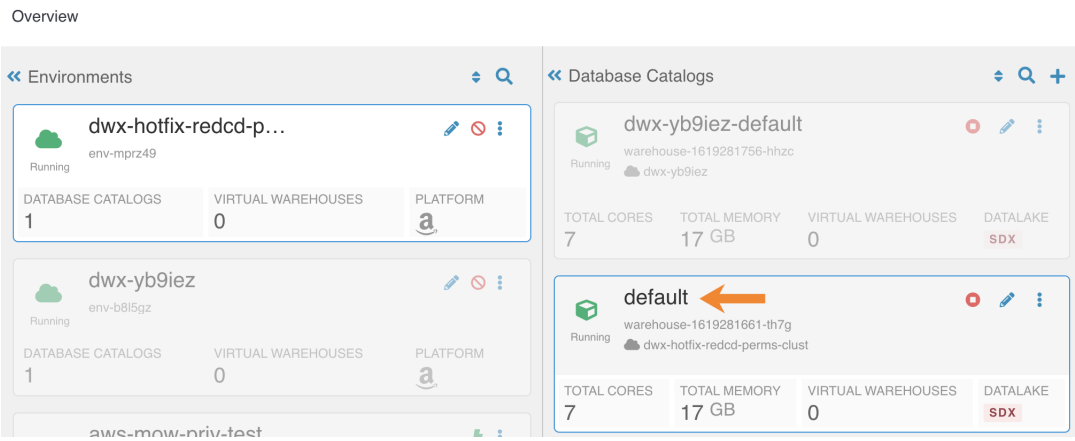

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Limited Hive image versions

Hive Virtual Warehouses you create in 1.8.1-b248 (released Nov 20, 2023) and later will run Istio 1.19.0. The new Istio version supports only new versions of Hive helm charts. If you have the CDW_VERSIONED_DEPLOY, only new Hive image versions appear in UI when you create a new Hive Virtual Warehouse.

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

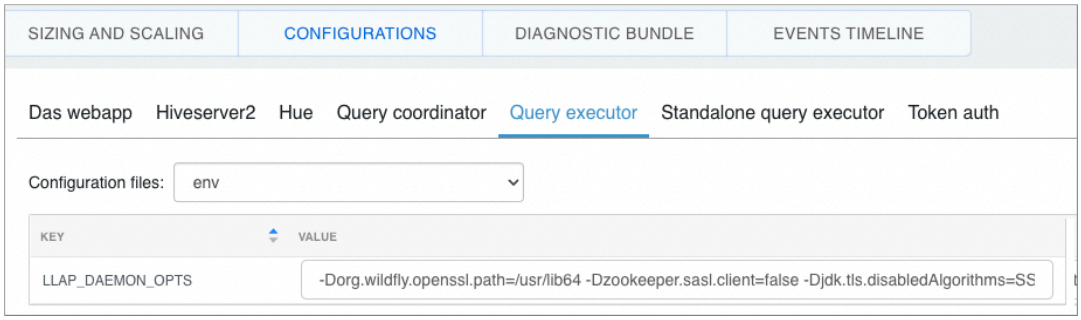

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
```

```
        - pods
        - persistentvolumeclaims
        - nodes
        verbs:
        - get
        - list
        - apiGroups:
        - apps
        resources:
        - deployments
        - statefulsets
        verbs:
        - get
        - list
        - apiGroups:
        - "edws.cloudera.com"
        resources:
        - computes
        verbs:
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

DWX-16899: Error while viewing Impala job status on the mini Job Browser

When you click on the application ID after submitting an Impala query on Hue that is running on the environment level, you may notice the following error: 401 Client Error: Unauthorized for url: http://coordinator.impala-xyz.svc.cluster.local:25000/queries?json=true Must authenticate with Basic authentication. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16913: “Results have expired” message while running a CTAS query

You may see the following message on the Hue UI when you submit a CREATE TABLE AS SELECT (CTAS) query from a Hue instance that is deployed on the environment level: “Results have expired, rerun the query if needed”. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16917: Failed to validate proxy privileges error while running queries from Hue

You may see the following error intermittently in Hue’s pod logs while running queries from a Hue instance that is deployed at the environment level: “Failed to validate proxy privileges of <username>”.

None.

DWX-16895: Incorrect status of Hue pods when you edit the Hue instance properties

When you update a configuration of a Hue instance that is deployed at the environment level, such as increasing or decreasing the size of the Hue instance, you see a success message on the CDW UI. After some time, the status of the Hue instance also changes from “Updating” to “Running”.

However, when you list the Hue pods using kubectl, you see that not all backend pods are in the running state–a few of them are still in the init state.

None. The pods come up successfully eventually after a sufficient time has passed.

DWX-16863: The upgrade button is present on the CDW UI, but Hue upgrades are not supported

You see the Upgrade button on the **Query Editor** page in the CDW UI when Hue is deployed at the environment level. However, on CDW version 1.8.1, upgrading the Hue instance that is deployed at the environment level is not supported.

None.

DWX-16893: A user can see all the queries in Job browser

In a Hue instance deployed at the environment level, by design, the Hue instances must not share the saved queries and query history with other Hue instances even for the same user. However, a logged in user is able to view all the queries executed by that user on all the Virtual Warehouses on a particular Database Catalog.

None.

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Carried over from the previous release: Iceberg

Concurrent compactions and modify statements can corrupt Iceberg tables

Hive or Impala DELETE/UPDATE/MERGE operations on Iceberg V2 tables can corrupt the tables if there is a concurrent table compaction from Spark. The issue happens if the compaction and modify statement runs in parallel, and if the compaction job commits before the modify statement. In that case the modify statement's position delete files still point to the old files. The results in the case of DELETE and in the case of UPDATE / MERGE are as follows:

- DELETE

  Delete records pointing to old files have no effect.
- UPDATE / MERGE

  Delete records pointing to old files have no effect. The table will also have the newly added data records, which means rewritten records will still be active.

Use one of the following workarounds:

- Do not run compactions and DELETE/UPDATE/MERGE statements in parallel.

- Do not compact the table via Iceberg's RewriteFiles operation. For example do not use Spark's rewriteDataFiles.

DWX-17210: Timeout issue querying Iceberg tables from Hive

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

- fs.s3.maxConnections=1000
- fs.s3a.connection.maximum=1000

Restart the Database Catalog and Virtual Warehouse.

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.

- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-17175: Impala Virtual Warehouse max executor limit issue

The Impala Virtual Warehouse has a limit of max 200 executors. The Executors slider has a max limit of 200 for creating or editing a Virtual Warehouse. If you select a bigger t-shirt size than the default, or set a custom t-shirt size, the default max limit set by the UI could be more than 200 in some cases. Exceeding the max limit causes an error during a future Edit/Upgrade/Rebuild operation.

Workaround: Manually set the max executors limit to a preferred value less than 200, but a multiple of t-shirt size, when creating or editing Impala Virtual Warehouse

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

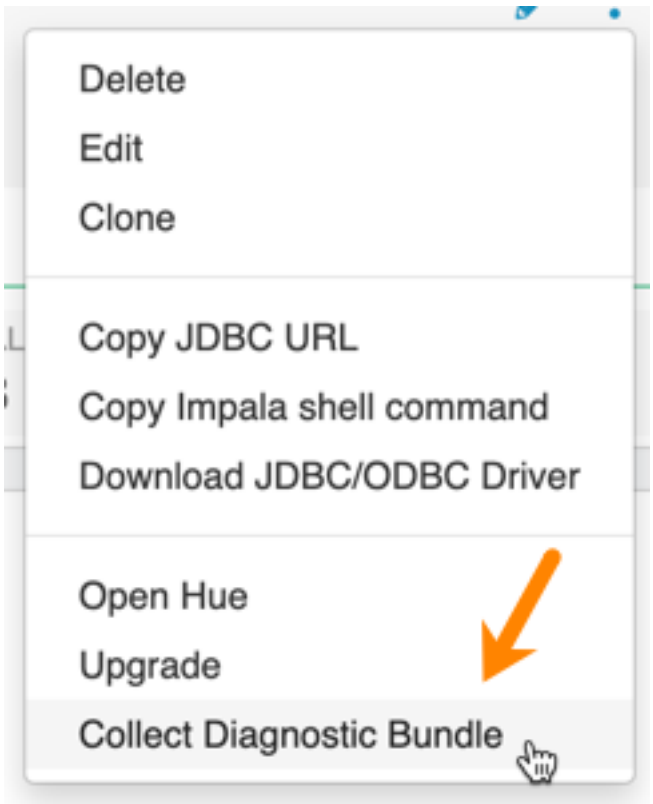

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2023-719: Cloudera Data Warehouse Backup/Restore of CDP Data Visualization incomplete

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature will encounter an issue with the restoration versions of Cloudera Data Visualization (CDV) older than CDV 7.1.6.2-3 due to a schema change in this release. If the Backup was taken from an older CDW environment that contained a version of CDV older than CDV 7.1.6.2-3, the Restore procedure will succeed. Though once the user opens the CDV Queries tab, the user could encounter the error message: “column jobs_jobschedule.owner_id does not exist…”

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete.

TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature might encounter an issue with the restoration of the Database Catalog, if the hue-query-processor.json configuration file of the Database Catalog has been edited. Even a minor edit to the hue-query-processor.json configuration file can result in this failure.

During the restoration process the Database Catalog will be created, but it will fail to start after a short period of time, and the Database Catalog will be in a Bad Health state on the CDW User Interface.

Inside the Kubernetes Cluster (Azure Kubernetes Service on Azure / Elastic Kubernetes Service on AWS) the StatefulSet of Hue Query Processor (hue-query-processor) is in CrashLoop. This is indicated with the following log in the Hue Query Processor StatefulSet Pod:

```
SQL State  : 3D000
Error Code : 0
Message    : FATAL: database "warehouse-1707832123-abcd_hueqpdb"
 does not exist



              at org.flywaydb.core.internal.jdbc.JdbcUtils.openC
onnection(JdbcUtils.java:65)
```

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW.

TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

Apache Hive (Hive) and Apache Impala (Impala) modify statements (DELETE/UPDATE/MERGE) on Apache Iceberg (Iceberg) V2 tables can corrupt the tables if there is a concurrent table compaction from Apache Spark. The issue happens when the compaction and modify statement run in parallel, and when the compaction job commits before the modify statement. In this case the position delete files of the modify statement still point to the old files. This means the following in case of

- DELETE statements
  - Deleting records pointing to old files have no effect
- UPDATE / MERGE statements
  - Deleting records pointing to old files have no effect
  - The table will also have the newly added data records
  - Rewritten records will still be active

This issue does not affect Apache NiFi (NiFi) and Apache Flink (Flink) as these components write equality delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the

rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

When working with Apache Hive (Hive) and Apache Iceberg (Iceberg) V2 tables, using the TRUNCATE statement may lead to unintended data deletion. This issue arises when the truncate command is applied to a branch of an Iceberg table. Instead of truncating the branch itself, the command affects the original (main) table, which results in unintended loss of data.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion.

Fixed issues in Cloudera Data Warehouse on Public Cloud

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

Hive-27643 CPDP-60943 Data Loss during compaction

Data loss no longer occurs during compaction when Apache Ranger policies for masking or row filtering are enabled and compaction users are included in the policies. Compaction queries are now automatically excluded from ALL ranger policies.

Enabling a private CDW environment in Azure Kubernetes Service is unstable

The networking issue at Azure Software Definition Layer (SDN) has been fixed by Microsoft. The issue affected CDW private environments while using the Azure Kubernetes Service (AKS) in 1.7.1-b755 (released Aug 30, 2023) - 1.8.4-b90 (released Jan 18, 2024). Enabling a private CDW environment in AKS is now available in 1.8.5-b35 (released Feb 29, 2024) on a General Availability (GA) basis. Cloudera now supports deploying a private CDW environment in Azure using AKS.

IMPALA-12681: Some local file descriptors not released when using remote spilling

The issue that occurred during remote spilling when writing spilled data to local buffers has been fixed. The disk space occupied by the file can now be reclaimed.

DWX-17368: Missing Root CA certificates in LdapConfig for ldaps should not result in error

If Root CA certificates are not passed as LdapConfigs, the certificates in the default path are used. In public cloud, DWX needs these certs for tls communication. This issue has been fixed by changing the error condition into informational log when CA certificates are not provided as part of LdapConfigs.

DWX-17157: Enable/Disable SSO doesn't set proper Impala configs

The CDW update logic now updates the SSO config related check to compare the current value of `EnableSSO` with previous value of `EnableSSO`.

DWX-17053: Catalog heap to pod memory configuration

The memory issue causing catalogd problems has been fixed by reserving 33% more memory for the catalogd pod than the maximum heap size for JVM.

DWX-17175: Impala Virtual Warehouse max executor limit issue

Earlier, you had to manually set the max executors limit to a preferred value less than 200, but a multiple of t-shirt size, when creating or editing Impala Virtual Warehouse. This issue has been resolved.

Technical Service Bulletins

TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW.

Behavior changes in Cloudera Data Warehouse on Public Cloud

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following behavior changes:

Summary:

Change in value for the hive.map.groupby.sorted property

Before this release:

The value of the hive.map.groupby.sorted property was set to 'true'.

After this release:

The value of the hive.map.groupby.sorted property is changed to 'false' to disable optimization. The change was introduced to address data correctness issues noticed in query results on tables with CLUSTER BY and SORT BY.

For more information, see HIVE-27876.

Summary:

Change in the upgrade and rebuild functionality

Before this release:

CDW provided the following three options to upgrade and rebuild Database Catalogs and Virtual Warehouses: Rebuild with same image version, rebuild with new image version, and upgrade.

After this release:

The KeepImageVersion option that was displayed when you rebuild the Virtual Warehouse has been removed. The rebuild operation preserves the image version and only redeploys the pods. The upgrade operation updates the CDW entities with the latest image version.

What's new (unsupported releases)

This section lists major features and updates for the Cloudera Data Warehouse (CDW) service on Public Cloud.

January 18, 2024

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

This release is a hot fix for an environment activation issue in CDW and other CDP data services. The latest EKS cluster deployment from AWS changes behaviors of CoreDNS. One of the changes breaks the functions of cdp-coredns-updater, causing a malfunction in CoreDNS. CDW 1.8.4 fixes the corefile issue and is compatible with the latest CoreDNS. For more information, see Known Issues, January 10, 2024.

January 10, 2024

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

AWS announced the Amazon Relational Database (RDS) Postgres 11 end of life is Feb 29, 2024. In Cloudera Data Platform, in an AWS environment, any new CDW created in this release 1.8.3-b130 Jan 10, 2024 or later will support Postgres 13 on RDS.

Any existing CDW in an AWS environments that uses Postgres 11 requires backup and restoration. You backup CDW in an AWS environment that supports Postgres 11 on RDS, and then restore CDW on an environment that supports Postgres 13 on RDS as described in "Backing up and restoring CDW".

December 19, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Upgrade your Virtual Warehouse to get the new CDW Runtime 2023.0.16.2-1 (Dec 19, 2023), which makes the following changes:

- CDPD-48059

  Unable to view and access S3 buckets from Hue configured with RAZ in CDW Public Cloud. For more information, including how to get this fix, see Fixed Issues for December 19, 2023.
- DWX-16923

  Fixes upgrade problem starting Hive Virtual Warehouse after upgrade.
- DWX-17088

  Addresses memory issues by disabling collection of KUBE-EVENTS for diagnostic bundle.
- DWX-17097

  Corrects metering problem causing overcharges.

  DWX-17110

  Populates several Grafana panels that were empty.
- DWX-17111 IMPALA-12486

  Missing database and table count metrics now appear in Grafana. For more information, including how to get this fix, see Fixed Issues for December 19, 2023.
- IMPALA-12580

  Fixes Iceberg exception when queries from Impala filter predicates.

December 1, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Upgrade your Virtual Warehouse to get the new CDW Runtime 2023.0.16.1-2 (released Dec 1, 2023), which fixed the following issues:

- CDPD-62960
- DWX-16690

See fixed issues for more information about the issues.

November 20, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Cloudera Data Warehouse Public Cloud 1.8.1-b248 changes, described in more detail below:

- Azure AKS 1.27 upgrade on page 113
- New AWS instance type on page 113
- Upgraded AWS and Azure environment security on page 114
- Diagnostic bundles for troubleshooting Data Visualization problems on page 114
- Resizing a Data Visualization instance on page 114
- Reduced permissions mode enhancement and tag change on page 114
- dbt adapters for using dbt with Hive, Impala and CDP on page 114

Cloudera Data Warehouse Public Cloud Runtime 2023.0.16.0-150 changes:

Hue

- Hue supports natural language query processing (Preview) on page 116
- Ability to deploy Hue at the environment level (Preview) on page 116

Iceberg

- Enhancement of expiring Iceberg snapshots on page 114
- Truncate partition support for Iceberg on page 115
- Insert into partition and insert overwrite partition support for Iceberg on page 115
- Iceberg branching and tagging technical preview on page 115

Impala

- Impala performance optimization for VALUES() expressions on page 115
- Impala skips scheduling bloom filter from full-build scan on page 115
- Impala events marked as 'Skip' occur prior to a manual REFRESH on page 115
- Allow setting separate mem_limit for coordinators on page 116

Azure AKS 1.27 upgrade

Cloudera supports the kubernetes version 1.27. In 1.8.1-b248 (released November 20, 2023), when you activate an environment, CDW automatically provisions Azure Kubernetes Service (AKS) 1.27. To upgrade to AKS 1.27 from 1.7.3 or earlier, you must backup and restore CDW. To avoid compatibility issues between CDW and AKS, upgrade to version 1.27.

Using the Azure CLI or Azure portal to upgrade the AKS cluster is not supported. Doing so can cause the cluster to become unusable and can cause downtime. For more information about upgrading, see Upgrading an Azure Kubernetes Service cluster for CDW.

New AWS instance type

This release supports the r6id.4xlarge AWS compute instance types. You select this instance type, or another supported one, when you activate your environment in CDW.

Private EKS API Server technical preview

In 1.8.1-b248 (released November 20, 2023), you can establish private connectivity between CDP services running on you cluster and AWS to prevent exposing data on the internet. To set up the Amazon Elastic Kubernetes Service (EKS) cluster in private mode and to enable the private EKS, run the following Beta CDP CLI command:

```
cdp dw create-cluster -\-aws-options "enablePrivateEKS=true" -\-environment-
crn "xyz"
```

The set up configures awsOptions. In this mode, a private endpoint uses the cluster-proxy (ccmv2) networking for control-plane to cluster communication. The freeipa security group is authorized in the eks-cluster's ingress rule, which opens the channel of communication from the control plane side through the freeipa pod.

This feature is a technical preview and not recommended for production deployments. Cloudera recommends that you use this feature in test and development environments only.

Upgraded AWS and Azure environment security

This release upgrades Istio security required for future Kubernetes compatibility. Hive Virtual Warehouses you create in 1.8.1-b248 (released Nov 20, 2023) will run Istio 1.19.0. Because the new Istio version supports only new versions of Hive helm charts, the following limitation exists: If you have the CDW_VERSIONED_DEPLOY, only new Hive image versions appear in UI when you create a new Hive Virtual Warehouse. For more information, see the known issue about Limited Hive image versions.

Diagnostic bundles for troubleshooting Data Visualization problems

You can collect a diagnostic bundle for troubleshooting Data Visualization, as well as Virtual Warehouse, Database Catalog, and environment/cluster. The diagnostic bundle is available for downloading and troubleshooting from the UI. For more information, see Diagnostic bundles for CDW and Kubernetes.

Resizing a Data Visualization instance

The size of the your Data Visualization instance is critical for achieving cost and performance goals. In CDW, after creating a Data Visualization instance you can change its size. You open the Data Visualization instance for editing, and in Size, select the size you want. When you click Apply Changes, the new size takes effect.

Reduced permissions mode enhancement and tag change

The dependent name of the tag key envID has been changed to Cloudera-Resource-Name, which increases strictness. For more information, see Reduced permissions mode template and Minimum set of IAM permissions required for reduced permissions.

dbt adapters for using dbt with Hive, Impala and CDP

You can access the dbt adapters for Hive and Impala from the Cloudera Data Warehouse service, which enable you to use the dbt data management workflow with Cloudera Data Platform. For more information, see Using dbt with Hive, Impala and CDP.

Support for Hive external data sources using data connectors

You can use Hive data connectors to map databases present in external data sources to a local Hive Metastore (HMS). The external data sources can be of different types, such as MySQL, PostgreSQL, Oracle, Redshift, Derby, or other HMS instances. You can create external tables to represent the data, and then query the tables. For more information, see Using Hive data connectors to support external data sources.

Enhancement of expiring Iceberg snapshots

In this release, you have more flexibility to expire snapshots. In addition to expiring snapshots older than a timestamp, you can now expire snapshots based on the following conditions:

- A snapshot having a given ID
- Snapshots having IDs matching a given list of IDs
- Snapshots within the range of two timestamps

You can keep snapshots you are likely to need, for example recent snapshots, and expire old snapshots. For example, you can keep daily snapshots for the last 30 days, then weekly snapshots for the past year, then monthly snapshots for the last 10 years. You can remove specific snapshots to meet the GDPR right to be forgotten requirements.

Truncate partition support for Iceberg

This release introduces the capability to truncate an Iceberg table. Truncation removes all rows from the table. A new snapshot is created. Truncation works for partitioned and unpartitioned tables.

Insert into partition and insert overwrite partition support for Iceberg

From Hive you can insert into, or overwrite data in, Iceberg tables that are statically or dynamically partitioned. For syntax and limitations, see "Insert into/overwrite partition support".

Iceberg branching and tagging technical preview

From Hive, you can manage the lifecycle of snapshots using the Iceberg branching and Iceberg tagging features. Branches are references to snapshots that have a lifecycle of their own. Tags identify snapshots you need for auditing and conforming to GDPR. Branching and tagging is available as a technical preview. Cloudera recommends that you use this feature in test and development environments. It is not recommended for production deployments.

Impala performance optimization for VALUES() expressions

Rewriting expressions in the Impala VALUES() clause can affect performance. If Impala evaluates an expression only once, especially a constant expression, the overhead might outweigh the potential benefits of a rewrite. Consequently, in this release, any attempts to rewrite expressions within the VALUES() clause has no impact. Impala skips expression rewrites entirely for VALUES during the analysis phase except rewrites of expressions separated by a compound vertical bar (||). These expressions are ultimately evaluated and materialized in the backend instead of the analysis phase.

The drop in performance caused by rewriting expressions might not follow a straightforward linear pattern. This drop becomes more pronounced as the number of columns increases. Using code generation for constant expressions in this context does not provide significant value. As part of this optimization, code generation is turned off for constant expressions within a UNION node if the UNION node is not within a subplan. This applies to all UNION nodes with constant expressions, not just those associated with a VALUES clause.

Here are some examples of queries of disabled expression rewrites and code generation for the UNION operator and VALUES clause.

```
select 1+2+3+4+5,2*1-1,3*3 union all select 1+2+3+4,5,6 union all select 7+1
-2,8+1+1,9-1-1-1;
```

```
insert into test_values_codegen values
  (1+1, '2015-04-09 14:07:46.580465000', base64encode('hello world')),
  (CAST(1*2+2-5 as INT), CAST(1428421382 as timestamp),
   regexp_extract('abcdef123ghi456jkl','.*?(\\d+)',0));
```

Impala skips scheduling bloom filter from full-build scan

PK-FK joins between a dimension table and a fact table are common occurrences in a query. Such joins often do not involve any predicate filters in the dimension table. As a result, a bloom filter generated from this kind of dimension table scan (PK) will most likely contain all values from the fact table column (FK). It becomes ineffective to generate this filter because it is unlikely to reject any rows, especially if the bloom filter size is large and has a high false positive probability (FPP) estimate.

As part of this optimization, Impala skips scheduling bloom filter from join node that has certain characteristics.

Impala events marked as 'Skip' occur prior to a manual REFRESH

If a table has been manually refreshed, the event processor skips any events occurring prior to the manual refresh. This optimization helps catalogd when it lags behind in processing events. Now, event processing has been optimized to determine if any manual refresh was executed after its eventTime. This, in turn, assists CatalogD in swiftly catching up with the HMS events. To activate this optimization, you must set enable_skipping_older_events to true.

Allow setting separate mem_limit for coordinators

The current mem_limit query option applies to all Impala coordinators and executors. This means that the same amount of memory gets reserved, but coordinators typically only handle the task of coordinating the query and thus may not necessarily require all the estimated memory. When we block the estimated memory on coordinators, it hinders the memory available for use by other queries.

The new MEM_LIMIT_COORDINATORS query option functions similarly to the MEM_LIMIT option but sets the query memory limit only on coordinators. This new option addresses the issue related to MEM_LIMIT and is recommended in scenarios where the query needs higher or lower memory on coordinators compared to the planner estimates.

Note: The MEM_LIMIT_COORDINATORS query option does not work in conjunction with MEM_LIMIT. If you set both, only MEM_LIMIT will be applied.

Hue supports natural language query processing (Preview)

Hue leverages the power of Large Language Models (LLM) to help you generate SQL queries from natural language prompts and also provides options to optimize, explain, and fix queries, ensuring efficiency and accuracy in data retrieval and manipulation. You can use several AI services and models such as OpenAI's GPT service, Amazon Bedrock, and Azure's OpenAI service to run the Hue SQL AI assistant. See SQL AI Assistant in Data Warehouse Public Cloud.

Ability to deploy Hue at the environment level (Preview)

Previously, you could run Hue only at a Virtual Warehouse-level. As a result, you would lose all query history when you delete or shut down a Virtual Warehouse. CDW now allows you to deploy Hue at the environment level and allows you to select a Virtual Warehouse from the Hue web interface. Query history and saved queries are retained as long as the environment is active. See Deploying Hue at Environment Level in Data Warehouse Public Cloud.

October 12, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

This release makes a HotFix available for the following issues:

- DWX-16387 : Hue UI unreachable after huebackend restart.
- DWX-16690: Unable to edit any mutable config present in flag file of impala-catalogd.

October 5, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Automatically backing up and restoring CDW

In earlier releases, the manual backup and restore feature replaced the in-place upgrade of AWS environments. This release adds automation to back up and restore procedures for AWS and Azure environments. You can still perform Hive and Impala in-place upgrades.

To get the supported Kubernetes version, you back up your old AWS or Azure environment and start up a new environment using the restoration process. The backup/restore feature saves your environment parameters, making it possible to recreate your environment with the same settings, URL, and connection strings you used in your previous environment.

CDP CLI Start/Stop Azure Kubernetes Service (AKS) Technical Preview

Using the Beta version of the CDP CLI, you can now start and stop a Cloudera Data Warehouse (CDW) cluster on Azure using the following commands:

- resumeCluster <cluster ID>

  Starts the cluster.

- suspendCluster <cluster ID>

  Stops the cluster.

The following prerequisites are required to use resumeCluster and suspendCluster :

- You obtained a cluster ID parameter to pass with the command.
- The cluster you want to stop is running.
- You stopped the Virtual Warehouses and Database Catalogs in the cluster you want to stop
- You stopped the cluster before attempting to start it.
- The cluster you want to start or stop is not in an error state.

The Microsoft documentation describes how resumeCluster and suspendCluster starts and stops the AKS cluster, including every node pool and AKS control plane, on the cloud provider side. The stopped AKS incurs zero cost. The Postgres database, which belongs to the environment, is not stopped.

Note: Heed the Microsoft documentation, which says "Don't repeatedly stop and start your clusters. This can result in errors. Once your cluster is stopped, you should wait at least 15-30 minutes before starting it again."

Limitations

- The capability to start and stop an AKS cluster is not available from the UI.
- You cannot start or stop an AKS cluster from the UI.
- You cannot manually trigger starting or stopping an AKS cluster from the cloud provider either by using the Azure CLI or the Azure portal.

  Manually triggering start and stop on the cloud provider does not synchronize the state of the AKS instance with Cloudera control plane. The AKS continues running and the CDW UI indicates a running state.

Impala Virtual Warehouse spills temporary data to S3 automatically

When you create an Impala Virtual Warehouse, Impala is now automatically configured to write temporary data to the S3 Data Lake bucket. No configuration is required to enable the spill to S3. However, you must configure an Impala policy for permission to the scratch location on the Data Lake bucket.

Kubernetes dashboard adds Azure support and has reached GA status

From AWS or Azure environments, you can use the K8S dashboard to view the state of resources, such as CPU and memory usage, see the status of pods, and download logs. The dashboard can provide insights into the performance and health of a CDW cluster.

August 30, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Cloudera Data Warehouse Public Cloud 1.7.1-b755 changes, described in more detail below:

- Amazon Machine Image updates on page 118
- AWS Elastic Kubernetes Service 1.26 support on page 118
- Amazon EKS 1.23 and from EKS 1.23 to 1.24 upgrades on page 118
- AWS restricted policy updates on page 118
- Automatic backup and restoration of Hue on page 118
- Correcting the Virtual Warehouse Size on page 119
- Deprecation of DAS on page 119
- Instance types available for compute nodes on page 119
- Java tool options now configurable with limitations on page 119
- Managing high partition loads on page 119
- Private Cluster control removed from UI on page 119
- Query results cache support on page 119

Cloudera Data Warehouse Public Cloud Runtime 2023.0.15.0-243 changes:

Hive

- Histogram statistics on page 119

Iceberg

- Iceberg position delete feature support on page 120
- Hive to Iceberg table migration from Impala on page 119

Impala

- Impala support for complex types on page 120
- Support ORDER BY for collections of fixed length types in SELECT list on page 120
- Support collections of fixed length types as non-passthrough children of unions on page 120
- Allow implicit casts between numeric and string types when inserting into table on page 120
- Improved memory estimation for aggregates and pre aggregates on page 120
- Improved CPU costing on page 120
- Added processing cost to control scan parallelism on page 120
- Impala WebUI improvements on page 121
- Skip reloading file metadata for some ALTER_TABLE events on page 121
- JWT auth for Impala on page 121
- Native High Availability (HA) for Impala Catalog Service on page 122
- Codegen for STRUCT type on page 122

Amazon Machine Image updates

This release supports dynamically updating the Amazon Machine Image (AMI) to prevent potential problems running workloads on an old AMI. You can update the AMI of the Cloudformation stack while keeping the current Elastic Kubernetes Service (EKS) version.

AWS Elastic Kubernetes Service 1.26 support

New CDW clusters using AWS environments you activate in this release 1.7.1-b755 (released August 30, 2023) of Cloudera Data Warehouse will use Amazon Kubernetes (EKS) version 1.26. For more information, see Upgrading Amazon Kubernetes Service.

Amazon EKS 1.23 and from EKS 1.23 to 1.24 upgrades

The release includes reduced permissions mode support for the following Amazon Elastic Kubernetes Service upgrades:

- Upgrading to EKS 1.23
- Upgrading from EKS 1.23 to 1.24

AWS restricted policy updates

The AWS restricted policy has been updated to conform to AWS file size requirements. The update divides the policy into two parts as described in the documentation.

Automatic backup and restoration of Hue

The CDW backup and restore processes for Hue have been automated. Manually backing up and restoring Hue is still available, but optional.

Correcting the Virtual Warehouse Size

After creating an Impala Virtual Warehouse, you can tune, or correct the T-shirt size of executor groups that drive the Impala Virtual Warehouse. The size of the executor groups is critical for achieving cost and performance goals.

Deprecation of DAS

Hue now replaces Data Analytics Studio (DAS). DAS has been deprecated and is no longer available in CDW Public Cloud. DAS features to support Hive and Tez such as running queries, defining HPL/SQL, the Job Browser, query explorer, query compare, and more, have been migrated to Hue, and the Hue Query Processor. After you upgrade to this release, you will not see the option to launch DAS from your Virtual Warehouse. Cloudera recommends you use Hue for all use cases where you might have previously used DAS.

Instance types available for compute nodes

In this release, when you activate an AWS environment, you can select the compute instance type you want to use. This release adds additional instance types you can select when you activate an Azure environment.

Java tool options now configurable with limitations

After creating an Impala Virtual Warehouse, you can change the XMX Java tool option.

Managing high partition loads

In this release, you can identify an error related to high partition workloads and tune your Hive Virtual Warehouse to run successfully.

Private Cluster control removed from UI

Enable Private Cluster has been removed from the environment activation dialog. Use CDP CLI for advanced configurations.

Query results cache support

Unified Analytics now supports the caching of Hive/Impala query results. Caching results of repetitive queries can reduce the load.

Histogram statistics

In this release, when you generate column statistics in a Hive Virtual Warehouse in Unified Analytics mode, you can create histogram statistics on columns. By default, the histogram stats are not created. You enable generation of histogram statistics by setting a Hive property: set hive.stats.kll.enable = true;

You can then run the ANALYZE command as usual:

```
ANALYZE TABLE [table_name] COMPUTE STATISTICS for COLUMNS [comma_separated_c
olumn_list];
```

Histogram statistics are supported for numeric data types, date, timestamp and boolean types but not for string/varchar/char columns. Histograms are used to estimate selectivity of range predicates (predicates involving <, <=, >, >= and BETWEEN). The better selectivity estimate allows the optimizer to generate more optimal query plans and improve performance for such queries.

Hive to Iceberg table migration from Impala

In this release, you can use Impala, as well as Hive, to migrate a Hive table to Iceberg tables. You use the ALTER TABLE statement. Syntax is described in Migrate Hive table to Iceberg feature and a step-by-step procedure is covered in Migrating a Hive table to Iceberg.

Iceberg position delete feature support

In this release, Impala, in addition to Hive, can delete Iceberg V2 tables using position delete files, a format defined by the Iceberg Spec. A position delete query evaluates rows from one table against a WHERE clause, and delete all the rows that match WHERE conditions.

Impala support for complex types

Complex types are now supported in the SELECT list. Although collections and structs were previously supported, nesting and mixing of complex types was not. For more information, including limitations, see "Allowing embedding complex types into other complex types" in Complex types.

Support ORDER BY for collections of fixed length types in SELECT list

This release supports collections of fixed length types in the sorting tuple. However, you cannot sort by these collection columns, but they can be in the SELECT list along with other column(s) by which you sort.

Support collections of fixed length types as non-passthrough children of unions

This release adds support for collections of fixed length types as non-passthrough children of unions. Plain UNIONs are not supported yet for any collections, but UNION ALL operations are supported.

Example:

```
select id, int_array from complextypestbl
union all select cast(id as tinyint), int_array from complextypestbl
```

Allow implicit casts between numeric and string types when inserting into table

The current implementation requires explicit casts for numeric and string-based literals. However, this release relaxes the implicit casting rules for these cases. This is controlled through a query option allow_unsafe_casts and turned off by default. This query option allows implicit casting between some numeric types and string types.

Improved memory estimation for aggregates and pre aggregates

This release introduces new query options to improve memory estimation for aggregation nodes. Also introduces better cardinality estimates to help in capping memory limits early on during query planning.

Improved CPU costing

This release introduces some changes to the query planner to improve parallel sizing and resource estimation. These changes are done for workload-aware autoscaling and will be available as query options. These additional query options are added for tuning purposes. This new functionality will allow more customers to enable multi-threaded queries globally for improved performance.

Added processing cost to control scan parallelism

Before this release, when a user executed a query with COMPUTE_PROCESSING_COST=1, Impala relied on the MT_DOP option to decide the degree of parallelism of the scan fragment. This release introduces the scan node's processing cost as another factor to consider raising scan parallelism beyond MT_DOP.

Scan node cost now includes the number of effective scan ranges. Each scan range is given a weight of (0.5% * min_processing_per_thread), which roughly means that one scan node instance can handle at most 200 scan ranges. This release also introduces a new query option MAX_FRAGMENT_INSTANCES_PER_NODE to cap the maximum number of fragment instances per node. This newly introduced query option works in conjunction with PROCESSING_COST_MIN_THREADS.

Impala WebUI improvements

This release enhanced the Impala daemon's Web UI to display the following additional details:

- Backends start time and version: In a large cluster, you can now use the Impala daemon's Web UI to view the start time and version for all the backends.
- Query performance characteristics: For a detailed report on how a query was executed and to understand the detailed performance characteristics of a query, you can use the built-in web server's UI and look at the timeline shown in the Gantt chart. This chart is an alternative to the PROFILE command and is a graphical display in the WebUI that renders timing information and dependencies.
- Export query plan and timeline: To understand the detailed performance characteristics for a query, you issue the PROFILE command in impala-shell immediately after executing a query. As an alternative to the profile download page, this release added support for exporting the graphical query plan and also for downloading the timeline in SVG/HTML format. Once you export the query plan or the timeline, memory resources consumed from the ObjectURLs get cleared.
- Historical/in-flight query performance: You can now use the query list and query details page to analyze historical or in-flight query performance by viewing the memory consumed, the amount of data read, and other information about the query.
- Aggregate CPU node utilization: You can now see the recent aggregate CPU node utilization samples for the different nodes.
- Scaling of timeticks and fragment timing diagram for better accessibility: You can now use the query timeline display to scroll horizontally through the fragment timing diagram and utilization chart. You can also zoom by horizontally scaling through mouse wheel events in addition to increasing/decreasing the precision of timetick values.

Skip reloading file metadata for some ALTER_TABLE events

Before this release, EventProcessor ignored trivial ALTER_TABLE events that only modify tblproperties like "transient_lastDdlTime," "totalSize," "numFilesErasureCoded," and "numFiles". For other non-rename ALTER_TABLE events, it triggered a full refresh on the table, which becomes expensive for tables with a large number of partitions or files.

From this release, to be more cost-efficient, the event processor skips reloading file metadata for some ALTER_TABLE events.

The following list contains the events that skip reloading file metadata:

- changing table comment
- adding/dropping columns
- changing column definition (name/type/comment)
- changing ownership
- setting customized tblproperties

For interoperability purposes, this release introduces a new start-up flag 'file_metadata_reload_properties' to list the table properties that need the file metadata reloaded when the properties are changed.

Note: To disable this optimization (in case of any unexpected issues), set file_metadata_reload_properties to an empty string.

JWT auth for Impala

Impala clients, such as Impala shell, can now authenticate to Impala using a JWT instead of a username/password. To connect to Impala using JWT authentication, specify JWT command-line options to the impala-shell command interpreter and enter the password when prompted.

Native High Availability (HA) for Impala Catalog Service

The High Availability (HA) mode of catalog service in CDW reduces the outage duration of the Impala cluster when the primary catalog service fails. Before this release, catalog HA was supported using the K8s leader election mechanism, and now it is natively supported in Impala.

Codegen for STRUCT type

Codegen uses query-specific information to generate specialized machine code for each query. As an Impala user, when you run a standard query, the query optimizer generates an optimized query plan and passes it to the executor for processing. With the codegen capability for STRUCT type in the SELECT list, the query specific information is converted to machine code for faster execution.

Before this release, having structs in the select list was only supported with codegen turned off. This release lifts this restriction, adding full codegen support for structs in the select list.

Example:

```
select small_struct from complextypes_structs
```

June 29, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

This hotfix release fixes problems in backing up and restoring CDW, primarily the issue restoring a Virtual Warehouse executor count. The workaround to the problem described in TSB-2023-683 is no longer necessary.

May 30, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Azure AKS 1.26 upgrade

The new kubernetes version 1.26 has breaking changes; thus, starting earlier workload (Impala, Hive, Hue) versions is not supported, and this option is now disabled. In this release 1.6.4-b161 (released May 30, 2023), when you activate an environment, CDW automatically provisions Azure Kubernetes Service (AKS) 1.26. To upgrade to AKS 1.26 from 1.6.3 or earlier, you must backup and restore CDW. To avoid compatibility issues between CDW and AKS, upgrade to version 1.26.

Using the Azure CLI or Azure portal to upgrade the AKS cluster is not supported. Doing so can cause the cluster to become unusable and can cause downtime. For more information about upgrading, see Upgrading an Azure Kubernetes Service cluster for CDW.

Azure AKS managed identity now mandatory

You must now specify a user-assigned, managed identity for the Azure Kubernetes Service (AKS) cluster when you activate the Azure environment from the CDP CLI in 1.6.4-b161 (released May 30, 2023). In 1.6.3 (released May 5, 2023), specifying the managed identity is also mandatory when you activate the Azure environment from the UI. A managed identity is required because AKS does not support certificate authentication for service principals. The cluster creation operation will fail if the userAssignedManagedIdentity is not set to your managed identity.

New Backup/Restore CDP CLI commands

The following CDP CLI 0.9.87, including Beta DW CLI commands are available and support backing up and restoring CDW:

- dw backup-cluster
- dw restore-cluster

May 5, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Cloudera Data Warehouse Public Cloud 1.6.3-b319 changes, described in more detail below:

- AWS Kubernetes version 1.24 support on page 123
- Selective support for AWS Kubernetes version 1.22 upgrade on page 123
- Changes to the standard required IAM permissions and AWS restricted policy on page 124
- Synchronized metadata across Impala Virtual Warehouses on page 124
- Upgrade and rebuild functionality on page 124
- New requirements in Azure to provide managed identity on page 125
- Ranger authorization (RAZ) in CDW on page 125
- Accessing buckets in a RAZ environment on page 125
- Configuring token-based authentication in CDW on page 125
- Impala coordinator shutdown in Unified Analytics on page 125
- Anti join support in Unified Analytics on page 125
- Debugging enhancement: HiveServer OOM dump on page 125
- Kubernetes dashboard (Preview) on page 126
- Workload Aware Auto-Scaling (Preview) on page 126

Cloudera Data Warehouse Public Cloud Runtime 2023.0.14.0-155 changes:

- Iceberg manifest caching on page 126
- Iceberg Execute Rollback feature from Impala on page 126
- Incremental rebuild of Iceberg materialized views on page 126
- Query hint for Impala table cardinality on page 126
- Improvement in compaction and Impala metadata refresh on page 126
- Optimize Refresh/Invalidate event processing on page 126
- Support for custom hash schema for Kudu range tables on page 126
- New shell option "hs2_fp_format" on page 127
- Support for complex types on page 127
- Ability to view Impala query details in Hue
- Ability to access S3 buckets from Hue with RAZ is GA on page 127
- Ability to access ADLS Gen2 containers from Hue with RAZ is GA on page 127
- The “enable_queries_list” configuration has been removed from Hue jobbrowser safety valve section on page 127

AWS Kubernetes version 1.24 support

New CDW clusters using AWS environments you activate in this release 1.6.3-b319 (released May 5, 2023) of Cloudera Data Warehouse will use Amazon Kubernetes (EKS) version 1.24.

Selective support for AWS Kubernetes version 1.22 upgrade

If you activated your pre-existing AWS environment in CDW version 1.4.2-b118 (released Aug 4, 2022) or later, you can upgrade to EKS 1.22 in this release 1.6.3-b319 (released May 5, 2023). AWS environments activated in earlier versions are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version of your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

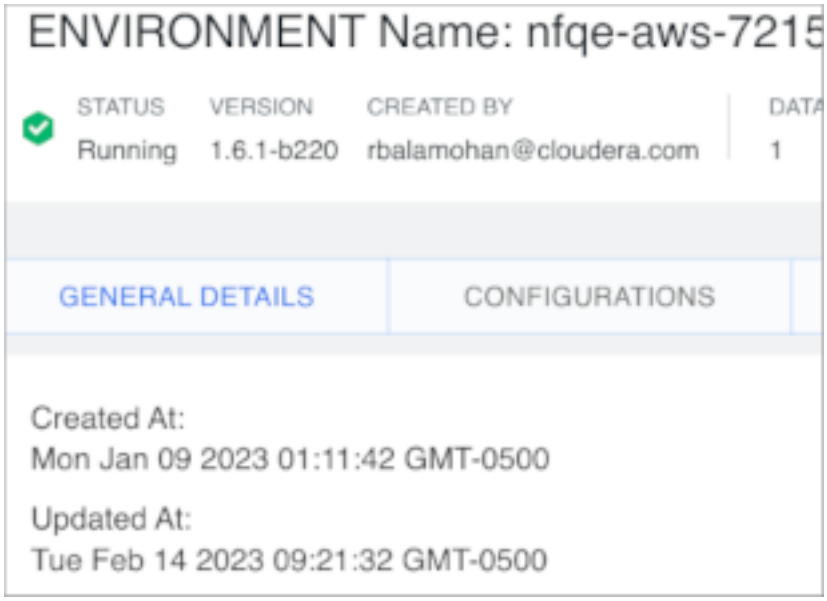

Changes to the standard required IAM permissions and AWS restricted policy

In this release, as an AWS environment user, you must make the following IAM policy updates:

- AWS restricted policy

  In "Sid": "gocode", add "iam:ListAttachedRolePolicies", only if you have a Ranger Authorization (RAZ) environment.
- Minimum set of IAM permissions required for reduced permissions mode

  Add "autoscaling:UpdateAutoScalingGroup".

  Add "iam:DeleteRolePolicy".

  Add "iam:ListAttachedRolePolicies".

Synchronized metadata across Impala Virtual Warehouses

Using Impala Virtual Warehouses that share a Database Catalog is easier in this CDW release. In past releases, after making changes to data and then refreshing tables or invalidating metadata from your Virtual Warehouse, only the catalog metadata and coordinator metadata for that particular Virtual Warehouse were affected. You had to rerun the commands from each Virtual Warehouse to synchronize metadata across multiple Virtual Warehouses that share a Database Catalog.

This release introduces an enhancement that raises events in the Hive metastore. Catalog daemons process events synchronously across all Virtual Warehouses that share metadata. Metadata is refreshed/invalidated in parallel across all your Virtual Warehouses. You need to run the commands only once in any one of your Virtual Warehouse.

To get this feature, you must upgrade your Virtual Warehouse to this release 1.6.3-b319 that has runtime 2023.0.14.0-155 (released May 5, 2023). You must also set the Impala catalogd flagfile key enable_reload_events to true. Newly created Virtual Warehouses use Impala version 2023.0.14.0-155, which has this feature enabled by default. For more information, including how to disable this feature, see Disabling metadata synchronization.

This enhancement does not synchronize metadata when you refresh tables or invalidate metadata from a Data Hub cluster.

Upgrade and rebuild functionality

The generic Database Catalog and Virtual Warehouse version upgrade process has changed. When you click the Upgrade button to upgrade your Virtual Warehouse or Database Catalog version, the upgrade that occurs includes the

rebuild functionality, which cleans up and redeploys your workload. For more information, Rebuilding a Database Catalog and Rebuilding a Virtual Warehouse.

New requirements in Azure to provide managed identity

During the Azure environment activation and private AKS activation, you must provide the resource ID of a user-assigned managed identity to create the Azure Kubernetes cluster resource.

Ranger authorization (RAZ) in CDW

In this release, if your Data Lake enables Ranger Authorization (RAZ), CDW will be RAZ-enabled to provide authentication in Hue, mainly. If you have an AWS environment that predates the option to enable RAZ for Cloudera Data Warehouse, you might want to manually enable RAZ in CDW.

Accessing buckets in a RAZ environment

In an environment that enables Ranger Authorization (RAZ), you must configure permissions to access an S3 bucket. "Accessing buckets in a RAZ environment" describes how to configure the permissions depending on the AWS account that owns the bucket.

Configuring token-based authentication in CDW

Using a JSON web token (JWT), your Virtual Warehouse client user can sign on to your Virtual Warehouse for a period of time instead of entering single-sign on (SSO) credentials every time your user wants to run a query. You use Impyla to access an Impala Virtual Warehouse and a JDBC connection to access a Hive Virtual Warehouse. For more information, see "Configuring token-based authentication".

Impala coordinator shutdown in Unified Analytics

Cloudera Data Warehouse has supported Impala coordinator shutdown for some time, but not in Unified Analytics until this release. When you create an Impala Virtual Warehouse, and enable Unified Analytics, you can configure Impala coordinator shutdown in Unified Analytics.

Anti join support in Unified Analytics

In this release, Unified Analytics enables an optimization for converting a join having a null filter to perform a SQL anti join.

Debugging enhancement: HiveServer OOM dump

In this release, when a hive-server pod out-of-memory (OOM) event occurs, a corresponding dump will be available in the HiveServer dumps directory:

```
<external-bucket>/clusters/<env-id>/<dbc-id>/warehouse/debug-artifacts/hive/
<compute-id>/hive-dumps/<node-name>/heapdump/dump.hprof
```

For example, the full path to the dump file looks something like this:

```
sharecdwdev-bucket-7cv9-dwx-external/clusters/env-9d7cv9/warehouse-167591873
2-q9m2/warehouse/debug-artifacts/hive/compute-1675918927-q8hs/hive-dumps/ip-
192-168-223-136.us-west-2.compute.internal/heapdump/hiveserver2-oom-dump.hpr
of
```

In the example above, ip-192-168-223-136.us-west-2.compute.internal is the name of the HiveServer node on which the pod was running.

Kubernetes dashboard (Preview)

A technical preview of the K8S dashboard is available in this release. You can view the state of resources, such as CPU and memory usage, see the status of pods, and download logs. The dashboard can provide insights into the performance and health of a CDW cluster.

Workload Aware Auto-Scaling (Preview)

Workload Aware Auto-Scaling (WAAS) is available as a technical preview. WAAS allocates Impala Virtual Warehouse resources based on the workload that is running. You choose an executor group set, analogous to a range of nodes instead of the fixed size of the previous auto-scaling implementation. Configuring WAAS requires an entitlement. Contact your account team if you are interested in previewing the feature.

Iceberg manifest caching

Apache Iceberg provides a mechanism to cache the contents of Iceberg manifest files in memory. The manifest caching feature helps to reduce repeated reads of small Iceberg manifest files by Impala Coordinators and Catalogd. You configure manifest caching in CDW. You can enable or disable caching and set a few other caching properties.

Iceberg Execute Rollback feature from Impala

In the event of a problem with your table, you can reset a table to a good state as long as the snapshot of the good table is available. You can roll back the table data based on a snapshot id or a timestamp. The Describe History feature output includes information that helps you use snapshot-related features.

Incremental rebuild of Iceberg materialized views

In this release, the materialized view is automatically updated under certain conditions. After inserting data into a source table of a materialized view, query rewriting is automatically enabled. An incremental rebuild, which updates only the changed part of the view, occurs. Incremental rebuilds tend to update the view much faster than full rebuilds.

Query hint for Impala table cardinality

This release supports a new TABLE_NUM_ROWS query hint to specify a table cardinality for cases where statistics are missing or invalid.

Improvement in compaction and Impala metadata refresh

In this release, handling of the COMMIT_COMPACTION_EVENT has been improved. During compaction, HMS raises events for ACID tables. Impala metadata is refreshed automatically.

Optimize Refresh/Invalidate event processing

Before this release, some metadata consistency issues led to query failures. These failures happen because the metadata updates from multiple coordinators cannot differentiate between self-generated events and those that are generated by a different coordinator. This issue is resolved in this release by adding a coordinator flag to each event, and when processing these events, the coordinator flag is checked to decide whether to ignore the event or consider it.

Support for custom hash schema for Kudu range tables

Impala now includes CREATE TABLE and ALTER TABLE syntax to allow a Kudu custom hash schema. HASH syntax within a partition is similar to the table-level syntax except that HASH clauses must follow the PARTITION clause, and commas are not allowed within a partition. You can use the SHOW HASH SCHEMA statement to view the hash schema information for each partition.

```
CREATE TABLE t1 (id int, c2 int, PRIMARY KEY(id, c2))
       PARTITION BY HASH(id) PARTITIONS 3 HASH(c2) PARTITIONS 4
       RANGE (c2)
       (
```

```
        PARTITION 0 <= VALUES < 10
        PARTITION 10 <= VALUES < 20
        HASH(id) PARTITIONS 2 HASH(c2) PARTITIONS 3
        PARTITION 20 <= VALUES < 30
        )
        STORED AS KUDU;
        ALTER TABLE t1 ADD RANGE PARTITION 30 <= VALUES < 40
        HASH(id) PARTITIONS 3 HASH(c2) PARTITIONS 4;
```

New shell option "hs2_fp_format"

This new option sets the printing format specification for floating point values when using HS2 protocol. The default behavior makes the values handled by Python's str() built-in method. Use '16G' to match Beeswax protocol's floating-point output format.

Support for complex types

A SELECT * statement did not expand to complex types to be compatible with earlier versions of Impala that did not support complex types in the result set. In the older versions of Impala, queries using SELECT * skip complex types by default and only expanded to primitive types even when the table contained complex-typed columns. This release adds a new query option EXPAND_COMPLEX_TYPES to include complex types in the SELECT * list.

Ability to view Impala query details in Hue

You can now view Impala query details, query plan, execution summary, and query metrics on the new **Impala Queries** tab on the **Job Browser** page using Hue in CDW, and use this information to tune and optimize your queries. You can also compare two queries and view all query details side-by-side. For more information, see Viewing Impala query details.

Ability to access S3 buckets from Hue with RAZ is GA

Granting fine-grained access to the per-user home directories in Amazon S3 and accessing them from the S3 File Browser in Hue using RAZ is GA. See Accessing S3 bucket from Hue in CDW with RAZ.

Ability to access ADLS Gen2 containers from Hue with RAZ is GA

Granting fine-grained access to the per-user home directories on ADLS Gen2 containers and accessing them from the ABFS File Browser in Hue using RAZ is GA. See Accessing ADLS Gen2 containers from Hue in CDW with RAZ.

The “enable_queries_list” configuration has been removed from Hue jobbrowser safety valve section

The enable_queries_list configuration in the CDW Hue safety valve was used to display or hide the **Queries** tab on the **Job Browser** page. This configuration has been removed. The **Queries** tab is displayed by default based on the type of Virtual Warehouse, whether it is Hive or Impala. You can override the query_store configuration and hide the **Queries** tab. For more information, see Hue configurations in Cloudera Data Warehouse.

February 14, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

AWS Kubernetes version upgrade

With this hot fix, the broken Upgrade button for upgrading an existing cluster to EKS 1.2.1 is now fixed. For more information about upgrading, see What's New, February 7, 2023.

February 7, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

AWS Elastic Kubernetes Service (EKS) version upgrade

The CDW application uses Kubernetes (K8S) clusters to deploy and manage Hive and Impala in the cloud. Kubernetes versions are updated every 3 months on average. When the version is updated, minor versions are deprecated.

To avoid compatibility issues between CDW and AWS resources, you must upgrade the version of Kubernetes that supports your existing CDW clusters to version 1.21.

AWS environments you activate using this release of Cloudera Data Warehouse will use version 1.22.

Important: Check to make sure your AWS environment has been migrated from Helm 2 to Helm 3 before you begin upgrading the Kubernetes version.

If a MIGRATE icon appears in the upper right corner of the environment tile, your AWS environment has not been migrated from Helm 2 to Helm 3. Update the Helm package manager for your environment before you attempt to upgrade its Kubernetes version.

For information about upgrading the AWS EKS Kubernetes version, see Upgrade CDW for EKS upgrade.

Support for Amazon Elastic Kubernetes Service 1.22 cluster

This release 1.6.1-b258 (released Feb 7, 2023) automatically uses and provisions Amazon Elastic Kubernetes Service (EKS) 1.22 when you activate an environment from CDW. In this release, upgrading a cluster to EKS 1.22 is not supported.

Changes to the managed policy ARN, standard IAM permissions, and restricted permissions policy

In this release, as an AWS environment user, you must update the managed policy ARN to handle Kubernetes CSI drivers for EBS and EFS. You must also update your restricted permissions policy if you use it.

Make the following changes:

- Restricted policy changes for updating a managed policy ARN

  Add "arn:aws:iam::<AWS_ACCOUNT>:policy/<noderole-inline-policy>" as shown in "Attaching a managed policy ARN".
- Restricted permissions policy

  Add "elasticfilesystem:PutFileSystemPolicy", to the ResourceTag object and move the "elasticfilesystem:CreateFileSystem", from the CloudFormation object to the RequestTag object, as shown in AWS restricted permissions policy.

Apache Iceberg GA in CDW

This release introduces the general availability of ACID transactions with Iceberg v2 tables from Hive in CDW Runtime 2023.0.13.0-20 (released 2023-2-7)). CDW Runtime 2022.0.12.0-90 (released 2022-12-13), introduced the general availability of ACID transactions with Iceberg V2 tables from Impala. You can run Apache Iceberg ACID transactions within some of the key data services in the Cloudera Data Platform (CDP) public cloud (AWS and Azure), including Cloudera Data Warehouse. From Hive or Impala, you use Apache Iceberg features in CDW, which include time travel, create table as select, and schema and partition evolution.

To access these features, create a new Virtual Warehouse or upgrade an existing one.

Support for Iceberg tables in Avro (Preview)

In this release, you can read Iceberg tables in Avro from Impala. There is a related known issue with using the DECIMAL data type in Avro this release.

Reading Iceberg tables in Avro format from Impala is available as a technical preview. Cloudera recommends that you use this feature in test and development environments. It is not recommended for production deployments.

Enhanced Iceberg support for materialized views

In this release, you can create a materialized view of an Iceberg V1 or V2 table based on an existing Hive table or an Iceberg table. Automatic rewriting of the materialized view occurs under certain conditions.

Iceberg load data inpath feature

From Impala, you can now load data into an Iceberg table using the load data inpath feature.

Querying Data Hub Kudu tables from an Impala Virtual Warehouse using Kudu

After configuring an Impala Virtual Warehouse to connect to Kudu, you can create Kudu tables using Impala clients, such as Hue, the Impala shell, or JDBC/ODBC. You can also ingest data using Spark/NiFi and query using Impala.

Flexible allow lists for Kubernetes cluster and load balancer

In previous releases, there was a single allow list for IP Classless Inter-Domain Routing (CIDRs) from which the Kubernetes cluster or the load balancer should accept incoming connections. In this release, you configure two separate allow lists for accepting incoming connections:

- Enable IP CIDR for Kubernetes cluster
- Enable IP CIDRs for the load balancer

Impala enhancements

This release of Cloudera Data Warehouse includes the following new Impala features:

- Binary support: Impala now supports BINARY columns for all table formats except Kudu. See the BINARY support topic for more information on using this arbitrary-length byte array data type in CREATE TABLE and SELECT statements.
- ALTER VIEW support: Before this release, altering only the VIEW definition, VIEW name, and owner was supported. Impala now supports altering the table properties of a VIEW by using the set tblproperties clause.
- Push down date literals to Kudu scanner: Impala now allows creating and pushing down Kudu predicates from the DATE type.

  Example:

  ```
  select * from functional_kudu.date_tbl where date_col = DATE "1970-01-01";

              PLAN-ROOT SINK
              |
              00:SCAN KUDU [functional_kudu.date_tbl]
              kudu predicates: date_col = DATE '1970-01-01'
              row-size=12B cardinality=1
              ---- DISTRIBUTEDPLAN
              PLAN-ROOT SINK
              |
              01:EXCHANGE [UNPARTITIONED]
              |
              00:SCAN KUDU [functional_kudu.date_tbl]
              kudu predicates: date_col = DATE '1970-01-01'
              row-size=12B cardinality=1
  ```
- Fix untracked memory in KRPC: Improved memory estimation of queries by accounting for untracked memory in KrpcDataStreamSender.
- Redhat UBI8 images: To address multiple CVEs, Impala images are built using UBI8 base image.

December 13, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Apache Iceberg V2

This release introduces the general availability of ACID transactions with Iceberg v2 tables from Impala in Cloudera Data Platform (CDP).

Apache Iceberg ACID transactions run within some of the key data services in the Cloudera Data Platform (CDP) public cloud (AWS and Azure), including Cloudera Data Warehouse (CDW).

The following features are included in this release:

- Impala reads of Iceberg V2 tables (row-level deletes or updates)
- Materialized views of Iceberg tables

Iceberg V2 documentation.

November 21, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

AWS EKS Kubernetes version upgrade

The CDW application uses Kubernetes (K8S) clusters to deploy and manage Hive and Impala in the cloud. Kubernetes versions are updated every 3 months on average. When the version is updated, minor versions are deprecated.

To avoid compatibility issues between CDW and AWS resources, you must upgrade the version of Kubernetes that supports your existing CDW clusters to version 1.21.

AWS environments you activate using this release of Cloudera Data Warehouse will use version 1.22.

Important: Check to make sure your AWS environment has been migrated from Helm 2 to Helm 3 before you begin upgrading the Kubernetes version.

If a MIGRATE icon appears in the upper right corner of the environment tile, your AWS environment has not been migrated from Helm 2 to Helm 3. Update the Helm package manager for your environment before you attempt to upgrade its Kubernetes version.

For information about upgrading the AWS EKS Kubernetes version, see Upgrade CDW for EKS upgrade.

Support for Amazon Elastic Kubernetes Service 1.22 cluster

This release 1.5.1-b110 (released November 22, 2022) automatically uses and provisions Amazon Elastic Kubernetes Service (EKS) 1.22 when you activate an environment from CDW. In this release, upgrading a cluster to EKS 1.22 is not supported.

Forward logs to your observability system

In this release, you can forward logs from environments activated in Cloudera Data Warehouse (CDW) to observability and monitoring systems such as Datadog, New Relic, or Splunk. You configure a CDW environment for these systems using the UI to provide a fluentd configuration.

Rebuild a Database Catalog or Virtual Warehouse

Rebuilding a Database Catalog and rebuilding a Virtual Warehouse, you clean up resources and redeploy your Database Catalog and Virtual Warehouse using your existing DBC or VW image, or the latest available image. You might consider rebuilding to get a feature in a later DBC or VW image, to perform housekeeping, or to troubleshoot a problem.

AWS compute instance type selection is available using the CDP CLI

This release of CDW supports selection of a single AWS compute instance type and additional fallback instance types. From the CDP CLI, you can select the following options:

- computeInstanceTypes

  An array of AWS compute instance types from which you can select a single instance type that the environment of your Virtual Warehouse will use.
- additionalInstanceTypes

  An array of additional AWS instance types for fallback in order of priority to fail over in the event the compute instance type is unavailable. You can specify one of the additional instance types as the default if you do not configure a compute instance type.

When selecting an instance type, consider your computing, memory, networking, and storage needs. To fetch a string of available instance types, use the following CLI command:

```
cdp dw describe-allowed-instance-types
```

Example output:

```
{
     "aws": {
     "default": [
     "r5d.4xlarge"
     ],
     "allowed": [
     "r5d.4xlarge",
     ...
```

Specify the instance types in aws-options of the create-cluster command. For example:

```
cdp \
     --profile ${PROFILE} \
     dw \
     create-cluster \
     --environment-crn ${AWS_CB_ENV_CRN} \
     --use-private-load-balancer \
     --aws-options computeInstanceTypes=r5dn.4xlarge,additionalInstanceTypes
=r5d.4xlarge,r5dn.4xlarge
```

Use list-clusters and describe-cluster to see cluster settings. CLI documentation link.

Enable CloudWatch Logs

If you use Amazon CloudWatch, you can enable CloudWatch logs when you activate an environment from CDW or edit environment details. Add required permissions to your IAM policy, and then access CloudWatch logs from AWS.

Nodes dashboards added to Grafana

New nodes dashboards show metrics for the AWS/Azure Virtual Machines that host the Kubernetes cluster.

Cloudera Data Warehouse audit events

You can retrieve the following audit events using the CDP CLI that occur in CDW:

CreateEnvironment	CloneDbCatalog	UpgradeHiveVirtualWarehouse
DeleteEnvironment	UpgradeDbCatalog	CreateImpalaVirtualWarehouse
UpdateEnvironment	CreateHiveVirtualWarehouse	DeleteImpalaVirtualWarehouse
UpgradeEnvironment	DeleteHiveVirtualWarehouse	StartImpalaVirtualWarehouse

CreateDbCatalog	StartHiveVirtualWarehouse	StopImpalaVirtualWarehouse
DeleteDbCatalog	StopHiveVirtualWarehouse	UpdateImpalaVirtualWarehouse
StartDbCatalog	UpdateHiveVirtualWarehouse	CloneImpalaVirtualWarehouse
StopDbCatalog	CloneHiveVirtualWarehouse	UpgradeImpalaVirtualWarehouse
UpdateDbCatalog		

For more information about experimental CLI commands for Cloudera Data Warehouse, go to Version Mapping. Click the CDP CLI Reference link for your CDW version. Scroll to Available Commands, and click dw.

Data Analytics Studio (DAS) is deprecated

DAS has been deprecated and is disabled by default. It will be removed in future releases. Cloudera encourages you to use Hue to run Hive LLAP workloads. If you need to use DAS, then you can enable it by setting the das.event-pipeline.enabled property to true in the Database Catalog configurations. For more information, see Enabling Data Analytics Studio in CDW Public Cloud.

Hue Query Processor scan frequency decreased to 5 minutes

The Hue Query Processor scans the event processor pipeline to retrieve the Hive query history and query details and display them on the **Job Browser** page. The scan frequency has been decreased from 2 milliseconds to 5 minutes to optimize resource utilization. As a result, you may notice a delay in viewing the query history and query details on the **Job Browser** page for queries that finish executing in less than 5 minutes. However, you can still view the query history from the Query history tab below the query editor.

September 15, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Support for Amazon Elastic Kubernetes Service 1.21 cluster

This release 1.4.3-b225 (released September 15, 2022) automatically uses and provisions Amazon Elastic Kubernetes Service (EKS) 1.21 when you activate an environment from CDW. In this release, upgrading a cluster from EKS 1.20 to 1.21 is not supported.

Alert notification configuration

You can now configure alert notifications that appear in user notification system. The notifications supplement alert information in charts on the Manage/Hive/Compaction Observability dashboard in Grafana.

Impala Virtual Warehouse catalog high availability

In earlier releases, the single instance limitation of the Impala Virtual Warehouse catalog made the catalog a single point of failure. In this release, you can optionally configure replication of the Impala catalog for high availability. One instance operates in active mode, the other in passive mode to provide failover. The passive instance serves as a backup and takes over if the active instance goes down.

General availability of Managed Storage Access

CDW capabilities for storing data for multiple tenants is now generally available (GA). You set up a managed storage warehouse for AWS or for Azure.

General availability of SSO to Virtual Warehouses

In this release, the capability to enable SSO (single sign-on) to your Virtual Warehouse from JDBC/ODBC clients is generally available (GA).

Hive to Iceberg table migration disallowed under certain conditions

If you attempt to migrate a Hive table having VARCHAR or CHAR columns to Iceberg, an exception will occur in this release. Migration under these conditions is not allowed in this release because incorrect query results can occur. To work around this limitation, change VARCHAR and CHAR columns to string columns, and then migrate the table to Iceberg.

Impala Autoscaler Web UI

This release includes an addition to the Impala Web UI for debugging the Impala Virtual Warehouse. The new UI provides insight into Autoscaler operations, accessing log messages, and resetting the log level.

Additions to IAM policy, standard and restricted, permissions

In this release, if you want to use Amazon CloudWatch, add logging permissions: to your IAM policy: "logs:CreateLogGroup","logs:CreateLogStream","logs:DescribeLogStreams", and "logs:PutLogEvents". The arn:aws:iam::aws:policy/CloudWatchAgentAdminPolicy must also be added to the restricted permissions.

Using CloudWatch is not required. Add logging permissions for CloudWatch only if you use this Amazon service. For more information about adding the CloudWatch logging permissions, see the following documentation:

- Minimum set of IAM permissions required for reduced permissions mode
- AWS restricted permissions policy

CREATE TABLE LIKE PARQUET FILE feature in Unified Analytics

You can base a new table on a schema in a Parquet file using the CREATE TABLE LIKE PARQUET FILE statement.

August 4, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Azure AKS 1.22 provisioning

In this release 1.4.2-b118 (released August 4, 2022), when you activate an environment, CDW automatically provisions AKS 1.22.

Do not upgrade your AKS cluster to 1.22 in this release using the Azure CLI. Doing so can cause the cluster to become unusable and can cause downtime. For more information about upgrading, see Upgrading an Azure Kubernetes Service cluster for CDW.

Azure AKS managed identity support

You can now specify a user-assigned, managed identity for the AKS cluster when you activate the Azure environment.

Support for fine-grained access using minimum permissions in Azure environments

You can configure minimum permissions to govern access control between CDW, Azure resources, and the Azure storage account.

Azure Kubernetes Service deployment configuration options

Changes in configuration properties simplify configuring a DNS zone for AKS.

Cluster subdomain definition improvement

In release version 1.4.2-b118 (released August 4, 2022), if you have the CDW_CUSTOM_CLUSTER_ID entitlement, you can define a cluster subdomain to retain your JDBC URLs. When you activate an environment, define your old subdomain using the following format:

env-xxx.dw

Defining the old subdomain in this way retains your old Virtual Warehouse names in the cluster. During environment activation, your old URLS continue to work, including JDBC URLS, Hive and Impala Virtual Warehouse URLS, Grafana URLS, and other URLS.

The new way of defining your old subdomain replaces the behavior in effect for Versions 2021.0.1.-b10 (released August 27, 2021) through 1.4.1-b86 (released June, 22, 2022). For information about the old way of defining a cluster domain, and the JDBC URL changes you had to made, see the August 27, 2021 release notes.

Simplification of Azure activation UI from CDW

Azure documentation for Data Warehouse has been updated to reflect improvements in the UI:

- Creating and registering the Azure environment
- Configuring a custom private DNS Zone
- Enabling kubenet networking
- Activating private AKS
- Custom tags

Apache Iceberg GA in CDW

The following new features are generally available (GA) in Iceberg V1 in CDW:

- Iceberg table rollback from Hive
- From Hive, create an Iceberg table with a metadata location
- Expire snapshots from Hive

Impala BI features were released on a GA basis already.

Impala enhancements

This release of Cloudera Data Warehouse includes the following new Impala features:

- Printing query results in vertical format

  Impala-shell now includes a new command option '-E' or '--vertical' to support printing of query results in vertical format.
- Resolving ORC columns by names

  Before this release, Impala resolved ORC columns by index. In this release, a query option ORC_SCHEMA_RESOLUTION is added to support resolving ORC columns by names.
- Retrieving the data file name

  Impala now supports including a virtual column in a standard SELECT statement select INPUT__FILE__NAME from <tablename> to retrieve the name of the data file that stores the actual row in a table.
- Consolidating the ranger audit logs for the same table

  Impala now consolidates the Ranger audit log entries of column accesses granted by the same policy for columns in the same table, after all the requests for accessing an object are processed.
- BYTES function support

  Impala now supports the BYTES() function. This function returns the number of bytes contained in a byte string.
- Configurable socket timeout for http transport

  Impala supports configuring socket timeout using the new impala-shell config option --http_socket_timeout_s. When you set a reasonable timeout, an impala-shell client can retry in case of connection issues.
- UTF-8 mode support

  Some Impala STRING types now support UTF-8 aware behavior to ensure consistent results for non-ASCII characters in the string in both Hive and Impala.

June 22, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Apache Iceberg GA in CDW

Apache Iceberg is now generally available (GA) in key data services on CDP, including Cloudera Data Warehouse. Iceberg integration with Cloudera Data Platform (CDP) enhances the Lakehouse architecture by extending multifunction analytics to petabyte scale for multi-cloud and hybrid use-cases. From Impala, you use Apache Iceberg features in CDW, which include time travel, create table as select, and schema and partition evolution. To access these features, create a new Virtual Warehouse or upgrade an existing one.

Cross account role permissions policy

For security reasons, if you do not want to provide PutRolePolicy permission in your cross account role, which would be used later to add an inline policy to the Node instance role, you can attach a managed policy ARN to a node role. This managed policy provides cross account role permissions. To access this feature, create a new NodeInstanceRole manually, and provide the ARN during activation of the environment from CDW.

Zipping unnest on arrays from Views

As part of this release, you can use zipping unnest functionality on arrays from Views. Before this release, this zipping functionality worked for arrays only in Tables but did not support Views as a source. For more information about using this zipping unnest functionality, see Zipping unnest on arrays from Views.

Updates to the Hive and Impala language reference manuals in Hue

Hive and Impala language reference manuals that are built into the Hue web interface have been updated to include the latest syntax and user-defined functions (UDF). Help on Iceberg concepts and syntax is available for Impala. Improvements have also been made to the query autocomplete predictions and syntax checker to support these changes.

May 10, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Compaction txn heartbeating after Worker timeout

This hot fix resolved problems with Database Catalogs timeouts.

Inhibit creation of JCEKS secret during updates and upgrades.

This hot fix resolved problems with Ranger policies after upgrading to 2022.0.7.0-80 (released April 27, 2022).

Recognition of regional cloud provider

This hot fix resolved issues with the DWX server recognizing the regional cloud provider and using environment-specific chartmap buckets.

April 27, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Metering fix

In this release, an important fix related to Cloudera Data Warehouse (CDW) metering has been added. This fix accurately communicates the status of your Virtual Warehouses to the Cloudera control plane, necessary for proper functioning of CDW. To include this fix, upgrade your Virtual Warehouses. Cloudera recommends performing this upgrade as soon as possible.

Unified Analytics

This is the GA release of Unified Analytics that brings SQL equivalency without syntax changes to CDW SQL engines. Unified Analytics brings significant optimization equivalency to these engines, unifying common techniques such as subquery processing, join ordering, and materialized views. Unified Analytics is available as a CDW Kubernetes service in AWS. Unified Analytics documentation includes features, limitations, and how to use Unified Analytics. To take advantage of this feature, create a new Impala Virtual Warehouse and enable the Unified Analytics option, or simply create a new Hive Virtual Warehouse. For more information, see this community blog.

Ability to enable a private CDW environment in Azure Kubernetes Service (AKS)

You can now activate CDW clusters in private mode which enables you to create all cloud resources without the need of public IP addresses. For more information, see Enabling a private CDW environment in Azure Kubernetes Service.

Downloadable link for Beeline CLI tarball

You can now download the Beeline CLI tarball from CDW and use the Beeline client to connect to a Hive Virtual Warehouse and run queries. The archive file contains all the dependent JARs and libraries that are required to run the Beeline script. Upgrade your existing Virtual Warehouse to use this feature.

Support for uploading non-UDF JARs

As part of this release, CDW enables administrators to upload additional non-UDF JARs to the Hive classpath that might be required to support dependency JARs, third-party Serde, or any Hive extensions.

Enhanced complex types support for Impala

This release allows you to use:

- Array in SELECT list
- UNNEST function for arrays in SELECT list

See Querying arrays for more information.

Ability to run Hive LLAP workloads using Hue without the “CDW_HUE_LLAP” entitlement

You can now use Hue to run Hive workloads in CDW without enabling the “CDW_HUE_LLAP” entitlement. Hue packs the combined features of Data Analytics Studio (DAS) and Hue in this new version. If you do not have the “CDW_HUE_LLAP” entitlement enabled in your environment, then you can get Hue by creating a new Hive Virtual Warehouse.

Ability to select the Hue image version

CDW allows you to select a Hue Runtime version from the Hue Image Version drop-down menu while creating a Virtual Warehouse. The compatible Hue image version is automatically selected based on the Hive or Impala image version that you select. If there is no corresponding Hue version for the selected Hive or Impala version, then the latest Hue version is automatically selected. For example, if you select Hive or Impala version 2022.0.7.0-70, then 2022.0.7.0-70 is selected as the Hue image version.

You can also check the Hue image version from the **Virtual Warehouse Details** page.

Hue Query Processor runs at the Database Catalog level

The Hue Query Processor indexes and retrieves Hive query history by reading query event data from a Hadoop Compatible File System (HCFS), such as S3, and writing them into a relational database, such as PostgreSQL. Previously, the Hue Query Processor process ran at the Virtual Warehouse level. Therefore, there was one Query Processor pod running for each Virtual Warehouse associated with a Database Catalog. Now, the Query Processor pod runs at the Database Catalog level. This helps in reducing the read cost in the cloud.

A new tab called “Hue query processor” has been added under the Database Catalog Edit CONFIGURATIONS section. The “hue-query-processor.json” and “hue-event-processor.json” files are no longer available under the Virtual Warehouse Edit CONFIGURATIONS Hue Configuration files drop-down menu.

If you are on an older version of Virtual Warehouse, then you must create a new Database Catalog and Virtual Warehouse to activate this feature.

To tune the data refresh rate, see Configuring the Hue Query Processor scan frequency.

Ability to grant fine-grained access to S3 buckets from the S3 File Browser in Hue (Preview)

For better security and ease of use for users, you can create per-user home directories within your S3 bucket and grant fine-grained access to these user directories from the S3 File Browser in Hue. To enable fine-grained access from the Hue S3 File Browser, you must enable Ranger Authorization Service (RAZ) when you register your AWS environment with CDP, and add the path to your S3 bucket in the hue-safety-valve field in CDW.

Note: You need to contact Cloudera to have this feature enabled.

Ability to grant fine-grained access to ADLS Gen2 containers from the ABFS File Browser in Hue (Preview)

For better security and ease of use for users, you can create per-user home directories within your ADLS Gen2 container and grant fine-grained access to these user directories from the ABFS File Browser in Hue. To enable fine-grained access from the Hue ABFS File Browser, you must enable RAZ when you register your Azure environment with CDP, and add the path to your ABFS container in the hue-safety-valve field in CDW.

Note: You need to contact Cloudera to have this feature enabled.

Performance and security enhancements in Hue

Python 2 has reached the end of life and is no longer supported. Hue in CDW Public Cloud now uses Python 3 which makes use of critical bug fixes and Common Vulnerabilities and Exposures (CVE) fixes for many third-party software dependencies.

Following changes have been made in the Hue codebase in this release of CDW Public Cloud:

- The Hue container now runs in the UBI8 image, which is upgraded from UBI7.
- Python libraries have been upgraded from Python 2.7 to Python 3.8.
- The Django server has been upgraded from version 1.11.29 to 3.2.12.
- Python modules such as django-auth-ldap, django-axes, djangorestframework-simplejwt, Mako, Markdown, pyth on-ldap, django-babel, django-mako, django-cors-headers, djangorestframework, eventlet, sqlparse, and so on have also been upgraded.
- Hue now uses Gunicorn as a front-end server. Previously, Hue used the CherryPy server.

These upgrades bring in significant performance improvement and stability in query execution, uploading, and importing files to S3 or ABFS. Operating System, Python version, and Python module upgrades have resulted in a stable environment and fixed more than 800 security vulnerabilities.

Also, the UBI8 image uses the latest Apache HTTP Server (httpd). This no longer requires you to limit the fully qualified domain name of your Virtual Warehouse to 64 characters.

Hue images now run as non-root users.

Selecting an availability zone when creating a Virtual Warehouse

In AWS environments, you can now accept the default availability zone, or select your preferred zone when creating a Virtual Warehouse. To use this feature, create a new Virtual Warehouse.

AWS restricted permissions policy

As Administrator, to restrict access to your Cloudera Data Warehouse service, your IAM role must include the AWS restricted policy. You must add the policy to your IAM role before you activate the environment in Cloudera Data Warehouse to make the service available to users.

March 8, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Apache Iceberg technical preview in Cloudera Data Warehouse

Cloudera Data Warehouse (CDW) supports Apache Iceberg as a technical preview. Iceberg is a cloud-native, open table format for organizing petabyte-scale analytic datasets on a file system or object store. In CDW, you can read and write Iceberg tables using Hive or Impala SQL engines. Create a new Virtual Warehouse, or upgrade your existing one, to use Iceberg in CDW.

Automating metadata invalidation after compaction

After compaction of ACID tables, metadata in Impala coordinator local caches might be stale. Invalidation refreshes the metadata and prevents possible query failure. This release introduces automatic metadata invalidation, which you can configure in the Impala Virtual Warehouse. You must upgrade your Database Catalog to get this feature.

Cloudera Data Warehouse availability in Asia Pacific region

Cloudera Data Warehouse (CDW) is now supported in the ap-1 (Australia) regional Control Plane. To use CDW in this regional Control Planes, your CDP administrator must create a new environment. For the list of all supported services for all supported Control Plane regions, see CDP Control Plane regions.

Data Visualization integration in Cloudera Data Warehouse

The status of Data Visualization integration in CDW is GA in this release. CDW integrates Data Visualization for building graphic representations of data, dashboards, and visual applications based on CDW data, or other data sources you connect to. Authorized users can explore data using graphics, such as pie charts and histograms and collaborate using dashboards. BI analysts who can access your environment can use these features. In a future release, the Enable Data Visualization option in the New Virtual Warehouse dialog will be deprecated.

Impala coordinator shutdown

To save cloud costs, you can configure Impala coordinators to automatically shutdown during idle periods. In this release, the coordinator shutdown feature status has changed to GA. The feature is available when you create an Impala Virtual Warehouse.

Impala Virtual Warehouse High Availability

A single Impala coordinator might not handle the number of concurrent queries you want to run or provide the memory your queries require. You can configure multiple active-active coordinators to resolve or mitigate these problems. To use this feature, create a new Virtual Warehouse.

Impala late materialization of columns

The Data Warehouse Runtime release of Impala introduces late materialization, which optimizes certain queries on Parquet tables by limiting table scanning. Only relevant data is materialized to improve query response.

Istio version update

In this release, new environments you create support Istio 1.11.2. In new environments, you must select the latest Cloudera Data Warehouse image version 2021.0.6.96 when you create a Database Catalog and Virtual Warehouse, as described in Known issues.

Log4j Update

Cloudera has released a new version of CDW that upgrades the embedded Log4j version to 2.17.1.

January 19, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Accessing Impala using the new Impala shell client

You need to provide commands to your client users for installing and launching the Impala shell to connect to your particular Impala Virtual Warehouse version. You can use new options in the UI for accessing Impala from the Impala shell client. You can obtain the commands to download the new Impala shell client from the CDW UI under Impala Virtual Warehouse options. Create a new Virtual Warehouse to get this feature.

Support for regional control planes for eu-1 (Germany)

The Admin must create a new environment to support these regional control planes.

December 22, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

CDW upgrade of embedded Log4j version

Cloudera has released a new version of CDW which upgrades the embedded Log4j version to 2.16. This provides a permanent fix for CVE-2021-44228.

To use this version you must upgrade your Database Catalog(s) and Virtual Warehouse(s) to the latest version, which is 2021.0.4.1-3. Additionally, you can create new Virtual Warehouses using the latest version, and those will also have the fix for this vulnerability.

If you had previously applied the configurations shared in CVE-2021-44228 remediation for CDW, then those are no longer required. Please reach out to Cloudera Support if you have questions.

November 20, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Supported AWS regions now include EU (Milan)

Previously in a Technical Preview state, support for the EU (Milan) region in AWS is in a GA (general availability) state. For more information, see Supported AWS Regions. To use this feature, create a new environment and activate it in CDW.

SSM and CBO federation

This release adds support for interoperation between the cost-based optimizer (CBO) in Cloudera Data Warehouse and the AWS Systems Manager (SSM) service.

Ability to set idle session timeout for Hue

CDW allows you to configure the idle session timeout value in Hue to automatically log out users from the Hue web interface after a period of inactivity. For more information, see Configuring idle session timeout for Hue.

CDW Impala/Hive cross-region connection

This release adds the capability to create external tables on data in S3 buckets that are hosted in a different region from the one where CDP is running. Requests to these buckets can use a proxy. For example, GDPR dictates that you cannot move data from your EMEA region, but your customers want to query the data from CDP in us-east-1. In

compliance with GDPR, you can create an external table based on data in a bucket in an EMEA region when you are running CDP in the us-east-1 region.

Hue usability improvements

As Admin in a Ranger Authorization Service (RAZ) environment, you can set the AWS S3 or ADLS path to a user directory in Hue. As the Hue user, you always land in your directory and do not have to repeatedly change URLs.

Hive backports

This release fixes issues with custom compaction queue settings and vectorized built-in functions having compound expressions in PARTITION BY or ORDER BY clause.

October 29, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Hive compaction observability

An alert notification about compaction status, issues, and suggested actions appear in the overview of your Database Catalog, which leads to an informative message about the issue. The following list describes a few of more than 25 notifications:

- Oldest initiated compaction passed threshold
- Large number of compaction failures
- More than one host is initiating compaction

Create an environment in this release to pick up this feature. For more information about compaction observability, see Compaction Observability.

Choosing a CDW version for a Database Catalog or Virtual Warehouse

When creating a Database Catalog or Virtual Warehouse, you can choose the latest version of CDW or an earlier version. Version mapping lists the CDW versions from 2021.0.0-b33 released on August 9, 2021 to the latest release. You can ensure backward compatibility for your scripts, for example, by running your jobs on the same version of CDW. The ability to choose a version of Database Catalog or Virtual Warehouse is available under entitlement CDW_VERSIONED_DEPLOY.

AWS EKS Kubernetes version upgrade

The CDW application uses Kubernetes (K8S) clusters to deploy and manage Hive and Impala in the cloud. Kubernetes versions are updated every 3 months on average. When the version is updated, minor versions are deprecated. Amazon Elastic Kubernetes Service (EKS) is updating to Kubernetes version 1.20 and is ending support for version 1.17.

AWS environments you activate using this release of Cloudera Data Warehouse, and later, will use version 1.20. To avoid compatibility issues between CDW and AWS resources, you must update the version of Kubernetes that supports your existing CDW clusters to version 1.20.

Important: Check to make sure your AWS environment has been migrated from Helm 2 to Helm 3 before you begin upgrading the Kubernetes version.

If a MIGRATE icon appears in the upper right corner of the environment tile, your AWS environment has not been migrated from Helm 2 to Helm 3. Update the Helm package manager for your environment before you attempt to upgrade its Kubernetes version.

For information about upgrading the AWS EKS Kubernetes version, see Upgrade CDW for EKS upgrade.

Hue supports Hive Hybrid Procedural SQL in CDW

You can run Hive Hybrid Procedural SQL (HPL/SQL) using the Hue query editor in CDW. For more information, see How to run a stored procedure from Hue in Cloudera Data Warehouse.

October 22, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Prevention of ineffective scaling in Impala Warehouses

CDW takes advantage of IMPALA-10244 to detect the case of queries being queued because of a resource bottleneck on the Impala Coordinator. In this case, adding more executor groups does not result in the ability to run more queries, and so, the warehouse does not scale up.

September 28, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Retaining PostgreSQL backups in the cluster

When you create a Cloudera Data Warehouse cluster using the CDP CLI create-cluster command, any PostgreSQL backup retention period you set on your Cloud provider side, is observed by CDP. For example, if you configure backupRetentionDays in Azure or BackupRetentionPeriod in AWS, and then create a cluster using the CDP CLI, the cluster will retain the PostgreSQL backups according to your configuration.

August 27, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces no new changes.

CDW Public Cloud endpoints change

The format of Hive, Impala, Hue, and Data Analytics Studio (DAS) endpoints have changed in the Cloudera Data Warehouse (CDW) Public Cloud release dated August 27, 2021:

- FROM:

```
`*.<env-id>.dw.<tenant-id>.<workload-domain-name>`
```

- TO:

```
`*.dw-<env-name>.<tenant-id>.<workload-domain-name>`
```

In the August 27, 2021 release, the format was changed from using the environment ID (env-id) to using the DW environment name (env-name). Consequently, the UI was simplified. If you had the CDW_CUSTOM_CLUSTER_ID entitlement in earlier releases, you might notice the Custom Cluster ID field in Activation Settings has been removed. This field is no longer necessary.

The new endpoint formatting affects the Hive and Impala JDBC URLs, the Impala shell URL, Hue URLs for Hive and Impala, and DAS URLs, as well as the Impala JDBC with coordinator shutdown URL.

Examples of endpoint changes

In these examples, the following values are used for the variables in the endpoint format above:

Variable: env-name

Example value: env1

Description: The name you provided when registering the environment in Management Console.

Variable: env-id

Example value: env-hfwbgd

Description: The random string for the environment ID that gets generated at the time of activation from CDW.

Variable: tenant-id

Example value: 'xcu2-9999'

Description: Your tenant ID.

Variable: workload-domain-name

Example value: dev.abcd.work

Description: The domain name.

Hive Virtual Warehouse name

Example value: hive1

Impala Virtual Warehouse name

Example value: impl1

Table 1:

Endpoint type	New URL (CDW August 27, 2021)	Old URL (CDW August 9, 2021 and earlier)
Hive JDBC	jdbc:hive2://hs2-hive1.dw-env1.xcu2-9999.dev.abcd.work/default;transportMode=http;httpPath=cliservice;socketTimeout=60;ssl=true;twoWayAuthenticationMode=true;trustStoreType=JKS;	jdbc:hive2://hs2-hive1.env-hfwbgd.dw.xcu2-9999.dev.abcd.work/default;transportMode=http;httpPath=cliservice;socketTimeout=6
Impala JDBC	jdbc:impala://coordinator-impl1.dw-env1.xcu2-9999.dev.abcd.work:443/default;AuthMech=3;transportMode=http;httpPath=cliservice;ssl=1;UID=workloadUsernameHere;PWD=workloadPasswordHere	jdbc:impala://coordinator-impl1.env-hfwbgd.dw.xcu2-9999.dev.abcd.work:443/default;AuthMech=3;transportMode=http;httpPath=cliservice;ssl=
Impala shell	impala-shell --protocol='hs2-http' --ssl -i 'coordinator-impl1.dw-env1.xcu2-9999.dev.abcd.work:443' -u <userid> –l	impala-shell --protocol='hs2-http' --ssl -i 'coordinator-impl1.env-hfwbgd.dw.xcu2-9999.dev.abcd.work:443' -u <userid> –l
Hue Hive	https://hue-hive1.dw-env1.xcu2-9999.dev.abcd.work/hue	https://hue-hive1.env-hfwbgd.dw.xcu2-9999.dev.abcd.work/hue
Hue Impala	https://hue-impl1.dw-env1.xcu2-9999.dev.abcd.work/hue	https://hue-impl1.env-hfwbgd.dw.xcu2-9999.dev.abcd.work/hue
Impala JDBC with coordinator shutdown enabled	jdbc:impala://impala-proxy-impl1.dw-env1.xcu2-9999.dev.abcd.work:443/default;AuthMech=3;transportMode=http;httpPath=cliservice;ssl=1;UID=workloadUsernameHere;PWD=workloadPasswordHere	jdbc:impala://impala-proxy-impl1.env-hfwbgd.dw.xcu2-9999.dev.abcd.work:443/default;AuthMech=3;transportMode=http;httpPath=cliservice;ssl=
Data Analytics Studio (DAS)	https://hive1.dw-env1.xcu2-9999.dev.abcd.work/	https://hive1.env-hfwbgd.dw.xcu2-9999.dev.abcd.work/

August 9, 2021

There are no new features in this release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud. This release mainly changed the CDW version numbering scheme.

July 29, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Improvements in the CDP CLI

The CDP CLI commands for creating and deleting an environment, and for creating and deleting a Virtual Warehouse, include all options available in the UI.

Auto-shutdown of Impala coordinators for cost saving

When CDP does not need resources to handle Impala queries, coordinators shut down without user intervention.

July 13, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Impala Debug Web UIs are now available in CDW

You can now use the Impala debug web UIs in CDW, which map to equivalent debug web UIs in Cloudera Manager as follows:

CDW Debug Web UI	Cloudera Manager Equivalent
Impala Catalog Web UI	Catalog Server Web UI
Impala Coordinator Web UI	Impala Daemon Web UI
Impala StateStore Web UI	StateStore Web UI

For more information about this feature, see Use the Web UI to debug Impala Virtual Warehouses.

Auto-scaling demos have been disabled

Due to critical security issues discovered in the auto-scaling demo components, the demo has been disabled.

CDW UI improvements

Error messages and UI labels have been enhanced for readability and usefulness.

June 14, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

S3 buckets created by CDW on AWS environments are now encrypted by default with SSE

When you use the latest cross-account role from Management Console to register an environment, S3 buckets created by CDW during AWS environment activation, are now encrypted by default with SSE-S3. This provides server-side object encryption (SSE), which protects data at rest. Before this enhancement, S3 buckets were created with only object-level encryption for every object in the bucket during environment activation.

Important: This only applies to S3 buckets created for managed tables and for external tables, which are actually created by CDW during environment activation. You must ensure that S3 buckets you add to CDW after environment activation are encrypted to meet your organization's security requirements.

Impala Virtual Warehouses can now be recovered after going into an error state

Impala resources are not deleted when the Virtual Warehouse goes into an error state. Consequently, if an Impala Virtual Warehouse goes into an error state, you can now use the Edit option to recover the Virtual Warehouse with its original configurations.

Impala Virtual Warehouses now use node memory more efficiently

Impala coordinators now can use ~13GiB more of available memory on a node to better support concurrent query execution.

June 1, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

S3 Guard disabled for custom Database Catalogs and default Database Catalogs on AWS environments

During new AWS environment activation in CDW:

- If you have not specified a DynamoDB table name during environment registration in Management Console, the S3 Guard feature is disabled for all Database Catalogs and Virtual Warehouses.
- If you have specified a DynamoDB table name during environment registration in Management Console, the S3 Guard feature is enabled for default Database Catalogs and Virtual Warehouses. However, it is disabled for non-default (custom) Database Catalogs.

For all environments that were activated before the June 1, 2021 release, S3 Guard is enabled.

May 21, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

Improved startup times for Impala Virtual Warehouses

The auto-scaler for Impala Virtual Warehouse has been updated to be more efficient thereby improving the startup times for Impala Virtual Warehouses.

CDW UI left navigation menu improvement

The left navigation menu for the CDW UI has been upgraded to be consistent with other CDP experiences.

AWS environments: Helm 2 to Helm 3 migration fixes

Various fixes applied to the Helm 2 to Helm 3 migration process for AWS environments.

May 3, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces these changes.

CDE Airflow integration to schedule ETL jobs for Hive Virtual Warehouses

In this release, CDE Airflow, based on Apache Airflow, which enables you to configure, schedule, and monitor workflows, can now be used to schedule ETL jobs that target Hive Virtual Warehouses running in CDW. For more information, see CDE Airflow ETL workloads in Hive Virtual Warehouses.

AWS environments updates

Helm 2 to Helm 3 migration

Helm is a package manager for the Kubernetes platform. CDW, which runs on Kubernetes, has migrated to Helm version 3 to take advantage of the improved security and upgrading capabilities offered by Helm 3. Any AWS environments that were created prior to this platform migration still run on Helm version 2. Although you can continue to use the Database Catalogs and Virtual Warehouses running on AWS environments that use Helm 2, there are significant limitations. Cloudera recommends that you migrate your AWS environments to Helm 3 as soon as you can.

AWS EKS Kubernetes version upgrade

The CDW application uses Kubernetes (K8S) clusters to deploy and manage Hive, Impala, and Real-time Event Store Analytics (Druid) in the cloud. Kubernetes versions are updated every 3 months on average. When the version is updated, minor versions are deprecated. Amazon Elastic Kubernetes Service (EKS) is updating to Kubernetes version 1.16 and is ending support for version 1.15 on May 2, 2021. To avoid compatibility issues between CDW and AWS resources, the version of Kubernetes that supports your CDW clusters must be updated.

Important: Check to make sure your AWS environment has been migrated from Helm 2 to Helm 3 before you begin upgrading the Kubernetes version.

For more information about upgrading AWS environments for AWS EKS Kubernetes cluster updates, see Upgrade CDW for EKS Kubernetes version upgrade.

Azure environments

Ability to set scratch space limit for spilling Impala queries on Azure environments

Memory-intensive SQL operations need temporary disk space when Impala is close to exceeding its memory limit on a particular host. You can configure the scratch space between 300 GiB and 16 TiB per node while creating an Impala Virtual Warehouse in the Cloudera Data Warehouse (CDW) service. For details, see Setting scratch space limit for spilling Impala queries in Azure environments.

April 9, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the improvements that are described in this topic.

Virtual Warehouse improvement

Ability to launch and use the Hue app while the Virtual Warehouse is starting up

When you restart a Virtual Warehouse, it takes a few minutes to start and change to the "Running' state. In the older CDW version, you had to wait to launch the Hue app until the Virtual Warehouse is up and running. Now you can open Hue and run queries even when it is in the "Starting" state.

March 31, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the improvements that are described in this topic.

Ability to use the internal load balancer with Azure Kubernetes Service (AKS)

You can enable an internal (private) load balancer to distribute traffic within your virtual network (VNET). An internal load balancer makes a Kubernetes service accessible only to applications running in the same or peered Azure VNET as the Kubernetes cluster, or applications running in on-premises networks that are connected using Azure ExpressRoute circuits. For more information, see Enabling internal load balancer for AKS clusters.

Behavioral changes

CDPD-24214: Virtual Warehouse compaction issues resolved

Description: The compaction processes, which are performed by Hive Virtual Warehouses, were failing to clean up directories, which caused query performance issues.

Resolution: A fix has been added from Apache HIVE-24900, which resolves this issue.

March 23, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Non-transparent proxy support for AWS environments

CDW Public Cloud now supports AWS environments that use non-transparent proxies. Using non-transparent proxies permits you to pass connection or security information along with the connection request that is sent by clients. Some organizations' security policies require the use of non-transparent proxies and now CDW can support that requirement. For more information about this feature, see Use a non-transparent proxy with CDW.

March 12, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Support for custom tags on AWS and Azure environments

You can specify tenant-level custom tags to identify and monitor resources within the CDW service for AWS and Azure environments at the time of environment creation. See Custom tags in Azure environments for the list of

resources that you can tag in Azure environments. If you are using AWS, you can also tag Virtual Warehouses using custom Virtual Warehouse-level tags. For a complete list of resources that you can tag in AWS environments, see Custom tags in AWS environments.

For information on applying custom tags, see Defining custom tags in the Management Console documentation.

February 23, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the following new features and behavioral changes.

Reduced permissions mode for AWS environments

This feature enables you to activate an AWS environment for CDW with fewer than half of the standard required IAM permissions on your AWS cross-account IAM role. This is useful if your InfoSec policies do not permit you to use the standard set of required permissions.

When activating an AWS environment in CDW, if the system detects that your account does not have the standard set of required IAM permissions, it invokes this feature automatically. Then you must manually create the cloud resources in your AWS account and tag them appropriately instead of having CDW do this for you. To assist you, CDW pre-populates the required CloudFormation template for you within the AWS Console, but you must perform the manual steps to create the stack. When you are finished using the stack, you must manually delete it in the AWS Console. For more information about this feature, see Reduced permissions mode for AWS environments.

February 10, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the following new features and behavioral changes.

Virtual Warehouse upgrade fixes

Multiple fixes were made to improve the upgrade process for Virtual Warehouses.

January 29, 2021

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Provisioning performance improvements for Virtual Warehouses

The provisioning time for Hive and Impala Virtual Warehouses has been significantly reduced for many cases. Generally, provisioning time has been reduced from 1.5 to 4 times over the previous time required for Virtual Warehouse provisioning.

December 21, 2020

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Use Data Visualization with Virtual Warehouses

You can now use CDP Data Visualization, a tool that enables you to explore data by using visual objects, with CDW Virtual Warehouses.

Start and stop Impala Virtual Warehouses

In this release, you can now explicitly start and stop Impala Virtual Warehouses by clicking the start or stop icon in the upper right corner of the Virtual Warehouse tile. In the Virtual Warehouse details page, you can use this feature by clicking the Suspend or Start options in the Actions menu. Use this feature to better control costs and resource usage in your cloud environment.

Note: This feature is only available on newly created Impala Virtual Warehouses.

Azure environments: Use DAS and Hue to create new custom functions

HiveServer2 localizes and packages the User-Defined Functions (UDF) jar files into Azure Blob storage. You can use DAS and Hue to create new custom functions using the external jar file. To create user-defined functions, see Using functions in the Hive documentation.

November 3, 2020

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Impala Virtual Warehouse improvements

The following improvements to Impala Virtual Warehouses have been added in this release:

- High availability improvements, including fault-tolerance and multi-threading improvements.
- Full support for reading tables based on the ORC file format. This improves collaboration across the data lifecycle. Now Hive and Impala Virtual Warehouses can use the same data format and the SQL engine can be selected to optimize the workload without restrictions tied to the file format.

- Ability to stop, start, and suspend Impala Virtual Warehouses:

Figure 1: Starting an Impala Virtual Warehouse

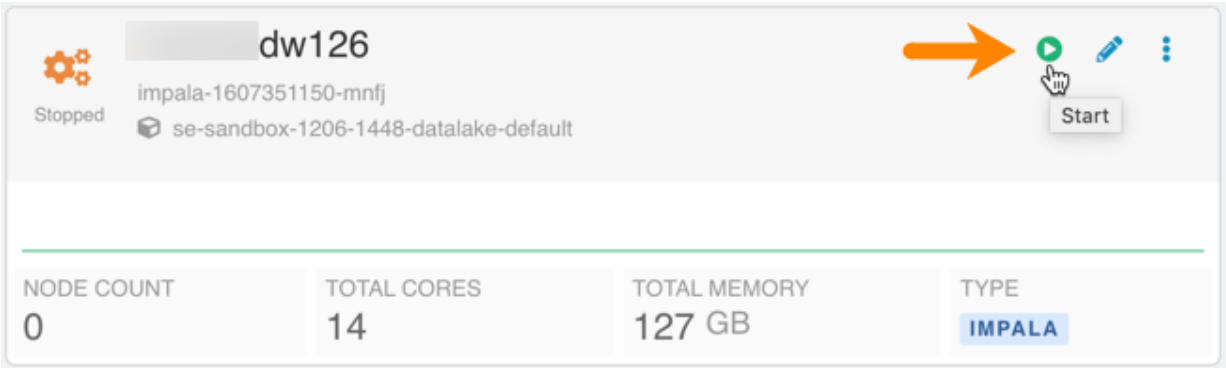

Figure 2: Stopping an Impala Virtual Warehouse

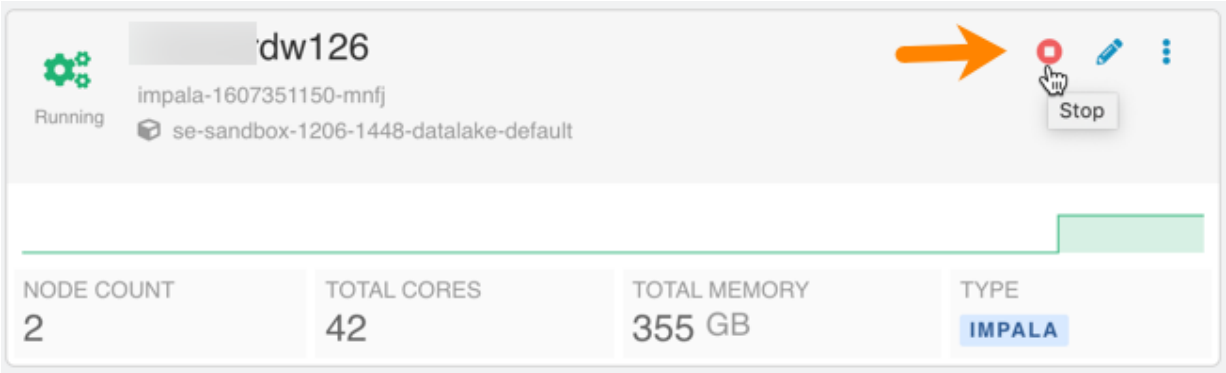

Figure 3: Suspending an Impala Virtual Warehouse

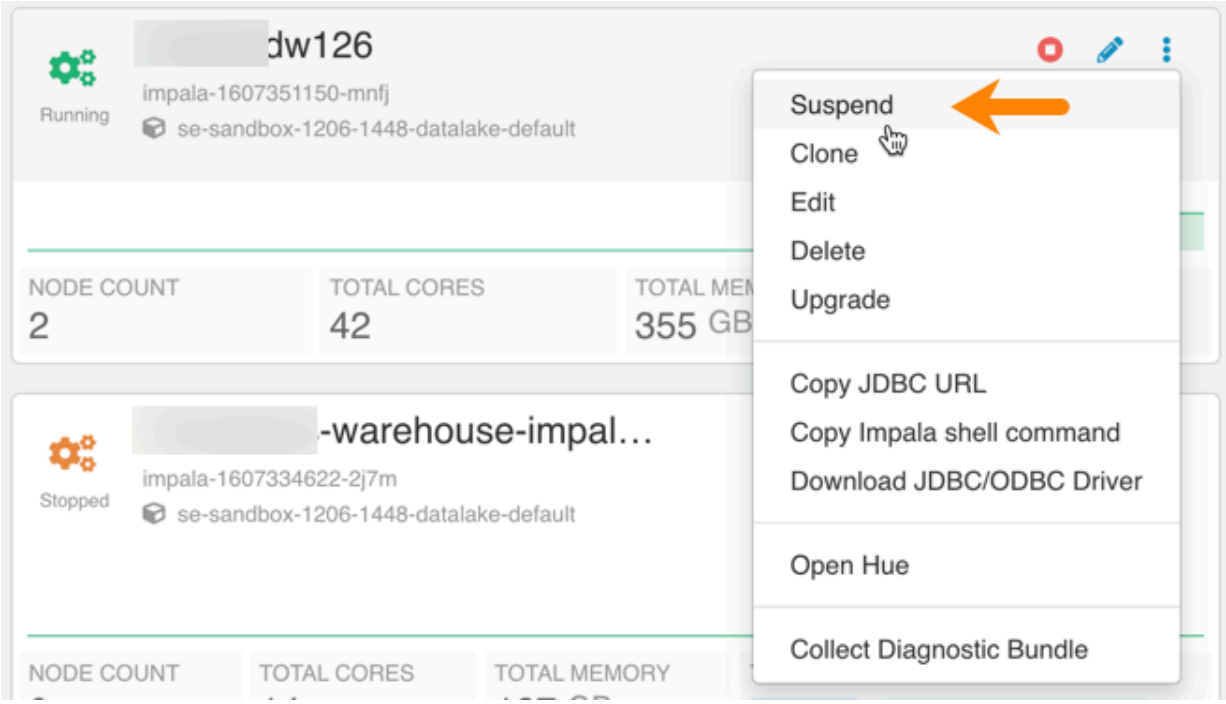

Hive Virtual Warehouse improvements

- Increased stability for Hive ACID and Compaction.
- Performance improvements:
  - For short-running ad-hoc queries by decreasing task startup latency.
  - For late projections by delay joins.
  - For increased optimization of 'shared work.'

Improved security with machine user implementation

When you activate a new environment in Cloudera Data Warehouse (CDW), a machine user is created in the CDP User Management System (UMS) and is synchronized with the FreeIPA Management System (FMS). The FreeIPA machine user is used by most of the services such as Hue, Impala, VizApp, etc. instead of the FreeIPA LDAP admin user to communicate with the services within the Data Lake (for example, Ranger) that is associated with the CDP environment. As a security measure, Cloudera recommends upgrading the CDW environment, Database Catalogs, and the Virtual Warehouses to the latest CDW release.

To upgrade the CDW environment, contact Cloudera Support.

To upgrade the Database Catalog and the Virtual Warehouse, see Upgrading Database Catalogs and Virtual Warehouses in Cloudera Data Warehouse Public Cloud.

Note: Before you upgrade all the Database Catalogs and Virtual Warehouses at the same time, upgrade any one Database Catalog or Virtual Warehouse and let the upgrade process complete. For example, if you have 2 Database Catalogs and 3 Virtual Warehouses in your CDW environment, then complete upgrading any one of the five components. After one of the components has successfully upgraded, you can upgrade the other four in parallel.

When you delete a CDW environment, the machine user is deleted from the UMS and the FMS.

Azure environments

- Executor nodes for Virtual Warehouses can now be scaled down to zero nodes, thereby using zero Azure resources for idle workloads, saving costs.

September 10, 2020

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Storage account consolidation for Azure environments

When using an Azure environment, the CDW service no longer creates storage separate from Data Lake storage. This change provides the following benefits:

- Reduced permissions required for the Azure app that is used to register the environment in Management Console:
  - The Storage Blob Contributor role is no longer needed for the storage account.
  - The app used to register the environment with Management Console, can now use the Contributor role instead of the Owner role.
- Reduced management overhead because there are fewer places to look for data and logs that are stored by the CDP platform.
- Simplified custom storage configurations because there no longer is a separate CDW service storage account in addition to the Data Lake storage account.
- Simplified security configuration if additional security controls are needed on storage accounts.
- Log file data and external table data persist in the storage account even after the Virtual Warehouse is deactivated.

For more information, see Azure environments for CDW.

CDW service endpoint domain names standardized

The domain names of CDW service endpoints are now standardized with other CDP service endpoint domain names. The URL syntax for all CDW service endpoints is now:

```
<CDW-ENVIRONMENT-ID>.dw.<SHORT-ACCOUNT-ID>.cloudera.site
```

You can get the *<CDW-ENVIRONMENT-ID>* from the CDW service UI by clicking the edit icon on either the Data Catalog or the Virtual Warehouse tiles on the Overview page:

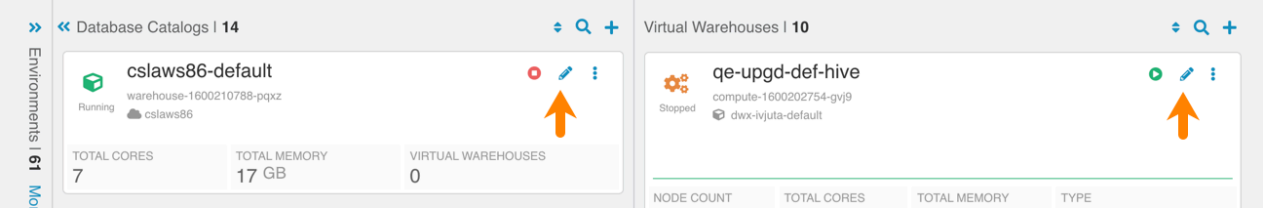

Then you can find the *<CDW-ENVIRONMENT-ID>* listed at the top of the Database Catalog or Virtual Warehouse details page:

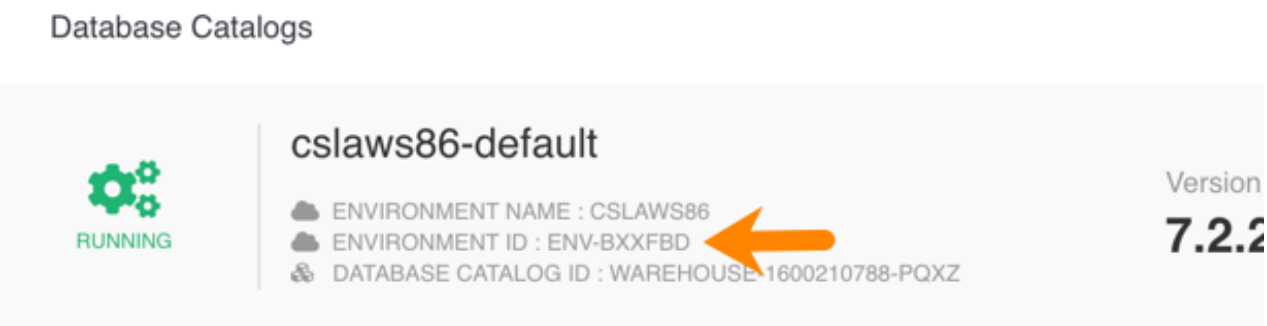

Your *<SHORT-ACCOUNT-ID>* is assigned to you during the account on-boarding process.

August 27, 2020

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Overlay networks now available for Azure environments

An overlay network is a software-defined layer of network abstraction that is used to run multiple separate, discrete virtualized network layers over the Azure virtual network. You use overlay networks to increase the number of IP addresses available to support nodes in your CDW cluster on Azure. For more information about using this feature in Azure environments and how to enable it, see Overlay networks for Azure environments.

August 14, 2020

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Stability and performance improvements

This release includes numerous stability and performance improvements, improvements to health checks, plus multiple bug fixes.

Ability to configure custom Hue properties

You can configure custom Hue properties that are not directly exposed through the CDW web UI by specifying them in a Hue configuration called hue-safety-valve for a Virtual Warehouse. For more details, see Configuring custom properties using safety valves.

Health check for the Hue load balancer

Unlike earlier, if the Hue server starts and the Hue load balancer has not started yet, Hue does not display an error message when you try to access the Hue endpoint (or the web UI). This is because Hue now checks the health of the load balancer as well as the Hue server before loading the Hue user interface.

July 30, 2020

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud introduces the new features and improvements that are described in this topic.

Support for Microsoft Azure environments now generally available

Support to run CDW service Virtual Warehouses on the Microsoft Azure cloud platform is now generally available. For more details about CDW support for Azure, see Azure support and Azure environments requirements checklist.

June 26, 2020

This release of the Cloudera Data Warehouse service on CDP Public Cloud introduces the following new features and improvements:

Query isolation for scan-heavy, data-intensive queries in Hive LLAP Virtual Warehouses

Hive Virtual Warehouses base auto-scaling on the total scan size of the query. HiveServer, which receives all incoming queries, has a query planner component. When the HiveServer query planner receives queries, it examines the total scan size of each query. That is, it looks at the number of bytes read from the file system required to run the query. If the Query Isolation feature has been enabled for a Virtual Warehouse and a query scans more data than the threshold set in the hive.query.isolation.scan.size.threshold parameter, the planner runs the query in isolation. This means that an isolated standalone executor group is spawned to run the data-intensive query. For more details, see Hive query isolation for scan-heavy, data-intensive queries.

Overlay network support for AWS environments

An overlay network is a software-defined layer of network abstraction that is used to run multiple separate, discrete virtualized network layers over the AWS VPC network. In the case of the CDW service, a custom CNI (Container Network Interface) plugin is used to enable the overlay network. It creates two network spaces:

- A node network space, which derives per-node IP addresses from the VPC.
- A Kubernetes pod network space, which derives per-pod IP addresses from the CNI plugin's own network space.

The overlay network is bridged into the node network. As a result, one IP address is required per node instead of one IP address needed per pod. Consequently, there are more available IP addresses and you can use the CDW service efficiently, auto-scaling Virtual Warehouses as needed to meet the demands of your workloads. For more information, see Use overlay networks for AWS environments in Cloudera Data Warehouse service and Enable overlay networks in AWS environments.

May 14, 2020

This release of the Cloudera Data Warehouse service on CDP Public Cloud introduces the following new features and improvements:

Simplified private networking deployments on AWS

Private networking deployments on AWS for Data Warehouse service have been simplified. When using the Private Load Balancer, Private Worker Nodes deployment mode, both public and private subnets in your AWS VPC are no

longer needed. Now this deployment mode only requires 3 private subnets. For more information, see Supported deployment modes for private networking in AWS.

Impala Virtual Warehouses offer faster SQL analytics

Impala Virtual Warehouses have been improved with the following enhancements:

- More granular control and performance optimizations on multi-threaded query execution.
- Improved read performance for ORC tables with nested type columns.
- Improved automatic updates of metadata.

Generate and download Impala diagnostic bundles

You can now generate and download diagnostic bundles of log files for Impala Virtual Warehouses on AWS environments. For more information, see Downloading Impala diagnostic bundles from AWS.

Launch Grafana dashboards from environment tiles in Cloudera Data Warehouse service

You can now launch Grafana dashboards from environment tiles in the Cloudera Data Warehouse service to monitor the following components:

- Hive
- Impala
- Istio mesh
- Kubernetes clusters

For more information, see Connecting to Grafana dashboards.

Single Sign-On (SSO) support for Hue with Hive Virtual Warehouses

You can now access Hue to query Hive Virtual Warehouses from the Virtual Warehouse tile:

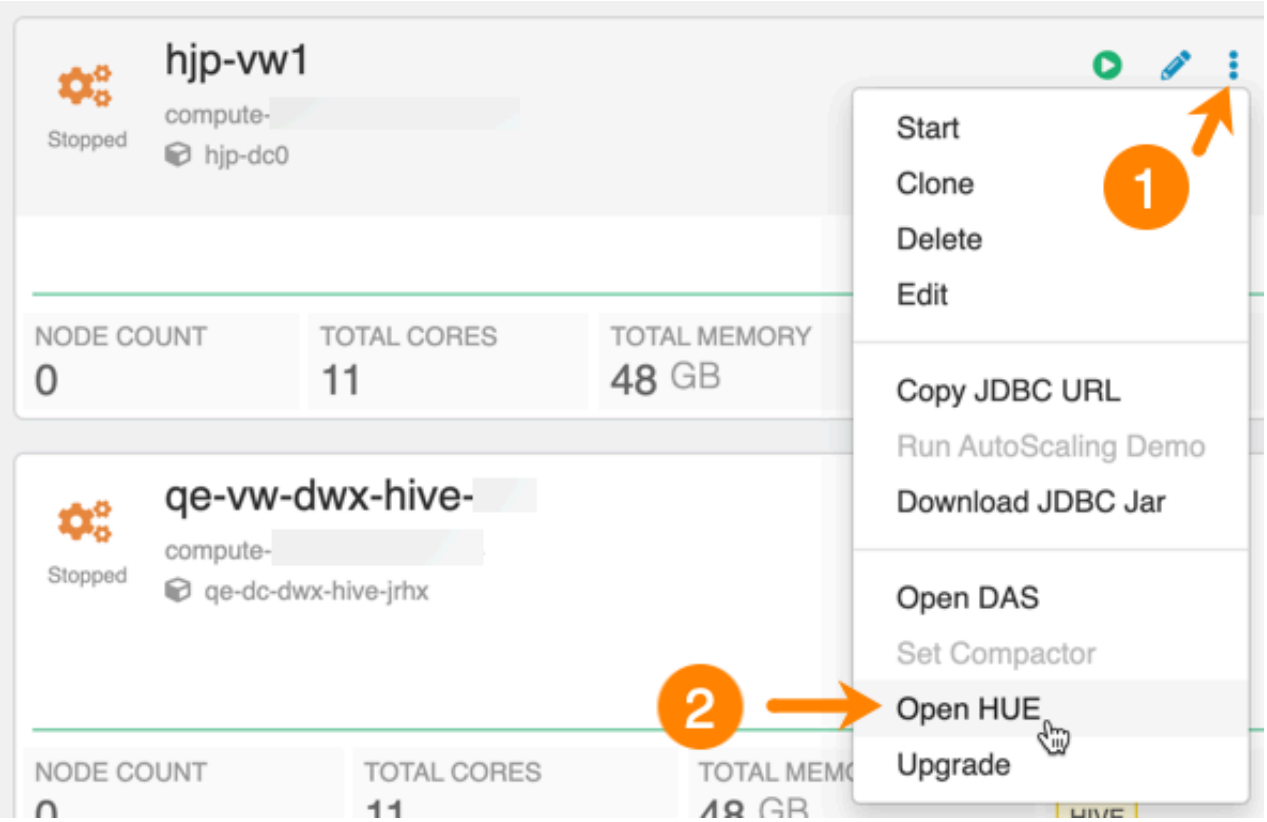

DAS updates

In this release, DAS has been updated as follows:

- You can now specify administrative groups for the DAS web app.
- Multi-DAG support for a single Hive query.

April 7, 2020

This release of the Cloudera Data Warehouse service on CDP Public Cloud introduces the following new features and improvements:

Hue single-sign-on (SSO) support using SAML

In this release, Hue now supports SSO by using SAML. For more information, see Authenticating users with SAML in the Hue documentation.

Private networking feature now available for AWS environments when only private subnets are available

When you are unable to use public subnets in your AWS VPC, CDP Data Warehouse service now supports private networking for your Data Warehouse in this situation. To configure private networking for AWS environments where only private subnets are available to use, register a CDP environment in AWS with three private subnets. Then network administrators must make sure there is outbound internet access from private subnets by way of the transit gateway or by another means. For information about configuring private networking with these AWS environments, see Step 4 in Activating an AWS environment with private subnet support.

Azure support improvements

- Additional regions are now supported, including:
  - Central US
  - East US
  - East US 2
  - West US
  - West US 2
  - Australia East
- Improved identity management.

Improved resiliency for Impala Virtual Warehouses

Resiliency for Impala Virtual Warehouses has been improved by reducing the amount of ephemeral storage used on compute nodes.

March 13, 2020

This release of the Cloudera Data Warehouse service on CDP Public Cloud introduces the following new features and improvements:

Azure support for Data Warehouse service

This release of Data Warehouse service now supports Microsoft Azure as a public cloud provider. For more details about this added support, see Azure support for Data Warehouse service.

Impala Virtual Warehouse improvements

Impala Virtual Warehouses now include the following improvements:

- Improved resiliency for client connections.

- Added more health checks to more accurately indicate the start of a Virtual Warehouse. This helps with query timeouts due to provisioning lags.

Known issues (unsupported releases)

Known issues in unsupported releases.

January 18, 2024

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.6.

Carried over from the previous release: Upgrade-related

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

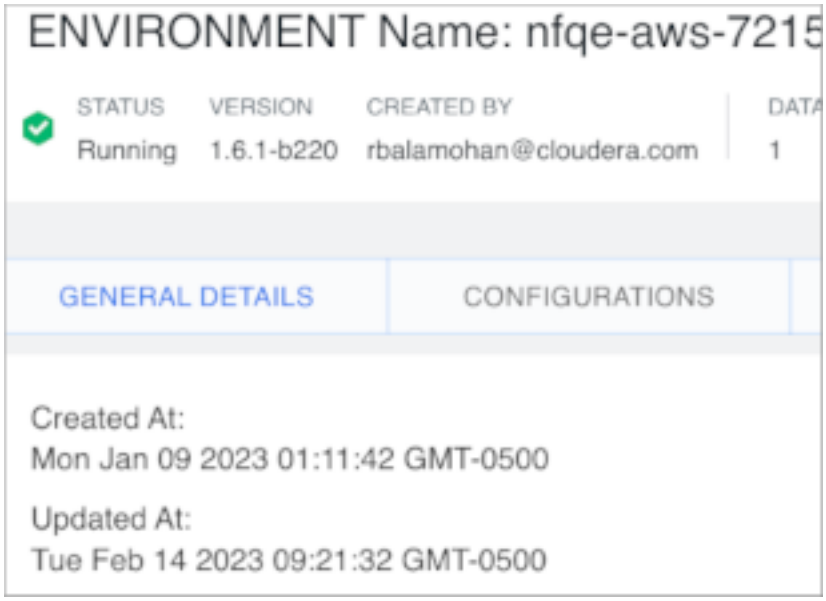

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

Compaction causes data loss under certain conditions

Data loss occurs during compaction when Apache Ranger policies for masking or row filtering are enabled and compaction users are included in the policies.

Workaround: Exclude compaction users from the policies as described in Compaction Prerequisites.

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

```
worker subnets should be selected from 3 different
availability zones. got: 0
```

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
```

```
        operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

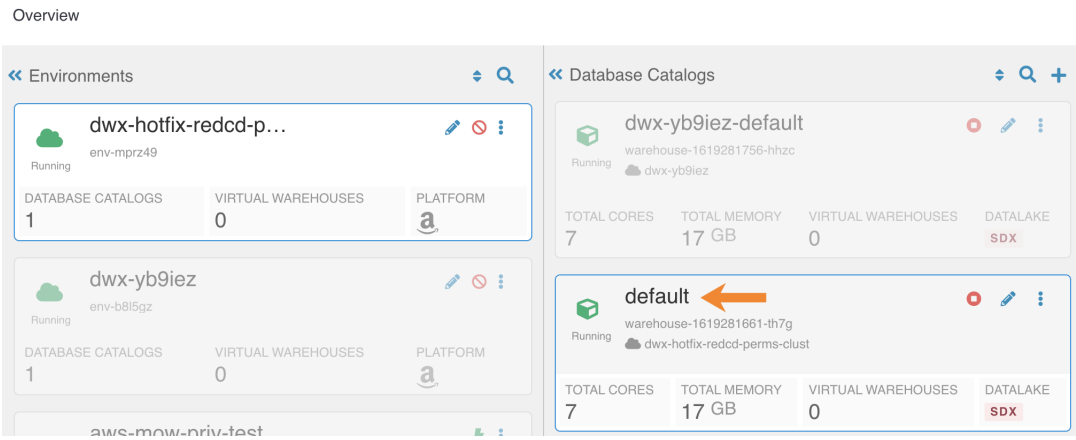

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Limited Hive image versions

Hive Virtual Warehouses you create in 1.8.1-b248 (released Nov 20, 2023) and later will run Istio 1.19.0. The new Istio version supports only new versions of Hive helm charts. If you have the CDW_VERSIONED_DEPLOY, only new Hive image versions appear in UI when you create a new Hive Virtual Warehouse.

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

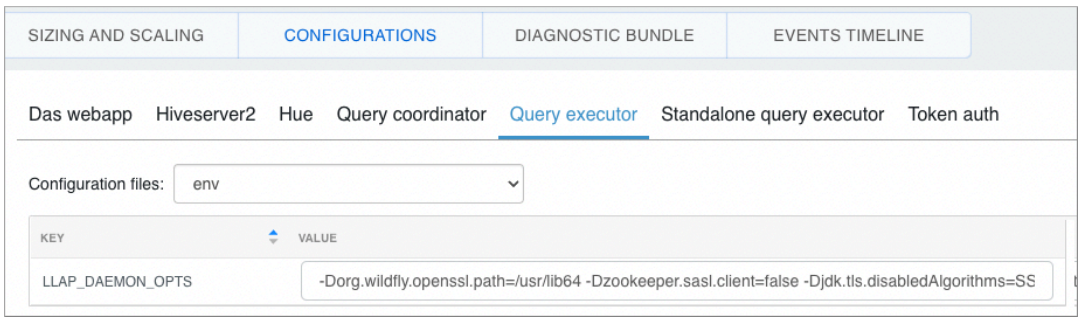

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
       - apps
       resources:
       - deployments
       - statefulsets
       verbs:
       - get
       - list
       - apiGroups:
       - "edws.cloudera.com"
       resources:
       - computes
       verbs:
```

```
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

DWX-16899: Error while viewing Impala job status on the mini Job Browser

When you click on the application ID after submitting an Impala query on Hue that is running on the environment level, you may notice the following error: 401 Client Error: Unauthorized for url: http://coordinator.impala-xyz.svc.cluster.local:25000/queries?json=true Must authenticate with Basic authentication. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16913: “Results have expired” message while running a CTAS query

You may see the following message on the Hue UI when you submit a CREATE TABLE AS SELECT (CTAS) query from a Hue instance that is deployed on the environment level: “Results have expired, rerun the query if needed”. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16917: Failed to validate proxy privileges error while running queries from Hue

You may see the following error intermittently in Hue’s pod logs while running queries from a Hue instance that is deployed at the environment level: “Failed to validate proxy privileges of <username>”.

None.

DWX-16895: Incorrect status of Hue pods when you edit the Hue instance properties

When you update a configuration of a Hue instance that is deployed at the environment level, such as increasing or decreasing the size of the Hue instance, you see a success message on the CDW UI. After some time, the status of the Hue instance also changes from “Updating” to “Running”. However, when you list the Hue pods using kubectl, you see that not all backend pods are in the running state–a few of them are still in the init state.

None. The pods come up successfully eventually after a sufficient time has passed.

DWX-16863: The upgrade button is present on the CDW UI, but Hue upgrades are not supported

You see the Upgrade button on the **Query Editor** page in the CDW UI when Hue is deployed at the environment level. However, on CDW version 1.8.1, upgrading the Hue instance that is deployed at the environment level is not supported.

None.

DWX-16893: A user can see all the queries in Job browser

In a Hue instance deployed at the environment level, by design, the Hue instances must not share the saved queries and query history with other Hue instances even for the same user. However, a logged in user is able to view all the queries executed by that user on all the Virtual Warehouses on a particular Database Catalog.

None.

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-17210: Timeout issue querying Iceberg tables from Hive

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

- fs.s3.maxConnections=1000
- fs.s3a.connection.maximum=1000

Restart the Database Catalog and Virtual Warehouse.

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-17175: Impala Virtual Warehouse max executor limit issue

The Impala Virtual Warehouse has a limit of max 200 executors. The Executors slider has a max limit of 200 for creating or editing a Virtual Warehouse. If you select a bigger t-shirt size than the default, or set a custom t-shirt size, the default max limit set by the UI could be more than 200 in some cases. Exceeding the max limit causes an error during a future Edit/Upgrade/Rebuild operation.

Workaround: Manually set the max executors limit to a preferred value less than 200, but a multiple of t-shirt size, when creating or editing Impala Virtual Warehouse

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml
```

```
        # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

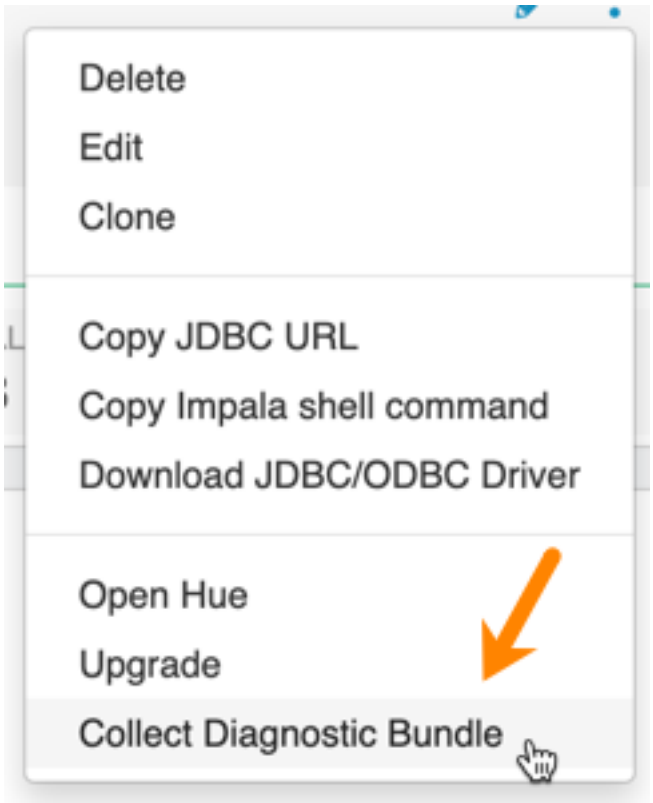

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2023-719: Cloudera Data Warehouse Backup/Restore of CDP Data Visualization incomplete

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature will encounter an issue with the restoration versions of Cloudera Data Visualization (CDV) older

than CDV 7.1.6.2-3 due to a schema change in this release. If the Backup was taken from an older CDW environment that contained a version of CDV older than CDV 7.1.6.2-3, the Restore procedure will succeed. Though once the user opens the CDV Queries tab, the user could encounter the error message: “column jobs_jobschedule.owner_id does not exist…”

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete.

TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature might encounter an issue with the restoration of the Database Catalog, if the hue-query-processor.json configuration file of the Database Catalog has been edited. Even a minor edit to the hue-query-processor.json configuration file can result in this failure.

During the restoration process the Database Catalog will be created, but it will fail to start after a short period of time, and the Database Catalog will be in a Bad Health state on the CDW User Interface.

Inside the Kubernetes Cluster (Azure Kubernetes Service on Azure / Elastic Kubernetes Service on AWS) the StatefulSet of Hue Query Processor (hue-query-processor) is in CrashLoop. This is indicated with the following log in the Hue Query Processor StatefulSet Pod:

```
SQL State  : 3D000
Error Code : 0
Message    : FATAL: database "warehouse-1707832123-abcd_hueqpdb"
 does not exist



              at org.flywaydb.core.internal.jdbc.JdbcUtils.openC
onnection(JdbcUtils.java:65)
```

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW.

TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW

Cloudera Data Warehouse (CDW) customers using Apache Impala (Impala) to read Apache Iceberg (Iceberg) V2 tables can encounter an issue of Impala returning incorrect results when the optimized V2 operator is used. The optimized V2 operator is enabled by default in the affected versions below. The issue only affects Iceberg V2 tables that have position delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW.

TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

Apache Hive (Hive) and Apache Impala (Impala) modify statements (DELETE/UPDATE/MERGE) on Apache Iceberg (Iceberg) V2 tables can corrupt the tables if there is a concurrent table compaction from Apache Spark. The issue happens when the compaction and modify statement run in parallel, and when the compaction job commits before the modify statement. In this case the position delete files of the modify statement still point to the old files. This means the following in case of

- DELETE statements
  - Deleting records pointing to old files have no effect
- UPDATE / MERGE statements
  - Deleting records pointing to old files have no effect
  - The table will also have the newly added data records
  - Rewritten records will still be active

This issue does not affect Apache NiFi (NiFi) and Apache Flink (Flink) as these components write equality delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables.

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

When working with Apache Hive (Hive) and Apache Iceberg (Iceberg) V2 tables, using the TRUNCATE statement may lead to unintended data deletion. This issue arises when the truncate command is applied to a branch of an Iceberg table. Instead of truncating the branch itself, the command affects the original (main) table, which results in unintended loss of data.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

January 10, 2024

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.6.

TSB 2024-733: Changes in CoreDNS of AWS EKS affect operations in CDW, CDE and CML clusters on Public Cloud

Affected users: All Cloudera Data Platform (CDP) Public Cloud AWS users who have:

- CDW environments with a version lower than CDW 1.8.4.
- CDE on EKS-1.25 and lower version
- CML on EKS-1.25 and lower versions

The latest Amazon Web Services (AWS) Elastic Kubernetes Service (EKS) cluster deployment has changed the behavior of CoreDNS. The changes to the CoreDNS Deployment sets the readinessProbe to use the /ready endpoint, which breaks the current functions of cdp-coredns-updater. This causes a malfunction in CoreDNS that affects the Cloudera Data Warehouse (CDW), Cloudera Data Engineering (CDE) and Cloudera Machine Learning (CML) cluster operations. For the AWS CoreDNS related changes, see important CoreDNS upgrade considerations.

For the latest update on this issue see the corresponding Knowledge article: Cloudera Customer Advisory 2023-733: Don't Activate Or Backup/Restore Clusters in AWS until CDW 1.8.4 released

Carried over from the previous release: Upgrade-related

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

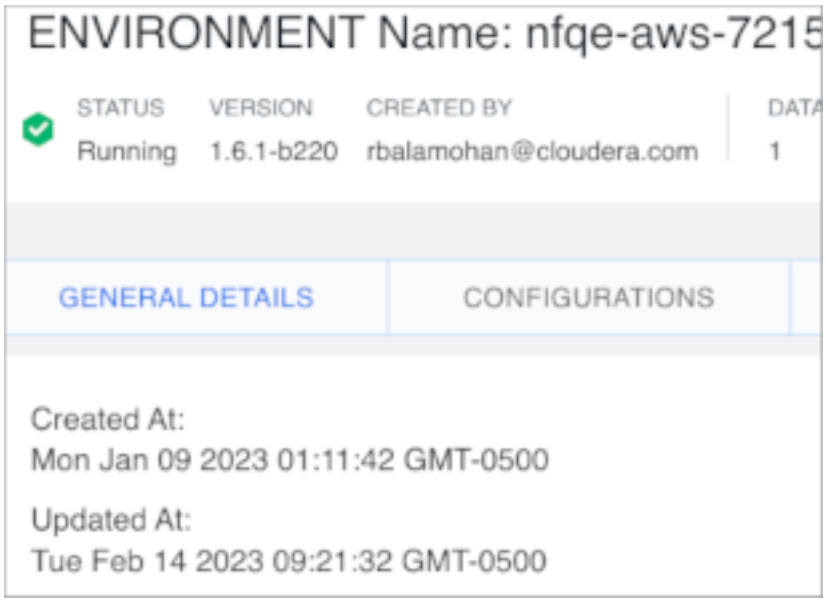

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

Compaction causes data loss under certain conditions

Data loss occurs during compaction when Apache Ranger policies for masking or row filtering are enabled and compaction users are included in the policies.

Workaround: Exclude compaction users from the policies as described in Compaction Prerequisites.

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

```
worker subnets should be selected from 3 different
availability zones. got: 0
```

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the Cr eateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersio ns line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
```

```
            "autoscaling:DescribeAutoScalingGroups",
            "autoscaling:DescribeAutoScalingInstances",
            "autoscaling:DescribeTags",
            "autoscaling:DescribeLaunchConfigurations",
            "autoscaling:SetDesiredCapacity",
            "autoscaling:TerminateInstanceInAutoScalingGroup",
            "ec2:DescribeLaunchTemplateVersions",
            "ec2:CreateTags"
            ],
```

7. Save changes.

Carried over from the previous release: Azure

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

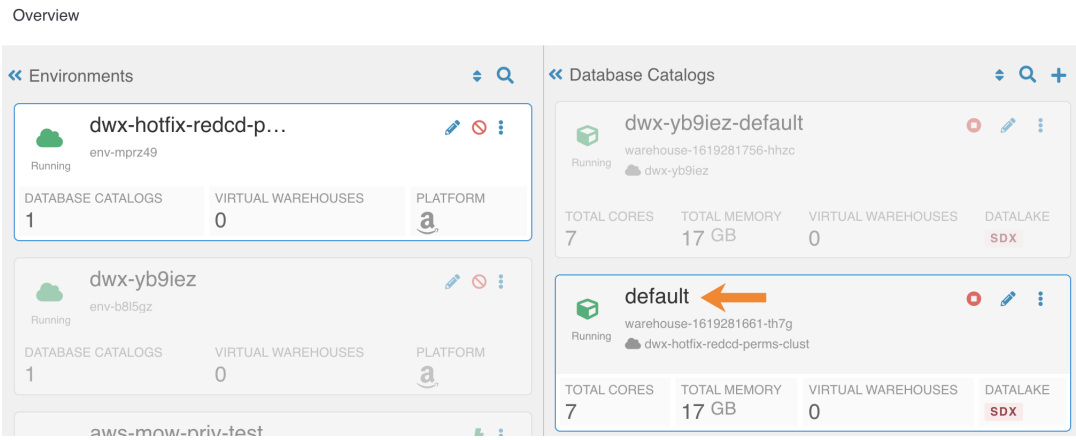

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Limited Hive image versions

Hive Virtual Warehouses you create in 1.8.1-b248 (released Nov 20, 2023) and later will run Istio 1.19.0. The new Istio version supports only new versions of Hive helm charts. If you have the CDW_VERSIONED_DEPLOY, only new Hive image versions appear in UI when you create a new Hive Virtual Warehouse.

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

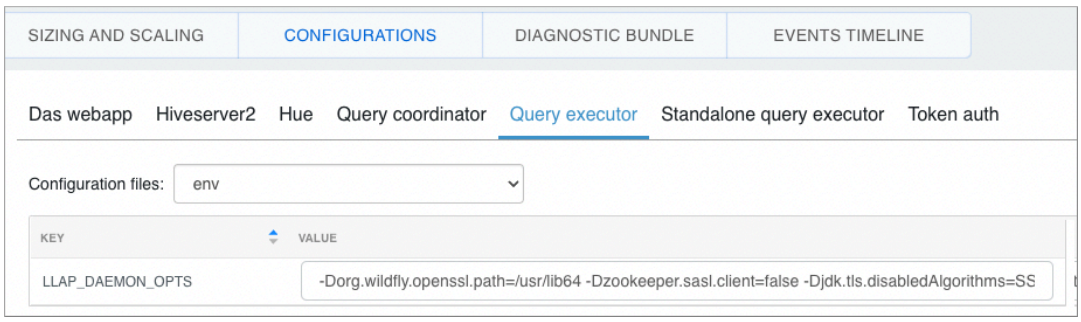

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
       - apps
       resources:
       - deployments
       - statefulsets
       verbs:
       - get
       - list
       - apiGroups:
       - "edws.cloudera.com"
       resources:
       - computes
       verbs:
```

```
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

DWX-16899: Error while viewing Impala job status on the mini Job Browser

When you click on the application ID after submitting an Impala query on Hue that is running on the environment level, you may notice the following error: 401 Client Error: Unauthorized for url: http://coordinator.impala-xyz.svc.cluster.local:25000/queries?json=true Must authenticate with Basic authentication. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16913: “Results have expired” message while running a CTAS query

You may see the following message on the Hue UI when you submit a CREATE TABLE AS SELECT (CTAS) query from a Hue instance that is deployed on the environment level: “Results have expired, rerun the query if needed”. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16917: Failed to validate proxy privileges error while running queries from Hue

You may see the following error intermittently in Hue’s pod logs while running queries from a Hue instance that is deployed at the environment level: “Failed to validate proxy privileges of <username>”.

None.

DWX-16895: Incorrect status of Hue pods when you edit the Hue instance properties

When you update a configuration of a Hue instance that is deployed at the environment level, such as increasing or decreasing the size of the Hue instance, you see a success message on the CDW UI. After some time, the status of the Hue instance also changes from “Updating” to “Running”. However, when you list the Hue pods using kubectl, you see that not all backend pods are in the running state–a few of them are still in the init state.

None. The pods come up successfully eventually after a sufficient time has passed.

DWX-16863: The upgrade button is present on the CDW UI, but Hue upgrades are not supported

You see the Upgrade button on the **Query Editor** page in the CDW UI when Hue is deployed at the environment level. However, on CDW version 1.8.1, upgrading the Hue instance that is deployed at the environment level is not supported.

None.

DWX-16893: A user can see all the queries in Job browser

In a Hue instance deployed at the environment level, by design, the Hue instances must not share the saved queries and query history with other Hue instances even for the same user. However, a logged in user is able to view all the queries executed by that user on all the Virtual Warehouses on a particular Database Catalog.

None.

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-17210: Timeout issue querying Iceberg tables from Hive

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

- fs.s3.maxConnections=1000
- fs.s3a.connection.maximum=1000

Restart the Database Catalog and Virtual Warehouse.

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-17175: Impala Virtual Warehouse max executor limit issue

The Impala Virtual Warehouse has a limit of max 200 executors. The Executors slider has a max limit of 200 for creating or editing a Virtual Warehouse. If you select a bigger t-shirt size than the default, or set a custom t-shirt size, the default max limit set by the UI could be more than 200 in some cases. Exceeding the max limit causes an error during a future Edit/Upgrade/Rebuild operation.

Workaround: Manually set the max executors limit to a preferred value less than 200, but a multiple of t-shirt size, when creating or editing Impala Virtual Warehouse

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml
```

```
        # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

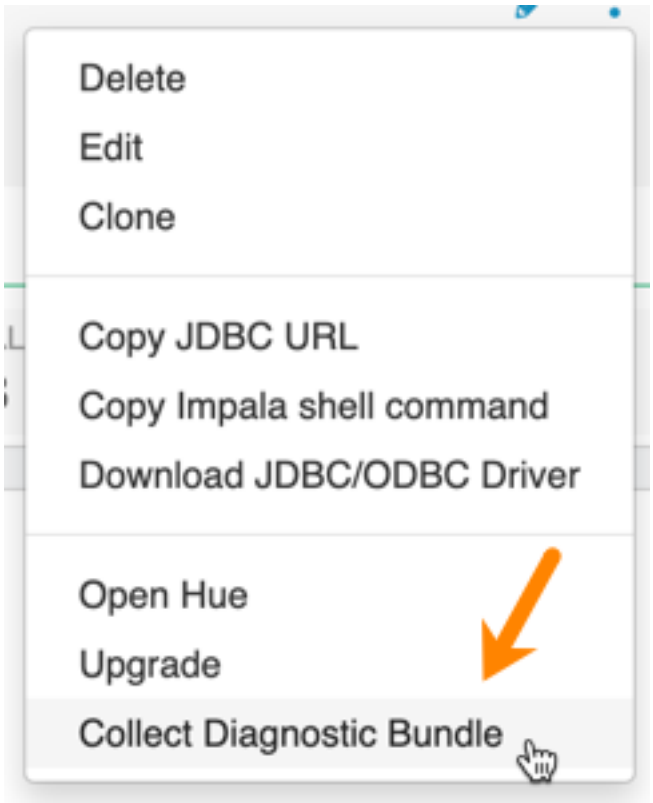

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2023-719: Cloudera Data Warehouse Backup/Restore of CDP Data Visualization incomplete

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature will encounter an issue with the restoration versions of Cloudera Data Visualization (CDV) older

than CDV 7.1.6.2-3 due to a schema change in this release. If the Backup was taken from an older CDW environment that contained a version of CDV older than CDV 7.1.6.2-3, the Restore procedure will succeed. Though once the user opens the CDV Queries tab, the user could encounter the error message: “column jobs_jobschedule.owner_id does not exist…”

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete.

TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature might encounter an issue with the restoration of the Database Catalog, if the hue-query-processor.json configuration file of the Database Catalog has been edited. Even a minor edit to the hue-query-processor.json configuration file can result in this failure.

During the restoration process the Database Catalog will be created, but it will fail to start after a short period of time, and the Database Catalog will be in a Bad Health state on the CDW User Interface.

Inside the Kubernetes Cluster (Azure Kubernetes Service on Azure / Elastic Kubernetes Service on AWS) the StatefulSet of Hue Query Processor (hue-query-processor) is in CrashLoop. This is indicated with the following log in the Hue Query Processor StatefulSet Pod:

```
SQL State  : 3D000
Error Code : 0
Message    : FATAL: database "warehouse-1707832123-abcd_hueqpdb"
 does not exist



              at org.flywaydb.core.internal.jdbc.JdbcUtils.openC
onnection(JdbcUtils.java:65)
```

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW.

TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW

Cloudera Data Warehouse (CDW) customers using Apache Impala (Impala) to read Apache Iceberg (Iceberg) V2 tables can encounter an issue of Impala returning incorrect results when the optimized V2 operator is used. The optimized V2 operator is enabled by default in the affected versions below. The issue only affects Iceberg V2 tables that have position delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW.

TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

Apache Hive (Hive) and Apache Impala (Impala) modify statements (DELETE/UPDATE/MERGE) on Apache Iceberg (Iceberg) V2 tables can corrupt the tables if there is a concurrent table compaction from Apache Spark. The issue happens when the compaction and modify statement run in parallel, and when the compaction job commits before the modify statement. In this case the position delete files of the modify statement still point to the old files. This means the following in case of

- DELETE statements
  - Deleting records pointing to old files have no effect
- UPDATE / MERGE statements
  - Deleting records pointing to old files have no effect
  - The table will also have the newly added data records
  - Rewritten records will still be active

This issue does not affect Apache NiFi (NiFi) and Apache Flink (Flink) as these components write equality delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables.

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

When working with Apache Hive (Hive) and Apache Iceberg (Iceberg) V2 tables, using the TRUNCATE statement may lead to unintended data deletion. This issue arises when the truncate command is applied to a branch of an Iceberg table. Instead of truncating the branch itself, the command affects the original (main) table, which results in unintended loss of data.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion.

December 1, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New Known Issues in this release

See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.6.

Carried over from the previous release: Upgrade-related

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

TSB 2023-719: Cloudera Data Warehouse Backup/Restore of CDP Data Visualization incomplete

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature will encounter an issue with the restoration versions of Cloudera Data Visualization (CDV) older than CDV 7.1.6.2-3 due to a schema change in this release. If the Backup was taken from an older CDW environment that contained a version of CDV older than CDV 7.1.6.2-3, the Restore procedure will succeed. Though once the user opens the CDV Queries tab, the user could encounter the error message: “column jobs_jobschedule.owner_id does not exist…”

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

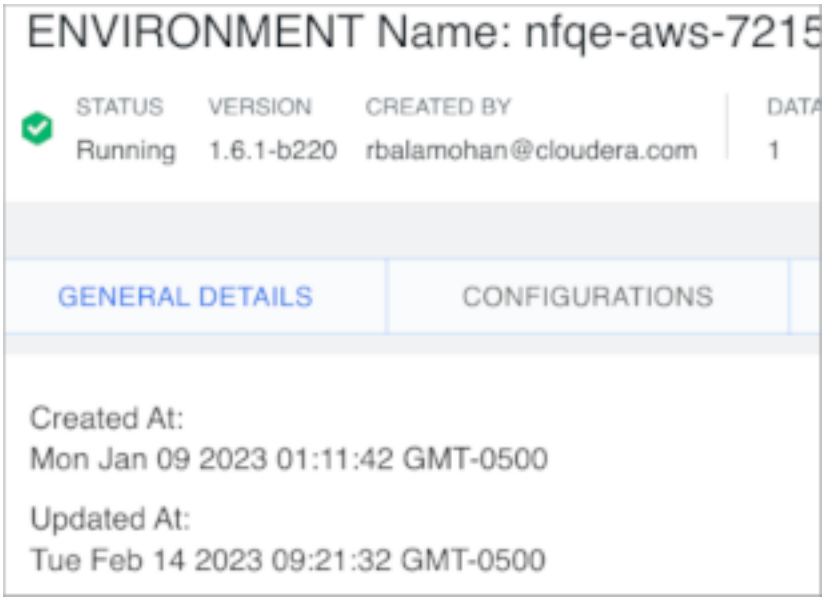

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

Compaction causes data loss under certain conditions

Data loss occurs during compaction when Apache Ranger policies for masking or row filtering are enabled and compaction users are included in the policies.

Workaround: Exclude compaction users from the policies as described in Compaction Prerequisites.

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment

validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

worker subnets should be selected from 3 different availability zones. got: 0

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

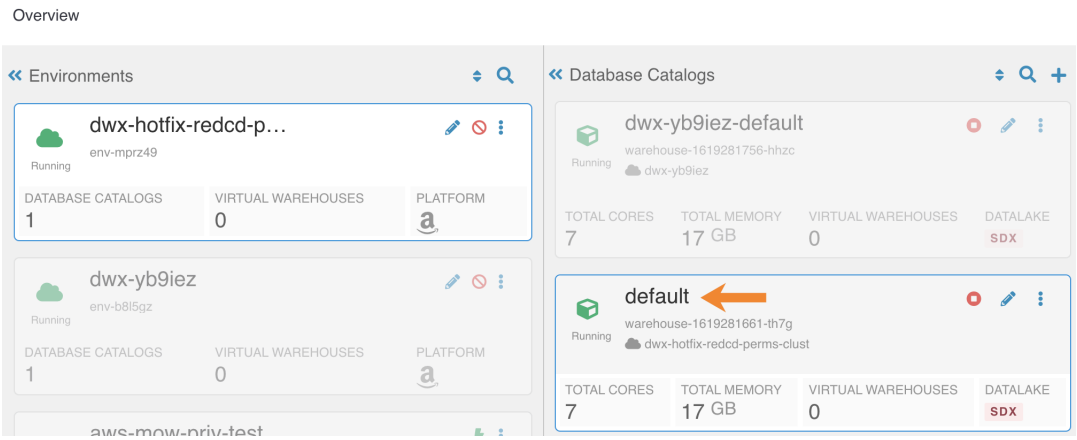

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Limited Hive image versions

Hive Virtual Warehouses you create in 1.8.1-b248 (released Nov 20, 2023) and later will run Istio 1.19.0. The new Istio version supports only new versions of Hive helm charts. If you have the CDW_VERSIONED_DEPLOY, only new Hive image versions appear in UI when you create a new Hive Virtual Warehouse.

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

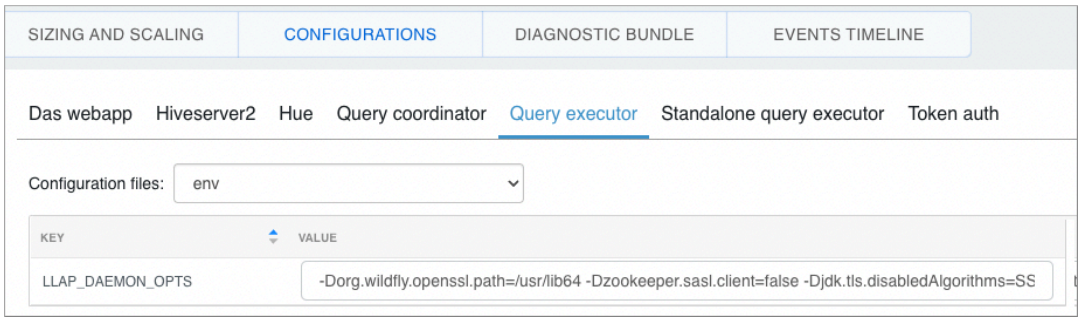

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
```

```
        - ""
        resources:
        - configmaps
        - events
        - pods
        - persistentvolumeclaims
        - nodes
        verbs:
        - get
        - list
        - apiGroups:
        - apps
        resources:
        - deployments
        - statefulsets
        verbs:
        - get
        - list
        - apiGroups:
        - "edws.cloudera.com"
        resources:
        - computes
        verbs:
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

DWX-16899: Error while viewing Impala job status on the mini Job Browser

When you click on the application ID after submitting an Impala query on Hue that is running on the environment level, you may notice the following error: 401 Client Error: Unauthorized for url: http://coordinator.impala-xyz.svc.cluster.local:25000/queries?json=true Must authenticate with Basic authentication. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16913: “Results have expired” message while running a CTAS query

You may see the following message on the Hue UI when you submit a CREATE TABLE AS SELECT (CTAS) query from a Hue instance that is deployed on the environment level: “Results have expired, rerun the query if needed”. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16917: Failed to validate proxy privileges error while running queries from Hue

You may see the following error intermittently in Hue’s pod logs while running queries from a Hue instance that is deployed at the environment level: “Failed to validate proxy privileges of <username>”.

None.

DWX-16895: Incorrect status of Hue pods when you edit the Hue instance properties

When you update a configuration of a Hue instance that is deployed at the environment level, such as increasing or decreasing the size of the Hue instance, you see a success message on the CDW UI. After some time, the status of the Hue instance also changes from “Updating” to “Running”.

However, when you list the Hue pods using kubectl, you see that not all backend pods are in the running state–a few of them are still in the init state.

None. The pods come up successfully eventually after a sufficient time has passed.

DWX-16863: The upgrade button is present on the CDW UI, but Hue upgrades are not supported

You see the Upgrade button on the **Query Editor** page in the CDW UI when Hue is deployed at the environment level. However, on CDW version 1.8.1, upgrading the Hue instance that is deployed at the environment level is not supported.

None.

DWX-16893: A user can see all the queries in Job browser

In a Hue instance deployed at the environment level, by design, the Hue instances must not share the saved queries and query history with other Hue instances even for the same user. However, a logged in user is able to view all the queries executed by that user on all the Virtual Warehouses on a particular Database Catalog.

None.

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

Concurrent compactions and modify statements can corrupt Iceberg tables

Hive or Impala DELETE/UPDATE/MERGE operations on Iceberg V2 tables can corrupt the tables if there is a concurrent table compaction from Spark. The issue happens if the compaction and modify statement runs in parallel, and if the compaction job commits before the modify statement. In that case the modify statement's position delete files still point to the old files. The results in the case of DELETE and in the case of UPDATE / MERGE are as follows:

- DELETE

  Delete records pointing to old files have no effect.
- UPDATE / MERGE

  Delete records pointing to old files have no effect. The table will also have the newly added data records, which means rewritten records will still be active.

Use one of the following workarounds:

- Do not run compactions and DELETE/UPDATE/MERGE statements in parallel.

- Do not compact the table via Iceberg's RewriteFiles operation. For example do not use Spark's rewriteDataFiles.

DWX-17210: Timeout issue querying Iceberg tables from Hive

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

- fs.s3.maxConnections=1000
- fs.s3a.connection.maximum=1000

Restart the Database Catalog and Virtual Warehouse.

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.

- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-17175: Impala Virtual Warehouse max executor limit issue

The Impala Virtual Warehouse has a limit of max 200 executors. The Executors slider has a max limit of 200 for creating or editing a Virtual Warehouse. If you select a bigger t-shirt size than the default, or set a custom t-shirt size, the default max limit set by the UI could be more than 200 in some cases. Exceeding the max limit causes an error during a future Edit/Upgrade/Rebuild operation.

Workaround: Manually set the max executors limit to a preferred value less than 200, but a multiple of t-shirt size, when creating or editing Impala Virtual Warehouse

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

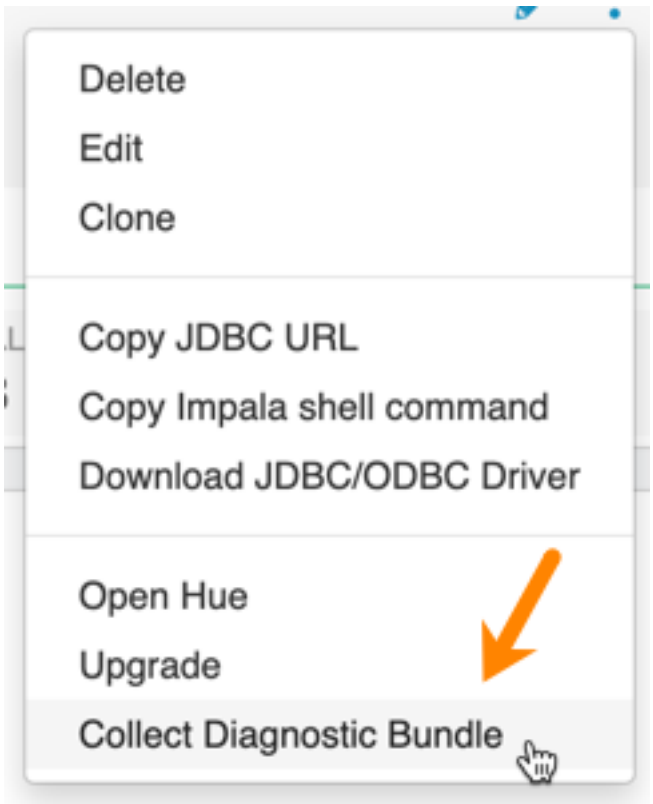

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2024-723: Hue RAZ is using logger role to Read and Upload/Delete (write) files

When using Cloudera Data Hub for Public Cloud (Data Hub) on Amazon Web Services (AWS), users can use the Hue File Browser feature to access the filesystem, and if permitted, read and write directly to the related S3 buckets. As AWS does not provide fine-grained access control, Cloudera Data Platform administrators can use the Ranger Authorization Service (RAZ) capability to take the S3 filesystem, and overlay it with user and group specific permissions, making it easier to allow certain users to have limited permissions, without having to grant those users permissions to the entire S3 bucket.

This bulletin describes an issue when using RAZ with Data Hub, and attempting to use fine-grained access control to allow certain users write permissions.

Through RAZ, an administrator may, for a particular user, specify permissions more limited than what AWS provides for an S3 bucket, allowing the user to have read/write (or other similar

fine grained access) permissions on only a subset of the files and directories within that bucket. However, under specific conditions, it is possible for such user to be able to read and write to the entire S3 bucket through Hue, due to Hue using the logger role (which will have full read/write to the S3 bucket) when using Data Hub with a RAZ enabled cluster. This problem also can affect the Hue service itself, by affecting proper access to home directories causing the service role to not start.

The root cause of this issue is, when accessing Amazon cloud resources, Hue uses the AWS Boto SDK library. This AWS Boto library has a bug that restricts permissions in certain AWS regions in such a way that it provides access to users who should not have it, regardless of RAZ settings. This issue only affects users in specific AWS regions, listed below, and it does not affect all AWS customers.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-723: Hue Raz is using logger role to Read and Upload/Delete (write) files.

December 19, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New Known Issues in this release

See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.6.

DWX-17175: Impala Virtual Warehouse max executor limit issue

The Impala Virtual Warehouse has a limit of max 200 executors. The Executors slider has a max limit of 200 for creating or editing a Virtual Warehouse. If you select a bigger t-shirt size than the default, or set a custom t-shirt size, the default max limit set by the UI could be more than 200 in some cases. Exceeding the max limit causes an error during a future Edit/Upgrade/Rebuild operation.

Workaround: Manually set the max executors limit to a preferred value less than 200, but a multiple of t-shirt size, when creating or editing Impala Virtual Warehouse

DWX-17210: Timeout issue querying Iceberg tables from Hive

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

- fs.s3.maxConnections=1000
- fs.s3a.connection.maximum=1000

Restart the Database Catalog and Virtual Warehouse.

Carried over from the previous release: Upgrade-related

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

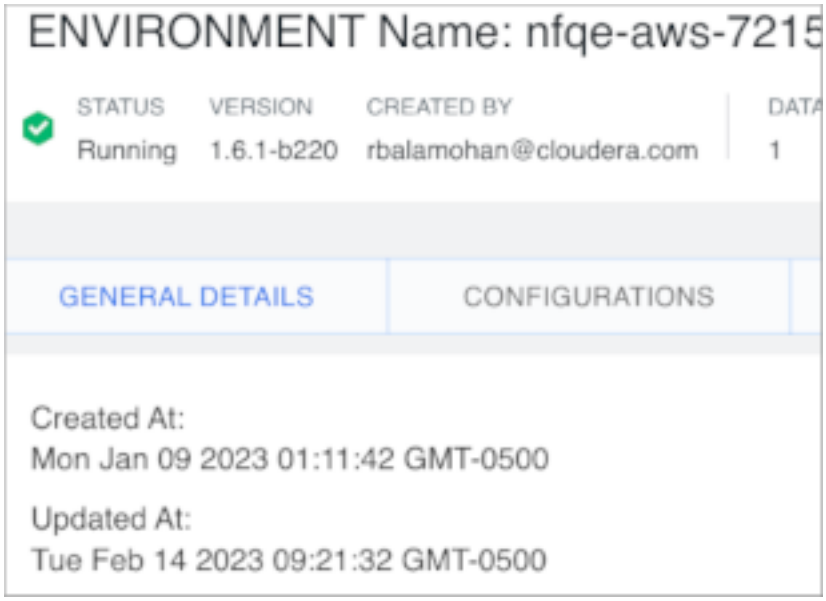

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

Compaction causes data loss under certain conditions

Data loss occurs during compaction when Apache Ranger policies for masking or row filtering are enabled and compaction users are included in the policies.

Workaround: Exclude compaction users from the policies as described in Compaction Prerequisites.

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

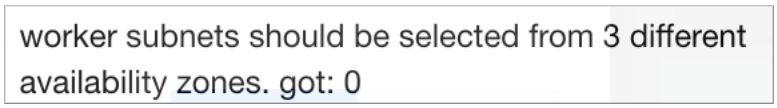

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
```

```
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).

- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

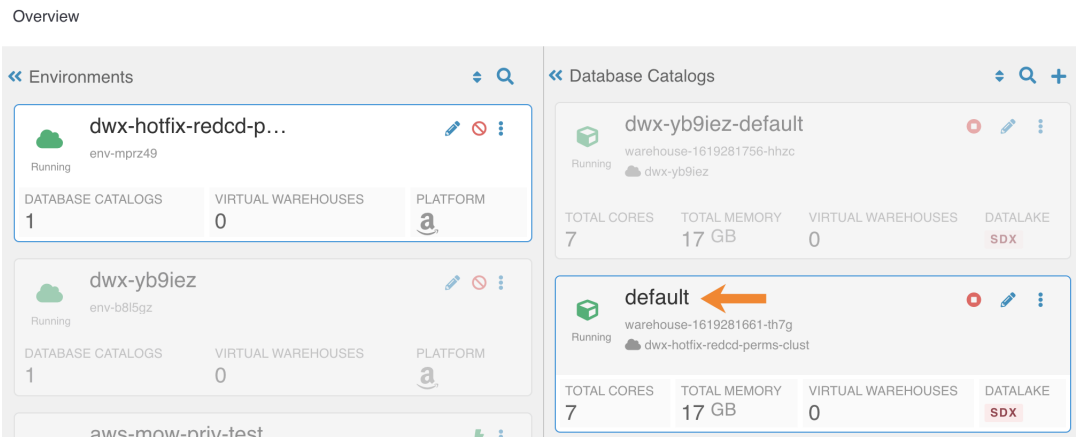

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Limited Hive image versions

Hive Virtual Warehouses you create in 1.8.1-b248 (released Nov 20, 2023) and later will run Istio 1.19.0. The new Istio version supports only new versions of Hive helm charts. If you have the CDW_VERSIONED_DEPLOY, only new Hive image versions appear in UI when you create a new Hive Virtual Warehouse.

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

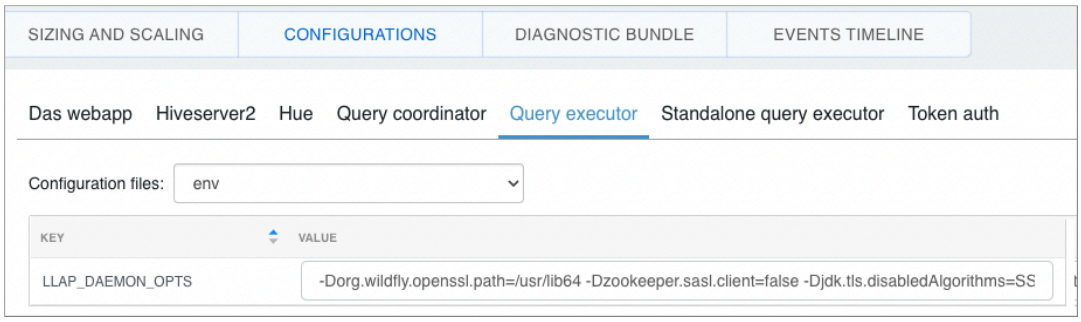

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
```

```
        - apps
        resources:
        - deployments
        - statefulsets
        verbs:
        - get
        - list
        - apiGroups:
        - "edws.cloudera.com"
        resources:
        - computes
        verbs:
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

DWX-16899: Error while viewing Impala job status on the mini Job Browser

When you click on the application ID after submitting an Impala query on Hue that is running on the environment level, you may notice the following error: 401 Client Error: Unauthorized for url: http://coordinator.impala-xyz.svc.cluster.local:25000/queries?json=true Must authenticate with Basic authentication. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16913: “Results have expired” message while running a CTAS query

You may see the following message on the Hue UI when you submit a CREATE TABLE AS SELECT (CTAS) query from a Hue instance that is deployed on the environment level: “Results have expired, rerun the query if needed”. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16917: Failed to validate proxy privileges error while running queries from Hue

You may see the following error intermittently in Hue’s pod logs while running queries from a Hue instance that is deployed at the environment level: “Failed to validate proxy privileges of <username>”.

None.

DWX-16895: Incorrect status of Hue pods when you edit the Hue instance properties

When you update a configuration of a Hue instance that is deployed at the environment level, such as increasing or decreasing the size of the Hue instance, you see a success message on the CDW UI. After some time, the status of the Hue instance also changes from “Updating” to “Running”.

However, when you list the Hue pods using kubectl, you see that not all backend pods are in the running state–a few of them are still in the init state.

None. The pods come up successfully eventually after a sufficient time has passed.

DWX-16863: The upgrade button is present on the CDW UI, but Hue upgrades are not supported

You see the Upgrade button on the **Query Editor** page in the CDW UI when Hue is deployed at the environment level. However, on CDW version 1.8.1, upgrading the Hue instance that is deployed at the environment level is not supported.

None.

DWX-16893: A user can see all the queries in Job browser

In a Hue instance deployed at the environment level, by design, the Hue instances must not share the saved queries and query history with other Hue instances even for the same user. However, a logged in user is able to view all the queries executed by that user on all the Virtual Warehouses on a particular Database Catalog.

None.

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.
```

```
kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

# Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

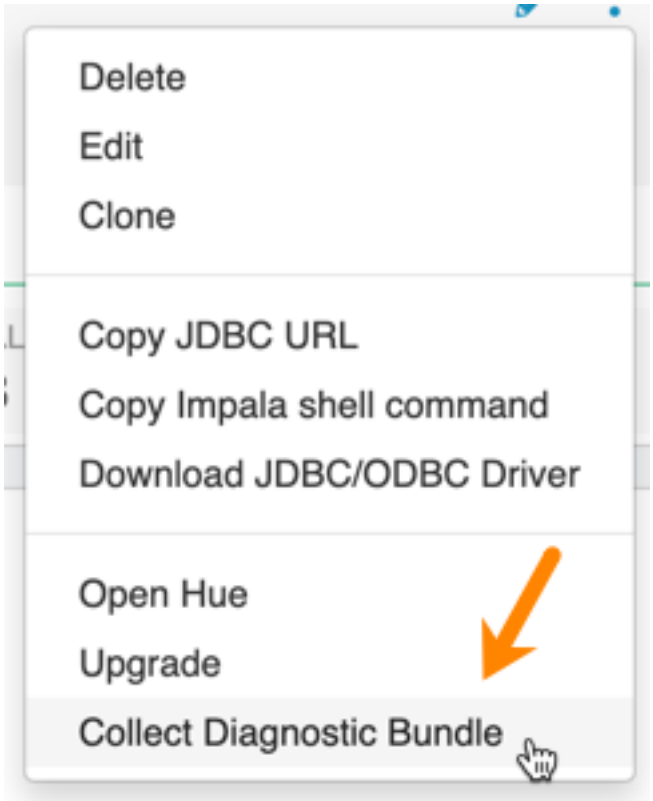

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2023-719: Cloudera Data Warehouse Backup/Restore of CDP Data Visualization incomplete

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature will encounter an issue with the restoration versions of Cloudera Data Visualization (CDV) older than CDV 7.1.6.2-3 due to a schema change in this release. If the Backup was taken from an older CDW environment that contained a version of CDV older than CDV 7.1.6.2-3, the Restore procedure will succeed. Though once the user opens the CDV Queries tab, the user could encounter the error message: “column jobs_jobschedule.owner_id does not exist…”

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete.

TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature might encounter an issue with the restoration of the Database Catalog, if the hue-query-processor.json configuration file of the Database Catalog has been edited. Even a minor edit to the hue-query-processor.json configuration file can result in this failure.

During the restoration process the Database Catalog will be created, but it will fail to start after a short period of time, and the Database Catalog will be in a Bad Health state on the CDW User Interface.

Inside the Kubernetes Cluster (Azure Kubernetes Service on Azure / Elastic Kubernetes Service on AWS) the StatefulSet of Hue Query Processor (hue-query-processor) is in CrashLoop. This is indicated with the following log in the Hue Query Processor StatefulSet Pod:

```
SQL State  : 3D000
Error Code : 0
Message    : FATAL: database "warehouse-1707832123-abcd_hueqpdb"
 does not exist



              at org.flywaydb.core.internal.jdbc.JdbcUtils.openC
onnection(JdbcUtils.java:65)
```

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW.

TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW

Cloudera Data Warehouse (CDW) customers using Apache Impala (Impala) to read Apache Iceberg (Iceberg) V2 tables can encounter an issue of Impala returning incorrect results when the optimized V2 operator is used. The optimized V2 operator is enabled by default in the affected versions below. The issue only affects Iceberg V2 tables that have position delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW.

TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

Apache Hive (Hive) and Apache Impala (Impala) modify statements (DELETE/UPDATE/MERGE) on Apache Iceberg (Iceberg) V2 tables can corrupt the tables if there is a concurrent table compaction from Apache Spark. The issue happens when the compaction and modify statement run in parallel, and when the compaction job commits before the modify statement. In this case the

position delete files of the modify statement still point to the old files. This means the following in case of

- DELETE statements
  - Deleting records pointing to old files have no effect
- UPDATE / MERGE statements
  - Deleting records pointing to old files have no effect
  - The table will also have the newly added data records
  - Rewritten records will still be active

This issue does not affect Apache NiFi (NiFi) and Apache Flink (Flink) as these components write equality delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables.

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

When working with Apache Hive (Hive) and Apache Iceberg (Iceberg) V2 tables, using the TRUNCATE statement may lead to unintended data deletion. This issue arises when the truncate

command is applied to a branch of an Iceberg table. Instead of truncating the branch itself, the command affects the original (main) table, which results in unintended loss of data.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

November 20, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New Known Issues in this release

See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.6.

Compaction causes data loss under certain conditions

Data loss occurs during compaction when Apache Ranger policies for masking or row filtering are enabled and compaction users are included in the policies.

Workaround: Exclude compaction users from the policies as described in Compaction Prerequisites.

Limited Hive image versions

Hive Virtual Warehouses you create in 1.8.1-b248 (released Nov 20, 2023) will run Istio 1.19.0. The new Istio version supports only new versions of Hive helm charts. If you have the CDW_VERSIONED_DEPLOY, only new Hive image versions appear in UI when you create a new Hive Virtual Warehouse.

DWX-16899: Error while viewing Impala job status on the mini Job Browser

When you click on the application ID after submitting an Impala query on Hue that is running on the environment level, you may notice the following error: 401 Client Error: Unauthorized for url: http://coordinator.impala-xyz.svc.cluster.local:25000/queries?json=true Must authenticate with Basic authentication. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16913: “Results have expired” message while running a CTAS query

You may see the following message on the Hue UI when you submit a CREATE TABLE AS SELECT (CTAS) query from a Hue instance that is deployed on the environment level: “Results have expired, rerun the query if needed”. This does not happen when you do the same from Hue deployed at a Virtual Warehouse level.

None.

DWX-16917: Failed to validate proxy privileges error while running queries from Hue

You may see the following error intermittently in Hue’s pod logs while running queries from a Hue instance that is deployed at the environment level: “Failed to validate proxy privileges of <username>”.

None.

DWX-16895: Incorrect status of Hue pods when you edit the Hue instance properties

When you update a configuration of a Hue instance that is deployed at the environment level, such as increasing or decreasing the size of the Hue instance, you see a success message on the CDW UI. After some time, the status of the Hue instance also changes from “Updating” to “Running”. However, when you list the Hue pods using kubectl, you see that not all backend pods are in the running state–a few of them are still in the init state.

None. The pods come up successfully eventually after a sufficient time has passed.

DWX-16863: The upgrade button is present on the CDW UI, but Hue upgrades are not supported

You see the Upgrade button on the **Query Editor** page in the CDW UI when Hue is deployed at the environment level. However, on CDW version 1.8.1, upgrading the Hue instance that is deployed at the environment level is not supported.

None.

DWX-16893: A user can see all the queries in Job browser

In a Hue instance deployed at the environment level, by design, the Hue instances must not share the saved queries and query history with other Hue instances even for the same user. However, a logged in user is able to view all the queries executed by that user on all the Virtual Warehouses on a particular Database Catalog.

None.

CDW backup and restore Data Visualization issue

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

TSB 2023-719: Cloudera Data Warehouse Backup/Restore of CDP Data Visualization incomplete

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature will encounter an issue with the restoration versions of Cloudera Data Visualization (CDV) older than CDV 7.1.6.2-3 due to a schema change in this release. If the Backup was taken from an older CDW environment that contained a version of CDV older than CDV 7.1.6.2-3, the Restore

procedure will succeed. Though once the user opens the CDV Queries tab, the user could encounter the error message: “column jobs_jobschedule.owner_id does not exist…”

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-719: Cloudera Data Warehouse Backup/Restore of Cloudera Data Visualization incomplete

Carried over from the previous release: Upgrade-related

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

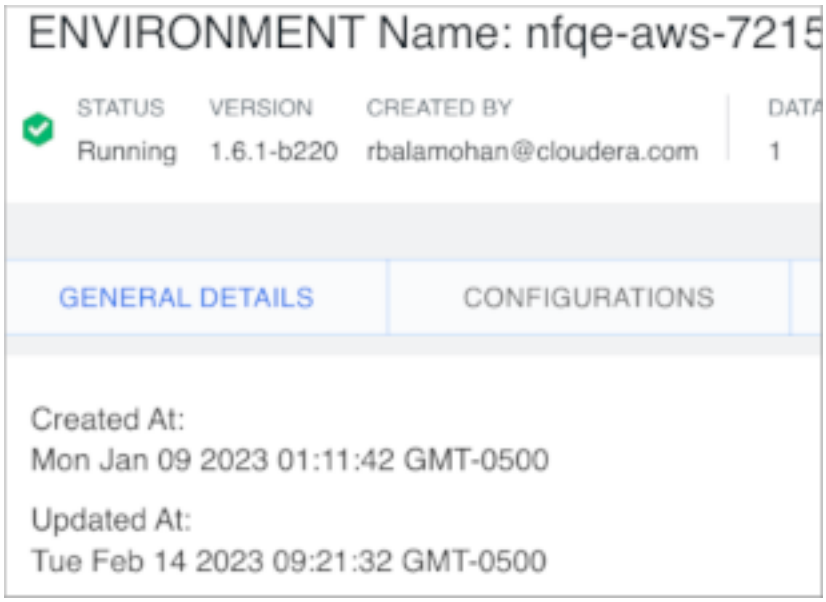

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

```
worker subnets should be selected from 3 different
availability zones. got: 0
```

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

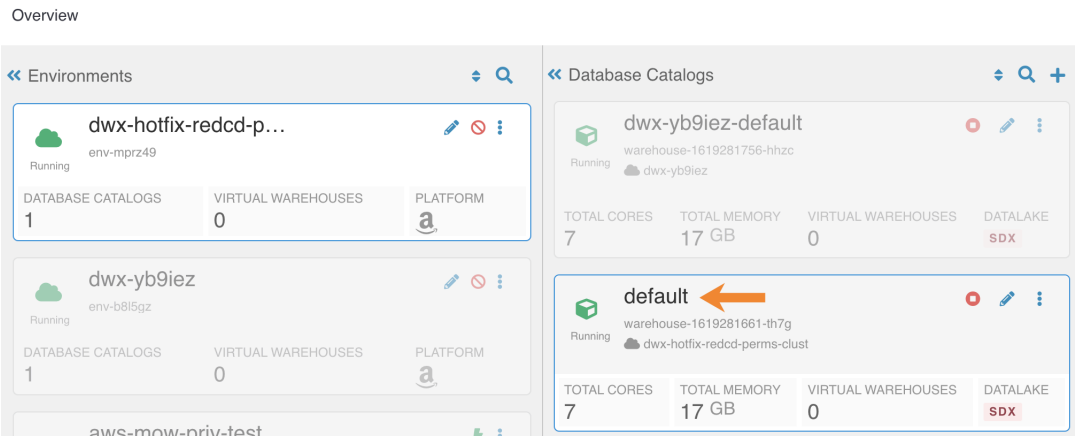

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

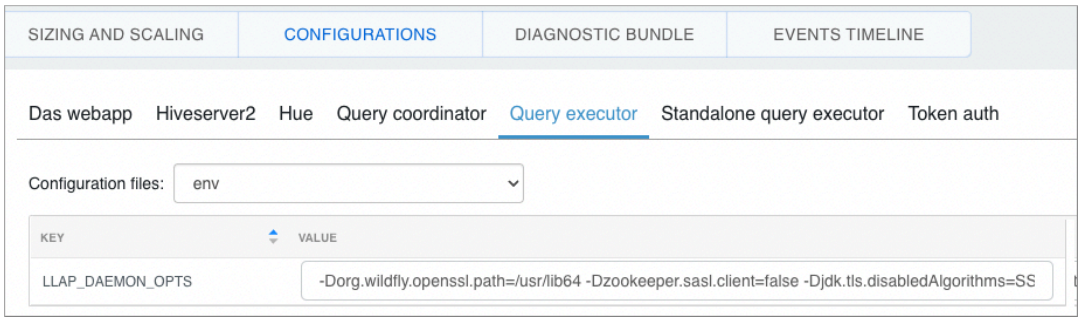

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
```

```
        - apps
        resources:
        - deployments
        - statefulsets
        verbs:
        - get
        - list
        - apiGroups:
        - "edws.cloudera.com"
        resources:
        - computes
        verbs:
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is

a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

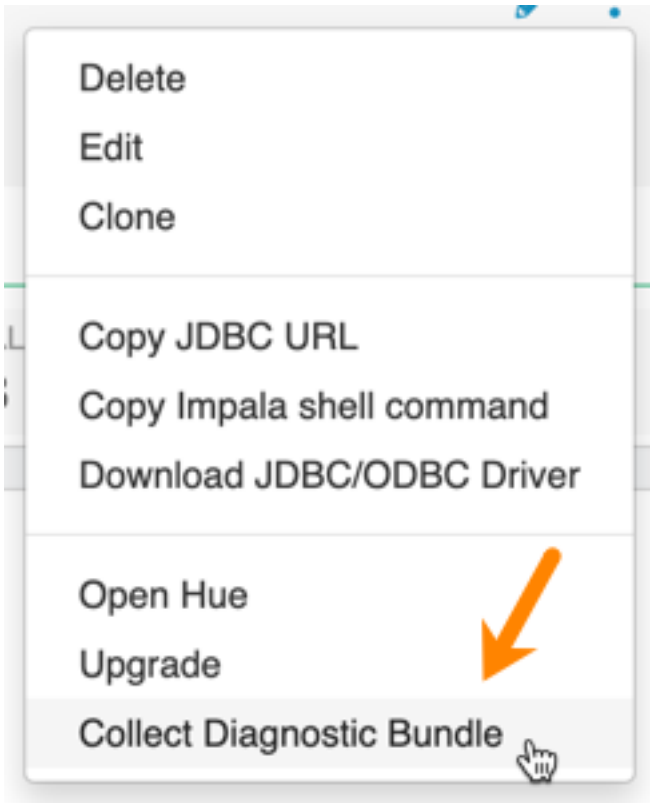

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2024-723: Hue RAZ is using logger role to Read and Upload/Delete (write) files

When using Cloudera Data Hub for Public Cloud (Data Hub) on Amazon Web Services (AWS), users can use the Hue File Browser feature to access the filesystem, and if permitted, read and write directly to the related S3 buckets. As AWS does not provide fine-grained access control, Cloudera Data Platform administrators can use the Ranger Authorization Service (RAZ) capability to take the S3 filesystem, and overlay it with user and group specific permissions, making it easier to allow certain users to have limited permissions, without having to grant those users permissions to the entire S3 bucket.

This bulletin describes an issue when using RAZ with Data Hub, and attempting to use fine-grained access control to allow certain users write permissions.

Through RAZ, an administrator may, for a particular user, specify permissions more limited than what AWS provides for an S3 bucket, allowing the user to have read/write (or other similar fine grained access) permissions on only a subset of the files and directories within that bucket. However, under specific conditions, it is possible for such user to be able to read and write to the entire S3 bucket through Hue, due to Hue using the logger role (which will have full read/write to the S3 bucket) when using Data Hub with a RAZ enabled cluster. This problem also can affect the Hue service itself, by affecting proper access to home directories causing the service role to not start.

The root cause of this issue is, when accessing Amazon cloud resources, Hue uses the AWS Boto SDK library. This AWS Boto library has a bug that restricts permissions in certain AWS regions in such a way that it provides access to users who should not have it, regardless of RAZ settings. This issue only affects users in specific AWS regions, listed below, and it does not affect all AWS customers.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-723: Hue Raz is using logger role to Read and Upload/Delete (write) files.

TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature might encounter an issue with the restoration of the Database Catalog, if the hue-query-processor.json configuration file of the Database Catalog has been edited. Even a minor edit to the hue-query-processor.json configuration file can result in this failure.

During the restoration process the Database Catalog will be created, but it will fail to start after a short period of time, and the Database Catalog will be in a Bad Health state on the CDW User Interface.

Inside the Kubernetes Cluster (Azure Kubernetes Service on Azure / Elastic Kubernetes Service on AWS) the StatefulSet of Hue Query Processor (hue-query-processor) is in CrashLoop. This is indicated with the following log in the Hue Query Processor StatefulSet Pod:

```
SQL State  : 3D000
Error Code : 0
Message    : FATAL: database "warehouse-1707832123-abcd_hueqpdb"
 does not exist



              at org.flywaydb.core.internal.jdbc.JdbcUtils.openC
onnection(JdbcUtils.java:65)
```

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW.

TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW

Cloudera Data Warehouse (CDW) customers using Apache Impala (Impala) to read Apache Iceberg (Iceberg) V2 tables can encounter an issue of Impala returning incorrect results when the optimized V2 operator is used. The optimized V2 operator is enabled by default in the affected versions below. The issue only affects Iceberg V2 tables that have position delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW.

TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

Apache Hive (Hive) and Apache Impala (Impala) modify statements (DELETE/UPDATE/MERGE) on Apache Iceberg (Iceberg) V2 tables can corrupt the tables if there is a concurrent table compaction from Apache Spark. The issue happens when the compaction and modify statement run in parallel, and when the compaction job commits before the modify statement. In this case the position delete files of the modify statement still point to the old files. This means the following in case of

- DELETE statements
  - Deleting records pointing to old files have no effect
- UPDATE / MERGE statements
  - Deleting records pointing to old files have no effect
  - The table will also have the newly added data records
  - Rewritten records will still be active

This issue does not affect Apache NiFi (NiFi) and Apache Flink (Flink) as these components write equality delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables.

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

When working with Apache Hive (Hive) and Apache Iceberg (Iceberg) V2 tables, using the TRUNCATE statement may lead to unintended data deletion. This issue arises when the truncate command is applied to a branch of an Iceberg table. Instead of truncating the branch itself, the command affects the original (main) table, which results in unintended loss of data.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

October 16, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New Known Issues in this release

See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.3.

Carried over from the previous release: Upgrade-related

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

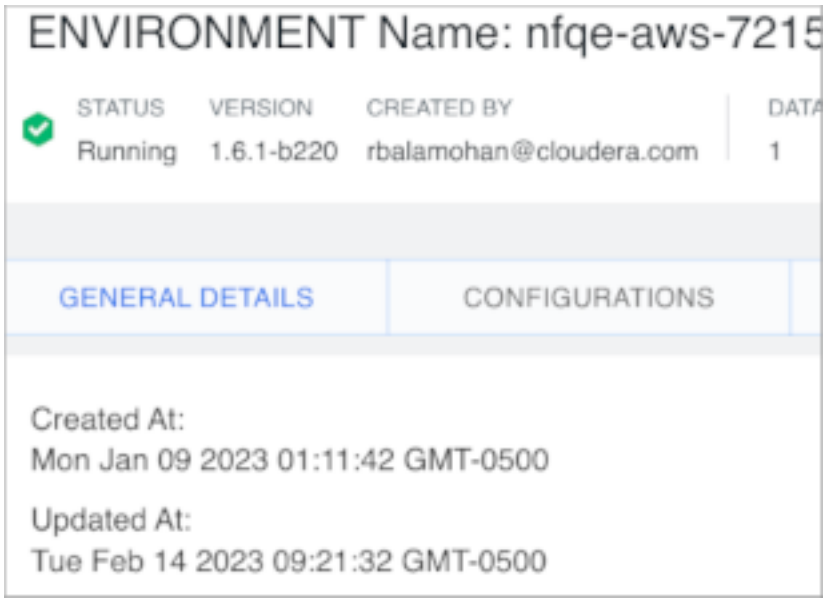

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

Compaction causes data loss under certain conditions

Data loss occurs during compaction when Apache Ranger policies for masking or row filtering are enabled and compaction users are included in the policies.

Workaround: Exclude compaction users from the policies as described in Compaction Prerequisites.

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

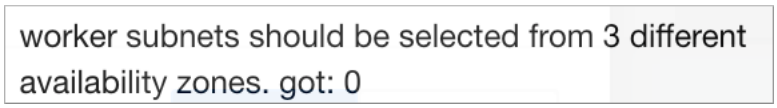

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

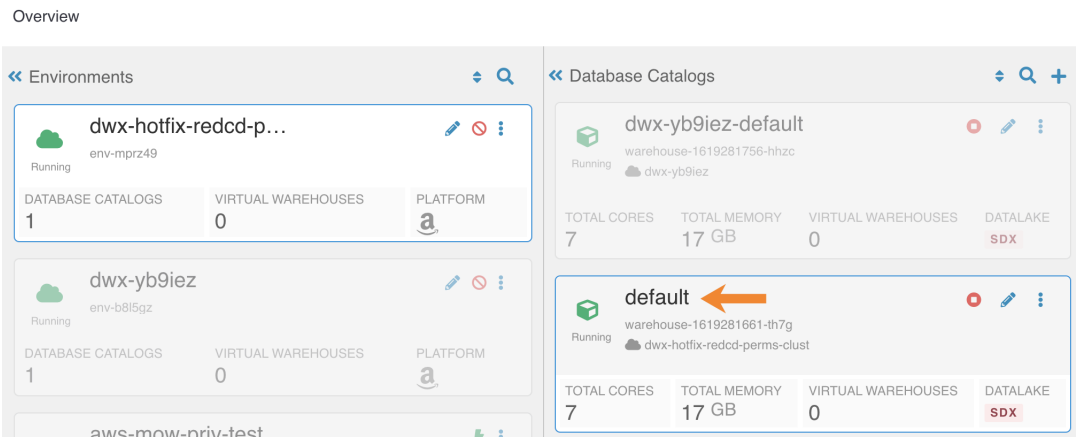

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

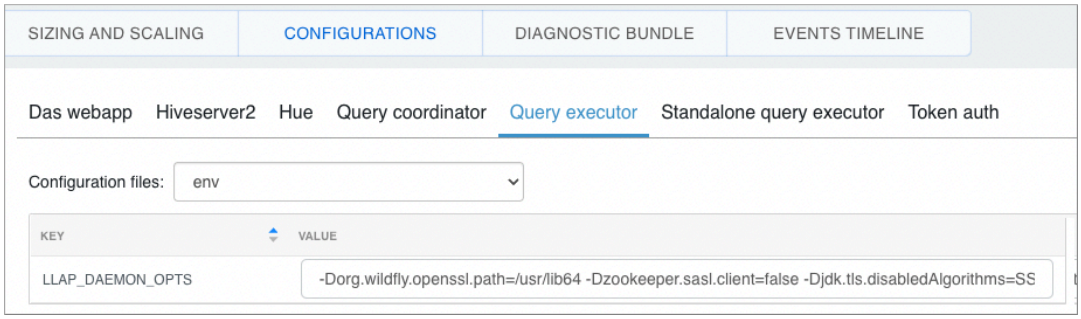

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
       - apps
       resources:
       - deployments
       - statefulsets
       verbs:
       - get
       - list
       - apiGroups:
       - "edws.cloudera.com"
```

```
        resources:
        - computes
        verbs:
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.

2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
```

```
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

MT_DOP related issue

Problem:Impala query with JOIN and LIMIT can hang when MT_DOP is greater than 0.

Workaround: Set MT_DOP to zero.

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml
```

```
        # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

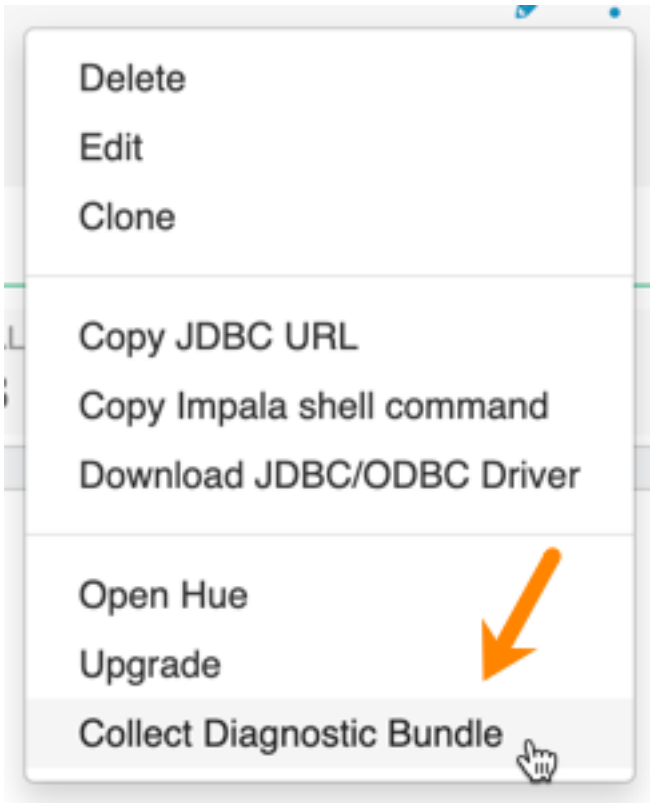

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2024-723: Hue RAZ is using logger role to Read and Upload/Delete (write) files

When using Cloudera Data Hub for Public Cloud (Data Hub) on Amazon Web Services (AWS), users can use the Hue File Browser feature to access the filesystem, and if permitted, read and write

directly to the related S3 buckets. As AWS does not provide fine-grained access control, Cloudera Data Platform administrators can use the Ranger Authorization Service (RAZ) capability to take the S3 filesystem, and overlay it with user and group specific permissions, making it easier to allow certain users to have limited permissions, without having to grant those users permissions to the entire S3 bucket.

This bulletin describes an issue when using RAZ with Data Hub, and attempting to use fine-grained access control to allow certain users write permissions.

Through RAZ, an administrator may, for a particular user, specify permissions more limited than what AWS provides for an S3 bucket, allowing the user to have read/write (or other similar fine grained access) permissions on only a subset of the files and directories within that bucket. However, under specific conditions, it is possible for such user to be able to read and write to the entire S3 bucket through Hue, due to Hue using the logger role (which will have full read/write to the S3 bucket) when using Data Hub with a RAZ enabled cluster. This problem also can affect the Hue service itself, by affecting proper access to home directories causing the service role to not start.

The root cause of this issue is, when accessing Amazon cloud resources, Hue uses the AWS Boto SDK library. This AWS Boto library has a bug that restricts permissions in certain AWS regions in such a way that it provides access to users who should not have it, regardless of RAZ settings. This issue only affects users in specific AWS regions, listed below, and it does not affect all AWS customers.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-723: Hue Raz is using logger role to Read and Upload/Delete (write) files.

TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature might encounter an issue with the restoration of the Database Catalog, if the hue-query-processor.json configuration file of the Database Catalog has been edited. Even a minor edit to the hue-query-processor.json configuration file can result in this failure.

During the restoration process the Database Catalog will be created, but it will fail to start after a short period of time, and the Database Catalog will be in a Bad Health state on the CDW User Interface.

Inside the Kubernetes Cluster (Azure Kubernetes Service on Azure / Elastic Kubernetes Service on AWS) the StatefulSet of Hue Query Processor (hue-query-processor) is in CrashLoop. This is indicated with the following log in the Hue Query Processor StatefulSet Pod:

```
SQL State  : 3D000
Error Code : 0
Message    : FATAL: database "warehouse-1707832123-abcd_hueqpdb"
 does not exist



              at org.flywaydb.core.internal.jdbc.JdbcUtils.openC
onnection(JdbcUtils.java:65)
```

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW.

TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW

Cloudera Data Warehouse (CDW) customers using Apache Impala (Impala) to read Apache Iceberg (Iceberg) V2 tables can encounter an issue of Impala returning incorrect results when the optimized V2 operator is used. The optimized V2 operator is enabled by default in the affected versions below. The issue only affects Iceberg V2 tables that have position delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW.

TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

Apache Hive (Hive) and Apache Impala (Impala) modify statements (DELETE/UPDATE/MERGE) on Apache Iceberg (Iceberg) V2 tables can corrupt the tables if there is a concurrent table compaction from Apache Spark. The issue happens when the compaction and modify statement run in parallel, and when the compaction job commits before the modify statement. In this case the position delete files of the modify statement still point to the old files. This means the following in case of

- DELETE statements
  - Deleting records pointing to old files have no effect
- UPDATE / MERGE statements
  - Deleting records pointing to old files have no effect
  - The table will also have the newly added data records
  - Rewritten records will still be active

This issue does not affect Apache NiFi (NiFi) and Apache Flink (Flink) as these components write equality delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables.

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table

- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

October 12, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New Known Issues in this release

See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.3.

Carried over from the previous release: Upgrade-related

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

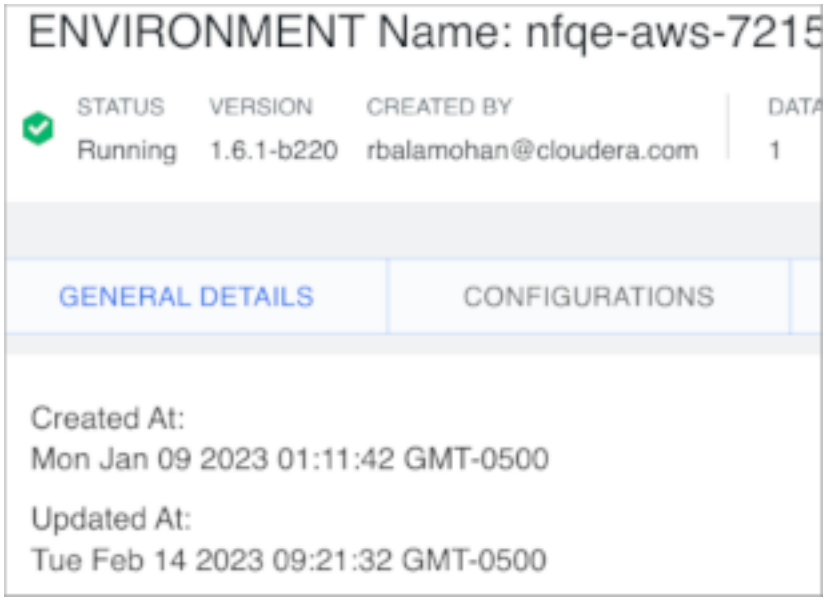

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

Compaction causes data loss under certain conditions

Data loss occurs during compaction when Apache Ranger policies for masking or row filtering are enabled and compaction users are included in the policies.

Workaround: Exclude compaction users from the policies as described in Compaction Prerequisites.

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

worker subnets should be selected from 3 different availability zones. got: 0

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
```

```
        Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses

old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

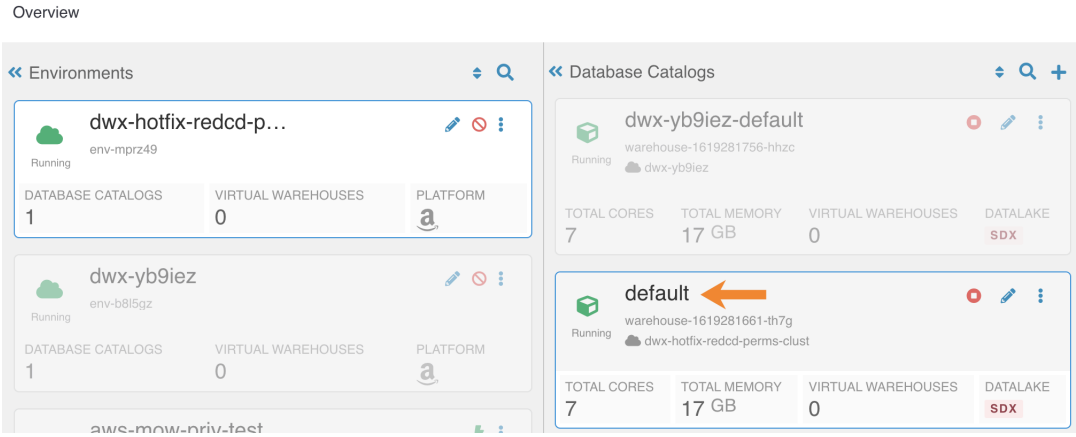

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

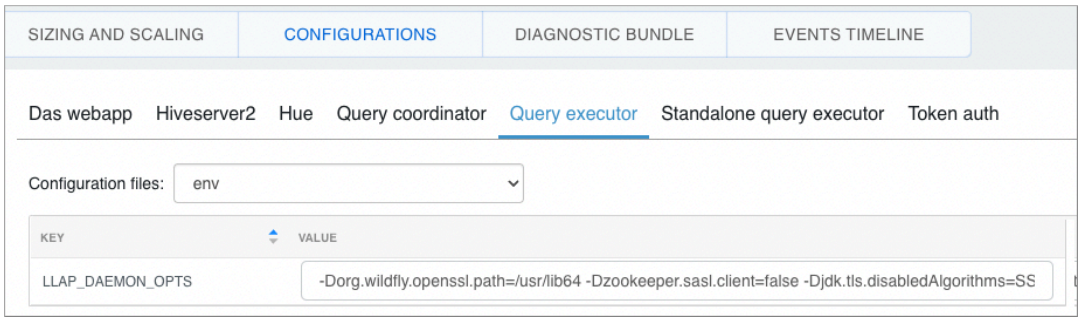

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
```

```
        - get
        - list
        - apiGroups:
        - apps
        resources:
        - deployments
        - statefulsets
        verbs:
        - get
        - list
        - apiGroups:
        - "edws.cloudera.com"
        resources:
        - computes
        verbs:
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is

a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

MT_DOP related issue

Problem:Impala query with JOIN and LIMIT can hang when MT_DOP is greater than 0.

Workaround: Set MT_DOP to zero.

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

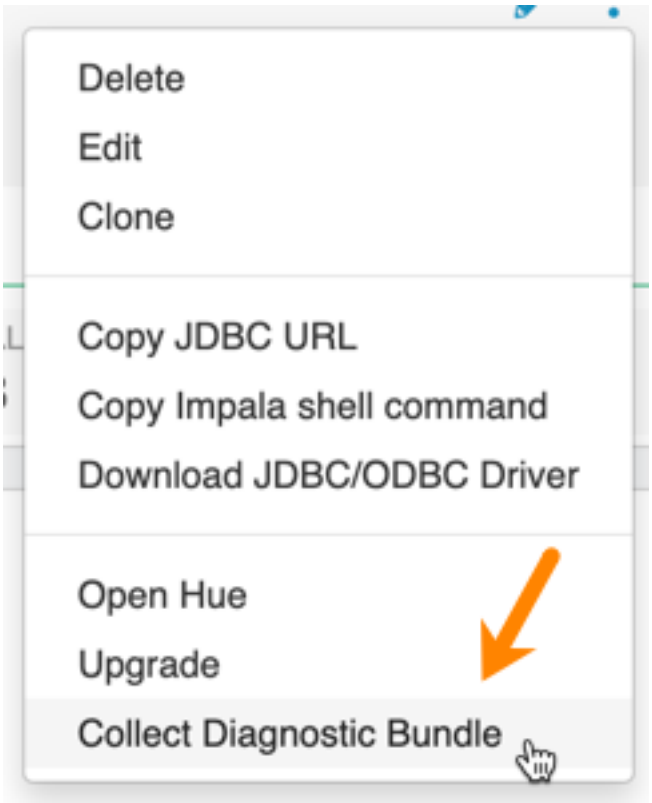

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2024-723: Hue RAZ is using logger role to Read and Upload/Delete (write) files

When using Cloudera Data Hub for Public Cloud (Data Hub) on Amazon Web Services (AWS), users can use the Hue File Browser feature to access the filesystem, and if permitted, read and write directly to the related S3 buckets. As AWS does not provide fine-grained access control, Cloudera Data Platform administrators can use the Ranger Authorization Service (RAZ) capability to take the S3 filesystem, and overlay it with user and group specific permissions, making it easier to allow certain users to have limited permissions, without having to grant those users permissions to the entire S3 bucket.

This bulletin describes an issue when using RAZ with Data Hub, and attempting to use fine-grained access control to allow certain users write permissions.

Through RAZ, an administrator may, for a particular user, specify permissions more limited than what AWS provides for an S3 bucket, allowing the user to have read/write (or other similar fine grained access) permissions on only a subset of the files and directories within that bucket. However, under specific conditions, it is possible for such user to be able to read and write to the entire S3 bucket through Hue, due to Hue using the logger role (which will have full read/write to the S3 bucket) when using Data Hub with a RAZ enabled cluster. This problem also can affect the Hue service itself, by affecting proper access to home directories causing the service role to not start.

The root cause of this issue is, when accessing Amazon cloud resources, Hue uses the AWS Boto SDK library. This AWS Boto library has a bug that restricts permissions in certain AWS regions in such a way that it provides access to users who should not have it, regardless of RAZ settings. This issue only affects users in specific AWS regions, listed below, and it does not affect all AWS customers.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-723: Hue Raz is using logger role to Read and Upload/Delete (write) files.

TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature might encounter an issue with the restoration of the Database Catalog, if the hue-query-processor.json configuration file of the Database Catalog has been edited. Even a minor edit to the hue-query-processor.json configuration file can result in this failure.

During the restoration process the Database Catalog will be created, but it will fail to start after a short period of time, and the Database Catalog will be in a Bad Health state on the CDW User Interface.

Inside the Kubernetes Cluster (Azure Kubernetes Service on Azure / Elastic Kubernetes Service on AWS) the StatefulSet of Hue Query Processor (hue-query-processor) is in CrashLoop. This is indicated with the following log in the Hue Query Processor StatefulSet Pod:

```
SQL State  : 3D000
Error Code : 0
Message    : FATAL: database "warehouse-1707832123-abcd_hueqpdb"
 does not exist



              at org.flywaydb.core.internal.jdbc.JdbcUtils.openC
onnection(JdbcUtils.java:65)
```

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW.

TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

Apache Hive (Hive) and Apache Impala (Impala) modify statements (DELETE/UPDATE/MERGE) on Apache Iceberg (Iceberg) V2 tables can corrupt the tables if there is a concurrent table compaction from Apache Spark. The issue happens when the compaction and modify statement run in parallel, and when the compaction job commits before the modify statement. In this case the position delete files of the modify statement still point to the old files. This means the following in case of

- DELETE statements
  - Deleting records pointing to old files have no effect
- UPDATE / MERGE statements
  - Deleting records pointing to old files have no effect
  - The table will also have the newly added data records
  - Rewritten records will still be active

This issue does not affect Apache NiFi (NiFi) and Apache Flink (Flink) as these components write equality delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables.

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

When working with Apache Hive (Hive) and Apache Iceberg (Iceberg) V2 tables, using the TRUNCATE statement may lead to unintended data deletion. This issue arises when the truncate command is applied to a branch of an Iceberg table. Instead of truncating the branch itself, the command affects the original (main) table, which results in unintended loss of data.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

October 5, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New Known Issues in this release

There are no new known issues in this release for CDW. See Data Visualization release notes for known issues in Cloudera Data Visualization 7.1.3.

Carried over from the previous release: Upgrade-related

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouse from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023) or later.

Issue backing up and restoring Workload Aware Auto-Scaling (WAAS)

You cannot use the CDW backup and restore process to restore the WAAS configurations.

Workaround: Recreate your Virtual Warehouse manually setting the right configurations.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

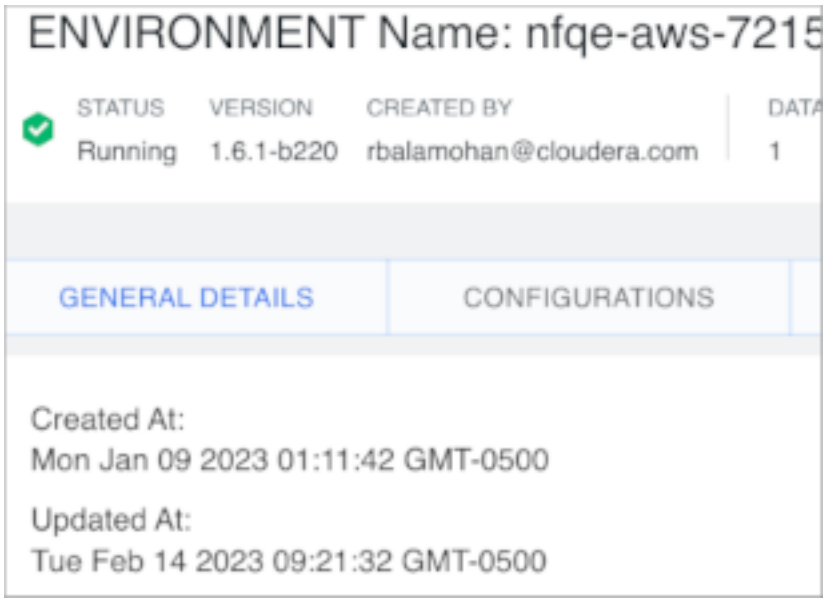

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

Compaction causes data loss under certain conditions

Data loss occurs during compaction when Apache Ranger policies for masking or row filtering are enabled and compaction users are included in the policies.

Workaround: Exclude compaction users from the policies as described in Compaction Prerequisites.

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

```
worker subnets should be selected from 3 different
availability zones. got: 0
```

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle. Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

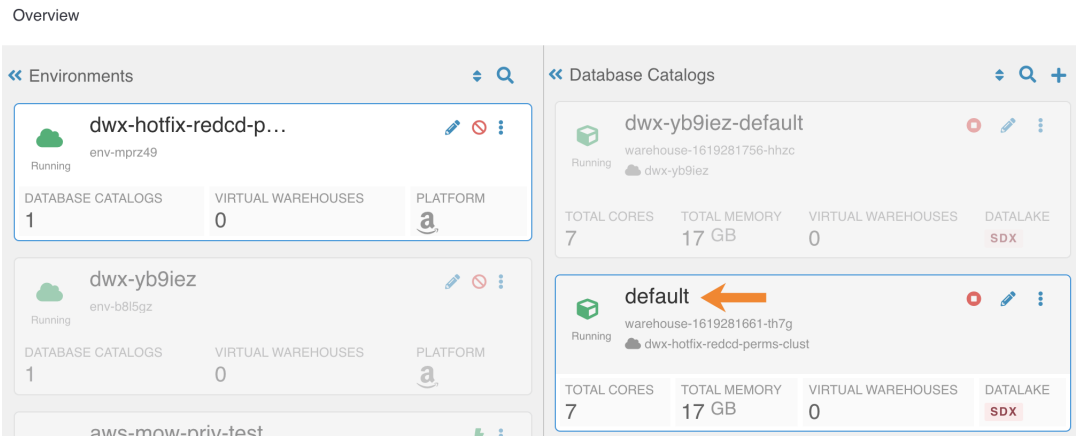

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

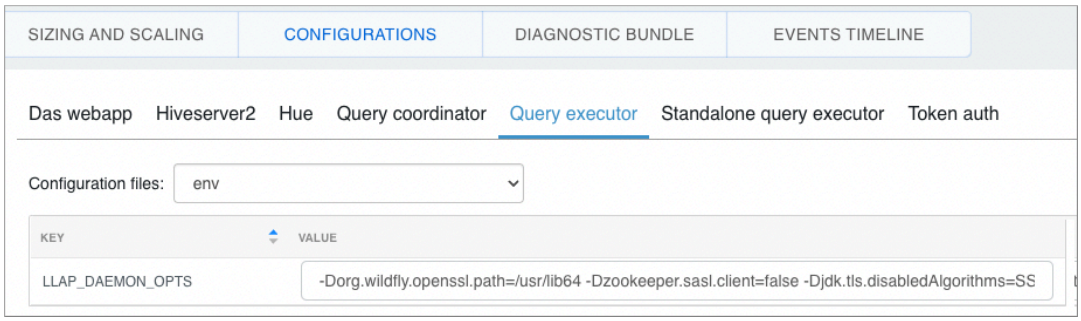

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
```

```
        - pods
        - persistentvolumeclaims
        - nodes
        verbs:
        - get
        - list
        - apiGroups:
        - apps
        resources:
        - deployments
        - statefulsets
        verbs:
        - get
        - list
        - apiGroups:
        - "edws.cloudera.com"
        resources:
        - computes
        verbs:
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is

a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

MT_DOP related issue

Problem:Impala query with JOIN and LIMIT can hang when MT_DOP is greater than 0.

Workaround: Set MT_DOP to zero.

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

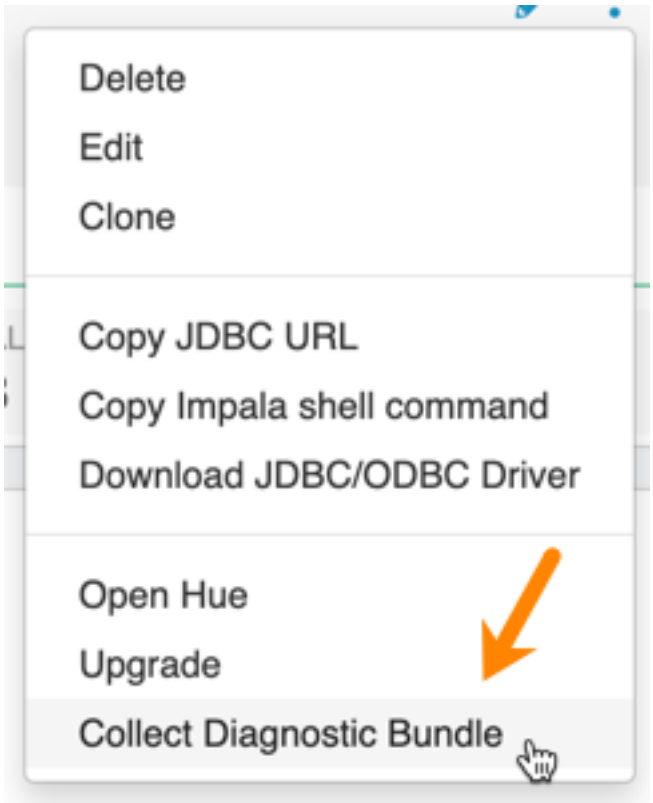

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2024-723: Hue RAZ is using logger role to Read and Upload/Delete (write) files

When using Cloudera Data Hub for Public Cloud (Data Hub) on Amazon Web Services (AWS), users can use the Hue File Browser feature to access the filesystem, and if permitted, read and write directly to the related S3 buckets. As AWS does not provide fine-grained access control, Cloudera Data Platform administrators can use the Ranger Authorization Service (RAZ) capability to take the S3 filesystem, and overlay it with user and group specific permissions, making it easier to allow certain users to have limited permissions, without having to grant those users permissions to the entire S3 bucket.

This bulletin describes an issue when using RAZ with Data Hub, and attempting to use fine-grained access control to allow certain users write permissions.

Through RAZ, an administrator may, for a particular user, specify permissions more limited than what AWS provides for an S3 bucket, allowing the user to have read/write (or other similar fine grained access) permissions on only a subset of the files and directories within that bucket. However, under specific conditions, it is possible for such user to be able to read and write to the entire S3 bucket through Hue, due to Hue using the logger role (which will have full read/write to the S3 bucket) when using Data Hub with a RAZ enabled cluster. This problem also can affect the Hue service itself, by affecting proper access to home directories causing the service role to not start.

The root cause of this issue is, when accessing Amazon cloud resources, Hue uses the AWS Boto SDK library. This AWS Boto library has a bug that restricts permissions in certain AWS regions in such a way that it provides access to users who should not have it, regardless of RAZ settings. This issue only affects users in specific AWS regions, listed below, and it does not affect all AWS customers.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-723: Hue Raz is using logger role to Read and Upload/Delete (write) files.

TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW

Cloudera Data Warehouse (CDW) customers using the CDW Automated Backup/Restore feature might encounter an issue with the restoration of the Database Catalog, if the hue-query-processor.json configuration file of the Database Catalog has been edited. Even a minor edit to the hue-query-processor.json configuration file can result in this failure.

During the restoration process the Database Catalog will be created, but it will fail to start after a short period of time, and the Database Catalog will be in a Bad Health state on the CDW User Interface.

Inside the Kubernetes Cluster (Azure Kubernetes Service on Azure / Elastic Kubernetes Service on AWS) the StatefulSet of Hue Query Processor (hue-query-processor) is in CrashLoop. This is indicated with the following log in the Hue Query Processor StatefulSet Pod:

```
SQL State  : 3D000
Error Code : 0
Message    : FATAL: database "warehouse-1707832123-abcd_hueqpdb"
 does not exist



              at org.flywaydb.core.internal.jdbc.JdbcUtils.openC
onnection(JdbcUtils.java:65)
```

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-743: Restoration of the Database Catalog can be incomplete due to an incorrect Hue Query Processor configuration in CDW.

TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW

Cloudera Data Warehouse (CDW) customers using Apache Impala (Impala) to read Apache Iceberg (Iceberg) V2 tables can encounter an issue of Impala returning incorrect results when the optimized V2 operator is used. The optimized V2 operator is enabled by default in the affected versions below. The issue only affects Iceberg V2 tables that have position delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-745: Impala returns incorrect results for Iceberg V2 tables when optimized operator is being used in CDW.

TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables

Apache Hive (Hive) and Apache Impala (Impala) modify statements (DELETE/UPDATE/MERGE) on Apache Iceberg (Iceberg) V2 tables can corrupt the tables if there is a concurrent table compaction from Apache Spark. The issue happens when the compaction and modify statement run in parallel, and when the compaction job commits before the modify statement. In this case the position delete files of the modify statement still point to the old files. This means the following in case of

- DELETE statements
  - Deleting records pointing to old files have no effect
- UPDATE / MERGE statements
  - Deleting records pointing to old files have no effect
  - The table will also have the newly added data records
  - Rewritten records will still be active

This issue does not affect Apache NiFi (NiFi) and Apache Flink (Flink) as these components write equality delete files.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-746: Concurrent compactions and modify statements can corrupt Iceberg tables.

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

When working with Apache Hive (Hive) and Apache Iceberg (Iceberg) V2 tables, using the TRUNCATE statement may lead to unintended data deletion. This issue arises when the truncate command is applied to a branch of an Iceberg table. Instead of truncating the branch itself, the command affects the original (main) table, which results in unintended loss of data.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

August 30, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New Known Issues in this release

Compaction causes data loss under certain conditions

Data loss occurs during compaction when Apache Ranger policies for masking or row filtering are enabled and compaction users are included in the policies.

Workaround: Exclude compaction users from the policies as described in Compaction Prerequisites.

MT_DOP related issue

Problem:Impala query with JOIN and LIMIT can hang when MT_DOP is greater than 0.

Workaround: Set MT_DOP to zero.

Impala Web UI incompatibility

In releases 1.6.5-b10 (released June 29, 2023) and earlier, when catalogd runs in HA mode there is single url for the Web UI. In this release 1.7.1-b755 (released August 30, 2023), when catalogd runs in HA mode there are two urls. A -0 appears in one URL, and -1 appears in the other. For example:

```
catalogd-web-0-impala-nonha.dw-dwx-9abs9a.svbr-nqvp.int.cldr.work
catalogd-web-1-impala-nonha.dw-dwx-9abs9a.svbr-nqvp.int.cldr.work
```

Workaround: Remove the -0 or the -1 from the URL.

Upgrading to EKS 1.24 could result in Impala coordinators shutting down.

This issue is not seen on the Impala Virtual Warehouse running Runtime 2023.0.15.0-x or later.

Workaround: Manually start the Impala Virtual Warehouose from the UI or cli. Alternatively, replace your runtime with 1.7.1-b755 (released August 30, 2023).

Carried over from the previous release: Upgrade-related

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

Issue backing up and restoring Workload Aware Auto-Scaling (WAAS)

You cannot use the CDW backup and restore process to restore the WAAS configurations.

Workaround: Recreate your Virtual Warehouse manually setting the right configurations.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

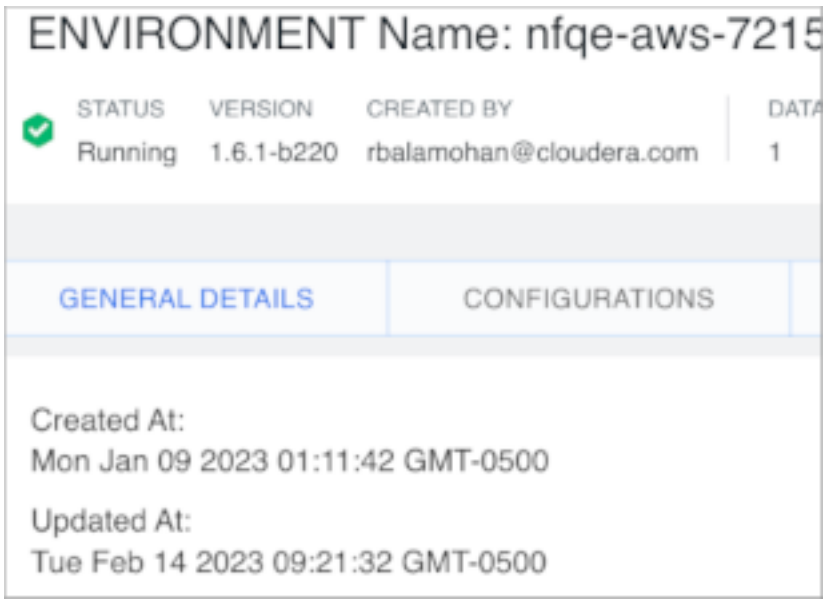

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

```
worker subnets should be selected from 3 different
availability zones. got: 0
```

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data

Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the Cr eateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected: CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 or later if they upgrade the node image

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle. Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

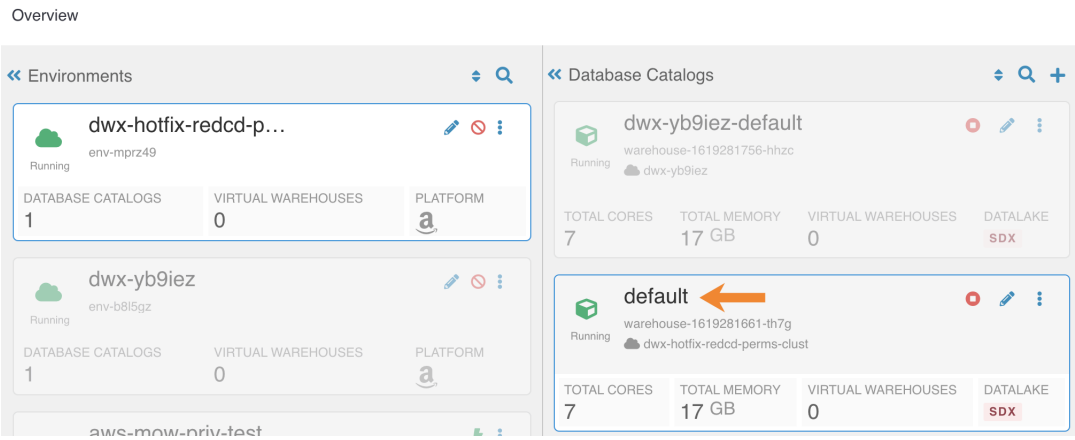

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

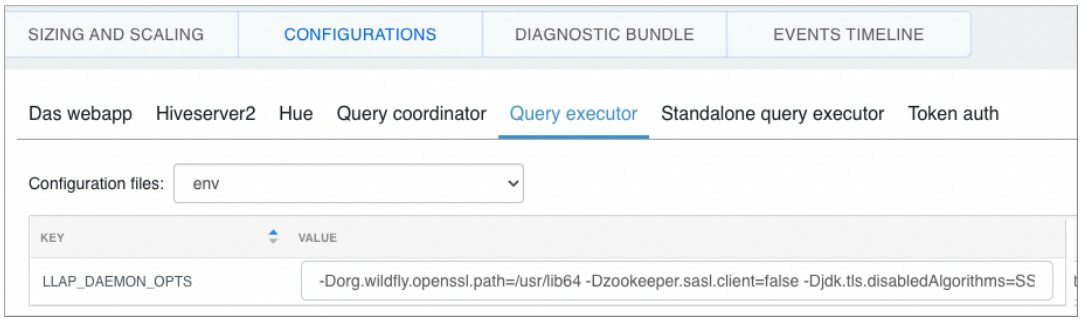

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
       - apps
       resources:
       - deployments
       - statefulsets
       verbs:
       - get
       - list
       - apiGroups:
       - "edws.cloudera.com"
       resources:
       - computes
       verbs:
       - get
       - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

Concurrent compactions and modify statements can corrupt Iceberg tables

Hive or Impala DELETE/UPDATE/MERGE operations on Iceberg V2 tables can corrupt the tables if there is a concurrent table compaction from Spark. The issue happens if the compaction and modify statement runs in parallel, and if the compaction job commits before the modify statement. In that case the modify statement's position delete files still point to the old files. The results in the case of DELETE and in the case of UPDATE / MERGE are as follows:

- DELETE

  Delete records pointing to old files have no effect.
- UPDATE / MERGE

  Delete records pointing to old files have no effect. The table will also have the newly added data records, which means rewritten records will still be active.

Use one of the following workarounds:

- Do not run compactions and DELETE/UPDATE/MERGE statements in parallel.
- Do not compact the table via Iceberg's RewriteFiles operation. For example do not use Spark's rewriteDataFiles.

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml
```

```
        # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

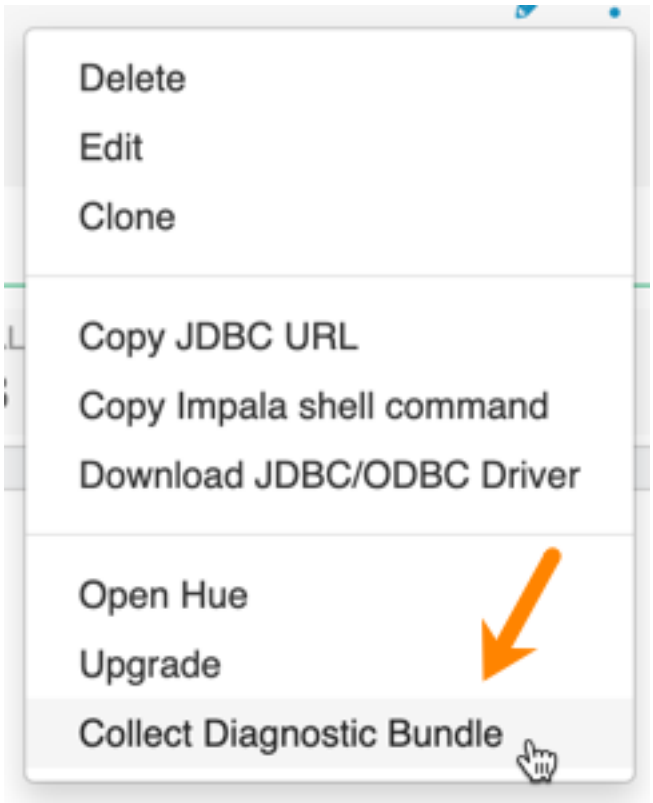

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

When working with Apache Hive (Hive) and Apache Iceberg (Iceberg) V2 tables, using the TRUNCATE statement may lead to unintended data deletion. This issue arises when the truncate command is applied to a branch of an Iceberg table. Instead of truncating the branch itself, the command affects the original (main) table, which results in unintended loss of data.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-758: Truncate command on Iceberg V2 branches cause unintentional data deletion

June 29, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New Known Issues in this release

Issue backing up and restoring Workload Aware Auto-Scaling (WAAS)

You cannot use the CDW backup and restore process to restore the WAAS configurations.

Workaround: Recreate your Virtual Warehouse manually setting the right configurations.

Carried over from the previous release: Upgrade-related

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

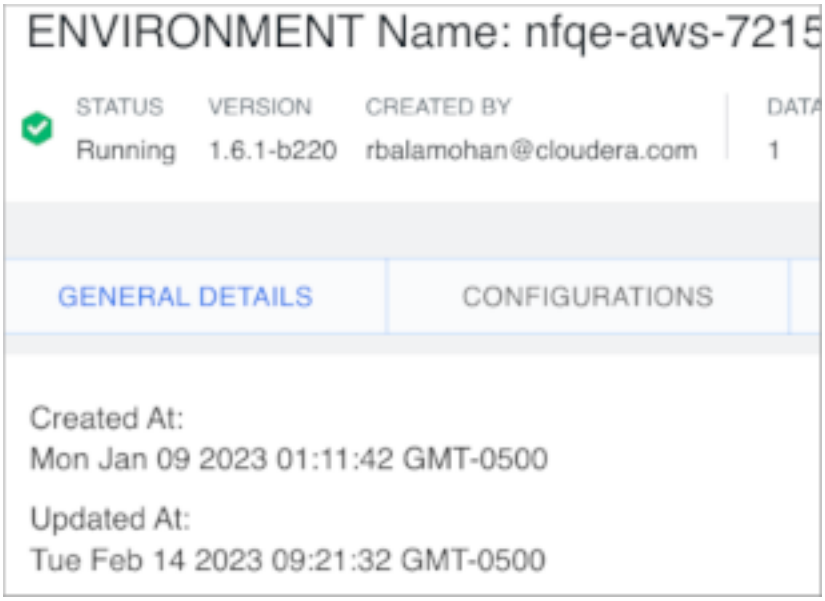

DWX-15114 Incorrect Database Catalog status indication incorrect after an upgrade

After a Database Catalog upgrade, if the Databus producer crashes, the default Database Catalog does not indicate the error. The Database Catalog status instead indicates "running". After a Database Catalog upgrade, if the Databus producer crashes, a non-default (custom) Database Catalog, indicates the correct error status.

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

worker subnets should be selected from 3 different availability zones. got: 0

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-15151 Runtime error after activating an environment

Activating an environment having a public load balancer can cause a 2083 runtime error

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. The following error can occur:

```
RuntimeErr with ErrCode=2083 (cause: 0 'coredns' replica(s) are
ready) Error Code : undefined
```

DWX-15171 Error after disabling SSO.

Disabling SSO for Impala Virtual Warehouse can cause an error.

To reproduce this issue 1) Create an Impala Virtual Warehouse having SSO enabled. 2) Edit the Virtual Warehouse to disable SSO.

The coordinator crashes.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-6619 Browser auto-close not working on some browsers after token-Based authentication for accessing CDW

The Firefox and Edge browser window does not close automatically after successful authentication.

DWX-9774 Database Catalog or Virtual Warehouse image version problem

Background: In Cloudera Data Warehouse 2021.0.3-b27 - 2021.0.5-b36, you can choose any supported image version when you create a Database Catalog or Virtual Warehouse, assuming you have the CDW_VERSIONED_DEPLOY entitlement.

Problem: In Cloudera Data Warehouse 2021.0.6-b96, you can choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse only if you use an existing environment. An existing environment is one you created in 2021.0.3-b27 - 2021.0.5-b36.

Workaround: In 2021.0.6-b96, use an environment you created before this release to choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

DWX-15246 Missing entries in subnet selection during CDW environment activation

The subnet selection during environment activation might not display any, or all, of the subnets registered in the environment.

Workaround: Carefully review the list of subnets you can select during activation. If subnets you want are not visible, enter the subnet-ids manually into the list, and then activate the environment.

DWX-14409 Starting a Virtual Warehouse in an AWS environments in the Asia-Pacific Southeast region fails

When you upgrade the Virtual Warehouse in this release and start it, a timing issue causes failure.

Workaround: Continue using the previous release 1.5.1-b110 (released Nov 22, 2022). Do not upgrade to 1.6.1-b282 (released Feb 7, 2023). Delay upgrading your Virtual Warehouse until the next release of CDW.

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected:

- CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 if they upgrade the node image
- CDW Azure users creating a new CDW environment with a Hue version earlier than 2023.0.14.0 (released May 5, 2023) during Hive or Impala Virtual Warehouse creation

Workaround: Upgrade the Virtual Warehouses to 1.6.3 or later and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

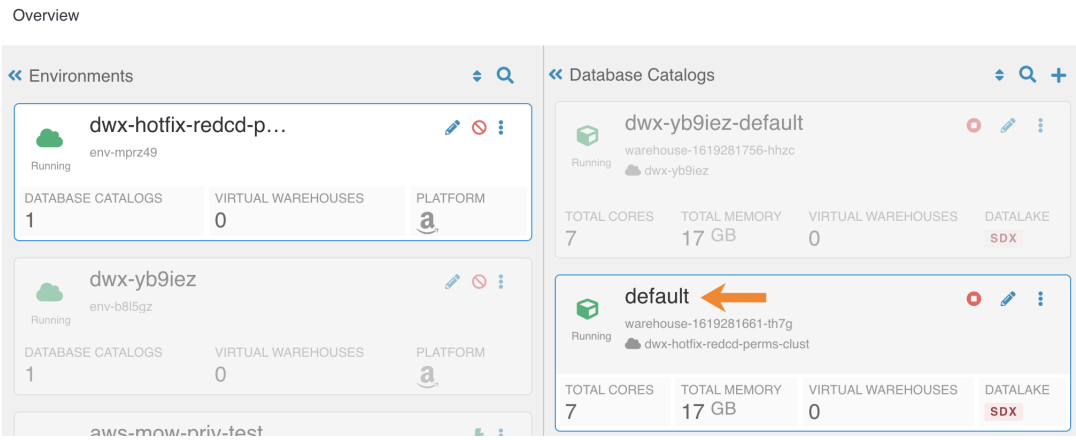

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

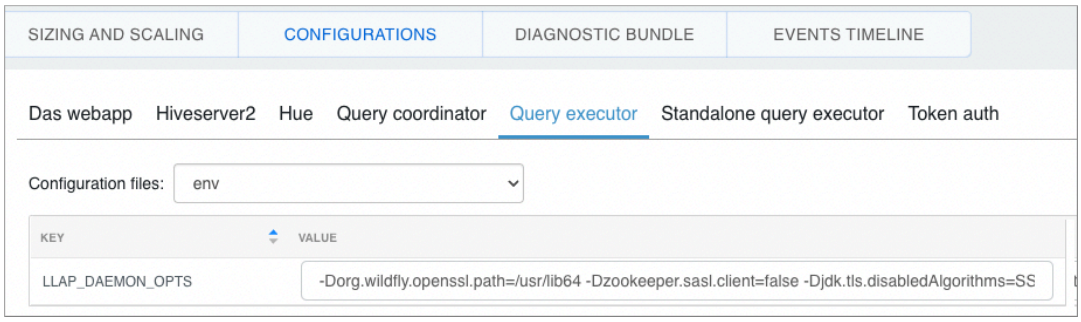

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
```

```
        - apps
        resources:
        - deployments
        - statefulsets
        verbs:
        - get
        - list
        - apiGroups:
        - "edws.cloudera.com"
        resources:
        - computes
        verbs:
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DWX-14286 Loading data to Iceberg error

Using LOAD DATA INPATH from an Impala Virtual Warehouse to load data to an Iceberg table on S3/ADLS when hidden files (files that are prefixed with . or _) are in the path can lead to unexpected query failures. For example:

```
AnalysisException: INPATH contains unsupported LOAD format, file
 '
            s3a://dwx-testdata/impala/sql_test/tests/load_data_in
path/runtime_data/0690a6fa9bfb11ed920c164053429bec/load_data_tes
t/A/impala_data/impala_alltypessmall_data/alltypessmall_parquet_
iceberg/year=2009/month=1/.hiddenfileforloaddatatest
            ' has 'This' magic string.
```

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

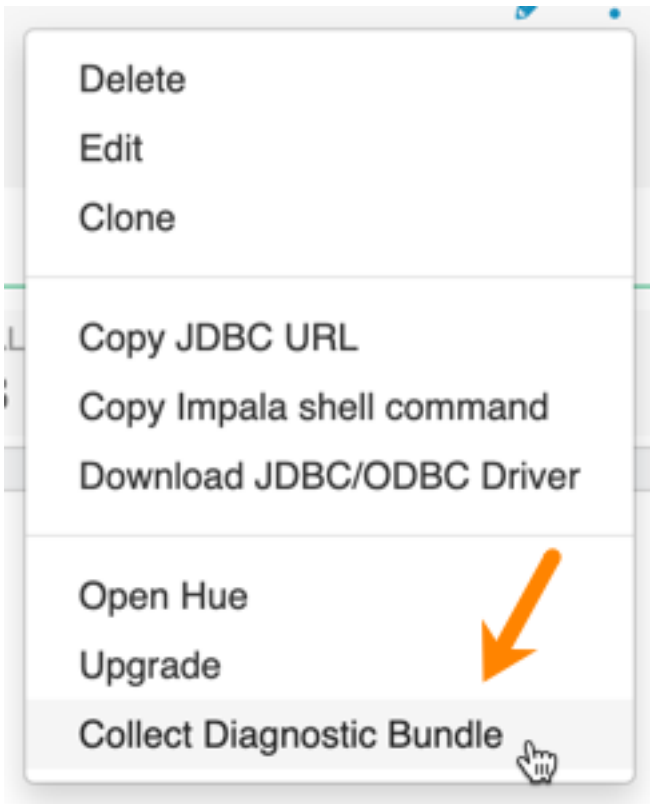

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-15110 Misaligned dialog in the UI

The Create Virtual Warehouse dialog is not aligned properly when no Virtual Warehouses are running.

DWX-14992 Cosmetic UI problem with Enable DAS link

In Azure environments, no gap appears between the previous text box and Enable DAS.

DWX-14970 Cosmetic UI problem with Virtual Warehouse start label

The start labels of the Impala Virtual Warehouse and Hive Virtual Warehouse are inconsistent with regard to upper- and lower-case.

DWX-14874 No indication that there are no Virtual Warehouses running

When no Data Visualization instance is running, you see "Currently no environments connected to Data Visualization cluster". The same indicator should be displayed when no Virtual Warehouses are running.

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

May 30, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New Known Issues in this release

DWX-15701 After a CDW upgrade, a Solr auditing problem exists

After a CDW upgrade, Solr cannot load ranger_audits collection and hence fails to push audit events to Ranger.

DWX-15395 Hive compaction fails in Azure

When RAZ is enabled in Cloudera Data Warehouse Azure environment, Hive compaction fails and returns an error message something like this:

```
ErrorMessage = FAILED: HiveAccessControlException Permission den
ied: user [ed64e526-a445-4945-9ed6-2eaf9560ff08] does not have [
READ] privilege on [[abfs:
```

Workaround:

1) In the CDW UI, in Overview, select a Hive Virtual Warehouse, and click Edit.

2) In Details, click CONFIGURATIONS HiveServer2 . 3) Select hive-site from the drop-down list, and search for metastore.compactor.run.as.user.

4) If the metastore.compactor.run.as.user key does not appear, click + and add it.

5) In Value, enter your Microsoft managed identity, and save the change.

Restart the Virtual Warehouse.

DWX-15473 Reactivating the AWS environment in CDW fails

When activating the AWS environment in CDW, the usage monitor can fail due to a problem connecting to prometheus. The connection problem can be caused by a restrictive AWS account policy. The follow error message occurs:

```
User: arn:aws:sts::OBFUSCATED:assumed-role/env-OBFUSCATED-dwx-st
ack-NodeInstanceRole-OBFUSCATED/i-OBFUSCATED is not authorized to
 perform: elasticfilesystem:TagResource on the specified resource
```

Workaround: After reactivating the CDW Environment, navigate to associated Cloudformation template for CDW, click Resources. Select NodeInstanceRole, and then select **efs inline policy**. Replace the JSON policy with the policy on the kubernetes-sigs github site.

Carried over from the previous release: Upgrade-related

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available. When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

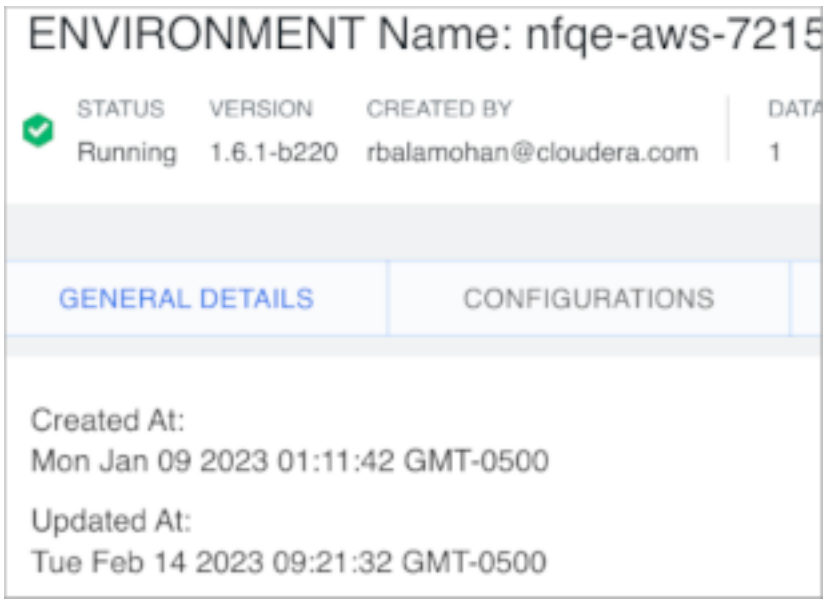

DWX-15114 Incorrect Database Catalog status indication incorrect after an upgrade

After a Database Catalog upgrade, if the Databus producer crashes, the default Database Catalog does not indicate the error. The Database Catalog status instead indicates "running". After a Database Catalog upgrade, if the Databus producer crashes, a non-default (custom) Database Catalog, indicates the correct error status.

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

Carried over from the previous release: General

DWX-14923 After JWT authentication, attempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

```
worker subnets should be selected from 3 different
availability zones. got: 0
```

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-15151 Runtime error after activating an environment

Activating an environment having a public load balancer can cause a 2083 runtime error

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. The following error can occur:

```
RuntimeErr with ErrCode=2083 (cause: 0 'coredns' replica(s) are
ready) Error Code : undefined
```

DWX-15171 Error after disabling SSO.

Disabling SSO for Impala Virtual Warehouse can cause an error.

To reproduce this issue 1) Create an Impala Virtual Warehouse having SSO enabled. 2) Edit the Virtual Warehouse to disable SSO.

The coordinator crashes.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-6619 Browser auto-close not working on some browsers after token-Based authentication for accessing CDW

The Firefox and Edge browser window does not close automatically after successful authentication.

DWX-9774 Database Catalog or Virtual Warehouse image version problem

Background: In Cloudera Data Warehouse 2021.0.3-b27 - 2021.0.5-b36, you can choose any supported image version when you create a Database Catalog or Virtual Warehouse, assuming you have the CDW_VERSIONED_DEPLOY entitlement.

Problem: In Cloudera Data Warehouse 2021.0.6-b96, you can choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse only if you use an existing environment. An existing environment is one you created in 2021.0.3-b27 - 2021.0.5-b36.

Workaround: In 2021.0.6-b96, use an environment you created before this release to choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

DWX-15246 Missing entries in subnet selection during CDW environment activation

The subnet selection during environment activation might not display any, or all, of the subnets registered in the environment.

Workaround: Carefully review the list of subnets you can select during activation. If subnets you want are not visible, enter the subnet-ids manually into the list, and then activate the environment.

DWX-14409 Starting a Virtual Warehouse in an AWS environments in the Asia-Pacific Southeast region fails

When you upgrade the Virtual Warehouse in this release and start it, a timing issue causes failure.

Workaround: Continue using the previous release 1.5.1-b110 (released Nov 22, 2022). Do not upgrade to 1.6.1-b282 (released Feb 7, 2023). Delay upgrading your Virtual Warehouse until the next release of CDW.

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the Cr eateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.

6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected:

- CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 if they upgrade the node image
- CDW Azure users creating a new CDW environment with a Hue version earlier than 2023.0.14.0 (released May 5, 2023) during Hive or Impala Virtual Warehouse creation

Workaround: Upgrade the Virtual Warehouses to 1.6.3 or later and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

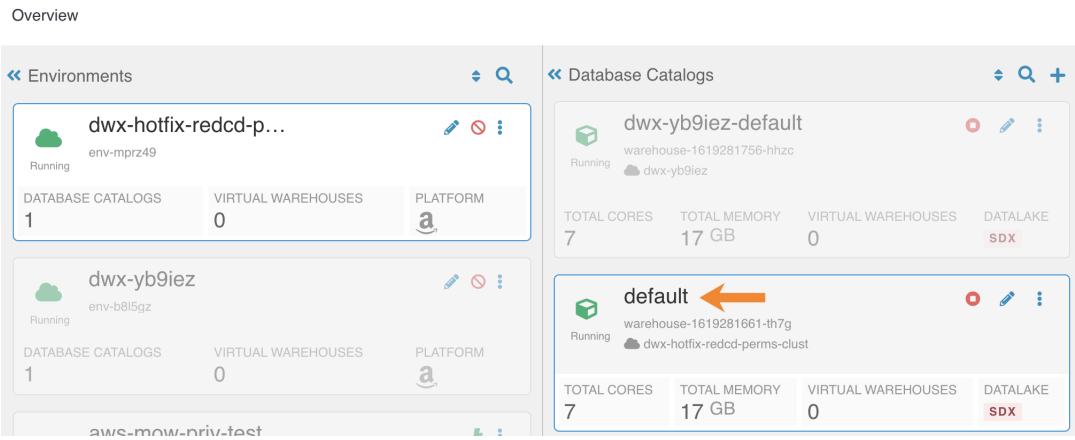

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

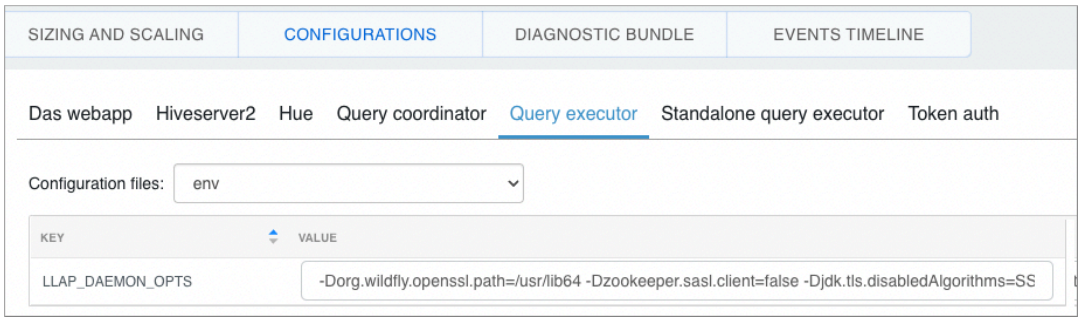

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
       - apps
       resources:
       - deployments
       - statefulsets
       verbs:
       - get
       - list
       - apiGroups:
       - "edws.cloudera.com"
       resources:
       - computes
       verbs:
       - get
       - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DWX-14286 Loading data to Iceberg error

Using LOAD DATA INPATH from an Impala Virtual Warehouse to load data to an Iceberg table on S3/ADLS when hidden files (files that are prefixed with . or _) are in the path can lead to unexpected query failures. For example:

```
AnalysisException: INPATH contains unsupported LOAD format, file
 '
            s3a://dwx-testdata/impala/sql_test/tests/load_data_in
path/runtime_data/0690a6fa9bfb11ed920c164053429bec/load_data_tes
t/A/impala_data/impala_alltypessmall_data/alltypessmall_parquet_
iceberg/year=2009/month=1/.hiddenfileforloaddatatest
            ' has 'This' magic string.
```

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

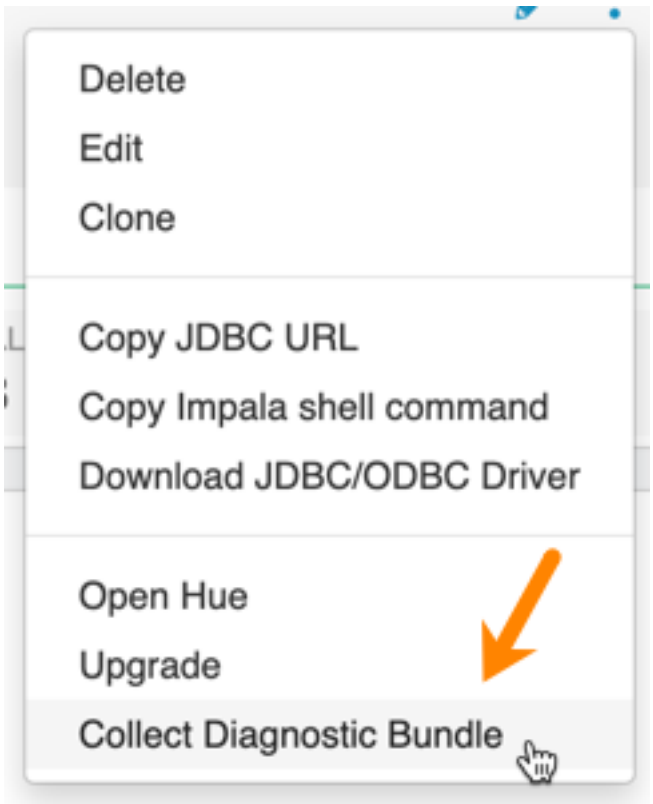

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

UI Issues

DWX-15110 Misaligned dialog in the UI

The Create Virtual Warehouse dialog is not aligned properly when no Virtual Warehouses are running.

DWX-14992 Cosmetic UI problem with Enable DAS link

In Azure environments, no gap appears between the previous text box and Enable DAS.

DWX-14970 Cosmetic UI problem with Virtual Warehouse start label

The start labels of the Impala Virtual Warehouse and Hive Virtual Warehouse are inconsistent with regard to upper- and lower-case.

DWX-14874 No indication that there are no Virtual Warehouses running

When no Data Visualization instance is running, you see "Currently no environments connected to Data Visualization cluster". The same indicator should be displayed when no Virtual Warehouses are running.

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

Technical Service Bulletins

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table
- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

May 5, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New known Issues in this release

TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

The Cloudera Data Warehouse (CDW) Public Cloud 2023.0.14.0 (DWX-1.6.3) version incorporated a feature for performing automatic metadata synchronization across multiple Apache Impala (Impala) Virtual Warehouses. The feature is enabled by default, and relies on the Hive MetaStore events. When a certain sequence of Data Definition Language (DDL) SQL commands are executed as described below, users may encounter a java.lang.NullPointerException (NPE). The exception causes the event processor to stop processing other metadata operations.

If a CREATE TABLE command (not CREATE TABLE AS SELECT) is followed immediately (approximately within 1 second interval) by INVALIDATE METADATA or REFRESH TABLE command on the same table (either on the same Virtual Warehouse or on a different one), there is a possibility that the second command will not find the table in the catalog cache of a peer Virtual Warehouse and generate an NPE.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-684: Automatic metadata synchronization across multiple Impala Virtual Warehouses (in CDW Public Cloud) may encounter an exception

Enabling a private CDW environment in Azure Kubernetes Service is unstable

Cloudera rolled back the General Availability (GA) status and support for deploying a private CDW environment in Azure using AKS until a solution for the issue is available.

When working together with Microsoft Azure (Azure), a networking issue has been identified at Azure Software Definition Layer (SDN) that impacts the use of CDW private environments while using the Azure Kubernetes Service (AKS). This issue is highly unpredictable and may cause timeouts during Cloudera Data Warehouse (CDW) environment activation, Virtual Warehouse creation, Virtual Warehouse modification, and/or during start and stop operations in CDW.

The following users are affected: CDW users on Azure who are activating a private CDW environment by selecting the Private CDW option on the User Interface (UI) or using CDP Command Line Interface (CLI) enablePrivateAks option for the dw sub-command create-cluster operation within the --aws-options group.

Activation can fail with a timeout issue, and other operation related actions may be interrupted.

Certain CDW versions cannot upgrade to EKS 1.22

AWS environments activated in version 1.4.1-b86 (released June, 22, 2022) or earlier are not supported for upgrade to EKS 1.22 due to incompatibility of some components with EKS 1.22. To determine the activation version your pre-existing environment, in the Data Warehouse service, expand Environments. In Environments, search for and locate the environment that you want to view. Click Edit. In Environment Details, you see the CDW version.

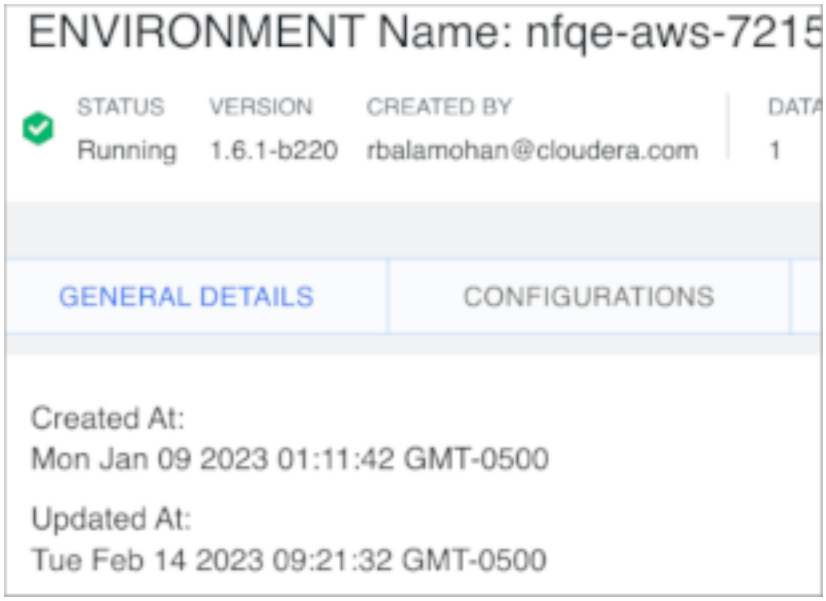

DWX-15473 Reactivating the AWS environment in CDW fails

When activating the AWS environment in CDW, the usage monitor can fail due to a problem connecting to prometheus. The connection problem can be caused by a restrictive AWS account policy. The follow error message occurs:

```
User: arn:aws:sts::OBFUSCATED:assumed-role/env-OBFUSCATED-dwx-st
ack-NodeInstanceRole-OBFUSCATED/i-OBFUSCATED is not authorized to
 perform: elasticfilesystem:TagResource on the specified resource
```

Workaround: After reactivating the CDW Environment, navigate to associated Cloudformation template for CDW, click Resources. Select NodeInstanceRole, and then select **efs inline policy**. Replace the JSON policy with the policy on the kubernetes-sigs github site.

DWX-15395 Hive compaction fails in Azure

When RAZ is enabled in Cloudera Data Warehouse Azure environment, Hive compaction fails and returns an error message something like this:

```
ErrorMessage = FAILED: HiveAccessControlException Permission den
ied: user [ed64e526-a445-4945-9ed6-2eaf9560ff08] does not have [
READ] privilege on [[abfs:
```

Workaround:

1) In the CDW UI, in Overview, select a Hive Virtual Warehouse, and click Edit.

2) In Details, click CONFIGURATIONS HiveServer2 . 3) Select hive-site from the drop-down list, and search for metastore.compactor.run.as.user.

4) If the metastore.compactor.run.as.user key does not appear, click + and add it.

5) In Value, enter your Microsoft managed identity, and save the change.

Restart the Virtual Warehouse.

DWX-15246 Missing entries in subnet selection during CDW environment activation

The subnet selection during environment activation might not display any, or all, of the subnets registered in the environment.

Workaround: Carefully review the list of subnets you can select during activation. If subnets you want are not visible, enter the subnet-ids manually into the list, and then activate the environment.

DWX-15214 DWX-15176 Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

The Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed:

```
kubelet Error: failed to create containerd task: failed to creat
e shim task: OCI runtime create failed: runc create failed: unab
le to start container process: exec: “run_httpd.sh”: cannot run
executable found relative to current directory: unknown
```

This issue affects all CDW releases before 1.6.3 (released May 5, 2023)running on CDP Public Cloud on Azure.

The following users are affected:

- CDW Azure users using Hue in Impala and Hive Virtual Warehouses in environments earlier than version 1.6.3 if they upgrade the node image
- CDW Azure users creating a new CDW environment with a Hue version earlier than 2023.0.14.0 (released May 5, 2023) during Hive or Impala Virtual Warehouse creation

Workaround: Upgrade the Virtual Warehouses to 1.6.3 and choose Hue version 2023.0.14.0.

Do not choose Hue versions older than 2023.0.14.0 when creating a Virtual Warehouse in environments of version 1.6.3.

DWX-15016 Impala query failure when using Catalog high availability (HA)

Impala queries could fail with InconsistentMetadataFetchException when using HA feature in CDW. This happens when leader election failover for Impala Catalog Service happens due to a transient network error.

Workaround: Disable Catalog HA. In CDW, edit your Virtual Warehouse, and turn off Enable Impala Catalog Server HA.

DWX-14452 Parquet table query might fail

Querying a table stored in Parquet from Hive might fail with the following exception message: java.lang.RuntimeException: java.lang.StackOverflowError. This problem can occur when the IN predicate has 243 values or more, and a small stack (-Xss = 256k) is configured for Hive in CDW.

Workaround: Change the value of the Xss in the JVM args to use the default value (1Mb) as follows: Select Edit from the Options menu of your Virtual Warehouse. Click Configurations > Query Executor > and select env.

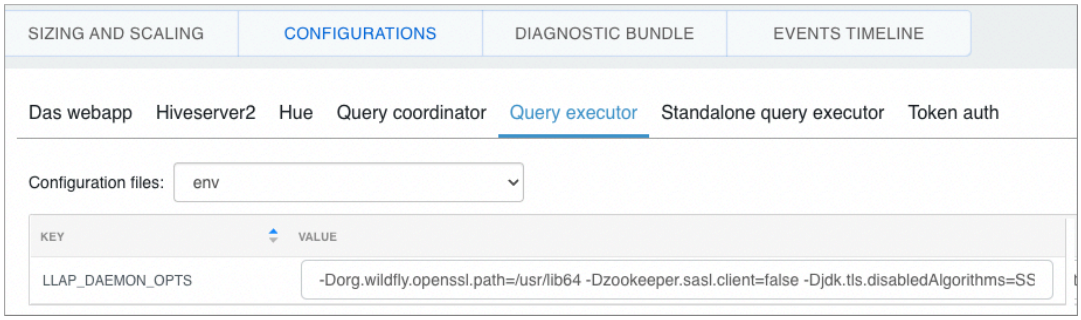

In the third line shown below, change the value of LLAP_DAEMON_OPTS from -Xss256k to -Xss1M, and then click Apply Changes:

FROM:

-Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Dhttp.maxConnections=12 -Xms24G -Xmx48G -Xss256k ...

TO:

... -Xss1M ...

DWX-15090: CSRF error intermittently seen in the Hue Job Browser

You may intermittently see the “403 - CSRF” error on the Hue web interface as well as in the Hue logs after running Hive queries from Hue.

Reload the browser or start a new Hue session.

DWX-15171 Error after disabling SSO.

Disabling SSO for Impala Virtual Warehouse can cause an error.

To reproduce this issue 1) Create an Impala Virtual Warehouse having SSO enabled. 2) Edit the Virtual Warehouse to disable SSO.

The coordinator crashes.

DWX-15151 Runtime error after activating an environment

Activating an environment having a public load balancer can cause a 2083 runtime error

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. The following error can occur:

```
RuntimeErr with ErrCode=2083 (cause: 0 'coredns' replica(s) are
ready) Error Code : undefined
```

DWX-15145 Environment validation popup error after activating an environment

Activating an environment having a public load balancer can cause an environment validatation popup error.

To reproduce this problem 1) Create a data lake. 2) Activate an enviroment having a public load balancer deployment type and subnets in three different availability zones. A environment validation popup can occur even through subnets are in different availability zones. Several differnt popups can occur, including the following one:

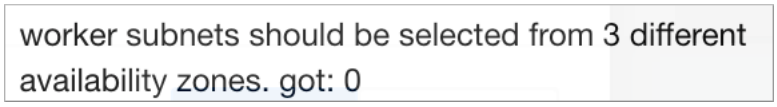

DWX-15144 Virtual Warehouse naming restrictions

You cannot create a Virtual Warehouse having the same name as another Virtual Warehouse even if the like-named Virtual Warehouses are in different environments. You can create a Database Catalog having the same name as another Database Catalog if the Database Catalogs are in different environments.

DWX-15115 Error displayed after clicking on hyperlink below Hue table browser

In Hue, below the table browser, clicking the hyperlink to a location causes an HTTP 500 error because the file browser is not enabled for environments that are not Ranger authorized (RAZ).

DWX-15114 Incorrect Database Catalog status indication incorrect after an upgrade

After a Database Catalog upgrade, if the Databus producer crashes, the default Database Catalog does not indicate the error. The Database Catalog status instead indicates "running". After a Database Catalog upgrade, if the Databus producer crashes, a non-default (custom) Database Catalog, indicates the correct error status.

DWX-15112 Enterprise Data Warehouse database configuration problems after a Helm-related rollback

If a Helm rollback fails due to an incorrect Enterprise Data Warehouse database configuration, the Virtual Warehouse and Database Catalog roll back to a previous configuration. The incorrect Enterprise Data Warehouse configuration persists, and can affect subsequent edit, upgrade, and rebuild operations on the rolled-back Virtual Warehouse or Database Catalog.

DWX-15110 Misaligned dialog in the UI

The Create Virtual Warehouse dialog is not aligned properly when no Virtual Warehouses are running.

DWX-15064 Hive Virtual Warehouse stops but appears healthy

Due to an istio-proxy problem, the query coordinator can unexpectedly enter a not ready state instead of the expected error-state. Subsequently, the Hive Virtual Warehouse stops when reaching the autosuspend timeout without indicating a problem.

DWX-15014 Loading airports table in demo data fails

This issue occurs only when using a non-default Database Catalog. A invalid path error message occurs when using the load command to load the demo data airports table in Iceberg, ORC, or Parquet format.

DWX-14992 Cosmetic UI problem with Enable DAS link

In Azure environments, no gap appears between the previous text box and Enable DAS.

DWX-14970 Cosmetic UI problem with Virtual Warehouse start label

The start labels of the Impala Virtual Warehouse and Hive Virtual Warehouse are inconsistent with regard to upper- and lower-case.

DWX-14968 Connection termination error in Impala queries tab after Hue inactivity

To reproduce the problem: 1) In the Impala job browser, navigate to Impala queries. 2) Wait for a few minutes.

After a few minutes of inactivity, the follow error is displayed:

Workaround: Refresh the page, or alternatively start a new session.

DWX-14927 Hue fails to list Iceberg snapshots

Hue does not recognize the Iceberg history queries from Hive to list table snapshots. For example, Hue indicates an error at the . before history when you run the following query.

```
select * from <db_name>.<table_name>.history
```

Delay in listing queries in Impala Queries in the Job browser

Listing an Impala query in the Job browser can take an inordinate amount of time.

DWX-14923 After JWT authentication, ttempting to connect the Impyla client using a user name or password should cause an error

Using a JWT token, you can connect to a Virtual Warehouse as the user who generated the token.

If you connect with the JWT token, and then pass a user name or password from Impyla to the Virtual Warehouse, the connection arguments are silently ignored. Such an action should indicate that it is illegal to specify user or password when using JWT authentication.

DWX-14874 No indication that there are no Virtual Warehouses running

When no Data Visualization instance is running, you see "Currently no environments connected to Data Visualization cluster". The same indicator should be displayed when no Virtual Warehouses are running.

DWX-14610 Upgrade icon indicates the Database Catalog is the latest version, but might be incorrect for a few seconds

The cluster fetches data regularly in particular time intervals, or when an action is triggered. Inbetween this time interval, which is very brief, the cluster might not be updated, but will be refreshed in a few seconds.

CDPD-54376: Clicking the home button on the File Browser page redirects to HDFS user directory

When you are previewing a file on any supported filesystem, such as S3 or ABFS, and you click on the Home button, you are redirected to the HDFS user home directory instead of the user home directory on the said filesystem.

None.

Carried over from the previous release: General

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-6619 Browser auto-close not working on some browsers after token-Based authentication for accessing CDW

The Firefox and Edge browser window does not close automatically after successful authentication.

DWX-9774 Database Catalog or Virtual Warehouse image version problem

Background: In Cloudera Data Warehouse 2021.0.3-b27 - 2021.0.5-b36, you can choose any supported image version when you create a Database Catalog or Virtual Warehouse, assuming you have the CDW_VERSIONED_DEPLOY entitlement.

Problem: In Cloudera Data Warehouse 2021.0.6-b96, you can choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse only if you use an existing environment. An existing environment is one you created in 2021.0.3-b27 - 2021.0.5-b36.

Workaround: In 2021.0.6-b96, use an environment you created before this release to choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse.

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

DWX-14409 Starting a Virtual Warehouse in an AWS environments in the Asia-Pacific Southeast region fails

When you upgrade the Virtual Warehouse in this release and start it, a timing issue causes failure.

Workaround: Continue using the previous release 1.5.1-b110 (released Nov 22, 2022). Do not upgrade to 1.6.1-b282 (released Feb 7, 2023). Delay upgrading your Virtual Warehouse until the next release of CDW.

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle. Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-xx-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-mvvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

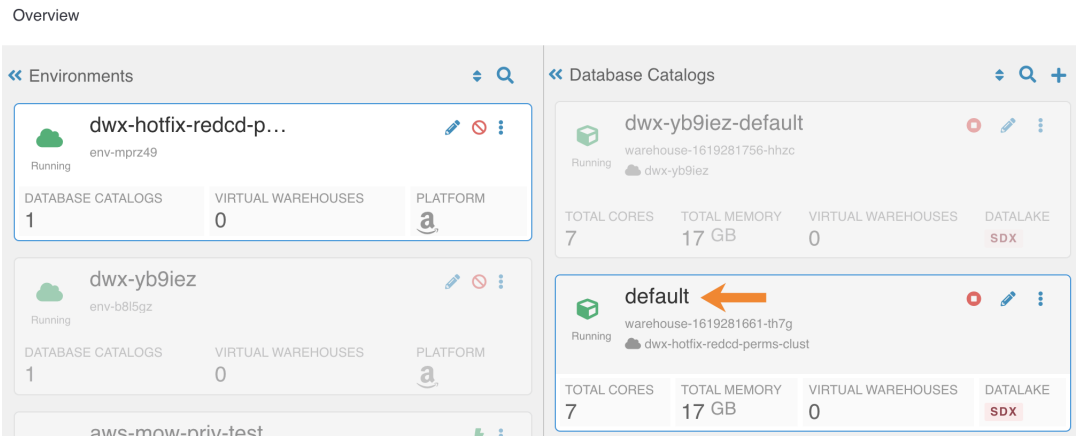

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
       - apps
       resources:
       - deployments
       - statefulsets
       verbs:
       - get
       - list
       - apiGroups:
       - "edws.cloudera.com"
       resources:
       - computes
       verbs:
       - get
       - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DWX-14286 Loading data to Iceberg error

Using LOAD DATA INPATH from an Impala Virtual Warehouse to load data to an Iceberg table on S3/ADLS when hidden files (files that are prefixed with . or _) are in the path can lead to unexpected query failures. For example:

```
AnalysisException: INPATH contains unsupported LOAD format, file
 '
            s3a://dwx-testdata/impala/sql_test/tests/load_data_in
path/runtime_data/0690a6fa9bfb11ed920c164053429bec/load_data_tes
t/A/impala_data/impala_alltypessmall_data/alltypessmall_parquet_
iceberg/year=2009/month=1/.hiddenfileforloaddatatest
            ' has 'This' magic string.
```

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

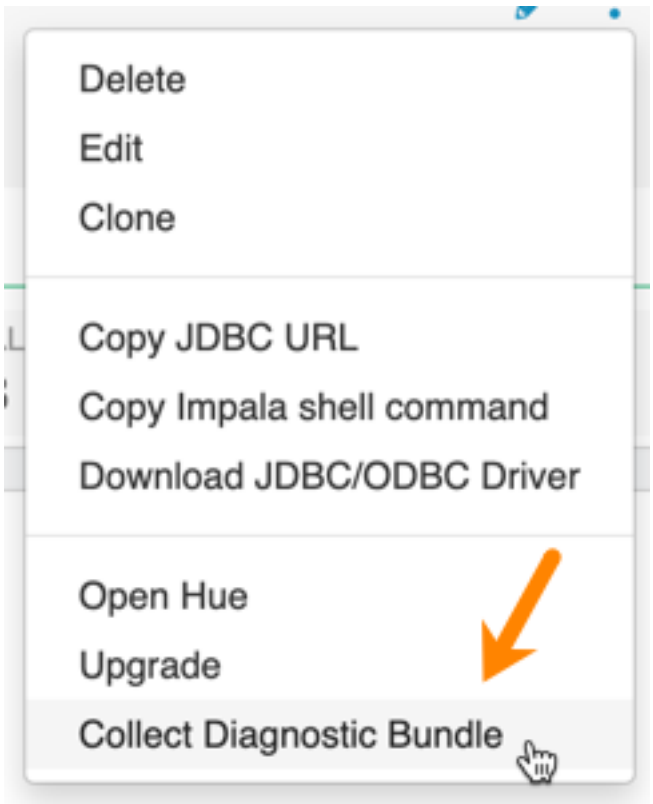

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes so the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

Technical Service Bulletins

TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables

The Spark Iceberg library includes two procedures - rewrite_data_files and rewrite_position_delete_files. The current implementation of rewrite_data_files has a limitation that the position delete files are not deleted and still tracked by the table metadata, even if they no longer refer to an active data file. This is called the dangling delete problem. To solve this, the rewrite_position_delete_files procedure is implemented in the Spark Iceberg library to remove these old “dangling” position delete files.

Due to the dangling delete limitation, when an Iceberg table with dangling deletes is queried in Impala, Impala tries to optimize select count(*) from iceberg_table query to return the results using stats. This optimization returns incorrect results.

The following conditions must be met for this issue to occur:

- All delete files in the Iceberg table are “dangling”
  - This would occur immediately after running Spark rewrite_data_files AND
    - Before any further delete operations are performed on the table OR
    - Before Spark rewrite_position_delete_files is run on the table

- Only stats optimized plain select count(*) from iceberg_table queries are affected. For example, the query should not have:
  - Any WHERE clause
  - Any GROUP BY clause
  - Any HAVING clause

For more information, see the Apache Iceberg documentation.

Remove dangling deletes: After rewrite_data_files, position delete records pointing to the rewritten data files are not always marked for removal, and can remain tracked by the live snapshot metadata of the table. This is known as the dangling delete problem.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-752: Dangling delete issue in Spark rewrite_data_files procedure causes incorrect results for Iceberg V2 tables.

February 7, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

Technical Service Bulletins

TSB-732 2024: Incorrect results are generated by Hive JOIN when bloom filter is activated

The bloom filter implemented in HIVE-23880 was designed to enhance performance for queries with JOIN statements, where one small table and another significantly larger table is joined on partition keys. However, the bloom filter introduced an issue in Apache Hive (Hive), when dynamic semijoin redaction is involved that generates incorrect query results. This issue is corrected in HIVE-26655.

Upstream JIRA

Hive-23880(cause)HIVE-26655(fix)

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-732: Incorrect results are generated by Hive JOIN when bloom filter is activated

New known Issues in this release

Do not upgrade this release to EKS 1.22

The latest version of Cloudera Data Warehouse (CDW) 1.6.1-b258 (released February 7, 2023) provisions AWS Elastic Kubernetes Service (EKS) 1.22 by default in AWS environments; however, upgrading your activated environments to EKS 1.22 or later is not supported. Do not upgrade your EKS clusters to 1.22.

DWX-14163 Limitations reading Iceberg tables in Avro file format from Impala

The Avro, Impala, and Iceberg specifications describe some limitations related to Avro, and those limitations exist in CDP. In addition to these, the DECIMAL type is not supported in this release.

DWX-14286 Loading data to Iceberg error

Using LOAD DATA INPATH from an Impala Virtual Warehouse to load data to an Iceberg table on S3/ADLS when hidden files (files that are prefixed with . or _) are in the path can lead to unexpected query failures. For example:

```
AnalysisException: INPATH contains unsupported LOAD format, file
 '
s3a://dwx-testdata/impala/sql_test/tests/load_data_inpath/runtime
_data/0690a6fa9bfb11ed920c164053429bec/load_data_test/A/impala_d
ata/impala_alltypessmall_data/alltypessmall_parquet_iceberg/year
=2009/month=1/.hiddenfileforloaddatatest
```

```
' has 'This' magic string.
```

Hive can crash due to the default setting of a caching property

Background: The relatively new hive.fetch.task.caching property was designed to prevent files from being cleaned up prematurely during the fetch phase. By default, hive.fetch.task.caching is enabled for queries that have been converted by the hive.fetch.task.conversion optimization.

Problem: The resulting caching activity can use too much heap while running queries that execute fetch tasks with a cache size of 1GB. Frequent garbage collection and high CPU usage can occur, causing Hue and other clients to timeout when returning results.

Workaround: Try setting hive.fetch.task.caching to false to disable the optimization. This action might cause queries to fail due to timing if the table is modified concurrent with the fetching. If problems continue, or cannot tolerate the low risk of a query failure, set hive.fetch.task.conversion to none. In this release, hive.fetch.task.conversion=none might add a little latency to some SELECT queries.

DWX-14409 Starting a Virtual Warehouse in an AWS environments in the Asia-Pacific Southeast region fails

When you upgrade the Virtual Warehouse in this release and start it, a timing issue causes failure.

Workaround: Continue using the previous release 1.5.1-b110 (released Nov 22, 2022). Do not upgrade to 1.6.1-b282 (released Feb 7, 2023). Delay upgrading your Virtual Warehouse until the next release of CDW.

DWX-14376 Upgrade Database Catalog button issue

A UI issue exists when you to upgrade the latest Virtual Warehouse 1.6.1-b258 (released Feb 7, 2023) and you are already running the latest release. The Upgrade Database Catalog button is not working when you attempt this futile Virtual Warehouse upgrade.

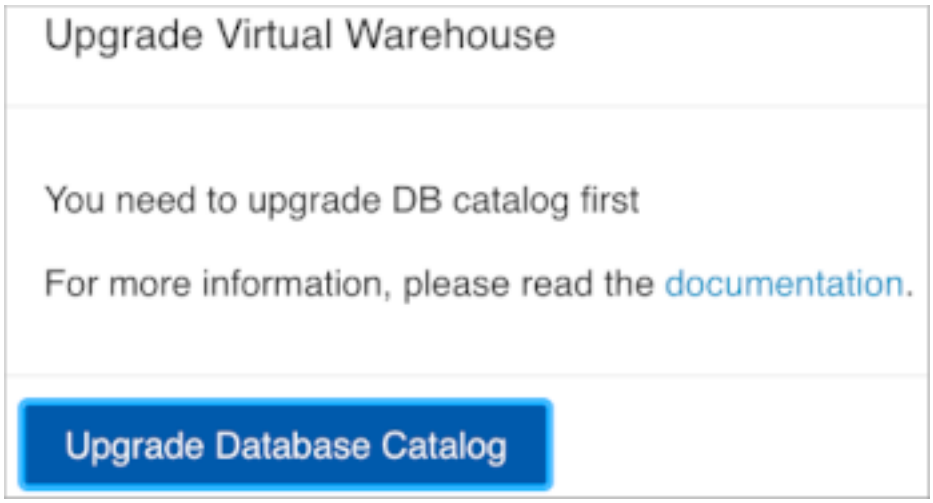

Carried over from the previous release: General

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-6619 Browser auto-close not working on some browsers after token-Based authentication for accessing CDW

The Firefox and Edge browser window does not close automatically after successful authentication.

DWX-9774 Database Catalog or Virtual Warehouse image version problem

Background: In Cloudera Data Warehouse 2021.0.3-b27 - 2021.0.5-b36, you can choose any supported image version when you create a Database Catalog or Virtual Warehouse, assuming you have the CDW_VERSIONED_DEPLOY entitlement.

Problem: In Cloudera Data Warehouse 2021.0.6-b96, you can choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse only if you use an existing environment. An existing environment is one you created in 2021.0.3-b27 - 2021.0.5-b36.

Workaround: In 2021.0.6-b96, use an environment you created before this release to choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

Workloads from CDW versions 1.4.2 and earlier cannot be deployed on new AWS environments

You must upgrade the version of Kubernetes that supports your existing CDW clusters to EKS version 1.21. AWS environments you activate using this release of Cloudera Data Warehouse, and later, will use version 1.22.

Because new environments are provisioned automatically to use AWS Elastic Kubernetes Service (EKS) 1.22, deprecated APIs used in workloads of versions 1.4.2-b118 (released August 4, 2022) and earlier are not supported in this version 1.5.1-b110 (released November 22, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads or use workloads from the previous release CDW 1.4.3--b225 (released September 15, 2022). Workloads from CDW 1.4.2-b118 (released August 4, 2022) and

earlier in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with EKS 1.22.

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the Cr eateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.

6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click Edit Diagnostic Bundle. Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

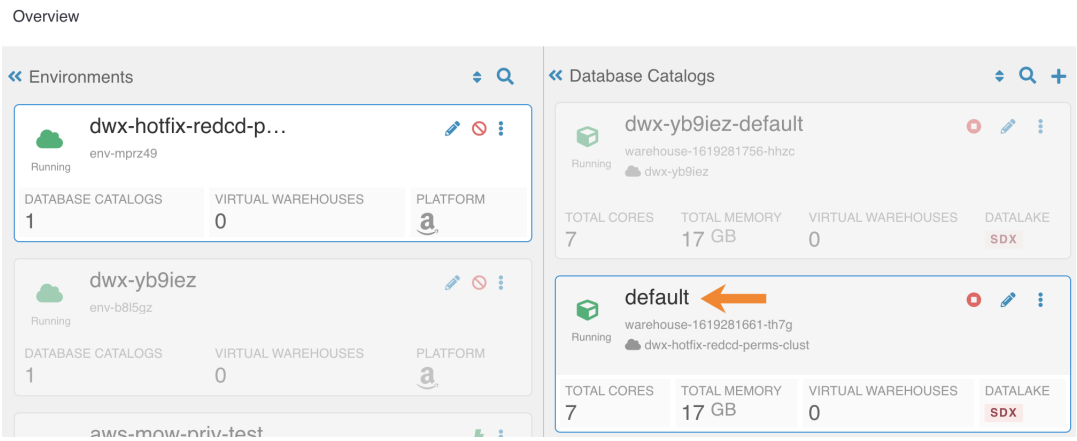

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
       rules:
       - apiGroups:
       - ""
       resources:
       - configmaps
       - events
       - pods
       - persistentvolumeclaims
       - nodes
       verbs:
       - get
       - list
       - apiGroups:
       - apps
       resources:
       - deployments
       - statefulsets
       verbs:
       - get
       - list
       - apiGroups:
       - "edws.cloudera.com"
       resources:
       - computes
       verbs:
       - get
       - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-10271: Missing log section in Hive query results

In a Hive Virtual Warehouse, when you run a query in Hue, the query results do not contain a logs section.

DWX-8118: INSERT INTO command fails under certain circumstances

This problem affects users who have a PostgreSQL database as the backend Hive database. If you create a table A and create a table B as select (CTAS) from an empty table A, inserting values into table B fails as follows:

```
Error while compiling statement: FAILED: Execution Error, return
 code 1 from
       org.apache.hadoop.hive.ql.exec.StatsTask.org.apache.thr
ift.transport.TTransportException
```

Workaround: Disable auto-stats gathering: Go to Cloudera Manager Data Warehouse Virtual Warehouses , and click the Hive VW name in the list. In Configuration HiveServer2 , set hive.cbo .enable to false.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

TSB 2023-670: Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed: kubelet Error: failed to create containerd task: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: “run_httpd.sh”: cannot run executable found relative to current directory: unknown

The root cause of the issue is a recent node image upgrade in Azure made by Microsoft, where the version of the container runtime (containerd) was upgraded to 1.6.18. This version upgraded Go SDK to 1.19.6, where the behavior of how program execution works has changed. Due to security concerns, the newer version of Go SDK does not resolve a program using an implicit or explicit path entry relative to the current directory (for more information, see the Go documentation), and containerd indirectly uses this Go exec API. The Docker command of the Hue frontend relies on the former behavior. Therefore the command fails on recent node images, because it cannot find the expected executable in the current working directory of the containers anymore. Cloudera first noticed the aforementioned containerd version in the Azure Kubernetes Service (AKS) released on 2023-03-05. Every node image which is newer than the version released on 2023-03-05 is affected by this issue. Unfortunately, Cloudera does not have any insight to when Microsoft rolls out new node images in a given region. Therefore, it is possible that in some regions, the older node images are still in use, where the issue does not arise until a given regional update is applied to the node images.

CDW environments, which are already activated and running, are not affected as long as customers do not trigger a node image upgrade to the latest available version either using Azure Command Line Interface (CLI) or on the Azure portal. If the node image is upgraded to the latest version,

then the Virtual Warehouses also need to be upgraded to the latest CDWH-2023.0.14.0 version. Newly created CDW environments are not affected, but customers are advised to not choose a Hue version lower than CDWH-2023.0.14.0 during any Virtual Warehouse creation because such a configuration is affected by this issue.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-670: TSB Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata if you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

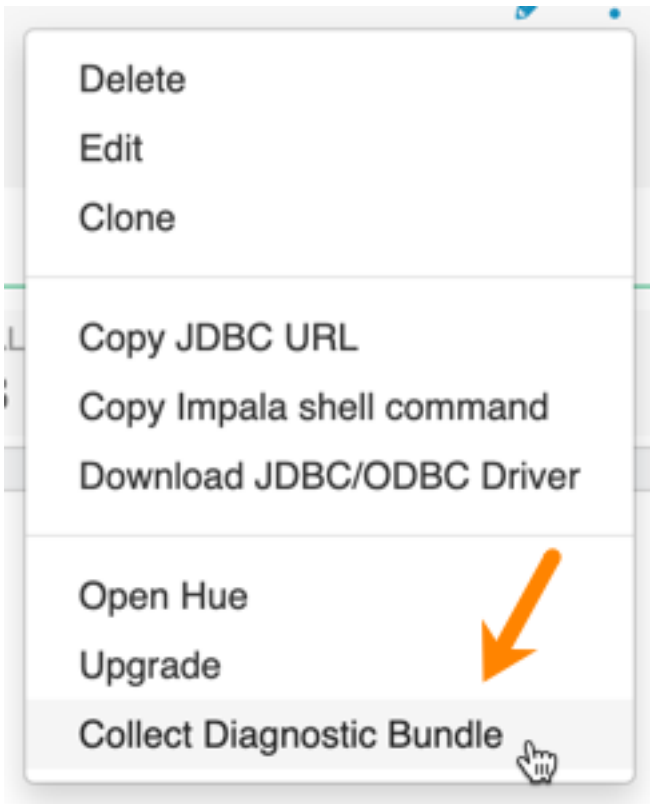

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case so the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

February 14, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

Technical Service Bulletins

TSB-732 2024: Incorrect results are generated by Hive JOIN when bloom filter is activated

The bloom filter implemented in HIVE-23880 was designed to enhance performance for queries with JOIN statements, where one small table and another significantly larger table is joined on partition keys. However, the bloom filter introduced an issue in Apache Hive (Hive), when dynamic semijoin redaction is involved that generates incorrect query results. This issue is corrected in HIVE-26655.

Upstream JIRA

Hive-23880(cause)HIVE-26655(fix)

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-732: Incorrect results are generated by Hive JOIN when bloom filter is activated

December 13, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

Technical Service Bulletins

CDP CLI CDW command issue when activating an Azure DWX environment

When clients try to activate a Microsoft Azure (Azure) environment on Cloudera Data Platform (CDP) Cloudera Data Warehouse (CDW) for Public Cloud using the CDP Command Line Interface (CLI) dw command's create-cluster sub-command with the --use-private-load-balancer switch, the created load balancer will be public instead of private.

Example command:

```
cdp \
--profile ${PROFILE} \
dw create-cluster \
--environment-crn ${ENV} \
--use-private-load-balancer
```

For the latest update on this issue see the corresponding Knowledge article:

TSB 2022-642: CDP CLI CDW command issue when activating an Azure DWX environment

TSB-732 2024: Incorrect results are generated by Hive JOIN when bloom filter is activated

The bloom filter implemented in HIVE-23880 was designed to enhance performance for queries with JOIN statements, where one small table and another significantly larger table is joined on partition keys. However, the bloom filter introduced an issue in Apache Hive (Hive), when dynamic semijoin redaction is involved that generates incorrect query results. This issue is corrected in HIVE-26655.

Upstream JIRA

Hive-23880(cause)HIVE-26655(fix)

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-732: Incorrect results are generated by Hive JOIN when bloom filter is activated

Carried over from the previous release: General

Do not upgrade this release to EKS 1.22

The latest version of Cloudera Data Warehouse (CDW) 1.5.1-b110 (released November 22, 2022) provisions AWS Elastic Kubernetes Service (EKS) 1.22 by default in Azure environments; however, upgrading your activated environments to EKS 1.22 or later is not supported. Do not upgrade your EKS clusters to 1.22.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-6619 Browser auto-close not working on some browsers after token-Based authentication for accessing CDW

The Firefox and Edge browser window does not close automatically after successful authentication.

DWX-9774 Database Catalog or Virtual Warehouse image version problem

Background: In Cloudera Data Warehouse 2021.0.3-b27 - 2021.0.5-b36, you can choose any supported image version when you create a Database Catalog or Virtual Warehouse, assuming you have the CDW_VERSIONED_DEPLOY entitlement.

Problem: In Cloudera Data Warehouse 2021.0.6-b96, you can choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse only if you use an existing environment. An existing environment is one you created in 2021.0.3-b27 - 2021.0.5-b36.

Workaround: In 2021.0.6-b96, use an environment you created before this release to choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

Workloads from CDW versions 1.4.2 and earlier cannot be deployed on new AWS environments

You must upgrade the version of Kubernetes that supports your existing CDW clusters to EKS version 1.21. AWS environments you activate using this release of Cloudera Data Warehouse, and later, will use version 1.22.

Because new environments are provisioned automatically to use AWS Elastic Kubernetes Service (EKS) 1.22, deprecated APIs used in workloads of versions 1.4.2-b118 (released August 4, 2022) and earlier are not supported in this version 1.5.1-b110 (released November 22, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads or use workloads from the previous release CDW 1.4.3--b225 (released September 15, 2022). Workloads from CDW 1.4.2-b118 (released August 4, 2022) and earlier in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with EKS 1.22.

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the Cr eateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersio ns line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
```

```
            "autoscaling:DescribeAutoScalingGroups",
            "autoscaling:DescribeAutoScalingInstances",
            "autoscaling:DescribeTags",
            "autoscaling:DescribeLaunchConfigurations",
            "autoscaling:SetDesiredCapacity",
            "autoscaling:TerminateInstanceInAutoScalingGroup",
            "ec2:DescribeLaunchTemplateVersions",
            "ec2:CreateTags"
            ],
```

7. Save changes.

Carried over from the previous release: Azure

DWX-13911 Problem activating Azure clusters from CDW using the CDP CLI

Activating an Azure environment from the CDP CLI 0.9.71 using the use-private-load-balancer switch creates a public instead of a private load-balancer. The problem occurs only when using version 0.9.71.

Affected command: cdp dw create-cluster

Example command:

```
cdp \
  --profile ${PROFILE} \
  dw create-cluster \
  --environment-crn ${ENV} \
  --use-private-load-balancer
```

Workaround: Add the following line to the end of the command to create the private load balancer:

```
--aws-options privateSubnetIds=dummy
```

For example, activate the environment using the following command:

```
cdp \
  --profile ${PROFILE} \
  dw create-cluster \
  --environment-crn ${ENV} \
  --use-private-load-balancer \
  --aws-options privateSubnetIds=dummy
```

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

DWX-8980 Data consistency problem occurs when sharing a Database Catalog

When you refresh, or invalidate, metadata in the Database Catalog from one Virtual Warehouse, the coordinators of any other Virtual Warehouses that share the Database Catalog might not receive the updates fast enough. Users can experience inconsistency in query results.

Workaround: Set the timeout for invalidating the tables to a very low value by following the steps below. Impala is forced to read files from the object store more frequently than every 10 minutes (the default timeout). Metadata is refreshed more frequently than normal, which can impact performance. The impact is proportional to the number of small files Impala must read to load metadata.

1) Log in to the CDP web interface and navigate to the Data Warehouse service.

2) In the Data Warehouse service, click Overview, select the first Virtual Warehouse, and click Edit.

3) In Configurations, in Impala catalogd, click +.

4) In Add Custom Configurations, set the invalidate_tables_timeout_s to a small value, and click Add.

The catalogd daemon caches metadata for tables and must be refreshed/invalidated quickly to prevent inconsistency.

5) In Impala coordinator, click +.

6) In Add Custom Configurations, set the invalidate_tables_timeout_s to a small value, and click Add.

The coordinator must also refresh/invalidate data quickly to prevent inconsistency.

7) Repeat these steps for each Virtual Warehouse that shares the same Database Catalog.

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).

- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

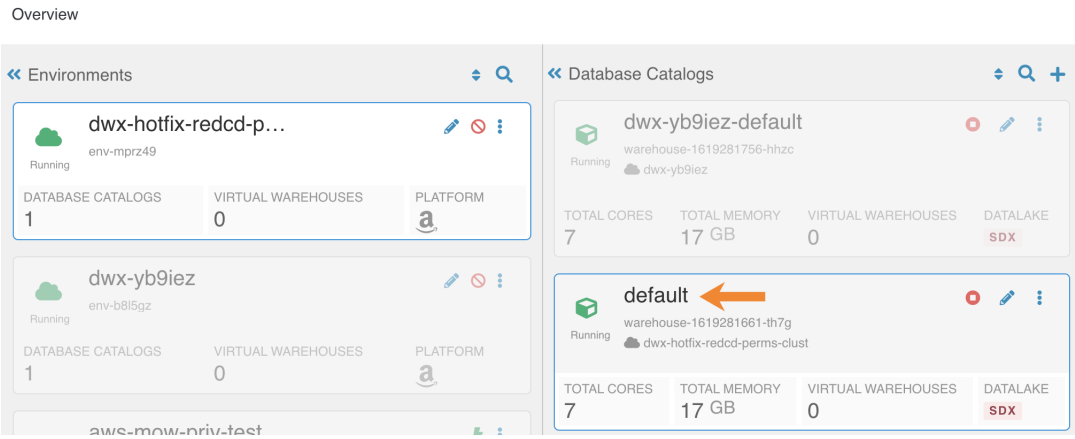

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
```

```
        rules:
        - apiGroups:
        - ""
        resources:
        - configmaps
        - events
        - pods
        - persistentvolumeclaims
        - nodes
        verbs:
        - get
        - list
        - apiGroups:
        - apps
        resources:
        - deployments
        - statefulsets
        verbs:
        - get
        - list
        - apiGroups:
        - "edws.cloudera.com"
        resources:
        - computes
        verbs:
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.par quet.write.int64.timestamp=true.

DWX-10271: Missing log section in Hive query results

In a Hive Virtual Warehouse, when you run a query in Hue, the query results do not contain a logs section.

DWX-8118: INSERT INTO command fails under certain circumstances

This problem affects users who have a PostgreSQL database as the backend Hive database. If you create a table A and create a table B as select (CTAS) from an empty table A, inserting values into table B fails as follows:

```
Error while compiling statement: FAILED: Execution Error, return
 code 1 from
       org.apache.hadoop.hive.ql.exec.StatsTask.org.apache.thr
ift.transport.TTransportException
```

Workaround: Disable auto-stats gathering: Go to Cloudera Manager Data Warehouse Virtual Warehouses , and click the Hive VW name in the list. In Configuration HiveServer2 , set hive.cbo .enable to false.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses

must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

TSB 2023-670: Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed: kubelet Error: failed to create containerd task: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: “run_httpd.sh”: cannot run executable found relative to current directory: unknown

The root cause of the issue is a recent node image upgrade in Azure made by Microsoft, where the version of the container runtime (containerd) was upgraded to 1.6.18. This version upgraded Go SDK to 1.19.6, where the behavior of how program execution works has changed. Due to security concerns, the newer version of Go SDK does not resolve a program using an implicit or explicit path entry relative to the current directory (for more information, see the Go documentation), and containerd indirectly uses this Go exec API. The Docker command of the Hue frontend relies on the former behavior. Therefore the command fails on recent node images, because it cannot find the expected executable in the current working directory of the containers anymore. Cloudera first noticed the aforementioned containerd version in the Azure Kubernetes Service (AKS) released on 2023-03-05. Every node image which is newer than the version released on 2023-03-05 is affected by this issue. Unfortunately, Cloudera does not have any insight to when Microsoft rolls out new node images in a given region. Therefore, it is possible that in some regions, the older node images are still in use, where the issue does not arise until a given regional update is applied to the node images.

CDW environments, which are already activated and running, are not affected as long as customers do not trigger a node image upgrade to the latest available version either using Azure Command Line Interface (CLI) or on the Azure portal. If the node image is upgraded to the latest version,

then the Virtual Warehouses also need to be upgraded to the latest CDWH-2023.0.14.0 version. Newly created CDW environments are not affected, but customers are advised to not choose a Hue version lower than CDWH-2023.0.14.0 during any Virtual Warehouse creation because such a configuration is affected by this issue.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-670: TSB Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata is you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

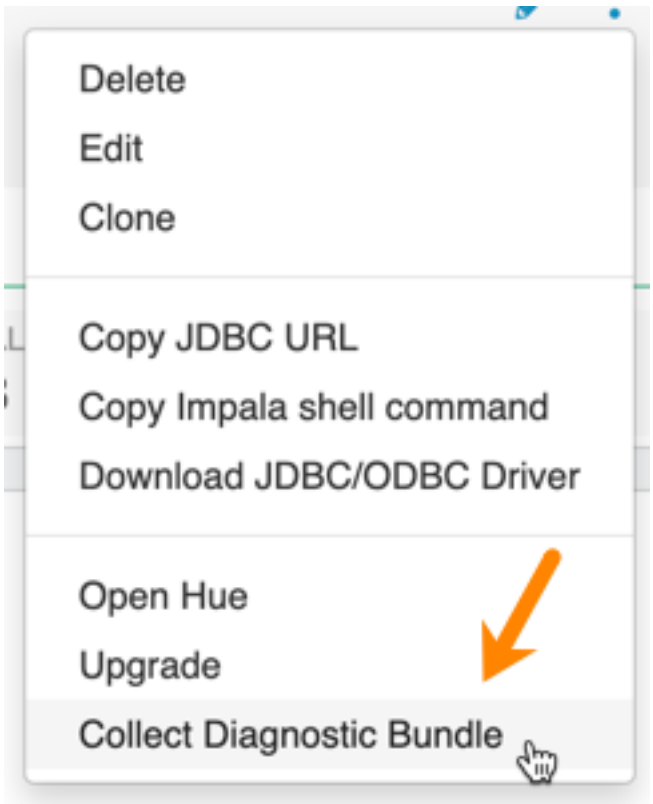

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case so the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

November 21, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

Technical Service Bulletins

TSB-732 2024: Incorrect results are generated by Hive JOIN when bloom filter is activated

The bloom filter implemented in HIVE-23880 was designed to enhance performance for queries with JOIN statements, where one small table and another significantly larger table is joined on partition keys. However, the bloom filter introduced an issue in Apache Hive (Hive), when dynamic semijoin redaction is involved that generates incorrect query results. This issue is corrected in HIVE-26655.

Upstream JIRA

Hive-23880(cause)HIVE-26655(fix)

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-732: Incorrect results are generated by Hive JOIN when bloom filter is activated

New known Issues in this release

Do not upgrade this release to EKS 1.22

The latest version of Cloudera Data Warehouse (CDW) 1.5.1-b110 (released November 22, 2022) provisions AWS Elastic Kubernetes Service (EKS) 1.22 by default in AWS environments; however,

upgrading your activated environments to EKS 1.22 or later is not supported. Do not upgrade your EKS clusters to 1.22.

Workloads from CDW versions 1.4.2 and earlier cannot be deployed on new AWS environments

You must upgrade the version of Kubernetes that supports your existing CDW clusters to EKS version 1.21. AWS environments you activate using this release of Cloudera Data Warehouse, and later, will use version 1.22.

Because new environments are provisioned automatically to use AWS Elastic Kubernetes Service (EKS) 1.22, deprecated APIs used in workloads of versions 1.4.2-b118 (released August 4, 2022) and earlier are not supported in this version 1.5.1-b110 (released November 22, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads or use workloads from the previous release CDW 1.4.3--b225 (released September 15, 2022). Workloads from CDW 1.4.2-b118 (released August 4, 2022) and earlier in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with EKS 1.22.

DWX-13911 Problem activating Azure clusters from CDW using the CDP CLI

Activating an Azure environment from the CDP CLI 0.9.71 using the use-private-load-balancer switch creates a public instead of a private load-balancer. The problem occurs only when using version 0.9.71.

Affected command: cdp dw create-cluster

Example command:

```
cdp \
  --profile ${PROFILE} \
  dw create-cluster \
  --environment-crn ${ENV} \
  --use-private-load-balancer
```

Workaround: Add the following line to the end of the command to create the private load balancer:

```
--aws-options privateSubnetIds=dummy
```

For example, activate the environment using the following command:

```
cdp \
  --profile ${PROFILE} \
  dw create-cluster \
  --environment-crn ${ENV} \
  --use-private-load-balancer \
  --aws-options privateSubnetIds=dummy
```

DEX-7946 Data loss during migration of a Hive table to Iceberg

In this release, by default the table property 'external.table.purge' is set to true, which deletes the table data and metadata is you drop the table during migration from Hive to Iceberg.

Workarounds: Either one of the following workarounds prevents data loss during table migration:

- Set the table property 'external.table.purge'='FALSE'.
- Do not drop a table during migration from Hive to Iceberg.

DWX-13733 Timeout issue querying Iceberg tables from Hive

A timeout issue can cause the query to fail.

Workaround: Add the following configuration to hadoop-core-site for the Database Catalog and the Virtual Warehouse.

```
fs.s3.maxConnections=1000
```

```
fs.s3a.connection.maximum=1000
```

Restart the Database Catalog and Virtual Warehouse.

Carried over from the previous release: General

TSB 2023-656: TSB Cloudera Data Warehouse does not clean up old Helm releases

Cloudera Data Warehouse (CDW) uses Helm release manager to deploy component releases into the Kubernetes cluster of the customer. However, CDW's current behavior in CDW keeps the Helm release history without any boundary. Since the storage for Helm in CDW uses secrets, this behavior can lead to a very high number of secrets (Impala/Hive scale-up/down creates new releases;clusters were seen with 6000+ secrets). Therefore, parsing and decoding a high number of Helm releases can require a significant amount of memory. As an example, for a cluster with 6000+ secrets, it could be 6-7 GB or more of memory utilized. This may result in performance degradation, instability and/or higher cloud cost because several CDW cluster components rely on secrets, which would cause the memory footprint of these other components to increase significantly.

This issue is addressed in the CDW version 1.5.1-b110 (workload image version 2022.0.11.0-122). Cloudera highly recommends upgrading Apache Impala (Impala) and Apache Hive (Hive) in CDW to 2022.0.11.0-122 or higher versions to ensure performance, stability and/or cloud cost are not impacted by this behavior.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-656: https://my.cloudera.com/knowledge/TSB-2023-656-Cloudera-Data-Warehouse-does-not-clean-up-old?id=368733

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-6619 Browser auto-close not working on some browsers after token-Based authentication for accessing CDW

The Firefox and Edge browser window does not close automatically after successful authentication.

DWX-9774 Database Catalog or Virtual Warehouse image version problem

Background: In Cloudera Data Warehouse 2021.0.3-b27 - 2021.0.5-b36, you can choose any supported image version when you create a Database Catalog or Virtual Warehouse, assuming you have the CDW_VERSIONED_DEPLOY entitlement.

Problem: In Cloudera Data Warehouse 2021.0.6-b96, you can choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse only if you use an existing environment. An existing environment is one you created in 2021.0.3-b27 - 2021.0.5-b36.

Workaround: In 2021.0.6-b96, use an environment you created before this release to choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreateTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle. Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Database Catalog

DWX-8980 Data consistency problem occurs when sharing a Database Catalog

When you refresh, or invalidate, metadata in the Database Catalog from one Virtual Warehouse, the coordinators of any other Virtual Warehouses that share the Database Catalog might not receive the updates fast enough. Users can experience inconsistency in query results.

Workaround: Set the timeout for invalidating the tables to a very low value by following the steps below. Impala is forced to read files from the object store more frequently than every 10 minutes (the default timeout). Metadata is refreshed more frequently than normal, which can impact performance. The impact is proportional to the number of small files Impala must read to load metadata.

1) Log in to the CDP web interface and navigate to the Data Warehouse service.

2) In the Data Warehouse service, click Overview, select the first Virtual Warehouse, and click Edit.

3) In Configurations, in Impala catalogd, click +.

4) In Add Custom Configurations, set the invalidate_tables_timeout_s to a small value, and click Add.

The catalogd daemon caches metadata for tables and must be refreshed/invalidated quickly to prevent inconsistency.

5) In Impala coordinator, click +.

6) In Add Custom Configurations, set the invalidate_tables_timeout_s to a small value, and click Add.

The coordinator must also refresh/invalidate data quickly to prevent inconsistency.

7) Repeat these steps for each Virtual Warehouse that shares the same Database Catalog.

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).

- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

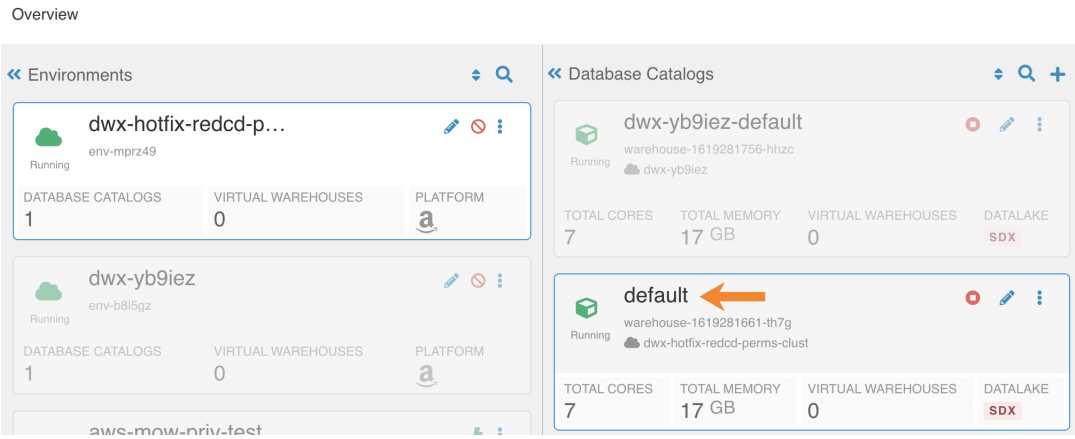

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
       kind: ClusterRole
       metadata:
       name: diagnostic-data-generator-role-monitor
```

```
        rules:
        - apiGroups:
        - ""
        resources:
        - configmaps
        - events
        - pods
        - persistentvolumeclaims
        - nodes
        verbs:
        - get
        - list
        - apiGroups:
        - apps
        resources:
        - deployments
        - statefulsets
        verbs:
        - get
        - list
        - apiGroups:
        - "edws.cloudera.com"
        resources:
        - computes
        verbs:
        - get
        - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.par quet.write.int64.timestamp=true.

DWX-10271: Missing log section in Hive query results

In a Hive Virtual Warehouse, when you run a query in Hue, the query results do not contain a logs section.

DWX-8118: INSERT INTO command fails under certain circumstances

This problem affects users who have a PostgreSQL database as the backend Hive database. If you create a table A and create a table B as select (CTAS) from an empty table A, inserting values into table B fails as follows:

```
Error while compiling statement: FAILED: Execution Error, return
 code 1 from
       org.apache.hadoop.hive.ql.exec.StatsTask.org.apache.thr
ift.transport.TTransportException
```

Workaround: Disable auto-stats gathering: Go to Cloudera Manager Data Warehouse Virtual Warehouses , and click the Hive VW name in the list. In Configuration HiveServer2 , set hive.cbo .enable to false.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses

must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Hue Query Editor

TSB 2023-670: Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed: kubelet Error: failed to create containerd task: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: “run_httpd.sh”: cannot run executable found relative to current directory: unknown

The root cause of the issue is a recent node image upgrade in Azure made by Microsoft, where the version of the container runtime (containerd) was upgraded to 1.6.18. This version upgraded Go SDK to 1.19.6, where the behavior of how program execution works has changed. Due to security concerns, the newer version of Go SDK does not resolve a program using an implicit or explicit path entry relative to the current directory (for more information, see the Go documentation), and containerd indirectly uses this Go exec API. The Docker command of the Hue frontend relies on the former behavior. Therefore the command fails on recent node images, because it cannot find the expected executable in the current working directory of the containers anymore. Cloudera first noticed the aforementioned containerd version in the Azure Kubernetes Service (AKS) released on 2023-03-05. Every node image which is newer than the version released on 2023-03-05 is affected by this issue. Unfortunately, Cloudera does not have any insight to when Microsoft rolls out new node images in a given region. Therefore, it is possible that in some regions, the older node images are still in use, where the issue does not arise until a given regional update is applied to the node images.

CDW environments, which are already activated and running, are not affected as long as customers do not trigger a node image upgrade to the latest available version either using Azure Command Line Interface (CLI) or on the Azure portal. If the node image is upgraded to the latest version,

then the Virtual Warehouses also need to be upgraded to the latest CDWH-2023.0.14.0 version. Newly created CDW environments are not affected, but customers are advised to not choose a Hue version lower than CDWH-2023.0.14.0 during any Virtual Warehouse creation because such a configuration is affected by this issue.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-670: TSB Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Iceberg

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: Impala Virtual Warehouse

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses

must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

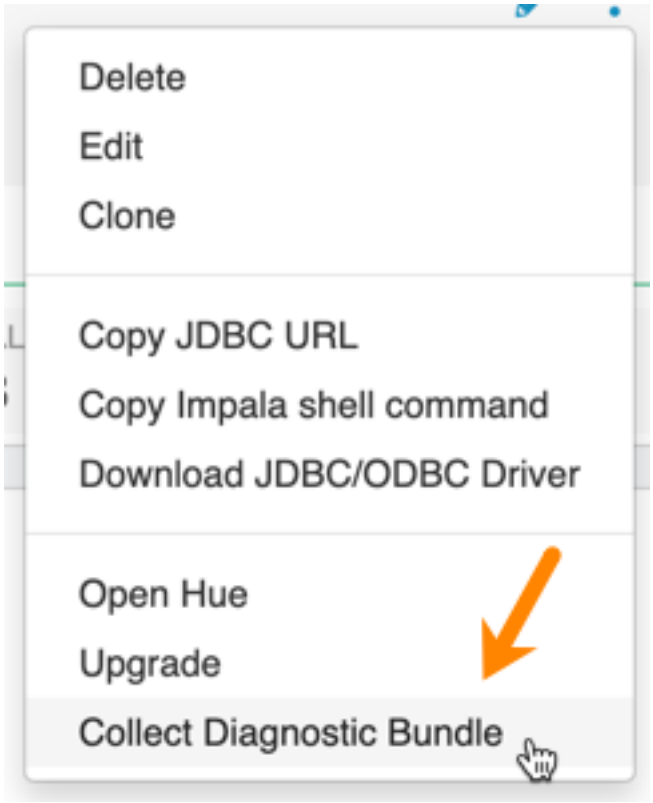

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case so the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

September 15, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

Technical Service Bulletins

TSB-732 2024: Incorrect results are generated by Hive JOIN when bloom filter is activated

The bloom filter implemented in HIVE-23880 was designed to enhance performance for queries with JOIN statements, where one small table and another significantly larger table is joined on partition keys. However, the bloom filter introduced an issue in Apache Hive (Hive), when dynamic semijoin redaction is involved that generates incorrect query results. This issue is corrected in HIVE-26655.

Upstream JIRA

Hive-23880(cause)HIVE-26655(fix)

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-732: Incorrect results are generated by Hive JOIN when bloom filter is activated

New known Issues in this release

DWX-8980 Data consistency problem occurs when sharing a Database Catalog

When you refresh, or invalidate, metadata in the Database Catalog from one Virtual Warehouse, the coordinators of any other Virtual Warehouses that share the Database Catalog might not receive the updates fast enough. Users can experience inconsistency in query results.

Workaround: Set the timeout for invalidating the tables to a very low value by following the steps below. Impala is forced to read files from the object store more frequently than every 10 minutes (the default timeout). Metadata is refreshed more frequently than normal, which can impact performance. The impact is proportional to the number of small files Impala must read to load metadata.

1) Log in to the CDP web interface and navigate to the Data Warehouse service.

2) In the Data Warehouse service, click Overview, select the first Virtual Warehouse, and click Edit.

3) In Configurations, in Impala catalogd, click +.

4) In Add Custom Configurations, set the invalidate_tables_timeout_s to a small value, and click Add.

The catalogd daemon caches metadata for tables and must be refreshed/invalidated quickly to prevent inconsistency.

5) In Impala coordinator, click +.

6) In Add Custom Configurations, set the invalidate_tables_timeout_s to a small value, and click Add.

The coordinator must also refresh/invalidate data quickly to prevent inconsistency.

7) Repeat these steps for each Virtual Warehouse that shares the same Database Catalog.

Diagnostic bundle download fails

After upgrading to version 1.4.3 (released September 15, 2022), downloading the diagnostic bundle for the Hive Virtual Warehouse results in an error. New environments do not have this problem. The problem is caused by missing permissions to access Kubernetes resources. Version 1.4.3 adds a requirement for the role-based access control (rbac) permissions to download the diagnostic bundle.

Workaround: Add the missing rbac permissions as follows:

1) Create the following file:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: diagnostic-data-generator-role-monitor
rules:
- apiGroups:
  - ""
  resources:
  - configmaps
  - events
  - pods
  - persistentvolumeclaims
  - nodes
  verbs:
  - get
  - list
- apiGroups:
  - apps
  resources:
  - deployments
  - statefulsets
  verbs:
  - get
  - list
- apiGroups:
  - "edws.cloudera.com"
  resources:
  - computes
  verbs:
  - get
  - list
```

2) Run the following command to apply the permissions:

```
kubectl apply -f <filename>
```

Workloads from earlier CDW versions cannot be deployed on new AWS environments

Because new environments are provisioned automatically to use AWS Elastic Kubernetes Service (EKS) 1.21, deprecated APIs used in workloads of earlier versions are not supported in this version 1.4.3-b225 (released September 15, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with EKS 1.21.

DWX-13062 Hive-26507 Converting a Hive table having CHAR or VARCHAR columns to Iceberg causes an exception

CHAR and VARCHAR data can be shorter than the length specified by the data type. Remaining characters are padded with spaces. Data is converted to a string in Iceberg. This process can yield incorrect results when you query the converted Iceberg table.

Workaround: Change columns from CHAR or VARCHAR to string types before converting the Hive table to Iceberg.

DWX-13276 Multiple inserts into tables having different formats can cause a deadlock.

Under the following conditions, a deadlock can occur:

- You run a query to insert data into multiple tables comprised of at least one Iceberg table and at least one non-Iceberg table.
- The STAT task locking feature is turned on (default = on).

Workarounds: Perform either one of the following workarounds:

- Run separate queries to insert data into only one table at a time.
- Turn off STAT task locking as follows:

```
set iceberg.hive.request-lock-on-stats-task=false;
```

Carried over from the previous release: General

TSB 2023-656: TSB Cloudera Data Warehouse does not clean up old Helm releases

Cloudera Data Warehouse (CDW) uses Helm release manager to deploy component releases into the Kubernetes cluster of the customer. However, CDW's current behavior in CDW keeps the Helm release history without any boundary. Since the storage for Helm in CDW uses secrets, this behavior can lead to a very high number of secrets (Impala/Hive scale-up/down creates new releases;clusters were seen with 6000+ secrets). Therefore, parsing and decoding a high number of Helm releases can require a significant amount of memory. As an example, for a cluster with 6000+ secrets, it could be 6-7 GB or more of memory utilized. This may result in performance degradation, instability and/or higher cloud cost because several CDW cluster components rely on secrets, which would cause the memory footprint of these other components to increase significantly.

This issue is addressed in the CDW version 1.5.1-b110 (workload image version 2022.0.11.0-122). Cloudera highly recommends upgrading Apache Impala (Impala) and Apache Hive (Hive) in CDW to 2022.0.11.0-122 or higher versions to ensure performance, stability and/or cloud cost are not impacted by this behavior.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-656: https://my.cloudera.com/knowledge/TSB-2023-656-Cloudera-Data-Warehouse-does-not-clean-up-old?id=368733

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures to/from prometheus to any other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-6619 Browser auto-close not working on some browsers after token-Based authentication for accessing CDW

The Firefox and Edge browser window does not close automatically after successful authentication.

DWX-9774 Database Catalog or Virtual Warehouse image version problem

Background: In Cloudera Data Warehouse 2021.0.3-b27 - 2021.0.5-b36, you can choose any supported image version when you create a Database Catalog or Virtual Warehouse, assuming you have the CDW_VERSIONED_DEPLOY entitlement.

Problem: In Cloudera Data Warehouse 2021.0.6-b96, you can choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse only if you use an existing environment. An existing environment is one you created in 2021.0.3-b27 - 2021.0.5-b36.

Workaround: In 2021.0.6-b96, use an environment you created before this release to choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-8573: EKS upgrade from DWX UI to K8s v1.20 fails in reduced permissions mode

In reduced permissions mode, attempting to update Amazon Elastic Kubernetes Service (EKS) to Kubernetes version 1.20 fails with an AccessDenied error.

Workaround:In AWS, add the following permissions to your reduced permissions mode JSON IAM permissions policy right after "iam:PutRolePolicy":

```
{
       ...
       "cloudformation:GetTemplate",
```

```
        "cloudformation:GetTemplateSummary",
        "iam:GetRole",
        "eks:ListUpdates",
        "ec2:CreateLaunchTemplateVersion",
        "ec2:DescribeLaunchTemplateVersions",
        "ec2:DescribeLaunchTemplates",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:DeleteAutoScalingGroup",
        "ec2:RunInstances",
        "autoscaling:DescribeScalingActivities",
        "autoscaling:TerminateInstanceInAutoScalingGroup",
        "autoscaling:DescribeScheduledActions",
        "autoscaling:SetDesiredCapacity",
        "iam:PassRole",
        "rds:DescribeDBInstances",
        "ec2:DescribeInstances"
        ...
```

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the Create Tags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.

3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 2022.0.9.0-120 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click Edit Diagnostic Bundle. Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Hue Query Editor

TSB 2023-670: Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed: kubelet Error: failed to create containerd task: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: “run_httpd.sh”: cannot run executable found relative to current directory: unknown

The root cause of the issue is a recent node image upgrade in Azure made by Microsoft, where the version of the container runtime (containerd) was upgraded to 1.6.18. This version upgraded Go SDK to 1.19.6, where the behavior of how program execution works has changed. Due to security concerns, the newer version of Go SDK does not resolve a program using an implicit or explicit path entry relative to the current directory (for more information, see the Go documentation), and containerd indirectly uses this Go exec API. The Docker command of the Hue frontend relies on the former behavior. Therefore the command fails on recent node images, because it cannot find the expected executable in the current working directory of the containers anymore. Cloudera first noticed the aforementioned containerd version in the Azure Kubernetes Service (AKS) released on 2023-03-05. Every node image which is newer than the version released on 2023-03-05 is affected by this issue. Unfortunately, Cloudera does not have any insight to when Microsoft rolls out new node images in a given region. Therefore, it is possible that in some regions, the older node images are still in use, where the issue does not arise until a given regional update is applied to the node images.

CDW environments, which are already activated and running, are not affected as long as customers do not trigger a node image upgrade to the latest available version either using Azure Command Line Interface (CLI) or on the Azure portal. If the node image is upgraded to the latest version, then the Virtual Warehouses also need to be upgraded to the latest CDWH-2023.0.14.0 version. Newly created CDW environments are not affected, but customers are advised to not choose a Hue version lower than CDWH-2023.0.14.0 during any Virtual Warehouse creation because such a configuration is affected by this issue.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-670: TSB Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Carried over from the previous release: Database Catalog

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 2022.0.9.0-120 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either one of these CDW versions:
  - 2022.0.8.0-89 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 2022.0.9.0-120 (released August 4, 2022).

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

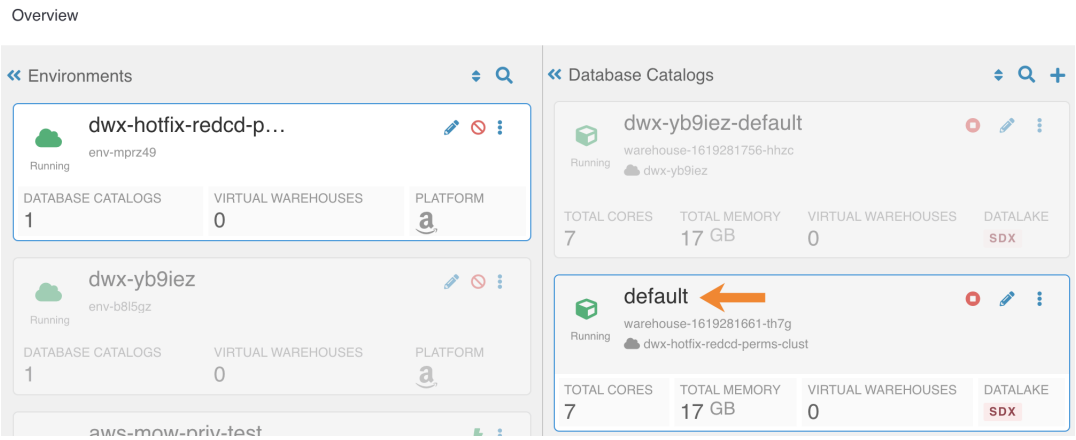

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

DWX-10271: Missing log section in Hive query results

In a Hive Virtual Warehouse, when you run a query in Hue, the query results do not contain a logs section.

DWX-8118: INSERT INTO command fails under certain circumstances

This problem affects users who have a PostgreSQL database as the backend Hive database. If you create a table A and create a table B as select (CTAS) from an empty table A, inserting values into table B fails as follows:

```
Error while compiling statement: FAILED: Execution Error, return
 code 1 from
       org.apache.hadoop.hive.ql.exec.StatsTask.org.apache.thr
ift.transport.TTransportException
```

Workaround: Disable auto-stats gathering: Go to Cloudera Manager Data Warehouse Virtual Warehouses , and click the Hive VW name in the list. In Configuration HiveServer2 , set hive.cbo.enable to false.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses

must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Impala Virtual Warehouse

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

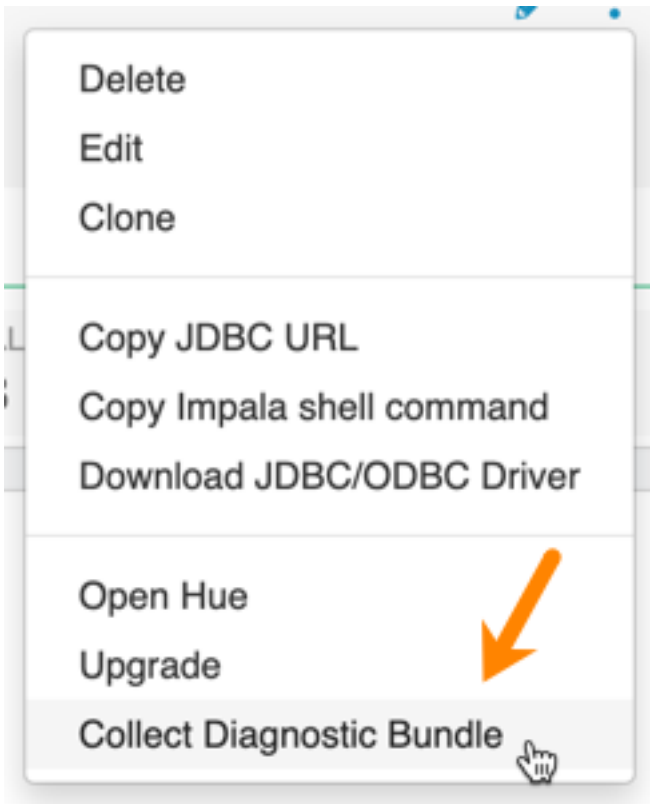

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case so the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

August 4, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New known Issues in this release

Do not upgrade this release to AKS 1.22

The latest version of Cloudera Data Warehouse (CDW) 1.4.2-b118 (released August 4, 2022) provisions AKS 1.22 by default in Azure environments; however. upgrading your activated environments to AKS 1.22 or later is not supported. Do not upgrade your AKS clusters to AKS 1.22.

DWX-13103 Cloudera Data Warehouse environment activation problem

When CDW environments are activated, a race condition can occur between the prometheus pod and istiod pod. The prometheus pod can be set up without an istio-proxy container, causing communication failures between prometheus and other pods in the Kubernetes cluster. Data Warehouse prometheus-related functionalities, such as autoscaling, stop working. Grafana dashboards, which get metrics from prometheus, are not populated.

Workaround: Restart the prometheus pod so that it gets the istio-proxy container.

DWX-13094 Diagnostic bundle job fails with missing service account

Due to a change to the name of the service accounts, after upgrading to DWX-1.4.2, the diagnostic bundle job can fail if you used the bundle in earlier versions. Due to a recent. Jobs from the latest

version are trying to run with the new name of the service account, but these are not reinstalled after the upgrade.

Workaround: 1) Create the new service account the job is looking for (diagnostic-data-generator-service-account) as follows:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: diagnostic-data-generator-service-account
  namespace: monitor
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: ddg-role-binding-custom
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: diagnostic-data-generator-role-monitor
subjects:
- kind: ServiceAccount
  name: diagnostic-data-generator-service-account
  namespace: monitor
```

2) Bind the service account to the cluster role that grants the necessary permissions to the job using the following command:

```
kubectl create -f <filename>
```

Non-default Database Catalogs created with several earlier CDW versions fails

This issue affects you only if you meet the following conditions:

- You have a versioned CDW deployment and the multi default DBC entitlement.
- You are using this release of CDW version 1.4.2-b118 (released August 4, 2022).
- You added Database Catalogs (not the automatically generated default Database Catalog) using either of these CDW versions:
  - 1.4.1-b86 (released June, 22, 2022)
  - 2022-0.7.1-2 (released May 10, 2022)

Do not attempt to upgrade the Database Catalog you created with these earlier releases. The Database Catalog will fail. Your existing Database Catalog created with the earlier release works fine with CDW Runtime. Recreating your existing Database Catalog updates CDW Runtime for 1.4.2-b118 (released August 4, 2022).

Workloads from earlier CDW versions cannot be deployed on new Azure environments

Because new environments are provisioned automatically to use Azure Kubernetes Service (AKS) 1.22, deprecated APIs used in workloads of earlier versions are not supported in this version 1.4.2-b118 (released August 4, 2022). This issue affects you only if you have the CDW_VERSIONED_DEPLOY entitlement.

Workaround: Create new workloads. Workloads in Database Catalogs, Hive, Impala, Hue, Data Visualization, and are incompatible with AKS 1.22.

CDPD-40730 Parquet change can cause incompatibility

Parquet files written by the parquet-mr library in this version of CDW, where the schema contains a timestamp with no UTC conversion will not be compatible with older versions of Parquet readers. The effect is that the older versions will still consider these timestamps as they would require UTC conversions and will thus end up with a wrong result. You can encounter this problem only when

you write Parquet-based tables using Hive, and tables have the non-default configuration hive.parquet.write.int64.timestamp=true.

Carried over from the previous release: General

TSB 2023-656: TSB Cloudera Data Warehouse does not clean up old Helm releases

Cloudera Data Warehouse (CDW) uses Helm release manager to deploy component releases into the Kubernetes cluster of the customer. However, CDW's current behavior in CDW keeps the Helm release history without any boundary. Since the storage for Helm in CDW uses secrets, this behavior can lead to a very high number of secrets (Impala/Hive scale-up/down creates new releases;clusters were seen with 6000+ secrets). Therefore, parsing and decoding a high number of Helm releases can require a significant amount of memory. As an example, for a cluster with 6000+ secrets, it could be 6-7 GB or more of memory utilized. This may result in performance degradation, instability and/or higher cloud cost because several CDW cluster components rely on secrets, which would cause the memory footprint of these other components to increase significantly.

This issue is addressed in the CDW version 1.5.1-b110 (workload image version 2022.0.11.0-122). Cloudera highly recommends upgrading Apache Impala (Impala) and Apache Hive (Hive) in CDW to 2022.0.11.0-122 or higher versions to ensure performance, stability and/or cloud cost are not impacted by this behavior.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-656: https://my.cloudera.com/knowledge/TSB-2023-656-Cloudera-Data-Warehouse-does-not-clean-up-old?id=368733

DWX-6619 Browser auto-close not working on some browsers after token-Based authentication for accessing CDW

The Firefox and Edge browser window does not close automatically after successful authentication.

DWX-9774 Database Catalog or Virtual Warehouse image version problem

Background: In Cloudera Data Warehouse 2021.0.3-b27 - 2021.0.5-b36, you can choose any supported image version when you create a Database Catalog or Virtual Warehouse, assuming you have the CDW_VERSIONED_DEPLOY entitlement.

Problem: In Cloudera Data Warehouse 2021.0.6-b96, you can choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse only if you use an existing environment. An existing environment is one you created in 2021.0.3-b27 - 2021.0.5-b36.

Workaround: In 2021.0.6-b96, use an environment you created before this release to choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"
```

```
        # Save and quit the editor, and then run the following
command to check your changes.

        kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

        # Make sure that the annotation you added is present.
```

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-8573: EKS upgrade from DWX UI to K8s v1.20 fails in reduced permissions mode

In reduced permissions mode, attempting to update Amazon Elastic Kubernetes Service (EKS) to Kubernetes version 1.20 fails with an AccessDenied error.

Workaround:In AWS, add the following permissions to your reduced permissions mode JSON IAM permissions policy right after "iam:PutRolePolicy":

```
{
        ...
        "cloudformation:GetTemplate",
        "cloudformation:GetTemplateSummary",
        "iam:GetRole",
        "eks:ListUpdates",
        "ec2:CreateLaunchTemplateVersion",
        "ec2:DescribeLaunchTemplateVersions",
        "ec2:DescribeLaunchTemplates",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:DeleteAutoScalingGroup",
        "ec2:RunInstances",
        "autoscaling:DescribeScalingActivities",
        "autoscaling:TerminateInstanceInAutoScalingGroup",
        "autoscaling:DescribeScheduledActions",
        "autoscaling:SetDesiredCapacity",
        "iam:PassRole",
        "rds:DescribeDBInstances",
        "ec2:DescribeInstances"
        ...
```

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
        operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the Create Tags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle. Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Hue Query Editor

TSB 2023-670: Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed: kubelet Error: failed to create containerd task: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: “run_httpd.sh”: cannot run executable found relative to current directory: unknown

The root cause of the issue is a recent node image upgrade in Azure made by Microsoft, where the version of the container runtime (containerd) was upgraded to 1.6.18. This version upgraded Go SDK to 1.19.6, where the behavior of how program execution works has changed. Due to security concerns, the newer version of Go SDK does not resolve a program using an implicit or explicit path entry relative to the current directory (for more information, see the Go documentation), and containerd indirectly uses this Go exec API. The Docker command of the Hue frontend relies on the former behavior. Therefore the command fails on recent node images, because it cannot find the expected executable in the current working directory of the containers anymore. Cloudera first noticed the aforementioned containerd version in the Azure Kubernetes Service (AKS) released on 2023-03-05. Every node image which is newer than the version released on 2023-03-05 is affected by this issue. Unfortunately, Cloudera does not have any insight to when Microsoft rolls out new node images in a given region. Therefore, it is possible that in some regions, the older node images are still in use, where the issue does not arise until a given regional update is applied to the node images.

CDW environments, which are already activated and running, are not affected as long as customers do not trigger a node image upgrade to the latest available version either using Azure Command Line Interface (CLI) or on the Azure portal. If the node image is upgraded to the latest version, then the Virtual Warehouses also need to be upgraded to the latest CDWH-2023.0.14.0 version. Newly created CDW environments are not affected, but customers are advised to not choose a Hue

version lower than CDWH-2023.0.14.0 during any Virtual Warehouse creation because such a configuration is affected by this issue.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-670: TSB Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Carried over from the previous release: Database Catalog

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

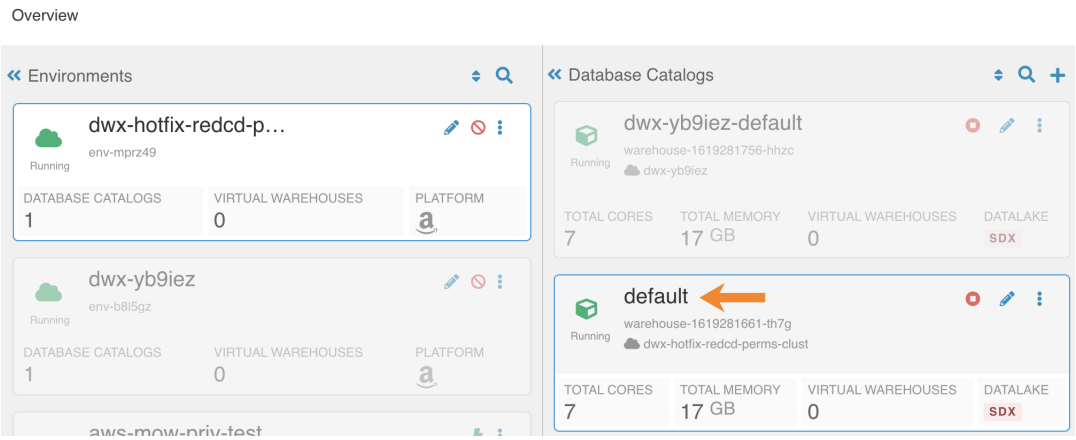

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

DWX-10271: Missing log section in Hive query results

In a Hive Virtual Warehouse, when you run a query in Hue, the query results do not contain a logs section.

DWX-8118: INSERT INTO command fails under certain circumstances

This problem affects users who have a PostgreSQL database as the backend Hive database. If you create a table A and create a table B as select (CTAS) from an empty table A, inserting values into table B fails as follows:

```
Error while compiling statement: FAILED: Execution Error, return
 code 1 from
       org.apache.hadoop.hive.ql.exec.StatsTask.org.apache.thr
ift.transport.TTransportException
```

Workaround: Disable auto-stats gathering: Go to Cloudera Manager Data Warehouse Virtual Warehouses , and click the Hive VW name in the list. In Configuration HiveServer2 , set hive.cbo .enable to false.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Impala Virtual Warehouse

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

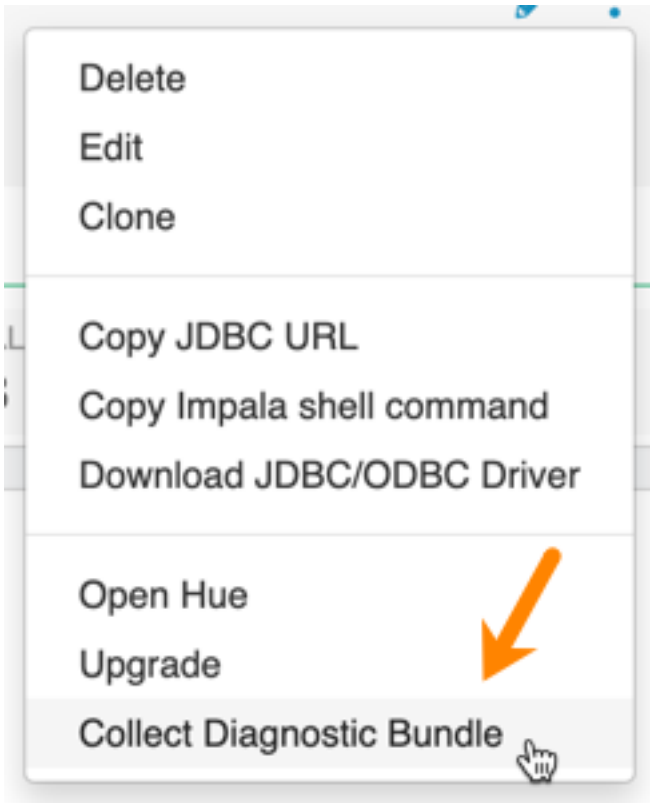

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case so the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

June 22, 2022

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following known issues:

New known Issues in this release

DWX-12758 DBC upgrade fails for some source versions

In this release, if you are using a Database Catalog that was created in CDW 2022.0.7.0-80 (released April 27, 2022) or 2022.0.7.1-2 (released May 10, 2022) a failure can occur when you try to upgrade the Database Catalog.

DWX-12528 Tags are not removed for environment activated in reduced permissions mode

You see a DWX server log error, "... error delete tags" when in reduced permissions mode. You can ignore this error.

DWX-12658 Rebuild materialized view after Database Catalog upgrade

You must rebuild materialized views created in version of the Database Catalog that is older than the version of your Virtual Warehouse. This is expected behavior.

Workaround: Run the following command to rebuild a materialized view after moving to a Virtual Warehouse based on a later version of the Database Catalog:

```
ALTER MATERIALIZED VIEW mv1 REBUILD;
```

DWX-12598 DESCRIBE EXTENDED does not show bucketing information about an Iceberg table.

This issue affects the Hive and Impala Virtual Warehouses. The Hive Virtual Warehouse, which supports DESCRIBE FORMATTED of Iceberg talble, has the same problem

CDPD-40749 IMPALA-11346 Iceberg query failure

When a partitioned legacy table is converted to Iceberg, Impala queries of the table might fail. The problem occurs when the query contains a WHERE clause plus a predicate on the partition columns.

Workaround: Replace Iceberg tables created prior to this GA release with new tables. Create a new table in Iceberg by selecting the old table as shown in Test driving Iceberg from Impala.

Carried over from the previous release: General

TSB 2023-656: TSB Cloudera Data Warehouse does not clean up old Helm releases

Cloudera Data Warehouse (CDW) uses Helm release manager to deploy component releases into the Kubernetes cluster of the customer. However, CDW's current behavior in CDW keeps the Helm release history without any boundary. Since the storage for Helm in CDW uses secrets, this behavior can lead to a very high number of secrets (Impala/Hive scale-up/down creates new releases;clusters were seen with 6000+ secrets). Therefore, parsing and decoding a high number of Helm releases can require a significant amount of memory. As an example, for a cluster with 6000+ secrets, it could be 6-7 GB or more of memory utilized. This may result in performance degradation, instability and/or higher cloud cost because several CDW cluster components rely on secrets, which would cause the memory footprint of these other components to increase significantly.

This issue is addressed in the CDW version 1.5.1-b110 (workload image version 2022.0.11.0-122). Cloudera highly recommends upgrading Apache Impala (Impala) and Apache Hive (Hive) in CDW to 2022.0.11.0-122 or higher versions to ensure performance, stability and/or cloud cost are not impacted by this behavior.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-656: https://my.cloudera.com/knowledge/TSB-2023-656-Cloudera-Data-Warehouse-does-not-clean-up-old?id=368733

DWX-6619 Browser auto-close not working on some browsers after token-Based authentication for accessing CDW

The Firefox and Edge browser window does not close automatically after successful authentication.

DWX-9774 Database Catalog or Virtual Warehouse image version problem

Background: In Cloudera Data Warehouse 2021.0.3-b27 - 2021.0.5-b36, you can choose any supported image version when you create a Database Catalog or Virtual Warehouse, assuming you have the CDW_VERSIONED_DEPLOY entitlement.

Problem: In Cloudera Data Warehouse 2021.0.6-b96, you can choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse only if you use an existing environment. An existing environment is one you created in 2021.0.3-b27 - 2021.0.5-b36.

Workaround: In 2021.0.6-b96, use an environment you created before this release to choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Carried over from the previous release: AWS

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-8573: EKS upgrade from DWX UI to K8s v1.20 fails in reduced permissions mode

In reduced permissions mode, attempting to update Amazon Elastic Kubernetes Service (EKS) to Kubernetes version 1.20 fails with an AccessDenied error.

Workaround:In AWS, add the following permissions to your reduced permissions mode JSON IAM permissions policy right after "iam:PutRolePolicy":

```
{
        ...
        "cloudformation:GetTemplate",
        "cloudformation:GetTemplateSummary",
        "iam:GetRole",
        "eks:ListUpdates",
        "ec2:CreateLaunchTemplateVersion",
        "ec2:DescribeLaunchTemplateVersions",
        "ec2:DescribeLaunchTemplates",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:DeleteAutoScalingGroup",
        "ec2:RunInstances",
        "autoscaling:DescribeScalingActivities",
        "autoscaling:TerminateInstanceInAutoScalingGroup",
        "autoscaling:DescribeScheduledActions",
        "autoscaling:SetDesiredCapacity",
        "iam:PassRole",
        "rds:DescribeDBInstances",
        "ec2:DescribeInstances"
        ...
```

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
       operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
       Description: DynamoDB table name for the SDX S3 file lis
tings, created through S3Guard
       Type: String
       Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the Create Tags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
          "Statement": [
          {
          "Action": [
          "autoscaling:DescribeAutoScalingGroups",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:SetDesiredCapacity",
          "autoscaling:TerminateInstanceInAutoScalingGroup",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:CreateTags"
          ],
```

7. Save changes.

Carried over from the previous release: Azure

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click ⋮ Edit Diagnostic Bundle. Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses

old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Carried over from the previous release: Hue Query Editor

TSB 2023-670: Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed: kubelet Error: failed to create containerd task: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: “run_httpd.sh”: cannot run executable found relative to current directory: unknown

The root cause of the issue is a recent node image upgrade in Azure made by Microsoft, where the version of the container runtime (containerd) was upgraded to 1.6.18. This version upgraded Go SDK to 1.19.6, where the behavior of how program execution works has changed. Due to security concerns, the newer version of Go SDK does not resolve a program using an implicit or explicit path entry relative to the current directory (for more information, see the Go documentation), and containerd indirectly uses this Go exec API. The Docker command of the Hue frontend relies on the former behavior. Therefore the command fails on recent node images, because it cannot find the expected executable in the current working directory of the containers anymore. Cloudera first noticed the aforementioned containerd version in the Azure Kubernetes Service (AKS) released on 2023-03-05. Every node image which is newer than the version released on 2023-03-05 is affected by this issue. Unfortunately, Cloudera does not have any insight to when Microsoft rolls out new node images in a given region. Therefore, it is possible that in some regions, the older node images are still in use, where the issue does not arise until a given regional update is applied to the node images.

CDW environments, which are already activated and running, are not affected as long as customers do not trigger a node image upgrade to the latest available version either using Azure Command Line Interface (CLI) or on the Azure portal. If the node image is upgraded to the latest version, then the Virtual Warehouses also need to be upgraded to the latest CDWH-2023.0.14.0 version. Newly created CDW environments are not affected, but customers are advised to not choose a Hue version lower than CDWH-2023.0.14.0 during any Virtual Warehouse creation because such a configuration is affected by this issue.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-670: TSB Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.

2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Carried over from the previous release: Database Catalog

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

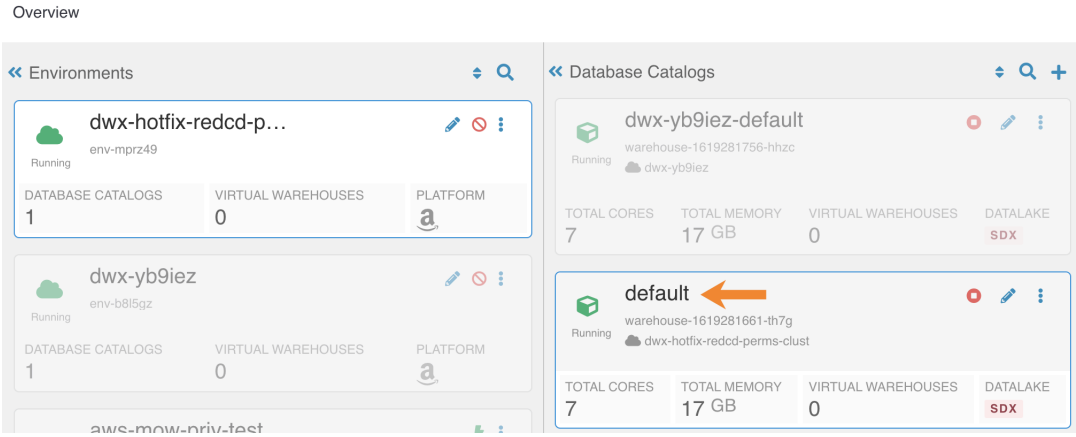

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Carried over from the previous release: Hive Virtual Warehouse

DWX-10271: Missing log section in Hive query results

In a Hive Virtual Warehouse, when you run a query in Hue, the query results do not contain a logs section.

DWX-8118: INSERT INTO command fails under certain circumstances

This problem affects users who have a PostgreSQL database as the backend Hive database. If you create a table A and create a table B as select (CTAS) from an empty table A, inserting values into table B fails as follows:

```
Error while compiling statement: FAILED: Execution Error, return
 code 1 from
       org.apache.hadoop.hive.ql.exec.StatsTask.org.apache.thr
ift.transport.TTransportException
```

Workaround: Disable auto-stats gathering: Go to Cloudera Manager Data Warehouse Virtual Warehouses , and click the Hive VW name in the list. In Configuration HiveServer2 , set hive.cbo .enable to false.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.
8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
       match the certificate subject provided by the peer (CN=*
.env-c25dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Carried over from the previous release: Impala Virtual Warehouse

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

       # Add the following under the metadata.annotations field
       service.beta.kubernetes.io/aws-load-balancer-ssl-negoti
ation-policy: "ELBSecurityPolicy-TLS-1-2-2017-01"

       # Save and quit the editor, and then run the following
command to check your changes.

       kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

       # Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

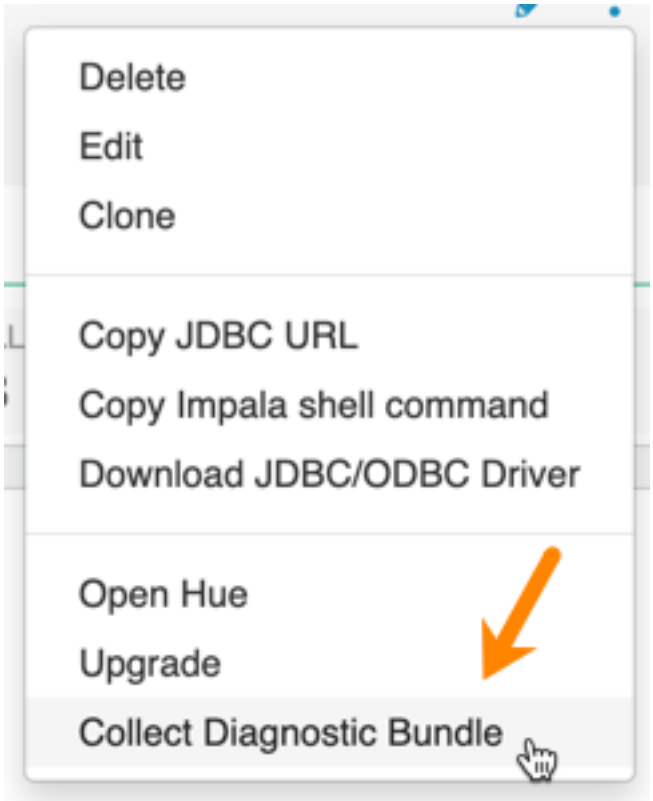

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case so the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

General known issues on public clouds

Learn about the general known issues in Cloudera Data Warehouse (CDW) service on public clouds, the impact or changes to the functionality, and the workaround.

TSB 2023-656: TSB Cloudera Data Warehouse does not clean up old Helm releases

Cloudera Data Warehouse (CDW) uses Helm release manager to deploy component releases into the Kubernetes cluster of the customer. However, CDW's current behavior in CDW keeps the Helm release history without any boundary. Since the storage for Helm in CDW uses secrets, this behavior can lead to a very high number of secrets (Impala/Hive scale-up/down creates new releases;clusters were seen with 6000+ secrets). Therefore, parsing and decoding a high number of Helm releases can require a significant amount of memory. As an example, for a cluster with 6000+ secrets, it could be 6-7 GB or more of memory utilized. This may result in performance degradation, instability and/or higher cloud cost because several CDW cluster components rely on secrets, which would cause the memory footprint of these other components to increase significantly.

This issue is addressed in the CDW version 1.5.1-b110 (workload image version 2022.0.11.0-122). Cloudera highly recommends upgrading Apache Impala (Impala) and Apache Hive (Hive) in CDW to 2022.0.11.0-122 or higher versions to ensure performance, stability and/or cloud cost are not impacted by this behavior.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-656: https://my.cloudera.com/knowledge/TSB-2023-656-Cloudera-Data-Warehouse-does-not-clean-up-old?id=368733

DWX-6619 Browser auto-close not working on some browsers after token-Based authentication for accessing CDW

The Firefox and Edge browser window does not close automatically after successful authentication.

DWX-9774 Database Catalog or Virtual Warehouse image version problem

Background: In Cloudera Data Warehouse 2021.0.3-b27 - 2021.0.5-b36, you can choose any supported image version when you create a Database Catalog or Virtual Warehouse, assuming you have the CDW_VERSIONED_DEPLOY entitlement.

Problem: In Cloudera Data Warehouse 2021.0.6-b96, you can choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse only if you use an existing environment. An existing environment is one you created in 2021.0.3-b27 - 2021.0.5-b36.

Workaround: In 2021.0.6-b96, use an environment you created before this release to choose an image version other than 2021.0.6-b96 when you create a Database Catalog or Virtual Warehouse.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

# Add the following under the metadata.annotations field
service.beta.kubernetes.io/aws-load-balancer-ssl-negotiation-pol
icy: "ELBSecurityPolicy-TLS-1-2-2017-01"

# Save and quit the editor, and then run the following command to
 check your changes.

kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

# Make sure that the annotation you added is present.
```

DWX-5742: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails

Problem: Upgrading multiple Hive and Impala Virtual Warehouses or Database Catalogs at the same time fails.

Workaround: If you need to upgrade or create multiple Hive and Impala Virtual Warehouses or Database Catalogs, perform the upgrade or creation sequentially one at a time.

Known issues in CDW running on AWS environments

Learn about the known issues in Cloudera Data Warehouse service in AWS environments, the impact or changes to the functionality, and workarounds.

AWS availability zone inventory issue

In this release, you can select a preferred availability zone when you create a Virtual Warehouse; however, AWS might not be able to provide enough compute instances of the type that Cloudera Data Warehouse needs.

Workaround: If you experience this AWS issue, try recreating the Virtual Warehouse and choosing a different availability zone.

DWX-8573: EKS upgrade from DWX UI to K8s v1.20 fails in reduced permissions mode

In reduced permissions mode, attempting to update Amazon Elastic Kubernetes Service (EKS) to Kubernetes version 1.20 fails with an AccessDenied error.

Workaround:In AWS, add the following permissions to your reduced permissions mode JSON IAM permissions policy right after "iam:PutRolePolicy":

```
{
 ...
                "cloudformation:GetTemplate",
                "cloudformation:GetTemplateSummary",
                "iam:GetRole",
                "eks:ListUpdates",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DescribeLaunchTemplates",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DeleteAutoScalingGroup",
                "ec2:RunInstances",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:TerminateInstanceInAutoScalingGrou
p",
                "autoscaling:DescribeScheduledActions",
                "autoscaling:SetDesiredCapacity",
                "iam:PassRole",
                "rds:DescribeDBInstances",
                "ec2:DescribeInstances"
 ...
```

DWX-7613: CloudFormation stack creation using AWS CLI broken for CDW Reduced Permissions Mode

Problem: If you use the AWS CLI to create a CloudFormation stack to activate an AWS environment for use in Reduced Permissions Mode, it fails and returns the following error:

```
An error occurred (ValidationError) when calling the CreateStack
 operation: Parameters: [SdxDDBTableName] must have values
```

The default value of SdxDDBTableName is not being set. If you create the CloudFormation stack using the AWS Console, there is no problem.

Workaround: If you must use the AWS CLI, edit the CloudFormation stack template file as follows:

```
SdxDDBTableName:
    Description: DynamoDB table name for the SDX S3 file listin
gs, created through S3Guard
    Type: String
    Default: " "
```

Then rerun the CloudFormation stack creation command using the AWS CLI.

ENGESC-8271: Helm 2 to Helm 3 migration fails on AWS environments where the overlay network feature is in use and namespaces are stuck in a terminating state

Problem: While using the overlay network feature for AWS environments and after attempting to migrate an AWS environment from Helm 2 to Helm 3, the migration process fails.

Workaround: Run the following kubectl command to determine whether you have any namespaces stuck in a terminating state:

```
kubectl get ns
```

Then contact Cloudera Technical Support to report and get help on this issue.

DWX-7349: In reduced permissions mode, default Database Catalog name does not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

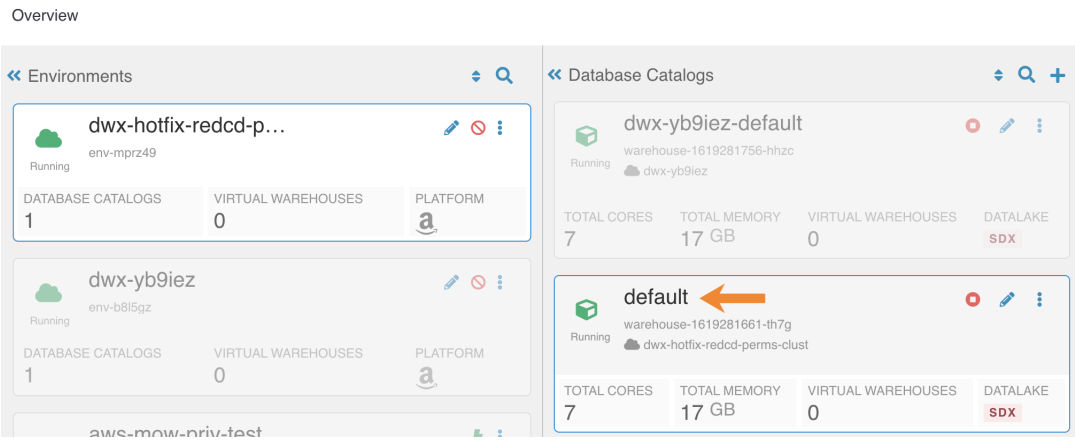

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6970: Tags do not get applied in existing CDW environments

Problem: You may see the following error while trying to apply tags to Virtual Warehouses in an existing CDW environment: An error occurred (UnauthorizedOperation) when calling the CreatetTags operation: You are not authorized to perform this operation and Compute node tagging was unsuccessful. This happens because the ec2:CreateTags privilege is missing from your AWS cluster-autoscaler inline policy for the NodeInstanceRole role.

Workaround: Add the ec2:CreateTags privilege to the cluster-autoscaler inline policy as follows:

1. Log into the AWS IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation panel, choose Roles.
3. Search the list of roles for NodeInstanceRole.
4. Click Permissions.
5. Select cluster-autoscaler and click Edit policy.
6. Add the ec2:CreateTags line in the Actions section after the ec2:DescribeLaunchTemplateVersions line as shown:

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeTags",
                "autoscaling:DescribeLaunchConfigurations",
```

```
                    "autoscaling:SetDesiredCapacity",
                    "autoscaling:TerminateInstanceInAutoScaling
Group",
                    "ec2:DescribeLaunchTemplateVersions",
                    "ec2:CreateTags"
                ],
```

7. Save changes.

Known issues in CDW running on Azure environments

Learn about the known issues in Cloudera Data Warehouse service in Azure environments, the impact or changes to the functionality, and the workaround.

Incorrect diagnostic bundle location

Problem:The path you see to the diagnostic bundle is wrong when you create a Virtual Warehouse, collect a diagnostic bundle of log files for troubleshooting, and click Edit Diagnostic Bundle . Your storage account name is missing from the beginning of the path.

Workaround:To find your diagnostic bundle, add your storage account name to the beginning of the path, for example:

```
my-storage-account-name/log-files/clusters/my-env/my-warehouse-x
x-yy/warehouse/debug-artifacts/hive/compute-zz-mvvf/compute-zz-m
vvf-0926210111-0927090111-01234.zip
```

To get your storage account name, in Cloudera Management Console, click Environments, select your environment, and scroll down to Logs Storage and Audits. The first part of the path is your storage account name.

Changed environment credentials not propagated to AKS

Problem: When you change the credentials of a running cloud environment using the Management Console, the changes are not automatically propagated to the corresponding active Cloudera Data Warehouse (CDW) environment. As a result, the Azure Kubernetes Service (AKS) uses old credentials and may not function as expected resulting in inaccessible Hive or Impala Virtual Warehouses.

Workaround: To resolve this issue, you must manually synchronize the changes with the CDW AKS resources. To synchronize the updated credentials, see Update AKS cluster with new service principal credentials in the Azure product documentation.

Known issues in Data Analytics Studio on public clouds

Learn about the known issues related to Data Analytics Studio (DAS) in Cloudera Data Warehouse service on public clouds, the impact or changes to the functionality, and the workaround.

DWX-929: DAS UI displays the internal JDBC URL.

Problem: DAS displays the internal JDBC URL on its About page instead of the correct JDBC URL to use to connect to the data warehouse.

Workaround: To copy the correct JDBC URL to use to connect to the data warehouse, in the Data Warehouse service Overview page, go to Virtual Warehouse , and then click Copy JDBC URL.

DWX-2592: DAS cannot parse certain characters in strings and comments.

Problem: DAS cannot parse semicolons (;) and double hyphens (--) in strings and comments. For example if you have a semicolon in a query such as the following, the query might fail:

```
SELECT * FROM properties WHERE prop_value = "name1;name2";
```

Queries with double hyphens (--) might also fail. For example:

```
SELECT * FROM test WHERE option = '--name';
```

Workaround: If a semicolon is present in a comment, then remove the semicolon before running the query or remove the comment entirely. For example:

```
SELECT * FROM test; -- SELECT * FROM test;
```

Should be changed to:

```
SELECT * FROM test; /* comment; comment */
```

In the same manner, remove any double-hyphens before running queries to avoid failure in DAS.

Older versions of Google Chrome browser might cause issues.

Problem: You might experience problems while using faceted search in older versions of the Google Chrome browser.

Workaround: Use the latest version (71.x or later) of Google Chrome.

BUG-94611: Visual Explain for the same query shows different graphs.

Problem: Visual Explain for the same query shows different graphs on the Compose page and the Query Details page.

Workaround: N/A

Known issues in Hue on public clouds

Learn about the known issues related to the Hue query editor in Cloudera Data Warehouse (CDW) service on public clouds, the impact or changes to the functionality, and the workaround.

TSB 2023-670: Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

Hue frontend for Apache Impala (Impala) and Apache Hive (Hive) Virtual Warehouses created in Cloudera Data Warehouse (CDW) can be stuck in a CrashLoopBackoff state on Microsoft Azure (Azure) platform, making it impossible to reach the Virtual Warehouse through Hue. In this case, the following error message is displayed: kubelet Error: failed to create containerd task: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: “run_httpd.sh”: cannot run executable found relative to current directory: unknown

The root cause of the issue is a recent node image upgrade in Azure made by Microsoft, where the version of the container runtime (containerd) was upgraded to 1.6.18. This version upgraded Go SDK to 1.19.6, where the behavior of how program execution works has changed. Due to security concerns, the newer version of Go SDK does not resolve a program using an implicit or explicit path entry relative to the current directory (for more information, see the Go documentation), and containerd indirectly uses this Go exec API. The Docker command of the Hue frontend relies on the former behavior. Therefore the command fails on recent node images, because it cannot find the expected executable in the current working directory of the containers anymore. Cloudera first noticed the aforementioned containerd version in the Azure Kubernetes Service (AKS) released on 2023-03-05. Every node image which is newer than the version released on 2023-03-05 is affected by this issue. Unfortunately, Cloudera does not have any insight to when Microsoft rolls out new node images in a given region. Therefore, it is possible that in some regions, the older node images

are still in use, where the issue does not arise until a given regional update is applied to the node images.

CDW environments, which are already activated and running, are not affected as long as customers do not trigger a node image upgrade to the latest available version either using Azure Command Line Interface (CLI) or on the Azure portal. If the node image is upgraded to the latest version, then the Virtual Warehouses also need to be upgraded to the latest CDWH-2023.0.14.0 version. Newly created CDW environments are not affected, but customers are advised to not choose a Hue version lower than CDWH-2023.0.14.0 during any Virtual Warehouse creation because such a configuration is affected by this issue.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-670: TSB Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

IMPALA-11447 Selecting certain complex types in Hue crashes Impala

Queries that have structs/arrays in the select list crash Impala if initiated by Hue.

Workaround: Do not select structs/arrays in Hue.

DWX-8460: Unable to delete, move, or rename directories within the S3 bucket from Hue

Problem: You may not be able to rename, move, or delete directories within your S3 bucket from the Hue web interface. This is because of an underlying issue, which will be fixed in a future release.

Workaround: You can move, rename, or delete a directory using the HDFS commands as follows:

1. SSH into your CDP environment host.
2. To delete a directory within your S3 bucket, run the following command:

```
hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
TORY-NAME***]
```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

```
hdfs dfs -mkdir [***DIRECTORY-NAME***]
```

```
hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
IRECTORY***]
```

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5650: Hue only makes the first user a superuser for all Virtual Warehouses within a Data Catalog

Problem: Hue marks the user that logs in to Hue from a Virtual Warehouse for the first time as the Hue superuser. But if multiple Virtual Warehouses are connected to a single Data Catalog, then the first user that logs in to any one of the Virtual Warehouses within that Data Catalog is the Hue superuser.

For example, consider that a Data Catalog DC-1 has two Virtual Warehouses VW-1 and VW-2. If a user named John logs in to Hue from VW-1 first, then he becomes the Hue superuser for all the Virtual Warehouses within DC-1. At this time, if Amy logs in to Hue from VW-2, Hue does not make her a superuser within VW-2.

None.

Known issues in Database Catalog on public clouds

Learn about the known issues related to Database Catalog in Cloudera Data Warehouse (CDW) service on public clouds, the impact or changes to the functionality, and the workaround.

DWX-7349: In reduced permissions mode, default Database Catalog name do not include the environment name

Problem:

When you activate an AWS environment in reduced permissions mode, the default Database Catalog name does not include the environment name:

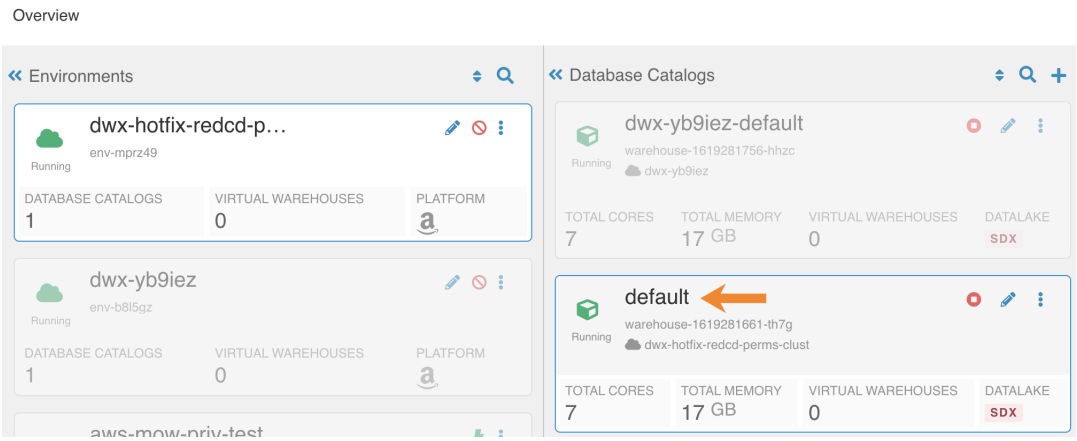

This does not cause collisions because each Database Catalog named "default" is associated with a different environment. For more information about reduced permissions mode, see Reduced permissions mode for AWS environments.

Workaround: None available.

DWX-6167: Maximum connections reached when creating multiple Database Catalogs

Problem:After creating 17 Database Catalogs on one AWS environment, Virtual Warehouses failed to start.

Workaround: Limit the number of Database Catalogs created on one environment to 5. This applies to both AWS and Azure environments.

Known issues in Hive Virtual Warehouses on public clouds

Learn about the known issues related to Hive Virtual Warehouse in Cloudera Data Warehouse (CDW) service on public clouds, the impact or changes to the functionality, and the workaround.

DWX-10271: Missing log section in Hive query results

In a Hive Virtual Warehouse, when you run a query in Hue, the query results do not contain a logs section.

DWX-8118: INSERT INTO command fails under certain circumstances

This problem affects users who have a PostgreSQL database as the backend Hive database. If you create a table A and create a table B as select (CTAS) from an empty table A, inserting values into table B fails as follows:

```
Error while compiling statement: FAILED: Execution Error, return
 code 1 from
          org.apache.hadoop.hive.ql.exec.StatsTask.org.apache.
thrift.transport.TTransportException
```

Workaround: Disable auto-stats gathering: Go to Cloudera Manager Data Warehouse Virtual Warehouses , and click the Hive VW name in the list. In Configuration HiveServer2 , set hive.cbo .enable to false.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

# Add the following under the metadata.annotations field
service.beta.kubernetes.io/aws-load-balancer-ssl-negotiation-pol
icy: "ELBSecurityPolicy-TLS-1-2-2017-01"

# Save and quit the editor, and then run the following command to
 check your changes.

kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

# Make sure that the annotation you added is present.
```

DWX-5926: Cloning an existing Hive Virtual Warehouse fails

Problem: If you have an existing Hive Virtual Warehouse that you clone by selecting Clone from the drop-down menu, the cloning process fails. This does not apply to creating a new Hive Virtual Warehouse.

Workaround: Make the following configuration change to resolve this issue:

1. In the Hive Virtual Warehouse tile, click the edit icon. This launches the Virtual Warehouse details page.
2. In the details page for the Virtual Warehouse, click the Configurations tab.
3. Click the Hiveserver2 sub-tab.
4. Select hive-site from the configuration file drop-down list menu.
5. Search for the configuration property hive.metastore.sasl.enabled.
6. Set the hive.metastore.sasl.enabled configuration property to true.

   Note: If the hive.metastore.sasl.enabled configuration property is already set to true, delete the setting and re-enter it.

7. Click Apply in the upper right corner of the page to save the configuration.

8. Click the Actions menu and select Clone to clone the Hive Virtual Warehouse:

DWX-2690: Older versions of Beeline return SSLPeerUnverifiedException when submitting a query

Problem: When submitting queries to Virtual Warehouses that use Hive, older Beeline clients return an SSLPeerUnverifiedException error:

```
javax.net.ssl.SSLPeerUnverifiedException: Host name ‘ec2-18-219-
32-183.us-east-2.compute.amazonaws.com’ does not
match the certificate subject provided by the peer (CN=*.env-c2
5dsw.dwx.cloudera.site) (state=08S01,code=0)
```

Workaround: Only use Beeline clients from CDP Runtime version 7.0.1.0 or later.

Known issues in Impala Virtual Warehouses on public clouds

Learn about the known issues related to Impala Virtual Warehouse in Cloudera Data Warehouse service on public clouds, the impact or changes to the functionality, and the workaround.

IMPALA-11039 Impala Virtual Warehouse instability

Problem: Executors can crash during the Parquet scans

Workaround: Disable Parquet late materialization in CDW version 2021.0.6-b96. To the default_query_options startup option, add the following property setting:

```
PARQUET_LATE_MATERIALIZATION_THRESHOLD=-1
```

Upgrade to the next CDW version when available.

CDPD-38204 Top-N query failure

Problem: A Top-N operator crashes in the Impala Virtual Warehouse

Workaround: Set the analytic rank pushdown threshold to zero.To the default_query_options startup option, add the following property:

```
ANALYTIC_RANK_PUSHDOWN_THRESHOLD=0
```

Upgrade to the next CDW version when available.

IMPALA-11045 Impala Virtual Warehouses might produce an error when querying transactional (ACID) table even after you enabled the automatic metadata refresh (version DWX 1.1.2-b2008)

Problem: Impala doesn't open a transaction for select queries, so you might get a FileNotFound error after compaction even though you refreshed the metadata automatically.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Impala Virtual Warehouses might produce an error when querying transactional (ACID) tables (DWX 1.1.2-b1949 or earlier)

Problem: If you are querying transactional (ACID) tables with an Impala Virtual Warehouse and compaction is run on the compacting Hive Virtual Warehouse, the query might fail. The compacting process deletes files and the Impala Virtual Warehouse might not be aware of the deletion. Then when the Impala Virtual Warehouse attempts to read the deleted file, an error can occur. This situation occurs randomly.

Workaround: Run the INVALIDATE METADATA statement on the transactional (ACID) table to refresh the metadata. This fixes the problem until the next compaction occurs. For information about running this statement, see INVALIDATE METADATA statement.

Do not use the start/stop icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier

Problem: If you use the stop/start icons in Impala Virtual Warehouses version 7.2.2.0-106 or earlier, it might render the Virtual Warehouse unusable and make it necessary for you to re-create it.

Workaround: Do not use the stop/start icons in these older Virtual Warehouses. Instead, these older versions automatically suspend and resume the Impala executors depending on the absence or presence of queries, making manual start or stop unnecessary.

DWX-6674: Hue connection fails on cloned Impala Virtual Warehouses after upgrading

Problem: If you clone an Impala Virtual Warehouse from a recently upgraded Impala Virtual Warehouse, and then try to connect to Hue, the connection fails.

Workaround: Create a new Impala Virtual Warehouse and do not clone from a recently upgraded warehouse. Then the connection to Hue from the new Impala Virtual Warehouse succeeds.

DWX-5841: Virtual Warehouse endpoints are now restricted to TLS 1.2

Problem: TLS 1.0 and 1.1 are no longer considered secure, so now Virtual Warehouse endpoints must be secured with TLS 1.2 or later, and then the environment that the Virtual Warehouse uses must be reactivated in CDW. This includes both Hive and Impala Virtual Warehouses. To reactivate the environment in the CDW UI:

1. Deactivate the environment. See Deactivating AWS environments or Deactivating Azure environments.
2. Activate the environment. See Activating AWS environments or Activating Azure environments

Workaround: If environment reactivation is not possible, you can perform manual steps using the kubectl command line tool to pick up the TLS 1.2 endpoint change. Open a terminal window on a system where the kubectl command line tool is installed, log in, and run the following commands:

```
kubectl edit svc nginx-service -n <CLUSTER-NAME>

# Add the following under the metadata.annotations field
service.beta.kubernetes.io/aws-load-balancer-ssl-negotiation-pol
icy: "ELBSecurityPolicy-TLS-1-2-2017-01"

# Save and quit the editor, and then run the following command to
 check your changes.

kubectl get svc nginx-service -n <CLUSTER-NAME> -o yaml

# Make sure that the annotation you added is present.
```

DWX-5276: Upgrading an older version of an Impala Virtual Warehouse can result in error state

Problem: If you upgrade an older version of an Impala Virtual Warehouse (DWX 1.1.1.1-4) to the latest version, the Virtual Warehouse can get into an Updating or Error state.

Workaround: none

DWX-3914: Collect Diagnostic Bundle option does not work on older environments

The Collect Diagnostic Bundle menu option in Impala Virtual Warehouses does not work for older environments:

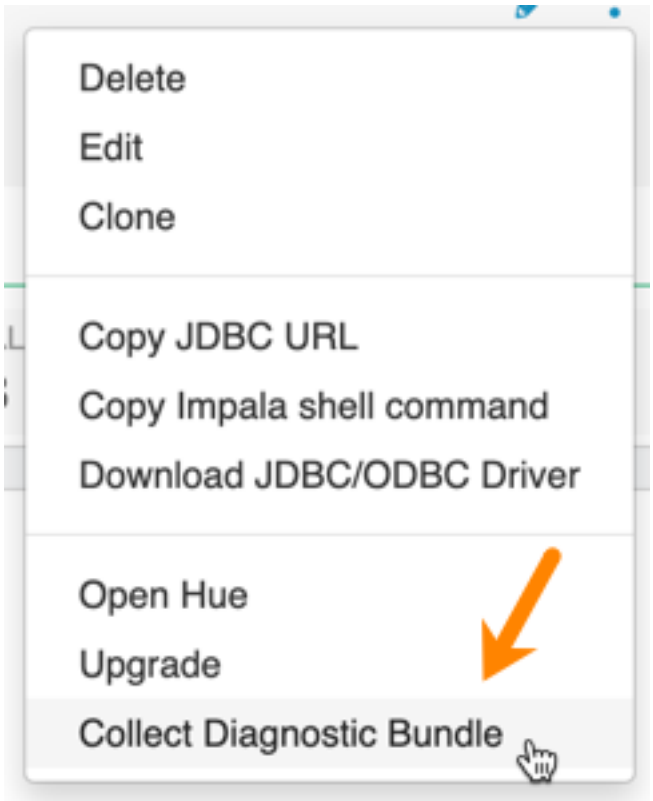

Data caching:

This feature is limited to 200 GB per executor, multiplied by the total number of executors.

Sessions with Impala continue to run for 15 minutes after the connection is disconnected.

When a connection to Impala is disconnected, the session continues to run for 15 minutes in case so the user or client can reconnect to the same session again by presenting the session_token. After 15 minutes, the client must re-authenticate to Impala to establish a new connection.

Behavior changes (unsupported releases)

Behavior changes in unsupported releases.

November 17, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following behavior changes:

Summary:

Removal of the engine.hive.enabled property

Before this release:

The engine.hive.enabled property was set to "true" to enable the creation of Hive-enabled Iceberg table.

After this release:

The engine.hive.enabled property is removed because Hive is already supported and SQL engines are not required to explicitly specify this property.

Summary:

Change in Backup and Restore

Before this release:

When backing up and restoring CDW, temporarily deploying at least one Virtual Warehouse that runs 2023.0.14.0-15 or later to your environment was not needed to back up Hue.

After this release:

Under certain conditions, you must deploy a Virtual Warehouse that runs 2023.0.14.0-15 or later for a successful Hue backup. For more information, see "Automatically backing up the environment" and "Backing up Hue".

Summary:

Change in supported Instance Metadata Service (IMDS) version

Before this release:

CDP supported AWS EC2 instance metadata from a running instance using only Instance Metadata Service V1 (IMDSv1).

After this release:

To enhance security in CDW clusters, CDP supports Instance Metadata Service V2 (IMDSv2) only in CDW clusters. CDP provisions instances for CDW clusters with IMDSv2. CDP no longer supports IMDSv1. Cloudera has removed the capability to access metadata using IMDSv1 in CDW clusters.

Summary:

Change in the way dates are parsed from string by ignoring trailing invalid characters

Before this release:

HIVE-20007 introduced changes in the way dates were parsed from strings. SQL functions or date operations involving invalid dates returned "null".

After this release:

HIVE-27586 extracts and returns a valid date from a string value if there is a valid date prefix in the string. This fix partially restores the behavior changes introduced as part of HIVE-20007 and also makes the current behavior of handling trailing invalid characters more consistent.

The following table illustrates the behavior changes before and after the fix:

Strong value	Behavior (before HIVE-20007)	Previous behavior (after HIVE-20007)	Current behavior (after HIVE-27586)
2023-08-03_16:02:00	2023-08-03	null	2023-08-03
2023-08-03-16:02:00	2023-08-03	null	2023-08-03
2023-08-0316:02:00	2024-06-11	null	2023-08-03
03-08-2023	0009-02-12	null	0003-08-20
2023-08-03 GARBAGE	2023-08-03	2023-08-03	2023-08-03
2023-08-03TGARBAGE	2023-08-03	2023-08-03	2023-08-03
2023-08-03_GARBAGE	2023-08-03	null	2023-08-03

This change affects various Hive SQL functions and operators that accept dates from string values, such as CAST (V AS DATE), CAST (V AS TIMESTAMP), TO_DATE, DATE_ADD, DATE_DIFF, WEEKOFYEAR, DAYOFWEEK, and TRUNC.

Summary:

Change in the way date and timestamp values are parsed.

Before this release:

Some of the Hive date and timestamp functions, such as unix_timestamp(), from_unixtime(), date_format(), and cast() were enhanced to use the DateTimeFormatter class for printing and parsing date and timestamp objects. Prior to this change, these functions used the SimpleDateFormat class.

After this release:

Starting from this release, a new configurable hive.datetime.formatter property is introduced through HIVE-25576 that enables you to choose between SimpleDateFormat and DateTimeFormatter for the unix_timestamp, from_unixtime, and date_format SQL functions.

Although the DateTimeFormatter class is an improvement over SimpleDateFormat, some users may want to retain the old behavior to ensure compatibility after migration, therefore, making it necessary for introducing this property.

The possible values for the hive.datetime.formatter property are 'DATETIME' and 'SIMPLE' representing DateTimeFormatter and SimpleDateFormat respectively. The default value is set to 'DATETIME'.

Summary:

Change in creation of Database Catalog

Before this release:

Cloudera Data Warehouse (CDW) supported default and non-default Database Catalogs for the latest release. When creating a Database Catalog you selected the type of data lake, depending on how you intend to use CDW.

After this release:

CDW supports default Database Catalogs only, and when you activate the Virtual Warehouse, the Database Catalog is created. A data lake is transparently associated with your Database Catalog based on your activation settings, such as managed storage access, and other factors. You no longer need to select a type of data lake.

August 30, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following behavior changes:

Before this release: There was just a single Web-UI corresponding to the Active Catalog instance.
After this release: Both active and passive Catalog's Web UI are now accessible to the end user.

Prior to this release HA was supported through K8s based leader election. This caused some behavior changes in the Impala Catalog Web UI. Both active and passive Catalog's Web UI are now accessible to the end user. In the past there was just a single Web-UI corresponding to the Active Catalog instance.

See the known issue and workaround related to this change.

February 7, 2023

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud has the following behavior changes:

Summary:

Change in Grafana metrics collection

Before this release:

Many metrics for Grafana monitoring were being stored for 45 days without limits on the disk space consumed. Resource overruns could occur and impact the availability of the entire environment.

After this release:

Storing metrics longer than 15 days or consuming more than 90GB of disk space is not supported. Metrics older than 15 days will be deleted. If the stored metrics consume more than 90GB of disk space, metrics will be deleted regardless of the number of days stored.

See also Grafana Limitations documentation.

Fixed issues (unsupported releases)

Fixed issues in unsupported releases.

January 18, 2024 Fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

Fixed the problem announced in TSB 2024-733: Changes in CoreDNS of AWS EKS affect operations in CDW, CDE and CML clusters on Public Cloud

This issue that caused a malfunction in CoreDNS that affects the Cloudera Data Warehouse (CDW), Cloudera Data Engineering (CDE) and Cloudera Machine Learning (CML) cluster operations has been fixed.

December 19, 2023 Fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

CDPD-48059: Unable to view and access S3 buckets from Hue configured with RAZ in CDW Public Cloud

Earlier, when you configured access to AWS S3 buckets using RAZ in the following regions, you couldn't access S3 buckets from the Hue File Browser: cn-*, eu-central, ap-northeast-2, ap-south-1, us-east-2, ca-central and eu-west. Due to a bug in AWS Boto SDK, accessing S3 buckets from Hue using RAZ was blocked. This issue has been fixed.

DWX-17111 IMPALA-12486 Missing database and table count metrics in Grafana

Catalog metrics for metadata loading have been added to Impala. Getting the fix requires a new environment activation using 1.8.2-b221 Dec 19, 2023 and Impala Virtual Warehouse with 2023.0.16.2-1 (Dec 19, 2023). If activating a new environment is not an option, then you can manually update Grafana charts with database and table counts to make the following change to the Prometheus query:

FROM:

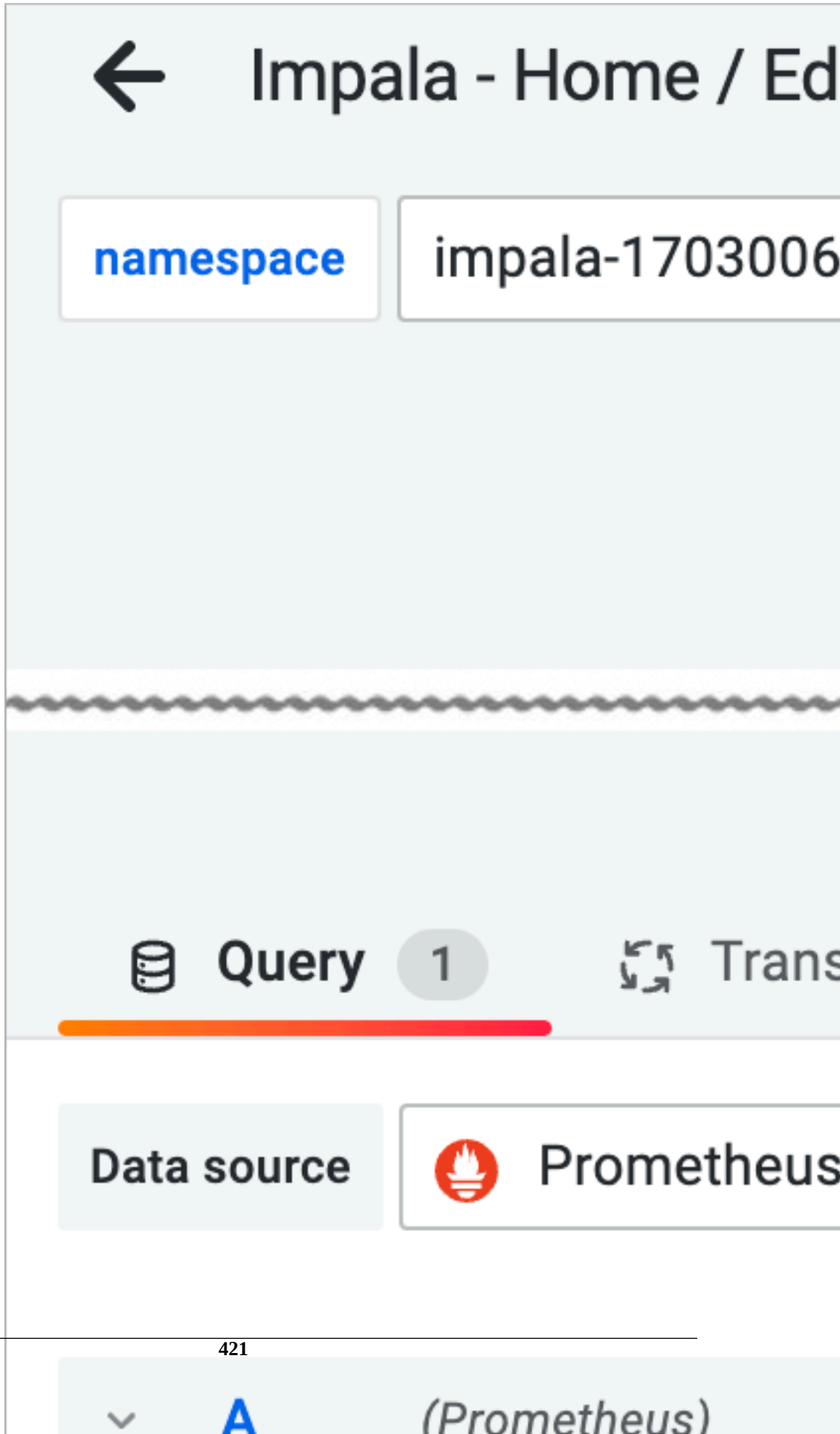
Impala - Home / Ed
namespace
impala-1703006
Query
1
Trans
Data source
Prometheus
A
(Prometheus)

TO: app="coordinator"

Impala Virtual Warehouse version 2023.0.16.2-1 (Dec 19, 2023) is required to make this manual update.

Technical Service Bulletins

TSB 2024-723: Hue RAZ is using logger role to Read and Upload/Delete (write) files

For the latest update on this issue see the corresponding Knowledge Article: TSB 2024-723: Hue Raz is using logger role to Read and Upload/Delete (write) files.

December 1, 2023 Fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

CDPD-62960: From the left assist panel, the pop-up link to Table Browser displays error 500

Earlier, when you clicked the i button next to one of the tables in the left assist panel and then clicked on Table Browser on the pop-up window, Hue displayed a 500 error and then the Table Browser auto reloaded. This issue has been fixed.

CDPD-63429: Manual configuration needed to support Bedrock using the Boto3 library

Bedrock is now supported with the public version of the Boto3 library. Anthropic Claude is the suggested model to use with Bedrock. Add the following lines in the hue-safety-valve field to get Hue SQL AI Assistant working with Anthropic Claude:

```
[aws]
[[bedrock_account]]
access_key_id = [***ACCESS-KEY***]
secret_access_key = [***SECRET-KEY***]
region = us-east-1
[desktop]
[[ai_interface]]
service='bedrock'
model='claude'
```

November 20, 2023 Fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

DWX-15859 Issue backing up and restoring Workload Aware Auto-Scaling (WAAS)

You can now use the CDW backup and restore process to restore the WAAS configurations. The issue related to restoring the WAAS configuration is fixed.

IMPALA-12233 MT_DOP issue

Impala query with JOIN and LIMIT no longer hangs when MT_DOP is greater than 0.

DWX-16275 Slow Data Scan on Azure for Impala Virtual Warehouses

The Impala Virtual Warehouse performance issue when using CDW Runtime 2023.0.15.0-243 - 2023.0.15.1-2 is fixed in this release 2023.0.16.2-1 (released December 19, 2023).

DWX-16825 Impala core-site configurations mutability issue

Hadoop core-site and the Hive site properties in the Impala Virtual Warehouse, previously immuatble, are now mutable. In the Impala Virtual Warehouse, click Options ⋮ Edit Configurations . Search or scroll to the property you want to edit and change the value.

VIZ-2269 CDW Virtual Warehouse connection to Data Visualization error

The "not authorized to delegate" error caused by impersonation configurations no longer occurs when creating or editing the Virtual Warehouse connection. For more information, see Data Visualization release notes.

CDPD-61589 Problem reading a file downloaded from Hue on ABFS

Downloading a file from the Hue File Browser, which is configured to use ABFS, no longer causes file corruption. The problem caused by Hue reading 1 byte less for every 1 MB of file content fetched has been fixed. Hue now reads the exact 1 MB chunk size. There is no byte loss.

CDPD-61550 HUE ABFS browser listing limitation

The Hue ABFS browser is no longer limited to listing 5000 files maximum. You can list an unlimited number of files.

October 5, 2023 Fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

DWX-16424 Impala Web UI incompatibility

The problem that causes two URLs for the Web UI to appear when catalogd ran in HA mode has been fixed. Only a single URL appears now.

August 30, 2023 Fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

DWX-15181 SAML Auth via ODBC/JDBC does not work with active-active coordinator

The JDBC authentication problem that occurred when the Impala coordinator was configured for high availability (HA) has been fixed.

DWX-15576 Impala autoscaler ConfigMap watcher can return 'unexpected type' error

Autoscaler code was treating every watch event in the same way, which caused errors of the 'unexpected type'. This issue has been fixed to handle different event types appropriately.

DWX-15110 Misaligned dialog in the UI

The Create Virtual Warehouse dialog is now aligned when no Virtual Warehouses are running.

DWX-14970 Cosmetic UI problem with Virtual Warehouse start label

The start labels of the Impala Virtual Warehouse and Hive Virtual Warehouse are now consistent with regard to upper- and lower-case.

DWX-14874 An indicator shows that no Virtual Warehouses running

An indicator is displayed when no Virtual Warehouses are running.

DWX-15171 Disabling SSO for Impala VW lands does not cause an error state

DWX-15749 The Restart dbc cli command has been implemented for Azure

DWX-14334 Impala coordinator, catalog, executors, and statestore CPU graphs in Grafana have been fixed

DWX-15362 The latest version of Impala shell 4.3.0a3 can be downloaded from the Impala shell UI link

DWX-15218 Duplicate application configurations have been removed from Impala Virtual Warehouses in Unified Analytics mode

DWX-15219 Filtering of hadoop-core-site configuration files now works

May 5, 2023 fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

TSB 2023-670: Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

For the latest update on this issue see the corresponding Knowledge article: TSB 2023-670: Hue frontend may become stuck in CrashLoopBackoff on CDW running on Azure

CDPD-48893: Hue logs get overwritten without a clear root cause

In previous implementations, multiple file handlers would write to a single log file, causing the Hue logs to be overwritten. Hue now uses a socket handler, which solves this problem.

DWX-14630 Issue creating a Virtual Warehouse using the CLI

A problem when creating an Impala Virtual Warehouse using the CLI in 1.6.2 has been fixed. The problem caused the Impala Coordinator auto shutdown to fail when HA was disabled.

DWX-8980 Data consistency problem occurs when sharing a Database Catalog

Events are now raised in the Hive metastore when a refresh/invalidate command is run. Catalog daemons process events synchronously across all Virtual Warehouses that share metadata. The catalog cache across all the Virtual Warehouses that share a Database Catalog is synchronized and consistent.

Feb 7, 2023 fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

Technical Service Bulletin Nov 2022

CDP CLI CDW command issue when activating an Azure DWX environment

When clients try to activate a Microsoft Azure (Azure) environment on Cloudera Data Platform (CDP) Cloudera Data Warehouse (CDW) for Public Cloud using the CDP Command Line Interface (CLI) dw command's create-cluster sub-command with the --use-private-load-balancer switch, the created load balancer will be public instead of private.

Example command:

```
cdp \
--profile ${PROFILE} \
dw create-cluster \
--environment-crn ${ENV} \
--use-private-load-balancer
```

For the latest update on this issue see the corresponding Knowledge article:

TSB 2022-642: CDP CLI CDW command issue when activating an Azure DWX environment

December 13, 2022 fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

IIMPALA-11721 Impala repeated retries query of updated Iceberg table

In Cloudera Data Warehouse (CDW) Runtime 2022.0.12.0-90 released 2022-12-13, the problem has been fixed that caused Iceberg table loading to fail in local catalog mode if the table gets updated frequently.

November 22, 2022 fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

TSB 2023-656: Cloudera Data Warehouse does not clean up old Helm releases

For the latest update on this issue see the corresponding Knowledge article: 2023-656: Cloudera Data Warehouse does not clean up old Helm releases

September 15, 2022 fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

DWX-13094 Diagnostic bundle job fails with missing service account

In Cloudera Data Warehouse (CDW) v1.4.2, the diagnostic bundle job no longer fails if it was used before with any previous versions of CDW. Jobs from the latest version run with the correct name of the service account installed after the upgrade.

August 4, 2022 fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

DWX-12787 DWX Access Control - Unable to Update the VW with UserGroup

The Virtual Warehouse is now updated with UserGroups.

DWX-12669 HEADROOM property editing disabled for single node Hive Virtual Warehouse and WAIT TIME takes precedence

This issue fixes potential problems caused by setting HEADROOM properties when using a single node Hive VW and by making HEADROOM and WAIT TIME settings mutually exclusive. WAIT TIME settings now take precedence over HEADROOM settings when both are configured.

CDPD-40730 PARQUET-1682 Forward compatibility for TIME/TIMESTAMP

PARQUET-1682 was backported to maintain forward compatibility for TIME/TIMESTAMP. This resolves the incompatibility between Hive's direct Parquet dependency and the transitive Parquet versions brought in by Iceberg.

April 27, 2022 fixed issues

This release of the Cloudera Data Warehouse (CDW) service on CDP Public Cloud fixes these issues.

CDPD-35041 Security Vulnerability with SSO for Hive Virtual Warehouses in CDW Public Cloud

This release fixes the SSO-related problem that occurred only in Hive Virtual Warehouses in Cloudera Data Warehouse version 2021.0.6-b96 released on March 8, 2022. This problem affected tenants who had the CDW_ALLOW_SSO_FOR_LLAP entitlement. In a misconfigured Hive Virtual Warehouse, unauthorized users who knew the JDBC connection string and had a valid userid, could execute arbitrary SQL queries on a Hive Virtual Warehouse. This security vulnerability no longer exists. Hive now properly handles the situation by using LDAP authentication as follows: If SSO authentication is enabled for a Hive Virtual Warehouse and a user bypasses authentication altogether by removing the auth=browser option from the JDBC connection string, Hive uses LDAP authentication.

CVE-2021-44228 remediation for Cloudera Data Warehouse

March 8, 2022 Update

Cloudera has released a new version of Cloudera Data Warehouse that upgrades the embedded Log4j version to 2.17.1.

December 22, 2021 Update

Cloudera has released a new version of Cloudera Data Warehouse which upgrades the embedded Log4j version to 2.16. This provides a permanent fix for CVE-2021-44228.

To use this version you must upgrade your Database Catalog(s) and Virtual Warehouse(s) to the latest version, which is 2021.0.4.1-3. Additionally, you can create new Virtual Warehouses using the latest version, and those will also have the fix for this vulnerability.

If you had previously applied the configurations shared below, then those are no longer required. Please reach out to Cloudera Support if you have additional questions.

Introduction

As mentioned in Cloudera Technical Service Bulletin 2021-545 (Critical vulnerability in log4j2 CVE-2021-44228), the Cloudera data service is impacted by the recent Apache Log4j2 vulnerability. As per that bulletin:

The Apache Security team has released a security advisory for CVE-2021-44228 which affects Apache Log4j2. A malicious user could exploit this vulnerability to run arbitrary code as the user or service account running the affected software. Software products using log4j versions 2.0 through 2.14.1 are affected and log4j 1.x is not affected.

Cloudera is making short-term workarounds available for affected software and is in the process of creating new releases containing fixes for this CVE.

Short Term Resolution - Cloudera Data Warehouse

If you are using the Cloudera Data Warehouse data service in Cloudera Public Cloud, please follow the steps below to implement an immediate manual workaround. This will change the configurations within your running Cloudera Data Warehouse environment so that this vulnerability is mostly, but not completely mitigated. There will still remain some potential attack vectors, as per the official Apache guidance.

Note that this will not upgrade the specific Log4j libraries to a version that does not have this vulnerability. Instead, it protects against most attack vectors via configuration changes. See the “Long Term Resolution - Cloudera Data Warehouse” section below for information on what Cloudera is doing for the permanent fix to the Cloudera Data Warehouse data service.

These manual workaround steps need to be applied to already-running kubernetes pods. Some of the steps involve running kubectl commands to edit the configurations of the pods. Once these kubectl commands are run, if you edit the corresponding Virtual Warehouse (VW) or Database Catalog (DBC) - via the Cloudera Data Warehouse UI or CLI - then the changes you made via kubectl will be overwritten. So either refrain from making further edits to the VW/DBC, or re-apply the kubectl commands.

If, after running the commands on existing pods, new pods are provisioned, then the commands will have to be run on those new pods as well. This would be required in scenarios such as

- Create new VW
- Existing VW that was stopped is now started (manually or via autostart)
- Already-running VW is autoscaled up
- Need to edit some other VW config (via the Cloudera Data Warehouse UI or CLI), in which case the VW will be restarted with new pods that have the original configurations as they existed prior to having run the kubectl commands
- Upgrade of an environment

We understand that it puts additional operational burden to manage this vulnerability while the release with the permanent fix is delivered. If you prefer to simplify the operation while still keeping your Cloudera Data Warehouse environment online, then we recommend you disable autosuspend and auto-scale until the permanent fix is released. If autosuspend is disabled, then you would need to manually suspend the VW whenever it is not needed to avoid extra cost; then when it starts back up the kubectl commands need to be run again on the new pods.

Please see Appendix A below for the manual configuration workaround steps.

Long Term Resolution - Cloudera Data Warehouse

See the “December 22, 2021 Update” section above for details on the permanent fix for CVE-2021-44228. Please note that Cloudera will be releasing a subsequent version containing Log4j 2.17 in January. Moving forward, Cloudera will continue to monitor for other CVEs that are found, and will release updates appropriately to address them.

Cloudera JDBC and ODBC Drivers

The ODBC drivers for Impala and Hive are not affected by this vulnerability.

The Apache Hive JDBC driver uses the SLF4J API, but does not bundle Log4j as a logging provider. If an end user or a client application makes Log4j available as a provider when using the driver, then the user/client must make sure that they are using a Log4j version that is not affected by this vulnerability.

The following JDBC drivers for Impala and Hive are affected. Of those published on the Cloudera Downloads page, only the following versions are impacted. Each of these use Log4j version 2.12.1 or 2.13.3.

Hortonworks JDBC Driver for Apache Hive

- v2.6.12

Hive JDBC Driver

- v2.6.12
- v2.6.13
- v2.6.14
- v2.6.15

Impala JDBC Driver

- v2.6.18
- v2.6.19
- v2.6.20
- v2.6.21
- v2.6.22
- v2.6.23
- v2.6.24

Note that earlier versions of these JDBC drivers use Log4j versions that are not affected by this vulnerability, and can be ignored.

Short Term Resolution - Drivers

Remove the JndiLookup class from the driver jar file. The following command shows an example of how to do this, using the Impala JDBC 2.6.24 version of the driver. This is one command, with all parts to be included on a single line.

```
zip -q -d ~/JDBC-Drivers/ClouderaImpalaJDBC-2.6.24.1029/ClouderaImpalaJDBC42
-2.6.24.1029/ImpalaJDBC42.jar
com/cloudera/impala/jdbc42/internal/apache/logging/log4j/core/lookup/Jndi
Lookup.class
```

More information on this workaround is provided by the Apache Software Foundation here: https://logging.apache.org/log4j/2.x/security.html

Long Term Resolution - Drivers

New versions of the drivers will be released shortly that are not affected by this vulnerability. As these become available Cloudera will notify customers and make the new drivers available on the Cloudera Downloads page.

If there are further questions related to configuration of these drivers, please open a support case with Cloudera.

Appendix A - Manual Configuration Workaround

In several of the steps below it is required to use kubectl. In order to do this you will need the kubeconfig for the kubernetes cluster. You can see how to do this on AWS here, and on Azure here. Also, the kubectl commands require an ID for the virtual warehouse that is being updated. This ID is of the form “compute-##########-####” or “impala-##########-####”, and it can be found underneath the VW name on the Cloudera Data Warehouse Overview page.

Please note that the configuration changes made via the web console (for Data Analytics Studio, Hive Metastore, and Hue) must be completed before the remaining changes, done via kubectl, are applied. Once the web console changes are made, be sure to wait for the DBC/VW to restart completely before continuing on with the kubectl commands.

Data Analytics Studio

1. Go to the DBC
2. Edit
3. Configuration # Das Event Processor # env # Add key

   LOG4J_FORMAT_MSG_NO_LOOKUPS=true
4. Apply

Hive Metastore

1. Go to the DBC
2. Edit
3. Configuration # Metastore # env # SERVICE_OPTS, append

   " -Dlog4j2.formatMsgNoLookups=true" (with a space in front of -D)

   Note: if the env doesn't show SERVICE_OPTS, select any other configuration file and then select "env" again.
4. Apply

Hue

1. Go to the Hive VW
2. Edit
3. Configuration # Hue # env #hue-query-processor-log4j2.yaml # add the following under Property

```
-name: formatMsgNoLookups
 value: true
```

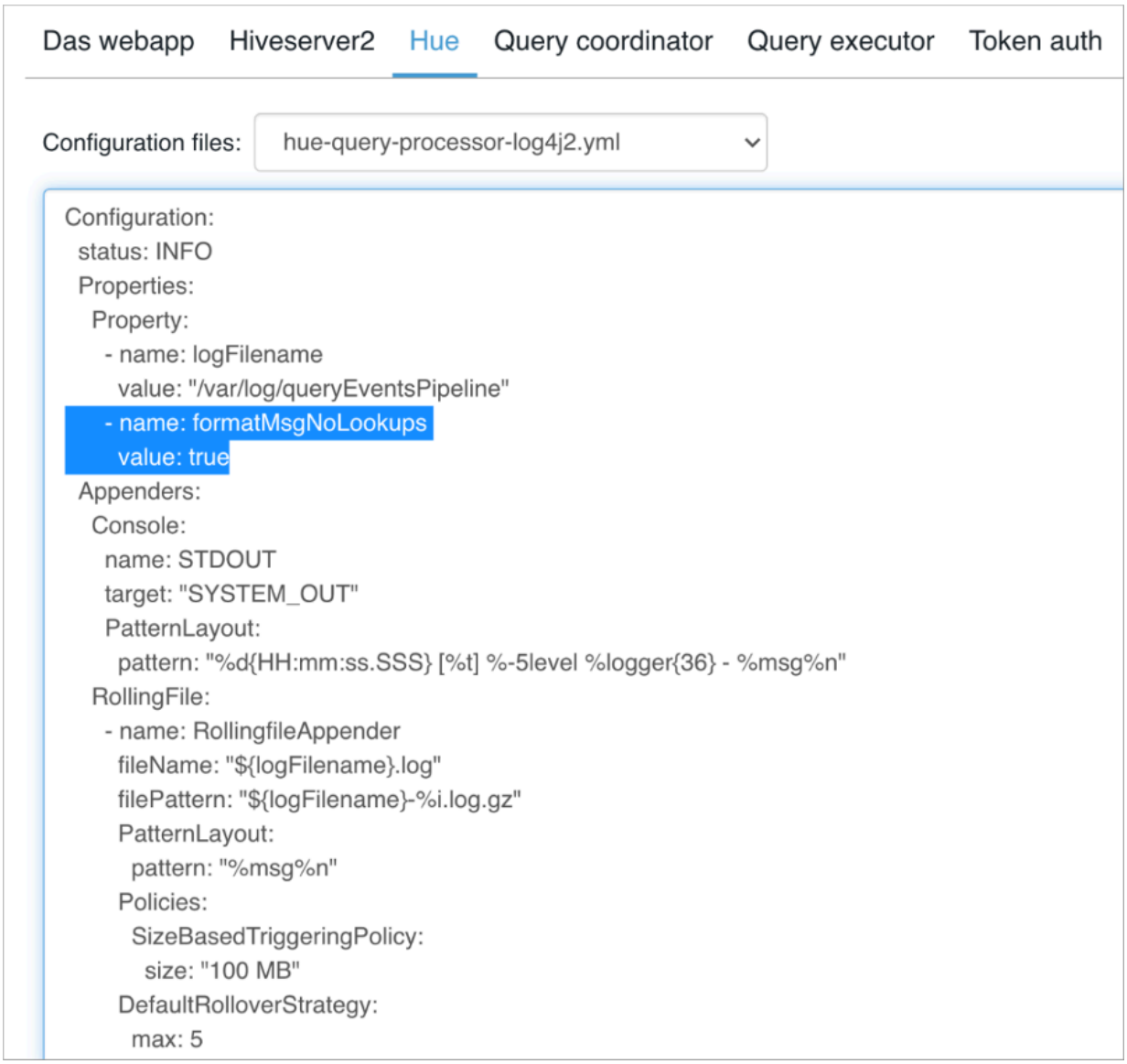

4. Apply

Hive Virtual Warehouse: Hive Server2

Note: These same steps are needed if you have an Impala Virtual Warehouse running in Unified Analytics mode (currently in Tech Preview). In that mode there is a Hive Server2 running within the Impala Virtual Warehouse, and it needs to be updated.

1. Fetch the statefulsets for hiveserver2 pod using “kubectl”

```
$ kubectl -n compute-1639516105-l76g get sts
NAME                           READY   AGE
das-webapp                     1/1     4h20m
hiveserver2                    1/1     4h20m
huebackend                     1/1     4h20m
hueep                          1/1     4h20m
hueqp                          1/1     4h20m
query-coordinator-0            2/2     4h20m
query-executor-0               2/2     4h20m
standalone-compute-operator    1/1     4h20m
```

2. Edit the statefulset and append -Dlog4j2.formatMsgNoLookups=true to the HADOOP_CLIENT_OPTS for container with EDWS_SERVICE_NAME=hiveserver2

```
kubectl -n compute-1639516105-l76g edit sts hiveserver2
```

In the editor, change the following section:

```
containers:
      - env:
        - name: EDWS_SERVICE_NAME
          value: hiveserver2
        - name: HADOOP_CLIENT_OPTS
          value: -DGroup_ID=compute-1639516105-l76g
-Dorg.wildfly.openssl.path=/usr/lib64
            -Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSL
v3,GCM
-Xms8192M   -Xmx11468M -Xss512k -Xloggc:/var/log/hive/hiveserver2-gc-%t.
log
-XX:+UseG1GC
            -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+PrintGCCause
-XX:+UseGCLogFileRotation
            -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=10M
-Dlog4j2.formatMsgNoLookups=true
```

3. This will restart the containers automatically. Wait until all the containers have restarted
4. To confirm that the change has taken effect, ssh to the container and execute something like “ps auxww”. This shows the hiveserver2 process and the command line should include the '-Dlog4j2.formatMsgNoLookups=true' property, which was added above.

```
hive         1  4.6  3.7 14128040 1228016 ?    Ssl  01:31   0:30
/etc/alternatives/jre/bin/java -Dproc_jar -Dproc_hiveserver2
-DGroup_ID=compute-1639516105-l76g -Dorg.wildfly.openssl.path=/usr/lib64
-Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM
-Xms8192M -Xmx11468M -Xss512k -Xloggc:/var/log/hive/hiveserver2-gc-%t.log
-XX:+UseG1GC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+PrintGCCause
-XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=10M
-Dlog4j2.formatMsgNoLookups=true
-javaagent:/jmx-exporter/jmx_prometheus_javaagent-0.11.0.jar=35000:/jmx-ex
porter/config.yml -Dlog4j.configurationFile=hive-edws-log4j2.properties
-Djava.util.logging.config.file=/usr/lib/hive/bin/../conf/parquet-loggi
ng.properties -Dyarn.log.dir=/usr/lib/hadoop/logs -Dyarn.log.file=hadoop
.log
-Dyarn.home.dir=/usr/lib/hadoop-yarn -Dyarn.root.logger=INFO,console
-Djava.library.path=/usr/lib/hadoop/lib/native
```

```
-Dhadoop.log.dir=/usr/lib/hadoop/logs -Dhadoop.log.file=hadoop.log -Dhadoo
p.home.dir=/usr/lib/hadoop -Dhadoop.id.str=hive
-Dhadoop.root.logger=INFO,console -Dhadoop.policy.file=hadoop-policy.xml
-Dhadoop.security.logger=INFO,NullAppender org.apache.hadoop.util.RunJar
/usr/lib/hive/lib/hive-service-3.1.3000.2021.0.5-b17.jar
org.apache.hive.service.server.HiveServer2
```

Hive Virtual Warehouse: Query Co-ordinator

1. Fetch the statefulsets for hiveserver2 pod using “kubectl”

```
$ kubectl -n compute-1639516105-l76g get sts
NAME                          READY   AGE
das-webapp                    1/1     4h20m
hiveserver2                   1/1     4h20m
huebackend                    1/1     4h20m
hueep                         1/1     4h20m
hueqp                         1/1     4h20m
query-coordinator-0           2/2     4h20m
query-executor-0              2/2     4h20m
standalone-compute-operator   1/1     4h20m
```

2. Edit the statefulset and append -Dlog4j2.formatMsgNoLookups=true to the JAVA_OPTS for container with EDWS_SERVICE_NAME=query-coordinator

```
kubectl -n compute-1639516105-l76g edit sts query-coordinator-0
containers:
      - env:
        - name: EDWS_SERVICE_NAME
          value: query-coordinator
        - name: JAVA_OPTS
          value: -Dorg.wildfly.openssl.path=/usr/lib64 -Dzookeeper.sasl
.client=false
            -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Xms2048M -Xmx2867M -se
rver
-Djava.net.preferIPv4Stack=true
            -XX:NewRatio=8 -XX:+UseNUMA -XX:+ResizeTLAB
-Xloggc:/var/log/hive/query-coordinator-gc-%t.log
            -XX:+UseG1GC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -ve
rbose:gc
-XX:+PrintGCCause
            -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10
-XX:GCLogFileSize=10M
            -Dlog4j2.formatMsgNoLookups=true
```

3. This will restart the containers automatically. Wait until all the containers have restarted
4. To confirm that the change has taken effect, ssh to the container and execute something like “ps auxww”. This shows the query coordinator process and the command line should include the '-Dlog4j2.formatMsgNoLookups=true' property, which was added above.

```
hive         1  5.9  2.8 8917460 918892 ?      Ssl  01:47   0:24
/etc/alternatives/jre/bin/java -Dproc_querycoordinator -classpath
/etc/hive/conf:/custom-jars/*:/custom-jars/lib/*:/usr/lib/hive/lib/*:/etc
/tez/conf:/usr/lib/tez/*:/usr/lib/tez/lib/*:/etc/hadoop/conf:/usr/lib/ha
doop/lib/*:/usr/lib/hadoop/.//*:/usr/lib/hadoop-hdfs/lib/*:/usr/lib/hado
op-hdfs/.//*:/usr/lib/hadoop-mapreduce/.//*:/usr/lib/hadoop-yarn/lib/*:/
usr/lib/hadoop-yarn/.//*:/efs/udf-jars/conf:/efs/udf-jars/lib/udfs/* -se
rver -Djava.net.preferIPv4Stack=true
-Dlog4j.configurationFile=tez-edws-log4j2.properties
-DisThreadContextMapInheritable=true
-DLog4jContextSelector=org.apache.logging.log4j.core.async.AsyncLoggerCon
textSelector -Dlog4j2.asyncLoggerRingBufferSize=1000000 -Dorg.wildfly.op
enssl.path=/usr/lib64
```

```
-Dzookeeper.sasl.client=false -Djdk.tls.disabledAlgorithms=SSLv3,GCM -Xms2
048M
-Xmx2867M -server -Djava.net.preferIPv4Stack=true -XX:NewRatio=8 -XX:+UseN
UMA
-XX:+ResizeTLAB -Xloggc:/var/log/hive/query-coordinator-gc-%t.log -XX:+U
seG1GC
-XX:+PrintGCDetails -XX:+PrintGCTimeStamps -verbose:gc -XX:+PrintGCCause
-XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=10M
-Dlog4j2.formatMsgNoLookups=true
-javaagent:/jmx-exporter/jmx_prometheus_javaagent-0.11.0.jar=35001:/jmx-ex
porter/config.yml org.apache.tez.dag.app.DAGAppMaster --session
```

Impala Virtual Warehouse

Note: If you have an Impala Virtual Warehouse running in Unified Analytics mode (currently in Tech Preview), then in addition to the steps listed below, please also follow the steps listed above in the “Hive Virtual Warehouse: Hive Server2” section. In Unified Analytics mode there is a Hive Server2 running within the Impala Virtual Warehouse, and it needs to be updated.

- Coordinator
  - Update coordinator statefulset and set -Dlog4j2.formatMsgNoLookups=true in JAVA_TOOL_OPTIONS environment variable

```
$kubectl --kubeconfig ~/kubeconfig edit sts coordinator -n impala-163953
0043-8fnb
...
        - /opt/impala/bin/impalad
        - --flagfile=/opt/impala/conf/flagfile
        command:
        - /opt/impala/bin/daemon_entrypoint.sh
        env:
        - name: JAVA_TOOL_OPTIONS
          value: -Xms2G -Xmx4G -XX:+HeapDumpOnOutOfMemoryError
-XX:HeapDumpPath=/opt/impala/dump.hprof
-XX:+UseConcMarkSweepGC -Dlog4j2.formatMsgNoLookups=true
        - name: POD_NAME
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: metadata.name
        image:
container-dev.repo.cloudera.com/cloudera/impalad_coordinator:2021.0.5-b1
7
        imagePullPolicy: IfNotPresent
        name: impalad-coordinator
...
```

- Executor
  - Update impala-executor-<xxx> statefulset and set -Dlog4j2.formatMsgNoLookups=true in JAVA_TOOL_OPTIONS environment variable

```
$kubectl --kubeconfig ~/kubeconfig edit sts impala-executor-000 -n
impala-1639530043-8fnb
...
  env:
  - name: JAVA_TOOL_OPTIONS
    value: -Xms2G -Xmx4G -XX:+HeapDumpOnOutOfMemoryError
           -XX:HeapDumpPath=/opt/impala/dump.hprof
           -XX:+UseConcMarkSweepGC -Dhttp.maxConnections=16
           -Dlog4j2.formatMsgNoLookups=true
        image:
```

```
container-dev.repo.cloudera.com/cloudera/impalad_executor:2021.0.5-b17
        imagePullPolicy: IfNotPresent
        lifecycle:
          preStop:
            exec:
              command:
              - /bin/bash
              - /opt/impala/bin/graceful_shutdown_backends.sh
              - "3600"
        name: impalad-executor
 ...
```

- If there are multiple impala-executor-<xxx> statefulsets then the above steps should be repeated for all of them.

- Catalogd
  - Update catalogd deployment and set -Dlog4j2.formatMsgNoLookups=true in JAVA_TOOL_OPTIONS environment variable

```
$kubectl --kubeconfig ~/kubeconfig edit deployment catalogd -n impala-16
39530043-8fnb
...
        env:
        - name: JAVA_TOOL_OPTIONS
          value: -Xms2G -Xmx6G -XX:+HeapDumpOnOutOfMemoryError
-XX:HeapDumpPath=/opt/impala/dump.hprof
            -XX:+UseConcMarkSweepGC
             -Dlog4j2.formatMsgNoLookups=true
        image:
container-dev.repo.cloudera.com/cloudera/catalogd:2021.0.5-b17
        imagePullPolicy: IfNotPresent
        livenessProbe:
          exec:
            command:
            - sh
            - -c
            - curl -I --silent http://localhost:25021/metrics | grep -q
 "HTTP/1.1
              200 OK"
          failureThreshold: 20
          initialDelaySeconds: 1
          periodSeconds: 20
          successThreshold: 1
          timeoutSeconds: 2
        name: catalogd
. . .
```

Cloudera Data Warehouse Public Cloud

Cloudera Data Warehouse Top Tasks

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Exploring a data lake **4**

Creating a Virtual Warehouse **7**

How to size a Virtual Warehouse 8
Auto-suspend timeout in Cloudera Data Warehouse 8
Configuring auto-scaling 9
Configuring Hive VW concurrency auto-scaling 9
Configuring Hive query isolation auto-scaling 10
Configuring Impala Virtual Warehouse auto-scaling 11
Configuring Impala coordinator shutdown 12
Configuring Impala coordinator high availability 14
Configuring Impala catalog high availability 17

Resizing the Hive Virtual Warehouse size **18**

Exploring a data lake

In Cloudera Data Warehouse, you explore sample airline database tables in your data lake from a Virtual Warehouse. You learn how to load the airline data. You view and query the tables.

Before you begin

- You obtained permissions to access a running environment for creating a Database Catalog and Virtual Warehouse.
- You obtained the DWAdmin role to perform Cloudera Data Warehouse tasks.
- You logged into the Cloudera web interface.
- You activated the environment from Cloudera Data Warehouse.
- You set up a Hadoop SQL policy in Ranger to access data in the data lake.

For more information about meeting prerequisites, see Getting started in Cloudera Data Warehouse.

About this task

In this task, you set up a minimal Virtual Warehouse for learning how to explore a data lake. You do not need to configure auto-scaling and optional features to explore a data lake. Plan to delete the Virtual Warehouse after your exploration.

Procedure

1. Navigate to Data Warehouse Database Catalogs New Database Catalog .
2. In New Database Catalog, in Name, specify a Database Catalog name.
3. In Environments, select the name of your environment activated from Cloudera Data Warehouse.
4. Accept default values for the image version and data lake type (SDX).
5. Turn on Load Demo Data to explore sample airline data from Hue, and click Create.
6. Click Virtual Warehouses New Virtual Warehouse .

7. Set up the Virtual Warehouse:
   - Specify a Name for the Virtual Warehouse.
   - In Type, click the SQL engine you prefer: Hive or Impala.
   - Select your Database Catalog and User Group if you have been assigned a user group.

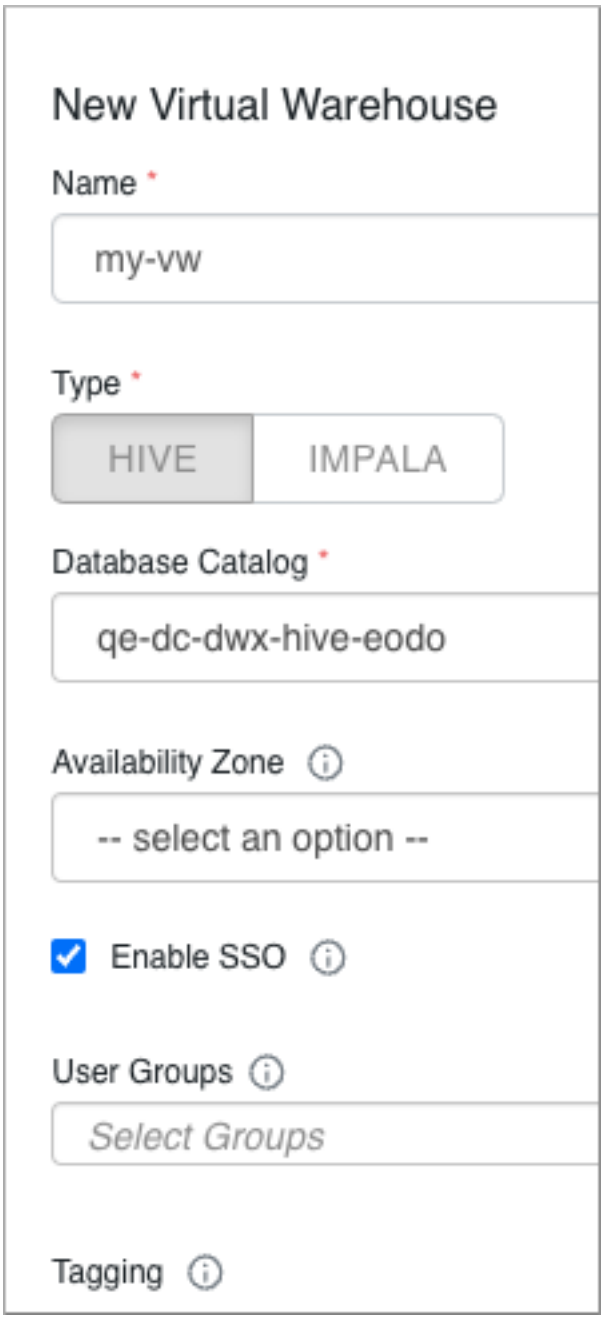

   - In Size, select the number of executors, for example xsmall-2Executors.
   - Accept default values for other settings.
8. Click Create.

9. After your Virtual Warehouse starts running, click Hue, and expand Tables to explore available data.

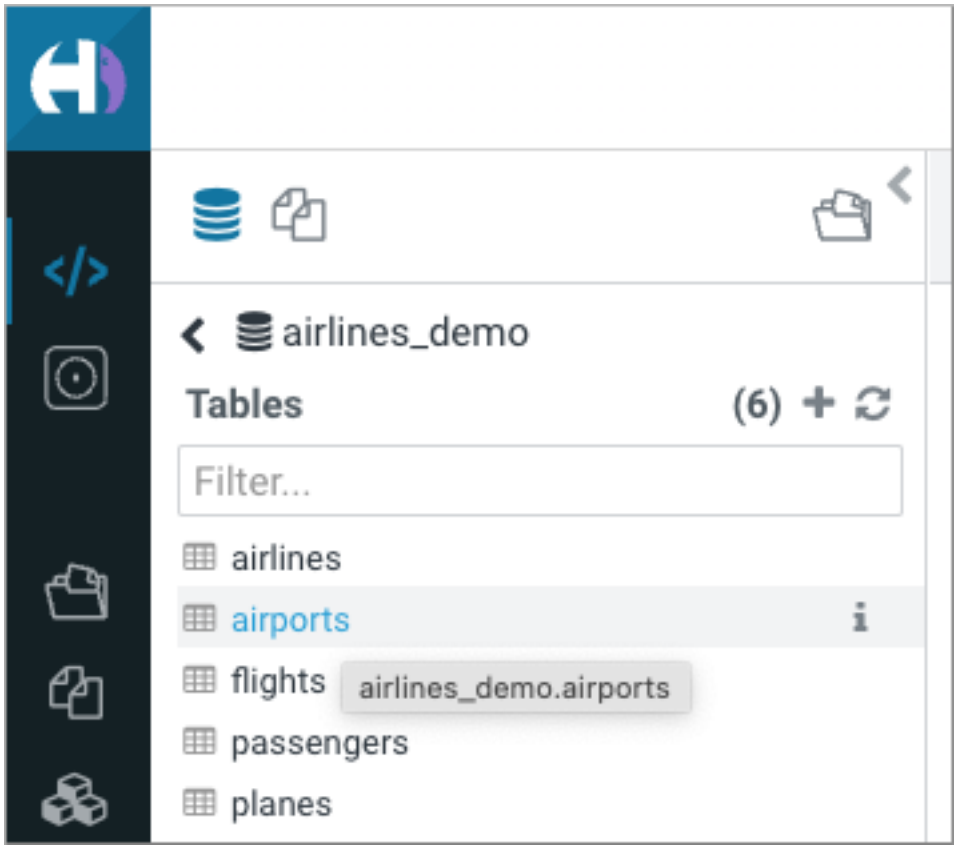

10. Explore data lake contents by running queries.
For example, select all data from the airlines table.

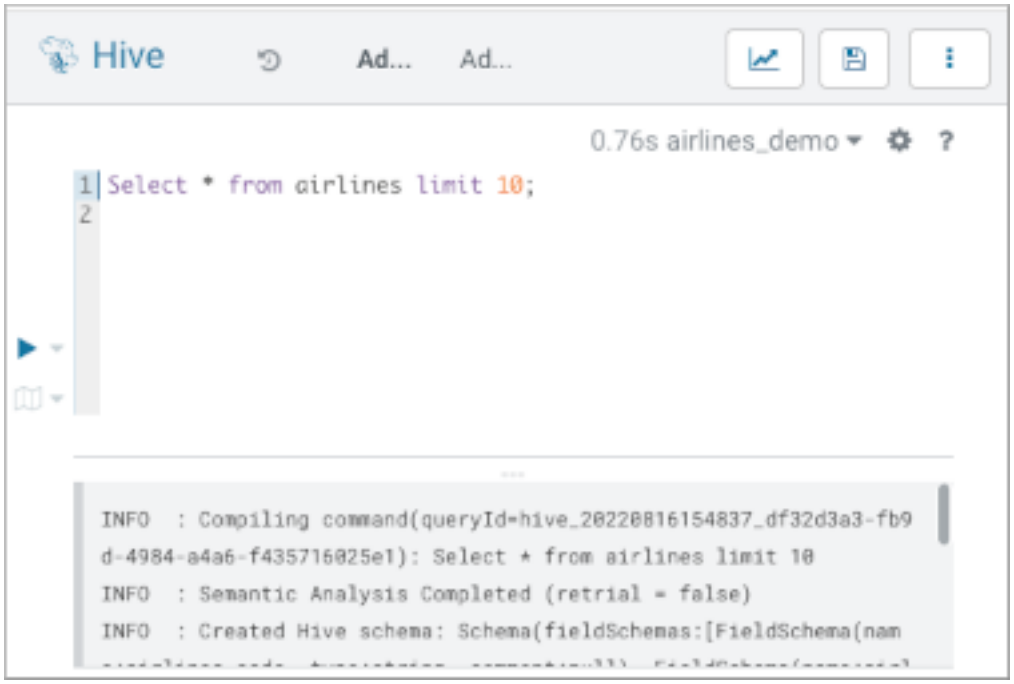

Creating a Virtual Warehouse

A Virtual Warehouse is an instance of compute resources in Public Cloud that is equivalent to an on-prem cluster. You learn how to create a new Virtual Warehouse in Cloudera Data Warehouse Public Cloud.

About this task

A Virtual Warehouse provides access to the data in tables and views in the data lake your Database Catalog uses. A Virtual Warehouse can access only the Database Catalog you select during creation of the Virtual Warehouse.

In this task and subtasks, you configure Virtual Warehouse features, including performance-related features for production workloads, such as the Virtual Warehouse size and auto-scaling. These features are designed to manage huge workloads in production, so if you are evaluating Cloudera Data Warehouse, or just learning, simply accept the default values. This task covers the bare minimum configurations.

In subtopics, you see details about how to configure features for production workloads, such as Hive query isolation and Impala catalog high availability.

Before you begin

- You obtained permissions to access a running environment for creating a Virtual Warehouse.
- You obtained the DWAdmin role to perform Cloudera Data Warehouse tasks.
- You logged into the Cloudera web interface.
- Your activated the environment from Cloudera Data Warehouse.

For more information about meeting these prerequisites, see Getting started in Cloudera Data Warehouse.

Procedure

1. Navigate to Data Warehouses Virtual Warehouses Create Virtual Warehouse .
2. In New Virtual Warehouse, specify a Name.

   Note: The fully qualified domain name of your Virtual Warehouse, which includes the Virtual Warehouse name plus the environment name must not exceed 64 characters; otherwise, Hue cannot load.

3. Select the Hive or Impala type of Virtual Warehouse you want.

   Virtual Warehouses can use Hive or Impala as the underlying SQL execution engine. Typically, Hive is used to support complex reports and enterprise dashboards. Impala is used to support interactive, ad-hoc analysis.
4. Select the Environment and Database Catalog for this Virtual Warehouse.
5. In AWS environments only, accept the default availability zone, or select an availability zone, such as us-east-1c.

   The default behavior is to randomly select an availability zone from the list of configured availability zones for the associated environment. Generally, it is fine to accept the default. All compute resources will run in the selected zone.

   Note: Selection of the zone is not an option in Azure environments.

6. Select the Compute Instance Types for the Virtual Warehouse based on your workload.

   For more information, see *Supported Compute Instance Types*.
7. Select the Hive Image Version or the Impala Image Version version, and the Hue Image version you want to use, or accept the default version (latest) at the top of the drop-down menus.
8. Select the Size of the Virtual Warehouse as described in the next subtopic.

   The “AutoSuspend Timeout” and “Concurrency Autoscaling” controls appear.
9. Configure auto-scaling as described in the subsequent subtopic.

10. In Authentication, select Enable SSO to enable single sign-on to your Virtual Warehouse, and in User Groups, select a user group set up in advance to access endpoints. If you do not have a user group set up for SSO, do not select Enable SSO.

    In Management Console User Management you set up a user group, required for enabling SSO, that identifies the users authorized to access to this Virtual Warehouse.
11. Enter keys and values for Tagging the Virtual Warehouse.
12. Accept default values for other settings, or change the values to suit your use case, and click Create Virtual Warehouse to create the new Virtual Warehouse.

    Click the tooltip for information about settings.

    When you create a Virtual Warehouse, a cluster is created in your cloud provider account. This cluster has two buckets. One bucket is used for managed data and the other is used for external data.

How to size a Virtual Warehouse

When you create a Virtual Warehouse, you need to carefully set the size of your Virtual Warehouse. The size of the Virtual Warehouse you select during Virtual Warehouse creation determines the number of executors and concurrent queries the Virtual Warehouse can run.

Before creating a Virtual Warehouse and setting the Virtual Warehouse size, learn about critical "Warehouse sizing requirements". If you misconfigure the size of your Virtual Warehouse, you can use one of the following methods, depending on the Impala or Hive type of Virtual Warehouse, to correct the size:

- Impala: Edit your Virtual Warehouse configuration to correct the size.
- Hive: Delete and recreate the Virtual Warehouse to correct the size.

When you create a Virtual Warehouse, you select one of the following Virtual Warehouse sizes:

Virtual Warehouse Size	Number of Executors
XSMALL	2
SMALL	10
MEDIUM	20
LARGE	40
Custom	Enter a value between '1' and '100'

If you are evaluating Cloudera Data Warehouse, or just learning, XSMALL is the recommended size. For production workloads, choose a size based on the following factors:

- The number of executors you typically use for clusters in an on-premises deployment.
- The complexity of your queries and the size of the data sets that they access.

Large warehouses with more executors can cache more data than small warehouses. Caching enhances performance.

Auto-suspend timeout in Cloudera Data Warehouse

Auto-suspend enables you to handle resources when the auto-scaler has scaled back to the last executor group. You can control the time that the original warehouse executor group idles after all other groups scale down and release their executors. The JDBC endpoint lives on to respond to queries from the result cache or statistics, but expensive executors no longer run.

When the auto-scaler increases or decreases the number of executor groups in the Virtual Warehouse, it can take several minutes before the scaling up or down takes effect. This slight delay is caused by the time required by your cloud provider to provision clusters.

When no queries are sent to an executor group, resources scale down and executors are released. When all executor groups are scaled back, when executors are idle, and after a period of idle time (Auto-suspend Timeout), the Virtual Warehouse is suspended.

You set an Auto-suspend Timeout to configure how long a Virtual Warehouse idles before shutting down.

Auto-suspend timeout is independent of the auto-scaling process and only applies to the original Virtual Warehouse and not to any additional warehouses that are created as a result of auto-scaling.

If a query is executed on a suspended Virtual Warehouse, then the query coordinator queues the query. When a queued query is detected, an executor group is immediately added (a scale-up occurs) to run the query on the Virtual Warehouse.

This feature can help lower cloud costs.

Auto-suspend differs from manually stopping a Virtual Warehouse. Sending a query to a stopped Virtual Warehouse returns an HTTP 503 "no healthy upstream" error, no scale-up occurs, and the query does not execute.

Auto-suspend is enabled by default.

While creating a Virtual Warehouse, set Auto-suspend Timeout to the number of seconds you want the Virtual Warehouse to idle before shutting down. For example, if you set the timeout to 300 seconds, the Virtual Warehouse suspends itself after 300 seconds to save compute resource expense. The suspended Virtual Warehouse restarts as soon as you run a query.

To disable auto-suspend, select the Disable Auto-suspend option. You can disable the auto-suspend option both while creating a Virtual Warehouse or after creating it.

Configuring auto-scaling

When you create a Virtual Warehouse, you set auto-scaling to increase and decrease resources according to demand. Auto-scaling relinquishes resources when demand decreases to limit unnecessary cloud expenses. Auto-scaling increases resources to speed performance.

Auto-scaling is designed for huge workloads in production, so if you are evaluating Cloudera Data Warehouse, or just learning, simply accept the default values. Later, you can edit the Virtual Warehouse to tune auto-scaling.

Before configuring or tuning auto-scaling, you need to understand the auto-scaling process to make good configuration decisions. As a Hive Virtual Warehouse user, choose the method of auto-scaling and see the relevant configuration subtopic (below):

- Concurrency for BI-type queries
- Isolation for ETL-type queries

To configure auto-scaling in a Hive or Impala Virtual Warehouse, see the topics below:

Configuring Hive VW concurrency auto-scaling

Configuring the Hive Virtual Warehouse to use concurrency auto-scaling is critical for controlling cloud expenses.

Before you begin

- You are familiar with the auto-scaling process.
- You are creating a Hive Virtual Warehouse for running BI-type queries.
- In Cloudera Data Warehouse, you added a Hive Virtual Warehouse, configured the size of the Hive Virtual Warehouse, and configured auto-suspend as described in previous topics.
- You obtained the DWAdmin role.

About this task

In this task, first you select the number of executors for your virtual cluster.

- Minimum executors

  The fewest executors you think you will need to provide sufficient resources for your queries. The number of executors needed for a cloud workload is analogous to the number of nodes needed for an on-premises workload.
- Maximum number

  The maximum number of executors you will need. This setting limits prevents running an infinite number of executors and runaway costs.

Consider the following factors when configuring the minimum and maximum number of executors:

- Number of concurrent queries
- Complexity of queries
- Amount of data scanned by the queries
- Number of queries

Next, you configure when your cluster should scale up and down based on one of the following factors:

- Headroom: The number of available coordinators that trigger auto-scaling. For example, if Desired Free Capacity is set to 1 on an XSMALL-sized Virtual Warehouse, which has 2 executors, when there is less than one free coordinator (2 queries are concurrently executing), the warehouse auto-scales up and an additional executor group is added.
- Wait Time: How long queries wait in the queue to execute. A query is queued if it arrives on HiveServer and no coordinator is available. For example, if WaitTime Seconds is set to 10, queries are waiting to execute in the queue for 10 seconds. The warehouse auto-scales up and adds an additional executor group.

Auto-scaling based on wait time is not as predictable as auto-scaling based on headroom. The scaling based on wait time might react to non-scalable factors to spin up clusters. For example, query wait times might increase due to inefficient queries, not query volume.

When headroom or wait time thresholds are exceeded, the Virtual Warehouse adds executor groups until the maximum setting for Minimum and Maximum has been reached.

Procedure

1. In Concurrency Autoscaling properties, limit auto-scaling by setting the minimum and maximum number of executor nodes that can be added.

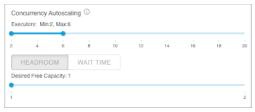

2. Choose when your cluster auto-scales up based on one of the following choices:
   - Headroom
   - Wait Time
3. Click Apply Changes.

Configuring Hive query isolation auto-scaling

You can configure your Hive Virtual Warehouse to add dedicated executors to run scan-heavy, data-intensive queries, also known as ETL queries. You learn how to enable a Virtual Warehouse auto-scaling feature and set a query isolation parameter in the Hive configuration.

Before you begin

- You are familiar with the auto-scaling process.
- You are creating a Hive Virtual Warehouse for running ETL-type queries.
- In Cloudera Data Warehouse, you added a Hive Virtual Warehouse, configured the size of the Hive Virtual Warehouse, and configured auto-suspend as described in previous topics.
- You obtained the DWAdmin role.

Note: You can enable query isolation only while creating a Virtual Warehouse. On existing warehouses, you can only disable the query isolation option.

About this task

In this task, first you configure the same auto-scaling properties as described in the previous topic Configuring Hive VW concurrency auto-scaling, and then you enable query isolation. Next, you set the hive.query.isolation.scan.size.threshold configuration parameter.

Procedure

1. In Concurrency Autoscaling properties, limit auto-scaling by setting the minimum and maximum number of executor nodes that can be added.

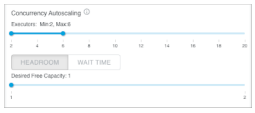

2. Choose when your cluster auto-scales up based on one of the following choices:
   - HEADROOM
   - WAIT TIME
3. In Concurrency Autoscaling, select Enable Query Isolation.

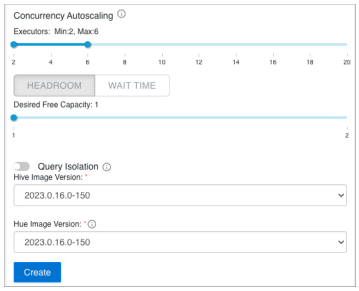

   Two configuration options appear: Max Concurrent Isolated Queries and Max Nodes Per Isolated Query.
4. In Max Concurrent Isolated Queries, set the maximum number of isolated queries that can run concurrently in their own dedicated executor nodes.

   Select this number based on the scan size of the data for your average scan-heavy, data-intensive query.
5. In Max Nodes Per Isolated Query, set how many executor nodes can be spawned for each isolated query.
6. Click Create Virtual Warehouse.
7. In the Cloudera Data Warehouse service **Overview** page, click Virtual Warehouses, and find your Virtual Warehouse. Click ⋮ Edit Configurations Hiveserver2 .
8. Select hive-site from the Configuration files drop-down menu and set hive.query.isolation.scan.size.threshold parameter to limit how much data can be scanned.

   For example, set 400GB.
9. Click Apply Changes.

Configuring Impala Virtual Warehouse auto-scaling

Configuring the Impala Virtual Warehouse to use concurrency auto-scaling is critical for controlling cloud expenses.

Before you begin

- You are familiar with the auto-scaling process.
- You are creating a Virtual Warehouse for running BI-type queries.
- In Cloudera Data Warehouse, you added an Impala Virtual Warehouse, configured the size of the Impala Virtual Warehouse, and configured auto-suspend as described in previous topics.

- You obtained the DWAdmin role.

About this task

In this task, you configure the following properties:

- Scale Up Delay: Sets the length of time in seconds to wait before adding more executors if queries wait in the queue.
- Scale Down Delay: Sets the length of time in seconds to wait before removing executors if executor groups are idle.

The time to auto-scale up or down is affected by the underlying Kubernetes configuration.

By default Impala Virtual Warehouses can run 3 large queries per executor group. Executors can handle more queries that are simpler and that do not utilize concurrency on the executor.

.

Procedure

1. In Scale Up Delay, set the seconds to wait before adding more executors if queries wait in the queue.

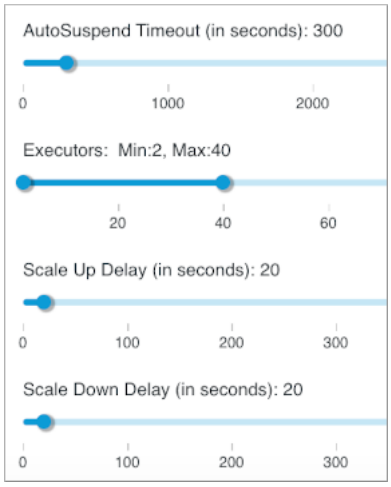

2. In Scale Down Delay, set the seconds to wait before removing executors when executor groups are idle.
3. Click Create Virtual Warehouse.

Configuring Impala coordinator shutdown

To cut cloud expenses, you need to know how to configure Impala coordinators to automatically shutdown during idle periods. You need to know how to prevent unnecessary restarts. Monitoring programs that periodically connect to Impala can cause unnecessary restarts.

About this task

When you create a Virtual Warehouse, you can configure Impala coordinators to automatically shutdown during idle periods. The coordinator start up can last several minutes, so clients connected to the Virtual Warehouse can time out.

Before you begin

- Update impyla, jdbc, impala shell clients if used to connect to Impala.

- When you create a Virtual Warehouse of SQL engine type Impala, and then you enable Unified Analytics, you can configure Impala coordinator shutdown only if you also configure Active-Passive mode.

Procedure

1. Follow instructions for "Creating a Virtual Warehouse".
2. Select a size for the Virtual Warehouse.
3. Do not select the Disable AutoSuspend option.

   The Impala coordinator does not automatically shutdown unless the Impala executors are suspended.

4. Select the Allow Shutdown of Coordinator option.

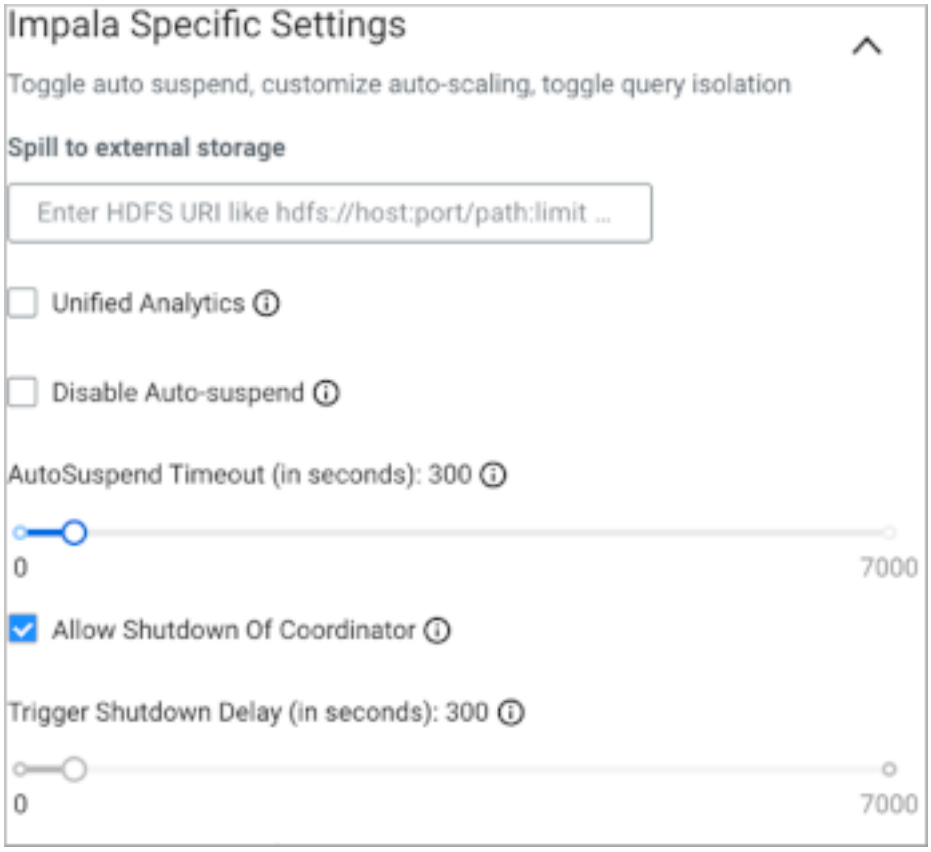

After Impala executors have been suspended, the Impala coordinator waits for the time period specified by the Trigger Shutdown Delay before shutting down.

For example, if AutoSuspend Timeout = 300 seconds and Trigger Shutdown Delay=150 seconds, after 300 seconds of inactivity Impala executors suspend, and then 150 seconds later, the Impala coordinator shuts down.

5. Accept default values for other settings, or change the values to suit your use case, and click Create Virtual Warehouse.

Click the tooltip for information about a setting.

Configuring Impala coordinator high availability

A single Impala coordinator might not handle the number of concurrent queries you want to run or provide the memory your queries require. You can configure multiple active coordinators to resolve or mitigate these problems. You can change the number of active coordinators later.

About this task

You can configure up to five active-active Impala coordinators to run in an Impala Virtual Warehouse. When you create an Impala Virtual Warehouse, Cloudera Data Warehouse provides you an option to configure Impala coordinator and Database Catalog high availability, described in the next topic. You can choose one of the following options:

Disabled

Disables Impala coordinator and Database Catalog high availability

Active-passive

Runs multiple coordinators (one active, one passive) and Database Catalogs (one active, one passive)

Active-active

Runs multiple coordinators (both active) and Database Catalogs (one active, one passive)

By using two coordinators in an active-passive mode, one coordinator is active at a time. If one coordinator goes down, the passive coordinator becomes active.

If you select the Impala coordinators to be in an active-active mode, the client software uses a cookie to keep a virtual connection to a particular coordinator. When a coordinator disappears for some reason, perhaps due to a coordinator shutting down, then the client software may print the error "Invalid session id" before it automatically reconnects to a new coordinator.

Using active-active coordinators, you can have up to five coordinators running concurrently in active-active mode with a cookie-based load-balancing.

An Impala Web UI is available for each coordinator which you can use for troubleshooting purposes.

Clients who connect to your Impala Virtual Warehouse using multiple coordinators must use the latest Impala shell. The following procedure covers these tasks.

Procedure

1. Follow instructions for "Creating Virtual Warehouse".
2. Select the number of executors you need from the Size dropdown menu.

   A number of additional options are displayed, including High availability (HA).

3. Select the Enabled (Active-Active) option from the High availability (HA) drop-down menu.
4. Select the number of coordinators you need from the Number of Active Coordinators drop-down menu ranging from 2 to 5.

   You can edit an existing Impala Virtual Warehouse to change the number of active coordinators.

   Important: Do not decrease the number of active-active coordinators you set up initially; otherwise, the Virtual Warehouse may shut down immediately. If clients are running queries on the Virtual Warehouse, the queries could fail.
5. Change values for other settings as needed, click Create Virtual Warehouse, and wait for the Impala Virtual Warehouse to be in the running state.

   Click ⓘ to learn more about the setting.

6. Go to Cloudera Data Warehouse Overview Impala Virtual Warehouse ⋮ Edit Web UI , and then click each Impala Coordinator Web UI *N* link to get information about the coordinator.

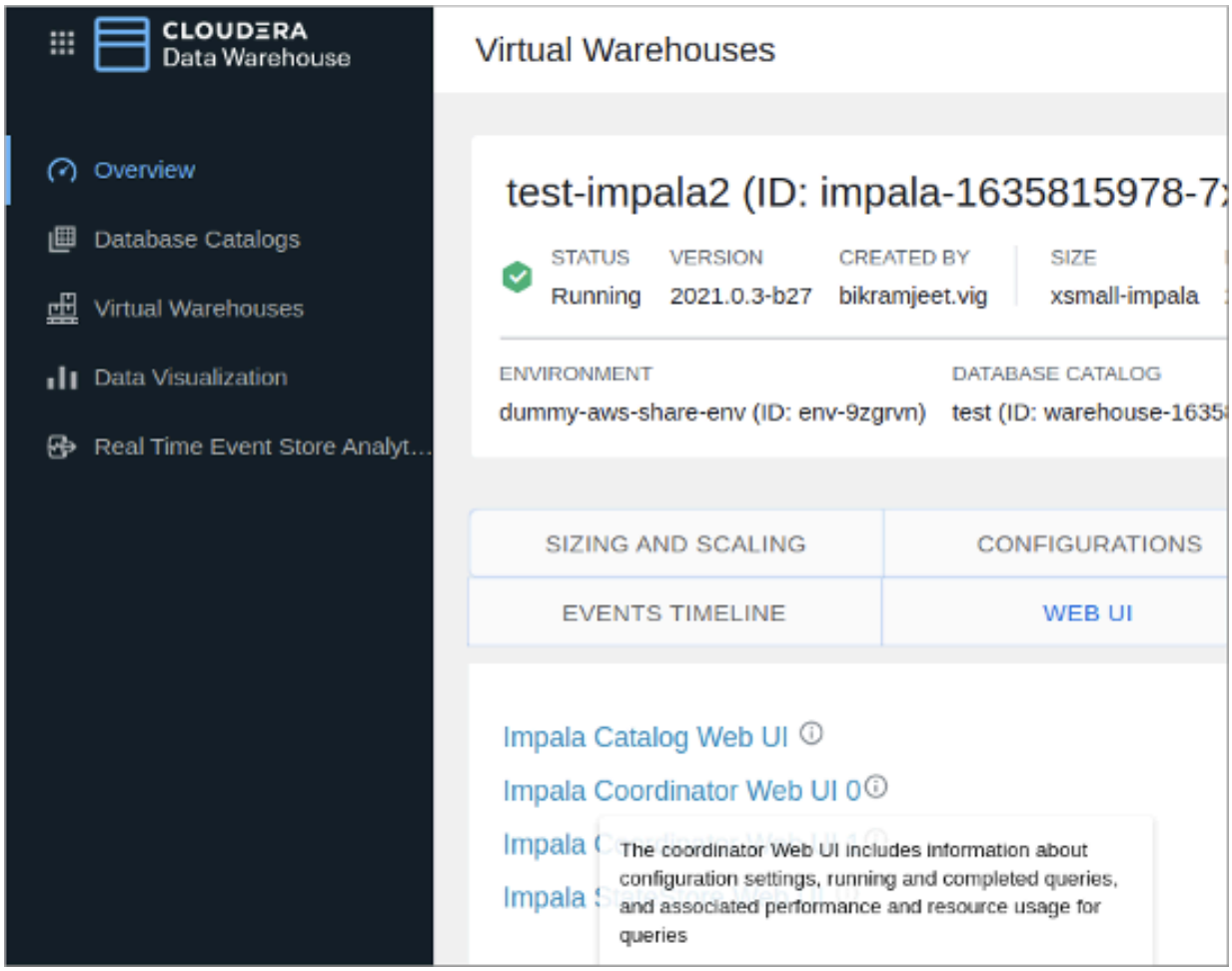

7. Go to Overview Impala Virtual Warehouse ⋮ and select the Copy Impala shell Download command option. The following command is copied to your clipboard:

```
pip install impala-shell==4.1.0
```

8. Provide the command to clients who want to connect to the Impala Virtual Warehouse with multiple coordinators using the Impala shell.
9. Instruct the client user to update impyla to version compatible with Cloudera Data Warehouse, as listed in Version information for Cloudera Data WarehousePublic Cloud components.

   For example, installing/updating impyla 0.18a2, is required to connect to your Virtual Warehouse active-active coordinators in Cloudera Data Warehouse 2021.0.3-b27.
10. Inform the client that to connect over ODBC to an HA-configured Impala Virtual Warehouse that uses active-active coordinators, you must append impala.session.id to the HTTPAuthCookies connector configuration option of the Cloudera ODBC driver.

Table 1: HTTPAuthCookies

Key Name	Value	Required
HTTPAuthCookies	impala.auth,JSESSIONID,KNOXSESSIONID,impala.session.id	Yes

Configuring Impala catalog high availability

By default, the Impala Virtual Warehouse runs a single instance of the catalog. The catalog stores databases, tables, resource usage information, configuration settings, and other objects managed by Impala. You can optionally configure running an additional Impala catalog instance. One catalog instance operates in active mode, the other in passive mode. The passive instance serves as a backup and takes over if the active instance goes down.

About this task

You enable catalog high availability when you create a New Virtual Warehouse. You cannot turn on, but you can turn off catalog high availability after creating an Impala Virtual Warehouse.

When you create an Impala Virtual Warehouse, you use the same UI dropdown to configure Impala coordinator, covered in the previous topic, and Database Catalog high availability. You can choose one of the following options:

- Disabled

  Disables Impala coordinator and Database Catalog high availability
- Active-passive

  Runs multiple coordinators (one active, one passive) and Database Catalogs (one active, one passive)
- Active-active coordinators

  Runs multiple coordinators (both active) and Database Catalogs (one active, one passive)

Before you begin

You must obtain the DWAdmin role.

Procedure

1. Follow instructions for Adding a new Virtual Warehouse.
2. In Size, select the number of executors, for example xsmall-2Executors.

   A number of additional options appear, including High availability.

3. In High availability (HA), accept the default Enabled (Active-Passive) or Enabled (Active-Active).

   Either option enables Database Catalog high availability in active-passive mode.

4. Accept default values for other settings, or change the values to suit your use case.

   Click the tooltip for information about settings.
5. Click Create Virtual Warehouse.

Resizing the Hive Virtual Warehouse size

The size of the Hive Virtual Warehouse you select during Virtual Warehouse creation determines the number of executors and concurrent queries the Virtual Warehouse can run. You need to know how to change the size of the Virtual Warehouse upward or downward to tune performance and manage cost.

About this task

You cannot change the size of a Hive Virtual Warehouse, but you can handle incorrect sizing in the following ways.

- You can delete the Virtual Warehouse, and then recreate it in a different size.
- You can change the auto-scaling thresholds to change the effective size of the Virtual Warehouse based on demand. The actual size does not change, but increases or decreases in resources occurs automatically.

This task assumes you have two Virtual Warehouses that you decide are incorrectly sized for some reason. You correct the sizing of one by deleting and recreating the Virtual Warehouse. You correct the effective sizing of the other by changing auto-scaling thresholds.

Before you begin

- You obtained the DWAdmin role.

Procedure

First Virtual Warehouse: Replace this Virtual Warehouse

1. Log in to the Cloudera web interface, navigate to Data Warehouse Overview , note the name of the Virtual Warehouse you want to modify, and note which Database Catalog it is configured to access.
2. Click the options of the Virtual Warehouse you want to delete, and select Delete.
3. Go to the Virtual Warehouses tab and click New Virtual Warehouse.
4. Set up the new Virtual Warehouse:
   - Type the same Name for the new Virtual Warehouse as you used for the old Virtual Warehouse.

     Note: The fully qualified domain name of your Virtual Warehouse, which includes the Virtual Warehouse name plus the environment name must not exceed 64 characters; otherwise, Hue cannot load.
   - In Type, click the SQL engine you prefer: Hive or Impala.
   - Select your Database Catalog and User Group if you have been assigned a user group.
   - In Size, select the number of executors, for example xsmall-2Executors.
   - Accept default values for other settings, or change the values to suit your use case.
5. Click Create Virtual Warehouse.

Second Virtual Warehouse: Change the Auto-Scaling Thresholds

6. In Data Warehouse Overview , click the options of the other Virtual Warehouse, a Hive Virtual Warehouse for example, to change auto-scaling thresholds, and select Edit.
7. Go to the Sizing And Scaling tab and in Concurrency Autoscaling, slide the control to change the Max number of executors.
8. Click Apply Changes.

Cloudera Data Warehouse Public Cloud

Troubleshooting

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Troubleshooting Cloudera Data Warehouse 4
Diagnostic bundles for Cloudera Data Warehouse and Kubernetes 4
Downloading Cloudera Data Warehouse and Kubernetes diagnostic bundles 5
AWS environment fails to activate 6
Opening Hue from Cloudera Data Warehouse causes an error 8
Troubleshooting Impala 8
Downloading Impala diagnostic bundles 8
Accessing Impala Workload Logs 9
Locations of Impala Log Files in S3 10
Locations of Impala Log Files in Azure 12
Analyzing Impala Virtual Warehouse Issues with the Events Timeline 14
Troubleshooting Impala 15
Debugging Impala Virtual Warehouses 15

Troubleshooting Cloudera Data Warehouse

Learn about common issues in Cloudera Data Warehouse, their cause, and the suggested steps to resolve them.

Diagnostic bundles for Cloudera Data Warehouse and Kubernetes

A diagnostic bundle captures information for troubleshooting and determining the root cause of problems in Cloudera Data Warehouse. Diagnostic bundles are available for troubleshooting your Virtual Warehouse, Database Catalog, Data Visualization, and environment/cluster.

You can obtain a diagnostic bundle collection for the following levels: Env (environment) DBC (Database Catalog), and VWH (Virtual Warehouse):

- Environment/cluster including Kubernetes resources
- DBC: Database Catalog including Kubernetes resources
- VWH:
  - Hive Virtual Warehouse including Kubernetes resources
  - Impala Virtual Warehouse including Kubernetes resources

The following information, and more, is included about Kubernetes resources:

- Pod
- Deployments
- CustomResource
- PVCs
- Statefulsets

At the Database Catalog and Environment level similar resource information and logs about the components are included in the bundle.

If you use Amazon CloudWatch and enable CloudWatch logs when you activate an AWS enviroment, or edit environment details, the following logs are included under their respective log streams and bundled at the ENV level:

- kube-scheduler
- kube-controller-manager
- kube-apiserver
- authenticator

For more information about enabling CloudWatch logs, see activate an environment from Cloudera Data Warehouse or edit environment details. You must add required permissions to your IAM policy.

The diagnostic bundle includes a history of events at the cluster level. Although Kubernetes only preserves information about events of the past hour, the diagnostic bundle backs up events for the last 30 min or 1, 12, 24 hours, or a custom time interval you can specify.

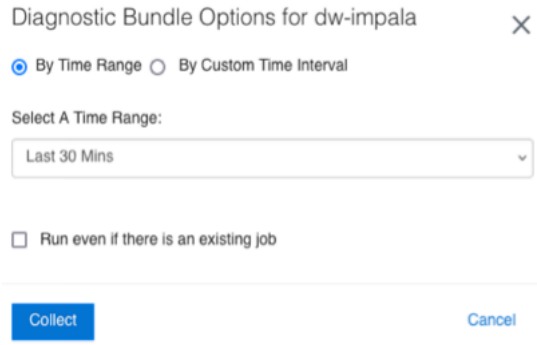

You can also specify a custom time interval for collecting information.

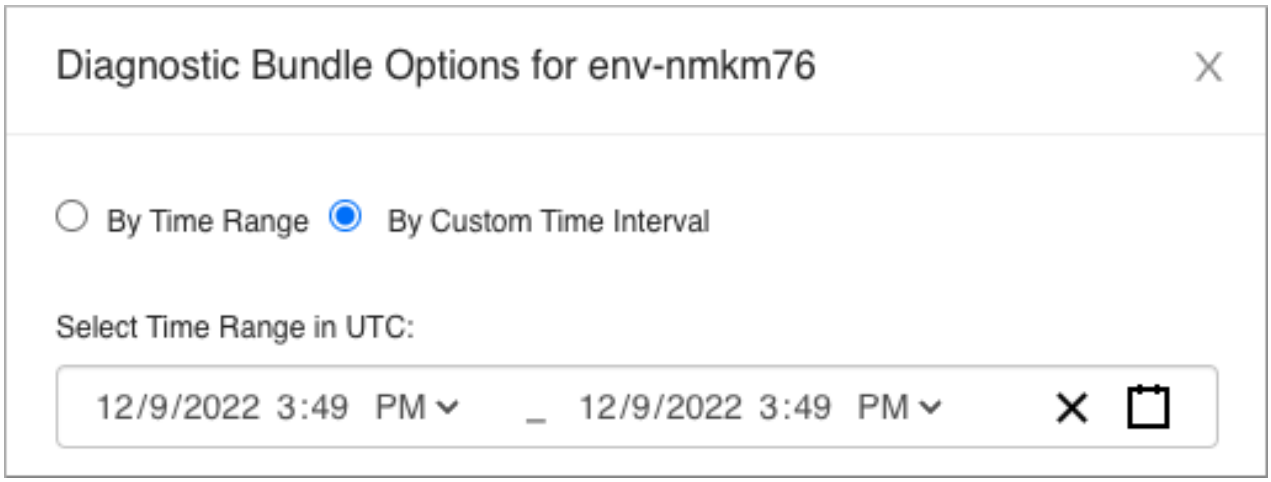

Downloading Cloudera Data Warehouse and Kubernetes diagnostic bundles

To troubleshoot issues with Cloudera Data Warehouse, download diagnostic bundles ZIP files stored in your public cloud account. This capability is available on AWS environments only.

Before you begin

- Before you can download log files, you must, of course, run workloads on your Hive or Impala Virtual Warehouse to generate the logs.
- Obtain the DWAdmin role.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. Collect a diagnostic bundle for your environment, Database Catalog, or Virtual Warehouse as follows:
   - Environment bundle: In Environments, find your environment, click ⋮, and then click Collect Diagnostic Bundle.
   - Database Catalog bundle: Click Overview Database Catalog Diagnostic Bundle , and click Collect Diagnostic Bundle.
   - Virtual Warehouse bundle: Click Overview in the left navigation panel, select your Virtual Warehouse, click ⋮, and click Collect Diagnostic Bundle.
3. Select information about events within selectable time ranges, or select By Custom Time Interval, and specify the interval you want.
4. The Collect For field is available only for AWS environments with Amazon CloudWatch enabled. It allows you to choose the type of logs to include in the diagnostic bundle.
5. Click Collect.

   After some time, depending on your cluster size and log sizes, but typically after 10 seconds, a message indicating completion appears indicating the diagnostic bundle is generated:

   ```
   Collection of Diagnostic Bundle … initiated.
   ```
6. Go to the **Overview** page, locate the environment, Database Catalog, or Virtual Warehouse from which you want to collect diagnostic data, and click ⋮ Edit Diagnostics Diagnostic Bundle .
7. In Diagnostic Bundles, copy the location of the diagnostic bundle ZIP file you want to download.

8. In the Amazon S3 management console, navigate to the location of the diagnostic bundle ZIP file, download the ZIP file, decompress it, and look at the troubleshooting information in the files.

Related Information

Send a diagnostic bundle to Cloudera Support
CDP CLI commands for generating a diagnostic bundle
Diagnostic bundle content

AWS environment fails to activate

Learn how to resolve AWS environment activation failure in Cloudera Data Warehouse Public Cloud

When activating an AWS environment in Cloudera Data Warehouse Public Cloud, the following error message might be returned if your cloud resources reside in the us-east-1 region:

```
TemplateURL must reference a valid S3 object to which you have access.
```

Cause of the issue

For this region, the endpoint URL is incorrect and cannot load the CloudFormation template to create the AWS CloudFormation stack in your AWS account. Cloudera Data Warehouse uses these stack resources for Database Catalogs and Virtual Warehouses in Cloudera Data Warehouse.

Steps to resolve the issue

Use the reduced permissions mode feature for AWS environments to manually create the CloudFormation stack for Cloudera Data Warehouse. This feature enables you to manually crate the CloudFormation stack resources from a template with a reduced set of IAM permissions. When you no longer need the environment, you must manually delete the CloudFormation stack resources in your AWS account.

1. Remove one of the permissions in your IAM permissions policy for the AWS account that you used to register the environment you want to activate for Cloudera Data Warehouse. For example, remove the s3:CreateBucket permission from your IAM permissions policy:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid":"VisualEditor0",
            "Effect":"Allow",
            "Action":[
                "ec2:AuthorizeSecurityGroupIngress",
                "acm:DeleteCertificate",
                "iam:RemoveRoleFromInstanceProfile",
                "s3:CreateBucket",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "dynamodb:DeleteTable",
                "ec2:DescribePlacementGroups",
                "rds:CreateDBSubnetGroup",
                "iam:AddRoleToInstanceProfile",
                "iam:DetachRolePolicy",
                "ec2:CreatePlacementGroup",
                "acm:RequestCertificate",
                "ec2:RevokeSecurityGroupEgress",
```

Remove

Removing one of the required permissions from your IAM policy causes Cloudera Data Warehouse to display the reduced permissions mode option in the system dialog box when you activate your environment in Cloudera Data Warehouse.

2. Follow the steps in Activating AWS environments in reduced permissions mode.

   Important:

   In Step 4 of the Activating AWS environments in reduced permissions mode procedure, edit the pre-populated CloudFormation template URL as follows.

   Change the URL from:

   ```
   https://<BUCKETNAME>.s3-region.amazonaws.com/cf-templates/<CDW-CLUSTERID>-cf-template.yml
   ```

   To:

   ```
   https://<BUCKETNAME>.s3.amazonaws.com/cf-templates/<CDW-CLUSTERID>-cf-template.yml
   ```

3. After you activate the environment and create the AWS CloudFormation stack resources in your AWS account, make sure that you apply the required tags to the stack that are listed in Required tags for CloudFormation stacks created with reduced permissions mode.
4. Add the s3:CreateBucket IAM permission back to your IAM permissions policy to make sure you have adequate permissions so Cloudera Data Warehouse can create CloudFormation stack resources for you when you activate environments later.

Deactivating environments created with the reduced permissions mode

When you no longer need the environment, you must manually delete the CloudFormation stack resources in the AWS Console by following the steps in Deactivating AWS environments created with reduced permissions mode.

Opening Hue from Cloudera Data Warehouse causes an error

You need to be aware of some naming limitations when you create an environment. Observe the character limits for the Virtual Warehouse domain name.

Problem: After creating a Virtual Warehouse, you get an HTTP 500 error when you open Hue.

Probable Cause: The fully qualified domain name of your Virtual Warehouse, which includes the Virtual Warehouse name plus the environment name, has likely exceeded the limit: 64 characters.

Solution: Recreate the Virtual Warehouse using a name having a length that when added to the length of the environment name conforms to the limit.

Troubleshooting Impala

Learn about common issues in Cloudera Data Warehouse Impala, their cause, and the suggested steps to resolve them.

Downloading Impala diagnostic bundles

Learn how to download diagnostic bundles to use for troubleshooting an Impala Virtual Warehouse in Cloudera Data Warehouse Public Cloud

About this task

To troubleshoot issues with your Impala Virtual Warehouses, download diagnostic bundles of log files for the sidecar containers that support Impala components and for the components themselves. These diagnostic bundles are ZIP files stored in your public cloud account.

Required role: DWAdmin

Before you begin

Before you can download log files, you must, of course, run workloads on your Impala Virtual Warehouse to generate the logs.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
2. Go to the Virtual Warehouses, locate the Virtual Warehouse from which you want to collect diagnostic data, and click ⋮ Collect Diagnostic Bundle .

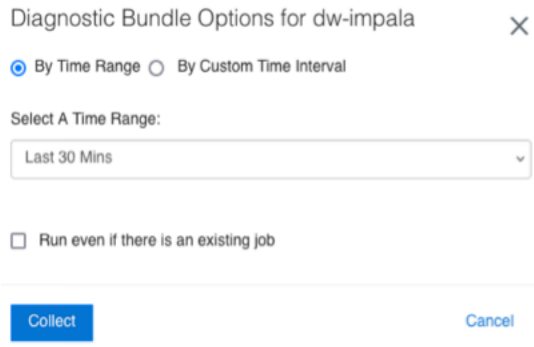

3. Set options that select which logs to generate for the diagnostic bundle.
   - By Time Range: Select a specific time range of log files to generate from the drop-down list, or you can choose a custom interval in the next option.
   - By Custom Time Interval: Select the start and end time from the drop-down list to define the specific time interval for the log files in the diagnostic bundle.
4. Click Collect to generate the bundle.

   After some time, depending on your cluster size and log sizes, but typically after 10 seconds, a message indicating completion appears indicating the diagnostic bundle is generated:

```
Collection of Diagnostic Bundle for impala-mpvt initiated. Please go to
details page for more information.
```

5. Go to the Virtual Warehouses tab, locate the Virtual Warehouse from which you want to collect diagnostic data, and click ⋮ Edit Diagnostics .
6. Click copy-to-clipboard to copy the path to the diagnostic bundle on S3/ABFS.
7. Paste the path to the diagnostic bundle into a text document, and navigate to the diagnostic bundle in AWS or Azure to download the ZIP file.

   When you expand the diagnostic bundle ZIP file that you downloaded, directories appear for log files and a diagnostic-data-generator.log file, which contains troubleshooting information.

Related Information

Send a diagnostic bundle to Cloudera Support

CDP CLI commands for generating a diagnostic bundle

Diagnostic bundle content

Accessing Impala Workload Logs

Describes how to locate Impala logs in S3 or Azure to diagnose some of the commonly encountered issues in Impala.

Using Impala Logs

The Impala logs record information about:

- Any errors Impala encountered.
- How Impala is configured.
- Jobs Impala has completed.

However, you can use the logs record information to troubleshoot only if the relevant logs are downloaded and then uploaded to a location where you can access them. To download the logs from S3 or Azure you must first identify the locations.

Locations of Impala Log Files in S3

This topic describes how to identify the Amazon S3 locations of Impala logs for the different Impala components.

About this task

The Cloudera Data Warehouse service collects logs from Impala Virtual Warehouses and uploads them to an Amazon S3 location. This S3 log location is configured under an external warehouse directory so that the logs are preserved even if the Virtual Warehouse they are collected from is destroyed.

Before you begin

To identify the location of the logs in S3, you must have the environment name, impala ID identifiers, and S3 bucket name.

Procedure

Finding the environment name, and impala ID identifiers

1. In the Data Warehouse service, expand the Environment column by clicking ⋮ .
2. From the Overview page, note down the environment name, and impala ID identifiers.

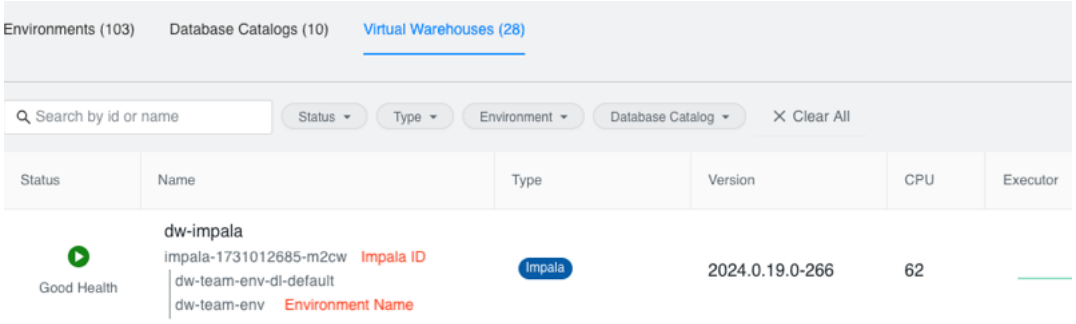

Identifying the bucket name

3. In the Management console navigate to the Environment page.
4. On the Environment page, click on your Environment and then on the Summary tab.

5. On the Logs Storage and Audits section. You can find your log bucket and prefix under the Storage Location. This name is required to identify the S3 location of the logs.

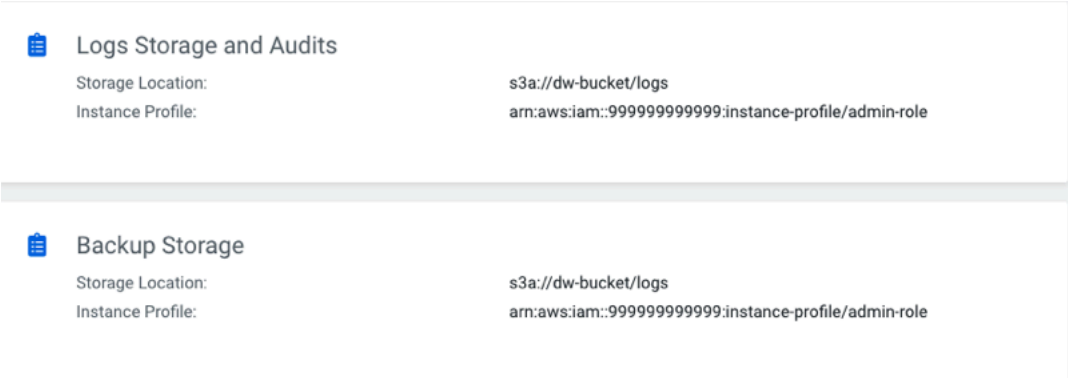

Log locations in S3

6. Now that you have identified the S3 bucket name, environment name, and impala ID identifiers, use the following prefix to find the logs generated by specific components in the following directories. Use the different directories listed here to view Impala/Hue logs.

```
PREFIX =
s3://<s3_bucket_name>/<prefix>/cdw/clusters/<environment_name>/logs/dt=
<date_stamp>/ns=<impala_ID>/
```

Impala component	S3 directory location
impalad	PREFIX + “app=impala-executor-log”
catalogd	PREFIX + “app=catalogd-log”
coordinator	PREFIX + “app=coordinator-log”
auto-scaler	PREFIX + “app=impala-autoscaler-log”
Hue	PREFIX + “app=huebackend-log” PREFIX + “app=hue-huedb-create-job-log” PREFIX + “app=huefrontend-log”
statestored	PREFIX + “app=statestored-log”

The impalad executor logs for 25 August 2024 are located in the following S3 location:

```
PREFIX = s3://<s3_bucket_name>/<prefix>/cdw/clusters/<environment_name>/
logs/dt=2024-08-25/app=impala-executor-log/
```

In the above location, you can find multiple logs that were generated on the specified day.

Impala Minidumps

7. Impala minidumps can be found under the debug-artifacts/impala directory

```
/cdw/clusters/<environment_name>/logs/debug-artifacts/impala/<impala_ID>/
minidump/<pod_name>/
```

Impala Query Profiles

8. Impala query profiles are written in thrift encoded format in this location:

Impala component	S3 directory location

Impala query profiles	PREFIX + “app=impala-profiles”

Use the binary tool to decode thrift to text. This binary tool is provided with the upstream runtime Impala 4.0 as a docker image. Run the following command to use this tool.

```
docker run -i apache/impala:4.0.0-impala_profile_tool < name of the thrift
 encoded file to decode
```

You can use the docker image available here to use this decoding tool.

Locations of Impala Log Files in Azure

This topic describes how to identify the Azure locations of Impala logs for the different Impala components.

About this task

The Cloudera Data Warehouse service collects logs from Impala Virtual Warehouses and uploads them to the Azure storage account that was provided while registering the Environment. This ABFS log location is configured under an external warehouse directory so that the logs are preserved even if the Virtual Warehouse they are collected from is destroyed.

Before you begin

To identify the location of the logs in Azure, you must have the environment name and impala ID identifiers and to access the logs from the Azure Portal you must know your storage account name.

Procedure

Finding the environment name, impala ID identifiers

1. In the Cloudera Data Warehouse service, go to the Virtual Warehouse tab.
2. From the Overview page, note down the environment name and impala ID identifiers.

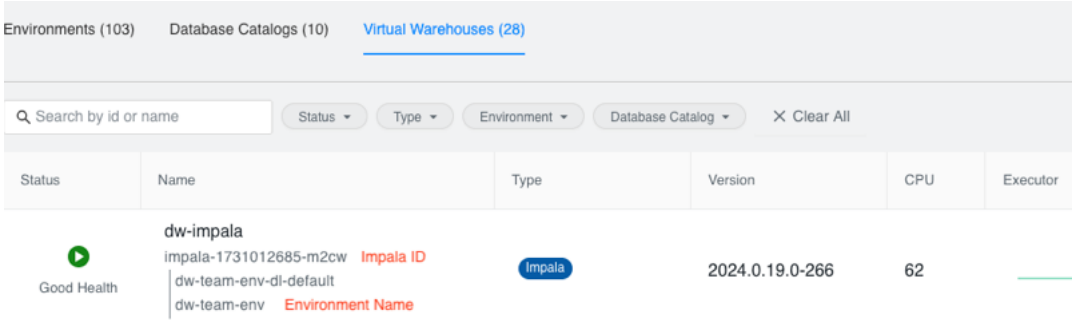

Retrieving your storage account name

3. In the Management Console navigate to the Environments page.
4. On the Environments page, click on your Environment and click on the Summary tab.

5. Scroll down to the Logs Storage and Audits section.

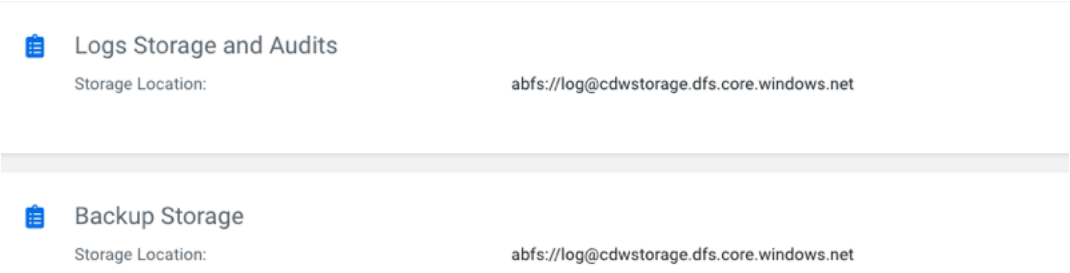

Note down your storage account name.

Accessing the different directories in the Azure Portal

6. Log in to the Azure Portal and search for your storage account name using the Search bar.
7. On the Overview page of your storage account, click on the Containers menu.
8. Click on the file system you used during the Environment registration.

Note: You need to enable the firewall rules, click on the Firewalls and virtual networks menu, and set Allow access to “All networks”, then save the changes to access the file system.

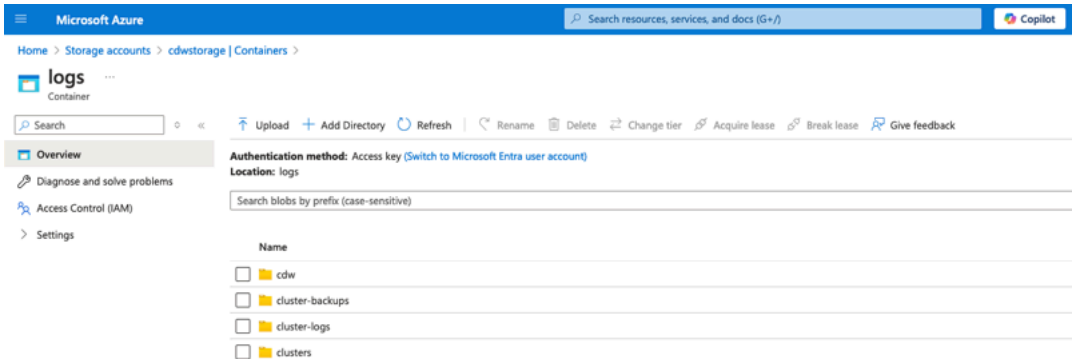

Log locations in ABFS

9. Use the environment name, and impala ID identifiers, in the following prefix to find the logs generated by specific components in the following directories. Use the different directories listed here to view Impala/Hue logs

```
PREFIX =
/cdw/clusters/<environment_name>/logs/dt=<date_stamp>/ns=<impala_ID>/
```

Impala component	ABFS directory location
impalad	PREFIX + “app=impala-executor-log”
catalogd	PREFIX + “app=catalogd-log”
coordinator	PREFIX + “app=coordinator-log”
auto-scaler	PREFIX + “app=impala-autoscaler-log”
Hue	PREFIX + “app=huebackend-log” PREFIX + “app=hue-huedb-create-job-log” PREFIX + “app=huefrontend-log”

statestored	PREFIX + “app=statestored-log”

The impalad executor logs for 25 August 2024 are located in the following ABFS location:

```
PREFIX = /cdw/clusters/<environment_name>/logs/dt=2024-08-25/app=impala-
executor-log/
```

In the above location, you can find multiple logs that were generated on the specified day.

Impala Minidumps

10. Impala minidumps can be found under the debug-artifacts/impala directory

```
/cdw/clusters/<environment_name>/logs/debug-artifacts/impala/<impala_ID>/
minidump/<pod_name>/
```

Impala Query Profiles

11. Impala query profiles are written in thrift encoded format in this location:

Impala component	S3 directory location
Impala query profiles	PREFIX + “app=impala-profiles”

Use the binary tool to decode thrift to text. This binary tool is provided with the upstream runtime Impala 4.0 as a docker image. Run the following command to use this tool.

```
docker run -i apache/impala:4.0.0-impala_profile_tool < name of the thrift
 encoded file to decode
```

You can use the docker image available here to use this decoding tool.

Analyzing Impala Virtual Warehouse Issues with the Events Timeline

The Cloudera Data Warehouse server gathers information about problems that cause Virtual Warehouse errors and displays a tooltip in the Virtual Warehouse tile. Also, from the user interface, you can get details about events that can help you solve problems.

In the Virtual Warehouse tile, you now see tooltips. For example:

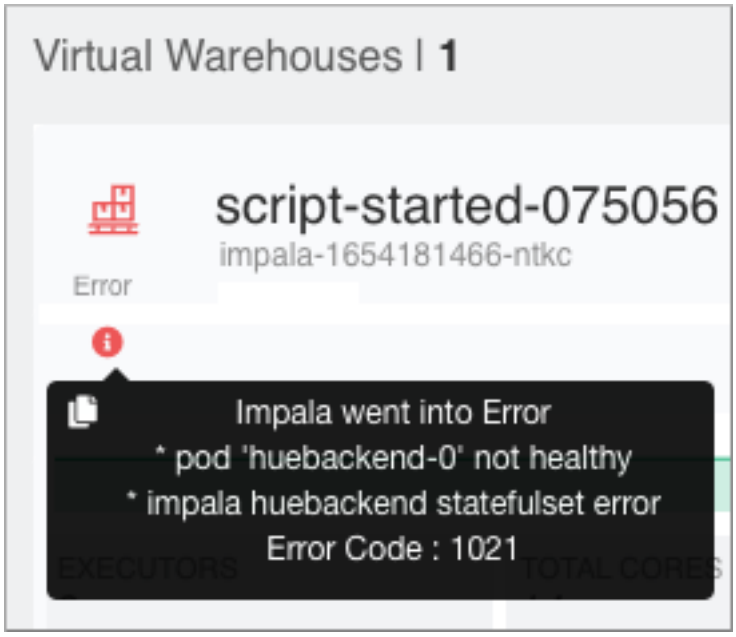

Go to the **Overview** page and navigate to Virtual Warehouses ⋮ Edit Diagnostics Events Timeline , you expand timeline items to get details that can help you troubleshoot problems.

Troubleshooting Impala

This topic describes the general troubleshooting procedures to diagnose some of the commonly encountered issues in Impala.

Symptom	Explanation	Recommendation
Impala takes a long time to start.	Impala instances with large numbers of tables, partitions, or data files take longer to start because the metadata for these objects is broadcast to all `impalad` nodes and cached.	Adjust timeout and synchronicity settings.
Joins fail to complete.	There may be insufficient memory. During a join, data from the second, third, and so on sets to be joined is loaded into memory. If Impala chooses an inefficient join order or join mechanism, the query could exceed the total memory available.	Start by gathering statistics with the COMPUTE STATS statement for each table involved in the join. Consider specifying the [SHUFFLE] hint so that data from the joined tables is split up between nodes rather than broadcast to each node. If tuning at the SQL level is not sufficient, add more memory to your system or join smaller data sets.
Queries return incorrect results.	Impala metadata may be outdated after changes are performed in Hive.	After inserting data, adding a partition, or other operation in Hive, refresh the metadata for the table with the REFRESH statement.
Attempts to complete Impala tasks such as executing INSERT SELECT statements fail. The Impala logs include notes that files could not be opened due to permission denied.	This can be the result of permissions issues. For example, you could use the Hive shell as the hive user to create a table. After creating this table, you could attempt to complete some action, such as an INSERT SELECT on the table. Because the table was created using one user and the INSERT SELECT is attempted by another, this action may fail due to permissions issues.	Ensure the Impala user has sufficient permissions to the table that the Hive user created.
Impala fails to start up, with the `impalad` logs referring to errors connecting to the statestored service and attempts to re-register.	A large number of databases, tables, partitions, and so on can require metadata synchronization, particularly on startup, that takes longer than the default timeout for the statestore service.	Configure the statestore timeout value and possibly other settings related to the frequency of statestore updates and metadata loading.

Debugging Impala Virtual Warehouses

You can use the Catalog Web UI, Coordinator Web UI, and the StateStore Web UI to debug Impala Virtual Warehouses in Cloudera Data Warehouse.

Table level events

In addition to global metrics described below, the following table metrics are available for debugging an Impala Virtual Warehouse:

- avg-events-process-duration
- events-consuming-delay-ms

avg-events-process-duration metric

This metric represents the sum of the time for processing all events. This metric is helpful to identify the average duration of processed events on the table and to identify which tables are causing the event-processor to lag behind. As a temporary workaround, you can disable event processing on that table. You can set the metric collection period to 1 minute, 5 minutes, and 15 minutes duration:

- avg-events-process-duration-1min-rate

  Exponentially weighted moving average (EWMA) of number of events processed in last 1 min
- avg-events-process-duration-5min-rate

  Exponentially weighted moving average (EWMA) of number of events processed in last 5 min
- avg-events-process-duration-1min-rate

  Exponentially weighted moving average (EWMA) of number of events processed in last 15 min

Metric output looks something like this:

localhost:25020/table_metrics?name=default.tb1

```
events-process-duration:
   Count: 2
   Mean rate: 0.0109
   1 min. rate: 0.0066
   5 min. rate: 0.0048
   15 min. rate: 0.002
   Min (msec): 24
   Max (msec): 33
   Mean (msec): 28.9089
   Median (msec): 33.4825
   75th-% (msec): 33.4825
   95th-% (msec): 33.4825
   99th-% (msec): 33.4825

hms-load-tbl-schema:
   Count: 1
   Mean rate: 0.0055
   1 min. rate: 0.0108
   5 min. rate: 0.1116
   15 min. rate: 0.1647
```

events-consuming-delay-ms metric

This metric represents the time difference between creating an event in the metastore and processing an event. Using this metric, you can gauge how long the event processor is lagging.

Metric output looks something like this:

```
localhost:25020/events

    Mean (msec): 67.9801
    Median (msec): 78.3973
    75th-% (msec): 78.3973
    95th-% (msec): 78.6436
    99th-% (msec): 78.6436

events-consuming-delay:
    Count: 12
    Mean rate: 0.0342
    1 min. rate: 0.0017
    5 min. rate: 0.0146
    15 min. rate: 0.0095
    Min (msec): 2000
    Max (msec): 10000
    Mean (msec): 3472.5061
    Median (msec): 2000
    75th-% (msec): 4000
    95th-% (msec): 8000
    99th-% (msec): 10000
```

About this task

The Impala daemons (impalad, statestored, and catalogd) debug Web UIs, which can be used in by using , is also available in the service. In service, the following Web UIs are provided:

- Impala Catalog Web UI

  This UI provides the same type of information as the Catalog Server Web UI in . It includes information about the objects managed by the Impala Virtual Warehouse. For more information about this debug Web UI, see Debug Web UI for Catalog Server.
- Impala Coordinator Web UI

  This UI provides the same type of information as the Impala Daemon Web UI in . It includes information about configuration settings, running and completed queries, and associated performance and resource usage for queries. For information about this debug Web UI, see Debug Web UI for Impala Daemon.
- Impala StateStore Web UI

  This UI provides the same type of information as the StateStore Web UI in . It includes information about memory usage, configuration settings, and ongoing health checks that are performed by the Impala statestored daemon. For information about this debug Web UI, see Debug Web UI for StateStore.
- Impala Autoscaler Web UI

  This UI gives you insight into autoscaler operations (regular as well as workload-aware autoscaling), accessing log messages, and resetting the log level. The autoscaler Web UI includes information about the queries queued and running, executor groups, suspended calls, scale up/down calls, the autoscaler config, and the autoscaler logs.

Required role: EnvironmentAdmin

Before you begin

Make sure that you note your workload user name and have set a password for it in the User Management module of . You need to use your workload user name and its associated password to log into the debug Web UIs. For more information, see Setting the workload password in the documentation set.

Procedure

1. In the UI on the Overview page, locate the Impala Virtual Warehouse for which you want to view the debug UIs, and select ⋮ Edit .

   The **Virtual Warehouse Details** page is displayed.
2. In the **Virtual Warehouse Details** page, select the Web UI tab. The list of debug Web UI links are displayed as shown in the following image:

   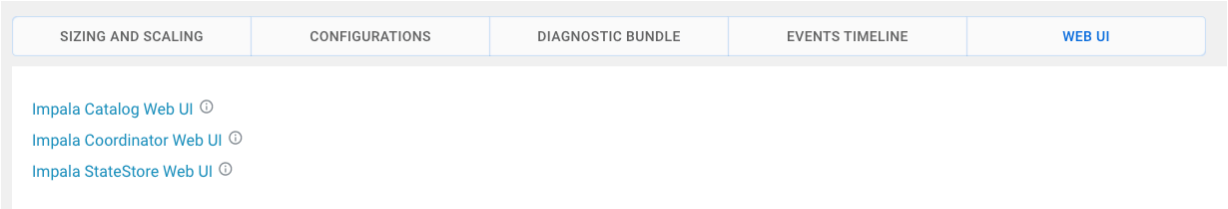

3. Click a Web UI link corresponding to an Impala daemon that you want to debug.

   You are prompted to enter your workload user name and password.

Results

After you are authenticated, you can view the debug Web UI and use the information to help you troubleshoot issues with your Impala Virtual Warehouse.

Cloudera Data Warehouse Public Cloud

Unified Analytics overview

Date published: 2024-01-01
Date modified: 2024-12-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Unified Analytics overview **4**

Unified Analytics architecture **5**

ORC vs Parquet formats **6**

Supported user defined functions **7**

Unified Analytics overview

You can take advantage of SQL engine enhancements in the Cloudera by using Unified Analytics. Unified Analytics includes semantics commonality, backward compatibility, and optimizations.

This release of Unified Analytics is the first product of a long-term vision that brings SQL equivalency without syntax changes to Cloudera Data Warehouse SQL engines. Unified Analytics is available in the following Cloudera environments:

- AWS
- Azure

Unified Analytics SQL engines support SQL:2016 syntax and semantics, including the following key operations:

- Automatic query rewrites to use materialized views
- Command-line materialized view recommender
- DataSketches functions and rewrites
- Ranger column masking and row filtering
- Query results cache
- SQL set operations and grouping sets
- Atlas integration
- Extensive subquery support
- Advanced join reordering with bushy plans generation
- Integrity constraints-based rewritings
- User defined functions (UDFs) in Hive
- Other extensions to query optimization, such as column pruning, sort/limit merge and pushdown

Unified Analytics also brings significant optimization equivalency to the SQL engines, unifying common techniques such as subquery processing, join ordering and materialized views.

Lexical conventions

The default behavior for SQL queries in Unified Analytics is to use single quotation marks for the literals. The ANSI SQL standard is to use single quotation marks for string literals. If a single quotation mark or special character appears within that string literal, it needs to be escaped. For example:

```
INSERT INTO MOVIES_INFO VALUES
(1,cast('Toy Story (1995)' as varchar(50)), 'Animation|Children\'s|Comedy'),
...
```

You can change the SQL behavior. In your Virtual Warehouse, click Edit Configurations HiveServer2 , and add the hive.support.quoted.identifiers property. Set this property to one of the following values:

- none

  Quotation of identifiers and special characters in identifiers are not allowed, but regular expressions in backticks are supported for column names.
- column

  Use the backtick character to enclose identifiers having special characters. `col1`. Use single quotation marks to enclose string literals, for example: 'value'. Double quotation marks are also accepted, but not recommended.
- standard (default)

  SQL standard way to enclose identifiers. Use double quotation marks to enclose identifiers having special characters "col1" and single quotation marks for string literals, for example 'value'.

Limitations

- Unified Analytics does not support left/right ANTI JOIN syntax.
- Unified Analytics does not support complex types - ARRAY, STRUCT, MAP.

Related Information

AWS environments overview

Azure environments overview

Unified Analytics architecture

The architecture of Unified Analytics supports Cloudera's long-term vision to make Impala and Hive engines equivalent in SQL functionality without syntax changes. You learn the part that each major Unified Analytics component in Cloudera Data Warehouse plays in improving your analytics experience.

The architecture brings significant optimization equivalency to the both SQL engines. Each engine has its own style of doing optimizations within the query processor, but there are common techniques such as subquery processing, join ordering and materialized views, which the architecture unifies. The architecture is compatible with the HiveServer (HS2) protocol.

The following diagram shows the Unified Analytics architecture.

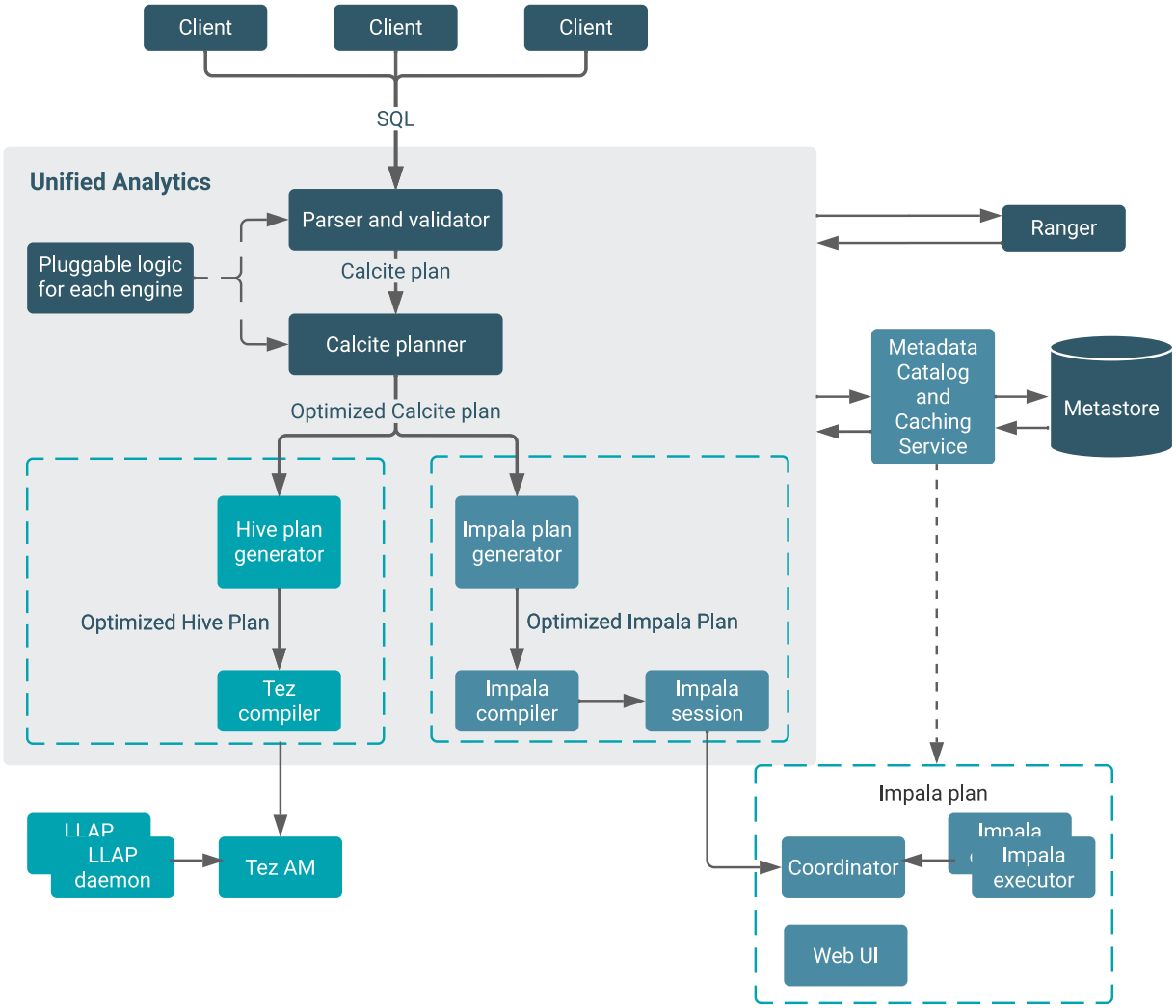

The diagram includes the following components:

Clients

Connect to the Unified Analytics endpoint parser and validator.

Ranger

Secures data and supports column masking and row filtering in both engines; whereas, only column masking is available to the un-unified Impala.

Metastore, metadata catalog and caching service

Shared by both engines

Calcite planner

Receives SQL queries from the parser/validator, and generates a plan for Hive or Impala.

Optimized Calcite plan

Generated for Hive or Impala.

Tez compiler

Hands off Tez tasks that run on LLAP daemons to the Tez Application Manager.

Impala session

Connects to the coordinator, runs on executors, and the Web UI shows runtime query profiles for diagnosing problems.

Tez AM and LLAP daemons

The Tez Application Manager (AM) runs the Hive job in LLAP mode, obtaining free threads from LLAP daemons to quickly execute fragments of the plan.

Impala plan

Capabilities and strengths of Impala, such as different types of join algorithms, different types of distribution, broadcast and partitioning for example, generate an Impala specific plan if your client connected to the Impala Virtual Warehouse.

Calcite provides SQL query planning capabilities to both engines. This represents no change to Hive, but adds many new enhancements to Impala. Because the Calcite logic is pluggable for each engine, Unified Analytics can support engine differences, such as type checking, and data and timestamp functions. The pluggable logic based on engine semantics generates appropriate expressions and types. You realize the following benefits of this logic:

- Function validation
- Type system implementation
- Expressions simplification
- Static partition pruning

For example, multiplying decimals returns results in Hive output different from the Impala output. This architecture makes backward compatibility and forward commonality possible. The pluggable framework expedites adding new Calcite transformation rules for delivering new cost-based optimizations.

Partition pruning and predicate push-down are supported by Hive and Impala. Unified Analytics seamlessly handles partition pruning semantics, which can differ because you can have expressions in the pruning predicates. Support for join reordering, materialized view rewriting, and other capabilities, formerly only in Hive, are available in Impala. The query results cache supported in Hive 3 is also now available to Impala.

ORC vs Parquet formats

The differences between Optimized Row Columnar (ORC) file format for storing data in SQL engines are important to understand. Query performance improves when you use the appropriate format for your application.

ORC and Parquet capabilities comparison

The following table compares SQL engine support for ORC and Parquet.

Table 1:

Capability	Data Warehouse	ORC	Parquet	SQL Engine
Read non-transactional data	Apache Hive	#	#	Hive
Read non-transactional data	Apache Impala	#	#	Impala
Read/Write Full ACID tables	Apache Hive	#		Hive
Read Full ACID tables	Apache Impala	#		Impala
Read Insert-only managed tables	Apache Impala	#	#	Impala
Column index	Apache Hive	#	#	Hive
Column index	Apache Impala		#	Impala
CBO uses column metadata	Apache Hive	#		Hive
Recommended format	Apache Hive	#		Hive
Recommended format	Apache Impala		#	Impala
Vectorized reader	Apache Hive	#	#	Hive
Read complex types	Apache Impala	#	#	Impala
Read/write complex types	Apache Hive	#	#	Hive

Supported user defined functions

Unified Analytics supports user defined functions in both SQL engines. You learn which types of UDFs are supported and see usage examples.

You can use the following types of UDFs:

- Hive Java UDF

```
create function hive_f(int) returns int LOCATION
S3a://my-bucket/path.jar SYMBOL='my_java_func'
```

- Native C++ UDFs created in Impala

```
create function native_fn(int) returns
int LOCATION S3a://my-bucket/path.so SYMBOL='my_cpp_func'
```

- Native C++ UDAs created in Impala

```
create aggregate function agg_fn(int)
returns int LOCATION S3a://my-bucket/path.so SYMBOL='agg_func'
```

After creating the C++ functions, you must run RELOAD FUNCTION or RELOAD FUNCTIONS to load UDFs you created from Impala into HiveServer (HS2). For example:

```
RELOAD FUNCTIONS
```

For example:

```
select native_fn(i1) from tbl1;
| _c0  |
3
3

select agg_fn(i1) from tbl1;
| _c0  |
2

select hive_f(s1) from tbl1;
| _c0  |
119
104
```

Cloudera AI

Using Cloudera AI Inference service

Date published: 2020-07-16
Date modified: 2025-01-29

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Cloudera AI Inference service.....5
- Key Features.....5
- Key Applications.....5
- Terminology.....6
- Limitations and Restrictions.....6
- Supported Model Artifact Formats.....7

Prerequisites for setting up Cloudera AI Inference service.....7
- Authentication of Cloudera AI Inference service.....8
- Authorization ofCloudera AI Inference service.....8
- Importing Models.....9
- Register an ONNX model to Cloudera AI Registry.....9

Cloudera AI Inference service Concepts.....10

Cloudera AI Inference service Configuration and Sizing.....11

Managing Cloudera AI Inference service.....15
- Managing Cloudera AI Inference service using Cloudera Data Platform CLI.....15
  - Creating a Cloudera AI Inference service instance.....15
  - Listing Cloudera AI Inference service instances.....17
  - Describing Cloudera AI Inference service instance.....18
  - Managing Node Groups.....18
  - Deleting Cloudera AI Inference service instance.....20
  - Obtaining Control Plane Audit Logs for Cloudera AI Inference service.....20
  - Obtaining the kubeconfig for the Cloudera AI Inference service Cluster.....20
- Managing Cloudera AI Inference service using the UI.....22
  - Creating a Cloudera AI Inference service instance using the UI.....22
  - Listing Cloudera AI Inference service instances using the UI.....24
  - Viewing details of a Cloudera AI Inference service instances using the UI.....24
  - Managing node groups using the UI.....24
  - Deleting Cloudera AI Inference service instances using the UI.....25
  - Obtaining Control Plane Audit Logs for Cloudera AI Inference service using the UI.....26
  - Obtaining the kubeconfig of Cloudera AI Inference service using the UI.....26

Managing Model Endpoints using UI.....26
- Creating a Model Endpoint using UI.....27
- Listing Model Endpoints using UI.....29
- Viewing details of a Model Endpoint using UI.....30
- Editing a Model Endpoint Configuration using UI.....30

Managing Model Endpoints using API.....31
- Preparing to interact with the Cloudera AI Inference service API.....32

Creating a Model Endpoint using API.... 32
Listing Model Endpoints using API.... 33
Describing a Model Endpoint using API.... 33
Deleting a Model Endpoint using API.... 34
Autoscaling Model Endpoints using API.... 35
Tuning auto-scaling sensitivity using the API.... 35
Running Models on GPU.... 36
Deploying models with Canary deployment using API.... 37

Interacting with Model Endpoints.... 38
Making an inference call to a Model Endpoint with an OpenAI API.... 38
Cloudera AI Inference service using OpenAI Python SDK client in a Cloudera AI Workbench Session.... 38
Cloudera AI Inference service using OpenAI Python SDK client on a local machine.... 39
OpenAI Inference Protocol Using Curl.... 40
Making an inference call to a Model Endpoint with Open Inference Protocol.... 40
Open Inference Protocol Using Python SDK.... 41
Open Inference Protocol Using Curl.... 41

Deploying Predictive Models.... 43

Accessing Cloudera AI Inference service Metrics.... 48

Known issues.... 48

Using Cloudera AI Inference service

Cloudera AI Inference service provides a production-grade serving environment for hosting predictive and generative AI. It is designed to handle the challenges of production deployments, such as high availability, performance, fault tolerance, and scalability. The Cloudera AI Inference service allows data scientists and machine learning engineers to deploy their models quickly, without worrying about the infrastructure and maintenance.

Cloudera AI Inference service is built with tight integration of *NVIDIA NIM* and *NVIDIA Triton Inference Server*, providing industry-leading inference performance on NVIDIA GPUs. Model endpoint orchestration and management are built with the *KServe* model inference platform, a Cloud Native Computing Foundation (CNCF) open-source project. The platform provides standard-compliant model inference protocols, such as *Open Inference Protocol* for predictive models and *OpenAI API* for generative AI models.

Cloudera AI Inference service is seamlessly integrated with the *Cloudera AI Registries* enabling users to store, manage, and track their AI models throughout their lifecycle. This integration provides a central location to store model and application artifacts, metadata, and versions, making it easier to share and reuse AI artifacts across different teams and projects. It also simplifies the process of deploying models and applications to production, as users can select artifacts from the registry and deploy them with a single command.

Related Information

KServe
Open Inference Protocol
OpenAI API
NVIDIA NIM
NVIDIA Triton Inference Server
Cloudera AI Registry

Key Features

The key features of Cloudera AI Inference service includes:

- Easy to use interface: Streamlines the complexities of deployment and infrastructure, meaningfully reducing time to value for AI use cases.
- Real-time predictions: Allows users to serve AI models in real-time, providing low latency predictions for client requests.
- Monitoring and logging: Includes functionality for monitoring and logging, making it easier to troubleshoot issues and optimize workload performance.
- Advanced deployment patterns: Includes functionality for advanced deployment patterns, such as canary and blue-green deployments, and supports A/B testing, enabling users to deploy new versions of models gradually and compare their performance before deploying them to production.
- Optimized Performance: Integrates with NVIDIA NIM microservices and NVIDIA Triton Inference Server to accelerate inference performance on NVIDIA accelerated infrastructure.
- Model access: Offers access to NVIDIA foundation models, tailored for NVIDIA hardware to increase inference throughput and to reduce latency.
- REST API: Provides APIs for deploying, managing, and monitoring of model endpoints. These APIs enable integration with continuous integration and continuous deployment (CI/CD) pipelines and other tools used in the Machine Learning Operations (MLOps) and Large Language Model Operations (LLMOps) workflows.

Key Applications

Large Language Models deployed on Cloudera AI Inference service with NVIDIA NIM enable the following applications:

- Chatbots & Virtual Assistants: Empowers bots with human-like language understanding and responsiveness.
- Content Generation & Summarization: Generates high-quality content or distills lengthy articles into concise summaries with ease.
- Sentiment Analysis: Understands user sentiments in real-time, driving better business decisions.
- Language Translation: Breaks language barriers with efficient and accurate translation services.

Terminology

Lists the Cloudera AI Inference service terminology and usage.

- CML Serving App: This is the term used by the Cloudera Data Platform CLI to refer to a specific instance of Cloudera AI Inference service.
- Model Endpoint: This refers to a deployed model that has a URL endpoint accessible over the network.
- Model Artifacts: Files stored in Cloudera AI Registry that are necessary for deploying an instance of the model, such as model weights, metadata, and so on.
- API standard: The protocol that is exposed by a Model Endpoint. It can be either OpenAI (for NVIDIA NIM) or Open Inference Protocol for predictive models.
- Cloudera Data Platform Workload Authentication Token: The bearer token used for authentication / authorization when accessing Cloudera AI Inference service API and model endpoints. Throughout this document this is referred to as “CDP_TOKEN”.
- Model ID: This is the ID assigned to the model when it is registered to the Cloudera AI Registry.
- Model Version: This is the version of a registered model in the Cloudera AI Registry.

Limitations and Restrictions

Lists the limitations and restrictions when using Cloudera AI Inference service.

- API Stability: Both the Cloudera AI Control Plane and Cloudera AI Inference service workload APIs and CLIs are under active development and are subject to change in a backward-incompatible way.
- Cloud Platforms: Cloudera AI Inference service is available on AWS and Azure.
- Supported Instance Types: Cloudera AI Inference service supports the same cloud instance types as those of Cloudera AI Workbenches with a few exceptions. See *Known Issues* for information on unsupported instance types. The type or size of the model you want to deploy determines the cloud compute instance type. Some highly optimized versions of Large Language Models, for instance, work only on specific GPU architectures.
- No Non-Transparent Proxy Support: Cloudera AI Inference service has not been tested with a non-transparent proxy (NTP) setup in a private cluster. However, it works in a vanilla private cluster.
- No UDR Support in Azure: Cloudera AI Inference service has not been tested with User-Defined Route (UDR) setup of Azure clusters.
- Public Load Balancer: By default, Cloudera AI Inference service uses a private load balancer for cluster ingress. If you use a public load balancer instead, set the use PublicLoadBalancer parameter value to true in the creation payload.

  If you are on AWS and use a private load balancer for cluster ingress, you must have a VPN connection between your corporate network and the Virtual Private Cloud (VPC) in which the Cloudera AI Inference service is deployed. The Cloudera AI Inference service UI requires VPN connection.
- Logging: All Kubernetes pod logs, including pods that are running model servers, are scraped by the platform log aggregator service (fluentd). Model endpoint logs can be viewed from the Cloudera AI Inference service GUI. To view logs of other pods, you must first obtain the kubeconfig of the cluster and use the `kubectl` command. It is not possible to retrieve historical pod logs, and therefore there is no Diagnostic bundle feature at this time.
- Namespace: Model endpoints can only be deployed in the serving-default namespace.

Related Information

Managing Cloudera AI Inference service using Cloudera CLI

Known Issues

Supported Model Artifact Formats

Lists Cloudera AI Inference service supported models:

- Text-generating and embedding Large Language Models (LLMs) packaged as *NVIDIA NIM*.
- Predictive models in the ONNX representation, registered to Cloudera AI Registry from a Cloudera AI Workbench. See *Register an ONNX model to Cloudera AI Registry* as an example showing how to convert a sklearn model to ONNX and then register it to the Cloudera AI Registry. Refer to *Export a PyTorch model to ONNX* or *Getting Started Converting TensorFlow to ONNX* documentation regarding how models using these frameworks can be converted to the ONNX representation.

Related Information

Register an ONNX model to Model Registry
Getting Started Converting TensorFlow to ONNX, ONNX Runtime documentation
Export a PyTorch model to ONNX, PyTorch documentation
Register an ONNX model to AI Registry
NVIDIA NIM Large Language Models

Prerequisites for setting up Cloudera AI Inference service

Consider the following prerequisites before setting up Cloudera AI Inference service.

Cloudera Data Platform CLI

You must dowload Cloudera Data Platform CLI and install it on your machine to create a Cloudera AI Inference service instance. For information on how to download Cloudera Data Platform CLI, in the Cloudera Data Platform console, click on the left navigation Help Download CLI . You must use Cloudera Data Platform CLI version 0.9.123 or higher with Cloudera AI Inference service.

Compute cluster-enabled Cloudera Data Platform environment

Cloudera AI Inference service requires a compute cluster-enabled Cloudera Data Platform environment. You must either create the compute cluster-enabled environment, or convert your existing environment.

Note: You cannot deploy Cloudera AI Inference service into the **default compute cluster** in the environment. You must create a separate compute cluster.

Cloudera AI Inference service is compatible with either a Containerized Data Lake (CDL) or the VM based Data Lake. If you convert an existing environment to a compute cluster-enabled environment, the environment will retain its existing VM-based data lake.

Data Lake and Cloudera Manager supported versions

JSON Web Token based authentication from Cloudera AI Workbenches to Cloudera AI Inference service requires Data Lake version 7.2.18 or above, and Cloudera Manager version 7.12 or above.

Cloudera AI Registry

A Cloudera AI Registry must first be deployed in the same Cloudera Data Platform environment in which you plan to deploy Cloudera AI Inference service. That is, Cloudera AI Inference service can only deploy models registered to a Cloudera AI Registry in the same Cloudera Data Platform environment. For this release, Cloudera AI Registry version CML2024.09-1 or higher version must be created before provisioning the Cloudera AI Inference service. If you have an existing Cloudera AI Registry in the environment, you must first upgrade it to version CML2024.09-1 before provisioning Cloudera AI Inference service.

Related Information

Installing Cloudera client
Using Compute Clusters in AWS environment
Using Compute Clusters in Azure environment
Setting up AI Registry

Authentication of Cloudera AI Inference service

Cloudera AI Inference service uses Cloudera Data Platform Workload Authentication JSON Web Token (JWT) to authenticate users/clients that interact with all HTTP endpoints exposed by the service workload.

External Clients' Authentication

External clients that run outside of Cloudera Private Cloud Data Services and that interact with Cloudera AI Inference service API and with model endpoints must obtain a JWT from the Cloudera Data Platform control plane, which must be passed as a bearer token in HTTP requests sent to the Cloudera AI Inference service API and model endpoints.

1. Obtain the JWT using Cloudera Data Platform CLI.

   ```
   $ CDP_TOKEN=$(cdp iam generate-workload-auth-token --workload-name [***W
   orkload name***]
    | jq -r '.token')
   ```

   The same workload name (DE) can be used for authentication to Cloudera AI Inference service as well.
2. Pass CDP_TOKEN in the HTTP request header.

   ```
   $ curl -H "Authorization: Bearer ${CDP_TOKEN}" [***URL***]
   ```

   The token obtained using this method expires in one hour by default. You must obtain a new token if your token is rejected by the server as expired. The server responds with a 401 HTTP response code if the token has expired.
3. Create tokens with longer lifetimes than the default 1 hour using the following Cloudera Data Platform command:

   ```
   $ cdp iam set-authentication-policy --workload-auth-token-expiration-sec
    [***EXPIRATION-TIME-IN-SECONDS***]
   ```

   Important: This setting is scoped for the Cloudera Data Platform account, it is not specific toCloudera AI Inference service. Increasing the token expiration has security implications. In contrast with API keys, JWT tokens cannot be revoked. Also note that the above command requires the AUTHENTICATION_POLICY CDP entitlement.

Internal Clients Authentication

Authenticated clients or users running inside another Cloudera services, like a Cloudera AI Workbench, or Hue UI in the same environment, can pass Data Lake issued JWTs for authentication to Cloudera AI Inference service running in the environment. For instance, inside a Cloudera AI Workbench session, this JWT is available as the access_token field in the /tmp/jwt file.

Authorization ofCloudera AI Inference service

Cloudera AI Inference service implements role-based access control.

To create an instance of the service in a Cloudera Data Platform environment, and to perform modifications to the instance, such as the addition of a node group, the user must have the following roles:

- EnvironmentAdmin
- MLAdmin

Model endpoints can be viewed, created, deleted, and modified by users having EnvironmentUser role along with either one of the following roles:

- MLAdmin
- MLUser

To perform inference on deployed model endpoints, the user must have the MLUser role.

Related Information

Granting Cloudera Users access to Cloudera AI Workbenches

Synchronize Users

Importing Models

Model Hub offers a curated list of top-performing models from the NVIDIA NGC Catalog and Hugging Face.

The **Model Hub** page displays the list of different models along with their source type, tags, and description. You can import models listed on the **Model Hub** page into the **AI Registries**. Imported models can be seen on the **Registered Models** page, and can be deployed using Cloudera AI Inference service.

Importing NVIDIA Foundation Model NIM

You can import NVIDIA NGC Catalog models listed on the **Model Hub** page into the Cloudera AI Registry. Imported models can be seen on the **Registered Models** page, and can be deployed using Cloudera AI Inference service.

For more information, see *Importing models from NVIDIA NGC*.

Importing a Hugging Face Model

You can import Hugging Face models listed on the **Model Hub** page into the Cloudera AI Registry. Imported models can be seen on the **Registered Models** page, and can be deployed using Cloudera AI Inference service.

For more information, see *Importing a Huggingface Model from Model Hub*.

If your desired Hugging Face model is unavailable on the **Model Hub** page, you can import those models from the Hugging Face website from the **Registered Models** page.

For more information, see *Importing a Huggingface Model from the Hugging Face website*.

Related Information

Importing models from NVIDIA NGC

Importing a Huggingface Model from Model Hub

Importing a Huggingface Model from the Hugging Face website

Register an ONNX model to Cloudera AI Registry

Register your ONNX model to Cloudera AI Registry.

To deploy a predictive ONNX model, first you must register your model to the Cloudera AI Registry running in the same environment as your Cloudera AI Inference service.

For information on how to register an ONNX model from a Cloudera AI Workbench, see *Deploying a Predictive Deep Learning Model*. For more information, see Registering a model using the Cloudera AI Registry user interface.

Related Information

Deploying a Predictive Model

Registering a model using the AI Registry user interface

Cloudera AI Inference service Concepts

Learn the followingCloudera AI Inference service concepts before setting up the Cloudera AI Inference service.

Platform

Cloudera AI Inference service is a Kubernetes-native model inference platform built using the Cloud Native Computing Foundation (CNCF)-hosted KServe model orchestration system. The Cloudera AI Inference service is built with enterprise-grade scalability, security and performance incorporated via integrations with NVIDIA NIM, NVIDIA Triton, and Cloudera Data Platform. It is designed to serve trained models for production-level use cases.

Runtime

Runtimes are the basic building blocks that are responsible for loading trained model artifacts into memory, and providing APIs that client applications can invoke to run inference requests. The Runtimes also provide metrics with which to monitor the performance of the models. The supported Runtimes in this release include various NVIDIA NIM versions and Hugging Face transformer for text-generation and embedding tasks, and NVIDIA Triton for deep-learning models using the ONNX backend.

Autoscaling

Cloudera AI Inference service provides Cluster Node autoscaling and Model Endpoint autoscaling.

Cluster Node Autoscaling

If aCloudera AI Inference service runs on an autoscaling Kubernetes cluster, it can be configured with multiple auto-scaling worker node groups. There are two default node groups that are meant to run certain core services, and an arbitrary number of worker node groups where user workloads are scheduled. You can add or delete worker node groups from the cluster. The autoscaling range of an existing node group in the cluster can be changed.

Model Endpoint Autoscaling

In addition to worker node autoscaling, Cloudera AI Inference service provides autoscaling at the model endpoint level by increasing or decreasing the number of replicas based on predefined, customizable scaling criteria. Scaling to or from zero replicas is also supported. The supported scaling criteria (or metrics) are Requests Per Second (RPS) and concurrency per replica of the model endpoint. For instance, you can configure your model endpoint to autoscale up if the number of concurrent RPS exceeds 100, so that request latencies are maintained at an acceptable level.

Model endpoint replicas are terminated when they have not received any request for a certain amount of time. For large language models, this idle timeout is set to 1 hour, whereas for other models it is set to 10 minutes.

Note: When the autoscale range lower bound is set to zero for model endpoints, the model endpoints are created without any active replicas. Therefore, these endpoints do not consume resources and will not be reachable until the first request is received. This behavior can result in an endpoint successfully created, but unable to scale up and accept traffic. Though this behavior is beneficial to save cloud costs and allows users to enable scale to zero without starting up any new pods, it also makes it harder to diagnose node group sizing and quota issues in a timely manner.

Autoscaling Latency

It is important to be aware of the time it takes to autoscale-up a model replica, which can adversely affect user experience. The following mathematical expression can be used to calculate the approximate scale-up latency:

```
T = Tn + Tc + Td + Tm
```

where:

- Tn: The time needed for a newly scaled up worker node to be ready. This may be zero if the new model replica is scheduled on a node that is already in the cluster. It can also be very large to infinite, depending on the availability of the instance type requested. In general, larger instance types take longer to reach the ready status.
- Tc: The time needed to pull container images of the model replica pod. This is negligible if the node already has the images.
- Td: The time taken to download the model artifacts from the Cloudera AI Registry storage to the instance volume. This can vary widely depending on the size of the model, ranging from a few seconds to even hours for the largest models. For large language models, such as Llama 3.1 70b or bigger, this term is the dominant one.
- Tm: This is the time required to load the model objects from the instance volume to one or more GPU RAMs for models that require GPU, or to the system memory for CPU-only models.

So based on the above equation, you can see that, for example, to scale up a large language model deployment from 0 to 1 replica can take a few minutes to an hour or more. You must keep this in mind when planning your model deployments and the expected SLAs.

Canary Rollout and Rollback

Cloudera AI Inference service lets you roll out a new version of a model without bringing down the currently running version, using the canary rollout feature. When a new model endpoint is created using version A of a model, 100% of the traffic is routed by the system to the deployed model version. You can roll out another version, say B, under the same model endpoint URL and give it a percentage of the traffic, say 10%. If the new version rolls out successfully, the system will send 90% of the traffic to version A, and 10% to version B. If version B does not successfully come up for some reason, the system will continue sending 100% of the traffic to the good version, which is A.

Note that each version of the model that is rolled out consumes the same amount of system resources allocated to each model endpoint replica. Therefore, if each replica is configured to use 2 CPUs and 5Gi of memory, the resource footprint will double when versions A and B are running. If version B turns out to be good, you can send 100% traffic to it and have version A be scaled to 0 replicas to reduce the resource consumption back to that of a single version. If, after sending 100% traffic to version B, you realize that version A is preferred, there is a way to rollback to the previous version. You can set the traffic for version B to 0% and the traffic is automatically sent back to version A. Setting the traffic to 0% rolls back to the last version that was ready with 100% traffic.

Example:

Suppose you begin with version A receiving 100% of the traffic. A new model, version B, is introduced and allocated 25% of the traffic. Upon evaluation, version B underperforms compared to version A, based on the collected metrics. While version B is still live, you proceed to deploy version C with 100% traffic. However, version C also underperforms relative to version A. Instead of redeploying version A manually, you can set version C's traffic to 0%, and all traffic reverts to version A. Note that version B, being an intermediary version that never reached 100% traffic, is not eligible for rollback.

Related Information

KServer Documentation Website

Cloudera AI Inference service Configuration and Sizing

Consider the following factors for the configuration and sizing of Cloudera AI Inference service.

Node Group Configuration

The configuration and size ofCloudera AI Inference service cluster is determined by the nature of the workloads you expect to deploy on the platform. Certain models might require GPUs, while other kinds of models might run only on CPUs. Similar considerations must be taken for the model endpoints. For example, you must determine the number of replicas of a model that are required to handle normal inference traffic, and there could be peak traffic for which additional replicas shall be spun up to keep user experience at acceptable levels.

Note: Cloudera AI Inference service uses a rolling update strategy for all model endpoints. Consequently, during a rolling update, both the older and newer revisions of the model endpoint are running at the same time until the update is completed. Therefore, your cluster must be able to accommodate this expanded resource footprint in order for the rolling update to succeed.

For example, when you deploy the following NVIDIA NIM for Llama 3.1:

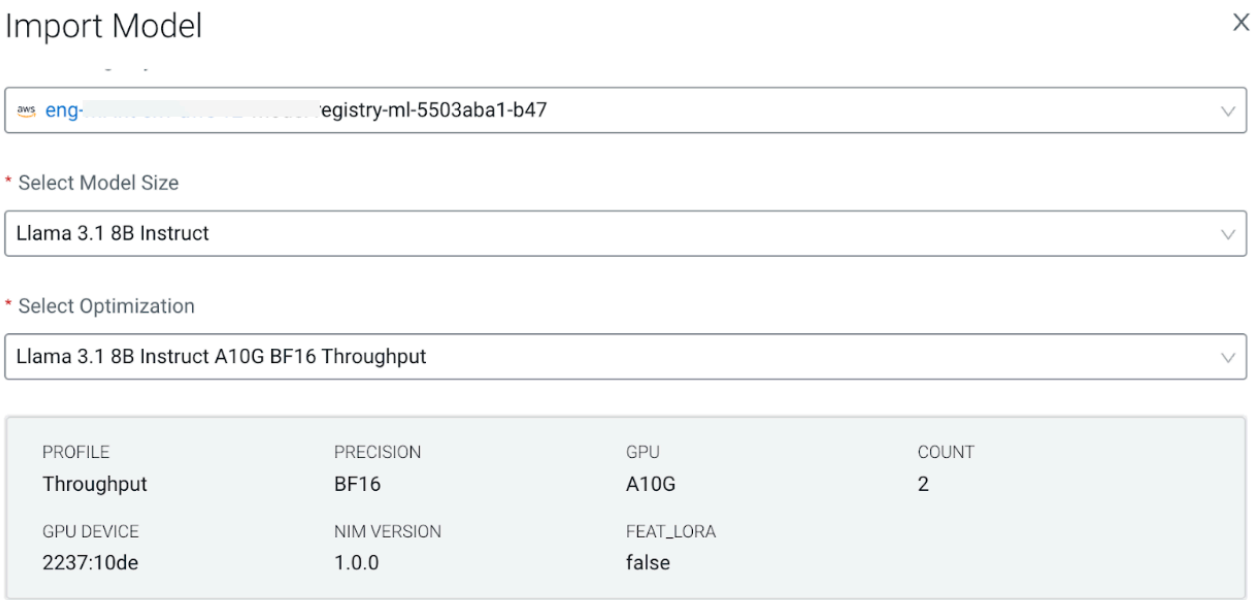

In the above image, you can see that each replica of this model requires two A10G GPUs (due to the model being optimized for A10G with a tensor parallelism of two). You can assume that two replicas are required to handle normal traffic, but an additional replica would be required during peak traffic. Therefore, you must configure the model endpoint to autoscale between two and three replicas. Consequently, for normal traffic 4 A10G GPUs must be allocated, and two more for peak traffic. To ensure seamless rolling update for this model endpoint, and assuming updates are made during off-peak traffic, your cluster must be able to add eight A10G GPUs on-demand. An optimal node group configuration for this scenario on AWS would be one that has 0-2 instances of the g5.12xlarge instance type. One instance runs two replicas during normal load, and the second instance is added using autoscaling during peak traffic, and during rolling update. Another, less cost-efficient, possibility is to use a single g5.48xlarge instance type in the node group, in which case all the resource headroom is available on a single node. The larger node also helps to ensure that autoscaling model replicas and rolling updates are quicker as you do not have to wait for a new node to be spun up, and the container images to be pulled.

Instance Volume Sizing

Cloudera AI Inference service service downloads model artifacts to the instance volume (also known as root volume) of the node where the model replica pod is scheduled. Large Language Model artifacts can be tens to hundreds of gigabytes large. The required instance volume size for a given node group can be estimated using the following formula:

```
S # So + #ni=1riSi + Sc
```

Where:

- So: Size of storage required by the operating system, typically 30 to 40 GB.
- Si: Size of the *I*-th model replica artifacts on a node.
- ri: Number of replicas of the *I*-th model on the node.
- Sc: Total size of all container images on the node. This is dominated by model runtime container images.

For instance, if you want to run 2 replicas of the instruction-tuned Llama 3.1 70b at FP16 precision on a node, you would need something like the following:

```
S ~ 40 + 2*148 + 20 = 356 GB
```

Where it is assumed that the aggregate size of container images is 20 GB.

Cloudera recommends that the instance volume is slightly over-provisioned to ensure that you do not run out of disk space. In the above example, for instance, it is recommended to round up to 512 GB of instance volume. A larger instance volume also provides higher IOPs, which helps reduce model endpoint startup times. Note that the instance volume of an existing node group cannot be modified. You must first delete the node group and then add it back to the cluster with the new instance volume size.

Choosing an NVIDIA NIM Profile

The following guidance here is in the context ofCloudera AI Inference service. See *Nvidia documentation* for information about NVIDIA NIM profiles.

NVIDIA NIM comes in three kinds of optimization profiles:

- Latency: This profile minimizes Time to First Token (TTFT) and Inter-Token Latency (ITIL) by using higher tensor parallelism, that is, using more GPUs.
- Throughput: This maximizes the token throughput per GPU by utilizing the minimum number of GPUs to host the model.
- Generic: Unlike the first two profiles, this profile uses the vLLM backend to load the model and run inference against it. This profile provides the most flexibility in terms of choosing GPU models at the cost of lower performance. Note that not all NIMs provide the generic profiles.

For a given precision, the latency profile provides the highest performance by utilizing the maximum number of GPUs while the generic profile offers the most flexibility by sacrificing performance and precision. The throughout profile strikes a good balance between the other two.

Cloudera AI Inference service lets you choose which NVIDIA NIM profile to deploy, so that only artifacts specific to the chosen profile are downloaded to your Cloudera AI Registry to save on storage costs.

The choice of profile is determined by the following:

- Hardware budget - which cloud instance types you have access to.
- Model performance requirement in terms of latency and throughput.

As an example, let us look at some of the available profiles for the instruction-tuned Llama 3.1 8b model. Each entry in the Optimization picker specifies the model name, GPU architecture, floating point precision, and profile type:

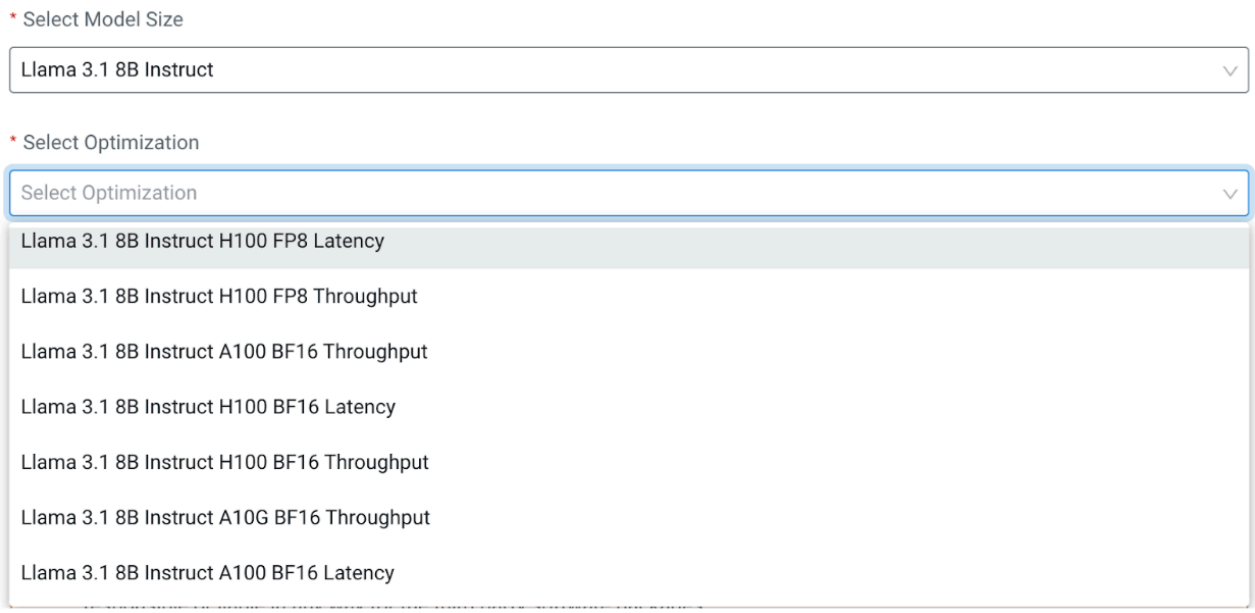

Let us compare the difference between A100 BF16 Throughput and A100 BF16 Latency profiles:

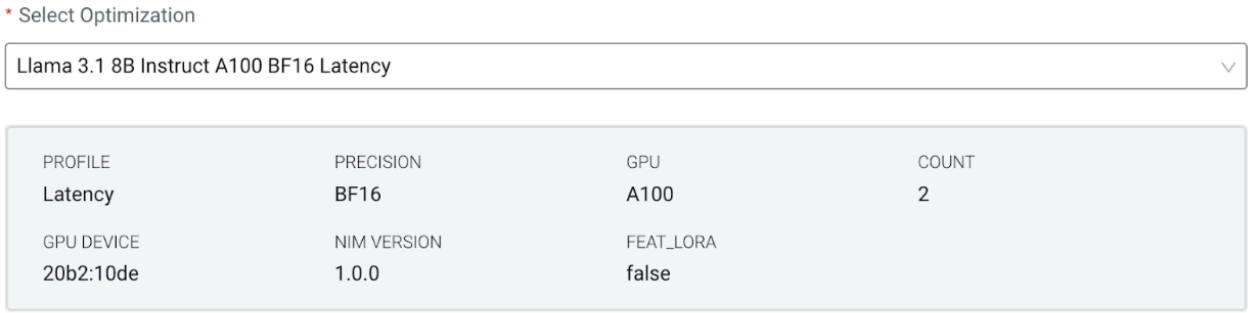

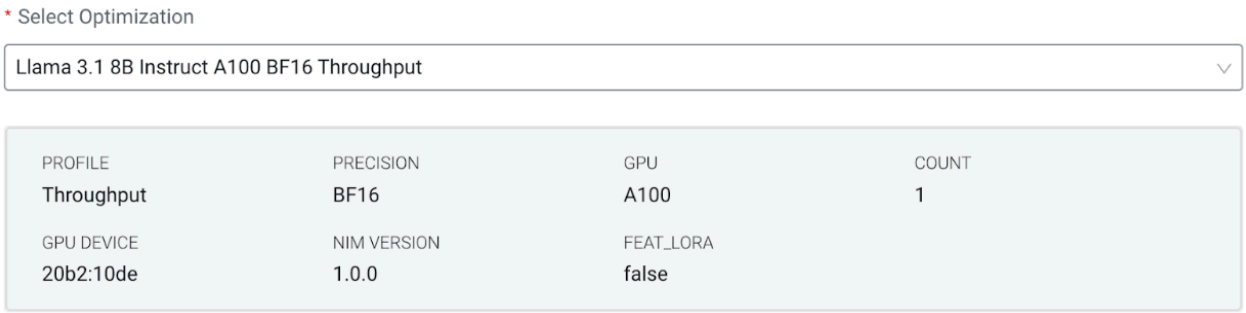

As shown in the figures, the GPU count per model replica of the latency profile is double the size of the throughput profile.

Quantization, for example, FP8 vs BF16. A quantized model, if available, will have lower resource footprint and higher performance (latency and throughput) than a non-quantized one.

The actual latency and throughput as seen by a client application is affected by the end-end performance of the network between the client and the model server, which will include authentication and authorization checks, in addition to the performance of the chosen NVIDIA NIM profile, as well as the number of concurrent connections. For any chosen profile, end-to-end latency and throughput can be optimized by increasing the number of model endpoint replicas for the model.

Related Information

Nvidia Documentation Website

Managing Cloudera AI Inference service

Cloudera AI Inference service provides an UI and CLI interface to manage the life cycle of the service and associated infrastructure.

Managing Cloudera AI Inference service using Cloudera Data Platform CLI

Cloudera AI Inference service provides a CLI interface to manage the life cycle of the service and associated infrastructure.

Creating a Cloudera AI Inference service instance

The recommended way to create a Cloudera AI Inference service is to first generate the CLI input skeleton, customize the JSON file, and then pass the file to the creation command.

The following example present how to create a Cloudera AI Inference service by generating the CLI input skeleton, customizing the JSON file, and then passing the file to the creation command:

1. Generate the JSON skeleton payload and save it to a file:

```
$ cdp ml create-ml-serving-app --generate-cli-skeleton > /tmp/create-ser
ving-app-input.json
```

2. Customize the JSON file to use it when creating a Cloudera AI Inference service instance. The following are the sample JSON files of AWS and Azure:

   AWS JSON file example

```
{
    "appName": "my-aws-caii-cluster",
    "environmentCrn": "[***CDP-ENVIRONMENT-CRN***]",
    “clusterCrn”: “[***COMPUTE-CLUSTER-CRN***]”,
    "provisionK8sRequest": {
        "instanceGroups": [
            {
                "instanceType": "m5.4xlarge",
                "instanceTier": "ON-DEMAND",
                "instanceCount": 1,
                "name": "[***OPTIONAL-LEAVE BLANK***]",
                "rootVolume": {
                    "size": 256
                },
                "autoscaling": {
                    "minInstances": 0,
                    "maxInstances": 5,
                    "enabled": true
                }
            },
            {
                "instanceType": "p4de.24xlarge",
                "instanceCount": 1,
                "rootVolume": {
                    "size": 1024
                },
                "autoscaling": {
                    "minInstances": 0,
```

```
                    "maxInstances": 5,
                    "enabled": true
                }
            }
        ],
        "environmentCrn": "[***CDP-ENVIRONMENT-CRN***]",
        "tags": [
            {
                "key": "experience",
                "value": "cml-serving"
            }
        ]
    },
    "usePublicLoadBalancer": true,
    "skipValidation": false,
    "loadBalancerIPWhitelists": [
        ""
    ],
    "subnetsForLoadBalancers": [
        ""
    ],
    "staticSubdomain": "mydomain"
 }
```

Azure JSON file example

```
{
    "appName": "my-azure-caii-cluster",
    "environmentCrn": "[***CDP-ENVIRONMENT-CRN***]",
    "clusterCrn": “[***COMPUTE-CLUSTER-CRN***]”,
    "provisionK8sRequest": {
        "instanceGroups": [
            {
                "instanceType": "Standard_D4s_v3",
                "instanceCount": 1,
                "rootVolume": {
                    "size": 256
                },
                "autoscaling": {
                    "minInstances": 0,
                    "maxInstances": 5,
                    "enabled": true
                }
            },
            {
                "instanceType": "Standard_ND96asr_A100_v4",
                "instanceCount": 1,
                "rootVolume": {
                    "size": 1024
                },
                "autoscaling": {
                    "minInstances": 0,
                    "maxInstances": 5,
                    "enabled": true
                }
            }
        ],
        "environmentCrn": "[***CDP-ENVIRONMENT-CRN***]",
        "tags": [
            {
                "key": "experience",
                "value": "cml-serving"
            }
```

```
        ]
    },
    "usePublicLoadBalancer": true,
    "skipValidation": false,
    "loadBalancerIPWhitelists": [
        ""
    ],
    "subnetsForLoadBalancers": [
        ""
    ],
    "staticSubdomain": "mydomain"
}
```

3. Use the JSON file created in the previous step to create the Cloudera AI Inference service instance:

```
$ cdp ml create-ml-serving-app --cli-input-json file:///tmp/create-servi
ng-app-input.json
```

After a successful invocation of the create command, the CRN of the Cloudera AI Inference service instance that is created is displayed. The command adds the requested compute worker node groups to the existing Kubernetes cluster specified by the clusterCrn field in the request body, and installs the necessary software components.

A typical configuration with two worker node groups would take about 15-20 minutes to complete creation.

Note: In the above example, you set the staticSubdomain parameter to mydomain. Set this parameter, if necessary, to have a custom ingress subdomain for your Cloudera AI Inference service. If you leave this value blank, a randomly generated subdomain name, such as ml-92373b63-5f1, will be created.

Listing Cloudera AI Inference service instances

You can list the Cloudera AI Inference service instances in your Cloudera Data Platform tenant with the help of a command.

Use the following command to list Cloudera AI Inference service instances in your Cloudera Data Platform tenant, across all environments:

```
$ cdp ml list-ml-serving-apps
```

A Cloudera AI Inference service instance that has been created successfully shall show the status as installation:fin ished similar to the following:

```
{
            "cloudPlatform": "AZURE",
            "appName": "serving-multi-gpu-az",
            "appCrn": "[***APPLICATION-CRN***]",
            "environmentCrn": "[***ENVIRONMENT-CRN***]",
            "environmentName": "eng-ml-env-azure-v2",
            "ownerEmail": "admin@example.com",
            "mlServingVersion": "1.2.0-b72",
            "isPrivateCluster": false,
            "creationDate": "2024-08-01T16:36:32.811000+00:00",
            "cluster": {
                "clusterName": "ml-92373b63-5f1",
                "domainName": "ml-92373b63-5f1.eng-ml-e.xcu2-8y8x.dev.cldr.
work",
                "liftieID": "liftie-wqq856vz",
                "isPublic": false,
                "ipAllowlist": "0.0.0.0/0",
                "authorizedIpRangesAllowList": false,
                "tags": [],
                "instanceGroups": [],
                "clusterCrn": "[***CLUSTER-CRN***]"
            },
```

```
            "status": "installation:finished",
            "usePublicLoadBalancer": false,
            "httpsEnabled": true,
            "subnetsForLoadBalancers": []
}
```

Describing Cloudera AI Inference service instance

The describe command shows you all the detailed configuration information about a specific Cloudera AI Inference service, including which node groups are currently configured.

You can describe a Cloudera AI Inference service instance using the following command:

```
$ cdp ml describe-ml-serving-app --app-crn [***APPLICATION-CRN***]
```

The following is a sample output from the above command, edited for brevity:

```
{
    "app": {
        "cloudPlatform": "AZURE",
        "appName": "serving-multi-gpu-az",
        "appCrn": "[***APPLICATION-CRN***]",
        "environmentCrn": "[***ENVIRONMENT_CRN***]",
        "environmentName": "eng-ml-int-env-azure-v2",
        "ownerEmail": "admin@example.com",
        "mlServingVersion": "1.2.0-b69",
        "isPrivateCluster": false,
        "creationDate": "2024-09-03T18:57:16.288000+00:00",
        "cluster": {
            "clusterName": "ml-b0504a3f-bbc",
            "domainName": "ml-b0504a3f-bbc.eng-ml-i.svbr-nqvp.int.cldr.wor
k",
            "liftieID": "liftie-vjyrtsfn",
            "isPublic": false,
            "ipAllowlist": "0.0.0.0/0",
            "authorizedIpRangesAllowList": false,
            "tags": [],
            "instanceGroups": [
             ...
            ],
            "clusterCrn": "crn:[***CLUSTER-CRN***]"
        },
        "status": "installation:finished",
        "usePublicLoadBalancer": false,
        "httpsEnabled": true,
        "subnetsForLoadBalancers": []
    }
}
```

Managing Node Groups

You can add or delete node groups to your cluster.

Adding a Node Group to an existing instance

You can add one or more node groups to your cluster if you do not have the right worker node hardware (for example, incorrect GPU models for a new workload) in your existing cluster configuration.

Procedure

1. Generate the JSON skeleton to create the instance:

```
$ cdp ml add-instance-groups-ml-serving-app --generate-cli-skeleton > /t
mp/add-instance-group.json
```

2. Edit the file to add the node group details:

```
{
    "appCrn": "[***APPLICATION-CRN***]",
    "instanceGroups": [
        {
            "instanceType": "g5.8xlarge",
            "instanceCount": 0,
            "rootVolume": {
                "size": 1024
            },
            "autoscaling": {
                "minInstances": 0,
                "maxInstances": 4,
                "enabled": true
            }
        },
        {
            "instanceType": "p3.8xlarge",
            "instanceCount": 0,
            "rootVolume": {
                "size": 1024
            },
            "autoscaling": {
                "minInstances": 0,
                "maxInstances": 4,
                "enabled": true
            }
        }
    ]
}
```

3. Use the JSON file created in the previous step to add the node group:

```
$ cdp ml add-instance-groups-ml-serving-app --cli-input-json file:///tmp/
add-instance-group.json
```

Modifying a node group in an existing instance

Follow the instructions to reconfigure an existing node group.

You can reconfigure an existing node group to change its autoscaling range, you can use the following command:

```
$ cdp ml modify-ml-serving-app --app-crn [***APPLICATION-CRN***] --instance-
group-name [***GROUP-NAME***] --min [***MINIMUM-NODE***] --max [***MAXIMUM-N
ODE***] --instance-type [***INSTANCE-TYPE***]
```

Consider the following example:

```
$ cdp ml modify-ml-serving-app --app-crn <your-application-crn> --instance-g
roup-name mlgpu1 --min 2 --max 4 --instance-type g5.24xlarge
```

The instance group name can be obtained using the `describe-ml-serving-app` command.

Related Information

Describing Cloudera AI Inference service instance

Deleting a node group from an existing Cloudera AI Inference service instance

Consider the guidelines here to delete a node group from an existing Cloudera AI Inference service instance.

Use the following command to delete a worker node group:

```
$ cdp ml delete-instance-group-ml-serving-app
```

Consider the following example:

```
$ cdp ml delete-instance-group-ml-serving-app --app-crn <YOUR-APP-CRN> --ins
tance-group-name mlgpu2
```

Related Information

Describing Cloudera AI Inference service instance

Deleting Cloudera AI Inference service instance

Consider the following instructions for deleting Cloudera AI Inference service instance.

You can delete a Cloudera AI Inference service instance if it is no longer needed:

```
$ cdp ml delete-ml-serving-app --app-crn <app-crn>
```

Note: The above command only deletes the Cloudera AI Inference service and its associated cloud resources, such as cluster node groups and load balancers. The underlying compute cluster will not be deleted. If you want to delete the compute cluster itself once Cloudera AI Inference service has been deleted, you can do it from the **Compute Clusters** tab in the environment UI.

Caution: Always delete Cloudera AI Inference service first before deleting the underlying compute cluster.

Obtaining Control Plane Audit Logs for Cloudera AI Inference service

Consider the following instructions for obtaining Control plane audit logs for Cloudera AI Inference service.

Use the following commands to obtain audit log entries and the logs therein for troubleshooting purposes:

```
cdp ml get-audit-events --resource-crn [***APPLICATION-CRN***]
cdp ml get-logs --request-id [***REQUEST-ID***] --resource-crn [***APPLIC
ATION-CRN***]
```

Obtaining the kubeconfig for the Cloudera AI Inference service Cluster

Troubleshoot Kubernetes-level issues in the cluster.

To troubleshoot Kubernetes-level issues in the cluster, such as installation or upgrade failure, or any other cluster level issue, you must first obtain the kubeconfig of the cluster.

Related Information

Granting Remote Access to ML Workbenches

Obtaining kubeconfig on AWS

Follow the guidelines for obtaining kubeconfig on AWS.

About this task

You need the user's Amazon Resource Name (ARN) to obtain the kubeconfig file. You can get the ARN from the user or look up a user's ARN in your AWS account. To obtain a user's ARN from the AWS account, complete the following steps:

Procedure

1. Obtain the Amazon Resource Name (ARN) .
   a) Using AWS UI, go to your organization's AWS Account Identity and Access Management (IAM) Users and look up the user. The ARN is available on the **Summary** page.
   b) Using AWS CLI, run the following command to get the ARN:

```
$ aws sts get-caller-identity
```

   Consider the following example:

```
# Sample output
{
"UserId": "ABCDE12345FGHIJKLMNO6789",
"Account": "888888888888",
"Arn": "arn:aws:iam::888888888888:user/joesmith"
}
```

2. Obtain your Cloudera AI Inference service CRN, run the following command:

```
$ cdp ml list-ml-serving-apps
```

3. Grant access to the Cloudera AI Inference service AWS cluster, run the following command:

```
$ cdp ml grant-ml-serving-app-access \
  --resource-crn [***APPLICATION-CRN***] \
  --identifier [***AWS_ARN***]
```

4. Obtain the kubeconfig used to access the above cluster by running:

```
$ cdp ml get-ml-serving-app-kubeconfig \
  --app-crn [***APPLICATION-CRN***] \
  | jq '.kubeconfig' | yq > myconfig.yaml
```

5. Send the downloaded Kubernetes configuration file to the user who has been granted access. To connect to the Amazon EKS cluster, you must have aws-iam-authenticator installed.
6. Get the logs of the Cloudera AI Inference service API server:

```
$ KUBECONFIG=myconfig.yaml kubectl get logs deployment/api -n cml-serving
```

Obtaining kubeconfig on Azure

Consider the following guidelines for obtaining kubeconfig on Azure.

Procedure

Locate the liftie cluster identifier for your Cloudera AI Inference service by issuing a describe command using Cloudera Data Platform CLI:

```
$ cdp ml describe-ml-serving-app --app-crn [***APPLICATION-NAME***]
```

```
{
    "app": {
        "appName": "[***APPLICATION-NAME***]",
        "appCrn": "[***APPLICATION-CRN***]",
        "environmentCrn": "[***ENVIRONMENT-CRN***]",
        "environmentName": "[***ENVIRONMENT-NAME***]",
        "ownerEmail": "user@org.com",
        "mlServingVersion": "1.2.0-b72",
        "isPrivateCluster": false,
```

```
        "creationDate": "2024-05-07T14:27:16.817000+00:00",
        "cluster": {
            "clusterName": "ml-1fcaa8cf-a94",
            "domainName": "[***DOMAIN-NAME***]"
            "liftieID": "liftie-3544nrdr",
            "isPublic": false,
            "ipAllowlist": "0.0.0.0/0",
            "authorizedIpRangesAllowList": false
        },
        "status": "installation:finished",
        "usePublicLoadBalancer": false,
        "httpsEnabled": true
    }
}
```

The cluster identifier is in app.cluster.liftieID.

Locate this entry in the Azure Portal's Kubernetes services page and do the following:

a) Open the entry.
b) Click on **Connect**.
c) Copy the command under **Download cluster credentials** to your terminal and execute.

   You are now authenticated and your kubectl context is set up to interact with the AKS cluster.

Managing Cloudera AI Inference service using the UI

You can manage the life cycle of the Cloudera AI Inference service and associated infrastructure using the UI.

Creating a Cloudera AI Inference service instance using the UI

You can create a Cloudera AI Inference service instance using the UI.

Before you begin

Cloudera AI Inference service requires a compute cluster enabled Cloudera environment. You must either create the compute cluster enabled environment or convert your existing environment. For more information, see *Prerequisites for setting up Cloudera AI Inference service.*

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click AI Inference Services under ADMINISTRATION on the left navigation menu.

   The AI Inference Services page is displayed.

3. Click the Create AI Inference Service button.

   The Create AI Inference Service page is displayed.

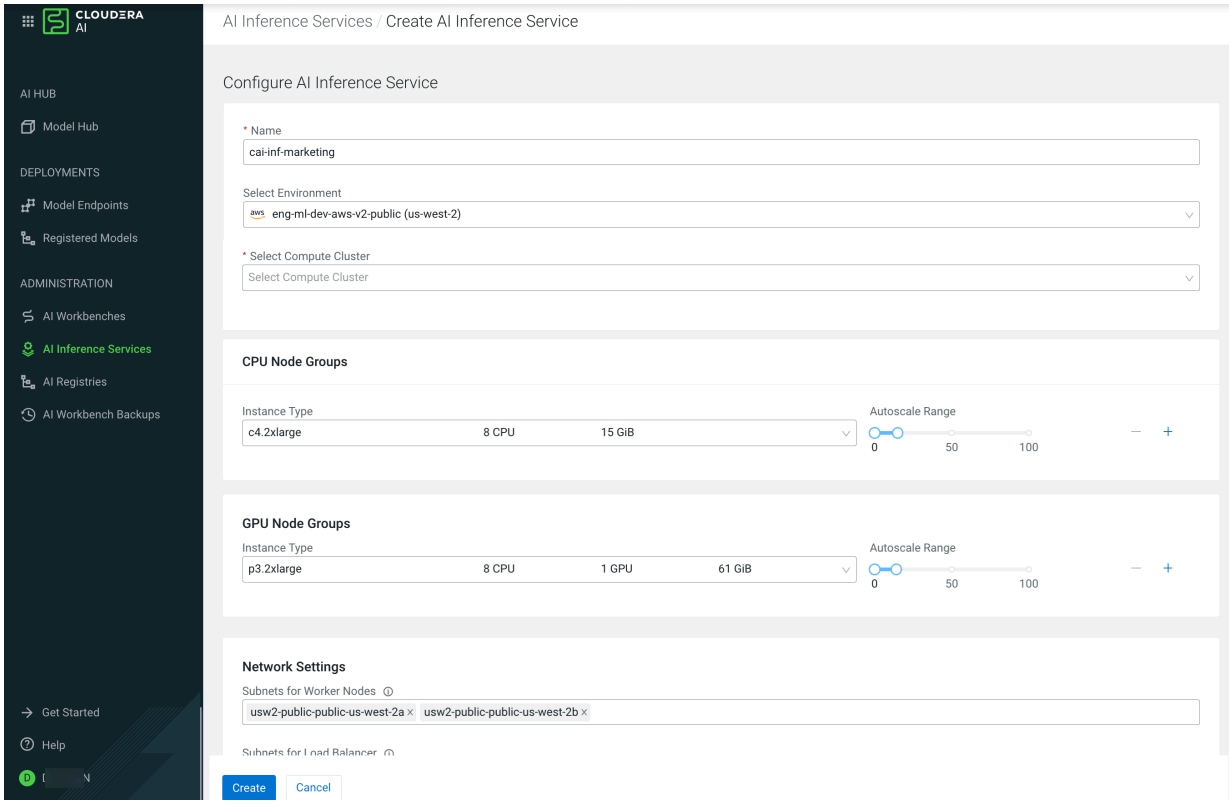

4. In the Name textbox, enter a name for the Cloudera AI Inference service instance.
5. In the Select Environment dropdown list, select your Cloudera environment within which you want to create the service.
6. In the Select Compute Cluster dropdown list, select the compute cluster.
7. In CPU Node Groups, select the CPU instance type and autoscale range. You can add more than one CPU node group by clicking the + button.
8. In GPU Node Groups, select the GPU instance type and autoscale range. You can add more than one GPU node group by clicking the + button.
9. In Network Settings, select the subnets for worker nodes and load balancer. You can select multiple subnets.
10. In Load Balancer Source Ranges, specify the IP ranges that are allowed to access the load balancer.
11. Select the Enable Public IP Address for Load Balancer checkbox to make Cloudera AI Inference service available on the public internet. When disabled, it is assumed that connectivity is achieved through a corporate Virtual Private Cloud (VPC).
12. In Static Subdomain, specify the static domain for the AI Inference Service.
13. Select the Skip Validation checkbox if you do not want to perform the validation checks before provisioning this AI Inference Service.
14. In Tags, add any custom key and value pairs for your own use.
15. Click Create.

Related Information

Installing Cloudera client

Using Compute Clusters in AWS environment

Using Compute Clusters in Azure environment

Setting up AI Registry

Listing Cloudera AI Inference service instances using the UI

You can view the list of all the Cloudera AI Inference service instances in your Cloudera Data Platform tenant. It provides details like the name of the Cloudera AI Inference service, status, associated environment name, created date and time, and the cloud provider details.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click AI Inference Services under ADMINISTRATION on the left navigation menu.

   The AI Inference Services page is displayed.
3. You can use the filter bar at the top of the window to filter the list of Cloudera AI Inference service instances by service name and environment name.
4. Select a Cloudera AI Inference service instance to see its description.

Viewing details of a Cloudera AI Inference service instances using the UI

You can view detailed configuration information about a specific Cloudera AI Inference service instance, including which node groups are currently configured.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click AI Inference Services under ADMINISTRATION on the left navigation menu.

   The AI Inference Services page is displayed.
3. Select a Cloudera AI Inference service instance to see its description.
4. You can view the configuration information like CRN, Environment CRN, Version, node configuration, events and logs, and so on.

Managing node groups using the UI

You can add, modify, or delete node groups to or from your Cloudera AI Inference service instance.

Adding a node group to an existing instance using the UI

You can add one or more node groups to your cluster if you do not have the right worker node hardware (for example, incorrect GPU models for a new workload) in your existing cluster configuration.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click AI Inference Services under ADMINISTRATION on the left navigation menu.

   The AI Inference Services page is displayed.
3. Select a Cloudera AI Inference service instance to which you want to add a node group.
4. Click Add CPU Node Group to add CPU nodes and select the instance type. Click Add GPU Node Group to add GPU nodes and select the instance type.

5. In the Autoscale Range Min - Max textbox, specify the autoscale range.

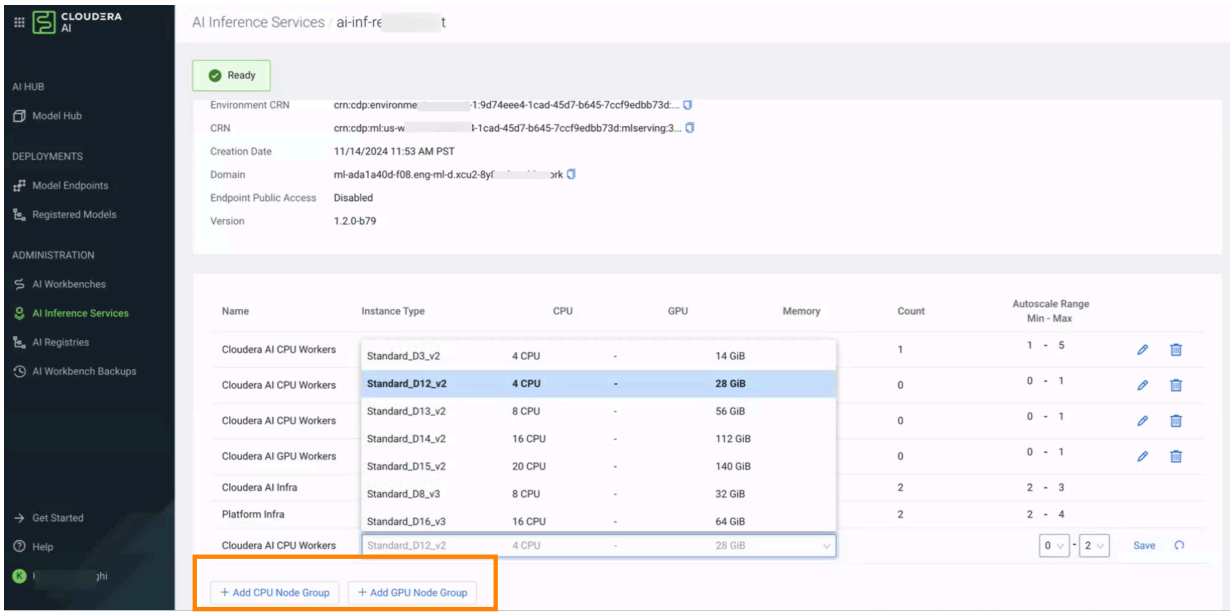

6. Click Save.

Modifying a node group in an existing instance using the UI

You can reconfigure an existing node group including the autoscaling range of an AI Inference service instance.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click AI Inference Services under ADMINISTRATION on the left navigation menu.

   The AI Inference Services page is displayed.
3. Select a Cloudera AI Inference service instance to which you want to modify a node group..
4. Click in the node group you want to modify. You can modify the instance type and the autoscale range.
5. Click Save.

Deleting a node group from an existing Cloudera AI Inference service instance using the UI

You can delete a node group from an existing Cloudera AI Inference service instance.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click AI Inference Services under ADMINISTRATION on the left navigation menu.

   The AI Inference Services page is displayed.
3. Select a Cloudera AI Inference service instance in which you want to delete a node group..
4. Click in the node group you want to delete. A confirmation message will be displayed.
5. Click OK to delete the node group.

Deleting Cloudera AI Inference service instances using the UI

You can delete a Cloudera AI Inference service instance if it is no longer needed..

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click AI Inference Services under ADMINISTRATION on the left navigation menu.

   The AI Inference Services page is displayed.
3. From the Actions menu, click Delete. A confirmation message will be displayed.
4. Click OK to delete the instance.

Obtaining Control Plane Audit Logs for Cloudera AI Inference service using the UI

You can obtain Control plane audit logs for Cloudera AI Inference service. You can use the audit log entries and the logs for troubleshooting purposes.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click AI Inference Services under ADMINISTRATION on the left navigation menu.

   The AI Inference Services page is displayed.
3. Select a Cloudera AI Inference service instance for which you want to obtain the event logs.
4. Click Events & Logs to view the event history.

   The **Events History** page is displayed.
5. In the **Events History** page, click View Event Logs to view the event history of the specific event. The log of the specific event is displayed.
6. Click Download Logs to download the logs of that specific event.

Obtaining the kubeconfig of Cloudera AI Inference service using the UI

To troubleshoot Kubernetes-level issues in the cluster, such as installation or upgrade failure, or any other cluster level issue, you must first obtain the kubeconfig of the cluster.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click AI Inference Services under ADMINISTRATION on the left navigation menu.

   The AI Inference Services page is displayed.
3. From the Actions menu of the Cloudera AI Inference service instance, click Download Kubeconfig.
   The *[***INSTANCE-NAME****]*-kubeconfig.yaml file will be downloaded to your local machine.

Managing Model Endpoints using UI

Cloudera AI Inference service lets you deploy models saved in the Cloudera AI Registry.

Cloudera AI Inference service provides a graphical user interface accessible from the Cloudera AI Control Plane for managing model endpoints. HTTP/REST APIs are also available for programmatic model endpoint management.

Cloudera AI Inference service UI provides new features to view, create, and edit model endpoints from the control plane UI. Each Cloudera AI Inference service instance can have multiple deployed model endpoints. The Model Endpoints page provides a comprehensive list view of all the deployed model endpoints in a given Cloudera Data Platform account.

Related Information

Supported Model Artifact Formats

Creating a Model Endpoint using UI

The **Create Model Endpoint** page allows you to select a specific Cloudera AI Inference service instance and a model version from Cloudera AI Registry to create a new model endpoint.

About this task

The following steps illustrate how to create a Llama 3.1 model endpoint.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click Model Endpoints under Deployments on the left navigation menu.

   The **Model Endpoints** landing page is displayed.

3. Click the Create Endpoint button.

   The **Create Endpoint** page displays the details to create an endpoint.

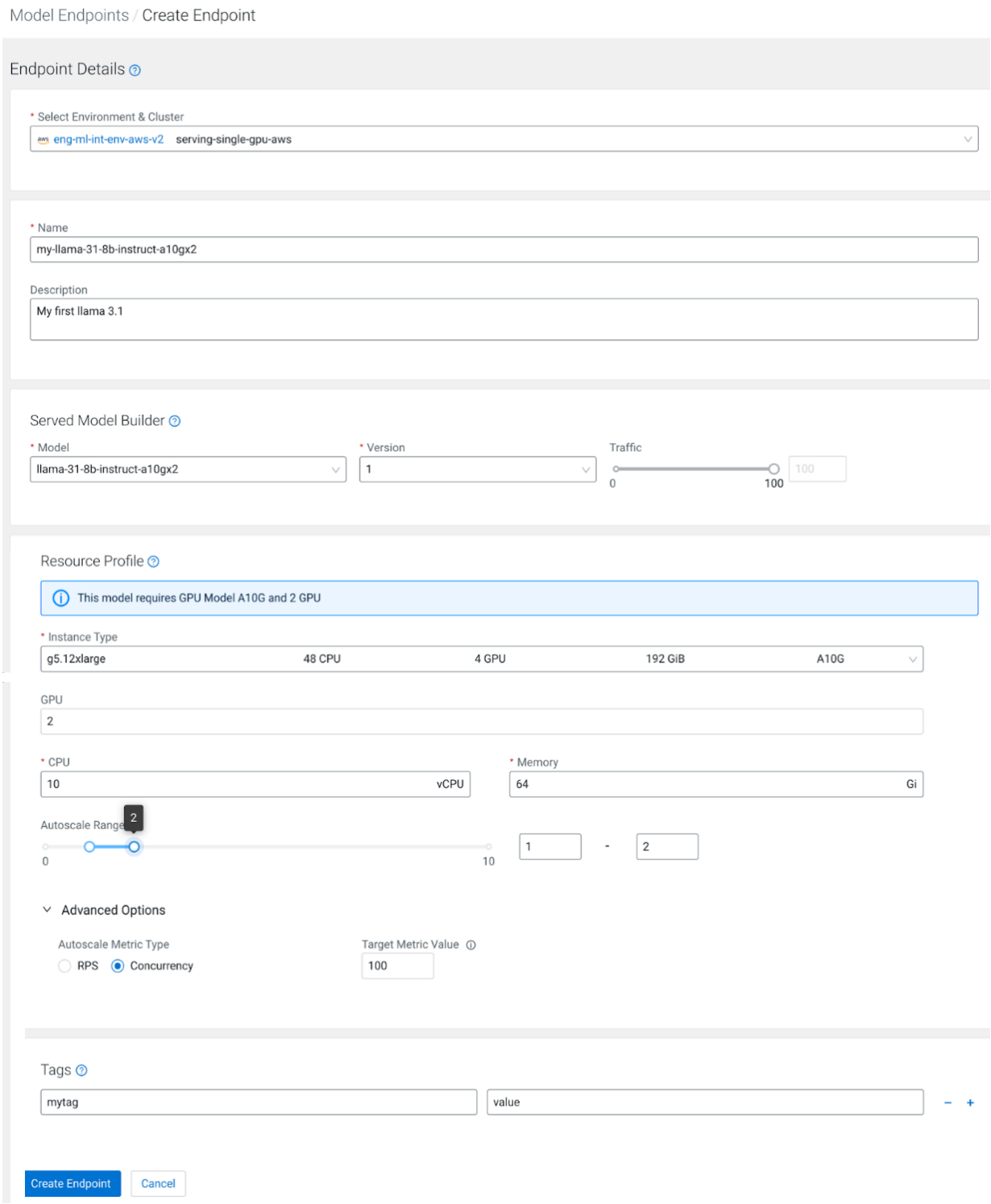

4. In the Select Environment & Cluster dropdown list, select your Cloudera Data Platform environment and the Cloudera AI Inference service instance within which you want to create the model endpoint.
5. In the Name textbox, enter a name for the model endpoint.
6. In the Description textbox add a short description of the model endpoint. Ensure that the description text you enter is below 5000 characters.

7. Under Served Model Builder, select the model (Model), and modelversion (Version) from the dropdown you want to deploy. Using the Traffic slider bar, specify the traffic split between different model versions that you deploy. It is always set to 100% for the first model version which you cannot change.
8. Under Resource Profile, select the type of the instance from the Instance Type dropdown list. For *NVIDIA NIM* models this field is mandatory. Specify which compute node instance type you wish to run on your model replicas. The instance type you choose depends on the capabilities of the instance type and on what is required by the NVIDIA NIM. The field is optional for normal predictive model endpoints.
9. In the GPU field, specify the number of GPUs each model endpoint replica requires. This field gets populated automatically for NVIDIA NIM models.
10. In the CPU field, specify the number of CPUs each model endpoint replica requires.
11. In the Memory field, specify the amount of CPU memory each model endpoint replica requires.
12. Using the Autoscale Range slider bar, specify the minimum and maximum number of replicas for the model endpoint. Based on which autoscaling parameter you choose, the system scales the number of replicas to meet the incoming load.
13. Under Advanced Options, in the Autoscale Metric Type select the autoscale metric type as Requests per second (RPS) or Concurrency per replica of the model endpoint.
    If you choose to scale as per RPS and the Target Metric Value is set to 200, then the system automatically adds a new replica when it sees that a single replica is handling 200 or more requests per second. If the RPS falls below 200, then the system scales down the model endpoint by terminating a replica.
14. In Tags, add any custom key and value pairs for your own use.
15. Click Create Endpoint to create the model endpoint.

    It can take tens of minutes for the model endpoint to be ready. The time taken is determined by the following points:

    - whether a new node has to be scaled-in to the cluster.
    - the time taken to pull the necessary container images.
    - the time taken to download model artifacts to the cluster nodes from the Cloudera AI Registry.

    In the above example, we are creating a model endpoint for the instruct variant of Llama 3.1 8B, which has been optimized to run on two NVIDIA A10G GPUs per replica. The cluster we are deploying into is in AWS, and it has a node group consisting of g5.12xlarge instance types. In this example there are no other node groups that have instance types containing A10G GPUs, so the g5.12xlarge instance type is the only choice. For NVIDIA NIM models that specify GPU models and count, the GPU field of the resource configuration page will be filled in automatically by the UI.

    Important: We set the autoscale range as 1-2, with the scaling metric set to concurrency. The scale metric target is set to 100, meaning that if the number of concurrent requests per replica exceeds 100, the system will automatically add a new replica, up to a maximum of two replicas. You must ensure that your cluster has sufficient capacity to expand the requisite node group to run the desired number of replicas.

Listing Model Endpoints using UI

The **Model Endpoints** landing page displays a table view that represents a comprehensive list of all the model endpoints across the running Cloudera AI Inference service instances in your Cloudera Data Platform account.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.

2. Click Model Endpoints under Deployments on the left navigation menu.

   The **Model Endpoints** landing page is displayed.

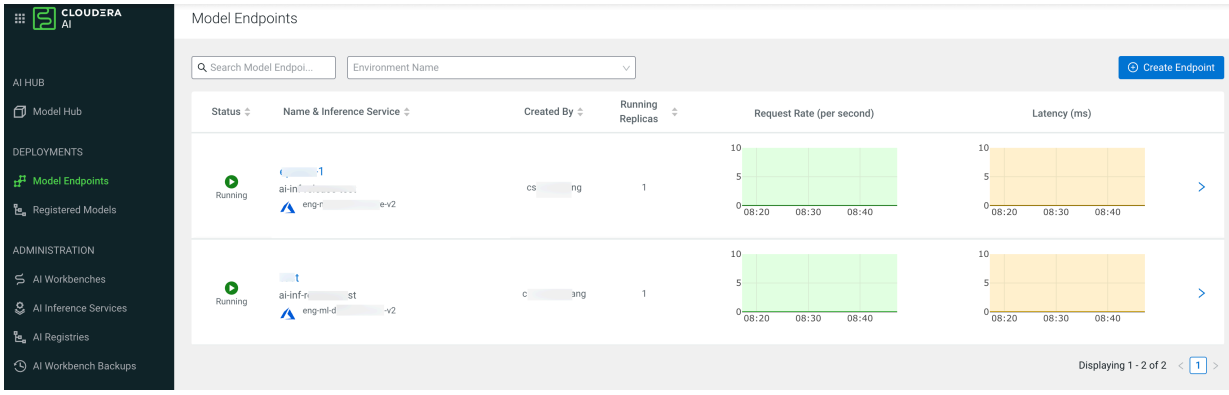

Viewing details of a Model Endpoint using UI

You can view the information of a Model Endpoint, edit its configuration, view metric charts on resource utilization, and so on.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click Model Endpoints under Deployments on the left navigation menu.

   The **Model Endpoints** landing page is displayed.
3. Select a model endpoint to see its details.

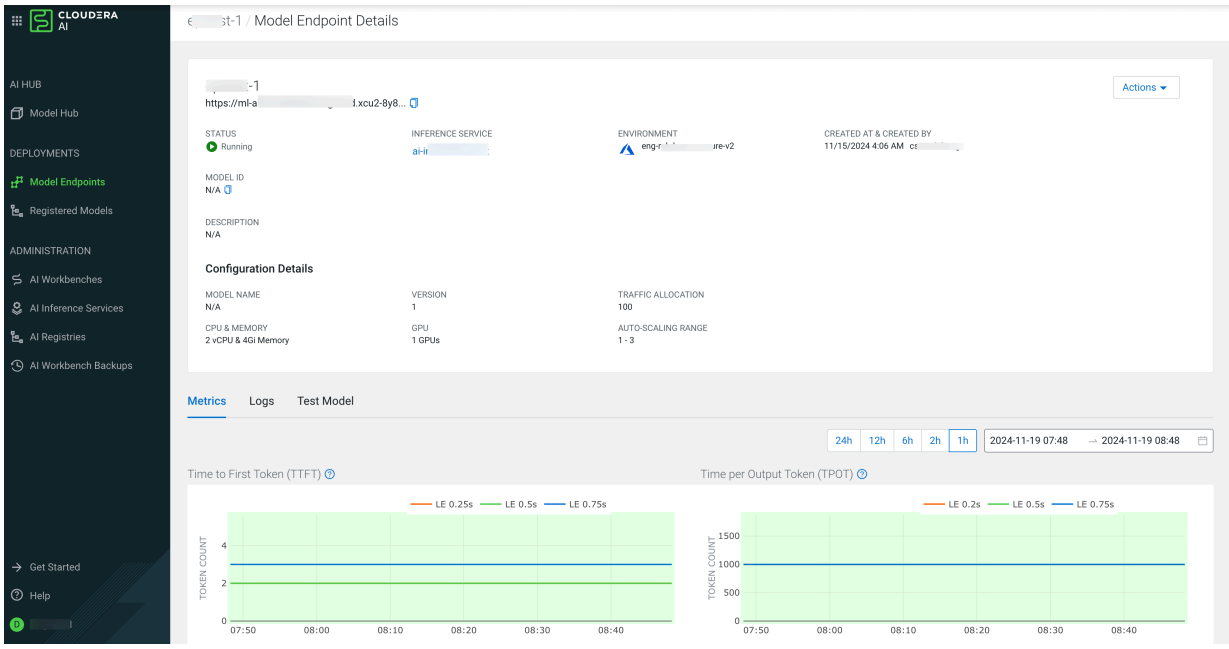

Editing a Model Endpoint Configuration using UI

You can edit the configuration of a specific model endpoint for resources or select model versions.

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click Model Endpoints under Deployments on the left navigation menu.

   The **Model Endpoints** landing page is displayed.
3. Select a model endpoint that you want to edit.
4. Select Edit Configuration from the Actions dropdown list.
5. Edit the resource profile or add model versions.
   a. In the Resource Profile tab, select the instance type, and specify the CPU, Memory, GPU, and the autoscale range.
   b. In the Served Models tab, add new model versions and specify the traffic percentage for the model versions.

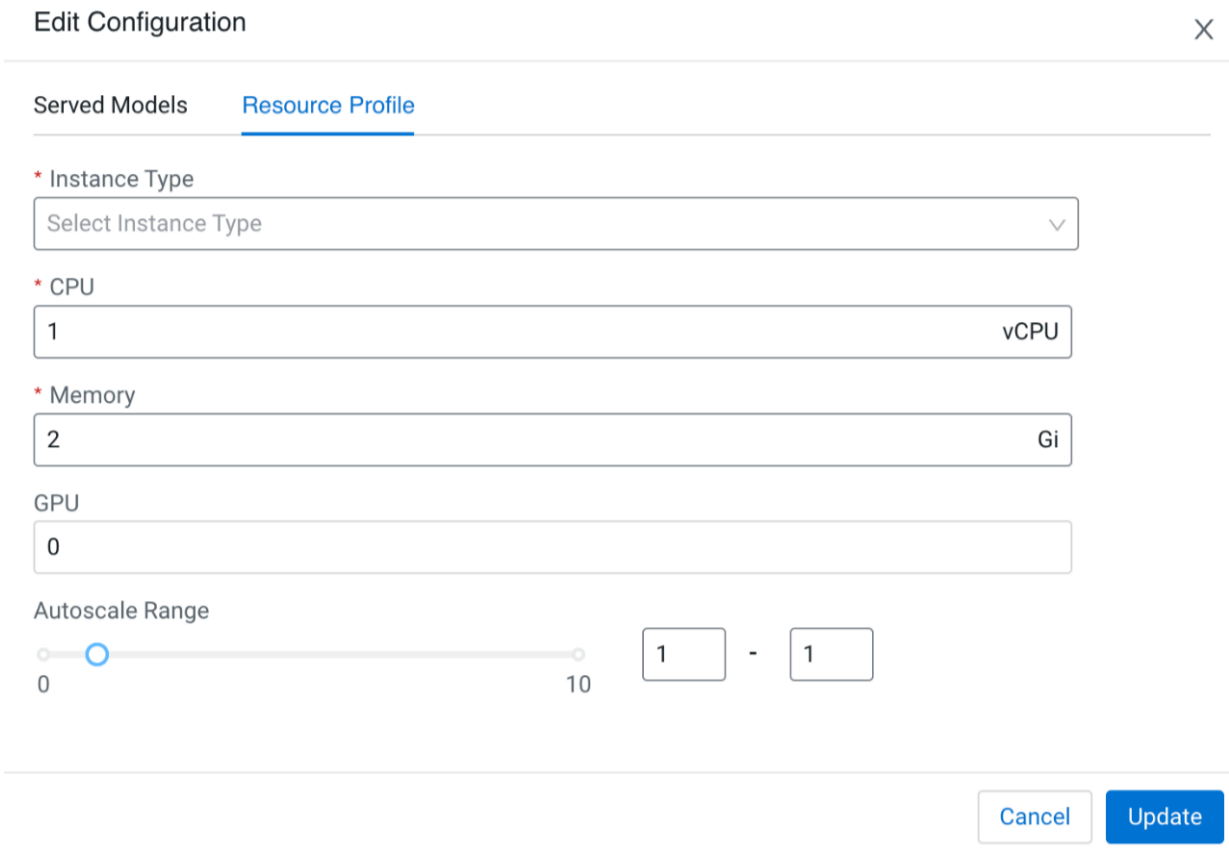

6. Click Update.

Managing Model Endpoints using API

You can use API to create, view, list, edit, describe, and delete model endpoints.

Related Information

Curl documentation

Preparing to interact with the Cloudera AI Inference service API

To interact with Cloudera AI Inference service API, you need to obtain the domain name of the Cloudera AI Inference service and your Cloudera Data Platform JSON Web Token (JWT) and save it as environment variables.

About this task

Important: The below commands use the `jq` command, which must be installed on your system.

Procedure

1. Obtain the domain name of the Cloudera AI Inference service and Cloudera Data Platform JSON Web Token (JWT) from the output of `list-ml-serving-apps` or `describe-ml-serving-app` Cloudera Data Platform commands.

```
export DOMAIN=$(cdp ml describe-ml-serving-app --app-crn [***APP CRN***] |
 jq -r '.app.cluster.domainName')
```

2. Store your CDP_TOKEN as an environment variable:

```
export CDP_TOKEN=$(cdp iam generate-workload-auth-token --workload-name DE
 | jq -r '.token')
```

Creating a Model Endpoint using API

You can select a specific Cloudera AI Inference service instance and a model version from Cloudera AI Registry to create a new model endpoint.

Procedure

1. Retrieve a registered model's model ID and model version.

   This information is available in the Registered Models page in the Cloudera AI control plane UI.

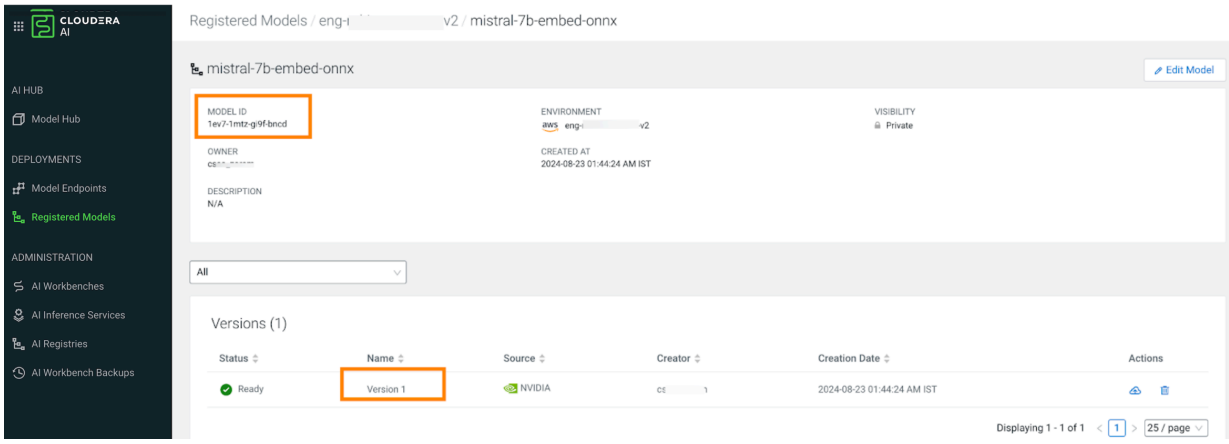

2. Create the model specification for the selected model.

```
# cat ./examples/mlflow/model-spec-cml-registry.json
```

   The following is the sample output:

```
{
```

```
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 2,
      "model_id": "3azn-tmqe-wsze-5u4s"
    }
  }
}
```

```
export DOMAIN=$(cdp ml describe-ml-serving-app --app-crn [***app-crn***] |
 jq -r '.app.cluster.domainName')
```

3. Create the model endpoint by using the following Cloudera AI serving `deployEndpoint` API:

```
curl -v  -H "Content-Type: application/json" -H "Authorization: Bearer $
{CDP_TOKEN}" "https://${DOMAIN}/api/v1alpha1/deployEndpoint" -d @./examp
les/mlflow/model-spec-cml-registry.json
```

The DOMAIN looks like ml-67814ad5-b79.eng-ml-d.xcu2-8y8x.dev.cldr.work.

You can retrieve the CDP_TOKEN by performing the steps from Preparing to interact with the Cloudera AI.

Note:

You can only specify serving-default as the namespace into which the Model Endpoint can be deployed.

Related Information

Register an ONNX model to AI Registry

Preparing to interact with the Cloudera AI Inference service API

Listing Model Endpoints using API

Consider the following details for listing Model Endpoints using API.

Procedure

List model endpoints with the help of the following command:

```
curl -H "Content-Type: application/json" -H "Authorization: Bearer ${CDP_TOK
EN}" "https://${DOMAIN}/api/v1alpha1/listEndpoints" -d '{
  "namespace": "serving-default"
}'

{"endpoints":[
  {
    "namespace":"serving-default",
    "name":"mlflow-wine-test-from-registry-onnx",
    "url":"https://[***domain***]/namespaces/serving-default/endpoints/[*
**endpoint_name***]/v2/models/[***model_name***]/infer",
    "state":"Loaded"
  }
]}%
```

Describing a Model Endpoint using API

Consider the instructions for describing a Model Endpoint using API.

Procedure

Use the following command to describe a given model endpoint. The `describeEndpoint` API shows you all the detailed configuration and runtime snapshot of the endpoint. The output below has been elided for brevity.

```
curl -s -H "Authorization: Bearer ${CDP_TOKEN}" -H "Content-Type: applicatio
n/json" https://${DOMAIN}/api/v1alpha1/describeEndpoint -d '{"namespace": "s
erving-default", "name": "[***ENDPOINT_NAME***]"}'|jq
```

You will receive a response similar to the following:

```
{
  "namespace": "serving-default",
  "name": "[***ENDPOINT_NAME***]",
  "url": "https://[***DOMAIN***]/namespaces/serving-default/endpoints/[***EN
DPOINT_NAME***]/v2/models/[***MODEL_ID***]/infer",
  "conditions": [
    {
      "type": "IngressReady",
      "status": "True",
      "severity": "",
      "last_transition_time": "1726713090",
      "reason": "",
      "message": ""
    },

  "observed_generation": 1,
  "replica_count": 1,

  ...
  "autoscaling": {
    "min_replicas": "1",
    "max_replicas": "80",
    "autoscalingconfig": {
      "metric": "concurrency",
      "target": "20",
    }
  },

 ...

  "api_standard": "oip",
  "has_chat_template": false,
  "metricFormat": "triton",
  "task": "inference"
}
```

Related Information

Nvidia documentation: Inputs and outputs

Preparing to interact with the Cloudera AI Inference service API

Deleting a Model Endpoint using API

Consider the following instruction to delete a Model Endpoint using API.

Procedure

Delete the Model Endpoint deployed above, using the API.

```
curl -v  -H "Content-Type: application/json" -H "Authorization:
Bearer ${CDP_TOKEN}"
"https://${DOMAIN}/api/v1alpha1/deleteEndpoint" -d
'{"namespace": "serving-default", "name": "[***ENDPOINT_NAME***]"}'
```

Autoscaling Model Endpoints using API

You can configure the Model Endpoints deployed on Cloudera AI Inference service to auto-scale to zero instances when there is no load.

Procedure

The following deployment specification shows an example model endpoint that auto-scales between zero and four replicas:

```
# cat ./examples/mlflow/model-spec-cml-registry.json
```

```
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 1,
      "model_id": "yf0o-hrxq-l0xj-8tk9"
    }
  },
  "autoscaling": {
    "min_replicas": "0",
    "max_replicas": "4"
  }
}
```

Tuning auto-scaling sensitivity using the API

To customize the autos-caling sensitivity and requisites, set the target and metric fields in the autoscaling.autoscalingconfig parameter.

About this task

The metric field is the data you are watching to make auto-scaling decisions, which you can set to either concurrency or to RPS (requests per second):

- Concurrency is the number of requests that each replica of the model shall aim to handle at once.
- RPS is the calculated requests per second handled over the polling period.
- Target is the target value of the metric that you aim to maintain. If the Target value is exceeded when calculating the metric value, new replicas will be spun up or down to maintain the Target value as the upper bound.

Procedure

Set these values as follows:

```
# cat ./examples/mlflow/model-spec-cml-registry.json
```

```
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 1,
      "model_id": "yf0o-hrxq-l0xj-8tk9"
    }
  },
  "autoscaling": {
    "min_replicas": "0",
    "max_replicas": "4",
    "autoscalingconfig": {
        "metric": "concurrency",
        "target": "25"
    }
  }
}
```

The above example scales between zero and four replicas aiming to maintain at most 25 concurrent requests per replica, scaling the number of replicas deployed to maintain this target.

Running Models on GPU

Follow the guidelines for running Models on GPU.

Procedure

To run a Model Endpoint on GPU, specify the number of GPUs to use per model replica, as follows. You must specify the GPU count as a *string*.

```
# cat ./examples/mlflow/model-spec-cml-registry.json
```

```
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 1,
      "model_id": "yf0o-hrxq-l0xj-8tk9"
    }
  },
  "resources": {
    "num_gpus": "1"
  }
}
```

The resources field supports specifying the following properties:

- req_cpu - the requested number of CPU cores, which is a string as per Kubernetes standards.
- req_memory - the requested amount of memory, which again follows Kubernetes conventions.
- num_gpus - the requested number of GPUs.

Related Information

Resource Units in Kubernetes - Kubernetes documentation
Memory Resource Units - Kubernetes documentation

Deploying models with Canary deployment using API

Cloudera AI Inference service allows users to control traffic percentage to specific model deployments.

Procedure

1. Deploy a model. The traffic value defaults to 100. The following is an example payload:

```
# cat ./examples/mlflow/model-spec-cml-registry.json
```

```
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 1,
      "model_id": "yf0o-hrxq-l0xj-8tk9"
    }
  }
}
```

2. After deploying a model to the endpoint, direct the traffic to the *Canary* model by updating the model with the desired traffic value set:

```
# cat ./examples/mlflow/model-spec-cml-registry-canary.json
```

```
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 2,
      "model_id": "yf0o-hrxq-l0xj-8tk9"
    }
  }
  "traffic": "35",
}
```

The percentage of the traffic set in the traffic field diverts to the newly deployed model version and the remaining traffic is directed to the previously deployed model version.

3. Update the model endpoint as follows, when you want your canary model to serve all of your incoming requests:

```
# cat ./examples/mlflow/model-spec-cml-registry-promote.json
```

```
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 2,
      "model_id": "yf0o-hrxq-l0xj-8tk9"
    }
  }
  "traffic": "100",
```

```
}
```

4. Use the following payload to content to the model:

```
curl -H "Content-Type: application/json" -H
"Authorization: Bearer ${CDP_TOKEN}"
https://${DOMAIN}/namespaces/serving-default/endpoints/mlflow-wine-test-f
rom-registry-onnx/v2/models/yf0o-hrxq-l0xj-8tk9/infer
-d @./examples/url/wine-input.json
{"model_name":"yf0o-hrxq-l0xj-8tk9","model_version":"2","outputs":[{"name"
:"variable","datatype":"FP32","shape":[1,1],"data":[0.0]}]}
```

Interacting with Model Endpoints

You can interact with the Cloudera AI Inference service API using an HTTP/REST client, such as cURL.

Making an inference call to a Model Endpoint with an OpenAI API

Language models for text generation are deployed using NVIDIA's NIM microservices. These model endpoints are compliant with the *OpenAI Protocol*. See NVIDIA NIM documentation for supported OpenAI APIs and NVIDIA NIM specific extensions such as Function Calling and Structured Generation.

Related Information

OpenAI Platform protocol

Nvidia documentation: API Reference

Cloudera AI Inference service using OpenAI Python SDK client in a Cloudera AI Workbench Session

Consider the following instructions on interacting with an instruction-tuned large language model endpoint hosted on Cloudera AI Inference service.

Procedure

You can use the following template to interact with an instruction-tuned large language model endpoint hosted on Cloudera AI Inference service:

```
from openai import OpenAI
import json

API_KEY = json.load(open("/tmp/jwt"))["access_token"]
MODEL_NAME = "[***MODEL_NAME***]"

client = OpenAI(
  base_url = "[***BASE_URL***]",
  api_key = API_KEY,
  )

completion = client.chat.completions.create(
  model=MODEL_NAME,
  messages=[{"role":"user","content":"Write a one-sentence definition of Ge
nAI."}],
  temperature=0.2,
  top_p=0.7,
  max_tokens=1024,
  stream=True
)
for chunk in completion:
```

```
  if chunk.choices[0].delta.content is not None:
    print(chunk.choices[0].delta.content, end="")
```

Where base_url is the model endpoint URL up to the API version v1. To get the base_url, copy the model endpoint URL and delete the last two path components.

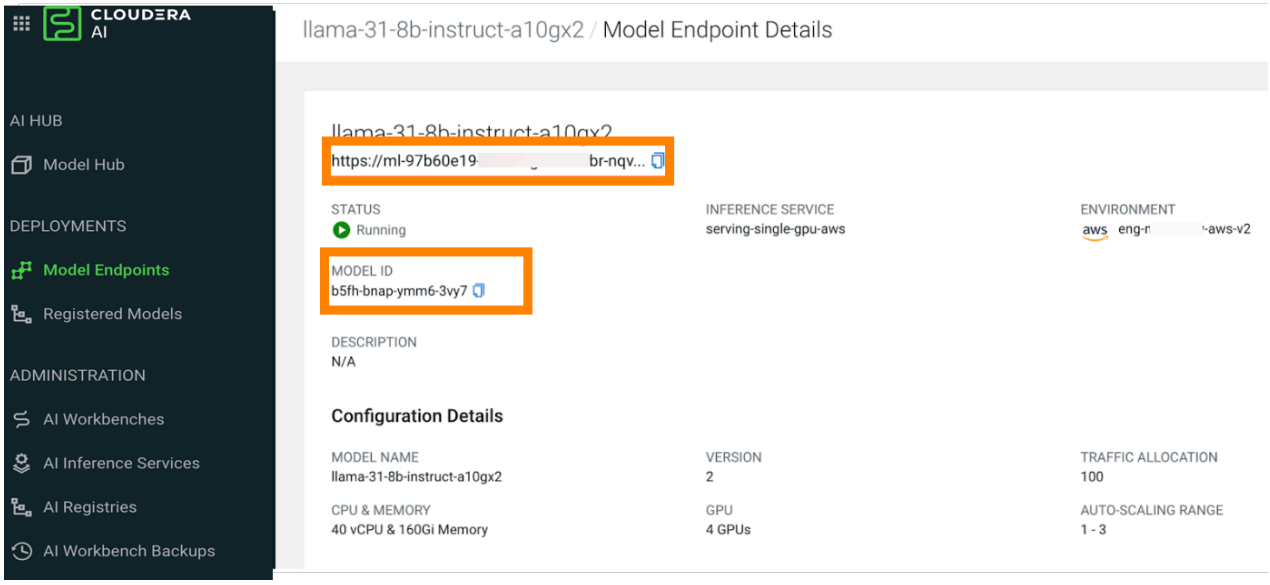

Copy the model endpoint URL from the **Model Endpoint Details** UI and modify it to

```
https://[***DOMAIN***]/namepaces/serving-default/endpo
ints/[***ENDPOINT_NAME***]/v1
```

MODEL_NAME is the model name assigned to the model when it is registered to the Cloudera AI Registry. You can find this in the **Model Endpoint Details** UI.

Cloudera AI Inference service using OpenAI Python SDK client on a local machine

Consider the following guidelines for Cloudera AI Inference service using OpenAI Python SDK client.

Procedure

Specify the CDP_TOKEN as the `API_KEY`. A common way is to export CDP_TOKEN as an environment variable and access that from your Python code:

```
from openai import OpenAI
import os
API_KEY = os.environ['CDP_TOKEN']
MODEL_NAME = [***MODEL_NAME***]

client = OpenAI(
  base_url = "[***BASE_URL***]",
  api_key = API_KEY,
  )

completion = client.chat.completions.create(
  model=MODEL_NAME,
  messages=[{"role":"user","content":"Write a one-sentence definition of Ge
nAI."}],
  temperature=0.2,
  top_p=0.7,
  max_tokens=1024,
  stream=True
```

```
)
for chunk in completion:
  if chunk.choices[0].delta.content is not None:
    print(chunk.choices[0].delta.content, end="")
```

Related Information

Preparing to interact with the Cloudera AI Inference service API

Nvidia documentation: API Reference

Authentication of Cloudera AI Inference service - External Clients Authentication

OpenAI Inference Protocol Using Curl

Consider this example for OpenAI Inference Protocol Using Curl.

An example inference payload for the OpenAI Protocol:

```
# cat ./llama-input.json
```

```
{
    "messages": [
        {
            "content": "You are a polite and respectful chatbot helping pe
ople plan a vacation.",
            "role": "system"
        },
        {
            "content": "What should I do for a 4 day vacation in Spain?",
            "role": "user"
        }
    ],
    "model": "meta/llama-3_1-8b-instruct",
    "max_tokens": 200,
    "top_p": 1,
    "n": 1,
    "stream": false,
    "stop": "\n",
    "frequency_penalty": 0.0
}
```

```
curl -H "Content-Type: application/json" -H "Authorization: Bearer ${CDP_TOK
EN}" "https://${DOMAIN}/namespaces/serving-default/endpoints/llama-3-1/v1/ch
at/completions" -d @./llama-input.json
```

You will receive response similar to the following:

```
Spain offers a diverse range of experiences, making it perfect for a 4-day v
acation…
```

Making an inference call to a Model Endpoint with Open Inference Protocol

Cloudera AI Inference service serves predictive ONNX models using the Nvidia Triton Server. The deployed model endpoints are compliant with Open Interface Protocol version 2.

Note: The model inference request and response payloads are different from models deployed on a Cloudera AI Workbench.

Related Information

Open Inference Protocol Specification

Nvidia Triton Inference Server

Open Inference Protocol Using Python SDK

Use the following code sample to interact with a model endpoint using the Open Inference Protocol.

Procedure

Run the following command to customize the ENDPOINT_NAME, DOMAIN, and MODEL_NAME placeholders. The inference request payload shape must match what your model is expecting, which you can determine with the read_model_metadata() function.

```
#!pip install open-inference-openapi

from open_inference.openapi.client import OpenInferenceClient, InferenceRe
quest
import httpx
import json
import os

#
# inspired by https://pypi.org/project/open-inference-openapi/
#

CDP_TOKEN = os.environ['CDP_TOKEN']
BASE_URL = 'https://[***DOMAIN***]/namespaces/serving-default/endpoints/[*
**ENDPOINT_NAME***]'
MODEL_NAME = '[***MODEL-NAME***]'
headers = {'Authorization': 'Bearer ' + CDP_TOKEN,
           'Content-Type': 'application/json'}

httpx_client = httpx.Client(headers=headers)
client = OpenInferenceClient(base_url=BASE_URL, httpx_client=httpx_client)

# Check that the server is live, and it has the model loaded
client.check_server_readiness()
metadata = client.read_model_metadata(MODEL_NAME)
metadata_str = json.dumps(json.loads(metadata.json()), indent=2)
print(metadata_str)

# Make an inference request
pred = client.model_infer(
    MODEL_NAME,
    request=InferenceRequest(
        inputs=[
            {
            "name": "float_input",
            "shape": [1, 11],
            "datatype": "FP32",
            "data": [7.4] * 11  # This is a shorter way to write the same d
ata
           }
        ]
    ),
)
json_resp_str = json.dumps(json.loads(pred.json()), indent=2)
print(json_resp_str)
```

Open Inference Protocol Using Curl

Consider the following instructions for using Open Interference Protocol Using Curl.

Procedure

You can construct the input of the inference call based on the Open Inference Protocol V2:

```
# cat ./examples/wine-input.json
```

Your input payload will look something like the following:

```
{
  "parameters": {
    "content_type": "pd"
  },
  "inputs": [
    {
      "name": "float_input",
      "shape": [
        1,
        11
      ],
      "datatype": "FP32",
      "data": [
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4
      ]
    }
  ]
}
```

```
curl -H "Content-Type: application/json" -H "Authorization: Bearer ${CDP_TOK
EN}" "https://${DOMAIN}/namespaces/serving-default/endpoints/[***ENDPOINT_NA
ME***]/v2/models/model/infer" -d @./examples/wine-input.json
```

You will receive response similar to the following:

```
{
  "model_name":"model",
  "model_version":"1",
  "outputs":[
    {
      "name":"variable",
      "datatype":"FP32",
      "shape":[1,1],
      "data":[5.535987377166748]
    }
  ]
}
```

Deploying Predictive Models

The following example illustrates how to train a model on a Cloudera AI workbench, register it, and then deploy it to Cloudera AI Inference.

About this task

In this section, the well-known wine classifier example is taken using the dataset from UC Irvine.

Procedure

1. Create a project in your Cloudera AI Workbench, open a session, and execute the following command to install the necessary Python packages:Upload the model artifact to your project's file system.

```
pip install open-inference-openapi mlflow onnx==1.13.1 onnxruntime skl2o
nnx pandas scikit-learn
```

2. Copy and paste the following Python code into your editor of choice in the workbench session:

```
import logging
import sys
import warnings
import onnx
from urllib.parse import urlparse
import numpy as np
import pandas as pd
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_s
core
from sklearn.model_selection import train_test_split
from sklearn import pipeline
from sklearn import linear_model
from urllib.parse import urlparse
from sklearn.preprocessing import FunctionTransformer
import mlflow
import mlflow.sklearn
from mlflow.models import infer_signature
from skl2onnx import to_onnx
from skl2onnx import convert_sklearn
from skl2onnx.common.data_types import FloatTensorType

def convert_to_onnx(model, data):
    #check the specifications
    initial_types = list(
    zip(
        data.columns.values.tolist(),
        [FloatTensorType([None, 1]) for _ in range(len(data.columns))],
    )
)
    onnx_model = convert_sklearn(model=model, initial_types=initial_types)
    print("onnx_model.type:",type(onnx_model))
    mlflow.set_tag("onnx_version",onnx.__version__)
    return onnx_model

def eval_metrics(actual, pred):
    rmse = np.sqrt(mean_squared_error(actual, pred))
    mae = mean_absolute_error(actual, pred)
    r2 = r2_score(actual, pred)
    return rmse, mae, r2
warnings.filterwarnings("ignore")
```

```
np.random.seed(40)

csv_url = (
        "http://archive.ics.uci.edu/ml"
        "/machine-learning-databases/wine-quality/winequality-red.csv"
    )
try:
    data = pd.read_csv(csv_url, sep=";")
except Exception as e:
    logger.exception(
        "Unable to download training & test CSV, check your internet conn
ection. Error: %s", e
    )
# Split the data into training and test sets. (0.75, 0.25) split.
train, test = train_test_split(data)
# The predicted column is "quality" which is a scalar from [3, 9]
train_x = train.drop(["quality"], axis=1)
test_x = test.drop(["quality"], axis=1)
train_y = train[["quality"]]
test_y = test[["quality"]]

alpha = 0.5
l1_ratio = 0.5

with mlflow.start_run():
    lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
    DummyScaler = FunctionTransformer(None)
    lr = pipeline.Pipeline(
    [("dummy", DummyScaler), ("reg", linear_model.ElasticNet(alpha=alpha,
                                                              l1_ratio=l1_r
atio,
                                                              random_state=
42))]
)
    lr.fit(train_x, train_y)
    predicted_qualities = lr.predict(test_x)
    (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
    print(f"Elasticnet model (alpha={alpha:f}, l1_ratio={l1_ratio:f}):")
    print(f"  RMSE: {rmse}")
    print(f"  MAE: {mae}")
    print(f"  R2: {r2}")
    mlflow.log_param("alpha", alpha)
    mlflow.log_param("l1_ratio", l1_ratio)
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("r2", r2)
    mlflow.log_metric("mae", mae)
    predictions = lr.predict(train_x)
    model_signature = infer_signature(train_x, predictions)
    onnx_model = convert_to_onnx(lr, test_x)
    mlflow.onnx.log_model(onnx_model, "model",
                          registered_model_name="ElasticnetWineModel",
                          signature=model_signature)
```

3. Run the above code. If it completes successfully, you should see the ElasticnetWineModel model listed on the **Registered Models** page on the **Cloudera AI** control plane UI.

4. Click on the model name to view details about the model, and deploy it to Cloudera AI Inference service service:

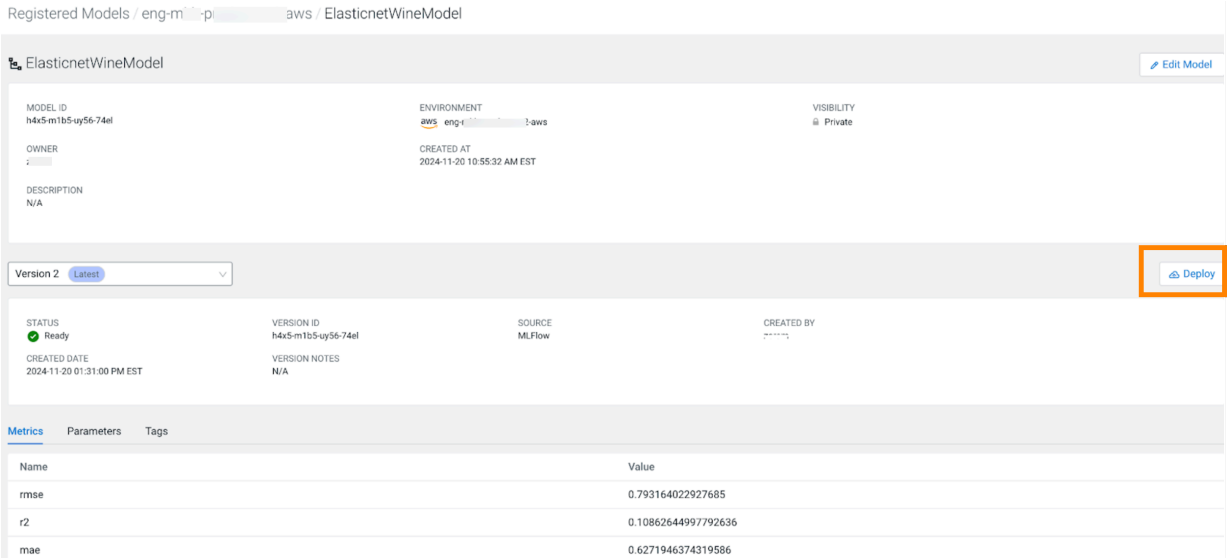

5. Click Deploy. The model endpoint creation dialogbox is displayed.
6. Select the Cloudera AI Inference service cluster you wish to deploy it to, and click Deploy.

   The **Create Endpoint** page is displayed.
7. Create the model endpoint using UI or API. Due to a known issue in the UI, choosing an instance type for non-NIM models is not working. If you want to choose an instance type for such models, you can use the Cloudera AI Inference service API with a payload like the following:

   Using UI

   You can use the **Create Model Endpoint** page to select a specific Cloudera AI Inference service instance and a model version from Cloudera AI Registry to create a new model endpoint. For more information, see Creating a Model Endpoint using UI

   Using API

```
$ cat deploy_wine_onnx.json
```

```
{
  "namespace": "serving-default",
  "name": "elasticnetwine",
  "source": {
    "registry_source": {
      "model_id": "<MODEL_ID>",
      "version": <VERSION>
    }
  },
  "resources": {
    "req_cpu": "2",
    "req_memory": "2Gi"
  },
  "instance_type": "m5.24xlarge",
  "autoscaling": {
    "min_replicas": "1",
    "max_replicas": "80",
    "autoscalingconfig": {
      "metric": "concurrency",
      "target": "100"
    }
  }
}
```

```
$ curl -XPOST -H "Authorization: Bearer ${CDP_TOKEN}" https://${DOMAIN}/
api/v1alpha1/deployEndpoint -d @./deploy_wine_onnx.json
```

8. When the model endpoint is in the Running state, you can interact with the endpoint. The following is an example of a Python client application running on Cloudera AI Workbench:

```
$ cat examples/new-wine-input.json
```

```
{
    "parameters": {
      "content_type": "pd"
    },
    "inputs": [
        {
          "name": "fixed_acidity",
          "shape": [1, 1],
          "datatype": "FP32",
          "data": [9.4]
        },
        {
          "name": "volatile_acidity",
          "shape": [1, 1],
          "datatype": "FP32",
          "data": [0.8000]
        },
        {
          "name": "citric_acid",
          "shape": [1, 1],
          "datatype": "FP32",
          "data": [0]
        },
        {
          "name": "residual_sugar",
          "shape": [1, 1],
          "datatype": "FP32",
          "data": [1.9]
        },
        {
          "name": "chlorides",
          "shape": [1, 1],
          "datatype": "FP32",
          "data": [0.076]
        },
        {
          "name": "free_sulfur_dioxide",
          "shape": [1, 1],
          "datatype": "FP32",
          "data": [11]
        },
        {
          "name": "total_sulfur_dioxide",
          "shape": [1, 1],
          "datatype": "FP32",
          "data": [34]
        },
        {
          "name": "density",
          "shape": [1, 1],
          "datatype": "FP32",
          "data": [0.9978]
        },
        {
```

```
            "name": "pH",
            "shape": [1, 1],
            "datatype": "FP32",
            "data": [3.51]
          },
          {
            "name": "sulphates",
            "shape": [1, 1],
            "datatype": "FP32",
            "data": [0.56]
          },
          {
            "name": "alcohol",
            "shape": [1, 1],
            "datatype": "FP32",
            "data": [9.4]
          }
      ]
}
```

9. You can use the above inference input payload in your Python client code as in the following example:

```
from open_inference.openapi.client import OpenInferenceClient, Inference
Request
import httpx
import requests
import json

CDP_TOKEN = json.load(open("/tmp/jwt"))["access_token"]
BASE_URL = '<ENDPOINT_BASE_URL>'
MODEL_NAME = '<MODEL_ID>'
headers = {'Authorization': 'Bearer ' + CDP_TOKEN,
           'Content-Type': 'application/json'}

httpx_client = httpx.Client(headers=headers)
client = OpenInferenceClient(base_url=BASE_URL, httpx_client=httpx_client)
# Check that the server is live, and it has the model loaded
client.check_server_readiness()
metadata = client.read_model_metadata(MODEL_NAME)
metadata_str = json.dumps(json.loads(metadata.json()), indent=2)
# Uncomment the next line to see model server metadata, which
# includes model name, version, deep learning platform type, and
# the shape of the input/output tensors supported by the model server.
#print(metadata_str)
# Read the input payload
payload = json.load(open("new-wine-input.json"))["inputs"]
# Make an inference request
pred = client.model_infer(
    MODEL_NAME,
    request=InferenceRequest(
        inputs=payload
    ),
)

json_resp_str = json.dumps(json.loads(pred.json()), indent=2)
print(json_resp_str)

# Output
#{
#  "model_name": "h2hy-o8t4-jg0p-qvnx",
#  "model_version": "1",
#  "outputs": [
#    {
```

```
#       "name": "variable",
#       "shape": [
#         1,
#         1
#       ],
#       "datatype": "FP32",
#       "data": [
#         5.535987377166748
#       ]
#     }
#   ]
#}
```

Related Information

ONNX models

Accessing Cloudera AI Inference service Metrics

Cloudera AI Inference service exposes Prometheus metrics for the deployed Model Endpoints. The UI displays plots of a few chosen metrics for each model endpoint.

For additional metrics, the Prometheus server can be accessed directly at https://${DOMAIN_NAME}/prometheus by passing the authorization bearer token.

Predictive models deployed with the NVIDIA Triton Inference Server and NVIDIA NIM embedding models export metrics prefixed with nv_. Refer to the *NVIDIA NIM documentation* for metrics exported by the NVIDIA NIM Large Language Model (LLM) runtimes for text-generation.

Related Information

Authentication of Cloudera AI Inference service

Nvidia NIM Large Language Models, Observability

Known issues

There are some known issues you might run into while using Cloudera AI Inference service.

- The following compute instance types are not supported by Cloudera AI Inference service:
  - Azure: NVadsA10_v5 series.
  - AWS: p4d.24xlarge
- Updating the description after a model has been added to a model endpoint will lead to a UI mismatch in the model builder for models listed by the model builder and the models deployed.
- When you create a model endpoint from the **Create Endpoint** page, even though the instance type selection is not mandatory, the instance creation fails if the instance type is not selected.
- DSE-39626: If no worker node can be found within 60 minutes to schedule a model endpoint that is either newly created or is scaling up from 0, the system will give up trying to create and schedule the replica. A common reason for this behavior is insufficient cloud quota, or capacity constraints on the cloud service provider's side. You could either ask for increased quota, or try to use an instance type that is more readily available.
- When updating an Model Endpoint with a specific GPU requirement, the instance type must be explicitly set again even if there is no change.

Related Information

Obtaining the kubeconfig for the Cloudera AI Inference service Cluster

NVadsA10_v5 Microsoft specification

p4d.24xlarge NAWS documentation

Cloudera AI

API

Date published: 2020-07-16
Date modified: 2024-11-21

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera AI API v2 ..... **4**

API v2 usage **6**

Cloudera AI API v2

Cloudera AI exposes a REST API that you can use to perform operations related to projects, jobs, and runs. You can use API commands to integrate Cloudera AI with third-party workflow tools or to control Cloudera AI from the command line.

API v2 supersedes the existing Jobs API. For more information on the Jobs API, see Jobs API in the Related information section, below.

How to view the API Specification

You can view the comprehensive API specification on the REST API v2 Reference page. See Related information, below, for the link.

You can also obtain the specification of the available API commands directly from Cloudera AI. In a browser, enter the following addresses:

- REST API: https://<domain name of Cloudera AI instance>/api/v2/swagger.html
- Python API: https://<domain name of Cloudera AI instance>/api/v2/python.html

You can also get json formatted output, by specifying swagger.json.

Note: If you test an API endpoint in the REST API HTML page, then in Authorize Value , enter Bearer < YOUR-API-KEY> and click Authorize. Otherwise, the API call returns the following error: "missing "Bearer " prefix in "authorization" header".

Quickstart

API key authentication

To get started, generate an API key. The API key is a randomly generated token that is unique to each user. It must be treated as highly sensitive information because it can be used to start jobs via the API. You need this API key to use in API calls.

1. Sign in to Cloudera AI.
2. In User Settings API Keys , click Create API Key.
3. Copy this API key to the clipboard.

Using `curl` from the command line

To use the `curl` command, it is convenient to store the domain and API key in environmental variables, as shown here:

1. Copy the API key.
2. Open a terminal, and store it to a variable. On unix-based systems:

   export API_KEY=<paste the API key value here>
3. Copy the domain name, which is in the address bar of the browser. On unix-based systems: export CDSW_DOMAIN=<domain> (a value like: ml-xxxx123456.com).

Example commands

If you have some projects, jobs, and runs already set up in your Cloudera AI Workbench, here are some commands to try:

- List available projects:

  ```
  curl -X GET -H "authorization: Bearer $API_KEY" https://$CDSW_DOMAIN/api/v2/projects | jq
  ```

  You can format the output for readability by piping through jq, a formatting utility.

- You can filter the output like so:

  ```
  curl -X GET -H “authorization: Bearer $API_KEY” https://$CDSW_DOMAIN/api/v2/projects?searchFilter=demo |  jq
  ```

  The output is limited to those projects that have the word “demo” in them.

  You can also paginate the output, for example by limiting each page to two projects. To do this, replace the string starting from the ‘?’ character with this: ?pageSize=2

  The output ends with a `next_page_token` and a string value. To get the next page use this: ?pageSize=2& pageToken=<token>

  Note: You have to add quotes around the entire https string because of the ampersand (&) character.

Using the Python client library

To use the Python API in your own code, first install the Python API client and point it to your cluster.

```
pip3 install https://$CDSW_DOMAIN/api/v2/python.tar.gz
```

Include the following code, and specify the values for <CDSW_DOMAIN> and <API_KEY> with variables or values for your installation.

```
# In a session:
    api_instance = default_client()
   # Outside a session:
   default_client("https://"+cluster, APIKEY)
```

Note: If you use default_client() in a session, no arguments are needed. If you use it outside of a session, you must provide the cluster name and API v2 key.

Then you can use commands in your script, such as a call to list projects:

```
projects = api_instance.list_projects()
```

The API returns objects that store values as attributes. To access the values, use dot notation. Do not use bracket notation as you would with a dictionary. For example:

```
myproj = client.create_project(...)
# This doesn't work:
myproj["id"]

# But this does
myproj.id
```

Check the Python documentation for a full list of the available API commands.

Using the Python client library in the REPL

Here is an example of a stub Python script that contains the environmental variables for your installation. Save it to your computer and run it locally to get a Python prompt for API commands.

demo.py

```
import clap
import argparse
```

```
parser = argparse.ArgumentParser(description='Test the generated python pac
kage.')
parser.add_argument("—host", type=str, help='The host name of yo
ur workbench")
parser.add_argument("—token", type=str, help='Your API key")
args = parser.parse_args()

config = clap.Configuration()
config.host = ars.host
client = cmlapi.ApiClient(config)
client.set_default_header("authorization", "Bearer " + args.token)
api = cmlapi.Apiapi(client)
```

Run the script from your command line:

```
python3 -i demo.py —host https://$CDSW_DOMAIN —token $API_KEY
```

This returns a Python prompt with api available. You can run api calls from the prompt, as follows:

```
>>> api
<cmlapi.api.api_api.ApiApi object at 0xlasjoid>
>>> api.api_list_projects()
```

You can specify a search filter, such as the following:

```
api.api_list_projects(searchFilter='demo')

api.api_list_projects(page_size=2)

api.api_list_projects(page_size=2, page_token='<token value>')
```

Related Information

Cloudera AI REST API v2 Reference

Cloudera AI Jobs API

Cloudera Data Science Workbench API v2

API v2 usage

You can use API v2 to perform actions on Projects, Jobs, Models, and Applications.

Set up the client

The client is the object you use for executing API commands. You obtain the client from the Cloudera AI cluster using your API key and the default_client() function.

Start by downloading the Python library directly from the workbench:

```
> pip3 install <workbench domain>/api/v2/python.tar.gz
```

Next, get an API key. Go to User Settings API Keys . Select Create API Key. Copy the generated API key value to the clipboard.

Create an instance of the API:

```
> import cmlapi
> client = cmlapi.default_client(url="<workbench domain>", cml_api_key="<api
 key>")
```

```
> client.list_projects()
```

If your workbench is using a custom self-signed certificate, you might need to include it when creating the client:

```
> config = cmlapi.Configuration()
> config.host = “<workbench domain>”
> config.ssl_ca_cert = "<path to SSL certificate>"
> api = cmlapi.ApiClient(config)
> api.set_default_header("authorization", "Bearer " + “<api key>”)
> client = cmlapi.CMLServiceApi(api)
```

Using the Project API

To list the available projects:

```
projects = client.list_projects() # returns the first 10
second_page_projects = client.list_projects(page_token=projects.next_page_t
oken) # returns the next 10
lots_of_projects = client.list_projects(page_size=100)
second_page_lots_of_projects = client.list_projects(page_size=100, page_t
oken=lots_of_projects.next_page_token) # must re-include the same page size
for future pages
filtered_projects = client.list_projects(search_filter=”production”) # retu
rns all projects with “production” in the name or description
```

Select a particular project ID, and then validate it with a command to fetch the project:

```
project = client.get_project(project_id=”<project id>”)
```

Delete a project with the following command:

```
client.delete_project(project_id=”<project id>”)
```

Using the Jobs API

You can list out the jobs in the project like so:

```
jobs = client.list_jobs(project_id=”<projectid>”)
```

The same searching and filtering rules apply as before. You can delete a job with:

```
client.delete_job(project_id=”<project id>”,job_id=”<job_id>”)
```

Finally you can get the job ID from a job response and create a job run for a job:

```
job_run =client.create_job_run(cmlapi.CreateJobRunRequest(), project_id=”<pr
oject id>”,job_id=”<job id>”)
```

If you wish to stop the job run, you can do so as well

```
client.stop_job_run(project_id=”<project id>”,job_id=”<job id>”, run_id=job_
run.id)
```

Check the status of a job:

```
client.list_job_runs(project_id, job_id, sort="-created_at", page_size=1)
```

Using the Models API

This example demonstrates the use of the Models API. To run this example, first do the following:

1. Create a project with the Python template and a legacy engine.
2. Start a session.
3. Run !pip3 install sklearn
4. Run fit.py

The example script first obtains the project ID, then creates and deploys a model.

```
projects = client.list_projects(search_filter=json.dumps({"name": “<your pro
ject name>”}))
project = projects.projects[0] # assuming only one project is returned by
 the above query
model_body = cmlapi.CreateModelRequest(project_id=project.id, name="Demo Mo
del", description="A simple model")
model = client.create_model(model_body, project.id)
model_build_body = cmlapi.CreateModelBuildRequest(project_id=project.id, mod
el_id=model.id, file_path="predict.py", function_name="predict", kernel="pyt
hon3")
model_build = client.create_model_build(model_build_body, project.id, mod
el.id)
while model_build.status not in [“built”, “build failed”]:
 print(“waiting for model to build...”)
time.sleep(10)
 model_build = client.get_model_build(project.id, model.id, model_build.id)
if model_build.status == “build failed”:
 print(“model build failed, see UI for more information”)
 sys.exit(1)
print(“model built successfully!”)
model_deployment_body = cmlapi.CreateModelDeploymentRequest(project_id=proje
ct.id, model_id=model.id, build_id=model_build.id)
model_deployment = client.create_model_deployment(model_deployment_body,
project.id, model.id, build.id)
while model_deployment.status not in [“stopped”, “failed”, “deployed”]:
 print(“waiting for model to deploy...”)
 time.sleep(10)
 model_deployment = client.get_model_deployment(project.id, model.id, model
_build.id, model_deployment.id)
if model_deployment.status != “deployed”:
 print(“model deployment failed, see UI for more information”)
 sys.exit(1)
print(“model deployed successfully!”)
```

Using the Applications API

Here is an example of using the Application API.

```
application_request = cmlapi.CreateApplicationRequest(
     name = "application_name",
     description = "application_description",
     project_id = project_id,
     subdomain = "application-subdomain",
     kernel = "python3",
     script = "entry.py",
```

```
     environment = {"KEY": "VAL"}
)
app = client.create_application(
     project_id = project_id,
     body = application_request
)
```

Using the Cursor class

The Cursor is a helper function that works with any endpoint used for listing items, such as list_projects, list_jobs, or list_runtimes. The Cursor returns an iterable object. The following example returns a list of runtimes.

```
cursor = Cursor(client.list_runtimes)
runtimes = cursor.items()
for rt in runtimes:
    print(rt.image_identifier)
```

The Cursor can also use a search filter, as shown in this example:

```
cursor = Cursor(client.list_runtimes, search_filter = json.dumps({"image_ide
ntifier":"jupyter"}))
```

End to end example

This example creates a project, job, and job run. For the job script, it uses the analysis.py file that is included in the Python template.

```
import cmlapi
import time
import sys
import random
import string
random_id=''.join(random.choice(string.ascii_lowercase) for i in range(10))
project_body = cmlapi.CreateProjectRequest(name="APIv2 Test Project " + ran
dom_id, description="Project for testing APIv2", template="Python")
project_result = client.create_project(project_body)
poll_retries = 5
while True: # wait for the project to reach the "success" state
  project = client.get_project(project_result.id)
  if project.creation_status == "success":
    break
  poll_retries -= 1
  if poll_retries == 0:
    print("failed to wait for project creation to succeed")
    sys.exit(1)
  time.sleep(2) # wait a couple seconds before the next retry

job_body = cmlapi.CreateJobRequest(name="APIv2 Test Job " + random_id, kerne
l="python3", script="analysis.py")
job_result = client.create_job(job_body, project_id=project_result.id)
job_run = client.create_job_run(cmlapi.CreateJobRunRequest(), project_id=p
roject_result.id, job_id=job_result.id)
```

Cloudera AI

Applications

Date published: 2020-07-16
Date modified: 2024-11-21

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Analytical Applications 4

Securing Applications 6

Limitations with Analytical Applications 6

Monitoring applications 7

Analytical Applications

This topic describes how to use a Cloudera AI Workbench to create long-running web applications.

About this task:

This feature gives data scientists a way to create Cloudera AI web applications/dashboards and easily share them with other business stakeholders. Applications can range from single visualizations embedded in reports, to rich dashboard solutions such as Tableau. They can be interactive or non-interactive.

Applications stand alongside other existing forms of workloads in Cloudera AI (sessions, jobs, experiments, models). Like all other workloads, applications must be created within the scope of a project. Each application is launched within its own isolated engine. Additionally, like models, engines launched for applications do not time out automatically. They will run as long as the web application needs to be accessible by any users and must be stopped manually when needed.

Before you begin:

Testing applications before you deploy

Before you deploy an application using the steps described here, make sure your application has been thoroughly tested. You can use sessions to develop, test, and debug your applications. You can test web apps by embedding them in sessions as described here: Web Applications Embedded in Sessions .

For Cloudera AI UI

1. Go to a project's **Overview** page.
2. Click Applications.
3. Click New Application.

4. Fill out the following fields.
   - **Name**: Enter a name for the application.
   - Run Application as: If the application is to run in a service account, select Service Account and choose the account from the dropdown menu.
   - **Subdomain**: Enter a subdomain that will be used to construct the URL for the web application. For example, if you use test-app as the subdomain, the application will be accessible at test-app.<ml-workspace-domain-name>.

     Subdomains should be valid DNS hostname characters: letters from a to z, digits from 0 to 9, and the hyphen.
   - **Description**: Enter a description for the application.
   - **Script**: Select a script that hosts a web application on either CDSW_READONLY_PORT or CDSW_APP_PORT. Applications running on either of these ports are available to any users with at least read access to the project. The Python template project includes an entry.py script that you can use to test this out.

     Note: Cloudera AI does not prevent you from running an application that allows a read-only user (i.e. Viewers) to modify files belonging to the project. It is up to you to make the application truly read-only in terms of files, models, and other resources belonging to the project.
   - **Engine Kernel and Resource Profile**: Select the kernel and computing resources needed for this application.

     Note: The list of options here is specific to the default engine you have specified in your Project Settings: ML Runtimes or Legacy Engines. Engines allow kernel selection, while ML Runtimes allow Editor, Kernel, Variant, and Version selection. Resource Profile list is applicable for both ML Runtimes and Legacy Engines.
   - **Set Environment Variables**: Click Set Environment Variables, enter the name and value for the new application variable, and click Add.

     If there is a conflict between the project-level and application-level environment variables, the application-level environment variables override the project-level environment variables.
5. Click Create Application.

For Cloudera AI APIv2

To create an application using the API, refer to this example:

Here is an example of using the Application API.

```
application_request = cmlapi.CreateApplicationRequest(
     name = "application_name",
     description = "application_description",
     project_id = project_id,
     subdomain = "application-subdomain",
     kernel = "python3",
     script = "entry.py",
     environment = {"KEY": "VAL"}
)
app = client.create_application(
     project_id = project_id,
     body = application_request
)
```

Results:

In a few minutes, you should see the application status change to **Running** on the **Applications** page. Click on the name of the application to access the web application interface.

What to do next:

You can Stop, Restart, or Delete an application from the **Applications** page.

If you want to make changes to an existing application, click Overview under the application name. Then go to the Settings tab to make any changes and update the application.

Securing Applications

You can provide access to Applications via either the CDSW_APP_PORT or the CDSW_READONLY_PORT. Any user with read or higher permissions to the project is able to access an application served through either port.

Important: Cloudera Public Cloud allows customers to maintain full ownership and control of their data and workloads and is designed to operate in some of the most restricted public cloud environments. Since Cloudera Public Cloud runs in a customer's cloud account, Security and Compliance is a shared responsibility between Cloudera and its public cloud customers. For more information, see *Cloudera's Shared Responsibility Model*.

- Securing project resources

  Cloudera AI applications are accessible by any user with read-only or higher permissions to the project. The creator of the application is responsible for managing the level of permissions the application users have on the project through the application. Cloudera AI does not actively prevent you from running an application that allows a read-only user (i.e. Viewers) to modify files belonging to the project.
- Public Applications

  By default, authentication for applications is enforced on all ports and users cannot create public applications. If desired, the Admin user can allow users to create public applications that can be accessed by unauthenticated users.

  To allow users to create public applications on an Cloudera AI Workbench:

  1. As an Admin user, turn on the feature flag in Admin Security by selecting Allow applications to be configured with unauthenticated access.
  2. When creating a new application, select Enable Unauthenticated Access.
  3. For an existing application, in Settings select Enable Unauthenticated Access.

  To prevent all users from creating public applications, go to Admin Security and deselect Allow applications to be configured with unauthenticated access. All existing public applications will immediately stop being publicly accessible.
- Transparent Authentication

  Cloudera AI can pass user authentication information to an Application, if the Application expects an authenticated request. The REMOTE-USER and REMOTE-USER-PERM HTTP headers will be injected to the Application hosted by the user.

  For example:

  ```
  Remote-user=<username>
  Remote-user-perm=<RO/RW/Unauthorized>
  ```

Related Information

Cloudera's Shared Responsibility Model

Limitations with Analytical Applications

This topic lists all the limitations associated with the Applications feature.

- Port availability

  Cloudera AI exposes only 2 ports per workload. Therefore, you can run a maximum of 2 web applications simultaneously, on these ports:

  - CDSW_APP_PORT
  - CDSW_READONLY_PORT

  By default, third-party browser-based editors run on CDSW_APP_PORT. Therefore, for projects that are already using browser-based editors, you are left with only one other port to run applications on: CDSW_READONLY_PORT.

Monitoring applications

You can monitor the CPU and memory usage of deployed applications.

To view the Monitoring Applications UI, in Applications, select the Monitoring Applications icon.

The Monitoring Applications UI provides the following information to help you monitor application health. The information is shown both as an instantaneous value and as a time series.

- CPU Usage: Shows the cpu usage as a percentage of total available CPU resources, and as amount of vCPU.
- Memory Usage: Shows the memory usage as a percentage of total available memory, and as amount of memory.

You can also select to view a specific application or all applications, and also select the time period to view.

Cloudera AI

Cloudera Accelerators for Machine Learning Projects (AMPs)

Date published: 2020-07-16
Date modified: 2025-01-29

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Accelerators for Machine Learning Projects....4

HuggingFace Spaces and Community AMPs.... 5
Enable users to deploy external spaces....5
Launch a HuggingFace or Community project....5
Launch a HuggingFace Space outside the catalog.... 6

Creating New AMPs....6

Creating New AMPs using API.... 6

Custom AMP Catalog.... 7

Add a catalog.... 8

Catalog File Specification.... 8

AMP Project Specification....10

Restarting a failed AMP setup....24

AMPs in airgapped environments.... 27
AMPs in airgapped environment with Proxy.... 27
AMPs in fully airgapped environments.... 27

Configuring Traefik readTimeout for large file uploads....28

Cloudera Accelerators for Machine Learning Projects

Cloudera Accelerators for Machine Learning Projects provide reference example projects in Cloudera AI. More than simplified quickstarts or tutorials, AMPs are fully-developed expert solutions created by Cloudera's research arm, Fast Forward Labs.

These solutions to common problems in this field demonstrate how to fully use the power of Cloudera AI. AMPs show you how to create Cloudera AI projects to solve your own use cases.

AMPs are available to install and run from the Cloudera AI user interface. As new AMPs are developed, they will become available to you for your study and use.

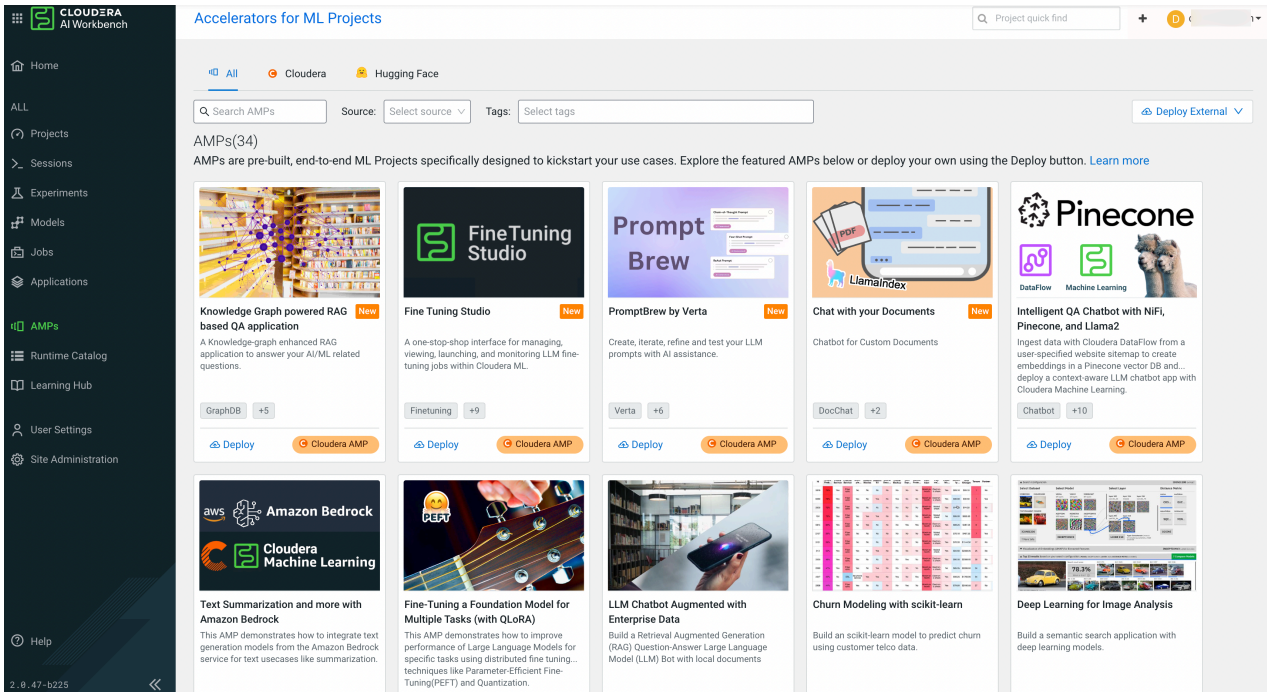

Using AMPs

Get started with AMPs:

1. Log in to your Cloudera AI workbench, and in the left panel click .
2. Click on an AMP tile to read its description.
3. Click Configure Project and provide any configuration values required by the AMP. The Description field explains how to determine these configuration values. After you click Launch Project, the installation process may take several minutes.
4. When the installation is complete, click Overview to read the documentation for the AMP and explore the code and project structure.

Note: If nothing appears in the panel, an administrator may need to reconfigure and refresh the catalog. In Site Administration , click Refresh. The administrator can also refresh periodically to add newly developed AMPs to the panel.

Related Information

Custom AMP Catalog

HuggingFace Spaces and Community AMPs

You can run HuggingFace Spaces or Community AMPs, which are both types of machine learning projects, in your Cloudera AI Workbench. In the HuggingFace tab in AMPs are several HF spaces that are tested and supported on Cloudera AI.

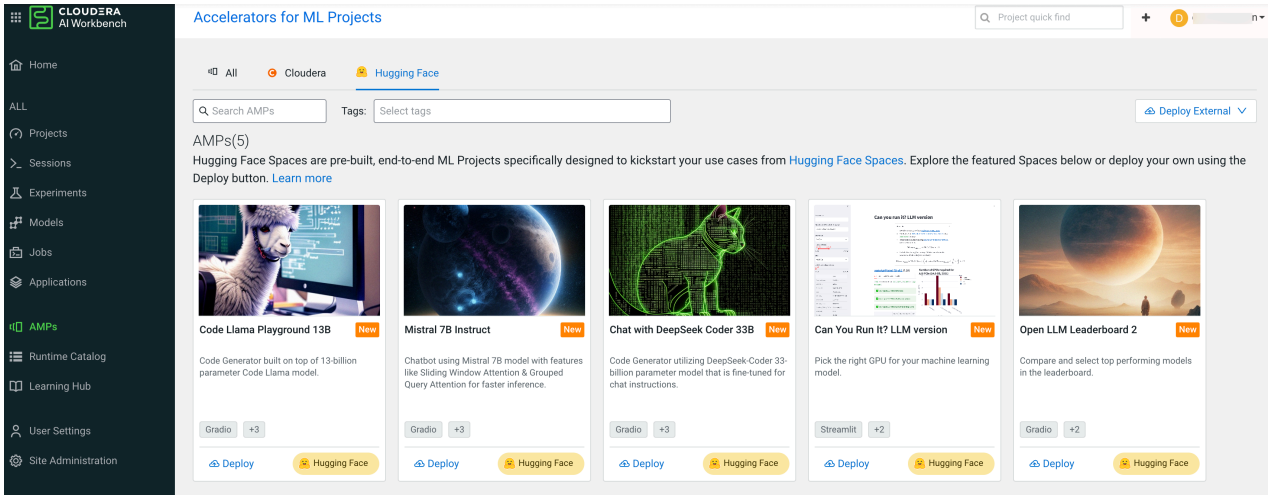

HuggingFace is an online community for machine learning (Hugging Face). It serves as a repository for machine learning projects and datasets. Spaces are individual projects or applications. Community AMPs are machine learning prototypes created by the Cloudera AI community.

There are a few things to keep in mind when launching HuggingFace Spaces or Community AMPs in your Cloudera AI Workbench.

- To launch a HuggingFace project, you first need to create a free account on HuggingFace and obtain an access token.
- Make sure that the Hugging Face project actually works. Many of the projects are community-created projects that may be experimental or no longer maintained.
- The HuggingFace Space must use the gradio or streamlit libraries to run on Cloudera AI.
- The Community tab is not visible by default. To enable this tab, in Site Administration Settings , select: Allow users to deploy community .

Enable users to deploy external spaces

By default, users cannot deploy Hugging Face spaces outside the curated collection in the Hugging Face tab.

To enable users to deploy external Hugging Face spaces, in Site Administration Settings , select:

- Allow users to deploy external Hugging Face Space

When selected, the ability to deploy external HuggingFace spaces is available.

Launch a HuggingFace or Community project

You can launch a HuggingFace Space or community project by clicking Deploy for the respective space.

In Configure Project, you can set some parameters for the space.

If needed, enter your Hugging Face access token. The environment variables HF_TOKEN and HUGGING_FACE_HUB_TOKEN both require the same value.

Note: Do not reduce any of the values specified for cpu, memory, or gpu.

Click Launch Project to deploy the project in your workbench. Cloudera AI will run scripts to download the project from the HuggingFace space and install it in Cloudera AI. This process can take some time.

Launch a HuggingFace Space outside the catalog

It is also possible to deploy Hugging Face spaces that are not included in the AMP catalog. Note that there are several caveats on which HuggingFace spaces may be able to work in Cloudera AI, as described above.

Click Deploy External to open the UI for launching an external project, and follow the instructions. The project is imported from HuggingFace or the specified repository

Creating New AMPs

One great use for AMPs is to showcase reference examples specific to your business by creating your own AMPs in-house. Once a data science project has been built in Cloudera AI, you can package it and have the Cloudera AI Admin add it to the AMP Catalog.

Each individual AMP requires a project metadata file, which defines the environmental resources needed by the AMP, and the setup steps to install the AMP in a Cloudera AI Workbench. See AMP Project Specification for details.

Note: You can store your AMPs in a git repo hosted on Github, Github Enterprise, or GitLab servers (not limited to github.com or gitlab.com.)

Additionally, only simple authentication is supported, such as passing an API key, or including the username and password, as part of the URL. If additional authentication steps are required, then that git host is not supported.

You can also look at an example for Cloudera Accelerators for Machine Learning Projects, such as: .project-metadata.yaml.

Related Information

AMP Project Specification

Creating New AMPs using API

You can create AMPs using API V2.

1. Get the valid runtime_identifier by using the GET request using filter parameters like kernel, editor, and edition.

   For example, if you want to create the AMP and it requires the following runtime based on project-metadata.yaml of the AMP, then use the filter like {"kernel":"Python 3.9","editor":"Workbench","edition":"Standard"} to get a smaller list of runtimes:

   ```
   curl -X GET "https://<workbenchURL>/api/v2/runtimes?search_filter=%7B%22
   kernel%22%3A%22Python%203.9%22%2C%22editor%22%3A%22Workbench%22%2C%22edi
   tion%22%3A%22Standard%22%7D" -H "accept: application/json" -H "Authoriza
   tion: Bearer <API Token>"
   ```

   This command will display a list of runtimes that match the filter.

2. Using the runtime_identifier obtained in the previous step, you can create an AMP. Following is the sample payload used to trigger the creation of one of the AMPs. You can add additional key-value pairs as needed under the `environment` object based on the project-metadata.yaml file of the AMP you are trying to create.

```
{
  "configure_prototype_request": {
    "execute_amp_steps": true,
    "run_import_tasks": true,
    "runtime_identifier": "docker.repository.cloudera.com/cloudera/cdsw/ml
-runtime-workbench-python3.9-standard:2024.05.2-b14"
  },
  "create_project_request": {
    "description": "amps v2",
    "environment": {
      "HF_TOKEN": "<HF TOKEN>"
    },
    "git_url": "https://github.com/cloudera/CML_AMP_Image_Analysis",
    "name": "amps_v2",
    "template": "git",
    "visibility": "private"
  }
  }
```

Example: The following is an example of creating an AMP that requires multiple environment variables with IBM WatsonXAI.

```
{
  "configure_prototype_request": {
    "execute_amp_steps": true,
    "run_import_tasks": true,
    "runtime_identifier": "docker.repository.cloudera.com/cloudera/cdsw/ml
-runtime-pbj-workbench-python3.9-standard:2024.10.1-b12"
  },
  "create_project_request": {
    "description": "watson amps v2",
    "environment": {
      "IBM_WATSONXAI_ENDPOINT": "https://us-south.ml.cloud.ibm.com",
      "IBM_API_KEY": "<Put valid key>",
      "IBM_PROJECT_ID": "<Put valid project ID>"
    },
    "git_url": "https://github.com/agupta-git/CML_AMP_watsonxai",
    "name": "watson_v2",
    "template": "git",
    "visibility": "private"
  }
}
```

3. You can now see that a new project has been created in your Cloudera AI Workbench which is carrying out the tasks listed in its project-metadata.yaml file. After all the tasks are executed, your new AMP-project will be ready to use.

Custom AMP Catalog

An AMP catalog is a collection of AMPs that can be added to a workbench as a group. Cloudera AI ships with the Cloudera AMP catalog, containing AMPs developed by Cloudera Fast Forward Labs, but you can also create and add your own catalog, containing AMPs developed by your organization.

To create an AMP catalog, you need to create a YAML file called the catalog file. This file is hosted on GitHub or another git server. This could be either a public or a private git server.

The catalog file contains information about each AMP in the catalog, and provides a link to the AMP repository itself. The catalog file specification is shown in Catalog File Specification.

You can also look at the Cloudera catalog file for an example. To view the file, click directly on the URL for Cloudera in Catalog Sources.

For more details on creating the AMPs that you will include in your catalog, see Creating New AMPs.

One use case you might consider is creating a fork of the Cloudera AMP catalog, in order to host it internally. In this case, you will need to edit the URLs in the catalog and project metadata files to point to the appropriate internal resources.

Related Information

Catalog File Specification

Creating New AMPs

Add a catalog

The collection of AMPs available to end users can draw from one or more sources. For example, you might have an internal company catalog in addition to the default Cloudera catalog. The catalog must be provided with a catalog file and one or more project metadata YAML files.

About this task

Specify Catalog File URL if your git hosting service allows you to access the raw content of the repo without authenticating. (That is, the source files can be retrieved with a curl command, and do not require logging into a web page). Otherwise, specify the Git Repository URL. To use a git repository as a catalog source, the catalog file and the AMP files must be in a repository that can be cloned with `git clone` without authentication.

Procedure

1. As an Administrator, go to Site Administration .
2. Select Git Repository URL or Catalog File URL to specify a new source. Paste or enter the URL to the new source, and file name for the catalog file if necessary.
3. Click Add Source.

   The catalog YAML file is loaded, and the projects found there are displayed in Catalog Entries.
4. If there are projects that are not yet ready for use, or that should not be displayed in the catalog, deselect Enabled in the Catalog Entries.

Catalog File Specification

The Catalog file is a YAML file that contains descriptive information and metadata for the displaying the AMP projects in the Project Catalog.

Fields

Fields are in snake_case. Each project in the catalog uses the following fields:

Field Name	Type	Example	Description
name	string	name: Cloudera	Required. Name of the catalog, displayed as Source in the Prototype Catalog tab.

Field Name	Type	Example	Description
entries	string	entries:	Required. Contains the entries for each project.
title	string	title: Churn Modeling	Required. The title of the AMP, as displayed in the Prototype Catalog.
label	string	label: churn-prediction	Required.
short_description	string	short_description: Build an scikit-learn model...	Required. A short description of the project. Appears on the project tile in the Prototype Catalog.
long_description	string	long_description: >- This project demonstrates...	Required. A longer description that appears when the user clicks on the project tile.
image_path	string	image_path: >- https://raw .git...	Required. Path to the image file that displays in the Prototype Catalog.
tags	string	tags: - Churn Prediction - Logistic Regression	Required. For sorting in the Prototype Catalog pane.
git_url	string	git_url: "https:..."	Required. Path to the git repository for the project.
git_ref	string	git_ref: 9e56b6578e37185777380e2a47b3/83d9ic6a3	Optional. Git ref (branch name, commit hash/tag name).
is_prototype	boolean	is_prototype: true	Optional. Indicates the AMP should be displayed in the Prototype Catalog. Use if coming_soon is not used.
coming_soon	boolean	coming_soon: true	Optional. Displays the AMP in the Prototype Catalog with a “COMING SOON” watermark. Use if is_prototype is not used.

Example:

```
name: Cloudera

entries:
   - title: Churn Modeling with scikit-learn
     label: churn-prediction
     short_description: Build an scikit-learn model to predict churn using
 customer telco data.
     long_description: >-
     This project demonstrates how to build a logistic regression classific
ation model to predict the probability
     that a group of customers will churn from a fictitious telecommunicatio
ns company. In addition, the model is
```

```
     interpreted using a technique called Local Interpretable Model-agnos
tic Explanations (LIME). Both the logistic
     regression and LIME models are deployed using Cloudera AI's real-time
 model deployment capability and interact with a
     basic Flask-based web application.
     image_path: >-
      https://raw.githubusercontent.com/cloudera/Applied-ML-Prototypes/mas
ter/images/churn-prediction.jpg
    tags:
      - Churn Prediction
      - Logistic Regression
      - Explainability
      - Lime
    git_url: "https://github.com/cloudera/CML_AMP_Churn_Prediction"
    is_prototype: true
```

AMP Project Specification

AMP projects include a project metadata file that provides configuration and setup details. These details may include environment variables and tasks to be run on startup.

YAML File Specification # Version 1.0

The project metadata file is a YAML file. It must be placed in your project's root directory, and must be named .project-metadata.yaml. The specifications for this file are listed below. You can also look at an example for one of the AMPs, such as:.project-metadata.yaml.

Fields

Fields for this YAML file are in snake_case. String fields are generally constrained by a fixed character size, for example string(64) is constrained to contain at most 64 characters. Click Show to see the list of fields.

Field Name	Type	Example	Description
name	string(200)	ML Demo	Required: The name of this project prototype. Prototype names do not need to be unique.
description	string(2048)	This demo shows off some cool applications of ML.	Required: A description for this project prototype.
author	string(64)	Cloudera Engineer	Required: The author of this prototype (can be the name of an individual, team, or organization).
date	date string	"2020-08-11"	The date this project prototype was last modified. It shall be in the format: "YYYY-MM-DD" (quotation marks are required).

Field Name	Type	Example	Description
specification_version	string(16)	0.1	Required: The version of the YAML file specification to use.
prototype_version	string(16)	1.0	Required: The version of this project prototype.
shared_memory_limit	number	0.0625	Additional shared memory in GB available to sessions running in this project. The default is 0.0625 GB (64MB).
environment_variables	environment variables object	See below	Global environment variables for this project prototype.
feature_dependencies	feature_dependencies	See below	A list of feature dependencies of this AMP. A missing dependency in workbench blocks the creation of the AMP.
engine_images	engine_images	See below	Engine images to be used with the AMP. These are recommendations and it does not prevent the user from launching an AMP with non recommended engine images.
runtimes	runtimes	See below	Runtimes to be used with the AMP. These are recommendations and it does not prevent the user from launching an AMP with non recommended runtimes.
tasks	task list	See below	A sequence of tasks, such as running Jobs or deploying Models, to be run after project import.

Example

```
name: ML Demo
description: >-
This demo shows off some cool applications of ML.
author: Cloudera Engineer
date: '2020-08-11T17:40:00.839Z'
specification_version: 1.0
environment_variables:
...
tasks:
```

```
...
```

Environment variables object

The YAML file can optionally define any number of global environment variables for the project under the environment field. This field is an object, containing keys representing the names of the environment variables, and values representing details about those environment variables. Click Show to see the list of fields in the Environment variables object.

Field Name	Type	Example	Description
default	string	"3"	The default value for this environment variable. Users may override this value when importing this project prototype.
description	string	The number of Model replicas, 3 is standard for redundancy.	A short description explaining this environment variable.
required	boolean	true	Whether the environment variable is required to have a non-empty value, the default is false.

Example: This example creates four environment variables.

```
environment_variables:
  AWS_ACCESS_KEY:
    default: ""
    description: "Access Key ID for accessing S3 bucket"
  AWS_SECRET_KEY:
    default: ""
    description: "Secret Access Key for accessing S3 bucket"
    required: true
  HADOOP_DATA_SOURCE:
    default: ""
    description: "S3 URL to large data set"
    required: false
  MODEL_REPLICAS:
    default: "3"
    description: "Number of model replicas, 3 is standard for redundancy"
    required: true
```

Feature Dependencies

AMPs might depend on some optional features of a workbench. The feature_dependencies field accepts a list of such features. Unsatisified feature dependencies prevent the AMP from being launched in a workbench, and display an appropriate error message. The supported feature dependencies are as follows:

- model_metrics

Runtimes Specification

The runtimes field accepts a list of runtimes objects defined as follows. This Runtimes specification can be added per task or per project.

```
- editor: the_name_of_the_editor # case-sensitive string required. e.g. Work
bench, Jupyter, etc. (how it appears in the UI)
  kernel: the_kernel # case-sensitive string required. e.g. Python 3.6, Pyth
on 3.8, R 3.6, etc. (how it appears in the UI)
  edition: the_edition # case-sensitive string required. e.g. Standard, N
vidia GPU, etc. (how it appears in the UI)
  version: the_short_version # case-sensitive string optional. e.g. 2021.
03, 2021.05, etc. (how it appears in the UI)
  addons: the_list_addons_needed # list of case-sensitive strings optional.
 e.g Spark 2.4.7 - CDP 7.2.11 - CDE 1.13, etc. (how it appears in the UI)
```

This example specifies the Runtimes the Workbench version for Python 3.8.

```
runtimes:
  - editor: Workbench
    kernel: Python 3.8
    edition: Standard
    addons: ['Spark 2.4.7 - CDP 7.2.11 - CDE 1.13']
```

Engine Images Specification

The engine_images field accepts a list of engine_image objects defined as follows:

```
- image_name: the_name_of_the_engine_image # string (required)
  tags: # list of strings (optional)
    - the_tag_of_engine_image
    - ...
```

This example specifies the official engine image with version 11 or 12:

```
engine_images:
  - image_name: engine
    tags:
      - 12
      - 11
```

This example specifies the most recent version of the dataviz engine image in the workbench:

```
engine_images:
  - image_name: cmldataviz
  - image_name: cdswdataviz
```

Note that when specifying CDV images, both cmldataviz and cdswdataviz must be specified. When tags are not specified, the most recent version of the engine image with the matching name is recommended. The following rule is used to determine the most recent engine_image with the matching name:

Official Engine (engine) and CDV (cmldataviz and cdswdataviz) images

Since the officially released engine images follow semantic versioning (where a newer version is always larger than any older version, when compared with >), the most recent engine image is the one with the largest tag. For example, engine:14 will be recommended over engine:13 and cmldataviz:6.3.4-b13 is recommended over cmldataviz:6.2.1-b12.

Custom engine images

There is no way for Cloudera AI to determine the rules for customer custom engine image tags, and therefore there is no reliable way to determine the most recent custom engine image. Use the engine image that has the correct matching name and has the newest ID. The newest ID means that the engine image is the most recently added engine image.

Task list

This defines a list of tasks that can be automatically run on project import. Each task will be run sequentially in the order they are specified in this YAML file. Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	create_job	Required: The type of task to be executed. See below for a list of allowed types.
short_summary	string	Creating a Job that will do a task.	A short summary of what this task is doing.
long_summary	string	Creating a Job that will do this specific task. This is important because it leads up to this next task.	A long summary of what this task is doing.

Jobs

Create Job

Example

```
- type: create_job
  name: howdy
  entity_label: howdy
  script: greeting.py
  arguments: Ofek 21
  short_summary: Creating a job that will greet you.
  environment_variables:
  SAMPLE_ENVIRONMENT_VARIABLE: CREATE/RUN_JOB
  kernel: python3
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	create_job	Required: Must be create_job.
name	string	howdy	Required: Job name.
entity_label	string	howdy	Required: Uniquely identifies this job for future tasks, i.e. run_job tasks. Entity labels must be lowercase alphanumeric,

Field Name	Type	Example	Description
			and may contain hyphens or underscores.
script	string	greeting.py	Required: Script for this Job to run.
kernel	string	python3	Required: What kernel this Job shall use. Acceptable values are python2, pyth on3, r, and scala. Note that scala might not be supported for every cluster.
arguments	string	Ofek 21	Command line arguments to be given to this Job when running.
environment_variables	environment variables object	See above	See above
cpu	number	1.0	The amount of CPU virtual cores to allocate for this Job, the default is 1.0.
memory	number	1.0	The amount of memory in GB to allocate for this Job, the default is 1.0.
gpu	integer	0	The amount of GPU to allocate for this Job, the default is 0.
timeout	integer	10	The amount of time in minutes to wait before timing out this Job, the default is 10.
timeout_kil	boolean	false	Whether or not to stop this Job when it times out, the default is false.

Run Job

Example run job task:

```
- type: run_job
  entity_label: howdy
  short_summary: Running the job that will greet you.
  long_summary: >-
    Running the job that will greet you. It will greet you by the name
    which is the first and only command line argument.
```

Most Job run tasks shall just contain the type and entity_label fields. Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	run_job	Required: Must be run_job.
entity_label	string	howdy	Required: Must match an entity_label of a previous create_job task.

However, they can optionally override previously defined fields. Click Show to see the list of fields.

Field Name	Type	Example	Description
script	string	greeting.py	Required: Script for this Job to run.
kernel	string	python3	Required: What kernel this Job shall use. Acceptable values are python2, pyth on3, r, and scala. Note that scala might not be supported for every cluster.
arguments	string	Ofek 21	Command line arguments to be given to this Job when running.
environment_variables	environment variables object	See above	See above
cpu	number	1.0	The amount of CPU virtual cores to allocate for this Job, the default is 1.0
memory	number	1.0	The amount of memory in GB to allocate for this Job, the default is 1.0.
gpu	integer	0	The amount of GPU to allocate for this Job, the default is 0.
shared_memory_limit	number	0.0625	Limits the additional shared memory in GB that can be used by this Job, the default is 0.0625 GB (64MB).

Models

Note: All models have authentication disabled, so their access key alone is enough to interact with them.

Resources object

Models may define a resources object which overrides the amount of resources to allocate per Model deployment.

Click Show to see the list of fields.

Field Name	Type	Example	Description
cpu	number	1.0	The number of CPU virtual cores to allocate per Model deployment.
memory	number	2.0	The amount of memory in GB to allocate per Model deployment.
gpu	integer	0	The amount of GPU to allocate per Model deployment.

For example:

```
resources:
  cpu: 1
  memory: 2
```

Replication policy object

Models may define a replication policy object which overrides the default replication policy for Model deployments.

Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	fixed	Must be fixed if present.
num_replicas	integer	1	The number of replicas to create per Model deployment.

For example:

```
replication_policy:
  type: fixed
  num_replicas: 1
```

Model examples list

Models may include examples, which is a list of objects containing a request and response field, each containing a valid object inside, as shown in the example:

```
examples:
  - request:
      name: Ofek
      age: 21
    response:
      greeting: Hello Ofek (21)
  - request:
      name: Jimothy
      age: 43
    response:
      greeting: Hello Coy (43)
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
request	string	See above	Required: An example request object.
response	string	See above	Required: The response to the above example request object.

Create Model

Example:

```
- type: create_model
  name: Say hello to me
  entity_label: says-hello
  description: This model says hello to you
  short_summary: Deploying a sample model that you can use to greet you
  access_key_environment_variable: SHTM_ACCESS_KEY
  default_resources:
  cpu: 1
  memory: 2
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	create_model	Required: Must be create_model.
name	string	Say hello to me	Required: Model name
entity_label	string	says-hello	Required: Uniquely identifies this model for future tasks, i.e. build_model and deploy_model tasks. Entity labels must be lowercase alphanumeric, and may contain hyphens or underscores.
access_key_environment_variable	string	SHTM_ACCESS_KEY	Saves the model's access key to an environment variable with the specified name.
default_resources	resources object	See above	The default amount of resources to allocate per Model deployment.

Field Name	Type	Example	Description
default_replication_policy	replication policy object	See above	The default replication policy for Model deployments.
description	string	This model says hello to you	Model description.
visibility	string	private	The default visibility for this Model.

Build Model

Example

```
- type: build_model
  entity_label: says-hello
  comment: Some comment about the model
  examples:
    - request:
        name: Ofek
        age: 21
    response:
      greeting: Hello Ofek (21)
  target_file_path: greeting.py
  target_function_name: greet_me
  kernel: python3
  environment_variables:
    SAMPLE_ENVIRONMENT_VARIABLE: CREATE/BUILD/DEPLOY_MODEL
```

Field Name	Type	Example	Description
type	string	build_model	Required: Must be buil d_model.
entity_label	string	says-hello	Required: Must match an entity_label of a previous create_model task.
target_file_path	string	greeting.py	Required: Path to file that will be run by Model.
target_function_name	string	greet_me	Required: Name of function to be called by Model.
kernel	string	python3	What kernel this Model shall use. Acceptable values are python2, pyth on3, r, and scala. Note that scala might not be supported for every cluster.

Field Name	Type	Example	Description
comment	string	Some comment about the model	A comment about the Model.
examples	model examples list	See above	A list of request/response example objects.
environment_variables	environment variables object	See above	See above

Deploy Model

Example:

```
- type: deploy_model
  entity_label: says-hello
  environment_variables:
  SAMPLE_ENVIRONMENT_VARIABLE: CREATE/BUILD/DEPLOY_MODEL
```

Most deploy model tasks shall only contain the type and entity_label fields. Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	deploy_model	Required: Must be depl oy_model.
entity_label	string	says-hello	Required: Must match an entity_label of a previous deploy_model task.

However, they can optionally override previously defined fields. Click Show to see the list of fields.

Field Name	Type	Example	Description
cpu	number	1.0	The number of CPU virutal cores to allocate for this Model deployment.
memory	number	2.0	The amount of memory in GB to allocate for this Model deployment.
gpu	integer	0	The amount of GPU to allocate for this Model deployment.
replication_policy	replication policy object	See above	The replication policy for this Model deployment.
environment_variables	environment variables object	See above	Overrides environment variables for this Model deployment.

Applications

Start Application

Example:

```
- type: start_application
  subdomain: greet
  script: greeting.py
  environment_variables:
    SAMPLE_ENVIRONMENT_VARIABLE: START_APPLICATION
  kernel: python3
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
bypass_authentication	boolean	True	When enabled, allows unauthenticated access to an application
type	string	start_application	Required: Must be start_application.
subdomain	string	greet	Required: Application subdomain, which must be unique per Application, and must be alphanumeric and hyphen-delimited. Application subdomains are also converted to lowercase.
kernel	string	python3	Required: What kernel this Application shall use. Acceptable values are python2, python3, r, and scala. Note that scala might not be supported for every cluster.
entity_label	string	greeter	Uniquely identifies this application for future tasks. Entity labels must be lowercase alphanumeric, and may contain hyphens or underscores.
script	string	greeting.py	Script for this Application to run.
name	string	Greeter	Application name, defaults to 'Untitled application'.

Field Name	Type	Example	Description
description	string	Some description about the Application	Application description, defaults to 'No description for the app'.
cpu	number	1.0	The number of CPU virutal cores to allocate for this Application.
memory	number	1.0	The amount of memory in GB to allocate for this Application.
gpu	integer	0	The amount of GPU to allocate for this Application.
shared_memory_limit	number	0.0625	Limits the additional shared memory in GB that can be used by this application, the default is 0.0625 GB (64MB).
environment_variables	environment variables object	See above	See above
static_subdomain	boolean	True	When enabled, subdomain will not get randomized.

Experiments

Run Experiment

Example:

```
- type: run_experiment
  script: greeting.py
  arguments: Ofek 21
  kernel: python3
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	run_experiment	Required: Must be run_experiment.
script	string	greeting.py	Required: Script for this Experiment to run.
entity_label	string	test-greeter	Uniquely identifies this experiment for future tasks. Entity labels must be lowercase alphanumeric,

Field Name	Type	Example	Description
			and may contain hyphens or underscores.
arguments	string	Ofek 21	Command line arguments to be given to this Experiment when running.
kernel	string	python3	What kernel this Experiment shall use. Acceptable values are python2, python3, r, and scala. Note that scala might not be supported for every cluster.
comment	string	Comment about the experiment	A comment about the Experiment.
cpu	number	1.0	The amount of CPU virtual cores to allocate for this Experiment.
memory	number	1.0	The amount of memory in GB to allocate for this Experiment.
gpu	number	0	The amount of GPU to allocate for this Experiment.

Sessions

Run Sessions

Example:

```
- type: run_session
  name: How to be greeted interactively
  code: |
    import os
    os.environ['SAMPLE_ENVIRONMENT_VARIABLE'] = 'SESSION'

    !python3 greeting.py Ofek 21

    import greeting
    greeting.greet_me({'name': 'Ofek', 'age': 21})
  kernel: python3
  memory: 1
  cpu: 1
  gpu: 0
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	run_session	Required: Must be run_session.
	string	See above for code, greeting.py for script	Required: Either the code or script field is required to exist for the run Session task, not both. code is a direct block of code that will be run by the Session, while script is a script file that will be executed by the Session.
kernel	string	python3	Required: What kernel this Session shall use. Acceptable values are python2, python3, r, and scala. Note that scala might not be supported for every cluster.
cpu	number	1.0	Required: The amount of CPU virtual cores to allocate for this Session.
memory	number	1.0	Required: The amount of memory in GB to allocate for this Session.
entity_label	string	greeter	Uniquely identifies this session for future tasks. Entity labels must be lowercase alphanumeric, and may contain hyphens or underscores.
name	string	How to be greeted interactively	Session name.
gpu	integer	0	The amount of GPU to allocate for this Session.

Restarting a failed AMP setup

If any task fails after initiating an AMP, you can resume it from the last unsuccessful step using the Resume option. You can use the Redeploy option to re-import and execute tasks based on the modified .project-metadata.yaml file.

Resuming an AMP

The resume action lets you resume the tasks from the last failed step. This eliminates the need to create a new project to launch the AMP again. You can resume an AMP setup process only from the failure state.

Redeploying an AMP

When you redeploy, it stops all the long-running workloads (Model Deployment or Application) linked to the current prototype, deletes the existing tasks, and re-imports tasks from the .project-metadata.yaml file.

- In the Accelerated ML Projects page, after launching a project, in the Setup Steps window, you can use the Redeploy or Resume option to restart the AMP setup process.

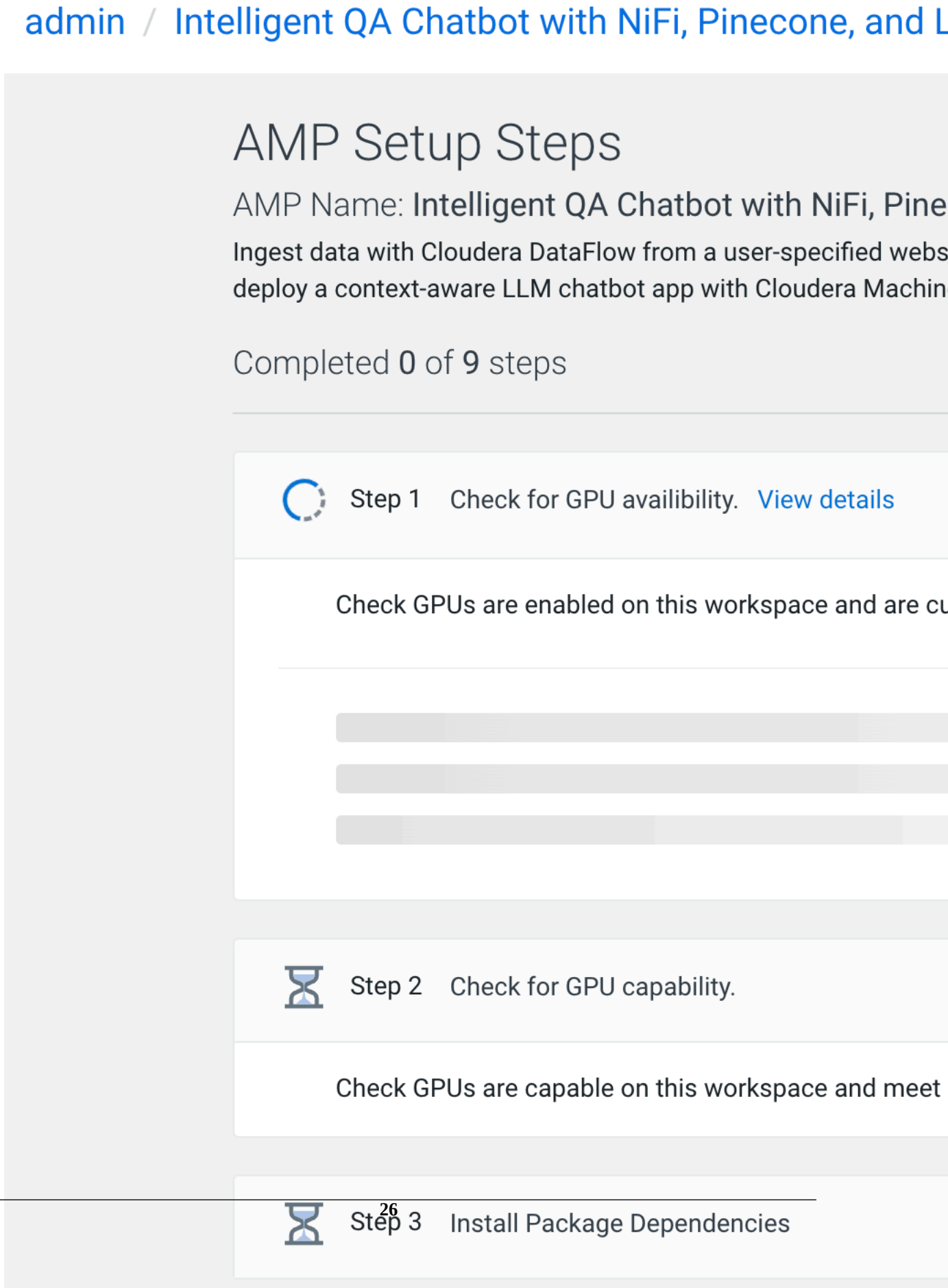

AMPs in airgapped environments

In an airgapped installation, the default AMPs catalog included at installation and default AMPs may be inaccessible. Hence, you must allow hostnames in your proxy configuration or configure your environment for AMPs to work.

AMPs in airgapped environment with Proxy

If you are using a proxy setup, then you must allow the following hostnames for AMPs to work.

- *.storage.googleapis.com
- *.raw.githubusercontent.com
- *.pypi.org
- *.pythonhosted.org
- *.pypi.python.org
- github.com
- *.github.com
- huggingface.co
- *.huggingface.co
- *.pinecone.io
- s3.amazonaws.com
- download.pytorch.org
- objects.githubusercontent.com

These host names shall be specified in the proxy, along with any other endpoints that are needed for your workbench.

AMPs in fully airgapped environments

Some AMPs are built to work in environments that does not have outbound network access. Learn how to set up the environment for AMPs to work.

Before you begin

- Host your internal PyPI server.
- Create your Custom AMP Catalog or download an AMP project from the GitHub repository listed in the below table. After downloading, upload it as a ZIP file during the project creation process.

Steps

- Each AMP has a requirements.txt file, which lists all the Python packages it needs. This file is present in the root of the public GitHub repository.
- You must ensure that all the packages listed in the requirements.txt are available on your custom-hosted PyPI server.
- Change the PIP_INDEX_URL environment variable to point to the custom-hosted PyPI server URL.

Below are the list of AMPs that can be used in the airgapped environments without outbound network access.

Title	Label	Short Description	Git URL
Churn Modeling with scikit-learn	churn-prediction	Build an scikit-learn model to predict churn using customer telco data.	Churn Modeling with scikit-learn
Deep Learning for Anomaly Detection	anomaly-detection	Apply modern and deep learning techniques for anomaly detection to identify network intrusions.	Deep Learning for Anomaly Detection

Title	Label	Short Description	Git URL
Canceled Flight Prediction	canceled-flight-prediction	Perform analytics on a large airline dataset with Spark and build an XGBoost model to predict flight cancellations.	Canceled Flight Prediction
Getting Started with the Cloudera AI API	apiv2	Demonstration of how to use the Cloudera AI API to interact with Cloudera AI.	Getting Started with the Cloudera AI API
AutoML with TPOT	automl-with-tpot	AutoML using TPOT, distributed with Dask.	AutoML with TPOT
Train Gensim's Word2Vec	gensim-w2v	Demonstration of how to train Gensim's Word2Vec for a non-language use case.	Train Gensim's Word2Vec
Continuous Model Monitoring	continuous-model-monitoring	Demonstration of how to perform continuous model monitoring on Cloudera AI using Model Metrics and Evidently.ai dashboards.	Continuous Model Monitoring
Distributed XGBoost with Dask on Cloudera AI	dask-on-cml	How to perform distributed training of an XGBoost model using Dask on Cloudera AI.	Distributed XGBoost with Dask on Cloudera AI

Configuring Traefik readTimeout for large file uploads

When uploading large files, the default readTimeout of 60 seconds in Traefik might not suffice. In such cases, you can configure Traefik to disable the default timeout.

1. Run the following Kubernetes command to edit the Traefik ConfigMap:

```
kubectl edit configmap traefik-config -n mlx
```

2. In the ConfigMap file, locate and modify the traefik.toml section to include the following custom `readTimeout`:

```
[entryPoints.websecure.transport]
     [entryPoints.websecure.transport.respondingTimeouts]
     readTimeout = 0
```

Note: The `readTimeout` value is set to 0 to indicate no timeout, ensuring uninterrupted uploads for large files.

For example, after adding the above changes, your traefik.toml file should look similar to:

```
# traefik.toml
[global]
  checkNewVersion = false
  sendAnonymousUsage = false
[entryPoints.web]
  address = ":80"
  [entryPoints.web.http.redirections.entryPoint]
 to = "websecure"
 scheme = "https"
[entryPoints.websecure]
  address = ":443"
 [entryPoints.websecure.transport]
   [entryPoints.websecure.transport.respondingTimeouts]
     readTimeout = 0
```

```
 [entryPoints.websecure.http.tls]
```

3. Save and apply the updated ConfigMap to your cluster for the changes to take effect.

```
kubectl rollout restart deployment/traefik -n mlx
```

Cloudera AI

Cloudera AI Architecture Overview

Date published: 2020-07-16
Date modified: 2024-11-21

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Architecture Overview **4**
Provisioning 4
Cloudera AI Architecture 4
ML Runtimes 5
Spark on Kubernetes 5
Autoscaling Workloads with Kubernetes 6
Autoscale Groups 7
Critical and Non-critical Pods 7

Architecture Overview

Cloudera AI is a Data Service on the Cloudera Data Platform. Residing on the Cloudera Data Platform helps Cloudera AI integrate with and leverage other Data Services residing on the platform, for example Cloudera Data Engineering and Cloudera Data Warehouse.

Provisioning

Cloudera AI utilizes the Cloudera Data Platform Control Plane to manage Data Services so you can provision and delete Cloudera AI Workbench. Cloudera Data Platform Control Plane leverages cloud native capabilities to dynamically access CPU, memory, and GPU resources along with cloud-managed Kubernetes (K8s) to provide the infrastructure.

During provisioning the Cloudera AI application is configured to authenticate end users of the service (Cloudera AI) via the Cloudera Data Platform identity provider, which is “chained” back to the customer’s identity provider. As a result, Cloudera AI provisioned instances allow for seamless customer SSO.

When you provision a Cloudera AI Workbench, the following happens:

- Cloudera Data Platform Control Plane performs the following in the cloud environment:
  - Requests a TLS Certificate and domain name with the cloudera.site domain
  - Identifies the SSO configuration
  - Identifies the SDX configuration for the environment
- Cloudera Data Platform Control Plane provisions a managed Kubernetes cluster
- Cloudera Data Platform Control Plane installs Cloudera AI into the Kubernetes environment
- Storage is mounted directly via managed service providers

Cloudera AI uses the cloud provider load balancer and networking infrastructure to partition the resources. Cloudera AI also leverages the cloud provider infrastructure to enable the customer to specify autoscaling.

Cloudera AI provisions the DNS and the certificate. Cloudera AI renews the certificates for the customer on an ongoing basis.

Cloudera AI Architecture

Once a Cloudera AI Workbench is provisioned, you can start using Cloudera AI for your end-to-end Machine Learning workflow.

Cloudera AI is a three-tier application that consists of a presentation tier, an application tier and a data tier.

Web tier

Cloudera AI is a web application that provides a UI that simplifies the action of managing workloads and resources for data scientists. It offers users a convenient way to deploy and scale their analytical pipeline and collaborate with their colleagues in a secure compartmentalized environment.

Cloudera AI communicates using HTTPS, Websocket, and gRPC. External communication is limited to HTTP and Websocket for the web UI and APIs. In-cluster service-to-service communication uses gRPC and is encrypted and mutually authenticated using TLS (Transport Layer Security).

Application tier

The application tier uses an actual set of workloads that users are running. These workloads are isolated in Kubernetes namespaces and run on specially marked compute nodes. Kubernetes/node level auto scaling is used to expand/contract the cluster size based on user demand.

User code gets baked into Docker images via a set of Source-to-Image pods (S2I), which includes a managing server, a queue for builds, a registry that serves the images for Docker, a git server, and the builders that actually perform the image building. Traditionally these images used the host machine's docker, but Cloudera AI switched to in-container builds for security and compliance on some platforms.

Data tier

Cloudera AI uses an internal Postgres database for persisting metadata of user workloads such as Sessions, Jobs, Models and Applications, which runs as a pod and is backed by a persistent volume, using the cloud-provider's block storage offering (for example, EBS for AWS and Premium_LRS for Azure).

Cloudera AI uses an NFS server, i.e. a POSIX-compliant file system, for storing users' project files, which include user code, libraries installed into a Session, and small data files. For AWS, Cloudera AI creates an Elastic File System (EFS) file system when provisioning the workbench for storing project files. For Azure, users can either provide an NFS volume created in Azure NetApp Files or an external NFS server when provisioning Cloudera AI Workbench for storing project files. This NFS server is supplied to the Cloudera AI Workbenches as Kubernetes Persistent Volumes (PVs). Persistent Volume Claims (PVCs) are set up per-namespace, and each user gets their own namespace - thus each user's view of the NFS server is limited to that exposed by the Cloudera Private Cloud.

ML Runtimes

ML Runtimes are responsible for running data science workloads and intermediating access to the underlying cluster.

Cloudera AI allows you to run any code via an interactive session, scheduled job, or deployed model or application. Data Scientists can use interactive sessions to explore data, or develop a model. They can create jobs and schedule them to run at specified times or productionize their work as a model to provide a REST endpoint or as an application that offers an interactive data dashboard for business users. All of these workloads run inside an ML Runtimes container on top of Kubernetes.

Cloudera ML Runtimes are purpose built to serve a specific use-case. They are available with a single editor (for example, Workbench, Jupyterlab), ship a single language Kernel (for example, Python 3.8 or R 4.0), and have a set of UNIX tools and utilities or language libraries and packages.

ML Runtimes have been open sourced and are available in the cloudera/ml-runtimes GitHub repository. If you need to understand your Runtime environments fully or want to build a new Runtime from scratch, you can access the Dockerfiles that were used to build the ML Runtimes container images in this repository.

There is a wide range of supported Runtimes out-of-the-box that cover the large majority of Data Science use-cases, but any special requirements can be satisfied by building a custom ML Runtimes container image.

Cloudera AI also supports quota management for CPU, GPU, and memory to limit the amount of resources users have access to within the Cloudera AI Workbench.

Before ML Runtimes, Cloudera AI offered similar functionalities via Legacy Engines. These deprecated container images followed a monolithic architecture, all kernels (Python, R, Scala), and all seemingly useful packages and libraries were included in the image.

Spark on Kubernetes

In Cloudera AI, multiple Spark versions are available through Runtime Addons. Data Scientists can select the version of Spark to be configured for any workload. Cloudera AI configures the Runtime container and mounts all of the dependencies.

Cloudera AI supports fully-containerized execution of Spark workloads through Spark's support for the Kubernetes cluster backend. Users can interact with Spark both interactively and in batch mode. In both batch and interactive modes, dependency management, including for Spark executors, is transparently managed by Cloudera AI and Kubernetes. No extra configuration is required. In interactive mode, Cloudera AI leverages the cloud provider for scalable project storage, and in batch mode, Cloudera AI manages dependencies through container images.

Cloudera AI also supports native cloud autoscaling via Kubernetes. When clusters do not have the required capacity to run workloads, they can automatically scale up additional nodes. Administrators can configure auto-scaling upper limits, which determine how large a compute cluster can grow. Since compute costs increase as cluster size increases, having a way to configure upper limits gives administrators a method to stay within a budget. Autoscaling policies can also account for heterogeneous node types such as GPU nodes.

In Cloudera AI, each project is owned by a user or team. Users can launch multiple sessions in a project. Workloads are launched within a separate Kubernetes namespace for each user, thus ensuring isolation between users at the Kubernetes level.

Autoscaling Workloads with Kubernetes

Kubernetes dynamically resizes clusters by using the `Kubernetes Cluster Autoscaler` (on Amazon EKS) or `cluster-autoscaler` (on Azure). The cluster autoscaler changes the desired capacity of an autoscaling group to expand or contract a cluster based on pod resource requests.

Scaling Up

The primary trigger for scaling up (or expanding) an autoscaling group is failure by the Kubernetes pod scheduler to find a node that meets the pod’s resource requirements. In Cloudera AI, if the scheduler cannot find a node to schedule an engine pod because of insufficient CPU or memory, the engine pod will be in “pending” state. When the autoscaler notices this situation, it will change the desired capacity of the autoscaling group (CPU or GPU) to provision a new node in the cluster. As soon as the new node is ready, the scheduler will place the session or engine pod there. In addition to the engine pod, certain Cloudera AI daemonset pods will also be scheduled on the new node.

The time taken to schedule an engine pod on a new node depends on the amount of time the autoscaler takes to add a new node into the cluster, plus time taken to pull the engine’s Docker image to the new node.

Scaling Down

A cluster is scaled down by the autoscaler by removing a node, when the resource utilization on the given node is less than a pre-defined threshold, provided the node does not have any non-evictable pods running on it. This threshold is currently set to 20% CPU utilization. That is, a node is removed if the following criteria are met:

- The node does not have non-evictable pods
- The node's CPU utilization is less than 20%
- The number of active nodes in the autoscaling group is more than the configured minimum capacity

It is possible that certain pods might be moved from the evicted node to some other node during the down-scaling process.

Note: By default, on AWS and Azure, autoscaling groups can include a maximum of 30 nodes. If more nodes are needed, contact your Cloudera representative.

Limitations on Azure

On Azure, there are some specific limitations to how autoscaling works.

- CPU nodes cannot scale down to zero. You can only have one or more CPU nodes.
- Autoscaling down is sometimes blocked by Azure services. You can check the cluster autoscaler logs to see if this is occurring.

Autoscale Groups

A Cloudera AI workbench or cluster consists of three different autoscaling groups: “infra”, “cpu” and “gpu”. These groups scale independently of one another.

Infra Autoscaling Group

The Infra autoscaling group is created automatically when a user provisions a Cloudera AI cluster, and is not configurable from the UI. This group is meant to run the core Cloudera AI services that are critical to the overall functioning of the workbench. This group is loosely analogous to the master node of legacy CDSW, however it can scale up or down if necessary. The instance count for this group ranges from 1 to 3, with the default set to 2. The instance type used for the group is m5.2xlarge on AWS, and Standard DS4 v2 on Azure. On AWS, this group is restricted to a single availability zone (AZ) because it relies on Elastic Block Store (EBS).

CPU Autoscaling Group

The CPU autoscaling group forms the main worker nodes of a Cloudera AI cluster, and is somewhat configurable from the UI. The user can choose from three different instance types, and can also set the autoscaling range from 0 to 30 CPU worker nodes. This group is meant to run general CPU-only workloads. This group is available in multiple availability zones.

GPU Autoscaling Group (not available on Azure)

The GPU autoscaling group consists of instances that have GPUs, and are meant for workloads that require GPU processing. Like the CPU group, this group is configurable from the UI. Unlike the CPU group, this group is meant exclusively for sessions that request > 0 GPUs, and are therefore fenced off from CPU-only workloads, in part because GPU instance types are much more expensive than regular instance types. This group is available in multiple availability zones.

Critical and Non-critical Pods

The pods running various Cloudera AI services and jobs broadly fall into critical and non-critical types.

Critical pods are protected from preemption by autoscaling to avoid interrupting important services. Most of the pods running in the “infra” autoscaling group are critical. Pods that run user sessions, such as engine pods and Spark executor pods, are also considered critical, and are marked as not safe to evict. Cloudera AI system services that are deployed as daemonsets (they run on all nodes in the cluster) are deemed important, but not critical. These pods are marked as “safe-to-evict” by autoscaling.

Cloudera AI

Command Line Reference

Date published: 2020-07-16
Date modified: 2024-11-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Command Line Tools in Cloudera AI....4

cdswctl Command Line Interface Client.... 4

Download and configure cdswctl....4
(Optional) Generate an SSH public or private key....5
Download cdswctl and add an SSH key....5
Initialize an SSH Endpoint....5
Log into cdswctl.... 7
Prepare to manage models using the model CLI....7
Create a model using the CLI....8
Build and deployment commands for models.... 10
Deploy a new model with updated resources.... 11
View replica logs for a model.... 11
Using ML Runtimes with cdswctl.... 11
Querying the engine type.... 12
Listing runtimes.... 12
Starting sessions and creating SSH endpoints.... 13
Creating a model....14

cdswctl command reference....14

Azure NetApp files management with the CLI....15

Command Line Tools in Cloudera AI

Cloudera AI ships with the following command line tools. The purpose of each tool differs.

- Cloudera Data Platform CLI for Cloudera AI - If you prefer to work in a terminal window, you can download and configure the Cloudera Data Platform client that gives you access to the Cloudera Data Platform CLI tool. The Cloudera Data Platform CLI allows you to perform the same actions as can be performed from the management console. Use this CLI to create, delete, upgrade, and manage Cloudera AI Workbenches on Cloudera Data Platform.

  To view all the available commands, run:

  ```
  cdp ml help
  ```

  To view help for a specific command, run:

  ```
  cdp ml <operation> help
  ```

  If you do not have the Cloudera Data Platform CLI set up yet, see *Installing the Cloudera Data Platform CLI Client*.
- cdswctl - Cloudera AI also ships with a CLI client that you can download from the Cloudera AI web UI. This is also referred to as the Model CLI client. The cdswctl client allows you to log in, create an SSH endpoint, launch new sessions, automate model deployment, model updates, and so on.

cdswctl Command Line Interface Client

Cloudera AI ships with a CLI client that you can download from the Cloudera AI web UI.

The cdswctl client allows you to perform the following tasks:

- Logging in
- Creating an SSH endpoint
- Listing sessions that are starting or running
- Starting or stopping a session
- Creating a model
- Building and deploying models
- Listing model builds and model deployments
- Checking the status of a deployment
- Redeploying a model with updated resources
- Viewing the replica logs for a model

Other actions, such as creating a project, require you to use the Cloudera AI web UI. For information about the available commands, run the following command:

```
cdswctl --help
```

Download and configure cdswctl

This topic describes how to download the cdswctl CLI client and configure your SSH public key to authenticate CLI access to sessions.

About this task

Before you begin, ensure that the following prerequisites are met:

- You have an SSH public/private key pair for your local machine.
- You have Contributor permissions for an existing project. Alternatively, create a new project you have access to.
- If you want to configure a third-party editor, make sure the Site Administrator has not disabled remote editing for Cloudera AI.

(Optional) Generate an SSH public or private key

Consider these optional instructions if you do not have SSH public or private key yet.

About this task

This task is optional. If you already have an SSH public/private key pair, skip this task. The steps to create an SSH public or private key pair differ based on your operating system. The following instructions are meant to be an example and are written for macOS using ssh-keygen.

Procedure

1. Open Terminal.
2. Run the following command and complete the fields:

   ```
   ssh-keygen -t rsa -f ~/.ssh/id_rsa
   ```

   Keep the following guidelines in mind:

   - Make sure that the SSH key you generate meets the requirements for the local IDE you want to use. For example, PyCharm requires the -m PEM option because PyCharm does not support modern (RFC 4716) OpenSSH keys.
   - Provide a passphrase when you generate the key pair. Use this passphrase when prompted for the SSH key passphrase.
   - Save the SSH key to the default ~/.ssh location.

Download cdswctl and add an SSH key

Follow the instructions to download cdswctl and to add an SSH key.

Procedure

1. Download cdswctl client for your operating system.

   Unpack it, and optionally, you can add it to the *PATH* environment variable on your system.
2. Open the Cloudera AI web UI and go to User Settings Remote Editing for your user account.
3. Add your SSH public key to SSH public keys for session access.

   Cloudera AI uses the SSH public key to authenticate your CLI client session, including the SSH endpoint connection to the Cloudera AI deployment.

   Any SSH endpoints that are running when you add an SSH public key must also be restarted.

Initialize an SSH Endpoint

This topic describes how to establish an SSH endpoint for Cloudera AI.

About this task

Creating an SSH endpoint is also the first step to configuring a remote editor for an Cloudera AI Workbench.

Procedure

1. Create a configuration file at: $HOME/.cdsw/config.yaml. The contents of config.yaml should be:

```
username: <USERNAME>
url: <ML_WORKBENCH_URL>
auth:
  authtype: 1
  basic: null
  apikey: <YOUR_LEGACY_API_KEY>
```

   To collect the values for these fields, first log in to your Cloudera AI Workbench using SSO:

   - username: The username with which you are logged into the Cloudera AI Workbench. Found in the top right corner of your Cloudera AI Workbench.
   - url: The complete URL used to access the Cloudera AI Workbench. For example: https://ml-<randomly-generated-cluster-name>
   - apikey: Go to User Settings API Keys . Copy the value of the Legacy API Key to this field.

2. Create a local SSH endpoint to Cloudera AI. Run the following command:

```
cdswctl ssh-endpoint -p <project_name> [-c <CPU_cores>] [-m <memory_in_GB>] [-g <number_of_GPUs>] [-r <runtime ID> ]
```

   If the project is configured to use ML Runtimes, the -r parameter must be specified, otherwise it must be omitted. See *Using ML Runtimes with cdswctl* documentation page for more information.

   The command uses the following defaults for optional parameters:

   - CPU cores: 1
   - Memory: 1 GB
   - GPUs: 0

   For example, the following command starts a session for the user milton under the customerchurn project with .5 cores, .75 GB of memory, 0 GPUs, and the Python3 kernel:

```
cdswctl ssh-endpoint -p customerchurn -c 0.5 -m 0.75
```

   To create an SSH endpoint in a project owned by another user or a team, for example finance, prepend the username to the project and separate them with a forward slash:

```
cdswctl ssh-endpoint -p finance/customerchurn -c 0.5 -m 0.75
```

   This command creates session in the project customerchurn that belongs to the team finance.

   Information for the SSH endpoint appears in the output:

```
...
You can SSH to it using
    ssh -p <some_port> cdsw@localhost
...
```

3. Open a new command prompt and run the outputted command from the previous step:

```
ssh -p <some_port> cdsw@localhost
```

For example:

```
ssh -p 9750 cdsw@localhost
```

You will be prompted for the passphrase for the SSH key you entered in the Cloudera AI Workbench web UI.

Once you are connected to the endpoint, you are logged in as the cdsw user and can perform actions as though you are accessing the terminal through the web UI.

4. Test the connection.

If you run ls, the project files associated with the session you created are shown. If you run whoami, the command returns the cdsw user.

5. Leave the SSH endpoint running as long as you want to use a local IDE.

Related Information

Using ML Runtimes with cdswctl

Log into cdswctl

This topic describes how to log into cdswctl.

Procedure

1. Open the Model CLI client.
2. Run the following command while specifying the actual values for the variables:

```
cdswctl login -u <WORKBENCH_URL> -n <USERNAME> -y <LEGACY_API_KEY>
```

where

- workbench_*URL* is the workbench URL including the protocol (http(s)://domain.com)
- *USERNAME* is your user name on the workbench
- *LEGACY_API_KEY* is the API key that you can obtain from the Cloudera AI UI. Go to Settings API Keys and copy the Legacy API Key (and not the API Key).

To see more information about the login command parameters, run

```
cdswctl login --help
```

If all goes well, then "Login succeeded" displays.

Prepare to manage models using the model CLI

Before you can start using the model CLI to automate model deployment or to perform any other tasks, you must install the scikit-learn machine learning library for Python through the Cloudera AI web UI.

About this task

You must perform this task through the Cloudera AI web UI.

Procedure

1. Create a new project with Python through the web UI.

   Python provides sample files that you can use to create models using CLI.
2. To start a new session, go to the **Sessions** page from the left navigation panel and click new session.

   The **Start the new session** page is displayed.
3. On **Start the new session** page, select Python 3 from the Engine Kernel drop-down menu, and click Launch Session.

   A new “Untitled Session” is created.
4. From the input prompt, install the scikit-learn machine learning library for Python by running the following command:

   ```
   !pip3 install sklearn
   ```
5. Open the fit.py file available within your project from the left navigation panel.

   You can use the fit.py file to create a fitted model which creates a model.pkl file that you can use to deploy the actual model.
6. Run the fit.py file by clicking Run Run all .

   The model.pkl directory is created that you can see within your project on the left navigation pane.
7. Close the session by clicking Stop.

Create a model using the CLI

Follow the instructions on how to create models using the model CLI.

Procedure

1. Open a terminal window and log into cdswctl.
2. Obtain the project ID as described in the following steps:
   a) Run the following command:

      ```
      cdswctl projects list
      ```

      The project ID, your username, and the project name are displayed. For example:

      1: john-smith/petal-length-predictor

   b) Note the project ID, which is a number in front of your project name.

      In this case, it is "1".

3. Run the following command while specifying the project name and note the engine image ID:

   Note: The following examples are specific to projects configured to use legacy engines and projects configured to use runtimes. Be sure to use the commands appropriate to your project configuration.

   For projects configured to use legacy engines:

   ```
   cdswctl engine-images list -p <PROJECT-NAME>
   ```

   For example,

   ```
   cdswctl engine-images list -p john-smith/petal-length-predictor
   ```

   For projects configured to use runtimes:

   ```
   cdswctl runtimes list
   ```

   Depending on your local setup, you may get a more readable output by post-processing the result with the following command:

   ```
   cdswctl runtimes list | python3 -m json.tool
   ```

   For this example you should pick a runtime with a Python kernel and Workbench editor. Depending on your local setup, you may filter the results using the following command:

   ```
   cdswctl runtimes list | jq  '.runtimes[] | select((.editor == "Workbench")
    and (.kernel | contains("Python")))'
   ```

4. Create a model by using the following command:

   Note: The following examples are specific to projects configured to use legacy engines and projects configured to use runtimes. Use the commands appropriate to your project configuration.

   For projects configured to use legacy engines:

   ```
   cdswctl models create
   --kernel="python3"
   --targetFilePath="predict.py"
   --targetFunctionName="predict"
   --name="Petal Length Predictor"
   --cpuMillicores=1000
   --memoryMb=2000
   --description="Model of the Iris dataset"
   --replicationType=fixed
   --numReplicas=1
   --visibility="private"
   --autoBuildModel
   --autoDeployModel
   --projectId=<PROJECT ID>
   --examples='{"request":{"petal_length":1}}'
   --engineImageId=<ENGINE IMAGE ID FROM BEFORE>
   ```

   For projects configured to use runtimes:

   ```
   cdswctl models create
   --targetFilePath="predict.py"
   --targetFunctionName="predict"
   --name="Petal Length Predictor"
   --cpuMillicores=1000
   --memoryMb=2000
   --description="Model of the Iris dataset"
   ```

```
--replicationType=fixed
--numReplicas=1
--visibility="private"
--autoBuildModel
--autoDeployModel
--projectId=<project ID>
--examples='{"request":{"petal_length":1}}'
--runtimeId=<runtime ID obtained above>
```

If the command runs successfully, the system displays the model details in a JSON format.

5. For more information about the models create command parameters, run the following command:

```
cdswctl models create --help
```

Build and deployment commands for models

Models have separate parameters for builds and deployments. When a model is built, an image is created. Whereas, the deployment is the actual instance of the model. You can list model builds and deployment, and monitor their state using from model CLI client (cdswctl).

Listing a model

To list the models, run the following command:

```
cdswctl models list
```

Monitoring the status of the model

To monitor the status of the build for a particular model, use the following command:

```
cdswctl models listBuild --modelId <MODEL_ID> --projectId <PROJECT_ID>
```

You can use the --latestModelDeployment flag to get the build for the latest deployment.

Listing a deployment

To list the deployment for a particular model, run the following command:

```
cdswctl models listDeployments --modelId <MODEL_ID>
```

Checking the status of a deployment

To check the status of your deployment, run the following command:

```
cdswctl models listDeployments --statusSet=deployed
```

Following is a list of arguments for the statusSet parameter:

- deployed
- deploying
- failed
- pending
- stopping
- stopped

Note: You can use the parameter more than once in a command to check multiple statuses of your deployed models. For example,

```
cdswctl models listDeployments --statusSet=deployed --statusSet=stop
ped --statusSet=failed
```

Deploy a new model with updated resources

You can republish a previously-deployed model in a new serving environment with an updated number of replicas or memory/CPU/GPU allocation by providing the model build ID of the model you want to rebuild.

To deploy a new model, use the following command:

```
cdswctl models deploy --modelBuildId=<BUILD_ID> --cpuMilli
cores=<NUM_OF_CPU_CORES> --memoryMb=<MEMORY_IN_MB> --numRepli
cas=<NUM_OF_REPLICAS> --replicationType=<REPLICATION_TYPE>
```

For example:

```
cdswctl models deploy --modelBuildId=<BUILD_ID> --cpuMillicores=1200 --mem
oryMb=2200 --numReplicas=2 --replicationType=fixed
```

Note: You must specify values for all the non-zero resources, even if you do not wish to update their values. For example, in your existing deployment, if you set the cpuMillicores capacity to 1200 and you do not wish to increase or decrease it, you must still specify cpuMillicores=1200 in the command.

View replica logs for a model

When a model is deployed, Cloudera AI enables you to specify the number of replicas that must be deployed to serve requests. If a replica crashes or fails to come up, you can diagnose it by viewing the logs for every replica using the model CLI.

Procedure

1. Obtain the modelReplicaId by using the following command:

   ```
   cdswctl models listReplicas --modelDeploymentId=<MODEL_DEPLOYMENT_ID>
   ```

   where the *MODEL_DEPLOYMENT_ID* is the ID of a successfully deployed model.
2. To view the replica logs, run the following command:

   ```
   cdswctl models getReplicaLogs --modelDeploymentId=<MODEL_DEPLOYMENT_ID> --
   modelReplicaId="<REPLICA_ID>" --streams=stdout
   ```

   For example:

   ```
   cdswctl models getReplicaLogs --modelDeploymentId=2 --modelReplicaId="pe
   tal-length-predictor-1-2-6d6496b467-hp6tz" --streams=stdout
   ```

   The valid values for the streams parameter are stdout and stderr.

Using ML Runtimes with cdswctl

If a project is configured to use Runtimes, cdswctl workflows for starting sessions or models are slightly different.

Querying the engine type

You can query whether a project is configured using ML Runtimes or Legacy Engine.

Procedure

To determine if a project is configured to use either ML Runtimes or Legacy Engines, use the cdswctl projects get EngineType command and specify the project with the -p parameter.

For example, to determine if configured to use ML Runtimes:

```
cdswctl projects getEngineType -p demouser/runtimeproject
ml_runtime
```

```
cdswctl projects getEngineType -p demouser/legacyproject
legacy_engine
```

Listing runtimes

The first step to working with projects using runtimes is to query the available runtimes using the cdswctl runtimes list command.

About this task

The cdswctl runtimes list command returns all runtimes in a large JSON result. For easier consumption, you can post-process this result with some 3rd-party tool, such as jq or Python's json.tool.

Procedure

To query the available runtimes, use the cdswctl runtimes list command.

Note: The following examples are for presentation purposes only. Neither Python's json.tool nor jq are supported directly by Cloudera.

The following example pipes the cdswctl runtimes list result through Python's json.tool to produce a more readable output:

```
user@host:~ $ cdswctl runtimes list | python3 -m json.tool
{
    "runtimes": [
        {
            "id": 1,
            "imageIdentifier": "docker.repository.cloudera.com/cdsw/ml-run
time-workbench-python3.6-standard:2020.11.1-b6",
            "editor": "Workbench",
            "kernel": "Python 3.6",
            "edition": "Standard",
            "shortVersion": "2020.11",
            "fullVersion": "2020.11.1-b6",
            "maintenanceVersion": 1,
            "description": "Standard edition Python runtime provided by Cl
oudera"
        },
        {
            "id": 2,
            "imageIdentifier": "docker.repository.cloudera.com/cdsw/ml-run
time-jupyterlab-python3.7-standard:2020.11.1-b6",
            "editor": "JupyterLab",
            "kernel": "Python 3.7",
            "edition": "Technical Preview",
            "shortVersion": "2020.11",
            "fullVersion": "2020.11.1-b6",
```

```
            "maintenanceVersion": 1,
            "description": "Technical Preview JupyterLab Python runtime pro
vided by Cloudera"
        }
    ]
}
```

The following example pipes the cdswctl runtimes list result through jq to transform the JSON output into arbitrary formats:

```
user@host:~ $ cdswctl runtimes list | jq -r '.runtimes[] | "\(.id) \(.imageI
dentifier)"'
1
docker.repository.cloudera.com/cdsw/ml-runtime-workbench-python3.6-standa
rd:2020.11.1-b6
2
docker.repository.cloudera.com/cdsw/ml-runtime-jupyterlab-python3.7-standa
rd:2020.11.1-b6
```

The following example filters the cdswctl runtimes list result using jq to only show runtimes with specific editors and kernels:

```
user@host:~ $ cdswctl runtimes list | jq  '.runtimes[] | select((.editor ==
"Workbench") and (.kernel | contains("Python")))'

{
  "id": 1,
  "imageIdentifier": "docker.repository.cloudera.com/cdsw/ml-runtime-workben
ch-python3.6-standard:2020.11.1-b6",
  "editor": "Workbench",
  "kernel": "Python 3.6",
  "edition": "Standard",
  "shortVersion": "2020.11",
  "fullVersion": "2020.11.1-b6",
  "maintenanceVersion": 1,
  "description": "Standard edition Python runtime provided by Cloudera"
}
```

Starting sessions and creating SSH endpoints

Once you choose a runtime, you can start a session using the cdswctl sessions start command and create SSH endpoints using the cdswctl ssh-endpoint command.

About this task

The runtime ID used in the following steps is obtained using the steps outlined in *Listing runtimes*.

Procedure

1. To start a session with a runtime, use the cdswctl sessions start command, specifying the runtime ID with the -r parameter and the project with the -p parameter.

   For example:

   ```
   cdswctl sessions start -r 2 -p demouser/runtimeproject
   ```

2. To specify SSH endpoints for the runtimes sessions, use the cdswctl ssh-endpoint command and specify the runtime ID using the -r parameter. and the project with the -p parameter.

   For example:

```
cdswctl ssh-endpoint -r 1 -p demouser/runtimeproject
```

Creating a model

Creating a model in a project that uses runtimes is similar to model creation with an legacy engine, but you must use a different parameter to specify the runtime ID.

About this task

To create a model in a project that uses runtimes you must use the --runtimeId= parameter to specify a runtime ID (instead of using the --engineImageId= and --kernel= parameters used for a legacy engine).

Procedure

To create a model in a project that uses runtimes use the --runtimeId= parameter to specify a runtime ID.

For example:

```
cdswctl models create --targetFilePath=predict.py --targetFunctionName=predi
ct
  --projectId=4 --name=created-using-cdswctl --description=created-using-cds
wctl
  --memoryMb=1024 --authEnabled --cpuMillicores=250 --autoBuildModel --aut
oDeployModel
  --examples='{"request":{"petal_length":1}}' --runtimeId=1
```

cdswctl command reference

You can manage your Cloudera AI Workbench cluster with the CLI client (cdswctl) that exists within the Cloudera AI Workbench. Running `cdswctl` without any arguments prints a brief description of each command.

Table 1: Model CLI Command Reference

Command	Description and usage
cdswctl login	Enables you to log into the model CLI client
cdswctl projects list	Lists the projects
cdswctl models create	Creates a model with the specified parameters
cdswctl models list	Lists all models You can refine the search by specifying the modelId
cdswctl models listBuild	Lists the builds for a model You can monitor the status of the build by specifying the modelId and the projectId
cdswctl models listDeployments	List the deployments for a model You can refine the search by specifying the modelId Use the statusSet parameter to check the status of the model being deployed

Command	Description and usage
cdswctl models deploy	Deploys a model with the specified parameters
cdswctl models listReplicas	Enables you to view the list of model replicas You also need this information to obtain replica logs
cdswctl models getReplicaLogs	Enables you to view the logs for a model replica
cdswctl models restart	Restarts a model Usage: cdswctl models restart --modelDeployment Id=<*DEPLOYMENT_ID*> Note: Running this command does not change the resources if you previously ran the cdswctl models update command
`cdswctl models update`	Changes the name, description, or visibility of the model To change a model's resources, use the `cdswctl models deploy` command
`cdswctl models delete`	Deletes a model Usage: cdswctl models delete --id=<*MODEL_ID*>

Azure NetApp files management with the CLI

You can manage the NetApp Files setup using the CLI. This can be helpful for automating setup and teardown of workbenches as Cloudera AI project needs change.

Create an Azure NetApp Files account

The following code sample creates an Azure NetApp Files account.

```
 az netappfiles account create \ --account-name my-anf-account \ --resource-group
    my-cdp-resource-group \ --location westus2
```

Create a capacity pool

A capacity pool is a storage container for volumes, which are accessed directly by Cloudera AI. The minimum size for an Azure NetApp Files capacity pool is 4 TiB

```
 MINIMUM_POOL_SIZE=4 # 4 TiB is the minimum az netappfiles pool create \
    --account-name my-anf-account \ --pool-name my-anf-pool \ --resource-group my-cdp-resource-group
    \ --service-level Standard \ --location westus2 \ --size ${MINIMUM_POOL_SIZE}
```

Create a volume

Create one or more volumes in the capacity pool. The "Usage threshold" is referred to as the "quota" in the Azure web portal. It is measured in GiB. The volume must support the NFSv3 protocol (which is the default).

```
 az netappfiles volume create \ --account-name my-anf-account \ --pool-name
    my-anf-account \ --volume-name my-anf-volume \ --resource-group my-cdp-resource-group \
```

```
    --location westus2 \ --file-path my-anf-volume \ --usage-threshold 1000
\ --vnet my-cdp-vnet \
    --subnet my-anf-subnet \ --service-level Standard
```

The mount path for this volume, or a dedicated, empty subdirectory inside that volume, must be provided for the "Existing NFS" field when provisioning Cloudera AI Workbench. It can be found in the "Mount Instructions" blade of the volume in the Azure portal.

Since each capacity pool has a large minimum, and each volume requires a dedicated subnet, users may wish to have a single volume that is shared between workbenches. This can be managed by having a VM that has the Azure volume mounted (instructions for doing this are also in the "Mount Instructions" blade of the volume in the Azure portal). This VM can then be used to quickly manage directories for individual workbenches on a single, shared volume. For instance:

```
    USER=      # username for accessing management VM
    VM=        # IP address or hostname for accessing management VM
    VOLUME=    # NFS volume name
    WORKBENCH= # Cloudera AI workbench name (or other unique directory name)
    ssh ${USER}@${VM} "sudo mkdir ${VOLUME}/${workbench}; sudo chown 8536:8
536 ${VOLUME}/${WORKBENCH}"
    # ...
    ssh ${USER}@${VM} "sudo rm -r ${VOLUME}/${WORKBENCH}"
```

Cloudera AI

Distributed Computing

Date published: 2020-07-16
Date modified: 2025-01-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Distributed Computing with Workers 4
Workers API 4
Launch Workers 4
List Workers 5
Stop Workers 5
Worker Network Communication 5

Distributed Computing with Workers

Cloudera AI provides basic support for launching multiple engine instances, known as workers, from a single interactive session. Any R or Python session can be used to spawn workers. These workers can be configured to run a script (for example, a Python file) or a command when they start up.

Workers can be launched using the launch_workers function. Other supported functions are list_workers and stop _workers. Output from all the workers is displayed in the workbench console of the session that launched them. These workers are terminated when the session exits.

Using Workers for Cloudera AI

The simplest example of using this feature would involve launching multiple workers from a session, where each one prints 'hello world' and then terminates right after. To extend this example, you can remove the print command and configure the workers to run a more elaborate script instead. For example, you can set up a queue of parameters (inputs to a function) in your main interactive session, and then configure the workers to run a script that pulls parameters off the queue, applies a function, and keeps doing this until the parameter queue is empty. This generic idea can be applied to multiple real-world use-cases. For example, if the queue is a list of URLs and the workers apply a function that scrapes a URL and saves it to a database, Cloudera AI can easily be used to do parallelized web crawling.

Hyperparameter optimization is a common task in machine learning, and workers can use the same parameter queue pattern described above to perform this task. In this case, the parameter queue would be a list of possible values of the hyperparameters of a machine learning model. Each worker would apply a function that trains a machine learning model. The workers run until the queue is empty, and save snapshots of the model and its performance.

Workers API

This section lists the functions available as part of the workers API.

Launch Workers

Launches worker engines into the cluster.

Syntax

```
launch_workers(n, cpu, memory, nvidia_gpu=0, kernel="python3", s
cript="", code="", env={})
```

Parameters

- n (int) - The number of engines to launch.
- cpu (float) - The number of CPU cores to allocate to the engine.
- memory (float) - The number of gigabytes of memory to allocate to the engine.
- nvidia_gpu (int, optional) - The number of GPU's to allocate to the engine.
- kernel (str, optional) - The kernel. Can be "r", "python2", "python3" or "scala". This parameter is only available for projects that use legacy engines.
- script (str, optional) - The name of a Python source file the worker should run as soon as it starts up.
- code (str, optional) - Python code the engine should run as soon as it starts up. If a script is specified, code will be ignored.
- env (dict, optional) - Environment variables to set in the engine.

Example Usage

Python

```
import cdsw
```

```
workers = cdsw.launch_workers(n=2, cpu=0.2, memory=0.5, code="pr
int('Hello from a CDSW Worker')")
```

R

```
library("cdsw")
              workers <- launch.workers(n=2, cpu=0.2, memory=0.5,
 env="", code="print('Hello from a Cloudera AI Worker')")
```

Note: The env parameter has been defined due to a bug that appears when parsing the launch.workers function. When not defined, the env parameter is serialized internally into a format that is incompatible with Cloudera AI. This bug does not affect the Python engine.

List Workers

Returns all information on all the workers in the cluster.

Syntax

```
list_workers()
```

Stop Workers

Stops worker engines.

Syntax

```
stop_workers(*worker_id)
```

Parameter

- worker_id (int, optional) - The ID numbers of the worker engines that must be stopped. If an ID is not provided, all the worker engines on the cluster will be stopped.

Worker Network Communication

This section demonstrates some trivial examples of how two worker engines communicate with the master engine.

Workers are a low-level feature to help use higher level libraries that can operate across multiple hosts. As such, you will generally want to use workers only to launch the backends for these libraries.

To help you get your workers or distributed computing framework components talking to one another, every worker engine run includes an environmental variable CML_MASTER_IP with the fully addressable IP of the master engine. Every engine has a dedicated IP access with no possibility of port conflicts.

For instance, the following are trivial examples of two worker engines talking to the master engine.

R

From the master engine, the following master.r script will launch two workers and accept incoming connections from them.

```
# master.r

library("cdsw")
# Launch two Cloudera AI workers. These are engines that will run in
# the same project, run a given code or script, and exit.
workers <- launch.workers(n=2, cpu=0.2, memory=0.5, env="", script="worker.r
")

# Accept two connections, one from each worker. Workers will
```

```
# run worker.r.
for(i in c(1,2)) {
  # Receive a message from each worker and return a response.
  con <- socketConnection(host="0.0.0.0", port = 6000, blocking=TRUE, serv
er=TRUE, open="r+")
  data <- readLines(con, 1)
  print(paste("Server received:", data))
  writeLines("Hello from master!", con)
  close(con)
}
```

The workers will run the following worker.r script and respond to the master.

```
# worker.r

print(Sys.getenv("CML_MASTER_IP"))
con <- socketConnection(host=Sys.getenv("CML_MASTER_IP"), port = 6000, bl
ocking=TRUE, server=FALSE, open="r+")
write_resp <- writeLines("Hello from Worker", con)
server_resp <- readLines(con, 1)
print(paste("Worker received:  ", server_resp))
close(con)
```

Python

From the master engine, the following master.py script will launch two workers and accept incoming connections from them.

```
# master.py
import cdsw, socket

# Launch two CDSW workers. These are engines that will run in
# the same project, run a given code or script, and exit.
workers = cdsw.launch_workers(n=2, cpu=0.2, memory=0.5, script="worker.py")

# Listen on TCP port 6000
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("0.0.0.0", 6000))
s.listen(1)

# Accept two connections, one from each worker. Workers will
# run worker.py.
conn, addr = s.accept()
for i in range(2):
    # Receive a message from each worker and return a response.
    data = conn.recv(20)
    if not data: break
    print("Master received:", data)
    conn.send("Hello From Server!".encode())
conn.close()
```

The workers will run the following worker.py script and respond to the master.

```
# worker.py
import os, socket

# Open a TCP connection to the master.
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((os.environ["CDSW_MASTER_IP"], 6000))

# Send some data and receive a response.
s.send("Hello From Worker!".encode())
```

```
data = s.recv(1024)
s.close()

print("Worker received:", data)
```

Cloudera AI

Managing Engines in Cloudera AI

Date published: 2020-07-16
Date modified: 2024-11-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing Engines....4
Creating Resource profiles....4
Configuring the engine environment....5
Set up a custom repository location....6
Burstable CPUs....6

Installing additional packages....6
Using Conda to manage dependencies....8

Engine environment variables....9
Engine environment variables....10
Accessing environmental variables from projects....11

Customized engine images....12
Creating a customized engine image....12
Create a Dockerfile for the custom image....13
Build the new Docker image....13
Distribute the image....13
Including images in allowlist for Cloudera AI projects....14
Add Docker registry credentials....14
Limitations with customized engines....14
End-to-end example: MeCab....15

Pre-Installed Packages in engines....16
Base Engine 15-cml-2021.09-1....16
Base Engine 14-cml-2021.05-1....17
Base Engine 13-cml-2020.08-1....19
Base Engine 12-cml-2020.06-2....21
Base Engine 11-cml1.4....22
Base Engine 10-cml1.3....24
Base Engine 9-cml1.2....25

Managing Engines

This topic describes how to manage engines and configure engine environments to meet your project requirements.

Required Role: Site Administrator

Site administrators and project administrators are responsible for making sure that all projects on the deployment have access to the engines they need. Site admins can create engine profiles, determine the default engine version to be used across the deployment, and white-list any custom engines that teams require. As a site administrator, you can also customize engine environments by setting global environmental variables and configuring any files/folders that need to be mounted into project environments on run time.

By default, Cloudera AI ships a base engine image that includes kernels for Python, R, and Scala, along with some additional libraries (see *Configuring Cloudera AI Engines* for more information) that can be used to run common data analytics operations. Occasionally, new engine versions are released and shipped with Cloudera AI releases.

Engine images are available in the Site Administrator panel at Admin Engines , under the Engine Images section. As a site administrator, you can select which engine version is used by default for new projects. Furthermore, project administrators can explicitly select which engine image should be used as the default image for a project. To do so, go to the project's Overview page and click Settings on the left navigation bar.

If a user publishes a new custom Docker image, site administrators are responsible for white-listing such images for use across the deployment. For more information on creating and managing custom Docker images, see *Configuring the Engine Environment*.

Related Information

Configuring the Engine Environment

Installing Additional Packages

Creating Resource profiles

Resource profiles define how many vCPUs and how much memory the product will reserve for a particular workload (for example, session, job, model).

About this task

As a site administrator you can create several different vCPU, GPU, and memory configurations which will be available when launching a session/job. When launching a new session, users will be able to select one of the available resource profiles depending on their project's requirements.

Procedure

1. To create resource profiles, go to the Site Administration Runtime/Engine page.
2. Add a new profile under Resource Profiles.

   Cloudera recommends that all profiles include at least 2 GB of RAM to avoid out of memory errors for common user operations.

   You will see the option to add GPUs to the resource profiles only if your Cloudera AI hosts are equipped with GPUs, and you have enabled them for use by setting the relevant properties in cdsw.conf.

Results

If there are two worker nodes and 10 vCPU available overall, if one user tries to establish a session with 8 vCPU, CDSW will not allow it. The memory and CPU must be contiguous (adjacent to each other). When a user spins a

session, the pod triggers on a single node and resources on the same node are utilized. This is expected behavior for Kubernetes.

Figure 1: Resource profiles available when launching a session

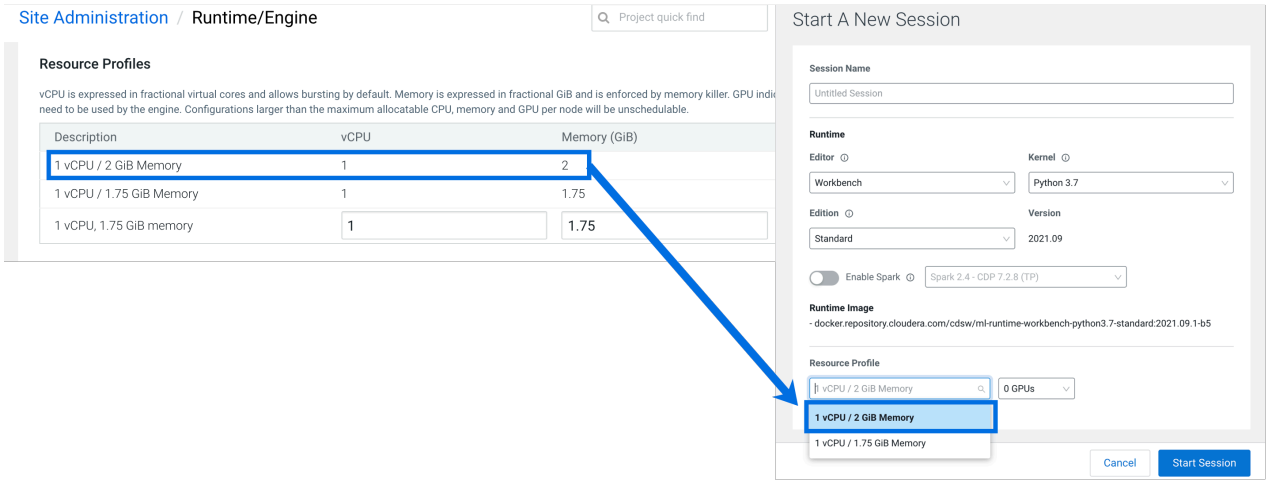

Configuring the engine environment

This section describes some of the ways you can configure engine environments to meet the requirements of your projects.

Install Additional Packages

For information on how to install any additional required packages and dependencies to your engine, see *Installing Additional Packages*.

Environmental Variables

For information on how environmental variables can be used to configure engine environments in Cloudera AI, see Engine Environment Variables.

Configuring Shared Memory Limit for Docker Images

You can increase the shared memory size for the sessions, experiments, and jobs running within an Engine container within your project. For Docker, the default size of the available shared memory is 64 MB.

To increase the shared memory limit:

1. From the web UI, go to Projects Project Settings Engine Advanced Settings
2. Specify the shared memory size in the Shared Memory Limit field.
3. Click Save Advanced Settings to save the configuration and exit.

This mounts a volume with the tmpfs file system to /dev/shm and Kubernetes will enforce the given limit. The maximum size of this volume is the half of your physical RAM in the node without the swap.

Related Information

Engine Environment Variables

Installing additional packages

Set up a custom repository location

You can set up a custom default location for Python and R code package repositories. This is especially useful for air-gapped clusters that are isolated from the PIP and CRAN repositories on the public internet.

Python PIP repository

Custom PIP repositories can be set as default for all engines at a site or project level. The environmental variables can be set at the Project or Site level. If the values are set at the Site level, they can be overridden at the Project level.

1. Set the environmental variables at the appropriate level.
   - For Site level, go to: Site Administration Engine
   - For Project level, go to: Project Settings Engine
2. To set a new default URL for the PIP index, enter:
   - PIP_INDEX_URL = <new url>
   - PIP_EXTRA_INDEX_URL = <new url>

CRAN repository

Custom CRAN repositories must be set in a session or as part of a custom engine. To set a new default URL for a CRAN repository, set the following in the /home/cdsw/.Rprofile file:

```
options(repos=structure(c(CRAN="<mirror URL>")))
```

Burstable CPUs

Cloudera AI configures no upper bound on the CPU resources that Workloads can use so that they can use all of the CPU resources available on the node where they are running. By configuring no CPU limits, Cloudera AI enables efficient use of the CPU resources available on your cluster nodes.

- If the CPUs are idle then the workloads can burst and take advantage of the free CPU cycles. For example, if you have launched a session with 1vCPU but the code inside it requires more than 1vCPU, the workload container can consume all the available CPU cycles on the node where it is launched.
- When the cluster is highly utilized and CPU resources are sparse, Workloads will be limited to use the number of CPU resources configured in their resource profile.
- If multiple containers are attempting to use excess CPU, CPU time is distributed in proportion to the amount of CPU initially requested by each container.

Installing additional packages

Cloudera AI engines are preloaded with a few common packages and libraries for R, Python, and Scala. However, a key feature of Cloudera AI is the ability of different projects to install and use libraries pinned to specific versions, just as you would on your local computer.

Note: Before downloading or using third-party content, you are responsible for reviewing and complying with any applicable license terms and making sure that they are acceptable for your use case.

Generally, Cloudera recommends you install all required packages locally into your project. This will ensure you have the exact versions you want and that these libraries will not be upgraded when Cloudera upgrades the base engine image. You only need to install libraries and packages once per project. From then on, they are available to any new engine you spawn throughout the lifetime of the project.

You can install additional libraries and packages from the workbench, using either the command prompt or the terminal.

Note:

Cloudera AI does not currently support installation of packages that require root access to the hosts. For such use-cases, you will need to create a new custom engine that extends the base engine image to include the required packages. For instructions, see *Creating a Customized Engine Image*.

(Python and R) install packages using Workbench command prompt

To install a package from the command prompt:

1. Navigate to your project's Overview page. Click Open Workbench and launch a session.
2. At the command prompt (see Native Workbench Console and Editor) in the bottom right, enter the command to install the package. Some examples using Python and R have been provided.

R

```
# Install from CRAN
install.packages("ggplot2")

# Install using devtools
install.packages('devtools')
library(devtools)
install_github("hadley/ggplot2")
```

Python 2

```
# Installing from console using ! shell operator and pip:
!pip install beautifulsoup

# Installing from terminal
pip install beautifulsoup
```

Python 3

```
# Installing from console using ! shell operator and pip3:
!pip3 install beautifulsoup4
# Installing from terminal
pip3 install beautifulsoup4
```

(Python only) using a requirements file

For a Python project, you can specify a list of the packages you want in a requirements.txt file that lives in your project. The packages can be installed all at once using pip/pip3.

1. Create a new file called requirements.txt file within your project:

   ```
   beautifulsoup4==4.6.0
   seaborn==0.7.1
   ```

2. To install the packages in a Python 3 engine, run the following command in the workbench command prompt.

   ```
   !pip3 install -r requirements.txt
   ```

   For Python 2 engines, use pip.

   ```
   !pip install -r requirements.txt
   ```

Related Information

Conda

Using Conda to manage dependencies

You can install additional libraries and packages from the workbench, using either the command prompt or the terminal. Alternatively, you might choose to use a package manager such as Conda to install and maintain packages and their dependencies. This topic describes some basic usage guidelines for Conda.

Cloudera AI recommends using pip for package management along with a requirements.txt file (as described in the previous section). However, for users that prefer Conda, the default engine in Cloudera AI includes two environments called python2.7, and python3.6. These environments are added to sys.path, depending on the version of Python selected when you launch a new session.

In Python 2 and Python 3 sessions and attached terminals, Cloudera AI automatically sets the CONDA_DEFAUL T_ENV and CONDA_PREFIX environment variables to point to Conda environments under /home/cdsw/.conda.

However, Cloudera AI does not automatically configure Conda to pin the actual Python version. Therefore if you are using Conda to install a package, you must specify the version of Python. For example, to use Conda to install the feat her-format package into the python3.6 environment, run the following command in the Workbench command prompt:

```
!conda install -y -c conda-forge python=3.6.9 feather-format
```

To install a package into the python2.7 environment, run:

```
!conda install -y -c conda-forge python=2.7.17 feather-format
```

Note that on sys.path, pip packages have precedence over conda packages.

Note:

- Cloudera AI does not automatically configure a Conda environment for R and Scala sessions and attached terminals. If you want to use Conda to install packages from an R or Scala session or terminal, you must manually configure Conda to install packages into the desired environment.

Creating an extensible engine with Conda

Cloudera AI also allows you to configure the engine environment, to include packages of your choice using Conda. To create an extended engine:

1. Add the following lines to a Dockerfile to extend the base engine, push the engine image to your Docker registry, and include the new engine in the allowlist, for your project. For more details on this step, see *Configuring the engine environment*.

   Python 2

   ```
   RUN mkdir -p /opt/conda/envs/python2.7
   RUN conda install -y nbconvert python=2.7.17 -n python2.7
   ```

   Python 3

   ```
   RUN mkdir -p /opt/conda/envs/python3.6
   RUN conda install -y nbconvert python=3.6.9 -n python3.6
   ```

2. Set the PYTHONPATH environmental variable as shown below. You can set this either globally in the site administrator dashboard, or for a specific project by going to the project's Settings Engine page.

   Python 2

   ```
   PYTHONPATH=$PYTHONPATH:/opt/conda/envs/python2.7/lib/python2.7/site-pack
   ages
   ```

   Python 3

   ```
   PYTHONPATH=$PYTHONPATH:/opt/conda/envs/python3.6/lib/python3.6/site-pack
   ages
   ```

Related Information

Conda

Configuring the engine environment

Engine environment variables

This topic describes how engine environmental variables work. It also lists the different scopes at which they can be set and the order of precedence that will be followed in case of conflicts.

Environmental variables allow you to customize engine environments for projects. For example, if you need to configure a particular timezone for a project, or increase the length of the session/job timeout windows, you can use environmental variables to do so.

Cloudera AI allows you to define environmental variables for the following scopes:

Global

A site administrator for your Cloudera AI deployment can set environmental variables on a global level. These values will apply to every project on the deployment.

To set global environmental variables, go to Admin Runtime/Engines .

Project

Project administrators can set project-specific environmental variables to customize the engines launched for a project. Variables set here will override the global values set in the site administration panel.

To set environmental variables for a project, go to the project's Overview page and click Settings Advanced .

Job

Environments for individual jobs within a project can be customized while creating the job. Variables set per-job will override the project-level and global settings.

To set environmental variables for a job, go to the job's Overview page and click Settings Set Environmental Variables .

Experiments

Engines created for execution of experiments are completely isolated from the project. However, these engines inherit values from environmental variables set at the project-level and/or global level. Variables set at the project-level will override the global values set in the site administration panel.

Models

Model environments are completely isolated from the project. Environmental variables for these engines can be configured during the build stage of the model deployment process. Models will also

inherit any environment variables set at the project and global level. However, variables set per-model build will override other settings.

Engine environment variables

The following table lists Cloudera AI environment variables that you can use to customize your project environments. These can be set either as a site administrator or within the scope of a project or a job.

Environment Variable	Description
MAX_TEXT_LENGTH	Maximum number of characters that can be displayed in a single text cell. By default, this value is set to 800,000 and any more characters will be truncated. Default: 800,000
PROJECT_OWNER	The name of the Team or user that created the project.
SESSION_MAXIMUM_MINUTES	Maximum number of minutes a session can run before it times out. Default: 60*24*7 minutes (7 days) Maximum Value: 35,000 minutes
JOB_MAXIMUM_MINUTES	Maximum number of minutes a job can run before it times out. Default: 60*24*7 minutes (7 days) Maximum Value: 35,000 minutes
IDLE_MAXIMUM_MINUTES	Maximum number of minutes a session can remain idle before it exits. Default: 60 minutes Maximum Value: 35,000 minutes Idle timeouts for sessions vary by workbench type (runtime). • Standard Workbench: Sessions timeout regardless of activity in the browser or terminal. • PBJ Workbench: Sessions timeout if there is no browser activity and no terminal window is open. If a terminal window is open, the session will not timeout, regardless of whether there is activity in the terminal window. • Jupyterlab: Sessions timeout if there is no browser activity. Terminal window activity is not considered. • Custom runtimes: No idle timeout behavior is enforced on custom or third-party workbenches.
CONDA_DEFAULT_ENV	Points to the default Conda environment so you can use Conda to install/manage packages in the Workbench. For more details on when to use this variable, see *Installing Additional Packages*.

Per-Engine Environmental Variables: In addition to the previous table, there are some more built-in environmental variables that are set by the Cloudera AI application itself and do not need to be modified by users. These variables are set per-engine launched by Cloudera AI and only apply within the scope of each engine.

Environment Variable	Description
CDSW_PROJECT	The project to which this engine belongs.
CDSW_PROJECT_ID	The ID of the project to which this engine belongs.
CDSW_ENGINE_ID	The ID of this engine. For sessions, this appears in your browser's URL bar.
CDSW_MASTER_ID	If this engine is a worker, this is the CDSW_ENGINE_ID of its master.
CDSW_MASTER_IP	If this engine is a worker, this is the IP address of its master.

Environment Variable	Description
CDSW_PUBLIC_PORT	**Note:** This property is deprecated. See CDSW_APP_PORT and CDSW_READONLY_PORT for alternatives. A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_PUBLIC_PORT will be available in browsers at: http(s)://read-only-<*$CDSW_ENGINE_ID*>.<*$CDSW_DOMAIN*>. By default, CDSW_PUBLIC_PORT is set to 8080. A direct link to these web services will be available from the grid icon in the upper right corner of the Cloudera AI web application, as long as the job or session is still running. For more details, see *Accessing Web User Interfaces from Cloudera AI*. In Cloudera AI, setting CDSW_PUBLIC_PORT to a non-default port number is not supported.
CDSW_APP_PORT	A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_APP_PORT will be available in browsers at: http(s)://read-only-<*$CDSW_ENGINE_ID*>.<*$CDSW_DOMAIN*>. Use this port for applications that grant some control to the project, such as access to the session or terminal. A direct link to these web services will be available from the grid icon in the upper right corner of the Cloudera AI web application as long as the job or session runs. Even if the web UI does not have authentication, only Contributors and those with more access to the project can access it. For more details, see *Accessing Web User Interfaces from Cloudera AI*. Note that if the Site Administrator has enabled Allow only session creators to run commands on active sessions, then the UI is only available to the session creator. Other users will not be able to access it. Use 127.0.0.1 as the IP.
CDSW_READONLY_PORT	A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_READONLY_PORT will be available in browsers at: http(s)://read-only-<*$CDSW_ENGINE_ID*>.<*$CDSW_DOMAIN*>. Use this port for applications that grant read-only access to project results. A direct link to these web services will be available to users with from the grid icon in the upper right corner of the Cloudera AI web application as long as the job or session runs. Even if the web UI does not have authentication, Viewers and those with more access to the project can access it. For more details, see *Accessing Web User Interfaces from Cloudera AI*. Use 127.0.0.1 as the IP.
CDSW_DOMAIN	The domain on which Cloudera AI is being served. This can be useful for iframing services, as demonstrated in *Accessing Web User Interfaces from Cloudera AI*.
CDSW_CPU_MILLICORES	The number of CPU cores allocated to this engine, expressed in thousandths of a core.
CDSW_MEMORY_MB	The number of megabytes of memory allocated to this engine.
CDSW_IP_ADDRESS	Other engines in the Cloudera AI cluster can contact this engine on this IP address.
CDSW_APP_POLLING_ENDPOINT	Specify a custom endpoint that Cloudera AI uses to check the status of the application. The default value is '/'.

Related Information

Installing Additional Packages

Accessing environmental variables from projects

This topic shows you how to access environmental variables from your code.

Environmental variables are injected into every engine launched for a project, contingent on the scope at which the variable was set (global, project, etc.). The following code samples show how to access a sample environment variable called *DATABASE_PASSWORD* from your project code.

R

```
database.password <- Sys.getenv("DATABASE_PASSWORD")
```

Python

```
import os
database_password = os.environ["DATABASE_PASSWORD"]
```

Scala

```
System.getenv("DATABASE_PASSWORD")
```

Appending Values to Environment Variables:

You can also set environment variables to append to existing values instead of replacing them. For example, when setting the LD_LIBRARY_PATH variable, you can set the value to LD_LIBRARY_PATH:/path/to/set.

Customized engine images

This topic explains how custom engines work and when they shall be used.

By default, Cloudera AI engines are preloaded with a few common packages and libraries for R, Python, and Scala. In addition to these, Cloudera AI also allows you to install any other packages or libraries that are required by your projects. However, directly installing a package to a project as described above might not always be feasible. For example, packages that require root access to be installed, or that must be installed to a path outside /home/cdsw (outside the project mount), cannot be installed directly from the workbench.

For such circumstances, Cloudera AI allows you to extend the base Docker image and create a new Docker image with all the libraries and packages you require. Site administrators can then include this new image in the allowlist for use in projects, and project administrators set the new white-listed image to be used as the default engine image for their projects. For an end-to-end example of this process, see *End-to-End Example: MeCab*.

Note: You will need to remove any unnecessary Cloudera sources or repositories that are inaccessible because of the paywall.

Note that this approach can also be used to accelerate project setup across the deployment. For example, if you want multiple projects on your deployment to have access to some common dependencies (package or software or driver) out of the box, or even if a package just has a complicated setup, it might be easier to simply provide users with an engine that has already been customized for their project(s).

Related Resources

- The Cloudera Engineering Blog post on *Customizing Docker Images in Cloudera Maching Learning* describes an end-to-end example on how to build and publish a customized Docker image and use it as an engine in Cloudera AI.
- For an example of how to extend the base engine image to include Conda, see *Installing Additional Packages*.

Related Information

End-to-end example: MeCab

Installing Additional Packages

Customizing Docker Images in Cloudera AI

Creating a customized engine image

This section walks you through the steps required to create your own custom engine based on the Cloudera AI base image.

For a complete example, see *End-to-End Example: MeCab*.

Create a Dockerfile for the custom image

This topic shows you how to create a Dockerfile for a custom image.

The first step when building a customized image is to create a Dockerfile that specifies which packages you would like to install in addition to the base image.

For example, the following Dockerfile installs the beautifulsoup4 package on top of the base Ubuntu image that ships with Cloudera AI.

```
# Dockerfile

# Specify a Cloudera AI base image
FROM docker.repository.cloudera.com/cloudera/cdsw/engine:13-cml-2021.02-1

# Update packages on the base image and install beautifulsoup4
RUN apt-get update
RUN pip install beautifulsoup4 && pip3 install beautifulsoup4
```

Build the new Docker image

This topic shows you how to use Docker to build a custom image.

A new custom Docker image can be built on any host where Docker binaries are installed. To install these binaries, run the following command on the host where you want to build the new image:

```
docker build -t <image-name>:<tag> . -f Dockerfile
```

If you want to build your image on the Cloudera AI Workbench, you must add the --network=host option to the build command:

```
docker build --network=host -t <image-name>:<tag> . -f Dockerfile
```

Distribute the image

This topic explains the different methods that can be used to distribute a custom engine to all the hosts.

Once you have built a new custom engine, use one of the following ways to distribute the new image to all your Cloudera AI hosts:

Push the image to a public registry such as DockerHub

For instructions, refer the Docker documentation *docker push* and *Push images to Docker Cloud*.

Push the image to your company's Docker registry

When using this method, make sure to tag your image with the following schema:

```
docker tag <image-name> <company-registry>/<user-name>/<image-na
me>:<tag>
```

Once the image has been tagged properly, use the following command to push the image:

```
docker push <company-registry>/<user-name>/<image-name>:<tag>
```

The MeCab example at the end of this topic uses this method.

Related Information

docker push

Including images in allowlist for Cloudera AI projects

This topic describes how to include custom images in the allowlist so that they can be used in projects.

Including a customized image in Cloudera AI is a two-step process.

1. Include the image in the allowlist for the whole deployment.

   First, a site administrator will need to clear the new image for use on the deployment.

   a. Log in as a site administrator.
   b. Click Admin Engines .
   c. Add <company-registry>/<user-name>/<image-name>:<tag> to the allowlist of engine images.
2. Include the image in the allowlist for a specific project

   If you want to start using the image in a project, the project administrator will need to set this image as the default image for the project.

   a. Go to the project Settings page.
   b. Click Engines.
   c. Select the new customized engine from the drop-down list of available Docker images. Sessions and jobs you run in your project will now have access to this engine.

Add Docker registry credentials

To enable Cloudera AI to fetch custom engines from a secure repository, as Administrator you need to add Docker registry credentials.

Create a `kubectl` secret named regcred for your secured Docker registry. The following command creates the secret in your Kubernetes cluster:

```
kubectl create secret docker-registry regcred
    --docker-server=<server host>
    --docker-username=<username>
    --docker-password=<password>
    -n <compute namespace eg. mlx>
```

The next time the engine image is pulled, the new secret will be picked up.

Limitations with customized engines

This topic lists some limitations associated with custom engines.

- Cloudera AI only supports customized engines that are based on the Cloudera AI base image.
- Cloudera AI does not support creation of custom engines larger than 10 GB.

  Cloudera Bug: DSE-4420
- Cloudera AI does not support pulling images from registries that require Docker credentials.

  Cloudera Bug: DSE-1521
- The contents of certain pre-existing standard directories such as /home/cdsw, /tmp, and so on, cannot be modified while creating customized engines. This means any files saved in these directories will not be accessible from sessions that are running on customized engines.

  Workaround: Create a new custom directory in the Dockerfile used to create the customized engine, and save your files to that directory.

End-to-end example: MeCab

This topic walks you through a simple end-to-end example on how to build and use custom engines.

This section demonstrates how to customize the Cloudera AI base engine image to include the MeCab (a Japanese text tokenizer) library.

This is a sample Dockerfile that adds MeCab to the Cloudera AI base image.

```
# Dockerfile

FROM docker.repository.cloudera.com/cloudera/cdsw/engine:13-cml-2021.02-1
RUN rm /etc/apt/sources.list.d/*
RUN apt-get update && \
    apt-get install -y -q mecab \
                          libmecab-dev \
                          mecab-ipadic-utf8 && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
RUN cd /tmp && \
    git clone --depth 1 https://github.com/neologd/mecab-ipadic-neologd.git
 && \
    /tmp/mecab-ipadic-neologd/bin/install-mecab-ipadic-neologd -y -n -p /v
ar/lib/mecab/dic/neologd && \
    rm -rf /tmp/mecab-ipadic-neologd
RUN pip install --upgrade pip
RUN pip install mecab-python==0.996
```

To use this image on your Cloudera AI project, perform the following steps.

1. Build a new image with the Dockerfile.

```
docker build --network=host -t <company-registry>/user/cdsw-mecab:latest .
 -f Dockerfile
```

2. Push the image to your company's Docker registry.

```
docker push <your-company-registry>/user/cdsw-mecab:latest
```

3. Whitelist the image, <your-company-registry>/user/cdsw-mecab:latest. Only a site administrator can do this.

   Go to Admin Engines and add <company-registry>/user/cdsw-mecab:latest to the list of whitelisted engine images.

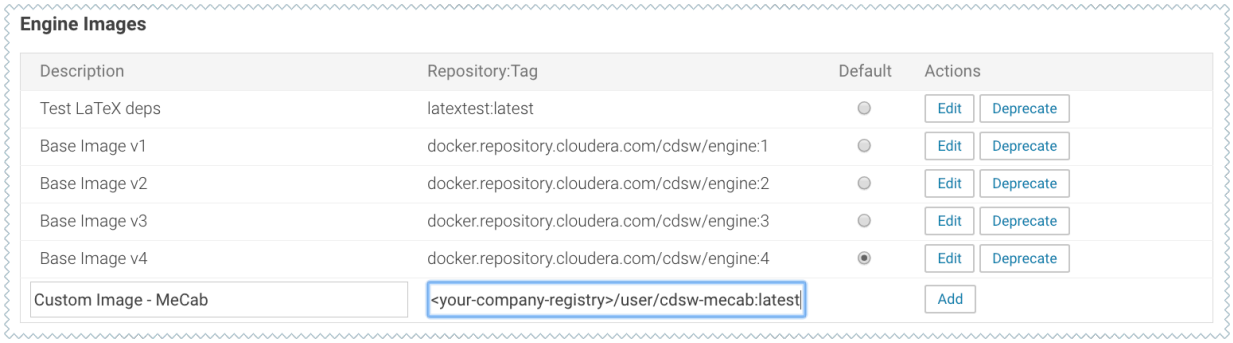

4. Ask a project administrator to set the new image as the default for your project. Go to the project Settings, click Engines, and select company-registry/user/cdsw-mecab:latest from the dropdown.

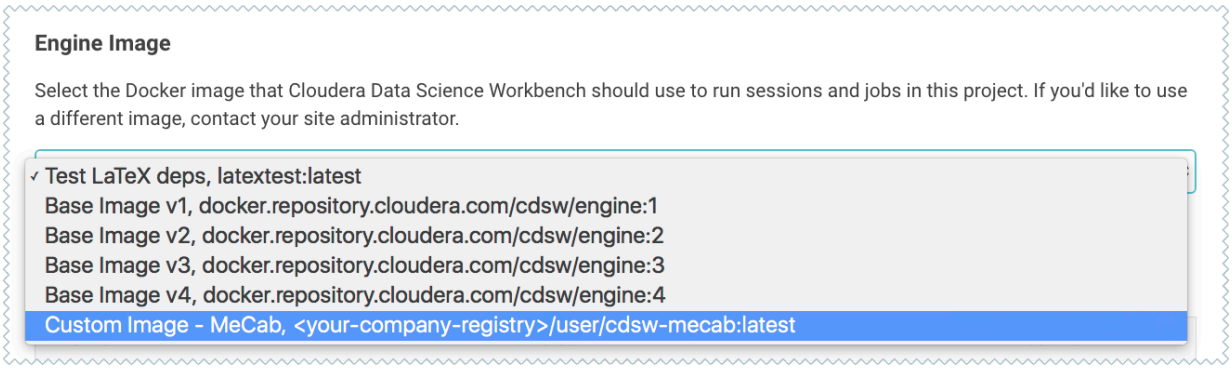

You shall now be able to run this project on the customized MeCab engine.

Pre-Installed Packages in engines

Cloudera AI ships with several base engine images that include Python and R kernels, and frequently used libraries.

Base Engine 15-cml-2021.09-1

Engine 15 ships Python versions 2.7.18 and 3.6.13, and R version 3.6.3.

Items in bold indicate a new version since the last release.

Table 1: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
matplotlib	2.2.4
seaborn	0.9.0
cython	0.29.13
six	1.16.0

Table 2: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.16.5

Library	Version
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.2.4
seaborn	0.9.0
cython	0.29.13
six	1.16.0

Table 3: R Libraries

Package	Version
RCurl	1.98.1.2
caTools	1.18.0
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.4
ggplot2	3.3.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.69
dplyr	1.0.0
httr	1.4.1
httpuv	1.5.4
jsonlite	1.6.1
magrittr	1.5
knitr	1.28
purrr	0.3.4
tm	0.7.7
proxy	0.4.24
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.12
devtools	2.3.0

Base Engine 14-cml-2021.05-1

Engine 14 ships Python versions 2.7.18 and 3.6.10, and R version 3.6.3.

Items in bold indicate a new version since the last release.

Table 4: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1
py4j	0.10.8.1
matplotlib	3.1.2
seaborn	0.9.0
cython	0.29.13
six	1.15.0

Table 5: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.10
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.2.4
seaborn	0.9.0
cython	0.29.13
six	1.15.0

Table 6: R Libraries

Package	Version
RCurl	1.98.1.2
caTools	1.18.0
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.4
ggplot2	3.3.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.69

Package	Version
dplyr	1.0.0
httr	1.4.1
httpuv	1.5.4
jsonlite	1.6.1
magrittr	1.5
knitr	1.28
purrr	0.3.4
tm	0.7.7
proxy	0.4.24
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.12
devtools	2.3.0

Related Information

Base Engine 9
Base Engine 10
Base Engine 11
Base Engine 12
Base Engine 13

Base Engine 13-cml-2020.08-1

Engine 13 ships Python versions 2.7.18 and 3.6.10, and R version 3.6.3.

Items in bold indicate a new version since the last release.

Note: This is the only engine available on CML Private Cloud 1.0.

Table 7: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1
py4j	0.10.8.1
matplotlib	3.1.2
seaborn	0.9.0
cython	0.29.13

Library	Version
six	1.15.0

Table 8: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.10
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.2.4
seaborn	0.9.0
cython	0.29.13
six	1.15.0

Table 9: R Libraries

Package	Version
RCurl	1.98.1.2
caTools	1.18.0
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.4
ggplot2	3.3.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.69
dplyr	1.0.0
httr	1.4.1
httpuv	1.5.4
jsonlite	1.6.1
magrittr	1.5
knitr	1.28
purrr	0.3.4
tm	0.7.7
proxy	0.4.24
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1

Package	Version
rJava	0.9.12
devtools	2.3.0

Related Information

Base Engine 9
Base Engine 10
Base Engine 11
Base Engine 12
Base Engine 14

Base Engine 12-cml-2020.06-2

Engine 12 ships Python versions 2.7.18 and 3.6.10, and R version 3.6.3.

Table 10: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1
py4j	0.10.8.1
matplotlib	3.1.2
seaborn	0.9.0
Cython	0.29.13

Table 11: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.2.4
seaborn	0.9.0
Cython	0.29.13

Table 12: R Libraries

Package	Version
RCurl	1.98.1.2
caTools	1.18.0
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.4
ggplot2	3.3.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.69
dplyr	1.0.0
httr	1.4.1
httpuv	1.5.4
jsonlite	1.6.1
magrittr	1.5
knitr	1.28
purrr	0.3.4
tm	0.7.7
proxy	0.4.24
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.12
devtools	2.3.0

Related Information

Base Engine 9
Base Engine 10
Base Engine 11
Base Engine 13
Base Engine 14

Base Engine 11-cml1.4

Engine 11 ships Python versions 2.7.17 and 3.6.9, and R version 3.6.2.

Table 13: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0

Library	Version
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1
py4j	0.10.8.1
matplotlib	2.0.0
seaborn	0.9.0
Cython	0.29.13

Table 14: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.0.0
seaborn	0.9.0
Cython	0.29.13

Table 15: R Libraries

Package	Version
RCurl	1.95.4.12
caTools	1.17.1.3
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.3
ggplot2	3.2.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.73
dplyr	0.8.3
httr	1.4.1
httpuv	1.5.2
jsonlite	1.6
magrittr	1.5
knitr	1.26
purrr	0.3.3

Package	Version
tm	0.7.7
proxy	0.4.23
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.11
devtools	2.2.1

Related Information

Base Engine 9
Base Engine 10
Base Engine 12
Base Engine 13
Base Engine 14

Base Engine 10-cml1.3

Engine 10 ships Python versions 2.7.17 and 3.6.9, and R version 3.5.1.

Table 16: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1
py4j	0.10.8.1
matplotlib	2.0.0
seaborn	0.9.0
Cython	0.29.13

Table 17: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0

Library	Version
matplotlib	2.0.0
seaborn	0.9.0
Cython	0.29.13

Table 18: R Libraries

Package	Version
RCurl	1.95.4.12
caTools	1.17.1.3
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.3
ggplot2	3.2.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.73
dplyr	0.8.3
httr	1.4.1
httpuv	1.5.2
jsonlite	1.6
magrittr	1.5
knitr	1.26
purrr	0.3.3
tm	0.7.7
proxy	0.4.23
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.11
devtools	2.2.1

Related Information

Base Engine 9
Base Engine 11
Base Engine 12
Base Engine 13
Base Engine 14

Base Engine 9-cml1.2

Engine 9 ships Python 2.7.11 and 3.6.8, and R version 3.5.1.

Table 19: Python Libraries

Library	Version
ipython	5.1.0
requests	2.13.0
Flask	0.12.0
simplejson	3.10.0
numpy	1.13.3
pandas	0.20.1
pandas-datareader	0.2.1
py4j	0.10.7
futures	2.1.4
matplotlib	2.0.0
seaborn	0.8.0
Cython	0.25.2
kudu-python	1.2.0

Table 20: R Libraries

Package	Version
RCurl	1.95.4.12
caTools	1.17.1.2
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.2
ggplot2	3.1.1
cluster	2.0.9
codetools	0.2.16
foreign	0.8.71
dplyr	0.8.1
httr	1.4.0
httpuv	1.5.1
jsonlite	1.6
magrittr	1.5
knitr	1.23
purrr	0.3.2
tm	0.7.6
proxy	0.4.23
data.table	1.12.2
stringr	1.4.0
Rook	1.1.1
rJava	0.9.11

Package	Version
devtools	2.0.2

Related Information

Base Engine 10
Base Engine 11
Base Engine 12
Base Engine 13
Base Engine 14

Cloudera AI

Experiments

Date published: 2020-07-16
Date modified: 2025-01-29

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Experiments with MLflow 4

Cloudera AI Experiment Tracking through MLflow API 4

Running an Experiment using MLflow 5

Visualizing Experiment Results 6

Using an MLflow Model Artifact in a Model REST API 8

Deploying an MLflow model as a Cloudera AI Model REST API 11

Automatic Logging 14

Setting Permissions for an Experiment 14

Known issues and limitations 14

MLflow transformers 15

Experiments with MLflow

Machine Learning requires experimenting with a wide range of datasets, data preparation steps, and algorithms to build a model that maximizes a target metric. Once you have built a model, you also need to deploy it to a production system, monitor its performance, and continuously retrain it on new data and compare it with alternative models.

Cloudera AI lets you train, reuse, and deploy models with any library, and package them into reproducible artifacts that other data scientists can use.

Cloudera AI packages the ML models in a reusable, reproducible form so you can share it with other data scientists or transfer it to production.

Cloudera AI is compatible with the MLflow™ tracking API and makes use of the MLflow client library as the default method to log experiments. Existing projects with existing experiments are still available and usable.

The functionality described in this document is for the new version of the Experiments feature, which replaces an older version of the Experiments feature that could not be used from within Sessions. In Projects that have existing Experiments created using the previous feature, you can continue to view these existing Experiments. New projects use the new Experiments feature.

Cloudera AI Experiment Tracking through MLflow API

Cloudera AI's experiment tracking features allow you to use the MLflow client library for logging parameters, code versions, metrics, and output files when running your machine learning code. The MLflow library is available in Cloudera AI Sessions without you having to install it. Cloudera AI also provides a UI for later visualizing the results. MLflow tracking lets you log and query experiments using the following logging functions:

Note: Cloudera AI currently supports only Python for experiment tracking.

- mlflow.create_experiment() creates a new experiment and returns its ID. Runs can be launched under the experiment by passing the experiment ID to mlflow.start_run.

  Cloudera recommends that you create an experiment to organize your runs. You can also create experiments using the UI.
- mlflow.set_experiment() sets an experiment as active. If the experiment does not exist, mlflow.set_experiment creates a new experiment. If you do not wish to use the set_experiment method, a default experiment is selected.

  Cloudera recommends that you set the experiment using mlflow.set_experiment.
- mlflow.start_run() returns the currently active run (if one exists), or starts a new run and returns a mlflow.ActivEeRun object usable as a context manager for the current run. You do not need to call start_run explicitly; calling one of the logging functions with no active run automatically starts a new one.
- mlflow.end_run() ends the currently active run, if any, taking an optional run status.
- mlflow.active_run() returns a mlflow.entities.Run object corresponding to the currently active run, if any.

  Note: You cannot access currently-active run attributes (parameters, metrics, etc.) through the run returned by mlflow.active_run. In order to access such attributes, use the mlflow.tracking.MlflowClient as follows:

  ```
  client = mlflow.tracking.MlflowClient()
  data = client.get_run(mlflow.active_run().info.run_id).data
  ```
- mlflow.log_param() logs a single key-value parameter in the currently active run. The key and value are both strings. Use mlflow.log_params() to log multiple parameters at once.

- mlflow.log_metric() logs a single key-value metric for the current run. The value must always be a number. MLflow remembers the history of values for each metric. Use mlflow.log_metrics() to log multiple metrics at once.

  Parameters:

  - key - Metric name (string)
  - value - Metric value (float). Note that some special values such as +/- Infinity may be replaced by other values depending on the store. For example, the SQLAlchemy store replaces +/- Infinity with max / min float values.
  - step - Metric step (int). Defaults to zero if unspecified.

  Syntax - mlflow.log_metrics(metrics: Dict[str, float], step: Optional[int] = None) # None
- mlflow.set_tag() sets a single key-value tag in the currently active run. The key and value are both strings. Use mlflow.set_tags() to set multiple tags at once.
- mlflow.log_artifact() logs a local file or directory as an artifact, optionally taking an artifact_path to place it within the run's artifact URI. Run artifacts can be organized into directories, so you can place the artifact in a directory this way.
- mlflow.log_artifacts() logs all the files in a given directory as artifacts, again taking an optional artifact_path.
- mlflow.get_artifact_uri() returns the URI that artifacts from the current run should be logged to.

For more information on MLflow API commands used for tracking, see MLflow Tracking.

Running an Experiment using MLflow

This topic walks you through a simple example to help you get started with Experiments in Cloudera AI.

Best practice: It is useful to display two windows while creating runs for your experiments: one window displays the Experiments tab and another displays the MLflow Session.

1. From your Project window, click New Experiment and create a new experiment. Keep this window open to return to after you run your new session.
2. From your Project window, click New Session.
3. Create a new session using ML Runtimes. Experiment runs cannot be created from sessions using Legacy Engine.
4. In your Session window, import MLflow by running the following code: import mlflow The ML Flow client library is installed by default, but you must import it for each session.
5. Start a run and then specify the MLflow parameters, metrics, models and artifacts to be logged. You can enter the code in the command prompt or create a project. See *Cloudera AI Experiment Tracking through MLflow API* for a list of functions you can use.

   For example:

```
mlflow.set_experiment(<experiment_name>)
mlflow.start_run()
mlflow.log_param("input", 5)
mlflow.log_metric("score", 100)
with open("data/features.txt", 'w') as f:
    f.write(features)
# Writes all files in "data" to root artifact_uri/states
mlflow.log_artifacts("data", artifact_path="states")
## Artifacts are stored in project directory under
/home/cdsw/.experiments/<experiment_id>/<run_id>/artifacts
```

```
mlflow.end_run()<
```

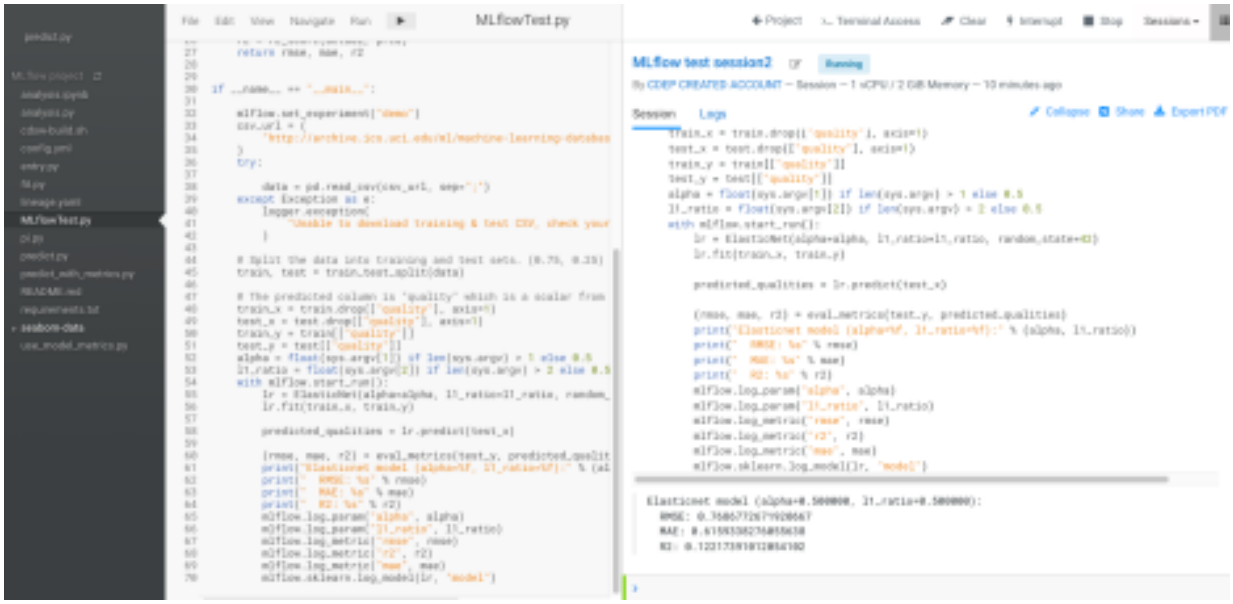

For information on using editors, see Using Editors for ML Runtimes.

6. Continue creating runs and tracking parameters, metrics, models, and artifacts as needed.
7. To view your run information, display the Experiments window and select your experiment name. Cloudera AI displays the Runs table.

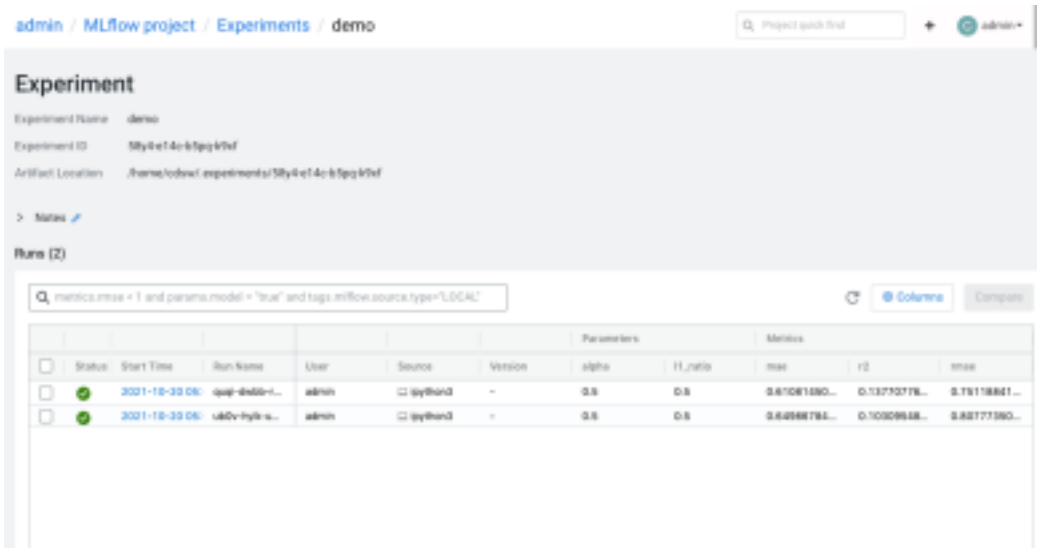

8. Click the Refresh button on the Experiments window to display recently created runs
9. You can customize the Run table by clicking Columns, and selecting the columns you want to display.

Related Information

Using Editors for ML Runtimes

Visualizing Experiment Results

After you create multiple runs, you can compare your results.

1. Go to Experiments and click on your experiment name. Cloudera AI displays the Runs table populated by all of the runs for the experiment.

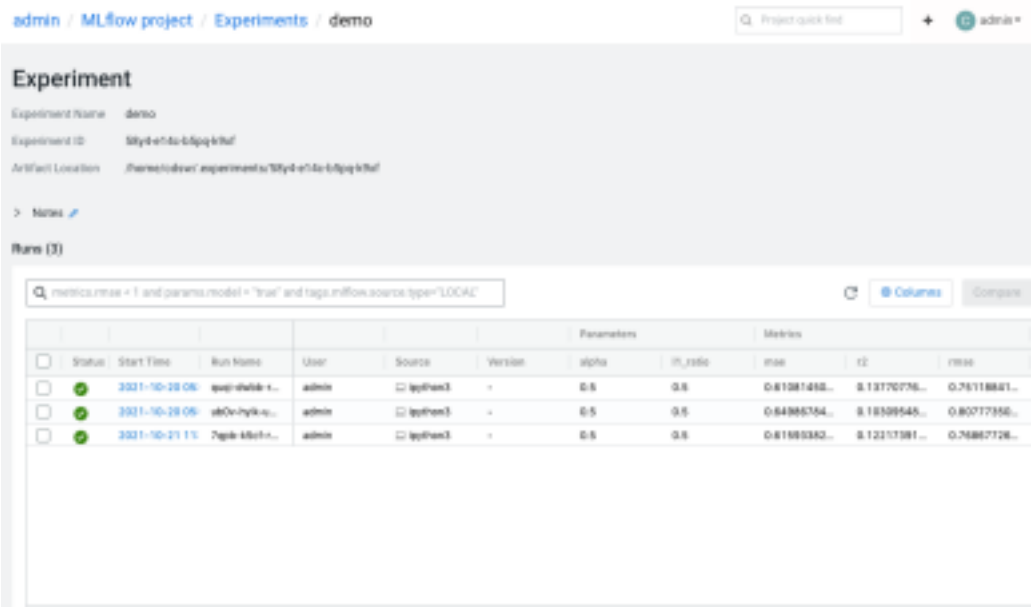

2. You can search your run information by using the search field at the top of the Run table.
3. You can customize the Run table by clicking Columns, and selecting the columns you want to display.
4. You can display details for a specific run by clicking the start time for the run in the Run table. You can add notes for the run by clicking the Notes icon. You can display the run metrics in a chart format by clicking the specific metric under Metrics.
5. To compare the data from multiple runs, use the checkbox in the Run table to select the runs you want to compare. You can use the top checkbox to select all runs in the table. Alternatively, you can select runs using the spacebar and arrow keys.

6. Click Compare. Alternatively, you can press Cmd/Ctrl + Enter. Cloudera AI displays a separate window containing a table titled Run Comparison and options for comparing your parameters and metrics.

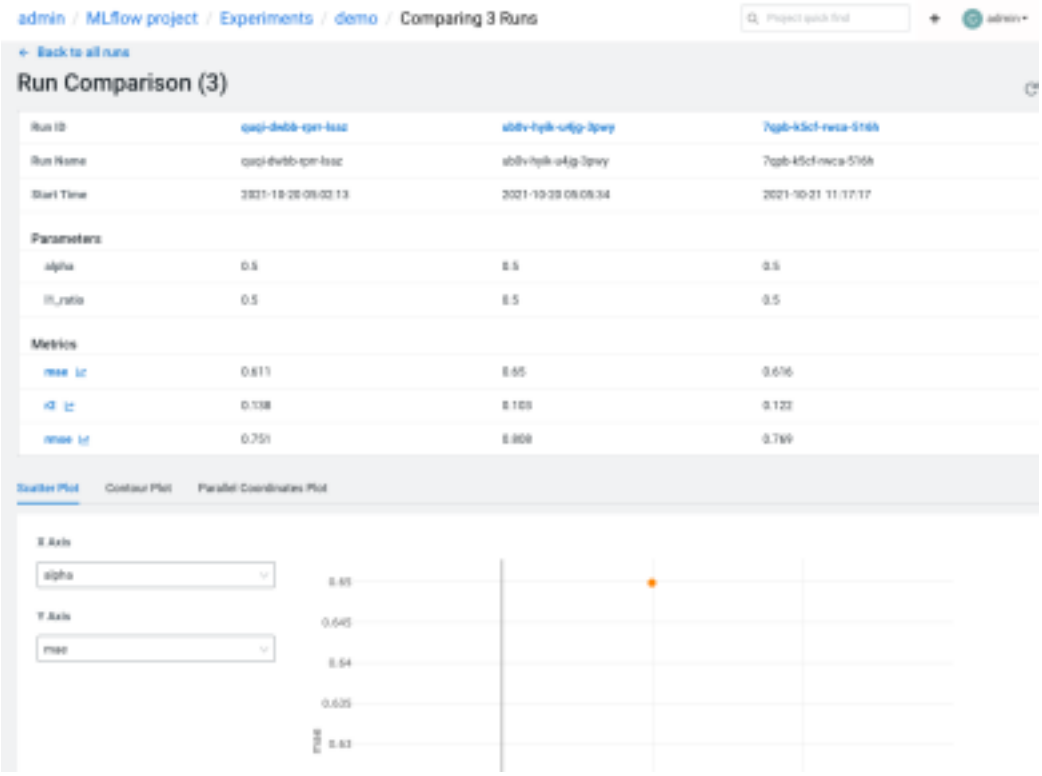

This Run Comparison table lists all of the parameters and the most recent metric information from the runs you selected. Parameters that have changed are highlighted

7. You can graphically display the Run metric data by clicking the metric names in the Metrics section. If you have a single value for your metrics, it will display as a bar chart. If your run has multiple values, the metrics comparison page displays the information with multiple steps, for example, over time. You can choose how the data is displayed:
   - Time (Relative): graphs the time relative to the first metric logged, for each run.
   - Time (Wall): graphs the absolute time each metric was logged.
   - Step: graphs the values based on the cardinal order.
8. Below the Run Comparison table, you can choose how the Run information is displayed:
   - Scatter Plot: Use the scatter plot to see patterns, outliers, and anomalies.
   - Contour Plot: Contour plots can only be rendered when comparing a group of runs with three or more unique metrics or parameters. Log more metrics or parameters to your runs to visualize them using the contour plot.
   - Parallel Coordinates Plot: Choose the parameters and metrics you want displayed in the plot.

Using an MLflow Model Artifact in a Model REST API

You can use MLflow to create, deploy, and manage models as REST APIs to serve predictions

1. To create an MLflow model add the following information when you run an experiment:

```
mlflow.log_artifacts ("output")
```

```
mlflow.sklearn.log_model(lr, "model")
```

For example:

```
# The data set used in this example is from http://archive.ics.uci.edu/ml/
datasets/Wine+Quality
# P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
# Modeling wine preferences by data mining from physicochemical properti
es. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.
import os
import warnings
import sys

import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_sc
ore
from sklearn.model_selection import train_test_split
from sklearn.linear_model import ElasticNet
from urllib.parse import urlparse
import mlflow
from mlflow.models import infer_signature
import mlflow.sklearn

import logging

logging.basicConfig(level=logging.WARN)
logger = logging.getLogger(__name__)

def eval_metrics(actual, pred):
    rmse = np.sqrt(mean_squared_error(actual, pred))
    mae = mean_absolute_error(actual, pred)
    r2 = r2_score(actual, pred)
    return rmse, mae, r2

if __name__ == "__main__":
    warnings.filterwarnings("ignore")
    np.random.seed(40)

    # Read the wine-quality csv file from the URL
    csv_url = (
        "https://raw.githubusercontent.com/mlflow/mlflow/master/tests/da
tasets/winequality-red.csv"
    )
    try:
        data = pd.read_csv(csv_url, sep=";")
    except Exception as e:
        logger.exception(
            "Unable to download training & test CSV, check your internet
 connection. Error: %s", e
        )

    # Split the data into training and test sets. (0.75, 0.25) split.
    train, test = train_test_split(data)

    # The predicted column is "quality" which is a scalar from [3, 9]
    train_x = train.drop(["quality"], axis=1)
    test_x = test.drop(["quality"], axis=1)
    train_y = train[["quality"]]
    test_y = test[["quality"]]

    alpha = float(sys.argv[1]) if len(sys.argv) > 1 else 0.5
    l1_ratio = float(sys.argv[2]) if len(sys.argv) > 2 else 0.5
```

```
    with mlflow.start_run():
        lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
        lr.fit(train_x, train_y)

        predicted_qualities = lr.predict(test_x)

        (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
        print("Elasticnet model (alpha={:f}, l1_ratio={:f}):".format(al
pha, l1_ratio))
        print("  RMSE: %s" % rmse)
        print("  MAE: %s" % mae)
        print("  R2: %s" % r2)

        mlflow.log_param("alpha", alpha)
        mlflow.log_param("l1_ratio", l1_ratio)
        mlflow.log_metric("rmse", rmse)
        mlflow.log_metric("r2", r2)
        mlflow.log_metric("mae", mae)

        predictions = lr.predict(train_x)
        signature = infer_signature(train_x, predictions)
        mlflow.sklearn.log_model(lr, "model", signature=signature, regist
ered_model_name="testmodel")
```

In this example we are training a machine learning model using linear regression to predict wine quality. This script creates the MLflow model artifact and logs it to the model directory: /home/cdsw/.experiments/<experiment_id>/<run_id>/artifacts/models

2. To view the model, navigate to the Experiments page and select your experiment name. Cloudera AI displays the Runs page and lists all of your current runs.
3. Click the run from step 1 that created the MLflow model. Cloudera AI displays the Runs detail page

4. Click Artifacts to display a list of all the logged artifacts for the run.

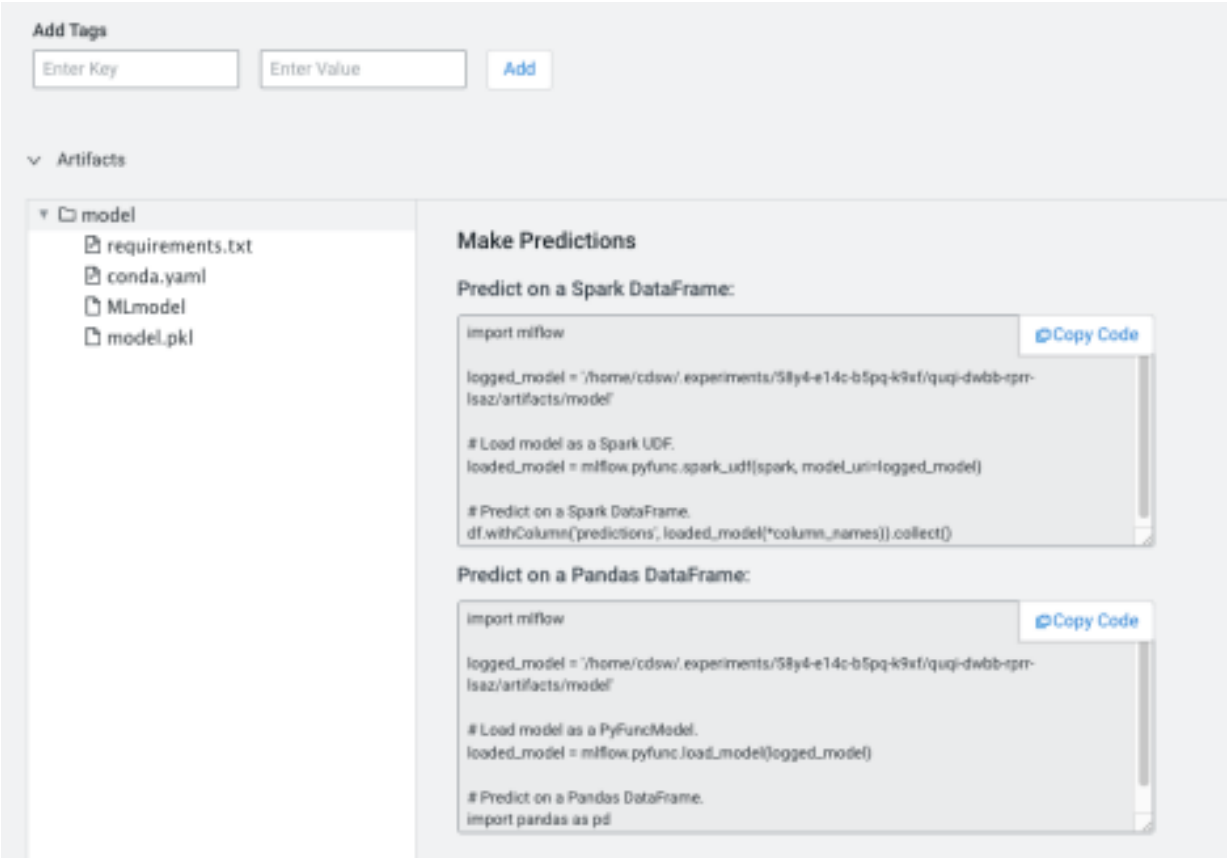

5. Click model. Cloudera AI displays the MLflow information you use to create predictions for your experiment.

Deploying an MLflow model as a Cloudera AI Model REST API

In the future, you will be able to register models to a Cloudera AI Registry and then deploy Model REST APIs with those models. Today, these models can be deployed using the following manual process instead

1. Navigate to your project. Note that models are always created within the context of a project.
2. Click Open Workbench and launch a new Python 3 session.
3. Create a new file within the project if one does not already exist: cdsw-build.sh This file defines the function that will be called when the model is run and will contain the MLflow prediction information.
4. Add the following information to the cdsw-build.sh file: pip3 install sklearn mlflow pandas
5. For non-Python template projects and old projects check the following.

   a. Check to make sure you have a .gitignore file. If you do not have the file, add it.
   b. Add the following information to the .gitignore file: !.experiments

   For new projects using a Python template, this is already present.
6. Create a Python file to call your model artifact using a Python function. For example:

   - Filename: mlpredict.py
   - Function: predict

7. Copy the MLflow model file path from the Make Predictions pane in the Artifacts section of the Experiments/Run details page and load it in the Python file. This creates a Python function which accepts a dictionary of the input variables and converts these to a Pandas data frame, and returns the model prediction. For example:

```
import mlflow
import pandas as pd
logged_model =
     '/home/cdsw/.experiments/7qwz-l620-d7v6-1922/glma-oqxb-szc7-c8hf/a
rtifacts/model'
def predict(args):
     # Load model as a PyFuncModel.
     data = args.get('input')
     loaded_model = mlflow.pyfunc.load_model(logged_model)
     # Predict on a Pandas DataFrame.
     return loaded_model.predict(pd.DataFrame(data))
```

Note: In practice, do not assume that users calling the model will provide input in the correct format or enter good values. Always perform input validation.

8. Deploy the predict function to a REST endpoint.

   a. Go to the project Overview page
   b. Click Models New Model .
   c. Give the model a Name and Description
   d. Enter details about the model that you want to build. In this case:

      - File: mlpredict.py
      - Function: predict
      - Example Input:

```
{
"input": [
[7.4, 0.7, 0, 1.9, 0.076, 11, 34, 0.9978,
3.51, 0.56, 9.4]
]
}
```

      - Example output:

```
[
5.575822297312952
```

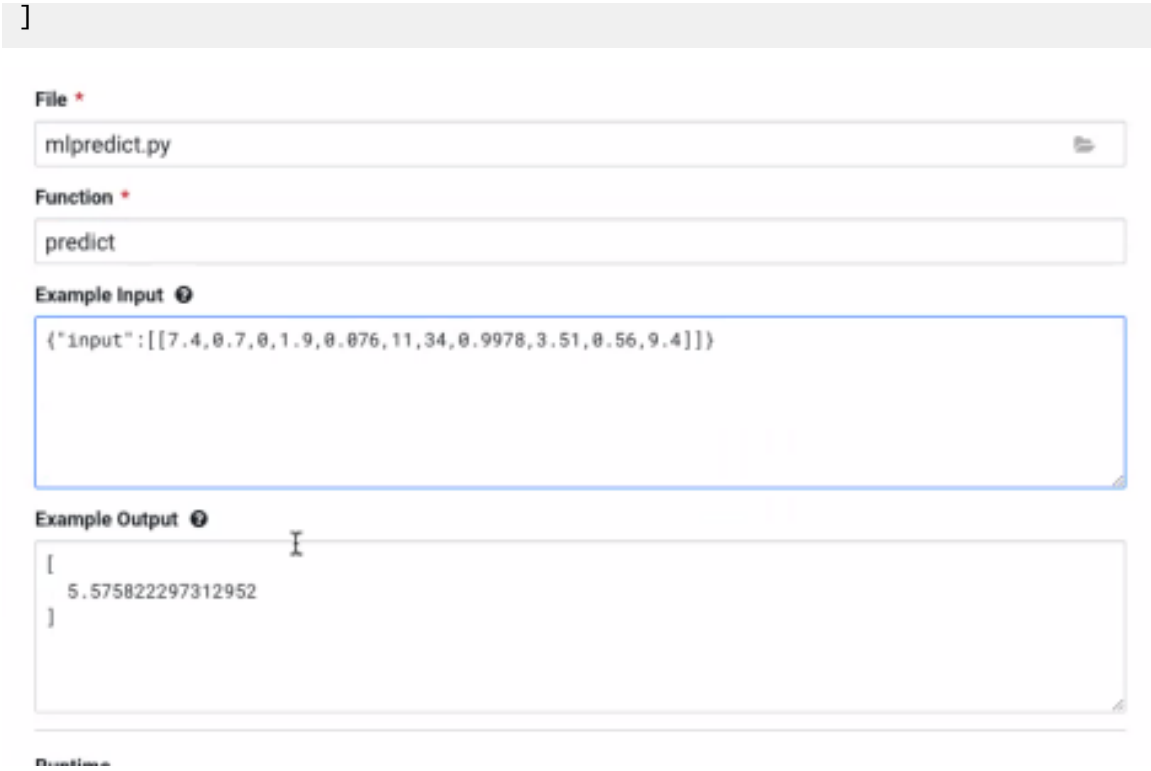

e. Select the resources needed to run this model, including any replicas for load balancing.

> **Note:** The list of options here is specific to the default engine you have specified in your Project Settings: ML Runtimes or Legacy Engines. Engines allow kernel selection, while ML Runtimes allow Editor, Kernel, Variant, and Version selection. Resource Profile list is applicable for both ML Runtimes and Legacy Engines.

f. Click Deploy Model.

9. Click on the model to go to its Overview page.
10. Click Builds to track realtime progress as the model is built and deployed. This process essentially creates a Docker container where the model will live and serve requests.

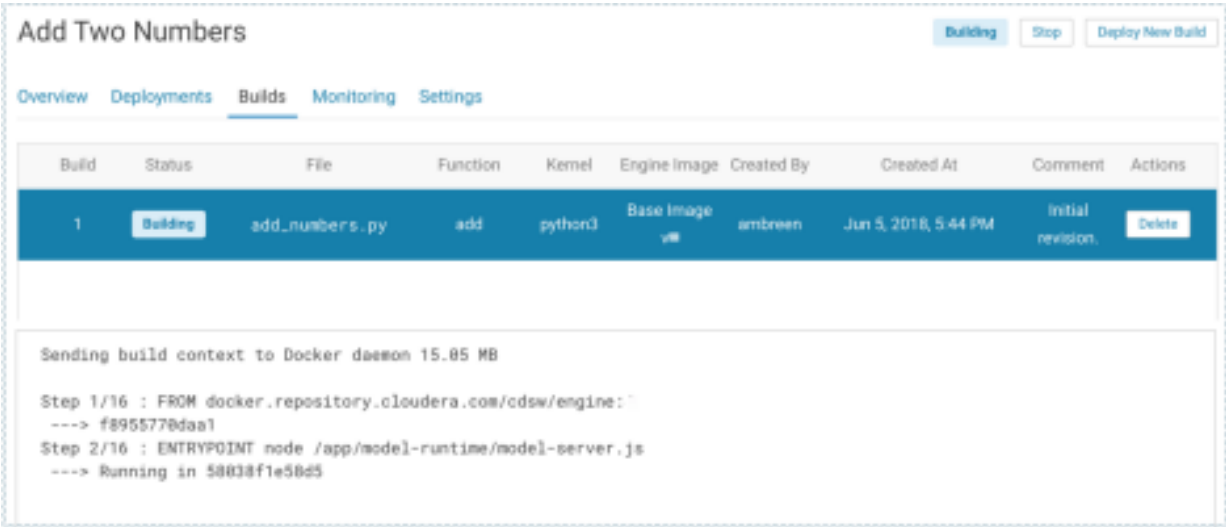

11. Once the model has been deployed, go back to the model Overview page and use the Test Model widget to make sure the model works as expected. If you entered example input when creating the model, the Input field will be pre-populated with those values.

12. Click Test. The result returned includes the output response from the model, as well as the ID of the replica that served the request.

    Model response times depend largely on your model code. That is, how long it takes the model function to perform the computation needed to return a prediction. It is worth noting that model replicas can only process one request at a time. Concurrent requests will be queued until the model can process them.

1.0 and 2.0 scoring API differences

The format for calling the model has changed between MLflow versions 1.0 and 2.0.

The MLflow Model scoring protocol has changed in MLflow version 2.0. If you are seeing an error, you are likely using an outdated scoring request format. To resolve the error, either update your request format or adjust your MLflow Model's requirements file to specify an older version of MLflow (for example, change the 'mlflow' requirement specifier to mlflow==1.30.0). If you are making a request using the MLflow client (for example, using mlflow.pyfunc.spark_udf()), upgrade your MLflow client to a version >= 2.0 in order to use the new request format.

For more information about the updated MLflow Model scoring protocol in MLflow 2.0, see *MLflow Models*.

Related Information

MLflow Models

Automatic Logging

Automatic logging allows you to log metrics, parameters, and models without the need for an explicit log statement.

You can perform autologging two ways:

1. Call mlflow.autolog() before your training code. This will enable autologging for each supported library you have installed as soon as you import it.
2. Use library-specific autolog calls for each library you use in your code. See below for examples.

For more information about the libraries supported by autologging, see Automatic Logging.

Setting Permissions for an Experiment

Experiments are associated with the project ID, so permissions are inherited from the project. If you want to allow a colleague to view the experiments of a project, you should give them Viewer (or higher) access to the project.

Known issues and limitations

Cloudera AI has the following known issues and limitations with experiments and MLflow.

- Cloudera AI currently supports only Python for experiment tracking.
- Experiment runs cannot be created from MLFlow on sessions using Legacy Engine. Instead, create a session using an ML Runtimes.
- The version column in the runs table is empty for every run. In a future release, this will show a git commit sha for projects using git.
- There is currently no mechanism for registering a model to a Cloudera AI Registry. In a future release, you will be able to register models to a Cloudera AI Registry and then deploy Model REST APIs with those models.
- Browsing an empty experiment will display a spinner that does not go away.
- Running an experiment from the workbench (from the dropdown menu) refers to legacy experiments and should not be used going forward.

- Tag/Metrics/Parameter columns that were previously hidden on the runs table will be remembered, but Cloudera AI won't remember hiding any of the other columns (date, version, user, etc.)
- Admins can not browse all experiments. They can only see their experiments on the global Experiment page.
- Performance issues may arise when browsing the run details of a run with a lot of metric results, or when comparing a lot of runs.
- Runs can not be deleted or archived.

MLflow transformers

This is an example of how MLflow transformers can be supported in Cloudera AI.

Note: This is an experimental feature.

This example shows how to implement a translation workflow using a translation model.

1. Save the following as a file, for example, named mlflowtest.py.

```
#!pip3 install torch transformers torchvision tensorflow

    import mlflow
    from mlflow.models import infer_signature
    from mlflow.transformers import generate_signature_output
    from transformers import pipeline

    en_to_de = pipeline("translation_en_to_de")

    data = "MLflow is great!"
    output = generate_signature_output(en_to_de, data)
    #signature = infer_signature(data, output)

    with mlflow.start_run() as run:
    mlflow.transformers.log_model(
    transformers_model=en_to_de,
    artifact_path="english_to_german_translator",
    input_example=data,
    registered_model_name="entodetranslator",
    )


    model_uri = f"runs:/{run.info.run_id}/english_to_german_translator"
    loaded = mlflow.pyfunc.load_model(model_uri)

    print(loaded.predict(data))
```

2. In the Cloudera AI Registry page, find the entodetranslator model. Deploy the model.
3. Make a request using the following payload:

```
{
  "dataframe_split": {
    "columns": [
      "data"
    ],
    "data": [
      [
        "MLflow is great!"
      ]
    ]
  }
```

```
}
```

4. In a session, run the mlflowtest.py file. It should print the following output.

```
print(loaded.predict(data))
['MLflow ist großartig!']
```

Note: For more information, see mlflow.transformers.

Cloudera AI

Using GPUs for Cloudera AI Projects

Date published: 2020-07-16
Date modified: 2024-11-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using GPUs for Cloudera AI projects 4

Testing GPU Setup 4

Testing ML Runtime GPU Setup 5

Using GPUs for Cloudera AI projects

A GPU is a specialized processor that can be used to accelerate highly parallelized computationally-intensive workloads. Because of their computational power, GPUs have been found to be particularly well-suited to deep learning workloads. Ideally, CPUs and GPUs are used in tandem for data engineering and data science workloads. A typical machine learning workflow involves data preparation, model training, model scoring, and model fitting. You can use existing general-purpose CPUs for each stage of the workflow, and optionally accelerate the math-intensive steps with the selective application of special-purpose GPUs. For example, GPUs allow you to accelerate model fitting using frameworks such as Tensorflow, PyTorch, and Keras.

By enabling GPU support, data scientists can share GPU resources available on Cloudera AI Workbenches. Users can request a specific number of GPU instances, up to the total number available, which are then allocated to the running session or job for the duration of the run.

Enabling GPUs on Cloudera AI Workbenches

If you are using ML Runtimes, you must use the ML Runtimes version for your Python library Nvidia GPU Edition.

If you are using a Legacy Engine, to enable GPU usage on Cloudera AI, select GPUs when you are provisioning the workbench. If your existing workbench does not have GPUs provisioned, contact your Cloudera AI administrator to provision a new one for you. For instructions, see *Provisioning Cloudera AI Workbenches*.

Important: Review your cloud service provider account limits

For example, AWS imposes certain default limits for AWS services, and you might not have default access to GPU instances at all. Make sure you review your account's current usage status and resource limits before you start provisioning GPU resources for Cloudera AI.

Related Information

Provisioning Cloudera AI Workbenches

Testing GPU Setup

Testing GPU Setup

Use these code samples to test that your GPU setup works with several common deep learning libraries. The specific versions of libraries depend on the particular GPU used and the GPU driver version. You can use this testing for GPU setup using ML Runtimes or Legacy Engines.

1. Go to a project that is using the CUDA engine and click Open Workbench.
2. Launch a new session with GPUs.
3. Run the following command in the workbench command prompt to verify that the driver was installed correctly:

```
! /usr/bin/nvidia-smi
```

4. Use any of the following code samples to confirm that the new engine works with common deep learning libraries.

PyTorch

```
!pip3 install torch==1.4.0
from torch import cuda
assert cuda.is_available()
assert cuda.device_count() > 0
print(cuda.get_device_name(cuda.current_device()))
```

Note: The PyTorch installation requires at least 4 GB of memory.

Tensorflow

```
!pip3 install tensorflow-gpu==2.1.0
from tensorflow.python.client import device_lib
assert 'GPU' in str(device_lib.list_local_devices())
device_lib.list_local_devices()
```

Keras

```
!pip3 install keras
from keras import backend
assert len(backend.tensorflow_backend._get_available_gpus()) > 0
print(backend.tensorflow_backend._get_available_gpus())
```

Testing ML Runtime GPU Setup

You can use the following simple examples to test whether the new ML Runtime is able to leverage GPUs as expected.

1. Go to a project that is using the ML Runtimes NVIDIA GPU edition and click Open Workbench.
2. Launch a new session with GPUs.
3. Run the following command in the workbench command prompt to verify that the driver was installed correctly:

   ```
   ! /usr/bin/nvidia-smi
   ```

4. Use any of the following code samples to confirm that the new engine works with common deep learning libraries.

Pytorch

```
!pip3 install torch==1.4.0
from torch import cuda
assert cuda.is_available()
assert cuda.device_count() > 0
print(cuda.get_device_name(cuda.current_device()))
```

Tensorflow

```
!pip3 install tensorflow-gpu==2.1.0
from tensorflow.python.client import device_lib
assert 'GPU' in str(device_lib.list_local_devices())
device_lib.list_local_devices()
```

Keras

```
!pip3 install keras
from keras import backend
assert len(backend.tensorflow_backend._get_available_gpus()) > 0
print(backend.tensorflow_backend._get_available_gpus())
```

Cloudera AI

Accessing Data from Cloudera AI

Date published: 2020-07-16
Date modified: 2024-11-21

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Data Access 4

Upload and work with local files 4

Connect to Cloudera Data Platform Data Lake 5
- Setup Data Lake Access 5
- Example: Connect a Spark session to the Data Lake 9
- Use Direct Reader Mode with PySpark 12
- Use Direct Reader Mode with SparklyR 13
- Create an Iceberg data connection 15
- Accessing Data from HDFS 16

Connecting to Cloudera Data Warehouse 17
- Accessing data with Spark 18
  - Use JDBC Connection with PySpark 18

Connect to a Cloudera Data Hub cluster 19

Connecting to external Amazon S3 buckets 20

Connect to External SQL Databases 21

Accessing Ozone storage 22
- Connecting to Ozone filesystem 22
- Accessing local files in Ozone 22

Data Access

Cloudera AI is a flexible, open platform supporting connections to many data sources.

Cloudera AI supports easy, secure data access through connection snippets and the cml.data library. This library, implemented in Python, abstracts all of the complexity of configuring, initializing, and authenticating data connections. Users choosing to manually create and configure the data connections shall follow the relevant instructions.

Related Information

Using data connection snippets

Upload and work with local files

This topic includes code samples that demonstrate how to access local data for Cloudera AI Workbench workloads.

If you want to work with existing data files (.csv, .txt, etc.) from your computer, you can upload these files directly to your project in the Cloudera AI Workbench. Go to the project's Overview page. Under the Files section, click Upload and select the relevant data files to be uploaded. These files will be uploaded to an NFS share available to each project.

Note: Storing large data files in your Project folder is highly discouraged. You can store your data files in the Data Lake.

The following sections use the tips.csv dataset to demonstrate how to work with local data stored in your project. Before you run these examples, create a folder called data in your project and upload the dataset file to it.

Python

```
import pandas as pd
tips = pd.read_csv('data/tips.csv')

tips \
  .query('sex == "Female"') \
  .groupby('day') \
  .agg({'tip' : 'mean'}) \
  .rename(columns={'tip': 'avg_tip_dinner'}) \
  .sort_values('avg_tip_dinner', ascending=False)
```

R

```
library(readr)
library(dplyr)

# load data from .csv file in project
tips <- read_csv("data/tips.csv")

# query using dplyr
tips %>%
  filter(sex == "Female") %>%
  group_by(day) %>%
  summarise(
    avg_tip = mean(tip, na.rm = TRUE)
  ) %>%
```

```
arrange(desc(avg_tip))
```

Connect to Cloudera Data Platform Data Lake

Cloudera AI has secure access to the data stored in the Cloudera Data Platform Data Lake.

The Data Connection Snippet feature now suggests using the cml.data library to connect to the Cloudera Data Platform Data Lake - these code snippets pop up as suggestions for every new session in a project. For more information, see *Data Exploration and Visualization*.

However, if you would still like to use raw code to connect, follow the instructions in the sections below.

Related Information

Data Exploration and Visualization

Setup Data Lake Access

Cloudera AI can access data tables stored in an AWS or Microsoft Azure Data Lake. As a Cloudera AI Admin, follow this procedure to set up the necessary permissions.

About this task

The instructions apply to Data Lakes on both AWS and Microsoft Azure. Follow the instructions that apply to your environment.

Procedure

1. Cloud Provider Setup

   Make sure the prerequisites for AWS or Azure are satisfied (see the Related Topics for AWS environments and Azure environments). Then, create a Cloudera Data Platform environment as follows.

   a) For environment logs, create an S3 bucket or ADLS Gen2 container.

   > **Note:** For ADLS Gen2, create a dedicated container for logs, and a dedicated container for data, within the same account.

   b) For environment storage, create an S3 bucket or ADLS Gen2 container.
   c) For AWS, create AWS policies for each S3 bucket, and create IAM roles (simple and instance profiles) for these policies.
   d) For Azure, create managed identities for each of the personas, and create roles to map the identities to the ADLS permissions.

   For detailed information on S3 or ADLS, see Related information.

2. Environment Setup

   In Cloudera Data Platform, set up paths for logs and native data access to the S3 bucket or ADLS Gen2 container.

   In the Environment Creation wizard, set the following:

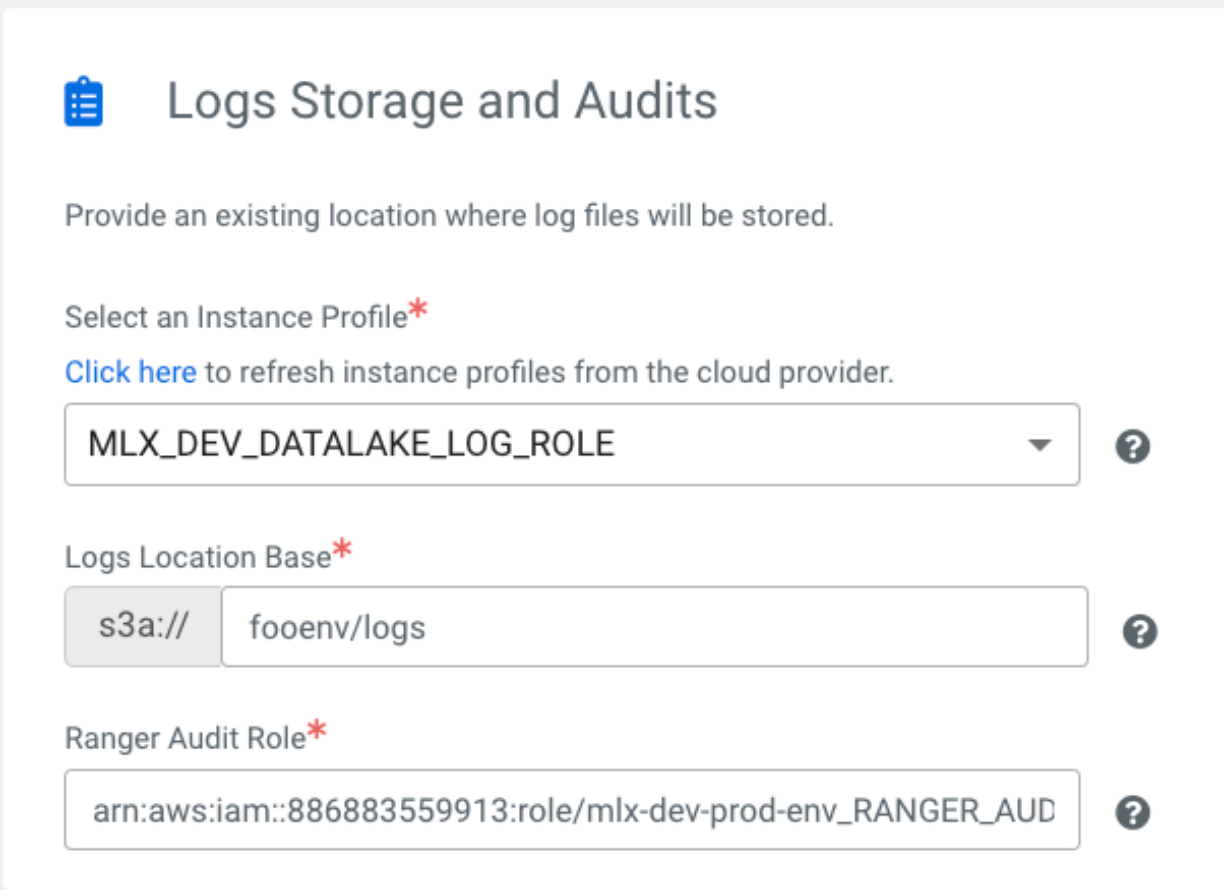

   a) Logs Storage and Audits

      1. Instance Profile - The IAM role or Azure identity that is attached to the master node of the Data Lake cluster. The Instance Profile enables unauthenticated access to the S3 bucket or ADLS container for logs.
      2. Logs Location Base - The location in S3 or ADLS where environment logs are saved.

         Note: The instance profile or Azure identity must refer to the same logs location base in S3 or ADLS.

         .

      3. Ranger Audit Role - The IAM role or Azure identity that has S3 or ADLS access to write Ranger audit events. Ranger uses Hadoop authentication, therefore it uses IDBroker to access the S3 bucket or ADLS container, rather than using Instance profiles or Azure identities directly.

   b) Data Access

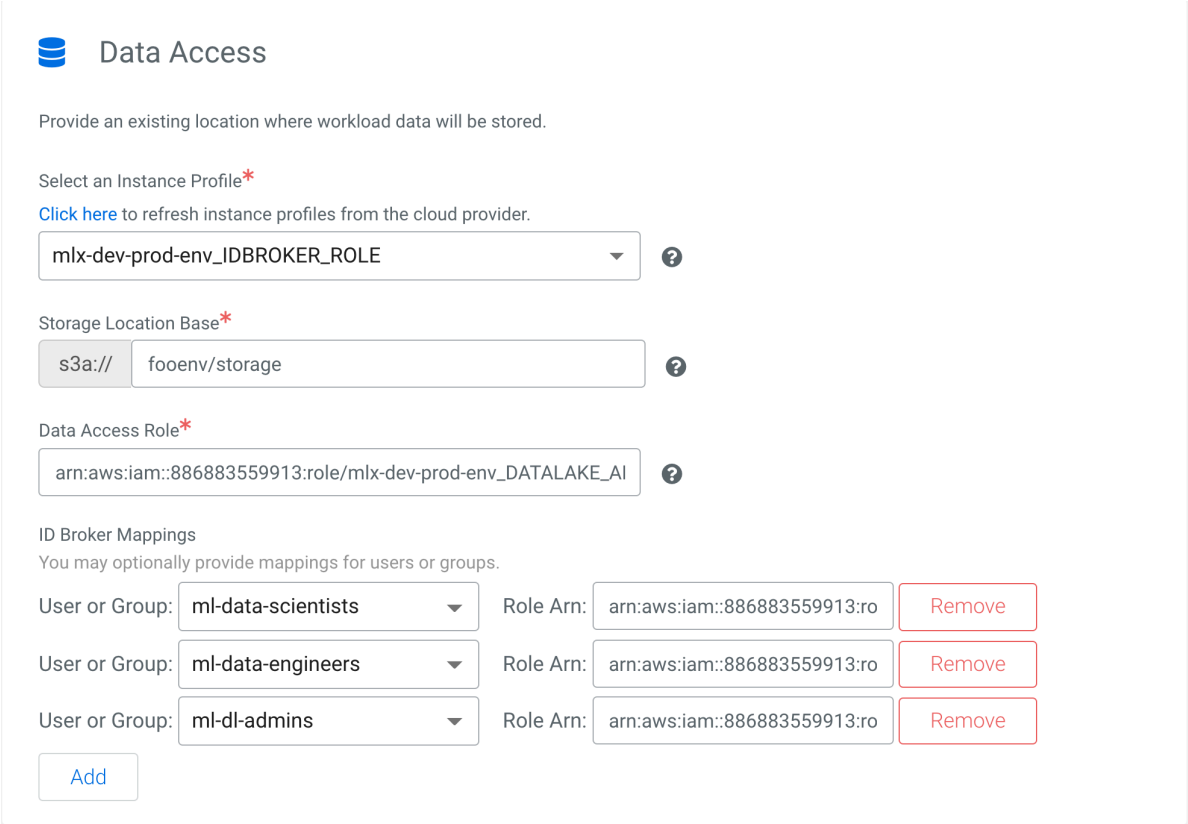

1. Instance Profile - The IAM role or Azure identity that is attached to the IDBroker node of the Data Lake cluster. IDBroker uses this profile to assume roles on behalf of users and get temporary credentials to access S3 buckets or ADLS containers.
2. Storage Location Base - The S3 or ADLS location where data pertaining to the environment is saved.
3. Data Access Role - The IAM role or Azure identity that has access to read or write environment data. For example, Hive creates external tables by default in the Cloudera Data Platform environments, where metadata is stored in HMS running in the Data Lake. The data itself is stored in S3 or ADLS. As Hive uses Hadoop authentication, it uses IDBroker to access S3 or ADLS, rather than using Instance profiles or Azure identities. Hive uses the data access role for storage access.

   Note: The data access role must have permissions to access the S3 or ADLS storage location.

4. ID Broker Mappings - These specify the mappings between the Cloudera Data Platform user or groups to the AWS IAM roles or Azure roles that have appropriate S3 or ADLS access. This setting enables IDBroker to get appropriate S3 or ADLS credentials for the users based on the role mappings defined.

   Note: There is no limit to the number of mappings that one can define but each user can only be assigned to one of the role mappings.

This completes installation of the environment.

3. User Group Mappings

   In Cloudera Data Platform, you can assign users to groups to simplify permissions management. For example, you could create a group called ml-data-scientists, and assign two individual users to it, as shown here. .

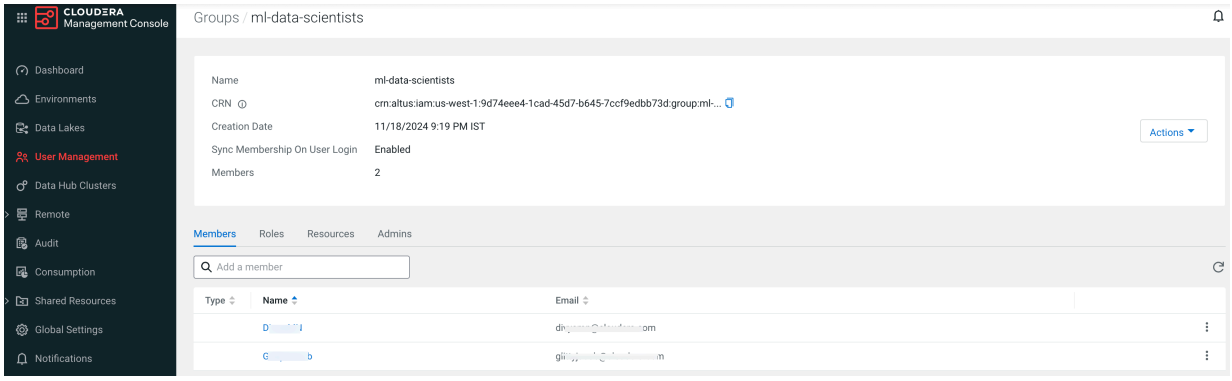

   a. Sync users

      Whenever you make changes to user and group mappings, make sure to sync the mappings with the authentication layer. In User Management > Actions, click Sync Users, and select the environment.

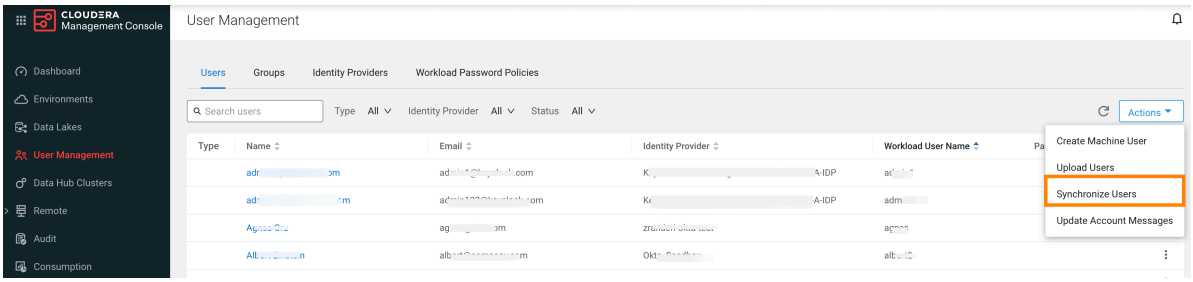

4. IDBroker

   IDBroker allows an authenticated and authorized user to exchange a set of credentials or a token for cloud vendor access tokens. You can also view and update the IDBroker mappings at this location. IDBroker mappings can be accessed through Environments > Manage Access. Click on the IDBroker Mappings tab. Click **Edit** to edit or add mappings. When finished, synchronize the mappings to push the settings from Cloudera Data Platform to the IDBroker instance running inside the Data Lake of the environment.

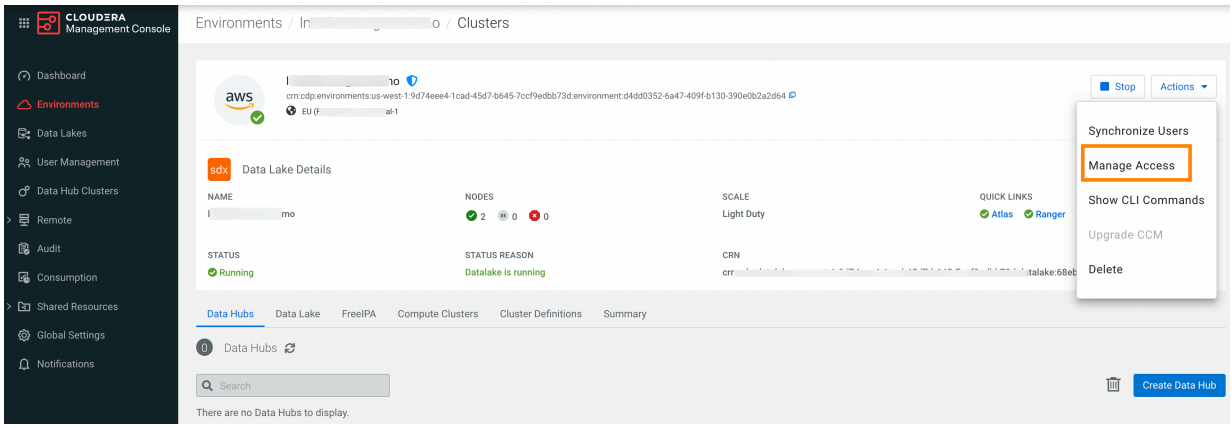

   At this point, Cloudera Data Platform resources can access the AWS S3 buckets or Azure ADLS storage.

5. Ranger

   To get admin access to Ranger, users need the EnvironmentAdmin role, and that role must be synced with the environment.

   a. Click Environments > Env > Actions > Manage Access > Add User
   b. Select EnvironmentAdmin resource role.
   c. Click Update Roles
   d. On the Environments page for the environment, in Actions, select Synchronize Users to FreeIPA.

   The permissions are now synchronized to the Data Lake, and you have admin access to Ranger.

   Update permissions in Ranger

   a. In Environments > Env > Data Lake Cluster, click Ranger.
   b. Select the Hadoop SQL service, and check that the users and groups have sufficient permissions to access databases, tables, columns, and urls.

   For example, a user can be part of these policies:

   - all - database,table,column
   - all - url

   This completes all configuration needed for Cloudera AI to communicate with the Data Lake.
6. Cloudera AI User Setup

   Now, Cloudera AI is able to communicate with the Data Lake. There are two steps to get the user ready to work.

   a. In Environments > Environment name > Actions > Manage Access > Add user, the Admin selects MLUser resource role for the user.
   b. The User logs into the workbench in Cloudera AI Workbench > Workbench name, click Launch Workbench.

   The user can now access the workbench.

Related Information

AWS environments
Azure environments

Example: Connect a Spark session to the Data Lake

After the Administrator sets up the correct permissions, you can access the Data Lake from your project, as this example shows.

Before you begin

Make sure you have access to a Data Lake containing your data.

The s3 or abfs path can be retrieved from the environment's cloud storage. To read from this location, in the Cloudera AI Workbenches UI, select the name of the environment, and in the Action menu, select Manage Access IdBroker Mappings . Ensure that the user (or group) has been assigned an AWS IAM role that can read this location. For Fine Grained Access Control (Raz) enabled environments, go to Ranger to ensure that the user (or group) has access through an appropriate s3 or adls policy.

Procedure

1. Create a project in your Cloudera AI Workbench.

2. Create a file named spark-defaults.conf, or update the existing file with the property:
   - For S3: spark.yarn.access.hadoopFileSystems=s3a://STORAGE LOCATION OF ENV>
   - For ADLS: spark.yarn.access.hadoopFileSystems=abfs://STORAGE CONTAINER OF ENV>@STORAGE ACCOUNT OF ENV>

   Use the same location you defined in *Setup Data Lake Access*.
3. Start a session (Python or Spark) and start a Spark session.

Results

Setting up the project looks like this:

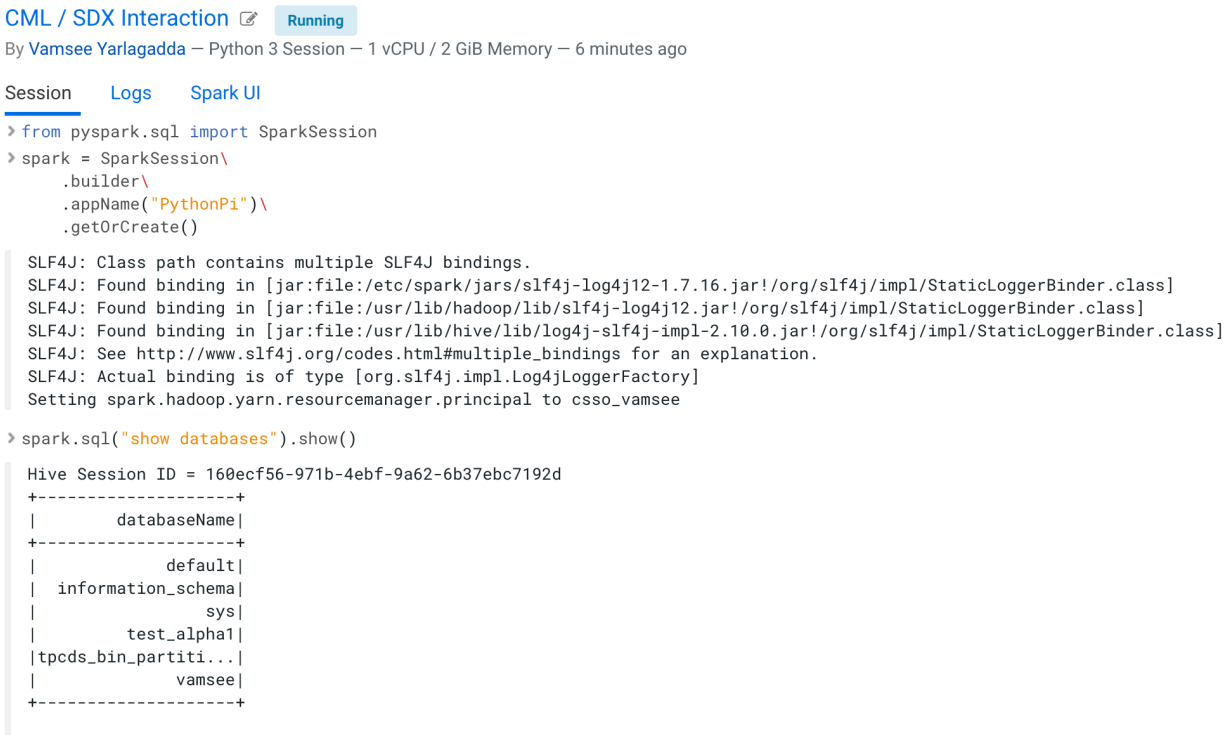

Now you can run Spark SQL commands. For example, you can:

- Create a database `foodb`.

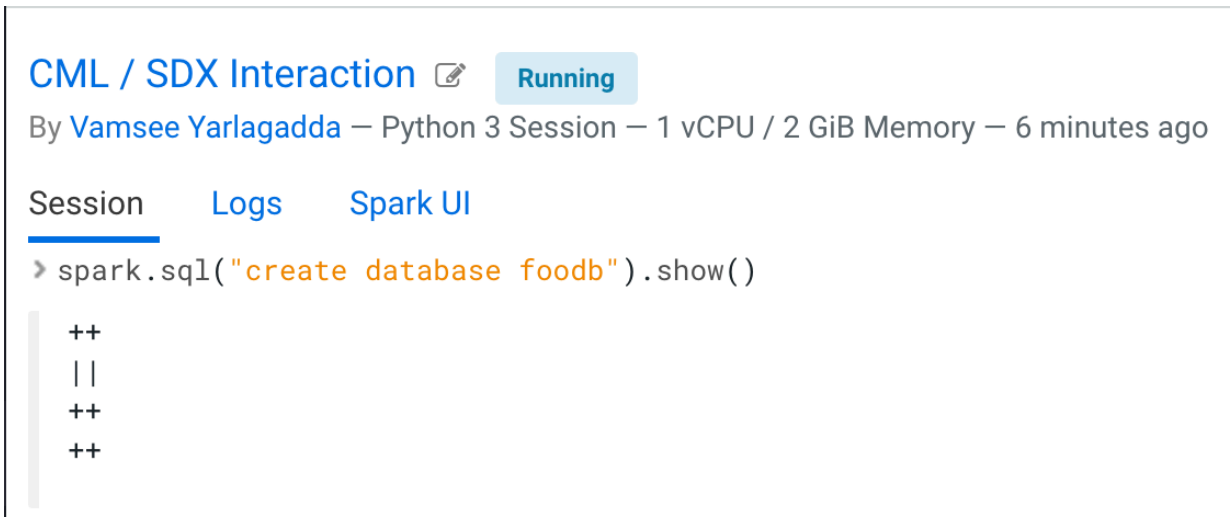

- List databases and tables.

CML / SDX Interaction Running

By Vamsee Yarlagadda — Python 3 Session — 1 vCPU / 2 GiB Memory — 6 minutes ago

Session Logs Spark UI

```
> spark.sql("use foodb").show()
```

```
++
||
++
++
```

```
> spark.sql("show tables").show()
```

```
+--------+---------+-----------+
|database|tableName|isTemporary|
+--------+---------+-----------+
+--------+---------+-----------+
```

- Create a table `bartable`.

CML / SDX Interaction Running

By Vamsee Yarlagadda — Python 3 Session — 1 vCPU / 2 GiB Memory — 6 minutes ago

Session Logs Spark UI

```
> spark.sql("create table bartable(id int)").show()
```

```
++
||
++
++
```

```
> spark.sql("show tables").show()
```

```
+--------+---------+-----------+
|database|tableName|isTemporary|
+--------+---------+-----------+
|   foodb| bartable|      false|
+--------+---------+-----------+
```

- Insert data into the table.

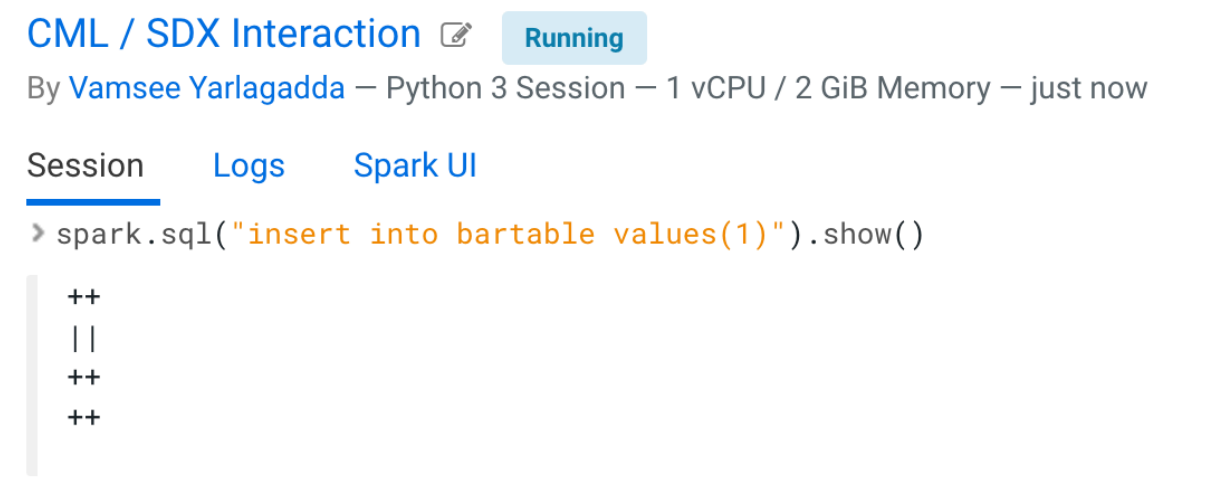

- Query the data from the table.

Related Information

Setup Data Lake Access

Use Direct Reader Mode with PySpark

You can use the Direct Reader Mode when your data has table-level access control, and does not have row-level security or column level masking (fine grained access.)

Before you begin

Obtain the location of the data lake before starting this task.

Continue with the next steps to set up the connection.

Procedure

Replace s3a://demo-aws-2/ in the code sample below with the correct S3 bucket location. This sets the value for DATALAKE_DIRECTORY.

The DATALAKE_DIRECTORY value is set for the spark.yarn.access.hadoopFileSystems property in the corresponding config statement.

Example

Note: The following example works for managed as well as external Hive tables, however, only for Spark 2.

```
from pyspark.sql import SparkSession
# Change to the appropriate Datalake directory location
DATALAKE_DIRECTORY = "s3a://demo-aws-2/"

spark = SparkSession\
.builder\
.appName("CDW-CML-Spark-Direct")\
.config("spark.sql.hive.hwc.execution.mode","spark")\
.config("spark.sql.extensions","com.qubole.spark.hiveacid.HiveAcidAutoCon
vertExtension")\
.config("spark.kryo.registrator","com.qubole.spark.hiveacid.util.HiveAcidKy
roRegistrator")\
.config("spark.yarn.access.hadoopFileSystems", DATALAKE_DIRECTORY)\
.config("spark.jars", "/opt/spark/optional-lib/hive-warehouse-connector-ass
embly.jar")\
.getOrCreate()
### The following commands test the connection
spark.sql("show databases").show()
spark.sql("describe formatted test_managed").show()
spark.sql("select * from test_managed").show()
spark.sql("describe formatted test_external").show()
spark.sql("select * from test_external").show()
```

Note: The following example works for both Spark 2 and Spark 3, but only works for external Hive tables, and does not support Hive managed tables.

```
from pyspark.sql import SparkSession
# Change to the appropriate Datalake directory location
DATALAKE_DIRECTORY = "s3a://demo-aws-2/"
spark = SparkSession\
.builder\
.appName("CDW-CML-Spark-Direct")\
.config("spark.sql.hive.hwc.execution.mode","spark")\
.config("spark.yarn.access.hadoopFileSystems", DATALAKE_DIRECTORY)\
.getOrCreate()

spark.sql("show databases").show()
spark.sql("describe formatted test_external").show()
spark.sql("select * from test_external").show()
```

Use Direct Reader Mode with SparklyR

Before you begin

You need to add the location of your data tables in the example below.

Procedure

In your session, open the workbench and add the following code.

Example

```
#Run this once
#install.packages(“sparklyr")

library(sparklyr)
config <- spark_config()

config$spark.security.credentials.hiveserver2.enabled="false"
config$spark.datasource.hive.warehouse.read.via.llap="false"
config$spark.sql.hive.hwc.execution.mode="spark"
#config$spark.datasource.hive.warehouse.read.jdbc.mode="spark"

# Required setting for HWC-direct reader - the hiveacid sqlextension does
 the automatic
# switch between reading through HWC (for managed tables) or spark-native
 (for external)
# depending on table type.
config$spark.sql.extensions="com.qubole.spark.hiveacid.HiveAcidAutoConver
tExtension"
config$spark.kryo.registrator="com.qubole.spark.hiveacid.util.HiveAcidKyroRe
gistrator"

# Pick the correct jar location by using Terminal Access
# to file the exact file path/name of the hwc jar under /usr/lib/hive_ware
house_connector
config$sparklyr.jars.default <- "/opt/spark/optional-lib/hive-warehouse-conn
ector-assembly.jar"
# File system read access - this s3a patha is available from the environment
 Cloud Storage.
# To read from this location go to Environment->Manage Access->IdBroker Map
pings
#     ensure that the user (or group) has been assigned an AWS IAM role that
 can
#.     read this location.
# For Fine Grained Access Control (Raz) enabled environments, go to Ranger
 to ensure
# that the user (or group) has access through an appropriate s3 or adls poli
cy
#config$spark.yarn.access.hadoopFileSystems="s3a://demo-aws-2/"
config$spark.yarn.access.hadoopFileSystems="s3a://demo-aws-2/datalake/war
ehouse/tablespace/managed/hive"

sc <- spark_connect(config = config)

intDf1 <- sparklyr::spark_read_table(sc, 'foo')
sparklyr::sdf_collect(intDf1)

intDf1 <- sparklyr::spark_read_table(sc, 'foo_ext')
sparklyr::sdf_collect(intDf1)

spark_disconnect(sc)

# Optional configuration - only needed if the table is in a private Data Ca
talog of CDW
#config$spark.sql.hive.hiveserver2.jdbc.url="jdbc:hive2://hs2-aws-2-hive-vi
z.env-j2ln9x.dw.ylcu-atmi.cloudera.site/default;transportMode=http;httpPath=
cliservice;ssl=true;retries=3;user=priyankp;password=!Password1"

#ss <- spark_session(sc)
#hive <- invoke_static(sc,"com.hortonworks.hwc.HiveWarehouseSession","ses
sion",ss)%>%invoke("build")
#df <- invoke(hive,"execute","select * from default.foo limit 199")
```

```
#sparklyr::sdf_collect(df)
```

Create an Iceberg data connection

Cloudera AI supports data connections to Iceberg data lakes.

You can connect automatically, using the Cloudera AI-provided data connection. If necessary, you can also set up a manual connection using the snippet provided below. To connect with Iceberg, you must use Spark 3.

In your project:

1. In Project Settings, view the Data Connections tab. There should be an available data connection to a Spark Data Lake.
2. Start a New Session.
3. Select Enable Spark, and choose Spark 3 from the dropdown.
4. Select Start Session.
5. In the Connection Code Snippet UI, select the Spark Data Lake connection.
6. In the code window, select Copy Code, then Close.
7. Select FileNew File , and paste the Cloudera AI-provided code snippet into the file.
8. Select Run.

You see a list of available databases in the data lake.

Instead of using the Cloudera AI-provided data connection, you can also manually connect to a Spark Data Lake using the Spark command as shown in the snippet below.

Make sure to set the following parameters:

- DATALAKE_DIRECTORY
- Valid database and table name in the describe formatted SQL command.

```
from pyspark.sql import SparkSession
# Change to the appropriate Datalake directory location
DATALAKE_DIRECTORY = "s3a://your-aws-demo/"

spark = (
  SparkSession.builder.appName("MyApp")
  .config("spark.sql.hive.hwc.execution.mode", "spark")
  .config("spark.sql.extensions", "com.qubole.spark.hiveacid.HiveAcidAutoCon
vertExtension, org.apache.iceberg.spark.extensions.IcebergSparkSessionExtens
ions")
  .config("spark.sql.catalog.spark_catalog.type", "hive")
  .config("spark.sql.catalog.spark_catalog", "org.apache.iceberg.spark.Spar
kSessionCatalog")
  .config("spark.yarn.access.hadoopFileSystems", DATALAKE_DIRECTORY)
  .config("spark.hadoop.iceberg.engine.hive.enabled", "true")
  .config("spark.jars", "/opt/spark/optional-lib/iceberg-spark-runtime.jar,
 /opt/spark/optional-lib/iceberg-hive-runtime.jar")
  .getOrCreate()
  )

spark.sql("show databases").show()
spark.sql("describe formatted <database_name>.<table_name>").show()
```

Accessing Data from HDFS

There are many ways to access HDFS data from R, Python, and Scala libraries. The following code samples assume that appropriate permissions have been set up in IDBroker or Ranger/Raz. The samples below demonstrate how to count the number of occurrences of each word in a simple text file in HDFS.

1. Navigate to your project and click Open Workbench.
2. Create a file called sample_text_file.txt and save it to your project in the data folder.
3. Write this file to HDFS. You can do this in one of the following ways:
   - Click Terminal above the Cloudera AI console and enter the following command to write the file to HDFS:

     ```
     hdfs dfs -put data/sample_text_file.txt s3a://<s3_data_directory>/tmp
     ```

     OR
   - Use the workbench command prompt:

     Python Session

     ```
     !hdfs dfs -put data/sample_text_file.txt s3a://<s3_data_directory>/tmp
     ```

     R Session

     ```
     system("hdfs dfs -put data/tips.csv /user/hive/warehouse/tips/")
     ```

The following examples use Python and Scala to read sample_text_file.txt from HDFS (written above) and perform the count operation on it.

Python

```
from __future__ import print_function
import sys, re
from operator import add
from pyspark.sql import SparkSession

spark = SparkSession\
  .builder\
  .appName("PythonWordCount")\
  .getOrCreate()

# Access the file
lines = spark.read.text("s3a://<s3_data_directory>/tmp/sample_text_file.txt
").rdd.map(lambda r: r[0])
counts = lines.flatMap(lambda x: x.split(' ')) \
  .map(lambda x: (x, 1)) \
  .reduceByKey(add) \
  .sortBy(lambda x: x[1], False)
output = counts.collect()
for (word, count) in output:
  print("%s: %i" % (word, count))

spark.stop()
```

Scala

```
//count lower bound
val threshold = 2

// read the file added to hdfs
```

```
val tokenized = sc.textFile("s3a://<s3_data_directory>/tmp/sample_text_fil
e.txt").flatMap(_.split(" "))

// count the occurrence of each word
val wordCounts = tokenized.map((_ , 1)).reduceByKey(_ + _)

// filter out words with fewer than threshold occurrences
val filtered = wordCounts.filter(_._2 >= threshold)

System.out.println(filtered.collect().mkString(","))
```

Connecting to Cloudera Data Warehouse

The Data Connection Snippet feature now suggests using the cml.data library to connect to Cloudera Data Warehouse virtual warehouses - these code snippets pop up as suggestions for every new session in a project. For further information, see *Using data connection snippets*.

However, if you would still like to use raw Python code to connect, follow the below details.

You can access data stored in the data lake using a Cloudera Data Warehouse cluster from a Cloudera AI Workbench, using the impyla Python package.

Configuring the connection

The Cloudera Data Warehouse connection requires a WORKLOAD_PASSWORD that can be configured following the steps described in *Setting the workload password*, linked below.

The VIRTUAL_WAREHOUSE_HOSTNAME can be extracted from the JDBC URL that can be found in Cloudera Data Warehouse, by selecting the Option menu > Copy JDBC String on a Virtual Warehouse.

For example, if the JDBC string copied as described above is:

```
jdbc:impala//<your-vw-host-name.site>/default;transportMode=http;httpPath=cl
iservice;socketTimeout=60;ssl=true;auth=browser;
```

Then, the extracted hostname to assign to the VWH_HOST is: <your-vw-host-name.site>

Connection code

Enter this code in your project file, and run it in a session.

```
# This code assumes the impyla package to be installed.
# If not, please pip install impyla

from impala.dbapi import connect
import os
USERNAME=os.getenv(HADOOP_USER_NAME)
PASSWORD=os.getenv(WORKLOAD_PASSWORD)
VWH_HOST = "<<VIRTUAL_WAREHOUSE_HOSTNAME>>"
VWH_PORT = 443
conn = connect(host=VWH_HOST, port=VWH_PORT, auth_mechanism="LDAP", user=USE
RNAME,  password=PASSWORD, use_http_transport=True, http_path="cliservice",
use_ssl=True)

dbcursor = conn.cursor()
dbcursor.execute("<<INSERT SQL QUERY HERE>>")
for row in dbcursor:
  print(row)

#Sample pandas code
```

```
#from impala.util import as_pandas
#import pandas
#dbcursor = conn.cursor()
#dbcursor.execute("<<INSERT SQL QUERY HERE>>")
#tables = as_pandas(cursor)
#tables
#dbcursor.close()
```

Related Information

Setting the workload password

Using data connection snippets

Accessing data with Spark

When you are using Cloudera Data Warehouse, you can use Java Database Connectivity (JDBC).

JDBC is useful in the following cases:

1. Use JDBC connections when you have fine-grained access.
2. If the scale of data sent over the wire is on the order of tens of thousands of rows of data.

Add the Python code as described below, in the Session where you want to utilize the data, and update the code with the data location information.

Permissions

In addition, check with the Administrator that you have the correct permissions to access the data lake. You will need a role that has read access only. For more information, see *Setup Data Lake Access*.

Set up a JDBC Connection

When using a JDBC connection, you read through a virtual warehouse that has Hive or Impala installed. You need to obtain the JDBC connection string, and paste it into the script in your session.

1. In Cloudera Data Warehouse, go to the Hive database containing your data.
2. From the kebab menu, click Copy JDBC URL.
3. Paste it into the script in your session.
4. You also have to enter your user name and password in the script. You should set up environmental variables to store these values, instead of hardcoding them in the script.

Related Information

Setup Data Lake Access

Use JDBC Connection with PySpark

PySpark can be used with JDBC connections, but it is not recommended. The recommended approach is to use Impyla for JDBC connections. For more information, see *Connect to CDW*.

Procedure

1. In your session, open the workbench and add the following code.
2. Obtain the JDBC connection string, as described above, and paste it into the script where the “jdbc” string is shown. You will also need to insert your user name and password, or create environment variables for holding those values.

Example

This example shows how to read external Hive tables using Spark and a Hive Virtual Warehouse.

```
from pyspark.sql import SparkSession
```

```
from pyspark_llap.sql.session import HiveWarehouseSession

spark = SparkSession\
.builder\
.appName("CDW-CML-JDBC-Integration")\
.config("spark.security.credentials.hiveserver2.enabled","false")\
.config("spark.datasource.hive.warehouse.read.jdbc.mode", "client")\
.config("spark.sql.hive.hiveserver2.jdbc.url",
"jdbc:hive2://hs2-aws-2-hive-viz.env-j2ln9x.dw.ylcu-atmi.cloudera.site/def
ault;\
transportMode=http;httpPath=cliservice;ssl=true;retries=3;\
user=<username>;password=<password>")\
.getOrCreate()

hive = HiveWarehouseSession.session(spark).build()
hive.showDatabases().show()
hive.setDatabase("default")
hive.showTables().show()
hive.sql("select * from foo").show()
```

Related Information

Connecting to Cloudera Data Warehouse

Connect to a Cloudera Data Hub cluster

The Data Connection Snippet feature helps simplify the user experience by abstracting the complexity of creating and configuring a data connection.

You can set up a Data Hub Impala or Hive connection by following the documentation: *Set up a data connection to Cloudera Data Hub*.

However, if you would still like to use raw Python code to connect, follow this Python example:

```
from impala.dbapi import connect

#Example connection string:
# jdbc:hive2://my-test-master0.eng-ml-i.svbr-nqvp.int.cldr.work/;ssl=true;
transportMode=http;httpPath=my-test/cdp-proxy-api/hive

USERNAME=os.getenv(HADOOP_USER_NAME)
PASSWORD=os.getenv(WORKLOAD_PASSWORD)

conn = connect(
    host = "my-test-master0.eng-ml-i.svbr-nqvp.int.cldr.work",
    port = 443,
    auth_mechanism = "LDAP",
    use_ssl = True,
    use_http_transport = True,
    http_path = "my-test/cdp-proxy-api/hive",
    user = USERNAME,
    password = PASSWORD)
cursor = conn.cursor()
cursor.execute("<<INSERT SQL QUERY HERE>>")

for row in cursor:
    print(row)
cursor.close()
conn.close()
```

Related Information

Set up a data connection to Cloudera DataHub

Connecting to external Amazon S3 buckets

Every language in Cloudera AI has libraries available for uploading to and downloading from Amazon S3.

To work with external S3 buckets in Python, do the following:

- Add your Amazon Web Services access keys to your project's environment variables as AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY.
- Add your Ozone S3 gateway to the environment variables as OZONE_S3_GATEWAY.

Python

```
# Install Boto to the project
!pip3 install boto3

# Make sure below environment variables are set
# ozone s3 gateway : os.environ['OZONE_S3_GATEWAY']
# s3 keys from os.environ['AWS_ACCESS_KEY_ID'] and os.environ['AWS_SECRET
_ACCESS_KEY']

import os
import boto3

# Use Boto to connect to S3 and get a list of objects from a bucket
conn = boto3.session.Session()

s3g = os.environ['OZONE_S3_GATEWAY']
access_key = os.environ['AWS_ACCESS_KEY_ID']
secret_key = os.environ['AWS_SECRET_ACCESS_KEY']

s3_client = conn.client(
    service_name='s3',
    endpoint_url=s3g
)


test_bucket = 'testozones3'
s3_client.create_bucket(Bucket=test_bucket)
all_buckets = s3_client.list_buckets()
print(f"All S3 Buckets are {[i['Name'] for i in all_buckets['Buckets']]}")

s3_client.put_object(Bucket=test_bucket, Key='README.md')

all_objs = s3_client.list_objects(Bucket=test_bucket)
print(f"All keys in {bucket_name} are {[i['Key']for i in all_objs['Contents'
]]}")

s3_client.get_object(Bucket=test_bucket, Key='README.md')

ssl = "true" if s3g.startswith("https") else "false"
s3a_path = f"s3a://{test_bucket}/"

hadoop_opts = f"-Dfs.s3a.access.key='{access_key}' -Dfs.s3a.secret.key='{
secret_key}' -Dfs.s3a.endpoint='{s3g}' -Dfs.s3a.connection.ssl.enabled={ssl}
  -Dfs.s3a.path.style.access=true"

!hdfs dfs {hadoop_opts} -ls "s3a://{test_bucket}/"
```

Connect to External SQL Databases

Every language in Cloudera AI has multiple client libraries available for SQL databases.

If your database is behind a firewall or on a secure server, you can connect to it by creating an SSH tunnel to the server, then connecting to the database on localhost.

If the database is password-protected, consider storing the password in an environmental variable to avoid displaying it in your code or in consoles. The examples below show how to retrieve the password from an environment variable and use it to connect.

Python

You can access data using pyodbc or SQLAlchemy

```
# pyodbc lets you make direct SQL queries.
!wget https://pyodbc.googlecode.com/files/pyodbc-3.0.7.zip
!unzip pyodbc-3.0.7.zip
!cd pyodbc-3.0.7;python setup.py install --prefix /home/cdsw
import os

# See http://www.connectionstrings.com/ for information on how to construct
 ODBC connection strings.
db = pyodbc.connect("DRIVER={PostgreSQL Unicode};SERVER=localhost;PORT=54
32;DATABASE=test_db;USER=cdswuser;OPTION=3;PASSWORD=%s" % os.environ["POSTGR
ESQL_PASSWORD"])
cursor = cnxn.cursor()
cursor.execute("select user_id, user_name from users")

# sqlalchemy is an object relational database client that lets you make data
base queries in a more Pythonic way.
!pip install sqlalchemy
import os

import sqlalchemy
from sqlalchemy.orm import sessionmaker
from sqlalchemy import create_engine
db = create_engine("postgresql://cdswuser:%s@localhost:5432/test_db" % os.en
viron["POSTGRESQL_PASSWORD"])
session = sessionmaker(bind=db)
user = session.query(User).filter_by(name='ed').first()
```

R

You can access remote databases with dplyr.

```
install.packages("dplyr")
library("dplyr")
db <- src_postgres(dbname="test_db", host="localhost", port=5432, user="cds
wuser", password=Sys.getenv("POSTGRESQL_PASSWORD"))
flights_table <- tbl(db, "flights")
select(flights_table, year:day, dep_delay, arr_delay)
```

Accessing Ozone storage

In Cloudera AI you can connect Cloudera AI to the Ozone object store using a script or command line commands.

Connecting to Ozone filesystem

In Cloudera AI, you can connect Spark to the Ozone object store with a script.

This script, in Scala, counts the number of word occurrences in a text file. The key point in this example is to use the following string to refer to the text file: ofs://omservice1/s3v/hivetest/spark/jedi_wisdom.txt

Word counting example in Scala

```
import sys.process._

// Put the input file into Ozone
//"hdfs dfs -put data/jedi_wisdom.txt ofs://omservice1/s3v/hivetest/spark" !
// Set the following spark setting in the file "spark-defaults.conf" on
 the Cloudera AI session using terminal
//spark.yarn.access.hadoopFileSystems=ofs://omservice1/s3v/hivetest
//count lower bound
val threshold = 2
// this file must already exist in hdfs, add a
// local version by dropping into the terminal.
val tokenized = sc.textFile("ofs://omservice1/s3v/hivetest/spark/jedi_wisd
om.txt").flatMap(_.split(" "))
// count the occurrence of each word
val wordCounts = tokenized.map((_ , 1)).reduceByKey(_ + _)
// filter out words with fewer than threshold occurrences
val filtered = wordCounts.filter(_._2 >= threshold)
System.out.println(filtered.collect().mkString(","))
```

Accessing local files in Ozone

You can access files in Ozone on a local file system using hdfscli. This method works with both legacy engines and runtime sessions.

The following commands enable a Cloudera AI session to connect to Ozone using the OFS protocol.

1. Put the input file into Ozone:

   ```
   hdfs dfs -put data/jedi_wisdom.txt ofs://omservice1/s3v/hivetest/spark
   ```

2. List the files in Ozone:

   ```
   hdfs dfs -ls ofs://omservice1/s3v/hivetest/
   ```

3. Download file from ozone to local:

   ```
   hdfs dfs -copyToLocal ofs://omservice1/s3v/hivetest/spark data/jedi_wisd
   om.txt
   ```

Cloudera AI

Managing Jobs and Pipelines in Cloudera AI

Date published: 2020-07-16
Date modified: 2024-11-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a Job 4

Creating a Pipeline 5

Viewing Job History 6

Legacy Jobs API (Deprecated) 6

Creating a Job

This topic describes how to automate analytics workloads with a built-in job and pipeline scheduling system that supports real-time monitoring, job history, and email alerts.

A job automates the action of launching an engine, running a script, and tracking the results, all in one batch process. Jobs are created within the purview of a single project and can be configured to run on a recurring schedule. You can customize the engine environment for a job, set up email alerts for successful or failed job runs, and email the output of the job to yourself or a colleague.

Jobs are created within the scope of a project. When you create a job, you will be asked to select a script to run as part of the job, and create a schedule for when the job should run. Optionally, you can configure a job to be dependent on another existing job, thus creating a pipeline of tasks to be accomplished in a sequence. Note that the script files and any other job dependencies must exist within the scope of the same project.

For Cloudera AI UI

1. Navigate to the project for which you want to create a job.
2. On the left-hand sidebar, click Jobs.
3. Click New Job.
4. Enter a Name for the job.
5. In Run Job as, if the job is to run in a service account, select Service Account and choose the account from the dropdown menu.
6. In Script, select a script to run for this job by clicking on the folder icon. You will be able to select a script from a list of files that are already part of the project. To upload more files to the project, see *Managing Project Files*.
7. In Arguments, enter arguments to provide to the script.

   Use the environment variable JOB_ARGUMENTS to access the arguments in a runtime-agnostic way.

   For non-PBJ Runtimes, the contents of this field are available as standard command line arguments.
8. Depending on the code you are running, select an Engine Kernel for the job from one of the following option: Python 3.
9. Select a Schedule for the job runs from one of the following options.
   - Manual - Select this option if you plan to run the job manually each time.
   - Recurring - Select this option if you want the job to run in a recurring pattern every X minutes, or on an hourly, daily, weekly or monthly schedule. Set the recurrence interval with the drop-down buttons.

     As an alternative, select Use a cron expression to enter a Unix-style cron expression to set the interval. The expression must have five fields, specifying the minutes, hours, day of month, month, and day of week. If the cron expression is deselected, the schedule indicated in the drop-down settings takes effect.
   - Dependent - Use this option when you are building a pipeline of jobs to run in a predefined sequence. From a dropdown list of existing jobs in this project, select the job that this one should depend on. Once you have configured a dependency, this job will run only after the preceding job in the pipeline has completed a successful run.
10. Select an Resource Profile to specify the number of cores and memory available for each session.
11. Enter an optional timeout value in minutes.
12. Click Set environment variables if you want to set any values to override the overall project environment variables.
13. Specify a list of Job Report Recipients to whom you can send email notifications with detailed job reports for job success, failure, or timeout. You can send these reports to yourself, your team (if the project was created under a team account), or any other external email addresses.
14. Add any Attachments such as the console log to the job reports that will be emailed.

15. Click Create Job.

You can use the API v2 to schedule jobs from third partly workflow tools. For details, see *Using the Jobs API* as well as the *Cloudera AI APIv2* tab.

For Cloudera AI APIv2

To create a job using the API, follow the code below:

```
job_body = cmlapi.CreateJobRequest()

# name and script
job_body.name = "my job name"
job_body.script = "pi.py"

# arguments
job_body.arguments = "arg1 arg2 \"all arg 3\""

# engine kernel
job_body.kernel = "python3" # or "r", or "scala"

# schedule
# manual by default
# for recurring/cron:
job_body.schedule = "* * * * 5" # or some valid cron string

# for dependent (don't set both parent_job_id and schedule)
job_body.parent_job_id = "abcd-1234-abcd-1234"

# resource profile (cpu and memory can be floating point for partial)
job_body.cpu = 1 # one cpu vcore
job_body.memory = 1 # one GB memory
job_body.nvidia_gpu = 1 # one nvidia gpu, cannot do partial gpus

# timeout
job_body.timeout = 300 # this is in seconds

# environment
job_body.environment = {"MY_ENV_KEY": "MY_ENV_VAL", "MY_SECOND_ENV_KEY":
"MY_SECOND_ENV_VAL"}

# attachment
job_body.attachments = ["report/1.txt", "report/2.txt"] # will attach /
home/cdsw/report/1.txt and /home/cdsw/report/2.txt to emails

# After setting the parameters above, create the job:
client = cmlapi.default_client("host", "api key")
client.create_job(job_body, project_id="id of project to create job in")
```

For some more examples of commands related to jobs, see: *Using the Jobs API*.

Related Information

Managing Project Files

Using the Jobs API

Legacy Jobs API (Deprecated)

Creating a Pipeline

This topic describes how to create a scheduled pipeline of jobs within a project.

About this task

As data science projects mature beyond ad hoc scripts, you might want to break them up into multiple steps. For example, a project may include one or more data acquisition, data cleansing, and finally, data analytics steps. For such projects, Cloudera AI allows you to schedule multiple jobs to run one after another in what is called a pipeline, where each job is dependent on the output of the one preceding it.

The Jobs overview presents a list of all existing jobs created for a project along with a dependency graph to display any pipelines you've created. Job dependencies do not need to be configured at the time of job creation. Pipelines can be created after the fact by modifying the jobs to establish dependencies between them. From the job overview, you can modify the settings of a job, access the history of all job runs, and view the session output for individual job runs.

Let us take an example of a project that has two jobs, Read Weblogs and Write Weblogs. Given that you must read the data before you can run analyses and write to it, the Write Weblogs job should only be triggered after the Read Weblogs job completes a successful run. To create such a two-step pipeline:

Procedure

1. Navigate to the project where the Read Weblogs and Write Weblogs jobs were created.
2. Click Jobs.
3. From the list of jobs, select *WRITE WEBLOGS*.
4. Click the Settings tab.
5. Click on the Schedule dropdown and select Dependent. Select Read Weblogs from the dropdown list of existing jobs in the project.
6. Click Update Job.

Viewing Job History

This topics shows you how to view the history for jobs run within a project.

Procedure

1. Navigate to the project where the job was created.
2. Click Jobs.
3. Select the relevant job.
4. Click the History tab. You will see a list of all the job runs with some basic information such as who created the job, run duration, and status. Click individual runs to see the session output for each run.

Legacy Jobs API (Deprecated)

This topic demonstrates how to use the legacy API to launch jobs.

Cloudera AI exposes a legacy REST API that allows you to schedule jobs from third-party workflow tools. You must authenticate yourself before you can use the legacy API to submit a job run request. The Jobs API supports HTTP Basic Authentication, accepting the same users and credentials as Cloudera AI.

Note: The Jobs API is now deprecated. See *Cloudera AI API v2* and API v2 usage for the successor API.

Legacy API Key Authentication

Cloudera recommends using your legacy API key for requests instead of your actual username/password so as to avoid storing and sending your credentials in plaintext. The legacy API key is a randomly generated token that is unique to each user. It must be treated as highly sensitive information because it can be used to start jobs using the API. To look up your Cloudera AI legacy API key:

1. Sign in to Cloudera AI.
2. From the upper right drop-down menu, switch context to your personal account.
3. Click Settings.
4. Select the API Key tab.

The following example demonstrates how to construct an HTTP request using the standard basic authentication technique. Most tools and libraries, such as Curl and Python Requests, support basic authentication and can set the required Authorization header for you. For example, with curl you can pass the legacy API key to the --user flag and leave the password field blank.

```
curl -v -XPOST http://cdsw.example.com/api/v1/<PATH_TO_JOB> --user
 "<LEGACY_API_KEY>:"
```

To access the API using a library that does not provide Basic Authentication convenience methods, set the request's Authorization header to Basic *<LEGACY_API_KEY_ENCODED_IN_BASE64>*. For example, if your API key is uysgxtj7jzkps96njextnxxmq05usp0b, set Authorization to Basic dXlzZ3h0ajdqemtwczk2bmpleHRueHhtcTA1dXNwMGI6.

Starting a Job Run Using the API

Once a job has been created and configured through the Cloudera AI web application, you can start a run of the job through the legacy API. This will constitute sending a POST request to a job start URL of the form: http://cdsw.example.com/api/v1/projects/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>/start.

To construct a request, use the following steps to derive the username, project name, and job ID from the job's URL in the web application.

1. Log in to the Cloudera AI web application.
2. Switch context to the team/personal account where the parent project lives.
3. Select the project from the list.
4. From the project's Overview, select the job you want to run. This will take you to the job Overview page. The URL for this page is of the form: http://cdsw.example.com/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>.
5. Use the $USERNAME, $PROJECT_NAME, and $JOB_ID parameters from the job Overview URL to create the following job start URL: http://cdsw.example.com/api/v1/projects/<$USERNAME>/<$PROJECT_NAME>/jobs/<$JOB_ID>/start.

   For example, if your job Overview page has the URL http://cdsw.example.com/alice/sample-project/jobs/123, then a sample POST request would be of the form:

   ```
   curl -v -XPOST http://cdsw.example.com/api/v1/projects/alice/sample-proj
   ect/jobs/123/start    \
   --user "<API_KEY>:" --header "Content-type: application/json"
   ```

   Note that the request must have the Content-Type header set to application/json, even if the request body is empty.

Setting Environment Variables

You can set environment variables for a job run by passing parameters in the API request body in a JSON-encoded object with the following format.

```
{
```

```
    "environment": {
        "ENV_VARIABLE": "value 1",
        "ANOTHER_ENV_VARIABLE": "value 2"
    }
}
```

The values set here will override the defaults set for the project and the job in the web application. This request body is optional and can be left blank.

Be aware of potential conflicts with existing defaults for environment variables that are crucial to your job, such as PATH and the Cloudera AI variables.

Sample Job Run

As an example, let's assume user Alice has created a project titled Risk Analysis. Under the Risk Analysis project, Alice has created a job with the ID, 208. Using curl, Alice can use her API Key (uysgxtj7jzkps96njextnxxmq05usp0b) to create an API request as follows:

```
curl -v -XPOST http://cdsw.example.com/api/v1/projects/alice/risk-analysis/j
obs/208/start      \
--user "uysgxtj7jzkps96njextnxxmq05usp0b:" --header "Content-type: applicat
ion/json"           \
--data '{"environment": {"START_DATE": "2017-01-01", "END_DATE": "2017-01-
31"}}'
```

In this example, START_DATE and END_DATE are environment variables that are passed as parameters to the API request in a JSON object.

In the resulting HTTP request, curl automatically encodes the Authorization request header in base64 format.

```
* Connected to cdsw.example.com (10.0.0.3) port 80 (#0)
* Server auth using Basic with user 'uysgxtj7jzkps96njextnxxmq05usp0b'
> POST /api/v1/projects/alice/risk-analysis/jobs/21/start HTTP/1.1
> Host: cdsw.example.com
> Authorization: Basic dXlzZ3h0ajdqemtwczk2bmpleHRueHhtcTA1dXNwMGI6
> User-Agent: curl/7.51.0
> Accept: */*
> Content-type: application/json
>
< HTTP/1.1 200 OK
< Access-Control-Allow-Origin: *
< Content-Type: application/json; charset=utf-8
< Date: Mon, 10 Jul 2017 12:00:00 GMT
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
<
{
  "engine_id": "cwg6wclmg0x482u0"
}
```

You can confirm that the job was started by going to the Cloudera AI web application.

Starting a Job Run Using Python

To start a job run using Python, Cloudera recommends using Requests, an HTTP library for Python; it comes with a convenient API that makes it easy to submit job run requests to Cloudera AI. Extending the Risk Analysis example from the previous section, the following sample Python code creates an HTTP request to run the job with the job ID, 208.

Python 2

```
# example.py
```

```
import requests
import json

HOST = "http://cdsw.example.com"
USERNAME = "alice"
API_KEY = "uysgxtj7jzkps96njextnxxmq05usp0b"
PROJECT_NAME = "risk-analysis"
JOB_ID = "208"

url = "/".join([HOST, "api/v1/projects", USERNAME, PROJECT_NAME, "jobs",
JOB_ID, "start"])
job_params = {"START_DATE": "2017-01-01", "END_DATE": "2017-01-31"}
res = requests.post(
    url,
    headers = {"Content-Type": "application/json"},
    auth = (API_KEY,""),
    data = json.dumps({"environment": job_params})
)

print "URL", url
print "HTTP status code", res.status_code
print "Engine ID", res.json().get('engine_id')
```

When you run the code, you should see output of the form:

```
python example.py
```

```
URL http://cdsw.example.com/api/v1/projects/alice/risk-analysis/jobs/208/sta
rt
HTTP status code 200
Engine ID r11w5q3q589ryg9o
```

Limitations

- Cloudera AI does not support changing your legacy API key, or having multiple API keys.
- Currently, you cannot create a job, stop a job, or get the status of a job using the Jobs API.

Related Information

API v2 usage

Basic Access Authentication

Creating a Pipeline

Environment Variables

Clouera AI API v2

Cloudera AI

Jupyter Magic Commands

Date published: 2020-07-16
Date modified: 2025-01-29

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Jupyter Magic Commands.....4
Python.....4
Scala.....4

Jupyter Magic Commands

Cloudera AI's Scala and Python kernels are based on Jupyter kernels. Jupyter kernels support varying magic commands that extend the core language with useful shortcuts. This section details the magic commands (magics) supported by Cloudera AI.

Line magics begin with a single %: for example, %timeit. Cell magics begin with a double %%: for example, %%bash.

Python

In the default Python engine, Cloudera AI supports most line magics, but no cell magics.

Cloudera AI supports the shell magic !: for example, !ls -alh /home/cdsw.

Cloudera AI supports the help magics ? and ??: for example, ?numpy and ??numpy. ? displays the docstring for its argument. ?? attempts to print the source code. You can get help on magics using the ? prefix: for example, ?%timeit.

Cloudera AI supports the line magics listed at https://ipython.org/ipython-doc/3/interactive/magics.html#line-magics, with the following exceptions:

- %colors
- %debug
- %edit
- %gui
- %history
- %install_default_config
- %install_profiles
- %lsmagic
- %macro
- %matplotlib
- %notebook
- %page
- %pastebin
- %pdb
- %prun
- %pylab
- %recall
- %rerun
- %save
- %sc

Related reference

Scala

Scala

Cloudera AI's Scala kernel is based on Apache Toree. It supports the line magics documented in the Apache Toree magic tutorial.

Related reference

Python

Cloudera AI

Cloudera AI Discovery & Exploration

Date published: 2020-07-16
Date modified: 2025-01-29

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Exploratory Data Science and Visualization 5

Prerequisites for Cloudera AI discovery and exploration 5

Starting Data Discovery and Visualization 6

Working with Data Discovery and Visualization 7

Set up a Hive or Impala data connection manually 8

Setting up a Spark data connection 10

Setting up Amazon S3 data connection 11

Setting up a data connection to Cloudera Data Hub 11
- Set up a DataHub data connection using raw code 13

Data connection management 13

Setting the workload password 14

Setting up a Custom Data Connection 14
- Custom Data Connection Development 17
- Developing and testing your first custom connection 17
- Loading custom connections 22

Using data connection snippets 23

Managing default and backup data connections 25

API Permissions For Projects 25

Troubleshooting: 401 Unauthorized 26

Troubleshooting: 401 Unauthorized when accessing Hive..... 26

Troubleshooting: Existing connection name.....27

Troubleshooting: Empty data page..... 27

Troubleshooting: Some connections not shown.....27

Exploratory Data Science and Visualization

Exploratory Data Science and Visualization makes it simple for data scientists to get started on a data science project.

When you start a data science project, you are presented with a blank notebook, and no indication of what data sources are available on the Cloudera Data Platform platform, or how to access them. The Exploratory Data Science and Visualization feature automatically discovers the data sources available to you, from within the standard Cloudera AI user interface.

The Exploration Data Science and Visualization experience enables all features for Exploratory Data Analysis. From the Data tab you can:

- Connect to data sources that are available in your project
- Explore the data with SQL to understand its basic shape and characteristics
- Create named datasets that you can reference later
- Quickly create visualizations of the data to understand its properties
- Create dashboards that you can share with your team

For more information on what you can do in the Data tab, see the documentation for *Cloudera Data Visualization*.

Related Information

Cloudera Data Visualization

Prerequisites for Cloudera AI discovery and exploration

Cloudera AI Discovery and Exploration has a few prerequisites.

- For both Hive and Impala connections, the HADOOP_USERNAME and WORKLOAD_PASSWORD must be set. The HADOOP_USERNAME is set automatically by the environment. As a data scientist, you must set the WORKLOAD_PASSWORD. For more information, see *Setting the workload password*.
- To manually create Impala or Hive data connections, you must have the related JDBC URL. You can obtain this from the option menu for the virtual warehouse to which you want to connect. These warehouses must be already created in the environment. For more information on how to create a data warehouse, see *Adding a New Virtual Warehouse*.
- To manually create a Spark connection, you need the Data Lake S3 bucket. This can be retrieved from the Data Lake page. For Spark data connections, you must use ML Runtimes, and specifically the Spark Runtime Add-on must be enabled before starting a workload (job or session).
- For Spark data connections, you must have permissions set correctly for external access to the S3 bucket.
- Hive and Impala data connections also require ML Runtimes. Legacy engines may work, but are not supported.
- There must be auto-discovered data connections, or your Administrator must create data connections to virtual warehouses or other data sources in your Cloudera Data Platform environment.

Known Issue

- Only Hive virtual warehouses with SSO disabled are supported.

Related Information

Create a Dashboard

Adding a New Virtual Warehouse

Setting the workload password

Starting Data Discovery and Visualization

You can start the Exploratory Data Science and Visualization application within Cloudera AI.

1. Create or open a project.
2. In the project, click Data in the left navigation pane.
3. If the application was previously started, it appears in the UI.
4. If not, click Launch Data Application to start the application. It takes a few minutes to start the first time.

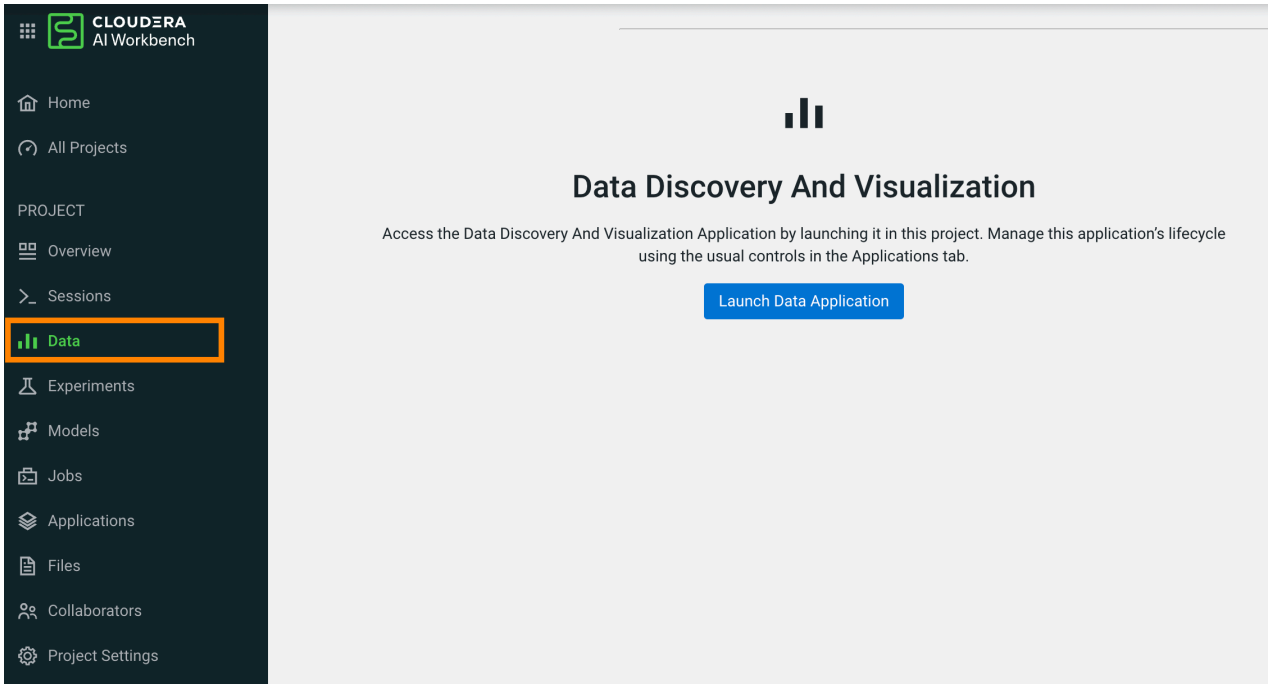

When the application starts, you can see four tabs at the top of the UI. From here, you can create new dashboard or dataset or try some of the tasks described in the next topics.

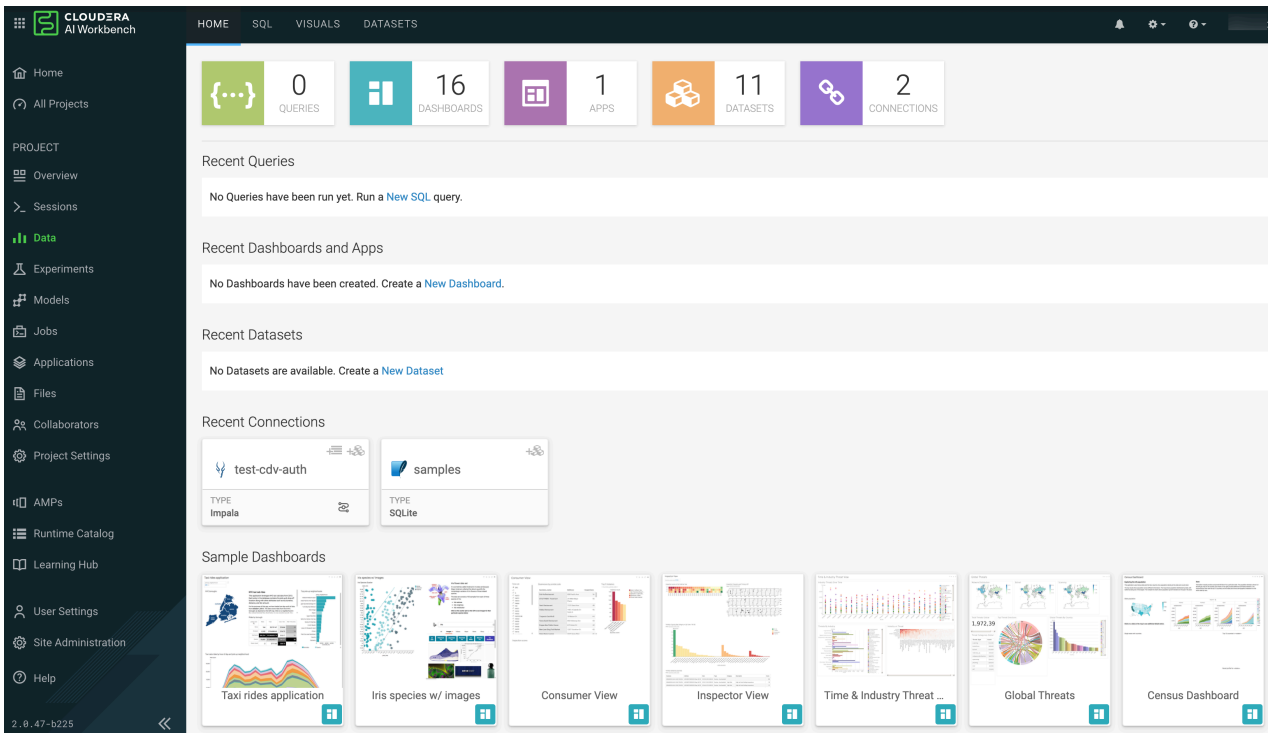

Related Information

Creating datasets

Create a Dashboard

Troubleshooting: 401 Unauthorized

Working with Data Discovery and Visualization

This section provides examples of the tasks you can perform using Data Discovery and Visualization.

Run an SQL query

On the SQL tab, select a Data Connection from the dropdown box.

Note: If you do not see the data connections that you have configured in the Project, refer to the Troubleshooting section for more information.

Once you select a Data Connection, you can explore the available databases and tables. You can start writing an SQL query and click the table names to insert them into the query string. For more information, see *Creating a dataset*.

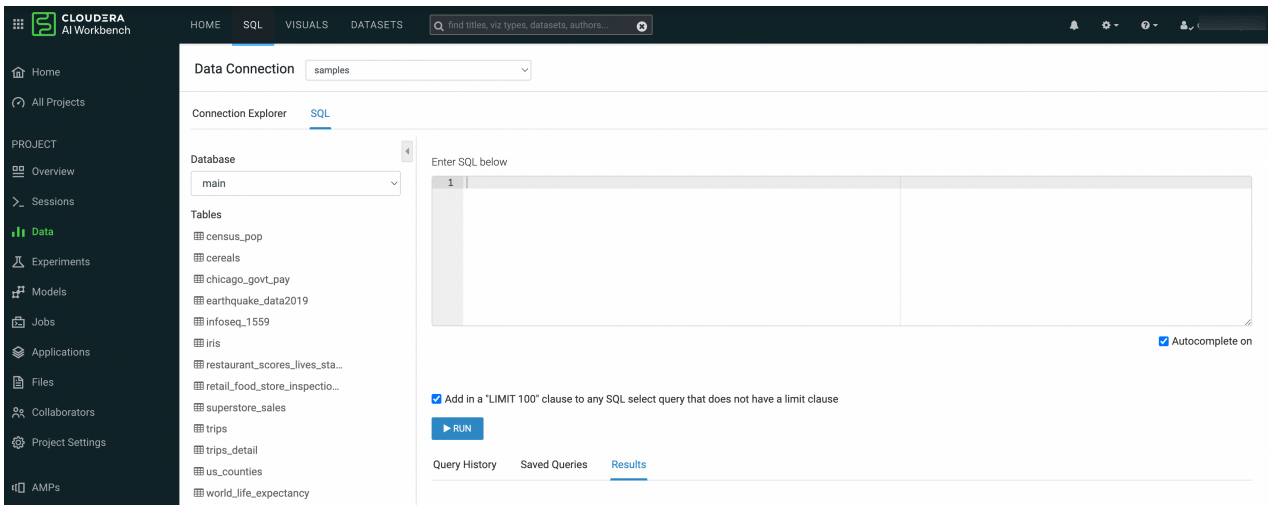

Build a dashboard

Click New Dashboard to access the Dashboard Designer. You can select from a wide variety of graphs, click to add them to your dashboard, and select the columns from your dataset to use in the visual. For more information, see *Create a Dashboard*.

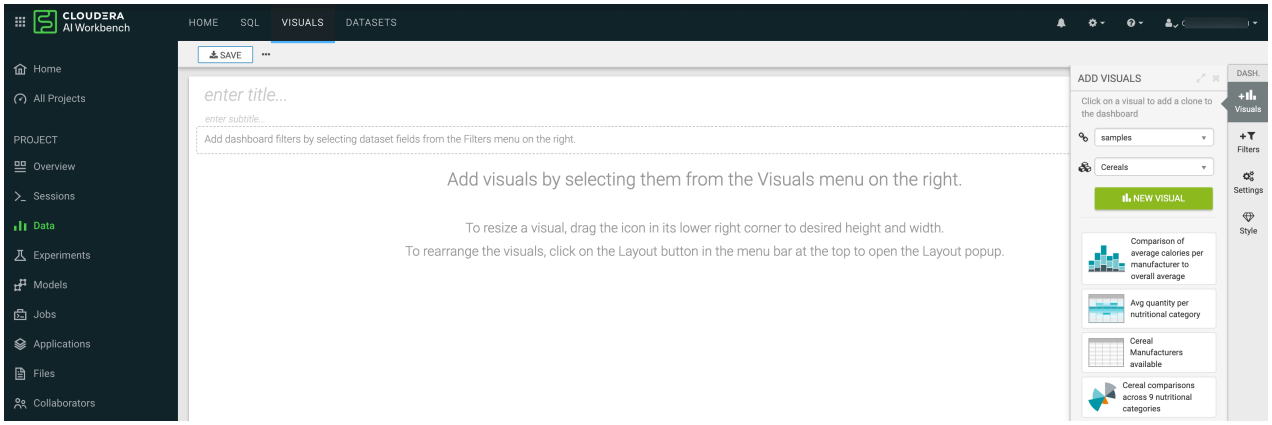

Related Information

Creating a dataset

Create a dashboard

Troubleshooting: 401 Unauthorized

Set up a Hive or Impala data connection manually

Data connections to Hive or Impala virtual warehouses within the same environment as the Cloudera AI Workbench are automatically discovered and configured. You can also set up a data connection manually, which works across Cloudera Data Platform environments. Follow this procedure to set up a Hive or Impala data connection.

Procedure

1. Log into the Cloudera Data Platform web interface and navigate to the Cloudera Data Warehouse service.
2. In the Cloudera Data Warehouse service, select Virtual Warehouses in the left navigation panel.

3. Select the options menu for the warehouse you want to access, and select Copy JDBC URL.

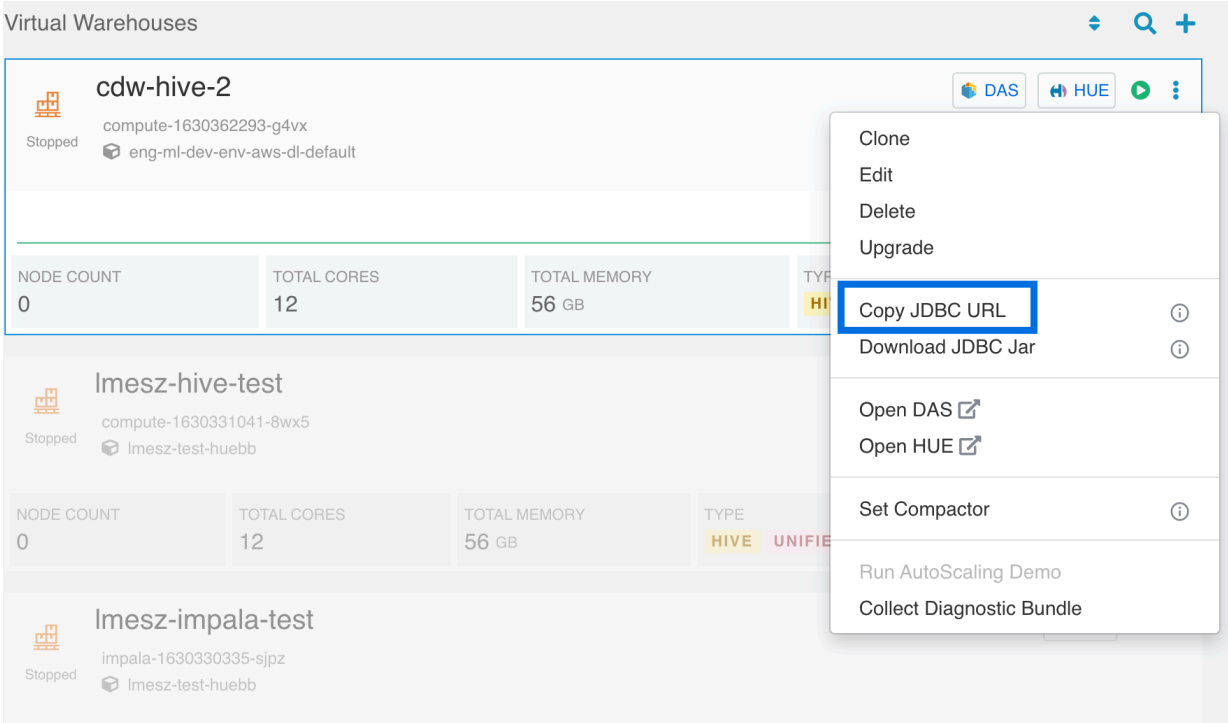

4. Return to the Cloudera AI service. In Site Administration Data Connections , select New Connection.
5. Enter the connection name. You cannot have duplicate names for data connections within a workbench or within a given project.
6. Select the connection type:
   a. Hive Virtual Warehouse
   b. Impala Virtual Warehouse
7. Paste the JDBC URL for the data connection.

8. (Optional) Enter the Virtual Warehouse Name. This is the name of the warehouse in Cloudera Data Warehouse.

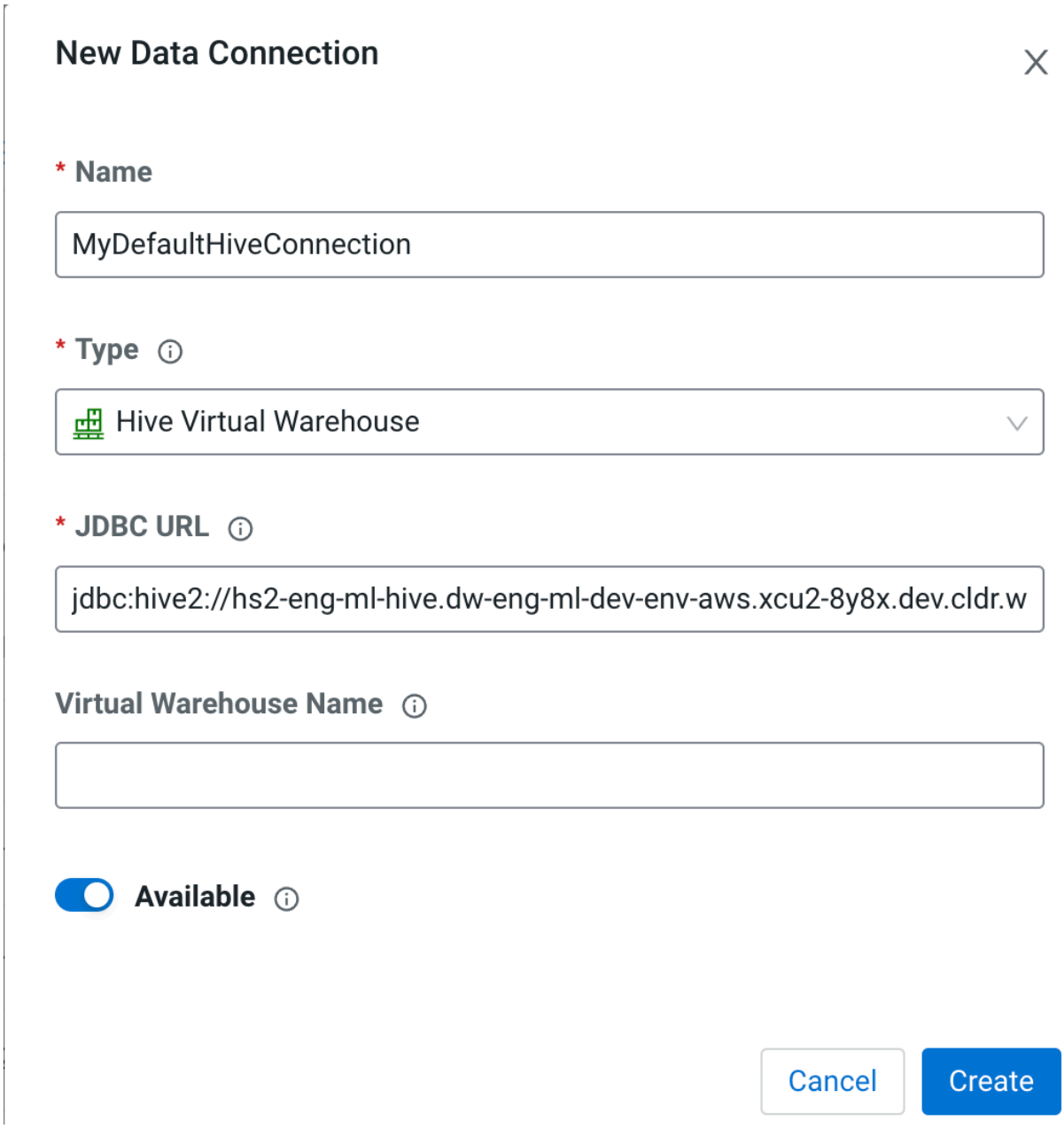

Results

The data connection is available to users by default. To change availability, click the Available switch. This switch determines if the data connection is displayed in Projects created within the workbench.

Setting up a Spark data connection

Spark data connections within the same environment as Cloudera AI are automatically discovered, but you can also set up a connection manually. Follow this procedure to set up a Spark data connection.

Procedure

1. In the Workbenches UI, select the link environment for the workbench you are using. This takes you to the Environments UI.

2. In Environments, select Data Lake Cloud Storage tabs.
3. Select the directory path shown for Hive Metastore External Warehouse, and copy it.
4. In Project Settings > Data Connections, click New Connection.
5. Enter a name for the connection.
6. Select the type: Spark Data Lake
7. Paste the value you copied in step 3 into Datalake Hive Metastore External Warehouse Directory.
8. Click Create.

Results

The data connection is available to users by default. To change availability, click the Available toggle.

Setting up Amazon S3 data connection

Amazon S3 object store connection is automatically created for Cloudera AI Workbenches to make it easier to connect to the data stored within the same environment. Other Data Connections can be configured to other S3 locations manually. You can also set up a connection manually.

About this task

Amazon S3 data connections are available only on AWS workbenches where RAZ (Ranger Authorization Service) is enabled to authorize connections to the environment's S3 buckets.

Procedure

1. In the **Cloudera** console, click the **Cloudera AI** tile.

   The **Home** page displays.
2. Click **Site Administration** in the left navigation menu.

   The **Site Administration** page displays.
3. In the Data Connections page, click New Connection.

   The **New Data Connection** window is displayed.
4. In the Name field, enter a name for the connection.
5. In the Type drop-down list, select the type as S3 Object Store.
6. Click Create.

   This data connection only supports per-bucket operations. For information on using the S3 data connection, and connection wrapper for the S3 boto client, see *Amazon Boto documentation*.

   The data connection is available to users by default. To change availability, click the Available toggle. This switch determines if the data connection is displayed in Projects created within the workbench.

Related Information

Amazon Boto documentation

Setting up a data connection to Cloudera Data Hub

You can set up a data connection to the DataHub cluster. You can set up a connection using the New Connection dialog, or by using raw code inside your project. Both approaches are shown below.

Procedure

1. Log into the console web UI.

2. Depending on which connection you want to use, click on either Connect to Hive or Connect to Impala tile.
3. Ensure your Data Hub cluster name is correct in the popup.
4. Copy the JDBC URL string.
5. Now click on the Build a Data Science Project tile to log into the the Cloudera AI Workbench.
6. In Site Administration Data Connections , select New Connection.
7. Return to the Cloudera AI service. In Site Administration Data Connections , select New Connection.
8. Enter the connection name. You cannot have duplicate names for data connections within a workbench or within a given project.
9. Select the connection type:
   a. Hive Virtual Warehouse
   b. Impala Virtual Warehouse
10. Paste the JDBC URL for the data connection.
11. (Optional) Enter the Virtual Warehouse Name. This is the name of the warehouse in Cloudera Data Warehouse.

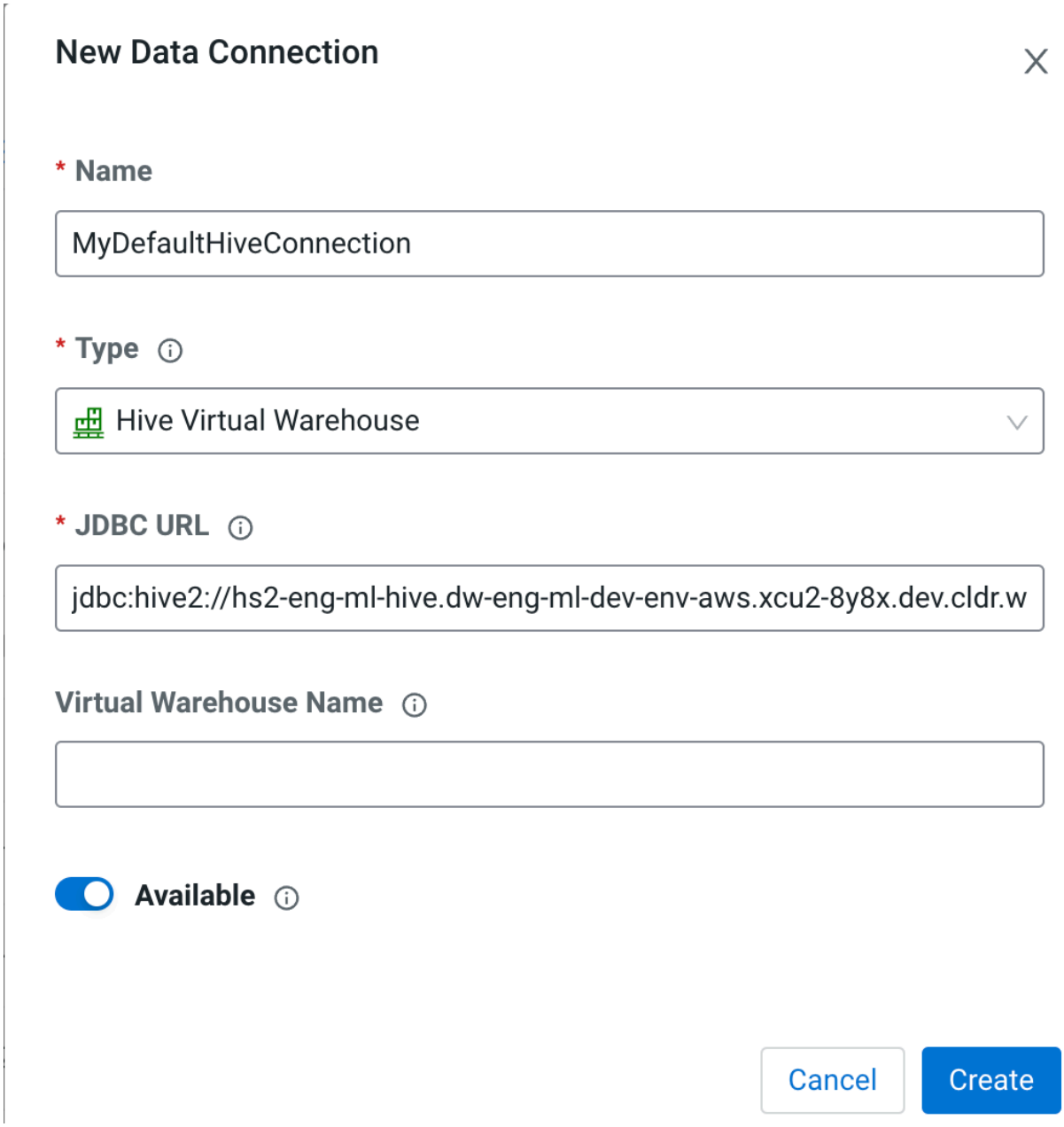

Results

The data connection is available to users by default. To change availability, click the Available switch. This switch determines if the data connection is displayed in Projects created within the workbench.

Set up a DataHub data connection using raw code

It is recommended to use the New Connection dialog to create a new data connection. If needed, you can also set up a data connection in your project code by using and adapting the following code snippet.

Example

```
from impala.dbapi import connect

#Example connection string:
# jdbc:hive2://my-test-master0.eng-ml-i.svbr-nqvp.int.cldr.work/;ssl=true;
transportMode=http;httpPath=my-test/cdp-proxy-api/hive

conn = connect(
    host = "my-test-master0.eng-ml-i.svbr-nqvp.int.cldr.work",
    port = 443,
    auth_mechanism = "LDAP",
    use_ssl = True,
    use_http_transport = True,
    http_path = "my-test/cdp-proxy-api/hive",
    user = "csso_me",
    password = "Test@123")
cursor = conn.cursor()
cursor.execute("select * from 3yearpop")

for row in cursor:
    print(row)
cursor.close()
conn.close()
```

Data connection management

There are a few things to keep in mind about data connections.

- Manage data connections in a workbench

  At the workbench level, you can check the data connections that are available in a workbench. In Project Settings Data Connections , check that your desired data source is present. You can also set the availability for any discovered connections, if necessary.
- Manage data connections in a project

  All the data connections that were available in the workbench when the project was created will be automatically created in the project as well. The available connections can then further be marked as unavailable if so desired. You can update any changes to the connections that were made at the workbench level by clicking Sync with workbench . Any changes made here only apply to your project.

- Data connection availability

  Keep in mind these two scenarios for setting data connection availability.

  1. If a workbench data connection is marked Unavailable, and you then create a project, the data connection will not appear in the project. If the connection is then changed to Available, and then the Sync with workbench button is clicked, the connection will appear in the project.
  2. If a workbench data connection is marked Available, and you then create a project, the connection shows up. If the workbench data connection is then toggled to Unavailable, and you click Sync with workbench in the project, the data connection will remain available in the project.

Setting the workload password

Set the workload password to connect your code to the data.

You need to set an environmental variable for the password (WORKLOAD_PASSWORD). This environmental variable is needed to access a Hive or Impala warehouse. The password you set in Cloudera AI must match the password you set in Management Console. For more information on setting the workload password in Management Console, see *Setting the workload password*.

To set the password:

1. In Cloudera AI, go to User Settings Environment .
2. Under the Reserved Environment Variables set your WORKLOAD_PASSWORD.
3. Click Save.
4. In Management Console, select your user name in the left navigation pane, and select Profile.
5. Click Set Workload Password.
6. Enter the same password as in step 2, and click Set Workload Password.

Note: You need to restart any sessions or applications that are running in your workbench.

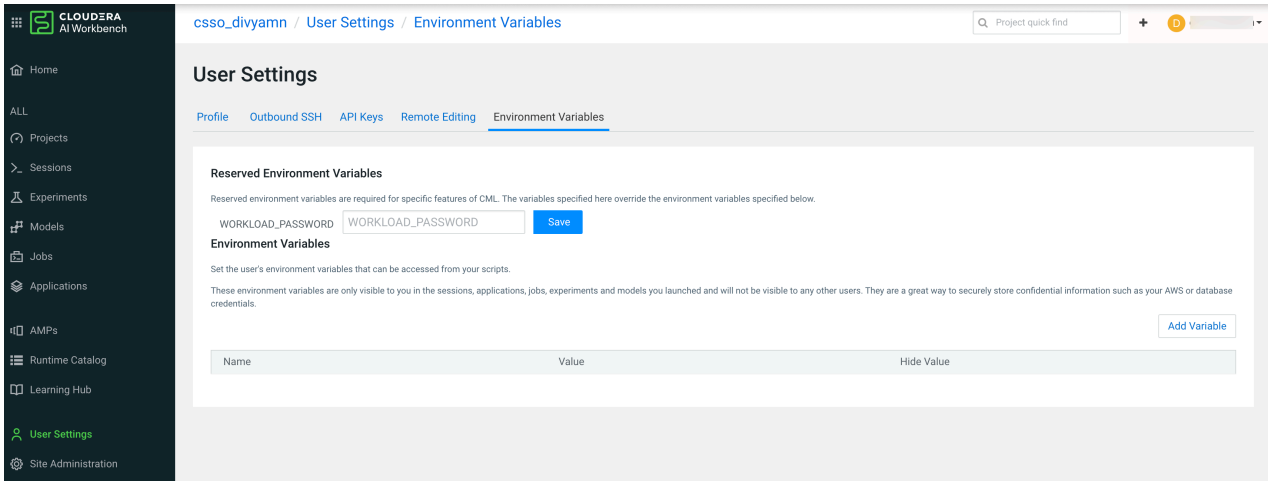

Related Information

Setting the workload password

Setting up a Custom Data Connection

Data connections that point to data sources outside of Cloudera Data Platform or require custom configurations can be created and made available to end users with Custom Data Connections. These Python implementations of the

Cloudera AI Data library are stored in the Data Connections Registry. Workbench users can track and connect to any data source and connection implementation a Cloudera AI Administrator makes available.

About this task

Consider the followings:

- Custom connections can only be created in projects created by the Administrator.
- The project source selection list in the Data Connection creation dialogue only displays projects created by the user.
- Team projects or projects with multiple collaborators will also not be displayed, only those directly created by the user.
- Custom connections at workbench level can only be edited by the creator, not other Administrator users. Attempts at editing workbench level custom connections will result in an error.

Before you begin

Before setting up a custom connection, you might want to create a dedicated Cloudera AI Team to collaborate on external connections. A good practice is to separate the connection code projects and and configure collaborators on the Team level to build and maintain the connection code.

Procedure

1. Develop your own custom data connection (see *Developing a Custom Data Connection*) in a Cloudera AI project, or clone an existing custom data connection files directory into a Cloudera AI project.
2. In Site Administration Data Connections , select New Connection.
3. Enter the connection name. You cannot have duplicate names for data connections within a workbench or within a given project.
4. Select the connection type: Custom Connection
5. Enter the Type Display name. This should be a descriptive label to help Cloudera AI project owners identify what this custom connection could be used for.
6. Select the Cloudera AI Project and Project directory which contains your custom connection implementation
   a. Connection files must be in a directory and not in the root of your project.
   b. A snapshot of all implementation files in the directory will be uploaded to the Cloudera AI Custom Data Connection registry located in the workbench.
   c. These uploaded files are safe from any changes to the originating project. To make changes to the files, create a new custom data connection.

7. (Optional) Enter any custom parameters. These are available during a session and can be validated or overridden depending on the interface implementation for the custom data connection. Refer to the implementation of your custom data connection for specific details on required keys and values.

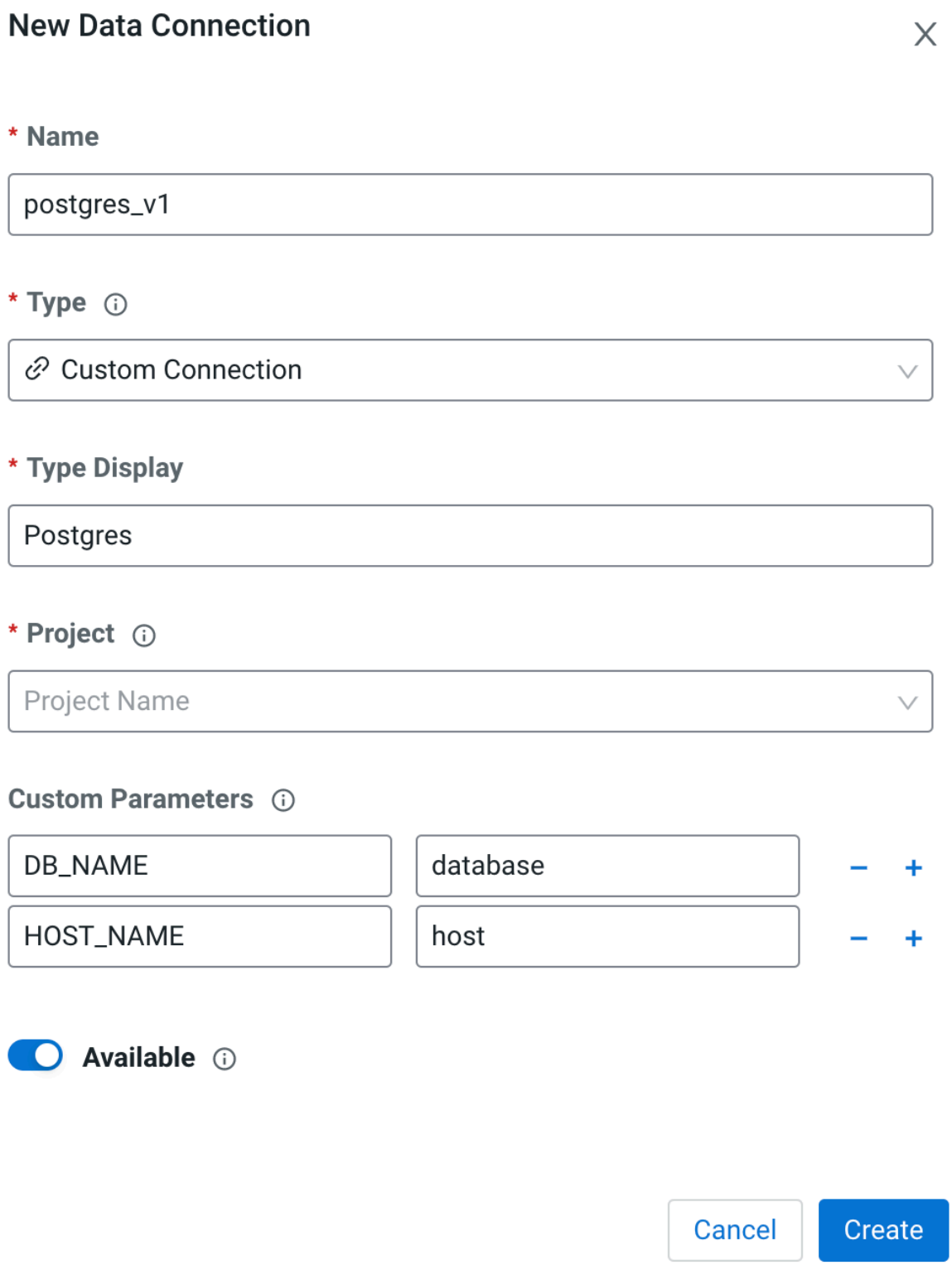

8. Click Create.

Results

The data connection is now available to all users. To change availability, click the Available switch. This switch determines if the data connection is displayed in Projects created within the workbench. Refer to *Data connection management* for availability of your newly created custom connections in new and existing Cloudera AI Projects.

Related Information

Data connection management

Custom Data Connection Development

Custom data connections can be developed from within Cloudera AI Workbench and Python Sessions using the Cloudera AI Python Data Library and implementing the Cloudera AI Custom Connection Interface.

You can view CustomConnection interface help descriptions within in a session:

```
import cml.data_v1 as cmldata
help(cmldata.customconnection)
```

Alternatively, you can inspect the source content as follows:

```
import cml.data_v1 as cmldata
import inspect
print(inspect.getsource(cmldata.customconnection))
```

Your custom connection code must implement the CustomConnection interface for the cml.data_v1 library to load your module dynamically (see *Loading custom connections*)

Two functions are already implemented so that the Cloudera AI Python Data Library can dynamically load your Python module implementation and make custom parameters available in self.parameters. In most cases, you will not need to reimplement these:

1. __init__(self, properties)
2. update_properties(self, properties)

The rest of the interface functions are included as common functions that you may want to implement.

1. get_base_connection(self)
2. get_pandas_dataframe(self, query)
3. get_cursor(self)
4. print_usage(self)
5. override_parameters(self)

See *Developing and testing your first custom connection* for a simple example of how to implement these.

Related Information

Loading custom connections

Developing and testing your first custom connection

Developing and testing your first custom connection

You can develop a custom connection in your own project, as shown in this example.

Procedure

1. Create a new directory hello-world-conn-dev and file hello-conn.py to contain your custom data connection files.

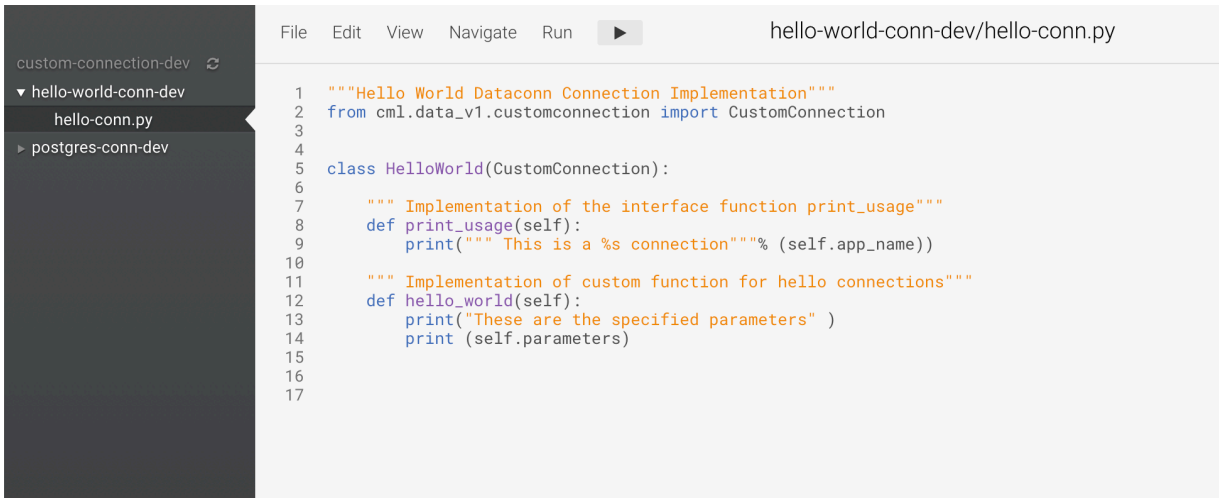

2. Implement your custom data connection in hello-conn.py. (There must only be one class which implements `CustomConnection` in this directory.)

```
"""Hello World Custom Connection Implementation"""
from cml.data_v1.customconnection import CustomConnection

class HelloWorld(CustomConnection):

    """ Implementation of the interface function print_usage"""
    def print_usage(self):
        print(""" This is a %s connection"""% (self.app_name))

    """ Implementation of custom function for hello connections"""
    def hello_world(self):
        print("These are the specified parameters" )
        print (self.parameters)
```

3. Test your custom data connection locally (for more information see *Loading custom connections*)

```
test_params = {"PARAM_1": "foo", "PARAM_2": "bar"}
import cml.data_v1 as cmldata

conn = cmldata.get_custom_connection_from_local(
  "hello-world-conn-dev",
  "my-hello-connection",
  test_params)
conn.print_usage()
```

```
conn.hello_world()
```

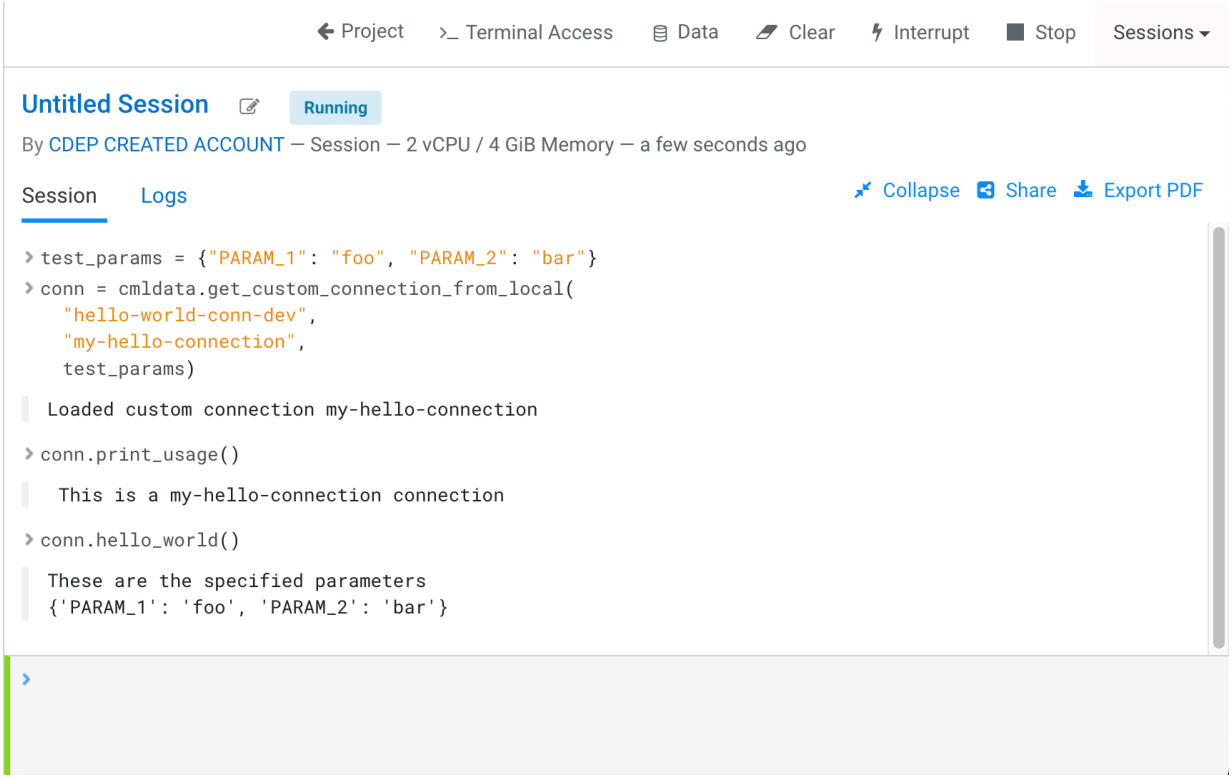

4. In Site Administration Data Connections , select New Connection, and fill in the fields as shown below.

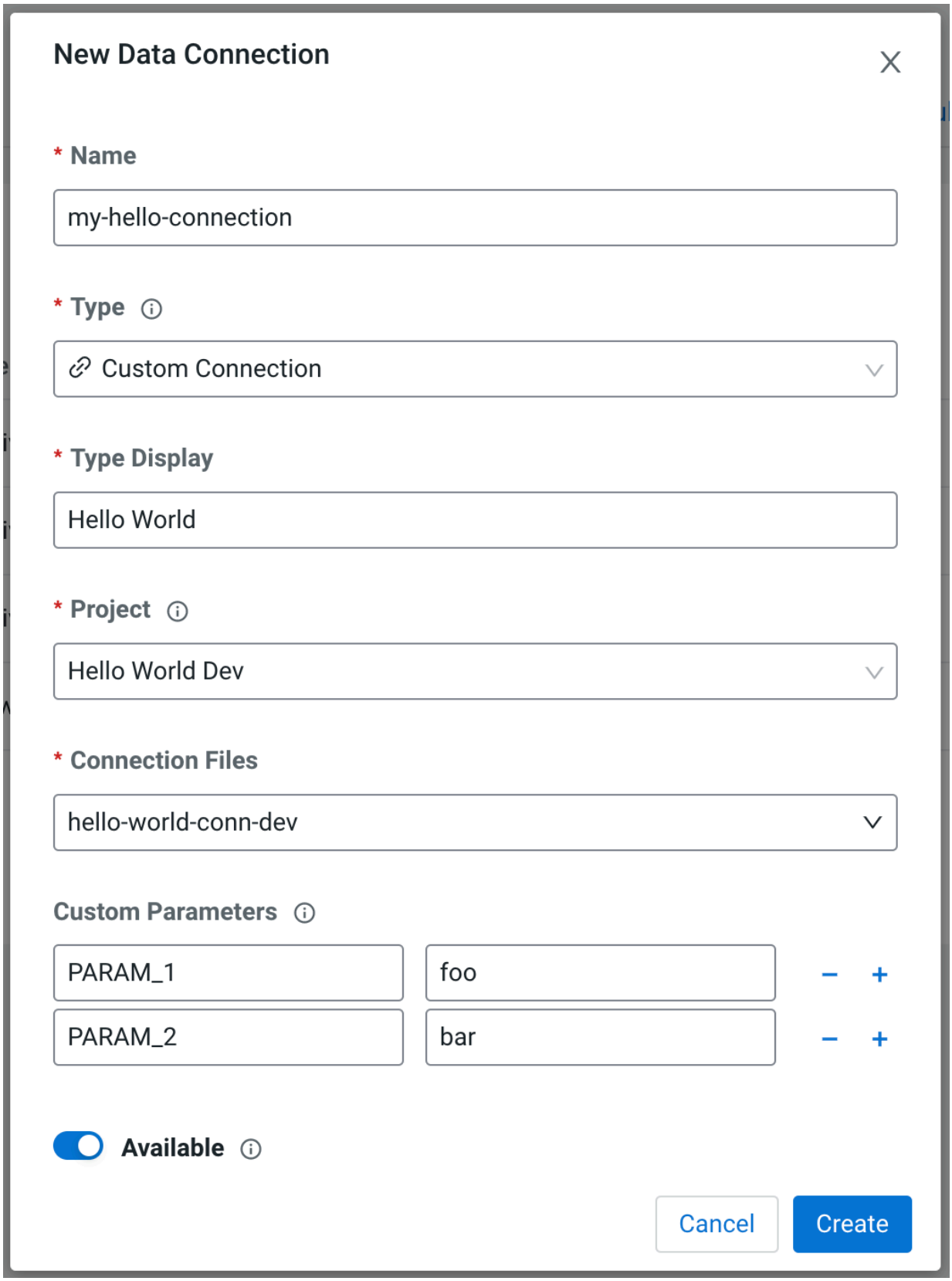

5. See *Data connection management* to set the availability of your newly created custom connection in new and existing Cloudera AI projects.

6. Create a project, and start a session to use your completed Hello World connection.

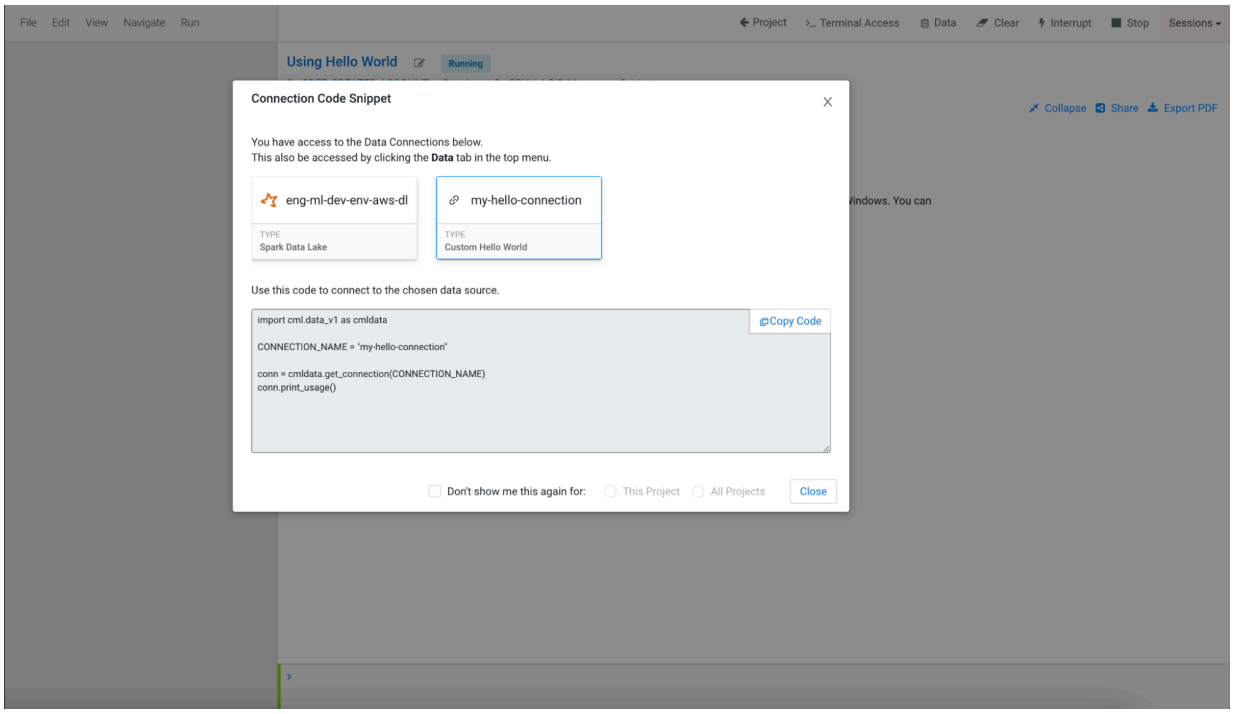

Note: A default custom connection code snippet is automatically displayed to the user.

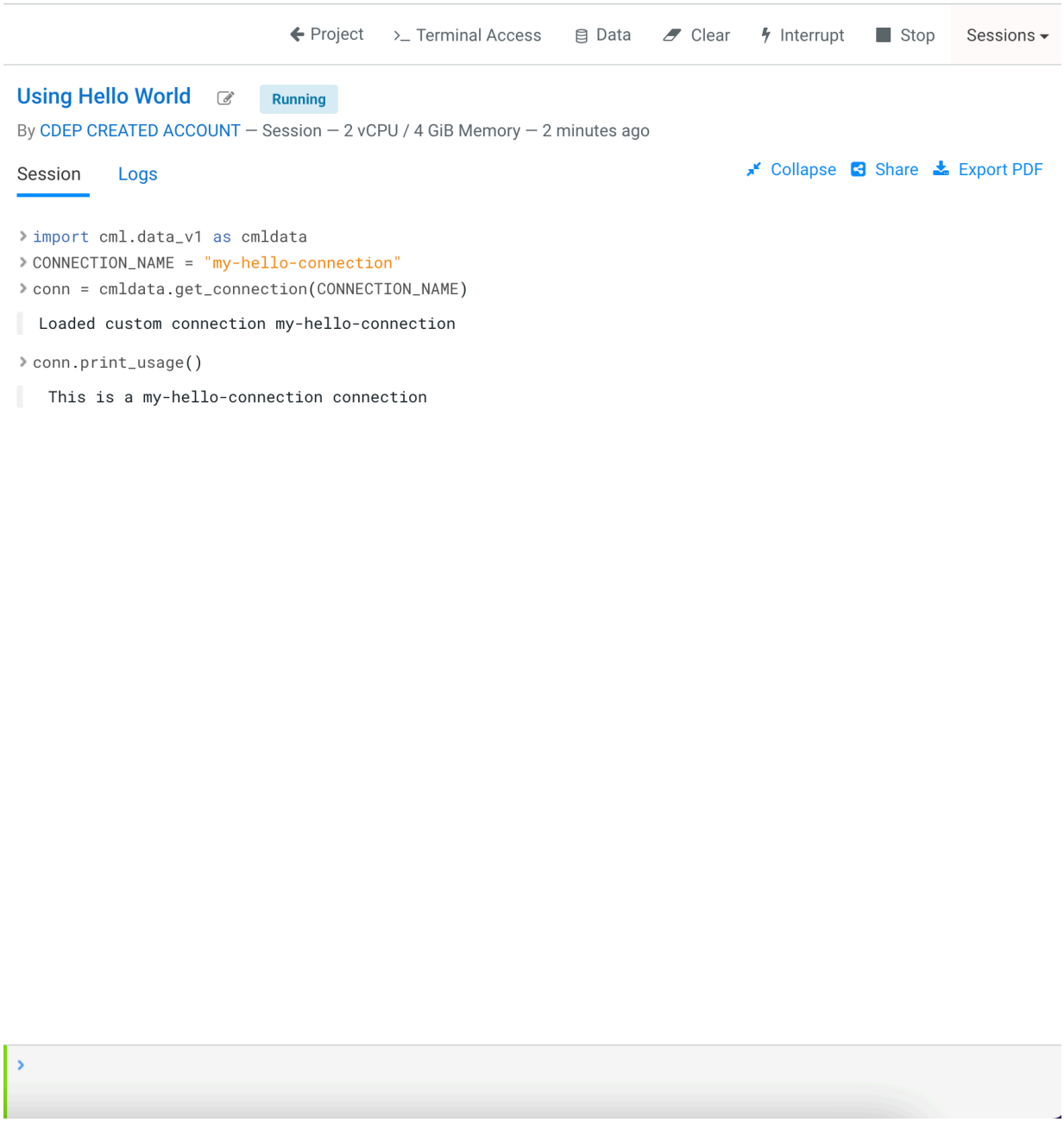

Note: When exporting custom connection files, git is recommended. Alternatively, download the custom connection directory only, instead of all project files.

Related Information

Loading custom connections

Data connection management

Using data connection snippets

Loading custom connections

You can instantiate a local connection for testing, using the name of your custom connection directory, a sample connection name, and an optional dictionary of parameters. This local connection object can then be used to test and implement functions in your custom connection.

The following code sample loads custom connection package directories in the same way that the Cloudera AI Data Library imports a registered custom connection when called with get_connection:

```
get_custom_connection_from_local(package_name, connection_name, parameters={
})
```

Returns a Cloudera AI Custom Data Connection object. For testing in-development Data Connection code.

Parameters:

- package_name (str): The accessible package name containing custom connection code to load.
- connection_name (str): The connection name to be used in Custom Connection loading.
- parameters (dict of str: str): Mapping of custom parameter keys and values that will be loaded by the custom connection code.

Return:

A custom connection object that implements cml.data_v1.customconnection.CustomConnection

Usage: conn = load_custom_connection_source("myconndir", {"HOSTNAME": "my.instance.host.com"})

Note: When you make changes to your custom connection file, get_custom_connection_from_local will dynamically re-import the contents, so the latest code on disk is instantiated for the local connection.

To load any created data connection that is available in the Cloudera AI project, use the get_connection instruction.

```
get_connection(dataconnection_name, parameters=None)
```

Usage: conn = get_connection(connection_name)

If the specified connection is of type “Custom” a snapshot of the Custom Data Connection files specified at the time of Connection creation is imported.

Using data connection snippets

As a data scientist, you can connect your project to data with a data connection snippet.

Procedure

1. Select New Project.
2. Enter the project name.
3. Start a new session. To use a Spark connection, make sure to select an ML Runtimes engine.

4. In the Connection Code Snippet pane, select the data connection to use.

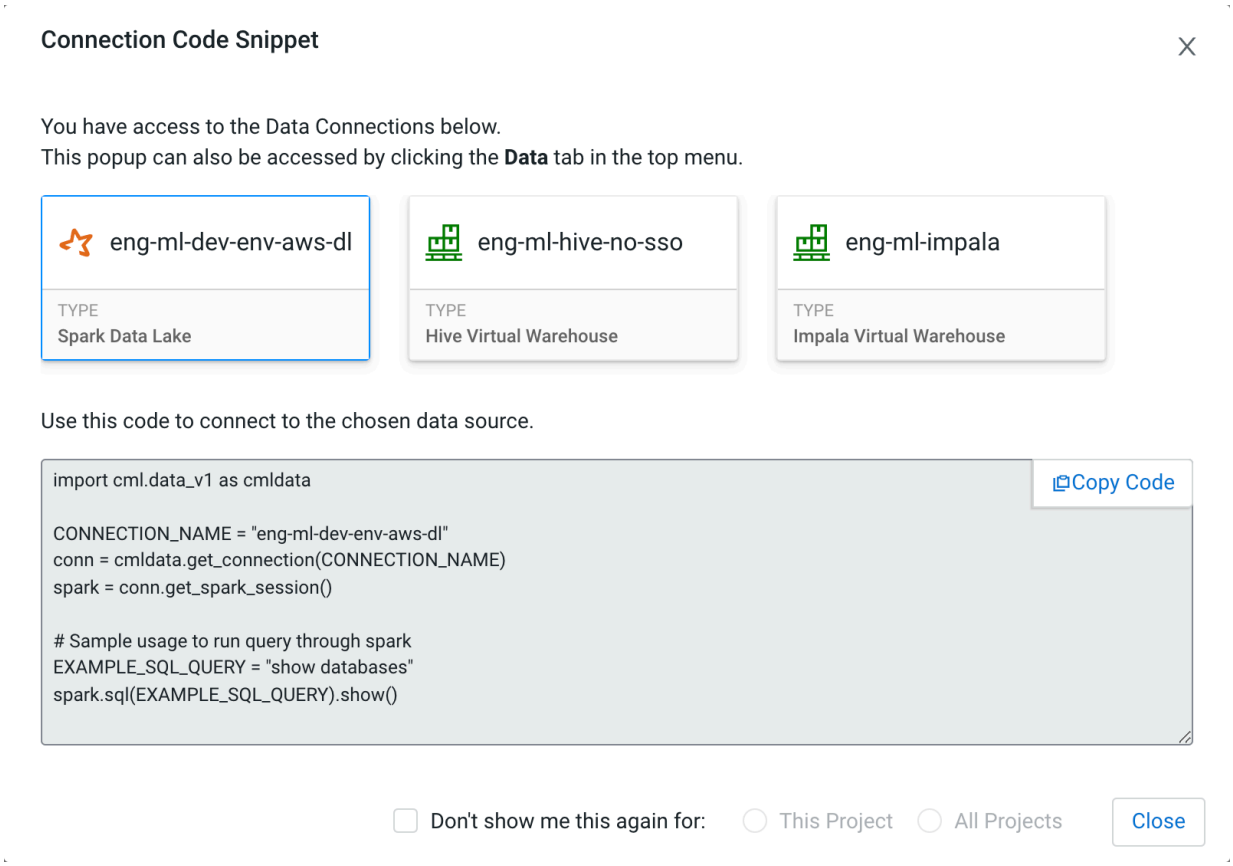

5. Select Copy Code.
6. Paste the snippet into your code.

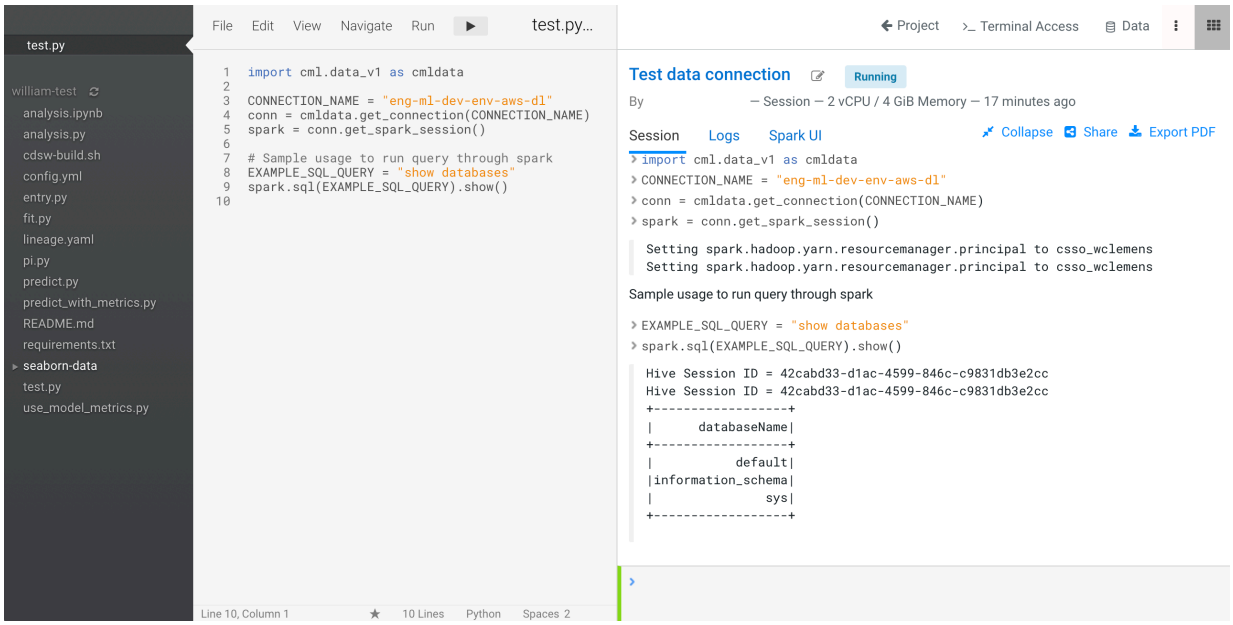

7. Edit the username and password, if necessary. (This is not necessary for the Spark Data Lake connection, or if the WORKLOAD_PASSWORD environmental variable is set up for the user.)
8. Enter the data connection name.
9. Uncomment and edit the SQL query.

10. Select Run.

Results

The results of the SQL query display in the command window.

What to do next

When you have finished exploring the available data sources, you can select **Don't show me this again** for this project on the Connection Code Snippet pane, and it will no longer display when you start a session in the project.

Managing default and backup data connections

As a data scientist, you may want to set default and backup data connections (virtual warehouses). This makes it easy to manage the case where the default data source (virtual warehouse or data lake) becomes unavailable, for example.

Before you begin

You need Admin privileges to perform this task. Here, the data connection names of default and backup are used as examples.

Procedure

1. In Site Administration Data connections , click Edit to change the name of the default data connection to default.
2. Change another connection to backup.
3. In Project Settings, click Sync with workbench to update the names of the data connections. If two different data connections have the same name, an error occurs during synchronization.
4. If the default data connection becomes unavailable, the workbench Administrator can go to the Data Connections tab, and rename the connections.

   For example, after changing default to unavailable, change backup to default. Projects that use library code for their connection to the default data connection will continue to operate, because it is now using the new default connection. Note: you need to click Sync with workbench at the project level to get the updated connections.

API Permissions For Projects

The tasks available to collaborators depends on their level of access.

A collaborator with read-only access is able to do the following:

- List data connections
- Get data connection details

Note: The list of data connections is unavailable in the UI to viewer-only users because the Setting tab in a project is hidden for viewer collaborators. However, they can use curl to submit api requests to list the data connections.

A collaborator with write access can:

- Create data connections
- Edit data connections
- Delete data connections
- List data connections
- Get data connection details

Troubleshooting: 401 Unauthorized

Problem: Session returns a 401: Unauthorized error.

When you are in a session and try to run the code for a Hive or Impala connection, the session returns a 401: Unauthorized HTTP error.

```
Modify next 2 lines to update your credentials

> USERNAME = os.getenv('HADOOP_USER_NAME')
> PASSWORD = os.getenv('WORKLOAD_PASSWORD')
> conn = cmldata.getConnection({
            'CONNECTION_NAME': CONNECTION_NAME,
            'USERNAME': USERNAME,
            'PASSWORD': PASSWORD
        })
HttpError: HTTP code 401: Unauthorized
HttpError                                 Traceback (most recent call last)
<ipython-input-1-a69c5e2af83b> in <module>
      2          'CONNECTION_NAME': CONNECTION_NAME,
      3          'USERNAME': USERNAME,
----> 4          'PASSWORD': PASSWORD
      5        })

~/.local/lib/python3.7/site-packages/cmldata.py in getConnection(properties)
    105     fmt_codesnippet = _getConnectionSnippet(properties)
    106     scope = {}
--> 107     exec(fmt_codesnippet, scope)
    108     return scope["conn"]

<string> in <module>

<string> in getCursor(self)

/usr/local/lib/python3.7/site-packages/impala/hiveserver2.py in cursor(self, user, configuration, convert_type
    127         log.debug('_cursor(): getting new session handle')
```

Solution: You need to set your workload password. See *Setting the workload password* for details. After setting the workload password, start a new session. If this error occurs in the Data tab, then restart the Data Discovery and Exploration app.

Related Information

Setting the workload password

Troubleshooting: 401 Unauthorized when accessing Hive

Problem: Session returns a 401: Unauthorized error when you are accessing a Hive database.

Check that the Hive or Impala Virtual Warehouse is working.

Solution: Follow these steps to check that Hive or Impala data warehouse is running, and restart if needed.

1. In the control plane, go to Cloudera Data Warehouse.
2. Click Virtual Warehouses, and select the data warehouse for the connection.
3. Check the status of that data warehouse, and make sure that it is running or in a good state.

Troubleshooting: Existing connection name

Problem: When the user attempts to sync data connections, an error message displays, stating the crn or name is a duplicate.

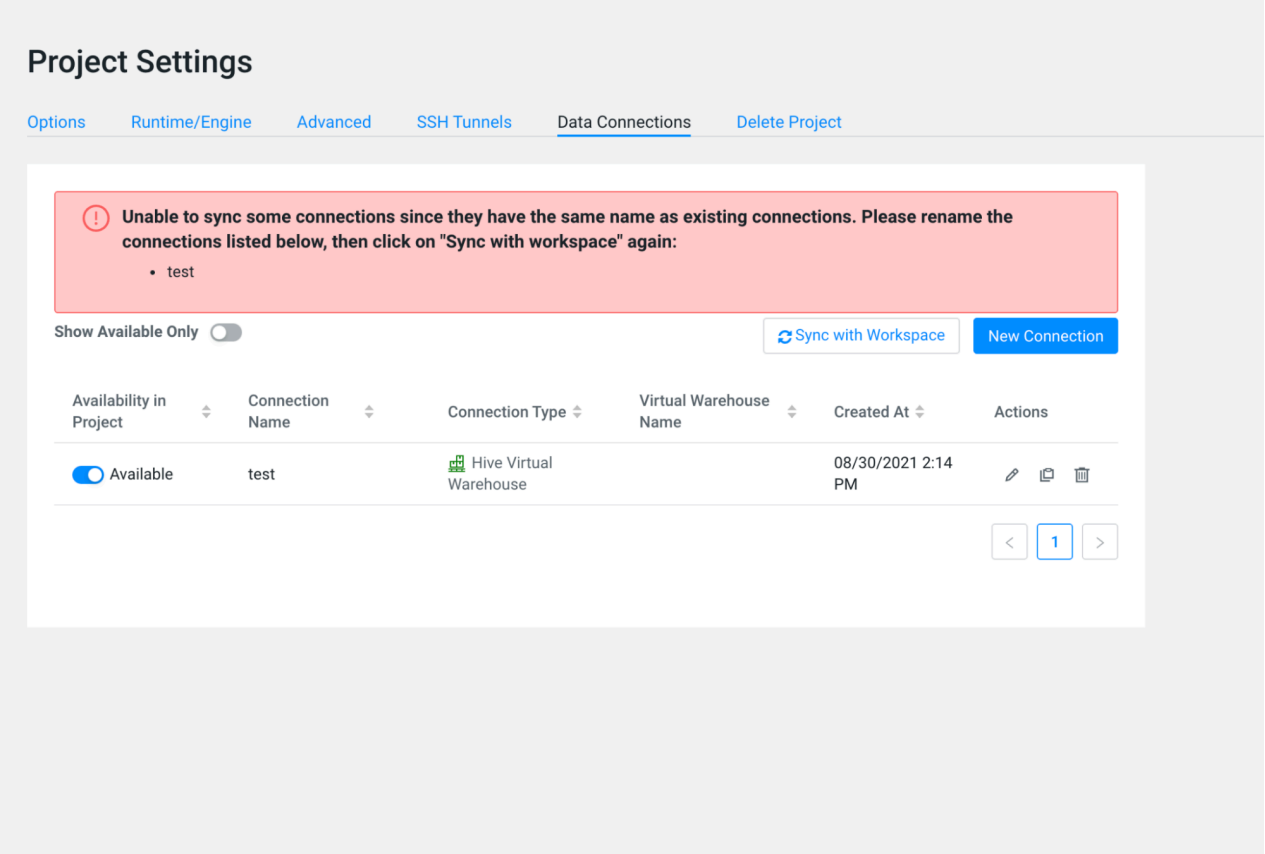

Solution: This indicates a project connection (one that is not copied from the workbench) has the same name or crn as a workbench connection. To resolve this, you need to change the name or crn of the data connection at the project level.

Troubleshooting: Empty data page

You launch the application and see the spinner, but then the empty data page returns.

On the applications tab, click Data Discovery and Visualization application, and check the logs. Check if another user stopped or restarted the application.

On the applications tab, click on Settings Resource Profile , and check if the available resources are sufficient. You can increase the resources if necessary, then restart the application.

Troubleshooting: Some connections not shown

The data application doesn't show all data connections.

See the error message to fix the workload password. To set the workload password, follow the procedure in *Setting the workload password*. You will need to restart the data application after setting the workload password.

Alternatively, you may need to click Sync under the Data Connections to get all of the data connections available at the project level.

Related Information

Setting the workload password

Cloudera AI

Model Governance

Date published: 2020-07-16
Date modified: 2024-11-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enabling model governance.....4

Registering training data lineage using a linking file..... 4

Viewing lineage for a model deployment in Atlas.....5

Enabling model governance

You must enable governance to capture and view information about your ML projects, models, and builds centrally from Apache Atlas (Data Catalog) for a given environment. If you do not select this option while provisioning Cloudera AI Workbenches, then integration with Atlas will not work.

About this task

Procedure

1. Go to Cloudera AI and click Provision Workbench on the top-right corner.
2. Enter the Cloudera AI Workbench name and other details.
3. Click Advanced Options.
4. Select Enable Governance.

Registering training data lineage using a linking file

The Cloudera AI projects, model builds, model deployments, and associated metadata are tracked in Apache Atlas, which is available in the environment's SDX cluster. You can also specify additional metadata to be tracked for a given model build. For example, you can specify metadata that links training data to a project through a special file called the linking file (lineage.yaml).

The lineage.yaml file describes additional metadata and the lineage relationships between the project's models and training data. You can use a single lineage.yaml file for all the models within the project.

Note: Your lineage file should be present in your project before you create a model build. The lineage file is parsed and metadata is attached during the model build process.

1. Create a YAML file in your Cloudera AI project called lineage.yaml.

   If you have used a template to create your project, a lineage.yaml file should already exist in your project.
2. Insert statements in the file that describe the relationships you want to track between a model and the training data. You can include additional descriptive metadata through key-value pairs in a metadata section.

YAML	YAML Structure	Description
Model name	Top-level entry	A Cloudera AI model name associated with the current project. There can be more than one model per linking file.
hive_table_qualified_names	Second-level entry	This pre-defined key introduces sequence items that list the names of Hive tables used as training data.
Table names	Sequence items	The qualified names of Hive tables used as training data enclosed in double quotation marks. Qualified names are of the format *DB-NAME.TABLE-NAME@CLUSTER-NAME*
metadata	Second-level entry	This pre-defined key introduces additional metadata to be included in the Atlas representation of the relationship between the model and the training data.
KEY:*VALUE*	Third-level entries	Key-value pairs that describe information about how this data is used in the model. For example, consider including the query text that is used to extract training data or the name of the training file used.

The following example linking file shows entries for two models in your project: modelName1 and modelName2:

```
modelName1:                                # the name of your model
  hive_table_qualified_names:              # this is a predefined key to link to
                                           # training data
```

```
    - "db.table1@namespace"                # the qualifiedName of the hive_table
                                           # object representing training data
    - "db.table2@ns"
  metadata:                                # this is a predefined key for
                                           # additional metadata
    key1: value1
    key2: value2
    query: "select id, name from table"    # suggested use case: query used to
                                           # extract training data
    training_file: "fit.py"                # suggested use case: training file
                                           # used
modelName2:                                # multiple models can be specified in
                                           # one file
  hive_table_qualified_names:
    - "db.table2@ns"
```

Viewing lineage for a model deployment in Atlas

You can view the lineage information for a particular model deployment and trace it back to the specific data that was used to train the model through the Atlas' Management Console.

Procedure

1. Navigate to Management Console Environments , select your environment, and then under Quick Links select Atlas.
2. Search for ml_model_deployment. Click the model deployment of your interest.
3. Click the Lineage tab to see a visualization of lineage information for the particular model deployment and trace it back to the specific data that was used to train the model.

   You can also search for a specific table, click through to its Lineage tab and see if the table has been used in any model deployments.

Cloudera AI

Using Model Hub

Date published: 2020-07-16
Date modified: 2025-01-29

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Model Hub.....4
Role-based authorization in Model Hub.....4
Importing models from NVIDIA NGC.....4
Importing models from Hugging Face (Technical Preview).....6

Using Model Hub

You can easily import the models listed in the Model Hub into the Registered Models page and then deploy it using the Cloudera AI Inference service. This streamlines the workflow of developers working on AI use cases by simplifying the process of discovering, deploying, and testing models.

Model Hub is a catalog of top-performing LLM and generative AI models. The Model Hub page displays the list of different models along with their source type, tags, and description. You can import models listed on the Model Hub page and deploy it from the Registered Models page.

Role-based authorization in Model Hub

Model Hub implements role-based access control.

Users need to have EnvironmentUser role along with one of the following roles in the environment tied to AI Registry that the model is being imported to:

- MLAdmin (admin user)
- MLUser

Without these roles, if you try to import models, the Failed to Fetch error or something similar is displayed.

For more information about the access control for the registered models, see *Model access control*.

Related Information

Model access control

Importing models from NVIDIA NGC

You can import the NVIDIA NGC Catalog models listed in the Model Hub page and deploy it from the Registered Models page.

Before you begin

To import models, you must add the following **NVIDIA GPU Cloud (NGC)** URL details so they can be allowed in the firewall's rules.

- prod.otel.kaizen.nvidia.com (NVIDIA open telemetry)
- api.ngc.nvidia.com
- files.ngc.nvidia.com

Procedure

1. In the **Cloudera** console, click the **Cloudera AI** tile.

   The **Cloudera AI Workbenches** page displays.
2. Click **Model Hub** under **AI Hub** in the left navigation menu.

   The **Model Hub** page displays. The page lists different models along with its source type, tags, and description.

3. Click Import on the model you want to import.
   The **Import Model** page displays.

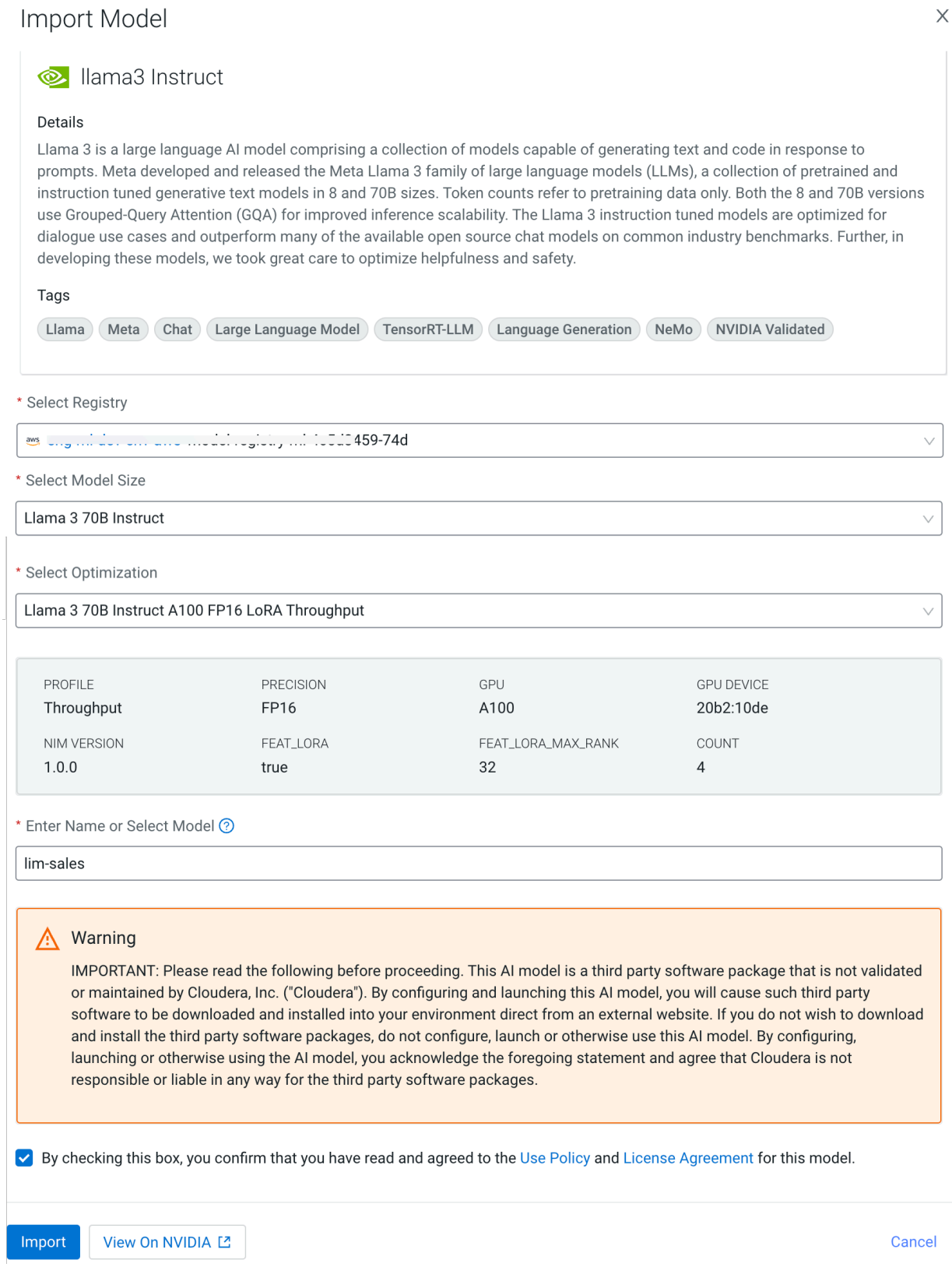

4. In the Select Registry drop-down list, select the AI registry to which you want to import the model.

5. In the Select Parameter drop-down list, select the model variant.

   Note: After you select the parameter, the View On NVIDIA option is enabled. Click on it to view the model details of the selected model on the *NVIDIA* website.

6. In the Select Optimization drop-down list, select the model variant that is optimized for your use case and domain.

   The details of the chosen optimization profile is displayed.

7. In the Enter Name or Select Model field, select a name from the existing list or enter a new name for the model you are importing.
8. If displayed, read the User Policy and License Agreement, and click the checkbox if you agree.
9. Click Import.

   The **Model Hub** page is displayed with the message that the model is imported successfully.

Results

You can click Registered Models in the left navigation menu to view the newly imported model.

Importing models from Hugging Face (Technical Preview)

You can import the Hugging Face models listed on the Model Hub page into your Model Registry. If your preferred model is not listed on the Model Hub page, you can import it from the Registered Models page. After you import the model, the newly imported model will be listed on the Registered Models page.

Note: This feature is in Technical Preview and not recommended for production deployments. Cloudera recommends that you try this feature in test or development environments.

Procedure

1. In the **Cloudera** console, click the **Cloudera AI** tile.

   The **Cloudera AI Workbenches** page displays.

2. Click **Model Hub** under **AI Hub** in the left navigation menu.

   The **Model Hub** page displays. The page lists different models along with its source type, tags, and description.

3. Click Import on the model you want to import.

The **Import Model** page displays.

Import Model

StarCoder

Details

The StarCoder models are 15.5B parameter models trained on 80+ programming
requests excluded. The model uses Multi Query Attention, a context window of 8
Middle objective on 1 trillion tokens.

Tags

bigCode | StarCoder | Code Generation | Text Generation | Multilingual suppo

* Select AI Registry

eng-ml-dev-env-azure model-registry-ml-a857b6bd-4a6

* Select Model Size

StarCoder

* Select Optimization

starcoder

A10G	A100	L40S
2	1	1

* Enter Name or Select Model

google-startcoder

4. In the Select AI Registry drop-down list, select the model registry to which you want to import the model.
5. In the Select Model Size drop-down list, select the model size.
6. In the Select Optimization drop-down list, select the optimization profile. It displays the recommended GPU counts for the specific GPU of the optimization profile.
7. In the Enter Name or Select Model field, select a name from the existing list or enter a new name for the model you are importing.
8. For the gated models of Hugging Face, an access token is required. The Hugging Face Token field is displayed only if it requires a token. Enter the access token obtained from the Hugging Face website if this field is displayed.

   Important: Consent on Hugging Face and Access Token Required

   Some models like gated models on Hugging Face require an access token. To obtain access tokens from the Hugging Face website, follow the instructions displayed on the Import Model page.

   In addition to access tokens, some models on Hugging Face require you to log in to your Hugging Face account, navigate to the model, and accept an agreement. After accepting the agreement, your existing Hugging Face token will work. Follow the instructions displayed on the Import Model page to know how to accept the agreement on *Hugging Face*.
9. Click Import. The Model Hub page displays a message that the model import has been triggered successfully along with a button to view the status of that import process.

Results

You can click Model Registry in the left navigation menu to view the newly imported model.

Cloudera AI

Model Metrics

Date published: 2020-07-16
Date modified: 2024-11-21

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enabling model metrics 4

Tracking model metrics without deploying a model 4

Tracking metrics for deployed models 5

Enabling model metrics

Metrics are used to track the performance of the models. When you enable model metrics while creating a workbench, the metrics are stored in a scalable metrics store. You can track individual model predictions and analyze metrics using custom code.

About this task

Procedure

1. Go to Cloudera AI and click Provision Workbench on the top-right corner.
2. Enter the Cloudera AI Workbench name and other details.
3. Click Advanced Options.
4. Select Enable Model Metrics.

   If you want to connect to an external (custom) Postgres database, then specify the details in the additional optional arguments that are displayed. If you do not specify these details, a managed Postgres database will be used to store the metrics.

Tracking model metrics without deploying a model

Cloudera recommends that you develop and test model metrics in a workbench session before actually deploying the model. This workflow avoids the need to rebuild and redeploy a model to test every change.

Metrics tracked in this way are stored in a local, in-memory datastore instead of the metrics database, and are no longer stored when the session exits. You can access these metrics in the same session using the regular metrics API in the cdsw.py file.

The following example demonstrates how to track metrics locally within a session, and use the read_metrics function to read the metrics in the same session by querying by the time window.

To try this feature in the local development mode, use the following files from the Python template project:

- use_model_metrics.py
- predict_with_metrics.py

The predict function from the predict_with_metrics.py file shown in the following example is similar to the function with the same name in the predict.py file. It takes input and returns output, and can be deployed as a model.
But unlike the function in the predict.py file, the predict function from the predict_with_metrics.py file tracks mathematical metrics. These metrics can include information such as input, output, feature values, convergence metrics, and error estimates. In this simple example, only input and output are tracked. The function is equipped to track metrics by applying the decorator cdsw.model_metrics.

```
@cdsw.model_metrics
def predict(args):
  # Track the input.
  cdsw.track_metric("input", args)

  # If this model involved features, ie transformations of the
  # raw input, they could be tracked as well.
  # cdsw.track_metric("feature_vars", {"a":1,"b":23})

  petal_length = float(args.get('petal_length'))
  result = model.predict([[petal_length]])
  # Track the output.
  cdsw.track_metric("predict_result", result[0][0])
```

```
  return result[0][0]
```

You can directly call this function in a workbench session, as shown in the following example:

```
predict(
{"petal_length": 3}
)
```

You can fetch the metrics from the local, in-memory datastore by using the regular metrics API. To fetch the metrics, set the dev keyword argument to True in the use_model_metrics.py file. You can query the metrics by model, model build, or model deployment using the variables cdsw.dev_model_crn and cdsw.dev_model_build_crn or cdsw.dev _model_deploy_crn respectively.

For example:

```
end_timestamp_ms=int(round(time.time() * 1000))
cdsw.read_metrics(model_deployment_crn=cdsw.dev_model_deployment_crn,
start_timestamp_ms=0,
end_timestamp_ms=end_timestamp_ms,
dev=True)
```

where CRN denotes Cloudera Resource Name, which is a unique identifier from Cloudera Data Platform, analogous to Amazon's ARN.

Tracking metrics for deployed models

When you have finished developing your metrics tracking code and the code that consumes the metrics, simply deploy the predict function from predict_with_metrics.py as a model. No code changes are necessary.

Calls to read_metrics, track_delayed_metrics, and track_aggregate_metrics need to be changed to take the CRN of the deployed model, build or deployment. These CRNs can be found in the model’s **Overview** page.

Calls to the call_model function also requires the model’s access key (model_access_key in use_model_metrics.py) from the model’s **Settings** page. If authentication has been enabled for the model (the default), a model API key for the user (model_api_token in use_model_metrics.py) is also required. This can be obtained from the user's **Settings** page.

Cloudera AI

Models

Date published: 2020-07-16
Date modified: 2025-01-29

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Model Training and Deployment Overview 5

Models 5
Models - Concepts and Terminology 6

Challenges with Machine Learning in production 8
Challenges with model deployment and serving 8
Challenges with model monitoring 8
Challenges with model governance 10
Model visibility 10
Model explainability, interpretability, and reproducibility 10
Model governance using Apache Atlas 11

Using Cloudera AI Registry 12
Setting up Cloudera AI Registry 13
Creating a Cloudera AI Registry 13
Creating a Cloudera AI Registry on an Azure UDR Private Cluster 14
Setting up access for Cloudera AI Registry in a RAZ-enabled environment 15
Setting up access for Cloudera AI Registry in a non-RAZ-enabled environment 18
Synchronizing Cloudera AI Registry with a workbench 18
Viewing details for Cloudera AI Registries 20
Cloudera AI Registry permissions 20
Model access control 20
Deleting Cloudera AI Registry 21
Registering and deploying models with Cloudera AI Registry 21
Creating a model using MLflow 21
Registering a model using the AI Registry user interface 21
Registering a model using MLflow SDK 22
Viewing registered model information 23
Creating a new version of a registered model 24
Deploying a model from the **AI Registry** page 24
Deploying a model from the destination Project page 27
Viewing Details for Cloudera AI Registry 27
Delete a model from Cloudera AI Registry 27
Disabling Cloudera AI Registry 28
Upgrade Cloudera AI Registry 28
Roll back the registry upgrade 28
Cloudera AI Registry standalone API 29
Prerequisites for Cloudera AI Registry standalone API 29
Authenticating clients for interacting with Cloudera AI Registry API 30
Role-based authorization 30
Using the REST Client 30
Cloudera AI Registry CLI Client 33
Known issues with Cloudera AI Registry standalone API 34
Updating Cloudera AI Registry configuration 35
Troubleshooting issues with Cloudera AI Registry API 35

Creating and deploying a Model....40

Usage guidelines for deploying models with Cloudera AI....44

Known Issues and Limitations with Model Builds and Deployed Models....45

Model Request and Response Formats....47

Testing calls to a Model....47

Securing Models....49
Access Keys for Models....49
API Key for Models....49
Enabling authentication....50
Generating an API key....50
Managing API Keys....51

Workflows for active Models....51

Technical metrics for Models....53

Debugging issues with Models....53

Deleting a Model....54

Configuring model request payload size....55

Managing Cloudera Copilot....55
Prerequisites to set up Cloudera Copilot....56
Configuring Cloudera Copilot....56
Using Cloudera Copilot....59

Example - Model training and deployment (Iris)....61
Training the Model....62
Deploying the Model....64

Model Training and Deployment Overview

This section provides an overview of model training and deployment using Cloudera AI.

Machine learning is a discipline that uses computer algorithms to extract useful knowledge from data. There are many different types of machine learning algorithms, and each one works differently. In general however, machine learning algorithms begin with an initial hypothetical model, determine how well this model fits a set of data, and then work on improving the model iteratively. This training process continues until the algorithm can find no additional improvements, or until the user stops the process.

A typical machine learning project will include the following high-level steps that will transform a loose data hypothesis into a model that serves predictions.

1. Explore and experiment with and display findings of data
2. Deploy automated pipelines of analytics workloads
3. Train and evaluate models
4. Deploy models as REST APIs to serve predictions

With Cloudera AI, you can deploy the complete lifecycle of a machine learning project from research to deployment.

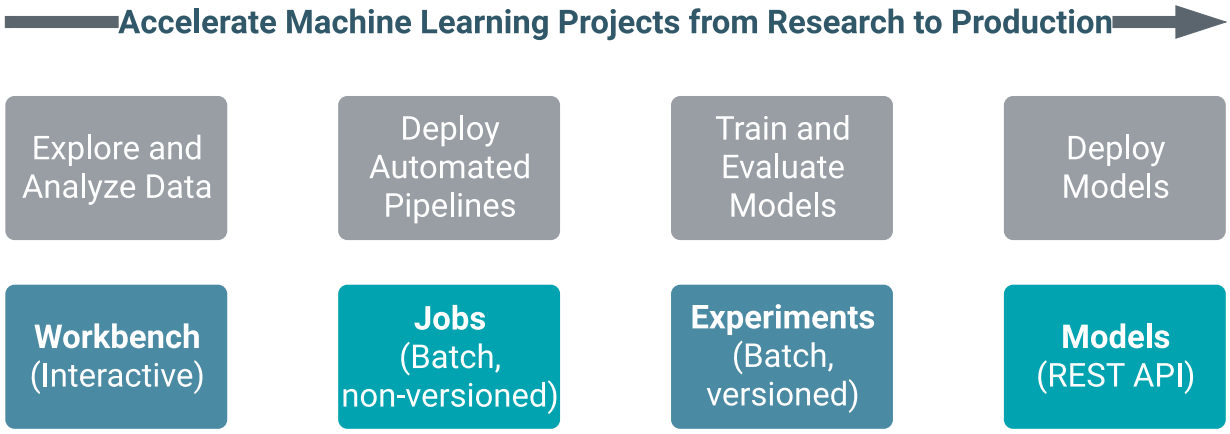

Models

Cloudera AI allows data scientists to build, deploy, and manage models as REST APIs to serve predictions.

Challenge

Data scientists often develop models using a variety of Python/R open source packages. The challenge lies in actually exposing those models to stakeholders who can test the model. In most organizations, the model deployment process will require assistance from a separate DevOps team who likely have their own policies about deploying new code.

For example, a model that has been developed in Python by data scientists might be rebuilt in another language by the devops team before it is actually deployed. This process can be slow and error-prone. It can take months to deploy new models, if at all. This also introduces compliance risks when you take into account the fact that the new re-developed model might not be even be an accurate reproduction of the original model.

Once a model has been deployed, you then need to ensure that the devops team has a way to rollback the model to a previous version if needed. This means the data science team also needs a reliable way to retain history of the models they build and ensure that they can rebuild a specific version if needed. At any time, data scientists (or any other stakeholders) must have a way to accurately identify which version of a model is/was deployed.

Solution

Cloudera AI allows data scientists to build and deploy their own models as REST APIs. Data scientists can now select a Python or R function within a project file, and Cloudera AI will:

- Create a snapshot of model code, model parameters, and dependencies.
- Package a trained model into an immutable artifact and provide basic serving code.
- Add a REST endpoint that automatically accepts input parameters matching the function, and that returns a data structure that matches the function's return type.
- Save the model along with some metadata.
- Deploy a specified number of model API replicas, automatically load balanced.

Models - Concepts and Terminology

Model

Model is a high level abstract term that is used to describe several possible incarnations of objects created during the model deployment process. For the purpose of this discussion you shall note that 'model' does not always refer to a specific artifact. More precise terms (as defined later in this section) shall be used whenever possible.

Stages of the Model Deployment Process

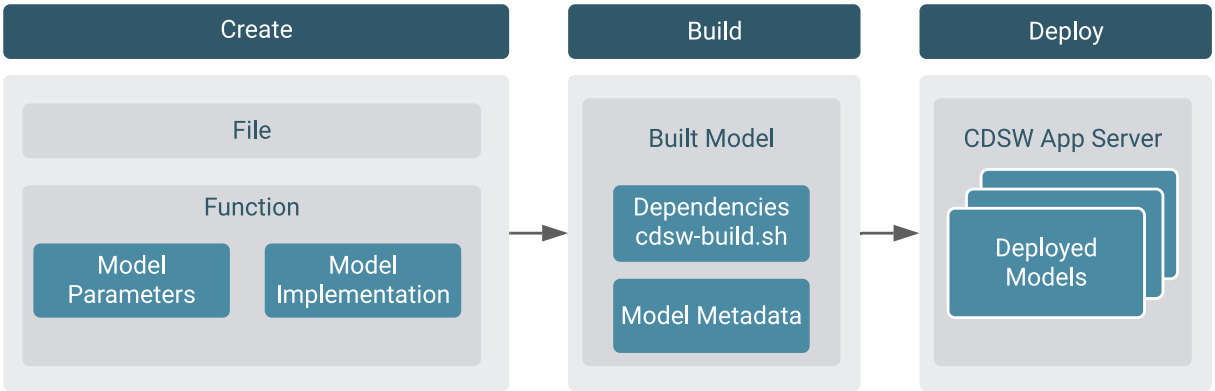

The rest of this section contains supplemental information that describes the model deployment process in detail.

Create

- File - The R or Python file containing the function to be invoked when the model is started.
- Function - The function to be invoked inside the file. This function should take a single JSON-encoded object (for example, a python dictionary) as input and return a JSON-encodable object as output to ensure compatibility with any application accessing the model using the API. JSON decoding and encoding for model input/output is built into Cloudera AI.

  The function will likely include the following components:

  - Model Implementation

    The code for implementing the model (e.g. decision trees, k-means). This might originate with the data scientist or might be provided by the engineering team. This code implements the model's predict function, along with any setup and teardown that may be required.
  - Model Parameters

    A set of parameters obtained as a result of model training/fitting (using experiments). For example, a specific decision tree or the specific centroids of a k-means clustering, to be used to make a prediction.

Build

This stage takes as input the file that calls the function and returns an artifact that implements a single concrete model, referred to as a model build.

- Built Model

  A built model is a static, immutable artifact that includes the model implementation, its parameters, any runtime dependencies, and its metadata. If any of these components need to be changed, for example, code changes to the implementation or its parameters need to be retrained, a new build must be created for the model. Model builds are versioned using build numbers.

  To create the model build, Cloudera AI creates a Docker image based on the engine designated as the project's default engine. This image provides an isolated environment where the model implementation code will run.

  To configure the image environment, you can specify a list of dependencies to be installed in a build script called cdsw-build.sh.

  For details about the build process and examples on how to install dependencies, see *Engines for Experiments and Models*.
- Build Number:

  Build numbers are used to track different versions of builds within the scope of a single model. They start at 1 and are incremented with each new build created for the model.

Deploy

This stage takes as input the memory/CPU resources required to power the model, the number of replicas needed, and deploys the model build created in the previous stage to a REST API.

- Deployed Model

  A deployed model is a model build in execution. A built model is deployed in a model serving environment, likely with multiple replicas.
- Environmental Variable

  You can set environmental variables each time you deploy a model. Note that models also inherit any environment variables set at the project and global level. (For more information see *Engine Environment Variables*.) However, in case of any conflicts, variables set per-model will take precedence.

  Note: If you are using any model-specific environmental variables, these must be specified every time you re-deploy a model. Models do not inherit environmental variables from previous deployments.
- Model Replicas

  The engines that serve incoming requests to the model. Note that each replica can only process one request at a time. Multiple replicas are essential for load-balancing, fault tolerance, and serving concurrent requests. Cloudera AI allows you to deploy a maximum of 9 replicas per model.
- Deployment ID

  Deployment IDs are numeric IDs used to track models deployed across Cloudera AI. They are not bound to a model or project.

Related Information

Engines Environment Variables

Challenges with Machine Learning in production

One of the hardest parts of Machine Learning (ML) is deploying and operating ML models in production applications. These challenges fall mainly into the following categories: model deployment and serving, model monitoring, and model governance.

Challenges with model deployment and serving

After models are trained and ready to deploy in a production environment, lack of consistency with model deployment and serving workflows can present challenges in terms of scaling your model deployments to meet the increasing numbers of ML usecases across your business.

Many model serving and deployment workflows have repeatable, boilerplate aspects which you can automate using modern DevOps techniques like high frequency deployment and microservices architectures. This approach can enable the ML engineers to focus on the model instead of the surrounding code and infrastructure.

Challenges with model monitoring

Machine Learning (ML) models predict the world around them which is constantly changing. The unique and complex nature of model behavior and model lifecycle present challenges after the models are deployed.

Cloudera AI provides you the capability to monitor the performance of the model on two levels: technical performance (latency, throughput, and so on similar to an Application Performance Management), and mathematical performance (is the model predicting correctly, is the model biased, and so on).

There are two types of metrics that are collected from the models:

- Time series metrics: Metrics measured in-line with model prediction. It can be useful to track the changes in these values over time. It is the finest granular data for the most recent measurement. To improve performance, older data is aggregated to reduce data records and storage.
- Post-prediction metrics: Metrics that are calculated after prediction time, based on ground truth and/or batches (aggregates) of time series metrics. To collect metrics from the models, the Python SDK has been extended to include the following functions that you can use to store different types of metrics:

To collect metrics from the models, the Python SDK has been extended to include the following functions that you can use to store different types of metrics:

- track_metrics: Tracks the metrics generated by experiments and models.
- read_metrics: Reads the metrics already tracked for a deployed model, within a given window of time.
- track_delayed_metrics: Tracks metrics that correspond to individual predictions, but are not known at the time the prediction is made. The most common instances are ground truth and metrics derived from ground truth such as error metrics.
- track_aggregate_metrics: Registers metrics that are not associated with any particular prediction. This function can be used to track metrics accumulated and/or calculated over a longer period of time.

The following two use-cases show how you can use these functions:

- Tracking accuracy of a model over time
- Tracking drift

Usecase 1: Tracking accuracy of a model over time

Consider the case of a large telco. When a customer service representative takes a call from a customer, a web application presents an estimate of the risk that the customer will churn. The service representative takes this risk into account when evaluating whether to offer promotions.

The web application obtains the risk of churn by calling into a model hosted on Cloudera AI. For each prediction thus obtained, the web application records the UUID into a datastore alongside the customer ID. The prediction itself is tracked in Cloudera AI using the track_metrics function.

At some point in the future, some customers do in fact churn. When a customer churns, they or another customer service representative close their account in a web application. That web application records the churn event, which is ground truth for this example, in a datastore.

An ML engineer who works at the telco wants to continuously evaluate the suitability of the risk model. To do this, they create a recurring Cloudera AI job. At each run, the job uses the read_metrics function to read all the predictions that were tracked in the last interval. It also reads in recent churn events from the ground truth datastore. It joins the churn events to the predictions and customer ID's using the recorded UUID's, and computes an Receiver operating characteristic (ROC) metric for the risk model. The ROC is tracked in the metrics store using the track_aggregate_metrics function.

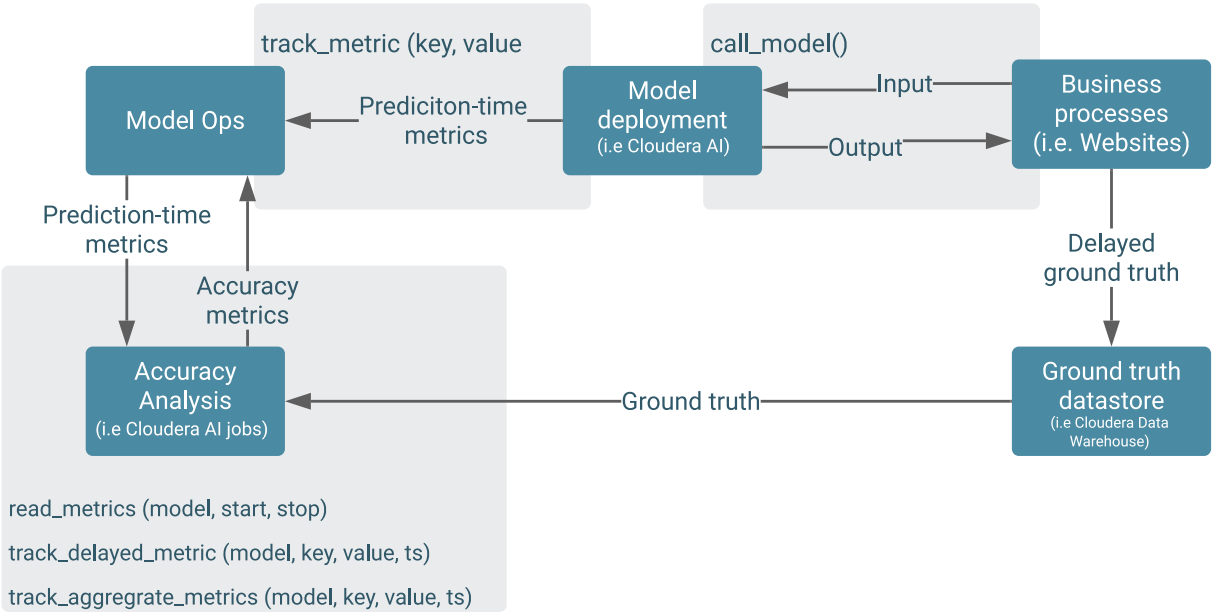

Note: You can store the ground truth in an external datastore, such as Cloudera Data Warehouse or in the metrics store.

Use-case 2: Tracking drift

Instead of or in addition to computing ROC, the ML engineer may need to track various types of drift. Drift metrics are especially useful in cases where ground truth is unavailable or is difficult to obtain.

The definition of drift is broad and somewhat nebulous and practical approaches to handling it are evolving, but drift is always about changing distributions. The distribution of the input data seen by the model may change over time and deviate from the distribution in the training dataset, and/or the distribution of the output variable may change, and/or the relationship between input and output may change.

All drift metrics are computed by aggregating batches of predictions in some way. As in the use case above, batches of predictions can be read into recurring jobs using the read_metrics function, and the drift metrics computed by the job can be tracked using the track_aggregate_metrics function.

Challenges with model governance

Businesses implement ML models across their entire organization, spanning a large spectrum of usecases. When you start deploying more than just a couple models in production, a lot of complex governance and management challenges arise.

Almost all the governance needs for ML are associated with data and are tied directly to the data management practice in your organization. For example, what data can be used for certain applications, who should be able to access what data, and based on what data are models created.

Some of the other unique governance challenges that you could encounter are:

- How to gain visibility into the impact your models have on your customers?
- How can you ensure you are still compliant with both governmental and internal regulations?
- How does your organization's security practices apply to the models in production?

Ultimately, the needs for ML governance can be distilled into the following key areas: model visibility, and model explainability, interpretability, and reproducibility.

Model visibility

A basic requirement for model governance is enabling teams to understand how machine learning is being applied in their organizations. This requires a canonical catalog of models in use. In the absence of such a catalog, many organizations are unaware of how their models work, where they are deployed, what they are being used for, and so on. This leads to repeated work, model inconsistencies, recomputing features, and other inefficiencies.

Model explainability, interpretability, and reproducibility

Models are often seen as a black box: data goes in, something happens, and a prediction comes out. This lack of transparency is challenging on a number of levels and is often represented in loosely related terms explainability, interpretability, and reproducibility.

- Explainability: Indicates the description of the internal mechanics of an Machine Learning (ML) model in human terms
- Interpretability: Indicates the ability to:
  - Understand the relationship between model inputs, features and outputs
  - Predict the response to changes in inputs
- Reproducibility: Indicates the ability to reproduce the output of a model in a consistent fashion for the same inputs

To solve these challenges, Cloudera AI provides an end-to-end model governance and monitoring workflow that gives organizations increased visibility into their machine learning workflows and aims to eliminate the blackbox nature of most machine learning models.

The following image shows the end-to-end producltion ML workflow:

Figure 1: Production ML Workflow

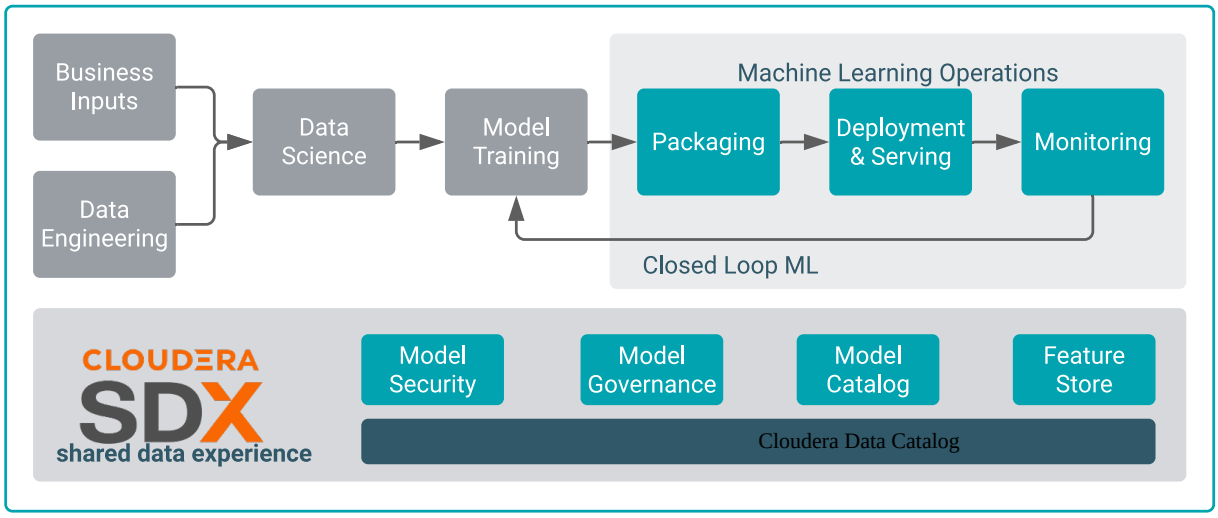

Model governance using Apache Atlas

To address governance challenges, Cloudera AI uses Apache Atlas to automatically collect and visualize lineage information for data used in Cloudera AI workflows — from training data to model deployments.

Lineage is a visual representation of the project. The lineage information includes visualization of the relationships between model entities such as code, model builds, deployments, and so on. and the processes that carry out transformations on the data involved, such as create project, build model, deploy model, and so on.

The Apache Atlas type system has pre-built governance features that can be used to define ML metadata objects. A type in Atlas is a definition of the metadata stored to describe a data asset or other object or process in an environment. For ML governance, Cloudera has designed new Atlas types that capture ML entities and processes as Atlas metadata objects.

In addition to the definition of the types, Atlas also captures the relationship between the entities and processes to visualize the end-to-end lineage flow, as shown in the following image. The blue hexagons represent an entity (also called the noun) and the green hexagons represent a process (also called the verb).

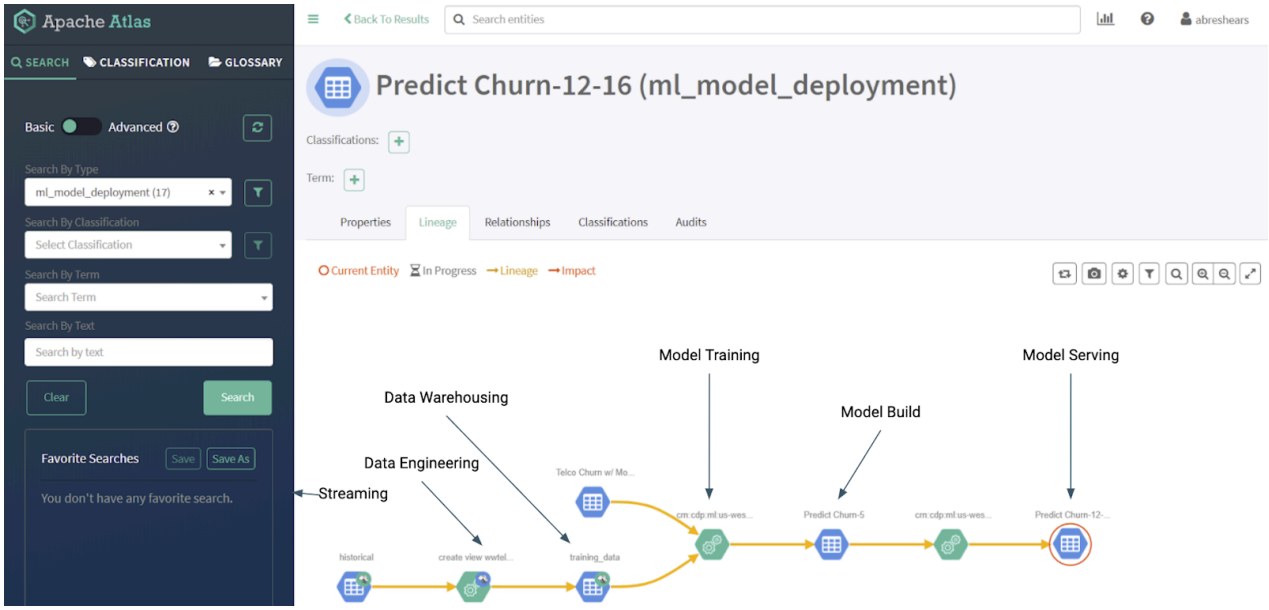

The ML metadata definition closely follows the actual machine learning workflow. Training data sets are the starting point for a model lineage flow. These data sets can be tables from a data warehouse or an embedded csv file. Once a data set has been identified, the lineage follows into training, building and deploying the model.

See *ML operations entities created in Atlas* for a list of metadata that Atlas collects from each Cloudera AI Workbench. Metadata is collected from machine learning projects, model builds, and model deployments, and the processes that create these entities.

Using Cloudera AI Registry

Cloudera AI Registry is the core enabler for MLOps, or DevOps for machine learning.

Cloudera AI Registry stores and manages machine learning models and associated metadata, such as the model's version, dependencies, and performance. The registry enables MLOps and facilitates the development, deployment, and maintenance of machine learning models in a production environment.

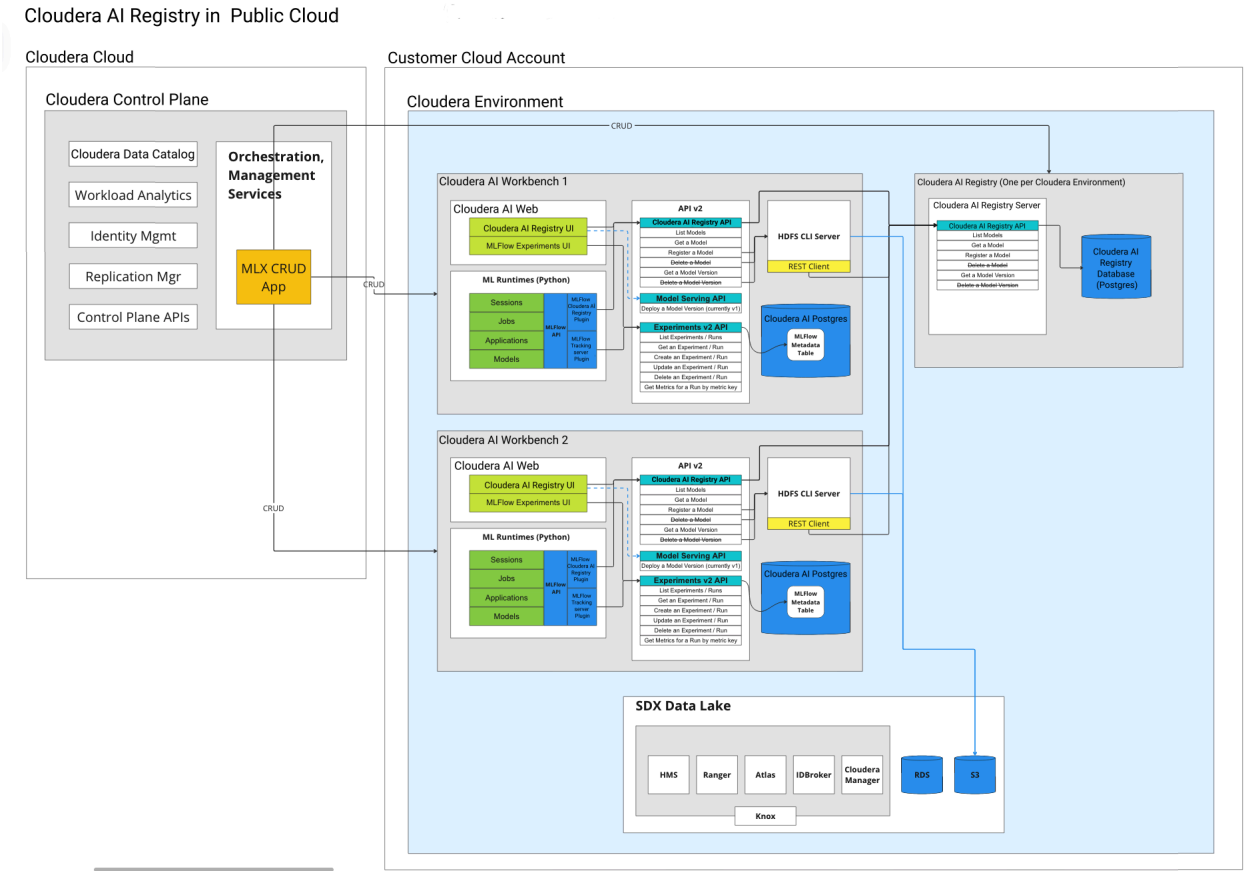

Cloudera AI Registry includes functionality for the following tasks:

- Storing and organizing different versions of a machine learning model and its associated metadata.
- Tracking the lineage of a model, including who created it, when it was created, and any changes made to it over time.
- Providing APIs for accessing and deploying models, as well as for querying and searching the registry.
- Integrating with CI/CD pipelines and other tools used in the MLOps workflow.

Cloudera AI Registry instances help organizations improve the quality and reliability of their machine learning models by providing a centralized location for storing and managing models, as well as enabling traceability and reproducibility of model development. They also make deploying and managing models in a production environment easier by providing a single source for model versions and dependencies.

The Cloudera AI Registry integrates MLFlow and maintains compatibility with the open source ecosystem.

Limitations

- Upgrade to the General Availability (GA) version of Cloudera AI Registry might not be supported. Alternatively, upgrade to the GA version of Cloudera AI Registry might require reinstalling Cloudera AI Registry which could result in loss of Cloudera AI Registry data configured with the technical preview (TP) version of Cloudera AI Registry.

Setting up Cloudera AI Registry

Cloudera AI Registry is the core enabler for MLOps, or DevOps for machine learning.

Creating a Cloudera AI Registry

Before you can start using Cloudera AI Registry you must create a AI registry for your environment.

Procedure

1. In the **Cloudera** console, click the **Cloudera AI** tile.
   The **Cloudera AI Workbenches** page displays.
2. Click AI Registries in the left navigation pane.
3. Click Create AI Registry.
   Cloudera AI displays the Create AI Registry dialog box.
4. Choose your environment from the Environment Name drop down list.
5. Depending on your environment, complete one of the following:
   a) If your environment is in AWS, **AI Registry** displays the following dialog box:

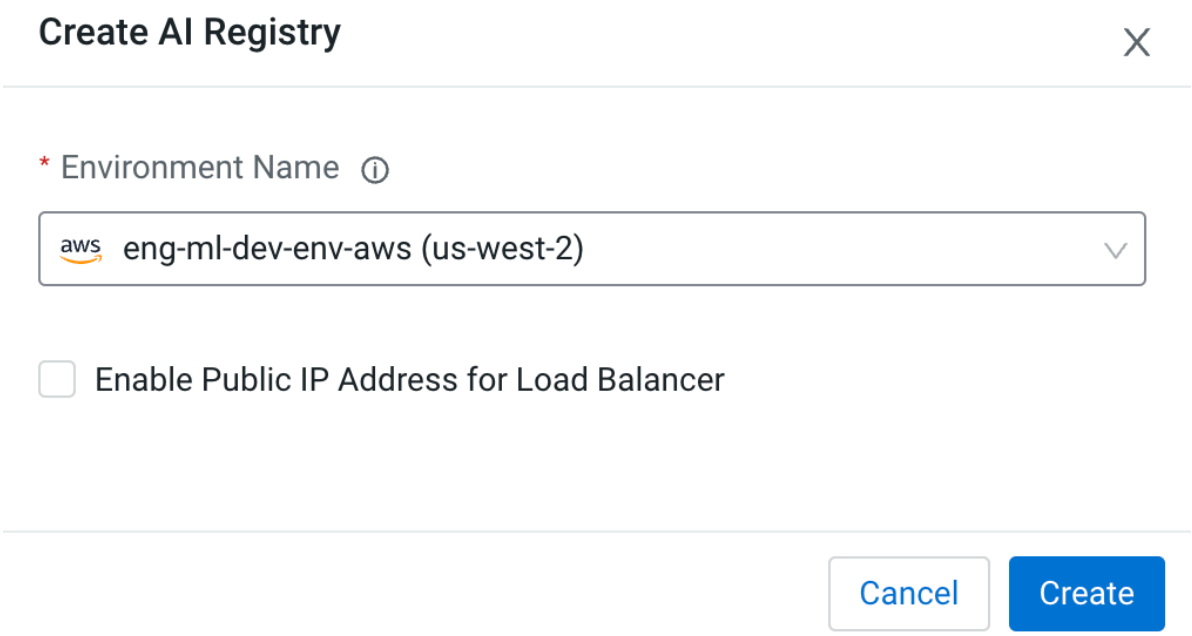

   1. From the Environment Name dropdown list, select your environment.
   2. Enable Public IP Address for Load Balancer: By default, AI Registry service uses a private load balancer for cluster ingress. If you use a public load balancer instead, select the Enable Public IP Address for Load

Balancer option. If you use a private load balancer for cluster ingress, you must have a VPN connection between your corporate network and the Virtual Private Cloud (VPC) in which the AI Registry is deployed.

3. click Create to create the Cloudera AI Registry.

b) If your environment is in Azure, Cloudera AI Registry displays the following dialog box:

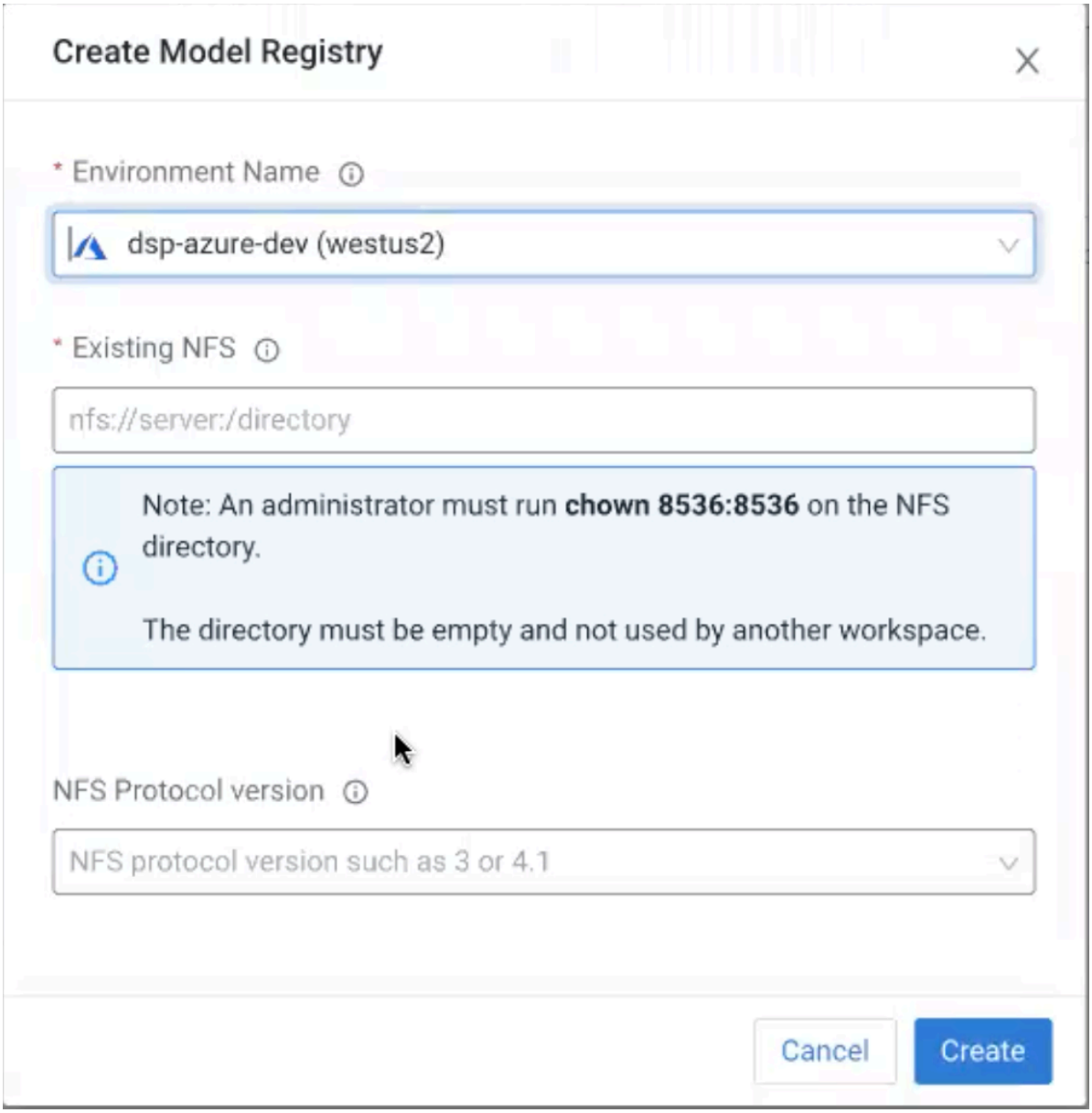

1. Choose the Azure environment for the Cloudera AI Registry.
2. Click Create to create the Cloudera AI Registry.

Creating a Cloudera AI Registry on an Azure UDR Private Cluster

Use the following template Cloudera Data Platform CLI command to create a UDR private cluster on Azure with a Cloudera AI Registry.

You must replace the following template items with your own information.

- <environment CRN>
- <environment name> (in two places)
- <subnet>

Model registries are also supported on Azure private clusters with UDR. For more information about UDR, see the Preview Feature documentation.

If you have not yet downloaded the Cloudera Data Platform CLI tool, see the documentation.

Use the latest version of the Cloudera Data Platform CLI.

Cloudera Data Platform CLI command to create a Cloudera AI Registry

This Cloudera Data Platform CLI command performs has three key sections:

1. Enables support for private clusters in Azure ("privateCluster": true,)
2. Enables UDR for the private cluster ("outboundTypes": ["OUTBOUND_TYPE_UDR"],)
3. Specifies the subnet for the UDR-enabled private cluster ("subnets")

```
cdp ml create-model-registry --cli-input-json {
    "environmentCrn": "<environment CRN>",
    "environmentName": "<environment name>",
    "privateCluster": true,
    "usePublicLoadBalancer": false,
    "outboundTypes": [
        "OUTBOUND_TYPE_UDR"
    ],
    "provisionK8sRequest": {
        "network": {
            "topology": {
                "subnets": [
                  "<subnet>" # subnet with a default route configuration to
forward the traffic to the network appliance or firewall. This is required t
o enable UDR.
                ]
            }
        }
    }
}
```

Setting up access for Cloudera AI Registry in a RAZ-enabled environment

In a RAZ-enabled environment you need to set up the S3-Ranger policy by manually adding the machine user name in the S3 Ranger policy.

To set up the S3-Ranger policy, complete the following:

1. On the AI Registries Details page, find and copy the Machine User Workload User Name in the Machine User Workload User Name field.

   For example, in the following screenshot, the Machine User Workload User Name field contains srv_cml_env_machine_user_8dcf2. Copy the Machine User Workload User Name which is 8dcf2.

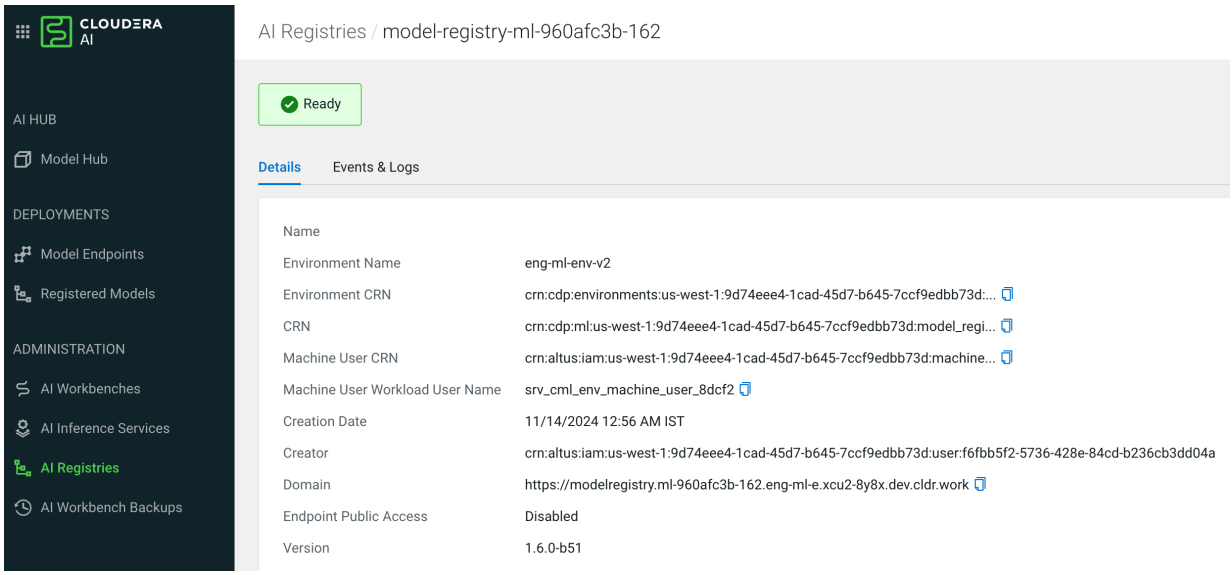

2. Go to the Ranger UI in the Datalake of the environment.

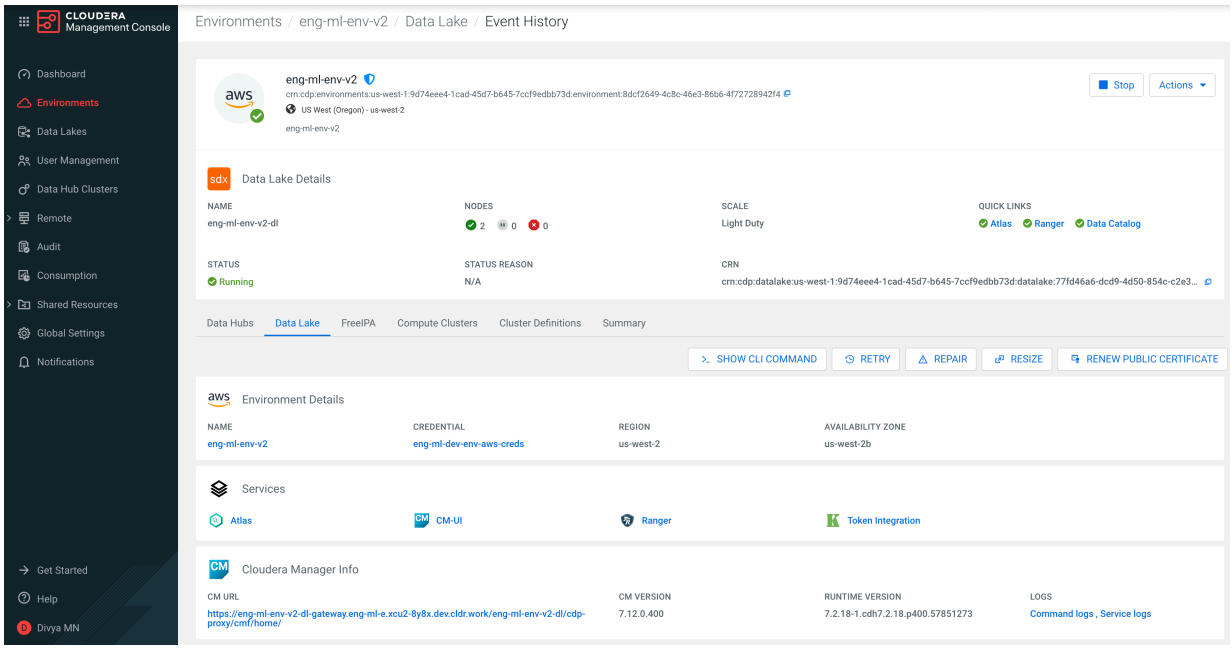

3. Depending on your environment, select cm_s3 (AWS) or cm_adls (Azure).

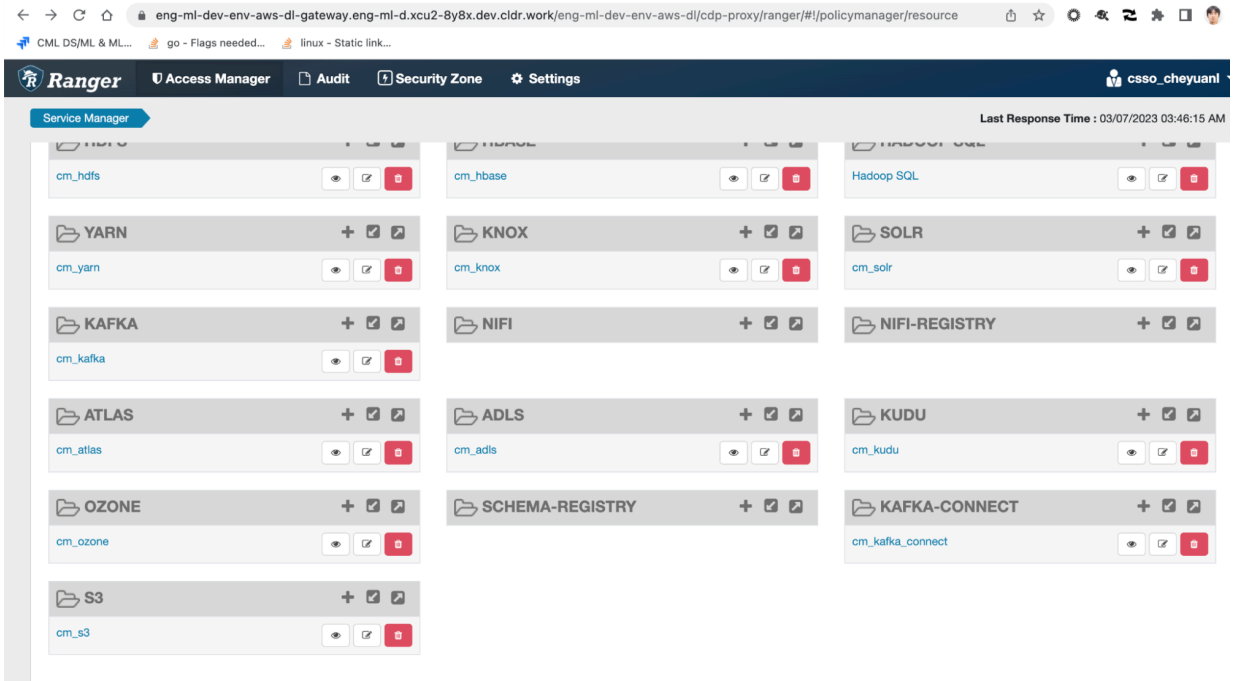

4. Go to the policy named all - bucket, path which controls the access to the object store bucket.

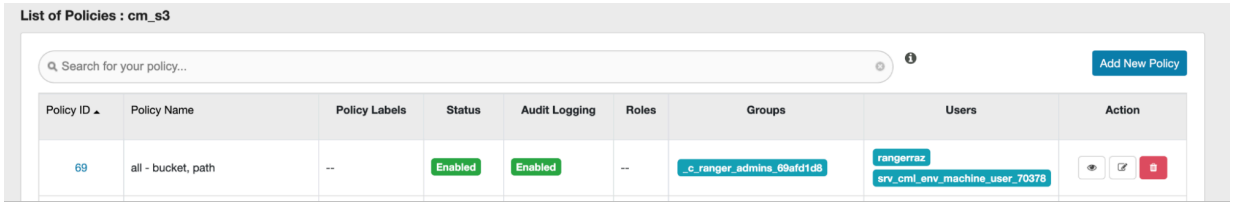

5. Enter the Machine User Workload User Name in the Select User field in the allow conditions section.

   For example, using the Machine User Workload User Name from Step 2, add the value which is 82a49.

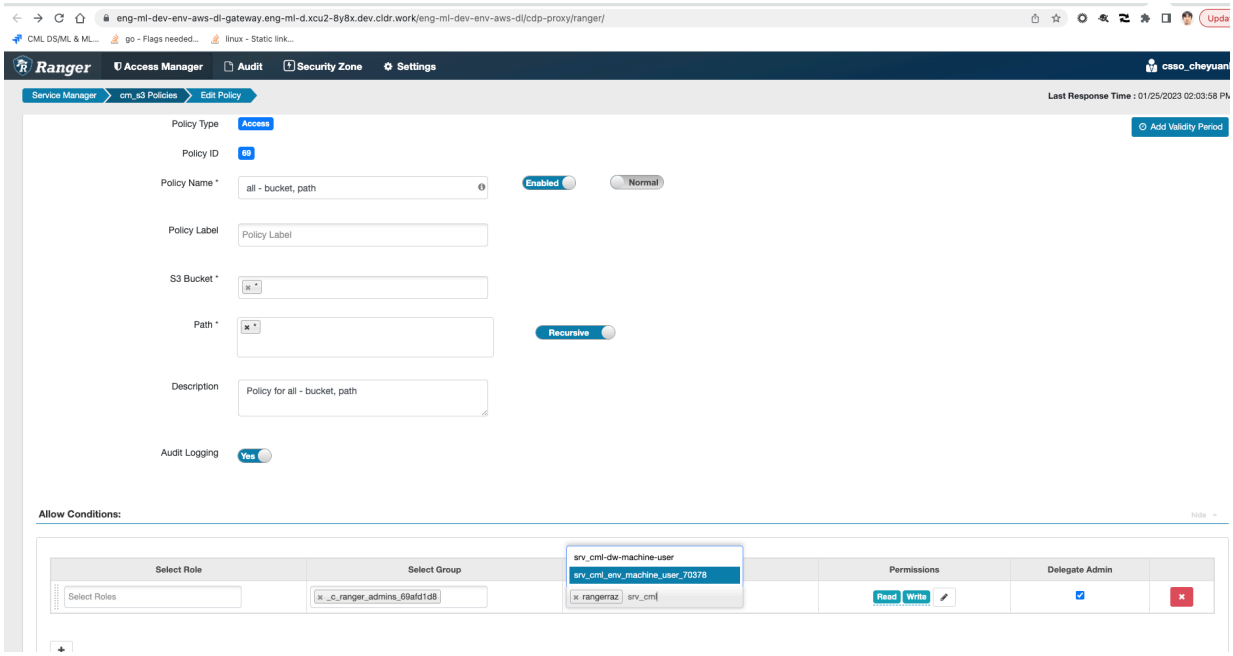

Setting up access for Cloudera AI Registry in a non-RAZ-enabled environment

In a non-RAZ-enabled environment you need to add the Machine User CRN to the IDBroker mapping in order to access the S3/ADLS buckets.

To add the Machine User CRN to the IDBroker mapping complete the following:

1. Locate the Machine User CRN in the AI Registries Details page.

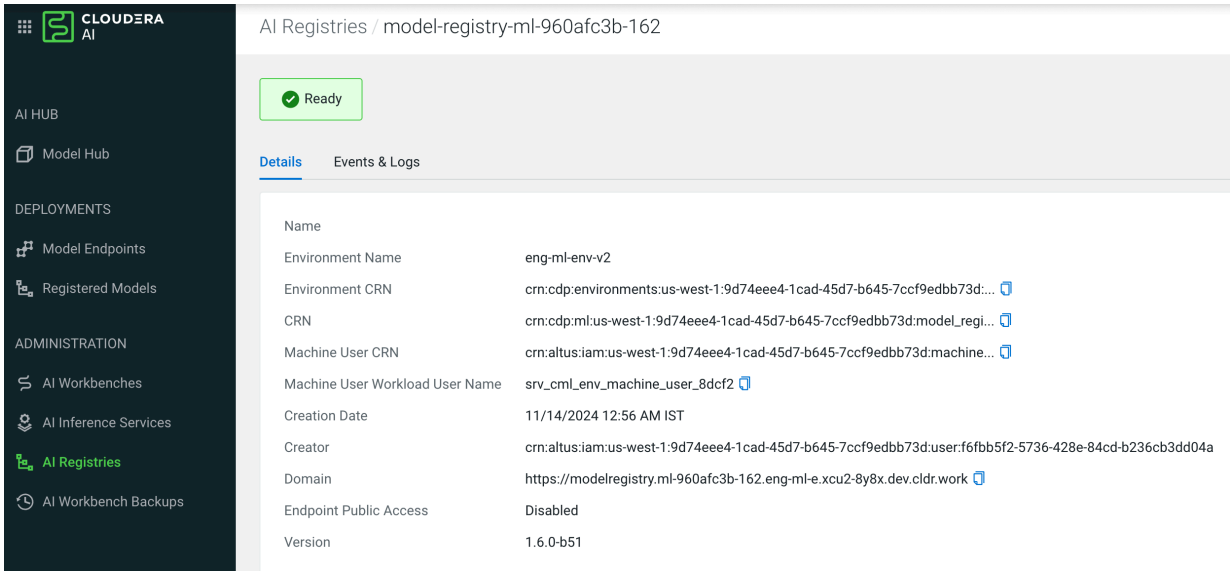

2. Copy the entire Machine User CRN mapping.
3. Navigate to the Environment Manage access idbroker page and add or choose the Machine User CRN mapping to the Data Access Role field.
4. Click Save and Sync.

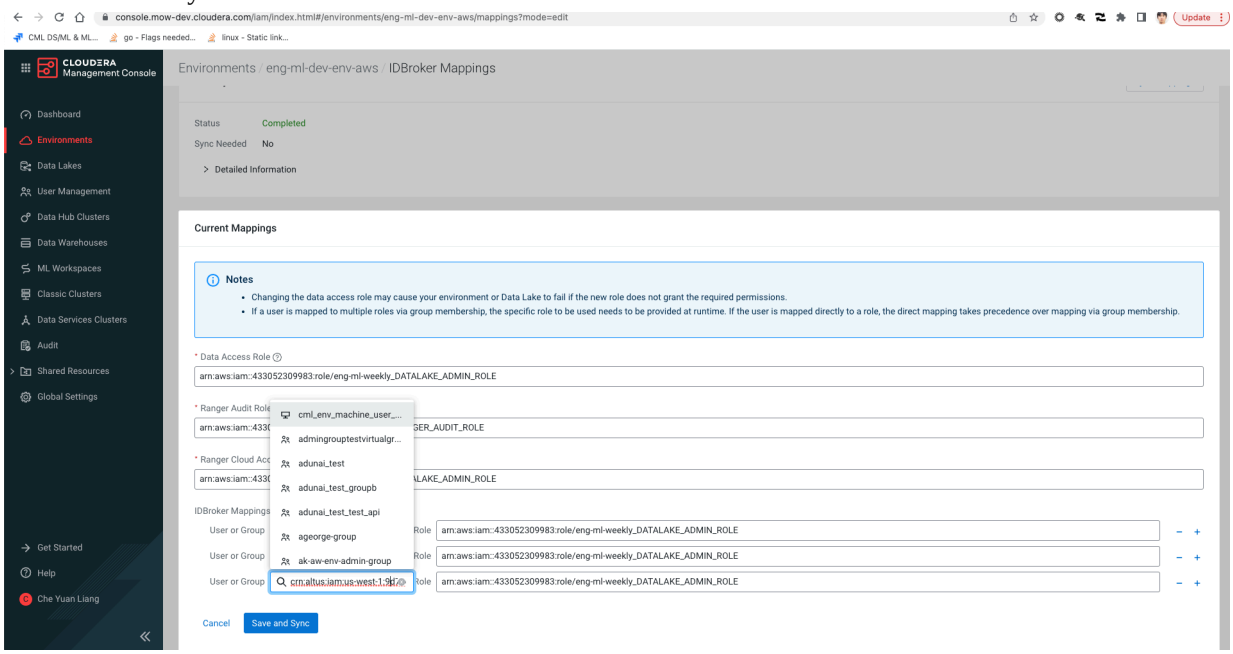

Synchronizing Cloudera AI Registry with a workbench

If you deploy a Cloudera AI Registry in an environment that contains one or more Cloudera AI Workbench, you must synchronize Cloudera AI Registry with the workbenches.

Important: If your Cloudera AI Workbench version is 2.0.46 or higher, or Cloudera AI Inference service version is 1.2.0 or higher, and if you deploy a Cloudera AI Registry in an environment that contains one or more Cloudera AI Workbenches, the Cloudera AI Registry is auto-discovered and periodically synchronized by Cloudera AI Inference service and Cloudera AI Workbenches and no manual synchronization is required.

Cloudera AI Workbench is auto-synchronized every five minutes and Cloudera AI Inference service is auto-synchronized every 30 seconds.

The outbound network access on AWS and Azure must be configured correctly for the auto synchronization to work.

Procedure

1. Click AI Registries to display the AI Registries window.
2. Choose the Cloudera AI Registry you want to synchronize with the workbenches in the environment.
3. From the Actions menu, click Synchronize.

   AI Registries displays the Confirm dialog box listing all of the workbenches in the environment.

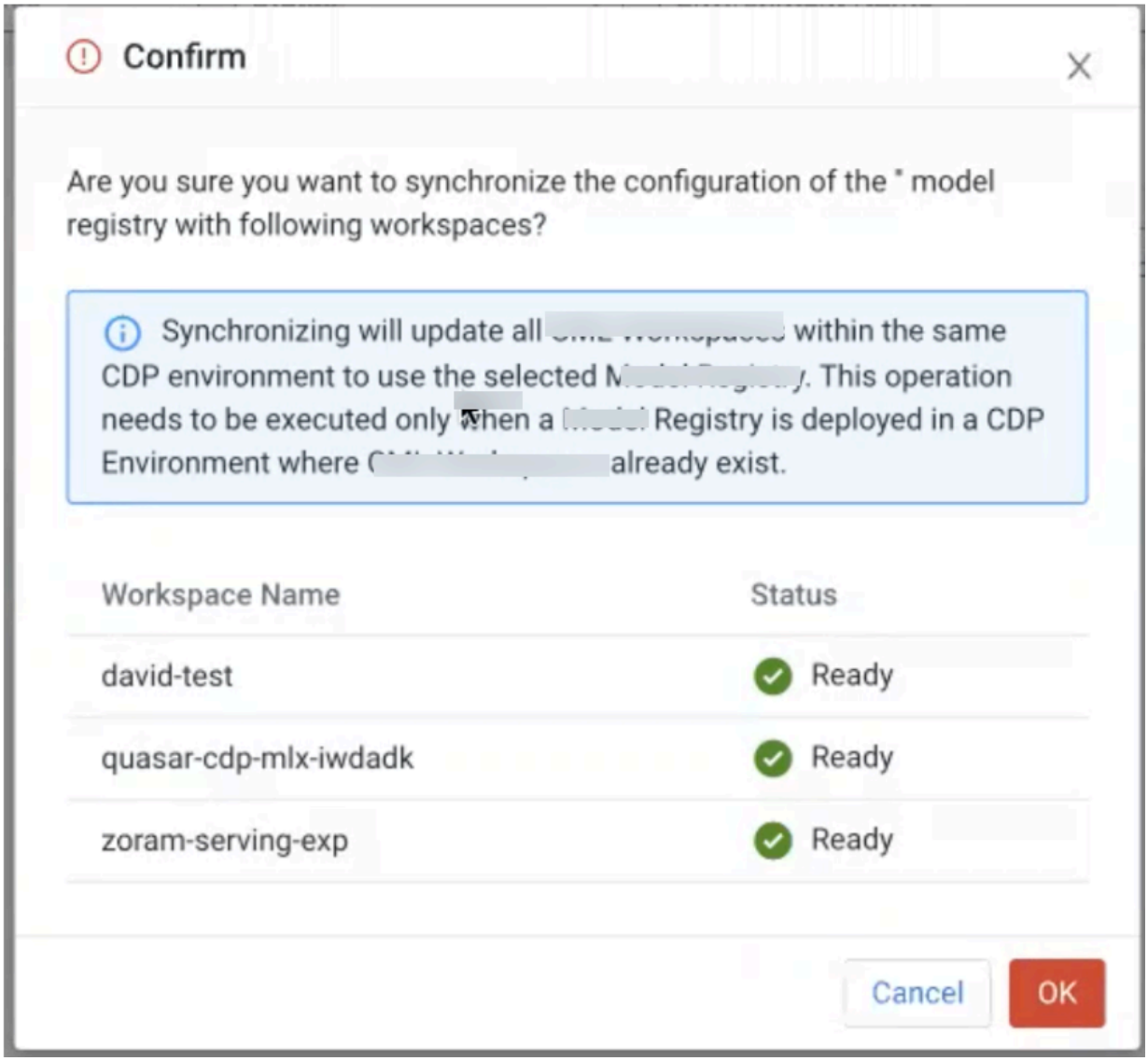

4. Click OK.

Viewing details for Cloudera AI Registries

You can view the information for your registered models using **AI Registries**.

Procedure

1. Select AI Registries from the left navigation pane.

   On the main AI Registries page, you can see all the models currently registered, their respective owners, location of creation, and the last updated time, if known.
2. Use the filter bar at the top of the window to filter the list of AI registries by name, status, and environment name.

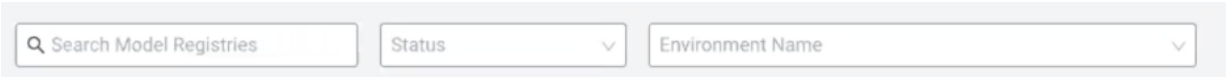

3. Select a AI Registry to see its description.

   Cloudera AI displays the Details page which lists the environment name, environment CRN, CRN, machine user CRN, creator, and creation date.

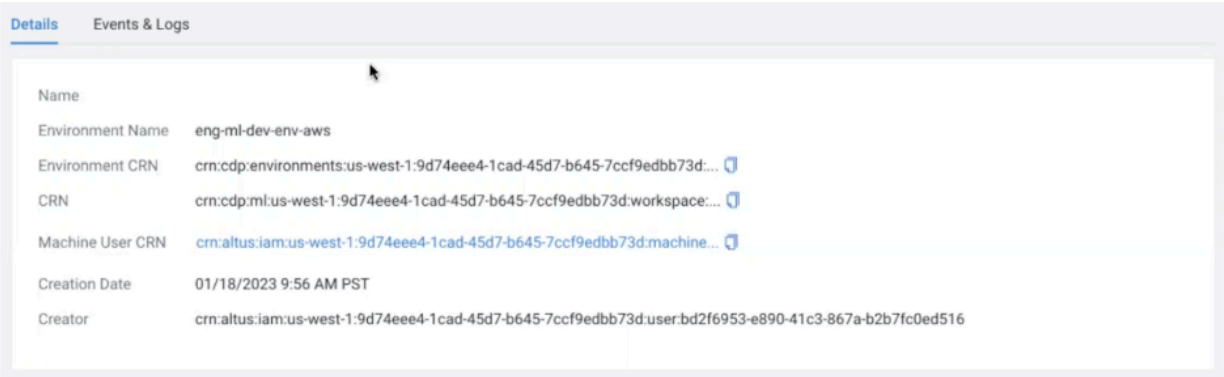

4. You can also click the Events & Logs tab to display information on the events and logs for the Cloudera AI Registry.

Cloudera AI Registry permissions

Cloudera AI Registry's permissions for the following actions are separate from workbench permissions, but they are inherited from environment level workbench permissions.

- create
- delete
- getKubeconfig
- grant/list/revoke access

Therefore, if you have the MLAdmin role on an environment, you can perform these actions for Cloudera AI Registry instances, but an MLUser cannot.

Remote access to a Cloudera AI Registry works similarly to workbench remote access. In addition to AI Registry, but an MLUser cannot.

Remote access to a AI Registry works similarly to workbench remote access. In addition to downloading the `kubeconfig` file, you need to use Grant/List/RevokeModelRegistryAccess endpoints to manage what cloud user identity can access the Kubernetes cluster using your cloud credential.

Model access control

Access to models is dependent on the user permissions, as described here.

- Only administrators are able to see all models, including those not created by them.
- A user, who is an owner of a given model, can set the visibility of the model to public or private on the **AI Registries** UI.

- When a user registers a new model, the user becomes the owner, and no other user has access.
- A user can make the visibility of a model public, and then all users receive viewer access to the model by default.

Deleting Cloudera AI Registry

If you no longer want to access Cloudera AI Registry, you can delete it.

Procedure

1. Select AI Registries to display the AI Registries window.
2. Choose the Cloudera AI Registry you want to delete.
3. From the Actions menu, choose Delete.
4. Click OK.

Force delete a Cloudera AI Registry

If the AI Registry remains in the Cloudera AI Registries list and cannot be deleted, you can force delete the Cloudera AI Registry with this CDP CLI command:

```
cdp ml delete-model-registry --model-registry-crn <AI REGISTRY CRN> --force
```

Registering and deploying models with Cloudera AI Registry

After you have set up Cloudera AI Registry, you can create, register, and deploy models with **AI Registry**.

Creating a model using MLflow

You can use MLflow to create a model.

Using MLflow to create a new model

To create a model using MLflow, see https://docs.cloudera.com/machine-learning/cloud/experiments/topics/ml-exp-v2-mlflow-model-artifact.html.

Registering a model using the AI Registry user interface

You can register a model using the **AI Registry** user interface or the MLFlow SDK.

Using the AI Registry user interface to register a model

Registering a model enables you to track your model and upload and share the model. Registering a model stores the model archives in the Cloudera AI Registry with a version tag. The first time you register a model, **AI Registry** automatically creates a model repository with the first version of the model.

Before you begin

You must have permission to access a project in which the model is created before you can register the model.

Procedure

1. Click Projects in the left navigation pane to display the Projects page.
2. Select the project that contains the model that you want to register.

   AI Registry displays all of the models under the specific project along with their source, deployment status, replicas, memory and a drop-down function for actions that can be made pertaining to that model for deployment.
3. Click the Experiments tab in the left navigation pane and select the experiment that contains the model you want to register.

4. Select the model you want to register.
   Cloudera AI displays the Experiment Run Details page.

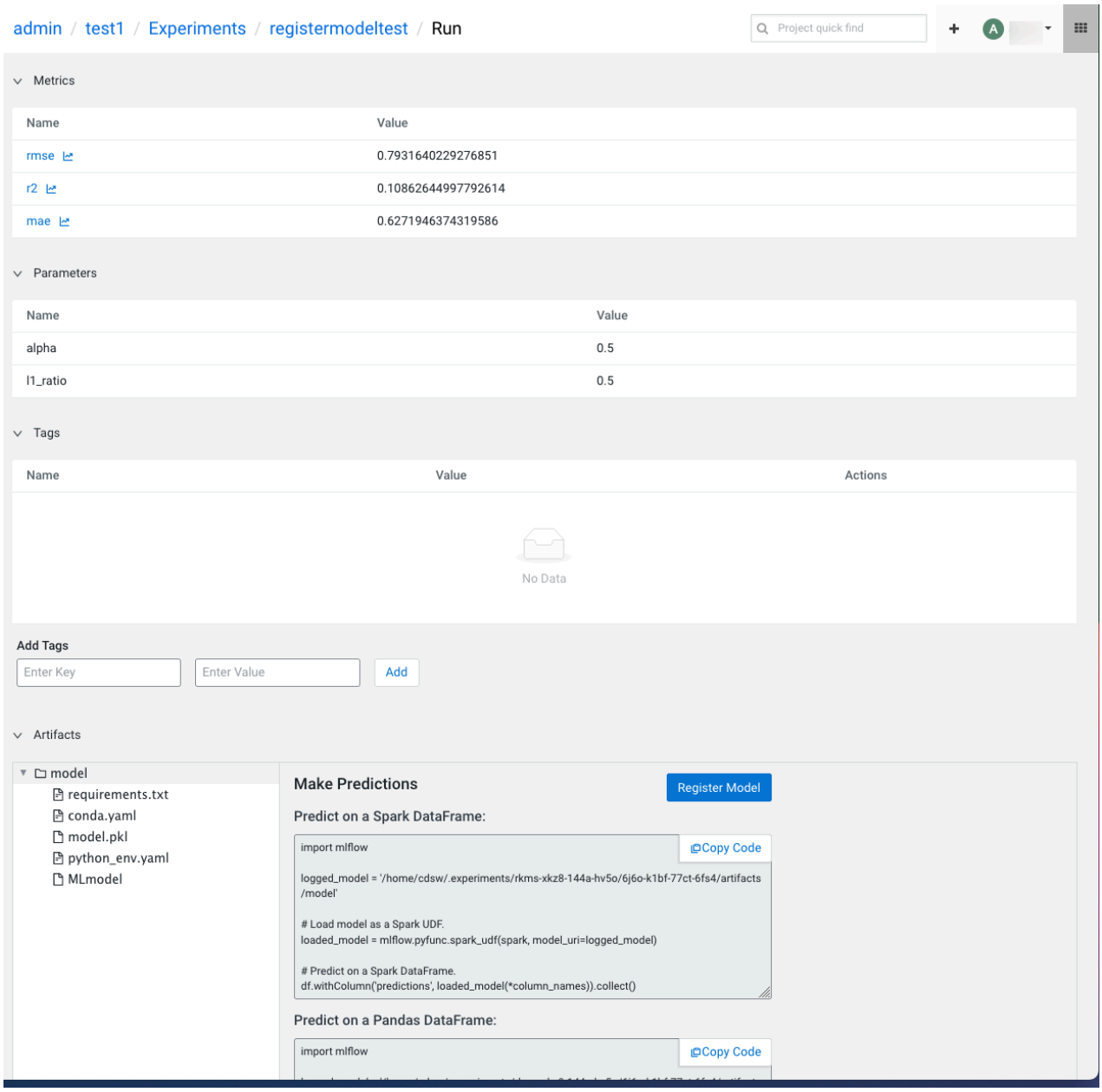

5. Select the run that contains the model you want to register.
6. Select Register Model to begin the registration process.
   AI Registry displays the Registry Model dialog box.
7. Enter the name of your registered model.
   You can also enter optional information for the description, version notes, and version tags.
8. Click OK to complete the registration.

Registering a model using MLflow SDK

You can register a model using the user interface or the MLFlow SDK.

Using MLflow SDK to register a model

Registering a model enables you to track your model and upload and share the model. Registering a model stores the model archives in the Cloudera AI Registry with a version tag. The first time you register a model, Cloudera AI Registry automatically creates a model repository with the first version of the model.

Procedure

1. To register a model using MLFlow SDK, specify the registered_model_name and assign a value:

   ```
   mlflow.<model_flavor>.log_model()
   ```

   For example:

   ```
   mlflow.sklearn.log_model(lr, "model", registered_model_name="ElasticnetW
   ineModel")
   ```

2. If you run the Python code again with the same model_name it will create an additional version for the model_na me.

Using MLflow SDK to register customized models

In MLflow, you can also deploy models that are not directly supported by MLFlow.

To learn more, see Serving LLMs with MLflow: Leveraging Custom PyFunc.

Viewing registered model information

1. From the Projects page in Cloudera AI, select **AI Registry** from the navigation pane. On the main **AI Registry** page, you can see all the models currently registered, their respective owners, location of creation, and the last updated time, if known.
2. Select a registered model to see its description.

   Cloudera AI displays the Details page which outlines the model description, ID, owner, and versions. Different versions of the same model can be deployed in the workbench.

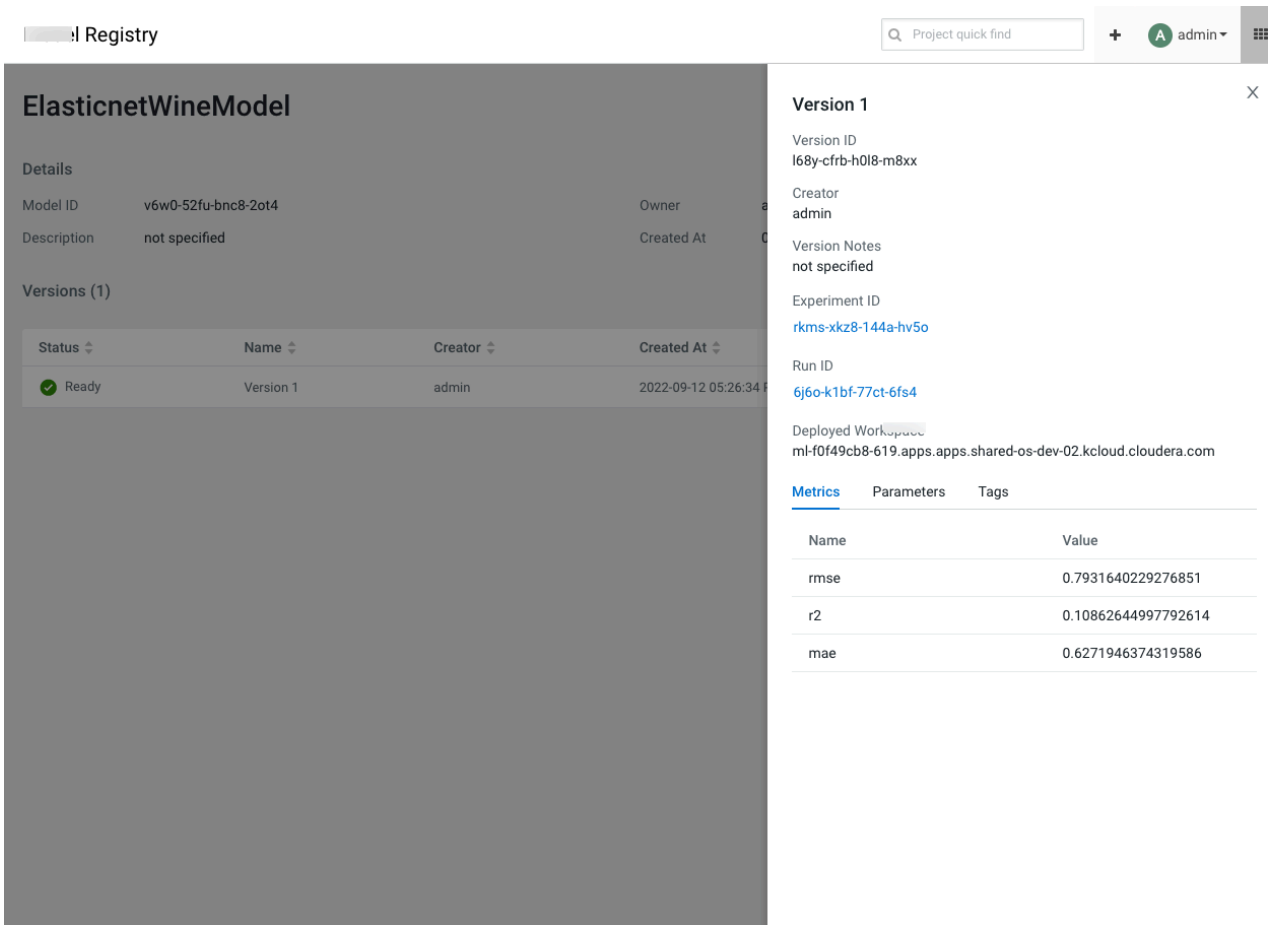

Creating a new version of a registered model

Follow the instructions to create a new version of a registered model.

Procedure

1. Click Projects in the left navigation pane to display the Projects page.
2. Select the project that contains the model for which you want to create a new version.
3. Click Experiments in the left navigation pane and select the experiment that contains the model you want to register.

   The system displays the Experiment Detail page.
4. Select the run that contains the model you want to register.
5. Scroll down the page to find the Artifacts section and click model.
6. Click Register Model.
7. From the Name field, choose the model for which you want to create a new version.
8. Click OK.

What to do next

You can also create a new model version using MLflow SDK. Simply run the Python code to register a model again with the same model_name. This will create an additional version for the model_name.

Deploying a model from the AI Registry page

You can deploy a model once or more times to create different versions of the model. You can also deploy a model you created in one workbench to a different workbench.

Procedure

1. Select **AI Registry** from the left navigation pane.
2. Select the model you want to deploy.
   AI Registry displays the Model Version List page.
3. Select the model version you want to deploy.
   AI Registry displays a side window that lists the version information. Dismiss this window to proceed.
4. Under the Actions menu, click Deploy.
5. Select the Project you want to deploy to in the dialog box and click Go.
   You can select either the project the model is located in or another project to deploy the model to.
   AI Registry displays the Deploy a Model page with the detailed model information auto populated.

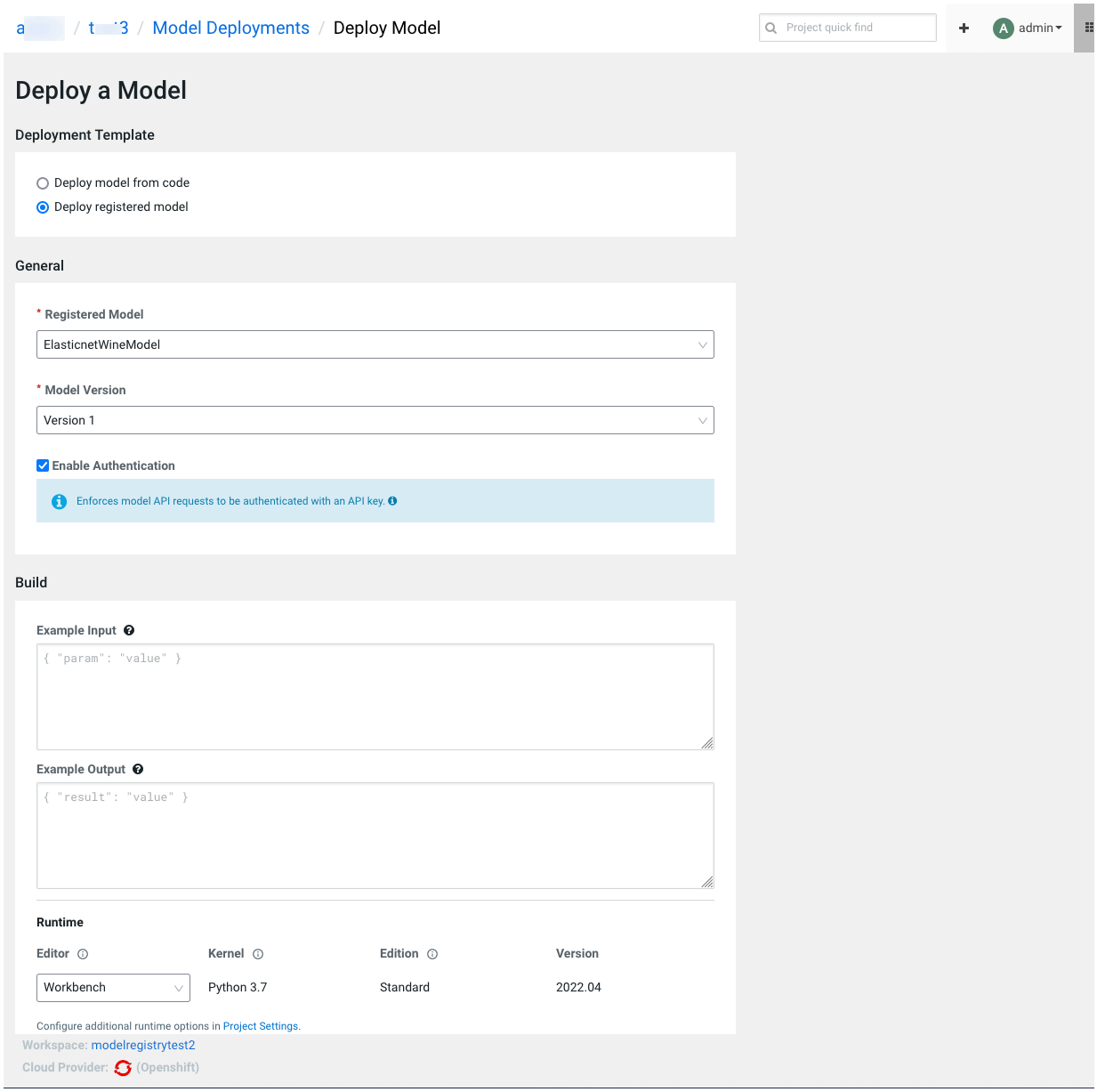

6. If you enable authentication, you will need to enter an API key to access and use the model in the case you have deployed the model to a shared project.
7. Click OK.

Deploying a model from the Cloudera AI Registry using APIv2

You can use the API v2 to deploy registered models from the **AI Registry** as part of your MLOps CI/CD pipeline.

The following example code shows how to deploy a model from the **AI Registry** by using three APIv2 calls: create a model, create a model build, and create a model deployment.

```
api_client = cmlapi.default_client()

model_body = cmlapi.CreateModelRequest(
  project_id=project_id,
  name="foo",                              # replace this with the model
 name
  description="Foo",
  disable_authentication=True,
  registered_model_id="xyo2-ohbr-w0n2-dx3s" # replace this with the register
ed model id
)

model = api_client.create_model(model_body, project_id)
print(model)

model_build_body = cmlapi.CreateModelBuildRequest(
  project_id=project_id,
  model_id=model.id,
  kernel="python3",
  runtime_identifier='docker.repository.cloudera.com/cloudera/cdsw/ml-run
time-pbj-workbench-python3.10-standard:2023.12.1-b8', # replace this with th
e runtime identifier
  registered_model_version_id="ar0a-z7sd-pjgb-2fn2" # replace this with the
 registered model id
)

model_build = api_client.create_model_build(model_build_body, project_id, mo
del.id)
print(model_build)

while model_build.status not in ["built", "build failed"]:
    print("waiting for model to build...")
    time.sleep(10)
    model_build = api_client.get_model_build(project_id, model.id, model_b
uild.id)
if model_build.status == "build failed":
    print("model build failed, see UI for more information")
    sys.exit(1)
print("model built successfully!")


model_deployment_body = cmlapi.CreateModelDeploymentRequest(project_id=proj
ect_id, model_id=model.id, build_id=model_build.id, replicas = model_replica
s)
model_deployment = api_client.create_model_deployment(model_deployment_bo
dy, project_id, model.id, model_build.id)

while model_deployment.status not in ["stopped", "failed", "deployed"]:
    print("waiting for model to deploy...")
    time.sleep(10)
    model_deployment = api_client.get_model_deployment(project_id, model.id,
 model_build.id, model_deployment.id)
if model_deployment.status != "deployed":
    print("model deployment failed, see UI for more information")
    sys.exit(1)
print("model deployed successfully!")
```

Deploying a model from the destination Project page

You can deploy a model one or more times to create different versions of the model. You can also deploy a model you created in one workbench to a different workbench.

Procedure

1. Navigate to the Project you want to deploy to.
2. Click Model Deployment in the left navigation pane.
3. Make sure you have clicked the Deploy registered model checkbox at the top of the window.
4. Select the registered model you want to deploy from the Deploy Registered Model field.
5. If you enable authentication, you will need to enter an API key to access and use the model in the case you have deployed the model to a shared project.
6. Select Deploy Model at the bottom of the window.

Viewing Details for Cloudera AI Registry

You can view detailed information for Cloudera AI Registry.

Procedure

1. Select AI Registry from the left navigation pane.

   On the main AI Registry page, you can see all the models currently registered, their environment name, respective owners, location of creation, and the last updated time, if known.
2. Choose the Cloudera AI Registry you want to synchronize with the workbenches in the environment.
3. You can use the filter bar at the top of the window to filter the list of model registries by name, status, and environment name.
4. Click OK.

Delete a model from Cloudera AI Registry

You can delete a model from Cloudera AI Registry through the UI or using an API call.

Deleting through the UI

1. In **AI Registry**, find the model to delete.
2. In Actions, select Delete.
3. Click OK to confirm deleting the model.

The model is deleted from the Cloudera AI Registry.

Deleting a model with an API call

You can run API calls in the session workbench to delete a model.

1. Use the first two commands to obtain the model_id:

   ```
   api_client=cmlapi.default_client()
   api_client.list_registered_models()
   ```

   The json output of the command includes an example model_id as shown here:

   ```
   'model_id': '7xwf-e6pl-tb28-iylh',
   ```
2. Insert the model_id (replace the example shown below with your own value) to the following command and run it.

   ```
   api_client.delete_registered_model(model_id='7xwf-e6pl-tb28-iylh')
   ```

The model is deleted from the Cloudera AI Registry.

Disabling Cloudera AI Registry

By default, Cloudera AI Registry is enabled in Cloudera AI. You can disable Cloudera AI Registry if you do not want to use this feature.

Procedure

1. Click Site Administration in the left navigation pane.
2. Click Settings to display the Setting Page.
3. Under the Feature Flags section, uncheck the Enable AI Registry checkbox.

Upgrade Cloudera AI Registry

When you upgrade a Cloudera AI Registry, you create a new version of the registry. You can roll back to the old version, or delete the old registry if it is no longer needed.

The Cloudera AI Registry upgrade process involves provisioning a new cluster, so the process can take some time, on the order of 40 to 60 minutes. The upgrade operation shall be performed during a maintenance window, when new models are not being added or updated in Cloudera AI. The window shall be available approximately 3 hours long at a minimum.

1. In AI Registries, choose a registry to upgrade.
2. Copy the CRN (Creator) shown in the AI Registries.
3. In the Cloudera Data Platform CLI, enter cdp ml upgrade-model-registry —crn <crn>

   In the AI Registries UI, the Status of the AI registry changes to Backup Initiated. The process starts provisioning a new registry (Status: Creating) and then performs a backup operation to restore the state.
4. When the new registry shows Ready, then in Actions, click Synchronize. Check the contents of the new registry to verify that all of the models have transferred.
5. Finally, after verifying the contents of the new registry, you can delete the old registry (check the Created at timestamp) in the Action menu. This action is irreversible.

To verify the data, go to a workbench. Check that all of the models are there. Deleting the old Cloudera AI registry cannot be reversed.

You should also check the event logs to see if any errors occurred during the upgrade. Can also be used real time to see that the process is running. If there is a need for an escalation ticket, make sure to include the logs.

If you upgrade a Cloudera AI registry, then synchronize, then rollback, any models that were added through synchronizing will not be present. The process does not transfer the data backwards to the old registry.

Roll back the registry upgrade

If there is a problem, you can roll back the upgrade using the endpoint: rollback-model-registry-upgrade.

At the command line prompt, enter the following command:

cdp ml rollback-model-upgrade-registry —crn <crn>

For CRN, use the CRN of the registry you upgraded (that is, the same CRN you used in the upgrade command).

For more information, you can check the help document of these commands:

- cdp ml rollback-model-upgrade-registry help
- cdp ml upgrade-model-registry help

Cloudera AI Registry standalone API

You can use the standalone Cloudera AI Registry API to communicate with the Cloudera AI Registry using the REST client or CLI client.

The Cloudera AI Registry standalone API supports the following functionalities:

- GET/PATCH/DELETE for the model and model version
- GET a curated list of NGC models
- Import external model from NVIDIA NGC or HuggingFace to Cloudera AI Registry through the POST method

Currently, the Cloudera AI Registry Standalone API does not support uploading the models through POST method from the local machine.

Cloud Platforms

Cloudera AI Registry API is available only on AWS and Azure.

API definition

The Swagger definition is available in the Cloudera AI API documentation.

Prerequisites for Cloudera AI Registry standalone API

To set up the Cloudera AI Registry standalone API, configure the Cloudera AI Inference service and import pretrained Models.

Prerequisites for Cloudera AI Inference service

Cloudera AI Registry is a prerequisite for Cloudera AI Inference service because the Cloudera AI Inference service needs to deploy the models that are stored in the Cloudera AI Registry.

- To use the Cloudera AI Inference service, the latest Cloudera AI Registry must be present in the same Cloudera Data Platform environment before the Cloudera AI Inference service is created.
- If there is an older Cloudera AI Registry in the environment that is created before May 14, 2024, follow the *Upgrade Cloudera AI Registry* instructions to upgrade the Cloudera AI Registry to the latest version before you create the Cloudera AI Inference service.
- If the Cloudera AI Registry is recreated, upgraded, or cert-renewed while the Cloudera AI Inference service is present, then follow the steps listed in the *Manually updating Cloudera AI Registry configuration* topic to ensure that the configuration of Cloudera AI Registry and Cloudera AI Inference service are synchronized.

Prerequisites to import pretrained models

You must add the URL details to allow them in the firewall rules.

NVIDIA GPU Cloud (NGC)

Add the following URL details so they can be allowed in the firewall's rules.

- prod.otel.kaizen.nvidia.com (NVIDIA open telemetry)
- api.ngc.nvidia.com
- files.ngc.nvidia.com

Hugging Face

Add the following URL details so they can be allowed in the firewall's rules.

- huggingface.co
- cdn-lfs.huggingface.co
- *.cloudfront.net (CDN)

Note: If required, you must allow more URLs based on your requirements.

Authenticating clients for interacting with Cloudera AI Registry API

Clients that interact with the Cloudera AI Registry Standalone API and with model endpoints must obtain a JSON Web Token (JWT) from the Cloudera Data Platform control plane, which must be passed as a Bearer token in HTTP requests sent to the serving API and endpoints.

To obtain JWT, run the following Cloudera Data Platform CLI command:

```
$ CDP_TOKEN=$(cdp iam generate-workload-auth-token --workload-name DE | jq -
r
   '.token')
```

In this comment, *DE* is the workload name.

Then pass CDP_TOKEN in the HTTP request header as follows

```
$ curl -H "Authorization: Bearer ${CDP_TOKEN}" <URL>
```

The token obtained using this method expires in one hour.

Role-based authorization

Cloudera AI Registry implements role-based access control.

Users must have the following roles to create an instance of the service in a Cloudera Data Platform environment:

- EnvironmentAdmin
- MLAdmin (admin user)

Registered Models can be viewed, created, deleted, and modified by users having EnvironmentUser role along with either one of the following roles:

- MLAdmin (admin user)
- MLUser

For more information about the access control for the registered models, see *Model access control*.

Related Information

Model access control

Using the REST Client

You need the domain information to use the REST client to interact with the registry.

Before you begin

To obtain the domain information, perform the following:

1. In the **Cloudera** console, click the **Cloudera AI** tile.
2. Click AI Registries in the left navigation menu. The AI Registries page displays.

3. Click on the name of the Cloudera AI Registry to display the Cloudera AI Registry information. The Domain name is displayed in the Details tab.

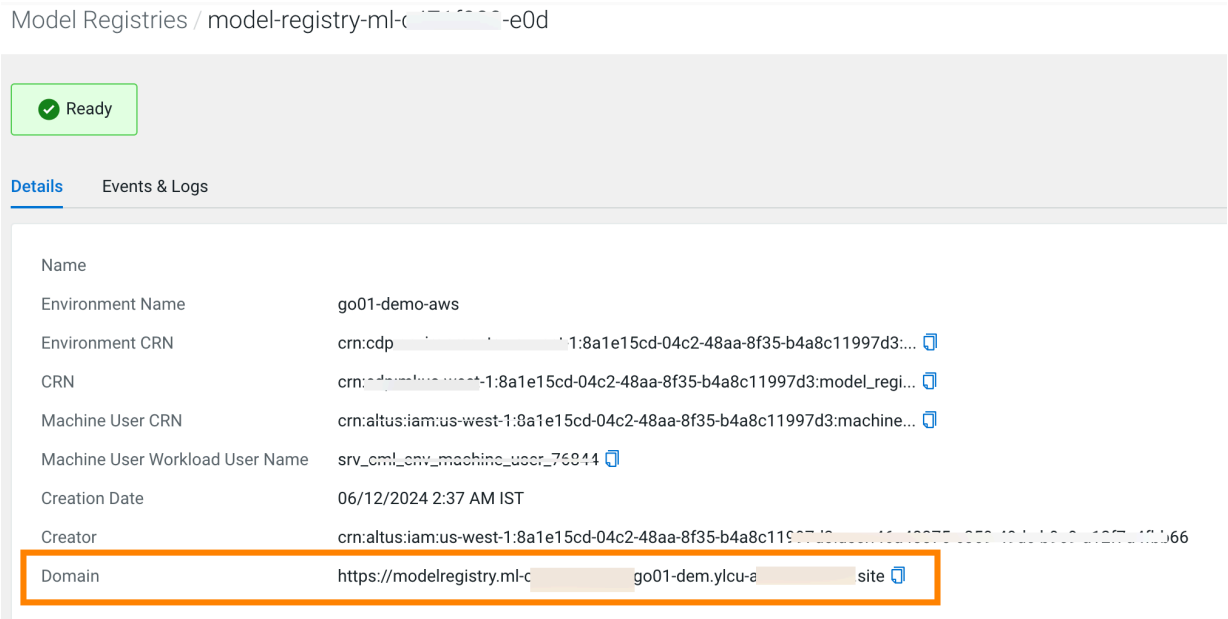

Get all Models

```
curl -s -H "Authorization: Bearer ${CDP_TOKEN}" ${DOMAIN}/api/v2/models | jq
        {
        "models": [
        {
        "created_at": "2024-04-18T15:54:15.543Z",
        "creator": {
        "user_name": "csso_cheyuanl"
        },
        "id": "5bwt-qqe2-e1vg-chqj",
        "name": "foo",
        "tags": null,
        "updated_at": "2024-04-18T15:54:15.543Z",
        "visibility": "private"
        },
```

Get a Model

```
curl -s -H "Authorization: Bearer ${CDP_TOKEN}" ${DOMAIN}/api/v2/models/fx0k
-baf7-yszl-jrt2 | jq
          {
          "created_at": "2024-04-18T15:54:24.940Z",
          "creator": {
          "user_name": "csso_cheyuanl"
          },
          "id": "fx0k-baf7-yszl-jrt2",
          "model_versions": [
          {
          "artifact_uri": "abfs://data@engmldevenvazuresan.dfs.core.windows
.net/modelregistry/fx0k-baf7-yszl-jrt2/y8d8-qluc-00md-h2pw/model.tar.gz",
          "created_at": "2024-04-18T15:54:24.942Z",
          "model_id": "fx0k-baf7-yszl-jrt2",
          "status": "READY",
          "tags": null,
          "updated_at": "2024-04-18T15:54:24.942Z",
```

```
          "user": {
          "user_name": "csso_cheyuanl"
          },
          "version": 1
          }
          ],
          "name": "foo2",
          "tags": null,
          "updated_at": "2024-04-18T15:54:24.940Z",
          "visibility": "private"
          }
```

Get a Model version

```
curl -s -H "Authorization: Bearer ${CDP_TOKEN}" ${DOMAIN}/api/v2/models/fx0k
-baf7-yszl-jrt2/versions/1 | jq
          {
          "artifact_uri": "abfs://data@engmldevenvazuresan.dfs.core.windows.
net/modelregistry/fx0k-baf7-yszl-jrt2/y8d8-qluc-00md-h2pw/model.tar.gz",
          "created_at": "2024-04-18T15:54:24.942Z",
          "model_id": "fx0k-baf7-yszl-jrt2",
          "status": "READY",
          "tags": null,
          "updated_at": "2024-04-18T15:54:24.942Z",
          "user": {
          "user_name": "csso_cheyuanl"
          },
          "version": 1
          }
```

Patch a Model

```
curl -XPATCH -s -H "Authorization: Bearer ${CDP_TOKEN}" ${DOMAIN}/api/v2/mod
els/fx0k-baf7-yszl-jrt2 -d ‘{
  “visibility”: “public”
}’
```

Patch a Model Version

```
curl -XPATCH -s -H "Authorization: Bearer ${CDP_TOKEN}" ${DOMAIN}/api/v2/mod
els/fx0k-baf7-yszl-jrt2/version/1 -d ‘{
  “tags”: [{“key”: “k1”, “value”: “v1”}, “key”: “k2”, “value”: “v2”}]
}’
```

Delete a Model

```
curl -XDELETE -s -H "Authorization: Bearer ${CDP_TOKEN}" ${DOMAIN}/api/v2/mo
dels/vuu6-gcfx-ydio-rit0
```

Delete a Model version

```
curl -XDELETE -s -H "Authorization: Bearer ${CDP_TOKEN}" ${DOMAIN}/api/v2/mo
dels/vuu6-gcfx-ydio-rit0/versions/1
```

Cloudera AI Registry CLI Client

Cloudera AI Registry client is a command line tool (CLI) that can be used to interact to a Cloudera AI Registry server. It can be downloaded from any Cloudera AI Registry server <domain>/apiv2/cli/<os>. Here, <os> is either Linux, Darwin for Mac, or Windows based on the operating system the Cloudera AI Registry CLI is installed on.

The swagger CLI is downloaded from `https://<domain>/apiv2/cli/<os>`. The following are some of the example usage of CLI.

Usage

After you download the CLI and add it to the path, you can use the `modelregistrycli` commands.

```
modelregistrycli help
Usage:
  modelregistrycli [command]

Available Commands:
  completion  Generate completion script
  help        Help about any command
  operations
Flags:
      --Authorization string
      --config string          config file path
      --debug                  output debug logs
      --dry-run                do not send the request to server
  -h, --help                   help for modelregistrycli
      --hostname string        hostname of the service (default "localhost")
      --scheme string          Choose from: [http] (default "http")
```

Create a Model

Create an imported model request

```
$ modelregistrycli --Authorization "Bearer nil" --hostname localhost:8188 op
erations CreateModel --body '{
    "name": "tiny",
    "createModelVersionRequestPayload": {
        "metadata": {
            "model_repo_type": "HF"
        },
        "downloadModelRepoRequest": {
            "source": "HF",
            "repo_id": "prajjwal1/bert-tiny"
        }
    }
}'
***Output:***

{"created_at":"2024-04-03T18:02:15.331Z","creator":{"user_name":"admin"},
"description":"model to classify catAndDogClassifier","id":"1w6s-8m6t-ngdr-3
qvr","model_versions":null,"name":"catAndDogClassifier","tags":[{"key":"cat"
,"value":"1"}],"updated_at":"2024-04-03T18:02:15.331Z","visibility":"public"
}
```

Get Models

```
$ modelregistrycli --Authorization "Bearer nil" --hostname localhost:8188 op
erations GetModels

***output***:
```

```
{"models":[{"created_at":"2024-04-03T18:02:15.331Z","creator":{"user_name
":"admin"},"description":"model to classify catAndDogClassifier","id":"1w6s-
8m6t-ngdr-3qvr","model_versions":null,"name":"catAndDogClassifier","tags":nu
ll,"updated_at":"2024-04-03T18:02:15.331Z","visibility":"public"},{"created_
at":"2024-04-03T18:08:43.130Z","creator":{"user_name":"admin"},"description"
:"create request model request with model version example","id":"8fts-rgpn-r
9xo-xlh0","model_versions":null,"name":"chain-classifier","tags":null,"updat
ed_at":"2024-04-03T18:08:43.130Z","visibility":"public"}]}
```

Get Model by ID

```
$ modelregistrycli --Authorization "Bearer nil" --hostname localhost:8188 op
erations GetModel --model_id '8fts-rgpn-r9xo-xlh0'
{"created_at":"2024-04-03T18:08:43.130Z","creator":{"user_name":"admin"},
"description":"create request model request with model version example","id"
:"8fts-rgpn-r9xo-xlh0","model_versions":[{"artifact_uri":"http://localhost:9
000/8fts-rgpn-r9xo-xlh0/r5eg-m0gp-i4qs-8b07/model.tar.gz","created_at":"2024
-04-03T18:08:43.132Z","model_id":"8fts-rgpn-r9xo-xlh0","notes":"create reque
st model request with model version example","status":"REGISTERING","tags":[
{"key":"chain","value":"2"}],"updated_at":"2024-04-03T18:08:43.132Z","user":
{"user_name":"admin"},"version":1}],"name":"chain-classifier","tags":null,"u
pdated_at":"2024-04-03T18:08:43.130Z","visibility":"public"}
```

Known issues with Cloudera AI Registry standalone API

These are some of the known issues you might run into while using Cloudera AI Registry standlone API.

NGC model download timeout

The NGC model import might time out, and the corresponding model version status is shown as “failed”. You can access the logs found in the API v2 pod by performing the steps mentioned in the *Debugging the model import failure* troubleshooting section.

```
2024/04/23 16:53:45 Error download model repo:  ohlfw0olaadg/ea-participants/llama-2-7b-chat:LLAMA-2-
7B-CHAT-4K-FP16-1-A100.24.01
2024/04/23 16:53:45 Error:  exit status 1
2024/04/23 16:53:45 Command output:  Connection failed; retrying... (Retries left: 5)
Connection failed; retrying... (Retries left: 4)
Connection failed; retrying... (Retries left: 3)
Connection failed; retrying... (Retries left: 2)
Connection failed; retrying... (Retries left: 1)
Error: Request timed out.
CLI_VERSION: Latest - 3.41.2 available (current: 3.41.1). Please update by using the command 'ngc ver
sion upgrade'


2024/04/23 16:53:45 Failed to download NGC model repo to local folder:  exit status 1
```

Retry the model import request again.

Cloudera AI Inference service is unable to discover the Cloudera AI Registry

In certain cases, the Cloudera AI Inference service is unable to discover the Cloudera AI Registry.

After upgrading Cloudera AI Registry, follow the *Manually Updating Cloudera AI Registry Configuration* steps listed in the Troubleshooting section or delete and create model-registry to ensure that Cloudera AI Inference service continues to work.

Model import failure

You can download the models concurrently only if their combined size is below approximately 400 GB. Exceeding this limit may result in import failures and unexpected behavior.

Request Throttling

Currently, there is no request throttling mechanism implemented. As a result, excessive concurrent requests may lead to model import failures. To minimize the risk, it is recommended to limit concurrent requests to a maximum of 5, which is considered a safe threshold.

Model Import progress indicator

A progress bar is not available for model imports. For reference, importing a 70 GB model typically takes approximately 1 hour. Users should plan accordingly and monitor the process through alternative options, if necessary.

Lack of Model Import failure details

Currently, the UI and API does not provide specific reasons for model import failures. You have to use Kubernetes logs to diagnose the issues.

Related Information

Debugging the model import failure

Manually Updating Cloudera AI Registry Configuration

Updating Cloudera AI Registry configuration

After the Cloudera AI Registry cluster is created, the following patch needs to be applied to the Kubernetes cluster for the standalone API authentication to work.

```
kubectl get configmap knox -n knox -o yaml | sed 's|https://consoleauth.cdp.
cloudera.com/oauth2/keys|https://consoleauth.altus.cloudera.com/oauth2/keys|
g' | kubectl apply -f -
```

For information on how to get the kubeconfig for the Cloudera AI Registry cluster, which is similar to the Cloudera AI Workbench Cluster, see *Manage remote access*.

Related Information

Manage remote access

Troubleshooting issues with Cloudera AI Registry API

Learn about some of the recommended series of steps to perform when troubleshooting issues related to the Cloudera AI Registry API.

Cloudera AI Inference service cannot discover Cloudera AI Registry

Learn about scenarios or issues that may be resolved by using the steps in the *Manually updating Cloudera AI Registry configuration* solution.

- ML Serving installed before Cloudera AI Registry

  If the Cloudera AI Registry has not been created yet, when ML Serving is installed it will not be able to generate the ConfigMap.

- Cloudera AI Registry upgraded after ML Serving installed

  When the Cloudera AI Registry is upgraded, it will require an updated ConfigMap.

- Cloudera AI Registry certificate expired

  After 90 days, Cloudera AI Registry will have a certificate update, it will require an updated ConfigMap.

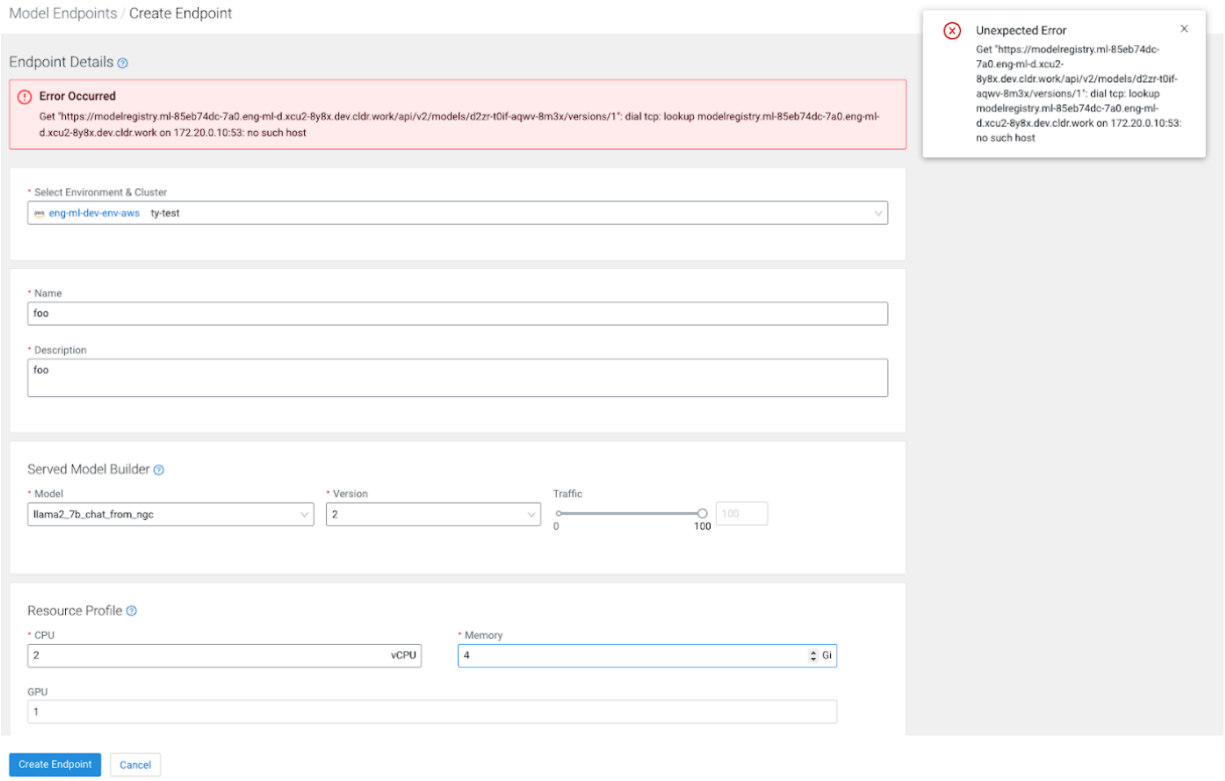

Manually updating Cloudera AI Registry configuration

If you upgrade the Cloudera AI Registry after creating your Cloudera AI Inference service cluster, the Cloudera AI Registry configuration stored by the Cloudera AI Inference service will get out of synchronization. Follow the steps below to manually reconcile the configuration, so that Cloudera AI Inference service will be able to connect to Cloudera AI Registry.

Important: You are required to access multiple clusters when performing the below steps. For information configuring and switching KubeConfig between multiple clusters, see Configure Access to Multiple Clusters

1. Get KubeConfig for ML Serving Cluster.

   a. In a terminal where Cloudera Data Platform CLI is installed, run the cdp ml list-ml-serving-apps command.

   b. Find the appCrn of the ML Serving instance you wish to update.

```
~/go/mlx-crud-app git:(master) 01:40 pm (1.452s)
cdp ml list-ml-serving-apps
2024-04-23 13:40:10,867 - MainThread - cdpcli.clidriver - WARNING - You are running an INTERNAL release of the CDP CLI, which has
 different capabilities from the standard public release. Find the public release at: https://pypi.org/project/cdpcli/
{
    "apps": [
        {
            "appName": "cml-systest-aws",
            "appCrn": "crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:mlserving:2c982e74-c60e-43b9-be85-357cffb1d419",
            "environmentCrn": "crn:cdp:environments:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:environment:3715dcbe-48a8-4292
-94a0-f34e44c3bf35",
```

   c. Retrieve the KubeConfig with this command:

```
cdp ml get-ml-serving-app-kubeconfig --app-crn <YOUR-APP-CRN> |jq .kubec
onfig|yq --prettyPrint
```

   Here:

   - `cdp ml get-ml-serving-app-kubeconfig --app-crn <YOUR-APP-CRN>` : returns raw json response containing KubeConfig.
   - `|jq .kubeconfig:` returns only the actual KubeConfig (removes json wrapper).
   - `|yq --prettyPrint`: prints the kKubeConfig response into a format both machine and user-readable.

```
~/go/mlx-crud-app git:(master) 02:58 pm (1.387s)
cdp ml get-ml-serving-app-kubeconfig --app-crn crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:mlserving:2c982e74-c60e-
43b9-be85-357cffb1d419|jq .kubeconfig|yq --prettyPrint
2024-04-23 14:58:31,684 - MainThread - cdpcli.clidriver - WARNING - You are running an INTERNAL release of the CDP CLI, which has
 different capabilities from the standard public release. Find the public release at: https://pypi.org/project/cdpcli/
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJQXZzUm0rYys1eTB3RFFZSktvWklodmNO
VFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6QWVGdzB5TkRBME1qTXhOak0zTkRsYUZ3MHpOREEwTWpFeE5qUXlORGxhTUJVeApFekFSQmdOVkJBT
RDbXQxWW1WeWJtVjBaWE13Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLCkFvSUJBUURCWElpTE95TVI4L01yaGZ2SkhNa05ua3dJRzdXTmR5ZzhPb2
UZDBxQkh6SEtMQzBSdFNPYjR3b1IKZTNDU2w0UEdlaUROMFY3ZEFMZ0JVcm4zZmpIeHhjWUZpZzVoWWk2cHV2ZTU5MGJNZFJXUXpYSW1TZ244cHJoZApJK3lWbEVCMkc
MVVWYUhHaXBzdDQ2dXNTUE1QdGJ2VWJHcHRxU1JDU3lzK0hJQkJKUE0zdWhtcjFXVkhTQzZFCmh5ZWZDR0dvNjhGTCtRc2JiekR0cTFaOWV2V25lbE1XeUcrd1lHQzNm
05pc3EzVVVNdU9sY0thV1hBY2ljM3gKcDA2ajlUbGVZVkxwdE5Ob2swVnovcUxkOGsrZENuY2ExR1FBWGhTNGNJayt5UERhcmNxemNRRlJNWEU5b21raQp6N1NjUWhoU
JrdHArVU1QaWRsUUFxNzQ3ZHovQWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRRUF3SUNwREFQCkJnTlZIUk1CQWY4RUJUQURBUUgvTUIwR0ExVWREZ1FXQkJRY0xRUj
PMlhnOThqZklTczRITHdSWnRWUzRUQVYKQmdOVkhSRUVEakFNZ2dwcmRXSmxjbTVsZEdWek1BMEdDU3FHU0liM0RRRUJDd1VBQTRJQkFRQThXZmt3SXE1Zwp0TjAxd2V
azU2MFNZRjdQekdRR2pwWitaOTJUM0JnN1RYTmVRSXYraXd5OFdlZjZTOVpRR0tQaXhvL2Z6aXN4CnRhajk4WGt0NnpUbEhZVEFvK3hoY1FRaDdNZVpQeUFEYzFxVmVM
GlaYkFNOTYxZDkwRDZaSzBkSkQ2QnFHeTIKRTMydUJGWEhPa3IzWDYrNDhldlphYVh3b0RkaEU4QlFMVXVyOE5JeDYvMWt4MUhDSVcrMVpxT2cxc0R3dTk5Ygo2M2o0T
dQaVJoR1ExRVJuUHVob2xjYWs1cXVQK2lDYTdBb29HV2haSTZPUVZ3MjJORVVaSTVDOGNoV3BkREpsCjd0aGFEZzl3b1RISTc0bkdpR2xKNWt2T3JLRjU3MGg0VSs2dn
HTWdiSjd5clQ2TkxkSU5wZURKRTNCTHBNb24KYjBGV2dLN0xqV0w5Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K
    server: https://7FD9E98AABFFCB22A61AE5AF9E1815A0.gr7.us-west-2.eks.amazonaws.com
  name: ml-a66eb62a-196
contexts:
- context:
    cluster: ml-a66eb62a-196
    user: ml-a66eb62a-196
  name: ml-a66eb62a-196
users:
- name: ml-a66eb62a-196
  user:
```

   d. Direct the output to the standard KubeConfig location:

```
cdp ml get-ml-serving-app-kubeconfig --app-crn <YOUR-APP-CRN> |jq .kubec
onfig|yq --prettyPrint>~/.kube/config
```

   Make sure this is the active KubeConfig by running the `kubectl config view` command. The output of the above command shall now show the KubeConfig data returned above.

2. Grant your user ML Serving App access.

   Add your cloud provider user identifier to the ML Serving App access control list. Use the following command, with your ARN, or other cloud provider user identifier.

```
cdp ml grant-ml-serving-app-access --resource-crn crn:cdp:ml:us-west-1:9
d74eee4-1cad-45d7-b645-7ccf9edbb73d:mlserving:2c982e74-c60e-43b9-be85-35
7cffb1d419 --identifier <Your Cloud Provider User ID>
```

3. Get new ConfigMap data.

   a. Find the new Cloudera AI Registry ConfigMap. Use the grant-model-registry-access API in CDP CLI to add your user name to the new Cloudera AI Registry, or use the UI, as shown:

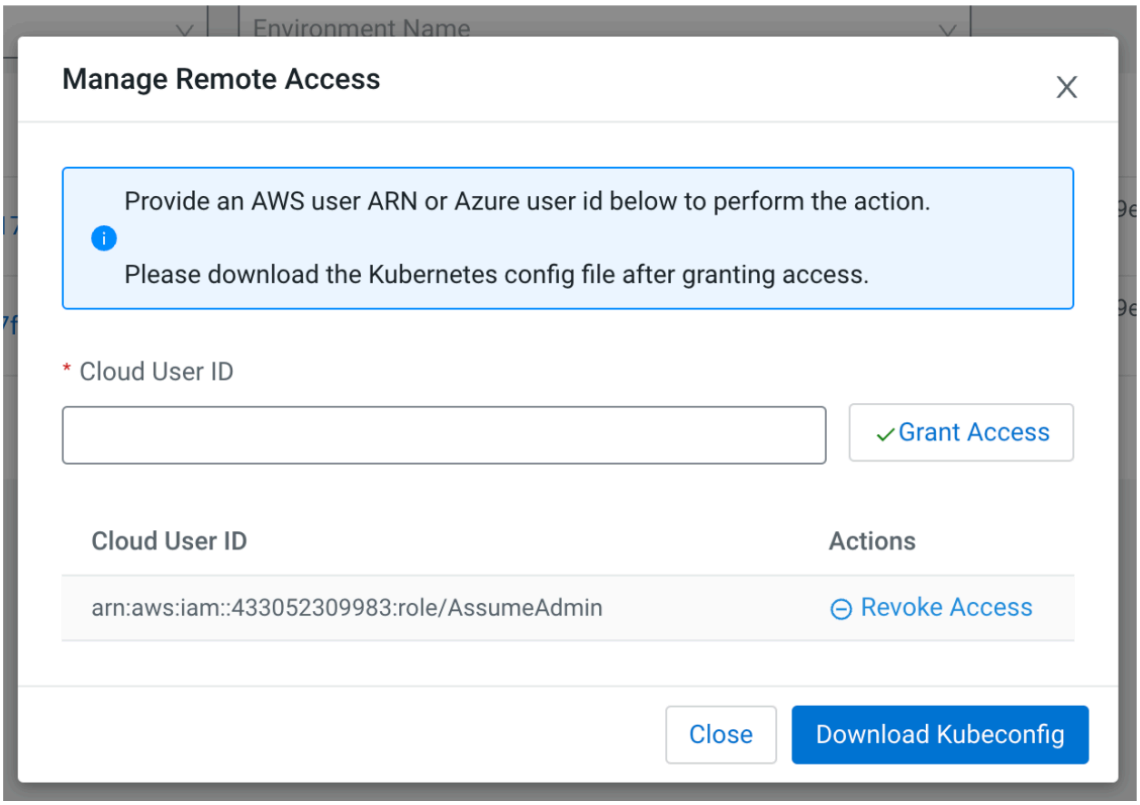

   b. After your user ARN has been granted access to the Cloudera AI Registry, get the ConfigMap data in the following way:

      - Download the Cloudera AI Registry KubeConfig.
      - Set the Cloudera AI Registry KubeConfig (save it to ~/.kube/config).
      - `kubectl get cm -n mlx`: lists all the available ConfigMaps.
      - `kubectl describe cm jwks-rootca -n mlx`: Returns the TLS Certificate for Cloudera AI Registry.
      - `cdp ml list-model-registries`: The response will contain the domain for the updated Cloudera AI Registry.

   c. Copy the entire rootca.pem output from the command.

```
kubectl describe cm jwks-rootca -n mlx
```

```
Data
====
rootca.pem:
----
-----BEGIN CERTIFICATE-----
MIIFmDCCA4CgAwIBAgIQU9C87nMpOIFKYpfvOHFHFDANBgkqhkiG9w0BAQsFADBm
MQswCQYDVQQGEwJVUzEzMDEGA1UEChMqKFNUQUdJTkcpIEludGVybmV0IFNlY3Vy
aXR5IFJlc2VhcmNoIEdyb3VwMSIwIAYDVQQDExkoU1RBR0lORykgUHJldGVuZCBQ
ZWFyIFgxMB4XDTE1MDYwNDExMDQzOFoXDTM1MDYwNDExMDQzOFowZjELMAkGA1UE
BhMCVVMxMzAxBgNVBAoTKihTVEFHSU5HKSBJbnRlcm5ldCBTZWN1cml0eSBSZXNl
YXJjaCBHcm91cDEiMCAGA1UEAxMZKFNUQUdJTkcpIFByZXRlbmQgUGVhciBYMTCC
AiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBALbagEdDTa1QgGBWSYkyMhsc
ZXENOBaVRTMX1hceJENgsL0Ma49D3MilI4KS38mtkmdF6cPWnL++fgehT0FbRHZg
jOEr8UAN4jH6omjrbTD++VZneTsMVaGamQmDdFl5g1gYaigkkmx8OiCO68a4QXg4
wSyn6iDipKP8utsE+x1E28SA75HOYqpdrk4HGxuULvlr03wZGTIf/oRt2/c+dYmD
oaJhge+GOrLAEQByO7+8+vzOwpNAPEx6LW+crEEZ7eBXih6VP19sTGy3yfqK5tPt
TdXXCOQMKAp+gCj/VByhmIr+0iNDC540gtvV303WpcbwnkkLYC0Ft2cYUyHtkstO
fRcRO+K2cZozoSwVPyB8/J9RpcRK3jgnX9lujfwA/pAbP0J2UPQFxmWFRQnFjaq6
rkqbNEBgLy+kFL1NEsRbvFbKrRi5bYy2lNms2NJPZvdNQbT/2dBZKmJqxHkxCuOQ
FjhJQNeO+Njm1Z1iATS/3rts2yZlqXKsxQUzN6vNbD8KnXRMEeOXUYvbV4lqfCf8
mS14WEbSiMy87GB5S9ucSV1XUrlTG5UGcMSZOBcEUpisRPEmQWUOTWIoDQ5FOia/
GI+Ki523r2ruEmbmG37EBSBXdxIdndqrjy+QVAmCebyDx9eVEGOIpn26bW5LKeru
mJxa/CFBaKi4bRvmdJRLAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNVHRMB
Af8EBTADAQH/MB0GA1UdDgQWBBS182Xy/rAKkh/7PH3zRKCsYyXDFDANBgkqhkiG
9w0BAQsFAAOCAgEAncDZNytDbrrVe68UT6py1lfF2h6Tm2p8ro42i87WWyP2LK8Y
nLHC0hvNfWeWmjZQYBQfGC5c7aQRezak+tHLdmrNKHkn5kn+9E9LCjCaEsyIIn2j
qdHlAkepu/C3KnNtVx5tW07e5bvIjJScwkCDbP3akWQixPpRFAsnP+ULx7k0aO1x
qAeaAhQ2rgo1F58hcflgqKTXnpPM02intVfiVVkX5GXpJjK5EoQtLceyGOrkxlM/
sTPq4UrnypmsqSagWV3HcUlYtDinc+nukFk6eR4XkzXBbwKajl0YjztfrCIHOn5Q
CJL6TERVDbM/aAPlv8kJ1sWGLuvvWYzMYaLzDul//rUF10gEMWaXVZV51KpS9DY/
```

The domain of the new Cloudera AI Registry is contained in the list-model-registries response:

```
~ 05:43 pm (1.773s)
cdp ml list-model-registries
2024-04-23 17:43:36,327 - MainThread - cdpcli.clidriver - WARNING - You are running an INTERNAL release of the CDP CLI, which has
 different capabilities from the standard public release. Find the public release at: https://pypi.org/project/cdpcli/
{
    "modelRegistries": [
        {
            "id": 990,
            "creator": "crn:altus:iam:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:user:1f77129b-9cb0-4a23-970f-a5f8af88fdea",
            "status": "installation:finished",
            "environmentCrn": "crn:cdp:environments:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:environment:3715dcbe-48a8-4292
-94a0-f34e44c3bf35",
            "createdAt": "2024-04-23T22:25:58.569000+00:00",
            "crn": "crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:model_registry:dfcff53f-4f0f-4a54-9eaf-4f1cd384387a
",
            "environmentName": "eng-ml-dev-env-aws",
            "workspaceName": "model-registry-ml-17bf05df-050",
            "machineUserCrn": "crn:altus:iam:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:machineUser:cml_env_machine_user_3715
dcbe-48a8-4292-94a0-f34e44c3bf35/5ce78dc4-e84d-498d-a2a0-ebe71d0dae2d",
            "serviceName": "model-registry-ml-17bf05df-050",
            "domain": "https://modelregistry.ml-17bf05df-050.eng-ml-d.xcu2-8y8x.dev.cldr.work"
        },
```

4. Apply ConfigMap Update.
   a. Update the KubeConfig back to the ML Serving KubeConfig.
   b. Edit the ConfigMap of the ML Serving Cluster:
      - kubectl edit cm modelregistry-config-controlplane -n serving: Update tls.crt with the data from above.
      - kubectl describe cm api-config -n serving: Update model.registry.url with the data from above.
   c. Restart the deployment to force the Cloudera Manager changes to take effect.
      - kubectl scale deployment api -n serving --replicas=0
      - kubectl scale deployment api -n serving --replicas=1

Debugging the model import failure

To debug errors that occurred on the Cloudera AI Registry server, you can access the logs found in the API v2 pod.

About this task

Access logs from Cloudera AI Registry Kubernetes cluster.

You can obtain the kubeconfig for the Cloudera AI Registry cluster.

1. In the **Cloudera** console, click the **Cloudera AI** tile.
2. Click AI Registries in the left navigation menu. The AI Registries page displays.
3. In the **Actions** menu, click ⋮ and select **Download Kubeconfig**.

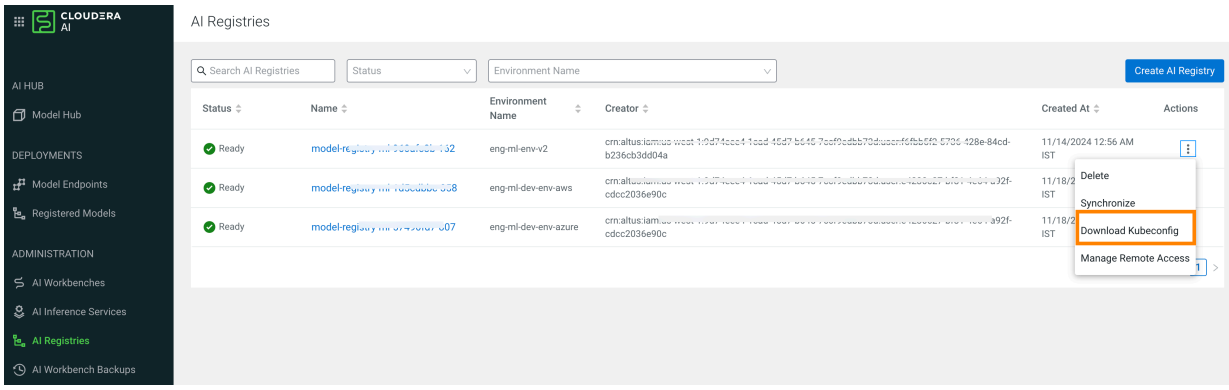

In AWS, you need to add your identity under Manage Remote Access to access the Kubernetes cluster.

You must add your identity under Manage Remote Access. For information on granting remote access, see *Granting Remote Access to Cloudera AI Workbench*. After the kubeconfig is set up, run the following `kubectl` command to get logs for the Cloudera AI Registry pod:

```
kubectl logs <AI registry pod name> -n mlx
```

Related Information

Granting Remote Access to Cloudera AI Workbenches

Creating and deploying a Model

Using Cloudera AI, you can create any function within a script and deploy it to a REST API. In a Cloudera AI project, this is typically a predict function that accepts an input and returns a prediction based on the model's parameters.

Using Cloudera AI, you can create any function within a script and deploy it to a REST API. In a machine learning project, this will typically be a predict function that will accept an input and return a prediction based on the model's parameters.

For the purpose of this quick start demo we are going to create a very simple function that adds two numbers and deploy it as a model that returns the sum of the numbers. This function will accept two numbers in JSON format as input and return the sum.

For Cloudera AI UI

1. Create a new project. Note that models are always created within the context of a project.
2. Click New Session and launch a new Python 3 session.
3. Create a new file within the project called add_numbers.py. This is the file where we define the function that will be called when the model is run. For example:

   add_numbers.py

   ```
   def add(args):
     result = args["a"] + args["b"]
     return result
   ```

   Note: In practice, do not assume that users calling the model will provide input in the correct format or enter good values. Always perform input validation.

4. Before deploying the model, test it by running the add_numbers.py script, and then calling the add function directly from the interactive workbench session. For example:

   ```
   add({"a": 3, "b": 5})
   ```

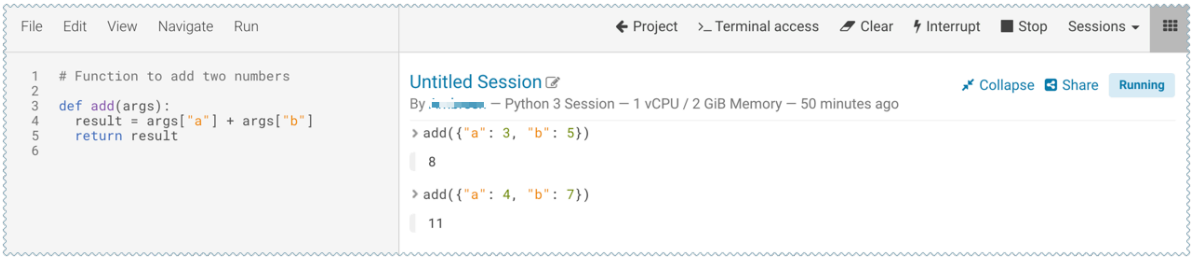

5. Deploy the add function to a REST endpoint.
   a. Go to the project Overview page.
   b. Click Models New Model .
   c. Give the model a Name and Description.
   d. In Deploy Model as, if the model is to be deployed in a service account, select Service Account and choose the account from the dropdown menu.
   e. Enter details about the model that you want to build. In this case:
      - File: add_numbers.py
      - Function: add
      - Example Input: {"a": 3, "b": 5}
      - Example Output: 8

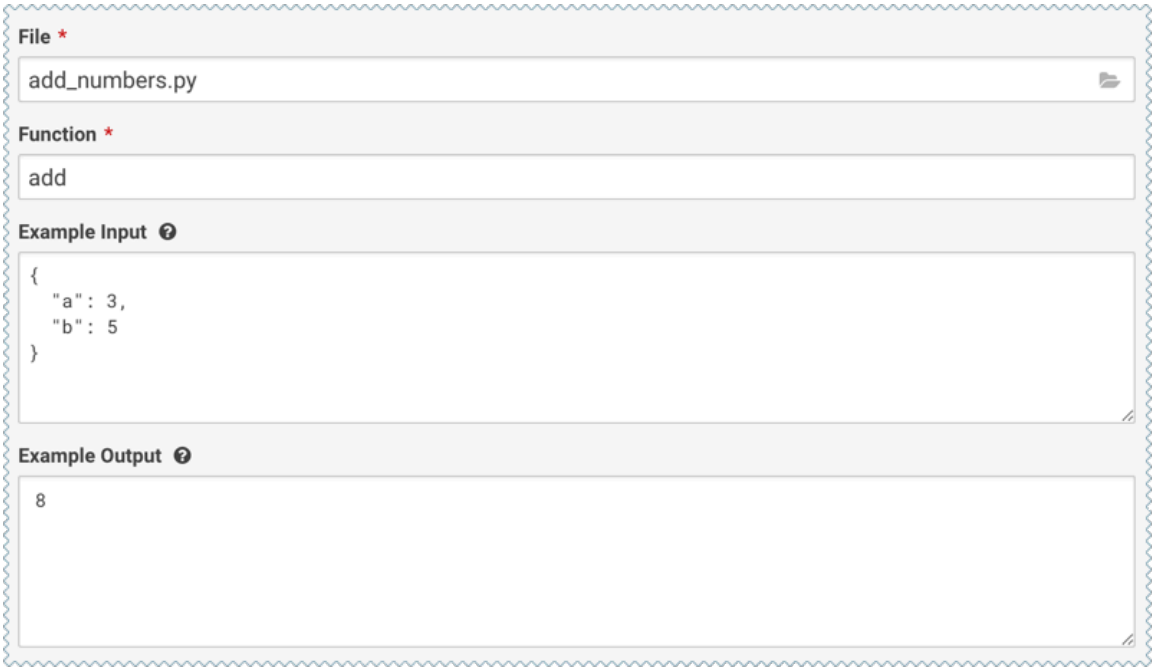

   f. Select the resources needed to run this model, including any replicas for load balancing. To specify the maximum number of replicas in a model deployment, go to Site Administration Settings Models . The default is 9 replicas, and up to 199 can be set.

      Note: The list of options here is specific to the default engine you have specified in your Project Settings: ML Runtimes or Legacy Engines. Engines allow kernel selection, while ML Runtimes allow Editor, Kernel, Variant, and Version selection. Resource Profile list is applicable for both ML Runtimes and Legacy Engines.
   g. Click Deploy Model.

6. Click on the model to go to its Overview page. Click Builds to track realtime progress as the model is built and deployed. This process essentially creates a Docker container where the model will live and serve requests.

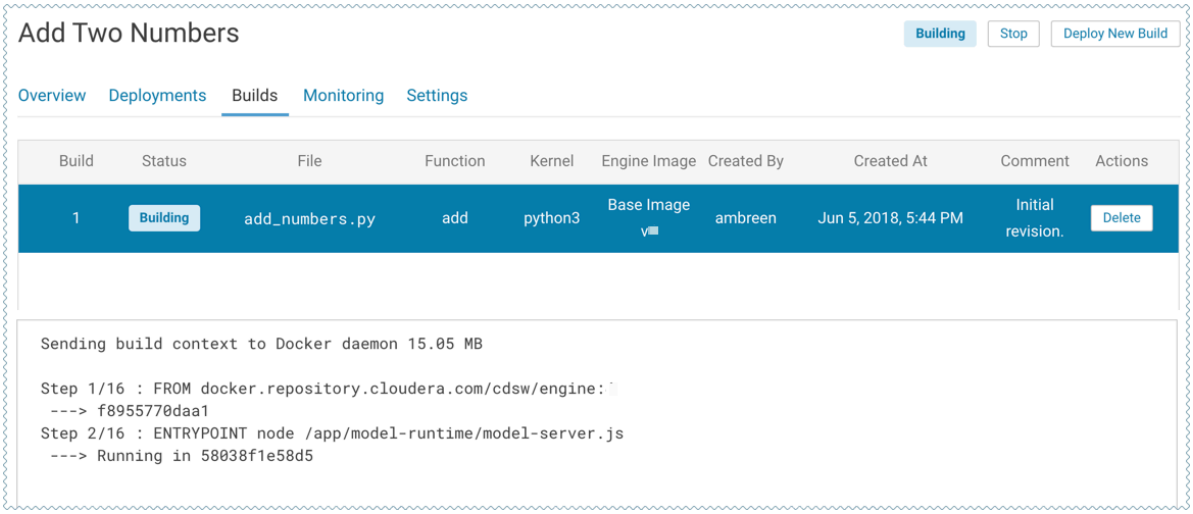

7. Once the model has been deployed, go back to the model Overview page and use the Test Model widget to make sure the model works as expected.

   If you entered example input when creating the model, the Input field will be pre-populated with those values. Click Test. The result returned includes the output response from the model, as well as the ID of the replica that served the request.

   Model response times depend largely on your model code. That is, how long it takes the model function to perform the computation needed to return a prediction. It is worth noting that model replicas can only process one request at a time. Concurrent requests will be queued until the model can process them.

For Cloudera AI APIv2

Create and deploy a model using the Models API as instructed in this example:

This example demonstrates the use of the Models API. To run this example, first do the following:

1. Create a project with the Python template and a legacy engine.
2. Start a session.
3. Run !pip3 install sklearn
4. Run fit.py

The example script first obtains the project ID, then creates and deploys a model.

```
projects = client.list_projects(search_filter=json.dumps({"name": “<your
 project name>”}))
project = projects.projects[0] # assuming only one project is returned by
 the above query
model_body = cmlapi.CreateModelRequest(project_id=project.id, name="Demo
 Model", description="A simple model")
model = client.create_model(model_body, project.id)
model_build_body = cmlapi.CreateModelBuildRequest(project_id=project.id,
 model_id=model.id, file_path="predict.py", function_name="predict", ker
nel="python3")
model_build = client.create_model_build(model_build_body, project.id, mod
el.id)
while model_build.status not in [“built”, “build failed”]:
 print(“waiting for model to build...”)
time.sleep(10)
 model_build = client.get_model_build(project.id, model.id, model_build
.id)
if model_build.status == “build failed”:
```

```
 print("model build failed, see UI for more information")
 sys.exit(1)
print("model built successfully!")
model_deployment_body = cmlapi.CreateModelDeploymentRequest(project_id=p
roject.id, model_id=model.id, build_id=model_build.id)
model_deployment = client.create_model_deployment(model_deployment_body,
project.id, model.id, build.id)
while model_deployment.status not in ["stopped", "failed", "deployed"]:
 print("waiting for model to deploy...")
 time.sleep(10)
 model_deployment = client.get_model_deployment(project.id, model.id, m
odel_build.id, model_deployment.id)
if model_deployment.status != "deployed":
 print("model deployment failed, see UI for more information")
 sys.exit(1)
print("model deployed successfully!")
```

Usage guidelines for deploying models with Cloudera AI

Consider these guidelines when deploying models with Cloudera AI.

Model Code

Models in Cloudera AI are designed to run any code that is wrapped into a function. This means you can potentially deploy a model that returns the result of a SELECT * query on a very large table. However, Cloudera strongly recommends against using the models feature for such use cases.

As a best practice, your models should be returning simple JSON responses in near-real time speeds (within a fraction of a second). If you have a long-running operation that requires extensive computing and takes more than 15 seconds to complete, consider using batch jobs instead.

Model Artifacts

Once you start building larger models, make sure you are storing these model artifacts in HDFS, S3, or any other external storage. Do not use the project filesystem to store large output artifacts.

In general, any project files larger than 50 MB must be part of your project's .gitignore file so that they are not included in *Engines for Experiments and Models* for future experiments/model builds.

Notice:

In case your models require resources that are stored outside the model itself, it is up to you to ensure that these resources are available and immutable as model replicas may be restarted at any time.

Resource Consumption and Scaling

Models should be treated as any other long-running applications that are continuously consuming memory and computing resources. If you are unsure about your resource requirements when you first deploy the model, start with a single replica, monitor its usage, and scale as needed.

If you notice that your models are getting stuck in various stages of the deployment process, check the guidelines on *Monitoring Active Models* to make sure that the cluster has sufficient resources to complete the deployment operation.

Security Considerations

As stated previously, models do not impose any limitations on the code they can run. Additionally, models run with the permissions of the user that creates the model (same as sessions and jobs).

Therefore, be conscious of potential data leaks especially when querying underlying data sets to serve predictions.

Cloudera AI models are not public by default. Each model has an access key associated with it. Only users/applications who have this key can make calls to the model. Be careful with who has permission to view this key.

Cloudera AI also prints stderr/stdout logs from models to an output pane in the UI. Make sure you are not writing any sensitive information to these logs.

Deployment Considerations

Models deployed using Cloudera AI in the public cloud are highly available subject to the following limitations:

- Model high availability is dependent on the high availability of the cloud provider's Kubernetes service. Please refer to your chosen cloud provider for precise SLAs.
- Model high availability is dependent on the high availability of the cloud provider's load balancer service. Please refer to your chosen cloud provider for precise SLAs.
- In the event that the Kubernetes pod running the model proxy service becomes unavailable, the Model may be unavailable for multiple seconds during failover.

There can only be one active deployment per model at any given time. This means you shall plan for model downtime if you want to deploy a new build of the model or re-deploy with more or fewer replicas.

Keep in mind that models that have been developed and trained using Cloudera AI are essentially Python or R code that can easily be persisted and exported to external environments using popular serialization formats such as Pickle, PMML, ONNX, and so on.

Related Information

Technical Metrics for Models

Known Issues and Limitations with Model Builds and Deployed Models

- Known Issues with Model Builds and Deployed Models
  - Re-deploying or re-building models results in model downtime (usually brief).
  - Re-starting Cloudera AI does not automatically restart active models. These models must be manually restarted so they can serve requests again.

    Cloudera Bug: DSE-4950
  - Model builds will fail if your project filesystem includes a .git directory (likely hidden or nested). Typical build stage errors include:

    ```
    Error: 2 UNKNOWN: Unable to schedule build: [Unable to create a checkpoi
    nt of current source: [Unable to push sources to git server: ...
    ```

    To work around this, rename the .git directory (for example, NO.git) and re-build the model.

    Cloudera Bug: DSE-4657
  - JSON requests made to active models should not be more than 5 MB in size. This is because JSON is not suitable for very large requests and has high overhead for binary objects such as images or video. Call the model with a reference to the image or video, such as a URL, instead of the object itself.
  - Any external connections, for example, a database connection or a Spark context, must be managed by the model's code. Models that require such connections are responsible for their own setup, teardown, and refresh.
  - Model logs and statistics are only preserved so long as the individual replica is active. Cloudera AI may restart a replica at any time it is deemed necessary (such as bad input to the model).
  - (MLLib) The MLLib model.save() function fails with the following sample error. This occurs because the Spark executors on Cloudera AI all share a mount of /home/cdsw which results in a race condition as multiple executors attempt to write to it at the same time.

    ```
    Caused by:
                      java.io.IOException: Mkdirs failed to create
                      file:/home/cdsw/model.mllib/metadata/_temporary ....
    ```

    Recommended workarounds:

    - Save the model to /tmp, then move it into /home/cdsw on the driver/session.
    - Save the model to either an S3 URL or any other explicit external URL.
- Limitations with Model Builds and Deployed Models
  - Scala models are not supported.
  - Spawning worker threads are not supported with models.
  - Models deployed using Cloudera AI in the public cloud are highly available subject to the following limitations:
    - Model high availability is dependent on the high availability of the cloud provider's Kubernetes service. Please refer to your chosen cloud provider for precise SLAs.
    - Model high availability is dependent on the high availability of the cloud provider's load balancer service. Please refer to your chosen cloud provider for precise SLAs.
    - In the event that the Kubernetes pod running the model proxy service becomes unavailable, the Model may be unavailable for multiple seconds during failover.
  - Dynamic scaling and auto-scaling are not currently supported. To change the number of replicas in service, you will have to re-deploy the build.

Related Information

Distributed Computing with Workers

Model Request and Response Formats

Every model function in Cloudera AI takes a single argument in the form of a JSON-encoded object, and returns another JSON-encoded object as output. This format ensures compatibility with any application accessing the model using the API, and gives you the flexibility to define how JSON data types map to your model's datatypes.

Model Requests

When making calls to a model, keep in mind that JSON is not suitable for very large requests and has high overhead for binary objects such as images or video. Consider calling the model with a reference to the image or video such as a URL instead of the object itself. Requests to models should not be more than 5 MB in size. Performance may degrade and memory usage increase for larger requests.

Ensure that the JSON request represents all objects in the request or response of a model call. For example, JSON does not natively support dates. In such cases consider passing dates as strings, for example in ISO-8601 format, instead.

For a simple example of how to pass JSON arguments to the model function and make calls to deployed model, see *Creating and deploying a Model*.

Model Responses

Models return responses in the form of a JSON-encoded object. Model response times depend on how long it takes the model function to perform the computation needed to return a prediction. Model replicas can only process one request at a time. Concurrent requests are queued until a replica is available to process them.

When Cloudera AI receives a call request for a model, it attempts to find a free replica that can answer the call. If the first arbitrarily selected replica is busy, Cloudera AI will keep trying to contact a free replica for 30 seconds. If no replica is available, Cloudera AI will return a model.busy error with HTTP status code 429 (Too Many Requests). If you see such errors, re-deploy the model build with a higher number of replicas.

Model request timeout

You can set the model request timeout duration to a custom value. The default value is 30 seconds. The timeout can be changed if model requests might take more than 30 seconds.

To set the timeout value:

1. As an Administrator user, open a CLI.
2. At the prompt, execute the following command. Substitute <value> with the number of seconds to set.

   ```
   kubectl set env deployment model-proxy MODEL_REQUEST_TIMEOUT_SECONDS=<va
   lue> -n mlx
   ```

   This edits the `kubeconfig` file and sets a new value for the timeout duration.

Related Information

Creating and deploying a Model

Workflows for Active Models

Testing calls to a Model

Cloudera AI provides two ways to test calls to a model:

- Test Model Widget

  On each model's Overview page, Cloudera AI provides a widget that makes a sample call to the deployed model to ensure it is receiving input and returning results as expected.

- Sample Request Strings

  On the model Overview page, Cloudera AI also provides sample curl and POST request strings that you can use to test calls to the model. Copy/paste the curl request directly into a Terminal to test the call.

  Note that these sample requests already include the example input values you entered while building the model, and the access key required to query the model.

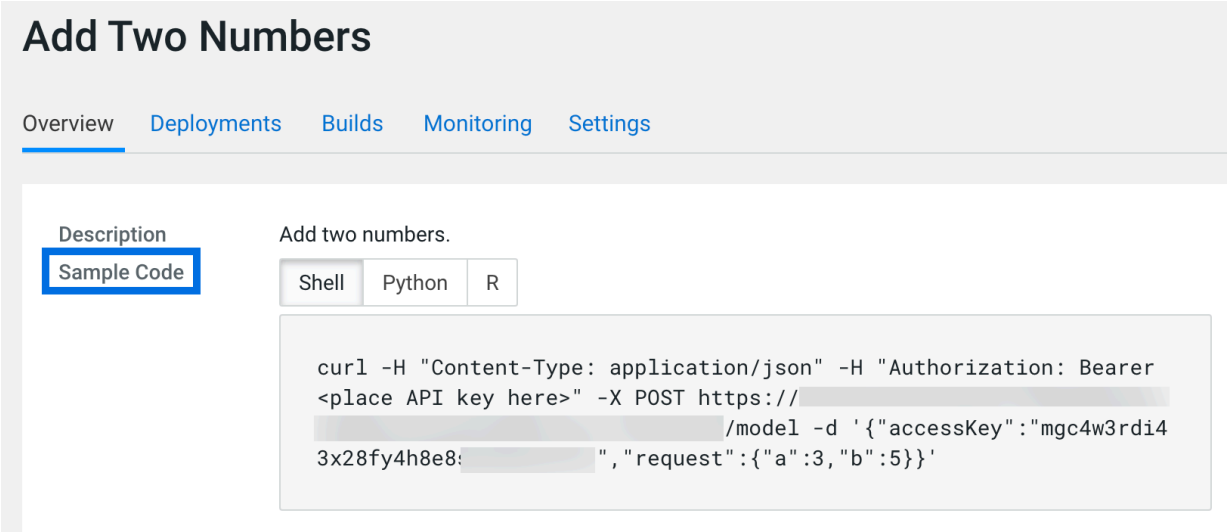

Securing Models

You can secure your Cloudera AI models using Access keys or API keys.

Important: Cloudera Public Cloud allows customers to maintain full ownership and control of their data and workloads and is designed to operate in some of the most restricted Public Cloud environments. Since Cloudera Public Cloud runs in a customer's cloud account, Security and Compliance is a shared responsibility between Cloudera and its Public Cloud customers. For more information, see *Cloudera's Shared Responsibility Model*.

Related Information

Cloudera's Shared Responsibility Model

Access Keys for Models

Each model in Cloudera AI has a unique access key associated with it. This access key is a unique identifier for the model.

Models deployed using Cloudera AI are not public. In order to call an active model your request must include the model's access key for authentication (as demonstrated in the sample calls above).

To locate the access key for a model, go to the model Overview page and click Settings.

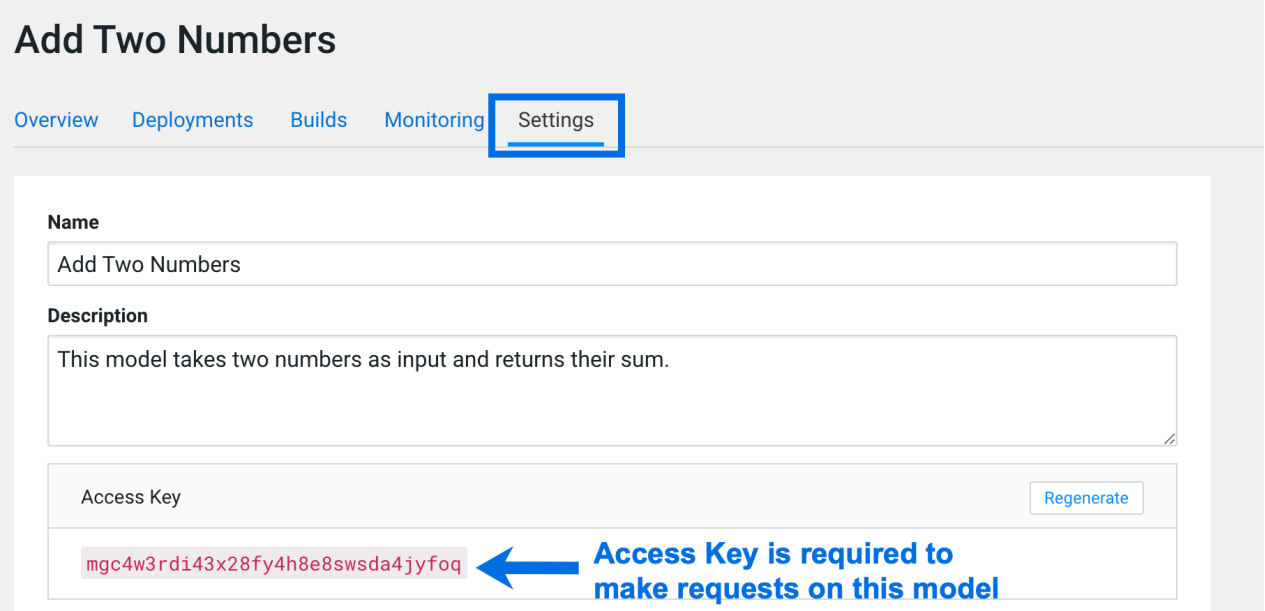

Important:

Only one access key per model is active at any time. If you regenerate the access key, you will need to re-distribute this access key to users/applications using the model.

Alternatively, you can use this mechanism to revoke access to a model by regenerating the access key. Anyone with an older version of the key will not be able to make calls to the model.

API Key for Models

You can prevent unauthorized access to your models by specifying an API key in the **Authorization** header of your model HTTP request. This topic covers how to create, test, and use an API key in Cloudera AI.

The API key governs the authentication part of the process and the authorization is based on what privileges the users already have in terms of the project that they are a part of. For example, if a user or application has read-only access to a project, then the authorization is based on their current access level to the project, which is “read-only”. If the users have been authenticated to a project, then they can make a request to a model with the API key. This is different from the previously described Access Key, which is only used to identify which model should serve a request.

Enabling authentication

Restricting access using API keys is an optional feature. By default, the **Enable Authentication** option is turned on. However, it is turned off by default for the existing models for backward compatibility. You can enable authentication for all your existing models.

To enable authentication, go to Projects Models Settings and check the Enable Authentication option.

Note: It can take up to five minutes for the system to update.

Generating an API key

If you have enabled authentication, then you need an API key to call a model. If you are not a collaborator on a particular project, then you cannot access the models within that project using the API key that you generate. You need to be added as a collaborator by the admin or the owner of the project to use the API key to access a model.

About this task

There are two types of API keys used in Cloudera AI:

- API Key: These are used to authenticate requests to a model. You can choose the expiration period and delete them when no longer needed.
- Legacy API Key: This is used in the CDSW-specific internal APIs for CLI automation. This cannot be deleted and neither does it expire. This API Key is not required when sending requests to a model.

You can generate more than one API keys to use with your model, depending on the number of clients that you are using to call the models.

Procedure

1. Sign in to Cloudera AI.
2. Click Settings from the left navigation pane.
3. On the User Settings page, click the API Keys tab.
4. Select an expiry date for the Model API Key, and click Create API keys.

   An API key is generated along with a Key ID.

   If you do not specify an expiry date, then the generated key is active for one year from the current date, or for the duration set by the Administrator. If you specify an expiration date that exceeds the duration value set by the Administrator, you will get an error. The Administrator can set the default duration value at Admin Security Default API keys expiration in days

   Note:

   - The API key is private and ephemeral. Copy the key and the corresponding key ID on to a secure location for future use before refreshing or leaving the page. If you miss storing the key, then you can generate another key.
   - You can delete the API keys that have expired or no longer in use. It can take up to five minutes by the system to take effect.

5. To test the API key:
   a) Navigate to your project and click Models from the left navigation pane.
   b) On the **Overview** page, paste the API key in the API key field that you had generated in the previous step and click Test.

      The test results, along with the HTTP response code and the Replica ID are displayed in the Results table.

      If the test fails and you see the following message, then you must get added as a collaborator on the respective project by the admin or the creator of the project:

```
"User APikey not authorized to access model": "Check APIKEY permissions
or model authentication permissions"
```

Managing API Keys

The administrator user can access the list of all the users who are accessing the workbench and can delete the API keys for a user.

About this task

To manage users and their keys:

Procedure

1. Sign in to Cloudera AI as an administrator user.
2. From the left navigation pane, click Admin.

   The **Site Administration** page is displayed.
3. On the **Site Administration** page, click on the Users tab.

   All the users signed under this workbench are displayed.

   The API Keys column displays the number of API keys granted to a user.
4. To delete a API key for a particular user:
   a) Select the user for which you want to delete the API key.

      A page containing the user's information is displayed.
   b) To delete a key, click Delete under the Action column corresponding to the Key ID.
   c) Click Delete all keys to delete all the keys for that user.

   Note: It can take up to five minutes by the system to take effect.

   As a non-admin user, you can delete your own API key by navigating to Settings User Settings API Keys .

Workflows for active Models

This topic walks you through some nuances between the different workflows available for re-deploying and re-building models.

Active Model - A model that is in the Deploying, Deployed, or Stopping stages.

You can make changes to a model even after it has been deployed and is actively serving requests. Depending on business factors and changing resource requirements, such changes will likely range from changes to the model code itself, to simply modifying the number of CPU/GPUs requested for the model. In addition, you can also stop and restart active models.

Depending on your requirement, you can perform one of the following actions:

Re-deploy an existing build

Re-deploying a model involves re-publishing a previously-deployed model in a new serving environment - this is, with an updated number of replicas or memory/CPU/GPU allocation. For example, circumstances that require a re-deployment might include:

- An active model that previously requested a large number of CPUs/GPUs that are not being used efficiently.
- An active model that is dropping requests because it is falling short of replicas.
- An active model needs to be rolled back to one of its previous versions.

Warning: Currently, Cloudera AI only allows one active deployment per model. This means when you re-deploy a build, the current active deployment will go offline until the re-deployment process is complete and the new deployment is ready to receive requests. Prepare for model downtime accordingly.

To re-deploy an existing model:

1. Go to the model Overview page.
2. Click Deployments.
3. Select the version you want to deploy and click Re-deploy this Build.

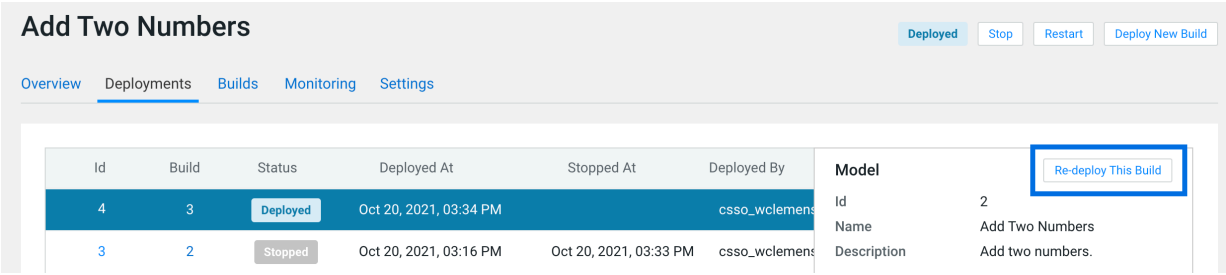

4. Modify the model serving environment as needed.
5. Click Deploy Model.

Deploy a new build for a Model

Deploying a new build for a model involves both, re-building the Docker image for the model, and deploying this new build. Note that this is not required if you only need to update the resources allocated to the model. As an example, changes that require a new build might include:

- Code changes to the model implementation.
- Renaming the function that is used to invoke the model.

Warning: Currently, Cloudera AI does not allow you to create a new build for a model without also deploying it. This combined with the fact that you can only have one active deployment per model means that once the new model is built, the current active deployment will go offline so that the new build can be deployed. Prepare for model downtime accordingly.

To create a new build and deploy it:

1. Go to the model Overview page.
2. Click Deploy New Build.

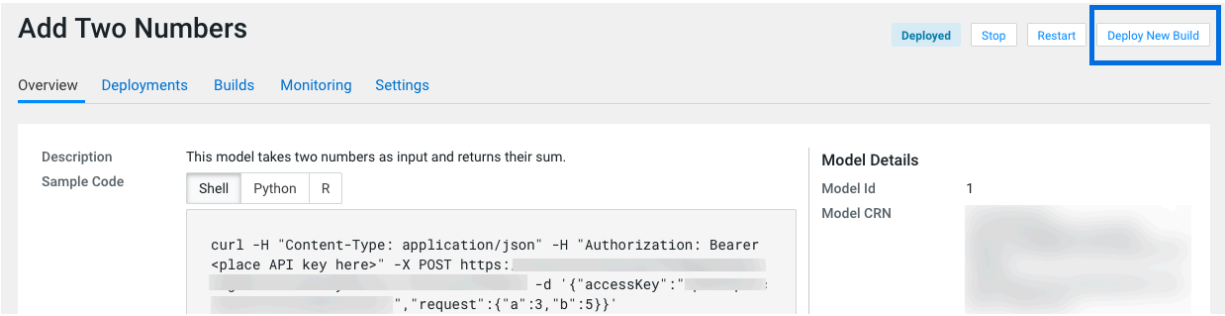

3. Complete the form and click Deploy Model.

Stop a Model

To stop a model (all replicas), go to the model Overview page and click Stop. Click OK to confirm.

Restart a Model

To restart a model (all replicas), go to the model Overview page and click Restart. Click OK to confirm.

Restarting a model does not let you make any code changes to the model. It should primarily be used as a way to quickly re-initialize or re-connect to resources.

Technical metrics for Models

You can observe the operation of your models by using charts provided for technical metrics. These charts can help you determine if your models are under- or over-resourced, or are experiencing some problem.

To check the performance of your model, go to Models, click on the model name, and select the Monitoring tab. You can choose to monitor all replicas of the model, or choose a specific replica. You can also select the time and date range to display. Up to two weeks of data is retained.

This tab displays charts for the following technical metrics:

- Requests per Second
- Number of Requests
- Number of Failed Requests
- Model Response Time
- All Model Replica CPU Usage
- All Model Replica Memory Usage
- Model Request & Response Size

All charts share a common time axis (the x axis), so it is easy to correlate CPU and memory usage with model response time or the number of failed requests, for example.

Debugging issues with Models

This topic describes some common issues to watch out for during different stages of the model build and deployment process.

As a general rule, if your model spends too long in any of the afore-mentioned stages, check the resource consumption statistics for the cluster. When the cluster starts to run out of resources, often models will spend some time in a queue before they can be executed.

Resource consumption by active models on a deployment can be tracked by site administrators on the Admin Models page.

Building

Live progress for this stage can be tracked on the model's Build tab. It shows the details of the build process that creates a new Docker image for the model. Potential issues:

- If you specified a custom build script (cdsw-build.sh), ensure that the commands inside the script complete successfully.
- If you are in an environment with restricted network connectivity, you might need to manually upload dependencies to your project and install them from local files.

Pushing

Once the model has been built, it is copied to an internal Docker registry to make it available to all the Cloudera AI hosts. Depending on network speeds, your model may spend some time in this stage.

Deploying

If you see issues occurring when Cloudera AI is attempting to start the model, use the following guidelines to begin troubleshooting:

- Make sure your model code works in a workbench session. To do this, launch a new session, run your model file, and then interactively call your target function with the input object. For a simple example, see the *Creating and deploying a Model*.
- Ensure that you do not have any syntax errors. For Python, make sure you have the kernel with the appropriate Python version (Python 2 or Python 3) selected for the syntax you have used.
- Make sure that your cdsw-build.sh file provides a complete set of dependencies. Dependencies manually installed during a session on the workbench are not carried over to your model. This is to ensure a clean, isolated, build for each model.
- If your model accesses resources such as data on the CDH cluster or an external database make sure that those resources can accept the load your model may exert on them.

Deployed

Once a model is up and running, you can track some basic logs and statistics on the model's Monitoring page. In case issues arise:

- Check that you are handling bad input from users. If your function throws an exception, Cloudera AI will restart your model to attempt to get back to a known good state. The user will see an unexpected model shutdown error.

  For most transient issues, model replicas will respond by restarting on their own before they actually crash. This auto-restart behavior helps keeping the model online as you attempt to debug runtime issues.
- Make runtime troubleshooting easier by printing errors and output to stderr and stdout. You can catch these on each model's Monitoring tab. Be careful not to log sensitive data here.
- The Monitoring tab also displays the status of each replica and will show if the replica cannot be scheduled due to a lack of cluster resources. It will also display how many requests have been served/dropped by each replica.

Related Information

Creating and deploying a Model

Technical Metrics for Models

Deleting a Model

Before you begin

Important:

- You must stop all active deployments before you delete a model. If the deployments are not stopped, the active models will continue serving requests and consuming resources even though they do not show up in Cloudera AI UI.
- Deleted models are not actually removed from disk. That is, this operation will not free up storage space.

Procedure

1. Go to the model Overview Settings .

2. Click Delete Model.

   Deleting a model removes all of the model's builds and its deployment history from Cloudera AI.

   You can also delete specific builds from a model's history by going to the model's Overview Build page.

Configuring model request payload size

Model metrics have a configuration that restricts model request payload to 100 KB. You can increase the payload size if required.

Procedure

1. Convert the payload size to bytes.

   For example, 100 KB (Kilobytes) = 100 * 1024 bytes = 102400 bytes.
2. Encode the values into the Base64 format.

   For example, 20 MB in Bytes is 20000000, to convert it to Base64 value, run the following command on the terminal:

   ```
   echo -n "20000000" | base64
   ```

   The output would be MjAwMDAwMDA=. Here, MjAwMDAwMDA= is the Base64 encoded value.
3. Edit the existing Secret object to specify the model request payload size.

   ```
   kubectl edit secret model-metrics-config -n mlx
   ```

   This opens your default editor and allows you to update the Base64 encoded values.
4. Locate the max.request.size.bytes field and update it with the Base64 encoded value.

   ```
   max.request.size.bytes: [***Base64-encoded-value***]
   ```

   For example, substitute the above 20 MB Base64 encoded value:

   ```
   max.request.size.bytes: MjAwMDAwMDA=
   ```
5. Save and close the editor.
6. Restart the model metrics pod by deleting the pod.

   ```
   kubectl delete pod modelmetrics [***pod nam-ext***] -n mlx
   ```

   This will force Kubernetes to restart the pod with the updated configuration.

Managing Cloudera Copilot

Learn how to configure and use Cloudera Copilot with Cloudera AI Inference service and Amazon Bedrock models.

Cloudera Copilot is an AI-powered coding assistant designed for seamless integration within JupyterLab ML Runtimes. With its chat interface and comprehensive code completion features, Cloudera Copilot enhances the development experience for machine learning projects. It offers compatibility with model endpoints deployed in Cloudera AI Inference service as well as Amazon Bedrock models, providing developers with flexibility and efficiency in their workflows.

Cloudera AI Inference service vs Amazon Bedrock

Cloudera AI Inference service model endpoints may be a good choice if you are concerned about proprietary data being sent to a third-party service provider. With Cloudera AI Inference service, you can run your own models and ensure that your proprietary data stays within your cloud deployment.

If you are not concerned about proprietary data being sent to a third-party service provider, and you do not expect high volumes of Cloudera Copilot usage, then using Amazon Bedrock models may be a simpler and more cost-effective solution.

Configuring Cloudera AI Inference service and Amazon Bedrock to set up Cloudera Copilot

To use Cloudera Copilot, as a Site Administrator, set up Cloudera AI Inference service model endpoints or configure the credentials in Amazon Bedrock depending on where you want to deploy your custom model.

For Cloudera AI Inference

Cloudera AI Inference service is a production-grade serving environment for traditional, generative AI, and LLM models. It is designed to handle the challenges of production deployments, such as high availability, fault tolerance, and scalability. Follow the steps to configure Cloudera AI Inference service model endpoints to use Cloudera Copilot.

1. Configure authentication, and authorization, and import a model from NGC.
2. Create a Cloudera AI Inference service instance.
3. Create a model endpoint using UI.

No additional credentials need to be configured to use Cloudera AI Inference service model endpoints with Cloudera Copilot.

For Amazon Bedrock

To use Cloudera Copilot, as a Site Administrator, you can configure the credentials to use Amazon Bedrock models.

1. Generate a pair of Access and Secret keys through AWS IAM. For more information, see Manage access keys for IAM users.
2. In the **Cloudera** console, click the **Cloudera AI** tile.

   The **Cloudera AI Workbenches** page displays.
3. Click on the workbench name.

   The **Workbenches Home** page displays.
4. Click **Site Administration** in the left navigation menu.

   The Site Administration page displays.
5. Click Settings Environment Variables and add the following obtained in Step 1:
   - AWS_SECRET_ACCESS_KEY
   - AWS_ACCESS_KEY_ID
   - AWS_DEFAULT_REGION

Configuring Cloudera Copilot

After setting up credentials, you must make configuration changes in the Cloudera AI UI before using Cloudera Copilot.

Choosing a model

Models vary in accuracy, and cost. Larger models will provide more accurate responses but will cost more. For Cloudera AI Inference service models, larger models require more expensive GPU hardware to run on, while in Amazon Bedrock, larger models will cost more per prompt.

Language models vs Embedding models

Cloudera Copilot supports the following model types:

- Language models: These are used for code completion, debugging, and chat.
- Embedding models: These are used for Retrieval Augmented Generation (RAG) use cases. This allows you to augment language model responses with specific information that a language model is not aware of. For example, you can provide internal company documents that map company acronyms to their definitions.

Recommended models

- Language models:
  - Llama 3.1 Instruct 70b (AI Inference)
  - Claude v3.5 Sonnet (Amazon Bedrock)
- Embedding models:
  - E5 Embedding v5 (AI Inference)
  - Titan Embed Text v2 (Amazon Bedrock)

Procedure

1. In the **Cloudera** console, click the **Cloudera AI** tile.

   The **Cloudera AI Workbenches** page displays.
2. Click on the workbench name.

   The **Workbenches Home** page displays.
3. Click **Site Administration** in the left navigation menu.

   The **Site Administration** page displays.
4. Click Settings , and select the Enable Cloudera Copilot checkbox under Feature Flags.

   A new navigation tab Cloudera Copilot appears at the top of the **Site Administration** page.
5. Click the Cloudera Copilot tab.

   The **Cloudera Copilot** page displays.
6. Select the type of model you want to add: Language Model or Embedding Model.

7. Click Add Model button.

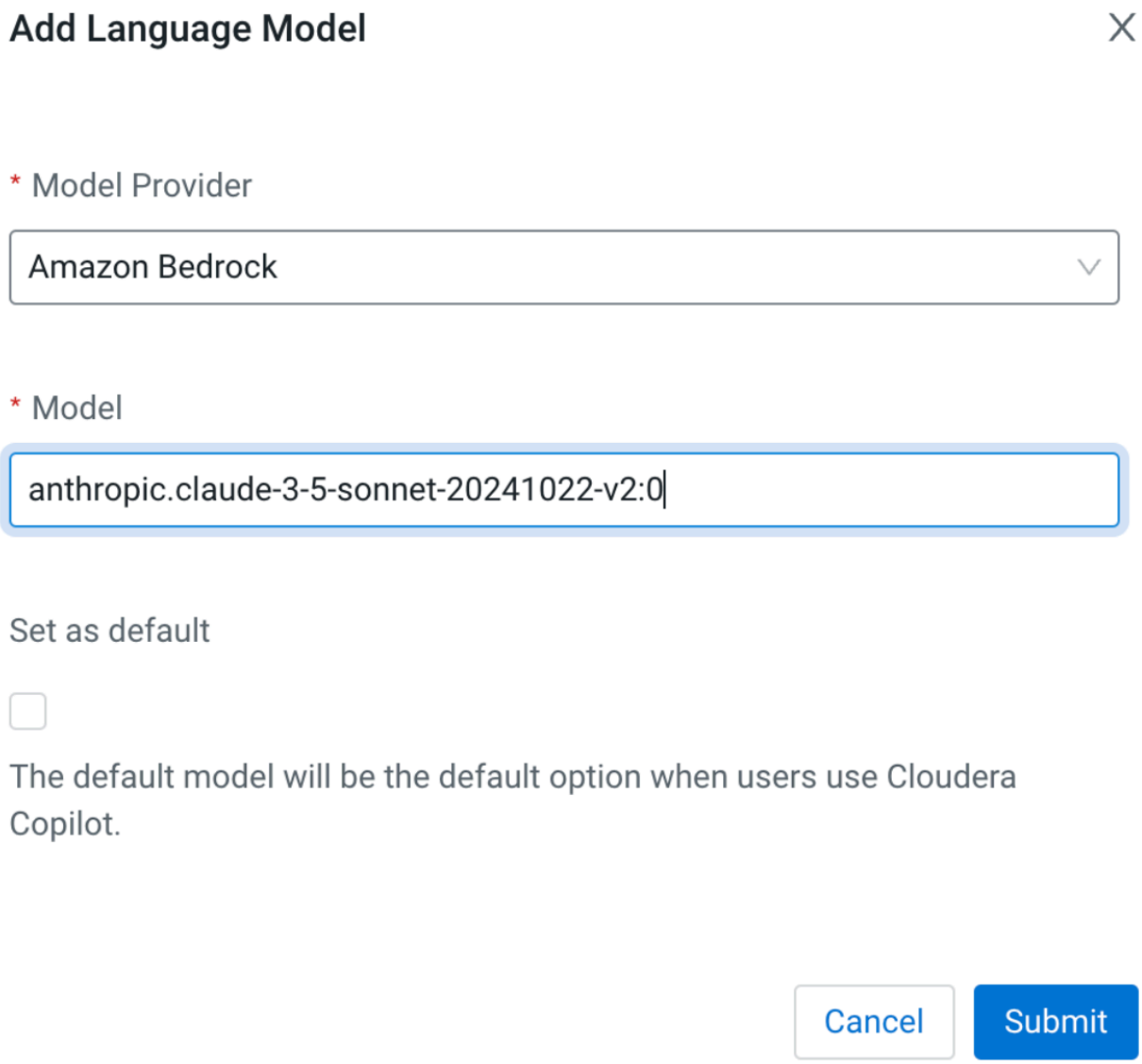

8. Select a model provider from the Model Provider dropdown list.
9. In the Model field, provide the model name:
   - For Bedrock models: Select a model name from the Model dropdown list.
   - For Cloudera AI Inference service models, provide the model endpoint and the `model_id` as the model name string. You can get the model endpoint and `model_id` information from the Model Endpoint details page.
     - Example Model Endpoint: *HTTPS://CAII-PROD-LONG-RUNNING.ENG-ML-L.VNU8-SQZE.YOURCOMPANY.SITE/NAMESPACES/SERVING-DEFAULT/ENDPOINTS/LLAMA-31-8B-INSTRUCT-2XA10G/V1/CHAT/COMPLETIONS*
     - Example model ID: *PHWQ-GQMD-4KOS-PERD*

   The model that you add for the first time is selected as the default model automatically and the deselect option is disabled. This is to enforce that there is always one default model. When you add more models, you can choose one of them to be the default model. You can have one default language model, and one default embedding model.
10. Click Add.

Results

The model appears under Models or Third Party Models depending on the model provider type you selected.

Using Cloudera Copilot

You can use Cloudera Copilot to assist you in code generation and code completion thus providing flexibility and efficiency in the Cloudera AI workflows.

Procedure

1. Start a session within your project and select JupyterLab as your editor.

2. Click the Jupyter AI Chat icon in the left navigation pane.

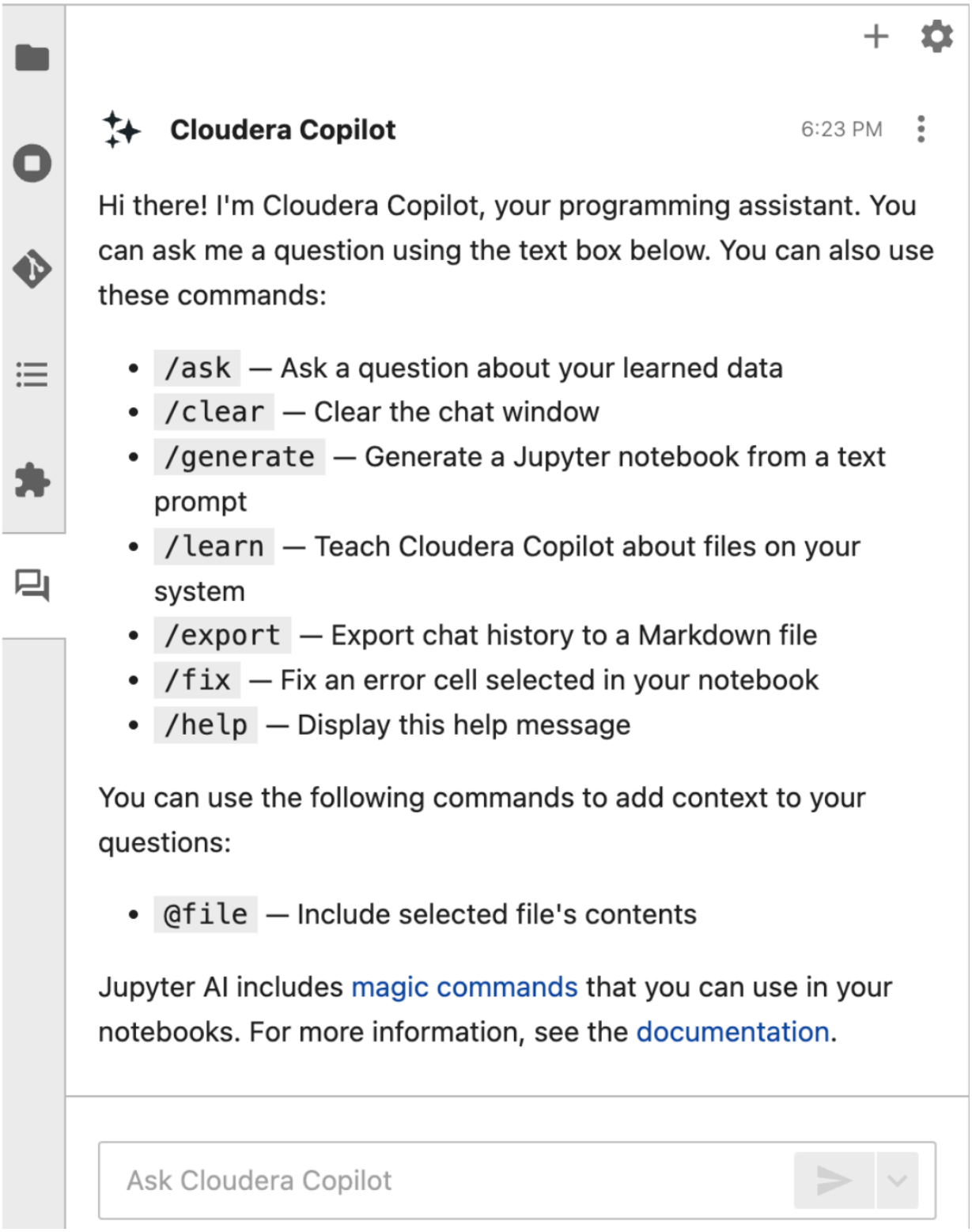

3. Start typing in the prompt and the Cloudera Copilot will assist you.

   For example, enter Help me write a fibonacci function.

   Example use cases for Cloudera Copilot:

   - Debugging/fixing code
   - Code completion
   - Explaining code or explaining errors
   - Code generation
   - Refer to Jupyter AI documentation for instructions on /ask /learn /fix and /generate commands.
4. **Magic commands:**
   - In addition to using the chat interface, you can also use Cloudera Copilot by calling Magic Commands within your notebook. Magic commands are special lines of code that will make calls to a language model that you have configured.
   - Example syntax:
     - %load_ext jupyter_ai_magics
     - %%ai anthropic.claude-3-sonnet-20240229-v1:0
     - def fibonacci(
   - Finding model ID strings for magic commands.
     - The Amazon Bedrock models recommended for Cloudera Copilot are:
       - anthropic.claude-3-5-sonnet-20241022-v2:0
       - mistral.mixtral-8x7b-instruct-v0:1
       - meta.llama3-1-70b-instruct-v1:0
       - meta.llama3-1-405b-instruct-v1:0
       - amazon.titan-embed-text-v2:0
     - For Cloudera AI Inference service, the model ID string will be of the form:
       - cloudera:<*MODEL_NAME*>. Here, the model_name is the same model ID from the Model Endpoints details page that you added when you configured Cloudera Copilot.

Example - Model training and deployment (Iris)

This topic uses Cloudera AI's built-in Python template project to walk you through an end-to-end example where we use experiments to develop and train a model, and then deploy it using Cloudera AI.

This example uses the canonical Iris dataset from Fisher and Anderson to build a model that predicts the width of a flower’s petal based on the petal's length.

The scripts for this example are available in the Python template project that ships with Cloudera AI. First, create a new project from the Python template:

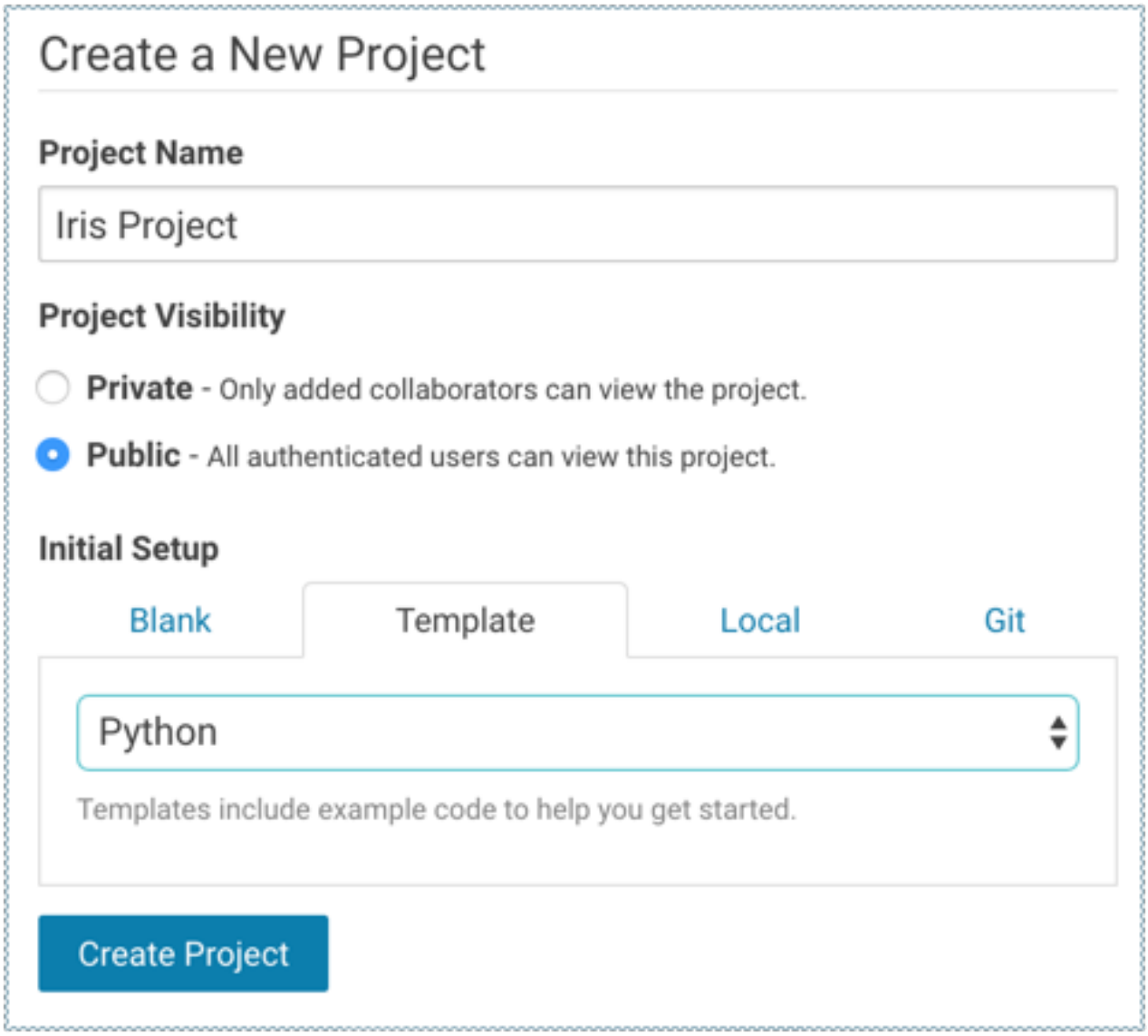

Once you have created the project, go to the project's Files page. The following files are used for the demo:

- cdsw-build.sh - A custom build script used for models and experiments. Pip installs our dependencies, primarily the scikit-learn library.
- fit.py - A model training example to be run as an experiment. Generates the model.pkl file that contains the fitted parameters of our model.
- predict.py - A sample function to be deployed as a model. Uses model.pkl produced by fit.py to make predictions about petal width.

Training the Model

This topic shows you how to run experiments and develop a model using the fit.py file.

About this task

The fit.py script tracks metrics, mean squared error (MSE) and R2, to help compare the results of different experiments. It also writes the fitted model to a model.pkl file.

Procedure

1. Navigate to the Iris project's Overview Experiments page.
2. Click Run Experiment.
3. Fill out the form as follows and click Start Run. Make sure you use the Python 3 kernel.

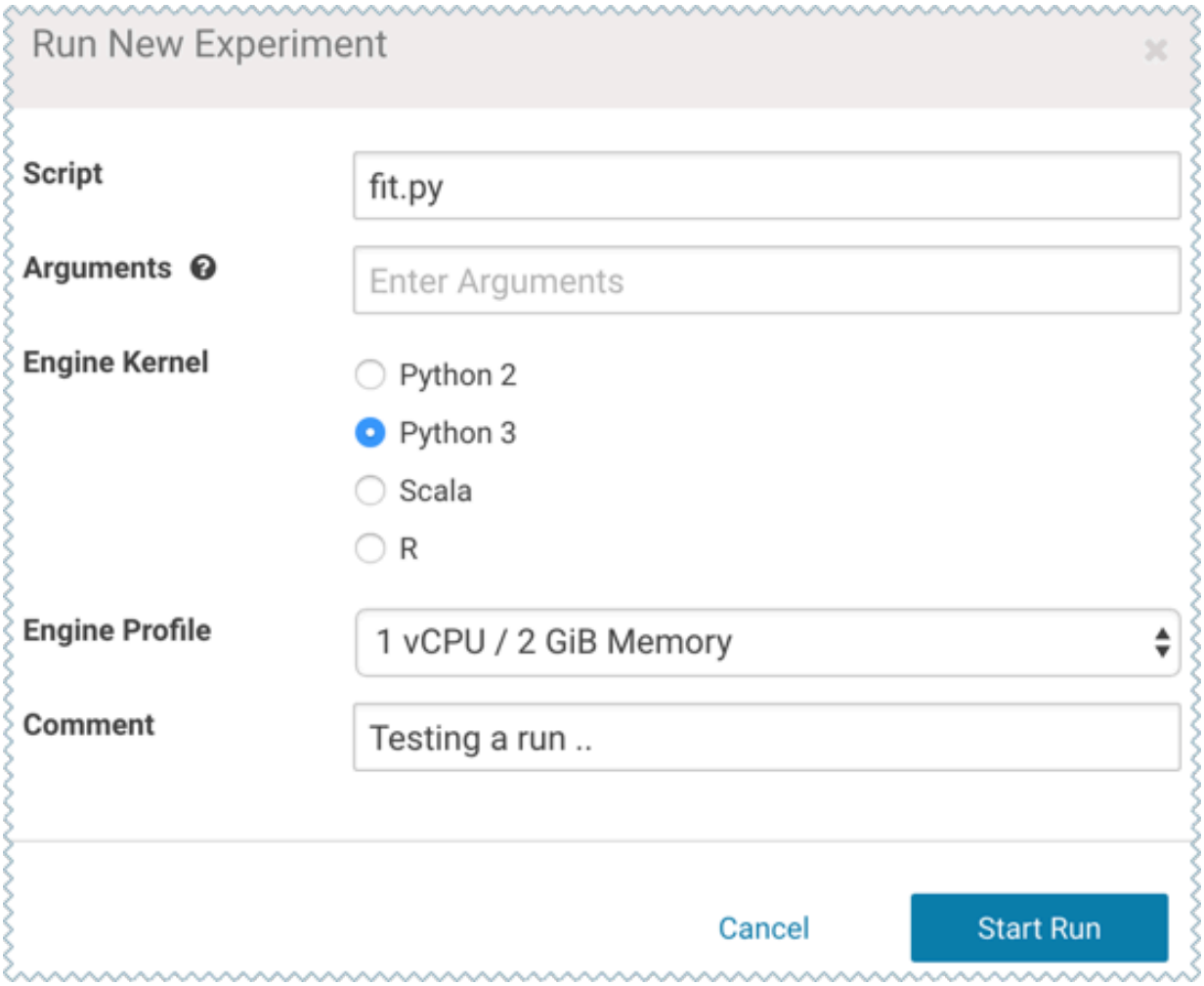

4. The new experiment shall now display on the Experiments table. Click on the Run ID to go to the experiment's Overview page. The Build and Session tabs display realtime progress as the experiment builds and executes.

5. Once the experiment has completed successfully, go back to its Overview page. The tracked metrics show us that our test set had an MSE of ~0.0078 and an R2 of ~0.0493. For the purpose of this demo, let's consider this an accurate enough model to deploy and use for predictions.

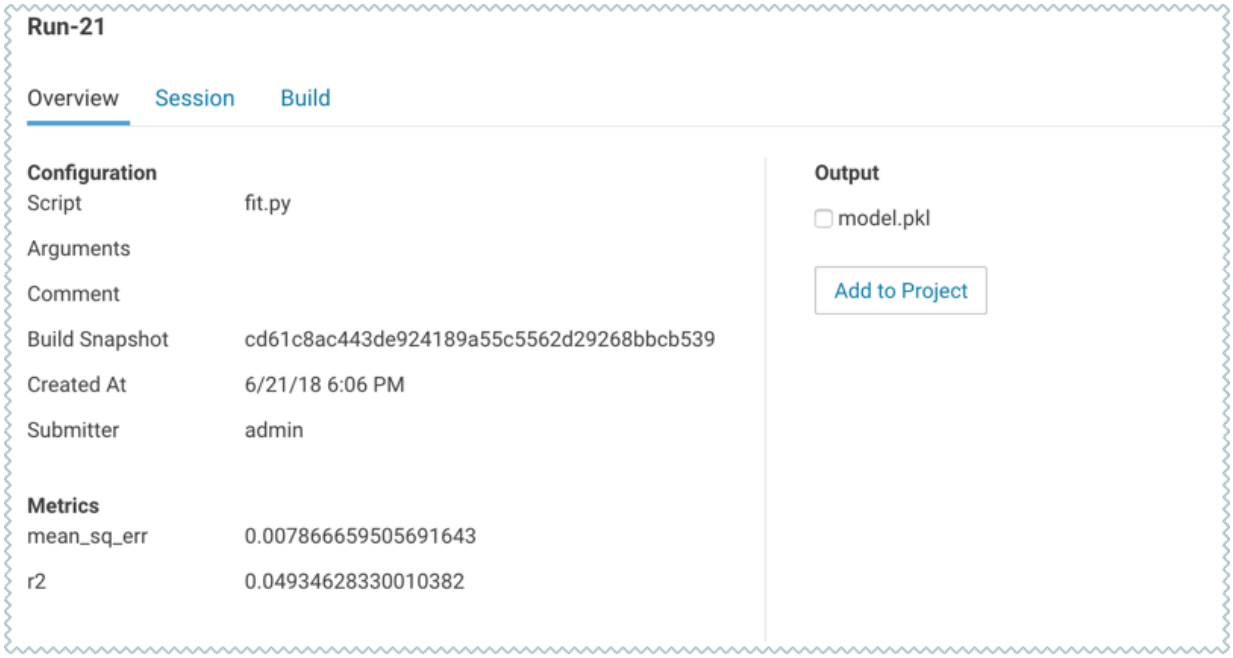

6. Once you have finished training and comparing metrics from different experiments, go to the experiment that generated the best model. From the experiment's Overview page, select the model.pkl file and click Add to Project.
   This saves the model to the project filesystem, available on the project's Files page. We will now deploy this model as a REST API that can serve predictions.

Deploying the Model

This topic shows you how to deploy the model using the predict.py script from the Python template project.

About this task

The predict.py script contains the predict function that accepts petal length as input and uses the model built in the previous step to predict petal width.

Procedure

1. Navigate to the Iris project's Overview Models page.

2. Click New Model and fill out the fields. Make sure you use the Python 3 kernel. For example:

Create a Model

General

Name *

Predict Petal Width

Description *

This model uses petal length to predict petal width.

Build

File *

predict.py

Function *

predict

Example Input

```
{
  "petal_length": 5.4
}
```

Example Output

```
{ "result": "value" }
```

Kernel

- ○ Python 2
- ◉ Python 3
- ○ R

Comment

Using Python 3 for this build

Deployment

Engine Profile

1 vCPU / 2 GiB Memory

Replicas

3

Set Environmental Variables

Deploy Model Cancel

3. Deploy the model.
4. Click on the model to go to its Overview page. As the model builds you can track progress on the Build page. Once deployed, you can see the replicas deployed on the Monitoring page.
5. To test the model, use the Test Model widget on the model's Overview page.

Cloudera AI Product Overview

Date published: 2020-07-16
Date modified: 2024-11-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera AI overview..... 4
AI applications..... 4
Exploratory Data Science..... 4
ML Ops..... 4
Core capabilities..... 5
Cloudera AI benefits..... 5
Key differences between Cloudera AI (Cloudera Public Cloud) and Cloudera Data Science Workbench..... 6

Cloudera AI overview

Machine learning has become one of the most critical capabilities for modern businesses to grow and stay competitive today. From automating internal processes to optimizing the design, creation, and marketing processes behind virtually every product consumed, machine learning models have permeated almost every aspect of our work and personal lives.

Machine learning development is iterative and complex, made even harder because most machine learning tools are not built for the entire machine learning lifecycle. Cloudera AI accelerates time-to-value by enabling data scientists to collaborate in a single unified platform that is all inclusive for powering any AI use case. Purpose-built for agile experimentation and production machine learning workflows, Cloudera AI manages everything from data preparation to MLOps, to predictive reporting. You can solve mission critical machine learning challenges along the entire lifecycle with greater speed and agility to discover opportunities which can mean the difference for your business.

Each Cloudera AI Workbench enables teams of data scientists to develop, test, train, and ultimately deploy machine learning models for building predictive applications all on the data under management within the enterprise data cloud. Cloudera AI Workbench support fully-containerized execution of Python, R, Scala, and Spark workloads through flexible and extensible engines.

AI applications

Analytical Applications provide a place to host long running applications within a Cloudera AI project.

While Cloudera AI offers a place for Data Scientists to perform advanced analytics and models into production, Analytical Applications provides a place to host long running applications within a Cloudera AI project. This opens up a larger group of users to the insights discovered in Cloudera AI. Applications can be built with a variety of frameworks like Flask and Streamlit. They run within their own isolated compute instance which keeps them from timing out and they take advantage of ML Runtimes. Applications are accessible to users through the web. Applications can be for a variety of use cases like hosting interactive data visualizations or providing a UI frontend for a deployed mode in Cloudera AI.

Exploratory Data Science

Cloudera AI enables data practitioners to discover, query, and easily visualize their data sets all within a single user interface.

The beginning of every data science project begins with finding and understanding the data you need. Cloudera AI brings all the tools you need for exploratory data analysis together in a single UI so that data practitioners don't have to jump between applications, and IT doesn't have to support multiple tools. Cloudera AI provides users with a view of available data assets that they can connect to, a sql editor to query those data sources, and an easy-to-use drag-and-drop visualization tool to understand your data and communicate insights.

ML Ops

Cloudera AI enables users to deploy machine learning and other models into production.

Cloudera AI enables users to deploy machine learning and other models into production, either as a batch process through the Jobs functionality, or as near-real-time REST APIs using the Models functionality. In addition, Cloudera AI provides a number of features to help maintain, monitor and govern these models in production. The Model Governance feature ensures that every deployed Model is tracked in the Cloudera Data Catalog, and allows the user to specify which data tables were used to train the model in order to provide model-data lineage. Deployed Models have a built-in dashboard for monitoring technical metrics relating to deployed Cloudera AI Models, such as request throughput, latency, and resource consumption. Additionally, users can track arbitrary business metrics relating to each inference event, and match the results with delayed metrics from a data warehouse or other source using an

automatically generated UUID. By analyzing these metrics, the user can assess the model for aggregated metrics such as accuracy on an ongoing basis.

Core capabilities

This section details the core capabilities for Cloudera AI.

Cloudera AI covers the end-to-end machine learning workflow, enabling fully isolated and containerized workloads - including Python, R, and Spark-on-Kubernetes - for scale-out data engineering and machine learning with seamless distributed dependency management.

- Sessions enable Data Scientists to directly leverage the CPU, memory, and GPU compute available across the Cloudera AI Workbenches, while also being directly connected to the data in the data lake.
- Experiments enable Data Scientists to run multiple variations of model training workloads, tracking the results of each Experiment in order to train the best possible Model.
- Models can be deployed in a matter of clicks, removing any roadblocks to production. They are served as REST endpoints in a high availability manner, with automated lineage building and metric tracking for MLOps purposes.
- Jobs can be used to orchestrate an entire end-to-end automated pipeline, including monitoring for model drift and automatically kicking off model re-training and re-deployment as needed.
- Applications deliver interactive experiences for business users in a matter of clicks. Frameworks such as Flask and Shiny can be used in development of these Applications, while Cloudera Data Visualization is also available as a point-and-click interface for building these experiences.

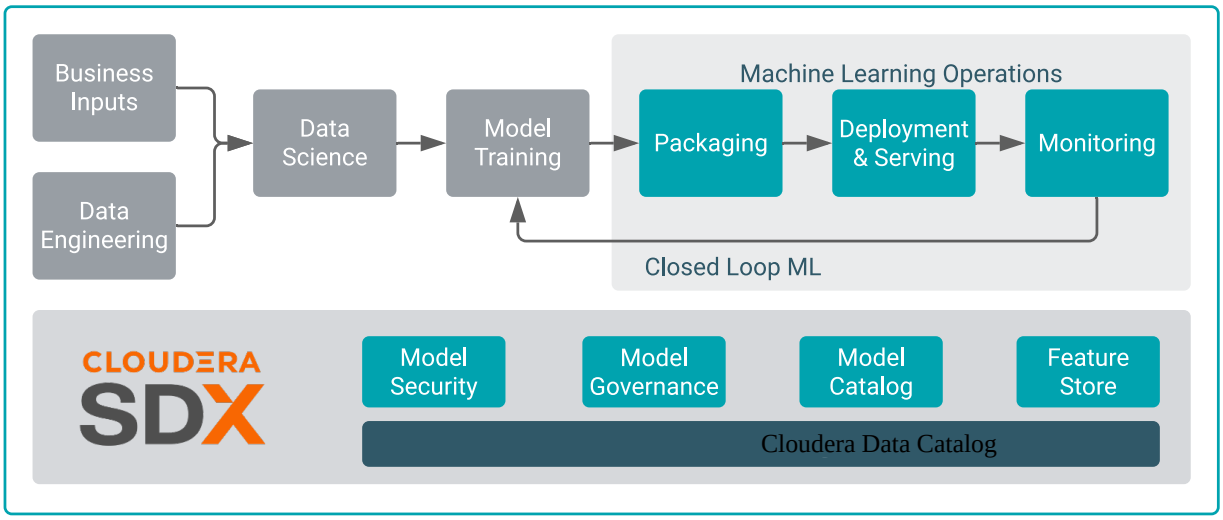

Cloudera AI benefits

This section details the Cloudera AI benefits for each type of user.

Cloudera AI is built for the agility and power of cloud computing, but is not limited to any one provider or data source. It is a comprehensive platform to collaboratively build and deploy machine learning capabilities at scale.

Cloudera AI provides benefits for each type of user.

Data Scientists

- Enable DS teams to collaborate and speed model development and delivery with transparent, secure, and governed workflows
- Expand AI use cases with automated Cloudera AI pipelines and an integrated and complete production Cloudera AI toolkit

- Empower faster decision making and trust with end-to-end visibility and auditability of data, processes, models, and dashboards

IT

- Increase data service productivity with visibility, security, and governance of the complete machine learning lifecycle
- Eliminate silos, blindspots, and the need to move/duplicate data with a fully integrated platform across the data lifecycle.
- Accelerate AI with self-service access and containerized Cloudera AI Workbench that remove the heavy lifting and get models to production faster

Business Users

- Access interactive Applications built and deployed by DS teams.
- Be empowered with predictive insights to more intelligently make business decisions.

Key differences between Cloudera AI (Cloudera Public Cloud) and Cloudera Data Science Workbench

This topic highlights some key differences between Cloudera Data Science Workbench and its cloud-native counterpart, Cloudera AI.

How is Cloudera AI related to Cloudera Data Science Workbench (CDSW)?

Cloudera AI expands the end-to-end workflow of Cloudera Data Science Workbench (CDSW) with cloud-native benefits like rapid provisioning, elastic autoscaling, distributed dependency isolation, and distributed GPU training.

It can run its own native distributed computing workloads without requiring a separate CDH cluster for scale-out compute. It is designed to run on Cloudera Data Platform in existing Kubernetes environments, reducing operational costs for some customers while delivering multi-cloud portability. On Cloudera Public Cloud, managed cloud Kubernetes services include EKS, AKS, or GKE, and on Cloudera Private Cloud, they include Red Hat OpenShift Container Platform or Cloudera Embedded Container Service.

Both products help data engineers and data science teams be more productive on shared data and compute, with strong security and governance. They share extensive code.

There is one primary difference:

- CDSW extends an existing CDH cluster, by running on gateway nodes and pushing distributed compute workloads to the cluster. CDSW requires and supports a single CDH cluster for its distributed compute, including Apache Spark.
- In contrast, Cloudera AI is self-contained and manages its own distributed compute, natively running workloads - including but not limited to Apache Spark - in containers on Kubernetes.

  Note: It can still connect to an existing cluster to leverage its distributed compute, data, or metadata (SDX).

Table 1: Key Differences

	CDSW	Cloudera AI
Architecture	CDSW can run on a Cloudera Data Platform-DC, CDH (5 or 6), and HDP cluster and runs on one or more dedicated gateway nodes on the cluster.	Cloudera AI is self-contained and does not require an attached CDH/HDP cluster.
	Notion of 1 master and multiple worker hosts.	No designated master and worker hosts; all nodes are ephemeral.
Security	Kerberos authentication integrated via the CDH/HDP cluster	Centralised identity management using FreeIPA via the Cloudera Data Platform.
	External authentication via LDAP/SAML.	

	CDSW	Cloudera AI
App Storage	Project files, internal postgresDB, and Livelog, are all stored persistently on the Master host.	All required persistent storage is on cloud-managed block store, NFS, and a relational data store. For example, for AWS, this is managed via EFS.
Compute	Python/R/Scala workloads run on the CDSW gateway nodes of the cluster.	Python/R/Scala workloads run on the Cloudera Data Platform/cloud-provider-managed K8s cluster.
	CDSW pushes distributed compute workloads, such as Spark-on-YARN, to the CDH/HDP cluster.	Spark-on-YARN is not supported; Spark-on-K8s instead. Workloads will run on a dedicated K8s cluster provisioned within the customer environment.
	No autoscaling.	Autoscaling via your cloud service provider. Kubernetes/node-level autoscaling will be used to expand/contract the cluster size based on demand.
Packaging	Available as a downloadable RPM and CSD.	Available as a managed service on Cloudera Data Platform.
	Spark is packaged with CDH.	Spark on K8s is packaged with Cloudera AI - no dependency on an external cluster.
Data Access	Data usually resides on the attached CDH/HDP cluster in HDFS, Hive, HBase, and so on.	Data can reside on object storage such as S3 or any pre-existing workload clusters registered with Cloudera Data Platform.

Cloudera AI

Projects in Cloudera AI

Date published: 2020-07-16
Date modified: 2025-01-21

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Collaborating on Projects with Cloudera AI.....5
- Project Collaborators.....5
- Teams.....5
- ML Business User.....5
- Forking Projects.....6
- Collaborating with Git.....6

Sharing Job and Session Console Outputs.....6

Projects in Cloudera AI.....7
- Creating a Project with Legacy Engine Variants.....7
- Creating a Project with ML Runtimes variants.....8
- Creating a project from a password-protected Git repo.....10
- Configuring Project-level Runtimes.....11
- Adding Project Collaborators.....11
- Modifying Project Settings.....12
- Managing Project Files.....13
- Custom Template Projects.....14
- Deleting a Project.....14

Native Workbench Console and Editor.....15
- Launch a Session.....15
- Run Code.....16
- Access the Terminal.....17
- Stop a Session.....18
- Workbench editor file types.....18

Environmental Variables.....19

Third-party Editors.....19
- Modes of configuring third-party Editors.....20
- Configure a browser-based IDE as an Editor.....21
  - Test a browser-based IDE in a Session before installation.....22
  - Configure a browser-based IDE at the Project level.....22
  - Configure a browser-based IDE at Legacy Engine level.....23
- Configure a local IDE using an SSH gateway.....25
- Configure PyCharm as a local IDE.....26
  - Add Cloudera AI as an Interpreter for PyCharm.....26
  - Configure PyCharm to use Cloudera AI as the remote console.....27
  - (Optional) Configure the Sync between Cloudera AI and PyCharm.....27
- Configure VS Code as a local IDE.....28
  - Download cdswctl and add an SSH Key.....28
  - Initialize an SSH connection to Cloudera AI for VS code.....28
  - Setting up VS Code.....30

(Optional) Using VS Code with Python.....33
(Optional) Using VS Code with R.....35
(Optional) Using VS Code with Jupyter.....36
(Optional) Using VS Code with Git integration.....39
Limiting files in Explorer view.....39

Git for Collaboration.....40
Linking an existing Project to a Git remote.....40

Web Applications Embedded in Sessions.....41
Example: A Shiny Application.....42

Collaborating on Projects with Cloudera AI

This topic discusses all the collaboration strategies available to Cloudera AI users.

Project Collaborators

If you want to work closely with trusted colleagues on a particular project, you can add them to the project as collaborators. This is recommended for collaboration over projects created under your personal account. Anyone who belongs to your organization can be added as a project collaborator.

Project Visibility Levels: When you create a project in your personal context, Cloudera AI asks you to assign one of the following visibility levels to the project - Private or Public. Public projects on Cloudera AI grant read-level access to everyone with access to the Cloudera AI application. For Private projects, you must explicitly add someone as a project collaborator to grant them access.

Project Collaborator Access Levels: You can grant project collaborators the following levels of access: Viewer, Operator, Contributor, Admin

Note:

Collaborating Securely on Projects

Before adding project collaborators, you must remember that assigning the Contributor or Admin role to a project collaborator is the same as giving them write access to your data in CDH. This is because project contributors and project administrators have write access to all your project code (including any library code that you might not be actively inspecting). For example, a contributor/admin could modify project file(s) to insert code that deletes some data on the cluster. The next time you launch a session and run the same code, it will appear as though you deleted the data yourself.

Additionally, project collaborators also have access to all actively running sessions and jobs by default. This means that a malicious user can easily impersonate you by accessing one of your active sessions. Therefore, it is extremely important to restrict project access to trusted collaborators only. Note that site administrators can restrict this ability by allowing only session creators to run commands within their own active sessions.

For these reasons, Cloudera recommends using Git to collaborate securely on shared projects.

Teams

Users who work together on more than one project and want to facilitate collaboration can create a Team. Teams allow streamlined administration of projects. Team projects are owned by the team, rather than an individual user. Only users that are already part of the team can be added as collaborators to projects created within the team context. Team administrators can add or remove members at any time, assigning each member different access permissions.

Team Member Access Levels: You can grant team members the following levels of access: Viewer, Operator, Contributor, Admin.

ML Business User

The ML Business User role is for a user who only needs to view any applications that are created within Cloudera AI. This is the ideal role for an employee who is not part of the Data Science team and does not need higher-level access to workbenches and projects, but needs to access the output of a Data Science workflow. MLBusinessUser seats are available for purchase separately.

Forking Projects

You can fork another user's project by clicking Fork on the Project page. Forking creates a new project under your account that contains all the files, libraries, configurations, jobs, and dependencies between jobs from the original project.

Creating sample projects that other users can fork helps to bootstrap new projects and encourage common conventions.

Collaborating with Git

Cloudera AI provides seamless access to Git projects. Whether you are working independently, or as part of a team, you can leverage all of benefits of version control and collaboration with Git from within Cloudera AI. Teams that already use Git for collaboration can continue to do so. Each team member will need to create a separate Cloudera AI project from the central Git repository.

For anything but simple projects, Cloudera recommends using Git for version control. You should work on Cloudera AI the same way you would work locally, and for most data scientists and developers that means using Git.

Sharing Job and Session Console Outputs

This topic describes how to share the results of your research (that is, output from sessions and jobs) with teammates and project stakeholders.

Cloudera AI lets you easily share the results of your analysis with one click. Using rich visualizations and documentation comments, you can arrange your console log so that it is a readable record of your analysis and results. This log continues to be available even after the session stops. This method of sharing allows you to show colleagues and collaborators your progress without your having to spend time creating a report.

To share results from an interactive session, click Share at the top of the console page. From here you can generate a link that includes a secret token that gives access to that particular console output. For jobs results, you can either share a link to the latest job result or a particular job run. To share the latest job result, click the Latest Run link for a job on the Overview page. This link will always have the latest job results. To share a particular run, click on a job run in the job's History page and share the corresponding link.

You can share console outputs with one of the following sets of users.

- All anonymous users with the link - By default, Cloudera AI allows anonymous access to shared consoles. However, site administrators can disable anonymous sharing at any time.

  Once anonymous sharing has been disabled, all existing publicly shared console outputs will be updated to be viewable only by authenticated users.
- All authenticated users with the link - This means any user with a Cloudera AI account will have access to the shared console.
- Specific users and teams - Click Change to search for users and teams to give access to the shared console. You can also come back to the session and revoke access from a user or team the same way.

Sharing Data Visualizations

If you want to share a single data visualization rather than an entire console, you can embed it in another web page. Click the small circular 'link' button located to the left of most rich visualizations to view the HTML snippet that you can use to embed the visualization.

Projects in Cloudera AI

Projects form the heart of Cloudera AI. They hold all the code, configuration, and libraries needed to reproducibly run analyses. Each project is independent, ensuring users can work freely without interfering with one another or breaking existing workloads.

Access the Projects page by clicking Projects in the navigation panel. The Projects page gives you a quick summary of project information.

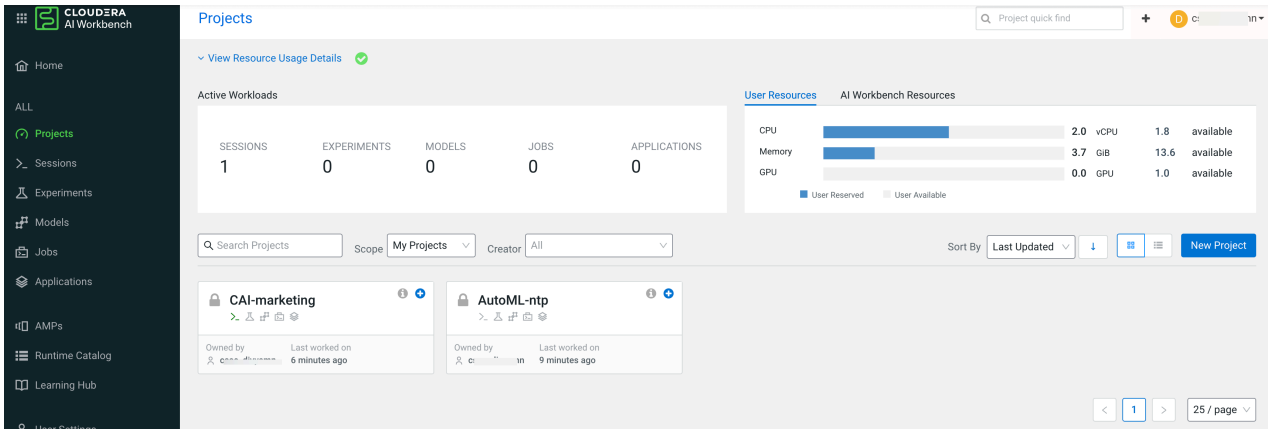

- Resource Usage Details - A collapsible section that displays resource usage.
  - Active Workloads - If there are active workloads running, this section describes the number of Sessions, Experiments, Models, Jobs, and Applications that are running.
  - User Resources and Workbench Resources
    - Click on the User Resources tab to see the CPU and memory resource usage for the user. The maximum usage of the vCPU and GB is calculated based on whether or not you have a quota. If you have a quota, the maximum usage will be based on your quota. If you do not have a quota, the maximum usage will be what is available on the cluster. If you have a GPU, you will also see the GPU usage.
    - Click on the Workbench Resources tab to see usage overall.
- Search Projects - Enter a term for keyword search across Project names.
- Scope - An additional filter only viewable by Administrators.
  - Selecting My Projects displays only the Projects that you have created or are a Collaborator of.
  - Selecting All Projects displays all Projects on the Cloudera AI Workbench.
- Creator - An additional filter to only display Projects created by a specified user.
- Projects View Selector - A setting that enables you to display Projects in a summary card-based view or a detailed table-based view.

The following topics describe how to create and manage projects in Cloudera AI.

Creating a Project with Legacy Engine Variants

Projects create an independent working environment to hold your code, configuration, and libraries for your analysis. This topic describes how to create a project with Legacy Engine variants in Cloudera AI.

Procedure

1. Go to Cloudera AI and on the left sidebar, click Projects.
2. Click New Project.

3. If you are a member of a team, from the drop-down menu, select the Account under which you want to create this project. If there is only one account on the deployment, you will not see this option.
4. Enter a Project Name.
5. Select Project Visibility from one of the following options.
   - Private - Only project collaborators can view or edit the project.
   - Team - If the project is created under a team account, all members of the team can view the project. Only explicitly-added collaborators can edit the project.
   - **Public** - All authenticated users of Cloudera AI will be able to view the project. Collaborators will be able to edit the project.
6. Under Initial Setup, you can either create a blank project, or select one of the following sources for your project files.
   - Built-in Templates - Template projects contain example code that can help you get started with Cloudera AI. They are available in R, Python, PySpark, and Scala. Using a template project is not required, but it helps you start using Cloudera AI right away.
   - Custom Templates - Site administrators can add template projects that are customized for their organization's use-cases. For details, see *Custom Template Projects*.
   - Local - If you have an existing project on your local disk, use this option to upload compressed files or folders to Cloudera AI.
   - Git - If you already use Git for version control and collaboration, you can continue to do so with Cloudera AI. Specifying a Git URL will clone the project into Cloudera AI. To use a password-protected Git repository, see *Creating a project from a password-protected Git repo*.
7. Click Create Project. After the project is created, you can see your project files and the list of jobs defined in your project.

   Note that as part of the project filesystem, Cloudera AI also creates the following .gitignore file.

```
R
node_modules
*.pyc
.*
!.gitignore
```

8. (Optional) To work with team members on a project, add them as collaborators to the project.

Related Information

Custom Template Projects

Adding a new SSH key to your GitHub account

Creating a project from a password-protected Git repo

Creating a Project with ML Runtimes variants

Projects create an independent working environment to hold your code, configuration, and libraries for your analysis. This topic describes how to create a project with ML Runtimes variants in Cloudera AI.

For Cloudera AI UI

1. Go to Cloudera AI and on the left sidebar, click Projects.
2. Click New Project.
3. If you are a member of a team, from the drop-down menu, select the Account under which you want to create this project. If there is only one account on the deployment, you will not see this option.
4. Enter a Project Name.

5. Select Project Visibility from one of the following options.
   - Private - Only project collaborators can view or edit the project.
   - Team - If the project is created under a team account, all members of the team can view the project. Only explicitly-added collaborators can edit the project.
   - Public - All authenticated users of Cloudera AI will be able to view the project. Collaborators will be able to edit the project.
6. Under Initial Setup, you can either create a blank project, or select one of the following sources for your project files.
   - Blank - The project will contain no information from a template, local file, or Git.
   - Templates - Template projects contain example code that can help you get started with Cloudera AI. They are available in R, Python, PySpark, and Scala. Using a template project is not required, but it helps you start using Cloudera AI right away.

     Note: Templates are designed for Engines and might not work with ML Runtimes.
   - Local - If you have an existing project on your local disk, use this option to upload compressed files or folders to Cloudera AI.
   - Git - If you already use Git for version control and collaboration, you can continue to do so with Cloudera AI. Specifying a Git URL will clone the project into Cloudera AI. To use a password-protected Git repository, see *Creating a project from a password-protected Git repo*.
7. If you would like to configure which Runtimes are available for this particular project, complete the following:
   - Scroll down to the Runtimes section and enter the appropriate information:
     - Note that the following Runtimes are configured by default:

       Runtime 1.

       - Kernel: The highest version of the Python kernel
       - Editor: PBJ Workbench and Jupyterlab
       - Edition: Standard & NVIDIA GPU
       - Version: The latest Runtime version

       Runtime 2.

       - Kernel: The highest version of the R kernel
       - Editor: PBJ Workbench
       - Edition: Standard
       - Version: The latest Runtime version
     - Use the **Advanced** view to add ML Runtimes based on a more detailed Editor, Kernel, Edition, and Version criteria.
     - Runtimes with the Enabled status and highest maintenance version will be configured as default settings for project creation as follows:
       - Upon new installation, runtimes will be calculated from variants recommended by Cloudera.
       - In cases other than new installations, variants that are set as default on the Runtime Catalog.
8. Click Create Project. After the project is created, you can see your project files and the list of jobs defined in your project.

   Note that as part of the project filesystem, Cloudera AI also creates the following .gitignore file.

```
R
node_modules
*.pyc
.*
!.gitignore
```

9. Set or verify the ML Runtimes settings for the project.

   Within the selected project, you can modify the default engine configuration:

   a. In the left navigation bar, click Project Settings.
   b. Select the Runtime/Engine tab.
   c. Next to Default Engine, select ML Runtimes.
   d. Click Save Engine.

   Note: The ability to switch the default engine is intended to support migration of projects from Engines to Runtimes. Cloudera does not recommend making this change frequently.

Once switched to ML Runtimes, all modules (Sessions, Jobs, Models, etc.) for the project will configure using ML Runtimes instead of Legacy Engines. Setting Engine parameters for the project will no longer be possible.

Existing and running instances (for example, Jobs, Models or Applications) previously configured with a particular Engine configuration will keep their configuration until you change the related settings. If you want to change the engine configuration for existing and running instances, you will need to update those based on the new, Runtime-based settings.

Note: Note that site administrators now have the ability to select ML Runtimes as a default for newly created projects. This setting can be found within the Admin menu below the Runtime/Engine tab.

For Cloudera AI APIv2

To create a project with ML Runtimes, follow this example:

```
project_body = cmlapi.CreateProjectRequest(
      name = "project_name",
      description = "project_description",
      default_project_engine_type = "ml_runtime",
      project_body.visibility = "public", # or "private" or "organization"
      template = "Python")
```

You also need to specify a runtime_identifier if this is used with an ml_runtime project. Obtain a list the runtimes with the following command:

```
client.list_runtimes()
```

For some more examples of commands related to projects, see: *Using the Projects API*.

Related Information

Using the Projects API

Creating a project from a password-protected Git repo

Creating a project from a password-protected Git repo

You can create projects in Cloudera AI by replicating the project files from a Git repo. The Git repo can be public, or it can be private, accessed by SSH or HTTPS authentication.

When you create a project, you can choose to create it from a Git repository. In the New Project page, under Initial Setup, choose the Git tab. Paste the Git URL in Git URL of Project.

There are two ways to authenticate a password-protected repo: SSH and HTTPS. The SSH key is automatically generated by Cloudera AI for each user or team account.

To clone a repo with SSH:

- Make sure you have a public SSH key added to your Github account. For more information, see *Adding an SSH Key to GitHub*.

- In the Git interface, select Code SSH and copy the URL.
- Paste the URL into Git URL of Project. It should appear similar to git@github.com:someuser/somerepo.git.

To clone a repo with HTTPS:

- In the Git interface, select Code SSH and copy the URL.
- Paste the URL into Git URL of Project.
- Insert the repo username and password into the URL, like so: https://<username>:<password>@github.com/someuser/somerepo.git

Continue with creating the project, and the files will be imported from the Git repository.

Configuring Project-level Runtimes

If you specified project-level Runtimes, you can view your chosen Runtime configuration by clicking Project Settings Runtime/Engine . Your chosen Runtimes are listed under Available Runtimes.

- If the Available Runtimes table is empty, users can select from all Runtimes available in the deployment to start new sessions or workloads.
- The filtering options in Project Settings only affect the user interface. The filtering options do not apply when projects are accessed using API v2.
- You can remove an available Runtime by selecting the corresponding "x" in the right most column. Runtimes can only be removed if there are no active workloads using them.
- You can add additional Runtimes by clicking Add Runtime and chosing additional Runtimes.
- If there is a newer version of a Runtime, a warning icon displays next to the appropriate Runtimes. You can apply the latest version by clicking Add Latest. The older Runtime version is not removed from the table. However, you can remove it by selecting the x" in the right most column.
- The Available Runtimes table shows related counters for non-interactive workloads.

Adding Project Collaborators

This topic shows you how to invite colleagues to collaborate on a project.

About this task

For a project created under your personal account, anyone who belongs to your organization can be added as a collaborator. For a project created under a team account, you can only add collaborators that already belong to the team. If you want to work on a project that requires collaborators from different teams, create a new team with the required members, and then create a project under that account. If your project was created from a Git repository, each collaborator must create the project from the same central Git repository.

You can grant project collaborators one of three levels of access:

- Viewer - Read-only access to code, data, and results.
- Operator - Read-only access to code, data, and results. Additionally, Operators can start and stop existing jobs in the projects that they have access to.
- Contributor - Can view, edit, create, and delete files and environmental variables, run sessions/experiments/jobs/models and run code in running jobs. Additionally, Contributors can set the default engine for the project.
- Admin - Has complete access to all aspects of the project. This includes the ability to add new collaborators, and delete the entire project.

Adding Teams as collaborators

Any administrator, owner, contributor or operator can add a team as a collaborator on a project. They can be added to team or individual projects. When a team is added, if there are members of the team who are already members of the project, their permissions can be upgraded to the permissions they have on the team (Inherit mode). Alternatively,

the permissions applied to the team are applied equally to each member on the team, even if they have different permissions independently on the project (Override mode).

Note:

Collaborating Securely on Projects

Before adding project collaborators, you must remember that assigning the Contributor or Admin role to a project collaborator is the same as giving them write access to your data in CDH. This is because project contributors and project administrators have write access to all your project code (including any library code that you might not be actively inspecting). For example, a contributor/admin could modify project file(s) to insert code that deletes some data on the CDH cluster. The next time you launch a session and run the same code, it will appear as though you deleted the data yourself.

Additionally, project collaborators also have access to all actively running sessions and jobs. This means that a malicious user can easily impersonate you by accessing one of your active sessions. Therefore, it is extremely important to restrict project access to trusted collaborators only. Note that site administrators can restrict this ability by allowing only session creators to run commands within their own active sessions.

For these reasons, Cloudera recommends using Git to collaborate securely on shared projects. This will also help avoid file modification conflicts when your team is working on more elaborate projects.

Procedure

1. In Cloudera AI, navigate to the project overview page.
2. Click Collaborators.
3. Search for collaborators by either name, team name, or email address and click Add.

Modifying Project Settings

Project contributors and administrators can modify aspects of the project environment such as the engine or ML Runtimess used to launch sessions, the environment variables, and to create SSH tunnels to access external resources.

Procedure

1. Switch context to the account where the project was created.
2. Click Projects.
3. From the list of projects, select the one to modify.

4. Click Project Settings to open the Project Settings dashboard.

General

Modify the project name, description, visibility (privacy settings), and owner on this page. For Project Owner, only individual users (not Teams) can be assigned, and only a Administrator can change the owner.

Note: After the owner is changed, the previous owner will not be able to see the project, unless they are added as a Collaborator.

Runtime/Engine

Cloudera AI ensures that your code is always run with the specific engine version you selected. You can select to use either Legacy Engines or ML Runtimes. For legacy engines only, you can also select the engine version and add third-party editors here.

Advanced

- Environment Variables - If there are any environmental variables that should be injected into all the engines running this project, you can add them to this page. For more details, see *Engine Environment Variables*.
- Ephemeral Storage Settings - Ephemeral storage space is scratch space a Cloudera AI session, job, application or model can use. For more information, see *Ephemeral storage*.
- Shared Memory Limit - You can specify additional shared memory available to sessions running with the project.

Note: You can specify additional shared memory available to sessions running with the project. The maximum size of this volume is the half of your physical RAM in the node, not including memory used for swap.

SSH Tunnels

In some environments, external databases and data sources reside behind restrictive firewalls. Cloudera AI provides a convenient way to connect to such resources using your SSH key. For instructions, see *SSH Keys*.

Data Connections

Data Connections connect projects to data sources. For more information, see *Data connection management*.

Delete Project

This page can only be accessed by project administrators. Remember that deleting a project is irreversible. All files, data, sessions, and jobs are removed.

Related Information

Managing Engines
Engine Environment Variables
SSH Keys
Data connection management
Ephemeral storage

Managing Project Files

Cloudera AI allows you to move, rename, copy, and delete files within the scope of the project where they live. You can also upload new files to a project, or download project files. For use cases beyond simple projects, Cloudera strongly recommends using *Git for Collaboration* to manage your projects using version control.

Procedure

1. Switch context to the account where the project was created.

2. Click Projects.
3. From the list of projects, click on the project you want to modify. This will take you to the project overview.
4. Click Files.

   Upload Files to a Project

   Files can only be uploaded within the scope of a single project. Therefore, to access a script or data file from multiple projects, you will need to manually upload it to all the relevant projects.

   Click Upload. Select Files or Folder from the dropdown, and choose the files or folder you want to upload from your local filesystem.

   In addition to uploading files or a folder, you can upload a .tar file of multiple files and folders. After you select and upload the .tar file, you can use a terminal session to extract the contents:

   a. On the project overview page, click Open Workbench and select a running session or create a new one.
   b. Click Terminal access.
   c. In the terminal window, extract the contents of the .tar file:

      ```
      tar -xvf <file_name>.tar.gz
      ```

      The extracted files are now available for the project.

   Download Project Files

   Click Download to download the entire project in a .zip file. To download only a specific file, select the checkbox next to the file(s) to be download and click Download.
5. You can also use the checkboxes to Move, Rename, or Delete files within the scope of this project.

Related Information

Git for Collaboration

Custom Template Projects

Site administrators can add template projects that have been customized for their organization's use-cases. These custom project templates can be added in the form of a Git repository.

Required Role: See User Role Authorization.

To add a new template project, go to Site Administration Settings . Under the Project Templates section, provide a template name, the URL to the project's Git repository, and click Add.

The added templates will become available in the Template tab on the Create Project page. Site administrators can add, edit, or delete custom templates, but not the built-in ones. However, individual built-in templates can be disabled using a checkbox in the Project Templates table at Admin Settings .

Deleting a Project

This topic demonstrates how to delete a project.

About this task

Important: Deleting a project is an irreversible action. All files, data, and history related to the project will be lost. This includes any jobs, sessions or models you created within the project.

Procedure

1. Go to the project Overview page.

2. On the left sidebar, click Project Settings.
3. Go to the Delete Project.
4. Click Delete Project and click OK to confirm.

Native Workbench Console and Editor

The workbench console provides an interactive environment tailored for data science, supporting R, Python and Scala. It currently supports R, Python, and Scala engines. You can use these engines in isolation, as you would on your laptop, or connect to your CDH cluster.

The workbench UI includes four primary components:

- An editor where you can edit your scripts.
- A console where you can track the results of your analysis.
- A command prompt where you can enter commands interactively.
- A terminal where you can use a Bash shell.

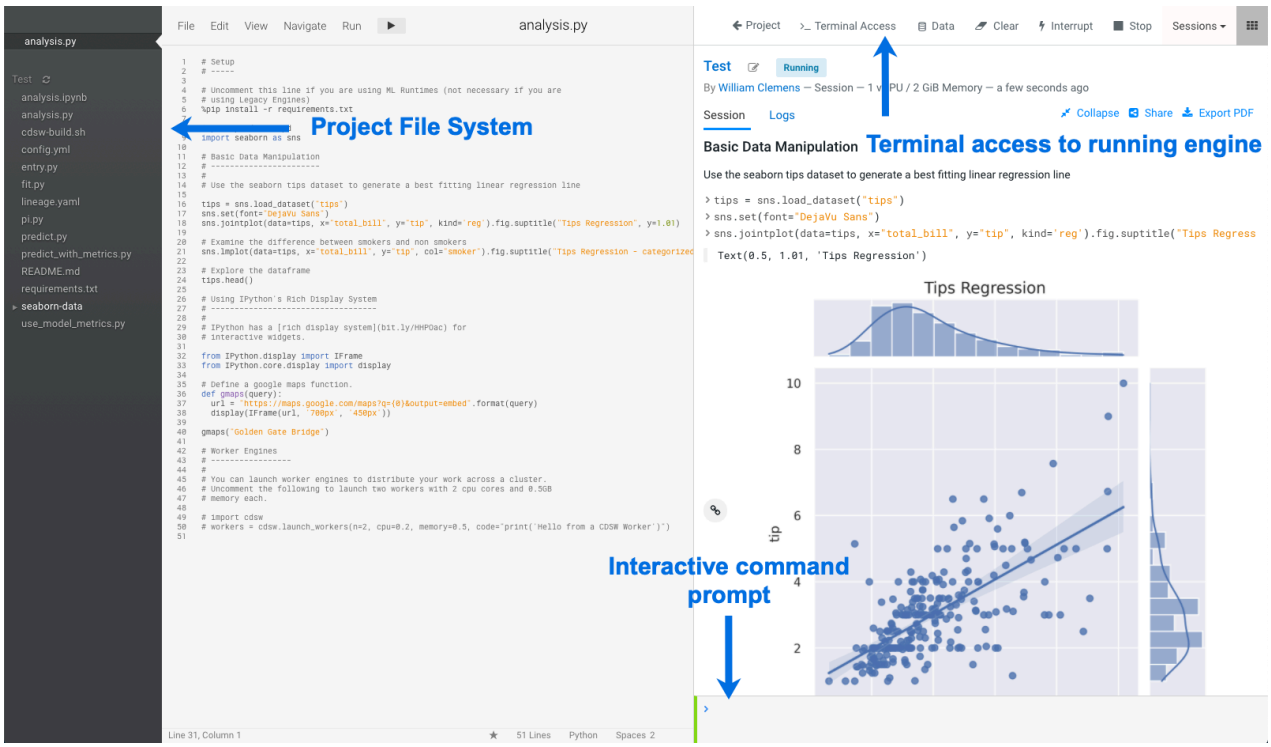

Typically, you would use the following steps to run a project in the workbench:

Related Information

Managing Engines

Launch a Session

Sessions allow you to perform actions such as run R or Python code. They also provide access to an interactive command prompt and terminal. This topic demonstrates how to launch a new session.

Procedure

1. Navigate to your project's Overview page.

2. Click New Session.

   New projects now default to using ML Runtimes. Legacy Engines are deprecated in the current release. However, you can change the default engine later in this task.

3. Check the settings for your session:

   If your project is using ML Runtimes, you will see the following settings:

   Editor

   Selects the Editor; currently only Workbench is supported and therefore the selector is static.

   Kernel

   Selects the Kernel, for example Python 3.7, R4.0.

   Edition

   Selects the Runtime Edition. Initially only Standard variants are supported.

   Version

   Selects the ML Runtimes version.

   Note: The selector options only consider the configurations supported by the actual deployments and certain selections will automatically limit others. For example, certain versions are only relevant for Python or certain editors are supported only with certain kernels.

   If your project is using Legacy Engines, you see the following settings:

   Editor

   Selects the Editor; currently only Workbench is supported and therefore the selector is static.

   Kernel

   Selects the Kernel. Initially only Python Runtimes are supported.

   Engine Image

   Displays the Advanced tab in Project Settings and allows you to set environment variables and the shared memory limit.

4. If your project is using Legacy Engines, you can modify the engine image used by this session:

   a) By Engine Image, click Configure.

      Cloudera AI displays the Project Settings page.

   b) Select the Runtime/Engine tab.

   c) Next to Default Engine, select ML Runtime or Legacy Engine.

   d) Click Save Engine.

5. Specify your Resource Profile.

   This attribute will define how many vCPUs and how much memory will be reserved to run the workload (for example, session including the runtime itself). The minimum configuration is 1vCPU and 2 GB memory.

6. Click Start Session.

   The command prompt at the bottom right of your browser window will turn green when the engine is ready. Sessions typically take between 10 and 20 seconds to start.

Run Code

This topic shows you how to enter and run code in the interactive Workbench command prompt or the editor after you launch a session. The editor can be best used for codes you want to keep, while the command prompt is an optimal choice for quick interactive exploration.

Command Promt

Enter a command and press Enter to run it. If you want to enter more than one line of code, use Shift+Enter to move to the next line. The output of your code, including plots, appears in the console.

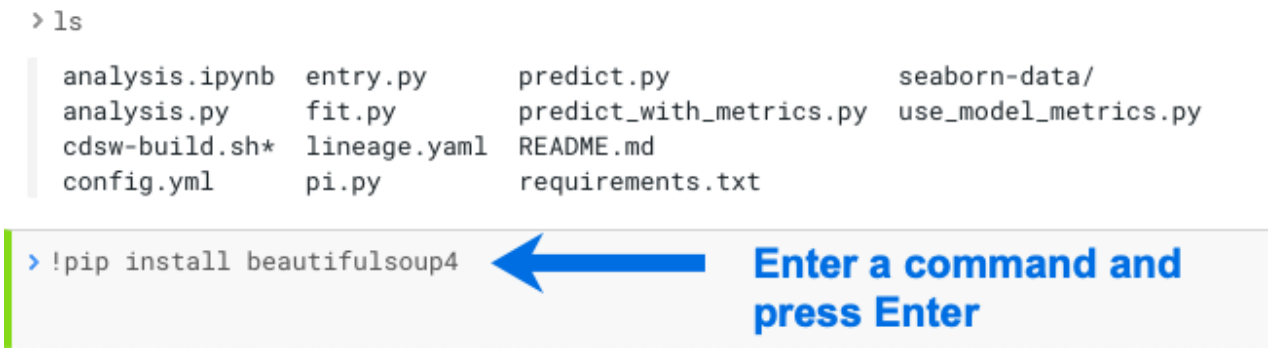

If you created your project from a template, you can see the project files in the editor. You can open a file in the editor by clicking the file name in the file navigation bar on the left.

Editor

To run code from the editor:

1. Select a script from the project files on the left sidebar.
2. To run the whole script click 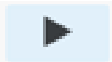 on the top navigation bar, or, highlight the code you want to run and press Ctrl+Enter (Windows/Linux) or cmd+Enter (macOS).

When working on an analysis, writing and executing your code from the editor rather than the command prompt enables to iteratively develop your code and save it along the way.

If you require more space for your editor, you can collapse the file list by double-clicking between the file list pane and the editor pane. You can hide the editor, using the editor's View menu.

Code Autocomplete

The Python and R kernels include support for automatic code completion, both in the editor and the command prompt. Use **single tab** to display suggestions and **double tab** for autocomplete.

Project Code Files

All project files are stored to persistent storage within the respective project directory at /var/lib/cdsw/current/projects. They can be accessed within the project in a typical directory structure. For example, you can import functions from one file to another within the same project.

Access the Terminal

Cloudera AI provides full terminal access to running engines from the web console. This topic show you how to access the Terminal from a running Workbench session.

You can use the terminal to move files around, run Git commands, access the YARN and Hadoop CLIs, or install libraries that cannot be installed directly from the engine. To access the Terminal from a running session, click Terminal Access above the session log pane.

```
Kernel: python2

Project workspace: /home/cdsw

Kerberos principal:

Runtimes:
  R: R version 3.4.1 (--) -- "Single Candle"
  Python 2: Python 2.7.11
  Python 3: Python 3.6.1
  Java: java version "1.8.0_144"

cdsw@frczgqdokx4k67un:~$
```

The terminal's default working directory is /home/cdsw, which is where all your project files are stored. Any modifications you make to this folder will persist across runs, while modifications to other folders are discarded.

If you are using Kerberos authentication, you can run klist to see your Kerberos principal. If you run hdfs dfs -ls you will see the files stored in your HDFS home directory.

Note that the terminal does not provide root or sudo access to the container. To install packages that require root access, see *Customized Engine Images*.

Related Information

Customized Engine Images

Stop a Session

This topic demonstrates how to stop a session to free up resources for other users when you are finished.

When you are done with the session, click Stop in the menu bar above the console, or use code to exit by typing the following command:

R

```
quit()
```

Python

```
exit
```

Scala

```
quit()
```

Sessions automatically stop after an hour of inactivity.

Workbench editor file types

The default workbench editor supports the following file types:

- Text
- CSS
- HTML
- JavaScript
- JSON
- PHP

- Scala
- C++
- C#
- CLike
- Java
- CoffeeScript
- R
- Julia
- Ruby
- Clojure
- Perl
- Python
- SASS
- Lua
- SQL
- Diff
- Markdown
- YAML
- Haxe

Environmental Variables

Environmental variables help you customize engine environments, both globally and for individual projects/jobs.

For example, if you need to configure a particular timezone for a project or increase the length of the session or job timeout windows, you can use environmental variables to do so.

For a list of the environmental variables you can configure and instructions on how to configure them, see *Engine Environment Variables*.

You can create new environment variables in the following pages:

- Project Settings Advanced
- New Job Job Settings
- New Application Application Settings
- User Settings Environment Variables
- Site Administration Runtime

To add an environmental variable:

1. Enter the name and value.
2. Click Add.
3. Optionally, click Hide/Show to hide the value from viewing by users. This can help with hiding sensitive values stored in environment variables, such as passwords. The value is set to Hide by default.

Third-party Editors

In addition to the built-in Cloudera AI editor, you can configure Cloudera AI to work with third-party, browser-based IDEs such as Jupyter and also certain local IDEs that run on your machine, such as PyCharm.

Note: Custom editors run inside Cloudera AI sessions. If the Cloudera AI session is stopped, this may cause unexpected behavior in the editor UI and, in some cases, may result in data loss. You should, therefore, use the custom editor's UI to shut the editor down first. This will automatically end the Cloudera AI session too.

In JupyterLab you do that by clicking "Shut Down" in the JupyterLab "File" menu." This applies to both engines and Runtimes, and all versions of Cloudera AI.

When you bring your own editor, you still get many of the benefits Cloudera AI behind an editor interface you are familiar with:

- Dependency management that lets you share code with confidence
- CDH client configurations
- Automatic Kerberos authentication through Cloudera AI
- Reuse code in other Cloudera AI features such as experiments and jobs
- Collaboration features such as teams
- Compliance with IT rules for where compute, data, and/or code must reside. For example, compute occurs within the Cloudera AI deployment, not the local machine. Browser IDEs run within a Cloudera AI session and follow all the same compliance rules. Local IDEs, on the other hand, can bring data or code to a user's machine. Therefore, Site Administrators can opt to disable local IDEs to balance user productivity with compliance concerns.

In the Cloudera AI documentation, browser-based IDEs like Jupyter will be referred to as "browser IDEs". IDEs such as PyCharm that run on your machine outside of your browser will be referred to as "local IDEs" because they run on your local machine. You can use the browser or local IDE of your choice to edit and run code interactively.

Note that you can only edit and run code interactively with the IDEs. Tasks such as creating a project or deploying a model require the Cloudera AI web UI and cannot be completed through an editor.

Modes of configuring third-party Editors

The configuration for an Integrated Development Environment (IDE) depends on which type of editor you want to use.

In addition to the native Cloudera AI editor, you can configure Cloudera AI to work with third-party, browser-based IDEs, such as Jupyter, and also certain local IDEs that run on your machine, such as PyCharm.

Workbench editor

The Workbench editor is the built-in editor for Cloudera AI. No additional configuration is required to use it. When you launch a session, select the Workbench editor.

Third-party, browser-based IDEs

Browser IDEs are editors such as Jupyter or RStudio. When you use a browser IDE, it runs within a session and allows you to edit and run code interactively. Changes that you make in the editor are propagated to the Cloudera AI project. Base Engine Image v8 and higher ships with Jupyter preconfigured as a browser IDE. You can select it when you start a session or add a different browser IDE. For more information, see *Configure a Browser IDE as an Editor*.

Keep the following in mind when using browser IDEs:

- Engine Version Requirements
  - Browser-based IDEs require Base Engine Image v8 or higher.
- When you are finished using a browser IDE, you must exit the IDE properly, including saving your work if necessary. Do not just stop the Cloudera AI session. Doing so will cause you to lose your session state. For example, if you want RStudio to save your state, including variables, to ~/.RData, exit the RStudio workbench using the power button in the top right of the RStudio UI.
- Depending on the behavior of the browser IDE, multiple users within a project may overwrite each other's state. For example, RStudio state is persisted in /home/cdsw/.RData that is shared by all users within a project.

- Browser IDEs do not adhere to the timeout set in IDLE_MAXIMUM_MINUTES. Instead, they use the timeout set in SESSION_MAXIMUM_MINUTES, which is 7 days by default. Cloudera recommends that users stop their session manually after using a browser-based editor. Running sessions continue to consume resources and may impact other users.
- Logs for browser IDEs are available on the Logs tab of the session window. This includes information that the IDE may generate, such as error messages, in addition to any Cloudera AI logs.

Local IDE Editors on your machine that can use SSH-based remote editing

These editors, referred to as Local IDEs in the documentation, are editors such as PyCharm that run on your local machine. They connect to Cloudera AI with an SSH endpoint and allow you to edit and run code interactively. You must manually configure some sort of file sync and ignore list between your local machine and Cloudera AI. You can use functionality within the local IDE, such as PyCharm's sync, or external tools that can sync via the SSH endpoint, such as Mutagen.

Keep the following in mind before setting up local IDEs:

- Local IDEs do not require a specific engine image, but Cloudera always recommends you use the latest engine image.
- Site Administrators should work with IT to determine the data access policies for your organization. For example, your data policy may not allow users to sync certain files to their machines from Cloudera AI. Verify that users understand the requirements and adhere to them when configuring their file sync behavior.
- Users should ensure that any IDEs that the IDEs they want to use support SSH. For example, VS Code supports "remote development over SSH," and PyCharm supports using a "remote interpreter over SSH."

Related Information

Configure a browser-based IDE as an Editor
Configure a local IDE using an SSH gateway

Configure a browser-based IDE as an Editor

When you use a browser-based Integrated Development Environment (IDE), changes that you make in the editor are propagated to the Cloudera AI project.

About this task

For example, if you create a new .py file or modify an existing one with the third-party editor, the changes are propagated to Cloudera AI. When you run the code from the IDE, execution is pushed from the IDE to Cloudera AI.

Base Engine Image v8 and higher for Cloudera AI comes preconfigured with Jupyter, and any browser IDEs you want to add must be added to Base Engine Image v8 or higher. Jupyter can be selected in place of the built-in Workbench editor when you launch a session, and no additional configuration is required. You can configure additional IDEs to be available from the dropdown.

You have two configuration options:

- Project Level: You can configure an editor at the project level so that any session launched within that project can use the editor configured. Other projects across the deployment will not be able to use any editors configured in such a manner. For steps, see *Configure a Browser IDE at the Project Level*.
- Engine Level: You can create a custom engine configured with the editor so that any project across the deployment that uses this custom engine can also use the editor configured. This might be the only option in case of certain browser IDEs (such as RStudio) that require root permission to install and therefore cannot be directly installed within the project. For steps, see Configure a Browser IDE at the Engine Level.

Cloudera recommends you first test the browser IDE you intend to install in a session before you install it to the project or build a custom engine with it. For steps, see *Test a Browser IDE in a Session Before Installation*.

Test a browser-based IDE in a Session before installation

This process can be used to ensure that a browser-based Integrated Development Environment (IDE) works as expected before you install it to a project or to a customized engine image. This process is not meant for browser-based IDEs that require root permission to install, such as RStudio.

About this task

These steps are only required if you want to use an editor that does not come preinstalled as part of the default engine image. Perform the following steps to configure an editor for your session:

Procedure

1. Ensure that your browser accepts pop-up windows and cookies from Cloudera AI web UI.
2. Open the Cloudera AI web UI.
3. Go to your project and launch a session with the kernel of your choice and the Workbench editor. Alternatively, open an existing session.
4. In the interactive command prompt or terminal for the session, install the editor you want to use. See the documentation for your editor for specific instructions.

   For example:

   Jupyter Lab

   Python 3

   The following example command installs Jupyter Lab for Python 3:

   ```
   !pip3 install jupyterlab
   ```

5. After the installation completes, enter the command to start the server for the notebook on the port specified in the CDSW_APP_PORT environment variable on IP address 127.0.0.1.

   For example, the following command starts the server for Jupyter Lab on the port specified in the CDSW_APP_PORT environment variable:

   ```
   !/home/cdsw/.local/bin/jupyter-lab --no-browser --ip=127.0.0.1 --port=${
   CDSW_APP_PORT} --NotebookApp.token= --NotebookApp.allow_remote_access=True
    --log-level=ERROR
   ```

6. Click on the grid icon in the top right.
   You should see the editor in the drop-down menu. If you select the editor, it opens in a new browser tab.

Configure a browser-based IDE at the Project level

The following steps are only required if you want to use an editor that is not included in the default engine image that ships with Cloudera AI.

Before you begin

Before you start, verify that you have installed the browser-based Integrated Development Environment (IDE) of your choice to the project. For information about how to install additional packages to a project, see *Installing Additional Packages*.

About this task

Perform the following steps to add an editor to a project:

Procedure

1. Open the Cloudera AI web UI.
2. Go to the project you want to configure an editor for.
3. Go to Settings Editors and click New Editor.

4. Complete the fields:
   - Name: Provide a name for the editor. This is the name that appears in the dropdown menu for Editors when you start a new session.
   - Command: Enter the command to start the server for the editor on the Cloudera AI public port specified in the CDSW_APP_PORT environment variable (default 8081).

     For example, the following command starts Jupyter Lab on the port specified by the CDSW_APP_PORT environment variable:

     ```
     /home/cdsw/.local/bin/jupyter-lab --no-browser --ip=127.0.0.1 --port=${C
     DSW_APP_PORT} --NotebookApp.token= --NotebookApp.allow_remote_access=Tru
     e --log-level=ERROR
     ```

     This is the same command you used to start the IDE to test it in a session.
5. Save the changes.
   When a user starts a new session, the editor you added is available in the list of editors. Browsers must be configured to accept cookies and allow pop-up windows from the Cloudera AI web UI.

Related Information

Installing Additional Packages

Configure a browser-based IDE at Legacy Engine level

You can make a browser-based Integrated Development Environment (IDE) available to any project within a Cloudera AI deployment by creating a customized legacy engine image, installing the editor to it, and adding it to the trusted list for a project. Additionally, browser IDEs that require root permission to install, such as RStudio, can only be used as part of a customized legacy engine image.

About this task

When a user launches a session, they can select the customized legacy engine with the editors available. The following steps describe how to make a customized legacy engine image for RStudio:

Procedure

1. Create a Dockerfile for the new custom image. Note that the Base Engine Image uses Ubuntu, and you must use Base Engine Image v9 or higher.

   The following sample Dockerfile is for RStudio:

   ```
   #Dockerfile
   FROM docker.repository.cloudera.com/cdsw/engine:9-cml1.1

   WORKDIR /tmp

   #Delete the Cloudera repository that is inaccessible because of the payw
   all
   RUN rm /etc/apt/sources.list.d/*

   #The RUN commands that install an editor
   #For example: RUN apt-get install myeditor

   RUN apt-get update && apt-get dist-upgrade -y && \
       apt-get install -y --no-install-recommends \
       libapparmor1 \
       libclang-dev \
       lsb-release \
       psmisc \
       sudo
   ```

```
#The command that follows RUN is the same command you used to install the
 IDE to test it in a the session.
RUN wget https://download2.rstudio.org/server/trusty/amd64/rstudio-server-
1.2.1335-amd64.deb && \
    dpkg -i rstudio-server-1.2.1335-amd64.deb

COPY rserver.conf /etc/rstudio/rserver.conf

COPY rstudio-cdsw /usr/local/bin/rstudio-cdsw

RUN chmod +x /usr/local/bin/rstudio-cdsw
```

2. Create rserver.conf:

```
# Must match CDSW_APP_PORT
www-port=8090
server-app-armor-enabled=0
server-daemonize=0
www-address=127.0.0.1
auth-none=1
auth-validate-users=0
```

Make sure that the www-port property matches the port set in the CDSW_APP_PORT environment variable (default 8090).

3. Create rstudio-cdsw:

```
#!/bin/bash
# This saves RStudio's user runtime information to /tmp, which ensures
several
# RStudio sessions can run in the same project simultaneously
mkdir -p /tmp/rstudio/sessions/active
mkdir -p /home/cdsw/.rstudio/sessions
if [ -d /home/cdsw/.rstudio/sessions/active ]; then rm -rf /home/cdsw/.rst
udio/sessions/active; fi
ln -s /tmp/rstudio/sessions/active /home/cdsw/.rstudio/sessions/active
# This ensures RStudio picks up the environment. This may not be necess
ary if
# you are installing RStudio Professional. See
# https://docs.rstudio.com/ide/server-pro/r-sessions.html#customizing-sess
ion-launches.
# SPARK_DIST_CLASSPATH is treated as a special case to workaround a bug in
 R
# with very long environment variables.
env | grep -v ^SPARK_DIST_CLASSPATH >> /usr/local/lib/R/etc/Renviron.site
echo "Sys.setenv(\"SPARK_DIST_CLASSPATH\"=\"${SPARK_DIST_CLASSPATH}\")"
 >> /usr/local/lib/R/etc/Rprofile.site

# Now start RStudio
/usr/sbin/rstudio-server start
```

4. Build the Dockerfile:

```
docker build -t <image-name>:<tag> . -f Dockerfile
```

If you want to build your image on a Cloudera AI Workbench, you must add the --network=host option to the build command:

```
docker build --network=host -t <image-name>:<tag> . -f Dockerfile
```

5. Distribute the image:
   - Push the image to a public registry such as DockerHub.

     For instructions, refer the Docker documentation.
   - Push the image to your company's Docker registry.

     When using this method, make sure to tag your image with the following schema:

     ```
     docker tag <image-name> <company-registry>/<user-name>/<image-name>:
     <tag>
     ```

     Once the image has been tagged properly, use the following command to push the image:

     ```
     docker push <company-registry>/<user-name>/<image-name>:<tag>
     ```
6. Add the image to the trusted list in Cloudera AI:
   a) Log in to the Cloudera AI web UI as a site administrator.
   b) Click Admin Engines .
   c) Add <company-registry>/<user-name>/<image-name>:<tag> to the list of trusted engine images.
7. Add the new legacy engine to the trusted list for a project:
   a) Go to the project Settings page.
   b) Click Engines.
   c) Select the new customized legacy engine from the dropdown list of available Docker images. Sessions and jobs you run in your project will have access to this engine.
8. Configure RStudio for the project. When this is done, you will be able to select RStudio from the dropdown list of editors on the Launch New Sesssion page.
   a) Go to Settings > Editors and click New Editor.
   b) Complete the fields:
      - Name: Provide a name for the editor. For example, RStudio. This is the name that appears in the dropdown menu for Editors when you start a new session.
      - Command: Enter the command to start the server for the editor.

        For example, the following command will start RStudio:

        ```
        /usr/local/bin/rstudio-cdsw
        ```
   c) Save the changes.

Related Information

Docker push

Limitations

Configure a local IDE using an SSH gateway

The specifics on how to configure a local IDE to work with Cloudera AI are dependent on the local IDE you want to use.

Cloudera AI relies on the SSH functionality of the editors to connect to the SSH endpoint on your local machine created with the cdswctl client. Users establish an SSH endpoint on their machine with the cdswctl client. This endpoint acts as the bridge that connects the editor on your machine and the Cloudera AI deployment.

The following steps are a high-level description of the steps a user must complete:

1. Establish an SSH endpoint with the Cloudera AI CLI client. See Initialize an SSH Endpoint.
2. Configure the local IDE to use Cloudera AI as the remote interpreter.

3. Optionally, sync files with tools (like mutagen, SSHFS, or the functionality built into your IDE) from Cloudera AI to your local machine. Ensure that you adhere to IT policies.
4. Edit the code in the local IDE and run the code interactively on Cloudera AI.
5. Sync the files you edited locally to Cloudera AI.
6. Use the Cloudera AI web UI to perform actions such as deploying a model that uses the code you edited.

You can see an end-to-end example for PyCharm configuration in the *Configure Pycharm as a Local IDE*.

Related Information

Configure PyCharm as a local IDE

Configure PyCharm as a local IDE

Cloudera AI supports using editors on your machine that allow remote execution and/or file sync over SSH, such as PyCharm.

About this task

This topic describes the tasks you need to perform to configure Cloudera AI to act as a remote SSH interpreter for PyCharm. Once finished, you can use PyCharm to edit and sync the changes to Cloudera AI. To perform actions such as deploying a model, use the Cloudera AIg web UI.

Note: These instructions were written for the Professional Edition of PyCharm Version 2019.1. See the documentation for your version of PyCharm for specific instructions.

Before you begin, ensure that the following prerequisites are met:

- You have an edition of PyCharm that supports SSH, such as the Professional Edition.
- You have an SSH public/private key pair for your local machine.
- You have Contributor permissions for an existing Cloudera AI project. Alternatively, create a new project you have access to.

Add Cloudera AI as an Interpreter for PyCharm

In PyCharm, you can configure an SSH interpreter. Cloudera AI uses this method to connect to PyCharm and act as its interpreter.

About this task

Before you begin, ensure that the SSH endpoint for Cloudera AI is running on your local machine. These instructions were written for the Professional Edition of PyCharm Version 2019.1 and are meant as a starting point. If additional information is required, see the documentation for your version of PyCharm for specific instructions.

Procedure

1. Open PyCharm.
2. Create a new project.
3. Expand Project Interpreter and select Existing interpreter.
4. Click on ... and select SSH Interpreter
5. Select New server configuration and complete the fields:
   - Host: localhost
   - Port: 2222
   - Username: cdsw

6. Select Key pair and complete the fields using the RSA private key that corresponds to the public key you added to the Remote Editing tab in the Cloudera AI web UI.

   For macOS users, you must add your RSA private key to your keychain. In a terminal window, run the following command:

   ```
   ssh-add -K <path to your prviate key>/<private_key>
   ```

7. Complete the wizard. Based on the Python version you want to use, enter one of the following parameters:

   - /usr/local/bin/python2
   - /usr/local/bin/python3

   You are returned to the New Project window. Existing interpreter is selected, and you should see the connection to Cloudera AI in the Interpreter field.

8. In the Remote project location field, specify the following directory:

   ```
   /home/cdsw
   ```

9. Create the project.

Configure PyCharm to use Cloudera AI as the remote console

Procedure

1. In your project, go to Settings and search for Project Interpreter.

   Depending on your operating system, Settings may be called Preferences.

2. Click the gear icon and select Show All.
3. Select the Remote Python editor that you added, which is connected to the Cloudera AI deployment.
4. Add the following interpreter path by clicking on the folder icon:

   ```
   /usr/local/bin/python2.7/site-packages
   ```

(Optional) Configure the Sync between Cloudera AI and PyCharm

Configuring what files PyCharm ignores can help you adhere to IT policies.

About this task

Before you configure syncing behavior between the remote editor and Cloudera AI, ensure that you understand the policies set forth by IT and the Site Administrator. For example, a policy might require that data remains within the Cloudera AI deployment but allow you to download and edit code.

Procedure

1. In your project, go to Settings and search for Project Interpreter.

   Depending on your operating system, Settings may be called Preferences.

2. Search for Deployment.
3. On the Connection tab, add the following path to the Root path field:

   ```
   /home/cdsw
   ```

4. Optionally, add a Deployment path on the Mappings tab if the code for your Cloudera AI project lives in a subdirectory of the root path.

5. Expand Deployment in the left navigation and go to Options Upload changed files automatically to the default server and set the behavior to adhere to the policies set forth by IT and the Site Administrator.

   Cloudera recommends setting the behavior to Automatic upload because the data remains on the cluster while your changes get uploaded.
6. Sync for the project file(s) to your machine and begin editing.

Configure VS Code as a local IDE

Follow the guidelines for configuring VS Code as a local Integrated Development Environment (IDE).

About this task

Cloudera AI supports using local IDEs on your machine that allow remote execution and/or file sync over SSH, such as VS Code. This topic describes the tasks you need to perform to configure Cloudera AI to act as a remote SSH interpreter for VS Code. Once finished, you can use VS Code to edit and syncronise the changes to Cloudera AI. To perform actions such as deploying a model, use the Cloudera AI web UI.

Before you begin, ensure that the following prerequisites are met:

- You have an edition of VS Code that supports SSH.
- You have an SSH public/private key pair for your local machine that is compatible with VS Code.
- You have Contributor permissions for an existing Cloudera AI project. Alternatively, create a new project you have access to.

Download cdswctl and add an SSH Key

The first step to configure VS Code as a local IDE is to download cdswctl and add an SSH key.

Procedure

1. Open the Cloudera AI web UI and go to User Settings Remote Editing for your user account.
2. Download the CLI client for your operating system.
3. In the terminal, run cat ~/.ssh/id_rsa.pub. If you used a different filename above when generating the key, use that filename instead. This command prints the key as a string.
4. Copy the key. It should resemble the following: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCha2J5mW3i3BgtZ25/FOsxywpLVkx1RgmZunI
5. In SSH public keys for session access, paste the key.

What to do next

Cloudera AI uses the SSH public key to authenticate your CLI client session, including the SSH endpoint connection to the Cloudera AI deployment. Any SSH endpoints that are running when you add an SSH public key must also be restarted.

For more information on logging in with `cdswctl`, see *Log into cdswctl*.

Related Information

Log into cdswctl

Initialize an SSH connection to Cloudera AI for VS code

The following task describes how to establish an SSH endpoint for Cloudera AI. Creating an SSH endpoint is the first step to configuring a remote editor for Cloudera AI.

Procedure

1. Log in to Cloudera AI with the CLI client.

```
cdswctl login -n <username> -u http(s)://cdsw.YOUR_DOMAIN.com
```

For example, the following command logs the user sample_user into the https://cdsw.*YOUR_DOMAIN*.com deployment:

```
cdswctl login -n SAMPLE_USER -u https://cdsw.YOUR_DOMAIN.com
```

2. Create a local SSH endpoint to Cloudera AI.

Run the following command:

```
cdswctl ssh-endpoint -p <username>/<project_name> [-c <CPU_cores>] [-m <
memory_in_GB>] [-g <number_of_GPUs>] [-r <runtime ID> ]
```

If the project is configured to use ML runtimes, the -r parameter must be specified, otherwise it must be omitted. To retrieve the Runtime ID, use the following command:

cdswctl runtimes list

See *Using ML runtimes with cdswctl* documentation page for more information.

The command uses the following defaults for optional parameters:

- CPU cores: 1
- Memory: 1 GB
- GPUs: 0

For example, the following command starts a session for the logged-in user sample_user under the customerchurn project with .5 cores, .75 GB of memory, 0 GPUs, and the Python3 kernel:

```
cdswctl ssh-endpoint -p customerchurn -c 0.5 -m 0.75
```

To create an SSH endpoint in a project owned by another user or a team, for example finance, prepend the username to the project and separate them with a forward slash:

```
cdswctl ssh-endpoint -p finance/customerchurn -c 0.5 -m 0.75
```

This command creates session in the project customerchurn that belongs to the team finance.

Information for the SSH endpoint appears in the output:

```
...
You can SSH to it using
    ssh -p <some_port> cdsw@localhost
...
```

3. Open a new command prompt and run the outputted command from the previous step:

```
ssh -p <some_port> cdsw@localhost
```

For example:

```
ssh -p 7847 cdsw@localhost
```

You will be prompted for the passphrase for the SSH key you entered in the Cloudera AI web UI.

The public key could be rejected when the new ssh key pair is generated with a special name such as id_rsa_syste st. If the public key is rejected, you must add the following information to the ~/.ssh/config file:

```
Host *
          AddKeysToAgent yes
          StrictHostKeyChecking no
          IdentityFile ~/.ssh/id_rsa_cdswctl
```

Once you are connected to the endpoint, you are logged in as the cdsw user and can perform actions as though you are accessing the terminal through the Cloudera AI web UI.

4. Test the connection.

If you run ls, the project files associated with the session you created are shown. If you run whoami, the command returns the cdsw user.

Once you are connected, you should see something like this:

```
$ cdswctl ssh-endpoint -p ml-at-scale -m 4 -c 2
Forwarding local port 7847 to port 2222 on session bhsb7k4eqmonap62 in p
roject finance/customerchurn.
You can SSH to the session using

    ssh -p 7847 cdsw@localhost
```

5. Add an entry into your SSH config file.

For example:

```
$ cat ~/.ssh/config
Host cdsw-public
    HostName localhost
    IdentityFile ~/.ssh/id_rsa
    User cdsw
    Port 7847
```

HostName is always localhost and User is always cdsw. You get the Port number from Step 2.

Setting up VS Code

In VS Code, you can configure an SSH interpreter. Cloudera AI uses this method to connect to VS Code and act as its interpreter.

Before you begin

Ensure that you have installed the following VS Code extensions:

- Remote SSH extension

  Remote Development using SSH
- [Optional] Python extension
- [Optional] R extension

About this task

Before you begin, ensure that the SSH endpoint for Cloudera AI is running on your local machine. If additional information is required, see the documentation for your version of VS Code for specific instructions.

Procedure

1. Verify that the SSH endpoint for Cloudera AI is running with cdswctl.

   If the endpoint is not running, start it.
2. Open VS Code.
3. Open the command pallet and connect to a remote host.

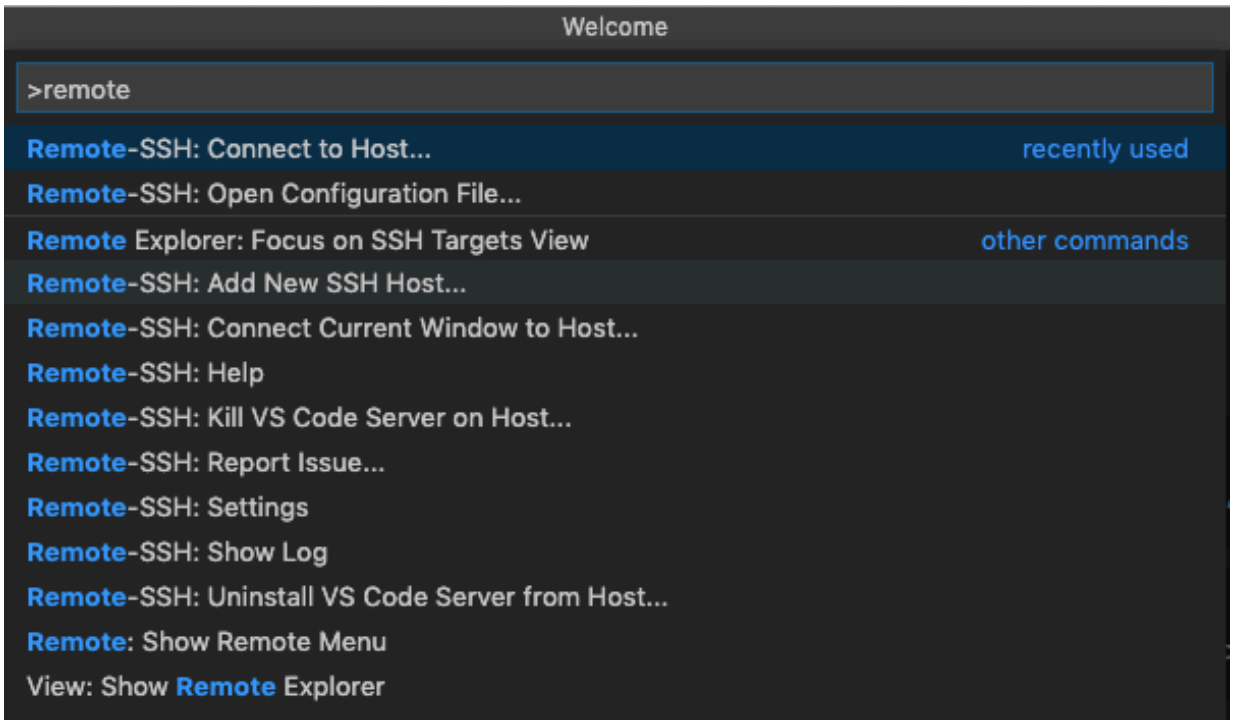

4. Connect to the host you added previously.

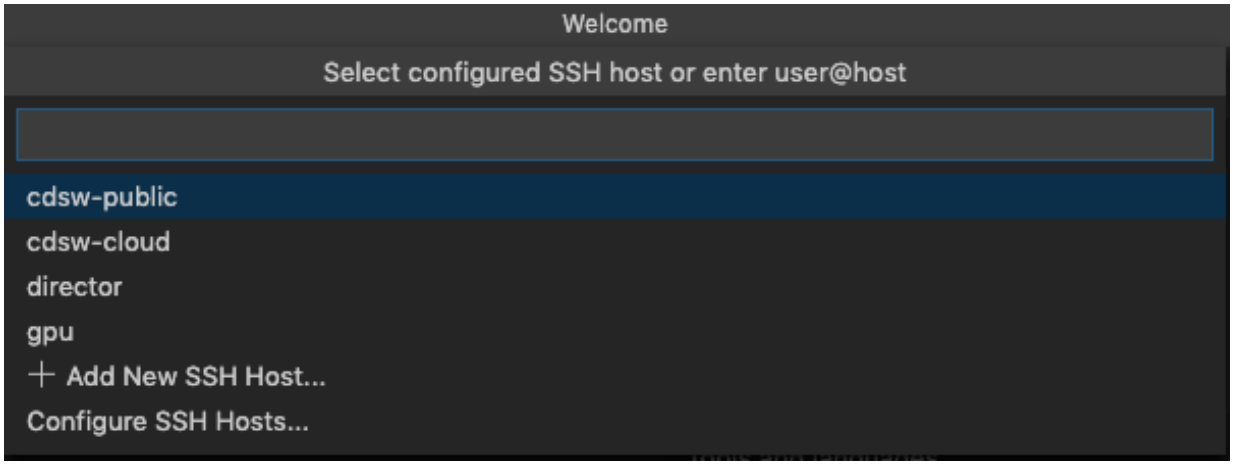

5. For the first connection, you must accept the fingerprint.

   You might not see a pop up, so pay attention to VS Code. If it is the first time your are connecting to a new session, or the port number changed, you will need to accept the fingerprint.

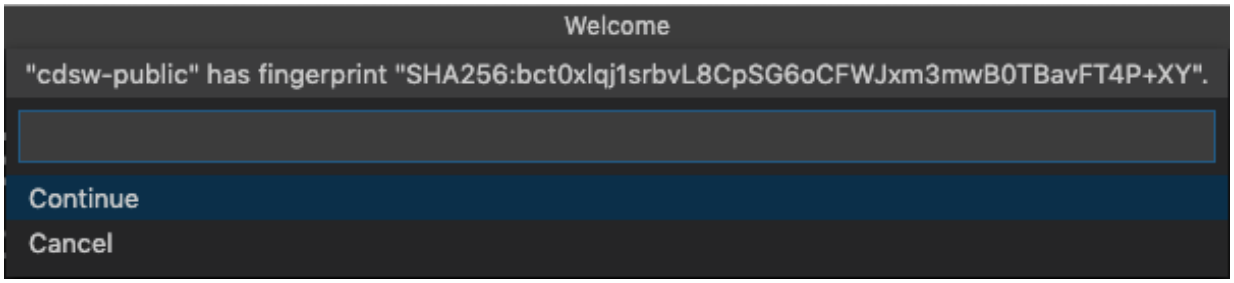

   While VS Code connects and sets up the remote connection, it installs some helper applications on the Cloudera AI server. Sometimes the remote session dies. Click Retry or if it's taking a long time, restart the remote session and it will recover.

   Note: If you get stuck in a loop during setup with VS Code reconnecting every 30 seconds or so, the issue is with the lock file that VS Code creates during the install. Close VS Code and in Cloudera AI terminal, delete the /home/cdsw/.vscode-server/ directory and start again.

6. After you are connected, you can open the Explorer and view and edit the files in the /home/cdsw directory.

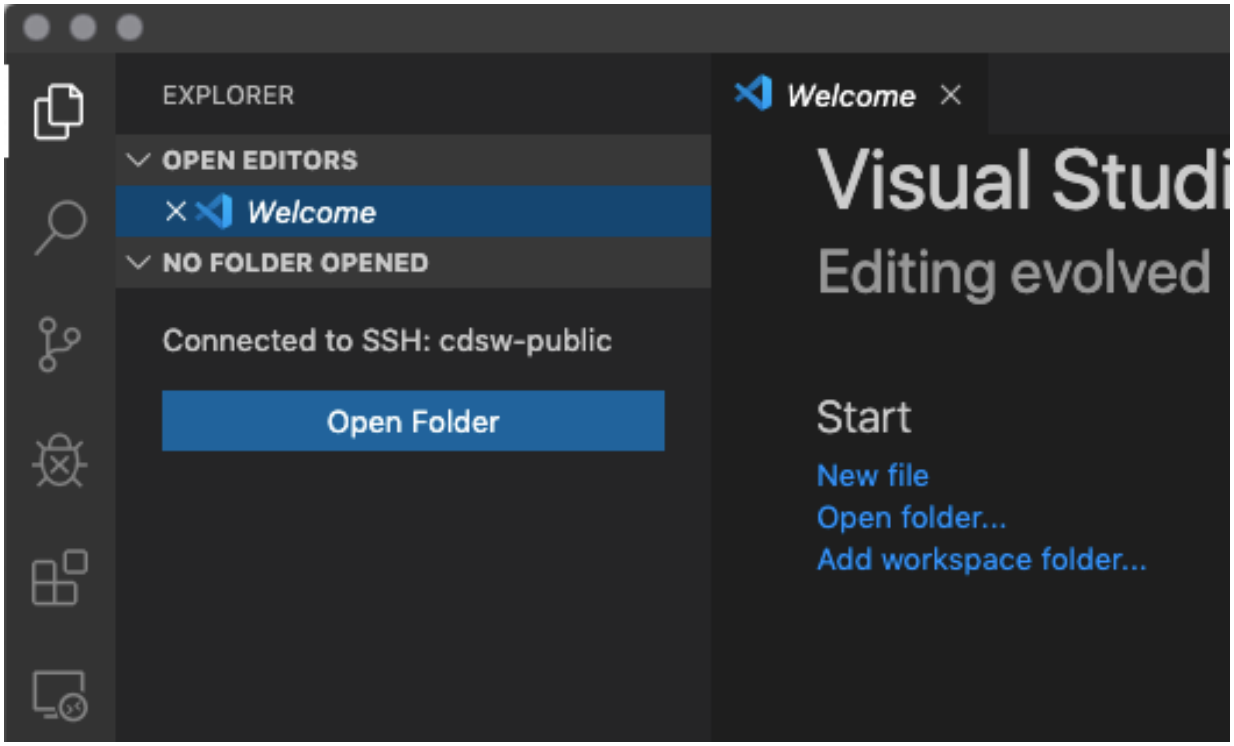

7. From the Explorer view, you can edit any of the files on your Cloudera AI server.

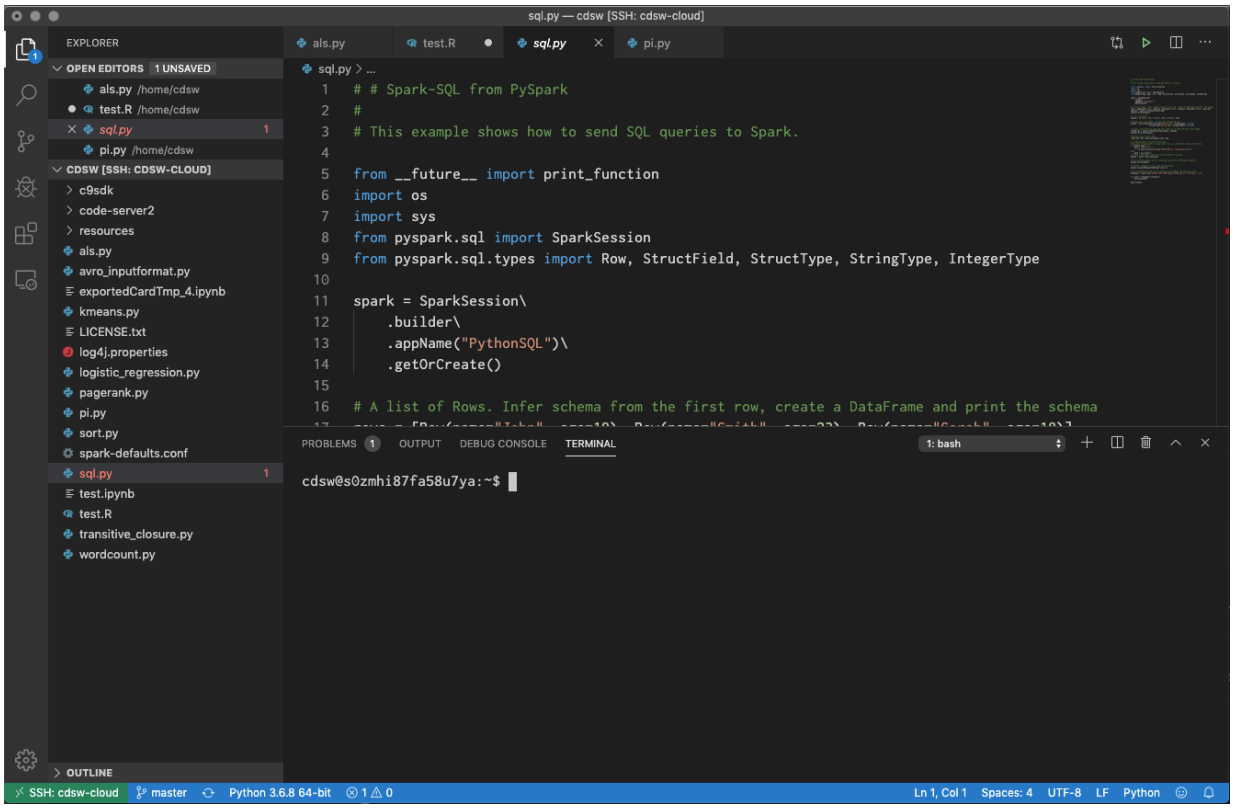

Using the Explorer view, you remotely edit and modify your Cloudera AI files. VS Code also supports Python and R, some powerful coding tools, that you can take advantage of over the remote connection.

(Optional) Using VS Code with Python

You can use VS Code with Python.

About this task

To take full advantage of VS Code Python tools, you must install the Python extension into the remote SSH session. You must install the extension the first time you connect a newly configured remote session.

Procedure

1. Install the Python extension.

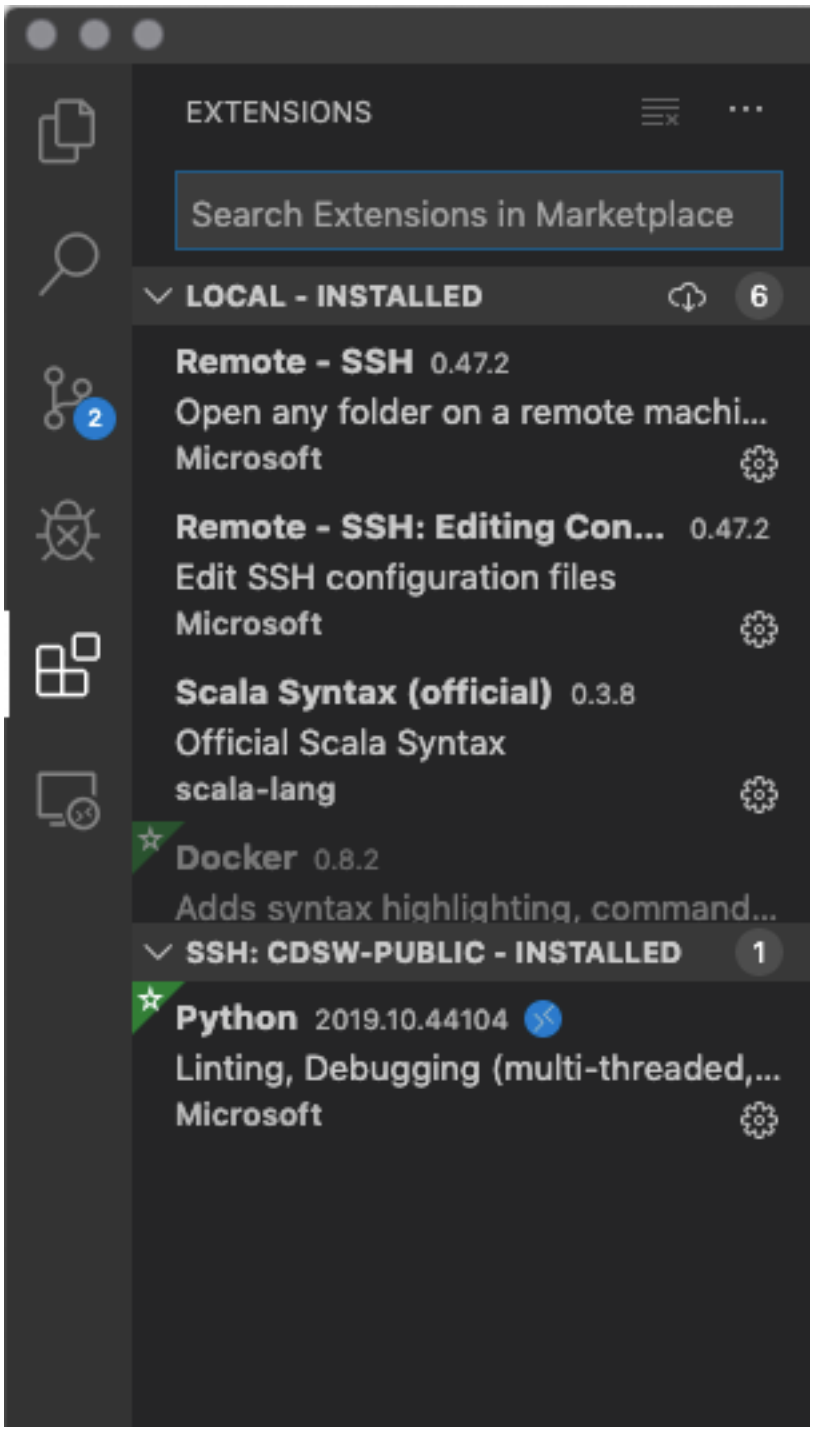

2. With the Extension installed, once you open your first python file, you will be prompted to install pylint Linter.

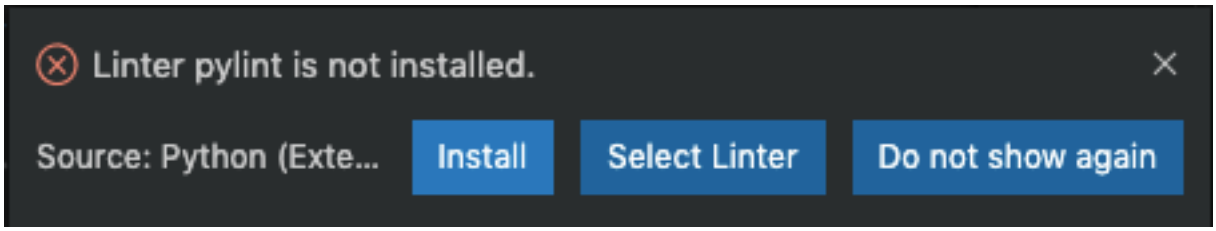

3. Click Install.

   VS Code opens a terminal and runs the code needed to install the linter. It's important to note that this is a remote terminal, running on an engine in Cloudera AI. It is the same as if you launched a terminal inside a running workbench.

4. If you want to run arbitrary Python code inside VS Code, open a Python file, select some code, right click, and select Run Selection/Line in Python Terminal.

   You can also just hit Shift-Enter in the code editor window. This will open up a new terminal, run the selected code. Since this is a remote session, you can run pyspark directly inside VS Code.

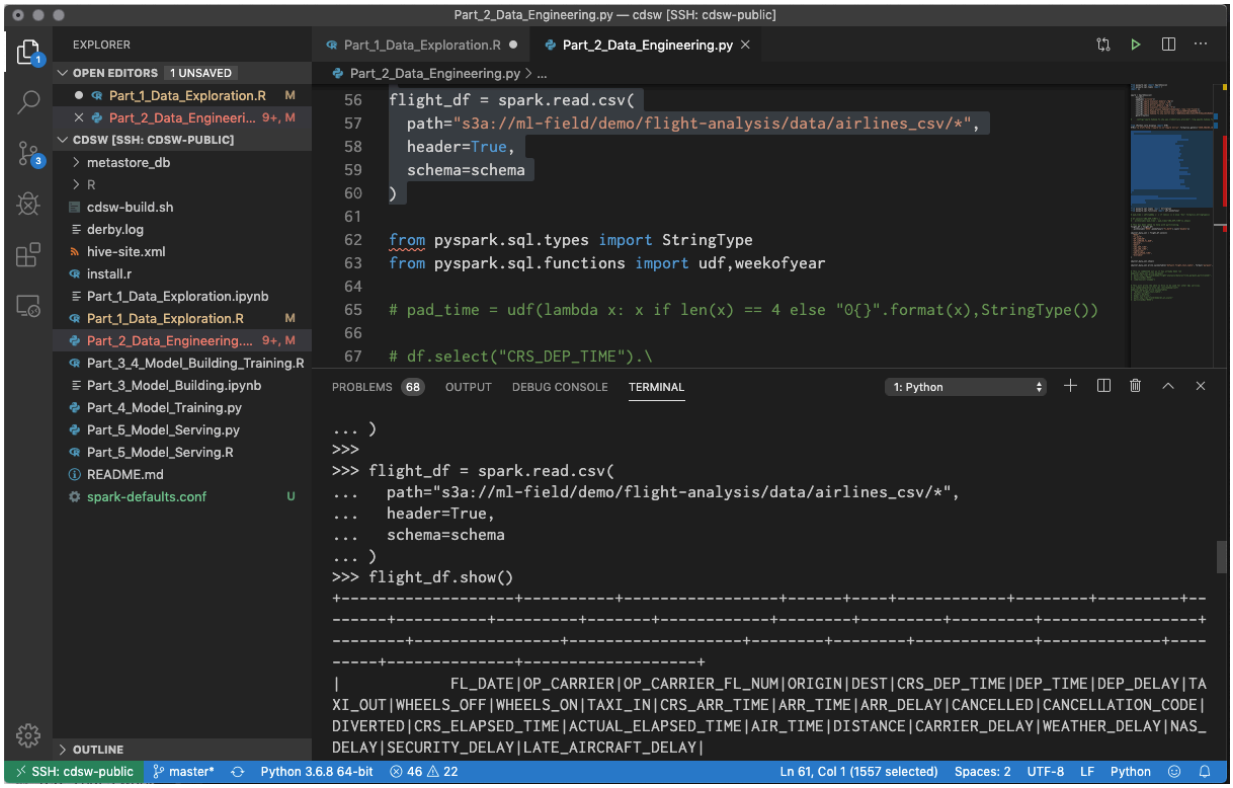
Part_2_Data_Engineering.py — cdsw [SSH: cdsw-public]
EXPLORER
OPEN EDITORS 1 UNSAVED
Part_1_Data_Exploration.R
Part_2_Data_Engineeri... 9+, M
CDSW [SSH: CDSW-PUBLIC]
metastore_db
R
cdsw-build.sh
derby.log
hive-site.xml
install.r
Part_1_Data_Exploration.ipynb
Part_1_Data_Exploration.R
Part_2_Data_Engineering.... 9+, M
Part_3_4_Model_Building_Training.R
Part_3_Model_Building.ipynb
Part_4_Model_Training.py
Part_5_Model_Serving.py
Part_5_Model_Serving.R
README.md
spark-defaults.conf
OUTLINE
Part_1_Data_Exploration.R
Part_2_Data_Engineering.py
Part_2_Data_Engineering.py > ...
flight_df = spark.read.csv(
path="s3a://ml-field/demo/flight-analysis/data/airlines_csv/*",
header=True,
schema=schema
)
from pyspark.sql.types import StringType
from pyspark.sql.functions import udf,weekofyear
# pad_time = udf(lambda x: x if len(x) == 4 else "0{}".format(x),StringType())
# df.select("CRS_DEP_TIME").\
PROBLEMS 68 OUTPUT DEBUG CONSOLE TERMINAL
1: Python
...)
>>>
>>> flight_df = spark.read.csv(
... path="s3a://ml-field/demo/flight-analysis/data/airlines_csv/*",
... header=True,
... schema=schema
...)
>>> flight_df.show()
+-----+-----+-----

Procedure

1. Prior to installing the R extension, check where your R binary lives in CDW by running which R and then pasting that into the R > Rterm: Linux setting in VS Code.

   The R binary is most likely located in /usr/local/bin/R, but its best to check.

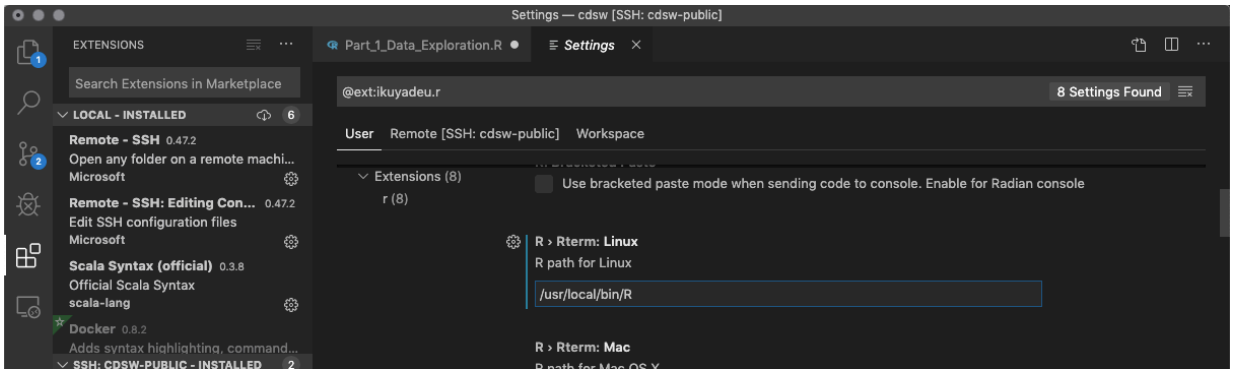

2. After you install the R extension, you can use sparklyr to run spark code using R inside VS Code.

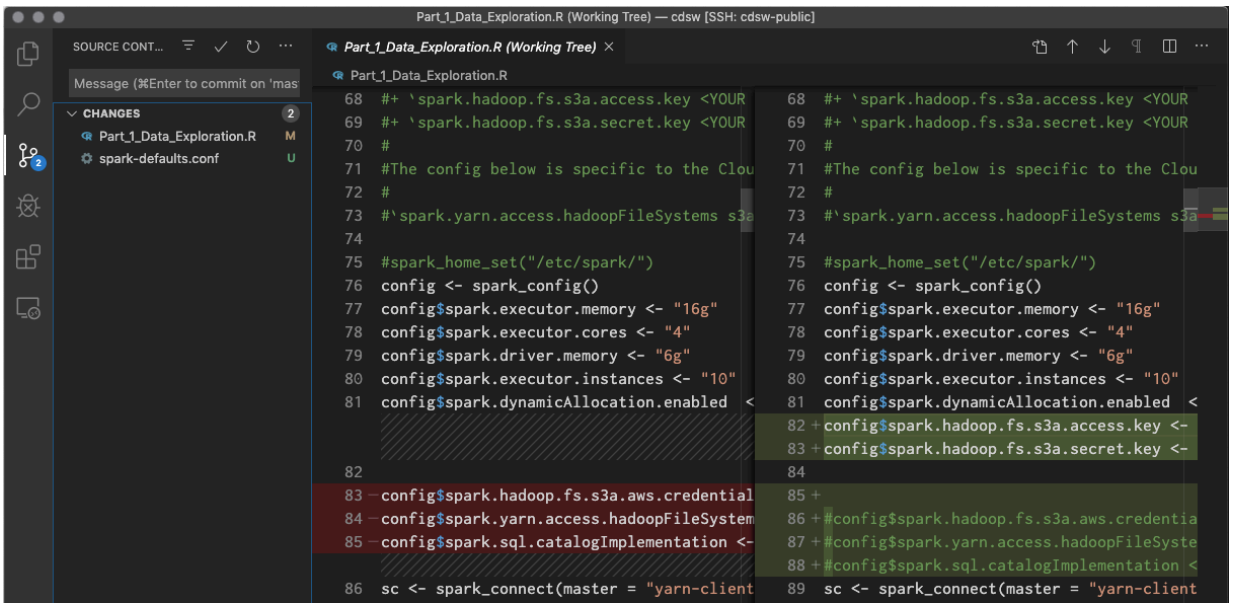

(Optional) Using VS Code with Jupyter

You can use VS Code with Jupyter Notebooks.

About this task

You can work on Jupyter Notebooks within VS Code. This gives you all the great code completion, syntax highlighting and documentation hints that are part of the VS Code experience and the interactivity of a Jupyter Notebook. Any changes you make to the Notebook will be reflected on the CDSW / Cloudera AI server and can be viewed online using Jupyter Notebook as a browser based editor.

Procedure

1. Because of the way Cloudera AI uses the internal networking and port forwarding of Kubernetes, when VS Code launches a Jupyter Server it binds to the wrong address and access is blocked. You therefore have to launch your own Jupyter Server and tell VS code to connect to that.
   a) The first setting you need to set is the Python > Data Science: Jupyter Server URI setting. Set this to http://127.0.0.1:8888/?token=[some-token].

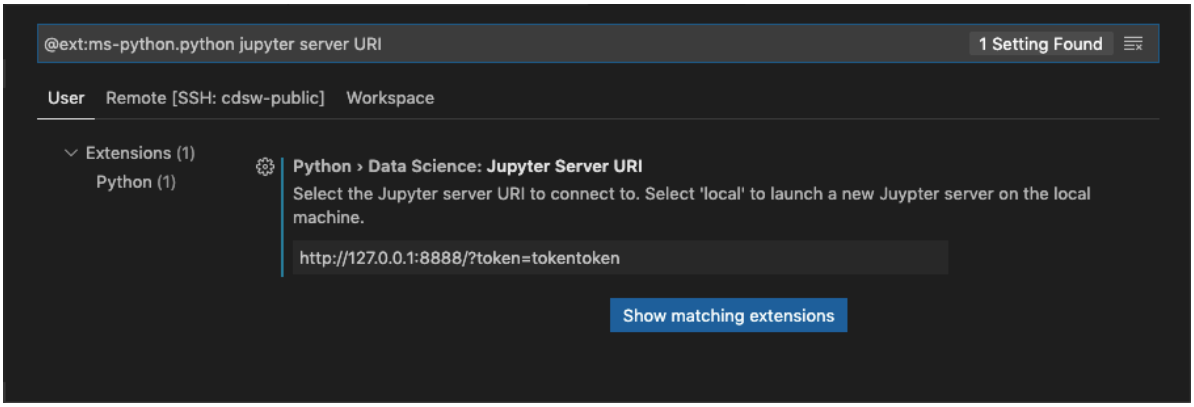

   b) Then you need to open a terminal to launch a Jupiter Notebook server.

      You can launch it using: /usr/local/bin/jupyter-notebook —no-browser —ip=127.0.0.1 —NotebookApp.token=[some-token] —NotebookApp.allow_remote_access=True .

      This creates a Jupyter server that any new Notebooks you launch will run in.

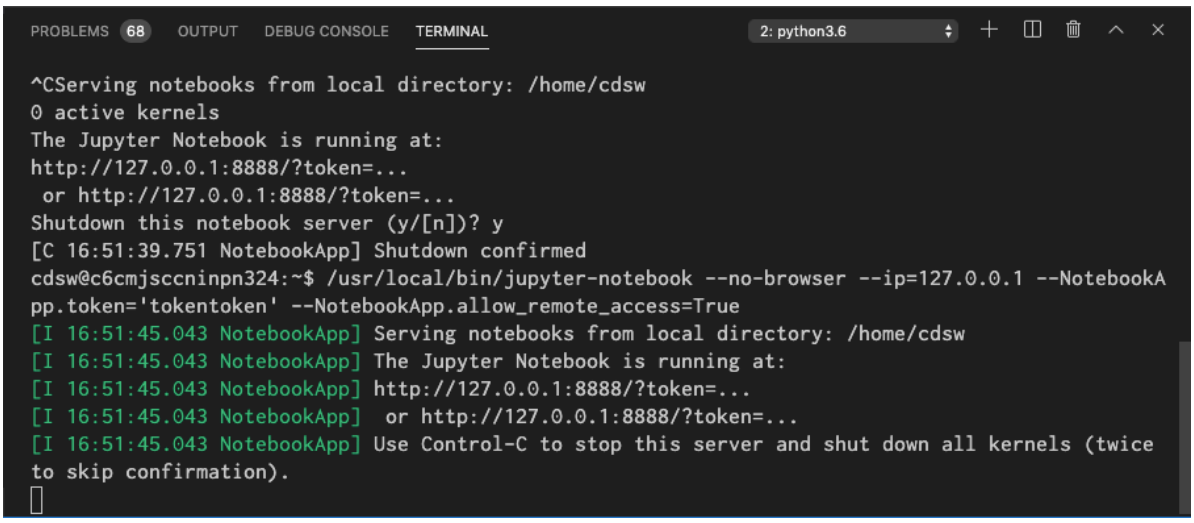

2. After you install the Jupyter Notebooks, you can use it inside VS Code.

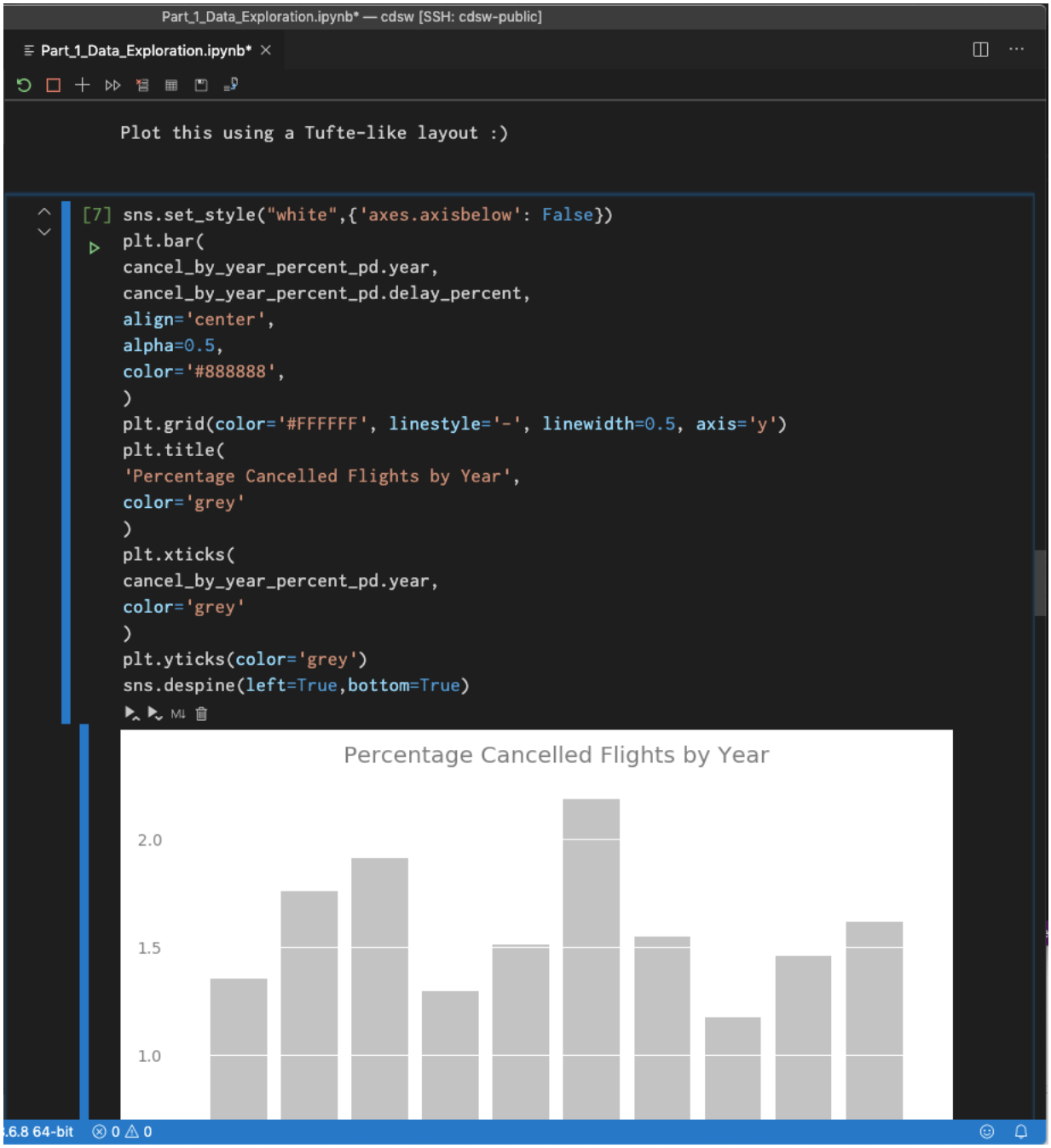

3. Another feature that you can use with VS Code is running a temporary Notebook for executing random code snippets. Select code you want to run, right click and click Run Current File in Python Interactive Window.

   This is less robust though and will create many Untitled*.ipynb files in your home directory.

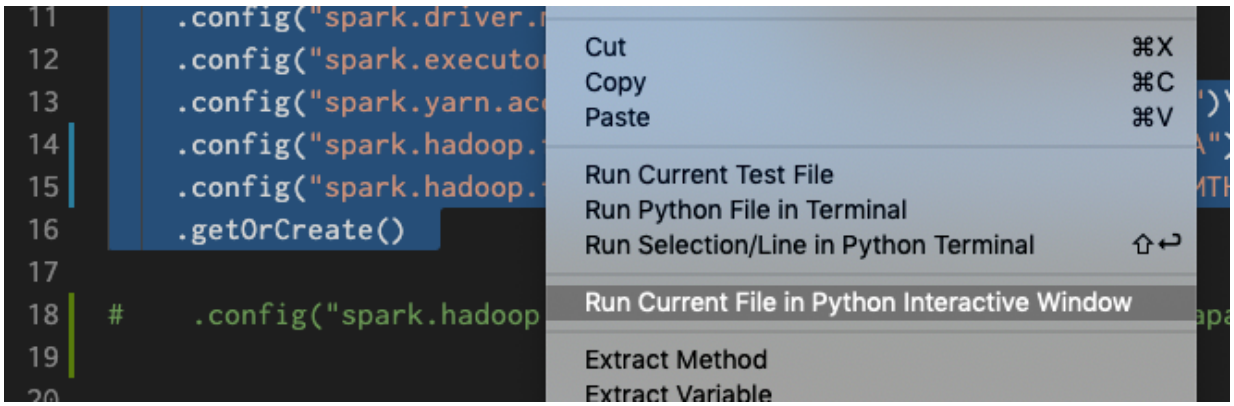

(Optional) Using VS Code with Git integration

VS Code has substantial Git integration.

About this task

If you created your project from a git repo or a custom template, your changes and outside changes made to the repo will automatically appear.

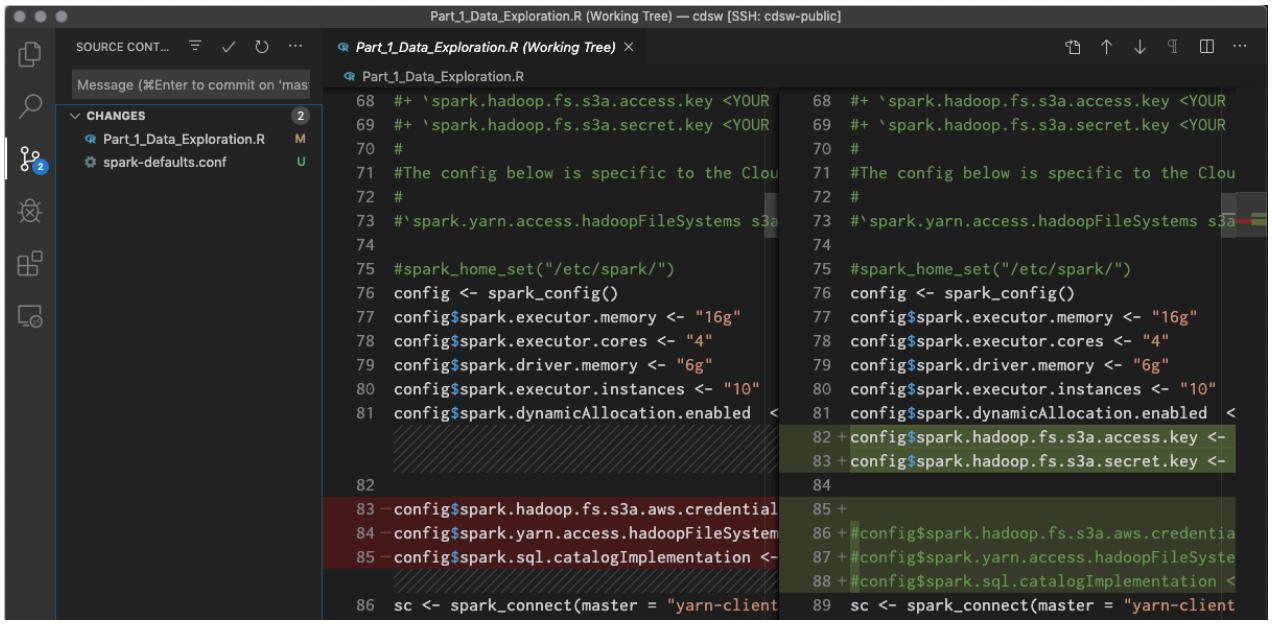

Limiting files in Explorer view

You can limit the number of files shown in the Explorer view.

About this task

If you end up with many of .[something] directories, in /home/cdsw, it can be difficult to navigate.

Procedure

If you add the **/.* pattern to the Files: Exclude setting, it will hide all those files and directories.

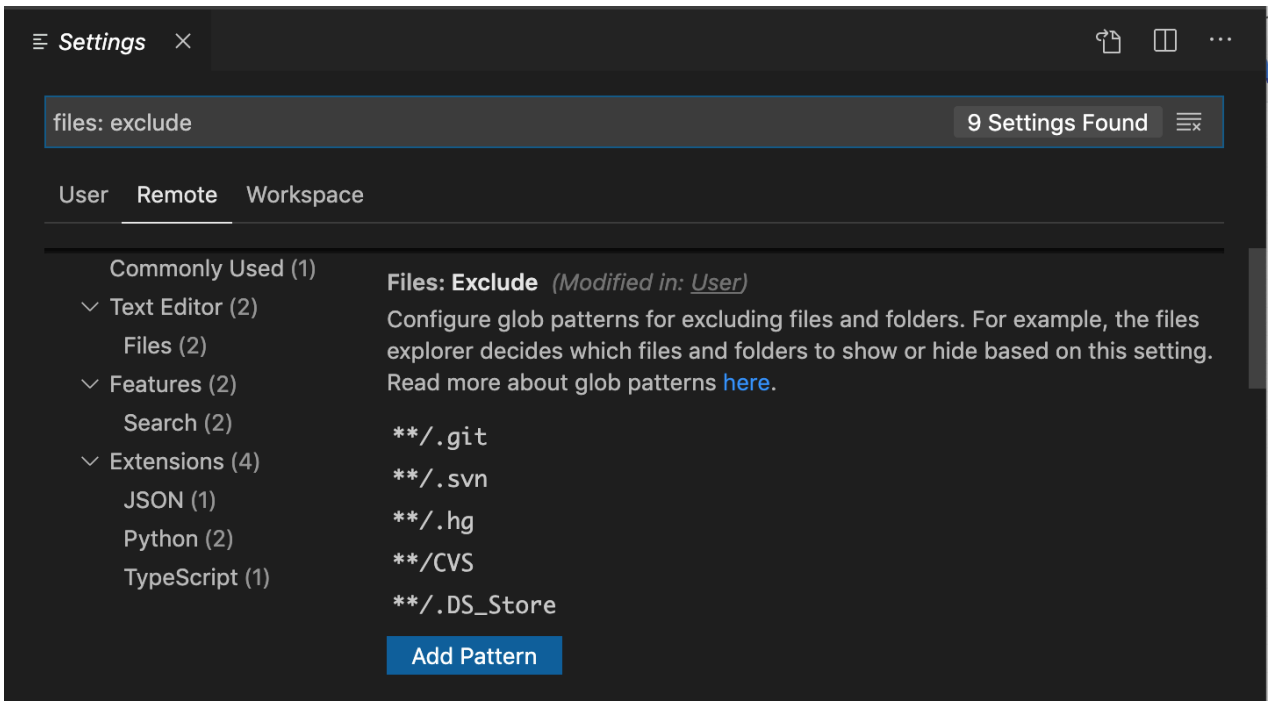

Git for Collaboration

Cloudera AI provides seamless access to Git projects. Whether you are working independently, or as part of a team, you can leverage all of benefits of version control and collaboration with Git from within Cloudera AI.

Teams that already use Git for collaboration can continue to do so. Each team member will need to create a separate Cloudera AI project from the central Git repository. For anything but simple projects, Cloudera recommends using Git for version control. You should work on Cloudera AI the same way you would work locally, and for most data scientists and developers that means using Git.

Cloudera AI does not include significant UI support for Git, but instead allows you to use the full power of the command line. If you launch a session and open a Terminal, you can run any Git command, including `init`, `add`, `commit`, `branch`, `merge` and `rebase`. Everything should work exactly as it does locally.

When you create a project, you can optionally supply an HTTPS or SSH Git URL that points to a remote repository. The new project is a clone of that remote repository. You can commit, push and pull your code by running a console and opening a Terminal. Note that if you want to use SSH to clone the repo, you will need to first add your personal Cloudera AI SSH key to your GitHub account. For instructions, see *Adding SSH Key to GitHub*.

If you see Git commands hanging indefinitely, check with your cluster administrators to make sure that the SSH ports on the Cloudera AI hosts are not blocked.

Related Information

Adding an SSH Key to GitHub

Creating a Project

Linking an existing Project to a Git remote

If you did not create your project from a Git repository, you can link an existing project to a Git remote (for example, git@github.com:username/repo.git) so that you can push and pull your code.

Procedure

1. Launch a new session.
2. Open a terminal.
3. Enter the following commands:

   Shell

   ```
   git init
   git add *
   git commit -a -m 'Initial commit'
   git remote add origin git@github.com:username/repo.git
   ```

   You can run git status after git init to make sure your .gitignore includes a folder for libraries and other non-code artifacts.

Web Applications Embedded in Sessions

This topic describes how Cloudera AI allows you to embed web applications for frameworks such as Spark 2, TensorFlow, Shiny, and so on within sessions and jobs.

Many data science libraries and processing frameworks include user interfaces to help track progress of your jobs and break down workflows. These are instrumental in debugging and using the platforms themselves. For example, TensorFlow visualizations can be run on TensorBoard. Other web application frameworks such as Shiny and Flask are popular ways for data scientists to display additional interactive analysis in the languages they already know.

Cloudera AI allows you to access these web UIs directly from sessions and jobs. This feature is particularly helpful when you want to monitor and track progress for batch jobs. Even though jobs don't give you access to the interactive workbench console, you can still track long running jobs through the UI. However, note that the UI is only active so long as the job or session is active. If your session times out after 60 minutes (default timeout value), so will the UI.

Important: If you want to share your web application as a long-running standalone application that other business users can access, Cloudera recommends you now use the Applications feature to support long-running web applications on Cloudera AI Workbenches.

If you are only running a server-backed visualization as part of your own analysis, then you can continue to keep embedding web applications in sessions as described in this topic. Note that running web applications in sessions is also the recommended way to develop, test, and debug analytical apps before deployment.

HTTP services running in containers that bind to CDSW_APP_PORT, CDSW_READONLY_PORT, and CDSW_PUBLIC_PORT are available in browsers at the following urls:

- CDSW_APP_PORT is available at

  ```
  https://<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>
  ```

- CDSW_READONLY_PORT is available at

  ```
  https://read-only-<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>
  ```

- CDSW_PUBLIC_PORT is available at

  ```
  https://public-<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>
  ```

Therefore, TensorBoard, Shiny, Flask or any other web framework accompanying a project can be accessed directly from within a session or job, as long as it is run on CDSW_APP_PORT or CDSW_READONLY_PORT.

CDSW_APP_PORT is meant for applications that grant some level of control to the project, such as access to the active session or terminal. CDSW_READONLY_PORT must be used for applications that grant read-only access to project results.

To access the UI while you are in an active session, click the grid icon in the upper right hand corner of the Cloudera AI web application, and select the UI from the dropdown. For a job, navigate to the job overview page and click the History tab. Click on a job run to open the session output for the job. You can now click the grid icon in the upper right hand corner of the Cloudera AI web application to access the UI for this session.

Limitations with port availability

Cloudera AI exposes only one port per-access level. This means, in version 1.6.0, you can run a maximum of 3 web applications simultaneously:

- one on CDSW_APP_PORT, which can be used for applications that grant some level of control over the project to Contributors and Admins,
- one on CDSW_READONLY_PORT, which can be used for applications that only need to give read-only access to project collaborators,
- and, one on the now-deprecated CDSW_PUBLIC_PORT, which is accessible by all users.

However, by default the editors feature runs third-party browser-based editors on CDSW_APP_PORT. Therefore, for projects that are already using browser-based third-party editors, you are left with only 2 other ports to run applications on: CDSW_READONLY_PORT and CDSW_PUBLIC_PORT. Keep in mind the level of access you want to grant users when you are selecting one of these ports for a web application.

Example: A Shiny Application

This example demonstrates how to create and run a Shiny application and view the associated UI while in an active session.

Create a new, blank project and run an R console. Create the files, ui.R and server.R, in the project, and copy the contents of the following example files provided by Shiny by RStudio:

R

```
# ui.R

library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

  # Application title
  titlePanel("Hello Shiny!"),

  # Sidebar with a slider input for the number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),

    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

R

```
# server.R
library(shiny)
# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should re-execute automatically
  #     when inputs change
  #  2) Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

Run the following code in the interactive workbench prompt to install the Shiny package, load the library into the engine, and run the Shiny application.

R

```
install.packages('shiny')

library('shiny')

runApp(port=as.numeric(Sys.getenv("CDSW_READONLY_PORT")), host="127.0.0.1",
 launch.browser="FALSE")
```

Finally, to access the web application, either:

- Click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application, and select the Shiny UI, Hello Shiny!, from the dropdown.
- Access the web application directly by visiting the URL: https://public-[session-id].[CML host]/

The UI will be active as long as the session is still running.

Cloudera AI

Using Registered Models

Date published: 2020-07-16
Date modified: 2025-01-29

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Registered Models.....4
Deploying a model from Registered Models.....4
Importing a Hugging Face Model (Technical Preview).....4
Viewing details of a registered model.....8
Editing model visibility.....8
Deleting a registered model version.....9

Using Registered Models

Registered Models offers a single view for models stored in Cloudera AI Registries across Cloudera Data Platform Environments and facilitate easy deployment to Cloudera AI Inference service.

When you import models from Model Hub, the models are listed under Registered Models. Review all imported models and associated metadata, such as the model's associated environment, visibility, owner name, and created date.

This page lists all imported models and associated metadata, such as the model's associated environment, visibility, owner name, and created date. You can click on any model to view details about that model, and its versions, and deploy any specific version of the model to the Cloudera AI Inference service. You can also delete a specific version of the model on this page. When you try to import a model with the same name, a new version of that model is added which can then be viewed under Registered Models.

Deploying a model from Registered Models

You can deploy a model from the Registered Models page into Cloudera AI Inference service.

Procedure

1. In the **Cloudera** console, click the **Cloudera AI** tile.

   The **Cloudera AI Workbenches** page displays.
2. Click **Registered Models** under **Deployments** in the left navigation menu.

   The **Registered Models** page displays. The page lists all the models of different Cloudera AI Registries along with the associated metadata.
3. Select the model you want to deploy. The **Registered Models** page displays the model information and the available versions of the model.
4. Click Deploy to deploy the latest version.

   You can select All from the version drop-down to view all the versions and click in the respective row of that version. You can deploy any version of the model the status of which is displayed as Ready. The Deploy Model dialog box is displayed.
5. In the Deploy Model dialog box, select the cluster of the Cloudera AI Inference service in which you want to deploy this model. The Create Endpoint page is displayed.
6. Enter the information and click OK.

   For information on creating an endpoint, see *Using Cloudera AI Inference service*.

Related Information

Using Cloudera AI Inference service

Importing a Hugging Face Model (Technical Preview)

If your desired Hugging Face model is unavailable on the Model Hub page, you can import those models from the *Hugging Face* website. After you import the model, the newly imported model will be listed on the Registered Models page.

Note: This feature is in Technical Preview and not recommended for production deployments. Cloudera recommends that you try this feature in test or development environments.

Procedure

1. In the **Cloudera** console, click the **Cloudera AI** tile.

   The **Cloudera AI Workbenches** page displays.
2. Click **Registered Models** under **Deployments** in the left navigation menu.

   The **Registered Models** page displays. The page lists all the models of different Cloudera AI Registries along with the associated metadata.

3. Click Import Model. The Import Model page displays.

Import Model

Technical Preview - Import Hugging Face Models

This feature is in Technical Preview, so some models may not fully integrate with Cloudera AI Inference Service.

* AI Registry

Select AI Registry

* Name

Visibility

Public Private

* Repository ID

Hugging Face Token

Enter your Hugging Face Token

Description

Version Notes

Cancel Import

4. In the AI Registry drop-down list, select the AI registry to which you want to import the model.
5. In the Name field, enter a new name for the model you are importing.
6. Select the Visibility as Public or Private. If you select Public, the model is available for other users. If you select Private, the model is displayed on the Registered Models page only for the user who imported it.
7. In the Repository ID field, enter the ID of the Hugging Face model. You can obtain the ID of a model from the Hugging Face website.
8. In the Hugging Face Token field, enter the token obtained from the Hugging Face website.
9. In the Description field, enter a description for the model.
10. In the Version Notes field, enter notes about this version of the model.
11. Click Import.

Results

You can view this newly imported model on the Registered Models page.

Viewing details of a registered model

You can view details like version information about the models in your AI Registries in the Registered Models page. By default, the latest version information is displayed. Model card describes the model, governing terms, the family of models, resources used, and further information on how to use the model, and so on. It also provides information about various versions, optimizations made in the specific version, and the minimum resource configuration required to deploy those versions.

Procedure

1. In the **Cloudera** console, click the **Cloudera AI** tile.

   The **Cloudera AI Workbenches** page displays.
2. Click **Registered Models** under **Deployments** in the left navigation menu.

   The **Registered Models** page displays. You can see all the registered models, associated environment, their owner, visibility, and the last updated time.
3. You can use the filter bar at the top of the window to filter the list of registered models by tag and environment name.
4. Select a registered model to see its description.

Editing model visibility

You can modify the visibility of the model to private or public status. If the visibility is set to Public, the registered model is available for all the users irrespective of their role. If the visibility is set to Private, the model is available only to the owner and the administrators of that environment.

Procedure

1. In the **Cloudera** console, click the **Cloudera AI** tile.

   The **Cloudera AI Workbenches** page displays.
2. Click **Registered Models** under **Deployments** in the left navigation menu.

   The **Registered Models** page displays.
3. Select the model whose visibility you want to change.
4. Click Edit Model and change the visibility of the model to Private or Public and the description of the model, if needed.
5. Click Update.

Deleting a registered model version

If you no longer want to access a version of a registered model, you can delete it.

Procedure

1. In the **Cloudera** console, click the **Cloudera AI** tile.
   The **Cloudera AI Workbenches** page displays.
2. Click **Registered Models** under **Deployments** in the left navigation menu.
   The **Registered Models** page displays.
3. Select the model you want to delete .
4. Click All from the version drop-down menu to view all the versions.
5. From the Actions menu, click .
6. Click OK to confirm the deletion.

Cloudera AI Release Notes

Date published: 2020-07-16
Date modified: 2025-01-29

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What's New 5
January 29, 2025 5
November 21, 2024 6
October 10, 2024 7

Older releases 8
October 8, 2024 8
September 26, 2024 8
July 17, 2024 9
June 20, 2024 9
June 11, 2024 10
May 29, 2024 10
May 15, 2024 10
April 25, 2024 11
February 20, 2024 11
March 6, 2024 12
February 8, 2024 12
January 23, 2024 12
December 15, 2023 13
November 15, 2023 14
October 19, 2023 14
August 31, 2023 15
July 25, 2023 15
July 12, 2023 16
May 31, 2023 16
May 16, 2023 17
April 26, 2023 17
April 5, 2023 18
March 27, 2023 18
February 14, 2023 18
February 10, 2023 19
February 7, 2023 19
November 29, 2022 19
October 19, 2022 20
September 27, 2022 20
August 30, 2022 21
July 21, 2022 21
May 31, 2022 21
April 21, 2022 22
March 14, 2022 22
February 10, 2022 23
January 12, 2022 23
December 15, 2021 23
December 13, 2021 23
October 27, 2021 24
October 18, 2021 25
August 31, 2021 25
August 23, 2021 25

July 8, 2021 26
June 24, 2021 26
May 13, 2021 27
March 22, 2021 27
February 3, 2021 27
December 21, 2020 28
November 23, 2020 28
October 29, 2020 29
August 04, 2020 29
June 30, 2020 30
June 9, 2020 31
May 5, 2020 31
April 14, 2020 32
March 16, 2020 32
February 13, 2020 33
January 30, 2020 33
December 19, 2019 33
November 1, 2019 34
September 23, 2019 34
August 22, 2019 35

Compatibility for Cloudera AI and Runtime components 35

CVE-2021-44228 Remediation for Cloudera AI Data Service 36

Known Issues and Limitations 40

What's New

Major features and updates for the Cloudera AI data service.

January 29, 2025

Release notes and fixed issues for version 2.0.46-b302.

New Features / Improvements

- Migrated Cloudera AI Workbench, AI Registry, and Cloudera AI Inference service images to chainguard to address CVEs.
- Added APIv2 support for Enhanced Group Sync.
- Added support to create AMPs (Cloudera Accelerators for Machine Learning Projects) using APIv2. Previously, this option was available only using UI.
- Added support for H100 GPU instances for Cloudera AI Inference service on Azure.
- Added support for AKS workload identity.
- Added support for AWS M7a, M7i, C7a, C7i, R7a, R7i instance families.
- Added support for Cloudera AI Inference service on EU Control Plane.
- Added support for EKS 1.30.
- Added support for AKS 1.30.
- Hugging Face support (Technical Preview): You can now import text-generating language models from Hugging Face and deploy them on Cloudera AI Inference service.
- Added profiles for HuggingFace Models and multi-modal models in the Model Hub catalog.
- Updated existing model manifests in the catalog after upgrading the NIM version in Cloudera AI Inference service.
- Enhanced error messages related to model import failure in the Model Hub UI.
- Carried enhancements in AI Registry to ensure that multi-modals can be supported.
- Added runtime support for Llama 3.2 11B and 90B Vision Language Model NIMs to ensure that they can be deployed using AI Inference. Only model profiles optimized for the H100 GPU are supported for these two models in this release.
- Llama 3 NIM is no longer supported since we now have both Llama 3.1 and Llama 3.2.
- Added support for Diagnostic Bundles in Cloudera AI Inference service.
- Upgraded text-generating and embedding NIMs.
- Added Code Sample functionality for endpoints deployed using Cloudera AI Inference service.
- Model endpoint replica events can now be viewed on the Model Endpoint details UI.You can now add numerous docker credentials using UI or API which can be used to enable Cloudera AI to fetch custom ML Runtimes from a secure repository. For more information, see Add Docker registry credentials and certificates.

Fixed Issues

- Previously, some Cloudera AI Inference service clusters did not have the 'creationDate' field. This field is now added.(DSE-38817)
- Previously, the deletion of backup for older workspaces was failing. This issue is now resolved. (DSE-41031)
- Previously, deleting a workbench backup created by a deleted user displayed an error. This issue is now resolved. (DSE-41052)
- Multiple UI improvements are made both in the Create, Read, Update, and Delete operations of Cloudera AI Inference service and while deploying or editing a model endpoint.
- The model_name field is now displayed instead of model_id in the Endpoint Details UI. (DSE-38937)

- Previously, the NIM model profile environment variable was only assigned for LLMs. Now support for Model Profile override is added for Embedding and Reranker NIMs. (DSE-40508)
- Previously, there was an issue with rendering of existing instance type in the "Edit Endpoint" UI. This issue is now resolved. (DSE-40636)
- Validated all node group (instance type) selection from UI. (DSE-40754)
- Previously, NGC manifest components were missing from the download. This issue is now resolved. (DSE-41055)
- The Create ML Serving application now enables the public load balancer. (DSE-41305)
- The Instance Type field in the Edit Model Endpoint UI is no longer mandatory. (DSE-41278)
- Added force delete option to delete the Cloudera AI Inference service using UI. (DSE-41035)
- The Cloudera AI Inference service UI now displays optimization profile details. (DSE-40927)
- You can now create, download, and delete log archives for Cloudera AI Inference service. (DSE-40921)
- The Test Model UI now fails gracefully when the replica is scaled down to zero for a model deployed using Cloudera AI Inference service. (DSE-40957)
- Previously, the Storage initializer had the wrong task values. This issue is now resolved. (DSE-41058)
- Enabled storage initializer to now handle more than two directories for NIM artifacts. (DSE-40986)
- Removed Llama 3 runtimes. (DSE-40956)
- Addressed SQL injection issue in AI Registry that allowed non-authorized but authenticated users to perform Create, Read, Update, and Delete operations on AI Registry’s metadata tables. (DSE-41542)

November 21, 2024

Release notes and fixed issues for version 2.0.46-b238.

New Features / Improvements

- Model Hub Enhancement: The model size is now shown in the user-friendly format both in the Model Hub UI and Cloudera AI Registry UI.
- Cloudera AI Inference service Enhancement: New AI Inference Services menu item is added to the left-navigation pane of the Cloudera AI UI to manage the lifecycle of Cloudera AI Inference service using UI. For more information, see Using Cloudera AI Inference service.
- Added Spark 3.5 ML Runtime Addon
- Product and features named:
  - *Clouder Machine Learning (CML)* is renamed to *Cloudera AI*.
  - *Cloudera Machine Learning Model Registry* is renamed to *Cloudera AI Registry*.
  - *Cloudera Machine Learning Workspace* is renamed to `Cloudera AI Workbench`.
  - *Cloudera Applied Machine Learning Prototypes* and `Accelerators for ML Projects` is renamed to *Cloudera Accelerators for Machine Learning Projects*.

Fixed Issues

- CVE fixes - This release includes numerous security fixes for critical and high Common Vulnerability and Exposures (CVE).
- Previously, the public and private settings did not carry forward after the AI Registry upgrade. This issue is now resolved. (DSE-36799)
- Enhanced the error message that was displayed when importing a model from Model Hub to Registered Models. (DSE-39897)
- Generic (vLLM) NIM profile deployment was returning an empty GPU list in the UI. This issue is now resolved. (DSE-39913)
- Previously, public cloud CDP CLI was not showing the instance type's GPU count. This issue is now resolved. (DSE-39539)

- Cloudera AI v2 API deployed application did not inherit user-level environment variables and site-level environment variables. This issue has been solved, and now an application created using APIv2 does not only inherit project-level environment variables but also user-level environment variables and site-level environment variables. (DSE-37611)
- Previously, scheduled jobs skipped job runs and did not specify the error. Now, the skipped jobs runs have improved exit code to distinguish them from failed jobs. (DSE-39976)
- Previously, the Next buttons on the Site Administration page did not work. This issue is now resolved. (DSE-34133).
- Previously, restarting the application using the Cloudera AI v2 API did not inherit account application-level environment variables. This issue is now resolved. (DSE-39894)
- Users can now view the existing applications in the Cloudera AI UI even if the creation of a new application is disabled. (DSE-39980)
- Previously, Python logging did not work with PBJ Runtimes. This issue is now resolved. (DSE-39929)
- Previously, reloading the session page would result in an incorrect state where the PBJ session's editor cell could appear green even if it is in a processing state (executing some commands). With this fix, an accurate representation of the processing state is displayed even after a refresh. (DSE-40049)

October 10, 2024

Release notes and fixed issues for version 2.0.46-b210.

New Features / Improvements

- Model Hub: Model Hub is now a fully supported feature. Model Hub is a catalog of top-performing models LLM and generative AI models. You can now easily import the models listed in the Model Hub into the Cloudera AI Registry and then deploy it using the Cloudera AI Inference service.

  For more information, see Using Model Hub.
- Cloudera AI Inference service Enhancements:
  - Added support for NVIDIA's NIM profiles requiring for the L40S GPU models.
  - Made auto-scale configuration which is rendered in UI during the creation of model endpoint user-friendly. (DSE-38845)
  - Optimized Cloudera AI UI service to become more responsive.
  - User actionable error messages are now rendered in Cloudera AI service UI.

    For more information, see Using Cloudera AI Inference service.

Fixed Issues

- Addressed scaling issues with web services to support high active user concurrency (DSE-39597).
- CVE fixes - This release includes numerous security fixes for critical and high Common Vulnerability and Exposures (CVE).
- Fixed CORS issue to ensure that DELETE/PATCH V1 API can be used from within a workbench. (DSE-39357)
- Made the NGC service key used to download Nvidia’s optimized models more restrictive. (DSE-39475)
- Previously, users were unable to copy the model-id from Cloudera AI UI. This issue is now resolved. (DSE-38889)
- Authorization issues related to the listing of Cloudera AI applications have been addressed. (DSE-39386)
- Fixed an issue to ensure that instance type validation is correctly carried out during the creation of a new model endpoint. (DSE-39634)
- Added required validation rules for the creation of a new model endpoint. (DSE-38412)
- Addressed an issue around empty model list during navigation from registry models to deployment of models. (DSE-39634)

Older releases

Overview of new features, enhancements, and changed behavior introduced in earlier releases of Cloudera Machine Learning.

October 8, 2024

Release notes and fixed issues for Cloudera AI Inference service version 1.2.0-b73.

New Features / Improvements

- Cloudera AI Inference service: Cloudera AI Inference service is now a fully supported data service. Cloudera AI Inference service is a production-grade serving environment for traditional, generative AI, and Large Language Models. It is designed to handle the challenges of production deployments, such as high availability, fault tolerance, and scalability. The service is now available to carry out inference on the following categories of models:
  - Optimized open-source Large Language Models.
  - Traditional machine learning models like classification, regression, and so on. Models need to be imported to the Cloudera AI Registry to be served using the Cloudera AI Inference service.

  For more information, see Using Cloudera AI Inference service.

September 26, 2024

Release notes and fixed issues for version 2.0.46-b200.

New Features / Improvements

- Model Hub (Technical Preview): Model Hub is a catalog of top-performing LLM and generative AI models. You can now easily import the models listed in the Model Hub into the Cloudera AI Registry and then deploy it using the Cloudera AI Inference service service. This streamlines the workflow of developers working on AI use cases by simplifying the process of discovering, deploying, and testing models.

  For more information, see Using Model Hub.
- Registered Models: Registered Models offers a single view for models stored in Cloudera AI Registry instances across Cloudera Data Platform Environments and facilitate easy deployment to the Cloudera AI Inference service service. When you import models from Model Hub, the models are listed under Registered Models. This page lists all imported models and associated metadata, such as the model's associated environment, visibility, owner name, and created date. You can click on any model to view details about that model, and its versions, and deploy any specific version of the model to the Cloudera AI Inference service service.

  For more information, see Using Registered Models.
- Cloudera AI Inference service (Technical Preview): Cloudera AI Inference service service is a production-grade serving environment for traditional, generative AI, and LLM models. It is designed to handle the challenges of production deployments, such as high availability, fault tolerance, and scalability. The service is now available for users to carry out inference on the following three categories of models:
  - TRT-LLMs: LLMs that are optimized to TRT engine and available in NVIDIA GPU Cloud catalog, also known as NGC catalog.
  - LLMs available through Hugging Face Hub.
  - Traditional machine learning models like classification, regression, and so on. Models need to be imported to the Cloudera AI Registry to be served using the Cloudera AI Inference service Service.

- Cloudera AI Registry Standalone API: Cloudera AI Registry Standalone API is now fully supported. This new API is available from the Cloudera AI Registry service to import, get, update and delete models without relying on the Cloudera AI Workbenchservice.

  For more information, see Cloudera AI Registry Standalone API.
- New Amazon S3 Data Connection: A new Amazon S3 object store connection is automatically created for Cloudera AI Workbenches to make it easier to connect to the data stored within the same environment. Other Data Connections can be configured to other S3 locations manually.

  For more information, see Setting up Amazon S3 data connection.
- Enhancements to Synced Team: Team administrators and Site administrators can now add multiple groups to a synced team, view members of a group, delete a group within a team, update roles for a group within a team, and update a custom role for a member within a group.

  For more information, see Managing a Synced Team.
- Auto synchronization of Cloudera AI Registry with a Cloudera AI Workbench: If you deploy a Cloudera AI Registry in an environment that contains one or more Cloudera AI Workbenches, the Model Registry is now auto-discovered and periodically synchronized by Cloudera AI Inference service service and Cloudera AI Workbenches and no manual synchronization is required. Cloudera AI Workbench is auto-synchronized every five minutes and Cloudera Cloudera AI Inference service service is auto-synchronized every 30 seconds.

  For more information, see Synchronizing the Cloudera AI Registry with a Cloudera AI Workbench.
- Environment: Support for Environment V2 is added for Cloudera AI Workbenches.
- Kubernetes: Support for AKS 1.29 and EKS 1.29 was added.
- Metering: Support for Metering V2 is added for new Cloudera AI Workbenches.

Fixed Issues

- DSE-35779: Fixed the issue related to a race condition between writing the JWT file by kinit container and reading by the engine container in the workload pod.
- DSE-37065: Previously, API V2 did not allow collaborators to be added as admin. This issue is now resolved.
- DSE-33647: Previously, workbench instances reset to default when upgraded. This issue is now resolved.

July 17, 2024

Release notes and fixed issues for version 2.0.45-b86.

Fixed Issues

- Previously, Cloudera Data Platform teams with the MLBusinessUser role were not available for Synced Teams in Cloudera AI workbenches. This issue is now resolved.

June 20, 2024

Release notes and fixed issues for version 2.0.45-b82.

New Features / Improvements

- You can now delete the Model Endpoint under AI Inference from the Cloudera Machine Learning UI.

Fixed Issues

- Previously, the Model Registry service was broken due to a change in the certificates. This issue is now resolved.

June 11, 2024

Release notes and fixed issues for CDSW version 2.0.45-b81 and MLX-CRUD-APP version 1.45.0-b71.

Fixed Issues

- Previously, users were unable to upgrade their model registry or create new model registries if their cluster was resized. This issue is now resolved.

May 29, 2024

Release notes and fixed issues for CDSW version 2.0.45-b81 and MLX-CRUD-APP version 1.45.0-b68.

New Features / Improvements

- The NodeSelector label can now be added for inference services. The label can be specified in the `instance_type` field of the deploy or update requests. This would enable you to direct inference service pods to specific nodes.
- Enhancements to the Export API to support the Observability APIs.

Fixed Issues

- Added support to create model registries in AWS with a public load balancer from the UI. Previously, this option was available only from the backend.

May 15, 2024

Release notes and fixed issues for version 2.0.45-b76.

New Features / Improvements

- Cloudera AI Inference Service (Technical Preview): AI Inference service is a production-grade serving environment for traditional, generative AI, and LLM models. It is designed to handle the challenges of production deployments, such as high availability, fault tolerance, and scalability. The service is now available for users to carry out inference on the following three categories of models:
  - TRT-LLMs: LLMs that are optimized to TRT engine and available in NVIDIA GPU Cloud catalog, also known as NGC catalog.
  - LLMs available through Hugging Face Hub.
  - Traditional machine learning models like classification, regression, and so on.

  Models need to be imported to the model registry to be served using the Cloudera AI Inference Service.

  For more information, see Using Cloudera AI Inference service.
- Cloudera Copilot (Technical Preview): Cloudera Copilot is a highly configurable AI coding assistant integrated with the JupyterLab editor. The Copilot improves developer productivity by debugging code, answering questions and generate notebooks.

  For more information, see Cloudera Copilot.
- Model Registry API (Technical Preview): New API is available from the Model Registry service to import, get, update and delete models without relying on the CML Workspace service.

  For more information, see Model Registry API.
- Ephemeral storage limit: The default ephemeral storage limit for CML Projects has been increased from 10 GB to 30 GB.

Fixed Issues

- Fixed an error that occurs while sorting public projects on the Jobs column.
- Fixed a bug that was uploading files to the root directory of a project instead to the specified subfolder.

April 25, 2024

Release notes and fixed issues for version 2.0.45-b54.

New Features / Improvements

- Model Registry - A model registry can now be upgraded with the Upgrade feature. Learn more: Upgrade model registry
- AMPs - various infrastructure improvements.
- API v2 - Modified model creation, build, and deployment endpoints more consistent with API v1, and support for CDV application creation was added.
- API v2 - Improvements made to List Projects endpoint, and other changes to support file uploads.
- Models - Model metrics are now supported with models that are deployed from a model registry.
- Kubernetes - Support for EKS 1.28 and AKS 1.28 was added.
- Workspace - The Create Workspace flow was improved to add validation of endpoint access and provide diagnostic responses.
- Azure - New Azure GPU instance types are supported: NC*A100, D16x v5, and D8s v5.
- AWS - New AWS GPU SKU is supported: p5
- Azure - The Middle East (Qatar Central) region is now supported.
- Runtimes - New Runtime Addons are released:
  - HadoopCLI 7.2.16.600
  - HadoopCLI 7.2.17.300
  - Spark 2.4.8
  - Spark 3.2.3
  - Spark 3.3.0

Fixed Issues

- DSE-28892 - Improved readability of Spark3 session startup WARN logs.
- DSE-35080 - Fixed an issue where Spark 3.2.3 configuration files were empty.
- DSE-35633 - Fixed an error that occurred when performing Create Project to create an AMP from a zip file.

February 20, 2024

Release notes and fixed issues for version 2.0.43-b229.

Fixed Issues

- DSE-33078 - Upgraded Go version to address CVE-2023-45285 and CVE-2023-39325.
- DSE-34663, DSE-34635 - Upgraded Grafana version and `kube-state-metrics:v1.9.3` image to address critical CVEs.
- DSE-32001 - Upgraded thunderhead components (`tgtgenerator`, `tgtloader` and `configtemplate`) to include CVE fixes that were carried out in `thunderhead`.
- DSE-34595 - Upgraded zookeeper version to address CVE-2023-44981.

March 6, 2024

Release notes and fixed issues for version 2.0.43-b233.

Fixed Issues

- DSE-34127, DSE-34692 - Allow MLUsers to create team projects. Project owner is added as a collaborator.
- DSE-35258 - Ensured that manual entries that were added to AMPs catalog sources in the past do not prevent upgrade of workbenches.
- DSE-35211 - Upgraded Traefik version to address CVEs: CVE-2023-44487.
- DSE-35270, DSE-34148 - Fixed the mlinfra scale-up issue during Resume and Upgrade Workflow.

February 8, 2024

Release notes and fixed issues for version 2.0.43-b220.

New Features / Improvements

- AMPs - The AMPs page has been upgraded to render images, make the UI more reactive and improve the overall experience.
- Azure - Added Azure Qatar Central as a supported region.

Fixed Issues

- DSE-33041 - Fixed a bug in API v2 to ensure that cron jobs created via API v2 are now adding the schedule correctly and not resulting in multiple instances of jobs running at any given moment.
- DSE-33793 - Fixed an issue in the UI to ensure that users are able to select the ‘Scala’ kernel while creating a new runtime.
- DSE-34166 - Made a change to ensure that auto-generated CML model sample code shown in the UI is not missing a quote.
- DSE-33082 - Upgraded the Go version to address CVEs: CVE-2023-45285 and CVE-2023-39325.
- DSE-34080 - Upgraded the UBI base images in CDSW to address multiple CVEs.
- DSE-33598 - Ensure that the error message about a missing workload password when JWT authorization is disabled is shown in the UI.
- DSE-34088 - Made a change to the renderSpark UI Executor Logs.

January 23, 2024

Release notes and fixed issues for version 2.0.43-b208.

New Features / Improvements

- Cloudera Data Warehouse - Automatic JWT-based authentication enables passwordless connectivity to CDW. Users do not need to use their workload password to query data from CML. This feature depends on Data Lake 7.2.18, please upgrade your environment when the new version is available.
- Redesigned AMP Catalog - The AMPs pane is redesigned to improve navigation and search capabilities.
- HuggingFace Spaces - A curated list of HuggingFace Spaces is available in the AMPs Catalog.
- Community AMPs - A selected list of community-created AMPs is available to run in CML in the AMPs pane.
- Azure - Support for new GPU instances: NVadsA10 v5-series (non-fractional)
- Azure - Certificate based authentication using Managed Identity to provision in AKS.
- AWS - Support for new GPU instances: g5
- AWS - Deprecated support for P2 instance types.

- AWS - Added support for CML workspaces in af-south-1, Africa (Cape Town) region.
- Kubernetes - Kubernetes version 1.27 is supported on both Azure and AWS.
- Restore workflow - Improved reliability of the workspace restore workflow.
- Private DNS Zone - CML now certified to work with private DNS zones.
- Project Migration tool - A command line argument is added to check if source and destination files are the same, covering job, app, model, project data and metadata files.
- Runtimes - The R version of cmladdon is upgraded to version 4.3.1.
- Runtimes - The HadoopCLI 7.2.17.100 Runtime Addon is released for the Public Cloud.
- Runtimes - ML Users can now register custom ML Runtimes.

  A new site-level configuration option has been introduced on the Site Administration page to enhance runtime registration capabilities:

  - Admins can enable a configuration option, granting users the capability to register runtimes.
  - The new option is disabled by default, ensuring that existing permissions remain unchanged.
  - With this update, we've implemented a mechanism to record the names of the users who register ML Runtimes, displayed on the Runtime Catalog Page.
  - Users now can add runtimes however, it's important to note that they are not permitted to deprecate/disable the added runtimes.
- Runtimes - We have introduced a new button in the Site Administration Runtime tab, where users are now able to update the Runtimes catalog at any time based on runtime repos.
- Runtimes - On the Project creation page:

  - Renamed the Runtime setup section to Runtime.
  - Updated the UI for Basic/Advanced option selection.
  - Modified basic configuration settings.
  - Added GPU-enabled runtime variant by default.
- Security - When adding project collaborators or team members, non-admins can be prevented from seeing the entire user list. This functionality can be restricted to Site Admins in Site Administration Security by selecting Allow all authenticated users to access /api/v1/users endpoint.
- Teams - The ownership structure for team creation is changed. Previously, the creator of a team was assigned the role of Owner. Now, the creator is assigned the role of Admin by default. Admins have the authority to add team members, including other admins. Each team must have at least one admin to manage the team.

Fixed Issues

- DSE-33545 - Fixed an issue where workspace shows as Ready even if workspace URL is returning 404.
- DSE-24423 - In Network Settings of a Workspace, removing last remaining allowed source IP range for Load Balancer throws error
- DSE-27910 - Fixed an issue where evicted pods caused CML backup to fail.
- DSE-23954 - Fixed a problem where load balancer subnet preflight validation is skipped if no load balancer is specified during workspace creation.
- DSE-23953 - Fixed a problem where worker subnet validation is skipped if no load balancer is specified during workspace creation.

December 15, 2023

Release notes and fixed issues for version 2.0.41-b238.

Fixed Issues

- ENGSEC-22894 - Fixed an issue with non-admin users being unable to add or remove collaborators on team projects.

- DSE-32894 - Fixed a problem with Model Registry causing the PATH variable to become too long and fail. See *Cloudera Customer Advisory (2023-725)* for more information.
- ENGSEC-23097 - Fixed an issue with CML upgrades failing when the control plane scales down nodes.

November 15, 2023

Release notes and fixed issues for version 2.0.41-b236.

Fixed Issues

- DSE-32250 - Spark Executors now report the correct create time and resource usage in the Site Administration Usage Export Usage List . Note that previously incorrect entries for Spark executors are reset to 0.
- DSE-32222 - On AWS, upgrading an NTP-enabled workspace to Kubernetes 1.25 AWS now incorporates various internal improvments.

October 19, 2023

Release notes and fixed issues for version 2.0.41-b225.

New Features / Improvements

- Model Registry - Model Registry is now GA. Model Registry is the core enabler for MLOps, or DevOps for machine learning. For more information, see Using Model Registry.
- Experiments - Experiments v2 is now GA. The Experiments feature now integrates with MLflow for managing the model lifecycle. For more information, see Experiments.
- Service Accounts - Service Accounts, which allow automated processes to run with their own user account, is now GA. For more information, see Service Accounts.
- Usage log tracking - Usage log records all workloads: sessions, jobs, models, applications and distributed compute, enabling administrators to export and analyze workload statistics on-demand.
- Kubernetes - Kubernetes 1.25 is now supported for EKS.
- Azure - New Azure instance types are supported: D4asv4, D16asv5 and D8asv5.
- Azure - On new installations, the nfs-csi-driver is now enabled.
- Azure - Cross-environment backup and restore of workspaces is now supported.
- Applications - Users can now see pod logs for applications. In Application Details, go to the Container Logs tab, and the pod logs are shown. Application and pod logs can be downloaded from the respective pages.
- Runtime Addons - CML now includes HadoopCLI Runtime Addon 7.2.15, and HadoopCLI 7.2.14 Runtime Addon is removed.

Fixed Issues

- DSE-30784 - Added more logging and event logging in Resume Workflow
- DSE-20734 - Fixed an issue where a machine user was shown as Unknown in events and log pages.
- DSE-30229 - Fixed an issue that caused CML upgrades to fail on AWS.
- DSE-29365 - Fixed an issue where the YQ version checker script was not working for latest versions of `yq` in the CML application.
- DSE-28187 - Fixed an issue so that Retry Workspace Installation is disabled if Liftie Provisioning fails.
- DSE-27000 - Upgraded cdp-js to version 4.2.5 in the CML application.
- DSE-26069 - Fixed an issue so that deleted and failed Spark addons do not appear as selectable addons in the New or Modify Workload pages.
- DSE-26966 - Added Datalake preflight checks to the upgrade API.

August 31, 2023

Release notes and fixed issues for version 2.0.40-b157.

Fixed Issues

- AWS : Non-Transparent Proxy Support

  Fixed issues with supporting non-transparent proxy related to web pod connectivity to control plane, image-puller, pod to pod connectivity and api pods missing configuration.
- Upgrades
  - Fixed an issue where workspace upgrade fails due to not deleting orphan pods.
  - Fixed an issue where workspace upgrade fails due to incorrect error handling.
- Fixed an issue where the Team Sync tab was disabled.
- Azure
  - Added a preflight check when creating NTP clusters: Azure does not support NTP.
  - The NCv2 series is no longer supported due to its upcoming end of support on September 6, 2023. See TSB-692 for details. The default GPU for creating Azure workspaces will now be the `Standard_NC6s_v3`. This new GPU offers nearly identical configurations to the previous default GPU. If you are currently using NCv2 series GPUs, it is recommended to make the necessary adjustments before the end of support date.

July 25, 2023

Release notes and fixed issues for version 2.0.40.

New Features / Improvements

- CML Home - A new landing experience that helps you to jump to your most recent Projects, walks you through the key capabilities of the platform and keeps you up-to-date with the latest developments.
- 3rd-party editor support - The PBJ architecture adds support for 3rd party editors, enabling building custom ML Runtimes from scratch with JupyterLab, RStudio, and other editors of your choice.
- Models - Models can now be deployed from the model registry to CML workspaces using API v2.
- Kubernetes - Kubernetes 1.26 support is available for Azure.
- Model Registry (Preview) - Model Registries can now be deployed on an Azure private cluster.
- CML Scalability - CML Control Plane flows have been verified for 100-node clusters and high volume workloads are now enabled by default.
- Project Migration - Support for project export and import from CDSW to CML public or private cloud environments, or for migration between development and production environments.
- Retry Install Workspace - Workflow-based support for retry of CML workspace creation, in the event workspace installation fails.
- Preflight Checks for Instance Groups - Pre-flight checks are run when an instance group is being modified to ensure the requested configuration is valid."
- Preflight Checks for Update Workspace - Pre-flight checks are run when a workspace update is being requested to ensure the requested change is valid.
- SDX - SDX 7.2.17 has been integrated and verified with CML.
- Runtimes - On the New Project page, CML code now defaults to the Python 3.9 ML Runtime Edition.
- HadoopCLI - The DL versions of HadoopCLI 7.2.8, 7.2.10 and 7.2.11 Runtime-Addon versions on Public Cloud reached End of Support and have been removed.
- Job notifications - All email-related control will be hidden by job creation or job settings if the SMTP host is not configured. If email recipients have been added previously to the job but the SMTP host is not configured, the Job Notifications section will be displayed as a warning message, informing the user of the problem.
- Project - Site administrators can now restrict project creation for users and/or teams.

- Environment variables - Users can now hide the values of sensitive environment variables on the Account/Project/Workload/Workspace level.

Fixed Issues

- Upgrade workspace (DSE-28827) - Fixed an issue where after upgrading a workspace, the workspace (such as instance type and autoscale range) could not be modified.
- Pod (DSE-28771) - Fixed an issue where the health-poller pod was crashlooping due to a missing UMS_HOST environment variable.
- Workspaces (DSE-28595) - Fixed an issue where workspaces with static subdomains would not register freeIPA DNS entries were not added correctly, for both created and restored workspaces.
- Installation (DSE-28160) - Fixed an issue where an installation is not marked 'Fail' when it has timed out for 3 hours. This happened on Azure private cluster (UDR) installations.
- Installation (DSE-28047) - Fixed an issue where when a failed installation on Azure is retried, pre-install validations are not performed again, which can lead the installation to eventually time out.
- Environmental variables (DSE-28005) - Fixed an issue where environment variables were not visible or modifiable on an application's settings page.
- CML Service Link (DSE-28072) - This change enables users to use the CML Service Link on the project session pages started either with Workbench or with the remote editor.
- Workspace details (DSE-27891) - Fixed an issue where Workspace Details shows some incorrect information (Creator, User, Workspace tags) when the workspace is created via a machine user.
- Project (DSE- 27516) - Fixed an issue where project creation in a workspace on Azure was failing due to an NFS error.
- Retry Install Workspace (DSE-26944) - Improved the language in the Retry Install Workspace modal.
- Preflight (DSE-26752) - Improved the instance preflight failure message to mention that validation is not skippable.
- Spark addons (DSE-26269) - Fixed an issue where deleted and failed Spark Addons appeared on New/Modify Workload pages as selectable Addon.
- CDP CML (DSE-26012) - Prefix and update all workload usage events with CDPCML.
- CML consumption (DSE-25951) - Made improvements to issues that caused discrepancies in measurement of CML consumption.
- Status icon on Runtime Catalog Page (DSE-24589) - This change enables users to find out if there is at least one runtime among the runtime variants with the enabled status.
- PBJ Workbench (DSE-21770) - In PBJ Workbench Sessions, history navigation (with UP key) is working now, and users are able to navigate to previously executed commands with this key.
- GPU (DSE-29159) - Fixed an issue where `P4d.24xlarge` GPU is displayed as a CPU, not GPU.

July 12, 2023

Release notes and fixed issues for version 2.0.38-H3.

Fixed Issues

- Team Sync (DSE-29009) - Fixed an issue where Team Sync did not work in AWS regions other than us-west-1.

May 31, 2023

Release notes and fixed issues for version 2.0.38-H2.

New Features / Improvements

- Embedding applications - CML applications can now be embedded in i-frames in another domain. See Embed a CML application in an external website for more information.

- Team collaborator - It is now possible to add a team account as a collaborator on a project. This feature is supported via UI as well as API. For more information, see Adding project collaborators.
- Project owner - It is now possible for an administrator to change the owner of a project. This feature is supported via UI as well as API. For more information, see Modifying project settings
- Load Balancer - You can now select from the environment's endpoint access gateway subnets in the Subnets for Load Balancer field.

Fixed Issues

- Projects (DSE-27795) - Fixed an issue where certain public projects owned by a user or where the user is a collaborator were not displayed in the user's My Projects view.
- Jobs (DSE-27535) - Fixed an issue so that a job can be created in disabled or paused state.
- API v2 (DSE-27463) - Added missing fields (timezone, paused) to createJobRequest endpoint.
- API v2 (DSE-27442) - Fixed an issue so that files in subfolders are uploaded to their respective subfolders, and not to a root directory, when using the uploadFile endpoint.
- API v2 (DSE-27391) - When creating a project via API v2, the ephemeral storage value is now set to NULL (instead of zero). This fixed an issue that was blocking certain CDSW to CML migration workflows.

May 16, 2023

Release notes and fixed issues for version 2.0.38-H1.

New Features / Improvements

- Private Cluster - Private clusters can now be created on AWS. For more information, see Private cluster support.
- Spark - INSERT events are now fired via HMS after a DML action from Spark

Fixed Issues

- Workspace (DSE-27291) - Fixed an issue where Instance Group Modification failed with an older version workspace.
- Workspace (DSE-27406) - Fixed an issue where SkipValidation was not observed when restoring a workspace.
- UI (DSE-27259) - Fixed an issue where the incorrect resource details were displayed for an instance type.
- Workspace (DSE-9552) - Fixed an issue to allow provisioning of a workspace with more than 30 nodes.

April 26, 2023

Release notes and fixed issues for version 2.0.38.

New Features / Improvements

- Custom Data Connections - Site administrators can now configure access to external data sources with the new Custom Data Connections feature. Data Scientists can access the external data via the `cml.data` library and its 2-liner abstractions.
- Add Data - Data Scientists can now upload files to Hive and Impala Virtual Warehouse tables from the Data tab of any CML Project.
- Model Replicas - Site Administrators can now configure the maximum number of model replicas that users can select for their models via the Maximum Model Replicas field on the Administration # Settings # Model Deployment Settings page.
- Models - Users can now deploy large ML models. Model size is not limited to 50 MB anymore.
- Model Registry - Administrators can now find the Machine User Workload User Name that is needed for configuring their model registries to access their S3 or ADLS Gen2 bucket on the Workspace Details page.
- Install Workspaces (Tech Preview) - Auto-retriable workflow for Install Workspace for Azure Private Cluster. See the Feature Preview doc for more information.

- Usage Monitoring - Usage data for Spark Executors is now recorded in the CSV file that Site Administrators can download from the Administration Usage tab.
- API Keys - Improved security by storing Legacy API keys as hashes in the database. Existing Legacy API keys are automatically rotated as part of the upgrade process to ensure that previous keys cannot be used. API v1 keys will not be usable after the upgrade. This does not impact the Models API, and Models authentication is not affected. To manually rotate a Legacy API key, do the following:
  - In User Settings API Keys , click Rotate to generate a new Legacy ApiKey and ApiKeyHash pair.
  - Copy the Api Key that is shown after rotation and use it in future requests.
  - Note - The API key will not be visible on the UI once you refresh the page. Make sure to copy it before leaving the page.
- Modify Instance Group Type - Administrators can easily change the CPU or GPU instance types of node groups for a CML workspace, without having to re-provision the workspace.

Fixed Issues

- Model Registry (DSE-25909) - Removed the 19 character limit for model names in Model Registry.
- Model Registry (DSE-25641) - Fixed an issue where redeploying models from model registry to a CML workspace may fail.
- Model Registry (DSE-24683) - Added workload name to Model Registry service account list.
- Model Registry (DSE-25906) - Model Registry is no longer limited to listing 10 models at a time.

April 5, 2023

Release notes and fixed issues for version 2.0.36-H4.

Fixed Issues

- Azure (DSE-26517) - Fixed an issue that caused a failure when upgrading CML Azure workspaces to AKS 1.25.

March 27, 2023

Release notes and fixed issues for version 2.0.36-H3.

New Features / Improvements

- Kubernetes - EKS 1.24 (AWS) and AKS 1.25 (Azure) are now supported.
- AWS - The eu-south-1 region is now supported.

Fixed Issues

- Entitlements (DSE-25941) - Fixed an issue where regional control planes could not access entitlements.

February 14, 2023

Release notes and fixed issues for version 2.0.36-H2.

Fixed Issues

- Model Registry (DSE-25331) - Fixed an issue where users may see an error message x509: Certificate signed by unknown authority on the Model Registry page in the CML workspace application. Note that if you are seeing this error, or if you create a model registry after you create one or more CML workspaces in an environment, you must synchronize the model registry with the workspaces.

February 10, 2023

Release notes and fixed issues for version 2.0.36-H1.

New Features / Improvements

- Model Registry (Tech Preview) - The Model Registry is a new CML Product to store and manage machine learning models and associated metadata, such as the model's version, dependencies, and performance. The registry enables MLOps and facilitates the development, deployment, and maintenance of machine learning models in a production environment.

  Note: Model Registry is not supported with R models.

February 7, 2023

Release notes and fixed issues for version 2.0.36.

New Features / Improvements

- Custom Runtime Addons - This feature enables administrators to mount shared dependencies like connection drivers or configuration files to all CML workloads. You can start by following the documentation.
- Cancellable CML Workspace Backup - Administrators now can cancel in-progress backups and resume the CML Workspace.
- AWS Support - The AWS Jakarta and Hong Kong regions are now supported.
- Kubernetes - EKS 1.23 (AWS) and AKS 1.24 (Azure) are now supported.

Fixed Issues

- Applications (DSE-22857) - Fixed an issue where unauthenticated users could access the terminal session in a running public application if the application was configured to allow anonymous access to the terminal endpoint. Now anonymous access to the terminal endpoint is redirected to an authentication page.
- Applications (DSE-24105) - Fixed an issue where the Application List page may not load correctly when there are a large amount of Applications accessible by the user.

November 29, 2022

Release notes and fixed issues for version 2.0.34.

New Features / Improvements

- Suspend/Resume Workspace - Administrators can optimize cloud costs by suspending CML Workspaces for weekends and resume operation for business hours. See Suspend and resume ML workspaces for details.
- PBJ Runtimes - PBJ Workbench Runtimes are now GA, providing a more consistent experience with the Jupyter ecosystem. With the elimination of proprietary code you can now build a new Runtime from scratch, on your custom base image with your selected kernel, with no need to build on top of a Cloudera image anymore.
- Experiment tracking - A new Experiments feature built on MLFlow is now available. You can now track your experiments in CML sessions with the preinstalled mlflow library and visualize and compare them in the CML application. This feature is enabled by default.
- Customizable Scratch Space - You can now configure the amount of ephemeral storage space (also known as scratch space) a CML session, job, application or model can use. This feature helps in better scheduling of CML pods, and provides a safety valve to ensure runaway computations do not consume all available scratch space on the node. By default, each user pod in CML is allowed to use up to 10 GB of scratch space.
- Iceberg support - Iceberg v2 is supported via Spark Runtime Addon, based on the CDE 1.17-h1 Runtime Addon version.

- Jobs UI - The Jobs List page and Job Details page now display the job ID and Created At time.
- Projects UI - When you select a runtime kernel on the project creation page, only the latest standard version of the selected kernel is added to the project.
- Data tab - The Cloudera Data Visualization application that provides the Data tab experience got upgraded to v7.0.2. You can review the changes here.
- Applications - Custom polling endpoints for Applications can now be specified in the UI.
- Register new runtimes via APIv2 - Site administrators can register new ML Runtimes using the APIv2.
- Runtime addon management - Site administrators can now also disable or deprecate specific Runtime Addons.
- Environmental variables - The new project environmental variable PROJECT_OWNER holds the username of the project owner.

Fixed Issues

- Python packages (DSE-21313) - Fixed an issue where Python packages installed via pip install may not be imported correctly until a new session is started.
- Jobs (DSE-21771) - Fixed an issue where Jobs scheduled with PBJ Runtimes may terminate with a Success state even when there were errors reported in job runs.
- PBJ Runtimes (DSE-19971) - Fixed an issue where comment blocks may not be rendered correctly in sessions or jobs launched with PBJ Runtimes.
- Models (DSE-22527) - Fixed an issue where the Model Monitoring Chart was displaying data from other projects or deployments.

October 19, 2022

Release notes and fixed issues for version 2.0.32-H4.

New Features / Improvements

- In-place Upgrades - In-place upgrades are now available for CML versions 2.0.29 and 2.0.30.
- Azure backups - When creating snapshots for an Azure backup, the resource group name defined in the environment can be used (if present) as part of the snapshot name.

Fixed Issues

- Environmental Variables (DSE-23098) - The environment variable WORKLOAD_PASSWORD is redacted in logs.
- Environmental Variables (DSE-23499) - Environment variables are redacted from grpc request logging.
- CVE (DSE-23534) - Upgraded several libraries to fix critical CVE vulnerabilities.
- UI (DSE-22128) - Improved behavior of New Project and New Session buttons.
- Secret Generator (DSE-22696) - Fixed an issue that was causing secret generator slowness.

September 27, 2022

Release notes and fixed issues for version 2.0.32-H3.

New Features / Improvements

- Iceberg support - CML Snippets now fully support the Iceberg v1 table format for all Spark, Hive, and Impala data connections. See Create an Iceberg data connection for details.
- Data connections - Data Hubs are now automatically discovered and connections are created for them in newly-created CML Workspaces.
- Backup / Restore Workspace - The default timeout is now 12 hours, and the estimated time to complete a backup (from the cloud provider) is now periodically added to the event logs.

August 30, 2022

Release notes and fixed issues for version 2.0.32-H2.

New Features / Improvements

- Private Cluster on Azure - This option is temporarily disabled. (CDPAM-3279)
- Backup / Restore Jobs - Timeouts can now be customized in both the Backup Workspace and Restore Workspace UI.

Fixed Issues

- Workspace Installation (DSE-22545) - Increased the workspace installation timeout limit to 1 hour from 30 min
- CML Workspace (DSE-22638) - Fixed an issue where CML workspace creation using a machine user caused an error.

July 21, 2022

Release notes and fixed issues for version 2.0.32.

New Features / Improvements

- Garbage collection for deleted projects - This feature allows you to trigger cleanup of deleted projects. A separate feature allows older orphaned projects to be marked for cleanup. For more information, see Project Garbage Collection.
- Disable Runtimes - It is now possible to disable and enable runtimes. For more information, see Disabling Runtimes.
- Monitoring for Applications - This feature allows you to monitor the technical health of deployed Applications, including statistics and visualizations of CPU and memory usage. For more information, see Monitoring Applications.
- Custom polling endpoints for applications - This feature allows the application creator to define what application endpoint servers poll to detect if the application is running, that avoids problems some applications have with polling the root endpoint. For more information, see Application polling endpoint.
- PBJ Workbench Runtimes (Tech Preview) now work with Sessions, Experiments, Jobs and Applications - This feature enables the classic workbench UI backed by the open-source Jupyter protocol. This architectural change improves consistency, stability, and ease of customization while eliminating the dependency on proprietary CML code. For more information, see *PBJ Workbench* in Preview Features.
- Kubernetes - Kubernetes 1.22 is now supported for both AWS and Azure.

Fixed Issues

- Job quotas (DSE-12664) - Fixed an issue where subsequent jobs in a job pipeline can fail if quota is enabled.
- HDFS via Python (DSE-19775) - Fixed an issue where accessing HDFS via Python libraries that connect natively to HDFS, such as Tensorflow or PyArrow, may fail due to an error that the libhdfs.so file cannot be found.

May 31, 2022

Release notes and fixed issues for version 2.0.30.

New Features / Improvements

- ML Discovery and Exploration, SQL and Visualization (GA) - This feature enables Data Scientists to understand their data using a SQL editor and drag-and-drop Visual Dashboards within CML. Users can start with their

pre-configured Data Connections and create Datasets that they can rely on for model development. For more information, see ML Discovery and Exploration.

- CML Endpoint Stability - This feature adds the ability for CML admin to define the prefix for the URL for the CML Workspace. This enables a new CML Workspace to be created and leverage the endpoint of a previously deleted CML Workspace. This ensures that deployed models and applications deployed in the new Workspace will have the same endpoint as the same models and applications deployed in the old Workspace. For more information, see *CML Static Subdomain* in Provisioning ML Workspaces.
- Add/Delete GPU Nodes - This feature enables MLAdmins to reconfigure CML Workspaces by adding or removing GPU Worker groups for existing deployments.

April 21, 2022

Release notes and fixed issues for version 2.0.29.

New Features / Improvements

- ML Discovery and Exploration, SQL and Visualization (Preview) - This feature enables Data Scientists to understand their data using a SQL editor and drag-and-drop Visual Dashboards within CML. Users can start with their pre-configured Data Connections and create Datasets that they can rely on for model development. For more information, see *ML Discovery and Exploration*.
- Model metrics visualization - This feature allows Data Scientists and Machine Learning Engineers to monitor technical metrics relating to their running models, such as resource consumption and request throughput, within Cloudera Machine Learning.

Control Plane Application

- Azure Files - Support for Azure Files NFS - TP
- Public Load Balancer - This feature allows MLAdmins to configure a Public Load Balancer for a CML workspace but with fully private (Private EKS API Server endpoint) EKS deployment (AWS only).

Fixed Issues

- Pagination widget (DSE-19937) - Fixed an issue where the pagination widget on the Session list page may not function as expected.
- Notification emails (DSE-20085) - Fixed a bug where Job report recipients who subscribed to notification emails when their jobs terminated, may receive notification emails for termination statuses that they did not subscribe to.
- Project list page (DSE-1975) - Fixed a bug where projects may not be sorted correctly on the project list page when users use the Created By field for sorting.

Related Information

Provisioning ML Workspaces

March 14, 2022

Release notes and fixed issues

New Features / Improvements

- Site Administration - The Project Creation Timeout (minutes) settings on the Site Administration Settings page now applies to project creation via both git clone and fork. Consider increasing this value in case you need to create a large project. If you have tried forking projects before doing this and they get "stuck," the only way to delete these projects is to manually navigate to the <user>/<project-name> URL, and select Project Settings Delete .

- Load Balancer subnet - Support specifying subnets for the Load Balancer created by CML on AWS. The Workspace Provisioning page now has a field for specifying a subnet for the Load Balancer. See *Provisioning a Workspace* for more information.

Fixed Issues

- Editor (DSE-11423) - Fixed an issue so that the editor, session, or workbench windows resize when the browser window resizes.

February 10, 2022

Release notes and fixed issues

New Features / Improvements

- ML Discovery and Exploration - Data Connections and Snippets are now Generally Available. CML workspaces now automatically discover data connections within the CDP environment and offer connection snippets for users. For more information, see ML Discovery and Exploration.
- ML Runtimes - You can now filter the list of ML Runtimes that can be used in a given project.
- Model Technical Metrics visualization is now available in CML (Technical Preview).
- API v2 - You can now specify an input data example when you create a model build.
- Backup and Restore - CLI-based Backup and Restore of CML workspaces is now available as a Preview Feature. AWS only.
- Kubernetes - Kubernetes 1.21 is now supported on Azure.

Fixed Issues

- API v2 (DSE-18782) - You can change the Runtimes and Spark versions (specified via the runtime addon setting) associated with a job, by using the Update Job API endpoint.

January 12, 2022

Release notes and fixed issues

New Features / Improvements

- AWS - Resolved an issue with the way certain AWS regions (such as eu-south-1) identify region specific endpoints, which caused workspace creation to fail.

December 15, 2021

Release notes and fixed issues

New Features / Improvements

- CML - Released remediation for CVE-2021-44228, the Apache Log4j2 vulnerability. See CVE-2021-44228 Remediation for CML for details.
- Engines / Runtime Addons - Released versions containing fixed version of Apache Log4j2.

December 13, 2021

Release notes and fixed issues.

New Features / Improvements

- What's New - CML Users can now discover and read about new features, blog posts, FFL research reports from within CML without leaving the product.
- Non-Transparent Proxy (Preview) - Non-transparent proxy support is now available for AWS.
- API v2 - API v2 now supports Spark 2 and 3 via ML Runtime Addons.
- Spark - Spark Dynamic Allocation Lite is now supported in CML.
- Projects - Job lineage/dependencies are now maintained when forking a project.
- User Management - CML Admins can now create CML Teams that have their membership synchronized with CDP Groups.
- User Management - Users who no longer have access to a Workspace are deactivated within CML after a user sync.
- Unified Diagnostics - Unified diagnostics are now integrated in CML support bundles.

Fixed issues

- Username display - Fixed an issue where the user's full name was left blank if it was not available from the identity provider. Now, if it is not available, the username is displayed next to the avatar.
- Terminal access button spins forever (DSE-18456) - Fixed an issue where the terminal access button on an Azure cluster failed to become ready.

October 27, 2021

Release notes and fixed issues.

New Features / Improvements

- ML Flow - Cloudera Machine Learning now supports experiment tracking using open MLflow standards. For more information, see *CML Experiment Tracking through MLflow (Preview)* in Preview Features.
- Spark 3 - Users can now use Spark 3 in Sessions, Jobs, Models or Applications in their projects configured to use ML Runtimes.
- cdswctl - Users can now use the cdswctl CLI client to create Sessions with Spark.
- Root volume size - On existing CML workspaces based on AWS, there is a known issue where the provisioned root volume capacity is not enough to accomodate CML deployments with K8s versions >= 1.21. To mitigate this issue, when upgrading existing workspaces with a K8s version older than 1.19 on AWS, the root volume size is automatically increased/updated to 128 GB. Please note that this is not an issue on Azure where the pre-provisioned capacity for workspaces is already 128GB
- Workbench Editor - The Workbench now supports wrapping lines in the editor window. Users can enable this by selecting the Line Wrapping Enabled option under the View menu item in the Workbench.
- Private AWS clusters - Support for Private AWS clusters is now available in Preview mode. With Private AWS cluster support, CML workspaces are provisioned with a Private K8s API server making the deployments more secure.
- Pre-flight checks - NFS Server pre-flight checks have been added for Azure where validations are done to make sure that the specified BYONFS server directory meet the following criteria NFS server directory specified should be:
  - Reachable
  - Mountable
  - The directory must be owned by CML user 8536.
  - The directory must be empty.

Fixed issues

- Atlas (DSE-16706) - Fixed an issue with reporting data lineage of CML model deployments to Atlas.

October 18, 2021

Release notes and fixed issues.

New Features / Improvements

- Kubernetes 1.20 - Kubernetes 1.20 is now supported on both AWS and Azure.

Fixed issues

- Workspace email (DSE-17615) - Fixed an issue where sending email from a CML workspace failed.
- Sessions (DSE-17776) - Fixed an issue where forking a project with a job failed.
- Data lake upgrade (DSE-17687) - Fixed an issue where upgrading the data lake failed if it had been replaced after it was initially installed.

August 31, 2021

Release notes and fixed issues.

New Features / Improvements

- API v2 - A new API for operations on projects, jobs, models, and applications is now generally available.
- ML Workflow Discovery and Exploration (Preview) - This feature accelerates the ML development workflow with preconfigured data connections and readily available snippets. See ML Discovery and Exploration for more information.
- Kubernetes 1.19 - Kubernetes 1.19 support has been added on AWS.

Fixed issues

- RStudio editor issue (DSE-16843) - Fixed an issue where users may see a blank page when launching Sessions with RStudio editor using legacy engines.
- Spark (DSE-17279) - Fixed an issue that could cause sessions to fail to start if the Enable Spark option was not selected.
- SSH tunnels (DSE-17406) - Fixed an issue where users could not create a new SSH tunnel on the Project Settings > SSH tunnels page.
- GPU node (DSE-17114) - The default GPU node was changed to a p3 generation instance.

August 23, 2021

Release notes and fixed issues.

New Features / Improvements

- AMPs and Runtimes - Runtimes requirements for tasks can now be specified in AMPs.
- API v2 - A new API endpoint for creating models is available.
- API v2 - An ephemeral API key, which is valid during the lifespan of Sessions/Jobs/Applications/Experiments, for accessing APIv2 endpoints is now available in the environmental variable CDSW_APIV2_KEY.
- Customer Master Keys - CMK encryption is now supported for AWS environments. For more information, see CMK Encryption on AWS (Preview) in Preview Features.
- User-level environmental variables - Users can now set environmental variables at the user level, which are available in all Sessions, Jobs, Applications, Experiments, and Models launched by the user.
- Session launch UX improvement - The console output page now shows autoscaling information of the CML workspaces when scheduling new Sessions, Jobs, Experiments or Applications

Fixed issues

- ML Runtimes (DSE-16429) - Fixed an issue so that warnings are displayed for models during runtime migration in the case of an engine type mismatch.
- ML Runtimes (DSE-14834) - Fixed an issue so that when workers are added with ML Runtimes, they are initialized with the correct runtime
- ML Runtimes (DSE-16163) - Fixed an issue where the New Session > Configure button opened the Runtime/Engine page, even if the Runtimes feature was disabled.
- ML Runtimes (DSE-16790) - Fixed an issue where the Add Runtime button on the Runtimes Catalog page is displayed for regular users.
- Spark (DSE-16714) - Enabled/disabled state of a Spark addon is now remembered across sessions.
- Grafana dashboards (DSE-16615) - Fixed an issue where Grafana dashboards did not appear on public cloud workspaces.

July 8, 2021

Release notes and fixed issues.

July 8, 2021

Fixed issues

- Large files (DSE-16884) - Fixed a bug that could truncate large file uploads or lose changes to large files.
- Custom editors (DSE-16570) - Fixed a bug that could cause unexpected UI refreshes when using custom editors such as JupyterLab.

June 24, 2021

Release notes and fixed issues.

June 24, 2021

- Engine Deprecation - Cloudera ML Runtimes are the default and recommended solution to run user workloads. New projects will be created with ML Runtimes configured by default and we recommend migrating existing projects to use ML Runtimes. Legacy Engines are deprecated and will be removed in a future release but workloads running on them remain fully supported.
- Register customized Runtime - Administrators can register an externally built Runtime to provide Data Scientists with a customized environment.
- Support user API key rotation - Administrators can rotate keys for all users, or users can rotate their own keys, just by clicking on a button.
- AMP specification for Runtimes - You can specify which Runtimes to use in AMPs.
- New base engine released - Engine:14-cml2021.05-1 is now available.
- Configurable engine image for Jobs - You can specify which engine to use for a Job. Jobs use the Project engine by default.
- Web session timeouts - Timeout limits for User and Admin User web sessions were changed to be a period of inactivity, instead of a set time limit.

Fixed issues

- Spark configuration (DSE-16422) - Fixed an issue where user-added Spark configurations in the ~/spark-defaults.conf file may not be populated to /etc/spark/conf/spark-defaults.conf correctly.
- Hive table (DSE-16108) - Fixed an issue where saving data to a managed Hive table using Hive Warehouse Connector via CDSW from CML may fail with a StreamCorruptedException.
- API keys (DSE-15678) - Fixed an issue where user API keys may be accessible through non-private projects.
- Web sessions (DSE-11394) - Fixed an issue where user web session may not be refreshed even when the user is actively using CML.

Related Information

Managing ML Runtimes

Pre-installed Packages in Engines

May 13, 2021

Release notes and fixed issues

May 13, 2021

New features

- Business User Experience - A new user role, ML BusinessUser, provides restricted access to view Applications created in CML.
- ML Runtimes - Runtimes provide a lightweight alternative to Engines.
- View All Applications - Admins can view all applications on the application list page.
- Jobs improvements - Job Scheduling UI now supports cron expressions, job notification email subject line now includes the project name, and the Job History page now shows detailed start and end timestamps.

Fixed issues

- DSE-15059 - Fixed an issue where project creators may not be able to authenticate to models created by project collaborators using model API key.

March 22, 2021

Release notes and fixed issues

March 22, 2021

New features

- CML Model DNS resolution - For CML on Azure and AWS, DataHub and CDE connectivity to CML services such as workspaces and model endpoints is now possible. This is supported by enabling internal DNS resolution for CML workspace endpoints and models on private IPs. However, any existing workspaces must be upgraded to the current release to obtain this functionality.
- RAZ integration - CML now integrates with RAZ for S3.
- Cdswctl support - ML Runtimes is now supported in `cdswctl`.
- External engine repositories - Admins can now add basic Docker credentials for external engine repositories.
- ML workspace provisioning - In case of validation errors, it is possible to skip preflight validation for ML workspace provisioning. See Provisioning ML Workspaces for more information.
- Cloud permissions - CML now supports configuring AWS environment cross-account roles with fine-grained cloud permissions and policies for provisioning and running workspaces. See *AWS restricted policies* for more information.

Fixed issues

- DSE-14481 - CML no longer needs Network Security group ports to be open for EFS on AWS (port 2049).
- DSE-14685 - Clarified impact of changing Load Balancer Source Range settings on Security Group settings. See Provisioning ML Workspaces for details.

February 3, 2021

Release notes and fixed issues

February 3, 2021

New features

- Applied ML Prototypes - Applied ML Prototypes (AMPs) are now generally available inside the product, which provide end-to-end prototypes to help kickstart real customer use cases. AMPs effectively package pre-built models for data scientists to tailor for their own needs, as well as enable data scientists to learn how the different parts of CML work together.
- Kubernetes Support - CML AWS now supports K8s 1.18, keeping up with updated EKS support.
- NFS version picker - In Provision Workspace, the NFS version field was changed to a pick list with the default version set to 3. Make sure that this is the correct version to use, otherwise the workspace installation will fail.

Fixed issues

- DSE-13866 - Export Session List includes duration information.
- DSE-13254 - Python 2 kernel is enabled or disabled on Job Update and Create/Update Application pages based on Admin setting.
- DSE-14251 - Fixed an issue where a workspace could not be upgraded if its configuration was modified.
- DSE-13928 - Fixed an issue where revoking a user's access to an Application may not take effect immediately.
- DSE-14170 - Fixed an issue where TLS certificate renewal may not succeed for ML workspaces that have been upgraded.

December 21, 2020

Release notes and fixed issues

December 21, 2020

New features

- Added support for re-configuring ML workspaces:
  - Autoscaling Group (ASG) limits
  - Authorized IP ranges for k8s api server
  - Load Balancer Source Ranges and allowed list
- AWS and Azure instance types expanded - Support for additional AWS and Azure instance types

Fixed issues

- DSE-13904 - Fixed an issue where CML workspace installation may take up to 10 minutes longer when the Autoscaling Group minimum is set to 0.
- DSE-13898 - Updated TGT image to fix an issue so that the freeIPA HA release retries on failure.
- CDPSDX-2673 - Added a Retry step to the login context to reduce the chance of PRE_AUTH failures.

November 23, 2020

Release notes and fixed issues

New features

- Updated Projects Dashboard - Upon logging in to CML, users have a new streamlined page to be able to access their Projects. Projects can be displayed in a Summary card-based view for direct access, or in a Detail table-based view to be able to jump directly to different workloads (Sessions, Experiments, Models, etc) within a particular Project. In addition, the Resource Usage Details can be displayed or hidden as needed.
- Scaling Workers on Azure - On Azure, the number of worker nodes can now be scaled to zero when the workspace is idle.
- ML Runtimes - As an alternative to the existing Engines in CML, ML Runtimes are more lightweight than the current monolithic Engines. By specifying the desired Editor, Kernel, Edition, and Version, a streamlined Runtime will be used to run the user's code in Sessions, Jobs, Experiments, Models, or Applications.
- Asynchronous Project Creation - New Project creation, in particular via Git or forking from an existing project, now executes in the background so that the user does not have to wait.

- Sessions List Export - The date format on the Sessions List Export has been updated to include the complete date plus hours, minutes, seconds and a decimal fraction of a second.
- Custom root volume size - The root volume size of CML workspaces can now be specified on the Provision Workspace UI. Also, the default root volume size of CML workspaces on AWS has been increased from 40 GiB to 96 GiB.

Fixed issues

- DSE-13246 - Fixed an issue where cdswctl version always returns "unknown".
- DSE-11685 - The Export PDF function is improved and will display charts, tables, and maps. However, some content still might not display well, such as tables with many columns and dynamic content, video, or iframe content containing dynamic content.

October 29, 2020

Release notes and fixed issues.

New features

- Scaling workers on AWS - On AWS, the number of worker nodes can now be scaled to zero when the workspace is idle.
- Kubernetes - Access to the Kubernetes API server of the ML workspaces can now be restricted to certain authorized IP ranges.
- ADLS Gen2 - Access to ADLS-Gen2 from CML can now be managed with Ranger policies.
- Subnets - Subnets used for ML workspaces can now be reviewed in the Workspace Details page.
- ML workspace upgrade - You can now retry ML workspace upgrade if the previous attempt fails.
- Deprecated Kubernetes versions - ML workspaces with deprecated Kubernetes versions will now be highlighted on the CDP control plane. It is recommended to upgrade your ML workspaces with the deprecated Kubernetes versions at the earliest opportunity.
- Enable Unauthenticated Access To Applications - Users can now create Analytical Applications in CML that can be accessed by unauthenticated users. This functionality can also be wholly disabled for an ML workspace by an administrator.
- Applications header - CML now supports authentication using REMOTE_USER HTTP header for the Applications created on CML.
- Audited events - Quota changes and “Download All” files are now audited in the user_events table.
- Sessions page redesign - The Sessions page has been improved, making it easier to take actions such as filtering down to running sessions and performing bulk deletion of old sessions.

Fixed issues

- DSE-12164 - Fixed a bug where applications may not use the engine image configured with the project upon restart.
- DSE-12033 - Fixed an issue where the Engine profile on the Job Overview page did not update when the profile was changed on the Job Settings page.
- DSE-12026 - Grafana deployments now include 8536 in the supplemental groups list.
- DSE-11988 - Fixed a UI issue when creating a new session, where the user had to delete Untitled Session before typing in the session name.
- DSE-11945 - Fixed an issue where a poor network connection could cause an “Engine Exited” error message.
- DSE-11911 - Fixed an issue where updating the Project name caused navigation to other tabs to fail.
- DSE-11891 - Fixed an issue where attempting to open a file with Chinese characters in the name fails.
- DSE-10408 - Fixed an issue where quota checks may cause sessions or jobs to fail.

August 04, 2020

Release notes and fixed issues

New features

- Session Start UX - Sessions start page is now displayed with a fresh design.
- Engine Schedule - Improve the page loading performance for the “Lag” view of the Site Administration Activity tab.
- Subnet Selection - Subnets (one per workspace on Azure, multiple subnets per workspace on Amazon) can now be, optionally, manually selected when creating a new workspace.
- Model Metrics Development Workflow - You can develop and test model functions that track model metrics in a workbench session without actually deploying the model. This method makes it easy for you to develop and test code that tracks metrics and the code which consumes the tracked metrics without having to rebuild and redeploy a model to test every change.
- Default Model API Key Expiration - Administrators can now set a default expiration for Model API keys. If the user sets a longer expiration date, an error is returned. The default expiration duration is set at Admin Security Default Model API keys expiration in days .

Fixed issues

- DSE-11770 - Fixed an issue where the SHM size configuration was not applied for sessions started via `cdswctl`.
- DSE-11436 - Fixed an issue where SSH connections to CML terminal via the `cdswctl ssh-endpoint` command may time out after a few minutes.
- DSE-11321 - Fixed an issue where uploading files that have Chinese characters in the file names may fail.
- DSE-11089 - Fixed an issue where accessing Applications in the browser directly results in a 401 error and a blank page. Now users are redirected to a login page, and then directed back to the Application after successful authentication.
- DSE-10977 - Fixed an issue where the Remote Editing link on the Teams page may display a 500 error.
- DSE-10784 - Fixed an issue where Applications hosted on CML may be stopped after 7 days.
- DSE-10616 - Fixed an issue where the Jobs dependency pipeline only shows the first job.
- DSE-10031 - Fixed an issue where Windows line endings were converted automatically to Linux line endings in the Workbench. This behavior is now configurable.
- DSE-9986 - Fixed an issue so that users cannot create public projects via API when public project creation is restricted.
- DSE-8698 - Fixed redirections for CDSW links when TLS is enabled.
- DSE-6068 - Fixed an issue where sessions, jobs. experiments, and models may fail to be created if the engine tag includes an underscore character.

June 30, 2020

Release notes and fixed issues

New features

- CML Test Drive - CML now includes guided interactive modules for setting up workspaces, running data science workflows, and getting models to production. These workflows are designed with real-world business use case prototypes.

- Base Engine v12 - The default base engine is now version 12 (12-cml-2020.06-2).

  Key changes:

  - Python 3.6.10 (was 3.6.9)
  - R 3.6.3 (was 3.6.2)
  - CDPD 7.1.0 parts (Spark, Hadoop, Hive, Avro, Parquet)
  - Some bundled Python and R data science library upgrades (see Pre-installed Packages in Engines).

  Bug fixes and improvements:

  - DSE-10250 - Fixed a bug that could cause an job or experiment to exit with status 7 if used to run a very short script.
  - DSE-9872 - Fixed a bug that prevented users from using the `locate` command.
  - DSE-8601 - The Python launch_workers command now defaults the kernel argument match the kernel of the current session (previously it used "python3" in Python 2 sessions).
  - Reduced the engine image size by approximately 400 MB

Fixed issues

- DSE-11395 - Shared memory size configuration is now applied to sessions, jobs, and experiments.
- DSE-11433 - Fixed an issue where files opened in the Workbench editor are deselected when a new session starts.
- DSE-11421 - Fixed an issue where ML workspace provisioning can fail when a GPU instance that is not available in the AWS region is selected.

June 9, 2020

Release notes and fixed issues

New features

- Production ML - Production ML functionality is available to all accounts by default.
- Firefox support - CML now supports Firefox on newly provisioned CML workspaces.
- Monitoring - Grafana dashboard now shows the user ID to help administrators monitor usage and resources.
- Session UI - Added a new pop-up window to the session start UI.

Fixed issues

- DSE-10985 - Fixed a bug where non-admin users were not able to start or list experiments.
- DSE-10893 - Fixed a bug where upgrading a monitoring-enabled CML workspace may flush the imported Grafana dashboards.
- DSE-10671 - Fixed a link on the Team page to go to Team settings.
- DSE-10618 - Fixed a bug where Engine Profile is not displayed on the Jobs page.
- DSE-10407 - Fixed a bug where selecting the Python2 kernel causes problems when creating an experiment or job.
- DSE-10405 - Fixed a bug where forking a CML project caused the project to be copied repeatedly until interrupted.
- DSE-10185 - Fixed a bug where CML Applications can get stuck in the “Starting” state due to user-specified subdomains that are not handled by CML correctly.
- DSE-10054 - (SAML 2.0 SSO) Fixed a bug to prevent arbitrary redirection of browsers for SSO to third-party applications.

May 5, 2020

Release notes and fixed issues

- General Availability of Production ML features - including:
  - Model Metrics and Monitoring
  - Model Technical Metrics
  - Model Governance
  - CLI for Models
  - Model API Key for securing Models
- CML on Azure Updates
  - Custom GPUs are now supported.
  - Availability Zones are no longer required to provision an ML Workspace.
- Enable Monitoring - Monitoring can be enabled when provisioning ML workspaces by CLI.

Fixed issues

- DSE-10631 - Fixed Default Quota initialization issue.
- DSE-10441 - Fixed a bug where NFS input field is not marked as a required field on the Provision Workspace page.
- DSE-10399 - Fixed a UI bug on the CML Models pages on Firefox.
- DSE-8836 - Fixed a bug where Jobs & Applications usage are not metered correctly.
- DSE-8779 - Fixed tooltip typo on UI which mixed up CML and CDSW.
- DSE-7944 - Upgraded Kubernetes to defend against CVE 2019-11253.

April 14, 2020

Release notes and fixed issues

- Command line arguments - The Create a Job UI now has an Arguments field where you can enter command line arguments for the executed script. This feature is available for R and Python scripts.
- User information required - Administrators can now require users to enter information when starting sessions, for audit purposes.
- UI update - In Site Administration, the Activity tab is now renamed to Usage.
- Accessibility - Improvements for W3C WCAG 2-AA Compliance.
- Grafana dashboards - A Grafana dashboard is available in Models.
- Cross-Origin Resource Sharing (CORS) - CORS is now disabled by default. Enable CORS if you have web applications on different domains that need cross-domain communication with the Cloudera Machine Learning API.

Fixed Issues

- DSE-9063 - Fixed an issue where a custom quota takes effect even if the quota feature is turned off.
- DSE-9469 - Fixed an issue where email attachments greater than 4 MB caused problems.
- DSE-6375 - Fixed an issue where creating a private project from a Git repo fails, even with an authorized team SSH key.
- DSE-9041 - Improved the SMTP settings for test email.
- DSE-9694 - Fixed an issue where the order of columns in the Workspace instances table was inconsistent.
- DSE-8601 - The kernel argument launch_workers should default to kernel of current session.
- DSE-9810 - Fixed an issue where users could not download files that have non-ASCII characters in the file name.

March 16, 2020

Release notes and fixed issues

- Cloudera Machine Learning on Azure - Cloudera Machine Learning workspaces can now be created in environments hosted in Azure. See the documentation for information on Azure network planning, limitations, and setup.

- Base Engine v11 - This release adds Hive Warehouse Connector (HWC) to the base engine (11-cml1.4-hwc-v1). Additionally, the base engine now has the ability to connect to DWX. No changes are made to the included library versions.
- Navigation improvement - Cloudera Machine Learning users can click the nine-dot icon to navigate to the Cloudera Manager console.
- Changing user names and addresses - In Cloudera Machine Learning instances where user authentication is performed through single sign-on (SSO), users cannot change their names and addresses in CML workspaces.
- Jupyter enabled by default - Fixed an issue where Jupyter was not enabled by default in the default base engine.

Fixed Issues

- DSE-8791 - Fixed an issue where Job or Model containers might not start or restart correctly.
- DSE-9264 - Fixed an issue where the Regenerate Access Key button on the Model Settings page does not work for non-admin users.

February 13, 2020

Release notes and fixed issues

- Kerberos Authentication Improvements - Previously, users needed to manually authenticate themselves by entering their CDP FreeIPA credentials into their CML workspaces (on the Account settings Hadoop Authentication page). This is no longer required.

  Users will now automatically receive the Kerberos credentials required for any CML workloads such as sessions, jobs, models, etc. Existing workspaces can be upgraded to take advantage of this improvement.

  Additionally, the Hadoop Authentication tab has been removed from the workspace UI.
- Workspace Monitoring Enabled - When you provision a Machine Learning workspace, Monitoring is enabled by default under Advanced Options.

January 30, 2020

Release notes and fixed issues

- Base Engine v11 - The default base engine is now version 11. The only change to the included libraries is the R interpreter, which is updated to version 3.6.2.
- Python 2 checkbox disabled - Python 2 sessions are now disabled by default on new clusters, but can be re-enabled by admin users.
- Load Balancer Source Ranges - When provisioning a Machine Learning workspace, in Advanced Settings, you can enter the CIDR range of IP addresses allowed to connect to the workspace. You must include the entire IP pool in the allowlist for your VPC to ensure that terminal sessions can connect to the workspace.
- Updated Open Workbench button name - When launching a session, the Open Workbench button has been renamed to New Session.
- Login error message - Fixed a bug where users might see the following error message upon login to a CML workspace: "Email already associated with an account."

December 19, 2019

Release notes and fixed issues

- Monitoring Workspaces with Grafana - CML now leverages Prometheus and Grafana to provide a dashboard that allows you to monitor how CPU, memory, storage, and other resources are being consumed by ML workspaces.
- Custom Quotas - CML workspace site administrators can now enable custom quotas to set resource usage per user.

- Tags - There are three new AWS resource tracking tags:
  - Cloudera-Resource-Name: <The CRN of the associated CML Workspace for which the resource was provisioned.>
  - Cloudera-Environment-Resource-Name: <The CRN of the Environment in which the resource was created.>
  - Cloudera-Creator-Resource-Name: <The CRN of the CDP user who requested creation of the resource.>

  AWS resource tags are set by default. They can be searched and viewed through the AWS console or CLI. These tags are helpful for tracking resource usage and cost.
- Granting remote access - The procedure to grant and revoke remote access to ML workspaces is improved. You can easily add new users. You can also see which users currently have access, and then quickly revoke access to specific users.
- CML instance type cost reduced - The node used to run the CML application was downsized to a more economical AWS instance type. The instance type was changed from m5.12xlarge to m5.4xlarge, which should result in a noticeable reduction in cloud costs.
- Base Engine v10-cml1.3 - The default engine is now v10. See the package listing for the updated versions of included libraries.
  - Python 3 is version 3.6.9 (was 3.6.8 in Engine v8).
  - Python 2 is version 2.7.17 (was 2.7.11 in Engine v8).

November 1, 2019

Release notes and fixed issues

- Analytical Applications - CML now gives data scientists a way to create long-running standalone ML web applications/dashboards that can easily be shared with other business stakeholders.
- Quotas - CML workspace site administrators can now enable CPU, GPU, and memory usage quotas per user. Quotas must be enabled separately for each workspace.

  Note: The Quotas feature is in Technical Preview.
- Diagnostic Bundles - CML now allows site administrators to download diagnostic bundles from the Site Admin panel.
- UI Improvements and Changes
  - You can now display the Details page by clicking on the Workspace Name in the ML Workspaces page.
  - The ML Workspace Details page now contains a Events tab.

    The Events tab displays high-level events for your workspace. You can click View Logs to display additional log information about the action.
  - The ML Workspace Details page now displays AWS workspace tags.

    The tag information displays both default tags and any tags you have specified. You can specify these workspace tags in the provision workspace page, under the Advanced options. The default workspace tags include:
    - Creator
    - Environment
    - Owner
    - WorkspaceName

September 23, 2019

Release notes and fixed issues

- Single Sign-on (SSO) Changes - You no longer need to create separate authorization groups to grant users SSO access to workspaces. Authorization groups are now managed per-environment using the MLAdmin and MLUser resource roles.

- Resource Tags - You can now add resource tags to all the cloud infrastructure, compute, and storage resources used by an ML workspace you provision.
- Remove Workspaces - New options added that allow you to retain project storage (in EFS) and force delete workspaces from CDP.
- View Workspace Details - Each workspace now has an associated details page where you can access links to the workspace itself, the environment where the workspace was created, and the underlying Kubernetes cluster (links to cloud service provider). A link to this page is available under the Actions menu.
- Search and Filter workspaces - New filter that allows you to display workspaces in a specific environment. You can also search for a workspace by name.

August 22, 2019

Release notes and fixed issues

This marks the General Availability (GA) release of Cloudera Machine Learning.

Compatibility for Cloudera AI and Runtime components

Learn about Cloudera AI and compatibility for Runtime components across different versions.

Table 1: Cloudera AI compatibility with Runtime component details

Cloudera AI	Data Lake	Supported Spark Version
2.0.46-b302	7.2.16 or higher	• Spark 2.4.8 • Spark 3.2.3 • Spark 3.3.0 • Spark 3.5.1
2.0.46-b238	7.2.16 or higher	• Spark 2.4.8 • Spark 3.2.3 • Spark 3.3.0 • Spark 3.5.1
2.0.46-b210	7.2.16 or higher	• Spark 2.4.8 • Spark 3.2.3 • Spark 3.3.0
2.0.46-b200	7.2.16 or higher	• Spark 2.4.8 • Spark 3.2.3 • Spark 3.3.0
2.0.45-b86	7.2.16 or higher	• Spark 2.4.8 • Spark 3.2.3 • Spark 3.3.0
2.0.45-b82	7.2.16 or higher	• Spark 2.4.8 • Spark 3.2.3 • Spark 3.3.0
2.0.45-b81	7.2.16 or higher	• Spark 2.4.8 • Spark 3.2.3 • Spark 3.3.0
2.0.45-b76	7.2.16 or higher	• Spark 2.4.8 • Spark 3.2.3 • Spark 3.3.0

Cloudera AI	Data Lake	Supported Spark Version
2.0.45-b54	7.2.16 or higher	• Spark 2.4.8 • Spark 3.2.3 • Spark 3.3.0
2.0.43-b233	7.2.16 or higher	
2.0.43-b229	7.2.16 or higher	
2.0.43-b220	7.2.16 or higher	
2.0.43-b208	7.2.16 or higher	
2.0.41-b225	7.2.15 or higher	
2.0.40-b157	7.2.14 or higher	
2.0.40-b150	7.2.14 or higher	
2.0.38-b126	7.2.14 or higher	
2.0.38-b125	7.2.14 or higher	
2.0.38-b121	7.2.14 or higher	
2.0.38-b101	7.2.14 or higher	
2.0.36-b121	7.2.14 or higher	
2.0.36-b118	7.2.14 or higher	
2.0.34-b116	7.2.14 or higher	
2.0.32-b123	7.2.14 or higher	
2.0.30-b114	7.2.14 or higher	
2.0.29-b61	7.2.11 or higher	
2.0.27-b64	7.2.11 or higher	
2.0.26-b180	7.2.11 or higher	
2.0.25-b110	7.2.11 or higher	
2.0.24-b100	7.2.11 or higher	

Upgraded Cloudera AI deployments keep multiple Hadoop CLI addon versions that administrators can configure to maintain compatibility between Cloudera AI and the Data Lake.

Spark support in Cloudera AI

The end of support (EoS) policy for Spark is the same in both Cloudera Data Engineering and Cloudera AI. For information about end of support for Spark, see *Cloudera Data Engineering Runtime end of support* .

Related Information

Cloudera Data Engineering Runtime end of support

CVE-2021-44228 Remediation for Cloudera AI Data Service

The procedure to remediate CVE-2021-44228 on Cloudera AI Data Service is described in this document.

On December 15 2021, the ML team released version 2.0.25-b110 of Cloudera AI Data Service for Cloudera Public Cloud. It addresses CVE-2021-44228 which affects Apache Log4j2 versions 2.0 through 2.14.1. We urge all customers to upgrade their workbenches to the latest version.

The Cloudera AI service code itself is not written in Java and hence is not vulnerable. However, the `log4j2` jar file does exist in the (now deprecated) engine as well as the Hadoop CLI and Spark ML Runtimes addons. As a result, the `log4j2` jar file is available in sessions, jobs, models, and applications. Because of this, there is no direct threat, but a data scientist could inadvertently use this `log4j2` jar file and expose themselves to this vulnerability.

As a result, we are releasing a new version of the engine and ML Runtimes addons that remove this vulnerability.

Upgrade Cloudera AI Workbench to the new version

Administrators can now upgrade the Cloudera AI Workbenches to version 2.0.25-b110. To upgrade the workbench, select the Actions icon and select Upgrade Workbench.

Once the workbench is upgraded, you can follow the steps below to ensure the appropriate engine and ML Runtimes addons are used.

ML Runtimes Add-ons

The first step involves the use of runtime addons. These addons consist of Hadoop CLI code and Spark code that is added to ML Runtimes. An administrator must go to the Site Administration -> Runtime/Engine page and ensure that in the “Hadoop CLI Version” drop down box, the chosen selection ends with “HOTFIX-1”.

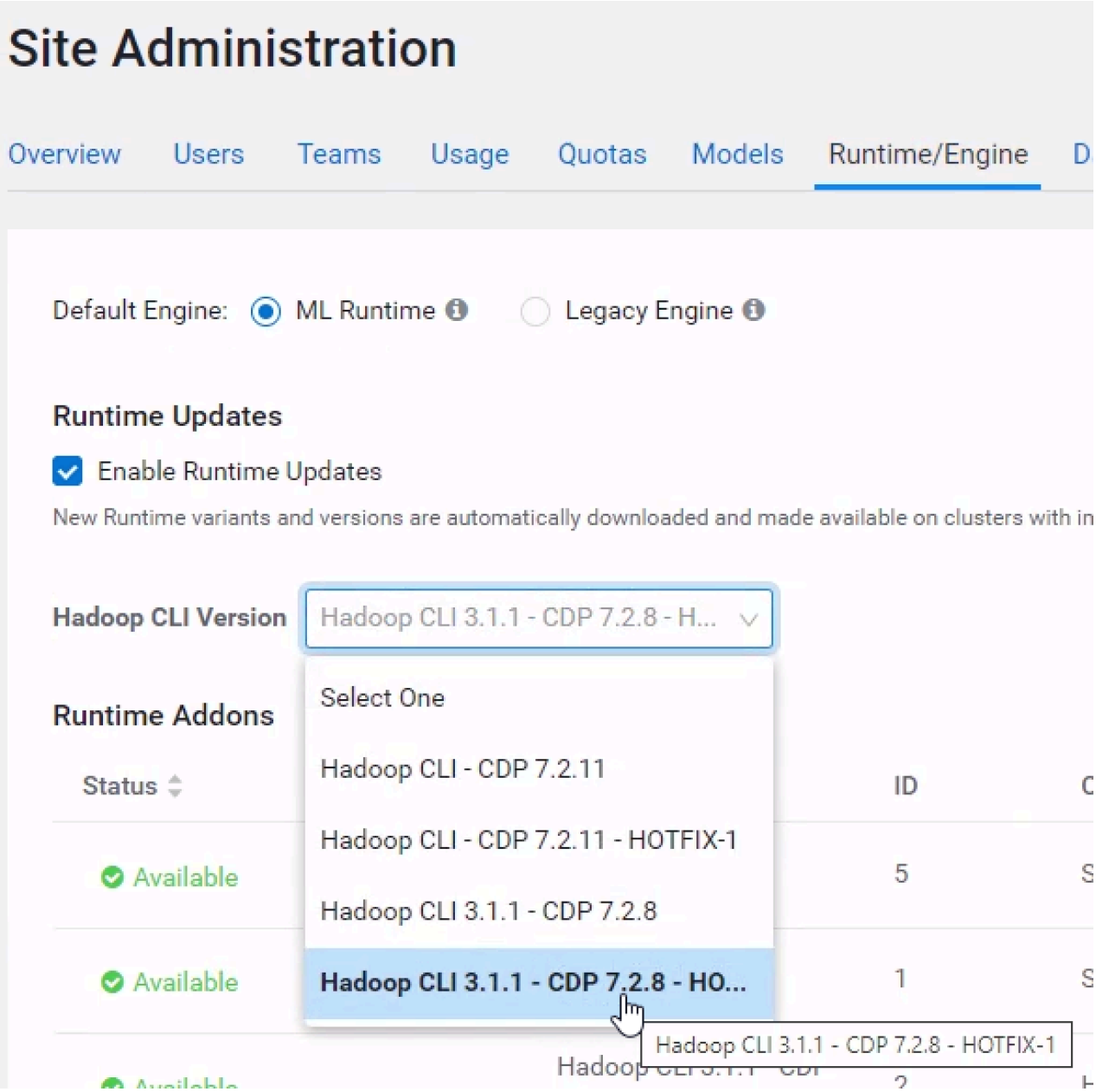

When starting a session, if “Enable Spark” is turned on, users must use a version of Spark ending with “HOTFIX-1”.

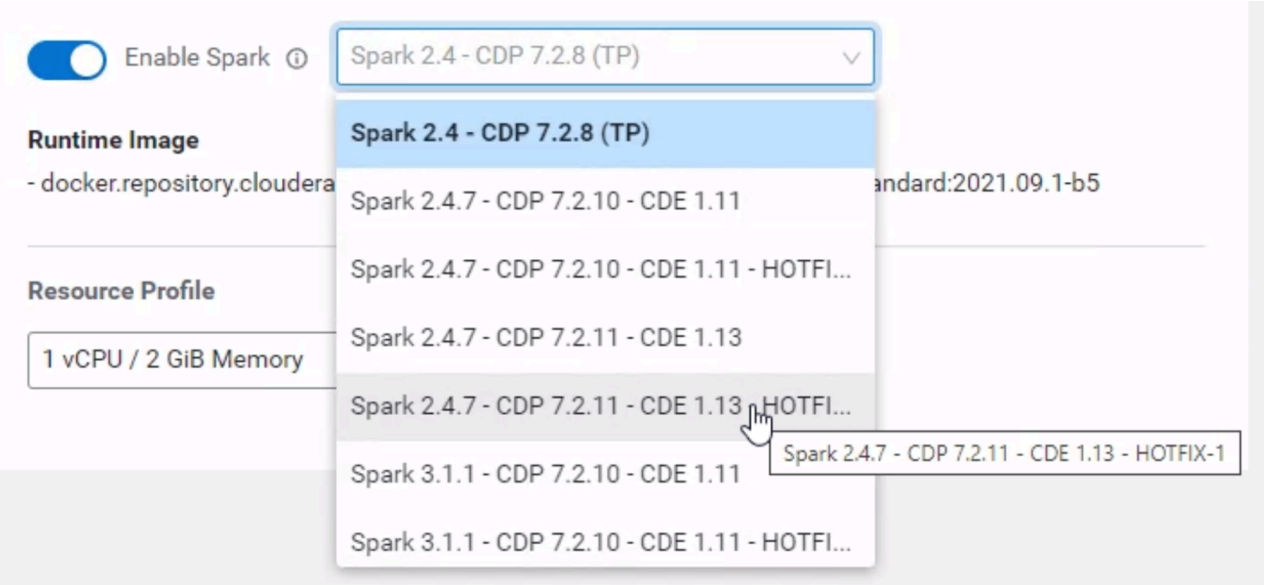

Jobs and applications that use Spark in projects using runtimes, ensure the job or application uses a correct version of the Spark addon by going to the job/application Settings page and selecting a version of Spark that ends with “HOTFIX-1”. Your applications will require a restart.

Models and Experiments that use Spark runtimes must be rebuilt and redeployed with a fixed version of Spark. The models and experiments do not need to be deleted.

Engines

For customers using projects with (deprecated) engines we have released a new engine version which fixes the vulnerability:

```
docker.repository.cloudera.com/cloudera/cdsw/engine:15-cml-2021.09-2
```

The “-2” at the end is important. It is identical to the engine ending in “-1”, except the vulnerability has been removed. This engine is now the default for new workbenches. To ensure this engine is used:

- An administrator should go to Site Administration Runtime/Engine and scroll down to Engine Images. Ensure the above version is selected as the default.
- For all projects using engines, go to Project Settings Runtime/Engine and verify the engine is set to the version above.
- For all applications and jobs in projects that use engines, go to the application or job settings page and ensure that under Select Job Engine the above version is selected.
- Deploy a new build for all models that use engines.

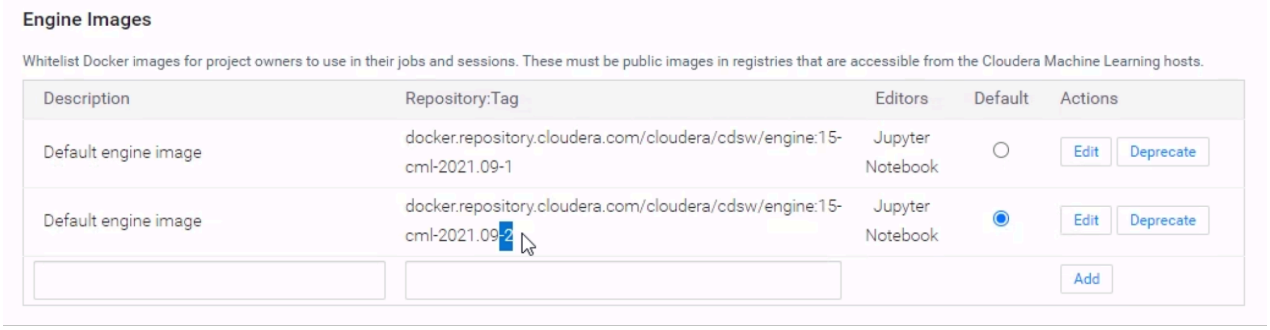

Technical Details of the Fix

The `log4j2` jar file exists in several places and is also packaged inside other jar files. Instead of upgrading the `log4j2` jar file, we have chosen to remove the vulnerable java class file. This is one of the mitigations proposed in the CVE text:

"it can be mitigated ... by removing the JndiLookup class from the classpath (example: zip -q -d log4j-core-*.jar */JndiLookup.class)."

As a result, older versions of the `log4j2` jar file still exist in sessions after the fix, but the offending class file has been removed. To verify this, you can run:

```
grep JndiLookup.class <jar file>
```

Known Issues and Limitations

There are some known issues you might run into while using Cloudera AI.

Unable to deploy ONNX optimization profiles for embedding and ranking NIMs on GPUs with optimization profiles (DSE-40509)

Deploying ONNX profiles for embedding and ranking NIMs on GPUs where compatible GPU profiles exist will lead to deployment failure. Before deploying an ONNX optimization profile for embedding or ranking NIMs from the Model Hub, ensure that the NIM does not have a supported profile for the target GPU.

Cloudera AI does not support non-transparent proxy with authentication (DSE-36512)

Cloudera AI on public cloud does not support non-transparent proxy with authentication. While configuring proxy using the Cloudera Data Platform console, do not specify your username and password.

Unable to paginate in Cloudera AI Workbench backup table (DSE-41406)

When selecting a backup in the Workbench backup table, pagination to the next page does not work as expected. This is due to an issue in the cuix library.

Workaround: To avoid the issue, do not try to navigate to the next page after selecting a backup. Instead, refresh the page to view details of other Workbench backups.

Too many Federated Identity Credentials have been assigned to the Managed Identity. error is displayed in the Event Log file (DSE-41063)

Each Azure managed identity, that is, Logger Identity in a Cloudera Data Platform environment can have a maximum of 20 federated identity credentials. Because each cluster requires 1–2 federated identities, there is a maximum number of clusters that can be created per environment. If you exceed the maximum number of clusters, Too many Federated Identity Credentials have been assigned to the Managed Identity." error is displayed.

Workaround: Perform the following:

1. Log into the Microsoft Azure portal.
2. Go to Managed Identity and select the logger identity you defined of your resource group.
3. Click Setting > Federated Credentials .
4. In the Federated Credentials page, delete the unused federated identity credentials (with no AKS cluster associated with it).

Cloudera AI Inference service upgrade is temporarily unavailable (DSE-41060)

Due to a known issue, Cloudera AI Inference service upgrade is disabled temporarily. When you try to upgrade the Cloudera AI Inference service using CDP CLI, an error is displayed.

Cloudera AI automatic JWT authorization to Cloudera Data Warehouse is failing due to a wrong KNOX URL (DSE-39855)

Due to an issue, there is a mismatch of the Data Lake name between the actual Data Lake name in the environment and the one parsed by the CML 2.0.43-b208 version or later.

Workaround:

1. Obtain the correct Data Lake version by running the following command using CDP CLI:

```
cdp datalake describe-datalake
```

2. Override the KNOX URL in the environment variable by performing the following:

   a. Run the following command to save the deployment status to a file:

```
kubectl get deployment ds-cdh-client -o json -n mlx > /tmp/rs.json
```

   b. Edit the /tmp/rs.json file and add the below object for ds-cdh-client environment under the spec.template.spec.containers.env section.

```
{
     "name": "FIXED_KNOX_URL",
     "value": "https://[***ENVIRONMENT-VARIABLE***]/value"
}
```

   c. Apply the configuration.

```
kubectl apply -f /tmp/rs.json
```

.

The `modify-ml-serving-app` command fails on Cloudera AI Inference service Azure cluster in the us-west-2 workload region (DSE-39826)

When you run `modify-ml-serving-app` API on Cloudera AI Inference service Azure cluster, the status of the application is displayed as modify:failed.

Workaround: You must first delete the instance group you want to modify using the `delete-instance-group-ml-serving-app` API. Then, recreate the instance group, modify the configuration based on your requirements, and add the instance group using the `add-instance-groups-ml-serving-app` API.

Too many Federated Identity Credentials have been assigned to the Managed Identity. error is displayed in the Event Log file (DSE-41063)

Each Azure managed identity, that is, Logger Identity in a Cloudera Data Platform environment can have a maximum of 20 federated identity credentials. Because each cluster requires 1–2 federated identities, there is a maximum number of clusters that can be created per environment. If you exceed the maximum number of clusters, Too many Federated Identity Credentials have been assigned to the Managed Identity." error is displayed.

Workaround: Perform the following:

1. Log into the Microsoft Azure portal.
2. Go to Managed Identity and select the logger identity you defined of your resource group.
3. Click Setting > Federated Credentials .
4. In the Federated Credentials page, delete the unused federated identity credentials (with no AKS cluster associated with it).

Automatic synchronization of teams and users feature is disabled in the UI (DSE-36718)

Due to some issues with the automatic synchronization of teams and users, this feature is disabled in the UI in the September 26, 2024 release for version 2.0.46-b200. New installation or upgraded workbench does not have the auto synchronization option in the UI.

Workaround: You can manually synchronize teams and synchronize users.

Web pod crashes if a project forking takes more than 60 minutes (DSE-35251)

The web pod crashes if a project forking takes more than 60 minutes. This is because the timeout is set to 60 minutes using the `grpc_git_clone_timeout_minutes` property. The following error is displayed after the web pod crash:

```
2024-04-23 22:52:36.384   1737    ERROR      AppServer.VFS.grpc
       crossCopy grpc error    data = [{"error":"1"},{"code":4,"details":"2"
,"metadata":"3"},"Deadline exceeded",{}]
["Error: 4 DEADLINE_EXCEEDED: Deadline exceeded\n    at callErrorFromStatus
 (/home/cdswint/services/web/node_modules/@grpc/grpc-js/build/src/call.js:31
:19)\n    at Object.onReceiveStatus (/home/cdswint/services/web/node_modules
/@grpc/grpc-js/build/src/client.js:192:76)\n    at Object.onReceiveStatus (/
home/cdswint/services/web/node_modules/@grpc/grpc-js/build/src/client-interc
eptors.js:360:141)\n    at Object.onReceiveStatus (/home/cdswint/services/we
b/node_modules/@grpc/grpc-js/build/src/client-interceptors.js:323:181)\n
at /home/cdswint/services/web/node_modules/@grpc/grpc-js/build/src/resolving
-call.js:94:78\n    at process.processTicksAndRejections (node:internal/proc
ess/task_queues:77:11)\nfor call at\n    at ServiceClientImpl.makeUnaryReque
st (/home/cdswint/services/web/node_modules/@grpc/grpc-js/build/src/client.j
s:160:34)\n    at ServiceClientImpl.crossCopy (/home/cdswint/services/web/no
de_modules/@grpc/grpc-js/build/src/make-client.js:105:19)\n    at /home/cdsw
int/services/web/server-dist/grpc/vfs-client.js:235:19\n    at new Promise (
<anonymous>)\n    at Object.crossCopy (/home/cdswint/services/web/server-dis
t/grpc/vfs-client.js:234:12)\n    at Object.crossCopy (/home/cdswint/service
s/web/server-dist/models/vfs.js:280:38)\n    at projectForkAsyncWrapper (/ho
me/cdswint/services/web/server-dist/models/projects/projects-create.js:229:1
9)"]
node:internal/process/promises:288
            triggerUncaughtException(err, true /* fromPromise */);
            ^Error: 4 DEADLINE_EXCEEDED: Deadline exceeded
    at callErrorFromStatus (/home/cdswint/services/web/node_modules/@grpc/
grpc-js/build/src/call.js:31:19)
    at Object.onReceiveStatus (/home/cdswint/services/web/node_modules/@grp
c/grpc-js/build/src/client.js:192:76)
    at Object.onReceiveStatus (/home/cdswint/services/web/node_modules/@gr
pc/grpc-js/build/src/client-interceptors.js:360:141)
    at Object.onReceiveStatus (/home/cdswint/services/web/node_modules/@grp
c/grpc-js/build/src/client-interceptors.js:323:181)
    at /home/cdswint/services/web/node_modules/@grpc/grpc-js/build/src/resol
ving-call.js:94:78
    at process.processTicksAndRejections (node:internal/process/task_queu
es:77:11)
for call at
    at ServiceClientImpl.makeUnaryRequest (/home/cdswint/services/web/node
_modules/@grpc/grpc-js/build/src/client.js:160:34)
    at ServiceClientImpl.crossCopy (/home/cdswint/services/web/node_modules/
@grpc/grpc-js/build/src/make-client.js:105:19)
    at /home/cdswint/services/web/server-dist/grpc/vfs-client.js:235:19
    at new Promise (<anonymous>)
    at Object.crossCopy (/home/cdswint/services/web/server-dist/grpc/vfs-
client.js:234:12)
    at Object.crossCopy (/home/cdswint/services/web/server-dist/models/vfs
.js:280:38)
```

```
    at projectForkAsyncWrapper (/home/cdswint/services/web/server-dist/model
s/projects/projects-create.js:229:19) {
  code: 4,
  details: 'Deadline exceeded',
  metadata: Metadata { internalRepr: Map(0) {}, options: {} }
}
```

Workaround: Increase the timeout limit using the `grpc_git_clone_timeout_minutes` property. For example, 120 minutes.

```
UPDATE site_config SET grpc_git_clone_timeout_minutes = <NEW VALUE>;
```

Enabling Service Accounts (DSE-32943)

Teams in the Cloudera AI Workbench can only run workloads within team projects with the Run as option for service accounts if they have previously manually added service accounts as a collaborator to the team.

Working with files larger than 1 MB in Jupyter causes error (OPSAPS-61524)

While working on files or saving files of size larger than 1 MB, Jupyter Notebook may display an error message such as 413 Request Entity Too Large.

Workaround:

Clean up the notebook cell results often to keep the notebook below 1 MB. Use the kubectl CLI to add the following annotation to the ingress corresponding to the session.

```
annotations:
        nginx.ingress.kubernetes.io/proxy-body-size: "0"
```

1. Get the session ID (the alphanumeric suffix in the URL) from the web UI.
2. Get the corresponding namespace:

   ```
   kubectl get pods -A | grep <session ID>
   ```

3. List the ingress in the namespace

   ```
   kubectl get ingress -n <user-namespace> | grep <session ID>
   ```

4. In the metadata, add the annotation.

   ```
   kubectl edit ingress <ingress corresponding to the session> -n <user-nam
   espace>
   ```

Terminal does not stop after time-out (DSE-12064)

After a web session times out, the terminal should stop running, but it remaings functional.

Cloudera AI Workbench upgrades disabled with NTP

Upgrades are disabled for Cloudera AI Workbench configured with non-transparent proxy (NTP). This issue is anticipated to be fixed in a subsequent hotfix release.

Using dollar character in environment variables in Cloudera AI

Environment variables with the dollar ($) character are not parsed correctly by Cloudera AI. For example, if you set PASSWORD="pass$123" in the project environment variables, and then try to read it using the echo command, the following output will be displayed: pass23

Workaround: Use one of the following commands to print the $ sign:

```
echo 24 | xxd -r -p
or
echo JAo= | base64 -d
```

Insert the value of the environment variable by wrapping it in the command substitution using $() or ``. For example, if you want to set the environment variable to ABC$123, specify:

```
ABC$(echo 24 | xxd -r -p)123
or
ABC`echo 24 | xxd -r -p`123
```

Models: Some API endpoints not fully supported

In the create_run api, run_name is not supported.

Also, search_experiments only supports pagination.

When a team added as a collaborator, it does not appear in the UI. (DSE-31570)

Run Job as displays even if the job is enabled on a service account. (DSE-31573)

If the job is enabled on a service account, the Run Job as option should not display. Even if me is selected at this point, the job still runs in the service account.

AMP archive upload fails if Project does not contain metadata YAML file

To create a zip file to deploy AMPs in Cloudera AI, do the following:

1. Download the AMP zip file from GitHub
2. Unzip it to a temp directory
3. From the command line navigate to the root directory of the zip
4. Run this command to create the new zip file: zip -r amp.zip .

Make sure you see the .project-metadata/yaml in the root of the zip file.

Cloning from Git using SSH is not supported via HTTP proxy

Workaround: Cloudera AI Projects support HTTPS for cloning git projects. It is suggested to use this as the workaround.

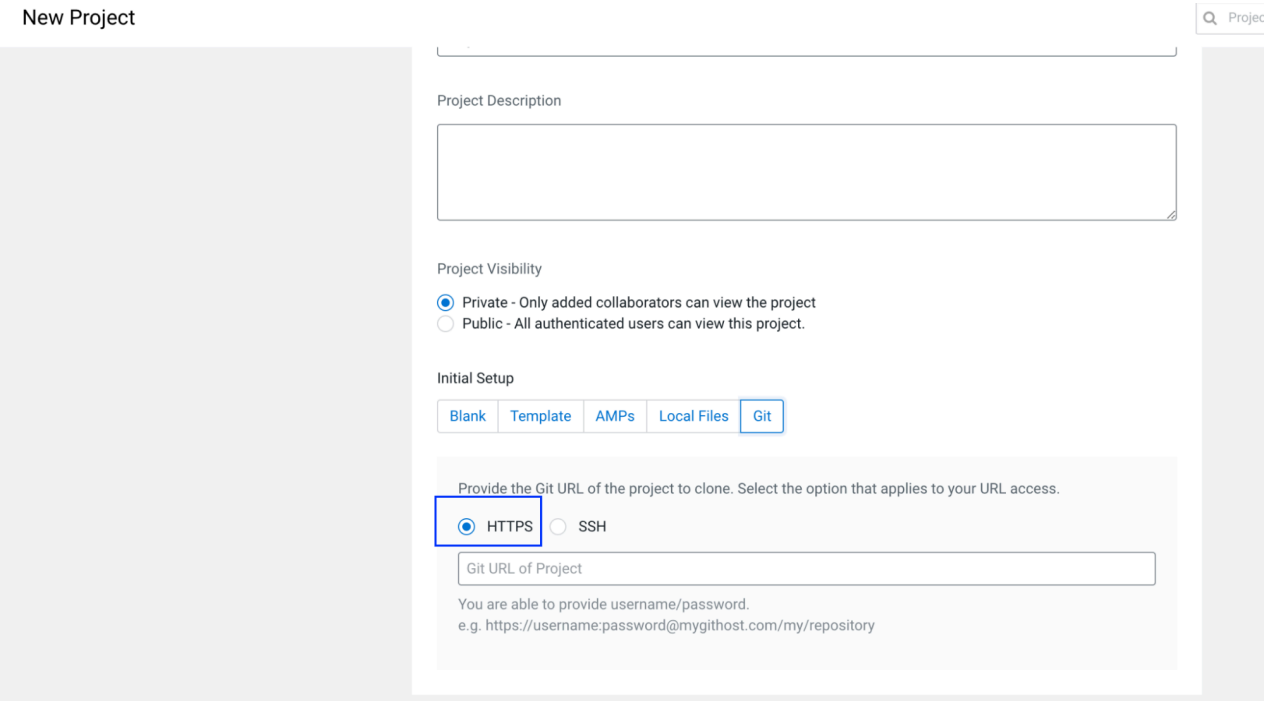

Model deployments requiring outbound access via proxy do not honor HTTP_PROXY, HTTPS_PROXY environment variables

Workaround: Add the `HTTP_PROXY`, `HTTPS_PROXY`, `http_proxy` and `https_proxy` environment variables to the cdsw-build.sh file of the Project Repository.

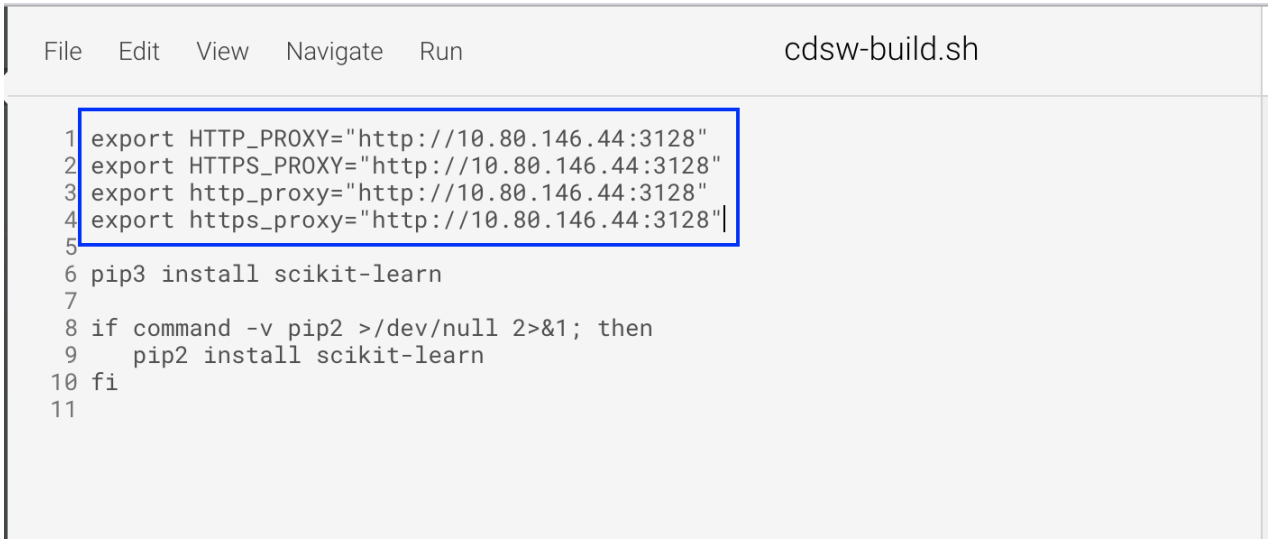

Application does not restart after upgrade or migration

An application may fail to automatically restart after a workbench upgrade or migration. In this case, manually restart the application.

Do not use backtick characters in environment variable names

Avoid using backtick characters (`) in environment variable names, as this will cause sessions to fail with exit code 2.

Cloudera AI Registry is not supported on R models

Cloudera AI Registry is not supported on R models.

The mlflow.log_model registered model files might not be available on NFS Server (DSE-27709)

When using mlflow.log_model, registered model files might not be available on the NFS server due to NFS server settings or network connections. This could cause the model to remain in the registering status.

Workaround:

- Re-register the model. It will register as an additional version, but it should correct the problem.
- Add the ARTIFACT_SYNC_PERIOD environment variable to hdfscli-server Kubernetes deployment and set it to an integer value. This will set the model registry retry operation to twice the number of seconds specified by the artifact sync period integer value. If the ARTIFACT_SYNC_PERIOD is set to 30 seconds then model registry will retry for 60 seconds. The default value is 10 and model registry retries for 20 seconds. For example: -name: ARTIFACT_SYNC_PERIOD value: “30”.

Applications appear in failed state after upgrade (DSE-23330)

After upgrading Cloudera AI from version 1.29.0 on AWS, some applications may be in a Failed state. The workaround is to restart the application.

Cannot use hashtag character in JDBC connection string

The special character # (hashtag) cannot be used in a password that is then used in a JDBC connection string. Avoid using this special character, or use '%23' instead.

Cloudera AI Workbench installation fails

Cloudera AI Workbench installation with Azure NetApp Files on NFS v4.1 fails. The workaround is to use NFS v3.

Spark executors fail due to insufficient disk space

Generally, the administrator should estimate the shuffle data set size before provisioning the workbench, and then specify the root volume size of the compute node that is appropriate given that estimate. For more specific guidelines, see the following resources.

- How do I avoid the "No space left on device" error where my disk is running out of space?
- How can I prevent a Hadoop or Spark job's user cache from consuming too much disk space in Amazon EMR?

Runtime Addon fails to load (DSE-16200)

A Spark runtime add-on may fail when upgrading a workbench.

Solution: To resolve this problem, try to reload the add-on. In Site Administration Runtime/Engine , in the option menu next to the failed add-on, select Reload.

Cloudera AI Workbench provisioning times out

When provisioning a Cloudera AI Workbench, the process may time out with an error similar to Warning FailedMo unt or Failed to sync secret cache:timed out waiting for the condition. This can happen on AWS or Azure.

Solution: Delete the workbench and retry provisioning.

Cloudera AI endpoint connectivity from Cloudera Data Hub and Cloudera Data Engineering (DSE-14882)

When Cloudera Data Platform services connect to Cloudera AI services, if the Cloudera AI Workbench is provisioned on a public subnet, traffic is routed out of the VPC first, and then routed back in. On Cloudera Private Cloud Cloudera AI, traffic is not routed externally.

NFS performance issues on AWS EFS (DSE-12404)

Cloudera AI uses NFS as the filesystem for storing application and user data. NFS performance may be much slower than expected in situations where a data scientist writes a very large number (typically in the thousands) of small files. Example tasks include: using `git clone` to clone a very large source repository (such as TensorFlow), or using `pip` to install a Python package that includes JavaScript code (such as `plotly`). Reduced performance is particularly common with Cloudera AI on AWS (which uses EFS), but it may be seen in other environments.

Disable file upload and download (DSE-12065)

You cannot disable file upload and download when using the Jupyter Notebook.

Remove Workbench operation fails (DSE-8834)

Remove Workbench operation fails if workbench creation is still in progress.

API does not enforce a maximum number of nodes for Cloudera AI Workbench

Problem: When the API is used to provision new Cloudera AI Workbench, it does not enforce an upper limit on the autoscale range.

Downscaling Cloudera AI Workbench nodes does not work as expected (MLX-637, MLX-638)

Problem: Downscaling nodes does not work as seamlessly as expected due to a lack of Bin Packing on the Spark default scheduler, and because dynamic allocation is not currently enabled. As a result, currently infrastructure pods, Spark driver/executor pods, and session pods are tagged as non-evictable using the cluster-autoscaler.kubernetes.io/safe-to-evict: "false" annotation.

Limitations

Cloudera AI Inference service and Cloudera AI Registry are not supported on Azure East Asia and Qatar Central regions

Cloudera AI Inference service and Cloudera AI Registry are not supported on Microsoft Azure East Asia and Qatar Central regions due to lack of support for Workload Identity by Microsoft Azure.

Technical Service Bulletins

TSB 2024-761: Orphan EBS Volumes in Cloudera AI Workbench

Cloudera AI provisions Elastic Block Store (EBS) volumes during provisioning of a workbench. Due to missing labels on Cloudera AI Workbench, delete operations on previously restored Cloudera AI Workbench didn't clean up a subset of the provisioned block volumes.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2024-761: Orphan EBS Volumes in Cloudera AI Workbench

TSB 2023-628: Sensitive user data getting collected in Cloudera AI Workbench or CDSW workbench diagnostic bundles

When using Cloudera Data Science Workbench (CDSW), Cloudera recommends users to store sensitive information, such as passwords or access keys, in environment variables rather than in the code. See Engine Environment Variables in the official Cloudera documentation for details. Cloudera recently learned that all session environment variables in the affected releases of CDSW and Cloudera AI are logged in web pod logs, which may be included in support diagnostic bundles sent to Cloudera as part of support tickets.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB-2023-628: Sensitive user data getting collected in Cloudera AI Workbench or in CDSW workbench diagnostic bundles

TSB 2022-588: Kubeconfig and new version of aws-iam-authenticator

Regenerate Kubeconfig and in conjunction use a newer version of aws-iam-authenticator on AWS. Kubeconfig in Cloudera Data Platform Cloudera Public Cloud Data Services needs to be regenerated because the Kubeconfig generated before June 15, 2022 uses an old APIVersion (client.authentication.k8s.io/v1alpha1) which is no longer supported. This causes compatibility issues with aws-iam-authenticator starting from v0.5.7. To be able to use the new aws-iam-authenticator, the Kubeconfig needs to be regenerated.

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB-2022-588: Kubeconfig and new version of aws-iam-authenticator

Related Information

AWS Limitations

Azure Limitations

Cloudera AI

AWS Requirements

Date published: 2020-07-16
Date modified: 2025-01-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

AWS Account Prerequisites for Cloudera AI Workbenches..... 4

Limitations on AWS..... 6

Network Planning for Cloudera AI on AWS..... 7

AWS IAM restricted roles and policies for compute and Cloudera AI.....7

Create IAM roles and instance profile pair..... 7

Create role and policy used to deploy Cloudera Data Platform environments for Cloudera AI..... 13

Use a non-transparent proxy with Cloudera AI on AWS environments.....23

Certified scale limitations for Cloudera AI Workbenches on AWS..... 24

AWS Account Prerequisites for Cloudera AI Workbenches

To successfully provision an Cloudera AI Workbench, there are many prerequisites that you must ensure are met. Carefully go through this section step by step.

1. Review the AWS Account Prerequisites for Cloudera Data Platform

   Verify that the AWS account that you would like to use for Cloudera Data Platform has the required resources and that you have the permissions required to manage these resources.

   Instructions: AWS Account Requirements
2. Review the Cloudera AI-Specific AWS Resource Requirements

   Provisioning an Cloudera AI Workbench will require access to the following AWS resources. Make sure your AWS account has access to these resources.

   - AWS Services used by Cloudera AI

     a. Compute - Amazon Elastic Kubernetes Service (EKS)
     b. Load Balancing - Amazon Network Load Balancer (NLB)
     c. Key Management - AWS Key Management Service (KMS)
     d. DNS - Amazon Route 53, hosted by Cloudera
     e. Persistent Storage - Amazon Elastic Block Store (EBS)
     f. Project File Storage - Amazon Elastic File System (EFS) for project file storage
     g. Command Line Interface - AWS Command Line Interface (CLI).
     h. Security Token Service - AWS Security Token Service (STS)
   - VPC Requirements - You can either use an existing VPC or allow Cloudera Data Platform to create one for you.
     - Option 1. Using your own VPC
       - Recommended requirements: Divide the address space according to the following recommended sizes:
         - 3 x /19 private subnets. Each subnet should be created in a separate Availability Zone for the EKS worker nodes.
         - 3 x /24 public subnets. These should also be created in three separate Availability Zones, using the same zones as the private subnets.
         - Ensure the CIDR block for the subnets is sized appropriately.
         - You must enable Amazon DNS with the VPC. Corporate DNS is not supported. For guidelines on how verify your DNS settings, refer to sections 1-3 in AWS environment requirements checklist for the Data Warehouse service.

           Note: There is no way to increase the subnet size without recreating the environment and VPC.

           Private subnets should have routable IPs over your internal VPN. If IPs are not routable, private Cloudera AI endpoints will need to be accessed via a SOCKS proxy. Cloudera recommends creating routable IPs by setting up VPN connections between networks, and not using any public load balancers.

If a fully-private network configuration is not feasible, use of a SOCKS proxy to access Cloudera AI is possible, but is not recommended.

Tag the VPC and the subnets as shared so that Kubernetes can find them. For load balancers to be able to choose the subnets correctly, you are also required to tag private subnets with the kubernetes.io/role/internal-elb:1 tag, and public subnets with the kubernetes.io/role/elb:1 tag.

- Option 2. Cloudera Data Platform creates a new VPC

  If you choose to allow Cloudera Data Platform to create a new VPC, three subnets will be created automatically. One subnet is created for each availability zone assuming three AZs per region; If a region has two AZs instead of three, then still three subnets are created, two in the same AZ.

  You will be asked to specify a valid CIDR in IPv4 range that will be used to define the range of private IPs for EC2 instances provisioned into these subnets.
  - Related AWS documentation: Amazon EKS - Cluster VPC Considerations, Creating a VPC for your Amazon EKS Cluster
- Ports Requirements

  HTTPS access to Cloudera AI Workbenches is available over port 443 for the following cases:

  - internal only - should be accessible from your organization's network, but not the public internet
  - internet facing - should be accessible from the public internet as well as your internal organization's network

  This is in addition to the ports requirements noted here for Cloudera Data Platform's default security group: Management Console - Security groups.
- Firewall requirements

  Installations must comply with firewall requirements set by cloud providers at all times. Ensure that ports required by the provider are not closed. For example, Kubernetes services have requirements documented in *Amazon EKS security group considerations*.

  Also, for information on repositories that must be accessible to set up Cloudera AI Workbenches, see *Outbound network access destinations for AWS*.

3. Review the default AWS service limits and your current AWS account limits

By default, AWS imposes certain default limits for AWS services, per-user account. Make sure you review your account's current usage status and resource limits before you start provisioning additional resources for Cloudera Data Platform and Cloudera AI.

For example, depending on your AWS account, you might only be allowed to provision a certain number of CPU instances, or you might not have default access to GPU instances at all. Make sure to review your AWS service limits before your proceed.

Related AWS documentation: AWS Service Limits, Amazon EC2 Resource Limits.

4. Review supported AWS regions

Cloudera Data Platform supports the following AWS regions: Supported AWS regions. However, the Cloudera AI service requires AWS Elastic Kubernetes Service (EKS). Make sure you select a region that includes EKS.

Related AWS documentation: Region Table (AWS Documentation).

5. Set up an AWS Cloud Credential

Create a role-based AWS credential that allows Cloudera Data Platform to authenticate with your AWS account and has authorization to provision AWS resources on your behalf. Role-based authentication uses an IAM role with an attached IAM policy that has the minimum permissions required to use Cloudera Data Platform.

Once you have created this IAM policy, register it in Cloudera Data Platform as a cloud credential. Then, reference this credential when you are registering the environment in the next step.

Instructions: Introduction to the role-based provisioning credential for AWS

6. Register an AWS Environment

   A Cloudera Data Platform User with the role of Power User must register an environment for their organization. An environment determines the specific cloud provider region and virtual network in which resources can be provisioned, and includes the credential that should be used to access the cloud provider account.

   Instructions: Register an AWS Environment
7. Ensure private subnets have outbound internet connectivity

   Also, ensure that your private subnets have outbound internet connectivity. Check the route tables of private subnets to verify the internet routing. Worker nodes must be able to download Docker images for Kubernetes, billing and metering information, and to perform API server registration.
8. Ensure the Amazon Security Token Service (STS) is activated

   To successfully activate an environment in the Data Warehouse service, you must ensure the Amazon STS is activated in your AWS VPC:

   a. In the AWS Management Console home page, select IAM under Security, Identity, & Compliance.
   b. In the Identity and Access Management (IAM) dashboard, select Account settings in the left navigation menu.
   c. On the Account settings page, scroll down to the section for Security Token Service (STS).
   d. In the Endpoints section, locate the region in which your environment is located and make sure that the STS service is activated.
9. Cloudera AI Role Requirements

   There are two Cloudera Data Platform user roles associated with the Cloudera AI service: MLAdmin and MLUser. Any Cloudera Data Platform user with the EnvironmentAdmin (or higher) access level must assign these roles to users who require access to the Cloudera AI service within their environment.

   Furthermore, if you want to allow users to log in to provisioned Cloudera AI Workbenches and run workloads on them, this will need to be configured separately.

   Instructions: Configuring User Access to Cloudera AI Workbenches

Related Information

Amazon EKS security group considerations
Outbound network access destinations for AWS

Limitations on AWS

This section lists some resource limits that Cloudera AI and AWS impose on workloads running in Cloudera AI Workbenches.

- Certificate creation (for TLS) uses LetsEncrypt which is limited to 2000 certs/week. As such a single tenant in Cloudera Data Platform can create a maximum of 2000 Cloudera AI Workbenches per week.
- Cloudera AI imposes a limit (50) on the number of pods a user can create at any point within a specific Cloudera AI Workbench. This limit is not configurable.
- Cloudera AI allows you to provision a maximum of 100 compute nodes per Cloudera AI Workbench. This does not include any additional infrastructure nodes Cloudera AI might need to provision to run the service.
- Amazon EKS imposes a limit on the number of pods you can run simultaneously on a node. This limit varies depending on your instance type. For details, see ENI Max Pods.
- Cloudera AI creates one Amazon Elastic File System per Cloudera AI Workbench. The number of Elastic File Systems in a region is limited to 1000 per account. Therefore, the number of Cloudera AI Workbenches in a region is limited to 1000 at any given time for a given account.

Related Information

Supported AWS regions

Network Planning for Cloudera AI on AWS

Before attempting to deploy your AWS virtual network and set up your AWS Environment and Cloudera AI Workbenches, you should plan the network.

Setting up Cloudera AI for AWS is comprehensively covered in the Getting Started guide. See *AWS account requirements* for more information.

Related Information

AWS account requirements

AWS IAM restricted roles and policies for compute and Cloudera AI

AWS IAM write permissions are used by the Cloudera AI compute infrastructure to create and delete roles and instance profiles.

Some customers may not be willing to provide IAM write permissions in the role's policy. Instead, customers can set up static pre-created roles and instance profiles defined and used by the Cloudera AI compute infrastructure to provision clusters.

Note:

- The compute infrastructure is only able to use the pre-created roles and instance profile if the entitlement LIFTIE_USE_PRECREATED_IAM_RESOURCES is set for the tenant in use.
- The pre-created roles and instance profiles should continue to exist for the lifetime of the cluster.

The two main tasks are:

1. Create roles and an instance profile.
2. Create restricted IAM policies for use by the compute infrastructure.

After these two tasks are completed, you can create the cross-account credential, if needed.

See the following topics for the procedures for creating the roles and policies.

Create IAM roles and instance profile pair

This step describes the roles and instance profiles that you create and attach to EKS master and worker instances at runtime. This step is needed in customer environments where write permissions are not provided to Cloudera AI. The roles created here are used exclusively within the customer's account.

Use the following CloudFormation template to create:

- IAM role called cdp-eks-master-role
- IAM role and Instance Profile pair called cdp-liftie-instance-profile

1. To apply the template, you need to provide values for the following parameters in the AWS console CloudFormation wizard:
   - Stack Name: Provide an appropriate name. (Example : compute-precreated-roles-and-instanceprofile)
   - TelemetryLoggingBucket: Name of the log bucket (just the name, not s3://) (Example : compute-logging-bucket)
   - TelemetryLoggingEnabled: Set it to true
   - TelemetryLoggingRootDir: Keep the default value (which is cluster-logs)
   - TelemetryKMSKeyARN: If the telemetry bucket is encrypted, give the KMS Key ARN. Default value is null.

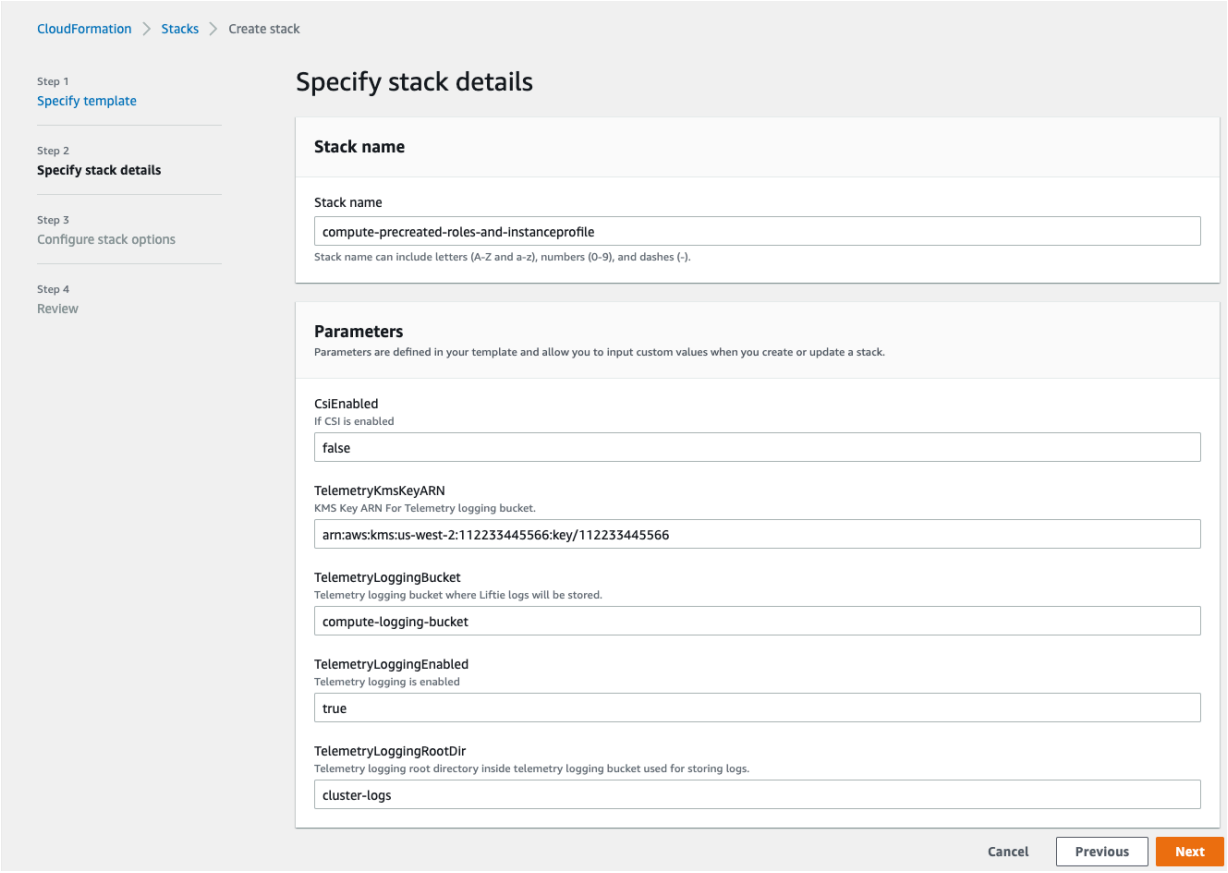

2. On the last page of the wizard, select the checkbox to allow creation of IAM resources with special names. If not selected, CloudFormation prepends the provided name with random prefixes to ensure uniqueness.

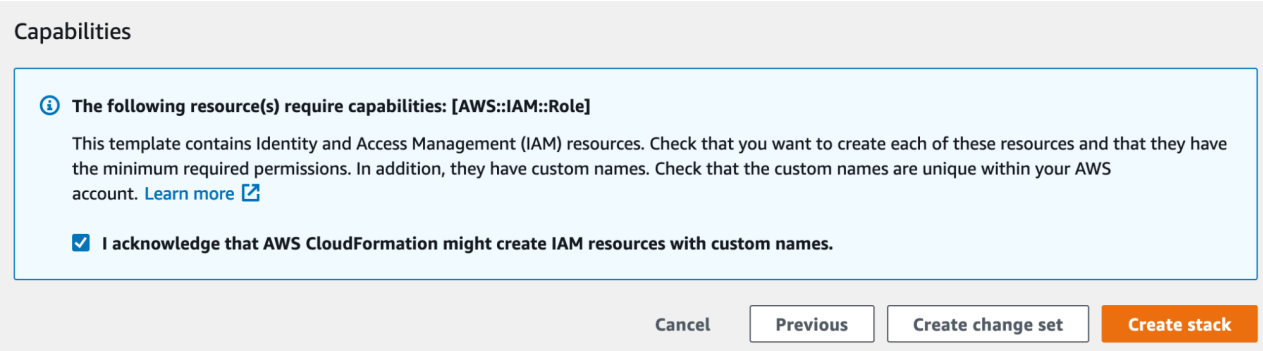

The result of this procedure resembles the following:

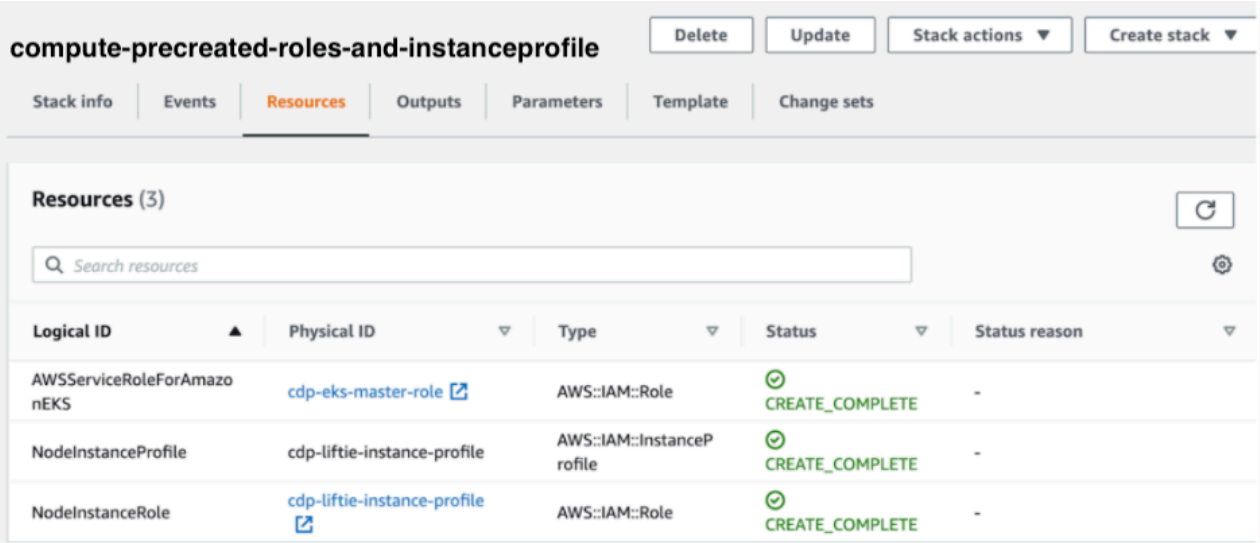

Use the following CloudFormation template for this process.

CloudFormation Template (format: YAML)

```
AWSTemplateFormatVersion: 2010-09-09
Description: Creates Liftie IAM resources
Parameters:
  TelemetryLoggingEnabled:
    Description: Telemetry logging is enabled
    Type: String
  TelemetryLoggingBucket:
    Description: Telemetry logging bucket where Liftie logs will be stored.
    Type: String
  TelemetryKmsKeyARN:
    Description: KMS Key ARN For Telemetry logging bucket.
    Type: String
    Default: ""
  TelemetryLoggingRootDir:
    Description: Telemetry logging root directory inside telemetry logging
bucket used for storing logs.
    Default: "cluster-logs"
    Type: String
Conditions:
  TelemetryLoggingEnabled:
    Fn::Equals:
      - {Ref: TelemetryLoggingEnabled}
      - true
  KMSKeyARNForTelemetryLoggingBucketIsEmpty: !Not [!Equals [!Ref Telemetry
KmsKeyARN, ""]]
Resources:
  AWSServiceRoleForAmazonEKS:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - eks.amazonaws.com
            Action:
              - sts:AssumeRole
      ManagedPolicyArns:
```

```
        - arn:aws:iam::aws:policy/AmazonEKSServicePolicy
        - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
      RoleName: cdp-eks-master-role
  NodeInstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: "/"
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
        - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
        - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
      RoleName: cdp-liftie-instance-profile
      Policies:
        - PolicyName: ssm-required
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - ssm:GetParameters
                Resource:
                  - "*"
        - PolicyName: cluster-autoscaler
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - autoscaling:DescribeAutoScalingGroups
                  - autoscaling:DescribeAutoScalingInstances
                  - autoscaling:DescribeTags
                  - autoscaling:DescribeLaunchConfigurations
                  - autoscaling:SetDesiredCapacity
                  - autoscaling:TerminateInstanceInAutoScalingGroup
                  - ec2:DescribeLaunchTemplateVersions
                Resource:
                  - "*"
        - PolicyName: ebs-csi
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - ec2:CreateSnapshot
                  - ec2:AttachVolume
                  - ec2:DetachVolume
                  - ec2:ModifyVolume
                  - ec2:DescribeAvailabilityZones
                  - ec2:DescribeInstances
                  - ec2:DescribeSnapshots
                  - ec2:DescribeTags
                  - ec2:DescribeVolumes
                  - ec2:DescribeVolumesModifications
                Resource: "*"
              - Effect: Allow
```

```
                  Action:
                    - ec2:CreateTags
                  Resource:
                    - "arn:aws:ec2:*:*:volume/*"
                    - "arn:aws:ec2:*:*:snapshot/*"
                  Condition:
                    StringEquals:
                      "ec2:CreateAction":
                        - CreateVolume
                        - CreateSnapshot
                - Effect: Allow
                  Action:
                    - ec2:DeleteTags
                  Resource:
                    - "arn:aws:ec2:*:*:volume/*"
                    - "arn:aws:ec2:*:*:snapshot/*"
                - Effect: Allow
                  Action:
                    - ec2:CreateVolume
                  Resource: "*"
                  Condition:
                    StringLike:
                      "aws:RequestTag/ebs.csi.aws.com/cluster": "true"
                - Effect: Allow
                  Action:
                    - ec2:CreateVolume
                  Resource: "*"
                  Condition:
                    StringLike:
                      "aws:RequestTag/CSIVolumeName": "*"
                - Effect: Allow
                  Action:
                    - ec2:CreateVolume
                  Resource: "*"
                  Condition:
                    StringLike:
                      "aws:RequestTag/kubernetes.io/cluster/*": "owned"
                - Effect: Allow
                  Action:
                    - ec2:DeleteVolume
                  Resource: "*"
                  Condition:
                    StringLike:
                      "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
                - Effect: Allow
                  Action:
                    - ec2:DeleteVolume
                  Resource: "*"
                  Condition:
                    StringLike:
                      "ec2:ResourceTag/CSIVolumeName": "*"
                - Effect: Allow
                  Action:
                    - ec2:DeleteVolume
                  Resource: "*"
                  Condition:
                    StringLike:
                      "ec2:ResourceTag/kubernetes.io/created-for/pvc/name":
"*"
                - Effect: Allow
                  Action:
                    - ec2:DeleteSnapshot
                  Resource: "*"
                  Condition:
```

```
                  StringLike:
                    "ec2:ResourceTag/CSIVolumeSnapshotName": "*"
                - Effect: Allow
                  Action:
                    - ec2:DeleteSnapshot
                  Resource: "*"
                  Condition:
                    StringLike:
                      "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
        - PolicyName: efs-csi
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - elasticfilesystem:DescribeAccessPoints
                  - elasticfilesystem:DescribeFileSystems
                  - elasticfilesystem:DescribeMountTargets
                  - elasticfilesystem:TagResource
                Resource: "*"
              - Effect: Allow
                Action:
                  - elasticfilesystem:CreateAccessPoint
                Resource: "*"
                Condition:
                  StringLike:
                    "aws:RequestTag/efs.csi.aws.com/cluster": "true"
              - Effect: Allow
                Action:
                  - elasticfilesystem:DeleteAccessPoint
                Resource: "*"
                Condition:
                  StringEquals:
                    "aws:ResourceTag/efs.csi.aws.com/cluster": "true"
        - !If
          - TelemetryLoggingEnabled
          - PolicyName: telemetry-s3-list-bucket
            PolicyDocument:
              Version: 2012-10-17
              Statement:
                - Effect: Allow
                  Action:
                    - s3:ListBucket
                  Resource:
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}'
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}/${Tele
metryLoggingRootDir}/*'
          - !Ref 'AWS::NoValue'
        - !If
          - TelemetryLoggingEnabled
          - PolicyName: telemetry-s3-read-write
            PolicyDocument:
              Version: 2012-10-17
              Statement:
                - Effect: Allow
                  Action:
                    - s3:*Object
                    - s3:AbortMultipartUpload
                    - s3:GetBucketAcl
                  Resource:
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}'
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}/${Telemet
ryLoggingRootDir}/*'
          - !Ref 'AWS::NoValue'
```

```
        - !If
          - KMSKeyARNForTelemetryLoggingBucketIsEmpty
          - PolicyName: s3-kms-read-write-policy
            PolicyDocument:
              Version: 2012-10-17
              Statement:
                - Effect: Allow
                  Action:
                    - kms:Decrypt
                    - kms:GenerateDataKey
                  Resource:
                    - !Sub ${TelemetryKmsKeyARN}
          - !Ref 'AWS::NoValue'
        - PolicyName: calico-cni
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - ec2:ModifyInstanceAttribute
                Resource:
                  - "*"
  NodeInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      InstanceProfileName: cdp-liftie-instance-profile
      Roles:
        - !Ref NodeInstanceRole
```

Create role and policy used to deploy Cloudera Data Platform environments for Cloudera AI

The Cloudera AI control plane requires a role and policies to create Cloudera Data Platform environments. In this step, you create a common policy for creating environments, as well as a policy that is specific to Cloudera AI environments.

The following two policies are created in this step:

- Compute infrastructure restricted IAM policy - A common policy for all data services deployed on Cloudera Data Platform.
- Cloudera AI restricted IAM policy - A policy with additional permissions for Cloudera AI.

There are two options for the timing of attaching the role: during environment creation, or prior to enabling the Cloudera AI data service.

Option #1: During environment creation

The Cloudbreak environment creation UI should be set up as shown here:

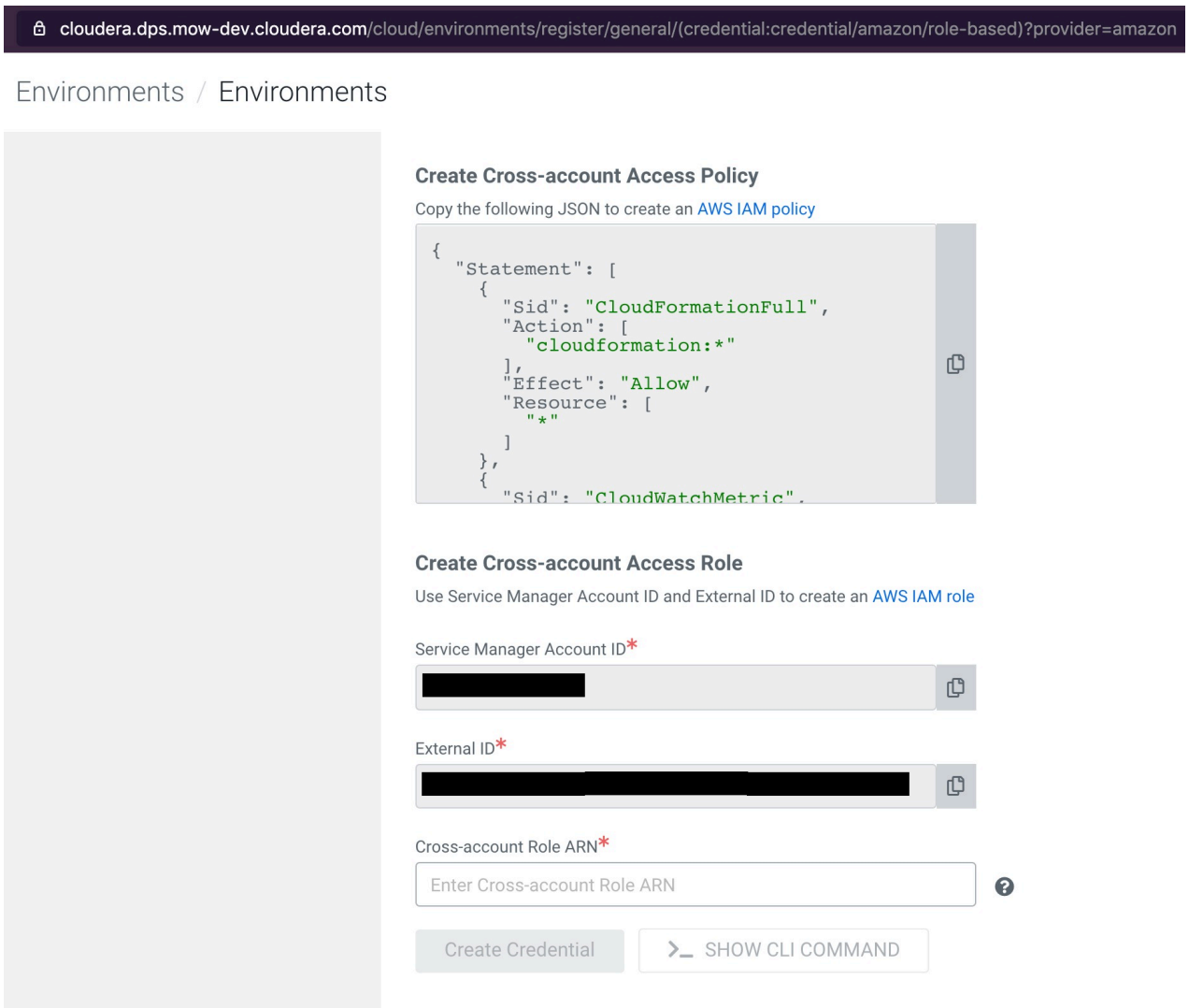

Note:

- For the AWS IAM policy mentioned in the “Create Cross-account Access Policy” section, use the Compute infrastructure Restricted IAM and Cloudera AI Restricted IAM policies below and create as new policies in AWS IAM. There may be one or more restricted policies already attached to the cross-account role, in addition to the Compute infrastructure and Cloudera AI restricted policies. For example, there may also be a Cloudera Data Hub restricted policy.
- For the “Create Cross-account Access Role” section, create the cross-account role as instructed (or update the role if one already exists) and attach the newly created Compute infrastructure Restricted IAM policy and Cloudera AI Restricted IAM policy. Finally, update the cross-account role to use it.

Option #2: Prior to enabling Cloudera AI data service

If the Cloudbreak environment has already been created, you can create and attach the Compute infrastructure Restricted IAM policy and Cloudera AI restricted IAM policy to the existing cross-account role associated with the environment.

To view the existing cross-account role, in the Environments section of the Cloudera Data Platform Management Console, on the Summary tab, see Credentials.

Note: There may be one or more restricted policies already attached to the cross-account role, in addition to the Compute infrastructure and Cloudera AI restricted policies. For example, there might be a Data Hub restricted policy. These should be left in place.

Compute (Liftie) Restricted IAM policy

Replace the following placeholders in the JSON file:

- [YOUR-ACCOUNT-ID] with your account ID in use.
- [YOUR-IAM-ROLE-NAME] with the IAM restricted role associated with this policy.
- [YOUR-SUBNET-ARN-*] supplied during the Cloudbreak Environment(s) creation. Note: Please provide all the subnets present in all the Cloudbreak Environment(s) that you intend to use it for the experience. If at any point a new Cloudbreak Environment is created or an existing one is updated for subnets, the same should be updated here.
- [YOUR-IDBROKER-ROLE-NAME] with the ID Broker Role name in use.
- [YOUR-LOG-ROLE-NAME] with the Log Role name in use.
- [YOUR-KMS-CUSTOMER-MANAGED-KEY-ARN] with KMS key ARN.
- [YOUR-ACCOUNT-REGION] with the AWS region.

```
{
  "Version": "2012-10-17",
  "Id": "ComputePolicy_v10",
  "Statement": [
    {
      "Sid": "SimulatePrincipalPolicy",
      "Effect": "Allow",
      "Action": [
        "iam:SimulatePrincipalPolicy"
      ],
      "Resource": [
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-IAM-ROLE-NAME]"
      ]
    },
    {
      "Sid": "RestrictedPermissionsViaClouderaRequestTag",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:CreateChangeSet",
        "ec2:createTags",
        "eks:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/Cloudera-Resource-Name": [
            "crn:cdp:*"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedPermissionsViaClouderaResourceTag",
      "Effect": "Allow",
      "Action": [
        "autoscaling:DetachInstances",
        "autoscaling:ResumeProcesses",
        "autoscaling:SetDesiredCapacity",
        "autoscaling:SuspendProcesses",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:DeleteTags",
        "autoscaling:TerminateInstanceInAutoScalingGroup",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStacks"
      ],
      "Resource": "*",
```

```
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/Cloudera-Resource-Name": [
            "crn:cdp:*"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedPermissionsViaCloudFormation",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateSecurityGroup",
        "ec2:DeleteSecurityGroup",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:CreateLaunchTemplate",
        "ec2:DeleteLaunchTemplate",
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:DeleteAutoScalingGroup",
        "autoscaling:CreateOrUpdateTags",
        "autoscaling:CreateLaunchConfiguration",
        "eks:CreateCluster",
        "eks:DeleteCluster"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedEC2PermissionsViaClouderaResourceTag",
      "Effect": "Allow",
      "Action": [
        "ec2:RebootInstances",
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ForAnyValue:StringLike": {
          "ec2:ResourceTag/Cloudera-Resource-Name": [
            "crn:cdp:*"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedIamPermissionsToClouderaResources",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-IDBROKER-ROLE-NAME]",
```

```
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-LOG-ROLE-NAME]",
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/liftie-*-eks-service-role",
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/liftie-*-eks-worker-nodes",
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/cdp-eks-master-role",
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/cdp-liftie-instance-profile"
      ]
    },
    {
      "Sid": "RestrictedKMSPermissionsUsingCustomerProvidedKey",
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*"
      ],
      "Resource": [
        "[YOUR-KMS-CUSTOMER-MANAGED-KEY-ARN]"
      ]
    },
    {
      "Sid": "AllowCreateDeleteTagsForSubnets",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags",
        "ec2:DeleteTags"
      ],
      "Resource": [
        "arn:aws:ec2:[YOUR-SUBNET-REGION]:[YOUR-ACCOUNT-ID]:subnet/*"
      ]
    },
    {
      "Sid": "OtherPermissionsViaCloudFormation",
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeScheduledActions",
        "autoscaling:DescribeTags",
        "autoscaling:DescribeAutoScalingInstances",
        "autoscaling:DescribeLaunchConfigurations",
        "autoscaling:DeleteLaunchConfiguration",
        "autoscaling:DescribeScalingActivities",
        "dynamodb:DescribeTable",
        "ec2:DeletePlacementGroup",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeImages",
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeLaunchTemplateVersions",
        "ec2:DescribeLaunchTemplates",
        "ec2:DescribePlacementGroups",
        "ec2:DescribeRegions",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVolumes"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
```

```
            "cloudformation.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "ModifyInstanceAttribute",
      "Effect": "Allow",
      "Action": [
        "ec2:ModifyInstanceAttribute"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "ec2:Attribute": "SourceDestCheck"
        }
      }
    },
    {
      "Sid": "OtherPermissionsViaClouderaResourceTag",
      "Effect": "Allow",
      "Action": [
        "cloudformation:DescribeChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:ListStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResource",
        "cloudformation:DescribeStackResources",
        "cloudwatch:deleteAlarms",
        "cloudwatch:putMetricAlarm",
        "logs:DescribeLogStreams",
        "logs:FilterLogEvents",
        "ec2:AttachVolume",
        "ec2:CreateNetworkInterface",
        "ec2:CreateVolume",
        "ec2:DeleteVolume",
        "ec2:RunInstances",
        "eks:ListUpdates",
        "eks:UpdateClusterConfig",
        "eks:UpdateClusterVersion",
        "eks:DescribeUpdate",
        "iam:GetRolePolicy",
        "iam:ListInstanceProfiles",
        "iam:ListRoleTags",
        "iam:RemoveRoleFromInstanceProfile",
        "iam:TagRole",
        "iam:UntagRole"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/Cloudera-Resource-Name": [
            "crn:cdp:*"
          ]
        }
      }
    },
```

```
    {
      "Sid": "OtherPermissions",
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateLaunchTemplateVersion",
        "ec2:CreatePlacementGroup",
        "ec2:DeleteKeyPair",
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInstanceTypes",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcs",
        "ec2:ImportKeyPair",
        "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
        "ec2:GetInstanceTypesFromInstanceRequirements",
        "eks:DescribeCluster",
        "elasticloadbalancing:DescribeLoadBalancers",
        "iam:GetRole",
        "iam:ListRoles",
        "iam:GetInstanceProfile"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowSsmParams",
      "Effect": "Allow",
      "Action": [
        "ssm:DescribeParameters",
        "ssm:GetParameter",
        "ssm:GetParameters",
        "ssm:GetParameterHistory",
        "ssm:GetParametersByPath"
      ],
      "Resource": [
        "arn:aws:ssm:*:*:parameter/aws/service/eks/optimized-ami/*"
      ]
    },
    {
      "Sid": "CfDeny",
      "Effect": "Deny",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ForAnyValue:StringLike": {
          "cloudformation:ImportResourceTypes": [
            "*"
          ]
        }
      }
    },
    {
      "Sid": "ForAutoscalingLinkedRole",
      "Effect": "Allow",
      "Action": [
```

```
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/autoscaling-
plans.amazonaws.com/AWSServiceRoleForAutoScalingPlans_EC2AutoScaling"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "autoscaling-plans.amazonaws.com"
        }
      }
    },
    {
      "Sid": "ForEksLinkedRole",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/eks.amazonaws
.com/AWSServiceRoleForEKS"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "eks.amazonaws.com"
        }
      }
    }
  ]
}
```

Supporting Customer Managed CMKs

Along with providing the KMS Customer Managed Customer Master Key (CMK) for volume encryption in the policy section with Sid: RestrictedKMSPermissionsUsingCustomerProvidedKey, you need to verify that the policy for the Customer Managed Customer Master Key (CMK) at KMS (this is not an IAM policy) has the following three permission blocks defined for AWSServiceRoleForAutoScaling.

```
{
  "Statement": [
    {
      "Sid": "AllowAutoscalingServiceLinkedRoleForAttachmentOfPersistentRes
ources",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/auto
scaling.amazonaws.com/AWSServiceRoleForAutoScaling"
      },
      "Action": "kms:CreateGrant",
      "Resource": "*",
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Sid": "AllowAutoscalingServiceLinkedRoleUseOfTheCMK",
      "Effect": "Allow",
      "Principal": {
```

```
        "AWS": "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/autos
caling.amazonaws.com/AWSServiceRoleForAutoScaling"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow EKS access to EBS.",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "kms:CreateGrant",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "[YOUR-ACCOUNT-ID]",
          "kms:viaService": "ec2.[YOUR-ACCOUNT-REGION].amazonaws.com"
        }
      }
    }
  ]
}
```

After the policy is attached, the KMS service page will show the CMS as having the policy attached, similar to this screen shot:

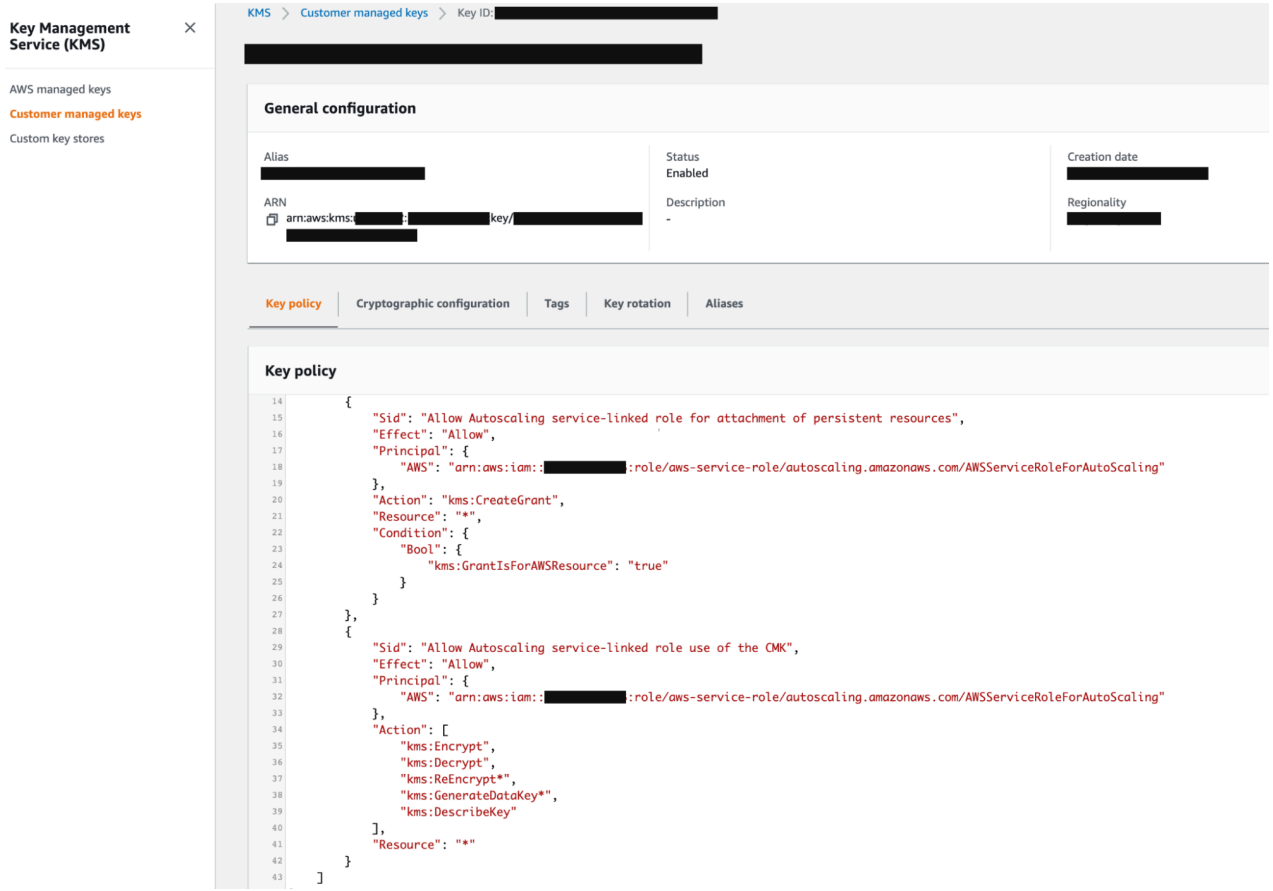

Cloudera AI restricted IAM policy

Replace the following placeholders in the JSON file:

- [YOUR-ACCOUNT-ID] with your account ID in use.
- [YOUR-IAM-ROLE-NAME] with the IAM restricted role with which this policy would be associated with.

```
{
    "Version": "2012-10-17",
    "Id": "CMLPolicy_v1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:SimulatePrincipalPolicy",
            "Resource": "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-IAM-ROLE-
NAME]"
        },
        {
            "Sid": "RestrictedPermissionsViaClouderaRequestTag",
            "Effect": "Allow",
            "Action":[
                "elasticfilesystem:CreateFileSystem"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike":{
                    "aws:RequestTag/Cloudera-Resource-Name": "crn:cdp:*"
                }
            }
```

```
        },
        {
            "Sid": "OtherPermissions",
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:DescribeMountTargets",
                "elasticfilesystem:DeleteAccessPoint",
                "elasticfilesystem:CreateMountTarget",
                "elasticfilesystem:DescribeAccessPoints",
                "elasticfilesystem:DescribeFileSystems",
                "elasticfilesystem:DeleteMountTarget",
                "elasticfilesystem:CreateAccessPoint",
                "elasticfilesystem:DeleteFileSystem",
                "elasticfilesystem:DescribeMountTargetSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ForEFSLinkedRole",
            "Effect": "Allow",
            "Action": [
              "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
              "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/elastic
filesystem.amazonaws.com/AWSServiceRoleForAmazonElasticFileSystem"
            ],
            "Condition": {
              "StringLike": {
                "iam:AWSServiceName": "elasticfilesystem.amazonaws.com"
              }
            }
          }
    ]
}
```

Use a non-transparent proxy with Cloudera AI on AWS environments

Cloudera AI can use non-transparent proxies if the environment is configured to use a network proxy in Management Console.

Enterprise customers frequently need to deploy Cloudera Data Platform in a virtual network that does not have direct internet access. Specifically, the proxy server may be located in a different virtual network, in order to filter traffic for allowed domains or IPs.

Transparent and non-transparent network proxies differ in the following ways.

Transparent network proxy

- Proxy is unknown to clients and requires no additional client configuration.
- Usually, connections by way of transparent proxies are configured in route tables on your AWS VPC.

Non-transparent proxy

- Clients are aware of non-transparent proxies and each client must be specifically configured to use the non-transparent proxy connection.

- You pass connection or security information (username/password) along with the connection request sent by clients.

You can configure an AWS environment to use non-transparent proxy connections when activating environments for Cloudera AI.

Use a non-transparent proxy in a different VPC

If the customer wants to copy the hostname for the non-transparent proxy and the non-transparent proxy is configured in a different VPC, then Cloudera Data Platform needs the CIDR of the non-transparent proxy to allow the inbound access. To configure this, in the Provision Cloudera AI Workbench UI, select Use hostname for non-transparent proxy and enter the CIDR range in Inbound Proxy CIDR Ranges.

Related Information

Using a non-transparent proxy

Certified scale limitations for Cloudera AI Workbenches on AWS

The listed scale limitations have been certified with Cloudera AI Workbenches on AWS.

The Cloudera AI Workbench is certified to support the following:

- 1000 logged in users using the product concurrently
- 850 concurrent interactive Sessions
- 500 concurrent cron jobs executing at the same frequency
- 500 concurrent Applications
- 300 Models deployed concurrently

Cloudera AI

Azure Requirements

Date published: 2020-07-16
Date modified: 2024-11-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Azure Account Requirements for Cloudera AI Workbenches..... 4

Limitations on Azure.....4

Network Planning for Cloudera AI on Azure..... 5

Azure Environment Setup for Cloudera AI Workbenches..... 5

Create subnets..... 6

Create a subnet with the CLI..... 6

Create Azure Files Storage Account and File Share..... 7

Create Azure NetApp Files Account, Capacity Pool and Volume..... 7

Create a Volume..... 8

Other NFS Options..... 8

Set up minimum permissions..... 9

Migrating from generic NFS to Azure Files NFS in Cloudera AI..... 11

Backing up workbench..... 15

Restoring old data to a new workbench..... 15

Azure Account Requirements for Cloudera AI Workbenches

The requirements for using Cloudera AI Workbenches in Azure are described in *Working with Azure environments*, which is linked in the Related information section, below.

In addition, Cloudera AI on Azure has a few additional requirements:

- Cloudera AI requires one Azure virtual network.
- Each Cloudera AI Workbench requires its own subnet.
- Each Cloudera AI Workbench requires its own NFS file share.
- Azure Files NFS 4.1, or Azure NetApp Files, are the recommended services, in order of preference. For Azure Files NFS 4.1, only NFS version 4.1 is supported. For Azure NetApp Files, only NFS version 3 is supported. We require the "No Root Squash" (or Allow Root Access) export option, and the volume must be shared with read and write access. The NFS volume must be created with at least 100 GB of storage capacity to ensure adequate I/O performance. It is also possible, although not recommended, to use an external NFS. For more information, see Other NFS Options in Related information.

Related Information

Working with Azure environments

Other NFS Options

Use Azure Firewall to protect Azure Kubernetes Service (AKS) Deployments

Limitations on Azure

This section lists some resource limits that Cloudera AI and Azure impose on workloads running in Cloudera AI Workbenches.

- There is no ability to grant or revoke remote access (via Kubeconfig) to specific users. Users with the MLAdmin role in the environment can download a Kubeconfig file. The Kubeconfig file will continue to allow access even if the MLAdmin role is later revoked.
- Support is limited to regions that provide AKS. Also, customers should check availability of Azure Files NFS or Azure NetApp Files and GPU instance types in their intended region. See Supported Azure regions for more information.
- Data is not encrypted in transit to Azure Files NFS or Azure NetApp Files or other NFS systems, so make sure to implement policies to ensure security at the network level.
- Each Cloudera AI Workbench requires a separate subnet. For more information on this issue, see Use kubenet networking with your own IP address ranges in Azure Kubernetes Service (AKS).
- Heavy AKS activity can cause default API rate limits to trigger, causing throttling and eventually failures for AKS clusters. For some examples, see AKS issue 1187 and AKS issue 1413.
- When you provision an Azure Kubernetes (AKS) cluster, a Standard load balancer is provisioned by default. The Standard load balancer always provisions a public IP for egress traffic, communication with the Kubernetes control plane, and backwards compatibility. Cloudera software does not use this public IP directly, or expose anything on it. For more information, see: Use a public Standard Load Balancer in Azure Kubernetes Service (AKS)
- The following Azure Policy add-on for AKS has a default policy that causes workbench upgrades to fail. To avoid this problem, the Kubernetes clusters should not allow container privilege escalation policy should not be enabled. For more information on AKS policies, see Azure Policy built-in definitions for Azure Kubernetes Service.
- The workbench Backup and Restore feature, which on Azure is only available through the Cloudera Data Platform CLI, does not perform a backup of NFS.

Related Information

Supported Azure regions

Network Planning for Cloudera AI on Azure

Before attempting to deploy your Azure virtual network and set up your Azure Environment and Cloudera AI Workbenches, you should plan the network.

As an example, a minimum architecture to support two Cloudera AI Workbenches would comprise the following:

- An Azure virtual network. Cloudera AI can use an existing virtual network if available.
- One subnet dedicated to the Azure NetApp Files service, if this service is utilized. Azure FIles NFS does not need a dedicated subnet.
- One subnet for each Cloudera AI Workbench.

Keep the following considerations in mind when planning your network:

- Each Cloudera AI Workbench requires one subnet in the virtual network.
- Plan the CIDR addresses for each subnet so that the ranges do not overlap.
- Each subnet should use a /26 CIDR. This should accommodate a maximum of 30 worker nodes as well as 4 infrastructure nodes for Cloudera AI.
- To use GPUs, create a virtual network with /25 CIDR subnets to accommodate a maximum of 30 GPU nodes. If you created a /26 CIDR network originally, and then subsequently need to add GPU support, you must create a new network with /25 CIDR subnets. Subnets cannot be resized.
- Azure Files NFS v4.1 (first preference) and Azure NetApp Files v3 (second preference) are the recommended NFS services for use with Cloudera AI on Azure.
- Subnets may not use the following reserved CIDR blocks: 10.0.0.0/16 or 10.244.0.0/16
- Even if private IP addresses are used for the Cloudera AI service, each AKS cluster provisions a public IP for egress traffic, communication with the Kubernetes control plane, and backwards compatibility. For more information, see Use a public Standard Load Balancer in Azure Kubernetes Service (AKS).
- In a Single Resource Group setup in Azure, the entire Cloudera Data Platform stack is supposed to use the resource group provided by the customer, and not create any new resource groups. However, note that Azure will automatically create new resource groups for Azure managed resources, such as a new resource group for AKS compute worker nodes.
- Each Cloudera AI Workbench cluster must not share any subnets or routing tables with any other Cloudera Data Platform experience or AKS cluster.
- After provisioning a workbench, do not remove firewall rules to allow inbound and outbound access.
- Installations must comply with firewall requirements set by cloud providers at all times. Ensure that ports required by the provider are not closed. For example, Kubernetes services have requirements documented in Use Azure Firewall to protect Azure Kubernetes Service (AKS) Deployments. Also, for information on repositories that must be accessible to set up workbenches, see *Outbound network access destinations for Azure*.

Related Information

Use a public Standard Load Balancer in Azure Kubernetes Service (AKS)

Use Azure Firewall to protect Azure Kubernetes Service (AKS) clusters

Outbound network access destinations for Azure

Azure Environment Setup for Cloudera AI Workbenches

After an Azure environment has been created, there are several setup steps to prepare the environment for Cloudera AI Workbenches. These steps are required when using Azure Files NFS or Azure NetApp Files, which are the recommended NFS services.

First, ensure that the Azure environment has been created. Instructions to do this can be found here:

- Working with Azure environments
- Azure subscription requirements

The following steps must be performed by an Azure administrator to prepare the environment for use with an Cloudera AI Workbenches.

Create subnets

Within a virtual network, you need to create one or more subnets for workbenches. Cloudera AI Workbenches cannot share subnets, so if you are planning to create multiple Cloudera AI Workbenches you will need to create one subnet for each workbench. In addition, if you plan to use Azure NetApp Files for NFS, one dedicated subnet must be created and delegated to Azure NetApp Files.

Procedure

1. Open the Virtual Network for your environment.
2. Click the Subnets in the left-hand navigation and then +Subnet.
3. Enter the name of the subnet, for example ‘workbench-1’.
4. Enter the IP address range as a CIDR address. A /25 network should be sufficient.
5. Create a subnet for each workbench you plan to provision.
6. Create a subnet named ‘netapp’. Only one subnet needs to be delegated to 'netapp'. The ‘netapp’ subnet is used for Cloudera AI infrastructure purposes only, and cannot be used for a workbench. Make sure that the CIDR ranges specified do not overlap.

   Note: This step is only needed if Azure NetApp Files is used.

7. In the ‘netapp’ subnet Info page, under Subnet delegation, select Microsoft.Netapp/volumes.

   Note: This step is only needed if Azure NetApp Files is used.

8. Click Save.

Create a subnet with the CLI

You can use the CLI to create subnets, as shown in this example.

Example

```
az network vnet subnet create \
--resource-group my-cdp-resource-group \
--vnet-name my-cdp-vnet \
--name my-cml-subnet-1 \
--address-prefix 10.0.10.0/25 \
--service-endpoints Microsoft.Sql
```

This example creates a subnet for the NetApp Files volume:

Note: This is only required if Azure NetApp Files is used.

```
az network vnet subnet create \
--resource-group my-cdp-resource-group \
```

```
--vnet-name my-cdp-vnet \
--name my-anf-subnet \
--address-prefix 10.0.10.0/25 \
--delegations Microsoft.NetApp/volumes
```

The --delegations and --service-endpoints flags can also be applied to an existing subnet by using the `az network vnet subnet` update command.

Create Azure Files Storage Account and File Share

Azure Files NFS v4.1 is a managed, POSIX compliant NFS service on Azure. The file share is used to store files for the Cloudera AI infrastructure and Cloudera AI Workbenches. This is the recommended NFS service for use with Cloudera AI.

About this task

Consult the *Azure Files for NFS documentation*, below, for detailed information on how to create an NFS share. We outline the sequence of steps here.

Procedure

1. Create a FileStorage storage account.
2. Disable secure transfer.
3. See *Create a private endpoint*, below, to create a private endpoint between the FileStorage storage account for each of the subnets that you plan to use for creating the Cloudera AI Workbench. Without this step, Cloudera AI Workbench nodes will not have connectivity to the NFS server.
4. Create NFS file shares, one for each Cloudera AI Workbench. Create each file share with an initial capacity of at least 100 GB. Ensure that the “No Root Squash” export policy is set. This is the default setting.
5. Create a Linux VM in the subnet in which you are going to create the Cloudera AI Workbench, and ensure that you can successfully mount the NFS share in the VM. This step will ensure that there are no connectivity issues to the NFS server from hosts in this subnet.
6. Create a directory within the NFS share to be used as the Cloudera AI Workbench’s projects folder. For example, if the NFS export path is ‘azurenfsv4.file.core.windows.net:/myshare’, and is mounted as ‘/mnt/myshare’, run ‘sudo mkdir /mnt/myshare/ml-workbench-1; sudo chown 8536:8536 /mnt/myshare/ml-workbench-1’ to set ownership to the ‘cdsw’ user ID and group ID.
7. Go to the mount instructions page to copy the server name and export path details. Enter this into the Cloudera AI control plane when prompted for an existing server. Due to a known issue in the control plane UI, the “nfs://” prefix must be specified if you are using the domain name of the file share server, e.g., “nfs://azurenfsv4.file.core.windows.net:/myshare/ml-workbench-1”. Remember to select the NFS protocol version as 4.1.
8. The Linux VM created above can be deleted if there are no issues with mounting the NFS share.

Related Information

Azure Files for NFS documentation

Create a private endpoint

Create Azure NetApp Files Account, Capacity Pool and Volume

Azure NetApp Files is a service on Azure that provides a fully managed, native file share system that is accesible in the Azure cloud. The following procedure is only required if you are using Azure NetApp Files, which is not the recommended NFS service. For more information, see *Quickstart: Set up Azure NetApp Files*.

Procedure

1. In Azure Services, select Azure NetApp Files.
2. In the New NetApp account, create an account.
3. Select the existing Resource Group.
4. In Location, select the region where the Cloudera AI environment is located.
5. Provision the service. When deployment is done, click Go to Resource
6. Under Storage Service # Capacity Pools, click Add pool.
7. For Capacity, enter 4 TiB.

   4 TiB is the minimum size for a capacity pool in Azure NetApp Files. We recommend a Service level of Premium or higher to ensure adequate I/O performance.
8. Click Create.

   In the next steps, create a Volume.

Related Information

Supported Azure regions

Quickstart: Set up Azure NetApp Files and create an NFS volume

Create a Volume

In the Azure portal, you create an Azure NetApp Files volume to provide a file system within the Capacity Pool.

Procedure

1. In the NetApp account screen, click Volumes in the menu on the left, the click Add Volume.
2. For Quota, set a value in GB. You can set an initial value, then increase it to add additional workbenches, without having to resize the pool itself. We recommend an initial volume capacity of at least 100GB in order to ensure adequate I/O performance.
3. Confirm the value in the Delegated Subnet: the only option should be the ‘netapp’ subnet, because that is the one that was delegated to this service.
4. On the Protocol tab, choose the protocol version. Version 3 is supported by Cloudera AI. Ensure that “Root Access” in the “Export policy” section is set to “On”, and “Access” is set to “Read & Write”.
5. On the Review and Create tab, click Create. The volume is deployed after several minutes.
6. After the volume is deployed, obtain the Mount path.
7. Create a Linux VM in the subnet in which the Cloudera AI Workbench will be created, and verify that the NFS volume can be mounted successfully in the VM. This step ensures that there are no connectivity issues between the NFS server and hosts running in the subnet.
8. Create a sub-directory to be used for the Cloudera AI Workbench project files. For example, if the NFS export path is 10.102.47.132:/netapp-vol and is mounted as /mnt/netapp-vol, run sudo mkdir /mnt/netapp-vol/ml-workbench-1; sudo chown 8536:8536 /mnt/netapp-vol/ml-workbench-1 to ensure the directory is owned by the cdsw user ID and group ID.
9. When prompted for the NFS path in the Cloudera AI control plane, enter the path as 10.102.47.132:/netapp-vol/ml-workbench-1.
10. The Linux VM created above can be deleted if there are no issues with mounting the NFS share.

Other NFS Options

As an alternative to Azure Files NFS or Azure NetApp Files, you can configure an NFS server that is external to the Cloudera AI cluster. This is not the recommended approach.

Make sure to check the following points:

- Cloudera AI requires an NFS service that is backed by a fully POSIX compliant file system. NFS service implemented over S3-like blob storage, such as NFS v3 over ADLS, is not supported.
- Currently, Cloudera AI supports NFS versions 4.1 and 3. The NFS client within Cloudera AI must be able to mount the NFS storage with default options, and also assumes these export options:

  ```
  rw,sync,no_root_squash,no_all_squash,no_subtree_check
  ```

  To prevent data loss on disk failure, it is recommended that the NFS export directory resides on a RAID disk. Before attempting to provision your Cloudera AI Workbench, create a Linux VM in the subnet you will use for Cloudera AI.

  Note: Verify that this VM can mount the directory exported by your NFS server.

- Before creating a Cloudera AI Workbench, the storage administrator must create a directory that will be exported to the cluster for storing Cloudera AI project files for that workbench. Either a dedicated NFS export path, or a subdirectory in an existing export must be specified for each workbench. If the directory was previously used for another workbench, there may be extraneous files remaining there. Make sure that all such remaining files are deleted.
- Each Cloudera AI Workbench needs a unique directory that does not have files in it from a different or previous workbench. For example, if 10 Cloudera AI Workbench are expected, the storage administrator will need to create 10 unique directories. Either one NFS export and 10 subdirectories within it need to be created, or 10 unique exports need to be created.

Create a workbench in a dedicated NFS share

For example, to use a dedicated NFS share for a workbench named “workbench1” from NFS server “nfs_server”, do the following:

1. Create NFS export directory “/workbench1”.
2. Change ownership for the exported directory
   a. Cloudera AI accesses this directory as a user with a UID and GID of 8536. So run chown 8536:8536 /workbench1
   b. Make the export directory group-writeable and set GID: chmod g+srwx /workbench1
3. Provide the NFS export path nfs_server:/workbench1 when prompted by the Cloudera AI Control Plane App while creating the workbench.
4. To use a subdirectory in an existing NFS share, say nfs_server:/export, do the following:
   a. Create a subdirectory /export/workbench1
   b. Change ownership: chown 8536:8536 /export/workbench1
   c. Set GID and make directory group writeable: chmod g+srwx /export/workbench1
   d. Provide the export path nfs_server:/export/workbench1 when prompted by the Cloudera AI Control Plane App.

Set up minimum permissions

The minimum permissions for Cloudera AI on Azure govern access control between the Cloudera AI Workbench, Azure resources, and the Azure storage account.

About this task

To set up the minimum permissions, you first create a custom role that contains those permissions, and then assign that role to a credential, called an app registration, in your Azure subscription.

Procedure

Create the custom role.

The following code example creates a custom role for Cloudera AI and assigns the minimum permissions needed. The permissions are listed in the actions section, so that Cloudera AI can access resources and operate correctly.

You need to substitute the following values:

- [YOUR-SUBSCRIPTION-ID]: Your subscription ID in use.
- [YOUR-RESTRICTED-ROLE-NAME]: The custom role name which is assigned to the application.
- [YOUR-SINGLE-RESOURCE-GROUP]: The original resource group name.

```
{
   "properties": {
       "roleName": [YOUR-RESTRICTED-ROLE-NAME],
       "description": "Custom restricted role for liftie",
       "isCustom": true,
       "assignableScopes": [
           "/subscriptions/[YOUR-SUBSCRIPTION-ID]/resourceGroups/[YOUR-SIN
GLE-RESOURCE-GROUP]"
       ],
       "permissions": [
           {
               "actions": [
                   "Microsoft.ContainerService/managedClusters/read",
                   "Microsoft.ContainerService/managedClusters/write",
                   "Microsoft.ContainerService/managedClusters/agentPools/r
ead",
                   "Microsoft.ContainerService/managedClusters/agentPools/
write",
                   "Microsoft.ContainerService/managedClusters/upgradeProfil
es/read",
   "Microsoft.ContainerService/managedClusters/agentPools/delete",
                   "Microsoft.ContainerService/managedClusters/delete",
                   "Microsoft.ContainerService/managedClusters/accessProfil
es/listCredential/action",
                   "Microsoft.ContainerService/managedClusters/agentPools/
upgradeProfiles/read",
                   "Microsoft.Storage/storageAccounts/read",
                   "Microsoft.Storage/storageAccounts/write",
                   "Microsoft.ManagedIdentity/userAssignedIdentities/assign/
action",
                   "Microsoft.Compute/virtualMachineScaleSets/write",
                   "Microsoft.Network/virtualNetworks/subnets/join/action",
                   "Microsoft.Network/virtualNetworks/subnets/read",
                   "Microsoft.Network/routeTables/read",
                   "Microsoft.Network/routeTables/write",
                   "Microsoft.Network/routeTables/routes/read",
                   "Microsoft.Network/routeTables/routes/write",
                   "Microsoft.Insights/diagnosticSettings/write",
                   "Microsoft.Insights/metrics/read",
                   "Microsoft.Insights/metricDefinitions/read",
                   "Microsoft.ManagedIdentity/userAssignedIdentities/fede
ratedIdentityCredentials/*"
               ],
               "notActions": [],
               "dataActions": [],
               "notDataActions": []
           }
       ]
   }
}
```

Related Information

Prerequisites for the provisioning credential

Migrating from generic NFS to Azure Files NFS in Cloudera AI

Cloudera AI enables you to migrate from generic NFS solutions (for example NetApp files) to Azure Files NFS for better performance and tighter integration. Migrating to the Azure ecosystem also allows Cloudera AI to provide the same full fledged backup-restore functionality for Azure, as it does for AWS.

1. Set up a blank Azure file NFS instance.

   You would need an empty Azure files volume to boot with. The size of the volume should be equal to or greater than the current NFS size in production use with your workbench. As with Azure files, the IOPS and network bandwidth increases with the capacity being provisioned. For more information on how to provision Azure files, see *Network Planning for Cloudera AI on Azure*.
2. Ensure that the workbench is in a steady state by scaling the web pods to 0. This will ensure that the workbench APIs become unavailable, and no read/writes are taking place in the workbench during copying of data.

   a. Download the kubeconfig from the Cloudera AI control pane, open a terminal session and set the environment variable “KUBECONFIG” to the path where the kubeconfig is downloaded by export KUBECONFIG=<path to kubeconfig> .

   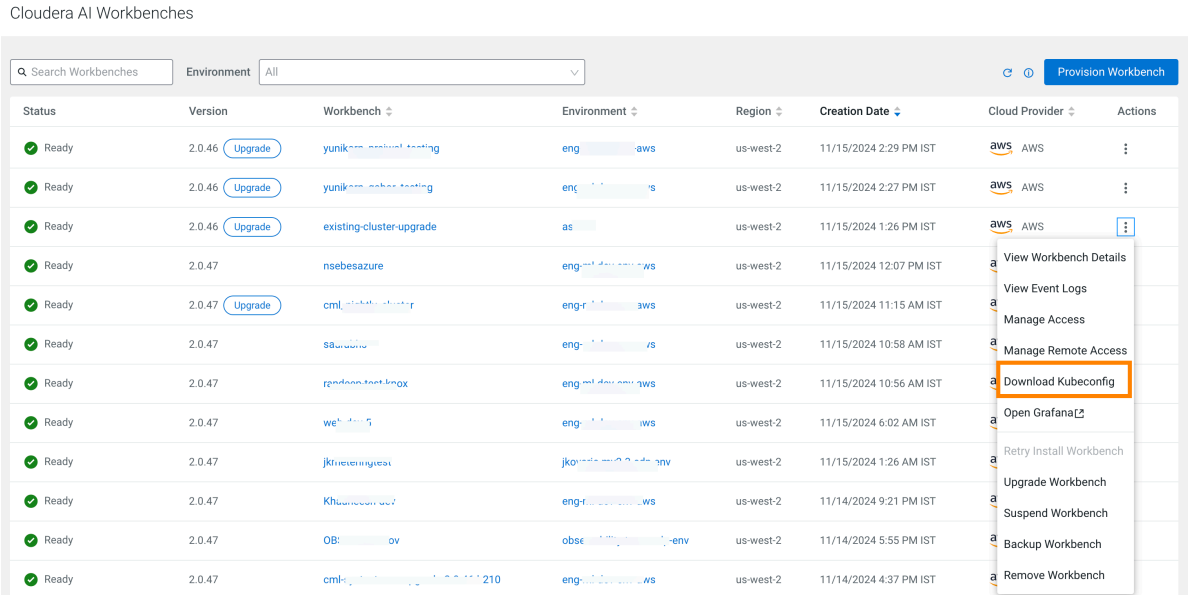

   b. To scale the web pods to 0, enter the following command.

   ```
   kubectl scale deployment/web -n mlx --replicas=0
   ```

   c. Retrieve the definition for your current NFS Persistent volume (PV) by entering the following command:

   ```
   kubectl get pv projects-share -o yaml
   ```
3. From the yaml file, note the values marked in bold in the following example:

   ```
   $ kubectl get pv projects-share -o yaml
   apiVersion: v1
   kind: PersistentVolume
   metadata:
     annotations:
   ```

```
    meta.helm.sh/release-name: mlx-mlx
    meta.helm.sh/release-namespace: mlx
    pv.kubernetes.io/bound-by-controller: "yes"
  creationTimestamp: "2022-05-05T09:36:59Z"
  finalizers:
  - kubernetes.io/pv-protection
  labels:
    app.kubernetes.io/managed-by: Helm
  name: projects-share
  resourceVersion: "5319"
  uid: ab4ff905-2c21-4cf6-9c31-6bd91ddc4fc8
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 50Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: projects-pvc
    namespace: mlx
    resourceVersion: "4747"
    uid: 2f58761d-2b13-4eb8-80eb-f2b97c77407f
  mountOptions:
  - nfsvers=3   #<oldNFSVersion>(this can be either 3 or 4.1)
  nfs:
  path: /eng-ml-nfs-azure/test-fs  # <oldNFSPath>
          server:10.102.47.132                  # <oldNFSServer>
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem
```

4. Note the server and path values from Azure files that are provisioned. For example, if the Azure Files NFS path is azurenfsv4.file.core.windows.net:/azurenfsv4/test-fs then:

   - AzureFileNFSServer = azurenfsv4.file.core.windows.net
   - AzureFileNFSPath = /azurenfsv4/test-fs
   - AzureFileNFSVersion = 4.1 (This is the only option Azure files offers)

   Ideally, you want a pod on which you can mount both NFS directories in a single pod, and copy over the contents. So you need to create PVs and PVCs.

5. Create a persistent volume (PV) by creating a yaml file and populating it with the following information: Replace the fields marked in bold with the variables defined in the previous step.

```
# nfs-migration-pv.yaml (the file name/location doesn't matter)
apiVersion: v1
kind: PersistentVolume
metadata:
  name: source-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 50Gi
  mountOptions:
  - nfsvers=<oldNFSVersion>
  nfs:
  path: <oldNFSPath>
    server: <oldNFSServer>
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem

---
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: destination-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  mountOptions:
    - nfsvers=4.1
  nfs:
  Server: <AzureFileNFSServer>
    path: <AzureFileNFSPath>
  volumeMode: Filesystem
```

6. Create a persistent volume claim (PVC) by creating a yaml file and populating it with the information in the following example.

```
#  nfs-migration-pvc.yaml


apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: source-pvc
  namespace: mlx
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: ""
  volumeMode: Filesystem
  volumeName: source-pv

---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: destination-pvc
  namespace: mlx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: ""
  volumeName: destination-pv
  volumeMode: Filesystem
```

7. Apply the kubernetes specs using the following commands:

```
$ kubectl apply -f  nfs-migration-pv.yaml
$ kubectl apply -f  nfs-migration-pvc.yaml
```

8. Create a helper pod to mount the PVCs created in the previous step and copy data from one NFS workbench to another.

   a. Create a yaml file and populate it with the following information.

```
# nfs-migration-helper-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: static-pod
  namespace: mlx
spec:
  imagePullSecrets:
    - name: jfrog-dev
  restartPolicy: Never
  volumes:
    - name: source-claim
      persistentVolumeClaim:
        claimName: source-pvc
    - name: destination-claim
      persistentVolumeClaim:
        claimName: destination-pvc
  containers:
    - name: static-container
    image: container.repository.cloudera.com/cloudera/cdsw/cdsw-ubi-m
inimal:<current Cloudera AI version>
      command:
        - "/bin/sh"
        - "-c"
      args:
        - "trap : TERM INT; sleep 9999999999d & wait"
      volumeMounts:
        - mountPath: /source
          name: source-claim
        - mountPath: /destination
          name: destination-claim
```

   b. Apply the kubernetes spec using the following command:

```
$ kubectl apply -f nfs-migration-helper-pod.yaml
```

9. After deploying the pod, wait for the pod to be in “running” status, then copy the shell into the pod using the following command:

```
kubectl exec -it -n mlx static-pod — sh .
```

10. Use rsync to copy the contents from the source (Generic NFS) to the destination (Azure files). Because the image does not come packaged with rsync, install it using the following command:

```
yum install -y rsync
```

After installing rsync, copy the contents by using the following command:

```
rsync -a -r -p --info=progress2 source/ destination
```

After the contents are copied, you can take a backup of the original workbench. Note that the backup only backs up the Azure disks (block volumes), therefore you had to back up the NFS manually.

Related Information

Network Planning for Cloudera AI on Azure

Backing up workbench

The Azure Back Up workbench feature is under entitlement. To enable Back Up workbench feature for Azure, you must use the entitlement ML_ENABLE_BACKUP_RESTORE. If you have the entitlement, the “Backup Catalog” is visible in the UI. Contact your account manager if you do not have the required entitlement.

Before you begin

Azure Back-Up has the following prerequisites:

- You must have an Azure resource group “cml-snapshots-<azure region>” preconfigured. Otherwise, the backup operation will throw preflight validation errors. For example, if your Cloudera AI Workbench is in the ‘westus2’ region, the resource group should be named “cml-snapshots-westus2”.

Procedure

1. To create a backup of the Azure workbench, enter the following commands:

```
$  cdpcp-account-helper % cdp ml backup-workspace --workspace-crn
                    crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edb
b73d:workspace:8d7ea6ba-718e-45e9-9532-767f8ef0b4a4
                    --backup-name azure-test-backup --profile dev
```

2. Be sure to note the backup CRN that CDPCLI returns:

```
{
    "backupCrn": "crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73
d:workspace_backup:bfc57661-0eb6-4a59-bcb9-ef793740558a"
}
```

3. Wait for the backup to be in a ready state, before proceeding further

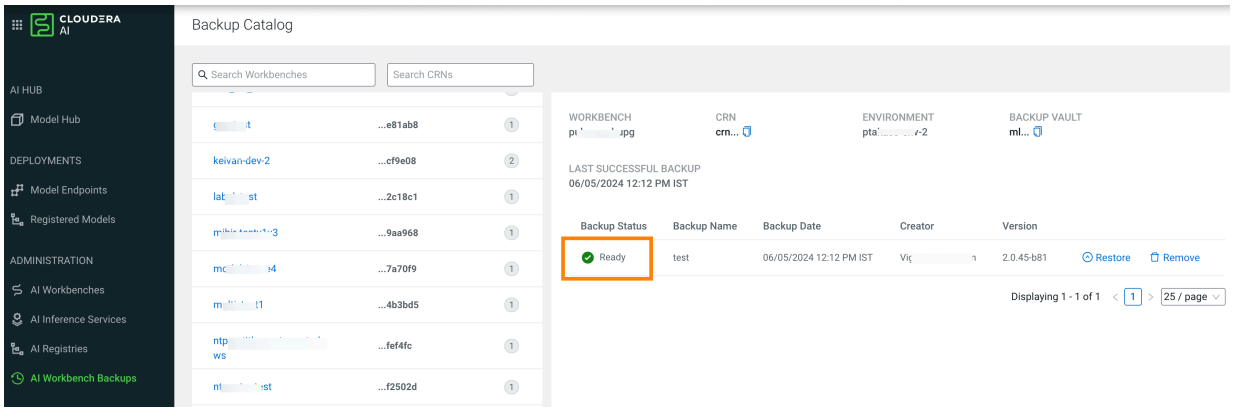

4. Bring up the original number of replicas (for example, 3) in order to make the original workbench functional.

```
kubectl scale deployment/web -n mlx --replicas=3
```

Results

At this point, you have the corresponding Azure disks and NFS backed up to be restored into a new workbench.

Restoring old data to a new workbench

You can restore all of the backed up data into a new workbench using the backup CRN. Use your previously defined Azure files NFS as the external NFS being supplied for workbench creation.

Procedure

1. Use the sample CLI command provided below to restore the backup. Change the field information to match with your own requirements. In particular, you must define ‘existingNFS’, ‘nfsVersion’ and ‘backupCrn’ with your specific information.

```
cdpcp-account-helper % cdp ml restore-workspace --cli-input-json '{
  "newWorkspaceParameters": {
    "environmentName": "eng-ml-dev-env-azure",
    "workspaceName": "workspace-with-migrated-nsf",
    "disableTLS": false,
    "usePublicLoadBalancer": false,
    "enableMonitoring": true,
    "enableGovernance": true,
    "enableModelMetrics": true,
    "whitelistAuthorizedIPRanges": false,
    "existingNFS": "nfs://azurenfsv4.file.core.windows.net:/azurenfsv4/t
est-fs",
    "nfsVersion": "4.1",
    "provisionK8sRequest": {
      "instanceGroups": [
        {
          "instanceType": "Standard_DS3_v2",
          "rootVolume": {
            "size": 128
          },
          "autoscaling": {
            "minInstances": 1,
            "maxInstances": 10
          }
        }
      ],
      "environmentName": "eng-ml-dev-env-azure",
      "tags": [],
      "network": {
        "topology": {
          "subnets": []
        }
      }
    },
    "skipValidation": true
  },
  "backupCrn": "crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73
d:workspace_backup:97711308-3014-418e-88d7-1f44cca495c7",
  "useStaticSubdomain": false
}' --profile dev
```

The CLI returns the following output:

```
{
    "workspaceCrn": "crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edb
b73d:workspace:e5520051-2e5f-406d-a003-a1e9dabbe1c4"
```

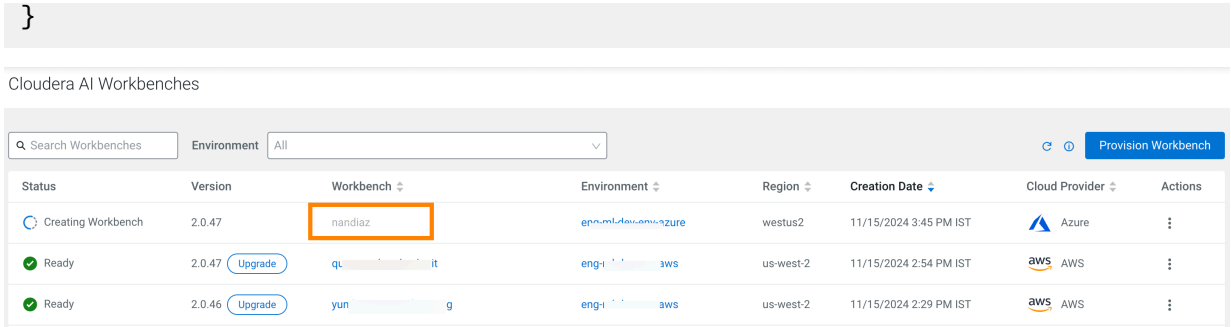

You can see the new workbench spinning up in the UI.

2. Once the workbench is in ready state, log into it. You should be able to see all the previous projects in the same state that you left off.

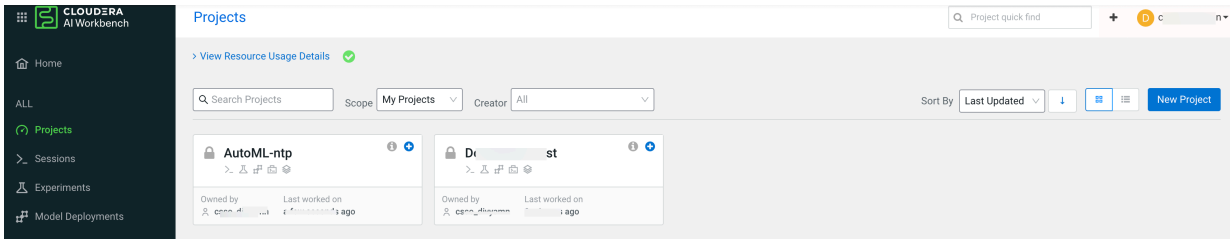

3. Try going into one of the projects and see if the files are still visible and accessible.

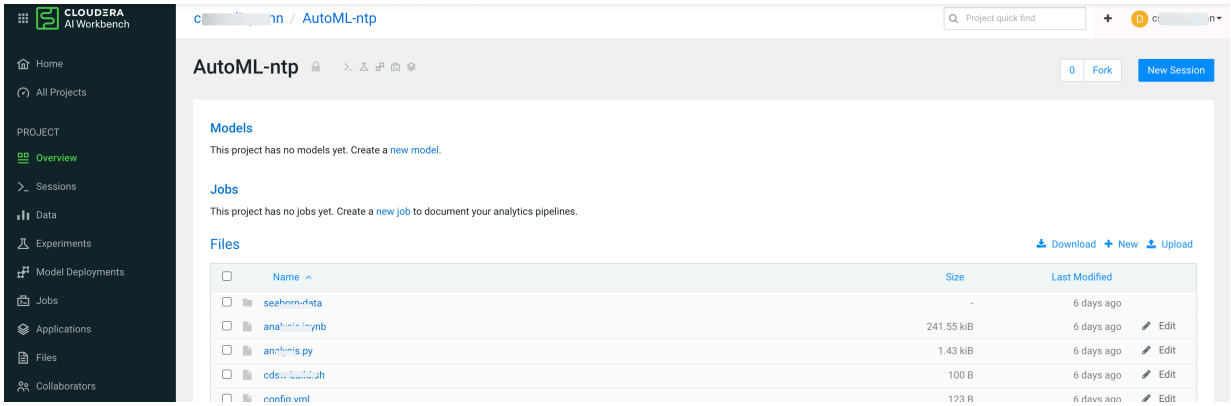

Cloudera AI

ML Runtimes

Date published: 2020-07-16
Date modified: 2025-01-29

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing ML Runtimes.....5
Adding new ML Runtimes.....5
Adding Custom ML Runtimes through the Runtime Catalog.....5
Adding ML Runtimes using Runtime Repo files.....5
ML Runtimes versus Legacy Engine.....7
Using Runtime Catalog.....8
Changing Docker credential setting for ML Runtime.....9
Disabling and Deleting Runtimes.....9
Enabling Runtimes.....10
Deleting Runtimes.....10

PBJ Workbench.....10
Dockerfile compatible with PBJ Workbench.....12
PBJ Runtimes and Models.....14
Example models with PBJ Runtimes.....15

Using ML Runtimes Addons.....16
Adding Hadoop CLI to ML Runtime Sessions.....16
Adding Spark to ML Runtime Sessions.....16
Turning off ML Runtimes Addons.....17

ML Runtimes NVIDIA GPU Edition.....17
Testing ML Runtime GPU Setup.....18

ML Runtimes NVIDIA RAPIDS Edition.....18

Using Editors for ML Runtimes.....20
Using JupyterLab with ML Runtimes.....20
Installing a Jupyter extension.....20
Installing a Jupyter kernel.....21
Using Conda Runtime.....21

Installing Additional ML Runtimes Packages.....22

Restrictions for upgrading R and Python packages.....23

Custom Runtime Addons with Cloudera AI.....24

ML Runtimes Environment Variables.....27
ML Runtimes Environment Variables List.....27

Accessing Environmental Variables from Projects....29

Customized Runtimes....29

Creating Customized ML Runtimes....30

Create a Dockerfile for the Custom Runtime Image....30

Metadata for Custom ML Runtimes....31

Editor Customization....32

Build the New Docker Image....32

Distribute the Image....32

Adding a new customized ML Runtime through the Runtime Catalog....32

Limitations....33

Adding Docker registry credentials and certificates....33

Pre-Installed Packages in ML Runtimes....35

Managing ML Runtimes

Provides overview, installation, set up, configuration, and customization information for Machine Learning Runtimes.

ML Runtimes are responsible for running the code written by users and intermediating access to the Data Hub.

You can think of an ML Runtime as a virtual machine, customized to have all the necessary dependencies to access the computing cluster while keeping each project's environment entirely isolated. To ensure that every ML Runtime has access to the parcels and client configuration managed by the Cloudera Manager Agent, a number of folders are mounted from the host into the container environment.

ML Runtimes have been open sourced and are available in the cloudera/ml-runtimes GitHub repository. If you need to understand your Runtime environments fully or want to build a new Runtime from scratch, you can access the Dockerfiles that were used to build the ML Runtime container images in this repository.

Adding new ML Runtimes

Cloudera AI provides two ways to add new Runtimes to the Runtime Catalog.

Note: You must have system administrator permission to add a new ML Runtime. Alternatively, in Site Administration Settings Access Control , an admin can enable Allow users to register ML Runtimes. If enabled, the names of users who register ML Runtimes is displayed on the Runtime Catalog page. Users are not permitted to deprecate or disable the added runtimes.

1. Adding Custom ML Runtimes through the Runtime Catalog

   Create your own Custom ML Runtime and add it through the Runtime Catalog
2. Adding ML Runtimes using Runtime Repo files

   Host a file from where Cloudera AI will automatically pull in new ML Runtimes and add them to the Runtime Catalog.

When adding Runtimes to Cloudera AI from password protected repositories, you need to add the necessary Docker credentials to Cloudera AI. See more on the *Add Docker registry credentials* page.

Related Information

Adding Docker registry credentials and certificates

Adding Custom ML Runtimes through the Runtime Catalog

Note: You must have system administrator permission to add a new ML Runtime.

When you open the Runtime Catalog as an administrator, you can add a new Custom Runtime with the "Add Runtime" button. You can read more about building a Custom Runtime image and adding to the catalog in the topic Creating Customized ML Runtimes.

Note, that you can not add Cloudera created Runtimes through the Runtime Catalog.

Related Information

Creating Customized ML Runtimes

Adding ML Runtimes using Runtime Repo files

Runtime Repo files are JSON files that contain all the details of ML Runtimes that are needed by Cloudera AI to add these ML Runtimes to the Runtime Catalog. When these files are hosted on URLs accessible to Cloudera AI, these URLs can be registered in Cloudera AI. If Enable Runtime Updates is selected in Site Administration Runtime , ML Runtimes that appear in the Runtime Repo files will be automatically added to the Runtime Catalog within 24 hours. Additionally, you can click Update Runtimes now to immediately update the Runtime Catalog. Using Runtime Repo files, both Custom and Cloudera provided Runtimes can be added to Cloudera AI Workbenches.

To add, edit, or remove Runtime Repo files

1. Log in as administrator.
2. Navigate to Site Administration Runtime .
3. Add, edit or delete Runtime Repo files in the Runtime Updates section.

Cloudera provided Runtime Repo files

Cloudera AI uses Runtime Repo files to automatically add new Runtimes to Cloudera AI Workbench when Runtime updates are enabled on the workbench. Cloudera hosts its own Runtime Repo files that always contain the details of the latest released ML Runtimes and Cloudera Data Visualization Runtimes. By default, you can find these Cloudera hosted Runtime Repos registered in your Cloudera AI Workbench:

Name: Cloudera ML Runtimes

URL: https://archive.cloudera.com/ml-runtimes/latest/artifacts/repo-assembly.json

Name: Cloudera DataViz Runtime

URL: https://archive.cloudera.com/cdv/latest/artifacts/repo-assembly.json

Self created Runtime Repo files

You can create your own Runtime Repo files and register them in Cloudera AI. Cloudera AI checks these Runtime Repo files for changes every 24 hours and adds any new ML Runtimes found in these files automatically to the Runtime Catalog.

To create a Repo assembly file:

1. Create a JSON file with the same structure as the Cloudera provided one:

```
{
    "assembly_metadata_version": 1,
    "runtimes": [
         {
           "image_identifier": string,
            "runtime_metadata_version": int,
            "editor": string,
            "edition": string,
            "description": string,
            "kernel": string,
            "full_version": string,
            "short_version": string,
            "maintenance_version": int,
            "git_hash": string,
            "gbn": int
         } ,
    ]
}
```

2. Fill in the details of one or more ML Runtimes. If you are adding Cloudera-created Runtimes, use the values from the Cloudera-provided Runtime Repo files. For Custom Runtimes, git_hash should be an empty string, and gbn should be set to zero. All other fields should be filled according to the information in *Metadata for Custom ML Runtimes*.
3. Host the JSON file on an URL that Cloudera AI is able to access.
4. Add the Runtime Repo file to Cloudera AI on the Site Administration Runtime page.

Related Information

Metadata for Custom ML Runtimes

ML Runtimes versus Legacy Engine

While Runtimes and the Legacy Engine are both container images that contain the Linux OS, interpreter(s), and libraries, ML Runtimes keeps the images small and improves performance, maintenance, and security.

Note: Legacy Engines are deprecated since June 2021 and will be removed in a future release. Starting with version 2.0.38, on new workbenches Legacy Engines are disabled by default and no Legacy Engine image is registered in the workbench. Cloudera recommends using ML Runtimes for all new projects, and urges customers to migrate existing Engine-based projects to ML Runtimes.

Runtimes and the Legacy Engine serve the same basic goal: they are container images that contain a complete Linux OS, interpreter(s), and libraries. They are the environment in which your code runs. However, ML Runtimes design keeps the images small, which improves performance, maintenance, and security.

There is one Legacy Engine. The Engine is monolithic. It contains the machinery necessary to run sessions using all four Engine interpreter options that Cloudera currently supports (Python 2, Python 3, R, and Scala) and a much larger set of UNIX tools including LaTeX. The Conda package manager was available in the Legacy Engine. Conda is not available in ML Runtimes.

Runtimes are the future of Cloudera AI. There are many Runtimes. Currently each Runtime contains a single interpreter (for example, Python 3.8, R 4.0) and a set of UNIX tools including `gcc`. Each Runtime supports a single UI for running code (for example, the Workbench or JupyterLab).

To migrate from Legacy Engine to Runtimes, you'll need to modify your project settings. See *Modifying Project Settings* for more information.

Disable Engines

Starting with version 2.0.38, Legacy Engines are disabled by default on the workbench level. This setting is accessible in Site Administration Runtime Disable Engines . Select this checkbox on upgraded workbenches to disable Legacy Engines and change all existing project types to ML Runtimes. Disabling Legacy Engines prohibits changing the project type to Legacy Engine.

This has the following effect for workloads using Legacy Engine kernels:

- Model builds cannot be redeployed
- Applications cannot be restarted
- Jobs cannot be started, including any kind of scheduled Jobs

To re-enable Legacy Engines go to Site Administration Runtime and deselect the Disable Engines checkbox under Engine Images.

Jupyter

Our Python Runtimes support JupyterLab, a general purpose IDE from the Jupyter project. The engine supports Jupyter Notebook, a simpler UI focused on Notebooks. If you prefer the simpler Notebook UI, choose Classic Notebook from the JupyterLab Help menu. To further customize the JupyterLab experience on Cloudera AI see *Using Editors for ML Runtimes*.

Build dependencies

Runtimes generally include fewer UNIX tools than the Legacy Engine. This means you are more likely to find that you cannot install a Python or R package because the Runtime is missing a build dependency such as a library. This should not happen often with Python. Most Python packages are distributed as precompiled “wheels”, so there are no build dependencies. It is more likely to happen with R packages because precompiled packages are not available for our architecture. We have tried to cover most common use cases, but if you find you cannot build something, then please contact customer support.

Using pip to install libraries in Python

To install a Python library from within Workbench or JupyterLab we recommend you use %pip (for example, %pip install sklearn. %pip is a “magic” command that is guaranteed to point to the right version of pip. This is a good habit to get into, as it will work outside Cloudera AI. Note you do not need to add “3” to install a Python 3 library.

If you prefer to use the pip executable directly, both pip and pip3 work. This is because Runtimes do not include Python 2. Like any shell command, precede it with “!” to run it from within Workbench or JupyterLab (for example, !pip install sklearn. In the Legacy Engine you must use pip3 to install Python 3 packages and the %pip magic command is not supported.

Python paths

Python Runtimes include preinstalled Python packages at /usr/local/lib/python/<version>/site-packages. The pre-installed packages and versions are documented in *Pre-Installed Packages in ML Runtimes*.

When you use pip, you install packages into the current project (not a runtime image) at /home/cdsw/.local/lib/python/<version>/site-packages. This means you need to reinstall packages if you change Python versions.

In most cases, you can install a newer version of a package preinstalled in /usr/local into your project. For example, we preinstall numpy and you can install a newer version. But there are some exceptions to this: if you install matplotlib, ipykernel, or its dependencies (ipython, traitlets, jupyter_client, and tornado) then you may break your ability to launch sessions.

If you accidentally install these packages (or you see unexpected behavior when you switch a project from Legacy Engine to Runtimes), the simplest solution is to delete /home/cdsw/.local/lib/python and reinstall your project’s dependencies from the project overview page.

R paths

R Runtimes include preinstalled R packages at /usr/local/lib/R/library/. The pre-installed packages and versions are documented in *Pre-Installed Packages in ML Runtimes*.

When you use install.packages(), you install packages into the current project (not a runtime image) at /home/cdsw/.local/lib/R/<version>/library (for example, $R_LIBS_USER). This means you need to reinstall packages if you change R versions.

Note the R project package path in Legacy Engines. If you use engines, you install packages to /home/cdsw/R. The change to /home/cdsw/.local/lib/R/<version>/library was made to support multiple versions of R.

In most cases, you can install a newer version of a package preinstalled /usr/local into your project. For example, we preinstall ggplot2 and you can install a newer version. But there are two exceptions to this. If you install Cairo or RServe they may break your ability to launch sessions.

If you accidentally install these packages (or you see unexpected behavior when you switch a project from Legacy Engine to Runtimes), the simplest solution is to delete /home/cdsw/.local/lib/python and reinstall your project’s dependencies from the project overview page.

Related Information

Creating a Project with Legacy Engine Variants

Modifying Project Settings

Using Runtime Catalog

You can use the Runtime catalog to list all runtimes that are available for your deployment. A Recommended label is added to some runtime variants recommended for use by Cloudera, so the adminstrator users can easily identify the optimal selections. Administrator users can set runtime variants as default runtime variants using the checkbox.

About this task

The Runtime Catalog lists information about each of the available runtimes. Information includes the editor and kernel supported by the runtime along with the edition, version, and a brief description.

Administrator users can use the Hide Disabled toggle so that only runtimes with Enabled or Deprecated status display. The toggle is on by default.

The Status filter also applies to runtime variants.

Procedure

Click Runtime Catalog in the left Navigation bar to display a list of all available runtimes.

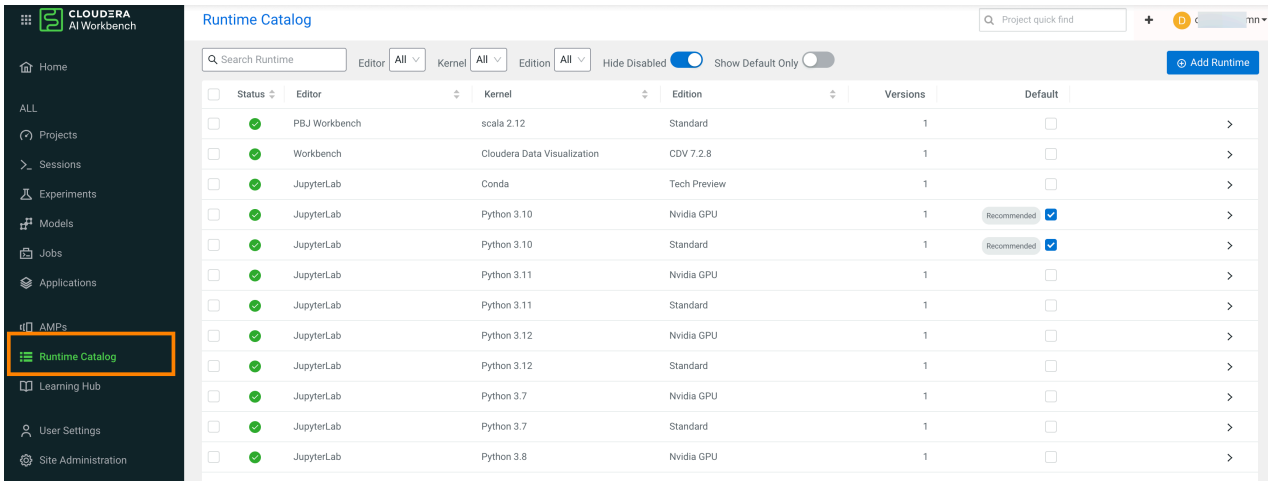

Changing Docker credential setting for ML Runtime

You can choose a specific Docker credential for Cloudera AI to use while fetching ML Runtime from a secure repository.

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click on the name of the workbench.

   The **Workbenches** Home page displays.
3. Click Runtime Catalog in the left navigation menu to display all available runtimes.
4. Select Actions > Add Credential.

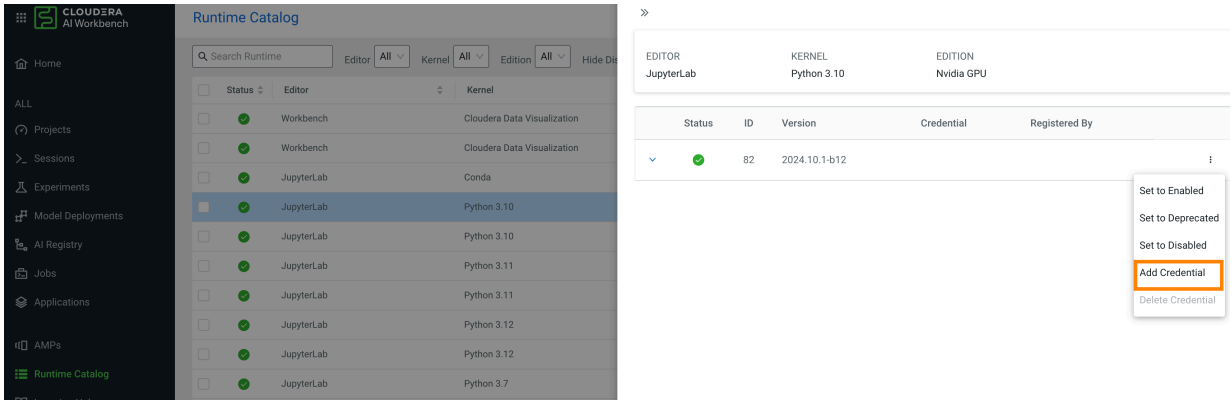

5. In the **Add Credential** window, select the Docker credential you want to use.
6. Click Add.

Disabling and Deleting Runtimes

A key feature of Cloudera Machine Learning, Runtime Catalog, is the ability to disable and enable one or more runtimes at once.

About this task

You can disable one or many runtimes at once.

Procedure

1. Select Runtime Catalog in the left-side Navigation to display all available runtimes.
2. Select one or more Runtimes.
3. Select Actions.
4. Select Set to Disabled.

Enabling Runtimes

A key feature of Cloudera Machine Learning, Runtime Catalog, is the ability to enable and disable one or more runtimes at once.

About this task

You can enable one or many runtimes at once.

Procedure

1. Select Runtime Catalog in the left-side Navigation to display all available runtimes.
2. Select one or more Runtimes.
3. Select Actions.
4. Select Set to Enabled.

Deleting Runtimes

Runtimes cannot be deleted from the Runtime Catalog. Instead, follow the steps to disable a runtime, and then ensure that the Hide Disabled toggle is set, so that disabled runtimes are not visible.

PBJ Workbench

The PBJ Workbench features the classic workbench UI backed by the open-source Jupyter protocol pre-packaged in a runtime image. Users can easily choose this runtime image when launching a session. The open-source Jupyter infrastructure eliminates the dependency on proprietary Cloudera AI code when building a Docker image, allowing runtime images to be built more quickly. The PBJ Workbench enables you to construct runtime images on Ubuntu base images (including non-Cloudera base images) and use them with the Cloudera AI Workbench.

ML Runtimes have been open sourced and are available in the cloudera/ml-runtimes GitHub repository. If you need to understand your Runtime environments fully or want to build a new Runtime from scratch, you can access the Dockerfiles that were used to build the ML Runtime container images in this repository.

The PBJ Workbench is available by default, but you have to select it when you launch a session.

1. Click New Session.
2. In Runtime Editor , select PBJ Workbench
3. Click Start Session.

Now you can use the PBJ Workbench as you would the normal workbench.

The requirements and preparatory steps for creating a PBJ Workbench are described in each section below.

PBJ Workbench setup: Python installation

PBJ Runtimes need to have Python installed, even if the Runtime is intended to run another kernel in Cloudera AI, e.g. R. The minimum Python version supported is 3.7. Python can be installed from the base image's package manager or compiled by the user.

The minimal requirements that must be satisfied by a custom PBJ Runtime image include:

- The actual Python binary or a symlink to it must exist at the following path: /usr/local/bin/python3
- The Python binary must be on the PATH under the name "python", meaning that executing the command "python" in a terminal shall start up Python.
- Executing "python --version" should result in a version greater than 3.7
- If the Runtime is configured to run a Python kernel in Cloudera AI, the commands "python" and "/usr/local/bin/python3" must start up the same Python process that is registered as a Jupyter Kernel (see below).

If the method you chose to install Python does not place the Python binary under /usr/local/bin/python3, or does not create the command "python", please create appropriate symlinks.

Install Jupyter dependencies and register your kernel

First, the Jupyter Kernel Gateway, version 2.5.2 must be installed into the Docker image. This example command may need to be modified depending on the filename and path of the pip executable in the image.

```
RUN pip3 install "jupyter-kernel-gateway==2.5.2"
```

The path to the Jupyter executable installed by pip should be noted, and the command to run Jupyter Kernel Gateway must be incorporated into the ML_RUNTIME_JUPYTER_KERNEL_GATEWAY_CMD environment variable in the Docker image:

```
ENV ML_RUNTIME_JUPYTER_KERNEL_GATEWAY_CMD="/path/to/jupyter kernelgateway"
```

Note: The highest supported version of the jupyter-client library is version 7.4.9. Do not upgrade to version 8 or higher.

When launching the Runtime in Cloudera AI, the correct IP address - port configuration for Jupyter Kernel Gateway is set automatically by Cloudera AI.

Next, a Jupyter kernel has to be registered. Each instance of the PBJ Workbench communicates with the Jupyter kernel installed in the Runtime via the Jupyter protocol. Kernels are available for a wide variety of languages and versions. Install the kernel of your choice to the image by following its installation instructions. A kernel named pyth on3 is registered by default when installing jupyter-kernel-gateway via pip. Installed Jupyter kernels can be listed by running the following command in a container created from the image:

```
path/to/jupyter kernelspec list
```

The name of your chosen kernel must be incorporated into the ML_RUNTIME_JUPYTER_KERNEL_NAME environment variable in the Docker image. For example, if your kernel's name is python3, the following must be included in the Dockerfile:

```
ENV ML_RUNTIME_JUPYTER_KERNEL_NAME=python3
```

Add the cdsw user

User code will be run in the image under user and group 8536:8536 . Associate these id's with the name cdsw in the image by adding the following command to the dockerfile:

```
RUN groupadd --gid 8536 cdsw && \
    useradd -c "CDSW User" --uid 8536 -g cdsw -m -s /bin/bash cdsw
```

Relax permissions so that Cloudera client config can be written

All code in the runtime container, including initial setup, will be run as the cdsw user. The initial setup includes linking client files for Cloudera data services out to their standard paths. To facilitate this, permissions to the following paths should be set so that user 8536 can write to them and to their subfolders:

- /etc
- /bin
- /usr/share/java
- /opt
- /usr

Also set the permissions of the following folders and all their subfolders to 777.

- /etc
- /etc/alternatives

Additional requirements

- ML_RUNTIME_METADATA_VERSION environment variable and the corresponding Docker label must be set to 2.
- To use the PBJ Workbench editor, the ML_RUNTIME_EDITOR environment variable and the corresponding Docker label must be set to "PBJ Workbench". If using a 3rd party editor (for example, JupyterLab or RStudio), set the ML_RUNTIME_EDITOR environment variable and Docker label to the desired value. Note that "Workbench" and "PBJ Workbench" are reserved names.
- Base image must be Ubuntu.
- Bash must be installed and must be configured as the default terminal used by the csdw user.
- In case the PBJ Runtime is running the R kernel, the kernel must be registered with the IRkernel package.
- The executable that is registered as a Jupyter kernel must be on PATH, must be found by the "which" command and must be named after the programming language of the kernel. E.g. the name of the executable must be:
  - `python` in case of a Python kernel (python3 is not sufficient)
  - `R` in case of an R kernel, etc.
- In the case of using a virtual / conda environment and a Python kernel, we recommend configuring PATH such that the default "pip" command corresponds to the python executable registered as a Jupyter kernel.
- Cloudera AI mounts the project’s filesystem under the path /home/cdsw and erases any file installed to that path in the Runtime image. Therefore, custom Runtime images should not install anything under the home folder of the cdsw user.
- On the other hand, once the Runtime image starts up in Cloudera AI, the kernel must be configured to install new packages to user site libraries under /home/cdsw. That way, newly installed packages will persist in the project’s filesystem.
- The package xz-utils must be installed on the Runtime image.
- The following binaries should be reachable on the PATH: `kinit`, `klist`, `ktutil`, and `sshd`. These are installed on Ubuntu as part of the following packages: `krb5-user` and `ssh`.

Dockerfile compatible with PBJ Workbench

This Dockerfile produces a runtime image that is compatible with the PBJ Workbench from a third-party base image.

Note: Apple silicon users should change the first line in the script to the following:

FROM --platform=linux/amd64 ubuntu:22.04

```
FROM ubuntu:22.04
USER root
```

```
# Install Python
# Note that the package python-is-python3 will alias python3 as python
RUN apt-get update && apt-get install -y --no-install-recommends \
   krb5-user python3.10 python3-pip python-is-python3 ssh xz-utils

# Configure pip to install packages under /usr/local
# when building the Runtime image
RUN pip3 config set install.user false

# Install the Jupyter kernel gateway.
# The IPython kernel is automatically installed
# under the name python3,
# so below we set the kernel name to python3.
RUN pip3 install "jupyter-kernel-gateway==2.5.2"

# Associate uid and gid 8536 with username cdsw
RUN \
  addgroup --gid 8536 cdsw && \
  adduser --disabled-password --gecos "CDSW User" --uid 8536 --gid 8536 cdsw

# Set up Python symlink to /usr/local/bin/python3
RUN ln -s $(which python) /usr/local/bin/python3

# Install any additional packages.
# apt-get install ...
# pip install ...

# configure pip to install packages to /home/cdsw
# once the Runtime image is loaded into Cloudera AI
# do not install Python packages in the Dockerfile after this line
RUN /bin/bash -c "echo -e '[install]\nuser = true'" > /etc/pip.conf

# Relax permissions to facilitate installation of Cloudera
# client files at startup
RUN for i in /bin /opt /usr /usr/share/java; do \
   mkdir -p ${i}; \
   chown cdsw ${i}; \
   chmod +rw ${i}; \
   for subfolder in `find ${i} -type d` ; do \
      chown cdsw ${subfolder}; \
      chmod +rw ${subfolder}; \
   done \
 done

RUN for i in /etc /etc/alternatives; do \
  mkdir -p ${i}; \
  chmod 777 ${i}; \
  done


# Set Runtime label and environment variables metadata
#ML_RUNTIME_EDITOR and ML_RUNTIME_METADATA_VERSION must not be changed.
ENV ML_RUNTIME_EDITOR="PBJ Workbench" \
    ML_RUNTIME_METADATA_VERSION="2" \
    ML_RUNTIME_KERNEL="Python 3.10" \
    ML_RUNTIME_EDITION="Custom Edition" \
    ML_RUNTIME_SHORT_VERSION="1.0" \
    ML_RUNTIME_MAINTENANCE_VERSION="1" \
    ML_RUNTIME_JUPYTER_KERNEL_GATEWAY_CMD="/usr/local/bin/jupyter kernelgate
way" \
    ML_RUNTIME_JUPYTER_KERNEL_NAME="python3" \
    ML_RUNTIME_DESCRIPTION="My first Custom PBJ Runtime"
```

```
ENV ML_RUNTIME_FULL_VERSION="$ML_RUNTIME_SHORT_VERSION.$ML_RUNTIME_MAINTE
NANCE_VERSION"
LABEL com.cloudera.ml.runtime.editor=$ML_RUNTIME_EDITOR \
      com.cloudera.ml.runtime.kernel=$ML_RUNTIME_KERNEL \
      com.cloudera.ml.runtime.edition=$ML_RUNTIME_EDITION \
      com.cloudera.ml.runtime.full-version=$ML_RUNTIME_FULL_VERSION \
      com.cloudera.ml.runtime.short-version=$ML_RUNTIME_SHORT_VERSION \
      com.cloudera.ml.runtime.maintenance-version=$ML_RUNTIME_MAINTENANCE
_VERSION \
      com.cloudera.ml.runtime.description=$ML_RUNTIME_DESCRIPTION \
      com.cloudera.ml.runtime.runtime-metadata-version=$ML_RUNTIME_METADA
TA_VERSION
```

PBJ Runtimes and Models

The PBJ (Powered by Jupyter) Runtime enables a wide variety of language kernels to be run as Cloudera AI workloads. Model workloads are currently only supported for Python and R kernels.

A new library, cml, is added to Runtime based Python and R workloads automatically and includes mostly the same functionality as the old cdsw library.

In non-PBJ Runtimes, you could point to a Python or R function and run it as a model. PBJ Runtimes open up the possibilities for users to create their own Runtimes and places few restrictions on how to do that. PBJ technology utilizes the Jupyter kernelgateway to communicate with language kernels and aims to be language independent in the future.

Migrating from the cdsw to Cloudera AI library

PBJ Runtimes currently includes two supported languages, Python and R. There are new function wrappers or decorators, described below, that must be used in order to allow previously written model code to function in a PBJ Runtime. It is important to note that these support decorators or wrappers are transparent, that is, they have no effect, in non-PBJ workloads and non-model workloads.

Table 1: Python names

CDSW	Cloudera AI
call_model	models_v1.call_model
-	models_v1.cml_model
get_auth	utils_v1.get_auth
track_file	Removed
model_metrics	Replaced by models_v1.cml_model(metrics=True)
read_metrics	metrics_v1.read_metrics
track_metric	metrics_v1.track_metric
track_delayed_metrics	metrics_v1.track_delayed_metrics
track_aggregate_metrics	metrics_v1.track_aggregate_metrics
launch_workers	workers_v1.launch_workers
list_workers	workers_v1.list_workers
stop_workers	workers_v1.stop_workers
await_workers	workers_v1.await_workers

Table 2: R names

CDSW	Cloudera AI
html	Planned for future release
iframe	Planned for future release
markdown	Planned for future release
display.help	Planned for future release
code	Planned for future release
image	Planned for future release
get.auth	get.auth
launch.workers	launch.workers
stop.workers	stop.workers
list.workers	list.workers
await.workers	await.workers
track.metric	Removed
track.file	Removed
-	cml_model

Library availability

In Cloudera AI, the cdsw and s libraries are available in different runtime and engine types.

Runtime / Engine type	Library availablility
Legacy Engine	cdsw
Classic Runtime	cdsw, cml
PBJ Runtime	cml

Note: The cml library is now used by default, instead of the cdsw library. However, if a project was created on a previous Cloudera AI release, it might use the deprecated cdsw library. Code using the cdsw library will not run on a PBJ Runtime. Code running on PBJ Runtimes must use the cml library.

Example models with PBJ Runtimes

The library cml includes the package, models_v1. This package includes the cml_model decorator that can be used to allow a function to work as a model in a PBJ Runtime. It can also be used to enable gathering of model metrics.

The package has an optional boolean argument, called metrics, that when set to True, enables the model_metric decorator functionality from the deprecated cdsw lib. The default value of the metrics parameter is False. Therefore, this example does not have model metric gathering enabled, but the R example does. In both examples the function to be used for the model is called predict.

Python example

Example #1

```
import cml.models_v1 as models

@models.cml_model
def predict(args):
    return args["x"]*2
```

Example #2:

```
import cml.models_v1 as models

@models.cml_model(metrics=True)
def predict(args):
    return args["x"]*2
```

R example

Almost all of the functionality in the cdsw library for R has also been migrated to the cml library and is available automatically for every R Runtime workload. It includes a new function wrapper, called cml_model, to be used for model entry point functions in PBJ Runtimes. The function to be used for the model in this case is called predict.

```
library(cml)

predict <- cml_model(function(args) {
    return(args$x*2)
    })
```

Using ML Runtimes Addons

ML Runtime Addons allow you to add Spark and Hadoop CLI to sessions run on projects using ML Runtime images.

While Legacy Engines include support for both Spark and Hadoop CLI, ML Runtimes do not contain Spark and Hadoop CLI binaries to keep them small and lean. Instead Spark and Hadoop binaries are stored in persistent storage on an NFS server and can be added to your ML Runtime sessions.

Adding Hadoop CLI to ML Runtime Sessions

Hadoop CLI can be enabled only on sessions that are selected to use Spark.

About this task

To add Hadoop CLI to sessions run on projects using ML Runtimes images:

Procedure

1. You can view all available ML Runtime addons by selecting the Site Administration>Runtime/Engine tab and viewing Runtime Addons.
2. To include Hadoop in all sessions created in the workbench, under Site Administration>Runtime/Engine, choose the desired Hadoop version in the pull down menu next to Hadoop CLI Version.

   Note: Hadoop CLI can be enabled only on sessions that are selected to use Spark.

   This will add Hadoop CLI to all sessions created in the workbench.
3. To use Hadoop commands for your session, click the Terminal Access button at the top of the Session window.

   Cloudera AI launches a terminal window in which you can use Hadoop commands.

Adding Spark to ML Runtime Sessions

You can add Spark to ML Runtime Sessions using the ML Runtimes Addons. Both Spark and Hadoop CLI are enabled when you enable Spark.

About this task

To add Spark to sessions run on projects using ML Runtimes images:

Procedure

1. You can view all available ML Runtime addons by selecting the Site Administration>Runtime/Engine tab and viewing Runtime Addons.
2. Start a New Session for an ML Runtimes project.
3. Click the Enable Spark option, then select the Spark version.
4. Click Start Session.
5. You can now run a Spark job for your session.

   The Logs tab displays the executors in your Spark job.
6. After you start a Spark job, you can access the Spark UI by clicking the Spark UI button at the top of the Session window.

Turning off ML Runtimes Addons

ML Runtimes Addons is turned on by default. However, you can disable ML Runtimes Addons.

Procedure

1. Select Site Administration > Settings.
2. Under Feature Flags, uncheck the checkbox next to Allow users to Run ML Runtimes Addons.

ML Runtimes NVIDIA GPU Edition

The NVIDIA GPU Edition Runtimes are built on top of NVIDIA CUDA docker images. CUDA is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers can dramatically speed up computing applications by harnessing the power of GPUs.

Cloudera Machine Learning supports GPUs using particular ML Runtime Editions. NVIDIA GPUs together with related specialized software can be utilized using the NVIDIA GPU Edition.

Version information:

Version	Additional software	Tested OS	Tested Driver Version
2021.09	CUDA 11.4	RHEL 7.6	NVIDIA driver 460.56

Compatibility information:

Runtime Version	CUDA Version	Kernels	Editors	Base Image
2021.02	CUDA 11.1	Python 3.7 and Python 3.8	Workbench and JupyterLab	

ML Runtimes inherit the compatibility requirements for NVIDIA CUDA. For compatibility information, visit the following pages:

- NVIDIA/CUDA compatibility matrix
- Tensorflow requirements
- Pytorch requirements

Testing ML Runtime GPU Setup

You can use the following simple examples to test whether the new ML Runtime is able to leverage GPUs as expected.

1. Go to a project that is using the ML Runtimes NVIDIA GPU edition and click Open Workbench.
2. Launch a new session with GPUs.
3. Run the following command in the workbench command prompt to verify that the driver was installed correctly:

```
! /usr/bin/nvidia-smi
```

4. Use any of the following code samples to confirm that the new engine works with common deep learning libraries.

Pytorch

```
!pip3 install torch==1.4.0
from torch import cuda
assert cuda.is_available()
assert cuda.device_count() > 0
print(cuda.get_device_name(cuda.current_device()))
```

Tensorflow

```
!pip3 install tensorflow-gpu==2.1.0
from tensorflow.python.client import device_lib
assert 'GPU' in str(device_lib.list_local_devices())
device_lib.list_local_devices()
```

Keras

```
!pip3 install keras
from keras import backend
assert len(backend.tensorflow_backend._get_available_gpus()) > 0
print(backend.tensorflow_backend._get_available_gpus())
```

ML Runtimes NVIDIA RAPIDS Edition

The RAPIDS Edition Runtimes are built on top of community built RAPIDS docker images. The RAPIDS suite of software libraries gives you the freedom to execute end-to-end data science and analytics pipelines entirely on GPUs. It relies on NVIDIA CUDA primitives for low-level compute optimization, but exposes that GPU parallelism and high-bandwidth memory speed through user-friendly Python interfaces.

Note: RAPIDS require NVIDIA Pascal or better GPUs. You need to use P3 or newer EC2 instances on AWS to meet this requirement.

Visit rapids.ai for more information.

Note: RAPIDS on Spark is not support as Spark3 is not supported on CDSW.

ML Runtimes RAPIDS edition differs from other Runtime Editions in the following ways:

- Python maintenance versions differ from what is being used in the Standard and NVIDIA GPU edition runtimes. These Python kernels are coming from the RAPIDS base image.

- Pre-installed Python packages and package versions differ from what's in Standard and NVIDIA GPU edition runtimes.

RAPIDS Runtimes are an optional extension of the Cloudera ML Runtime distribution. The RAPIDS images are not distributed automatically. Administrators can register them in the Runtime Catalog. The following RAPIDS editions are available for the 2021.04 Runtime version:

RAPIDS and CUDA Version	Kernel	Editor	Base Image	Docker
RAPIDS 0.18 CUDA 11.0	Python 3.7	Workbench editor	rapidsai/rapidsai-core:0.18-cuda11.0-base-ubuntu20.04-py3.7	docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.7-rapids:2021.12.1-b17
RAPIDS 0.18 CUDA 11.0	Python 3.8	Workbench editor	rapidsai/rapidsai-core:0.18-cuda11.0-base-ubuntu20.04-py3.8	docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.8-rapids:2021.12.1-b17
RAPIDS 0.18 CUDA 11.0	Python 3.7	JupyterLab editor	rapidsai/rapidsai-core:0.18-cuda11.0-base-ubuntu20.04-py3.7 Note: This image is not based on NVIDIA's JupyterLab installation so RAPIDS library examples are not installed.	docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-jupyterlab-python3.7-rapids:2021.12.1-b17
RAPIDS 0.18 CUDA 11.0	Python 3.8	JupyterLab editor	rapidsai/rapidsai-core:0.18-cuda11.0-base-ubuntu20.04-py3.8 Note: This image is not based on NVIDIA's JupyterLab installation so RAPIDS library examples are not installed.	docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-jupyterlab-python3.8-rapids:2021.12.1-b17

RAPIDS Python environment

The RAPIDS python libraries are distributed and present as a conda environment in the docker image. While it's possible to install additional libraries for your project via conda, these packages will be installed to a non-persistent part of the file system and won't persist between sessions and jobs.

Installing additional libraries via pip is supported and works the same way as in other Runtimes.

Note: The RAPIDS images have the official Anaconda channels configured for conda, if you decide to use conda to install packages, make sure you are familiar with Anaconda's license requirements.

Related Information

Testing ML Runtime GPU Setup

Using Editors for ML Runtimes

Cloudera Machine Learning provides a Workbench UI to edit and run code, but we also provide a preconfigured JupyterLab runtime to allow this. Choose the editor you prefer when launching a ML Runtime session.

Using JupyterLab with ML Runtimes

JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data.

You can use JupyterLab to configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab allows you to use extensions that add new components and integrate with existing ones.

You must install any files necessary for JupyterLab, including configuration, customization, extensions, and kernels, into /home/cdsw for each project.

Code Completion in JupyterLab

You can use JupyterLab to write and run code in notebooks or in a traditional .py file. Completion works out of the box in notebooks and consoles. If you are working in a .py file in the text editor, you must create a console to provide the completion suggestions. To do this:

1. Open the .py file.
2. Right click on the document and choose "Create console for editor".
3. Choose the option appropriate to the type of completion you want to perform.

   For example, to get completions on "math", you must first "import math" in the console.

Set JupyterLab Timeouts

You can control JupyterLab timeouts with the following environment variables:

```
JUPYTER_KERNEL_TIMEOUT_SECONDS
JUPYTER_SERVER_TIMEOUT_SECONDS
```

Installing a Jupyter extension

Extensions modify the appearance or behaviour of Jupyter applications (including JupyterLab and Jupyter Notebook).

Before you begin

You must install any files necessary for JupyterLab, including configuration, customization, extensions, and kernels, into /home/cdsw for each project.

Note: Any Jupyter Extension that is not specifically designed for JupyterLab should work fine with Jupyter Notebook, which is preinstalled in Legacy Engine engine:14.

About this task

The following example installs the ipython-sql extension. This extension adds a %sql magic command that allows you to communicate directly with any database supported by SQLAlchemy.

Procedure

1. Launch a JupyterLab session.

2. Open a terminal and run the following command:

```
pip install ipython-sql
```

Note: The previous command is only an example for installing ipython-sql.

3. Open a notebook or console and run the following command:

```
%load_ext sql
```

Results

You can now use the %sql magic command to connect to a particular database, and the %%sql magic command to create a cell containing an SQL query to run against that database. See the JupyterLab documentation for additional information on Jupyter magic commands.

Installing a Jupyter kernel

Jupyter kernels are “connections” through which a notebook (or other part of JupyterLab) can talk to a particular interpreter. CDSW includes one kernel in each JupyterLab Runtime: Python 3.6, Python 3.7, or Python 3.8.

Jupyter kernels are “connections” through which a notebook (or other part of JupyterLab) can talk to an interpreter. ML Runtimes come with a Jupyter kernel for Python 3.6, 3.7 or 3.8 preinstalled, but you may also install Jupyter kernels for bash or JavaScript/TypeScript.

bash

The following example describes how to enable a bash kernel in a particular project.

1. Launch a JupyterLab session.
2. Open a terminal window and run the following command:

```
pip install bash_kernel
python -m bash_kernel.install
```

You should now have the option to launch bash notebooks and consoles.

JavaScript and TypeScript

The following example describes how to enable JavaScript and TypeScript kernels in a particular project. .

1. Customize your PATH environment variable:
   a. Navigate to Project Settings/Advanced.
   b. Set PATH to $HOME/node_modules/.bin:$PATH.
2. Launch a JupyterLab session.
3. Open a terminal window and run the following command:

```
npm install tslab
tslab install
```

You should now have the option to launch JavaScript and TypeScript notebooks and console.

Using Conda Runtime

Users now can create their own Python or R Conda environments within their Cloudera AI Projects that they can use in the JupyterLab editor.

Environments, installed packages, and kernel specifications are persisted on the Project file system. An example flow for creating a new Python 3.10 Conda environment is the following:

```
conda create --name myenv python=3.10
conda activate myenv
conda install ipykernel
ipython kernel install --user --name=myenv
```

JupyterLab Conda Tech Preview Runtime

You might run into some known issues while using JupyterLab Conda Runtime.

Sessions

When starting a Notebook or a Console for a specific environment, the installed packages will be available and the interpreter used to evaluate the contents of the Notebook or Console will be the one installed in the environment. However, the Conda environment is not "activated" in these sessions, therefore commands like !which python will return with the base Python 3.10 interpreter on the Runtime. The recommended ways to modify a Conda environments or install packages are the following:

- conda commands must be used with the -n or --name argument to specify the environment, for example conda -n myenv install pandas
- When installing packages with pip, use the %pip magic to install packages in the active kernel's environment, for example %pip install pandas

Applications and Jobs

To start an Application or Job, first create a launcher Python script containing the following line: ! source activate <conda_env_name> && python <job / application script.py>

When starting the Application or Job, select the launcher script as the "Script".

Models

Models are currently not supported for the Conda Runtime.

Spark

Spark is not supported in JupyterLab Notebooks and Consoles.

Spark workloads are supported in activated Conda environments in JupyterLab Terminals, or in Jobs or Applications.

The CDSW libraries for Python and R are not available for the Conda Runtimes.

Installing Additional ML Runtimes Packages

ML Runtimes are preloaded with a few common packages and libraries for Python. However, a key feature of Cloudera AI is the ability of different projects to install and use libraries pinned to specific versions, just as you would on your local computer.

Before you begin

Before downloading or using third-party content, you are responsible for reviewing and complying with any applicable license terms and making sure that they are acceptable for your use case.

About this task

Generally, Cloudera recommends you install all required packages locally into your project. This will ensure you have the exact versions you want and that these libraries will not be upgraded when Cloudera upgrades the ML Runtimes image. You only need to install libraries and packages once per project. From then on, they are available to any new ML Runtimes you spawn throughout the lifetime of the project.

You can install additional libraries and packages from the workbench, using either the command prompt or the terminal.

Note:

Cloudera AI does not currently support installation of packages that require root access to the hosts. For such use-cases, you will need to create a new custom runtime that extends the base image with the required packages. For instructions, see Customized Runtimes on page 29.

About this task

(Python) Install Packages Using Workbench Command Prompt

To install a package from the command prompt:

1. Navigate to your project's Overview page. Click New Session and launch a session.
2. At the command prompt (see Native Workbench Console and Editor) in the bottom right, enter the command to install the package. Some examples using Python have been provided.

Python 3

```
# Installing from console using ! shell operator and pip3:
!pip3 install beautifulsoup4
# Installing from terminal
pip3 install beautifulsoup4
```

About this task

(Python Only) Using a Requirements File

For a Python project, you can specify a list of the packages you want in a requirements.txt file that lives in your project. The packages can be installed all at once using pip/pip3.

1. Create a new file called requirements.txt file within your project:

   ```
   beautifulsoup4==4.6.0
   seaborn==0.7.1
   ```

2. To install the packages in a Python 3 ML Runtimes, run the following command in the workbench command prompt.

   ```
   !pip3 install -r requirements.txt
   ```

Restrictions for upgrading R and Python packages

Some R and Python packages should not be upgraded because doing so will break the Workbench UI.

For R Runtimes

Do not upgrade the Cairo or RServe packages in your projects using R Runtimes. If you do, you may be unable to launch sessions, jobs, experiments, or applications.

If these packages are upgraded, you will need to start a new project or to delete the directory .local/lib using the CDSW/Cloudera AI Files UI ("show hidden").

For Python Runtimes

Do not upgrade the ipykernel package or its dependencies (ipython, traitlets, jupyter_client, and tornado) in your project using Python Runtimes. If you do, you may be unable to launch sessions, jobs, experiments, or applications.

If these packages are upgraded, you will need to start a new project or to delete the directory .local/lib using the CDSW/Cloudera AI Files UI ("show hidden").

Python Runtimes in Cloudera AI fail to import the setuptools Python library and can fail installing some Python packages

Python Runtimes in Cloudera AI fail to import the setuptools Python library and therefore can fail installing some Python packages when the library setuptools is present on the Runtime or is installed into the Cloudera AI project with version 60.0.0 or higher.

Python 3.10 Runtimes from the 2023.05 Runtime release ship with a newer version of setuptools, so customers can run into this issue when they are using that Runtime. Also they can run into this issue when they are using Custom Runtimes that has a newer setuptools library version or when they install a new setuptools version into their project (regardless of what Runtime they use).

Workaround: Set the environmental variable SETUPTOOLS_USE_DISTUTILS=stdlib either on a project level under Project Settings -> Advanced or on a workbench level under Site Administration -> Runtime -> Environment variables.

Custom Runtime Addons with Cloudera AI

Custom ML Runtimes enable you to create runtimes with your own choice of libraries and applications, but it is also possible to further customize existing ML runtimes with additional configuration files or binaries such as connection drivers, without the effort of creating a new custom ML runtime.

Warning: Using unsecured or unlicensed libraries can cause security threats. It is your responsibility to diligently add the custom ML Runtime addons. For more information, see *Cloudera's Shared Responsibility Model*.

How this feature works

Package your dependencies into a tarball file, following the specified format. A site administrator can then register the custom runtime addon by uploading the tarball via an API call on the command line.

After a custom runtime addon has been registered, the site administrator can enable or disable addons in the UI.

When an addon is enabled, it is mounted to all newly started runtime-based workloads, in read-only mode.

Creating the Custom Runtime Addon

1. You must create a structure of files and folders that follows the pattern shown here, and package it as a tar.gz archive. For example, to mount a folder called custom_folder, with all of its content under /usr/bin/custom_folder, and custom_file at /usr/custom_file, the internal structure of the tarball should look like this:

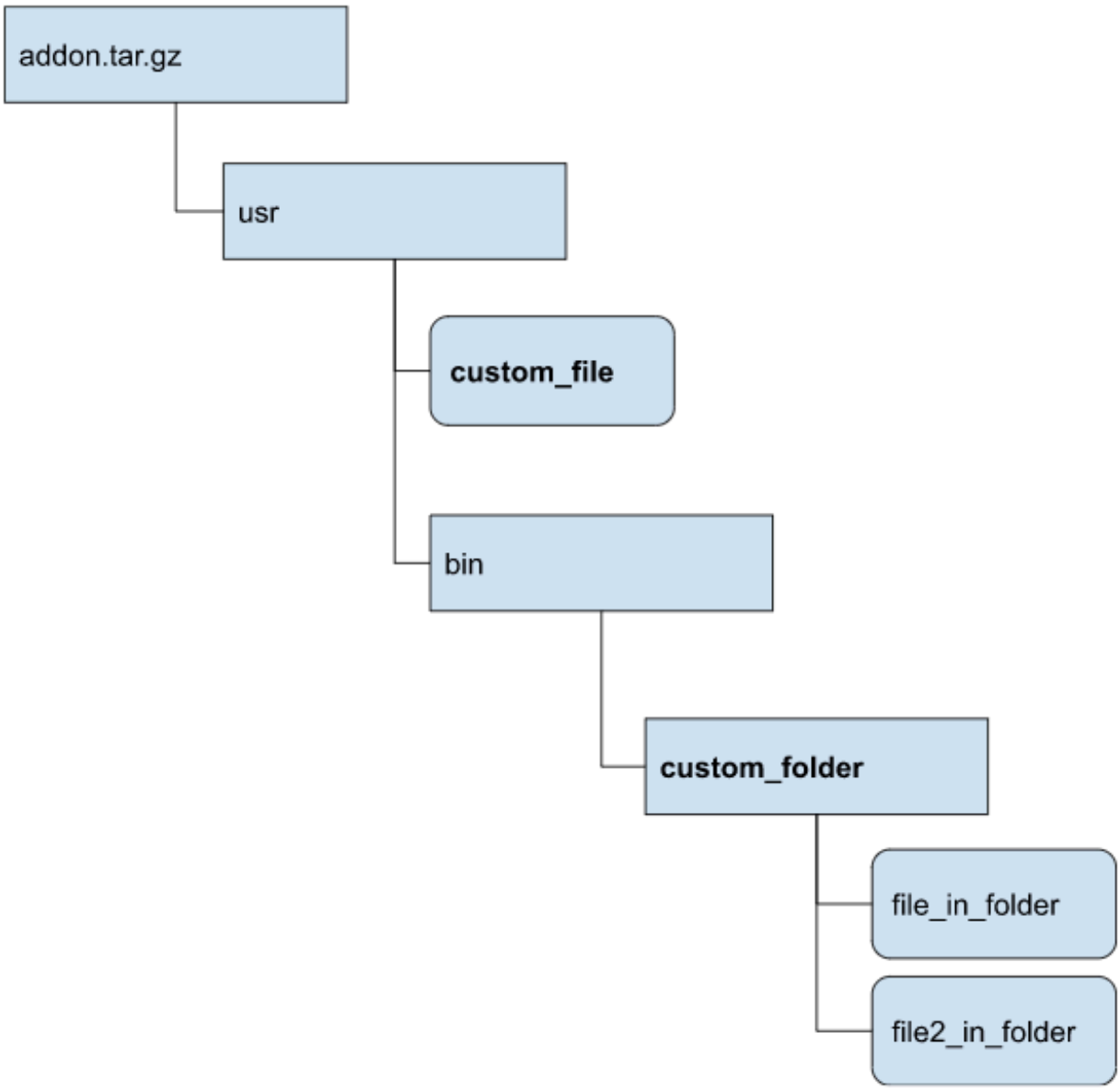

2. Next, you create a json file that describes the custom runtime addon to register in Cloudera AI. The json file should follow this structure:

```
{
    "name" : "name-of-the-addon",
    "spec" : {"paths" : ["/usr/custom_file", "/usr/bin/custom_folder"]}
}
```

3. Finally, the json file and tar.gz file are uploaded by a Site administrator. Use the following curl command, with these substitutions:
   - metadata.json: the name of the json file
   - addon.tar.gz: the name of the tar.gz file

```
curl -v -L '<cml_cluster>/api/v2/runtimeaddons/custom'  -F metadata=@met
adata.json
-F tarball=@addon.tar.gz  -H 'Authorization: Bearer <APIv2 key>'
```

   After the runtime add-on has been uploaded, it will be enabled and mounted to all newly started Runtime-based workloads.

Requirements

1. Name of the addon must contain only lowercase letters, numbers, underscores and hyphens. Also, it must be no longer than 35 characters. When added to Cloudera AI, the addon will be prefixed with the term "custom-addon-".
2. There should be at least one path mentioned in the uploaded json file.
3. Paths in the metadata file should not end with a forward slash (/) character.
4. Requested paths will be symlinked in the workloads's filesystem by the "cdsw" user, therefore these locations must not exist in the workload's file system and "cdsw" user must have write access to these locations.

Limitations

1. Injecting Python or R libraries into workloads via Custom Runtime Addons is not supported. To accomplish this, create a Custom Runtime instead. For more information, see *Creating Customized ML Runtimes*.
2. Misconfigured Custom Runtime Addons can break user workloads (e.g. if metadata refers to paths that do not exist in the uploaded tarball archive).
3. It is not possible to add Custom Runtime Addons through the UI.
4. Custom Runtime Addons can not be updated or removed, only disabled.
5. Custom Runtime Addons are mounted to workloads in read only mode.

Troubleshooting

When a Custom Runtime Addon tries to mount in a file or folder that falls into one of the below categories, workloads might fail to come up. In that case, Cloudera AI UI will show the error messages "Failed setting up runtime addons", also "Engine exited with status 33". The exact error message can help determine what scenario causes the issue.

Possible root causes Cloudera AI not being able to mount in files or folders from Custom Runtime Addons:

The file or folder does not exist in the uploaded tarball or is not located under the proper path within the tarball archive.

Solution: Adjust paths in the tarball/metadata file according to the Requirements.

The file or folder a Custom Runtime Addon tries to mount exists already in the filesystem. Either the file/folder exists on the Runtime image or is mounted in by another Runtime Addon.

Solution: Due to a limitation, Custom Runtime Addons can not overwrite existing files. Customers should change the path of the mounted files.

cdsw user has no write access in the pod's file system to create symlinks to mounted files.

Solution: Change the path of the mounted files to a location that is writable by the cdsw user. Alternatively, create a Custom Runtime image with adjusted permissions and use that in all workload

Related Information

Creating Customized ML Runtimes

Cloudera's Shared Responsibility Model

ML Runtimes Environment Variables

This topic describes how ML Runtimes environmental variables work. It also lists the different scopes at which they can be set and the order of precedence that will be followed in case of conflicts.

ML Runtimes environment variables behave the same way for Legacy Engines and ML Runtimes.

Environmental variables allow you to customize ML Runtimes environments for projects. For example, if you need to configure a particular timezone for a project, or increase the length of the session/job timeout windows, you can use environmental variables to do so.

Cloudera AI allows you to define environmental variables for the following scopes:

Global

A site administrator for your Cloudera AI deployment can set environmental variables on a global level. These values will apply to every project on the deployment.

To set global environmental variables, go to Admin Runtime/Engine .

Project

Project administrators can set project-specific environmental variables to customize the ML Runtimes launched for a project. Variables set here will override the global values set in the site administration panel.

To set environmental variables for a project, go to the project's Overview page and click Settings Advanced .

Job

Environments for individual jobs within a project can be customized while creating the job. Variables set per-job will override the project-level and global settings.

To set environmental variables for a job, go to the job's Overview page and click Settings Set Environmental Variables .

Experiments

ML Runtimes created for execution of experiments are completely isolated from the project. However, these ML Runtimes inherit values from environmental variables set at the project-level and/or global level. Variables set at the project-level will override the global values set in the site administration panel.

Models

Model environments are completely isolated from the project. Environmental variables for these ML Runtimes can be configured during the build stage of the model deployment process. Models will also inherit any environment variables set at the project and global level. However, variables set per-model build will override other settings.

ML Runtimes Environment Variables List

The following table lists Cloudera AI environment variables that you can use to customize your project environments. These can be set either as a site administrator or within the scope of a project or a job.

Environment Variable	Description
MAX_TEXT_LENGTH	Maximum number of characters that can be displayed in a single text cell. By default, this value is set to 800,000 and any more characters will be truncated. Default: 800,000

Environment Variable	Description
SESSION_MAXIMUM_MINUTES	Maximum number of minutes a session can run before it times out. Default: 60*24*7 minutes (7 days) Maximum Value: 35,000 minutes
JOB_MAXIMUM_MINUTES	Maximum number of minutes a job can run before it times out. Default: 60*24*7 minutes (7 days) Maximum Value: 35,000 minutes
IDLE_MAXIMUM_MINUTES	Maximum number of minutes a session can remain idle before it exits. Default: 60 minutes Maximum Value: 35,000 minutes Idle timeouts for sessions vary by workbench type (runtime). • Standard Workbench: Sessions timeout regardless of activity in the browser or terminal. • PBJ Workbench: Sessions timeout if there is no browser activity and no terminal window is open. If a terminal window is open, the session will not timeout, regardless of whether there is activity in the terminal window. • Jupyterlab: Sessions timeout if there is no browser activity. Terminal window activity is not considered. • Custom runtimes: No idle timeout behavior is enforced on custom or third-party workbenches.

Per-Engine Environmental Variables: In addition to the previous table, there are some more built-in environmental variables that are set by the Cloudera AI application itself and do not need to be modified by users. These variables are set per-engine launched by Cloudera AI and only apply within the scope of each engine.

Environment Variable	Description
CDSW_PROJECT	The project to which this engine belongs.
CDSW_ENGINE_ID	The ID of this engine. For sessions, this appears in your browser's URL bar.
CDSW_MASTER_ID	If this engine is a worker, this is the CDSW_ENGINE_ID of its master.
CDSW_MASTER_IP	If this engine is a worker, this is the IP address of its master.
CDSW_PUBLIC_PORT	**Note:** This property is deprecated. See CDSW_APP_PORT and CDSW_READONLY_PORT for alternatives. A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_PUBLIC_PORT will be available in browsers at: http(s)://<*$CDSW_ENGINE_ID*>.<*$CDSW_DOMAIN*>. By default, CDSW_PUBLIC_PORT is set to 8080. A direct link to these web services will be available from the grid icon in the upper right corner of the Cloudera AI web application, as long as the job or session is still running. For more details, see *Accessing Web User Interfaces from Cloudera AI*. In Cloudera AI, setting CDSW_PUBLIC_PORT to a non-default port number is not supported.
CDSW_APP_PORT	A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_APP_PORT will be available in browsers at: http(s)://<*$CDSW_ENGINE_ID*>.<*$CDSW_DOMAIN*>. Use this port for applications that grant some control to the project, such as access to the session or terminal. A direct link to these web services will be available from the grid icon in the upper right corner of the Cloudera AI web application as long as the job or session runs. Even if the web UI does not have authentication, only Contributors and those with more access to the project can access it. For more details, see *Accessing Web User Interfaces from Cloudera AI*. Note that if the Site Administrator has enabled Allow only session creators to run commands on active sessions, then the UI is only available to the session creator. Other users will not be able to access it. Use 127.0.0.1 as the IP.

Environment Variable	Description
CDSW_READONLY_PORT	A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_READONLY_PORT will be available in browsers at: http(s)://<*$CDSW_ENGINE_ID*>.<*$CDSW_DOMAIN*>. Use this port for applications that grant read-only access to project results. A direct link to these web services will be available to users with from the grid icon in the upper right corner of the Cloudera AI web application as long as the job or session runs. Even if the web UI does not have authentication, Viewers and those with more access to the project can access it. For more details, see *Accessing Web User Interfaces from Cloudera AI*. Use 127.0.0.1 as the IP.
CDSW_DOMAIN	The domain on which Cloudera AI is being served. This can be useful for iframing services, as demonstrated in *Accessing Web User Interfaces from Cloudera AI*.
CDSW_CPU_MILLICORES	The number of CPU cores allocated to this engine, expressed in thousandths of a core.
CDSW_MEMORY_MB	The number of megabytes of memory allocated to this engine.
CDSW_IP_ADDRESS	Other engines in the Cloudera AI cluster can contact this engine on this IP address.

Accessing Environmental Variables from Projects

This topic shows you how to access environmental variables from your code.

Environmental variables are injected into every engine launched for a project, contingent on the scope at which the variable was set (global, project, etc.). The following code samples show how to access a sample environment variable called *DATABASE_PASSWORD* from your project code.

Python

```
import os
database_password = os.environ["DATABASE_PASSWORD"]
```

Appending Values to Environment Variables:

You can also set environment variables to append to existing values instead of replacing them. For example, when setting the LD_LIBRARY_PATH variable, you can set the value to LD_LIBRARY_PATH:/path/to/set.

Customized Runtimes

This topic explains how custom Runtimes work and when they should be used.

Note: Cloudera, Inc. (“Cloudera”) makes the Custom Runtime feature available to allow its users to add, run, and manage their own container images for their workloads as ‘Custom Runtimes’. Cloudera’s support covers only the integration points of these images with Cloudera AI, to the extent adding, running, and managing Custom Runtime images fulfill the documented requirements and/or pre-requisites. Cloudera does not provide any support for any other configurations and/or third party software added or installed to the image by customers.

By default, ML Runtimes are preloaded with a few common packages and libraries for R, Python, and Scala. In addition to these, ML Runtimes also allows you to install any other packages or libraries that are required by your projects. However, directly installing a package to a project as described above might not always be feasible. For example, packages that require root access to be installed, or that must be installed to a path outside /home/cdsw (outside the project mount), cannot be installed directly from the workbench.

For such circumstances, Cloudera AI allows you to extend the base Docker image and create a new Docker image with all the libraries and packages you require. Site administrators can then add this new image in the allowlist for use in projects.

Note: You will need to remove any unnecessary Cloudera sources or repositories that are inaccessible because of the paywall.

PBJ Custom Runtimes can be built on top of any Ubuntu base image, and users have to install the kernel themselves. However, non-PBJ Runtime images can only be built on top of Cloudera-released non-PBJ Runtime images, and users cannot change the kernel.

Note that this approach can also be used to accelerate project setup across the deployment. For example, if you want multiple projects on your deployment to have access to some common dependencies (package or software or driver) out of the box, or even if a package just has a complicated setup, it might be easier to simply provide users with a Runtime that has already been customized for their project(s).

Related Resources

- The Cloudera Engineering Blog post on *Customizing Docker Images in Cloudera AI* describes an end-to-end example on how to build and publish a customized Docker image and use it as an engine in Cloudera AI.
- For an example of how to extend the base engine image to include Conda, see *Installing Additional Packages*.

Creating Customized ML Runtimes

This section walks you through the steps required to create your own custom ML Runtimes based on one of the Cloudera provided ML Runtime images.

Note: In case of PBJ Runtimes, the Custom runtime does not have to be built on the top of a Cloudera provided ML Runtime image.

Create a Dockerfile for the Custom Runtime Image

This topic shows you how to create a Dockerfile for a custom image.

First, select an appropriate source image for your customization. For a non-PBJ Runtime, you must use a Runtime image released by Cloudera. Image tags can be seen on the Session Start page on the user interface when you select a Runtime. The second step when building a customized image is to create a Dockerfile that specifies which packages you would like to install in addition to the base image.

For example, the following Dockerfile installs the telnet package, the sklearn Python package and upgraded base packages on top of an ML Runtime image released by Cloudera.

```
# Dockerfile
# Specify an ML Runtime base image
FROM docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-jupyterlab-pyth
on3.7-standard:2021.12.1-b17
# Install telnet in the new image
RUN apt-get update && apt-get install -y --no-install-recommends telnet &&
 apt-get clean && rm -rf /var/lib/apt/lists/*
# Upgrade packages in the base image
RUN apt-get update && apt-get upgrade -y && apt-get clean && rm -rf /var/lib
/apt/lists/*
# Install the python package sklearn
RUN pip install --no-cache-dir sklearn
# Override Runtime label and environment variables metadata
ENV ML_RUNTIME_EDITION="Telnet Edition" \
        ML_RUNTIME_SHORT_VERSION="1.0" \
        ML_RUNTIME_MAINTENANCE_VERSION=1 \
        ML_RUNTIME_DESCRIPTION="This runtime includes telnet and sklearn
and upgraded packages"
```

```
ENV ML_RUNTIME_FULL_VERSION="${ML_RUNTIME_SHORT_VERSION}.${ML_RUNTIME_MAI
NTENANCE_VERSION}"
LABEL com.cloudera.ml.runtime.edition=$ML_RUNTIME_EDITION \
        com.cloudera.ml.runtime.full-version=$ML_RUNTIME_FULL_VERSION \
        com.cloudera.ml.runtime.short-version=$ML_RUNTIME_SHORT_VERSION \
        com.cloudera.ml.runtime.maintenance-version=$ML_RUNTIME_MAINTENANCE
_VERSION \
        com.cloudera.ml.runtime.description=$ML_RUNTIME_DESCRIPTION
```

Metadata for Custom ML Runtimes

This topic addresses the metadata for custom Runtimes.

All new custom Runtimes must override the Edition metadata of existing Runtimes. The rest of the metadata can be overriden to communicate the expectations for the consumers of the image. Both the Docker label and the environment variable match in a custom Runtime image. In order to add or register a custom Runtime to a deployment, the user facing metadata combination should be unique in that deployment. For example, of the following, Editor, Edition, Kernel, Version, and Maintenance Version, at least the later should be incremented for adding a next iteration of the same image.

See the following reference table for more details on ML Runtime metadata.

Environment variable	Docker Label	Description	Override in custom non-PBJ runtime	Value in PBJ Runtimes
ML_RUNTIME_METADATA_VERSION	com.cloudera.ml.runtime.runtime-metadata-version	Metadata version	Not allowed	Must be set to 2
ML_RUNTIME_EDITOR	com.cloudera.ml.runtime.editor	CDSW/Cloudera AI Editor installed in the image.	Allowed	Required
ML_RUNTIME_EDITION	com.cloudera.ml.runtime.edition	Edition of the image, a notion of the Runtime capabilities.	Required	Required
ML_RUNTIME_DESCRIPTION	com.cloudera.ml.runtime.description	Longer description of the Runtime image capabilities.	Recommended	Required
ML_RUNTIME_KERNEL	com.cloudera.ml.runtime.kernel	Main kernel included in the image, e.g., Python 3.8	Not allowed	Required
ML_RUNTIME_SHORT_VERSION	com.cloudera.ml.runtime.short-version	Main version of the image, e.g., 1.0. This shows up as Version in the selection screen.	Recommended	Required
ML_RUNTIME_FULL_VERSION	com.cloudera.ml.runtime.full-version	Full version consists of the short version + maintenance version, e.g., 1.0.1	Optional	Optional
ML_RUNTIME_MAINTENANCE_VERSION	com.cloudera.ml.runtime.maintenance-version	Maintenance version must be an integer, e.g., 1. Only the largest maintenance version of the set of the same short version images are visible for users to select. Increment this number if you create a drop in replacement of an existing Runtime. Start with 1 with a new version or edition.	Recommended	Required
ML_RUNTIME_CUDA_VERSION	com.cloudera.ml.runtime.cuda-version	CUDA version installed in the image.	Not allowed	Do not set

ML_RUNTIME_GIT_HASH	com.cloudera.ml.runtime.git-hash	Git hash of runtime source	Not allowed	Do not set
ML_RUNTIME_GBN	com.cloudera.ml.runtime.gbn	Cloudera internal build number	Not allowed	Do not set

Editor Customization

This topic addresses customizing a third-party editor to work with ML Runtimes.

For a third-party editor to work with ML Runtimes, you must provide a starting script. For example:

```
#! /bin/bash
"$ZEPPELIN_HOME/bin/zeppelin.sh"
```

For Cloudera Machine Learning to interpret this as an editor startup script, you must create a symlink to the editor as/usr/local/bin/ml-runtime-editor. This will be created in the Dockerfile of the customized runtime.

Note: Third-party editors provide a way to run arbitrary code that is not distributed by Cloudera. Use third-party editors at your own risk. You should absolutely trust the code that you want to run.

Build the New Docker Image

This topic shows you how to use Docker to build a custom image.

A new custom Docker image can be built on any host where Docker binaries are installed, source images are available, and OS repositories and other package repositories are available. To install these binaries, run the following command on the host where you want to build the new image:

```
docker build -t <image-name>:<tag> . -f Dockerfile
```

If your Dockerfile makes any outside connection (for example, apt-get, update, pip install, curl), you must add the --network=host option to the build command:

```
docker build --network=host -t <image-name>:<tag> . -f Dockerfile
```

Distribute the Image

This topic explains the different methods that can be used to distribute a custom ML Runtime to all the hosts.

Once you have built a new custom ML Runtime, use one of the following methods to distribute the new image to all your Cloudera Machine Learning hosts:

Push the image to a public registry such as DockerHub

For instructions, refer to the Docker documentation *docker push* and *Push images to Docker Cloud*.

Push the image to your company's Docker registry

When using this method, make sure to tag your image with the following schema:

```
docker tag <image-name> <company-registry>/<user-name>/<image-name>:<tag>
```

Once the image has been tagged properly, use the following command to push the image:

```
docker push <company-registry>/<user-name>/<image-name>:<tag>
```

Adding a new customized ML Runtime through the Runtime Catalog

Cloudera AI enables you to add customized ML Runtimes from the Runtime Catalog window.

About this task

Note: You must have system administrator permission to add a new ML Runtime.

Note: If you add a Custom Runtime from a private docker registry, you need to add the docker credentials first to Cloudera AI. See *Add Docker registry credentials and certificates* for more information.

Procedure

1. Click Runtime Catalog from the Navigation panel.
2. Click the Add Runtime button in the upper right corner.
3. In the Add Runtime window, enter the url of the Runtime Docker image you want to upload.

   As ML Runtimes are identified based on certain attributes, metadata (such as Editor, Kernel, Edition, Version, and Maintenance Version) must be unique to add new Customized Runtimes to a deployment. Customized ML Runtimes must have different Edition text compared to Cloudera supported versions.
4. If you want to use a non-default credential, select a docker credential in the Select Credentials dropdown list.

   The Docker server name and the user name of the credential is displayed.
5. Click Validate.

   Cloudera AI will use the provided URL to fetch the Docker image and validate if it can be used as a customized Runtime.

   If the Docker image is successfully validated, Cloudera AI will display the metadata information of the image. The new customized Runtime will be visible in the Runtime Catalog and accessible over the different workloads.

Related Information

Adding Docker registry credentials and certificates

Limitations

This topic lists some limitations associated with customized ML Runtime images.

- The contents of certain pre-existing standard directories such as /home/cdsw, /tmp, and so on, cannot be modified while creating customized non-PBJ ML Runtimes. This means any files saved in these directories will not be accessible from sessions that are running on customized ML Runtimes.

  Workaround: Create a new custom directory in the Dockerfile used to create the customized ML Runtime, and save your files to that directory.

For PBJ Runtimes, note the following limitations:

- PBJ Runtimes work as models only with R and Python kernels.

Adding Docker registry credentials and certificates

To enable Cloudera AI to fetch custom ML Runtimes from a secure repository, as an Administrator you need to add Docker registry credentials. If you want to use different credentials for different runtimes, you can add more docker credentials using the UI and API v2 and use that credentials to fetch custom ML Runtimes.

1. In the **Cloudera** console, click the Cloudera AI tile.

   The **Cloudera AI Workbenches** page displays.
2. Click on the name of the workbench.

   The **Workbenches** Home page displays.

3. Click Site Administration in the left navigation menu.

   The Site Administration page displays.
4. Click Runtimes, and click Add Credentials in the Docker Credentials section.

   The Add Credentials dialog box is displayed.

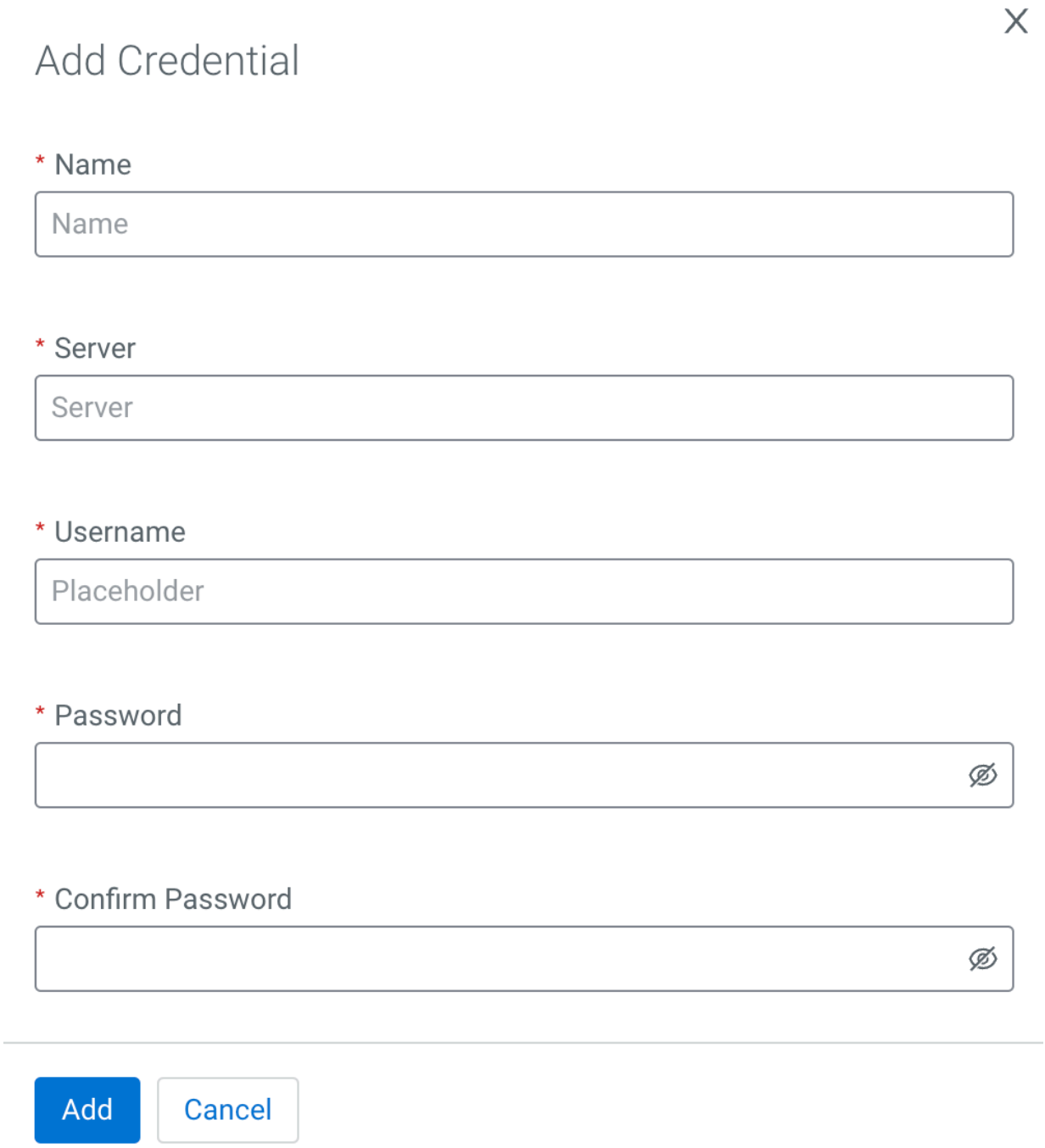

5. In Name, provide a name for the credential. This name will be displayed to manage the credentials.
6. In Server, provide the docker server address.
7. In Username, provide the username of the credential.
8. In Password, provide the password for the credential.
9. In Confirm Password, repeat the same password given in the Password field.
10. Click Add.

The credentials name and server name are displayed under the Docker Credentials section. The default credentials are used when ML Runtimes images are pulled, and non-default credentials are used when they are added to a Custom Runtime image in **Runtime Catalog**.

Click Add Credentials to add more docker credentials. You can use the toggle button to set the default credential. The default credential is picked up automatically for Custom Runtimes coming from the server stored in the default credential. To use your other credentials, add them to Custom Runtimes in Runtime Catalog.

Click to edit the Docker credential and click to delete the Docker credential.

Note: Refer the documentation of your chosen docker registry to understand what docker login credentials it expects. The expected credentials might be different from your own username and password. Many registries require docker authentication using tokens or API keys.

If the Custom Runtime is uploaded to AWS ECR, then the Docker password would come from your AWS account (aws ecr get-login-password --region <region>). You have to use this password when you add the credential to Cloudera AI.

You need to update these Docker credentials for Cloudera AI again if you back up and then restore your Cloudera AI Workbench.

Pre-Installed Packages in ML Runtimes

Cloudera AI ships with several base engine images that include Python kernels, and frequently used libraries.

Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

As the software and versions listed in *ML Runtimes Pre-installed Packages* are installed in the images, you shall not upgrade these versions as this may lead to dependency conflicts. If any related software in this list needs to be at a higher version, you must use a more recent version of the legacy engine or ML Runtime.

Related Information

ML Runtimes Pre-installed Packages

Cloudera AI

ML Runtimes Pre-installed Packages

Date published: 2020-07-16
Date modified: 2025-01-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ML Runtimes Pre-installed Packages overview **8**

ML Runtimes 2025.01.1 ..... **8**
Python 3.12 Libraries for Conda 8
Python 3.12 Libraries for Workbench 13
Python 3.11 Libraries for Workbench 19
Python 3.10 Libraries for Workbench 26
Python 3.9 Libraries for Workbench 32
Scala 2.12 Libraries for Workbench 38
Python 3.12 Libraries for JupyterLab 42
Python 3.11 Libraries for JupyterLab 51
Python 3.10 Libraries for JupyterLab 61
Python 3.9 Libraries for JupyterLab 71
R 4.4 Libraries 81

ML Runtimes 2024.10.1 ..... **86**
Python 3.10 Libraries for Conda 87
Python 3.12 Libraries for Workbench 91
Python 3.11 Libraries for Workbench 98
Python 3.10 Libraries for Workbench 104
Python 3.9 Libraries for Workbench 112
Python 3.8 Libraries for Workbench 122
Python 3.7 Libraries for Workbench 133
Scala 2.11 Libraries for Workbench 140
Python 3.12 Libraries for JupyterLab 140
Python 3.11 Libraries for JupyterLab 150
Python 3.10 Libraries for JupyterLab 159
Python 3.9 Libraries for JupyterLab 169
Python 3.8 Libraries for JupyterLab 179
Python 3.7 Libraries for JupyterLab 189
R 4.4 Libraries 195

ML Runtimes 2024.05.2 ..... **199**
Python 3.10 Libraries for Conda 199
Python 3.11 Libraries for Workbench 204
Python 3.10 Libraries for Workbench 210
Python 3.9 Libraries for Workbench 217
Python 3.8 Libraries for Workbench 226
Python 3.7 Libraries for Workbench 235
Scala 2.11 Libraries for Workbench 242
Python 3.11 Libraries for JupyterLab 242
Python 3.10 Libraries for JupyterLab 251
Python 3.9 Libraries for JupyterLab 260
Python 3.8 Libraries for JupyterLab 270
Python 3.7 Libraries for JupyterLab 279
R 4.4 Libraries 285

ML Runtimes 2024.05.1 290
Python 3.10 Libraries for Conda 290
Python 3.11 Libraries for Workbench 295
Python 3.10 Libraries for Workbench 300
Python 3.9 Libraries for Workbench 307
Python 3.8 Libraries for Workbench 316
Python 3.7 Libraries for Workbench 324
Scala 2.11 Libraries for Workbench 331
Python 3.11 Libraries for JupyterLab 331
Python 3.10 Libraries for JupyterLab 340
Python 3.9 Libraries for JupyterLab 350
Python 3.8 Libraries for JupyterLab 359
Python 3.7 Libraries for JupyterLab 369
R 4.4 Libraries 375

ML Runtimes 2024.02.1 377
Python 3.10 Libraries for Conda 377
Python 3.11 Libraries for Workbench 381
Python 3.10 Libraries for Workbench 386
Python 3.9 Libraries for Workbench 393
Python 3.8 Libraries for Workbench 401
Python 3.7 Libraries for Workbench 410
Scala 2.11 Libraries for Workbench 417
Python 3.11 Libraries for JupyterLab 417
Python 3.10 Libraries for JupyterLab 424
Python 3.9 Libraries for JupyterLab 431
Python 3.8 Libraries for JupyterLab 438
Python 3.7 Libraries for JupyterLab 445
R 4.3 Libraries 451

ML Runtimes 2023.12.1 455
Python 3.10 Libraries for Conda 455
Python 3.11 Libraries for Workbench 459
Python 3.10 Libraries for Workbench 464
Python 3.9 Libraries for Workbench 471
Python 3.8 Libraries for Workbench 479
Python 3.7 Libraries for Workbench 487
Scala 2.11 Libraries for Workbench 494
Python 3.11 Libraries for JupyterLab 494
Python 3.10 Libraries for JupyterLab 501
Python 3.9 Libraries for JupyterLab 507
Python 3.8 Libraries for JupyterLab 514
Python 3.7 Libraries for JupyterLab 522
R 4.3 Libraries 528

ML Runtimes 2023.08.2 532
Python 3.10 Libraries for Conda 532
Python 3.10 Libraries for Workbench 536
Python 3.9 Libraries for Workbench 542
Python 3.8 Libraries for Workbench 550
Python 3.7 Libraries for Workbench 558

Scala 2.11 Libraries for Workbench....564
Python 3.10 Libraries for JupyterLab....565
Python 3.9 Libraries for JupyterLab....571
Python 3.8 Libraries for JupyterLab....578
Python 3.7 Libraries for JupyterLab....585
R 4.3 Libraries....591
R 4.1 Libraries....595
R 4.0 Libraries....599
R 3.6 Libraries....603

ML Runtimes 2023.08....607
Python 3.10 Libraries for Conda....607
Python 3.10 Libraries for Workbench....610
Python 3.9 Libraries for Workbench....617
Python 3.8 Libraries for Workbench....625
Python 3.7 Libraries for Workbench....632
Scala 2.11 Libraries for Workbench....639
Python 3.10 Libraries for JupyterLab....640
Python 3.9 Libraries for JupyterLab....646
Python 3.8 Libraries for JupyterLab....653
Python 3.7 Libraries for JupyterLab....660
R 4.3 Libraries....666
R 4.1 Libraries....670
R 4.0 Libraries....674
R 3.6 Libraries....678

ML Runtimes 2023.05....682
Python 3.10 Libraries for Conda....682
Python 3.10 Libraries for Workbench....685
Python 3.9 Libraries for Workbench....691
Python 3.8 Libraries for Workbench....698
Python 3.7 Libraries for Workbench....706
Scala 2.11 Libraries for Workbench....712
Python 3.10 Libraries for JupyterLab....712
Python 3.9 Libraries for JupyterLab....719
Python 3.8 Libraries for JupyterLab....726
Python 3.7 Libraries for JupyterLab....733
R 4.0 Libraries....740
R 4.1 Libraries....743
R 3.6 Libraries....747

ML Runtimes 2022.11....750
Python 3.9 Libraries for Workbench....751
Python 3.8 Libraries for Workbench....755
Python 3.7 Libraries for Workbench....760
Scala 2.11 Libraries for Workbench....764
Python 3.9.6 Libraries for JupyterLab....765
Python 3.8 Libraries for JupyterLab....769
Python 3.7 Libraries for JupyterLab....774
R 4.1 Libraries....779
R 4.0 Libraries....783
R 3.6 Libraries....787
Python 3.9 Libraries for PBJ Workbench....791

Python 3.8 Libraries for PBJ Workbench....793
Python 3.7 Libraries for PBJ Workbench....795
PBJ R 4.1 Libraries....797
PBJ R 4.0 Libraries....799
PBJ R 3.6 Libraries....801

ML Runtimes 2022.04....803
Python 3.9 Libraries for Workbench....803
Python 3.8 Libraries for Workbench....805
Python 3.7 Libraries for Workbench....808
Scala 2.11 Libraries for Workbench....813
Python 3.9.6 Libraries for JupyterLab....814
Python 3.8 Libraries for JupyterLab....818
Python 3.7 Libraries for JupyterLab....823
R 4.0 Libraries....828
R 4.1 Libraries....830
R 3.6 Libraries....832

ML Runtimes 2021.12....834
Python 3.9 Libraries for Workbench....834
Python 3.8 Libraries for Workbench....836
Python 3.7 Libraries for Workbench....848
Scala 2.11 Libraries for Workbench....860
Python 3.9.6 Libraries for JupyterLab....861
Python 3.8 Libraries for JupyterLab....865
Python 3.7 Libraries for JupyterLab....879
R 4.0 Libraries....893
R 4.1 Libraries....895
R 3.6 Libraries....897

ML Runtimes 2021.09....899
Python 3.9 Libraries for Workbench....900
Python 3.8 Libraries for Workbench....902
Python 3.7 Libraries for Workbench....904
Python 3.6 Libraries for Workbench....907
Scala 2.11 Libraries for Workbench....909
Python 3.9.6 Libraries for JupyterLab....909
Python 3.8 Libraries for JupyterLab....913
Python 3.7 Libraries for JupyterLab....918
Python 3.6 Libraries for JupyterLab....922
R 4.0 Libraries....927
R 3.6 Libraries....929

ML Runtimes 2021.06....931
Python 3.8.6 Libraries for Workbench....931
Python 3.7.9 Libraries for Workbench....933
Python 3.6.12 Libraries for Workbench....935
Python 3.8.6 Libraries for JupyterLab....938
Python 3.7.9 Libraries for JupyterLab....941
Python 3.6.12 Libraries for JupyterLab....946
R 4.0 Libraries....949
R 3.6 Libraries....952

ML Runtimes 2021.04 954
RAPIDS Runtime PIP Python 3.7.8 Libraries for Workbench 954
RAPIDS Runtime PIP Python 3.8.6 Libraries for Workbench 958
RAPIDS Runtime PIP Python 3.7.8 Libraries for JupyterLab 963
RAPIDS Runtime PIP Python 3.8.6 Libraries for JupyterLab 967

ML Runtimes 2021.02 972
Python 3.8 Libraries for Workbench 972
Python 3.7 Libraries for Workbench 974
Python 3.6 Libraries for Workbench 976
Python 3.8 Libraries for JupyterLab 978
Python 3.7 Libraries for JupyterLab 982
Python 3.6 Libraries for JupyterLab 986
R 4.0 Libraries 990
R 3.6 Libraries 992

ML Runtimes 2020.11 994

ML Runtimes Pre-installed Packages overview

Cloudera AI ships with several base engine images that include Python kernels, and frequently used libraries.

Note: New ML Runtime releases are automatically added to the deployment, if Internet connection is available.

As the software and versions listed in *ML Runtimes Pre-installed Packages* are installed in the images, you shall not upgrade these versions as this may lead to dependency conflicts. If any related software in this list needs to be at a higher version, you must use a more recent version of the legacy engine or ML Runtime.

Related Information

Archive Runtimes list

Managing ML Runtimes

ML Runtimes 2025.01.1

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2025.01.1.

Note:

All images are now based on Ubuntu 24.04.

Spark and Python Supportability

Specific Python versions are compatible with specific Spark versions. For detailed information about the compatibility levels for various Python versions across different Spark releases, see Spark Python Supportability Matrix.

Python 3.12 Libraries for Conda

This section lists the packages in Python 3.12 Conda Runtimes that ship with ML Runtimes 2025.01.1.

Table 1: Packages in Python 3.12.8 PBJ Conda Runtimes - Standard

Library	Version
aiohappyeyeballs	2.4.0
aiohttp	3.10.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4

Library	Version
attrs	24.2.0
babel	2.16.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146
cattrs	24.1.0
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.2
contourpy	1.2.1
cryptography	43.0.0
cycler	0.12.1
dask	2024.8.2
dataclasses-json	0.6.7
debugpy	1.8.2
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.8.2
docstring-to-markdown	0.15
entrypoints	0.4
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.20.0
fonttools	4.53.1
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.9.0
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0

Library	Version
httpcore	1.0.5
httpx	0.27.2
idna	3.7
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.29.5
ipython	8.26.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.4
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
langchain	0.2.17
langchain-aws	0.1.16
langchain-community	0.2.19
langchain-core	0.2.43

Library	Version
langchain-text-splitters	0.2.4
langsmith	0.1.114
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5
marshmallow	3.22.0
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclassic	1.1.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.2
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.7
overrides	7.7.0
packaging	24.1
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
partd	1.4.2
pexpect	4.9.0
pillow	10.4.0
platformdirs	4.2.2
ply	3.11
prometheus_client	0.20.0
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3

Library	Version
py4j	0.10.9.5
pycparser	2.22
pydantic	2.9.0
pydantic_core	2.23.2
pygls	1.3.1
Pygments	2.18.0
pyparsing	3.1.2
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.2
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2
Send2Trash	1.8.3
setuptools	71.1.0
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.34
stack-data	0.6.3
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
toolz	0.12.1
tornado	6.4.2
traitlets	5.14.3

Library	Version
trino	0.330.0
types-python-dateutil	2.9.0.20241206
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0
wheel	0.45.1
yarl	1.9.11
zict	3.0.0
zipp	3.20.1

Python 3.12 Libraries for Workbench

This section lists the packages in Python 3.12 Workbench Runtimes that ship with ML Runtimes 2025.01.1.

Table 2: Packages in Python 3.12.8 PBJ Workbench Runtimes - Standard

Library	Version
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	24.2.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.2.1

Library	Version
cryptography	43.0.0
cycler	0.12.1
debugpy	1.8.2
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.20.0
fonttools	4.53.1
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.29.5
ipython	8.26.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.4
jmespath	1.0.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab_pygments	0.3.0
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
MarkupSafe	2.1.5
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
nbclassic	1.1.0
nbclient	0.10.0

Library	Version
nbconvert	7.16.4
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
overrides	7.7.0
packaging	24.1
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
pexpect	4.9.0
pillow	10.4.0
platformdirs	4.2.2
prometheus_client	0.20.0
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
Pygments	2.18.0
pyparsing	3.1.2
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.2
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2

Library	Version
Send2Trash	1.8.3
setuptools	75.6.0
six	1.16.0
sniffio	1.3.1
soupsieve	2.6
stack-data	0.6.3
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20241206
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0
wheel	0.45.1

Table 3: Packages in Python 3.12.8 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-5	12.5.1-1
cuda-libraries-dev-12-5	12.5.1-1
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	24.2.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146

Library	Version
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.2.1
cryptography	43.0.0
cycler	0.12.1
debugpy	1.8.2
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.20.0
fonttools	4.53.1
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.29.5
ipython	8.26.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.4
jmespath	1.0.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab_pygments	0.3.0
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
MarkupSafe	2.1.5

Library	Version
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
nbclassic	1.1.0
nbclient	0.10.0
nbconvert	7.16.4
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
overrides	7.7.0
packaging	24.1
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
pexpect	4.9.0
pillow	10.4.0
platformdirs	4.2.2
prometheus_client	0.20.0
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
Pygments	2.18.0
pyparsing	3.1.2
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.2
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3

Library	Version
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2
Send2Trash	1.8.3
setuptools	75.6.0
six	1.16.0
sniffio	1.3.1
soupsieve	2.6
stack-data	0.6.3
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20241206
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0
wheel	0.45.1

Python 3.11 Libraries for Workbench

This section lists the packages in Python 3.11 Workbench Runtimes that ship with ML Runtimes 2025.01.1.

Table 4: Packages in Python 3.11.11 PBJ Workbench Runtimes - Standard

Library	Version
anyio	4.7.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0

Library	Version
asttokens	2.4.1
attrs	24.2.0
beautifulsoup4	4.12.3
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.2.0
cryptography	42.0.8
cycler	0.12.1
debugpy	1.8.9
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.21.1
fonttools	4.44.3
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.29.5
ipython	8.30.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.4
jmespath	1.0.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2024.10.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.7.2

Library	Version
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab_pygments	0.3.0
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.5.1
MarkupSafe	3.0.2
matplotlib	3.7.0
matplotlib-inline	0.1.7
mistune	3.0.2
nbclassic	1.1.0
nbclient	0.10.1
nbconvert	7.16.4
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
pexpect	4.9.0
pillow	10.3.0
platformdirs	4.3.6
prometheus_client	0.21.1
prompt_toolkit	3.0.48
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
Pygments	2.18.0
pyparsing	3.1.1
pyspnego	0.11.0
python-dateutil	2.8.2

Library	Version
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.2
pyzmq	26.2.0
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
sniffio	1.3.1
soupsieve	2.6
stack-data	0.6.3
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.4.0
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20241206
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0

Table 5: Packages in Python 3.11.11 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-5	12.5.1-1
cuda-libraries-dev-12-5	12.5.1-1

Library	Version
anyio	4.7.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	24.2.0
beautifulsoup4	4.12.3
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.2.0
cryptography	42.0.8
cycler	0.12.1
debugpy	1.8.9
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.21.1
fonttools	4.44.3
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.29.5
ipython	8.30.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.4
jmespath	1.0.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2024.10.1

Library	Version
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab_pygments	0.3.0
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.5.1
MarkupSafe	3.0.2
matplotlib	3.7.0
matplotlib-inline	0.1.7
mistune	3.0.2
nbclassic	1.1.0
nbclient	0.10.1
nbconvert	7.16.4
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
pexpect	4.9.0
pillow	10.3.0
platformdirs	4.3.6
prometheus_client	0.21.1
prompt_toolkit	3.0.48
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22

Library	Version
Pygments	2.18.0
pyparsing	3.1.1
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.2
pyzmq	26.2.0
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
sniffio	1.3.1
soupsieve	2.6
stack-data	0.6.3
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.4.0
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20241206
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0

Python 3.10 Libraries for Workbench

This section lists the packages in Python 3.10 Workbench Runtimes that ship with ML Runtimes 2025.10.1.

Table 6: Packages in Python 3.10.16 PBJ Workbench Runtimes - Standard

Library	Version
anyio	4.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	24.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.8.9
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.29.5
ipython	8.10.0
ipython-genutils	0.2.0

Library	Version
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.3
jmespath	1.0.1
jsonpointer	2.3
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.14.1
jupyter_server_terminals	0.5.3
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.26.4
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.3.6
prometheus-client	0.16.0

Library	Version
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	26.2.0
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.4.2
traitlets	5.9.0
trino	0.330.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2

Library	Version
uri-template	1.2.0
urllib3	2.2.2
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0

Table 7: Packages in Python 3.10.16 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-5	12.5.1-1
cuda-libraries-dev-12-5	12.5.1-1
anyio	4.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	24.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.8.9
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1

Library	Version
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.29.5
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.3
jmespath	1.0.1
jsonpointer	2.3
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.14.1
jupyter_server_terminals	0.5.3
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.26.4
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0

Library	Version
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.3.6
prometheus-client	0.16.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	26.2.0
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1

Library	Version
tornado	6.4.2
traitlets	5.9.0
trino	0.330.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.2.0
urllib3	2.2.2
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2025.01.1

Table 8: Packages in Python 3.9.21 PBJ Workbench Runtimes - Standard

Library	Version
anyio	4.7.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
beautifulsoup4	4.12.3
bitarray	2.6.0
bleach	6.2.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.8.9

Library	Version
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
fastjsonschema	2.21.1
fonttools	4.43.0
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
importlib-resources	5.12.0
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.29.5
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.4
jmespath	1.0.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.4
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab_pygments	0.3.0
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
MarkupSafe	3.0.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.10.1

Library	Version
nbconvert	7.16.4
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.4
notebook_shim	0.2.4
numpy	1.26.4
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.3.6
prometheus_client	0.21.1
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3

Library	Version
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
sniffio	1.3.1
soupsieve	2.6
stack-data	0.6.2
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.4.0
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	1.26.19
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
zipp	3.15.0

Table 9: Packages in Python 3.9.21 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-5	12.5.1-1
cuda-libraries-dev-12-5	12.5.1-1
anyio	4.7.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
beautifulsoup4	4.12.3
bitarray	2.6.0
bleach	6.2.0

Library	Version
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.8.9
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
fastjsonschema	2.21.1
fonttools	4.43.0
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
importlib-resources	5.12.0
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.29.5
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.4
jmespath	1.0.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.4
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3

Library	Version
jupyterlab_pygments	0.3.0
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
MarkupSafe	3.0.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.10.1
nbconvert	7.16.4
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.4
notebook_shim	0.2.4
numpy	1.26.4
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.3.6
prometheus_client	0.21.1
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7

Library	Version
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
sniffio	1.3.1
soupsieve	2.6
stack-data	0.6.2
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.4.0
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	1.26.19
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
zipp	3.15.0

Scala 2.12 Libraries for Workbench

This section lists the packages in Scala 2.12 Workbench Runtimes that ship with ML Runtimes 2025.01.1.

Table 10: Scala 2.12 Libraries for PBJ Workbench - Standard

Library	Version
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	24.2.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.2.1
cryptography	43.0.0
cycler	0.12.1
debugpy	1.8.2
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.20.0
fonttools	4.53.1
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.29.5
ipython	8.26.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.4
jmespath	1.0.1
jsonpointer	3.0.0

Library	Version
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab_pygments	0.3.0
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
MarkupSafe	2.1.5
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
nbclassic	1.1.0
nbclient	0.10.0
nbconvert	7.16.4
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
overrides	7.7.0
packaging	24.1
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
pexpect	4.9.0
pillow	10.4.0
platformdirs	4.2.2
prometheus_client	0.20.0
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3

Library	Version
py4j	0.10.9.5
pycparser	2.22
Pygments	2.18.0
pyparsing	3.1.2
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.2
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2
Send2Trash	1.8.3
setuptools	75.6.0
six	1.16.0
sniffio	1.3.1
soupsieve	2.6
stack-data	0.6.3
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20241206
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1

Library	Version
websocket-client	1.8.0
wheel	0.45.1

Python 3.12 Libraries for JupyterLab

This section lists the packages in Python 3.12 JupyterLab Runtimes that ship with ML Runtimes 2025.01.1.

Table 11: Packages in Python 3.12.8 PBJ JupyterLab Runtimes - Standard

Library	Version
aiohappyeyeballs	2.4.0
aiohttp	3.10.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4
attrs	24.2.0
babel	2.16.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146
cattrs	24.1.0
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.2
contourpy	1.2.1
cryptography	43.0.0
cycler	0.12.1
dask	2024.8.2

Library	Version
dataclasses-json	0.6.7
debugpy	1.8.2
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.8.2
docstring-to-markdown	0.15
entrypoints	0.4
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.20.0
fonttools	4.53.1
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.9.0
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5
httpx	0.27.2
idna	3.7
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.29.5
ipython	8.26.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1

Library	Version
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.4
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
langchain	0.2.17
langchain-aws	0.1.16
langchain-community	0.2.19
langchain-core	0.2.43
langchain-text-splitters	0.2.4
langsmith	0.1.114
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5
marshmallow	3.22.0
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclassic	1.1.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.2
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7

Library	Version
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.7
overrides	7.7.0
packaging	24.1
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
partd	1.4.2
pexpect	4.9.0
pillow	10.4.0
platformdirs	4.2.2
ply	3.11
prometheus_client	0.20.0
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
pydantic	2.9.0
pydantic_core	2.23.2
pygls	1.3.1
Pygments	2.18.0
pyparsing	3.1.2
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.2
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1

Library	Version
rpds-py	0.22.3
s3transfer	0.10.2
Send2Trash	1.8.3
setuptools	71.1.0
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.34
stack-data	0.6.3
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
toolz	0.12.1
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20241206
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0
wheel	0.45.1
yarl	1.9.11
zict	3.0.0
zipp	3.20.1

Table 12: Packages in Python 3.12.8 PBJ JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-5	12.5.1-1
cuda-libraries-dev-12-5	12.5.1-1
aiohappyeyeballs	2.4.0
aiohttp	3.10.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4
attrs	24.2.0
babel	2.16.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146
cattrs	24.1.0
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.2
contourpy	1.2.1
cryptography	43.0.0
cycler	0.12.1
dask	2024.8.2
dataclasses-json	0.6.7
debugpy	1.8.2
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.8.2

Library	Version
docstring-to-markdown	0.15
entrypoints	0.4
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.20.0
fonttools	4.53.1
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.9.0
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5
httpx	0.27.2
idna	3.7
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.29.5
ipython	8.26.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.9
jupyter_core	5.7.2

Library	Version
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.4
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
langchain	0.2.17
langchain-aws	0.1.16
langchain-community	0.2.19
langchain-core	0.2.43
langchain-text-splitters	0.2.4
langsmith	0.1.114
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5
marshmallow	3.22.0
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclassic	1.1.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.2
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.7
overrides	7.7.0
packaging	24.1
pandas	2.1.4

Library	Version
pandocfilters	1.5.1
parso	0.8.4
partd	1.4.2
pexpect	4.9.0
pillow	10.4.0
platformdirs	4.2.2
ply	3.11
prometheus_client	0.20.0
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
pydantic	2.9.0
pydantic_core	2.23.2
pygls	1.3.1
Pygments	2.18.0
pyparsing	3.1.2
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.2
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2
Send2Trash	1.8.3
setuptools	71.1.0
six	1.16.0
smmap	5.0.1

Library	Version
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.34
stack-data	0.6.3
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
toolz	0.12.1
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20241206
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0
wheel	0.45.1
yarl	1.9.11
zict	3.0.0
zipp	3.20.1

Python 3.11 Libraries for JupyterLab

This section lists the packages in Python 3.11 JupyterLab Runtimes that ship with ML Runtimes 2024.10.1.

Table 13: Packages in Python 3.11.11 PBJ JupyterLab Runtimes - Standard

Library	Version
aiohappyeyeballs	2.4.3
aiohttp	3.11.8

Library	Version
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.7.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4
attrs	24.2.0
babel	2.16.0
beautifulsoup4	4.12.3
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.162
botocore	1.34.162
cattrs	24.1.2
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.1.0
colorama	0.4.6
comm	0.2.2
contourpy	1.2.0
cryptography	42.0.8
cycler	0.12.1
dask	2024.11.2
dataclasses-json	0.6.7
debugpy	1.8.9
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.11.2
docstring-to-markdown	0.15
entrypoints	0.4
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.21.1

Library	Version
fonttools	4.44.3
fqdn	1.5.1
frozenlist	1.5.0
fsspec	2024.10.0
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.1.1
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.7
httpx	0.27.2
idna	3.7
importlib_metadata	8.5.0
impyla	0.18.0
ipykernel	6.29.5
ipython	8.30.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.4
jmespath	1.0.1
json5	0.10.0
jsonpatch	1.33
jsonpath-ng	1.7.0
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2024.10.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0

Library	Version
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.5.1
langchain	0.2.17
langchain-aws	0.1.17
langchain-community	0.2.19
langchain-core	0.2.43
langchain-text-splitters	0.2.4
langsmith	0.1.147
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	3.0.2
marshmallow	3.23.1
matplotlib	3.7.0
matplotlib-inline	0.1.7
mistune	3.0.2
msgpack	1.1.0
multidict	6.1.0
mypy-extensions	1.0.0
nbclassic	1.1.0
nbclient	0.10.1
nbconvert	7.16.4
nbdime	4.0.2
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.12
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
partd	1.4.2
pexpect	4.9.0
pillow	10.3.0

Library	Version
platformdirs	4.3.6
ply	3.11
prometheus_client	0.21.1
prompt_toolkit	3.0.48
propcache	0.2.0
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
pydantic	2.10.2
pydantic_core	2.27.1
pygls	1.3.1
Pygments	2.18.0
pyparsing	3.1.1
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.2
pyzmq	26.2.0
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
requests-toolbelt	1.0.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.4
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.36

Library	Version
stack-data	0.6.3
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.4.0
toolz	1.0.0
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20241206
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.18.0
zict	3.0.0
zipp	3.21.0

Table 14: Packages in Python 3.11.11 PBJ JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-5	12.5.1-1
cuda-libraries-dev-12-5	12.5.1-1
aiohappyeyeballs	2.4.3
aiohttp	3.11.8
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.7.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0

Library	Version
asttokens	2.4.1
async-lru	2.0.4
attrs	24.2.0
babel	2.16.0
beautifulsoup4	4.12.3
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.162
botocore	1.34.162
cattrs	24.1.2
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.1.0
colorama	0.4.6
comm	0.2.2
contourpy	1.2.0
cryptography	42.0.8
cycler	0.12.1
dask	2024.11.2
dataclasses-json	0.6.7
debugpy	1.8.9
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.11.2
docstring-to-markdown	0.15
entrypoints	0.4
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.21.1
fonttools	4.44.3
fqdn	1.5.1
frozenlist	1.5.0
fsspec	2024.10.0
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.1.1

Library	Version
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.7
httpx	0.27.2
idna	3.7
importlib_metadata	8.5.0
impyla	0.18.0
ipykernel	6.29.5
ipython	8.30.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.4
jmespath	1.0.1
json5	0.10.0
jsonpatch	1.33
jsonpath-ng	1.7.0
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2024.10.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.5.1
langchain	0.2.17
langchain-aws	0.1.17

Library	Version
langchain-community	0.2.19
langchain-core	0.2.43
langchain-text-splitters	0.2.4
langsmith	0.1.147
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	3.0.2
marshmallow	3.23.1
matplotlib	3.7.0
matplotlib-inline	0.1.7
mistune	3.0.2
msgpack	1.1.0
multidict	6.1.0
mypy-extensions	1.0.0
nbclassic	1.1.0
nbclient	0.10.1
nbconvert	7.16.4
nbdime	4.0.2
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.12
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
partd	1.4.2
pexpect	4.9.0
pillow	10.3.0
platformdirs	4.3.6
ply	3.11
prometheus_client	0.21.1
prompt_toolkit	3.0.48
propcache	0.2.0
protobuf	4.25.3
psutil	6.1.0

Library	Version
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
pydantic	2.10.2
pydantic_core	2.27.1
pygls	1.3.1
Pygments	2.18.0
pyparsing	3.1.1
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.2
pyzmq	26.2.0
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
requests-toolbelt	1.0.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.4
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.36
stack-data	0.6.3
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.4.0

Library	Version
toolz	1.0.0
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20241206
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.18.0
zict	3.0.0
zipp	3.21.0

Python 3.10 Libraries for JupyterLab

This section lists the packages in Python 3.10 JupyterLab Runtimes that ship with ML Runtimes 2025.01.1.

Table 15: Packages in Python 3.10.16 PBJ JupyterLab Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
annotated-types	0.7.0
anyio	4.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	24.2.0
Babel	2.12.1
backcall	0.2.0

Library	Version
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.149
botocore	1.34.162
cattrs	23.2.3
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.8.9
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.4

Library	Version
httpx	0.27.0
idna	3.7
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.29.5
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.14.1
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.27.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.17
langchain-aws	0.1.16
langchain-community	0.2.19
langchain-core	0.2.43
langchain-text-splitters	0.2.2

Library	Version
langsmith	0.1.147
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.2
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbdime	4.0.1
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.3.6
ply	3.11
prometheus-client	0.16.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2

Library	Version
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0
pydantic_core	2.10.0
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	26.2.0
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
requests-toolbelt	1.0.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.1
tornado	6.4.2

Library	Version
traitlets	5.9.0
trino	0.330.0
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.2.0
urllib3	2.2.2
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Table 16: Packages in Python 3.10.16 PBJ JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-5	12.5.1-1
cuda-libraries-dev-12-5	12.5.1-1
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
annotated-types	0.7.0
anyio	4.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	24.2.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.149
botocore	1.34.162

Library	Version
cattrs	23.2.3
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.8.9
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.7
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.29.5

Library	Version
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.14.1
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.27.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.17
langchain-aws	0.1.16
langchain-community	0.2.19
langchain-core	0.2.43
langchain-text-splitters	0.2.2
langsmith	0.1.147
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.2
marshmallow	3.21.2

Library	Version
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbdime	4.0.1
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.3.6
ply	3.11
prometheus-client	0.16.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0
pydantic_core	2.10.0
pygls	1.3.1

Library	Version
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	26.2.0
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
requests-toolbelt	1.0.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.1
tornado	6.4.2
traitlets	5.9.0
trino	0.330.0
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3

Library	Version
tzlocal	5.2
uri-template	1.2.0
urllib3	2.2.2
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Python 3.9 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2025.01.1.

Table 17: Packages in Python 3.9.21 PBJ JupyterLab Runtimes - Standard

Library	Version
aiohappyeyeballs	2.3.4
aiohttp	3.10.0
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.7.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.2.0
Babel	2.15.0
backcall	0.2.0
beautifulsoup4	4.12.3
bitarray	2.6.0
bleach	6.2.0
boto3	1.34.149
botocore	1.34.162
cattrs	23.2.3
certifi	2024.7.4

Library	Version
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2024.7.1
dataclasses-json	0.6.7
debugpy	1.8.9
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.7.1
docstring-to-markdown	0.15
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.21.1
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.6.1
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5
httpx	0.27.0
idna	3.7
importlib-resources	5.12.0
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.29.5
ipython	8.10.0

Library	Version
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.17
langchain-aws	0.1.16
langchain-community	0.2.19
langchain-core	0.2.43
langchain-text-splitters	0.2.2
langsmith	0.1.147
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	3.0.2
marshmallow	3.21.3
matplotlib	3.7.0

Library	Version
matplotlib-inline	0.1.3
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclassic	1.0.0
nbclient	0.10.1
nbconvert	7.16.4
nbdime	4.0.1
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.4
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.6
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.3.6
ply	3.11
prometheus_client	0.21.1
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0
pydantic_core	2.10.0
pygls	1.3.1
Pygments	2.15.1

Library	Version
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
requests-toolbelt	1.0.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.31
stack-data	0.6.2
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.4.0
tomli	2.0.1
toolz	0.12.1
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3

Library	Version
tzlocal	5.2
uri-template	1.3.0
urllib3	1.26.19
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Table 18: Packages in Python 3.9.21 PBJ JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-5	12.5.1-1
cuda-libraries-dev-12-5	12.5.1-1
aiohappyeyeballs	2.3.4
aiohttp	3.10.0
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.7.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.2.0
Babel	2.15.0
backcall	0.2.0
beautifulsoup4	4.12.3
bitarray	2.6.0
bleach	6.2.0
boto3	1.34.149
botocore	1.34.162
cattrs	23.2.3
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2

Library	Version
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2024.7.1
dataclasses-json	0.6.7
debugpy	1.8.9
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.7.1
docstring-to-markdown	0.15
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.21.1
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.6.1
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5
httpx	0.27.0
idna	3.7
importlib-resources	5.12.0
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.29.5
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0

Library	Version
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.17
langchain-aws	0.1.16
langchain-community	0.2.19
langchain-core	0.2.43
langchain-text-splitters	0.2.2
langsmith	0.1.147
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	3.0.2
marshmallow	3.21.3
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	3.0.2

Library	Version
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclassic	1.0.0
nbclient	0.10.1
nbconvert	7.16.4
nbdime	4.0.1
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.4
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.6
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.3.6
ply	3.11
prometheus_client	0.21.1
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	6.1.0
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0
pydantic_core	2.10.0
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0

Library	Version
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
requests-toolbelt	1.0.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.31
stack-data	0.6.2
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.4.0
tomli	2.0.1
toolz	0.12.1
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0

Library	Version
urllib3	1.26.19
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

R 4.4 Libraries

This section lists the R 4.4.0 libraries that ship with ML Runtimes Runtimes 2025.01.1.

Table 19: Packages in R 4.4.0 PBJ Workbench Runtimes - Standard - R packages

Package	Version
askpass	1.2.0
base	4.4.0
base64enc	0.1-3
boot	1.3-30
Cairo	1.6-2
class	7.3-22
cli	3.6.2
cluster	2.1.6
codetools	0.2-20
colorspace	2.1-0
compiler	4.4.0
crayon	1.5.3
datasets	4.4.0
digest	0.6.37
evaluate	1.0.1
fansi	1.0.6
farver	2.1.1
fastmap	1.2.0
foreign	0.8-86
ggplot2	3.5.1

Package	Version
glue	1.7.0
graphics	4.4.0
grDevices	4.4.0
grid	4.4.0
gtable	0.3.5
htmltools	0.5.8.1
IRdisplay	1.1
IRkernel	1.3.2
isoband	0.2.7
jsonlite	1.8.9
KernSmooth	2.23-22
labeling	0.4.3
lattice	0.22-6
lifecycle	1.0.4
magrittr	2.0.3
MASS	7.3-60.2
Matrix	1.7-0
methods	4.4.0
mgcv	1.9-1
munsell	0.5.1
nlme	3.1-164
nnet	7.3-19
openssl	2.1.2
parallel	4.4.0
pbdZMQ	0.3-13
pillar	1.9.0
pkgconfig	2.0.3
R6	2.5.1
RColorBrewer	1.1-3
repr	1.1.7
rlang	1.1.3
rpart	4.1.23

Package	Version
Rserve	1.8-13
scales	1.3.0
spatial	7.3-17
splines	4.4.0
stats	4.4.0
stats4	4.4.0
survival	3.5-8
sys	3.4.2
tcltk	4.4.0
tibble	3.2.1
tools	4.4.0
utf8	1.2.4
utils	4.4.0
uuid	1.2-1
vctrs	0.6.5
viridisLite	0.4.2
withr	3.0.0

Table 20: Packages in R 4.4.0 PBJ Workbench Runtimes - Standard - Python libraries

Library	Version
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	24.2.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.2.1

Library	Version
cryptography	43.0.0
cycler	0.12.1
debugpy	1.8.2
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.20.0
fonttools	4.53.1
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.29.5
ipython	8.26.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.4
jmespath	1.0.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab_pygments	0.3.0
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
MarkupSafe	2.1.5
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
nbclassic	1.1.0
nbclient	0.10.0

Library	Version
nbconvert	7.16.4
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
overrides	7.7.0
packaging	24.1
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
pexpect	4.9.0
pillow	10.4.0
platformdirs	4.2.2
prometheus_client	0.20.0
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
Pygments	2.18.0
pyparsing	3.1.2
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.2
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.22.3
s3transfer	0.10.2

Library	Version
Send2Trash	1.8.3
setuptools	75.6.0
six	1.16.0
sniffio	1.3.1
soupsieve	2.6
stack-data	0.6.3
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
tornado	6.4.2
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20241206
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0
wheel	0.45.1

ML Runtimes 2024.10.1

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2024.10.1.

Note:

Nvidia GPU Editions with a Workbench Editor come with CUDA 11.4.1 and cuDNN 8.2 versions preinstalled.

Nvidia GPU Editions with JupyterLab or a PBJ Workbench Editor and with Python 3.7 come with CUDA 11.8.0 and cuDNN 8.7 versions preinstalled.

All the other Nvidia GPU Edition Runtimes come with CUDA 12.3 and cuDNN 9.0 versions preinstalled.

Spark and Python Supportability

Specific Python versions are compatible with specific Spark versions. For detailed information about the compatibility levels for various Python versions across different Spark releases, see Spark Python Supportability Matrix.

Python 3.10 Libraries for Conda

This section lists the packages in Python 3.10 Conda Runtimes that ship with ML Runtimes 2024.10.1.

Table 21: Packages in Python 3.10.13 Conda Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
annotated-types	0.7.0
anyio	4.4.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.149
botocore	1.34.162
cattrs	23.2.3
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	5.1.1

Library	Version
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.7
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6

Library	Version
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.14.1
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.27.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.13
langchain-aws	0.1.16
langchain-community	0.2.12
langchain-core	0.2.34
langchain-text-splitters	0.2.2
langsmith	0.1.103
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.2
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.6.3
nbconvert	7.2.9
nbdime	4.0.1
nbformat	5.7.3
nest-asyncio	1.5.5
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4

Library	Version
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.16.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0
pydantic_core	2.10.0
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.1.2
raz_client	1.1.0
referencing	0.34.0
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0

Library	Version
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
typeguard	2.13.3
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.2.0
urllib3	2.2.2
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Python 3.12 Libraries for Workbench

This section lists the packages in Python 3.12 Workbench Runtimes that ship with ML Runtimes 2024.10.1.

Table 22: Packages in Python 3.12.4 PBJ Workbench Runtimes - Standard

Library	Version
anyio	4.4.0
argon2-cffi	23.1.0

Library	Version
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.2.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.2.1
cryptography	43.0.0
cycler	0.12.1
debugpy	1.8.2
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.20.0
fonttools	4.53.1
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.29.5
ipython	8.26.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.4
jmespath	1.0.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2

Library	Version
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab_pygments	0.3.0
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
MarkupSafe	2.1.5
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
nbclassic	1.1.0
nbclient	0.10.0
nbconvert	7.16.4
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
overrides	7.7.0
packaging	24.1
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
pexpect	4.9.0
pillow	10.4.0
platformdirs	4.2.2
prometheus_client	0.20.0
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
Pygments	2.18.0
pyparsing	3.1.2

Library	Version
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.1
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.19.0
s3transfer	0.10.2
Send2Trash	1.8.3
setuptools	71.1.0
six	1.16.0
sniffio	1.3.1
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
tornado	6.4.1
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240316
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
wheel	0.44.0

Table 23: Packages in Python 3.12.4 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.2.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.2
contourpy	1.2.1
cryptography	43.0.0
cycler	0.12.1
debugpy	1.8.2
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.20.0
fonttools	4.53.1
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.29.5
ipython	8.26.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1

Library	Version
Jinja2	3.1.4
jmespath	1.0.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab_pygments	0.3.0
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
MarkupSafe	2.1.5
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
nbclassic	1.1.0
nbclient	0.10.0
nbconvert	7.16.4
nbformat	5.10.4
nest-asyncio	1.6.0
notebook	6.5.7
notebook_shim	0.2.4
numpy	1.26.4
overrides	7.7.0
packaging	24.1
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
pexpect	4.9.0
pillow	10.4.0
platformdirs	4.2.2
prometheus_client	0.20.0
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0

Library	Version
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
Pygments	2.18.0
pyparsing	3.1.2
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.1
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.19.0
s3transfer	0.10.2
Send2Trash	1.8.3
setuptools	71.1.0
six	1.16.0
sniffio	1.3.1
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
tornado	6.4.1
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240316
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2

Library	Version
wcwidth	0.2.13
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
wheel	0.44.0

Python 3.11 Libraries for Workbench

This section lists the packages in Python 3.11 Workbench Runtimes that ship with ML Runtimes 2024.10.1.

Table 24: Packages in Python 3.11.9 PBJ Workbench Runtimes - Standard

Library	Version
anyio	4.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.0
contourpy	1.2.0
cryptography	42.0.8
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
gssapi	1.8.3

Library	Version
idna	3.7
impyla	0.18.0
ipykernel	6.26.0
ipython	8.17.2
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.3
jmespath	1.0.1
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.5.0
jupyter_server	2.14.1
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.5.1
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.6
notebook_shim	0.2.3
numpy	1.26.4
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3

Library	Version
pexpect	4.8.0
pillow	10.3.0
platformdirs	4.0.0
prometheus-client	0.18.0
prompt-toolkit	3.0.41
protobuf	4.25.3
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.16.1
pyparsing	3.1.1
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	24.0.1
raz_client	1.1.0
referencing	0.31.0
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.0
s3transfer	0.10.2
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.13.0

Library	Version
trino	0.330.0
types-python-dateutil	2.8.19.14
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.10
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0

Table 25: Packages in Python 3.11.9 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	4.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.0
contourpy	1.2.0
cryptography	42.0.8
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4

Library	Version
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.26.0
ipython	8.17.2
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.3
jmespath	1.0.1
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.5.0
jupyter_server	2.14.1
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.5.1
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.6
notebook_shim	0.2.3
numpy	1.26.4

Library	Version
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pillow	10.3.0
platformdirs	4.0.0
prometheus-client	0.18.0
prompt-toolkit	3.0.41
protobuf	4.25.3
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.16.1
pyparsing	3.1.1
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	24.0.1
raz_client	1.1.0
referencing	0.31.0
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.0
s3transfer	0.10.2
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0

Library	Version
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.13.0
trino	0.330.0
types-python-dateutil	2.8.19.14
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.10
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0

Python 3.10 Libraries for Workbench

This section lists the packages in Python 3.10 Workbench Runtimes that ship with ML Runtimes 2024.10.1.

Table 26: Packages in Python 3.10.14 Workbench Runtimes - Standard

Library	Version
asttokens	2.2.1
backcall	0.2.0
bitarray	2.7.3
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.6.7
decorator	5.1.1
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0

Library	Version
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.19.1
jmespath	1.0.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.26.4
packaging	23.2
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
raz_client	1.1.0
requests	2.31.0

Library	Version
requests-kerberos	0.15.0
s3transfer	0.10.2
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
tzdata	2023.3
tzlocal	5.2
urllib3	1.26.18
wcwidth	0.2.5

Table 27: Packages in Python 3.10.14 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0

Library	Version
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.3
jmespath	1.0.1
jsonpointer	2.3
jsonschema	4.18.0
jsonschema-specifications	2023.12.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.14.1
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.26.4
overrides	7.7.0

Library	Version
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
protobuf	4.25.3
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.0.2
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
s3transfer	0.10.2
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0

Library	Version
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
tzdata	2023.3
tzlocal	5.2
uri-template	1.2.0
urllib3	2.2.2
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0

Table 28: Packages in Python 3.10.14 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.6.6

Library	Version
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.3
jmespath	1.0.1
jsonpointer	2.3
jsonschema	4.18.0
jsonschema-specifications	2023.12.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.14.1
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6

Library	Version
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.26.4
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
protobuf	4.25.3
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.0.2
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
s3transfer	0.10.2
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0

Library	Version
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
tzdata	2023.3
tzlocal	5.2
uri-template	1.2.0
urllib3	2.2.2
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2024.10.1.

Table 29: Packages in Python 3.9.19 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.6.7
decorator	5.1.1

Library	Version
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
gssapi	1.8.3
idna	3.7
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.19.1
jmespath	1.0.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.26.4
packaging	23.2
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0

Library	Version
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
raz_client	1.1.0
requests	2.31.0
requests-kerberos	0.15.0
s3transfer	0.10.2
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
typing_extensions	4.11.0
tzdata	2023.3
tzlocal	5.2
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 30: Packages in Python 3.9.19 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.6.7

Library	Version
decorator	5.1.1
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
gssapi	1.8.3
idna	3.7
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.19.1
jmespath	1.0.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.26.4
packaging	23.2
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9

Library	Version
pyspnego	0.11.0
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
raz_client	1.1.0
requests	2.31.0
requests-kerberos	0.15.0
s3transfer	0.10.2
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
typing_extensions	4.11.0
tzdata	2023.3
tzlocal	5.2
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 31: Packages in Python 3.9.19 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2

Library	Version
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.1
Jinja2	3.1.3
jmespath	1.0.1
jsonpointer	2.4
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.4
jupyter_core	4.12.0
jupyter_server	2.14.1
jupyter_server_terminals	0.5.3
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
MarkupSafe	2.0.1
matplotlib	3.7.0

Library	Version
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.26.4
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
protobuf	4.25.3
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4

Library	Version
rfc3986-validator	0.1.1
rpds-py	0.18.1
s3transfer	0.10.2
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing_extensions	4.3.0
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	1.26.19
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
zipp	3.15.0

Table 32: Packages in Python 3.9.19 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0

Library	Version
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.1
Jinja2	3.1.3
jmespath	1.0.1
jsonpointer	2.4
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.4

Library	Version
jupyter_core	4.12.0
jupyter_server	2.14.1
jupyter_server_terminals	0.5.3
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.26.4
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
protobuf	4.25.3
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2

Library	Version
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
s3transfer	0.10.2
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing_extensions	4.3.0
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	1.26.19
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
zipp	3.15.0

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with ML Runtimes 2024.10.1.

Table 33: Packages in Python 3.8.19 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
backports.zoneinfo	0.2.1
bitarray	2.6.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.6.7
decorator	5.1.1
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
gssapi	1.8.3
idna	3.7
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
jedi	0.19.1
jmespath	1.0.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.4

Library	Version
packaging	23.2
pandas	2.0.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
raz_client	1.1.0
requests	2.31.0
requests-kerberos	0.15.0
s3transfer	0.10.2
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
typing_extensions	4.11.0
tzdata	2023.3
tzlocal	5.2
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 34: Packages in Python 3.8.19 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
backports.zoneinfo	0.2.1
bitarray	2.6.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.6.7
decorator	5.1.1
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
gssapi	1.8.3
idna	3.7
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
jedi	0.19.1
jmespath	1.0.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5

Library	Version
numpy	1.24.4
packaging	23.2
pandas	2.0.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
raz_client	1.1.0
requests	2.31.0
requests-kerberos	0.15.0
s3transfer	0.10.2
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
typing_extensions	4.11.0
tzdata	2023.3
tzlocal	5.2
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 35: Packages in Python 3.8.19 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
backports.zoneinfo	0.2.1
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.1

Library	Version
Jinja2	3.1.3
jmespath	1.0.1
jsonpointer	2.4
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.4
jupyter_core	4.12.0
jupyter_server	2.14.1
jupyter_server_terminals	0.5.3
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
overrides	7.7.0
packaging	23.2
pandas	2.0.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.13.1
prompt-toolkit	3.0.38
protobuf	4.25.3

Library	Version
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
s3transfer	0.10.2
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing_extensions	4.3.0
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	1.26.19

Library	Version
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0
zipp	3.15.0

Table 36: Packages in Python 3.8.19 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
backports.zoneinfo	0.2.1
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
boto3	1.34.149
botocore	1.34.162
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0

Library	Version
fqdn	1.5.1
gssapi	1.8.3
idna	3.7
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.1
Jinja2	3.1.3
jmespath	1.0.1
jsonpointer	2.4
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.4
jupyter_core	4.12.0
jupyter_server	2.14.1
jupyter_server_terminals	0.5.3
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
overrides	7.7.0

Library	Version
packaging	23.2
pandas	2.0.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.13.1
prompt-toolkit	3.0.38
protobuf	4.25.3
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
s3transfer	0.10.2
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0

Library	Version
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing_extensions	4.3.0
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	1.26.19
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0
zipp	3.15.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with ML Runtimes 2024.10.1.

Table 37: Packages in Python 3.7.17 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2024.7.4
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.7
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2

Library	Version
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.8.0

Table 38: Packages in Python 3.7.17 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2024.7.4
charset-normalizer	2.0.6

Library	Version
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.7
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5

Library	Version
zipp	3.8.0

Table 39: Packages in Python 3.7.17 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.7
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.1.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4

Library	Version
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1

Library	Version
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Table 40: Packages in Python 3.7.17 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.7
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0

Library	Version
jedi	0.18.1
Jinja2	3.1.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0

Library	Version
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with ML Runtimes 2024.10.1.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-reflect	2.11.12
org.scala-lang:scala-compiler	2.11.12
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2
org.scalameta:scalameta	1.0.0

Python 3.12 Libraries for JupyterLab

This section lists the packages in Python 3.12 JupyterLab Runtimes that ship with ML Runtimes 2024.10.1.

Table 41: Packages in Python 3.12.4 JupyterLab Runtimes - Standard

Library	Version
aiohappyeyeballs	2.4.0
aiohttp	3.10.5

Library	Version
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4
attrs	24.2.0
babel	2.16.0
backcall	0.2.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146
cattrs	24.1.0
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.2
contourpy	1.2.1
cryptography	43.0.0
cycler	0.12.1
dask	2024.8.2
dataclasses-json	0.6.7
debugpy	1.8.2
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.8.2
docstring-to-markdown	0.15
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.20.0

Library	Version
fonttools	4.53.1
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.9.0
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5
httpx	0.27.2
idna	3.7
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	8.6.2
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.4
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3

Library	Version
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
langchain	0.2.16
langchain-aws	0.1.16
langchain-community	0.2.16
langchain-core	0.2.38
langchain-text-splitters	0.2.4
langsmith	0.1.114
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5
marshmallow	3.22.0
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.2
nbformat	5.10.4
nest-asyncio	1.6.0
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.7
overrides	7.7.0
packaging	24.1
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
partd	1.4.2
pexpect	4.9.0
pickleshare	0.7.5
pillow	10.4.0
platformdirs	4.2.2
ply	3.11
prometheus_client	0.20.0

Library	Version
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22
pydantic	2.9.0
pydantic_core	2.23.2
pygls	1.3.1
Pygments	2.18.0
pyparsing	3.1.2
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.2
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.20.0
s3transfer	0.10.2
Send2Trash	1.8.3
setuptools	71.1.0
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.34
stack-data	0.6.3
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1

Library	Version
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
toolz	0.12.1
tornado	6.4.1
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240821
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.8.0
webencodings	0.5.1
websocket-client	1.8.0
wheel	0.44.0
yarl	1.9.11
zict	3.0.0
zipp	3.20.1

Table 42: Packages in Python 3.12.4 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohappyeyeballs	2.4.0
aiohttp	3.10.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1

Library	Version
async-lru	2.0.4
attrs	24.2.0
babel	2.16.0
backcall	0.2.0
beautifulsoup4	4.12.3
bitarray	2.9.2
bleach	6.1.0
boto3	1.34.146
botocore	1.34.146
cattrs	24.1.0
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.2
contourpy	1.2.1
cryptography	43.0.0
cycler	0.12.1
dask	2024.8.2
dataclasses-json	0.6.7
debugpy	1.8.2
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.8.2
docstring-to-markdown	0.15
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.20.0
fonttools	4.53.1
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.9.0
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3

Library	Version
h11	0.14.0
httpcore	1.0.5
httpx	0.27.2
idna	3.7
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	8.6.2
jupyter_core	5.7.2
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.4
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.6.0
langchain	0.2.16
langchain-aws	0.1.16
langchain-community	0.2.16
langchain-core	0.2.38
langchain-text-splitters	0.2.4

Library	Version
langsmith	0.1.114
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5
marshmallow	3.22.0
matplotlib	3.9.1
matplotlib-inline	0.1.7
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.2
nbformat	5.10.4
nest-asyncio	1.6.0
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.7
overrides	7.7.0
packaging	24.1
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.4
partd	1.4.2
pexpect	4.9.0
pickleshare	0.7.5
pillow	10.4.0
platformdirs	4.2.2
ply	3.11
prometheus_client	0.20.0
prompt_toolkit	3.0.47
protobuf	4.25.3
psutil	6.0.0
ptyprocess	0.7.0
pure-sasl	0.6.2
pure_eval	0.2.3
py4j	0.10.9.5
pycparser	2.22

Library	Version
pydantic	2.9.0
pydantic_core	2.23.2
pygls	1.3.1
Pygments	2.18.0
pyparsing	3.1.2
pyspnego	0.11.0
python-dateutil	2.9.0.post0
python-json-logger	2.0.7
pytz	2024.1
PyYAML	6.0.2
pyzmq	26.0.3
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.20.0
s3transfer	0.10.2
Send2Trash	1.8.3
setuptools	71.1.0
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.34
stack-data	0.6.3
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
toolz	0.12.1
tornado	6.4.1
traitlets	5.14.3
trino	0.330.0
types-python-dateutil	2.9.0.20240821

Library	Version
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2024.1
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.13
webcolors	24.8.0
webencodings	0.5.1
websocket-client	1.8.0
wheel	0.44.0
yarl	1.9.11
zict	3.0.0
zipp	3.20.1

Python 3.11 Libraries for JupyterLab

This section lists the packages in Python 3.11 JupyterLab Runtimes that ship with ML Runtimes 2024.10.1.

Table 43: Packages in Python 3.11.9 JupyterLab Runtimes - Standard

Library	Version
aiohappyeyeballs	2.4.0
aiohttp	3.10.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4
attrs	24.2.0
babel	2.16.0
backcall	0.2.0
beautifulsoup4	4.12.3
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.149

Library	Version
botocore	1.34.162
cattrs	24.1.0
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.0
contourpy	1.2.0
cryptography	42.0.8
cycler	0.12.1
dask	2024.8.2
dataclasses-json	0.6.7
debugpy	1.8.0
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.8.2
docstring-to-markdown	0.15
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.20.0
fonttools	4.44.3
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.9.0
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5
httpx	0.27.2
idna	3.7
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0

Library	Version
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	8.6.0
jupyter_core	5.5.0
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.5.1
langchain	0.2.16
langchain-aws	0.1.17
langchain-community	0.2.16
langchain-core	0.2.38
langchain-text-splitters	0.2.4
langsmith	0.1.114
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5
marshmallow	3.22.0
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2

Library	Version
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.2
nbformat	5.10.4
nest-asyncio	1.5.8
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.7
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.3
partd	1.4.2
pexpect	4.9.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.0.0
ply	3.11
prometheus_client	0.20.0
prompt-toolkit	3.0.41
protobuf	4.25.3
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.9.0
pydantic_core	2.23.2
pygls	1.3.1
Pygments	2.17.2
pyparsing	3.1.1
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7

Library	Version
pytz	2023.3.post1
PyYAML	6.0.2
pyzmq	25.1.1
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.20.0
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.34
stack-data	0.6.3
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
toolz	0.12.1
tornado	6.3.3
traitlets	5.14.0
trino	0.330.0
types-python-dateutil	2.9.0.20240821
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.12
webcolors	24.8.0
webencodings	0.5.1

Library	Version
websocket-client	1.8.0
yarl	1.9.11
zict	3.0.0
zipp	3.20.1

Table 44: Packages in Python 3.11.9 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohappyeyeballs	2.4.0
aiohttp	3.10.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4
attrs	24.2.0
babel	2.16.0
backcall	0.2.0
beautifulsoup4	4.12.3
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.149
botocore	1.34.162
cattrs	24.1.0
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.0
contourpy	1.2.0

Library	Version
cryptography	42.0.8
cycler	0.12.1
dask	2024.8.2
dataclasses-json	0.6.7
debugpy	1.8.0
decorator	5.1.1
deepmerge	2.0
defusedxml	0.7.1
distributed	2024.8.2
docstring-to-markdown	0.15
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.20.0
fonttools	4.44.3
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.9.0
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5
httpx	0.27.2
idna	3.7
importlib_metadata	8.4.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0

Library	Version
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	8.6.0
jupyter_core	5.5.0
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.5
krb5	0.5.1
langchain	0.2.16
langchain-aws	0.1.17
langchain-community	0.2.16
langchain-core	0.2.38
langchain-text-splitters	0.2.4
langsmith	0.1.114
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5
marshmallow	3.22.0
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.2
nbformat	5.10.4
nest-asyncio	1.5.8
notebook_shim	0.2.4
numpy	1.26.4

Library	Version
orjson	3.10.7
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.3
partd	1.4.2
pexpect	4.9.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.0.0
ply	3.11
prometheus_client	0.20.0
prompt-toolkit	3.0.41
protobuf	4.25.3
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.9.0
pydantic_core	2.23.2
pygls	1.3.1
Pygments	2.17.2
pyparsing	3.1.1
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.2
pyzmq	25.1.1
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.20.0

Library	Version
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.6
SQLAlchemy	2.0.34
stack-data	0.6.3
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
toolz	0.12.1
tornado	6.3.3
traitlets	5.14.0
trino	0.330.0
types-python-dateutil	2.9.0.20240821
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	2.2.2
wcwidth	0.2.12
webcolors	24.8.0
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.11
zict	3.0.0
zipp	3.20.1

Python 3.10 Libraries for JupyterLab

This section lists the packages in Python 3.10 JupyterLab Runtimes that ship with ML Runtimes 2024.10.1.

Table 45: Packages in Python 3.10.14 JupyterLab Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
annotated-types	0.7.0
anyio	4.4.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.149
botocore	1.34.162
cattrs	23.2.3
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	5.1.1
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0

Library	Version
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.7
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.14.1

Library	Version
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.27.2
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.13
langchain-aws	0.1.16
langchain-community	0.2.12
langchain-core	0.2.34
langchain-text-splitters	0.2.2
langsmith	0.1.103
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.2
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.6.3
nbconvert	7.2.9
nbdime	4.0.1
nbformat	5.7.3
nest-asyncio	1.5.5
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2

Library	Version
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.16.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0
pydantic_core	2.10.0
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.1.2
raz_client	1.1.0
referencing	0.34.0
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4

Library	Version
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
typeguard	2.13.3
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.2.0
urllib3	2.2.2
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Table 46: Packages in Python 3.10.14 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
annotated-types	0.7.0
anyio	4.4.0

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.149
botocore	1.34.162
cattrs	23.2.3
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	5.1.1
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0

Library	Version
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.7
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.14.1
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.27.2
kerberos	1.3.1

Library	Version
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.13
langchain-aws	0.1.16
langchain-community	0.2.12
langchain-core	0.2.34
langchain-text-splitters	0.2.2
langsmith	0.1.103
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.2
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.6.3
nbconvert	7.2.9
nbdime	4.0.1
nbformat	5.7.3
nest-asyncio	1.5.5
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.16.0
prompt-toolkit	3.0.38

Library	Version
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0
pydantic_core	2.10.0
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.1.2
raz_client	1.1.0
referencing	0.34.0
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3

Library	Version
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
typeguard	2.13.3
typing-inspect	0.9.0
typing_extensions	4.12.2
tzdata	2023.3
tzlocal	5.2
uri-template	1.2.0
urllib3	2.2.2
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Python 3.9 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2024.10.1.

Table 47: Packages in Python 3.9.19 JupyterLab Runtimes - Standard

Library	Version
aiohappyeyeballs	2.3.4
aiohttp	3.10.0
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argcomplete	2.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
async-lru	2.0.4

Library	Version
async-timeout	4.0.3
attrs	23.2.0
Babel	2.15.0
backcall	0.2.0
beautifulsoup4	4.12.3
bitarray	2.6.0
bleach	6.1.0
boto3	1.34.149
botocore	1.34.162
cattrs	23.2.3
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2024.7.1
dataclasses-json	0.6.7
debugpy	1.6.7
decorator	5.1.1
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.7.1
docstring-to-markdown	0.15
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.20.0
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.6.1
gitdb	4.0.11
GitPython	3.1.43

Library	Version
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5
httpx	0.27.0
idna	3.7
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.13
langchain-aws	0.1.16

Library	Version
langchain-community	0.2.12
langchain-core	0.2.34
langchain-text-splitters	0.2.2
langsmith	0.1.95
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5
marshmallow	3.21.3
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.1
nbformat	5.10.4
nest-asyncio	1.5.5
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.6
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus_client	0.20.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2

Library	Version
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0
pydantic_core	2.10.0
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	25.1.2
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.19.1
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.5
SQLAlchemy	2.0.31
stack-data	0.6.2
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3

Library	Version
traitlets	5.9.0
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	1.26.19
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Table 48: Packages in Python 3.9.19 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohappyeyeballs	2.3.4
aiohttp	3.10.0
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argcomplete	2.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.2.0
Babel	2.15.0
backcall	0.2.0

Library	Version
beautifulsoup4	4.12.3
bitarray	2.6.0
bleach	6.1.0
boto3	1.34.149
botocore	1.34.162
cattrs	23.2.3
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2024.7.1
dataclasses-json	0.6.7
debugpy	1.6.7
decorator	5.1.1
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.7.1
docstring-to-markdown	0.15
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.20.0
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.6.1
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5

Library	Version
httpx	0.27.0
idna	3.7
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.13
langchain-aws	0.1.16
langchain-community	0.2.12
langchain-core	0.2.34
langchain-text-splitters	0.2.2
langsmith	0.1.95

Library	Version
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5
marshmallow	3.21.3
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.1
nbformat	5.10.4
nest-asyncio	1.5.5
notebook_shim	0.2.4
numpy	1.26.4
orjson	3.10.6
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.1
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus_client	0.20.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0

Library	Version
pydantic_core	2.10.0
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	25.1.2
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.19.1
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.5
SQLAlchemy	2.0.31
stack-data	0.6.2
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing-inspect	0.9.0

Library	Version
typing_extensions	4.11.0
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0
urllib3	1.26.19
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with ML Runtimes 2024.10.1.

Table 49: Packages in Python 3.8.19 JupyterLab Runtimes - Standard

Library	Version
aiohappyeyeballs	2.3.4
aiohttp	3.10.0
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argcomplete	2.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.2.0
Babel	2.15.0
backcall	0.2.0
backports.zoneinfo	0.2.1
beautifulsoup4	4.12.3
bitarray	2.6.0
bleach	6.1.0

Library	Version
boto3	1.34.149
botocore	1.34.162
cattrs	23.2.3
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2023.5.0
dataclasses-json	0.6.7
debugpy	1.6.7
decorator	5.1.1
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2023.5.0
docstring-to-markdown	0.15
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.20.0
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.6.1
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5
httpx	0.27.0
idna	3.7
importlib-resources	5.12.0

Library	Version
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.13
langchain-aws	0.1.16
langchain-community	0.2.12
langchain-core	0.2.34
langchain-text-splitters	0.2.2
langsmith	0.1.95
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5

Library	Version
marshmallow	3.21.3
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.1
nbformat	5.10.4
nest-asyncio	1.5.5
notebook_shim	0.2.4
numpy	1.24.4
orjson	3.10.6
overrides	7.7.0
packaging	23.2
pandas	2.0.3
pandocfilters	1.5.1
parso	0.8.3
partd	1.4.1
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
ply	3.11
prometheus_client	0.20.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0
pydantic_core	2.10.0
pygls	1.3.1

Library	Version
Pygments	2.15.1
pyparsing	3.0.9
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	25.1.2
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.19.1
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.5
SQLAlchemy	2.0.31
stack-data	0.6.2
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3

Library	Version
tzlocal	5.2
uri-template	1.3.0
urllib3	1.26.19
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Table 50: Packages in Python 3.8.19 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohappyeyeballs	2.3.4
aiohttp	3.10.0
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.7.0
anyio	4.4.0
argcomplete	2.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.2.0
Babel	2.15.0
backcall	0.2.0
backports.zoneinfo	0.2.1
beautifulsoup4	4.12.3
bitarray	2.6.0
bleach	6.1.0
boto3	1.34.149
botocore	1.34.162

Library	Version
cattrs	23.2.3
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cryptography	42.0.8
cycler	0.11.0
dask	2023.5.0
dataclasses-json	0.6.7
debugpy	1.6.7
decorator	5.1.1
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2023.5.0
docstring-to-markdown	0.15
entrypoints	0.4
exceptiongroup	1.2.2
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.20.0
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.6.1
gitdb	4.0.11
GitPython	3.1.43
greenlet	3.0.3
gssapi	1.8.3
h11	0.14.0
httpcore	1.0.5
httpx	0.27.0
idna	3.7
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0

Library	Version
ipykernel	6.23.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.4
jmespath	1.0.1
json5	0.9.25
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	3.0.0
jsonschema	4.23.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.5
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.14.2
jupyter_server_terminals	0.5.3
jupyterlab	4.2.1
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.27.3
kerberos	1.3.1
kiwisolver	1.4.4
krb5	0.5.1
langchain	0.2.13
langchain-aws	0.1.16
langchain-community	0.2.12
langchain-core	0.2.34
langchain-text-splitters	0.2.2
langsmith	0.1.95
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.5
marshmallow	3.21.3
matplotlib	3.7.0

Library	Version
matplotlib-inline	0.1.3
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.10.0
nbconvert	7.16.4
nbdime	4.0.1
nbformat	5.10.4
nest-asyncio	1.5.5
notebook_shim	0.2.4
numpy	1.24.4
orjson	3.10.6
overrides	7.7.0
packaging	23.2
pandas	2.0.3
pandocfilters	1.5.1
parso	0.8.3
partd	1.4.1
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
ply	3.11
prometheus_client	0.20.0
prompt-toolkit	3.0.38
protobuf	4.25.3
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.22
pydantic	2.4.0
pydantic_core	2.10.0
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9

Library	Version
pyspnego	0.11.0
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	25.1.2
raz_client	1.1.0
referencing	0.35.1
requests	2.32.3
requests-kerberos	0.15.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.19.1
s3transfer	0.10.2
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.1
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.5
SQLAlchemy	2.0.31
stack-data	0.6.2
tblib	3.0.0
tenacity	8.5.0
terminado	0.18.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.3.0
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
trino	0.330.0
types-python-dateutil	2.9.0.20240316
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
tzlocal	5.2
uri-template	1.3.0

Library	Version
urllib3	1.26.19
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.8.0
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with ML Runtimes 2024.10.1.

Table 51: Packages in Python 3.7.17 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3

Library	Version
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.41
idna	3.7
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.3
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0

Library	Version
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1

Library	Version
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

Table 52: Packages in Python 3.7.17 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0

Library	Version
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.41
idna	3.7
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.3
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4

Library	Version
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0

Library	Version
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

R 4.4 Libraries

This section lists the R 4.4 libraries that ship with ML Runtimes Runtimes 2024.10.1.

Table 53: R 4.4 Libraries for Workbench

Library	Version
askpass	1.2.0
bases	4.4.0
bitopss	1.0-7
boots	1.3-30
brews	1.0-10
Cairos	1.2-0
caToolss	1.18.2
cdsw	2.1
class	7.3-22
cli	3.6.2
cluster	2.1.6
codetools	0.2-20
colorspace	2.1-0
compiler	4.4.0

Library	Version
curl	5.2.1
datasets	4.4.0
fansi	1.0.6
farver	2.1.1
foreign	0.8-86
ggplot	23.5.1
glue	1.7.0
graphics	4.4.0
grDevices	4.4.0
grid	4.4.0
gtable	0.3.5
httr	1.4.7
isoband	0.2.7
jsonlite	1.8.8
KernSmooth	2.23-22
labeling	0.4.3
lattice	0.22-6
lifecycle	1.0.4
magrittr	2.0.3
MASS	7.3-60.2
Matrix	1.7-0
methods	4.4.0
mgcv	1.9-1
mime	0.12
munsel	l0.5.1
nlme	3.1-164
nnet	7.3-19
openssl	2.1.2
parallel	4.4.0
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8
R	62.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.9
rlang	1.1.3
Rook	1.2
rpart	4.1.23
Rserve	1.7-0

Library	Version
scales	1.3.0
spatial	7.3-17
splines	4.4.0
stats	4.4.0

Table 54: Packages in Python 3.8.10 PBJ R 4.4 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0
certifi	2024.7.4
cffi	1.16.0
charset-normalizer	3.3.2
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fqdn	1.5.1
idna	3.7
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.1
Jinja2	3.1.3
jsonpointer	2.4
jsonschema	4.22.0
jsonschema-specifications	2023.12.1

Library	Version
jupyter-client	7.4.4
jupyter-core	4.12.0
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-server	2.14.1
jupyter-server-terminals	0.5.3
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook-shim	0.2.3
overrides	7.7.0
packaging	23.2
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pkgutil-resolve-name	1.3.10
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.22
Pygments	2.15.1
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0.1
pyzmq	26.0.3
referencing	0.35.1
requests	2.32.3
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1

Library	Version
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.3
types-python-dateutil	2.9.0.20240316
typing-extensions	4.3.0
uri-template	1.3.0
urllib3	1.26.19
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.8.0
zipp	3.15.0

ML Runtimes 2024.05.2

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2024.05.2

Note:

Nvidia GPU Editions with a Workbench Editor come with CUDA 11.4.1 and cuDNN 8.2 versions preinstalled.

Nvidia GPU Editions with JupyterLab or a PBJ Workbench Editor and with Python 3.7 come with CUDA 11.8.0 and cuDNN 8.7 versions preinstalled.

All the other Nvidia GPU Edition Runtimes come with CUDA 12.3 and cuDNN 9.0 versions preinstalled.

Spark and Python Supportability

Specific Python versions are compatible with specific Spark versions. For detailed information about the compatibility levels for various Python versions across different Spark releases, see Spark Python Supportability Matrix.

Python 3.10 Libraries for Conda

This section lists the packages in Python 3.10 Conda Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 55: Packages in Python 3.10.13 Conda Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.94
botocore	1.34.94
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4

Library	Version
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.4
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.11.2
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0

Library	Version
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.6.3
nbconvert	7.2.9
nbdime	4.0.1
nbformat	5.7.3
nest-asyncio	1.5.5
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11

Library	Version
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.2.1

Library	Version
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typeguard	2.13.3
typing-inspect	0.9.0
typing_extensions	4.5.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Python 3.11 Libraries for Workbench

This section lists the packages in Python 3.11 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 56: Packages in Python 3.11.6 PBJ Workbench Runtimes - Standard

Library	Version
anyio	4.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
certifi	2023.7.22
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.0
contourpy	1.2.0

Library	Version
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.26.0
ipython	8.17.2
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.3
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.5.0
jupyter_server	2.11.2
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.5
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.6

Library	Version
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pillow	10.3.0
platformdirs	4.0.0
prometheus-client	0.18.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.16.1
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	24.0.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.0
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1

Library	Version
tornado	6.3.3
traitlets	5.13.0
types-python-dateutil	2.8.19.14
tzdata	2023.3
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.10
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4

Table 57: Packages in Python 3.11.6 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	4.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
certifi	2023.7.22
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3

Library	Version
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.26.0
ipython	8.17.2
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.3
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.5.0
jupyter_server	2.11.2
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.5
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.6
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0

Library	Version
pillow	10.3.0
platformdirs	4.0.0
prometheus-client	0.18.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.16.1
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	24.0.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.0
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.13.0
types-python-dateutil	2.8.19.14
tzdata	2023.3
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.10
webcolors	1.13

Library	Version
webencodings	0.5.1
websocket-client	1.6.4

Python 3.10 Libraries for Workbench

This section lists the packages in Python 3.10 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 58: Packages in Python 3.10.13 Workbench Runtimes - Standard

Library	Version
asttokens	2.2.1
backcall	0.2.0
bitarray	2.7.3
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
idna	3.2
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.2
pandas	2.1.4
parso	0.8.3

Library	Version
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5

Table 59: Packages in Python 3.10.13 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22

Library	Version
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.3
jsonpointer	2.3
jsonschema	4.18.0
jsonschema-specifications	2023.12.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.11.2
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0

Library	Version
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.25.2
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.0.2
referencing	0.35.1
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1

Library	Version
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.9.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Table 60: Packages in Python 3.10.13 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0

Library	Version
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.3
jsonpointer	2.3
jsonschema	4.18.0
jsonschema-specifications	2023.12.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.11.2
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.25.2
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0

Library	Version
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.0.2
referencing	0.35.1
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.9.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.6

Library	Version
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 61: Packages in Python 3.9.18 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
idna	3.2
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.19.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5

Library	Version
numpy	1.25.2
packaging	23.2
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
typing_extensions	4.11.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 62: Packages in Python 3.9.18 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0

Library	Version
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
idna	3.2
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.19.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.2
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5

Library	Version
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
typing_extensions	4.11.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 63: Packages in Python 3.9.18 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4

Library	Version
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0
fqdn	1.5.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.1
Jinja2	3.1.3
jsonpointer	2.4
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.4
jupyter_core	4.12.0
jupyter_server	2.11.2
jupyter_server_terminals	0.5.3
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2

Library	Version
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
referencing	0.35.1
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.3
types-python-dateutil	2.9.0.20240316

Library	Version
typing_extensions	4.3.0
tzdata	2023.3
uri-template	1.3.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 64: Packages in Python 3.9.18 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0

Library	Version
fqdn	1.5.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.1
Jinja2	3.1.3
jsonpointer	2.4
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.4
jupyter_core	4.12.0
jupyter_server	2.11.2
jupyter_server_terminals	0.5.3
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0

Library	Version
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
referencing	0.35.1
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.3
types-python-dateutil	2.9.0.20240316
typing_extensions	4.3.0
tzdata	2023.3
uri-template	1.3.0
urllib3	1.26.18

Library	Version
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 65: Packages in Python 3.8.18 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
idna	3.2
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
jedi	0.19.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0

Library	Version
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.4
packaging	23.2
pandas	2.0.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
typing_extensions	4.11.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 66: Packages in Python 3.8.18 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0

Library	Version
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
idna	3.2
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
jedi	0.19.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.4
packaging	23.2
pandas	2.0.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2

Library	Version
pure-sasl	0.6.2
py4j	0.10.9.5
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
typing_extensions	4.11.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 67: Packages in Python 3.8.18 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1

Library	Version
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0
fqdn	1.5.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.1
Jinja2	3.1.3
jsonpointer	2.4
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.4
jupyter_core	4.12.0
jupyter_server	2.11.2
jupyter_server_terminals	0.5.3
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4

Library	Version
notebook_shim	0.2.3
numpy	1.24.4
overrides	7.7.0
packaging	23.2
pandas	2.0.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
referencing	0.35.1
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1

Library	Version
tornado	6.3.3
traitlets	5.14.3
types-python-dateutil	2.9.0.20240316
typing_extensions	4.3.0
tzdata	2023.3
uri-template	1.3.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 68: Packages in Python 3.8.18 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1

Library	Version
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0
fqdn	1.5.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.1
Jinja2	3.1.3
jsonpointer	2.4
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.4
jupyter_core	4.12.0
jupyter_server	2.11.2
jupyter_server_terminals	0.5.3
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
overrides	7.7.0

Library	Version
packaging	23.2
pandas	2.0.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	26.0.3
referencing	0.35.1
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.3
types-python-dateutil	2.9.0.20240316

Library	Version
typing_extensions	4.3.0
tzdata	2023.3
uri-template	1.3.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	24.6.0
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 69: Packages in Python 3.7.16 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5

Library	Version
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.8.0

Table 70: Packages in Python 3.7.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3

Library	Version
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.8.0

Table 71: Packages in Python 3.7.16 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0

Library	Version
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.1.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10

Library	Version
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2

Library	Version
zipp	3.7.0

Table 72: Packages in Python 3.7.16 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.1.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0

Library	Version
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1

Library	Version
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-reflect	2.11.12
org.scala-lang:scala-compiler	2.11.12
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2
org.scalameta:scalameta	1.0.0

Python 3.11 Libraries for JupyterLab

This section lists the packages in Python 3.11 JupyterLab Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 73: Packages in Python 3.11.6 JupyterLab Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
annotated-types	0.6.0
anyio	4.1.0
argon2-cffi	23.1.0

Library	Version
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4
attrs	23.1.0
Babel	2.13.1
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.2
certifi	2023.11.17
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.8.0
decorator	5.1.1
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.13
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.11
GitPython	3.1.41

Library	Version
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.6
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.14
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	8.6.0
jupyter_core	5.5.0
jupyter_server	2.11.2
jupyter_server_terminals	0.4.4
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.25.2
kerberos	1.3.1
kiwisolver	1.4.5
langchain	0.1.19
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56

Library	Version
locket	1.0.0
lsprotocol	2023.0.0
MarkupSafe	2.1.3
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbdime	4.0.1
nbformat	5.9.2
nest-asyncio	1.5.8
notebook_shim	0.2.3
numpy	1.26.4
orjson	3.10.3
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.9.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.0.0
ply	3.11
prometheus-client	0.19.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
pydantic	2.7.1
pydantic_core	2.18.2

Library	Version
pygls	1.2.1
Pygments	2.17.2
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	25.1.1
referencing	0.31.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.1
s3transfer	0.10.1
Send2Trash	1.8.2
six	1.16.0
smmap	5.0.1
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.5
SQLAlchemy	2.0.30
stack-data	0.6.3
tblib	3.0.0
tenacity	8.3.0
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.2.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.14.0
types-python-dateutil	2.8.19.14
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.3.0
urllib3	2.1.0

Library	Version
wcwidth	0.2.12
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Table 74: Packages in Python 3.11.6 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
annotated-types	0.6.0
anyio	4.1.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4
attrs	23.1.0
Babel	2.13.1
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.2
certifi	2023.11.17
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6

Library	Version
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.8.0
decorator	5.1.1
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.13
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.11
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.6
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.14
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.4
jsonschema	4.20.0

Library	Version
jsonschema-specifications	2023.11.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	8.6.0
jupyter_core	5.5.0
jupyter_server	2.11.2
jupyter_server_terminals	0.4.4
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.25.2
kerberos	1.3.1
kiwisolver	1.4.5
langchain	0.1.19
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.0
MarkupSafe	2.1.3
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbdime	4.0.1
nbformat	5.9.2
nest-asyncio	1.5.8
notebook_shim	0.2.3
numpy	1.26.4
orjson	3.10.3
overrides	7.4.0

Library	Version
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.9.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.0.0
ply	3.11
prometheus-client	0.19.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
pydantic	2.7.1
pydantic_core	2.18.2
pygls	1.2.1
Pygments	2.17.2
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	25.1.1
referencing	0.31.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.1
s3transfer	0.10.1
Send2Trash	1.8.2
six	1.16.0
smmap	5.0.1
sniffio	1.3.0

Library	Version
sortedcontainers	2.4.0
soupsieve	2.5
SQLAlchemy	2.0.30
stack-data	0.6.3
tblib	3.0.0
tenacity	8.3.0
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.2.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.14.0
types-python-dateutil	2.8.19.14
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.12
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Python 3.10 Libraries for JupyterLab

This section lists the packages in Python 3.10 JupyterLab Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 75: Packages in Python 3.10.13 JupyterLab Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
anyio	3.6.2

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.94
botocore	1.34.94
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1

Library	Version
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.4
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.11.2
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20

Library	Version
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.6.3
nbconvert	7.2.9
nbdime	4.0.1
nbformat	5.7.3
nest-asyncio	1.5.5
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2

Library	Version
py4j	0.10.9.5
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typeguard	2.13.3
typing-inspect	0.9.0

Library	Version
typing_extensions	4.5.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Table 76: Packages in Python 3.10.13 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.94
botocore	1.34.94
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1

Library	Version
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.4
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.3

Library	Version
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.11.2
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.6.3
nbconvert	7.2.9
nbdime	4.0.1

Library	Version
nbformat	5.7.3
nest-asyncio	1.5.5
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0

Library	Version
s3transfer	0.10.1
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typeguard	2.13.3
typing-inspect	0.9.0
typing_extensions	4.5.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Python 3.9 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with Cloudera Machnine Learning Runtimes 2024.05.2.

Table 77: Packages in Python 3.9.18 JupyterLab Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.6.0
anyio	4.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.3
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.6.0
distributed	2024.5.0

Library	Version
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.15.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	2.10
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.5
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.11.2

Library	Version
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.1.2
jupyterlab_git	0.50.0
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.1
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	4.0.1
nbformat	5.3.0
nest-asyncio	1.5.5
notebook_shim	0.2.3
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.4.3
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5

Library	Version
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.20
pydantic	2.0
pydantic_core	2.0.1
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
sortedcontainers	2.4.0
soupsieve	2.3.2
SQLAlchemy	2.0.30
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.9.2

Library	Version
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.1.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Table 78: Packages in Python 3.9.18 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.6.0
anyio	4.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
async-lru	2.0.4
async-timeout	4.0.3

Library	Version
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.3
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.6.0
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.15.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0

Library	Version
httpcore	1.0.4
httpx	0.27.0
idna	2.10
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.5
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.11.2
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.1.2
jupyterlab_git	0.50.0
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0

Library	Version
lsprotocol	2023.0.1
MarkupSafe	2.1.1
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	4.0.1
nbformat	5.3.0
nest-asyncio	1.5.5
notebook_shim	0.2.3
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.4.3
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.20
pydantic	2.0
pydantic_core	2.0.1
pygls	1.3.1

Library	Version
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
sortedcontainers	2.4.0
soupsieve	2.3.2
SQLAlchemy	2.0.30
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.1.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5

Library	Version
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 79: Packages in Python 3.8.18 JupyterLab Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.6.0
anyio	4.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.3
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
click	8.1.7

Library	Version
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2023.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.6.0
distributed	2023.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.15.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	2.10
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1

Library	Version
json5	0.9.5
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.11.2
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.1.2
jupyterlab_git	0.50.0
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.1
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	4.0.1
nbformat	5.3.0

Library	Version
nest-asyncio	1.5.5
notebook_shim	0.2.3
numpy	1.24.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.0.3
pandocfilters	1.4.3
parso	0.8.3
partd	1.4.1
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
ply	3.11
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.20
pydantic	2.0
pydantic_core	2.0.1
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1

Library	Version
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
sortedcontainers	2.4.0
soupsieve	2.3.2
SQLAlchemy	2.0.30
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.1.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Table 80: Packages in Python 3.8.18 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1

Library	Version
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.6.0
anyio	4.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.3
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2023.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.6.0
distributed	2023.5.0
docstring-to-markdown	0.12

Library	Version
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.15.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	2.10
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.5
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter_client	7.4.4
jupyter_core	5.3.0
jupyter_server	2.11.2
jupyter_server_terminals	0.5.3

Library	Version
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.1.2
jupyterlab_git	0.50.0
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.1
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	4.0.1
nbformat	5.3.0
nest-asyncio	1.5.5
notebook_shim	0.2.3
numpy	1.24.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.0.3
pandocfilters	1.4.3
parso	0.8.3
partd	1.4.1
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0

Library	Version
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
ply	3.11
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.20
pydantic	2.0
pydantic_core	2.0.1
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.3
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
sortedcontainers	2.4.0
soupsieve	2.3.2
SQLAlchemy	2.0.30
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.9.2

Library	Version
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.1.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 81: Packages in Python 3.7.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1

Library	Version
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.41
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.3
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0

Library	Version
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0

Library	Version
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

Table 82: Packages in Python 3.7.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1 + cuda11.8
libcudnn8-dev	8.7.0.84-1 + cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3

Library	Version
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.41
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.3
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6

Library	Version
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.15.1
pyparsing	3.0.9

Library	Version
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

R 4.4 Libraries

This section lists the R 4.4 libraries that ship with Cloudera Machine Learning Runtimes 2024.05.2.

Table 83: R 4.4 Libraries for Workbench

Library	Version
askpass	1.2.0

Library	Version
base	4.4.0
bitops	1.0-7
boot	1.3-30
brew	1.0-10
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-22
cli	3.6.2
cluster	2.1.6
codetools	0.2-20
colorspace	2.1-0
compiler	4.4.0
curl	5.2.1
datasets	4.4.0
fansi	1.0.6
farver	2.1.1
foreign	0.8-86
ggplot2	3.5.1
glue	1.7.0
graphics	4.4.0
grDevices	4.4.0
grid	4.4.0
gtable	0.3.5
httr	1.4.7
isoband	0.2.7
jsonlite	1.8.8
KernSmooth	2.23-22
labeling	0.4.3
lattice	0.22-6
lifecycle	1.0.4
magrittr	2.0.3
MASS	7.3-60.2
Matrix	1.7-0
methods	4.4.0
mgcv	1.9-1
mime	0.12
munsell	0.5.1
nlme	3.1-164

Library	Version
nnet	7.3-19
openssl	2.1.2
parallel	4.4.0
pillar	1.9.0
pkgconfig	2.0.30
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.9
rlang	1.1.3
Rook	1.2
rpart	4.1.23
Rserve	1.7-0
scales	1.3.0
spatial	7.3-17
splines	4.4.0
stats	4.4.0
stats4	4.4.0
survival	3.5-8
sys	3.4.2
tcltk	4.4.0
tibble	3.2.1
tools	4.4.0
utf8	1.2.4
utils	4.4.0
vctrs	0.6.5
viridisLite	0.4.2
withr	3.0.0

Table 84: Packages in Python 3.8.10 PBJ R 4.4 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.2.1
attrs	23.2.0
backcall	0.2.0
beautifulsoup4	4.12.2

Library	Version
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fqdn	1.5.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.1
Jinja2	3.1.3
jsonpointer	2.4
jsonschema	4.22.0
jsonschema-specifications	2023.12.1
jupyter-client	7.4.4
jupyter-core	4.12.0
jupyter-events	0.10.0
jupyter-kernel-gateway	2.5.2
jupyter-server	2.11.2
jupyter-server-terminals	0.5.3
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4

Library	Version
notebook	6.5.4
notebook-shim	0.2.3
overrides	7.7.0
packaging	23.2
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pkgutil-resolve-name	1.3.10
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.15.1
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0.1
pyzmq	26.0.3
referencing	0.35.1
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.1
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.3
types-python-dateutil	2.9.0.20240316
typing-extensions	4.3.0
uri-template	1.3.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	24.6.0

Library	Version
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

ML Runtimes 2024.05.1

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2024.05.1

Note:

Nvidia GPU Editions with a Workbench Editor come with CUDA 11.4.1 and cuDNN 8.2 versions preinstalled.

Nvidia GPU Editions with JupyterLab or a PBJ Workbench Editor and with Python 3.7 come with CUDA 11.8.0 and cuDNN 8.7 versions preinstalled.

All the other Nvidia GPU Edition Runtimes come with CUDA 12.3 and cuDNN 9.0 versions preinstalled.

Spark and Python Supportability

Specific Python versions are compatible with specific Spark versions. For detailed information about the compatibility levels for various Python versions across different Spark releases, see Spark Python Supportability Matrix.

Related Information

Adding new ML Runtimes

Archive Runtimes

Python 3.10 Libraries for Conda

This section lists the packages in Python 3.10 Conda Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 85: Packages in Python 3.10.13 Conda Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0

Library	Version
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.94
botocore	1.34.94
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4

Library	Version
httpx	0.27.0
idna	3.4
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter-ydoc	2.0.1
jupyter_client	7.4.4
jupyter_collaboration	2.0.5
jupyter_core	5.3.0
jupyter_server	2.7.0
jupyter_server_fileid	0.9.0
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56

Library	Version
locket	1.0.0
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.6.3
nbconvert	7.2.9
nbdime	4.0.1
nbformat	5.7.3
nest-asyncio	1.5.5
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
pycrdt	0.8.16
pycrdt-websocket	0.12.7

Library	Version
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typeguard	2.13.3
typing-inspect	0.9.0
typing_extensions	4.5.0
tzdata	2023.3

Library	Version
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Python 3.11 Libraries for Workbench

This section lists the packages in Python 3.11 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 86: Packages in Python 3.11.6 PBJ Workbench Runtimes - Standard

Library	Version
anyio	4.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
certifi	2023.7.22
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3

Library	Version
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.26.0
ipython	8.17.2
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.3
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.5.0
jupyter_server	2.10.1
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.5
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.6
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0

Library	Version
pillow	10.3.0
platformdirs	4.0.0
prometheus-client	0.18.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.16.1
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	24.0.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.0
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.13.0
types-python-dateutil	2.8.19.14
tzdata	2023.3
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.10
webcolors	1.13

Library	Version
webencodings	0.5.1
websocket-client	1.6.4

Table 87: Packages in Python 3.11.6 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	4.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
certifi	2023.7.22
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.26.0
ipython	8.17.2
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1

Library	Version
Jinja2	3.1.3
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.5.0
jupyter_server	2.10.1
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.5
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.6
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pillow	10.3.0
platformdirs	4.0.0
prometheus-client	0.18.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2

Library	Version
py4j	0.10.9.5
pycparser	2.21
Pygments	2.16.1
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	24.0.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.0
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.13.0
types-python-dateutil	2.8.19.14
tzdata	2023.3
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.10
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4

Python 3.10 Libraries for Workbench

This section lists the packages in Python 3.10 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 88: Packages in Python 3.10.13 Workbench Runtimes - Standard

Library	Version
asttokens	2.2.1
backcall	0.2.0
bitarray	2.7.3
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
idna	3.2
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.2
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2

Library	Version
pure-sasl	0.6.2
py4j	0.10.9.5
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5

Table 89: Packages in Python 3.10.13 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4

Library	Version
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.3
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3

Library	Version
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.0.2
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.9.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1

Library	Version
websocket-client	1.5.1

Table 90: Packages in Python 3.10.13 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.3

Library	Version
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1

Library	Version
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.0.2
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.9.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 91: Packages in Python 3.9.18 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0

Library	Version
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
idna	3.2
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.19.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.2
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5

Library	Version
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
typing_extensions	4.11.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 92: Packages in Python 3.9.18 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1 + cuda11.4
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
idna	3.2
importlib-resources	5.12.0
importlib_metadata	7.1.0

Library	Version
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.19.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.2
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
typing_extensions	4.11.0
tzdata	2023.3

Library	Version
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 93: Packages in Python 3.9.18 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.1.3
jsonschema	4.4.0
jupyter-client	7.1.2

Library	Version
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
packaging	23.1
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0

Library	Version
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.1.1
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 94: Packages in Python 3.9.18 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0

Library	Version
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.1.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
packaging	23.1
pandas	2.1.4

Library	Version
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.1.1
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 95: Packages in Python 3.8.18 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
idna	3.2
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
jedi	0.19.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.4
packaging	23.2
pandas	2.0.3
parso	0.8.3
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
typing_extensions	4.11.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 96: Packages in Python 3.8.18 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn	8 8.2.2.26-1 + cuda11.4
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7

Library	Version
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.43.0
idna	3.2
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
jedi	0.19.1
jupyter_client	7.4.4
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.4
packaging	23.2
pandas	2.0.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	25.1.2

Library	Version
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.3
traitlets	5.9.0
typing_extensions	4.11.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 97: Packages in Python 3.8.18 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0
idna	3.3
importlib-metadata	4.11.0

Library	Version
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.1.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandas	2.0.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2

Library	Version
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.1.1
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 98: Packages in Python 3.8.18 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0

Library	Version
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.43.0
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.1.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0

Library	Version
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandas	2.0.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.1.1

Library	Version
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 99: Packages in Python 3.7.16 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1

Library	Version
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.8.0

Table 100: Packages in Python 3.7.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1 + cuda11.4
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0

Library	Version
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.8.0

Table 101: Packages in Python 3.7.16 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0

Library	Version
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.1.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0

Library	Version
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Table 102: Packages in Python 3.7.16 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.1.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1

Library	Version
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1

Library	Version
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-reflect	2.11.12
org.scala-lang:scala-compiler	2.11.12
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2
org.scalameta:scalameta	1.0.0

Python 3.11 Libraries for JupyterLab

This section lists the packages in Python 3.11 JupyterLab Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 103: Packages in Python 3.11.6 JupyterLab Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
annotated-types	0.6.0
anyio	4.1.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1

Library	Version
async-lru	2.0.4
attrs	23.1.0
Babel	2.13.1
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.2
certifi	2023.11.17
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.8.0
decorator	5.1.1
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.13
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.11
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4

Library	Version
httpx	0.27.0
idna	3.6
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.14
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter-ydoc	2.0.1
jupyter_client	8.6.0
jupyter_collaboration	2.0.5
jupyter_core	5.5.0
jupyter_server	2.11.1
jupyter_server_fileid	0.9.0
jupyter_server_terminals	0.4.4
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.25.2
kerberos	1.3.1
kiwisolver	1.4.5
langchain	0.1.19
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56

Library	Version
locket	1.0.0
lsprotocol	2023.0.0
MarkupSafe	2.1.3
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbdime	4.0.1
nbformat	5.9.2
nest-asyncio	1.5.8
notebook_shim	0.2.3
numpy	1.26.4
orjson	3.10.3
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.9.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.0.0
ply	3.11
prometheus-client	0.19.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
pycrdt	0.8.16
pycrdt-websocket	0.12.7

Library	Version
pydantic	2.7.1
pydantic_core	2.18.2
pygls	1.2.1
Pygments	2.17.2
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	25.1.1
referencing	0.31.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.1
s3transfer	0.10.1
Send2Trash	1.8.2
six	1.16.0
smmap	5.0.1
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.5
SQLAlchemy	2.0.30
stack-data	0.6.3
tblib	3.0.0
tenacity	8.3.0
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.2.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.14.0
types-python-dateutil	2.8.19.14
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3

Library	Version
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.12
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Table 104: Packages in Python 3.11.6 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
annotated-types	0.6.0
anyio	4.1.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4
attrs	23.1.0
Babel	2.13.1
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.2
certifi	2023.11.17
cffi	1.16.0
charset-normalizer	3.3.2
click	8.1.7

Library	Version
cloudpickle	3.0.0
colorama	0.4.6
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.8.0
decorator	5.1.1
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.13
executing	2.0.1
faiss-cpu	1.8.0
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.11
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.6
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.3
jmespath,	1.0.1
json5	0.9.14
jsonpatch	1.33
jsonpath-ng	1.6.1

Library	Version
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter-ydoc	2.0.1
jupyter_client	8.6.0
jupyter_collaboration	2.0.5
jupyter_core	5.5.0
jupyter_server	2.11.1
jupyter_server_fileid	0.9.0
jupyter_server_terminals	0.4.4
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab_git	0.50.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.25.2
kerberos	1.3.1
kiwisolver	1.4.5
langchain	0.1.19
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.0
MarkupSafe	2.1.3
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbdime	4.0.1
nbformat	5.9.2

Library	Version
nest-asyncio	1.5.8
notebook_shim	0.2.3
numpy	1.26.4
orjson	3.10.3
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.9.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	4.0.0
ply	3.11
prometheus-client	0.19.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
pycrdt	0.8.16
pycrdt-websocket	0.12.7
pydantic	2.7.1
pydantic_core	2.18.2
pygls	1.2.1
Pygments	2.17.2
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	25.1.1
referencing	0.31.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4

Library	Version
rfc3986-validator	0.1.1
rpds-py	0.13.1
s3transfer	0.10.1
Send2Trash	1.8.2
six	1.16.0
smmap	5.0.1
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.5
SQLAlchemy	2.0.30
stack-data	0.6.3
tblib	3.0.0
tenacity	8.3.0
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.2.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.14.0
types-python-dateutil	2.8.19.14
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.12
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Python 3.10 Libraries for JupyterLab

This section lists the packages in Python 3.10 JupyterLab Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 105: Packages in Python 3.10.13 JupyterLab Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.94
botocore	1.34.94
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4

Library	Version
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	3.4
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter-ydoc	2.0.1
jupyter_client	7.4.4
jupyter_collaboration	2.0.5
jupyter_core	5.3.0
jupyter_server	2.7.0
jupyter_server_fileid	0.9.0

Library	Version
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.6.3
nbconvert	7.2.9
nbdime	4.0.1
nbformat	5.7.3
nest-asyncio	1.5.5
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5

Library	Version
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
pycrdt	0.8.16
pycrdt-websocket	0.12.7
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0

Library	Version
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typeguard	2.13.3
typing-inspect	0.9.0
typing_extensions	4.5.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Table 106: Packages in Python 3.10.13 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
async-lru	2.0.4
async-timeout	4.0.3

Library	Version
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
boto3	1.34.94
botocore	1.34.94
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.7.1
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.16.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0

Library	Version
httpcore	1.0.4
httpx	0.27.0
idna	3.4
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.11
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter-ydoc	2.0.1
jupyter_client	7.4.4
jupyter_collaboration	2.0.5
jupyter_core	5.3.0
jupyter_server	2.7.0
jupyter_server_field	0.9.0
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.2.2
jupyterlab_git	0.50.0
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1

Library	Version
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.6.3
nbconvert	7.2.9
nbdime	4.0.1
nbformat	5.7.3
nest-asyncio	1.5.5
notebook_shim	0.2.2
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.21
pycrdt	0.8.16

Library	Version
pycrdt-websocket	0.12.7
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.4
SQLAlchemy	2.0.24
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typeguard	2.13.3
typing-inspect	0.9.0
typing_extensions	4.5.0

Library	Version
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
yarl	1.9.4
zict	3.0.0
zipp	3.18.1

Python 3.9 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with Cloudera Machnine Learning Runtimes 2024.05.1.

Table 107: Packages in Python 3.9.18 JupyterLab Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.6.0
anyio	4.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.3

Library	Version
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.6.0
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.15.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	2.1
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0

Library	Version
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.5
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter-ydoc	2.0.1
jupyter_client	7.4.4
jupyter_collaboration	2.0.5
jupyter_core	5.3.0
jupyter_server	2.7.0
jupyter_server_fileid	0.9.1
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.1.2
jupyterlab_git	0.50.0
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.1
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5

Library	Version
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	4.0.1
nbformat	5.3.0
nest-asyncio	1.5.5
notebook_shim	0.2.3
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.4.3
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.2
pycrdt	0.8.16
pycrdt-websocket	0.12.7
pydantic	2
pydantic_core	2.0.1
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7

Library	Version
pytz	2023.3.post1
PyYAML	6
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
sortedcontainers	2.4.0
soupsieve	2.3.2
SQLAlchemy	2.0.30
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.1.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
yarl	1.9.4

Library	Version
zict	3.0.0
zipp	3.15.0

Table 108: Packages in Python 3.9.18 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.6.0
anyio	4.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.3
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0

Library	Version
dask	2024.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.6.0
distributed	2024.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.15.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	2.10
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.5
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1

Library	Version
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter-ydoc	2.0.1
jupyter_client	7.4.4
jupyter_collaboration	2.0.5
jupyter_core	5.3.0
jupyter_server	2.7.0
jupyter_server_field	0.9.1
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.1.2
jupyterlab_git	0.50.0
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.1
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	4.0.1
nbformat	5.3.0
nest-asyncio	1.5.5
notebook_shim	0.2.3

Library	Version
numpy	1.26.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.4.3
parso	0.8.3
partd	1.4.2
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
platformdirs	3.5.0
ply	3.11
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.20
pycrdt	0.8.16
pycrdt-websocket	0.12.7
pydantic	2.0
pydantic_core	2.0.1
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0

Library	Version
s3transfer	0.10.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
sortedcontainers	2.4.0
soupsieve	2.3.2
SQLAlchemy	2.0.30
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.1.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 109: Packages in Python 3.8.18 JupyterLab Runtimes - Standard

Library	Version
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.6.0
anyio	4.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.3
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2023.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.6.0
distributed	2023.5.0

Library	Version
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.15.3
fonttools	4.43.0
fqdn	1.5.1
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	2.1
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.5
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter-ydoc	2.0.1
jupyter_client	7.4.4
jupyter_collaboration	2.0.5

Library	Version
jupyter_core	5.3.0
jupyter_server	2.7.0
jupyter_server_fileid	0.9.1
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.1.2
jupyterlab_git	0.50.0
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.1
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	4.0.1
nbformat	5.3.0
nest-asyncio	1.5.5
notebook_shim	0.2.3
numpy	1.24.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.0.3
pandocfilters	1.4.3
parso	0.8.3

Library	Version
partd	1.4.1
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
ply	3.11
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.2
pycrdt	0.8.16
pycrdt-websocket	0.12.7
pydantic	2
pydantic_core	2.0.1
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
sortedcontainers	2.4.0

Library	Version
soupsieve	2.3.2
SQLAlchemy	2.0.30
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tiktoken	0.6.0
tinycss2	1.1.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Table 110: Packages in Python 3.8.18 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-12-3	12.3.2-1
cuda-libraries-dev-12-3	12.3.2-1
libcudnn9-cuda-12	9.0.0.312-1
libcudnn9-dev-cuda-12	9.0.0.312-1
aiohttp	3.9.5
aiosignal	1.3.1
aiosqlite	0.20.0
annotated-types	0.6.0
anyio	4.3.0
argcomplete	2.0.0

Library	Version
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.1
async-lru	2.0.4
async-timeout	4.0.3
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
boto3	1.34.94
botocore	1.34.94
cattrs	23.2.3
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
click	8.1.7
cloudpickle	3.0.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
dask	2023.5.0
dataclasses-json	0.6.6
debugpy	1.6.7
decorator	4.4.2
deepmerge	1.1.1
defusedxml	0.6.0
distributed	2023.5.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
faiss-cpu	1.8.0
fastjsonschema	2.15.3
fonttools	4.43.0
fqdn	1.5.1

Library	Version
frozenlist	1.4.1
fsspec	2024.3.1
gitdb	4.0.10
GitPython	3.1.41
greenlet	3.0.3
h11	0.14.0
httpcore	1.0.4
httpx	0.27.0
idna	2.1
importlib-resources	5.12.0
importlib_metadata	7.1.0
impyla	0.18.0
ipykernel	6.23.0
ipython	8.10.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.4
Jinja2	3.1.3
jmespath	1.0.1
json5	0.9.5
jsonpatch	1.33
jsonpath-ng	1.6.1
jsonpointer	2.3
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-events	0.10.0
jupyter-lsp	2.2.2
jupyter-server-mathjax	0.2.6
jupyter-ydoc	2.0.1
jupyter_client	7.4.4
jupyter_collaboration	2.0.5
jupyter_core	5.3.0
jupyter_server	2.7.0
jupyter_server_fileid	0.9.1
jupyter_server_terminals	0.5.3
jupyterlab	4.1.5
jupyterlab-lsp	5.1.0
jupyterlab-pygments	0.1.2
jupyterlab_git	0.50.0

Library	Version
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
langchain	0.1.20
langchain-community	0.0.38
langchain-core	0.1.52
langchain-text-splitters	0.0.1
langsmith	0.1.56
locket	1.0.0
lsprotocol	2023.0.1
MarkupSafe	2.1.1
marshmallow	3.21.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
msgpack	1.0.8
multidict	6.0.5
mypy-extensions	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	4.0.1
nbformat	5.3.0
nest-asyncio	1.5.5
notebook_shim	0.2.3
numpy	1.24.4
orjson	3.10.3
overrides	7.7.0
packaging	23.2
pandas	2.0.3
pandocfilters	1.4.3
parso	0.8.3
partd	1.4.1
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
ply	3.11
prometheus-client	0.9.0

Library	Version
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.5
pycparser	2.2
pycrdt	0.8.16
pycrdt-websocket	0.12.7
pydantic	2
pydantic_core	2.0.1
pygls	1.3.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6
pyzmq	25.1.2
referencing	0.34.0
regex	2024.5.10
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.18.0
s3transfer	0.10.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
sortedcontainers	2.4.0
soupsieve	2.3.2
SQLAlchemy	2.0.30
stack-data	0.6.2
tblib	3.0.0
tenacity	8.3.0
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3

Library	Version
tiktoken	0.6.0
tinycss2	1.1.1
tomli	2.0.1
toolz	0.12.1
tornado	6.3.3
traitlets	5.9.0
typing-inspect	0.9.0
typing_extensions	4.11.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
yarl	1.9.4
zict	3.0.0
zipp	3.15.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 111: Packages in Python 3.7.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1

Library	Version
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.41
idna	2.1
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.3
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2

Library	Version
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.2
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6
pyzmq	22.3.0
requests	2.31.0

Library	Version
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

Table 112: Packages in Python 3.7.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1 + cuda11.8
libcudnn8-dev	8.7.0.84-1 + cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0

Library	Version
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.41
idna	2.1
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.3
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0

Library	Version
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.2
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2

Library	Version
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

R 4.4 Libraries

This section lists the R 4.4 libraries that ship with Cloudera Machine Learning Runtimes 2024.05.1.

Table 113: R 4.4 Libraries for Workbench

Library	Version
askpass	1.2.0
base	4.4.0
bitops	1.0-7

Library	Version
boot	1.3-30
brew	1.0-10
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-22
cli	3.6.2
cluster	2.1.6
codetools	0.2-20
colorspace	2.1-0
compiler	4.4.0
curl	5.2.1
datasets	4.4.0
fansi	1.0.6
farver	2.1.1
foreign	0.8-86
ggplot2	3.5.1
glue	1.7.0
graphics	4.4.0
grDevices	4.4.0
grid	4.4.0
gtable	0.3.5
httr	1.4.7
isoband	0.2.7
jsonlite	1.8.8
KernSmooth	2.23-22
labeling	0.4.3
lattice	0.22-6
lifecycle	1.0.4
magrittr	2.0.3
MASS	7.3-60.2
Matrix	1.7-0
methods	4.4.0
mgcv	1.9-1
mime	0.12
munsell	0.5.1
nlme	3.1-164
nnet	7.3-19
openssl	2.1.2

Library	Version
parallel	4.4.0
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.9
rlang	1.1.3
Rook	1.2
rpart	4.1.23
Rserve	1.7-0
scales	1.3.0
spatial	7.3-17
splines	4.4.0
stats	4.4.0
stats4	4.4.0
survival	3.5-8
sys	3.4.2
tcltk	4.4.0
tibble	3.2.1
tools	4.4.0
utf8	1.2.4
utils	4.4.0
vctrs	0.6.5
viridisLite	0.4.2
withr	3.0.0

ML Runtimes 2024.02.1

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2024.02.1

Note: Nvidia GPU Editions with a Workbench Editor come with CUDA 11.4.1 preinstalled.

All other Nvidia GPU Edition Runtimes come with CUDA 11.8.0 preinstalled.

Python 3.10 Libraries for Conda

This section lists the packages in Python 3.10 Conda Runtimes that ship with ML Runtimes 2024.02.1.

Table 114: Packages in Python 3.10 Conda Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.37
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0

Library	Version
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1

Library	Version
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1

Library	Version
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Python 3.11 Libraries for Workbench

This section lists the packages in Python 3.11 Workbench Runtimes that ship with ML Runtimes 2024.02.1.

Table 115: Packages in Python 3.11.6 PBJ Workbench Runtimes - Standard

Library	Version
anyio	4.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
certifi	2023.7.22
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4

Library	Version
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.26.0
ipython	8.17.2
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.2
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.5.0
jupyter_server	2.10.1
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.5
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.6
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandas	2.1.4

Library	Version
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
Pillow	10.1.0
platformdirs	4.0.0
prometheus-client	0.18.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.16.1
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	24.0.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.0
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.13.0
types-python-dateutil	2.8.19.14
tzdata	2023.3
uri-template	1.3.0

Library	Version
urllib3	2.1.0
wcwidth	0.2.10
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4

Table 116: Packages in Python 3.11.6 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	4.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
certifi	2023.7.22
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.26.0
ipython	8.17.2

Library	Version
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.2
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.5.0
jupyter_server	2.10.1
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.5
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.6
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
Pillow	10.1.0
platformdirs	4.0.0
prometheus-client	0.18.0
prompt-toolkit	3.0.41
psutil	5.9.6

Library	Version
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.16.1
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	24.0.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.0
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.13.0
types-python-dateutil	2.8.19.14
tzdata	2023.3
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.10
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4

Python 3.10 Libraries for Workbench

This section lists the packages in Python 3.10 Workbench Runtimes that ship with ML Runtimes 2024.02.1.

Table 117: Packages in Python 3.10.9 Workbench Runtimes - Standard

Library	Version
asttokens	2.2.1
backcall	0.2.0
bitarray	2.7.3
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.1
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2

Library	Version
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5

Table 118: Packages in Python 3.10.9 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4

Library	Version
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.2
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3

Library	Version
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.0.2
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.9.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1

Library	Version
websocket-client	1.5.1

Table 119: Packages in Python 3.10.9 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.2

Library	Version
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1

Library	Version
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	25.0.2
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.9.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2024.02.1.

Table 120: Packages in Python 3.9.11 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0

Library	Version
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.1
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1

Library	Version
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 121: Packages in Python 3.9.11 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0

Library	Version
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.1
pandas	2.1.4
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18

Library	Version
wcwidth	0.2.5
zipp	3.15.0

Table 122: Packages in Python 3.9.11 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2

Library	Version
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
packaging	23.1
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0

Library	Version
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 123: Packages in Python 3.9.11 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1

Library	Version
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
packaging	23.1
pandas	2.1.4
pandocfilters	1.5.0

Library	Version
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with ML Runtimes 2024.02.1.

Table 124: Packages in Python 3.8.16 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.4
packaging	23.1
pandas	2.0.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prompt-toolkit	3.0.38

Library	Version
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 125: Packages in Python 3.8.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4

Library	Version
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.4
packaging	23.1
pandas	2.0.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0

Library	Version
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 126: Packages in Python 3.8.16 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0

Library	Version
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandas	2.0.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9

Library	Version
pyrsistent	0.18.1
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 127: Packages in Python 3.8.16 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0

Library	Version
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4

Library	Version
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandas	2.0.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pytz	2023.3.post1
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
tzdata	2023.3
urllib3	1.26.18
wcwidth	0.2.5

Library	Version
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with ML Runtimes 2024.02.1.

Table 128: Packages in Python 3.7.16 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0

Library	Version
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.8.0

Table 129: Packages in Python 3.7.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1

Library	Version
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.8.0

Table 130: Packages in Python 3.7.16 PBJ Workbench - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22

Library	Version
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3

Library	Version
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Table 131: Packages in Python 3.7.16 PBJ Workbench - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8

Library	Version
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3

Library	Version
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18

Library	Version
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with ML Runtimes 2024.02.1.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-reflect	2.11.12
org.scala-lang:scala-compiler	2.11.12
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2
org.scalameta:scalameta	1.0.0

Python 3.11 Libraries for JupyterLab

This section lists the packages in Python 3.11 JupyterLab Runtimes that ship with ML Runtimes 2024.02.1.

Table 132: Packages in Python 3.11.6 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	4.1.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
Babel	2.13.1
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
cattrs	23.2.2
certifi	2023.11.17

Library	Version
cffi	1.16.0
charset-normalizer	3.3.2
colorama	0.4.6
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
docstring-to-markdown	0.13
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
gitdb	4.0.11
GitPython	3.1.40
idna	3.6
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.2
json5	0.9.14
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-lsp	2.2.1
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.5
jupyter_client	8.6.0
jupyter_core	5.5.0
jupyter_server	2.11.1
jupyter_server_fileid	0.9.0
jupyter_server_terminals	0.4.4
jupyter_server_ydoc	0.6.1

Library	Version
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.25.2
kerberos	1.3.1
kiwisolver	1.4.5
lsprotocol	2023.0.0
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbdime	3.2.1
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.9.0
pickleshare	0.7.5
Pillow	10.1.0
platformdirs	4.0.0
prometheus-client	0.19.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pygls	1.2.0

Library	Version
Pygments	2.17.2
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	25.1.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.1
Send2Trash	1.8.2
six	1.16.0
smmap	5.0.1
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.0
types-python-dateutil	2.8.19.14
tzdata	2023.3
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.12
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4
y-py	0.6.2
ypy-websocket	0.8.4

Table 133: Packages in Python 3.11.6 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1

Library	Version
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	4.1.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
Babel	2.13.1
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
cattrs	23.2.2
certifi	2023.11.17
cffi	1.16.0
charset-normalizer	3.3.2
colorama	0.4.6
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
docstring-to-markdown	0.13
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
gitdb	4.0.11
GitPython	3.1.40
idna	3.6
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0

Library	Version
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.2
json5	0.9.14
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-lsp	2.2.1
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.5
jupyter_client	8.6.0
jupyter_core	5.5.0
jupyter_server	2.11.1
jupyter_server_fileid	0.9.0
jupyter_server_terminals	0.4.4
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.25.2
kerberos	1.3.1
kiwisolver	1.4.5
lsprotocol	2023.0.0
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbdime	3.2.1
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0

Library	Version
packaging	23.2
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.9.0
pickleshare	0.7.5
Pillow	10.1.0
platformdirs	4.0.0
prometheus-client	0.19.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pygls	1.2.0
Pygments	2.17.2
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0.1
pyzmq	25.1.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.1
Send2Trash	1.8.2
six	1.16.0
smmap	5.0.1
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1

Library	Version
tornado	6.3.3
traitlets	5.14.0
types-python-dateutil	2.8.19.14
tzdata	2023.3
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.12
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4
y-py	0.6.2
ypy-websocket	0.8.4

Python 3.10 Libraries for JupyterLab

This section lists the packages in Python 3.10 JupyterLab Runtimes that ship with ML Runtimes 2024.02.1.

Table 134: Packages in Python 3.10.9 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7

Library	Version
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.37
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0

Library	Version
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7

Library	Version
pytz	2023.3.post1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Table 135: Packages in Python 3.10.9 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0

Library	Version
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.37
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0

Library	Version
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1
pandas	2.1.4
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0

Library	Version
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Python 3.9 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2024.02.1.

Table 136: Packages in Python 3.9.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7

Library	Version
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.37
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1

Library	Version
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
packaging	23.1
pandas	2.1.4
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7

Library	Version
pytz	2023.3.post1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Table 137: Packages in Python 3.9.11 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0

Library	Version
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.37
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0

Library	Version
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
packaging	23.1

Library	Version
pandas	2.1.4
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2023.3.post1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1

Library	Version
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with ML Runtimes 2024.02.1.

Table 138: Packages in Python 3.8.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0

Library	Version
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.37
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1

Library	Version
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandas	2.0.3
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7

Library	Version
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Table 139: Packages in Python 3.8.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.37

Library	Version
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0

Library	Version
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandas	2.0.3
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0

Library	Version
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
tzdata	2023.3
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with ML Runtimes 2024.02.1.

Table 140: Packages in Python 3.7.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1

Library	Version
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.37
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0

Library	Version
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0

Library	Version
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

Table 141: Packages in Python 3.7.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3

Library	Version
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.37
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6

Library	Version
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.15.1
pyparsing	3.0.9

Library	Version
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

R 4.3 Libraries

This section lists the R 4.3 libraries that ship with ML Runtimes 2024.02.1.

Table 142: PBJ R 4.3.1 Libraries for Workbench

Library	Version
anyio	3.7.0

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.12.0
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4

Library	Version
notebook	6.5.4
notebook-shim	0.2.3
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.15.1
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 143: R 4.3.1 Libraries for Workbench

Library	Version
askpass	1.1
base	4.3.1
bitops	1.0-7
boot	1.3-28.1

Library	Version
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-22
cli	3.6.1
cluster	2.1.4
codetools	0.2-19
colorspace	2.1-0
compiler	4.3.1
curl	5.0.1
datasets	4.3.1
fansi	1.0.4
farver	2.1.1
foreign	0.8-84
ggplot2	3.4.2
glue	1.6.2
graphics	4.3.1
grDevices	4.3.1
grid	4.3.1
gtable	0.3.3
httr	1.4.6
isoband	0.2.7
jsonlite	1.8.7
KernSmooth	2.23-21
labeling	0.4.2
lattice	0.21-8
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-60
Matrix	1.5-4.1
methods	4.3.1
mgcv	1.8-42
mime	0.12
munsell	0.5.0
nlme	3.1-162
nnet	7.3-19
openssl	2.0.6
parallel	4.3.1

Library	Version
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.8
rlang	1.1.1
Rook	1.2
rpart	4.1.19
Rserve	1.7-0
scales	1.2.1
spatial	7.3-16
splines	4.3.1
stats	4.3.1
stats4	4.3.1
survival	3.5-5
sys	3.4.2
tcltk	4.3.1
tibble	3.2.1
tools	4.3.1
utf8	1.2.3
utils	4.3.1
vctrs	0.6.3
viridisLite	0.4.2
withr	2.5.0

ML Runtimes 2023.12.1

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2023.12.1

Note: Nvidia GPU Editions with a Workbench Editor come with CUDA 11.4.1 preinstalled.

All other Nvidia GPU Edition Runtimes come with CUDA 11.8.0 preinstalled.

Python 3.10 Libraries for Conda

This section lists the packages in Python 3.10 Conda Runtimes that ship with ML Runtimes 2023.12.1.

Table 144: Packages in Python 3.10 Conda Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.35
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0

Library	Version
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1

Library	Version
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0

Library	Version
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Python 3.11 Libraries for Workbench

This section lists the packages in Python 3.11 Workbench Runtimes that ship with ML Runtimes 2023.12.1.

Table 145: Packages in Python 3.11.6 PBJ Workbench Runtimes - Standard

Library	Version
anyio	4.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
certifi	2023.7.22
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3

Library	Version
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.26.0
ipython	8.17.2
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.2
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.5.0
jupyter_server	2.10.1
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.5
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.6
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
Pillow	10.1.0

Library	Version
platformdirs	4.0.0
prometheus-client	0.18.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.16.1
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0.1
pyzmq	24.0.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.0
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.13.0
types-python-dateutil	2.8.19.14
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.10
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4

Table 146: Packages in Python 3.11.6 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	4.0.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
certifi	2023.7.22
cffi	1.16.0
charset-normalizer	3.3.2
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.26.0
ipython	8.17.2
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
Jinja2	3.1.2
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1

Library	Version
jupyter-events	0.9.0
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.5.0
jupyter_server	2.10.1
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.5
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.6
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
Pillow	10.1.0
platformdirs	4.0.0
prometheus-client	0.18.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.16.1
pyparsing	3.1.1
python-dateutil	2.8.2

Library	Version
python-json-logger	2.0.7
PyYAML	6.0.1
pyzmq	24.0.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.0
Send2Trash	1.8.2
six	1.16.0
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.13.0
types-python-dateutil	2.8.19.14
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.10
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4

Python 3.10 Libraries for Workbench

This section lists the packages in Python 3.10 Workbench Runtimes that ship with ML Runtimes 2023.12.1.

Table 147: Packages in Python 3.10.9 Workbench Runtimes - Standard

Library	Version
asttokens	2.2.1
backcall	0.2.0
bitarray	2.7.3
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3

Library	Version
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2

Library	Version
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
urllib3	1.26.18
wcwidth	0.2.5

Table 148: Packages in Python 3.10.9 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2

Library	Version
Jinja2	3.1.2
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1

Library	Version
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	25.0.2
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.9.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Table 149: Packages in Python 3.10.9 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0

Library	Version
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.2
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6

Library	Version
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	25.0.2
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1

Library	Version
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.9.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2023.12.1.

Table 150: Packages in Python 3.9.11 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2

Library	Version
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 151: Packages in Python 3.9.11 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1

Library	Version
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.25.2
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0

Library	Version
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 152: Packages in Python 3.9.11 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4

Library	Version
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1

Library	Version
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 153: Packages in Python 3.9.11 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.7.0
argon2-cffi	21.3.0

Library	Version
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0

Library	Version
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2

Library	Version
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with ML Runtimes 2023.12.1.

Table 154: Packages in Python 3.8.16 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4

Library	Version
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.4
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 155: Packages in Python 3.8.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0

Library	Version
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.4
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1

Library	Version
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.15.0

Table 156: Packages in Python 3.8.16 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3

Library	Version
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2

Library	Version
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 157: Packages in Python 3.8.16 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0

Library	Version
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10

Library	Version
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5

Library	Version
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with ML Runtimes 2023.12.1.

Table 158: Packages in Python 3.7.16 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0

Library	Version
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.8.0

Table 159: Packages in Python 3.7.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-4	11.4.1-1
libcudnn8	8.2.2.26-1+cuda11.4
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1

Library	Version
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
zipp	3.8.0

Table 160: Packages in Python 3.7.16 PBJ Workbench - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22

Library	Version
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3

Library	Version
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Table 161: Packages in Python 3.7.16 PBJ Workbench - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8

Library	Version
libcudnn8-dev	8.7.0.84-1+cuda11.8
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3

Library	Version
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.18

Library	Version
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with ML Runtimes 2023.12.1.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-reflect	2.11.12
org.scala-lang:scala-compiler	2.11.12
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2
org.scalameta:scalameta	1.0.0

Python 3.11 Libraries for JupyterLab

This section lists the packages in Python 3.11 JupyterLab Runtimes that ship with ML Runtimes 2023.12.1.

Table 162: Packages in Python 3.11.6 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	4.1.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
Babel	2.13.1
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
cattrs	23.2.2
certifi	2023.11.17

Library	Version
cffi	1.16.0
charset-normalizer	3.3.2
colorama	0.4.6
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
docstring-to-markdown	0.13
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
gitdb	4.0.11
GitPython	3.1.40
idna	3.6
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.2
json5	0.9.14
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-lsp	2.2.1
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.5
jupyter_client	8.6.0
jupyter_core	5.5.0
jupyter_server	2.11.1
jupyter_server_fileid	0.9.0
jupyter_server_terminals	0.4.4
jupyter_server_ydoc	0.6.1

Library	Version
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.25.2
kerberos	1.3.1
kiwisolver	1.4.5
lsprotocol	2023.0.0
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbdime	3.2.1
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.9.0
pickleshare	0.7.5
Pillow	10.1.0
platformdirs	4.0.0
prometheus-client	0.19.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pygls	1.2.0
Pygments	2.17.2

Library	Version
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0.1
pyzmq	25.1.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.1
Send2Trash	1.8.2
six	1.16.0
smmap	5.0.1
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.0
types-python-dateutil	2.8.19.14
uri-template	1.3.0
urllib3	2.1.0
wcwidth	0.2.12
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4
y-py	0.6.2
ypy-websocket	0.8.4

Table 163: Packages in Python 3.11.6 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0

Library	Version
aiosqlite	0.19.0
anyio	4.1.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
attrs	23.1.0
Babel	2.13.1
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.8.3
bleach	6.1.0
cattrs	23.2.2
certifi	2023.11.17
cffi	1.16.0
charset-normalizer	3.3.2
colorama	0.4.6
comm	0.2.0
contourpy	1.2.0
cycler	0.12.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
docstring-to-markdown	0.13
executing	2.0.1
fastjsonschema	2.19.0
fonttools	4.44.3
fqdn	1.5.1
gitdb	4.0.11
GitPython	3.1.40
idna	3.6
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.19.1
jedi-language-server	0.41.2
Jinja2	3.1.2

Library	Version
json5	0.9.14
jsonpointer	2.4
jsonschema	4.20.0
jsonschema-specifications	2023.11.1
jupyter-events	0.9.0
jupyter-lsp	2.2.1
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.5
jupyter_client	8.6.0
jupyter_core	5.5.0
jupyter_server	2.11.1
jupyter_server_fileid	0.9.0
jupyter_server_terminals	0.4.4
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab_pygments	0.3.0
jupyterlab_server	2.25.2
kerberos	1.3.1
kiwisolver	1.4.5
lsprotocol	2023.0.0
MarkupSafe	2.1.3
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	3.0.2
nbclassic	1.0.0
nbclient	0.9.0
nbconvert	7.11.0
nbdime	3.2.1
nbformat	5.9.2
nest-asyncio	1.5.8
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
overrides	7.4.0
packaging	23.2
pandocfilters	1.5.0
parso	0.8.3

Library	Version
pexpect	4.9.0
pickleshare	0.7.5
Pillow	10.1.0
platformdirs	4.0.0
prometheus-client	0.19.0
prompt-toolkit	3.0.41
psutil	5.9.6
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pygls	1.2.0
Pygments	2.17.2
pyparsing	3.1.1
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0.1
pyzmq	25.1.1
referencing	0.31.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rpds-py	0.13.1
Send2Trash	1.8.2
six	1.16.0
smmap	5.0.1
sniffio	1.3.0
soupsieve	2.5
stack-data	0.6.3
terminado	0.18.0
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.3
traitlets	5.14.0
types-python-dateutil	2.8.19.14
uri-template	1.3.0
urllib3	2.1.0

Library	Version
wcwidth	0.2.12
webcolors	1.13
webencodings	0.5.1
websocket-client	1.6.4
y-py	0.6.2
ypy-websocket	0.8.4

Python 3.10 Libraries for JupyterLab

This section lists the packages in Python 3.10 JupyterLab Runtimes that ship with ML Runtimes 2023.12.1.

Table 164: Packages in Python 3.10.9 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4

Library	Version
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.35
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3

Library	Version
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0

Library	Version
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Table 165: Packages in Python 3.10.9 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2

Library	Version
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.35
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6

Library	Version
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.25.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1

Library	Version
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Python 3.9 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2023.12.1.

Table 166: Packages in Python 3.9.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.35
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0

Library	Version
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4

Library	Version
notebook_shim	0.2.3
numpy	1.25.2
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1

Library	Version
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Table 167: Packages in Python 3.9.11 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0

Library	Version
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.35
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1

Library	Version
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.25.2
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1

Library	Version
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with ML Runtimes 2023.12.1.

Table 168: Packages in Python 3.8.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.35
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0

Library	Version
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4

Library	Version
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0

Library	Version
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Table 169: Packages in Python 3.8.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22

Library	Version
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.35
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0

Library	Version
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20

Library	Version
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with ML Runtimes 2023.12.1.

Table 170: Packages in Python 3.7.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.35
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0

Library	Version
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1

Library	Version
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

Table 171: Packages in Python 3.7.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
cuda-libraries-11-8	11.8.0-1
cuda-libraries-dev-11-8	11.8.0-1
libcudnn8	8.7.0.84-1+cuda11.8
libcudnn8-dev	8.7.0.84-1+cuda11.8
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0

Library	Version
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.35
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3

Library	Version
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.24.4
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	10.0.1
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1

Library	Version
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.18
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

R 4.3 Libraries

This section lists the R 4.3 libraries that ship with ML Runtimes 2023.12.1.

Table 172: PBJ R 4.3.1 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0

Library	Version
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.12.0
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook-shim	0.2.3
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5

Library	Version
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.15.1
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.18
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 173: R 4.3.1 Libraries for Workbench

Library	Version
askpass	1.1
base	4.3.1
bitops	1.0-7
boot	1.3-28.1
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-22
cli	3.6.1
cluster	2.1.4

Library	Version
codetools	0.2-19
colorspace	2.1-0
compiler	4.3.1
curl	5.0.1
datasets	4.3.1
fansi	1.0.4
farver	2.1.1
foreign	0.8-84
ggplot2	3.4.2
glue	1.6.2
graphics	4.3.1
grDevices	4.3.1
grid	4.3.1
gtable	0.3.3
httr	1.4.6
isoband	0.2.7
jsonlite	1.8.7
KernSmooth	2.23-21
labeling	0.4.2
lattice	0.21-8
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-60
Matrix	1.5-4.1
methods	4.3.1
mgcv	1.8-42
mime	0.12
munsell	0.5.0
nlme	3.1-162
nnet	7.3-19
openssl	2.0.6
parallel	4.3.1
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.8
rlang	1.1.1

Library	Version
Rook	1.2
rpart	4.1.19
Rserve	1.7-0
scales	1.2.1
spatial	7.3-16
splines	4.3.1
stats	4.3.1
stats4	4.3.1
survival	3.5-5
sys	3.4.2
tcltk	4.3.1
tibble	3.2.1
tools	4.3.1
utf8	1.2.3
utils	4.3.1
vctrs	0.6.3
viridisLite	0.4.2
withr	2.5.0

ML Runtimes 2023.08.2

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2023.08.2

Note: Nvidia GPU Editions with a Workbench Editor come with CUDA 11.4.1 preinstalled. All other Nvidia GPU Edition Runtimes come with CUDA 11.8.0 preinstalled.

Python 3.10 Libraries for Conda

This section lists the packages in Python 3.10 Conda Runtimes that ship with ML Runtimes 2023.08.2.

Table 174: Packages in Python 3.10 Conda Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1

Library	Version
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.32
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2

Library	Version
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5

Library	Version
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.14
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9

Library	Version
ypy-websocket	0.8.2

Python 3.10 Libraries for Workbench

This section lists the packages in Python 3.10 Workbench Runtimes that ship with ML Runtimes 2023.08.2.

Table 175: Packages in Python 3.10.9 Workbench Runtimes - Standard

Library	Version
asttokens	2.2.1
backcall	0.2.0
bitarray	2.7.3
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.2
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0

Library	Version
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
urllib3	1.26.6
wcwidth	0.2.5

Table 176: Packages in Python 3.10.9 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6

Library	Version
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.2
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1

Library	Version
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	25.0.2
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.9.0
uri-template	1.2.0
urllib3	1.26.15
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1

Library	Version
websocket-client	1.5.1

Table 177: Packages in Python 3.10.9 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.2
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.2

Library	Version
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0

Library	Version
pyzmq	25.0.2
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.9.0
uri-template	1.2.0
urllib3	1.26.15
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2023.08.2.

Table 178: Packages in Python 3.9.11 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2

Library	Version
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.22.0
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3

Library	Version
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 179: Packages in Python 3.9.11 Workbench Runtimes - Nvidia GPU Edition

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.22.0
packaging	23.1

Library	Version
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 180: Packages in Python 3.9.11 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22

Library	Version
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4

Library	Version
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 181: Packages in Python 3.9.11 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1

Library	Version
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0

Library	Version
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with ML Runtimes 2023.08.2.

Table 182: Packages in Python 3.8.16 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2

Library	Version
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.22.0
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 183: Packages in Python 3.8.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
argcomplete	2.0.0
asttokens	2.2.1

Library	Version
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.22.0
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1

Library	Version
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 184: Packages in Python 3.8.16 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1

Library	Version
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0

Library	Version
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 185: Packages in Python 3.8.16 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0

Library	Version
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4

Library	Version
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with ML Runtimes 2023.08.2.

Table 186: Packages in Python 3.7.16 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2

Library	Version
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 187: Packages in Python 3.7.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.15.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 188: Packages in Python 3.7.16 PBJ Workbench - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4

Library	Version
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27

Library	Version
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Table 189: Packages in Python 3.7.16 PBJ Workbench - Nvidia GPU Edition

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0

Library	Version
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0

Library	Version
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with ML Runtimes 2023.08.2.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12

Library	Version
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-reflect	2.11.12
org.scala-lang:scala-compiler	2.11.12
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2
org.scalameta:scalameta	1.0.0

Python 3.10 Libraries for JupyterLab

This section lists the packages in Python 3.10 JupyterLab Runtimes that ship with ML Runtimes 2023.08.2.

Table 190: Packages in Python 3.10.9 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4

Library	Version
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.32
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3

Library	Version
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0

Library	Version
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.14
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Table 191: Packages in Python 3.10.9 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22

Library	Version
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.32
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1

Library	Version
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1

Library	Version
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.14
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Python 3.9 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2023.08.2.

Table 192: Packages in Python 3.9.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.32
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0

Library	Version
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4

Library	Version
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1

Library	Version
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Table 193: Packages in Python 3.9.11 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0

Library	Version
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.32
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2

Library	Version
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7

Library	Version
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with ML Runtimes 2023.08.2.

Table 194: Packages in Python 3.8.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0

Library	Version
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.32
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0

Library	Version
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3

Library	Version
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0

Library	Version
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Table 195: Packages in Python 3.8.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0

Library	Version
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.32
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4

Library	Version
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0

Library	Version
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with ML Runtimes 2023.08.2.

Table 196: Packages in Python 3.7.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0

Library	Version
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.32
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1

Library	Version
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20

Library	Version
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

Table 197: Packages in Python 3.7.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0

Library	Version
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.32
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0

Library	Version
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1

Library	Version
pycparser	2.20
Pygments	2.15.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

R 4.3 Libraries

This section lists the R 4.3 libraries that ship with ML Runtimes 2023.08.2.

Table 198: PBJ R 4.3.1 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.12.0
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10

Library	Version
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook-shim	0.2.3
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.15.1
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 199: R 4.1.3 Libraries for Workbench

Library	Version
askpass	1.1

Library	Version
base	4.3.1
bitops	1.0-7
boot	1.3-28.1
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-22
cli	3.6.1
cluster	2.1.4
codetools	0.2-19
colorspace	2.1-0
compiler	4.3.1
curl	5.0.0
datasets	4.3.1
fansi	1.0.4
farver	2.1.1
foreign	0.8-84
ggplot2	3.4.2
glue	1.6.2
graphics	4.3.1
grDevices	4.3.1
grid	4.3.1
gtable	0.3.3
httr	1.4.6
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-21
labeling	0.4.2
lattice	0.21-8
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-60
Matrix	1.5-4.1
methods	4.3.1
mgcv	1.8-42
mime	0.12
munsell	0.5.0
nlme	3.1-162

Library	Version
nnet	7.3-19
openssl	2.0.6
parallel	4.3.1
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.8
rlang	1.1.1
Rook	1.2
rpart	4.1.19
Rserve	1.7-0
scales	1.2.1
spatial	7.3-16
splines	4.3.1
stats	4.3.1
stats4	4.3.1
survival	3.5-5
sys	3.4.1
tcltk	4.3.1
tibble	3.2.1
tools	4.3.1
utf8	1.2.3
utils	4.3.1
vctrs	0.6.2
viridisLite	0.4.2
withr	2.5.0

R 4.1 Libraries

This section lists the R 4.1 libraries that ship with ML Runtimes 2023.08.2.

Table 200: PBJ R 4.1.1 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1

Library	Version
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.12.0
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook-shim	0.2.3
packaging	23.1

Library	Version
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.15.1
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 201: R 4.1.1 Libraries for Workbench

Library	Version
askpass	1.1
base	4.1.1
bitops	1.0-7
boot	1.3-28
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2

Library	Version
cdsw	2.1
class	7.3-19
cli	3.6.1
cluster	2.1.2
codetools	0.2-18
colorspace	2.1-0
compiler	4.1.1
curl	5.0.0
datasets	4.1.1
fansi	1.0.4
farver	2.1.1
foreign	0.8-81
ggplot2	3.4.2
glue	1.6.2
graphics	4.1.1
grDevices	4.1.1
grid	4.1.1
gtable	0.3.3
httr	1.4.6
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-20
labeling	0.4.2
lattice	0.20-44
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-54
Matrix	1.3-4
methods	4.1.1
mgcv	1.8-36
mime	0.12
munsell	0.5.0
nlme	3.1-152
nnet	7.3-16
openssl	2.0.6
parallel	4.1.1
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8

Library	Version
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.8
rlang	1.1.1
Rook	1.2
rpart	4.1-15
Rserve	1.7-0
scales	1.2.1
spatial	7.3-14
splines	4.1.1
stats	4.1.1
stats4	4.1.1
survival	3.2-11
sys	3.4.1
tcltk	4.1.1
tibble	3.2.1
tools	4.1.1
utf8	1.2.3
utils	4.1.1
vctrs	0.6.2
viridisLite	0.4.2
withr	2.5.0

R 4.0 Libraries

This section lists the R 4.0 libraries that ship with ML Runtimes 2023.08.2.

Table 202: PBJ R 4.0.5 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0

Library	Version
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.12.0
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook-shim	0.2.3
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.38

Library	Version
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.15.1
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 203: R 4.0.5 Libraries for Workbench

Library	Version
askpass	1.1
base	4.0.5
bitops	1.0-7
boot	1.3-27
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-18
cli	3.6.1
cluster	2.1.1
codetools	0.2-18
colorspace	2.1-0

Library	Version
compiler	4.0.5
curl	5.0.0
datasets	4.0.5
fansi	1.0.4
farver	2.1.1
foreign	0.8-81
ggplot2	3.4.2
glue	1.6.2
graphics	4.0.5
grDevices	4.0.5
grid	4.0.5
gtable	0.3.3
httr	1.4.6
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-18
labeling	0.4.2
lattice	0.20-41
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-53.1
Matrix	1.3-2
methods	4.0.5
mgcv	1.8-34
mime	0.12
munsell	0.5.0
nlme	3.1-152
nnet	7.3-15
openssl	2.0.6
parallel	4.0.5
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.8
rlang	1.1.1
Rook	1.2
rpart	4.1-15

Library	Version
Rserve	1.7-0
scales	1.2.1
spatial	7.3-13
splines	4.0.5
stats	4.0.5
stats4	4.0.5
survival	3.2-10
sys	3.4.1
tcltk	4.0.5
tibble	3.2.1
tools	4.0.5
utf8	1.2.3
utils	4.0.5
vctrs	0.6.2
viridisLite	0.4.2
withr	2.5.0

R 3.6 Libraries

This section lists the R 3.6 libraries that ship with ML Runtimes 2023.08.2.

Table 204: PBJ R 3.6.3 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1

Library	Version
executing	1.2.0
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.12.0
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook-shim	0.2.3
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.15.1
pyrsistent	0.18.1
python-dateutil	2.8.2

Library	Version
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 205: R 3.6.3 Libraries for Workbench

Library	Version
askpass	1.1
base	3.6.3
bitops	1.0-7
boot	1.3-24
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-15
cli	3.6.1
cluster	2.1.0
codetools	0.2-16
colorspace	2.1-0
compiler	3.6.3
curl	5.0.0
datasets	3.6.3
fansi	1.0.4
farver	2.1.1
foreign	0.8-75

Library	Version
ggplot2	3.4.2
glue	1.6.2
graphics	3.6.3
grDevices	3.6.3
grid	3.6.3
gtable	0.3.3
httr	1.4.6
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-16
labeling	0.4.2
lattice	0.20-38
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-51.5
Matrix	1.2-18
methods	3.6.3
mgcv	1.8-31
mime	0.12
munsell	0.5.0
nlme	3.1-144
nnet	7.3-12
openssl	2.0.6
parallel	3.6.3
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.8
rlang	1.1.1
Rook	1.2
rpart	4.1-15
Rserve	1.7-0
scales	1.2.1
spatial	7.3-11
splines	3.6.3
stats	3.6.3
stats4	3.6.3

Library	Version
survival	3.1-8
sys	3.4.1
tcltk	3.6.3
tibble	3.2.1
tools	3.6.3
utf8	1.2.3
utils	3.6.3
vctrs	0.6.2
viridisLite	0.4.2
withr	2.5.0

ML Runtimes 2023.08

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2023.08.

Note: Nvidia GPU Editions with a Workbench Editor come with CUDA 11.4.1 preinstalled. All other Nvidia GPU Edition Runtimes come with CUDA 11.8.0 preinstalled.

Python 3.10 Libraries for Conda

This section lists the packages in Python 3.10 Conda Runtimes that ship with ML Runtimes 2023.08.

Table 206: Packages in Python 3.10 Conda Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22

Library	Version
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1

Library	Version
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1

Library	Version
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.14
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Python 3.10 Libraries for Workbench

This section lists the packages in Python 3.10 Workbench Runtimes that ship with ML Runtimes 2023.08.

Table 207: Packages in Python 3.10.9 Workbench Runtimes - Standard

Library	Version
asttokens	2.2.1
backcall	0.2.0
bitarray	2.7.3
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.24.2
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2

Library	Version
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
urllib3	1.26.6
wcwidth	0.2.5

Table 208: Packages in Python 3.10.9 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3

Library	Version
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.2
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.2.0

Library	Version
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.14.0
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	25.0.2
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.9.0
uri-template	1.2.0
urllib3	1.26.15
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Table 209: Packages in Python 3.10.9 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
anyio	3.6.2
argon2-cffi	21.3.0

Library	Version
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.2
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.2
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1

Library	Version
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.14.0
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	25.0.2
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0

Library	Version
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.9.0
uri-template	1.2.0
urllib3	1.26.15
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2023.08.

Table 210: Packages in Python 3.9.11 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0

Library	Version
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.22.0
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 211: Packages in Python 3.9.11 Workbench Runtimes - Nvidia GPU Edition

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.22.0
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0

Library	Version
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 212: Packages in Python 3.9.11 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4

Library	Version
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0

Library	Version
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 213: Packages in Python 3.9.11 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0

Library	Version
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10

Library	Version
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5

Library	Version
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with ML Runtimes 2023.08.

Table 214: Packages in Python 3.8.16 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.22.0

Library	Version
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 215: Packages in Python 3.8.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0

Library	Version
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.22.0
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
stack-data	0.6.2

Library	Version
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.3.2
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 216: Packages in Python 3.8.16 PBJ Workbench Runtimes - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0

Library	Version
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1

Library	Version
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 217: Packages in Python 3.8.16 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1

Library	Version
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with ML Runtimes 2023.08.

Table 218: Packages in Python 3.7.16 Workbench Runtimes - Standard

Library	Version
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0

Library	Version
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 219: Packages in Python 3.7.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2023.7.22
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8

Library	Version
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 220: Packages in Python 3.7.16 PBJ Workbench - Standard

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3

Library	Version
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1

Library	Version
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Table 221: Packages in Python 3.7.16 PBJ Workbench - Nvidia GPU Edition

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bitarray	2.6.0
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1

Library	Version
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyter_core	4.12.0
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.8.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5

Library	Version
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
terminado	0.13.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.7.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with ML Runtimes 2023.08.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-reflect	2.11.12
org.scala-lang:scala-compiler	2.11.12

Library	Version
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2
org.scalameta:scalameta	1.0.0

Python 3.10 Libraries for JupyterLab

This section lists the packages in Python 3.10 JupyterLab Runtimes that ship with ML Runtimes 2023.08.

Table 222: Packages in Python 3.10.9 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3

Library	Version
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3

Library	Version
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4

Library	Version
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.14
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Table 223: Packages in Python 3.10.9 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2023.7.22
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6

Library	Version
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0

Library	Version
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2

Library	Version
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.14
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Python 3.9 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2023.08.

Table 224: Packages in Python 3.9.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0

Library	Version
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0

Library	Version
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3

Library	Version
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0

Library	Version
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Table 225: Packages in Python 3.9.11 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4

Library	Version
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1

Library	Version
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1

Library	Version
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with ML Runtimes 2023.08.

Table 226: Packages in Python 3.8.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10

Library	Version
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3

Library	Version
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0

Library	Version
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5

Library	Version
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Table 227: Packages in Python 3.8.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3

Library	Version
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.10.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5

Library	Version
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.22.0
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.38
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0

Library	Version
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.3.2
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with ML Runtimes 2023.08.

Table 228: Packages in Python 3.7.16 JupyterLab Runtimes - Standard

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1

Library	Version
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0

Library	Version
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7

Library	Version
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

Table 229: Packages in Python 3.7.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0

Library	Version
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2023.7.22
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0

Library	Version
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.1
nbconvert	7.4.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.3
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2

Library	Version
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.31.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

R 4.3 Libraries

This section lists the R 4.3 libraries that ship with ML Runtimes 2023.08.

Table 230: PBJ R 4.3.1 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0

Library	Version
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.12.0
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook-shim	0.2.3

Library	Version
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.11.2
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 231: R 4.1.3 Libraries for Workbench

Library	Version
askpass	1.1
base	4.3.1
bitops	1.0-7
boot	1.3-28.1
brew	1.0-8
Cairo	1.2-0

Library	Version
caTools	1.18.2
cdsw	2.1
class	7.3-22
cli	3.6.1
cluster	2.1.4
codetools	0.2-19
colorspace	2.1-0
compiler	4.3.1
curl	5.0.0
datasets	4.3.1
fansi	1.0.4
farver	2.1.1
foreign	0.8-84
ggplot2	3.4.2
glue	1.6.2
graphics	4.3.1
grDevices	4.3.1
grid	4.3.1
gtable	0.3.3
httr	1.4.6
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-21
labeling	0.4.2
lattice	0.21-8
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-60
Matrix	1.5-4.1
methods	4.3.1
mgcv	1.8-42
mime	0.12
munsell	0.5.0
nlme	3.1-162
nnet	7.3-19
openssl	2.0.6
parallel	4.3.1
pillar	1.9.0
pkgconfig	2.0.3

Library	Version
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.8
rlang	1.1.1
Rook	1.2
rpart	4.1.19
Rserve	1.7-0
scales	1.2.1
spatial	7.3-16
splines	4.3.1
stats	4.3.1
stats4	4.3.1
survival	3.5-5
sys	3.4.1
tcltk	4.3.1
tibble	3.2.1
tools	4.3.1
utf8	1.2.3
utils	4.3.1
vctrs	0.6.2
viridisLite	0.4.2
withr	2.5.0

R 4.1 Libraries

This section lists the R 4.1 libraries that ship with ML Runtimes 2023.08.

Table 232: PBJ R 4.1.1 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0
certifi	2023.7.22

Library	Version
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.12.0
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook-shim	0.2.3
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1

Library	Version
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.11.2
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 233: R 4.1.1 Libraries for Workbench

Library	Version
askpass	1.1
base	4.1.1
bitops	1.0-7
boot	1.3-28
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-19
cli	3.6.1
cluster	2.1.2
codetools	0.2-18

Library	Version
colorspace	2.1-0
compiler	4.1.1
curl	5.0.0
datasets	4.1.1
fansi	1.0.4
farver	2.1.1
foreign	0.8-81
ggplot2	3.4.2
glue	1.6.2
graphics	4.1.1
grDevices	4.1.1
grid	4.1.1
gtable	0.3.3
httr	1.4.6
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-20
labeling	0.4.2
lattice	0.20-44
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-54
Matrix	1.3-4
methods	4.1.1
mgcv	1.8-36
mime	0.12
munsell	0.5.0
nlme	3.1-152
nnet	7.3-16
openssl	2.0.6
parallel	4.1.1
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.8
rlang	1.1.1
Rook	1.2

Library	Version
rpart	4.1-15
Rserve	1.7-0
scales	1.2.1
spatial	7.3-14
splines	4.1.1
stats	4.1.1
stats4	4.1.1
survival	3.2-11
sys	3.4.1
tcltk	4.1.1
tibble	3.2.1
tools	4.1.1
utf8	1.2.3
utils	4.1.1
vctrs	0.6.2
viridisLite	0.4.2
withr	2.5.0

R 4.0 Libraries

This section lists the R 4.0 libraries that ship with ML Runtimes 2023.08.

Table 234: PBJ R 4.0.5 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4

Library	Version
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.12.0
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook-shim	0.2.3
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.11.2
pyrsistent	0.18.1

Library	Version
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 235: R 4.0.5 Libraries for Workbench

Library	Version
askpass	1.1
base	4.0.5
bitops	1.0-7
boot	1.3-27
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-18
cli	3.6.1
cluster	2.1.1
codetools	0.2-18
colorspace	2.1-0
compiler	4.0.5
curl	5.0.0
datasets	4.0.5
fansi	1.0.4
farver	2.1.1

Library	Version
foreign	0.8-81
ggplot2	3.4.2
glue	1.6.2
graphics	4.0.5
grDevices	4.0.5
grid	4.0.5
gtable	0.3.3
httr	1.4.6
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-18
labeling	0.4.2
lattice	0.20-41
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-53.1
Matrix	1.3-2
methods	4.0.5
mgcv	1.8-34
mime	0.12
munsell	0.5.0
nlme	3.1-152
nnet	7.3-15
openssl	2.0.6
parallel	4.0.5
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.8
rlang	1.1.1
Rook	1.2
rpart	4.1-15
Rserve	1.7-0
scales	1.2.1
spatial	7.3-13
splines	4.0.5
stats	4.0.5

Library	Version
stats4	4.0.5
survival	3.2-10
sys	3.4.1
tcltk	4.0.5
tibble	3.2.1
tools	4.0.5
utf8	1.2.3
utils	4.0.5
vctrs	0.6.2
viridisLite	0.4.2
withr	2.5.0

R 3.6 Libraries

This section lists the R 3.6 libraries that ship with ML Runtimes 2023.08.

Table 236: PBJ R 3.6.3 Libraries for Workbench

Library	Version
anyio	3.7.0
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.12.2
bleach	4.1.0
certifi	2023.7.22
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.17.1
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.12.0

Library	Version
ipykernel	6.9.0
ipython	8.10.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.12.0
jupyter-kernel-gateway	2.5.2
jupyter-server	1.24.0
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	1.0.0
nbclient	0.5.10
nbconvert	7.4.0
nbformat	5.9.0
nest-asyncio	1.5.4
notebook	6.5.4
notebook-shim	0.2.3
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.11.2
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.31.0
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0

Library	Version
soupsieve	2.4.1
stack-data	0.6.2
terminado	0.13.1
tinycss2	1.2.1
tornado	6.3.2
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.5.2
zipp	3.15.0

Table 237: R 3.6.3 Libraries for Workbench

Library	Version
askpass	1.1
base	3.6.3
bitops	1.0-7
boot	1.3-24
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-15
cli	3.6.1
cluster	2.1.0
codetools	0.2-16
colorspace	2.1-0
compiler	3.6.3
curl	5.0.0
datasets	3.6.3
fansi	1.0.4
farver	2.1.1
foreign	0.8-75
ggplot2	3.4.2
glue	1.6.2
graphics	3.6.3
grDevices	3.6.3
grid	3.6.3

Library	Version
gtable	0.3.3
httr	1.4.6
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-16
labeling	0.4.2
lattice	0.20-38
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-51.5
Matrix	1.2-18
methods	3.6.3
mgcv	1.8-31
mime	0.12
munsell	0.5.0
nlme	3.1-144
nnet	7.3-12
openssl	2.0.6
parallel	3.6.3
pillar	1.9.0
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.8
rlang	1.1.1
Rook	1.2
rpart	4.1-15
Rserve	1.7-0
scales	1.2.1
spatial	7.3-11
splines	3.6.3
stats	3.6.3
stats4	3.6.3
survival	3.1-8
sys	3.4.1
tcltk	3.6.3
tibble	3.2.1
tools	3.6.3

Library	Version
utf8	1.2.3
utils	3.6.3
vctrs	0.6.2
viridisLite	0.4.2
withr	2.5.0

ML Runtimes 2023.05

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2023.05.

Note: Nvidia GPU Editions with a Workbench Editor come with CUDA 11.4.1 preinstalled. All other Nvidia GPU Edition Runtimes come with CUDA 11.8.0 preinstalled.

Python 3.10 Libraries for Conda

This section lists the packages in Python 3.10 Conda Runtimes that ship with ML Runtimes 2023.05.

Table 238: Packages in Python 3.10 Conda Runtimes - Standard

Library	Version
python	3.10.9
pip	23.1.2
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2022.12.7
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6

Library	Version
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.1.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0

Library	Version
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2

Library	Version
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.28.2
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.1
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.14
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Python 3.10 Libraries for Workbench

This section lists the packages in Python 3.10 Workbench Runtimes that ship with ML Runtimes 2023.05.

Table 239: Packages in Python 3.10.9 PBJ Workbench Runtimes - Standard

Library	Version
python	3.10.9
pip	23.1.2
anyio	3.6.2

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2022.12.7
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.1.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.2
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.1
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2

Library	Version
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.14.0
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	25.0.2
requests	2.28.2
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0

Library	Version
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.2
traitlets	5.9.0
uri-template	1.2.0
urllib3	1.26.15
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Table 240: Packages in Python 3.10.9 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.10.9
pip	23.1.2
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	22.2.0
backcall	0.2.0
beautifulsoup4	4.12.0
bitarray	2.7.3
bleach	6.0.0
certifi	2022.12.7
cffi	1.15.1
charset-normalizer	3.1.0
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.6
decorator	5.1.1
defusedxml	0.7.1

Library	Version
entrypoints	0.4
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
idna	3.4
impyla	0.18.0
ipykernel	6.9.0
ipython	8.1.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
Jinja2	3.1.2
jsonpointer	2.3
jsonschema	4.17.3
jupyter-events	0.6.3
jupyter-kernel-gateway	2.5.1
jupyter_client	7.4.9
jupyter_core	5.3.0
jupyter_server	2.5.0
jupyter_server_terminals	0.4.4
jupyterlab-pygments	0.2.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.6
mistune	2.0.5
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.0
nbformat	5.8.0
nest-asyncio	1.5.6
notebook	6.5.3
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3

Library	Version
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.2.0
prometheus-client	0.16.0
prompt-toolkit	3.0.38
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.14.0
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	25.0.2
requests	2.28.2
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tornado	6.2
traitlets	5.9.0
uri-template	1.2.0
urllib3	1.26.15
wcwidth	0.2.6
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2023.05.

Table 241: Packages in Python 3.9.11 Workbench Runtimes - Standard

Library	Version
python	3.9.11
pip	21.1.3
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.1.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
packaging	23.1
parso	0.8.3
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 242: Packages in Python 3.9.11 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.11
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7

Library	Version
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.1.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0

Library	Version
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 243: Packages in Python 3.9.11 PBJ Workbench Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
bitarray	2.6.0
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	0.4
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.1.0
ipython-genutils	0.2.0

Library	Version
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.11.2
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.12
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0

Library	Version
six	1.16.0
stack-data	0.6.2
terminado	0.13.1
testpath	0.5.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Table 244: Packages in Python 3.9.11 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.6
pip	21.1.3
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
bitarray	2.6.0
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0

Library	Version
impyla	0.18.0
ipykernel	6.9.0
ipython	8.1.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.1
jupyter_core	4.11.2
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.12
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1

Library	Version
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
stack-data	0.6.2
terminado	0.13.1
testpath	0.5.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with ML Runtimes 2023.05.

Table 245: Packages in Python 3.8.16 Workbench Runtimes - Standard

Library	Version
python	3.8.16
pip	23.1.2
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0

Library	Version
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.1.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.9.0

Library	Version
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 246: Packages in Python 3.8.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.16
pip	23.1.2
cuda-libraries	11.4.1-1
libcudnn8	8.2.2.26-1
argcomplete	2.0.0
asttokens	2.2.1
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.2
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.1.0
jedi	0.18.2
jupyter-client	7.2.2
jupyter_core	5.3.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.7.0
matplotlib-inline	0.1.3
nest-asyncio	1.5.5

Library	Version
numpy	1.21.6
packaging	23.1
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
stack-data	0.6.2
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.9.0
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.15.0

Table 247: Packages in Python 3.8.16 PBJ Workbench Runtimes - Standard

Library	Version
python	3.8.16
pip	23.1.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
bitarray	2.6.0

Library	Version
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.1.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.1
jupyter_core	4.11.2
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.12
numpy	1.21.6
packaging	21.3

Library	Version
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
stack-data	0.6.2
terminado	0.13.1
testpath	0.5.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Table 248: Packages in Python 3.8.16 PBJ Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.16
pip	23.1.2
cuda	11.8.0-1
libcudnn8	8.7.0.84-1

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
bitarray	2.6.0
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
contourpy	1.0.7
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
fonttools	4.39.3
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	8.1.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.1
jupyter_core	4.11.2
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10

Library	Version
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.12
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
stack-data	0.6.2
terminado	0.13.1
testpath	0.5.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with ML Runtimes 2023.05.

Table 249: Packages in Python 3.7.16 Workbench Runtimes - Standard

Library	Version
python	3.7.16
pip	23.1.2
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1

Library	Version
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 250: Packages in Python 3.7.16 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.7.16
pip	23.1.2
cuda	11.4.1-1
libcudnn8	8.2.2.26
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter_core	4.12.0
kerberos	1.3.1
kiwisolver	1.4.4

Library	Version
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 251: Packages in Python 3.7.16 PBJ Workbench - Standard

Library	Version
python	3.7.16
pip	23.1.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.6.0
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0

Library	Version
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.1
jupyter_core	4.11.2
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.12
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1

Library	Version
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Table 252: Packages in Python 3.7.16 PBJ Workbench - Nvidia GPU Edition

Library	Version
python	3.7.16
pip	23.1.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.6.0
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0

Library	Version
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.18.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-kernel-gateway	2.5.1
jupyter_core	4.11.2
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.12
numpy	1.21.6
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0

Library	Version
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.16.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with ML Runtimes 2023.05.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-reflect	2.11.12
org.scala-lang:scala-compiler	2.11.12
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2
org.scalameta:scalameta	1.0.0

Python 3.10 Libraries for JupyterLab

This section lists the packages in Python 3.10 JupyterLab Runtimes that ship with ML Runtimes 2023.05.

Table 253: Packages in Python 3.10.9 JupyterLab Runtimes - Standard

Library	Version
python	3.10.9
pip	23.1.2
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2022.12.7
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	3.4
impyla	0.18.0

Library	Version
ipykernel	6.22.0
ipython	8.1.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2

Library	Version
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.28.2
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1

Library	Version
tornado	6.1
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.14
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Table 254: Packages in Python 3.10.9 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.10.9
pip	23.1.2
cuda	11.8.0-1
libcudnn8	8.7.0.84
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.6.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arrow	1.2.3
asttokens	2.2.1
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.2
bitarray	2.7.3
bleach	6.0.0
cattrs	22.2.0
certifi	2022.12.7
cffi	1.15.1
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7

Library	Version
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.7.1
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.16.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	3.4
impyla	0.18.0
ipykernel	6.22.0
ipython	8.1.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.2
json5	0.9.11
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.23.4
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.2.2
jupyterlab_server	2.20.0

Library	Version
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.2
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	2.0.5
nbclassic	0.5.3
nbclient	0.6.3
nbconvert	7.2.9
nbdime	3.2.0
nbformat	5.7.3
nest-asyncio	1.5.5
notebook	6.5.4
notebook_shim	0.2.2
numpy	1.24.2
packaging	23.1
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
platformdirs	3.5.0
prometheus-client	0.16.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.19.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0

Library	Version
pyzmq	22.3.0
requests	2.28.2
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.3.0
soupsieve	2.4
stack-data	0.6.2
terminado	0.17.1
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.2.1
tomli	2.0.1
tornado	6.1
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.5.0
uri-template	1.2.0
urllib3	1.26.14
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.5.1
y-py	0.5.9
ypy-websocket	0.8.2

Python 3.9 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2023.05.

Table 255: Packages in Python 3.9.16 JupyterLab Runtimes - Standard

Library	Version
python	3.9.16
pip	23.1.2
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0

Library	Version
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2020.11.8
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.1.0
ipython-genutils	0.2.0
isoduration	20.11.0

Library	Version
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.12
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.29.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0

Library	Version
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Table 256: Packages in Python 3.9.11 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.16
pip	23.1.2
cuda	11.8.0-1
libcudnn8	8.7.0.84
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2020.11.8
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2

Library	Version
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.1.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1

Library	Version
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.12
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
PyYAML	6.0
pyzmq	22.3.0
requests	2.29.0

Library	Version
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with ML Runtimes 2023.05.

Table 257: Packages in Python 3.8.16 JupyterLab Runtimes - Standard

Library	Version
python	3.8.16
pip	23.1.2
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0

Library	Version
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2020.11.8
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.1.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2

Library	Version
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.12
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5

Library	Version
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.29.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0

Library	Version
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Table 258: Packages in Python 3.8.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.16
pip	23.1.2
cuda	11.4.1-1
libcudnn8	8.7.0.84
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
asttokens	2.2.1
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cattrs	22.2.0
certifi	2020.11.8
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
comm	0.1.3
contourpy	1.0.7
cycler	0.11.0
debugpy	1.6.7

Library	Version
decorator	4.4.2
defusedxml	0.6.0
docstring-to-markdown	0.12
entrypoints	0.4
exceptiongroup	1.1.1
executing	1.2.0
fastjsonschema	2.15.3
fonttools	4.39.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.22.0
ipython	8.1.0
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.18.2
jedi-language-server	0.40.0
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.6.3
jupyter-lsp	2.1.0
jupyter-server	1.24.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	5.3.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	4.1.0
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1

Library	Version
kerberos	1.3.1
kiwisolver	1.4.4
lsprotocol	2023.0.0a1
MarkupSafe	2.1.1
matplotlib	3.7.0
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.12
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
platformdirs	3.5.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
psutil	5.9.5
ptyprocess	0.6.0
pure-eval	0.2.2
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
pydantic	1.10.7
pygls	1.0.1
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1

Library	Version
PyYAML	6.0
pyzmq	22.3.0
requests	2.29.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
stack-data	0.6.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.9.0
typeguard	2.13.3
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.15.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with ML Runtimes 2023.05.

Table 259: Packages in Python 3.7.16 JupyterLab Runtimes - Standard

Library	Version
python	3.7.16
pip	23.1.2
aiofiles	22.1.0

Library	Version
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2020.11.8
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3

Library	Version
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.12
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2

Library	Version
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.29.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

Table 260: Packages in Python 3.7.16 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.7.16

Library	Version
pip	22.1.2
cuda	11.4.1-1
aiofiles	22.1.0
aiosqlite	0.19.0
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
arrow	1.2.3
async-generator	1.10
attrs	23.1.0
Babel	2.12.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
cached-property	1.5.2
certifi	2020.11.8
cffi	1.14.4
charset-normalizer	3.1.0
colorama	0.4.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
fqdn	1.5.1
gitdb	4.0.10
GitPython	3.1.31
idna	2.10
importlib-metadata	4.11.3
importlib-resources	5.12.0
impyla	0.18.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
isoduration	20.11.0
jedi	0.17.2
Jinja2	3.1.1

Library	Version
json5	0.9.5
jsonpointer	2.3
jsonschema	4.17.3
jupyter-client	7.2.2
jupyter-events	0.5.0
jupyter-lsp	1.5.1
jupyter-server	1.21.0
jupyter-server-mathjax	0.2.6
jupyter-ydoc	0.2.4
jupyter_core	4.12.0
jupyter_server_fileid	0.9.0
jupyter_server_ydoc	0.6.1
jupyterlab	3.6.0
jupyterlab-git	0.41.0
jupyterlab-lsp	3.10.2
jupyterlab-pygments	0.1.2
jupyterlab_server	2.22.1
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbdime	3.2.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.12
numpy	1.21.6
packaging	23.1
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pkgutil_resolve_name	1.3.10
prometheus-client	0.9.0

Library	Version
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	3.0.9
pyrsistent	0.17.3
python-dateutil	2.8.2
python-json-logger	2.0.7
pytz	2021.1
PyYAML	6.0
pyzmq	22.3.0
requests	2.29.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
Send2Trash	1.8.0
six	1.16.0
smmap	5.0.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.16.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tomli	2.0.1
tornado	6.1
traitlets	5.1.1
typing_extensions	4.3.0
uri-template	1.2.0
urllib3	1.26.6
wcwidth	0.2.5
webcolors	1.13
webencodings	0.5.1
websocket-client	1.2.1
y-py	0.5.9
ypy-websocket	0.8.2
zipp	3.8.0

R 4.0 Libraries

This section lists the R 4.0 libraries that ship with ML Runtimes 2023.05.

Table 261: PBJ R 4.0.5 Libraries for Workbench

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
ipykernel	6.9.0
ipython	8.1.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.11.2
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4

Library	Version
notebook	6.4.12
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
stack-data	0.6.2
terminado	0.13.1
testpath	0.5.0
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Table 262: R 4.0.5 Libraries for Workbench

Library	Version
askpass	1.1
base	4.0.5
bitops	1.0-7
boot	1.3-27
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2

Library	Version
cdsw	2.1
class	7.3-18
cli	3.6.0
cluster	2.1.1
codetools	0.2-18
colorspace	2.1-0
compiler	4.0.5
curl	5.0.0
datasets	4.0.5
fansi	1.0.4
farver	2.1.1
foreign	0.8-81
ggplot2	3.4.0
glue	1.6.2
graphics	4.0.5
grDevices	4.0.5
grid	4.0.5
gtable	0.3.1
httr	1.4.4
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-18
labeling	0.4.2
lattice	0.20-41
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-53.1
Matrix	1.3-2
methods	4.0.5
mgcv	1.8-34
mime	0.12
munsell	0.5.0
nlme	3.1-152
nnet	7.3-15
openssl	2.0.5
parallel	4.0.5
pillar	1.8.1
pkgconfig	2.0.3
png	0.1-8

Library	Version
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.7
rlang	1.0.6
Rook	1.2
rpart	4.1-15
Rserve	1.7-0
scales	1.2.1
spatial	7.3-13
splines	4.0.5
stats	4.0.5
stats4	4.0.5
survival	3.2-10
sys	3.4.1
tcltk	4.0.5
tibble	3.1.8
tools	4.0.5
utf8	1.2.2
utils	4.0.5
vctrs	0.5.2
viridisLite	0.4.1
withr	2.5.0

R 4.1 Libraries

This section lists the R 4.1 libraries that ship with ML Runtimes 2023.05.

Table 263: PBJ R 4.1.1 Libraries for Workbench

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1

Library	Version
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
ipykernel	6.9.0
ipython	8.1.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.11.2
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.12
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2

Library	Version
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
stack-data	0.6.2
terminado	0.13.1
testpath	0.5.0
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Table 264: R 4.1.1 Libraries for Workbench

Library	Version
askpass	1.1
base	4.1.1
bitops	1.0-7
boot	1.3-28
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-19
cli	3.6.0
cluster	2.1.2
codetools	0.2-18
colorspace	2.1-0
compiler	4.1.1
curl	5.0.0
datasets	4.1.1
fansi	1.0.4
farver	2.1.1
foreign	0.8-81
ggplot2	3.4.0
glue	1.6.2
graphics	4.1.1

Library	Version
grDevices	4.1.1
grid	4.1.1
gtable	0.3.1
httr	1.4.4
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-20
labeling	0.4.2
lattice	0.20-44
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-54
Matrix	1.3-4
methods	4.1.1
mgcv	1.8-36
mime	0.12
munsell	0.5.0
nlme	3.1-152
nnet	7.3-16
openssl	2.0.5
parallel	4.1.1
pillar	1.8.1
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.7
rlang	1.0.6
Rook	1.2
rpart	4.1-15
Rserve	1.7-0
scales	1.2.1
spatial	7.3-14
splines	4.1.1
stats	4.1.1
stats4	4.1.1
survival	3.2-11
sys	3.4.1
tcltk	4.1.1

Library	Version
tibble	3.1.8
tools	4.1.1
utf8	1.2.2
utils	4.1.1
vctrs	0.5.2
viridisLite	0.4.0
withr	2.5.0

R 3.6 Libraries

This section lists the R 3.6 libraries that ship with ML Runtimes 2023.05.

Table 265: PBJ R 3.6.3 Libraries for Workbench

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
attrs	21.4.0
backcall	0.2.0
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
executing	1.2.0
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
ipykernel	6.9.0
ipython	8.1.0
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.11.2

Library	Version
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
MarkupSafe	2.0.1
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.12
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-eval	0.2.2
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.9
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
stack-data	0.6.2
terminado	0.13.1
testpath	0.5.0
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Table 266: R 3.6.3 Libraries for Workbench

Library	Version
askpass	1.1
base	3.6.3
bitops	1.0-7
boot	1.3-24
brew	1.0-8
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-15
cli	3.6.0
cluster	2.1.0
codetools	0.2-16
colorspace	2.1-0
compiler	3.6.3
curl	5.0.0
datasets	3.6.3
fansi	1.0.4
farver	2.1.1
foreign	0.8-75
ggplot2	3.4.0
glue	1.6.2
graphics	3.6.3
grDevices	3.6.3
grid	3.6.3
gtable	0.3.1
httr	1.4.4
isoband	0.2.7
jsonlite	1.8.4
KernSmooth	2.23-16
labeling	0.4.2
lattice	0.20-38
lifecycle	1.0.3
magrittr	2.0.3
MASS	7.3-51.5
Matrix	1.2-18
methods	3.6.3
mgcv	1.8-31
mime	0.12

Library	Version
munsell	0.5.0
nlme	3.1-144
nnet	7.3-12
openssl	2.0.5
parallel	3.6.3
pillar	1.8.2
pkgconfig	2.0.3
png	0.1-8
R6	2.5.1
RColorBrewer	1.1-3
RJSONIO	1.3-1.7
rlang	1.0.6
Rook	1.2
rpart	4.1-15
Rserve	1.7-0
scales	1.2.1
spatial	7.3-11
splines	3.6.3
stats	3.6.3
stats4	3.6.3
survival	3.1-8
sys	3.4.1
tcltk	3.6.3
tibble	3.1.8
tools	3.6.3
utf8	1.2.2
utils	3.6.3
vctrs	0.5.2
viridisLite	0.4.1
withr	2.5.0

ML Runtimes 2022.11

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2022.11.

Note: Nvidia GPU Edition comes with CUDA 11.4.1 preinstalled.

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2022.11.

Table 267: Packages in Python 3.9.11 Workbench Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1

Library	Version
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 268: Packages in Python 3.9.11 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.4.4

Library	Version
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 269: Packages in Python 3.9.11 PBJ Workbench Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0

Library	Version
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1

Library	Version
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with ML Runtimes 2022.11.

Table 270: Packages in Python 3.8.13 Workbench Runtimes - Standard

Library	Version
python	3.8.6
pip	21.1.3
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.11.0

Library	Version
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 271: Packages in Python 3.8.13 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2

Library	Version
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 272: Packages in Python 3.8.13 PBJ Workbench Runtimes - Standard

Library	Version
python	3.8.6
pip	21.1.3
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3

Library	Version
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0

Library	Version
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with ML Runtimes 2022.11.

Table 273: Packages in Python 3.7.13 Workbench Runtimes - Standard

Library	Version
python	3.7.13
pip	21.1.3
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3

Library	Version
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 274: Packages in Python 3.7.13 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.7.13
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.6.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2

Library	Version
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.4.4
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.21.6
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	3.0.9
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.16.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 275: Packages in Python 3.7.13 PBJ Workbench - Standard

Library	Version
python	3.7.13
pip	21.1.3
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2

Library	Version
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with ML Runtimes 2022.11.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-reflect	2.11.12
org.scala-lang:scala-compiler	2.11.12
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2
org.scalameta:scalameta	1.0.0

Python 3.9.6 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2022.11.

Table 276: Packages in Python 3.9.11 JupyterLab Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3

Library	Version
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.21.6
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20

Library	Version
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

Table 277: Packages in Python 3.9.11 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0

Library	Version
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3

Library	Version
numpy	1.21.6
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with ML Runtimes 2022.11.

Table 278: Packages in Python 3.8.13 JupyterLab Runtimes - Standard

Library	Version
python	3.8.13
pip	21.1.3
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2

Library	Version
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.21.6
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0

Library	Version
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

Table 279: Packages in Python 3.8.13 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.13
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3

Library	Version
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.21.6
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20

Library	Version
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.3.0
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with ML Runtimes 2022.11.

Table 280: Packages in Python 3.7.13 JupyterLab Runtimes - Standard

Library	Version
python	3.7.13
pip	21.1.3
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0

Library	Version
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0

Library	Version
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.21.6
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

Table 281: Packages in Python 3.7.13 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.7.13
pip	21.1.3
cuda	11.4.1-1
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.6.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.11.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2

Library	Version
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.4.4
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.21.6
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0

Library	Version
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

R 4.1 Libraries

This section lists the R 4.1 libraries that ship with ML Runtimes 2022.11.

Table 282: PBJ R 4.1.1 Libraries for Workbench

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1

Library	Version
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1

Library	Version
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Table 283: R 4.1.1 Libraries for Workbench

Library	Version
askpass	1.1
base	4.1.1
bitops	1.0-7
boot	1.3-28
brew	1.0-6
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-19
cli	3.0.1
cluster	2.1.2
codetools	0.2-18
colorspace	2.0-2
compiler	4.1.1
crayon	1.4.1
curl	4.3.2
datasets	4.1.1
digest	0.6.27
ellipsis	0.3.2
fansi	0.5.0
farver	2.1.0
foreign	0.8-81
ggplot2	3.3.5
glue	1.4.2
graphics	4.1.1
grDevices	4.1.1

Library	Version
grid	4.1.1
gtable	0.3.0
httr	1.4.2
isoband	0.2.5
jsonlite	1.7.2
KernSmooth	2.23-20
labeling	0.4.2
lattice	0.20-44
lifecycle	1.0.0
magrittr	2.0.1
MASS	7.3-54
Matrix	1.3-4
methods	4.1.1
mgcv	1.8-36
mime	0.11
munsell	0.5.0
nlme	3.1-152
nnet	7.3-16
openssl	1.4.4
parallel	4.1.1
pillar	1.6.2
pkgconfig	2.0.3
png	0.1-7
R6	2.5.1
RColorBrewer	1.1-2
RJSONIO	1.3-1.5
rlang	0.4.11
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.1
spatial	7.3-14
splines	4.1.1
stats	4.1.1
stats4	4.1.1
survival	3.2-11
sys	3.4
tcltk	4.1.1
tibble	3.1.4

Library	Version
tools	4.1.1
utf8	1.2.2
utils	4.1.1
vctrs	0.3.8
viridisLite	0.4.0
withr	2.4.2

R 4.0 Libraries

This section lists the R 4.0 libraries that ship with ML Runtimes 2022.11.

Table 284: PBJ R 4.0.5 Libraries for Workbench

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1

Library	Version
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1

Library	Version
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Table 285: R 4.0.5 Libraries for Workbench

Library	Version
askpass	1.1
base	4.0.5
bitops	1.0-7
boot	1.3-27
brew	1.0-6
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-18
cli	3.0.1
cluster	2.1.1
codetools	0.2-18
colorspace	2.0-2
compiler	4.0.5
crayon	1.4.1
curl	4.3.2
datasets	4.0.5
digest	0.6.27
ellipsis	0.3.2
fansi	0.5.0
farver	2.1.0
foreign	0.8-81
ggplot2	3.3.5
glue	1.4.2
graphics	4.0.5
grDevices	4.0.5
grid	4.0.5
gtable	0.3.0
httr	1.4.2
isoband	0.2.5

Library	Version
jsonlite	1.7.2
KernSmooth	2.23-18
labeling	0.4.2
lattice	0.20-41
lifecycle	1.0.0
magrittr	2.0.1
MASS	7.3-53.1
Matrix	1.3-2
methods	4.0.5
mgcv	1.8-34
mime	0.11
munsell	0.5.0
nlme	3.1-152
nnet	7.3-15
openssl	1.4.4
parallel	4.0.5
pillar	1.6.2
pkgconfig	2.0.3
png	0.1-7
R6	2.5.1
RColorBrewer	1.1-2
RJSONIO	1.3-1.5
rlang	0.4.11
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.1
spatial	7.3-13
splines	4.0.5
stats	4.0.5
stats4	4.0.5
survival	3.2-10
sys	3.4
tcltk	4.0.5
tibble	3.1.4
tools	4.0.5
utf8	1.2.2
utils	4.0.5
vctrs	0.3.8

Library	Version
viridisLite	0.4.0
withr	2.4.2

R 3.6 Libraries

This section lists the R 3.6 libraries that ship with ML Runtimes 2022.11.

Table 286: PBJ R 3.6.3 Libraries for Workbench

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2

Library	Version
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5

Library	Version
webencodings	0.5.1
zipp	3.7.0

Table 287: R 3.6.3 Libraries for Workbench

Library	Version
askpass	1.1
base	3.6.3
bitops	1.0-7
boot	1.3-24
brew	1.0-6
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-15
cli	3.0.1
cluster	2.1.0
codetools	0.2-16
colorspace	2.0-2
compiler	3.6.3
crayon	1.4.1
curl	4.3.2
datasets	3.6.3
digest	0.6.27
ellipsis	0.3.2
fansi	0.5.0
farver	2.1.0
foreign	0.8-75
ggplot2	3.3.5
glue	1.4.2
graphics	3.6.3
grDevices	3.6.3
grid	3.6.3
gtable	0.3.0
httr	1.4.2
isoband	0.2.5
jsonlite	1.7.2
KernSmooth	2.23-16
labeling	0.4.2
lattice	0.20-38

Library	Version
lifecycle	1.0.0
magrittr	2.0.1
MASS	7.3-51.5
Matrix	1.2-18
methods	3.6.3
mgcv	1.8-31
mime	0.11
munsell	0.5.0
nlme	3.1-144
nnet	7.3-12
openssl	1.4.4
parallel	3.6.3
pillar	1.6.2
pkgconfig	2.0.3
png	0.1-7
R6	2.5.1
RColorBrewer	1.1-2
RJSONIO	1.3-1.5
rlang	0.4.11
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.1
spatial	7.3-11
splines	3.6.3
stats	3.6.3
stats4	3.6.3
survival	3.1-8
sys	3.4
tcltk	3.6.3
tibble	3.1.4
tools	3.6.3
utf8	1.2.2
utils	3.6.3
vctrs	0.3.8
viridisLite	0.4.0
withr	2.4.2

Python 3.9 Libraries for PBJ Workbench

This section lists the packages in Python 3.9 PBJ Workbench Runtimes that ship with ML Runtimes 2022.11.

Table 288: Packages in Python 3.9.11 PBJ Workbench Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4

Library	Version
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Python 3.8 Libraries for PBJ Workbench

This section lists the packages in Python 3.8 PBJ Workbench Runtimes that ship with ML Runtimes 2022.11.

Table 289: Packages in Python 3.8.13 PBJ Workbench Runtimes - Standard

Library	Version
python	3.8.6
pip	21.1.3
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4

Library	Version
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

Python 3.7 Libraries for PBJ Workbench

This section lists the packages in Python 3.7 PBJ Workbench Runtimes that ship with ML Runtimes 2022.11.

Table 290: Packages in Python 3.7.13 PBJ Workbench Runtimes - Standard

Library	Version
python	3.7.13
pip	21.1.3
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4

Library	Version
matplotlib-inline	0.1.3
mistune	0.8.4
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

PBJ R 4.1 Libraries

This section lists the PBJ R 4.1 libraries that ship with ML Runtimes 2022.11.

Table 291: PBJ R 4.1.1 Libraries for Workbench

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4

Library	Version
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

PBJ R 4.0 Libraries

This section lists the PBJ R 4.0 libraries that ship with ML Runtimes 2022.11.

Table 292: PBJ R 4.0.5 Libraries for Workbench

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4

Library	Version
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

PBJ R 3.6 Libraries

This section lists the PBJ R 3.6 libraries that ship with ML Runtimes 2022.11.

Table 293: PBJ R 3.6.3 Libraries for Workbench

Library	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
bitarray	2.3.6
bleach	4.1.0
certifi	2021.10.8
cffi	1.15.0
charset-normalizer	2.0.11
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
idna	3.3
importlib-metadata	4.11.0
importlib-resources	5.4.0
impyla	0.17.0
ipykernel	6.9.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3
jsonschema	4.4.0
jupyter-client	7.1.2
jupyter-core	4.9.1
jupyter-kernel-gateway	2.5.1
jupyterlab-pygments	0.1.2
kerberos	1.3.1
kiwisolver	1.3.2
MarkupSafe	2.0.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4

Library	Version
nbclient	0.5.10
nbconvert	6.4.2
nbformat	5.1.3
nest-asyncio	1.5.4
notebook	6.4.8
numpy	1.21.5
packaging	21.3
pandocfilters	1.5.0
parso	0.8.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.13.1
prompt-toolkit	3.0.27
ptyprocess	0.7.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.21
Pygments	2.11.2
pyparsing	3.0.7
pyrsistent	0.18.1
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.27.1
Send2Trash	1.8.0
six	1.16.0
terminado	0.13.1
testpath	0.5.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.0.1
urllib3	1.26.8
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.7.0

ML Runtimes 2022.04

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2022.04.

Note: Nvidia GPU Edition comes with CUDA 11.4.1 preinstalled.

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2022.04.

Table 294: Packages in Python 3.9.11 Workbench Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
Pillow	9.0.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 295: Packages in Python 3.9.11 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0

Library	Version
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with ML Runtimes 2022.04.

Table 296: Packages in Python 3.8.13 Workbench Runtimes - Standard

Library	Version
python	3.8.6
pip	21.1.3
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.2
pyzmq	22.3.0

Library	Version
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 297: Packages in Python 3.8.13 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5

Library	Version
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with ML Runtimes 2022.04.

Table 298: Packages in Python 3.7.13 Workbench Runtimes - Standard

Library	Version
python	3.7.11
pip	21.1.3
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0

Library	Version
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.2
pyzmq	22.3.0
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 299: Packages in Python 3.7.13 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.7.11
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
argcomplete	2.0.0
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
entrypoints	0.4
idna	3.2
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
jedi	0.17.2
jupyter-client	7.2.2
jupyter-core	4.9.2
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
nest-asyncio	1.5.5
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.2

Library	Version
pyzmq	22.3.0
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
zipp	3.8.0

Table 300: Packages in Python 3.7.13 PBJ Workbench - Standard

Library	Version
python	3.7.11
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10

Library	Version
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1

Library	Version
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with ML Runtimes 2022.04.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-reflect	2.11.12
org.scala-lang:scala-compiler	2.11.12
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2
org.scalameta:scalameta	1.0.0

Python 3.9.6 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2022.04.

Table 301: Packages in Python 3.9.11 JupyterLab Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2

Library	Version
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0

Library	Version
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

Table 302: Packages in Python 3.9.11 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4

Library	Version
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0

Library	Version
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with ML Runtimes 2022.04.

Table 303: Packages in Python 3.8.13 JupyterLab Runtimes - Standard

Library	Version
python	3.8.6
pip	21.1.3
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10

Library	Version
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1

Library	Version
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1

Library	Version
websocket-client	1.2.1
zipp	3.8.0

Table 304: Packages in Python 3.8.13 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0

Library	Version
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0

Library	Version
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with ML Runtimes 2022.04.

Table 305: Packages in Python 3.7.13 JupyterLab Runtimes - Standard

Library	Version
python	3.7.11
pip	21.1.3
anyio	3.3.0
argcomplete	2.0.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0

Library	Version
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5

Library	Version
Pillow	9.0.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

Table 306: Packages in Python 3.7.13 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.7.11
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0
argcomplete	2.0.0

Library	Version
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
beautifulsoup4	4.11.1
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
debugpy	1.6.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.4
fastjsonschema	2.15.3
idna	2.10
importlib-metadata	4.11.3
impyla	0.17.0
ipykernel	6.5.0
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	3.1.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	7.2.2
jupyter-core	4.9.2
jupyter-server	1.16.0
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	2.1.1
matplotlib	3.3.4
matplotlib-inline	0.1.3

Library	Version
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.5.0
nbformat	5.3.0
nest-asyncio	1.5.5
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.0.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.2
pytz	2021.1
pyzmq	22.3.0
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
soupsieve	2.3.2
terminado	0.9.2
thrift	0.11.0
thrift-sasl	0.4.3
tinycss2	1.1.1
tornado	6.1
traitlets	5.1.1
typing-extensions	4.1.1
urllib3	1.26.6

Library	Version
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.8.0

R 4.0 Libraries

This section lists the R 4.0 libraries that ship with ML Runtimes 2022.04.

Table 307: R 4.0.5 Libraries for Workbench

Library	Version
askpass	1.1
base	4.0.5
bitops	1.0-7
boot	1.3-27
brew	1.0-6
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-18
cli	3.0.1
cluster	2.1.1
codetools	0.2-18
colorspace	2.0-2
compiler	4.0.5
crayon	1.4.1
curl	4.3.2
datasets	4.0.5
digest	0.6.27
ellipsis	0.3.2
fansi	0.5.0
farver	2.1.0
foreign	0.8-81
ggplot2	3.3.5
glue	1.4.2
graphics	4.0.5
grDevices	4.0.5
grid	4.0.5
gtable	0.3.0

Library	Version
httr	1.4.2
isoband	0.2.5
jsonlite	1.7.2
KernSmooth	2.23-18
labeling	0.4.2
lattice	0.20-41
lifecycle	1.0.0
magrittr	2.0.1
MASS	7.3-53.1
Matrix	1.3-2
methods	4.0.5
mgcv	1.8-34
mime	0.11
munsell	0.5.0
nlme	3.1-152
nnet	7.3-15
openssl	1.4.4
parallel	4.0.5
pillar	1.6.2
pkgconfig	2.0.3
png	0.1-7
R6	2.5.1
RColorBrewer	1.1-2
RJSONIO	1.3-1.5
rlang	0.4.11
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.1
spatial	7.3-13
splines	4.0.5
stats	4.0.5
stats4	4.0.5
survival	3.2-10
sys	3.4
tcltk	4.0.5
tibble	3.1.4
tools	4.0.5
utf8	1.2.2

Library	Version
utils	4.0.5
vctrs	0.3.8
viridisLite	0.4.0
withr	2.4.2

R 4.1 Libraries

This section lists the R 4.1 libraries that ship with ML Runtimes 2022.04.

Table 308: R 4.1.1 Libraries for Workbench

Library	Version
askpass	1.1
base	4.1.1
bitops	1.0-7
boot	1.3-28
brew	1.0-6
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-19
cli	3.0.1
cluster	2.1.2
codetools	0.2-18
colorspace	2.0-2
compiler	4.1.1
crayon	1.4.1
curl	4.3.2
datasets	4.1.1
digest	0.6.27
ellipsis	0.3.2
fansi	0.5.0
farver	2.1.0
foreign	0.8-81
ggplot2	3.3.5
glue	1.4.2
graphics	4.1.1
grDevices	4.1.1
grid	4.1.1
gtable	0.3.0

Library	Version
httr	1.4.2
isoband	0.2.5
jsonlite	1.7.2
KernSmooth	2.23-20
labeling	0.4.2
lattice	0.20-44
lifecycle	1.0.0
magrittr	2.0.1
MASS	7.3-54
Matrix	1.3-4
methods	4.1.1
mgcv	1.8-36
mime	0.11
munsell	0.5.0
nlme	3.1-152
nnet	7.3-16
openssl	1.4.4
parallel	4.1.1
pillar	1.6.2
pkgconfig	2.0.3
png	0.1-7
R6	2.5.1
RColorBrewer	1.1-2
RJSONIO	1.3-1.5
rlang	0.4.11
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.1
spatial	7.3-14
splines	4.1.1
stats	4.1.1
stats4	4.1.1
survival	3.2-11
sys	3.4
tcltk	4.1.1
tibble	3.1.4
tools	4.1.1
utf8	1.2.2

Library	Version
utils	4.1.1
vctrs	0.3.8
viridisLite	0.4.0
withr	2.4.2

R 3.6 Libraries

This section lists the R 3.6 libraries that ship with ML Runtimes 2022.04.

Table 309: R 3.6.3 Libraries for Workbench

Library	Version
askpass	1.1
base	3.6.3
bitops	1.0-7
boot	1.3-24
brew	1.0-6
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-15
cli	3.0.1
cluster	2.1.0
codetools	0.2-16
colorspace	2.0-2
compiler	3.6.3
crayon	1.4.1
curl	4.3.2
datasets	3.6.3
digest	0.6.27
ellipsis	0.3.2
fansi	0.5.0
farver	2.1.0
foreign	0.8-75
ggplot2	3.3.5
glue	1.4.2
graphics	3.6.3
grDevices	3.6.3
grid	3.6.3
gtable	0.3.0

Library	Version
httr	1.4.2
isoband	0.2.5
jsonlite	1.7.2
KernSmooth	2.23-16
labeling	0.4.2
lattice	0.20-38
lifecycle	1.0.0
magrittr	2.0.1
MASS	7.3-51.5
Matrix	1.2-18
methods	3.6.3
mgcv	1.8-31
mime	0.11
munsell	0.5.0
nlme	3.1-144
nnet	7.3-12
openssl	1.4.4
parallel	3.6.3
pillar	1.6.2
pkgconfig	2.0.3
png	0.1-7
R6	2.5.1
RColorBrewer	1.1-2
RJSONIO	1.3-1.5
rlang	0.4.11
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.1
spatial	7.3-11
splines	3.6.3
stats	3.6.3
stats4	3.6.3
survival	3.1-8
sys	3.4
tcltk	3.6.3
tibble	3.1.4
tools	3.6.3
utf8	1.2.2

Library	Version
utils	3.6.3
vctrs	0.3.8
viridisLite	0.4.0
withr	2.4.2

ML Runtimes 2021.12

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2021.12.

Note: Nvidia GPU Edition comes with CUDA 11.4.1 preinstalled.

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2021.12.

Table 310: Packages in Python 3.9.7 Workbench Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
decorator	4.4.2
idna	3.2
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1

Library	Version
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Table 311: Packages in Python 3.9.7 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
decorator	4.4.2
idna	3.2
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7

Library	Version
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with ML Runtimes 2021.12.

Table 312: Packages in Python 3.8.6 Workbench Runtimes - Standard

Library	Version
python	3.8.6
pip	21.1.3
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0

Library	Version
decorator	4.4.2
idna	3.2
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Table 313: Packages in Python 3.8.6 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.6
pip	21.1.3
cuda	11.4.1-1

Library	Version
libcudnn8	8.2.2.26-1
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
decorator	4.4.2
idna	3.2
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Table 314: Packages in Python 3.8.6 Workbench Runtimes - RAPIDS

Library	Version
python	3.8.6
pip	21.1.3
_libgcc_mutex	0.1
_openmp_mutex	4.5
abseil-cpp	20210324.2
aiohttp	3.8.1
aiosignal	1.2.0
anyio	3.4.0
appdirs	1.4.4
argon2-cffi	21.1.0
arrow-cpp	5.0.0
arrow-cpp-proc	3.0.0
async-timeout	4.0.1
async_generator	1.10
attrs	21.2.0
aws-c-auth	0.6.7
aws-c-cal	0.5.12
aws-c-common	0.6.17
aws-c-compression	0.2.14
aws-c-event-stream	0.2.7
aws-c-http	0.6.10
aws-c-io	0.10.13
aws-c-mqtt	0.7.9
aws-c-s3	0.1.27
aws-c-sdkutils	0.1.1
aws-checksums	0.1.12
aws-crt-cpp	0.17.8
aws-sdk-cpp	1.9.145
backcall	0.2.0
backports	1.0
backports.functools_lru_cache	1.6.4
bleach	4.1.0
blosc	1.21.0
bokeh	2.4.0
boost	1.74.0
boost-cpp	1.74.0
brotli	1.0.9
brotli-bin	1.0.9

Library	Version
brotlipy	0.7.0
brunsli	0.1
bzip2	1.0.8
c-ares	1.18.1
c-blosc2	2.0.4
ca-certificates	2021.10.8
cachetools	4.2.4
cairo	1.16.0
cdsw	1.1.0
certifi	2021.5.30
cffi	1.15.0
cfitsio	3.470
charls	2.2.0
charset-normalizer	2.0.6
click	8.0.3
click-plugins	1.1.1
cligj	0.7.2
cloudpickle	2.0.0
colorama	0.4.4
colorcet	3.0.0
cryptography	36.0.1
cucim	21.12.00
cudatoolkit	11.0.221
cudf	21.12.02
cudf_kafka	21.12.02
cudnn	8.2.1.32
cugraph	21.12.00
cuml	21.12.00
cupy	9.6.0
curl	7.80.0
cusignal	21.12.00
cuspatial	21.12.00
custreamz	21.12.02
cuxfilter	21.12.00
cycler	0.11.0
cyrus-sasl	2.1.27
cytoolz	0.11.2
dask	2021.11.2
dask-core	2021.11.2

Library	Version
dask-cuda	21.12.00
dask-cudf	21.12.02
datashader	0.11.1
datashape	0.5.4
debugpy	1.5.1
decorator	5.1.0
defusedxml	0.7.1
distributed	2021.11.2
dlpack	0.5
entrypoints	0.3
expat	2.4.1
faiss-proc	1.0.0
fastavro	1.4.7
fastrlock	0.8
fiona	1.8.20
font-ttf-dejavu-sans-mono	2.37
font-ttf-inconsolata	3.000
font-ttf-source-code-pro	2.038
font-ttf-ubuntu	0.83
fontconfig	2.13.1
fonts-conda-ecosystem	1
fonts-conda-forge	1
fonttools	4.28.4
freetype	2.10.4
freexl	1.0.6
frozenlist	1.2.0
fsspec	2021.11.1
gdal	3.3.2
geopandas	0.9.0
geopandas-base	0.9.0
geos	3.9.1
geotiff	1.7.0
gettext	0.19.8.1
gflags	2.2.2
giflib	5.2.1
git	2.34.1
glog	0.5.0
gpuci-tools	0.3.1
grpc-cpp	1.41.1

Library	Version
hdf4	4.2.15
hdf5	1.12.1
heapdict	1.0.1
icu	68.2
idna	3.2
imagecodecs	2021.8.26
imageio	2.13.3
importlib-metadata	4.8.3
importlib_metadata	4.8.3
importlib_resources	5.4.0
ipykernel	6.6.0
ipython	7.30.1
ipython_genutils	0.2.0
ipywidgets	7.6.5
jbig	2.1
jedi	0.18.1
jinja2	3.0.3
joblib	1.1.0
jpeg	9d
json-c	0.15
jsonschema	4.3.1
jupyter-server-proxy	3.2.0
jupyter_client	7.1.0
jupyter_core	4.9.1
jupyter_server	1.13.1
jupyterlab_pygments	0.1.2
jupyterlab_widgets	1.0.2
jxrlib	1.1
kealib	1.4.14
kiwisolver	1.3.2
krb5	1.19.2
lcms2	2.12
ld_impl_linux-64	2.36.1
lerc	3.0
libaec	1.0.6
libblas	3.9.0
libbrotlicommon	1.0.9
libbrotlidec	1.0.9
libbrotlienc	1.0.9

Library	Version
libcblas	3.9.0
libcucim	21.12.00
libcudf	21.12.02
libcudf_kafka	21.12.02
libcugraph	21.12.00
libcuml	21.12.00
libcumlprims	21.12.00
libcurl	7.80.0
libcusolver	11.3.2.107
libcuspatial	21.12.00
libdap4	3.20.6
libdeflate	1.8
libedit	3.1.20191231
libev	4.33
libevent	2.1.10
libfaiss	1.7.0
libffi	3.4.2
libgcc-ng	9.4.0
libgcrypt	1.9.4
libgdal	3.3.2
libgfortran-ng	11.2.0
libgfortran5	11.2.0
libglib	2.70.2
libgpg-error	1.42
libgsasl	1.10.0
libhwloc	2.3.0
libiconv	1.16
libkml	1.3.0
liblapack	3.9.0
libllvm11	11.1.0
libnetcdf	4.8.1
libnghttp2	1.43.0
libnsl	2.0.0
libntlm	1.4
libopenblas	0.3.18
libpng	1.6.37
libpq	13.5
libprotobuf	3.18.1
librdkafka	1.6.1

Library	Version
librmm	21.12.00
librttopo	1.1.0
libsodium	1.0.18
libspatialindex	1.9.3
libspatialite	5.0.1
libssh2	1.10.0
libstdcxx-ng	9.4.0
libthrift	0.15.0
libtiff	4.3.0
libutf8proc	2.7.0
libuuid	2.32.1
libuv	1.42.0
libwebp	1.2.1
libwebp-base	1.2.1
libxcb	1.13
libxgboost	1.5.0dev.rapidsai21.12
libxml2	2.9.12
libzip	1.8.0
libzlib	1.2.11
libzopfli	1.0.3
llvm-openmp	12.0.1
llvmlite	0.37.0
locket	0.2.0
lz4-c	1.9.3
mapclassify	2.4.3
markdown	3.3.6
markupsafe	2.0.1
matplotlib-base	3.5.1
matplotlib-inline	0.1.3
mistune	0.8.4
msgpack-python	1.0.3
multidict	5.2.0
multipledispatch	0.6.0
munch	2.5.0
munkres	1.1.4
nbclient	0.5.9
nbconvert	6.3.0
nbformat	5.1.3
nccl	2.11.4.1

Library	Version
ncurses	6.2
nest-asyncio	1.5.4
networkx	2.6.3
nodejs	14.17.4
notebook	6.4.6
nspr	4.32
nss	3.73
numba	0.54.1
numpy	1.20.3
nvtx	0.2.3
olefile	0.46
openjpeg	2.4.0
openssl	1.1.1l
orc	1.7.1
packaging	21.3
pandas	1.3.5
pandoc	2.16.2
pandocfilters	1.5.0
panel	0.12.4
param	1.12.0
parquet-cpp	1.5.1
parso	0.8.3
partd	1.2.0
pcre	8.45
pcre2	10.37
perl	5.32.1
pexpect	4.8.0
pickleshare	0.7.5
pillow	8.4.0
pip	21.3.1
pixman	0.40.0
pooch	1.5.2
poppler	21.09.0
poppler-data	0.4.11
postgresql	13.5
proj	8.1.0
prometheus_client	0.12.0
prompt-toolkit	3.0.24
protobuf	3.18.1

Library	Version
psutil	5.8.0
pthread-stubs	0.4
ptxcompiler	0.2.0
ptyprocess	0.7.0
py-xgboost	1.5.0dev.rapidsai21.12
py4j	0.10.9
pyarrow	5.0.0
pycparser	2.21
pyct	0.4.6
pyct-core	0.4.6
pydeck	0.5.0
pyee	8.1.0
pygments	2.10.0
pynvml	11.4.1
pyopenssl	21.0.0
pyparsing	3.0.6
pyppeteer	0.2.6
pyproj	3.1.0
pyrsistent	0.18.0
pysocks	1.7.1
python	3.8.12
python-confluent-kafka	1.6.0
python-dateutil	2.8.2
python_abi	3.8
pytz	2021.3
pyviz_comms	2.1.0
pywavelets	1.2.0
pyyaml	6.0
pyzmq	22.3.0
rapids	21.12.00
rapids-xgboost	21.12.00
re2	2021.11.01
readline	8.1
requests	2.26.0
rmm	21.12.00
rtree	0.9.7
s2n	1.3.0
scikit-image	0.18.1
scikit-learn	1.0.1

Library	Version
scipy	1.7.3
send2trash	1.8.0
setuptools	59.6.0
shapely	1.8.0
simpervisor	0.4
six	1.16.0
snappy	1.1.8
sniffio	1.2.0
sortedcontainers	2.4.0
spdlog	1.8.5
sqlite	3.37.0
streamz	0.6.3
tblib	1.7.0
terminado	0.12.1
testpath	0.5.0
threadpoolctl	3.0.0
tifffile	2021.11.2
tiledb	2.3.4
tk	8.6.11
toolz	0.11.2
tornado	6.1
tqdm	4.62.3
traitlets	5.1.1
treelite	2.1.0
treelite-runtime	2.1.0
typing-extensions	4.0.1
typing_extensions	4.0.1
tzcode	2021e
tzdata	2021e
ucx	1.11.2+gef2bbcf
ucx-proc	1.0.0
ucx-py	0.23.0
unicodedata2	13.0.0.post2
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.3
websockets	9.1
wheel	0.37.0

Library	Version
widgetsnbextension	3.5.2
xarray	0.20.2
xerces-c	3.2.3
xgboost	1.5.0dev.rapidsai21.12
xorg-kbproto	1.0.7
xorg-libice	1.0.10
xorg-libsm	1.2.3
xorg-libx11	1.7.2
xorg-libxau	1.0.9
xorg-libxdmcp	1.1.3
xorg-libxext	1.3.4
xorg-libxrender	0.9.10
xorg-renderproto	0.11.1
xorg-xextproto	7.3.0
xorg-xproto	7.0.31
xz	5.2.5
yaml	0.2.5
yarl	1.7.2
zeromq	4.3.4
zfp	0.5.5
zict	2.0.0
zipp	3.6.0
zlib	1.2.11
zstd	1.5.0

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with ML Runtimes 2021.12.

Table 315: Packages in Python 3.7.11 Workbench Runtimes - Standard

Library	Version
python	3.7.11
pip	21.1.3
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
decorator	4.4.2

Library	Version
idna	3.2
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Table 316: Packages in Python 3.7.11 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.7.11
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1

Library	Version
backcall	0.2.0
bitarray	2.1.0
certifi	2021.5.30
charset-normalizer	2.0.6
cycler	0.10.0
decorator	4.4.2
idna	3.2
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.26.0
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Table 317: Packages in Python 3.7.11 Workbench Runtimes - RAPIDS

Library	Version
python	3.7.11
pip	21.1.3
cuda	11.4.1-1
_libgcc_mutex	0.1
_openmp_mutex	4.5
abseil-cpp	20210324.2
aiohttp	3.8.1
aiosignal	1.2.0
anyio	3.4.0
appdirs	1.4.4
argcomplete	1.12.3
argon2-cffi	21.1.0
arrow-cpp	5.0.0
arrow-cpp-proc	3.0.0
async-timeout	4.0.1
async_generator	1.10
asynctest	0.13.0
attrs	21.2.0
aws-c-auth	0.6.7
aws-c-cal	0.5.12
aws-c-common	0.6.17
aws-c-compression	0.2.14
aws-c-event-stream	0.2.7
aws-c-http	0.6.10
aws-c-io	0.10.13
aws-c-mqtt	0.7.9
aws-c-s3	0.1.27
aws-c-sdkutils	0.1.1
aws-checksums	0.1.12
aws-crt-cpp	0.17.8
aws-sdk-cpp	1.9.145
backcall	0.2.0
backports	1.0
backports.functools_lru_cache	1.6.4
bleach	4.1.0
blosc	1.21.0
bokeh	2.4.0
boost	1.74.0

Library	Version
boost-cpp	1.74.0
brotli	1.0.9
brotli-bin	1.0.9
brotlipy	0.7.0
brunsli	0.1
bzip2	1.0.8
c-ares	1.18.1
c-blosc2	2.0.4
ca-certificates	2021.10.8
cachetools	4.2.4
cairo	1.16.0
cdsw	1.1.0
certifi	2021.5.30
cffi	1.15.0
cfitsio	3.470
charls	2.2.0
charset-normalizer	2.0.6
click	8.0.3
click-plugins	1.1.1
cligj	0.7.2
cloudpickle	2.0.0
colorama	0.4.4
colorcet	3.0.0
cryptography	36.0.1
cucim	21.12.00
cudatoolkit	11.0.221
cudf	21.12.02
cudf_kafka	21.12.02
cudnn	8.2.1.32
cugraph	21.12.00
cuml	21.12.00
cupy	9.6.0
curl	7.80.0
cusignal	21.12.00
cuspatial	21.12.00
custreamz	21.12.02
cuxfilter	21.12.00
cycler	0.11.0
cyrus-sasl	2.1.27

Library	Version
cytoolz	0.11.2
dask	2021.11.2
dask-core	2021.11.2
dask-cuda	21.12.00
dask-cudf	21.12.02
datashader	0.11.1
datashape	0.5.4
debugpy	1.5.1
decorator	5.1.0
defusedxml	0.7.1
distributed	2021.11.2
dlpack	0.5
entrypoints	0.3
expat	2.4.1
faiss-proc	1.0.0
fastavro	1.4.7
fastrlock	0.8
fiona	1.8.20
font-ttf-dejavu-sans-mono	2.37
font-ttf-inconsolata	3.000
font-ttf-source-code-pro	2.038
font-ttf-ubuntu	0.83
fontconfig	2.13.1
fonts-conda-ecosystem	1
fonts-conda-forge	1
fonttools	4.28.4
freetype	2.10.4
freexl	1.0.6
frozenlist	1.2.0
fsspec	2021.11.1
gdal	3.3.2
geopandas	0.9.0
geopandas-base	0.9.0
geos	3.9.1
geotiff	1.7.0
gettext	0.19.8.1
gflags	2.2.2
giflib	5.2.1
git	2.34.1

Library	Version
glog	0.5.0
gpuci-tools	0.3.1
grpc-cpp	1.41.1
hdf4	4.2.15
hdf5	1.12.1
heapdict	1.0.1
icu	68.2
idna	3.2
imagecodecs	2021.8.26
imageio	2.13.3
importlib-metadata	4.8.3
importlib_metadata	4.8.3
importlib_resources	5.4.0
ipykernel	6.6.0
ipython	7.30.1
ipython_genutils	0.2.0
ipywidgets	7.6.5
jbig	2.1
jedi	0.18.1
jinja2	3.0.3
joblib	1.1.0
jpeg	9d
json-c	0.15
jsonschema	4.3.1
jupyter-server-proxy	3.2.0
jupyter_client	7.1.0
jupyter_core	4.9.1
jupyter_server	1.13.1
jupyterlab_pygments	0.1.2
jupyterlab_widgets	1.0.2
jxrlib	1.1
kealib	1.4.14
kiwisolver	1.3.2
krb5	1.19.2
lcms2	2.12
ld_impl_linux-64	2.36.1
lerc	3.0
libaec	1.0.6
libblas	3.9.0

Library	Version
libbrotlicommon	1.0.9
libbrotlidec	1.0.9
libbrotlienc	1.0.9
libcblas	3.9.0
libcucim	21.12.00
libcudf	21.12.02
libcudf_kafka	21.12.02
libcugraph	21.12.00
libcuml	21.12.00
libcumlprims	21.12.00
libcurl	7.80.0
libcusolver	11.3.2.107
libcuspatial	21.12.00
libdap4	3.20.6
libdeflate	1.8
libedit	3.1.20191231
libev	4.33
libevent	2.1.10
libfaiss	1.7.0
libffi	3.4.2
libgcc-ng	9.4.0
libgcrypt	1.9.4
libgdal	3.3.2
libgfortran-ng	11.2.0
libgfortran5	11.2.0
libglib	2.70.2
libgpg-error	1.42
libgsasl	1.10.0
libhwloc	2.3.0
libiconv	1.16
libkml	1.3.0
liblapack	3.9.0
libllvm11	11.1.0
libnetcdf	4.8.1
libnghttp2	1.43.0
libnsl	2.0.0
libntlm	1.4
libopenblas	0.3.18
libpng	1.6.37

Library	Version
libpq	13.5
libprotobuf	3.18.1
librdkafka	1.6.1
librmm	21.12.00
librttopo	1.1.0
libsodium	1.0.18
libspatialindex	1.9.3
libspatialite	5.0.1
libssh2	1.10.0
libstdcxx-ng	9.4.0
libthrift	0.15.0
libtiff	4.3.0
libutf8proc	2.7.0
libuuid	2.32.1
libuv	1.42.0
libwebp	1.2.1
libwebp-base	1.2.1
libxcb	1.13
libxgboost	1.5.0dev.rapidsai21.12
libxml2	2.9.12
libzip	1.8.0
libzlib	1.2.11
libzopfli	1.0.3
llvm-openmp	12.0.1
llvmlite	0.37.0
locket	0.2.0
lz4-c	1.9.3
mapclassify	2.4.3
markdown	3.3.6
markupsafe	2.0.1
matplotlib-base	3.5.1
matplotlib-inline	0.1.3
mistune	0.8.4
msgpack-python	1.0.3
multidict	5.2.0
multipledispatch	0.6.0
munch	2.5.0
munkres	1.1.4
nbclient	0.5.9

Library	Version
nbconvert	6.3.0
nbformat	5.1.3
nccl	2.11.4.1
ncurses	6.2
nest-asyncio	1.5.4
networkx	2.6.3
nodejs	14.17.4
notebook	6.4.6
nspr	4.32
nss	3.73
numba	0.54.1
numpy	1.20.3
nvtx	0.2.3
olefile	0.46
openjpeg	2.4.0
openssl	1.1.1l
orc	1.7.1
packaging	21.3
pandas	1.3.5
pandoc	2.16.2
pandocfilters	1.5.0
panel	0.12.4
param	1.12.0
parquet-cpp	1.5.1
parso	0.8.3
partd	1.2.0
pcre	8.45
pcre2	10.37
perl	5.32.1
pexpect	4.8.0
pickle5	0.0.12
pickleshare	0.7.5
pillow	8.4.0
pip	21.3.1
pixman	0.40.0
pooch	1.5.2
poppler	21.09.0
poppler-data	0.4.11
postgresql	13.5

Library	Version
proj	8.1.0
prometheus_client	0.12.0
prompt-toolkit	3.0.24
protobuf	3.18.1
psutil	5.8.0
pthread-stubs	0.4
ptxcompiler	0.2.0
ptyprocess	0.7.0
py-xgboost	1.5.0dev.rapidsai21.12
py4j	0.10.9
pyarrow	5.0.0
pycparser	2.21
pyct	0.4.6
pyct-core	0.4.6
pydeck	0.5.0
pyee	8.1.0
pygments	2.10.0
pynvml	11.4.1
pyopenssl	21.0.0
pyparsing	3.0.6
pyppeteer	0.2.6
pyproj	3.1.0
pyrsistent	0.18.0
pysocks	1.7.1
python	3.7.12
python-confluent-kafka	1.6.0
python-dateutil	2.8.2
python_abi	3.7
pytz	2021.3
pyviz_comms	2.1.0
pywavelets	1.2.0
pyyaml	6.0
pyzmq	22.3.0
rapids	21.12.00
rapids-xgboost	21.12.00
re2	2021.11.01
readline	8.1
requests	2.26.0
rmm	21.12.00

Library	Version
rtree	0.9.7
s2n	1.3.0
scikit-image	0.18.1
scikit-learn	1.0.1
scipy	1.7.3
send2trash	1.8.0
setuptools	59.6.0
shapely	1.8.0
simpervisor	0.4
six	1.16.0
snappy	1.1.8
sniffio	1.2.0
sortedcontainers	2.4.0
spdlog	1.8.5
sqlite	3.37.0
streamz	0.6.3
tblib	1.7.0
terminado	0.12.1
testpath	0.5.0
threadpoolctl	3.0.0
tifffile	2021.11.2
tiledb	2.3.4
tk	8.6.11
toolz	0.11.2
tornado	6.1
tqdm	4.62.3
traitlets	5.1.1
treelite	2.1.0
treelite-runtime	2.1.0
typing-extensions	4.0.1
typing_extensions	4.0.1
tzcode	2021e
tzdata	2021e
ucx	1.11.2+gef2bbcf
ucx-proc	1.0.0
ucx-py	0.23.0
unicodedata2	13.0.0.post2
urllib3	1.26.6
wcwidth	0.2.5

Library	Version
webencodings	0.5.1
websocket-client	1.2.3
websockets	9.1
wheel	0.37.0
widgetsnbextension	3.5.2
xarray	0.20.2
xerces-c	3.2.3
xgboost	1.5.0dev.rapidsai21.12
xorg-kbproto	1.0.7
xorg-libice	1.0.10
xorg-libsm	1.2.3
xorg-libx11	1.7.2
xorg-libxau	1.0.9
xorg-libxdmcp	1.1.3
xorg-libxext	1.3.4
xorg-libxrender	0.9.10
xorg-renderproto	0.11.1
xorg-xextproto	7.3.0
xorg-xproto	7.0.31
xz	5.2.5
yaml	0.2.5
yarl	1.7.2
zeromq	4.3.4
zfp	0.5.5
zict	2.0.0
zipp	3.6.0
zlib	1.2.11
zstd	1.5.0

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with ML Runtimes 2021.12.

Scala 2.11.12 Libraries for Workbench - Standard

Library	Version
org.scala-lang:scala-library	2.11.12
org.scalatest:scalatest	2.2.4
org.scala-lang:scala-compiler	2.11.12
io.circe:circe-generic	0.11.2
io.circe:circe-parser	0.11.2

Library	Version
org.scalameta:scalameta	1.0.0

Python 3.9.6 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2021.12.

Table 318: Packages in Python 3.9.6 JupyterLab Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.2.8

Library	Version
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4

Library	Version
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1

Table 319: Packages in Python 3.9.6 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5

Library	Version
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0

Library	Version
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with ML Runtimes 2021.012.

Table 320: Packages in Python 3.8.6 JupyterLab Runtimes - Standard

Library	Version
python	3.8.6
pip	21.1.3
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10

Library	Version
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1

Library	Version
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1

Table 321: Packages in Python 3.8.6 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6

Library	Version
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8

Library	Version
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1

Table 322: Packages in Python 3.8.6 JupyterLab Runtimes - RAPIDS

Library	Version
python	3.8.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
_libgcc_mutex	0.1
_openmp_mutex	4.5
abseil-cpp	20210324.2
aiohttp	3.8.1
aiosignal	1.2.0
anyio	3.3.0
appdirs	1.4.4

Library	Version
argon2-cffi	20.1.0
arrow-cpp	5.0.0
arrow-cpp-proc	3.0.0
async-timeout	4.0.1
async_generator	1.10
attrs	20.3.0
aws-c-auth	0.6.7
aws-c-cal	0.5.12
aws-c-common	0.6.17
aws-c-compression	0.2.14
aws-c-event-stream	0.2.7
aws-c-http	0.6.10
aws-c-io	0.10.13
aws-c-mqtt	0.7.9
aws-c-s3	0.1.27
aws-c-sdkutils	0.1.1
aws-checksums	0.1.12
aws-crt-cpp	0.17.8
aws-sdk-cpp	1.9.145
babel	2.9.1
backcall	0.2.0
backports	1.0
backports.functools_lru_cache	1.6.4
bleach	3.3.1
blosc	1.21.0
bokeh	2.4.0
boost	1.74.0
boost-cpp	1.74.0
brotli	1.0.9
brotli-bin	1.0.9
brotlipy	0.7.0
brunsli	0.1
bzip2	1.0.8
c-ares	1.18.1
c-blosc2	2.0.4
ca-certificates	2021.10.8
cachetools	4.2.4
cairo	1.16.0
cdsw	1.1.0

Library	Version
certifi	2020.11.8
cffi	1.14.4
cfitsio	3.470
chardet	4.0.0
charls	2.2.0
charset-normalizer	2.0.6
click	8.0.3
click-plugins	1.1.1
cligj	0.7.2
cloudpickle	2.0.0
colorama	0.4.4
colorcet	3.0.0
cryptography	36.0.1
cucim	21.12.00
cudatoolkit	11.0.221
cudf	21.12.02
cudf_kafka	21.12.02
cudnn	8.2.1.32
cugraph	21.12.00
cuml	21.12.00
cupy	9.6.0
curl	7.80.0
cusignal	21.12.00
cuspatial	21.12.00
custreamz	21.12.02
cuxfilter	21.12.00
cycler	0.11.0
cyrus-sasl	2.1.27
cytoolz	0.11.2
dask	2021.11.2
dask-core	2021.11.2
dask-cuda	21.12.00
dask-cudf	21.12.02
datashader	0.11.1
datashape	0.5.4
debugpy	1.5.1
decorator	4.4.2
defusedxml	0.6.0
distributed	2021.11.2

Library	Version
dlpack	0.5
entrypoints	0.3
expat	2.4.1
faiss-proc	1.0.0
fastavro	1.4.7
fastrlock	0.8
fiona	1.8.20
font-ttf-dejavu-sans-mono	2.37
font-ttf-inconsolata	3.000
font-ttf-source-code-pro	2.038
font-ttf-ubuntu	0.83
fontconfig	2.13.1
fonts-conda-ecosystem	1
fonts-conda-forge	1
fonttools	4.28.4
freetype	2.10.4
freexl	1.0.6
frozenlist	1.2.0
fsspec	2021.11.1
gdal	3.3.2
geopandas	0.9.0
geopandas-base	0.9.0
geos	3.9.1
geotiff	1.7.0
gettext	0.19.8.1
gflags	2.2.2
giflib	5.2.1
git	2.34.1
glog	0.5.0
gpuci-tools	0.3.1
grpc-cpp	1.41.1
hdf4	4.2.15
hdf5	1.12.1
heapdict	1.0.1
icu	68.2
idna	2.10
imagecodecs	2021.8.26
imageio	2.13.3
importlib-metadata	4.8.3

Library	Version
importlib_metadata	4.8.3
importlib_resources	5.4.0
ipykernel	5.3.4
ipython	7.16.1
ipython_genutils	0.2.0
ipywidgets	7.6.5
jbig	2.1
jedi	0.17.2
jinja2	2.11.3
joblib	1.1.0
jpeg	9d
json-c	0.15
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyter-server-proxy	3.2.0
jupyterlab	3.2.8
jupyterlab-server	2.7.2
jupyterlab_pygments	0.1.2
jupyterlab_widgets	1.0.2
jxrlib	1.1
kealib	1.4.14
kiwisolver	1.3.2
krb5	1.19.2
lcms2	2.12
ld_impl_linux-64	2.36.1
lerc	3.0
libaec	1.0.6
libblas	3.9.0
libbrotlicommon	1.0.9
libbrotlidec	1.0.9
libbrotlienc	1.0.9
libcblas	3.9.0
libcucim	21.12.00
libcudf	21.12.02
libcudf_kafka	21.12.02
libcugraph	21.12.00

Library	Version
libcuml	21.12.00
libcumlprims	21.12.00
libcurl	7.80.0
libcusolver	11.3.2.107
libcuspatial	21.12.00
libdap4	3.20.6
libdeflate	1.8
libedit	3.1.20191231
libev	4.33
libevent	2.1.10
libfaiss	1.7.0
libffi	3.4.2
libgcc-ng	9.4.0
libgcrypt	1.9.4
libgdal	3.3.2
libgfortran-ng	11.2.0
libgfortran5	11.2.0
libglib	2.70.2
libgpg-error	1.42
libgsasl	1.10.0
libhwloc	2.3.0
libiconv	1.16
libkml	1.3.0
liblapack	3.9.0
libllvm11	11.1.0
libnetcdf	4.8.1
libnghttp2	1.43.0
libnsl	2.0.0
libntlm	1.4
libopenblas	0.3.18
libpng	1.6.37
libpq	13.5
libprotobuf	3.18.1
librdkafka	1.6.1
librmm	21.12.00
librttopo	1.1.0
libsodium	1.0.18
libspatialindex	1.9.3
libspatialite	5.0.1

Library	Version
libssh2	1.10.0
libstdcxx-ng	9.4.0
libthrift	0.15.0
libtiff	4.3.0
libutf8proc	2.7.0
libuuid	2.32.1
libuv	1.42.0
libwebp	1.2.1
libwebp-base	1.2.1
libxcb	1.13
libxgboost	1.5.0dev.rapidsai21.12
libxml2	2.9.12
libzip	1.8.0
libzlib	1.2.11
libzopfli	1.0.3
llvm-openmp	12.0.1
llvmlite	0.37.0
locket	0.2.0
lz4-c	1.9.3
mapclassify	2.4.3
markdown	3.3.6
markupsafe	1.1.1
matplotlib-base	3.5.1
matplotlib-inline	0.1.3
mistune	0.8.4
msgpack-python	1.0.3
multidict	5.2.0
multipledispatch	0.6.0
munch	2.5.0
munkres	1.1.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nccl	2.11.4.1
ncurses	6.2
nest-asyncio	1.5.1
networkx	2.6.3
nodejs	14.17.4

Library	Version
notebook	6.4.3
nspr	4.32
nss	3.73
numba	0.54.1
numpy	1.20.3
nvtx	0.2.3
olefile	0.46
openjpeg	2.4.0
openssl	1.1.1l
orc	1.7.1
packaging	20.9
pandas	1.3.5
pandoc	2.16.2
pandocfilters	1.4.3
panel	0.12.4
param	1.12.0
parquet-cpp	1.5.1
parso	0.7.1
partd	1.2.0
pcre	8.45
pcre2	10.37
perl	5.32.1
pexpect	4.8.0
pickleshare	0.7.5
pillow	8.4.0
pip	21.3.1
pixman	0.40.0
pooch	1.5.2
poppler	21.09.0
poppler-data	0.4.11
postgresql	13.5
proj	8.1.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
protobuf	3.18.1
psutil	5.8.0
pthread-stubs	0.4
ptxcompiler	0.2.0
ptyprocess	0.6.0

Library	Version
py-xgboost	1.5.0dev.rapidsai21.12
py4j	0.10.9
pyarrow	5.0.0
pycparser	2.20
pyct	0.4.6
pyct-core	0.4.6
pydeck	0.5.0
pyee	8.1.0
pygments	2.8.1
pynvml	11.4.1
pyopenssl	21.0.0
pyparsing	2.4.7
pyppeteer	0.2.6
pyproj	3.1.0
pyrsistent	0.17.3
pysocks	1.7.1
python	3.8.12
python-confluent-kafka	1.6.0
python-dateutil	2.8.1
python_abi	3.8
pytz	2021.1
pyviz_comms	2.1.0
pywavelets	1.2.0
pyyaml	6.0
pyzmq	19.0.2
rapids	21.12.00
rapids-xgboost	21.12.00
re2	2021.11.01
readline	8.1
requests	2.25.1
requests-unixsocket	0.2.0
rmm	21.12.00
rtree	0.9.7
s2n	1.3.0
scikit-image	0.18.1
scikit-learn	1.0.1
scipy	1.7.3
send2trash	1.5.0
setuptools	59.6.0

Library	Version
shapely	1.8.0
simpervisor	0.4
six	1.15.0
snappy	1.1.8
sniffio	1.2.0
sortedcontainers	2.4.0
spdlog	1.8.5
sqlite	3.37.0
streamz	0.6.3
tblib	1.7.0
terminado	0.9.2
testpath	0.4.4
threadpoolctl	3.0.0
tifffile	2021.11.2
tiledb	2.3.4
tk	8.6.11
toolz	0.11.2
tornado	6.1
tqdm	4.62.3
traitlets	4.3.3
treelite	2.1.0
treelite-runtime	2.1.0
typing-extensions	4.0.1
typing_extensions	4.0.1
tzcode	2021e
tzdata	2021e
ucx	1.11.2+gef2bbcf
ucx-proc	1.0.0
ucx-py	0.23.0
unicodedata2	13.0.0.post2
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
websockets	9.1
wheel	0.37.0
widgetsnbextension	3.5.2
xarray	0.20.2
xerces-c	3.2.3

Library	Version
xgboost	1.5.0dev.rapidsai21.12
xorg-kbproto	1.0.7
xorg-libice	1.0.10
xorg-libsm	1.2.3
xorg-libx11	1.7.2
xorg-libxau	1.0.9
xorg-libxdmcp	1.1.3
xorg-libxext	1.3.4
xorg-libxrender	0.9.10
xorg-renderproto	0.11.1
xorg-xextproto	7.3.0
xorg-xproto	7.0.31
xz	5.2.5
yaml	0.2.5
yarl	1.7.2
zeromq	4.3.4
zfp	0.5.5
zict	2.0.0
zipp	3.6.0
zlib	1.2.11
zstd	1.5.0

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with ML Runtimes 2021.12.

Table 323: Packages in Python 3.7.11 JupyterLab Runtimes - Standard

Library	Version
python	3.7.11
pip	21.1.3
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8

Library	Version
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
importlib-metadata	3.4.0
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
Pillow	8.4.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.4.0

Table 324: Packages in Python 3.7.11 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.7.11
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0

Library	Version
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.1
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
charset-normalizer	2.0.6
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
importlib-metadata	3.4.0
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.2.8
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7

Library	Version
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.4.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.4.0

Table 325: Packages in Python 3.7.11 JupyterLab Runtimes - RAPIDS

Library	Version
python	3.7.11
pip	21.1.3
_libgcc_mutex	0.1
_openmp_mutex	4.5
abseil-cpp	20210324.2
aiohttp	3.8.1
aiosignal	1.2.0
anyio	3.3.0
appdirs	1.4.4
argcomplete	1.12.3
argon2-cffi	20.1.0
arrow-cpp	5.0.0
arrow-cpp-proc	3.0.0
async-timeout	4.0.1
async_generator	1.10
asynctest	0.13.0
attrs	20.3.0
aws-c-auth	0.6.7
aws-c-cal	0.5.12
aws-c-common	0.6.17
aws-c-compression	0.2.14
aws-c-event-stream	0.2.7
aws-c-http	0.6.10
aws-c-io	0.10.13
aws-c-mqtt	0.7.9
aws-c-s3	0.1.27
aws-c-sdkutils	0.1.1
aws-checksums	0.1.12
aws-crt-cpp	0.17.8
aws-sdk-cpp	1.9.145
babel	2.9.1
backcall	0.2.0
backports	1.0
backports.functools_lru_cache	1.6.4
bleach	3.3.1
blosc	1.21.0
bokeh	2.4.0
boost	1.74.0

Library	Version
boost-cpp	1.74.0
brotli	1.0.9
brotli-bin	1.0.9
brotlipy	0.7.0
brunsli	0.1
bzip2	1.0.8
c-ares	1.18.1
c-blosc2	2.0.4
ca-certificates	2021.10.8
cachetools	4.2.4
cairo	1.16.0
cdsw	1.1.0
certifi	2020.11.8
cffi	1.14.4
cfitsio	3.470
chardet	4.0.0
charls	2.2.0
charset-normalizer	2.0.6
click	8.0.3
click-plugins	1.1.1
cligj	0.7.2
cloudpickle	2.0.0
colorama	0.4.4
colorcet	3.0.0
cryptography	36.0.1
cucim	21.12.00
cudatoolkit	11.0.221
cudf	21.12.02
cudf_kafka	21.12.02
cudnn	8.2.1.32
cugraph	21.12.00
cuml	21.12.00
cupy	9.6.0
curl	7.80.0
cusignal	21.12.00
cuspatial	21.12.00
custreamz	21.12.02
cuxfilter	21.12.00
cycler	0.11.0

Library	Version
cyrus-sasl	2.1.27
cytoolz	0.11.2
dask	2021.11.2
dask-core	2021.11.2
dask-cuda	21.12.00
dask-cudf	21.12.02
datashader	0.11.1
datashape	0.5.4
debugpy	1.5.1
decorator	4.4.2
defusedxml	0.6.0
distributed	2021.11.2
dlpack	0.5
entrypoints	0.3
expat	2.4.1
faiss-proc	1.0.0
fastavro	1.4.7
fastrlock	0.8
fiona	1.8.20
font-ttf-dejavu-sans-mono	2.37
font-ttf-inconsolata	3.000
font-ttf-source-code-pro	2.038
font-ttf-ubuntu	0.83
fontconfig	2.13.1
fonts-conda-ecosystem	1
fonts-conda-forge	1
fonttools	4.28.4
freetype	2.10.4
freexl	1.0.6
frozenlist	1.2.0
fsspec	2021.11.1
gdal	3.3.2
geopandas	0.9.0
geopandas-base	0.9.0
geos	3.9.1
geotiff	1.7.0
gettext	0.19.8.1
gflags	2.2.2
giflib	5.2.1

Library	Version
git	2.34.1
glog	0.5.0
gpuci-tools	0.3.1
grpc-cpp	1.41.1
hdf4	4.2.15
hdf5	1.12.1
heapdict	1.0.1
icu	68.2
idna	2.10
imagecodecs	2021.8.26
imageio	2.13.3
importlib-metadata	3.4.0
importlib_metadata	4.8.3
importlib_resources	5.4.0
ipykernel	5.3.4
ipython	7.16.1
ipython_genutils	0.2.0
ipywidgets	7.6.5
jbig	2.1
jedi	0.17.2
jinja2	2.11.3
joblib	1.1.0
jpeg	9d
json-c	0.15
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyter-server-proxy	3.2.0
jupyterlab	3.2.8
jupyterlab-server	2.7.2
jupyterlab_pygments	0.1.2
jupyterlab_widgets	1.0.2
jxrlib	1.1
kealib	1.4.14
kiwisolver	1.3.2
krb5	1.19.2
lcms2	2.12

Library	Version
ld_impl_linux-64	2.36.1
lerc	3.0
libaec	1.0.6
libblas	3.9.0
libbrotlicommon	1.0.9
libbrotlidec	1.0.9
libbrotlienc	1.0.9
libcblas	3.9.0
libcucim	21.12.00
libcudf	21.12.02
libcudf_kafka	21.12.02
libcugraph	21.12.00
libcuml	21.12.00
libcumlprims	21.12.00
libcurl	7.80.0
libcusolver	11.3.2.107
libcuspatial	21.12.00
libdap4	3.20.6
libdeflate	1.8
libedit	3.1.20191231
libev	4.33
libevent	2.1.10
libfaiss	1.7.0
libffi	3.4.2
libgcc-ng	9.4.0
libgcrypt	1.9.4
libgdal	3.3.2
libgfortran-ng	11.2.0
libgfortran5	11.2.0
libglib	2.70.2
libgpg-error	1.42
libgsasl	1.10.0
libhwloc	2.3.0
libiconv	1.16
libkml	1.3.0
liblapack	3.9.0
libllvm11	11.1.0
libnetcdf	4.8.1
libnghttp2	1.43.0

Library	Version
libnsl	2.0.0
libntlm	1.4
libopenblas	0.3.18
libpng	1.6.37
libpq	13.5
libprotobuf	3.18.1
librdkafka	1.6.1
librmm	21.12.00
librttopo	1.1.0
libsodium	1.0.18
libspatialindex	1.9.3
libspatialite	5.0.1
libssh2	1.10.0
libstdcxx-ng	9.4.0
libthrift	0.15.0
libtiff	4.3.0
libutf8proc	2.7.0
libuuid	2.32.1
libuv	1.42.0
libwebp	1.2.1
libwebp-base	1.2.1
libxcb	1.13
libxgboost	1.5.0dev.rapidsai21.12
libxml2	2.9.12
libzip	1.8.0
libzlib	1.2.11
libzopfli	1.0.3
llvm-openmp	12.0.1
llvmlite	0.37.0
locket	0.2.0
lz4-c	1.9.3
mapclassify	2.4.3
markdown	3.3.6
markupsafe	1.1.1
matplotlib-base	3.5.1
matplotlib-inline	0.1.3
mistune	0.8.4
msgpack-python	1.0.3
multidict	5.2.0

Library	Version
multipledispatch	0.6.0
munch	2.5.0
munkres	1.1.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nccl	2.11.4.1
ncurses	6.2
nest-asyncio	1.5.1
networkx	2.6.3
nodejs	14.17.4
notebook	6.4.3
nspr	4.32
nss	3.73
numba	0.54.1
numpy	1.20.3
nvtx	0.2.3
olefile	0.46
openjpeg	2.4.0
openssl	1.1.1l
orc	1.7.1
packaging	20.9
pandas	1.3.5
pandoc	2.16.2
pandocfilters	1.4.3
panel	0.12.4
param	1.12.0
parquet-cpp	1.5.1
parso	0.7.1
partd	1.2.0
pcre	8.45
pcre2	10.37
perl	5.32.1
pexpect	4.8.0
pickle5	0.0.12
pickleshare	0.7.5
pillow	8.4.0
pip	21.3.1

Library	Version
pixman	0.40.0
pooch	1.5.2
poppler	21.09.0
poppler-data	0.4.11
postgresql	13.5
proj	8.1.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
protobuf	3.18.1
psutil	5.8.0
pthread-stubs	0.4
ptxcompiler	0.2.0
ptyprocess	0.6.0
py-xgboost	1.5.0dev.rapidsai21.12
py4j	0.10.9
pyarrow	5.0.0
pycparser	2.20
pyct	0.4.6
pyct-core	0.4.6
pydeck	0.5.0
pyee	8.1.0
pygments	2.8.1
pynvml	11.4.1
pyopenssl	21.0.0
pyparsing	2.4.7
pyppeteer	0.2.6
pyproj	3.1.0
pyrsistent	0.17.3
pysocks	1.7.1
python	3.7.12
python-confluent-kafka	1.6.0
python-dateutil	2.8.1
python_abi	3.7
pytz	2021.1
pyviz_comms	2.1.0
pywavelets	1.2.0
pyyaml	6.0
pyzmq	19.0.2
rapids	21.12.00

Library	Version
rapids-xgboost	21.12.00
re2	2021.11.01
readline	8.1
requests	2.25.1
requests-unixsocket	0.2.0
rmm	21.12.00
rtree	0.9.7
s2n	1.3.0
scikit-image	0.18.1
scikit-learn	1.0.1
scipy	1.7.3
send2trash	1.5.0
setuptools	59.6.0
shapely	1.8.0
simpervisor	0.4
six	1.15.0
snappy	1.1.8
sniffio	1.2.0
sortedcontainers	2.4.0
spdlog	1.8.5
sqlite	3.37.0
streamz	0.6.3
tblib	1.7.0
terminado	0.9.2
testpath	0.4.4
threadpoolctl	3.0.0
tifffile	2021.11.2
tiledb	2.3.4
tk	8.6.11
toolz	0.11.2
tornado	6.1
tqdm	4.62.3
traitlets	4.3.3
treelite	2.1.0
treelite-runtime	2.1.0
typing-extensions	3.7.4.3
tzcode	2021e
tzdata	2021e
ucx	1.11.2+gef2bbcf

Library	Version
ucx-proc	1.0.0
ucx-py	0.23.0
unicodedata2	13.0.0.post2
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
websockets	9.1
wheel	0.37.0
widgetsnbextension	3.5.2
xarray	0.20.2
xerces-c	3.2.3
xgboost	1.5.0dev.rapidsai21.12
xorg-kbproto	1.0.7
xorg-libice	1.0.10
xorg-libsm	1.2.3
xorg-libx11	1.7.2
xorg-libxau	1.0.9
xorg-libxdmcp	1.1.3
xorg-libxext	1.3.4
xorg-libxrender	0.9.10
xorg-renderproto	0.11.1
xorg-xextproto	7.3.0
xorg-xproto	7.0.31
xz	5.2.5
yaml	0.2.5
yarl	1.7.2
zeromq	4.3.4
zfp	0.5.5
zict	2.0.0
zipp	3.4.0
zlib	1.2.11
zstd	1.5.0

R 4.0 Libraries

This section lists the R 4.0 libraries that ship with ML Runtimes 2021.12.

Table 326: R 4.0.5 Libraries for Workbench

Library	Version
askpass	1.1
base	4.0.5
bitops	1.0-7
boot	1.3-27
brew	1.0-6
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-18
cli	3.0.1
cluster	2.1.1
codetools	0.2-18
colorspace	2.0-2
compiler	4.0.5
crayon	1.4.1
curl	4.3.2
datasets	4.0.5
digest	0.6.27
ellipsis	0.3.2
fansi	0.5.0
farver	2.1.0
foreign	0.8-81
ggplot2	3.3.5
glue	1.4.2
graphics	4.0.5
grDevices	4.0.5
grid	4.0.5
gtable	0.3.0
httr	1.4.2
jsonlite	1.7.2
KernSmooth	2.23-18
labeling	0.4.2
lattice	0.20-41
lifecycle	1.0.0
magrittr	2.0.1
MASS	7.3-53.1
Matrix	1.3-2

Library	Version
methods	4.0.5
mgcv	1.8-34
mime	0.11
munsell	0.5.0
nlme	3.1-152
nnet	7.3-15
openssl	1.4.4
parallel	4.0.5
pillar	1.6.2
pkgconfig	2.0.3
png	0.1-7
R6	2.5.1
RColorBrewer	1.1-2
RJSONIO	1.3-1.5
rlang	0.4.11
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.1
spatial	7.3-13
splines	4.0.5
stats	4.0.5
stats4	4.0.5
survival	3.2-10
sys	3.4
tcltk	4.0.5
tibble	3.1.4
tools	4.0.5
utf8	1.2.2
utils	4.0.5
vctrs	0.3.8
viridisLite	0.4.0
withr	2.4.2

R 4.1 Libraries

This section lists the R 4.1 libraries that ship with ML Runtimes 2021.12.

Table 327: R 4.0.5 Libraries for Workbench

Library	Version
askpass	1.1
base	4.1.1
bitops	1.0-7
boot	1.3-28
brew	1.0-6
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-19
cli	3.0.1
cluster	2.1.2
codetools	0.2-18
colorspace	2.0-2
compiler	4.1.1
crayon	1.3.4
curl	4.3.2
datasets	4.1.1
digest	0.6.27
ellipsis	0.3.2
fansi	0.5.0
farver	2.1.0
foreign	0.8-81
ggplot2	3.3.5
glue	1.4.2
graphics	4.1.1
grDevices	4.1.1
grid	4.1.1
gtable	0.3.0
httr	1.4.2
jsonlite	1.7.2
KernSmooth	2.23-20
labeling	0.4.2
lattice	0.20-44
lifecycle	1.0.0
magrittr	2.0.1
MASS	7.3-54
Matrix	1.3-4

Library	Version
methods	4.1.1
mgcv	1.8-36
mime	0.11
munsell	0.5.0
nlme	3.1-152
nnet	7.3-16
openssl	1.4.4
parallel	4.1.1
pillar	1.6.2
pkgconfig	2.0.3
png	0.1-7
R6	2.5.1
RColorBrewer	1.1-2
RJSONIO	1.3-1.5
rlang	0.4.11
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.1
spatial	7.3-14
splines	4.1.1
stats	4.1.1
stats4	4.1.1
survival	3.2-11
sys	3.4
tcltk	4.1.1
tibble	3.1.4
tools	4.1.1
utf8	1.2.2
utils	4.1.1
vctrs	0.3.8
viridisLite	0.4.0
withr	2.4.2

R 3.6 Libraries

This section lists the R 3.6 libraries that ship with ML Runtimes 2021.12.

Table 328: R 3.6.3 Libraries for Workbench

Library	Version
askpass	1.1
base	3.6.3
bitops	1.0-7
boot	1.3-24
brew	1.0-6
Cairo	1.2-0
caTools	1.18.2
cdsw	2.1
class	7.3-15
cli	3.0.1
cluster	2.1.0
codetools	0.2-16
colorspace	2.0-2
compiler	3.6.3
crayon	1.4.4
curl	4.3.2
datasets	3.6.3
digest	0.6.27
ellipsis	0.3.2
fansi	0.5.0
farver	2.1.0
foreign	0.8-75
ggplot2	3.3.5
glue	1.4.2
graphics	3.6.3
grDevices	3.6.3
grid	3.6.3
gtable	0.3.0
httr	1.4.2
jsonlite	1.7.2
KernSmooth	2.23-16
labeling	0.4.2
lattice	0.20-38
lifecycle	0.1.0
magrittr	2.0.1
MASS	7.3-51.5
Matrix	1.2-18

Library	Version
methods	3.6.3
mgcv	1.8-31
mime	0.11
munsell	0.5.0
nlme	3.4-144
nnet	7.3-12
openssl	1.4.4
parallel	3.6.3
pillar	1.6.2
pkgconfig	2.0.3
png	0.1-7
R6	2.5.1
RColorBrewer	1.1-2
RJSONIO	1.3-1.5
rlang	0.4.11
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.1
spatial	7.3-11
splines	3.6.3
stats	3.6.3
stats4	3.6.3
survival	3.1-8
sys	3.4
tcltk	3.6.3
tibble	3.1.4
tools	3.6.3
utf8	1.2.2
utils	3.6.3
vctrs	0.3.8
viridisLite	0.4.0
withr	2.4.2

ML Runtimes 2021.09

This section lists the Python, R, and Scala libraries that ship with ML Runtimes 2021.09.

Note: Nvidia GPU Edition comes with CUDA 11.4.1 preinstalled.

Python 3.9 Libraries for Workbench

This section lists the packages in Python 3.9 Workbench Runtimes that ship with ML Runtimes 2021.09.

Table 329: Packages in Python 3.9.6 Workbench Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0

Library	Version
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Table 330: Packages in Python 3.9.6 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1

Library	Version
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Python 3.8 Libraries for Workbench

This section lists the packages in Python 3.8 Workbench Runtimes that ship with ML Runtimes 2021.09.

Table 331: Packages in Python 3.8.6 Workbench Runtimes - Standard

Library	Version
python	3.8.6
pip	21.1.3
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4

Library	Version
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Table 332: Packages in Python 3.8.6 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2

Library	Version
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Python 3.7 Libraries for Workbench

This section lists the packages in Python 3.7 Workbench Runtimes that ship with ML Runtimes 2021.09.

Table 333: Packages in Python 3.7.11 Workbench Runtimes - Standard

Library	Version
python	3.7.11
pip	21.1.3
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0

Library	Version
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Table 334: Packages in Python 3.7.11 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.7.11
pip	21.1.3

Library	Version
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6

Library	Version
wcwidth	0.2.5

Python 3.6 Libraries for Workbench

This section lists the packages in Python 3.6 Workbench Runtimes that ship with ML Runtimes 2021.09.

Table 335: Packages in Python 3.6.14 Workbench Runtimes - Standard

Library	Version
python	3.6.14
pip	21.1.3
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1

Library	Version
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Table 336: Packages in Python 3.6.14 Workbench Runtimes - Nvidia GPU Edition

Library	Version
python	3.6.14
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kerberos	1.3.1
kiwisolver	1.3.1
matplotlib	3.3.4
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prompt-toolkit	3.0.8

Library	Version
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5

Scala 2.11 Libraries for Workbench

This section lists the packages in Scala 2.11.12 Workbench Runtimes that ship with ML Runtimes 2021.09.

Scala 2.11.12 Libraries for Workbench - Standard

- org.scala-lang:scala-library:2.11.12
- org.scalatest:scalatest:2.2.4:test
- org.scala-lang:scala-reflect:2.11.12
- org.scala-lang:scala-compiler:2.11.12
- io.circe:circe-generic:0.11.2
- io.circe:circe-parser:0.11.2
- org.scalameta:scalameta:1.0.0

Python 3.9.6 Libraries for JupyterLab

This section lists the packages in Python 3.9 JupyterLab Runtimes that ship with ML Runtimes 2021.09.

Table 337: Packages in Python 3.9.6 JupyterLab Runtimes - Standard

Library	Version
python	3.9.6
pip	21.1.3
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0

Library	Version
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.0.17
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9

Library	Version
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1

Table 338: Packages in Python 3.9.6 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.9.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1

Library	Version
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.0.17
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3

Library	Version
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1

Python 3.8 Libraries for JupyterLab

This section lists the packages in Python 3.8 JupyterLab Runtimes that ship with ML Runtimes 2021.09.

Table 339: Packages in Python 3.8.6 JupyterLab Runtimes - Standard

Library	Version
python	3.8.6
pip	21.1.3
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.0.17
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4

Library	Version
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1

Library	Version
websocket-client	1.2.1

Table 340: Packages in Python 3.8.6 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.8.6
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.0.17
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1

Library	Version
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1

Library	Version
traitlets	4.3.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1

Python 3.7 Libraries for JupyterLab

This section lists the packages in Python 3.7 JupyterLab Runtimes that ship with ML Runtimes 2021.09.

Table 341: Packages in Python 3.7.11 JupyterLab Runtimes - Standard

Library	Version
python	3.7.11
pip	21.1.3
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
importlib-metadata	3.4.0
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0

Library	Version
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.0.17
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0

Library	Version
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.4.0

Table 342: Packages in Python 3.7.11 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.7.11
pip	21.1.3
cuda	11.4.1-1
libcudnn8	8.2.2.26-1
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
importlib-metadata	3.4.0
impyla	0.17.0

Library	Version
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.0.17
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7

Library	Version
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.4.0

Python 3.6 Libraries for JupyterLab

This section lists the packages in Python 3.6 JupyterLab Runtimes that ship with ML Runtimes 2021.09.

Table 343: Packages in Python 3.6.14 JupyterLab Runtimes - Standard

Library	Version
python	3.6.14
pip	21.1.3
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8

Library	Version
cffi	1.14.4
chardet	4.0.0
contextvars	2.4
cycler	0.10.0
dataclasses	0.8
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
immutables	0.14
importlib-metadata	3.4.0
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.0.17
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3

Library	Version
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.6
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.4.0

Table 344: Packages in Python 3.6.14 JupyterLab Runtimes - Nvidia GPU Edition

Library	Version
python	3.6.14
pip	21.1.3
cuda	11.4.1-1

Library	Version
libcudnn8	8.2.2.26-1
anyio	3.3.0
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.1
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
contextvars	2.4
cycler	0.10.0
dataclasses	0.8
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
immutables	0.14
importlib-metadata	3.4.0
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.10.2
jupyterlab	3.0.17
jupyterlab-pygments	0.1.2
jupyterlab-server	2.7.2
kerberos	1.3.1
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.4

Library	Version
mistune	0.8.4
nbclassic	0.3.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.3
nest-asyncio	1.5.1
notebook	6.4.3
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.3.1
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
requests-unixsocket	0.2.0
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.6

Library	Version
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.2.1
zipp	3.4.0

R 4.0 Libraries

This section lists the R 4.0 libraries that ship with ML Runtimes 2021.09.

Table 345: R 4.0.5 Libraries for Workbench

Library	Version
askpass	1.1
assertthat	0.2.1
base	4.0.5
bitops	1.0-6
boot	1.3-27
brew	1.0-6
Cairo	1.2-0
caTools	1.18.0
cdsw	2.1
class	7.3-18
cli	2.0.1
cluster	2.1.1
codetools	0.2-18
colorspace	1.4-1
compiler	4.0.5
crayon	1.3.4
curl	4.3
datasets	4.0.5
digest	0.6.23
ellipsis	0.3.0
fansi	0.4.1
farver	2.0.3
foreigns	0.8-81
ggplot2	3.2.1
glue	1.3.1
graphics	4.0.5
grDevices	4.0.5
grid	4.0.5

Library	Version
gtable	0.3.0
httr	1.4.1
jsonlite	1.6
KernSmooth	2.23-18
labeling	0.3
lattice	0.20-41
lazyeval	0.2.2
lifecycle	0.1.0
magrittr	1.5
MASS	7.3-53.1
Matrix	1.3-2
methods	4.0.5
mgcv	1.8-34
mime	0.8
munsell	0.5.0
nlme	3.1-152
nnet	7.3-15
openssl	1.4.1
parallel	4.0.5
pillar	1.4.3
pkgconfig	2.0.3
plyr	1.8.5
png	0.1-7
R6	2.4.1
RColorBrewer	1.1-2
Rcpp	1.0.3
reshape2	1.4.3
RJSONIO	1.3-1.4
rlang	0.4.4
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.0
spatial	7.3-13
splines	4.0.5
stats	4.0.5
stats4	4.0.5
stringi	1.4.5
stringr	1.4.0

Library	Version
survival	3.2-10
sys	3.3
tcltk	4.0.5
tibble	2.1.3
tools	4.0.5
utf8	1.1.4
utils	4.0.5
vctrs	0.2.2
viridisLite	0.3.0
withr	2.1.2

R 3.6 Libraries

This section lists the R 3.6 libraries that ship with ML Runtimes 2021.09.

Table 346: R 3.6.3 Libraries for Workbench

Library	Version
askpass	1.1
assertthat	0.2.1
base	3.6.3
bitops	1.0-6
boot	1.3-24
brew	1.0-6
Cairo	1.2-0
caTools	1.18.0
cdsw	2.1
class	7.3-15
cli	2.0.1
cluster	2.1.0
codetools	0.2-16
colorspace	1.4-1
compiler	3.6.3
crayon	1.3.4
curl	4.3
datasets	3.6.3
digest	0.6.23
ellipsis	0.3.0
fansi	0.4.1
farver	2.0.3

Library	Version
foreigns	0.8-75
ggplot2	3.2.1
glue	1.3.1
graphics	3.6.3
grDevices	3.6.3
grid	3.6.3
gtable	0.3.0
httr	1.4.1
jsonlite	1.6
KernSmooth	2.23-16
labeling	0.3
lattice	0.20-38
lazyeval	0.2.2
lifecycle	0.1.0
magrittr	1.5
MASS	7.3-51.5
Matrix	1.2-18
methods	3.6.3
mgcv	1.8-31
mime	0.8
munsell	0.5.0
nlme	3.4-144
nnet	7.3-12
openssl	1.4.1
parallel	3.6.3
pillar	1.4.3
pkgconfig	2.0.3
plyr	1.8.5
png	0.1-7
R6	2.4.1
RColorBrewer	1.1-2
Rcpp	1.0.3
reshape2	1.4.3
RJSONIO	1.3-1.4
rlang	0.4.4
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.0

Library	Version
spatial	7.3-11
splines	3.6.3
stats	3.6.3
stats4	3.6.3
stringi	1.4.5
stringr	1.4.0
survival	3.1-8
sys	3.3
tcltk	3.6.3
tibble	2.1.3
tools	3.6.3
utf8	1.1.4
utils	3.6.3
vctrs	0.2.2
viridisLite	0.3.0
withr	2.1.2

ML Runtimes 2021.06

This section lists the Python and R libraries that ship with ML Runtimes 2021.06.

Note: Nvidia GPU Edition comes with CUDA 11.1 preinstalled.

Python 3.8.6 Libraries for Workbench

This section lists the Python 3.8.6 libraries for Workbench that ship with ML Runtimes 2021.06.

Table 347: Python 3.8.6 Libraries for Workbench - Standard

Library	Version
backcall	0.2.0
bitarray	2.10.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4

Library	Version
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Table 348: Python 3.8.6 Libraries for Workbench - Nvidia GPU Edition

Library	Version
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0

Library	Version
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Python 3.7.9 Libraries for Workbench

This section lists the Python 3.7.9 libraries for Workbench that ship with ML Runtimes 2021.06.

Table 349: Python 3.7.9 Libraries for Workbench - Standard

Library	Version
backcall	0.2.0
bitarray	2.10.0
certifi	2020.12.5

Library	Version
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Table 350: Python 3.7.9 Libraries for Workbench - Nvidia GPU Edition

Library	Version
backcall	0.2.0
bitarray	2.1.0

Library	Version
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Python 3.6.12 Libraries for Workbench

This section lists the Python 3.6.12 Scala libraries for Workbench that ship with ML Runtimes 2021.06.

Table 351: Python 3.6.12 Libraries for Workbench - Standard

Library	Version
backcall	0.2.0
bitarray	2.10.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Table 352: Python 3.6.12 Libraries for Workbench - Nvidia GPU Edition

Library	Version
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Python 3.8.6 Libraries for JupyterLab

This section lists the Python 3.8.6 libraries for JupyterLab that ship with ML Runtimes 2021.06.

Table 353: Python 3.8.6 Libraries for JupyterLab - Standard

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6

Library	Version
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1

Table 354: Python 3.8.6 Libraries for JupyterLab - Nvidia GPU Edition

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2

Library	Version
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1

Python 3.7.9 Libraries for JupyterLab

This section lists the Python 3.7.9 libraries for JupyterLab that ship with ML Runtimes 2021.06.

Table 355: Python 3.7.9 Libraries for JupyterLab - Standard

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
importlib-metadata	3.4.0
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6
nbclient	0.5.1

Library	Version
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.4.0

Table 356: Python 3.7.9 Libraries for JupyterLab - Nvidia GPU Edition

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
importlib-metadata	3.4.0
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7

Library	Version
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.4.0

Python 3.6.12 Libraries for JupyterLab

This section lists the Python 3.6.12 libraries for JupyterLab that ship with ML Runtimes 2021.06.

Table 357: Python 3.6.12 Libraries for JupyterLab - Standard

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
contextvars	2.4
cycler	0.10.0
dataclasses	0.8
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
immutables	0.14
importlib-metadata	3.4.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1

Library	Version
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
py4j	0.10.9.1
pycparser	2.20
Pygments	2.7.2
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.4.0

Table 358: Python 3.6.12 Libraries for JupyterLab - Nvidia GPU Edition

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
contextvars	2.4
cycler	0.10.0
dataclasses	0.8
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
immutables	0.14
importlib-metadata	3.4.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6
nbclient	0.5.1

Library	Version
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
py4j	0.10.9.1
pycparser	2.20
Pygments	2.7.2
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.4.0

R 4.0 Libraries

This section lists the R 4.0 libraries that ship with ML Runtimes 2021.06.

Table 359: R 4.0 Libraries for Workbench

Library	Version
askpass	1.1
assertthat	0.2.1
base	4.0.4
bitops	1.0-6
boot	1.3-26
brew	1.0-6
Cairo	1.2-0
caTools	1.18.0
cdsw	2.1
class	7.3-18
cli	2.0.1
cluster	2.1.0
codetools	0.2-18
colorspace	1.4-1
compiler	4.0.4
crayon	1.3.4
curl	4.3
datasets	4.0.4
digest	0.6.23
ellipsis	0.3.0
fansi	0.4.1
farver	2.0.3
foreigns	0.8-81
ggplot2	3.2.1
glue	1.3.1
graphics	4.0.4
grDevices	4.0.4
grid	4.0.4
gtable	0.3.0
httr	1.4.1
jsonlite	1.6
KernSmooth	2.23-18
labeling	0.3
lattice	0.20-41
lazyeval	0.2.2
lifecycle	0.1.0
magrittr	1.5

Library	Version
MASS	7.3-53
Matrix	1.3-2
methods	4.0.4
mgcv	1.8-33
mime	0.8
munsell	0.5.0
nlme	3.1-152
nnet	7.3-15
openssl	1.4.1
parallel	4.0.4
pillar	1.4.3
pkgconfig	2.0.3
plyr	1.8.5
png	0.1-7
R6	2.4.1
RColorBrewer	1.1-2
Rcpp	1.0.3
reshape2	1.4.3
RJSONIO	1.3-1.4
rlang	0.4.4
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.0
spatial	7.3-13
splines	4.0.4
stats	4.0.4
stats4	4.0.4
stringi	1.4.5
stringr	1.4.0
survival	3.2-7
sys	3.3
tcltk	4.0.4
tibble	2.1.3
tools	4.0.4
utf8	1.1.4
utils	4.0.4
vctrs	0.2.2
viridisLite	0.3.0

Library	Version
withr	2.1.2

R 3.6 Libraries

This section lists the R 3.6 libraries that ship with ML Runtimes 2021.06.

Table 360: R 3.6 Libraries for Workbench

Library	Version
askpass	1.1
assertthat	0.2.1
base	3.6.3
bitops	1.0-6
boot	1.3-24
brew	1.0-6
Cairo	1.2-0
caTools	1.18.0
cdsw	2.1
class	7.3-15
cli	2.0.1
cluster	2.1.0
codetools	0.2-16
colorspace	1.4-1
compiler	3.6.3
crayon	1.3.4
curl	4.3
datasets	3.6.3
digest	0.6.23
ellipsis	0.3.0
fansi	0.4.1
farver	2.0.3
foreigns	0.8-75
ggplot2	3.2.1
glue	1.3.1
graphics	3.6.3
grDevices	3.6.3
grid	3.6.3
gtable	0.3.0
httr	1.4.1
jsonlite	1.6

Library	Version
KernSmooth	2.23-16
labeling	0.3
lattice	0.20-38
lazyeval	0.2.2
lifecycle	0.1.0
magrittr	1.5
MASS	7.3-51.5
Matrix	1.2-18
methods	3.6.3
mgcv	1.8-31
mime	0.8
munsell	0.5.0
nlme	3.4-144
nnet	7.3-12
openssl	1.4.1
parallel	3.6.3
pillar	1.4.3
pkgconfig	2.0.3
plyr	1.8.5
png	0.1-7
R6	2.4.1
RColorBrewer	1.1-2
Rcpp	1.0.3
reshape2	1.4.3
RJSONIO	1.3-1.4
rlang	0.4.4
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.0
spatial	7.3-11
splines	3.6.3
stats	3.6.3
stats4	3.6.3
stringi	1.4.5
stringr	1.4.0
survival	3.1-8
sys	3.3
tcltk	3.6.3

Library	Version
tibble	2.1.3
tools	3.6.3
utf8	1.1.4
utils	3.6.3
vctrs	0.2.2
viridisLite	0.3.0
withr	2.1.2

ML Runtimes 2021.04

This section lists the Python and R libraries that ship with ML Runtimes 2021.04.

Note: Nvidia GPU Edition comes with CUDA 11.1 preinstalled.

RAPIDS Runtime PIP Python 3.7.8 Libraries for Workbench

This section lists the RAPIDS Runtime Python 3.8 libraries for Workbench that ship with ML Runtimes 2021.04.

Table 361: RAPIDS Runtime Python 3.7.8 Libraries for Workbench - Standard

Package	Version
aiohttp	3.7.3
appdirs	1.4.4
argon2-cffi	20.1.0
async-generator	1.10
async-timeout	3.0.1
attrs	20.3.0
bleach	3.3.0
bokeh	2.2.3
brotlipy	0.7.0
cdsw	1.0.0
certifi	2020.12.5
cffi	1.14.5
chardet	4.0.0
click	7.1.2
click-plugins	1.1.1
cligj	0.7.1
cloudpickle	1.6.0
colorcet	2.0.6

Package	Version
confluent-kafka	1.5.0
cryptography	3.4.4
cudf	0.18.0a0+254.g1544474166
cudf-kafka	0.18.0a0+254.g1544474166
cugraph	0.18.0a0+301.g0cc951f7.dirty
cuml	0.18.0a0+134.g7ca2db287
cupy	8.0.0
cusignal	0.18.0a0+33.gbc8598c
cuspatial	0.18.0a0+24.g1a2fc18
custreamz	0.18.0a0+254.g1544474166
cuxfilter	0.18.0a0+68.g236aa39
cycler	0.10.0
cytoolz	0.11.0
dask	2021.2.0
dask-cuda	0.18.0a210225
dask-cudf	0.18.0a0+254.g1544474166
datashader	0.11.1
datashape	0.5.4
decorator	4.4.2
defusedxml	0.6.0
distributed	2021.2.0
entrypoints	0.3
fastavro	1.3.2
fastrlock	0.5
Fiona	1.8.18
fsspec	0.8.7
GDAL	3.1.4
geopandas	0.8.1
HeapDict	1.0.1
idna	2.10
importlib-metadata	3.7.0
ipykernel	5.5.0
ipython	7.20.0
ipython-genutils	0.2.0
ipywidgets	7.6.3
jedi	0.18.0
Jinja2	2.11.3
joblib	1.0.1
jsonschema	3.2.0

Package	Version
jupyter-client	6.1.11
jupyter-core	4.7.1
jupyter-server-proxy	1.6.0
jupyterlab-pygments	0.1.2
jupyterlab-widgets	1.0.0
kiwisolver	1.3.1
llvmlite	0.35.0
locket	0.2.0
Markdown	3.3.3
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
msgpack	1.0.2
multidict	5.1.0
multipledispatch	0.6.0
munch	2.5.0
nbclient	0.5.2
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.4.3
networkx	2.5
notebook	6.2.0
numba	0.52.0
numpy	1.19.5
nvtx	0.2.3
olefile	0.46
packaging	20.9
pandas	1.1.5
pandocfilters	1.4.2
panel	0.10.3
param	1.10.1
parso	0.8.1
partd	1.1.0
pexpect	4.8.0
pickle5	0.0.11
pickleshare	0.7.5
Pillow	8.1.0
pip	21.0.1
prometheus-client	0.9.0

Package	Version
prompt-toolkit	3.0.16
protobuf	3.15.2
psutil	5.8.0
ptyprocess	0.7.0
py4j	0.10.9.1
pyarrow	1.0.1
pycparser	2.20
pyct	0.4.6
pydeck	0.5.0
pyee	7.0.4
Pygments	2.8.0
pynvml	8.0.4
pyOpenSSL	20.0.1
pyparsing	2.4.7
pyppeteer	0.2.2
pyproj	2.6.1.post1
PyQt5	5.12.3
PyQt5-sip	4.19.18
PyQtChart	5.12
PyQtWebEngine	5.12.1
pyrsistent	0.17.3
PySocks	1.7.1
python-dateutil	2.8.1
pytz	2021.1
pyviz-comms	2.0.1
PyYAML	5.4.1
pyzmq	22.0.3
requests	2.25.1
rmm	0.18.0
Rtree	0.9.7
scikit-learn	0.24.1
scipy	1.6.0
Send2Trash	1.5.0
setuptools	49.6.0.post20210108
Shapely	1.7.1
simpervisor	0.4
six	1.15.0
sortedcontainers	2.3.0
streamz	0.6.2

Package	Version
tblib	1.6.0
terminado	0.9.2
testpath	0.4.4
threadpoolctl	2.1.0
toolz	0.11.1
tornado	6.1
tqdm	4.57.0
traitlets	5.0.5
treelite	1.0.0
treelite-runtime	1.0.0
typing-extensions	3.7.4.3
ucx-py	0.18.0a0+19.ge17a4dc
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1
websockets	8.1
wheel	0.36.2
widgetsnbextension	3.5.1
xarray	0.16.2
xgboost	1.3.3
yarl	1.6.3
zict	2.0.0
zipp	3.4.0

RAPIDS Runtime PIP Python 3.8.6 Libraries for Workbench

This section lists the RAPIDS Runtime Python 3.8 libraries for Workbench that ship with ML Runtimes 2021.04.

Table 362: RAPIDS Runtime Python 3.7.8 Libraries for Workbench - Standard

Package	Version
aiohttp	3.7.3
appdirs	1.4.4
argon2-cffi	20.1.0
async-generator	1.10
async-timeout	3.0.1
attrs	20.3.0
backcall	0.2.0
backports.functools-lru-cache	1.6.1
bleach	3.3.0

Package	Version
bokeh	2.2.3
brotlipy	0.7.0
cdsw	1.0.0
certifi	2020.12.5
cffi	1.14.5
chardet	4.0.0
click	7.1.2
click-plugins	1.1.1
cligj	0.7.1
cloudpickle	1.6.0
colorcet	2.0.6
confluent-kafka	1.5.0
cryptography	3.4.4
cudf	0.18.0a0+254.g1544474166
cudf-kafka	0.18.0a0+254.g1544474166
cugraph	0.18.0a0+301.g0cc951f7.dirty
cuml	0.18.0a0+134.g7ca2db287
cupy	8.0.0
cusignal	0.18.0a0+33.gbc8598c
cuspatial	0.18.0a0+24.g1a2fc18
custreamz	0.18.0a0+254.g1544474166
cycler	0.10.0
cytoolz	0.11.0
dask	2021.2.0
dask-cuda	0.18.0a210225
dask-cudf	0.18.0a0+254.g1544474166
datashader	0.11.1
datashape	0.5.4
decorator	4.4.2
defusedxml	0.6.0
distributed	2021.2.0
entrypoints	0.3
fastavro	1.3.2
fastrlock	0.5
Fiona	1.8.18
fsspec	0.8.7
GDAL	3.1.4
geopandas	0.8.1
HeapDict	1.0.1

Package	Version
hupper	1.10.2
idna	2.10
importlib-metadata	3.7.0
ipykernel	5.5.0
ipython	7.20.0
ipython-genutils	0.2.0
ipywidgets	7.6.3
jedi	0.18.0
Jinja2	2.11.3
joblib	1.0.1
jsonschema	3.2.0
jupyter-client	6.1.11
jupyter-core	4.7.1
jupyter-server-proxy	1.6.0
jupyterlab-pygments	0.1.2
jupyterlab-widgets	1.0.0
kiwisolver	1.3.1
llvmlite	0.35.0
locket	0.2.0
Markdown	3.3.3
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
msgpack	1.0.2
multidict	5.1.0
multipledispatch	0.6.0
munch	2.5.0
nbclient	0.5.2
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.4.3
networkx	2.5
notebook	6.2.0
numba	0.52.0
numpy	1.19.5
nvtx	0.2.3
olefile	0.46
packaging	20.9
pandas	1.1.5

Package	Version
pandocfilters	1.4.2
panel	0.10.3
param	1.10.1
parso	0.8.1
partd	1.1.0
PasteDeploy	2.1.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
pip	21.0.1
plaster	1.0
plaster-pastedeploy	0.7
prometheus-client	0.9.0
prompt-toolkit	3.0.16
protobuf	3.15.2
psutil	5.8.0
ptyprocess	0.7.0
py4j	0.10.9.1
pyarrow	1.0.1
pycparser	2.20
pyct	0.4.6
pydeck	0.5.0
pyee	7.0.4
Pygments	2.8.0
pynvml	8.0.4
pyOpenSSL	20.0.1
pyparsing	2.4.7
pyppeteer	0.2.2
pyproj	2.6.1.post1
PyQt5	5.12.3
PyQt5-sip	4.19.18
PyQtChart	5.12
PyQtWebEngine	5.12.1
pyramid	2.0
pyrsistent	0.17.3
PySocks	1.7.1
python-dateutil	2.8.1
pytz	2021.1
pyviz-comms	2.0.1

Package	Version
PyYAML	5.4.1
pyzmq	22.0.3
rapids	0.0.1
requests	2.25.1
rmm	0.18.0
Rtree	0.9.7
scikit-learn	0.24.1
scipy	1.6.0
Send2Trash	1.5.0
setuptools	49.6.0.post20210108
Shapely	1.7.1
simpervisor	0.4
six	1.15.0
sortedcontainers	2.3.0
streamz	0.6.2
tblib	1.6.0
terminado	0.9.2
testpath	0.4.4
threadpoolctl	2.1.0
toolz	0.11.1
tornado	6.1
tqdm	4.57.0
traitlets	5.0.5
translationstring	1.4
treelite	1.0.0
treelite-runtime	1.0.0
typing-extensions	3.7.4.3
ucx-py	0.18.0a0+19.ge17a4dc
urllib3	1.26.3
venusian	3.0.0
wcwidth	0.2.5
webencodings	0.5.1
WebOb	1.8.7
websockets	8.1
wheel	0.36.2
widgetsnbextension	3.5.1
xarray	0.16.2
xgboost	1.3.3
yarl	1.6.3

Package	Version
zict	2.0.0
zipp	3.4.0
zope.deprecation	4.4.0
zope.interface	5.4.0

RAPIDS Runtime PIP Python 3.7.8 Libraries for JupyterLab

This section lists the RAPIDS Runtime Python 3.7.8 libraries for JupyterLab that ship with ML Runtimes 2021.04.

Table 363: RAPIDS Runtime Python 3.7.8 Libraries for JupyterLab - Standard

Package	Version
aiohttp	3.7.3
anyio	2.0.2
appdirs	1.4.4
argon2-cffi	20.1.0
async-generator	1.10
async-timeout	3.0.1
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
backports.functools-lru-cache	1.6.1
bleach	3.3.0
bokeh	2.2.3
brotlipy	0.7.0
cdsw	1.0.0
certifi	2020.12.5
cffi	1.14.5
chardet	4.0.0
click	7.1.2
click-plugins	1.1.1
cligj	0.7.1
cloudpickle	1.6.0
colorcet	2.0.6
confluent-kafka	1.5.0
cryptography	3.4.4
cudf	0.18.0a0+254.g1544474166
cudf-kafka	0.18.0a0+254.g1544474166
cugraph	0.18.0a0+301.g0cc951f7.dirty
cuml	0.18.0a0+134.g7ca2db287

Package	Version
cupy	8.0.0
cusignal	0.18.0a0+33.gbc8598c
cuspatial	0.18.0a0+24.g1a2fc18
custreamz	0.18.0a0+254.g1544474166
cuxfilter	0.18.0a0+68.g236aa39
cycler	0.10.0
cytoolz	0.11.0
dask	2021.2.0
dask-cuda	0.18.0a210225
dask-cudf	0.18.0a0+254.g1544474166
datashader	0.11.1
datashape	0.5.4
decorator	4.4.2
defusedxml	0.6.0
distributed	2021.2.0
entrypoints	0.3
fastavro	1.3.2
fastrlock	0.5
Fiona	1.8.18
fsspec	0.8.7
GDAL	3.1.4
geopandas	0.8.1
HeapDict	1.0.1
idna	2.10
importlib-metadata	3.7.0
ipykernel	5.5.0
ipython	7.20.0
ipython-genutils	0.2.0
ipywidgets	7.6.3
jedi	0.18.0
Jinja2	2.11.3
joblib	1.0.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.11
jupyter-core	4.7.1
jupyter-server	1.2.2
jupyter-server-proxy	1.6.0
jupyterlab	3.0.6

Package	Version
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
jupyterlab-widgets	1.0.0
kiwisolver	1.3.1
llvmlite	0.35.0
locket	0.2.0
Markdown	3.3.3
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
msgpack	1.0.2
multidict	5.1.0
multipledispatch	0.6.0
munch	2.5.0
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.4.3
networkx	2.5
notebook	6.2.0
numba	0.52.0
numpy	1.19.5
nvtx	0.2.3
olefile	0.46
packaging	20.9
pandas	1.1.5
pandocfilters	1.4.2
panel	0.10.3
param	1.10.1
parso	0.8.1
partd	1.1.0
pexpect	4.8.0
pickle5	0.0.11
pickleshare	0.7.5
Pillow	8.1.0
pip	21.0.1
prometheus-client	0.9.0
prompt-toolkit	3.0.16

Package	Version
protobuf	3.15.2
psutil	5.8.0
ptyprocess	0.7.0
py4j	0.10.9.1
pyarrow	1.0.1
pycparser	2.20
pyct	0.4.6
pydeck	0.5.0
pyee	7.0.4
Pygments	2.8.0
pynvml	8.0.4
pyOpenSSL	20.0.1
pyparsing	2.4.7
pyppeteer	0.2.2
pyproj	2.6.1.post1
PyQt5	5.12.3
PyQt5-sip	4.19.18
PyQtChart	5.12
PyQtWebEngine	5.12.1
pyrsistent	0.17.3
PySocks	1.7.1
python-dateutil	2.8.1
pytz	2021.1
pyviz-comms	2.0.1
PyYAML	5.4.1
pyzmq	22.0.3
requests	2.25.1
rmm	0.18.0
Rtree	0.9.7
scikit-learn	0.24.1
scipy	1.6.0
Send2Trash	1.5.0
setuptools	49.6.0.post20210108
Shapely	1.7.1
simpervisor	0.4
six	1.15.0
sniffio	1.2.0
sortedcontainers	2.3.0
streamz	0.6.2

Package	Version
tblib	1.6.0
terminado	0.9.2
testpath	0.4.4
threadpoolctl	2.1.0
toolz	0.11.1
tornado	6.1
tqdm	4.57.0
traitlets	5.0.5
treelite	1.0.0
treelite-runtime	1.0.0
typing-extensions	3.7.4.3
ucx-py	0.18.0a0+19.ge17a4dc
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1
websockets	8.1
wheel	0.36.2
widgetsnbextension	3.5.1
xarray	0.16.2
xgboost	1.3.3
yarl	1.6.3
zict	2.0.0
zipp	3.4.0

RAPIDS Runtime PIP Python 3.8.6 Libraries for JupyterLab

This section lists the RAPIDS Runtime Python 3.8.6 libraries for JupyterLab that ship with ML Runtimes 2021.04.

Table 364: RAPIDS Runtime Python 3.86 Libraries for JupyterLab - Standard

Package	Version
aiohttp	3.7.3
anyio	2.0.2
appdirs	1.4.4
argon2-cffi	20.1.0
async-generator	1.10
async-timeout	3.0.1
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0

Package	Version
backports.functools-lru-cache	1.6.1
bleach	3.3.0
bokeh	2.2.3
brotlipy	0.7.0
cdsw	1.0.0
certifi	2020.12.5
cffi	1.14.5
chardet	4.0.0
click	7.1.2
click-plugins	1.1.1
cligj	0.7.1
cloudpickle	1.6.0
colorcet	2.0.6
confluent-kafka	1.5.0
cryptography	3.4.4
cudf	0.18.0a0+254.g1544474166
cudf-kafka	0.18.0a0+254.g1544474166
cugraph	0.18.0a0+301.g0cc951f7.dirty
cuml	0.18.0a0+134.g7ca2db287
cupy	8.0.0
cusignal	0.18.0a0+33.gbc8598c
cuspatial	0.18.0a0+24.g1a2fc18
custreamz	0.18.0a0+254.g1544474166
cycler	0.10.0
cytoolz	0.11.0
dask	2021.2.0
dask-cuda	0.18.0a210225
dask-cudf	0.18.0a0+254.g1544474166
datashader	0.11.1
datashape	0.5.4
decorator	4.4.2
defusedxml	0.6.0
distributed	2021.2.0
entrypoints	0.3
fastavro	1.3.2
fastrlock	0.5
Fiona	1.8.18
fsspec	0.8.7
GDAL	3.1.4

Package	Version
geopandas	0.8.1
HeapDict	1.0.1
hupper	1.10.2
idna	2.10
importlib-metadata	3.7.0
ipykernel	5.5.0
ipython	7.20.0
ipython-genutils	0.2.0
ipywidgets	7.6.3
jedi	0.18.0
Jinja2	2.11.3
joblib	1.0.1
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.11
jupyter-core	4.7.1
jupyter-server	1.2.2
jupyter-server-proxy	1.6.0
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
jupyterlab-widgets	1.0.0
kiwisolver	1.3.1
llvmlite	0.35.0
locket	0.2.0
Markdown	3.3.3
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
msgpack	1.0.2
multidict	5.1.0
multipledispatch	0.6.0
munch	2.5.0
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.4.3
networkx	2.5

Package	Version
notebook	6.2.0
numba	0.52.0
numpy	1.19.5
nvtx	0.2.3
olefile	0.46
packaging	20.9
pandas	1.1.5
pandocfilters	1.4.2
panel	0.10.3
param	1.10.1
parso	0.8.1
partd	1.1.0
PasteDeploy	2.1.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
pip	21.0.1
plaster	1.0
plaster-pastedeploy	0.7
prometheus-client	0.9.0
prompt-toolkit	3.0.16
protobuf	3.15.2
psutil	5.8.0
ptyprocess	0.7.0
py4j	0.10.9.1
pyarrow	1.0.1
pycparser	2.20
pyct	0.4.6
pydeck	0.5.0
pyee	7.0.4
Pygments	2.8.0
pynvml	8.0.4
pyOpenSSL	20.0.1
pyparsing	2.4.7
pyppeteer	0.2.2
pyproj	2.6.1.post1
PyQt5	5.12.3
PyQt5-sip	4.19.18
PyQtChart	5.12

Package	Version
PyQtWebEngine	5.12.1
pyramid	2.0
pyrsistent	0.17.3
PySocks	1.7.1
python-dateutil	2.8.1
pytz	2021.1
pyviz-comms	2.0.1
PyYAML	5.4.1
pyzmq	22.0.3
rapids	0.0.1
requests	2.25.1
rmm	0.18.0
Rtree	0.9.7
scikit-learn	0.24.1
scipy	1.6.0
Send2Trash	1.5.0
setuptools	49.6.0.post20210108
Shapely	1.7.1
simpervisor	0.4
six	1.15.0
sniffio	1.2.0
sortedcontainers	2.3.0
streamz	0.6.2
tblib	1.6.0
terminado	0.9.2
testpath	0.4.4
threadpoolctl	2.1.0
toolz	0.11.1
tornado	6.1
tqdm	4.57.0
traitlets	5.0.5
translationstring	1.4
treelite	1.0.0
treelite-runtime	1.0.0
typing-extensions	3.7.4.3
ucx-py	0.18.0a0+19.ge17a4dc
urllib3	1.26.3
venusian	3.0.0
wcwidth	0.2.5

Package	Version
webencodings	0.5.1
WebOb	1.8.7
websockets	8.1
wheel	0.36.2
widgetsnbextension	3.5.1
xarray	0.16.2
xgboost	1.3.3
yarl	1.6.3
zict	2.0.0
zipp	3.4.0
zope.deprecation	4.4.0
zope.interface	5.4.0

ML Runtimes 2021.02

This section lists the Python and R libraries that ship with ML Runtimes 2021.02.

Note: Nvidia GPU Edition comes with CUDA 11.1 preinstalled.

Python 3.8 Libraries for Workbench

This section lists the Python 3.8 libraries for Workbench that ship with ML Runtimes 2021.02.

Table 365: Python 3.8.6 Libraries for Workbench - Standard

Library	Version
backcall	0.2.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1

Library	Version
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
py4j	0.10.9.1
Pygments	2.7.2
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Table 366: Python 3.8.6 Libraries for Workbench - Nvidia GPU Edition

Library	Version
backcall	0.2.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
Pillow	8.1.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
py4j	0.10.9.1
Pygments	2.7.2
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Python 3.7 Libraries for Workbench

This section lists the Python 3.7 libraries for Workbench that ship with ML Runtimes 2021.02.

Table 367: Python 3.7.9 Libraries for Workbench - Standard

Library	Version
backcall	0.2.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0

Library	Version
pickleshare	0.7.5
Pillow	8.1.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
py4j	0.10.9.1
Pygments	2.7.2
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Table 368: Python 3.7.9 Libraries for Workbench - Nvidia GPU Edition

Library	Version
backcall	0.2.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0

Library	Version
py4j	0.10.9.1
Pygments	2.7.2
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Python 3.6 Libraries for Workbench

This section lists the Python 3.6 Scala libraries for Workbench that ship with ML Runtimes 2021.02.

Table 369: Python 3.6.12 Libraries for Workbench - Standard

Library	Version
backcall	0.2.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0

Library	Version
py4j	0.10.9.1
Pygments	2.7.2
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Table 370: Python 3.6.12 Libraries for Workbench - Nvidia GPU Edition

Library	Version
backcall	0.2.0
bitarray	2.1.0
certifi	2020.12.5
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
jupyter-client	6.1.7
jupyter-core	4.6.3
kiwisolver	1.3.1
matplotlib	3.3.2
numpy	1.19.4
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1

Library	Version
Pygments	2.8.1
pyparsing	2.4.7
python-dateutil	2.8.1
pyzmq	19.0.2
requests	2.25.1
six	1.15.0
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5

Python 3.8 Libraries for JupyterLab

This section lists the Python 3.8 libraries for JupyterLab that ship with ML Runtimes 2021.02.

Table 371: Python 3.8.6 Libraries for JupyterLab - Standard

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2

Library	Version
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1

Library	Version
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1

Table 372: Python 3.8.6 Libraries for JupyterLab - Nvidia GPU Edition

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3

Library	Version
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
py4j	0.10.9.1
pycparser	2.20
Pygments	2.7.2
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
tornado	6.1

Library	Version
traitlets	4.3.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1

Python 3.7 Libraries for JupyterLab

This section lists the Python 3.7 libraries for JupyterLab that ship with ML Runtimes 2021.02.

Table 373: Python 3.7.9 Libraries for JupyterLab - Standard

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
importlib-metadata	3.4.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2

Library	Version
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
py4j	0.10.9.1
pycparser	2.20
Pygments	2.7.2
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.3

Library	Version
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.4.0

Table 374: Python 3.7.9 Libraries for JupyterLab - Nvidia GPU Edition

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
cycler	0.10.0
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
importlib-metadata	3.4.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4

Library	Version
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
py4j	0.10.9.1
pycparser	2.20
Pygments	2.7.2
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
tornado	6.1
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.4.0

Python 3.6 Libraries for JupyterLab

This section lists the Python 3.6 libraries for JupyterLab that ship with ML Runtimes 2021.02.

Table 375: Python 3.6.12 Libraries for JupyterLab - Standard

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bitarray	2.1.0
bleach	3.3.0
certifi	2020.12.8
cffi	1.14.4
chardet	4.0.0
contextvars	2.4
cycler	0.10.0
dataclasses	0.8
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
immutables	0.14
importlib-metadata	3.4.0
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3

Library	Version
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.2.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
pure-sasl	0.6.2
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1
traitlets	4.3.3

Library	Version
typing-extensions	3.7.4.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.4.0

Table 376: Python 3.6.12 Libraries for JupyterLab - Nvidia GPU Edition

Library	Version
anyio	2.0.2
argon2-cffi	20.1.0
async-generator	1.10
attrs	20.3.0
Babel	2.9.0
backcall	0.2.0
bleach	3.3.0
certifi	2020.11.8
cffi	1.14.4
chardet	4.0.0
contextvars	2.4
cycler	0.10.0
dataclasses	0.8
decorator	4.4.2
defusedxml	0.6.0
entrypoints	0.3
idna	2.10
immutables	0.14
importlib-metadata	3.4.0
impyla	0.17.0
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
jedi	0.17.2
Jinja2	2.11.3
json5	0.9.5
jsonschema	3.2.0
jupyter-client	6.1.7
jupyter-core	4.6.3
jupyter-server	1.2.2
jupyterlab	3.0.6

Library	Version
jupyterlab-pygments	0.1.2
jupyterlab-server	2.1.3
kiwisolver	1.3.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mistune	0.8.4
nbclassic	0.2.6
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.1.2
nest-asyncio	1.5.1
notebook	6.2.0
numpy	1.19.4
packaging	20.9
pandocfilters	1.4.3
parso	0.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.1.0
prometheus-client	0.9.0
prompt-toolkit	3.0.8
ptyprocess	0.6.0
py4j	0.10.9.1
pycparser	2.20
Pygments	2.8.1
pyparsing	2.4.7
pyrsistent	0.17.3
python-dateutil	2.8.1
pytz	2021.1
pyzmq	19.0.2
requests	2.25.1
Send2Trash	1.5.0
six	1.15.0
sniffio	1.2.0
terminado	0.9.2
testpath	0.4.4
thrift	0.11.0
thrift-sasl	0.4.3
tornado	6.1

Library	Version
traitlets	4.3.3
typing-extensions	3.7.4.3
urllib3	1.26.3
wcwidth	0.2.5
webencodings	0.5.1
zipp	3.4.0

R 4.0 Libraries

This section lists the R 4.0 libraries that ship with ML Runtimes 2021.02.

Table 377: R 4.0 Libraries for Workbench

Library	Version
askpass	1.1
assertthat	0.2.1
base	4.0.3
bitops	1.0-6
boot	1.3-25
brew	1.0-6
Cairo	1.2-0
caTools	1.18.0
cdsw	2.1
class	7.3-17
cli	2.0.1
cluster	2.1.0
codetools	0.2-16
colorspace	1.4-1
compiler	4.0.3
crayon	1.3.4
curl	4.3
datasets	4.0.3
digest	0.6.23
ellipsis	0.3.0
fansi	0.4.1
farver	2.0.3
foreigns	0.8-80
ggplot2	3.2.1
glue	1.3.1
graphics	4.0.3

Library	Version
grDevices	4.0.3
grid	4.0.3
gtable	0.3.0
httr	1.4.1
jsonlite	1.6
KernSmooth	2.23-17
labeling	0.3
lattice	0.20-41
lazyeval	0.2.2
lifecycle	0.1.0
magrittr	1.5
MASS	7.3-53
Matrix	1.2-18
methods	4.0.3
mgcv	1.8-33
mime	0.8
munsell	0.5.0
nlme	3.1-149
nnet	7.3-14
openssl	1.4.1
parallel	4.0.3
pillar	1.4.3
pkgconfig	2.0.3
plyr	1.8.5
png	0.1-7
R6	2.4.1
RColorBrewer	1.1-2
Rcpp	1.0.3
reshape2	1.4.3
RJSONIO	1.3-1.4
rlang	0.4.4
Rook	1.1-1
rpart	4.1-15
Rserve	1.7-0
scales	1.1.0
spatial	7.3-12
splines	4.0.3
stats	4.0.3
stats4	4.0.3

Library	Version
stringi	1.4.5
stringr	1.4.0
survival	3.2-7
sys	3.3
tcltk	4.0.3
tibble	2.1.3
tools	4.0.3
utf8	1.1.4
utils	4.0.3
vctrs	0.2.2
viridisLite	0.3.0
withr	2.1.2

R 3.6 Libraries

This section lists the R 3.6 libraries that ship with ML Runtimes 2021.02.

Table 378: R 3.6 Libraries for Workbench

Library	Version
askpass	1.1
assertthat	0.2.1
base	3.6.3
bitops	1.0-6
boot	1.3-24
brew	1.0-6
Cairo	1.2-0
caTools	1.18.0
cdsw	2.1
class	7.3-15
cli	2.0.1
cluster	2.1.0
codetools	0.2-16
colorspace	1.4-1
compiler	3.6.3
crayon	1.3.4
curl	4.3
datasets	3.6.3
digest	0.6.23
ellipsis	0.3.0

Library	Version
fansi	0.4.1
farver	2.0.3
foreigns	0.8-75
ggplot2	3.2.1
glue	1.3.1
graphics	3.6.3
grDevices	3.6.3
grid	3.6.3
gtable	0.3.0
httr	1.4.1
jsonlite	1.6
KernSmooth	2.23-16
labeling	0.3
lattice	0.20-38
lazyeval	0.2.2
lifecycle	0.1.0
magrittr	1.5
MASS	7.3-51.5
Matrix	1.2-18
methods	3.6.3
mgcv	1.8-31
mime	0.8
munsell	0.5.0
nlme	3.4-144
nnet	7.3-12
openssl	1.4.1
parallel	3.6.3
pillar	1.4.3
pkgconfig	2.0.3
plyr	1.8.5
png	0.1-7
R6	2.4.1
RColorBrewer	1.1-2
Rcpp	1.0.3
reshape2	1.4.3
RJSONIO	1.3-1.4
rlang	0.4.4
Rook	1.1-1
rpart	4.1-15

Library	Version
Rserve	1.7-0
scales	1.1.0
spatial	7.3-11
splines	3.6.3
stats	3.6.3
stats4	3.6.3
stringi	1.4.5
stringr	1.4.0
survival	3.1-8
sys	3.3
tcltk	3.6.3
tibble	2.1.3
tools	3.6.3
utf8	1.1.4
utils	3.6.3
vctrs	0.2.2
viridisLite	0.3.0
withr	2.1.2

ML Runtimes 2020.11

This section lists the Python and R libraries that ship with ML Runtimes 2020.11.

All ML Runtimes 2020.11 contain NumPy 1.19.2 and Matpolotlib 3.1.2.

- Matplotlib - 3.1.2
- NumPy - 1.19.2 and the Python versions, which are 3.6.10, 3.7.9, and 3.8.3

Cloudera AI

ML Runtimes Release Notes

Date published: 2020-07-16
Date modified: 2025-01-21

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ML Runtimes What's New **4**
What's new in ML Runtimes version 2025.01.1 4
What's New in ML Runtimes version 2024.10.1 6
What's New in ML Runtimes version 2024.05.2 7
What's New in ML Runtimes older releases 8
What's New in ML Runtimes version 2024.05.1 8
What's New in ML Runtimes version 2024.02.1 8
ML Runtimes Version 2023.12.1 9
ML Runtimes Version 2023.08.2 9
ML Runtimes Version 2023.08.1 10
ML Runtimes Version 2023.05.2 10
ML Runtimes Version 2023.05.1 10
ML Runtimes Version 2022.11.2 11
ML Runtimes Version 2022.11 12
ML Runtimes Version 2022.04 12
ML Runtimes Version 2021.12 13
ML Runtimes Version 2021.09.02 13
ML Runtimes Version 2021.09 14
ML Runtimes Version 2021.06 14
ML Runtimes Version 2021.04 14
ML Runtimes Version 2021.02 15
ML Runtimes Version 2020.11 15

ML Runtimes Known Issues and Limitations **16**
Known Issues and Limitations in ML Runtimes version 2025.01.01 16
Known Issues and Limitations in ML Runtimes version 2024.10.01 19
Known Issues and Limitations in ML Runtimes version 2024.05.02 21
Known Issues and Limitations in ML Runtimes older releases 24

ML Runtimes What's New

This section lists major features and updates for Machine Learning (ML) Runtimes.

Note:

Cloudera recommends upgrading to the latest ML Runtimes available, as the latest version contains possible fixes that are not available in earlier versions. For finding the latest version of ML Runtimes, see section *ML Runtimes Pre-installed Packages*. You can also check the latest version of ML Runtimes in the Archive Runtimes list.

Related Information

Archive Runtimes
Adding new ML Runtimes
ML Runtimes Nvidia GPU Edition
Upgrading R and Python Packages
ML Runtimes Environment Variables

What's new in ML Runtimes version 2025.01.1

Review the major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2025.01.1.

Note: New ML Runtimes releases are automatically added to the deployment, if Internet connection is available.

New Features

- ML Runtimes' base image has been updated from Ubuntu 20.04 to Ubuntu 24.04. This update enables many new features, a better experience for end users, faster and better fixes of CVEs and enhanced security to prevent malicious activities.

  Note:

  Due to the extensive changes with SSL between Ubuntu 20.04 and Ubuntu 24.04, we recommend testing your workflows if they contain any https/tls/ssl/... operations.
- The new ML Runtimes are not drop-in replacements of the previously released ML Runtimes. If you install any additional software into Cloudera AI projects, it might result in issues and might require reinstallation or recompilation.
- CUDA images are based on nvidia/cuda:12.5.1-devel-ubuntu24.04. The CUDA version on the Nvidia GPU Runtimes has also been also updated.
- All Runtime images are now PBJ-based (Powered by Jupyter); Workbench-based Runtime images are no longer released.

  Note: You can no longer use any of the new ML Runtimes going forward with Cloudera Data Science Workbench. (CDSW support ended in May 2024)

Behavioral changes

- **Python** changes
  - Workbench-based Python images have been removed, only PBJ Workbench Python images and JupyterLab Runtime images will be released going forward.
  - Python 3.7- and 3.8-based images are no longer supported.

    Note:

    Note, that only Python 3.7 Runtime is compatible with Spark 2. If you want to use Python with Spark 2, you need to install older ML Runtimes to your Cloudera AI Workbenches.
  - All ML Runtime images have updated PIP (Python package manager) package versions.
- **JupyterLab** changes
  - JupyterLab has been migrated to PBJ (Powered by Jupyter).
  - node.js has been removed.
- **R** changes
  - Non-PBJ Workbench Runtime images are no longer available.
  - Python has been updated to Python version 3.12.
- **Scala** changes
  - Scala has been migrated to PBJ (Powered by Jupyter).
  - Python has been updated to Python 3.12.
  - Scala image has been updated to support Spark 3.x.
  - Spark version 2.x is no longer supported (starting a Spark 2.x session will fail with an error).
  - Scala has been updated to version 2.12. The Scala update might bring backward compatibility issues.
  - When building a custom PBJ Scala Runtime image either use the Cloudera PBJ Scala Runtime or start from the Cloudera-provided public Dockerfiles. The custom PBJ Scala Runtime image provided by Cloudera contains necessary updates, such as the Scala chunker on the Runtime image and other necessary enhancements.
- **Conda** changes
  - Conda has been migrated to PBJ (Powered by Jupyter).
  - Python has been updated to 3.12.
- To ensure the compatibility of AMPs with ML Runtimes 2025.01.1 using JupyterLab, you need to proceed with either of the following options:
  - Switch to PBJ Workbench in the AMPs' .project-metadata.yaml file.:

    ```
    runtimes:
      - editor: PBJ Workbench
    ```
  - Use jobs instead of sessions for automated tasks.

    Sessions are intended for interactive use, while AMP setup steps are executed in a programmed manner or automatically. Instead of running a session, change a session-based task into a job, create and execute a job.

    Instead of running a session:

    ```
    tasks:
      - type: run_session
        name: Install Dependencies
        script: scripts/install_dependencies.py
        memory: 2
        cpu: 1
    ```

    Use the following example entry in the .project-metadata.yaml file for creating and executing a job:

    ```
    tasks:
    ```

```
  - type: create_job
    name: Install Dependencies
    entity_label: install_deps
    script: scripts/install_dependencies.py
    arguments: None
    cpu: 1
    memory: 2
    short_summary: Create job to install project dependencies.
    environment:
      TASK_TYPE: CREATE/RUN_JOB

  - type: run_job
    entity_label: install_deps
    short_summary: Running install dependencies job.
    long_summary: >-
      Running the job to install dependencies.
```

- The cdsw Python module has been replaced by the cml module.

Improvements

- Cloudera Copilot enhancements:
  - Support for default embedding models.
  - Support for additional Amazon Bedrock models: Claude v3.5 Haiku, Claude v3.5 Sonnet, Llama 3.1 405b Instruct, and Mistral Large 24.02.
  - Increase in maximum tokens limit to 2048 for Cloudera AI Inference model's responses.
- Python maintenance versions have been upgraded as follows:
  - 3.9.19 -> 3.9.21
  - 3.10.14 -> 3.10.16
  - 3.11.9 -> 3.11.11
  - 3.12.4 -> 3.12.8

Fixed issues

- R workloads' .Libpaths() changed

  Earlier the order of R workloads' .Libpaths() was:

  1. Cloudera AI project
  2. CMLAddon
  3. Runtime image

  Now R workloads that are running PBJ Runtimes have the .Libpaths() set up in this order:

  1. Cloudera AI project
  2. Runtime image
  3. CMLAddon

  Reinstall your R packages in your projects if the above detailed change of order in R workload's library path might cause any issues.

What's New in ML Runtimes version 2024.10.1

Review the major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2024.10.1.

Note: New ML Runtimes releases are automatically added to the deployment, if Internet connection is available.

New Feautre

Cloudera Copilot is generally available (GA): Cloudera Copilot is a highly configurable AI coding assistant integrated with the JupyterLab editor. The Copilot improves developer productivity by debugging code, answering questions and generating notebooks.

Cloudera Copilot is available in the Runtimes with JupyterLab Editor.

For more information, see Using Cloudera Copilot.

New Runtimes

- Python 3.12 PBJ Workbench
- Python 3.12 PBJ NVIDIA GPU Workbench
- Python 3.12 JupyterLab
- Python 3.12 JupyterLab NVIDIA GPU

Note: Python 3.12 Runtimes are compatible only with Spark version 4.0.

Behavioral Changes

- JupyterLab Real-Time Collaboration (RTC) plug-in is not installed by default.
- On Workbench and Jupyterlab Runtimes, the `await_workers` function in the CDSW R and Python libraries will not time out when `timeout_seconds` is set to 0. Instead, the command will block until the workers are ready. Earlier, if the `timeout_seconds` set to 0 the function call timed out immediately. The new implementation matches the documented behaviour of this function.

Improvements

- Python maintenance versions - Python maintenance versions have been upgraded to 3.7.17, 3.8.19, 3.9.19, 3.10.14, and 3.11.9.
- CVE fixes - This maintenance release includes numerous improvements related to Common Vulnerability and Exposures (CVE).
- Update setup tools in Conda runtime to resolve CVE-2024-6345.

What's New in ML Runtimes version 2024.05.2

Review the major features and updates for Machine Learning Runtimes.

This release is available with Machine Learning Runtimes version 2024.05.2.

Note: New Machine Learning Runtime releases are automatically added to the deployment, if internet connection is available.

Behavioral Changes

- JupyterLab Real-Time Collaboration (RTC) plug-in is not installed by default.

Improvements

- CVE fixes - This maintenance release includes numerous improvements related to Common Vulnerability and Exposures (CVE).

What's New in ML Runtimes older releases

Overview of new features, enhancements, and changed behavior introduced in earlier releases of Machine Learning Runtimes.

What's New in ML Runtimes version 2024.05.1

Review the major features and updates for Machine Learning Runtimes.

This release is available with Machine Learning Runtimes version 2024.05.1.

Note: New Machine Learning Runtime releases are automatically added to the deployment, if internet connection is available.

New features

- Cloudera Copilot is an AI-powered coding assistant designed for seamless integration within JupyterLab Machine Learning Runtimes. With its chat interface and comprehensive code completion features, Cloudera Copilot enhances the development experience for machine learning projects. It offers compatibility with both custom models deployed in Cloudera AI Inference Service and Amazon Bedrock models, providing developers with flexibility and efficiency in their workflows.
- JupyterLab is upgraded to 4.1.5 for Python 3.8 and for higher releases.
- Workbench R 4.4 Runtime is available.
- JupyterLab, PBJ Workbench Editor, and Nvidia GPU Runtime variants with Python 3.8 and higher versions are upgraded from CUDA 11.8 and cuDNN 8.7 versions to CUDA 12.3 and cuDNN 9.0 versions. It is recommmended to install TensorFlow with the [and-cuda] option, to install TensorFlow's NVIDIA CUDA library dependencies as well, as the current latest TensorFlow version (2.16.1) still has dependencies on cuDNN8.

  Due to the known TensorFlow issue https://github.com/tensorflow/tensorflow/issues/63362 for installing TensorFlow 2.16.1 it is also needed to set the LD_LIBRARY_PATH environment variable to /home/cdsw/.local/lib/python3.X/site-packages/nvidia/cudnn/lib/:$LD_LIBRARY_PATH where python3.X stands for the used Python version, for example python3.9.

Fixed issues

- Python was missing from Scala Runtime, which could lead to warnings or the Scala engine not starting at all. Python interpreter is now added to Scala Runtime.
- With the upgrade of the PY4J library version to 0.10.9.5, the SPARK-37004 bug in Spark workloads is fixed, making those workloads more robust against kernel interruptions.

Improvements

- This release includes numerous improvements related to Common Vulnerability and Exposures (CVE).

What's New in ML Runtimes version 2024.02.1

Review the major features and updates for Machine Learning Runtimes.

This release includes Machine Learning Runtimes version 2024.02.1.

Note: New Machine Learning Runtime releases are automatically added to the deployment, if internet connection is available.

New features

- Pandas is now included in the Python 3.8 and higher Runtimes.
- You can now see hidden files in the JupyterLab file browser.

Improvements

- Multiple small improvements and bug fixes
- Multiple CVE fixes
- Added the following libraries to improve vision and audio processing model support:
  - libgl-dev
  - libjpeg-dev
  - libpng-dev
  - ffmpeg

ML Runtimes Version 2023.12.1

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2023.12.1.

Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

New Runtimes

- Python 3.11 PBJ Standard Workbench
- Python 3.11 PBJ Nvidia GPU Workbench
- Python 3.11 JupyterLab Standard
- Python 3.11 JupyterLab Nvidia GPU

Note: Python 3.11 Runtimes are not compatible with Spark versions prior to Spark 3.4.

Improvements

- Python maintenance versions - Python maintenance versions have been upgraded to 3.8.18, 3.9.18, and 3.10.13.

Discontinued Runtimes

The following runtimes are no longer included.

- R 3.6 PBJ
- R 4.0 PBJ
- R 4.1 PBJ
- R 3.6 Workbench
- R 4.0 Workbench
- R 4.1 Workbench

Fixed Issue

Workbench editor for Python 3.8 and 3.9 now can parse multiline strings properly. (DSE-27222)

ML Runtimes Version 2023.08.2

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2023.08.2.

Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Improvements

- CVE fixes - This maintenance release includes numerous improvements related to Common Vulnerability and Exposures (CVE).

ML Runtimes Version 2023.08.1

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2023.08.1.

Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

New Features / Improvements

- New Runtimes - Python 3.10 Standard Workbench, R 4.3 Standard Workbench, and R 4.3 Standard PBJ Workbench are included in this release.

Fixed Issues

- Chunker (DSE-27222) - Fixed an issue where the chunker failed to correctly parse multiline string inputs in classic Workbench Runtimes.
- Jupyter (DSE-28134) - `jupyter-kernel-gateway` has been upgraded to version 2.5.2. As a result, users cannot install incompatible versions of `jupyter-client` breaking their Python environments.
- Security - Various security fixes were implemented.

ML Runtimes Version 2023.05.2

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2023.05.2.

Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Fixed Issues

- This maintenance release is to automatically update those ML Runtimes versions that were referenced with a wrong image identifier when ML Runtimes 2023.05.1 was released.

ML Runtimes Version 2023.05.1

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2023.05.1.

Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

New features

- This release of ML Runtimes includes the following features that improve support for Large Language (LLMs) within CML:
  - Git Large File Storage (LFS) support is now included in all ML Runtimes images. Users can now easily download Large Language models via Git.

    LFS replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise.
  - Upgraded CUDA to version 11.8.0 in JupyterLab Python CUDA runtimes. This change allows customers to use the latest pytorch version in CML. Affected Python versions are 3.7, 3.8, 3.9, and 3.10.
  - NVIDIA GPU versions of PBJ Runtimes are now available for Python 3.7, 3.8, 3.9 and 3.10. The base image used for CUDA ML Runtimes are the “devel” series CUDA image provided by NVIDIA. This change, for example, enables the usage of the Numba JIT compiler.

    This change also supports the latest version of tensorflow in CML and some other libraries that need to compile GPU code in run-time.
  - wget is now installed on all ML Runtimes. This enables users another way to download large language models in CML in addition to git-lfs.
- This release introduces a new Technical Preview Runtime, adding support for Anaconda, based on the JupyterLab editor. Conda environments can be created, modified, and activated from the Terminal without any limitations. For more information, see Using Conda Runtimes.
- New Python 3.10 Runtimes are available.
- Added the following extensions to our JupyterLab Runtimes (Python 3.8, 3.9, 3.10): Language Server Protocol (jupyterlab-lsp) and git (jupyterlab-git). Language Server Protocol integration provides coding assistance for JupyterLab (code navigation, hover suggestions, linters, autocompletion, rename).
- Added the git (jupyterlab-git) extension to our JupyterLab Runtimes (Python 3.7).
- The Workbench Runtimes is being deprecated in favor of PBJ. As a result, the CUDA version for the Workbench Runtimes is not updated for the current release.
- Implya client is now updated to version 0.18.0 which has support for authentication via JWT token.

Version 2023.05.1

Fixed Issues

- DSE-26724 wget is now installed on all ML Runtimes. This enables users to download large language models in CML two ways: via Gi and via wget.
- DSE-25078 The OS package cmake is now installed for all ML Runtime images. This change enables users to install packages like nloptr.
- DSE-23762 Updated installed R package versions to make sparklyr R package usable in the same session as it was installed in.
- DSE-23751 Upgraded Jupyter to version 3.6.0 in all JupyterLab Runtimes.
- DSE-23613 Upgraded CUDA to version 11.8.0 in JupyterLab Python CUDA runtimes. This change was needed to allow customers to use the latest pytorch version in CML. Affected Python versions are 3.7, 3.8, 3.9, and 3.10.
- DSE-22425 Added OS packages to R Runtimes to support recent versions of the "devtools" package.
- DSE-20234 Jupyter session-specific data is no longer persisted in the Project directory. This prevents errors when starting multiple Jupyter sessions.

ML Runtimes Version 2022.11.2

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2022.11.2.

Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Version 2022.11.2

Fixed Issues

- Multiple CVEs have been fixed.

ML Runtimes Version 2022.11

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2022.11

Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Version 2022.11

New features

- PBJ (Powered by Jupyter) Runtime GA: PBJ Runtimes were previously available as a Tech Preview only. In ML Runtimes 2022.11 the PBJ Runtimes are available to all users (GA). The following PBJ Runtimes are included:
  - Python 3.7
  - Python 3.8
  - Python 3.9
  - R 3.6
  - R 4.0
  - R 4.1

  These PBJ Runtimes are available on Cloudera Public Cloud (from 2022.10-1) and Cloudera Private Cloud (from 1.5.0) form factors, but are not available on CDSW.

  The new PBJ Runtimes appear on the Runtime Selector elements when users start new workloads. Their usage is the same as the non-PBJ Runtimes, apart from Models. Models require the usage of a new library called cml for Python and R, that was introduced in Cloudera Public Cloud (2022.10-1) and the upcoming Private Cloud (1.5.0) versions.

  Usage of cml with regard to models is detailed here: PBJ Runtimes and Models

  Examples can be found here: Example models with PBJ Runtimes.

Fixed Issues

- DSE-22433 Upgrade numpy version in Python runtimes to resolve issues.
- DSE-22374 Reduced log level to WARN log level can be manipulated through the HADOOP_ROOT_LOGGER environment variable. This is set to WARN,console in order to keep console logging, while reducing log level from INFO to WARN.
- DSE-22251 Add flextable dependencies to R Runtimes Added flextable dependencies (libfontconfig1-dev, libcairo2-dev) to R runtimes. Added two OS libraries so flextable can be installed and used in CML.

Known issues

- The CUDA version (11.4.1) being used to build Nvidia GPU Runtimes is not supported by newer Torch versions (1.13+).

  Workaround: Use Torch versions up to 1.12.1.

  Cloudera Bug: DSE-24038
- When installing sparklyr for R, the installation will succeed, but importing the library in the same session will fail. This happens, because earlier versions of dependent packages (vctrs, lifecycle, rlang, cli) are already loaded in the R session, therefore newer versions cannot be loaded.

  Workaround: Start a new session, import and use sparklyr there.

  Cloudera Bug: DSE-23762

ML Runtimes Version 2022.04

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2022.04.

Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Version 2022.04

New features

- *Powered By Jupyter (PBJ) Workbench* (Preview) - The Powered By Jupyter (PBJ) Workbench features the classic workbench UI backed by the open-source Jupyter protocol pre-packaged in an ML Runtime image. Users can easily choose this Runtime when launching a session. The open-source Jupyter infrastructure eliminates the dependency on proprietary CML code when building a Docker image, allowing runtime images to be built with open standards. The first PBJ Workbench Runtimes are available as Tech Preview.
- The Global Registry for npm is set to https://registry.npmjs.org/ in all of our runtimes. With this change, customers can use npm install in their Dockerfiles, when they build custom runtimes on top of ours.

Note: RAPIDS Runtimes are now an optional extension of the Cloudera ML Runtime distribution. The RAPIDS images will no longer be distributed automatically. Administrators can Register them in the Runtime Catalog.

The Vim text editor is no longer supported for Cloudera provided ML Runtimes.

Fixed issues

- DSE-2064 Upgrade of Python Kernels across of all ML Runtimes:
  - Python 3.7.12 -> 3.7.13
  - Python 3.8.12 -> 3.8.13
  - Python 3.9.7 -> Python 3.9.11

ML Runtimes Version 2021.12

Major features and updates for ML Runtimes.

This release is available with ML Runtimes version 2021.12.

Note: New ML Runtime releases are automatically added to the deployment, if internet connection is available.

Version 2021.12

New features

- New versions provided for all ML Runtimes.
- Added R 4.1 Runtimes
- RAPIDS base image was updated to use 21.12 RAPIDS release (DSE-19745)

Note: Python 3.6 ML Runtimes are not longer provided due to Python 3.6 end of life

Fixed issues

- DSE-17817 psycopg2 now can be installed for Models.
- DSE-17547 Fixed a matplotlib warning in RAPIDS Runtimes.
- DSE-10146 Fixed issue causing R workbench to crash due to an illegal character.
- DSE-17657 Upgrade jupyterlab and notebook for CVE-2021-32797 and CVE-2021-32798.
- Various security and library version updates
- DSE-19805 Fixed CVE-2022-22817 critical vulnerability in Python Runtimes

ML Runtimes Version 2021.09.02

Major features and updates for ML Runtimes.

Version 2021.09.02

Fixed issues

- DSE-19496 The log4j build dependency is removed for the Scala ML Runtime. Once this ML Runtimes version is deployed, the user interface will always point to this latest version of the 2021.09.02 Scala Runtime to run workloads using the Scala kernel (until a newer version is released). Users will not be able to select the originally released 2021.09 Scala Runtime.

ML Runtimes Version 2021.09

Major features and updates for ML Runtimes.

Version 2021.09

New features

- Python 3.9 - ML Runtimes now support Python 3.9 kernels.
- Scala - Scala kernel is supported for running Sessions and Jobs.
- CUDA 11.4 - Nvidia GPU Editions are using CUDA 11.4.

Fixed issues

- DSE-17126 Starting a worker from Runtime session will create a worker with engine image

  This issue is resolved in ML Runtimes 2021.09 when used with the latest ML Workspace versions.

  For workers to function properly with ML Runtimes, please use ML Runtimes 2021.09 or later with CML Workspace version of 2.0.22 or later.

ML Runtimes Version 2021.06

Major features and updates for ML Runtimes.

Version 2021.06

New features

- Impyla - Impyla is now preinstalled in standard Python Runtimes.

  Cloudera bug: DSE-14293
- Runtimes support for Models - Runtimes now support running models.

  Cloudera bug: DSE-14294
- JupyterLab shutdown_no_activity_timout configurable - JupyterLab shutdown_no_activity_timout is now configurable through environment variable.

  Cloudera bug: DSE-14817

Fixed issues

- DSE-15191 - Fixed installation issue with thrift-sasl python package.
- DSE-14886 - Fixed an issue preventing matplotlib installation into project.
- DSE-14737 - Fixed warnings appearing in the log due to using JupyterLab 3.
- DSE-15215 - Fixed an issue due to decorators that could not be used with Python 3.8 and Workbench editor.

ML Runtimes Version 2021.04

Major features and updates for ML Runtimes.

Version 2021.04

New features

- RAPIDS Runtimes - The RAPIDS Edition Runtimes are built on top of community built RAPIDS docker images. The RAPIDS suite of software libraries relies on NVIDIA CUDA primitives for low-level compute optimization, but exposes that GPU parallelism and high-bandwidth memory speed through user-friendly Python interfaces.

Fixed issues

The following fixed issus relate only to RAPIDS Runtimes.

- DSE-13743 - Idle JupyterLab sessions are now ended after around IDLE_MAXIMUM_MINUTES. (See *ML Runtimes Environment Variables Environment Variables*).

  Note: These sessions may run for an additional 5 minutes after IDLE_MAXIMUM_MINUTES. (DSE-13743)

- DSE-14979 - matplotlib figures restore the styling used in engine:13 (MLRuntimes 2020.11 used the matplotlib defaults)
- DSE-12881 - For Python runtimes, py4j is now installed.
- Security fixes, python library version updates

ML Runtimes Version 2021.02

Major features and updates for ML Runtimes.

Version 2021.02

New features

- Nvidia GPU support - Runtimes supporting Nvidia GPUs are available with preinstalled CUDA software for both Workbench and JupyterLab editors. (See *ML Runtimes Nvidia GPU Edition*.)
- Runtimes supporting R 3.6 and R 4.0 available for Workbench editor

  Note: User R libraries are installed in /home/cdsw/.local/lib/R rather than /home/cdsw/R, which is where they were installed in engine:13. This change was made to support simultaneous use of R 3.6 and R 4.0. Because of this change you will need to reinstall user R packages when migrating an existing project from engine:13 to MLRuntimes 2021.02, or from R 3.6 to R 4.0

  Cloudera bug: DSE-3686
- Support for JupyterLab runtimes - The JupyterLab runtimes now use JupyterLab 3.0 (was 2.2) and are considered generally available using Standard Edition ML Runtimes. See JupyterLab blog for notes on this upgrade.
- Python runtimes include a C++ compiler - Python runtimes now include a C++ compiler (g++), which enables the installation and use of certain Python libraries such as impyla and pystan.

  Cloudera bug: DSE-14492
- Preinstalled Python libraries (See *ML Runtimes 2021.02*) are now installed in /usr/local/lib rather than /var/lib/cdsw, which is where they were installed in engine:13 and runtimes 2020.11 (DSE-12177). This means that you can upgrade these packages more easily, but there are some packages you should not upgrade. (See *Upgrading R and Python Packages*)
- ML runtimes are supported on CDSW (version 1.9.0 or higher).

Fixed issues

- DSE-13743 - Idle JupyterLab sessions are now ended after around IDLE_MAXIMUM_MINUTES. (See *ML Runtimes Environment Variables Environment Variables*).

  Note: These sessions may run for an additional 5 minutes after IDLE_MAXIMUM_MINUTES. (DSE-13743)

- DSE-14979 - matplotlib figures restore the styling used in engine:13 (MLRuntimes 2020.11 used the matplotlib defaults)
- DSE-12881 - For Python runtimes, py4j is now installed.
- Security fixes, python library version updates

ML Runtimes Version 2020.11

Release notes and fixed issues

Version 2020.11

New features

- New Runtimes versions added

Note: Nvdia GPU Edition comes with CUDA 11.1 preinstalled.

Fixed issues

- DSE-13904 - Fixed an issue where CML workspace installation may take up to 10 minutes longer when the Autoscaling Group minimum is set to 0.
- DSE-13898 - Updated TGT image to fix an issue so that the freeIPA HA release retries on failure.
- CDPSDX-2673 - Added a Retry step to the login context to reduce the chance of PRE_AUTH failures.

ML Runtimes Known Issues and Limitations

This section lists the known issues you might run into while using ML Runtimes.

Known Issues and Limitations in ML Runtimes version 2025.01.01

You might run into some known issues while using ML Runtimes 2025.01.1.

DSE-35335 PBJ R Runtimes print some output to logs instead of printing it to the session UI

In PBJ R Runtime, messages generated in R (using IRkernel) are sent to the logs instead of being displayed in the UI. This behavior is due to the IRkernel, which sends the stderr output to the logs, rather than forwarding it to the UI. This behavior is not related to the PBJ R Runtime, but observed related to the IRkernel operation.

DSE-25143 Assembling plots in PBJ R Runtimes does not work

When trying to plot additional content on already existing plots, PBJ R Runtimes throw an error. Plots can only be created using the plot function.

DSE-41555 Cloudera Copilot UI displays error on startup intermittently

When embedded or language models are not configured, the following error message is displayed:

There seems to be a problem with the Chat backend, please look at the JupyterLab server logs or contact your administrator to correct this problem.

Workaround: Configure Cloudera Copilot with both language models and embedding models.

DSE-41570 Cloudera AI Inference service is temporarily unavailable with Cloudera Copilot

Due to a known issue, Cloudera AI Inference service is temporarily unavailable with Cloudera Copilot. When you try to use the Cloudera AI Inference service, an error is displayed.

DSE-32839 Extra configuration needed when using Spark in a PBJ Workbench-based R Runtime

When using Spark in R workloads that are running PBJ Workbench Runtimes, the environmental variable R_LIBS_USER must be passed to the Spark executors with the value set to /home/cdsw/.local/lib/R/<R_VERSION>/library.

For example, when using `sparklyr` with a PBJ Workbench R 4.3 Runtime, the correct way to set up a `sparklyr` connection is:

```
library(sparklyr)
     config <- spark_config()
```

```
     config$spark.executorEnv.R_LIBS_USER="/home/cdsw/.local/lib/R/4.3/libra
ry"
     sc <- spark_connect(config = config)
```

Packages can fail to load in a session

When installing R or Python packages in a session, the kernel might not be able to load the package in the same session, if a previous version of the package or its newly installed dependencies have been loaded in the same session. Such issues are observable more often in PBJ R Runtimes, which automatically load basic R packages like `vctrs`, `lifecycle`, `rlang`, `cli` at session startup.

Workaround: Start a new session, import and use the newly installed package there.

Version of jupyter-client Python package must be of lower version than version 8 for PBJ Runtimes

Upgrading the Python package jupyter-client with a version greater than 7.4.9 can temporarily break a project. Workloads using PBJ Runtimes will not be able to start Projects if the jupyter-client version is greater than 7.4.9.

Workaround: Launch the same version of Python, but not on a PBJ Runtime (either Workbench or JupyterLab). Open a Terminal window and uninstall the jupyter-client package from the Project by executing pip3 uninstall jupyter-client. Verify your change by running pip3 list and checking that the version of the jupyter-client package is less than version 8.

DSE-9818 JupyterLab Conda Technical Preview Runtime

Sessions

When starting a Notebook or a Console for a specific environment, the installed packages will be available and the interpreter used to evaluate the contents of the Notebook or Console will be the one installed in the environment. However, the Conda environment is not "activated" in these sessions, therefore commands like !which python will return with the base Python interpreter on the Runtime. The recommended ways to modify a Conda environments or install packages are the following:

- Conda commands must be used with the -n or --name argument to specify the environment, for example conda -n myenv install pandas
- When installing packages with pip, use the %pip magic to install packages in the active kernel's environment, for example %pip install pandas

Applications and Jobs

To start an Application or Job, first create a launcher Python script containing the following line: ! source activate <conda_env_name> && python <job / application script.py>

When starting the Application or Job, select the launcher script as the "Script".

Models

Models are currently not supported for the Conda Runtime.

Spark

Spark is not supported in JupyterLab Notebooks and Consoles.

Spark workloads are supported in activated Conda environments in JupyterLab Terminals, or in Jobs or Applications.

The CDSW libraries for Python and R are not available for the Conda Runtimes.

Adding a new ML Runtimes when using a custom root certificate might generate error messages

When trying to add new ML Runtimes, a number of error messages might appear in various places when using a custom root certificate. For example, you might see: the Could not fetch the image metadata or the certificate signed by unknown authority messages. This is caused by the runtime-puller pods not having access to the custom root certificate that is in use.

Workaround:

1. Create a directory at any location on the master node:

   For example:

   mkdir -p /certs/
2. Copy the full server certificate chain into this folder. It is usually easier to create a single file with all of your certificates (server, intermediate(s), root):

```
# copy all certificates into a single file:
        cat server-cert.pem intermediate.pem root.pem > /certs/cert-cha
in.crt
```

3. (Optional) If you are using a custom docker registry that has its own certificate, you need to copy this certificate chain into this same file:

```
cat docker-registry-cert.pem >> /certs/cert-chain.crt
```

4. Copy the global CA certificates into this new file:

```
# cat /etc/ssl/certs/ca-bundle.crt >> /certs/cert-chain.crt
```

5. Edit your deployment of runtime manager and add the new mount.

   Do not delete any existing objects.

   #kubectl edit deployment runtime-manager
6. Under VolumeMounts, add the following lines.

   Note that the text is white-space sensitive - use spaces and not tabs.

```
- mountPath: /etc/ssl/certs/ca-certificates.crt
        name: mycert
        subPath: cert-chain.crt #this should match the new file name crea
ted in step 4
```

   Under Volumes add the following text in the same edit:

```
- hostPath:
        path: /certs/  #this needs to match the folder created in step 1
        type: ""
        name: mycert
```

7. Save your changes:

   wq!

   Once saved, you will receive the message "deployment.apps/runtime-manager edited" and the pod will be restarted with your new changes.
8. To persist these changes across cluster restarts, use the following Knowledge Base article to create a kubernetes patch file for the runtime-manager deployment: https://community.cloudera.com/t5/Customer/Patching-CDSW-Kubernetes-deployments/ta-p/90241

Cloudera Bug: DSE-20530

DSE-17981 - Disable Scala runtimes in models, experiments and applications runtime selection

Scala Runtimes should not appear as an option for Models, Experiments, and Applications in the user interface. Currently Scala Runtimes only support Session and Jobs.

Known Issues and Limitations in ML Runtimes version 2024.10.01

You might run into some known issues while using ML Runtimes 2024.10.01.

DSE-25143 Assembling plots in PBJ R Runtimes does not work

When trying to plot additional content on already existing plots, PBJ R Runtimes throw an error. Plots can only be created using the plot function.

DSE-32839 Extra configuration needed when using Spark in a PBJ Workbench-based R Runtime

When using Spark in R workloads that are running PBJ Workbench Runtimes, the environmental variable R_LIBS_USER must be passed to the Spark executors with the value set to "/home/cdsw/.local/lib/R/<R_VERSION>/library".

E.g. when using `sparklyr` with a PBJ Workbench R 4.3 Runtime, the correct way to set up a `sparklyr` connection is:

```
library(sparklyr)
config <- spark_config()
config$spark.executorEnv.R_LIBS_USER="/home/cdsw/.local/lib/R/4.3/library"
sc <- spark_connect(config = config)
```

DSE-27222 Workbench editor for Python Runtimes and PBJ Runtimes cannot parse multiline strings properly

Workbench editor for Python 3.7 Runtime cannot parse multiline strings properly. Trying to evaluate multiline strings in some cases result in a "SyntaxError: EOF while scanning triple-quoted string literal" error message. This has been fixed in Python 3.8 and higher.

Workaround: Transform multiline strings to single line strings in code that is entered into the workbench directly.

Packages can fail to load in a session

When installing R or Python packages in a Session, the kernel might not be able to load the package in the same session, if a previous version of the package or its newly installed dependencies have been loaded in the same Session. Such issues are observable more often in PBJ R Runtimes, which automatically load basic R packages like `vctrs`, `lifecycle`, `rlang`, `cli` at session startup.

Workaround: Start a new session, import and use the newly installed package there.

Python Runtimes in Cloudera AI fail to import the setuptools Python library and can fail installing some Python packages

Python Runtimes in Cloudera AI fail to import the setuptools Python library and therefore can fail installing some Python packages when the library setuptools is present on the Runtime or is installed into the Cloudera AI project with version 60.0.0 or higher.

Python 3.10 Runtimes from the 2023.05 Runtime release ship with a newer version of setuptools, so customers can run into this issue when they are using that Runtime. Also they can run into this issue when they are using Custom Runtimes that has a newer setuptools library version or when they install a new setuptools version into their project (regardless of what Runtime they use).

Workaround: Set the environmental variable SETUPTOOLS_USE_DISTUTILS=stdlib either on a project level under Project Settings -> Advanced or on a workbench level under Site Administration -> Runtime -> Environment variables.

Version of jupyter-client Python package must be less than version 8 for PBJ Runtimes

Upgrading the Python package jupyter-client with a version greater than 7.4.9 can temporarily break a Project. Workloads using PBJ Runtimes will not be able to start Projects if the jupyter-client version is greater than 7.4.9.

Workaround: Launch the same version of Python, but not on a PBJ Runtime (either Workbench or JupyterLab). Open a Terminal window and uninstall the jupyter-client package from the Project by executing pip3 uninstall jupyter-client. Verify your change by running pip3 list and checking that the version of the jupyter-client package is less than version 8.

DSE-9818 JupyterLab Conda Technical Preview Runtime

Sessions

When starting a Notebook or a Console for a specific environment, the installed packages will be available and the interpreter used to evaluate the contents of the Notebook or Console will be the one installed in the environment. However, the Conda environment is not "activated" in these sessions, therefore commands like !which python will return with the base Python 3.10 interpreter on the Runtime. The recommended ways to modify a Conda environments or install packages are the following:

- conda commands must be used with the -n or --name argument to specify the environment, for example conda -n myenv install pandas
- When installing packages with pip, use the %pip magic to install packages in the active kernel's environment, for example %pip install pandas

Applications and Jobs

To start an Application or Job, first create a launcher Python script containing the following line: ! source activate <conda_env_name> && python <job / application script.py>

When starting the Application or Job, select the launcher script as the "Script".

Models

Models are currently not supported for the Conda Runtime.

Spark

Spark is not supported in JupyterLab Notebooks and Consoles.

Spark workloads are supported in activated Conda environments in JupyterLab Terminals, or in Jobs or Applications.

The CDSW libraries for Python and R are not available for the Conda Runtimes.

Adding a new ML Runtimes when using a custom root certificate might generate error messages

When trying to add new ML Runtimes, a number of error messages might appear in various places when using a custom root certificate. For example, you might see: "Could not fetch the image metadata" or "certificate signed by unknown authority". This is caused by the runtime-puller pods not having access to the custom root certificate that is in use.

Workaround:

1. Create a directory at any location on the master node:

   For example:

   mkdir -p /certs/
2. Copy the full server certificate chain into this folder. It is usually easier to create a single file with all of your certificates (server, intermediate(s), root):

```
# copy all certificates into a single file:
cat server-cert.pem intermediate.pem root.pem > /certs/cert-chain.crt
```

3. (Optional) If you are using a custom docker registry that has its own certificate, you need to copy this certificate chain into this same file:

```
cat docker-registry-cert.pem >> /certs/cert-chain.crt
```

4. Copy the global CA certificates into this new file:

```
# cat /etc/ssl/certs/ca-bundle.crt >> /certs/cert-chain.crt
```

5. Edit your deployment of runtime manager and add the new mount.

   Do not delete any existing objects.

   #kubectl edit deployment runtime-manager
6. Under VolumeMounts, add the following lines.

   Note that the text is white-space sensitive - use spaces and not tabs.

```
- mountPath: /etc/ssl/certs/ca-certificates.crt
   name: mycert
   subPath: cert-chain.crt #this should match the new file name created in
 step 4
```

   Under Volumes add the following text in the same edit:

```
- hostPath:
   path: /certs/  #this needs to match the folder created in step 1
   type: ""
name: mycert
```

7. Save your changes:

   wq!

   Once saved, you will receive the message "deployment.apps/runtime-manager edited" and the pod will be restarted with your new changes.
8. To persist these changes across cluster restarts, use the following Knowledge Base article to create a kubernetes patch file for the runtime-manager deployment: https://community.cloudera.com/t5/Customer/Patching-CDSW-Kubernetes-deployments/ta-p/90241

Cloudera Bug: DSE-20530

Spark Runtime Add-on required for Spark 2 integration with Scala Runtimes

Scala Runtimes on Cloudera AI require Spark Runtime Addon to enable Spark2 integration. Spark3 is not supported with the Scala Runtime.

DSE-17981 - Disable Scala runtimes in models, experiments and applications runtime selection

Scala Runtimes should not appear as an option for Models, Experiments, and Applications in the user interface. Currently Scala Runtimes only support Session and Jobs.

Known Issues and Limitations in ML Runtimes version 2024.05.02

You might run into some known issues while using ML Runtimes 2024.05.02.

TensorFlow 2.16.1 fails to work with GPUs

Due to the known TensorFlow issue https://github.com/tensorflow/tensorflow/issues/63362, it does not work with GPUs.

Workaround: Set the LD_LIBRARY_PATH environment variable to /home/cdsw/.local/lib/python3.X/site-packages/nvidia/cudnn/lib/:$LD_LIBRARY_PATH. Here, python3.X is the Python version you use, for example python3.9.

DSE-25143 Assembling plots in PBJ R Runtimes does not work

When trying to plot additional content on already existing plots, PBJ R Runtimes throw an error. Plots can only be created using the plot function.

DSE-32839 Extra configuration needed when using Spark in a PBJ Workbench-based R Runtime

When using Spark in R workloads that are running PBJ Workbench Runtimes, the environmental variable R_LIBS_USER must be passed to the Spark executors with the value set to "/home/cdsw/.local/lib/R/<R_VERSION>/library".

E.g. when using `sparklyr` with a PBJ Workbench R 4.3 Runtime, the correct way to set up a `sparklyr` connection is:

```
library(sparklyr)
config <- spark_config()
config$spark.executorEnv.R_LIBS_USER="/home/cdsw/.local/lib/R/4.3/library"
sc <- spark_connect(config = config)
```

DSE-27222 Workbench editor for Python Runtimes and PBJ Runtimes cannot parse multiline strings properly

Workbench editor for Python 3.7 Runtime cannot parse multiline strings properly. Trying to evaluate multiline strings in some cases result in a "SyntaxError: EOF while scanning triple-quoted string literal" error message. This has been fixed in Python 3.8 and higher.

Workaround: Transform multiline strings to single line strings in code that is entered into the workbench directly.

Packages can fail to load in a session

When installing R or Python packages in a Session, the kernel might not be able to load the package in the same session, if a previous version of the package or its newly installed dependencies have been loaded in the same Session. Such issues are observable more often in PBJ R Runtimes, which automatically load basic R packages like `vctrs`, `lifecycle`, `rlang`, `cli` at session startup.

Workaround: Start a new session, import and use the newly installed package there.

Python Runtimes in CML fail to import the setuptools Python library and can fail installing some Python packages

Python Runtimes in CML fail to import the setuptools Python library and therefore can fail installing some Python packages when the library setuptools is present on the Runtime or is installed into the CML project with version 60.0.0 or higher.

Python 3.10 Runtimes from the 2023.05 Runtime release ship with a newer version of setuptools, so customers can run into this issue when they are using that Runtime. Also they can run into this issue when they are using Custom Runtimes that has a newer setuptools library version or when they install a new setuptools version into their project (regardless of what Runtime they use).

Workaround: Set the environmental variable SETUPTOOLS_USE_DISTUTILS=stdlib either on a project level under Project Settings -> Advanced or on a workbench level under Site Administration -> Runtime -> Environment variables.

Version of jupyter-client Python package must be less than version 8 for PBJ Runtimes

Upgrading the Python package jupyter-client with a version greater than 7.4.9 can temporarily break a Project. Workloads using PBJ Runtimes will not be able to start Projects if the jupyter-client version is greater than 7.4.9.

Workaround: Launch the same version of Python, but not on a PBJ Runtime (either Workbench or JupyterLab). Open a Terminal window and uninstall the jupyter-client package from the Project by executing pip3 uninstall jupyter-client. Verify your change by running pip3 list and checking that the version of the jupyter-client package is less than version 8.

DSE-9818 JupyterLab Conda Tech Preview Runtime

Sessions

When starting a Notebook or a Console for a specific environment, the installed packages will be available and the interpreter used to evaluate the contents of the Notebook or Console will be the one installed in the environment. However, the Conda environment is not "activated" in these sessions, therefore commands like !which python will return with the base Python 3.10 interpreter on the Runtime. The recommended ways to modify a Conda environments or install packages are the following:

- conda commands must be used with the -n or --name argument to specify the environment, for example conda -n myenv install pandas
- When installing packages with pip, use the %pip magic to install packages in the active kernel's environment, for example %pip install pandas

Applications and Jobs

To start an Application or Job, first create a launcher Python script containing the following line: ! source activate <conda_env_name> && python <job / application script.py>

When starting the Application or Job, select the launcher script as the "Script".

Models

Models are currently not supported for the Conda Runtime.

Spark

Spark is not supported in JupyterLab Notebooks and Consoles.

Spark workloads are supported in activated Conda environments in JupyterLab Terminals, or in Jobs or Applications.

The CDSW libraries for Python and R are not available for the Conda Runtimes.

Adding a new ML Runtimes when using a custom root certificate might generate error messages

When trying to add new ML Runtimes, a number of error messages might appear in various places when using a custom root certificate. For example, you might see: "Could not fetch the image metadata" or "certificate signed by unknown authority". This is caused by the runtime-puller pods not having access to the custom root certificate that is in use.

Workaround:

1. Create a directory at any location on the master node:

   For example:

   mkdir -p /certs/
2. Copy the full server certificate chain into this folder. It is usually easier to create a single file with all of your certificates (server, intermediate(s), root):

   ```
   # copy all certificates into a single file:
   cat server-cert.pem intermediate.pem root.pem > /certs/cert-chain.crt
   ```
3. (Optional) If you are using a custom docker registry that has its own certificate, you need to copy this certificate chain into this same file:

   ```
   cat docker-registry-cert.pem >> /certs/cert-chain.crt
   ```

4. Copy the global CA certificates into this new file:

```
# cat /etc/ssl/certs/ca-bundle.crt >> /certs/cert-chain.crt
```

5. Edit your deployment of runtime manager and add the new mount.

   Do not delete any existing objects.

   #kubectl edit deployment runtime-manager
6. Under VolumeMounts, add the following lines.

   Note that the text is white-space sensitive - use spaces and not tabs.

```
- mountPath: /etc/ssl/certs/ca-certificates.crt
   name: mycert
   subPath: cert-chain.crt #this should match the new file name created in
 step 4
```

   Under Volumes add the following text in the same edit:

```
- hostPath:
   path: /certs/  #this needs to match the folder created in step 1
   type: ""
name: mycert
```

7. Save your changes:

   wq!

   Once saved, you will receive the message "deployment.apps/runtime-manager edited" and the pod will be restarted with your new changes.
8. To persist these changes across cluster restarts, use the following Knowledge Base article to create a kubernetes patch file for the runtime-manager deployment: https://community.cloudera.com/t5/Customer/Patching-CDSW-Kubernetes-deployments/ta-p/90241

Cloudera Bug: DSE-20530

Spark Runtime Add-on required for Spark 2 integration with Scala Runtimes

Scala Runtimes on CML require Spark Runtime Addon to enable Spark2 integration. Spark3 is not supported with the Scala Runtime.

DSE-17981 - Disable Scala runtimes in models, experiments and applications runtime selection

Scala Runtimes should not appear as an option for Models, Experiments, and Applications in the user interface. Currently Scala Runtimes only support Session and Jobs.

Known Issues and Limitations in ML Runtimes older releases

You might run into some known issues while using ML Runtimes.

TensorFlow 2.16.1 fails to work with GPUs

Due to the known TensorFlow issue https://github.com/tensorflow/tensorflow/issues/63362, it does not work with GPUs.

Workaround: Set the LD_LIBRARY_PATH environment variable to /home/cdsw/.local/lib/python3.X/site-packages/nvidia/cudnn/lib/:$LD_LIBRARY_PATH. Here, python3.X is the Python version you use, for example python3.9.

DSE-25143 Assembling plots in PBJ R Runtimes does not work

When trying to plot additional content on already existing plots, PBJ R Runtimes throw an error. Plots can only be created using the plot function.

DSE-32839 Extra configuration needed when using Spark in a PBJ Workbench-based R Runtime

When using Spark in R workloads that are running PBJ Workbench Runtimes, the environmental variable R_LIBS_USER must be passed to the Spark executors with the value set to "/home/cdsw/.local/lib/R/<R_VERSION>/library".

E.g. when using `sparklyr` with a PBJ Workbench R 4.3 Runtime, the correct way to set up a `sparklyr` connection is:

```
library(sparklyr)
config <- spark_config()
config$spark.executorEnv.R_LIBS_USER="/home/cdsw/.local/lib/R/4.3/library"
sc <- spark_connect(config = config)
```

DSE-27222 Workbench editor for Python Runtimes and PBJ Runtimes cannot parse multiline strings properly

Workbench editor for Python 3.7 Runtime cannot parse multiline strings properly. Trying to evaluate multiline strings in some cases result in a "SyntaxError: EOF while scanning triple-quoted string literal" error message. This has been fixed in Python 3.8 and higher.

Workaround: Transform multiline strings to single line strings in code that is entered into the workbench directly.

Packages can fail to load in a session

When installing R or Python packages in a Session, the kernel might not be able to load the package in the same session, if a previous version of the package or its newly installed dependencies have been loaded in the same Session. Such issues are observable more often in PBJ R Runtimes, which automatically load basic R packages like `vctrs`, `lifecycle`, `rlang`, `cli` at session startup.

Workaround: Start a new session, import and use the newly installed package there.

Python Runtimes in CML fail to import the setuptools Python library and can fail installing some Python packages

Python Runtimes in CML fail to import the setuptools Python library and therefore can fail installing some Python packages when the library setuptools is present on the Runtime or is installed into the CML project with version 60.0.0 or higher.

Python 3.10 Runtimes from the 2023.05 Runtime release ship with a newer version of setuptools, so customers can run into this issue when they are using that Runtime. Also they can run into this issue when they are using Custom Runtimes that has a newer setuptools library version or when they install a new setuptools version into their project (regardless of what Runtime they use).

Workaround: Set the environmental variable SETUPTOOLS_USE_DISTUTILS=stdlib either on a project level under Project Settings -> Advanced or on a workbench level under Site Administration -> Runtime -> Environment variables.

Version of jupyter-client Python package must be less than version 8 for PBJ Runtimes

Upgrading the Python package jupyter-client with a version greater than 7.4.9 can temporarily break a Project. Workloads using PBJ Runtimes will not be able to start Projects if the jupyter-client version is greater than 7.4.9.

Workaround: Launch the same version of Python, but not on a PBJ Runtime (either Workbench or JupyterLab). Open a Terminal window and uninstall the jupyter-client package from the Project by executing pip3 uninstall jupyter-client. Verify your change by running pip3 list and checking that the version of the jupyter-client package is less than version 8.

DSE-9818 JupyterLab Conda Tech Preview Runtime

Sessions

When starting a Notebook or a Console for a specific environment, the installed packages will be available and the interpreter used to evaluate the contents of the Notebook or Console will be the one installed in the environment. However, the Conda environment is not "activated" in these sessions, therefore commands like !which python will return with the base Python 3.10 interpreter on the Runtime. The recommended ways to modify a Conda environments or install packages are the following:

- conda commands must be used with the -n or --name argument to specify the environment, for example conda -n myenv install pandas
- When installing packages with pip, use the %pip magic to install packages in the active kernel's environment, for example %pip install pandas

Applications and Jobs

To start an Application or Job, first create a launcher Python script containing the following line: ! source activate <conda_env_name> && python <job / application script.py>

When starting the Application or Job, select the launcher script as the "Script".

Models

Models are currently not supported for the Conda Runtime.

Spark

Spark is not supported in JupyterLab Notebooks and Consoles.

Spark workloads are supported in activated Conda environments in JupyterLab Terminals, or in Jobs or Applications.

The CDSW libraries for Python and R are not available for the Conda Runtimes.

Adding a new ML Runtimes when using a custom root certificate might generate error messages

When trying to add new ML Runtimes, a number of error messages might appear in various places when using a custom root certificate. For example, you might see: "Could not fetch the image metadata" or "certificate signed by unknown authority". This is caused by the runtime-puller pods not having access to the custom root certificate that is in use.

Workaround:

1. Create a directory at any location on the master node:

   For example:

   mkdir -p /certs/
2. Copy the full server certificate chain into this folder. It is usually easier to create a single file with all of your certificates (server, intermediate(s), root):

   ```
   # copy all certificates into a single file:
   cat server-cert.pem intermediate.pem root.pem > /certs/cert-chain.crt
   ```
3. (Optional) If you are using a custom docker registry that has its own certificate, you need to copy this certificate chain into this same file:

   ```
   cat docker-registry-cert.pem >> /certs/cert-chain.crt
   ```
4. Copy the global CA certificates into this new file:

   ```
   # cat /etc/ssl/certs/ca-bundle.crt >> /certs/cert-chain.crt
   ```

5. Edit your deployment of runtime manager and add the new mount.

   Do not delete any existing objects.

   #kubectl edit deployment runtime-manager
6. Under VolumeMounts, add the following lines.

   Note that the text is white-space sensitive - use spaces and not tabs.

```
- mountPath: /etc/ssl/certs/ca-certificates.crt
   name: mycert
   subPath: cert-chain.crt #this should match the new file name created in
 step 4
```

   Under Volumes add the following text in the same edit:

```
- hostPath:
   path: /certs/  #this needs to match the folder created in step 1
   type: ""
name: mycert
```

7. Save your changes:

   wq!

   Once saved, you will receive the message "deployment.apps/runtime-manager edited" and the pod will be restarted with your new changes.
8. To persist these changes across cluster restarts, use the following Knowledge Base article to create a kubernetes patch file for the runtime-manager deployment: https://community.cloudera.com/t5/Customer/Patching-CDSW-Kubernetes-deployments/ta-p/90241

Cloudera Bug: DSE-20530

Spark Runtime Add-on required for Spark 2 integration with Scala Runtimes

Scala Runtimes on CML require Spark Runtime Addon to enable Spark2 integration. Spark3 is not supported with the Scala Runtime.

DSE-17981 - Disable Scala runtimes in models, experiments and applications runtime selection

Scala Runtimes should not appear as an option for Models, Experiments, and Applications in the user interface. Currently Scala Runtimes only support Session and Jobs.

Cloudera AI

Securing Cloudera AI

Date published: 2020-07-16
Date modified: 2024-11-21

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring external authentication with LDAP and SAML..... 4
Configuring LDAP/Active Directory authentication..... 4
LDAP general settings.....4
LDAP group settings..... 5
Test LDAP Configuration..... 6
Configuring SAML authentication.....6
Configuration options..... 7

Configuring HTTP Headers for Cloudera AI..... 8
Enable HTTP security headers.....9
Enable HTTP Strict Transport Security (HSTS)..... 9
Enable Cross-Origin Resource Sharing (CORS)..... 10

SSH Keys..... 10
Personal key.....10
Team key.....10
Adding an SSH key to GitHub..... 11
Creating an SSH tunnel.....11

Hadoop authentication for Cloudera AI Workbenches.....11

Cloudera AI and outbound network access.....12

Non-transparent proxy and egress trusted list.....12

Configuring external authentication with LDAP and SAML

Important: Cloudera recommends you leverage Single Sign-On for users via the Cloudera Management Console. For instructions on how to configure this, see Configuring User Access to Cloudera AI. If you cannot do this, we recommend contacting Cloudera Support before attempting to use the LDAP or SAML instructions provided in this section.

Cloudera AI supports user authentication against its internal local database, and against external services such as Active Directory, OpenLDAP-compatible directory services, and SAML 2.0 Identity Providers. By default, Cloudera AI performs user authentication against its internal local database. This topic describes the signup process for the first user, how to configure authentication using LDAP, Active Directory or SAML 2.0, and an optional workaround that allows site administrators to bypass external authentication by logging in using the local database in case of misconfiguration.

Configuring LDAP/Active Directory authentication

This topic describes how to set up LDAP authentication for a workbench.

Important: This is not the recommended method to set up LDAP authentication. Cloudera recommends you use the Cloudera Management Console to set this up: Configuring User Access to Cloudera AI.

Cloudera AI supports both search bind and direct bind operations to authenticate against an LDAP or Active Directory directory service. The search bind authentication mechanism performs an ldapsearch against the directory service, and binds using the found Distinguished Name (DN) and password provided. The direct bind authentication mechanism binds to the LDAP server using a username and password provided at login.

You can configure Cloudera AI to use external authentication methods by clicking the Admin link on the left sidebar and selecting the Security tab. Select LDAP from the list to start configuring LDAP properties.

LDAP general settings

Lists the general settings required to configure LDAP authentication.

- LDAP Server URI: Required. The URI of the LDAP/Active Directory server against which Cloudera AI shall authenticate. For example, ldaps://ldap.*COMPANY*.com:636.
- Use Direct Bind: If checked, the username and password provided at login are used with the LDAP Username Pattern for binding to the LDAP server. If unchecked, Cloudera AI uses the search bind mechanism and two configurations, LDAP Bind DN and LDAP Bind Password, are required to perform the ldapsearch against the LDAP server.
- LDAP Bind DN: Required when using search bind. The DN to bind to for performing ldapsearch. For example, cn=admin,dc=company,dc=com.
- LDAP Bind Password: Required when using search bind. This is the password for the LDAP Bind DN.
- LDAP Search Base: Required. The base DN from which to search for the provided LDAP credentials. For example, ou=Engineering,dc=company,dc=com.
- LDAP User Filter: Required. The LDAP filter for searching for users. For example, (&(sAMAccountName={0})(objectclass=person)). The {0} placeholder will be replaced with the username provided at login.

- LDAP User Username Attribute: Required. The case-sensitive username attribute of the LDAP directory service. This is used by Cloudera AI to perform the bind operation and extract the username from the response. Common values are uid, sAMAccountName, or userPrincipalName.

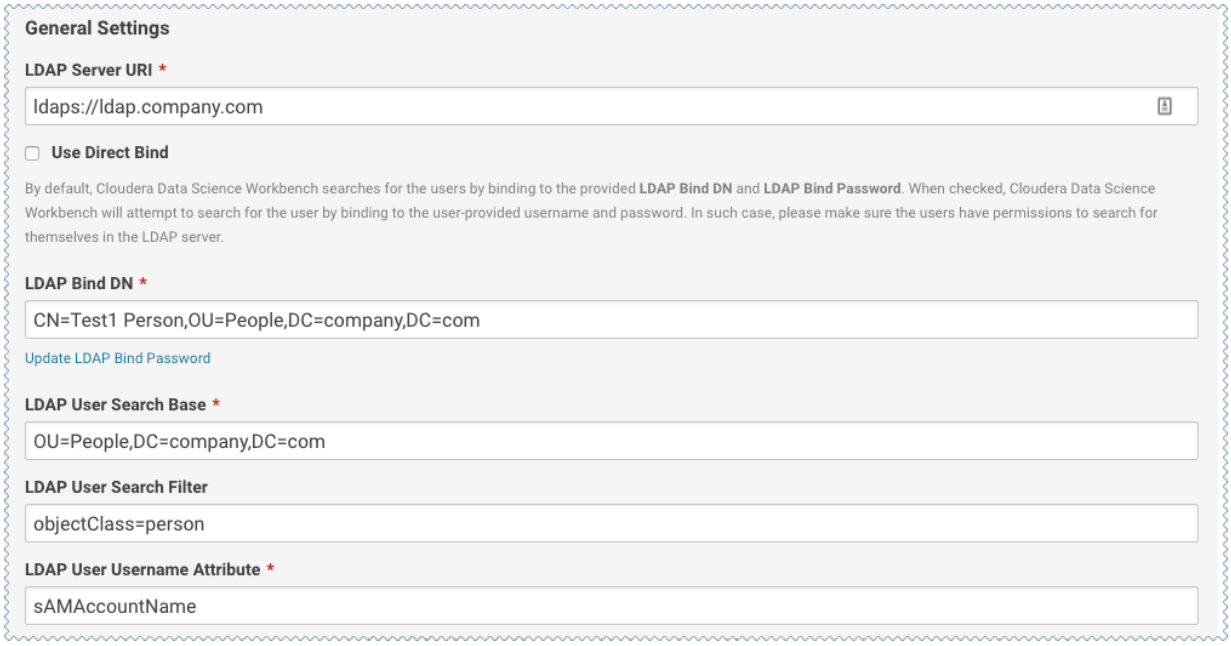

When you select Use Direct Bind, Cloudera AI performs a direct bind to the LDAP server using the LDAP Username Pattern with the credentials provided on login (not LDAP Bind DN and LDAP Bind Password).

By default, Cloudera AI performs an LDAP search using the bind DN and credentials specified for the LDAP Bind DN and LDAP Bind Password configurations. It searches the subtree, starting from the base DN specified for the LDAP Search Base field, for an entry whose attribute specified in LDAP User Username Attribute, has the same value as the username provided on login. Cloudera AI then validates the user-provided password against the DN found as a result of the search.

LDAP group settings

In addition to the general LDAP settings, you can use group settings to restrict the access to Cloudera AI to certain groups in LDAP.

- LDAP Group Search Base: The base distinguished name (DN) where Cloudera AI will search for groups.
- LDAP Group Search Filter: The LDAP filter that Cloudera AI will use to determine whether a user is affiliated to a group.

  A group object in LDAP or Active Directory typically has one or more member attributes that stores the DNs of users in the group. If LDAP Group Search Filter is set to member={0}, Cloudera AI will automatically substitute the {0} placeholder for the DN of the authenticated user.
- LDAP User Groups: A list of LDAP groups whose users have access to Cloudera AI. When this property is set, only users that successfully authenticate themselves AND are affiliated to at least one of the groups listed here, will be able to access Cloudera AI.

  If this property is left empty, all users that can successfully authenticate themselves to LDAP will be able to access Cloudera AI.

- LDAP Full Administrator Groups: A list of LDAP groups whose users are automatically granted the site administrator role on Cloudera AI.

  The groups listed under LDAP Full Administrator Groups do not need to be listed again under the LDAP User Groups property.

  Figure 1: Example

  If you want to restrict access to Cloudera AI to members of a group whose DN is:

  ```
  CN=MLUsers,OU=Groups,DC=company,DC=com
  ```

  And automatically grant site administrator privileges to members of a group whose DN is:

  ```
  CN=MLAdmins,OU=Groups,DC=company,DC=com
  ```

  Add the CNs of both groups to the following settings in Cloudera AI:

  - LDAP User Groups: MLUsers
  - LDAP Full Administrator Groups: MLAdmins

Test LDAP Configuration

Use the **Test LDAP Configuration** form to test your settings.

You can test your LDAP/Active Directory configuration by entering your username and password in the Test LDAP Configuration section. This form simulates the user login process and allows you to verify the validity of your LDAP/Active Directory configuration without opening a new window.

Before using this form, make sure you click Update to save the LDAP configuration you want to test.

Configuring SAML authentication

This topic describes how to set up SAML for Single Sign-on authentication for a workbench.

Important: This is not the recommended method to set up SSO. Cloudera recommends you use the Cloudera Management Console to set this up: Configuring User Access to Cloudera AI.

Cloudera AI supports the Security Assertion Markup Language (SAML) for Single Sign-on (SSO) authentication; in particular, between an identity provider (IDP) and a service provider (SP). The SAML specification defines three roles: the principal (typically a user), the IDP, and the SP. In the use case addressed by SAML, the principal (user agent) requests a service from the service provider. The service provider requests and obtains an identity assertion from the IDP. On the basis of this assertion, the SP can make an access control decision—in other words it can decide whether to perform some service for the connected principal.

Note: The user sync feature only works with the SAML IDP provided by the control plane. If a custom SAML IDP is provided then customer has to make sure to turn usersync off. Otherwise, there is a risk that users will be deactivated and therefore causing cron jobs scheduled by users that are deactivated to fail.

The primary SAML use case is called web browser single sign-on (SSO). A user with a user agent (usually a web browser) requests a web resource protected by a SAML SP. The SP, wanting to know the identity of the requesting user, issues an authentication request to a SAML IDP through the user agent. In the context of this terminology, Cloudera AI operates as a SP.

Cloudera AI supports both SP- and IDP-initiated SAML 2.0-based SSO. Its Assertion Consumer Service (ACS) API endpoint is for consuming assertions received from the Identity Provider. If your Cloudera AI domain root were cdsw.*COMPANY*.com, then this endpoint would be available at http://cdsw.*COMPANY*.com/api/v1/saml/acs. SAML 2.0 metadata is available at http://cdsw.*COMPANY*.com/api/v1/saml/metadata for IDP-initiated SSO. Cloudera AI uses HTTP Redirect Binding for authentication requests and expects to receive responses from HTTP POST Binding. Note: When visiting these pages, view the Page Source in the browser to see the full XML.

When Cloudera AI receives the SAML responses from the Identity Provider, it expects to see at least the following user attributes in the SAML responses:

- The unique identifier or username. Valid attributes are:
  - uid
  - urn:oid:0.9.2342.19200300.100.1.1
- The email address. Valid attributes are:
  - mail
  - email
  - urn:oid:0.9.2342.19200300.100.1.3
- The common name or full name of the user. Valid attributes are:
  - cn
  - urn:oid:2.5.4.3

  In the absence of the cn attribute, Cloudera AI will attempt to use the following user attributes, if they exist, as the full name of the user:

  - The first name of the user. Valid attributes are:
    - givenName
    - urn:oid:2.5.4.42
  - The last name of the user. Valid attributes are:
    - sn
    - urn:oid:2.5.4.4

Configuration options

List of properties to configure SAML authentication and authorization in Cloudera AI.

Cloudera AI settings

- Entity ID: Required. A globally unique name for Cloudera AI as a Service Provider. This is typically the URI.
- NameID Format: Optional. The name identifier format for both Cloudera AI and Identity Provider to communicate with each other regarding a user. Default: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress.
- Authentication Context: Optional. SAML authentication context classes are URIs that specify authentication methods used in SAML authentication requests and authentication statements. Default: urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport.

Signing SAML authentication requests

- CDSW Private Key for Signing Authentication Requests: Optional. If you upload a private key, you must upload a corresponding certificate as well so that the Identity Provider can use the certificate to verify the authentication requests sent by Cloudera AI. You can upload the private key used for both signing authentication requests sent to Identity Provider and decrypting assertions received from the Identity Provider.
- Cloudera AI Certificate for Signature Validation: Required if the Cloudera AI Private Key is set, otherwise optional. You can upload a certificate in the PEM format for the Identity Provider to verify the authenticity of the authentication requests generated by Cloudera AI. The uploaded certificate is made available at the http://cdsw.*COMPANY*.com/api/v1/saml/metadata endpoint.

SAML assertion decryption

Cloudera AI uses the following properties to support SAML assertion encryption & decryption.

- Cloudera AI Certificate for Encrypting SAML Assertions - Must be configured on the Identity Provider so that Identity Provider can use it for encrypting SAML assertions for Cloudera AI

- Cloudera AI Private Key for Decrypting SAML Assertions - Used to decrypt the encrypted SAML assertions.

Identity provider

- Identity Provider SSO URL: Required. The entry point of the Identity Provider in the form of URI.
- Identity Provider Signing Certificate: Optional. Administrators can upload the X.509 certificate of the Identity Provider for Cloudera AI to validate the incoming SAML responses.

  Cloudera AI extracts the Identity Provider SSO URL and Identity Provider Signing Certificate information from the uploaded Identity Provider Metadata file. Cloudera AI also expects all Identity Provider metadata to be defined in a <md:EntityDescriptor> XML element with the namespace "urn:oasis:names:tc:SAML:2.0:metadata", as defined in the SAMLMeta-xsd schema.

  For on-premises deployments, you must provide a certificate and private key, generated and signed with your trusted Certificate Authority, for Cloudera AI to establish secure communication with the Identity Provider.

Authorization

When you're using SAML 2.0 authentication, you can use the following properties to restrict the access to Cloudera AI to certain groups of users:

- SAML Attribute Identifier for User Role: The Object Identifier (OID) of the user attribute that will be provided by your identity provider for identifying a user’s role/affiliation. You can use this field in combination with the following SAML User Groups property to restrict access to Cloudera AI to only members of certain groups.

  For example, if your identity provider returns the OrganizationalUnitName user attribute, you would specify the OID of the OrganizationalUnitName, which is urn:oid:2.5.4.11, as the value for this property.
- SAML User Groups: A list of groups whose users have access to Cloudera AI. When this property is set, only users that are successfully authenticated AND are affiliated to at least one of the groups listed here, will be able to access Cloudera AI.

  For example, if your identity provider returns the OrganizationalUnitName user attribute, add the value of this attribute to the SAML User Groups list to restrict access to Cloudera AI to that group.

  If this property is left empty, all users that can successfully authenticate themselves will be able to access Cloudera AI.
- SAML Full Administrator Groups: A list of groups whose users are automatically granted the site administrator role on Cloudera AI.

  The groups listed under SAML Full Administrator Groups do not need to be listed again under the SAML User Groups property.

Configuring HTTP Headers for Cloudera AI

This topic explains how to customize the HTTP headers that are accepted by Cloudera AI.

Required Role: Site Administrator

These properties are available under the site administrator panel at Admin Security .

Enable Cross-Origin Resource Sharing (CORS)

Most modern browsers implement the Same-Origin Policy, which restricts how a document or a script loaded from one origin can interact with a resource from another origin. When the Enable cross-origin resource sharing property is enabled on Cloudera AI, web servers will include the Access-Control-Allow-Origin: * HTTP header in their HTTP responses. This gives web applications on different domains permission to access the Cloudera AI API through browsers.

This property is disabled by default .

If this property is disabled, web applications from different domains will not be able to programmatically communicate with the Cloudera AI API through browsers.

Enable HTTP security headers

When Enable HTTP security headers is enabled, the following HTTP headers will be included in HTTP responses from servers:

- X-XSS-Protection
- X-DNS-Prefetch-Control
- X-Frame-Options
- X-Download-Options
- X-Content-Type-Options

This property is enabled by default .

Disabling this property could leave your Cloudera AI deployment vulnerable to clickjacking, cross-site scripting (XSS), or any other injection attacks.

Enable HTTP Strict Transport Security (HSTS)

Note: Without TLS/SSL enabled, configuring this property will have no effect on your browser.

When both TLS/SSL and this property (Enable HTTP Strict Transport Security (HSTS)) are enabled, Cloudera AI will inform your browser that it should never load the site using HTTP. Additionally, all attempts to access Cloudera AI using HTTP will automatically be converted to HTTPS.

This property is disabled by default .

If you ever need to downgrade to back to HTTP, use the following sequence of steps: First, deactivate this checkbox to disable HSTS and restart Cloudera AI. Then, load the Cloudera AI web application in each browser to clear the respective browser's HSTS setting. Finally, disable TLS/SSL across the cluster. Following this sequence should help avoid a situation where users get locked out of their accounts due to browser caching.

Enable HTTP security headers

When Enable HTTP security headers is enabled, the following HTTP headers will be included in HTTP responses from servers:

- X-XSS-Protection
- X-DNS-Prefetch-Control
- X-Frame-Options
- X-Download-Options
- X-Content-Type-Options

This property is enabled by default .

Disabling this property could leave your Cloudera AI deployment vulnerable to clickjacking, cross-site scripting (XSS), or any other injection attacks.

Enable HTTP Strict Transport Security (HSTS)

Note: Without TLS/SSL enabled, configuring this property will have no effect on your browser.

When both TLS/SSL and this property (Enable HTTP Strict Transport Security (HSTS)) are enabled, Cloudera AI will inform your browser that it should never load the site using HTTP. Additionally, all attempts to access Cloudera AI using HTTP will automatically be converted to HTTPS.

This property is disabled by default .

If you ever need to downgrade to back to HTTP, use the following sequence of steps: First, deactivate this checkbox to disable HSTS and restart Cloudera AI. Then, load the Cloudera AI web application in each browser to clear the respective browser's HSTS setting. Finally, disable TLS/SSL across the cluster. Following this sequence shall help tto avoid a situation where users get locked out of their accounts due to browser caching.

Enable Cross-Origin Resource Sharing (CORS)

Most modern browsers implement the Same-Origin Policy, which restricts how a document or a script loaded from one origin can interact with a resource from another origin. When the Enable cross-origin resource sharing property is enabled on Cloudera AI, web servers will include the Access-Control-Allow-Origin: * HTTP header in their HTTP responses. This gives web applications on different domains permission to access the Cloudera AI API through browsers.

This property is disabled by default .

If this property is disabled, web applications from different domains will not be able to programmatically communicate with the Cloudera AI API through browsers.

SSH Keys

This topic describes the different types of SSH keys used by Cloudera AI, and how you can use those keys to authenticate to an external service such as GitHub.

Personal key

Cloudera AI automatically generates an SSH key pair for your user account. You can rotate the key pair and view your public key on your user settings page. It is not possible for anyone to view your private key.

Every console you run has your account's private key loaded into its SSH-agent. Your consoles can use the private key to authenticate to external services, such as GitHub. For instructions, see *Adding an SSH Key to GitHub*.

Related Information

Adding an SSH Key to GitHub

Team key

Team SSH keys provide a useful way to give an entire team access to external resources such as databases or GitHub repositories (as described in the next section).

Like Cloudera AI users, each Cloudera AI team has an associated SSH key. You can access the public key from the team's account settings. Click Account, then select the team from the drop-down menu at the upper right corner of the page.

When you launch a console in a project owned by a team, you can use that team's SSH key from within the console.

Adding an SSH key to GitHub

Cloudera AI creates a public SSH key for each account. You can add this SSH public key to your GitHub account if you want to use password-protected GitHub repositories to create new projects or collaborate on projects.

Procedure

1. Sign in to Cloudera AI.
2. Go to the upper right drop-down menu and switch context to the account whose key you want to add. This could be a individual user account or a team account.
3. On the left sidebar, click User Settings.
4. Go to the Outbound SSH tab and copy the User Public SSH Key.
5. Sign in to your GitHub account and add the Cloudera AI key copied in the previous step to your GitHub account. For instructions, refer the GitHub documentation on Adding a new SSH key to your GitHub account.

Creating an SSH tunnel

You can use your SSH key to connect Cloudera AI to an external database or cluster by creating an SSH tunnel.

About this task

In some environments, external databases and data sources reside behind restrictive firewalls. A common pattern is to provide access to these services using a bastion host with only the SSH port open. Cloudera AI provides a convenient way to connect to such resources using an SSH tunnel.

If you create an SSH tunnel to an external server in one of your projects, then all engines that you run in that project are able to connect securely to a port on that server by connecting to a local port. The encrypted tunnel is completely transparent to the user and code.

Procedure

1. Open the Project Settings page.
2. Open the Tunnels tab.
3. Click New Tunnel.
4. Enter the server IP Address or DNS hostname.
5. Enter your username on the server.
6. Enter the local port that should be proxied, and to which remote port on the server.

What to do next

On the remote server, configure SSH to accept password-less logins using your individual or team SSH key. Often, you can do so by appending the SSH key to the file /home/username/.ssh/authorized_keys.

Hadoop authentication for Cloudera AI Workbenches

Cloudera AI does not assume that your Kerberos principal is always the same as your login information. Therefore, you will need to make sure Cloudera AI knows your Kerberos identity when you sign in.

About this task

This procedure is required if you want to run Spark workloads in an Cloudera AI Workbench. This is also required if connecting Cloudera Data Visualization running in Cloudera AI to an Impala instance using Kerberos for authentication.

Procedure

1. Navigate to your Cloudera AI Workbench.
2. Go to the top-right dropdown menu, click Account settings Hadoop Authentication .
3. To authenticate, either enter your password or click Upload Keytab to upload the keytab file directly.

Results

Once successfully authenticated, Cloudera AI uses your stored credentials to ensure you are secure when running workloads.

Cloudera AI and outbound network access

Cloudera AI expects access to certain external networks. See the related information *Outbound internet access and proxy* for further information.

Note: The outbound network access destinations listed in *Outbound internet access and proxy* are only the minimal set required for Cloudera Data Platform installation and operation. For environments with limited outbound internet access due to using a firewall or proxy, access to Python or R package repositories such as Python Package Index or CRAN may need to be whitelisted if your use cases require installing packages from those repositories. Alternatively, you may consider creating mirrors of those repositories within your environment.

Related Information

Outbound internet access and proxy

Non-transparent proxy and egress trusted list

Cloudera AI, when used on AWS Public Cloud, supports non-transparent proxies. Non-transparent proxy enables Cloudera AI to proxy web requests without requiring any particular browser setup.

Egress Trusted List

In normal operation, Cloudera AI requires the ability to reach several external domains. See Outbound internet access and proxy for more information.

Related Information

Outbound internet access and proxy

Cloudera AI

Site Administration

Date published: 2020-07-16
Date modified: 2025-01-29

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing Users 5

Service Accounts 6
- Creating a machine user and synchronizing to workbench 6
- Sync machine users from the Synced team 6
- Run workloads using a service account 7

Configuring Quotas 7

Creating Resource profiles 8

Disable or deprecate Runtime addons 9

Onboarding Business Users 10

Adding a Collaborator 11

User Roles 11
- Business Users and Cloudera AI 12
- Managing your Personal Account 12
- Creating a Team 13
- Managing a Team Account 14
- Managing a Synced Team 14

Monitoring Cloudera AI Activity 15
- Tracked User events 16
- Monitoring User Events 19

Monitoring active Models across the Workbench 21

Monitoring and alerts 22

Application polling endpoint 22

Choosing default engine 22

Controlling User access to features 23

Cloudera AI email notifications 25

Downloading diagnostic bundles for a workbench 25

Web session timeouts 26

Project garbage collection 26

Ephemeral storage 26

Installing a non-transparent proxy in a Cloudera AI environment 27

Ports used by Cloudera AI 29

Export Usage List 29

Private cluster support 29

Enable a private cluster 30

User Defined Routing (UDR) 30

Embed a Cloudera AI application in an external website 32

Setting up Cloudera AI Workbenches for high volume Workloads 33

Host name required by Learning Hub 34

Managing Users

This topic describes how to manage an Cloudera AI Workbench as a site administrator. Site administrators can monitor and manage all user activity across a workbench, add new custom engines, and configure certain security settings.

By default, the first user that logs in to a workbench must always be a site administrator. That is, they must have the MLAdmin role granted by a Cloudera Data Platform PowerUser.

Important: Site administrators have complete access to all activity on the deployment. This includes access to all teams and projects on the deployment, even if they have not been explicitly added as team members or collaborators.

Only site administrators have access to a Site Administration dashboard that can be used to manage the workbench. To access the site administrator dashboard:

1. Go to the Cloudera AI web application and log in as a site administrator.
2. On the left sidebar, click Site Administration. You will see an array of tabs for all the tasks you can perform as a site administrator.

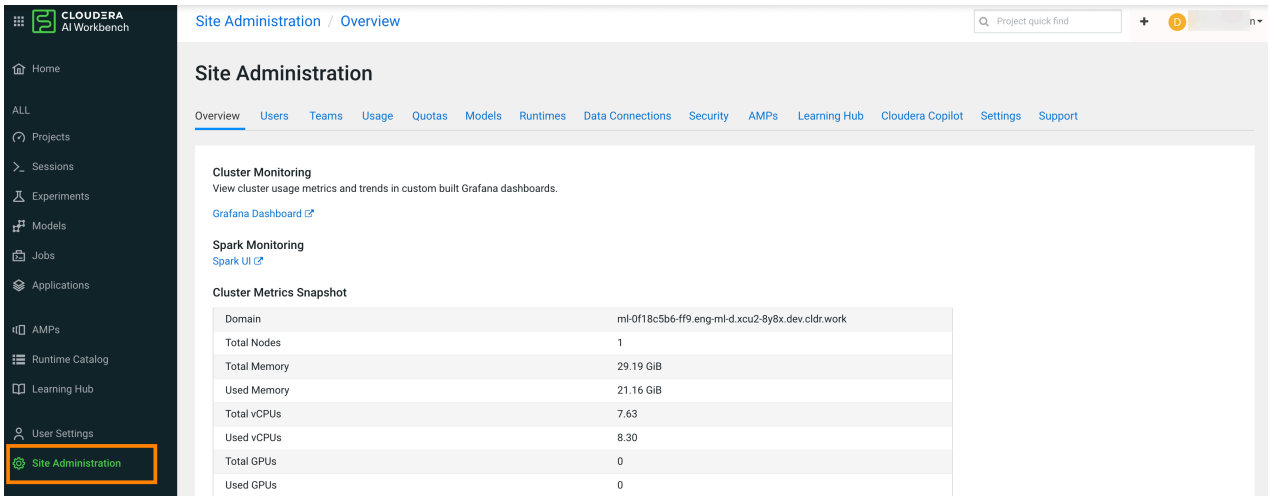

Monitoring Users

The Users tab on the **Administrator** dashboard displays the complete list of users. You can see which users are currently active, and when a user last logged in to Cloudera AI. You can search for a user by entering their User ID, Username, or Email in the User quick find box. To modify a user's username, email or permissions, click the Edit button under the **Action** column.

Note: The Disabled checkbox does not have any effect when external authentication is in use.

Synchronizing Users

You can synchronize users within an Cloudera AI Workbench with those users that have been defined access at the Environment level (through the MLAdmin, MLUser, and MLBusinessUser roles). Doing so for new users enables you to take administrative actions such as setting Team assignments, defining Project Collaborators, and more, all prior to the new users' first time logging in to the Workbench.

To synchronize users, go to Site Administration Users , and click Run Sync Now. This adds any users defined at the Environment level to the workbench, updates any role changes that have been made, and deactivates any users that have been deactivated.

Note: The Administrator shall periodically perform user synchronization to ensure that users who are deactivated on the environment level are also deactivated in Cloudera AI.

Synchronizing Groups

Groups of users can be created in the Cloudera Data Platform management console and imported to Cloudera AI. However, changes made in Cloudera Data Platform do not automatically update in Cloudera AI. You need to manually trigger an update, using Sync Teams. For more information, see *Creating a Team*.

Related Information

Cloudera AI email notifications

Creating a Team

Service Accounts

Service accounts are used by machine users that require a user account, without needing to use an account of an actual user.

Like other users, this machine user can be granted necessary permissions and roles, and be added as a collaborator to projects in order to run workloads. Machine users can also create projects and workloads.

Creating a machine user and synchronizing to workbench

The MLAdmin role is required to create machine users.

Procedure

1. In Management Console, go to User Management.
2. In Actions, click Create Machine User.
3. Enter a name for the machine user and click Create.
4. In the Cloudera AI Workbenches UI, find your workbench and in Actions, click Manage Access.
5. Search for the machine user name you just created, and in Update Resource Roles, assign the MLWorkspaceAdmin or MLWorkspaceUser role. Click Update Roles.

   Note: Machine users can alternatively be assigned to environments.

6. Return to the workbench in Cloudera AI, and in Site Administration Users , click Run Sync Now to manually synchronize the users for the workbench.
7. In Site Administration, search for the machine user name.

Sync machine users from the Synced team

You can sync machine users that are part of a synced team to your project.

Procedure

1. In Management Console User Management Groups , click Create Group.
2. Enter the name for the group, and click Create.
3. Click Add Members to search for and add group members, including machine users.
4. To add the team (group) to your environment, go to Environments Actions Manage Access .

5. Click Update Role to update the role as follows, and click Update Roles.
   - Environment User: Only users who have read access to the environment are synced. Alternatively, you can assign the Environment User role to the machine user.
   - MLAdmin or MLUser role: only users with either role are synced to Cloudera AI Workbench.
6. Click Synchronize Users and wait for synchronization to complete. Then return to your Cloudera AI Workbench.
7. In Site Administration Teams , select Sync Teams and then choose the group to sync.
8. Click Create Team, and the team is created in Cloudera AI.

What to do next

To add members to a synced team, add them in the control plane and synchronize them to Cloudera AI via the Site Administration Teams Sync Teams option. You cannot add users to a group manually in Cloudera AI.

Run workloads using a service account

You can run various types of workloads using a service account. First, make sure the service account is available in your project.

1. Create a project, or enter an existing project.
2. In Collaborators, add the service account. Specify the Operator or Admin role and click Add.

Run a job with a service account

1. In Jobs, click New Job.
2. For Run Job as, select Service Account and choose the account from the list.
3. Make other settings as needed, and click Create Job.

Run an application with a service account

1. Click New Application.
2. For Run Job as, select Service Account and choose the account from the list.
3. Make other settings as needed, and click Create Application.

Run a model with a service account

1. In Models, click New Model.
2. For Deploy Model as, select Service Account and choose the account from the list.
3. Make other settings as needed, and click Deploy Model.

Configuring Quotas

This topic describes how to configure CPU, GPU, and memory quotas for users of an Cloudera AI Workbench.

Before you begin

Required Role: MLAdmin

Make sure you are assigned the MLAdmin role in Cloudera Data Platform. Only users with the MLAdmin role will be logged into Cloudera AI Workbenches with Site Administrator privileges.

There are two types of quota: Default and Custom. Default quotas apply to all users of the workbench. Custom quotas apply to individual users in the workbench, and take precedence over the default quota.

Procedure

1. Log in to the web interface.
2. Click Cloudera AI Workbenches, then open the workbench for which you want to set quotas.
3. Click AdminQuotas.
4. Switch the Default Quotas toggle to ON.

   This applies a default quota of 2 vCPU and 8 GB memory to each user in the workbench.

   If your workbench was provisioned with GPUs, a default quota of 0 GPU per user applies. If you want users to have access to GPUs, you must modify the default quotas as described in the next step.
5. If you want to change the default quotas, click on Default (per user) .

   Cloudera AI displays the Edit default quota dialog box.
6. Enter the CPU, Memory, and GPU quota values that should apply to all users of the workbench.
7. Click Update.
8. To add a custom quota for a specific user, click Add User.
9. Enter the user name, and enter the quotas for CPU, Memory, and GPU.
10. Click Add.

Results

Enabling and modifying quotas will only affect new workloads. If users have already scheduled workloads that exceed the new quota limits, those will continue to run uninterrupted. If a user is over their limit, they will not be able to schedule any more workloads.

What to do next

To specify the maximum number of replicas in a model deployment, go to Site Administration Settings Model Deployment Settings . The default is 9 replicas, and up to 199 can be set.

Creating Resource profiles

Resource profiles define how many vCPUs and how much memory the product will reserve for a particular workload (for example, session, job, model).

About this task

As a site administrator you can create several different vCPU, GPU, and memory configurations which will be available when launching a session/job. When launching a new session, users will be able to select one of the available resource profiles depending on their project's requirements.

Procedure

1. To create resource profiles, go to the Site Administration Runtime/Engine page.
2. Add a new profile under Resource Profiles.

   Cloudera recommends that all profiles include at least 2 GB of RAM to avoid out of memory errors for common user operations.

   You will see the option to add GPUs to the resource profiles only if your Cloudera AI hosts are equipped with GPUs, and you have enabled them for use by setting the relevant properties in cdsw.conf.

Results

If there are two worker nodes and 10 vCPU available overall, if one user tries to establish a session with 8 vCPU, CDSW will not allow it. The memory and CPU must be contiguous (adjacent to each other). When a user spins a

session, the pod triggers on a single node and resources on the same node are utilized. This is expected behavior for Kubernetes.

Figure 1: Resource profiles available when launching a session

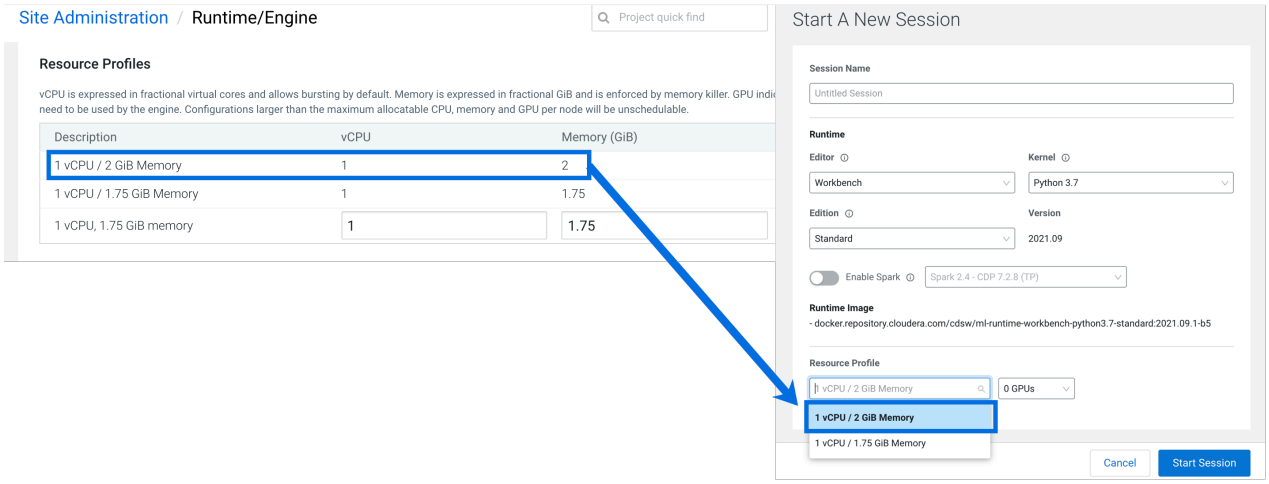

Disable or deprecate Runtime addons

Disable or deprecate a Spark Runtime addon.

About this task

You can disable or deprecate any Spark Runtime addon from the Runtime/Engine tab of Site Administration.

Procedure

1. Select Site Administration in the left Navigation bar.
2. Select the Runtime/Engine tab.

3. Select Disabled or Deprecated from Actions next to any *SPARK* addon.

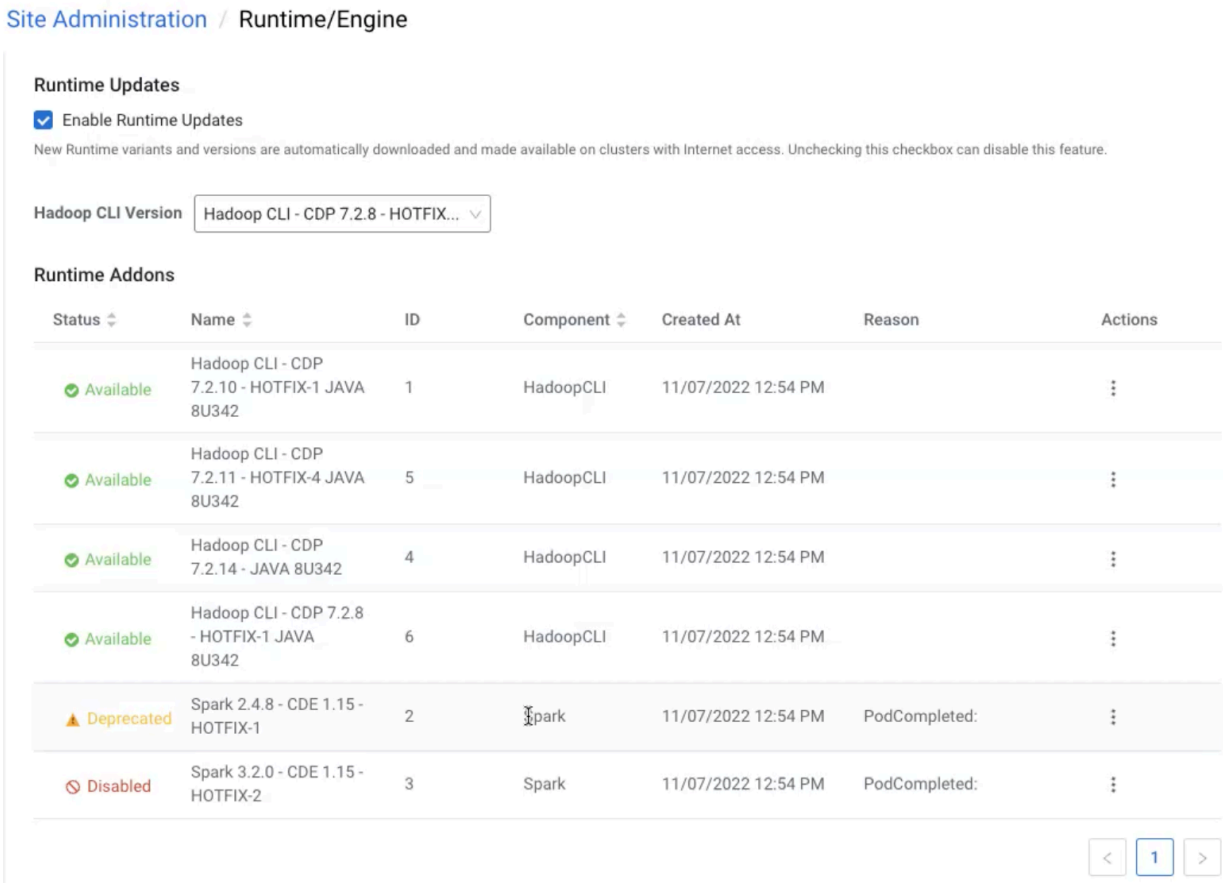

Note: You can also return the status to Available using Actions.

Onboarding Business Users

There are two procedures required for adding Business Users to Cloudera AI. First, an Administrator ensures the Business User has the correct permissions, and second, a Project Owner adds the Business User as a Collaborator.

Before you begin

Make sure the user is already assigned in your external identity provider, such as LDAP.

About this task

The Admin user performs these steps:

Procedure

1. In Environments, select the correct environment where the Cloudera AI Workbench is hosted.
2. In Manage Access, search for the user, and add the ML Business User role. Make sure the user does not already have a higher-level permission, such as ML Admin or ML User, either through a direct role assignment or a group membership.
3. Click Update Roles.

4. Inside the Cloudera AI Workbench, go to Site Administration > Users , and click Synchronize Users. This adds the necessary Users defined at the Environment level to the workbench, and updates any role changes that have been made.

What to do next

Add the ML Business User as a Collaborator to a Project.

Related Information

Adding a Collaborator

Adding a Collaborator

Project Owners can add Collaborators to a project.

About this task

The Project Owner performs these steps:

Procedure

1. Go to Collaborators, and enter the user id in the Search box.
2. Choose the User ID, and click Add. The user or team is added with their role displayed.

Results

Now, when the Business User logs in, they are able to access the Applications under this project.

User Roles

Users in Cloudera AI are assigned one or more of the following roles.

Important: Cloudera Public Cloud allows customers to maintain full ownership and control of their data and workloads and is designed to operate in some of the most restricted Public Cloud environments. Since Cloudera Public Cloud runs in a customer's cloud account, Security and Compliance is a shared responsibility between Cloudera and its Public Cloud customers. User roles form the first layer of security for securing the Cloudera AI workloads.

It is your responsibility to diligently allocate the permissions to the users. For more information, see *Cloudera's Shared Responsibility Model*.

There are two categories of roles: environment resource roles, which apply to a given Cloudera Data Platform environment, and workbench resource roles, which apply to a single workbench. To use workbench resource roles, you may need to upgrade the workbench or create a new workbench.

If a user has more than one role, then the role with the highest level of permissions takes precedence. If a user is a member of a group, it may gain additional roles through that membership.

Environment resource roles

- MLAdmin: Grants a Cloudera Data Platform user the ability to create and delete Cloudera AI Workbenches within a given Cloudera Data Platform environment. MLAdmins also have Administrator level access to all the workbenches provisioned within this environment. They can run workloads, monitor, and manage all user activity on these workbenches. They can also add the MLUser and MLBusinessUser roles to their assigned environment. This user also needs the account-level role of IAMViewer, in order to access the environment Manage Access page. To create or delete workbenches, this user also needs the EnvironmentAdmin role.

- MLUser: Grants a Cloudera Data Platform user the ability to view Cloudera AI Workbenches provisioned within a given Cloudera Data Platform environment. MLUsers are also able to run workloads on all the workbenches provisioned within this environment.
- MLBusinessUser: Grants permission to list Cloudera AI Workbench for a given Cloudera Data Platform environment. MLBusinessUsers are able to only view applications deployed under the projects that they have been added to as a Business User.

Workbench resource roles

Workbench roles are for users who are granted access to only a single workbench.

- MLWorkspaceAdmin: Grants permission to manage all Cloudera AI workloads and settings inside a specific workbench. To perform resource role assignment, the IAMViewer role is also needed. A user with this role can administer the workbench, but is not able to utilize Cloudera Data Platform APIs that modify a workbench (for example, creating or upgrading workbenches).
- MLWorkspaceBusinessUser: Grants permission to view shared Cloudera AI applications inside a specific workbench.
- MLWorkspaceUser: Grants permission to run Cloudera AI workloads inside a specific workbench.

Using the workbench resource roles

A power user or account administrator must assign the first MLWorkspaceAdmin to a workbench. Subsequently, if the MLWorkspaceAdmin also has the IAMViewer role, they can assign resource roles to the workbench.

An MLAdmin (an environment resource role) is not automatically able assign workbench resource roles on the Manage access page. A role such as MLWorkspaceAdmin is needed to do this.

You can check the permissions for a given resource role by clicking the Information icon by each resource role shown in User Management, on the Resources tab for a user, or in a Cloudera Data Platform user profile.

Note: Any user that lists users or assigns resource roles also needs the account-level role of IAMViewer.

Business Users and Cloudera AI

A user is treated as a Business User inside of Cloudera AI if they are granted the MLBusinessUser role on the Environment of the given Cloudera AI Workbench. Inside the workbench, a Business User is able to access and view applications, but does not have privileges to access any other workloads in the workbench.

Logging in as a Business User

When you log in as a Business User, the only page you see is the Applications page. The page shows any applications associated with any projects that you have been added to as a Collaborator, even though you do not have rights to access the other assets associated with those projects.

In order for applications to appear in your view, contact the Project Owner to add you as a Collaborator to the project. If you have not been added to any projects, or none of the projects that you have been added to have applications, the Applications page displays the message, You currently don’t have any applications.

Managing your Personal Account

You can edit personal account settings such as email, SSH keys and Hadoop credentials.

About this task

You can also access your personal account settings by clicking Account settings in the upper right-hand corner drop-down menu. This option will always take you to your personal settings page, irrespective of the context you are currently in.

Procedure

1. Sign in to Cloudera AI.
2. From the upper right drop-down menu, switch context to your personal account.
3. Click Settings.

   Profile

   You can modify your name, email, and bio on this page.

   Teams

   This page lists the teams you are a part of and the role assigned to you for each team.

   SSH Keys

   Your public SSH key resides here. SSH keys provide a useful way to access to external resources such as databases or remote Git repositories. For instructions, see *SSH Keys*.

Related Information

SSH Keys

Creating a Team

Users who work together on more than one project and want to facilitate collaboration can create a Team. Teams enable you to efficiently manage the users assigned to projects.

About this task

Team projects are owned by the team, rather than an individual user. Team administrators can add or remove members at any time, assigning each member different permissions. A team cannot be deleted and at least one member must be there in the team.

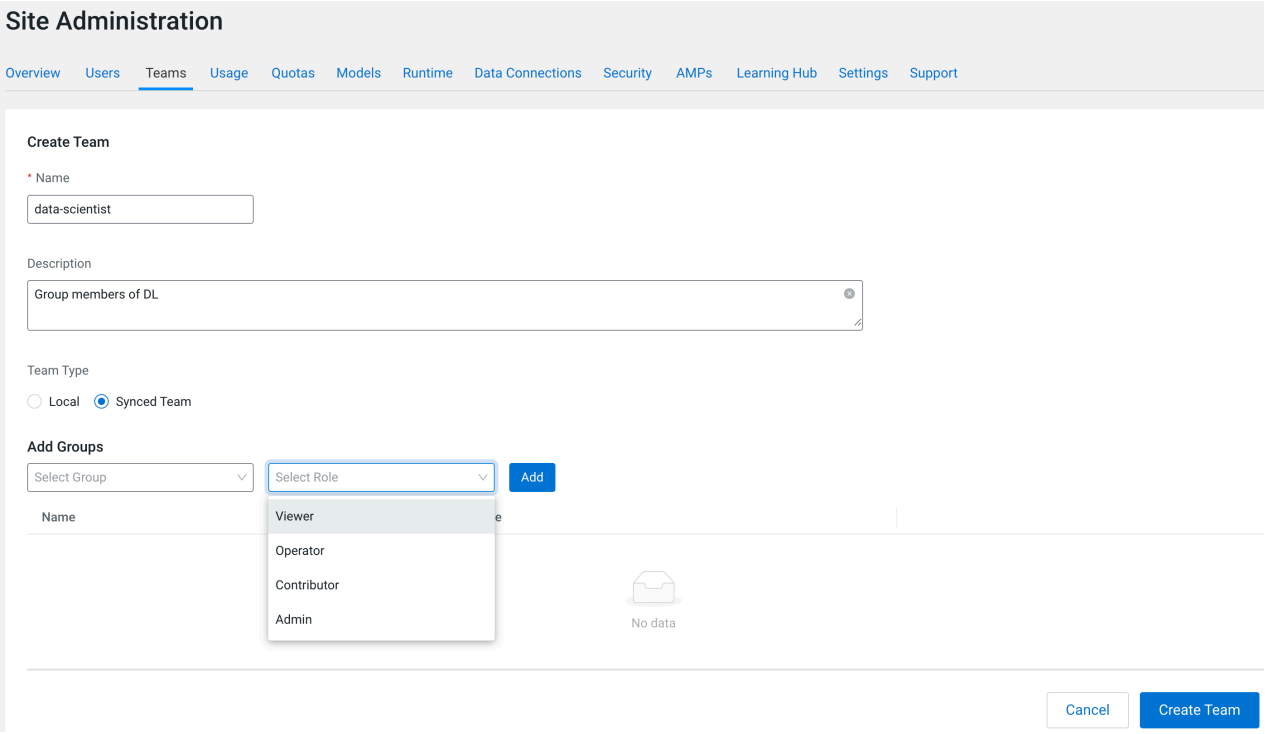

Procedure

1. In Site Administration Teams , select New Team.

2. Enter the name of the team.
3. Select Local or Synced Team.

   Cloudera Data Platform manages the member data of a Synced Team. The members and information about the members of a Local team is not managed by Cloudera Data Platform.
4. If Synced Team is selected, under Add Groups, select a group name and the role for the group and click Add. You can add multiple groups and roles using the Add option.

   Note: By default, each member will inherit the role of the groups they belong to. If a member belongs to multiple groups, their effective role in the team is the highest role assigned to the member (Viewer < Operator < Contributor < Admin).
5. Enter a Description, if needed.
6. Add or invite team members. Team members can have one of the following privilege levels:
   - Viewer - The Viewer has read-only access to team projects. The Viewere cannot create new projects within the team but can be added to existing ones.
   - Operator - The Operator has read-only access to team projects. Additionally, Operators can start and stop existing jobs in the projects that they have access to.
   - Contributor - The Contributor has write-level access to all team projects to all team projects with Team or Public visibility. The Contributor can create new projects within the team. They can also be added to existing team projects.
   - Admin - The Administrator has complete access to all team projects, can add new team members, and modify team account information. The creator of the team is assigned the Administrator privilege, and can also assign other team members the Administrator privilege. Each team must have at least one Administrator user.
7. Select Create Team.
8. Select Sync Teams to update the teams with information in the Management Console.

Managing a Team Account

Team administrators can modify account information, add or invite new team members, and view/edit privileges of existing members.

Procedure

1. From the upper right drop-down menu, switch context to the team account.
2. Click Settings to open up the Account Settings dashboard.
3. Modify any of the following settings:

   Profile

   Modify the team description on this page.

   Members

   You can add new team members on this page, and modify privilege levels for existing members.

   SSH Keys

   The team's public SSH key resides here. Team SSH keys provide a useful way to give an entire team access to external resources such as databases. For instructions, see *SSH Keys*. Generally, team SSH keys should not be used to authenticate against Git repositories. Use your personal key instead.

Related Information

SSH Keys

Managing a Synced Team

Team administrators and Site administrators can view members of a group, delete a group within a team, update roles for a group within a team, and update a custom role for a member within a group.

Viewing members of a group

You can view the members of a group along with their roles for a particular group within a team.

1. In the Cloudera Data Platform console, click the Cloudera AI tile.

   The **Home** page displays.
2. Click **Site Administration** in the left navigation menu.
3. Click Teams tab.
4. In the Teams page, click on the group name to view the members' information.
5. In the Groups tab, all the groups and their role is displayed.
6. Click the Members tab to view the list of members, their highest role, and all the groups that they belong to.

   The Inherit label is displayed if the role is inherited as part of the group. The Custom Role Set label is displayed if a custom role is assigned to the user.

Adding a custom role for a member

By default, members inherit the role of the group. You can set a custom role for a specific member within a group.

1. In the Cloudera Data Platform console, click the Cloudera AI tile.

   The **Home** page displays.
2. Click **Site Administration** in the left navigation menu.
3. Click Teams tab.
4. In the Teams page, click on the group name.
5. Click the Members tab.

   List of the members, their highest role, and all the groups that they belong to is displayed.
6. Under Role, select the role you want to assign to the member from the drop-down list.

   The Custom Role Set label is displayed after a custom role is assigned to the user.

Note: When you change the role of the group, the custom role of the member does not change. You must change the custom role to the Inherit role from the Role drop-down list for the member to inherit the group's role.

Updating the role of a group within a team

You can update the role for a particular group within a team.

1. In the Cloudera Data Platform console, click the Cloudera AI tile.

   The **Home** page displays.
2. Click **Site Administration** in the left navigation menu.
3. Click Teams tab.
4. In the Teams page, click on the group name.
5. In the Groups tab, under Role, select the role you want to assign to the group from the drop-down list.

The new role will be implemented for all members who inherit their roles within the group. Members with custom roles will remain unaffected.

Monitoring Cloudera AI Activity

This topic describes how to monitor user activity on an Cloudera AI Workbench.

Required Role: Site Administrator

The Admin Overview tab displays basic information about your deployment, such as the number of users signed up, the number of teams and projects created, memory used, and some average job scheduling and run times. You can also see the version of Cloudera AI you are currently running.

The Admin Activity tab of the dashboard displays the following time series charts. These graphs should help site administrators identify basic usage patterns, understand how cluster resources are being utilized over time, and how they are being distributed among teams and users.

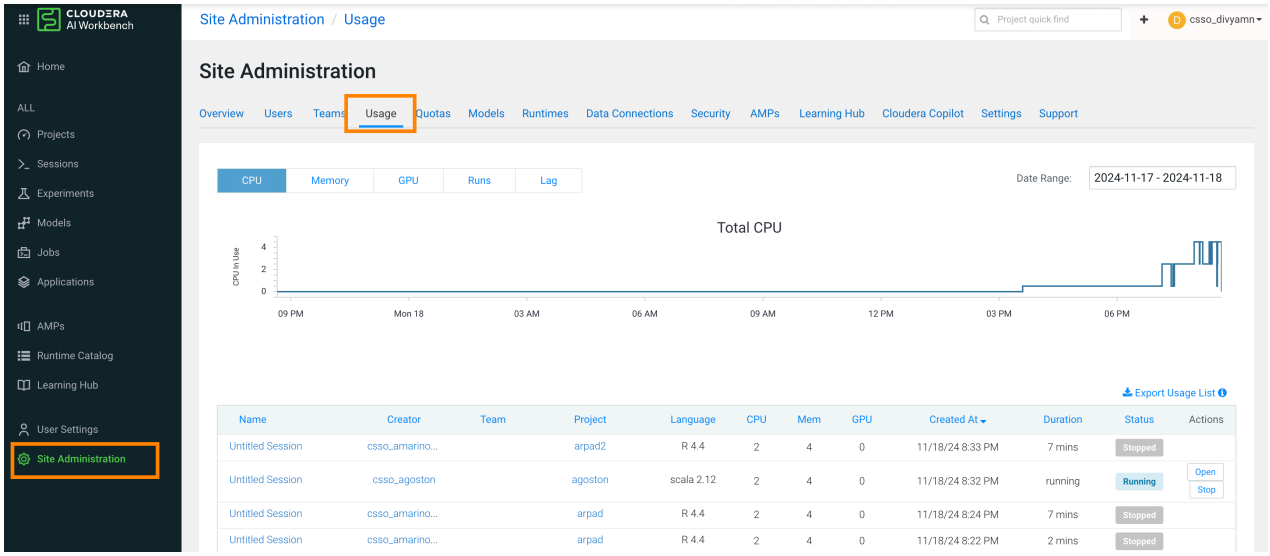

Important: The graphs and numbers on the Admin Activity page do not account for any resources used by active models on the deployment. For that information, go to Admin Models page.

- CPU - Total number of CPUs requested by sessions running at this time.

  Note that code running inside an n-CPU session, job, experiment or model replica can access at least n CPUs worth of CPU time. Each user pod can utilize all of its host's CPU resources except the amount requested by other user workloads or Cloudera AI application components. For example, a 1-core Python session can use more than 1 core if other cores have not been requested by other user workloads or Cloudera AI application components.
- Memory - Total memory (in GiB) requested by sessions running at this time.
- GPU - Total number of GPUs requested by sessions running at this time.
- Runs - Total number of sessions and jobs running at this time.
- Lag - Depicts session scheduling and startup times.
  - Scheduling Duration: The amount of time it took for a session pod to be scheduled on the cluster.
  - Starting Duration: The amount of time it took for a session to be ready for user input. This is the amount of time since a pod was scheduled on the cluster until code could be executed.

The Export Sessions List provides a CSV export file of the columns listed in the table. It is important to note that the exported duration column is in seconds for a more detailed output.

Tracked User events

The tables on this page describe the user events that are logged by Cloudera AI.

Table 1: Database Columns

When you query the user_events table, the following information can be returned:

Information	Description
id	The ID assigned to the event.

Information	Description
user_id	The UUID of the user who triggered the event.
ipaddr	The IP address of the user or component that triggered the event. 127.0.0.1 indicates an internal component.
user agent	The user agent for this action, such as the web browser. For example: `Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36`
event_name	The event that was logged. The tables on this page list possible events.
description	This field contains the model name and ID, the user type (NORMAL or ADMIN), and the username.
created_at	The date (YYYY-MM-DD format) and time (24-hour clock) the event occurred .

Table 2: Events Related to Engines

Event	Description
engine environment vars updated	-
engine mount created	-
engine mount deleted	-
engine mount updated	-
engine profile created	-
engine profile deleted	-
engine profile updated	-

Table 3: Events Related to Experiments

Event	Description
experiment run created	-
experiment run repeated	-
experiment run cancelled	-

Table 4: Events Related to Files

Event	Description
file downloaded	-
file updated	-
file deleted	-
file copied	-
file renamed	-
file linked	The logged event indicates when a symlink is created for a file or directory.
directory uploaded	-

Table 5: Events Related to Models

Event	Description
model created	-
model deleted	-

Table 6: Events Related to Jobs

Event	Description
job created	-
job started	-
stopped all runs for job	-
job shared with user	-
job unshared with user	-
job sharing updated	The logged event indicates when the sharing status for a job is changed from one of the following options to another: • All anonymous users with the link • All authenticated users with the link • Specific users and teams

Table 7: Events Related to Licenses

Event	Description
license created	-
license deleted	-

Table 8: Events Related to Projects

Event	Description
project created	-
project updated	-
project deleted	-
collaborator added	-
collaborator removed	-
collaborator invited	-

Table 9: Events Related to Sessions

Event	Description
session launched	-
session terminated	-
session stopped	-
session shared with user	-
session unshared with user	-

Event	Description
update session sharing status	The logged event indicates when the sharing status for a session is changed from one of the following options to another: • All anonymous users with the link • All authenticated users with the link • Specific users and teams

Table 10: Events Related to Admin Settings

site config updated	The logged event indicates when a setting on the Admin Settings page is changed.

Table 11: Events Related to Teams

Event	Description
add member to team	-
delete team member	-
update team member	-

Table 12: Events Related to Users

Event	Description
forgot password	-
password reset	-
update user	If the logged event shows that a user is banned, that means that the user account has been deactivated and does not count toward the license.
user signup	-
user login	The logged event includes the authorization method, LDAP/SAML or local.
user logout	-
ldap/saml user creation	The logged event indicates when a user is created with LDAP or SAML.

Monitoring User Events

This topic shows you how to query the PostgresSQL database that is embedded within the Cloudera AI deployment to monitor or audit user events.

About this task

Querying the PostgresSQL database that is embedded within the Cloudera AI deployment requires root access to the Cloudera AI Master host.

Procedure

1. SSH to the Cloudera AI Master host and log in as root.

   For example, the following command connects to cdsw-master-host. as root:

```
ssh root@cdsw-master-host.yourcdswdomain.com
```

2. Get the name of the database pod:

```
kubectl get pods -l role=db
```

The command returns information similar to the following example:

```
NAME                   READY   STATUS    RESTARTS   AGE
db-86bbb69b54-d5q88    1/1     Running   0          4h46m
```

3. Enter the following command to log into the database as the sense user:

```
 kubectl exec <database pod> -ti -- psql -U sense
```

For example, the following command logs in to the database on pod db-86bbb69b54-d5q88:

```
kubectl exec db-86bbb69b54-d5q88 -ti -- psql -U sense
```

You are logged into the database as the sense user.

4. Run queries against the user_events table.

For example, run the following query to view the most recent user event:

```
select * from user_events order by created_at DESC LIMIT 1
```

The command returns information similar to the following:

```
id          | 3658
user_id     | 273
ipaddr      | ::ffff:127.0.0.1
user_agent  | node-superagent/2.3.0
event_name  | model created
description | {"model":"Simple Model 1559154287-ex5yn","modelId":"50","
userType":"NORMAL","username":"DonaldBatz"}
created_at  | 2019-05-29 18:24:47.65449
```

5. Optionally, you can export the user events to a CSV file for further analysis:
   a) Copy the user_events table to a CSV file:

```
copy user_events to '/tmp/user_events.csv' DELIMITER ',' CSV HEADER;
```

   b) Find the container that the database runs on:

```
docker ps | grep db-86bbb
```

   The command returns output similar to the following:

```
c56d04bbd58 c230b2f564da "docker-entrypoint..." 7 days ago Up 7 days k8s
_db_db-86bbb69b54-fcfm6_default_8b2dd23d-88b9-11e9-bc34-0245eb679f96_0
```

   The first entry is the container ID.
   c) Copy the user_events.csv file out of the container into a temporary directory on the Master host:

```
docker cp <container ID>:/tmp/user_events.csv /tmp/user_events.csv
```

   For example:

```
docker cp 8c56d04bbd58:/tmp/user_events.csv /tmp/user_events.csv
```

   d) Use SCP to copy /tmp/user_events.csv from the Cloudera AI Master host to a destination of your choice.

   For example, run the following command on your local machine to copy user_events.csv to a local directory named events:

```
scp root@cdsw-master-host.yourcdswdomain.com:/tmp/user_events.csv /local
/directory/events/
```

What to do next

For information about the different user events, see *Tracked User Events*.

Related Information

Tracked User events

Monitoring active Models across the Workbench

This topic describes how to monitor all active models currently deployed on your workbench.

What is an active Model?

A model that is in the Deploying, Deployed, or Stopping stages is referred to as an active model.

Monitoring all active Models across the Workbench

Required Role: Site Administrator

To see a complete list of all the models that have been deployed on a deployment, and review resource usage across the deployment by models alone, go to Admin Models . On this page, site administrators can also Stop/Restart/Rebuild any of the currently deployed models.

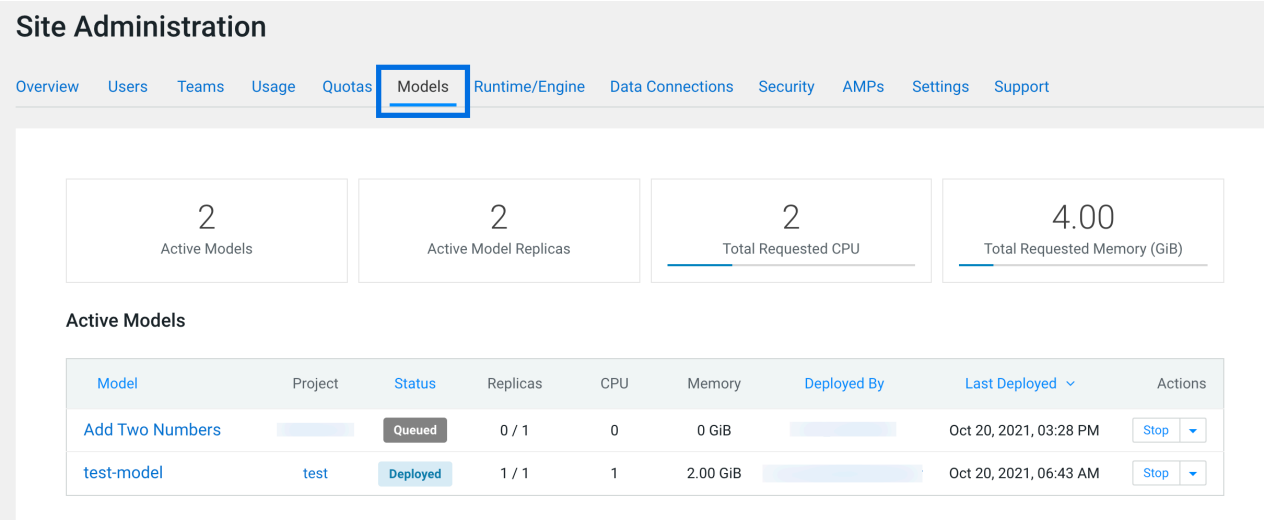

Monitoring and alerts

Cloudera AI leverages Cloudera Data Platform Monitoring based on Prometheus and Grafana to provide dashboards that allow you to monitor how CPU, memory, storage, and other resources are being consumed by your Cloudera AI Workbenches.

Prometheus is an internal data source that is auto-populated with resource consumption data for each deployment. Grafana is the monitoring dashboard that allows you to create visualizations for resource consumption data from Prometheus. By default, Cloudera AI provides three Grafana dashboards: K8 Cluster, K8s Containers, and K8s Node. You can extend these dashboards or create more panels for other metrics. For more information, see the Grafana documentation.

Related Information

Grafana documentation

Application polling endpoint

The Cloudera AI server periodically polls applications for their status. The default polling endpoint is the root endpoint (/), but a custom polling endpoint can be specified if the server or other application has difficulty with the default endpoint.

When creating or modifying an application, you can specify a new value for the CDSW_APP_POLLING_END POINT environmental variable. Just replace the default value / that is shown. For more information, see *Analytical Applications*.

You can also set the environmental value in Project Settings Advanced . In this case, any setting made here can be overridden by settings in a given application. However, settings made in Project Settings Advanced also apply when polling sessions.

Related Information

Analytical Applications

Choosing default engine

This topic describes how to choose a default engine for creating projects.

Before you begin

Required Role: MLAdmin

Note: On Private Cloud, the corresponding role is EnvironmentAdmin.

Make sure you are assigned the MLAdmin role in Cloudera Data Platform. Only users with the MLAdmin role will be logged into Cloudera AI Workbenches with Site Administrator privileges.

There are two types of default engines: and Legacy Engines. However, legacy engines are deprecated in the current release and project settings default to ML Runtime.

Legacy engines Engines contain the machinery necessary to run sessions using all four interpreter options that Cloudera AI currently supports (Python 2, Python 3, R and Scala) and other support utilities (C and Fortran compilers, LaTeX, etc.). ML Runtimes are thinner and more lightweight than legacy engines. Rather than supporting multiple programming languages in a single engine, each Runtime variant supports a single interpreter version and a subset of utilities and libraries to run the user's code in Sessions, Jobs, Experiments, Models, or Applications.

Procedure

1. Log in to the web interface.
2. Click Cloudera AI Workbenches , then open the workbench for which you want to set Default Engine.
3. Click Admin Runtime/Engine .
4. Choose the Default Engine you would like to use as the default for all newly created projects in this workbench.

   Note: Legacy Engines are deprecated in this release and Cloudera recommends using Runtime.

5. Modify the remaining information on the page:
   - Resource Profiles listed in the table are selectable resource options for both legacy Engines and ML Runtime (for example, when starting a Session or Job)
   - The remaining information on the page applies to site-level settings specific for legacy Engines.

Controlling User access to features

Cloudera AI provides Site Administrators with the ability to restrict or hide specific functionality that non-Site Administrator users have access to in the UI. For example, a site administrator can hide the models and experiments features from the Cloudera AI Workbench UI.

The settings on this page can be configured through the Security and Settings tabs on the Administration page.

Table 13: Security Tab

Property	Description
Allow remote editing	Disable this property to prevent users from connecting to the Cloudera AI deployment with cdswctl and using local IDEs, such as PyCharm.

Property	Description
Allow only session creators to run commands on active sessions	By default, a user's permission to active sessions in a project is the same as the user's permission to that project, which is determined by the combination of the user's permission as a project collaborator, the user's permission in the team if this is a team project, and whether the user is a Site Administrator. By checking this checkbox, only the user that created the active session will be able to run commands in that session. No other users, regardless of their permissions in the team or as project collaborators, will be able to run commands on active sessions that are not created by them. Even Site Administrators will not be able to run commands in other users' active sessions.
Allow console output sharing	Disable this property to remove the Share button from the project workbench and workbench UI as well as disable access to all shared console outputs across the deployment. Note that re-enabling this property does not automatically grant access to previously shared consoles. You will need to manually share each console again
Allow anonymous access to shared console outputs	Disable this property to require users to be logged in to access shared console outputs.
Allow file upload/download through UI	Use this checkbox to show/hide file upload/download UI in the project workbench. When disabled, Cloudera AI API will forbid request of downloading file(s) as attachment. Note that the backend API to upload/edit/read the project files are intact regardless of this change in order to support basic Cloudera AI functionality such as file edit/read.

Table 14: Settings Tab

Property	Description
Require invitation to sign up	Enable this property to send email invitations to users when you add them to a group. To send email, an SMTP server must first be configured in Settings Email .
Allow users to create public projects	Disable this property to restrict users from creating new public projects. Site Administrators will have to create any new public projects.
Allow Legacy Engine users to use the Python 2 kernel	Enable this property to allow Legacy Engine users to select the Python 2 kernel when creating a job. Python 2 is disabled by default.
Allow users to create projects	Disable this property to restrict users from creating new projects. Site Administrators will have to create any new projects.
Allow users to create teams	Disable this property to restrict users from creating new teams. Site Administrators will have to create any new teams.
Allow users to run experiments	Disable this property to hide the Experiments feature in the UI. Note that this property does not affect any active experiments. It will also not stop any experiments that have already been queued for execution.
Allow users to create models	Disable this property to hide the Models feature in the UI. Note that this property does not affect any active models. In particular, if you do not stop active models before hiding the Models feature, they continue to serve requests and consume computing resources in the background.
Allow users to create applications	Disable this property to hide the Applications feature in the UI. Note that this property does not affect any active applications. In particular, if you do not stop active applications before hiding the feature, they continue to serve requests and consume computing resources in the background.

Cloudera AI email notifications

Cloudera AI allows you to send email notifications when you add collaborators to a project, share a project with a colleague, and for job status updates (email recipients are configured per-job). This topic shows you how to specify email address for such outbound communications.

Note that email notifications are not currently enabled by default. Emails are not sent when you create a new project. Email preferences cannot currently be configured at an individual user level.

Option 1: If your existing corporate SMTP server is accessible from the VPC where your Cloudera AI Workbench is running, you can continue to use that server. Go to the Admin Settings tab to specify an email address for outbound invitations and job notifications.

Option 2: If your existing SMTP solution cannot be used, consider using an email service provided by your cloud provider service. For example, Amazon provides Amazon Simple Email Service (Amazon SES).

Mail relay hosts often limit the authenticated sender reply address. Make sure to select a No reply email which you are allowed to use, otherwise email sending may fail.

Downloading diagnostic bundles for a workbench

This topic describes how to download diagnostic bundles for an Cloudera AI Workbench.

Before you begin

Required Role: MLAdmin

Make sure you are assigned the MLAdmin role in Cloudera Data Platform. Only users with the MLAdmin role will be logged into Cloudera AI Workbenches with Site Administrator privileges.

Procedure

1. Log in to the web interface.
2. In Cloudera AI Workbenches, select a workbench.
3. Select Site Administration Support Generate Log Archive .
4. Select the time period from the dropdown.
5. Ensure Include Engines is selected if engine logs are needed (included by default).
6. Select Send to Cloudera to send the diagnostic logs to Cloudera Support.
7. Select Create to generate the logs.
8. When Status is Complete, select Download to download the diagnostics bundles to your machine.

What to do next

The data in the contained bundles may be incomplete. If it does not contain logs for time period you are looking for, there are a number of possible reasons:

- There is a delay between the time the logs are initially generated by a workload and the time they are visible in cloud storage. This may be approximately 1 minute due to buffering during streaming, but can be significantly longer due to eventual consistency in the cloud storage.
- Another user or process may have deleted data from your bucket; this is beyond the control of Cloudera AI.
- There may be a misconfiguration or an invalid parameter in your request. Retrieving logs requires a valid cloud storage location to be configured for logging, as well as authentication for Cloudera AI to be set up properly for it. Requests must pertain to a valid engine in a valid project.

Web session timeouts

You can set web sessions to time out and require the user to log in again. This time limit is not based on activity, it is the maximum time allowed for a web session.

You can set timeout limits for Users and Admin Users in Site AdministrationSecurity.

- User Web Browser Timeout (minutes) - This timeout sets the default maximum length of time that a web browser session can remain inactive. You remain logged in if you are actively using the session. If you are not active, then after a 5-minute warning, you are automatically logged out. Any changes to the setting take effect for any subsequent user logins.
- Admin User Web Browser Timeout (minutes) - This timeout sets the default maximum length of time that a web browser session for an Admin user can remain inactive. You remain logged in if you are actively using the session. If you are not active, then after a 5-minute warning, you are automatically logged out. Any changes to the setting take effect for any subsequent Admin user logins.

Project garbage collection

Marks orpaned files for deletion from a project and cleans up projects that are marked for deletion.

Procedure

1. Click Site Administration Settings .
2. Scroll to Project Garbage Collection.

   Click Garbage Collect Projects to permanently delete projects marked for deletion.

   Click Clean Up Orphaned Projects to mark orphaned projects for deletion.

Results

Orphaned project files are marked for deletion. All files marked for deletion are permanently deleted when you click Garbage Collect Projects.

Ephemeral storage

Ephemeral storage space is scratch space a Cloudera AI session, job, application or model can use. This feature helps in better scheduling of Cloudera AI pods, and provides a safety valve to ensure runaway computations do not consume all available scratch space on the node.

By default, each user pod in Cloudera AI is allocated 0 GB of scratch space, and it is allowed to use up to 10 GB. These settings can be applied to an entire site, or on a per-project basis.

Change site-wide ephemeral storage configuration

In Site Administration Settings Advanced , you can see the fields to change the ephemeral storage request (minimum) and maximum limit.

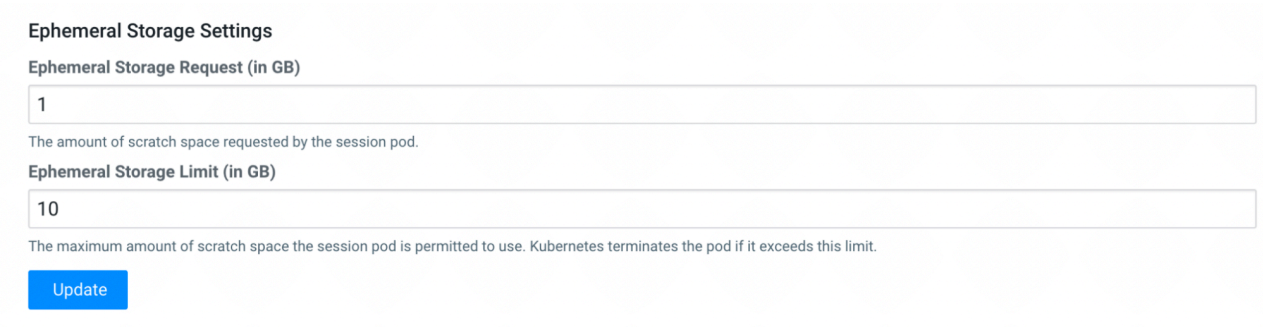

Override Site-wide ephemeral storage configuration

If you want to customize the ephemeral storage settings, you can do so on a per-project basis. Open your project, then click on Project Settings Advanced and adjust the ephemeral storage parameters.

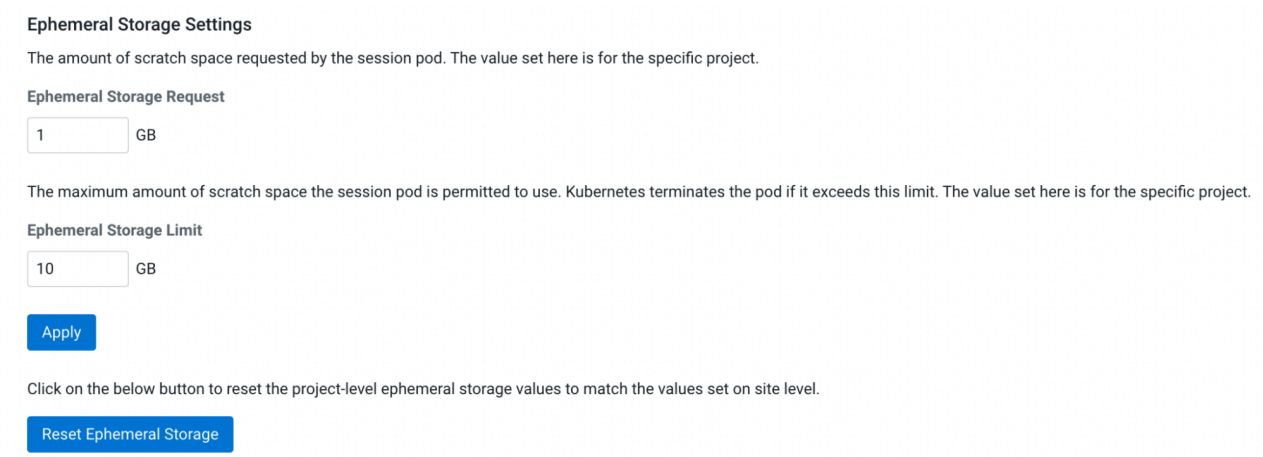

AWS Known Issues

There is a known issue with the cluster autoscaler that affects autoscaling from 0->1 if a non-zero value for Ephemeral Storage Request is set. This affects both CPU and GPU node groups of the Cloudera AI Workbench. The autoscaler throws the following error when this happens:

```
pod didn't trigger scale-up: 1 Insufficient ephemeral-storage
```

This is occurring even though the nodes in the Cloudera AI autoscaling groups have sufficient ephemeral storage space in their group templates. See this github issue for details. Even though the issue is closed, the problem still persists.

The issue only affects node groups that have [0, x] autoscaling range.

Set the Ephemeral Storage Request value to 0 in both the site-wide and project settings if you run into this issue.

Installing a non-transparent proxy in a Cloudera AI environment

If Cloudera AI is used in an air-gapped environment, a proxy configuration is not mandatory. If a non-transparent proxy is used, then certain endpoints must be added to the allowed list for the proxy.

If your Cloudera Data Platform Private Cloud deployment uses a non-transparent network proxy, configure proxy hosts that the workloads can use for connections with Cloudera AI Workbenches. You can configure the proxy configuration values from the Management Console.

Note: The settings configured using this procedure reflect in newly provisioned Cloudera AI Workbenches in a CDP Private Cloud Experiences deployment using the Experiences Compute Service (ECS). In an OpenShift deployment, the default values are used.

1. Sign in to the CDP console.
2. Click Management Console.
3. On the Management Console home page, select Administration Networks to view the Networks page.
4. Configure the following options for the proxy values:

Field	Description
HTTPS Proxy	The HTTP or HTTPS proxy connection string for use in connections with Cloudera AI Workbenches. You must specify this connection string in the form: http(s): //<username>:<password>@<host>:<port>. **Note:** The <username> and <password> parameters are optional. You can specify the connection proxy string without these parameters.
HTTP Proxy	The HTTP or HTTPS proxy connection string for use in connections with Cloudera AI Workbenches. You must specify this connection string in the form: http(s): //<username>:<password>@<host>:<port>. **Note:** The <username> and <password> parameters are optional. You can specify the connection proxy string without these parameters.
No Proxy	Comma-separated list of hostnames, IP addresses, or hostnames and IP addresses that should not be accessed through the specified HTTPS or HTTP proxy URLs. In case of ECS deployments, you must include no-proxy URLs for the following: • All the ECS hosts in your deployment • Any CDP Private Cloud Base cluster that you want to access • CIDR IP addresses for internal operations in the ECS cluster: 10.42.0.0/16 and 10.43.0.0/16

5. Click Save
6. Ensure that the following endpoint is allowed:

Description/ Usage	CDP service	Destination	Protocol and Authentication	IP Protocol/ Port	Comments
Accelerators for ML Projects (AMPs)	Machine Learning	https://raw.githubusercontent.com	HTTPS	TCP/443	Files for AMPs are hosted on GitHub.

Description/ Usage	CDP service	Destination	Protocol and Authentication	IP Protocol/ Port	Comments
		https://github.com			

Ports used by Cloudera AI

Certain ports must be accessible through the firewall for proper operation of Cloudera AI.

Firewall restrictions must be disabled across Cloudera AI and Cloudera Data Platform cluster hosts. Internally, the Cloudera AI master and worker hosts require full connectivity with no firewalls. Externally, end users connect to Cloudera AI exclusively through a web server running on the master host, and therefore do not need direct access to any other internal Cloudera AI or Cloudera Data Platform services.

Communication with the Cloudera Data Platform cluster	Cloudera AI -> Cloudera Data Platform As a gateway service, Cloudera AI must have access to all the ports used by Cloudera Data Platform and Cloudera Manager.
Communication with the Web Browser	The Cloudera AI web application is available at port 80. HTTPS access is available over port 443.

Export Usage List

You can export a list of sessions, jobs, workers, and experiments. You can either download a complete list of workloads or you can filter the workloads by date to download a more concise list. Timestamps in the list are given in Coordinated Universal Time (UTC).

Procedure

1. Select Site Administration in the navigation pane.
2. Select the Usage tab.
3. If you want a list of workloads specific to a date range, you can filter the list of workloads by setting the Date Range.
4. Select Export Usage List to download the list of workloads.

Results

The list downloads to your computer as a .csv file.

Private cluster support

Each type of network architecture supported by Cloudera Data Platform has a unique set of tradeoffs among ease of setup, security, workloads (Experiences) supported, and so on.

Private Cloud Clusters provide a simple way to create a secure cluster, where the API server and the workloads themselves only rely on private IP addresses that are not accessible from the internet. Connectivity to the cluster from the Cloudera Data Platform control plane is provided by the Cluster Connectivity Manager v2 (CCM v2). CCMv2 uses an agent running in the cluster, and an inverting proxy running on Cloudera Data Platform, which creates a HTTPS tunnel between the workload and the control plane.

Note: Private Cloud clusters are Generally Available (GA) on AWS, but are in Preview on Azure, and require an entitlement.

Enable a private cluster

To enable a private cluster, select the option when provisioning the workbench.

Procedure

1. In Cloudera AI Workbenches, select Provision Workbench.
2. Enter a Workbench Name, and select Environment.
3. Select the Advanced Options toggle.
4. In Network Settings, select Enable Fully Private Cluster.
5. Make any other settings needed, and select Provision Workbench.

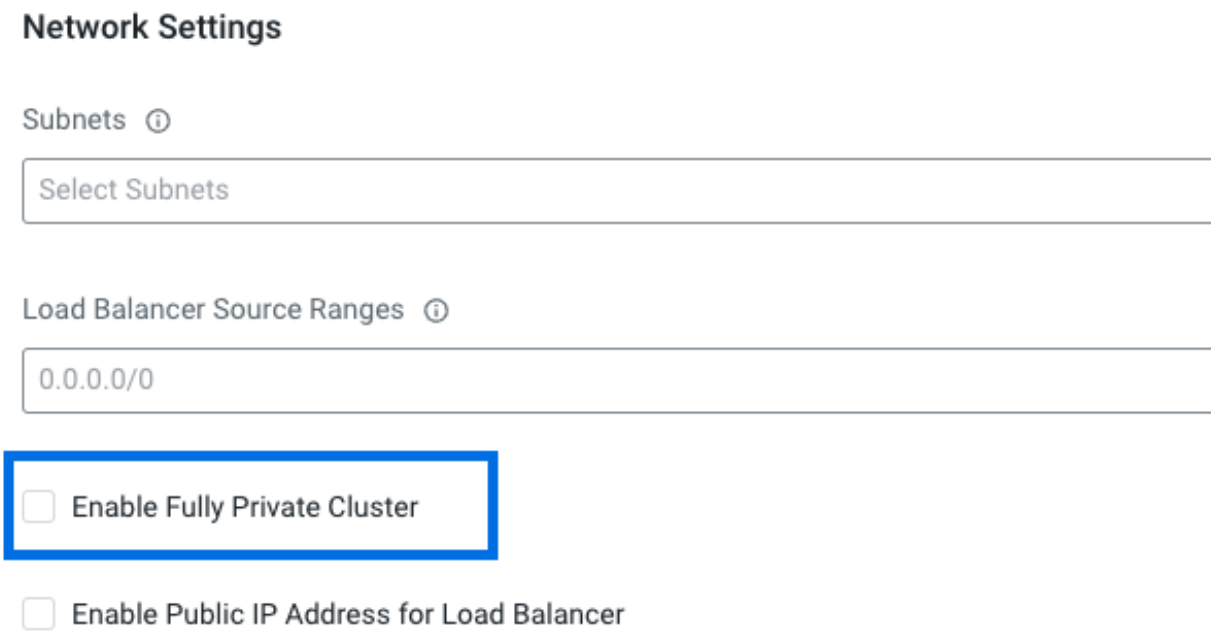

Results

The workbench is provisioned using a fully private cluster.

What to do next

User Defined Routing (UDR)

With the Fully Private Cluster configuration, Azure still creates some public IP resources to support load balancer egress. If necessary, you can avoid creating public IP addresses in the Cloudera AI cluster by using a User Defined Routing (UDR) table. A UDR table can be configured in the cluster subnet to route packets to a customer-configured firewall, for example to limit internet access or analyze traffic. For more information on setting up UDR, see the Microsoft articles Virtual appliance scenario or Virtual network traffic routing.

About this task

To utilize a UDR and firewall in the Azure Cloudera AI Private Cloud cluster, select the following when setting up the cluster.

Procedure

1. Select a subnet with a default route configuration to forward the traffic to the network appliance or firewall.

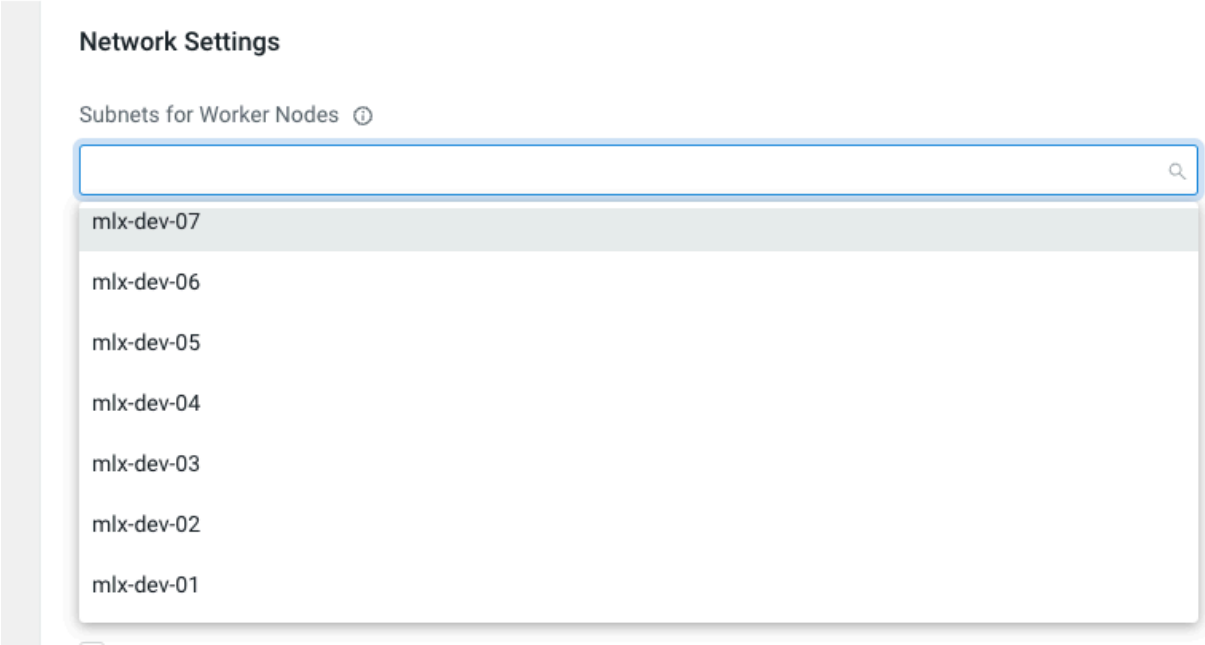

2. Create load balancers with private IP addresses. This is the default choice when creating clusters in Cloudera AI.

3. Select Enable User Defined Routing.

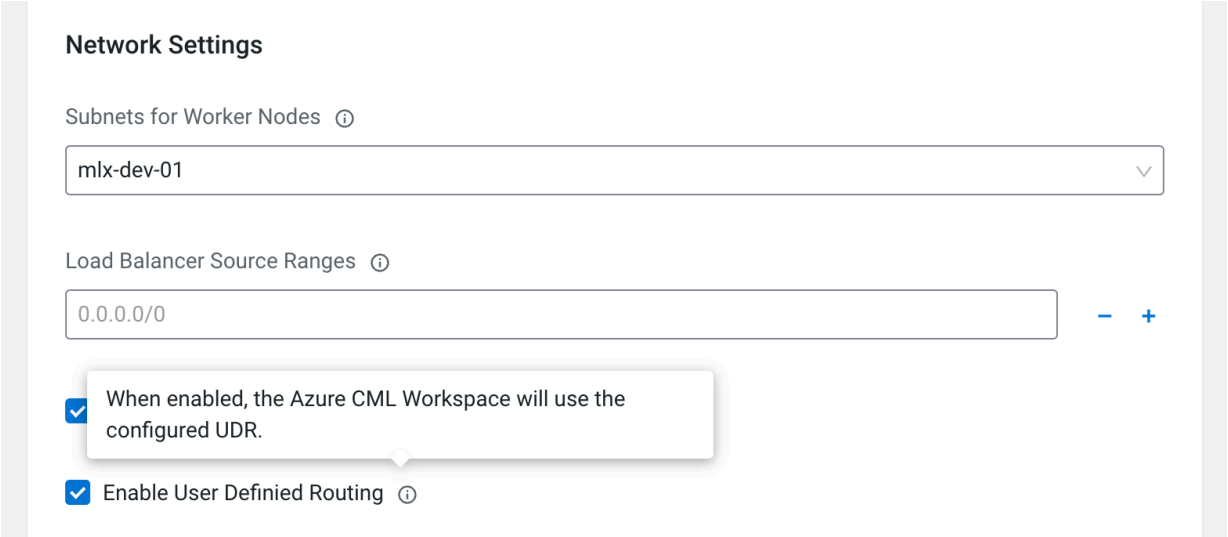

Embed a Cloudera AI application in an external website

You can embed a Cloudera AI application into an I-frame on a web page. You need to specify the frame-ancestor attribute, otherwise the browser security policy will prevent the application from rendering in the page.

The frame-ancestor attribute prevents "Clickjacking" attacks by specifying which domains are allowed to provide embedded content to your site. To enable a domain to embed a Cloudera AI application, set the environmental variable CDSW_FRAME_ANCESTORS to cointain one or more websites as follows:

- The name of the embedding website, specified in host-source form.
- You can specify multiple websites as a comma-separated list.

You can set the environmental value in Project Settings Advanced .

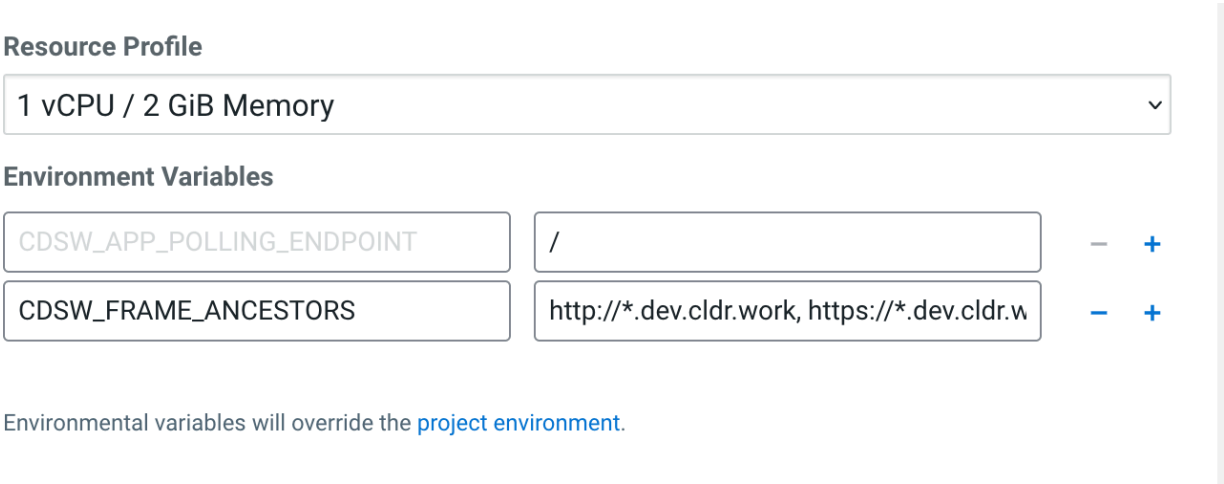

For more information on host-source form, see: *CSP: frame-ancestors*.

Related Information

CSP: frame-ancestors

Setting up Cloudera AI Workbenches for high volume Workloads

Autoscaling in Cloudera AI enables seamless scaling up of clusters, accommodating sessions, experiments, model metrics, jobs, and applications with increased user demand. In order to ensure seamless functioning of Cloudera AI Workbenches with high volume workloads, core Cloudera AI flows such as workbench suspend, resume, backup, upgrade, and editing are validated with high volume workloads. Additionally, clusters can be efficiently downsized by adjusting the autoscale range in workloads without disrupting control plane capabilities.

To prepare a Cloudera AI Workbench for high volume workloads, you need to perform the following steps to modify the pod quota limit for the Cloudera AI tenant, and then do the same for the Cloudera AI Workbench.

1. Go inside the workbench and create a project.

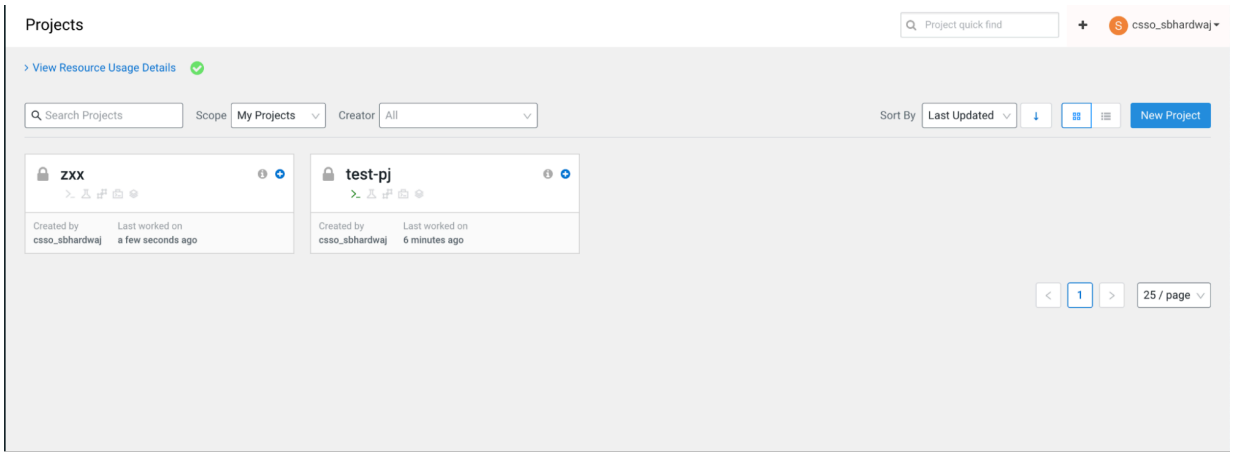

2. In Project settings, click the lock button.

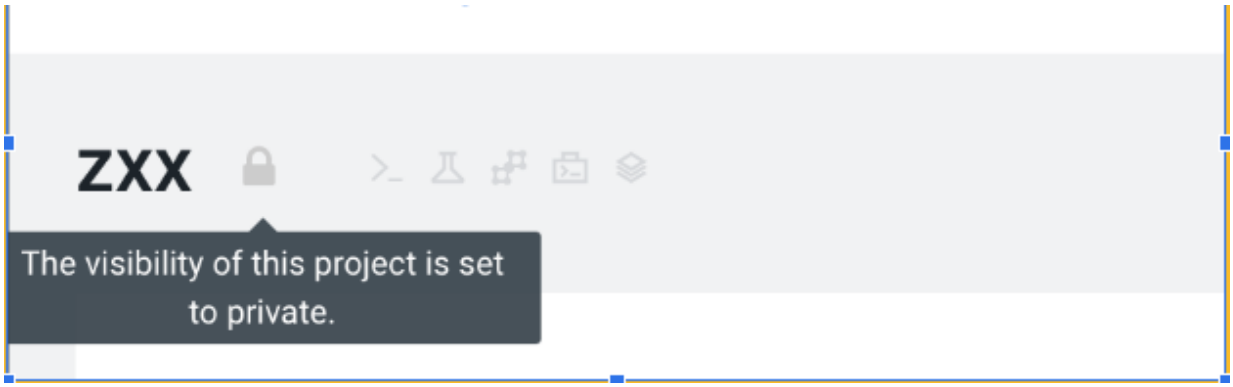

3. In Advanced settings, add the key OVERRIDE_PODQUOTA and enter the value for the new pod limit to set.

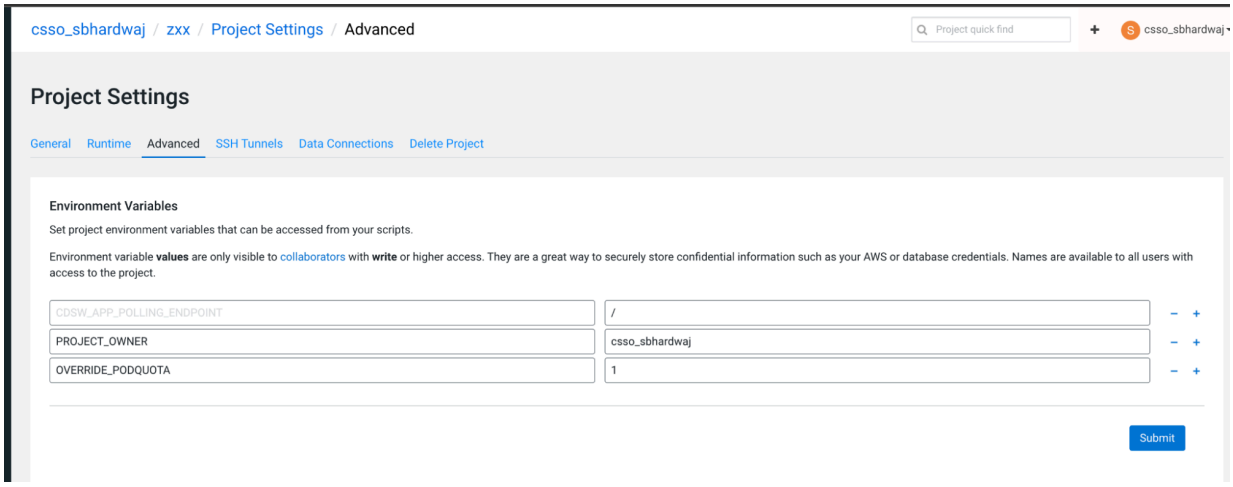

Verified flows and configurations

1. Enhanced Pod Count Limit

   Scaling up to 250 Pods within a single user namespace is verified, assuming that the workbench possesses sufficient computational resources and storage capacity to accommodate these pods.
2. Auto-Scaling Certification

   The Control Plane actions have been verified to scale up to 100 Nodes.
3. Suspension and resumption of 100 Nodes has been successfully tested.
4. Workbench Management
   - Backup of workbenches with up to 100 Nodes.
   - Upscale and downscale of cluster nodes between 1-100 is now verified by changing the worker node range within the edit section of workbench.
   - Successfully tested upgrading a 100 node Workbench from version at 'V-1'.
5. Basic sanity testing of the following workload flows was conducted for workbenchess with 100 Nodes:
   - Creation of Sessions, Applications, Jobs, Model Metrics and Experiments for a user after workbench creation and upgrading.
   - Deletion of Sessions, Applications, Jobs, Model Metrics and Experiments for high volume workloads.
   - I/O, memory usage validation during upscale and downscale operations.

Known issues

There are a few known issues that have been identified during the certification process. These issues are currently being addressed.

1. During concurrent creation of sessions via Cloudera AI Workbench users, some pods may fail to come up, with a ~3% failure rate.
2. For high volume workbenches, Cloudera AI Workbench applications may fail to restart after modify/resume and upgrade operations. In this case, manually restart the affected application.

Host name required by Learning Hub

Learning Hub requires internet access to link to the displayed content. Learning Hub cannot be supported on a fully airgapped cluster.

The following domain must be added to the allow list so that links from the content will work:

- *.raw.githubusercontent.com

Cloudera AI

Using Apache Spark 2 with Cloudera AI

Date published: 2020-07-16
Date modified: 2024-11-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Spark 2 and Spark 3 on Cloudera AI 4

Apache Spark supported versions 5

Spark configuration files 5

Managing memory available for Spark drivers 5

Managing dependencies for Spark 2 jobs 5

Spark Log4j Configuration 6

Setting up an HTTP proxy for Spark 2 7

Spark web UIs 7

Using Spark 2 from Python 7

Example: Monte Carlo estimation 8

Example: Locating and adding JARs to Spark 2 configuration 9

Using Spark 3 from R 10

Using Spark 2 from Scala 10

Managing dependencies for Spark 2 and Scala 11

Apache Spark 2 and Spark 3 on Cloudera AI

Apache Spark is a general purpose framework for distributed computing that offers high performance for both batch and stream processing. It exposes APIs for Java, Python, R, and Scala, as well as an interactive shell for you to run jobs.

In Cloudera AI, Spark and its dependencies are bundled directly into the Cloudera AI engine Docker image.

Cloudera AI supports fully-containerized execution of Spark workloads via Spark's support for the Kubernetes cluster backend. Users can interact with Spark both interactively and in batch mode.

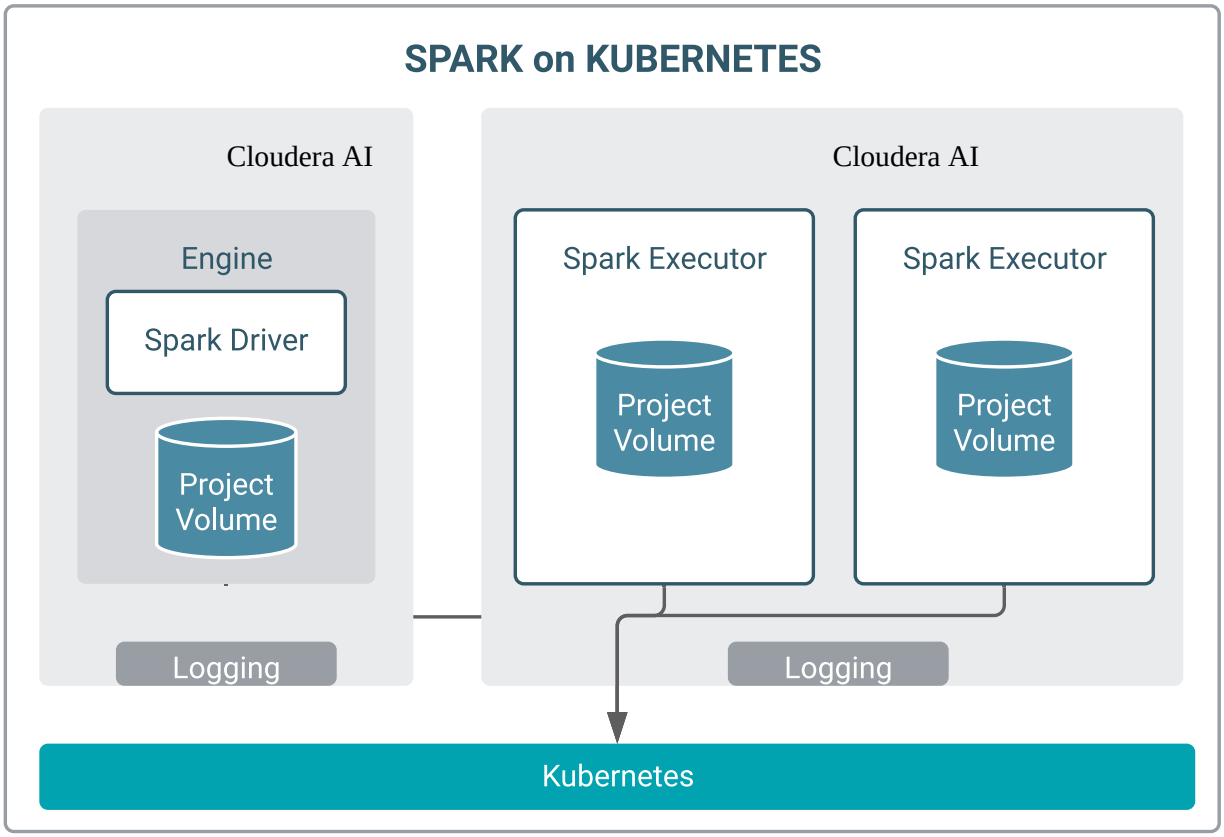

Dependency Management: In both batch and interactive modes, dependency management, including for Spark executors, is transparently managed by Cloudera AI and Kubernetes. No extra required configuration is required. In interactive mode, Cloudera AI leverages your cloud provider for scalable project storage, and in batch mode, Cloudera AI manages dependencies though container images.

Autoscaling: Cloudera AI also supports native cloud autoscaling via Kubernetes. When clusters do not have the required capacity to run workloads, they can automatically scale up additional nodes. Administrators can configure auto-scaling upper limits, which determine how large a compute cluster can grow. Since compute costs increase as cluster size increases, having a way to configure upper limits gives administrators a method to stay within a budget. Autoscaling policies can also account for heterogeneous node types such as GPU nodes.

Dynamic Resource Allocation: If a Spark job requires increasing memory or CPU resources as it executes a job, Spark can automatically increase the allocation of these resources. Likewise, the resources are automatically returned to the cluster when they are no longer needed. This mechanism is especially useful when multiple applications are sharing the resources of a cluster.

Workload Isolation: In Cloudera AI, each project is owned by a user or team. Users can launch multiple sessions in a project. Workloads are launched within a separate Kubernetes namespace for each user, thus ensuring isolation between users at the K8s level.

Observability: Monitoring of Spark workloads, such as resources being consumed by Spark executors, can be performed using Grafana dashboards. For more information, see *Monitoring and Alerts* and *Monitoring Cloudera AI Workbenches*.

Related Information

Monitoring and Alerts

Monitoring Cloudera AI Workbenches

Apache Spark supported versions

Spark 2.4.7 and Spark 3.2.3 are available through Runtime addons that can be selected when starting a session.

Note: Spark 3 does not work with Scala runtimes.

Spark configuration files

Cloudera AI supports configuring Spark 2 and Spark 3 properties on a per project basis with the spark-defaults.conf file. If there is a file called spark-defaults.conf in your project root, this will be automatically be added to the global Spark defaults.

To specify an alternate file location, set the environmental variable, SPARK_CONFIG, to the path of the file relative to your project. If you're accustomed to submitting a Spark job with key-values pairs following a --conf flag, these can also be set in a spark-defaults.conf file instead. For a list of valid key-value pairs, refer to *Spark Configuration*.

Administrators can set environment variable paths in the /etc/spark/conf/spark-env.sh file.

Related Information

Spark Configuration

Managing memory available for Spark drivers

By default, the amount of memory allocated to Spark driver processes is set to a 0.8 fraction of the total memory allocated for the runtime container. If you want to allocate more or less memory to the Spark driver process, you can override this default by setting the spark.driver.memory property in spark-defaults.conf (as described above).

Note: The memory allocated to a Cloudera AI session does not include memory taken by Spark executors.

Managing dependencies for Spark 2 jobs

As with any Spark job, you can add external packages to the executor on startup. To add external dependencies to Spark jobs, specify the libraries you want added by using the appropriate configuration parameter in a spark-defaul ts.conf file.

The following table lists the most commonly used configuration parameters for adding dependencies and how they can be used:

Property	Description
spark.files	Comma-separated list of files to be placed in the working directory of each Spark executor.
spark.submit.pyFiles	Comma-separated list of .zip, .egg, or .py files to place on PYTHONPATH for Python applications.
spark.jars	Comma-separated list of local jars to include on the Spark driver and Spark executor classpaths.
spark.jars.packages	Comma-separated list of Maven coordinates of jars to include on the Spark driver and Spark executor classpaths. When configured, Spark will search the local Maven repo, and then Maven central and any additional remote repositories configured by spark.jars.ivy. The format for the coordinates are groupId:artifactId:version.
spark.jars.ivy	Comma-separated list of additional remote repositories to search for the coordinates given with spark.jars.packages.

Example spark-defaults.conf

Here is a sample spark-defaults.conf file that uses some of the Spark configuration parameters discussed in the previous section to add external packages on startup.

```
spark.jars.packages org.scalaj:scalaj-http_2.11:2.3.0
spark.jars my_sample.jar
spark.files data/test_data_1.csv,data/test_data_2.csv
```

spark.jars.packages

The scalaj package will be downloaded from Maven central and included on the Spark driver and executor classpaths.

spark.jars

The pre-existing jar, my_sample.jar, residing in the root of this project will be included on the Spark driver and executor classpaths.

spark.files

The two sample data sets, test_data_1.csv and test_data_2.csv, from the /data directory of this project will be distributed to the working directory of each Spark executor.

For more advanced configuration options, visit the Apache 2 reference documentation.

Related Information

Spark Configuration

LOG4J Configuration

Making Python on Apache Hadoop Easier with Anaconda and CDH

Spark Log4j Configuration

Cloudera AI allows you to update Spark's internal logging configuration on a per-project basis.

Spark 2 uses Apache Log4j, which can be configured through a properties file. By default, a log4j.properties file found in the root of your project will be appended to the existing Spark logging properties for every session and job. To specify a custom location, set the environmental variable LOG4J_CONFIG to the file location relative to your project.

The Log4j documentation has more details on logging options.

Increasing the log level or pushing logs to an alternate location for troublesome jobs can be very helpful for debugging. For example, this is a log4j.properties file in the root of a project that sets the logging level to INFO for Spark jobs.

```
shell.log.level=INFO
```

PySpark logging levels should be set as follows:

```
log4j.logger.org.apache.spark.api.python.PythonGatewayServer=<LOG_LEVEL>
```

And Scala logging levels should be set as:

```
log4j.logger.org.apache.spark.repl.Main=<LOG_LEVEL>
```

Setting up an HTTP proxy for Spark 2

Follow the guidelines on how to set up an HTTP proxy for Spark 2.

If you are using an HTTP proxy, you must set the Spark configuration parameter extraJavaOptions at runtime to be able to support web-related actions in Spark.

```
spark.driver.extraJavaOptions= \
-Dhttp.proxyHost=<YOUR HTTP PROXY HOST> \
-Dhttp.proxyPort=<HTTP PORT> \
-Dhttps.proxyHost=<YOUR HTTPS PROXY HOST> \
-Dhttps.proxyPort=<HTTPS PORT>
```

Spark web UIs

This topic describes how to access Spark web UIs from the Cloudera AI UI.

Spark 2 exposes one web UI for each Spark application driver running in Cloudera AI. The UI will be running within the container, on the port specified by the environmental variable CDSW_SPARK_PORT. By default, CDSW_SPARK_PORT is set to 20049. The web UI will exist only as long as a SparkContext is active within a session. The port is freed up when the SparkContext is shutdown.

Spark 2 web UIs are available in browsers at: https://spark-<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>. To access the UI while you are in an active session, click the grid icon in the upper right hand corner of the Cloudera AI web application, and select Spark UI from the dropdown. Alternatively, the Spark UI is also available as a tab in the session itself. For a job, navigate to the job overview page and click the History tab. Click on a job run to open the session output for the job.

Using Spark 2 from Python

Cloudera AI supports using Spark 2 from Python via PySpark. This topic describes how to set up and test a PySpark project.

PySpark environment variables

The default Cloudera AI engine currently includes Python 2.7.17 and Python 3.6.9. To use PySpark with lambda functions that run within the CDH cluster, the Spark executors must have access to a matching version of Python. For

many common operating systems, the default system Python will not match the minor release of Python included in Cloudera AI.

To ensure that the Python versions match, Python can either be installed on every CDH host or made available per job run using Spark's ability to distribute dependencies. Given the size of a typical isolated Python environment, Cloudera recommends installing Python 2.7 and 3.6 on the cluster if you are using PySpark with lambda functions.

You can install Python 2.7 and 3.6 on the cluster using any method and set the corresponding PYSPARK_PYTHON environment variable in your project. Cloudera AI includes a separate environment variable for Python 3 sessions called PYSPARK3_PYTHON. Python 2 sessions continue to use the default PYSPARK_PYTHON variable. This will allow you to run Python 2 and Python 3 sessions in parallel without either variable being overridden by the other.

Creating and running a PySpark project

To get started quickly, use the PySpark template project to create a new project. For instructions, see *Create a Project from a Built-in Template*.

To run a PySpark project, navigate to the project's overview page, open the workbench console and launch a Python session. For detailed instructions, see *Native Workbench Console and Editor*.

Testing a PySpark project in Spark local mode

Spark's local mode is often useful for testing and debugging purposes. Use the following sample code snippet to start a PySpark session in local mode.

```
from pyspark.sql import SparkSession

spark = SparkSession\
    .builder \
    .appName("LocalSparkSession") \
    .master("local") \
    .getOrCreate()
```

For more details, refer to the Spark documentation: *Running Spark Application*.

Related Information

Native Workbench Console and Editor

Example: Monte Carlo estimation

Consider the example for Monte Carlo estimation.

Within the template PySpark project, pi.py is a classic example that calculates Pi using the Montecarlo Estimation.

What follows is the full, annotated code sample that can be saved to the pi.py file.

```
# # Estimating $\pi$
#
# This PySpark example shows you how to estimate $\pi$ in parallel
# using Monte Carlo integration.

from __future__ import print_function
import sys
from random import random
from operator import add
# Connect to Spark by creating a Spark session
from pyspark.sql import SparkSession
spark = SparkSession\
    .builder\
    .appName("PythonPi")\
```

```
    .getOrCreate()

partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2
n = 100000 * partitions

def f(_):
    x = random() * 2 - 1
    y = random() * 2 - 1
    return 1 if x ** 2 + y ** 2 < 1 else 0

# To access the associated SparkContext
count = spark.sparkContext.parallelize(range(1, n + 1), partitions).map(f)
.reduce(add)
print("Pi is roughly %f" % (4.0 * count / n))

spark.stop()
```

Example: Locating and adding JARs to Spark 2 configuration

Consider the example for locating and adding JARs to Spark 2 configuration.

This example shows how to discover the location of JAR files installed with Spark 2, and add them to the Spark 2 configuration.

```
# # Using Avro data
#
# This example shows how to use a JAR file on the local filesystem on
# Spark on Yarn.

from __future__ import print_function
import os,sys
import os.path
from functools import reduce
from pyspark.sql import SparkSession
from pyspark.files import SparkFiles

# Add the data file to HDFS for consumption by the Spark executors.
!hdfs dfs -put resources/users.avro /tmp

# Find the example JARs provided by the Spark parcel. This parcel
# is available on both the driver, which runs in Cloudera AI, and the
# executors, which run on Yarn.
exampleDir = os.path.join(os.environ["SPARK_HOME"], "examples/jars")
exampleJars = [os.path.join(exampleDir, x) for x in os.listdir(exampleDir)]
# Add the Spark JARs to the Spark configuration to make them available for
use.
spark = SparkSession\
    .builder\
    .config("spark.jars", ",".join(exampleJars))\
    .appName("AvroKeyInputFormat")\
    .getOrCreate()
sc = spark.sparkContext

# Read the schema.
schema = open("resources/user.avsc").read()
conf = {"avro.schema.input.key": schema }
avro_rdd = sc.newAPIHadoopFile(
    "/tmp/users.avro", # This is an HDFS path!
    "org.apache.avro.mapreduce.AvroKeyInputFormat",
    "org.apache.avro.mapred.AvroKey",
```

```
    "org.apache.hadoop.io.NullWritable",
    keyConverter="org.apache.spark.examples.pythonconverters.AvroWrapperT
oJavaConverter",
    conf=conf)
output = avro_rdd.map(lambda x: x[0]).collect()
for k in output:
    print(k)
spark.stop()
```

Using Spark 3 from R

R users can access Spark 3 using sparklyr. Although Cloudera does not ship or support sparklyr, we do recommend using sparklyr as the R interface for Cloudera AI.

Before you begin

The spark_apply() function requires the R Runtime environment to be pre-installed on your cluster. This will likely require intervention from your cluster administrator. For details, refer the RStudio documentation.

Procedure

1. Install the latest version of sparklyr:

   ```
   install.packages("sparklyr")
   ```

2. Optionally, connect to a local or remote Spark 2 cluster:

   ```
   ## Connecting to Spark 3
   # Connect to an existing Spark 3 cluster using the spark_connect function.
   library(sparklyr)
   system.time(sc <- spark_connect())
   # The returned Spark 3 connection (sc) provides a remote dplyr data sou
   rce to the Spark 3 cluster.
   ```

   For a complete example, see *Importing Data into Cloudera AI*.

Related Information

sparklyr: R interface for Apache Spark

sparklyr Requirements

Using Spark 2 from Scala

This topic describes how to set up a Scala project for CDS 2.x Powered by Apache Spark along with a few associated tasks. Cloudera AI provides an interface to the Spark 2 shell (v 2.0+) that works with Scala 2.11.

Unlike PySpark or Sparklyr, you can access a SparkContext assigned to the *SPARK* (SparkSession) and *SC* (SparkContext) objects on console startup, just as when using the Spark shell.

By default, the application name will be set to CML_*SESSIONID*, where sessionId is the ID of the session running your Spark code. To customize this, set the spark.app.name property to the desired application name in a spark-defaul ts.conf file.

Pi.scala is a classic starting point for calculating Pi using the Monte Carlo estimation.

This is the full, annotated code sample.

```
//Calculate pi with Monte Carlo estimation
```

```
import scala.math.random
//make a very large unique set of 1 -> n
val partitions = 2
val n = math.min(100000L * partitions, Int.MaxValue).toInt
val xs = 1 until n

//split up n into the number of partitions we can use
val rdd = sc.parallelize(xs, partitions).setName("'N values rdd'")

//generate a random set of points within a 2x2 square
val sample = rdd.map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
  (x, y)
}.setName("'Random points rdd'")

//points w/in the square also w/in the center circle of r=1
val inside = sample.filter { case (x, y) => (x * x + y * y < 1) }.setName(
"'Random points inside circle'")
val count = inside.count()

//Area(circle)/Area(square) = inside/n => pi=4*inside/n

println("Pi is roughly " + 4.0 * count / n)
```

Key points to note:

- import scala.math.random

  Importing included packages works just as in the shell, and need only be done once.
- Spark context (*SC*).

  You can access a SparkContext assigned to the variable *SC* on console startup.

  ```
  val rdd = sc.parallelize(xs, partitions).setName("'N values rdd'")
  ```

Managing dependencies for Spark 2 and Scala

This topic demonstrates how to manage dependencies on local and external files or packages.

Example: Read files from the cluster local filesystem

Use the following command in the terminal to read text from the local filesystem. The file must exist on all hosts, and the same path for the driver and executors. In this example you are reading the file ebay-xbox.csv.

```
sc.textFile(“file:///tmp/ebay-xbox.csv”)
```

Adding remote packages

External libraries are handled through line magics. Line magics in the Toree kernel are prefixed with %. You can use Apache Toree's AddDeps magic to add dependencies from Maven central. You must specify the company name, artifact ID, and version. To resolve any transitive dependencies, you must explicitly specify the --transitive flag.

```
%AddDeps org.scalaj scalaj-http_2.11 2.3.0
import scalaj.http._
val response: HttpResponse[String] = Http("http://www.omdbapi.com/").param(
"t","crimson tide").asString
response.body
response.code
response.headers
```

```
response.cookies
```

Adding remote or local Jars

You can use the AddJars magic to distribute local or remote JARs to the kernel and the cluster. Using the *-F* option ignores cached JARs and reloads.

```
%AddJar http://example.com/some_lib.jar -f
%AddJar file:/path/to/some/lib.jar
```

Cloudera AI Top Tasks Map

Date published: 2022-09-19
Date modified: 2022-09-19

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Top Tasks for Cloudera AI.....4

Top Tasks for Cloudera AI

There are some questions we often hear from customers as they get started on using Cloudera AI in their work. We have listed several below, with links to the specific guidance you need, to give you a quick boost.

How do I bring Cloudera Data Platform data into Cloudera AI using data connections?

Cloudera AI offers a variety of ways to connect to data in a Cloudera Data Platform environment, such as connecting to Hive, Impala, or Spark data sources, or to a Cloudera Data Hub. These are detailed in *Exploratory Data Science and Visualization*.

How do I deploy a web-based application with Cloudera AI?

Cloudera AI makes it easy to create web-based applications, such as dashboards, to display the results of your data science work to business users or other consumers. This is explained in *Analytical Applications*.

How can I monitor a model running in production?

You can easily monitor active models in your workbench. For more information, see *Monitoring Active Models Across the Workbench*.

Cloudera AI

Troubleshooting Cloudera AI

Date published: 2020-07-16
Date modified: 2025-01-29

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Troubleshooting 4

Preflight checks 4
Instance type preflight check fails 14

Cloudera AI service with Data Lake upgrades 14

Common Cloudera AI errors and solutions 16

Troubleshooting Cloudera AI Workbenches on AWS 18

Troubleshooting Cloudera AI Workbenches on Azure 20

Logs for Cloudera AI Workbenches 21
Downloading diagnostic bundles for a workbench 21

Troubleshooting issues with workloads 22
Troubleshooting Spark issues 23
Troubleshooting Kerberos issues 23

Troubleshooting

This topic describes a recommended series of steps to help you start diagnosing issues with a Cloudera AI Workbench.

- Issues with Provisioning Cloudera AI Workbenches: If provisioning an Cloudera AI Workbench fails, first go to your cloud provider account and make sure that you have all the resources required to provision an Cloudera AI Workbench. If failures persist, start debugging by reviewing the error messages on the screen. Check the workbench logs to find where the problem originates from. For more details on the troubleshooting resources available to you, see Troubleshooting Cloudera AI Workbenches on AWS on page 18.
- Issues with Accessing Cloudera AI Workbenches: If your Cloudera AI Admin has already provisioned a workbench for you but attempting to access the workbench fails, confirm with your Cloudera AI Admin that they have completed all the steps required to grant you access.See: Configuring User Access to Cloudera AI
- Issues with Running Workloads: If you have access to a workbench but are having trouble running sessions/jobs/experiments, and so on, see if your error is already listed here: Troubleshooting issues with workloads on page 22 .

Cloudera Support

If you need assistance, contact Cloudera Support. Cloudera customers can register for an account to create a support ticket at the support portal. For Cloudera Data Platform issues in particular, make sure you include the Request ID associated with your error message in the support case you create.

Preflight checks

Part of provisioning and managing a Kubernetes cluster in the cloud is ensuring that your account and your environment is properly configured before beginning. The Cloudera Data Platform automatically runs a series of preflight checks which can help in determining if there is a problem before creating new resources or adjusting existing ones.

When creating a brand new cluster, preflight validation checks are used to ensure that the resources you are requesting as well as your environment are ready and configured correctly. For existing clusters, many of the infrastructure validations are skipped since they are not needed anymore. Instead, the validations which concern the resources being adjusted are executed.

Results

The result of running a collection of preflight checks is represented as a single aggregated value that lets you know whether it is safe to proceed with your actions. Each individual validation which was run is included in the response.

A preflight validation contains information concerning what it was checking for and what the result of that check was. For example, this is a very simple check to ensure that the type of node pool image is valid in the region:

Name	**Instance type**
Description	Instance groups must have an instance type that exists in the region in which they will be created. For EKS, there is additional verification for EKS support and usage class.
Category	COMMON
Status	FAILED

Name	Instance type
Message	The instance type validation failed.
Detailed Message	Instance type validation failed for Standard_B2s1 in west us2. Check to ensure that this instance type is valid in that region.
Duration	659 ms

PASSED

The validation successfully passed all criteria.

WARNING

The validation was either unable to fully check all of its criteria or it found a potential issue which could affect the success of the operation. This type of failure will not stop a cluster from being created or modified, but it does require further investigation.

SKIPPED

The validation was skipped because it does not apply to the current request. This can happen for many reasons, such as using a cloud provider that is not applicable to the preflight check.

FAILED

The validation failed its expected criteria and provisioning or updating of a cluster cannot proceed. In this case, there is an identified problem that needs resolution before continuing.

List of preflight checks

Node / Agent Pools

The following preflight validation checks pertain to the node / agent instance groups which are created to launch new nodes inside of the Kubernetes cluster.

Name	Description	Cloud	Type
Instance Count	The number of nodes in each instance group must not exceed a predefined threshold. Remediation: Change the requested size of an instance group to be within the boundaries specified by the error message.	All	Create Update
Group Count	The number of distinct node/agent pool groups must not exceed a predefined threshold. Remediation: Change the number of distinct instance groups to be within the boundaries specified by the error message.	All	Create Update

Name	Description	Cloud	Type
Instance Naming	The name of each instance group is restricted by cloud providers. There are differences in the size and characters allowed by each provider. Remediation: Change the name of the instance group to conform to the cloud provider's requirements. This could include changing the overall length of the name or only using certain approved characters.	All	Create Update
Instance Type	Instance image types are not universal across all regions. Some providers further restrict this to service level, and whether they can be used for Kubernetes. Remediation: Choose a different region or service type for the desired instance type. If this is not possible, then a different instance type must be chosen.	All	Create Update
Kubernetes Version	Each cloud provider supports different versions of Kubernetes. Cloudera Data Platform has also only been certified to work with particular versions. Remediation: Choose a different version of Kubernetes that is supported by your cloud provider in the region you are deploying in. https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-versions.html	All	Create
Placement Rules	Some instance types are not allowed to be grouped within a single availability zone.	AWS	Create Update

Name	Description	Cloud	Type
	Remediation: Remove the restriction on single availability zone, or choose a different instance type.		

Infrastructure

The following preflight validation checks pertain to Cloudera's control plane infrastructure and your specific account within that control plane.

Name	Description	Cloud	Type
Restricted IAM Policies	If your account is trying to provision with restricted IAM policies, then it needs to have those policies defined before deploying the cluster. Remediation: Check Cloudera's documentation on restricted IAM policies to ensure that you have the correctly named policies defined and are accessible.	AWS	Create
Proxy Connectivity	When provisioning a private cluster, your environment must have the cluster proxy enabled and it must be healthy. Remediation: Create a new environment which has the cluster proxy service enabled or check that your existing FreeIPA Virtual Machine is running and healthy. https://docs.cloudera.com/management-console/test/connection-to-private-subnets/topics/mc-ccm-overview.html	All	Create Update
Data Lake Connectivity	A healthy data lake with a functioning FreeIPA DNS server is required in order to provision a new cluster. Remediation: Check to ensure that the FreeIPA DNS server is running and healthy inside of your network.	All	Create

Name	Description	Cloud	Type
	https://docs.cloudera.com/management-console/cloud/data-lakes/topics/mc-data-lake.html https://docs.cloudera.com/management-console/cloud/identity-management/topics/mc-identity-management.html		

Networking

The following preflight validation checks pertain to specific network configurations inside of the cloud provider.

Name	Description	Cloud	Type
Shared Virtual Private Cloud	When a Virtual Private Cloud is shared between multiple subscriptions, access to modify this Virtual Private Cloud needs to be granted. Remediation: Check the permission for which roles can make modifications to the Virtual Private Cloud https://docs.aws.amazon.com/vpc/latest/userguide/vpc-sharing.html https://docs.cloudera.com/cdp-public-cloud/cloud/requirements-aws/topics/mc-aws-req-vpc.html	AWS	Create
Subnet Availability Zone	All subnets which are part of the environment must be located in at least 2 different Availability Zones. Remediation: Recreate your environment and choose subnets that satisfy the requirement of being in at least 2 different Availability Zones.	AWS	Create
Subnet Load Balancer Tagging	In order for load balancers to choose subnets correctly, a subnet needs to have either the public or private ELB tags defined.	AWS	Create

Name	Description	Cloud	Type
	Remediation: Tag subnets with either kubernetes.io/role/elb or kubernetes.io/role/internal-elb based on whether they are public or private. https://docs.cloudera.com/cdp-public-cloud/cloud/requirements-aws/topics/mc-aws-req-vpc.html		
API Server Access	Validates that there are no conflicting requests between Kubernetes API CIDR ranges and private AKS clusters. Remediation: When using a private AKS cluster, Kubernetes API CIDR ranges are not supported,	All	Create Update
Available Subnets	Subnets cannot be shared when provisioning Kubernetes clusters on Azure. At least one available subnet must exist that is not being used by another AKS cluster and must not have an existing route table with conflicting pod CIDRs. Remediation: Create a new subnet to satisfy this requirement or delete an old and unused cluster to free an existing subnet.	Azure	Create
Delegated Subnet	A subnet which has been delegated for a particular service cannot be used to provision an Azure AKS cluster. Remediation: Choose a different subnet or remove the delegated service from at least one subnet in the environment.	All	Create
Kubernetes API Server Security	Validates that the supplied IP CIDR ranges are valid and do not overlap any reserved IP ranges. Each	All	Create

Name	Description	Cloud	Type
	cloud provider has a limit set on the maximum number of allowed CIDRs. Remediation: Use valid CIDR formats and ranges when limiting access to the Kubernetes API server and limit the number of ranges specified.		
Kubernetes Service CIDR Validation	Validates that the specified service CIDR for Kubernetes services does not overlap any restricted CIDR ranges and is a valid CIDR format. Remediation: Change the service CIDR so that it does not conflict with any pod CIDRs or other routes on the subnet.	All	Create
Autoscale Parameters	Azure's built-in autoscaler has limitations on the ranges of values for scale-up and scale-down operations. Remediation: Adjust the specified parameters from the error message which are not within the required ranges.	Azure	Create Update

Example

```
{
   "result": "PASSED",
   "summary": {
       "passed": 8,
       "warning": 0,
       "failed": 0,
       "skipped": 10,
       "total": 18
   },
   "message": "The cluster validation has passed, but some checks were skip
ped",
   "validations": [
       {
           "name": "Instance Count",
           "description": "Each instance count must be between minInstance a
nd maxInstance inclusively. The minInstance and maxInstance of infrastructur
e group should comply with minimum number of infra nodes and maximum number
of infra nodes.",
           "category": "COMMON",
           "status": "PASSED",
```

```
            "message": "The minimum and maximum instance counts are correct
 for all instance groups.",
            "detailedMessage": "The minimum and maximum instance counts are c
orrect for all instance groups.",
            "duration": "1µs"
        },
        {
            "name": "Instance Group Count",
            "description": "Total instance group count must be less than or
equal to maximum instance group limit.",
            "category": "COMMON",
            "status": "PASSED",
            "message": "The number of instance groups in the request is less
 than or equal to the maximum allowed.",
            "detailedMessage": "The total instance group count of 2 is within
 the limit.",
            "duration": "3µs"
        },
        {
            "name": "Instance Group Naming",
            "description": "Each instance group name must conform the rest
rictions of the cloud provider. This includes using valid characters and adh
ering to length restrictions.",
            "category": "COMMON",
            "status": "PASSED",
            "message": "All instance groups meet the naming restrictions for
Azure.",
            "detailedMessage": "All instance groups meet the naming restrict
ions for Azure.",
            "duration": "12µs"
        },
        {
            "name": "Instance Type",
            "description": "Instance groups must have an instance type that
 exists in the region in which they will be created. For EKS, there is addi
tional verifiation for EKS support and usage class.",
            "category": "COMMON",
            "status": "PASSED",
            "message": "All instance groups have valid instance types.",
            "detailedMessage": "The following instance types were validated
 for westus2: Standard_B2s",
            "duration": "599ms"
        },
        {
            "name": "Kubernetes Version",
            "description": "Each cloud provider (Amazon, Azure, Google, etc)
supports different versions of Kubernetes.",
            "category": "COMMON",
            "status": "PASSED",
            "message": "The specified Kubernetes version 1.18 has been resol
ved to 1.18.17 and is valid on Azure",
            "detailedMessage": "The specified Kubernetes version 1.18 has
been resolved to 1.18.17 and is valid on Azure",
            "duration": "388ms"
        },
        {
            "name": "Placement Rule",
            "description": "Instance Types must be allowed by the placement r
ule.",
            "category": "COMMON",
            "status": "SKIPPED",
            "message": "Skipping validation since the cloud platform is Azure
.",
```

```
            "detailedMessage": "Skipping validation since the cloud platform
is Azure.",
            "duration": "6µs"
        },
        {
            "name": "Entitlement Check",
            "description": "When the entitlement LIFTIE_USE_PRECREATED_IAM_RE
SOURCES is enabled, the expected profile (cdp-liftie-instance-profile) shoul
d exist and it needs to have the necessary roles attached to it. ",
            "category": "ENTITLEMENTS",
            "status": "SKIPPED",
            "message": "IAM Resource Entitlement validation skipped for Cl
oud Provider azure.",
            "detailedMessage": "IAM Resource Entitlement validation skipped
 for Cloud Provider azure.",
            "duration": "14µs"
        },
        {
            "name": "Cluster Proxy Connectivity",
            "description": "Verifies connectivity to the cluster proxy ser
vice which is used to register private cluster endpoints.",
            "category": "CONTROL_PLANE",
            "status": "SKIPPED",
            "message": "Connectivity to the cluster connectivity manager will
 be skipped since this is not a private cluster.",
            "detailedMessage": "The cluster being provisioned is not marked
 as private in the provisioning request.",
            "duration": "1µs"
        },
        {
            "name": "Cluster Proxy Enabled",
            "description": "Verifies that the environment was created with th
e cluster proxy service enabled.",
            "category": "CONTROL_PLANE",
            "status": "SKIPPED",
            "message": "Skipping the environment check for cluster proxy c
onnectivity since the cluster is public.",
            "detailedMessage": "Skipping the environment check for cluster
 proxy connectivity since the cluster is public.",
            "duration": "1µs"
        },
        {
            "name": "DataLake Connectivity",
            "description": "Validates whether DataLake connection is reach
able and if FreeIPA is available.",
            "category": "CONTROL_PLANE",
            "status": "PASSED",
            "message": "DataLake validation succeeded.",
            "detailedMessage": "Data lake is healthy and reachable. Service
 Discovery Feature is enabled, verified DNS entries retrieved for Data Lake
s. Datalake URL : localhost:8081  Service discovery URL : localhost:8082 "
        },
        {
            "name": "AWS Shared VPC Access",
            "description": "When a shared VPC is used, proper access should
 be granted.",
            "category": "NETWORK",
            "status": "SKIPPED",
            "message": "Skipping validation since the cloud platform is Azur
e.",
            "detailedMessage": "Skipping validation since the cloud platform
 is Azure.",
            "duration": "3µs"
        },
```

```
        {
            "name": "AWS Subnet Availability Zones",
            "description": "When existing AWS subnets are provided for provi
sioning an EKS cluster, the subnets must be in at least 2 different Availabi
lity Zones.",
            "category": "NETWORK",
            "status": "SKIPPED",
            "message": "Skipping validation since the cloud platform is Azure
.",
            "detailedMessage": "Skipping validation since the cloud platform
is Azure.",
            "duration": "2µs"
        },
        {
            "name": "AWS Subnet Tagging",
            "description": "In order for load balancers to choose subnets co
rrectly a subnet needs to have either the public or private ELB tags defined
.",
            "category": "NETWORK",
            "status": "SKIPPED",
            "message": "Skipping validation since the cloud platform is Azu
re.",
            "detailedMessage": "Skipping validation since the cloud platform
 is Azure.",
            "duration": "26µs"
        },
        {
            "name": "Azure API Access Parameters",
            "description": "Verifies that the security parameters for locking
 down access to the Azure Kubernetes API Server are correct.",
            "category": "NETWORK",
            "status": "PASSED",
            "message": "The API server access parameters specified in the cl
uster request are valid.",
            "detailedMessage": "The cluster will be provisioned as public
with the following whitelist CIDRs: ",
            "duration": "48µs"
        },
        {
            "name": "Azure Available Subnets",
            "description": "When an existing Azure subnet is chosen for pro
visioning an AKS cluster, the subnet must not be in use by any other cluster
. This is a restriction of Kubenet, which is the CNI used on the new cluster
. Although the subnet may have a routing table, it may not have any existing
 IP address associations.",
            "category": "NETWORK",
            "status": "PASSED",
            "message": "At least 1 valid subnet was found and can be used for
 cluster creation.",
            "detailedMessage": "The cluster can be provisioned using subnet
 liftie-dev.internal.2.westus2 in virtual network liftie-dev and resource gr
oup liftie-test",
            "duration": "917ms"
        },
        {
            "name": "Kubernetes API Server CIDR Security",
            "description": "CIDR blocks for whitelisting access to the Kube
rnetes API Server must not overlap restricted IP ranges.",
            "category": "NETWORK",
            "status": "SKIPPED",
            "message": "Skipping CIDR validation for whitelisting because it
 is not enabled.",
            "detailedMessage": "The ability to secure access the Kubernetes
 API server via a list of allowed CIDRs is not enabled. This can be enabled
```

```
 either in the controlplane (currently false) or via the provisioning reques
t (currently false).",
            "duration": "11µs"
        },
        {
            "name": "Service CIDR Validation",
            "description": "CIDR blocks that Kubernetes assigns service IP
addresses from should not overlap with any other networks that are peered or
 connected to existing VPC.",
            "category": "NETWORK",
            "status": "SKIPPED",
            "message": "Service CIDR is missing in Network Profile.",
            "detailedMessage": "VPC validation is executed only if the VPC a
lready exists and a service CIDR is specified in the network profile.",
            "duration": "426µs"
        },
        {
            "name": "Azure Autoscale Parameters",
            "description": "The following autoscale parameters for Azure,
which are specified during provisioning and update, need to be in multiples
of 60s. Autoscale parameters: scaleDownDelayAfterAdd, scaleDownDelayAfterFai
lure, scaleDownUnneededTime, scaleDownUnreadyTime.",
            "category": "DEPLOYMENT",
            "status": "SKIPPED",
            "message": "There are no autoscale parameters specified in the
request.",
            "detailedMessage": "The request did not contain an Autoscaler
structure.",
            "duration": "0s"
        }
    ]
}
```

Instance type preflight check fails

On AWS, in a given region, the corresponding Availability Zones (AZs) do not always support the same instance types. The instance type preflight check will fail if the desired instance type is not supported in all AZs in the region.

If the preflight check fails, check all failed AZs for the supported instance types using this command:

```
aws ec2 describe-instance-type-offerings --location-type availability-zone -
-filters Name=location,Values=<az>
   --query "InstanceTypeOfferings[].InstanceType" --region <region>
```

Remediation:

It is possible, but not guaranteed, that skipping the validation will result in successful provisioning of a Cloudera AI Workbench. AWS selects the AZ for creating an instance group based on the VPC and Subnet configuration of the customer’s datalake; the user cannot choose the AZ.

Alternatively, select an instance type that is supported by all AZs in the region.

Cloudera AI service with Data Lake upgrades

Cloudera Data Platform environments have two services which can be upgraded individually, the FreeIPA service and the Data Lake (DL) service. Cloudera AI Workbenches run in Cloudera Data Platform environments. The FreeIPA service provides identity management, and the Data Lake service provides SDX capabilities to Cloudera AI Workbenches.

In this document we provide FAQs for the behavior of Cloudera AI Workbenches during a Data Lake upgrade. For information on FreeIPA upgrades, see *Upgrade FeeIPA*.

What kinds of Data Lake upgrades are possible?

The Data Lake service supports the following upgrades.

- Hotfix upgrades
- Version upgrades
- OS version upgrades

These upgrades can all be done from the Cloudera Data Platform Data Lake service UI, or with the Cloudera Data Platform CLI. DL upgrades require downtime. DL upgrades preserve the state of the data lake during the upgrade.

During DL upgrades, the shape of the data lake cannot change. For example, we cannot change a LIGHT_DUTY shape to a MEDIUM_DUTY_HA shape during the DL upgrade.

What is Data Lake migration and is it supported?

To change the DL shape, perform a DL migration, also called DL scaling. For example, a DL migration can change a DL LIGHT_DUTY shape to a MEDIUM_DUTY_HA shape.

DL migration is currently available as a technical preview. Speak to your Cloudera account team to discuss whether it may be suitable for your situation.

What happens if the Data Lake upgrade/migration fails?

There is no automated backup and restore process for the DL. It is recommended to perform a backup of the DL before starting the upgrade process. If the DL upgrade fails, the recommended option is to delete the failed DL using the Cloudera Data Platform CLI (`cdp datalake delete-datalake --datalake-name <dl name>`) and recreate the DL using the `cdpcli`. Once the DL is recreated, you need to restore the DL state from the backup. For more information, refer to *Backup and Restore for the Data Lake*.

Do not delete the environment service during a failed DL upgrade process. Deleting an environment causes all Cloudera AI Workbench running in this environment to be unusable.

Can the environment service be deleted and recreated at any point if the Data Lake upgrade or migration process has an error?

No, environments with experiences running inside them cannot be deleted at any time. If you delete the environment, then all the experiences (such as a Cloudera AI Workbench) need to be deleted.

Unless a Cloudera AI Workbench is first backed up and restored, then all state information is lost and you need to start from a fresh workbench. For more information, see *Backing up Cloudera AI Workbenches*.

What are the Cloudera AI Workbench prerequisites for Data Lake upgrades/migrations?

Do the following before upgrading or migrating a Data Lake.

- Upgrade Cloudera AI Workbenches to the latest version (if an upgrade is available).
- Stop any jobs, sessions, experiments or any workloads that need DL access before performing a DL upgrade.
- Announce to the team that there will be planned downtime for Cloudera AI Workbenches during the DL upgrade process.

Are the Cloudera AI Workbenches operational during Data Lake upgrades/migrations?

It is recommended NOT to use Cloudera AI Workbenches during DL upgrades.

However, the observed behavior of Cloudera AI Workbenches during DL upgrades is as follows.

- Cloudera AI Workbenches remain accessible during DL upgrades. Users can log in to a Cloudera AI Workbench.

- Users can launch sessions, run jobs, experiments, models, and so on which do not require DL access. For example, jobs that do not require IDBroker or SDX/HMS access will function normally.
- Any compute instance that requires IDBroker or SDX/HMS access will fail.
- Any scheduled jobs that require IDBroker or SDX access will fail.
- With regular DL upgrades and migrates, you should suspend Cloudera AI Workbenches. This require downtime.
- With Zero Downtime Upgrade (ZDU) of DL, Cloudera AI Workbenches will be functional during upgrades. If the DL version is 7.2.17 or lower, you may encounter the CDPD-66549 known issue.

Are any changes to Cloudera AI Workbenches needed after a DL is upgraded or migrated successfully?

No further actions are required on a Cloudera AI Workbench after a successful DL upgrade. Cloudera AI Workbenches continue to function normally. Make sure to announce to the team that they can start using Cloudera AI Workbenches as usual.

Related Information

Backup and restore for the Data Lake

Backing up ML workspaces

Upgrade FreeIPA

Common Cloudera AI errors and solutions

The following sections describe recommended steps to start debugging common error messages you might see in the workbench logs (found under EventsView Logs).

Before you begin

Make sure you have reviewed the list of resources available to you for debugging on Cloudera AI and AWS: Troubleshooting Cloudera AI Workbenches on AWS on page 18

Timezone not properly set for scheduled job

If the timezone is no longer correctly set for a scheduled job, then you should simply set it again. Go to Job Settings, edit the timezone, and update the job.

AWS account resource limits exceeded (Compute, VPC, etc.)

Cloudera AI Workbench provisioning fails because Cloudera Data Platform could not get access to all the AWS resources needed to deploy a Cloudera AI Workbench. This is likely because your AWS account either does not have access to those resources or is hitting the resource limits imposed on it.

Sample errors include (from EventsView Logs):

```
Failed to provision cluster. Reason: Failed to wait for provisioner: Wait fo
r status failed with status CREATE_FAILED: error creating eks cluster (cause
: InvalidParameterException: Provided subnets subnet-0a648a0cc5976b7a9 Free
IPs: 0 , need at least 3 IPs in each subnet to be free for this operation
```

```
Failed to mount storage. Reason: Failed to create mount target: NoFreeAddres
sesInSubnet: The specified subnet does not have enough free addresses to sat
isfy the request.
```

AWS accounts have certain hard and soft resource limits imposed on them by default. For example, certain CPU/GPU instances that Cloudera AI allows you to provision might even have an initial default limit of 0 (set by AWS). This means if you attempt to provision a cluster with those instance types, your request will fail.

Aside from the CPU and GPU compute resource limits, there are other types of limits you can run into. For example, the second error shows that the subnets in your VPC do not have any more free IP addresses for the workbench (and each of the underlying Kubernetes pods). This occurs if the CIDR range mentioned while registering the environment was not large enough for your current needs.

You can use the AWS console to request an increase in limits as needed. Go to the AWS console for the region where the environment was provisioned and then navigate to EC2 Limits.

For networking failures, navigate to EC2VPC. Search for the environment's VPC ID (available on environment Summary page) to see the list of available IP addresses for each subnet. Request more resources as needed.

Related AWS documentation: AWS Service Limits, Amazon EC2 Resource Limits, EKS Cluster VPC Considerations, AWS CNI Custom Networking.

Access denied to AWS credential used for authentication

The cloud credential used to give Cloudera Data Platform access to your AWS account failed authentication. Therefore, Cloudera Data Platform could not provision the resources required to deploy a Cloudera AI Workbench.

Sample error (from EventsView Logs):

```
Failed to provision storage. Reason: Failed to create new file system: Acces
sDenied: User: arn:aws:iam::1234567890:user/cross-account-trust-user is not
authorized to perform:
```

Your cloud credential gives Cloudera Data Platform access to the region and virtual network that make up the environment thus allowing Cloudera Data Platform to provision resources within that environment. If authentication fails, go to your environment to see how the cloud credentials were set up and confirm whether your account has the permission to perform these actions.

Cloudera AI installation failures

While the steps to provision resources on AWS were completed successfully, the Cloudera AI Workbench installation on EKS failed.

Sample error (from EventsView Logs):

```
Failed to install Cloudera AI Workbench. Reason:Error: release mlx-mlx fail
ed: timed out waiting for the condition
```

If you are an advanced user, you can log in to the underlying EKS cluster and use kubectlto investigate further into which pods are failing.

Note: This error might be an indication that DNS has been turned off for the VPC. Go to the AWS console for the region where the environment was provisioned and then navigate to EC2 Load Balancers to confirm that DNS is configured properly for the environment's VPC.

Related AWS documentation: EKS and kubectl

Failures due to API throttling

These errors can be harder to prepare for due to their seemingly random nature. Occasionally, AWS will block API calls if it receives too many requests at the same time. For example, this can occur when multiple users are attempting to provision/delete/upgrade clusters at the same time.

Sample error (from EventsView Logs):

```
Failed to delete cluster. Reason: Failed to wait for deletion: Wait for stat
us failed with status DELETE_FAILED: Throttling: Rate exceeded
```

Currently, if you see a 'Throttling: Rate exceeded' error, our recommendation is that you simply try again later.

Related AWS documentation: AWS API Request Throttling

De-provisioning failures

De-provisioning operations can sometimes fail if AWS resources are not terminated in the right order. This is usually due to timing issues where certain resources might take too long to terminate. This can result in a cascading set of failures where AWS cannot delete the next set of resources because they still have active dependencies on the previous set.

Sample error (from EventsView Logs):

```
Failed to delete cluster. Reason: Failed to wait for deletion: DELETE_FAILED
: msg: failed to delete aws stack
Cloudformation says resource xyz has a dependent object (Service: AmazonEC2;
 Status Code: 400; Error Code: DependencyViolation; Request ID: 815928e2-2
77e-4b8b-9fed-4b89716a205b) EKS - cluster still existed, was blocking CF del
ete
```

Cloudera AI includes a Force Delete option now that will remove the workbench from the Cloudera AI service. However, this not mean all the underlying resources have been cleaned up. This is where tags are very useful.

If you assigned tags to the workbench at the time of provisioning, you can use the AWS console or the CLI to query the tags associated with the workbench to see if any resources need to be cleaned up manually. Tags associated with a workbench are available on the workbench Details page.

You can search by tags in the EC2 and VPC services. You can also use the AWS CLI to search for specific tags: resourcegroupstaggingapi

Users unable to access provisioned Cloudera AI Workbenches

If you have provisioned a workbench but your colleagues cannot automatically access the workbench using Cloudera Data Platform Single-Sign on, make sure that you have completed all the steps required to grant users access to workbenches: Configuring User Access to Cloudera AI. All Cloudera AI users must have Cloudera Data Platform accounts.

Troubleshooting Cloudera AI Workbenches on AWS

This topic describes the Cloudera AI Workbench provisioning workflow and tells you how to start debugging issues with Cloudera AI Workbenches on AWS.

Cloudera AI Workbench Provisioning Workflow

When you provision a new Cloudera AI Workbench on AWS, Cloudera AI performs the following actions:

1. Communicates with the Cloudera Management Console to check your AWS credentials. It will also enable Single Sign-On so that authorized Cloudera Data Platform users are automatically logged in to the workbench that will be created.
2. Provisions an NFS filesystem for the workbench on your cloud service provider. On AWS, Cloudera AI will provision storage on EFS.
3. Provisions a Kubernetes cluster on your cloud service provider. This cluster runs the workbench infrastructure and compute resources. On AWS, Cloudera AI provisions an EKS cluster.
4. Mounts the provisioned NFS filesytem to the Kubernetes cluster.
5. Provisions TLS certificates for the workbench using LetsEncrypt.
6. Registers the workbench with the cloud provider's DNS service. On AWS, this is Route53.
7. Installs Cloudera AI onto the EKS cluster.

Troubleshooting resources

Any of the steps listed above can experience failures. To start debugging, you will require access to one or more of the following resources.

- Workbench > Details Page

  Each workbench has an associated Details page that lists important information about the workbench. To access this page, sign in to Cloudera Data Platform, go to Cloudera AI Workbenches and click on the workbench name.

  This page lists basic information about the workbench such as who created it and when. More importantly, it includes a link to the environment where the workbench was created, a link to the underlying EKS cluster on AWS, a list of tags associated with the workbench, and the computing resources in use. The rest of this topic explains how to use these resources.
- Workbench > Events Page

  Each workbench also has an associated Events page that captures every action performed on the workbench. This includes creating, upgrading, and removing the workbench, among other actions. To access this page, sign in to Cloudera Data Platform, go to Cloudera AI Workbenchs, click on the workbench name, and then click Events.

  Click the View Logs button associated with an action to see a high-level overview of all the steps performed by Cloudera AI to complete the action.

  The Request ID associated with each action is especially useful in case of a failure as it allows Cloudera Support to efficiently track the series of operations that led to the failure.
- Environment > Summary Page

  Cloudera AI Workbenches depend quiteCloudera AI Workbencheshe environment in which they are provisioned. Each environment's Summary page lists useful information that can help you debug issues with the Cloudera AI service. You can access the environment directly from the workbench Details page.

  This page includes important information such as:

  - Credential Setup - Tells you how security has been configured for the environment. Your cloud credential gives Cloudera Data Platform access to the region and virtual network that make up the environment thus allowing Cloudera Data Platform to provision resources within that environment.
  - Region - The AWS region where the environment is provisioned. This is especially important because it tells you which region's AWS console you might need to access for further debugging.
  - Network - The VPC and subnets that were created for the environment. Each Cloudera AI Workbench requires a set of unique IP addresses to run all of its associated Kubernetes services. If you begin to run out of IP addresses, you will need these VPC and subnet IDs to debug further in the AWS console.
  - Logs - When you create a Cloudera Data Platform environment, you are asked to specify an S3 bucket in that environment that will be used to store logs. All Cloudera AI operational logs and Spark logs are also written to this bucket.

    You can use the AWS console to access these logs. Alternatively, Site administrators can download these logs directly from their workbench Site Admin panel (Admin Support).

    Note: If you file a support case, Cloudera Support will not automatically have access to these logs because they live in your environment.

- AWS Management Console

  If you have all the relevant information about the environment and the workbench, you can go to the AWS console (for the region where your environment was created) to investigate further. The AWS Management Console has links to dashboards for all the services used by Cloudera AI.

  - EC2

    You can use the EC2 service dashboards to check the instance-type (CPU, GPU), VPC, subnet, and security group limits imposed on your AWS account. For example, there is typically a limit of 5 VPCs per region.

    If you need more resources, submit a request to Amazon to raise the limit of a resource.
  - EKS

    EKS will give you more information such as the version of Kubernetes Cloudera AI is using, network information, and the status of the cluster. The workbench Details page gives you a direct link to the provisioned EKS cluster on the AWS console.

    Note: By default, users do not have Kubernetes-level access to the EKS cluster. If a user wants to use kubectl to debug issues with the EKS cluster directly, an MLAdmin must explicitly grant access using the instructions provided here: Granting Remote Access to Cloudera AI Workbenches on EKS.
  - VPC

    Use the VPC ID obtained from the Cloudera Data Platform environment Summary page to search for the relevant VPC where you have provisioned or are trying to provision an Cloudera AI Workbench. Each Cloudera AI Workbench requires a set of unique IP addresses to run all of its associated Kubernetes services. You can use this service to see how many IP addresses are available for each subnet.
  - S3

    Use the S3 bucket configured for the environment to check/download logs for more debugging.
  - Tags

    When provisioning an Cloudera AI Workbench, you will have the option to assign one or more tags to the workbench. These tags are then applied to all the underlying AWS resources used by the workbench. If failures occur during provisioning or de-provisioning, it can be very useful to simply query the tags associated with the workbench to see if any resources need to be cleaned up manually. Tags associated with a workbench are available on the workbench Details page.

    You can search by tags in the EC2 and VPC services. You can also use the AWS CLI to search for specific tags: resourcegroupstaggingapi
  - Trusted Advisor (available with AWS Support)

    Use the Trusted Advisor dashboard for a high-level view of how you are doing with your AWS account. The dashboard displays security risks, service limits, and possible areas to optimize resource usage. If you have access to AWS Support, Cloudera recommends to review your current account status with Trusted Advisor before you start provisioning Cloudera AI Workbenches.

Troubleshooting Cloudera AI Workbenches on Azure

You can collect logs to troubleshoot issues that occur in Cloudera AI Workbenches with Azure.

How to access Azure logs

Logs from the AKS control plane can be found in the "Logs" blade of the liftie-xxxxxxx resource group (not to be confused with the "Logs" blade of the AKS cluster itself or the Log Analytics Workbench in that resource group). The logs can be looked up using a query language developed by Microsoft.

Cluster fails to scale down

If a worker node is idle but is not being scaled down, check the cluster autoscaler logs.

Use this example to look up the logs:

```
AzureDiagnostics | where Category ==
         "cluster-autoscaler"
```

The logs list the pods that are scheduled on a given node that are preventing it from being scaled down, or other reasons for its scaling decisions. Services running in the `kube-system` namespace (such as `tunnelfront`, or `metrics-server`) have been known to delay scale-down when scheduled on an otherwise idle node.

Delete Cloudera AI Workbench fails

If you delete a workbench, and the delete operation fails, you can use Force delete to remove the workbench.

In this case, Cloudera AI attempts to delete associated cloud resources for the workbench including metadata files. However, users should check that all such resources have been deleted, and delete manually if necessary.

Logs for Cloudera AI Workbenches

You can access logs to troubleshoot issues with the Cloudera AI service and your workloads on Cloudera AI Workbenches.

Access to logs

When you create a Cloudera Data Platform environment, you specify an S3 bucket (on AWS) or an Azure Storage container (on Azure) in that environment for storing logs. If you have access to the log storage, you can use the AWS or Azure console to access certain Cloudera AI and Spark logs directly. You can get the details of the specific bucket or container from the Troubleshooting Cloudera AI Workbenches on AWS page for the environment.

Note: If you file a support case, Cloudera Support will not automatically have access to these logs because they live in your environment.

Cloudera AI Workbench access to logs

Cloudera AI Workbench users also have access to these logs depending on their authorization level:

- Site Administrators

  Site administrators can download the same logs directly from their workbench Site Admin panel (Admin Support). For more details, see Downloading diagnostic bundles for a workbench on page 21.
- Data Scientists

  While data scientists do not have access to the full set of workbench logs, they do have access to engine logs for their own workloads (sessions/jobs/experiments). While in an interactive session or on a job/experiment's Overview page, click Download Logs at any time to review the full set of logs for that workload's engine. In the case of Spark workloads, Spark executor and event logs are also downloaded as part of this bundle.

Related Information

Configure lifecycle management for logs on AWS

Configure lifecycle management for logs on Azure

Downloading diagnostic bundles for a workbench

This topic describes how to download diagnostic bundles for an Cloudera AI Workbench.

Before you begin

Required Role: MLAdmin

Make sure you are assigned the MLAdmin role in Cloudera Data Platform. Only users with the MLAdmin role will be logged into Cloudera AI Workbenches with Site Administrator privileges.

Procedure

1. Log in to the web interface.
2. In Cloudera AI Workbenches, select a workbench.
3. Select Site Administration Support Generate Log Archive .
4. Select the time period from the dropdown.
5. Ensure Include Engines is selected if engine logs are needed (included by default).
6. Select Send to Cloudera to send the diagnostic logs to Cloudera Support.
7. Select Create to generate the logs.
8. When Status is Complete, select Download to download the diagnostics bundles to your machine.

What to do next

The data in the contained bundles may be incomplete. If it does not contain logs for time period you are looking for, there are a number of possible reasons:

- There is a delay between the time the logs are initially generated by a workload and the time they are visible in cloud storage. This may be approximately 1 minute due to buffering during streaming, but can be significantly longer due to eventual consistency in the cloud storage.
- Another user or process may have deleted data from your bucket; this is beyond the control of Cloudera AI.
- There may be a misconfiguration or an invalid parameter in your request. Retrieving logs requires a valid cloud storage location to be configured for logging, as well as authentication for Cloudera AI to be set up properly for it. Requests must pertain to a valid engine in a valid project.

Troubleshooting issues with workloads

This section describes some potential issues data scientists might encounter once the Cloudera AI Workbench is running workloads.

401 Error caused by incompatible Data Lake version

The following error might occur due to an incompatible Data Lake version:

```
org.apache.ranger.raz.hook.s3.RazS3ClientCredentialsException: Exception in
Raz Server;
Check the raz server logs for more details, HttpStatus: 401
```

To avoid this issue, ensure that:

- Data Lake and Runtime (server) version is 7.2.11 or higher.
- Hadoop Runtime add-on (client) used in the Cloudera AI session is 7.2.11 or higher.
- Spark Runtime add-on version must be CDE 1.13 or higher.

Engines cannot be scheduled due to lack of CPU or memory

A symptom of this is the following error message in the Workbench: "Unschedulable: No node in the cluster currently has enough CPU or memory to run the engine."

Either shut down some running sessions or jobs or provision more hosts for Cloudera AI.

Workbench prompt flashes red and does not take input

The Workbench prompt flashing red indicates that the session is not currently ready to take input.

Cloudera AI does not currently support non-REPL interaction. One workaround is to skip the prompt using appropriate command-line arguments. Otherwise, consider using the terminal to answer interactive prompts.

PySpark jobs fail due to Python version mismatch

```
Exception: Python in worker has different version 2.6 than that in driver 2.
7, PySpark cannot run with different minor versions
```

One solution is to install the matching Python 2.7 version on all the cluster hosts. A better solution is to install the Anaconda parcel on all CDH cluster hosts. Cloudera AI Python engines will use the version of Python included in the Anaconda parcel which ensures Python versions between driver and workers will always match. Any library paths in workloads sent from drivers to workers will also match because Anaconda is present in the same location across all hosts. Once the parcel has been installed, set the PYSPARK_PYTHON environment variable in the Cloudera AI Admin dashboard.

Troubleshooting Spark issues

Consider some common Spark issues and their recommended solutions.

Distributing SparklyR packages

When using SparklyR, you may see an error from Spark similar to the following:

```
ERROR sparklyr: RScript (4922) terminated
unexpectedly: namespace ‘vctrs’ 0.6.3 is being loaded, but >= 0.6.4 is re
quired
```

This means you need to upgrade the vctrs package, and configure Spark to use the R libraries available in the .home directory of the Cloudera AI session. This works because with Spark on Kubernetes, the Spark executors are running in the same cluster and have access to the same underlying filesystem. Include the following code when creating the Spark session:

```
install.packages("vctrs")
print(packageVersion("vctrs")) # expect 0.6.5
...
config$spark.executorEnv.R_LIBS="/home/cdsw/.local/lib/R/4.3/library"
config$spark.executorEnv.R_LIBS_USER="/home/cdsw/.local/lib/R/4.3/library"
config$spark.executorEnv.R_LIBS_SITE="/opt/cmladdons/r/libs"
...
spark_apply(test_fx, packages = TRUE) # true by default
```

Troubleshooting Kerberos issues

This topic describes some common Kerberos issues and their recommended solutions.

HDFS commands fail with Kerberos errors even though Kerberos authentication is successful in the web application

If Kerberos authentication is successful in the web application, and the output of klist in the engine reveals a valid-looking TGT, but commands such as hdfs dfs -ls / still fail with a Kerberos error, it is possible that your cluster is missing the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File. The JCE policy file is required when Red Hat uses AES-256 encryption. This library shall be installed on each cluster host and will live under $JAVA_HOME. For more information, see Using AES-256 Encryption.

Cannot find renewable Kerberos TGT

Cloudera AI runs its own Kerberos TGT renewer which produces non-renewable TGT. However, this confuses Hadoop's renewer which looks for renewable TGTs. If the Spark 2 logging level is set to WARN or lower, you may see exceptions such as:

```
16/12/24 16:38:40 WARN security.UserGroupInformation: Exception encountered
while running the renewal command. Aborting renew thread. ExitCodeException
exitCode=1: kinit: Resource temporarily unavailable while renewing credentia
ls

16/12/24 16:41:23 WARN security.UserGroupInformation: PriviledgedActionExcep
tion as:user@CLOUDERA.LOCAL (auth:KERBEROS) cause:javax.security.sasl.SaslEx
ception: GSS initiate failed [Caused by GSSException: No valid credentials p
rovided (Mechanism level: Failed to find any Kerberos tgt)]
```

This is not a bug. Spark 2 workloads will not be affected by this. Access to Kerberized resources should also work as expected.

Cloudera AI

Personal and Team Accounts

Date published: 2020-07-16
Date modified: 2025-01-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

User Roles 4

Business Users and Cloudera AI 5

Managing your Personal Account 5

Creating a Team 6

Managing a Team Account 7

User Roles

Users in Cloudera AI are assigned one or more of the following roles.

Important: Cloudera Public Cloud allows customers to maintain full ownership and control of their data and workloads and is designed to operate in some of the most restricted Public Cloud environments. Since Cloudera Public Cloud runs in a customer's cloud account, Security and Compliance is a shared responsibility between Cloudera and its Public Cloud customers. User roles form the first layer of security for securing the Cloudera AI workloads.

It is your responsibility to diligently allocate the permissions to the users. For more information, see *Cloudera's Shared Responsibility Model*.

There are two categories of roles: environment resource roles, which apply to a given Cloudera Data Platform environment, and workbench resource roles, which apply to a single workbench. To use workbench resource roles, you may need to upgrade the workbench or create a new workbench.

If a user has more than one role, then the role with the highest level of permissions takes precedence. If a user is a member of a group, it may gain additional roles through that membership.

Environment resource roles

- MLAdmin: Grants a Cloudera Data Platform user the ability to create and delete Cloudera AI Workbenches within a given Cloudera Data Platform environment. MLAdmins also have Administrator level access to all the workbenches provisioned within this environment. They can run workloads, monitor, and manage all user activity on these workbenches. They can also add the MLUser and MLBusinessUser roles to their assigned environment. This user also needs the account-level role of IAMViewer, in order to access the environment Manage Access page. To create or delete workbenches, this user also needs the EnvironmentAdmin role.
- MLUser: Grants a Cloudera Data Platform user the ability to view Cloudera AI Workbenches provisioned within a given Cloudera Data Platform environment. MLUsers are also able to run workloads on all the workbenches provisioned within this environment.
- MLBusinessUser: Grants permission to list Cloudera AI Workbench for a given Cloudera Data Platform environment. MLBusinessUsers are able to only view applications deployed under the projects that they have been added to as a Business User.

Workbench resource roles

Workbench roles are for users who are granted access to only a single workbench.

- MLWorkspaceAdmin: Grants permission to manage all Cloudera AI workloads and settings inside a specific workbench. To perform resource role assignment, the IAMViewer role is also needed. A user with this role can administer the workbench, but is not able to utilize Cloudera Data Platform APIs that modify a workbench (for example, creating or upgrading workbenches).
- MLWorkspaceBusinessUser: Grants permission to view shared Cloudera AI applications inside a specific workbench.
- MLWorkspaceUser: Grants permission to run Cloudera AI workloads inside a specific workbench.

Using the workbench resource roles

A power user or account administrator must assign the first MLWorkspaceAdmin to a workbench. Subsequently, if the MLWorkspaceAdmin also has the IAMViewer role, they can assign resource roles to the workbench.

An MLAdmin (an environment resource role) is not automatically able assign workbench resource roles on the Manage access page. A role such as MLWorkspaceAdmin is needed to do this.

You can check the permissions for a given resource role by clicking the Information icon by each resource role shown in User Management, on the Resources tab for a user, or in a Cloudera Data Platform user profile.

Note: Any user that lists users or assigns resource roles also needs the account-level role of IAMViewer.

Related Information

Cloudera's Shared Responsibility Model

Business Users and Cloudera AI

A user is treated as a Business User inside of Cloudera AI if they are granted the MLBusinessUser role on the Environment of the given Cloudera AI Workbench. Inside the workbench, a Business User is able to access and view applications, but does not have privileges to access any other workloads in the workbench.

Logging in as a Business User

When you log in as a Business User, the only page you see is the Applications page. The page shows any applications associated with any projects that you have been added to as a Collaborator, even though you do not have rights to access the other assets associated with those projects.

In order for applications to appear in your view, contact the Project Owner to add you as a Collaborator to the project. If you have not been added to any projects, or none of the projects that you have been added to have applications, the Applications page displays the message, You currently don't have any applications.

Managing your Personal Account

You can edit personal account settings such as email, SSH keys and Hadoop credentials.

About this task

You can also access your personal account settings by clicking Account settings in the upper right-hand corner drop-down menu. This option will always take you to your personal settings page, irrespective of the context you are currently in.

Procedure

1. Sign in to Cloudera AI.
2. From the upper right drop-down menu, switch context to your personal account.
3. Click Settings.

   Profile

   You can modify your name, email, and bio on this page.

   Teams

   This page lists the teams you are a part of and the role assigned to you for each team.

   SSH Keys

   Your public SSH key resides here. SSH keys provide a useful way to access to external resources such as databases or remote Git repositories. For instructions, see *SSH Keys*.

Related Information

SSH Keys

Creating a Team

Users who work together on more than one project and want to facilitate collaboration can create a Team. Teams enable you to efficiently manage the users assigned to projects.

About this task

Team projects are owned by the team, rather than an individual user. Team administrators can add or remove members at any time, assigning each member different permissions. A team cannot be deleted and at least one member must be there in the team.

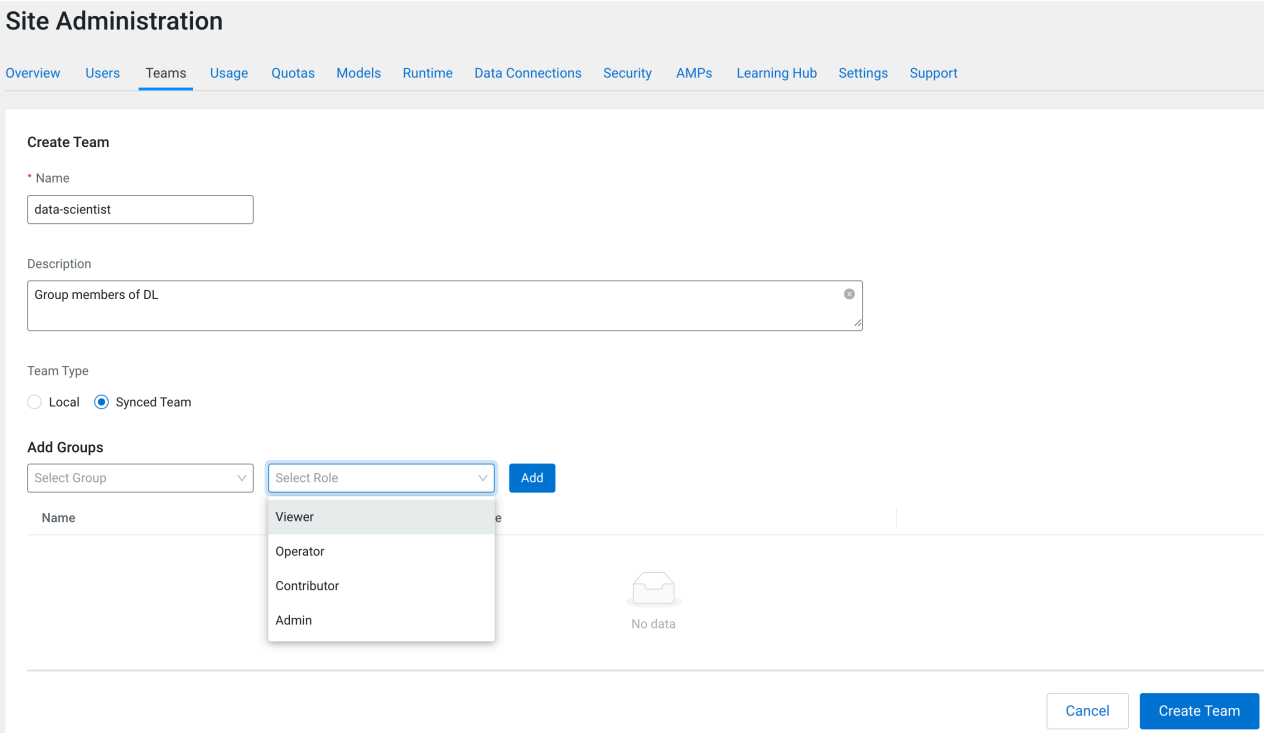

Procedure

1. In Site Administration Teams , select New Team.
2. Enter the name of the team.
3. Select Local or Synced Team.

   Cloudera Data Platform manages the member data of a Synced Team. The members and information about the members of a Local team is not managed by Cloudera Data Platform.
4. If Synced Team is selected, under Add Groups, select a group name and the role for the group and click Add. You can add multiple groups and roles using the Add option.

   Note: By default, each member will inherit the role of the groups they belong to. If a member belongs to multiple groups, their effective role in the team is the highest role assigned to the member (Viewer < Operator < Contributor < Admin).
5. Enter a Description, if needed.

6. Add or invite team members. Team members can have one of the following privilege levels:
   - Viewer - The Viewer has read-only access to team projects. The Viewere cannot create new projects within the team but can be added to existing ones.
   - Operator - The Operator has read-only access to team projects. Additionally, Operators can start and stop existing jobs in the projects that they have access to.
   - Contributor - The Contributor has write-level access to all team projects to all team projects with Team or Public visibility. The Contributor can create new projects within the team. They can also be added to existing team projects.
   - Admin - The Administrator has complete access to all team projects, can add new team members, and modify team account information. The creator of the team is assigned the Administrator privilege, and can also assign other team members the Administrator privilege. Each team must have at least one Administrator user.
7. Select Create Team.
8. Select Sync Teams to update the teams with information in the Management Console.

Managing a Team Account

Team administrators can modify account information, add or invite new team members, and view/edit privileges of existing members.

Procedure

1. From the upper right drop-down menu, switch context to the team account.
2. Click Settings to open up the Account Settings dashboard.
3. Modify any of the following settings:

   Profile

   Modify the team description on this page.

   Members

   You can add new team members on this page, and modify privilege levels for existing members.

   SSH Keys

   The team's public SSH key resides here. Team SSH keys provide a useful way to give an entire team access to external resources such as databases. For instructions, see *SSH Keys*. Generally, team SSH keys should not be used to authenticate against Git repositories. Use your personal key instead.

Related Information

SSH Keys

Cloudera AI

Data Visualization

Date published: 2020-07-16
Date modified: 2025-01-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Built-in Cloudera AI Visualizations..... 4
Simple Plots..... 4
Saved Images..... 4
HTML Visualizations..... 5
IFrame Visualizations..... 5
Grid Displays..... 6
Documenting Your Analysis..... 7

Cloudera Data Visualization for Cloudera AI.....7

Built-in Cloudera AI Visualizations

You can use built-in Cloudera AI tools to create data visualizations including simple plots, saved images, HTML and iFrame visualizations, and grid displays.

Simple Plots

Cloudera AI supports using simple plot to create data visualizations.

To create a simple plot, run a console in your favorite language and paste in the following code sample:

R

```
# A standard R plot
plot(rnorm(1000))
# A ggplot2 plot
library("ggplot2")
qplot(hp, mpg, data=mtcars, color=am,
facets=gear~cyl, size=I(3),
xlab="Horsepower", ylab="Miles per Gallon")
```

Python

```
import matplotlib.pyplot as plt
import random
plt.plot([random.normalvariate(0,1) for i in xrange(1,1000)])
```

Cloudera AI processes each line of code individually (unlike notebooks that process code per-cell). This means if your plot requires multiple commands, you will see incomplete plots in the workbench as each line is processed.

To get around this behavior, wrap all your plotting commands in one Python function. Cloudera AI will then process the function as a whole, and not as individual lines. You shall see then your plots as expected.

Saved Images

You can display images within your reports.

Use the following commands:

R

```
library("cdsw")

download.file("https://upload.wikimedia.org/wikipedia/commons/2/29/Minard.
png", "/cdn/Minard.png")
image("Minard.png")
```

Python

```
import urllib
from IPython.display import Image
urllib.urlretrieve("http://upload.wikimedia.org/wikipedia/commons/2/29/Minar
d.png", "Minard.png")
```

```
Image(filename="Minard.png")
```

HTML Visualizations

Your code can generate and display HTML in Cloudera AI.

To create an HTML widget, paste in the following:

R

```
library("cdsw")
html('<svg><circle cx="50" cy="50" r="50" fill="red" /></svg>')
```

Python

```
from IPython.display import HTML
HTML('<svg><circle cx="50" cy="50" r="50" fill="red" /></svg>')
```

Scala

Cloudera AI allows you to build visualization libraries for Scala using jvm-repr. The following example demonstrates how to register a custom HTML representation with the "text/html" mimetype in Cloudera AI. This output will render as HTML in your workbench session.

```
//HTML representation
case class HTML(html: String)
//Register a displayer to render html
Displayers.register(classOf[HTML],
  new Displayer[HTML] {
    override def display(html: HTML): java.util.Map[String, String] = {
      Map(
        "text/html" -> html.html
      ).asJava
    }
  })

val helloHTML = HTML("<h1> <em> Hello World </em> </h1>")

display(helloHTML)
```

IFrame Visualizations

Most visualizations require more than basic HTML. Embedding HTML directly in your console also risks conflicts between different parts of your code. The most flexible way to embed a web resource is using an IFrame.

Note:

Cloudera AI versions 1.4.2 (and higher) added a new feature that allowed users to HTTP security headers for responses to Cloudera AI. This setting is enabled by default. However, the X-Frame-Options header added as part of this feature blocks rendering of iFrames injected by third-party data visualization libraries.

To work around this issue, a site administrator can go to the Admin Security page and disable the Enable HTTP security headers property. Restart Cloudera AI for this change to take effect.

R

```
library("cdsw")
```

```
iframe(src="https://www.youtube.com/embed/8pHzROP1D-w", width="854px", heigh
t="510px")
```

Python

```
from IPython.display import HTML
HTML('<iframe width="854" height="510" src="https://www.youtube.com/embed/8
pHzROP1D-w"></iframe>')
```

You can generate HTML files within your console and display them in IFrames using the /cdn folder. The cdn folder persists and services static assets generated by your engine runs. For instance, you can embed a full HTML file with IFrames.

R

```
library("cdsw")
f <- file("/cdn/index.html")
html.content <- paste("<p>Here is a normal random variate:", rnorm(1), "</
p>")
writeLines(c(html.content), f)
close(f)
iframe("index.html")
```

Python

```
from IPython.display import HTML
import random

html_content  = "<p>Here is a normal random variate: %f </p>" % random.norma
lvariate(0,1)

file("/cdn/index.html", "w").write(html_content)
HTML("<iframe src=index.html>")
```

Cloudera AI uses this feature to support many rich plotting libraries such as htmlwidgets, Bokeh, and Plotly.

Grid Displays

Cloudera AI supports built-in grid displays of DataFrames across several languages.

Python

Using DataFrames with the pandas package requires per-session activation:

```
import pandas as pd
pd.DataFrame(data=[range(1,100)])
```

For PySpark DataFrames, use pandas and run df.toPandas() on a PySpark DataFrame. This will bring the DataFrame into local memory as a pandas DataFrame.

Note:

A Python project originally created with engine 1 will be running pandas version 0.19, and will not auto-upgrade to version 0.20 by simply selecting engine 2 in the project's Settings Engine page.

The pandas data grid setting only exists starting in version 0.20.1. To upgrade, manually install version 0.20.1 at the session prompt.

```
!pip install pandas==0.20.1
```

R

In R, DataFrames will display as grids by default. For example, to view the Iris data set, you would just use:

```
iris
```

Similar to PySpark, bringing Sparklyr data into local memory with as.data.frame will output a grid display.

```
sparkly_df %>% as.data.frame
```

Scala

Calling the display() function on an existing dataframe will trigger a collect, much like df.show().

```
val df = sc.parallelize(1 to 100).toDF()
display(df)
```

Documenting Your Analysis

Cloudera AI supports Markdown documentation of your code written in comments.

This allows you to generate reports directly from valid Python and R code that runs anywhere, even outside Cloudera AI. To add documentation to your analysis, create comments in Markdown format:

R

```
# Heading
# -------
#
# This documentation is **important.**
#
# Inline math: $e^ x$
#
# Display math: $$y = \Sigma x + \epsilon$$

print("Now the code!")
```

Python

```
# Heading
# -------
#
# This documentation is **important.**
#
# Inline math: $e^ x$
#
# Display math: $$y = \Sigma x + \epsilon$$

print("Now the code!")
```

Cloudera Data Visualization for Cloudera AI

Cloudera Data Visualization enables you to explore data and communicate insights across the whole data lifecycle by using visual objects. The fast and easy self-service data visualization streamlines collaboration in data analytics through the common language of visuals.

Using this rich visualization layer enables you to accelerate advanced data analysis. The web-based, no-code, drag-and-drop user interface is highly intuitive and enables you to build customized visualizations on top of your datasets, build dashboards and applications, and publish them anywhere across the data lifecycle. This solution allows for customization and collaboration, and it provides you with a dynamic and data-driven insight into your business.

Cloudera Data Visualization is integrated with Cloudera AI in all form factors. You can use the same visualization tool for structured, unstructured/text, and Cloudera AI analytics, which means deeper insights and more advanced dashboard applications. You can create native data visualizations to provide easy predictive insights for business users and accelerate production Cloudera AI workflows from raw data to business impact.

For more information, see the *Cloudera Data Visualization documentation*.

Related Information

Cloudera Data Visualization

Cloudera AI Workbenches

Date published: 2020-07-16
Date modified: 2025-01-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Provisioning Cloudera AI Workbenches....4

Configuring User access to Cloudera AI.... 7
Granting Cloudera Data Platform Users access to Cloudera AI Workbenches....7

Granting remote access to Cloudera AI Workbenches.... 8

Accessing Cloudera AI Workbenches via SOCKS Proxy.... 9

Monitoring Cloudera AI Workbenches....10

Suspend and resume Cloudera AI Workbenches....11

Backing up Cloudera AI Workbenches.... 12
Workbench backup and restore prerequisites.... 13
Backing up an Cloudera AI Workbench....17
Restore a Cloudera AI Workbench.... 18
Restoring to a different environment.... 19

Removing Cloudera AI Workbenches....19

Upgrading Cloudera AI Workbenches....20

Cloudera AI upgrades using Backup/Restore.... 21
Step 1 : Backing up the workbench....22
Step 2: Restoring into a new workbench with a different workbench URL/domain endpoint.... 25
Step 3: Delete the backed-up workbench.... 27
Step 4: Restore into a new workbench with same URL/domain endpoint as backed up workbench....27
Step 5: Delete the interim restored workbench.... 29
Frequently Asked Questions....29

Tagging disks to avoid garbage collection.... 29

Modify Instance Group Type....30

Provisioning Cloudera AI Workbenches

This topic describes how to provision Cloudera AI Workbenches.

Before you begin

The first user to access the Cloudera AI Workbench after it is created must have both the MLAdmin role and the EnvironmentAdmin account role assigned. See *Configuring User Access to Cloudera AI* and *Understanding account roles and resource roles* for information about this resource role.

Procedure

1. Log in to the Cloudera AI web interface.

   On Public Cloud, log in to https://console.cdp.cloudera.com using your corporate credentials or any other credentials that you received from your Cloudera Data Platform administrator.
2. Click Cloudera AI Workbenches.
3. Click Provision Workbench.
4. Fill out the following fields.
   - Workbench Name - Give the Cloudera AI Workbench a name. For example, *USER1_DEV*. Do not use capital letters in the workbench name.
   - Select Environment - From the dropdown, select the environment where the Cloudera AI Workbenches must be provisioned. If you do not have any environments available to you in the dropdown, contact your Cloudera Data Platform administrator to gain access.

     Note: You cannot choose an environment when the Environment or associated DataLake and FreeIPA is not in an available or running state.
   - Existing NFS - (Azure only) Enter the mount path from the environment creation procedure.
   - NFS Protocol version - (Azure only) Specify the protocol version to use when communicating with the existing NFS server.

5. Switch the toggle to display Advanced Settings.
   a) CPU Settings - From the dropdown, select the following:
      - Instance Type: You must select an instance type that is supported by Cloudera AI, or the associated validation check will fail (See *Other Settings*, below).
      - Autoscale Range
      - Root Volume Size: If necessary, you can also change the default size of the root volume disk for the nodes in the group.
   b) GPU Settings - Click the GPU Instances toggle to enable GPUs for the cluster, and set the following:
      - Instance Type: You must select an instance type that is supported by Cloudera AI, or the associated validation check will fail (See *Other Settings*, below).
      - Autoscale Range
      - Root Volume Size: If necessary, you can also change the default size of the root volume disk for the nodes in the group.

      Note: In addition to the CPU and GPU instances selected here, Cloudera AI also provisions two extra CPU instance groups to run infrastructure pods for the Cloudera AI Workbenches, as follows:

      AWS:

      - Cloudera AI infrastructure node group: m5.2xlarge, with an autoscale range of 2 to 3.
      - Platform infrastructure node group: m5.large, with an autoscale range of 2 to 4.

      Azure:

      - Cloudera AI infrastructure node group: Standard_D3s_v2, with an autoscale range of 2 to 3.
      - Platform infrastructure node group: Standard_D2s_v3, with an autoscale range of 2 to 4.

      These are not configurable by users.
   c) Kubernetes Config - Upload or directly enter the Kubernetes config information.
   d) Network Settings
      - Subnets for Worker Nodes: (AWS only) Optionally select one or more subnets to use for Kubernetes worker nodes.
      - Subnets for Load Balancer: Optionally select one or more subnets to use for the Load Balancer.
      - Load Balancer Source Ranges: (Azure only) Enter a CIDR range of IP addresses allowed to access the cluster.
        - If the Cloudera AI Workbench is provisioned with public access, enter the allowed public IP address range.
        - If the Cloudera AI Workbench is provisioned with private access, enter the allowed private IP address range.

        Note: When you change the Load Balancer Source Range setting, the changes are propagated to both the deployed Load Balancer in EKS and the underlying Security Group (SG).
      - Enable Fully Private Cluster: This Preview Feature provides a simple way to create a secure cluster. Only available in AWS environments in Cloudera Data Platform.
      - Enable Public IP Address for Load Balancer

        (AWS only) You can create a load balancer with a public IP address for the private cluster. This is useful in cases where there is no VPN between the Cloudera AI VPC and the customer network. In this case, the connection is over the internet.

        Note: In this network configuration, to use `kubectl` commands in the private cluster, you need to execute the commands in a network that is peered with the cluster VPN. Enabling the public IP address for the load balancer is not sufficient to allow `kubectl` commands to work.
      - Restrict access to Kubernetes API server to authorized IP ranges

You can specify a range of IP addresses in CIDR format that are allowed to access the Kubernetes API server. By default, the Kubernetes API services of Cloudera AI Workbenches are accessible to all public IP addresses (0.0.0.0/0) that have proper credentials.

To specify an address to authorize, enter an address in CIDR format (for example, 1.0.0.0/0) in API Server Authorized IP Ranges, and click the plus (+) icon. In this case, the API server is accessible by the user-provided address as well as control-plane-exit-ips over the public internet.

If the feature is enabled and no IP authorized addresses are specified, then the Kubernetes API server is only accessible by control-plane-exit-ips from the public internet.

Note: Both the Amazon EKS and Azure AKS have a quota or upper limit for the maximum number of public endpoint access CIDR ranges per cluster. See the Amazon EKS service quotas or the Azure AKS documentation for more details. When the feature is enabled, the Cloudera Control Plane exit IP addresses will be automatically added to the authorized IP ranges for accessing the Kubernetes API server for Cloudera Data Platform operations, which will use three CIDR blocks against the per-cluster limit.

- Use hostname for a non-transparent proxy

  Enter a CIDR range allowed for non-transparent proxy server access to the cluster.

e) Production Cloudera AI

- Enable Governance - Must be enabled to capture and view information about your Cloudera AI projects, models, and builds from Apache Atlas for a given environment. If you do not select this option, then integration with Atlas will not work.
- Enable Model Metrics - When enabled, stores metrics in a scalable metrics store, enables you to track individual model predictions, and also track and analyze metrics using custom code.

f) Other Settings

- Enable TLS - Select this checkbox if you want the workbench to use HTTPS for web communication.
- Enable public Internet access - When enabled, the Cloudera AI Workbench will be available on the public Internet. When disabled, it is assumed that connectivity is achieved through a corporate VPC.
- Enable Monitoring - Administrators (users with the MLAdmin role) can use a Grafana dashboard to monitor resource usage in the provisioned workbench.
- Skip Validation - If selected, validation checks are not performed before a workbench is provisioned. Select this only if validation checks are failing incorrectly.
- Tags - Tags added to cloud infrastructure, compute, and storage resources associated with this Cloudera AI Workbench.

  Note that these tags are propagated to your cloud service provider account. See *Related information* for links to AWS and Azure tagging strategies.
- Cloudera AI Static Subdomain - This is a custom name for the workbench endpoint, and it is also used for the URLs of models, applications, and experiments. You can create or restore a workbench to this same endpoint name, so that external references to the workbench do not have to be changed. Only one workbench with the specific subdomain endpoint name can be running at a time.

  Note: The endpoint name can have a maximum of 15 characters, using alphanumerics and hyphen or underscore only, and must start and end with an alphanumeric character.

6. Click Provision Workbench.

Results

It can take up to an hour for an Cloudera AI Workbench to be provisioned and installed. Once the status changes to show that the workbench has been successfully provisioned, click on the workbench name to go to the web application.

Note that the domain name for the provisioned workbench is randomly generated and cannot be changed.

What to do next

Grant users access to this Cloudera AI Workbench using the instructions at *Configuring User access to Cloudera AI*.

Related Information

Configuring User access to Cloudera AI
Understanding account roles and resource roles
Best Practices for Tagging AWS Resources
AWS EKS cluster endpoint access control
AWS Elastic Kubernetes Service endpoints and quotas
Create an AKS cluster with API service authorized IP ranges enabled
Use tags to organize your Azure resources and management hierarchy
Use a non-transparent proxy with Cloudera AI on AWS environments

Configuring User access to Cloudera AI

This topic describes how to grant users/groups access to an environment so that they can provision and/or list Cloudera AI Workbenches within that environment. The same users will also be granted Single Sign-on (SSO) access to the workbenches. In addition, if a user needs to provision, upgrade, or delete an Cloudera AI Workbench, they also need the account-level EnvironmentAdmin role assigned. For more information, see Understanding account roles and resource roles.

Note: This topic applies to public cloud releases.

Required Role: PowerUser

There are two Cloudera Data Platform user roles associated with the Cloudera AI service: MLAdmin and MLUser. A Cloudera Data Platform PowerUser will need to assign these roles to users who require access to the Cloudera AI service within an environment.

- MLAdmin - This role grants a Cloudera Data Platform user/group the ability to create and delete Cloudera AI Workbenches within a given Cloudera Data Platform environment. MLAdmins will also have Site Administrator level access to all the workbenches provisioned within this environment. That is, they can run workloads, monitor, and manage all user activity on these workbenches.
- MLUser - This role grants a Cloudera Data Platform user/group the ability to list Cloudera AI Workbenches provisioned within a given Cloudera Data Platform environment. MLUsers will also be able to run workloads on all the workbenches provisioned within this environment.

For instructions, see *Granting Cloudera Data Platform Users Access to Cloudera AI Service*.

Related Information

Understanding account roles and resource roles

Granting Cloudera Data Platform Users access to Cloudera AI Workbenches

This topic describes how to grant the MLAdmin and MLUser roles to users/groups that must be allowed to provision/list and access Cloudera AI Workbenches within a specific environment.

Note: This topic applies only to Public Cloud releases.

Procedure

1. Log in to the Cloudera AI web interface.

2. For a specific environment, grant the MLAdmin and MLUser roles to users/groups that must be allowed to provision/list and access Cloudera AI Workbenches within that environment. In addition, if a user needs to provision, upgrade, or delete an Cloudera AI Workbench, they also need the account-level EnvironmentAdmin role assigned. For more information, see Understanding account roles and resource roles.
   a) Click Environments.
   b) Search for the environment and navigate to the environment's Clusters page.
   c) Expand the Actions dropdown and click Manage Access.
   d) Search for the user or group that requires access to the Cloudera AI service in this environment and assign one of the following roles to each user/group:
      - MLAdmin - This role grants a Cloudera Data Platform user/group the ability to create and delete Cloudera AI Workbenches within a given Cloudera Data Platform environment. MLAdmins will also have Site Administrator level access to all the workbenches provisioned within this environment. That is, they can run workloads, monitor, and manage all user activity on these workbenches.

        OR
      - MLUser - This role grants a Cloudera Data Platform user/group the ability to list Cloudera AI Workbenches provisioned within a given Cloudera Data Platform environment. MLUsers will also be able to run workloads on all the workbenches provisioned within this environment.
   e) Click Update Roles.
   f) If necessary, search for and assign the EnvironmentAdmin role in the same way.

   Note: The first user to log in to an Cloudera AI Workbench must always be a Site Admin (that is, a user with the MLAdmin role assigned to them). If a user assigned the MLUser role attempts to access the workbench first, the web application will display an error.

Related Information

Understanding account roles and resource roles

Granting remote access to Cloudera AI Workbenches

This topic shows you how to allow specific users remote access to the underlying cluster that powers an Cloudera AI Workbench.

About this task

Note: This topic applies to AWS public cloud. On Azure public cloud, a user with the MLAdmin role can download the `kubeconfig` file, and this file alone grants access to any user who has it.

Required Role: MLAdmin

Before you begin

As part of this process, you will be required to enter the user's Amazon Resource Name (ARN). Make sure you have access to this information before you begin. Either get the ARN from the user OR look up a user's ARN in your AWS account. For the latter, go to your organisation's AWS AccountIdentity and Access Management (IAM)Users and lookup the user. The ARN is available on their Summary page.

If you are using the AWS CLI, you can run the following command to get the ARN:

```
aws sts get-caller-identity
```

```
#Sample output
{
"UserId": "ABCDE12345FGHIJKLMNO6789",
"Account": "888888888888",
```

```
"Arn": "arn:aws:iam::888888888888:user/<USERNAME>"
}
```

Procedure

1. Log in to the Cloudera AI web interface.
2. Click Cloudera AI Workbenches.
3. Click Actions to expand the dropdown menu.
4. Click Manage Remote Access.
5. Enter the user's ARN, or select the user's name.
6. Click Grant Access.

   To remove access for a user, in the Actions column, click Revoke Access next to the user's name.
7. Click Download Kubeconfig.

What to do next

Send the downloaded Kubernetes config file to the user who has been granted access. To be able to connect to the EKS cluster, they will need to have aws-iam-authenticator installed.

Related Information

Installing aws-iam-authenticator (AWS Documentation)

Accessing Cloudera AI Workbenches via SOCKS Proxy

This topic describes how to configure a SOCKS proxy to access Cloudera AI Workbenches on non-publicly routable VPCs. A SOCKS proxy server allows your web browser to connect directly and securely to your Cloudera AI Workbenches without exposing their ports outside the subnet.

About this task

Note: This topic applies to Public Cloud releases.

Procedure

1. In the non routable VPC, create an EC2 instance for your SOCKS server (for example, *MY-EC2-SOCKS-SERVER*) with a public IP and an SSH key-pair (for example, *MY-KEY-FILE.PEM*).

   Use the AWS documentation to create the EC2 instance: Getting Started with Amazon EC2 Instances

   Depending on whether you want multiple users to share the SOCKS server or have everyone create their own server, pick the SSH key pair for the instance accordingly. More information is available in the AWS documentation: Amazon EC2 Key Pairs.
2. Set up a SOCKS proxy server with SSH to access the EC2 instance, *MY-EC2-SOCKS-SERVER*.

   ```
   nohup ssh -i
           "my-key-file.pem" -CND 8157
           ec2-user@<public ip for my-ec2-socks-server> &
   ```

   - nohup (optional) is a POSIX command to ignore the HUP (hangup) signal so that the proxy process is not terminated automatically if the terminal process is later terminated.
   - *MY-KEY-FILE*.pem is the private key you used to create the EC2 instance where the SOCKS server is running.
   - C sets up compression.
   - N suppresses any command execution once established.

- D 8157 sets up the SOCKS 5 proxy on the port. (The port number 8157 in this example is arbitrary, but must match the port number you specify in your browser configuration in the next step.)
- ec2-user is the AMI username for the EC2 instance. The AMI username can be found in the details for the instance displayed in the AWS Management Console on the Instances page under the Usage Instructions tab.
- *<PUBLIC IP FOR MY-EC2-SOCKS-SERVER>* is the public IP address of the EC2 instance running the SOCKS server.
- & (optional) causes the SSH connection to run as an operating system background process, independent of the command shell. (Without the &, you leave your terminal open while the proxy server is running and use another terminal window to issue other commands.)

3. Configure Your Browser to Use the Proxy. This example uses Google Chrome.

By default, Google Chrome uses system-wide proxy settings on a per-profile basis. To get around that you can start Chrome using the command line and specify the following:

- The SOCKS proxy port to use (must be the same value used in step 1)
- The profile to use (this example creates a new profile)

This creates a new profile and launches a new instance of Chrome that does not interfere with any currently running instance.

- Linux

```
/usr/bin/google-chrome \
--user-data-dir="$HOME/chrome-with-proxy" \
--proxy-server="socks5://localhost:8157"
```

- MacOS

```
"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" \
--user-data-dir="$HOME/chrome-with-proxy" \
--proxy-server="socks5://localhost:8157"
```

- Windows

```
"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" ^
--user-data-dir="%USERPROFILE%\chrome-with-proxy" ^
--proxy-server="socks5://localhost:8157"
```

Results

You shall now be able to navigate to any Cloudera AI Workbench in the browser launched using SOCKS proxy.

When you connect to the Cloudera AI Workbench, the browser actually connects to the proxy server, which performs the required SSH tunneling.

Monitoring Cloudera AI Workbenches

This topic shows you how to monitor resource usage on your Cloudera AI Workbenches.

About this task

Cloudera AI leverages Prometheus and Grafana to provide a dashboard that allows you to monitor how CPU, memory, storage, and other resources are being consumed by Cloudera AI Workbenches. Prometheus is an internal data source that is auto-populated with resource consumption data for each workbench. Grafana is a monitoring dashboard that allows you to create visualizations for resource consumption data from Prometheus.

Each Cloudera AI Workbench has its own Grafana dashboard.

Before you begin

Required Role: MLAdmin

You need the MLAdmin role to view the Workbench details page.

Procedure

1. Log in to the Cloudera AI web interface.
2. Click Cloudera AI Workbenches.
3. For the workbench you want to monitor, click Actions Open Grafana .

Results

Cloudera AI provides you with several default Grafana dashboards:

- K8s Cluster: Shows cluster health, deployments, and pods
- K8s Containers: Shows pod info, cpu and memory usage
- K8s Node: Shows node CPU and memory usage, disk usage and network conditions
- Models: Shows response times, requests per second, CPU and memory usage for model replicas.

You might choose to add new dashboards or create more panels for other metrics. For more information, see the *Grafana documentation*.

What to do next

Note: Prometheus captures data for the previous two weeks.

Related Information

Grafana documentation

Monitoring and Alerts

Suspend and resume Cloudera AI Workbenches

Cloud consumption costs are a pain point for many public cloud users. The Cloudera AI Suspend feature allows users to scale down the Kubernetes pods running on Cloudera AI infra and CPU/GPU nodes for a given Cloudera AI Workbench. When the resume operation is performed on the suspended workbench, the suspended pods scale up.

About this task

A suspended Cloudera AI Workbench has all its autoscaling node groups, except the Platform Infra node group, shrunk to zero instances, thereby saving compute instance costs for the duration the workbench is suspended. However, Kubernetes pods running on Platform Infra nodes continue to run when a workbench is suspended.

When a workbench is suspended, you cannot access the workbench URL, and all associated models, applications, sessions, and jobs also become unavailable. The suspend operation terminates sessions and jobs, so the suspend should be started only after those operations have finished. When the workbench is resumed, models and applications automatically resume operation at the same URLs as before.

Note: Make sure that disks are tagged to avoid garbage collection during backup, restore, upgrade, or suspend operations on Cloudera AI Workbenches. For more information, see *Tagging disks to avoid garbage collection*.

Procedure

1. To suspend a Cloudera AI Workbench, in the workbenches UI, select Actions Suspend Workbench for the workbench to suspend. Then click OK to start the suspend process.
2. To resume a Cloudera AI Workbench, in the workbenches UI, select Actions Resume Workbench for the workbench to resume. Then click OK to start the resume process.

Related Information

Tagging disks to avoid garbage collection

Suspend Cloudera AI workbench API documentation

Backing up Cloudera AI Workbenches

Cloudera AI makes it easy to create machine learning projects, jobs, experiments, ML models, and applications in workbenches. The data and metadata of these artifacts are stored in different types of storage systems in the cloud .

You can backup an Cloudera AI Workbench, and restore it to a new workbench later. The backup preserves all files, models, applications and other assets in the workbench (files are not backed up by Cloudera AI automatically for external NFS-based workbenches). All workbench backups can be viewed in the Workbench Backup Catalog UI.

The Backup and Restore feature gives you the ability to backup all of your data (except files in external NFS-backed workbenches) to protect your machine learning artifacts against disasters. If your Cloudera AI Workbench is backed up, this feature lets you restore the saved data into a new Cloudera AI Workbench so that you can recover your Cloudera AI artifacts as they were saved in the desired backup. The Backup and Restore feature gives the administrator the ability to take “on-demand” backups of Cloudera AI Workbenches. Core services running in the workbench are shut down during the backup process to ensure consistency in the backup data. It is recommended that backups are taken during off-peak hours to minimize user impacts.

The time required to complete backing up a workbench depends on the amount of data to copy. The backup process copies data from both EBS volumes and EFS. In general, the time taken to backup EFS is more significant than for EBS. Due to the incremental nature of backups, the first backup always takes the longest amount of time. Subsequent backups should complete faster as they are built on top of the initial backup copy. For this reason, we recommend that Cloudera AI Workbenches be backed up regularly.

The time to backup EFS is highly dependent on the amount of data, and on the nature and number of files. It is also affected by available bandwidth in the AWS cloud backend. We have seen first-time backup of a 600 GB EFS file system taking around 10 hours. If you have much more than 600 GB on your EFS file system, the default backup timeout of 12 hours may not be long enough. In such cases, we recommend you take your first backup with a lower timeout, such as 2 hours. The Cloudera AI Control Plane may abort the backup due to the timeout expiry. However, the Control Plane does not cancel the underlying backup jobs. You can monitor these backup jobs on the AWS Backup console, and if all eventually complete successfully, you can initiate the backup operation again from the Cloudera AI Control Plane. This should complete in a relatively shorter time, and you will have a good backup copy to restore from if necessary.

There is currently no restriction on the number of backups one can take, and the backup snapshots are retained indefinitely in the backup service vault of the underlying cloud platform . Cloudera AI Workbench backup details are stored in the Workbench Backup Catalog UI in the Cloudera AI control plane, and these entries may be listed, viewed, deleted or restored as desired.

Restoring a backup creates a new Cloudera AI Workbench wherein the restored data is automatically imported. All the projects, jobs, applications, etc., that were in existence during the backup are automatically available in the new workbench. Restoring a Cloudera AI backup provisions a new cluster, and then launches restore jobs to create storage volumes from the backup snapshots. The restore process takes longer than a regular workbench provisioning operation due to the extra work in copying data from backup to the new storage volumes. While backups are incremental, restores are always full-copy restores. The time to restore is dominated by EFS restoration, which takes at least as long as the time to backup the file system. The restored workbench is always created with the latest

Cloudera AI software version, which may be different from the Cloudera AI version of the original workbench that was backed up.

Note: At this time, the Cloudera AI Workbench Backup and Restore feature is available on AWS, both through the UI and CLI. On Azure, this feature is only available through CLI.

Note: Make sure that disks are tagged to avoid garbage collection during backup, restore, upgrade, or suspend operations on Cloudera AI Workbenches. For more information, see *Tagging disks to avoid garbage collection*.

Note: A restored workbench from the workbench backup is considered a new separate workbench independent of the original workbench. Users' roles in the Control Plane from the original or backup workbench are not copied and assigned to the restored workbenches by the restore process to avoid security concerns and have to be moved after restore manually if intended.

Related Information

Tagging disks to avoid garbage collection

Workbench backup and restore prerequisites

To backup and restore workbenches, check that the following prerequisites are satisfied.

AWS backup service opt-in

Login to your AWS account and navigate to the AWS Backup Service console. Make sure the AWS region matches the region where you have your Cloudera AI Workbench. Click on Settings in the navigation pane, and in the Service opt-in table, ensure that EBS and EFS services are enabled for protection by the AWS Backup service, as shown here.

EBS	Enabled
EC2	Enabled
EFS	Enabled

For additional information about this feature, see July 2022: Cloudera Customer Advisory: The new feature Cloudera AI Backup and Restore on AWS requires changes to IAM permissions in their cross account roles (requires login).

AWS Cloudera Data Platform cross-account role permissions

1. Install the Backup IAM Policy. On the AWS console, navigate to the IAM service and click on Policies Create Policy . Click on the JSON tab, and replace the default text with the contents of the following JSON file. (Click here to download the file)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "backup:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "backup-storage:*",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticfilesystem:DescribeFilesystems",
```

```
                "elasticfilesystem:Backup",
                "elasticfilesystem:DescribeTags"
            ],
            "Resource": "arn:aws:elasticfilesystem:*:*:file-system/*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "ec2:DescribeSnapshots",
                "ec2:DescribeVolumes",
                "ec2:describeAvailabilityZones",
                "ec2:DescribeVpcs",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribePlacementGroups",
                "ec2:DescribeInstances",
                "ec2:DescribeTags"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteSnapshot"
            ],
            "Resource": "arn:aws:ec2:*::snapshot/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteSnapshot",
                "ec2:CreateSnapshot"
            ],
            "Resource": [
                "arn:aws:ec2:*::snapshot/*",
                "arn:aws:ec2:*:*:volume/*"
            ]
        },
        {
            "Action": [
                "ec2:DeleteSnapshot"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": [
                        "backup.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Action": [
                "tag:GetTagKeys",
                "tag:GetTagValues",
                "tag:GetResources"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
```

```
        {
            "Action": [
                "iam:ListRoles",
                "iam:GetRole"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::*:role/*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "backup.amazonaws.com"
                }
            }
        },
        {
            "Action": [
                "kms:ListKeys",
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:ListAliases"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "kms:CreateGrant"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "kms:EncryptionContextKeys": "aws:backup:backup-vault"
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                },
                "StringLike": {
                    "kms:ViaService": "backup.*.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "backup.amazonaws.com"
                }
            }
        }
    ]
}
```

Save this policy as cml-backup-policy.

2. Install the Restore Policy. On the AWS console, navigate to the IAM service and click on Policies Create Policy . Click on the JSON tab, and replace the default text with the contents of the following JSON file. (Click here to download the file)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVolume",
                "ec2:DeleteVolume"
            ],
            "Resource": [
                "arn:aws:ec2:*::snapshot/*",
                "arn:aws:ec2:*:*:volume/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*::snapshot/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSnapshots",
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:Restore",
                "elasticfilesystem:CreateFilesystem",
                "elasticfilesystem:DescribeFilesystems",
                "elasticfilesystem:DeleteFilesystem",
                "elasticfilesystem:TagResource"
            ],
            "Resource": "arn:aws:elasticfilesystem:*:*:file-system/*"
        },
        {
            "Effect": "Allow",
            "Action": "kms:DescribeKey",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "ec2.*.amazonaws.com",
```

```
                        "elasticfilesystem.*.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

Save this as cml-restore-policy.

3. Set up the Trust Relationship. AWS Backup service needs to be able to assume the AWS cross-account role that is used by the Cloudera Data Platform control plane to manage AWS cloud resources. To enable this, add the following trust relationship to your AWS cross-account role's Trust relationships (navigate to the IAM service console, then find your cross-account role by clicking on Roles).

```
{
      "Effect": "Allow",
      "Principal": {
        "Service": "backup.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
```

4. Attach the Backup and Restore policies to the Cross-Account Role. While still on the configuration page of your cross-account role in the IAM console, click on the Permissions tab. Click Attach policies to attach the cml-back up-policy and cml-restore-policy policies created above. This step ensures that the AWS Backup service will have the necessary permissions to call the EBS and EFS services on behalf of the cross-account role to manage backups.

Backing up an Cloudera AI Workbench

Backing up an Cloudera AI Workbench preserves all files, models, applications, and other assets in the workbench, although files in external NFS-backed workbenches are not backed up by Cloudera AI automatically.

Procedure

1. In the Workbenches UI, find the workbench to back up. The workbench must be in the Installation completed state, otherwise backup is disabled.
2. Enter the workbench, and manually stop all workloads (sessions, jobs, applications, and models).

   For external NFS backed workbenches, manually back up the configured external NFS data to another location. This manual backup of the NFS data will be used when this particular backup is restored in future. Ignore this step if the workbench is configured with internal NFS, as internal NFS data is backed up and restored automatically by Cloudera AI.
3. In the Actions menu for that workbench, select Backup Workbench.
4. In the Backup Workbench modal, enter a Backup Name to identify the workbench, and enter an appropriate timeout value.
5. Click Backup to start the process.

Results

The workbench shuts down, and the backup process begins. The workbench state changes to reflect the ongoing backup progress. If necessary, click Cancel to cancel the backup process. The backup process can take some time to complete, depending on the amount of data to copy.

Note: The default timeout is 12 hours. The estimated time to complete a backup (from the cloud provider) is now periodically added to the event logs.

What to do next

Monitoring event logs

You can monitor the progress of the backup process by checking the event logs. In the Actions menu for the workbench, click View Event Logs, and then on the Events & Logs tab, click View Event Logs again for the latest backup event.

When the backup process completes, the workbench enters the Installation completed state again.

If there were issues during backup, appropriate error messages will be displayed in the event logs. However the workbench will recover from failure and will be reverted back to the original state when backup was triggered.

Restore a Cloudera AI Workbench

Restoring a backup creates a new Cloudera AI Workbench, and recreates all of the projects, jobs, applications and so on in the original workbench.

About this task

Note: Restoring a workbench is a non-reversible operation. The restore process overwrites the existing workpace with older backup data. Any data in the running workbench that is not backed up will be lost. To save the current state, take a new backup before proceeding with the restore operation.

Procedure

1. In the Workbench Backups UI, find the workbench to restore. You can search for the workbench name or CRN. There can be multiple backups for a given workbench.
2. Enter the workbench, and manually stop all workloads (sessions, jobs, applications, and models).

   For external NFS backed workbenches, copy the manual backup of external NFS data (corresponding to this particular backup) to the configured external NFS export. Ignore this step if the workbench is configured with internal NFS, as internal NFS data is backed up and restored automatically by Cloudera AI.
3. Look for the backup to restore, and click Restore. The restore process starts, and the workplace state changes to Creating workbench.
4. Provision a new workbench that is in the same Cloudera Data Platform environment as the original workbench.

Results

The restore process can take some time, depending on the amount of data to copy. When it is complete, you can find the restored workbench in the Workbenches UI.

Note: If there is an issue during the restore process, the event log will show the relevant error messages. In case of error, the workbench will not recover from the failure automatically and will not revert back to the original state prior to the restore operation.

What to do next

Monitoring event logs

You can monitor the progress of the backup process by checking the event logs. In the Actions menu for the workbench, click View Event Logs, and then on the Events & Logs tab, click View Event Logs again for the latest backup event.

When the backup process completes, the workbench enters the installation completed state again.

If there were issues during backup, appropriate error messages will be displayed in the event logs. However the workbench will recover from failure and will be reverted back to the original state when backup was triggered.

Restoring to a different environment

A backup can be restored to a different Cloudera Data Platform environment, as long as it is within the same AWS account and region. Make sure the following requirements are met:

- Environment roles must be within the same AWS account and region.
- The target environment must have the necessary restore-related permissions, entitlements, and trust relationships.
- Within the environment where the backup is stored, the user must have the ml/listWorkspaceBackups permission.
- Within the environment where the workbench will be restored, the user must have the ml/createWorkspace permission.

Removing Cloudera AI Workbenches

This topic describes how to remove an existing Cloudera AI Workbench and clean up any cloud resources associated with the workbench. Currently, only Cloudera Data Platform users with both the MLAdmin role and the EnvironmentAdmin account role can remove workbenches.

Procedure

1. Log in to the Cloudera AI web interface.
2. Click Cloudera AI Workbenches.
3. Click on the Actions icon and select Remove Workbench.
   a) Remove EFS Storage - This option is enabled by default. If you want to retain project files on EFS, disable this property.
   b) Force Delete - This property is not required by default. You should first attempt to remove your workbench with this property disabled.

      Enabling this property will delete the workbench from Cloudera Data Platform but does not guarantee that the underlying cloud resources used by the workbench will be cleaned up properly. Go to your cloud service provider account to make sure that the cloud resources have been successfully deleted.

      When manually cleaning up resources, make sure that the following types of shared resources are not deleted:

      AWS:

      - VPCs
      - Subnets
      - Storage (S3 buckets and bucket entries)
      - AWS IAM roles

      Microsoft Azure

      - Virtual networks
      - Subnets
      - ADLS storage
      - Azure resource groups (RGs named <liftie-id> and MC_<liftie-id>_<azure-region>)

4. Click OK to confirm.

Note: On Azure public cloud, you also need to delete NFS storage after removing the workbench, if the NFS service is no longer needed.

Upgrading Cloudera AI Workbenches

This topic describes how to upgrade existing Cloudera AI Workbenches. Currently, only Cloudera Data Platform users with both the MLAdmin role and the EnvironmentAdmin account role can create, upgrade, or remove workbenches.

Existing Cloudera AI Workbenches periodically should be upgraded. Upgrading the workbench upgrades the Cloudera AI software version to the current version, and may also upgrade cluster software. In case the underlying Kubernetes software must be upgraded, a warning banner displays, notifying you that you shall upgrade the workbench promptly.

- During an upgrade, any running models and applications shut down, but they automatically restart after the upgrade is complete.
- To upgrade Kubernetes, only use the upgrade method provided in Cloudera AI. Do not upgrade Kubernetes directly in the cloud console or through the CLI. Follow the instructions here to upgrade Kubernetes. If there is some error, then repeat the instructions. This applies to both Microsoft Azure and AWS.
- You should back up your workbench before starting the upgrade. For more information, see Backing up Cloudera AI Workbenches.

When is an upgrade necessary?

Cloud service providers define their generally available version of Kubernetes based on their Kubernetes version support policies. For AKS refer to Supported Kubernetes versions in Azure Kubernetes Service (AKS) and for EKS refer to Amazon EKS Kubernetes release calendar.

Cloud service providers may have different deprecation policies for Kubernetes versions:

- For AWS deprecation policy, refer their FAQ section in Amazon EKS version support and FAQ.
- For Azure, refer to the Azure Kubernetes FAQ.

If any Kubernetes version used in your Cloudera AI Workbenches is deprecated by the cloud providers and Cloudera AI upgrades are enabled, the warning banner displays.

```
ACTION REQUIRED: A new Cloudera AI version is available and it is highly
 recommended to upgrade
    to the latest version as soon as possible. To perform an upgrade, select
 Upgrade
    Workbench from the Actions menu.
```

In order to avoid unplanned service interruption caused by the automatic Kubernetes upgrade by EKS and continue to receive support from AKS for your Cloudera AI Workbenches on Azure, it is important to make sure that your Cloudera AI Workbenches are using supported Kubernetes versions. Upgrading a Cloudera AI Workbench will automatically upgrade the Kubernetes to a supported version. We recommend our users to upgrade the Cloudera AI Workbenches promptly when the warning banner appears.

What type of upgrades does Cloudera AI Support?

In-place Cloudera AI upgrades

Upgrades are done in-place on the existing Cloudera AI Workbenches. This may involve a Kubernetes upgrade (if there is an upgrade available) followed by upgrading the Cloudera AI software.

Note: Make sure to backup your workbench before starting the upgrade process. For more information, see Backing up Cloudera AI Workbenches.

1. Log in to the Cloudera Data Platform web interface.
2. Click Cloudera AI Workbenches.
3. For a given workbench, click on the Actions icon and select Upgrade Workbench.
4. Click OK to confirm.

The upgrade process may take anywhere between two to four hours, approximately.

Upgrades through Cloudera AI backup & restore

If a Cloudera AI Workbench upgrade from a specific version could not be validated due to Kubernetes version deprecations on cloud providers or is deemed risky, in-place upgrades will be disabled for these versions.

In such cases, depending on the version of Cloudera AI either the upgrade button is disabled or the in-place upgrade pre-flight check will fail, with a failure message pops up that says:

```
In-place upgrades from <EXISTING_VERSION> are not supported. Follow the
 documentation for the backup based upgrade steps.
```

Here, *<EXISTING_VERSION>* is the version number of your workbench.

In this case, it is recommended to go with Backup/Restore to upgrade to the latest Cloudera AI version, essentially performing a workbench upgrade with all your previous data in place. Refer to Cloudera AI Upgrades using Backup/Restore for more information.

Note: Make sure that disks are tagged to avoid garbage collection during backup, restore, upgrade, or suspend operations on Cloudera AI Workbenches. For more information, see *Tagging disks to avoid garbage collection*.

Related Information

Backing up Cloudera AI Workbenches
Cloudera AI upgrades using Backup/Restore
Supported Kubernetes versions in Azure Kubernetes Service (AKS)
Azure Kubernetes FAQ
Amazon EKS Kubernetes release calendar
Amazon EKS version support and FAQ
Tagging disks to avoid garbage collection

Cloudera AI upgrades using Backup/Restore

Cloudera strongly recommends following the Cloudera AI release cadence by upgrading to every version soon after they are released. Following this process ensures that the Cloudera AI Workbench is up to date with the latest security and bug fixes as well to benefit from new feature development. This document will take you through some considerations to be aware of before performing an upgrade, options you have when performing the upgrade, and the steps to complete the upgrade.

Before you begin

If a Cloudera AI Workbench upgrade from a specific version could not be validated due to Kubernetes version EOL or is deemed risky, in-place upgrades will be disabled for these versions.

In-place upgrades will be disabled from Cloudera AI versions if the underlying Kubernetes versions are deprecated or going to be deprecated very soon. In such cases, depending on the version of Cloudera AI either the upgrade button is disabled or the in-place upgrade pre-flight check will fail, with a failure message pops up that says: In-place upgrades from <existing_version> are not supported. Follow the documentation for the backup based upgrade steps.

In this case, it is recommended to go with Cloudera AI Backup/Restore to upgrade to the latest Cloudera AI version, essentially performing a workbench upgrade with all your previous data in place. Since a restore always installs the latest Cloudera AI version, it essentially performs a workbench upgrade with all your existing workbench data intact. Backup/Restore is the recommended path to upgrade when a Cloudera AI Workbench cannot be reliably in-place upgraded from its current version.

Prerequisites

Backup/Restore on AWS

For AWS, Backup/Restore functionality is GA, and is usable from the UI. The documentation is already available in Backing up Cloudera AI Workbenches. Please see the documentation for prerequisites for using Backup/Restore on AWS.

Backup/Restore on Azure

Currently, Backup/Restore in Azure is available only through the Cloudera Data Platform CLI.

Additionally, the Backup/Restore feature does not perform a backup of NFS.

Steps to upgrade workbenches using Backup/Restore

There are five major steps to go through to upgrade older workbenches to the current version. Make sure to go through the following steps in order.

Note: Make sure that disks are tagged to avoid garbage collection during backup, restore, upgrade, or suspend operations on Cloudera AI Workbenches. For more information, see *Tagging disks to avoid garbage collection*.

Related Information

Upgrading Cloudera AI Workbenches
Tagging disks to avoid garbage collection

Step 1 : Backing up the workbench

After Step 1, Backing up the workbench, you shall follow steps 2 through 5 in order to restore the workbench.

Backing up AWS workbench

For information on backing up workbenches, see Backing up Cloudera AI Workbenches.

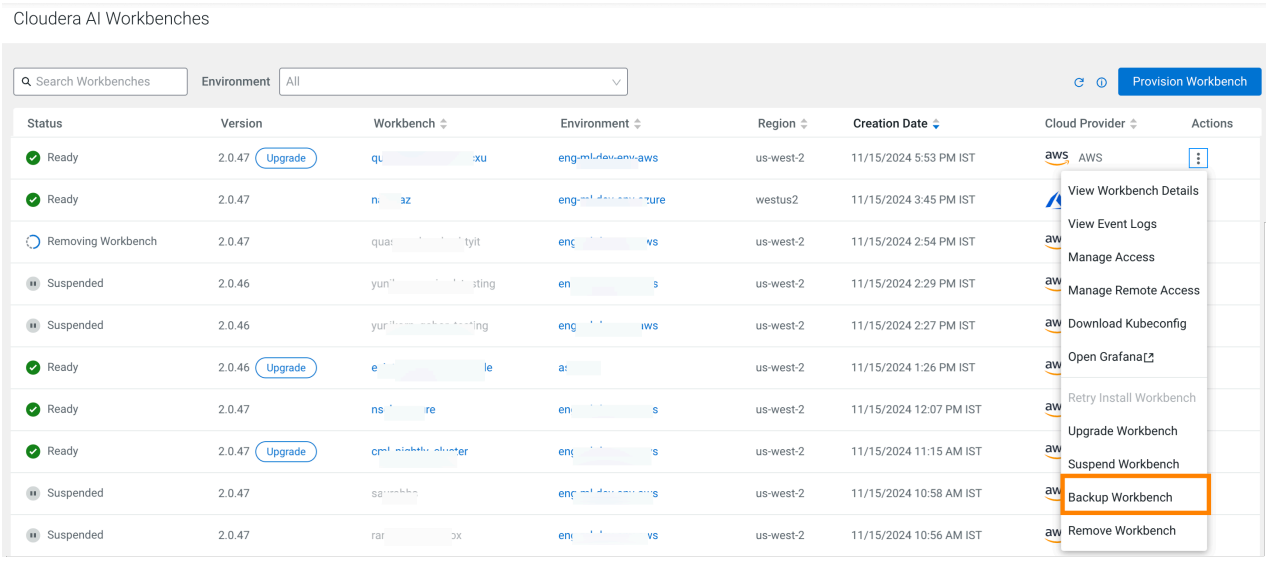

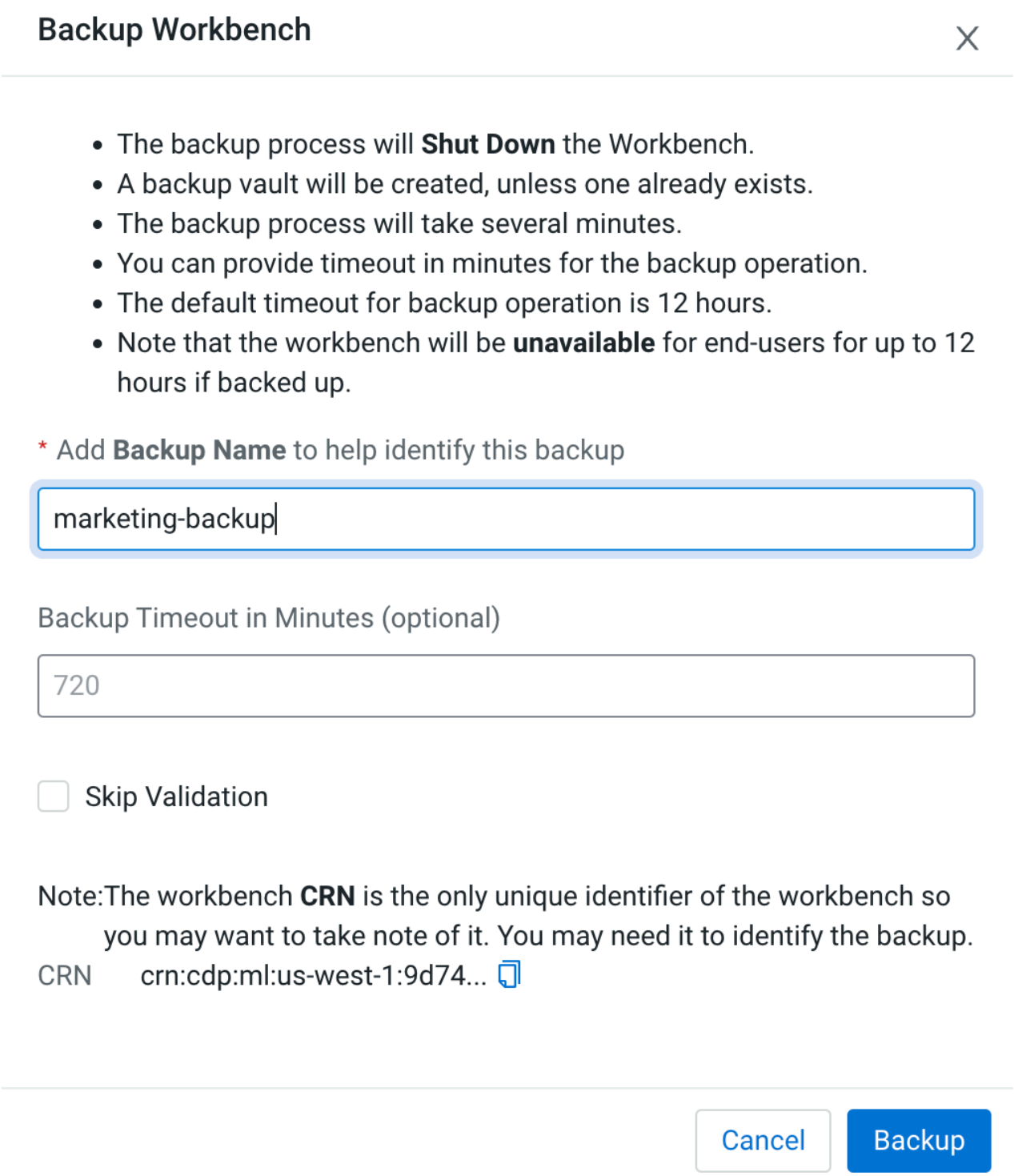

The time required to backup or restore AWS based workbenches mainly depends on the size of EFS (File System for projects storage) associated with the workbench. To get the EFS ID associated with the workbench, click on View Workbench Details from the UI and note the Filesystem ID. The size associated with the EFS can be retrieved from the AWS console using the Filesystem ID.

The Backup timeout for storage volumes (EFS and EBS) is by default set to 12 hours, but the user can customize the timeout. While clicking on the Backup Workbench , there will be an option to specify Backup Timeout (in minutes), to accommodate users with large EFS size.

Similarly, if the user suspects that the EFS is too large to be restored within the default 12 hours, custom restore timeouts can be set in the advanced settings during restoring an AWS workbench from a backup snapshot through the UI.

Tracking running backups on AWS

For AWS based backup jobs, Cloudera AI prints out backup job completion percentage for all backup jobs associated with a backup snapshot at an interval of 15 minutes as part of event logs. However, the completion percentage is an estimate returned by AWS APIs, and Cloudera AI just surfaces the same. Cloudera AI is not running any mechanisms/heuristics to calculate the completion percentage for a particular backup.

From our experience, we have seen that AWS-provided completion percentages can vary wildly, jump abruptly and can be downright misleading. Cloudera AI advises to take the percentage numbers with a grain of salt.

Canceling long running backups on AWS

Backups generally take a long time if you have lots of data in EFS. This is expected behavior. However if the backup is taking longer than expected, you can cancel the backup jobs from AWS dashboard and Cloudera AI will detect this in a while, and will fail the backup.

To cancel the corresponding AWS backup jobs from AWS dashboard:

- Go to AWS Backup Jobs Backup Jobs and identify the running backups associated with the backup snapshot. The backups should have started at approximately the same time you triggered the backup from the Cloudera AI console.
- Abort all such backup jobs.

You can retry the backups again using the above mentioned steps for backing up the workbench.

Backing up Azure workbench

Prerequisites

There are a few prerequisites to Azure Backup/Restore:

- If your environment is configured with a pre-existing resource group, then Cloudera AI backup service would use the same resource group for taking snapshots of Azure Disks. Else, please ensure that you have a resource group created in your Azure Account with the nomenclature `cml-snapshots-<azure_region>` . For example, if your Azure workbench resides in the `westus2` region, there should be a resource group present named `cml-snapshots-westus2`.
- Please refer to Azure documentation for roles needed to perform a backup: Use Azure role-based access control to manage Azure Backup recovery points.

Suspending the workbench

Suspend the workbench to ensure correctness of data in NFS on Azure during backup. Suspend the workbench by clicking on the Suspend Workbench option for the workbench and wait for the suspend operation to complete successfully.

Since the workbench is now in a suspended state, it is now guaranteed that no writes/mutations are happening on the NFS or Azure disks associated with the workbench.

Invoking Backup

Once you have the prerequisites sorted, please note the workbench CRN from the View Workbench Details page, and run the following command from Cloudera Data PlatformCLI to initiate workbench backup. Please replace the values in brackets with your own values.

```
$ cdp ml backup-workspace --workspace-crn <crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:workspace:792f8cfc-ba33-428d-9c80-b9bc6e799ce9> --backup-name <name-of-backup-for-upgrade>

    —-
    {
    "backupCrn": ""crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:workspace_backup:b6cee77e-9e38-4e30-9d72-481088f43de0""
    }
```

Please note the backup CRN returned from the CLI call, as it is the backup snapshot which will be used to restore into a new workbench.

Use these variables to restore into a new workbench:

- backupCRN: The CRN returned in the response of CLI call for backup-workbench
- existingNFS: The existing NFS server path can be retrieved from View Workbench Details Filesystem ID .
- existingNFSVersion: The existing NFS version can be retrieved from View Workbench Details NFS Protocol version .

Note: Please do not perform any operations with the existing NFS in use with the suspended workbench as we will attach the same NFS in the new (restored) workbench.

Step 2: Restoring into a new workbench with a different workbench URL/ domain endpoint

Restore into a new workbench with useStaticSubdomain set to false in the Cloudera Data Platform CLI. This brings up a workbench with a different URL/domain endpoint from the original workbench that was backed up. This step is needed to ensure that we can safely validate that restoration of the workbench is successful before executing Step 4 below to delete the original workbench. If any of the following steps fail, please contact your customer support representative.

Restore on AWS

For information on this step, see Restore an Cloudera AI Workbench.

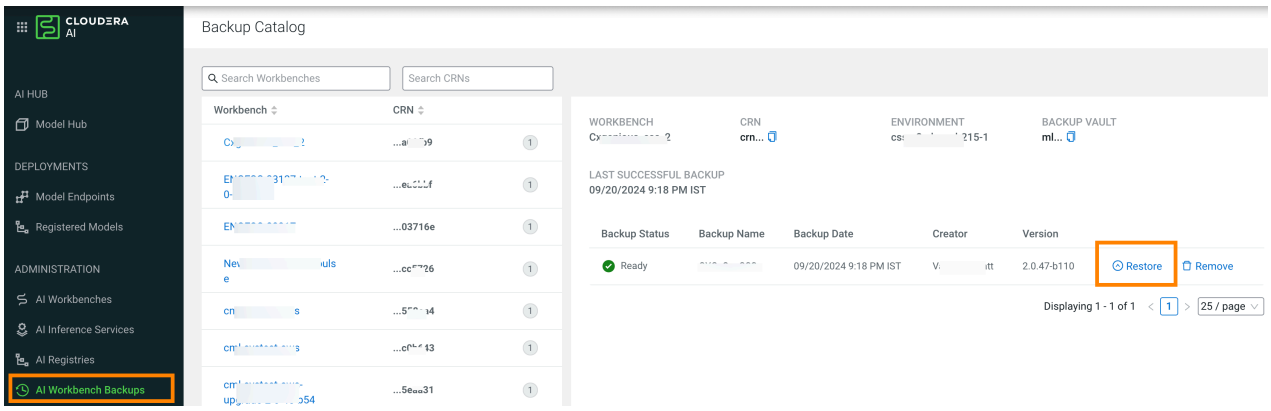

The UI for restore is quite similar to the Provision Workbench UI and shall be familiar.

Restore on Azure

To restore into a new workbench from the backup taken above, please run the following Cloudera Data Platform CLI command. The workbench provisioning parameters of the request needs to be configured according to your needs. Please ensure that no “write operations” are undertaken on this restored workbench since we will be using the same NFS in Step 5. This is to ensure that there is no state mismatch between the restored Azure disks and the NFS.

```
$ cdp ml restore-workspace --cli-input-json ‘{
  "newWorkspaceParameters": {
    "environmentName": "eng-ml-dev-env-azure",
    "workspaceName": "new-workspace",
    "disableTLS": false,
    "usePublicLoadBalancer": false,
    "enableMonitoring": true,
    "enableGovernance": true,
    "enableModelMetrics": true,
    "whitelistAuthorizedIPRanges": false,
    "existingNFS": "<existingNFS>",
    "nfsVersion": "<existingNFSVersion>",
    "provisionK8sRequest": {
      "instanceGroups": [
        {
          "instanceType": "Standard_DS3_v2",
          "rootVolume": {
            "size": 128
          },
          "autoscaling": {
            "minInstances": 1,
            "maxInstances": 10
          }
        }
      ],
      "environmentName": "eng-ml-dev-env-azure",
      "tags": [],
      "network": {
        "topology": {
          "subnets": []
        }
      }
    },
  },
  "backupCrn": "<backupCRN>",
  "useStaticSubdomain": false
}
’



—-
{
    "workspaceCrn": "crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb
73d:workspace:081ee5d2-4e82-487c-9404-2537a0ab4019"
}
```

Wait for the restore operation to succeed.

After restore, please login to the newly created workbench and verify that all projects from the original workbenches are available. Do not launch any sessions or applications, create new projects, or otherwise make any changes to the workbench, as that can make changes to the NFS file system that will be incompatible with what will be restored in step 4 below.

Related Information

Restore a Cloudera AI Workbench

Step 3: Delete the backed-up workbench

If Step 2 completes successfully, then delete the old (original) workbench.

After all is validated, and you confirm that the projects are in place, delete the original backed up workbench. To do so, from the UI, select Remove Workbench .

Step 4: Restore into a new workbench with same URL/domain endpoint as backed up workbench

If Step 2 and 3 complete successfully, then restore into a new workbench with the same URL/endpoint as the backed-up workbench.

Since a restored workbench is a brand new workbench with data from an older workbench, a restored workbench gets a new subdomain by default. This means that any endpoints (for models, applications, etc.) that you were using from the old workbench is not valid.

To maintain the endpoints that were configured with the older workbench, check the option useStaticSubdomain in the restore payload to provision the new restored workbench with the same URL as the older one. Additionally, Use Static Subdomain is also provided as a checkbox in the Restore UI.

Restore on AWS

To restore a workbench, see Restore as Cloudera AI Workbench. While restoring, please ensure that Use Static Subdomain is checked in the restore UI.

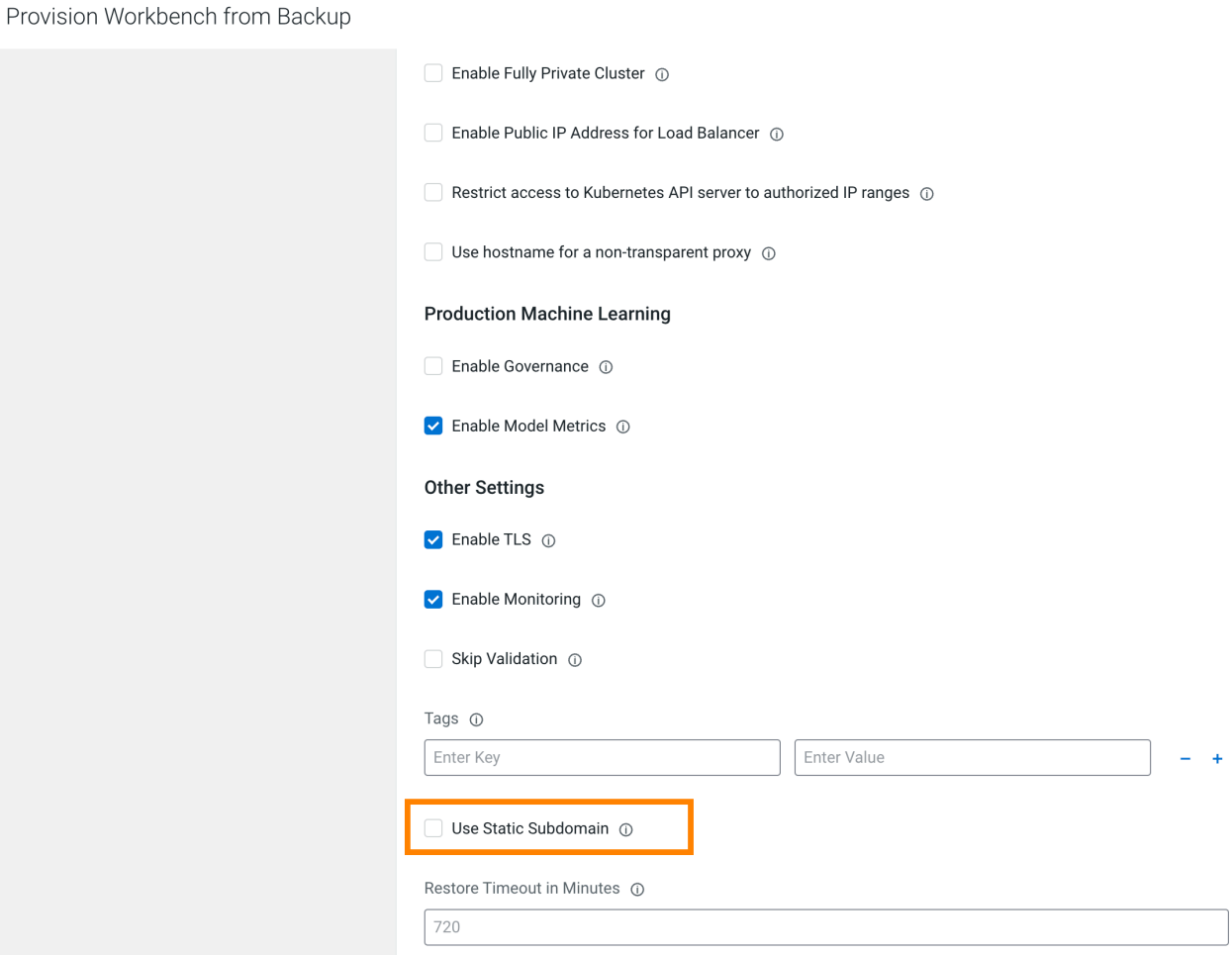

Restore on Azure

To restore into a new workbench from the backup taken above, please run the following Cloudera Data Platform CLI command. You need to tune various parameters of the request to suit workbench configuration needs.

```
$ cdp ml restore-workspace --cli-input-json '{
  "newWorkspaceParameters": {
    "environmentName": "eng-ml-dev-env-azure",
    "workspaceName": "new-workspace",
    "disableTLS": false,
    "usePublicLoadBalancer": false,
    "enableMonitoring": true,
    "enableGovernance": true,
    "enableModelMetrics": true,
    "whitelistAuthorizedIPRanges": false,
    "existingNFS": "<existingNFS>",
    "nfsVersion": "<existingNFSVersion>",
    "provisionK8sRequest": {
      "instanceGroups": [
        {
          "instanceType": "Standard_DS3_v2",
          "rootVolume": {
            "size": 128
          },
          "autoscaling": {
            "minInstances": 1,
            "maxInstances": 10
```

```
          }
        }
      ],
      "environmentName": "eng-ml-dev-env-azure",
      "tags": [],
      "network": {
        "topology": {
          "subnets": []
        }
      }
    },
  },
  "backupCrn": "<backupCRN>",
  "useStaticSubdomain": true
}
'



—-
{
    "workspaceCrn": "crn:cdp:ml:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb7
3d:workspace:081ee5d2-4e82-487c-9404-2537a0ab4019"
}
```

Related Information

Restore a Cloudera AI Workbench

Step 5: Delete the interim restored workbench

The upgraded workbench which was restored from the backup, in order to check the sanity of the restored workbench in Step 2, can now be safely deleted. Please identify the workbench from your control plane UI and delete the same.

Frequently Asked Questions

Some frequently asked questions about upgrading workbenches with the Backup/Restore feature.

How long does it take for a Cloudera AI Workbench to be backed-up and/ or restored?

Cloudera AI relies on cloud provider's native services for backup/ restore. Time consumed for backup and restore depends on multiple factors such as infrastructure, network latency, data size, file structure, number of files etc and can vary across workbenches. For internal test parameters, the backup of 600GB data took approximately 10 hours on AWS.

What happens to customizations done on Kubernetes Clusters during Restore?

Cloudera AI does not support applying customizations during Backup and Restore. All customizations will have to be applied through automation or manually post Cloudera AI Workbench Restore.

Tagging disks to avoid garbage collection

During backup, restore, upgrade, or suspend operations on Cloudera AI Workbenches, EBS or Azure disks can be left in a temporarily detached or unmanaged state when the associated EKS or AKS cluster is still running. If there is a garbage collection script running, and it is not properly configured, the disks used by the workbench can be deleted unintentionally. If the disk is not backed up, then the data will be lost.

Garbage collection scripts need to check for the following tags and ignore disks that are tagged with the corresponding values.

Cloud	Tag key	Tag value
Azure	k8s-azure-created-by	kubernetes-azure-dd
AWS	kubernetes.io/cluster/liftie-*	owned

Modify Instance Group Type

You can easily change the instance type of Cloudera AI Workbenches, which is beneficial for optimizing performance and cost.

Key Benefits of Modifying Instance Groups:

1. Scalability and Flexibility: Scale up or down to meet user workload needs, handle peak traffic or save costs during off-peak periods.
2. Cloud Provider Compatibility: We understand that cloud providers may retire or end-of-service (EOS) certain instance types. Our Modify Instance Group feature takes this into account and allows you to seamlessly adapt to changes in the cloud provider's offerings, ensuring your Cloudera AI Workbench stays up-to-date.

Currently, Instance Group modification is only supported for CPU and GPU Worker instance groups.

Note: As cloud platforms do not support heterogeneous instance types within a single instance group, the current Modify Instance Group Type workflow involves deleting the existing instance group and recreating it with the desired instance type. However, it's important to note that this process may disrupt user workloads running in the user namespace of the Cloudera AI Workbench, including user-created sessions, jobs, models, and applications.

Modify the Instance Group from the Cloudera Data Platform CLI

The following example shows how to modify the instance group from the command line.

```
cdp ml modify-cluster-instance-group --workspace-crn <workspace-crn>
   --instance-group-name <instance-group-name> –instance-type <instance-typ
e>
```

Modify the Instance Group from the UI

1. Go to the Workbench Details page.
2. Navigate to the Workbench Instances section on the Workbench Details page.
3. Click on the Edit button (as shown in Figure 1).
4. Choose the appropriate Instance Type from the list of available instance types (as shown in Figure 2).
5. Click on the Save button. A confirmation box will appear (as shown in Figure 3).
6. Click on the Ok button to modify the instance group with the chosen instance type.

Figure 1: Figure 1: Edit workbench Instance Group option in Workbench Details Page

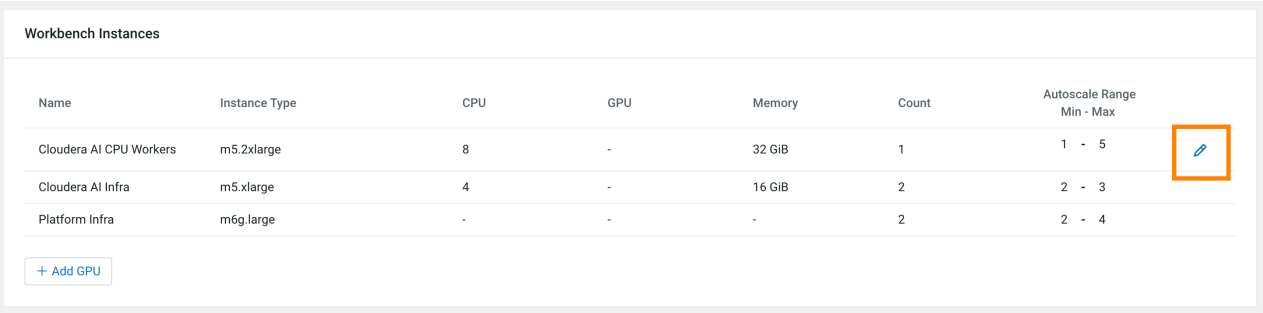

Figure 2: Figure 2: List of Instance Types for a Instance Group

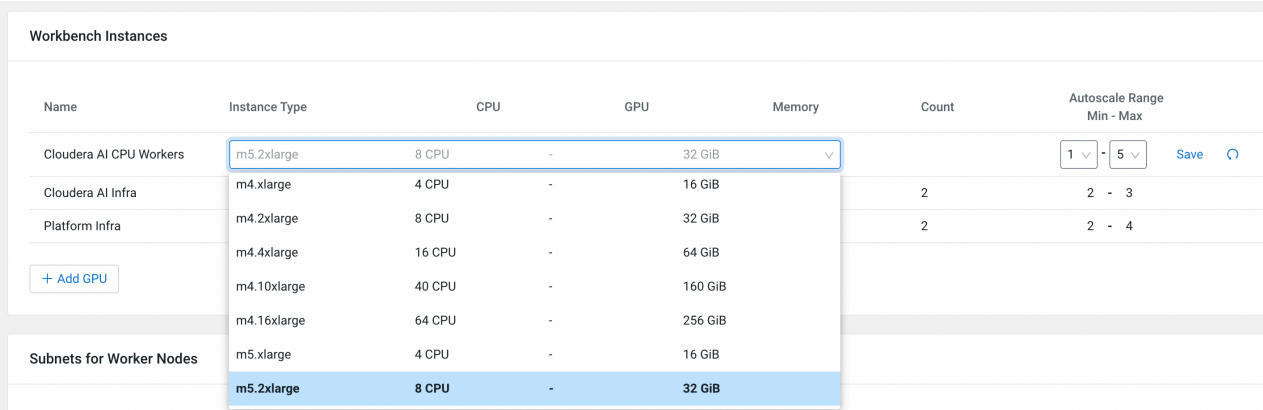

Figure 3: Figure 3: Confirmation box for Modify Instance Group Type

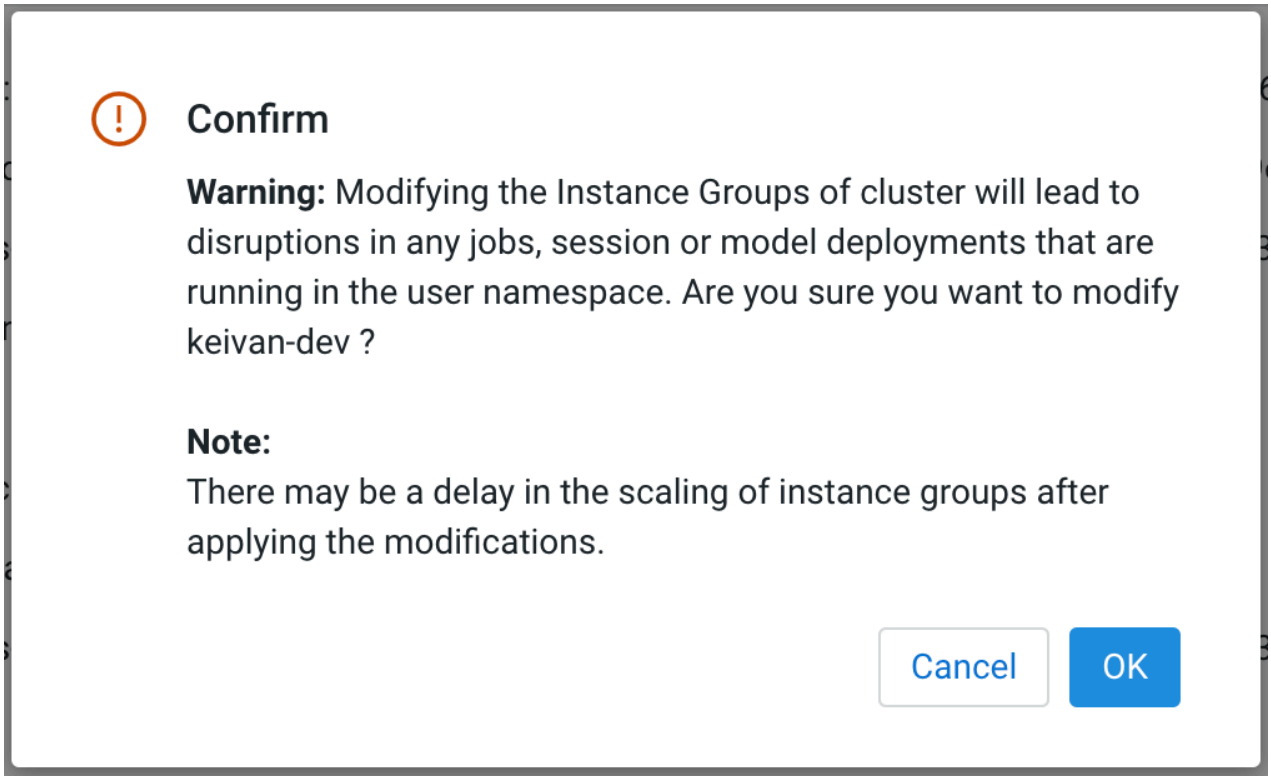

Note: During the modification process, there may be errors such as Bad Gateway or 404 Page not found. In this case, the applications can fail and may need to be restarted.

Cloudera Management Console

Cloudera Control Plane Auditing

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Auditing Cloudera Control Plane activity 4

Cloudera Control Plane auditing data model 4
- Cloudera service event 5
- API request event 6
- Interactive login event 6

Cloudera service event sources and names 7

Retrieving audit events 8

AWS setup for audit archiving 10
- Configuring audit event archiving through the UI 10
- Setting up an AWS policy and role to configure archiving using the CLI 12
- Creating an AWS credential for audit event archiving using the CLI 14
- Setting up audit archiving in AWS using the CLI 14

Azure setup for audit archiving 16
- Creating a storage account in Azure 16
- Creating a container in Azure 19
- Configuring audit event archiving through the UI 20
- Creating a Cloudera credential for audit archiving on Azure using the CLI 23
- Configuring audit archiving for Azure using the CLI 24

GCP setup for audit archiving 25
- Configuring audit event archiving through the UI 25
- Configure GCP audit event archiving using the CLI 27

Pull-based audit archiving 28

What archiving looks like 30

Auditing Cloudera Control Plane activity

Auditing is used to collect or log evidence of activity in a system that auditors can use to both track and analyze to answer questions such as: Who made a change to the system? When did a change happen? What exactly changed? Why was a change authorized?

Cloudera Control Plane auditing is based on the concept of an audit event. An audit event is a record of an audited action which is typically a change in the system that is important enough to keep a record of. However, even some read-only actions are audited, because it might be important to know who was able to see information in the system, and not just who could alter it.

Cloudera Control Plane auditing is scoped to actions that occur within the Cloudera Control Plane. Audit events are not collected from workload clusters; in fact, many Cloudera Control Plane audit events are collected without the need for any workload clusters to exist.

The auditing system initially stores generated audit events into a cloud provider managed database. After a specific amount of time, audit logs are exported to customer-managed storage, in their own cloud provider.

The audit records are kept in the system for a maximum of six months. After the six-month period, the records are removed from the internal storage of Cloudera regardless of the archive status. In case archiving is enabled and the access to the destination is lost, the archiving process will be retried for any records that are not archived until the access is restored or the six-month limit is reached. The auto-archiving process can be enabled or disabled as necessary. Once the archiving is enabled after disabling, all of the records that have not been archived will be archived regardless of age. The pull-based audit archiving can be used in case the automatic archiving is disabled.

Related Information

Cloudera Control Plane auditing data model
Cloudera service event sources and names
Retrieving audit events
AWS setup for audit archiving
Azure setup for audit archiving
What archiving looks like

Cloudera Control Plane auditing data model

Three are three categories of audit events.

Cloudera service events

For actions that a service within the Cloudera Control Plane undertakes. These actions are often as a result of human activity, but can also result from autonomous processes within the Cloudera Control Plane.

API request events

For calls to public API endpoints. These events are analogous to access logs kept by web servers. Because an API call often leads to actions within the Cloudera Control Plane, an API request often connects to one or more Cloudera service events.

Interactive login events

For logins to the Cloudera Control Plane.

All three categories of audit events share the following common fields:

Field name	Description	Example
version	Version of the audit event model	1.0.0

ID	Unique identifier for the event	a random UUID
source	Control plane service that submitted the event	iam
name	Type of action being audited	createGroupServiceEvent
timestamp	Time when the action occurred	2020-03-18T01:02:03Z
actor identity	Who initiated the action	see below
account ID	Identifier for account within which the action occurred	a UUID
request ID	Identifier for API request that led to the action	a UUID
result code	A string describing the result of the action, whether successful or not	INVALID_ARGUMENT
result message	A short message further describing the result	The group already exists

Each event source (service) defines its own event names. So, two sources may emit events with the same name, but for different actions. Events for an action within one source always use the same name.

An actor is an entity that causes an action to occur. In an audit event, an actor may be specified one of two ways.

Actor CRN

For a human actor, or for the special "internal" actor

Actor service name

For an antonymous process initiated by a control plane service (reserved for future use)

It is possible for the actor service name and the source in an audit event to be different. For example, a high-level service A may kick off an autonomous process that makes calls to another service B to make changes; audit events from that process would have actor service name A but source B.

Every call to the Cloudera Control Plane public API receives a request ID. The request ID propagates through the Control Plane to services that perform actions, and audit events from those services include the request ID. Therefore, a request ID can be used to tie together multiple audit events under the umbrella of a single API request.

Most audit events include result information, but sometimes that information may be missing. This indicates that the event source experienced a failure such that it could not submit the result information for an event after submitting its initial, known set of information.

For example, consider the action to grant a role to a user. The event source responsible for this action starts by submitting an audit event for role creation, including all the information known before attempting the action: the user CRN, the role CRN, and perhaps more. After role creation either succeeds or fails, the source appends result information to the event. However, if the source crashes, it cannot append the result information. When this happens, at least the initial, known information is recorded in an audit event.

Cloudera service event

A Cloudera service event contains additional fields.

Field name	Description	Example
details version	Schema version for the additional details content	2020-03-31
additional details	JSON containing additional event-specific information	{ "groupCrn" : "crn:..." }
resource CRNs	CRNs for an affected resource, if applicable (may be multiple values)	crn:altus:iam:us-west-1:altus:role:PowerUser

Usually there is much more information about an action than can fit into the common audit event fields. The additional details field holds that additional information in a structured way. Each event source defines the details structure for each type of event (by event source / name) it generates.

API request event

An API request event contains additional fields.

Field name	Description	Example
request parameters	JSON format of API request	{ "param1": … }
response parameters	JSON format of API response	{ "param1": … }
mutating	Boolean indicating if this API call changes resource data	true
apiVersion	Version of the API called	2020-03-31
source IP address	IP address from where the request originated	192.168.0.1
user agent	User agent string of the request	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:74.0) Gecko/20100101 Firefox/74.0

The request ID field can be used to tie an API request event to corresponding Cloudera service events, because Cloudera Control Plane services propagate the request ID initially generated for an API request through all resulting service activity.

Interactive login event

There are sufficient additional details about Cloudera Control Plane login events that they merit their own category. An interactive login event contains the following additional fields. Here, an identity provider is an authentication system outside of the Cloudera Control Plane that keeps identity information about users, such as Okta.

Field name	Description	Example
identity provider CRN	CRN of the identity provider as known by the control plane	crn:altus:iam:us-west-1:altus:samlProvider:cloudera-sso
identity provider session ID	identifier assigned to the login session by the identity provider	TBD
identity provider user ID	identifier of the user as stored in the identity provider	spongebob@cloudera.com (for Cloudera SSO, email is used)
email	email address of the user logging in	spongebob@cloudera.com
first name	first (given) name of the user logging in	Spongebob
last name	last name (family name / surname) of the user logging in	Squarepants
account admin	a Boolean flag indicating if the login is for an administrative user	true
groups	names of groups to which the user belongs in their account	TBD
source IP address	IP address of the user logging in	192.168.0.1
user CRN	CRN of the user logging in	crn:...

The account admin flag is only available if login succeeds. The user CRN is not known until login is attempted, and the CRN is not always recorded when login fails (for example, if the user's account cannot be determined by the Cloudera Control Plane). Remaining fields are filled in before the user login attempt, and so should be present in every event.

Cloudera service event sources and names

Cloudera defines many service event sources and names.

The defined Cloudera service event sources and names will expand in the future.

Event source	Event name	Action
iam	AssignRoleServiceEvent	Assignment of a role to a user
	CreateUserServiceEvent	Creation of a new user in the control plane
	CreateGroupServiceEvent	Creation of a new group
	DeleteGroupServiceEvent	Deletion of an existing group
	InteractiveLogoutEvent	Interactive logout by a user in the control plane.
	UnassignRoleServiceEvent	Removal of a role from a user
datahub	CreateDataHubCluster	Creation of a Cloudera Data Hub cluster
	DeleteDataHubCluster	Deletion of a Cloudera Data Hub cluster
	InstanceDeleteDatahubCluster	Deletion of a Cloudera Data Hub cluster instance
	MaintainDatahubCluster	Upscaling/downscaling/starting/stopping a Data Hub cluster
	ManualRepairDatahubCluster	Manual repair of a Cloudera Data Hub cluster
	RetryDatahubCluster	Retrying a Cloudera Data Hub cluster
	StartDatahubCluster	Starting of a Cloudera Data Hub cluster
	StopDatahubCluster	Stopping of a Cloudera Data Hub cluster
datalake	CreateDatalakeCluster	Creation of a Data Lake cluster
	DeleteDatalakeCluster	Deletion of a Data Lake cluster
	InstanceDeleteDatalakeCluster	Deletion of a Data Lake cluster instance
	ManualRepairDatalakeCluster	Manual repair of a Data Lake cluster
	ResizeDatalakeCluster	Resizing of a Data Lake cluster
	RetryDatalakeCluster	Retrying of a Data Lake cluster
	StartDatalakeCluster	Starting of a Data Lake cluster
	StopDatalakeCluster	Stopping of a Data Lake cluster

The schemas for the additional details JSON for each event are defined in CDP Control Plane Audit Event Details Documentation.

Cloudera Data Warehouse audit events

You can retrieve the following audit events using the CDP CLI that occur in Cloudera Data Warehouse:

Event source	Event name
dw	CreateEnvironment
	DeleteEnvironment
	UpdateEnvironment
	UpgradeEnvironment
	CreateDbCatalog
	DeleteDbCatalog
	StartDbCatalog
	StopDbCatalog
	UpdateDbCatalog
	CloneDbCatalog
	UpgradeDbCatalog
	CreateHiveVirtualWarehouse
	DeleteHiveVirtualWarehouse
	StartHiveVirtualWarehouse
	StopHiveVirtualWarehouse
	UpdateHiveVirtualWarehouse
	CloneHiveVirtualWarehouse
	UpgradeHiveVirtualWarehouse
	CreateImpalaVirtualWarehouse
	DeleteImpalaVirtualWarehouse
	StartImpalaVirtualWarehouse
	StopImpalaVirtualWarehouse
	UpdateImpalaVirtualWarehouse
	CloneImpalaVirtualWarehouse
	UpgradeImpalaVirtualWarehouse

For more information about experimental CLI commands for Cloudera Data Warehouse, go to Version Mapping. Click the CDP CLI Reference link for your Cloudera Data Warehouse version. Scroll to Available Commands, and click dw.

Retrieving audit events

The Cloudera Control Plane provides access to view/retrieve audit events and archive history in two ways: through the Management Console UI and through the Cloudera API offered by the Cloudera Control Plane audit service.

Required role: PowerUser

Cloudera Management Console UI

With the required permissions, you can view both the audit events and the audit archive history in the Cloudera Management Console UI.

A list of both the audit events and the archive history are accessible from the main left-hand navigation menu under Audit:

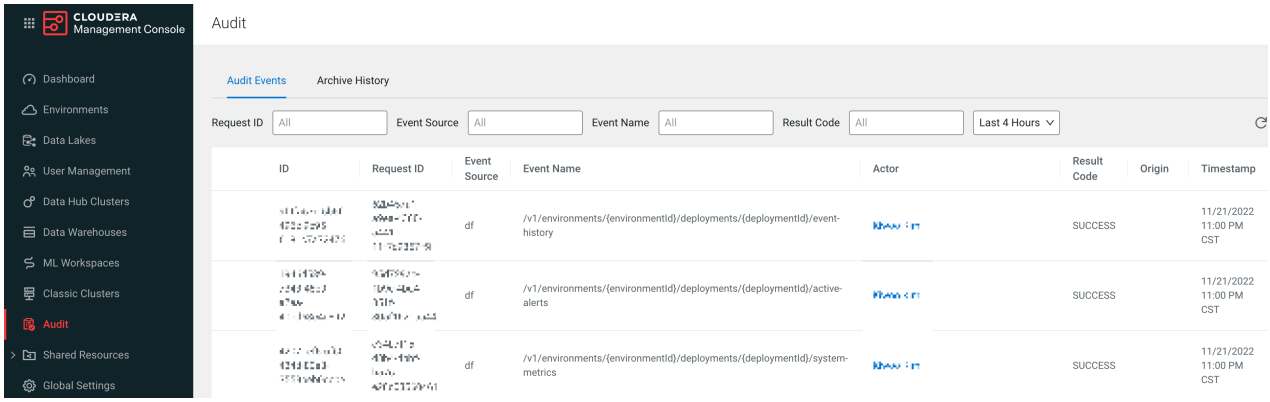

The **Audit Events** page contains a record of all the events that generate an audit record, along with their name, origin, source, and timestamp.

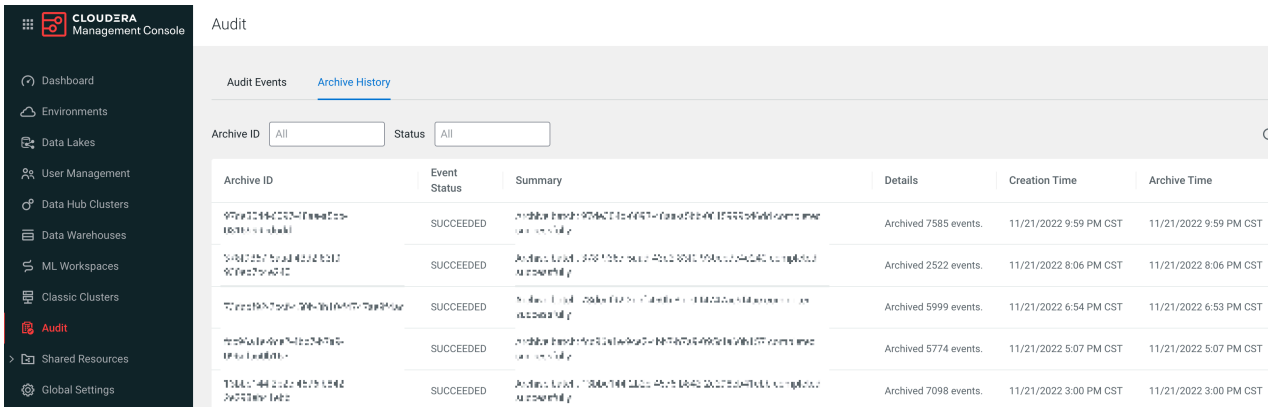

The **Archive History** page contains a record of each archive batch that is sent to cloud storage, along with details related to that batch, such as the status, number of events archived, and time of both creation and archive. The archive history gives greater visibility into the audit process and will alert you if an archive run fails to export to cloud storage.

Public listing API

The Cloudera Control Plane audit service offers a basic API endpoint for listing audit events. The API accepts the following parameters:

Parameter name	Description	Required?
from timestamp	Beginning of time range within which to retrieve audit events	Y
to timestamp	End of time range within which to retrieve audit events	Y
request ID	Request ID to filter on	N
event source	Event source to filter on	N
page size	Number of audit events to return in one response; maximum and default value = 50	N

page token	Opaque value from previous listing call used to retrieve the next page	N

The result of a listing call is a set of audit events, possibly along with a page token value. In order to get a listing of the next page of audit events, pass the token value in another call, leaving other parameters the same.

Calls to the listing API endpoint are protected as follows:

- The caller must have the audit/listAuditEvents right.

When an account is first created in the control plane, it is not configured with the information needed to archive audit events to cloud storage. You can use the control plane without configuring archiving, but audit events are subject to purging after 90 days. So, it is important to configure archiving to avoid data loss.

Access to the listing API endpoint is strictly throttled, to prevent the critical audit service from becoming too busy to accept audit events from Cloudera Control Plane services. The primary means of retrieving audit events is intended to be through cloud storage.

CDP CLI Syntax

To call the API endpoint for listing audit events, use a command line like the following, which includes examples of all of the required options:

```
cdp audit list-events
  --from-timestamp 2020-03-01T00:00:00Z
  --to-timestamp 2020-04-01T00:00:00Z
```

To additionally filter by request ID, use the --request-id option. To additionally filter by event source, use the --event-source option. Paging options work the same way as with other CDP CLI commands.

You can also use the CDP CLI to view the audit archive history. The following command returns the most recent audit archive exports:

```
cdp audit list-recent-archive-runs
```

AWS setup for audit archiving

While the auditing system stores generated audit events initially into a cloud provider managed database, after a specific amount of time, audit logs are exported to customer-managed storage, in their own cloud provider. As a result, you need to set up audit archiving for your platform.

Configuring audit event archiving through the UI

To configure archiving, you must create an AWS IAM role specifically for audit event archiving. Then create an audit credential using the cross-account role ARN.

Before you begin

In AWS, create a new S3 bucket or designate an existing bucket for audit archiving. Be sure to block all public access. The audit logs are written to the S3 bucket even without public access as the writing process is based on the AWS role and not on the internet access.Audit events will be archived under the /cdp/cp folder, which will be created automatically by Cloudera.

Required Role: PowerUser

Procedure

1. Log in to the Cloudera interface.

2. In the left-side navigation menu, click Global SettingsAudit Data Configuration and then click Create.
3. Select the AWS icon.

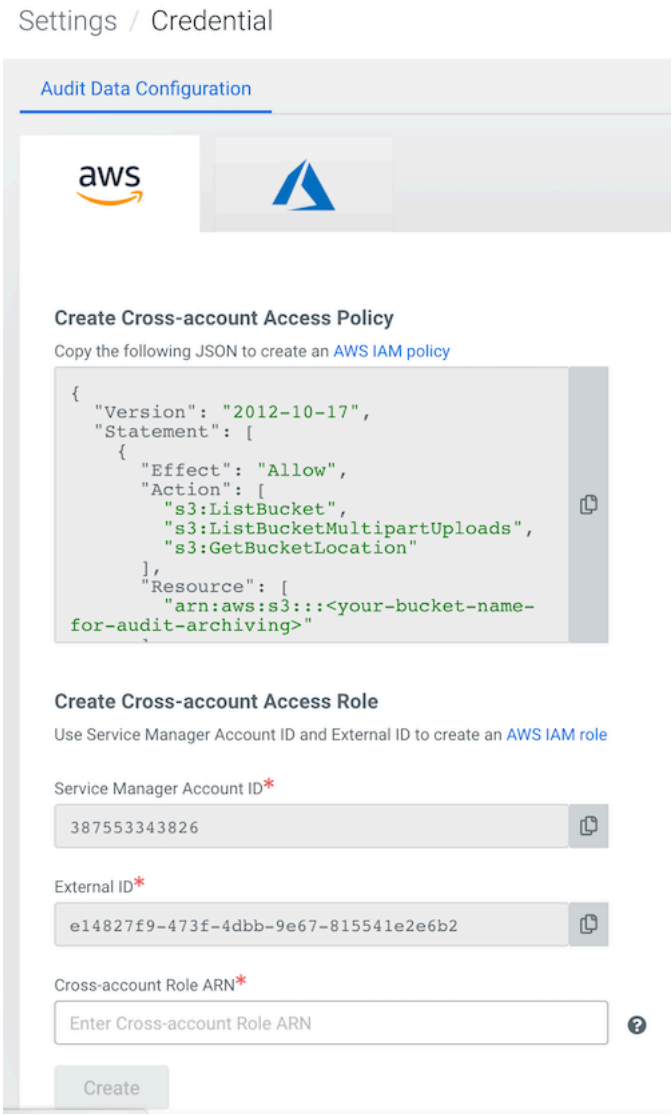

4. Copy the cross-account access policy provided to you in the **Create Cross-account Access Policy** field, substituting your bucket name where indicated.

5. From the AWS account hosting the bucket, create an IAM role for another AWS account, in this case, for the account running the Cloudera Control Plane.

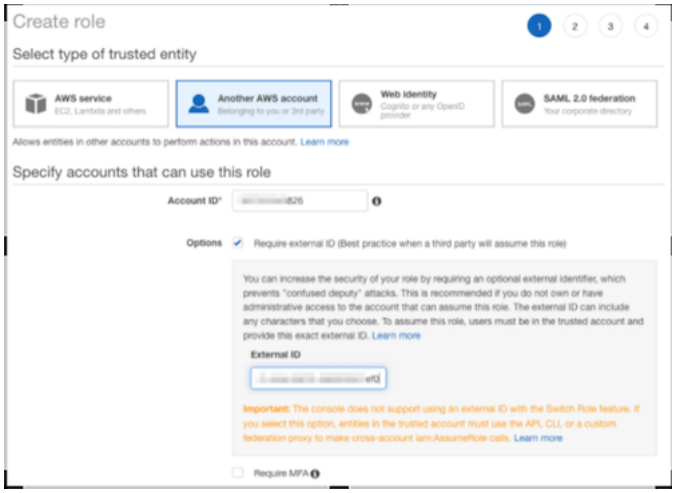

Include the policy that you copied as the only one in the role.

For detailed instructions on creating an AWS IAM cross-account role, see Create a cross-account IAM role, starting with "1. Log into the AWS Management Console." Although the audit event archiving credential requires a unique policy and cross-account Role ARN, the process is largely the same as creating a role-based credential during environment registration.

6. When you finish creating the new cross-account policy and role in AWS, copy the Role ARN from the Role **Summary** page in the AWS Management Console and return to Cloudera. Paste the Role ARN into the Cross-account role ARN field.
7. Click Create.
8. Configure the audit data location with the name of the S3 bucket that you designated as the audit archive bucket.
9. Select the AWS region where storage services should be accessed.
10. Use the toggle button to enable or disable audit log export to the configured storage location.
11. Click Save.

Results

Audit event archiving configuration is complete.

Setting up an AWS policy and role to configure archiving using the CLI

To configure archiving through the CDP CLI, you must first set up a policy and cross-account role in AWS IAM and obtain the cross-account role ARN.

Before you begin

In AWS, create a new S3 bucket or designate an existing bucket. Be sure to block all public access. The audit logs are written to the S3 bucket even without public access as the writing process is based on the AWS role and not on the internet access. Audit events will be archived under the /cdp/cp folder, which will be created automatically by Cloudera.

Procedure

1. From the AWS account hosting the bucket, create an IAM policy that permits read and write access to the bucket.

   For example, substituting your bucket name where indicated:

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "cdpauditb",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::<your-bucket-name-for-audit-archiving>"
            ]
        },
        {
            "Sid": "cdpauditbo",
            "Effect": "Allow",
            "Action": [
                "s3:ListMultipartUploadParts",
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::<your-bucket-name-for-audit-archiving>/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeRegions"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

2. From the AWS account hosting the bucket, create an IAM role for another AWS account, in this case, for the account running the Cloudera Control Plane.

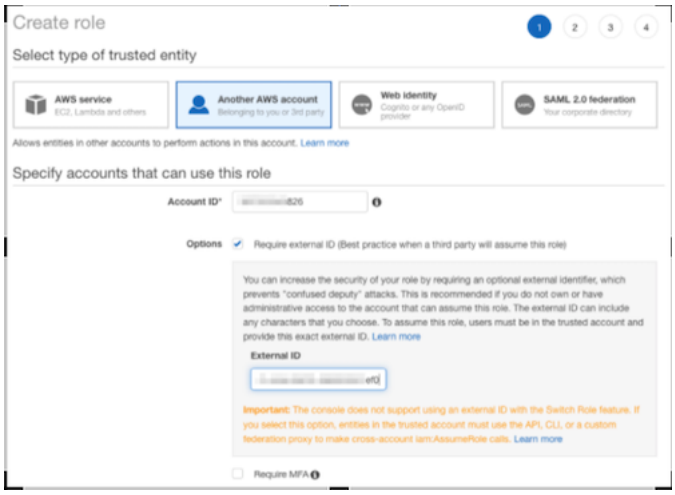

Include the policy you just created as the only one in the role.

For detailed instructions on creating an AWS IAM policy and cross-account role, see Create a cross-account IAM role, starting with "1. Log into the AWS Management Console." Although the audit event archiving credential

requires a unique policy and Role ARN, the process is largely the same as creating a role-based credential during environment registration.

3. To finish creating the role you will need the account ID (service manager account ID) and external ID. Run the following CDP CLI command:

```
cdp environments get-audit-credential-prerequisites \
  --cloud-platform AWS
```

Note: The control plane uses a different external ID for each account or tenant. So, be sure to look up the account ID and external ID using a user in the account which is being configured.

4. When you finish the role creation process, copy the Role ARN from the role **Summary** page in the AWS Management Console. You will need it to create the audit event archiving credential in the next task.

What to do next

Follow the process in the next topic, *Creating an AWS credential for audit event archiving using the CLI*, to create the audit archive credential.

Creating an AWS credential for audit event archiving using the CLI

To configure archiving, you must set up a credential for audit event archiving and then configure Cloudera.

Before you begin

You must have the cross-account role ARN obtained in the previous task to finish setting up an AWS credential for audit event archiving.

About this task

Complete this task before you configure audit archiving. The audit credential that you create here is not tied to an environment, and exists outside of any environment, like the control plane itself. The associated role / permissions require write access to the storage location, including the ability to create files and folders.

Required Role: PowerUser

Procedure

1. Use the following commands to create a new audit credential:

```
cdp environments set-aws-audit-credential \
    --role-arn arn:aws:...
```

The role-arn information was provided when you created an IAM role.

You can view audit credentials with this command:

```
cdp environments list-audit-credentials
```

2. Make note of the credential name created by the command.

What to do next

Proceed to *Setting up audit archiving in AWS using the CLI* to complete the audit archiving setup.

Setting up audit archiving in AWS using the CLI

After you set up a credential for audit event archiving, you need to configure Cloudera. This process is not necessary if you used the UI to create the audit archiving credential and configure the data storage location.

About this task

Archiving configuration includes the following fields.

Field name	General description	Details
storage location	Where in cloud storage to save audit events	S3 bucket name. Use only the bucket name and not the "s3://" prefix or any sub-folders.
credential	Name of the credential to use when writing to cloud storage	credential name as saved in the control plane
enabled	A Boolean indicating whether archiving is enabled	true
storage region	Region where storage services should be accessed	AWS region, e.g., us-west-2

Create the credential for audit event archiving before configuring archiving itself. The credential is not tied to an environment, and exists outside of any environment, like the control plane itself. The associated role / permissions require write access to the storage location, including the ability to create files and folders.

The storage region is the region where the audit service (the control plane) accesses the cloud provider's storage service. For best results, this should be the same as the region where the control plane is running. Ideally, the bucket should be created in the same region.

Required Role: PowerUser

Procedure

1. To call the API endpoint for configuring archiving, run the cdp audit configure-archiving command. To begin, include the optional --verify-only flag, which the service uses to first verify that the configuration works:

   ```
   cdp audit configure-archiving \
     --storage-location <$MYBUCKET> \
     --credential-name myauditcredential \
     --storage-region <$HOME_REGION>
     --enabled
     --verify-only
   ```

   For the credential-name parameter, use the CRN of the credential that you noted when you set up the credential for audit event archiving.

   If successful, the command will archive a test audit event to the container.

2. Run cdp audit configure-archiving again, omitting the --verify-only option, to apply the configuration to the account of the caller.

   ```
   cdp audit configure-archiving \
     --storage-location <$MYBUCKET> \
     --credential-name myauditcredential \
     --storage-region <$HOME_REGION>
     --enabled
   ```

   The command returns the archiving configuration for the account of the caller.

   You can also include the following options:

   - To disable archiving while retaining the other configuration information, use the --no-enabled option instead of the --enabled option.
   - To retrieve the current archiving configuration, use a command like the following (there are no required options):

     ```
     cdp audit get-archiving-config
     ```

Azure setup for audit archiving

While the auditing system stores generated audit events initially into a cloud provider managed database, after a specific amount of time, audit logs are exported to customer-managed storage, in their own cloud provider. As a result, you need to set up audit archiving for your platform.

Creating a storage account in Azure

To set up audit archiving, you must first designate a storage account and container for audit archiving in the Azure portal. While you can use an existing storage account, Cloudera recommends that you create a dedicated storage account for audit archiving, especially if you are archiving any sensitive data.

Before you begin

When creating a new storage account, you'll need to know the following pieces of information for creating a new Service Principal in Azure later on:

- Resource group name
- Storage account name

Although it is not required, it can be helpful to create a new resource group when you create a new storage account, so that future management becomes easier. You can create a new resource group in an Azure shell in the Azure Portal by running the following command:

```
az group create --location <location> --name <resource-group-name>
```

Procedure

1. In Azure, create a storage account.

   In the **Basics** page, you should be able to accept the defaults for all settings other than the name. You should also be able to accept the defaults on the **Advanced** and **Networking** pages.

   Note: Ensure that the Enable public access on all network setting is enabled on the **Networking** page for the Storage account as the audit archiving can fail if the default setting is changed.

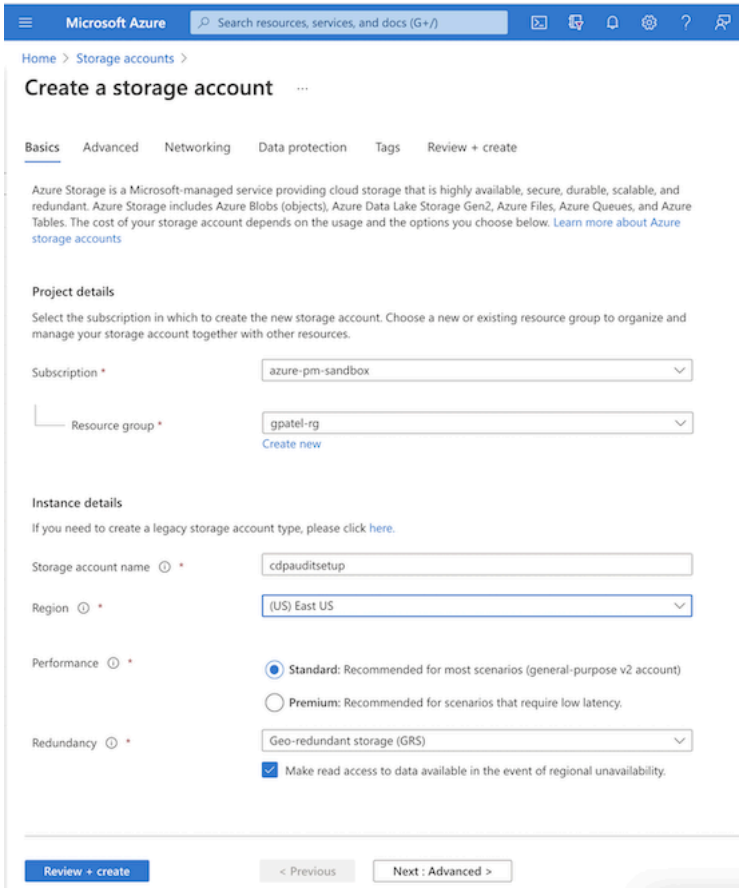

2. When you reach the **Data Protection** page, you must disable the soft delete options for blobs, containers, and file shares. These options are enabled by default, so be sure to uncheck the selection boxes for these options, as shown below:

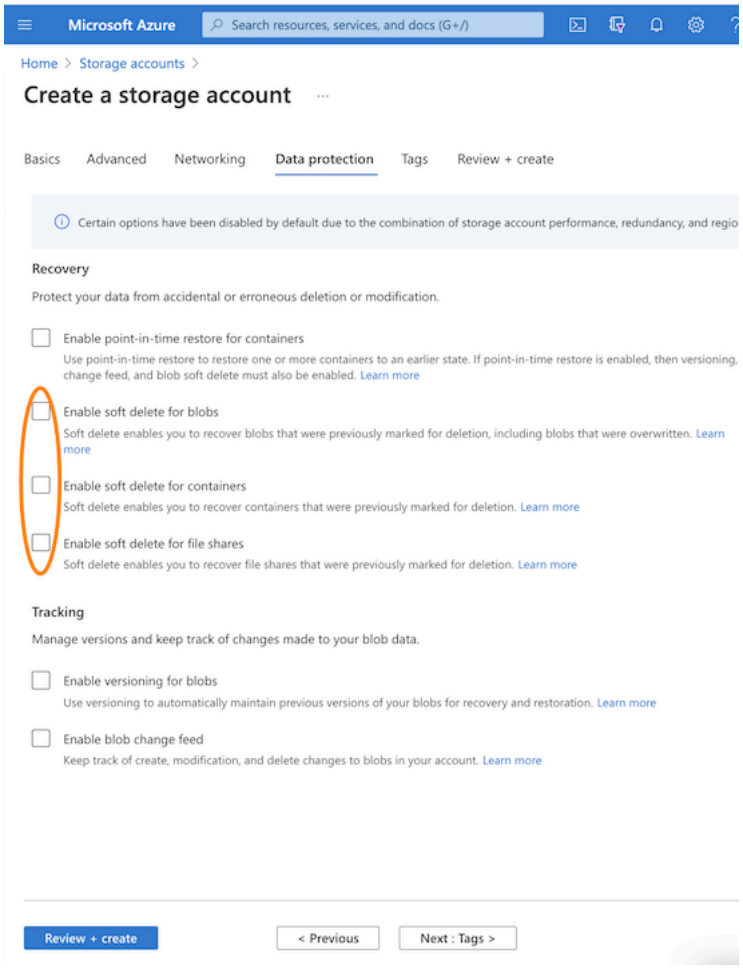

3. Review your changes in the next screen and verify that the soft delete options are disabled. If everything looks correct, click the Create button.

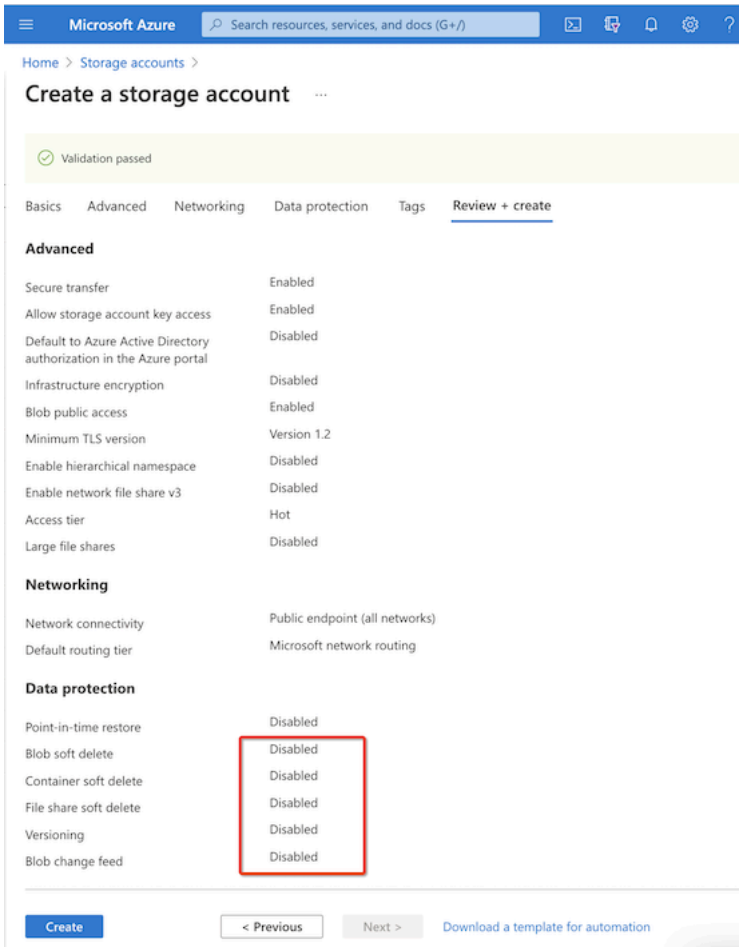

Alternatively, you can create a storage account using the Azure shell by running the following command:

```
az storage account create --name <storage-account-name> --kind <kind> --
location <location> --resource-group <resource-group-name>
```

Creating a container in Azure

To set up audit archiving, you must designate a container for audit archiving in the Azure portal. While you can use an existing container, Cloudera recommends that you create a dedicated container for audit archiving.

Before you begin

Before you create a container, take note of the location property for the storage account at the top of the page in the Storage Account overview section. You will use this later as the storageRegion value when you configure archiving using the CDP CLI or UI.

Procedure

1. Navigate to your newly created storage account resource and click the Containers button:

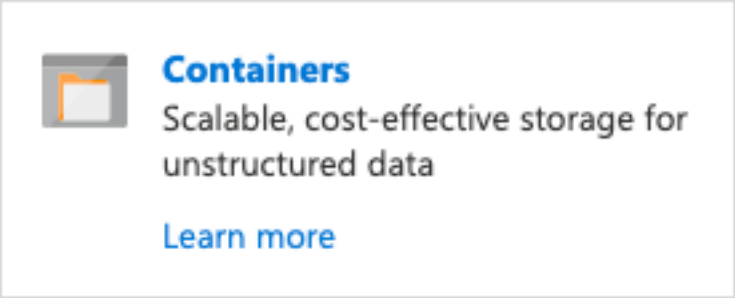

2. In the next screen, click the + Container button, then enter a name for the container and the default Private access level.

   You will use the container name when you create a new service principal in Azure.

   Alternatively, you can create a new container using the Azure shell by running the following command:

```
az storage container create --name <container-name> --account-name <stor
age-account-name>
```

3. Obtain the URL for the newly created container.

   To retrieve the URL, open the container and click the Properties link in the left navigation bar under Settings. Copy the URL shown. You will need this value when you configure audit archiving as the value for the storage location URL. The format is: https://<storage-account-name>.blob.core.windows.net/container-name{{}}

Configuring audit event archiving through the UI

To configure audit event archiving for Azure through the Cloudera UI, create a new audit archiving credential and then configure the audit data storage and location.

About this task

The process of creating an audit archiving credential is largely the same as creating a credential for an Azure environment, however, this credential is a unique credential used solely for audit event archiving.

Before you begin

You must have an Azure resource group as well as an Azure storage account and a container that will be used to store the audit archive logs. You can use an existing resource group, storage account, and container, but Cloudera recommends creating a new storage account and container dedicated to audit archiving. See the topic *Creating ADLS Gen2 storage account and container*.

Required Role: PowerUser

Procedure

1. In the left-side navigation menu, click Global SettingsAudit Data Configuration and then click Create.
2. Select the Azure icon.
3. Use the provided command in an Azure shell to identify your subscription ID and tenant ID:

```
az account list|jq '.[]|{"name": .name, "subscriptionId": .id, "tenantId
": .tenantId, "state": .state}'
```

4. In the corresponding fields, enter the Subscription ID and Tenant ID that are returned by the command.

Settings / Credential

Audit Data Configuration Telemetry Tags

aws

Paste the following command into Azure Shell to identify your Subscription Id and your Tenant Id:

```
az account list|jq '.[]|{"name": .name, "subs
```

Subscription Id*

Enter Azure Subscription Id

Tenant Id*

Enter Tenant Id (Directory Id)

In order to create an application, you could use following command in Azure Shell or you could create it on Azure Portal.

```
az ad sp create-for-rbac \    --name http://{
```

App Id*

Enter Application Id

Password*

Enter the key generated for your app

Create

5. Use the provided command in an Azure shell to register a new application in Azure. Substitute your application name (whatever you chose to give it), subscription ID, resource group name, storage account name, and container name where indicated:

```
az ad sp create-for-rbac \    --name http://{app-name} \    --role "Stor
age Blob Data Contributor" \    --scopes /subscriptions/{subscriptionId}
/resourceGroups/{resource-group-name}/providers/Microsoft.Storage/storag
eAccounts/{storage-account-name}/blobServices/default/containers/{contai
ner-name}
```

   Save the App ID and password from the command output to enter in the Audit Data Configuration **Credential** screen.

   Alternatively, you can follow steps 1-8 the in topic *Create an app-based credential* to register the new app through the Azure portal. When you add a role assignment, use the more limited "Storage Blob Data Contributor" role.

6. Back in the Cloudera **Audit Data Configuration** screen, enter the App Id and Password in the corresponding fields and click Create.
7. Configure the audit data location with the path to the ADLS container that you designated as the audit archive container. Use the following format:

```
https://<storage-account-name>.blob.core.windows.net/<container-name>
```

8. Select the region where storage services should be accessed.
9. Use the toggle button to enable or disable audit log export to the configured storage location.
10. Click Save.

Results

Audit event archiving configuration is complete.

Related Information

Creating ADLS Gen2 storage account and container

Create an app-based credential (Azure)

Creating a Cloudera credential for audit archiving on Azure using the CLI

After you create (or designate) a storage account and container in Azure, you need to create a Cloudera credential for audit archiving on Azure.

Before you begin

Before proceeding with this task, open the Azure shell in the Azure Portal.

Required Role: PowerUser

Procedure

1. Use the provided command in an Azure shell to identify your subscription ID and tenant ID:

```
az account list|jq '.[]|{"name": .name, "subscriptionId": .id, "tenantId
": .tenantId, "state": .state}'
```

   Take note of the subscription ID (the subscription used to create the storage account that you designated for auditing) and the tenant ID for later use.

2. Use the provided command in an Azure shell to create a new service principal in Azure. Substitute your service principal name (whatever you chose to give it, for example the container name), subscription ID, resource group name, storage account name, and container name where indicated:

```
az ad sp create-for-rbac \    --name http://{app-name} \    --role "Stor
age Blob Data Contributor" \    --scopes /subscriptions/{subscriptionId}
/resourceGroups/{resource-group-name}/providers/Microsoft.Storage/storag
eAccounts/{storage-account-name}/blobServices/default/containers/{contai
ner-name}
```

   Save the App ID and password from the command output to use in the credential creation command.

3. In the CDP CLI, use the following command to create the Cloudera credential:

```
cdp environments set-azure-audit-credential --profile <profile-name> --s
ubscription-id <subscription-id> --tenant-id <tenant-id> --app-based app
licationId=<application-id>,secretKey=<password>
```

   The value of the secretKey parameter is the password returned in the output of the previous command that you ran to create the Azure service principal.

   Running this command creates the Cloudera credential for audit archiving on Azure. Take note of the crn returned in the command output, as you will need it to configure archiving in the next task.

What to do next

After you create the credential for auditing, proceed to the next task: *Configuring audit archiving for Azure using the CLI*.

Configuring audit archiving for Azure using the CLI

After you create a Cloudera credential, configure Cloudera for audit archiving on Azure.

Before you begin

Required Role: PowerUser

Procedure

1. Test the audit configuration by running the following command:

```
cdp audit configure-archiving --profile <profile-name> --storage-location
 <storage-location-url>
--credential-name <credential-name> --enabled --storage-region <storage-re
gion> --verify-only
```

   For the credential-name parameter, pass the CRN returned in the command output when you created the credential in the previous task.

   For the storage location, use the URL obtained from the **Properties** page of the container in the Azure Management Console. The format is: https://<storage-account-name>.blob.windows.core.net/<container-name>

   If successful, the command will archive a test audit event to the container.

2. When the response to the previous command is successful, use the command again without the verify-only flag to set the audit archive configuration:

```
cdp audit configure-archiving --storage-location <storage-location-URL>
--credential-name <credential-Name> --enabled --storage-region <storage-
Region> --profile <profile-name>
```

Results

Audit archive configuration is complete.

GCP setup for audit archiving

While the auditing system stores generated audit events initially into a cloud provider managed database, after a specific amount of time, audit logs are exported to customer-managed storage, in their own cloud provider. As a result, you need to set up audit archiving for your platform.

Configuring audit event archiving through the UI

To configure archiving for GCP, you must create a GCP service account specifically for audit event archiving, download the service account private key in JSON format, and then upload the service account private key to Cloudera.

Before you begin

In GCP, create a new GCS bucket or designate an existing bucket for audit archiving. Be sure to block all public access. Audit events will be archived under the /cdp/cp folder, which will be created automatically by Cloudera.

Required Role: PowerUser

Procedure

1. Log in to the Cloudera interface.
2. In the left-side navigation menu, click Global SettingsAudit Data Configuration and then click Create.

3. Select the GCP icon.

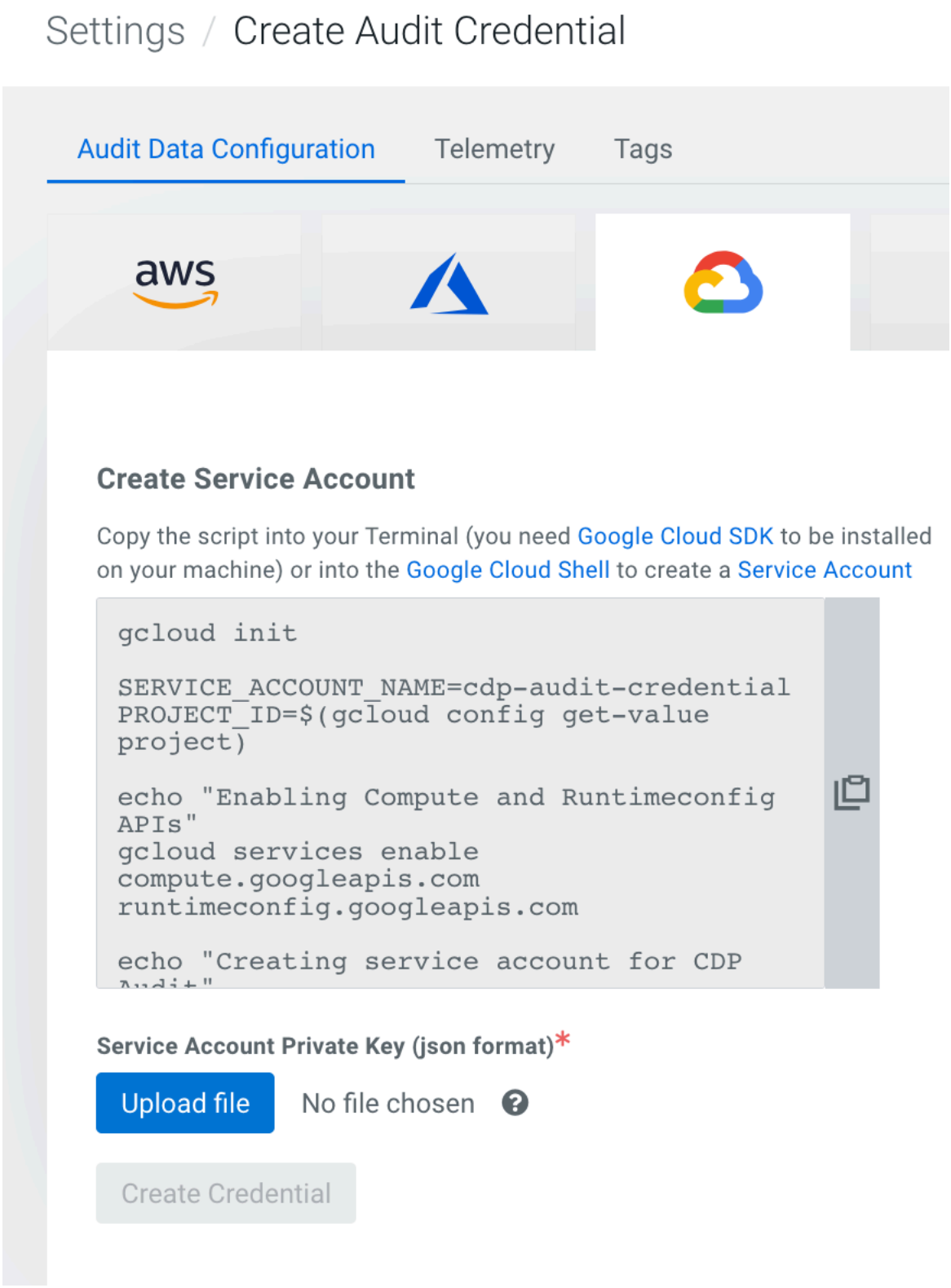

4. Copy the script provided into your terminal or Google Cloud Shell to create a new service account and generate the service account private key.

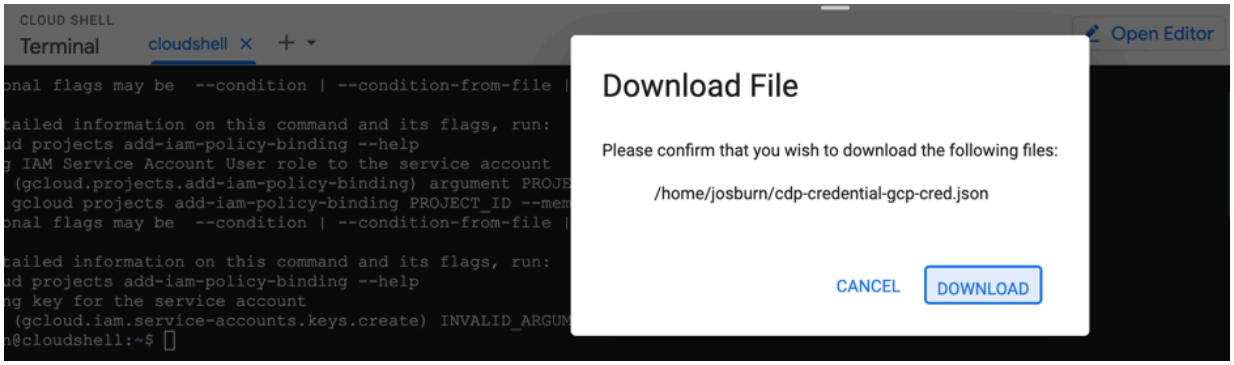

When the script finishes running, it begins a download of the service account private key.

5. When you have finished creating the new service account and have the service account private key, click Upload file on the **Create Audit Credential** page in Cloudera to upload the private key JSON to Cloudera.
6. Click Create Credential.
7. After the credential has been created, you must configure the audit data location. In the Storage location field, provide the full path to the GCS bucket that you created or designated to be the audit log bucket.
8. Select the bucket region, then decide whether or not to export the audit logs to the configured storage location. You can also verify the configuration before saving.
9. Click Save Configuration.

Results

Audit event archiving configuration is complete.

Configure GCP audit event archiving using the CLI

Use the CDP CLI to configure audit event archiving in GCP.

Procedure

1. Use the CDP CLI to get the GCP command that creates the GCP service account and private key required for audit credential generation. In the CDP CLI, run the following command:

```
cdp environments get-audit-credential-prerequisites --cloud-platform gcp
```

The CDP CLI returns the service account creation command (in base64 decoded format) for GCP credential creation. Run this command in your terminal (with Google SDK installed) or Google Cloud Shell to create the new service account and download the private key that is required in the next step.

2. In the CDP CLI, run the command to upload the service account private key:

```
cdp environments set-gcp-audit-credential --credential-key <path to priv
ate key JSON file>
```

For example:

```
cdp environments set-gcp-audit-credential --credential-key file:///Users/
jo/cdp_tools/artifacts/gcpauditcred.json
```

This command returns the credential name/CRN that you will need for the next step.

3. Run the following command to configure GCP audit event archiving in Cloudera:

```
cdp audit configure-archiving --storage-location <full path to bucket> --
credential-name <audit credential name or CRN> --enabled
```

Optionally, you can use the --storage-region parameter to designate the storage region or the --verify-only parameter to verify the audit configuration but not update it.

For example:

```
cdp audit configure-archiving --storage-location gs://cdp/auditbucket --
credential-name audit-credential-5617y894 --enabled --verify-only
```

Pull-based audit archiving

Pull-based audit archiving allows you to pull audit events for archiving purposes without any extra configuration beyond Cloudera Control Plane API usage.

About this task

The Cloudera Control Plane auditing system archives auditing events by writing them to cloud storage that you configure and manage yourself. If you do not want to provide the network access or credentials required for the Cloudera Control Plane to automatically export audit logs to cloud storage in your cloud provider, you can use pull-based audit archiving to retrieve the events yourself in the same format (see *What archiving looks like*). Using pull-based audit archiving, you can group the audit events into batches, list the event batches that have not been marked as archived, retrieve those batches, and then mark them as ready for purging from the Cloudera Control Plane database.

Important: Pull-based audit archiving commands will return an error if automated audit archiving is enabled.

Procedure

1. On a command line, run cdp audit batch-events-for-archiving to begin the asynchronous process of grouping the audit events into batches. For example:

```
cdp audit batch-events-for-archiving \
 --from-timestamp 2020-03-01T00:00:00Z \
 --to-timestamp 2020-04-01T00:00:00Z
```

Note that if there is already a batch event operation in-progress, running the command again is not allowed. If you run the command when an operation is already in progress, you will receive an error.

If successful, the command returns a task ID for tracking the status of the process. For example:

```
{
    "taskId": "0b67c29c-bce9-4bbd-ac3e-8445df029f4f"
}
```

2. Use the cdp audit get-batch-events-for-archiving-status command and the task ID to poll the asynchronous task repeatedly, until it completes successfully or with an error:

```
cdp audit get-batch-events-for-archiving-status \
  --task-id 0b67c29c-bce9-4bbd-ac3e-8445df029f4f
```

While the task is running, the status will be "OPEN":

```
{
    "status": "OPEN",
```

```
    "eventBatches": []
}
```

When it completes successfully, the status will be "COMPLETED," and identifiers will be returned for the event batches:

```
{
    "status": "COMPLETED",
    "eventBatches": [
        {
            "accountId": "37t8i20c-cd82-4e8b-39e4-dcae1f9cd7ef",
            "eventCount": 11,
            "archiveId": "c5b57c79-6721-4e27-9hr9-67f5d299b1gq",
            "archiveTimestamp": 0
        }
    ]
 }
```

3. Run cdp audit list-outstanding-archive-batches to determine event batches which have not yet been marked as archived. The output appears similar to the example below:

```
{
    "eventBatches": [
        {
            "accountId": "37t8120c-cd82-4e8b-39e4-dcae1f9cd7ef",
            "eventCount": -1,
            "archiveId": "c5b57c79-6721-4e27-9hr9-67f5d299b1gq",
            "archiveTimestamp": 0
        }
    ]
}
```

4. For each batch that has not been marked as archived, run cdp audit list-events-in-archive-batch to retrieve the batch of events:

```
cdp audit list-events-in-archive-batch \
  --archive-id c5b57c79-6721-4e27-9hr9-67f5d299b1gq
```

The output of which will be similar to:

```
{
    "auditEvents": [
        {
            "version": "1.0.0",
            ...
        },
        ...
     ]
 }
```

Optionally, you can use shell builtins and utilities to convert the output to a gzipped JSON lines format, like the archives produced by automated audit archiving, with a file name that includes the account ID, a timestamp, and the batch archive ID. For example:

```
cdp audit list-events-in-archive-batch \
  --archive-id c5b57c79-6721-4e27-9hr9-67f5d299b1gq \
  | jq -c '.auditEvents[]' \
  | gzip > 37t8120c-cd82-4e8b-39e4-dcae1f9cd7ef_`date -u +%Y%m%dT%H%M`Z_
c5b57c79-6721-4e27-9hr9-67f5d299b1gq.json.gz
```

5. Once you have saved an archive to your storage destination, use cdp audit mark-archive-batches-as-successful to mark a batch as successfully archived, so that it can later be purged automatically from the Cloudera Control Plane database. You can provide one or more batches for the --archive-ids parameter. For example:

```
cdp audit mark-archive-batches-as-successful \
  --archive-ids c5b57c79-6721-4e27-9hr9-67f5d299b1gq
```

```
{
    "archiveIds": [
        "c5b57c79-6721-4e27-9hr9-67f5d299b1gq"
    ],
    "archiveTimestamp": "2021-08-10T21:54:59.223000+00:00"
}
```

Related Information

What archiving looks like

What archiving looks like

An audit event is eligible to be archived once it is complete, including all result data for the associated activity. The audit service will allow up to one hour for missing result data to become available for an event before archiving it. After that timeout period, however, the incomplete event will become eligible for archiving.

Audit events are not saved in individual files. Instead, groups of audit events from a continuous time span are saved into a single file, with one JSON object / audit event per line. The file is compressed using gzip for efficiency. The Cloudera Control Plane does not encrypt the files. If you would like the files encrypted, you will need to configure your cloud storage service to perform it.

Files are organized in cloud storage based on the dates of their timestamps, as follows:

```
/cdp/cp
  # yyyy (four-digit year)
    # MM (two-digit month)
      # dd (two-digit day of month)
```

Audit events are not saved to cloud storage immediately after they are generated, to enable efficiency of storage and to alleviate performance concerns for the audit service. On an hourly basis, the audit service batches up audit events that are eligible for archiving and writes them to cloud storage.

An audit event is eligible to be archived once it is complete, including all result data for the associated activity. The audit service will allow up to one hour for missing result data to become available for an event before archiving it. After that timeout period, however, the incomplete event will become eligible for archiving.

The Cloudera Control Plane (audit service) periodically saves audit events to your cloud storage facility, for example, an S3 bucket. Cloud storage is intended to be the primary location for storage, retrieval, and analysis of audit events in the long term.

Cloudera Management Console

Managing Classic Clusters

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing classic clusters....4

Enabling admin and user access to classic clusters....4

Prerequisites for adding classic clusters.... 4

Adding an HDP cluster....6

Adding a CDH cluster....8

Adding a Cloudera Private Cloud Base cluster....10

Cloudera Replication Manager use case.... 10

Cloudera Replication Manager and Cloudera Data Catalog use case.... 12

Non-transparent proxy....13

Troubleshooting classic clusters....14

Resume the cluster registration.... 16

Delete an unregistered classic cluster....16

Managing a classic cluster....17

Upgrading a classic cluster from CCMv1 to CCMv2....20

Managing classic clusters

You can register CDH, HDP, and Cloudera Private Cloud Base clusters to Cloudera and later replicate data and workloads from these clusters to the clusters in your Cloudera environments. These clusters are called Classic Clusters in Cloudera.

To add or register CDH, HDP, and Cloudera Private Cloud Base to Cloudera:

1. Make sure all the prerequisites are met before adding the classic cluster.
2. Configure the classic cluster to establish connectivity to Cloudera.
3. On the Cloudera Management Console, add the classic cluster.

For more information, refer to the following documentation:

Related Information

Prerequisites for adding classic clusters

Enabling admin and user access to classic clusters

Troubleshooting classic cluster registration errors (CCMv2)

Resume the cluster registration

Upgrading a classic cluster from CCMv1 to CCMv2

Enabling admin and user access to classic clusters

In order to grant admin and user access to a classic cluster, you should assign the required roles.

Consider the following when granting access to admins and users of your classic clusters:

- You need to be a ClassicClustersCreator in order to register classic clusters.
- The user who registers a classic cluster gets the Owner role for that classic cluster.
- Once a classic cluster is registered, the following roles can be assigned to a user or a group on that cluster:
- Owner - Grants all permissions on the classic cluster in Cloudera including the ability to delete them. It does not grant any cluster-level access (such as Cloudera Manager access).
- ClassicClusterAdmin - Grants permission to perform any operation on the cluster, except deleting it. Grants permission to assign access to the cluster to other users.
- ClassicClusterUser - Grants permission to access details of the classic cluster.

The roles are described in detail in Resource roles. The steps for assigning the roles are described in Assigning resource roles to users and Assigning resource roles to groups.

Related Information

Account roles

Resource roles

Assigning resource roles to users

Assigning resource roles to groups

Prerequisites for adding classic clusters

Make sure you verify the version requirements of your classic clusters and install or configure all of the classic cluster requirements before you try to add or register them to Cloudera.

Required roles

You need to have the ClassicClustersCreator role to register a classic cluster in Cloudera.

Check the version compatibility

Make sure that the following version requirements are met.

- For adding HDP clusters:
  - HDP - HDP 2.6.5.50000 plus any required patch version
  - DLM Engine version - 1.7.0.0-x
  - Ambari - 2.7.3 or 2.6.2.2 or 2.6.2
- For adding CDH clusters:
  - CDH - 5.x and 6.x
  - Cloudera Manager - 5.x and 6.x
- For adding Cloudera Private Cloud Base clusters
  - Cloudera Private Cloud Base cluster 7.1.1 or later, and its respective Cloudera Manager version; For example, Cloudera Private Cloud Base cluster 7.1.6 uses Cloudera Manager 7.3.1.

Note: Currently, classic cluster registration only supports CentOS and RHEL operating systems, versions 6, 7 and 8.

Verify the required roles are assigned

- Make sure that the user can log in as the admin user to Ambari in the HDP cluster.
- Make sure the user can log in as the admin user to Cloudera Manager in the CDH cluster or the Cloudera Private Cloud Base cluster.

Install the required components (HDP only)

Fulfill the following requirements if using HDP. If using CDH or Cloudera Private Cloud Base, skip this section.

- Make sure that Ambari Metrics is installed in Ambari in the HDP clusters.
- Make sure that Ambari is configured with LDAP.
- Make sure that Knox is installed and the default topology is configured with LDAP.

  Note: Knox does not need to be used by other services in the cluster; It is only required for Cloudera communication.

- Make sure that there is at least one topology in the Knox setup with the same LDAP as Ambari.
- If there are policies restricting access through Knox, make sure that Ranger policies allow communication through Knox.
- Make sure that the cluster is configured with Kerberos.
- Make sure that the user credential used for registering the classic cluster is a valid LDAP user with an admin role in Ambari.

Open ports for Cluster Connectivity Manager

Ensure that outgoing traffic is allowed on the port range 443 on the legacy CDH, HDP, or Cloudera Private Cloud Base cluster.

After making sure that your clusters meet all the requirements, you can add your CDH, HDP, and Cloudera Private Cloud Base clusters to Cloudera.

Related Information

Cluster Connectivity Manager (CCM)

Using classic clusters with a non-transparent proxy

Adding an HDP classic cluster

You must register HDP clusters with Cloudera before you can use them with Cloudera Public Cloud services and components.

About this task

To ensure optimum security, clusters within the customer environment are not accessible for communication: They have private IP addresses and cannot be accessed outside the firewall. To add your cluster to Cloudera, a communication line needs to be established.

The Cluster Connectivity Manager inverted proxy solves the problem by establishing a connection from the on-premise cluster to Cloudera. You must download and install the jumpgate agent and the connectivity install scripts to establish a secure two-way communication channel. The jumpgate agent ensures that the connectivity is stable. The connectivity scripts and their installation ensure safe connectivity and communication.

Before you begin

- HDP clusters must be managed by Ambari. HDP clusters that are not managed by Ambari cannot be registered to Cloudera.
- HDP clusters must include Knox.
- HDP clusters must include Ranger policy settings
- LDAP/AD must be set up and synced in Ambari.

  LDAP settings are automatically detected from the default topology setup in Knox. If the default topology does not have the LDAP setup, you will be asked to provide another topology name where you have configured the LDAP. If that topology has LDAP, the setup continues. If the LDAP is not configured, you will receive an error message.
- Kerberos must be enabled on the HDP cluster and the LDAP/AD must be set up in the Kerberos authentication so that the same set of LDAP/AD credentials can be used to access Ambari APIs as well as Beacon APIs.
- All clusters must meet the requirements identified in Prerequisites for adding classic clusters.

Caution: After you register an HDP cluster in Cloudera, do not change the cluster name in Ambari. A cluster name change in Ambari does not currently propagate to Cloudera, which can result in issues when using the HDP cluster with Cloudera clusters and components.

Caution: The jumphost and connectivity-install scripts files must be stored in a secure environment

Steps

The process to register an HDP cluster is as follows:

1. Log in to Cloudera and navigate to the Cloudera Management Console.
2. Click Classic Clusters in the left navigation panel.
3. Click Add Cluster.

   CDP displays the Cluster Details dialog box.
4. If you are a first time user, under Step 1 in the Register Classic Cluster wizard, click GET STARTED. If you are not a first time user, click the ADD CLUSTER button on the right side of the listing page.

   CDP displays the Cluster Details dialog box.
5. Select HDP.

6. Provide the following connectivity information for your new cluster:
   a. Knox IP Address
   b. Knox port
   c. Data center
   d. Click CONNECT

   Step 1 might take up to 5 minutes if you are adding the cluster for the first time. After Classic Cluster establishes connection, Cloudera will highlight Step 2 in the Register Classic Cluster wizard.
7. Start the download and installation process for the connectivity files by clicking the Files button in Step 2 of the wizard.
8. Follow the instructions in the Setup Connectivity Client dialog box. You need to download the jumpgate-agent rpm file and the cluster_connectivity_setup_files zip files and copy them to your Knox node. Or if your Knox is running in HA mode, you need to copy the files to the Knox proxy host in the cluster:
   a. Download the jumpgate-agent RPM and cluster_connectivity_setup_files.zip.
   b. In the command line interface, copy the two files to the Knox or Knox proxy host.
   c. SSH to the host.
   d. Install the jumpgate-agent rpm using:

   ```
   yum --nogpgcheck localinstall < downloaded-jumpgate-agent-rpm >
   ```

   a. Unzip the cluster_connectivity_setup_files file. Inside this zip file there is a script install.sh.
   b. Run install.sh by using ./install.sh command.
   c. Check service status to see if the agent has been connected:

   ```
   systemctl status jumpgate-agent.service
   ```
9. Enter the following information as the install script prompts for it:
   a. Enter Ambari URL (http(s)://host:[port]):
   b. Enter Ambari Username:
   c. Enter Ambari Password:
10. If Knox is not installed on a proxy server, proceed to Step 12. Classic Cluster sets up the topology for the Knox server and establishes the connection.
11. If Knox is installed on a proxy server, Classic Cluster displays the following message:We discovered that your Knox is installed in HA mode. Please confirm if this node is your proxy node (yes/no):Enter yes. Classic Cluster generates XML content that you will need to add to your Knox hosts. Classic Cluster also displays three steps you must perform on all of your Knox hosts:
    a. Copy the generated XML to /usr/hdp/current/knox-server/conf/topologies/cdp_default.xml
    b. Run chown knox:knox cdp_default.xml
    c. Check the Knox logs/deployment directory to verify that the cdp_default topology is deployed.
    d. After you have completed steps a through c on all of your Knox hosts, type Enter to continue.This sets up the topology for the Knox server or Knox proxy host and establishes the connection.
12. On the Classic Clusters page, click Test Connection in the Step 2 pane to verify whether the connection is successful.
13. Cloudera starts checking the connectivity with the HDP cluster. When the connectivity is successful, proceed to Step 3 in the wizard.

    If the connection attempts fail or if there is an error in the connectivity, Cloudera displays troubleshooting information in Step 2 of the registration wizard. Follow the troubleshooting information to fix the connectivity error, then click Test connection.

    Note: After you download the files, the cluster_connectivity_setup files download is disabled. At this point, you can regenerate the cluster connectivity setup files using the regenerate files option. This option comes in handy if you lose the files previously downloaded before you can set the cluster connectivity in the cluster.

14. Click Register in Step 3 of the registration wizard.
15. In the Cluster Details dialog box, provide the username and password to access the cluster, then click CONNECT.

    The user should have admin access to the cluster services.
16. Finish registering the cluster by providing the following information.

    a. Cluster Location
    b. Data Center
    c. Tags (optional)
    d. Description (optional)

    If LDAP is not set up on the default topology, the system will ask for the following additional information: Enter knox topology name that contains LDAP setup.
17. Click ADD.

Result

Once the cluster has been registered, you can use it with Cloudera.

Adding a CDH classic cluster

You must register CDH clusters in Cloudera before you can use them with Cloudera Public Cloud services and components.

About this task

To ensure optimal security, clusters within the customer environment are not accessible for communication; They have private IP addresses and cannot be accessed outside the firewall. To add your cluster to the Cloudera, a communication line needs to be established.

The Cluster Connectivity Manager inverted proxy solves the problem by establishing a connection from the on-premise cluster to Cloudera. You must download and install the jumpgate agent and the connectivity install scripts to establish a secure two-way communication channel. The jumpgate agent ensures that the connectivity is stable. The connectivity scripts and their installation ensure safe connectivity and communication.

The high-level steps to register a CDH cluster using Cluster Connectivity Manager are as follows:

1. In the Cloudera Management Console, you enter the private IP address of your cluster and provide the cluster details.
2. You download the jumpgate agent rpm from the specified location and the connectivity installation scripts from Cloudera on to the cluster.
3. You install jumpgate agent on the cluster.
4. You register the cluster for performing further operations.

Detailed steps are provided below.

Before you begin

- CDH clusters must have been created using Cloudera Manager. Clusters that are not managed by Cloudera Manager cannot be registered to Cloudera.
- All clusters must meet the requirements identified in Prerequisites for adding classic clusters.

Caution: After you register a CDH cluster in Cloudera, do not change the cluster name in Cloudera Manager. A cluster name change in Cloudera Manager does not currently propagate to Cloudera, which can result in issues when using the CDH cluster with Cloudera clusters and components.

Caution: The jumpgate agent and connectivity-install scripts files must be stored in a secure location.

Steps

Perform the following steps to add a CDH cluster:

1. Log in to Cloudera and navigate to the Cloudera Management Console.
2. Click Classic Clusters in the left navigation panel.
3. Click Add Cluster.

   CDP displays the Cluster Details dialog box.
4. Click CDH.
5. Provide the connectivity information for your new cluster:

   Note: Make sure that the Data Center name is different from the Data Center names that have already been registered. If the Data Center name exists, make sure that the combination of the Data Center name and the cluster name is unique. Else, you may get an error when you try to add a cluster with an existing Data Center-cluster name combination.

   a. Cloudera Manger IP address
   b. Cloudera Manager Port
   c. Data center
   d. Select the My cluster runs on HTTPS option if the CDH cluster uses HTTPS.

   Step 1 might take up to five minutes if you are adding the cluster for the first time. After Step 1 is complete, Cloudera will highlight Step 2.
6. Start the download and installation process for the connectivity files by clicking the Files button in the Step 2 pane.
7. Follow the instructions in the Setup Connectivity Client dialog box. You need to download the jumpgate-agent rpm file and the cluster_connectivity_setup_files zip file onto Cloudera Manager host in your new cluster:

   a. Download the jumpgate-agent RPM and cluster_connectivity_setup_files.
   b. In the command line interface, copy the two files to the Cloudera Manager host.
   c. SSH to the Cloudera Manager host.
   d. Install the jumpgate-agent rpm using:

   ```
   yum --nogpgcheck localinstall <
   downloaded-jumpgate-agent-rpm >
   ```

   e. Unzip the cluster_connectivity_setup_files file. Inside this zip file there is a script install.sh.
   f. Run install.sh by using ./install.sh command.
   g. Check service status to see if the agent has been connected:

   ```
   systemctl status jumpgate-agent.service
   ```

   On the Classic Clusters page, click Test Connection in the Step 2 pane to verify whether the connection is successful.

   After CDP successfully connects to your new cluster, it will highlight Step 2. the connectivity with the cluster. When the connectivity is successful, proceed to Step 3 in the UI.

   If the connection attempts fail or if there is an error in the connectivity, Cloudera displays troubleshooting information in the Step 2 pane. Follow the troubleshooting information to fix the connectivity error, then click Test connection.

   Note: After you download the files, the cluster_connectivity_setup file download is disabled. At this point, you can regenerate the cluster connectivity setup files using the regenerate files option. This option comes in handy if you lose the files previously downloaded before you can set the cluster connectivity in the cluster.
8. Click Register in the Step 3 pane.
9. Provide the username and password of the Cloudera Manager user to access the cluster.

10. Finishing registering the cluster by providing the following information:
    a. Cluster Location
    b. Data Center
    c. Tags (optional)
    d. Description (optional)
11. Click Submit.

Result

Once the cluster has been registered, you can use it with Cloudera.

Adding a Cloudera Private Cloud Base classic cluster

You must register Cloudera Private Cloud Base clusters with Cloudera before you can use them with Cloudera Public Cloud services and components.

To ensure optimum security, clusters within the customer environment are not accessible for communication: They have private IP addresses and cannot be accessed outside the firewall. To add your cluster to the Cloudera, a communication line needs to be established.

The Cluster Connectivity Manager inverted proxy solves the problem by establishing a connection from the on-premise cluster to Cloudera. You must download and install the jumpgate agent and the connectivity install scripts to establish a secure two-way communication channel. The jumpgate agent ensures that the connectivity is stable. The connectivity scripts and their installation ensure safe connectivity and communication.

Note: The jumphost and connectivity-install scripts files must be stored in a secure environment.

To register the Cloudera Private Cloud Base cluster as a classic cluster, you enter the Cloudera Private Cloud Base cluster details. The Management Console acquires the configuration details from the Cluster Connectivity Manager service and saves it as ZIP files. You download the ZIP files, install the acquired configurations, and then register the Cloudera Private Cloud Base cluster as a classic cluster.

Note: After you register a Cloudera Private Cloud Base cluster in Cloudera, do not change the cluster name in Cloudera Manager. A cluster name change in Cloudera Manager does not currently propagate to Cloudera, which can result in issues when using clusters with Cloudera clusters and components.

All the clusters must meet the requirements identified in Prerequisites for adding classic clusters.

You have two options for registering your Cloudera Private Cloud Base cluster:

- If you would like to use the Cloudera Private Cloud Base cluster with Cloudera Replication Manager, register the cluster using Cloudera Manager.
- If you would like to use the Cloudera Private Cloud Base cluster with Cloudera Replication Manager and Cloudera Data Catalog, register the cluster using Cloudera Manager and Knox.

Important: Cloudera Private Cloud Base clusters can be used in Cloudera Data Catalog by registering them using Cloudera Manager and Knox endpoints. Note that this is a technical preview feature and is under development. Do not use this in your production environment. If you have feedback, contact Support by logging a case on the Cloudera Support Portal at https://my.cloudera.com/support.html. Technical preview features are not guaranteed troubleshooting and fixes

For Cloudera Private Cloud Base cluster registration steps in Cloudera, see the following documentation:

Adding Cloudera Private Cloud Base cluster for use in Cloudera Replication Manager (CCMv2)

Register a Cloudera Private Cloud Base cluster as a classic cluster using Cloudera Manager so that you can use this cluster as a source cluster in Replication Manager.

Before you begin

All the clusters must meet the requirements identified in Prerequisites for adding classic clusters.

Steps

1. Log in to Cloudera Management Console.
2. Click Classic Clusters.
3. On the Classic Clusters page, click ADD CLUSTER.
4. In the Add Cluster dialog box, navigate to the Cloudera Private Cloud Base tab and enter the following details:
   a. If your cluster is not reachable by a public network, click “My cluster is accessible only in my private network”.
   b. Cloudera Manager IP address - Enter the IP address of the Cloudera Manager of the Cloudera Private Cloud Base cluster. The Cloudera Management Console uses this IP address to identify the cluster for registration purposes.
   c. Cloudera Manager Port - Enter the port of the Cloudera Manager of the Cloudera Private Cloud Base cluster.
   d. Data center - Enter a unique data center name for the Cloudera Private Cloud Base cluster.
   e. Select the My cluster runs on HTTPS option if the Cloudera Private Cloud Base cluster uses HTTPS.
   f. Clear the Register KNOX endpoint (Optional) option, if selected.
   g. Click CONNECT.

   The Cloudera Management Console acquires the configuration details from Cluster Connectivity Manager service. After Cloudera successfully connects to your new cluster (which should take no more than 5 minutes), it will highlight Step 2.
5. On the Classic Clusters page, click Files in the Step 2 pane.
6. Follow the instructions in the Setup Connectivity Client dialog box. You need to download the jumpgate-agent rpm file and the cluster_connectivity_setup_files zip file onto Cloudera Manager host in your new cluster:
   a. In the command line interface, copy the jumpgate-agent RPM and cluster_connectivity_setup_files.zip to the Cloudera Manager host.
   b. SSH to the Cloudera Manager host.
   c. Install the jumpgate-agent rpm using:

   ```
   yum --nogpgcheck localinstall <
   downloaded-jumpgate-agent-rpm >
   ```

   a. Unzip the cluster_connectivity_setup_files file. Inside this zip file there is a script install.sh.
   b. Run install.sh by using ./install.sh command.
   c. Check service status to see if the agent has been connected:

   ```
   systemctl status jumpgate-agent.service
   ```

   Note: If you regenerate the script files, you cannot use the previously downloaded cluster_connectivity _setup_files.zip file because the file is no longer valid.
7. On the Classic Clusters page, click Test Connection in the Step 2 pane to verify whether the connection is successful.
8. Click Register in the Step 3 pane.
9. In the Cluster Details dialog box, enter the Cloudera Manager credentials that have Admin access to Cloudera Manager and the cluster services.
10. Click CONNECT.

11. To complete the registration, enter the following details on the Classic Clusters page:
    a. Cluster Location - Enter the geographical location of the Data Lake.
    b. Data Center - Ensure that the data center name is the name that you provided for Cloudera Private Cloud Base cluster during registration.
    c. Tags - Optionally, enter the tags for the cluster.
    d. Description - Optionally, enter a description.
12. Click Add.

Result

You can use the registered classic cluster in the Cloudera Replication Manager.

Adding Cloudera Private Cloud Base cluster for use in Cloudera Replication Manager and Cloudera Data Catalog (CCMv2)

Register a Cloudera Private Cloud Base cluster as a classic cluster using Cloudera Manager and Knox endpoints so that you can use this cluster in Cloudera Replication Manager and Cloudera Data Catalog.

Before you begin

All the clusters must meet the requirements identified in Prerequisites for adding classic clusters.

Additionally, ensure that the following components and roles are available:

- The Cloudera Private Cloud Base cluster has an active Knox service.
- You can proxy to Cloudera Manager through Knox for communication purposes.
- LDAP is configured in the Cloudera Manager of Cloudera Private Cloud Base cluster. For more information, see Configure authentication using an LDAP-compliant identity service.
- A minimum of one LDAP user with the Full Administrator role.
- An LDAP-based topology cdp_defaul.xml with CM-API, CM-UI, ATLAS, ATLAS-API, RANGERUI, and RANGER services exists. The topology name is used during the classic cluster registration process.

Note: If there are policies that restrict access through Knox, then add the topology name to the cdp_default Ranger policy so that the Ranger policies can communicate through Knox.

Important: Cloudera Private Cloud Base clusters can be used in Cloudera Data Catalog by registering them using Cloudera Manager and Knox endpoints. Note that this is a technical preview feature and is under development. Do not use this in your production environment. If you have feedback, contact Support by logging a case on the Cloudera Support Portal at https://my.cloudera.com/support.html. Technical preview features are not guaranteed troubleshooting and fixes.

Steps

1. Log in to the Cloudera Management Console.
2. Click Classic Clusters.
3. On the Classic Clusters page, click ADD CLUSTER.

4. In the Add Cluster dialog box, navigate to the Cloudera Private Cloud Base tab and enter the following details:
   a. If your cluster is not reachable by a public network, click “My cluster is accessible only in my private network”.
   b. Cloudera Manager IP address - Enter the IP address of the Cloudera Manager of the Cloudera Private Cloud Base cluster. The Management Console uses this IP address to identify the cluster for registration purposes.
   c. Cloudera Manager Port - Enter the port of the Cloudera Manager of the Cloudera Private Cloud Base cluster.
   d. Data center - Enter a unique datacenter name for the Cloudera Private Cloud Base cluster.
   e. Select the My cluster runs on HTTPS option if the Cloudera Private Cloud Base cluster uses HTTPS.
   f. Select the Register KNOX endpoint (Optional) option.
   g. KNOX IP Address - Enter the IP address of the Knox host for the Cloudera Private Cloud Base cluster.
   h. KNOX Port - Enter the port for the Knox service.
   i. Click CONNECT.

   The Management Console acquires the configuration details from Cluster Connectivity Manager (CCM) service. After CDP successfully connects to your new cluster (which should take no more than 5 minutes), it will highlight Step 2.
5. On the Classic Clusters page, click Files in the Step 2 pane.
6. Follow the instructions in the Setup Connectivity Client dialog box. You need to download the jumpgate-agent rpm file and the cluster_connectivity_setup_files zip file onto Cloudera Manager host in your new cluster:
   a. In the command line interface, copy the RPM and ZIP files to the Cloudera Manager host.
   b. SSH to the Cloudera Manager host.
   c. Install the jumpgate-agent rpm using yum --nogpgcheck localinstall < downloaded-jumpgate-agent-rpm >
   d. Unzip the cluster_connectivity_setup_files file. Inside this zip file there is a script install.sh.
   e. Run install.sh by using ./install.sh command.
   f. Check service status to see if the agent has been connected: systemctl status jumpgate-agent.service

   Note: If you regenerate the script files, you cannot use the previously downloaded cluster_connectivity_setup_files.zip file because the file is no longer valid.
7. On the Classic Clusters page, click Test Connection in the Step 2 pane to verify whether the connection is successful.
8. On the Classic Clusters page, click Register in the Step 3 pane.
9. In the Cluster Details dialog box, enter the Cloudera Manager credentials that have Admin access to Cloudera Manager and the cluster services.
10. Click CONNECT.
11. To complete the registration, enter the following details on the Classic Clusters page:
    a. Cluster Location - Enter the geographical location of the Data Lake.
    b. Data Center - Ensure that the data center name is the name that you provided for CDP Private Cloud Base cluster during registration.
    c. Tags - Optionally, enter the tags for the cluster, if any.
    d. Description - Optionally, enter a description.
12. Click Add.

Result

You can use the registered classic cluster in Cloudera Replication Manager and Cloudera Data Catalog.

Using classic clusters with a non-transparent proxy

If your organization has a non-transparent proxy on the Cloudera Manager/Knox node, the following steps must be performed prior to classic cluster registration.

Note: These steps only apply if you have a non-transparent proxy. You do not need to perform them if you have a transparent proxy.

Note: An https proxy is supported only if the certificate is added to the system trust store.

When you register a cluster in Cloudera as a classic cluster, Cloudera installs Cluster Connectivity Manager on the Cloudera Manager/Knox node of CDH and HDP clusters to establish connection between the on-premise cluster and Cloudera, allowing communication with the Cloudera Control Plane to kick off replication jobs on schedule. To do this, Cluster Connectivity Manager must be able to connect to the outside of the Data Center.

Steps

Create the following file on the Cloudera Manager node (in case of a CDH or Cloudera Private Cloud Base cluster) or on the Knox node (in case of an HDP cluster):

```
/etc/cdp/proxy.env
```

The file should include a proxy link. The format of the file should be:

```
https_proxy=http://<username>:<password>@<proxy.com>
```

- The <username> and <password> should be replaced with an actual username that allows access to the proxy.
- The <proxy.com> should be replaced with the URL of the proxy server.

Once you've performed these steps, you can proceed to registering your cluster in Cloudera.

Troubleshooting classic cluster registration errors (CCMv2)

While trying to resume the registration process, you can identify the problem behind the error and fix it.

Issues during registration in Cloudera

Error or issue details	Resolution
Alert: Registration is pending for a cluster with the same details.	• A cluster with the same IP Address and Data Center cannot be registered again. • Check if you have registered this cluster already. To check this, navigate to the Classic Clusters page and search for your cluster. • Check if the IP address is correct. • Provide a different Data Center name.

Cluster side issues

Error or issue details	Resolution
Connection refused even though the systemctl status jumpgate-agent.service shows that the jumpgate agent is running.	Make sure that you copied the right setup files for the cluster or check if the port number CCM_TUNNEL_SERVICE_PORT in cluster_connectivity.conf is your Cloudera Manager's port number in case of CDH/Cloudera Private Cloud Base cluster and Knox port number in case of HDP cluster.

Error or issue details	Resolution

Test connection failure	Try the following:
	• Check if the jumpgate agent is running:
	systemctl status jumpgate-agent.service
	• If the jumpgate agent is not running, start the agent by running the install script.
	• If the agent is active, check if the port number entered during registration is correct.
	• If the port number is incorrect, delete the registration attempt from the UI, remove all the setup-related files from the cluster node and re-register the cluster with the correct information.
	• If the port number is correct, check if outbound connection to the Cloudera Control Plane is allowed.
	  • cat cluster_connectivity.conf and collect the RELAY_SERVER
	  • Check if the RELAY_SERVER is reachable from the node:
	  • On the node, execute the command nslookup <RELAY_SERVER> to check if the RELAY_SERVER is reachable from the node.
	Or execute curl command as mentioned here for connectivity check: curl https://<RELAY_SEVRER>:443
	Or execute telnet command as mentioned here: telnet <RELAY_SERVER> 443
	  • If this fails, then the traffic to the cloudera network is blocked on the customer's VPC.
	  • You should check the outbound rules on your VPC to make sure that the traffic to the Cloudera network is allowed. If the agent is active, check if there are any Ranger policies set up to deny access to the cluster. If such policies exist on the cluster, modify or set up policies to allow access to the cluster cdp_default topology
	• If the agent is active, check if there are any Ranger policies set up to deny access to the cluster. If such policies exist on the cluster, modify or set up policies to allow access to the cluster cdp_default topology
	• If the tunnel is active, check if the port number entered during registration is correct.
	• If the port number is incorrect, delete the registration attempt from the UI, remove all the setup-related file from the cluster node and re-register the cluster with the correct information.
	• If the port number is correct, check if outbound connection to CDP control plane (host/port = CCM_HOST/CCM_SSH_PORT) is allowed on the directory which has the ssh setup files.
	  • cat reverse_tunnel.conf or cat cluster_connectivity.conf and collect the following properties - CCM_HOST, CCM_SSH_PORT, CCM_TUNNEL_INITIATOR_ID
	  • Check if the NLB is reachable from the node:
	  • On the node, execute the command nslookup <CCM_HOST> to check if the NLB is reachable from the node.
	  • If this fails, then the traffic to the cloudera network is blocked on the customer's VPC.
	  • The customer should check the outbound rules on their VPC to make sure that the traffic to the cloudera network is allowed.

Cluster Connectivity Manager issues

Classic Clusters registration uses Cluster Connectivity Manager, enabled by the installation of the Cluster Connectivity Manager client and secrets on the CM node.

Error or issue details	Resolution
“Unable to identify a backendId for this request. This usually happens if either the backendId is invalid or the agent has not yet connected to the relay server” OR “No route found for backend with id <backend-id>. This usually happens if either the backendId is invalid or the agent has not yet connected to the relay server."	The CCMv2 Network Load Balancers (NLBs) might be unreachable. To verify, execute cat cluster_connectivity.conf and collect the RELAY_SERVER field’s value and make sure that the RELEY_SERVER on port 443 is reachable from the legacy CDH/HDP master nodes.

Other issues

Error or issue details	Resolution
If Cluster name and Display name are different, few details are missing from the cluster detail page.	Cluster name and Display name must be the same.

If registering your cluster for use with Cloudera Data Catalog, also check the following:

- Check Proxy to Cloudera Manager through Apache Knox is enabled.
- Make sure that you created the cdp_default topology on the Knox host with required services and LDAP configuration; it's a prerequisite.

Resume the cluster registration

If the cluster registration does not succeed due to some factors, you can resume the registration, identify and fix any issues, and complete the registration process.

Procedure

1. Go to the Classic Clusters page.
2. From the list of unregistered clusters that appears, select your cluster and start the process from the point you left off.

Delete an unregistered classic cluster

If you started registering an on-prem cluster in Cloudera Public Cloud using CCMv1 and have not yet completed it, we recommend that you delete the unregistered cluster and re-register it using CCMv2.

Classic clusters that are already registered with CCMv1 continue to use CCMv1, but new cluster registrations should be using CCMv2. If you started registering an on-prem cluster in Cloudera Public Cloud using CCMv1 and have not yet completed it, we recommend that you delete the unregistered cluster and re-register it using CCMv2. We suggest this as it will save the future efforts required to upgrade from CCMv1 to CCMv2.

Steps

1. If you reached step 2 or 3 in classic cluster registration, you should first stop the CCM tunnel and then clean up the setup files that you must have downloaded for CCM v1. Run the following commands on the Cloudera Manager host (CDH or Cloudera Private Cloud Base cluster) or on the KNOX host (HDP cluster).
   a. Check the tunnel status:

      ```
      systemctl status ccm-tunnel@CM.service
      ```

   b. Use below commands to stop tunnel and cleanup CCMv1 resources:
      - CentOS7/RHEL7:

        CDH or Cloudera Private Cloud Base cluster

        ```
        systemctl stop ccm-tunnel@CM.service
        yum remove autossh
        ```

        HDP cluster

        ```
        systemctl stop ccm-tunnel@KNOX.service
        yum remove autossh
        ```

      - CentOS6/RHEL6:

        CDH or Cloudera Private Cloud Base cluster

        ```
        service reverse-tunnel stop 'CM'
        yum remove autossh
        ```

        HDP cluster

        ```
        service reverse-tunnel stop 'KNOX'
        yum remove autossh
        ```

2. Delete the cluster from the classic cluster UI by clicking on the context menu next to the cluster name and selecting Remove.
3. Enter the cluster name and click Remove.

Managing a classic cluster

You can manage a CDH, HDP, or Cloudera Private Cloud Base cluster using the Classic Cluster user interface.

About this task

You can perform the following management functions using the Classic Cluster user interface.

Procedure

1. You can delete a cluster if it has not been completely registered by clicking the delete icon next to the cluster entry in the list of clusters.

2. You can view cluster details from the UI by clicking the cluster name in the list of clusters.

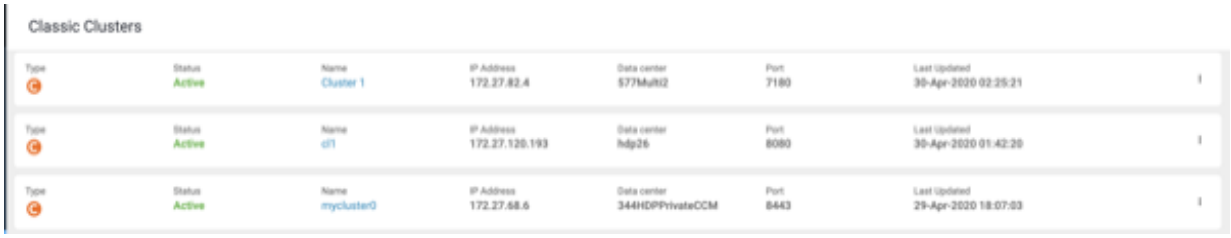

Classic Cluster displays the Details dialog box.

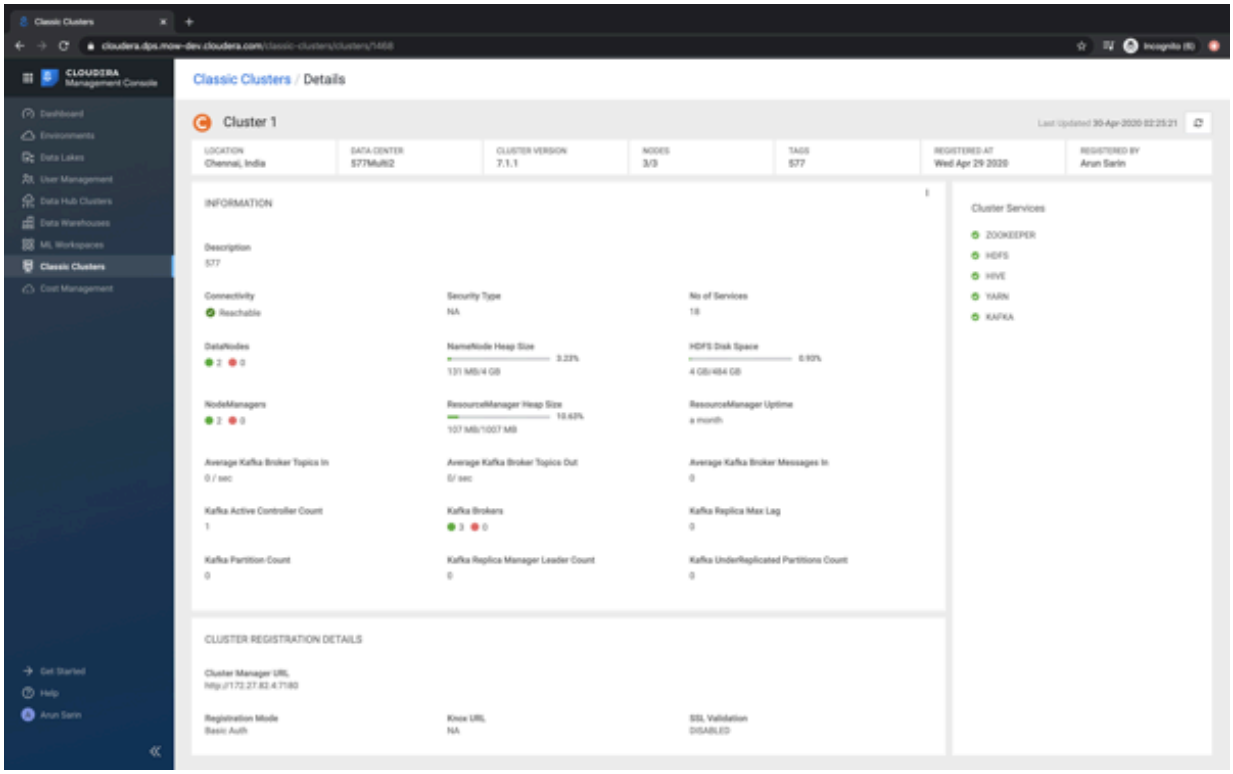

3. You can edit a cluster from the UI by choosing the Edit option from the pull down menu next to the cluster entry in the list of clusters.

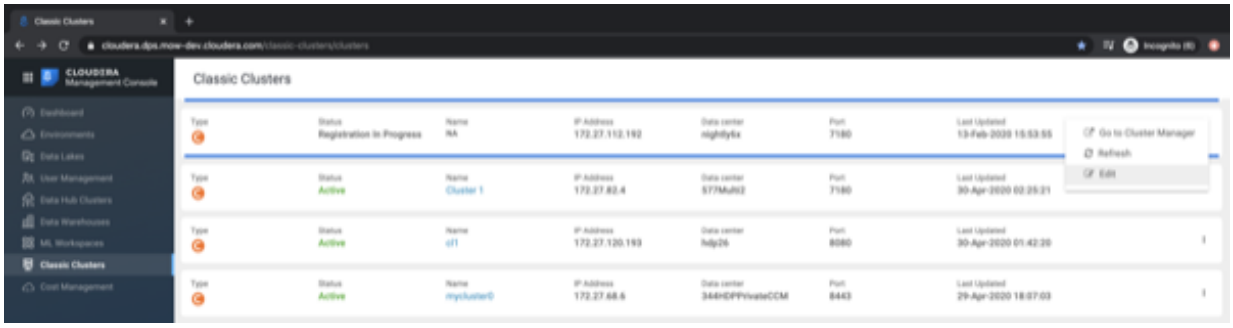

Classic Cluster displays the Edit dialog box.

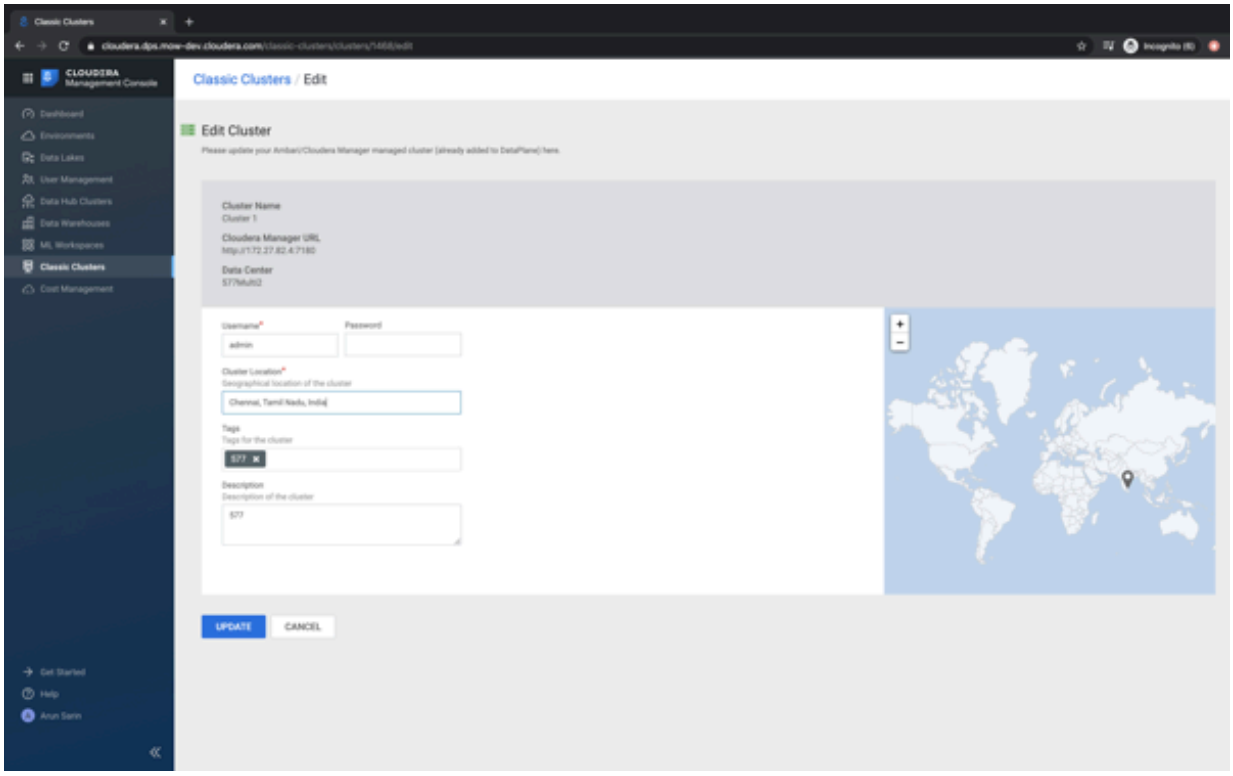

4. You can refresh cluster information for a registered cluster by choosing the Refresh option from the pull down menu next to the cluster entry in the list of clusters.

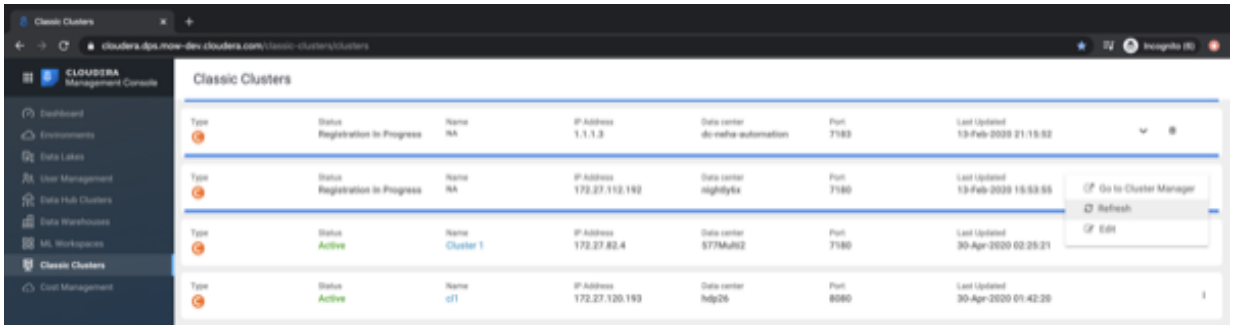

Upgrading a classic cluster from CCMv1 to CCMv2

Existing classic clusters (CDH, HDP, or Cloudera Private Cloud Base) registered in Cloudera Public Cloud's Management Console with Cloudera Private Cloud Base v1 (CCMv1) should be upgraded to use CCMv2.

Before you begin

Prior to upgrading, open ports for CCMv2. If you would like to use Cluster Connectivity Managerv2 (CCMv2), ensure that outgoing traffic is allowed on port 443 on the legacy CDH, HDP, or Cloudera Private Cloud Base cluster.

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. From the left navigation pane, select Classic clusters.
4. Find your previously registered classic cluster.
5. If your cluster need to be upgraded, you will see the icon next to its name. After clicking on this icon, you will see the Upgrade Now button, as shown in the following screenshot:

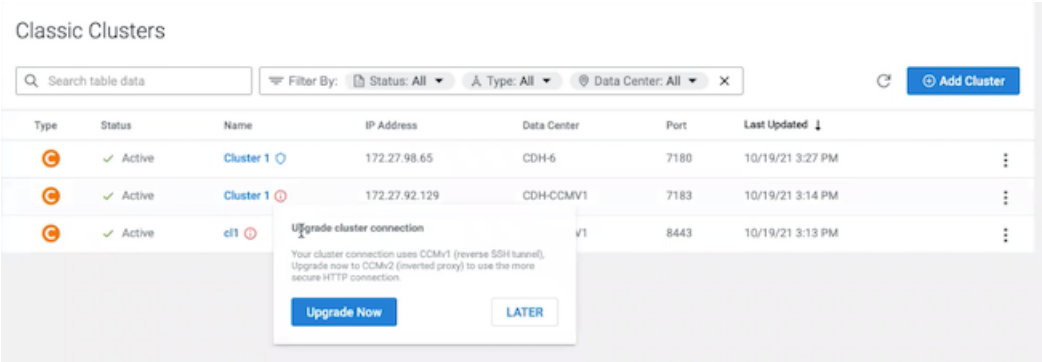

6. Click the Upgrade Now button to start the upgrade process.
7. Information about three upgrade steps is displayed:

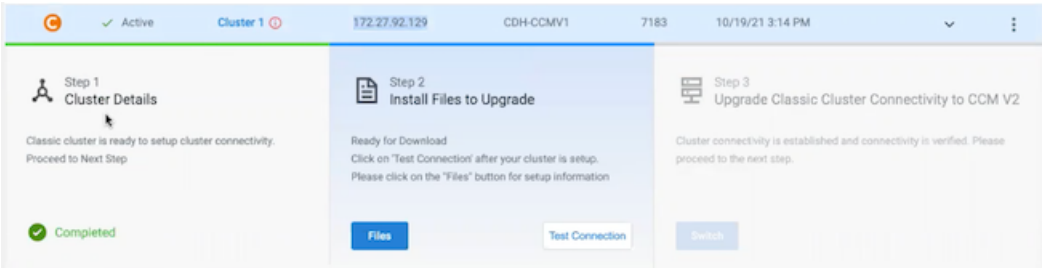

8. Cloudera automatically starts step 1. It normally takes 2-4 minutes. Once step 1 is marked as Completed, you can proceed to step 2.

9. In step 2:
   a) Click on Files.
   b) On the Setup Connectivity Client page, select the operating system of your cluster.
   c) Download the files mentioned under Files.
   d) Follow the installation steps described under How to Install. You will need to SSH to the host mentioned in these instructions and run the commands.
   e) Once done, click Close.
   f) Click on Test Connection.
   g) Once the connection is successful, step 2 is marked as Completed. You can proceed to step 3.
10. In step 3:
   a) Click on Switch.
   b) A pop-up window appears with instructions on how to clean up CCMv1 resources. Under Select OS, select your operating system and then follow the cleanup steps described for your cluster type. You need to SSH to the specific cluster node indicated in the instructions in order to perform the cleanup. The steps that you need to perform are as follows (The same steps are printed in the UI):

      - Centos7/RHEL7

        On a CDH or Cloudera Private Cloud Base cluster, run the following on the Cloudera Manager host:

        ```
        systemctl stop ccm-tunnel@CM.service
        yum remove autossh
        ```

        On an HDP cluster, run the following on the KNOX host:

        ```
        systemctl stop ccm-tunnel@KNOX.service
        yum remove autossh
        ```

      - Centos6/RHEL6

        On a CDH or Cloudera Private Cloud Base cluster, run the following on the Cloudera Manager host:

        ```
        service reverse-tunnel stop 'CM'
        yum remove autossh
        ```

        On an HDP cluster, run the following on the KNOX host:

        ```
        service reverse-tunnel stop KNOX
        yum remove autossh
        ```

11. Once your cluster has been upgraded the icon appears next to its name.
12. From the ICON menu of the upgraded cluster select Refresh to synchronize the cluster with Cloudera:

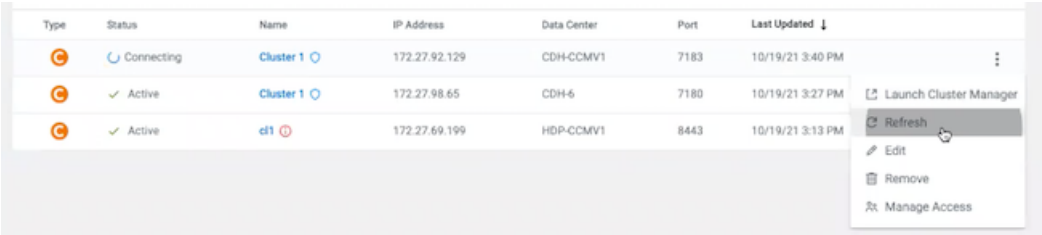

13. The cluster status changes to Active.

Results

Once you have performed the upgrade, you can continue using your cluster as usual.

Cloudera Management Console

Connection to Private Subnets

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cluster Connectivity Manager..... 4
Upgrading from Cluster Connectivity Manager v1 to Cluster Connectivity Manager v2..... 9
Outbound network access destinations for Cluster Connectivity Manager v2..... 10
Configuring a private VPC in AWS..... 11
Configuring a VNet with private IPs in Azure..... 12
Configuring a VPC with private IPs in GCP..... 12
Enabling Cluster Connectivity Manager in the Cloudera Management Console..... 13
Troubleshooting Cluster Connectivity Manager v1..... 13
Troubleshooting Cluster Connectivity Manager v2..... 15
Cloudera Data Warehouse and private networking..... 16

Public Endpoint Access Gateway.....16
Enable Public Endpoint Access Gateway for AWS..... 17
Enable Public Endpoint Access Gateway for Azure..... 19
Enable Public Endpoint Access Gateway for GCP..... 20

Azure Load Balancers in Data Lakes and Cloudera Data Hub clusters..... 21

Cluster Connectivity Manager

Cloudera can communicate with Data Lake, Cloudera Data Hub clusters, Cloudera data services workload clusters, and on-prem classic clusters that are on private subnets. This communication occurs via the Cluster Connectivity Manager. Cluster Connectivity Manager is available for Cloudera deployments on AWS, Azure, and GCP.

Note:

When you register a new AWS, Azure, or GCP environment in Cloudera via web interface, Cluster Connectivity Manager is enabled by default and the option to disable it has been removed from the web interface. Cluster Connectivity Manager is also enabled by default when registering an environment via CDP CLI.

Cluster Connectivity Manager enables the Cloudera Control Plane to communicate with workload clusters that do not expose public IPs. Communication takes place over private IPs without any inbound network access rules required, but Cloudera requires that clusters allow outbound connections to Cloudera Control Plane.

Cluster Connectivity Manager provides enhanced security for communication between customer workload clusters and the Cloudera Control Plane. A Cloudera environment supports public, private, and semi-private networks for running workloads. In a public network, Cloudera Control Plane initiates a connection to the nodes in a workload cluster; However, when using a private or semi-private environment, this option is not available due to the private nature of the subnet and some of the hosts. In such cases, Cluster Connectivity Manager is required to simplify the network configuration in the customer’s subnet.

Cluster Connectivity Manager implements an inverted proxy that initiates communication from the secure, private workload subnet to Cloudera Control Plane. With Cluster Connectivity Manager enabled, the traffic direction is reversed so that the private workload subnet does not require inbound access from Cloudera's network. In this setup, configuring security groups is not as critical as in the public network setup. All communication via Cluster Connectivity Manager is encrypted via TLS v1.2.

From a data security perspective, no data or metadata leaves the workload subnet. The Cluster Connectivity Manager connection is used to send control signals, logs and heartbeats, and communicate the health status of various components with the Cloudera Control Plane.

When deploying environments without public IPs, a mechanism for end users to connect to the Cloudera endpoints should already be established via a Direct Connection, VPN or some other network setup. In the background, the Cloudera Control Plane must also be able to communicate with the entities deployed in your private network.

The following diagram illustrates the Cloudera components that are responsible for communication via Cluster Connectivity Manager:

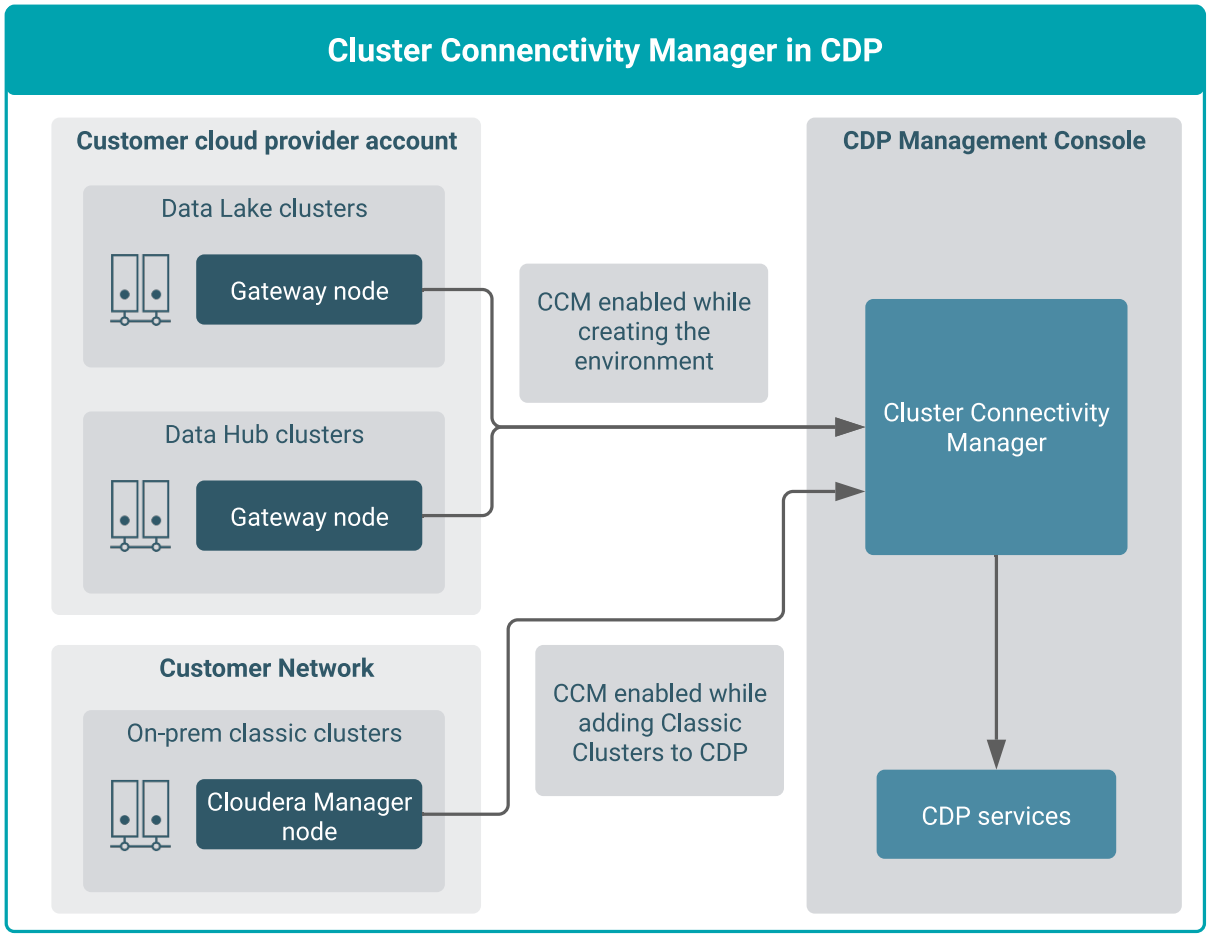

Cluster Connectivity Manager was initially released as Cluster Connectivity Manager v1 and later Cluster Connectivity Manager v2 was released to replace it. While Cluster Connectivity Manager v1 establishes and uses a tunnel based on the SSH protocol, with Cluster Connectivity Manager v2 the connection is via HTTPS. All new environments created with Cloudera Runtime 7.2.6 or newer use Cluster Connectivity Manager v2. Existing environments and new environments created with Runtime older than 7.2.6 continue to use Cluster Connectivity Manager v1. All newly registered classic clusters use Cluster Connectivity Manager v2, but previously registered classic clusters continue to use Cluster Connectivity Manager v1.

The following diagram illustrates connectivity to a customer account without using Cluster Connectivity Manager:

Figure 1: Connectivity to customer account withCluster Connectivity Manager disabled

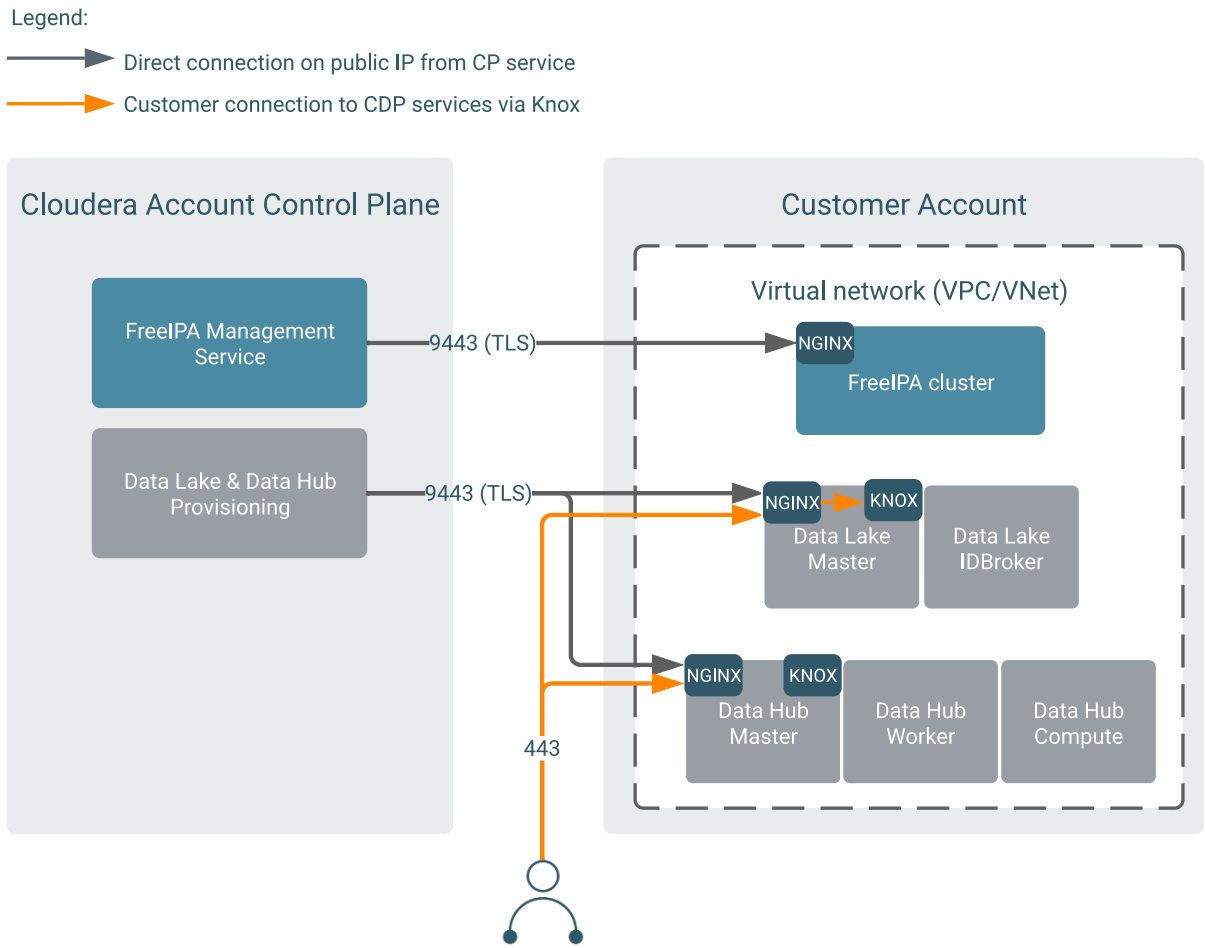

Cluster Connectivity Manager v2

Cluster Connectivity Manager v2 agents deployed on FreeIPA nodes initiate an HTTPS connection to the Cloudera Control Plane. This connection is then used for all communication thereafter. Data Lake and Cloudera Data Hub instances receive connections from the Cloudera Control Plane via the agents deployed onto FreeIPA nodes. This is illustrated in the diagram below.

Cluster Connectivity Manager v2 also supports classic clusters. You can use Cloudera Replication Manager with your on-premise CDH, HDP, and Cloudera Private Cloud Base clusters accessible via a private IPs to assist with data migration and synchronization to cloud storage by first registering your cluster using classic cluster registration.

When Cluster Connectivity Manager v2 is enabled, the traffic direction is reversed so the environment does not require inbound access from Cloudera's network. Since in this setup, inbound traffic is only allowed on the private subnets, configuring security groups is not as critical as in the public IP mode outlined in the previous diagram; However, in case of bridged networks it may be useful to restrict access to a certain range of private IPs.

The following diagram illustrates connectivity to a customer account using Cluster Connectivity Managerv2:

Figure 2: Connectivity to customer account withCluster Connectivity Manager v2 enabled

Cluster Connectivity Manager v2 with Token Authentication

In the current scheme of things, the communication between agent and Cloudera Control Plane services uses a two-way SSL or client certificate based authentication mechanism.

In order to enable traffic inspection which could further pave the way for traffic monitoring and anomaly detection in traffic, the communication between agent and Cloudera Control Plane can optionally be configured to use a combination of TLS (to validate the server) and bespoke validation (to validate the client).

This approach does away with the client certificate based agent authentication on the Cloudera Control Plane side and instead uses request signing and authorization to validate incoming requests from the Cluster Connectivity Manager agent.

Cluster Connectivity Manager v1

The below diagram illustrates the Cloudera connectivity to a customer account with Cluster Connectivity Manager v1 enabled. Cluster Connectivity Manager v1 agents are deployed not only on the FreeIPA cluster (like in Cluster Connectivity Manager v2), but also on the Data Lake and Cloudera Data Hub. While Cluster Connectivity Manager v2 establishes a connection via HTTPS, Cluster Connectivity Manager v1 uses a tunnel based on the SSH protocol. Workload clusters initiate an SSH tunnel to the Cloudera control plane, which is then used for all communication thereafter.

Figure 3: Connectivity to customer account with Cluster Connectivity Manager v1 enabled

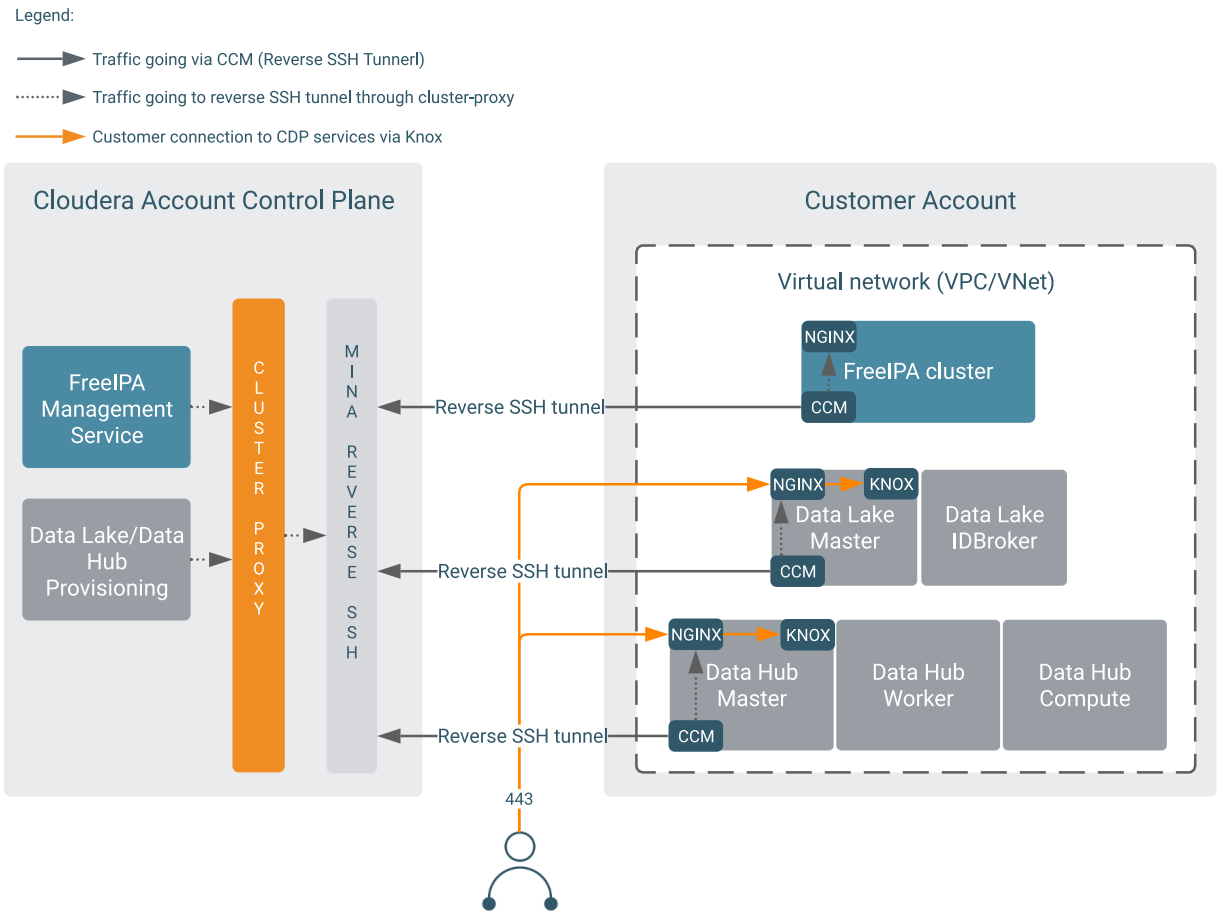

Supported services

The following Cloudera services are supported by Cluster Connectivity Manager:

Cluster Connectivity Managerv2

Supports environments with Cloudera Runtime 7.2.6+

Cloudera service	AWS	Azure	GCP
Data Lake	GA	GA	GA
FreeIPA	GA	GA	GA
Cloudera Data Engineering	Preview	Preview	
Cloudera Data Hub	GA	GA	GA
Cloudera Data Warehouse		GA	
Cloudera DataFlow	GA	GA	
Cloudera AI	GA	GA	
Cloudera Operational Database	GA	GA	GA

Cluster Connectivity Manager v1

Supports environments with Cloudera Runtime <7.2.6 and environments created prior to Cluster Connectivity Manager v2 GA.

Cloudera service	AWS	Azure	GCP
Data Lake	GA	GA	GA
FreeIPA	GA	GA	GA
Cloudera Data Engineering			
Cloudera Data Hub	GA	GA	GA
Cloudera Data Warehouse			
Cloudera DataFlow			
Cloudera AI			
Cloudera Operational Database			

To learn more about Cluster Connectivity Manager, refer to the following documentation:

Upgrading from Cluster Connectivity Manager v1 to Cluster Connectivity Manager v2

Upgrading from Cluster Connectivity Manager v1 to Cluster Connectivity Manager v2 is available for customers who created their environments with Cluster Connectivity Manager v1. The upgrade requires no downtime. If your environment needs to be updated, you will see a notification printed in your environment details.

Both Cluster Connectivity Manager v1 and Cluster Connectivity Manager v2 provide the same functionality but version 2 is an improvement over version 1. Cluster Connectivity Manager v2 is more efficient in communicating with the Cloudera Control Plane and has a smaller footprint on the customer network.

The differences between Cluster Connectivity Manager v2 and Cluster Connectivity Manager v1 are related to improving reliability and efficiency. These differences are as follows:

- Cluster Connectivity Manager v2 uses fewer agents (its agents are on FreeIPA nodes only), whereas Cluster Connectivity Manager v1 has a larger footprint, with its agents running across not only on FreeIPA nodes but also on Data Lake and Cloudera Data Hub nodes.
- Cluster Connectivity Manager v2 uses the HTTPS protocol while Cluster Connectivity Manager v1 is based on the SSH protocol. The use of HTTPS makes CCMv2 much more customer firewall friendly; it uses standard outbound ports which are allowed in almost all customer networks.
- Since Cluster Connectivity Manager v2 uses HTTPS, it is more proxy-friendly. Customers can set up a proxy server such as Squid and inspect the underlying traffic to be more confident about the data that is being sent to the Cloudera Control Plane.
- Cluster Connectivity Manager v2 uses token authentication.
- Newer Cloudera services, such as Cloudera AI, Cloudera Data Engineering, Cloudera DataFlow, Cloudera Data Warehouse, and others, only support Cluster Connectivity Manager v2.

Before you begin

Prior to upgrading Cluster Connectivity Manager, you should ensure that:

1. You are using the latest FreeIPA image. To upgrade your FreeIPA, follow the steps described in Upgrade FreeIPA.
2. You have the network configured so that Cluster Connectivity Manager v2 endpoints are available. See ”Cloudera Cluster Connectivity Manager v2” in the AWSAzureGCP outbound network access destinations.

Steps

1. If your environment should be updated from Cluster Connectivity Manager v1 to Cluster Connectivity Manager v2, you will see a "Time to upgrade Cluster Connectivity Manager version" notification similar to the following in your environment details:

2. Upgrade Cluster Connectivity Manager either via Cloudera web interface or CLI:
   a. Cloudera web interface: Navigate to the Cloudera Management Console > Environments and click on the tile corresponding to a specific environment. Trigger the upgrade by clicking on the Upgrade button present on the UI element.
   b. CDP CLI: Use the following command:

```
cdp environments upgrade-ccm --environment <ENV_NAME_OR_CRN>
```

3. The status of your environment changes while the upgrade is pending. All environment components get upgraded (FreeIPA, Data Lake, and then Cloudera Data Hub clusters). Check the event history to confirm that the upgrade has been completed. Once the upgrade has been completed, the status of the environment changes back to “Running”. If an error occurs during the process, Cloudera rolls back the upgrade and reverts the environment to Cluster Connectivity Manager v1.

Outbound network access destinations for Cluster Connectivity Manager v2

When using Cluster Connectivity Manager in an environment with limited internet access or proxy, make sure to add the following outbound network access destinations to the allow list.

Outbound network access destinations for Cluster Connectivity Manager v2

Description/Usage	Cloudera Service	Destination	Protocol & Authentication	IP Protocol / Port	Comments
Cloudera Cluster Connectivity Manager v2 Persistent Cloudera Control Plane connection	All services	US-based Cloudera Control Plane: *.v2.us-west-1.ccm.cdp.cloudera.com 35.80.24.128/27 EU-based Cloudera Control Plane: *.v2.ccm.eu-1.cdp.cloudera.com 3.65.246.128/27 AP-based Cloudera Control Plane: *.v2.ccm.ap-1.cdp.cloudera.com 3.26.127.64/27	HTTPS with mutual authentication	TCP/443	Multiple long-lived/ persistent connections

Description/Usage	Cloudera Service	Destination	Protocol & Authentication	IP Protocol / Port	Comments
Cloudera Cluster Connectivity Manager v1 Persistent Cloudera Control Plane connection	All services	*.ccm.cdp.cloudera.com 44.234.52.96/27	SSH public/private key authentication	TCP/6000-6049	One connection per cluster configured; persistent.

Note: If you have existing environments using Cluster Connectivity Manager v1, you shouldn't remove the previously added Cluster Connectivity Manager v1 specific outbound rules (ports 6000-6049).

Configuring a private VPC in AWS

When you create your Cloudera environment, you have two options: Have Cloudera set up your private network and security groups or set up the VPC with your private IPs and security groups. Either way, you must enable Cluster Connectivity Manager.

Have Cloudera set up a private VPC network and security groups

If you choose to have Cloudera create your private network and security groups when you are testing or sandboxing Cluster Connectivity Manager, when you enable Cluster Connectivity Manager, Cloudera performs the following:

- Creates 3 public and 3 private subnets
- Creates the Data Lake and Cloudera Data Hub clusters in the private subnet.

The public subnets have an internet gateway attached. The private subnets have a NAT gateway attached. When Cloudera creates the security group, it opens two ports to the CIDR range that you specify, port 22 and port 443. Use these ports only to access these clusters. For the list of security group rules that Cloudera creates, see Default security group settings on AWS.

Set up your own private VPC network and security groups

If you choose to configure your own VPCs with private IPs, you will need the following:

- At least two private subnets in at least two different availability zones (AZs).
- Outbound traffic via the HTTP secure connection initiated by Cluster Connectivity Manager to the Cloudera hosted NLBs on workload nodes.

In the AWS console, configure the following:

1. Create one public subnet and place the NAT gateway there to allow outbound connectivity in the private subnet.
2. Assign an internet gateway to the public subnet.
3. All inbound traffic must be on private subnets.
4. Create three private subnets. The private subnets must be in different availability zones (AZs).
5. Route the private subnet to the NAT gateway.
6. You must configure outbound traffic for Cloudera resources.
7. The workload clusters containing Cluster Connectivity Manager (Knox, master, or Cloudera Manager for Classic Cluster) must be able to reach the Network Load Balancers (NLBs). You can use port 443 to connect to the NLBs.

Create your security groups as described in Security groups.

Configuring a VNet with private IPs in Azure

When you create your Cloudera environment without public IPs, you have two options: Have Cloudera set up your private network and security groups, or set up the VNet with your private IPs and security groups. Either way, you must enable Cluster Connectivity Manager.

Have Cloudera set up a VNet with private IPs and security groups

If you choose to have Cloudera create your private network and security groups when you are testing or sandboxing Cluster Connectivity Manager, when you enable Cluster Connectivity Manager, Cloudera performs the following:

- Creates more than 30 subnets. Azure does not distinguish between public and private subnets. By default they are all private.
- When Cloudera creates the security group, it opens two ports to the CIDR range that you specify, port 22 and port 443. Use these ports only to access these clusters. For a list of security group rules that Cloudera creates, see Default security group settings on Azure.

Set up your own VNet with private IPs and security groups

If you choose to configure your own VNets without public IPs, you will need to configure the following in your Azure Portal:

1. Create at least one virtual network subnet. See VNet and subnet planning to determine the exact number of subnets needed.
2. You must configure outbound traffic for Cloudera resources.
3. The workload clusters containing Cluster Connectivity Manager (Knox, master, or Cloudera Manager for Classic Cluster) must be able to reach the Network Load Balancers (NLBs).
4. You can use port 443 to connect to the NLBs.
5. Create your security groups as described in Network security groups.

Configuring a VPC with private IPs in GCP

Prior to registering your GCP environment in Cloudera, you should set up a VPC network with private IPs, and create firewall rules.

You need the following:

- At least one subnet for hosts that will use Cluster Connectivity Manager.
- Outbound traffic via the HTTP connection initiated by Cluster Connectivity Manager should be allowed to the Cloudera hosted Network Load Balancers (NLBs) on workload nodes.

In the Google Cloud console, configure the following:

1. Create a VPC with custom (preferred) subnet configuration in the desired GCP region.
2. Create a GCP cloud router in the desired region.
3. Create a GCP NAT gateway specifying the previously created GCP cloud router.
4. You must configure outbound traffic for Cloudera resources.
5. The workload clusters containing Cluster Connectivity Manager (Knox, master, or Cloudera Manager for Classic Clusters) must be able to reach the Network Load Balancers (NLBs).
6. You can use port 443 to connect to the NLBs.
7. Create your firewall rules as described in Firewall rules.

Enabling Cluster Connectivity Manager in the Cloudera Management Console

When you register a new AWS, Azure, or GCP environment in Cloudera via web interface, Cluster Connectivity Manager is enabled by default and the option to disable it has been removed from the web interface. Cluster Connectivity Manager is also enabled by default when registering an environment via CDP CLI.

Related Information

Register an AWS environment from the Cloudera UI

Register an AWS environment from the CDP CLI

Register an Azure environment from the Cloudera UI

Register an Azure environment from the CDP CLI

Register a GCP environment from the Cloudera UI

Register a GCP environment from the CDP CLI

Troubleshooting Cluster Connectivity Manager v1

You can troubleshoot cluster connection issues that might occur when you use Cluster Connectivity Manager v1 by referring to the syslog on your Cloudera Manager node for classic clusters and your gateway and Knox nodes for Data Lake and Cloudera Data Hub clusters.

Search for auto-ssh messages to diagnose any connection problems.

For example, Cloudera can communicate with clusters that are on private subnets with only private IPs without any additional network configuration or setup. However, Cloudera requires that these clusters have outbound connections to AWS NLBs hosted in the Cloudera 's AWS account.

Common cases when connection via Cluster Connectivity Manager v1 fails

The following list includes the most common cases when connection via Cluster Connectivity Manager fails and steps for how to fix the problems.

In all the cases listed below, it is essential to check if the Cluster Connectivity Manager domain name “.ccm.cdp.cloudera.com” has been whitelisted and the port range 6000-6049 has been whitelisted for SSL connections. This is because by default SSL connections are allowed only on the port 443 but connections to the Cloudera Control Plane via Cluster Connectivity Manager go through the ports in the range 6000-6049.

Reason for connection failure	How to fix the problem
You have not whitelisted the NLB domains or the port range 6000-6049 on your VPC.	Both the incoming and the outgoing traffic must be allowed from Cluster Connectivity Manager NLB and port ranges.
You are using a transparent proxy and you did not explicitly whitelist NLB domains and the ports on the transparent proxies.	If you are using a transparent proxy, you need to explicitly whitelist the NLB domains and the ports on the transparent proxies.
You are using a non-transparent proxy but you did not register it in Cloudera.	If you are using a non-transparent proxy, you need to perform the steps in Setting up a non-transparent proxy in Cloudera documentation.
Your firewall doesn’t allow traffic to/from NLBs.	The firewall configured on your VPC should allow the traffic to/from the NLBs and the port range 6000-6049.
The IPs of the NLB do not fall under the elastic IP CIDR block assigned to Cluster Connectivity Manager NLBs.	Cluster Connectivity Manager NLBs have IPs assigned to them which are taken from the CIDR block 44.234.52.96/27. If the NLB assigned to the environment/cluster returns IPs that do not fall under this CIDR block when an nslookup call is made, then the IPs returned need to be explicitly whitelisted so that the traffic from these IPs is allowed in your VPCs.

If you are still unable to determine the root cause and fix the issue, you may consider collecting more information.

Collecting information from the cluster where connection fails

Obtaining relevant information such as the NLB domain name, tunnel type (for example, Cloudera Manager), autossh client logs, and the syslogs from a cluster helps narrow down the root cause.

The following steps explain how to obtain this information.

1. Check if the autossh process is running on the cluster:
   a. SSH into the node on which the issue is being seen. Execute ps -aux | grep autossh to see if the autossh client responsible for creating the tunnel from the node to Cloudera is running.
   b. If the process is running, it would show the command similar to:

```
autossh -M 0 -o ServerAliveInterval 30 -o ServerAliveCountMax 3 -o UserK
nownHostsFile=/etc/ccm/ccm.pub -N -T -R 0.0.0.0:0:localhost:<LOCAL_PORT>
 -i /tmp/pk.key -p <NLB_PORT> <INITIATOR_ID>@<NLB_HOST_NAME> -vvv
```

   c. Collect the values of NLB_HOST_NAME, NLB_PORT, INITIATOR_ID from the above step.
   d. Share the collected information with Cloudera support.
2. Check if the NLB is reachable from the node:
   a. On the node, execute the command nslookup <NLB_HOST_NAME> to check if the NLB is reachable from the node.
   b. Use tools such as telnet or netcat to check if it is possible to reach the NLB on the specific NLB_PORT. For example:

```
telnet <NLB_HOST_NAME> <NLB_PORT>
```

   c. If telnet is not installed on the node, you can use curl telnet as an alternative. For example:

```
curl -vv telnet://<NLB_HOST_NAME>:<NLB_PORT>
```

   d. If this fails, then the traffic to the Cloudera network is blocked on your VPC. This means that you should check the outbound rules on their VPC to make sure that the traffic to the cloudera network is allowed.
3. Collect the following logs and share them with Cloudera support:
   a. The dump of /var/log/messages (which stores the global system messages) is necessary to debug the issue with the SSH tunnel setup by Cluster Connectivity Manager.
   b. The dump of the output of the tcpdump command on the problematic node captures the network messages with the description of contents of packets between the node and the Cloudera Control Plane. Cloudera support may ask for this if needed or when the already collected information is not sufficient to debug the issue.
   c. The dump of /var/log/socket_wait_cleanup.log (which stores the logs related to the restarts of autossh process) Note that this log file is only present in environments created after December 12, 2020.
4. Collect the firewall logs and share them with Cloudera support:
   a. It is useful for Cloudera support to see the logs from your firewall to see if the firewall is closing the connection to the Cloudera Control Plane.
5. Collect and share the environment/cluster and Cluster Connectivity Manager related information:
   a. To look up the logs, Cloudera support needs the account id of the customer, environment CRN, and the time period to scan the logs for the issue.

FreeIPA sync issues

Error or issue details	Resolution

502 or "Bad Gateway" error such as:
• "Bad Gateway, cause=com.cloudera.cdp.cm.ApiException: Bad Gateway"
• Unexpected response from FreeIPA; details: code: 502
• Status: 502 Bad Gateway Response
	A new parameter defining connection timeout was added to the autossh script placed on images used to provision clusters. Without it, there is no timeout for Cluster Connectivity Manager connections and Cluster Connectivity Manager related services won't restart automatically. If you provisioned FreeIPA or Data Lake prior to the implementation of this parameter,you need to update the script by using the following steps:
	1. Modify autossh command on FreeIPA, Data Lake and Cloudera Data Hub nodes in this script:
	`/cdp/bin/reverse-tunnel.sh`
	Add new parameter to the command:
	`-o "ConnectTimeout 30"`
	Original command:
	`exec autossh -M 0 -o "ServerAliveInterval 30" -o "ServerAliveCountMax 3" -o UserKnownHostsFile=${CCM_PUBLIC_KEY_FILE} -N -T -R ${LOCAL_IP}:0:localhost:${CCM_TUNNEL_SERVICE_PORT} -i ${PRIVATE_KEY} -p ${CCM_SSH_PORT} ${USER}@${CCM_HOST} -vvv`
	New command:
	`exec autossh -M 0 -o "ConnectTimeout 30" -o "ServerAliveInterval 30" -o "ServerAliveCountMax 3" -o UserKnownHostsFile=${CCM_PUBLIC_KEY_FILE} -N -T -R ${LOCAL_IP}:0:localhost:${CCM_TUNNEL_SERVICE_PORT} -i ${PRIVATE_KEY} -p ${CCM_SSH_PORT} ${USER}@${CCM_HOST} -vvv`
	2. Restart ccm-tunnel@KNOX service on Data Lake and Cloudera Data Hub nodes and ccm-tunnel@GATEWAY on FreeIPA, Data Lake and Cloudera Data Hub nodes:
	`systemctl restart ccm-tunnel@KNOX`
	`systemctl restart ccm-tunnel@GATEWAY`
	3. Check if modification of the command was successful:
	`ps aux | grep autossh`

Troubleshooting Cluster Connectivity Manager v2

This page lists common issues related to troubleshooting workload connectivity using Cluster Connectivity Manager v2.

Incorrect network setup

Reason for connection failure	How to fix the problem
For Cluster Connectivity Manager to work, the agent must be able to connect to its counterpart running inside the Cloudera Control Plane. A misconfigured network often blocks this path. This can happen in following ways: • Customer's network does not resolve Cloudera Data Hub DNS addresses. • Customer's network does not allow outbound connections. • Customer's network does not allow outbound connections to Cloudera IPs. • Customer's network does not allow outbound connections to the required ports. • Customer expects us to use a non-transparent proxy which does not allow connection to the required Cloudera IPs / ports. • Customer expects us to use a non-transparent proxy which does deep-packet inspection to sniff underlying traffic and blocks Cluster Connectivity Manager traffic.	Running the cdp-telemetry doctor command helps diagnose most problems of this category. See Collecting workload logs on page 16. Fixing the problem usually means ensuring that you set up the outbound network access destinations that are listed as Cloudera prerequisites. See: • AWS outbound network access destinations • Azure outbound network access destinations • GCP outbound network access destinations

Collecting workload logs

The easiest and recommended way to collect logs is to use the cdp-telemetry doctor tool.

The ccm module of the cdp-telemetry doctor tools collects all the information required on a given node.

```
cdp-telemetry doctor ccm status
cdp-telemetry doctor ccm report
```

Note: Using cdp-telemetry doctor should report all the information Cloudera support requires on the cluster for debugging. However, the tool needs to be launched on every node where an agent is expected to run.

To generate the diagnostic bundle, you need to ssh into each node of concern and invoke the following command:

```
cdp-telemetry doctor ccm report
```

Errors related to classic clusters

See Troubleshooting classic cluster registration errors.

Cloudera Data Warehouse and private networking

Cloudera Data Warehouse service supports private deployments in AWS, which use private subnets.

For information on setting up an AWS environment with private subnet support, refer to Setting up private networking in AWS environments documentation. in Cloudera Data Warehouse documentation.

Public Endpoint Access Gateway

If the network into which you are deploying your Cloudera environment does not have pre-established connectivity with your corporate network, enabling the Public Endpoint Access Gateway can reduce the complexity users face when interacting with the Cloudera endpoints.

The recommended way to deploy production-ready Cloudera environments is to deploy them on private networks, but this additional security makes it difficult for users to access UIs and APIs without configuring complex network connectivity between users and internal cloud provider networks. The Public Endpoint Access Gateway provides secure connectivity to UIs and APIs in Data Lake and Cloudera Data Hub clusters deployed using private networking,

allowing users to access these resources without complex changes to their networking or creating direct connections to cloud provider networks.

You can enable the Public Endpoint Access Gateway when registering your AWS, Azure, or GCP environment in Cloudera. The gateway interfaces the Knox service, which is automatically integrated with your identity provider configured in Cloudera, allowing you to authenticate using your SSO credentials without any additional configuration. All communication with the gateway is over TLS, so connections are secure. You can control the IP ranges from where connections to the gateway can be established by configuring your security groups.

The following diagram illustrates this setup:

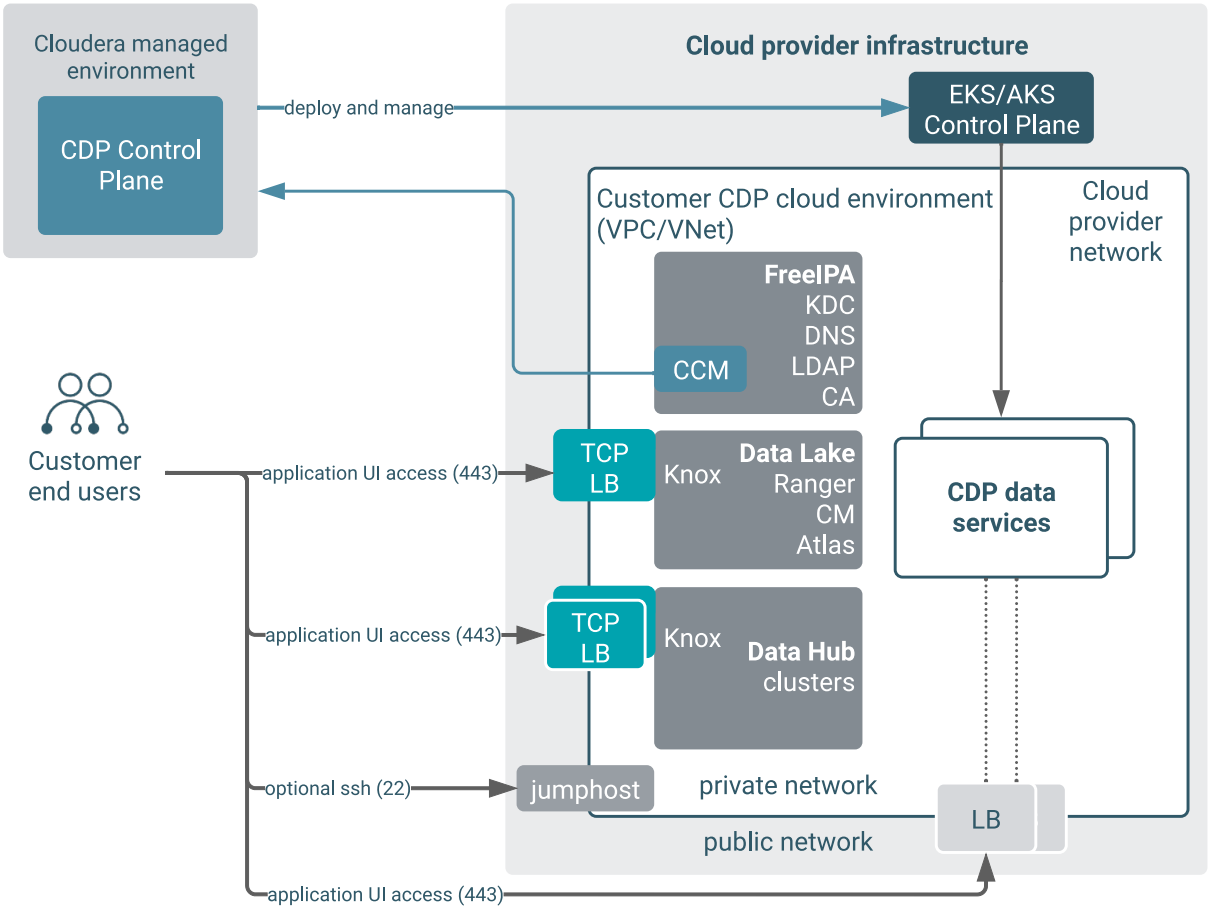

Note: The gateway provides secure connectivity to UIs and APIs. All Knox-enabled endpoints are supported. The gateway does not cover SSH or data access paths (such as Kafka Broker and NiFi Site2Site endpoints). Cloudera recommends that you set up connectivity between private networks in the public cloud and internal customer networks for secure and fast Kafka and NiFi deployments.

Enable Public Endpoint Access Gateway for AWS

You can enable Public Endpoint Access Gateway during AWS environment registration.

Once activated, the gateway will be used for the Data Lake and all the Cloudera Data Hub clusters within the environment. There is no way to activate it on a per Data Lake or per Cloudera Data Hub cluster level. Once it is enabled for an environment, there is no way to deactivate it. The gateway can be used either with an existing VPC or with a new VPC created by Cloudera.

Prerequisites

- If you choose to enable Public Endpoint Access Gateway, Cloudera will create two AWS network load balancers (AWS NLB) per cluster (that is for each Data Lake and Cloudera Data Hub cluster). Make sure that your AWS NLB limits allow for the load balancer creation.
- If you are using your existing network, you should have at least 2 public subnets in the VPC that you would like to use for Cloudera. The availability zones of the public and private subnets must match.

Steps

For CDP UI

When registering your AWS environment, make sure to do the following:

1. On the Region, Networking, and Security page, under Network, select your existing VPC or select to have a new VPC created.
2. If you selected an existing VPC, select at least two existing private subnets (or at least three subnets if you would like to provision Cloudera Data Warehouse instances).
3. Click on Enable Public Endpoint Access Gateway to enable it. This enables UIs and APIs of the Data Lake and Cloudera Data Hub clusters to be accessible over the internet.
4. If you selected an existing VPC, under Select Endpoint Access Gateway Subnets, select the public subnets for which you would like to use the gateway. The availability zones of the public subnets must be the same as the availability zones of the private subnets selected under Select Subnets.
5. Under Security Access Settings, make sure to restrict access to only be accepted from sources coming from your external network range.

   Note: The security access settings do not apply to the network load balancer used by the Public Endpoint Access Gateway, but they apply to the instances that are running in private subnets and to which the Public Endpoint Access Gateway routes traffic. Therefore the security access settings should allow the users' public IP ranges to be able to connect through the public load balancer.
6. Finish registering your environment.

For CDP CLI

During environment registration via CDP CLI, you can optionally enable public endpoint access gateway using the following CLI parameters:

```
--endpoint-access-gateway-scheme PUBLIC
--endpoint-access-gateway-subnet-ids subnet-0232c7711cd864c7b subnet-05
d4769d88d875cda
```

The first parameter enables the gateway and the second one allows you to specify public subnets. The availability zones of the public subnets must be the same as the availability zones of the private subnets specified under --subnet-ids. For example:

```
cdp environments create-aws-environment \
--environment-name gk1dev \
--credential-name gk1cred \
--region "us-west-2" \
--security-access cidr=0.0.0.0/0 \
--authentication publicKeyId="gk1" \
--log-storage storageLocationBase=s3a://gk1priv-cdp-bucket,instanceProf
ile=arn:aws:iam::152813717728:instance-profile/mock-idbroker-admin-role \
--vpc-id vpc-037c6d94f30017c24 \
--subnet-ids subnet-0232c7711cd864c7b subnet-05d4769d88d875cda \
--endpoint-access-gateway-scheme PUBLIC \
--endpoint-access-gateway-subnet-ids subnet-0232c7711cd864c7b subnet-05d47
69d88d875cda \
--free-ipa instanceCountByGroup=1 \
```

Equivalent CLI JSON for an environment request looks like this:

```
"endpointAccessGatewayScheme": "PUBLIC",
"endpointAccessGatewaySubnetIds":
       ["subnet-0232c7711cd864c7b",
       "subnet-05d4769d88d875cda"],
```

Enable Public Endpoint Access Gateway for Azure

You can enable Public Endpoint Access Gateway during Azure environment registration.

Once activated, the gateway will be used for the Data Lake and all the Cloudera Data Hub cluster s within the environment. There is no way to activate it on a per Data Lake or per Cloudera Data Hub cluster level. Once it is enabled for an environment, there is no way to deactivate it. The gateway can be used either with an existing VNet or with a new VNet created by Cloudera.

If you choose to enable Public Endpoint Access Gateway, Cloudera will create two Azure load balancers per cluster (that is, two for each Data Lake and Cloudera Data Hub cluster).

Steps

For CDP UI

When registering your Azure environment, make sure to do the following:

1. On the Region, Networking, and Security page, under Network, select your existing VNet or select to have a new VNet created.
2. If you selected an existing VNet, select at least one existing private subnet (or at least three subnets if you would like to provision Cloudera Data Warehouse instances).
3. Click on Enable Public Endpoint Access Gateway to enable it. This enables UIs and APIs of the Data Lake and Cloudera Data Hub cluster s to be accessible over the internet.
4. Under Security Access Settings, make sure to restrict access to only be accepted from sources coming from your external network range.

   Note: The security access settings do not apply to the load balancer used by the Public Endpoint Access Gateway, but they apply to the instances that are running in private subnets and to which the Public Endpoint Access Gateway routes traffic. Therefore the security access settings should allow the users' public IP ranges to be able to connect through the public load balancer.
5. Finish registering your environment.

For CDP CLI

During Azure environment registration via CDP CLI, you can optionally enable public endpoint access gateway using the --endpoint-access-gateway-scheme CLI parameter. For example:

```
cdp environments create-azure-environment
...
--endpoint-access-gateway-scheme PUBLIC
```

Equivalent CLI JSON for an environment request looks like this:

```
cdp environments create-azure-environment
...
"endpointAccessGatewayScheme": "PUBLIC"
```

Enable Public Endpoint Access Gateway for GCP

You can enable Public Endpoint Access Gateway during GCP environment registration.

Once activated, the gateway will be used for the Data Lake and all the Cloudera Data Hub cluster s within the environment. There is no way to activate it on a per Data Lake or per Cloudera Data Hub cluster level. Once it is enabled for an environment, there is no way to deactivate it.

If you choose to enable Public Endpoint Access Gateway, Cloudera will create two Google Cloud Load Balancers (GCLB) per cluster (that is, two for each Data Lake and two for each Cloudera Data Hub cluster).

Prerequisites

If you would like to use this feature, make sure that "Private Google Access" is disabled on at least one subnet in the VPC.

Steps

For CDP UI

When registering your GCP environment, make sure to do the following:

1. On the Region, Networking, and Security page, under Network, select your existing VPC network.
2. Select at least one existing private subnet.
3. Click on Enable Public Endpoint Access Gateway to enable it. This enables UIs and APIs of the Data Lake and Cloudera Data Hub cluster s to be accessible over the internet.

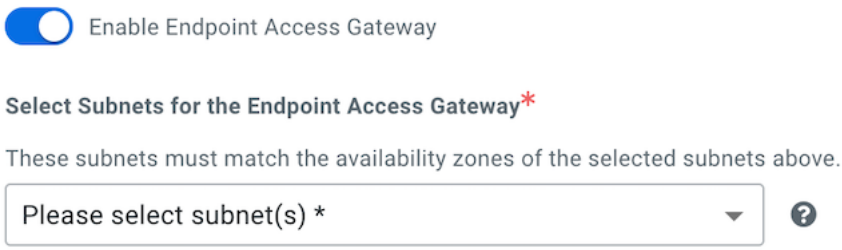

4. If you selected an existing VPC, under Select Endpoint Access Gateway Subnets, select the public subnets for which you would like to use the gateway. The availability zones of the public subnets must be the same as the availability zones of the private subnets selected under Select Subnets.
5. Under Security Access Settings, make sure to restrict access to only be accepted from sources coming from your external network range.

   Note: The security access settings do not apply to the load balancer used by the Public Endpoint Access Gateway, but they apply to the instances that are running in private subnets and to which the Public Endpoint Access Gateway routes traffic. Therefore the security access settings should allow the users' public IP ranges to be able to connect through the public load balancer.
6. Finish registering your environment.

For CDP CLI

During GCP environment registration via CDP CLI, you can optionally enable Public Endpoint Access Gateway using the following CLI parameter:

```
--endpoint-access-gateway-scheme PUBLIC
```

For example:

```
cdp environments create-gcp-environment
 ...
 --endpoint-access-gateway-scheme PUBLIC
```

Equivalent CLI JSON for an environment request looks like this:

```
cdp environments create-gcp-environment
...
"endpointAccessGatewayScheme": "PUBLIC"
```

Azure Load Balancers in Data Lakes and Cloudera Data Hub clusters

The Azure Load Balancer is used in multiple places in Cloudera Data Lakes and Cloudera Data Hub clusters. It is used as a frontend for Knox in both Data Lakes and Cloudera Data Hub clusters, and for Oozie HA in HA Cloudera Data Hub clusters.

The following table describes all use cases where an Azure Load Balancer is used in Data Lakes and Cloudera Data Hub clusters running on Azure:

Cloudera component	Azure Load Balancer use case
Data Lake	A load balancer is configured in front of Knox for Data Lakes of all shapes.
HA Cloudera Data Hub	A load balancer is configured for all Cloudera Data Hub clusters created from a default template where Knox and/or Oozie is running in HA mode. This can be overridden by setting the “enableLoadBalancer” setting in a custom template to “false”.
An environment with Public Endpoint Access Gateway enabled	When the Endpoint Gateway is enabled, Load balancers are created in front of Knox for all Data Lakes and Cloudera Data Hub clusters attached to the environment.

In the event that a Data Lake or Cloudera Data Hub cluster uses private networks (meaning the “Create Public IPs” option is disabled during environment creation and the Public Endpoint Access Gateway is not enabled), an internal load balancer is created for ingress traffic to Knox in all Data Lakes and in Knox HA Cloudera Data Hub clusters running in that environment.

Because Cloudera uses a Standard SKU Azure Load Balancer, the private load balancer does not allow for public egress, and a public load balancer is required for public outbound destinations as described in the Azure network reference architecture. If you are creating an environment with an existing network of your own configuration, which is assumed to be fully private (the "Create Public IPs" option and Public Endpoint Access Gateway are disabled), it is your responsibility to create egress connectivity for the required subnets in your VNet. This can be accomplished through a NAT gateway setup or user-defined routing.

If you want Cloudera to create a secondary load balancer for egress in an existing network of your own configuration, be aware that it requires certain public IP permissions that are granted as part of the required Azure permissions. However, you can create this secondary public egress load balancer when you create an Azure environment through the CLI with the options –no-enable-outbound-load-balancer and –enable-outbound-load balancer. The secondary public egress load balancer created by Cloudera has only outbound rules defined, and does not handle ingress traffic.

This is illustrated in the following diagram:

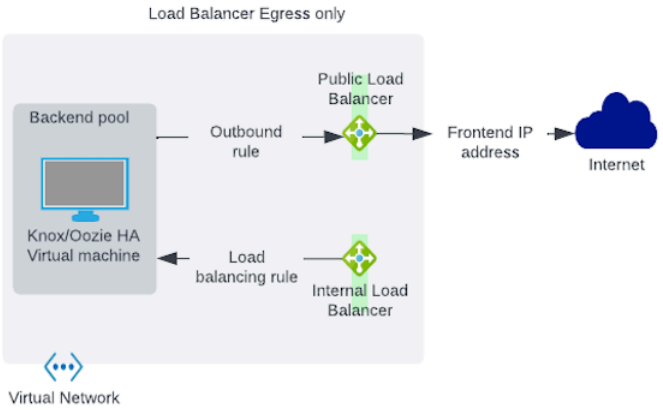

If you are creating a new network during environment registration, Cloudera ensures that egress connectivity is available. If the "Create Public IPs" option and Public Endpoint Access Gateway are disabled in your network, a separate load balancer is created for egress, though this load balancer requires certain public IP permissions that are granted as part of the required Azure permissions. If either "Create Public IPs" or Public Endpoint Access Gateway is enabled, then a public load balancer is created to handle both public ingress to port 443 and public egress.

Note: When an Azure VNet has both a NAT Gateway and a public load balancer for outbound connectivity, the NAT Gateway takes precedence. For more information, see Design virtual networks with NAT gateway.

Cloudera Management Console

AWS Credentials

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Working with AWS credentials 4

AWS credential overview 4

Create AWS credential 5

Obtain CLI commands 6

Viewing available credentials 7

Modify a credential 7

Change environment's credential 8

Delete a credential 8

Working with provisioning credentials for AWS

Refer to the following documentation to learn about creating and managing AWS credentials in Cloudera:

Related Information

Introduction to the role-based provisioning credential for AWS

Create a provisioning credential for AWS

Obtain CLI commands for creating a credential

Viewing available credentials

Modify a credential

Change environment's credential

Delete a credential

Introduction to the role-based provisioning credential for AWS

Creating a credential is a prerequisite for creating an environment. On AWS, you have a single option for creating a cloud credential: a role-based credential.

When working with an AWS environment, you are required to configure a way for Cloudera to authenticate with your AWS account and obtain authorization to create resources on your behalf. On AWS, there is a single option for doing this: the role-based credential. Role-based authentication uses an IAM role with an attached IAM policy that has the minimum permissions required to use Cloudera. As stated in AWS docs:

"An IAM role is similar to an IAM user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session."

The following table provides more information on the role-based credential:

	Role-based
IAM entity used by the credential	A cross-account IAM role
Security mechanism	An IAM role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when an entity assumes a role, temporary security credentials for the role session are generated. Since Cloudera is set up with an AssumeRole policy, it can assume the IAM role. For more information about AssumeRole and granting permissions to create temporary security credentials, refer to AWS docs.
Use case	Suitable for an organization as it allows multiple users to use the same IAM role.
Overview of configuration steps	1. In the IAM console on AWS, create an IAM policy and a cross-account IAM role, and then assign the IAM policy to the IAM role. 2. Register the role ARN as a credential in Cloudera. Once done, Cloudera can assume the IAM role.

Related Information

IAM roles (AWS)

Granting permissions to create temporary security credentials (AWS)

Create a provisioning credential for AWS

Create a role-based credential referencing the IAM role created earlier. This can be done from the Cloudera web interface or CDP CLI.

Before you begin

Prior to performing these steps. you should create a cross-account role. See Cross-account access IAM role.

These steps assume that you have the ARN of the IAM role that you created in an earlier step. You can obtain the IAM Role ARN from the IAM console > Roles on AWS by selecting a role and clicking on your IAM role to navigate to its summary and then copying the Role ARN:

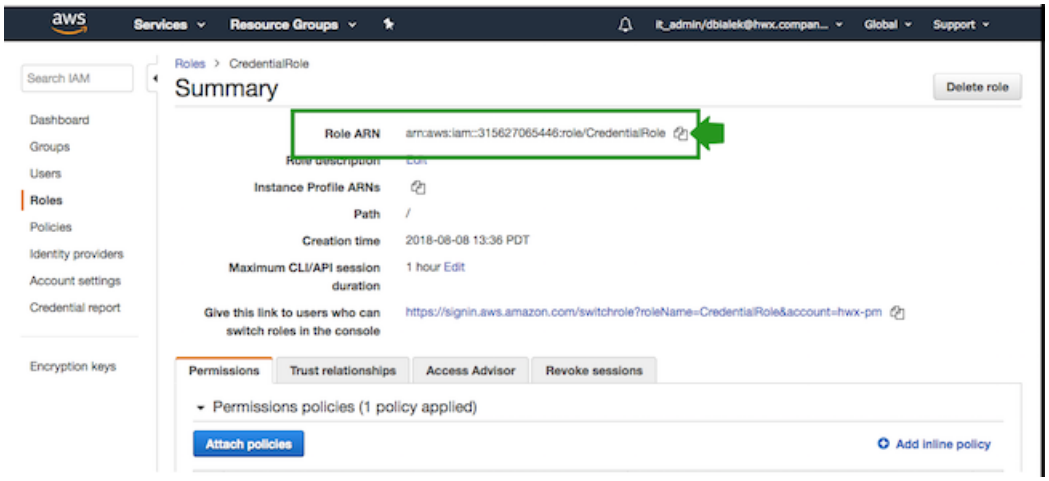

Required role: EnvironmentCreator

Steps

For CDP UI

1. Log in to the Cloudera web interface.

   Note:

   These steps show you how to create a credential from the Credentials page in the Cloudera web interface, but you may also create a credential directly from the Environments page as part of environment registration process.
2. Navigate to the Cloudera Management Console.
3. Select Shared Resources > Credentials from the navigation pane.
4. Click Create Credential.
5. Select AWS to access credential options for Amazon Web Services.
6. Provide the following information:

Parameter	Description
Select Credential Type	Select Role Based (default value).
Name	Enter a name for your credential.
Description	(Optional) Enter a description.
Enable Permission Verification	Activate the Enable Permission Verification button if you want Cloudera to check permissions for your credential. CDP will verify that you have the required permissions for your environment.

Parameter	Description
IAM Role ARN	Paste the IAM Role ARN corresponding to the “CredentialRole” that you created earlier. For example arn:aws:iam::315627065446:role/CredentialRole is a valid IAM Role ARN.

7. Click Create.
8. Your credential should now be displayed in the Credentials pane.

For CDP CLI

1. You have three options:
   a. (The simplest option) In CDP CLI, use the following command to create a credential:

```
cdp environments create-aws-credential \
--credential-name <value> --role-arn <value>
```

   b. Alternatively, you can provide the credential information in the Cloudera web interface > Cloudera Management Console > Environments > Shared Resources > Credentials > Create Credential and then click on SHOW CLI COMMAND and copy the JSON snippet. Next, save the JSON in a text file and use the following command to create a credential:

```
cdp environments create-aws-credential --cli-input-json <value>
```

   c. Alternatively, you can use the following commands: Use the first command to obtain the JSON snipped, then provide the missing information, and then use the second command to create the credential:

```
cdp environments create-aws-credential --generate-cli-skeleton
cdp environments create-aws-credential --cli-input-json <value>
```

After you finish

Now that you have created the credential, you can use it to register your AWS environment.

Obtain CLI commands for creating a credential

In addition to being able to obtain CDP CLI commands for credential creation from CDP CLI help, you can also obtain them from the Cloudera web interface.

You can quickly obtain CDP CLI commands for creating a credential from the Cloudera Management Console:

- (Required role: Owner) From details of an existing credential by clicking >_ in the top right corner:

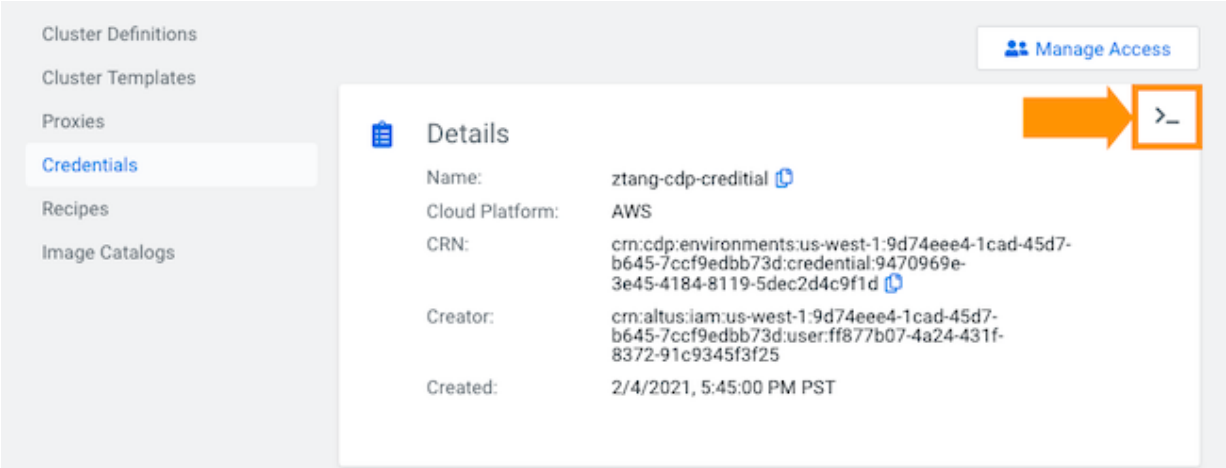

  If you obtain the command from this page, you should change the value of the --credential-name before running the command.
- (Required role: EnvironmentCreator) From the create credential wizard by providing all required parameters and then clicking >_ SHOW CLI COMMAND in the bottom of the page.
- (Required role: EnvironmentCreator) From the register environment wizard by providing all required parameters and then clicking >_ SHOW CLI COMMAND in the bottom of the credential section.

Viewing available credentials

You can access all existing credentials from Cloudera web interface by navigating to the Cloudera Management Console > Shared Resources > Credentials page. You can also list credentials from CDP CLI using the cdp envir onments list-credentials command.

Modify a credential

You can modify an existing credential if needed. You can modify all parameters except the credential name. This shows you how to modify a credential from the Cloudera web interface. There is currently no way to modify a credential from CDP CLI.

The modify credential option is meant for cases such as updating your access and secret key on AWS in case it expired or changed.

If the credential that you are trying to modify is already attached to an environment, ensure that the modification that you are making still allows access to the resources, such as the data lake, running within that environment.

Note:

If you are trying to update the IAM policy used for the credential, do not follow these instructions. Instead, edit the policy via the AWS Management Console or AWS CLI as described in Editing IAM Policies in AWS docs.

Required role: SharedResourceUser or Owner

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Shared Resources > Credentials from the navigation pane.
4. Select the credential that you would like to modify and click on or navigate to the credential's details and click Edit.
5. Provide the new credential information. You must provide all required parameters, even if they are same as previously.
6. Click on Save.

   This updates the credential across all the environments that are currently using it.

For CDP CLI

If you would like to modify a credential from the CDP CLI, use cdp environments modify-credential.

Change environment's credential

You can change the credential attached to an environment as long as the new credential provides the required level of access to the same AWS account as the old credential.

Required roles:

- EnvironmentAdmin or Owner of the environment
- SharedResourceUser or Owner of the credential

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Environments from the navigation pane.
4. Click on a specific environment.
5. Navigate to the Summary tab.
6. Scroll down to the Credential section.
7. Click to access credential edit options.
8. Select the new credential that you would like to use.
9. Click to save the changes.

For CDP CLI

If you would like to delete a credential from the CDP CLI, use:

```
cdp environments update-environment-credential --environment-name <value>
 --credential-name <value>
```

Delete a credential

You can delete a credential as long as no environment is currently using it.

Before you begin

A credential can only be deleted if it is not currently being used by any environments. If the credential that you are trying to delete is currently being used by one or more environments, you must remove the dependency by either deleting all such environments or changing their credential.

Required role: Owner

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Shared Resources > Credentials from the navigation pane.
4. Select the credential that you would like to delete and click on .
5. Confirm the deletion by clicking Yes.

   This deletes the credential.

For CDP CLI

If you would like to delete a credential from the CDP CLI, use

```
cdp environments delete-credential --credential-name <value>
```

.

Cloudera Management Console

Azure Credentials

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Working with Azure credentials 4

Azure credential overview 4

Create Azure credential 5

Obtain CLI commands 5

Modify a credential 6

Change environment's credential 7

Delete a credential 7

Working with provisioning credentials for Azure

Refer to the following documentation to learn about creating and managing Azure credentials in Cloudera:

Related Information

Introduction to the app-based provisioning credential on Azure

Create a provisioning credential for Azure

Obtain CLI commands for creating a credential

Modify a credential

Change environment's credential

Delete a credential

Introduction to the app-based provisioning credential on Azure

Creating a credential is a prerequisite for creating an environment. On Azure, you can create a credential using the app-based credential.

When working with an Azure environment, you can use the app-based credential to authenticate your Azure account and obtain authorization to create resources on your behalf. The app-based credential allows you to manually configure the service principal created within the Azure Active Directory.

The following table provides a list of app-based cloud credential options:

	App-based
Overview of steps	Manually create an application, assign a role to it, and then provide the application information as part of credential creation.
Required role	Refer to Azure permissions.
Requirements	You must be able to manage your organization's Azure Active Directory. If you do not have the required permissions, then you should ask your Azure administrator to register an application on Azure and then assign the Contributor role (or its equivalent) to it.
More information	The app-based method allows you to create a credential without logging in to the Azure account, as long as you have been given all the information. This means that if you can't complete app registration and role assignment steps by yourself, you can ask your Azure administrator to complete these steps and then provide you with the required information that can be provided during credential creation. In addition to providing your Subscription ID and Tenant ID, you must provide information for your previously created Azure AD application (its ID and key which allows access to it). Cloudera stores these credentials.

For more information about applications and service principal objects, refer to Azure docs.

Related Information

Application and service principal objects in Azure Active Directory (Azure)

Create a provisioning credential for Azure

Follow these steps to create an app-based provisioning credential for Azure. This can be done from the Cloudera web interface or CDP CLI.

Before you begin

Meet the prerequisites described in Prerequisites for the provisioning credential.

Required role: EnvironmentCreator

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. From the Cloudera home screen, click the Cloudera Management Console icon.
3. In the Management Console, select Shared Resources > Credentials from the navigation pane.
4. Click Create Credential.
5. Select Azure to access credential options for Microsoft Azure.
6. On the Configure credential page, provide the following parameters:

Parameter	Description
Name	Enter a name for your credential.
Description	(Optional) Enter a description.
Credential Type	Select App based Login.
Subscription Id	Copy and paste the Subscription ID from your Subscriptions.
Tenant Id	Copy and paste your Tenant ID from your Active Directory > Properties.
App Id	Copy and paste the Application ID from your Azure Active Directory > App Registrations > your app registration's Overview.
Password	This is your application key. You can generate it from your Azure Active Directory app registration's Certificates & secrets.

7. Click Create.

For CDP CLI

Use the cdp environments create-azure-credential command:

```
cdp environments create-azure-credential \
               --credential-name <CREDENTIAL_NAME> \
               --subscription-id <SUBSCRIPTION_ID> \
               --tenant-id <TENANT_ID> \
               --app-based applicationId=<APP_ID>,secretKey=<SECRET_KEY>
```

Now that you have created the credential, you can register it as part of an environment.

Obtain CLI commands for creating a credential

In addition to being able to obtain CDP CLI commands for credential creation from CDP CLI help, you can also obtain them from the Cloudera web interface.

You can quickly obtain CDP CLI commands for creating a credential from the Cloudera Management Console:

- (Required role: Owner) From details of an existing credential by clicking >_ in the top right corner:

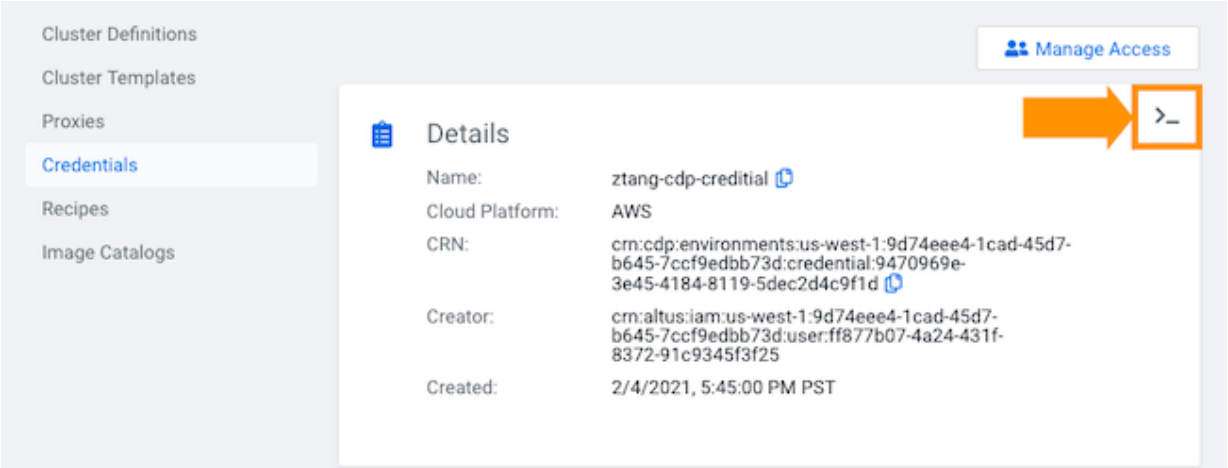

  If you obtain the command from this page, you should change the value of the --credential-name before running the command.
- (Required role: EnvironmentCreator) From the create credential wizard by providing all required parameters and then clicking >_ SHOW CLI COMMAND in the bottom of the page.
- (Required role: EnvironmentCreator) From the register environment wizard by providing all required parameters and then clicking >_ SHOW CLI COMMAND in the bottom of the credential section.

Modify a credential

You can modify an existing credential if needed. You can modify all parameters except the credential name. This shows you how to modify a credential from the Cloudera web interface. There is currently no way to modify a credential from CDP CLI.

The modify credential option is meant for cases such as updating your application's password in case it expired or changed.

If the credential that you are trying to modify is already attached to an environment, ensure that the modification that you are making still allows access to the resources, such as the data lake, running within that environment.

Required role: SharedResourceUser or Owner

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Shared Resources > Credentials from the navigation pane.
4. Select the credential that you would like to modify and click on or navigate to the credential's details and click Edit.
5. Provide the new credential information. You must provide all required parameters, even if they are same as previously.
6. Click on Save.

   This updates the credential across all the environments that are currently using it.

For CDP CLI

If you would like to modify a credential from the CDP CLI, use cdp environments update-credential:

```
cdp environments update-azure-credential
 --credential-name <value>
 --subscription-id <value>
 --tenant-id <value>
 --app-based <value>
```

Change environment's credential

You can change the credential attached to an environment as long as the new credential provides the required level of access to the same Azure subscription as the old credential.

Required roles:

- EnvironmentAdmin or Owner of the environment
- SharedResourceUser or Owner of the credential

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Environments from the navigation pane.
4. Click on a specific environment.
5. Navigate to the Summary tab.
6. Scroll down to the Credential section.
7. Click to access credential edit options.
8. Select the new credential that you would like to use.
9. Click to save the changes.

For CDP CLI

If you would like to delete a credential from the CDP CLI, use:

```
cdp environments update-environment-credential --environment-name <value>
 --credential-name <value>
```

Delete a credential

You can delete a credential as long as no environment is currently using it.

Before you begin

A credential can only be deleted if it is not currently being used by any environments. If the credential that you are trying to delete is currently being used by one or more environments, you must remove the dependency by either deleting all such environments or changing their credential.

Required role: Owner

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Shared Resources > Credentials from the navigation pane.
4. Select the credential that you would like to delete and click on .
5. Confirm the deletion by clicking Yes.

   This deletes the credential.

For CDP CLI

If you would like to delete a credential from the CDP CLI, use

```
cdp environments delete-credential --credential-name <value>
```

.

Cloudera Management Console

GCP Credentials

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Working with GCP credentials 4

GCP credential overview 4

Create GCP credential 4

Change environment's credential 5

Delete a credential 6

Working with provisioning credentials for GCP

Refer to the following documentation to learn about creating and managing GCP credentials in Cloudera:

Related Information

Introduction to the provisioning credential for Google Cloud

Register a GCP credential in Cloudera

Change environment's credential

Delete a credential

Introduction to the provisioning credential for Google Cloud

The provisioning credential for Google Cloud relies on a service account that can be assumed by Cloudera.

The following flow describes how the Google Cloud provisioning credential works:

1. Your GCP account administrator creates a service account and assigns the minimum permissions allowing Cloudera to create and manage resources in your Google Cloud account. Next, the administrator generates a service account access key pair for the service account. The minimum permissions are described in Service account for the Cloudera provisioning credential.
2. The service account is registered as a credential in Cloudera and its access key is uploaded to Cloudera.
3. The credential is then used for registering your Google Cloud environment in Cloudera.
4. Once this is done, Cloudera uses the credential for provisioning environment-related resources, workload clusters, and resources for other Cloudera services that you run in Cloudera.

Register a GCP credential in Cloudera

Once you have created a service account and generated a JSON access key in GCP, register the access key as a credential in GCP.

Before you begin

Prior to registering a GCP credential in Cloudera, you should create a service account in Google Cloud and assign the required permissions to it. Next, you should generate a JSON access key for the service account. See Service account for the provisioning credential. Once done, you can register the Google Cloud credential in Cloudera using the steps below.

Required role: EnvironmentCreator

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Shared Resources > Credentials from the navigation pane.
4. Click Create Credential.
5. Under Select Cloud Provider, select Google to access credential options for Google Cloud Platform.

6. Provide the following information:

Parameter	Description
Name	Enter a name for your credential.
Description	(Optional) Enter a description.
Key Type	JSON is pre-selected. Do not change this. P12 key type is only included for backward compatibility, not for new deployments.
Service Account Private Key (JSON format)	Upload the JSON access key generated for your pre-created service account. You should have saved it on your computer in an earlier step.

7. Click Create Credential.

For CDP CLI

Once you've created a service account and generated an access key, you can provide the access key to CDP by creating a credential.

Use the following CDP CLI command to create a credential:

```
cdp environments create-gcp-credential \
--credential-name YOUR-CREDENTIAL-NAME \
--credential-key file:///PATH/TO/FILE.json
```

The --credential-key parameter to point to the location of the JSON access key file on your computer.

For example:

```
cdp environments create-gcp-credential \
--credential-name test-gcp-cred \
--credential-key file:///Users/jsmith/Downloads/test-gcp-181d5f172bf1.json
```

Use the following command to verify that the credential has been created:

```
cdp environments list-credentials
```

After you finish

Now that you have created the credential, you can register it as part of a Cloudera environment.

Change environment's credential

You can change the credential attached to an environment as long as the new credential provides the required level of access to the same GCP account as the old credential.

Required roles:

- EnvironmentAdmin or Owner of the environment
- SharedResourceUser or Owner of the credential

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Environments from the navigation pane.
4. Click on a specific environment.

5. Navigate to the Summary tab.
6. Scroll down to the Credential section.
7. Click to access credential edit options.
8. Select the new credential that you would like to use.
9. Click to save the changes.

For CDP CLI

If you would like to delete a credential from the CDP CLI, use:

```
cdp environments update-environment-credential --environment-name <value>
 --credential-name <value>
```

Delete a credential

You can delete a credential as long as no environment is currently using it.

Before you begin

A credential can only be deleted if it is not currently being used by any environments. If the credential that you are trying to delete is currently being used by one or more environments, you must remove the dependency by either deleting all such environments or changing their credential.

Required role: Owner

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Shared Resources > Credentials from the navigation pane.
4. Select the credential that you would like to delete and click on .
5. Confirm the deletion by clicking Yes.

   This deletes the credential.

For CDP CLI

If you would like to delete a credential from the CDP CLI, use

```
cdp environments delete-credential --credential-name <value>
```

.

Cloudera Management Console

Credit Consumption and Usage Insights

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Credit consumption and usage insights.....4

Credit consumption and usage insights

Cloudera includes a user interface that allows you to monitor your credit consumption and download your consumption records.

Cloudera Public Cloud displays consumption data in terms of credits. A credit is a unit of consumption in Cloudera Public Cloud.

There is a 3-day delay for the consumption data displayed on the Consumption page. This means that consumption data from the past three days is not displayed.

Required roles: BillingAdmin

Steps

1. Navigate to the Cloudera web interface and select Cloudera Management Console.

2. Select Consumption from the navigation pane.

   On the **Consumption** page, you can view the **Credit Summary**, **Credits Burndown** and **Usage Insights**. By default, the consumption summaries are shown for all environments. You can further customize the visualization of consumption based on time, specific environments, and tags.

   Credit Summary

   Displays credit information based on your credit subscription.

   Credits and Usage Insights

   - The **Credits Burndown** chart shows the credit used over a period of time.

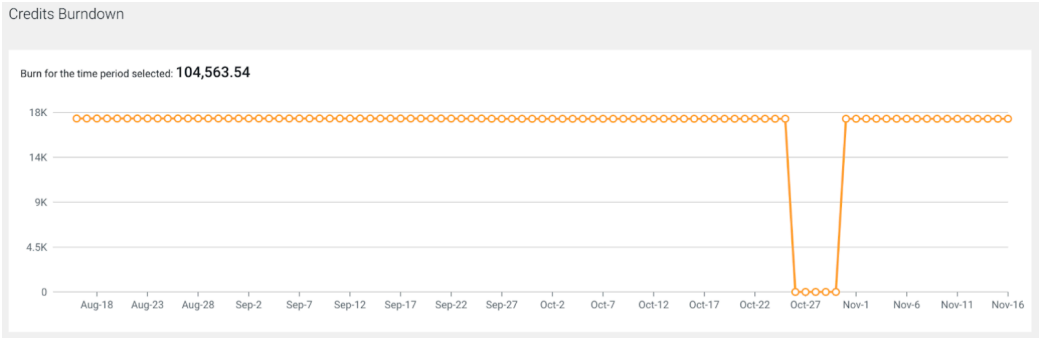

   - The **Usage Insights** bar chart shows the total consumption of the services over a period of time.

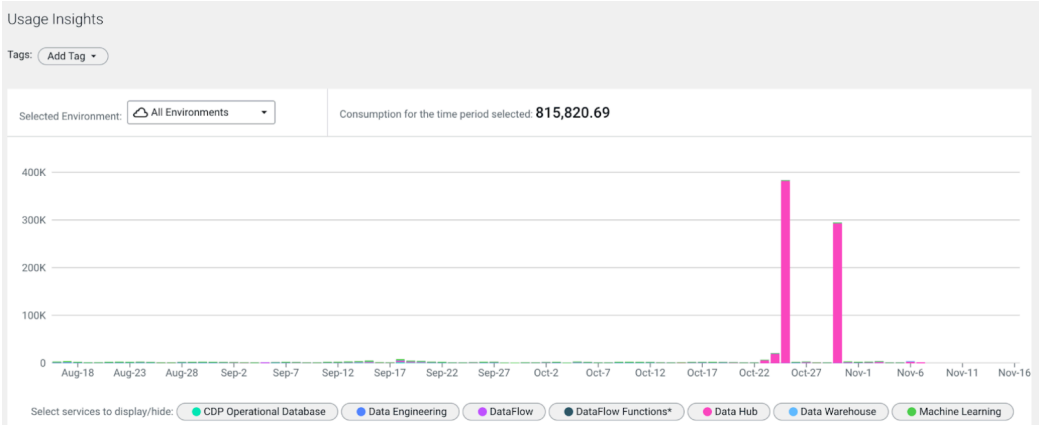

   - The **Clusters** table shows the number of the number of credits used by the clusters, number of hours the cluster was running, and how many clusters belong to the specific services.

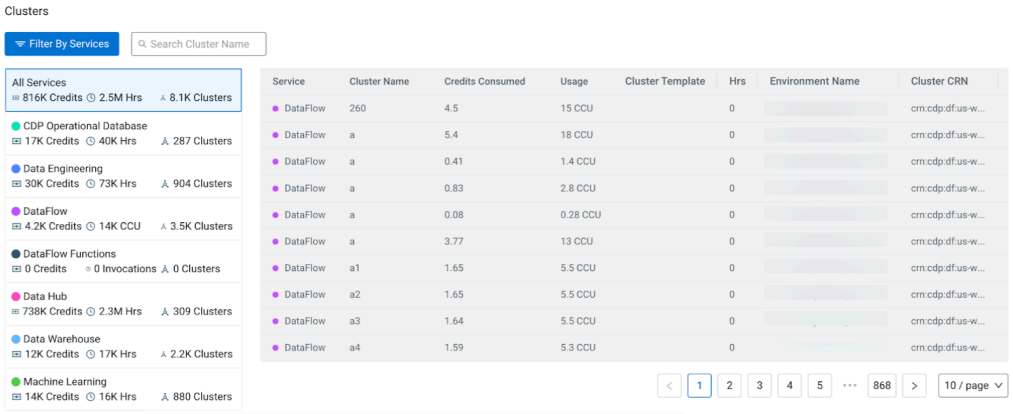

3. Select a date range to display the consumption for a time period. The selected time period can be one month, three months, six months and one year in a daily or monthly view.
4. Select tags to further filter the clusters in the environments for **Usage Insights**. The filter uses the user-level tags of the clusters. When selecting multiple tags, the clusters are displayed using OR logic. For example, if you select

department=engineering and department=finance, the clusters will be shown on the **Usage Insights** that have a department tag value of engineering or finance.

5. Select all or specific environments.

   You can view the **Usage Insights** of the clusters for all existing environments or select a specific one.

   a. Hover over a bar to display consumption information by services.

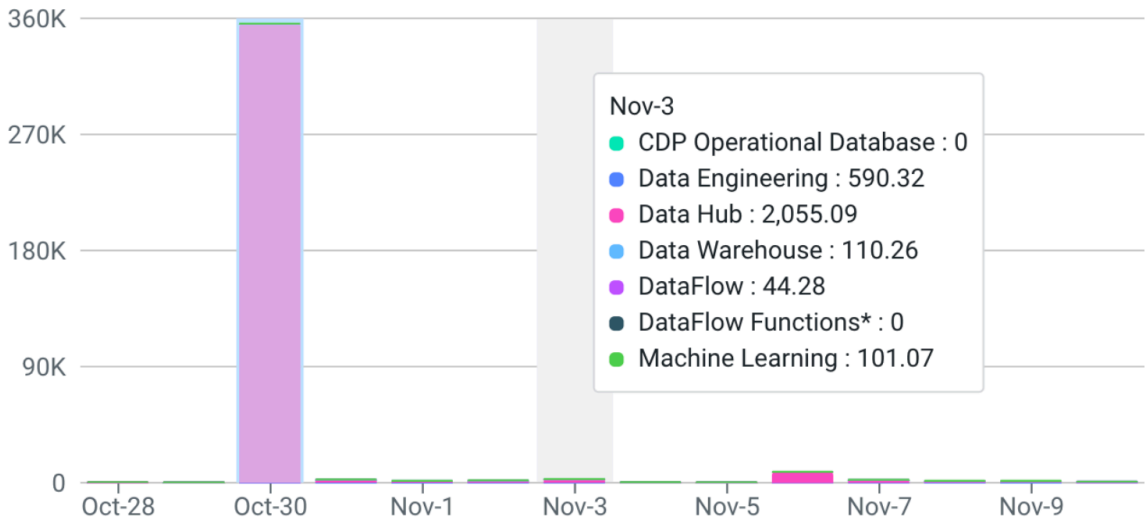

   Select or hide services that are displayed on the **Usage Insights** chart.

   Displaying the consumption information of the specific services associated with the environments can be enabled and disabled.

   b. Click on a bar to display consumption information by clusters.

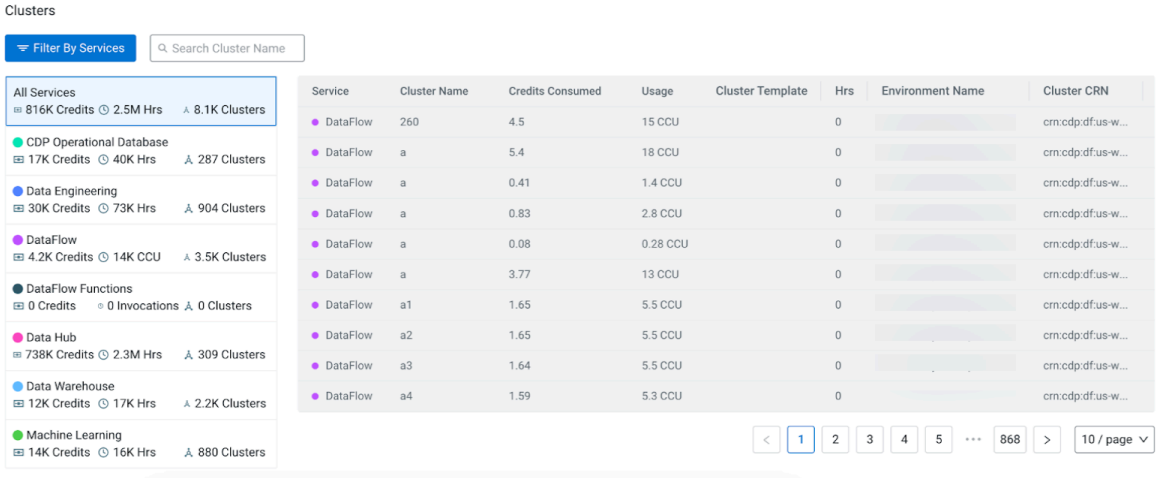

   You can view credit and usage information of clusters that belong to all services or a specific service.

   Clusters can also be searched by providing their name in the search field.

Download consumption summary from Cloudera web interface

Use the Download Usage Details option to download the consumption data:

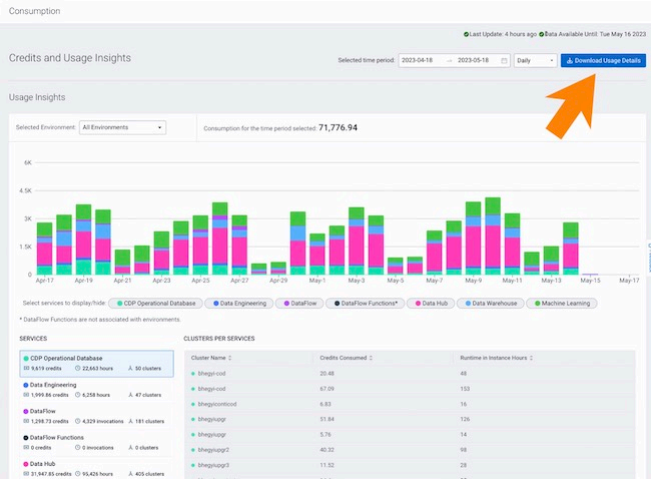

Download consumption summary from CDP CLI

Use the cdp consumption list-compute-usage-records command to download the consumption data. For example:

```
cdp consumption list-compute-usage-records \
--from-timestamp '2023-02-05T00:00:00Z' \
--to-timestamp '2023-02-07T00:00:00Z'
```

You can optionally set page size and page token.

To set page size (the number of records you want on one page), add the --page-size <value> option to the above command..

To set page token (which signifies which page you would like in the response), add the --starting-token <value> option to the above command. You will get the starting token for the next page in the response of the previous page.

CDP Public Cloud Management Console

Custom images and catalogs

Date published: 2022-01-31
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Custom images and image catalogs **4**

Creating a custom image and image catalog with the CDP CLI **5**

Switching image catalogs **7**

Custom images and image catalogs

If necessary, you can use a custom Runtime or FreeIPA image for compliance or security reasons. You can then use the CDP CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create a Data Lake or Cloudera Data Hub cluster or environment with a custom FreeIPA image.

Overview

A custom image should inherit most of its attributes from its source image, which is a default image that you select from the cdp-default image catalog.

The typical method of creating a Data Lake or Cloudera Data Hub cluster picks up the latest pre-warmed image from the cdp-default image catalog for the specified version of Runtime. These default images are pre-warmed VM images that contain a base URL to the default parcels in the Cloudera archive, amongst other configurations. If the default pre-warmed images do not suit your business needs, you can specify that the Data Lake or Cloudera Data Hub cluster or the environment (in the case of FreeIPA) uses a custom image instead.

What is a custom image?

A custom image is an entry in a custom image catalog that inherits most of its attributes from a source (default) image.

Custom image entries have:

- An image type: Cloudera Runtime [which includes Cloudera Data Hub and Data Lake images] or FreeIPA
- A source image ID that points to an image in the cdp-default image catalog
- A timestamp of creation
- An option to specify a VM region and image reference (such as an AMI ID) if you are overriding the source image with a custom VM image
- An option to override the parcel base URL

Why use a custom image?

You might require a custom image for compliance or security reasons (a “hardened” image), or to have your own packages pre-installed on the image, for example monitoring tools or software. You might also want to specify a custom image if you need to use a default image with a specific Cloudera Runtime maintenance version applied, rather than simply specifying the latest major Cloudera Runtime version.

Note: When customizing VM images, note that certain customizations (for example, CIS hardening rules) may not be compatible with Cloudera.

What can you customize?

In a custom image entry, you can override the VM images themselves with your own custom images that are sufficiently hardened. Importantly, you should only customize a default image from the cdp-default catalog as opposed to creating one from scratch. You can also override the default parcel base URL (at archive.cloudera.com) with your own host site.

What is a custom image catalog?

A custom image catalog is simply a catalog that holds custom images. A custom image catalog can contain a single or multiple custom image entries.

Custom image catalogs have:

- A name. The name is a unique identifier and is used to refer to the catalog during environment, Data Lake, and Cloudera Data Hub creation; as well as during catalog operations like creating an image.
- A description.
- An owner. The owner is the user who runs the command to create the catalog.

What is the process for creating a custom image and catalog?

- If you are replacing the VM images in a custom image entry with a customized version, you should first prepare the image by modifying an official Cloudera default image, which you can find under Shared Resources > Image Catalogs > cdp-default.
- Select a source image from the cdp-default image catalog to be the source of customization. When you run the CLI command to find a default image, you specify the Runtime version, provider, image type, or a combination of the three.
- Create a custom image catalog, or identify an existing catalog where you want to save the custom image entry.
- Apply the necessary changes to the custom image entry, like the override AMI IDs with the new, customized AMIs; or add a new parcel base URL using the `--base-parcel-url` command when you set the custom image.
- You can then create an environment, Data Lake, or Cloudera Data Hub, based on custom catalogs via the CDP CLI.

Creating a custom image and image catalog with the CDP CLI

You can create a custom Runtime or FreeIPA image and image catalog through the CDP CLI.

Before you begin

If you are replacing the VM images in a custom image entry with a customized version, you must first prepare the image by modifying an official Cloudera default image, which you can find under Shared ResourcesImage Catalogscdp-default. Take note of the image reference, such as the AMI ID.

Procedure

1. To find a source image from the cdp-default catalog that you want to use as the source of your custom image entry, run the following command:

   ```
   cdp imagecatalog find-default-image --provider <cloud-provider> --image-type <image-type> --runtime-version <Cloudera-Runtime-version>
   ```

   For example:

   ```
   cdp imagecatalog find-default-image --provider AWS --image-type RUNTIME --runtime-version 7.2.12
   ```

2. A custom image requires a custom catalog. If you haven't yet created a custom catalog, or if you want to create a new one for a new custom image, run the following command:

   ```
   cdp imagecatalog create-custom-catalog --catalog-name <unique-catalog-name> --description <catalog-description>
   ```

   For example:

   ```
   cdp imagecatalog create-custom-catalog --catalog-name my-custom-catalog --description test-catalog
   ```

3. Within the custom image catalog that you created (or an older custom catalog that you want to use), create a custom image entry with the selected source image marked as its source. Providing your own VM images is optional. Run the following command:

```
cdp imagecatalog set-<image-type>-image --catalog-name <name-of-the-cust
om-catalog> --vm-images region=<region-of-customized-image>,imageReferen
ce=<cloud-provider-specific-ID-of-a-customized-image> --source-image-id
<cdp-image-ID-of-source-image>
```

For example:

```
cdp imagecatalog set-runtime-image --catalog-name my-custom-catalog --vm-
images region=eu-central-1,imageReference=ami-7torotmhqi6q7438y --source-
image-id 8t4y9853-12b6-3n6z-75dh-tx775k4c793w
```

4. You can then apply the necessary changes to the custom image entry, like overriding AMI IDs with new, customized ones, or adding a new parcel base URL.

Command	**Description**
cdp imagecatalog find-default-image	Finds the default images in the cdp-default image catalog for a specified version of Runtime. Parameters: • --provider • --image-type • --runtime-version You can provide any combination of these parameters.
cdp imagecatalog create-custom-catalog	Creates a custom catalog. Parameters: • --catalog-name • --description –catalog-name is required.
cdp imagecatalog set-runtime-image or cdp imagecatalog set-freeipa-image	Creates a custom image entry (either Runtime [Data Hub/Data Lake] or FreeIPA) within the specified catalog. Parameters: • --catalog-name • --vm-images [region,imageReference] • --source-image-id • --image-id • --base-parcel-url --catalog-name and --source-image-id are required.

Switching image catalogs

You can switch the image catalog of an already existing Cloudera Data Hub, Data Lake, or FreeIPA cluster. You may want to switch the image catalog for a cluster in order to restrict which Runtime version can be upgraded to, or in order to move to custom images for an existing cluster.

Use the following CDP CLI commands to switch the image catalog for an existing cluster:

- FreeIPA:

```
cdp environments set-catalog --environment $ENVIRONMENT_NAME --catalog $
CATALOG
```

Parameter	Description
--environment	Name or CRN of the environment that holds the FreeIPA installation.
--catalog	URL of the FreeIPA catalog to be used.

- Data Hub:

```
cdp datahub set-catalog --cluster $DATAHUB_NAME --catalog-name $CATALOG
```

Parameter	Description
--cluster	Name or CRN of the Cloudera Data Hub for which you want to use the new image catalog.
--catalog-name	Name of the image catalog to be used. The image catalog must be a JSON based catalog, and switching is only possible from catalogs that are JSON based.

- Data Lake:

```
cdp datalake set-catalog --datalake $DATALAKE_NAME --catalog-name $CATALOG
```

Parameter	Description
--datalake	Name of the Data Lake for which you want to use the new image catalog.
--catalog-name	Name of the image catalog to be used. The image catalog must be a JSON based catalog, and switching is only possible from catalogs that are JSON based.

Cloudera Management Console

Data Lakes

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to Data Lakes 5
Data Lake storage 6
Data Lake scale 6
Data Lake repair 12

Creating an AWS environment with a medium duty data lake using the CLI 12

Understanding Data Lake details 13

Accessing Data Lake services 15

Accessing a Data Lake cluster via SSH 17

Administering a Data Lake 18

Monitoring a Data Lake 18

Vertically scaling instances and disks 22
Vertically scaling instance types 22
Vertically scaling disks 23
Modifying disks 24
Adding disks 26
Deleting disks 27

Retry a Data Lake 28

Upgrading Data Lake or Cloudera Data Hub database 28
Database upgrade known limitations and troubleshooting 37
Installing Postgres 14 packages manually 38
Installing Postgres 11 packages manually 39

Data Lake upgrade 40
Data Lake upgrade support matrix 43
Before you begin 43
Upgrading a Data Lake 45
Upgrading a Data Lake manually via CLI 47
Data Lake rolling upgrades 49

Data Lake rolling upgrade limitations and issues....50
Recovering from failed upgrades....51

Performing manual Data Lake repair....52

Backup and restore for the Data Lake.... 53
Cross-version support for Data Lake backup and restore....56
Configuring and running Data Lake backups....57
Checking the status of a Data Lake backup....61
Troubleshooting Data Lake backup operations....64
Configuring and running Data Lake restore....65
Showing Data Lake restore status....69
Restoring to a RAZ Data Lake....71
Upgrade Ranger and HMS schema after Data Lake restore....74
Troubleshooting Data Lake restore operations....75

Data Lake resizing....76
Checking that Atlas is up-to-date....80
Resizing the Data Lake through the Cloudera UI....81
Resizing the Data Lake through the CDP CLI....82
Resizing post-requisites....83
Recovering after a failed resizing operation....84
Refreshing Cloudera AI governance pods....84

Rotating database certificates....84
Rotating database certificates when SSL enforcement is enabled....86
Rotating database certificates when SSL enforcement is disabled....88

Managing public and private certificates....89
Renewing private/host certificates on Data Lake and Cloudera Data Hub clusters....90
Manually renewing public certificates for Data Lake and Cloudera Data Hub clusters....92

Recipes....93
Writing recipes....93
Recipe and cluster template parameters....95
Example: Recipe with parameters....99
Register a recipe....99
Update a recipe....100
Managing recipes from CLI....102

Introduction to Data Lakes

A Data Lake is a service which provides a protective ring around the data stored in a cloud object store, including authentication, authorization, and governance support.

A Data Lake provides a way for you to centrally apply and enforce authentication, authorization, and audit policies across multiple workload clusters—even as the workload clusters are created and terminated based on demand. When you register an environment in Cloudera, a Data Lake is automatically deployed for that environment. The Data Lake runs in the virtual network of the environment and provides security and governance layer for the environment's workload resources (such as Data Hub clusters). All workload resources are automatically "attached" to the Data Lake: the attached cluster workloads access data and run in the security context provided by the Data Lake.

While workloads are temporary, the security policies around your data schema are long-running and shared for all workloads. As your workloads come and go, the Data Lake instance lives on, providing consistent and available security policy definitions and auditing that are available for current and future workloads. All information related to schema (Hive Metastore), security policies (Ranger), audit (Ranger), and metadata management and governance (Atlas) is stored on external locations (external databases and cloud storage). These external locations leverage the security and availability features guaranteed by the cloud provider to ensure that even if one or all virtual hosts in a Data Lake fail, the storage remains and the Data Lake hosts can be replaced and reattached to the data storage with little or no downtime and no data loss.

A Data Lake cluster uses Apache Knox to provide a protected gateway for access to Data Lake component UIs. Knox is also installed on all workload clusters, providing a protected gateway for access to cluster component UIs.

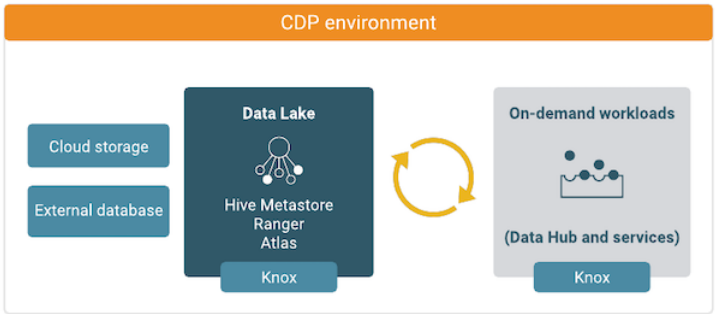

The following technologies provide capabilities for the Data Lake:

Component	Technology	Description
Schema	Apache Hive Metastore	Provides Hive schema (tables, views, and so on). If you have two or more workloads accessing the same Hive data, you need to share schema across these workloads.
Policy	Apache Ranger	Defines security policies around Hive schema. If you have two or more users accessing the same data, you need security policies to be consistently available and enforced.
Audit	Apache Ranger	Audits user access and captures data access activity for the workloads.
Governance	Apache Atlas	Provides metadata management and governance capabilities.
Gateway	Apache Knox	Supports a single workload endpoint that can be protected with SSL and enabled for authentication to access to resources.
Storage	Cloud provider storage, such as AWS S3 or Azure Storage	Isolates Data Lake storage from the compute resources. Data Lake storage is created when the Data Lake is created and is deleted when the Data Lake is terminated. Once created, the Data Lake storage lifecycle is separate from the Data Lake hosts' lifecycle: in case of a Data Lake host failure, the Data Lake storage remains and is reattached to new Data Lake host or hosts.

Related Information

Data Lake security
Apache Ranger authorization
Audit overview
Governance overview
Azure Load Balancers in Data Lakes and Cloudera Data Hub clusters

Data Lake storage

Data Lake storage leverages the security and high-availability guarantees from the cloud provider, allowing Data Lakes to regenerate hosts as needed, without data loss and with little or no downtime for workload services.

Data Lake storage is designed to reside in external storage locations separate from the hosts running the Data Lake services. This configuration protects Cloudera workloads from data loss should one or all of the Data Lake nodes fail. New hosts created in the Data Lake repair process are re-attached to the persistent data storage and Data Lake services return to normal.

The following table provides links to information from cloud providers about the service level guarantees they provide for each storage type.

AWS	Azure	GCP	Component Storage Content
Amazon RDS See Automated backups, snapshots, and automatic host replacement.	Azure Database for PostgreSQL See Availability guarantees.	Google Cloud SQL See Cloud SQL.	HMS SQL catalog Ranger policy data Cloudera Manager metadata
Amazon S3 See Availability and durability guarantees.	Azure Disk Storage See Resiliency and disaster recovery protections.	Google Cloud Storage See Google Cloud Storage.	Ranger audits Component logs HMS /warehouse directories
Amazon EBS See Availability and durability levels.	Azure Managed Disks See Availability and durability levels.	Google Persistent Disk See Persistent Disk.	Atlas search index (Solr) Kafka data (to support Atlas) Zookeeper metadata

Data Lake storage is created when a Data Lake is instantiated for an environment. When an environment is no longer needed and is terminated, the corresponding Data Lake is terminated and the external storage is cleaned up.

Data Lake storage persists through a Data Lake repair cycle; new hosts created in the repair are re-attached to the storage locations.

Note: Any S3 bucket that you designate for Data Lake cloud storage on AWS must be in the same region as the environment.

Data Lake scale

The scale of a Data Lake affects how many workload clusters can access your data using the security and governance services configured in the Data Lake, as well as resiliency of the Data Lake.

Cloudera supports both light duty Data Lakes and enterprise Data Lakes for AWS, Azure, and GCP. Medium duty and enterprise Data Lakes incur additional cost over light duty Data Lakes, but are required for production scenarios that require resiliency and scale. Medium duty and enterprise Data Lakes also have the ability to service a larger number of clients concurrently. See the sections below to understand the differences between light duty, medium duty and enterprise Data Lakes.

Important:

Enterprise Data Lakes are recommended for production workloads that require resiliency and scale.

Light duty Data Lakes do not provide resiliency. They are susceptible to data loss and down-time in case of a node failure. Cloudera recommends using this form factor for development and test use cases.

At this time, the following Data Lake scales are supported in Cloudera:

Feature	Light Duty Data Lake	Enterprise Data Lake	Medium Duty Data Lake (Discontinued as of Runtime 7.2.18)
High availability	Not available	✓	✓
Backups	✓	✓	✓
Availability Zones	Single availability zone	Multiple availability zones (AWS and Azure only)	Single availability zone
Security	Kerberos + LDAP/AD	Kerberos + LDAP/AD	Kerberos + LDAP/AD
Scale	About 5 concurrent workload clusters	About 20 concurrent workload clusters	About 20 concurrent workload clusters
Node count	1 master node running SDX Services 1 IDBroker node running networking authentication services	2 IDBroker nodes running authentication services 2 master nodes running core services 3 core nodes running HDFS, Kafka, Solr, and HBase 2 gateway nodes 1 auxiliary node	2 IDBroker nodes running authentication services 2 master nodes running core services in HA-enabled mode, with replication for resilience and scale 3 core nodes running HDFS, Kafka, Solr, and HBase 2 gateway nodes running services with API/UI access 1 auxiliary node for services that cannot run in HA mode

Feature	Light Duty Data Lake	Enterprise Data Lake	Medium Duty Data Lake (Discontinued as of Runtime 7.2.18)
Fault tolerance	Services unavailable during cluster node repair	Availability of services depends on the node being repaired. With the exception of the gateway and auxiliary nodes, the remaining groups can typically survive a single node failure without affecting workloads or UI/API access. In the event of a gateway node failure on a medium duty Data Lake, the load-balancer will seamlessly route to the other gateway node. As Cloudera Manager runs on only one gateway node (either 0 or 1), if the Cloudera Manager server gateway node fails, Cloudera Manager will not be available at all, but UI and API calls that bypass Cloudera Manager will be routed to the healthy gateway node by the load balancer. If the non-Cloudera Manager server gateway node goes down, Cloudera Manager will still be available, and the load balancer will seamlessly route to the healthy gateway node.	Availability of services depends on the node being repaired. With the exception of the gateway and auxiliary nodes, the remaining groups can typically survive a single node failure without affecting workloads or UI/API access. In the event of a gateway node failure on a medium duty Data Lake, the load-balancer will seamlessly route to the other gateway node. As Cloudera Manager runs on only one gateway node (either 0 or 1), if the Cloudera Manager server gateway node fails, Cloudera Manager will not be available at all, but UI and API calls that bypass Cloudera Manager will be routed to the healthy gateway node by the load balancer. If the non-Cloudera Manager server gateway node goes down, Cloudera Manager will still be available, and the load balancer will seamlessly route to the healthy gateway node.
Cloud-based load balancer	Not applicable, since there is only one instance of services running.	Network-based load balancer for front UI and API services.	Network-based load balancer for front UI and API services.
Additional comments		Enterprise Data Lake is available for environments using Runtime 7.2.17 and newer.	Medium duty Data Lake has been discontinued as of Runtime 7.2.18.

Light Duty Data Lakes

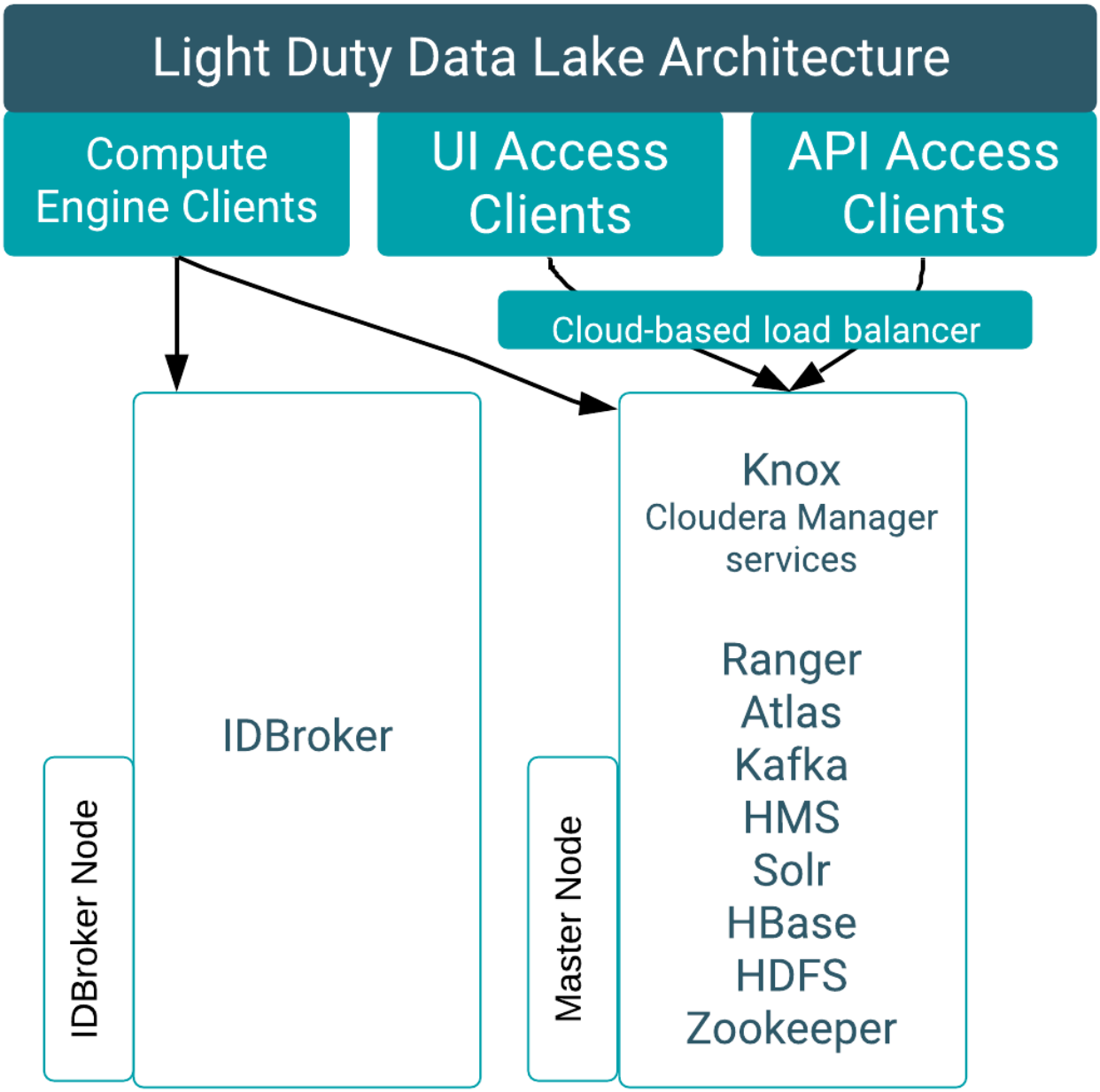

If the master node of a light duty Data Lake fails, compute engine clients such as Hive, Impala, and Spark, are partially resilient due to caching; but new queries cannot run without updated policy information, and audit information can also be affected. Because the Knox gateway also runs on the master node, clients with UI (such as the Ranger Admin UI and Atlas UI) or API access are unavailable in the event of a master node failure. In a light-duty Data Lake, the cloud-based load balancer exists for networking purposes and has no effect on the scale.

If the IDBroker node fails, compute-engine clients are affected because cloud access tokens cannot be verified. Clients with UI/API access remain available.

Enterprise Data Lakes

Enterprise Data Lakes, available for Cloudera Runtime 7.2.17 and newer, are a redefined version of medium duty Data Lakes that still offer failure resilience, but utilize resources and allocate memory more efficiently than a medium duty Data Lake at the same cost. Enterprise Data Lakes are configured such that services that do not need to scale are in the master hostgroup; services that need to scale vertically are in the gateway hostgroup; and services that can scale both horizontally and vertically are in the core hostgroup.

Enterprise Data Lakes can handle more intensive workloads than medium duty Data Lakes, support Ranger tag and user sync in HA mode, and when deployed in multi-AZ mode, remain operational during an availability zone outage.

Note: Multi-AZ deployments are currently only supported on AWS and Azure.

When compared to medium duty Data Lakes, RAZ will perform better and memory allocations for HBase, Solr, and Atlas have been increased. Other memory configurations have also been optimized.

Important: If you create an enterprise Data Lake through the CDP CLI, use the most recent version of the CLI. Older CLI versions do not support EDL creation.

Attention: The horizontal scale hostgroups noted in the diagram above contain a single service. Horizontal scaling is not yet supported, but enterprise Data Lakes are horizontal-scale ready for when the feature is available. When the feature is available, you will be able to use these hostgroups to horizontally scale these services as needed.

Medium Duty Data Lakes (Discontinued as of Cloudera Runtime 7.2.18)

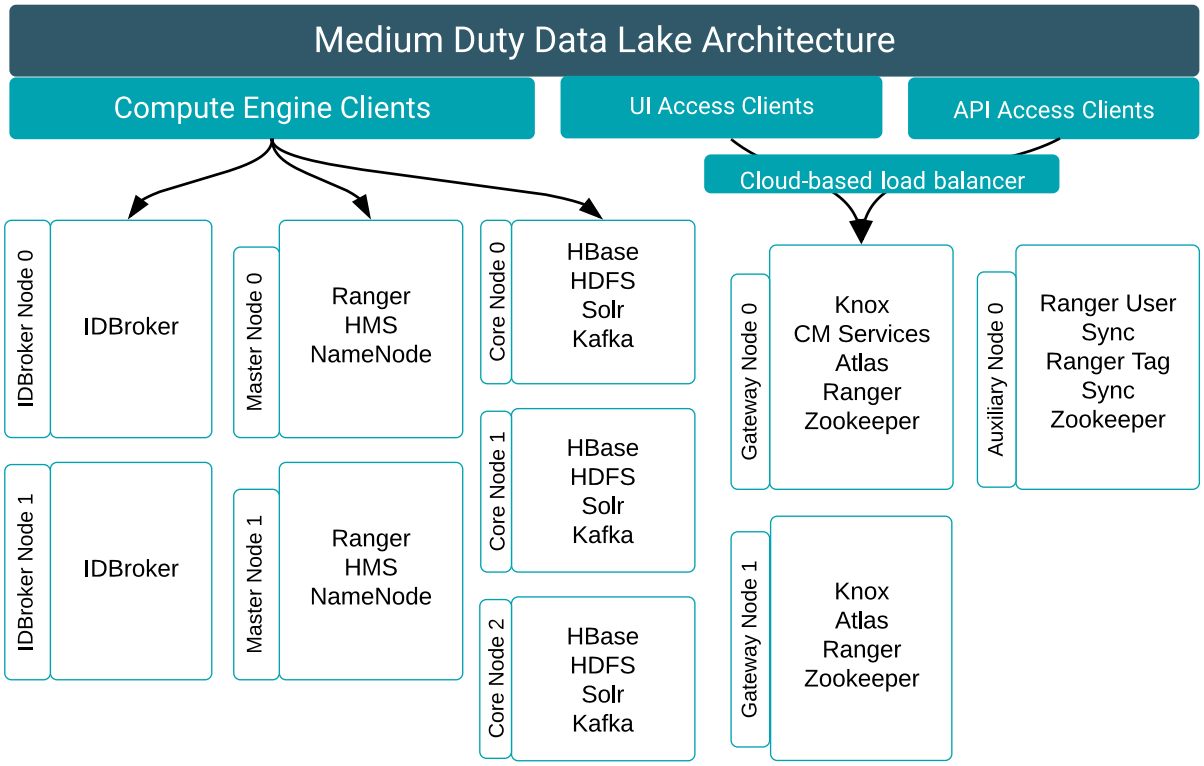

Medium duty Data Lakes provide failure resilience for compute engine clients such as Hive, Impala, and Spark; as well as failure resilience for clients with UI and API access, such as the Ranger Admin UI and the Atlas UI.

Note that while Cloudera Manager is shown as running on Gateway Node 0, it can be installed on either gateway node 0 or gateway node 1. You can see which node has Cloudera Manager installed by looking at the **Hardware** tab of the Data Lake for the gateway node marked "CM Server."

Failures in a medium duty Data Lake impact services as follows:

- Master node failure. Compute engine clients (for example, Hive, Impala, and Spark) are resilient to the failure, due to fallback high availability with smart client failover.
- IDBroker node failure. Both compute engine clients that use standard data connectors (Hive, Impala, Spark) and compute engine clients that use custom data connectors (for example, Hue) are resilient to the failure.
- Gateway node failure. Load-balanced UI and API access are available without interruption.
- Core node failure. Compute engine clients (for example, Hive, Impala, and Spark) are resilient to the failure, due to fallback high availability with smart client failover.
- Auxiliary node failure. Ranger user and tag sync are unavailable.

Important: Medium duty Data Lakes have been discontinued as of Cloudera Runtime 7.2.18. You can upgrade a medium duty Data Lake from 7.2.16 to 7.2.17, but will not be able to upgrade it further. You can create a new 7.2.17 medium duty Data Lake through the CDP CLI, but Cloudera recommends using the Enterprise Data Lake for new deployments.

If you want to scale an existing light or medium duty Data Lake to an enterprise Data Lake, you can perform Data Lake resizing.

Related Information

Azure Load Balancers in Data Lakes and Cloudera Data Hub clusters
Deploying Cloudera in multiple availability zones

Data Lake repair

If a Data Lake node fails, an administrator can trigger a manual repair process to restore the failed node and reconnect it to the persistent Data Lake storage.

For each Data Lake cluster, Cloudera detects the following failures indicate that one or more nodes needs repair:

- The node is unresponsive, from a crash or termination
- The Cloudera Manager agent process is unresponsive

Note: There may be failure modes that are not exposed at the level of the Cloudera Data Lake. If you encounter a service that is not responsive or is running but giving unexpected or incorrect results, no error will appear in the Data Lake details. Service-level errors should be managed through Cloudera Manager. For example, from Cloudera Manager, you can review the service's logs to determine the cause of the problem, make changes to the service configuration, and restart the service.

When Cloudera detects a node failure, a Cloudera administrator has the option to repair the failure manually. Note that during the repair process, the Data Lake services are not available to the attached workload clusters. Therefore, before triggering a Data Lake repair, consider stopping any jobs running on your workload clusters and restarting them after the Data Lake is restored. Audits and metadata will continue to be queued for collection through the restoration process.

When a node fails, you'll see a notification about node failure printed in the Event History tab for the Data Lake, the affected node is marked as unhealthy in the Hardware tab, and a button to start the repair process appears at the top of the Data Lake details. You can also select the Repair icon next to a host group on the Hardware tab to select specific nodes for repair. When your Cloudera administrator triggers node repair, the repair process:

1. Detaches all non-ephemeral disks from the failed nodes.
2. Removes the failed nodes.
3. Provisions new nodes of the same type, no upgrades are applied.
4. Reattaches the disks to the new volumes.
5. Reconnects services to the external database.

Related Information

Performing manual Data Lake repair
Data Lake storage
Cloudera Manager Health Tests
Cloudera Manager logs

Creating an AWS environment with a medium duty data lake using the CLI

You can use the CDP CLI to create an AWS environment with a medium duty data lake.

About this task

Required role: EnvironmentCreator

Before you begin

Before you use the CDP CLI, run the following command to verify that your environment is pointing to the correct profile:

```
cdp --profile {PROFILE}
```

As a sanity check, run the following command to verify that your environment name is not already taken:

```
environments describe-environment --environment-name {ENVNAME}
```

Procedure

1. Create a new environment:

   ```
   cdp environments create-aws-environment --cli-input-json file://{ENV_FIL
   E_PATH}
   ```

2. To set the IDBroker mappings, run the following command:

   ```
   cdp environments set-id-broker-mappings --environment-name "$ENVNAME" --
   data-access-role "$DATAACCESSROLE" --baseline-role "$BASELINEROLE" --set-
   empty-mappings
   ```

3. Run the following command to create the data lake cluster within the environment, where INSTANCEPROFILE is the instance profile for your specific account, and BUCKET is the path of a valid S3 location to store the data. This S3 path can be either the root of a bucket or a sub-folder:

   ```
   cdp datalake create-aws-datalake --datalake-name "NAME" --environment-name
    "ENVNAME" --cloud-provider-configuration instanceProfile="INSTANCEPROFILE
   ",storageBucketLocation="s3://MYBUCKET" --scale MEDIUM_DUTY_HA  --runtime
    7.2.7
   ```

4. Run the following command to check the status of the Data Lake:

   ```
   cdp datalake list-datalakes --environment-name ${ENVNAME}
   ```

   You should be able to look at the list of data lakes, locate yours by ENVNAME and check the status.

Understanding Data Lake details

To access information related to your Data Lake cluster and access cluster actions, navigate to Cloudera Management Console Data Lakes .

Each Data Lake cluster is represented by an entry on the Data Lakes page. To get more information about a specific Data Lake cluster, click on the tile representing your cluster.

Environment Details

This section includes information related to the Cloudera environment in which the Data Lake cluster is running:

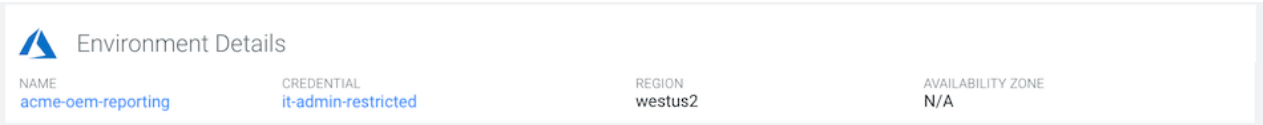

Item	Description
Cloud Provider	The logo of the cloud provider where the cluster is running.
Name	The name of the environment used to create the cluster.
Credential	The name of the credential used to create the cluster.
Region	The region in which the cluster is running in the cloud provider infrastructure.
Availability Zone	The availability zone within the region in which the cluster is running.

Services

Click logos in the Services section to open the user interface for the components that are running in the Data Lake cluster.

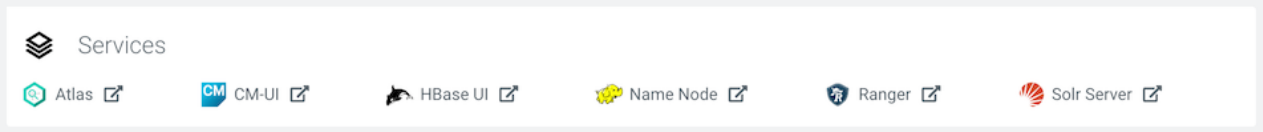

Cloudera Manager Info

The Cloudera Manager Info section provides the following information:

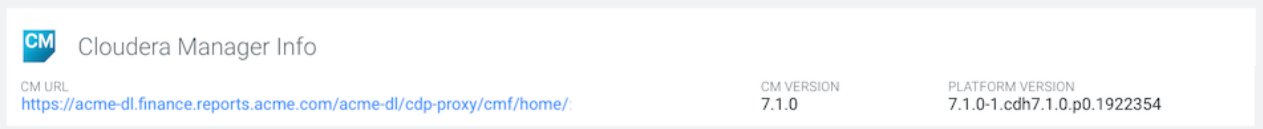

Item	Description
Cloudera Manager URL	Link to the Cloudera Manager web UI.
Cloudera Manager Version	The Cloudera Manager version which the cluster is currently running.
Platform Version	The Cloudera Runtime platform version which the cluster is currently running.

Event History and other tabs

The Data Lake page provides additional details organized in tabs, starting with the Event History tab:

Item	Description
Event History	Events logged for the cluster, with the most recent event at the top. The Download option allows you to download the event history to a local file. The events are formatted in JSON and compressed.

Item	Description
Hardware	Information about your cluster instances: instance names, instance IDs, instance types, their status, fully qualified domain names (FQDNs), and private and public IP addresses. Click » to access information about the instance, storage, image, and packages installed on the image.
Cloud Storage	External storage locations for database and files used by Data Lake services, such as HMS database, Ranger audit database, HBase files (storage for Atlas metadata).
Tags	User-defined tags, listed in the order they were added.
Endpoints	Endpoints for various cluster services, such as the URL for the Ranger user interface for defining data access policies.
Recipes	Future home of a list of custom scripts attached to this Data Lake. Each "recipe" lists its name, type, and the host group on which it was executed.
Attached clusters	The workload clusters using the services of this Data Lake; this information repeats the list of clusters found in the other tabs of this Cloudera Environment.
Repository Details	Cloudera Manager and Cloudera Runtime repository information, in more detail than shown in the Cloudera Manager Info section.
Image Details	Cluster node base image details.
Network	Names of the network and subnet in which the cluster is running and the links to the related cloud provider console
Telemetry	The instance profile and cloud storage location specified during environment setup under Log Storage and Audits for service logs.

Show CLI Command

You can click the Show CLI Command button to review the CLI command used to create the Data Lake, and copy it if you want to create a similar Data Lake through the CDP CLI. Ensure that any Data Lake that you create has a unique name. For more information on the CDP CLI commands to create a Data Lake, review the CLI documentation for Data Lakes.

Accessing Data Lake services

You can access your Data Lake security and governance services such as Atlas and Ranger from the Endpoints tab from Data Lake details in the Cloudera Management Console.

Required role: EnvironmentAdmin, Data Steward, or Owner of the environment

To access data lake UIs and endpoints navigate to the Cloudera Management Console Data Lakes and click on the tile representing your Data Lake. This brings you to the Data Lake cluster details page:

- The URLs to data lake service UIs are listed directly on this page, in the Services section. Click on the URL for the service that you would like to access and you will be logged in automatically with your Cloudera credentials. All the UIs are accessible via the Knox gateway: The URLs listed connect you to a chosen service via Knox and Knox securely passes your Cloudera credentials.
- To access API endpoints, navigate to the Endpoints tab. If you need to access the endpoints, refer to Accessing Non-SSO Interfaces Using IPA Credentials.

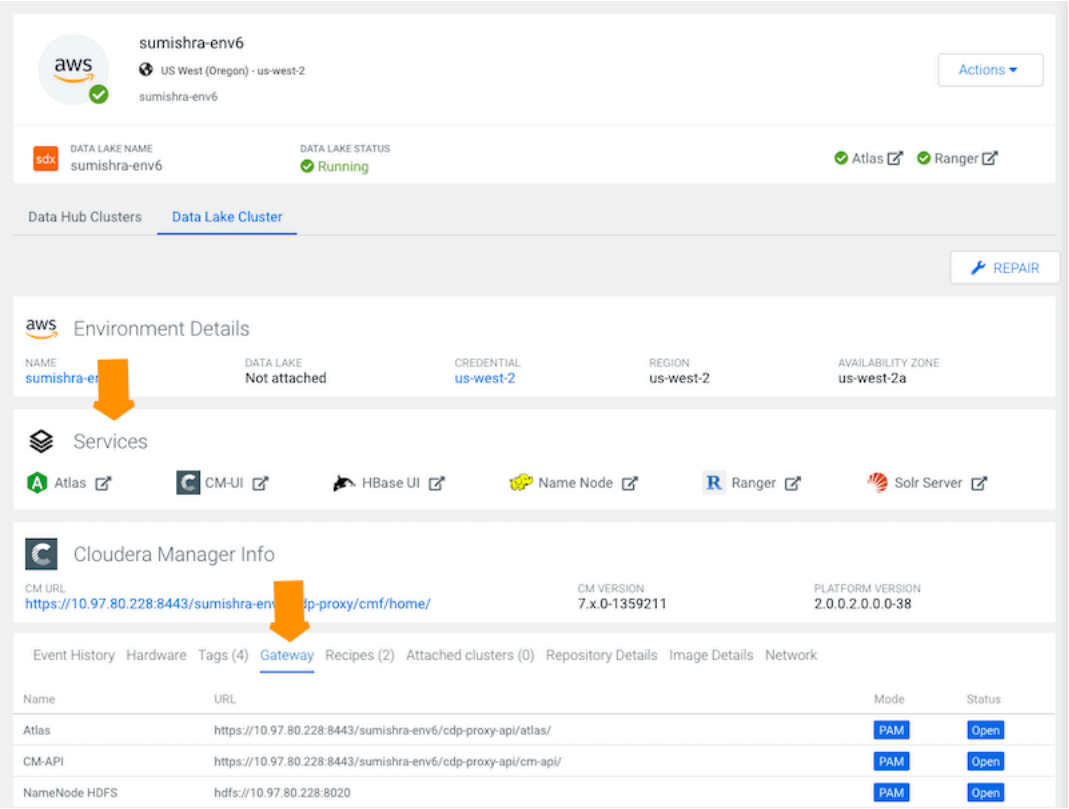

Security exception

The first time you access the UIs, your browser will attempt to confirm that the SSL Certificate is valid. Since CDP automatically generates a certificate with self-signed CA, your browser will warn you about an untrusted connection and ask you to confirm a security exception. Depending on your browser, perform the steps below to proceed:

Browser	Steps
Firefox	Click Advanced > Click Add Exception… > Click Confirm Security Exception
Safari	Click Continue
Chrome	Click Advanced > Click Proceed…

You can also view your available data lakes via CDP CLI using the following commands:

```
cdp datalake list-datalakes
cdp datalake describe-datalake
cdp datalake get-cluster-host-status
cdp datalake get-cluster-service-status
```

Related Information

Apache Ranger authorization

Audit overview

Governance overview

Accessing a Data Lake cluster via SSH

If you plan to access a Data Lake (for example for troubleshooting purposes) via a command line client, SSH into the master node.

Non-root SSH access to a Data Lake node

Required role: EnvironmentUser, Data Steward, or EnvironmentAdmin

A user who has the required Cloudera role assigned to them can SSH to Data Lake cluster nodes using their own Cloudera user.

To execute privileged operating system actions on a Cloudera Data Hub, the EnvironmentPrivilegedUser role is required.

For SSH access through a workload user name and password:

When accessing Cloudera for the first time, you must set a workload password. The password also needs to be reset each time you are added to a new environment.

For more information about workload passwords and instructions for setting/resetting it, refer to Set or Reset Workload Password.

For instructions on how to find your workload user name, refer to Finding Your Workload User Name.

On Mac OS, you can use the following syntax to SSH to the VM:

```
$ ssh <workloaduser>@<nodeIPaddress>
```

For example:

```
$ ssh jsmith@190.101.0.132
```

To SSH to a cluster using the private key file that pairs with the public key associated with a user, use the ssh utility:

```
$ ssh -i <path-to-private-key-file> <cdpusername>@<nodeIPaddress>
```

For example:

```
$ ssh -i ~/.ssh/my-private-key jsmith@192.12.141.12
```

On Windows, you can access your cluster via SSH by using an SSH client such as PuTTY. For more information, refer to How to use PuTTY on Windows.

Root SSH access to a Data Lake node

Required role: No Cloudera role is required

Cloudera administrators can access Data Lake cluster nodes as cloudbreak user with the SSH key provided during cluster creation.

On Mac OS, you can use the following syntax to SSH to the VM:

```
ssh -i <path-to-cloudbreak-private-key-file> cloudbreak@<nodeIPaddress>
```

For example:

```
ssh -i ~/.ssh/cloudbreak-private-key cloudbreak@90.101.0.132
```

On Windows, you can access your cluster via SSH by using an SSH client such as PuTTY. For more information, refer to How to use PuTTY on Windows.

Administering a Data Lake

To manage authorization and audit policies, and metadata use Apache Ranger and Apache Atlas.

Required role: Environment Admin, DataSteward, or the Owner of the environment

Close integration of Atlas with Apache Ranger enables you to define, administer, and manage security and compliance policies consistently across all components of the Hadoop stack.

Documentation	Description
Security: Apache Ranger authorization	How to set up fine grained access control for Cloudera Runtime services.
Security: Apache Ranger auditing	How to set up access auditing and reporting for Cloudera Runtime services.
Governance: Apache Atlas	Searching with metadata, working with classifications, exploring using lineage, and more.

For links to documentation, refer to:

Related Information

Security documentation

Governance documentation

Monitoring a Data Lake

You can monitor the status of your Data Lake from the Cloudera web UI or the CDP CLI.

Required role: EnvironmentAdmin, Data Steward, or Owner of the environment

Monitoring Data Lake cluster via UI

To access information related to your Data Lake cluster from the Cloudera web UI, navigate to Management Console Data Lakes . Each Data Lake cluster is represented by an entry on the Data Lakes page. To get more information about a specific Data Lake cluster, click on the tile representing your cluster. When a Data Lake cluster is healthy, its status should be Running.

To check health of specific hosts and services, navigate to Cloudera Manager.

Monitoring Data Lake cluster via CLI

You can view your available Data Lake clusters via CDP CLI using the following commands:

```
cdp datalake list-datalakes
cdp datalake describe-datalake
cdp datalake get-cluster-host-status
cdp datalake get-cluster-service-status
```

The cdp datalake list-datalakes command allows you to view a list of all available Data Lakes. For example:

```
cdp environments list-datalakes
{
    "datalakes": [
        {
            "datalakeName": "zookeeper-190920-144828-vg7",
```

```
            "crn": "crn:cdp:datalake:us-west-1:9d74eee4-1cad-45d7-b645-7cc
f9edbb73d:datalake:4529591f-53ea-4196-90fc-5d780d7063a8",
            "status": "RUNNING",
            "environmentCrn": "crn:cdp:environments:us-west-1:9d74eee4-1c
ad-45d7-b645-7ccf9edbb73d:environment:b1935d68-85d5-4f50-a023-56fa96d01c45",
            "creationDate": "2019-09-20T12:49:55.669000+00:00",
            "statusReason": "Datalake is running"
        },
        {
            "datalakeName": "zookeeper-sqqsx",
            "crn": "crn:cdp:datalake:us-west-1:9d74eee4-1cad-45d7-b645-7c
cf9edbb73d:datalake:92d66fed-c5d2-437c-a6eb-a54e40d36287",
            "status": "RUNNING",
            "environmentCrn": "crn:cdp:environments:us-west-1:9d74eee4-1cad-
45d7-b645-7ccf9edbb73d:environment:1eb291b3-dd23-4bdd-a3e8-09579afdf5a8",
            "creationDate": "2019-09-25T09:24:08.017000+00:00",
            "statusReason": "Datalake is running"
        }
    ]
}
```

The cdp datalake describe-datalake command allows you to obtain basic information about a specific Data Lake cluster. For example:

```
cdp datalake describe-datalake --datalake-name test-data-lake
{
    "datalake": {
        "crn": "crn:cdp:datalake:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9ed
bb73d:datalake:aa2e8e3e-2d6f-410b-bf3c-a3e02112bfc8",
        "datalakeName": "test-data-lake",
        "status": "RUNNING",
        "environmentCrn": "crn:cdp:environments:us-west-1:9d74eee4-1cad-45d
7-b645-7ccf9edbb73d:environment:574aa1cb-7a51-45a2-97ae-dead97072145",
        "credentialCrn": "crn:altus:environments:us-west-1:9d74eee4-1cad-
45d7-b645-7ccf9edbb73d:credential:83c861b6-5f62-4b83-a466-06de751a3964",
        "cloudPlatform": "AWS",
        "creationDate": "2019-09-20T22:09:22.422000+00:00",
        "clouderaManager": {
            "version": "7.x.0",
            "clouderaManagerRepositoryURL": "http://cloudera-build-us-west
-1.vpc.cloudera.com/s3/build/1445641/cm7/7.0.1/redhat7/yum/",
            "clouderaManagerServerURL": "https://adar-test-data-lake.adar-
tes.xcu2-8y8x.workload-dev.cloudera.com:8443/test-data-lake/cdp-proxy/cmf/ho
me/"
        },
        "productVersions": [
            {
                "name": "CDH",
                "version": "7.0.1-1.cdh7.0.1.p0.1443705"
            }
        ],
        "statusReason": "Datalake is running",
        "awsConfiguration": {
            "instanceProfile": "arn:aws:iam::069336058373:instance-profile/
idbroker-assume-role"
        }
    }
}
```

The cdp datalake get-cluster-host-status command allows you to obtain information about the health of each of your Data Lake hosts. For example:

```
cdp datalake get-cluster-host-status --cluster-name test-data-lake
{
    "hosts": [
        {
            "hostid": "5c8fb276620f0aa54bdd111e33ba5f58",
            "hostname": "idbroker1.cloudera.site",
            "healthSummary": "GOOD"
        },
        {
            "hostid": "30f27ab8472c9677985f04efc2b800c4",
            "hostname": "master0.cloudera.site",
            "healthSummary": "GOOD"
        }
    ]
}
```

The cdp datalake get-cluster-service-status command allows you to obtain information about the health of each service running on the Data Lake cluster. For example:

```
cdp datalake get-cluster-service-status --cluster-name test-data-lake
{
    "services": [
        {
            "type": "ZOOKEEPER",
            "state": "STARTED",
            "healthSummary": "GOOD",
            "healthChecks": [
                {
                    "name": "ZOOKEEPER_SERVERS_HEALTHY",
                    "summary": "GOOD"
                }
            ]
        },
        {
            "type": "HDFS",
            "state": "STARTED",
            "healthSummary": "GOOD",
            "healthChecks": [
                {
                    "name": "HDFS_DATA_NODES_HEALTHY",
                    "summary": "GOOD"
                },
                {
                    "name": "HDFS_VERIFY_EC_WITH_TOPOLOGY",
                    "summary": "DISABLED"
                }
            ]
        },
        {
            "type": "SOLR",
            "state": "STARTED",
            "healthSummary": "GOOD",
            "healthChecks": [
                {
                    "name": "SOLR_SOLR_SERVERS_HEALTHY",
                    "summary": "GOOD"
                }
            ]
        },
        {
```

```
            "type": "HIVE",
            "state": "STARTED",
            "healthSummary": "GOOD",
            "healthChecks": [
                {
                    "name": "HIVE_HIVEMETASTORES_HEALTHY",
                    "summary": "GOOD"
                }
            ]
        },
        {
            "type": "RANGER",
            "state": "STARTED",
            "healthSummary": "GOOD",
            "healthChecks": [
                {
                    "name": "RANGER_RANGER_ADMIN_HEALTHY",
                    "summary": "GOOD"
                },
                {
                    "name": "RANGER_RANGER_RANGER_TAGSYNC_HEALTH",
                    "summary": "GOOD"
                },
                {
                    "name": "RANGER_RANGER_RANGER_USERSYNC_HEALTH",
                    "summary": "GOOD"
                }
            ]
        },
        {
            "type": "HBASE",
            "state": "STARTED",
            "healthSummary": "GOOD",
            "healthChecks": [
                {
                    "name": "HBASE_REGION_SERVERS_HEALTHY",
                    "summary": "GOOD"
                }
            ]
        },
        {
            "type": "KAFKA",
            "state": "STARTED",
            "healthSummary": "GOOD",
            "healthChecks": [
                {
                    "name": "KAFKA_KAFKA_BROKER_HEALTHY",
                    "summary": "GOOD"
                }
            ]
        },
        {
            "type": "ATLAS",
            "state": "STARTED",
            "healthSummary": "GOOD",
            "healthChecks": [
                {
                    "name": "ATLAS_ATLAS_SERVER_HEALTHY",
                    "summary": "GOOD"
                }
            ]
        },
        {
            "type": "KNOX",
```

```
            "state": "STARTED",
            "healthSummary": "GOOD",
            "healthChecks": [
                {
                    "name": "KNOX_IDBROKER_HEALTHY",
                    "summary": "GOOD"
                },
                {
                    "name": "KNOX_KNOX_GATEWAY_HEALTHY",
                    "summary": "GOOD"
                }
            ]
        }
    ]
}
```


Vertically scaling instances and disks

If necessary, you can select a larger or smaller instance type for a Data Hub or Data Lake cluster after it has been deployed, or add, delete, or modify attached disks.

Vertically scaling instance types

If necessary, you can select a larger or smaller instance type for a Cloudera Data Hub or Data Lake cluster after it has been deployed in AWS and Azure.

Before you begin

You must stop the Data Lake or Cloudera Data Hub cluster before you vertically scale any of the instances.

About this task

Selecting a larger instance type adds more vCPU and/or RAM to your instances. Instances can be scaled both up and down, but scaling down to a smaller size requires 4 CPU and a minimum of 4 GB memory.

If you are using an instance without ephemeral disks, you can scale up or down to a new instance with ephemeral disks; however, the reverse is not supported. You cannot start with an instance with ephemeral disks and move to an instance without ephemeral disks.

Vertical scaling is supported on AWS and Azure.

Note: Vertical scaling to Azure v5 instances is not supported and results in the following error:

```
Unable to resize since changing from resource disk to non-resource d
isk VM size and vice-versa is not allowed.
```

For more information, see the Azure VM sizes with no local temporary disk documentation.

Data Lake and Cloudera Data Hub instances must be stopped before scaling. See Change the instance type in AWS documentation for more information.

For information on vertically scaling FreeIPA, see Vertically scale FreeIPA instances.

Procedure

1. In the Cloudera main navigation menu, click Data Hubs or Data Lakes and select the cluster that requires a larger instance type.
2. Scroll to the bottom of the page and click the **Nodes** tab.

3. Click the Vertical Scaling icon on the top right of the host group that you want to scale.

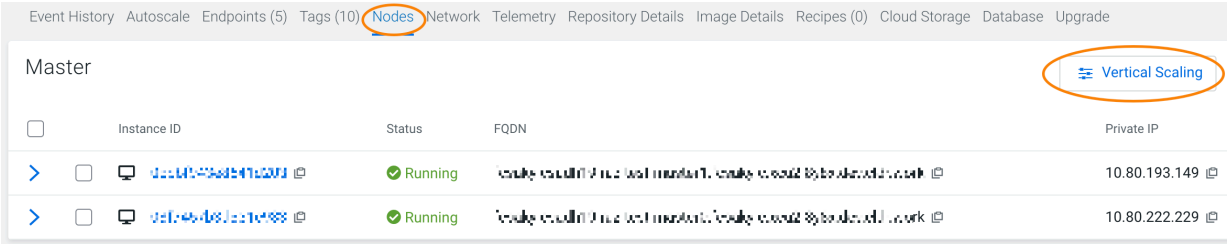

4. Select a larger instance type from the drop-down menu of suggested instance types.

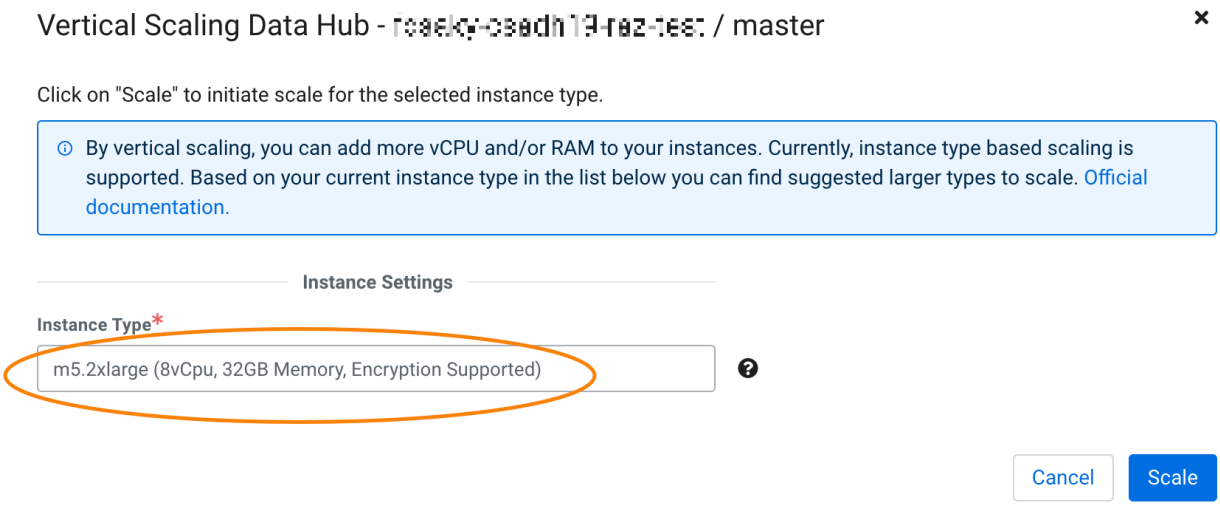

5. Click Scale. You can monitor the action from the **Event History** tab.

   Alternatively, you can use the CDP CLI to select a new instance for the Data Lake or Cloudera Data Hub cluster:

   Data Lake cluster:

```
cdp datalake start-datalake-vertical-scaling
--datalake <your-data-lake-name-or-its-crn>
--group <master>
--instance-template instanceType="<m5.4xlarge>"
```

   Data Hub cluster:

```
cdp datahub start-cluster-vertical-scaling
--datahub <your-data-hub-name-or-its-crn>
--group <master>
--instance-template instanceType="<m5.4xlarge>"
```

What to do next

After you have vertically scaled the cluster, configure the services on the cluster to use the additional or reduced resources/memory.

Vertically scaling disks

With the growing amount of data, it might be necessary to add, delete, or modify disks attached to Data Lake or Cloudera Data Hub clusters in AWS.

There are many clusters that are deployed with standard magnetic storage. With the growing lineage data, these disks are running out of space on core nodes. These need to be moved to General Purpose SSDs (gp2/gp3 on AWS) and or resized to a bigger disk.

The disks attached to the Data Lake and Cloudera Data Hub clusters can be changed or resized in AWS without downtime.

Limitations

When using this preview feature, be aware of the following limitations:

- This feature is only available for AWS.
- The disks can only be resized up, meaning you cannot reduce the size of an attached block storage. If there are multiple disks of different sizes, the size of all the disks attached to the instances in the group that are smaller or lesser than the requested size will be increased to the requested size.
- This feature will only resize additional block storages in an instance and not the root volume.
- Clusters and cluster services must be in running state before disk vertical scaling is performed.
- Current implementation does not support this feature through the Cloudera UI; It is available only through Beta CDP CLI. To install Beta CDP CLI, refer to Installing Beta CDP CLI.
- The disk modification feature on AWS can only be used once in 6 hours. This is a limitation on the AWS side.

Permissions

This feature requires the following permissions to be added to the cross-account policy described in Cross-account access IAM role.

- ec2:ModifyVolume
- ec2:DescribeVolumesModifications
- ec2:DescribeVolumeStatus

The following table explains why Cloudera needs these permissions:

Permission	Description
ec2:ModifyVolume	It is required to modify the volume attributes such as type, size and IOPS capacity. Without this, volume modifications cannot be performed by Cloudera.
ec2:DescribeVolumesModifications	It is required to verify whether the volume modifications performed by Cloudera were successful. Only upon successful modification, other steps like resizing will be done.
ec2:DescribeVolumeStatus	This is required to make sure that the volume being modified is attached to an instance and not an orphaned volume.

Modifying disks

The disk volumes attached to the instances in a host group can be modified using CLI commands.

Modifying disks using CDP CLI

Use the following Beta CDP CLI command to modify the volumes attached to the instances in a host group. Replace the placeholders with actual values. For example *[***DATALAKE-NAME***]* should be replaced with an actual name. As part of this update, the instance-template parameter in the vertical scaling command has been made optional. But one of the instance-template or disk-options have to be provided.

Additional parameters for vertical scaling have been added to both the datalake and datahub commands. The modification request is sent as part of the --disk-options parameter.

The *[***VOLUME-TYPE***]* placeholder is for the type of volume the disks are being modified to. It is an optional field and should be added if the volume type has to be modified, in which case the type will not be modified.

The *[***SIZE***]* placeholder is optional as well and is for the size the disks are being increased to in GB. Specify only the integer value.

```
//DATA LAKE
cdp datalake start-datalake-vertical-scaling
          --datalake [***DATALAKE-NAME***]
          --group [***INSTANCE-GROUP-NAME***]
          --disk-options modifyDisks="{volumeType=\"[***VOLUME-TYPE***]\"
,size=[***SIZE***]}"



//DATA HUB

cdp datahub start-cluster-vertical-scaling
          --datahub [***DATAHUB-NAME***]
          --group [***INSTANCE-GROUP-NAME***]
          --disk-options modifyDisks="{volumeType=\"[***VOLUME-TYPE***]\",
size=[***SIZE***]}"
```

Verifying that modification is complete

The change to the disk can be verified through Cloudera UI, AWS console, or by logging into the instances directly, after the flow is completed in the Event History.

The following is a screenshot from Event History showing the completion of the disk update:

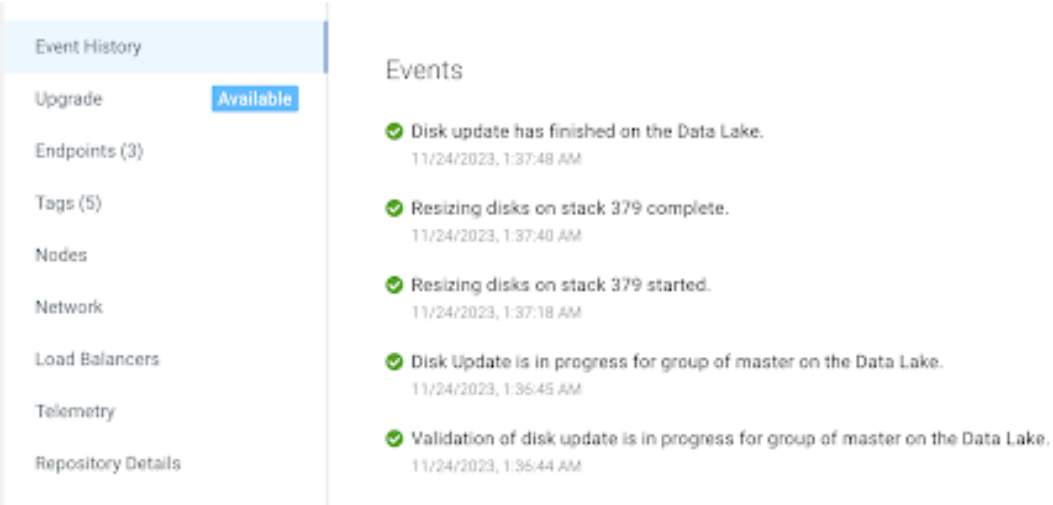

Verifying disk size/type in Cloudera UI

1. Navigate to the Cloudera Management Console Data Lake or Management Console Data Hub
2. Click into the Data Lake or Cloudera Data Hub that was modified.
3. Click Nodes in the left hand tree.
4. Open the accordion of the instance group that was modified.
5. Open the Storage Settings accordion in any of the instances in the group.

The Storage Type and Volume Size are updated based on the request.

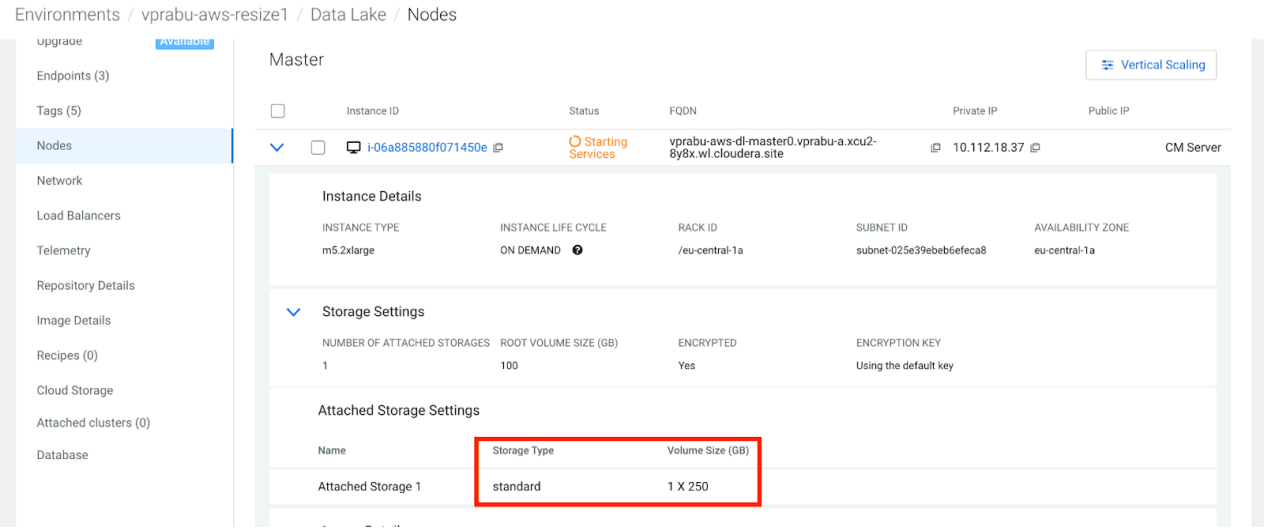

Verifying disk size/type in AWS Console

1. Navigate to the Cloudera Management Console Data Lake or Cloudera Management Console Data Hub
2. Click into the Data Lake or Cloudera Data Hub that was modified.
3. Click Nodes in the left hand tree.
4. Click on the AWS link for the instance and log into AWS Console.
5. Select the Storage tab for the instance.

The volume size is updated as per the request.

You can also click into an individual volume to make sure the type and size are modified correctly.

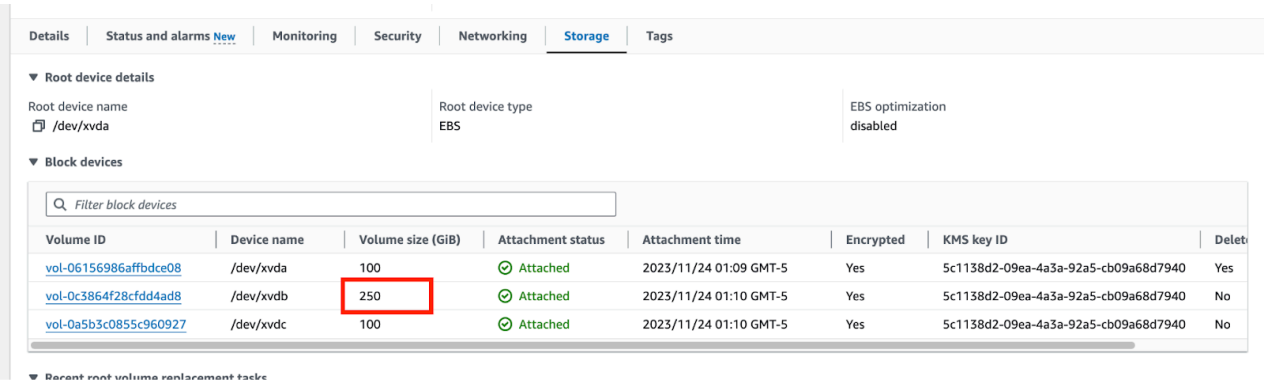

Adding disks

Additional volumes can be added to the instances in a host group using CLI commands.

In some cases when customers run out of memory, it is cheaper to add an additional volume instead of resizing to a bigger volume. This feature allows for adding additional block storages to an instance group with minimum interruption. Only the services running on the instance group are stopped, as the additional volume has to be mounted and configured for use on the instances.

Based on the request, ‘N’ number of disks of the same type and size can be added to the instance group.

Adding disks using CDP CLI

Use the following CDP CLI Beta command to add additional volumes attached to the instances in a host group. Replace the placeholders with actual values. For example, *[***DATALAKE-NAME***]* should be replaced with an

actual name. As part of this update, the instance-template parameter in the vertical scaling command has been made optional. But one of the -- instance-template or -- disk-options have to be provided.

The addDisks request is sent as part of the --disk-options parameter. All parameters for the add disk input are required.

The *[***NUMBER_OF_DISKS***]* placeholder is for the number of volumes being added.

The *[***VOLUME-TYPE***]* placeholder is for the type of volume being added.

The *[***SIZE***]* placeholder is for the size the disks being added in GB. Specify only the integer value.

The cloudVolumeUsageType is the purpose for which the disk is being added; it is an enum field with either “GEN ERAL” or “DATABASE” value.

```
//DATALAKE

cdp datalake start-datalake-vertical-scaling --datalake=[***DATALAKE_NAME***
] --group=[***DATALAKE_INSTANCE_GROUP_NAME***] --disk-options addDisks="{num
berOfDisks=[***NUMBER_OF_DISKS_TO_ADD***], volumeType=\"[***TYPE_OF_COLUME_T
O_ADD***]\",size=[***SIZE_OF_VOLUME***],cloudVolumeUsageType=\"GENERAL\"|\"D
ATABASE\"}"

//DATAHUB
cdp datahub start-cluster-vertical-scaling --datahub=[***DATAHUB_NAME***] --
group=[***DATAHUB_INSTANCE_GROUP_NAME***] --disk-options addDisks="{numberO
fDisks=[***NUMBER_OF_DISKS_TO_ADD***], volumeType=\"[***TYPE_OF_COLUME_TO_AD
D***]\",size=[***SIZE_OF_VOLUME***],cloudVolumeUsageType=\"GENERAL\"|\"DATAB
ASE\"}"
```

Deleting disks

To save costs, you can delete volumes attached to instances in an instance group using CLI commands.

In cases where only compute services are being run on an instance group, any block storage attached to the instance is going to cost the customer even if the cluster is stopped. As compute services do not store any persistent data on attached volumes, having additional volumes to store temporary data is unnecessary. In such cases, customers can use this command to delete all attached volumes on instances in an instance group.

This can be done only for compute instance groups on only Data Hubs, as Data Lakes need persistent volumes. This command deletes all additional volumes for instances in an instance group.

Deleting disks using CDP CLI

Use the following CDP CLI Beta command to delete all attached additional volumes to the instances in a host group. Replace the placeholders with actual values. For example, *[***DATALAKE-NAME***]* should be replaced with an actual name. As part of this update, the instance-template parameter in the vertical scaling command has been made optional. But one of the -- instance-template or -- disk-options have to be provided.

The delete disks request is sent as part of the --disk-options parameter. All parameters for the add disk input are required.

The deleteDisks parameter accepts a boolean value, either true or false.

```
//DATAHUB

cdp datahub start-cluster-vertical-scaling --datahub=[***DATAHUB_NAME***] --
group=[***DATAHUB_INSTANCE_GROUP_NAME***] --disk-options deleteDisks=true|
false
```

Retry a Data Lake

When stack provisioning or cluster creation fails, use the Retry option to resume the process from the last failed step.

About this task

Required role: EnvironmentAdmin or the Owner of the environment

Only failed stack or Data Lake creation can be resolved using a retry operation. You can potentially run a retry operation any number of times on a failed creation process, where each time it runs it resumes the creation process after the last successful step.

In some cases the cause of a failed stack provisioning or Data Lake creation may be eliminated by simply rerunning the process. For example, in case of a temporary network outage, a retry operation may be successful. In other cases, a manual modification is required before a retry operation can succeed. For example, if you are using a custom image but some configuration is missing causing the creation process to fail, you must log in to the provisioned node and fix the issue; after that you can run the retry operation to resume the Data Lake creation process.

Procedure

1. Log in to the Cloudera web interface.
2. Navigate to Cloudera Management Console Data Lakes .
3. Browse to the Data Lake details.
4. Click Retry.

   Note:

   Only failed stack or Data Lake creation are affected by a retry operation, so the option is only available in these cases.
5. Click Yes to confirm. The operation continues from the last failed step.

Upgrading Data Lake or Cloudera Data Hub database

This document describes the process to upgrade the database to the latest version supported by Cloudera Public Cloud services. You may use the Cloudera UI or CDP CLI to perform this upgrade.

About this task

Note: If you are upgrading an Azure Single Server database to Azure Flexible Server, you can read a specific process description in Upgrading Azure Single Server to Flexible Server.

Several Cloudera Public Cloud services, including the Data Lake cluster and the Cloudera Data Hub cluster templates and Data Services, require a relational database. Most of these databases are external and are provisioned during the initial deployment of the respective service.

The databases used by the Data Lake and some of the Cloudera Data Hub templates are hosted on external instances that are provisioned during the initial deployment of the respective service. For these external databases Cloudera Public Cloud leverages cloud-native service offerings of the three supported Cloud Service Providers (AWS RDS for PostgreSQL, Azure Database for PostgreSQL, and Cloud SQL for PostgreSQL).

Databases used by other Cloudera Data Hub templates are hosted on an embedded database instance, typically co-located on the Cloudera Manager host, in order to reduce the resource footprint.

Cloudera provides a database upgrade capability in Cloudera Public Cloud that allows moving both external and embedded databases to a higher major version.

The database upgrade is a fully automated operation. The upgrade process itself completes all of the required steps, including creating a backup, stopping and upgrading the database, restarting the database, and running post-upgrade maintenance tasks. You are not required to manually stop the Postgres instances before the upgrade.

Attention: In accordance with the PostgreSQL Versioning Policy, the cloud database services mentioned above may end support for PostgreSQL major version 11 on November 9, 2023 or shortly thereafter. Different cloud providers may have extended support for PostgreSQL 11. Despite this, Cloudera recommends upgrading to PostgreSQL 14 when the upgrade is available to you in Cloudera.

Important: In order to avoid disruption to the deployed Data Lake and Cloudera Data Hub services, caused by configuration changes to the underlying database service by the Cloud Service Providers, it is recommended that the database upgrade in Cloudera Public Cloud is performed before the End of Life date.

If you wish to disregard this recommendation, you may do so considering the risks involved as per the Cloud Service Provider policies.

The database upgrade is a separate operation, complementary to the existing maintenance, minor/major version and OS upgrades, as described in the Cloudera Public Cloud Upgrade Advisor.

This is a one-time operation. Once the database of a Data Lake or Cloudera Data Hub has been successfully upgraded to the newer major version, no further action is needed for the respective cluster.

Note: Cloudera recommends that the database upgrade is performed separately from other upgrade actions.

If a cluster uses a database that requires an upgrade, you will receive a notification, as shown below, on the Cloudera Management Console UI.

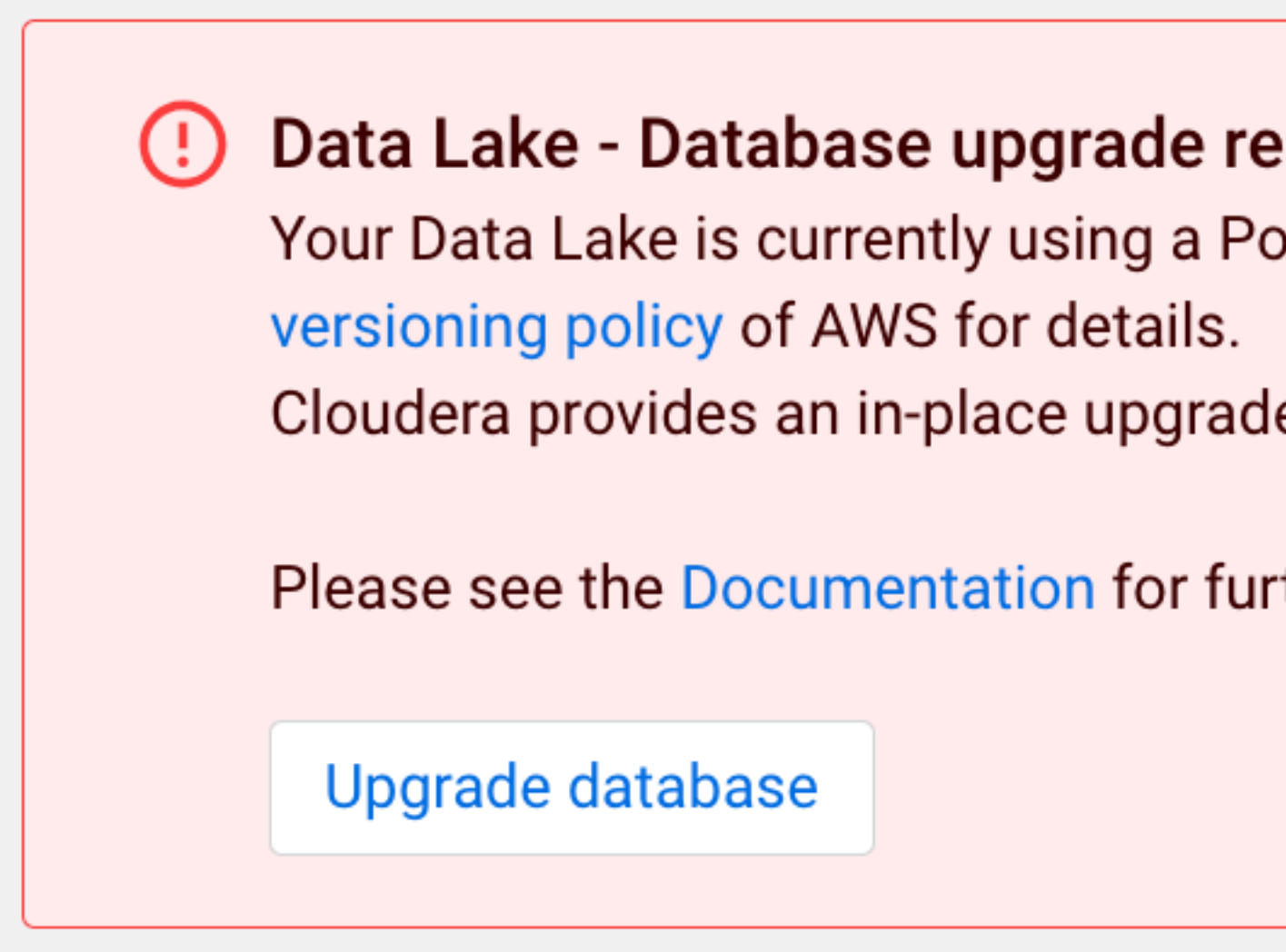

Running the database upgrade operation on the Cloudera Data Hub cluster will mean that all cluster services (Cloudera Manager and Cloudera Runtime services) are stopped on the cluster automatically without having to stop them manually. For the Data Lake database upgrade, it is recommended that attached Cloudera Data Hub clusters and Data services are in stopped state.

Note: Cloudera strongly recommends stopping all workloads in Data Services that interact with the Data Lake.

If you are concerned about stopping the workloads in your deployment, contact Cloudera support for a custom upgrade path.

For AWS and GCP environments, the Database Upgrade operation will trigger a backup and a major version upgrade for the attached external database. But for Azure environments, the mechanism is different; as in the background, it will create a new database instance with a higher major version and transfer the data from the older database instance.

Note: During Postgres database upgrade for Data Lakes and Cloudera Data Hub clusters on AWS and Azure, there is a possibility that manually changed configs of the database server will be reverted to the original configs. For more information, see Database upgrade known limitations.

Instructions

Here are the UI and CLI instructions to perform Database Upgrade on Data Lake and Cloudera Data Hub:

For CDP UI

Steps

1. In Cloudera Management Console, go to Environments. Select the cluster to perform the upgrade from the list of available clusters. The clusters are eligible for this upgrade are indicated in the right most column:

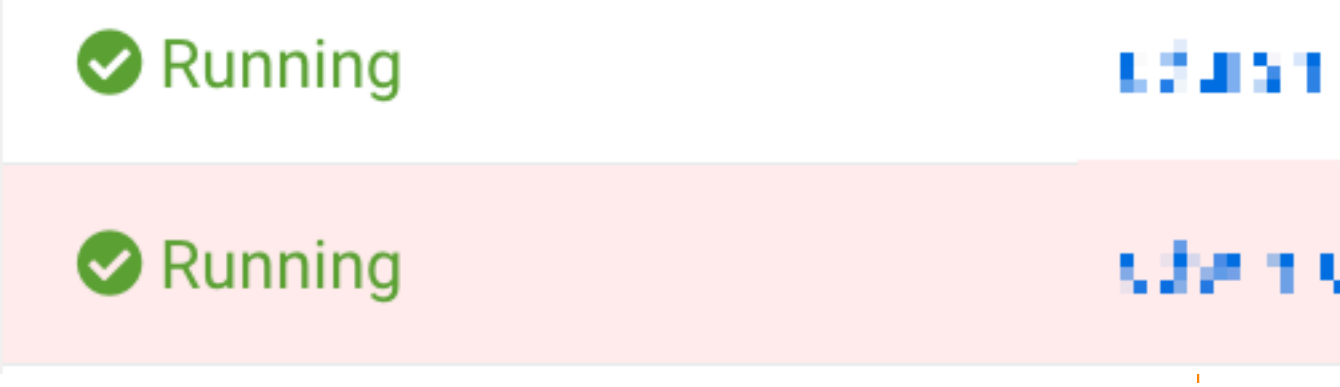

2. Once you select the cluster, you will see a message asking to update the Postgres version. Click the Upgrade database.

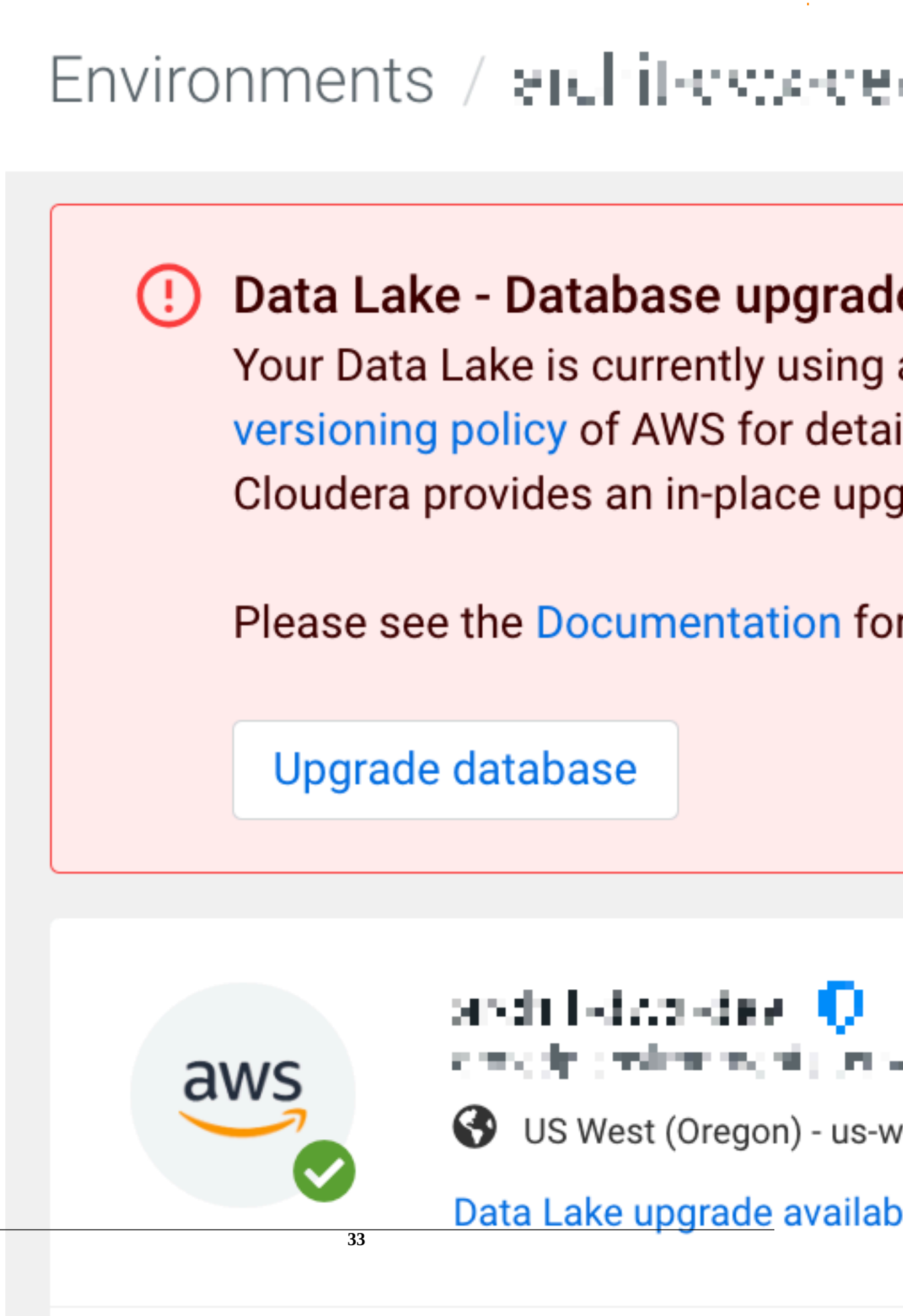
Environments /
Data Lake - Database upgrad
Your Data Lake is currently using
versioning policy of AWS for detai
Cloudera provides an in-place upg
Please see the Documentation fo
Upgrade database
aws
US West (Oregon) - us-w
Data Lake upgrade availab

3. Click Upgrade in the confirmation box.

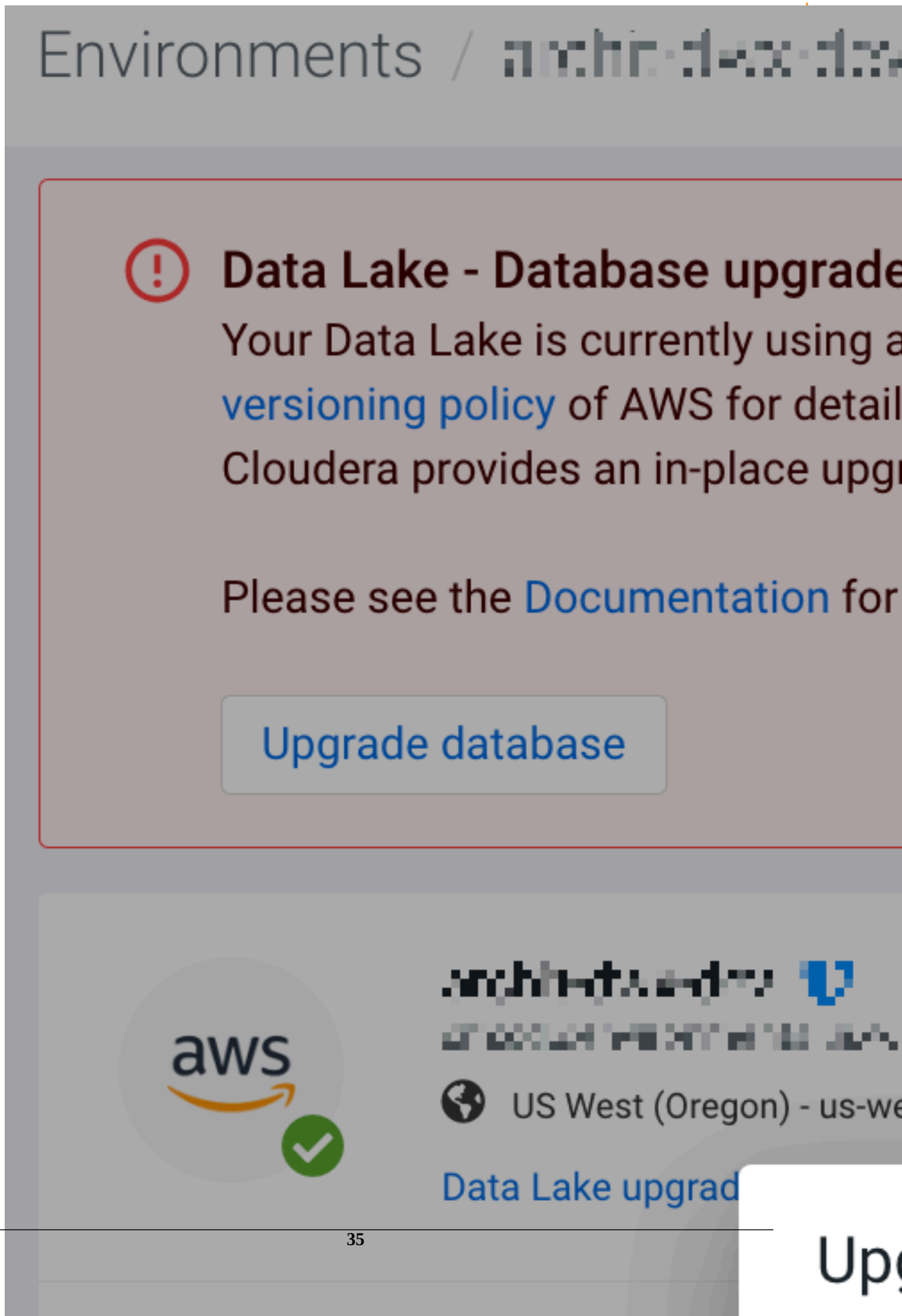
Environments /
Data Lake - Database upgrade
Your Data Lake is currently using a
versioning policy of AWS for detail
Cloudera provides an in-place upg
Please see the Documentation for
Upgrade database
aws
US West (Oregon) - us-we
Data Lake upgrad
Upg

4. Once the Data Lake database is updated, check for the Cloudera Data Hub clusters for that Data Lake, if there is any database upgrade notification and perform the database upgrade as described above.

   Note: The Database upgrade needs to be performed in every Data Lake and Cloudera Data Hub cluster separately, one by one.

For CDP CLI

Data Lake Database upgrade:

You can perform Data Lake database upgrade using cdp datalake start-database-upgrade CLI command.

The --target-version parameter is optional. If you do not provide it, the database will be upgraded to PostgreSQL 14.

```
cdp datalake start-database-upgrade --help --form-factor public
NAME
       start-database-upgrade  -  Upgrades the database of the Data Lake
 clus-
       ter.
DESCRIPTION
       This command initiates the upgrade of the database  of  the  Data
 Lake
       cluster.
SYNOPSIS
            start-database-upgrade
          --datalake <value>
          --target-version <value>
          [--cli-input-json <value>]
          [--generate-cli-skeleton]

OPTIONS
       --datalake (string)
          The name or CRN of the Data Lake.

       --target-version (string)
          The database engine major version to upgrade to.

          Possible values:

          o VERSION_14
```

Data Hub Database upgrade:

You can perform Cloudera Data Hub database upgrade using cdp datahub start-database-upgrade CLI command.

The --target-version parameter is optional. If you do not provide it, the database will be upgraded to PostgreSQL 14.

```
cdp datahub start-database-upgrade --help --form-factor public
NAME
       start-datahub-upgrade  -  Upgrades the database of the Data Hub c
lus-
       ter.

DESCRIPTION
       This command initiates the upgrade of the database  of  the  Data
Hub
       cluster.
SYNOPSIS
            start-database-upgrade
          --datahub <value>
```

```
          --target-version <value>
          [--cli-input-json <value>]
          [--generate-cli-skeleton]

OPTIONS
       --datahub (string)
          The name or CRN of the Data Hub.

       --target-version (string)
          The database engine major version to upgrade to.
          Possible values:
          o VERSION_14
```

The progress of the upgrade can be tracked on the respective service's Event History page. You can verify a successful database upgrade in the Event History or in the Database tab of the cluster. Once the upgrade is complete, Cloudera recommends verifying your workloads before attempting an additional Runtime or OS upgrade.

Note: As part of the database upgrade operation, PostgreSQL 14 client binaries will be installed on the cluster hosts, replacing earlier client versions. This may impact third-party components or custom services running on the cluster hosts.

Database upgrade known limitations and troubleshooting

Below are the known limitations associated with the database upgrade of Data Lake and Cloudera Data Hub clusters and ways to troubleshoot them.

Note: For troubleshooting information related to upgrading to Azure Flexible Server, see Troubleshooting Flexible Server. When you are upgrading an Azure Single Server database to PostgreSQL version 14, the Azure Single Server is automatically updated to Azure Flexible Server.

Known limitations and troubleshooting:

- Performing the Database Upgrade on Cloudera Runtime versions 7.2.6 or below

  Cloudera has verified PostgreSQL version 11 compatibility for Cloudera Runtime version 7.2.7 and above. There is no known reason why older Cloudera Runtime versions should not be compatible with PostgreSQL version 10.

  Workaround: You can request an entitlement that allows the Database Upgrade to be performed on older Cloudera Runtime versions on an exceptional basis.
- Performing the Database Upgrade on Data Lakes with attached Cloudera Data Hub clusters that cannot be stopped

  Technically, the Database Upgrade can be performed on a Data Lake without stopping the attached Cloudera Data Hub clusters. However, please be aware that during the upgrade, the Hive Metastore database will likely become temporarily unavailable and this can cause serious disruption or in the worst case can result in an inconsistent state for workloads running in Cloudera Data Hub clusters or Data Services.

  Workaround: If you acknowledge the risk and confirm that all cluster services and third party components relying on the Hive Metastore will be stopped for the time of the Database Upgrade, Cloudera can grant an entitlement that allows performing the upgrade with a running Cloudera Data Hub cluster on an exceptional basis.
- PostgreSQL client binaries will be upgraded to version 11 on all clusters hosts

  As part of the upgrade process we will try to install the PostgreSQL 11 libraries, pulling them from archive.cloudera.com. If the installation of these libraries does not succeed, a notification message will be sent that installation was attempted, but failed for some reason (network connectivity issues, etc).

  Workaround: Follow the process to install the libraries manually, see Installing PostergeSQL 11 packages manually.

  Note: Failing to install the PostgreSQL 11 client libraries as part of the Database Upgrade process will cause the Data Lake backup and restore operations to stop working correctly.

- Upgrading embedded databases

  Data Hub clusters using an embedded database will not require the Database Upgrade operation to be performed. The embedded database, including client libraries will be automatically upgraded during an OS upgrade.

  Note: This capability is currently disabled and will be activated later.

  Workaround: If you need to upgrade the embedded databases of your Cloudera Data Hub clusters, contact Cloudera to enable this capability on an exceptional basis. Once this entitlement has been granted, your embedded databases can be upgraded by performing an OS upgrade.
- Exceeding the End of Life deadline

  Data Lake and Data Hub clusters that are not upgraded until November 10, 2022 will continue to run on a PostgreSQL version 10 instance of the underlying AWS, Azure or GCP database service. As this instance will be considered End-of-Life (EoL) by the respective Cloud Service provider, they may reserve the right to schedule an automated major version upgrade, resulting in a temporary downtime. In the case of extreme events the Cloud Service Provider may also stop the instance, see Versioning policy- Azure Database for PostgreSQL. In either case, your Cloudera Public Cloud workloads may be seriously impacted.

  Workaround: Cloudera recommends performing the Database Upgrade via the CDP UI, or CLI as soon as possible.
- Possibility of custom config reset after Database upgrade on AWS and Azure

  During Postgres database upgrade for Data Lakes and Cloudera Data Hub cluters there is a possibility that manually changed configs of the database server, that the control plane does not know about, will be reverted to the original configs.

  Reason: On Azure the custom config can possibly reset during the database upgrade because Cloudbreak deletes and recreates the database server with the configs that the control plane knows about, so custom configs will be reverted.

  On AWS if SSL enforcement is enabled then the database server uses a custom parameter group with the SSL enforcement settings (created by control plane) and if the customer made any custom changes to this custom parameter group then those changed will be reverted, because the database upgrade requires the recreation of the custom parameter group.

Installing Postgres 14 packages manually

Steps for manual installation of PostgreSQL 14 packages.

About this task

The last step of the Database upgrade flow is the installation of PostgreSQL 14 packages on the cluster hosts. This is relevant in the case of an operating system image that does not yet contain the PostgreSQL 14 packages.

The required repositories are being hosted in the same location that is used for Cloudera Runtime upgrades: https://archive.cloudera.com/p/postgresql/postgresql14/redhat7/

If for some reason the package installation fails, it is required for the customers to manually install the aforementioned packages because otherwise the pg_dump utility driving the backup functionality will stop working.

- Method 1 : Installation using Cloudera hosted

  This method works only if you have proper network access and paywall credentials to the archive.cloudera.com repository as the required metadata is already pushed onto the nodes during the RDS upgrade process.

  SSH into the master node and run the following with superuser privileges:

```
{code}
source activate_salt_env
salt '*' state.apply postgresql/pg14-install
```

```
{code}
```

- Method 2: Manual installation

  Using this method you will install PostgreSQL packages using the official repo file

  1. SSH into the master node and run the following with superuser privileges (install PostgreSQL packages using the official repo file)

     ```
     yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL
     -7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
     ```

  2. Install required packages

     ```
     yum install -y postgresql14-server postgresql14 postgresql14-contrib pos
     tgresql14-docs
     ```

Installing Postgres 11 packages manually

Steps for manual installation of PostgreSQL 11 packages.

About this task

The last step of the Database upgrade flow is the installation of PostgreSQL 11 packages on the cluster hosts. This is relevant in the case of an operating system image that does not yet contain the PostgreSQL 11 packages.

The required repositories are being hosted in the same location that is used for Cloudera Runtime upgrades: https://archive.cloudera.com/p/postgresql/11/redhat7/

If for some reason the package installation fails, it is required for the customers to manually install the aforementioned packages because otherwise the pg_dump utility driving the backup functionality will stop working.

- Method 1 : Installation using Cloudera hosted

  This method works only if you have proper network access and paywall credentials to the archive.cloudera.com repository as the required metadata is already push onto the nodes during the RDS upgrade process.

  SSH into the master node and run the following with superuser privileges.

  ```
  source activate_salt_env
  salt '*' state.apply postgresql/pg11-install
  ```

- Method 2: Manual installation

  Using this method you will install PostgreSQL packages using the official repo file

  1. SSH into the master node and run the following with superuser privileges (install PostgreSQL packages using the official repo file)

     ```
     yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL
     -7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
     ```

  2. Install required packages

     ```
     yum install -y postgresql11-server postgresql11 postgresql11-contrib pos
     tgresql11-docs
     ```

Data Lake upgrade

When new versions or builds of Cloudera Runtime/Cloudera Manager are available for the Data Lake service, you can initiate a Data Lake upgrade. An OS upgrade may also be available. Use either the CDP CLI or the Management Console to initiate an upgrade.

There are several potential types of Data Lake upgrades:

- Cloudera Runtime and Cloudera Manager version upgrades, called "major/minor version" upgrades, which are available when a new Cloudera Runtime and Cloudera Manager version is released.
- Service pack upgrades, which do not change the major/minor Cloudera Runtime or Cloudera Manager release, but upgrade the Data Lake to the latest Cloudera Manager and/or Cloudera Runtime service packs available in the given version. These upgrades are made available as needed and can deliver targeted bug fixes for Cloudera Runtime, Cloudera Manager, or both.
- OS upgrades, which do not change any Cloudera Manager or Cloudera Runtime builds, but update the underlying cloud image. If an OS upgrade is available along with a major/minor version upgrade, the major/minor upgrade will incorporate the OS upgrade.
- Rolling upgrades allow you to upgrade the Cloudera Runtime and OS of Data Lakes and Cloudera Data Hub clusters, without stopping the cluster and its services. This means that you can upgrade a Data Lake without stopping the attached Cloudera Data Hub cluster and Data Services, or upgrade a Cloudera Data Hub cluster without workload downtime. Rolling upgrades for the Data Lake are limited to certain Data Lake Cloudera Runtime versions and shapes. For more information, see Data Lake rolling upgrades.

The type of upgrade that is available for a Data Lake is explicit on the Upgrade tab of the Data Lake details page. Any available upgrades are visible in the Target Runtime Version drop-down menu. For example:

A major/minor version upgrade, where the major/minor versions of CM and Runtime are upgraded together:

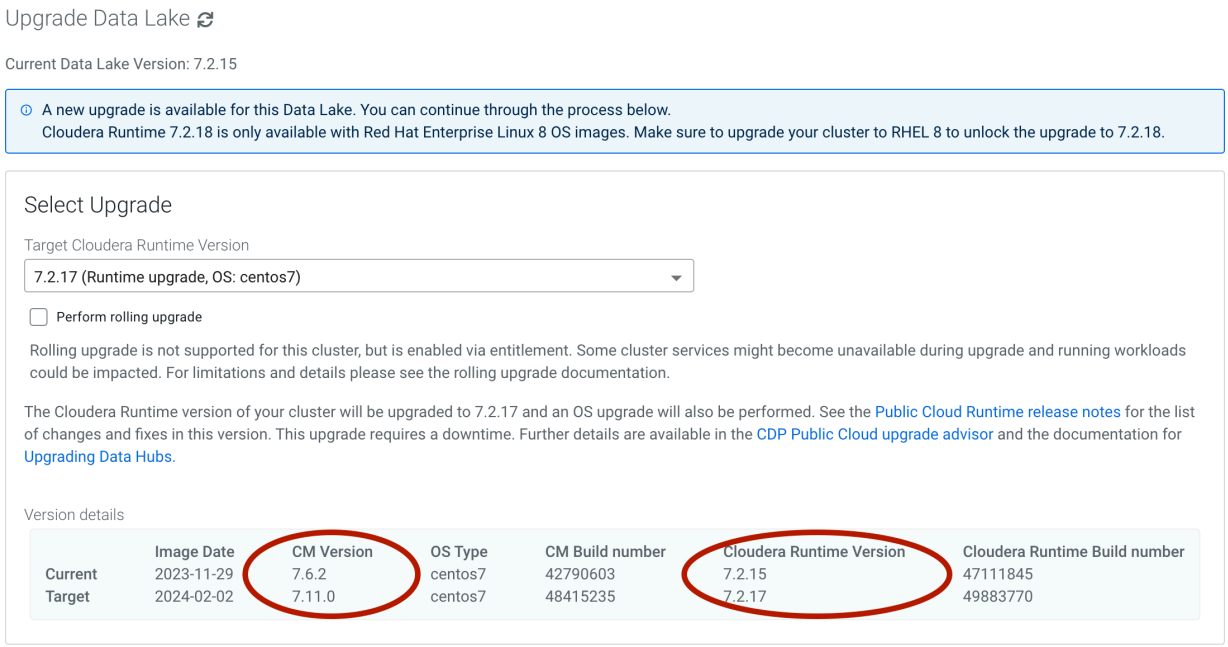

A service pack upgrade, where no version upgrade is available, but a new Cloudera Runtime and/or Cloudera Manager service pack is available for upgrade:

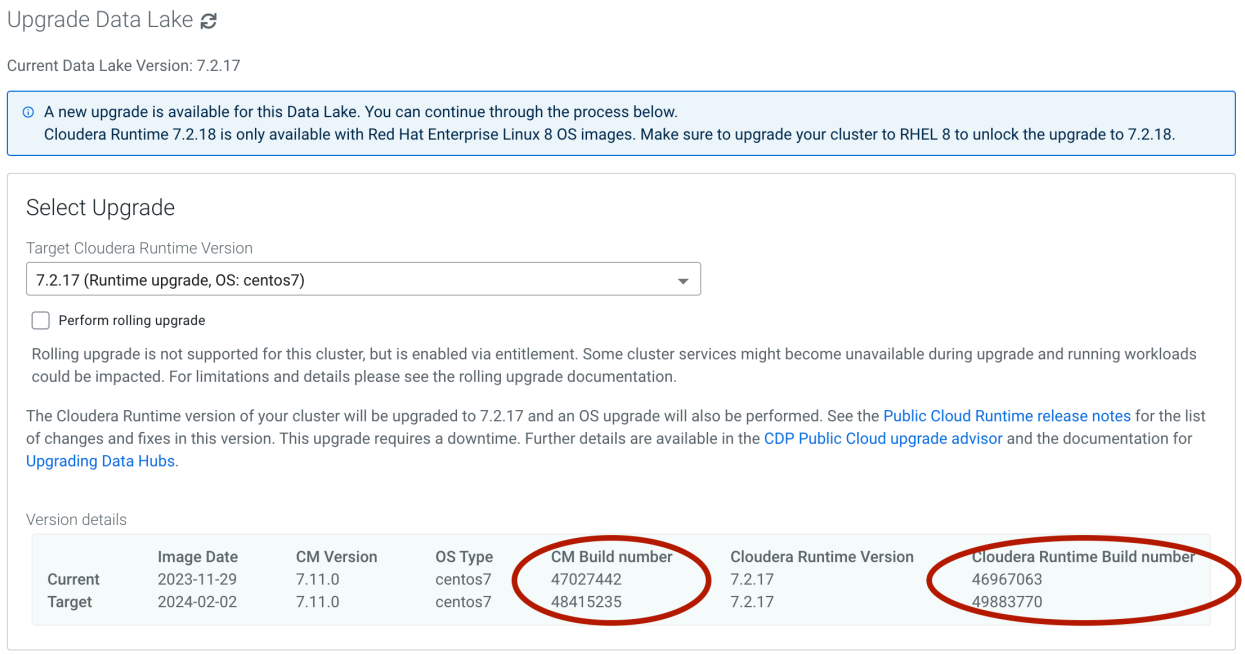

An OS upgrade is designated by “(OS upgrade, OS: <target-os>)” when you select the drop-down menu:

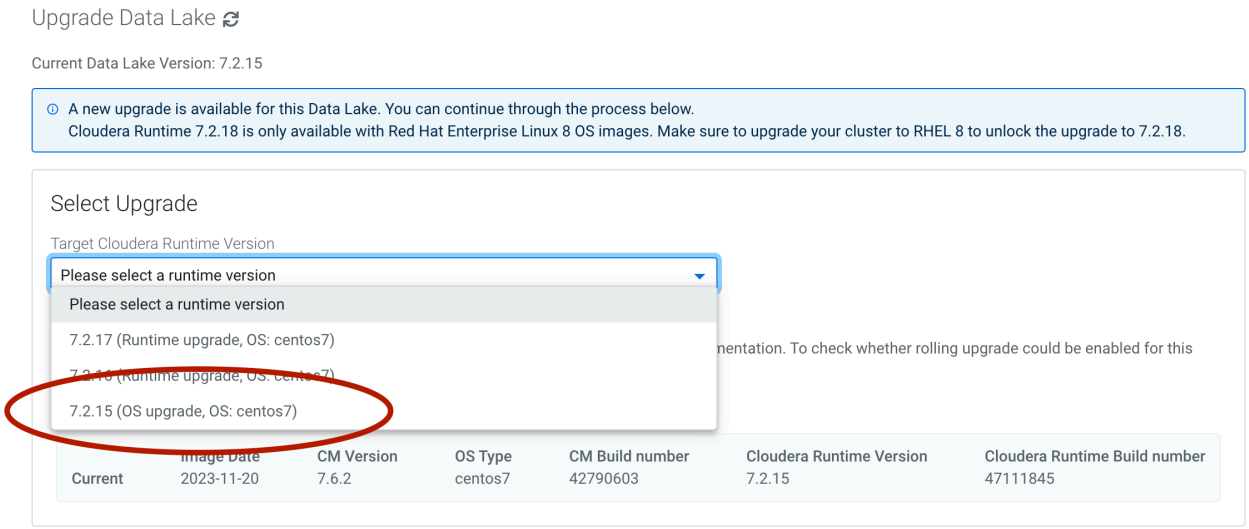

Attention: There are important differences between the nature of major/minor, service pack, and OS upgrades. Read the details below carefully to understand which type of upgrade is appropriate for your situation.

Process

The Data Lake upgrade process will:

1. Check for a newer Cloudera Manager and Cloudera Runtime version or build, and a new OS image.
2. Automatically create a backup of the Data Lake (for major/minor and service pack upgrades).
3. Execute the Data Lake upgrade.
4. Verify the Data Lake state.

Major/minor version upgrades

Major/minor version upgrades are available as new versions of Cloudera Runtime and Cloudera Manager are released. Version upgrades combine a Runtime and Cloudera Manager upgrade into one operation. For example, this upgrade might involve upgrading from Runtime 7.2.8 to 7.2.9, and Cloudera Manager from 7.4.0 to 7.4.1.

Data Lake version upgrades require you to stop all of the Cloudera Data Hub clusters in the environment before performing the Data Lake upgrade. The upgrade process stops all of the Data Lake services, downloads and distributes the new Cloudera Runtime, restarts the services, and deletes the old Cloudera Runtime. This process includes launching entirely new instance(s) from new cloud image(s). Once the Data Lake upgrade is complete, you can then either upgrade your Cloudera Data Hub clusters to the same version as the Data Lake (if your Cloudera Data Hub cluster type is supported for upgrade), or delete and recreate the clusters with the new Data Lake version. If your Data Hub is on Runtime version 7.2.16 or later, it is compatible with a Data Lake on a newer Cloudera Runtime version (7.2.17+). You can independently upgrade your Cloudera Data Hub clusters at a later time if you choose to, though it is not required

Important: Cloudera Runtime 7.2.17 has new dependencies that were not present in most of the previous Cloudera Runtime versions. Because of this, you may be unable to upgrade directly to this version and future service packs without first upgrading to a more recent Runtime version.

If you plan to upgrade your existing Data Lakes from a previous release to Cloudera Runtime 7.2.17 or later, you can verify whether or not you will first be required to perform an additional upgrade step:

1. Select the Data Lake that you want to upgrade and click on the Upgrade tab.
2. If you see a warning message about missing prerequisites, follow the given steps to perform the additional upgrade before you upgrade to 7.2.17.

If your current Runtime version is 7.2.16 or any 7.2.16 service packs, these additional steps will include first performing an OS upgrade before upgrading to 7.2.17.

If your current Runtime version is 7.2.12, 7.2.14, or 7.2.15, these steps may include upgrading to the most recent service pack of your current Cloudera Runtime version, as well as performing a separate OS upgrade for your current Cloudera Runtime version, before you can perform a major/minor version upgrade to 7.2.17.

Service pack upgrades

The service pack upgrade process checks to see if a new Cloudera Manager or Cloudera Runtime build is available, and then upgrades the Data Lake to the newest builds. Service pack upgrades do not upgrade to a new version of Cloudera Manager or Cloudera Runtime; they only upgrade to the latest service pack of a particular version. For example, a service pack upgrade cannot take the Data Lake from Cloudera Runtime version 7.2.11 to version 7.2.12, but instead upgrades Cloudera Runtime 7.2.11 to a newer 7.2.11 build. These upgrades are made available as needed and can deliver targeted bug fixes for Cloudera Runtime, Cloudera Manager, or both. If desired, specific older service packs can also be chosen using the CDP CLI. The service pack upgrade process follows the same steps as the version upgrade process.

OS upgrades

OS upgrades may be available sporadically as new images are created. OS upgrades are typically released to address security vulnerabilities or other issues on the host system. The OS upgrade process includes launching entirely new instances with the new OS image. An OS upgrade triggers the execution of any pre-service-deployment, post-cluster-manager-start, or post-service-deployment recipes.

Rolling upgrades

Certain Data Lake upgrades can be performed in a rolling fashion, depending on the Data Lake shape, Data Lake OS, and the Cloudera Runtime version you are upgrading to and from. For more information, see Data Lake rolling upgrades.

Data Lake upgrade support matrix

The following Data Lake upgrade paths are supported.

In the below table, find your current Cloudera Runtime version to find out the newest Cloudera Runtime version that you can upgrade to:

Table 1:

Current Runtime Version	Target Runtime Version
7.2.7	7.2.16
7.2.8	7.2.16
7.2.9	7.2.16
7.2.10	7.2.16
7.2.11	7.2.16
7.2.12	7.2.17
7.2.14	7.2.17
7.2.15	7.2.17
7.2.16	7.2.17
7.2.17 (RHEL + EDL)	7.2.18

Important:

- In order to upgrade to 7.2.18, your Data Lake must be using RHEL 8 and Enterprise or Light Duty shape.
- If you are upgrading to 7.2.17, you may be required to perform a service pack upgrade first.

If you are planning an update to Cloudera Runtime 7.2.18, refer to Upgrading to Runtime 7.2.18.

- For service pack upgrades, both non-RAZ (Ranger Authorization) and RAZ-enabled Data Lakes are eligible for upgrade from versions 7.2.7+.
- For major/minor version upgrades, Ranger Authorization (RAZ) enabled Data Lakes are eligible for upgrade from versions 7.2.10-7.2.12 to 7.2.14+.

Before you begin

Before you begin a Data Lake upgrade, note the requirements and limitations listed below.

Requirements

- Required role to perform Data Lake upgrade: EnvironmentAdmin or Owner over the environment
- The Data Lake must be running and in a healthy state.
- You should stop any Cloudera Data Hub clusters and any data services (such as Cloudera Data Warehouse or Cloudera Data Engineering) that are running. For the Cloudera Data Warehouse data service, you should stop any Virtual Warehouses that are running prior to beginning any upgrade or backup/restore process. Stopping data services is not required for service pack upgrades, but any Cloudera Data Hub clusters or data services that are not stopped will error out during the upgrade process.
- If you use a custom image catalog and you don’t see upgrades available, you may need to update your custom image catalog with new images.
- If the upgrade involves upgrading from CentOS to RHEL, review the Prerequisites for upgrading from CentOS to RHEL.

- Expect at least two hours of downtime while the upgrade completes. Plan the upgrade during a time of low activity.
- Optionally, you can take a backup of the Data Lake. The Data Lake upgrade process will automatically take a backup before the upgrade procedure begins, but you have the option of disabling the automatic backup if you would prefer to do this step separately. For instructions on performing a backup and restore, see *Backup and restore for the Data Lake*. If the upgrade fails for any reason, you can restore the Data Lake from the backup.

  Important: Even if you are using the automatic backup integrated with the Data Lake upgrade flow, verify that you have correctly configured the Data Lake backup process. Configuring the backup process includes granting the required permissions from the cloud provider side, and adding a RAZ policy for RAZ-enabled environments. See *Configuring and running Data Lake backups* for more information on required configurations for backup. You do not need to actually run the backup if you plan to use the auto-backup during a Data Lake upgrade.

  For RAZ-enabled AWS environments, it is required that you manually add the RAZ backup policy for any AWS Data Lake created with a Cloudera Runtime version prior to 7.2.15. However, for new environments created with a Data Lake running Runtime versions 7.2.15+, the RAZ backup policy is automatically configured for RAZ-enabled environments. If your Data Lake has been upgraded to 7.2.15 (as opposed to a new Data Lake created with 7.2.15), you will still have to manually add the RAZ backup policy.

  For RAZ-enabled Azure environments, it is required that you manually add the RAZ backup policy for any Azure Data Lake created with a Cloudera Runtime version prior to 7.2.14. However, for new environments created with a Data Lake running Runtime versions 7.2.14+, the RAZ backup policy is automatically configured for RAZ-enabled environments. If your Data Lake has been upgraded to 7.2.14+ (as opposed to a new Data Lake created with 7.2.14), you will still have to manually add the RAZ backup policy.

The upgrade requires 27 GB space on the Cloudera Manager server node and 20 GB on every other instance. If space is insufficient on your Data Lake, upgrade will not be permitted.

Limitations

Note the following limitations for the Data Lake upgrade:

- Data Lake upgrade does not include the upgrade of the FreeIPA software or the operating system on the instance(s) running FreeIPA. To upgrade FreeIPA, see *Upgrade FreeIPA*.
- Data Lake resizing (for example, moving from a light duty to a medium duty Data Lake) during an upgrade is not supported.
- Before being able to upgrade to Cloudera Runtime 7.3.1, you need to upgrade Data Lakes and all Cloudera Data Hub databases to a to PostgreSQL 14.
- When upgrading the Data Lake from Cloudera Runtime 7.2.17 to 7.2.18 or 7.3.1, if Iceberg metadata is required to be captured in Atlas, then do not use mixed versions (7.2.18 or 7.3.1 Data Lake with 7.2.17 Cloudera Data Hub), as the mismatch in the Iceberg models between 7.2.17 and 7.2.18 or 7.3.1 can cause problems.
- If a Data Lake has attached Cloudera Data Hub clusters that are not eligible for upgrade, the Data Lake itself is not eligible for upgrade. You must delete any Cloudera Data Hub clusters that are ineligible for upgrade before proceeding with the Data Lake upgrade. See *Cloudera Data Hub Upgrade* for more information about which Cloudera Data Hub clusters are eligible for upgrade.
- Service pack upgrades for RAZ-enabled Data Lakes are available only for Cloudera Runtime versions 7.2.7+.
- Major/minor version upgrades for RAZ-enabled Data Lakes are available only for Cloudera Runtime versions 7.2.12+.
- A Data Lake must be using Cloudera Runtime 7.2.17 to be eligible for CentOS to RHEL upgrade. If you do not see the option to upgrade from CentOS to RHEL, ensure that your Data Lake is using Cloudera Runtime 7.2.17.
- Runtime 7.2.18 and newer do not support Medium Duty Data Lake shape and no upgrades are possible from 7.2.17 to 7.2.18 without doing a resize operation on the Data Lake prior to upgrading to 7.2.18.

Related Information

Backup and restore for the Data Lake

Upgrade FreeIPA

Cloudera Data Hub Upgrade
Configuring and running Data Lake backups

Upgrading a Data Lake

If a new Cloudera Runtime/Cloudera Manager version or build is available for the Data Lake, you can initiate an upgrade from the Management Console. An OS upgrade may also be available.

About this task

In most cases it is not required that you destroy/recreate any Cloudera Data Hub clusters attached to the Data Lake cluster. For major/minor version upgrades, you must upgrade the Cloudera Data Hub clusters themselves after you upgrade the Data Lake, with the exception of Cloudera Data Hub clusters on Cloudera Runtime version 7.2.16 and later. If your Cloudera Data Hub cluster is on Cloudera Runtime version 7.2.16 or later, it is compatible with a Data Lake on a newer Cloudera Runtime version (7.2.17+). You can independently upgrade your Cloudera Data Hub clusters at a later time if you choose to, though it is not required.

Any Cloudera Data Hub clusters or data services that are not stopped during a Data Lake upgrade will error out during the upgrade process.

Required role: EnvironmentAdmin or Owner over the environment

Procedure

1. Stop all Cloudera Data Hub clusters attached to the environment.
2. From the Management Console, click Data Lakes <Environment Name> , scroll to the bottom of the Data Lake details page, and click the Upgrade tab.

3. Click the Target Runtime Version drop-down menu to see any available upgrades for a given Cloudera Runtime version.

   If a new build is available for the selected version, the UI displays the current and target versions and build numbers. If only an OS upgrade is available, the UI displays “(OS upgrade only).”

   When a major/minor version upgrade is available, you'll be able to select a new Cloudera Runtime version:

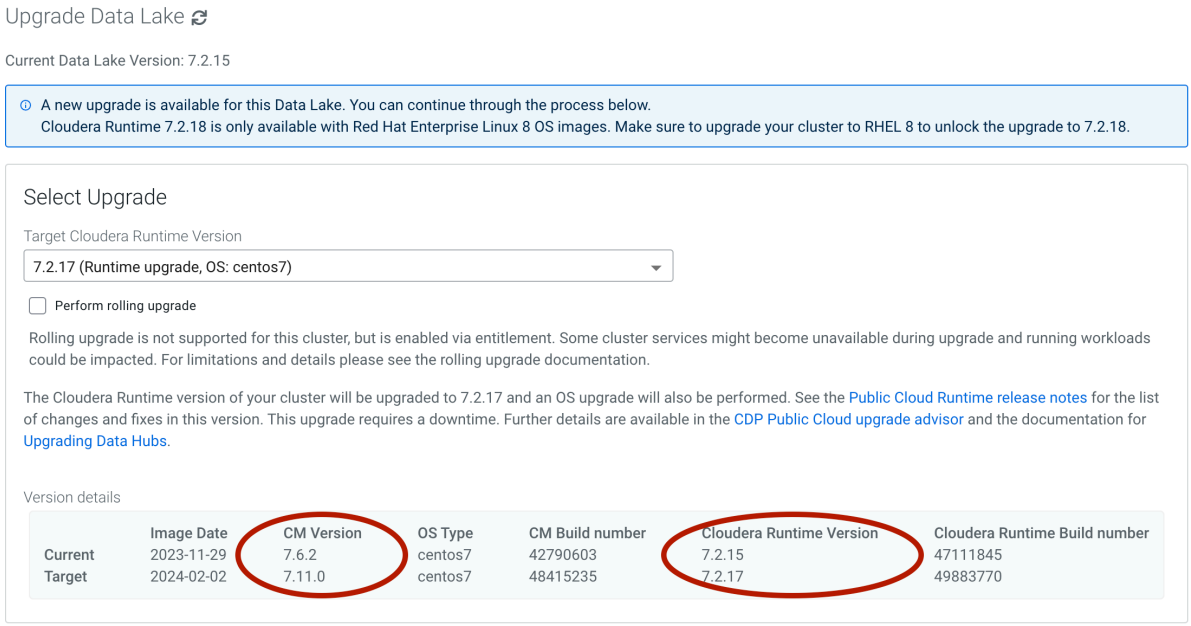

   If a rolling upgrade is available, select the Perform rolling upgrade checkbox if you would like to perform this type of upgrade. The availability of a rolling upgrade depends on the current and target Cloudera Runtime versions, the Data Lake shape, and the Data Lake OS. See Data Lake rolling upgrades for more information.

4. If you want to skip the automatic backup that is taken before the upgrade, uncheck the Automatic backup box. For more information on what is backed up during a Data Lake backup, see *Data Lake backup and restore*.
5. Click Validate and Prepare to check for any configuration issues and begin the Cloudera Runtime parcel download and distribution. Using the validate and prepare option does not require downtime and makes the maintenance window for an upgrade shorter. Validate and prepare also does not make any changes to your cluster and can be run independently of the upgrade itself. Although you can begin the upgrade without first running the validate and prepare option, using it will make the process smoother and the downtime shorter.
6. Click Upgrade to initiate the upgrade.
7. Click the Event History tab to monitor the upgrade process and verify that it completes successfully.

   If the upgrade fails for any reason, check the Data Lake logs through Cloudera Manager for troubleshooting information and retry the upgrade. If you cannot fix the problem manually, you may be able to recover the Data Lake cluster after a failed upgrade. For more information see *Recovering from failed upgrades*.

What to do next

For major/minor upgrades, if the upgrade is successful, you can proceed to upgrading your attached Cloudera Data Hub clusters if required. Cloudera Data Hub clusters must run the same Cloudera Runtime version as the Data Lake, with the exception of Cloudera Data Hub clusters on Cloudera Runtime version 7.2.16 and later. If your Cloudera Data Hub cluster is on Cloudera Runtime version 7.2.16 or later, it is compatible with a Data Lake on a newer Cloudera Runtime version (7.2.17+). You can independently upgrade your Cloudera Data Hub clusters at a later time if you choose to, though it is not required. For service pack and OS upgrades, you can restart your Cloudera Data Hub clusters, data services, and any stopped Virtual Warehouses.

Related Information

Backup and restore for the Data Lake

Recovering from failed upgrades

Upgrading a Data Lake manually via CLI

You can initiate a Data Lake upgrade with the CDP CLI. Using the same CLI command, you can also search for and validate available images to upgrade to, and generate JSON templates for specific upgrade scenarios.

Obtain image ID

If your Data Lake upgrade includes upgrading from CentOS to RHEL 8, prior to attempting an upgrade you need to obtain an ID of a target RHEL 8 image. You can obtain it from the image catalog by finding an image with your target Runtime version which has an OS Type of RHEL8.

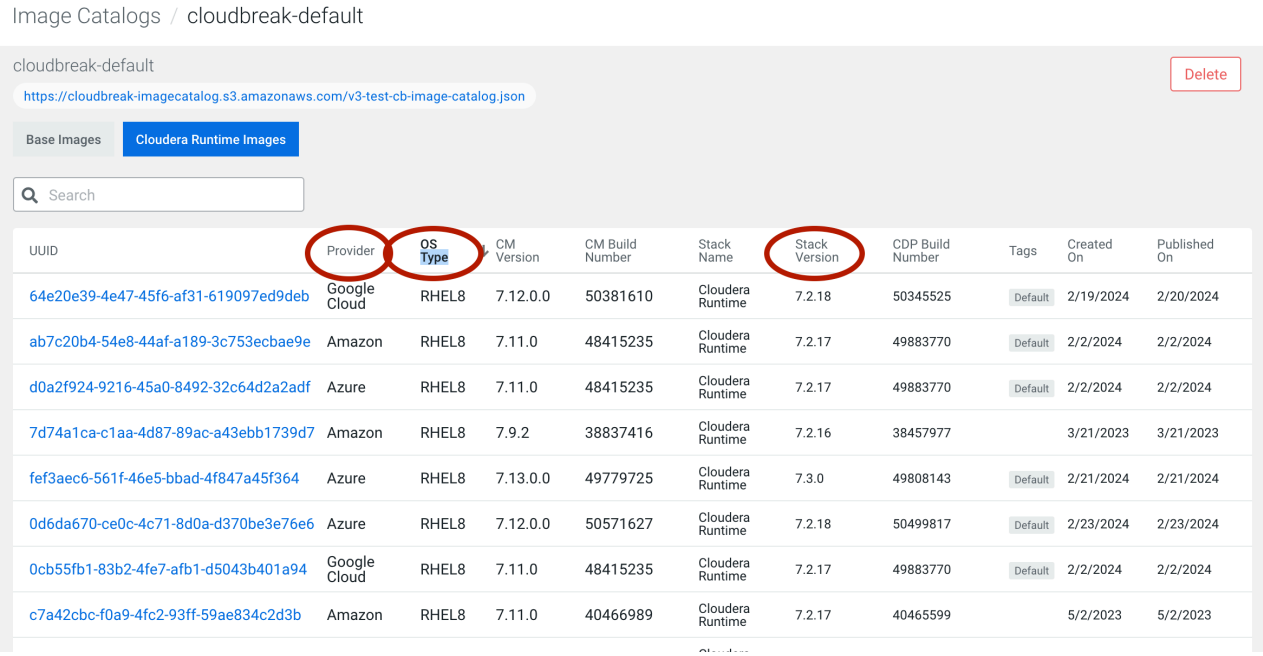

UUID	Provider	OS Type	CM Version	CM Build Number	Stack Name	Stack Version	CDP Build Number	Tags	Created On	Published On
64e20e39-4e47-45f6-af31-619097ed9deb	Google Cloud	RHEL8	7.12.0.0	50381610	Cloudera Runtime	7.2.18	50345525	Default	2/19/2024	2/20/2024
ab7c20b4-54e8-44af-a189-3c753ecbae9e	Amazon	RHEL8	7.11.0	48415235	Cloudera Runtime	7.2.17	49883770	Default	2/2/2024	2/2/2024
d0a2f924-9216-45a0-8492-32c64d2a2adf	Azure	RHEL8	7.11.0	48415235	Cloudera Runtime	7.2.17	49883770	Default	2/2/2024	2/2/2024
7d74a1ca-c1aa-4d87-89ac-a43ebb1739d7	Amazon	RHEL8	7.9.2	38837416	Cloudera Runtime	7.2.16	38457977		3/21/2023	3/21/2023
fef3aec6-561f-46e5-bbad-4f847a45f364	Azure	RHEL8	7.13.0.0	49779725	Cloudera Runtime	7.3.0	49808143	Default	2/21/2024	2/21/2024
0d6da670-ce0c-4c71-8d0a-d370be3e76e6	Azure	RHEL8	7.12.0.0	50571627	Cloudera Runtime	7.2.18	50499817	Default	2/23/2024	2/23/2024
0cb55fb1-83b2-4fe7-afb1-d5043b401a94	Google Cloud	RHEL8	7.11.0	48415235	Cloudera Runtime	7.2.17	49883770	Default	2/2/2024	2/2/2024
c7a42cbc-f0a9-4fc2-93ff-59ae834c2d3b	Amazon	RHEL8	7.11.0	40466989	Cloudera Runtime	7.2.17	40465599		5/2/2023	5/2/2023

Once you have identified the ID, you can provide it in the upgrade CLI command by using the --image-id flag.

Upgrade steps

1. Run the cdp datalake upgrade-datalake command. In order to use this command for upgrading from CentOS to RHEL, ensure to provide an image ID of a RHEL 8 image.

   The command has the following options:

```
cdp datalake upgrade-datalake
          --datalake-name <value>
          [--image-id <value>]
          [--runtime <value>]
          [--lock-components | --no-lock-components]
          [--dry-run | --no-dry-run]
          [--show-available-images | --no-show-available-images]
          [--show-available-image-per-runtime | --no-show-available-image-
per-runtime]
          [--skip-backup | --no-skip-backup]
          [--skip-ranger-hms-metadata | --no-skip-ranger-hms-metadata]
          [--skip-atlas-metadata | --no-skip-atlas-metadata]
          [--skip-ranger-audits | --no-skip-ranger-audits]
          [--skip-backup-validation | --no-skip-backup-validation]
          [--cli-input-json <value>]
```

```
            [--generate-cli-skeleton]
```

Option	Description
--datalake-name (string)	Required. The name or CRN of the Data Lake to upgrade.
--image-id (string)	The ID of an image to upgrade to. If upgrading from CentOS to RHEL, make sure to provide an image ID of a target RHEL image.
--runtime (string)	The Runtime version to upgrade to. When you specify the Cloudera Runtime version, the upgrade uses the latest image ID of the given Cloudera Runtime version from the same image catalog used for Data Lake creation.
--lock-components | --no-lock-components (boolean)	Use --lock components to perform an OS upgrade only.
--dry-run | --no-dry-run (boolean)	Checks the eligibility of an image to upgrade. Can be used in conjunction with any other parameter, returning the available image (with respect to image Id, Runtime or lock-components set) without performing any actions.
--show-available-images | --no-show-available-images (boolean)	Returns the list of images that are eligible to upgrade to.
--show-available-image-per-runtime | --no-show-available-image-per-runtime (boolean)	Returns the latest image that is eligible to upgrade to, for each Cloudera Runtime version with at least one available upgrade candidate.
--skip-backup | --no-skip-backup	If provided, will skip the backup flow for the upgrade process.
--skip-ranger-hms-metadata | --no-skip-ranger-hms-metadata	Skips the backup of the databases backing HMS/Ranger services. Redundant if –skip-backup is included. If this option is not provided, the HMS/Ranger services are backed up by default.
--skip-atlas-metadata | --no-skip-atlas-metadata	Skips the backup of the Atlas metadata. Redundant if –skip-backup is included. If this option is not provided, the Atlas metadata is backed up by default.
--skip-ranger-audits | --no-skip-ranger-audits	Skips the backup of the Ranger audits. Redundant if –skip-backup is included. If this option is not provided, Ranger audits are backed up by default.
--skip-backup-validation | --no-skip-backup-validation	Skips the validation steps that run prior to the backup. Redundant if –skip-backup is included. If this option is not provided, the validations are performed by default.
--cli-input-json (string)	Performs service operation based on the JSON string provided. The JSON string follows the format provided by --generate-cli-skeleton. If other arguments are provided on the command line, the CLI values will override the JSON-provided values.
--generate-cli-skeleton (boolean)	Prints a sample input JSON to standard output. Note the specified operation is not run if this argument is specified. The sample input can be used as an argument for --cli-input-json.

When you run the cdp datalake upgrade-datalake command to initiate an upgrade, you have one of three options:

a. Specify one of either --image-id, --runtime, or --lockComponents, which makes an explicit choice of the exact image, Runtime (latest OS), or latest OS (same Runtime) for upgrade.

b. Specify both --image-id and --lockComponents, which specifies an image and ensures the image represents an OS only upgrade.

c. Specify none of the --image-id, --runtime, or --lockComponents parameters, which initiates a Runtime/CM upgrade to the latest compatible version and OS image.

Outside of upgrade, you can use the following options:

- --show-available-images/--no-show-available-images
- --show-available-images-per-runtime/--no-show-available-images-per-run time
- --dry-run

Examples of valid inputs:

```
cdp datalake upgrade-datalake --datalake-name my-datalake --dry-run
cdp datalake upgrade-datalake --datalake-name my-datalake --image-id d1c
520b1-987d-461f-7860-918f43994c04
cdp datalake upgrade-datalake --datalake-name my-datalake --image-id d1c52
0b1-987d-461f-7860-918f43994c04 --dry-run
cdp datalake upgrade-datalake --datalake-name my-datalake --runtime 7.2.11
cdp datalake upgrade-datalake --datalake-name my-datalake --runtime 7.2
.11 --dry-run
cdp datalake upgrade-datalake --datalake-name my-datalake--lock-components
cdp datalake upgrade-datalake --datalake-name my-datalake --show-availa
ble-image-per-runtime
cdp datalake upgrade-datalake --datalake-name my-datalake- --show-availabl
e-images
```

Examples of incorrect inputs:

```
cdp datalake upgrade-datalake --datalake-name my-datalake --image-id 7.2
.11
cdp datalake upgrade-datalake --datalake-name my-datalake--runtime d1c52
0b1-987d-461f-7860-918f43994c04
cdp datalake upgrade-datalake --datalake-name my-datalake--lock-components
 --imageid imageid --runtime runtime
cdp datalake upgrade-datalake --datalake-name my-datalake --show-availabl
e-image-per-runtime --show-available-images
cdp datalake upgrade-datalake --datalake-name my-datalake --show-availab
le-image-per-runtime --dry-run
cdp datalake upgrade-datalake --datalake-name my-datalake --show-availabl
e-images --dry-run
```

Data Lake rolling upgrades

The Data Lake rolling upgrade allows you to upgrade the Data Lake Cloudera Runtime and OS without stopping attached Cloudera Data Hub clusters or Data Services. This allows workloads to continue running during the Data Lake upgrade operation.

Similarly to the classic Data Lake upgrade logic, a Data Lake rolling upgrade first upgrades the Cloudera Runtime version and then the OS.

To run a Data Lake rolling upgrade, the following requirements must be met:

- The Data Lake must be Cloudera Runtime version 7.2.17.300+ to perform a rolling upgrade directly to 7.2.18+.
- The Data Lake must be Cloudera Runtime version 7.2.17.200 - 7.2.17.500 or 7.2.18.0 - 7.2.18.200 to perform a rolling upgrade directly to Cloudera Runtime 7.3.1.
- The Data Lake OS must be RHEL 8. Rolling upgrades are not generally available from Data Lakes on CentOS.
- The Data Lake must be an Enterprise Data Lake. Rolling upgrades to 7.2.18 or later versions are not generally available from Medium Duty or Light Duty Data Lakes.

If your Data Lake does not meet these requirements, you will first have to do a traditional Cloudera Runtime upgrade, upgrade your OS from CentOS to RHEL, or resize your Data Lake to Enterprise Data Lake before a rolling upgrade can be performed.

Current Runtime version	Current OS	Current Data Lake Shape	Rolling upgrade support?
7.2.17.300+	RHEL 8	Enterprise Data Lake	Yes, directly to 7.2.18+
7.2.17.200 - 7.2.17.500	RHEL 8	Enterprise Data Lake	Yes, directly to 7.3.1
7.2.17.200 - 7.2.17.500	RHEL 8	Enterprise Data Lake	Yes, directly to 7.3.1

In some circumstances, a rolling upgrade may not be supported for a Data Lake cluster, but can be enabled through entitlement. Some cluster services might become unavailable during this type of upgrade, and running workloads could be impacted. The Data Lake upgrade UI displays information about whether a rolling upgrade is available, unavailable, or may be available under entitlement. For instructions on performing a Data Lake upgrade, including rolling upgrades, see Upgrading a Data Lake. For information about obtaining an entitlement for rolling upgrade, contact Cloudera Customer Support.

Data Lake rolling upgrade limitations and issues

The Data Lake rolling upgrade has the following limitations:

- Long running Cloudera Data Engineering and Cloudera AI Spark jobs might loose connectivity to the Hive Catalog and fail during and after a Data Lake rolling upgrade. We recommend stopping these jobs prior to attempting the upgrade. The jobs will work again fine after re-submitting them.
- Cloudera recommends performing the upgrade outside of working hours, as user-facing UI/API endpoints may become unstable. Workloads running on Cloudera Data Hub clusters and Data Services use different internal endpoints, so they are not affected. The impact of this is that you may not be able to view or edit Ranger permissions and the Ranger audit log, browse Atlas/Cloudera Data Catalog, or make changes to Atlas at certain times during the rolling Data Lake upgrade (see Known Issues in Apache Atlas for more details on Atlas issues). If you are using custom-built applications that interact with the Data Lake using these endpoints, we recommend implementing retry logic in your clients to handle temporary unavailability of these endpoints. This is a best practice, irrespective of rolling upgrades.
- Atlas Authorization may return a “403-Access Denied” in response to Atlas REST API calls. After the rolling upgrade finishes and Ranger Admin is back up, these services and their endpoints will continue to function normally.
- The Ranger RAZ server becomes unreachable during and after the OS upgrade for some time, and an UnknownHostException may be seen in the RAZ client. During this period all authentication calls to the Ranger RAZ server are expected to fail.
- During OS upgrades, attempts to access Knox on the host being upgraded may produce occasional 403 HTTP responses. Wait and retry the failed requests.
- If Knox is HA and one of the Knox servers is down, then accessing the service through a Control Plane endpoint URL (i.e., through cloud load balancer) will take approximately 30 seconds to failover the request to the available Knox instance. This also means that the services that are reached through Knox will not be available behind Knox during this period time.
- Solr supports rolling upgrades from release 7.2.18 and higher. Upgrading from a lower version means that all the Solr Server instances are shut down, parcels upgraded and activated, and then the Solr Servers are started again. This causes a service interruption of several minutes, the actual value depending on cluster size. Services like Atlas and Ranger that depend on Solr may face issues because of this service interruption.

- Certain workloads may experience downtime during the Data Lake rolling upgrade operations:
  - Any workloads configured to use a single HMS endpoint (Hive Warehouse Connector configurations).
  - Hue File Browser may be unavailable for a short period during a rolling upgrade in a RAZ-enabled environment.
  - Other clients in use in your workloads could be impacted. Cloudera recommends testing how your workloads function during Data Lake rolling upgrades, before you adopt this new feature.
  - During the rolling upgrade, Hive and HBase grant and revoke commands will not function.
  - Generally, if you have any workload that is using a single Data Lake service endpoint, it will likely experience a temporary outage. This may not necessarily result in a workload failure.
- Rolling upgrades for an enterprise Data Lake will take longer than a classic upgrade that requires downtime. This is because OS image upgrade will be performed sequentially, node by node.
- Certain operations (create, upgrade, and resume) for Cloudera Data Hub clusters and Data Services are not recommended during a Data Lake rolling upgrade.
- When upgrading Cloudera Data Hub clusters to Cloudera Runtime 7.2.18.100, you might encounter staleness in knox.jwt.client.gateway.address configuration in case its value points to the address of the Data Lake node. If staleness occurs after the upgrade, you need to run Deploy Client Configuration in Cloudera Manager.

Recovering from failed upgrades

If a Data Lake upgrade fails and you are unable to manually troubleshoot the problem, you may be able to use the recovery process to return the cluster to its pre-upgrade state.

About this task

If FreeIPA is available and the Data Lake cluster is in a recoverable state (meaning that there has been an uncorrected failed upgrade or failed recovery), a recovery option may be available after a failed upgrade. Recovery after a failed upgrade retains the Data Lake CRN, UMS mappings, load balancers, and RDS instance and brings up new instances with the original image and Runtime version, but new disks and new databases.

Note: Data backup and restore is not currently part of the recovery process. Ensure that you have a Data Lake backup from which you can manually restore the Data Lake data after the successful recovery. The presence of the backup is not validated by the Cloudera Management Console.

Procedure

1. Use the CDP CLI to recover the Data Lake after a failed upgrade:

```
cdp datalake recover-datalake
--datalake-name <value>
[--recovery-type <value>]
```

Parameter	Description
--datalake-name	Name or CRN of the Data Lake that you want to recover after a failed upgrade.
--recovery-type	The type of the recovery. The default value is RECOVER_WITHOUT_DATA.

Parameter	Description
	Currently, the option RECOVER_WITH_DATA is not supported.

The status of the Data Lake appears as "Datalake recovery in progress. Recovery process takes a while as the nodes are being terminated and new nodes are launched with the original runtime."

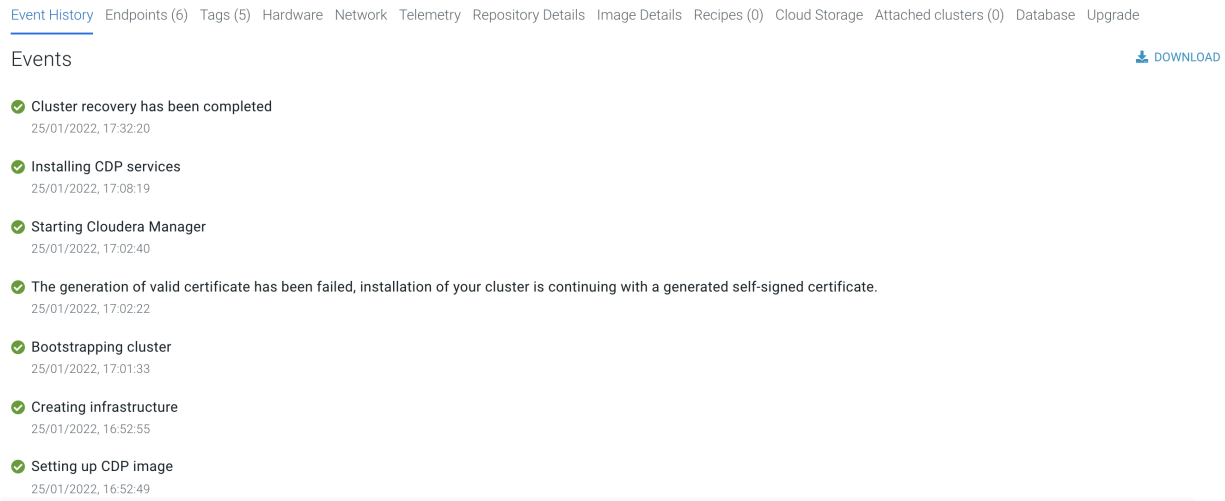

2. Restore the Data Lake from the pre-upgrade backup. For more information, see *Restore Data Lake content*.
3. If necessary, run the cdp datalake sync-component-versions-from-cm command from the CDP CLI.

   When an upgrade fails, the versions of Cloudera Manager, Cloudera Runtime, and other components may become out-of-sync with the Cloudera Management Console. Similarly, if you try to fix errors by installing parcels manually, it may not be reflected in the Cloudera Management Console.

   To overcome the mismatch between versions reflected in the Management Console, run the cdp datalake sync-component-versions-from-cm CDP CLI command. This commands reads the Cloudera Management Console, Cloudera Runtime, and other parcel versions (if applicable) from Cloudera Manager and updates the versions in the Cloudera Management Console. Using this command forces the Cloudera Management Console back in sync so that it shows the actual versions installed in Cloudera Manager.

```
cdp datalake sync-component-versions-from-cm --datalake-name <datalake n
ame>
```

Related Information

Configuring and running Data Lake restore

Performing manual Data Lake repair

If a Data Lake node fails, an administrator can start a manual recovery process from the Cloudera web interface. Because the state of Data Lake services is stored externally, the repair operation is able to deploy the services on a new node and reattach the all workload clusters without data loss and with minimal downtime.

Required role: EnvironmentAdmin or Owner of the environment

When a Data Lake cluster has unhealthy nodes, warnings appear in the Data Lake page:

- Nodes are marked as "UNHEALTHY" in the Hardware tab for the Data Lake.
- Data Lake cluster's Event History shows "Manual recovery is needed for the following failed nodes."

You can perform manual repair from the Cloudera web UI or the CDP CLI.

Manual repair from web UI

To perform manual repair from Cloudera web UI:

1. Log in to the Cloudera web interface.
2. Navigate to the affected Data Lake using Cloudera Management Console Data Lakes .
3. In the Data Lake details page, click choose one of the following options:
   - To repair failed nodes in a specific host group, click Repair and select the host group that should be repaired. Only one host group can be selected at a time. Then click Repair.

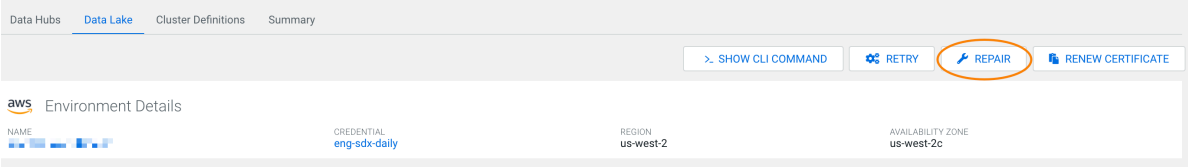

   Note: If no host groups are listed as in need of repair, use Cloudera Manager to determine what might be causing the problem you are experiencing.

   - To repair a single node failure or select certain nodes within a host group to repair, select the Hardware tab and then the repair icon next to the host group that contains the failed node(s).
4. When you initiate a repair from the Hardware tab, you also have the option to delete any volumes attached to the instance. This can be useful if a volume is lost on the cloud provider side. To delete the attached volumes, select the Delete Volumes checkbox.

When the recovery flow is completed, the cluster status changes to "RUNNING".

Manual repair from CLI

To perform manual repair from the CLI, use the following commands:

- cdp datalake list-datalakes – Check the status and health of your Data Lake clusters
- cdp datalake describe-datalake – Check the status and health of a specific Data Lake cluster
- cdp datalake repair-datalake – Perform Data Lake cluster repair.

Related Information

Data Lake repair

Data Lake storage

Cloudera Manager health tests

Cloudera Manager logs

Backup and restore for the Data Lake

You can backup and restore the metadata maintained in the Data Lake services. Use this comprehensive backup to restore your Data Lake's metadata to the state it was at when the backup was taken.

The backup and restore operation creates a comprehensive backup that improves the likelihood of data in the backup to be synchronized across all the services running in the Data Lake.

Required role: EnvironmentAdmin

Note: Data Lake backup and restore is supported from Cloudera Runtime 7.2.1+ on AWS and Cloudera Runtime 7.2.2+ on Azure, Cloudera Runtime 7.2.15+ on GCP, and Cloudera Runtime 7.1.0/7.2.0 on HDFS.

Use the Data Lake backup and restore:

- in preparation for a Data Lake upgrade.
- for archiving Data Lake content for later reference.

- to promote Data Lake content from test to production.

There will be a downtime when a Data Lake backup is performed, as some Data Lake services will be stopped. Additionally, access to the HMS database will be blocked by default for the duration of the backup, but you can optionally bypass this using the CLI option no-close-db-connnections when you run the backup. This option allows Data Hub workloads to continue running during the Data Lake backup process. See *Configuring and running Data Lake backups* for more information.

Important: Data Lake backup/restore operations should be performed when the Data Lake is quiescent. In order to have a consistent backup Cloudera recommends that no workloads are running when the backup is performed. In order to make sure there are no updates to HMS, the backup and restore process closes all the connections to the HMS backend database. This is important for migration use cases where you would like to move metadata from one environment to another. However, you are not required to stop the Data Hub workloads before performing backup. If you want to keep Cloudera Data Hub clusters running during a backup, you must keep the HMS database connections open to the Data Lake using the no-close-db-connnections CLI option. If the database connections are closed, Data Hub workloads will fail.

Note that a Data Lake Backup includes metadata about your cluster workloads and does not include the data itself.

At this time, you can trigger a Data Lake backup through a Cloudera command-line operation from any host with connectivity to the Data Lake. The system checks to make sure there isn't another backup or restore in progress.

What's backed up?

The backup operation saves a full snapshot of data from all SDX services:

- Atlas:
  - Audit events, saved in HBase tables
  - Lineage data, saved as Janus graph data in HBase tables
  - Edge, vertex, and full text indexes, saved in Solr collections
- Ranger:
  - Audit logs, saved as a Solr collection
  - Permissions and tags, saved in RDBMS tables
- HMS Metadata, saved in RDBMS tables

How do I keep the backup secure?

The backup files are stored on AWS S3, ADLS, or Google Cloud Storage with encryption enabled.

How often should backups run?

You can run backups as part of these events:

- Upgrades: a backup can be performed before performing an upgrade. This backup can be used to restore the existing environment or create a new environment in case the upgrade fails in a manner that requires you to re-build the original environment.
- Moving the Data Lake metadata (Atlas lineages, Ranger policies and audit information, and HMS metadata) from one environment to another.

When backups are taken, there is downtime for principal services (see "Principal services" below) due to the requirement to shut down HMS service on any attached Data Hubs.

Is there any validation that occurs before the backup/restore operations?

Yes. Before a backup or restore operation begins, a validation process checks for the most common causes of a backup or restore failure. These validations include verifying that the required permissions are granted on the backup location with the cloud provider (AWS and Azure) and that the required Ranger RAZ permissions are granted on the

backup location. For the backup operation only, the validation process also checks if there are any stale Ranger Audit permissions that are over 90 days old.

You can use CLI commands when you run the backup/restore operations to skip the validation process, or run the validation process without proceeding to the backup/restore operations. See *Configuring and running Data Lake backups* and *Configuring and running Data Lake restore* for more information.

Data Lake Restore vs Repair

Data Lake repair replaces the compute resources and reconnects them to the persistent Data Lake storage. Data Lake restore replaces the existing Data Lake content with content from a Data Lake backup.

Principal services

The following principal services affect backup and restore operations:

- On the Data Lake:
  - Atlas
  - HMS
- HMS Services on any attached Cloudera Data Hub cluster.

When a backup is performed, the Atlas service is stopped. The HMS service will be impacted if the "--no-close-db-connections" option is not provided.

Dependent services

Backup and restore operations are dependent on the following services:

- HBase
- Solr
- ZooKeeper
- Databases services (e.g. Postgres)

They must be running during a backup or restore operation.

Other considerations

Do not stop or restart Data Lake services in Cloudera Manager if you are planning on running backup or restore operations. If you stop or restart Data Lake services from Cloudera Manager, such as restarting all services, or restarting the HBase or Solr services, Data Lake backup and restore operations from the CLI will be allowed to go forward, but may fail.

In order to have a consistent backup Cloudera recommends that no workloads are running when the backup is performed. In order to make sure there are no updates to HMS, the backup and restore process closes all the connections to the HMS backend database. This is important for migration use cases where you would like to move metadata from one environment to another. However, you are not required to stop the Data Hubs attached to a Data Lake before performing backup. If you want to keep Data Hubs running during a backup and restore operation, you must keep the HMS database connections open to the Data Lake using the no-close-db-connnections CLI option. If the database connections are closed, Data Hub workloads will fail.

You should stop all Cloudera Data Hub clusters before performing a restore operation; a running Cloudera Data Hub cluster will prevent restore from functioning, and may result in inconsistent data in the backup.

What's supported?

Support for AWS	Cloudera Runtime 7.2.1 and above
Support for Azure	Cloudera Runtime 7.2.2 and above
Support for GCP	Cloudera Runtime 7.2.15 and above

Support for Backup and restore across Data Lakes with different shapes For example, a backup taken on a light duty Data Lake can be used to restore a medium duty Data Lake.	Cloudera Runtime 7.2.8 and above
Support for RAZ-enabled Data Lakes	Cloudera Runtime 7.2.10 and above
Support for Cloudera Runtime 7.1.0 and 7.2.0	Data Lake backup only writes to HDFS on the Data Lake. The backup should be moved to and from cloud storage using the provided procedures. Contact Cloudera Support for more information.
Cross-version support (Restoring a backup taken from a different version of Cloudera Runtime)	You can take a backup of a Data Lake that runs one version of Cloudera Runtime and restore the backup to a Data Lake that runs a different version of Cloudera Runtime. The backup version must be an earlier/lower version Cloudera Runtime than the Data Lake that you are restoring to. Version limitations apply and a Ranger/HMS schema upgrade may be required. See *Cross-version support for Data Lake backup and restore* for more details.

What's not supported?

Backup and restore across different cloud providers is not supported	Example: Restoring a backup of an AWS Data Lake to an Azure Data Lake, or an Azure Data Lake to an AWS Data Lake is not supported.
Recovery for individual settings or pieces of metadata.	Example: Recovering just the Hive SQL Ranger policies from a given backup, or recovering just the HMS metadata of a particular database.
Point in Time Recovery of data synced with metadata	A restore will only recover HMS metadata and will apply that to the existing data used by the Data Hubs of the environment. This means that HMS metadata may not be in sync with the data.

Limitations in the backup and restore system

- With Cloudera Runtime 7.2.1, the initial backup on AWS may be written to an S3 bucket. All subsequent backups must be written to the same bucket as the initial backup. This limitation does not exist for later runtime versions.

Related Information

Cross-version support for Data Lake backup and restore

Cross-version support for Data Lake backup and restore

You can take a backup of a Data Lake that runs one version of Cloudera Runtime and restore the backup to a Data Lake running a different version of Runtime.

Important: Depending on the target backup and restore versions, a schema update may be required for Ranger and HMS. See Upgrade Ranger and HMS schema after Data Lake restore.

Cross-version backup and restore is supported for the following Data Lake versions:

Source (Backup) Data Lake Runtime Version	Target (Restore) Data Lake Runtime Version	Schema Upgrade Required?
7.2.12	7.2.15	Yes
7.2.12	7.2.16	Yes
7.2.14	7.2.15	No
7.2.14	7.2.16	Yes
7.2.15	7.2.16	Yes
7.2.15	7.2.17	Yes
7.2.16	7.2.17	No

Source (Backup) Data Lake Runtime Version	Target (Restore) Data Lake Runtime Version	Schema Upgrade Required?
7.2.16	7.2.18	Yes
7.2.17	7.2.18	Yes

For these Cloudera Runtime versions, cross-version backup and restore is supported for all cloud providers and all Data Lake shapes. You can also restore your Data Lake in a new environment using the created backup from a previously created environment if the environments belong to the same tenant ID.

Configuring and running Data Lake backups

The Data Lake provides a command line interface for managing Data Lake backup and restore operations. The system checks to make sure there isn't another backup or restore in progress.

Configure the backup

Before you begin

- Create the S3, ABFS, or GCS backup location before performing the backup. For Azure, the container where the backup is stored should be in the same storage account as the Data Lake being backed up.
- Shut down principal services (see Principal services on page 55). This will help avoid mismatches between Data Lake metadata and data used by workloads and mismatches among the metadata stored in the Data Lake.
- Stop all Cloudera Data Hub clusters attached to the Data Lake before you perform any backup or restore operations.
- Stop any Virtual Warehouses that are running.

Configuring backups for AWS:

- Apply the IAM policy for Data Lake backup to the following roles:
  - DATALAKE_ADMIN_ROLE
  - RANGER_AUDIT_ROLE

    For more informaiton on IAM roles, see *Minimal setup for cloud storage*.

    In the IAM policy for Data Lake backup, be sure to replace the <BACKUP_BUCKET> variable with the backup location used.

Note that if you plan to restore the Data Lake backup that you are taking, you must also apply a restore policy to certain roles. For more information on restore see Configuring and running Data Lake restore on page 65.

Configuring RAZ for backup

This section applies only to RAZ-enabled AWS Data Lakes. For RAZ-enabled Azure Data Lakes, see the section below.

Add a RAZ policy to allow the backups to be written to and read from the backup location.

- Open the Ranger UI.
- Go to the cm_s3 policy list.

- Add a new policy:
  - Policy name: backup_policy
  - S3 bucket: The bucket where backups will be stored
  - Path: The path(s) in the bucket where backup will be written

    Note: If more than one bucket will be used for backup, create a separate policy for each bucket.

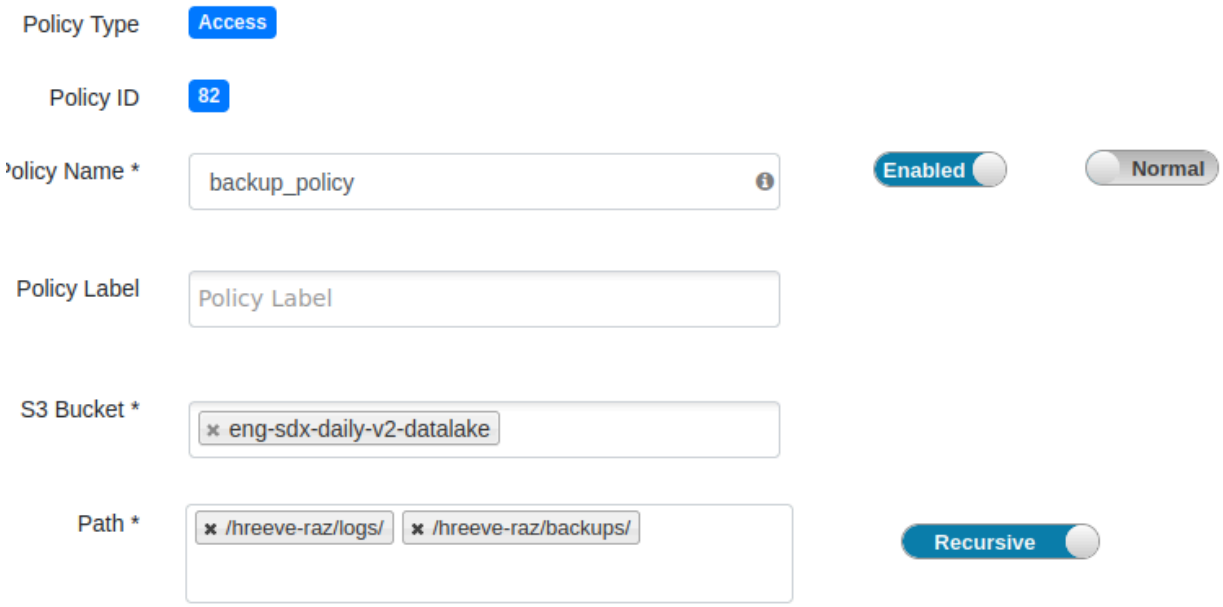

- Add read and write permissions for the atlas, hbase, hdfs, and solr users under “Allow Conditions.”

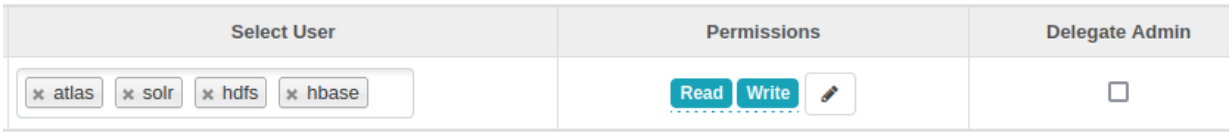

Note: If RAZ is enabled on the Data Lake, then during the backup you will see denied audit logs in Ranger related to the HBase user attempting to change ownership of the backup directory. These are expected and don't include a problem with the backup.

Configuring backups for Azure:

- Verify that the following identities have the "Storage Blob Data Contributor" role on the container where the backup is stored:
  - Data Lake Admin identity
  - Ranger Audit Logger identity

Configuring RAZ for backup

This section applies only to RAZ-enabled Azure Data Lakes. For RAZ-enabled AWS Data Lakes, see the section above.

Add a RAZ policy to allow the backups to be written to and read from the backup location.

- Open the Ranger UI.
- Go to the cm_adls policy list.

- Add a new policy:
  - Policy name: backup_policy
  - Storage Account: The storage account where backups will be stored
  - Storage Account Container: The container where backups will be stored
  - Path: The path(s) in the bucket where backup will be written

    Note: If more than one storage account or container will be used for backup, create a separate policy for each account/container.

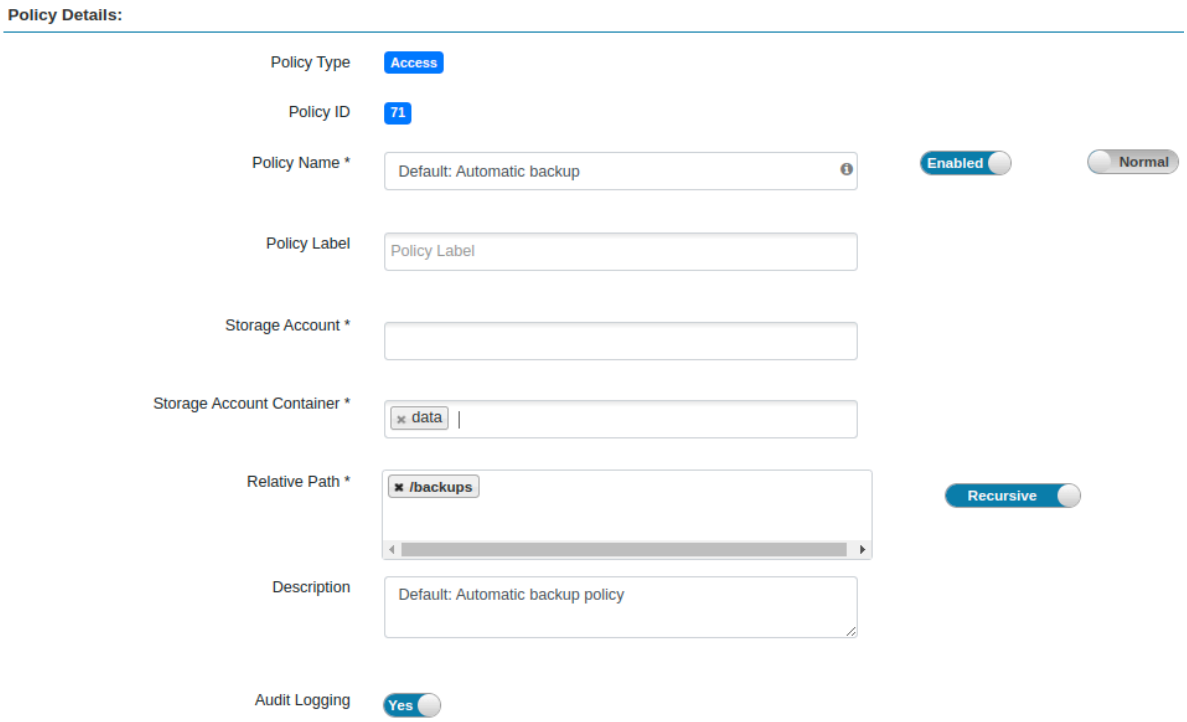

- Add read, write, list, delete, delete recursive, and move permissions for the atlas, hbase, hdfs, and solr users under “Allow Conditions."

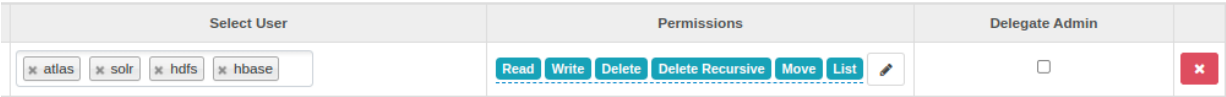

Note: If RAZ is enabled on the Data Lake, then during the backup you will see denied audit logs in Ranger related to the HBase user attempting to change ownership of the backup directory. These are expected and don't include a problem with the backup.

Configuring backups for GCP:

Verify that the Ranger Audit Service account has the following required permissions:

- resourcemanager.projects.get
- resourcemanager.projects.list
- storage.buckets.get
- storage.objects.create
- storage.objects.delete
- storage.objects.get
- storage.objects.getIamPolicy
- storage.objects.list
- storage.objects.setIamPolicy
- storage.objects.update

Note that the Ranger Audit service account permissions listed above should be granted to a custom role, not the default Storage Object Admin role.

You should also modify the scope of the Data Lake Admin and Ranger Audit service accounts to include the Backups bucket, if the bucket is different from the main data storage bucket. For more information see Minimum setup for cloud storage.

Run the backup

Procedure

1. Log into a computer that has connectivity to the Data Lake host.
2. Install the CDP CLI Client.
3. Switch to a user account that has the environment admin role.
4. Run a backup.

   Use the following command to run the Data Lake backup: $ cdp datalake backup-datalake --datalake-name < name> --backup-location <cloud storage location>

   Where the options are the following:

Option	Example	Description
--datalake-name	finance-dl	This is the name of the Data Lake as configured in the Cloudera environment. Required.
--backup-location	s3a://acme-finance-admin-bucket/backup-archiveor abfs://<container-name>@mydatalakesan.dfs.core.windows.net/backup_01/	The fully qualified name of the S3 bucket and object or ABFS location where the backup operation writes files. For S3 use the "S3a" file system protocol. Required.
--backup-name	pre-upgrade0420	An optional label that helps you identify one backup from another. The backup name can be used to identify a backup for restoring.
--close-db-connections | --no-close-db-connections		Specifies whether Ranger/HMS connections to the Data Lake should be closed or not during the backup. If you want to take the backup without workload downtime, use no-close-db-connections. Using this option means there could be changes to the Ranger/HMS data while the backup is performed. The connections are closed by default.
--skip-validation | --no-skip-validation		Using --skip-validation skips the validation that occurs before the backup process begins. This validation checks for required permissions that are often the source of backup/restore failures. See *Backup and restore for the Data Lake* for more details.
--validation-only | --no-validation-only		--validation-only runs the pre-backup and restore validation process, but does not proceed to the actual backup/restore operation. See *Backup and restore for the Data Lake* for more details.
--skip-ranger-hms-metadata | --no-skip-ranger-hms-metadata		Skips the backup of the databases backing HMS/Ranger services. If this option is not provided, the HMS/Ranger services are backed up by default.
--skip-atlas-metadata | --no-skip-atlas-metadata		Skips the backup of the Atlas metadata. If this option is not provided, the Atlas metadata is backed up by default.

Option	Example	Description
--skip-ranger-audits | --no-skip-ranger-audits		Skips the backup of the Ranger audits. If this option is not provided, Ranger audits are backed up by default.

For backups, the --skip-ranger-hms-metadata and --skip-atlas-metadata flags cannot be used at the same time.

On AWS:

```
$ cdp datalake backup-datalake --datalake-name finance-dl
     --backup-location s3a://acme-finance-admin-bucket/backup-archive
     --backup-name pre-upgrade0420
```

On Azure:

```
$ cdp datalake backup-datalake --datalake-name my-datalake
      --backup-location abfs://<container-name>@mydatalakesan.dfs.core.w
indows.net/backup_01/
```

On GCP:

```
$ cdp datalake backup-datalake --datalake-name my-datalake –backup-locat
ion gs://<bucket-name>/backup
```

The output of the command shows the current status of the operation. Note the internal state shows the status of each separate backup operation. If any of the individual backups fail, the overall status is failed and the backup cannot be restored. (Line breaks added for readability.)

```
{
    "accountId": "9d74eee4-1cad-45d7-b654-7ccf9edbb73d",
    "backupId": "415927d9-9f7d-4d42-8000-71630e5938ca",
    "internalState": "{ATLAS_ENTITY_AUDIT_EVENTS_TABLE=IN_PROGRESS, EDGE_
INDEX_COLLECTION=IN_PROGRESS, DATABASE=SUCCESSFUL, FULLTEXT_INDEX_COLLEC
TION=IN_PROGRESS, ATLAS_JANUS_TABLE=IN_PROGRESS, RANGER_AUDITS_COLLECTIO
N=IN_PROGRESS, VERTEX_INDEX_COLLECTION=IN_PROGRESS}",
    "status": "IN_PROGRESS",
    "startTIme": "2021-04-20 20:10:16.567"
    "endTIme": "2021-04-20 20:10:45.341"
    "backupLocation":"s3a://acme-finance-admin-bucket/backup-archive
/backup-archive",
    "backupName": "pre-upgrade0420"
    "failureReason": ""
}
```

What to do next

To see the status of the backup after the initial command, see “Checking the status of a Data Lake backup.”

Related Information

Backup and restore for the Data Lake

Minimal setup for cloud storage (AWS)

Minimal setup for cloud storage (Azure)

Minimum setup for cloud storage (GCP)

Checking the status of a Data Lake backup

Checking the status of a Data Lake backup

After configuring and running a backup of your Data Lake, you can check the status of the backup operation.

Checking backup status

Use the following command to see the status of a Data Lake backup:

```
$ cdp datalake backup-datalake-status
      --datalake-name <name>
     [--backup-id <generated-ID>]
     [--backup-name <name>]
     [--cli-input-json <string>]
     [--generate-cli-skeleton]
```

where the options are the following:

Option	Example	Description
--datalake-name	finance-dl	This is the name of the Data Lake as configured in the Cloudera environment.
[--backup-id]	415927d9-9f7d-4d42-8000-71630e5938ca	The system-generated ID for the backup. If you don't know the ID or name for the backup, run the list-datalake-backups command to see the available backups. If neither an ID or a name is provided, the command shows the status of the most recent backup operation.
[--backup-name]	pre-upgrade0420	The user-provided name for the backup. If you don't know the name or ID for the backup, run the list-datalake-backups command to see the available backups. If neither an ID or a name is provided, the command shows the status of the most recent backup operation.
--cli-input-json <string>		Performs service operation based on the JSON string provided. The JSON string follows the format provided by --generate-cli-skeleton. If other arguments are provided on the command line, the CLI values will override the JSON-provided values.
--generate-cli-skeleton		Prints a sample JSON configuration file to standard output. If this argument is specified, only the template is produced: the list-backup command does not run.

For example:

```
$ cdp datalake backup-datalake-status
      --datalake-name finance-dl --backup-id
      415927d9-9f7d-4d42-8000-71630e5938ca
```

The output of the command shows the current status of the backup operation. Note the internal state shows the status of each separate backup operation. If any of the individual backups fail, the overall status is failed and the backup cannot be restored.

```
{
    "accountId": "9d74eee4-1cad-45d7-b654-7ccf9edbb73d",
    "userCrn": "crn:altus:iam:us-west-1:9d74eee4-1cad-45d7-b654-7ccf9
edbb73d:user:c44ac52c-625b-410c-a46c-8db204de4d92",
    "internalState": "{ATLAS_ENTITY_AUDIT_EVENTS_TABLE=SUCCESSFUL, EDGE_IND
EX_COLLECTION=SUCCESSFUL, DATABASE=SUCCESSFUL, FULLTEXT_INDEX_COLLECTION=SUC
CESSFUL, ATLAS_JANUS_TABLE=SUCCESSFUL, RANGER_AUDITS_COLLECTION=SUCCESSFUL,
VERTEX_INDEX_COLLECTION=SUCCESSFUL}",
    "status": "SUCCESSFUL",
```

```
    "startTIme": "2021-04-20 20:10:16.567"
    "endTIme": "2021-04-20 20:32:22.012"
    "backupLocation":"s3a://acme-finance-admin-bucket/backup-archive
/backup-archive",
    "backupName": "pre-upgrade0420"
    "failureReason": ""
}
```

Listing metadata for all backups

Use the following command to show all Data Lake backups in a given storage location:

```
$ cdp datalake list-datalake-backups
      --datalake-name <name>
     [--cli-input-json <string>]
     [--generate-cli-skeleton]
```

where the options are the following:

Option	Example	Description
--datalake-name <name>	finance-dl	This is the name of the Data Lake as configured in the Cloudera environment.
--cli-input-json <string>		Performs service operation based on the JSON string provided. The JSON string follows the format provided by --generate-cli-skeleton. If other arguments are provided on the command line, the CLI values will override the JSON-provided values.
--generate-cli-skeleton		Prints a sample JSON configuration file to standard output. If this argument is specified, only the template is produced: the list-backup command does not run.

For example:

```
$ cdp datalake list-datalake-backups
          --datalake-name finance-dl
```

The output of the command shows the metadata for all backups stored for this Data Lake. Note that if a backup is listed with status failed, it cannot be restored.

```
{
    "accountId": "9d74eee4-1cad-45d7-b654-7ccf9edbb73d",
    "userCrn": "crn:altus:iam:us-west-1:9d74eee4-1cad-45d7-b654-7ccf9
edbb73d:user:c44ac52c-625b-410c-a46c-8db204de4d92",
    "backupId": "415927d9-9f7d-4d42-8000-71630e5938ca",
    "internalState": "{ATLAS_ENTITY_AUDIT_EVENTS_TABLE=SUCCESSFUL, EDGE_IND
EX_COLLECTION=SUCCESSFUL, DATABASE=SUCCESSFUL, FULLTEXT_INDEX_COLLECTION=SUC
CESSFUL, ATLAS_JANUS_TABLE=SUCCESSFUL, RANGER_AUDITS_COLLECTION=SUCCESSFUL,
VERTEX_INDEX_COLLECTION=SUCCESSFUL}",
    "status": "SUCCESSFUL",
    "startTIme": "2021-04-20 20:10:16.567"
    "endTIme": "2021-04-20 20:32:22.012"
    "backupLocation":"s3a://acme-finance-admin-bucket/backup-archive
/backup-archive",
    "backupName": "pre-upgrade0420"
    "failureReason": "null"
}
{
    "accountId": "9d74eee4-1cad-45d7-b654-7ccf9edbb73d",
```

```
    "userCrn": "crn:altus:iam:us-west-1:9d74eee4-1cad-45d7-b654-7ccf9
edbb73d:user:c44ac52c-625b-410c-a46c-8db204de4d92",
    "backupId": "6543de7d-8d22-23e4-9123-54375ec123b4",
    "internalState": "{ATLAS_ENTITY_AUDIT_EVENTS_TABLE=SUCCESSFUL, EDGE_INDE
X_COLLECTION=SUCCESSFUL, DATABASE=SUCCESSFUL, FULLTEXT_INDEX_COLLECTION=SUCC
ESSFUL, ATLAS_JANUS_TABLE=SUCCESSFUL, RANGER_AUDITS_COLLECTION=SUCCESSFUL, V
ERTEX_INDEX_COLLECTION=SUCCESSFUL}",
    "status": "SUCCESSFUL",
    "startTIme": "2021-04-19 20:09:41.341"
    "endTIme": "2021-04-19 20:28:22.822"
    "backupLocation":"s3a://acme-finance-admin-bucket/backup-archive
/backup-archive",
    "backupName": ""
    "failureReason": "null"
}...
```

Troubleshooting Data Lake backup operations

Possible issues with Data Lake backups and suggested resolutions.

"failureReason": "[Gateway Timeout]"

This probably caused by a network or process timeout issue. If this doesn't resolve itself after a few minutes, check the messages at the environment and Data Lake levels to make sure there's not some larger issue happening.

"failureReason": "[HBase service HBase does not have a running Master.]"

This happens when the HBase service is not running. Check the HBase service page in Cloudera Manager to resolve any problems and restart the service.

"failureReason": "[Unable to get user data from UMS for CRN...

```
"failureReason": "[Unable to get user data from UMS for CRN crn:altus:iam:us
-west-1:7d24tin4-1ced-47d2-v375-8ccf3ndjj71d:user:7e4e753v-8n6t-4bj6-49op-g6
0894bc063y]"
```

This error appears when the user is not authorized to start a backup. The user needs to be an environment admin role. (Go to User Management in the Cloudera Management Console)

"failureReason": "Failed to backup core=ranger_audits_[...]

```
(cdpclienv) [Wed Jun  3 16:32:27 CDT 2020 - smith@smith-7681-mbp15:/Users/sm
ith/git]$cdp datalake backup-datalake --datalake-name smith-dr-7  --backup-l
ocation hdfs://smith-dr-7-master0.smith.xcu2-8y8x.dev.cldr.work:8020/smithba
ck --backup-name "smith-test-7-6"
{
    "accountId": "7d24tin4-1ced-47d2-v375-8ccf3ndjj71d",
    "backupId": "32732sa2-1c95-4e33-a957-16d7fb645807",
    "internalState": "{ATLAS_ENTITY_AUDIT_EVENTS_TABLE=IN_PROGRESS, EDGE_IN
DEX_COLLECTION=IN_PROGRESS, DATABASE=SUCCESSFUL, FULLTEXT_INDEX_COLLECTION=I
N_PROGRESS, ATLAS_JANUS_TABLE=IN_PROGRESS, RANGER_AUDITS_COLLECTION=IN_PROGR
ESS, VERTEX_INDEX_COLLECITON=IN_PROGRESS}",
    "status": "IN_PROGRESS",
    "startTime": "2020-06-04 14:35:38.195",
    "endTime": "Thu Jun 04 14:35:47 GMT 2020",
    "backupLocation": "hdfs://smith-dr-7-master0.smith.xcu2-8y8x.dev.cldr.wo
rk:8020/smithback/",
    "failureReason": ""
```

```
}
(cdpclienv) [Thu Jun  4 09:35:47 CDT 2020 - smith@smith-7681-mbp15:/Users/
smith/git]$cdp datalake backup-datalake-status --datalake-name smith-dr-7
--backup-name "smith-test-7-6"
{
    "accountId": "7d24tin4-1ced-47d2-v375-8ccf3ndjj71d",
    "userCrn": "crn:altus:iam:us-west-1:7d24tin4-1ced-47d2-v375-8ccf3ndjj7
1d:user:7e4e753v-8n6t-4bj6-49op-g60894bc063y",
    "internalState": "{ATLAS_ENTITY_AUDIT_EVENTS_TABLE=SUCCESSFUL, EDGE_I
NDEX_COLLECTION=SUCCESSFUL, FULLTEXT_INDEX_COLLECTION=SUCCESSFUL, DATABASE=S
UCCESSFUL, RANGER_AUDITS_COLLECTION=FAILED, ATLAS_JANUS_TABLE=SUCCESSFUL, VE
RTEX_INDEX_COLLECITON=SUCCESSFUL}",
    "status": "FAILED",
    "startTime": "2020-06-04 14:35:38.195",
    "endTime": "2020-06-04 14:35:58.833",
    "backupLocation": "hdfs://smith-dr-7-master0.smith.xcu2-8y8x.dev.cldr.
work:8020/smithback/",
    "failureReason": "Failed to backup core=ranger_audits_shard1_replica_
n21 because org.apache.solr.core.SolrCoreInitializationException: SolrCore '
ranger_audits_shard1_replica_n21' is not available due to init failure: org.
apache.hadoop.ipc.RemoteException(org.apache.hadoop.ipc.RetriableException):
 NameNode still not started\n\tat org.apache.hadoop.hdfs.server.namenode.Nam
eNodeRpcServer.checkNNStartup(NameNodeRpcServer.java:2239)\n\tat org.apache.
hadoop.hdfs.server.namenode.NameNodeRpcServer.setSafeMode(NameNodeRpcServer.
java:1225)\n\tat org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolSer
verSideTranslatorPB.setSafeMode(ClientNamenodeProtocolServerSideTranslatorPB
.java:853)\n\tat org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtoco
lProtos$ClientNamenodeProtocol$2.callBlockingMethod(ClientNamenodeProtocolPr
otos.java)\n\tat org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcI
nvoker.call(ProtobufRpcEngine.java:528)\n\tat org.apache.hadoop.ipc.RPC$Serv
er.call(RPC.java:1070)\n\tat org.apache.hadoop.ipc.Server$RpcCall.run(Server
.java:984)\n\tat org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:912)\n
\tat java.security.AccessController.doPrivileged(Native Method)\n\tat javax.
security.auth.Subject.doAs(Subject.java:422)\n\tat org.apache.hadoop.securit
y.UserGroupInformation.doAs(UserGroupInformation.java:1876)\n\tat org.apache
.hadoop.ipc.Server$Handler.run(Server.java:2882)\n"
}
```

When the Data Lake is shut down and then restarted, sometimes the Solr service starts incorrectly, causing backups and restore to fail.

To resolve, restart Solr service after the Data Lake is started.

{{Failed to check the existance of s3a://eng-sdx-datalake/smith-perf-1/backup_01/. Is it valid?}} or Could not verify the existence of s3a://eng-sdx-datalake/smith-perf-1/backup_01/ -- Is it accessible?

These error messages can be caused by either a permissions issue (the Data Lake/environment does not have access to the bucket), or you have designated a bucket that does not exist.

"Solr failure: Could not find a backup repository with name ..."

If you receive this error and you are attempting a backup on Cloudera Runtime version 7.2.0 or earlier, then backup and restore operations are not supported on your current version.

Configuring and running Data Lake restore

Data Lake restore replaces the Data Lake data content: the metadata managed by each of the Data Lake services.

About this task

You may need to restore a Data Lake backup when:

- You are migrating Data Lake content to a new environment.
- A Data Lake repair fails.
- You need to delete and re-create a Data Lake.
- An upgrade fails or needs to be rolled back.

The Data Lake restore removes existing data stores and recreates them from the specified backup. This includes dropping database tables, dropping HBase tables, and deleting Solr collections.

Note: When using the Data Lake backup and restore system, you should avoid using backups from other sources: the backup operation minimizes inconsistencies among service metadata. If data is restored from other sources such as an independent database backup, the restore operation can't guarantee consistency across the Data Lake metadata.

The system checks to make sure there isn't another backup or restore in progress.

Before you begin

There is downtime when a Data Lake restore is performed, as some Data Lake services will be stopped. Additionally, access to the HMS/Ranger databases will be blocked for the duration of the restore. Do not attempt to run workloads when you are running Data Lake restore.

For AWS:

- Apply the IAM policy for Data Lake restore to the following roles:
  - DATALAKE_ADMIN_ROLE
  - RANGER_AUDIT_ROLE
  - LOG_ROLE

    For more informaiton on IAM roles, see *Minimal setup for cloud storage*.

    In the IAM policy for Data Lake restore, be sure to replace the <your-backup-bucket> variable with the backup location used.

For Azure:

- Verify that the following identities have the "Storage Blob Data Contributor" role on the container where the backup is stored:
  - Data Lake Admin identity
  - Ranger Audit Logger identity

For GCP:

Verify that the Logger Service account has the following required permissions:

- storage.buckets.get
- storage.objects.create
- storage.objects.get
- storage.objects.list

Verify that the Ranger Audit Service account has the following required permissions:

- resourcemanager.projects.get
- resourcemanager.projects.list
- storage.buckets.get
- storage.objects.get
- storage.objects.getIamPolicy

Note that the Ranger Audit service account permissions listed above should be granted to a custom role, not the default Storage Object Admin role.

Note: It is highly recommended to use the same data bucket (AWS and GCP)/storage account/container (Azure) and IAM roles/identities in the new environment where the metadata is restored.

If you want to use a different data bucket (AWS and GCP)/storage account/container (Azure) along with IAM Roles/Identities, additional steps are required:

- Make sure the IAM Roles/Identities used in the new environment have read/write permissions on the data location used in the older environment, as well as the backup location.
- If a different data bucket (AWS and GCP)/storage account/container (Azure) is used, understand that the new data will be stored in the new location provided and the existing data would still be in the older location, causing the data to be distributed in two different locations. This may not be desirable.

For Cloudera Data Warehouse:

If you are a Cloudera Data Warehouse user restoring a Data Lake to a new environment, perform the following steps from the Cloudera Data Warehouse UI before running a Data Lake restore:

1. Record details of your Database Catalogs and Virtual Warehouses, such as:
   - Number of Virtual Warehouses associated with each Database Catalog and their names
   - Configurations associated with each Virtual Warehouse and Database Catalog, especially the configurations which were customized

   These configurations are not backed up and restored during the Data Lake backup and restore process. When you activate Cloudera Data Warehouse again in the new Cloudera environment, you must re-apply the Cloudera Data Warehouse configurations.
2. Delete existing Virtual Warehouses and user-created Database Catalogs associated with your Cloudera Data Warehouse environment. The meta-data for the Virtual Warehouses associated with the default Database Catalog are preserved. The data for your tables would be preserved, as long as the cloud storage locations like S3 buckets are intact.
3. Deactivate the Cloudera Data Warehouse environment, which deletes the default Database Catalog.

After you perform the Data Lake restore, you can activate the Cloudera Data Warehouse environment from the Cloudera Data Warehouse UI and re-create any Virtual Warehouses.

Important considerations regarding Cloudera Data Warehouse metadata and data after Data Lake restore:

For Virtual warehouses associated with Default Database Catalog:

- Metadata like databases, tables and views will be restored.
- Data associated with existing tables would be visible as long as the cloud storage locations, such as objects in S3 buckets, were not deleted/modified after steps 2 and 3 above.
- Query historys and saved queries in Hue or DAS would not be visible.
- Any customizations to Virtual Warehouse or Database Catalog configurations are not retained after restore.
- The Hive/Impala Runtime version in the Virtual Warehouse and Database Catalog would be the latest inline with the corresponding Cloudera Data Warehouse version.

For Virtual warehouses associated with a non-default Database Catalog:

- No metadata or data would be restored/visible.

Internal Ranger users password reset

As part of the restore operation, the RDS database will be restored unless it is specifically excluded. Note that the passwords for internal Ranger users (admin, keyadmin, etc.) are stored in RDS. When the RDS database is restored, it will replace the passwords for all internal Ranger users with the password for that user that was saved when the backup was originally done. This does not impact users that log into Ranger via SSO.

The user accounts impacted include, but are not limited to:

- admin user
- keyadmin user
- tagsync user

- usersync user

After a restore, a user with Ranger admin access can log into the Ranger UI to update the passwords of these users if desired.

Important: Because the admin user is impacted by this behavior, at least one SSO account on the Data Lake being restored to should have Ranger administrative access, to prevent a potential loss of administrative access if for some reason the admin user's password is not known after the restore.

Procedure

1. Switch to a user account with environment admin role.
2. Restore the backup.

   Use the following command to restore a Data Lake backup:

```
$ cdp datalake restore-datalake --datalake-name <name>
```

   where the options are the following:

Option	Example	Description
--datalake-name	finance-dl	Name of the Data Lake as configured in the Cloudera environment. Required.
--backup-id	415927d9-9f7d-4d42-8000-71630e5938ca	System-generated ID for the backup. If you don't know the ID, run the list-datalake-backups command to see the IDs. If neither an ID nor backup name is provided, the command restores the most recent successful backup operation. If both an ID and a backup name are provided and don't refer to the same backup, the backup specified by the ID is used.
--backup-name		User-supplied name for the backup. If a backup ID is provided, the backup name is not needed.
--include-database | --no-include-database	TRUE	DEPRECATED. The database is included in the restore by default. To skip it, use the –skip-database flag.
--skip-ranger-hms-metadata | --no-skip-ranger-hms-metadata		Skips the restore of the databases backing HMS/Ranger services. If this option is not provided, then by default the Atlas lineage will be restored if the backup used includes the Atlas lineage information.
--skip-atlas-metadata | --no-skip-atlas-metadata		Skips the restore of the Atlas metadata. If this option is not provided, then by default the Atlas metadata will be restored if the backup used includes the Atlas metadata.
--skip-ranger-audits | --no-skip-ranger-audits		Skips the restore of the Ranger audits. If this option is not provided, then by default the Ranger audits will be restored if the backup used includes the Ranger audits.
--skip-validation | --no-skip-validation		Using --skip-validation skips the validation that occurs before the backup process begins. This validation checks for required permissions that are often the source of backup/restore failures. See *Backup and restore for the Data Lake* for more details.

Option	Example	Description
--validation-only | --no-validation-only		--validation-only runs the pre-backup and restore validation process, but does not proceed to the actual backup/restore operation. *Backup and restore for the Data Lake*
--backup-location-override	s3a://acme-finance-admin-bucket/backup-archive /backup-archive	Backup location. When provided, will be used to lookup the backup. If provided, the –backup-id parameter is required.

```
$ cdp datalake restore-datalake
--datalake-name finance-dl
--backup-id 415927d9-9f7d-4d42-8000-71630e5938ca
--backup-name <value>]
--no-skip-ranger-hms-metadata
--no-skip-atlas-metadata
--no-skip-ranger-audits
--backup-location-override s3a://acme-finance-admin-bucket/backup-archive
/backup-archive
```

The output of the command shows the current status of the operation. Note the internal state shows the status of each separate restore operation. If any of the individual restore operations fail, the overall status is failed and the restoration is stopped (not transactional). If this happens, review and correct the failure and run the restore again.

```
{
    "accountId": "8g49gju4-4has-97h7-b391-7jre9edve47n",
    "restoreId": "f0gq74h7-3b13-477a-b07c-cb74v211b81c",
    "backup-id": 415927d9-9f7d-4d42-8000-71630e5938ca",
    "internalState": "{ATLAS_ENTITY_AUDIT_EVENTS_TABLE=IN_PROGRESS, EDGE
_INDEX_COLLECTION=IN_PROGRESS, FULLTEXT_INDEX_COLLECTION=IN_PROGRESS, ED
GE_INDEX_COLLECTION_DELETE=IN_PROGRESS, VERTEX_INDEX_COLLECITON_DELETE=I
N_PROGRESS, RANGER_AUDITS_COLLECTION_DELETE=IN_PROGRESS, ATLAS_JANUS_TAB
LE=IN_PROGRESS, RANGER_AUDITS_COLLECTION=IN_PROGRESS, VERTEX_INDEX_COLLE
CITON=IN_PROGRESS, FULLTEXT_INDEX_COLLECTION_DELETE=IN_PROGRESS}",
    "status": "IN_PROGRESS",
    "startTIme": "2021-04-21 10:30:01.022"
    "endTIme": ""
    "backupLocation":"s3a://acme-finance-admin-bucket/backup-archive
/backup-archive",
    "failureReason": "null"
}
```

What to do next

To see the status of the backup after the initial command, see *Checking the status of a Data Lake backup*.

Related Information

Backup and restore for the Data Lake

Minimal setup for cloud storage (AWS)

Minimal setup for cloud storage (Azure)

Showing Data Lake restore status

How to check the status of a Data Lake restore operation.

Check status of Data Lake restore

Use the following command to see the status of a Data Lake restore:

```
$ cdp datalake restore-datalake-status
      --datalake-name <name>
     [--cli-input-json <string>]
     [--generate-cli-skeleton]
```

where the options are the following:

Option	Example	Description
--datalake-name	finance-dl	This is the name of the Data Lake as configured in the Cloudera environment.
[--restore-id]	f0da74a9-3b22-477a-b07c-cb69b211b81c	ID for a specific restore, as reported in the output of the original restore command. If a restore ID is not specified, the command returns the status of the most recent restore operation.
--cli-input-json <string>		Performs service operation based on the JSON string provided. The JSON string follows the format provided by --generate-cli-skeleton. If other arguments are provided on the command line, the CLI values will override the JSON-provided values.
--generate-cli-skeleton		Prints a sample JSON configuration file to standard output. If this argument is specified, only the template is produced: the list-backup command does not run.

For example:

```
$ cdp datalake restore-datalake-status
          --datalake-name finance-dl
```

The output of the command shows the current status of the restore operation. Note the internal state shows the status of each separate backup operation. If any of the individual restore operations fail, the overall status is failed and the restoration is aborted. Note that the internal restore status lists operations for deleting the existing Solr collections in addition to the operations to restore the backed up collections.

```
{
    "accountId": "8y63idy3-2ygn-98h6-j630-7uie9renq93e",
    "restoreId": "f0da74a9-3b22-477a-b07c-cb69b211b81c",
    "backup-id": 415927d9-9f7d-4d42-8000-71630e5938ca",
    "userCrn": "crn:altus:iam:us-west-1:8y63idy3-2ygn-98h6-j630-7uie9renq93e
:user:c87db52v-639m-613g-j94w-8hy944hn4i64",
    "internalState": "{ATLAS_ENTITY_AUDIT_EVENTS_TABLE=SUCCESSFUL, EDGE_IND
EX_COLLECTION=SUCCESSFUL, FULLTEXT_INDEX_COLLECTION=SUCCESSFUL, EDGE_INDEX_C
OLLECTION_DELETE=SUCCESSFUL, VERTEX_INDEX_COLLECITON_DELETE=SUCCESSFUL, RANG
ER_AUDITS_COLLECTION_DELETE=SUCCESSFUL, ATLAS_JANUS_TABLE=SUCCESSFUL, RANGER
_AUDITS_COLLECTION=SUCCESSFUL, VERTEX_INDEX_COLLECITON=SUCCESSFUL, FULLTEXT_
INDEX_COLLECTION_DELETE=SUCCESSFUL}",
    "status": "SUCCESSFUL",
    "startTIme": "2021-04-21 10:30:01.022"
    "endTIme": "2021-04-21 11:22:22.055"
    "backupLocation":"s3a://acme-finance-admin-bucket/backup-archive
/backup-archive",
    "backupName": "pre-upgrade0420"
    "failureReason": "null"
}
```

Restoring to a RAZ Data Lake

You can restore a Data Lake backup from a non-RAZ Data Lake to a RAZ Data Lake, or from a RAZ Data Lake to a different RAZ Data Lake.

Best Practices

For best results, it's recommended to use the same Storage Location Base, Logs Location Base, and Backup Location Base between the source Data Lake and the destination RAZ Data Lake. After the restore, the Ranger policies will be replaced with the policies from the original Data Lake. This means that if the Storage Location Base, Log Location Base, and/or Backup Location Base are different between the source Data Lake and the destination Data Lake, the restored Ranger policies will reference the locations from the source Data Lake.

- If you intend to use the locations from the source Data Lake, make sure the roles associated with the destination RAZ Data Lake have sufficient permissions to access the original storage locations.
- If you intend to use the destination RAZ Data Lake locations, the Ranger policies (cm_s3 for AWS, cm_adls for Azure) will need to be updated to reference the correct storage locations after the restore.

Preparing the Data Lakes

Add a Ranger policy to allow the backups to be read from the original backup location. For the restore operation, only write permissions are not required. This must be done on both the original source Data Lake before the backup is taken, and on the RAZ destination Data Lake before the restore is done.

AWS

On the source Data Lake:

1. Open the Ranger UI.
2. Go to the cm_s3 policy list.

3. Add a new policy:
   a. Policy name: restore_to_raz
   b. S3 bucket: The bucket where the original backups were written
   c. Path: The path in the bucket where the original backups were written

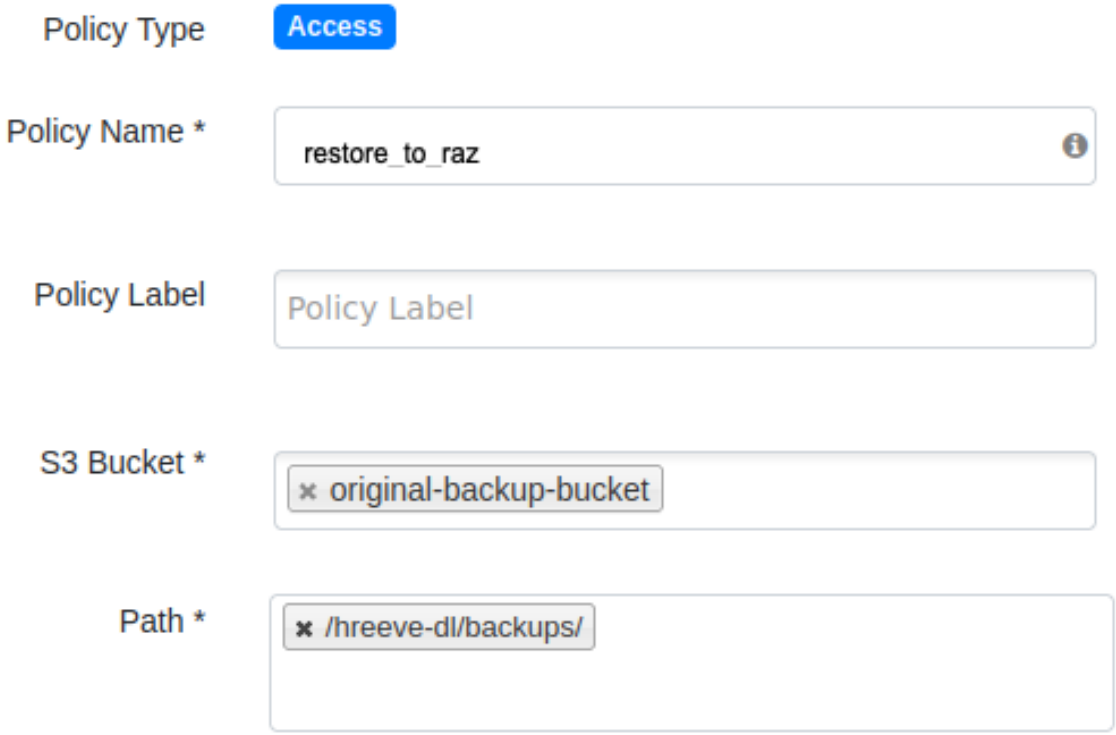

4. Add read permissions for the atlas, hbase, hdfs, and solr users under “Allow Conditions”.

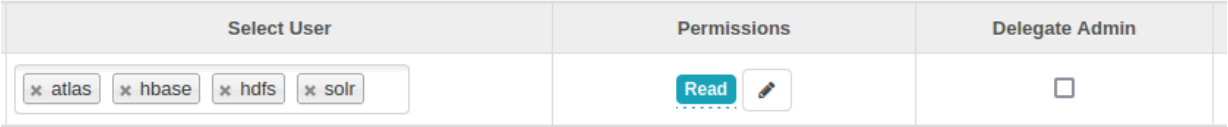

5. Repeat steps on the destination Data Lake.

Azure

Note: The source Data Lake and destination Data Lake must be configured to use the same Storage Account for the Log, Storage, and Backup Location Bases. They do not have to be configured to use the same Storage Container.

On the source Data Lake:

1. Open the Ranger UI.
2. Go to the cm_adls policy list.

3. Add a new policy:
   a. Policy name: restore_to_raz
   b. Storage Account: The storage account where backups will be stored.
   c. Storage Account Container: The container where backups will be stored.
   d. Path: The path(s) in the bucket where backup will be written.

   Note: If more than one storage container will be used for backup, create a separate policy for each container.

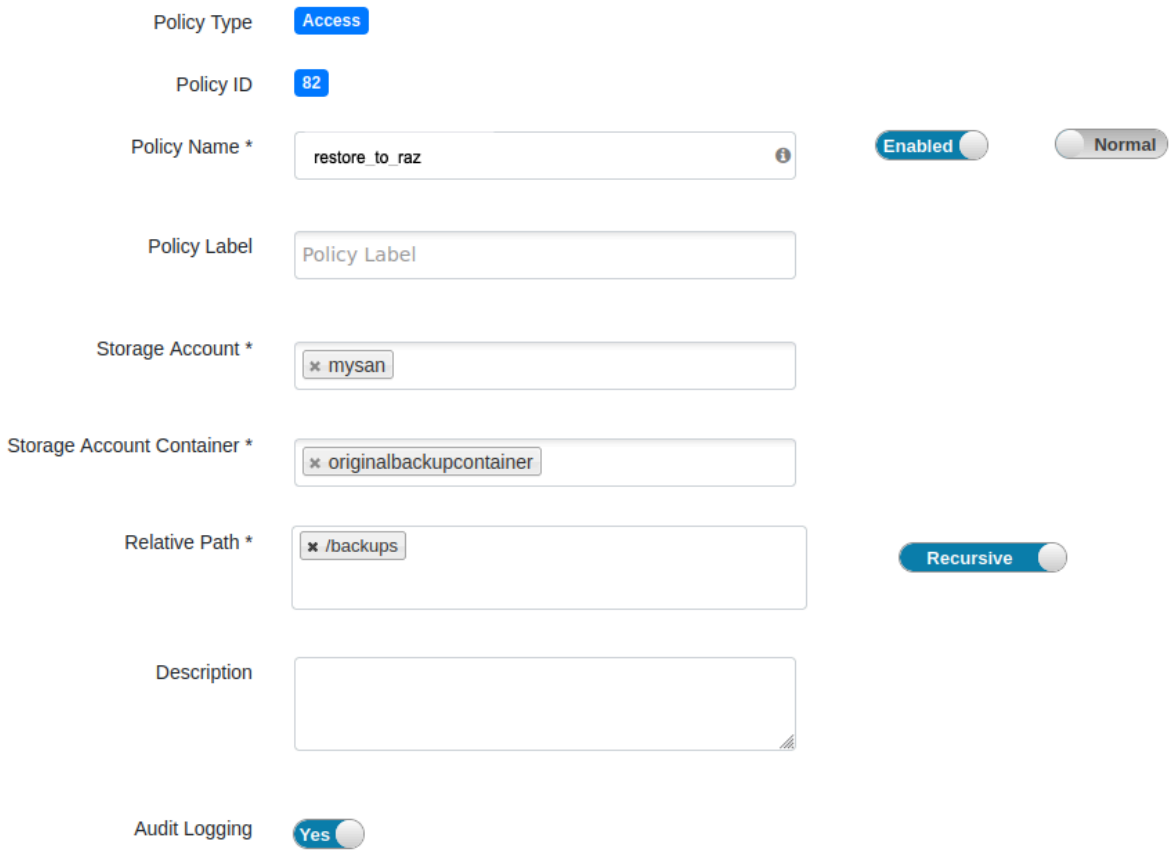

4. Add read and list permissions for the atlas, hbase, hdfs, and solr users under "Allow Conditions."

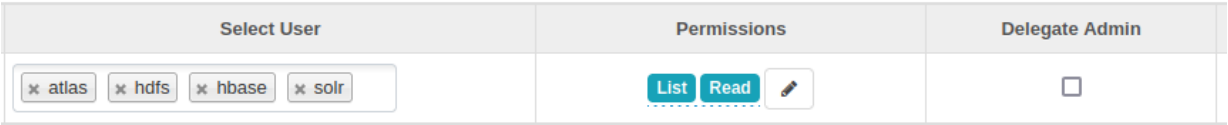

5. Repeat steps on the destination Data Lake.

Take the backup of the non-RAZ Data Lake

After you prepare the Data Lakes, see Configure backups for a Data Lake for instructions on running the backup.

Run the restore

After you take the Data Lake backup, see Restore Data Lake content for instructions on running the restore. Use the backup-id from the backup taken in the previous step.

After the restore

- Depending on the target backup and restore versions, a schema update may be required for Ranger and HMS. See Upgrade Ranger and HMS schema after Data Lake restore.

- If the Storage Location Base, Logs Location Base, and Backup Locations Base are the same between the source Data Lake and the destination Data Lake, this section can be skipped.

  After the restore, the Ranger policies will be replaced with the policies from the original Data Lake. This means that if the Storage Location Base, Log Location Base, and/or Backup Location Base are different between the original Data Lake and the RAZ Data Lake, the restored Ranger policies will reference the locations from the original Data Lake.

  If you intend to use the locations from the original Data Lake, make sure the roles associated with the RAZ Data Lake have sufficient permissions to access the original storage locations. If intending to use the RAZ Data Lake locations, the Ranger policies (cm_s3 for AWS, cm_adls for Azure) will need to be updated to reference the correct storage locations after the restore.

Upgrade Ranger and HMS schema after Data Lake restore

When Data Lake is restored using a backup that was taken from an older version of Cloudera Runtime, Ranger and HMS schema may need to be upgraded manually. This is only needed for certain Cloudera Runtime versions, as specified in Cross-version support for Data Lake backup and restore.

A Data Lake administrator should perform the below steps after the restore is complete. The steps need to be performed in Data Lake's Cloudera Manager.

Steps

1. SSH into the host where the Data Lake Cloudera Manager is running:

   ```
   ssh -i ~/.ssh/<PRIVATE_KEY_NAME>.pem cloudbreak@<HOSTNAME>
   ```

   - Replace the <PRIVATE_KEY_NAME> with the PEM key name that you provided during environment registration.
   - Replace the <HOSTNAME> with the hostname or IP address of a Data Lake node where the Cloudera Manager is running. In case of Light duty it is the Master node. In case of enterprise Data Lake Gateway 0 is the node.
2. Obtain Cloudera Manager user's DB name, user name, and password:

   ```
   sudo cat /etc/cloudera-scm-server/db.properties
   ```

   Copy the output. You will need it in later steps.
3. Connect to the backend database:

   ```
   psql -U <USERNAME> -p <PASSWORD> -h <DB_HOST> -d clouderamanager
   ```

   - Replace the <USERNAME>, <PASSWORD> and <DB_HOST> with the actual database host name obtained earlier.
4. Grant your CSSO user performing schema upgrade the "ROLE_ADMIN" Cloudera Manager role by running the below command:

   ```
   INSERT INTO user_auth_roles SELECT user_id, auth_role_id
   FROM users, auth_roles
   WHERE users.user_name='<CSSO_USER_PERFORMING_RESTORE>' AND
   auth_roles.name='ROLE_ADMIN';
   ```

   - Replace the <CSSO_USER_PERFORMING_RESTORE> with your CSSO user name.
5. Restart Cloudera Manager:

   ```
   service cloudera-scm-server restart
   ```

6. Remove the BROWSER cookies. In the Chrome browser, you can do this in: Settings # Privacy and security # Clear browsing data # Check “Cookies and other side data” # Clear data.
7. Re-login to Cloudera Manager.
8. Upgrade HMS schema by clicking on Hive Metastore # Actions # Validate Hive Metastore Schema. If the validation succeeds, move to step 9. If the validation fails, you need to first upgrade the schema by following these steps:
   a. Stop Hive Metastore.
   b. Click on Hive Metastore # Actions # Upgrade Hive Metastore Database Schema
   c. If you now run the validation again, it should pass.
   d. Start Hive Metastore.
9. Upgrade Ranger schema:
   a. Stop Ranger.
   b. Click on Ranger # Actions # Upgrade Ranger Database and apply patches
   c. Start Ranger.

Troubleshooting Data Lake restore operations

Possible issues with Data Lake restore and suggested resolutions.

Principal services running during restore

The most likely errors in restoring data from backup is that a service is in a state that is incompatible with the restore. Principal services (see Principal services on page 55) must be stopped before running the restore. Dependent services (see Dependent services on page 55) must be running to allow the restore to recreate their data. The restore checks the status of the principal services; however, if one of the dependent services is stopped and cannot be accessed to perform the restore operation, the restore operation will fail.

"failureReason": "[Datalake database restore failed.]"

If the principal services are running on the datalake during a restore operation, restore will fail with the following error message:

```
{
    "accountId": "8y63idy3-2ygn-98h6-j630-7uie9renq93e",
    "restoreId": "7c5c92c7-e3d3-408c-b18f-03bcfe0c9369",
    "backupId": "003b9882-e2fa-4fcc-ae8f-528de176c668",
    "userCrn": "crn:altus:iam:us-west-1:8y63idy3-2ygn-98h6-j630-7uie9renq93e
:user:c87db52v-639m-613g-j94w-8hy944hn4i64",
    "internalState": "{ATLAS_ENTITY_AUDIT_EVENTS_TABLE=SUCCESSFUL, EDGE_IND
EX_COLLECTION=SUCCESSFUL, DATABASE=FAILED, FULLTEXT_INDEX_COLLECTION=SUCCESS
FUL, EDGE_INDEX_COLLECTION_DELETE=SUCCESSFUL, VERTEX_INDEX_COLLECITON_DELETE
=SUCCESSFUL, RANGER_AUDITS_COLLECTION_DELETE=SUCCESSFUL, RANGER_AUDITS_COLLE
CTION=SUCCESSFUL, ATLAS_JANUS_TABLE=SUCCESSFUL, VERTEX_INDEX_COLLECITON=SUCC
ESSFUL, FULLTEXT_INDEX_COLLECTION_DELETE=SUCCESSFUL}",
    "status": "FAILED",
    "startTime": "2020-08-28 18:27:54.11",
    "endTime": "2020-08-28 18:29:55.507",
    "backupLocation": "s3a://eng-sdx-daily-datalake/smith-br-1/backup_01/",
    "failureReason": "[Datalake database restore failed.]"
}
```

To correct this scenario, stop the principal services and re-run the restore-datalake operation.

Failed restore renders Data Lake inoperable

If the restore operation fails, the Data Lake will be rendered inoperable. A restore-datalake operation must be re-run and complete successfully for the Data Lake to re-gain functionality

Related Information

Backup and restore for the Data Lake

Data Lake resizing

Data Lake resizing is the process of scaling up a light duty or medium duty Data Lake to the medium duty or enterprise form factor, which have greater resiliency than light duty and can service a larger number of clients. You can trigger the resize in the CDP UI or through the CDP CLI. As part of Data Lake resizing via CDP CLI, you can also resize from single-AZ to multi-AZ.

Overview

During a typical Data Lake scaling operation, the metadata maintained in the Data Lake services is automatically backed up, a new enterprise or medium duty Data Lake is created within the environment, and the Data Lake metadata is automatically restored to the new enterprise or medium duty Data Lake.

As part of the Data Lake resizing, you can optionally resize an existing single availability zone (single-AZ) Data Lake to a multiple availability zone (multi-AZ) Data Lake. To resize your Data Lake from single to multi-AZ, add the --multi-az flag to the Data Lake resize command.

Supportability matrix

The following table illustrates your Data Lake resizing options:

	Source		Target		Supported?
Runtime version	Scale	Deployment	Scale	Deployment	
7.2.16 and prior	Light	SingleAZ	Medium	SingleAZ	Yes
7.2.16 and prior	Light	SingleAZ	Medium	MultiAZ	Yes
7.2.16 and prior	Light	SingleAZ	Enterprise	Any	No
7.2.16 and prior	Medium	Any	Enterprise	Any	No
7.2.17+	Light	SingleAZ	Medium	Any	No
7.2.17+	Light/Medium	SingleAZ	Enterprise	SingleAZ	Yes
7.2.17+	Light/Medium	SingleAZ	Enterprise	MultiAZ	Yes
7.2.17+	Medium	MultiAZ	Enterprise	SingleAZ	No
7.2.17+	Medium	MultiAZ	Enterprise	MultiAZ	Yes

Note:

Note that:

- Resizing is only supported for Cloudera Runtime versions 7.2.7 and above, because medium duty Data Lakes are not supported for earlier versions.
- Resizing from a light or medium duty Data Lake to an enterprise Data Lake is supported only for Data Lakes on Cloudera Runtime versions 7.2.17 and above, because enterprise Data Lakes are supported only on Runtime versions 7.2.17+.
- On Runtime versions 7.2.17 and above, you can only resize to an enterprise Data Lake, because medium duty Data Lakes were deprecated in Cloudera Runtime 7.2.17.
- If you need to resize a light duty Data Lake on Cloudera Runtime 7.2.16 or earlier, it can only be resized to a medium duty Data Lake.
- There might be a requirement to vertically scale the Data Lake nodes to increase the available resources in case high resource utilization is observed after Data Lake resize to Enterprise and Operating System upgrade to RHEL from CentOS.
- When resizing from Medium Duty to Enterprise, a new Data Lake is created with new FQDNs, but the endpoint URLs remain unchanged. Make sure to update the FQDNs in your third-party applications and scripts after resizing.

Note:

RAZ-enabled Data Lakes are currently eligible for automatic restore during a resizing operation only if you are resizing:

- An AWS Data Lake on Cloudera Runtime version 7.2.15+
- An Azure Data Lake on Cloudera Runtime version 7.2.16+

For older Runtime versions, the Data Lake will be automatically backed up, but must be manually restored after the resizing is complete. If RAZ is in use on a Cloudera Runtime version that is ineligible for automatic restore, before you start the Data Lake backup, make sure that the appropriate Ranger policy exists with access to the backup location in the cloud. See instructions for manually restoring a RAZ-enabled Data Lake here.

Resizing a Data Lake is supported with all data services.

Before you begin, note the following:

- The resizing operation requires an outage and should be performed during a maintenance window. No metadata changes may occur during the resizing, as these changes will no longer be present once the resizing operation completes (the previously backed up metadata is being restored). Suspend any operations that may result in any SDX metadata change during the resizing operation.
- Cloudera Data Hub clusters should be stopped before the resizing operation begins. For any cluster that cannot be stopped, stop all of the services on the Cloudera Data Hub cluster through the Cloudera Manager UI.
- With CDF 2.0 or lower, some flows must be re-created after a resizing operation.

Limitations

1. Cloudera Manager configurations are not retained when a Data Lake is resized (they are lost when a new Data Lake cluster is created as part of backup and restore operation). Therefore, prior to performing a resize you should note all the custom Cloudera Manager configurations of your Data Lake and then once the resizing operation is completed, reapply them.

2. If a Data Lake has been vertically scaled, the following limitations apply:
   - If Data Lake VM instances are vertically scaled using runbooks or via vertical scaling, they will return to default types after resizing.
   - If Data Lake storage disks are vertically scaled using runbooks or via vertical scaling, they will return to default sizes after resizing.
   - If the storage and/or image type of the remote database are resized using the cloud provider console, they will fall back to defaults after resizing.

   As a workaround, you need to resize the Data Lake via CDP CLI (instead of Cloudera user interface) and make sure to add one or more of the following to the resize command to reapply the scaling changes made during vertical scaling:

   If Data Lake VM instances were scaled:

   ```
   --custom-instance-types <VALUE>
   ```

   For example:

   ```
   --custom-instance-types core=r5.xlarge
   ```

   If Data Lake remote database was scaled:

   ```
   --custom-database-compute-storage <VALUE>
   ```

   For example:

   ```
   --custom-database-compute-storage db.m5.large=500
   ```

   If Data Lake storage disks were scaled:

   ```
   --custom-instance-disks <VALUE>
   ```

   For example:

   ```
   --custom-instance-disks core=3000
   ```

   If you do not do this, your scaling changes will be lost after the resizing.
3. If the Data Lake root disk volume was manually resized to a size larger than 200 GB (200 GB being the default value), then it goes back to 200 GB after resize. In this case you need to resize the disk size manually again after Data Lake resize.
4. If resizing from Medium Duty to Enterprise Data Lake, you must be on Cloudera Runtime 7.2.17 before attempting the resize.
5. If you would like to resize an existing single availability zone (single-AZ) Data Lake to a multiple availability zone (multi-AZ) Data Lake, the following limitations apply:
   - The AZ resizing functionality is currently available for AWS and Azure only.
   - Existing Cloudera Data Hub clusters attached to the Data Lake are not resized to multi-AZ as part of the Data Lake resizing process.
   - The single to multi-AZ resizing is only available when resizing a Data Lake via CDP CLI. The single-AZ to multi-AZ resizing is not available via the Data Lake resizing option in the Cloudera web interface.

Prerequisites

Prior to resizing the Data Lake, ensure that the following are in place:

1. The Data Lake must be running to perform the resizing operation.

2. The Data Lake and Cloudera Data Hub databases must be upgraded to PostgreSQL 14. For more information, see Upgrading Data Lake/Cloudera Data Hub database.
3. If you are using an Azure environment, you must upgrade the PostgreSQL database used by the Data Lake to Azure Flexible Server before you can perform the Data Lake resize operation. For more information, see Upgrading Azure Single Server to Flexible Server. The Single to Flexible Server upgrade happens automatically when you upgrade from PostgreSQL 11 to PostgreSQL 14. Make sure to review both the database upgrade and the Azure Single to Flexible Server upgrade documentation.
4. For RAZ-enabled Data Lakes, update the appropriate Ranger policy to give the backup and restore feature permission to access the backup location in the cloud. See instructions for configuring RAZ for backup here.
5. Make sure that Atlas is up to date and has processed all the lineage data in Kafka. To do this, follow the steps in Checking that Atlas is up-to-date. If Atlas is not up to date, lineage/audit information in Kafka that is not processed by Atlas will be lost.
6. If you are using Cloudera Data Warehouse, you must upgrade to version 1.4.1 or higher before you can resize the Data Lake. Determine the Cloudera Data Warehouse version you are on by clicking edit on the environment:

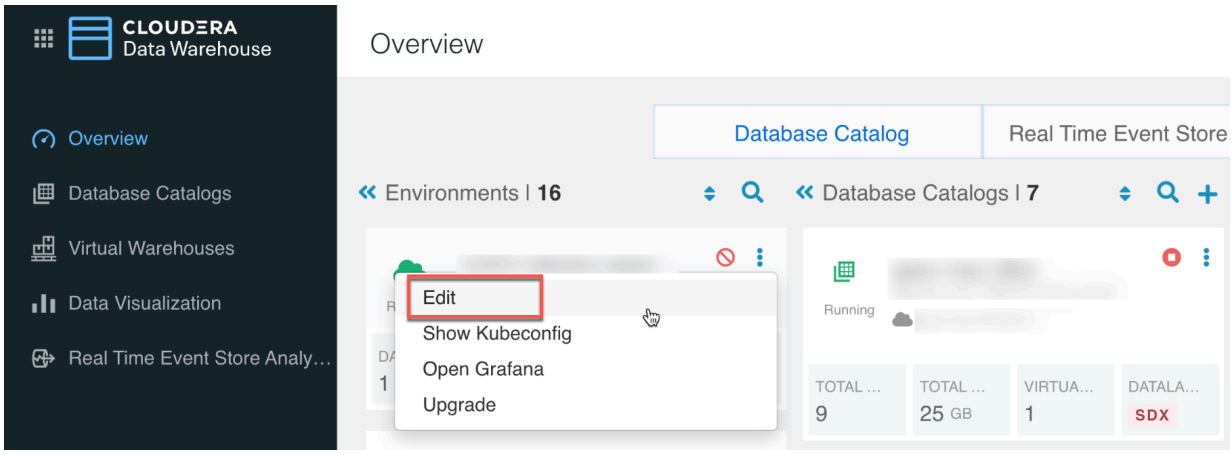

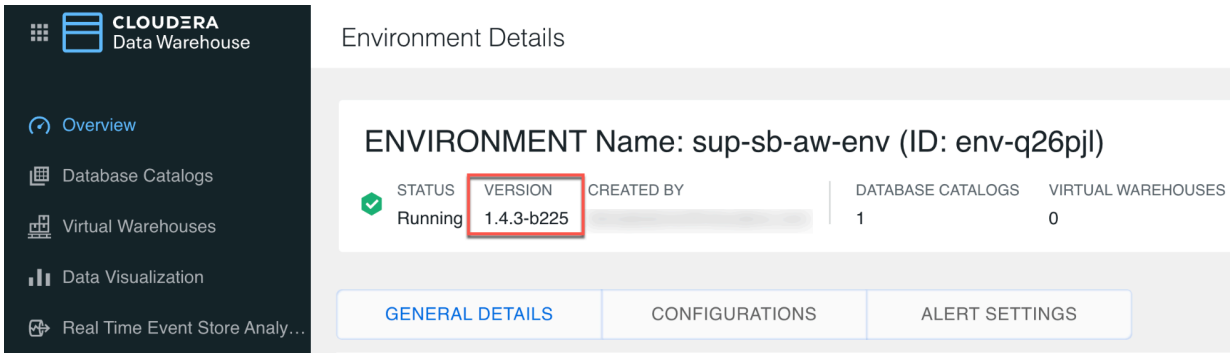

7. If you are using Cloudera Data Warehouse, stop the virtual warehouses and data catalogs associated with the environment.
8. If you are using a lower version than Cloudera Data Engineering 1.15, upgrade to Cloudera Data Engineering 1.15 and complete the following steps:
   a. Take a backup of your jobs following Backing up Cloudera Data Engineering jobs.
   b. Create a new Cloudera Data Engineering service and virtual cluster.
   c. Restore the jobs following the instructions in Restoring Cloudera Data Engineering jobs from backup.

   For CDE 1.15 and higher versions, there are no prerequisites for Data Lake resizing.
9. If you are using Cloudera AI:
   a. Backup Cloudera AI workbenches (AWS only). If backup is not supported, then proceed to the next step.
   b. Suspend Cloudera AI workbenches. If the suspend capability is not available, follow the steps in Refreshing Cloudera AI governance pods after resizing the Data Lake.

Checking that Atlas is up-to-date

Follow the steps below to ensure that Atlas is up-to-date and has processed all the lineage data in Kafka.

Procedure

1. SSH into the master node of your light duty Data Lake.
2. Switch to the super user for the node by running sudo su.
3. Copy over the following script into a file called check_atlas_updated.sh:

```
#!/usr/bin/env bash
# Determine Atlas keytab path.
ATLAS_KT=$(find / -wholename "*atlas-ATLAS_SERVER/atlas.keytab" 2>/dev/n
ull | head -n 1)

# Setup required configuration files if needed.
if [[ ! -f jaas.conf ]]; then
 ATLAS_PRINCIPAL=$(klist -kt "${ATLAS_KT}" | grep -o -m 1 "atlas\/\S*")
 printf "KafkaClient {
 \tcom.sun.security.auth.module.Krb5LoginModule required
 \tuseKeyTab=true
 \tkeyTab=\"%s\"
 \tprincipal=\"%s\";\n};\n" "${ATLAS_KT}" "${ATLAS_PRINCIPAL}" > jaas.conf
fi

if [[ ! -f client.config ]]; then
 printf "security.protocol=SASL_SSL\nsasl.kerberos.service.name=kafka\n" >
 client.config
fi
# Determine the Kafka bootstrap server to use.
KAFKA_SERVER=$(grep --line-buffered -oP "atlas.kafka.bootstrap.servers=\
K.*" \
 /etc/atlas/conf/atlas-application.properties | awk -F',' '{print $1}')

# Export Kafka-specific environment variables.
export KAFKA_HEAP_OPTS="-Xms512m -Xmx1g"
export KAFKA_OPTS="-Djava.security.auth.login.config=${PWD}/jaas.conf"

# Kinit into Atlas keytab as Atlas user.
kinit -kt "$ATLAS_KT" "atlas/$(hostname -f)" 2>/dev/null

# Obtain Atlas lineage information.
LINEAGE_INFO=$(/opt/cloudera/parcels/CDH/lib/kafka/bin/kafka-consumer-gro
ups.sh \
 --bootstrap-server "${KAFKA_SERVER}" --describe --group atlas \
 --command-config="${PWD}/client.config" 2>/dev/null \
 | awk '{print $2, $6}')

if [[ -z "$LINEAGE_INFO" ]]; then
 echo "*ERROR*: Unable to get lineage info for Atlas. Please look at the
 created configuration files to make sure they look correct."
 exit 1
fi

# Parse lineage information and determine if Atlas is out of date.
LINEAGE_LAG_VALS=($LINEAGE_INFO)
NUM_LAG_VALS=${#LINEAGE_LAG_VALS[@]}
OUT_OF_DATE_TOPICS=""
for (( i = 2; i < ${NUM_LAG_VALS}; i += 2 )); do
 if [[ ${LINEAGE_LAG_VALS[${i} + 1]} != '-' && ${LINEAGE_LAG_VALS[${i} +
1]} != '0' ]]; then
```

```
      OUT_OF_DATE_TOPICS="${OUT_OF_DATE_TOPICS}${LINEAGE_LAG_VALS[$i]}, "
 fi
done

if [[ -z "$OUT_OF_DATE_TOPICS" ]]; then
 echo "Atlas is up to date! Feel free to continue with the migration."
else
 echo "The following Atlas topics are not up to date: ${OUT_OF_DATE_TOPICS
%??}!"
 echo "Please wait until Atlas is entirely up to date before continuing
with the migration."
fi
```

4. Allow the new script to be run by running chmod +x check_atlas_updated.sh
5. Run the script with ./check_atlas_updated.sh. The script will tell you if Atlas is up to date or not. If it isn't, wait a while and check again. You should only begin the resizing process if the script tells you that Atlas is up to date.

Resizing the Data Lake through the Cloudera UI

You can resize a Data Lake from light or medium duty to medium duty or enterprise through the Cloudera UI.

About this task

Required role: EnvironmentAdmin or Owner of the environment

Before you begin

Cloudera Manager configurations are not retained when a Data Lake is resized (they are lost when a new Data Lake cluster is created as part of backup and restore operation). Therefore, prior to performing a resize you should note all the custom Cloudera Manager configurations of your Data Lake and then once the resizing operation is completed, reapply them.

Procedure

1. Stop all of the attached Cloudera Data Hub clusters that can be stopped, to make sure that there are no changes to HMS metadata during the resizing operation. For any cluster that cannot be stopped, stop all of the services on the Cloudera Data Hub cluster through the Cloudera Manager UI.
2. Verify that the DATALAKE_ADMIN_ROLE, RANGER_AUDIT_ROLE, and LOG_ROLE have read/write permissions to the backup location. See the Data Lake backup and restore documentation for more information on these permissions. LOG_ROLE is specific to Data Lake restore.
3. In the Cloudera UI, click Data Lakes and select the Data Lake that you want to resize.

4. Click Resize.

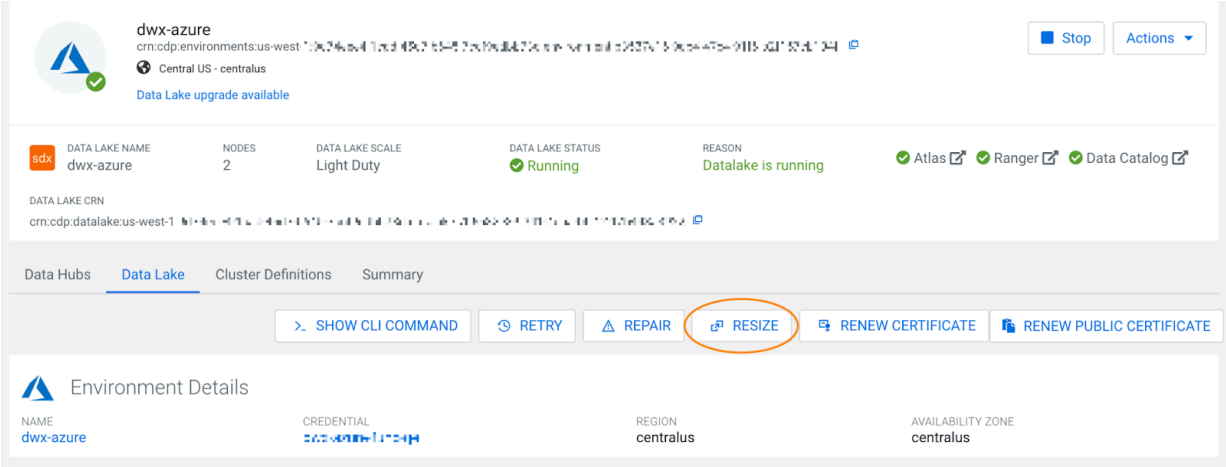

You will be asked to confirm that you want to resize the Data Lake, after which the resizing process will begin. The resizing operation is finished when the Cloudera Data Hub clusters have been automatically refreshed, which happens after the original Data Lake has been deleted. Check the Event History to verify that the Cloudera Data Hub clusters have been refreshed.

5. RAZ-enabled Data Lakes are currently eligible for automatic restore during a resizing operation only if you are resizing:

   - An AWS Data Lake on Cloudera Runtime version 7.2.15+
   - An Azure Data Lake on Cloudera Runtime version 7.2.16+

   For older Cloudera Runtime versions, the Data Lake will be automatically backed up, but you must manually restore the Data Lake after the resizing is complete. If RAZ is in use on a Cloudera Runtime version that is ineligible for automatic restore, before you start the Data Lake backup, make sure that the restore_to_raz policy Ranger policy exists with access to the backup location in the cloud. See instructions for manually restoring a RAZ-enabled Data Lake here.

Resizing the Data Lake through the CDP CLI

You can resize a Data Lake from light or medium duty to medium duty or enterprise through the CDP CLI. As part of Data Lake resizing via CDP CLI, you can also resize from single-AZ to multi-AZ.

About this task

Required role: EnvironmentAdmin or Owner of the environment

Before you begin

Cloudera Manager configurations are not retained when a Data Lake is resized (they are lost when a new Data Lake cluster is created as part of backup and restore operation). Therefore, prior to performing a resize you should note all the custom Cloudera Manager configurations of your Data Lake and then once the resizing operation is completed, reapply them.

Procedure

1. Stop all of the attached Cloudera Data Hub clusters that can be stopped, to make sure that there are no changes to HMS metadata during the resizing operation. For any cluster that cannot be stopped, stop all of the services on the Cloudera Data Hub cluster through the Cloudera Manager UI.
2. Verify that the DATALAKE_ADMIN_ROLE, RANGER_AUDIT_ROLE, and LOG_ROLE have read/write permissions to the backup location. See the Data Lake backup and restore documentation for more information on these permissions. LOG_ROLE is specific to Data Lake restore.

3. To trigger resizing from the CDP CLI, run the cdp datalake resize-datalake command. For example:

```
cdp datalake resize-datalake --datalake-name <mydatalake> --target-size
MEDIUM_DUTY_HA
```

Option	Description
–datalake-name	Name or CRN of the Data Lake that you want to upscale.
--target-size	MEDIUM_DUTY_HA or ENTERPRISE

Use the cdp datalake resize-datalake command with the --multi-az flag to resize your Data Lake from single-AZ medium duty to enterprise multi-AZ:

```
cdp datalake resize-datalake \
  --datalake-name <VALUE> \
  --target-size <VALUE> \
  --multi-az
```

If the source Data Lake is multi-AZ, the --multi-az flag is ignored.

4. Monitor the Event History. The resizing operation is finished when the Cloudera Data Hub clusters have been automatically refreshed, which happens after the original light duty Data Lake has been deleted. Check the Event History to verify that the Cloudera Data Hub clusters have been refreshed.
5. RAZ-enabled Data Lakes are currently eligible for automatic restore during a resizing operation only if you are resizing:
   - An AWS Data Lake on Cloudera Runtime version 7.2.15+
   - An Azure Data Lake on Cloudera Runtime version 7.2.16+

   For older Cloudera Runtime versions, the Data Lake will be automatically backed up, but you must manually restore the Data Lake after the resizing is complete. If RAZ is in use on a Cloudera Runtime version that is ineligible for automatic restore, before you start the Data Lake backup, make sure that the restore_to_raz policy Ranger policy exists with access to the backup location in the cloud. See instructions for manually restoring a RAZ-enabled Data Lake here.

Resizing post-requisites

Complete the following tasks after you resize a Data Lake.

Procedure

1. If RAZ is not being used, resync the IDBroker mappings to the Data Lake.
2. Reapply your custom Cloudera Manager configurations.
3. Start the Cloudera Data Warehouse Database Catalogs and Virtual Warehouses. For each virtual warehouse, Cloudera recommends that you start, stop, and start again. This will completely refresh the Data Lake details for the virtual warehouse.

   Warning: You must be aware of an issue where the Database Catalog goes into an error state after a Data Lake resize. For more information about this known issue, see the *Cloudera Data Warehouse 1.9.3-b166 (released December 5, 2024) release notes*.
4. Start the Cloudera Data Hub cluster services if you stopped them before the resizing operation. Data Hubs that were stopped before the resizing operation should continue to work when the resizing completes, by communicating with the new Data Lake automatically when they are re-started.
5. Resize up the FreeIPA cluster. See Resize FreeIPA.

6. With Cloudera DataFlow 2.1 or higher, the steps below are sufficient. For older Cloudera DataFlow versions, you must re-create the impacted flows.
   a) An alert suggesting restart of the flow is triggered in the Cloudera DataFlow service.
   b) Restart the Flows.
   c) After restart, the flows should start working with the resized Data Lake. Not all Cloudera Flows will be impacted by a resized Data Lake. Only those Flows that have a dependency on the Data Lake will be alerted.
7. Resume the Cloudera AI workbenches.

Related Information

Cloudera Data Warehouse Release Notes

Recovering after a failed resizing operation

Recover from a failed resizing operation using the recovery command in the CDP CLI.

You can recover from a failed resizing operation by returning a Data Lake to its original state before the resize operation was started. Be sure that this is what you want to do before proceeding.

Data Lake recovery simply reattaches and starts the original Data Lake in the environment. The original data lake could be light or medium duty. All of the instances, disks, and databases are unchanged from their original state.

If recovery cannot be started, or fails for any reason, reach out to the Cloudera support team, who can manually recover your Data Lake.

Trigger the recovery command through the CDP CLI:

```
cdp datalake recover-datalake --datalake-name <mydatalake>
```

Refreshing Cloudera AI governance pods

If backing up and suspending Cloudera AI workbenches is not possible, refresh Cloudera AI governance pods after resizing a Data Lake.

Procedure

1. Generate kubeconfig for Cloudera AI workbench for remote access (instructions here).
2. Configure Cloudera AI kubeconfig to access the workbench (see Kubernetes documentation).
3. Run the following commands to restart governance pods in Cloudera AI:

```
kubectl scale deployments governance --replicas 0 -n mlx
#wait for 30 sec
kubectl scale deployments governance --replicas 1 -n mlx
```

Rotating database certificates

AWS requires the rotation of the SSL/TLS certificates used for secure communication between Cloudera Public Cloud Data Lakes and certain Data Hubs and the external AWS RDS database instances that they rely on. Cloudera Public Cloud provides multiple options to perform the required RDS certificate rotation.

Cloudera Public Cloud Data Lakes and certain Data Hubs rely on Amazon Web Services (AWS) Relational Database Service (RDS) database instances that are provisioned by Cloudera during the creation of the respective resources. Similarly to other compute resources, these database instances are created in the cloud accounts of customers. The deployed Cloudera resources use a secure connection to communicate with the database using the SSL/TLS certificates issued by AWS.

SSL/TLS certificates used by Amazon RDS instances created before December 2022 are potentially using certificates that will expire in 2024. Amazon RDS instances with an expiring certificate require that a certification rotation is performed. This will replace the soon expiring certificate with a newer one. If the RDS certification rotation is not performed by 22nd August, 2024, Amazon will perform the rotation for each RDS instance and as a result, the affected Cloudera clusters will not be able to connect to the external database instances.

The actions you need to take depend on whether you have SSL enforcement enabled or disabled on the Data Lake or Cloudera Data Hub cluster as shown in the **Database** tab of the Data Lake or Cloudera Data Hub details page. This setting controls whether the database server mandates the use of an encrypted connection. It is recommended to check SSL enforcement setting on the Data Lake of your environment and separately for each Cloudera Data Hub cluster.

Cloudera strongly recommends that, when SSL enforcement is enabled, you do not perform the RDS certificate rotation directly using AWS tools, as this can lead to a service degradation or an outage of Cloudera services using the respective database. Instead, follow the instructions below so that the new certificates can be properly installed by Cloudera before changing them on the database instances. Once you have performed the required steps in Cloudera, there are no additional actions needed in the AWS console. During the database certificate rotation, Cloudera will automatically make the changes required and rotate the certificate of the attached AWS RDS database.

To determine whether you have SSL enforcement enabled on your Data Lake, perform the following steps:

1. Sign in to the Cloudera web interface.
2. Navigate to Cloudera Management Console Environments .
3. Select the environment by clicking the environment name.
4. Click the Data Lake tab.
5. Scroll to the bottom of the page and select Database.

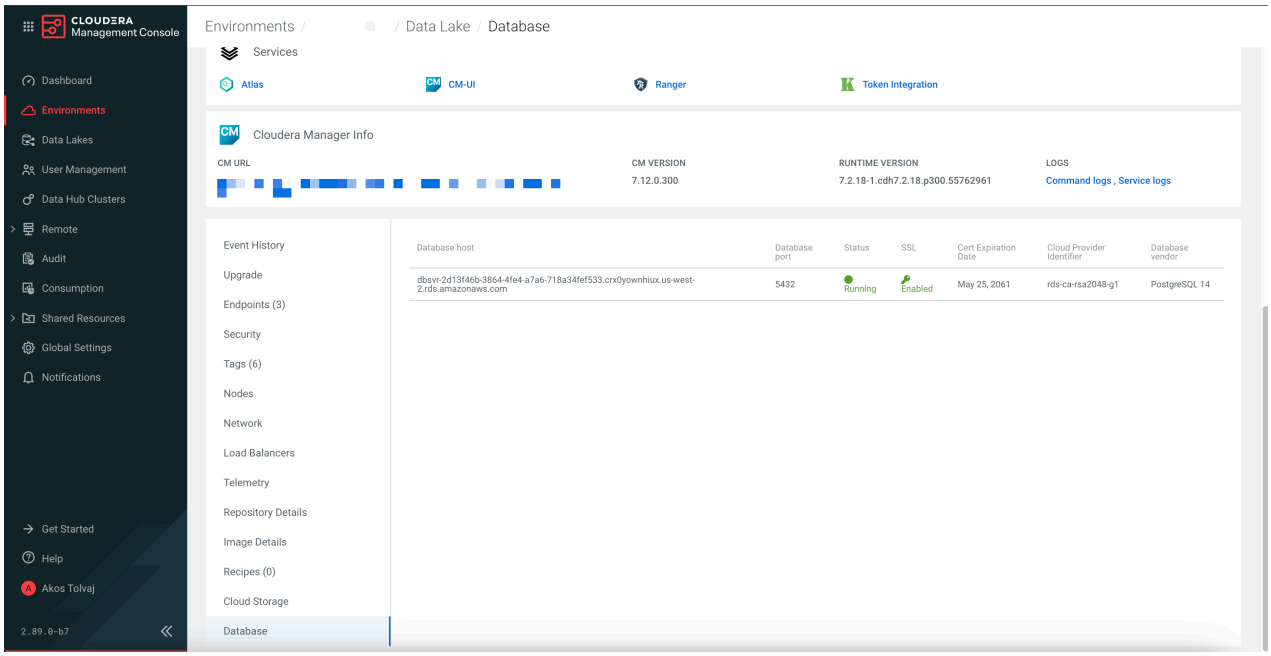

Similarly, perform the following steps on each Cloudera Data Hub cluster to determine whether you have SSL enforcement enabled:

1. Sign in to the Cloudera web interface.
2. Navigate to Cloudera Management Console Environments .
3. Select the environment by clicking the environment name.
4. Click the Data Hubs tab.
5. Select a Cloudera Data Hub.
6. Scroll to the bottom of the Cloudera Data Hub details page and select Database.

If the value for **SSL** is **Enabled**, it means SSL enforcement is enabled on your Data Lake or Cloudera Data Hub cluster, and you must perform certificate rotation as described in *Rotating database certificates when SSL enforcement is enabled*. If the value is **Disabled**, no action is required on that particular Data Lake or Cloudera Data Hub cluster. For further information, see *Rotating database certificates when SSL enforcement is disabled*.

Rotating database certificates when SSL enforcement is enabled

When SSL enforcement is enabled on your database server, do not perform the RDS certificate rotation directly using AWS tools, as this can lead to a service degradation or an outage of the respective resource. Use the options provided by Cloudera Public Cloud to perform the required RDS certificate rotation instead.

Required role (Data Lakes): EnvironmentAdmin

Required role (Cloudera Data Hub): DataHubAdmin or EnvironmentAdmin

If the following warning message is displayed on top of the Cloudera Management Console window, you must perform the TLS/SSL certificate rotation:

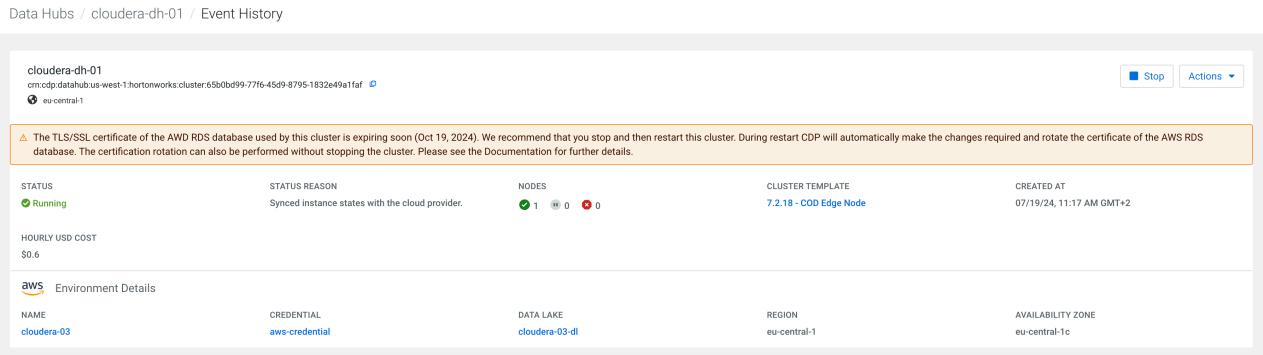

If you do not see the warning message, your clusters are not affected or the rotation has already been performed when you recently stopped and started your cluster or the environment.

Cloudera provides two options for rotating the expiring database certificates, with and without stopping your cluster:

Rotating the database certificate by stopping and starting your cluster

This is the default and most simple way to let Cloudera perform the certificate rotation and all changes required.

Instructions

1. Stop all Cloudera Data Hub clusters where the warning about expiring database certificates is shown.
2. Stop and start your Data Lakes where the warning message is displayed.

   Cloudera will automatically perform all Data Lake and AWS RDS changes required.
3. Start all Cloudera Data Hub clusters that you have previously stopped.

   Cloudera will automatically perform all Cloudera Data Hub and AWS RDS changes required where the warning message is displayed.

You can perform the stop and start either via UI or CLI, see Stop a cluster.

Note: It is recommended that you also suspend workloads running on Data Services attached to the Data Lake during this operation.

Rotating the database certificate without stopping the cluster

Stopping the Data Lake or Cloudera Data Hub clusters is not feasible in some scenarios, as workloads need to be stopped and data stored on ephemeral disks or ephemeral cache is lost during cluster restart.

In certain cases, Cloudera allows the database certificate rotation to be performed without stopping your cluster. If your cluster supports this, you will see a Rotate Database Certificate button in the warning message.

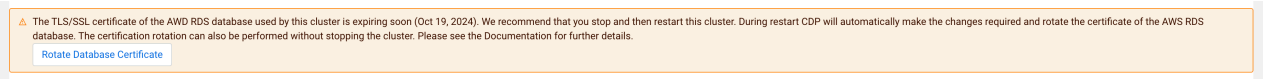

The Rotate Database Certificate button is only available if a rolling restart is supported on your cluster. The list of supported Data Lakes and Cloudera Data Hubs is available in the Rolling upgrades overview.

Important:

The certificate rotation can also be performed on Cloudera Operational Database databases by opening the cluster backing the Cloudera Operational Database database in the Data Hub screen and clicking the Rotate Database Certificate button there. However, this is only supported for Cloudera Operational Database clusters whose storage is selected as HDFS or cloud without ephemeral storage, and not supported on using Cloud Storage with Caching storage type.

Under some circumstances, a rolling certificate rotation may not be supported for a Data Lake or Cloudera Data Hub cluster, but can be enabled through entitlement. For information about obtaining this entitlement, contact Cloudera Customer Support.

Instructions

To perform a rolling restart, click the Rotate Database Certificate button on top of the screen, inside the warning message.

Note: Your cluster must be in a healthy status, otherwise the certificate rotation is not possible.

Important: You must rotate the certificates on each Cloudera Data Hub cluster first, and then proceed with rotating the certificates on the Data Lake. Otherwise, the Data Lake rotate database certificate operation will fail with a warning message.

You can also use the following CLI commands to perform the certificate rotation with minimal downtime and without stopping the cluster. The same roles are required as for the Rotate Database Certificate action through the UI.

1. Use the following command to rotate the certificate for each Cloudera Data Hub cluster.

```
cdp datahub rotate-db-certificate --datahubName <VALUE CAN BE cluster CRN
 or name>
```

2. Use the following command to rotate the certificate for the Data Lake.

```
cdp datalake rotate-db-certificate --datalakeName <VALUE CAN BE cluster
CRN or name>
```

Note: If you are performing the certificate rotation using the Rotate Database Certificate button on the UI that was made available to you through entitlement or if you are using the CLI commands, it is possible that a rolling restart will not happen because it is not supported on your cluster. In such cases, a regular restart action will be evoked. In this case, minimal downtime can be expected, however, your cluster will not be stopped.

Overriding default root certificate

Cloudera automatically chooses the target root certificate for the RDS instance. Currently, the following certificate authority (CA) certificates are available for the RDS database:

- rds-ca-rsa2048-g1, which is valid for 40 years
- rds-ca-rsa4096-g1, which is valid for 100 years
- rds-ca-ecc384-g1, which is valid for 100 years

By default, rds-ca-rsa2048-g1 is the root certificate in all supported regions that is picked as a target certificate during the certificate rotation. You have the option to override the default certificate, and set one of the certificates available in your region as a target root certificate that will be picked during the certificate rotation. Overriding the default root certificate is not tied to any particular RDS instance, but applicable to the whole AWS region in the target AWS account.

The list of available root certificates in your region can be retrieved with the following AWS CLI command:

```
aws --region [*** REGION ***] rds describe-certificates
```

The command returns the output JSON containing the properties of the RDS root certificates that are currently available in the given region. The properties also detail if an override is already set for the root certificate.

If there are more than one entries, you can use the following command to set one of the root certificates as a target, and override what is configured as default:

```
aws --region [*** REGION ***] rds modify-certificates --certificate-ident
ifier [*** ROOT CERTIFICATE ***]
```

The following command is an example to override the default rds-ca-rsa2048-g1 certificaate in the us-west-1 to rds-ca-ecc384-g1:

```
aws --region us-west-1 rds modify-certificates --certificate-identifier rds-
ca-ecc384-g1
```

Note: You can also set up the override for the default root certificate. This ensures that the root certificate will remain the same in case the default certificate is changed by AWS. You can use the following command to remove the override:

```
aws --region [*** REGION ***] rds modify-certificates --remove-custom
er-override
```

The command returns the root certificate that will be used by default.

Related Information

Describing certificates | AWS CLI

Modifying certificates | AWS CLI

Rotating database certificates when SSL enforcement is disabled

If the Data Lake for your environment or your Cloudera Data Hub cluster is using an RDS where SSL enforcement is disabled, no action is required on your side. You can simply let the root certificate expire and be replaced by AWS upon expiry.

It is recommended to check SSL enforcement setting on the Data Lake of your environment and separately for each Cloudera Data Hub cluster.

A Data Lake or a Cloudera Data Hub cluster using an RDS that is shown as SSL Disabled are essentially immune to the validity of the RDS root certificate. This is because DB connections made from Cloudera cluster services do not explicitly validate the certificate chain received from the RDS instance in such cases. The AWS RDS instance may, therefore, be left as is safely, letting its root certificate expire and be replaced automatically by AWS upon the expiry date.

Alternatively, you can instead opt to change the RDS root certificate manually using standard AWS tools like the AWS RDS Console or the AWS CLI, as described in Updating your CA certificate by modifying your DB instance or cluster in the AWS documentation.

When manually rotating the root certificate, you have the option to choose from the following available RDS certificates based on your region:

- rds-ca-rsa2048-g1, which is valid for 40 years
- rds-ca-rsa4096-g1, which is valid for 100 years
- rds-ca-ecc384-g1, which is valid for 100 years

Cloudera does not provide automation for the rotation of the RDS root certificate for databases where SSL enforcement is disabled.

Note: The Cloudera Management Console does not reflect the change of the RDS root certificate after any manual rotation.

Managing public and private certificates

There are two types of certificates within Cloudera that you must manage: public and private, also called host certificates.

- Public certificates are Let's Encrypt-issued certificates for Cloudera Data Hub and Data Lake clusters. These certificates are available on port 443 (HTTPS) of the cluster and are responsible for enabling TLS in front of Knox and other available services on that port. They are valid for 90 days, and in most circumstances Cloudera will renew these certificates automatically before they expire.

  Note the following limitations in regards to automatic renewal of public certificates:

  - Cloudera Data Hub or Data Lake clusters created on or after March 7, 2022 are eligible for automatic renewal of public certificates. Clusters created before March 7, 2022, must be renewed manually once following the instructions in *Manually renewing public certificates for Data Lake and Cloudera Data Hub clusters*. After the public certificate for a cluster has been manually renewed once from the Cloudera UI or CDP CLI, it is eligible for automatic certificate renewal in the future.
  - If an automatic renewal fails, the renewal service will retry the renewal for three consecutive days or three attempts. Any cluster that cannot be renewed by these retry attempts must be renewed manually through the Cloudera UI or CDP CLI.
  - The automatic renewal is tried three times: on the 69th, 72nd and 78th day after the certificate creation date. For example, if a certificate is getting expired on September 24th, 2022, the renewal will be tried in the following sequence:

    - First renewal: September 3rd, 2022 2:00 A.M.
    - Second renewal: September 6th, 2022 2:00 A.M.
    - Third renewal: September 12th, 2022 2:00 A.M.

    In case the renewal is successful on the first attempt, the renewal will not be tried again.
  - Renewal of the certificates happens at 2 A.M. of the Cloudera Control Plane time. If the Cloudera Control Plane is in the United States region, the renewal starts at 2 A.M. Pacific Daylight Time (PDT). If the Cloudera Control Plane is in the European region, the renewal starts at 2 A.M. Central European Summer Time (CEST). If the Cloudera Control Plane is in the East-Asian and Pacific region, the renewal starts at 2 A.M. Australian Eastern Standard Time (AEST).
  - The auto renewal service does not know the status of the cluster. If the cluster is down or performing another operation, the automatic renewal may fail and you should initiate the renewal from the UI or CLI manually. Certificate renewal will not happen if the Cloudera Data Hub and Data Lake clusters or the ClouderaPublic Cloud environment has a Stopped state.
  - If the cluster is down during the renewal attempts and comes back up after the renewal retries are exhausted, automatic renewal will not happen for that cluster. The certificate has to be renewed manually from the UI or CLI.
  - If a public certificate expires, you'll receive a warning that your connection is not secure when you attempt to access a Data Lake or Cloudera Data Hub cluster through the Cloudera UI.

  See *Manually renewing public certificates for Data Lake and Cloudera Data Hub clusters* for instructions on renewing the public certificates manually.

- Private certificates, or host certificates, are certificates created during cluster provisioning for every host with Auto-TLS. Private/host certificates have a default expiration date of one year. As private certificates get closer to expiration, the Cloudera UI displays a warning that the certificate is about to expire.

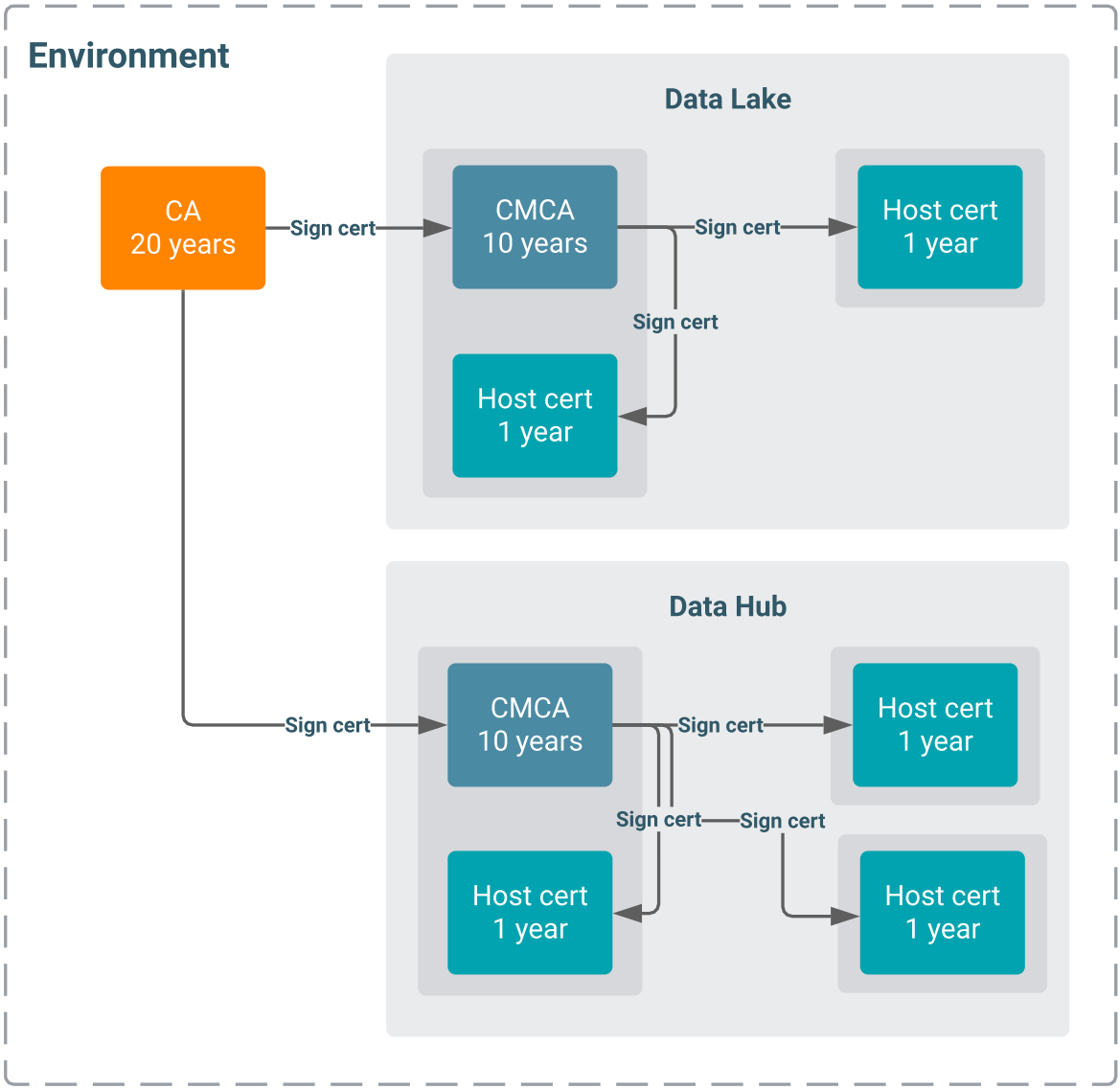

Though the Cloudera UI displays a warning about the expiration of private/host certificates, you are still responsible for renewing them through the UI or CDP CLI. After the certificates expire, the cluster is not functional, so you must renew them before expiration.

Renewing private/host certificates on Data Lake and Cloudera Data Hub clusters

Private (host) certificates have a default expiration date of one year; to keep the Data Lake and Data Hub clusters running, you must renew the host certificates before they expire.

About this task

Required role (Data Lakes): EnvironmentAdmin or Owner of the environment

Required role (Cloudera Data Hub): DatahubAdmin, Owner of the Cloudera Data Hub cluster, EnvironmentAdmin, or Owner of the environment

During cluster provisioning, Cloudera Manager creates an intermediate certificate (CMCA) signed by FreeIPA CA. The CMCA is used to create certificates for every host with Auto-TLS.

There are two ways to renew a private/host certificate. To renew the private/host certificates at any time, use the following CLI commands:

Data Lake certificate renewal:

```
cdp datalake rotate-private-certificates --datalake <Data Lake name or CRN>
```

Data Hub certificate renewal:

```
cdp datahub rotate-private-certificates --datahub <Data Hub name or CRN>
```

Alternatively, you can wait until the host certificate is close to expiration. During periodic cluster state synchronization, Cloudera uses the Cloudera Manager API to check that the HOST_AGENT_CERTIFICATE_EXPIRY apiHealthCheck alert is in a GOOD state. If the apiHealthCheck is not in a GOOD state, Cloudera displays a warning in the UI.

These UI warnings will display on the associated environments, Data Lakes, or Cloudera Data Hub clusters list and details pages. For example:

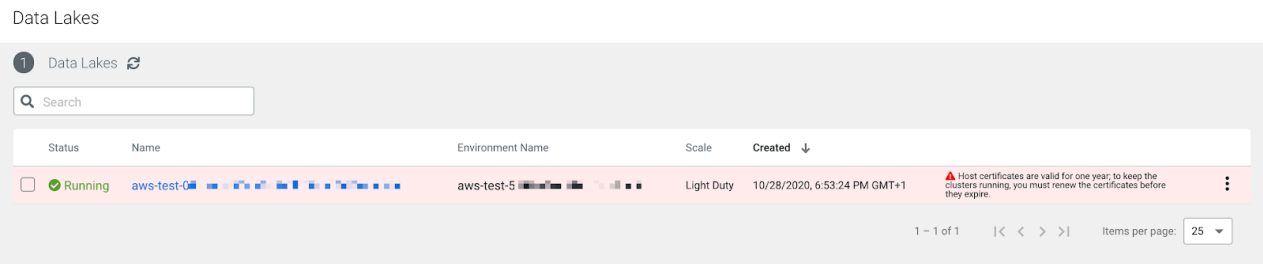

To renew the host certificate once you receive an expiration warning, follow the steps below.

Note: Renewing host certificates causes some downtime because all services are restarted. The Cloudera Manager UI will be inaccessible for some time because the Cloudera Manager server is also restarted.

Procedure

1. On the Environments, Data Lakes, or Cloudera Data Hub clusters list pages, click the three vertical dots next to the expiration message.
2. Click Renew Host Certificates or Renew Data Lake Host Certificates.

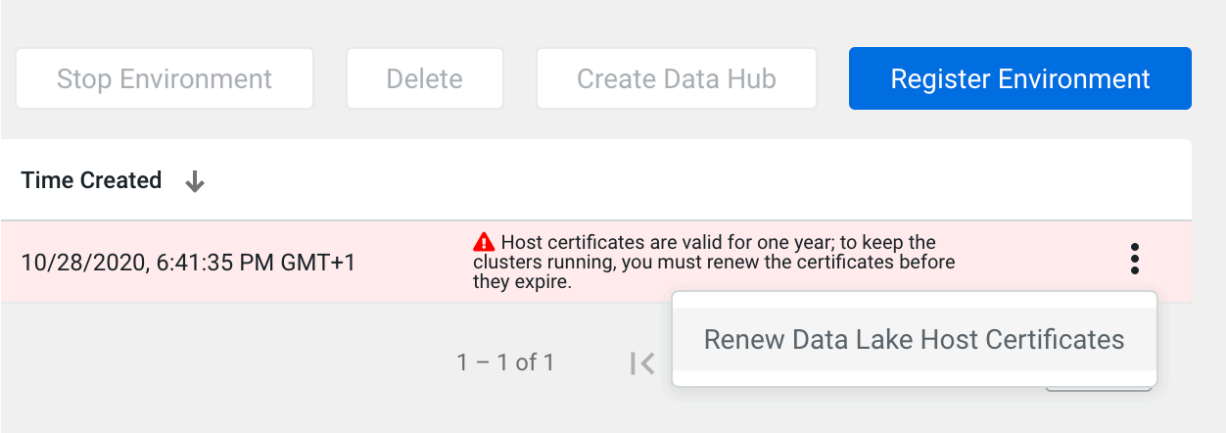

3. Click Yes when you are asked if you want to renew the certificates.

Manually renewing public certificates for Data Lake and Cloudera Data Hub clusters

Public certificates are responsible for enabling TLS in front of Knox and other available services on port 443 of Data Lake and Cloudera Data Hub clusters. Public certificates expire every 90 days and are often automatically renewed by Cloudera. If automatic renewal fails, you can renew these certificates manually.

Required role (Data Lakes): EnvironmentAdmin or Owner of the environment

Required role (Cloudera Data Hub): DatahubAdmin, Owner of the Cloudera Data Hub cluster, EnvironmentAdmin, or Owner of the environment

To renew a public certificate, click the Renew Public Certificate button on the details page of a chosen Data Lake:

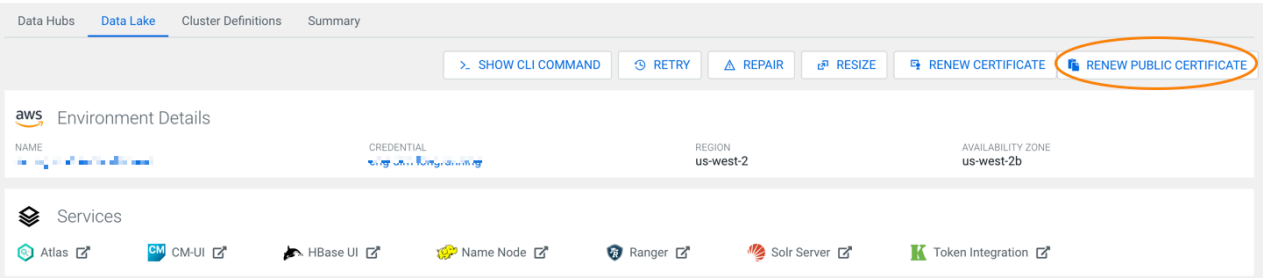

Or from the Actions menu of a Cloudera Data Hub cluster details page:

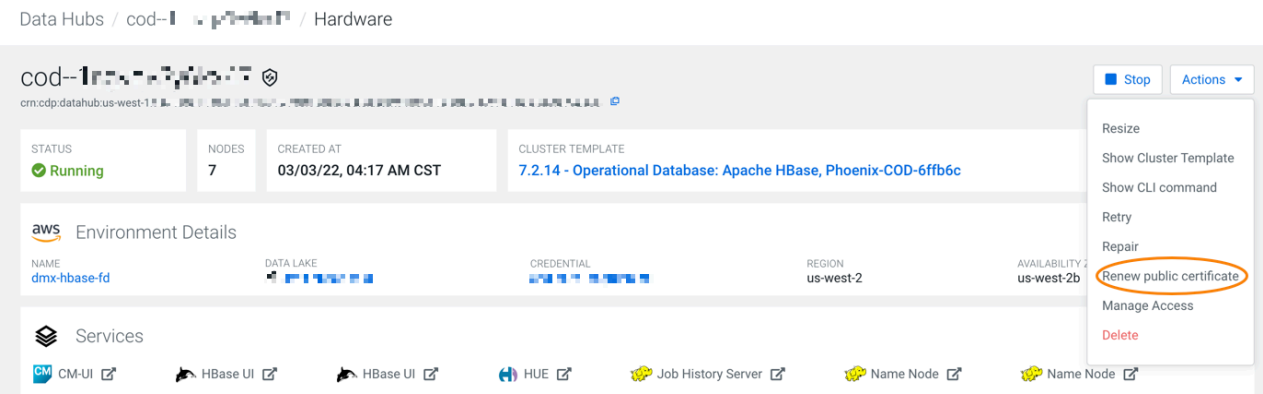

Triggering the certificate renewal may cause a minor cluster downtime of a few seconds. The entire renewal process takes a few minutes.

If you prefer to renew the certificates using the CLI, use the following commands:

Data Lake public certificate renewal:

```
cdp datalake renew-public-certificate --datalake <Data Lake name or CRN>
```

Cloudera Data Hub public certificate renewal:

```
cdp datahub renew-public-certificate --datahub <Data Hub name or CRN>
```

Recipes

A recipe is a script that runs on all nodes of a selected host group at a specific time. You can use recipes to create and run scripts that perform specific tasks on your Cloudera Data Hub, Data Lake, or FreeIPA cluster nodes.

You can use recipes for tasks such as installing additional software or performing advanced cluster configuration. For example, you can use a recipe to put a JAR file on the Hadoop classpath.

Recipes can be uploaded and managed via the Cloudera web interface or CLI and then selected, when needed, for a specific cluster and for a specific host group. If selected, they will be executed on a specific host group at a specified time.

Depending on the need, a recipe can be executed at various times. Available recipe execution times are:

- Before Cloudera Manager server start
- After Cloudera Manager server start
- After cluster installation
- Before cluster termination

Recipes are stored on the Cloudera Manager server for the lifetime of the master node, and are executed at specific times of your choosing:

- pre-service-deployment (formerly pre-cluster-manager-start): During a Cloudera Data Hub, Data Lake, or environment deployment, the script will be executed on every node before the Cloudera Manager server starts, and after node repair or OS upgrade of a cluster.
- post-cluster-manager-start: During a Cloudera Data Hub or Data Lake deployment, the script will be executed on every node after the Cloudera Manager server starts, but before cluster installation. post-cluster-manager start recipes are also executed after node repair or OS upgrade of a cluster. This option is not available for FreeIPA recipes.
- post-service-deployment (formerly post-cluster-install): The script will be executed on every node after cluster installation on the Cloudera Manager server is finished, and after node repair or OS upgrade of a cluster.
- pre-termination: The script will be executed on every node before cluster termination.

Note: On the master node, recipes are triggered when the Cloudera Manager server starts; on other nodes, recipes are triggered when the Cloudera Manager agent starts.

Writing recipes

Refer to these guidelines when creating your recipes.

When using recipes, consider the following guidelines:

- Running bash and python scripts as recipes is supported. We recommend using scripts with Shebang character sequence, for example:

```
#!/bin/sh
#!/bin/bash
#!/usr/bin/sh
#!/usr/bin/bash
#!/usr/bin/env sh
#!/usr/bin/env bash
#!/bin/sh -x
#!/usr/bin/python
#!/usr/bin/env python
```

- The scripts are executed as root. The recipe output is written to /var/log/recipes on each node on which it was executed.

- Supported parameters can be specified as variables by using mustache kind of templating with "{{{ }}}" syntax. Once specified in a recipe, these variables are dynamically replaced when the recipe is executed, generating the actual values that you provided as part of cluster creation process. For the list of parameters, refer to Recipe and cluster template parameters. For an example, see Example: Recipe using parameters.

  Note: Using variable parameters is not supported for FreeIPA recipes.

  For example, if your cluster includes an external LDAP and your recipe includes {{{ldap.connectionURL}}}, as demonstrated in the following example

  ```
  #!/bin/bash -e

  main() {
   ping {{{ ldap.connectionURL }}}
  }
  [[ "$0" == "$BASH_SOURCE" ]] && main "$@"
  ```

  then, when this recipe runs, the {{{ldap.connectionURL}}} is replaced with the actual connection URL specified as part of cluster creation process, as demonstrated in the following example:

  ```
  #!/bin/bash -e

  main() {
   ping 192.168.59.103
  }
  [[ "$0" == "$BASH_SOURCE" ]] && main "$@"
  ```

- Recipe logs can be found at /var/log/recipes/${RECIPE_TYPE}/${RECIPE_NAME}.log
- The scripts are executed on all nodes of the host groups that you select (such as “master”, “worker”, “compute”).
- In order to be executed, your script must be in a network location which is accessible from the Cloudera Management Console and the virtual network in which your cluster is located.
- Make sure to follow Linux best practices when creating your scripts. For example, don’t forget to script “Yes” auto-answers where needed.
- Do not execute yum update –y as it may update other components on the instances (such as salt) – which can create unintended or unstable behavior.

Example Python script

```
#!/usr/bin/python
print("An example of a python script")
import sys
print(sys.version_info)
```

Example bash script for yum proxy settings

```
#!/bin/bash
cat >> /etc/yum.conf
<<ENDOF
proxy=http://10.0.0.133:3128
ENDOF
```

Example recipe including variables

Original recipe:

```
#!/bin/bash -e
```

```
function setupAtlasServer() {
    curl -iv -u {{{ general.userName }}}:{{{ general.password }}} -H "X-Re
quested-By: ambari" -X POST -d '{"RequestInfo":{"command":"RESTART","context
":"Restart all components required ATLAS","operation_level":{"level":"SERVIC
E","cluster_name":"{{{ general.clusterName }}}","service_name":"ATLAS"}},"Re
quests/resource_filters":[{"hosts_predicate":"HostRoles/stale_configs=false&
HostRoles/cluster_name={{{ general.clusterName }}}"}]}' http://$(hostname -f
):8080/api/v1/clusters/{{{ general.clusterName }}}/requests
}

main() {
    setupAtlasServer
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Generated recipe (to illustrate how the variables from the original recipe were replaced during cluster creation):

```
#!/bin/bash -e

function setupAtlasServer() {
    curl -iv -u admin:admin123 -H "X-Requested-By: ambari" -X POST -d '{"R
equestInfo":{"command":"RESTART","context":"Restart all components required
ATLAS","operation_level":{"level":"SERVICE","cluster_name":"super-cluster","
service_name":"ATLAS"}},"Requests/resource_filters":[{"hosts_predicate":"Hos
tRoles/stale_configs=false&HostRoles/cluster_name=super-cluster"}]}' http://
$(hostname -f):8080/api/v1/clusters/super-cluster/requests
}

main() {
    setupAtlasServer
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Recipe and cluster template parameters

The following supported parameters can be specified as variables in recipes or cluster templates by using mustache kind of templating with "{{{ }}}" syntax.

Note: Using variable parameters is not supported for FreeIPA recipes.

The parameter keys listed below follow the following general conventions:

- { } indicates that the parameter key has multiple supported values, which are provided in this documentation. For example {fileSystemType} can be one of the following: s3, adls_gen_2, or wasb.
- [index] indicates that the parameter includes an index value for example sharedService.datalakeComponents.[index] can be "sharedService.datalakeComponents.[0]", "sharedService.datalakeComponents.[1]", and so on. There is no easy way to find out what the index will be, but you may still be able to use these parameters (for example by creating a condition to filter them).

Custom properties

Any custom property specified in the cluster template can be used as a recipe parameter. Refer to Custom properties documentation.

General

The general parameter group includes parameters related to general cluster configuration.

Parameter key	Description	Example key	Example value
general.gatewayInstanceMetadataPresented	Flag indicating if gateway instance metadata is present	general.gatewayInstanceMetadataPresented	true
general.instanceGroupsPresented	Flag indicating that instance groups are presented	general.instanceGroupsPresented	true
general.clusterName	Name of cluster	general.clusterName	testcluster
general.stackName	Name of stack	general.stackName	teststack
general.uuid	UUID of cluster	general.uuid	9aab7fdb-8940-454b-bc0a-62f04bce6519
general.executorType	Type of execution. Possible values: DEFAULT or CONTAINER	general.executorType	DEFAULT
general.clusterManagerIp	Cloudera Manager IP	general.clusterManagerIp	127.0.0.1
general.orchestratorType	Type of cluster orchestration. Possible values: HOST or CONTAINER	general.orchestratorType	HOST
general.containerExecutor	Flag indicating that the cluster is running containers	general.containerExecutor	false
general.nodeCount	Number of nodes	general.nodeCount	5
general.primaryGatewayInstanceDiscoveryFQDN	FQDN of primary gateway instance	general.primaryGatewayInstanceDiscoveryFQDN	ip-10-0-0-123.example.com
general.kafkaReplicationFactor	Number indicating the Kafka replication factor (3 or 1)	general.kafkaReplicationFactor	1

Blueprint

The blueprint parameter group includes parameters related to cluster template configuration.

Parameter key	Description	Example key	Example value
blueprint.blueprintText	Blueprint text in JSON format	blueprint.blueprintText	
blueprint.version	Version of blueprint	blueprint.version	3.2
blueprint.type	Type of blueprint	blueprint.type	HDF

Cloud storage

The fileSystemConfigs parameter group includes parameters related to cloud storage configuration.

When forming the parameter keys, the {fileSystemType} should be replaced with an actual cloud storage type such as "s3", "adls_gen_2", or "wasb".

Parameter key	Description	Example key	Example value
File system common configurations			
fileSystemConfigs.{fileSystemType}.storageContainer	Name of container in Azure storage account (Cloudbreak + stackId)	fileSystemConfigs.s3.storageContainer	cloudbreak123
fileSystemConfigs.{fileSystemType}.type	Type of filesystem	fileSystemConfigs.s3.type	S3
fileSystemConfigs.{fileSystemType}.defaultFs	Flag to indicate if the file system is the default filesystem	fileSystemConfigs.s3.defaultFs	false
fileSystemConfigs.{fileSystemType}.locations.[index].configFile	Configuration file used to configure the filesystem	fileSystemConfigs.s3.locations.[0].configFile	hbase-site

Parameter key	Description	Example key	Example value
fileSystemConfigs.{fileSystemType}.locations.[index].property	Property key of filesystem path in defined config	fileSystemConfigs.s3.locations.[0].property	hbase.rootdir
fileSystemConfigs.{fileSystemType}.locations.[index].value	Value of filesystem path in defined config	fileSystemConfigs.s3.locations.[0].value	s3a://ahorvathtestranger/testrecipe2/apps/hbase/data
Amazon S3 configurations			
fileSystemConfigs.s3.instanceProfile	ARN of related instance profile in AWS	fileSystemConfigs.s3.instanceProfile	arn:aws:iam::980678866538:instance-profile/CloudbreakRole
fileSystemConfigs.s3.storageContainer	Generated name (cloudbreak + stack id number)	fileSystemConfigs.s3.storageContainer	cloudbreak7941
fileSystemConfigs.s3.locations.[index]	JSON which contains all the value below for one hadoop component	fileSystemConfigs.s3.locations.[0]	{configFile=zeppelin-site, property=zeppelin.notebook.dir, value=s3a://storagename/clustername/zeppelin/notebook}
fileSystemConfigs.s3.locations.[index].configFile	Hadoop component configuration file	fileSystemConfigs.s3.locations.[0].configFile	zeppelin-site
fileSystemConfigs.s3.locations.[index].property	Component property name	fileSystemConfigs.s3.locations.[0].property	zeppelin.notebook.dir
fileSystemConfigs.s3.locations.[index].value	Component property value	fileSystemConfigs.s3.locations.[0].value	s3a://storagename/clustername/zeppelin/notebook
WASB configurations			
fileSystemConfigs.wasb.accountKey	Access key of the corresponding Azure storage account	fileSystemConfigs.wasb.accountKey	81a9blll-bebf-436f-a333-f67b29880f1z
fileSystemConfigs.wasb.accountName	Name of the corresponding Azure storage account	fileSystemConfigs.wasb.accountName	teststorageaccount
fileSystemConfigs.wasb.secure	Flag indicating that the file system is secure	fileSystemConfigs.wasb.secure	true
fileSystemConfigs.wasb.resourceGroupName	Name of the corresponding Azure resource group	fileSystemConfigs.wasb.resourceGroupName	testresourcegroup
fileSystemConfigs.wasb.storageContainerName	Name of container in Azure storage account	fileSystemConfigs.wasb.storageContainerName	testcontainer
ADLS Gen2 configurations			
fileSystemConfigs.adls_gen_2.accountName	Name of the corresponding Azure storage account	fileSystemConfigs.adls_gen_2.accountName	teststorageaccount
fileSystemConfigs.adls_gen_2.accountKey	Access key of the corresponding Azure storage account	fileSystemConfigs.adls_gen_2.accountKey	81k9blll-bebf-436f-a333-f67b29880f1z
fileSystemConfigs.adls_gen_2.storageContainerName	Name of container in Azure storage account	fileSystemConfigs.adls_gen_2.storageContainerName	testcontainer

External database

The rds parameter group includes parameters related to external database configuration.

When forming the parameter keys, the {rdsType} should be replaced with the actual database type such as "cloudera_manager", "beacon", "druid", "hive", "oozie", "ranger", "superset", or some other user-defined type.

Parameter key	Description	Example key	Example value
rds.{rdsType}.connectionURL	JDBC connection URL	rds.hive.connectionURL	Value is specified in the following format: jdbc:postgresql://host:port/database
rds.{rdsType}.connectionDriver	JDBC driver used for connection	rds.hive.connectionDriver	org.postgresql.Driver

Parameter key	Description	Example key	Example value
rds.{rdsType}.connectionUserName	Username used for the JDBC connection	rds.hive.connectionUserName	testuser
rds.{rdsType}.connectionPassword	Password used for the JDBC connection	rds.hive.connectionPassword	TestPssword123
rds.{rdsType}.databaseName	Target database of the JDBC connection	rds.hive.databaseName	myhivedb
rds.{rdsType}.host	Host of the JDBC connection	rds.hive.host	mydbhost
rds.{rdsType}.hostWithPortWithJdbc	Host of JDBC connection with port and JDBC prefix	rds.hive.hostWithPortWithJdbc	Value is specified in the following format: jdbc:postgresql://host:port
rds.{rdsType}.subprotocol	Sub-protocol from the JDBC URL	rds.hive.subprotocol	postgresql
rds.{rdsType}.connectionString	URL of JDBC the connection. In case of Ranger, this does not contain the port	rds.hive.connectionString	Value is specified in the following format: jdbc:postgresql://host:port/database
rds.{rdsType}.databaseVendor	Database vendor	rds.hive.databaseVendor	POSTGRES
rds.{rdsType}.withoutJDBCPrefix	URL of the JDBC connection without JDBC prefix	rds.hive.withoutJDBCPrefix	Value is specified in the following format: host:port/database

Gateway

The gateway parameter group includes parameters related to Knox gateway configuration.

Parameter key	Description	Example key	Example value
gateway.gatewayType	Type of gateway. Possible values: CENTRAL/INDIVIDUAL	gateway.gatewayType	CENTRAL
gateway.path	Base path of gateway (typically this is the name of the cluster)	gateway.path	test
gateway.ssoType	Type of SSO. Possible values: SSO_PROVIDER/NONE	gateway.ssoType	SSO_PROVIDER
gateway.ssoConfigured	Flag indicating if SSO is provided	gateway.ssoConfigured	true
gateway.ssoProvider	Path to the SSO provider	gateway.ssoProvider	/test/sso/api/v1/websso
gateway.signKey	Base64 encoded signing key	gateway.signKey	
gateway.signPub	Signing certificate (x509 format)	gateway.signPub	
gateway.signCert	Public SSH key used for signing (standard public key format)	gateway.signCert	

Shared services

The sharedService parameter group includes parameters related to Data Lake configuration.

Parameter key	Description	Example key	Example value
sharedService.rangerAdminPassword	Admin password of the Ranger component	sharedService.rangerAdminPassword	Admin1234!
sharedService.attachedCluster	Flag indicating that the cluster is attached to a data lake cluster	sharedService.attachedCluster	true
sharedService.datalakeCluster	Flag indicating that the cluster is a data lake cluster	sharedService.datalakeCluster	true
sharedService.rangerAdminPort	Admin port of the Ranger component	sharedService.rangerAdminPort	6080
sharedService.datalakeClusterManagerIp	Cloudera Manager IP of data lake cluster	sharedService.datalakeClusterManagerIp	127.0.0.1

Parameter key	Description	Example key	Example value
sharedService.datalakeClusterManagerFqdn	Cloudera Manager FQDN of data lake cluster (or the IP if FQDN cannot be found)	sharedService.datalakeClusterManagerFqdn	ip-10-88-28.example.com
sharedService.datalakeComponents.[index]	Data lake component list	sharedService.datalakeComponents.[0]	METRICS_COLLECTOR

Example: Recipe with parameters

If you pass the supported parameters in a recipe, their values are dynamically fetched and replaced.

Note: Using variable parameters is not supported for FreeIPA recipes.

Example recipe template (the {{{general.clusterName}}} is included as a template):

```
#!/bin/bash -e

function setupDefaultClusterFolder() {
  mkdir -p /var/log/{{{general.clusterName}}}
}

main() {
    setupDefaultClusterFolder
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Example recipe after {{{general.clusterName}}} is set to my-super-cluster based on the actual cluster name:

```
#!/bin/bash -e

function setupDefaultClusterFolder() {
  mkdir -p /var/log/my-super-cluster
}

main() {
    setupDefaultClusterFolder
}
[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Register a recipe

In order to use your recipe for clusters, you must first register it with the Cloudera Management Console.

About this task

Required role: EnvironmentCreator can create a shared resource and then assign users to it.

SharedResourceUser or Owner of the shared resource can use the resource.

Before you begin

If you are using Cloudera with a proxy, note that the Cloudera proxy settings do not apply to cluster recipes. If you planning to use the recipes, then you can set the proxy settings manually. You can find the proxy settings in the /etc/cdp/proxy.env file.

Procedure

1. Place your script in a network location accessible from Management Console and from the virtual network in which your clusters are located.
2. Log in to the Cloudera web interface.
3. Navigate to Shared ResourcesRecipes and click Register Recipe.
4. Provide the following:

Parameter	Value
Name	Enter a name for your recipe.
Description	(Optional) Enter a description for your recipe.
Execution Type	Select one of the following options: • pre-service-deployment (formerly pre-cluster-manager-start): During a Cloudera Data Hub, Data Lake, or environment deployment, the script will be executed on every node (in the host group where you assigned the recipe) before the Cloudera Manager server starts. • post-cluster-manager-start: During a Cloudera Data Hub or Data Lake deployment, the script will be executed on every node (in the host group where you assigned the recipe) after the Cloudera Manager server starts, but before cluster installation. This option is not available for FreeIPA recipes. • post-service-deployment (formerly post-cluster-install):: The script will be executed on every node (in the host group where you assigned the recipe) after cluster installation on the Cloudera Manager server is finished. • pre-termination: The script will be executed on every node (in the host group where you assigned the recipe) before cluster termination.
Script	Select one of: • File: Point to a file on your machine that contains the recipe. • Text: Paste the script.

5. Click Register.

What to do next

- When you create a Cloudera Data Hub cluster, you can select a previously added recipe on the advanced Cluster Extensions page of the create cluster wizard.
- When you create an environment, you can select a previously added recipe on the Data Access and Data Lake Scaling page of the environment creation wizard, under Advanced Options > Cluster Extensions > Recipes.
- When you create an environment, you can select a previously added FreeIPA recipe on the **Region, Networking, and Security** page of the environment creation wizard, under Advanced OptionsCluster ExtensionsRecipes.
- You can also attach recipes to Cloudera Data Hub or Data Lake clusters when you create an environment/Data Lake or Cloudera Data Hub cluster through the CDP CLI.

Update a recipe

You can attach or detach recipes to/from existing Data Lake clusters in an available state. Using this capability, you can update a recipe by removing it from the cluster, replacing the old recipe with a modified recipe of the same type, and attaching the new modified recipe to the cluster.

About this task

Attaching or detaching a recipe will not execute the recipe. The next execution of the recipe will take place based on the type of the recipe. After an upscale, a newly attached recipe runs only on the new hosts.

Required role (one of the following):

- PowerUser on Cloudera tenant
- Owner of the environment
- EnvironmentAdmin

Procedure

1. Create a new recipe (with updated/modified content) of the same type as the old recipe that you want to replace.
2. Click Data Lakes<Data Lake Name> and scroll to the Data Lake details pane at the bottom of the page. Click the Recipes tab and find the recipe that you want to remove in the list of recipes for the Data Lake.
3. Click Remove Recipe next to the name of the recipe that you want to remove, then click Yes in the confirmation window.
4. Once you have removed the old recipe, click on the Add Recipe button for the Data Lake and select the same host group that you previously used for the old recipe. Then select the name of the new recipe that contains the modified content and click Add.

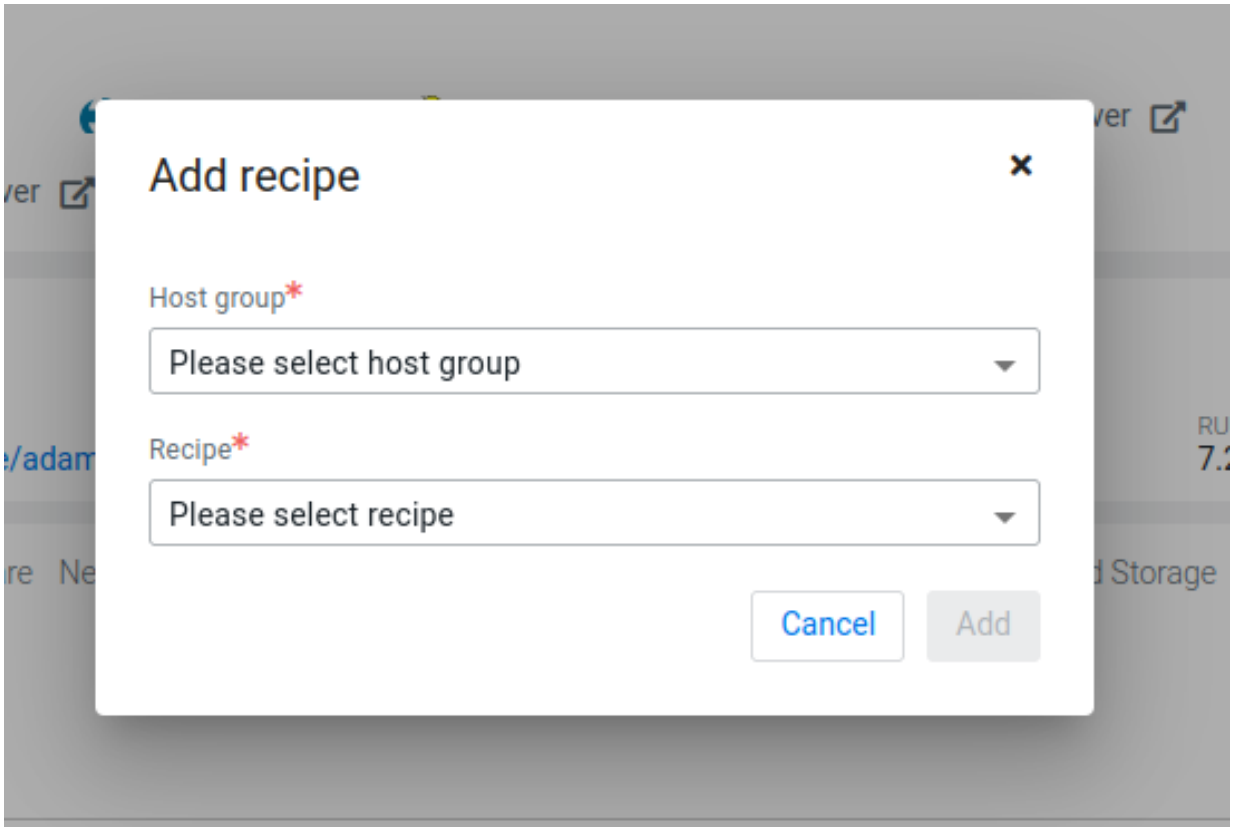

5. Alternatively, you can use the CDP CLI to attach or detach recipes from a host group:

```
cdp datalake replace-recipes --datalake <CRN or name> --instance-group-r
ecipes instanceGroupName=<instance group name>,recipeNames=<recipe names>
```

Note that the same command is used to both detach and attach a recipe. When you attach a recipe, use the recipeNames parameter to specify the recipe or recipes that you want to attach; when you detach a recipe, give the instanceGroupName but do not provide a recipeName. For example:

To detach all recipes from an instance group:

```
cdp datalake replace-recipes --datalake myDL --instance-group-recipes
```

```
            instanceGroupName=worker,recipeNames=
```

To attach a new recipe:

```
cdp datalake replace-recipes --datalake myDL --instance-group-recipes
            instanceGroupName=worker,recipeNames=myrecipe
```

For instance groups with multiple recipes, give the recipeNames that you would like to keep. Any recipes not specified will be detached.

Results

You should see the new recipe appear for the same host group. After this change, the next recipe execution will execute the new script.

Managing recipes from CLI

You can manage recipes from CLI using cdp datahub commands.

Required role: EnvironmentCreator can create a shared resource and then assign users to it.

SharedResourceUser or Owner of the shared resource can use the resource. The Owner of the shared resource can delete it.

Note: Currently, recipes use cdp datahub commands regardless of whether the recipe is intended to run on Data Hub, Data Lake, or FreeIPA cluster nodes.

- Register a new recipe: cdp datahub create-recipe --recipe-name <value> --recipe-content <value> --type <value>

  Supported types:

  - PRE_SERVICE_DEPLOYMENT (formerly PRE_CLOUDERA_MANAGER_START)
  - POST_CLOUDERA_MANAGER_START (this option is not available for FreeIPA recipes)
  - POST_SERVICE_DEPLOYMENT (formerly POST_CLUSTER_INSTALL)
  - PRE_TERMINATION
- List all available recipes: cdp datahub list-recipes
- Describe a specific recipe: cdp datahub describe-recipe --recipe-name <value>
- Delete one or more existing recipes: cdp datahub delete-recipes --recipe-name <value>

Cloudera Management Console

Data Protection

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ensuring data recovery and preventing data loss in AWS..... 4
Monitoring activities and data.....4
Backing up versions and objects.....4
Restoring versions and objects.....6

Ensuring data recovery and preventing data loss in AWS

Different recovery and preventive methods can be used to ensure recovery in case of accidental deletes and corruption of cloud storage data. Monitoring activities and the state of data can help determine what caused the data loss, and with the available backup and restore tools you can ensure the recovery of data stored in S3.

Monitoring activities and data

You can use Cloudera auditing, CloudTrail and S3 Storage Lens to monitor activities and events in the AWS S3 bucket.

Cloudera auditing

Server and audit logs can give a more detailed view of requests made to the AWS S3 bucket through Cloudera. The audit archiving can collect or log evidence of activities in a system. The audit and server logs are enabled by default from Cloudera Runtime 7.2.15. For more information, see the *Enable AWS S3 Access logs for server side logging* and the *Auditing documentation* in Cloudera Public Cloud.

CloudTrail

CloudTrail is integrated with Amazon S3, providing a record of actions performed by a user, role or AWS service. The events are captured as a subset of API calls. CloudTrail can be enabled to log the events happening on the S3 data. For more information, see the *Logging Amazon S3 API calls using AWS CloudTrail* documentation.

S3 Storage Lens

S3 Storage Lens provides an overview of the data stored in S3 and activity trends. S3 Storage Lens can be used to have a brief summary of what the state of the data is in S3 for various buckets. For more information, see the *S3 Storage Lens* introduction.

Related Information

Enable AWS S3 Access logs for server side logging

Auditing documentation

Logging Amazon S3 API calls using AWS CloudTrail

S3 Storage Lens

Object Store Auditing

Backing up versions and objects

You can use S3 versioning and AWS Backup for S3 to create backup of data stored in S3.

S3 versioning

S3 versioning can keep multiple versions of the objects stored in the S3 bucket. Using the versioning, it is easier to recover the data that was deleted by accident or due to application failure. For more information, see the *Using versioning in S3 buckets* documentation.

As versioning is not enabled by default, Cloudera recommends enabling the S3 versioning to ensure the recovery of lost data using the *Enabling versioning on buckets* and *Working with versioned S3 buckets* documentation.

You can also find how to retrieve object versions from a bucket when versioning is enabled in the *Amazon S3* documentation. LifeCycle Rules can be used with versioning to manage the lifecycle of the objects in an S3 bucket. For more information, see the *Managing your storage lifecycle* documentation.

Ease of recovery

The version ID of the object is required when recovering an object to its previous version. In case the whole bucket or more than one object needs to be recovered, there must be a list of all the versions for each object and the recovery is based on the timestamp of the objects.

Recovery scenarios

If there is a request from a job to remove a certain file/folder, instead of actually removing the object, the current file must have a ‘delete marker’. This ‘delete marker’ serves as a new version on top of the current version of the object, and the ‘delete marker’ is be deleted. In case of a job failure or accidental deletion, it is hard to decide if an object is deleted by accident or it was a ‘delete marker’ file. There is no way to reset all of the objects to a particular time before a job was run. In case someone deletes an older version permanently or a retention policy removes an older version, there is no way to recover the version.

Security and encryption

The same IAM roles are applied for the versioning as for the S3 bucket. Versions are available based on the access of the user to the bucket data. Versioned data also follows the same encryption mechanism as the S3 stored data, such as SSE, SSE-KMS, SSE-c, and so on.

Object level recovery

Versioning happens on an object level: versioning a bucket applies the versioning to all the objects, and recovery with versioning is per object level.

Backup scheduling and location

Lifecycle policies can be configured to manage the scheduling of backups. Only one version is considered as the current version, a new current version can be created by modifying the object. The versions are stored in the same bucket as the objects. These versions can be moved across different buckets. Versioning can be enabled for all regions.

AWS Backup for S3

AWS Backup offers backup management, policy-based and tag-based backup, lifecycle management policies, cross-region and cross-account backup features, and can be used with S3 versioning. When using AWS Backup, Cloudera recommends setting a lifecycle expiration period for the S3 versions as all unexpired versions of the S3 data will be backed up and stored, which can increase cloud cost. For more information about the AWS Backup features and availability, see the *What is AWS Backup* documentation.

Ease of recovery

Using AWS Backup, you can choose from a list of recovery points that indicate the state of S3 data at that point in time. The whole bucket can be restored or up to 5 prefixes to the recovery point’s data. Data can be restored in the source bucket, another existing bucket or in a newly created bucket, but the bucket must be in the same region as the backup vault.

Recovery scenarios

If a certain file or folder is removed due to a job request, the bucket can be recovered to a point of time before the job started or if the prefix (or path folder) is identified, the file or folder can be restored from that previous recovery point.

If someone permanently deletes some versions and objects from the bucket, the previous recovery point can be used to recover the data. Separate IAM roles for backup and S3 can ensure separation in access to both data.

If someone deletes the recovery points, there is no way to recover that recovery point as the backup is incremental. Vault lock can be used to prevent the deletion of recovery points.

Security and encryption

Different IAM roles defined for backup vault and the S3 bucket can ensure different access permissions. A KMS key is required to encrypt all backed up data, which can be a KMS key you created or a default AWS one.

Object level recovery

AWS Backup allows up to 5 prefixes to recover objects or the whole bucket data is recovered.

AWS Backup creates a backup of all your S3 versions, but restores only the latest version from the version stack at any point in time. This AWS Backup limitation can be harmful when recovering from corrupted data. In this case, the corrupted version needs to be the 'delete marker' and restoring from the previous version, or permanently deleting the corrupted version and using the previous one as the current version.

Backup scheduling and location

Creating a backup plan allows you to schedule and prepare for backup when needed in the defined interval. However, on-demand backup can also be created beside the scheduled backup. A bucket can only be backed up in the supported regions of AWS backup. Restoring can be done to the same region bucket as well. AWS Backup is supported for Amazon S3 in all regions except China (Beijing), China (Ningxia), Europe (Spain), Europe (Zurich), Asia Pacific (Hyderabad), Asia Pacific (Melbourne) Region.

Comparison of S3 versioning and AWS Backup

Both S3 versioning and AWS Backup are easy to implement and use for backing up the S3 bucket and with the help of IAM roles a more sufficient security level can be configured for both tools. However, when using S3 versioning, there is no straight way to provide the whole bucket's previous state at a particular time, and an external script is required to restore prefixes/directories to a specific time. For AWS Backup, if a current version (non-delete marker) is present and a previous version needs to be restored, the restore job prefers the current version, so it could mean that in cases of data corruption we fall back to versioning as our solution to restore a particular version. You also need to enable the S3 versioning to use AWS Backup.

The S3 versioning and AWS Backup have different pricing based on usage. For more information, see the *AWS S3 Pricing* page.

Related Information

Using versioning in S3 buckets
Enabling versioning on buckets
Working with versioned S3 buckets
Amazon S3
Managing your storage lifecycle
What is AWS Backup
AWS S3 Pricing

Restoring versions and objects

You can use Point-in-time Restore and S3 pit restore when multiple files and larger directory trees need to be restored from versioned buckets.

Point-in-time Restore

The S3 Versioning and the Amazon EventBridge can restore individual objects or large datasets. This solution provides a complete bucket inventory and determines object changes from the specified timestamp, the snapshot can be restored into the same or a different bucket. For more information about the process and enablement of point-in-time restore, see *Point-in-time restore for Amazon S3 buckets*.

S3 pit restore Tool

The S3 pit restore Tool can restore previous versions of the data stored in S3 when S3 versioning is enabled. You can restore some or all of the files to a certain point in time to a local file system, the same S3 bucket, or a different S3 bucket. For more information, see the *S3 pit restore* documentation.

Comparison of Point-in-time Restore and S3 pit restore

Both Point-in-time Restore and S3 pit restore can be used to restore directories/prefixes to a previous version when S3 versioning is enabled. The Point-in-time Restore can also provide audit/event logs to show changes in S3, which give a more detailed view of the performed operations. However, the Point-in-time Restore can only restore changed objects and relies on other AWS services and requires more steps. The dependency on other AWS services also affects the pricing calculation for Point-in-time Restore. For more information, see *Point-in-time restore for Amazon S3 buckets* and the *AWS S3 Pricing* page. While the S3 pit restore Tool has no additional cost, the maintainability of the tool is not certain and requires users to configure their own access, secret key, and region. Furthermore, Delegation Tokens will not work.

Related Information

Point-in-time restore for Amazon S3 buckets

S3 pit restore

AWS S3 Pricing

Cloudera Management Console

AWS Environments

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Working with AWS environments 5

Introduction to AWS environments 5

Register environment (UI) 6

Register environment (CLI) 11

Enabling admin and user access 11

Obtain CLI commands 12

Understanding environment UI options 13

Using Compute Clusters 15

Setting up Compute Cluster IAM permissions 15

Creating IAM roles and instance profile for EKS 15

Creating Compute Restricted IAM policy 21

Enabling default Compute Cluster for new environments 29

Enabling default Compute Cluster for existing environments 32

Adding more Compute Clusters 33

Managing Compute Clusters 34

Suspending and resuming Compute Clusters 35

Monitoring an environment 36

Environment status options 37

Stop and restart an environment 39

Delete an environment 39

Cleaning up a failed environment 41

Add subnets to an environment 41

Add security groups 42

Add root SSH key to an environment 42

Change environment's credential 43

Enabling environment telemetry 43
Defining anonymization rules 45

Adding a customer managed encryption key to a Cloudera environment running on AWS 47
Environment and Cloudera Data Hub encryption options 47
AWS prerequisites for using a CMK 47
Register an AWS environment with a CMK 47
Set a CMK for an existing AWS environment 48

Deploying Cloudera in multiple AWS availability zones 49
Enabling multi-AZ 53

Defining custom tags 55

Updating instance metadata to IMDSv2 57

Restricting access for Cloudera services 61

Configure lifecycle management for logs on AWS 62

Troubleshooting for RAZ-enabled AWS environment 66

Working with AWS environments

Refer to the following documentation to learn about creating and managing AWS environments in Cloudera:

Related Information

Managing provisioning credentials for AWS
Managing Data Lakes
Managing FreeIPA

Introduction to AWS environments

In Cloudera, an environment is a logical subset of your cloud provider account including a specific virtual private network. You can register as many environments as you require.

The “environment” concept of Cloudera is closely related to the virtual private network in your cloud provider account. Registering an environment provides Cloudera with access to your cloud provider account and identifies the resources in your cloud provider account that Cloudera services can access or provision. A single environment is contained within a single cloud provider region, so all resources deployed by Cloudera are deployed within that region within one specific virtual network. Once you’ve registered an environment in Cloudera, you can start provisioning Cloudera resources such as clusters, which run on the physical infrastructure in an AWS data center.

The following diagram enumerates the components of an environment:

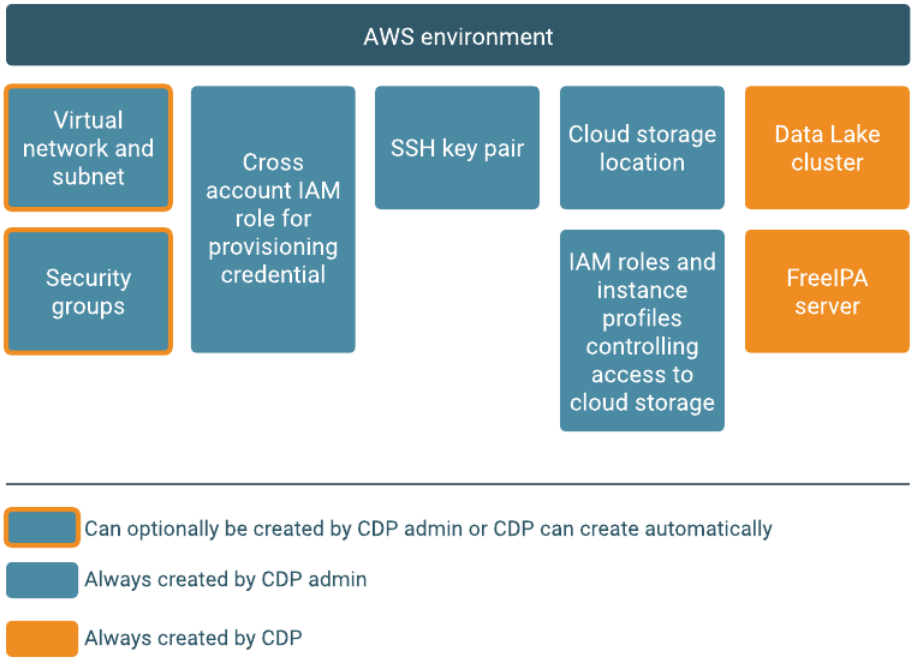

The diagram illustrates all major user-created and Cloudera-created components of an environment:

- The items in dark blue boxes with orange outlines can either be automatically provisioned by Cloudera on your AWS account, or your can optionally pre-create them and provide them when registering an environment.

- The items in dark blue boxes must be pre-created by your Cloudera administrator prior to environment registration and then provided when registering an environment.
- The items in orange boxes are automatically provisioned on AWS by Cloudera as part of environment provisioning.

Note: The items that are user-created don't get terminated during environment deletion.

As shown in the diagram, an environment consists of the following resources:

Environment component	Description
Virtual network with subnets	An environment corresponds to one specific virtual network and subnets in which Cloudera resources are provisioned. The network is located within one specified region; therefore, you also need to select a region for each environment.
Security groups	Security groups act as a virtual firewall for your instances to control inbound and outbound traffic. All VM instances provisioned within an environment use your specified security access settings allowing inbound access to your instances from your organization's computers.
Cross-account role for provisioning credential	Cloudera uses a provisioning credential for authorization to provision resources (such as compute instances) within your cloud provider account. On AWS, the credential uses a cross-account IAM role with an attached IAM policy listing all required permissions.
SSH public key	When registering an environment on a public cloud, a Cloudera administrator provides an SSH public key. This way, the administrator has root-level access to the Data Lake instance and Cloudera Data Hub cluster instances.
Cloud storage location and AIM roles and instance profiles allowing access to that location	When registering an environment, you must provide one or more S3 buckets for storing: • All workload cluster data • Ranger audits, and FreeIPA logs • Cluster service logs You must also provide IAM instance profiles that control access to the bucket(s).
Data lake	A data lake is automatically provisioned when an environment is created. It provides a mechanism for storing, accessing, organizing, securing, and managing data.
FreeIPA	A FreeIPA server is automatically provisioned when an environment is created. It is responsible for synchronizing your users and making them available to Cloudera services, Kerberos service principal management, and more.

You may want to register multiple environments corresponding to different regions that your organization would like to use. Once your environment is running, you can provision Cloudera Data Hub clusters, Cloudera Data Warehouses, and other resources in it.

Register an AWS environment from the Cloudera UI

Once you've met the AWS cloud provider requirements, register your AWS environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in AWS requirements.

Required role: EnvironmentCreator

Steps

1. Navigate to the Cloudera Management Console > Environments > Register environment:
2. On the Register Environment page, provide the following information:

Parameter	Description
General Information	
Environment Name (Required)	Enter a name for your environment. The name: • Must be between 5 and 28 characters long. • Can only include lowercase letters, numbers, and hyphens. • Must start with a lowercase letter.
Description	Enter a description for your environment.
Select Cloud Provider (Required)	Select Amazon.
Credential (Required)	
Select Credential	Select an existing credential or select Create new credential. For instructions on how to create a credential, refer to Creating a role-based credential. **Note:** Activate the Enable Permission Verification button if you want Cloudera to check permissions for your credential. Cloudera will verify that you have the required permissions for your environment.

3. Click Next.
4. On the Data Access and Data Lake Scaling page, provide the following information:

Parameter	Description
Data Lake Settings	
Data Lake Name (Required)	Enter a name for the Data Lake cluster that will be created for this environment. The name: • Must be between 5 and 100 characters long • Must contain lowercase letters • Cannot contain uppercase letters • Must start with a letter • Can only include the following accepted characters are: a-z, 0-9, -.
Data Lake Version (Required)	Select Cloudera Runtime version that should be deployed for your Data Lake. The latest stable version is used by default. All Cloudera Data Hub clusters provisioned within this Data Lake will be using the same Cloudera Runtime version.
Fine-grained access control on S3	
Enable Ranger authorization for AWS S3 Identity	Enable this if you would like to use Fine-grained access control. Next, from the Select AWS IAM role for Ranger authorizer dropdown, select the DATALAKE_ADMIN_ROLE IAM role created in Minimal setup for cloud storage.
Data Access and Audit	
Assumer Instance Profile (Required)	Select the IDBROKER_ROLE instance profile created in Minimal setup for cloud storage.
Storage Location Base (Required)	Provide the S3 location created for data storage in Minimal setup for cloud storage.
Data Access Role (Required)	Select the DATALAKE_ADMIN_ROLE IAM role created in Minimal setup for cloud storage.
Ranger Audit Role (Required)	Select the RANGER_AUDIT_ROLE IAM role created in Minimal setup for cloud storage.
IDBroker Mappings	We recommend that you leave this out and set it up after registering your environment as part of Onboarding Cloudera users and groups for cloud storage. **Note:** If you are using Fine-grained access control, this option is disabled, because you should onboard your users and groups via Ranger instead of using IDBroker mappings.

Parameter	Description
Scale (Required)	Select Data Lake scale. By default, “Light Duty” is used. For more information on Data Lake scale, refer to Data Lake scale.
Enable **Compute Cluster**	Enable Compute Clusters if you would like to deploy a containerized platform on Kubernetes for data services and shared services.

5. Click on Advanced Options to make additional configurations for your Data Lake. The following options are available:

Parameter	Description
Network and Availability	
Enable Multiple Availability Zones for Data Lake	Click the Enable Multiple Availability Zones for Data Lake toggle button to enable multi-AZ for the Data Lake. This option is disabled by default and is only available when a Medium Duty Data Lake is selected. Refer to Deploying Cloudera in multiple AWS availability zones.
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Amazon EC2 instance types.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on a specific Data Lake host group. For more information, see Recipes.

6. Click Next.
7. On the Region, Networking and Security page, provide the following information:

Parameter	Description
Region	
Select Region (Required)	Select the region that you would like to use for Cloudera. If you would like to use a specific existing virtual network, the virtual network must be located in the selected region.
Customer-managed Keys	
Enable Customer-Managed Keys	Enable this if you would like to provide a Customer-Managed Key (CMK) to encrypt environment's disks and databases. Next, under Select Encryption Key, select an existing CMK. For more information, refer to Customer managed encryption keys.
Select Encryption Key	Select an existing CMK.
Network	
Select Network (Required)	You have two options: • Select the existing virtual network where you would like to provision all Cloudera resources. Refer to VPC and subnet. • Select Create new network to have a new network with three subnets created. One subnet is created for each availability zone assuming three AZs per region; If a region has two AZs instead of three, then still three subnets are created, two in the same AZ.
Select Subnets (Required)	This option is only available if you choose to use an existing network. Multiple subnets must be selected and Cloudera distributes resources evenly within the subnets.
Network CIDR (Required)	This option is only available if you select to create a new network. If you selected to create a new network, provide Network CIDR that determines the range of private IPs that EC2 instances will use. This must be a valid private IP CIDR IP in IPv4 range. For example 10.10.0.0/16 are valid IPs. /16 is required to allow for enough IP addresses.
Create Private Subnets	This option is only available if you select to have a new network and subnets created. Is is turned on by default so that private subnets are created in addition to public subnets. If you disable it, only public subnets will be created. **Important:** For production deployments, Cloudera Data Platform recommends that you use private subnets. Work with your internal IT teams to ensure that users can access the browser interfaces for cluster services.

Parameter	Description
Create Private Endpoints	This option is only available if you select to have a new network and subnets created. It is disabled by default. Enable this option to use private endpoints instead of public endpoints for the following services: • Amazon EC2 • Amazon ECR - api and dkr • Amazon EFS • Amazon RDS for PostgreSQL • AWS Auto Scaling • AWS CloudFormation • AWS ELB • AWS S3 • AWS STS
Enable Public Endpoint Access Gateway	When Cluster Connectivity Manager is enabled, you can optionally enable Public Endpoint Access Gateway to provide secure connectivity to UIs and APIs in Data Lake and Cloudera Data Hub clusters deployed using private networking. If you are using your existing VPC, under Select Endpoint Access Gateway Subnets, select the public subnets for which you would like to use the gateway. The number of subnets must be the same as under Select Subnets and the availability zones must match. For more information, refer to Public Endpoint Access Gateway documentation.
Proxies	
Select Proxy Configuration	Select a proxy configuration if previously registered. For more information refer to Setting up a proxy server.
Security Access Settings	
Select Security Access Type (Required)	This determines inbound security group settings that allow connections to the Data Lake and Cloudera Data Hub clusters from your organization's computers. You have two options: • Create new security groups - Allows you to provide custom CIDR IP range for all new security groups that will be created for the Data Lake and Cloudera Data Hub clusters so that users from your organization can access cluster UIs and SSH to the nodes. This must be a valid CIDR IP in IPv4 range. For example: 192.168.27.0/24 allows access from 192.168.27.0 through 192.168.27.255. You can specify multiple CIDR IP ranges separated with a comma. For example: 192.168.27.0/24,192.168.28.0/24 If you use this setting, several security groups will get created: one for each Data Lake host group the Data Lake and one for each host group), one for each FreeIPA host group, and one for RDS; Furthermore, the security group settings specified will be automatically used for Cloudera Data Hub, Cloudera Data Warehouse, and Cloudera AI clusters created as part of the environment. • Provide existing security groups (Only available for an existing VPC) - Allows you to select two existing security groups, one for Knox-installed nodes and another for all other nodes. If you select this option, refer to Security groups to ensure that you open all ports required for your users to access environment resources.
Kubernetes	
Select **Private Kubernetes Cluster** or provide **Authorized IP Ranges**	If you have enabled Compute Clusters, you have the following options to configure the necessary networking information for the Kubernetes cluster: • Enable Private Kubernetes Cluster to create a private cluster that blocks all access to the API Server endpoint. • Provide the CIDRs to the Kubernetes API Server Authorized IP Ranges field to specify a set of IP ranges that will be allowed to access the Kubernetes API server. You need to provide the advanced configurations only once when creating your environment. The configurations will be applied to all compute clusters in the environment.
Worker Node Subnets	Uses the same set of subnets provided in **Network** section. You have the option to not use all of the previously provided subnets.
SSH Settings	

Parameter	Description
New or existing SSH public key (Required)	You have two options for providing a public SSH key: • Select a key that already exists on your AWS account within the specific region that you would like to use. • Upload a public key directly from your computer. **Note:** Cloudera does not use this SSH key. The matching private key can be used by your Cloudera administrator for root-level access to the instances provisioned for the Data Lake and Cloudera Data Hub.
Add tags	You can optionally add tags to be created for your resources on AWS. Refer to Defining custom tags.

8. Click on Advanced Options to make additional configurations for the FreeIPA cluster. The following options are available:

Parameter	Description
Network and Availability	
Enable Multiple Availability Zones for Data Lake	Click the Enable Multiple Availability Zones for Data Lake toggle button to enable multi-AZ for the FreeIPA cluster. Refer to Deploying Cloudera in multiple AWS availability zones.
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Amazon EC2 instance types.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on a specific FreeIPA host group. For more information, see Recipes.

9. Click Next.
10. On the Storage page, provide the following information:

Parameter	Description
Logs	
Logger Instance Profile (Required)	Select the LOG_ROLE instance profile created in Minimal setup for cloud storage.
Logs Location Base (Required)	Provide the S3 location created for log storage in Minimal setup for cloud storage.
Backup Location Base	Provide the S3 location created for FreeIPA and Data Lake backups in Minimal setup for cloud storage. If not provided, the default Backup Location Base uses the Logs Location Base.
Telemetry	
Enable Workload Analytics	Enables Cloudera Observability support for workload clusters created within this environment. When this setting is enabled, diagnostic information about job and query execution is sent to Cloudera Observability. For more information, refer to Enabling workload analytics and logs collection.
Enable Deployment Cluster Logs Collection	When this option is enabled. the logs generated during deployments will be automatically sent to Cloudera. For more information, refer to Enabling workload analytics and logs collection.

11. Click on Register Environment to trigger environment registration.
12. The environment creation takes about 60 minutes. The creation of the FreeIPA server and Data Lake cluster is triggered. You can monitor the progress from the web UI. Once the environment creation has been completed, its status will change to “Running”.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.

- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Register an AWS environment from the CDP CLI

Once you've met the AWS cloud provider requirements, register your AWS environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in AWS requirements.

Required role: EnvironmentCreator

Steps

Unlike in the Cloudera web interface, in CDP CLI environment creation is a three-step process with environment creation, setting IDBroker mappings and Data Lake creation being three separate steps. The easiest way to obtain the correct commands is to provide all parameters in Cloudera web interface and then generate the CDP CLI commands on the last page of the wizard. For detailed steps, refer to Obtain CLI commands for registering an environment.

To learn more about how to create Compute Cluster enabled environments with CLI, see Enabling default Compute Cluster for new environments.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Enabling admin and user access to environments

In order to grant admin and user access to an environment that you registered in Cloudera, you should assign the required roles.

You need to be an EnvironmentCreator in order to register an environment. Once an environment is running, the following roles can be assigned:

- EnvironmentAdmin - Grants all rights to the environment and Cloudera Data Hub clusters running in it, except the ability to delete the environment. The user who registers the environment automatically becomes its EnvironmentAdmin.
- EnvironmentUser - Grants permission to view Cloudera Data Hub clusters and set the workload password for the environment. This role should be used in conjunction with service-specific roles such as DataHubAdmin, DWAdmin, DWUser, MLAdmin, MLUser, and so on. When assigning one of these service-specific roles to users, make sure to also assign the EnvironmentUser role.
- DataSteward - Grants permission to perform user/group management functions in Ranger and Atlas Admin, manage ID Broker mappings, and start user sync for the environment.
- Owner - Grants the ability to mange the environment in Cloudera, including deleting the environment. The user who registers the environment automatically becomes its Owner. The Owner role does not grant access the environment's clusters (Data Lakes, Cloudera Data Hub clusters).

The roles are described in detail in Resource roles. The steps for assigning the roles are described in Assigning resource roles to users and Assigning resource roles to groups.

Related Information

Resource roles
Assigning resource roles to users
Assigning resource roles to groups

Obtain CLI commands for registering an environment

Although you can obtain CDP CLI commands for environment creation from CDP CLI help, the easiest way to obtain them is from the Cloudera web interface.

You can quickly obtain CDP CLI commands for creating an environment:

- From details of an existing environment
- From the register environment wizard

Create an environment from an existing environment

Obtain an environment template from an existing environment to create an environment with the exact same settings.

Since creating an environment, setting IDBroker mappings, and creating a Data Lake are separate actions in CDP CLI, you need to obtain three commands.

Required role: EnvironmentUser, EnvironmentAdmin, or Owner

Steps

1. Log in to the Cloudera web interface and navigate to the Cloudera Management Console
2. Navigate to environment details > Actions > Show CLI commands:

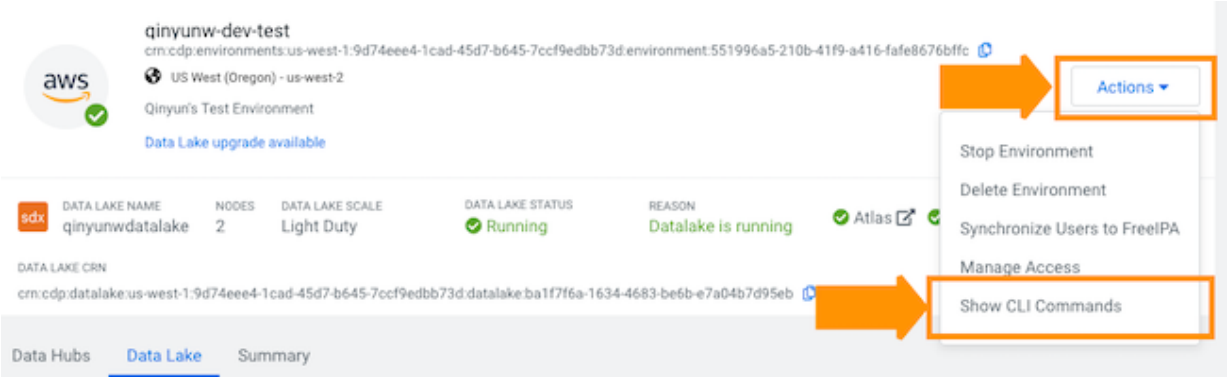

3. Click COPY three times to copy the three commands. These commands allow you to:
   a. Create an environment with the same settings as the existing environment.
   b. Set the same IDBroker mappings as in the original environment.
   c. Create a Data Lake with the same settings.

4. Before you can use these commands, make sure to update the following:
   a. In cdp environments create-<cloud-platform>-environment, update the value of --environment-name. It should be unique within Cloudera.
   b. In cdp environments set-id-broker-mappings, update the value of --environment-name. It should reference the name of the new environment that you are planning to create.
   c. In cdp datalake create-<cloud-platform>-datalake, update the value of --datalake-name. It should be unique within Cloudera.
   d. In cdp datalake create-<cloud-platform>-datalake, update the value of --environment-name. It should reference the name of the new environment that you are planning to create.
5. Run the three commands to:
   a. Register an environment.
   b. Set IDBroker mappings.
   c. Create a Data Lake.

Obtain CLI commands from the register environment wizard

Provide environment parameters in the environment wizard, and then on the last page of the wizard generate a CDP CLI template to create an environment and Data Lake with the parameters specified in the wizard. The obtained cluster template can be used to create an environment with the same settings via CDP CLI.

Since creating an environment, setting IDBroker mappings, and creating a Data Lake are separate actions in CDP CLI, you need to obtain three commands.

Required role: EnvironmentCreator

Steps

1. Log in to the Cloudera web interface and navigate to the Cloudera Management Console
2. Navigate to Environments > Register Environment.
3. Provide all the parameters for your environment.
4. On the last page, click SHOW CLI COMMAND in the bottom of the page.
5. Copy the three commands (for creating an environment, setting IDBroker mappings, and creating a Data Lake).
6. Run the commands:
   a. First run the command that creates the environment.
   b. Once the command finishes and the environment is running, run the command that sets IDBroker mappings.
   c. Next, run the command that creates the Data Lake.

Understanding environment UI options

To access information related to your environment, navigate to the Cloudera Management Console service > Environments and click on your environment.

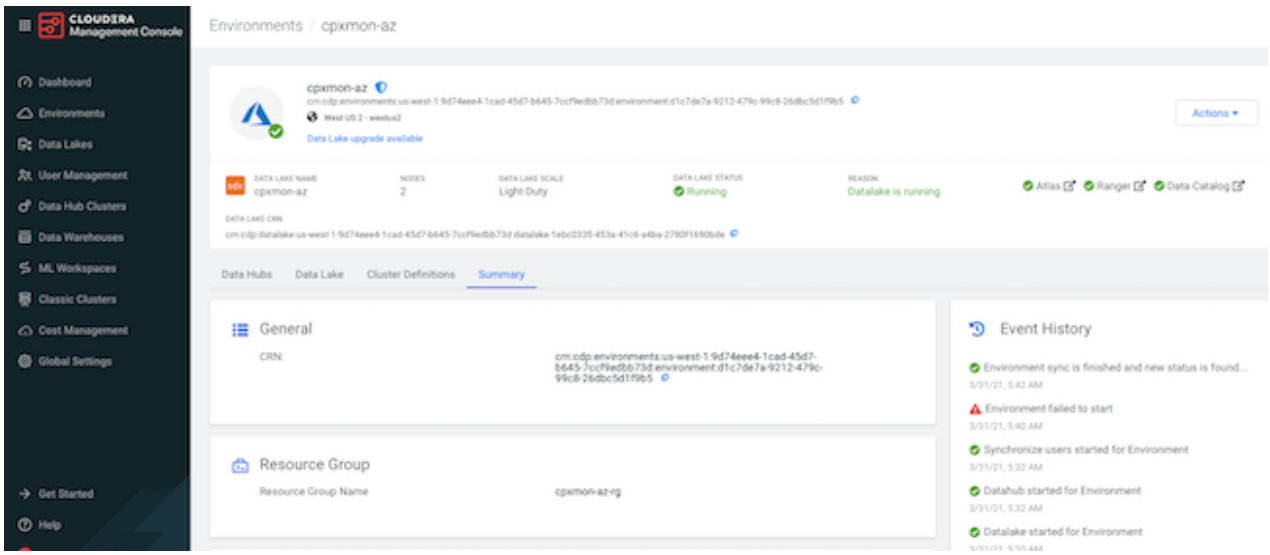

You need to have the EnvironmentUser role or higher for the environment in order to access details of that environment.

From environments details, you can access the following:

- From the Cloudera Management Console tab, you can create, manage, and access Cloudera Data Hub clusters within the environment.
- From the Data Lake tab, you can monitor, manage, and access the Data Lake cluster.
- From the Cluster Definitions tab, you can access all cluster definitions that can be used with the environment.
- From the Summary tab, you can manage and monitor your environment.

The Summary includes the following information:

Option	Description
General	This includes your environment's CRN. CRN is an ID that Cloudera uses to identify a resource.
Credential	This links the provisioning credential associated with the environment and includes the option to change the credential.
Region	This lists the region in which your environment is deployed.
Network	This lists the networking resources used by your environment, provided by you or created by Cloudera during environment registration. You can add additional subnets for Cloudera Data Hub clusters deployed in the future.
Security Access	This lists the security groups used by your environment, provided by you or created by Cloudera during environment registration. You can provide new security groups for Data Hub clusters deployed in the future.
FreeIPA	This includes details of a FreeIPA server running in the environment and includes an Actions menu with FreeIPA management options.
Log Storage and Audits	This lists the cloud storage location used for logs and audits that you provided during environment registration. There is no way to update this location once your environment is running.
Telemetry	This includes your environment's telemetry settings. You can change them for any Cloudera Data Hub clusters created in the future.
Advanced	This lists the name of your root SSH key. You can add an additional SSH key if needed. The newly added SSH key will only be used forCloudera Data Hub clusters created in the future.

Related Information

Understanding Cloudera Data Hub details
Understanding Data Lake details

Using Compute Clusters

Compute Clusters enable you to deploy a containerized platform on Kubernetes for Data Services and shared services.

The Compute Cluster architecture offers simplified management, enhanced efficiency, and centralized control that leads to faster deployments, reduced configuration errors and improved system reliability. As multiple Data Services can optionally share the same Compute Cluster, it also lowers the cost of ownership.

When registering a Compute Cluster enabled environment, a default Compute Cluster is created. This default Compute Cluster is tied to the environment, and is used for running critical applications. The default Compute Cluster is labeled as **Default Cluster** in your environment to distinguish it from the other Compute Clusters.

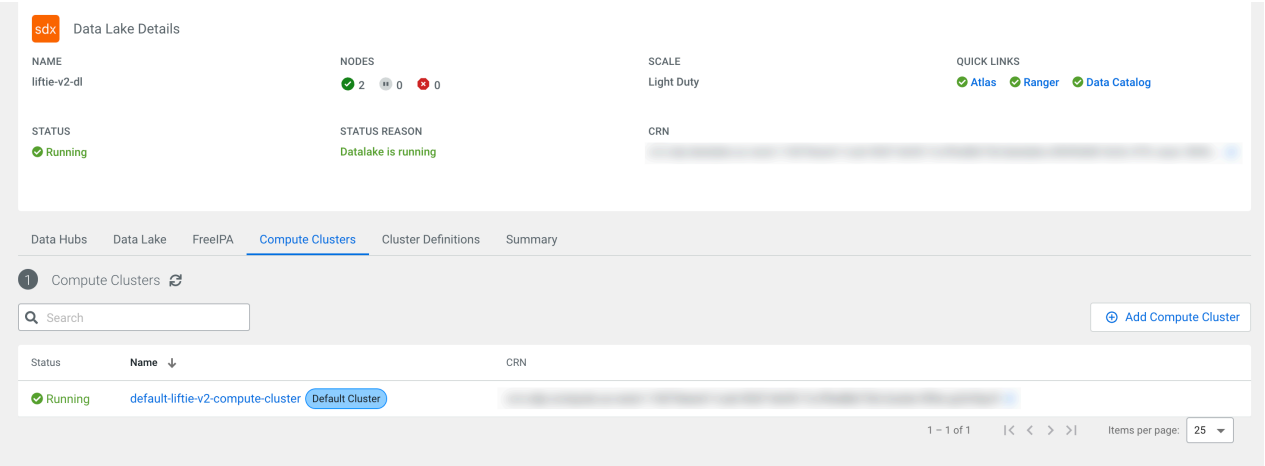

You can create additional Compute Clusters that inherit the configuration of the default Compute Cluster. You can manage the lifecycle of the additional Compute Clusters as they are independent of the environment.

Setting up Compute Cluster IAM permissions

Before enabling Compute Clusters for your environment, you need to ensure that the required IAM roles and policies are set up.

Setting up the Compute Cluster IAM permissions are only needed when using **Reduced access policies** detailed on Cross-account access IAM role page, and completing these steps are not required when using default policies.

Creating IAM roles and instance profile for EKS

Complete the steps to create the required IAM roles and profile for EKS.

Procedure

1. Apply the following CloudFormation template to create the following:
   - IAM role called cdp-eks-master-role
   - IAM role and instance profile pair called cdp-liftie-instance-profile

```
AWSTemplateFormatVersion: 2010-09-09
Description: Creates Liftie IAM resources
Parameters:
```

```
  TelemetryLoggingEnabled:
    Description: Telemetry logging is enabled
    Type: String
  TelemetryLoggingBucket:
    Description: Telemetry logging bucket where Liftie logs will be store
d.
    Type: String
  TelemetryKmsKeyARN:
    Description: KMS Key ARN For Telemetry logging bucket.
    Type: String
    Default: ""
  TelemetryLoggingRootDir:
    Description: Telemetry logging root directory inside telemetry logg
ing bucket used for storing logs.
    Default: "cluster-logs"
    Type: String
Conditions:
  TelemetryLoggingEnabled:
    Fn::Equals:
      - {Ref: TelemetryLoggingEnabled}
      - true
  KMSKeyARNForTelemetryLoggingBucketIsEmpty: !Not [!Equals [!Ref Telemetry
KmsKeyARN, ""]]
Resources:
  AWSServiceRoleForAmazonEKS:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - eks.amazonaws.com
            Action:
              - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonEKSServicePolicy
        - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
      RoleName: cdp-eks-master-role
  NodeInstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: "/"
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
        - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
        - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
      RoleName: cdp-liftie-instance-profile
      Policies:
        - PolicyName: ssm-required
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
```

```
                Action:
                  - ssm:GetParameters
                Resource:
                  - "*"
        - PolicyName: cluster-autoscaler
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - autoscaling:DescribeAutoScalingGroups
                  - autoscaling:DescribeAutoScalingInstances
                  - autoscaling:DescribeLaunchConfigurations
                  - autoscaling:DescribeScalingActivities
                  - autoscaling:DescribeTags
                  - ec2:DescribeImages
                  - ec2:DescribeInstanceTypes
                  - ec2:DescribeLaunchTemplateVersions
                  - ec2:GetInstanceTypesFromInstanceRequirements
                  - eks:DescribeNodegroup
                Resource:
                  - "*"
              - Effect: Allow
                Action:
                  - autoscaling:SetDesiredCapacity
                  - autoscaling:TerminateInstanceInAutoScalingGroup
                Resource:
                  - "*"
                Condition:
                  StringEquals:
                    "aws:ResourceTag/k8s.io/cluster-autoscaler/enabled":
"true"
        - PolicyName: ebs-csi
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - ec2:CreateSnapshot
                  - ec2:AttachVolume
                  - ec2:DetachVolume
                  - ec2:ModifyVolume
                  - ec2:DescribeAvailabilityZones
                  - ec2:DescribeInstances
                  - ec2:DescribeSnapshots
                  - ec2:DescribeTags
                  - ec2:DescribeVolumes
                  - ec2:DescribeVolumesModifications
                Resource: "*"
              - Effect: Allow
                Action:
                  - ec2:CreateTags
                Resource:
                  - "arn:aws:ec2:*:*:volume/*"
                  - "arn:aws:ec2:*:*:snapshot/*"
                Condition:
                  StringEquals:
                    "ec2:CreateAction":
                      - CreateVolume
                      - CreateSnapshot
              - Effect: Allow
                Action:
                  - ec2:DeleteTags
                Resource:
```

```
                  - "arn:aws:ec2:*:*:volume/*"
                  - "arn:aws:ec2:*:*:snapshot/*"
              - Effect: Allow
                Action:
                  - ec2:CreateVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "aws:RequestTag/ebs.csi.aws.com/cluster": "true"
              - Effect: Allow
                Action:
                  - ec2:CreateVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "aws:RequestTag/CSIVolumeName": "*"
              - Effect: Allow
                Action:
                  - ec2:CreateVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "aws:RequestTag/kubernetes.io/cluster/*": "owned"
              - Effect: Allow
                Action:
                  - ec2:DeleteVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
              - Effect: Allow
                Action:
                  - ec2:DeleteVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "ec2:ResourceTag/CSIVolumeName": "*"
              - Effect: Allow
                Action:
                  - ec2:DeleteVolume
                Resource: "*"
                Condition:
                  StringLike:
                    "ec2:ResourceTag/kubernetes.io/created-for/pvc/name":
"*"
              - Effect: Allow
                Action:
                  - ec2:DeleteSnapshot
                Resource: "*"
                Condition:
                  StringLike:
                    "ec2:ResourceTag/CSIVolumeSnapshotName": "*"
              - Effect: Allow
                Action:
                  - ec2:DeleteSnapshot
                Resource: "*"
                Condition:
                  StringLike:
                    "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
        - PolicyName: efs-csi
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
```

```
                Action:
                  - elasticfilesystem:DescribeAccessPoints
                  - elasticfilesystem:DescribeFileSystems
                  - elasticfilesystem:DescribeMountTargets
                Resource: "*"
              - Effect: Allow
                Action:
                  - elasticfilesystem:CreateAccessPoint
                Resource: "*"
                Condition:
                  StringLike:
                    "aws:RequestTag/efs.csi.aws.com/cluster": "true"
              - Effect: Allow
                Action:
                  - elasticfilesystem:DeleteAccessPoint
                Resource: "*"
                Condition:
                  StringEquals:
                    "aws:ResourceTag/efs.csi.aws.com/cluster": "true"
        - !If
          - TelemetryLoggingEnabled
          - PolicyName: telemetry-s3-list-bucket
            PolicyDocument:
              Version: 2012-10-17
              Statement:
                - Effect: Allow
                  Action:
                    - s3:ListBucket
                  Resource:
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}'
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}/${Tele
metryLoggingRootDir}/*'
          - !Ref 'AWS::NoValue'
        - !If
          - TelemetryLoggingEnabled
          - PolicyName: telemetry-s3-read-write
            PolicyDocument:
              Version: 2012-10-17
              Statement:
                - Effect: Allow
                  Action:
                    - s3:*Object
                    - s3:AbortMultipartUpload
                    - s3:GetBucketAcl
                  Resource:
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}'
                    - !Sub 'arn:aws:s3:::${TelemetryLoggingBucket}/${Tel
emetryLoggingRootDir}/*'
          - !Ref 'AWS::NoValue'
        - !If
          - KMSKeyARNForTelemetryLoggingBucketIsEmpty
          - PolicyName: s3-kms-read-write-policy
            PolicyDocument:
              Version: 2012-10-17
              Statement:
                - Effect: Allow
                  Action:
                    - kms:Decrypt
                    - kms:GenerateDataKey
                  Resource:
                    - !Sub ${TelemetryKmsKeyARN}
          - !Ref 'AWS::NoValue'
        - PolicyName: calico-cni
          PolicyDocument:
```

```
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - ec2:ModifyInstanceAttribute
                Resource:
                  - "*"
                Condition:
                  StringEquals:
                    "ec2:Attribute": "SourceDestCheck"
  NodeInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      InstanceProfileName: cdp-liftie-instance-profile
      Roles:
        - !Ref NodeInstanceRole
```

2. In the AWS console Cloudformation wizard, provide values for the following properties:
   - Stack Name: Provide an appropriate name. Example: compute-precreated-roles-and-instanceprofile)
   - TelemetryLoggingBucket: Name of the log bucket. Example: compute-logging-bucket
   - TelemetryLoggingEnabled: Set it to true.
   - TelemetryLoggingRootDir: Verify that it is set to the default value cluster-logs.
   - TelemetryKMSKeyARN: If the telemetry bucket is encrypted, specify the KMS Key ARN. The default value is null.

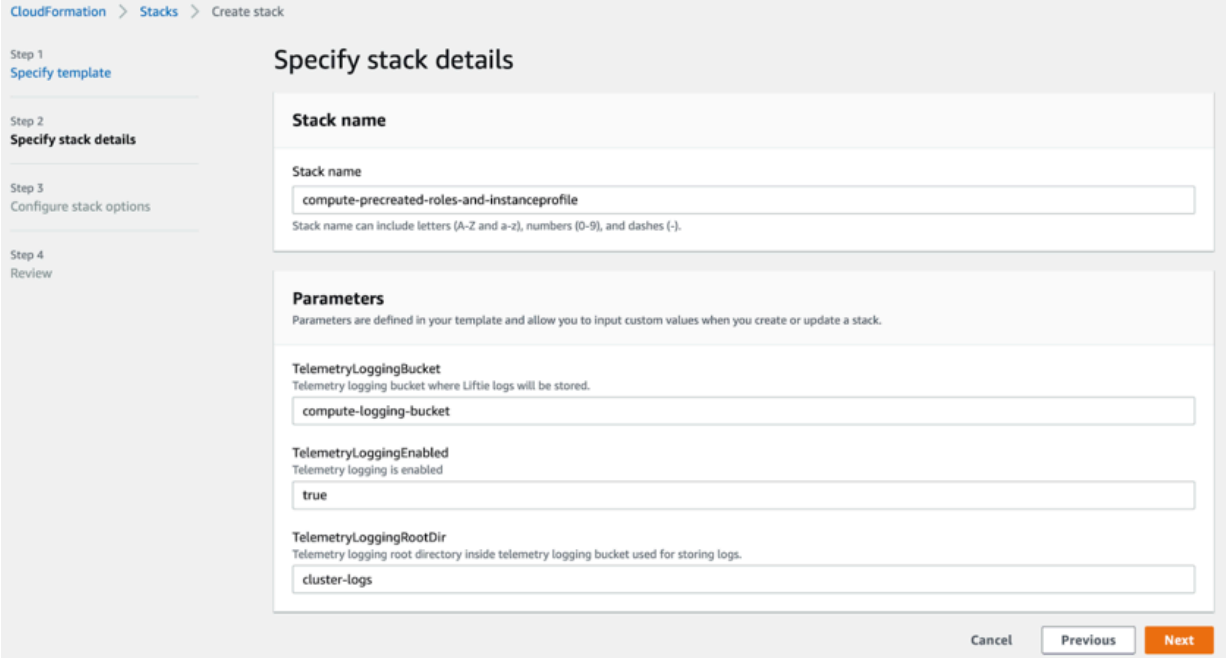

3. On the last page in the wizard process, click the I acknowledge... checkbox to allow creation of IAM resources with special names.

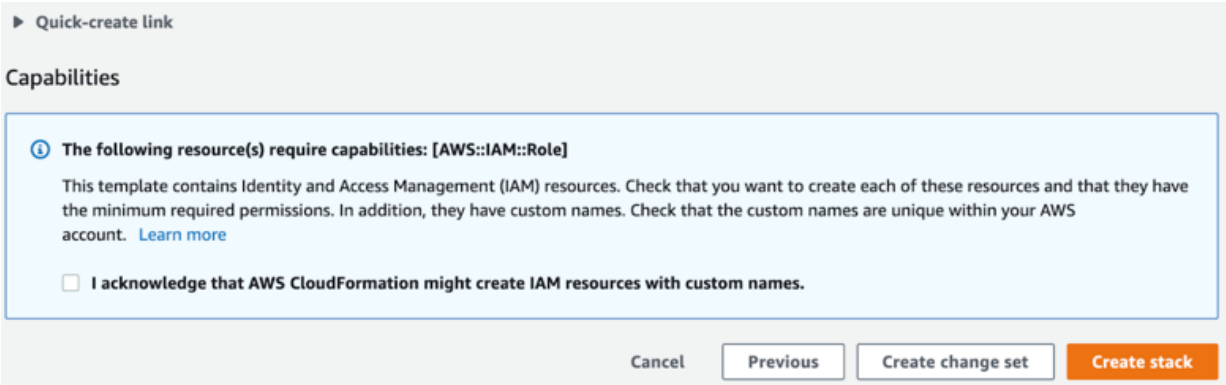

4. Click Create stack.

Results

On the Cloudformation **Resources** tab, you find the precreated role and instance profile.

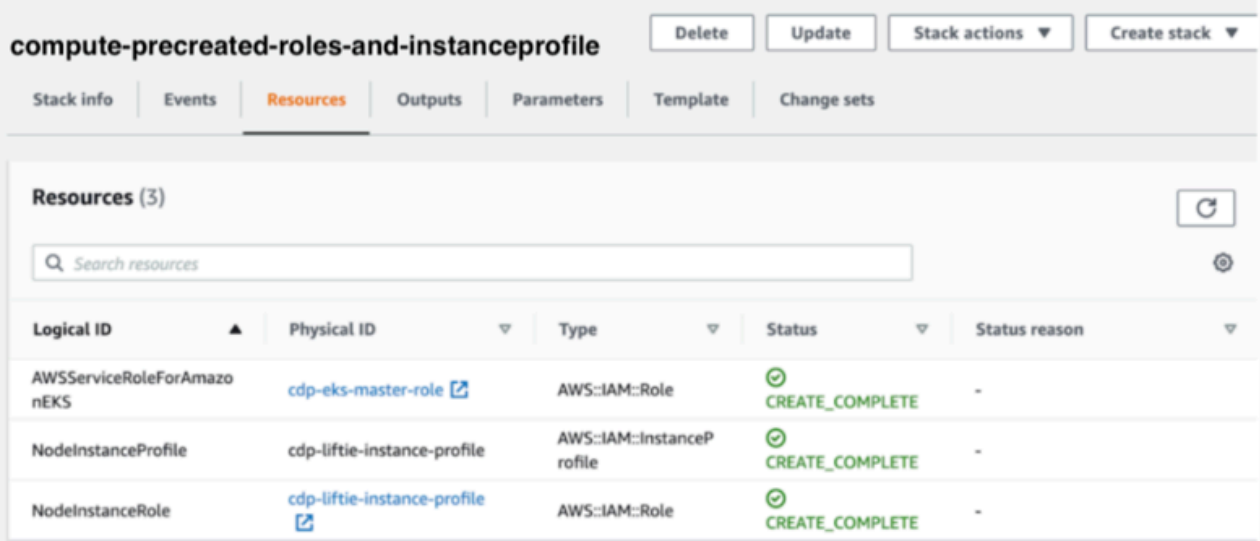

What to do next

Update the environment role to use the restricted role and policy.

Creating Compute Restricted IAM policy

Complete the steps to attach the Compute Restricted IAM policy with the cross-account role associated with your environment.

Procedure

1. Go to the **Environments** page.

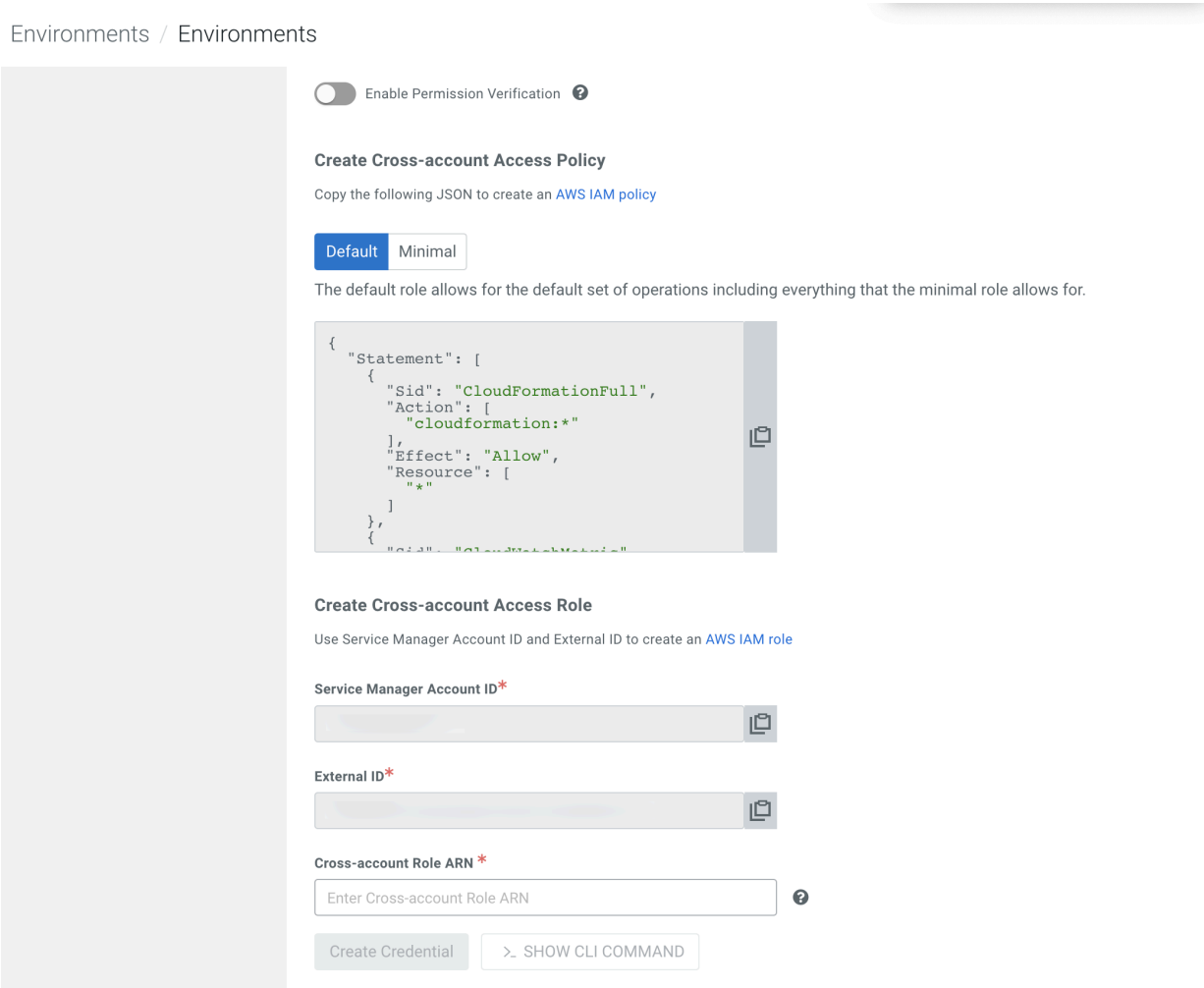

2. In the Create Cross-account Access Policy field, attach the Compute Restricted IAM policy:

   Replace the following placeholders in the JSON file:

   - *[YOUR-ACCOUNT-ID]* with your account ID in use.
   - *[YOUR-IAM-ROLE-NAME]* with the IAM restricted role associated with this policy.
   - *[YOUR-SUBNET-ARN-*]* supplied during the Cloudera Environment(s) creation.

     Note: Provide all the subnets present in all the Cloudera Environment(s) that you intend to use it for the experience. If at any point a new Cloudera Environment is created or an existing one is updated for subnets, provide it here.
   - *[YOUR-IDBROKER-ROLE-NAME]* with the ID Broker Role name in use.
   - *[YOUR-LOG-ROLE-NAME]* with the Log Role name in use.
   - *[YOUR-KMS-CUSTOMER-MANAGED-KEY-ARN]* with KMS key ARN.

```
{
  "Version": "2012-10-17",
  "Id": "ComputePolicy_v10",
  "Statement": [
    {
      "Sid": "SimulatePrincipalPolicy",
      "Effect": "Allow",
      "Action": [
```

```
        "iam:SimulatePrincipalPolicy"
      ],
      "Resource": [
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-IAM-ROLE-NAME]"
      ]
    },
    {
      "Sid": "RestrictedPermissionsViaClouderaRequestTag",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:CreateChangeSet",
        "ec2:createTags",
        "eks:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/Cloudera-Resource-Name": [
            "crn:cdp:*"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedPermissionsViaClouderaResourceTag",
      "Effect": "Allow",
      "Action": [
        "autoscaling:DetachInstances",
        "autoscaling:ResumeProcesses",
        "autoscaling:SetDesiredCapacity",
        "autoscaling:SuspendProcesses",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:DeleteTags",
        "autoscaling:TerminateInstanceInAutoScalingGroup",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStacks"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/Cloudera-Resource-Name": [
            "crn:cdp:*"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedPermissionsViaCloudFormation",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateSecurityGroup",
        "ec2:DeleteSecurityGroup",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:CreateLaunchTemplate",
        "ec2:DeleteLaunchTemplate",
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:DeleteAutoScalingGroup",
        "autoscaling:CreateOrUpdateTags",
        "autoscaling:CreateLaunchConfiguration",
        "eks:CreateCluster",
```

```
        "eks:DeleteCluster"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedEC2PermissionsViaClouderaResourceTag",
      "Effect": "Allow",
      "Action": [
        "ec2:RebootInstances",
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ForAnyValue:StringLike": {
          "ec2:ResourceTag/Cloudera-Resource-Name": [
            "crn:cdp:*"
          ]
        }
      }
    },
    {
      "Sid": "RestrictedIamPermissionsToClouderaResources",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-IDBROKER-ROLE-NAME]",
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/[YOUR-LOG-ROLE-NAME]",
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/liftie-*-eks-service-role",
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/liftie-*-eks-worker-nodes",
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/cdp-eks-master-role",
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/cdp-liftie-instance-profile"
      ]
    },
    {
      "Sid": "RestrictedKMSPermissionsUsingCustomerProvidedKey",
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*"
      ],
      "Resource": [
        "[YOUR-KMS-CUSTOMER-MANAGED-KEY-ARN]"
      ]
    },
    {
      "Sid": "AllowCreateDeleteTagsForSubnets",
      "Effect": "Allow",
```

```
        "Action": [
          "ec2:CreateTags",
          "ec2:DeleteTags"
        ],
        "Resource": [
          "arn:aws:ec2:[YOUR-SUBNET-REGION]:[YOUR-ACCOUNT-ID]:subnet/*"
        ]
      },
      {
        "Sid": "OtherPermissionsViaCloudFormation",
        "Effect": "Allow",
        "Action": [
          "autoscaling:DescribeScheduledActions",
          "autoscaling:DescribeTags",
          "autoscaling:DescribeAutoScalingInstances",
          "autoscaling:DescribeLaunchConfigurations",
          "autoscaling:DeleteLaunchConfiguration",
          "autoscaling:DescribeScalingActivities",
          "dynamodb:DescribeTable",
          "ec2:DeletePlacementGroup",
          "ec2:DescribeAccountAttributes",
          "ec2:DescribeImages",
          "ec2:DescribeInstanceStatus",
          "ec2:DescribeInstances",
          "ec2:DescribeKeyPairs",
          "ec2:DescribeLaunchTemplateVersions",
          "ec2:DescribeLaunchTemplates",
          "ec2:DescribePlacementGroups",
          "ec2:DescribeRegions",
          "ec2:DescribeRouteTables",
          "ec2:DescribeSecurityGroups",
          "ec2:DescribeVolumes"
        ],
        "Resource": [
          "*"
        ],
        "Condition": {
          "ForAnyValue:StringEquals": {
            "aws:CalledVia": [
              "cloudformation.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "ModifyInstanceAttribute",
        "Effect": "Allow",
        "Action": [
          "ec2:ModifyInstanceAttribute"
        ],
        "Resource": [
          "*"
        ],
        "Condition": {
          "StringEquals": {
            "ec2:Attribute": "SourceDestCheck"
          }
        }
      },
      {
        "Sid": "OtherPermissionsViaClouderaResourceTag",
        "Effect": "Allow",
        "Action": [
          "cloudformation:DescribeChangeSet",
```

```
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:ListStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResource",
        "cloudformation:DescribeStackResources",
        "cloudwatch:deleteAlarms",
        "cloudwatch:putMetricAlarm",
        "logs:DescribeLogStreams",
        "logs:FilterLogEvents",
        "ec2:AttachVolume",
        "ec2:CreateNetworkInterface",
        "ec2:CreateVolume",
        "ec2:DeleteVolume",
        "ec2:RunInstances",
        "eks:ListUpdates",
        "eks:UpdateClusterConfig",
        "eks:UpdateClusterVersion",
        "eks:DescribeUpdate",
        "iam:GetRolePolicy",
        "iam:ListInstanceProfiles",
        "iam:ListRoleTags",
        "iam:RemoveRoleFromInstanceProfile",
        "iam:TagRole",
        "iam:UntagRole"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/Cloudera-Resource-Name": [
            "crn:cdp:*"
          ]
        }
      }
    },
    {
      "Sid": "OtherPermissions",
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateLaunchTemplateVersion",
        "ec2:CreatePlacementGroup",
        "ec2:DeleteKeyPair",
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInstanceTypes",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcs",
        "ec2:ImportKeyPair",
        "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
        "ec2:GetInstanceTypesFromInstanceRequirements",
        "eks:DescribeCluster",
        "elasticloadbalancing:DescribeLoadBalancers",
        "iam:GetRole",
        "iam:ListRoles",
        "iam:GetInstanceProfile"
      ],
```

```
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowSsmParams",
      "Effect": "Allow",
      "Action": [
        "ssm:DescribeParameters",
        "ssm:GetParameter",
        "ssm:GetParameters",
        "ssm:GetParameterHistory",
        "ssm:GetParametersByPath"
      ],
      "Resource": [
        "arn:aws:ssm:*:*:parameter/aws/service/eks/optimized-ami/*"
      ]
    },
    {
      "Sid": "CfDeny",
      "Effect": "Deny",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ForAnyValue:StringLike": {
          "cloudformation:ImportResourceTypes": [
            "*"
          ]
        }
      }
    },
    {
      "Sid": "ForAutoscalingLinkedRole",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/autoscaling-
plans.amazonaws.com/AWSServiceRoleForAutoScalingPlans_EC2AutoScaling"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "autoscaling-plans.amazonaws.com"
        }
      }
    },
    {
      "Sid": "ForEksLinkedRole",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/eks.amazo
naws.com/AWSServiceRoleForEKS"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "eks.amazonaws.com"
```

```
          }
        }
      }
    ]
}
```

3. Provide and verify your Customer Managed Key (CMK) to be used for EBS encryption.

   Along with providing the KMS Customer Managed Key (CMK) for volume encryption in the policy section with Sid: RestrictedKMSPermissionsUsingCustomerProvidedKey, you need to verify that the policy for the Customer Managed Key (CMK) at KMS (this is not an IAM policy) has the following three permission blocks defined for AWSServiceRoleForAutoScaling.

```
{
  "Statement": [
    {
      "Sid": "AllowAutoscalingServiceLinkedRoleForAttachmentOfPersisten
tResources",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/auto
scaling.amazonaws.com/AWSServiceRoleForAutoScaling"
      },
      "Action": "kms:CreateGrant",
      "Resource": "*",
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Sid": "AllowAutoscalingServiceLinkedRoleUseOfTheCMK",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::[YOUR-ACCOUNT-ID]:role/aws-service-role/autos
caling.amazonaws.com/AWSServiceRoleForAutoScaling"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow EKS access to EBS.",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "kms:CreateGrant",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
```

```
            "StringEquals": {
              "kms:CallerAccount": "[YOUR-ACCOUNT-ID]",
              "kms:viaService": "ec2.[YOUR-ACCOUNT-REGION].amazonaws.com"
            }
          }
        }
      ]
    }
```

After the policy is attached, the KMS service page will show the CMS as having the policy attached as shown in the following example:

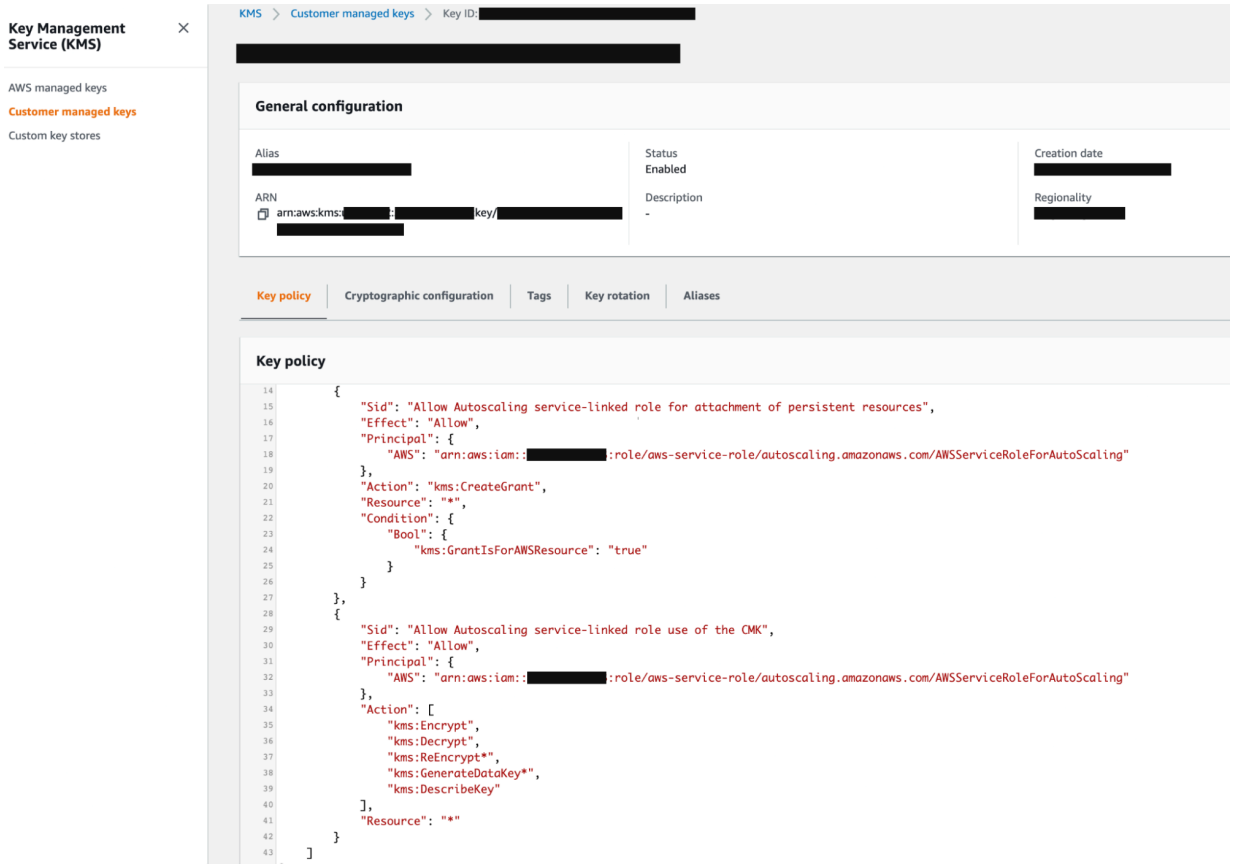

Enabling default Compute Cluster for new environments

When creating your environment, you can enable the default Compute Cluster using the Cloudera Management Console or CDP CLI to be able to run your data and shared services on the containerized platform.

Required role: EnvironmentAdmin

Before you begin

- Ensure that your AWS account has all the resources required by Cloudera.

  For more information, see AWS account requirements.
- Ensure that the IAM permissions are correctly set up for your environment.

  For more information, see Setting up Compute Cluster IAM permissions.

Using Cloudera Management Console

When creating your environment in **Cloudera Management Console**, ensure that you use the **Enable Compute Cluster** setting to create the Compute Cluster enabled environment.

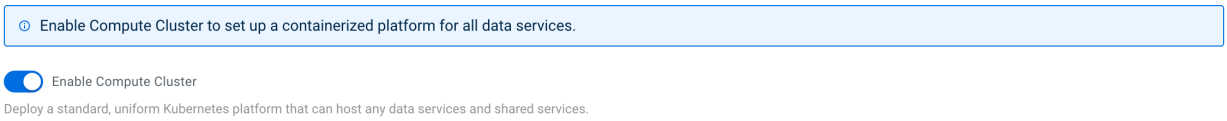

After completing the step for **Data Access and Data Lake Scaling**, configure the networking settings for Kubernetes with either selecting the **Private Kubernetes Cluster** or providing **Authorized IP Ranges** on the **Region, Networking and Security** page. **Worker Node Subnets** are automatically pre-filled with the same set of subnets provided in **Network** section, but you have the option to not use all of the available subnets.

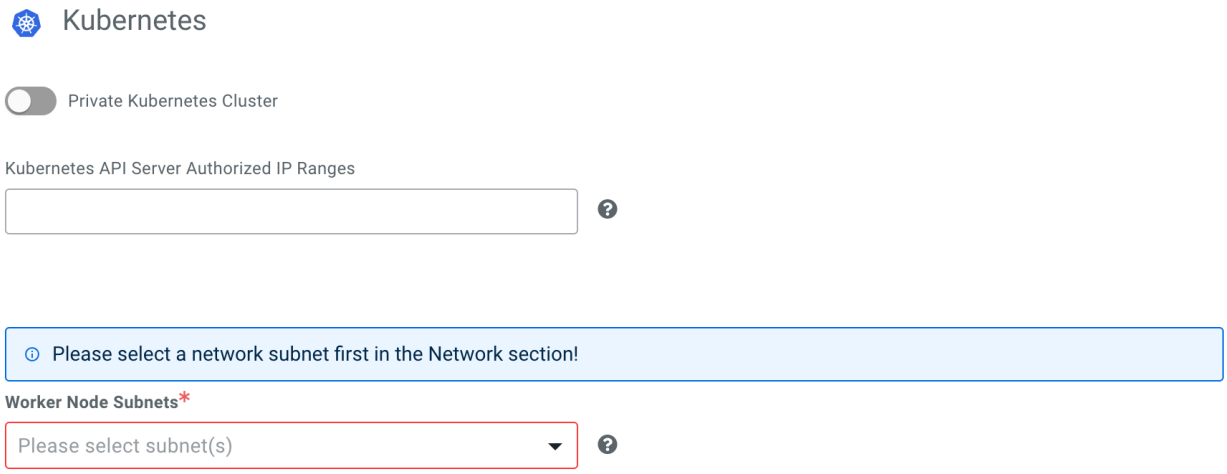

For more information about creating your environment, see the Register environment (UI) documentation.

Using CDP CLI

Run the following command to create the Compute Cluster enabled environment:

For Without private cluster

```
cdp environments create-aws-environment
--environment-name [***ENVIRONMENT NAME***] \
--credential-name [***CREDENTIAL NAME***] \
--region [***REGION***] \
--security-access [***SECURITY CONTROL CONFIGURATIONS***] \
--authentication [***PUBLIC SSH KEY***] \
--log-storage [***STORAGE CONFIGURATION***] \
--enable-compute-cluster \
--compute-cluster-configuration \
privateCluster=FALSE, \
kubeApiAuthorizedIpRanges=[***CIDR1***],[***CIDR2***]
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

For With private cluster

```
cdp environments create-aws-environment
--environment-name [***ENVIRONMENT NAME***] \
```

```
--credential-name [***CREDENTIAL NAME***] \
--region [***REGION***] \
--security-access [***SECURITY CONTROL CONFIGURATIONS***] \
--authentication [***SSH KEY***] \
--log-storage [***STORAGE CONFIGURATION***] \
--enable-compute-cluster \
--compute-cluster-configuration \
privateCluster=TRUE
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

After the command runs, you can verify if the environment was successfully created with the default Compute Cluster with using the following commands:

- Describing the environment:

```
cdp environments describe-environment --env-name-or-crn [***ENVIRONMENT
 NAME OR CRN***]

...
        "awsComputeClusterConfiguration": {
            "privateCluster": false,
            "kubeApiAuthorizedIpRanges": [
                "0.0.0.0/0"
            ]
        },
        "enableComputeCluster": "true"
...
```

- Listing Compute Clusters:

```
cdp compute list-clusters --env-name-or-crn [***ENVIRONMENT NAME OR
 CRN***]
```

Note: FreeIPA must be created and running before the default Compute Cluster is created.

You can use the following command to retry the environment creation with the default Compute Cluster:

For Without private cluster

```
cdp environments initialize-aws-compute-cluster
--environment-name [***ENVIRONMENT NAME***] \
--compute-cluster-configuration \
privateCluster=FALSE, \
kubeApiAuthorizedIpRanges=[***CIDR1***],[***CIDR2***]
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

For With private cluster

```
cdp environments initialize-aws-compute-cluster
--environment-name [***ENVIRONMENT NAME***] \
--compute-cluster-configuration \
privateCluster=TRUE
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

Enabling default Compute Cluster for existing environments

In case you already have an environment, but would like to have your services to run on the containerized platform enabled by Compute Clusters, you can add the default Compute Cluster to your existing environment.

Required resource role: EnvironmentAdmin

Before you begin

- Ensure that the environment has no default Compute Cluster provisioned.
- Ensure that the environment is started and available.
- Ensure that IAM permissions are correctly set up during the environment creation.

  For more information, see Setting up Compute Cluster IAM permissions.

Using Cloudera Management Console

1. Navigate to your environment.
2. Click Compute Clusters.

   The following message will indicate that Compute Cluster are not enabled for your environment:

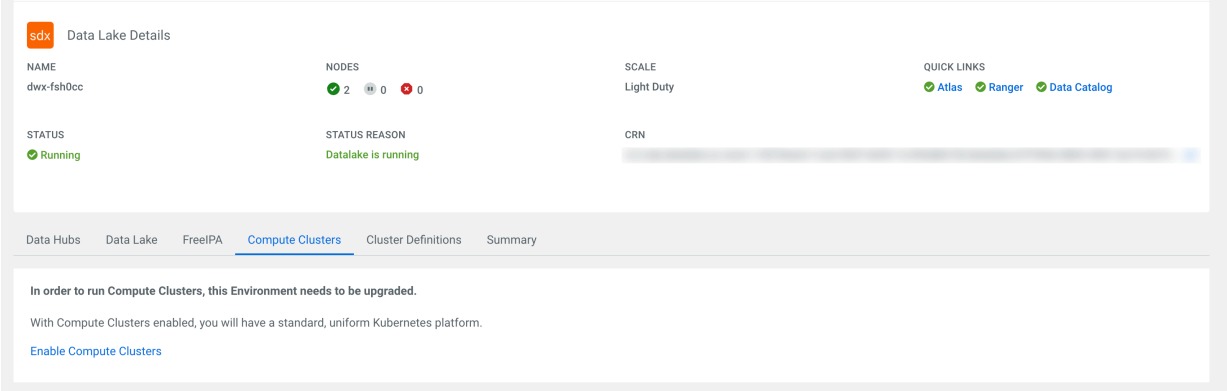

3. Click Enable Compute Clusters.
4. Provide the necessary networking information for the **Kubernetes** cluster.

   a. If you need to create a Private Cluster, enable Private Kubernetes Cluster to create a private cluster that blocks all access to the API Server endpoint.

   b. If you do not need to create a Private Cluster, provide the CIDRs to the Kubernetes API Server Authorized IP Ranges field to specify a set of IP ranges that will be allowed to access the Kubernetes API server.

   c. **Worker Node Subnets** are automatically pre-filled with the same set of subnets provided during environment registration, but you have the option to not use all of the available subnets.

5. Click Submit.

   You will be redirected to the **Compute Clusters** tab, where you can track the creation process of the default Compute Cluster.

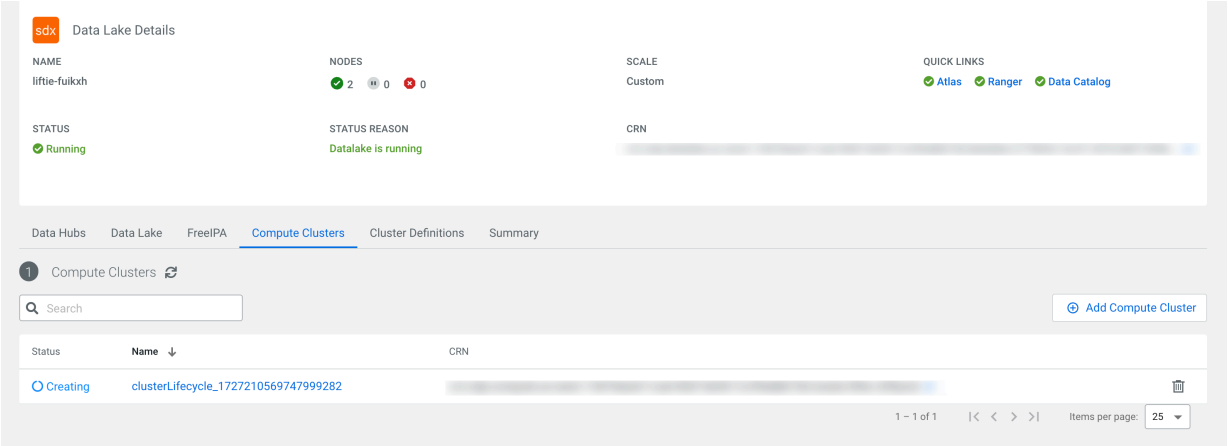

Using CDP CLI

Run the following command to add the default Compute Cluster to the environment:

For Without private cluster

```
cdp environments initialize-aws-compute-cluster
--environment-name [***ENVIRONMENT NAME***] \
--compute-cluster-configuration \
privateCluster=FALSE, \
kubeApiAuthorizedIpRanges=[***CIDR1***],[***CIDR2***]
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

For With private cluster

```
cdp environments initialize-aws-compute-cluster
--environment-name [***ENVIRONMENT NAME***] \
--compute-cluster-configuration \
privateCluster=TRUE
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

The environment will have COMPUTE_CLUSTER_CREATION_IN_PROGRESS status. You can use the following command to check the status of the environment creation, the statusReason field will contain the information about the process:

```
cdp environments describe-environment --env-name-or-crn [***ENVIRONMENT NAME
 OR CRN***]
```

For more detailed status information about the cluster creation, you can use the following command:

```
cdp compute list-clusters --env-name-or-crn [***ENVIRONMENT NAME OR CRN***]
```

Adding more Compute Clusters

You can add as many additional Compute Clusters as required beside the default Compute Cluster using Cloudera Management Console or CDP CLI.

Required role: EnvironmentAdmin

Using Cloudera Management Console

You can create additional Compute Clusters beside the default Compute Cluster using Cloudera Management Console.

1. Navigate to your environment.
2. Select Compute Clusters tab.
3. Click Add Compute Cluster.

   The **Add Compute Cluster** wizard appears.

   Add Compute Cluster

   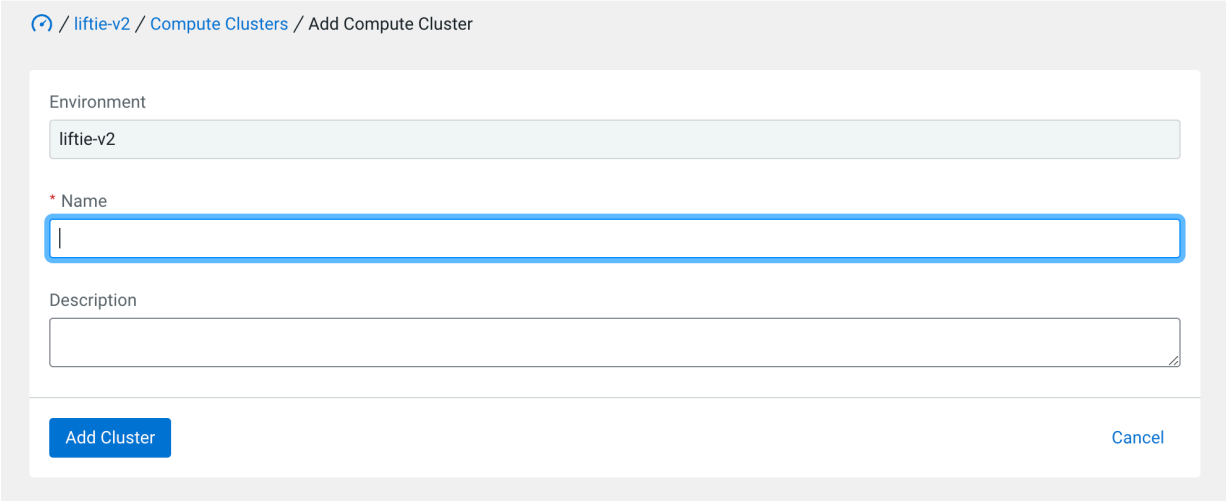

4. Provide a Name to the cluster, and optionally a Description.
5. Click Add Cluster.

   You will be redirected to the **Compute Clusters** tab, where you can track the creation process of the additional Compute Cluster.

Using CDP CLI

You can use the following CDP CLI command to create additional Compute Clusters after the default Compute Cluster is created:

```
cdp compute create-cluster
--environment --env-name-or-crn [***ENVIRONMENT NAME OR CRN***] \
--name [***CLUSTER NAME***]
```

After the command runs, you can verify if the additional Compute Cluster creation was successful using the following command:

```
cdp compute list-clusters --env-name-or-crn [***ENVIRONMENT NAME OR CRN***]
```

The additional Compute Cluster status should be in RUNNING state.

Managing Compute Clusters

After creation, you can view the Compute Cluster details, manage the access of the clusters, download the Kubeconfig file, and delete the created additional Compute Clusters.

Required resource role: EnvironmentAdmin or Owner

Required account role: IamViewer

Using Cloudera Management Console

1. Click on the Name of the Compute Cluster.

   You will be redirected to the **Cluster Details** page.

   On the **Cluster Details** page, you can view the **Status**, **Creation Date**, **Created By**, **Description** and **CRN** of the Compute Cluster.
2. Click Actions to open the drop-down menu.

   Note: Ensure that you have one of the following roles to access the **Actions** menu
   - EnvironmentOwner or EnvironmentAdmin
   - ClusterCreator or ClusterOwner

   a. Click Manage Access to update the list of users who have access to manage the Compute Cluster and use it for installing Data Services.

      1. Click Update roles to update the **Resource Role** of a user or group.

         You can assign the Owner resource role to a user or group to provide permission to manage the Compute Cluster and install services on it.

   b. Click Kubernetes Access to grant access for users to the Kubernetes API server.

      1. Enter the User ARN.
      2. Click Grant Access.

      You also have the option to Download the Kubeconfig from this page.

   c. Click Delete Compute Cluster, if you no longer need the additional Compute Cluster.

      Warning: Ensure that the data services running on the Compute Cluster are deleted before deleting the Compute Cluster itself.

      1. Confirm the deletion of the additional Compute Cluster by clicking Remove.
3. Click Networking tab to view the subnet information of the Compute Cluster.

   a. Click to update the **Kubernetes API Server Authorized IP Ranges**.

      Note: Ensure that you have one of the following roles to access the **Actions** menu
      - EnvironmentOwner or EnvironmentAdmin
      - ClusterCreator or ClusterOwner

      1. Add or remove the CIDRs, and click Save.
4. Click **Encryption** tab to view the encryption key of the Compute Cluster.
5. Click **Node Groups** tab to have an illustrated overview of the resource utilization of the different services running on the Compute Cluster.
6. Click **Compute Cluster Version** to view the Kubernetes and Insfrastructure Service versions of the cluster.
7. Click **Logs** to check the different events of the Compute Cluster.

Suspending and resuming Compute Clusters

You can suspend and resume your default and additional Compute Clusters in an environment to manage your cloud costs using CDP CLI.

Before you begin

- Ensure that the environment is started and available.
- Esnure that there are no Data Service workload instances running in the Compute Cluster.

Important: Currently this feature is only available for Cloudera environments on AWS.

Required resource role: EnvironmentAdmin or Owner

Procedure

1. Run the following command to suspend the Compute Cluster:

```
cdp compute suspend-cluster --cluster-crn [***CLUSTER CRN***]
```

After running the command, the cluster status changes to SUSPENDING, and will progress to SUSPENDED once the operation completes. The status of the cluster can be polled using describe-cluster command:

```
cdp compute describe-cluster --cluster-crn [***CLUSTER CRN***]
```

2. Run the following command to resume the Compute Cluster with SUSPENDED status:

```
cdp compute resume-cluster --cluster-crn [***CLUSTER CRN***]
```

After running the command, the cluster status changes to RESUMING, and will progress to RUNNING status once the operation completes.

Monitoring an environment

Once an environment exists, you can access it from the Cloudera Management Console.

Required role: EnvironmentUser, EnvironmentAdmin, or Owner

Steps

For CDP UI

1. To access an existing environment, navigate to Cloudera Management Console > Environments and click on your environment.
2. Click on the Summary tab to access environment details.
3. You can monitor the status of your environment from this page.

For CDP CLI

You can also list available environments from CDP CLI using the cdp environments list-environments command. For example:

```
cdp environments list-environments
{
    "environments": [
        {
            "environmentName": "cdp-demo",
            "crn": "crn:altus:environments:us-west-1:c8dbde4b-ccce-4f8d-
a581-830970ba4908:environment:d3361b40-39ab-4d87-bd5b-abc15f16b90c",
            "status": "DELETE_FAILED",
            "region": "us-east-2",
            "cloudPlatform": "AWS",
            "credentialName": "cdp-demo",
            "description": "Cdp demo"
        },
        {
            "environmentName": "cdp-new",
```

```
            "crn": "crn:altus:environments:us-west-1:c8dbde4b-ccce-4f8d-a5
81-830970ba4908:environment:1d2bacde-5c96-47c1-a597-9f276b824028",
            "status": "AVAILABLE",
            "region": "us-east-2",
            "cloudPlatform": "AWS",
            "credentialName": "cdp-demo",
            "description": ""
        }
}
```

To get more information about a specific environment, you can use the following commands:

```
cdp environments describe-environment --environment-name <value>
```

```
cdp environments get-id-broker-mappings --environment-name <value>
```

Related Information

Accessing Data Lake services
Understanding Data Lake details
Understanding Cloudera Data Hub cluster details
Managing FreeIPA

Environment status options

This topic lists all possible environment status options for the UI and CLI and explains what they mean.

Environment status	Description
Environment creation	
CREATION_INITIATED	Environment creation request was registered in the database and Cloudera is starting the environment creation flow.
ENVIRONMENT_INITIALIZATION_IN_PROGRESS	Setting up the region and network metadata (public/private and cidr).
ENVIRONMENT_VALIDATION_IN_PROGRESS	Setting up the region and network metadata (public/private and cidr).
NETWORK_CREATION_IN_PROGRESS	If the user chose the create new network option, then Cloudera creates the network on cloud provider side.
PUBLICKEY_CREATE_IN_PROGRESS	If the user choose the create new SSH key option, then Cloudera creates the SSH key on cloud provider side.
FREEIPA_CREATION_IN_PROGRESS	Creating the FreeIPA resources for an environment.
Environment update	
UPDATE_INITIATED	Environment update was requested and Cloudera is starting the update flow (network update, load balancer update, SSH key update).
Environment deletion	
DELETE_INITIATED	Environment deletion request was registered and Cloudera is starting the deletion flow.
NETWORK_DELETE_IN_PROGRESS	If the user chose the create new network option, then Cloudera deletes the network on cloud provider side.
PUBLICKEY_DELETE_IN_PROGRESS	If the user choosing the create new SSH key option, then Cloudera deletes the SSH key on cloud provider side.
FREEIPA_DELETE_IN_PROGRESS	Deleting the FreeIPA resources for an environment.
EXPERIENCE_DELETE_IN_PROGRESS	Deleting all the attached clusters (Cloudera Data Warehouse, Cloudera AI, and so on).

Environment status	Description
RDBMS_DELETE_IN_PROGRESS	Deleting all the provisioned RDS instances that are related to an environment.
CLUSTER_DEFINITION_DELETE_PROGRESS	Deleting all the cluster definitions that are created for an environment.
UMS_RESOURCE_DELETE_IN_PROGRESS	Deleting all the related UMS resources for an environment.
IDBROKER_MAPPINGS_DELETE_IN_PROGRESS	Deleting all the IBroker mapping for an environment.
S3GUARD_TABLE_DELETE_IN_PROGRESS	Deleting all the Dynamo DB tables for an environment.
DATAHUB_CLUSTERS_DELETE_IN_PROGRESS	Deleting all the attached Cloudera Data Hub clusters.
DATALAKE_CLUSTERS_DELETE_IN_PROGRESS	Deleting the attached Data Lake cluster.
ARCHIVED	Environment has been deleted (not shown on the UI).
Environment is running	
AVAILABLE	Environment is available (ready to use).
Environment process failed	
CREATE_FAILED	Environment creation failed (Detailed message in the statusReason).
DELETE_FAILED	Environment deletion failed (Detailed message in the statusReason).
UPDATE_FAILED	Environment update failed (Detailed message in the statusReason).
Environment stop	
STOP_DATAHUB_STARTED	Stopping all the Cloudera Data Hub clusters in an environment.
STOP_DATAHUB_FAILED	Stopping all the Cloudera Data Hub clusters in an environment failed (Detailed message in the statusReason).
STOP_DATALAKE_STARTED	Stopping the Data Lake cluster in an environment.
STOP_DATALAKE_FAILED	Stopping the Data Lake cluster in an environment failed (Detailed message in the statusReason).
STOP_FREEIPA_STARTED	Stopping the FreeIPA instances in an environment.
STOP_FREEIPA_FAILED	Stopping the FreeIPA instances in an environment failed (Detailed message in the statusReason).
ENV_STOPPED	Environment was successfully stopped.
Environment start	
START_DATAHUB_STARTED	Starting all the Cloudera Data Hub clusters in an environment.
START_DATAHUB_FAILED	Starting all the Cloudera Data Hub clusters in an environment failed (Detailed message in the statusReason).
START_DATALAKE_STARTED	Starting the Data Lake cluster in an environment.
START_DATALAKE_FAILED	Starting the Data Lake cluster in an environment failed (Detailed message in the statusReason).
START_FREEIPA_STARTED	Starting all the FreeIPA instances in an environment.
START_FREEIPA_FAILED	Starting all the FreeIPA instances failed in an environment (Detailed message in the statusReason).
START_SYNCHRONIZE_USERS_STARTED	Starting user sync for all the clusters in an environment.
START_SYNCHRONIZE_USERS_FAILED	Starting user sync for all the clusters in an environment failed (Detailed message in the statusReason).
FreeIPA instance deletion	
FREEIPA_DELETED_ON_PROVIDER_SIDE	The FreeIPA instance has been deleted on cloud provider side.
Load balancer	
LOAD_BALANCER_ENV_UPDATE_STARTED	Start updating the LoadBalancer on Data Lake in an environment.

Environment status	Description
LOAD_BALANCER_ENV_UPDATE_FAILED	Failed to update the LoadBalancer on Data Lake in an environment (Detailed message in the statusReason).
LOAD_BALANCER_STACK_UPDATE_STARTED	Start updating the LoadBalancer on Cloudera Data Hub clusters in an environment.
LOAD_BALANCER_STACK_UPDATE_FAILED	Failed to update the LoadBalancer on Cloudera Data Hub clusters in an environment (Detailed message in the statusReason).

Stop and restart an environment

You can stop an environment if you need to suspend but not terminate the resources within the environment. When you stop an environment, all of the resources within the environment are also stopped, including Data Lakes and Cloudera Data Hub clusters. You can also restart the environment.

Warning:

The Cloudera AI service does not support environment stop and restart. This means that if Cloudera AI workbenches are running or expected to be provisioned within an environment, then the environment should not be stopped. If done, this will disrupt running Cloudera AI workbenches and prevent successful provisioning of Cloudera AI workbenches in the environment.

Required role: EnvironmentAdmin or Owner

Steps

For CDP UI

1. Navigate to the environment in Cloudera Management Console > Environments.
2. Click Actions Stop Environment and confirm the action.
3. To restart the environment, click Actions Start Environment .

Warning: We have not tested or certified restarting the environment while Cloudera Data Engineering is running.

For CDP CLI

Use the following command to stop an environment:

```
cdp environments stop-environment --environment-name <ENVIRONMENT_NAME>
```

Use the following commands to start an environment:

```
cdp environments start-environment --environment-name <ENVIRONMENT_NAME>
 [--with-datahub-start]
```

Use the following commands to start an environment and all Cloudera Data Hub clusters running in it:

```
cdp environments start-environment --environment-name <ENVIRONMENT_NAME>
 --with-datahub-start
```

Delete an environment

Deleting an environment terminates all resources within the environment including the Data Lake.

Before you begin

To delete an environment, you should first terminate all clusters running in that environment.

Required role: Owner or PowerUser

Steps

For CDP UI

1. In Cloudera Management Console, navigate to Environments.
2. Click on your environment.
3. Click Actions Delete .
4. Check the box next to "I would like to delete all connected resources" if you have Data Lake and Cloudera Data Hub clusters running within the environment. This will delete the Data Lake and Cloudera Data Hub clusters together with the rest of the environment.

   Note: The "I would like to delete all connected resources" option does not delete any Cloudera Data Warehouse or Cloudera AI clusters running within the environment, so these always need to be terminated prior to environment termination.
5. Check the box next to "I understand this action cannot be undone". This is required.
6. Click Delete.

For CDP CLI

When terminating an environment from the CDP CLI, you need to first terminate the Data Lake:

1. Terminate the Data Lake using the following command:

   ```
   cdp datalake delete-datalake --datalake-name <value>
   ```
2. Wait until the Data Lake terminates before proceeding. Use the following commands to check on the status of Data Lake:

   ```
   cdp datalake get-cluster-host-status --cluster-name <value>
   ```

   ```
   cdp datalake list-datalakes
   ```
3. Delete the environment using the following command:

   ```
   cdp environments delete-environment --environment-name <value> --cascadi
   ng
   ```

   The --cascading option deletes or Cloudera Data Hub clusters running in the environment.

If environment deletion fails, you can:

- Repeat the environment deletion steps, but also check "I would like to force delete the selected environment". Force deletion removes Cloudera resources from Cloudera, but leaves cloud provider resources running.
- Clean up cloud resources that were left on your cloud provider account. See Cleaning up a failed AWS environment.

Only the resources that were provisioned as part of the environment are deleted. For example, if a new network was created by Cloudera for the environment, the network will be deleted; But if you provided your existing network, it will not be deleted as part of environment deletion.

Cleaning up a failed environment

When environment creation fails, you should delete the environment. If environment termination fails, you should clean up any resources that might have already been created on your AWS account.

When environment creation fails, you should delete the environment by using the steps described in Delete an Environment.

If environment termination fails, you should clean up any resources that might have already been created on your AWS account. To do this:

1. Navigate to the AWS Management Console your region CloudFormation .
2. Use the search box to find the stacks corresponding to your Data Lake cluster and FreeIPA server.

   The CF stacks use the following naming convention:

   - The Data Lake cluster stack uses the name that you provided for your data lake
   - The FreeIPA server stack uses the name that you provided for your environment
3. Delete the CloudFormation stacks corresponding to your Data Lake and FreeIPA server instances:

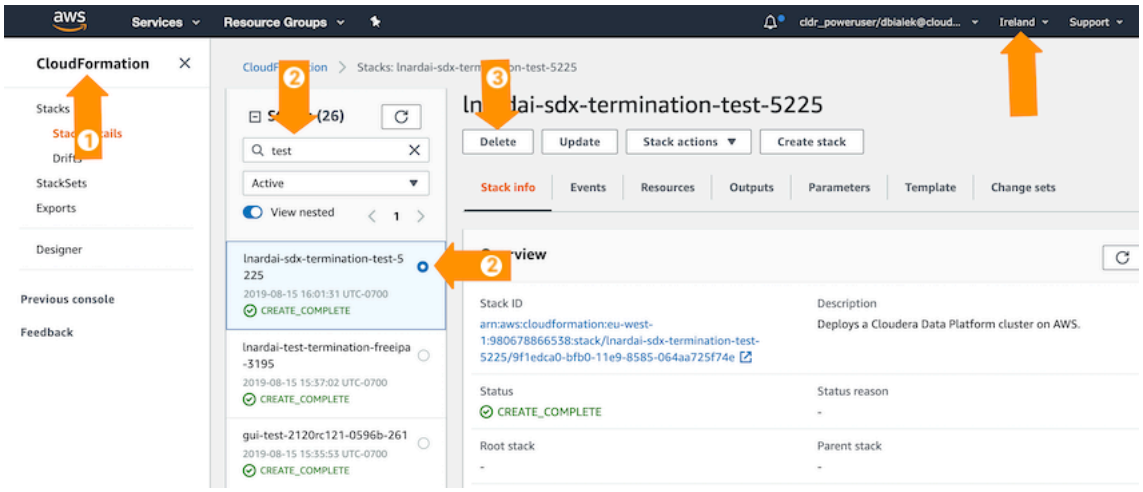

Add subnets to an environment

You can add additional subnets to an existing environment. These subnets will only be used for all Cloudera Data Hub clusters created within the environment in the future.

Before you begin

These steps assume that you have already created the subnets that you want to add to the environment.

Required role: EnvironmentAdmin or Owner

Steps

1. Navigate to Cloudera Management Console > Environments and click on the environment you want to modify.
2. Click the Summary tab.
3. Scroll down to the Network section.
4. Click the (edit) button, then click the Select Subnets pull down menu and select the subnet you want to add to the Environment.

5. Click the (save) button.

   You should see the new subnet listed in the Network section.

   Note:

   The newly added subnets will not be used for any Cloudera services other than Cloudera Data Hub. The newly added subnets will only be used for the Cloudera Data Hub clusters created within the environment after the new subnets were added. All the existing environment resources such as the Data Lake, FreeIPA, and any existing Cloudera Data Hub clusters will remain within the subnets originally defined during environment creation.

Add security groups to an environment

You can add additional security groups to an existing environment. These security groups will be used for all Cloudera Data Hub clusters created within the environment in the future.

Before you begin

These steps assume that you have already created the security groups that you want to add to the environment.

Required role: EnvironmentAdmin or Owner

Steps

1. Navigate to Cloudera Management Console > Environments and click on the environment you want to modify.
2. Click the Summary tab.
3. Scroll down to the Security Access section.
4. Click the (edit) button, then under Select Security Access Type select the Provide Existing Security Groups option and select the security groups that you want to add to the Environment.
5. Click the (save) button.

   You should see the new security groups listed in the Security Access section.

   Note:

   The newly added security groups will only be used for the Cloudera Data Hub clusters created within the environment after the new security groups were added. All the existing environment resources such as the Data Lake, FreeIPA, and any existing Cloudera Data Hub clusters will remain within the security groups originally defined during environment creation.

Add root SSH key to an environment

You can add an additional SSH public key to an existing environment. This SSH public key will be used for root access to the Cloudera Data Hub clusters created within the environment in the future.

Before you begin

These steps assume that you have already have the SSH key pair that you want to add.

Required role: EnvironmentAdmin or Owner

Steps

1. Navigate to Cloudera Management Console > Environments and click on the environment you want to modify.
2. Click the Summary tab.
3. Scroll down to the Advanced section.

4. Click the (edit) button, then under Root SSH select New SSH public key and paste the SSH public key that you want to add to the Environment. If your environment is on AWS, you can also select Existing SSH public key and then select the SSH public key as long as you have previously uploaded it on AWS.
5. Click the (save) button.

Note:

The newly added SSH key will only be used for root access to the Cloudera Data Hub clusters created within the environment after the new SSH key pair was added. All the existing environment resources such as the Data Lake, FreeIPA, and any existing Cloudera Data Hub clusters will remain accessible via the SSH key pair originally defined during environment creation.

Change environment's credential

You can change the credential attached to an environment as long as the new credential provides the required level of access to the same AWS account as the old credential.

Required roles:

- EnvironmentAdmin or Owner of the environment
- SharedResourceUser or Owner of the credential

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Environments from the navigation pane.
4. Click on a specific environment.
5. Navigate to the Summary tab.
6. Scroll down to the Credential section.
7. Click to access credential edit options.
8. Select the new credential that you would like to use.
9. Click to save the changes.

For CDP CLI

If you would like to delete a credential from the CDP CLI, use:

```
cdp environments update-environment-credential --environment-name <value>
 --credential-name <value>
```

Enabling environment telemetry

You can optionally enable workload analytics so that diagnostic information about job and query execution is sent to Cloudera Observability for Cloudera Data Hub clusters. Similarly, you can optionally enable logs collection so that logs generated during deployments will be automatically sent to Cloudera.

Required role:

- PowerUser can set environment telemetry settings for the whole tenant.
- EnvironmentCreator can set environment telemetry settings during environment registration.
- EnvironmentAdmin or Owner can set environment telemetry settings for an existing environment.

Enabling workload analytics

If you enable workload analytics, diagnostic information about job and query execution is sent to Cloudera Observability for Cloudera Data Hub clusters created within all environments. This is disabled by default and can be enabled:

- For the whole tenant:
  - From the Cloudera web interface by navigating to Cloudera Management Console>Global Settings>Telemetry, by turning on the Enable Workload Analytics.
  - Or from the CDP CLI using the following command:

    ```
    cdp environments set-account-telemetry --workload-analytics
    ```

- For a specific environment only:
  - During environment creation from the Cloudera web interface, by turning on the Enable Workload Analytics option under Logs Storage and Audits in the environment creation wizard.
  - For an existing environment, from environment details > Telemetry by turning on the Enable Workload Analytics.
  - For an existing environment, from the CDP CLI using the following command:

    ```
    cdp environments set-telemetry-features --environment-name <some-name> -
    -workload-analytics
    ```

The environment-level setting overrides the tenant-level setting.

Note:

Only Cloudera Data Hub clusters created after enabling workload analytics on an environment will send data to Cloudera Observability. Cloudera Data Hub clusters created before workload analytics was enabled will not start sending data to Cloudera Observability if workload analytics is enabled for their parent environment.

Enabling cluster logs collection

If you enable cluster logs collection, logs generated during deployments will be automatically sent to Cloudera. This is disabled by default and can be enabled:

- For the whole tenant:
  - From the Cloudera web interface by navigating to Cloudera Management Console>Global Settings>Telemetry, by turning on Enable Cluster Logs Collection.
  - Or from the CDP CLI using the following command:

    ```
    cdp environments set-account-telemetry --report-deployment-logs
    ```

- For a specific environment only:
  - During environment creation from the Cloudera web interface, by turning on the Enable Cluster Logs Collection option under Logs Storage and Audits in the environment creation wizard.
  - For an existing environment, from environment details > Summary >Telemetry by turning on the Enable Cluster Logs Collection.
  - For an existing environment, from the CDP CLI using the following command:

    ```
    cdp environments set-telemetry-features --environment-name <some-name> -
    -report-deployment-logs
    ```

The environment-level setting overrides the tenant-level setting.

Disable cloud storage logging for an existing environment

By default, Cloudera sends collected service logs from VM nodes to the cloud storage that you provided for logs during environment registration. In some cases, you may want to disable this for an existing environment.

You can disable this option from environment details > Summary > Telemetry by turning off Enable Cloud Storage Logging.

Note:

Disabling this option will affect only Cloudera Data Hub clusters created after the option was disabled.

Related Information

Configure lifecycle management for logs on AWS

Defining anonymization rules for Cloudera logs

Cloudera includes a set of default anonymization rules and allows you to define custom anonymization rules in order to remove sensitive information from Cloudera logs.

Use PCRE convention for writing custom anonymization rule patterns.

Anonymization rules are applied to the following logs:

- Cluster logs collected during deployments and automatically sent to Cloudera. See Enabling environment telemetry.
- Diagnostics logs that can be collected for troubleshooting and sent to Cloudera Support. See Generating a VM-based diagnostic bundle.

Note: Anonymization rules are only applied to environments created after the rules were added in Cloudera.

Default anonymization rules

Cloudera includes a set of default anonymization rules that anonymize the following:

Anonymization rule (PCRE)	Replacement	Description
\b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-\._]*[A-Za-z0-9])@(([A-Za-z0-9]|[A-Za-z][A-Za-z0-9\-]*[A-Za-z0-9])\.)+([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-]*[A-Za-z0-9])\b	email@redacted.host	Email addresses
\d{4}[^\w]\d{4}[^\w]\d{4}[^\w]\d{4}	XXXX-XXXX-XXXX-XXXX	Credit card numbers
\d{3}[^\w]\d{2}[^\w]\d{4}	XXX-XX-XXXX	SSN
FPW\:\s+[\w|\W].*	FPW: [REDACTED]	FreeIPA (workload) password
cdpHashedPassword=.*[']	[CDP PWD ATTRS REDACTED]	Hashed FreeIPA (workload) password.

Creating anonymization rule patterns

Use PCRE convention for writing anonymization rule patterns. For each pattern, come up with a replacement string.

Define custom anonymization rules

You can define custom anonymization rules in Cloudera. The anonymization rules are only applied to environments created after the rules were added in Cloudera.

Required role: PowerUser

Steps

For CDP UI

1. Once you have created the rules, navigate to Cloudera web interface > Cloudera Management Console > Global Settings > Telemetry > Anonymization rules.
2. Default rules are pre-populated.
3. Click on New rule and add a pattern and replacement string for your rule. Repeat for multiple rules.
4. Test the rules from the same page on the UI under Test rules:
   a. Under Input text paste an example text with sensitive content that should get anonymized by the rules that you added.
   b. Click Test all rules.
   c. The sensitive content should be removed nad replaced in the output printed in the Anonymized result text box.
5. Click Save Changes.

Note:

You can use the Set defaults button if you would like to revert to the default rules.

For CDP CLI

1. If you would like to add new rules, you should first prepare the patterns and replacement strings, and then test them with the following command:

```
cdp environments test-account-telemetry-rules --cli-input-json {
    "testInput": "Email: myemail@cloudera.com",
    "rules": [
        {
            "value": "\\b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\\-\\._]*[A-
Za-z0-9])@(([A-Za-z0-9]|[A-Za-z][A-Za-z0-9\\-]*[A-Za-z0-9])\\.)+([A-Za-
z0-9]|[A-Za-z0-9][A-Za-z0-9\\-]*[A-Za-z0-9])\\b",
            "replacement": "email@redacted.host"
        }
    ]
}
```

2. Run the following command to get your current telemetry settings in JSON format:

```
cdp environments get-account-telemetry
```

3. Copy the JSON file that you obtained in the output of this command and paste it into a text editor.
4. Update the JSON file, updating the settings or adding new rules.

Note:

Make sure to preserve all the existing rules, or else they will be deleted. Also, make sure to pass the workloadAnalytics and cloudStorageLogging parameters. If you don't pass all of the parameters, the parameters that are not passed will get reset to their default values.

5. Once you have the JSON file updated, run the cdp environments set-account-telemetry command. For example:

```
cdp environments set-account-telemetry --cli-input-json {
    "workloadAnalytics": true,
    "cloudStorageLogging": true,
    "rules": [
        {
            "value": "\\b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\\-\\._]*[A-
Za-z0-9])@(([A-Za-z0-9]|[A-Za-z][A-Za-z0-9\\-]*[A-Za-z0-9])\\.)+([A-Za-
z0-9]|[A-Za-z0-9][A-Za-z0-9\\-]*[A-Za-z0-9])\\b",
            "replacement": "email@redacted.host"
```

```
        }
    ]
}
```

Adding a customer managed encryption key to a Cloudera environment running on AWS

By default, Data Lake and FreeIPA's Amazon Elastic Block Store (EBS) volumes and Relational Database Service (RDS) are encrypted using a default key from Amazon’s KMS, but you can optionally configure encryption using Customer Managed Keys (CMK). Data Hubs inherit environment's encryption key by default but you have an option to specify a different CMK during Cloudera Data Hub creation.

Amazon offers the option to encrypt EBS volumes and RDS instances using a default key from Amazon's Key Management System (KMS) or using an external customer-managed KMS. By default, Data Lake and FreeIPA are encrypted using the default key from Amazon’s KMS present in the region where the environment is running, but you can provide a customer-managed KMS key instead of the default key.

By default, Data Hubs use the same default key from Amazon’s KMS or CMK as the parent environment but you have an option to pass a different CMK during Cloudera Data Hub creation.

Encryption is configured for block devices and root devices. When encryption is configured for a given cluster, it is automatically applied to all the disk devices of any new VM instances added as a result of cluster scaling or repair.

This documentation covers the following topics:

- AWS disk encryption options
- AWS prerequisites for using a CMK
- Registering a new AWS environment and specifying a CMK
- Adding a CMK to an existing AWS environment

Environment and Cloudera Data Hub encryption options

To learn about encryption options that Cloudera offers for Data Lake, FreeIPA, and Cloudera Data Hub clusters, refer to Environment and Data Hub encryption options.

AWS prerequisites for using a CMK

You can use your existing AWS CMK or create a new AWS CMK.

For detailed requirements, refer to AWS Requirements: Customer managed encryption keys.

Register an AWS environment with a CMK

You can specify an existing customer managed key (CMK) during AWS environment registration and the encryption key will be used to encrypt the EBS volumes and RDS instances running in the environment.

Steps

For CDP UI

You can register your environment as described in Register an AWS environment from CDP UI, just make sure that on the Data Access and Audit page you enable the following:

1. Under Customer-Managed Keys, click Enable Customer-Managed Keys.
2. In the same section, select the CMK:

The following screenshot shows the UI options:

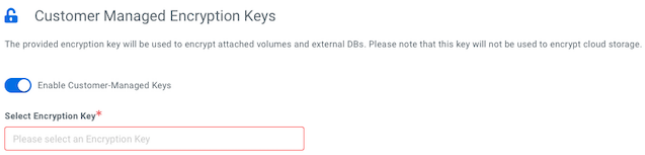

For CDP CLI

You can use your usual CDP CLI command to create an environment with a CMK, just add the --encryption-key-arn parameter and provide the encryption key created earlier.

Use the following CDP CLI command to create an environment with the encryption key created earlier. Replace the placeholders with actual values. For example <ENVIRONMENT-NAME> should be replaced with an actual name. The parameter important for this feature are highlighted:

```
cdp environments create-aws-environment \
--environment-name <ENVIRONMENT-NAME> \
--credential-name <EXISTING_CREDENTIAL> \
--region "<REGION>" \
--security-access cidr=<CIDR> \
--authentication publicKeyId="<SSH_KEY>" \
--log-storage storageLocationBase=<BUCKET_URL>,instanceProfile=<IDBROKE
R_IP> \
--vpc-id <VPC_ID> \
--subnet-ids <SUBNETS \
--encryption-key-arn <ENCRYPTION_KEY_ARN>
```

The ARN of the encryption key created earlier should be passed in the parameter --encryption-key-arn

If the customer-managed encryption key ARN is not passed, then the AWS region-specific default encryption key is used for encrypting EBS volumes and RDS instances.

You can obtain more complete commands using the instructions in Obtain CLI commands for registering an environment.

Set a CMK for an existing AWS environment

You can set a CMK for an existing environment. The CMK will be only used for encrypting disks of Cloudera Data Hub clusters created after the CMK was added.

Note:

The CMK added to an existing environment will only be used for encrypting disks of Data Hubs created after the CMK was added. Data Lake and FreeIPA disks and databases are not encrypted with the CMK.

Steps

For CDP UI

You can add an encryption key for an existing environment that does not have an encryption key set by navigating to the Summary page of the environment.

1. In the Cloudera Management Console, navigate to Environments and click on the environment for which you would like to set a CMK.
2. Click on the Summary tab.
3. Scroll down to the Customer Managed Encryption Key section.
4. If no CMK has been set, you will see a message stating that there is no customer-managed key enabled.
5. Click on the edit button in the top right corner of the tab.

6. Click on the toggle button next to Enable Customer-Managed Keys to enable adding a CMK.
7. Provide the encryption key ARN. You can copy and paste it from AWS KMS:

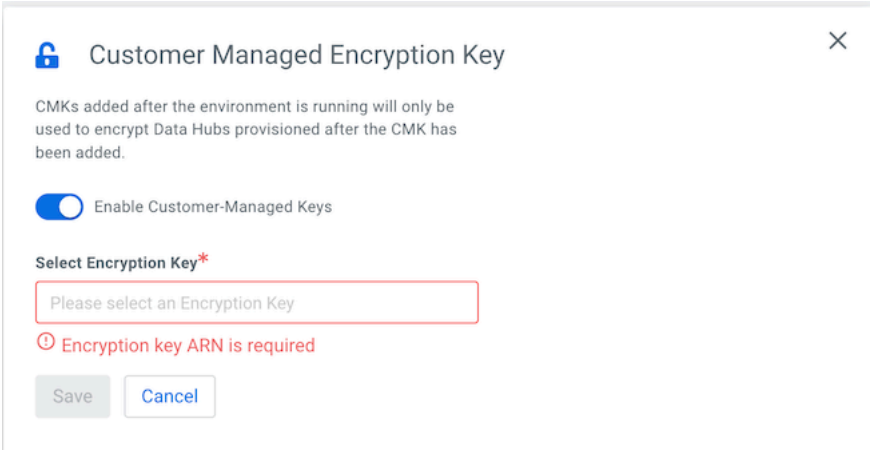

8. Click Save.

For CDP CLI

You can add an encryption key for an existing environment that does not yet have encryption enabled.

```
cdp environments update-aws-disk-encryption-parameters \
  --environment-name <ENVIRONMENT_NAME>
  --encryption-key-arn <CMK_ARN>
```

Deploying Cloudera in multiple AWS availability zones

By default, Cloudera provisions Data Lake, FreeIPA and Cloudera Data Hub clusters in a single AWS availability zone (AZ), but you can optionally choose to deploy them across multiple availability zones (multi-AZ). It is possible to enable it either for all or some of these components.

Single-AZ vs multi-AZ deployment

Each AWS region has multiple availability zones, which act as failure domains, preventing small outages from affecting entire regions. By default, a Cloudera environment (FreeIPA and Data Lake) and Cloudera Data Hub clusters running in it are deployed across one subnet, meaning that each of them has VMs spread across a single availability zone (as on AWS every subnet is related to a single availability zone). This is illustrated in the following diagram, where all of the resources run in the eu-central region, and each of them is in a different subnet and availability zone (eu-central-1a for the FreeIPA, eu-central-1b for the Data Lake, and eu-central-1c for the Cloudera Data Hub clusters):

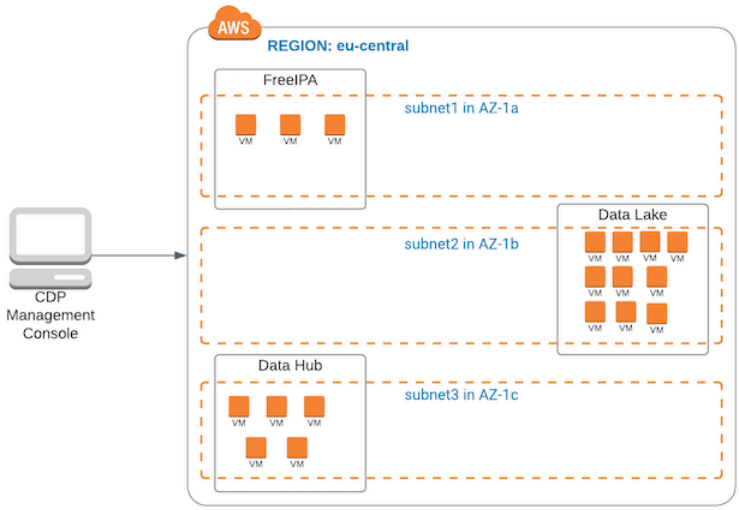

If you choose to deploy your Cloudera environment (FreeIPA and Data Lake) and Cloudera Data Hub clusters across multiple subnets and availability zones, each of these components is spread across three or more availability zones, providing high availability and fault tolerance. This is illustrated in the following diagram, where each of the components (FreeIPA, Data Lake, and Cloudera Data Hub clusters) is deployed across multiple subnets and multiple availability zones: eu-central-1a, eu-central-1b, and eu-central-1c:

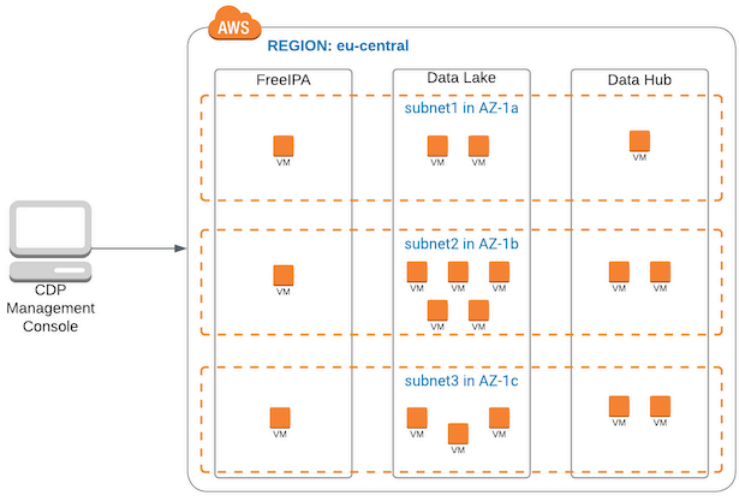

The Data Lake deployed in this setup is the enterprise Data Lake, which has enough nodes to utilize multiple availability zones for providing high availability and fault tolerance.

Multi-AZ can be enabled for Data Lake, FreeIPA, and Cloudera Data Hub clusters. It is possible to enable it either for all or some of these components.

Instances within a host group are distributed across availability zones equally, with the least used availability zone(s) preferred. For example, if there is a group of instances across three availability zones with 100 instances in AZ-1, 30 in AZ-2 and 30 in AZ-3, when an upscale of 10 nodes is requested, Cloudera provisions 5 instances in AZ-2 and 5 in AZ-3.

When a failure happens and a cluster needs to be repaired, the replacement VMs are always provisioned in the same subnet and availability zone as the old ones since the detached disks can only be reattached to a VM in the same availability zone. This means that if there is an availability zone outage, cluster repair is not possible.

Use cases

A multi-AZ Data Lake and FreeIPA constitute a resilient environment that provides a solid basis for multi-AZ Data Hubs and Cloudera data services. Data Hubs and Cloudera data services depend on the FreeIPA instance in the Data Lake to provide DNS resolution. Deploying FreeIPA across multiple availability zones ensures that critical DNS resolution is available in the event of an availability zone outage. Furthermore, an enterprise Data Lake provides high availability, and additional compute and memory resources for key SDX services and is recommended for production workloads.

Deploying your Cloudera Data Hub clusters across multiple availability zones is key if your mission-critical applications depend on HBase and Kafka. Multiple availability zone deployment for operational workloads is considered best practice by the cloud vendors. It ensures that your applications can continue to run in the event of a single availability zone outage.

When an entire availability zone fails, HBase automatically rebalances regions among the remaining instances in the cluster to maintain availability. The write-ahead log (WAL), which is replicated across the three availability zones is automatically replayed by the newly assigned region servers in other availability zones to ensure writes to the database are not lost.

When using the multi availability zone feature, Cloudera ensures that Kafka replicates partitions across brokers in different availability zones. During an availability zone failure this ensures that no data is lost and applications can continue to access the data they need. Cruise Control, which is deployed alongside every Kafka cluster in Cloudera Public Cloud, detects that topics need to be rebalanced to the remaining brokers. Once the availability zone is back online, you can repair your Kafka cluster, restoring the initial broker distribution across availability zones. Afterwards Cruise Control kicks in and ensures that all topic partitions are balanced across the cluster.

Limitations

The following limitations apply when deploying a multi-AZ Cloudera:

- When an AZ is down, you cannot create a new Cloudera Data Hub, and create or activate Cloudera data services within the environment. Existing workloads will continue to work.
- When an AZ is down, you cannot resize, stop, or restart Cloudera Data Hub clusters.

- When an AZ is down, existing running workloads will continue to run. Some Data Lake services will still be functioning (because they run across more than one AZ) but others may be down if the AZ where they are running is down.

  The following diagram shows how Data Lake services are distributed across AZs (assuming that there are 3 AZs per region). The Data Lake services marked in red run in a single AZ only and will be down if that AZ is down:

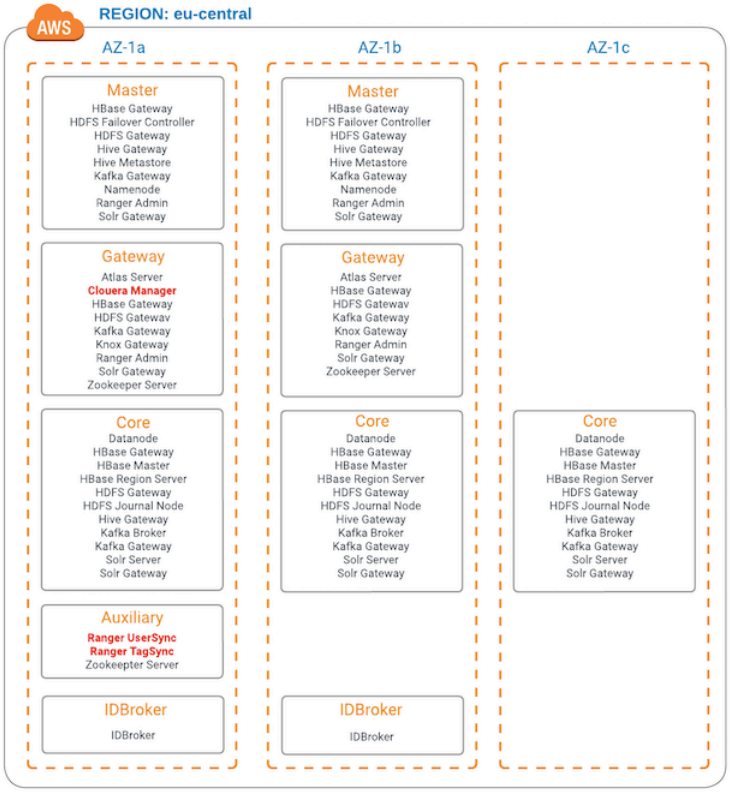

The following table shows which services aren't affected by an AZ outage (column 1 and 2) and those that may potentially be affected (column 3):

Services running in all available AZs	Services running in two AZs	Services running in a single AZ

Datanode	Atlas Server	Cloudera Manager
HBase Gateway	HDFS Failover Controller	Ranger UserSync
HBase Master	Hive Metastore	Ranger TagSync
HBase Region Server	IDBroker	
HDFS Gateway	Knox Gateway	
HDFS Journal Node	Namenode	
Hive Gateway	Ranger Admin	
Kafka Broker	Solr Gateway	
Kafka Gateway	Zookeeper Server	
Solr Server		

As shown in the diagram and the table, the following three services will be down if the AZ where they are running is down:

- Cloudera Manager - When Cloudera Manager is down, you cannot perform administrative tasks that you would normally perform via Cloudera Manager.
- Ranger UserSync - When Ranger UserSync is down, existing users are not impacted, but synchronizing new users to Ranger is disrupted. For example, if you add a new user and sync it to FreeIPA while Ranger UserSync is down, the newly added user will not be synced to Ranger and as a result the user will not be able to access workload data.
- Ranger TagSync - When Ranger TagSync is down, existing tags and policies are not impacted, but synchronizing new, updated, or removed tags with an external metadata service such as Apache Atlas and creating Ranger policies with these tags is disrupted.

- Single AZ environments or clusters cannot be converted to multi-AZ.
- While CDP CLI allows you to specify multiple subnets per Cloudera Data Hub cluster's host group, due to load balancer requirements only one subnet per AZ can be configured for the Cloudera Manager node group of the Cloudera Data Hub cluster.
- While by default Cloudera deploys Data Lake, FreeIPA, and Cloudera Data Hub clusters using CloudFormation, in case of the multi-AZ setup no CloudFormation is used. This affects vertical scaling: After vertical scaling (for example during cluster repair or upgrade) Cloudera always provisions the original instance types. For example, if you are using vertical scaling runbooks, repair and upgrade operations will use the previous instance type and root volume settings.

Enabling multi-AZ

Multi-AZ can be enabled for Data Lake, FreeIPA, and Cloudera Data Hub clusters running on AWS. It is possible to enable it either for all or some of these components.

To enable multi-AZ, you should register a multi-AZ Cloudera environment with an enterprise Data Lake. You can also create multi-AZ Cloudera Data Hub clusters within any existing environment. Detailed steps are provided below.

Register a multi-AZ environment

You can register a multi-AZ AWS environment via Cloudera UI or CDP CLI. You may choose to enable multi-AZ for Data Lake only or for FreeIPA only. There is no requirement to enable both.

Steps

For CDP UI

Register your environment as usual, just make sure to do the following:

1. On the Data Access and Data Lake Scaling page, select the enterprise Data Lake.
2. On the same page, scroll down and in the bottom of the page enable the Advanced Options.

3. In the Network and Availability section enable the Enable Multiple Availability Zones for Data Lake toggle button in order to enable multi-AZ for Data Lake. The option is disabled by default. The option only appears when an enterprise Data Lake is selected.
4. On the Region, Networking, and Security page, scroll down and in the bottom of the page enable the Advanced Options.
5. In the Network and Availability section enable the Enable Multiple Availability Zones for Data Lake toggle button in order to enable multi-AZ for FreeIPA. The option is disabled by default.

For CDP CLI

Register your AWS environment via CDP CLI as usual, but when creating a template for the environment registration commands, add the following to enable multi-AZ:

- In the environment creation command, add the multiAz=true to the --free-ipa option.
- In the Data Lake creation command, add the --multi-az option.
- In the Data Lake creation command, make sure to specify the enterprise Data Lake.

The following example illustrates these necessary additions, which are emphasized using bold typeface:

```
cdp environments create-aws-environment \
  --environment-name tb-multiaz-env \
  --credential-name tb-cldr-acc \
  --region "eu-central-1" \
  --security-access cidr=0.0.0.0/0 \
  --no-enable-tunnel \
  --authentication publicKeyId="test" \
  --log-storage storageLocationBase=s3a://cb-group/test,instanceProfile=ar
n:aws:iam::152413716728:instance-profile/mock-idbroker-admin-role \
  --vpc-id vpc-03e505817e3619238 \
  --subnet-ids subnet-013855b2gc32c2cd8 subnet-02b9054ef829374fe subnet-0
85c9ff36b38c0b35 \
  --free-ipa instanceCountByGroup=3,multiAz=true
```

```
cdp environments set-id-broker-mappings \
  --environment-name tb-multiaz-env \
  --data-access-role arn:aws:iam::152413716728:role/mock-idbroker-admin-
role \
  --ranger-audit-role arn:aws:iam::152413716728:role/mock-idbroker-admin-
role \
  --set-empty-mappings
```

```
cdp datalake create-aws-datalake \
  --datalake-name tb-multiaz-dl \
  --environment-name tb-multiaz-env \
  --cloud-provider-configuration instanceProfile=arn:aws:iam::15241371672
8:instance-profile/mock-idbroker-assumer,storageBucketLocation=s3a://cb-
group/test \
  --scale ENTERPRISE_HA \
  --runtime 7.2.12 \
  --no-enable-ranger-raz \
  --profile mowdev \
  --multi-az
```

Create multi-AZ Cloudera Data Hub clusters

If you would like to create multi-AZ Cloudera Data Hub clusters, see Creating a multi-AZ Data Hub on AWS.

Defining custom tags

In the Cloudera Management Console clusters user interface, you can define tenant-level or environment-level custom tags across all instances and resources provisioned in your organization’s cloud provider account.

Resource tagging

When you create an environment or other resources shared across your cloud provider account, Cloudera clusters automatically adds default tags to the Cloudera-created resources in your cloud provider account. You can also define additional custom tags that Cloudera applies to the cluster-related resources in your account.

You can use tags to protect the cloud resources of your Cloudera environment. Using the tags, you can exclude the resources that should not be removed during housekeeping or security deletion activities that can result in data corruption and data loss.

Default tags

By default, Cloudera applies certain tags to cloud provider resources whenever you create the resource, for example an environment.

Cloudera applies the following tags by default:

- Cloudera-Resource-Name: the workload-appropriate Cloudera resource name. For example, an IPA CRN for an IPA, a data lake CRN for a data lake, or a Cloudera Data Hub CRN for a Cloudera Data Hub cluster. This CRN serves as a unique identifier for the resource over time.
- Cloudera-Creator-Resource-Name: Cloudera resource name of the Cloudera user that created the resource.
- Cloudera-Environment-Resource-Name: name of the environment with which the resource is associated.

Custom tags

There are two types of custom tags that you can define in the Cloudera Management Console: tenant-level tags that apply to Cloudera-created resources across your organization's entire cloud provider account, and environment-level tags.

In the Cloudera Management Console user interface, you can define tenant-level tags across all instances and resources provisioned in your organization's cloud provider account. These resources include not only provisioned instances, but disks, networks, and other resources as well. In your cloud provider account you can search or filter on either the tag key or value. Tenant-level tags cannot be overridden during environment creation.

In addition to tenant-level tags, you can also define environment-level tags. Environment-level tags are inherited by the resources specific to an environment. For example, environment-level tags are inherited by the following resources:

- FreeIPA
- Data Lakes
- Cloudera Data Hub clusters
- Data Warehouses
- Cloudera Operational Databases

As with tenant-level tags, you can search or filter on the key tag or key value in your cloud provider account.

Note: Cloudera applies custom tags during creation of the resources. For example, you can only define environment-level tags during environment registration. If you want to add or update cloud provider resource tags, you must do so through the cloud provider API.

For more information about using tags on cloud provider resources, consult AWS, Azure, or Google Cloud documentation. It is your responsibility to ensure that your tags meet your cloud provider requirements.

Supported services

While some Cloudera services such as Cloudera Data Hub inherit environment-level tags, others require that you add tags when provisioning or enabling the data service. The following table shows how tags can be added for various Cloudera services:

Cloudera service	How to add tags
Data Lake	Inherits tenant or environment level tags.
FreeIPA	Inherits tenant or environment level tags.
Cloudera Data Engineering	Does not inherit tenant or environment level tags but you can define tags when enabling Cloudera Data Engineering.
Cloudera Data Hub	Inherits tenant or environment level tags and you can add more tags when creating a Cloudera Data Hub.
Cloudera Data Warehouse	Inherits tenant or environment level tags.
Cloudera DataFlow	Inherits tenant level tags and you can add more tags when enabling Cloudera DataFlow.
Cloudera AI	Does not inherit tenant or environment level tags but you can define tags when creating a Cloudera AI workbench.
Cloudera Operational Database	Inherits tenant or environment level tags and you can add more tags when creating a Cloudera Operational Database database via CLI.

Defining tenant-level tags

Required roles: PowerUser can define tags for the whole tenant.

- EnvironmentAdmin or Owner can set environment telemetry settings for a specific environment.

Steps

1. In the Cloudera Management Console, click Global Settings Tags .
2. Click Add.
3. Define both a key and a value for the tag. Both the key and the value must be between 4- 255 characters, with the following restrictions:

   Key

   Allowed characters are hyphens (-), underscores (_), lowercase letters, and numbers. Keys must not start with 'aws'. Keys must start with a lowercase letter and must not start or end with spaces.

   Value

   Allowed characters are hyphens (-), underscores (_), lowercase letters, and numbers. Values must not start or end with spaces. You can configure variables in the *{{{VARIABLENAME}}}* format. The following variables are supported for tenant-level tags:

   - *{{{CLOUDPLATFORM}}}* = AWS, AZURE or GCP
   - *{{{USERNAME}}}* = Cloudera username
   - *{{{USERCRN}}}* = Customer Resource Number (CRN) of Cloudera user
   - *{{{CREATORCRN}}}* = CRN of Cloudera resource creator
   - *{{{TIME}}}* = Actual time
   - *{{{ACCOUNTID}}}* = Cloudera account ID
   - *{{{RESOURCECRN}}}* = Generated string of Cloudera resource CRN
4. Click Add, and if necessary repeat the process for additional tags.

Note: Tenant-level tags are applied only to resources created after you define the tag. Any changes to the tag value do not propagate to existing resources that have the tag.

Defining environment-level tags

You define environment-level tags during environment registration.

Required roles: EnvironmentCreator can set tags for a specific environment during environment registration.

Steps

1. In the Cloudera Management Console, click Environments Register Environment .
2. Proceed through the environment registration steps.
3. After you define data access, add any environment-level tags by clicking Add and defining the tag key and value.

Related Information

Tagging your Amazon EC2 resources (AWS)

Updating instance metadata to IMDSv2

Cloudera can use IMDSv2 or IMDSv1 for accessing EC2 instance metadata from a running instance.

Cloudera currently uses IMDSv2 for accessing EC2 instance metadata on all newly created Data Lakes, FreeIPA clusters, and Cloudera Data Hub clusters, as long as an IMDSv2-compatible image is used. Prior to Cloudera supporting IMDSv2, Data Lakes, FreeIPA clusters, and Cloudera Data Hub clusters used IMDSv1; These clusters created with IMDSv1 can now be updated to IMDSv2 as long as an IMDSv2-compatible image was used to create the cluster.

Update an existing cluster to IMDSv2

You can update an existing Data Lake, FreeIPA cluster and Cloudera Data Hub cluster that is currently using IMDSv1 to IMDSv2. This is a zero downtime operation and does not disrupt any existing processes or jobs

Prerequisites

The following prerequisites needs to be met:

- Image must be compatible with IMDSv2

  You can only update to IMDSv2 if the image used for creating the cluster is compatible with IMDSv2. If the image is not compatible or if a cluster is already using IMDSv2, the Update to IMDSv2 button is grayed out:

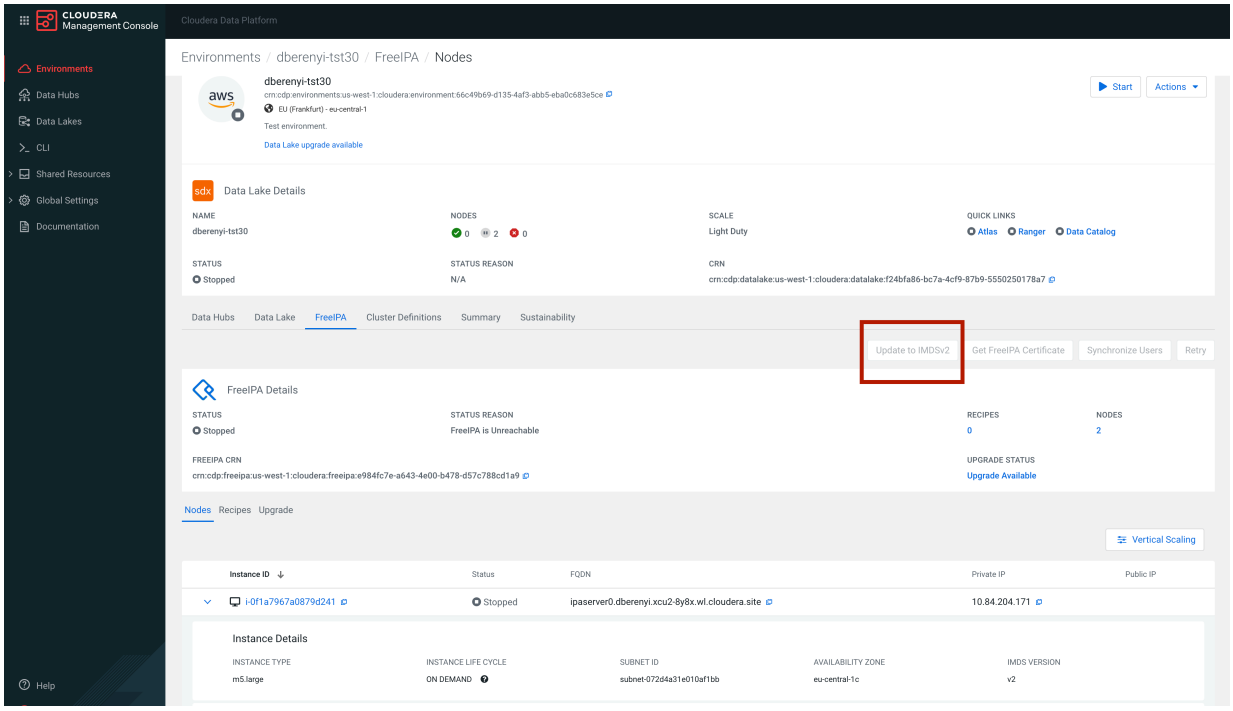

  If you would like to check image compatibility manually, see Checking if cluster image is compatible with IMDSv2 on page 59.

- You may need to update recipes.

  If you are using recipes, you need to update them first to ensure that they are compatible with IMDSv2. See:

  - Update FreeIPA recipes
  - Update Data Lake recipes
  - Update Data Hub recipes

  For a quick check, you can search for usage of "169.254.169.254" IP in the recipe content (as this is the IP pointing to the AWS IMDS). You can find an example in the Retrieve instance metadata documentation. You may also want to review How Instance Metadata Service Version 2 works.

Required roles: EnvironmentAdmin or Owner of the environment

Steps

For CDP UI

1. In the Cloudera Management Console, navigate to Data Lake, FreeIPA or Data Hub details.
2. Navigate to Nodes.

3. The option to Update to IMDSv2 is available as follows:

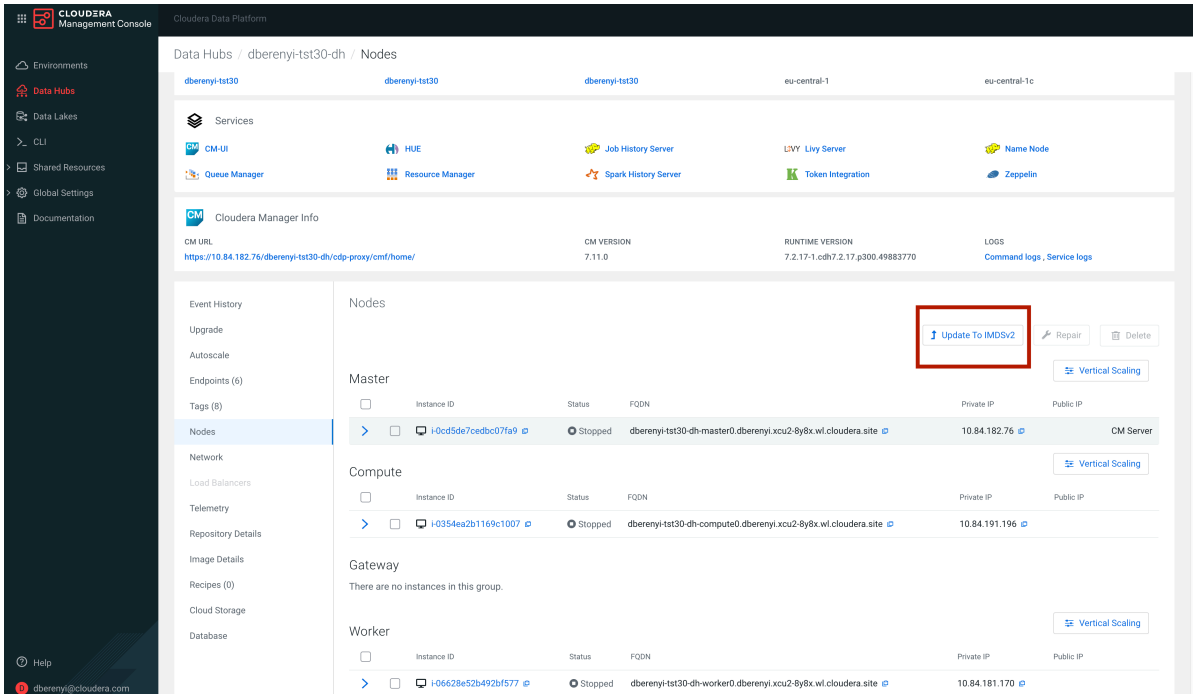

4. The update should happen within a few seconds. You can track the status in event history.

If you would like to verify that the update happened correctly, navigate to one of the EC2 instances in your AWS console.

Once the update is complete, the Update to IMDSv2 button is grayed out.

For CDP CLI

Steps CLI

Use the following commands to update a cluster from IMDSv1 ro IMDSv2:

Data Lake:

```
cdp datalake update-to-aws-imds-v2 --crn <SPECIFY_CRN>
```

FreeIPA:

```
cdp environments update-freeipa-to-aws-imds-v2 --environment-crn <SPECIF
Y_CRN>
```

Cloudera Data Hub:

```
cdp datahub update-to-aws-imds-v2 --crn <SPECIFY_CRN>
```

The update should happen within a few seconds.

Checking if cluster image is compatible with IMDSv2

To check if the image is compatible, follow these steps:

Steps

For CDP UI

1. In the Cloudera Management Console, navigate to your cluster.
2. Navigate to the Image details tab.
3. Click on the image ID.
4. Scroll down to Package Versions details of the image:

5. Package versions include a variable called “imds”. Find this variable and ensure that its value is “v2

For CDP CLI

1. You can use the describe CDP CLI command of the given cluster (FreeIPA, Data Lake, Cloudera Data Hub. The commands are as follows:
   - cdp datahub describe-cluster
   - cdp datalake describe-datalake
   - cdp environment describe-environment
2. The response should contain image and image catalog JSON details.

   Here is an example from image catalog where an image is compatible (note the "package-versions" section, which includes “imds” version. If the value is “v2”, the image is compatible. For example, see the highlighted “imds” section in the following image catalog file:

```
{
        "created": 1709942743,
        "published": 1709951597,
        "date": "2024-03-09",
        "description": "Official Cloudbreak image",
        "images": {
          "aws": {
            "eu-central-1": "ami-0faa3e25092764091",
            "us-west-1": "ami-043b94655fb95edf1",
            "us-west-2": "ami-0f9ed7da8f775d00e"
          }
        },
        "os": "redhat8",
        "os_type": "redhat8",
        "uuid": "9b84a914-a15a-4856-94a9-3cfda722b0b8",
        "package-versions": {
          "blackbox-exporter": "0.19.0",
```

```
          "cdp-logging-agent": "1.3.2_b1",
          "cdp-minifi-agent": "1.22.07",
          "cdp-prometheus": "2.36.2",
          "cdp-request-signer": "0.2.4",
          "cdp-telemetry": "1.3.2_b1",
          "cloudbreak_images": "8cf7cd165b58343011091c4908f9f796d4dceb
92",
       "imds": "v2",
          "inverting-proxy-agent": "3.0.7-b1",
          "inverting-proxy-agent_gbn": "41924420",
          "java": "8",
          "java11": "11.0.21",
          "java17": "17.0.9",
          "java21": "21.0.2",
          "java8": "1.8.0_392",
          "metering_agent": "2.0.0",
          "node-exporter": "1.3.1",
          "psql": "14",
          "psql11": "11.22",
          "psql14": "14.11",
          "python36": "3.6.8-38.module+el8.5.0+12207+5c5719bc",
          "python38": "3.8.16-1.module+el8.8.0+18967+20d359ae.1",
          "python39": "3.9.16-1.module+el8.8.0+20025+f2100191.2",
          "salt": "3001.8",
          "salt-bootstrap": "0.13.6-2022-05-20T08:57:17",
          "source-image": "ami-039ce2eddc1949546"
        },
        "tags": {
          "fips-mode": "disabled",
          "hardening": "cis_server_l1"
        }
      },
```

Restricting access for Cloudera services that create their own security groups on AWS

The security groups that you select to use during environment registration are only used for the Data Lake, FreeIPA, Cloudera Data Hub clusters, and Cloudera Operational Databases running in that environment. The Kubernetes-based Cloudera services (Cloudera Data Engineering, Cloudera DataFlow, Cloudera Data Warehouse, and Cloudera AI) create their own security groups with rules that should be restricted separately.

The following table explains where and when you can restrict these rules:

Note: If you do not restrict these endpoints, Cloudera defaults to opening access to all (0.0.0.0/0).

Cloudera service	Type of access that can be restricted	When and where to restrict	Link to related documentation
Cloudera DataFlow	Admin access to the Kubenetes API Server endpoint can be restricted. End user access can be restricted.	Restrict admin access to Kubernetes endpoints during or after enabling Cloudera DataFlow via the Kubernetes API Server Endpoint Access setting. Restrict end user access to the the Cloudera DataFlow endpoints during or after enabling Cloudera DataFlow via the Load Balancer Endpoint Access setting. .	Enabling DataFlow for an environment Managing Kubernetes API Server user access
Cloudera Data Engineering	Admin access to Kubenetes endpoints can be restricted. End user access can only be restricted manually from the AWS management console.	Restrict admin access to Kubernetes endpoints during enabling Cloudera Data Engineering via the Whitelist IPs parameter. Restrict end user access manually from the AWS management console.	Enabling Cloudera Data Engineering and Limiting Incoming Endpoint Traffic for Data Engineering Services
Cloudera Data Warehouse	Both admin access to Kubernetes endpoints and end user access are always set to the same range that can be set in environment activation settings. While the access to the Kubernetes endpoints is a combination of the Cloudera Control Plane's CIDR and your CIDR provided in environment activation settings, the access to the end user access points (JDBC, UI) is only your CIDR provided in environment activation settings.	In Cloudera Data Warehouse environment's activation settings.	Restricting access to endpoints in AWS environments and Editing the IP CIDRs in the trusted list for endpoints in AWS environments
Cloudera AI	There are two separate options, one for admin access to Kubernetes endpoints and another for end user access.	During Cloudera AI workbench provisioning, under Network Settings: • The Load Balancer Source Ranges parameter can be used to restrict end user access. • Selecting the checkmark Restrict access to Kubernetes API server to authorized IP ranges allows you to restrict admin access to Kubernetes endpoints.	Provisioning Cloudera AI workbenches

Configure lifecycle management for logs on AWS

To avoid unnecessary costs related to Amazon S3 cloud storage, you should create lifecycle management rules for your cloud storage bucket used by Cloudera for storing logs so that these logs get deleted once they are no longer useful.

Some examples of Cloudera logs stored in cloud storage are: cloudera server logs, cloudera agent logs, autossh logs, freeipa logs, ranger audit logs, datahub services logs, datalake logs, cm management services logs, and so on. These logs are mostly useful for troubleshooting, so they can be periodically deleted.

AWS allows you to set up lifecycle management rules for your S3 buckets. For example, you can set a specified expiration period for a cloud storage location so that the files in that location get deleted automatically on a scheduled basis. Cloudera recommends that you do this for the cloud storage location that you provided to Cloudera for log storage.

Consider the following when setting up lifecycle management rules:

- As logs and data locations may overlap with each other (in case the same bucket or container is used for both), ensure to use the correct path prefixes in order to delete only the logs. The prefixes are listed below.
- When setting an expiration period, consider how long you would like to keep the logs to allow enough time for troubleshooting. For example, in case your Data Lake, FreeIPA or Cloudera Data Hub cluster is ever down, you should be able to access the logs for troubleshooting.

Prefixes based on AWS environment's logs location base

Prior to creating lifecycle management rules in S3, review this information to ensure that you use the correct path.

Note:

Path logic changed in February 2021. Starting in February 2021, the path automatically contains the cluster-logs folder as a peer of the cluster-backups folder, creating a better structural separation between logs and backups.

	The "cluster-logs" prefix is automatically generated if a bucket name without any subdirectories is used as logs location	The "cluster-logs" prefix is automatically generated if subdirectories are provided	If your environment was registered prior to February 2021: If you defined a sub-directory, then that subdirectory is used instead of "cluster-logs"
Logs location provided during environment registration	**s3a://my-bucket/**	**s3a://my-bucket/my-dl**	**s3a://my-bucket/my-dl**
FreeIPA prefix for lifecycle rule	cluster-logs/freeipa	my-dl/cluster-logs/freeipa	my-dl/freeipa
DataLake prefix for lifecycle rule	cluster-logs/datalake	my-dl/cluster-logs/datalake	my-dl/datalake
DataHub prefix for lifecycle rule	cluster-logs/datahub	my-dl/cluster-logs/datahub	my-dl/datahub

Creating expiration rules for an S3 bucket via AWS CLI

If you would like to create expiration rules via AWS CLI, refer to the following examples of JSON files in order to create your own JSON file.

If the log base location is s3://mybucket, create a JSON file similar to the following:

```
{
 "Rules": [
  {
     "Filter": {
         "Prefix": "cluster-logs/"
      },
     "Status": "Enabled",
     "AbortIncompleteMultipartUpload": {
         "DaysAfterInitiation": 7
      },
    "NoncurrentVersionExpiration": {
        "NoncurrentDays": 7
     },
    "Expiration": {
        "Days": 7
     },
     "ID": "cleanup cloud storage data after 7 days"

  }]
```

```
}
```

If the log base location is s3://mybucket/mypath and your environment was registered in February 2021 or later, create a JSON file similar to the following:

```
{
 "Rules": [
  {
     "Filter": {
         "Prefix": "mypath/cluster-logs/"
      },
     "Status": "Enabled",
     "AbortIncompleteMultipartUpload": {
         "DaysAfterInitiation": 7
      },
    "NoncurrentVersionExpiration": {
        "NoncurrentDays": 7
     },
    "Expiration": {
        "Days": 7
     },
     "ID": "cleanup cloud storage data after 7 days"
  }]
}
```

If the log base location is s3://mybucket/mypath and your environment was registered prior to February 2021, create a JSON file similar to the following:

Note: Multiple rules are needed in this case to ensure that the cluster-backups folder (which is also located in "mypath") does not get deleted. Starting in February 2021, this is no longer needed because the cluster-backups folder is always a peer of the cluster-logs folder.

```
{
 "Rules": [
  {
     "Filter": {
         "Prefix": "mypath/freeipa"
      },
     "Status": "Enabled",
     "AbortIncompleteMultipartUpload": {
         "DaysAfterInitiation": 7
      },
    "NoncurrentVersionExpiration": {
        "NoncurrentDays": 7
     },
    "Expiration": {
        "Days": 7
     },
     "ID": "cleanup cloud storage data after 7 days (freeipa)"
  },
  {
     "Filter": {
         "Prefix": "mypath/datalake"
      },
     "Status": "Enabled",
     "AbortIncompleteMultipartUpload": {
         "DaysAfterInitiation": 7
      },
    "NoncurrentVersionExpiration": {
        "NoncurrentDays": 7
     },
    "Expiration": {
        "Days": 7
```

```
      },
      "ID": "cleanup cloud storage data after 7 days (datalake)"

  },
{
      "Filter": {
          "Prefix": "mypath/datahub"
       },
      "Status": "Enabled",
      "AbortIncompleteMultipartUpload": {
          "DaysAfterInitiation": 7
       },
     "NoncurrentVersionExpiration": {
         "NoncurrentDays": 7
      },
     "Expiration": {
         "Days": 7
      },
      "ID": "cleanup cloud storage data after 7 days (datahub)"
  }
]
}
```

Once you have the JSON file ready, post it with an AWS CLI command similar to:

```
aws s3api put-bucket-lifecycle-configuration /
--bucket mybucket /
--lifecycle-configuration file:///my/path/rules.json
```

If you have existing rules on the nodes, you can use the get-bucket-lifecycle-configuration command first, edit that JSON (adding the new rules there), and then post the updated JSON.

Creating expiration rules for an S3 bucket via S3 console

If you would like to create expiration rules via S3 console, navigate to your S3 bucket > Management tab > Create lifecycle rule.

The following screenshots show an example setup for a single prefix:

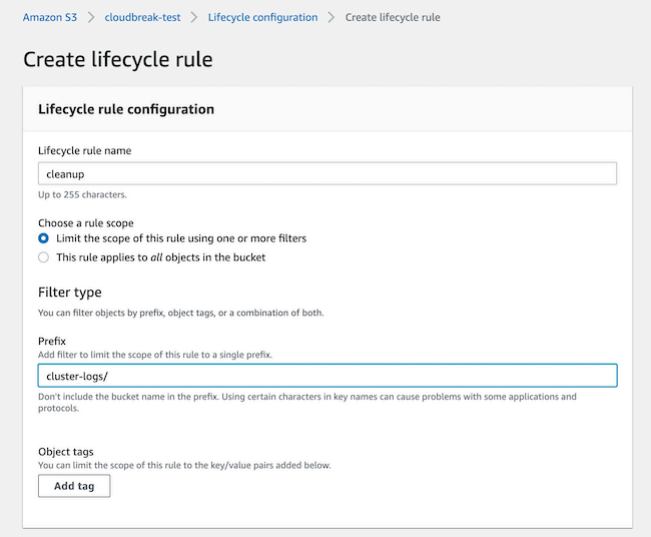

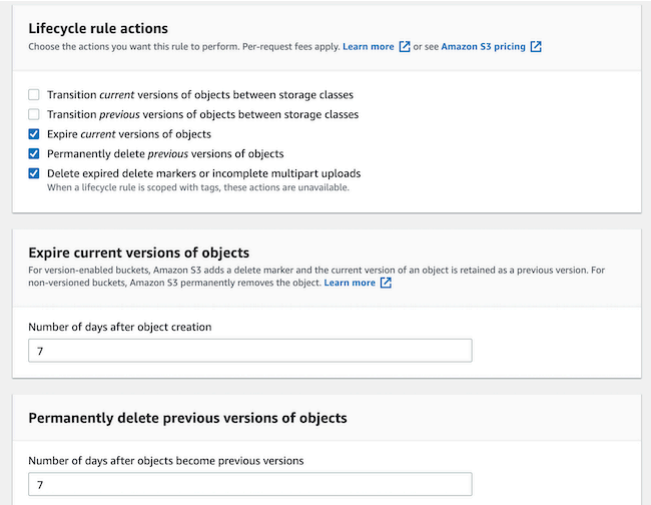

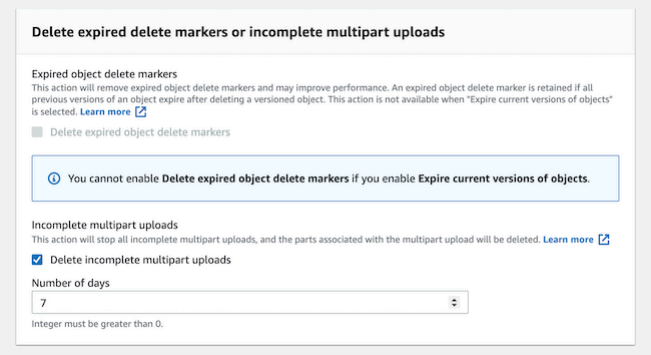

Related Information

Managing your storage lifecycle

Amazon S3 – Object Expiration

Enabling environment telemetry

Troubleshooting for RAZ-enabled AWS environment

This section includes common errors that might occur while using a RAZ-enabled AWS environment and the steps to resolve the issues.

Cloudera Management Console

Azure Environments

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Working with Azure environments 5

Introduction to Azure environments 5

Register environment (UI) 6

Register environment (CLI) 11

Enabling admin and user access 11

Obtain CLI commands 12

Understanding environment UI options 13

Using Compute Clusters 15

Enabling default Compute Cluster for new environments 15
Enabling default Compute Cluster for existing environments 18
Adding more Compute clusters 19
Managing Compute Clusters 20

Monitoring an environment 21

Environment status options 22

Stop and restart an environment 24

Delete an environment 25

Cleaning up a failed environment 26

Add subnets to an environment 26

Add security groups 27

Change environment's credential 27

Enabling environment telemetry....28
Defining anonymization rules....30

Adding a customer managed encryption key to a Cloudera environment running on Azure....31
Azure prerequisites for using a CMK....32
Register an Azure environment with a CMK....32
Create a Cloudera Data Hub on Azure with a CMK....33
Check Azure environment's CMK....33
Check Cloudera Data Hub's CMK....34
Set a CMK for an existing Azure environment....35

Defining custom tags....36

Using Azure Database for PostgreSQL Flexible Server....38
Azure prerequisites for Flexible Server....39
Enabling Flexible Server during Azure environment creation....42
Enabling Flexible Server during Cloudera Data Hub cluster creation....46
Enabling Flexible Server with Private Link on an existing environment....48
Upgrading Azure Single Server to Flexible Server....50
Upgrading databases using the Cloudera UI....51
Upgrading databases using the CDP CLI....54
Post-requisites after upgrading databases....55
Configuring a CMK for data encryption in Azure Database for PostgreSQL Flexible Server....55
Enabling Single Server....57
Troubleshooting Flexible Server....58

Enabling a private endpoint for Azure Postgres....59

Deploying Cloudera in multiple Azure availability zones....61

Restricting access for Cloudera services....65

Configure log lifecycle management....66

Working with Azure environments

Refer to the following documentation to learn about creating and managing Azure environments in Cloudera:

Related Information

Managing provisioning credentials for Azure
Managing Data Lakes
Managing FreeIPA

Introduction to Azure environments

In Cloudera, an environment is a logical subset of your cloud provider account including a specific virtual private network. You can register as many environments as you require.

The “environment” concept of Cloudera is closely related to the virtual private network in your cloud provider account. Registering an environment provides Cloudera with access to your cloud provider account and identifies the resources in your cloud provider account that Cloudera services can access or provision. A single environment is contained within a single cloud provider region, so all resources deployed by Cloudera are deployed within that region within one specific virtual network. Once you’ve registered an environment in Cloudera, you can start provisioning Cloudera resources such as clusters, which run on the physical infrastructure in an Azure data center.

The following diagram enumerates the components of an environment:

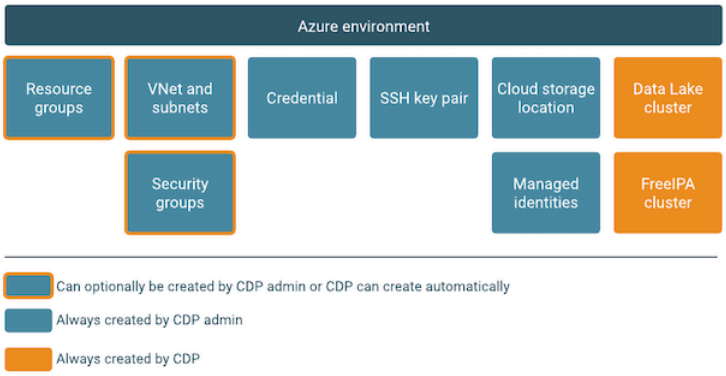

The diagram illustrates all major user-created and Cloudera-created components of an environment:

- The items in dark blue boxes with orange outlines can either be automatically provisioned by Cloudera on your Azure account, or you can optionally pre-create them and specify them when registering an environment.
- The items in dark blue boxes must be pre-created by your Cloudera administrator prior to environment registration and then specified when registering an environment.
- The items in orange boxes are automatically provisioned on Azure by Cloudera as part of environment provisioning.

Note: The items that are user-created don’t get terminated during environment deletion.

As shown in the diagram, an environment consists of the following resources:

Environment component	Description
Virtual network with subnets	An environment corresponds to one specific virtual network (called VNet on Azure) and subnets in which Cloudera resources are provisioned.
Security groups	Security groups (called NSGs on Azure) act as a virtual firewall for your instances to control inbound and outbound traffic. All VM instances provisioned within an environment use your specified security access settings allowing inbound access to your instances from your organization's computers.
Credential	Cloudera uses the credential for authorization to provision resources (such as compute instances) within your cloud provider account. On Azure, credential creation involves creating an application and a service principal within the Azure Active Directory manually (app-based credential).
SSH public key	When registering an environment on a public cloud, a Cloudera administrator provides an SSH public key. This way, the administrator has root-level access to the Data Lake instance and Cloudera Data Hub cluster instances.
Cloud storage location and managed identities	When registering an environment, you must provide an ADLS Gen2 location for storing: • All workload cluster data • Cluster service logs and Ranger audits Furthermore, you must create and assign managed identities so that Cloudera can access the storage location.
Data Lake	A data lake is automatically provisioned when an environment is created. It provides a mechanism for storing, accessing, organizing, securing, and managing data.
FreeIPA	A FreeIPA server is automatically provisioned when an environment is created. It is responsible for synchronizing your users and making them available to Cloudera services, Kerberos service principal management, and more.
Resource groups	Cloudera creates multiple new resource groups for resources that it deploys on your Azure account.

You may want to register multiple environments corresponding to different regions that your organization would like to use. Once your environment is running, you can provision Cloudera Data Hub clusters, Cloudera Data Warehouses, and other resources in it.

Register an Azure environment from the Cloudera UI

Once you've met the Azure cloud provider requirements, register your Azure environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in Azure requirements.

Required role: EnvironmentCreator

Steps

1. Navigate to the Cloudera Management Console > Environments > Register environment:
2. On the Register Environment page, provide the following information:

Parameter	Description
General Information	

Parameter	Description
Environment Name (Required)	Enter a name for your environment. The name: • Must be between 5 and 28 characters long. • Can only include lowercase letters, numbers, and hyphens. • Must start with a lowercase letter.
Description	Enter a description for your environment.
Select Cloud Provider (Required)	Select Azure.
Microsoft Azure Credential (Required)	
Select Credential	Select an existing credential or select Create new credential. For instructions on how to create a credential, refer to Create an app-based credential.

3. Click Next.
4. On the Data Access and Data Lake Scaling page, provide the following information:

Parameter	Description
Data Lake Settings	
Data Lake Name (Required)	Enter a name for the Data Lake cluster that will be created for this environment. The name: • Must be between 5 and 100 characters long • Must contain lowercase letters • Cannot contain uppercase letters • Must start with a letter • Can only include the following accepted characters are: a-z, 0-9, -.
Data Lake Version (Required)	Select Cloudera Runtime version that should be deployed for your Data Lake. The latest stable version is used by default. All Cloudera Data Hub clusters provisioned within this Data Lake will be using the same Cloudera Runtime version.
Fine-grained access control on ADLS Gen2	
Enable Ranger authorization for ADLS Gen2 Identity	If you would like to use Fine-grained access control, enable this option and then select the Ranger RAZ managed identity created in the Minimal setup for cloud storage.
Data Access and Audit	
Assumer Identity (Required)	Select the Assumer managed identity created in Minimal setup for cloud storage.
Storage Location Base (Required)	Provide the ADLS Gen2 location created for data storage in Minimal setup for cloud storage.
Data Access Identity (Required)	Select the Data Lake Admin managed identity created in Minimal setup for cloud storage.
Ranger Audit Identity (Required)	Select the Ranger Audit managed identity created in Minimal setup for cloud storage.
IDBroker Mappings	We recommend that you leave this out and set it up after registering your environment as part of Onboarding Cloudera users and groups for cloud storage. **Note:** If you are using Fine-grained access control, this option is disabled, because you should onboard your users and groups via Ranger instead of using IDBroker mappings.
Scale (Required)	Select Data Lake scale. By default, “Light Duty” is used. For more information on data lake scale, refer to Data Lake scale.
Enable **Compute Cluster**	Enable Compute Clusters if you would like to deploy a containerized platform on Kubernetes for data services and shared services.

5. Click on Advanced Options to make additional configurations for your Data Lake. The following options are available:

Parameter	Description
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Sizes for virtual machines in Azure.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on a specific Data Lake host group. For more information, see Recipes.

6. Click Next.
7. On the Region, Networking and Security page, provide the following information:

Parameter	Description
Region	
Select Region (Required)	Select the region that you would like to use for accessing and provisioning resources from Cloudera. If you would like to use a specific existing virtual network, the virtual network must be located in the selected region.
Resource Group	
Select Resource Group (Required)	You have two options: • Select one existing resource group. If you select this, all Cloudera resources will be provisioned into that resource group. • Select Create new resource groups to have Cloudera create multiple resource groups.
Customer Managed Encryption Keys	
Enable Customer-Managed Keys	Enable this if you would like to provide a Customer-Managed Key (CMK) to encrypt environment's disks and databases. For more information, refer to Customer managed encryption keys.
Select Encryption Key Resource Group	Select the resource group where the CMK is located.
Encryption key URL	Provide the URL of the key value where the CMK resides. This is the same as the key identifier that you can copy directly from Azure Portal.
Managed identity for encryption	If using Azure Database for PostgreSQL Flexible Server, you can optionally select a managed identity created for encrypting it. For more information, refer to Managed identity for encrypting Azure Database for PostgreSQL Flexible Server.
Network	
Select Network (Required)	You have two options: • Select the existing virtual network where you would like to provision all Cloudera resources. Refer to VNet and subnets. • Select Create new network to have a new network with three subnets created.
Select Subnets (Required)	This option is only available if you choose to use an existing network. Multiple subnets must be selected and Cloudera distributes resources evenly within the subnets.
Network CIDR (Required)	This option is only available if you select to create a new network. If you selected to create a new network, provide Network CIDR that determines the range of private IPs that VMs will use. This must be a valid private IP CIDR IP in IPv4 range. For example 10.10.0.0/16 are valid IPs. /16 is required to allow for enough IP addresses.
Create Private Subnets	This option is only available if you select to have a new network and subnets created. Is is turned on by default so that private subnets are created in addition to public subnets. If you disable it, only public subnets will be created. **Important:** For production deployments, Cloudera recommends that you use private subnets. Work with your internal IT teams to ensure that users can access the browser interfaces for cluster services.

Parameter	Description
Enable Public Endpoint Access Gateway	When Cluster Connectivity Manager is enabled, you can optionally enable Public Endpoint Access Gateway to provide secure connectivity to UIs and APIs in Data Lake and Cloudera Data Hub clusters deployed using private networking. If you are using your existing VPC, under Select Endpoint Access Gateway Subnets, select the public subnets for which you would like to use the gateway. The number of subnets must be the same as under Select Subnets and the availability zones must match. For more information, refer to Public Endpoint Access Gateway documentation.
Create Private Endpoints	By default, the PostgreSQL Azure database provisioned for your Data Lake is reachable via a service endpoint (public IP address). To increase security, you can optionally select to have it reachable via a private endpoint instead of a service endpoint. **Note:** This option is only available if an existing resource group is selected. **Note:** Only the subnets that have Azure private endpoint network policies turned off are eligible for private endpoint creation. At least one such subnet is required. If you select to create a private endpoint and you are using your own VNet, you have two options: • Select “Create new private DNS zone” and Cloudera creates and manages a private DNS zone for you in the provided existing resource group. • Select your existing private DNS zone. If you select to create a private endpoint and you would like for Cloudera to create a new VNet, Cloudera creates a private DNS zone for you. For more information, refer to Private endpoint for Azure Postgres.
Create Public IPs	This option is disabled by default when Cluster Connectivity Manager is enabled and enabled by default when Cluster Connectivity Manager is disabled.
Flexible Server	During environment registration in Cloudera, the Flexible Server in public service mode is used by default, but you can specify to use the Flexible Server in private service mode (“Flexible Server with Private Link” or "Flexible Server with Delegated Subnet (deprecated)"). For more information, refer to Using Azure Database for PostgreSQL Flexible Server.
Proxies	
Select Proxy Configuration	Select a proxy configuration if previously registered. For more information refer to Setting up a proxy server.
Security Access Settings	
Select Security Access Type (Required)	This determines inbound security group settings that allow connections to the Data Lake and Cloudera Data Hub clusters from your organization’s computers. You have two options: • Create new security groups - Allows you to provide custom CIDR IP range for all new security groups that will be created for the Data Lake and Cloudera Data Hub clusters so that users from your organization can access cluster UIs and SSH to the nodes. This must be a valid CIDR IP in IPv4 range. For example: 192.168.27.0/24 allows access from 192.168.27.0 through 192.168.27.255. You can specify multiple CIDR IP ranges separated with a comma. For example: 192.168.27.0/24,192.168.28.0/24. If you use this setting, several security groups will get created: one for each Data Lake host group the Data Lake and one for each host group), one for each FreeIPA host group, and one for RDS; Furthermore, the security group settings specified will be automatically used for Cloudera Data Hub, Data Warehouse, and Cloudera AI clusters created as part of the environment. • Provide existing security groups (Only available for an existing VPC) - Allows you to select two existing security groups, one for Knox-installed nodes and another for all other nodes. If you select this option, refer to Security groups to ensure that you open all ports required for your users to access environment resources.
Kubernetes	

Parameter	Description
Enable **Private Kubernetes Cluster** or provide **Authorized IP Ranges**	If you have enabled Compute Clusters, you have the following options to configure the necessary networking information for the Kubernetes cluster: • Enable Private Kubernetes Cluster to create a private cluster that blocks all access to the API Server endpoint. • Provide the CIDRs to the Kubernetes API Server Authorized IP Ranges field to specify a set of IP ranges that will be allowed to access the Kubernetes API server. You need to provide the advanced configurations only once when creating your environment. The configurations will be applied to all compute clusters in the environment.
Enable **User Defined Routing**	Enable User Defined Routing (UDR) in case public IPs are blocked for egress.In case you enable UDR, you must select the specific worker node subnet where the UDR is configured.
AKS Private DNS Zone ID	When selecting **Private Kubernetes Cluster**, you also need to select an existing private DNS zone or select creating a new private DNS zone by Cloudera on your Azure account for the database.
Worker Node Subnets	Uses the same set of subnets provided in **Network** section. You have the option to not use all of the previously provided subnets.
SSH Settings	
New SSH public key (Required)	Upload a public key directly from your computer. **Note:** Cloudera does not use this SSH key. The matching private key can be used by your Cloudera administrator for root-level access to the instances provisioned for the Data Lake and Cloudera Data Hub.
Add tags	You can optionally add tags to be created for your resources on Azure. Refer to Defining custom tags.

8. Click on Advanced Options to make additional configurations for FreeIPA. The following options are available:

Parameter	Description
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Sizes for virtual machines in Azure.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on FreeIPA nodes. For more information, see Recipes.

9. Click Next.
10. On the Storage page, provide the following information:

Parameter	Description
Logs	
Logger Identity (Required)	Select the Logger managed identity created in Minimal setup for cloud storage.
Logs Location Base (Required)	Provide the ADLS Gen2 location created for log storage in Minimal setup for cloud storage.
Backup Location Base	Provide the ADLS Gen2 location created for FreeIPA and Data Lake backups in Minimal setup for cloud storage. If not provided, the default Backup Location Base uses the Logs Location Base.
Telemetry	
Enable Workload Analytics	Enables Cloudera Observability support for workload clusters created within this environment. When this setting is enabled, diagnostic information about job and query execution is sent to Cloudera Observability. For more information, refer to Enabling workload analytics and logs collection.
Enable Deployment Cluster Logs Collection	When this option is enabled. the logs generated during deployments will be automatically sent to Cloudera. For more information, refer to Enabling workload analytics and logs collection.

11. Click on Register Environment to trigger environment registration.

12. The environment creation takes about 60 minutes. The creation of the FreeIPA server and Data Lake cluster is triggered. You can monitor the progress from the web UI. Once the environment creation has been completed, its status will change to “Running”.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Register an Azure environment from the CDP CLI

Once you’ve met the Azure cloud provider requirements, register your Azure environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in Azure requirements.

Required role: EnvironmentCreator

Steps

Unlike in the Cloudera web interface, in CDP CLI environment creation is a three-step process with environment creation, setting IDBroker mappings and Data Lake creation being three separate steps. The easiest way to obtain the correct commands is to provide all parameters in the Cloudera web interface and then generate the CDP CLI commands on the last page of the wizard. For detailed steps, refer to Obtain CLI commands for registering an environment.

To learn more about how to create Compute Cluster enabled environments with CLI, see Enabling default Compute Cluster for new environments.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Enabling admin and user access to environments

In order to grant admin and user access to an environment that you registered in Cloudera, you should assign the required roles.

You need to be an EnvironmentCreator in order to register an environment. Once an environment is running, the following roles can be assigned:

- EnvironmentAdmin - Grants all rights to the environment and Cloudera Data Hub clusters running in it, except the ability to delete the environment. The user who registers the environment automatically becomes its EnvironmentAdmin.
- EnvironmentUser - Grants permission to view Cloudera Data Hub clusters and set the workload password for the environment. This role should be used in conjunction with service-specific roles such as DataHubAdmin, DWAdmin, DWUser, MLAdmin, MLUser, and so on. When assigning one of these service-specific roles to users, make sure to also assign the EnvironmentUser role.
- DataSteward - Grants permission to perform user/group management functions in Ranger and Atlas Admin, manage ID Broker mappings, and start user sync for the environment.
- Owner - Grants the ability to mange the environment in Cloudera, including deleting the environment. The user who registers the environment automatically becomes its Owner. The Owner role does not grant access the environment's clusters (Data Lakes, Cloudera Data Hub clusters).

The roles are described in detail in Resource roles. The steps for assigning the roles are described in Assigning resource roles to users and Assigning resource roles to groups.

Related Information

Resource roles

Assigning resource roles to users

Assigning resource roles to groups

Obtain CLI commands for registering an environment

Although you can obtain CDP CLI commands for environment creation from CDP CLI help, the easiest way to obtain them is from the Cloudera web interface.

You can quickly obtain CDP CLI commands for creating an environment:

- From details of an existing environment
- From the register environment wizard

Create an environment from an existing environment

Obtain an environment template from an existing environment to create an environment with the exact same settings.

Since creating an environment, setting IDBroker mappings, and creating a Data Lake are separate actions in CDP CLI, you need to obtain three commands.

Required role: EnvironmentUser, EnvironmentAdmin, or Owner

Steps

1. Log in to the Cloudera web interface and navigate to the Cloudera Management Console
2. Navigate to environment details > Actions > Show CLI commands:

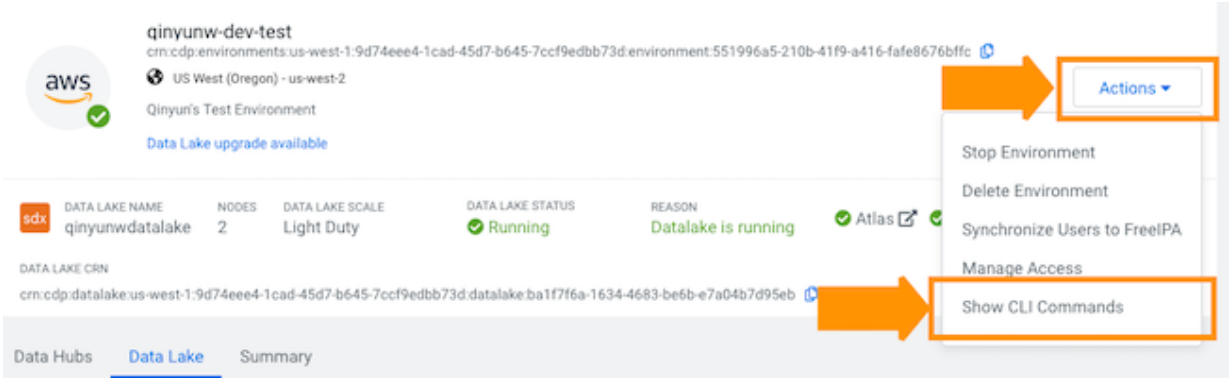

3. Click COPY three times to copy the three commands. These commands allow you to:
   a. Create an environment with the same settings as the existing environment.
   b. Set the same IDBroker mappings as in the original environment.
   c. Create a Data Lake with the same settings.
4. Before you can use these commands, make sure to update the following:
   a. In cdp environments create-<cloud-platform>-environment, update the value of --environment-name. It should be unique within Cloudera.
   b. In cdp environments set-id-broker-mappings, update the value of --environment-name. It should reference the name of the new environment that you are planning to create.
   c. In cdp datalake create-<cloud-platform>-datalake, update the value of --datalake-name. It should be unique within Cloudera.
   d. In cdp datalake create-<cloud-platform>-datalake, update the value of --environment-name. It should reference the name of the new environment that you are planning to create.
5. Run the three commands to:
   a. Register an environment.
   b. Set IDBroker mappings.
   c. Create a Data Lake.

Obtain CLI commands from the register environment wizard

Provide environment parameters in the environment wizard, and then on the last page of the wizard generate a CDP CLI template to create an environment and Data Lake with the parameters specified in the wizard. The obtained cluster template can be used to create an environment with the same settings via CDP CLI.

Since creating an environment, setting IDBroker mappings, and creating a Data Lake are separate actions in CDP CLI, you need to obtain three commands.

Required role: EnvironmentCreator

Steps

1. Log in to the Cloudera web interface and navigate to the Cloudera Management Console
2. Navigate to Environments > Register Environment.
3. Provide all the parameters for your environment.
4. On the last page, click SHOW CLI COMMAND in the bottom of the page.
5. Copy the three commands (for creating an environment, setting IDBroker mappings, and creating a Data Lake).
6. Run the commands:
   a. First run the command that creates the environment.
   b. Once the command finishes and the environment is running, run the command that sets IDBroker mappings.
   c. Next, run the command that creates the Data Lake.

Understanding environment UI options

To access information related to your environment, navigate to the Cloudera Management Console service > Environments and click on your environment.

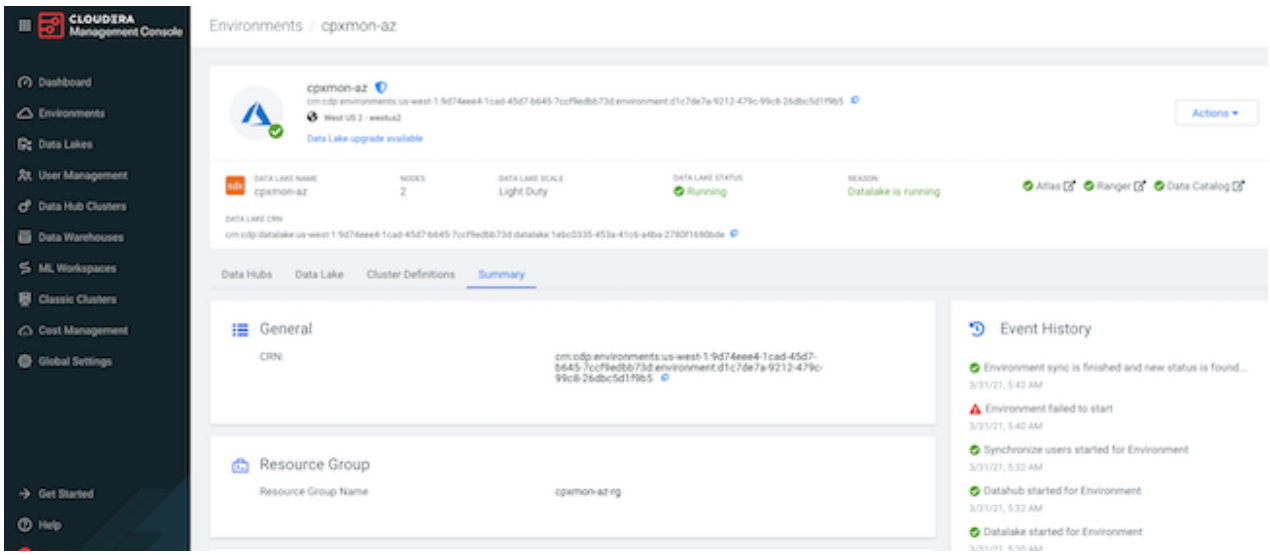

You need to have the EnvironmentUser role or higher for the environment in order to access details of that environment.

From environments details, you can access the following:

- From the Cloudera Management Console tab, you can create, manage, and access Cloudera Data Hub clusters within the environment.
- From the Data Lake tab, you can monitor, manage, and access the Data Lake cluster.
- From the Cluster Definitions tab, you can access all cluster definitions that can be used with the environment.
- From the Summary tab, you can manage and monitor your environment.

The Summary includes the following information:

Option	Description
General	This includes your environment's CRN. CRN is an ID that Cloudera uses to identify a resource.
Resource group	This lists the resource group(s) that your environment resources are associated with. During environment registration, you either provide an existing resource group or Cloudera creates multiple resource groups; There is no way to change this once an environment is running.
Credential	This links the provisioning credential associated with the environment and includes the option to change the credential.
Region	This lists the region in which your environment is deployed.
Network	This lists the networking resources used by your environment, provided by you or created by Cloudera during environment registration. You can add additional subnets for Cloudera Data Hub clusters deployed in the future.
Security Access	This lists the security groups used by your environment, provided by you or created by Cloudera during environment registration. You can provide new security groups for Data Hub clusters deployed in the future.
FreeIPA	This includes details of a FreeIPA server running in the environment and includes an Actions menu with FreeIPA management options.
Log Storage and Audits	This lists the cloud storage location used for logs and audits that you provided during environment registration. There is no way to update this location once your environment is running.
Telemetry	This includes your environment's telemetry settings. You can change them for any Cloudera Data Hub clusters created in the future.
Advanced	This lists the name of your root SSH key.

Related Information

Understanding Cloudera Data Hub details
Understanding Data Lake details

Using Compute Clusters

Compute Clusters enable you to deploy a containerized platform on Kubernetes for Data Services and shared services.

The Compute Cluster architecture offers simplified management, enhanced efficiency, and centralized control that leads to faster deployments, reduced configuration errors and improved system reliability. As multiple Data Services can optionally share the same Compute Cluster, it also lowers the cost of ownership.

When registering a Compute Cluster enabled environment, a default Compute Cluster is created. This default Compute Cluster is tied to the environment, and is used for running critical applications. The default Compute Cluster is labeled as **Default Cluster** in your environment to distinguish it from the other Compute Clusters.

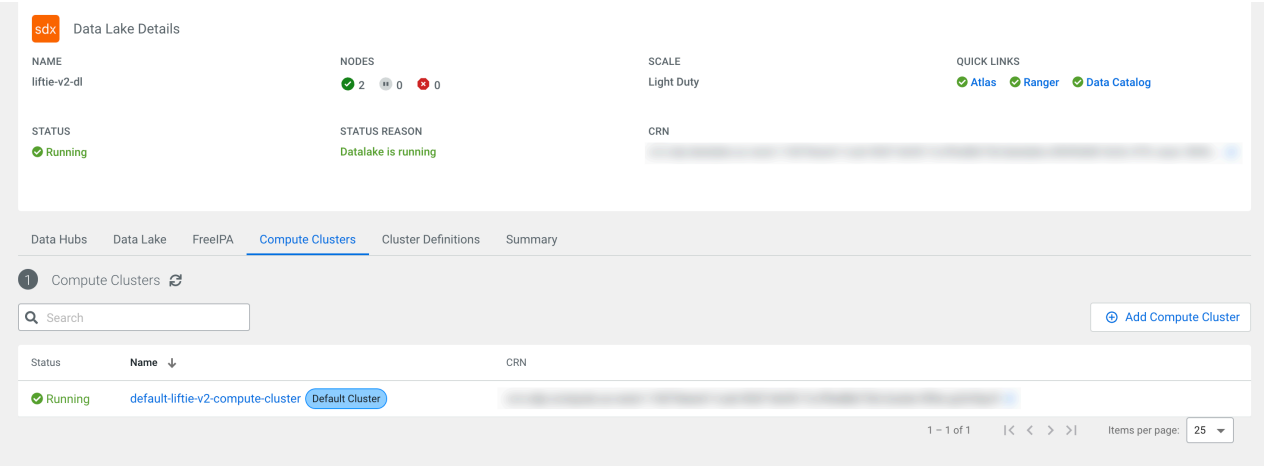

You can create additional Compute Clusters that inherit the configuration of the default Compute Cluster. You can manage the lifecycle of the additional Compute Clusters as they are independent of the environment.

Enabling default Compute Cluster for new environments

When creating your environment, you can enable the default Compute Cluster using the Cloudera Management Console or CDP CLI to be able to run your data and shared services on the containerized platform.

Required role: EnvironmentAdmin

Before you begin

- Ensure that your Azure account has all the resources required by CDP.

  For more information, see Azure subscription requirements.

Using Cloudera Management Console

When creating your environment in **Cloudera Management Console**, ensure that you use the **Enable Compute Cluster** setting to create the Compute Cluster enabled environment.

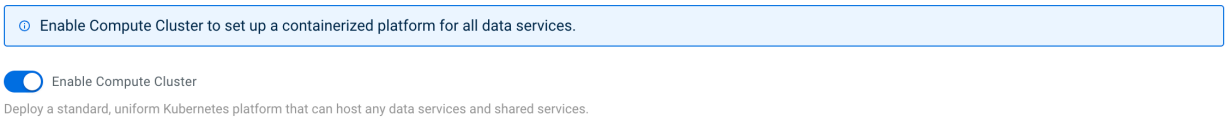

After completing the step for **Data Access and Data Lake Scaling**, configure the networking settings for Kubernetes with enabling **User Defined Routing**, and selecting **Private Kubernetes Cluster** or providing **Authorized IP Ranges** on the **Region, Networking and Security** page. When selecting **Private Kubernetes Cluster**, you also need to select an existing private DNS zone or select creating a new private DNS zone by Cloudera on your Azure account for the database. **Worker Node Subnets** are automatically pre-filled with the same set of subnets provided in **Network** section, but you have the option to not use all of the available subnets.

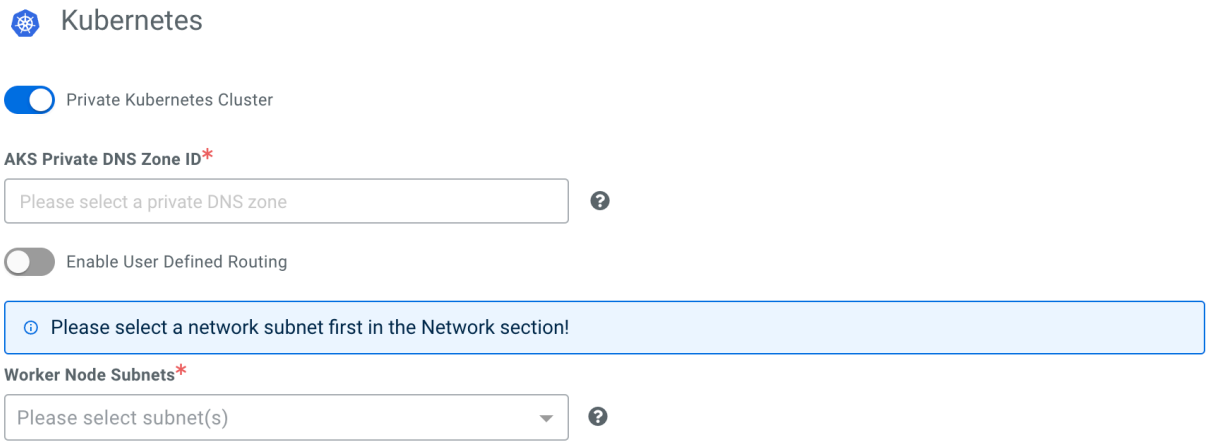

For more information about creating your environment, see the Registering environment (UI) documentation.

Using CDP CLI

Run the following command to create the Compute Cluster enabled environment:

For Without private cluster

```
cdp environments create-azure-environment \
--environment-name [***ENVIRONMENT NAME***] \
--credential-name [***CREDENTIAL NAME***] \
--region [***REGION***] \
--public-key [***PUBLIC SSH KEY***] \
--security-access [***SECURITY ACCESS CONFIGURATION***] \
--use-public-ip | --no-use-public-ip \
--log-storage [***STORAGE CONFIGURATION***] \
--enable-compute-cluster \
--compute-cluster-configuration \
privateCluster=FALSE, \
kubeApiAuthorizedIpRanges=[***CIDR1***],[***CIDR2***]
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

For With private cluster

```
cdp environments create-azure-environment \
--environment-name [***ENVIRONMENT NAME***] \
--credential-name [***CREDENTIAL NAME***] \
--region [***REGION***] \
```

```
--public-key [***PUBLIC SSH KEY***] \
--security-access [***SECURITY ACCESS CONFIGURATION***] \
--use-public-ip | --no-use-public-ip \
--log-storage [***STORAGE CONFIGURATION***] \
--enable-compute-cluster \
--compute-cluster-configuration \
privateCluster=TRUE
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

After the command runs, you can verify if the environment was successfully created with the default Compute Cluster with using the following commands:

- Describing the environment:

```
cdp environments describe-environment --env-name-or-crn [***ENVIRONMENT
 NAME OR CRN***]

...
        "azureComputeClusterConfiguration": {
            "privateCluster": false,
            "kubeApiAuthorizedIpRanges": [
                "0.0.0.0/0"
            ]
        },
        "enableComputeCluster": "true"
...
```

- Listing Compute Clusters:

```
cdp compute list-clusters --env-name-or-crn [***ENVIRONMENT NAME OR
 CRN***]
```

Note: FreeIPA must be created and running before the default Compute Cluster is created.

You can use the following command to retry the environment creation with the default Compute Cluster:

For Without private cluster

```
cdp environments initialize-azure-compute-cluster
--environment-name [***ENVIRONMENT NAME***] \
--compute-cluster-configuration \
privateCluster=FALSE, \
kubeApiAuthorizedIpRanges=[***CIDR1***],[***CIDR2***]
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

For With private cluster

```
cdp environments initialize-azure-compute-cluster
--environment-name [***ENVIRONMENT NAME***] \
--compute-cluster-configuration \
privateCluster=TRUE
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

Enabling default Compute Cluster for existing environments

In case you already have an environment, but would like to have your services to run on the containerized platform enabled by Compute Clusters, you can add the default Compute Cluster to your existing environment.

Required role: EnvironmentAdmin

Before you begin

- Ensure that the environment has no default Compute Cluster provisioned.
- Ensure that the environment is started and available.

Using Cloudera Management Console

1. Navigate to your environment.
2. Click Compute Clusters.

   The following message will indicate that Compute Cluster are not enabled for your environment:

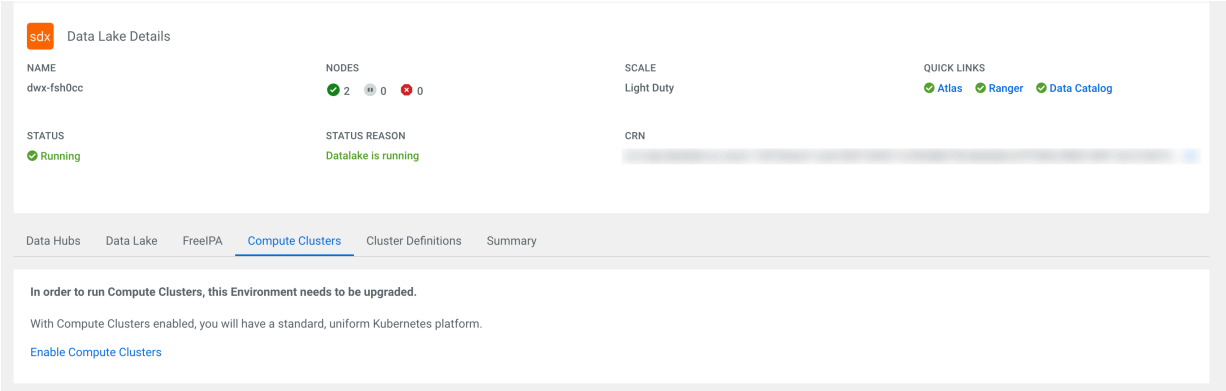

3. Click Enable Compute Clusters.
4. Provide the necessary networking information for the **Kubernetes** cluster.
   a. Enable User Defined Routing (UDR) in case public IPs are blocked for egress.In case you enable UDR, you must select the specific worker node subnet where the UDR is configured.
   b. If you need to create a Private Cluster, enable Private Kubernetes Cluster to create a private cluster that blocks all access to the API Server endpoint.
   c. If you do not need to create a Private Cluster, provide the CIDRs to the Kubernetes API Server Authorized IP Ranges field to specify a set of IP ranges that will be allowed to access the Kubernetes API server.
5. Click Submit.

   You will be redirected to the **Compute Clusters** tab, where you can track the creation process of the default Compute Cluster.

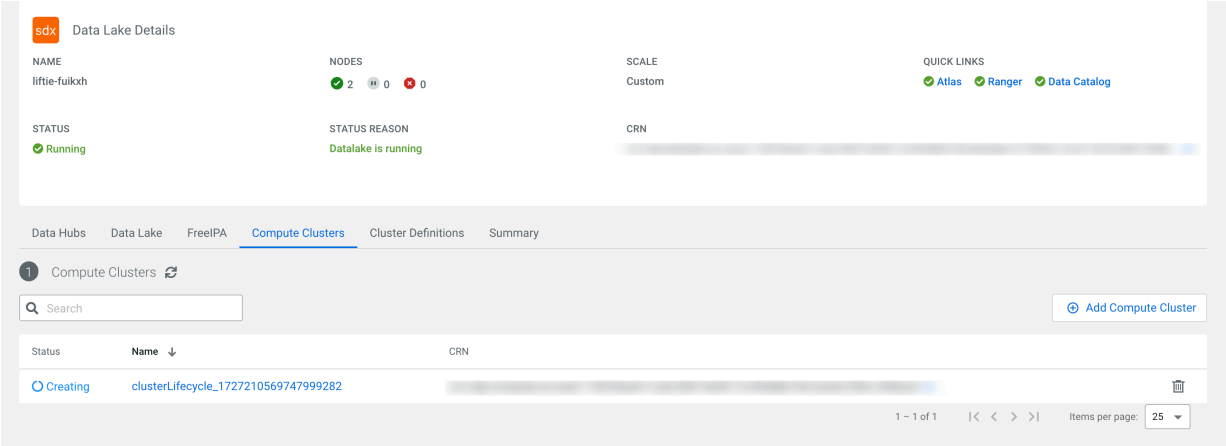

Using CDP CLI

Run the following command to add the default Compute Cluster to the environment:

For Without private cluster

```
cdp environments initialize-azure-compute-cluster
--environment-name [***ENVIRONMENT NAME***] \
--compute-cluster-configuration \
privateCluster=FALSE, \
kubeApiAuthorizedIpRanges=[***CIDR1***],[***CIDR2***]
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

For With private cluster

```
cdp environments initialize-azure-compute-cluster
--environment-name [***ENVIRONMENT NAME***] \
--compute-cluster-configuration \
privateCluster=TRUE
workerNodeSubnets=[***SUBNET1***],[***SUBNET2***]
```

The environment will have COMPUTE_CLUSTER_CREATION_IN_PROGRESS status. You can use the following command to check the status of the environment creation, the statusReason field will contain the information about the process:

```
cdp environments describe-environment --env-name-or-crn [***ENVIRONMENT
 NAME***]
```

For more detailed status information about the cluster creation, you can use the following command:

```
cdp compute list-clusters --env-name-or-crn [***ENVIRONMENT NAME***]
```

Adding more Compute clusters

You can add as many additional Compute Clusters as required beside the default Compute Cluster using Cloudera Management Console or CDP CLI.

Required role: EnvironmentAdmin

Using Cloudera Management Console

You can create additional Compute Clusters beside the default Compute Cluster using Cloudera Management Console.

1. Navigate to your environment.
2. Select Compute Clusters tab.

3. Click Add Compute Cluster.

   The **Add Compute Cluster** wizard appears.

   Add Compute Cluster

   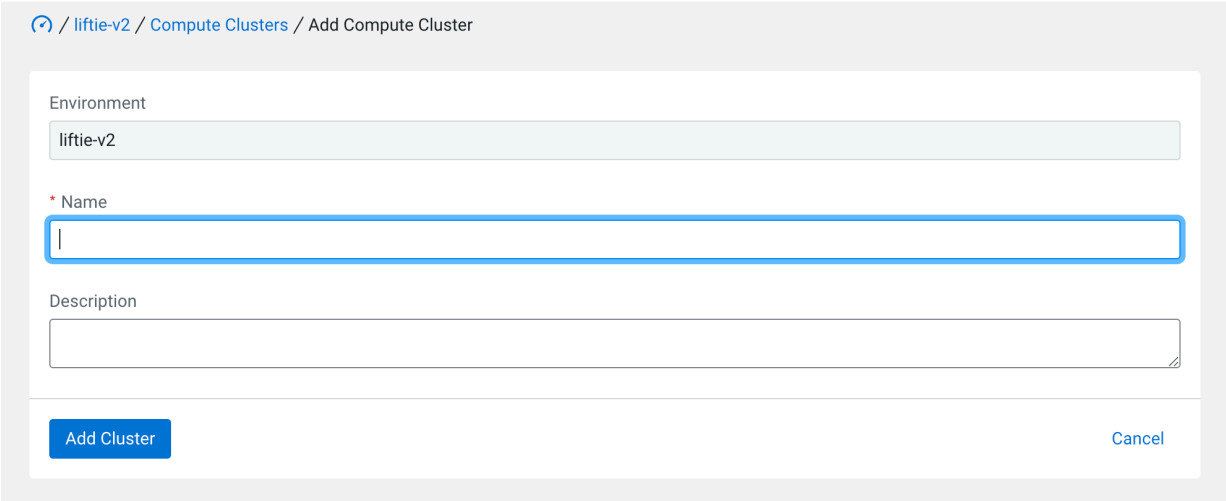

4. Provide a Name to the cluster, and optionally a Description.
5. Click Add Cluster.

   You will be redirected to the **Compute Clusters** tab, where you can track the creation process of the additional Compute Cluster.

Using CDP CLI

You can use the following CDP CLI command to create additional Compute Clusters after the default Compute Cluster is created:

```
cdp compute create-cluster
--environment --env-name-or-crn [***ENVIRONMENT NAME OR CRN***] \
--name [***CLUSTER NAME***]
```

After the command runs, you can verify if the additional Compute Cluster creation was successful using the following command:

```
cdp compute list-clusters --env-name-or-crn [***ENVIRONMENT NAME OR CRN***]
```

The additional Compute Cluster status should be in RUNNING state.

Managing Compute Clusters

After creation, you can view the Compute Cluster details, manage the access of the clusters, download the Kubeconfig file, and delete the created additional Compute Clusters.

Required resource role: EnvironmentAdmin or Owner

Required account role: IamViewer

Using Management Console

1. Click on the Name of the Compute Cluster.

   You will be redirected to the **Cluster Details** page.

   On the **Cluster Details** page, you can view the **Status**, **Creation Date**, **Created By**, **Description** and **CRN** of the Compute Cluster.
2. Click Actions to open the drop-down menu.

   Note: Ensure that you have one of the following roles to access the **Actions** menu
   - EnvironmentOwner or EnvironmentAdmin
   - ClusterCreator or ClusterOwner

   a. Click Manage Access to update the list of users who have access to manage the Compute Cluster and use it for installing Data Services.

      1. Click Update roles to update the **Resource Role** of a user or group.

         You can assign the Owner resource role to a user or group to provide permission to manage the Compute Cluster and install services on it.

   b. Click Kubeconfig to download the Kubeconfig file.

      1. Click Download Kubeconfig.

   c. Click Delete Compute Cluster, if you no longer need the additional Compute Cluster.

      Warning: Ensure that the data services running on the Compute Cluster are deleted before deleting the Compute Cluster itself.

      1. Confirm the deletion of the additional Compute Cluster by clicking Remove.
3. Click Networking tab to view the subnet information of the Compute Cluster.

   a. Click to update the **Kubernetes API Server Authorized IP Ranges**.

      Note: Ensure that you have one of the following roles to access the **Actions** menu
      - EnvironmentOwner or EnvironmentAdmin
      - ClusterCreator or ClusterOwner

      1. Add or remove the CIDRs, and click Save.
4. Click **Encryption** tab to view the encryption key of the Compute Cluster.
5. Click **Node Groups** tab to have an illustrated overview of the resource utilization of the different services running on the Compute Cluster.
6. Click **Compute Cluster Version** to view the Kubernetes and Insfrastructure Service versions of the cluster.
7. Click **Logs** to check the different events of the Compute Cluster.

Monitoring an environment

Once an environment exists, you can access it from the Cloudera Management Console.

Required role: EnvironmentUser, EnvironmentAdmin, or Owner

Steps

For CDP UI

1. To access an existing environment, navigate to Cloudera Management Console > Environments and click on your environment.
2. Click on the Summary tab to access environment details.

3. You can monitor the status of your environment from this page.

For CDP CLI

You can also list available environments from CDP CLI using the cdp environments list-environments command. For example:

```
cdp environments list-environments
{
    "environments": [
        {
            "environmentName": "cdp-demo",
            "crn": "crn:altus:environments:us-west-1:c8dbde4b-ccce-4f8d-
a581-830970ba4908:environment:d3361b40-39ab-4d87-bd5b-abc15f16b90c",
            "status": "DELETE_FAILED",
            "region": "us-east-2",
            "cloudPlatform": "AWS",
            "credentialName": "cdp-demo",
            "description": "Cdp demo"
        },
        {
            "environmentName": "cdp-new",
            "crn": "crn:altus:environments:us-west-1:c8dbde4b-ccce-4f8d-a5
81-830970ba4908:environment:1d2bacde-5c96-47c1-a597-9f276b824028",
            "status": "AVAILABLE",
            "region": "us-east-2",
            "cloudPlatform": "AWS",
            "credentialName": "cdp-demo",
            "description": ""
        }
}
```

To get more information about a specific environment, you can use the following commands:

```
cdp environments describe-environment --environment-name <value>
```

```
cdp environments get-id-broker-mappings --environment-name <value>
```

Related Information

Accessing Data Lake services
Understanding Data Lake details
Understanding Cloudera Data Hub cluster details
Managing FreeIPA

Environment status options

This topic lists all possible environment status options for the UI and CLI and explains what they mean.

Environment status	Description
Environment creation	
CREATION_INITIATED	Environment creation request was registered in the database and Cloudera is starting the environment creation flow.
ENVIRONMENT_INITIALIZATION_IN_PROGRESS	Setting up the region and network metadata (public/private and cidr).
ENVIRONMENT_VALIDATION_IN_PROGRESS	Setting up the region and network metadata (public/private and cidr).
NETWORK_CREATION_IN_PROGRESS	If the user chose the create new network option, then Cloudera creates the network on cloud provider side.

Environment status	Description
PUBLICKEY_CREATE_IN_PROGRESS	If the user choose the create new SSH key option, then Cloudera creates the SSH key on cloud provider side.
FREEIPA_CREATION_IN_PROGRESS	Creating the FreeIPA resources for an environment.
Environment update	
UPDATE_INITIATED	Environment update was requested and Cloudera is starting the update flow (network update, load balancer update, SSH key update).
Environment deletion	
DELETE_INITIATED	Environment deletion request was registered and Cloudera is starting the deletion flow.
NETWORK_DELETE_IN_PROGRESS	If the user chose the create new network option, then Cloudera deletes the network on cloud provider side.
PUBLICKEY_DELETE_IN_PROGRESS	If the user choosing the create new SSH key option, then Cloudera deletes the SSH key on cloud provider side.
FREEIPA_DELETE_IN_PROGRESS	Deleting the FreeIPA resources for an environment.
EXPERIENCE_DELETE_IN_PROGRESS	Deleting all the attached clusters (Cloudera Data Warehouse, Cloudera AI, and so on).
RDBMS_DELETE_IN_PROGRESS	Deleting all the provisioned RDS instances that are related to an environment.
CLUSTER_DEFINITION_DELETE_PROGRESS	Deleting all the cluster definitions that are created for an environment.
UMS_RESOURCE_DELETE_IN_PROGRESS	Deleting all the related UMS resources for an environment.
IDBROKER_MAPPINGS_DELETE_IN_PROGRESS	Deleting all the IBroker mapping for an environment.
S3GUARD_TABLE_DELETE_IN_PROGRESS	Deleting all the Dynamo DB tables for an environment.
DATAHUB_CLUSTERS_DELETE_IN_PROGRESS	Deleting all the attached Cloudera Data Hub clusters.
DATALAKE_CLUSTERS_DELETE_IN_PROGRESS	Deleting the attached Data Lake cluster.
ARCHIVED	Environment has been deleted (not shown on the UI).
Environment is running	
AVAILABLE	Environment is available (ready to use).
Environment process failed	
CREATE_FAILED	Environment creation failed (Detailed message in the statusReason).
DELETE_FAILED	Environment deletion failed (Detailed message in the statusReason).
UPDATE_FAILED	Environment update failed (Detailed message in the statusReason).
Environment stop	
STOP_DATAHUB_STARTED	Stopping all the Cloudera Data Hub clusters in an environment.
STOP_DATAHUB_FAILED	Stopping all the Cloudera Data Hub clusters in an environment failed (Detailed message in the statusReason).
STOP_DATALAKE_STARTED	Stopping the Data Lake cluster in an environment.
STOP_DATALAKE_FAILED	Stopping the Data Lake cluster in an environment failed (Detailed message in the statusReason).
STOP_FREEIPA_STARTED	Stopping the FreeIPA instances in an environment.
STOP_FREEIPA_FAILED	Stopping the FreeIPA instances in an environment failed (Detailed message in the statusReason).
ENV_STOPPED	Environment was successfully stopped.
Environment start	
START_DATAHUB_STARTED	Starting all the Cloudera Data Hub clusters in an environment.

Environment status	Description
START_DATAHUB_FAILED	Starting all the Cloudera Data Hub clusters in an environment failed (Detailed message in the statusReason).
START_DATALAKE_STARTED	Starting the Data Lake cluster in an environment.
START_DATALAKE_FAILED	Starting the Data Lake cluster in an environment failed (Detailed message in the statusReason).
START_FREEIPA_STARTED	Starting all the FreeIPA instances in an environment.
START_FREEIPA_FAILED	Starting all the FreeIPA instances failed in an environment (Detailed message in the statusReason).
START_SYNCHRONIZE_USERS_STARTED	Starting user sync for all the clusters in an environment.
START_SYNCHRONIZE_USERS_FAILED	Starting user sync for all the clusters in an environment failed (Detailed message in the statusReason).
FreeIPA instance deletion	
FREEIPA_DELETED_ON_PROVIDER_SIDE	The FreeIPA instance has been deleted on cloud provider side.
Load balancer	
LOAD_BALANCER_ENV_UPDATE_STARTED	Start updating the LoadBalancer on Data Lake in an environment.
LOAD_BALANCER_ENV_UPDATE_FAILED	Failed to update the LoadBalancer on Data Lake in an environment (Detailed message in the statusReason).
LOAD_BALANCER_STACK_UPDATE_STARTED	Start updating the LoadBalancer on Cloudera Data Hub clusters in an environment.
LOAD_BALANCER_STACK_UPDATE_FAILED	Failed to update the LoadBalancer on Cloudera Data Hub clusters in an environment (Detailed message in the statusReason).

Stop and restart an environment

You can stop an environment if you need to suspend but not terminate the resources within the environment. When you stop an environment, all of the resources within the environment are also stopped, including Data Lakes and Cloudera Data Hub clusters. You can also restart the environment.

Warning:

The Cloudera AI service does not support environment stop and restart. This means that if Cloudera AI workbenches are running or expected to be provisioned within an environment, then the environment should not be stopped. If done, this will disrupt running Cloudera AI workbenches and prevent successful provisioning of Cloudera AI workbenches in the environment.

Required role: EnvironmentAdmin or Owner

Steps

For CDP UI

1. Navigate to the environment in Cloudera Management Console > Environments.
2. Click Actions Stop Environment and confirm the action.
3. To restart the environment, click Actions Start Environment .

Warning: We have not tested or certified restarting the environment while Cloudera Data Engineering is running.

For CDP CLI

Use the following command to stop an environment:

```
cdp environments stop-environment --environment-name <ENVIRONMENT_NAME>
```

Use the following commands to start an environment:

```
cdp environments start-environment --environment-name <ENVIRONMENT_NAME>
 [--with-datahub-start]
```

Use the following commands to start an environment and all Cloudera Data Hub clusters running in it:

```
cdp environments start-environment --environment-name <ENVIRONMENT_NAME>
 --with-datahub-start
```

Delete an environment

Deleting an environment terminates all resources within the environment including the Data Lake.

Before you begin

To delete an environment, you should first terminate all clusters running in that environment.

Required role: Owner or PowerUser

Steps

For CDP UI

1. In Cloudera Management Console, navigate to Environments.
2. Click on your environment.
3. Click Actions Delete .
4. Check the box next to "I would like to delete all connected resources" if you have Data Lake and Cloudera Data Hub clusters running within the environment. This will delete the Data Lake and Cloudera Data Hub clusters together with the rest of the environment.

   Note: The "I would like to delete all connected resources" option does not delete any Cloudera Data Warehouse or Cloudera AI clusters running within the environment, so these always need to be terminated prior to environment termination.

   Note: To speed up future environment deployment, the cloudbreak-images resource group, which stores VHDs used for VM deployment, and its content are not deleted during environment deletion.
5. Check the box next to "I understand this action cannot be undone". This is required.
6. Click Delete.

For CDP CLI

When terminating an environment from the CDP CLI, you need to first terminate the Data Lake:

1. Terminate the Data Lake using the following command:

   ```
   cdp datalake delete-datalake --datalake-name <value>
   ```

2. Wait until the Data Lake terminates before proceeding. Use the following commands to check on the status of Data Lake:

```
cdp datalake get-cluster-host-status --cluster-name <value>
```

```
cdp datalake list-datalakes
```

3. Delete the environment using the following command:

```
cdp environments delete-environment --environment-name <value> --cascadi
ng
```

The --cascading option deletes or Cloudera Data Hub clusters running in the environment.

If environment deletion fails, you can:

- Repeat the environment deletion steps, but also check "I would like to force delete the selected environment". Force deletion removes Cloudera resources from Cloudera, but leaves cloud provider resources running.
- Clean up cloud resources that were left on your cloud provider account. See Cleaning up a failed AWS environment.

Only the resources that were provisioned as part of the environment are deleted. For example, if a new network was created by Cloudera for the environment, the network will be deleted; But if you provided your existing network, it will not be deleted as part of environment deletion.

Cleaning up a failed Azure environment

When environment creation fails, you should delete the environment. If environment termination fails, you should clean up any resources that might have already been created on your AWS account.

When environment creation fails, you should delete the environment by using the steps described in Delete an Environment.

If environment termination fails, you should clean up any resources that might have already been created on your Azure account. To do this, log in to your Azure Portal and do one of the following:

- If Cloudera created all resource groups for your Cloudera resources, find it and delete these resource groups. The names of these resource groups will include the name of the environment.
- If you used your own existing resource group, navigate to it and delete all the resources that were created by Cloudera. The names of these resources will include the name of the environment.

Add subnets to an environment

You can add additional subnets to an existing environment. These subnets will only be used for all Cloudera Data Hub clusters created within the environment in the future.

Before you begin

These steps assume that you have already created the subnets that you want to add to the environment.

Required role: EnvironmentAdmin or Owner

Steps

1. Navigate to Cloudera Management Console > Environments and click on the environment you want to modify.
2. Click the Summary tab.
3. Scroll down to the Network section.

4. Click the (edit) button, then click the Select Subnets pull down menu and select the subnet you want to add to the Environment.
5. Click the (save) button.

   You should see the new subnet listed in the Network section.

   Note:

   The newly added subnets will not be used for any Cloudera services other than Cloudera Data Hub. The newly added subnets will only be used for the Cloudera Data Hub clusters created within the environment after the new subnets were added. All the existing environment resources such as the Data Lake, FreeIPA, and any existing Cloudera Data Hub clusters will remain within the subnets originally defined during environment creation.

Add security groups to an environment

You can add additional security groups to an existing environment. These security groups will be used for all Cloudera Data Hub clusters created within the environment in the future.

Before you begin

These steps assume that you have already created the security groups that you want to add to the environment.

Required role: EnvironmentAdmin or Owner

Steps

1. Navigate to Cloudera Management Console > Environments and click on the environment you want to modify.
2. Click the Summary tab.
3. Scroll down to the Security Access section.
4. Click the (edit) button, then under Select Security Access Type select the Provide Existing Security Groups option and select the security groups that you want to add to the Environment.
5. Click the (save) button.

   You should see the new security groups listed in the Security Access section.

   Note:

   The newly added security groups will only be used for the Cloudera Data Hub clusters created within the environment after the new security groups were added. All the existing environment resources such as the Data Lake, FreeIPA, and any existing Cloudera Data Hub clusters will remain within the security groups originally defined during environment creation.

Change environment's credential

You can change the credential attached to an environment as long as the new credential provides the required level of access to the same Azure subscription as the old credential.

Required roles:

- EnvironmentAdmin or Owner of the environment
- SharedResourceUser or Owner of the credential

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Environments from the navigation pane.
4. Click on a specific environment.
5. Navigate to the Summary tab.
6. Scroll down to the Credential section.
7. Click to access credential edit options.
8. Select the new credential that you would like to use.
9. Click to save the changes.

For CDP CLI

If you would like to delete a credential from the CDP CLI, use:

```
cdp environments update-environment-credential --environment-name <value>
 --credential-name <value>
```

Enabling environment telemetry

You can optionally enable workload analytics so that diagnostic information about job and query execution is sent to Cloudera Observability for Cloudera Data Hub clusters. Similarly, you can optionally enable logs collection so that logs generated during deployments will be automatically sent to Cloudera.

Required role:

- PowerUser can set environment telemetry settings for the whole tenant.
- EnvironmentCreator can set environment telemetry settings during environment registration.
- EnvironmentAdmin or Owner can set environment telemetry settings for an existing environment.

Enabling workload analytics

If you enable workload analytics, diagnostic information about job and query execution is sent to Cloudera Observability for Cloudera Data Hub clusters created within all environments. This is disabled by default and can be enabled:

- For the whole tenant:
  - From the Cloudera web interface by navigating to Cloudera Management Console>Global Settings>Telemetry, by turning on the Enable Workload Analytics.
  - Or from the CDP CLI using the following command:

    ```
    cdp environments set-account-telemetry --workload-analytics
    ```

- For a specific environment only:
  - During environment creation from the Cloudera web interface, by turning on the Enable Workload Analytics option under Logs Storage and Audits in the environment creation wizard.
  - For an existing environment, from environment details > Telemetry by turning on the Enable Workload Analytics.
  - For an existing environment, from the CDP CLI using the following command:

```
cdp environments set-telemetry-features --environment-name <some-name> -
-workload-analytics
```

The environment-level setting overrides the tenant-level setting.

Note:

Only Cloudera Data Hub clusters created after enabling workload analytics on an environment will send data to Cloudera Observability. Cloudera Data Hub clusters created before workload analytics was enabled will not start sending data to Cloudera Observability if workload analytics is enabled for their parent environment.

Enabling cluster logs collection

If you enable cluster logs collection, logs generated during deployments will be automatically sent to Cloudera. This is disabled by default and can be enabled:

- For the whole tenant:
  - From the Cloudera web interface by navigating to Cloudera Management Console>Global Settings>Telemetry, by turning on Enable Cluster Logs Collection.
  - Or from the CDP CLI using the following command:

```
cdp environments set-account-telemetry --report-deployment-logs
```

- For a specific environment only:
  - During environment creation from the Cloudera web interface, by turning on the Enable Cluster Logs Collection option under Logs Storage and Audits in the environment creation wizard.
  - For an existing environment, from environment details > Summary >Telemetry by turning on the Enable Cluster Logs Collection.
  - For an existing environment, from the CDP CLI using the following command:

```
cdp environments set-telemetry-features --environment-name <some-name> -
-report-deployment-logs
```

The environment-level setting overrides the tenant-level setting.

Disable cloud storage logging for an existing environment

By default, Cloudera sends collected service logs from VM nodes to the cloud storage that you provided for logs during environment registration. In some cases, you may want to disable this for an existing environment.

You can disable this option from environment details > Summary > Telemetry by turning off Enable Cloud Storage Logging.

Note:

Disabling this option will affect only Cloudera Data Hub clusters created after the option was disabled.

Related Information

Configure lifecycle management for logs on Azure

Defining anonymization rules for Cloudera logs

Cloudera includes a set of default anonymization rules and allows you to define custom anonymization rules in order to remove sensitive information from Cloudera logs.

Use PCRE convention for writing custom anonymization rule patterns.

Anonymization rules are applied to the following logs:

- Cluster logs collected during deployments and automatically sent to Cloudera. See Enabling environment telemetry.
- Diagnostics logs that can be collected for troubleshooting and sent to Cloudera Support. See Generating a VM-based diagnostic bundle.

Note: Anonymization rules are only applied to environments created after the rules were added in Cloudera.

Default anonymization rules

Cloudera includes a set of default anonymization rules that anonymize the following:

Anonymization rule (PCRE)	Replacement	Description
\b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-\._]*[A-Za-z0-9])@(([A-Za-z0-9]|[A-Za-z][A-Za-z0-9\-]*[A-Za-z0-9])\.)+([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-]*[A-Za-z0-9])\b	email@redacted.host	Email addresses
\d{4}[^\w]\d{4}[^\w]\d{4}[^\w]\d{4}	XXXX-XXXX-XXXX-XXXX	Credit card numbers
\d{3}[^\w]\d{2}[^\w]\d{4}	XXX-XX-XXXX	SSN
FPW\:\s+[\w|\W].*	FPW: [REDACTED]	FreeIPA (workload) password
cdpHashedPassword=.*[']	[CDP PWD ATTRS REDACTED]	Hashed FreeIPA (workload) password.

Creating anonymization rule patterns

Use PCRE convention for writing anonymization rule patterns. For each pattern, come up with a replacement string.

Define custom anonymization rules

You can define custom anonymization rules in Cloudera. The anonymization rules are only applied to environments created after the rules were added in Cloudera.

Required role: PowerUser

Steps

For CDP UI

1. Once you have created the rules, navigate to Cloudera web interface > Cloudera Management Console > Global Settings > Telemetry > Anonymization rules.
2. Default rules are pre-populated.
3. Click on New rule and add a pattern and replacement string for your rule. Repeat for multiple rules.
4. Test the rules from the same page on the UI under Test rules:
   a. Under Input text paste an example text with sensitive content that should get anonymized by the rules that you added.
   b. Click Test all rules.
   c. The sensitive content should be removed nad replaced in the output printed in the Anonymized result text box.

5. Click Save Changes.

Note:

You can use the Set defaults button if you would like to revert to the default rules.

For CDP CLI

1. If you would like to add new rules, you should first prepare the patterns and replacement strings, and then test them with the following command:

```
cdp environments test-account-telemetry-rules --cli-input-json {
    "testInput": "Email: myemail@cloudera.com",
    "rules": [
        {
            "value": "\\b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\\-\\._]*[A-
Za-z0-9])@(([A-Za-z0-9]|[A-Za-z][A-Za-z0-9\\-]*[A-Za-z0-9])\\.)+([A-Za-
z0-9]|[A-Za-z0-9][A-Za-z0-9\\-]*[A-Za-z0-9])\\b",
            "replacement": "email@redacted.host"
        }
    ]
}
```

2. Run the following command to get your current telemetry settings in JSON format:

```
cdp environments get-account-telemetry
```

3. Copy the JSON file that you obtained in the output of this command and paste it into a text editor.
4. Update the JSON file, updating the settings or adding new rules.

Note:

Make sure to preserve all the existing rules, or else they will be deleted. Also, make sure to pass the workloadAnalytics and cloudStorageLogging parameters. If you don't pass all of the parameters, the parameters that are not passed will get reset to their default values.

5. Once you have the JSON file updated, run the cdp environments set-account-telemetry command. For example:

```
cdp environments set-account-telemetry --cli-input-json {
    "workloadAnalytics": true,
    "cloudStorageLogging": true,
    "rules": [
        {
            "value": "\\b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\\-\\._]*[A-
Za-z0-9])@(([A-Za-z0-9]|[A-Za-z][A-Za-z0-9\\-]*[A-Za-z0-9])\\.)+([A-Za-
z0-9]|[A-Za-z0-9][A-Za-z0-9\\-]*[A-Za-z0-9])\\b",
            "replacement": "email@redacted.host"
        }
    ]
}
```

Adding a customer managed encryption key to a Cloudera environment running on Azure

By default, local Data Lake, FreeIPA, and Cloudera Data Hub disks attached to Azure VMs and the PostgreSQL server instance used by the Data Lake and Cloudera Data Hub clusters are encrypted with server-side encryption (SSE) using Platform Managed Keys (PMK), but you can optionally configure SSE with Customer Managed Keys (CMK).

The CMK can be specified during environment registration and, if present, is used for encrypting Data Lake, FreeIPA, and Cloudera Data Hub disks and PostgreSQL server instances. Alternatively, it is possible to specify it once the environment is running, in which case the CMK will only be used for Cloudera Data Hub clusters created after the CMK was added.

The disks that are attached to the VMs of the Data Lake, FreeIPA, and Cloudera Data Hub clusters will be associated with a Disk Encryption Set (DES) that is created with the key URL as the underlying encryption key version. The DES dedicated to the Cloudera environment will be created in the resource group of the environment before the FreeIPA launch at the beginning of the environment creation process.

This documentation covers the following topics:

- Azure prerequisites for using a CMK
- Registering a new Azure environment and specifying a CMK
- Creating a Cloudera Data Hub with the same CMK
- Checking whether a CMK exists for an Azure environment
- Adding a CMK to an existing Azure environment

Azure prerequisites for using a CMK

You can use your existing Azure vault and vault key or create a new Azure vault and vault key.

For detailed requirements, refer to Azure Prerequisites: Customer managed encryption keys.

Register an Azure environment with a CMK

You can specify an existing customer managed key (CMK) during Azure environment registration and the encryption key will be used to encrypt the VMs and databases running in the environment.

Steps

For CDP UI

You can register your environment as described in Register an Azure environment from the Cloudera UI, just make sure that on the Region, Networking and Security page you enable the following:

1. Under Customer-Managed Keys, click Enable Customer-Managed Keys.
2. In the same section, under Select Resource group select the resource group where the CMK is located.
3. Provide the URL of the key value where the CMK resides. This is the same as the key identifier that you can copy directly from Azure Portal.

For CDP CLI

Use the following CDP CLI command to create an environment with the encryption key created earlier. Replace the placeholders with actual values. For example <ENVIRONMENT-NAME> should be replaced with an actual name.

The parameters important for this feature are highlighted:

```
cdp environments create-azure-environment \
--environment-name <ENVIRONMENT_NAME> \
--credential-name <CREDENTIAL_NAME> \
--region <REGION> \
--security-access cidr=<YOUR-ORGANIZATION’S_CIDR> \
--public-key '<SSH_PUBLIC_KEY>' \
--log-storage <STORAGE_LOCATION> \
--description '<DESCRIPTION>' \
--resource-group-name <ENV_RESOUCE_GROUP_NAME> \
--encryption-key-resouce-group-name <CMK_RESOUCE_GROUP_NAME> \
--encryption-key-url <KEY_RESOUCE_ID>
```

```
cdp environments set-id-broker-mappings \
--environment-name <ENVIRONMENT_NAME> \
--data-access-role <DATA_ACCESS_IDENTITY>\
--ranger-audit-role <RANGER_AUDIT_IDENTITY> \
--set-empty-mappings

cdp datalake create-azure-datalake \
--datalake-name <DATALAKE_NAME> \
--environment-name <ENVIRONMENT_NAME> \
--cloud-provider-configuration <STORAGE_LOCATION_BASE_CONFIGURATION>
```

The encryption key can be present in a separate resource group than in which the environment is being created. The resource group in which encryption key is present can be provided using --encryption-key-resouce-group-name. Either --encryption-key-resouce-group-name or --resource-group-name must be present. Depending on the resource group where you environment and encryption key reside, there are three possible use cases:

- If your encryption key is in the same resource group as you wish to create the environment in, you can provide a common resource group with --resource-group-name parameter. In this case, you do not need to provide the --encryption-key-resouce-group-name parameter.
- If your encryption key is in a different resource group than you wish to create the environment in and you wish to use some other existing resource group for the environment, you can provide encryption key's resource group with --encryption-key-resouce-group-name and the environment's resource group with --resource-group -name.

If you don't want to specify an existing resource group for the environment, you just need to provide the encryption key's resource group using --encryption-key-resouce-group-name and the environment's resource groups will be created by Cloudera.

You can obtain more complete commands using the instructions in Obtain CLI commands for registering an environment.

If you are using Azure Database for PostgreSQL Flexible Server in Cloudera, you can also specify a managed identity and thus use the same CMK for encrypting the Azure Flexible Server database instance used by Cloudera. For more information, see Configuring a CMK for data encryption in Azure Database for PostgreSQL Flexible Server .

Create a Cloudera Data Hub on Azure with a CMK

Use Cloudera web interface or CDP CLI to create a Cloudera Data Hub cluster. Note that this doesn't require any extra steps, so you can refer to the Cloudera Data Hub documentation.

The disks that are attached to the VMs of the Cloudera Data Hub cluster will be associated with a Disk Encryption Set (DES) that is created with the key URL as the underlying encryption key version. The DES dedicated to the Cloudera environment will be created in the resource group of the environment at the beginning of the environment creation process (before the FreeIPA launch, specifically).

Check Azure environment's CMK

You can check in an Azure environment's summary whether a CMK is used to encrypt a given environment.

Steps

For CDP UI

1. In the Cloudera Management Console, navigate to Environments and click on the environment for which you would like to set a CMK.
2. Click on the Summary tab.

3. Scroll down to the Customer Managed Encryption Key section.

   If a CMK exists, the entry will look similar to:

4. You can also find the CMK information by clicking on the Data Lake tab and then navigating to the Hardware tab.
5. Click on >> on the right side of the screen next to any of the nodes to expand the node related information.
6. Under Storage Details, the following information is included about the CMK:
   - Whether the node has been encrypted with a CMK
   - CMK encryption key URL
   - Disk Encryption Set ID

   For example:

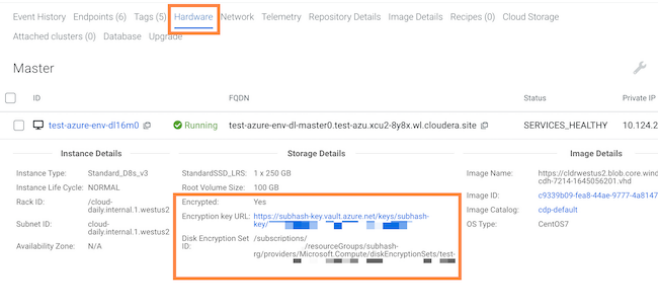

For CDP CLI

You can use the cdp environments update-azure-encryption-resources command to check environment's CMK encryption status.

If encryption key is set, the cdp environments update-azure-encryption-resources command will return the disk encryption set, encryption key, and encryption key's resource group details.

Check Cloudera Data Hub's CMK

You can check in a Cloudera Data Hub's hardware tab whether a CMK is used to encrypt it.

Note: A Cloudera Data Hub's CMK is the same as the one used for the environment in which the Cloudera Data Hub is running, so you can also find this information in the environment's details.

Steps

1. Navigate to the Cloudera Data Hub details > Hardware tab.
2. Click on >> on the right side of the screen next to any of the nodes to expand the node related information.
3. Under Storage Details, Under Storage Details, the following information is included about the CMK:
   - Whether the node has been encrypted with a CMK
   - CMK encryption key URL
   - Disk Encryption Set ID

Set a CMK for an existing Azure environment

You can set a CMK for an existing environment. The CMK will be only used for encrypting disks of Cloudera Data Hub clusters created after the CMK was added.

Note:

The CMK added to an existing environment will only be used for encrypting disks of Cloudera Data Hub clusters created after the CMK was added. Data Lake and FreeIPA disks and databases are not encrypted with the CMK.

Steps

For CDP UI

You can add the encryption key to an existing environment that does not have an encryption key set by navigating to the Summary page of the environment.

1. In the Cloudera Management Console, navigate to Environments and click on the environment for which you would like to set a CMK.
2. Click on the Summary tab.
3. Scroll down to the Encryption section.
4. If no CMK has been set, you will see a message stating that there is no customer-managed key enabled.
5. Click on the edit button in the top right corner of the tab.
6. Click on the toggle button next to Enable Customer-Managed Keys to enable adding a CMK.
7. Provide the following:
   a. Provide the encryption key URL. This is the same as the key identifier that you can copy directly from Azure Portal.
   b. If your encryption key resource group is different from the environment's resource group, provide the name of the resource group.
   c. Click Save.

For CDP CLI

You can add an encryption key for an existing environment that does not yet have encryption enabled.

If your encryption key resource group is same as environment resource group, use:

```
cdp environments update-azure-encryption-resources \
  --environment-name <ENVIRONMENT_NAME> \
  --encryption-key-url <CMK_URL> \
```

If your encryption key resource group is different from environment resource group, use:

```
cdp environments update-azure-encryption-resources \
  --environment-name <ENVIRONMENT_NAME> \
  --encryption-key-url <CMK_URL> \
  --encryption-key-resource-group-name <CMK_RESOURCE_GROUP>
```

If you are using Azure Database for PostgreSQL Flexible Server in Cloudera, you can also specify a managed identity and thus use the same CMK for encrypting the Azure Flexible Server database instance used by Cloudera. For more information, see Configuring a CMK for data encryption in Azure Database for PostgreSQL Flexible Server .

Defining custom tags

In the Cloudera Management Console clusters user interface, you can define tenant-level or environment-level custom tags across all instances and resources provisioned in your organization’s cloud provider account.

Resource tagging

When you create an environment or other resources shared across your cloud provider account, Cloudera clusters automatically adds default tags to the Cloudera-created resources in your cloud provider account. You can also define additional custom tags that Cloudera applies to the cluster-related resources in your account.

You can use tags to protect the cloud resources of your Cloudera environment. Using the tags, you can exclude the resources that should not be removed during housekeeping or security deletion activities that can result in data corruption and data loss.

Default tags

By default, Cloudera applies certain tags to cloud provider resources whenever you create the resource, for example an environment.

Cloudera applies the following tags by default:

- Cloudera-Resource-Name: the workload-appropriate Cloudera resource name. For example, an IPA CRN for an IPA, a data lake CRN for a data lake, or a Cloudera Data Hub CRN for a Cloudera Data Hub cluster. This CRN serves as a unique identifier for the resource over time.
- Cloudera-Creator-Resource-Name: Cloudera resource name of the Cloudera user that created the resource.
- Cloudera-Environment-Resource-Name: name of the environment with which the resource is associated.

Custom tags

There are two types of custom tags that you can define in the Cloudera Management Console: tenant-level tags that apply to Cloudera-created resources across your organization's entire cloud provider account, and environment-level tags.

In the Cloudera Management Console user interface, you can define tenant-level tags across all instances and resources provisioned in your organization's cloud provider account. These resources include not only provisioned instances, but disks, networks, and other resources as well. In your cloud provider account you can search or filter on either the tag key or value. Tenant-level tags cannot be overridden during environment creation.

In addition to tenant-level tags, you can also define environment-level tags. Environment-level tags are inherited by the resources specific to an environment. For example, environment-level tags are inherited by the following resources:

- FreeIPA
- Data Lakes
- Cloudera Data Hub clusters
- Data Warehouses
- Cloudera Operational Databases

As with tenant-level tags, you can search or filter on the key tag or key value in your cloud provider account.

Note: Cloudera applies custom tags during creation of the resources. For example, you can only define environment-level tags during environment registration. If you want to add or update cloud provider resource tags, you must do so through the cloud provider API.

For more information about using tags on cloud provider resources, consult AWS, Azure, or Google Cloud documentation. It is your responsibility to ensure that your tags meet your cloud provider requirements.

Supported services

While some Cloudera services such as Cloudera Data Hub inherit environment-level tags, others require that you add tags when provisioning or enabling the data service. The following table shows how tags can be added for various Cloudera services:

Cloudera service	How to add tags
Data Lake	Inherits tenant or environment level tags.
FreeIPA	Inherits tenant or environment level tags.
Cloudera Data Engineering	Does not inherit tenant or environment level tags but you can define tags when enabling Cloudera Data Engineering.
Cloudera Data Hub	Inherits tenant or environment level tags and you can add more tags when creating a Cloudera Data Hub.
Cloudera Data Warehouse	Inherits tenant or environment level tags.
Cloudera DataFlow	Inherits tenant level tags and you can add more tags when enabling Cloudera DataFlow.
Cloudera AI	Does not inherit tenant or environment level tags but you can define tags when creating a Cloudera AI workbench.
Cloudera Operational Database	Inherits tenant or environment level tags and you can add more tags when creating a Cloudera Operational Database database via CLI.

Defining tenant-level tags

Required roles: PowerUser can define tags for the whole tenant.

- EnvironmentAdmin or Owner can set environment telemetry settings for a specific environment.

Steps

1. In the Cloudera Management Console, click Global Settings Tags .
2. Click Add.
3. Define both a key and a value for the tag. Both the key and the value must be between 4- 255 characters, with the following restrictions:

   Key

   Allowed characters are hyphens (-), underscores (_), lowercase letters, and numbers. Keys must not start with the following words: 'azure', 'microsoft', and 'windows'. Keys must start with a lowercase letter and must not start or end with spaces.

   Value

   Allowed characters are hyphens (-), underscores (_), lowercase letters, and numbers. Values must not start or end with spaces. You can configure variables in the *{{{VARIABLENAME}}}* format. The following variables are supported for tenant-level tags:

   - *{{{CLOUDPLATFORM}}}* = AWS, AZURE or GCP
   - *{{{USERNAME}}}* = Cloudera username
   - *{{{USERCRN}}}* = Customer Resource Number (CRN) of Cloudera user
   - *{{{CREATORCRN}}}* = CRN of Cloudera resource creator
   - *{{{TIME}}}* = Actual time
   - *{{{ACCOUNTID}}}* = Cloudera account ID
   - *{{{RESOURCECRN}}}* = Generated string of Cloudera resource CRN
4. Click Add, and if necessary repeat the process for additional tags.

Note: Tenant-level tags are applied only to resources created after you define the tag. Any changes to the tag value do not propagate to existing resources that have the tag.

Defining environment-level tags

You define environment-level tags during environment registration.

Required roles: EnvironmentCreator can set tags for a specific environment during environment registration.

Steps

1. In the Cloudera Management Console, click Environments Register Environment .
2. Proceed through the environment registration steps.
3. After you define data access, add any environment-level tags by clicking Add and defining the tag key and value.

Related Information

Use Tags to Organize Azure Resources (Azure)

Using Azure Database for PostgreSQL Flexible Server

CDP uses Azure Database for PostgreSQL Flexible Server. The Flexible Server allows a highly available database to be deployed for Data Lake and Data Hub clusters. You can create Flexible Server instances with public access, where the Azure Database for PostgreSQL server is accessed through a public endpoint, or with private access, where the flexible server has no public endpoint accessible through the internet. The latter option requires a private DNS zone to be specified, and is either possible through the use of Azure Private Link for the Azure Database for PostgreSQL service, or, when Private Link is not used, it requires a delegated subnet to be created and added to your Cloudera Azure environment beforehand.

Using the Flexible Server offers the following benefits to Cloudera customers:

- Flexible Server allows you to deploy PostgreSQL version 14 and above. See *Supported PostgreSQL major versions in Azure Database for PostgreSQL - Flexible Server*.
- Unlike the previously used Single Server database instances, Flexible Server database instances can be stopped and restarted during Data Lake and Cloudera Data Hub cluster stop and restart. This offers a great cost-saving opportunity.

  Note: After 7 days of stopping the Flexible Server database instance, Azure automatically restarts the database. See *Limits in Azure Database for PostgreSQL - Flexible Server*.

- Flexible Server is multi-AZ capable and offers zone-redundant High Availability. With the Flexible Server, Data Lakes are backed with a highly-available PostgreSQL configuration of two instances. When using a multi-AZ deployment, the Flexible Server instances are deployed in multiple availability zones for additional fault tolerance.

For a detailed comparison of Single Server and Flexible Server offerings, refer to the *Comparison chart: Azure Database for PostgreSQL - Flexible Server vs. Single Server* in the Azure documentation.

Database server options

You can create Flexible Server instances with public access, where the Azure Database for PostgreSQL server is accessed through a public endpoint, or with private access , where the Flexible Server has no public endpoint accessible through the internet. There are two private flexible server options: Flexible Server using Azure Private Link for the Azure Database for PostgreSQL service or Flexible Server using a Delegated Subnet that needs to be created and added to your Cloudera Azure environment beforehand. Flexible Server with Private Link is the more advanced and recommended option as it offers easy connectivity to other Azure services that are utilizing Private Link for networking, the ability for a server to be reachable from public and private networks via both private and public addressing at the same time, and does not require delegated subnets to be created. For details, see *Private Link with Azure Database for PostgreSQL - Flexible Server*. Both private access options require a private DNS zone to be specified.

The Flexible Server with public access is used by default (as it does not require any special networking setup), but during environment creation you can specify to use the Flexible Server with private access, either with Private Link or with delegated subnet. It is also possible to specify to use the Single Server. However, the Flexible Server with

Delegated Subnet and the Single Server options are marked as deprecated. It is not recommended to use these as they will be phased out. Azure Single Server only supports Postgres versions up to 11. Postgres versions newer than 11 require the use of Azure Flexible Server. For details, see *What happens to Azure Database for PostgreSQL - Single Server after the retirement announcement?*

New Cloudera environments on Azure automatically use Flexible Server with public endpoints and Cloudera Data Hub clusters automatically inherit the settings from the environment they run in, but you can also enable Flexible Server when creating a Cloudera Data Hub cluster.

In general, when registering an Azure environment and creating a Cloudera Data Hub cluster, you can choose to use Flexible Server, Flexible Server with Private Link, Flexible Server with Delegated Subnet (deprecated), Single Server (deprecated), or Single Server with Private Link (deprecated), but the exact options vary depending on the environment settings. You have the following options when Data Lake or Cloudera Data Hub cluster creation is initiated:

- If the parent environment has been configured for private access, and the database type is set as Flexible Server, the Cloudera Data Hub or Data Lake cluster is launched with Flexible Server with Private Link if no delegated subnets are specified.
- If the parent environment has been configured for private access, and the database type is set as Flexible Server, the Cloudera Data Hub or Data Lake cluster is launched with Flexible Server with delegated subnet if there is a delegated subnet specified.
- If the parent environment has been configured for private access, and the database type is not set, a cluster with a Flexible Server is launched. You must specify the database type as Single Server or Single Server with Private Link to launch the Cloudera Data Hub or Data Lake with a Single Server database.
- If the environment has been configured for public access, and the database type is not set specifically as Single Server, the Cloudera Data Hub or Data Lake cluster is launched with public Flexible Server.
- If the environment has been configured for public access, and the database type is set as Single Server, the Cloudera Data Hub or Data Lake cluster is launched with public Single Server.
- If an environment is configured with Azure Single Server and Service Endpoints, by default new Cloudera Data Hub clusters provisioned in this environment will be created with Public Flexible Servers. If in this case you would still like to use a Single Server instead, then the --database-type=SINGLE_SERVER CLI parameter should be used, or you can achieve the same on the UI under the Advanced Cloudera Data Hub options.

Limitations

The following limitations currently apply:

- If the server-side encryption (SSE) that the Data Lake or Cloudera Data Hub clusters are encrypted with is configured with Customer Managed Keys (CMK), an upgrade from Azure Single Server to Azure Flexible Server is currently not available. This capability is expected to be released soon.

In order to set up this feature, you should review Azure prerequisites and then you can enable Flexible Server during Azure environment registration in Cloudera or upgrade existing Azure Databse for PostreSQL Single Servers to Flexible servers as described in the linked documentation:

Related Information

Supported PostgreSQL major versions in Azure Database for PostgreSQL - Flexible Server

Limits in Azure Database for PostgreSQL - Flexible Server

Comparison chart: Azure Database for PostgreSQL - Flexible Server vs. Single Server

Private Link with Azure Database for PostgreSQL - Flexible Server

What happens to Azure Database for PostgreSQL - Single Server after the retirement announcement?

Upgrading Azure Single Server to Flexible Server

Azure prerequisites for Flexible Server

The following Azure prerequisites must be in place in order to use the Flexible Server with Cloudera:

- If you would like to use zone redundant-HA, select a region that supports it.
- The policy provided for the Azure credential must have the required permissions mentioned below.
- When Flexible Server with Delegated Subnets is selected from the Azure database type options, Flexible Server instances need to be deployed in a delegated subnet within the VNet. This is not required either when public endpoints are used or when Azure Private Link is used to achieve the “private service” mode.
- When deployed in “private service” mode, either using Azure Private Link or delegated subnets, providing the private DNS zone information is mandatory in order to be able to perform DNS resolution. This is not required when public endpoints are used.

Read on to learn how to meet these prerequisites.

Supported regions

Ensure that your selected region has compute availability for Flexible Server. See Flexible Server Azure Regions.

Credential permissions

The Prerequisites for the provisioning credential documentation lists role definitions that includes sufficient permissions for Flexible Server. The role definitions include the following additional permissions that are required for Flexible Server:

Permission	Description
Microsoft.DBforPostgreSQL/flexibleServers/write	Creates a server with the specified parameters or updates the properties or tags for the server.
Microsoft.DBforPostgreSQL/flexibleServers/delete	Deletes an existing server.
Microsoft.DBforPostgreSQL/flexibleServers/start/action	Starts an existing server.
Microsoft.DBforPostgreSQL/flexibleServers/stop/action	Stops an existing server.
Microsoft.DBforPostgreSQL/flexibleServers/firewallRules/write	Restrict which networks can access the database instance.
Microsoft.DBforPostgreSQL/flexibleServers/configurations/write	This permission is required to change Flexible Server configuration.

Delegated subnet

You have two options to deploy Flexible Server instances in private service mode (without public endpoints): using Private Link or without Private Link. Flexible Server with Private Link is the recommended option. If Private Link is not used , Flexible Server instances need to be deployed in a “delegated subnet” within the VNet. The Flexible Server with Delegated Subnet option, although still available on the UI, is deprecated and Cloudera recommends that you use the Azure Private Link option instead.

As mentioned in Azure documentation, to be able to utilize private access with VNet integration, when private endpoints are not used, it is a prerequisite to delegate a subnet to Microsoft.DBforPostgreSQL/flexibleServers. This delegation means that only Azure Database for PostgreSQL Flexible Servers can use that subnet. No other Azure resource types can be in the delegated subnet.

You need to create such a delegated subnet and provide it to Cloudera during environment registration. This delegated subnet will be used by Azure Database for PostgreSQL instances. The delegated subnet provided during environment registration will be used by default for all Azure Database for PostgreSQL instances used in Cloudera.

Note: Although you can currently select multiple subnets, the larger subnet is always used by Cloudera. That is, if there are two subnets provided, one with 128 available IPs and another with 256 available IPs, the second one will be picked. Even though you can select multiple delegated subnets, we recommend that you provide only one.

Creating a delegated subnet

For a step-by-step official guide on how to perform the delegation, see Delegate a subnet to an Azure service. For considerations on the delegated subnet’s sizing, see Virtual network concepts.

Here is a screenshot from Azure Portal showing the desired setting:

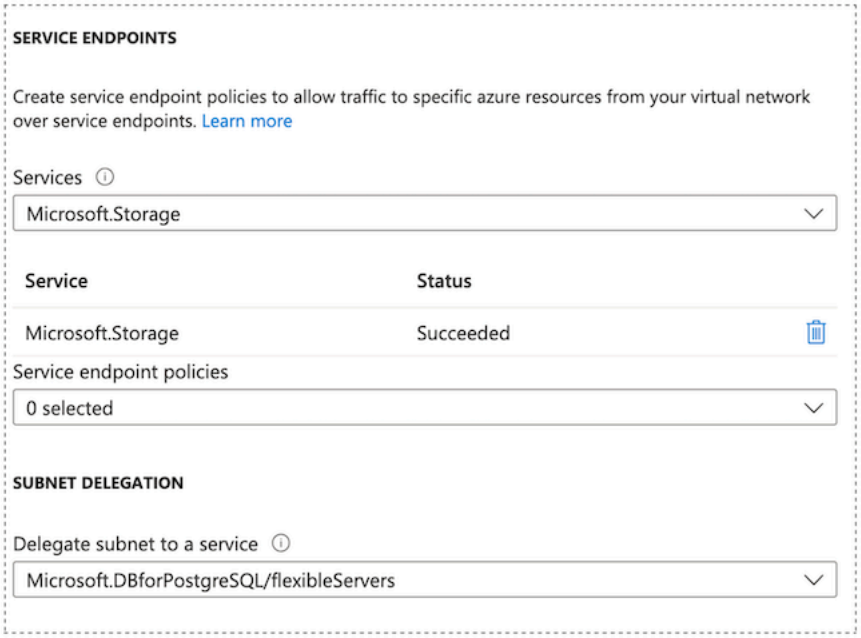

The Microsoft.Storage service endpoint is set automatically during deployment by Azure.

After the subnet has successfully been delegated, don't forget to record the full subnet ID or the name of the subnet for later use as an input (referred to as <delegated-subnet-id>). For example: /subscriptions/3ddda1c7-d1f5-4e7b-ac81-abcdefgh/resourceGroups/rg/providers/Microsoft.Network/virtualNetworks/vnet/subnets/subnet

Private DNS Zone

When using "private service" mode with Azure virtual network, a private DNS zone is mandatory in order to be able to perform DNS resolution.

As mentioned in Using a Private DNS Zone, a private DNS zone is used for creating a new Azure Database for PostgreSQL Flexible Server using private network access. Specifically, in case of Flexible Server with Private Link, the private DNS zone must be named privatelink.postgres.database.azure.com. In case of Flexible Server with a delegated subnet, a private DNS zone that ends with .postgres.database.azure.com should be created.

If you do not provide the private DNS zone, Cloudera creates a new private DNS zone in your resource group with the naming convention <resource-group-name>.flexible.postgres.database.azure.com. In case of Flexible Server with Private Link, the naming convention for the Cloudera-created private DNS zone is <resource-group-name>.privatelink.postgres.database.azure.com.

Note: If you are editing an existing environment to enable Flexible Server on it, you must provide a Private DNS Zone (unlike during environment creation where Cloudera can create it for you).

Note: Only one Private DNS Zone is created per resource group.

Creating a Private DNS Zone

When selecting a name for the Private DNS zone, in case of Flexible Server with Private Link, use privatelink.postgres.database.azure.com. Use one of the following forms for Flexible Server with Delegated Subnet:

[name1].[name2].postgres.database.azure.com

or

[name].postgres.database.azure.com

After the Private DNS zone has been created, don't forget to record the full Private DNS Zone ID, referred as <dns-zone-id>, for later use as an input. For example: /subscriptions/3ddda1c7-d1f5-4e7b-ac81-abcdefgh/resourceGroups/drg/providers/Microsoft.Network/privateDnsZones/flexible.private.postgres.database.azure.com

For instructions on how to create a Private DNS zone, see Quickstart: Create an Azure private DNS zone using the Azure portal.

Virtual Network Link

After the Private DNS Zone has been created in Azure, you need to link the VNet that you would like to use for your Cloudera environment to the Private DNS Zone. Once linked, resources hosted in that VNet can access the Private DNS Zone.

If you let Cloudera create the Private DNS Zone, then Cloudera creates the required Virtual Network Link connecting the VNet of your deployment with the created Private DNS Zone.

To learn more about virtual network links, see What is a virtual network link?.

Linking the virtual network

For instructions on how to link the VNet to the previously created Private DNS Zone, see Link the virtual network.

Enabling Flexible Server during Azure environment creation

During environment registration in Cloudera, the Flexible Server in public service mode is used by default. You also have two options to use the Flexible Server in private service mode: Flexible Server with Private Link and Flexible Server with Delegated Subnet. Flexible Server with Private Link is the recommended option, while Flexible Server with Delegated Subnet is still available but deprecated

When Cloudera is deployed in "private service" mode (without public endpoints), during environment creation you need to select the private DNS zone in case of using a Flexible Server with Private Link, or provide the following information when using a Flexible Server with delegated subnet:

- An ID of the delegated subnet
- A private DNS zone ID (optional parameter).

The virtual network link does not need to be specified as input. If you do not provide the private DNS zone, Cloudera creates it for you.

The steps below show you how to enable a private Flexible Server in Flexible Server with Private Link mode or Flexible Server with Delegated Subnet mode. If you would like to enable Flexible Server instances with public access, you do not need to do anything special, as this option is used by default.

Prerequisites

See Azure prerequisites for Flexible Server.

Steps

For CDP UI

Flexible Server with Private Link (recommended)

1. In the Database section, select Flexible Server with Private Link from the dropdown.

2. Select a private DNS zone for the private Flexible Server. If you do not select one, it will be created automatically. The private DNS zone must be named privatelink.postgres.database.azure.com.

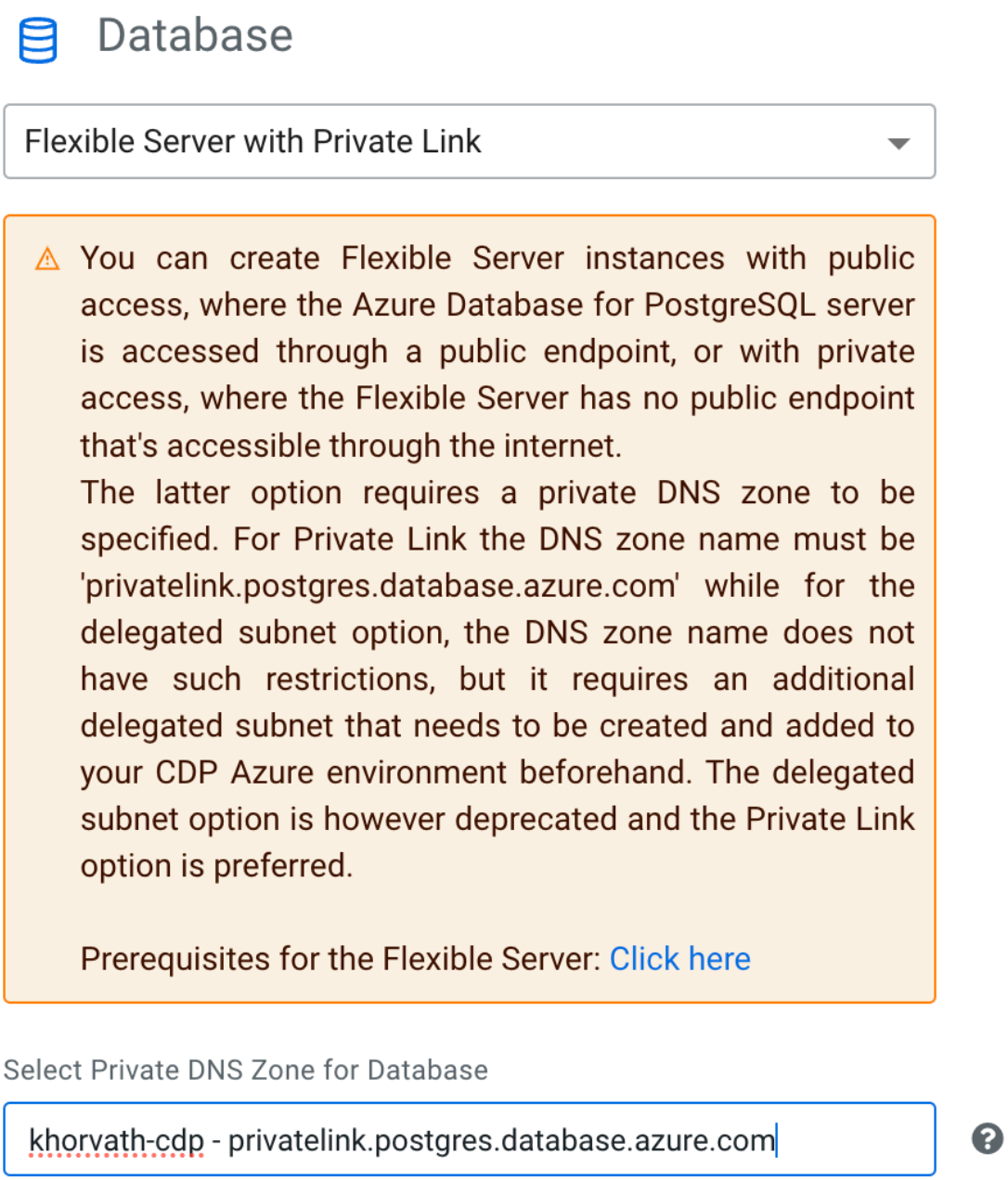

3. Finish registering your Azure environment in Cloudera.

Flexible Server with Delegated Subnet (deprecated)

1. In the Database section, select Flexible Server with Delegated Subnet from the dropdown.

2. Select a delegated subnet for the private Flexible Server.

   Note: Although you can currently select multiple subnets, the larger subnet is always used by Cloudera. That is, if there are two subnets provided, one with 128 available IPs and another with 256 available IPs, the second one will be picked. Even though you can select multiple delegated subnets, we recommend that you provide only one.

3. Select a private DNS zone for the Private Flexible Server. If you do not select one, it will be created automatically.

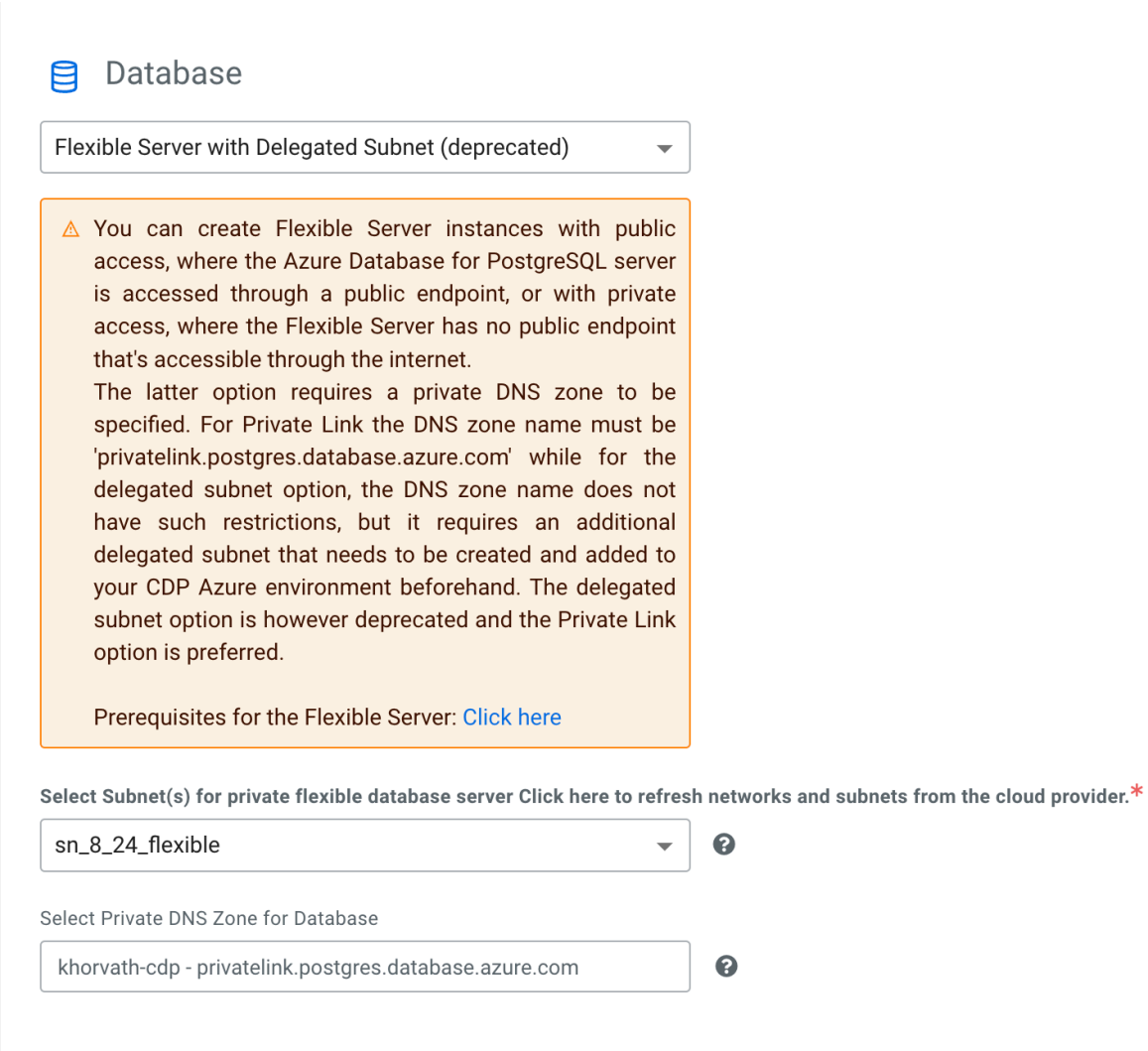

4. Finish registering your Azure environment in Cloudera.

For CDP CLI

1. Register an Azure environment using the cdp create-azure-environment CDP CLI command including --existing-network-params with a reference to the Private DNS Zone ID and, in case of Flexible Server with

Delegated Subnet, --flexible-server-subnet-ids with a reference to the delegated subnet ID. The virtual network link does not need to be specified as input.

Note: Although you can currently provide multiple subnets, the larger subnet is always used by Cloudera. That is, if there are two subnets provided, one with 128 available IPs and another with 256 available IPs, the second one will be picked. Even though you can select multiple delegated subnets, we recommend that you provide only one.

For example:

```
cdp environments create-azure-environment
  --environment-name <env-name>
...
  --existing-network-params networkId=dp-rg-test-vnet,resourceGroupName=
dp-rg,subnetIds=/subscriptions/3ddda1c7-d1f5-4e7b-ac81-0523f483b3b3/reso
urceGroups/dp-rg/providers/Microsoft.Network/virtualNetworks/dp-rg-vnet/
subnets/a,/subscriptions/3ddda1c7-d1f5-4e7b-ac81-0523f483b3b3/resourceGr
oups/dp-rg/providers/Microsoft.Network/virtualNetworks/dp-rg-vnet/subnet
s/2,default,databasePrivateDnsZoneId=<dns-zone-id>
  –-flexible-server-subnet-ids <delegated-subnet-id>
```

The following table explains the required parameters:

Parameter name	Description	Possible values
existing-network-params.databasePrivateDnsZoneId (string)	The ID of an existing private DNS zone used for the database.	Full resource reference
flexible-server-subnet-ids (array)	Required only for Flexible Server with Delegated Subnet option. If not specified, a Flexible Server with Private Link is launched. Comma separated list of the subnet names or full resource IDs delegated for flexible server. This can be specified in two formats: subnet1,subnet2 or /subscriptions/3ddda1c7-d1f5-4e7b-ac81-0523f483b3b3/resourceGroups/dp-rg/providers/Microsoft.Network/virtualNetworks/dp-rg-vnet/subnets/1,/subscriptions/3ddda1c7-d1f5-4e7b-ac81-0523f483b3b3/resourceGroups/dp-rg/providers/Microsoft.Network/virtualNetworks/dp-rg-vnet/subnets/2 The parameter takes a list of subnet IDs (or creates the IDs in case subnet names are provided), validates if they are indeed delegated, and takes the subnet with the largest CIDR range.	List of full resource reference(s). Although this parameter accepts a comma-separated list of subnets, the subnet with the largest CIDR range is always used. Therefore, you should provide only one subnet.

2. Set IDBroker mappings as usual using the cdp environments set-id-broker-mappings command.
3. Create a Data Lake as usual using the cdp create-azure-datalake CDP CLI command, including a reference to the database HA type and the database PostgreSQL engine version.

Related Information

Register an Azure environment from the Cloudera UI

Register an Azure environment from the CDP CLI

Enabling admin and user access to environments

Enabling Flexible Server during Cloudera Data Hub cluster creation

A Cloudera Data Hub cluster uses the same Flexible Server or Single Server settings as the environment in which it runs, but you can choose to enable a Flexible Server on Cloudera Data Hub clusters running in an environment that uses a Single Server. This option can be specified during Cloudera Data Hub cluster creation in the Advanced Options under Advanced options Network and Availability Azure Database Type .

There are the following possible scenarios:

- If your environment uses Flexible Server with Private Link, and you have specified a private DNS zone on the environment level (either during environment creation or later by editing the environment), then from the Azure Database Type dropdown you can select to use Flexible Server with Private Link or Single Server with Private Link (deprecated). You cannot select the public Flexible Server or the public Single Server option, or Flexible Server with Delegated Subnet in this case. The Flexible Server with Private Link used for your Cloudera Data Hub cluster will use the private DNS zone specified on the environment level.
- If your environment uses Single Server with Private Link, then from the Azure Database Type dropdown you can select to use Flexible Server with Private Link or Single Server with Private Link (deprecated) during Cloudera Data Hub cluster creation.

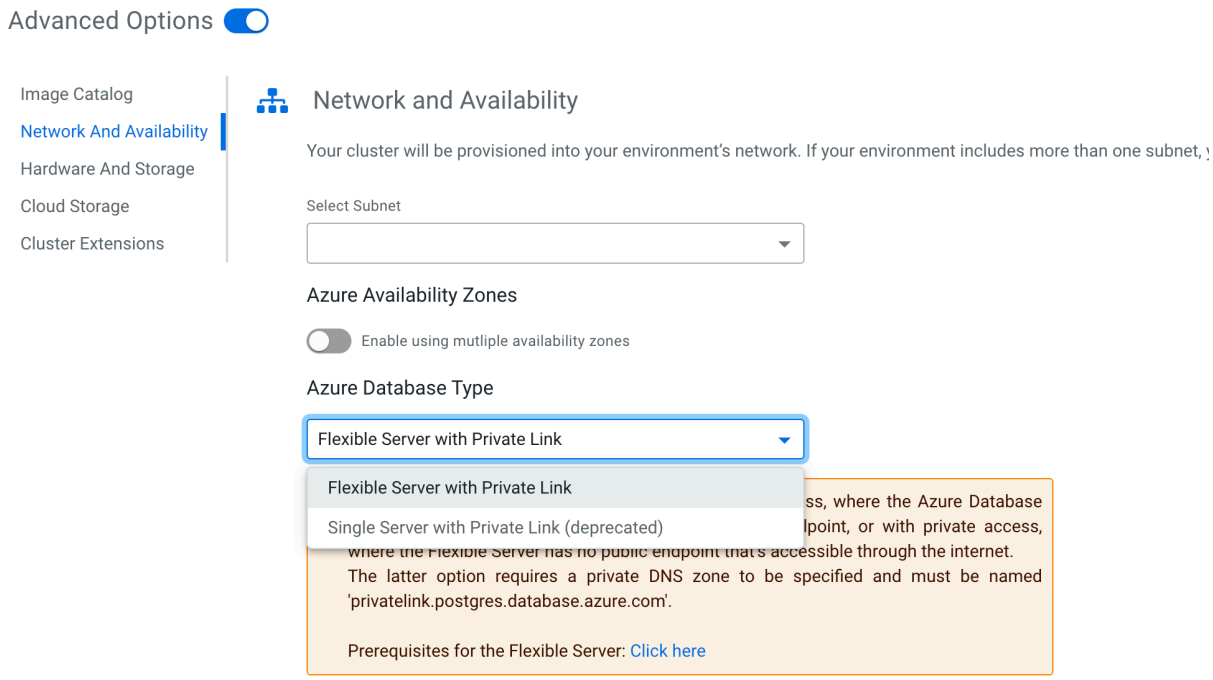

- If your environment uses Private Flexible Server with Delegated Subnet, and you have specified a delegated subnet and a private DNS zone on the environment level (either during environment creation or later by editing the environment), then from the Azure Database Type dropdown you can select to use Flexible Server with Delegated Subnet (deprecated) or Single Server (deprecated) for your Cloudera Data Hub cluster. You cannot select to use the public Flexible Server in this case. If you choose to use the Flexible Server with Delegated Subnet option for your Data Hub, it will use the delegated subnet and private DNS zone specified on the environment level.
- If you choose to use the public Flexible Server option and do not specify a delegated subnet and a private DNS zone on the environment level, then from the Azure Database Type dropdown you can select to use Flexible Server or Single Server for your Cloudera Data Hub cluster. You cannot select to use either private Flexible Server options in this case.

- If you choose to use the public Single Server option on the environment level, then from the Azure Database Type dropdown you can select to use Flexible Server or Single Server for your Cloudera Data Hub cluster. You cannot select to use either private Flexible Server options in this case.

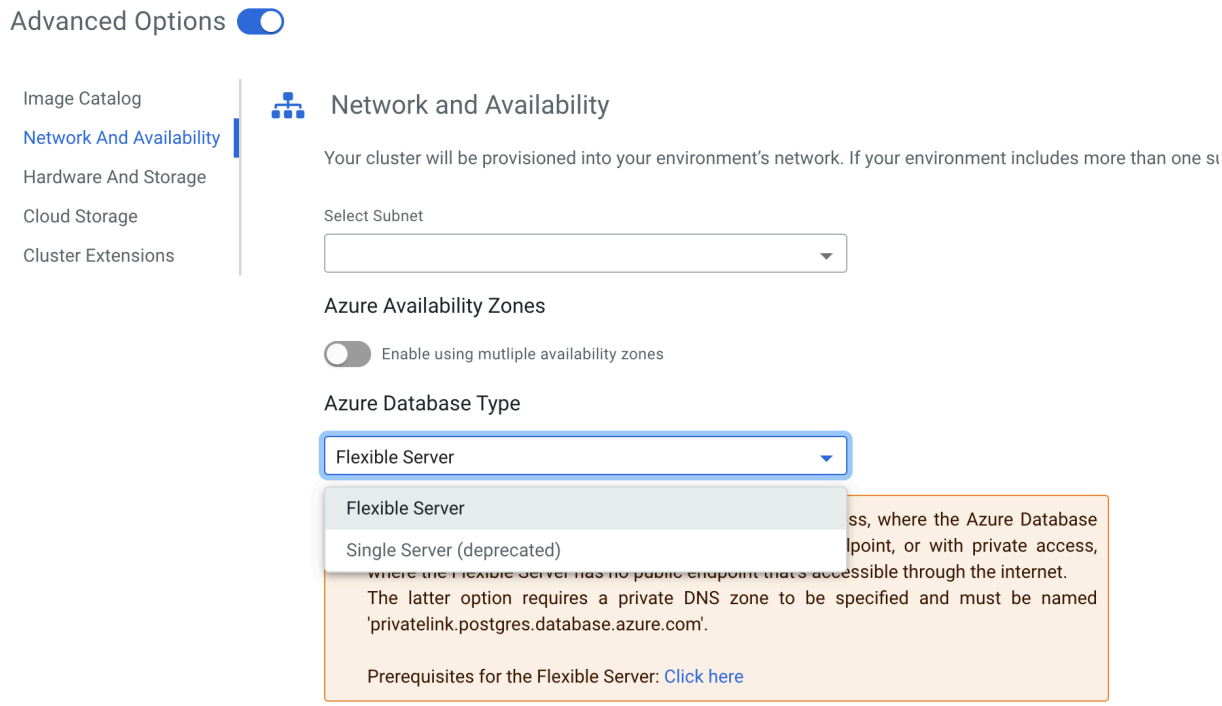

- If an environment is configured with Azure Single Server and Service Endpoints, by default new Cloudera Data Hub clusters provisioned in this environment will be created with Public Flexible Servers. If in this case you would still like to use a Single Server instead, then the --database-type=SINGLE_SERVER CLI parameter should be used, or you can achieve the same on the UI under the Advanced Data Hub options.

If you do not explicitly specify any Azure Database Server option, Cloudera follows the logic described in Private Flexible Server setup > Database Server Options.

Steps

For CDP UI

This option can be specified during Cloudera Data Hub cluster creation in the Advanced Options under Network and Availability > Select Azure Database Server.

For CDP CLI

Create a Cloudera Data Hub cluster as usual using the cdp create-azure-cluster CDP CLI command and specifying the —-database-type.

For example:

```
create-azure-cluster —-database-type FLEXIBLE_SERVER
```

The following table explains the required parameters:

Parameter	Description	Possible values

database-type (string)	The type of the azure database. FLEXIBLE_SERVER is the next generation managed PostgreSQL service in Azure that provides maximum flexibility over your database, built-in cost-optimizations. SINGLE_SERVER is a fully managed database service with minimal requirements for customizations of the database.	• FLEXIBLE_SERVER • SINGLE_SERVER

Related Information

Network and availability

Enabling Flexible Server with Private Link on an existing environment

You can change an existing environment that uses either a public Flexible Server or Flexible Server with Delegated Subnet to launch the new Data Hubs with a Flexible Server with Private Link. You can achieve this by editing the Network settings of your environment.

Note: Once you have enabled Flexible Server with Private Link by performing the following steps, you can no longer change the Azure Database Server type back to any other as this is the most advanced database server type.

Prerequisites

See Azure prerequisites for Flexible Server.

Steps

For CDP UI

1. In the Cloudera Data Hub cluster, navigate to Environments and then navigate to a specific environment.
2. Click the Summary tab.
3. Scroll down to Network and click the (edit) button.
4. Click the toggle button next to Enable Private Flexible Server to enable the feature.
5. Select Flexible Server with Private Link.

6. Select a Private DNS Zone for the Private Flexible Server. This is required.

Note: If you are editing an existing environment to enable a Flexible Server on it, you must provide a Private DNS Zone (unlike during environment creation where Cloudera cluster can create it for you).

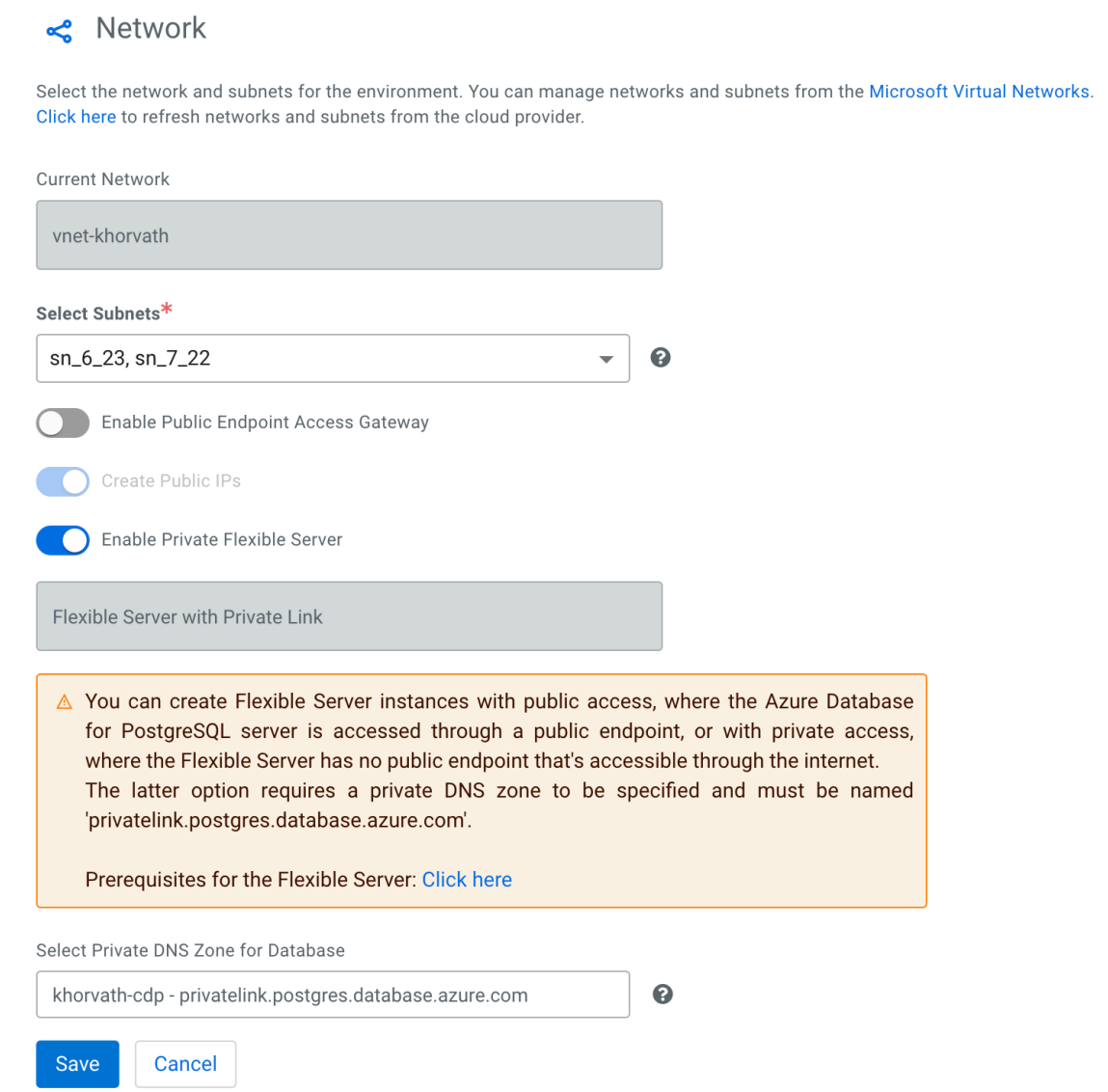

7. Click Save.

For CDP CLI

Use the following command:

```
cdp environments update-azure-database-resources
 --environment <ENVIRONMENT_NAME_OR_CRN>
 --database-private-dns-zone-id <SPECIFY-DNS-ZONE-ID>
```

For example:

```
cdp environments update-azure-database-resources
```

```
  --environment eng-ml-dev-env-azure
  --database-private-dns-zone-id /subscriptions/1e0c1142-6256-61b9-
b80a-c5888d6e1e22/resourceGroups/b3x-at50645-weu-bde/providers/
Microsoft.Network/privateDnsZones/b3xeng.postgres.database.azure.com
```

Upgrading Azure Single Server to Flexible Server

PostgreSQL version 11 databases used by Data Lakes and Cloudera Data Hub clusters need to be upgraded to PostgreSQL 14. During this process, the Azure Single Server database will be upgraded to Azure Flexible Server. If there is a Private DNS Zone added to the environment, a Private Link will be created for the database and the Flexible Server will be private. Otherwise, a public Flexible Server will be created.

Important: Cloudera strongly recommends opting out of Microsoft's PostgreSQL—Single Server auto-migration, as the auto migration can lead to service degradation or an outage of the respective resource. For details, see Cloudera Customer Advisory 2024-805: Azure auto-migration of PostgreSQL Single Server to Flexible Server.

This process describes how to upgrade a database used by a Data Lake or Cloudera Data Hub clusters to Azure Flexible Server. For a general description of Data Lake and Cloudera Data Hub database upgrades, see Upgrading Data Lake/Data Hub database.

During the upgrade process, a backup is created from the Azure Single Server database, the server is deleted, and a new Azure Flexible Server is created with the same name. This way there is no need to change hostnames. However, the username format has changed between Single Server and Flexible Server. For details, see the Comparison table in the Microsoft Azure documentation.

Feature / Capability	Azure Database for PostgreSQL Single Server	Azure Database for PostgreSQL Flexible Server
Username in connection string	<user_name>@server_name. For example, pgadmusr@mypgServer.	Just username. For example, pgadmusr.

This change is automatically handled by Cloudera, however, as a result, Cloudera Runtime services using the database will be reconfigured during the upgrade process to use the new username format. Any third-party applications using the database would need to reflect this change by updating the username.

Limitations

- If you are using a public Single Server and want to upgrade to a private Flexible Server, you need to add a Private DNS Zone to your environment first. For instructions on adding a Private DNS Zone, see Private DNS Zone.
- If you would like to upgrade from Azure Single Server to a private Flexible Server, the upgrade to Flexible Server with delegated subnet is not available, as the Flexible Server with delegated subnet option is deprecated. It is recommended to use Flexible Server with Private Link instead.
- If you are already using a private Flexible Server with a delegated subnet, it is not yet supported to migrate to a private Flexible Server with Private Link.
- If you are performing a Data Lake resize because you want to upgrade to Cloudera Runtime 7.2.18 or later version, you must perform the upgrade of the database first.

Prerequisites

- When you are upgrading from Azure Single Server to Azure Flexible Server, new permissions are required. These permissions are being validated during the first phase of the upgrade. For a detailed list of prerequisites for Azure Flexible Server, see Azure prerequisites for Flexible Server.

  Important: There is a permission which is not required for enabling Flexible Server on a new environment but is specific to upgrading to Flexible Server from Single Server. Check the Azure prerequisites for Flexible Server even if you have enabled Flexible Server on your environment earlier.

  If you have missed adding any of the required permissions and the upgrade to Flexible Server fails, you need to fix the permission issues on Azure side and then perform a retry through the Cloudera Management Console.
- If you are upgrading from a public Single Server, make sure that port 5432 is open for outbound traffic. Otherwise the database upgrade might fail with a connection error. Public Azure Single Server access works with Service Endpoints, but these are not available for Flexible Server.
- If you are using Cloudera Data Warehouse, perform the following steps:
  - Upgrade to Cloudera Data Warehouse version 1.4.1 or higher before you can upgrade to Azure Flexible Server. Determine the Cloudera Data Warehouse version you are on by clicking edit on the environment:

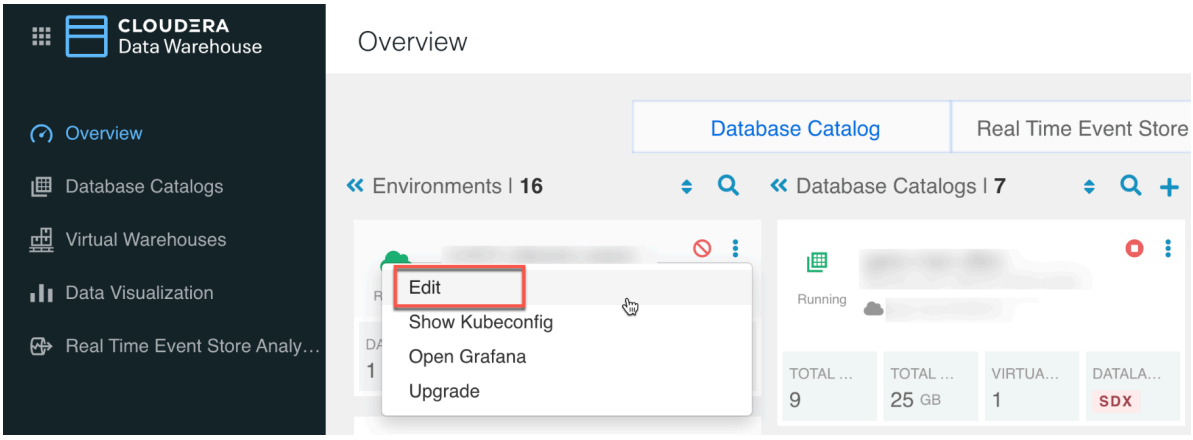

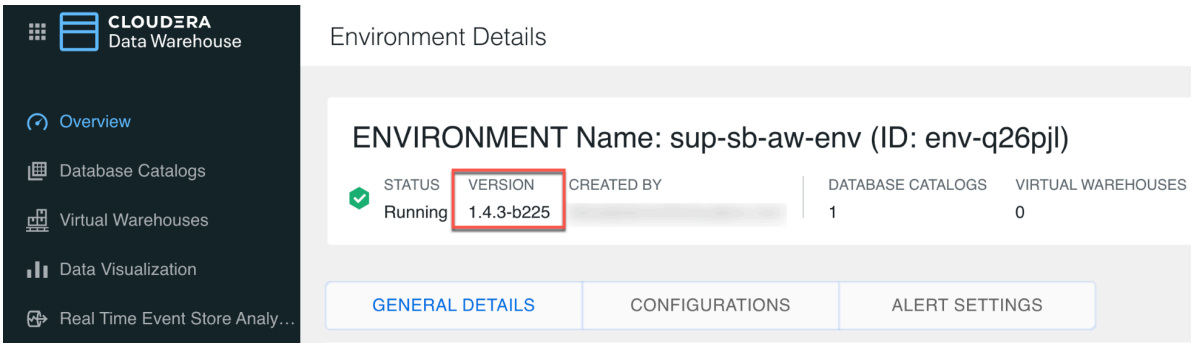

  - Stop the virtual warehouses and data catalogs associated with the environment.
- If you are using a Customer Managed Key (CMK) for data encryption, a managed identity is required to be able to upgrade from Azure Database for PostgreSQL Single Server to Flexible Server.

  For details on managed identity, see Managed identity for encrypting Azure Database for PostgreSQL Flexible Server.

  For information on how to add a managed identity to an environment already configured with CMK, see Updating an existing environment to add the managed identity Flexible Server.

Upgrading databases using the Cloudera UI

It is possible to upgrade from Azure Single Server to Azure Flexible Server using the Cloudera UI. The process can differ depending on PostgreSQL version that you are upgrading from, and depending on whether you are upgrading from private or public Single Server.

Upgrading a PostgreSQL 11 public Single Server to a PostgreSQL 14 public Flexible Server

Before you begin, make sure the prerequisites listed in Upgrading Azure Single Server to Flexible Server are met.

1. In the Cloudera Management Console, navigate to Environments and then navigate to a specific environment.

   Environments that require an upgrade are highlighted with a red background.

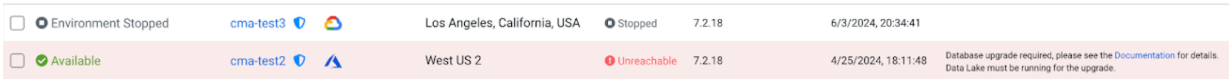

2. Click the Upgrade database button inside the warning message at the top of the screen.

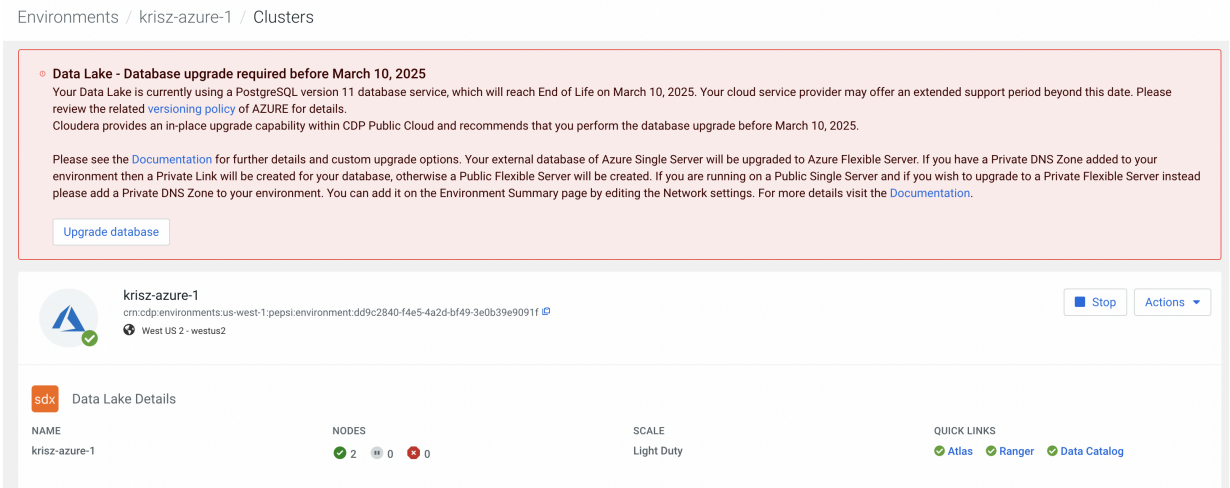

3. Once the Data Lake database is updated, check for the Cloudera Data Hub clusters of that Data Lake, if there is any database upgrade notification and perform the database upgrade as described above.

   Note: The database upgrade needs to be performed on every Data Lake and Cloudera Data Hub cluster separately, one by one.

Upgrading a PostgreSQL 10 or PostgreSQL 11 private Single Server to PostgreSQL 14 private Flexible Server with Private Link

Before you begin, make sure the prerequisites listed in Upgrading Azure Single Server to Flexible Server are met.

1. In the Cloudera Management Console, navigate to Environments and then navigate to a specific environment.
2. Click the Upgrade database button inside the warning message at the top of the screen.
3. Once the Data Lake database is updated, check for the Cloudera Data Hub clusters of that Data Lake, if there is any database upgrade notification and perform the database upgrade as described above.

   Note: The database upgrade needs to be performed on every Data Lake and Cloudera Data Hub cluster separately, one by one.

Upgrading a PostgreSQL 11 public Single Server to PostgreSQL 14 private Flexible Server with Private Link

Before you begin, make sure the prerequisites listed in Upgrading Azure Single Server to Flexible Server are met.

If you would like to upgrade from a public to a Private Flexible Server, you need to enable Private Flexible Server first, and then upgrade.

1. In the Cloudera Management Console, navigate to Environments.

   Environments that require a PostgreSQL database upgrade are listed with a red background.
2. Select a specific environment.
3. Click the Summary tab.

4. Scroll down to Network and click the (edit) button.
5. Click the toggle button next to Enable Private Flexible Server.

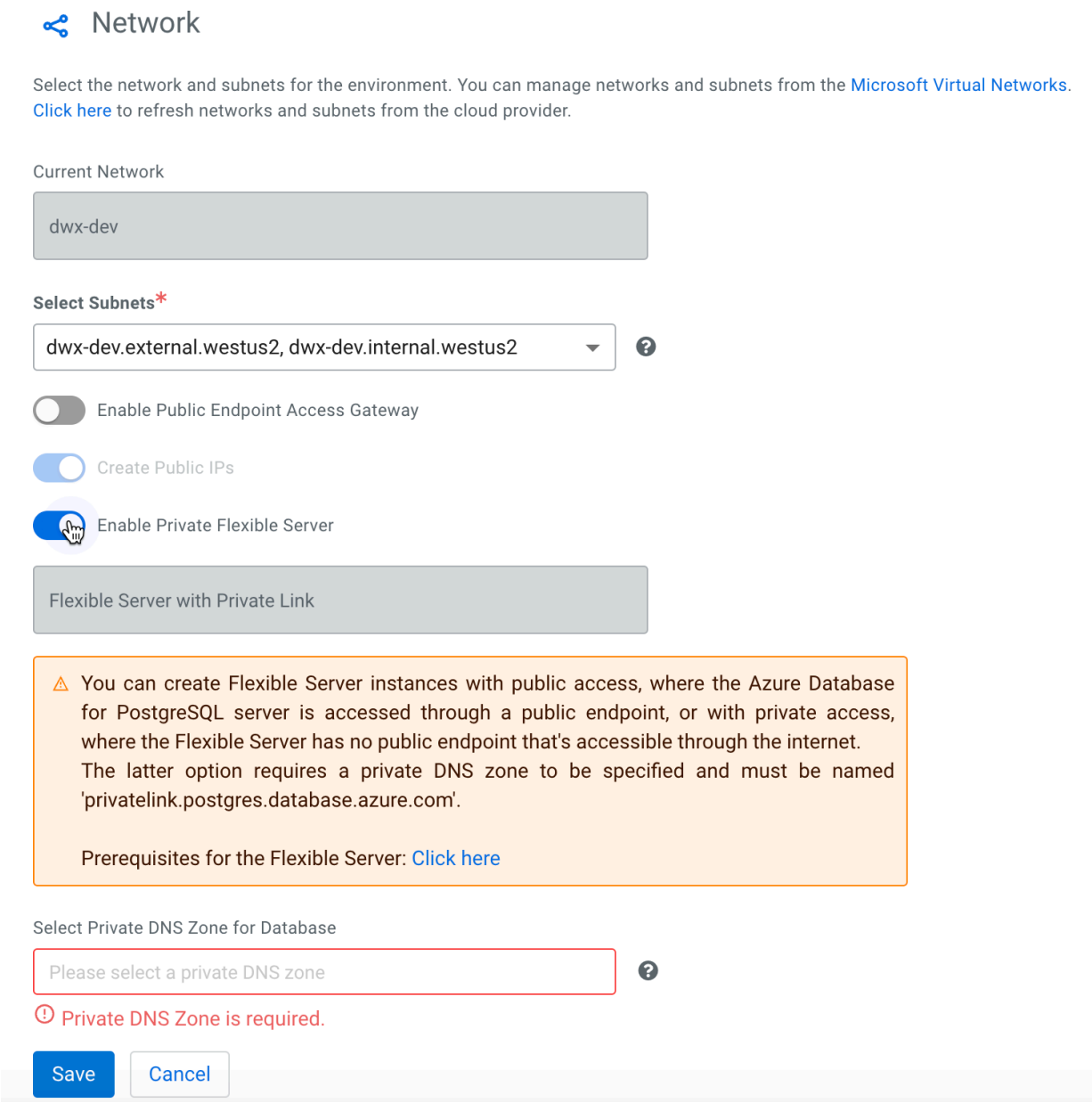

6. Select a Private DNS Zone.

   The name of the Private DNS Zone must be privatelink.postgres.database.azure.com. You can only select from already existing Private DNS Zones. The Create new Private DNS Zone option is not available here.
7. Click Save.

   The Private Endpoint becomes enabled and the Private DNS Zone is displayed on the Network card of the Summary tab.

   Note: This does not affect the existing clusters of the environment, only the newly created clusters will utilize the Private Endpoint.
8. Click the Upgrade database button inside the warning message at the top of the screen.

9. Once the Data Lake database is updated, check for the Cloudera Data Hub clusters for that Data Lake, if there is any database upgrade notification and perform the database upgrade as described above.

   Note: The database upgrade needs to be performed on every Data Lake and Cloudera Data Hub cluster separately, one by one.

Upgrading databases using the CDP CLI

It is possible to upgrade from Azure Single Server to Azure Flexible Server using the CDP CLI. You can use the regular Cloudera Data Hub and Data Lake database upgrade commands to do this. The upgade to PostgreSQL 14 automatically upgrades the database from Single Server to Flexible Server.

Data Lake database upgrade

Before you begin, make sure the prerequisites listed in Upgrading Azure Single Server to Flexible Server are met.

You can upgrade Data Lake database using the cdp datalake start-database-upgrade CLI command.

The --target-version parameter is optional. If you do not provide it, the database will be upgraded to PostgreSQL 14.

```
cdp datalake start-database-upgrade --help --form-factor public
NAME
       start-database-upgrade  -  Upgrades the database of the Data Lake cl
uster.

DESCRIPTION
       This command initiates the upgrade of the database  of  the  Data  La
ke cluster.
SYNOPSIS
            start-database-upgrade
          --datalake <value>
          [--target-version <value>]
          [--cli-input-json <value>]
          [--generate-cli-skeleton]
OPTIONS
       --datalake (string)
          The name or CRN of the Data Lake.
       --target-version (string)
          The database engine major version to upgrade to.

          Possible values:

          o VERSION_14
```

Cloudera Data Hub database upgrade

Before you begin, make sure the prerequisites listed in the *Prerequisites* section of Upgrading Azure Single Server to Flexible Server are met.

You can perform Cloudera Data Hub database upgrade using the cdp datahub start-database-upgrade CLI command.

The --target-version parameter is optional. If you do not provide it, the database will be upgraded to PostgreSQL 14

```
cdp datahub start-database-upgrade --help --form-factor public
NAME
       start-datahub-upgrade  -  Upgrades the database of the Data Hub clust
er.
DESCRIPTION
       This command initiates the upgrade of the database  of  the  Data  H
ub cluster.
SYNOPSIS
```

```
            start-database-upgrade
          --datahub <value>
          [--target-version <value>]
          [--cli-input-json <value>]
          [--generate-cli-skeleton]

OPTIONS
       --datahub (string)
          The name or CRN of the Data Hub.

       --target-version (string)
          The database engine major version to upgrade to.

          Possible values:

          o VERSION_14
```

Post-requisites after upgrading databases

Complete the following tasks after upgrading from Azure Single Server to Azure Flexible Server.

- Start the Cloudera Data Warehouse Database Catalogs and Virtual Warehouses. For each virtual warehouse, Cloudera recommends that you start, stop, and start again. This will completely refresh the Data Lake details for the virtual warehouse.

  Warning: You must be aware of an issue where the Database Catalog goes into an error state after upgrading to Azure Fexible Server. For more information about this known issue, see the *Cloudera Data Warehouse 1.9.3-b166 (released December 5, 2024) release notes*.

Related Information

Cloudera Data Warehouse Release Notes

Configuring a CMK for data encryption in Azure Database for PostgreSQL Flexible Server

You can optionally use a Customer Managed Key (CMK) for encrypting data in the Azure Flexible Server database instance used by Cloudera.

As described in Adding a customer managed encryption key to a Cloudera environment running on Azure, by default Cloudera clusters are encrypted with server-side encryption (SSE) using Platform Managed Keys (PMK) and Cloudera allows you to provide an existing CMK for encrypting Cloudera clusters.

When using the CMK for encrypting Cloudera clusters, you can also use that same CMK for encrypting the Azure Flexible Server database instance used by Cloudera. If you would like to do this, in addition to the typical CMK prerequisites, you should create a managed identity with specific permissions and then after providing the other CMK-related parameters (CMK resource group and URL) during Cloudera environment registration on Azure provide that managed identity during Azure environment registration in Cloudera.

Azure prerequisites

You should first meet the CMK-related prerequisites described in Azure Requirements: Customer managed encryption keys (add additional credential permissions and create a key vault and vault key).

In addition to that, you should create a managed identity as described in Managed identity for encrypting Azure Database for PostgreSQL Flexible Server.

Creating an environment with a CMK for encrypting Flexible Server

Steps

For CDP UI

Follow the usual steps for creating a Cloudera environment on Azure and make sure to do the following:

1. In the Register Environment wizard, on the Region, Networking and Security page find the Customer-Managed Keys section.
2. Click Enable Customer-Managed Keys.
3. In the same section, under Select Resource group select the resource group where your CMK is located.
4. Provide the URL of the key value where the CMK resides. This is the same as the key identifier that you can copy directly from Azure Portal.
5. Under Managed identity for encryption, select the managed identity created as a prerequisite.

For CDP CLI

Add the following CDP CLI parameters to the cdp environments create-azure-environment command:

```
--encryption-key-resource-group-name <CMK_RESOUCE_GROUP_NAME>
--encryption-key-url <KEY_RESOUCE_ID>
--encryption-user-managed-identity <EXISTING_MANAGED_IDENTITY>
```

Note: The former --user-managed-identity parameter has been deprecated. Use the --encryption-user-managed-identity instead, as shown above.

The --encryption-key-resource-group-name is not needed in some cases, as described in CDP CLI steps listed in Adding a customer managed encryption key to a Cloudera environment running on Azure.

While the first two parameters are required for using CMK for Cloudera in general the third parameter must be added in addition for encrypting a Flexible Server. It should be specified as in the following example:

```
--encryption-user-managed-identity /subscriptions/3ddda1c7-d1f5-4e4b-ac8
1-0523f483b3b1/resourcegroups/test-daily-rg/providers/Microsoft.ManagedI
dentity/userAssignedIdentities/test-adminIdentity
```

Updating an existing environment to add the managed identity

Steps

For CDP UI

The steps for adding the managed identity are similar as those described in Set a CMK for an existing Azure environment, just in the Managed identity for encryption field, a managed identity should be provided in addition to the Encryption Key Resource Group and the Encryption Key URL.

The managed identity can be either added when you are adding the CMK, or can be added to an environment already configure with CMK. In the latter case, it is enough to populate the Managed identity for encryption field, the other fields are already populated.

For CDP CLI

The steps for adding the managed identity are similar as those described in Set a CMK for an existing Azure environment, just the --encryption-user-managed-identity should be specified in addition.

The managed identity can be either added when you are adding the CMK, or can be added to an already existing CMK setup. In both cases, use the full command with all the listed parameters.

Use the following CDP CLI command:

```
cdp environments update-azure-encryption-resources \
 --environment <ENVIRONMENT_NAME> \
 --encryption-key-url <VAUL_KEY_URL> \
```

```
  --encryption-key-resource-group-name <RG_NAME> \
  --encryption-user-managed-identity <EXISTING_MANAGED_IDENTITY>
```

Enabling Single Server

During environment registration in Cloudera, the Flexible Server in public service mode is used by default, but you can specify to use the Single Server (deprecated) or Single Server with Private Link (deprecated) options.

Steps

For CDP UI

During environment registration in Cloudera, in the Database section, enable one of the Single Server options by selecting Single Server (deprecated) or Single Server with Private Link (deprecated) from the dropdown. The Flexible Server option is pre-selected by default.

For CDP CLI

1. Register an Azure environment using the cdp create-azure-environment CDP CLI command as usual.
2. Set IDBroker mappings as usual using the cdp environments set-id-broker-mappings command.
3. Create a Data Lake using the cdp create-azure-datalake CDP CLI command, including a reference to the database HA type. For example:

```
cdp datalake create-azure-datalake
  --datalake-name <datalake-name>
...
  --database-type SINGLE_SERVER
```

The following table explains the required parameters:

Parameter	Description	Possible values
database-type (string)	The type of the azure database. FLEXIBLE_SERVER (recommended) is the next generation managed PostgreSQL service in Azure that provides maximum flexibility over your database, built-in cost-optimizations. SINGLE_SERVER is a fully managed database service with minimal requirements for customizations of the database.	• FLEXIBLE_SERVER • SINGLE_SERVER

4. Create a Cloudera Data Hub using the create-azure-cluster CDP CLI command, including a reference to the database HA type and the database PostgreSQL engine version. For example:

```
cdp datahub create-azure-cluster
 --cluster-name <marketplace-data-hub>
...
 --datahub-database HA
 --database-type SINGLE_SERVER
```

The following table explains the required parameters:

Parameter	Description	Possible values
datahub-database (string)	Represents the database availability type.	• HA (This means, same zone HA) • NON_HA

database-type (string)	The type of the azure database. FLEXIBLE_SERVER (default value) is the next generation managed PostgreSQL service in Azure that provides maximum flexibility over your database, built-in cost-optimizations. SINGLE_SERVER is a fully managed database service with minimal requirements for customizations of the database. If you do not specify this parameter, FLEXIBLE_SERVER is used by default.	• FLEXIBLE_SERVER • SINGLE_SERVER

Troubleshooting Flexible Server

Refer to this documentation for troubleshooting Azure environments using Flexible Server.

Capacity limitation related warning during Flexible Server upgrade

Problem:

A capacity related validation warning is shown during the upgrade from Azure Single Server to Flexible Server.

Solution:

Perform the following steps:

1. File an Azure Support Ticket, specifying the Azure subscription and region that is used for the Cloudera setup. Microsoft Azure support will ensure that the necessary capacity is allocated to unblock provisioning Flexible Server same-zone HA database instances.
2. Once the capacity has been increased, retry the Single Server to Flexible Server upgrade operation.

Missing default outbound access

Problem:

Data Lake provisioning failed due to missing outbound access.

Solution:

By default, Single Server utilizes service endpoints (Azure Database for PostgreSQL server - Microsoft.Sql) that provide secure and direct connectivity to DB service over an optimized route over the Azure backbone network from the Virtual Machines, but that is not the case with Flexible Server. This means that there are certain cases when you have to explicitly ensure that outbound (i.e. egress) network connectivity from the selected Virtual Network is set up using NAT-Gateway, UDR, or similar. If you are using a public Flexible Server, make sure that port 5432 is open for outbound traffic.

This requirement is applicable in the following conditions:

- An existing network is being used.
- Create Public IP is disabled.
- Public Endpoint Gateway is disabled.

The lack of default outbound access can be mitigated by using private setup or using public IPs instead.

For more information, see Default outbound access in Azure.

Enabling a private endpoint for Azure Postgres

By default Cloudera uses service endpoints, but you can select to use private endpoints instead. During environment registration you can optionally select the “Create Private Endpoint” option to use private endpoints instead of using a service endpoint. Currently, only one service or private endpoint is used, for Azure Postgres.

The endpoint type - either service or private endpoints - and whether to use your own private DNS zones or have Cloudera create them are decided at environment creation. All services capable of that endpoint type will use the selected endpoint type - this currently means only the Postgres server.

Note:

The endpoint type cannot be changed after the environment was created.

By default service endpoints are used, and in order to use private endpoints the user has to explicitly enable the “Create Private Endpoints“ option from either the Cloudera UI or CDP CLI. There is no option to not use either. Once you have turned on private endpoints then you can specify your own private DNS zone if you wish.

Prerequisites

You should also decide if you would like to use your own Azure private DNS zone or have Cloudera create an Azure private DNS zone for you, and meet the related prerequisites. Review the requirements described in Private endpoint for Azure Postgres.

Enabling private endpoints

You can enable private endpoints and select to use a Cloudera-managed Azure private DNS zone or your own Azure private DNS zone during Azure environment registration in the Cloudera Management Console.

Required role: EnvironmentCreator

Steps

For CDP UI

When registering an environment specify the following:

1. In the Resource group section, select to use your pre-created resource group.
2. In the Network section of the register environment wizard, enable the "Create Private Endpoints" option.

3. A new drop down option appears. You have two options:
   a. Select an existing private DNS zone.
   b. If you would like Cloudera to create the private DNS zone, select “Create new private DNS zone”.

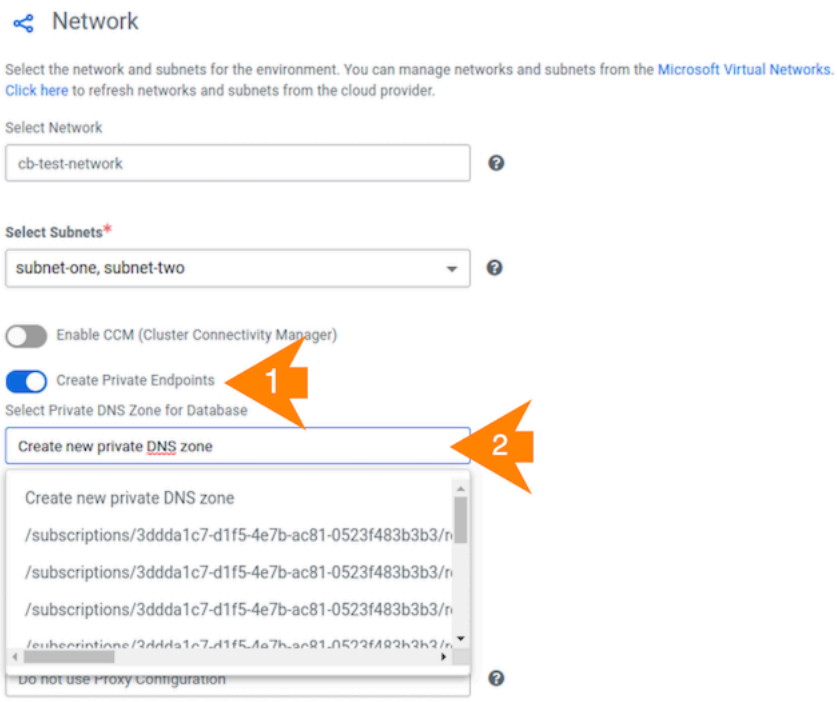

4. Provide other Azure environment parameters as usual and complete environment registration.

For CDP CLI

The parameter createPrivateEndpoints defines that Azure Postgres will be configured with a Private Endpoint and a Private DNS Zone. When this option is disabled, Azure Service Endpoints are created. This option is disabled by default, so Azure Service Endpoints are used by default.

This is how to specify this via CDP CLI when running the cdp environments create-azure-environment command:

```
--create-private-endpoints
--no-create-private-endpoints
```

If you would like to specify your own Azure private DNS zone, you should additionally include the databasePrivateDnsZoneId as part of existing-network-params:

```
--existing-network-params networkId=<NETWORK_ID>,resourceGroupName=<NETWORK_RG_NAME>,subnetIds=<SUBNET_1>,<SUBNET_2>,databasePrivateDnsZoneId=<PRIVATE_DNS_ZONE_RESOURCE_ID>
```

The <PRIVATE_DNS_ZONE_RESOURCE_ID> that you need to provide should look similar to:

```
/subscriptions/<SUBSCRIPTION_ID>/resourceGroups/<RESOURCE_GROUP_NAME>/providers/Microsoft.Network/privateDnsZones/privatelink.postgres.database.azure.com
```

For example:

```
/subscriptions/a9a10f9c-5323-4fd2-8s14-747b2d68784c/resourceGroups/my-
resource-group/providers/Microsoft.Network/privateDnsZones/contoso.com
```

Related Information

Register an Azure environment from the Cloudera UI

Register an Azure environment from the CDP CLI

Deploying Cloudera in multiple Azure availability zones

You can optionally choose to deploy Data Lake, FreeIPA, and Cloudera Data Hub clusters across multiple availability zones (multi-AZ). With multi-AZ support, newly created Azure environments, enterprise Data Lakes and Cloudera Data Hub clusters using HA templates can be deployed across multiple availability zones of the selected Azure region. This provides fault tolerance during the extreme event of an availability zone outage.

Note: Only Enterprise Data Lakes can use multi-AZ; other Data Lake shapes do not support it. Enterprise Data Lake is only available in Cloudera Runtime 7.2.17 and newer; therefore, multi-AZ for Cloudera on Azure is only available in Cloudera Runtime 7.2.17 and newer.

Each Azure region has multiple availability zones, which act as failure domains, preventing small outages from affecting entire regions. If you choose to deploy your Cloudera environment (FreeIPA and Data Lake) and Cloudera Data Hub clusters across multiple availability zones, each of these components is spread across three availability zones, providing high availability and fault tolerance. This is illustrated in the following diagram:

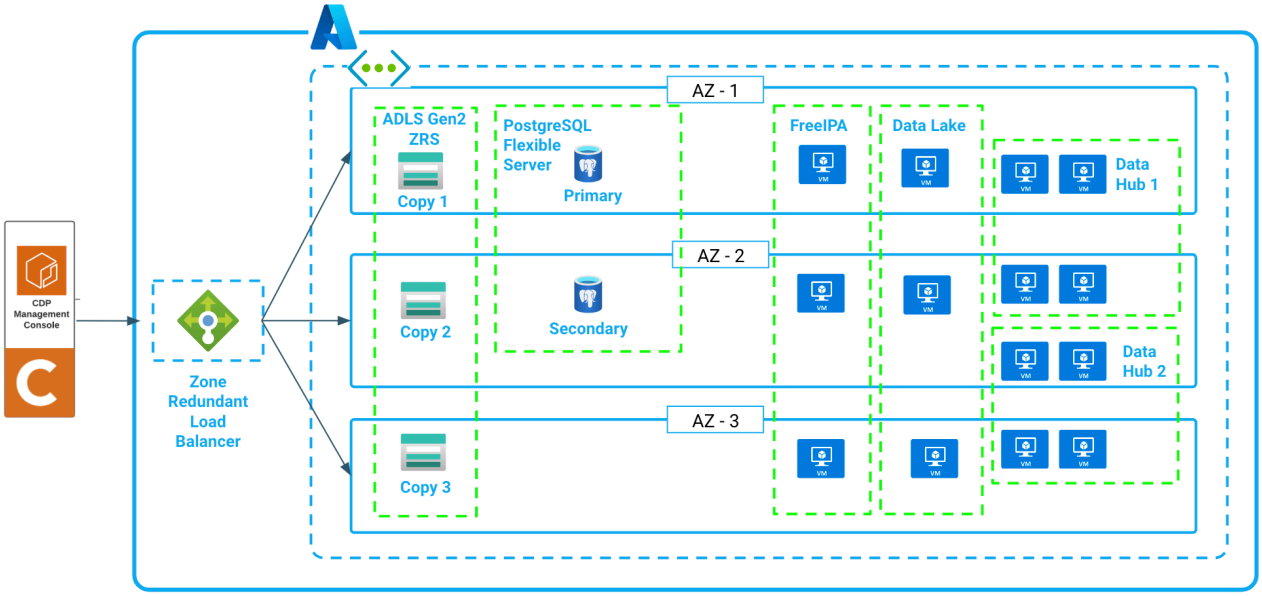

With the multi-AZ option enabled, your services are deployed in the following way:

- Azure environments are always created with three FreeIPA servers, deployed on virtual machines spread across three available zones.
- In an Azure Enterprise Data Lake each host group is configured so that virtual machines of all critical services are spread across three available zones.
- In HA Cloudera Data Hub clusters, virtual machines of each host group are evenly spread across three availability zones, following a round-robin logic.

When a zone failure happens and a cluster needs to be repaired, the replacement VMs are always provisioned in the same subnet and availability zone as the old ones since the detached disks can only be reattached to a VM in the same availability zone. This means that if there is an availability zone outage, cluster repair is not possible.

By default, If you do not enable multi-AZ, Cloudera and Cloudera customers do not have visibility into how Azure distributes VMs across availability zones, because the Azure Portal or CLI do not provide this information.

When creating Cloudera Data Hub clusters via CDP CLI, you have the option to specify the AZ, which, in addition to allowing you to select the AZs that should be used, allows you to set up AZ targeting, where all nodes of the cluster are placed on the same AZ. This enables creating disaster recovery scenarios, where a primary and secondary cluster are running in different AZs. If an AZ outage occurs and the primary cluster is lost, it is guaranteed that the secondary cluster is not impacted.

Use cases

A multi-AZ Data Lake and FreeIPA constitute a resilient environment that provides a solid basis for multi-AZ Cloudera Data Hub clusters and Cloudera data services. Cloudera Data Hub clusters and Cloudera data services depend on the FreeIPA instance in the Data Lake to provide DNS resolution. Deploying FreeIPA across multiple availability zones ensures that critical DNS resolution is available in the event of an availability zone outage. Furthermore, a medium duty or enterprise Data Lake provides high availability, and additional compute and memory resources for key SDX services and is recommended for production workloads.

Deploying your Cloudera Data Hub clusters across multiple availability zones is key if your mission-critical applications depend on HBase and Kafka. Multiple availability zone deployment for operational workloads is considered best practice by the cloud vendors. It ensures that your applications can continue to run in the event of an availability zone outage.

When an entire availability zone fails, HBase automatically rebalances regions among the remaining instances in the cluster to maintain availability. The write-ahead log (WAL), which is replicated across the three availability zones is automatically replayed by the newly assigned region servers in other availability zones to ensure writes to the database are not lost.

When using the multi availability zone feature, Cloudera ensures that Kafka replicates partitions across brokers in different availability zones. During an availability zone failure this ensures that no data is lost and applications can continue to access the data they need. Cruise Control, which is deployed alongside every Kafka cluster in Cloudera Public Cloud, detects that topics need to be rebalanced to the remaining brokers. Once the availability zone is back online, you can repair your Kafka cluster, restoring the initial broker distribution across availability zones. Afterwards Cruise Control kicks in and ensures that all topic partitions are balanced across the cluster.

Limitations

The following limitations apply when deploying a multi-AZ Cloudera:

- When an AZ is down, you cannot create a new Cloudera Data Hub cluster, and create or activate Cloudera data services within the environment. Existing workloads will continue to work.
- When an AZ is down, you cannot resize, stop, or restart Cloudera Data Hub clusters.
- Non-AZ environments or clusters cannot be converted to multi-AZ.

Azure requirements

In order to use multi-AZ, you should meet the following Azure requirements:

1. The Azure region that you select should support setting up Azure PostgreSQL Flexible Server in Zone-Redundant HA mode and also the instance types to be used. See Flexible Server Azure Regions.
2. The ADLS Gen2 storage account should be created as zone-redundant storage (ZRS). To specify ZRS via Azure CLI during storage account creation, the --sku option should be set to Standard_ZRS. Below is a sample Azure CLI command:

```
azure % az storage account create \
 --name test-storage \
 --resource-group rg-test-rg \
 --access-tier Cool \
 --allow-blob-public-access false \
 --allow-cross-tenant-replication false \
```

```
--allow-shared-key-access true \
--enable-hierarchical-namespace true \
--skuStandard_ZRS
```

Register a multi-AZ environment

You can register a multi-AZ AWS environment via Cloudera UI or CDP CLI. You may choose to enable multi-AZ for Data Lake only or for FreeIPA only. There is no requirement to enable both.

Steps

For CDP UI

Register your environment as usual, just make sure to do the following:

1. On the Data Access and Data Lake Scaling page:
   a. Select to use the Enterprise Data Lake.
   b. On the same page, scroll down and in the bottom of the page enable the Advanced Options.
   c. In the Network and Availability section enable the Enable Multiple Availability Zones for Data Lake toggle button in order to enable multi-AZ for Data Lake. The option is disabled by default. The option only appears when the Enterprise Data Lake is selected.
2. On the Region, Networking, and Security page:
   a. Scroll down and in the bottom of the page enable the Advanced Options.
   b. In the Network and Availability section enable the Enable Multiple Availability Zones for Data Lake toggle button in order to enable multi-AZ for FreeIPA. The option is disabled by default.
3. Finish registering your environment as usual.

For CDP CLI

Use the following CDP CLI commands to register an environment with a multi-AZ Data Lake and FreeIPA:

1. Register an Azure environment using thecdp environments create-azure-environment command and include multiAz=true in the--free-ipa parameter as shown in this example:

   ```
   cdp environments create-azure-environment \
   --environment-name test-env \
   ...
   --free-ipa instanceCountByGroup=3,multiAz=true \
   ```

   If you do not include the multiAz=true, the default AZ distribution will be used.

   You can also optionally include the--availability-zonesparameter to select the specific availability zones that should be used. Valid values for availability zones are 1,2 and 3. If this parameter is not provided, all AZs are used. For example:

   ```
   cdp environments create-azure-environment \
   --environment-name test-env \
   ...
   --free-ipa instanceCountByGroup=3,multiAz=true \
   --availability-zones 1 2
   ```

2. Set IDBroker mappings as usual using thecdp environments set-id-broker-mappingscommand.
3. Create a Data Lake using the cdp datalake create-azure-datalake command and adding the --multi-az parameter. For example:

   ```
   cdp datalake create-azure-datalake \
   --datalake-name test-dl \
   --environment-name test-env \
   ...
   --scale ENTERPRISE \
   ```

```
 --runtime 7.2.17 \
 --multi-az
```

Create a multi-AZ Cloudera Data Hub cluster

You can create multi-AZ Cloudera Data Hub clusters within any existing environment. Detailed steps are provided below.

Prerequisites

You can create a multi-AZ Cloudera Data Hub cluster in a multi-AZ environment only. If you are trying to create a multi-AZ Cloudera Data Hub cluster in an environment that uses the default AZ distribution, you need to first edit that environment and add AZs to it.

Steps

For CDP UI

To enable multi-AZ when creating a Cloudera Data Hub cluster on Azure, navigate to the Advanced Options > Network And Availability and in the “Azure Availability Zones” section click the toggle button next to Enable using multiple availability zones.

For CDP CLI

You can create a multi-AZ Cloudera Data Hub cluster by adding the --multi-az option to the Cloudera Data Hub cluster creation command.

In the --instance-groups parameter, you can optionally include the availabilityZones to select the specific availability zones that should be used. If this parameter is not provided, all three AZs are used. For example:

```
cdp datahub create-azure-cluster \
 --cluster-name test-cluster1 \
 --environment-name test-env \
 --cluster-template-name "7.2.17 - Data Engineering: Apache Spark, Apache
Hive, Apache Oozie" \
 --multi-az \
```

```
cdp datahub create-azure-cluster \
 --cluster-name test-cluster1 \
 --environment-name test-env \
 --cluster-template-name "7.2.17 - Data Engineering: Apache Spark, Apache
Hive, Apache Oozie" \
 --multi-az \
 --instance-groups
          nodeCount=1,instanceGroupName=compute,instanceGroupType=CORE,ins
tanceType=Standard_D5_v2,rootVolumeSize=100,attachedVolumeConfiguration=
\[\{volumeSize=100,volumeCount=0,volumeType=StandardSSD_LRS\}\],recovery
Mode=MANUAL,availabilityZones=\[1,2\]
          nodeCount=0,instanceGroupName=gateway,instanceGroupType=CORE,
instanceType=Standard_D8_v3,rootVolumeSize=100,attachedVolumeConfigurati
on=\[\{volumeSize=100,volumeCount=1,volumeType=StandardSSD_LRS\}\],recov
eryMode=MANUAL,availabilityZones=\[2,3\]
          nodeCount=1,instanceGroupName=master,instanceGroupType=GATEWA
Y,instanceType=Standard_D16_v3,rootVolumeSize=100,attachedVolumeConfigur
ation=\[\{volumeSize=100,volumeCount=1,volumeType=StandardSSD_LRS\}\],re
coveryMode=MANUAL,availabilityZones=\[1,2,3\]
          nodeCount=3,instanceGroupName=worker,instanceGroupType=CORE,inst
anceType=Standard_D5_v2,rootVolumeSize=100,attachedVolumeConfiguration=\
[\{volumeSize=100,volumeCount=1,volumeType=StandardSSD_LRS\}\],recoveryM
ode=MANUAL,availabilityZones=\[1,3\]
```

Modify environment's AZs for new Cloudera Data Hub clusters

You can modify an environment's AZs. In this case, the added AZs will only be used for new Cloudera Data Hub clusters. The environment clusters and existing Cloudera Data Hub clusters running in it will continue to use the original AZs.

Steps

For CDP UI

1. In the Cloudera Management Console, navigate to environment details > Summary.
2. Scroll down to the Region section.
3. Under Select Azure availability zones, select the availability zones:

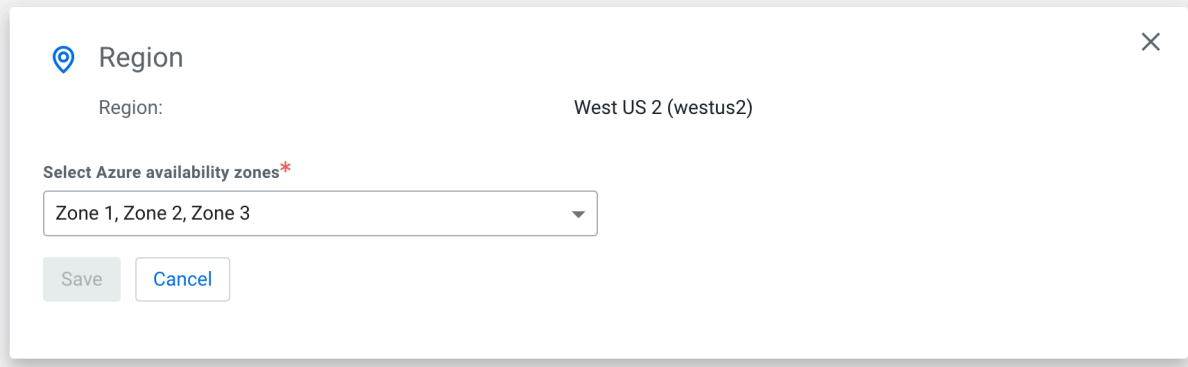

4. Click Save.

For CDP CLI

Below is the CLI command for updating AZs of an existing environment:

```
cdp environments update-azure-availability-zones \
--environment-name test-env \
--availability-zones12 3
```

Restricting access for Cloudera services that create their own security groups on Azure

The security groups that you select to use during environment registration are only used for the Data Lake, FreeIPA, Cloudera Data Hub clusters, and Operational Databases running in that environment. The Kubernetes-based Cloudera services (Cloudera Data Warehouse and Cloudera AI) create their own security groups with rules that should be restricted separately.

The following table explains where and when you can restrict these rules:

Note:

If you do not restrict these endpoints, Cloudera defaults to opening access to all (0.0.0.0/0).

CDP service	Type of access that can be restricted	When and where to restrict	Link to related documentation
Cloudera Data Engineering	Admin access to Kubenetes endpoints can be restricted.	Restrict admin access to Kubernetes endpoints during enabling Data Engineering via the Whitelist IPs parameter.	Enabling Cloudera Data Engineering

CDP service	Type of access that can be restricted	When and where to restrict	Link to related documentation
Cloudera Data Warehouse	Both admin access to Kubernetes endpoints and end user access are always set to the same range that can be set in environment activation settings. While the access to the Kubernetes endpoints is a combination of the Cloudera Control Plane's CIDR and your CIDR provided in environment activation settings, the access to the end user access points (JDBC, UI) is only your CIDR provided in environment activation settings.	In Cloudera Data Warehouse environment's activation settings.	Editing environment details
Cloudera AI	There are two separate options, one for admin access to Kubernetes endpoints and another for end user access.	During Cloudera AI workbench provisioning, under Network Settings: • The Load Balancer Source Ranges parameter can be used to restrict end user access. • Selecting the checkmark Restrict access to Kubernetes API server to authorized IP ranges allows you to restrict admin access to Kubernetes endpoints.	Provisioning Cloudera AI workbenches

Configure lifecycle management for logs on Azure

To avoid unnecessary costs related to ALS Gen2 cloud storage, you should create lifecycle management rules for your cloud storage container used by Cloudera for storing logs so that these logs get deleted once they are no longer useful.

Some examples of Cloudera logs stored in cloud storage are: cloudera server logs, cloudera agent logs, autossh logs, freeipa logs, ranger audit logs, datahub services logs, datalake logs, cm management services logs, and so on. These logs are mostly useful for troubleshooting, so they can be periodically deleted.

Azure allows you to set up lifecycle management rules for your ADLS cloud storage. For example, you can set a specified expiration period for a cloud storage location so that files in that location get deleted automatically on a scheduled basis. Cloudera recommends that you do this for the cloud storage location that you provided to Cloudera for log storage.

Consider the following when setting up lifecycle management rules:

- As logs and data locations may overlap with each other (in case the same bucket or container is used for both), ensure to use the correct path prefixes in order to delete only the logs. The prefixes are listed below.
- When setting an expiration period, consider how long you would like to keep the logs to allow enough time for troubleshooting. For example, in case your Data Lake, FreeIPA or Cloudera Data Hub cluster is ever down, you should be able to access the logs for troubleshooting.

To set up lifecycle management:

- Review the prefixes listed below.
- Follow the instructions in Optimize costs by automating Azure Blob Storage access tiers.

Prefixes based on Azure environment's logs location base

Prior to creating lifecycle management rules, review this information to ensure that you use the correct path.

Note: Path logic changed in February 2021. Starting in February 2021, the path automatically contains the cluster-logs folder as a peer of the cluster-backups folder, creating a better structural separation between logs and backups.

	The “cluster-logs” prefix is automatically generated if a bucket name without any subdirectories is used as logs location	The “cluster-logs” prefix is automatically generated if subdirectories are provided	If your environment was registered prior to February 2021: If you defined a sub-directory, then that subdirectory is used instead of “cluster-logs”
Logs location provided during environment registration	**abfs:// mycontainer@myaccount.dfs.core.windows.n**	**abfs://mycontainer/my-dl@myaccount.dfs.core.window**	**abfs://mycontainer/my-dl@myaccount.dfs.core.window**
FreeIPA prefix for lifecycle rule	mycontainer/cluster-logs/freeipa	mycontainer/my-dl/cluster-logs/freeipa	mycontainer/my-dl/freeipa
DataLake prefix for lifecycle rule	mycontainer/cluster-logs/datalake	mycontainer/my-dl/cluster-logs/datalake	mycontainer/my-dl/datalake
DataHub prefix for lifecycle rule	mycontainer/cluster-logs/datahub	mycontainer/my-dl/cluster-logs/datahub	mycontainer/my-dl/datahub

Related Information

Enabling environment telemetry

Cloudera Management Console

GCP Environments

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Working with GCP environments 4

Introduction to Google Cloud environments 4

Register environment (UI) 5

Register environment (CLI) 8

Enabling admin and user access 12

Understanding environment UI options 12

Monitoring an environment 13

Environment status options 14

Stop and restart an environment 16

Delete an environment 17

Change environment's credential 18

Defining custom tags 18

Adding a customer managed encryption key for GCP 20

Working with GCP environments

Refer to the following documentation to learn about creating and managing GCP environments in Cloudera:

Related Information

Managing provisioning credentials for GCP
Managing Data Lakes
Managing FreeIPA

Introduction to Google Cloud environments

In Cloudera, an environment is a logical subset of your cloud provider account including a specific virtual private network. You can register as many environments as you require.

Registering your GCP environment in Cloudera provides Cloudera with limited access to your GCP account and identifies a set of resources in your GCP account that Cloudera services can access. Once you've registered your GCP environment in Cloudera, you can start provisioning Cloudera resources such as clusters, which run on the physical infrastructure in an GCP data center.

The following diagram enumerates the components of a GCP environment:

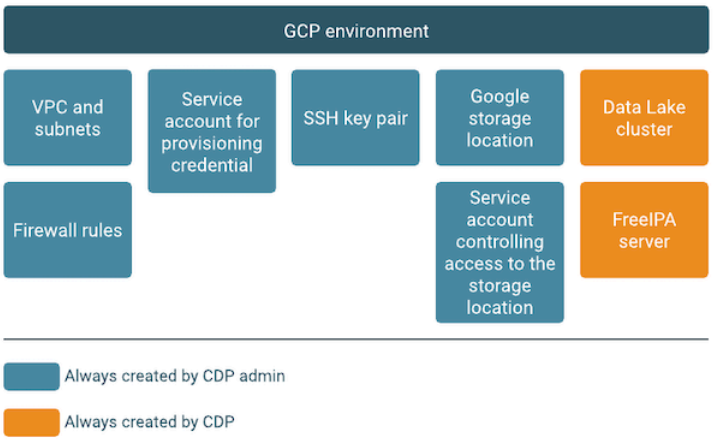

The diagram illustrates all major user-created and Cloudera-created components of an environment:

- The items in dark blue boxes must be pre-created by your Cloudera administrator prior to environment registration and then specified when registering an environment.
- The items in orange boxes are automatically provisioned on GCP by Cloudera as part of environment provisioning.

Note: The items that are user-created don't get terminated during environment deletion.

As shown in the diagram, an environment consists of the following resources:

Environment component	Description
Virtual network with subnets	An environment corresponds to one specific VPC network and one or more subnets in which Cloudera resources are provisioned.

Environment component	Description
Firewall rules	Firewall rules (similar to security groups on other cloud providers) act as a virtual firewall for your instances to control inbound and outbound traffic. All VM instances provisioned within an environment use your specified security access settings allowing inbound access to your instances from your organization's computers.
Service account for provisioning credential	Cloudera uses a provisioning credential for authorization to provision resources (such as compute instances) within your cloud provider account. In GCP, credential creation involves creating a service account, assigning a set of minimum IAM permissions to it, and providing Cloudera with the access key generated for the service account.
SSH public key	When registering an environment on a public cloud, a Cloudera administrator provides an SSH public key. This way, the administrator has root-level access to the Data Lake instance and Cloudera Data Hub cluster instances.
Google storage location and a service account controlling access to it	When registering an environment, you must provide a Google storage location for storing: • All workload cluster data • Cluster service logs and Ranger audits Furthermore, you must create and assign a service account on the scope of this storage location so that Cloudera can access it.
Data Lake cluster	A data lake is automatically provisioned when an environment is created. It provides a mechanism for storing, accessing, organizing, securing, and managing data.
FreeIPA server	A FreeIPA server is automatically provisioned when an environment is created. It is responsible for synchronizing your users and making them available to Cloudera services, Kerberos service principal management, and more.

Once your environment is running, you can provision Cloudera Data Hub clusters in it.

Register a GCP environment from the Cloudera UI

Once you've met the Google Cloud cloud provider requirements, register your GCP environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in GCP requirements.

Required role: EnvironmentCreator

Steps

1. Navigate to the Cloudera Management Console > Environments > Register environment.
2. On the Register Environment page, provide the following information:

Parameter	Description
General Information	
Environment Name (Required)	Enter a name for your environment. The name: • Must be between 5 and 28 characters long. • Can only include lowercase letters, numbers, and hyphens. • Must start with a lowercase letter.
Description	Enter a description for your environment.

Parameter	Description
Select Cloud Provider (Required)	Select Google Cloud.
Google Cloud Platform Credential (Required)	
Select Credential	Select an existing credential or select Create new credential. For instructions on how to create a credential for Google Cloud, refer to Create a provisioning credential for GCP.

3. Click Next.
4. On the Data Access and Data Lake Scaling page, provide the following information:

Parameter	Description
Data Lake Settings	
Data Lake Name (Required)	Enter a name for the Data Lake cluster that will be created for this environment. The name: • Must be between 5 and 100 characters long • Must contain lowercase letters • Cannot contain uppercase letters • Must start with a letter • Can only include the following accepted characters are: a-z, 0-9, -.
Data Lake Version (Required)	Select Cloudera Runtime version that should be deployed for your Data Lake. The latest stable version is used by default. All Data Hub clusters provisioned within this Data Lake will be using the same Cloudera Runtime version. Note: Google Cloud environments can only be provisioned in Cloudera with Cloudera Runtime version 7.2.8 or newer.
Data Access and Audit	
Assumer Service Account (Required)	Select the IDBroker service account created in Minimum setup for cloud storage.
Storage Location Base (Required)	Select the Google Storage location created for data in Minimum setup for cloud storage.
Data Access Service Account (Required)	Select the Data Lake Admin service account created in Minimum setup for cloud storage.
Ranger Audit Service Account (Required)	Select the Ranger Audit service account created in Minimum setup for cloud storage.
IDBroker Mappings	We recommend that you leave this out and set it up after registering your environment as part of Onboarding Cloudera users and groups for cloud storage.
Scale (Required)	Select Data Lake scale. By default, “Light Duty” is used. For more information on Data Lake scale, refer to Data Lake scale.

5. Click on Advanced Options to make additional configurations for your Data Lake. The following options are available:

Parameter	Description
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Machine type families.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on a specific Data Lake host group. For more information, see Recipes.

6. Click Next.

7. On the Region, Networking and Security page, provide the following information:

Parameter	Description
Region	
Select Region (Required)	Select the region where your VPC network is located.
Select Zone (Required)	Select a zone within the selected region.
Network	
Use shared VPC	This option is disabled by default. Enable this if you would like to use your existing shared VPC. Next enter: • Host project ID • Network name • Subnet name(s). If providing multiple, provide a comma separated list.
Select Network (Required)	Select the existing VPC network that you created as a prerequisite in the VPC network and subnets step. All Cloudera resources will be provisioned into this network.
Select Subnets (Required)	Select at least one existing subnet.
Create Public IPs	This option is disabled by default when Cluster Connectivity Manager is enabled and enabled by default when Cluster Connectivity Manager is disabled.
Proxies	Select a proxy configuration if previously registered. For more information refer to Setting up a proxy server.
Security Access Settings	
Select Security Access Type (Required)	You have two options: • Do not create firewall rule: If you are using a shared VPC you can set the firewall rules directly on the VPC. If you did so, you can select this option. • Provide existing firewall rules: If not all of your firewall rules are set directly on the VPC, provide the previously created firewall rules for SSH an UI access. You should select two existing firewall rules, one for Knox gateway-installed nodes and another for all other nodes. You may select the same firewall rule in both places if needed. For information on required ports, see Firewall rules.
SSH Settings	
New SSH public key (Required)	Upload a public key directly from your computer. Note: Cloudera does not use this SSH key. The matching private key can be used by your CDP administrator for root-level access to the instances provisioned for the Data Lake and Cloudera Data Hub.
Add tags	You can optionally add tags to be created for your resources on GCP. Refer to Defining custom tags.

8. Click on Advanced Options to make additional configurations for FreeIPA. The following options are available:

Parameter	Description
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Machine type families.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on FreeIPA nodes. For more information, see Recipes.

9. Click Next.

10. On the Storage page, provide the following information:

Parameter	Description
Logs	
Logger Service Account (Required)	Select the Logger service account created in Minimum setup for cloud storage.
Logs Location Base (Required)	Select the Google Storage location created for logs in Minimum setup for cloud storage.
Backup Location Base	Select the Google Storage location created for FreeIPA backups in Minimum setup for cloud storage. If not provided, the default Backup Location Base uses the Logs Location Base.
Telemetry	
Enable Workload Analytics	Enables Cloudera Observability support for workload clusters created within this environment. When this setting is enabled, diagnostic information about job and query execution is sent to the Cloudera Observability.
Enable Deployment Cluster Logs Collection	When this option is enabled. the logs generated during deployments will be automatically sent to Cloudera.

11. Click Register Environment to trigger environment registration.
12. The environment creation takes about 60 minutes. The creation of the FreeIPA server and Data Lake cluster is triggered. You can monitor the progress from the web UI. Once the environment creation has been completed, its status will change to “Running”.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Register a GCP environment from the CDP CLI

Once you’ve met the Google Cloud cloud provider requirements, register your GCP environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in GCP requirements.

Required role: EnvironmentCreator

Steps

Unlike in the Cloudera web interface, in the CDP CLI environment creation is a two-step process with environment creation and data lake creation being two separate steps. The following commands can be used to create an environment in Cloudera.

1. Once you’ve met the prerequisites, register your GCP environment in Cloudera using the cdp environments create-gcp-environment command and providing the CLI input parameters. For example:

```
cdp environments create-gcp-environment --cli-input-json '{
    "environmentName": "test-env",
    "description": "Test GCP environment",
```

```
    "credentialName": "test-gcp-crd",
    "region": "us-west2",
    "publicKey": "ssh-rsa AAAAB3NzaZ1yc2EAAAADAQABAAABAQDwCI/wmQzbNn9YcA8v
dU+Ot41IIUWJfOfiDrUuNcULOQL6ke5qcEKuboXzbLxV0YmQcPFvswbM5S4FlHjy2VrJ5spy
GhQajFEm9+PgrsybgzHkkssziX0zRq7U4BVD68kSn6CuAHj9L4wx8WBwefMzkw7uO1CkfifI
p8UE6ZcKKKwe2fLR6ErDaN9jQxIWhTPEiFjIhItPHrnOcfGKY/p6OlpDDUOuMRiFZh7qMzfg
vWI+UdN/qjnTlc/M53JftK6GJqK6osN+j7fCwKEnPwWC/gmy8El7ZMHlIENxDut6X0qj9Okc/
JMmG0ebkSZAEbhgNOBNLZYdP0oeQGCXjqdv",
    "enableTunnel": true,
    "usePublicIp": true,
    "existingNetworkParams": {
        "networkName": "eng-private",
        "subnetNames": [
            "private-us-west2"
        ],
        "sharedProjectId": "dev-project"
    },
    "logStorage": {
        "storageLocationBase": "gs://logs",
        "serviceAccountEmail": "logger@dev-project.iam.gserviceaccount.com
"
    }
}'
```

Parameter	Description
environmentName	Provide a name for your environment.
credentialName	Provide the name of the credential created earlier.
region	Specify the region where your existing VPC network is located. For example ”us-west2” is a valid region.
publicKey	Paste your SSH public key.
existingNetworkParams	Provide a JSON specifying the following: `{ "networkName": "string", "subnetNames": ["string", ...], "sharedProjectId": "string" }` Replace the values with the actual VPC network name, one or more subnet names and shared project ID. The sharedProjectId value needs to be set in the following way: • For a shared VPC, set it to the GCP host project ID • For a non-shared VPC, set it to the GCP project ID of the project where Cloudera is being deployed.
enableTunnel	By default Cluster Connectivity Manager is enabled (set to “true”). If you would like to disable it, set it to “false”. If you disable it, then you must also add the following to your JSON definition to specify two security groups as follows: `"securityAccess": { "securityGroupIdForKnox": "string", "defaultSecurityGroupId": "string" }`

Parameter	Description
usePublicIp	Set this to “true” or “false”, depending on whether or not you want to create public IPs.
logStorage	Provide a JSON specifying your configuration for cluster and audit logs: `{ "storageLocationBase": "string", "serviceAccountEmail": "string" }` The storageLocationBase should be in the following format: gs://my-bucket-name.

Note: CDP CLI includes the cdp environments create-gcp-environment --generate-cli-skeleton command option, which allows you to generate a CLI JSON template. You can also use CLI help to get some information about specific CLI command options.

2. To verify that your environment is running, use:

```
cdp environments list-environments
```

You can also log in to the Cloudera web interface to check the deployment status.

3. Once your environment and Data Lake are running, you should set IDBroker Mappings. To create the mappings, run the cdp environments set-id-broker-mappings command. For example:

```
cdp environments set-id-broker-mappings \
 --environment-name test-env \
 --data-access-role dl-admin@dev-project.iam.gserviceaccount.com \
 --ranger-audit-role ranger-audit@dev-project.iam.gserviceaccount.com \
 --mappings '[{"accessorCrn": "crn:altus:iam:us-west-1:45ca3068-42a6-422
7-8394-13a4493e2ac0:user:430c534d-8a19-4d9e-963d-8af377d16963", "role":
"data-science@dev-project.iam.gserviceaccount.com"},{"accessorCrn":"crn:
altus:iam:us-west-1:45ca3068-42a6-4227-8394-13a4493e2ac0:machineUser:mfox-
gcp-idbmms-test-mu/2cbca867-647b-44b9-8e41-47a01dea6c19","role":"data-eng@
dev-project.iam.gserviceaccount.com"}]'
```

Parameter	Description
environment-name	Specify a name of the environment created earlier.
data-access-role	Specify an email address of the Data Lake admin service account created earlier.
ranger-audit-role	Specify an email address of the Ranger audit service account created earlier.
mappings	Map Cloudera users or groups to GCP service accounts created earlier. Use the following syntax: `[ { "accessorCrn": "string", "role": "string" } ... ]` You can obtain user or group CRN from the Management Console > User Management by navigating to details of a specific user or group. The role should be specified as service account email.

4. Next, sync IDBroker mappings:

```
cdp environments sync-id-broker-mappings --environment-name demo3
```

5. Finally, check the sync status:

```
cdp environments get-id-broker-mappings-sync-status --environment-name d
emo3
```

6. One your environment is running, you can create a Data Lake using the cdp datalake create-gcp-datalake command and providing the CLI input parameters:

```
cdp datalake create-gcp-datalake --cli-input-json '{
    "datalakeName": "my-dl",
    "environmentName": "test-env",
    "scale": "LIGHT_DUTY",
    "cloudProviderConfiguration": {
        "serviceAccountEmail": "idbroker@dev-project.iam.gserviceaccount
.com",
        "storageLocation": "gs://data-storage"
    }
}'
```

Parameter	Description
datalakeName	Provide a name for your Data Lake.
environmentName	Provide a name of the environment created earlier.
scale	Provide Data Lake scale. It must be one of: • LIGHT_DUTY or • MEDIUM_DUTY_HA.
cloudProviderConfiguration	Provide the name of the data storage bucket and the email of the IDBroker service account.

Note: CDP CLI includes the cdp datalake create-gcp-datalake --generate-cli-skeleton command option, which allows you to generate a CLI JSON template. You can also use CLI help to get some information about specific CLI command options.

7. To verify that your Data Lake is running, use:

```
cdp datalake list-datalakes
```

You can also log in to the Cloudera web interface to check the deployment status.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Enabling admin and user access to environments

In order to grant admin and user access to an environment that you registered in Cloudera, you should assign the required roles.

You need to be an EnvironmentCreator in order to register an environment. Once an environment is running, the following roles can be assigned:

- EnvironmentAdmin - Grants all rights to the environment and Cloudera Data Hub clusters running in it, except the ability to delete the environment. The user who registers the environment automatically becomes its EnvironmentAdmin.
- EnvironmentUser - Grants permission to view Cloudera Data Hub clusters and set the workload password for the environment. This role should be used in conjunction with service-specific roles such as DataHubAdmin, DWAdmin, DWUser, MLAdmin, MLUser, and so on. When assigning one of these service-specific roles to users, make sure to also assign the EnvironmentUser role.
- DataSteward - Grants permission to perform user/group management functions in Ranger and Atlas Admin, manage ID Broker mappings, and start user sync for the environment.
- Owner - Grants the ability to mange the environment in Cloudera, including deleting the environment. The user who registers the environment automatically becomes its Owner. The Owner role does not grant access the environment's clusters (Data Lakes, Cloudera Data Hub clusters).

The roles are described in detail in Resource roles. The steps for assigning the roles are described in Assigning resource roles to users and Assigning resource roles to groups.

Related Information

Resource roles

Assigning resource roles to users

Assigning resource roles to groups

Understanding environment UI options

To access information related to your environment, navigate to the Cloudera Management Console service > Environments and click on your environment.

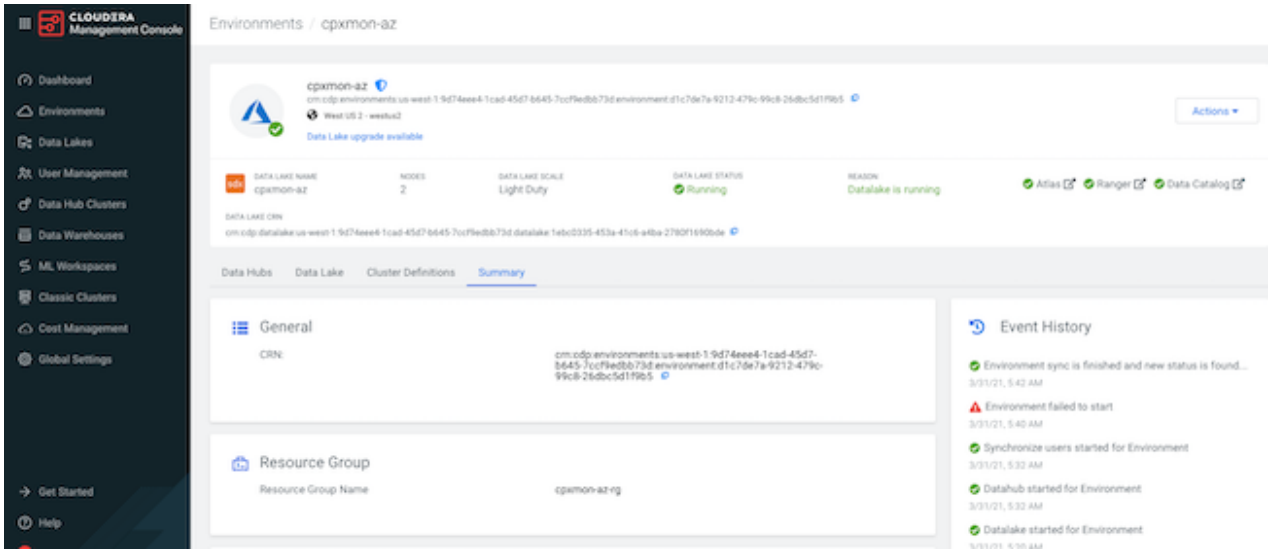

You need to have the EnvironmentUser role or higher for the environment in order to access details of that environment.

From environments details, you can access the following:

- From the Cloudera Management Console tab, you can create, manage, and access Cloudera Data Hub clusters within the environment.
- From the Data Lake tab, you can monitor, manage, and access the Data Lake cluster.
- From the Cluster Definitions tab, you can access all cluster definitions that can be used with the environment.
- From the Summary tab, you can manage and monitor your environment.

The Summary includes the following information:

Option	Description
General	This includes your environment's CRN. CRN is an ID that Cloudera uses to identify a resource.
Credential	This links the provisioning credential associated with the environment and includes the option to change the credential.
Region	This lists the region in which your environment is deployed.
Network	This lists the networking resources used by your environment, provided by you or created by Cloudera during environment registration. You can add additional subnets for Cloudera Data Hub clusters deployed in the future.
Security Access	This lists the firewall rules used by your environment, provided by you or created by Cloudera during environment registration. You can provide new firewall rules for Data Hub clusters deployed in the future.
FreeIPA	This includes details of a FreeIPA server running in the environment and includes an Actions menu with FreeIPA management options.
Log Storage and Audits	This lists the cloud storage location used for logs and audits that you provided during environment registration. There is no way to update this location once your environment is running.
Telemetry	This includes your environment's telemetry settings. You can change them for any Cloudera Data Hub clusters created in the future.
Advanced	This lists the name of your root SSH key.

Related Information

Understanding Data Hub details

Understanding Data Lake details

Monitoring an environment

Once an environment exists, you can access it from the Cloudera Management Console.

Required role: EnvironmentUser, EnvironmentAdmin, or Owner

Steps

For CDP UI

1. To access an existing environment, navigate to Cloudera Management Console > Environments and click on your environment.
2. Click on the Summary tab to access environment details.
3. You can monitor the status of your environment from this page.

For CDP CLI

You can also list available environments from CDP CLI using the cdp environments list-environments command. For example:

```
cdp environments list-environments
{
    "environments": [
        {
            "environmentName": "cdp-demo",
            "crn": "crn:altus:environments:us-west-1:c8dbde4b-ccce-4f8d-
a581-830970ba4908:environment:d3361b40-39ab-4d87-bd5b-abc15f16b90c",
            "status": "DELETE_FAILED",
            "region": "us-east-2",
            "cloudPlatform": "AWS",
            "credentialName": "cdp-demo",
            "description": "Cdp demo"
        },
        {
            "environmentName": "cdp-new",
            "crn": "crn:altus:environments:us-west-1:c8dbde4b-ccce-4f8d-a5
81-830970ba4908:environment:1d2bacde-5c96-47c1-a597-9f276b824028",
            "status": "AVAILABLE",
            "region": "us-east-2",
            "cloudPlatform": "AWS",
            "credentialName": "cdp-demo",
            "description": ""
        }
}
```

To get more information about a specific environment, you can use the following commands:

```
cdp environments describe-environment --environment-name <value>
```

```
cdp environments get-id-broker-mappings --environment-name <value>
```

Environment status options

This topic lists all possible environment status options for the UI and CLI and explains what they mean.

Environment status	Description
Environment creation	
CREATION_INITIATED	Environment creation request was registered in the database and Cloudera is starting the environment creation flow.
ENVIRONMENT_INITIALIZATION_IN_PROGRESS	Setting up the region and network metadata (public/private and cidr).
ENVIRONMENT_VALIDATION_IN_PROGRESS	Setting up the region and network metadata (public/private and cidr).
NETWORK_CREATION_IN_PROGRESS	If the user chose the create new network option, then Cloudera creates the network on cloud provider side.
PUBLICKEY_CREATE_IN_PROGRESS	If the user choose the create new SSH key option, then Cloudera creates the SSH key on cloud provider side.
FREEIPA_CREATION_IN_PROGRESS	Creating the FreeIPA resources for an environment.
Environment update	
UPDATE_INITIATED	Environment update was requested and Cloudera is starting the update flow (network update, load balancer update, SSH key update).
Environment deletion	

Environment status	Description
DELETE_INITIATED	Environment deletion request was registered and Cloudera is starting the deletion flow.
NETWORK_DELETE_IN_PROGRESS	If the user chose the create new network option, then Cloudera deletes the network on cloud provider side.
PUBLICKEY_DELETE_IN_PROGRESS	If the user choosing the create new SSH key option, then Cloudera deletes the SSH key on cloud provider side.
FREEIPA_DELETE_IN_PROGRESS	Deleting the FreeIPA resources for an environment.
EXPERIENCE_DELETE_IN_PROGRESS	Deleting all the attached clusters (Cloudera Data Warehouse, Cloudera AI, and so on).
RDBMS_DELETE_IN_PROGRESS	Deleting all the provisioned RDS instances that are related to an environment.
CLUSTER_DEFINITION_DELETE_PROGRESS	Deleting all the cluster definitions that are created for an environment.
UMS_RESOURCE_DELETE_IN_PROGRESS	Deleting all the related UMS resources for an environment.
IDBROKER_MAPPINGS_DELETE_IN_PROGRESS	Deleting all the IBroker mapping for an environment.
S3GUARD_TABLE_DELETE_IN_PROGRESS	Deleting all the Dynamo DB tables for an environment.
DATAHUB_CLUSTERS_DELETE_IN_PROGRESS	Deleting all the attached Cloudera Data Hub clusters.
DATALAKE_CLUSTERS_DELETE_IN_PROGRESS	Deleting the attached Data Lake cluster.
ARCHIVED	Environment has been deleted (not shown on the UI).
Environment is running	
AVAILABLE	Environment is available (ready to use).
Environment process failed	
CREATE_FAILED	Environment creation failed (Detailed message in the statusReason).
DELETE_FAILED	Environment deletion failed (Detailed message in the statusReason).
UPDATE_FAILED	Environment update failed (Detailed message in the statusReason).
Environment stop	
STOP_DATAHUB_STARTED	Stopping all the Cloudera Data Hub clusters in an environment.
STOP_DATAHUB_FAILED	Stopping all the Cloudera Data Hub clusters in an environment failed (Detailed message in the statusReason).
STOP_DATALAKE_STARTED	Stopping the Data Lake cluster in an environment.
STOP_DATALAKE_FAILED	Stopping the Data Lake cluster in an environment failed (Detailed message in the statusReason).
STOP_FREEIPA_STARTED	Stopping the FreeIPA instances in an environment.
STOP_FREEIPA_FAILED	Stopping the FreeIPA instances in an environment failed (Detailed message in the statusReason).
ENV_STOPPED	Environment was successfully stopped.
Environment start	
START_DATAHUB_STARTED	Starting all the Cloudera Data Hub clusters in an environment.
START_DATAHUB_FAILED	Starting all the Cloudera Data Hub clusters in an environment failed (Detailed message in the statusReason).
START_DATALAKE_STARTED	Starting the Data Lake cluster in an environment.
START_DATALAKE_FAILED	Starting the Data Lake cluster in an environment failed (Detailed message in the statusReason).
START_FREEIPA_STARTED	Starting all the FreeIPA instances in an environment.

Environment status	Description
START_FREEIPA_FAILED	Starting all the FreeIPA instances failed in an environment (Detailed message in the statusReason).
START_SYNCHRONIZE_USERS_STARTED	Starting user sync for all the clusters in an environment.
START_SYNCHRONIZE_USERS_FAILED	Starting user sync for all the clusters in an environment failed (Detailed message in the statusReason).
FreeIPA instance deletion	
FREEIPA_DELETED_ON_PROVIDER_SIDE	The FreeIPA instance has been deleted on cloud provider side.
Load balancer	
LOAD_BALANCER_ENV_UPDATE_STARTED	Start updating the LoadBalancer on Data Lake in an environment.
LOAD_BALANCER_ENV_UPDATE_FAILED	Failed to update the LoadBalancer on Data Lake in an environment (Detailed message in the statusReason).
LOAD_BALANCER_STACK_UPDATE_STARTED	Start updating the LoadBalancer on Cloudera Data Hub clusters in an environment.
LOAD_BALANCER_STACK_UPDATE_FAILED	Failed to update the LoadBalancer on Cloudera Data Hub clusters in an environment (Detailed message in the statusReason).

Stop and restart an environment

You can stop an environment if you need to suspend but not terminate the resources within the environment. When you stop an environment, all of the resources within the environment are also stopped, including Data Lakes and Cloudera Data Hub clusters. You can also restart the environment.

Warning:

The Cloudera AI service does not support environment stop and restart. This means that if Cloudera AI workbenches are running or expected to be provisioned within an environment, then the environment should not be stopped. If done, this will disrupt running Cloudera AI workbenches and prevent successful provisioning of Cloudera AI workbenches in the environment.

Required role: EnvironmentAdmin or Owner

Steps

For CDP UI

1. Navigate to the environment in Cloudera Management Console > Environments.
2. Click Actions Stop Environment and confirm the action.
3. To restart the environment, click Actions Start Environment .

Warning: We have not tested or certified restarting the environment while Cloudera Data Engineering is running.

For CDP CLI

Use the following command to stop an environment:

```
cdp environments stop-environment --environment-name <ENVIRONMENT_NAME>
```

Use the following commands to start an environment:

```
cdp environments start-environment --environment-name <ENVIRONMENT_NAME>
 [--with-datahub-start]
```

Use the following commands to start an environment and all Cloudera Data Hub clusters running in it:

```
cdp environments start-environment --environment-name <ENVIRONMENT_NAME>
 --with-datahub-start
```

Delete an environment

Deleting an environment terminates all resources within the environment including the Data Lake.

Before you begin

To delete an environment, you should first terminate all clusters running in that environment.

Required role: Owner or PowerUser

Steps

For CDP UI

1. In Cloudera Management Console, navigate to Environments.
2. Click on your environment.
3. Click Actions Delete .
4. Check the box next to "I would like to delete all connected resources" if you have Data Lake and Cloudera Data Hub clusters running within the environment. This will delete the Data Lake and Cloudera Data Hub clusters together with the rest of the environment.

   Note: The "I would like to delete all connected resources" option does not delete any Cloudera Data Warehouse or Cloudera AI clusters running within the environment, so these always need to be terminated prior to environment termination.
5. Check the box next to "I understand this action cannot be undone". This is required.
6. Click Delete.

For CDP CLI

When terminating an environment from the CDP CLI, you need to first terminate the Data Lake:

1. Terminate the Data Lake using the following command:

   ```
   cdp datalake delete-datalake --datalake-name <value>
   ```
2. Wait until the Data Lake terminates before proceeding. Use the following commands to check on the status of Data Lake:

   ```
   cdp datalake get-cluster-host-status --cluster-name <value>
   ```

   ```
   cdp datalake list-datalakes
   ```
3. Delete the environment using the following command:

   ```
   cdp environments delete-environment --environment-name <value> --cascadi
   ng
   ```

   The --cascading option deletes or Cloudera Data Hub clusters running in the environment.

If environment deletion fails, you can:

- Repeat the environment deletion steps, but also check "I would like to force delete the selected environment". Force deletion removes Cloudera resources from Cloudera, but leaves cloud provider resources running.

- Clean up cloud resources that were left on your cloud provider account.

Only the resources that were provisioned as part of the environment are deleted. For example, if a new network was created by Cloudera for the environment, the network will be deleted; But if you provided your existing network, it will not be deleted as part of environment deletion.

Change environment's credential

You can change the credential attached to an environment as long as the new credential provides the required level of access to the same GCP account as the old credential.

Required roles:

- EnvironmentAdmin or Owner of the environment
- SharedResourceUser or Owner of the credential

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Environments from the navigation pane.
4. Click on a specific environment.
5. Navigate to the Summary tab.
6. Scroll down to the Credential section.
7. Click to access credential edit options.
8. Select the new credential that you would like to use.
9. Click to save the changes.

For CDP CLI

If you would like to delete a credential from the CDP CLI, use:

```
cdp environments update-environment-credential --environment-name <value>
 --credential-name <value>
```

Defining custom tags

In the Cloudera Management Console clusters user interface, you can define tenant-level or environment-level custom tags across all instances and resources provisioned in your organization’s cloud provider account.

Resource tagging

When you create an environment or other resources shared across your cloud provider account, Cloudera clusters automatically adds default tags to the Cloudera-created resources in your cloud provider account. You can also define additional custom tags that Cloudera applies to the cluster-related resources in your account.

You can use tags to protect the cloud resources of your Cloudera environment. Using the tags, you can exclude the resources that should not be removed during housekeeping or security deletion activities that can result in data corruption and data loss.

Default tags

By default, Cloudera applies certain tags to cloud provider resources whenever you create the resource, for example an environment.

Cloudera applies the following tags by default:

- Cloudera-Resource-Name: the workload-appropriate Cloudera resource name. For example, an IPA CRN for an IPA, a data lake CRN for a data lake, or a Cloudera Data Hub CRN for a Cloudera Data Hub cluster. This CRN serves as a unique identifier for the resource over time.
- Cloudera-Creator-Resource-Name: Cloudera resource name of the Cloudera user that created the resource.
- Cloudera-Environment-Resource-Name: name of the environment with which the resource is associated.

Custom tags

There are two types of custom tags that you can define in the Cloudera Management Console: tenant-level tags that apply to Cloudera-created resources across your organization's entire cloud provider account, and environment-level tags.

In the Cloudera Management Console user interface, you can define tenant-level tags across all instances and resources provisioned in your organization's cloud provider account. These resources include not only provisioned instances, but disks, networks, and other resources as well. In your cloud provider account you can search or filter on either the tag key or value. Tenant-level tags cannot be overridden during environment creation.

In addition to tenant-level tags, you can also define environment-level tags. Environment-level tags are inherited by the resources specific to an environment. For example, environment-level tags are inherited by the following resources:

- FreeIPA
- Data Lakes
- Cloudera Data Hub clusters
- Cloudera Operational Databases

As with tenant-level tags, you can search or filter on the key tag or key value in your cloud provider account.

Note: Cloudera applies custom tags during creation of the resources. For example, you can only define environment-level tags during environment registration. If you want to add or update cloud provider resource tags, you must do so through the cloud provider API.

For more information about using tags on cloud provider resources, consult AWS, Azure, or Google Cloud documentation. It is your responsibility to ensure that your tags meet your cloud provider requirements.

Supported services

While some Cloudera services such as Cloudera Data Hub inherit environment-level tags, others require that you add tags when provisioning or enabling the data service. The following table shows how tags can be added for various Cloudera services:

Cloudera service	How to add tags
Data Lake	Inherits tenant or environment level tags.
FreeIPA	Inherits tenant or environment level tags.
Cloudera Data Hub	Inherits tenant or environment level tags and you can add more tags when creating a Cloudera Data Hub.
Cloudera Operational Database	Inherits tenant or environment level tags and you can add more tags when creating a Cloudera Operational Database database via CLI.

Defining tenant-level tags

Required roles: PowerUser can define tags for the whole tenant.

- EnvironmentAdmin or Owner can set environment telemetry settings for a specific environment.

Steps

1. In the Cloudera Management Console, click Global Settings Tags .
2. Click Add.
3. Define both a key and a value for the tag. Both the key and the value must be between 4- 255 characters, with the following restrictions:

   Key

   Allowed characters are hyphens (-), underscores (_), lowercase letters, and numbers. Keys must start with a lowercase letter and must not start or end with spaces.

   Value

   Allowed characters are hyphens (-), underscores (_), lowercase letters, and numbers. Values must not start or end with spaces. You can configure variables in the *{{{VARIABLENAME}}}* format. The following variables are supported for tenant-level tags:

   - *{{{CLOUDPLATFORM}}}* = AWS, AZURE or GCP
   - *{{{USERNAME}}}* = Cloudera username
   - *{{{USERCRN}}}* = Customer Resource Number (CRN) of Cloudera user
   - *{{{CREATORCRN}}}* = CRN of Cloudera resource creator
   - *{{{TIME}}}* = Actual time
   - *{{{ACCOUNTID}}}* = Cloudera account ID
   - *{{{RESOURCECRN}}}* = Generated string of Cloudera resource CRN
4. Click Add, and if necessary repeat the process for additional tags.

Note: Tenant-level tags are applied only to resources created after you define the tag. Any changes to the tag value do not propagate to existing resources that have the tag.

Defining environment-level tags

You define environment-level tags during environment registration.

Required roles: EnvironmentCreator can set tags for a specific environment during environment registration.

Steps

1. In the Cloudera Management Console, click Environments Register Environment .
2. Proceed through the environment registration steps.
3. After you define data access, add any environment-level tags by clicking Add and defining the tag key and value.

Related Information

Label format

Adding a customer managed encryption key for GCP

By default, a Google-managed encryption key is used to encrypt disks and Cloud SQL instances in Data Lake, FreeIPA, and Cloudera Data Hub clusters, but you can optionally configure Cloudera to use a customer-managed encryption key (CMEK) instead.

To set up a CMEK, you should:

1. Meet the CMEK prerequisites.
2. Register a GCP environment in Cloudera via Cloudera web UI or CLI. During environment registration, specify the encryption key that you would like to use.

CMEK prerequisites

Refer to GCP Prerequisites: Customer managed encryption keys.

Create a Cloudera environment with a CMEK

You can pass the CMEK during GCP environment registration in Cloudera via Cloudera web interface or CDP CLI.

Steps

For CDP UI

You can register your environment as described in Register a GCP environment from CDP UI, just make sure that on the Data Access and Audit page you enable the following:

1. Under Customer-Managed Encryprion Keys, click Enable Customer-Managed Keys.
2. In the same section, select the CMEK:

The following screenshot shows the UI options:

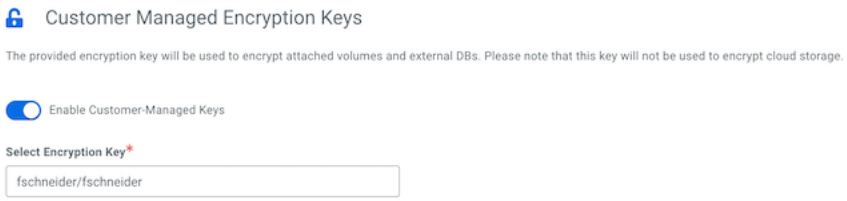

For CDP CLI

The steps below can only be performed via CDP CLI. Create an environment passing the --encryption-key parameter as shows in this example:

```
cdp environments create-gcp-environment \
  --no-use-public-ip \
  --environment-name <ENVIRONMENT_NAME> \
  --credential-name <EXISTING_CREDENTIAL-NAME>\
  --region <REGION>\
  --security-access securityGroupIdForKnox=<SG_NAME1>,defaultSecurityGro
upId=<SG_NAME2> \
  --public-key <PUBLIC_SSH_KEY>\
  --log-storage storageLocationBase=<LOGS_STORAGE_LOCATION> \
  --existing-network-params networkName=<NETWORK>,subnetNames=<SUBNET>,
sharedProjectId=<PROJECT_ID>\
  --workload-analytics \
  --encryption-key <PATH_TO_THE_ENCRYPTION_KEY>
```

Note: If the --encryption-key parameter is not provided the GCP resources are not encrypted using CMEK, falling back to the default behavior of Google managed encryption.

Next, create a Data Lake and IDBroker mappings using the usual commands. Once the environment is running, Data Hubs can be created using the usual steps.

Cloudera Management Console

Fine-grained access control for Amazon S3

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to RAZ on AWS environments..... 4

AWS requirements for RAZ-enabled AWS environment.....5

Registering a RAZ-enabled AWS environment..... 5
Using the Cloudera UI to register RAZ-enabled AWS environment.....5
Using the CDP CLI to register RAZ-enabled AWS environment.....6

Cluster templates available in RAZ-enabled AWS environment.....7
Creating a Cloudera Data Hub cluster with custom path.....7

Provisioning Cloudera AI workbench for RAZ-enabled AWS environment.....11

Ranger policy options for RAZ-enabled AWS environment.....11
Creating Ranger policy to use in RAZ-enabled AWS environment.....13

Troubleshooting for RAZ-enabled AWS environment.....14
Why does the "AccessDeniedException" error appear? How do I resolve it?..... 14
Why does the "Permission denied" error appear? How do I resolve it?..... 15
Why does the "S3 access in hive/spark/mapreduce fails despite having an "allow" policy" error message appear? How do I resolve it?..... 15
An error related to Apache Flink appears after the job fails. How do I resolve this issue?.....16
What do I do to display the Thread ID in logs for RAZ?.....17
What do I do when a long-running job consistently fails with the expired token error?.....17

Managing a RAZ-enabled AWS environment.....18

Introduction to RAZ on AWS environments

Cloudera Public Cloud defaults to using cloud storage which might be challenging while managing data access across teams and individual users. The Ranger Authorization Service (RAZ) resolves this challenge by enabling Amazon S3 users to use fine-grained access policies and audit capabilities available in Apache Ranger similar to those used with HDFS files in an on-premises or IaaS deployment.

The core RAZ for AWS for Data Lakes and several Cloudera Data Hub templates are available for production use platform-wide.

Many of the use cases that RAZ for S3 enables are cases where access control on files or directories is needed. Some examples include:

- Per-user home directories.
- Cloudera Data Engineering (Spark) efforts that require access to cloud storage objects and directories.
- Cloudera Data Warehouse queries (Hive/Impala) that use external tables.
- Access to Ranger's rich access control policies such as date-based access revocation, user/group/role-based controls, along with corresponding audit.
- Tag-based access control using the classification propagation feature that originates from directories.

Prior to the introduction of RAZ, controlling access to S3 could be enforced at coarse-grained group level using IDBroker mappings. This required rearchitecting the implementation of important file-centric activities as well as admin-level access to both the AWS account and the Cloudera account.

Important: It is recommended that you do not setup IDBroker mapping for workload users in a RAZ-enabled AWS environment.

In HDP and CDH deployments, files and directories are protected with a combination of HDFS Access Control Lists (ACLs) (in CDH, HDP) and Ranger HDFS policies (in HDP). Similarly, in an AWS Cloudera Public Cloud environment with RAZ for S3 enabled, Ranger's rich access control policies can be applied to Cloudera's access to S3 buckets, directories, and files and can be controlled with admin-level access to Cloudera alone.

You can backup and restore the metadata maintained in the Data Lake services of RAZ-enabled environments. For more information, see Data Lake Backup and Restore.

Limitations to use RAZ in AWS environments

The following limitations and known issues have been identified and are under development:

- Currently, there is no automated way to enable RAZ in an existing Cloudera environment that does not have RAZ enabled.

Architecture diagram

The following architecture diagram shows how RAZ integrates and interacts with other components in a RAZ-enabled AWS environment:

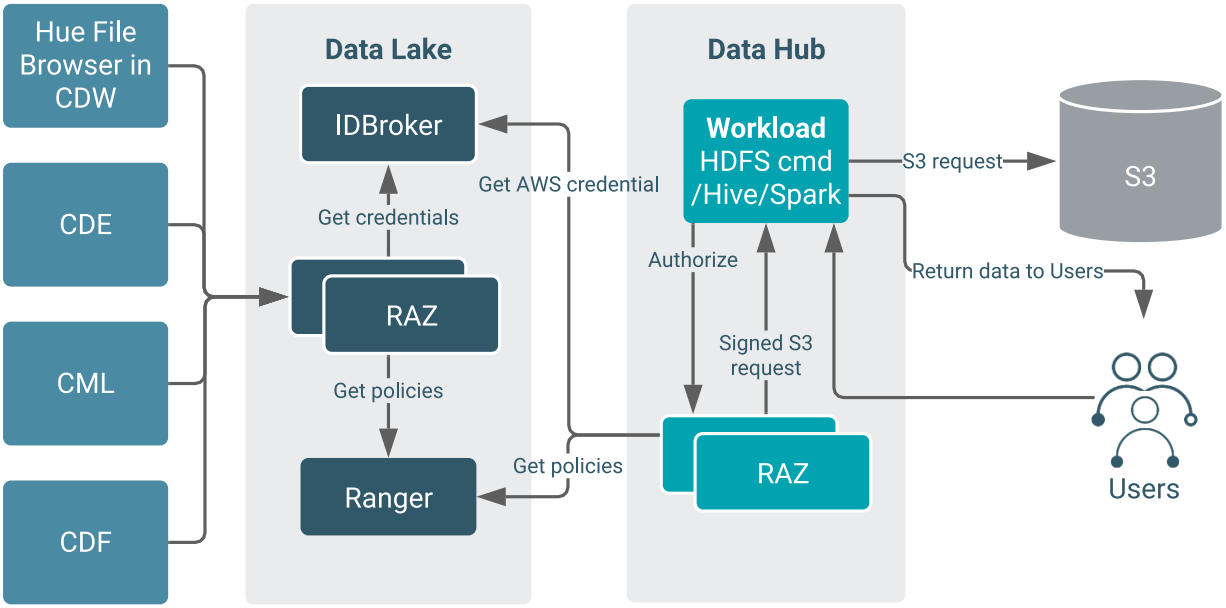

AWS requirements for RAZ-enabled AWS environment

Before you enable RAZ on an AWS environment, you must ensure that Cloudera has access to the resources in your AWS account and that your AWS account has all the necessary resources required by Cloudera.

For information about the requirements that must be met to use AWS environment, see AWS requirements documentation.

There are no additional requirements for a RAZ-enabled AWS environment, but RAZ can use the DATALAKE_ADMIN_ROLE IAM role. For more information about the IAM role, see Minimal setup for cloud storage.

Registering a RAZ-enabled AWS environment

You can use the Cloudera web interface or CDP CLI to register a RAZ-enabled AWS environment.

You can enable RAZ on the latest available version of Cloudera Runtime. The minimum version supporting RAZ for AWS environments is Cloudera Runtime 7.2.11.

When you register an AWS environment, enable the Fine-grained access control on S3 option, and then select the DATALAKE_ADMIN_ROLE role to use RAZ in your AWS environment. For more information about the role, see Minimal setup for cloud storage.

Using the Cloudera UI to register RAZ-enabled AWS environment

You can use the Cloudera web interface to register a RAZ-enabled AWS environment.

Procedure

1. Log in to the Cloudera web interface.
2. Click Register environment on the Cloudera Management Console Environments page.

3. Provide an Environment Name.
4. Select a provisioning credential.
5. Click Next.
6. Provide a Data Lake Name.
7. Make sure to select Runtime 7.2.11 or higher as the Data Lake version.
8. In the Data Access and Audit section, provide your data storage location and IAM roles created for minimal setup for cloud storage.
9. In the Fine-grained access control on S3 section, click on the toggle button to enable Ranger authorization for S3. Select DATALAKE_ADMIN_ROLE or RAZ_ROLE (if created) as the AWS IAM role.

   The following image shows the Fine-grained access control on S3 section where you can enable the Ranger authorization for AWS S3 and choose an AWS IAM role for Ranger authorizer:

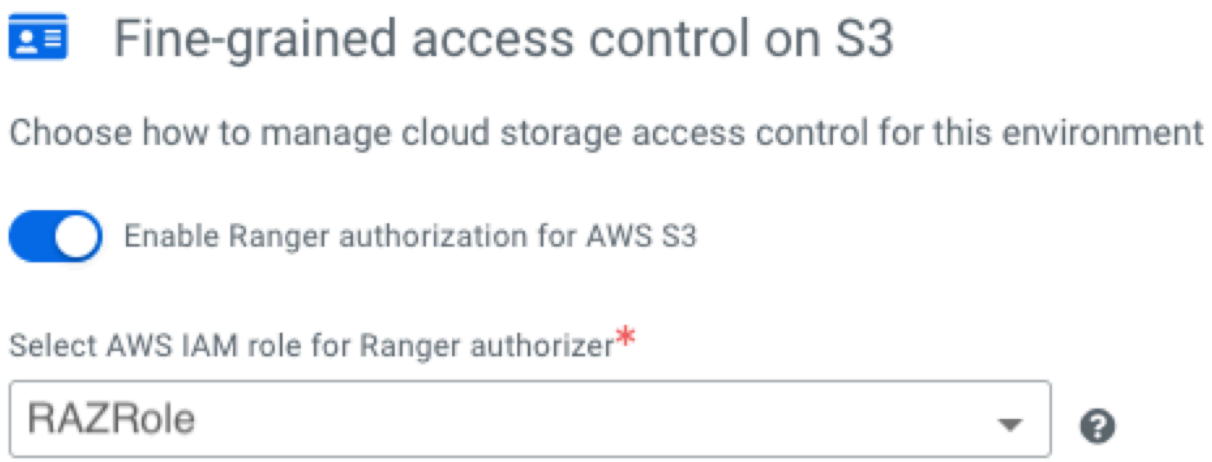

10. Click Next.
11. Select your region, network, security groups, and provide an SSH key. If required, add tags.
12. Click Next.
13. In the Logs section, provide your logs storage location and managed identities created for minimal setup for cloud storage.
14. Click Register Environment.

Using the CDP CLI to register RAZ-enabled AWS environment

You can use the CDP CLI to register a RAZ-enabled AWS environment. You must download and install beta CDP CLI, and then use CDP CLI commands to register a RAZ-enabled AWS environment.

Procedure

1. To install beta CDP CLI, see Installing Beta CDP CLI.
2. To register a RAZ-enabled AWS environment, you can use the --ranger-cloud-access-authorizer-role *[***RAZ_ROLE***]* and --enable-ranger-raz CDP CLI commands.

   If you have CDP CLI templates to create an AWS environment, modify them by adding the additional parameters required for RAZ.

   The additional option is highlighted in the following sample snippet:

```
cdp environments create-aws-environment \
    --environment-name [***ENVIRONMENT_NAME***] \
    --credential-name [***CREDENTIAL_NAME***] \
    --region [***REGION***] \
```

```
    --security-access cidr=[***YOUR_CIDR***] \
    --authentication publicKeyId=[***SSH_PUBLIC_KEY***] \
    --log-storage storageLocationBase=[***S3-LOCATION***],instancePro
file=[***LOG_ROLE***] \
    --vpc-id [***VPC***] \
    --subnet-ids [***SUBNETS***]

 cdp environments set-id-broker-mappings \
    --environment-name [***ENVIRONMENT-NAME***] \
    --data-access-role [***DATALAKE-ADMIN-ROLE***] \
    --ranger-audit-role [***RANGER_AUDIT_ROLE***] \
    --ranger-cloud-access-authorizer-role [***RAZ_ROLE***] \
    --set-empty-mappings
cdp datalake create-aws-datalake \
    --datalake-name [***DATALAKE_NAME***] \
    --environment-name [***ENVIRONMENT-NAME***]\
    --cloud-provider-configuration instanceProfile=[***INSTANCE-
PROFILE***],storageBucketLocation=s3a://[***BUCKET***]/[***PATH***] \
   --enable-ranger-raz
```

Note:

RAZ on AWS environment requires S3Guard to be disabled. Do not pass --S3-guard-table-name in the cdp-environments create-aws-environments command.

You can obtain CDP CLI commands for environment creation from CDP CLI help on Cloudera web interface. For more information, see Obtain CLI commands for registering an environment.

Cluster templates available in RAZ-enabled AWS environment

After you register your RAZ-enabled AWS environment, you can create a Cloudera Data Hub cluster for the environment.

The following cluster templates have been tested and can be used with RAZ:

- Data Engineering
- Data Engineering HA
- Data Engineering Spark3
- Data Mart
- Operational Database with SQL

You can create custom variants of these templates in a RAZ-enabled AWS environment.

Warning: If you create a Cloudera Data Hub with custom paths for a RAZ-enabled AWS environment, your cluster might fail with an AccessDeniedException exception on the custom Zeppelin path. To fix the problem, you must use the workaround mentioned in Creating a Cloudera Data Hub cluster with custom path on page 7

Creating a Cloudera Data Hub cluster with custom path

When you create a Cloudera Data Hub cluster with custom paths for Cloudera Data Hub clusters, your cluster might fail with an access exception. To resolve this issue, you must add the policy path to the default Zeppelin notebooks policy using Ranger.

About this task

Cloudera Data Hub cluster creation fails and a Zeppelin exception is logged in the Cloudera Manager logs, if you create a Cloudera Data Hub cluster with a custom path <bucket>/custom-dh, overwriting the YARN location and Zeppelin notebook location to <bucket>/yarn-logs and <bucket>/zeppelin-notebook respectively.

The following image shows the Create Cluster Advanced Options Cloud Storage page where you can optionally specify the base storage location used for YARN and Zeppelin:

Figure 1: Cloud Storage page

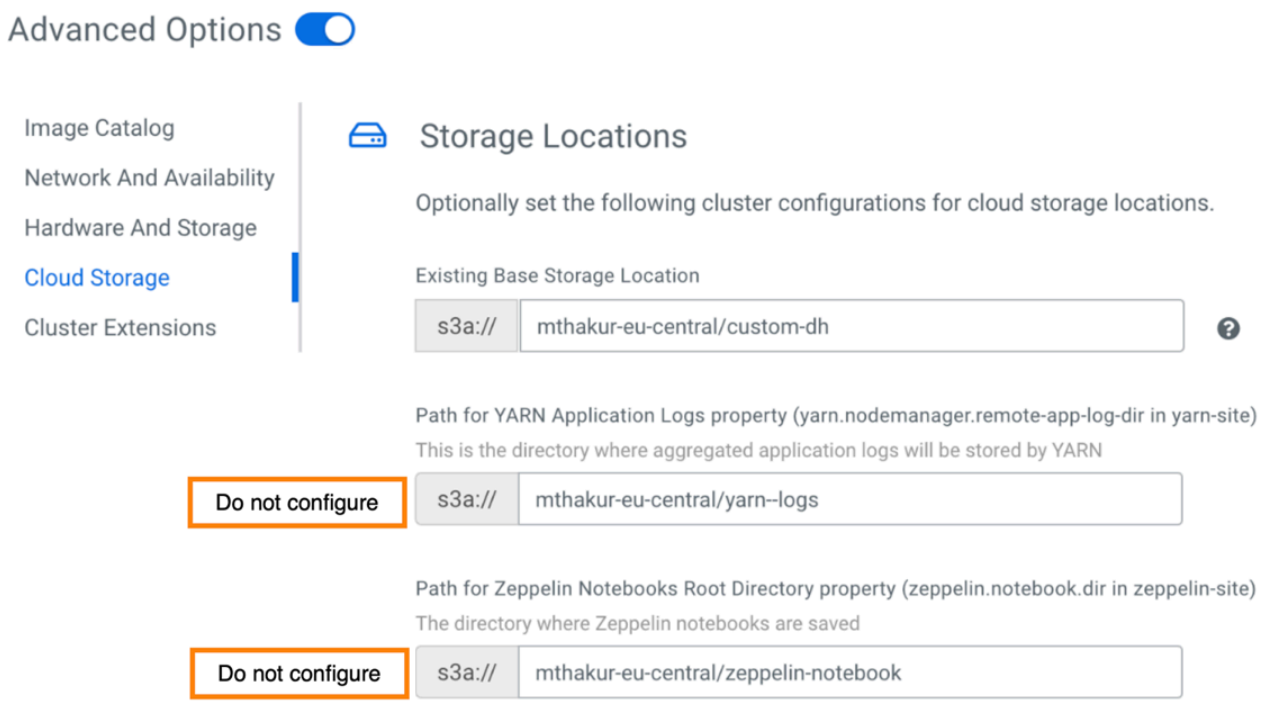

The following image shows the Zeppelin exception in the Cloudera Manager logs:

Figure 2: Zeppelin exception in Cloudera Manager logs

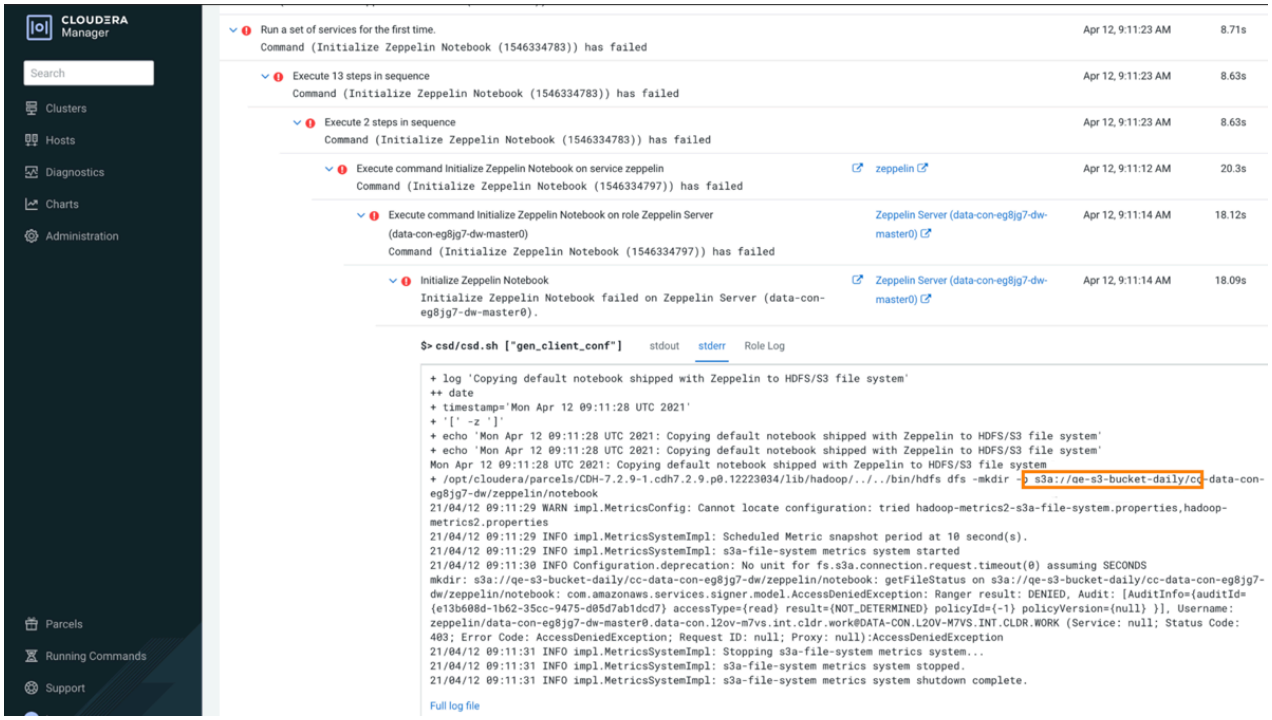

To resolve this issue, perform the following steps:

Procedure

1. Add the policy path to the default Zeppelin notebooks policy using Ranger.

   The following image shows the options to add the policy path to the Zeppelin notebooks policy in Ranger UI:

   Figure 3: Policy path in Ranger UI

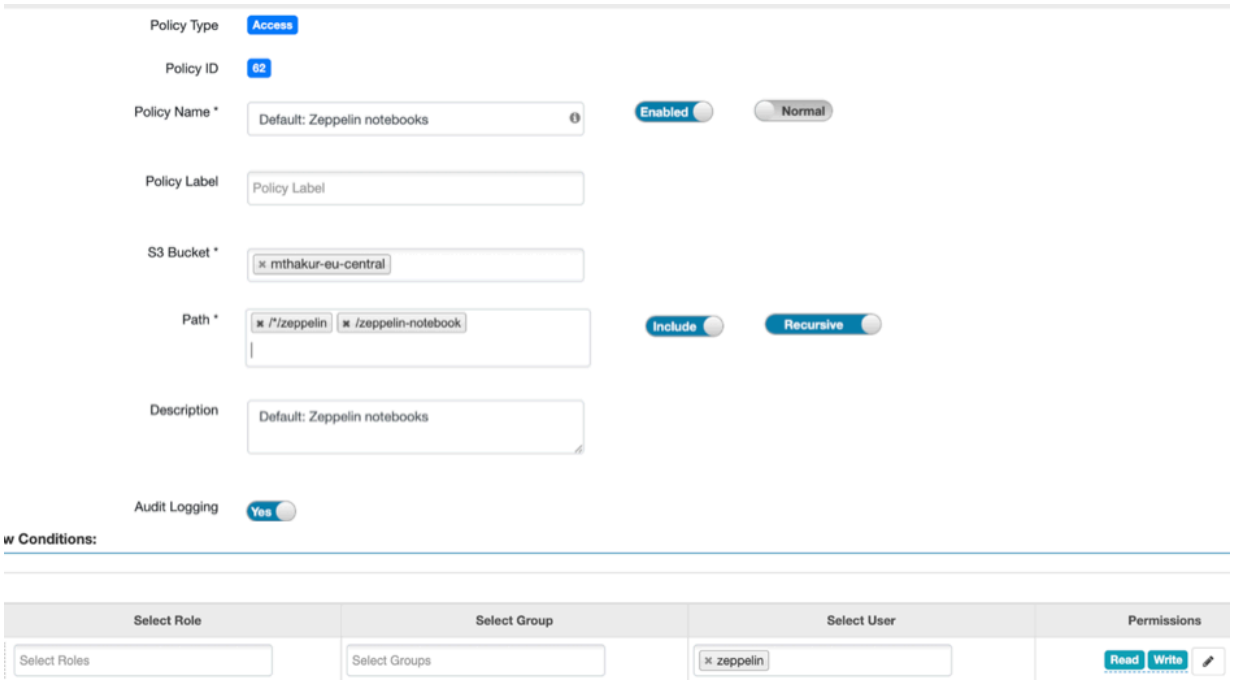

2. For YARN logs aggregation, add the following using Ranger:

   - Add /yarn--logs/{USER} path to the Default: Hadoop App Logs - user default policy.
   - Add /yarn--logs path to the Default: Hadoop App Logs default policy.

   The following sample image shows the update required in the Default: Hadoop App Logs - user default policy:

   Figure 4: Default: Hadoop App Logs - user default policy

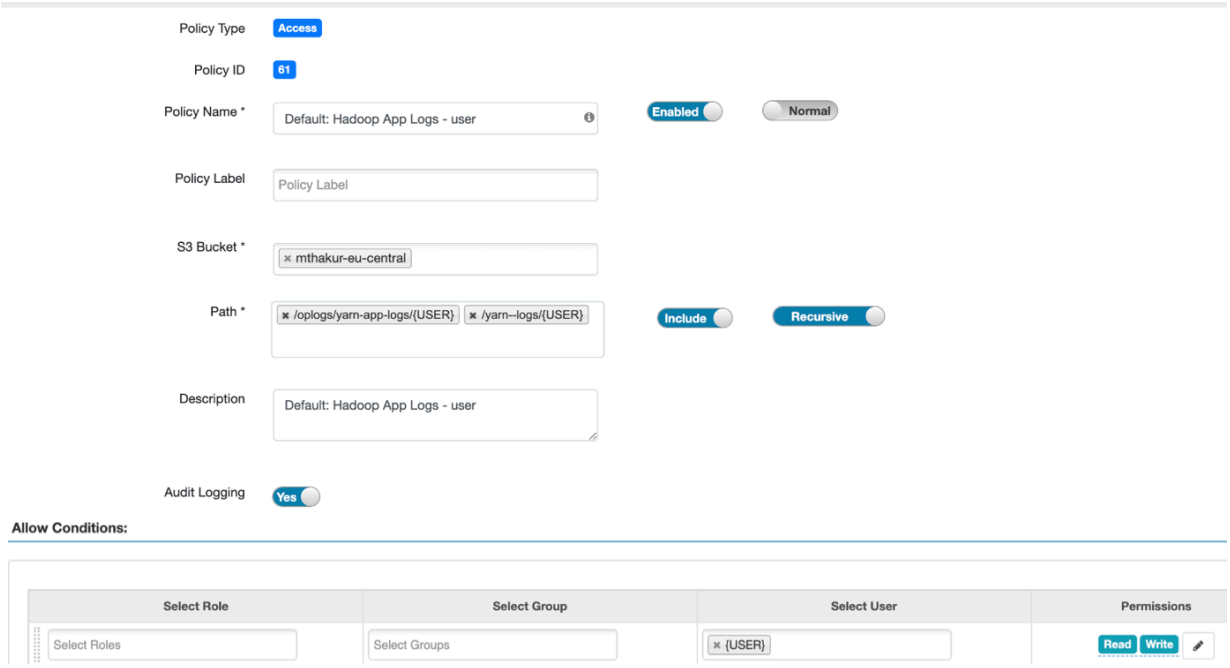

   The following sample image shows the update required in the Default: Hadoop App Logs default policy:

   Figure 5: Default: Hadoop App Logs default policy

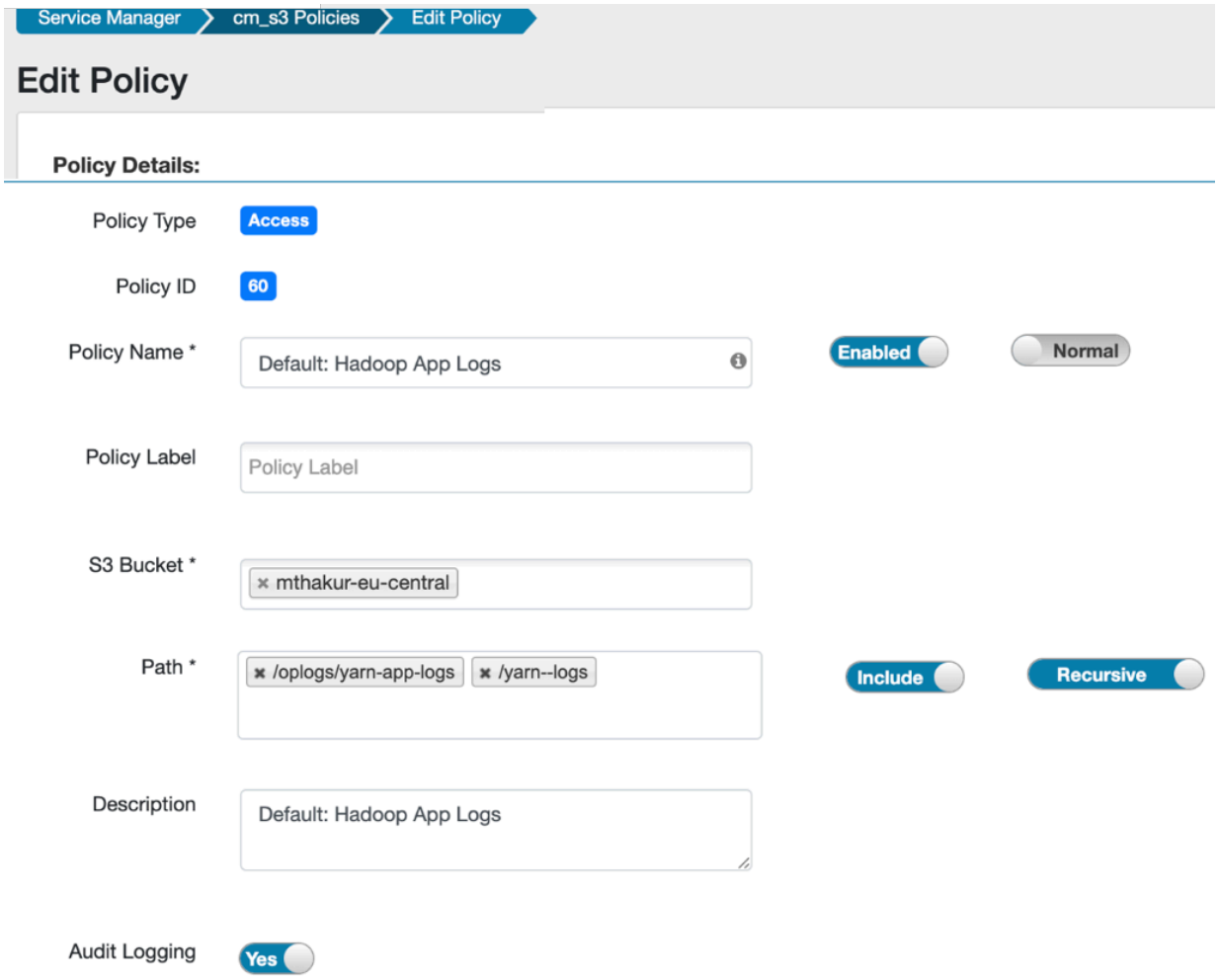

3. After adding the policies, perform the following steps:
   a. In Cloudera Manager, re-run the failed commands.
   b. After the commands run successfully, go to the Cloudera Management Console Data Hub Clusters page, and click Actions Retry Yes option.

      The operation continues from the last failed step.

Provisioning Cloudera AI workbench for RAZ-enabled AWS environment

If you require Cloudera AI, you must provision a Cloudera AI workbench. A Cloudera AI workbench enables teams of data scientists to develop, test, train, and ultimately deploy machine learning models for building predictive applications all on the data under management within the enterprise data cloud.

For information about provisioning a Cloudera AI workbench, see Provisioning Cloudera AI workbenches.

Ranger policy options for RAZ-enabled AWS environment

After you register the RAZ-enabled AWS environment, you can log in to Ranger to create the policies for granular access to the environment's cloud storage location.

Ranger includes a set of preloaded resource-based services and policies. You need the following additional policies for granular access to the environment's cloud storage location:

- Policies for Spark jobs
- Policies for Hive external tables and Spark jobs
- Policies for Hive managed tables and Spark jobs

Policies for Spark jobs

A Spark job on an S3 path requires an S3 policy for the end user on the specific path. For information to create the policies, see Creating Ranger policy to use in RAZ-enabled AWS environment on page 13.

For example, a Spark job on s3a://bucket/data/logs/tabledata requires an S3 policy in cm_s3 repo on s3a://bucket/data/logs/tabledata for end user.

The following sample image shows the S3 policy created in the cm_s3 repo for the user csso_abc to read and write data in s3a://abc-eu-central/abc/test.csv:

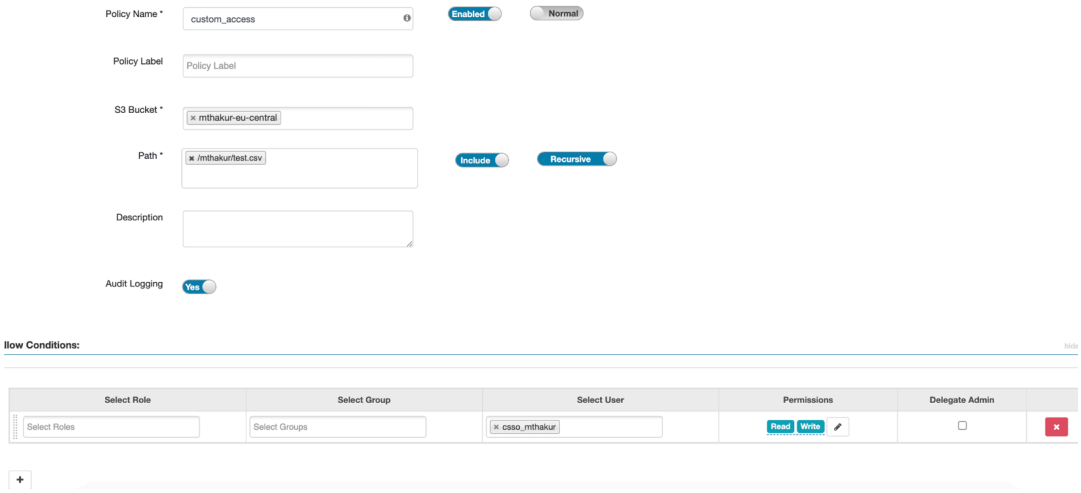

Policies for Hive external tables and Spark jobs

Running the create external table *[***TABLE DEFINITION***]* location 's3a://bucket/data/logs/tabledata' command in Hive requires the following Ranger policies:

- An S3 policy in the cm_s3 repo on s3a://bucket/data/logs/tabledata for hive user to perform recursive read/write.
- An S3 policy in the cm_s3 repo on s3a://bucket/data/logs/tabledata for the end user.
- A Hive URL authorization policy in the Hadoop SQL repo on s3a://bucket/data/logs/tabledata for the end user.

Access to the same external table location using Spark shell requires an S3 policy (Ranger policy) in the cm_s3 repo on s3a://bucket/data/logs/tabledata for the end user.

For information to create the policies, see Creating Ranger policy to use in RAZ-enabled AWS environment on page 13.

Policies for Hive managed tables and Spark jobs

Operations such as create, insert, delete, select, and so on, on a Hive managed table do not require any custom Ranger policies.

For information to create the policies, see Creating Ranger policy to use in RAZ-enabled AWS environment on page 13.

Creating Ranger policy to use in RAZ-enabled AWS environment

After you register the RAZ-enabled AWS environment, you can log in to Ranger to create the policies for granular access to the environment's cloud storage location. To create the Ranger policy, you must first create the required S3 policy and then a Hive URL authorization policy, on an S3 path for the end user.

Procedure

1. To create the required S3 policy on an S3 path for end user, perform the following steps:
   a) Navigate to the Ranger UI.
   b) On the S3 tab, click cm_s3.
   c) Click Add New Policy in the top right corner.
   d) Provide the following policy details:
      1. Enter Policy Name.
      2. Enter an S3 Bucket name.
      3. Provide a Path within the S3 bucket.
      4. Select users and permissions to assign to the end user.

         Only Read and Write permissions can be assigned to the end user.

      The following sample image shows the **Create Policy** page in Ranger UI to create an S3 policy on an S3 path for an end user.

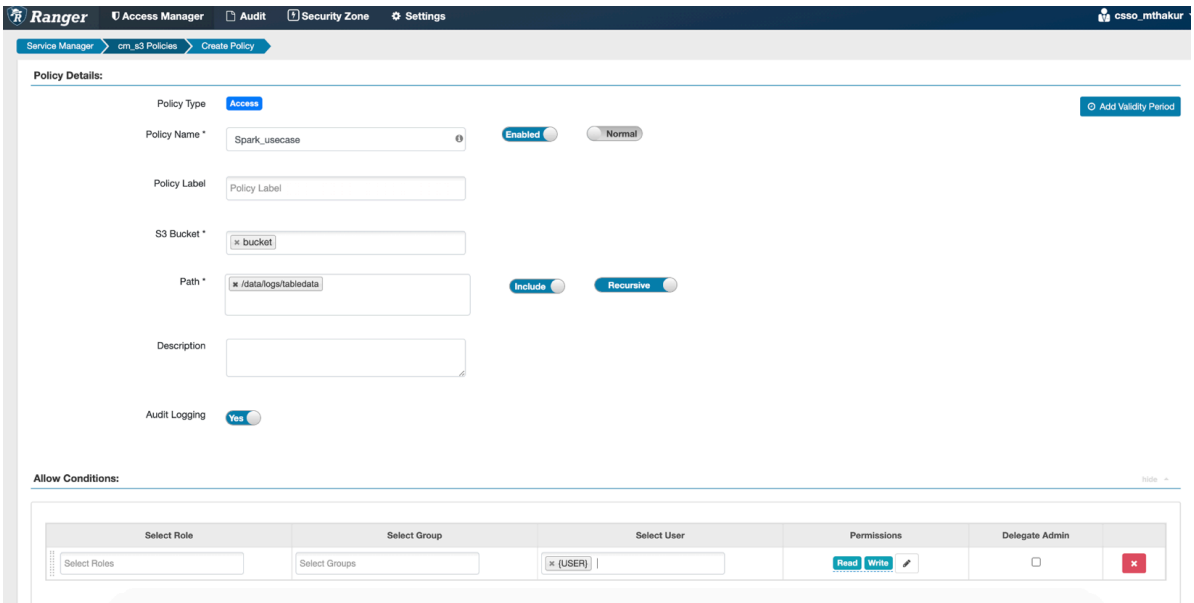

   e) Click Add to save the policy.

2. To create a Hive URL authorization policy on an S3 path for the end user, perform the following steps:
   a) Navigate to the Ranger UI.
   b) On the Hadoop SQL tab, click Hadoop SQL.
   c) Click Add New Policy in the top right corner.
   d) Provide the policy details. The following sample image shows the policy details:
      1. Enter Policy Name.
      2. Enter the Hive URL authorization path in the url field, and enable the Recursive option.
      3. Provide a Path within the S3 bucket.

         Note: You can append *, also known as a "wildcard", to the path name. For example: s3a://bucket/* . Appending * to a URL path grants (or denies) access to the child directories in that path.

      4. Select users and permissions to assign to the end user.

         Note: Only Read permission can be assigned to the end user.

      The following sample image shows the Policy Details page in Ranger UI to create a Hive URL authorization policy on an S3 path for an end user.

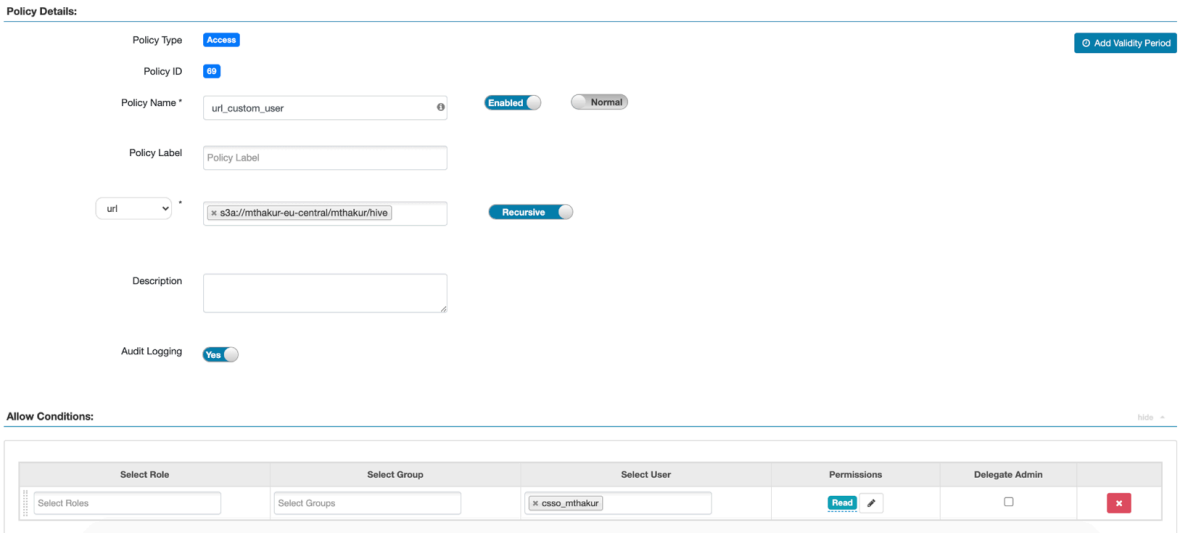

   e) Click Add to save the policy.

Troubleshooting for RAZ-enabled AWS environment

This section includes common errors that might occur while using a RAZ-enabled AWS environment and the steps to resolve the issues.

Why does the "AccessDeniedException" error appear? How do I resolve it?

Complete error snippet:

```
org.apache.hadoop.hive.ql.metadata.HiveException: java.nio.file.AccessDenied
Exception:
s3a://abc-eu-central/abc/hive: getFileStatus on s3a://abc-eu-central/abc/hiv
e:
```

```
com.amazonaws.services.signer.model.AccessDeniedException: Ranger result: DE
NIED, Audit:
[AuditInfo={auditId={null} accessType={read} result={NOT_DETERMINED} polic
yId={-1} policyVersion={null} }],
Username: hive/demo-dh-master0.abc.xcu2-8y8x.dev.cldr.work@abc.XCU2-8Y8X.
DEV.CLDR.WORK (Service: null;
Status Code: 403; Error Code: AccessDeniedException; Request ID: f137c7b2
-4448-4599-b75e-d96ae9308c5b;
Proxy: null):AccessDeniedException
```

Cause

This error appears when the hive user does not have the required S3 policy.

Remedy

Add the required policy for the user.

For more information, see Ranger policy options for RAZ-enabled AWS environment on page 11.

Why does the "Permission denied" error appear? How do I resolve it?

Complete error snippet:

```
Error: Error while compiling statement:
FAILED: HiveAccessControlException Permission denied: user [csso_abc] does
not have
[READ] privilege on [s3a://abc-eu-central/abc/hive] (state=42000,code=40000)
```

Cause

This error appears if you did not add the required Hive URL authorization policy to the user.

Remedy

Procedure

Add the required policy for the user.

For more information, see Ranger policy options for RAZ-enabled AWS environment on page 11.

Why does the "S3 access in hive/spark/mapreduce fails despite having an "allow" policy" error message appear? How do I resolve it?

Complete error snippet:

```
S3 access in hive/spark/mapreduce fails despite having an "allow" policy def
ined for the path
with ‘java.nio.file.AccessDeniedException: s3a://sshiv-cdp-bucket/data/ex
ttable/000000_0: getFileStatus on
s3a://sshivalingamurthy-cdp-bucket/data/exttable/000000_0: com.amazonaws.
services.signer.model.AccessDeniedException:
Ranger result: NOT_DETERMINED, Audit: [AuditInfo={auditId={a6904660-1a0f-3
149-8a0b-c0792aec3e19} accessType={read}
result={NOT_DETERMINED} policyId={-1} policyVersion={null} }] (Service: nu
ll; Status Code: 403; Error Code: AccessDeniedException;
Request ID: null):AccessDeniedException’
```

Cause

In some cases, RAZ S3 requires a non-recursive READ access on the parent S3 path. This error appears when the non-recursive READ access is not provided to the parent S3 path.

Remedy

Procedure

1. On the Ranger Audit page, track the user and the path where authorization failed.
2. Add the missing Ranger policy to the end user.

   The following sample image shows the **Access** tab on the Ranger Audit page where you can track the user and the path:

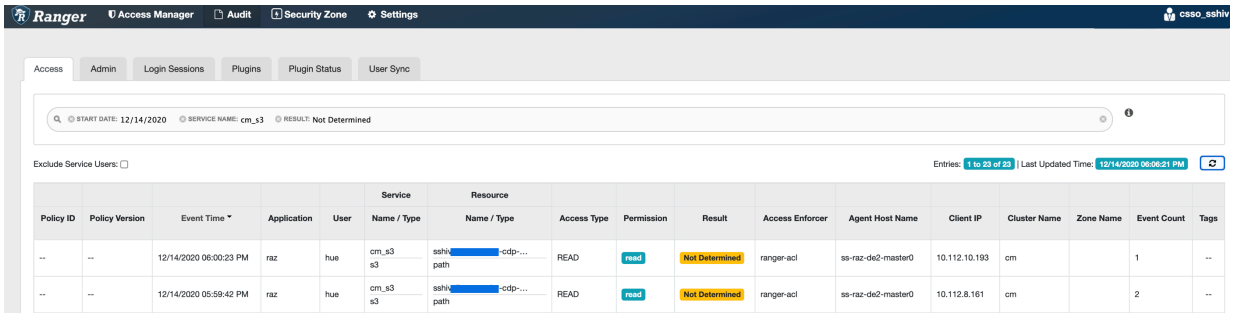

An error related to Apache Flink appears after the job fails. How do I resolve this issue?

Complete error snippet:

```
2021-06-01 00:00:18,872 INFO  org.apache.flink.runtime.entrypoint.ClusterEnt
rypoint
- Shutting YarnJobClusterEntrypoint down with application status FAILED. Di
agnostics java.nio.file.AccessDeniedException:
s3a://jon-s3-raz/data/flink/araujo-flink/ha/application_1622436888784_0031/b
lob: getFileStatus on
s3a://jon-s3-raz/data/flink/araujo-flink/ha/application_1622436888784_0031/
blob: com.amazonaws.services.signer.model.AccessDeniedException:
Ranger result: DENIED, Audit: [AuditInfo={auditId={84aba1b3-82b5-3d0c-a05b-8
f5512e0fd36} accessType={read} result={NOT_DETERMINED}
policyId={-1} policyVersion={null} }], Username: srv_kafka-client@PM-AWS-R.A
465-4K.CLOUDERA.SITE (Service: null; Status Code: 403;
Error Code: AccessDeniedException; Request ID: null; Proxy: null):AccessDeni
edException
```

Cause

This error appears if there is no policy granting the necessary access to the /data/flink/araujo-flink/ha path.

Remedy

Procedure

Add the policy to grant the required access to the /data/flink/araujo-flink/ha path.

To configure the policy, see Configuring Ranger policies for Flink.

What do I do to display the Thread ID in logs for RAZ?

When you enable the DEBUG level for RAZ, a detailed verbose log is generated. It is cumbersome and tedious to identify the issue pertaining to a specific RAZ authorization request; therefore, Thread ID is good information to capture in the debug log.

Remedy

Procedure

1. In Cloudera Manager, go to the Ranger RAZ service on the Configuration tab.
2. Search for the Ranger Raz Server Logging Advanced Configuration Snippet (Safety Valve) property, and enter the following information:

```
log4j.logger.org.apache.ranger.raz=DEBUG
log4j.appender.RFA.layout.ConversionPattern=[%p] %d{yy/MM/dd HH:mm:ss,S
SS} [THREAD ID=%t]
          [CLASS=(%C:%L)] %m%
```

3. Search for the Ranger Raz Server Max Log Size property, enter 200, and choose MiB.
4. Click Save Changes.
5. Restart the Ranger RAZ service.

   The service prints the debug log details with Thread ID. To debug an issue, you can filter the logs based on the Thread ID.

   The sample snippet shows the log generated with Thread ID:

What do I do when a long-running job consistently fails with the expired token error?

Sample error snippet - “Caused by: com.amazonaws.services.s3.model.AmazonS3Exception: The provided token has expired. (Service: Amazon S3; Status Code: 400; Error Code: ExpiredToken; Request ID: xxxx; S3 Extended Request ID:xxxx...“

Cause

This issue appears when the S3 Security token has expired for the s3 call.

Remedy

Procedure

1. Log in to the Cloudera Management Console as an Administrator and go to your environment.
2. From the Data Lake tab, open Cloudera Manager.
3. Go to the Clusters *HDFS SERVICE* Configurations tab.
4. Search for the core-site.xml file corresponding to the required Cloudera Data Hub cluster.
5. Open the file and enter fs.s3a.signature.cache.max.size=0 to disable the signature caching in Ranger RAZ.
6. Save and close the file.
7. On the Home Status tab, click Actions to the right of the cluster name and select Deploy Client Configuration.
8. Click Deploy Client Configuration.
9. Restart the HDFS service.

Note: Ensure that there are no stale configurations in Ranger RAZ and HDFS services.

Managing a RAZ-enabled AWS environment

You can manage a RAZ-enabled environment in a similar manner as any other Cloudera environment.

For information on how to manage and monitor Cloudera environments running in AWS, refer to Working with AWS environments.

Cloudera Management Console

Fine-grained access control for ADLS Gen2

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to RAZ on Azure environments..... 4

Prepare to register RAZ-enabled Azure environment..... 5
Creating Ranger RAZ managed identity for RAZ-enabled Azure environment..... 5
Creating custom role to use in RAZ-enabled Azure environment..... 6

Registering a RAZ-enabled Azure environment..... 7
Using the Cloudera UI to register RAZ-enabled Azure environment..... 7
Using the CDP CLI to register RAZ-enabled Azure environment.....8

Cluster templates available in RAZ-enabled Azure environment..... 9

Configuring Data Mart for RAZ-enabled Azure environment.....9

Ranger policies for RAZ-enabled Azure environment..... 10

Troubleshooting for RAZ-enabled Azure environment.....10
How do I view logs and RAZ configuration in a RAZ-enabled Azure environment?..... 10
How do I enable the debug level for RAZ server and RAZ client?.....11
Is there a checklist to refer while troubleshooting RAZ-enabled Azure environments?..... 12
What are the most common errors in RAZ-enabled Azure environments?.....12
How do I generate a UserDelegationKey manually?..... 13
How do I download RAZ policies manually?..... 14

Managing a RAZ-enabled Azure environment..... 14

Introduction to RAZ on Azure environments

Shared Data Experience (SDX) in Cloudera Public Cloud provides Ranger Authorization Server (RAZ) service for fine grained access control and auditing of various services and workloads running in Enterprise Data Cloud. To use RAZ server capabilities, you must first enable RAZ in an Azure environment in Cloudera Public Cloud.

Cloudera Public Cloud defaults to using cloud storage which might be challenging while managing data access across teams and individual users. The Ranger Authorization Service (RAZ) resolves this challenge by enabling ADLS Gen2 users to use fine-grained access policies and audit capabilities available in Apache Ranger similar to those used with HDFS files in an on-premises or IaaS deployment.

Prior to the introduction of RAZ, controlling access to ADLS Gen2 could be enforced at coarse-grained group level (using IDBroker mappings). This required rearchitecting the implementation of important file-centric activities as well as admin-level access to both the Azure subscription and Cloudera account.

Important: It is recommended that you do not setup IDBroker mapping for workload users in a RAZ-enabled Azure environment.

In HDP and CDH deployments, files and directories are protected with a combination of HDFS Access Control Lists (ACLs) (in CDH, HDP) and Ranger HDFS policies (in HDP). Similarly, in an Azure Cloudera Public Cloud environment with RAZ for ADLS Gen2 enabled, Ranger's rich access control policies can be applied to Cloudera's access to ADLS Gen2 containers, directories, and files and can be controlled with admin-level access to Cloudera alone.

Important: It is recommended that you do not setup IDBroker mapping for workload users in a RAZ-enabled Azure environment.

Supported use cases for RAZ in Azure environments

Many of the use cases that RAZ for Azure enables are cases where access control on files or directories is needed. Some examples include:

- Per-user home directories.
- Data engineering (Spark) efforts that require access to cloud storage objects and directories.
- Data warehouse queries (Hive/Impala) that use external tables.
- Access to Ranger's rich access control policies such as date-based access revocation, user/group/role-based controls, along with corresponding audit.
- Tag-based access control using the classification propagation feature that originates from directories.

The core RAZ for Azure for Data Lakes and several Cloudera Data Hub templates are available for production use. The following Cloudera Data Hub cluster types are supported:

- Cloudera Data Engineering
- Cloudera Data Engineering HA
- Cloudera Data Engineering Spark3
- Cloudera Operational Database with SQL

Specifically, Hive, Spark, HBase, and Oozie are supported.

RAZ is fully integrated with the following Cloudera data services:

- Cloudera DataFlow
- Cloudera Data Engineering
- Cloudera Machine Learning
- Cloudera Operational Database

You can backup and restore the metadata maintained in the Data Lake services of RAZ-enabled environments. For more information, see Data Lake Backup and Restore.

Important: Integrations with some components are under technical preview or has limited availability. Contact your account team before you use RAZ for your production use cases.

Limitations to use RAZ in Azure environments

The following limitations and known issues have been identified and are under development:

- Currently, there is no automated way to enable RAZ in an existing Cloudera environment that does not have RAZ enabled.
- RAZ integration is under technical preview for the following Cloudera services:
  - Cloudera Data Warehouse
  - Integration with Cloudera Data Hub Hue's File Browser
- Solr, Kudu, and NiFi are not supported by RAZ.

Prepare to register RAZ-enabled Azure environment

Before you register a RAZ-enabled Azure environment in ClouderaPublic Cloud, you must ensure that the cloud provider prerequisites are in place.

The RAZ service consists of RAZ client and RAZ server. The RAZ client is integrated into the HDFS driver. The Azure environment in ClouderaPublic Cloud can use the RAZ client and RAZ server capabilities after you register the RAZ-enabled Azure environment.

The Azure prerequisites for a RAZ-enabled Azure environment are described in the Azure requirements. When meeting these prerequisites, ensure that you do the following:

- Pre-create a resource group on Azure.

  Using resource groups created by Cloudera with a RAZ-enabled environment is not supported. See Resource groups.
- Create a designated Ranger RAZ managed identity as described in the Minimum setup for cloud storage and in Creating Ranger RAZ managed identity for RAZ-enabled Azure environment on page 5. You can optionally create a custom policy for the Ranger RAZ managed identity.

Creating Ranger RAZ managed identity for RAZ-enabled Azure environment

In addition to creating the required managed identities, you should create an additional managed identity named Ranger RAZ for RAZ-enabled Azure environment. You can also optionally create a custom role that can be used instead of Storage Blob Data Owner.

About this task

You can create the required managed identities as described in Minimal setup for cloud storage, and then create the following managed identity to use RAZ in Azure environment.

Managed identity	Managed identity is used for	Roles to assign to the managed identity
Ranger RAZ	Storage Account	• Storage Blob Data Owner or equivalent Creating custom role to use in RAZ-enabled Azure environment on page 6 • Storage Blob Delegator

Procedure

1. Perform the following steps to create the *RANGER RAZ* managed identity using Azure Portal:
   a) On Azure Portal, navigate to Managed Identities.
   b) Click +New.
   c) Select the Resource group used for Cloudera.
   d) Select your environment's Region.
   e) Specify managed identity Name. For example, Ranger RAZ.
   f) Provide tags if required by your organization.
   g) Click Review + create.
2. Perform the following steps to assign the two roles to the *RANGER RAZ* managed identity on the scope of the storage account created for Cloudera:
   a) In your Azure Portal, navigate to the Storage accounts *[***YOUR STORAGE ACCOUNT***]* Access Control (IAM) page.
   b) Click +Add Add role assignment .
   c) In the Add role assignment section, choose the following options:
      1. Select Storage Blob Data Owner as Role.
      2. Select User assigned managed identity as Assign access to.
      3. Select the Ranger RAZ managed identity that you created earlier.
      4. Click Save.
   d) To assign the Storage Blob Delegator role to *RANGER RAZ* managed identity, repeat steps a through c.

Creating custom role to use in RAZ-enabled Azure environment

Your Azure administrator can optionally create a custom role in the Azure subscription that can be used instead of Storage Blob Data Owner in a RAZ-enabled Azure environment.

Procedure

The Azure administrator can optionally create a custom role with required permissions to use instead of Storage Blob Data Owner. This role can be used to register a RAZ-enabled Azure environment and to create Cloudera Data Hub clusters and Cloudera Operational Databases using the following policy definition:

```
{
   {
        "properties": {
            "roleName": "Cloudera CDP Storage Authorization",
            "description": "Provide privileges that Cloudera CDP requires f
or storage access",
            "assignableScopes": [
            "/subscriptions/abce3e07-b32d-4b41-8c78-2bcaffe4ea27"
            ],
            "permissions": [
                {
                    "actions": ["Microsoft.Storage/storageAccounts/blobSer
vices/generateUserDelegationKey/action"],
                    "notActions": [],
                    "dataActions": ["Microsoft.Storage/storageAccounts/blobS
ervices/containers/blobs/manageOwnership/action",
                    "Microsoft.Storage/storageAccounts/blobServices/contai
ners/blobs/modifyPermissions/action",
                    "Microsoft.Storage/storageAccounts/blobServices/contai
ners/blobs/read",
                    "Microsoft.Storage/storageAccounts/blobServices/contai
ners/blobs/write",
```

```
                    "Microsoft.Storage/storageAccounts/blobServices/conta
iners/blobs/delete",
                    "Microsoft.Storage/storageAccounts/blobServices/contain
ers/blobs/move/action"
                            ],
                    "notDataActions": []
                }
            ]
        }
    }
```

Registering a RAZ-enabled Azure environment

You can use the Cloudera web interface or CDP CLI to register a RAZ-enabled Azure environment.

You can enable RAZ on the latest available version of Cloudera Runtime. The minimum Cloudera Runtime version supporting RAZ for Azure environments is 7.2.11.

When you register an Azure environment, enable the Fine-grained access control on ADLS Gen2 option, and then provide the managed identity that you created earlier.

After you implement RAZ by registering the RAZ-enabled Azure environment, the following RAZ authorization steps are completed by the RAZ server automatically:

1. Coordinates with IDBroker to generate and cache the user delegation token.
2. Updates the cached user delegation token periodically.
3. Authorizes the collated information from the RAZ client. When an action is performed on a given cloud storage or path, the RAZ client collates the information and sends it to the RAZ server for further processing.

The RAZ server generates the responses based on the requests.

The following table lists the request and the response generated by the RAZ server for the request:

Request	Response
Allowed	RAZ server responds with a DSAS token. HDFS driver uses this token information to access cloud storage or path. For more information about the request, login to Ranger Admin UI and check the access audit reports.
Denied	RAZ server returns **Access denied** response.
Not Determined	RAZ server responds with a DSAS token only if the RAZ service is configured to fallback. Otherwise, the RAZ server returns an **Access denied** response.
Server failed	If RAZ failed to process a request, the **RAZ server failed to process the request** error appears.

Using the Cloudera UI to register RAZ-enabled Azure environment

Once you've met the Azure cloud provider requirements, register your Azure environment in Cloudera.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in Azure requirements.

Required role: EnvironmentCreator

Procedure

1. Log in to the Cloudera web interface.

2. Navigate to the Cloudera Management Console Environments Register environment page.
3. On the Register Environment page, provide the following information:
   a) Enter an Environment Name.
   b) Select a provisioning credential.
4. Click Next.
5. On the Data Access and Data Lake Scaling page, provide the following information:
   a) Enter a Data Lake Cluster Name.
   b) Make sure to select Runtime 7.2.11 or a higher version from the Data Lake version drop-down menu.
   c) In the Data Access and Audit section, provide your data storage location and managed identities created for minimal setup for cloud storage.
   d) In the Fine-grained access control on ADLS Gen2 section, click on the toggle button to enable Ranger authorization for ADLS Gen2 and select the managed identity created for RAZ.

      The following image shows the Fine-grained access control on ADLS Gen2 section where you can enable the Ranger authorization for ADLS and choose an Azure managed identity for Ranger authorizer:

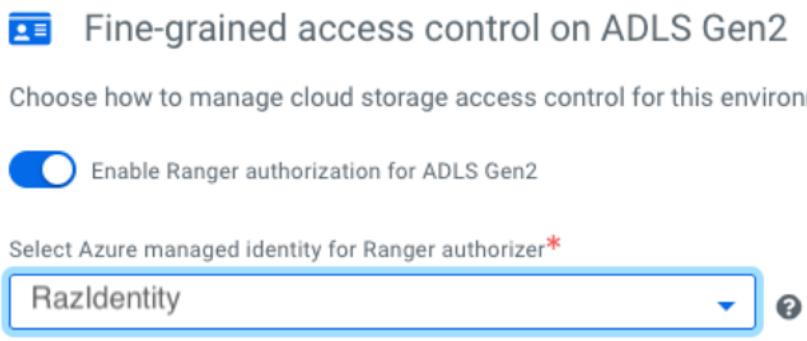

6. Click Next.
7. On the Region, Networking and Security page, provide the following information:
   a) Select the region.
   b) Select an existing resource group. Creating new resource groups is not supported.
   c) Select the network, security groups, and provide an SSH key. If required, add tags.
8. Click Next.
9. On the Storage page, provide your logs storage location and managed identities created for minimal setup for cloud storage.
10. Click Register Environment to trigger environment registration.

Using the CDP CLI to register RAZ-enabled Azure environment

You can use the CDP CLI to register a RAZ-enabled Azure environment. You must download and install beta CDP CLI, and then use CDP CLI commands to register a RAZ-enabled Azure environment.

Procedure

1. To install beta CDP CLI, see Installing Beta CDP CLI.
2. To register a RAZ-enabled Azure environment, use the --ranger-cloud-access-authorizer-role *[***RAZ_IDENTITY***]* CDP CLI command.

   If you have CDP CLI templates to create an Azure environment, modify them by adding the additional parameter required for RAZ.

   The additional option is highlighted in the following sample snippet:

```
cdp environments create-azure-environment \
--environment-name [***ENVIRONMENT_NAME***] \
```

```
--credential-name [***CREDENTIAL_NAME***] \
--region [***REGION***] \
--security-access cidr=[***YOUR_CIDR***] \
--public-key [***SSH_PUBLIC_KEY***] \
--log-storage [***LOG_STORAGE_CONFIGURATION***]\
--use-public-ip
--resource-group-name [***EXISTING_RESOURCE_GROUP***]

cdp environments set-id-broker-mappings \
--environment-name [***ENVIRONMENT_NAME***] \
--data-access-role [***DATA_ACCESS_IDENTITY***]\
--ranger-audit-role [***RANGER_AUDIT_IDENTITY***] \
--ranger-cloud-access-authorizer-role [***RAZ_IDENTITY***] \
--set-empty-mappings

cdp datalake create-azure-datalake \
--datalake-name [***DATALAKE_NAME***] \
--environment-name [***ENVIRONMENT_NAME***] \
--cloud-provider-configuration [***STORAGE_LOCATION_BASE_CONFIGURATION***]
 \
--enable-ranger-raz
```

Note: You can obtain CDP CLI commands for environment creation from CDP CLI help on Cloudera web interface. For more information, see Obtain CLI commands for registering an environment.

Cluster templates available in RAZ-enabled Azure environment

After your Azure environment is running, use the usual steps to create Cloudera Data Hub clusters to use in a RAZ-enabled Azure environment. You can create custom variants of these templates in a RAZ-enabled Azure environment.

The following cluster templates have been tested and can be used in a RAZ-enabled Azure environment:

- Cloudera Data Engineering
- Cloudera Data Engineering HA
- Cloudera Data Engineering Spark3
- Data Mart
- Cloudera Operational Database with SQL

Configuring Data Mart for RAZ-enabled Azure environment

The Data Mart template provides a ready to use, fully capable, standalone deployment of Impala. To use a Data Mart cluster in a RAZ-enabled Azure environment, you need to configure additional Data Mart configuration steps in the Data Mart for Azure cluster definition.

Procedure

1. Log in to the Cloudera Manager for the Data Mart.

2. Select Impala Configuration , and then set the value of the Impala Daemon Data Cache Per Directory Capacity property to 10 GB.

   This property must be set to avoid the following Impala Daemon start failure:

```
F0615 07:11:40.206655 62270 impalad-main.cc:73] Insufficient space for /
hadoopfs/fs1/impala/datacache.
Required 130.00 GB. Only 29.53 GB is available
. Impalad exiting.
```

3. Create a Ranger ALDS policy in the cm_adls service with the following properties:

   Note that the following property values are sample values. You can enter the values as required for your environment.

   - Storage Account *= mpdsasv2san
   - Storage Account Container * = data
   - Relative Path * = /
   - Recursive = ON
   - User = impala
   - Permissions = read, add, create, write, delete, list, move, superuser

   This policy must be created to avoid the following failure:

```
E0615 07:58:02.019865 71092 impala-server.cc:383] Could not read the root
 directory at abfs://data@mpdsasv2san.dfs.core.windows.net/mp-dsasv2-dm.
 Error was:
Failed to acquire a SAS token for list on / due to org.apache.hadoop.se
curity.AccessControlException: org.apache.ranger.raz.intg.RangerRazExcep
tion: request failed: status=403
```

4. Restart the Impala service in Cloudera Manager, then verify that the Ranger Azure authorization is working as expected.

Ranger policies for RAZ-enabled Azure environment

Ranger enables you to configure resource-based services for Hadoop components and add access policies to those services. Ranger includes a set of preloaded resource-based services and policies which you can use in a RAZ-enabled Azure environment.

For more information, see Preloaded resource-based services and policies

Troubleshooting for RAZ-enabled Azure environment

This section includes FAQs, and discusses some common errors that might occur while using a RAZ-enabled Azure environment and the steps to resolve the issues.

How do I view logs and RAZ configuration in a RAZ-enabled Azure environment?

You can use the following methods to view logs and RAZ configuration in a RAZ-enabled Azure environment:

- To view RAZ server logs, perform the following steps:
  1. Identify the host on which the RAZ service is installed in Cloudera Manager.
  2. SSH to log into the RAZ server.
  3. Locate the ranger-raz-*[***HOST-NAME***]*-*-rangerraz.log file in the /var/log/ranger/raz/ directory.
- To view the RAZ client logs, go to the respective component logs.

  RAZ authorization is triggered when a Cloudera Public Cloud service accesses the ADLS storage through the HDFS client layer. The logs related to these calls are stored by the respective service or component logs.

  The following table lists some services and the RAZ client log location:

Service	RAZ client log location
HiveServer2	/var/log/hive/ folder. The associated HS2 and HMS logs in this directory provide the details about the Apache RAZ client calls.
YARN MapReduce Hive on Tez Spark	After the job run, you can locate the RAZ client logs in the corresponding application log. The Resource Manager creates the application log. You can access the logs through Resource Manager or use YARN command line tools.

- To view RAZ server configuration, perform the following steps:
  1. Identify the host on which the RAZ service is installed in Cloudera Manager.
  2. SSH to log into the RAZ server.
  3. Run the ps -ef | grep rangerraz | awk '{split($0, array,"classpath"); print array[2]}' | cut -d: -f1 command to identify the RAZ_CONF_DIR directory.
  4. Run the cat *[***RAZ_CONF_DIR***]*/ranger-raz-site.xml command to view the RAZ server configuration details.

How do I enable the debug level for RAZ server and RAZ client?

When you enable the DEBUG level for RAZ server and RAZ client, a detailed verbose log is generated. Alternatively, you can only enable debug on a package level, so that only the package logs show detailed verbose logs.

- To configure debug for RAZ server at package level, perform the following steps:
  1. Go to the Cloudera Manager *RANGER RAZ SERVICE* Configuration tab.
  2. Search for the **Ranger Raz Server Logging Advanced Configuration Snippet (Safety Valve) property** property, and enter the following information:

     ```
     log4j.logger.org.apache.ranger.raz.[***package name***]=DEBUG
     log4j.logger.org.apache.hadoop.security=DEBUG
     log4j.appender.RFA.layout.ConversionPattern=[%p] %d{dd/MM/yyyy HH:mm:ss,
     SSS} [THREAD ID=%t] [CLASS=(%C{1}:%L)] %m%n
     ```
  3. Click Save Changes.
  4. Restart the Ranger RAZ service.
- You can configure the DEBUG level for RAZ clients.

  For example, to view the HDFS log in DEBUG level, run the HADOOP_ROOT_LOGGER=DEBUG,console hdfs *[***OPTIONS***]* *[***ARGUMENTS***]* command.

Is there a checklist to refer while troubleshooting RAZ-enabled Azure environments?

You can use the following brief checklist to help you to troubleshoot RAZ-enabled environments:

- Is Hierarchical NameSpace enabled for ADLS Gen2?
- Does the Ranger RAZ managed identity have the Storage Blob Delegator and Storage Blob Data Owner roles?
- Does the RAZ-enabled Azure environment use a zone-redundant storage (ZRS)?
- Are the Ranger Admin cloud policies (cm_adls) created with the correct storage account & paths based on the corresponding environment?
- Do you have sufficient permissions in Ranger Admin cloud policies (cm_adls) in the corresponding environment?
- Can the RAZ servers in Data Lake and workload clusters download Ranger policies from the Ranger Admin UI?
- Can the RAZ servers in Data Lake and workload clusters download Ranger UserStore?
- Can the RAZ servers in Data Lake and workload clusters connect to IDBroker and download the UserDelegationKey?

What are the most common errors in RAZ-enabled Azure environments?

The following table lists the most common errors, their causes, and solution to resolve the errors:

Error and error location	Cause and Solution
Failed to communicate with Ranger Admin and Error getting UserStore errors appear in RAZ Server logs.	These errors appear when the RAZ server cannot download the latest version of the user-store (user information) because the Ranger Admin service cannot reach the RAZ server host. To resolve this issue, verify whether the Ranger Admin service is up and reachable from the RAZ server host. If not, fix the connectivity issue.
AbfsTokenProvider: IdBroker initialization failed error appears in RAZ Server logs.	This error appears when the RAZ server cannot connect to IDBroker to obtain the UserDelegationKey that is used for authorizing requests in ADLS paths. To resolve this issue, verify whether IDBroker is up and reachable for the RAZ server host. If not, fix the connectivity issue.
DTAwsCredentialsProvider failed when trying to get token from idbroker, or AbfsTokenProvider: ==> AbfsTokeProvider.fetchUserDelegationKey and fetchAccessToken(): null response received from IDBroker errors appear in RAZ Server logs.	These errors appear when the RAZ server cannot connect to IDBroker to obtain the UserDelegationKey that is used for authorizing requests in S3 paths. To resolve this issue, verify whether IDBroker is up and reachable for the RAZ server host. If not, fix the connectivity issue.
Failed to communicate with all Raz URLs, Verify that the URLs are correct and corresponding services are running & accessible, and Multiple service types are not supported messages appear in RAZ client logs.	This issue appears when the RAZ server setup fails because the ranger.raz.bootstrap.servicetypes property in the ranger-raz-site.xml file has multiple service-types such as adls, s3 that are not supported. To resolve this issue, configure the ranger.raz.bootstrap.servicetypes property in the ranger-raz-site.xml file to the required cluster type, remove the other service types, and save the file. For example, enter s3 for AWS cluster type and enter adls for Azure cluster type.
Server did not process this request due to in-sufficient auth details, Failed to acquire a SAS token… AccessControlException:, HTTP Status 401 – Unauthorized appear in RAZ client logs.	These error messages appear when the RAZ server denies the authorization request because no or incomplete authentication details are available. To resolve this issue, you must ensure that the Ranger RAZ server has Kerberos or RAZ-DT & JWT authentication. You must have a valid TGT when the RAZ server supports Kerberos authentication.

Error and error location	Cause and Solution
Failed to get DSAS token from Raz, HttpStatus: 403, Failed to acquire a SAS token … : Permission denied messages appear in RAZ client logs.	These errors might appear because of various reasons, the following are a few possible causes and solutions: • The required Ranger policies are not set. To verify, the administrator must check the Ranger Admin Access Audits and if required, assign the required policies. • The RAZ server might not be able to fetch the User Delegation key from IDBroker. To verify, check the Ranger RAZ server logs. To resolve this issue, generate the User Delegation Key manually. If you cannot generate DSAS manually, this indicates that a network connectivity issue exists. To proceed, you must resolve the connectivity issues. • RAZ server cannot fetch the latest Ranger admin policies/users. To verify, download the RAZ policies manually. If you cannot download RAZ policies manually, this indicates a network connectivity issue. To proceed, resolve the connectivity issues.

How do I generate a UserDelegationKey manually?

Perform the following steps to generate a UserDelegationKey manually:

Procedure

1. Identify the host on which the RAZ service is installed in Cloudera Manager.
2. SSH to log into the RAZ server.
3. Run following commands to get the required parameters:
   a. ps -ef | grep rangerraz | awk '{split($0, array,"classpath"); print array[2]}' | cut -d: -f1 to identify the RAZ_CONF_DIR directory.
   b. klist -kt [***RAZ_CONF_DIR***]/../ranger_raz.keytab to identify the RAZ_PRINCIPLE parameter. The RAZ_PRINCIPLE parameter starts with *RANGERRAZ*.
   c. cat /etc/alternatives/hadoop-conf/core-site.xml | grep -a1 'fs.azure.ext.cab.address' | grep 'value' to identify the complete ID_BROKER_URL.
4. Update the expiry date in the sasPost.xml file to another valid date.
5. Run following commands:
   a. kinit -kt [***RAZ_CONF_DIR***]/../ranger_raz.keytab [***RAZ_PRINCIPLE***]
   b. DT=$(curl --negotiate -u : "[***ID_BROKER_URL***]/dt/knoxtoken/api/v1/token" | jq -r '.access_token')
   c. AT=$(curl -k -H "Authorization: Bearer $DT" '[***ID_BROKER_URL***]/azure-cab/cab/api/v1/credentials' | jq -r '.access_token')
   d. curl -v -i -X POST -d @sasPost.xml -H "x-ms-version: 2018-11-09" -H "Authorization: Bearer $AT" "https://[***ACCOUNT***].blob.core.windows.net/?restype=service&comp=userdelegationkey"

   The UserDelegationKey is generated.
6. In case, the UserDelegationKey is not generated as expected, run the following commands to verify whether the identity mappings are available:
   a. SSH to the IDBroker host.
   b. Go to the cd /var/log/knox/idbroker/ IDBroker logs directory.
   c. Run the grep -rl “addIdentitiesToVM” * command to view the list of identities. Note that the output must not be empty.
   d. Run the grep . -rnwe ‘addIdentitiesToVM’ command to view the assumerIdentity, adminIdentity, and rangerraz identities.

How do I download RAZ policies manually?

You can perform the following steps to download the RAZ policies manually:

Procedure

1. Identify the host on which the RAZ service is installed in Cloudera Manager.
2. SSH to log into the RAZ server.
3. Run following commands to identify the required parameters:
   a. ps -ef | grep rangerraz | awk '{split($0, array,"classpath"); print array[2]}' | cut -d: -f1 to identify the RAZ_CONF_DIR directory.
   b. klist -kt *[***RAZ_CONF_DIR**]*/../ranger_raz.keytab to identify the RAZ_PRINCIPLE. The RAZ_PRINCIPLE starts with rangerraz.
   c. cat /var/run/cloudera-scm-agent/process/1546336485-ranger_raz-RANGER_RAZ_SERVER/ranger-raz-conf/ranger-raz-site.xml | grep -a1 'ranger.raz.policy.rest.url' | grep 'value' to identify the ADMIN_URL. Use the ADMIN_URL after you remove any trailing slash.
4. Run following commands to download the RAZ policies:
   a. kinit -kt [***RAZ_CONF_DIR***]/../ranger_raz.keytab *[***RAZ_PRINCIPLE***]*
   b. curl -ikv --negotiate -u : -X GET -H "Accept:application/json" -H "Content-Type:application/json" "*[***ADMIN_URL***]*/service/plugins/secure/policies/download/cm_adls

   The RAZ policies for the ADLS service are downloaded to the current directory of the logged in host or server.

Managing a RAZ-enabled Azure environment

You can manage a RAZ-enabled environment in a similar manner as any other Cloudera environment.

For information on how to manage and monitor Cloudera environments running in Azure, refer to Working with Azure environments.

Cloudera Management Console

Identity Management

Date published: 2019-08-22
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing FreeIPA 4

Creating an environment configured with FreeIPA 5

Show FreeIPA instance status 6
- FreeIPA instance status options 8

Vertically scale FreeIPA instances 8

Upgrade FreeIPA 10

Repair FreeIPA 13
- Check FreeIPA repair status 14

Resize FreeIPA 15

Configuring workload password policies 16
- Check the current workload password policy 17
- Set a password policy 18
- Reset a password policy 20
- Unset user's minimum password lifetime 21

Accessing a FreeIPA cluster via SSH 21

Recipes 22
- Writing recipes 23
- Register a recipe 25
- Update a FreeIPA recipe 26
- Managing recipes from CLI 27

Managing FreeIPA

FreeIPA is the backbone of the Cloudera Identity Management functionality. After you configure a Cloudera environment, FreeIPA works to provide user identities without the need for your attention. In case of problems, you may need to perform troubleshooting to ensure the health of the identity management system.

FreeIPA availability types

Historically, FreeIPA node count (also known as "availability types") can be one of the following:

- 3 nodes (HA)
- 2 nodes
- 1 node

When registering a Cloudera environment via web UI, you cannot select FreeIPA node count, but Cloudera adjusts FreeIPA based on the Data Lake scale that you select:

- For an Enterprise Data Lake, a FreeIPA cluster with 3 nodes (HA) is provisioned
- For a Medium Duty Data Lake, a FreeIPA cluster with 3 nodes (HA) is provisioned
- For a Light Duty Data Lake, a FreeIPA cluster with 2 nodes is provisioned

Attention: If you scale an existing Data Lake, for example scaling from light duty to medium duty, the FreeIPA node count is unaffected.

When registering a Cloudera environment via CDP CLI, you can select the node count manually. You can choose to create 1+ (up to a maximum of 3). If you do not specify the node count, Cloudera automatically configures FreeIPA with one node only.

When HA is configured, this mode allows automatic failover should one FreeIPA instance fail and a scripted manual process to recover the system with no downtime should it fail.

The Cloudera environment backs up the FreeIPA state periodically (by default, hourly). The backup data is stored on an attached volume (AWS) or managed disk (Azure). This backup allows the state to be recovered in the event of a failure. Without HA mode enabled, recovering from a FreeIPA failure requires a recovery process that is facilitated by Cloudera technical support.

For each running environment, the host and status of the FreeIPA instance is displayed in the environment's Summary tab in the Cloudera Management Console.

FreeIPA HA

By default, Cloudera creates multiple FreeIPA instances and replicates identity management data across all of them. Should there be a conflict synchronizing across instances, the system maintains the "last in" content. If one of the FreeIPA instances fails to pass the environment's status checks, the overall status for FreeIPA turns gray. The FreeIPA clients switch to another FreeIPA instance and the system remains functional. After a week in this state, the identity management system may start to fail from certificates expiring and other problems.

When you see a status other than "Running", follow these steps to investigate and resolve problems:

1. Determine which FreeIPA instance or instances need attention.

   To retrieve a detailed status, use the CDP command-line interface. See “Show FreeIPA instance status”.

2. Repair one or more FreeIPA instances.

   The CDP command-line interface includes a command to trigger a FreeIPA check and reboot repair process. The repair command should resolve most problems with the identity management system. For example, it checks to see if FreeIPA hosts are stopped and restarts them. If hosts are running, the repair process will restart them.

   The repair command may cause cluster outage:

   - If at least one instance is running, the repair command can avoid an outage.
   - If all instances have failed, the repair command won't be able to avoid an outage.
   - If the repair command is run with its "force" option against all instances, the command will cause an outage.

   If the FreeIPA status for the environment returns to "Running", you can stop here.
3. Rebuild the identity management system.

   When repairing a FreeIPA instance, the CDP command-line interface includes a command to trigger a rebuild process that destroys and rebuilds N-1 nodes in the identity management system. This process restores content from the most recent backup and does not require cluster downtime.

   If all nodes need to be repaired or if the unsupervised rebuild process fails, Cloudera Technical Support can help you perform a rebuild of the identity management system and restore content from a backup. This process will require cluster downtime.

FreeIPA failure scenarios

Because FreeIPA is a background system, you are not likely to encounter any failures that include a specific reference to FreeIPA in the error text. Instead, problems with FreeIPA show up as DNS problems, user login problems that raise Kerberos errors, and authentication errors when provisioning workload clusters. If you encounter these general errors, consider checking the status of the FreeIPA system.

Related Information

Creating an environment configured with FreeIPA
Show FreeIPA instance status
Upgrade FreeIPA
Repair FreeIPA
Resize FreeIPA
Configuring workload password policies
Accessing a FreeIPA cluster via SSH
Cloudera identity management

Creating an environment configured with FreeIPA

High-availability FreeIPA is automatically enabled when you register an environment via Cloudera web UI or CLI. The CDP CLI provides environment creation commands that include an option for setting multiple FreeIPA instances, which triggers the system to set up the identity management cluster.

FreeIPA HA cannot be added to an existing environment, only configured during environment creation.

Configuring FreeIPA node count via UI

When registering a Cloudera environment via web UI, you cannot select FreeIPA node count, but Cloudera adjusts FreeIPA based on the Data Lake scale that you select:

- For an Enterprise Data Lake, a FreeIPA cluster with 3 nodes (HA) is provisioned
- For a Medium Duty Data Lake, a FreeIPA cluster with 3 nodes (HA) is provisioned
- For a Light Duty Data Lake, a FreeIPA cluster with 2 nodes is provisioned

Configuring FreeIPA node count via CLI

When registering an environment using CDP CLI, Cloudera configures FreeIPA with 1 node by default. For production, you should change this setting to 3 nodes. To create an environment with selected FreeIPA node count via CLI:

1. Run the CDP CLI command to create an environment and include an additional parameter in the JSON-formatted command input:

   ```
   "freeIpa":{"instanceCountByGroup":<SPECIFY-NODE-COUNT>}
   ```

   Or pass the following parameter:

   ```
   --free-ipa instanceCountByGroup=<SPECIFY-NODE-COUNT>
   ```

   replacing *<SPECIFY-NODE-COUNT>* with the number of instances of FreeIPA you want. The maximum number of instances is 3. Choose 2 or 3 based on the level of redundancy you want. The recommended setting is 3 nodes.
2. After successfully registering the environment, continue to create the Data Lake as described in the cloud-provider specific instructions.

Related Information

Registering an AWS environment
Registering an Azure environment
Show FreeIPA instance status
Installing the Cloudera client

Show FreeIPA instance status

To see the status of each FreeIPA instance in an environment and determine which one(s) need to be repaired, run the CDP CLI get-freeipa-status command.

Steps

For CDP UI

The host and status of the FreeIPA instances is displayed in the environment's FreeIPA Nodes tab in the Cloudera Management Console. If FreeIPA HA is enabled, tab will show status for all hosts:

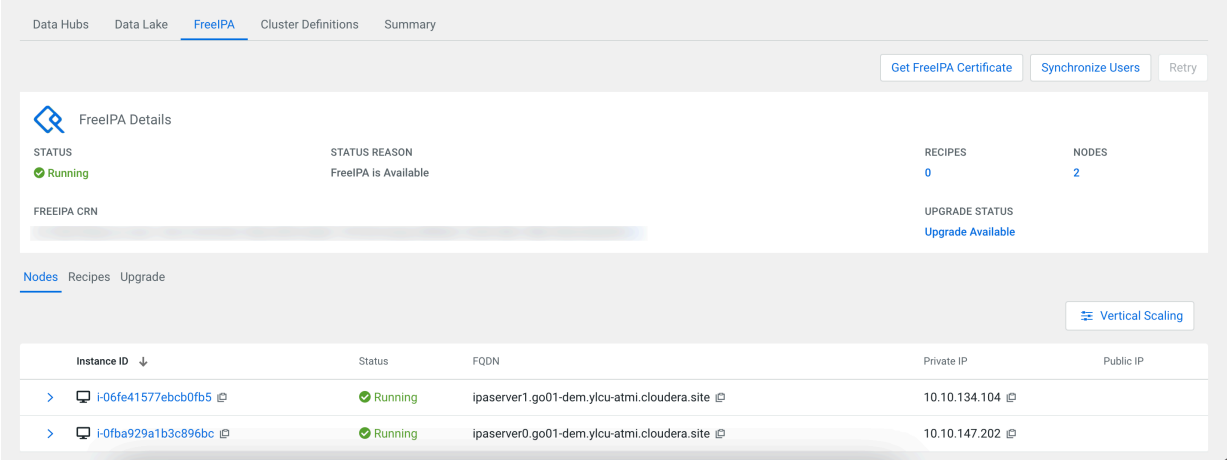

The status shows the general health of the system and the cloud-provider ID of the host. If shown, the ID links to the cloud-provider's page for the host. A full list of status options can be found in FreeIPA instance status options.

For CDP CLI

Run the FreeIPA status command. Run this command from a computer that has network access to the FreeIPA hosts

```
cdp environments get-freeipa-status --environment-name <value>
                      [--cli-input-json <value>]
                      [--generate-cli-skeleton]
```

where the options are the following:

Option	Description
--environment-name *<VALUE>*	Specifies the FreeIPA's environment name or CRN. The environment CRN is listed in Environment Summary General .

This command can take 15 to 45 seconds to run as it gathers information in real-time. The output of the status command provides the status for each FreeIPA node associated with the environment (in JSON format):

```
{
    "environmentCrn": "crn:cdp:environments:us-west-1:12a0079b-1591-dd33-b
721-a446bda74e67:environment:36853fcc-2fef-4094-834c-557b4aea34ee",
    "environmentName": "finance-reporting-set4",
    "status": "AVAILABLE",
    "instances": {
        "i-078ba50f9feb6638f": {
            "hostname": "ipaserver1.datalake.xcu2-8y8x.cloudera.site",
            "status": "CREATED",
            "issues": []
        },
        "i-09e8b54a343b33d2": {
            "hostname": "ipaserver0.datalake.xcu2-8y8x.cloudera.site",
            "status": "CREATED",
            "issues": []
        }
    }
```

Element	Data Type	Description
environmentCrn	string	The CRN of the environment.
environmentName	string	The name of the environment.
status	string	The overall status of the FreeIPA cluster. The status values retrieved can reflect an earlier status form the cloud provider. For example, AWS may show the host as "Running," but the CDP CLI status command may show the status as "UNREACHABLE." When this occurs, it is a temporary status. Calling the CDP CLI status command again; eventually the status will change from "UNREACHABLE" to "CREATED," which is the Cloudera equivalent to AWS "Running." A full list of status values can be found in “FreeIPA instance status options”.
instances	map	Status of individual nodes in the FreeIPA cluster.
key	string	Each entry includes the cloud-provider ID for the host. The corresponding value includes the status and issues of an individual FreeIPA node.
hostname	string	The hostname of the IPA instance.
issues	array	A list of issues the node is having. If there are no issues, the content is a blank string.

Related Information

Repair FreeIPA

FreeIPA instance status options

Installing the Cloudera client

FreeIPA instance status options

A full list of FreeIPA instance status options for UI and CLI. This information is displayed in the environment's Summary tab in the Cloudera Management Console, or via the cdp environments get-freeipa-status command.

Scenario	UI Status	CLI Overall Status	CLI Instance Status
Provisioning	Blue "Requested", then blue "Create in progress"	REQUESTED then CREATE_IN_PROGRESS then STACK_AVAILABLE then CREATE_IN_PROGRESS	CREATE_IN_PROGRESS
Provisioning failed	Red "Create failed"	CREATE_FAILED	
Running and all nodes have good health	Green "Running"	AVAILABLE	CREATED
Repairing	Blue "Update requested", then blue "Update in progress"	UPDATE_REQUESTED then UPDATE_IN_PROGRESS	At the end the new instance is CREATED and the old instance is TERMINATED
Repair Failed	Orange "Unhealthy"	UNHEALTHY	
Starting	Blue "Start in progress"	START_IN_PROGRESS	
Start Failed	Red "Start failed"	START_FAILED	
Stopping	Blue "Stop requested", then blue "Stop in progress"	STOP_REQUESTED then STOP_IN_PROGRESS	
Stop Failed	Red "Stop Failed"	STOP_FAILED	
Stopped	Gray "Stopped"	STOPPED	STOPPED
Deleted	Red "Terminated"	DELETE_COMPLETED	TERMINATED
Delete Failed	Red "Termination failed"	DELETE_FAILED	
Deleting	Red "Terminating"	DELETE_IN_PROGRESS	
Instance deleted on cloud provider’s side	Orange "Unhealthy"	UNHEALTHY	DELETED_ON_PROVIDER_SIDE
All nodes deleted	Red "Deleted on provider side"	DELETED_ON_PROVIDER_SIDE	DELETED_ON_PROVIDER_SIDE
Network connectivity lost to some nodes but not	Orange "Unhealthy"	UNHEALTHY	UNREACHABLE
Network connectivity lost to all nodes	Orange "Unreachable"	UNREACHABLE	UNREACHABLE
Some nodes stopped on cloud provider side but not all	Orange "Unhealthy"	UNHEALTHY	STOPPED
All nodes stopped on cloud provider side	Gray "Stopped"	STOPPED	STOPPED
Some nodes with running with health check failures	Orange "Unhealthy"	UNHEALTHY	Bad nodes are UNHEALTHY
All nodes unhealthy	Orange "Unhealthy"	UNHEALTHY	UNHEALTHY

Vertically scale FreeIPA instances

If necessary, you can select a larger or smaller instance type for FreeIPA after the environment has been created.

About this task

Selecting a larger instance type adds more vCPU and/or RAM to your instances. Instances can be scaled both up and down, but scaling down to a smaller size requires 4 CPU and a minimum of 4 GB memory.

If you are using an instance without ephemeral disks, you can scale up or down to a new instance with ephemeral disks; however, the reverse is not supported. You cannot start with an instance with ephemeral disks and move to an instance without ephemeral disks.

Vertical scaling is supported on AWS. It is also supported on Azure by entitlement. Contact Cloudera customer support for entitlements.

FreeIPA instances must be stopped before scaling. See Change the instance type in AWS documentation for more information.

Procedure

1. In the Cloudera main navigation menu, click Environments and select the environment that requires a larger FreeIPA instance type.
2. Scroll to the bottom of the page and click the **FreeIPA** tab.
3. Click Vertical Scaling.

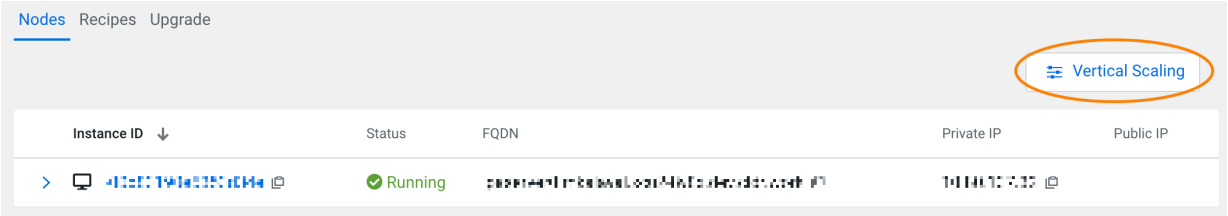

4. From the drop-down menu of available instance types, select the instance that you want to scale to.

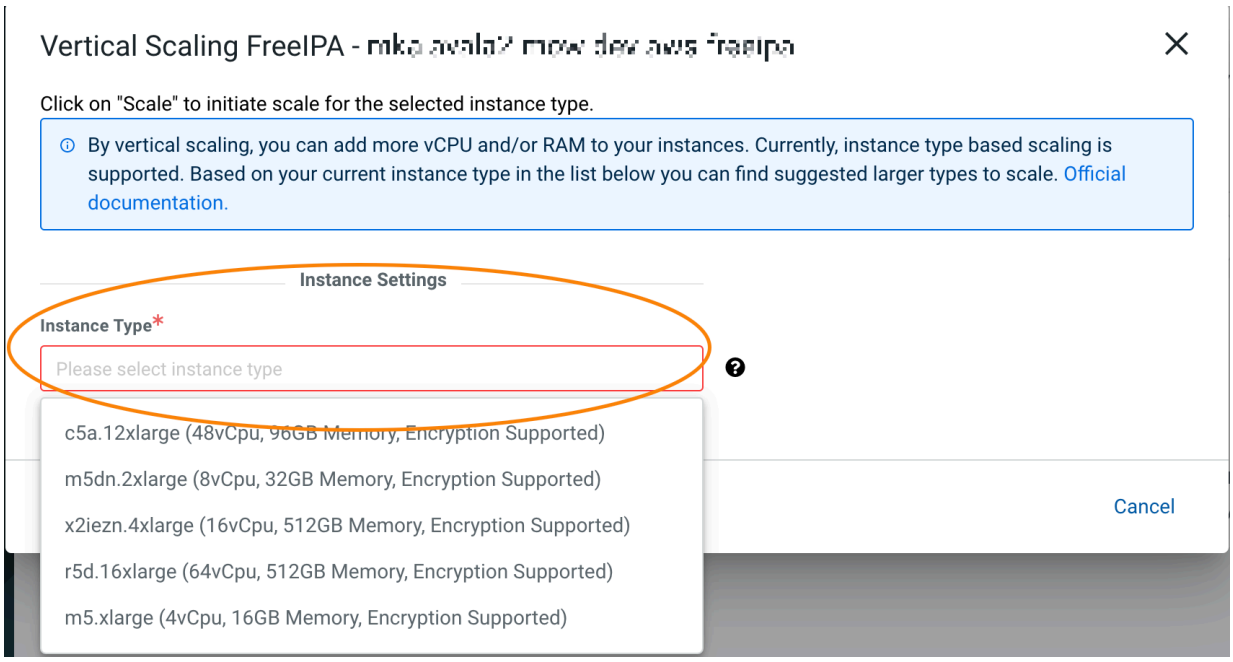

5. Click Scale. You can monitor the action from the **Event History** tab.

   Alternatively, you can use the CDP CLI to select a new instance for FreeIPA:

```
cdp environments start-free-ipa-vertical-scaling
--environment <your-environment-name-or-its-crn>
--instance-template instanceType="<m5.4xlarge>"
```

Upgrade FreeIPA

To ensure that your FreeIPA nodes are running with the latest patches, you should periodically upgrade your FreeIPA cluster.

FreeIPA is the core component of an environment that runs on one or more instances.

To ensure that your FreeIPA nodes are running with the latest patches, you should periodically upgrade your FreeIPA cluster. Cloudera currently allows you to upgrade all FreeIPA clusters, updating OS-level security patches on the cluster nodes. The upgrade process includes launching entirely new instances with the new OS image.

The upgrade process is a rolling upgrade consisting of the following steps:

1. A new instance is provisioned from a newer image.
2. Cloudera validates that the new node is reachable, all services are up, and that replication works.
3. The old instance is removed.
4. If there is more than one instance in the cluster, the previous 3 steps are repeated for the second instance only after the first instance has been successfully upgraded.

Your environment remains functional during the upgrade process, with an exception of a short (one minute or less) downtime. During and after the FreeIPA upgrade, the status of the environment changes to **Update Started** as the environment is refreshing all the clusters to the new FreeIPA configurations (IP addresses for DNS resolution). Refreshing the clusters in the environment can take some time. You can minimize the time of the FreeIPA upgrade by not initiating any cluster related activities such as scaling and provisioning.

During FreeIPA provisioning a full backup is performed and there are periodic incremental backups afterwards. These backups can be used in case a recovery is needed.

Note:

The rollback option is not supported for FreeIPA upgrade.

Use either the Cloudera web interface or CDP CLI to initiate an upgrade. The upgrade process takes about one hour. The duration varies depending on the cloud provider and the number of nodes.

Performing a FreeIPA upgrade

You can trigger the FreeIPA upgrade via Cloudera web interface or CDP CLI.

Prerequisites

- If the upgrade involves upgrading from CentOS to RHEL, review the Prerequisites for upgrading from CentOS to RHEL.
- On Azure, before you run the FreeIPA upgrade, make sure that the resource group has neither a DELETE nor a READ-ONLY lock applied.
- Before performing the FreeIPA upgrade for Cloudera Data Hub clusters where autoscaling is enabled, you must disable autoscaling and start all compute nodes to ensure the healthy state of Cloudera Data Hub clusters. After upgrading FreeIPA, autoscaling can be enabled again. For more information, see the Autoscaling must be stopped before performing FreeIPA upgrade description in the Known issue.
- Before performing the FreeIPA upgrade, if you have any Cloudera Data Hub clusters where autoscaling is enabled, then you must disable autoscaling and start all compute nodes to ensure the healthy state of Cloudera Data Hub clusters.

Required roles: EnvironmentAdmin or Owner of the environment

Steps

For CDP UI

1. Log in to the Cloudera web interface.

2. From the navigation pane, select Environments.
3. Click on your environment.
4. Click on the FreeIPA Upgrade tab.
5. From the Target image dropdown, select a target image. If you would like to upgrade from CentOS to RHEL, ensure that you select a RHEL image. Once you have reviewed the information related to your current base image and the target base image that will be used for provisioning new instances during the upgrade process. Once you've reviewed the information, click on Upgrade:

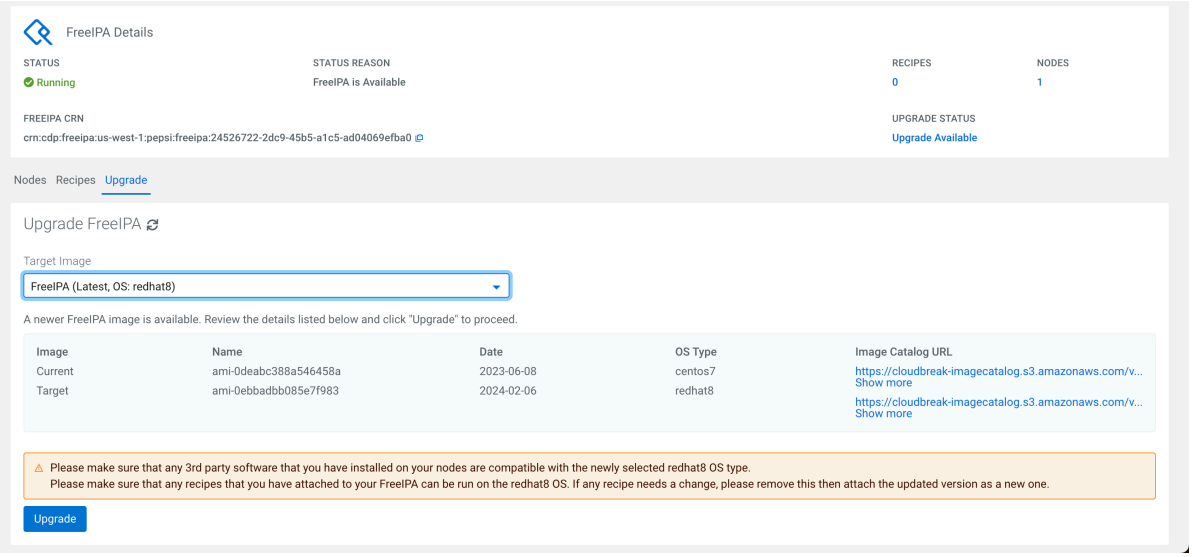

6. The upgrade process starts. During the upgrade, FreeIPA status switches to **Update in progress**, but the environment remains available. A new instance is provisioned, then the old instance is removed. If more than one instance is present, the process is repeated for the additional instances.
7. Once the upgrade is completed, the FreeIPA status changes to **Running**.

For CDP CLI

Use the following command to trigger FreeIPA upgrade (including a CentOS to RHEL upgrade):

```
cdp environments upgrade-freeipa --environment-name <ENVIRONMENT_NAME_OR
_CRN>
```

For example:

```
cdp environments upgrade-freeipa --environment-name my-env
```

The command will return a JSON including target and original image information and the operation ID.

```
{
    "targetImage": {
        "catalog": "https://gist.githubusercontent.com/lacikaaa/c096c999a
de874a60bf15be897220eed/raw/e30ed12b14c77175b82ceb9caf26500ab1340f26/fre
eipa-test-catalog.json",
        "id": "9c1c8959-86a7-4b7d-af5a-be252f8b395d",
        "os": "centos7",
        "imageName": "ami-0dfafedeed3a4474f",
        "date": "2021-06-10"
    },
    "originalImage": {
        "catalog": "https://gist.githubusercontent.com/lacikaaa/c096c99
9ade874a60bf15be897220eed/raw/e30ed12b14c77175b82ceb9caf26500ab1340f26/f
reeipa-test-catalog.json",
        "id": "0b73e149-6e22-4667-acc5-1d0ba82f8245",
        "os": "centos7",
```

```
        "imageName": "ami-030e1d907cfa5ca33",
        "date": "2021-06-04"
    },
    "operationId": "28432209-a4ff-4aea-a8ec-4604d67cdb7c"
}
```

Use the following command to track the progress of the upgrade process:

```
cdp environments get-repair-freeipa-status --operation-id <OPERATION-ID-FROM-UPGRADE-FREEIPA-COMMAND-OUTPUT>
```

The operation ID can be obtained from the output of the upgrade-freeipa command.

What to do next

After upgrading FreeIPA, autoscaling can be enabled again. For more information, see the Autoscaling must be stopped before performing FreeIPA upgrade description in the Known issue.

In rare cases, the FreeIPA upgrade process might fail. In such cases, you should trigger a retry of the FreeIPA upgrade.

Retry a FreeIPA upgrade

In rare cases, the FreeIPA upgrade process might fail. In such cases, you should trigger a retry of the FreeIPA upgrade.

Steps

For CDP UI

If you are performing the upgrade via Cloudera web interface, you can find the retry option on the same **FreeIPA** tab where the upgrade option is located:

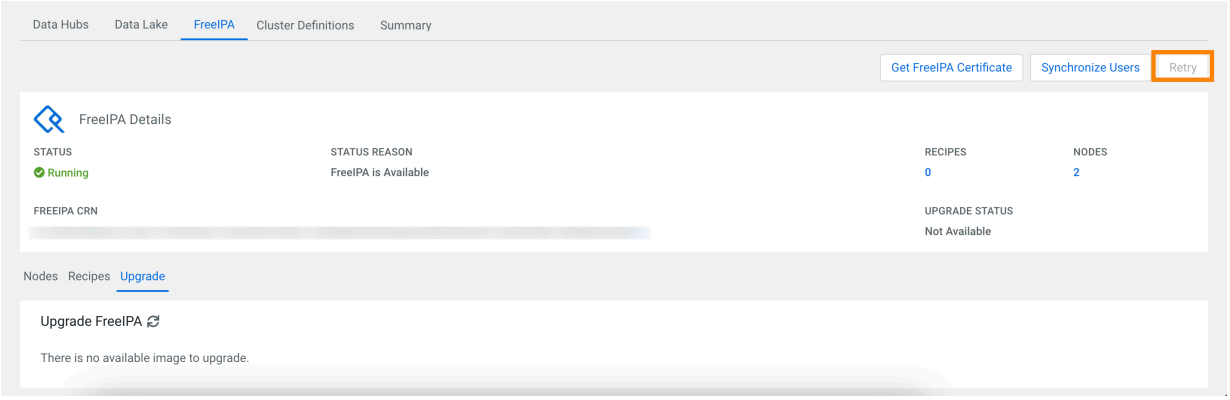

The Retry option is grayed out if there is no reason to trigger the repair and is only enabled when FreeIPA upgrade encounters problems.

For CDP CLI

If you would like to trigger the repair via CDP CLI, use the following command:

```
cdp environments retry-freeipa –environment <MY-ENVIRONMENT-NAME-OR-CRN>
```

The command returns an operation ID that you can pass in the following command to track the status of the repair:

```
cdp environments get-repair-freeipa-status --operation-id <OPERATION-ID-FROM-UPGRADE-FREEIPA-COMMAND-OUTPUT>
```

Troubleshooting

Issue	Workaround
After performing a FreeIPA upgrade, Cloudera Data Warehouse VMs are broken due to the new FreeIPA server instances' IP addresses not being reflected in Cloudera Data Warehouse.	Manually restart every cdp-coredns-updater pod running in your Cloudera Data Warehouse Kubernetes cluster namespace.
After performing a FreeIPA upgrade, Cloudera AI workbenches are broken because the IP addresses of the new FreeIPA server instances are not reflected in Cloudera AI.	Restart the cdp-coredns-updater pod by running kubectl delete -n kube-system $(kubectl get pods -n kube-system -l app.kubernetes.io/name=cdp-coredns-updater --no-headers -o name) against your Cloudera AI kubernetes cluster as soon as the FreeIPA upgrade has completed.

Related Information

Repair FreeIPA

Repair FreeIPA

When running in high-availability mode, the identity management system runs multiple instances of FreeIPA on separate hosts. In case of failure, you can repair failed hosts using the CDP CLI within one week of a node failing.

The repair process for FreeIPA hosts performs the following steps, stopping when a healthy status returns:

- Start any FreeIPA hosts that are stopped.
- Reboot FreeIPA hosts. (same physical host, same public DNS name, private IP address, and associated storage).
- Restore identity management data from backup.
- Stop and restart FreeIPA hosts (rebuild using new hardware).

This procedure uses the CDP CLI. If you haven't already installed the CLI, see *Installing the Cloudera client* for instructions.

On Azure, before you run the repair command, make sure that the resource group has neither a DELETE nor a READ-ONLY lock applied.

Steps

Run the FreeIPA repair command. Run this command from a computer that has network access to the FreeIPA hosts.

```
cdp environments repair-freeipa --environment-name <value>
                  [--force | --no-force]
                  [--instances <value>]
                  [--cli-input-json <value>]
                  [--generate-cli-skeleton]
                  [--repair-type <string>]
```

where the options are the following:

Option	Description
--environment-name *<VALUE>*	Specifies the FreeIPA's environment name or CRN. The environment CRN is listed in Environment Summary General .
--force | --no-force	Choose to force the repair even if the status if the FreeIPA nodes are good. If not specified, defaults to no-force.
--instances *<VALUE>*	Specifies the instance IDs to repair. Use a space to separate multiple instance IDs. If no IDs are provided then all instances are considered for repair. The FreeIPA instance IDs are listed in Environment Summary FreeIPA . You can get the IDs from the output of the cdp environments get-freeipa-status command.
--cli-input-json *<VALUE>*	Performs the operations indicated in the command provided in JSON format. Call generate-cli-skeleton to see a template of the required JSON content. If other arguments are provided on the command line, the command line values override the JSON-provided values.

Option	Description
--generated-cli-skeleton	Displays an example of the format of the input JSON. The repair command does not run.
--repair-type <string>	The type of FreeIPA repair to perform. Possible values: • AUTO - Currently, this is the same as reboot but this may change in the future. • REBOOT - Repair the failed instances by rebooting them. • REBUILD - Repair the failed instances by deleting them and creating new instances, then replicate data from an existing instance to the new instances. Needs powerUser role.

For example, the following command repairs two instances without forcing the repair:

```
$ cdp environment repair-freeipa --environment-name crn:cdp:environments:us-
west-1:12a0079b-1591-dd33-b721-a446bda74e67:environment:36853fcc-2fef-4094-8
34c-557b4aea34ee --instances i-078ba50f9feb6638f i-09e8b54a343b33d2
```

```
cdp environments repair-freeipa --environment-name john-doe-env2-25793 --ins
tances i-0288d991ed998ec03 --force
{
    "operationId": "edda2c68-5a29-4f60-a150-aca963b36ead",
    "status": "RUNNING",
    "successfulOperationDetails": [],
    "failureOperationDetails": [],
    "startDate": "2020-10-01T19:48:36.009000+00:00"
}
```

If you see the following error, consider rerunning the repair command with the --force option:

```
An error occurred: {"message":"No unhealthy instances to reboot. Maybe use t
he force option."} (Status Code: 404; Error Code: NOT_FOUND; Service: enviro
nments; Operation: repairFreeipa; Request ID: eca3a7fb-aa6b-48f7-ae29-b88116
1869e5;)
```

Related Information

Installing the Cloudera client

Check FreeIPA repair status

You can check the status of an in-progress repair operation with get-repair-freeipa-status.

This procedure uses the CDP CLI. If you haven't already installed the CLI, see *Installing the Cloudera client* for instructions.

Steps

Run the FreeIPA status command. Run this command from a computer that has network access to the FreeIPA hosts.

```
cdp environments get-repair-freeipa-status --operation-id <value>
          [--cli-input-json <value>]
          [--generate-cli-skeleton]
```

where the options are the following:

Option	Description
--operation-id <*VALUE*>	Operation-id for the previously requested repair operation.
--cli-input-json <*VALUE*>	Performs service operation based on the JSON string provided. The JSON string follows the format provided by --generate-cli-skeleton. If other arguments are provided on the command line, the CLI values will override the JSON-provided values.

Option	Description
--generate-cli-skeleton	Prints a sample input JSON to standard output. Note the specified operation is not run if this argument is specified. The sample input can be used as an argument for --cli-input-json.

This command can take 15 to 45 seconds to run as it gathers information in real-time. The output of the status command provides the status for the repair operation (in JSON format).

```
cdp environments get-repair-freeipa-status --operation-id edda2c68-5a29-4f60
-a150-aca963b36ead
{
    "status": "COMPLETED",
    "successfulOperationDetails": [
        {
            "environmentCrn": "crn:cdp:environments:us-west-1:9d74eee4-1ca
d-45d7-b645-7ccf9edbb73d:environment:08c55413-6e2b-4664-8367-ef3fc0787773"
        }
    ],
    "failureOperationDetails": [],
    "startDate": "2020-10-01T19:48:36.009000+00:00",
    "endDate": "2020-10-01T19:49:08.392000+00:00"
    }
```

Element	Data Type	Description
status	string	Status of a repair operation. Possible values: REQUESTED, RUNNING, COMPLETED, FAILED, REJECTED, TIMEDOUT.
successfulOperationDetails	array	List of operation details for all successes. If the repair is only partially successful both successful and failure operation details will be populated.
failureOperationDetails	array	List of operation details for failures. If the repair is only partially successful both successful and failure operation details will be populated.
error	string	If there is any error associated. The error will be populated on any error and it may be populated when the operation failure details are empty. The error will typically contain the high level information such as the associated repair failure phase.
startDate	datetime	Date when the operation started.
endDate	datetime	Date when the operation ended. Omitted if operation has not ended.

Resize FreeIPA

After resizing your Data Lake from Light Duty to Medium Duty you should also resize your FreeIPA cluster. You can do this via CDP CLI.

When you register a Cloudera environment, the FreeIPA node count (also known as “availability type”) depends on the Data Lake cluster shape:

- When an environment with a Medium Duty Data Lake is provisioned, an HA FreeIPA cluster consisting of three instances is created along with it.
- When an environment with a Light Duty Data Lake is provisioned, a two-node FreeIPA cluster is created along with it.

When you resize (or “migrate”) your Data Lake from Light Duty to Medium Duty, you should resize your FreeIPA cluster accordingly from two to three nodes so that a Medium Duty Data Lake can be backed by a FreeIPA server that offers higher availability.

In general, the following FreeIPA resizing scenarios are supported:

- Upscaling from 1 to 2 nodes

- Upscaling from 1 to 3 nodes
- Upscaling from 2 to 3 nodes
- Downscaling from 3 to 2 nodes
- Downscaling from 4 to 3 nodes. During a failed upgrade, it is possible to have 4 instances. This safety option covers these scenarios.

The following validations are performed before the scaling operation:

- Scale is possible given the current node count.
- The instances are not deleted and all the instances are available.
- FreeIPA stack is available.
- The scaling path is supported by Cloudera Management Console

Scale up FreeIPA

To upscale your FreeIPA cluster, use the following command:

```
cdp environments upscale-freeipa \
  --environment-name <ENVIRONENT-NAME> \
  --target-availability-type HA
```

For example:

```
cdp environments upscale-freeipa \
  --environment-name my-env \
  --target-availability-type HA
```

This operation requires the --target-availability-type parameter, with the following possible values:

- HA - 3-node HA cluster
- TWO_NODE_BASED - 2-node cluster

Scale down FreeIPA

To downscale your FreeIPA cluster, use the following command:

```
cdp environments downscale-freeipa \
  --environment-name <ENVIRONENT-NAME> \
  --target-availability-type <AVAILABILITY-TYPE>
```

For example:

```
cdp environments downscale-freeipa \
  --environment-name my-env \
  --target-availability-type HA
```

This operation requires the --target-availability-type parameter, with the following possible values:

- HA - 3-node HA cluster
- TWO_NODE_BASED - 2-node cluster

Configuring workload password policies

In order to bring your workload password complexity requirements in line with company policy, you can set your FreeIPA password policies via Cloudera web interface and CDP CLI. Password policies can be configured for length, complexity, expiration, and scope.

Note: Configuring password policies takes effect in all environments within a tenant, but applies to newly set passwords only. As such, admins should advise users to reset their passwords to achieve compliance with their new password policy.

Note: Prior to introducing this feature, Cloudera web UI had stricter password complexity requirements than the CLI. With the release of the password policy feature, both the UI and CLI enforce the same stronger password policy by default.

Warning: There is currently no stable notification system in place that would inform users that their password expired. When users SSH to a node and their password has expired, they may be prompted to reset their password in the SSH session. As resetting the password in the SSH session may only work for a short period, the users should instead set a new workload password using the Cloudera Management Console. For instructions, see Setting the workload password.

Workload password policy types

There are two types of password policies:

- Global policies - Apply to all users including machine users
- Machine user policies - Apply to machine users only

You can set either or both policies. By default, global policies are applied to all users, including machine users. An optional override for configuring a different policy for machine users is available. For example, setting strict password expiration policies for machine users may not be desired, as password expiration in those accounts may cause upstream failures in the applications that use them.

Default workload password policy

If a password policy has not been set, the following default password policy is used:

- A minimum password length of 8 characters
- Must include at least 1 upper case character, lowercase character, number and special character. Supported special characters are: "#", "&", "*", "$", "%", "@", "^", ".", "_", and "!".
- All previous passwords can be reused
- The password can be changed at any time
- The password never expires

For detailed information on how to manage workload password policies, refer to the following documentation:

Check the current workload password policy

You can check your current password policy from the Workload Password Policies page or using the cdp iam get-account CLI command.

Required role: PowerUser

Steps

For CDP UI

You can check the current password policy from the Workload Password Policies page. To access this page:

1. Log in to the Cloudera web interface.

2. Navigate to the Cloudera Management Console User Management Workload Password Policies :

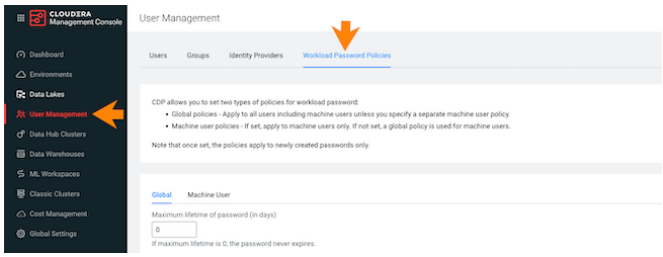

For CDP CLI

Use the cdp iam get-account to obtain your current password policy.

Set a password policy

You can set password policies from the Workload Password Policies page or using the via CDP CLI using the cdp iam set-workload-password-policy command.

Note: Prior to introducing this feature, Cloudera web UI had stricter password complexity requirements than the CLI. With the release of the password policy feature, both the UI and CLI enforce the same stronger password policy by default.

Required role: PowerUser

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console User Management Workload Password Policies :

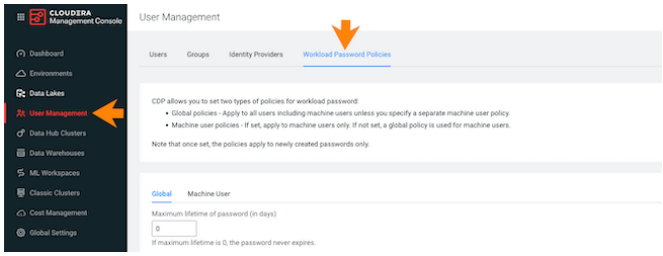

3. In the Global tab, specify a policy that applies to all Cloudera users and machine users. The following options are available:

UI option	Description
Minimum lifetime of password (in days)	Once set, the password must remain the same for this period of time. Note: If a user forgets their password, they will be unable to reset it until the minimum period has passed or until the PowerUser unsets the minimum lifetime of the password for them.
Maximum lifetime of password (in days)	Allows you to specify password expiration period in days.
Minimum password length	Allows you to specify minimum password length, must be between 6 and 256 characters.

Number of previous passwords that can't be reused	If set to 0, all previous passwords can be reused. Any number larger than 0 indicates the number of most recent passwords that can't be reused. The maximum allowed value for this parameter is 20, so you can prevent users from reusing up to 20 recent passwords. Note: Password history information is only recorded when password history size is set to a value other than zero. This means that when the password history size is initially set from zero to non-zero, the previous passwords that were set (while the password history size was as at 0) are not considered when the password history check is done.
Must include uppercase characters	When checked, at least one uppercase character is required
Must include lowercase characters	When checked, at least one lowercase character is required
Must include numbers	When checked, at least one number is required
Must include symbols	When checked, at least one special character is required. Supported special characters are: "#", "&", "*", "$", "%", "@", "^", ".", "_", and "!".

4. Click Update.
5. By default, global policies are applied to machine users. If you would like to set a different policy for machine users:
   a. Navigate to the Machine User tab.
   b. Uncheck Inherit from global policy.
   c. Set a desired policy (the available options are the same as for global policy).
   d. Click Update.

For CDP CLI

The following example creates a global policy:

```
cdp iam set-workload-password-policy --global-password-policy minPasswor
dLength=8,mustIncludeUpperCaseCharacters=true,mustIncludeLowerCaseCharac
ters=true,mustIncludeNumbers=true,mustIncludeSymbols=false
```

The following example creates a machine user policy. This overrides the global policy for machine users:

```
cdp iam set-workload-password-policy --machine-users-password-policy min
PasswordLength=8,mustIncludeUpperCaseCharacters=true,mustIncludeLowerCas
eCharacters=true,mustIncludeNumbers=true,mustIncludeSymbols=true
```

The following password complexity requirements can be set as part of your policy:

CLI option	Type	Description	Default value
minPasswordLifetimeDays	integer	Minimum period in days during which password cannot be changed once it is set Note: If a user forgets their password, they will be unable to reset it until the minimum period has passed or until the PowerUser unsets the minimum lifetime of the password for them.	0
maxPasswordLifetimeDays	integer	Expiration period in days	0
minPasswordLength	integer	Minimum password length; must be between 6 and 256 characters.	8

passwordHistorySize	integer	Number of previous passwords that can't be reused. 0 indicates that all previous passwords can be reused. Any number above 0 indicates the number of most recent passwords that can't be reused. The maximum allowed value for this parameter is 20, so you can prevent users from reusing up to 20 recent passwords. Note: Password history information is only recorded when password history size is set to a value other than zero. This means that when the password history size is initially set from zero to non-zero, the previous passwords that were set (while the password history size was as at 0) are not considered when the password history check is done.	0
mustIncludeUpperCaseCharacters	true|false	At least one uppercase character is required	true
mustIncludeLowerCaseCharacters	true|false	At least one lowercase character is required	true
mustIncludeNumbers	true|false	At least one number is required	true
mustIncludeSymbols	true|false	At least one special character is required. Supported special characters are: "#", "&", "*", "$", "%", "@", "^", ".", "_", and "!".	true

Reset a password policy

You can reset password policies via CDP CLI using the cdp iam unset-workload-password-policy command. As a result, default password policies will be reinstated.

A global password policy is always present for an account. When the global password policy is unset, the policy will revert to the documented defaults. A machine user password policy may or may not be present in the account. When the machine user policy is not set for the account, the global password policy will be enforced for machine users.

Required role: PowerUser

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console User Management Workload Password Policies :

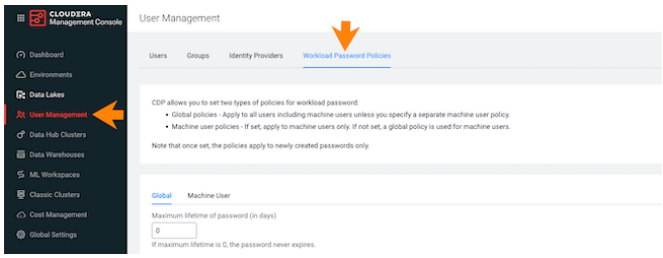

3. Navigate to the Machine User tab and make sure that Inherit from global policy is checked.

   Note: If this option remains unchecked, machine user policies will not be reset.

4. Click on Reset to default values.
5. Click OK to confirm.

For CDP CLI

Use the following commands to reset password policies:

```
cdp iam unset-workload-password-policy --unset-global-password-policy
```

```
cdp iam unset-workload-password-policy --unset-machine-users-password-po
licy
```

Unset user's minimum password lifetime

With the minimum workload password lifetime enabled, it may happen in very rare cases that a user becomes locked out of their account and their minimum password lifetime needs to be temporarily unset.

If your organization's workload password policy has the minimum password lifetime enabled, a user is unable to set a new workload password until the minimum password lifetime duration has passed. Consequently, a user who forgets their password will be unable to set a new password until the minimum password lifetime duration has passed. In such a case, a PowerUser can unset the minimum lifetime date for the user who has been locked out of their account. This is a one-time override that will allow the user to set a new workload password.

Required role: PowerUser

Steps

Use the following command to unset the minimum password lifetime for a specific user:

```
cdp iam unset-workload-password-min-lifetime --user <USER-CRN-OR-ID>
```

This is a one-time override. Once the user sets their password, the minimum password lifetime will be reset to the original value that you set for your organization.

Accessing a FreeIPA cluster via SSH

If you plan to access a FreeIPA cluster (for example for troubleshooting purposes) via a command line client, SSH into the master node.

Non-root SSH access to a FreeIPA node

Required role: EnvironmentUser, Data Steward, or EnvironmentAdmin

A user who has the required Cloudera role assigned to them can SSH to FreeIPA cluster nodes using their own Cloudera user.

To execute privileged operating system actions on a Cloudera Data Hub, the EnvironmentPrivilegedUser role is required.

For SSH access through a workload user name and password:

When accessing Cloudera for the first time, you must set a workload password. The password also needs to be reset each time you are added to a new environment.

For more information about workload passwords and instructions for setting/resetting it, refer to Set or Reset Workload Password.

For instructions on how to find your workload user name, refer to Finding Your Workload User Name.

On Mac OS, you can use the following syntax to SSH to the VM:

```
$ ssh <workloaduser>@<nodeIPaddress>
```

For example:

```
$ ssh jsmith@190.101.0.132
```

To SSH to a cluster using the private key file that pairs with the public key associated with a user, use the ssh utility:

```
$ ssh -i <path-to-private-key-file> <cdpusername>@<nodeIPaddress>
```

For example:

```
$ ssh -i ~/.ssh/my-private-key jsmith@192.12.141.12
```

On Windows, you can access your cluster via SSH by using an SSH client such as PuTTY. For more information, refer to How to use PuTTY on Windows.

Root SSH access to a FreeIPA node

Required role: No Cloudera role is required

Cloudera administrators can access FeeeIPA cluster nodes as cloudbreak user with the SSH key provided during environment creation.

On Mac OS, you can use the following syntax to SSH to the VM:

```
ssh -i <path-to-cloudbreak-private-key-file> cloudbreak@<nodeIPaddress>
```

For example:

```
ssh -i ~/.ssh/cloudbreak-private-key cloudbreak@90.101.0.132
```

On Windows, you can access your cluster via SSH by using an SSH client such as PuTTY. For more information, refer to How to use PuTTY on Windows.

Recipes

A recipe is a script that runs on all nodes of a selected host group at a specific time. You can use recipes to create and run scripts that perform specific tasks on your Cloudera Data Hub, Data Lake, or FreeIPA cluster nodes.

You can use recipes for tasks such as installing additional software or performing advanced cluster configuration. For example, you can use a recipe to put a JAR file on the Hadoop classpath.

Recipes can be uploaded and managed via the Cloudera web interface or CLI and then selected, when needed, for a specific cluster and for a specific host group. If selected, they will be executed on a specific host group at a specified time.

Depending on the need, a recipe can be executed at various times. Available recipe execution times are:

- Before Cloudera Manager server start

- After Cloudera Manager server start
- After cluster installation
- Before cluster termination

Recipes are stored on the Cloudera Manager server for the lifetime of the master node, and are executed at specific times of your choosing:

- pre-service-deployment (formerly pre-cluster-manager-start): During a Cloudera Data Hub, Data Lake, or environment deployment, the script will be executed on every node before the Cloudera Manager server starts, and after node repair or OS upgrade of a cluster.
- post-cluster-manager-start: During a Cloudera Data Hub or Data Lake deployment, the script will be executed on every node after the Cloudera Manager server starts, but before cluster installation. post-cluster-manager start recipes are also executed after node repair or OS upgrade of a cluster. This option is not available for FreeIPA recipes.
- post-service-deployment (formerly post-cluster-install): The script will be executed on every node after cluster installation on the Cloudera Manager server is finished, and after node repair or OS upgrade of a cluster.
- pre-termination: The script will be executed on every node before cluster termination.

Note: On the master node, recipes are triggered when the Cloudera Manager server starts; on other nodes, recipes are triggered when the Cloudera Manager agent starts.

Writing recipes

Refer to these guidelines when creating your recipes.

When using recipes, consider the following guidelines:

- Running bash and python scripts as recipes is supported. We recommend using scripts with Shebang character sequence, for example:

```
#!/bin/sh
#!/bin/bash
#!/usr/bin/sh
#!/usr/bin/bash
#!/usr/bin/env sh
#!/usr/bin/env bash
#!/bin/sh -x
#!/usr/bin/python
#!/usr/bin/env python
```

- The scripts are executed as root. The recipe output is written to /var/log/recipes on each node on which it was executed.
- Supported parameters can be specified as variables by using mustache kind of templating with "{{{ }}}" syntax. Once specified in a recipe, these variables are dynamically replaced when the recipe is executed, generating the actual values that you provided as part of cluster creation process. For the list of parameters, refer to Recipe and cluster template parameters. For an example, see Example: Recipe using parameters.

  Note: Using variable parameters is not supported for FreeIPA recipes.

  For example, if your cluster includes an external LDAP and your recipe includes {{{ldap.connectionURL}}}, as demonstrated in the following example

```
#!/bin/bash -e

main() {
 ping {{{ ldap.connectionURL }}}
}
```

```
[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

then, when this recipe runs, the {{{ldap.connectionURL}}} is replaced with the actual connection URL specified as part of cluster creation process, as demonstrated in the following example:

```
#!/bin/bash -e

main() {
 ping 192.168.59.103
}
[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

- Recipe logs can be found at /var/log/recipes/${RECIPE_TYPE}/${RECIPE_NAME}.log
- The scripts are executed on all nodes of the host groups that you select (such as “master”, “worker”, “compute”).
- In order to be executed, your script must be in a network location which is accessible from the Cloudera Management Console and the virtual network in which your cluster is located.
- Make sure to follow Linux best practices when creating your scripts. For example, don’t forget to script “Yes” auto-answers where needed.
- Do not execute yum update –y as it may update other components on the instances (such as salt) – which can create unintended or unstable behavior.

Example Python script

```
#!/usr/bin/python
print("An example of a python script")
import sys
print(sys.version_info)
```

Example bash script for yum proxy settings

```
#!/bin/bash
cat >> /etc/yum.conf
<<ENDOF
proxy=http://10.0.0.133:3128
ENDOF
```

Example recipe including variables

Original recipe:

```
#!/bin/bash -e

function setupAtlasServer() {
    curl -iv -u {{{ general.userName }}}:{{{ general.password }}} -H "X-Re
quested-By: ambari" -X POST -d '{"RequestInfo":{"command":"RESTART","context
":"Restart all components required ATLAS","operation_level":{"level":"SERVIC
E","cluster_name":"{{{ general.clusterName }}}","service_name":"ATLAS"}},"Re
quests/resource_filters":[{"hosts_predicate":"HostRoles/stale_configs=false&
HostRoles/cluster_name={{{ general.clusterName }}}"}]}' http://$(hostname -f
):8080/api/v1/clusters/{{{ general.clusterName }}}/requests
}

main() {
    setupAtlasServer
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Generated recipe (to illustrate how the variables from the original recipe were replaced during cluster creation):

```
#!/bin/bash -e

function setupAtlasServer() {
    curl -iv -u admin:admin123 -H "X-Requested-By: ambari" -X POST -d '{"R
equestInfo":{"command":"RESTART","context":"Restart all components required
ATLAS","operation_level":{"level":"SERVICE","cluster_name":"super-cluster","
service_name":"ATLAS"}},"Requests/resource_filters":[{"hosts_predicate":"Hos
tRoles/stale_configs=false&HostRoles/cluster_name=super-cluster"}]}' http://
$(hostname -f):8080/api/v1/clusters/super-cluster/requests
}

main() {
    setupAtlasServer
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Register a recipe

In order to use your recipe for clusters, you must first register it with the Cloudera Management Console.

About this task

Required role: EnvironmentCreator can create a shared resource and then assign users to it.

SharedResourceUser or Owner of the shared resource can use the resource.

Before you begin

If you are using Cloudera with a proxy, note that the Cloudera proxy settings do not apply to cluster recipes. If you planning to use the recipes, then you can set the proxy settings manually. You can find the proxy settings in the /etc/cdp/proxy.env file.

Procedure

1. Place your script in a network location accessible from Management Console and from the virtual network in which your clusters are located.
2. Log in to the Cloudera web interface.
3. Navigate to Shared ResourcesRecipes and click Register Recipe.
4. Provide the following:

Parameter	Value
Name	Enter a name for your recipe.
Description	(Optional) Enter a description for your recipe.

Parameter	Value
Execution Type	Select one of the following options: • pre-service-deployment (formerly pre-cluster-manager-start): During a Cloudera Data Hub, Data Lake, or environment deployment, the script will be executed on every node (in the host group where you assigned the recipe) before the Cloudera Manager server starts. • post-cluster-manager-start: During a Cloudera Data Hub or Data Lake deployment, the script will be executed on every node (in the host group where you assigned the recipe) after the Cloudera Manager server starts, but before cluster installation. This option is not available for FreeIPA recipes. • post-service-deployment (formerly post-cluster-install):: The script will be executed on every node (in the host group where you assigned the recipe) after cluster installation on the Cloudera Manager server is finished. • pre-termination: The script will be executed on every node (in the host group where you assigned the recipe) before cluster termination.
Script	Select one of: • File: Point to a file on your machine that contains the recipe. • Text: Paste the script.

5. Click Register.

What to do next

- When you create a Cloudera Data Hub cluster, you can select a previously added recipe on the advanced Cluster Extensions page of the create cluster wizard.
- When you create an environment, you can select a previously added recipe on the Data Access and Data Lake Scaling page of the environment creation wizard, under Advanced Options > Cluster Extensions > Recipes.
- When you create an environment, you can select a previously added FreeIPA recipe on the **Region, Networking, and Security** page of the environment creation wizard, under Advanced OptionsCluster ExtensionsRecipes.
- You can also attach recipes to Cloudera Data Hub or Data Lake clusters when you create an environment/Data Lake or Cloudera Data Hub cluster through the CDP CLI.

Update a FreeIPA recipe

You can attach or detach recipes to/from existing FreeIPA clusters in an available state. Using this capability, you can update a recipe by removing it from the cluster, replacing the old recipe with a modified recipe of the same type, and attaching the new modified recipe to the cluster.

Attaching or detaching a recipe will not execute the recipe. The next execution of the recipe will take place based on the type of the recipe. After an upscale, a newly attached recipe runs only on the new hosts.

Required role (one of the following):

- PowerUser on Cloudera tenant
- Owner of the environment
- EnvironmentAdmin

Steps

For CDP UI

1. Create a new recipe (with updated/modified content) of the same type as the old recipe that you want to replace.
2. In the Cloudera Management Console, select Environments from the left pane and click on your environment. Next, click on FreeIPA tab and then on the Recipes tab.

3. Find the recipe that you want to remove in the list of recipes for the FreeIPA.
4. Click Remove Recipe next to the name of the recipe that you want to remove, then click Yes in the confirmation window.
5. Once you have removed the old recipe, click on the Add Recipe button for the FreeIPA. Then select the name of the new recipe that contains the modified content and click Add.

For CDP CLI

1. Create a new recipe (with updated/modified content) of the same type as the old recipe that you want to replace.
2. You can use the CDP CLI to detach and attach recipes from a FreeIPA cluster:

```
cdp environments detach-free-ipa-recipes
--environment <value>
--recipes <value>

cdp environments attach-free-ipa-recipes
--environment <value>
--recipes <value>
```

Result: You should see the new recipe appear for the same host group. After this change, the next recipe execution will execute the new script.

Managing recipes from CLI

You can manage recipes from CLI using cdp datahub commands.

Required role: EnvironmentCreator can create a shared resource and then assign users to it.

SharedResourceUser or Owner of the shared resource can use the resource. The Owner of the shared resource can delete it.

Note: Currently, recipes use cdp datahub commands regardless of whether the recipe is intended to run on Data Hub, Data Lake, or FreeIPA cluster nodes.

- Register a new recipe: cdp datahub create-recipe --recipe-name <value> --recipe-content <value> --type <value>

  Supported types:

  - PRE_SERVICE_DEPLOYMENT (formerly PRE_CLOUDERA_MANAGER_START)
  - POST_CLOUDERA_MANAGER_START (this option is not available for FreeIPA recipes)
  - POST_SERVICE_DEPLOYMENT (formerly POST_CLUSTER_INSTALL)
  - PRE_TERMINATION
- List all available recipes: cdp datahub list-recipes
- Describe a specific recipe: cdp datahub describe-recipe --recipe-name <value>
- Delete one or more existing recipes: cdp datahub delete-recipes --recipe-name <value>

Cloudera Management Console

Notifications

Date published: 2019-08-22
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Receiving notifications.....4

Subscribing to resource notifications..... 5

Setting up email announcements..... 5

Creating and managing distribution lists..... 6

Receiving notifications

You can subscribe to receive announcements and notifications in Cloudera Public Cloud about various events from product updates to data service specific alerts. Announcements and notifications are received in the application or through emails.

Announcements include Cloudera product related announcements that can include updates related to End of Life (EOL), End of Support (EOS), Technical Service Bulletins (TSBs), and maintenance updates. Announcements are broadcasted to all Cloudera users, and either to all or specific tenants.

Notifications include service and resource related alerts that are automatically generated by the platform. For example, cluster state changes, certification and upgrade alerts, resource exhaustion and consumption notifications. Notifications can be received by users of a tenant who have subscribed to the resource events of a Cloudera service, such as Cloudera DataFlow or User Management.

For more infromation about how to set up notifications for Cloudera DataFlow, see *Setting up service and deployment notifications*.

You can receive notifications in-app and through email messages.

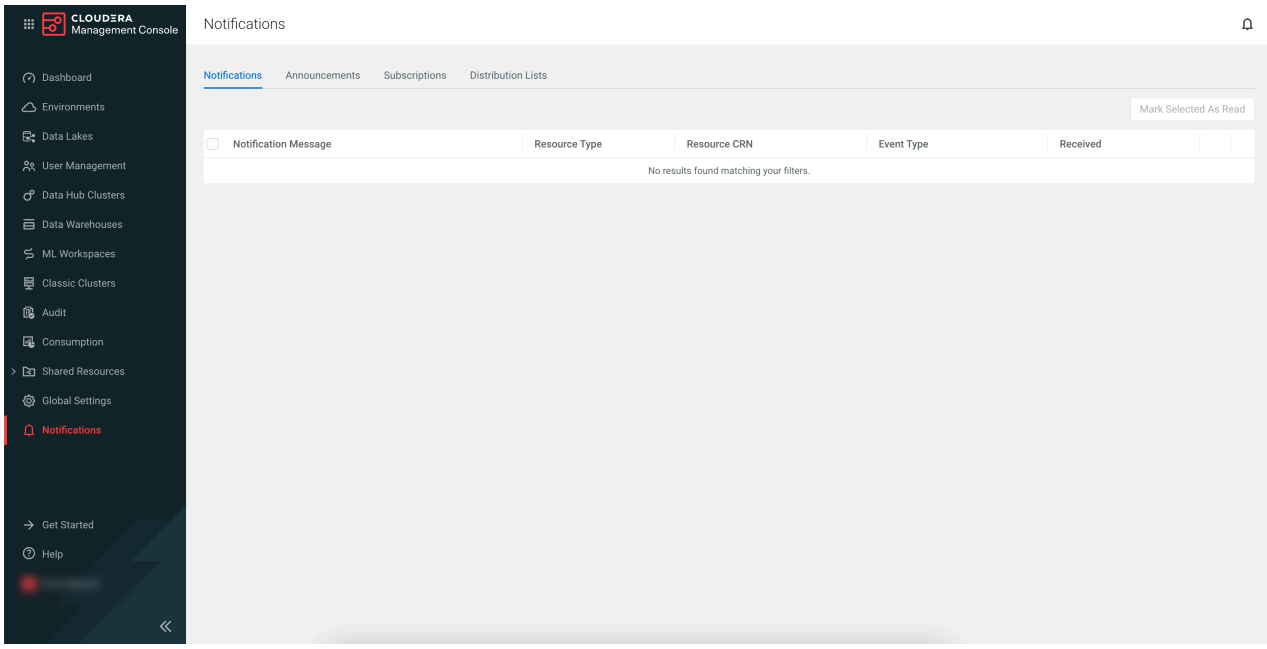

The **Notifications** page consists of the following tabs:

- **Notifications** and **Announcements** list the active and read notifications about the product and resource.
- **Subscriptions** lists the active notification subscriptions for the services.
- **Distrbituion Lists** lists the group subscriptions for the services that are created based on allowed email domains.

In-app notifications

A new notification is indicated by 🔔 at the top right corner on the **Notifications** page in Cloudera Management Console. You can open the notifications view using 🔔, and manage your notifications from in-app using ⋮. You can choose from the following actions:

- See all notifications and manage the subscriptions on the **Notifications** page
- Mark the notifications as read

In-app notifications are enabled by default. You can also view the in-app notifications by navigating to the **Notifications** page, and selecting the **Notifications** or **Announcements** tab.

Email notifications

You can subscribe to receive the notifications through email as well. Email notifications are sent from Cloudera's email address. You can subscribe to receive an email about all or a specific type of global announcement. You can also create Distribution lists to subscribe a set of users to receive resource notifications through email.

Related Information

Setting up service and deployment notifications | Cloudera DataFlow

Subscribing to resource notifications

Resource notifications enable you to receive automated alerts from service events in Cloudera Management Console or through email. To be able to subscribe to a resource notification, you must enable notifications at the data or platform service, then configure the resource notification. When selecting email as a notification channel, you can also create a distribution list that allows you to subscribe multiple users to the same set of email notifications.

Required role: NotificationSubscriber

You can enable notifications to receive automated events, alerts and service information of a data or platform service that supports notifications. You can create a subscription on the data or platform service side. Creating the subscription will redirect you to Cloudera Management Console Notifications where you can choose the type of notification you want to subscribe to and how you want to receive the notification.

You can subscribe to receive Cloudera DataFlow environment and deployment related notifications. For more information about how to enable and configure notifications in Cloudera DataFlow, see the *Setting up service and deployment notifications* documentation.

Creating and managing subscriptions

After enabling notifications at the data or platform service, you are redirected to Cloudera Management Console Notifications Create Subscription where you need to select the type of resource notification and how to receive the notification.

When the subscription is created, you can use the Subscription tab to view the list of **Active Subscriptions**. You also have the option to edit the subscriptions if you want to unsubscribe from a type of notification or you can change how to receive the notifications. You can also delete an active subscription if you no longer need it.

Related Information

Setting up service and deployment notifications | Cloudera DataFlow

Setting up email announcements

You can configure global announcements to be sent out through email as well. Email announcements are sent a Cloudera email address.

Procedure

1. Navigate to the Notifications page.
2. Select the Announcements tab.

3. Click Settings.
   The **Global Announcement Settings** window appears.

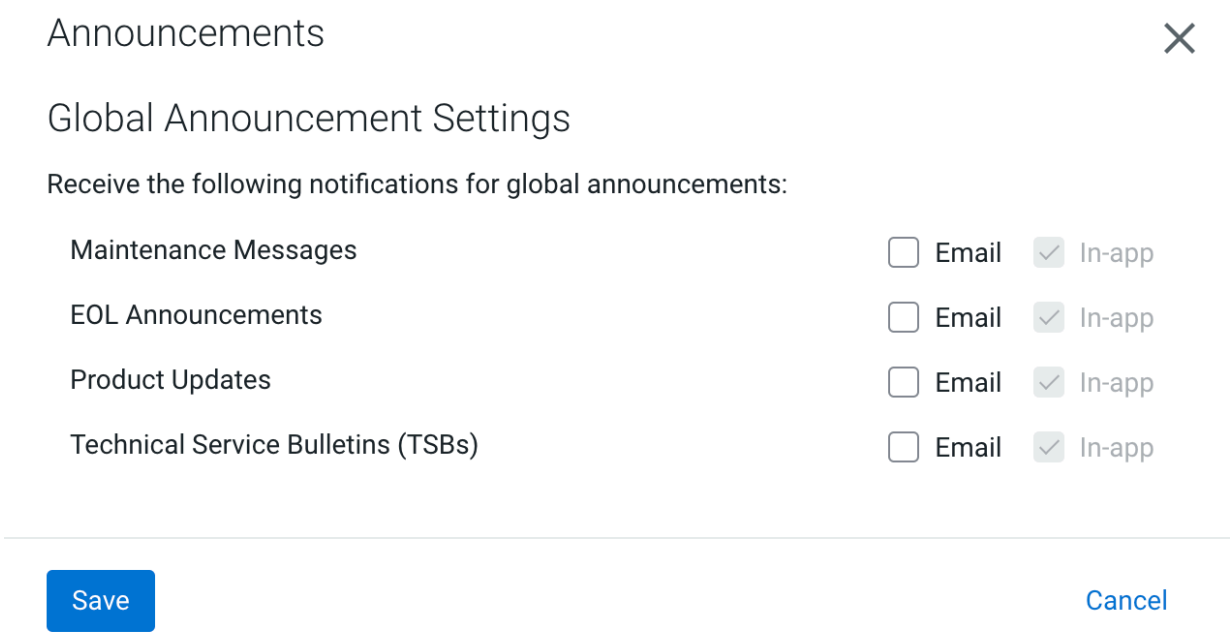

4. Select all or a specific type of announcement that should be sent in email.
   You can choose from the following type of announcements:
   - Maintenance Messages - include announcements about planned Cloudera maintenance work.
   - EOL Announcements - include announcements about end of life (EOL) date for versions and services.
   - Product Updates - include announcements about Cloudera product updates.
   - Technical Service Bulletins (TSBs) - include announcements about communicated TSBs related to the product
5. Click Save.

Creating and managing distribution lists

Distribution lists allow you to subscribe a group of users to receive resource notifications. You can create the distribution lists based on which users should receive certain types of resource notifications. To create a distribution list, the email addresses must belong to an allowed domain. Admins can create five allowed domains for a tenant. Admins can subscribe or unsubscribe users to the email notifications that are enabled for the distribution list they belong to.

Required role: NotificationDistributionListAdmin

Procedure

1. Navigate to the Notifications page.
2. Select the Distribution Lists tab.

3. Click Settings.
   The **Distribution List Settings** window appears.

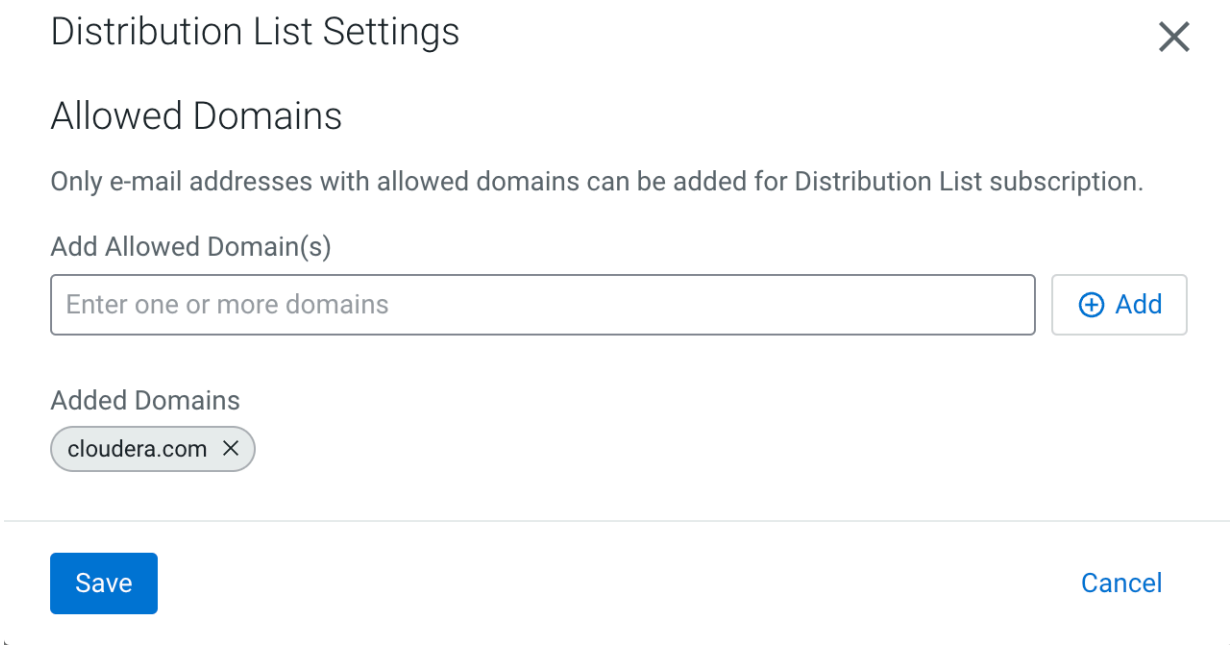

4. Provide the domain that will be used in the email addresses.
5. Click Save.
6. Navigate to the resource where you want to subscribe to notifications, and provide the email addresses that you want to receive notifications.

   You can manage the group subscriptions on the **Notifications** page under Distribution Lists. Using ⋮, you can go to the subscribed resource, edit and delete the distribution list.

Cloudera Management Console Overview

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Management Console 4

Interfaces 5

Core concepts 6

Environments 6
Credentials 7
Data Lakes 7
Shared resources 8
Classic clusters 9

Cloudera accounts 9

Cloudera Management Console

The Cloudera Management Console is an administrative service used by Cloudera administrators to manage environments, users, and Cloudera services.

The Cloudera Management Console allows you to:

- Configure SSO, manage users, and decide who can access which resources.
- Register your existing on-prem Cloudera Distribution of Hadoop (CDH) or Hortonworks Data Platform (HDP) clusters in order to burst your data to the cloud. In the Cloudera world, these clusters are called classic clusters.
- Register your cloud provider environments (such as your AWS, Azure, or GCP account) in Cloudera and then launch Cloudera Data Hub clusters, Cloudera Data Warehouses, Cloudera AI workbenches, Data Engineering and DataFlow clusters, and Cloudera Operational Databases within these environments and determine which users have access to which resources. These clusters are attached to a Data Lake that runs within the environment and provides security and governance for all attached clusters.
- Utilize services such as Cloudera Data Catalog, Cloudera Observability, and Cloudera Replication Manager:
  - Cloudera Data Catalog - A centralized management tool for searching, organizing, securing, and governing data across environments.
  - Cloudera Observability - A centralized management tool for analyzing and optimizing workloads within and across environments.
  - Cloudera Replication Manager - A centralized management tool for replicating and migrating data and the associated metadata across environments.

The following diagram illustrated the functionality described above:

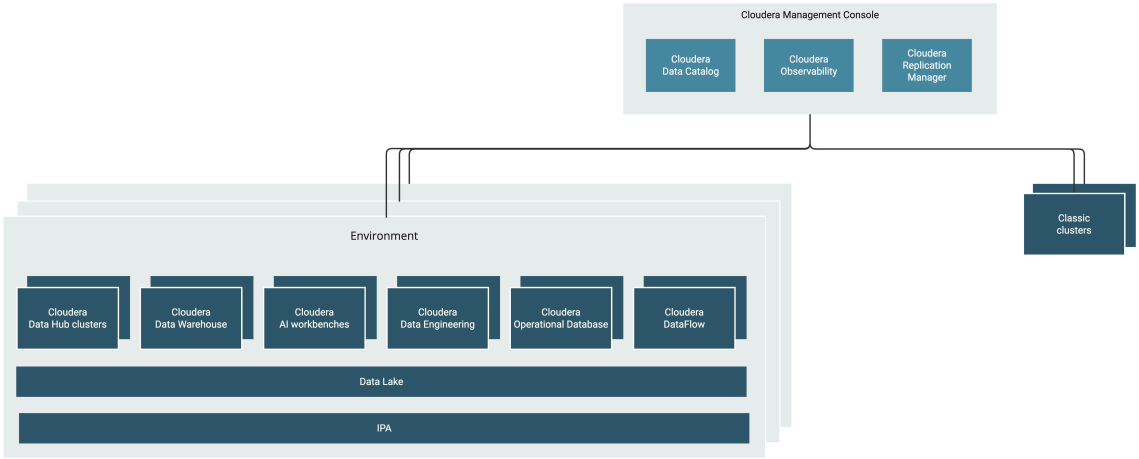

In addition, the Cloudera Management Console allows you to:

- Configure SSO, manage users, and decide who can access which resources.
- Download and configure CDP CLI.

Figure 1: Cloudera Management Console

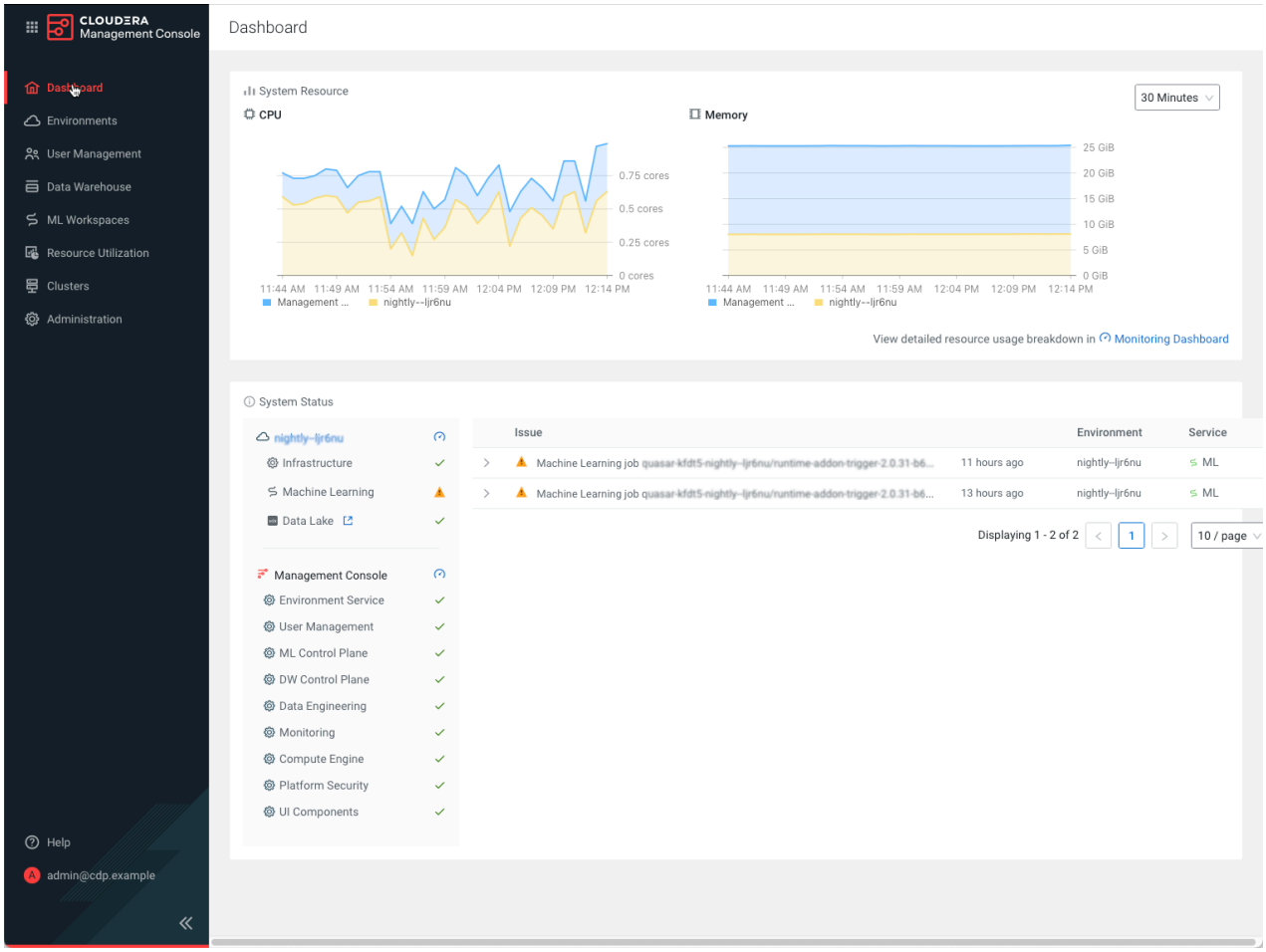

Related Information

Interfaces

Core concepts

Cloudera accounts

Interfaces

There are three basic ways to access and use Cloudera services: web interface, CLI client, and SDK.

Cloudera web interface

The Cloudera web interface provides a web-based, graphical user interface accessible via a browser. As an admin user, you can use the Cloudera web interface to register environments, manage users, and provision Cloudera service resources for end users. As an end user, you can use the Cloudera web interface to access Cloudera service web interfaces to perform data engineering or data analytics tasks.

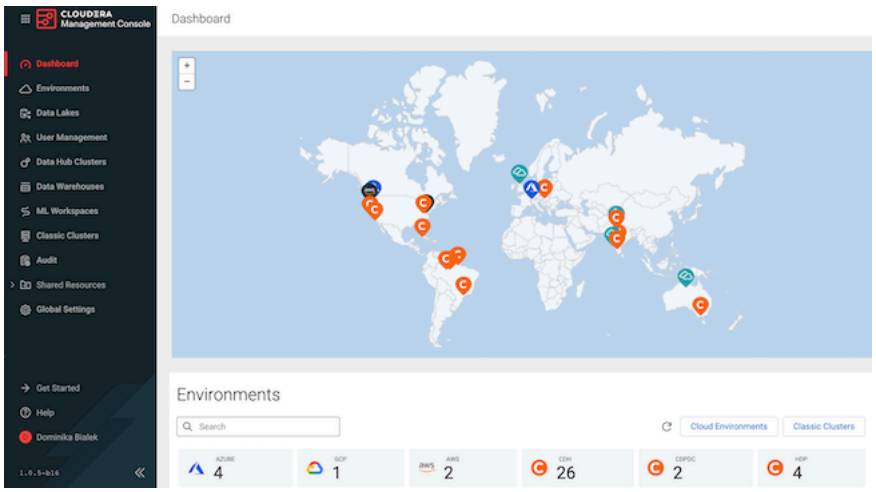

The Cloudera web interface allows you to access multiple Cloudera services. You can switch between different Cloudera services by using the icon in the top left corner.

CDP CLI

If you prefer to work in a terminal window, you can download and configure the CDP client that gives you access to the CDP CLI tool. The CDP CLI allows you to perform the same actions as can be performed from the management console. Furthermore, it allows you to automate routine tasks such as Data Hub cluster creation.

Cloudera SDK

You can use the Cloudera SDK for Java to integrate Cloudera services with your applications. Use the Cloudera SDK to connect to Cloudera services, create and manage clusters, and run jobs from your Java application or other data integration tools that you may use in your organization.

Core concepts

The following concepts are key to understanding the Cloudera Management Console service and Cloudera in general:

Environments

In Cloudera, an environment is a logical subset of your cloud provider account including a specific virtual network. You can register as many environments as you require.

In the on premise world, your clusters run on machines in your data center and are accessible on your corporate network. In contrast, when you launch clusters in the cloud, your cloud provider, such as AWS, Azure, or Google Cloud, provides all the infrastructure including private networks and VMs. You provision a private network in a selected region, configure access settings, and then provision resources into that private network. All these resources are accessible as part of your cloud provider account and can be shared by users within your organization.

The “environment” concept of Cloudera is closely related to the private network in your cloud provider account. Registering an environment provides CDP with access to your cloud provider account and identifies the resources in your cloud provider account that Cloudera services can access or provision. Once you’ve registered an environment in Cloudera, you can start provisioning Cloudera resources such as clusters, which run on the physical infrastructure in an AWS, Azure, or Google Cloud data center.

You may want to register multiple environments corresponding to different regions that your organization would like to use.

For more information about environments, refer to cloud provider specific documentation linked below.

Related Information

Introduction to AWS environments

Introduction to Azure environments

Introduction to Google Cloud environments

Credentials

A credential allows Cloudera to authenticate with your cloud provider account and obtain authorization to provision cloud provider resources on your behalf.

The authentication and authorization process varies depending on the cloud provider, but is typically done by assigning a specific role (with a specific set of permissions) that can be assumed by Cloudera, allowing it to perform certain actions within your cloud provider account.

A credential is a core component of an environment, providing access to the region and virtual network that make up the environment and allowing Cloudera to provision resources within that environment. Credentials are managed separately from environments, because you can reuse the same credential across multiple environments if needed. For example, the following diagram presents a scenario where one credential (credential A) is used by a single environment (cloud provider environment 1) but another credential (credential B) is used by multiple environments (cloud provider environment 2 and 3). In this case, it is implied that cloud provider environment 2 and 3 must represent the same AWS. Azure, Google Cloud account, but may correspond to different regions and/or VPCs/subnets.

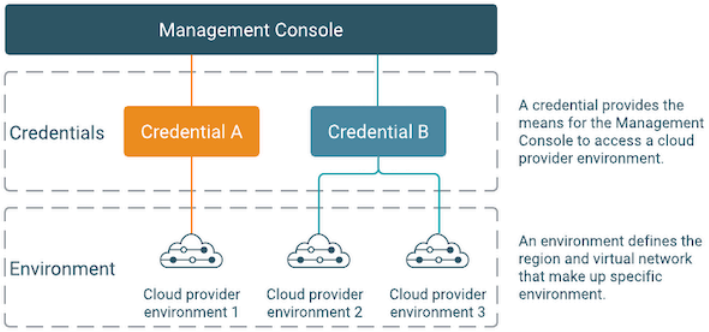

Related Information

Role-based credential on AWS

App-based credential on Azure

Provisioning credential for Google Cloud

Data Lakes

In Cloudera, a Data Lake is a service for creating a protective ring of security and governance around your data, whether the data is stored in cloud object storage or HDFS.

When you register an environment in Cloudera, a Data Lake is automatically deployed for that environment. The Data Lake runs in the virtual network of the environment and provides security and governance layer for the environment's workload resources, such as Cloudera Data Hub clusters.

Note: Currently, the Data Lake automatically deployed is scaled for light duty. In the future, you'll have more options for scaling your Data Lake.

The Data Lake provides a way for you to create, apply, and enforce user authentication and authorization and to collect audit and lineage metadata from across multiple ephemeral workload clusters. When you start a workload cluster in the context of a Cloudera environment, the workload cluster is automatically "attached" with the security and governance infrastructure of the Data Lake. "Attaching" your workload resources to the Data Lake instance allows the attached cluster workloads to access data and run in the security context provided by the Data Lake.

A Data Lake cluster includes Apache Knox. Knox provides a protected gateway for access to Data Lake UIs. Knox is also installed on all workload clusters, providing a protected gateway for access to cluster UIs.

While workloads can be short-lived, the security policies around your data schema are long-running and shared for all workloads. The Data Lake instance provides consistent and available security policy definitions that are available for current and future ephemeral workloads. All information related to metadata, policies, and audits is stored on external locations (external databases and cloud storage).

The Data Lake stores its metadata, policies, and audits in external databases and cloud storage, reducing the resource footprint on the cluster.

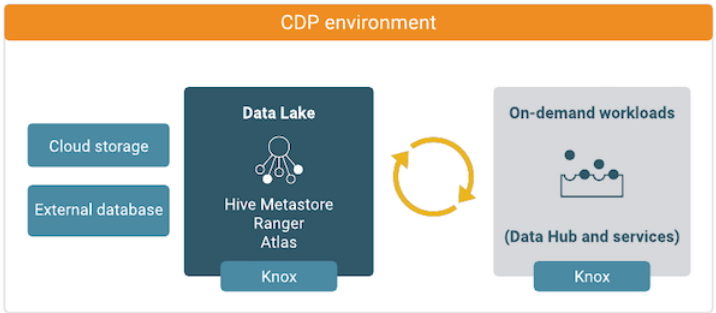

The following technologies provide capabilities for the Data Lake:

Component	Technology	Description
Schema	Apache Hive Metastore	Provides Hive schema (tables, views, and so on). If you have two or more workloads accessing the same Hive data, you need to share schema across these workloads.
Authorization Policies	Apache Ranger	Defines security policies around Hive schema. If you have two or more users accessing the same data, you need security policies to be consistently available and enforced.
Audit Tracking	Apache Ranger	Audits user access and captures data access activity for the workloads.
Governance	Apache Atlas	Provides metadata management and governance capabilities.
Security Gateway	Apache Knox	Supports a single workload endpoint that can be protected with SSL and enabled for authentication to access to resources.

Related Information

Data lakes

Shared resources

Cloudera allows you to manage and reuse certain resources across environments.

Shared resources include:

Resource	Description
Credentials	A credential allows your Cloudera environment to authenticate with your cloud provider account and to obtain authorization to provision cloud provider resources for Cloudera Data Hub clusters and other resources provisioned via Cloudera. There is one credential per environment.

Resource	Description
Cluster definitions	A cluster definition defines cloud provider specific Cloudera Data Hub cluster settings such as instance types, storage, and so on. Cluster definitions are specified in JSON format. Each Cloudera Data Hub cluster is based on a single cluster definition. Each cluster definition references a single cluster template.
Cluster templates	A cluster template is a blueprint that defines cluster topology, including the number of host groups and all components and sub-components installed on each host group. Cluster templates are specified in JSON format. Each Cloudera Data Hub cluster is based on a single cluster template.
Image catalogs	By default, Cloudera Data Hub includes an image catalog with default images. If necessary, you can customize a default image and register it as part of a custom image catalog. These images are used for creating Cloudera Data Hub clusters.
Recipes	A recipe is a custom script that can be uploaded and used for running a specific task on Cloudera Data Hub or Data Lake clusters. You can upload and select multiple recipes to have them executed on a Cloudera Data Hub or Data Lake cluster at a specific time.

Shared resources only need to be registered once and then can be reused for multiple environments. Since shared resources are not attached to a specific workload cluster, their lifespan is not limited to the lifespan of the cluster.

Related Information

Credentials
Cluster definitions
Cluster templates
Recipes

Classic clusters

Classic clusters are on-prem Cloudera Distribution of Hadoop (CDH) or CDP Data Center (CDP-DC) clusters registered in ClouderaPublic Cloud.

You can register your existing CDH or CDP-DC classic clusters in order to burst or migrate a workload to their public cloud environment by replicating the data and creating a Cloudera Data Hub cluster to host the workload.

Related Information

Classic clusters

Cloudera accounts

A Cloudera account (sometimes called a Cloudera tenant) is the management console where users log in to access their services. A Cloudera account contains one or more Cloudera environments.

Cloudera Accounts

Physically, all management consoles run in a shared infrastructure (the Cloudera Control Plane). Each management console has a unique ID and is logically isolated from other Cloudera accounts. For example, if you have two Cloudera accounts, you cannot view the resources in account 2 if you are logged into account 1. Each Cloudera account, including sub-accounts, has a separate invoice.

A Cloudera account can have one or more identity providers. Each identity provider is connected to one SAML provider, which in turn is connected to an Active Directory. A Cloudera account can be connected to multiple Azure Active Directory organizations (via identity provider) and multiple subscriptions. Cloudera creates one Cloudera account per customer; you can create additional Cloudera accounts through a manual approvals process.

For example, the "Cloudera CDP Demos" account is connected to:

- 3 identity providers, each connected to an LDAP Server via SAML:
  - Corporate OKTA, so that anybody with a cloudera.com email can log in (as long as they are entitled).
  - An LDAP server used for workshops, so that we can provision temporary users without having to go through corporate account lifecycle management.
  - An LDAP server hosting demo users, so that we can create new personas on demand for demo scenarios.
- Multiple Azure subscriptions; one per budget item (Sales, Product Management, Services, Tradeshow Demos etc.).
- This allows different groups to host their own demos and pay for them using their own cloud account.

Cloudera Environments

A Cloudera environment exists inside a Cloudera account, and each Cloudera account can have many environments. An environment is a logical container for a Data Lake cluster and any workloads that are attached to that cluster, such as Cloudera Data Hub clusters, Cloudera Data Warehouses, Cloudera AI environments, etc. All services running within an environment share the same metadata and use the same Data Lake.

Each environment can be associated with only one cloud account. This association is called a Cloudera credential and points to either:

- A service principal for a specific Azure subscription
- A cross-account role for a specific AWS account

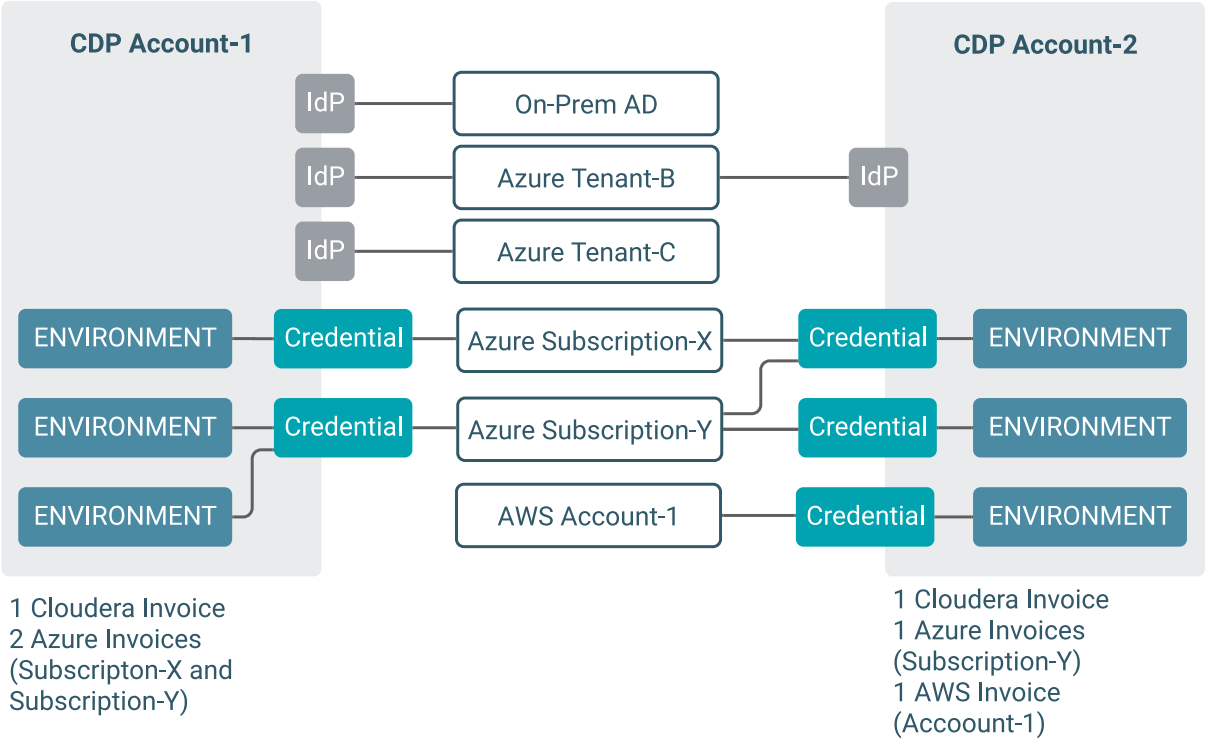

Cloudera Management Console

Outbound Internet Access and Proxy

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Outbound internet access and proxy.....4

Using a non-transparent proxy.....4

Setting up a non-transparent proxy in Cloudera.....5

Setting up a web proxy for TLS inspection.....9

Outbound internet access and proxy

This section provides information on the outbound network destinations for Cloudera, and instructions on how to configure Cloudera to use a proxy for outbound access.

Depending on your enterprise requirements, you may have limited or restricted outbound network access and/or require the use of an internet proxy. Registering a cloud provider environment in Cloudera, as well as creating clusters within an environment requires outbound network access to certain destinations, and in some cases must go through a proxy.

Scenario	Documentation
My environment has limited outbound internet access	Refer to the following documentation for information on network rules: • AWS outbound network access destinations • Azure outbound network access destinations • GCP outbound network access destinations
My environment requires use of a proxy for outbound internet access	Refer to Using a non-transparent proxy on page 4.

Related Information

Using a non-transparent proxy

Using a non-transparent proxy

Refer to this section if your environment requires all internet traffic to go through an internet proxy. You can use a proxy server to control the connections that are allowed from your VPC or VNet and block unauthorized connections initiated from your environment.

When creating a Cloudera environment, you can set up an HTTP proxy such as Squid or a comparable product. For a majority of use cases, this is enough to direct the traffic through a proxy.

Proxy servers can be used for:

- FreeIPA backups: Backups created on an hourly basis are uploaded to cloud storage S3/ADLS Gen2.
- Parcel downloads: Although Cloudera currently only supports pre-warmed images, it is a requirement to download parcels from archive.cloudera.com when an upgrade is performed.
- Cluster Connectivity Manager: Communication via CCMv1 and CCMv2.
- TLS and Deep Packet Inspection (DPI): TLS and DPI Inspection can be performed through the use of a proxy. To see how to configure this, refer to the Setting up a web proxy for TLS inspection section below.

The following diagram illustrates the communication between the customer’s Cloudera environment and the Cloudera Control Plane in a cloud provider network (VPC/VNet) via a web proxy:

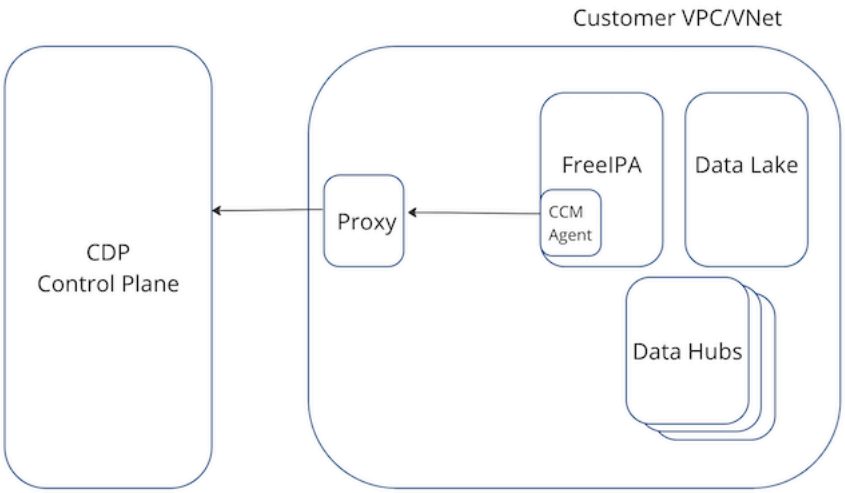

Note: The above diagram illustrates a simple architecture where the web proxy is located within the customer VPC/VNet. However, the web proxy could be located anywhere in the network between the Cluster Connectivity Manager agent and the Cluster Connectivity Manager server: in the customer VPC outside of Cloudera (as above), in another customer VPC, or in a cloud-based web proxy service.

Supported Cloudera services

The following Cloudera services allow the use of a web proxy:

CDP service	AWS	Azure	GCP
Data Lake	GA	GA	GA
FreeIPA	GA	GA	GA
Cloudera Data Engineering	GA		
Cloudera Data Hub	GA	GA	GA
Cloudera Data Warehouse	GA		
Cloudera DataFlow	GA		
Cloudera AI	GA		
Cloudera Operational Database			

Note that in order to use a non-transparent proxy with Cloudera data services (such as Cloudera Data Engineering, Cloudera Data Warehouse, Cloudera DataFlow, and Cloudera AI), you must first configure it at the environment level and then once again when enabling/activating the Cloudera data service.

Setting up a non-transparent proxy in Cloudera

To set up a proxy server you can register an http proxy server as a shared resource and then add that shared resource when you set up your environment.

Note: Once you register a proxy in Cloudera, there is no option to edit the proxy registration. You need to delete the proxy registration in Cloudera and register the proxy again.

Required role: EnvironmentCreator can register a proxy in Cloudera and manage user access to the proxy. Owner or SharedResourceUser can view the proxy details. Owner can delete the proxy registration from Cloudera.

Steps

For CDP UI

1. Log in to the Cloudera web interface.
2. Navigate to the Cloudera Management Console.
3. Select Shared Resources Proxies from the left navigation pane.
4. Click Create Proxy Configuration.

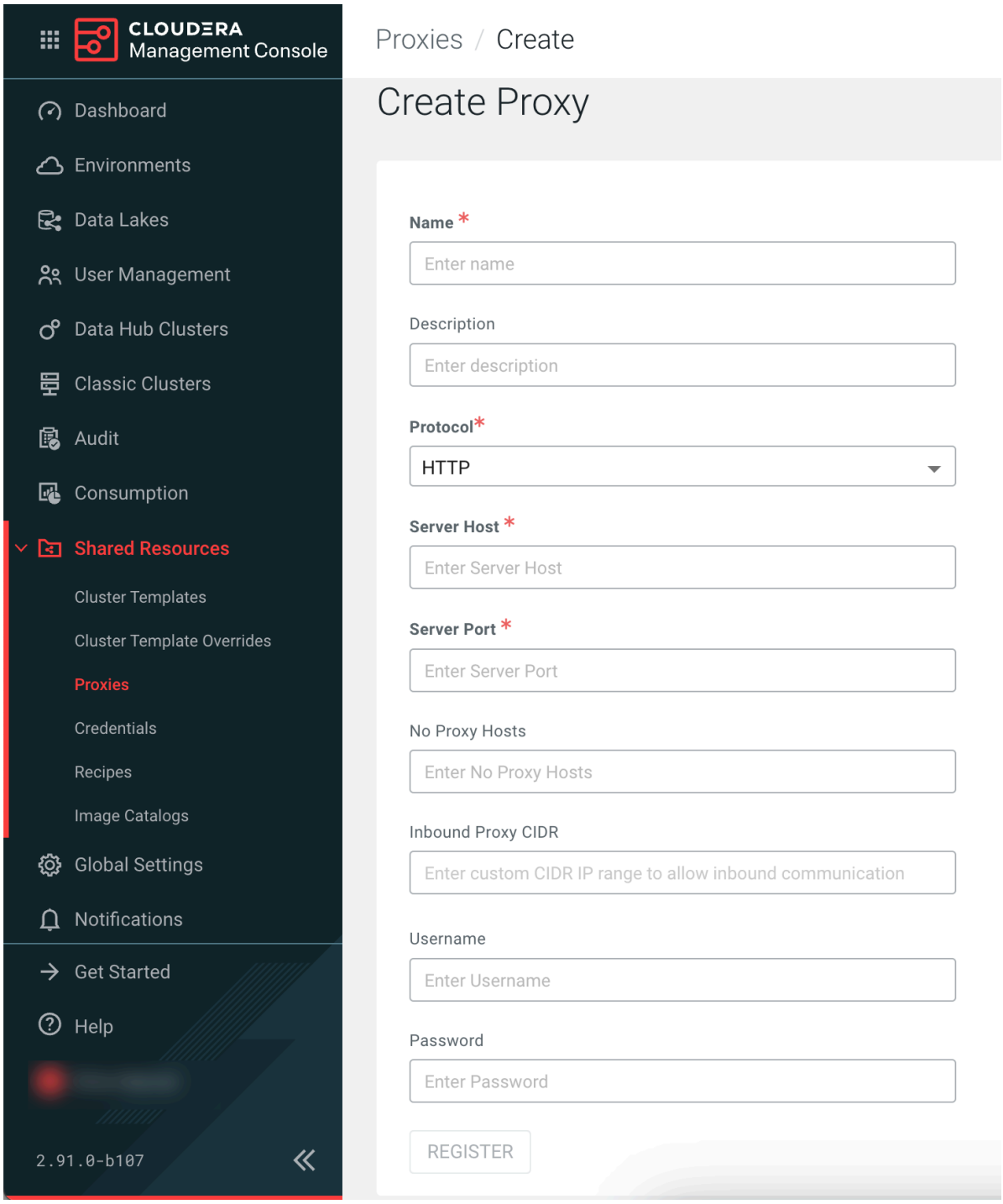

5. Enter the information for your proxy server:

Parameter	Description
Name (Required)	Provide a name for the proxy. The name will be used for this specific proxy in CDP.
Description	You can optionally specify a longer description for this proxy.

Parameter	Description
Protocol (Required)	Select the protocol used by the proxy: HTTP or HTTPS.
Server Host (Required)	Provide proxy server's host.
Server Port (Required)	Provide the proxy server's port.
No Proxy Hosts	The no-proxy field allows you to designate specific IP addresses, domains, or subdomains that bypass the proxy. This setting can be useful for locally resolvable and internal endpoints, for example the CCMv2 agent or the metering agent. Enter the values for this field in a comma-separated list. For example: 172.100.0.110,domainname.com,my.host.com Note the following guidelines: • The period character (".") is allowed as a prefix for domain names only • CIDR notation is not allowed **Note:** If you are running a Cloudera environment on Azure using Cloudera Runtime 7.2.14 or newer and you don't want to make an exception for management.azure.com (which is mentioned as required in Azure outbound network access destinations), you can add the following to the No Proxy Hosts list: `localhost,127.0.0.1,169.254.169.254,168.63.129.16`
Inbound Proxy CIDR	Provide a custom CIDR IP range to allow inbound communication. Required when you use an FQDN instead of an IP address to define your proxy. Without providing the CIDR IP range security groups cannot be adjusted to allow communication with the proxy and Kubernetes server.
User name	If needed, provide a user name to access the proxy.
Password	If needed, provide a password to access the proxy.

6. Click REGISTER.
7. Click Environments in the left navigation pane, then click Register Environment.
8. Add your environment information, navigating through the Register Environment and Data Lake Scaling steps.

9. When you reach the Region, Networking and Security steps, choose the Proxy you registered.

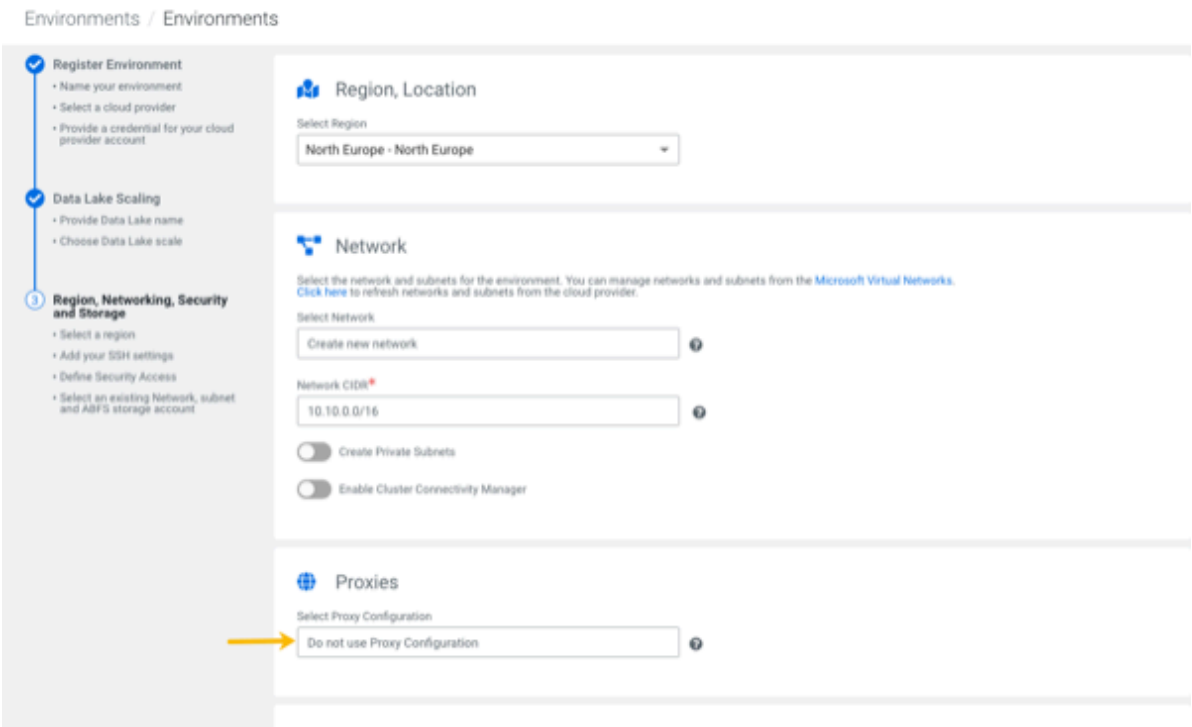

10. Finish setting up your Environment.

For CDP CLI

As an alternative to using the UI, you can also register the proxy using the CLI.

1. Use the following commands:

```
cdp environments create-proxy-config \
  --proxy-config-name companyProxy \
  --host 10.102.0.19 \
  --port 3128 \
  --user squid \
  --password squid \
  --protocol http
```

2. Provide the proxyConfigName in the environment JSON:

```
...
"subnetIds": [
  "subnet-1",
  "subnet-2",
  "subnet-3"
  ],
  "proxyConfigName": "companyProxy" must be on the root level
}
```

3. Or in the --proxy-config-name argument of the environment creation command, enter the following:

   AWS:

```
cdp environments create-aws-environment \
  --cli-input-json '{...}' \
```

```
  --proxy-config-name companyProxy
```

Azure:

```
cdp environments create-azure-environment \
  --cli-input-json '{...}' \
  --proxy-config-name companyProxy
```

Note: These proxy settings do not apply to cluster recipes. If you planning to use the recipes, then you can set the proxy settings manually, if needed. You can find the proxy settings in the /etc/cdp/proxy.env file.

Related Information

Use a non-transparent proxy with Cloudera AI on AWS environments

Setting up a web proxy for TLS inspection

After setting up the proxy server in Cloudera, you can further configure it to perform TLS interception and Deep Packet Inspection (DPI).

Without a web proxy, a single TLS session is initiated from the Cluster Connectivity Manager agent and terminated at the Cluster Connectivity Manager server within the Cloudera Control Plane. With the introduction of the web proxy, there are two TLS sessions: (1) a TLS session initiated from the Cluster Connectivity Manager agent terminating at the proxy and (2) a TLS session initiated from the proxy terminating at the Cluster Connectivity Manager server within the Cloudera Control Plane. The web proxy decrypts the packets of the TLS session, performs any operations on the clear text (such as DPI), and re-encrypts the packets onto the second TLS session. Thus the proxy behaves as a man-in-the-middle (MITM) that is able to view the communications between the Cluster Connectivity Manager agent and the Cluster Connectivity Manager server using TLS inspection.

The following diagram illustrates the communication between the customer's Cloudera environment and the Cloudera Control Plane in a cloud provider network (VPC/VNet) via a web proxy:

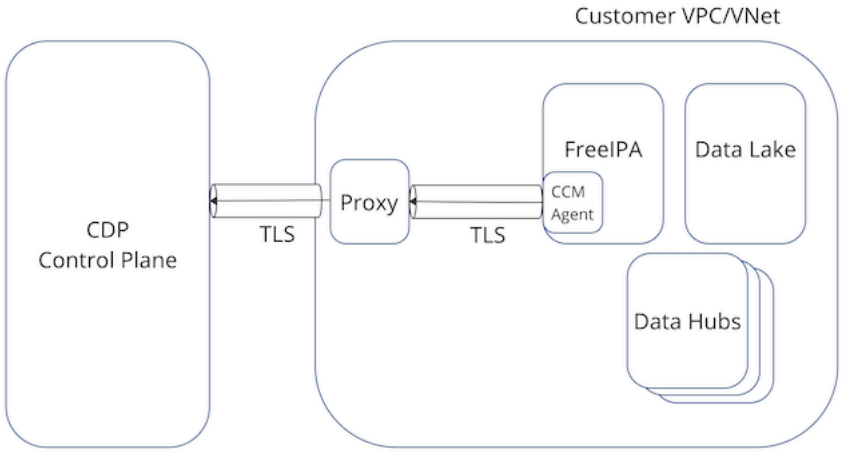

Note: The above diagram illustrates a simple architecture where the web proxy is located within the customer VPC/VNet. However, the web proxy could be located anywhere in the network between the Cluster Connectivity Manager agent and the Cluster Connectivity Manager server: in the customer VPC outside of Cloudera (as above), in another customer VPC, or as a cloud-based web proxy service.

The Cloudera architecture with and without proxy-based TLS inspection is illustrated in the following two diagrams.

The following diagram illustrates Cluster Connectivity Manager communication without a web proxy as MITM:

The following diagram illustrates Cluster Connectivity Manager communication with a web proxy as MITM:

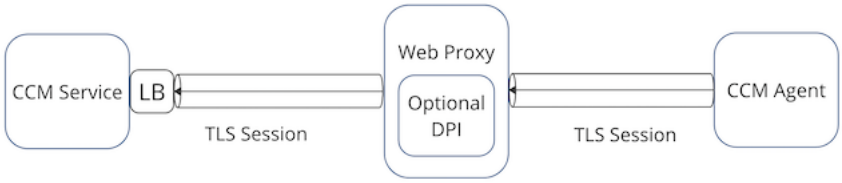

To configure TLS inspection, you need to set up your proxy to trust the certificate of Cluster Connectivity Manager, and, in turn, make sure that Cluster Connectivity Manager trusts the proxy's CA certificate.

Steps

1. Register a new Cloudera environment.
2. After the FreeIPA nodes are running, SSH into the FreeIPA nodes and perform the following set of steps:
   a. Get the CA certificate from /etc/jumpgate/config.toml and grab the pinned CA certificate from the agent.relayServerCertificate parameter.
   b. Configure your proxy server to trust this certificate for the Cluster Connectivity Manager traffic.
   c. Copy your proxy server's CA certificate and replace the contents of agent.relayServerCertificate in /etc/jumpgate/config.toml.
   d. Configure your proxy to start MITM-ing the underlying TLS connection.

Cloudera Management Console Release Notes

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What's new **4**
January 30, 2025 4
January 15, 2025 4
Older releases 4
2023 4
2022 9
2021 16
2020 29
2019 32

Log4j vulnerabilities **34**

Image catalog updates **34**

Known issues for Cloudera Management Console **35**

What's new

This section lists major features and updates for the Cloudera Management Console service.

January 30, 2025

This release of the Cloudera Management Console service introduces the following changes:

Enabling Secure Boot option for GCP (Preview)

VMs on GCP are created without the Secure Boot option enabled by default. You can request to have the Secure Boot option for subsequently provisioned VMs. For more information, see Enabling Secure Boot option for GCP.

Note:

You need to contact Cloudera Support to have this feature enabled.

January 15, 2025

This release of the Cloudera Management Console service introduces the following changes:

Added support for upgrading CMK enabled Azure Single Servers to Flexible Server

A managed identity is required for being able to upgrade to upgrade a CMK enabled Azure Single Server to Azure Flexible Server. It is now possible to add a managed identity an environment with Azure Single Server that is already encrypted with a Customer Managed Key thus making the upgrade from Azure Single Server to Azure Flexible Server possible for environments already configured with a CMK.

For more information, see Upgrading Azure Single Server to Flexible Server Prerequisites.

Older releases

Overview of new features, enhancements, and changed behavior introduced in earlier releases of Management Console.

2023

December 20, 2023

This release of the Management Console service introduces the following changes:

VHD images hosted in Azure Marketplace

Cloudera now publishes VHD images on Azure Marketplace for each minor Runtime release (for example, 7.2.17) and CDP uses these images by default during environment and Data Hub creation. In order for CDP to be able to load these images, customers need to accept Azure Marketplace terms and conditions either via CDP web UI (recommended) or Azure CLI. Note that RHEL 8 images on Azure are only available via Azure Marketplace.

In order to use this feature, you need to Grant the service principal additional Azure permissions:

1. On the scope of your Azure subscription:

```
"Microsoft.MarketplaceOrdering/offertypes/publishers/offers/plans/agreem
ents/write",
```

```
"Microsoft.MarketplaceOrdering/offerTypes/publishers/offers/plans/agree
ments/read”
```

2. On the scope of the CDP Azure resource group:

```
“Microsoft.Resources/deployments/whatIf/action”
```

For more information, see CDP images hosted in Azure Marketplace.

December 1, 2023

This release of the Management Console service introduces the following changes:

Additional permissions needed in the cross-account role

You should update the AWS cross-account role to include ec2:DescribeLaunchTemplateVersions policy action to allow CDP update AWS Launch Template UserData. CDP Public Cloud cluster operations such as Cluster Connectivity Manager version upgrades, proxy, and Salt credential management require updating the AWS Launch Template UserData. Without this capability, node repair, upgrade, and scale operations performed on the CDP Public Cloud environment, Data Lake (SDX), and Data Hub may fail.

November 1, 2023

This release of the Management Console service introduces the following changes:

Resize an existing Data Lake from single-AZ to multi-AZ

As part of the Data Lake resizing via CDP CLI, you can optionally resize a Data Lake from single-AZ to multi-AZ by adding the --multi-az flag to the Data Lake resize command. This is available via CDP CLI only. For more information, see Data lake scaling and Scaling the Data Lake through the CDP CLI.

October 30, 2023

This release of the Management Console service introduces the following changes:

Qatar Doha (me-central1) and KSA (me-central2) GCP region support

CDP adds support for launching environments, Data Lakes, and Data Hubs in the Qatar Doha (me-central1) and KSA (me-central2) GCP regions. See updated Supported GCP regions.

October 18, 2023

This release of the Management Console service introduces the following changes:

Elastic load balancer deletion protection on AWS

When Data Lakes and Data Hubs are created, CDP deploys load balancers on AWS for endpoint stability. With the release of this update, all newly created load balancers for Data Lakes and Data Hubs on AWS are configured with deletion protection enabled, provided that your cross-account policy has the required permissions.

Cloudera has updated the AWS cross-account policy definition with the additional elasticloadbalancing:ModifyLoadBalancerAttributes permission required to set and remove the load balancer deletion protection flag. If you would like to use this feature, please update your cross account policy on AWS by adding the elasticloadbalancing:ModifyLoadBalancerAttributes permission. If you are using the old policy definition, this new feature will not be available (that is, the deletion protection will not be set).

October 9, 2023

This release of the Management Console service introduces the following changes:

Resizing to a Enterprise Data Lake

You can now resize a light or medium duty Data Lake to an Enterprise Data Lake (EDL). Resizing to an EDL is available only on Runtime 7.2.17 and above. For more information, see Data Lake resizing.

October 5, 2023

This release of the Management Console service introduces the following changes:

Public Endpoint Access Gateway for GCP

You can enable Public Endpoint Access Gateway during GCP environment registration. For more information, see Public Endpoint Access Gateway.

September 20, 2023

This release of the Management Console service introduces the following changes:

Support for il-central-1 AWS region

The Tel Aviv (il-central-1) AWS region is now supported. You can register CDP environments and provision Data Hubs in that region. See updated Supported AWS regions.

September 6, 2023

This release of the Management Console service introduces the following changes:

Audit event archiving for GCP

You can now configure audit event archiving for Google Cloud Platform.

For more information, see GCP setup for audit archiving.

August 30, 2023

This release of the Management Console service introduces the following changes:

Data Lake database upgrade and default major version change

Newly-deployed Data Lake clusters on AWS or GCP with Cloudera Runtime 7.2.7 or above are now configured to use a PostgreSQL version 14 database by default.

Newly-deployed Data Lake clusters on Azure with Cloudera Runtime 7.2.7 or above will continue to use a PostgreSQL version 11 database by default.

The database for Data Lake clusters on AWS and GCP can now be upgraded to PostgreSQL version 14. If your AWS or GCP cluster requires an upgrade to PostgreSQL 14, you will receive a notification in the Management Console UI.

Cloudera strongly recommends that the database upgrade to PostgreSQL 14 for AWS and GCP clusters is performed on all clusters running PostgreSQL version 11 by November 9, 2023.

A database upgrade to PostgreSQL 14 for Azure Data Lakes will be available in the future. Any Data Lake clusters on Azure that require a database upgrade will be upgraded from PostgreSQL 10 to PostgreSQL 11.

For more information, see Upgrading Data Lake/Data Hub database

August 23, 2023

This release of the Management Console service introduces the following changes:

Configure proxy for TLS interception and Deep Packet Inspection

After setting up a web proxy server in CDP, you can further configure it to perform TLS interception and Deep Packet Inspection (DPI). For instructions, see Setting up a web proxy for TLS inspection.

Setting a default identity provider in CDP

You can optionally set a default identity provider (IdP) in CDP. If you do so, CDP will use the default IdP instead of the oldest IdP. For instructions, see Setting a default identity provider in CDP.

July 20, 2023

This release of the Management Console service introduces the following changes:

Enterprise Data Lakes

A new Data Lake shape called the enterprise Data Lake is now available for new deployments of Cloudera Runtime version 7.2.17. The enterprise Data Lake is a redefined version of medium duty Data Lakes that still offer failure resilience, but utilize resources and allocate memory more efficiently than a medium duty Data Lake at the same cost. For more information see Data Lake scale.

Medium duty Data Lakes deprecated

Medium duty Data Lakes are deprecated as of Runtime 7.2.17. You can upgrade a medium duty Data Lake from 7.2.16 to 7.2.17, but will not be able to upgrade it further. You can create a new 7.2.17 medium duty Data Lake through the CDP CLI, but Cloudera recommends using the enterprise Data Lake for new deployments. The ability to create medium duty Data Lakes will be removed from both the UI and CLI from 7.2.18.

July 19, 2023

This release of the Management Console service introduces the following changes:

CCM enabled by default in the UI

When you register a new AWS, Azure, or GCP environment in CDP via web interface, Cluster Connectivity Manager (CCM) is enabled by default and the option to disable it has been removed from the web interface. Also, CCM is now enabled by default in CDP CLI.

CCMv2 upgrade

CCMv2 upgrade is available for all customers. If your existing CDP environment was created with CCMv1, you will see a notification in your environment details to upgrade to CCMv2. See Upgrading from CCMv1 to CCMv2.

June 28, 2023

This release of the Management Console service introduces the following changes:

Cloudera Runtime 7.2.17

Cloudera Runtime 7.2.17 is now available and can be used for registering an environment with a 7.2.17 Data Lake and creating Data Hub clusters. For more information about the new Runtime version, see Cloudera Runtime. If you need to upgrade your existing CDP environment, refer to Data Lake upgrade and Data Hub upgrade documentation.

Rolling Data Lake upgrades

With the release of Cloudera Runtime 7.2.17, you can now upgrade your Data Lake in a rolling manner. The rolling upgrade allows you to upgrade the Data Lake Runtime and OS without stopping attached Data Hubs or Data Services. This allows workloads to continue running during the Data Lake upgrade operation.

Note: You need to contact Cloudera to have this feature enabled.

GCS fine-grained access control (Preview)

You can now register a CDP environment on GCP with RAZ enabled to use fine-grained access policies and audit capabilities available in Apache Ranger. See GCS Fine-Grained Access Control.

Note: You need to contact Cloudera to have this feature enabled.

Cross-version compatibility for Data Lake backup and restore

You can take a backup of a Data Lake that runs one version of Cloudera Runtime and restore the backup to a Data Lake that runs a different version of Runtime. The backup version must be an earlier/lower version Runtime than the Data Lake that you are restoring to. Version limitations apply and a Ranger/HMS schema upgrade may be required. See Cross-version support for Data Lake backup and restore for more details.

Cross-version compatibility for Data Lakes and Data Hubs

Backward compatibility between Data Lakes and Data Hubs has been introduced with Cloudera Runtime 7.2.17. It is no longer required that you perform Data Hub upgrades in lock-step with the Data Lake upgrade. If your Data Hub is on Runtime version 7.2.16 or later, it is compatible with a Data Lake on a newer Runtime version (7.2.17+). You can independently upgrade your Data Hubs at a later time if you choose to, though it is not required.

AWS Middle East UAE region

You can now register a CDP environment and create Data Hubs in the AWS Middle East UAE region (me-central-1). See updated Supported AWS regions.

Data Lake upgrade validations for Python dependency

If you are planning to upgrade the Runtime version in your existing Data Lake clusters to 7.2.17 or higher versions, you may be required to perform an additional step before upgrading.

You can verify whether your cluster is impacted by navigating to the Upgrade tab of your Data Lake cluster. If there is a warning message about missing prerequisites on the Upgrade page, you need to perform an additional upgrade step before moving to the 7.2.17 Cloudera Runtime version. The required steps may be different depending on your current Runtime version. See Data Lake upgrade for more details.

June 13, 2023

This release of the Management Console service introduces the following changes:

Zone selection within a GCP region

During GCP environment registration you can now select a zone within the selected GCP region. For example, if you selected the us-west1 region, you can select one of its three zones: us-west1-a, us-west1-b, or us-west1-c.

CDP credit consumption and usage insights

CDP includes a user interface that allows you to monitor your credit consumption and download your consumption records. See CDP credit consumption and usage insights.

April 19, 2023

This release of the Management Console service introduces the following changes:

FreeIPA and Data Lake instance type selection

You can now select the FreeIPA and Data Lake instance types when you register an environment. For more information see Register an AWS environment from CDP UI, Register an Azure environment from CDP UI, and Register a GCP environment from CDP UI.

AWS GP3 support for attached storage disks

AWS Data Lake and FreeIPA instances now support GP3 (SSD) volume types for attached storage. GP3 volumes allow you to increase performance (independently provisioning IOPS and throughput) without increasing storage size. GP3 volumes will deliver similar performance as GP2 volumes at a lower cost. GP3 is now the default attached storage type for instances that previously used GP2 storage.

March 29, 2023

This release of the Management Console service introduces the following changes:

FreeIPA instance type on GCP

Default FreeIPA instance type on GCP changed from n1-standard-2 to e2-standard-2.

January 11, 2023

This release of the Management Console service introduces the following changes:

Cloudera Runtime 7.2.16

Cloudera Runtime 7.2.16 is now available and can be used for registering an environment with a 7.2.16 Data Lake and creating Data Hub clusters. For more information about the new Runtime version, see Cloudera Runtime. If you need to upgrade your existing CDP environment, refer to Data Lake upgrade and Data Hub upgrade documentation.

Instance type selection for Data Lake and FreeIPA on AWS

If necessary, you can now select a larger or smaller instance type for a Data Lake or FreeIPA after the environment has been deployed. This is only supported on AWS. For more information see Vertically scaling instances types - Data Lake and Vertically scale FreeIPA instances.

2022

November 23, 2022

This release of the Management Console service introduces the following changes:

Data Lake resizing

You can now scale up a light duty Data Lake to the medium duty form factor, which has greater resiliency than light duty and can service a larger number of clients. You can trigger the scale-up in the CDP UI or through the CDP CLI. For more information, see Data Lake resizing.

November 3, 2022

This release of the Management Console service introduces the following changes:

Data Lake backup and restore for GCP

Backing up and restoring a GCP Data Lake is now supported. For more information, see Backup and restore for the Data Lake.

October 20, 2022

This release of the Management Console service introduces the following changes:

Azure Reference Network Architecture

New conceptual overview of the CDP Public Cloud network architecture for Azure, its use cases, and personas who should be using it.

For more information, see Azure Reference Network Architecture.

September 27, 2022

This release of the Management Console service introduces the following changes:

Database Upgrade and default major version change

Newly deployed Data Lake and Data Hub clusters with Cloudera Runtime 7.2.7 or above are now configured to use a PostgreSQL version 11 database by default.

A new Database Upgrade capability is now available for existing Data Lake and Data Hub clusters. If you are running clusters on Cloudera Runtime version 7.2.6 or below, upgrade to a more recent version before performing the database upgrade.

The major version of the database used by Data Lake or Data Hub clusters is now also displayed on the Database page of the respective service.

Cloudera strongly recommends that the Database Upgrade is performed on all clusters running PostgreSQL version 10 before November 10, 2022.

For more information, see Upgrading database to Postgres 11

FreeIPA recipes and recipe type changes

You can register and attach recipes to run on a specific FreeIPA host group. For more information, see Recipes.

The following recipe types have been renamed for Data Hub, Data Lake, and FreeIPA recipes:

- pre-service-deployment (formerly pre-cluster-manager-start)
- post-service-deployment (formerly post-cluster-install)

These changes will not affect existing recipe automation.

September 1, 2022

This release of the Management Console service introduces the following changes:

Validate and prepare for upgrade

Before you perform a Data Hub upgrade, you can run the new Validate and Prepare option to check for any configuration issues and begin the Cloudera Runtime parcel download and distribution. Using the validate and prepare option does not require downtime and makes the maintenance window for an upgrade shorter. For more information see Preparing for an upgrade.

AWS Hong Kong region

You can now register a CDP environment and create Data Hubs in the AWS Hong Kong Region (ap-east-1). See updated Supported AWS regions.

July 28, 2022

This release of the Management Console service introduces the following changes:

Changed permissions for managing proxies in CDP

You no longer need to be a PowerUser to register and manage a proxy in CDP. The new minimal roles are as follows:

- EnvironmentCreator can register a proxy in CDP.
- Owner or SharedResourceUser can view details of a proxy.
- Owner can delete a proxy registration from CDP.

This change has been introduced for new proxy registrations only; That is, proxies registered prior to this change continue to be managed by a PowerUser.

See updated Setting up a non-transparent proxy in CDP.

July 12, 2022

This release of the Management Console service introduces the following changes:

New documentation for CDP Public Cloud upgrade

The CDP Public Cloud upgrade advisor, which gives an overview and FAQ of the upgrade process, is now available. See CDP Public Cloud upgrade advisor.

FreeIPA scaling

You can resize your existing FreeIPA cluster via CDP CLI. Upscaling FreeIPA is recommended after performing Data Lake scaling. For more information, see Resize FreeIPA.

July 1, 2022

This release of the Management Console service introduces the following changes:

Support for Cloudera AI in ap-1 and eu-1 regional Control Planes

Cloudera AI is now supported in the ap-1 (Australia) and eu-1 (Germany) regional Control Planes.

June 29, 2022

This release of the Management Console service introduces the following changes:

New "Advanced Options" section in environment registration wizard

The environment registration UI now features a new "Advanced Options" section on some of the pages, which includes some options that were previously featured in the main UI sections. The options that have been moved to the "Advanced Options" sections include:

- On the Data Access and Data Lake Scaling page:
  - Multi-AZ configuration for Data Lake and FreeIPA (available for AWS only)
  - Recipe selection for Data Lake

More options will be added to the "Advanced Options" in the future.

New option to delete attached volumes during Data Lake repair

When you initiate a repair from the Data Lake Hardware tab, you have the option to delete any volumes attached to the instance. For more information see Performing manual Data Lake repair.

June 27, 2022

This release of the Management Console service introduces the following changes:

Public Endpoint Access Gateway for Azure

During Azure environment registration, you can optionally enable Public Endpoint Access Gateway, which provides secure connectivity to UIs and APIs in Data Lake and Data Hub clusters deployed using private networking, allowing users to access these resources without complex changes to their networking or creating direct connections to cloud provider networks. With this release, Public Endpoint Access Gateway is general availability for AWS and Azure, and it remains preview for GCP. See Public Endpoint Access Gateway.

Generate workload username based on email

By default, workload usernames are generated using the identity provider user ID. Alternatively, you can now generate workload usernames based on users' email addresses. This is useful in cases when the identity provider user ID is an opaque ID, like a uuid or employee ID, which gives equally opaque workload usernames. For more information, see Generating workload usernames based on email.

AWS Jakarta region

You can now register a CDP environment and create Data Hubs in the AWS Jakarta Region (ap-southeast-3). See updated Supported AWS regions.

Support for Operational Database in ap-1 and eu-1 regional Control Planes

Cloudera Operational Database is now supported in the ap-1 (Australia) and eu-1 (Germany) regional Control Planes.

Restricting all Cloudera SSO access

For added security, you can now restrict all Cloudera SSO access (including account administrator access) by contacting Cloudera Support and they can disable or enable the "Cloudera SSO All Login Enabled" setting for the account. Previously, account administrator access could not be restricted. For more information, see Disabling the Cloudera SSO login.

June 16, 2022

This release of the Management Console service introduces the following changes:

Customer managed encryption keys on GCP

By default, a Google-managed encryption key is used to encrypt disks and Cloud SQL instances in Data Lake, FreeIPA, and Data Hub clusters, but you can optionally configure CDP to use a customer-managed encryption key (CMEK) instead. This can only be configured using CDP CLI. There is no UI option available for specifying a GCP CMEK in CDP. For more information, refer to Adding a customer managed encryption key for GCP.

June 7, 2022

This release of the Management Console service introduces the following changes:

Customer managed encryption keys on AWS

By default, Data Lake and FreeIPA's Amazon Elastic Block Store (EBS) volumes and Relational Database Service (RDS) are encrypted using a default key from Amazon’s KMS, but you can optionally configure encryption using Customer Managed Keys (CMK). Data Hubs inherit environment's encryption key by default but you have an option to specify a different CMK during Data Hub creation. For more information, refer to Adding a customer managed encryption key to a CDP environment running on AWS.

Deploying CDP in multiple AWS availability zones

By default, CDP provisions Data Lake, FreeIPA, and Data Hubs in a single AWS availability zone (AZ), but you can optionally choose to deploy them across multiple availability zones (multi-AZ). It is possible to enable it either for all or some of these components. For more information, refer to Deploying CDP in multiple AWS availability zones.

June 3, 2022

This release of the Management Console service introduces the following changes:

SCIM for Azure AD

CDP supports SCIM with Microsoft Azure Active Directory (Azure AD). For more information, see Configure SCIM with Azure AD.

May 26, 2022

This release of the Management Console service introduces the following changes:

New permissions were added to the default cross-account AWS policy

The cross-account access IAM role that is used for the CDP credential has been changed to include a set of new permissions required for Cloudera Data Engineering (CDE), Cloudera DataFlow (CDF), and Cloudera AI. The new AWS permissions are required to simplify the creation of the Kubernetes cluster on AWS. As a result of this change, all customers using or planning to use CDE, CDF, or Cloudera AI in CDP Public Cloud on AWS must update their existing cross-account permissions to ensure that these three data services can be created, enabled, or updated.

If you are using or planning to use CDE, CDF, or Cloudera AI, add the following permissions to the cross-account role:

```
{
"Effect": "Allow",
 "Action": [
```

```
 "ssm:DescribeParameters",
 "ssm:GetParameter",
 "ssm:GetParameters",
 "ssm:GetParameterHistory",
 "ssm:GetParametersByPath"
 ],
 "Resource": [
 "arn:aws:ssm:*:*:parameter/aws/service/eks/optimized-ami/*"
 ]
}
```

May 17, 2022

This release of the Management Console service introduces the following changes:

Data Lake scaling (Preview)

Data Lake scaling is the process of scaling up a light duty Data Lake to the medium duty form factor, which has greater resiliency than light duty and can service a larger number of clients. You can trigger the scale-up in the CDP UI or through the CDP CLI. See Data Lake scaling (Preview).

Note:

You need to contact Cloudera to have this feature enabled.

May 12, 2022

This release of the Management Console service introduces the following changes:

Cloudera Runtime 7.2.15

Cloudera Runtime 7.2.15 is now available and can be used for registering an environment with a 7.2.15 Data Lake and creating Data Hub clusters. For more information about the new Runtime version, see Cloudera Runtime. If you need to upgrade your existing CDP environment, refer to Data Lake upgrade and Data Hub upgrade documentation.

Support for Replication Manager in ap-1 and eu-1 regional Control Planes

Cloudera Replication Manager is now supported in the ap-1 (Australia) and eu-1 (Germany) regional Control Planes.

May 10, 2022

This release of the Management Console service introduces the following changes:

Bring your own Azure private DNS zone

CDP supports using a private endpoint for Azure Postgres with an existing Azure private DNS zone. The private DNS zone can now be pre-created and provided by you, or created by CDP. Previously, CDP always created the private DNS zone when a private endpoint was created.

See updated Azure requirements for using a private endpoint for Azure Postgres and Enabling a private endpoint for Azure Postgres in CDP.

Extended upgrade version support for RAZ-enabled environments

Data Lake major/minor version upgrades for RAZ-enabled environments are now available for Runtime versions 7.2.10-7.2.12 to 7.2.14+.

April 19, 2022

This release of the Management Console service introduces the following changes:

Changed FreeIPA Azure VM type

The Azure VM type used for the FreeIPA server was changed from Standard_D3_v2 to Standard_DS3_v2 so that FreeIPA nodes can be encrypted at host. Standard_D3_v2 doesn't support encryption at host while Standard_DS3_v2 does. See updated Overview of Azure resources used by CDP.

Setting IDBroker mappings in a RAZ environment is disabled

If a CDP environment has RAZ enabled, setting IDBroker mappings is disabled during environment creation and when the environment is already running. If your environment has RAZ enabled, you should be using Ranger for authorizing user and group access to the S3 or ADLS Gen 2 cloud storage used by the Data Lake.

Azure Load Balancer

The Standard SKU Azure Load Balancer is used in multiple places in CDP Data Lakes and Data Hubs. It is used as a frontend for Knox in both Data Lakes and Data Hubs, and for Oozie HA in HA Data Hubs. See Azure Load Balancers in Data Lakes and Data Hubs.

Upgrading classic clusters from CCMv1 to CCMv2

You can now upgrade your CDH, HDP, or CDP Private Cloud Base clusters that were previously registered in CDP from CCMv1 to CCMv2. See Upgrading a classic cluster from CCMv1 to CCMv2.

March 29, 2022

This release of the Management Console service introduces the following changes:

Upgrading FreeIPA

To ensure that your FreeIPA nodes are running with the latest patches, you should periodically upgrade your FreeIPA cluster. CDP currently allows you to upgrade all FreeIPA clusters, updating OS-level security patches on the cluster nodes. See Upgrade FreeIPA.

Upgrading the Data Lake

Major/minor version upgrades of Cloudera Runtime and Cloudera Manager are generally available. Data Lake maintenance upgrades for RAZ-enabled environments versions 7.2.7+ are generally available. For more information see Data Lake upgrade.

March 21, 2022

This release of the Management Console service introduces the following changes:

New classic cluster roles

As part of the new authorization model released in 2021, CDP introduces a new account role and resource roles related to classic clusters:

	Roles	Description
New account role	ClassicClustersCreator	This role is required to register a new classic cluster. If this role is not present then the “Add Cluster” button is not visible to the user.
New resource roles	ClassicClusterAdmin ClassicClusterUser	These roles can be assigned on the scope of a specific classic cluster.

For more information, see Enabling admin and user access to classic clusters.

Data Lake backup and restore options

New CLI options have been added to the Data Lake backup and restore feature. These options allow for explicitly including or skipping certain data during a backup and restore operation:

- You can skip or include the backup/restore of the HMS and Ranger databases.
- You can skip or include the HBase Atlas tables, and all Solr collections except ranger_audit.
- You can skip or include the Solr ranger_audit collection.

For more information, see Configure backups for a Data Lake.

March 7, 2022

This release of the Management Console service introduces the following changes:

Public certificate auto-renewal

Most public (Let's Encrypt-issued) certificates for Data Lake and Data Hub clusters will now auto-renew without intervention from a user. For more information, refer to Managing Certificates.

Data Lake recipes

Support for attaching/detaching recipes on a Data Lake cluster is now available through both the CDP UI and CDP CLI. For more information see Recipes.

February 25, 2022

This release of the Management Console service introduces the following changes:

Cloudera Runtime 7.2.14

Cloudera Runtime 7.2.14 is now available and can be used for registering an environment with a 7.2.14 Data Lake and creating Data Hub clusters. See Cloudera Runtime.

Resource list filtering

CDP users other than PowerUsers and CDP administrators can only list the resources that they are authorized to access. Prior to this change, all CDP users were able to list all resources, but if they tried to access a resource that they were not authorized to access, CDP would return an error.

February 18, 2022

This release of the Management Console service introduces the following changes:

Customer managed encryption keys on Azure

By default, local Data Lake, FreeIPA, and Data Hub disks attached to Azure VMs and the PostgreSQL server instance used by the Data Lake and Data Hubs are encrypted with server-side encryption (SSE) using Platform Managed Keys (PMK), but you can optionally configure SSE with Customer Managed Keys (CMK). For more information, refer to Adding a customer managed encryption key for Azure.

February 14, 2022

This release of the Management Console service introduces the following changes:

Support for CDW in ap-1 regional Control Plane

Cloudera Data Warehouse (CDW) is now supported in the ap-1 (Australia) regional Control Plane. To use CDW in this regional Control Planes, your CDP administrator must create a new environment.

January 31, 2022

This release of the Management Console service introduces the following changes:

Workload password policies

In order to bring your workload password complexity requirements in line with company policy, you can set your FreeIPA password policies via CDP web interface and CDP CLI. Password policies can be configured for length, complexity, expiration, and scope. For more information, refer to Configuring workload password policies.

Pull-based audit archiving

Pull-based audit archiving allows you to pull audit events for archiving purposes without any extra configuration beyond Control Plane API usage. For more information refer to Pull-based audit archiving.

Custom images and catalogs

If necessary, you can use a custom Runtime or FreeIPA image for compliance or security reasons. You can then use the CDP CLI to register a custom image catalog and set the custom image within the custom image catalog. For more information refer to Custom images and catalogs.

Support for CDW in eu-1 regional Control Plane

Cloudera Data Warehouse (CDW) is now supported in the eu-1 (Germany) regional Control Plane. To use CDW in one this Control Plane, your CDP administrator must create a new environment.

2021

December 17, 2021

This release of the Management Console service introduces the following changes:

Fine-grained Access Control for ADLS Gen2 and S3

The fine-grained access control for ADLS Gen2 and S3 cloud storage via the Ranger Authorization Service (RAZ) enables Amazon S3 and ADLS Gen2 users to control access per user and per directory in cloud storage. By specifying Apache Ranger policies for cloud storage, admins can provide home directories and audit capabilities similar to those used with HDFS files in an on-premises or IaaS deployment.

For more information and setup steps refer to:

- Fine grained access control for AWS
- Fine grained access control for Azure

Related Information

Cloudera Management Console Overview

November 22, 2021

This release of the Management Console service introduces the following changes:

AWS Milan region is supported for CDW

Cloudera Data Warehouse (CDW) introduces support for the eu-south-1 (Milan) AWS region. See updated AWS regions supported by CDP.

No-proxy option for non-transparent proxies

When you set up a non-transparent proxy server, you now have the option of configuring specific IP addresses, domains, or subdomains to bypass the proxy. For more information, see Using a non-transparent proxy.

CDW diagnostics collection

You can trigger a diagnostics bundle collection for Cloudera Data Warehouse (CDW). See updated Send a diagnostic bundle to Cloudera Support.

Updated GCP provisioning credential's permissions

A new GCP granular permission is required for creating Data Hubs using the Data Engineering HA template:

```
compute.regionHealthChecks.useReadOnly
```

If your GCP provisioning credential uses a custom IAM role with granular permissions, you should update it to include this permission.

See updated Service account for the provisioning credential.

November 17, 2021

This release of the Management Console service introduces the following changes:

Updated Azure provisioning credential's permissions

The following new Azure permissions are required for the CDP provisioning credential:

```
"Microsoft.Network/loadBalancers/backendAddressPools/join/action",
"Microsoft.Network/loadBalancers/delete",
"Microsoft.Network/loadBalancers/read",
"Microsoft.Network/loadBalancers/write",
```

If you have created a custom role for the CDP provisioning credential, you should update your application registration on Azure, assigning these additional permissions. If you have assigned the built-in Contributor role instead of granular permissions, you do not need to take any action.

Documentation has been updated. See Prerequisites for the provisioning credential.

FreeIPA HA for GCP environments

FreeIPA HA is now supported and used by default for all newly created GCP environments.

November 9, 2021

This release of the Management Console service introduces the following changes:

Cluster Connectivity Manager v2 (CCMv2)

CCMv2 replaces CCMv1. While CCMv1 establishes and uses a tunnel based on the SSH protocol, with CCMv2 the connection is via HTTPS. All new environments created with Runtime 7.2.6 or newer after enabling CCMv2 on your tenant use CCMv2. Existing environments and new environments created with Runtime older than 7.2.6 continue to use CCMv1. All newly registered classic clusters use CCMv2, but previously registered classic clusters continue to use CCMv1. If your CDP tenant has not been granted the CDP_CCM_V2 entitlement yet, it continues to use CCMv1.

The steps to register an environment with CCMv2 are similar to CCMv1 configuration steps. The main differences are:

- If you are deploying in an environment with restricted outbound network access, port 443 needs to be open and new destinations need to be added to the allow list.
- If you are registering a classic cluster, the steps have changed.

For more information, see Cluster Connectivity Manager.

October 26, 2021

This release of the Management Console service introduces the following changes:

Medium duty Data Lakes for GCP

Medium duty Data Lakes for GCP have added an additional gateway node to provide failure resilience for UI and API clients. Load-balanced UI and API access are now available without interruption. For more information see Data Lake scale.

Cloudera Runtime 7.2.12

Cloudera Runtime 7.2.12 is now available and can be used for registering an environment with a 7.2.12 Data Lake and creating Data Hub clusters. See Cloudera Runtime.

September 28, 2021

This release of the Management Console service introduces the following changes:

New GCP permissions for provisioning credential

The list of permissions for the provisioning credential's service account has been updated to include new permission required for load balancing between HA components of the Data Lake. If you are running CDP in GCP, you should update the provisioning credential's service account to include either the Compute Load Balancer Admin (roles/compute.loadBalancerAdmin) IAM role or the following granular permissions:

- compute.addresses.create
- compute.addresses.delete
- compute.addresses.get
- compute.addresses.use
- compute.instanceGroups.create
- compute.instanceGroups.delete
- compute.instanceGroups.get
- compute.instanceGroups.list
- compute.instanceGroups.update
- compute.instanceGroups.use
- compute.forwardingRules.create
- compute.forwardingRules.delete
- compute.forwardingRules.get
- compute.forwardingRules.list
- compute.forwardingRules.setLabels
- compute.forwardingRules.update
- compute.forwardingRules.use
- compute.regionBackendServices.create
- compute.regionBackendServices.delete
- compute.regionBackendServices.get
- compute.regionBackendServices.list
- compute.regionBackendServices.update
- compute.regionBackendServices.use
- compute.regionHealthChecks.create
- compute.regionHealthChecks.delete
- compute.regionHealthChecks.get
- compute.regionHealthChecks.list
- compute.regionHealthChecks.update
- compute.regionHealthChecks.use

See updated Permissions for the provisioning credential's service account.

September 21, 2021

This release of the Management Console service introduces the following changes:

New authorization model

CDP introduces a new authorization model. The following table summarizes new, changed, and deprecated roles. The roles that are not mentioned in this table are unchanged.

Account roles

	Roles	Description
New account role	• EnvironmentCreator	This is a new account-level role.
Deprecated account roles	• EnvironmentAdmin • EnvironmentUser	These roles have been deprecated in June 2020 and have been removed from the official documentation.

Resource roles

	Roles	Description
New environment resource roles	• DataSteward • DataHubCreator	These roles can be assigned on the scope of a specific environment.
New Data Hub resource role	• DataHubAdmin (Technical Preview)	This role can be assigned on the scope of a specific Data Hub.
New shared resource role	• SharedResouceUser	This role can be assigned on the scope of a specific shared resource (cluster template, credential, image catalog, proxy, or recipe).
New resource role applicable to environments, Data Hubs, shared resources, and classic clusters	• Owner	Grants all permissions required to manage the resource in CDP including the ability to delete it, but does not grant any cluster-level access. The user creating the resource automatically gets the Owner role on that resource.

Steps for assigning roles

- The steps for assigning account roles and managing access to environments are unchanged.
- The steps for managing access to Data Hubs, shared resources (cluster templates, credentials, image catalogs, and recipes), and classic clusters are similar to the steps for managing access to environments: You can use the Manage access option from the resource details page.

Updated documentation

- For updated information about all built-in roles in CDP, refer to Understanding account-level roles and resource roles.
- For updated instructions for how to manage access to resources, refer to Assigning a resource role to a user and Assigning a resource role to a group.
- Other new and updated documentation:
  - Enabling admin and user access to environments
  - Enabling admin and user access to Data Hubs

Dots are now supported in group names

The list of supported characters in group names was extended to include dots. See updated documentation:

- Creating a group
- Synchronizing group membership

Improved AWS cloud storage setup documentation

AWS cloud storage setup documentation has been improved to include exact steps for creating the required S3 bucket, IAM policies, and IAM roles. See Minimal setup for cloud storage and Onboarding CDP users and groups for cloud storage .

September 9, 2021

This release of the Management Console service introduces the following changes:

Cloudera Runtime 7.2.11

Cloudera Runtime 7.2.11 is now available and can be used for registering an environment with a 7.2.11 Data Lake and creating Data Hub clusters. See Cloudera Runtime.

Specifying multiple existing AWS security groups

When using your existing security groups for registering an AWS environment in CDP via CDP CLI, you can provide a comma-separated list of multiple security groups for both Knox (securityGroupIdForKnox) and Default (defaultSecurityGroupId). This is a CLI-only feature.

Specifying multiple GCP shared subnets

When using an existing shared VPC for registering a GCP environment in CDP via CDP web interface or CLI, you can specify multiple shared subnets.

Support for Bahrain (me-south-1) AWS region

Registering an environment and provisioning Data Hubs is now supported in the Bahrain (me-south-1) AWS region. See Supported AWS regions.

Updated outbound network access destinations

If you are using Cloudera AI, Data Engineering, or DataFlow data services and have restricted egress access, starting on September 7, 2021, you need to add the following new endpoints to your egress rules:

- *.s3.{eu-west-1, us-west-2, ap-southeast-1}.amazonaws.com
- s3-r-w.{eu-west-1, us-west-2, ap-southeast-1}.amazonaws.com
- *.execute-api.{eu-west-1, us-west-2, ap-southeast-1}.amazonaws.com

The region selected should be the region that is geographically closest to where the environment is deployed.

Customers operating in outbound restricted networks will be unable to download docker images, which will impact creating new clusters. Existing environments deployed in outbound restricted networks may experience operational issues, including limited ability to start, scale and repair the data service clusters.

For more information, see:

- Outbound network access destinations for AWS
- Outbound network access destinations for Azure
- Outbound network access destinations for GCP

August 31, 2021

This release of the Management Console service introduces the following changes:

New Shared Resources navigation menu item

Management options for provisioning credentials, proxies, and Data Hub cluster templates, recipes, and image catalogs can now be easily accessed from the new Shared Resources item in the navigation menu.

August 12, 2021

This release of the Management Console service introduces the following changes:

Send a diagnostic bundle to Cloudera support

CDP introduces a web interface for sending a diagnostic bundle to Cloudera support. Currently diagnostics can be collected for Data Lake, FreeIPA, and Data Hub. See Send a diagnostic bundle to Cloudera support.

August 9, 2021

This release of the Management Console service introduces the following changes:

New permission for GCP Logger service account

In addition to the previously documented permissions, if you would like to use a bucket path (gs://<bucket>/<path>) instead of a bucket (gs://<bucket>) for the Logs or Backups bucket, you should assign the storage.objects.list permission to the custom role. See Minimum setup for cloud storage.

"Create public IPs" is disabled with CCM

The Create public IPs option available on the UI during Azure and GCP environment registration is now disabled by default when CCM is enabled.

July 12, 2021

This release of the Management Console service introduces the following changes:

Updated quick starts

The AWS, Azure, and GCP quick starts have been updated to include the optional FreeIPA Backup Location Base introduced in a recent release.

- AWS quick start
- Azure quick start
- GCP quick start

July 8, 2021

This release of the Management Console service introduces the following changes:

FreeIPA HA repair

FreeIPA HA repair is now available for all newly created AWS and Azure environments in CDP. When running in high-availability mode, the identity management system runs multiple instances of FreeIPA on separate hosts. In case of failure, you can now repair failed hosts using the CDP command-line within one week of a node failing. For more information, see Repair a FreeIPA instance.

"Don't create public IPs" option was renamed

The Don't create public IPs option available during Azure and GCP environment registration was renamed to Create public IPs and is enabled by default.

June 24, 2021

This release of the Management Console service introduces the following changes:

FreeIPA backup location

During AWS, Azure, or GCP environment registration via CDP web interface or CDP CLI, you can optionally specify a separate cloud storage location (FreeIPA "Backup Location Base") for FreeIPA backups. If no separate location is specified, FreeIPA backups are stored in Logs Location Base. For more information, see:

- Minimal AWS setup for cloud storage
- Minimal Azure setup for cloud storage
- Minimal GCP setup for cloud storage

FreeIPA HA enabled by default for new AWS and Azure environments

FreeIPA HA is now enabled for all newly created AWS and Azure environments in CDP. The number of nodes used for the FreeIPA server depends on the Data Lake scale selected: light duty uses 2 nodes and medium duty uses 3. The FreeIPA HA toggle button has been removed from the environment registration UI, but, if needed, it is possible to customize FreeIPA node count when registering an AWS or Azure environment via CDP CLI. For more information, see Managing FreeIPA.

June 21, 2021

This release of the Management Console service introduces the following changes:

Cloudera Runtime 7.2.10

Cloudera Runtime 7.2.10 is now available and can be used for registering an environment with a 7.2.10 Data Lake and creating Data Hub clusters. See Cloudera Runtime.

S3Guard removal

S3Guard is no longer used with newly registered AWS environments using Runtime version 7.2.2 or newer. Consequently, the "Enable S3Guard" environment registration option has been removed and there is no need to create a DynamoDB table for your environment when planing to use Runtime version 7.2.2 or newer. Environments created prior to this change continue to use S3Guard.

June 4, 2021

This release of the Management Console service introduces the following changes:

Updated GCP Quick Start

GCP Quick Start has been updated to include environment registration steps using CDP web interface. See GCP Quick Start.

Updated CCM troubleshooting documentation

CCM troubleshooting documentation has been updated to include information on common cases when connection via CCM fails and steps for collecting information from clusters. See Troubleshooting CCM.

April 29, 2021

This release of the Management Console service introduces the following changes:

Cloudera Runtime 7.2.9

Cloudera Runtime 7.2.9 is now available and can be used for registering an environment with a 7.2.9 Data Lake and creating Data Hub clusters. See Cloudera Runtime.

Registering CDP Private Cloud Base clusters in CDP Public Cloud

You can now register CDP Private Cloud Base clusters as classic clusters in CDP:

- The CDP Private Cloud Base clusters can be registered via Cloudera Manager for use in Replication Manager.
- Additionally, you can register CDP Private Cloud Base clusters via Cloudera Manager and Knox for use in Data Catalog and Replication Manager. This is a technical preview feature that should not be used in a production environment.

For documentation, see Adding a CDP Private Cloud Base cluster.

April 27, 2021

The following updates were made:

CDP Public Cloud onboarding documentation was moved

The following publications were moved to the CDP Public Cloud library:

- Getting Started in CDP Public Cloud
- AWS/Azure/GCP Requirements
- AWS/Azure/GCP Quick Starts
- CDP Public Cloud Security Overview

This CDP Public Cloud library is accessible via the Get Started link on the https://docs.cloudera.com homepage or via the https://docs.cloudera.com/cdp-public-cloud/cloud/index.html link:

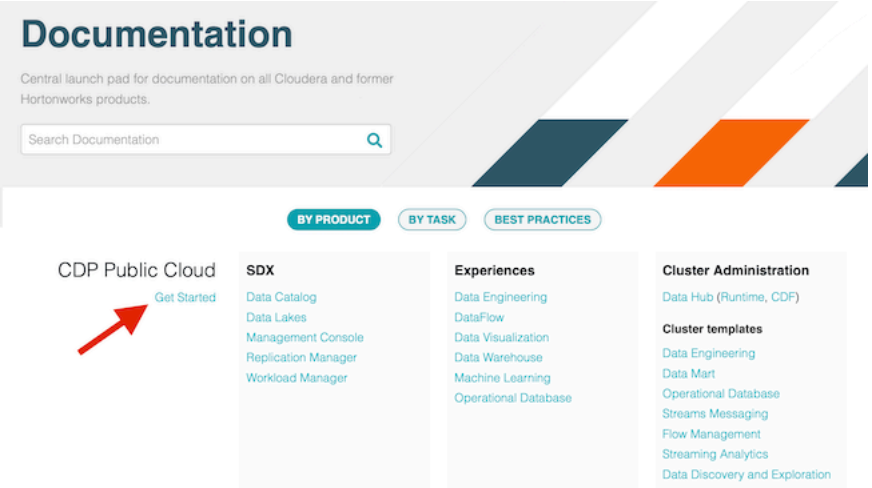

The documentation that was moved is available from the following links:

- https://docs.cloudera.com/cdp/latest/getting-started/index.html
- https://docs.cloudera.com/cdp/latest/aws-quickstart/index.html
- https://docs.cloudera.com/cdp/latest/azure-quickstart/index.html
- https://docs.cloudera.com/cdp/latest/gcp-quickstart/index.html
- https://docs.cloudera.com/cdp/latest/requirements-aws/index.html
- https://docs.cloudera.com/cdp/latest/requirements-azure/index.html
- https://docs.cloudera.com/cdp/latest/requirements-gcp/index.html
- https://docs.cloudera.com/cdp/latest/security-overview/index.html

URL redirects were added temporarily; They will eventually be removed, so make sure to update your bookmarks.

This change was made in effort to make CDP Public Cloud onboarding documentation easier to find. The previous location of this content (in the Management Console library) was unintuitive to many users.

AWS/Azure/GCP planning documentation was consolidated

The AWS/Azure/GCP requirements content was consolidated in one place in the CDP Public Cloud library mentioned above.

If the content that you have bookmarked throws a 404 error, it is most likely in one of the following publications:

- https://docs.cloudera.com/cdp/latest/requirements-aws/index.html
- https://docs.cloudera.com/cdp/latest/requirements-azure/index.html
- https://docs.cloudera.com/cdp/latest/requirements-gcp/index.html

To fix the error, you have three options:

- Update the URL by replacing /management-console/cloud/environments-<cloud-provider>/ with /cdp-public-cloud/cloud/requirements-<cloud-provider>/, replacing <cloud-provider> with "aws", "azure", or "gcp". This works for the content that was moved, but not for topics that were consolidated into other documentation and removed.
- On the https://docs.cloudera.com homepage, search the website for the content that moved. Search results will direct you to the correct location.
- Navigate to one of the libraries linked above and find the content that you are looking for.

This change was made in effort to consolidate all documentation related to cloud provider requirements in one place. Previously, the documentation was scattered and users had to click on many links in order to find content.

April 20, 2021

This release of the Management Console service introduces the following changes:

Cluster Definitions page was moved to environment details

The Cluster Definitions page that used to be available in the Shared Resources section was removed. Instead, you can access all cluster definitions related to a specific environment from the Cluster Definitions tab available in the environment's details. You can save new cluster definitions using the Save As New Definition option available from the Create Data Hub wizard or from CDP CLI using the cdp datahub create-cluster-definition command.

Ranger Audit environment parameter was moved to Data Access section

The option to specify the Ranger Audit role (AWS) managed identity (Azure) or service account (GCP) during environment registration was moved from the Logs - Storage and Audit section to the Data Access section. Consequently, these sections were renamed to Logs and Data Access and Audit.

You can select specific nodes to repair within a Data Lake host group

From the Hardware tab of the Data Lake details, you can click the Repair icon to select specific nodes within a host group to repair.

Updated IAM policy for the provisioning credential for AWS

The IAM policy for the provisioning credential has been updated to include new permissions related to load balancers. The following permissions are now required:

```
cloudformation:UpdateStack
cloudformation:ListStackResources
elasticloadbalancing:DescribeLoadBalancers
elasticloadbalancing:DescribeTargetHealth
elasticloadbalancing:RegisterTargets
elasticloadbalancing:DeregisterTargets
```

If you are using a restricted IAM policy for your provisioning credential, you must add these additional permissions.

April 9, 2020

This release of the Management Console service introduces the following changes:

GCP quick start

GCP quick start is now available, allowing you to quickly set up a CDP environment. See GCP quick start.

March 30, 2021

This release of the Management Console service introduces support for Google Cloud Platform and support for medium duty Data Lakes on Microsoft Azure.

Google Cloud Platform (GCP) support

CDP supports registering GCP environments and creating Data Hub clusters. See the following new and updated documentation:

- Getting Started as an admin in CDP
- Google Cloud requirements
- Introduction to Google Cloud environments
- Register a Google Cloud environment
- Medium duty Data Lakes on Google Cloud
- Default Data Hub configurations available for Google Cloud
- Create a Data Hub cluster on Google Cloud

Medium duty Data Lakes on Microsoft Azure

The medium duty Data Lake configuration is now available for Microsoft Azure. Light duty is still used by default, but you can change this when registering an environment from CDP user interface or when creating a Data Lake from CDP CLI using the --scale MEDIUM_DUTY_HA option. For information about available configurations, see Data Lake scale.

March 25, 2021

This release of the Management Console service introduces the following changes:

Cloudera Runtime 7.2.8

Cloudera Runtime 7.2.8 is now available and can be used for registering an environment with a 7.2.8 Data Lake and creating Data Hub clusters. See Cloudera Runtime.

Medium duty Data Lake for AWS

The medium duty Data Lake configuration is now available for AWS. Light duty is still used by default, but you can change this when registering an environment from CDP user interface or when creating a Data Lake from CDP CLI using the --scale MEDIUM_DUTY_HA option. For information about available configurations, see Data Lake scale.

New supported AWS regions

The eu-south-1 (Europe - Milan) and af-south-1 (Africa - Cape Town) regions are now available as technical preview for creating Data Hub clusters. See updated Supported AWS regions.

Cluster definitions location

Cluster definitions that can be used with a given environment are now listed in the details of that environment. To view them navigate your environment and access the Cluster Definitions tab.

March 19, 2021

This release of the Management Console service introduces the following changes:

Single existing Azure resource group

The option to use your existing Azure resource group is now available in CDP. This allows you to have your credential's role definition scoped to that particular resource group instead of the whole subscription. The option to "Select Resource Group" is available on the UI on the Region, Networking and Security page of the register environment wizard. The corresponding CLI JSON parameter is resourceGroupName and the cdp environment s create-azure-environment CLI option to enable is --resource-group-name <value>. If these parameters are not present, CDP defaults to creating new resource groups. See updated Azure Permissions and Resource group in Azure Requirements documentation, and updated Register an Azure environment.

Private endpoints for Azure database for PostgreSQL

The option to create private endpoints instead of service endpoints for Azure Database for PostgreSQL is now available when registering an Azure environment in CDP. The option to "Create Private Endpoints" is available on the UI in the "Network" section of the register environment wizard. The corresponding CLI JSON parameter is createPrivateEndpoints and the cdp environments create-azure-environment CLI option to enable private endpoints is --create-private-endpoints. The option is disabled by default. It can only be used with the single resource group feature and can only be enabled on subnets that have Azure network policies turned off. See Private endpoint for PostgreSQL.

Public Endpoint Access Gateway for AWS

During AWS environment registration, you can optionally enable Public Endpoint Access Gateway, which provides secure connectivity to UIs and APIs in Data Lake and Data Hub clusters deployed using private networking, allowing

users to access these resources without complex changes to their networking or creating direct connections to cloud provider networks. See Public Endpoint Access Gateway.

Generate CLI command from the UI

You can quickly obtain CDP CLI commands from the CDP web interface for creating the following:

- Environment: From details of an existing environment or from the last page of the register environment wizard. See Obtain CLI commands for registering an environment.
- Data Lake: From details of an existing Data Lake. See Understanding Data Lake details: Show CLI Command.
- Credential: From details of an existing credential or from the create credential wizard. See Obtain CLI commands for creating a credential.
- Data Hub: From details of an existing Data Hub or from the create Data Hub wizard. See Create a Data Hub cluster on AWS from an existing cluster and Create a Data Hub cluster on Azure from an existing cluster.

Disable cloud storage logging for an existing environment

By default, CDP sends collected service logs from VM nodes to the cloud storage that you provided for logs during environment registration. You can disable cloud storage logging for a specific environment, by navigating to environment details > Summary > Telemetry and turning off "Enable Cloud Storage Logging". Disabling this option will affect only newly created Data Hub clusters in that environment. See Enabling environment telemetry>Disable cloud storage logging for an existing environment.

Obtain your CDP tenant ID

You can now obtain your CDP tenant ID from CDP web interface,. See Obtain your CDP tenant ID.

February 25, 2021

This release of the Management Console service introduces the following new features and behavioral changes:

Cloudera Runtime 7.2.7

Cloudera Runtime 7.2.7 is now available and can be used for registering an environment with a 7.2.7 Data Lake and creating Data Hub clusters. See Cloudera Runtime.

User delete

CDP administrators now have the ability to delete users in CDP through both the user interface and the CLI. Deleting a user removes all access keys and SSH keys associated with the user, and unassigns all roles and resource roles assigned to the user. The user is also removed from all groups that they belong to. For more information, refer to Deleting users and machine users.

FreeIPA HA

CDP administrators can configure your CDP environment to run FreeIPA in high-availability mode. By default, creating an environment configures a single instance of FreeIPA on its own host, but you can explicitly enable FreeIPA HA during environment registration via CPD web UI or CLI. For more information, refer to Managing FreeIPA.

Interactive login for CDP CLI and CDP SDK

If you would prefer that user access to the CLI/SDK is shorter-lived, you can use the "interactive" method of logging into the CDP CLI/SDK. By default, this login method grants a 12-hour access key to the CLI/SDK. The access key will time out after one hour of inactivity. The interactive method integrates with any SAML-compliant external identity provider. For more information, refer to Logging into the CDP CLI/SDK.

Anonymization rules

CDP includes a set of default anonymization rules and allows CDP administrators to define custom anonymization rules in order to remove sensitive information from CDP logs. For more information, refer to Defining anonymization rules for CDP logs.

Changes to delete machine user behavior

Deleting a machine user removes all access keys and SSH keys associated with the machine user, and unassigns all roles and resource roles assigned to the machine user. The machine user is also removed from all groups that they belong to. Previously, these steps had to be performed manually prior to machine user deletion. It takes around 2 minutes to fully delete a machine user in CDP. During that time you will not be able to recreate the machine user (that is, for 2 minutes you will not be able to create a machine user with the same machine user name).

Group name length limit

CDP user management framework supports group names of up to 64 characters. Previously up to 32 characters were supported.

Identity provider configuration improvements

The user interface and the overall flow of the identity provider configuration in CDP was improved for better usability.

New CDP SAML Service Provider certificate

The current CDP SAML Service Provider certificate is expiring on March 8, 2021 at 18:05:49 GMT. A replacement certificate is available for any customer whose identity provider will verify the CDP SAML service provider certificate. You can obtain the certificate from this document or by logging it to CDP web interface, navigating to > User Management > Identity Providers, clicking on your identity provider, and the last field "CDP SAML Service Provider Metadata" now contains 2 certificates: the one that expires on March 8, 2021 and the new one. Please consult your identity provider's documentation for how to update service provider certificates. CDP will start using the new certificate for SAML starting March 8, 2021.

Here is the new CDP SAML Service Provider certificate:


```
-----BEGIN CERTIFICATE-----
MIIEKTCCAxGgAwIBAgIUF7LjOby+L8dcCVzWN4ChnTtybiowDQYJKoZIhvcNAQEL
BQAwgaMxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTEWMBQGA1UEBwwNU2FuIEZy
YW5jaXNjbzEVMBMGA1UECgwMQ2xvdWRlcmEgSW5jMRAwDgYDVQQLDAdDRFAgSUFN
MSEwHwYDVQQDDBhjb25zb2xlLmNkcC5jbG91ZGVyYS5jb20xIzAhBgkqhkiG9w0B
CQEWFHN1cHBvcnRAY2xvdWRlcmEuY29tMB4XDTIxMDIyMzE5NDgxMVoXDTI0MDIy
ODE5NDgxMVowgaMxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTEWMBQGA1UEBwwN
U2FuIEZyYW5jaXNjbzEVMBMGA1UECgwMQ2xvdWRlcmEgSW5jMRAwDgYDVQQLDAdD
RFAgSUFNMSEwHwYDVQQDDBhjb25zb2xlLmNkcC5jbG91ZGVyYS5jb20xIzAhBgkq
hkiG9w0BCQEWFHN1cHBvcnRAY2xvdWRlcmEuY29tMIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEAszIxvwxxsAE4PqNLfZ2+4zfYI9UpiiePEOKJuL1Q8Mbh
ArA53EmZradpYNIQ54a3vGQNeEoi782gcp/JbzLTY0AESnKXzpPXOhX8qMWytrcL
QKmSW/eVbZsVEYnyf1wFxtpOcLbHfYB12W1ScD3FKr5BUns6bRCclfiFW1Ei5XLQ
yzgSGdKXSvB/8izRr4yyyDT2IX8PelHbONiIKb6OTuuHPwo259RMjZZd2pwMurif
JUGBckwYPh7Dkmiw9mTXVSD5fdSP1HvP2RTuPqmkTSgJRwdJD4G6wF1NFOQwItIr
7vf6OzPZJM6A2JCN8RQApMnYyNgT75wWtCNOF8F2cQIDAQABo1MwUTAdBgNVHQ4E
FgQUGfVSdXrVb3JsJy5nf4OYp2sJn8IwHwYDVR0jBBgwFoAUGfVSdXrVb3JsJy5n
f4OYp2sJn8IwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAkNxk
+X2sCbXAIhSUNKUYQEM++ZDSnWzMgdavNeVUzWgTfGdwvDolFzvqU66wiQ8kedK0
qLW6gRZkG+GJUq5vY93pfNSQ5C4P9hhFqpd6tfHme7uHlZCtZh/wjOeYoOpgr0eI
qtXxg6U6+6qLqzBi/9Zdc0sLZFNbjQLEFkNHoU7rFODcnLNHemngw+ui2rofsBhK
F9Zcqiy91mmCto6OrQMAkXyfrU40S8+Yr9s+wnJEmNIkVN9mfH0TfRJNEvHcuvZ+
WHc4HD/Vu0sL/APPADfLh158MYb9gUNXtE12PxjGYCj4RsFt0/Fbju9mGl+W/n69
qaRFxZmubutaQ1WCzw==
```

```
-----END CERTIFICATE-----
```

CDP CLI reference

CDP CLI reference documentation is now available at https://cloudera.github.io/cdp-dev-docs/cli-docs/.

Documentation for configure lifecycle management for logs on AWS and Azur

To avoid unnecessary costs related to Amazon S3 pr ADLS Gen2 cloud storage, you should create lifecycle management rules for your cloud storage location used by CDP for storing logs so that these logs get deleted once they are no longer useful. See Configure lifecycle management for logs on AWS and Configure lifecycle management for logs on Azure.

Consolidated documentation for restricting admin and end user access for CDP services

Consolidated documentation for restricting admin and end user access for CDP services is now available. Previously these options were only covered in documentation related to specific CDP workload services. See Restricting access for CDP services that create their own security groups on AWS and Restricting access for CDP services that create their own security groups on Azure.

Updated AWS and Azure requirements documentation

AWS and Azure requirements documentation was updated to include more requirements related to Data Engineering, Data Warehouse, and Cloudera AI. These requirements were previously only documented in service-specific docs. See updated AWS Requirements and Azure Requirements.

January 19, 2021

This release of the Management Console service introduces the following new features and behavioral changes:

Control Plane audit archiving

You can configure Control Plane audit archiving from CDP web interface. Previously, this feature was only available via CDP CLI. For updated documentation, refer to AWS setup for audit archiving and Azure setup for audit archiving.

New documentation for pre-creating ADLS Gen2 account for storing OS images

CDP uses an ADLS Gen2 storage account for storing images used for VMs. By default, CDP creates this account during environment registration, but you can optionally pre-create it. If needed, you can also copy the VHD files and create image resources manually. For instructions on how to do this, refer to ADLS Gen2 account for storing images.

January 6, 2021

This release of the Management Console service introduces the following new features and behavioral changes:

Python 3.6 is required for CDP CLI

Starting in January 2021, the CDP CLI requires Python 3.6 for execution. Python 2 reached end of life on January 1, 2020 and is no longer receiving updates. While your existing CDP CLI installation running on Python 2 will keep working, new features and bug fixes are available only for CDP CLI installations running on Python 3.6. To check your Python version, use:

python -V

If you need to install Python 3, you can:

- Visit https://www.python.org/downloads/
- Use a package manager for your OS
- Use a Python management tool

Once you have Python 3, install CDP client by following the usual CDP client installation instructions.

2020

December 21, 2020

This release of the Management Console service introduces the following new features and behavioral changes:

New functionality

- Data Lake metadata backup and restore: You can backup and restore the metadata maintained in the Data Lake services. Data Lake backup and restore is supported from Cloudera Runtime 7.2.1+ on AWS and Cloudera Runtime 7.2.2+ on Azure. See Backup and restore for the Data Lake.

Changed functionality

- The option to set tenant-level tags was moved from Environments > Shared Resources > Tags to a new location under Global Settings > Tags.
- The option to enable workload analytics and deployments cluster logs collection on a tenant-level was moved from Environments > Shared Resources > Telemetry to a new location under Global Settings > Telemetry.

New documentation

- Azure resources used by CDP

Updated documentation

- Azure permissions
- Renew host certificates on Data Lake and Data Hub clusters

Deprecated functionality

The following account roles are deprecated:

- EnvironmentAdmin
- EnvironmentUser

November 20, 2020

This release of the Management Console service introduces the following new features and behavioral changes:

New documentation

- AWS outbound network access destinations
- Azure outbound network access destinations
- Reserved group names
- VM-based on demand diagnostics (technical preview)

Updated documentation

- Updated requirements for Azure Database for PostgreSQL
- IAM role and managed identity setup for access to cloud storage was updated to grant the IDBROKER_ROLE or Assumer identity the same permissions as the LOG_ROLE or Logger identity, allowing IDBroker to write to logs storage location. See updated Minimal setup for cloud storage (AWS) and Minimal setup for cloud storage (Azure).
- CDP CLI, API, and SDK documentation was moved out of the Management Console library and is now available from the general CDP Public Cloud library. See CDP APIs, CDP CLI and CDP SDK.

October 27, 2020

This release of the Management Console service introduces the following new features and behavioral changes:

New features

- Renew host certificates on Data Lake and Data Hub clusters

New documentation

- Prerequisites for adding classic clusters with a non-transparent proxy
- Creating a machine user in CDP

Updated documentation

- Register an AWS environment - Reflects updated environment wizard
- Register an Azure environment - Reflects updated environment wizard

September 24, 2020

This release of the Management Console service introduces the following new features and behavioral changes:

New features

- cdpcurl utility: The cdpcurl utility has been released to open source. This provides curl-like access to the Management Console API as another alternative to the CDP CLI.

Behavioral changes

- The option to stop and restart a Data Lake was removed. You should stop and restart your whole environment instead, as described in Stop and restart an environment.
- The option to delete a Data Lake was removed from the Actions menu on the environment details page. When deleting an environment from the CDP UI, you no longer need to delete the environment's Data Lake separately. See updated Delete an environment documentation.
- The option to Check for Data Lake Upgrade was removed.
- The options to Enable Workload Analytics and Enable Cluster Logs Collection were removed from the Actions menu on the environments details page. You can now control these options from the Management Console > Environments > Shared Resources > Telemetry (tenant-level setting), or you can set them during environment registration (environment-level setting). For updated instructions, see Enable workload analytics and cluster logs collection.
- The option to Get FreeIPA certificate were removed from the environment's Actions menu and is now available from FreeIPA's Actions menu. To find the option, navigate to the Management Console > Environments > navigate to a specific environment > Summary > FreeIPA > Actions > Get FreeIPA certificate.

New documentation

- Performing user sync
- Configuring Azure Active Directory identity federation in CDP
- VNet and subnet planning for Azure
- Supported AWS regions includes an updated list of AWS regions where Cloudera AI can be deployed
- Supported Azure regions includes an updated list of Azure regions where Cloudera AI can be deployed
- Default security group settings for Azure

August 24, 2020

This release of the Management Console service introduces the following new features:

SSH key management

A Power User can add and delete SSH keys for all users and users can add and delete their own SSH keys. For more information, see Managing SSH keys.

Choose Data Lake version at deployment

When deploying an environment, you can now select the Data Lake version. See updated instructions for Register an AWS environment and Register an Azure environment..

Edit parameters of a previously deployed environment

You can edit the following environment components:

- Add new subnets: See Add subnets to an environment.
- Add new security groups: See Add security groups to an environment.
- Add a new root SSH key: See Add root SSH key to an environment.

July 31, 2020

This release of the Management Console service introduces the following new features:

CDP Control Plane Public API Reference

CDP Control Plane now features reference information for APIs located at CDP Control Plane Public API Documentation.You can use this reference documentation to look up descriptions and syntax for SDK functions and CLI commands.

June 30, 2020

This release of the Management Console service introduces the following new features:

Groups sync disabled by default

When you create a Group in CDP, the Sync Membership flag is now off by default. This will eliminate the situation when users would be removed from a CDP group if they weren't part of the same group in the ID Provider.

Support for HDP 2.6.x Classic Clusters using CCM

Support for registering Classic clusters using the reverse SSH mechanism is extended to legacy HDP 2.6.x clusters. This will be behind the CLASSIC_CLUSTERS entitlement.

Auditing Control Plane activity

Auditing is used to collect or log evidence of activity in a system that auditors can use to both track and analyze to answer questions such as: Who made a change to the system? When did a change happen? What exactly changed? Why was a change authorized?

May 30, 2020

This release of the Management Console service introduces the following new features:

Edit subnets

You can now add new subnets to the configuration of a running environment. This might be useful when you add a new service such as Cloudera AI and Clouder Data Warehouse (CDW) which require their own subnet.

Improvements in Data Hub cluster scaling

A number of significant improvements were made to properly clean up cloud resources for scaling up and down Data Hub clusters. This will reduce cloud costs that would have been incurred by any orphaned instances.

April 27, 2020

This release of the Management Console service introduces the following new features:

Support for connecting to private subnets

CDP supports communicating with Data Lake and Data Hub workload clusters that are on private subnets. environments in Azure.

You can use Cluster Connectivity Manager (CCM) to communicate with Data Lake and Data Hub workload clusters that are on private subnets. Communication takes place over private IPs without any inbound network access rules required. CDP requires that these clusters have outbound connections to AWS NLBs hosted in Cloudera's AWS account. Workload clusters initiate an SSH tunnel to the CDP control plane, which is then used for all communication thereafter.

Proxy server (technical preview)

Proxy server works with DataLake and Datahub in both transparent and non-transparent mode. The proxy server works with Cloudera AI and Cloudera Data Warehouse (CDW) in transparent mode only. Proxy server does not currently work with Cloudera AI and CDW in non-transparent proxy mode.

You can use a proxy server to control the connections that are allowed from your VPC or VNet and prevent unattended connections initiated from your environment.

April 6, 2020

This release of the Management Console service introduces the following new features:

Tagging CDP resources

When you create an environment or other resources shared across your cloud provider account, CDP automatically adds default tags to the Cloudera-created resources in your cloud provider account. You can also define additional custom tags at either the environment level or tenant level.

February 28, 2020

This release of the Management Console service introduces the following new features:

Creating environments in Azure

CDP supports creating environments in Azure.

Register your Azure account and then launch Data Hub clusters, Data Warehouse clusters, and Cloudera AI workbenches within the Azure environment and determine which users have access to which resources. These clusters are attached to a Data Lake that runs within the environment and provides security and governance for all attached clusters.

2019

December 19, 2019

This release of the Management Console service introduces the following new features:

Specifying multiple CIDRs on security groups

CDP supports specifying multiple comma-separated CIDRs during environment registration under Security Access Settings > Access CIDR.

Enabling workload analytics for Data Hub clusters

For each environment, you can manually enable and disable workload analytics so that diagnostic information about job and query execution is sent to Workload Manager for Data Hub clusters created within this environment. The option is available during environment creation under Logs Storage and Audits > Enable Workload Analytics. You can also update it once the environment is running by navigating to environment details > Actions > Enable/Disable Workload Analytics:

IAM role selection

When providing IAM instance profiles or IAM roles required for environment's Logs Storage and Audits and Data Access configuration, you can now select from available instance profiles and roles instead of manually providing IAM role ARNs.

November 14, 2019

This release of the Management Console service introduces the following new features:

Wire encryption

CDP now offers TLS encryption across all endpoints for both data and control traffic, providing an internal Certificate Authority (CA) with commonly trusted certificates and automatic certificate expiry warnings. Adding new cluster hosts or services to a cluster with auto-TLS enabled automatically (which is CDP's default setting) creates and deploys the required certificates. CDP supports TLS 1.2.

Existing DynamoDB table

When configuring S3Guard as part of environment creation, you can now specify an existing DynamoDB table. Note that:

- The table must be in the same region as the environment
- One table can be used with one environment only
- CDP will not delete this table when deleting the environment

Removal of environment groups

There is no longer need to create the cdp_$env group for each newly created environment and assign it to the environment.

Consistent FreeIPA password across all environments

There is no longer need to reset the FreeIPA password after creating a new environment. The password is automatically propagated to each newly created environment. You can still manage your FreeIPA password from the User Management page and reset it if needed.

Knox uses port 443 instead of 8443

Knox gateway now uses port 443. If you select for CDP to create security groups automatically, this port is automatically open to your organization's CIDR as needed. If you are creating your environment's security groups manually, you should open port 443 instead of 8443 to your organization's CIDR.

September 23, 2019

This release of the Management Console service introduces the following new features:

Adding or updating IDBroker mappings on a running environment

To add or update the IDBroker mappings on a running environment, navigate to Environments > select an environment > Actions > Manage Access > IDBroker Mappings. For more information, refer to Editing IAM role to S3 mappings.

Retry option for Data Lake

When stack provisioning or Data Lake cluster creation fails, the retry option to resume the process from the last failed step. For more information, refer to Retry a Data Lake.

August 22, 2019

This is the first release of the Management Console service. For an overview of Management Console functionality, refer to:

Log4j vulnerabilities

Cloudera has released hotfixes for Public Cloud Runtime versions 7.2.7 and newer that address Log4j2 vulnerabilities.

The following vulnerabilities have been addressed for Public Cloud Runtime versions 7.2.7 through 7.2.12:

CVE-2021-44228

CVE-2021-45046

LOGBACK-1591

CVE-2021-45105

CVE-2021-44832

You should upgrade your CDP services running Runtime versions 7.2.7+ so that they include the latest hotfixes. You can update your existing Data Lake and Data Hubs by doing a maintenance upgrade. You should first upgrade the Data Lake and then upgrade all the Data Hubs that are using the Data Lake. Refer to Data Lake upgrade and Data Hub upgrade documentation. Data Lake and Data Hub maintenance upgrade is supported only in technical preview for maintenance upgrades from Runtime versions 7.2.7 and higher.

If you are running a version of Runtime prior to 7.2.7, contact Cloudera Support for details on how to upgrade Runtime.

For more information about these hotfixes, refer to the CDP Public Cloud Runtime Release Notes for the version of Runtime that you use.

Image catalog updates

This section lists prewarmed image updates used to Data Lakes, Cloudera Data Hub clusters and FreeIPA.

The following prewarmed images were added on the following dates:

First released	Platform	Cloudera Runtime component versions	Cloudera Manager version	Available OSs
April 3, 2024	Cloudera Runtime7.2.18	7.2.18 Component Versions	Cloudera Manager 7.12.0	RHEL 8.8 EUS
June 27, 2023	Cloudera Runtime 7.2.17	7.2.17 Component Versions	Cloudera Manager 7.11.0	RHEL 8.8 EUS, CentOS 7
January 11, 2023	Cloudera Runtime 7.2.16	7.2.16 Component Versions	Cloudera Manager 7.9.0	CentOS 7
May 12, 2022	Cloudera Runtime 7.2.15	7.2.15 Component Versions	Cloudera Manager 7.6.2	CentOS 7
February 25, 2022	Cloudera Runtime 7.2.14	7.2.14 Component Versions	Cloudera Manager 7.6.0	CentOS 7
October 25, 2021	Cloudera Runtime 7.2.12	7.2.12 Component Versions	Cloudera Manager 7.5.2	CentOS 7
September 9, 2021	Cloudera Runtime 7.2.11	7.2.11 Component Versions	Cloudera Manager 7.4.3	CentOS 7
June 14, 2021	Cloudera Runtime 7.2.10	7.2.10 Component Versions	Cloudera Manager 7.4.2	CentOS 7
April 30, 2021	Cloudera Runtime 7.2.9	7.2.9 Component Versions	Cloudera Manager 7.4.1	CentOS 7
March 25, 2021	Cloudera Runtime 7.2.8	7.2.8 Component Versions	Cloudera Manager 7.4.0	CentOS 7

First released	Platform	Cloudera Runtime component versions	Cloudera Manager version	Available OSs
February 9, 2021	Cloudera Runtime 7.2.7	7.2.7 Component Versions	Cloudera Manager 7.3.0	CentOS 7
December 11, 2020	Cloudera Runtime 7.2.6	7.2.6 Component Versions	Cloudera Manager 7.2.6	CentOS 7
October 8, 2020	Cloudera Runtime 7.2.2	7.2.2 Component Versions	Cloudera Manager 7.2.2	CentOS 7
July 31, 2020	Cloudera Runtime 7.2.1	7.2.1 Component Versions	Cloudera Manager 7.2.1	CentOS 7
June 16, 2020	Cloudera Runtime 7.2.0	7.2.0 Component Versions	Cloudera Manager 7.2.0	CentOS 7
February 20, 2020	Cloudera Runtime 7.1.0	7.1.0 Component Versions	Cloudera Manager 7.1.0	CentOS 7

For the latest image catalog updates, refer to Cloudera web interface > Cloudera Management Console > Shared Resources > Image Catalogs.

Known issues for Cloudera Management Console

This section lists known issues that you might run into while using the Cloudera Management Console service.

CDPD-50436 Kudu service user not authorized to access Hive warehouse in RAZ enabled AWS cluster

Problem: The Kudu service user is not authorized to access Hive warehouse locations in a RAZ enabled AWS cluster on cloud object stores, which under certain conditions can prevent Kudu tables from being created. This results in the following error:

```
ImpalaRuntimeException: Error creating Kudu table 'default.truckspeedevents'
CAUSED BY: NonRecoverableException: failed to create HMS catalog entry for
table  [id=b764bceb167746b7bb3dc1e8722e66e6]: failed to create Hive MetaStor
e table: TException - service has thrown: MetaException(message=Got exceptio
n: java.nio.file.AccessDeniedException
```

Workaround: Add "kudu" service user to the allow list for "Default: Hive warehouse locations" in the cm_s3 Ranger repository in your S3 cloud storage.

ENGESC-19520 Cloudera can't validate permissions

Problem: If in your AWS organization you have Service Control Policies (SCPs) where certain regions are enabled or disabled, Cloudera can't validate the permissions correctly due to an AWS SDK limitation.

Workaround: Disable the permission validations.

OPSAPS-64580 installation is failing because of failure in Install Longhorn step

Problem: When using Cloudera Manager to install , the installation may fail due to a failure in the Longhorn install step. You will see an error similar to the following in the stderr output:

```
++ openssl passwd -stdin -apr1 + echo 'cm-longhorn:$apr1$gp2nrbtq$1KYPGI0QNl
FJ2lo5sV62l0' + kubectl -n longhorn-system create secret generic basic-auth
--from-file=auth + rm -f auth + kubectl -n longhorn-system apply -f /opt/clo
udera/cm-agent/service/ecs/longhorn-ingress.yaml Error from server (Internal
Error): error when creating "/opt/cloudera/cm-agent/service/ecs/longhorn-ing
ress.yaml": Internal error occurred: failed calling webhook "validate.nginx.
ingress.kubernetes.io": Post "https://rke2-ingress-nginx-controller-admissio
n.kube-system.svc:443/networking/v1/ingresses?timeout=10s": x509: certificat
e signed by unknown authority
```

Workaround: Retry the installation by clicking the Resume button.

OPSX-735 Kerberos service may report errors if Cloudera Manager is not running

Problem: The Cloudera Manager Server in the base cluster must be running in order to generate Kerberos principals for Cloudera Private Cloud. If there is downtime, you may observe Kerberos-related errors.

Workaround: Resolve downtime on Cloudera Manager. If you encountered Kerberos errors, you can retry the operation (such as retrying creation of the Virtual Warehouse).

OPSX-1405 Multiple users creating the same environment may result in an unusable environment

Problem: If two users try to create an environment with the same name at the same time, it might result in an unusable environment.

Workaround: Delete the existing environment and try again with only one user creating the environment.

OPSX-1412 Creating an Environment may fail

When multiple users try to create the same environment at the same time or use automation to create an environment with retries, environment creation may fail on collision with a previous request to create an environment.

Workaround: Delete the existing environment, wait 5 minutes, and try again.

OPSX-2062 Platform not shown on the Compute Cluster tab in Environments

Problem: In the Cloudera Management Console Environments page the platform does not display on the Compute Clusters tab.

OPSX-2713 ECS Installation fails to perform First Run of services.

Problem: If an issue is encountered during the Install Cloudera Control Plane step of Containerized Cluster First Run, installation will be re-attempted infinitely rather than the command failing.

Workaround: Since the control plane is installed and uninstalled in a continuous cycle, it is often possible to address the cause of the failure while the command is still running, at which point the next attempted installation should succeed. If this is not successful, abort the First Run command, delete the Containerized Cluster, address the cause of the failure, and retry from the beginning of the Add Cluster wizard. Any nodes that are re-used must be cleaned before re-attempting installation.

OPSX-2772 Unable to modify permissions for Account Administrator

Problem: When a user with administrative privileges accesses the User Management Update Roles page in the Cloudera Management Console, the user is presented with options to select various roles. Selecting or deselecting these roles does not change this user's privileges -- an administrative user, by default, has all privileges, and those privileges cannot be changed.

CB-11925: Knox Load Balancer API requests initiated from Knox Gateway hosts can fail with Connection timeout error

Problem: When logged into a Data Lake or Cloudera Data Hub node that has a Knox Gateway service instance configured on it, making Knox API calls through the Knox load balancer can result in a connection timeout error. This is because for security reasons, the IP address of the request is preserved in the traffic passed through the load balancer. Preserving the IP address means that the load balancer will reject "loopback" traffic, meaning traffic that originates and is directed back to the same node.

Workaround: If Knox API calls need to be made while logged into a Knox gateway node, use the hostname of the node instead of the load balancer hostname in the API call.

The Knox load balancer hostname can be identified by the "-gateway" suffix in the first clause of the hostname with no numeric identifier. For example: <cluster-name>-gateway.<env-shortname>.<hash>.cloudera.site is the load balancer hostname, and <cluster-name>-gateway0.<env-shortname>.<hash>.cloudera.site is a direct node hostname.

OPSAPS-59129: Cloudera Manager reports the HDFS warning "Secure DataNode configuration is valid, but not recommended."

Problem: On Cloudera Runtime versions 7.2.9 and earlier, Data Lake clusters that include the HDFS service show this warning in Cloudera Manager: "Secure DataNode configuration is valid, but not recommended." This warning is benign and can be ignored.

CDPSDX-2879 Ranger import fails when you create a Hive replication policy for a medium duty Data Lake cluster

Problem: When you create a Hive replication policy with the Include Sentry Permissions with Metadata or Skip URI Privileges option for a medium duty Data Lake cluster, Ranger import fails. Before you choose the Include Sentry Permissions with Metadata option for a Hive replication policy for a medium duty Data Lake cluster, contact Cloudera Data Platform Support.

CB-10535, CB-10372 Generate CLI command for existing environment should show 3 commands instead of 1

Problem: If you try to obtain the CDP CLI commands from an existing environment > Actions > Show CLI commands, only the create environment command is displayed instead of all three commands required for registering an environment from CDP CLI.

Workaround: You can obtain the command for creating a Data Lake from Data Lake details. The command to obtain the set IDbroker mappings can be obtained from an existing environment or from CDP CLI help, but you need to modify it manually to set the same mappings as in the source environment.

CB-10706 SSO is not working for Solr/Namenode UI links

Problem: SSO login to an environment with a medium duty Data Lake breaks access to Solr and Namenode UI links.

Workaround: After you deploy a medium duty Data Lake, login to Gateway0 and run:

```
openssl rand -base64 12
```

Then login as root to both gateway nodes and run:

```
export KNOX_GATEWAY_DATA_DIR=/var/lib/knox/gateway/data
```

```
/opt/cloudera/parcels/CDH/lib/knox/bin/knoxcli.sh create-alias pac4j.passwor
d --cluster knoxsso --value “the value from above"
```

Then in Cloudera Manager, restart the Knox service.

DWX-6635 Tags are not being added to S3 buckets

Problem: S3 buckets that are part of your AWS environment registered in Cloudera are not being tagged during environment registration. This is because the PutBucketTagging policy is missing from the cross-accunt IAM role that Cloudera requires you to create for your environment’s credential.

Workaround: You can:

- Manually add tags to your S3 buckets used for existing environments.
- Add the PutBucketTagging policy to the IAM role used for your provisioning credential so that any environments registered in Cloudera in the future can automatically add S3 bucket tags.

CB-6924 Workaround for ZooKeeper external volume bug

Problem: In the current version of Cloudera, ZooKeeper might be configured to write to Cloudera's root disk which is too small to accommodate the ZooKeeper data. To correct this issue, you need to reconfigure ZooKeeper to write to an external volume and move any ZooKeeper data to that volume.

Workaround: To check if ZooKeeper is configured to use an external volume, complete the following:

1. Open ZooKeeper and navigate to: ZooKeeper menu item -> Configuration tab -> Filter to Server.
2. If the dataDir and dataLogDir fields contain /hadoopfs/fs1/zookeeper you do not need to do anything.
3. If the fields contain any other values, you must reconfigure ZooKeeper.

To reconfigure ZooKeeper, complete the following:

1. ssh into the machine where the ZooKeeper server is running .
2. Run the following command to change the user:

   sudo -su zookeeper
3. Run the following command:

   cp -R /var/lib/zookeeper/ /hadoopfs/fs1/zookeeper
4. Open the cluster from the Cloudbreak user interface.
5. Log into the Cloudera Manager user interface.
6. Find ZooKeeper on the Cloudera Manager page and navigate to the configuration with either the Search box or select it from the side menu: ZooKeeper menu item -> Configuration tab -> Filter to Server.
7. Change the following properties:

   ```
   dataDir: /hadoopfs/fs1/zookeeper
   dataLogDir: /hadoopfs/fs1/zookeeper
   ```
8. Save your changes
9. Restart the Stale configuration

   You do not need to redeploy ZooKeeper.

CB-3876 Cloudera Data Warehouse and Cloudera AI create security groups

Problem: When during environment registration you choose to use your own security groups, the Cloudera Data Warehouse and Cloudera AI services do not use these security groups but create their own.

Workaround: For instructions on how to restrict access on the security groups created by the Cloudera Data Warehouse service, refer to Restricting access to endpoints in AWS environments.

CRB-971 Cloudera Data Warehouse creates IAM, S3, and DynamoDB resources

Problem: The Cloudera Data Warehouse service creates its own S3 buckets, DynamoDB tables, and IAM roles and policies. It does not use the environment's S3 bucket(s), DynamoDB table, and IAM roles and policies.

Workaround: There is no workaround.

CB-4176 Data Lake cluster repair fails after manual stop

Problem: Data Lake cluster repair fails after an instance has been stopped manually via AWS console or AWS CLI.

Workaround: After stopping a cluster instance manually, restart it manually via the AWS console or AWS CLI, and then use the Sync option in Cloudera to sync instance state.

CB-2813 Environment with Cloudera AI workbenches in it can be deleted

Problem: When deleting an environment that uses a customer-created VPC and subnets, there is no mechanism in place to check for any existing Cloudera AI workbenches running within the environment. As a result, an environment can be deleted when Cloudera AI workbenches are currently running in it.

Workaround: If using an environment created within your existing VPC and subnets, prior to deleting an environment, ensure that there are no Cloudera AI workbenches running within the environment.

CB-3459 Subnet dependency error when deleting an environment

Problem: When deleting an environment that uses a VPC and subnets created by Cloudera, the environment deletion fails with an error similar to:

```
com.sequenceiq.cloudbreak.cloud.exception.CloudConnectorException: AWS Cloud
Formation stack reached an error state: DELETE_FAILED reason: The subnet 'su
bnet-05606fd72fda58c8c' has dependencies and cannot be deleted. (Service: Am
azonEC2; Status Code: 400; Error Code: DependencyViolation; Request ID: da9a
7fe0-ac43-467e-9942-94f10e6bd2b7)
```

This error occurs if there are resources such as instances used for Cloudera Data Warehouse, or Cloudera AI cluster nodes that were not deleted prior to environment termination.

Workaround: Prior to terminating an environment, you must terminate all clusters running within that environment.

CB-4248 Expired certificate causes untrusted connection warning

Problem: Cloudera automatically generates an SSL certificate for every Data Lake and Cloudera Data Hub cluster. There are two possibilities:

- By default, Cloudera generates a trusted certificate valid for 3 months.
- If generating a trusted certificate fails, Cloudera generates a self-signed certificate valid for 2 years.

In the first case, if your cluster stays active for over 3 months, the trusted certificate will expire and you will see an "untrusted connection", warning when trying to access cluster UIs from your browser.

Workaround: To fix this, you should generate a new certificate by using the following steps:

1. Use the Renew certificate UI option:
   - For Data Lake - Click the Renew certificate button on the Data Lake details page.
   - For Cloudera Data Hub - Click Actions > Renew certificate on Cloudera Data Hub cluster details page.

   During certificate renewal, several related messages will be written to Event History. Once the certificate renewal has been completed, the following message appears: "Renewal of the cluster's certificate finished."
2. Additionally, if your cluster was created prior to December 19, you need to perform the following manual steps:
   a. SSH to the Knox gateway host on your cluster.
   b. Run the hostname command to get your domain name.
   c. Run the following commands (just replace the domain name test-master.dev.cldr.work with your correct, fully-qualified domain name):

   ```
   sudo sh -c '/opt/salt_2017.7.5/bin/salt --out=newline_values_only 'test-
   master.dev.cldr.work' pillar.get gateway:userfacingcert > /etc/certs-use
   r-facing/server.pem'
   sudo systemctl reload nginx.service
   ```

CB-23926: Autoscaling must be stopped before performing FreeIPA upgrade

Problem: During autoscaling, compute nodes are stopped and cannot receive the latest FreeIPA IP address update in a stopped state. This causes the Cloudera Data Hub clusters to become unhealthy after the FreeIPA upgrade when autoscaling is enabled.

Workaround:

1. Disable autoscaling before perfoming FreeIPA upgrade.
2. Start all compute nodes, which are in a STOPPED state.
3. Perform the FreeIPA upgrade.
4. Enable autoscaling after FreeIPA upgrade is successful.

Management Console

Cloudera Management Console Top Tasks

Date published: 2022-08-30
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Verify your credential prerequisites.....4
Cross-account access IAM role.....4
Prerequisites for the provisioning credential.....4
Service account for the provisioning credential.....4

Register your first environment.....4
Register an AWS environment from the Cloudera UI.....4
Register an AWS environment from the CDP CLI.....9
Register an Azure environment from the Cloudera UI.....9
Register an Azure environment from the CDP CLI.....14
Register a GCP environment from the Cloudera UI.....14
Register a GCP environment from the CDP CLI.....17

Verify your credential prerequisites

Cross-account access IAM role

To allow Cloudera to create resources in your AWS account, you create a cross-account access IAM role in your AWS account and grant Cloudera access to the role as a trusted principal by specifying a specific AWS account and an external ID.

The policy for the cross-account access IAM role must have the permissions enumerated in the documentation linked below. In addition, the IAM role must reference the specific AWS account ID and external ID provided in the Cloudera Management Console.

Prerequisites for the provisioning credential

To allow Cloudera to create resources on your Azure account, you must create the app-based credential. The credential allows Cloudera to access and provision a set of resources in your Azure account.

Cloudera uses an app-based credential to authenticate your Azure account and obtain authorization to create resources on your behalf. The app-based credential requires that you to manually configure the service principal created within your Azure Active Directory. The app-based method requires Owner role to be able to create a service principal, which must be given Contributor role or its equivalent.

To meet Azure prerequisites for Cloudera:

- Review the provided policies
- Perform the step 1 and step 2 described in the documentation for creating an app-based credential:

Service account for the provisioning credential

The provisioning credential for Google Cloud relies on a service account that can be assumed by Cloudera.

The following flow describes how the Google Cloud provisioning credential works:

1. Your GCP account administrator creates a service account and assigns the minimum permissions allowing Cloudera to create and manage resources in your Google Cloud account. Next, the administrator generates a service account access key pair for the service account.
2. The service account is registered as a credential in Cloudera and its access key is uploaded to Cloudera.
3. The credential is then used for registering your Google Cloud environment in Cloudera.
4. Once this is done, Cloudera uses the credential for provisioning environment-related resources, workload clusters, and resources for other Cloudera services that you run in Cloudera.

Review the following to learn about the permissions required for the credential and how to create the service account:

Register your first environment

Register an AWS environment from the Cloudera UI

Once you've met the AWS cloud provider requirements, register your AWS environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in AWS requirements.

Required role: EnvironmentCreator

Steps

1. Navigate to the Cloudera Management Console > Environments > Register environment:
2. On the Register Environment page, provide the following information:

Parameter	Description
General Information	
Environment Name (Required)	Enter a name for your environment. The name: • Must be between 5 and 28 characters long. • Can only include lowercase letters, numbers, and hyphens. • Must start with a lowercase letter.
Description	Enter a description for your environment.
Select Cloud Provider (Required)	Select Amazon.
Credential (Required)	
Select Credential	Select an existing credential or select Create new credential. For instructions on how to create a credential, refer to Creating a role-based credential. **Note:** Activate the Enable Permission Verification button if you want Cloudera to check permissions for your credential. Cloudera will verify that you have the required permissions for your environment.

3. Click Next.
4. On the Data Access and Data Lake Scaling page, provide the following information:

Parameter	Description
Data Lake Settings	
Data Lake Name (Required)	Enter a name for the Data Lake cluster that will be created for this environment. The name: • Must be between 5 and 100 characters long • Must contain lowercase letters • Cannot contain uppercase letters • Must start with a letter • Can only include the following accepted characters are: a-z, 0-9, -.
Data Lake Version (Required)	Select Cloudera Runtime version that should be deployed for your Data Lake. The latest stable version is used by default. All Cloudera Data Hub clusters provisioned within this Data Lake will be using the same Cloudera Runtime version.
Fine-grained access control on S3	
Enable Ranger authorization for AWS S3 Identity	Enable this if you would like to use Fine-grained access control. Next, from the Select AWS IAM role for Ranger authorizer dropdown, select the DATALAKE_ADMIN_ROLE IAM role created in Minimal setup for cloud storage.
Data Access and Audit	
Assumer Instance Profile (Required)	Select the IDBROKER_ROLE instance profile created in Minimal setup for cloud storage.
Storage Location Base (Required)	Provide the S3 location created for data storage in Minimal setup for cloud storage.
Data Access Role (Required)	Select the DATALAKE_ADMIN_ROLE IAM role created in Minimal setup for cloud storage.
Ranger Audit Role (Required)	Select the RANGER_AUDIT_ROLE IAM role created in Minimal setup for cloud storage.

Parameter	Description
IDBroker Mappings	We recommend that you leave this out and set it up after registering your environment as part of Onboarding Cloudera users and groups for cloud storage. **Note:** If you are using Fine-grained access control, this option is disabled, because you should onboard your users and groups via Ranger instead of using IDBroker mappings.
Scale (Required)	Select Data Lake scale. By default, “Light Duty” is used. For more information on Data Lake scale, refer to Data Lake scale.
Enable **Compute Cluster**	Enable Compute Clusters if you would like to deploy a containerized platform on Kubernetes for data services and shared services.

5. Click on Advanced Options to make additional configurations for your Data Lake. The following options are available:

Parameter	Description
Network and Availability	
Enable Multiple Availability Zones for Data Lake	Click the Enable Multiple Availability Zones for Data Lake toggle button to enable multi-AZ for the Data Lake. This option is disabled by default and is only available when a Medium Duty Data Lake is selected. Refer to Deploying Cloudera in multiple AWS availability zones.
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Amazon EC2 instance types.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on a specific Data Lake host group. For more information, see Recipes.

6. Click Next.
7. On the Region, Networking and Security page, provide the following information:

Parameter	Description
Region	
Select Region (Required)	Select the region that you would like to use for Cloudera. If you would like to use a specific existing virtual network, the virtual network must be located in the selected region.
Customer-managed Keys	
Enable Customer-Managed Keys	Enable this if you would like to provide a Customer-Managed Key (CMK) to encrypt environment's disks and databases. Next, under Select Encryption Key, select an existing CMK. For more information, refer to Customer managed encryption keys.
Select Encryption Key	Select an existing CMK.
Network	
Select Network (Required)	You have two options: • Select the existing virtual network where you would like to provision all Cloudera resources. Refer to VPC and subnet. • Select Create new network to have a new network with three subnets created. One subnet is created for each availability zone assuming three AZs per region; If a region has two AZs instead of three, then still three subnets are created, two in the same AZ.
Select Subnets (Required)	This option is only available if you choose to use an existing network. Multiple subnets must be selected and Cloudera distributes resources evenly within the subnets.
Network CIDR (Required)	This option is only available if you select to create a new network. If you selected to create a new network, provide Network CIDR that determines the range of private IPs that EC2 instances will use. This must be a valid private IP CIDR IP in IPv4 range. For example 10.10.0.0/16 are valid IPs. /16 is required to allow for enough IP addresses.

Parameter	Description
Create Private Subnets	This option is only available if you select to have a new network and subnets created. Is is turned on by default so that private subnets are created in addition to public subnets. If you disable it, only public subnets will be created. **Important:** For production deployments, Cloudera Data Platform recommends that you use private subnets. Work with your internal IT teams to ensure that users can access the browser interfaces for cluster services.
Create Private Endpoints	This option is only available if you select to have a new network and subnets created. It is disabled by default. Enable this option to use private endpoints instead of public endpoints for the following services: • Amazon EC2 • Amazon ECR - api and dkr • Amazon EFS • Amazon RDS for PostgreSQL • AWS Auto Scaling • AWS CloudFormation • AWS ELB • AWS S3 • AWS STS
Enable Public Endpoint Access Gateway	When Cluster Connectivity Manager is enabled, you can optionally enable Public Endpoint Access Gateway to provide secure connectivity to UIs and APIs in Data Lake and Cloudera Data Hub clusters deployed using private networking. If you are using your existing VPC, under Select Endpoint Access Gateway Subnets, select the public subnets for which you would like to use the gateway. The number of subnets must be the same as under Select Subnets and the availability zones must match. For more information, refer to Public Endpoint Access Gateway documentation.
Proxies	
Select Proxy Configuration	Select a proxy configuration if previously registered. For more information refer to Setting up a proxy server.
Security Access Settings	
Select Security Access Type (Required)	This determines inbound security group settings that allow connections to the Data Lake and Cloudera Data Hub clusters from your organization's computers. You have two options: • Create new security groups - Allows you to provide custom CIDR IP range for all new security groups that will be created for the Data Lake and Cloudera Data Hub clusters so that users from your organization can access cluster UIs and SSH to the nodes. This must be a valid CIDR IP in IPv4 range. For example: 192.168.27.0/24 allows access from 192.168.27.0 through 192.168.27.255. You can specify multiple CIDR IP ranges separated with a comma. For example: 192.168.27.0/24,192.168.28.0/24 If you use this setting, several security groups will get created: one for each Data Lake host group the Data Lake and one for each host group), one for each FreeIPA host group, and one for RDS; Furthermore, the security group settings specified will be automatically used for Cloudera Data Hub, Cloudera Data Warehouse, and Cloudera AI clusters created as part of the environment. • Provide existing security groups (Only available for an existing VPC) - Allows you to select two existing security groups, one for Knox-installed nodes and another for all other nodes. If you select this option, refer to Security groups to ensure that you open all ports required for your users to access environment resources.
Kubernetes	
Select **Private Kubernetes Cluster** or provide **Authorized IP Ranges**	If you have enabled Compute Clusters, you have the following options to configure the necessary networking information for the Kubernetes cluster: • Enable Private Kubernetes Cluster to create a private cluster that blocks all access to the API Server endpoint. • Provide the CIDRs to the Kubernetes API Server Authorized IP Ranges field to specify a set of IP ranges that will be allowed to access the Kubernetes API server. You need to provide the advanced configurations only once when creating your environment. The configurations will be applied to all compute clusters in the environment.

Parameter	Description
Worker Node Subnets	Uses the same set of subnets provided in **Network** section. You have the option to not use all of the previously provided subnets.
SSH Settings	
New or existing SSH public key (Required)	You have two options for providing a public SSH key: • Select a key that already exists on your AWS account within the specific region that you would like to use. • Upload a public key directly from your computer. **Note:** Cloudera does not use this SSH key. The matching private key can be used by your Cloudera administrator for root-level access to the instances provisioned for the Data Lake and Cloudera Data Hub.
Add tags	You can optionally add tags to be created for your resources on AWS. Refer to Defining custom tags.

8. Click on Advanced Options to make additional configurations for the FreeIPA cluster. The following options are available:

Parameter	Description
Network and Availability	
Enable Multiple Availability Zones for Data Lake	Click the Enable Multiple Availability Zones for Data Lake toggle button to enable multi-AZ for the FreeIPA cluster. Refer to Deploying Cloudera in multiple AWS availability zones.
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Amazon EC2 instance types.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on a specific FreeIPA host group. For more information, see Recipes.

9. Click Next.
10. On the Storage page, provide the following information:

Parameter	Description
Logs	
Logger Instance Profile (Required)	Select the LOG_ROLE instance profile created in Minimal setup for cloud storage.
Logs Location Base (Required)	Provide the S3 location created for log storage in Minimal setup for cloud storage.
Backup Location Base	Provide the S3 location created for FreeIPA and Data Lake backups in Minimal setup for cloud storage. If not provided, the default Backup Location Base uses the Logs Location Base.
Telemetry	
Enable Workload Analytics	Enables Cloudera Observability support for workload clusters created within this environment. When this setting is enabled, diagnostic information about job and query execution is sent to Cloudera Observability. For more information, refer to Enabling workload analytics and logs collection.
Enable Deployment Cluster Logs Collection	When this option is enabled. the logs generated during deployments will be automatically sent to Cloudera. For more information, refer to Enabling workload analytics and logs collection.

11. Click on Register Environment to trigger environment registration.
12. The environment creation takes about 60 minutes. The creation of the FreeIPA server and Data Lake cluster is triggered. You can monitor the progress from the web UI. Once the environment creation has been completed, its status will change to “Running”.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.

- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Register an AWS environment from the CDP CLI

Once you've met the AWS cloud provider requirements, register your AWS environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in AWS requirements.

Required role: EnvironmentCreator

Steps

Unlike in the Cloudera web interface, in CDP CLI environment creation is a three-step process with environment creation, setting IDBroker mappings and Data Lake creation being three separate steps. The easiest way to obtain the correct commands is to provide all parameters in Cloudera web interface and then generate the CDP CLI commands on the last page of the wizard. For detailed steps, refer to Obtain CLI commands for registering an environment.

To learn more about how to create Compute Cluster enabled environments with CLI, see Enabling default Compute Cluster for new environments.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Register an Azure environment from the Cloudera UI

Once you've met the Azure cloud provider requirements, register your Azure environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in Azure requirements.

Required role: EnvironmentCreator

Steps

1. Navigate to the Cloudera Management Console > Environments > Register environment:
2. On the Register Environment page, provide the following information:

Parameter	Description
General Information	
Environment Name (Required)	Enter a name for your environment. The name: • Must be between 5 and 28 characters long. • Can only include lowercase letters, numbers, and hyphens. • Must start with a lowercase letter.

Parameter	Description
Description	Enter a description for your environment.
Select Cloud Provider (Required)	Select Azure.
Microsoft Azure Credential (Required)	
Select Credential	Select an existing credential or select Create new credential. For instructions on how to create a credential, refer to Create an app-based credential.

3. Click Next.
4. On the Data Access and Data Lake Scaling page, provide the following information:

Parameter	Description
Data Lake Settings	
Data Lake Name (Required)	Enter a name for the Data Lake cluster that will be created for this environment. The name: • Must be between 5 and 100 characters long • Must contain lowercase letters • Cannot contain uppercase letters • Must start with a letter • Can only include the following accepted characters are: a-z, 0-9, -.
Data Lake Version (Required)	Select Cloudera Runtime version that should be deployed for your Data Lake. The latest stable version is used by default. All Cloudera Data Hub clusters provisioned within this Data Lake will be using the same Cloudera Runtime version.
Fine-grained access control on ADLS Gen2	
Enable Ranger authorization for ADLS Gen2 Identity	If you would like to use Fine-grained access control, enable this option and then select the Ranger RAZ managed identity created in the Minimal setup for cloud storage.
Data Access and Audit	
Assumer Identity (Required)	Select the Assumer managed identity created in Minimal setup for cloud storage.
Storage Location Base (Required)	Provide the ADLS Gen2 location created for data storage in Minimal setup for cloud storage.
Data Access Identity (Required)	Select the Data Lake Admin managed identity created in Minimal setup for cloud storage.
Ranger Audit Identity (Required)	Select the Ranger Audit managed identity created in Minimal setup for cloud storage.
IDBroker Mappings	We recommend that you leave this out and set it up after registering your environment as part of Onboarding Cloudera users and groups for cloud storage. **Note:** If you are using Fine-grained access control, this option is disabled, because you should onboard your users and groups via Ranger instead of using IDBroker mappings.
Scale (Required)	Select Data Lake scale. By default, “Light Duty” is used. For more information on data lake scale, refer to Data Lake scale.
Enable **Compute Cluster**	Enable Compute Clusters if you would like to deploy a containerized platform on Kubernetes for data services and shared services.

5. Click on Advanced Options to make additional configurations for your Data Lake. The following options are available:

Parameter	Description
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Sizes for virtual machines in Azure.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on a specific Data Lake host group. For more information, see Recipes.

6. Click Next.
7. On the Region, Networking and Security page, provide the following information:

Parameter	Description
Region	
Select Region (Required)	Select the region that you would like to use for accessing and provisioning resources from Cloudera. If you would like to use a specific existing virtual network, the virtual network must be located in the selected region.
Resource Group	
Select Resource Group (Required)	You have two options: • Select one existing resource group. If you select this, all Cloudera resources will be provisioned into that resource group. • Select Create new resource groups to have Cloudera create multiple resource groups.
Customer Managed Encryption Keys	
Enable Customer-Managed Keys	Enable this if you would like to provide a Customer-Managed Key (CMK) to encrypt environment's disks and databases. For more information, refer to Customer managed encryption keys.
Select Encryption Key Resource Group	Select the resource group where the CMK is located.
Encryption key URL	Provide the URL of the key value where the CMK resides. This is the same as the key identifier that you can copy directly from Azure Portal.
Managed identity for encryption	If using Azure Database for PostgreSQL Flexible Server, you can optionally select a managed identity created for encrypting it. For more information, refer to Managed identity for encrypting Azure Database for PostgreSQL Flexible Server.
Network	
Select Network (Required)	You have two options: • Select the existing virtual network where you would like to provision all Cloudera resources. Refer to VNet and subnets. • Select Create new network to have a new network with three subnets created.
Select Subnets (Required)	This option is only available if you choose to use an existing network. Multiple subnets must be selected and Cloudera distributes resources evenly within the subnets.
Network CIDR (Required)	This option is only available if you select to create a new network. If you selected to create a new network, provide Network CIDR that determines the range of private IPs that VMs will use. This must be a valid private IP CIDR IP in IPv4 range. For example 10.10.0.0/16 are valid IPs. /16 is required to allow for enough IP addresses.
Create Private Subnets	This option is only available if you select to have a new network and subnets created. Is is turned on by default so that private subnets are created in addition to public subnets. If you disable it, only public subnets will be created. **Important:** For production deployments, Cloudera recommends that you use private subnets. Work with your internal IT teams to ensure that users can access the browser interfaces for cluster services.

Parameter	Description
Enable Public Endpoint Access Gateway	When Cluster Connectivity Manager is enabled, you can optionally enable Public Endpoint Access Gateway to provide secure connectivity to UIs and APIs in Data Lake and Cloudera Data Hub clusters deployed using private networking. If you are using your existing VPC, under Select Endpoint Access Gateway Subnets, select the public subnets for which you would like to use the gateway. The number of subnets must be the same as under Select Subnets and the availability zones must match. For more information, refer to Public Endpoint Access Gateway documentation.
Create Private Endpoints	By default, the PostgreSQL Azure database provisioned for your Data Lake is reachable via a service endpoint (public IP address). To increase security, you can optionally select to have it reachable via a private endpoint instead of a service endpoint. **Note:** This option is only available if an existing resource group is selected. **Note:** Only the subnets that have Azure private endpoint network policies turned off are eligible for private endpoint creation. At least one such subnet is required. If you select to create a private endpoint and you are using your own VNet, you have two options: • Select “Create new private DNS zone” and Cloudera creates and manages a private DNS zone for you in the provided existing resource group. • Select your existing private DNS zone. If you select to create a private endpoint and you would like for Cloudera to create a new VNet, Cloudera creates a private DNS zone for you. For more information, refer to Private endpoint for Azure Postgres.
Create Public IPs	This option is disabled by default when Cluster Connectivity Manager is enabled and enabled by default when Cluster Connectivity Manager is disabled.
Flexible Server	During environment registration in Cloudera, the Flexible Server in public service mode is used by default, but you can specify to use the Flexible Server in private service mode (“Flexible Server with Private Link” or "Flexible Server with Delegated Subnet (deprecated)"). For more information, refer to Using Azure Database for PostgreSQL Flexible Server.
Proxies	
Select Proxy Configuration	Select a proxy configuration if previously registered. For more information refer to Setting up a proxy server.
Security Access Settings	
Select Security Access Type (Required)	This determines inbound security group settings that allow connections to the Data Lake and Cloudera Data Hub clusters from your organization’s computers. You have two options: • Create new security groups - Allows you to provide custom CIDR IP range for all new security groups that will be created for the Data Lake and Cloudera Data Hub clusters so that users from your organization can access cluster UIs and SSH to the nodes. This must be a valid CIDR IP in IPv4 range. For example: 192.168.27.0/24 allows access from 192.168.27.0 through 192.168.27.255. You can specify multiple CIDR IP ranges separated with a comma. For example: 192.168.27.0/24,192.168.28.0/24. If you use this setting, several security groups will get created: one for each Data Lake host group the Data Lake and one for each host group), one for each FreeIPA host group, and one for RDS; Furthermore, the security group settings specified will be automatically used for Cloudera Data Hub, Data Warehouse, and Cloudera AI clusters created as part of the environment. • Provide existing security groups (Only available for an existing VPC) - Allows you to select two existing security groups, one for Knox-installed nodes and another for all other nodes. If you select this option, refer to Security groups to ensure that you open all ports required for your users to access environment resources.
Kubernetes	

Parameter	Description
Enable **Private Kubernetes Cluster** or provide **Authorized IP Ranges**	If you have enabled Compute Clusters, you have the following options to configure the necessary networking information for the Kubernetes cluster: • Enable Private Kubernetes Cluster to create a private cluster that blocks all access to the API Server endpoint. • Provide the CIDRs to the Kubernetes API Server Authorized IP Ranges field to specify a set of IP ranges that will be allowed to access the Kubernetes API server. You need to provide the advanced configurations only once when creating your environment. The configurations will be applied to all compute clusters in the environment.
Enable **User Defined Routing**	Enable User Defined Routing (UDR) in case public IPs are blocked for egress.In case you enable UDR, you must select the specific worker node subnet where the UDR is configured.
AKS Private DNS Zone ID	When selecting **Private Kubernetes Cluster**, you also need to select an existing private DNS zone or select creating a new private DNS zone by Cloudera on your Azure account for the database.
Worker Node Subnets	Uses the same set of subnets provided in **Network** section. You have the option to not use all of the previously provided subnets.
SSH Settings	
New SSH public key (Required)	Upload a public key directly from your computer. **Note:** Cloudera does not use this SSH key. The matching private key can be used by your Cloudera administrator for root-level access to the instances provisioned for the Data Lake and Cloudera Data Hub.
Add tags	You can optionally add tags to be created for your resources on Azure. Refer to Defining custom tags.

8. Click on Advanced Options to make additional configurations for FreeIPA. The following options are available:

Parameter	Description
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Sizes for virtual machines in Azure.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on FreeIPA nodes. For more information, see Recipes.

9. Click Next.
10. On the Storage page, provide the following information:

Parameter	Description
Logs	
Logger Identity (Required)	Select the Logger managed identity created in Minimal setup for cloud storage.
Logs Location Base (Required)	Provide the ADLS Gen2 location created for log storage in Minimal setup for cloud storage.
Backup Location Base	Provide the ADLS Gen2 location created for FreeIPA and Data Lake backups in Minimal setup for cloud storage. If not provided, the default Backup Location Base uses the Logs Location Base.
Telemetry	
Enable Workload Analytics	Enables Cloudera Observability support for workload clusters created within this environment. When this setting is enabled, diagnostic information about job and query execution is sent to Cloudera Observability. For more information, refer to Enabling workload analytics and logs collection.
Enable Deployment Cluster Logs Collection	When this option is enabled. the logs generated during deployments will be automatically sent to Cloudera. For more information, refer to Enabling workload analytics and logs collection.

11. Click on Register Environment to trigger environment registration.

12. The environment creation takes about 60 minutes. The creation of the FreeIPA server and Data Lake cluster is triggered. You can monitor the progress from the web UI. Once the environment creation has been completed, its status will change to “Running”.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Register an Azure environment from the CDP CLI

Once you’ve met the Azure cloud provider requirements, register your Azure environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in Azure requirements.

Required role: EnvironmentCreator

Steps

Unlike in the Cloudera web interface, in CDP CLI environment creation is a three-step process with environment creation, setting IDBroker mappings and Data Lake creation being three separate steps. The easiest way to obtain the correct commands is to provide all parameters in the Cloudera web interface and then generate the CDP CLI commands on the last page of the wizard. For detailed steps, refer to Obtain CLI commands for registering an environment.

To learn more about how to create Compute Cluster enabled environments with CLI, see Enabling default Compute Cluster for new environments.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Register a GCP environment from the Cloudera UI

Once you’ve met the Google Cloud cloud provider requirements, register your GCP environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in GCP requirements.

Required role: EnvironmentCreator

Steps

1. Navigate to the Cloudera Management Console > Environments > Register environment.
2. On the Register Environment page, provide the following information:

Parameter	Description
General Information	
Environment Name (Required)	Enter a name for your environment. The name: • Must be between 5 and 28 characters long. • Can only include lowercase letters, numbers, and hyphens. • Must start with a lowercase letter.
Description	Enter a description for your environment.
Select Cloud Provider (Required)	Select Google Cloud.
Google Cloud Platform Credential (Required)	
Select Credential	Select an existing credential or select Create new credential. For instructions on how to create a credential for Google Cloud, refer to Create a provisioning credential for GCP.

3. Click Next.
4. On the Data Access and Data Lake Scaling page, provide the following information:

Parameter	Description
Data Lake Settings	
Data Lake Name (Required)	Enter a name for the Data Lake cluster that will be created for this environment. The name: • Must be between 5 and 100 characters long • Must contain lowercase letters • Cannot contain uppercase letters • Must start with a letter • Can only include the following accepted characters are: a-z, 0-9, -.
Data Lake Version (Required)	Select Cloudera Runtime version that should be deployed for your Data Lake. The latest stable version is used by default. All Data Hub clusters provisioned within this Data Lake will be using the same Cloudera Runtime version. Note: Google Cloud environments can only be provisioned in Cloudera with Cloudera Runtime version 7.2.8 or newer.
Data Access and Audit	
Assumer Service Account (Required)	Select the IDBroker service account created in Minimum setup for cloud storage.
Storage Location Base (Required)	Select the Google Storage location created for data in Minimum setup for cloud storage.
Data Access Service Account (Required)	Select the Data Lake Admin service account created in Minimum setup for cloud storage.
Ranger Audit Service Account (Required)	Select the Ranger Audit service account created in Minimum setup for cloud storage.
IDBroker Mappings	We recommend that you leave this out and set it up after registering your environment as part of Onboarding Cloudera users and groups for cloud storage.
Scale (Required)	Select Data Lake scale. By default, “Light Duty” is used. For more information on Data Lake scale, refer to Data Lake scale.

5. Click on Advanced Options to make additional configurations for your Data Lake. The following options are available:

Parameter	Description
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Machine type families.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on a specific Data Lake host group. For more information, see Recipes.

6. Click Next.
7. On the Region, Networking and Security page, provide the following information:

Parameter	Description
Region	
Select Region (Required)	Select the region where your VPC network is located.
Select Zone (Required)	Select a zone within the selected region.
Network	
Use shared VPC	This option is disabled by default. Enable this if you would like to use your existing shared VPC. Next enter: • Host project ID • Network name • Subnet name(s). If providing multiple, provide a comma separated list.
Select Network (Required)	Select the existing VPC network that you created as a prerequisite in the VPC network and subnets step. All Cloudera resources will be provisioned into this network.
Select Subnets (Required)	Select at least one existing subnet.
Create Public IPs	This option is disabled by default when Cluster Connectivity Manager is enabled and enabled by default when Cluster Connectivity Manager is disabled.
Proxies	Select a proxy configuration if previously registered. For more information refer to Setting up a proxy server.
Security Access Settings	
Select Security Access Type (Required)	You have two options: • Do not create firewall rule: If you are using a shared VPC you can set the firewall rules directly on the VPC. If you did so, you can select this option. • Provide existing firewall rules: If not all of your firewall rules are set directly on the VPC, provide the previously created firewall rules for SSH an UI access. You should select two existing firewall rules, one for Knox gateway-installed nodes and another for all other nodes. You may select the same firewall rule in both places if needed. For information on required ports, see Firewall rules.
SSH Settings	
New SSH public key (Required)	Upload a public key directly from your computer. Note: Cloudera does not use this SSH key. The matching private key can be used by your CDP administrator for root-level access to the instances provisioned for the Data Lake and Cloudera Data Hub.
Add tags	You can optionally add tags to be created for your resources on GCP. Refer to Defining custom tags.

8. Click on Advanced Options to make additional configurations for FreeIPA. The following options are available:

Parameter	Description
Hardware and Storage	For each host group you can specify an instance type. For more information on instance types, see Machine type families.
Cluster Extensions	
Recipes	You can optionally select and attach previously registered recipes to run on FreeIPA nodes. For more information, see Recipes.

9. Click Next.
10. On the Storage page, provide the following information:

Parameter	Description
Logs	
Logger Service Account (Required)	Select the Logger service account created in Minimum setup for cloud storage.
Logs Location Base (Required)	Select the Google Storage location created for logs in Minimum setup for cloud storage.
Backup Location Base	Select the Google Storage location created for FreeIPA backups in Minimum setup for cloud storage. If not provided, the default Backup Location Base uses the Logs Location Base.
Telemetry	
Enable Workload Analytics	Enables Cloudera Observability support for workload clusters created within this environment. When this setting is enabled, diagnostic information about job and query execution is sent to the Cloudera Observability.
Enable Deployment Cluster Logs Collection	When this option is enabled. the logs generated during deployments will be automatically sent to Cloudera.

11. Click Register Environment to trigger environment registration.
12. The environment creation takes about 60 minutes. The creation of the FreeIPA server and Data Lake cluster is triggered. You can monitor the progress from the web UI. Once the environment creation has been completed, its status will change to “Running”.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Register a GCP environment from the CDP CLI

Once you’ve met the Google Cloud cloud provider requirements, register your GCP environment.

Before you begin

This assumes that you have already fulfilled the environment prerequisites described in GCP requirements.

Required role: EnvironmentCreator

Steps

Unlike in the Cloudera web interface, in the CDP CLI environment creation is a two-step process with environment creation and data lake creation being two separate steps. The following commands can be used to create an environment in Cloudera.

1. Once you've met the prerequisites, register your GCP environment in Cloudera using the cdp environments create-gcp-environment command and providing the CLI input parameters. For example:

```
cdp environments create-gcp-environment --cli-input-json '{
    "environmentName": "test-env",
    "description": "Test GCP environment",
    "credentialName": "test-gcp-crd",
    "region": "us-west2",
    "publicKey": "ssh-rsa AAAAB3NzaZ1yc2EAAAADAQABAAABAQDwCI/wmQzbNn9YcA8v
dU+Ot41IIUWJfOfiDrUuNcULOQL6ke5qcEKuboXzbLxV0YmQcPFvswbM5S4FlHjy2VrJ5spy
GhQajFEm9+PgrsybgzHkkssziX0zRq7U4BVD68kSn6CuAHj9L4wx8WBwefMzkw7uO1CkfifI
p8UE6ZcKKKwe2fLR6ErDaN9jQxIWhTPEiFjIhItPHrnOcfGKY/p6OlpDDUOuMRiFZh7qMzfg
vWI+UdN/qjnTlc/M53JftK6GJqK6osN+j7fCwKEnPwWC/gmy8El7ZMHlIENxDut6X0qj9Okc/
JMmG0ebkSZAEbhgNOBNLZYdP0oeQGCXjqdv",
    "enableTunnel": true,
    "usePublicIp": true,
    "existingNetworkParams": {
        "networkName": "eng-private",
        "subnetNames": [
            "private-us-west2"
        ],
        "sharedProjectId": "dev-project"
    },
    "logStorage": {
        "storageLocationBase": "gs://logs",
        "serviceAccountEmail": "logger@dev-project.iam.gserviceaccount.com
"
    }
}'
```

Parameter	Description
environmentName	Provide a name for your environment.
credentialName	Provide the name of the credential created earlier.
region	Specify the region where your existing VPC network is located. For example "us-west2" is a valid region.
publicKey	Paste your SSH public key.
existingNetworkParams	Provide a JSON specifying the following: `{ "networkName": "string", "subnetNames": ["string", ...], "sharedProjectId": "string" }` Replace the values with the actual VPC network name, one or more subnet names and shared project ID. The sharedProjectId value needs to be set in the following way: • For a shared VPC, set it to the GCP host project ID • For a non-shared VPC, set it to the GCP project ID of the project where Cloudera is being deployed.

Parameter	Description
enableTunnel	By default Cluster Connectivity Manager is enabled (set to “true”). If you would like to disable it, set it to “false”. If you disable it, then you must also add the following to your JSON definition to specify two security groups as follows: `"securityAccess": { "securityGroupIdForKnox": "string", "defaultSecurityGroupId": "string" }`
usePublicIp	Set this to “true” or “false”, depending on whether or not you want to create public IPs.
logStorage	Provide a JSON specifying your configuration for cluster and audit logs: `{ "storageLocationBase": "string", "serviceAccountEmail": "string" }` The storageLocationBase should be in the following format: gs://my-bucket-name.

Note: CDP CLI includes the cdp environments create-gcp-environment --generate-cli-skeleton command option, which allows you to generate a CLI JSON template. You can also use CLI help to get some information about specific CLI command options.

2. To verify that your environment is running, use:

```
cdp environments list-environments
```

You can also log in to the Cloudera web interface to check the deployment status.

3. Once your environment and Data Lake are running, you should set IDBroker Mappings. To create the mappings, run the cdp environments set-id-broker-mappings command. For example:

```
cdp environments set-id-broker-mappings \
 --environment-name test-env \
 --data-access-role dl-admin@dev-project.iam.gserviceaccount.com \
 --ranger-audit-role ranger-audit@dev-project.iam.gserviceaccount.com \
 --mappings '[{"accessorCrn": "crn:altus:iam:us-west-1:45ca3068-42a6-422
7-8394-13a4493e2ac0:user:430c534d-8a19-4d9e-963d-8af377d16963", "role":
"data-science@dev-project.iam.gserviceaccount.com"},{"accessorCrn":"crn:
altus:iam:us-west-1:45ca3068-42a6-4227-8394-13a4493e2ac0:machineUser:mfox-
gcp-idbmms-test-mu/2cbca867-647b-44b9-8e41-47a01dea6c19","role":"data-eng@
dev-project.iam.gserviceaccount.com"}]'
```

Parameter	Description
environment-name	Specify a name of the environment created earlier.
data-access-role	Specify an email address of the Data Lake admin service account created earlier.
ranger-audit-role	Specify an email address of the Ranger audit service account created earlier.

Parameter	Description

mappings	Map Cloudera users or groups to GCP service accounts created earlier. Use the following syntax:
	`[`
	`{`
	`"accessorCrn": "string",`
	`"role": "string"`
	`}`
	`...`
	`]`
	You can obtain user or group CRN from the Management Console > User Management by navigating to details of a specific user or group.
	The role should be specified as service account email.

4. Next, sync IDBroker mappings:

```
cdp environments sync-id-broker-mappings --environment-name demo3
```

5. Finally, check the sync status:

```
cdp environments get-id-broker-mappings-sync-status --environment-name d
emo3
```

6. One your environment is running, you can create a Data Lake using the cdp datalake create-gcp-datalake command and providing the CLI input parameters:

```
cdp datalake create-gcp-datalake --cli-input-json '{
    "datalakeName": "my-dl",
    "environmentName": "test-env",
    "scale": "LIGHT_DUTY",
    "cloudProviderConfiguration": {
        "serviceAccountEmail": "idbroker@dev-project.iam.gserviceaccount
.com",
        "storageLocation": "gs://data-storage"
    }
}'
```

Parameter	Description
datalakeName	Provide a name for your Data Lake.
environmentName	Provide a name of the environment created earlier.
scale	Provide Data Lake scale. It must be one of: • LIGHT_DUTY or • MEDIUM_DUTY_HA.
cloudProviderConfiguration	Provide the name of the data storage bucket and the email of the IDBroker service account.

Note: CDP CLI includes the cdp datalake create-gcp-datalake --generate-cli-skeleton command option, which allows you to generate a CLI JSON template. You can also use CLI help to get some information about specific CLI command options.

7. To verify that your Data Lake is running, use:

```
cdp datalake list-datalakes
```

You can also log in to the Cloudera web interface to check the deployment status.

After you finish

After your environment is running, perform the following steps:

- You must assign roles to specific users and groups for the environment so that selected users or user groups can access the environment. Next, you need to perform user sync. For steps, refer to Enabling admin and user access to environments.
- You must onboard your users and/or groups for cloud storage. For steps, refer to Onboarding Cloudera users and groups for cloud storage.
- You must create Ranger policies for your users. For instructions on how to access your Data Lake, refer to Accessing Data Lake services. Once you've accessed Ranger, create Ranger policies to determine which users have access to which databases and tables.

Troubleshooting the Cloudera Management Console

Date published: 2019-08-22
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Checking Cloudera logs **4**
Collecting Data Lake diagnostics via Cloudera Manager 4
Accessing Cloudera Manager logs 5

Obtain Cloudera tenant ID **5**

Credential creation errors on AWS **5**

Credential creation errors on Azure **6**

Specifying a group when user belongs to multiple groups **8**

General networking and authentication errors **10**

Problem deleting resources when using AWS Firewall Manager **10**

Checking Cloudera logs

When troubleshooting Environment issues and Data Lake clusters, you can access the following logs:

Collecting Data Lake diagnostics via Cloudera Manager

Cloudera Manager can collect system status, configuration details, logs, and other information from a Data Lake cluster into a zip file to send to Cloudera Support.

To help with solving problems when using Cloudera Manager on your data lake cluster, Cloudera Manager can collect diagnostic data. You can choose to trigger this collection to aid with resolving a problem, or you can configure Cloudera Manager to send a diagnostic bundle to Cloudera on a regular schedule. Cloudera Support analyses the diagnostic information to proactively identify problems.

To trigger creation of a diagnostic bundle for a Data Lake, navigate to the Cloudera Manager UI that manages the Data Lake cluster. In the lower section of the left navigation pane, choose Support Send Diagnostic Data and set the options in the dialog box that opens.

Specifically:

1. Log in to Cloudera web interface.
2. In the left navigation panel, click Environments, and then search for the appropriate environment and click the environment name.
3. In the environment detail page, open the Data Lake tab.
4. Open Cloudera Manager by clicking its link in the Services section.
5. In the bottom of the left navigation pane, click Support Send Diagnostic Data .
6. Choose whether to send the diagnostic bundle to Cloudera automatically or only to collect the bundle.

   If the Cloudera Manager host does not have an internet connection, you may want to collect the bundle then move it to a host with access to the internet.
7. If appropriate, enter a Cloudera Support case number.
8. Select the Data Lake cluster if this instance of Cloudera Manager is managing more than one cluster.
9. If appropriate, limit the diagnostic collection to a single host, service, or role.

   Open the Restrict log and metrics collection and choose the Data Lake host, service, or role for which you want to collect diagnostics.
10. Set the logic for what data to collect, by target size or by date range.

    If you choose a target size, set the end time to a time to a few minutes after the event that you are trying to capture diagnostics for. The time range is based on the timezone of the host where Cloudera Manager Server is running.
11. Add a comment to describe the reason for collecting the diagnostic data.
12. Start the collection by clicking Collect Diagnostic Data or Collect and Upload Diagnostic Data.

    The Cloudera Manager task dialog appears to track the jobs involved in collecting diagnostics.
13. When the collection tasks are complete, click Download to save the diagnostic bundle locally.

Related Information

Diagnostic Data Collection in Cloudera Manager

Collecting Usage and Diagnostic Data

Accessing Cloudera Manager logs

Cloudera uses Cloudera Manager to orchestrate the installation of Data Lake components. Each instance in the cluster runs an Cloudera Manager agent which connects to the Cloudera Manager server. Cloudera Manager server is declared by the user during the cluster installation wizard.

Refer to the following documentation for information on how to access Cloudera Manager logs:

Related Information

Cloudera Manager logs

Obtain Cloudera tenant ID

Your Cloudera tenant ID can be found in the user and machine user profile pages, just below the resource CRN.

You may need to obtain your Cloudera tenant ID, for example when working with Cloudera support. Use the following steps to find your Cloudera tenant ID.

Steps

1. In Cloudera web interface, navigate to Cloudera Management Console > User Management.
2. Find your user and click on it to navigate to details.
3. Your Tenant ID is listed among other user details.

Credential creation errors on AWS

The following section lists common issues related to creating a credential on AWS and steps to resolve them.

User: arn:aws:iam:::user/assume-only-user is not authorized to perform: sts:AssumeRole

Error: The following error occurs when creating a role-based AWS credential:

```
User: arn:aws:iam:::user/assume-only-user is not authorized to perform: sts:
AssumeRole
```

Solution: The error occurs when Cloudera is not authorized to use the role that you are trying to register as part of cloud credential creation. The most common reason for this error is that when creating the cross-account IAM role you did not provide the AWS account ID. Refer to the documentation for creating a role-based credential on AWS for correct steps to create a cross-account IAM role.

Internal error when creating an AWS credential from CDP CLI

Error: The following error occurs when creating a role-based AWS credential via CDP CLI:

```
An error occurred: An internal error has occurred. Retry your request, but i
f the problem persists, contact us with details by posting a message on the
Cloudera Community forums. (Status Code: 500; Error Code: UNKNOWN; Service:
environments; Operation: createAWSCredential; Request ID: dbf1fb6f-3161-46e1
-80e9-a894461ceec3;)
```

Solution: A common reason for this error is that the external ID that you used for your cross-account IAM role is associated with another user. The external ID is tied to the Cloudera user who obtained it; Therefore, only the user who obtained the external ID is able to complete the credential creation flow in Cloudera with a given external ID. If a Cloudera user tries using an IAM role with an external ID obtained by another Cloudera user, Cloudera will return

this error message. To resolve the issue, log in to CDP CLI, obtain a new external ID, create a new cross-account IAM role, and then try creating the credential again.

Related Information

Create a cross-account IAM role

Credential creation errors on Azure

The following section lists common issues related to creating a credential on Azure and steps to resolve them.

You don't have enough permissions to assign roles

Error: You don't have enough permissions to assign roles, please contact with your administrator

Solution: This error occurs in the Azure console because you are not authorized to perform role assignment. To solve the problem, ask your Azure administrator to perform the step of assigning the Contributor role to your application.

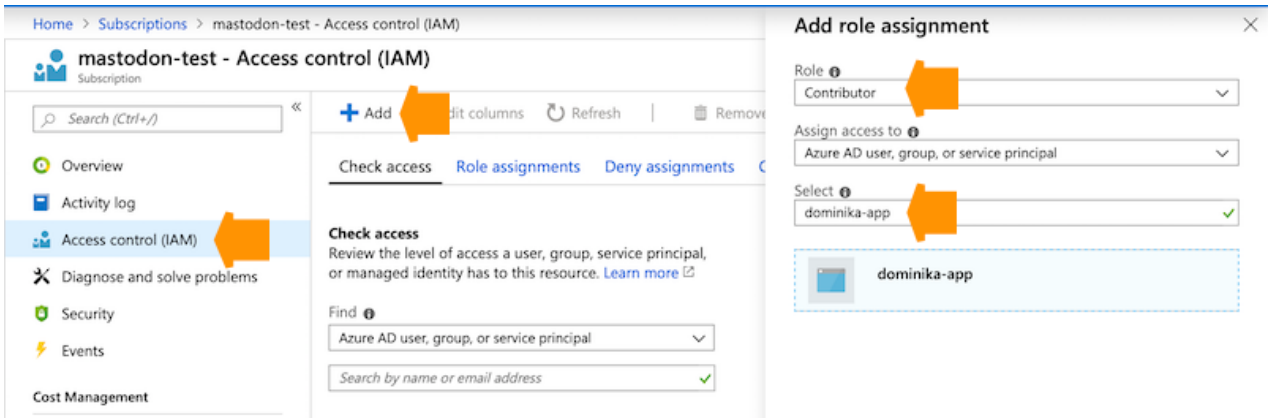

Client does not have authorization

Error: Failed to verify credential: Status code 403, {“error”:{“code”:”AuthorizationFailed”, “message”:”The client ‘X’ with object id ‘z’ does not have authorization to perform action ‘Microsoft.Storage/storageAccounts/read’ over scope ‘subscriptions/…’”}

Solution: If you get this error during app-based credential creation, then the reason is that you did not assign the Contributor role to your app. To solve the issue, either assign the Contributor role to the app or - if you don't have Owner role - ask your Azure administrator to perform this step.

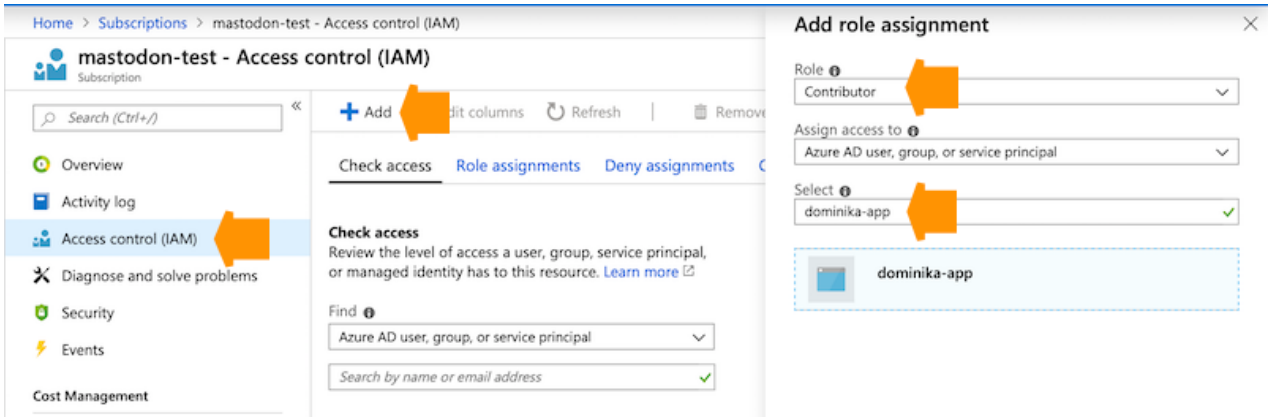

Application with identifier was not found in the directory

Error: The following error appears during the interactive credential creation flow:

```
Application with identifier was not found in the directory. This can happen
if the application has not been installed by the administrator of the tenant
 or consented to by any user in the tenant. You may have sent your authentic
ation request to the wrong tenant.
```

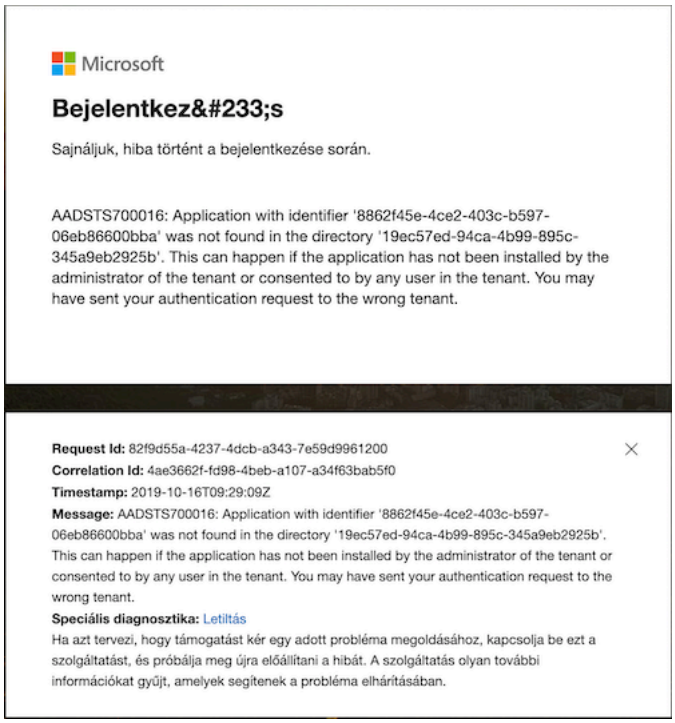

Solution: To fix the issue, wait for one minute and then refresh the page. Afterwards, you should see the application consent page:

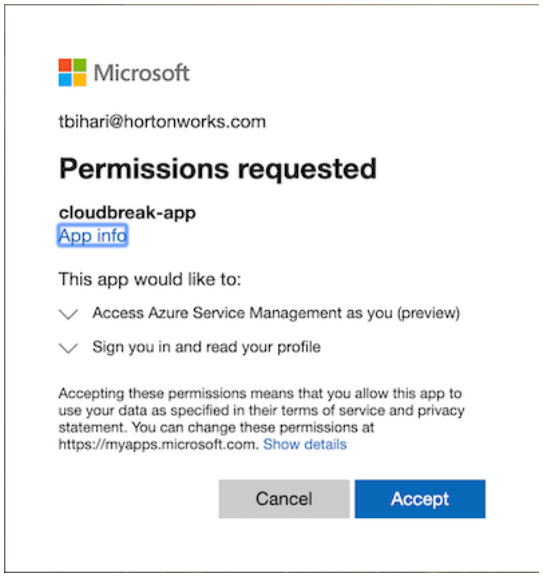

Specifying a group when user belongs to multiple groups

If a user is mapped to multiple roles via group membership, the specific role to be used needs to be provided at runtime.

Note:

Spark does not work with Hive Metastore if there are multiple IDBroker group mappings. If you would like to use multiple group mappings, you should use the Ranger Authorization Service (RAZ).

If you do not specify the group, you will get an error similar to the following:

```
Ambiguous group role mappings for the authenticated user.
```

To provide the role, use the following commands:

```
fs.<s3a | azure | gs>.ext.cab.required.group
```

This command specifies the name of a particular group from which the credentials should be resolved. This can be used, when multiple user groups resolve to cloud roles, to specify for which group credentials are needed.

```
fs.<s3a | azure | gs>.ext.cab.required.role
```

This command specifies a particular cloud IAM role for which the credentials are needed. This can be used when the required cloud IAM role is known, and the IDBroker is configured in any way to resolve the user to that role.

The following examples illustrate how to use these commands in different contexts:

HDFS CLI

The following command sets a Java system property (-D) called "fs.s3a.ext.cab.required.group" with the value of the group to use (GROUP).

```
hdfs dfs -Dfs.s3a.ext.cab.required.group=GROUP -ls s3a://BUCKET/PATH
```

The system property is used by the s3a binding to request that group specifically instead of falling back to looking up all the groups for the user.

Spark

Here is how to set the config for a Spark job:

```
spark-shell --conf spark.yarn.access.hadoopFileSystems=s3a://BUCKET/ --conf
spark.hadoop.fs.s3a.ext.cab.required.group=GROUP

val textFile = spark.read.text("s3a://BUCKET/PATH")
textFile.count()
textFile.first()
```

Livy

Here is how to set the config for a batch job:

```
# Ensure credentials
kinit

# Create test.py with following content
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder.appName("SimpleApp").getOrCreate()
textFile = spark.read.text("s3a://cldr-cdp-dl-1/user/systest/test.csv")
textFile.count()

# Upload file to cloud storage
hdfs dfs -copyFromLocal -f test.py s3a://cldr-cdp-dl-1/user/systest/test.py

# Submit Livy batch job
curl --negotiate -u: -i -XPOST --data '{"file": "s3a://cldr-cdp-dl-1/user/
systest/test.py", "conf":{"spark.yarn.access.hadoopFileSystems":"s3a://cldr
-cdp-dl-1/", "spark.hadoop.fs.s3a.ext.cab.required.group":"GROUP"}}' -H "Con
tent-Type: application/json" http://krisden-1.gce.cloudera.com:8998/batches
# Check Livy batch job
curl --negotiate -u: -i http://krisden-1.gce.cloudera.com:8998/batches/0
curl --negotiate -u: -i http://krisden-1.gce.cloudera.com:8998/batches/0/log
```

Here is how to create a new session in Livy using the right config:

```
# Ensure credentials
kinit

# Create Livy session
curl --negotiate -u: -i -XPOST --data '{"kind": "pyspark", "conf":{"spark.
yarn.access.hadoopFileSystems":"s3a://cldr-cdp-dl-1/", "spark.hadoop.fs.s3a.
ext.cab.required.group":"GROUP"}}' -H "Content-Type: application/json" http:
//krisden-1.gce.cloudera.com:8998/sessions

# Check Livy session is idle
curl --negotiate -u: -i http://krisden-1.gce.cloudera.com:8998/sessions
curl --negotiate -u: -i http://krisden-1.gce.cloudera.com:8998/sessions/0
curl --negotiate -u: -i http://krisden-1.gce.cloudera.com:8998/sessions/0/
state

# Use Livy session
curl --negotiate -u: -i -XPOST --data '{"kind": "pyspark", "code": "spark
.read.text(\"s3a://cldr-cdp-dl-1/user/systest/test.csv\").count()"}' -H "Con
tent-Type: application/json" http://krisden-1.gce.cloudera.com:8998/sessions
/0/statements

# Check the session statement output
curl --negotiate -u: -i http://krisden-1.gce.cloudera.com:8998/sessions/0/st
atements/0

# Delete the session
curl --negotiate -u: -i -XDELETE http://krisden-1.gce.cloudera.com:8998/ses
sions/0
curl --negotiate -u: -i http://krisden-1.gce.cloudera.com:8998/sessions
```

Zeppelin

Make sure to set the following configs in the Livy interpreter config before running the notebook:

- Set livy.spark.yarn.access.hadoopFileSystems to s3a://cldr-cdp-dl-1/
- Set livy.spark.hadoop.fs.s3a.ext.cab.required.group to GROUP

Next, create a Zeppelin notebook with the following content:

```
%livy.pyspark
textFile = spark.read.text("s3a://cldr-cdp-dl-1/user/systest/test.csv")
textFile.count()
```

Note:

Livy interpreter uses Livy sessions and not batches.

General networking and authentication errors

Are you finding that your Cloudera interactions are hitting errors when they shouldn't? Check to see if your identity management system is working. FreeIPA errors can be hidden as failures from Knox, Kerberos, or other systems that act as clients to FreeIPA.

You can eliminate FreeIPA issues by checking the following status:

- Cloudera Management Console: Select the Environment Summary FreeIPA status .

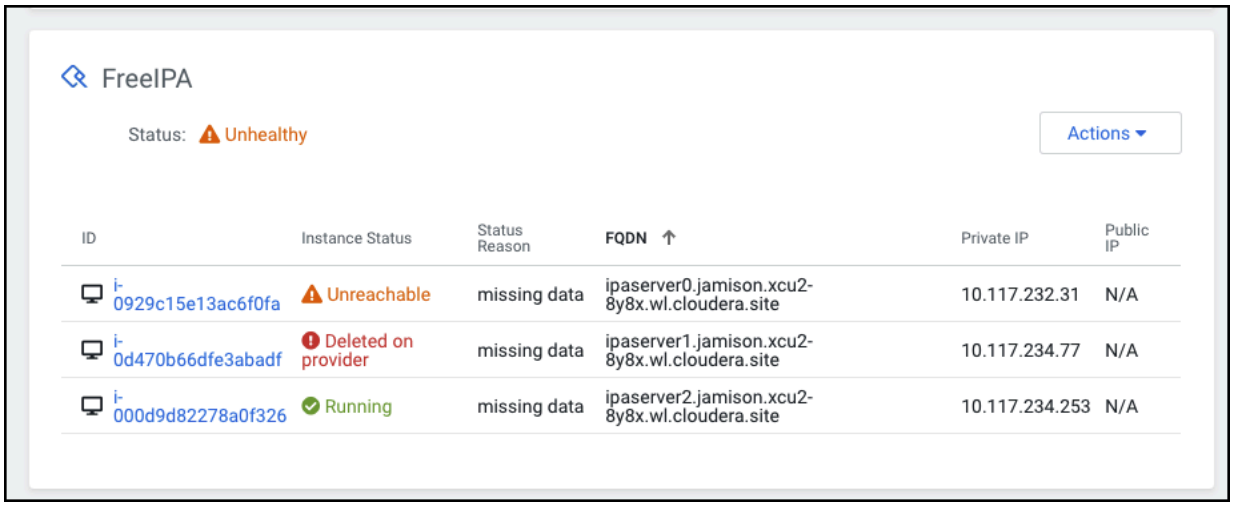

- Cloud provider instance status: Follow the link for one of the FreeIPA hosts in the status to open the cloud provider's status page for that host. If one of the FreeIPA hosts is stopped, restart it. It is possible to still see a "Running" status for FreeIPA even if one of the hosts is stopped.

Related Information

Managing FreeIPA

Problem deleting resources when using AWS Firewall Manager

If you have AWS Firewall Manager enabled, ensure that your AWS Firewall Manager policies or rules don't affect any AWS resources created by Cloudera. If they do, you may experience problems deleting these resources due to dependency violation.

For more information, refer to Security group policies > Avoid conflicts if you also use outside sources to manage security groups in AWS documentation.

Cloudera Management Console

User Management

Date published: 2019-08-22
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing user access 5

Onboarding users 6
- Configuring identity providers in Cloudera 6
  - Generate IdP metadata 7
  - Set up IdP in Cloudera 7
  - Set up IdP as service provider 9
  - Synchronizing group membership 10
  - Updating an identity provider 11
  - Disabling the Cloudera SSO login 12
- Configure AAD in Cloudera 13
- Configure SCIM with Azure AD 19
- Importing or uploading users 23
- Generating workload usernames based on email 23
- Known issues and troubleshooting related to IdP setup in Cloudera 24

Understanding Cloudera user accounts 25
- Cloudera account administrator 25
- Cloudera user 25
- Cloudera workload user 26
- Cloudera machine user 26

Understanding Cloudera roles 27
- Account roles 27
- Resource roles 29
- Group membership admin roles 34
- Example role assignment scenario 34

User and group limits 35

Managing users and machine users 36
- Creating a machine user in Cloudera 36
- Deleting users and machine users 36
- Assigning account roles to users 38
- Assigning resource roles to users 39
  - Assign environment role 39
  - Assign shared resource role 40
  - Assign Cloudera Data Hub role 41
  - Assign classic cluster role 42

Managing groups in Cloudera 44
- Reserved group names 44
- Understanding Cloudera groups 45

Synchronizing group membership....45
Creating a group....46
Adding or removing a user....47
Assigning account roles to groups....49
Assigning resource roles to groups....50
Assign environment role....50
Assign shared resource role....51
Assign Cloudera Data Hub role....52
Assign classic cluster role....53
Assigning a group admin....54
Updating a group....55
Removing account roles from a group....56
Deleting a group....56

Performing user sync....57

Access paths to Cloudera....59

Setting a default identity provider in Cloudera....60

Logging in as workload user....61

Setting the workload password....61

Managing SSH keys....64

Generating an API access key....65

Retrieve keytabs for workload users....66

Managing user access and authorization

To provide access to resources such as environments and clusters, you must add users and groups and assign roles and resources to them.

Using the Cloudera Management Console, you can perform the following tasks:

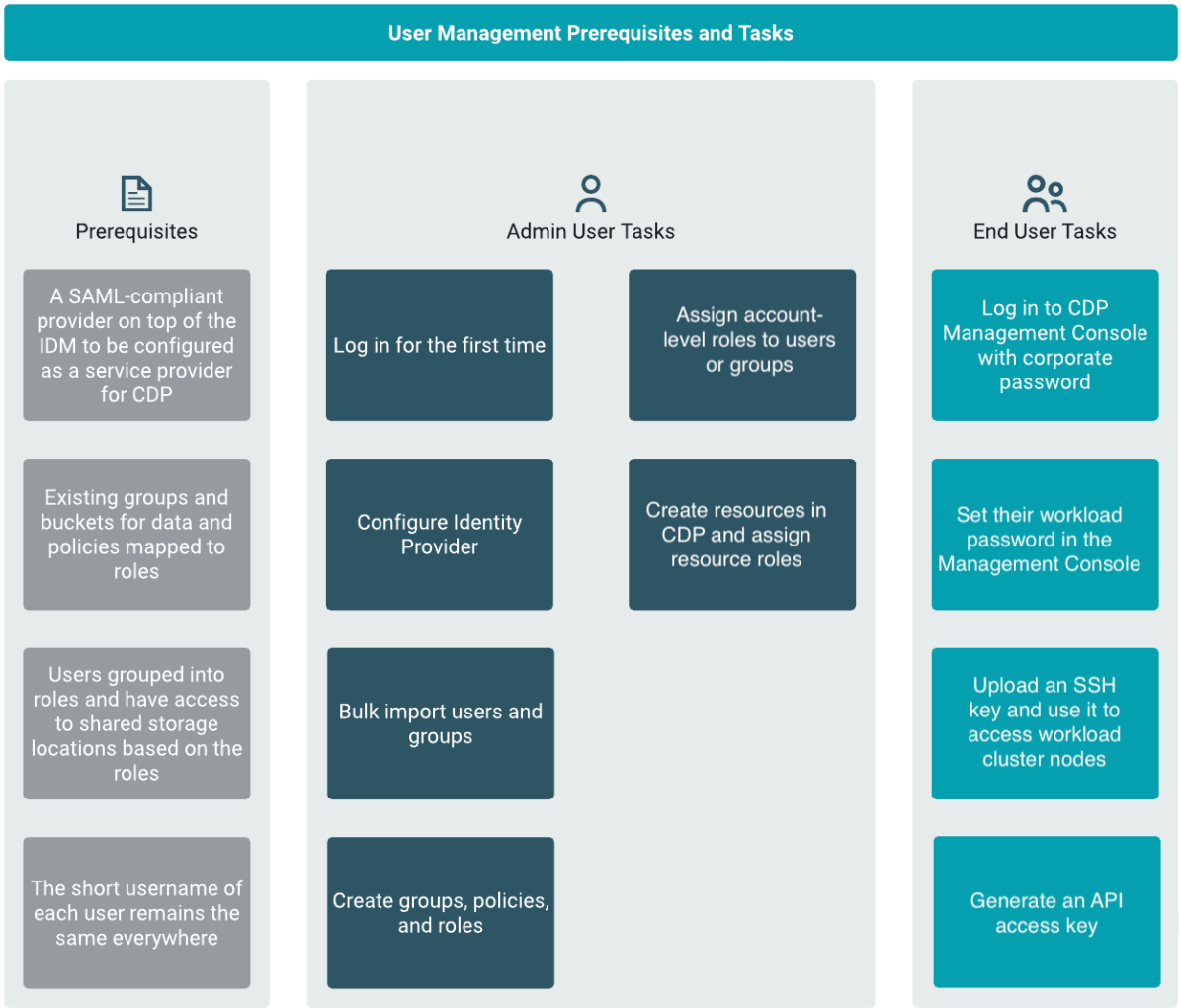

Related Information

Onboarding users
Understanding Cloudera user accounts
Understanding account roles and resource roles
Managing users and machine users in Cloudera
Managing groups in Cloudera
Performing user sync
Access paths to Cloudera and its components
Accessing non-SSO interfaces using workload user and password
Setting the workload password
Managing SSH keys

Generating an API access key
Retrieve keytabs for workload users

Onboarding users

To enable users to work on the various Cloudera components and services, you can onboard them by configuring identity providers or importing users in bulk.
https://www.youtube.com/embed/F-L1I7MLQlk

Configuring identity providers in Cloudera

An account administrator or PowerUser must onboard users by setting up identity federation with Cloudera.

If your organization uses an enterprise identity provider (IdP) that is compliant with Security Assertion Markup Language (SAML), you must set up identity federation with Cloudera. Identity federation allows users within your organization to log in to Cloudera through the authentication system in your organization without registering with Cloudera or creating a Cloudera account.

The following diagram illustrates how identity federation works with Cloudera:

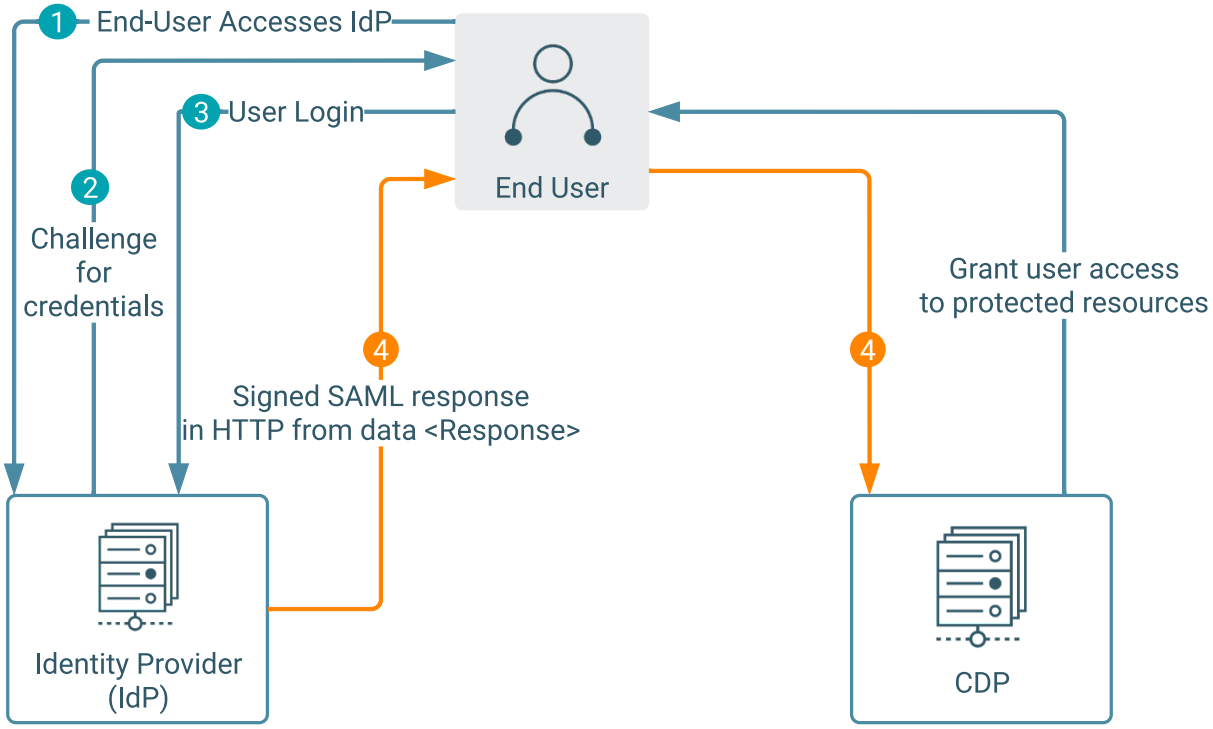

Note:

As shown in the diagram, there is no network communication required between Cloudera and customer IdP, so there is no need to create firewall rules.

Cloudera supports the following:

- Cloudera supports the SAML 2.0 standard. You can set up any identity provider for Cloudera that uses SAML 2.0.
- You can set up a maximum of 10 SAML 2.0-compliant identity providers in Cloudera.

Setting up an identity provider for Cloudera involves the following steps:

1. The IdP administrator in your organization generates the SAML metadata that describes your enterprise IdP.
2. The Cloudera administrator sets up the identity provider in Cloudera.
3. The IdP administrator configures the enterprise IdP in your organization to work with Cloudera as a service provider.

Generating the identity provider metadata

Use your enterprise IdP user interface to generate the identity provider SAML metadata file.

CDP has the following requirements for the identity provider SAML metadata file:

- The file must be a valid XML file.
- The metadata must include at least one IDPSSOdescriptor element.
- The metadata must contain information about at least one valid x.509 certificate that can be used to verify signed assertions.

The following XML file example shows the elements to include in the identity provider SAML metadata file:

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" enti
tyID="http://www.IdP.com/entity_ID">
   <md:IDPSSODescriptor WantAuthnRequestsSigned="false" protocolSupportEnum
eration="urn:oasis:names:tc:SAML:2.0:protocol">
      <md:KeyDescriptor use="signing">
         <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
              <ds:X509Data><ds:X509Certificate>full_x509-certificate_stri
ng</ds:X509Certificate></ds:X509Data>
         </ds:KeyInfo>
      </md:KeyDescriptor>
      <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAdd
ress</md:NameIDFormat>
      <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindin
gs:HTTP-POST"
           Location="https://application.IdP.com/app/.../sso/saml"/>
      <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:
HTTP-Redirect"
           Location="https://application.IdP.com/app/.../sso/saml"/>
   </md:IDPSSODescriptor>
</md:EntityDescriptor>
```

Note:

The WantAuthnRequestsSigned=true option is not supported.

Setting up the identity provider in Cloudera

In Cloudera, you must create an identity provider to capture the SAML metadata and connection information for your enterprise IdP. To create an identity provider in Cloudera, you must be a Cloudera account administrator or have the PowerUser role.

About this task

Note:

Creating and integrating with identity providers should be considered a privileged action and you must think about how it will affect group memberships within Cloudera. Each identity provider manages their own unique set of group names and memberships. However, different identity providers can define the same group names. When users from different identity providers federate to Cloudera (with group sync on) the identity providers may provide the same group names. Since groups in Cloudera are defined by their name - as provided by the identity provider - when this happens, users will be added to the same group in Cloudera, even though they federated from different identity providers. This may grant unexpected permissions. If your organization needs group names to be unique across all identity providers federating to Cloudera, our recommended approach is to create different Cloudera accounts for each identity provider, and only set up one identity provider to federate to a Cloudera account.

Note:

There are certain group names that are reserved and therefore cannot be synchronized to Cloudera. See Reserved group names.

Required role: Account administrator or PowerUser

Procedure

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. In the User Management section of the side navigation panel, click Identity Providers.
4. Click Create Identity Provider.
5. On the Create Identity Provider window, enter the name you want to use for the Cloudera identity provider.
6. Select whether to synchronize the user group membership in Cloudera with the user group membership in your enterprise IdP.
7. To synchronize the groups, select the Sync Groups on Login option.

   For more information about user group synchronization, see Synchronizing group membership.
8. In Provider Metadata, select File Upload to upload a file that contains the identity provider SAML metadata or select Direct Input to paste the identity provider SAML metadata directly.
9. Click Create.

Results

Cloudera adds the new identity provider to the list of Cloudera identity providers on the Identity Providers page.

After you create the identity provider in Cloudera, you can view its properties to get the information you need to configure your enterprise IdP to work with Cloudera.

On the Identity Providers page, click the name of the new Cloudera identity provider to see its properties:

Property	Description
Name	Name of the Cloudera identity provider.
ID	ID generated for the Cloudera identity provider.
Sync Groups on Login	Indicates whether Cloudera synchronizes a user's group membership in Cloudera with the user's group membership in your enterprise IdP when a user logs in. For more information about user group synchronization, see Group Membership Synchronization.
CRN	The Cloudera resource name assigned to the Cloudera identity provider.

Property	Description
SAML Identity Provider Metadata	The identity provider SAML metadata for your enterprise IdP that you provided when you created the Cloudera identity provider.
Generate workload username by email	You can optionally check this if you use an opaque ID for SAML NameID and SCIM userName so that the workload username is generated based on the email instead of the default. For more information, see Generating workload usernames based on email.
Enable SCIM	You can optionally check this to enable SCIM for Azure AD. For more information, see Configure SCIM with Azure AD.
Cloudera SAML Service Provider Metadata	The Cloudera SAML service provider metadata to configure your enterprise IdP.

Configuring your enterprise IdP to work with Cloudera as a service provider

Cloudera provides a service provider SAML metadata file that describes the information that Cloudera requires to enable users to log in to Cloudera through your enterprise IdP.

You can get the Cloudera SAML metadata XML from the Identity Providers page in Cloudera web interface by navigating to the details of your identity provider configuration.

The Cloudera SAML metadata file includes the following information:

Information	Attribute	Description
Name ID formats that Cloudera supports	NameIDFormat	The metadata includes multiple name ID formats. Use one of the formats in the list for the user ID. Cloudera supports any type of name ID format other than transient. Cloudera requires that you use name ID formats that are globally unique within your identity provider. The name ID format should also be stable over time. Cloudera does not recommend using email addresses because, although they can be unique, they are typically not stable over time. If your NameID is an opaque ID (such as a UUID), you can Generate workload usernames based on email. The value of NameID is case-sensitive.
Cloudera SSO URL	Location	It should be the same as the "Location" value of the "<AssertionConsumerService>" in the Cloudera SAML Service Provider metadata. It has a fixed format. For Cloudera Control Plane region us-west-1:https://consoleauth.altus.cloudera.com/saml?samlProviderId=CDP-assigned-ID For any other Cloudera Control Plane region: https://consoleauth.<CONTROL_PLANE_REGION>.cdp.cloudera.com/consoleauth/saml?samlProviderId=CDP-assigned-ID For more information about the ID that Cloudera generates and assigns to the Cloudera identity provider, see Setting Up the Identity Provider in CDP. This attribute is required.

Information	Attribute	Description
Endpoint for binding	Binding	Use the following URN as the endpoint that your enterprise IdP must bind to: urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST This attribute is required.
User email address	RequestedAttribute: mail	Set the email address attribute to the following URN: urn:oid:0.9.2342.19200300.100.1.3 Cloudera also accepts: mail This attribute is required. Although Cloudera requires the user email address, it is used for display purposes only.
(Optional) List of groups that the user is a member of	RequestedAttribute: groups	Set the group list attribute to the following URN: https://cdp.cloudera.com/SAML/Attributes/groups This attribute is optional. For more information about the group list and how Cloudera synchronizes group membership, see Synchronizing Group Membership.
(Optional) User first name	RequestedAttribute: firstName	Set the user first name attribute to the following URN: https://cdp.cloudera.com/SAML/Attributes/firstName This attribute is optional; used for display purposes only.
(Optional) User last name	RequestedAttribute: lastName	Set the user last name attribute to the following URN: https://cdp.cloudera.com/SAML/Attributes/lastName This attribute is optional; used for display purposes only.

If your enterprise IdP allows it, you can upload the Cloudera SAML metadata file to your enterprise IdP. Otherwise, use your enterprise IdP user interface to set up Cloudera as a service provider.

Synchronizing group membership

Cloudera can synchronize the user's group membership provided by your enterprise IdP with the user's group membership in Cloudera.

When a user initially logs in to Cloudera through the identity management system in your organization, Cloudera creates a Cloudera user account for the user. However, without being assigned Cloudera roles, the user cannot perform tasks in Cloudera. Cloudera recommends that you create Cloudera groups with assigned roles and add users to the groups so that the users can take on the roles assigned to the groups.

When you create an identity provider, you can select the Sync Groups on Login option to enable Cloudera to synchronize the user group membership. By default, the Sync Groups on Login option is disabled. Clear the option selection if you do not want Cloudera to synchronize the user group membership.

Group names must be alphanumeric, may include dots (.), hyphens (-), and underscores (_), and must be fewer than 64 characters long. Additionally, names can only start with an alphabetic character or an underscore.

Note:

There are certain group names that are reserved and therefore cannot be synchronized to Cloudera. See Reserved group names.

Note: Cloudera has default limits in place with regard to how many users, machine users, and groups can be added per account. Review User and group limits and make sure that you do not exceed these limits.

Sync Groups on Login enabled

When the Sync Groups on Login option is enabled, Cloudera synchronizes a user's group in the following manner:

- The group membership that your enterprise IdP specifies for a user overrides the group membership set up in Cloudera. Each time a user logs in, Cloudera updates the user's group membership based on the groups that your enterprise IdP specifies for the user.
- If the group exists in Cloudera, Cloudera adds the user to the group. The user takes on all the roles associated with the group.
- If the group does not exist in Cloudera, Cloudera creates the group and adds the user to the group. However, no roles are assigned to the new group, so a member of the new group does not take on roles from the group.
- If the user is a member of a group in Cloudera that is not included in the list provided by your enterprise IdP, Cloudera removes the user from the group.
- If the list of groups from your enterprise IdP is empty, Cloudera removes the user from all groups in Cloudera. After login, the user will not be a member of any Cloudera group and will not have roles from any group.

To ensure that users can perform tasks in Cloudera, Cloudera recommends that you set up the groups in Cloudera with appropriate roles before you assign them to users.

Sync Groups on Login disabled

When the Sync Groups on Login option is disabled, Cloudera does not synchronize the user's group membership in CDP with the user's group membership provided by the IdP. After login, a user's group membership in Cloudera is determined by the Cloudera groups assigned to the user in Cloudera. The groups assigned to the user in your enterprise IdP are ignored.

Sync Membership option for a newly created group

Additionally, once you have synced your IdP and you create a new group in Cloudera, you have an option called Sync Membership that determines whether group membership is synced to IdP when a user logs in. By default, Sync Membership is enabled when Sync Groups on Login is enabled.

The following table describes how the global Sync Groups on Login and the per-group Sync Membership options can be used:

	IdP Sync Groups on Login on	IdP Sync Groups on Login off
Group Sync Membership on	Group membership for the specific group is reflected in IdP.	Group membership for the specific group is not reflected in IdP.
Group Sync Membership off	Group membership for the specific group is not reflected in IdP.	Group membership for the specific group is not reflected in IdP.

In other words, if Sync Groups on Login is off at the IdP level, then no groups are getting synced regardless of what the setting for Sync Membership is. But if Sync Groups on Login is turned on at the IDP level, then you have the option to override it for certain groups that you explicitly leave off.

Updating an identity provider

You can update the group synchronization option and the provider metadata in a Cloudera identity provider. To update an identity provider in Cloudera, you must be a Cloudera account administrator or have the PowerUser role.

About this task

You might want to update the Cloudera identity provider to change the group synchronization option or if you want to update the list of x.509 certificates in the provider metadata.

Required role: Account administrator or PowerUser

Procedure

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. In the User Management section of the side navigation panel, click Identity Providers.
4. Find the Cloudera identity provider that you want to update.
5. Click the Actions button and select Update Identity Provider.
6. On the Identity Provider window:
   - You can change the Sync Groups on Login option.
   - You can add or edit the SAML Identity Provider Metadata.
   - You can check the Generate workload username by email box to have the workload username is generated based on the email instead of the default. See Generating workload usernames based on email.
   - You can enable SCIM for Azure AD. See Configure SCIM with Azure AD.
   - You cannot change the name of the Cloudera identity provider.
7. Verify the updates and click Update.

   Cloudera updates the information for the Cloudera identity provider.

Disabling the Cloudera SSO login

After you complete the identity federation setup between Cloudera and your enterprise IdP, you should disable the Cloudera SSO login option so that user logins are subject to your customer specific enterprise security policies and controls. Cloudera SSO login option is only meant to serve as the bootstrap IdP; it is not meant for enterprise user access controls and it lacks company specific policy compliance.

When you disable Cloudera SSO login for non-administrator users, Cloudera users must log in to Cloudera through the identity management system in your organization. Only the designated account administrator for your Cloudera subscription can log in to Cloudera through the Cloudera registration and login page.

Even after you disable the Cloudera SSO login, the designated account administrator can log in to Cloudera using their Cloudera SSO credentials. For added security, you can restrict all Cloudera SSO access (including the designated account administrator's access) by contacting Cloudera Support and they can disable or enable the "Cloudera SSO All Login Enabled" setting for your account. You can see "Cloudera SSO All Login Enabled" setting on the UI. When all Cloudera SSO logins are restricted, you will see this on the UI:

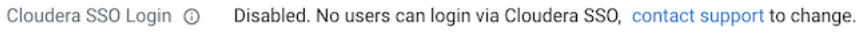

Required role: Account administrator or PowerUser

Steps

1. Sign in to the Cloudera web interface.
2. From the Cloudera home page, click Cloudera Management Console.
3. In the User Management section of the side navigation panel, click Identity Providers.

   The Identity Providers page shows the status of the Cloudera SSO Login option.
4. Click Disable to prevent users from logging in through the Cloudera registration and login page.

   When the Cloudera SSO Login option is disabled, all Cloudera users except the Cloudera account administrators must log in through the identity management system in your organization. To log in to Cloudera, a user must be among the users included in the identity providers that you set up in Cloudera.

Configuring Azure Active Directory identity federation in Cloudera

You can onboard users by configuring Azure Active Directory (Azure AD) identity federation with Cloudera.

Before you begin

- Cloudera requires the Azure AD sAMAccountName attribute for the SAML group claim mapping. As per Configure group claims for applications by using Azure Active Directory, sAMAccountName is not available with SAML on groups created in Azure AD; It is available only on groups created with on-premises AD and synced to Azure AD. If you only use Azure AD without on premises AD you can use SCIM to sync group memberships to Cloudera. See Configure SCIM with Azure AD.
- In order to be compatible with Cloudera, your Azure AD should be configured to synchronize with on-premises Active Directory via Azure AD Connect sync tool. As per Azure documentation, you must use Azure AD Connect 1.2.70.0 or newer.

Required role: PowerUser

Steps

1. Log in to Cloudera web interface and navigate to Cloudera Management Console> User Management, select the Identity Providers tab and click on Create Identity Provider to create an identity provider.

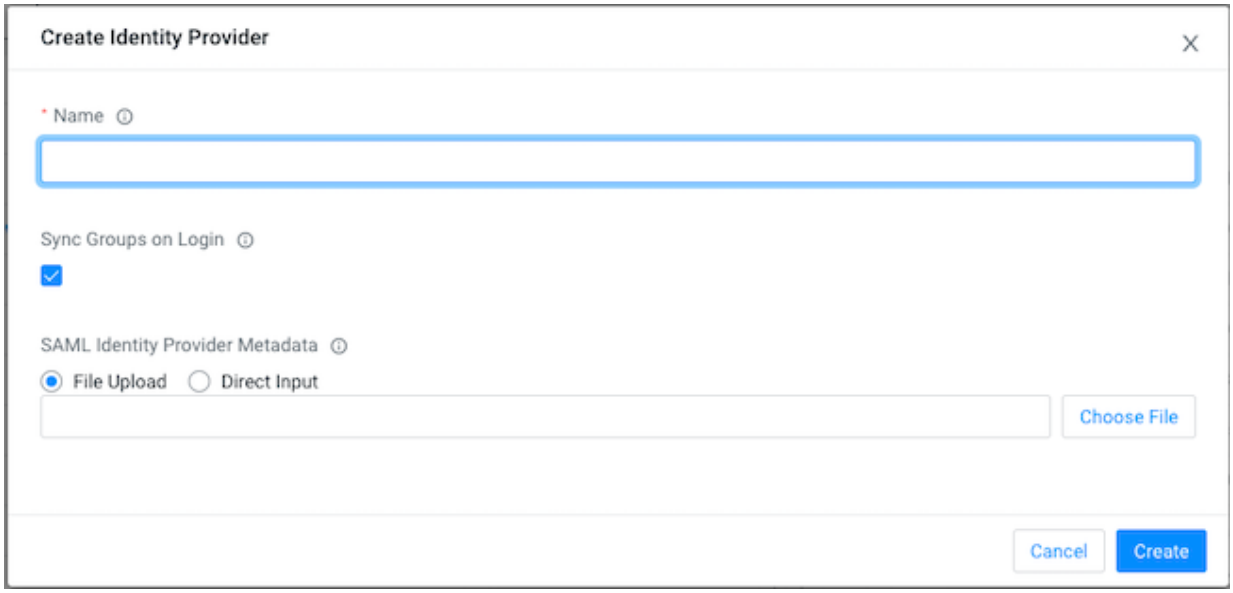

2. Name your identity provider in Cloudera, for example MyCompany_AAD.
3. Click Create.
4. Click on MyCompany_AAD on Cloudera console and copy the CDP SAML Service Provider Metadata to an XML file (for example, saml-metadata.xml). You will need it later.
5. Open another web browser window, navigate to https://portal.azure.com/, and log in to your Azure Portal.
6. On your Azure Portal, navigate to the Azure Active Directory.
7. Select the Enterprise applications service.
8. Click on the +New application button.

9. Select the Non-gallery application.

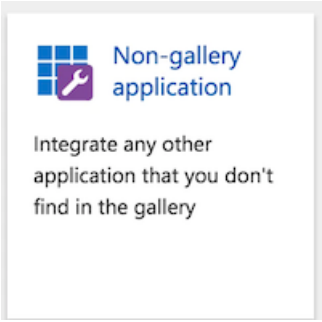

10. Give the application a name, for example. CDP_id_federation.
11. Click on the +Add button.
12. Once the application is added, go to 2. Set up single sign on .
13. Upload the metadata XML file that you saved in the earlier step.

14. Download the Federation Metadata xml for your Azure AD application and save it on your computer.

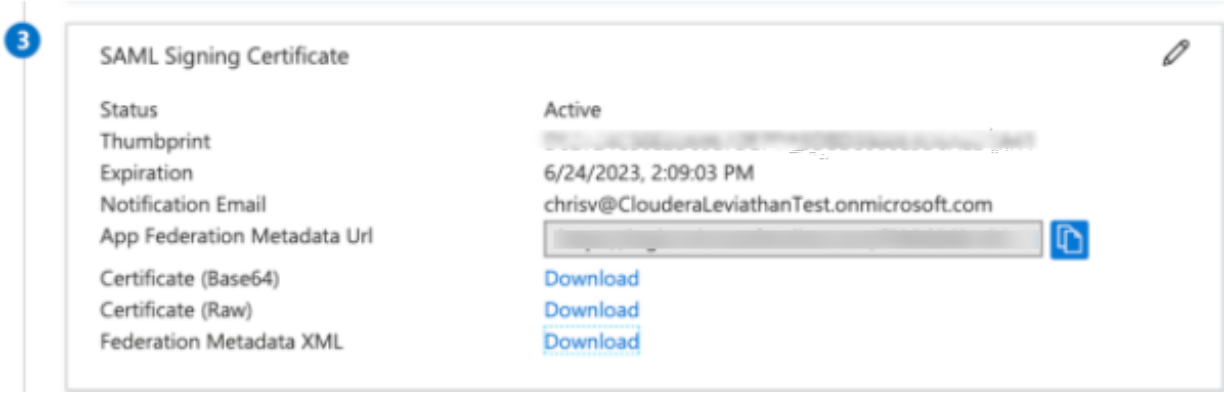

15. Switch back to Cloudera web interface and upload the metadata saved from AD and update the identity provider.
    a. Find the identity provider that you just created in Cloudera.
    b. Click the Actions button and select Update Identity Provider.
    c. On the Identity Provider window, upload the metadata XML file that you saved previously or copy and paste the content of that XML file:

    d. Verify the updates and click Update.
16. Switch back to Azure AD Azure Portal browser window.
17. Edit 1.Basic SAML Configuration:
    a. Make sure that the value for Identifier (Entity ID) is populated, for example “urn:cloudera:cdp:<Identity-Provider-Id>” or “urn:cloudera:altus” for legacy identity provider. Check the Service Provider Metadata to determine which identifier to use.
    b. From Cloudera SAML Service Provider Metadata you saved earlier, copy the AssertionConsumerService > Location value and paste it into the line Reply URL (Assertion Consumer Service URL).

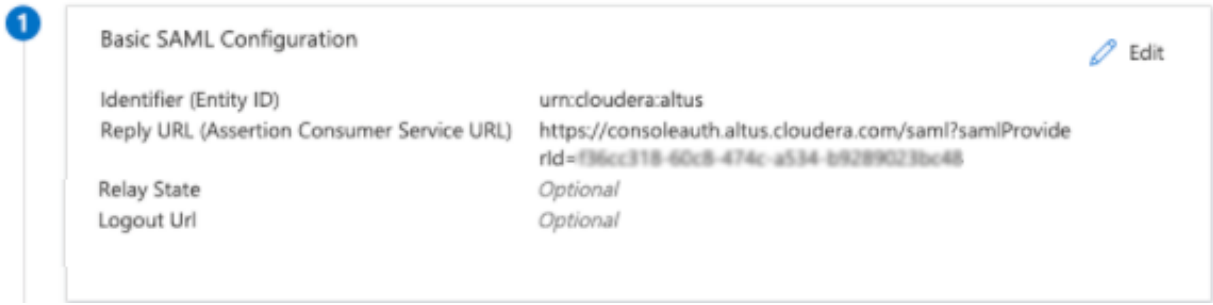

18. Edit 2.User Attributes & Claims:
    a. If the customer is using on-prem Active Directory and Active Directory Connect to sync with Azure AD, you will be able to import Azure AD groups into Cloudera. Click +Add a group claim.
    b. On the Group Claims blade, do the following:
        1. Select Security groups or Groups assigned to the application.
        2. Select Source Attribute sAMAccountName.
        3. Check the Customize the name of the group claim. Optionally configure Group filtering for fine control of the list of groups that are included as part of the group claim. When a filter is configured, only groups

that match the filter will be included in the group's claim that is sent to Cloudera. For more information, see Azure AD group filtering.

4. Enter “groups” in Name (Required).
5. Namespace enter https://cdp.cloudera.com/SAML/Attributes.

Group Claims

Manage the group claims used by Azure AD to populate SAML tokens issued to your app

Which groups associated with the user should be returned in the claim?

- ○ None
- ○ All groups
- ◉ Security groups
- ○ Directory roles
- ○ Groups assigned to the application

Source attribute *

sAMAccountName

⚠ This source attribute only works for groups synchronized from an on-premises Active Directory using AAD Connect Sync 1.2.70.0 or above. Learn More

Advanced options

☐ Filter groups

Attribute to match

Match with

String

☑ Customize the name of the group claim

Name (required)

groups

Namespace (optional)

https://cdp.cloudera.com/SAML/Attributes

☐ Emit groups as role claims ⓘ

☐ Apply regex replace to groups claim content

6. Click Save.

c. For the rest of the claims, follow the instructions at Configuring your enterprise IdP to work with CDP.

If your NameID is an opaque ID (such as a UUID), you can Generate workload usernames based on email.

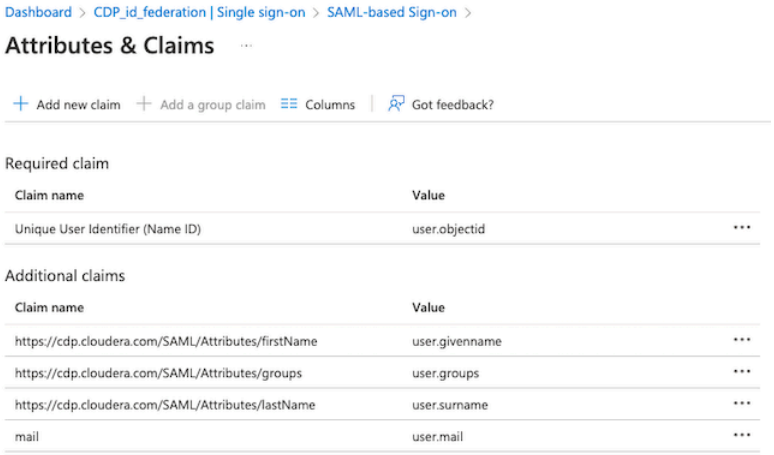

Name	Namespace	Source	Source Attribute
Unique User Identifier (Name ID)		Attribute	user.objectid
firstName	https://cdp.cloudera.com/SAML/Attributes	Attribute	user.givenname
lastName	https://cdp.cloudera.com/SAML/Attributes	Attribute	user.surname
mail		Attribute	user.mail

19. Click on the SAML-based Sign-On on the top.
20. Test this application.
21. Once these steps are completed, a Cloudera user will be able to log in with their integrated Azure AD identity through their Office 365 applications page (Office.com). A new tile will appear for the Cloudera application created above:

Once a user signs in, the User and Groups will show up on the Cloudera Management Console's User Management screen. Navigate to your office 365 applications page, click on the new tile that was created for Cloudera, and verify that you are able to log in.

22. Once a user signs in, the User and Groups will show up on the Cloudera Management Console's User Management screen.

What to do next:

- Assign users to groups within your Azure AD that you will map to roles in Cloudera.
- Assign Cloudera roles to either the new users or the groups as appropriate.

Related Information

Configure group claims for applications with Azure Active Directory
Configure SCIM with Azure AD

Configure SCIM with Azure AD

Cloudera supports SCIM with Microsoft Azure Active Directory (Azure AD).

SCIM is a common way to get around:

- The Azure AD SAML 150 groups-per-claim limit.
- The Azure AD SAML sAMAccountName not available on groups created in Azure AD limitation (this is where group names are sent as their Object IDs in SAML instead of their human readable name).

For more information on these limitations, see Configure group claims for applications by using Azure Active Directory.

Refer to this documentation if you would like to configure Cloudera to use SCIM. Prior to configuring Cloudera to use SCIM, you should be aware of the following limitations:

- You can only configure one identity provider per Cloudera account to use SCIM.
- Once you start using SCIM, you should not update users and groups in Cloudera, as they will get out of sync with Azure AD, and you may notice unexpected changes if/when Azure AD realizes the differences and attempts to re-sync the users/groups.
- Updating group names is not supported in Cloudera.
- Updating userName is not supported in CDP. You must use an Azure AD field that will not change as a user Name to map via SCIM to Cloudera. If that field is an opaque ID (for example, a UUID) then you should generate workload usernames from email as described below.

Once you are aware of the limitations, you can proceed to configuring Cloudera to use SCIM. The steps include:

- Prerequisites on page 19
- Enable SCIM for your identity provider in Cloudera on page 19
- Set up SCIM in Azure AD on page 20

Prerequisites

Prior to configuring Cloudera to use SCIM, ensure that you can meet the following requirements:

- Prior to configuring Cloudera to use SCIM, you should Configure Azure AD in CDP. SCIM is meant to be used in conjunction with SAML identity federation to synchronize users and groups from your identity provider to Cloudera.
- Setting up SCIM requires administrative operations in both Cloudera and Azure AD, and so it requires an Azure AD admin to perform the Azure AD steps.
- Additionally, you need to be a Cloudera account administrator or have the PowerUser role in Cloudera.

Enable SCIM for your identity provider in Cloudera

This task involves enabling SCIM for your identity provider and getting the SCIM URL that you will need to configure Azure AD so it can connect to Cloudera. Next, you create an access token for Azure AD to securely communicate with Cloudera for SCIM and get the access token secret you will need to configure Azure AD.

You will need to provide the lifetime for the access token used for Azure AD to communicate to Cloudera. Typical values are 1 year or 3 years; consult your security policies. You will need to rotate the access token before it expires.

In order to perform these steps, you need to be a Cloudera account administrator or have the PowerUser role in Cloudera.

Steps

1. Sign in to the Cloudera console.
2. Navigate to the Cloudera Management Console.
3. Select User Management from the left pane.
4. Click on Identity Providers.

5. Select your identity provider, and from the ⋮ (context menu) select Update Identity Provider.
6. On the Update Identity Provider window, check the box for Enable SCIM. This will update the identity provider to be ready to accept SCIM API calls.
7. (Optional) If you use an opaque ID for SAML NameID and SCIM userName, on the Update Identity Provider window, check the box for Generate workload username by email. Check this box if you use an opaque ID for SAML NameID and SCIM userName. For more information, see Generating workload usernames based on email.
8. Click Update.
9. Select your identity provider, and from the ⋮ (context menu) select View Identity Provider.
10. Copy the SCIM URL. You will need it later.
11. Click Close.
12. Select your identity provider and from the ⋮ (context menu) select Update SCIM Access Tokens.
13. On the Update SCIM Access Tokens window, click the button to Create SCIM Token.
14. Set your lifetime, in days. This is how long you will have before you need to rotate your SCIM access token. Note that you can always rotate your tokens earlier than their expiration date and you can revoke tokens at any time.
15. Click the Create button. This creates a SCIM access token that is used to authenticate SCIM API calls to this identity provider.
16. Your Access Token Secret will be shown. Copy it somewhere. You will need it later and it will not be shown again.

Set up SCIM in Azure AD

After enabling SCIM for your identity provider in Cloudera, you should set up SCIM in Azure AD. These steps must be performed by an Azure AD admin.

Steps

1. Sign in to the Azure Portal.
2. Navigate to Azure AD and then click on Enterprise Applications.
3. Select the application you used to configure identity federation to Cloudera. Or, if you are setting up SCIM in a new Azure AD app, create a new non-gallery enterprise application.
4. Under the Manage menu on the left, click Provisioning.
5. From the Provisioning Mode dropdown, select Automatic.
6. Expand the Admin Credentials section.
7. In the box for Tenant URL, paste in the SCIM URL that you saved earlier.
8. In the box for Secret Token, paste in the Access Token Secret that you saved earlier.
9. Click Test Connection and wait for success.
10. Click Save.
11. Still in the same Provisioning blade, expand the Mappings section. This section is grayed out until the connection has been successfully tested.
12. Click on Provision Azure Active Directory Groups.
13. Update the Attribute Mappings as follows:

Azure Active Directory Attribute	customappsso Attribute	Matching precedence
displayName	displayName	1

members	members	

As an outcome, your configuration should look similar to:

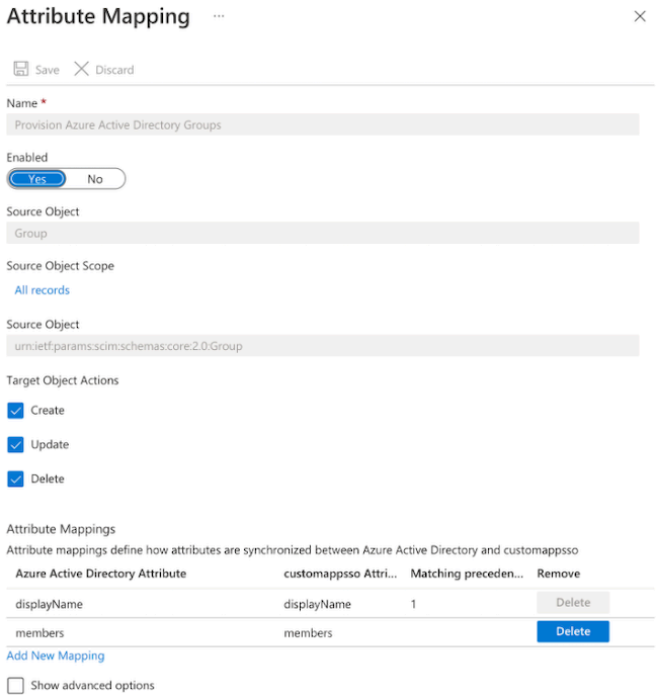

Note: Updating group names is not supported in CDP.

14. Click Save to save the group attribute mappings.
15. Close this blade to return to the Provisioning blade.
16. Click on Provision Azure Active Directory Users.
17. Update the Attribute Mappings as follows:

Azure Active Directory Attribute	customappsso Attribute	Matching precedence
objectId	userName	1
mail	emails[type eq "work"].value	
givenName	name.givenName	

surname	name.familyName	

As an outcome, your configuration should look similar to:

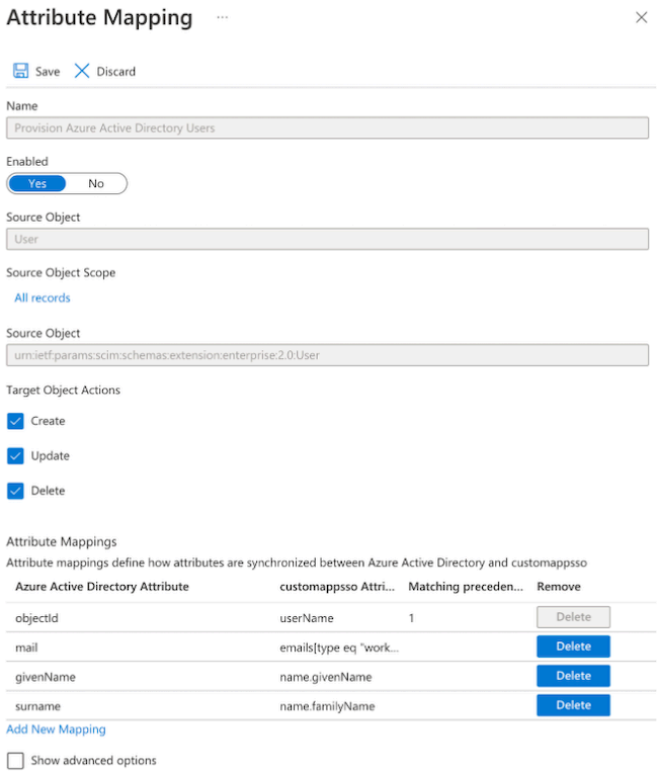

Note: The "customappsso Attribute" userName must be the same as your SAML NameID attribute. The value of the field mapped to NameID/userName is used to uniquely identify a user in Cloudera and must also be immutable, or you will run into known issues. See Known issues and troubleshooting related to IdP setup in CDP.

18. Click Save to save the user attribute mappings.
19. Close this blade to return to the Provisioning blade.
20. Click Save in the Provisioning blade.
21. Still in the Provisioning blade, expand the Settings section.
22. Set Scope to Sync only assigned users and groups.
23. Click Save.
24. Now in the blade for the application, under the Manage menu click Users and groups.
25. If there are no users or groups, add a few. You will use this to test SCIM later.
26. Back in the blade for the application, under the Manage menu click Provisioning.
27. Click Start provisioning.
28. Wait for the sync cycle to run.

    The initial provisioning cycle may take a few minutes to start but subsequent provisioning cycles run using a fixed cadence. You can see the cadence under the View provisioning details expandable section.

    Note: Provisioning cycles are set by Azure AD and are not customizable.

29. Once the provisioning cycle is complete, check your users and groups in Cloudera. If there are any errors, contact your Cloudera representative.

Attention:

- Cloudera does not treat PATCH requests as atomic. PATCH operations are idempotent on Cloudera.
- Querying resources using HTTP POST is not supported.

Related Information

Configuring Azure Active Directory identity federation in Cloudera

Known issues and troubleshooting related to IdP setup in Cloudera

Importing or uploading users

You can bulk import users so as to assign policies to users and groups without requiring the users to log in at least once.

Before you begin

- Make sure the identity provider user ID matches with the NameId attribute value that is passed for the user in the SAML response using the associated SAML provider.
- Cloudera has default limits in place with regard to how many users, machine users, and groups can be added per account. Review User and group limits and make sure that you do not exceed these limits.

Required role: PowerUser

Steps

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. Click User Management in the left navigation panel.

   The Users page displays the list of all Cloudera users.
4. Click Actions.
5. From the dropdown list that appears, click Upload Users.

   The Upload Users page appears.
6. Select an identity provider from the dropdown list.
7. Select an option for adding user details:
   - File Upload - If you want to upload a CSV file with the details of the users that you like to add.
   - Direct Input - Enter the user details in the format specified. Make sure you add the header row as specified in the sample.
8. Click Next.
9. In the Preview Users screen that appears, verify if all the details are uploaded or added accurately.
10. Click Upload.
11. Confirmation of success is shown. Click Finish.

You can view newly added users from the Users tab.

Generating workload usernames based on email

-->Cloudera offers an option to generate workload usernames for Cloudera users based on user email addresses.

By default, workload usernames are generated using the identity provider user ID. For SAML logins that is the SAML NameID, for SCIM that is the SCIM userName, and when using the Cloudera APIs that is the identity-provide r-user-id. Sometimes the identity provider user ID is an opaque ID, like a uuid or employee ID, which gives equally opaque workload usernames.

Alternatively, you can generate workload usernames based on users' email addresses instead of using the default workload usernames. For example, if your identity-provider-user-id is 8d16a2ea, and your email is bob@example.com, by default your workload username will be "8d16a2ea". If you choose to generate workload usernames by email, your workload username will instead be "bob".

Note: Once your Cloudera users are created and have the default or email-based workload usernames assigned, you cannot change the workload usernames already generated before the setting was changed. At that point, changing this setting does not change or regenerate any existing workload usernames. Specifically, for SAML logins users are created in Cloudera when they log in for the first time, and for SCIM users are created when the identity provider runs a SCIM sync cycle.

Steps

For CDP UI

When creating or updating an identity provider in Cloudera, you can check the Generate workload username by email box to have workload usernames generated based on email addresses.

For CDP CLI

From the CDP CLI, you can change how workload usernames are generated when you create (iam create-saml-provider) or update (iam update-saml-provider) a SAML provider by using the --generate-workload-username-by-email or --no-generate-workload-username-by-email flags. See:

```
iam create-saml-provider --help
iam update-saml-provider --help
```

Related Information

Setting up the identity provider in Cloudera

Updating an identity provider

Known issues and troubleshooting related to IdP setup in Cloudera

This topic covers known issues that you may encounter when setting up an identity provider in Cloudera and steps to troubleshoot them.

Known issues with mutable SAML NameID or SCIM userName

Issue (If using SAML without SCIM):

If you use SAML without SCIM, and you set the SAML NameID field to a mutable Azure AD field (such as an email), then you will end up with duplicate users in Cloudera when the Azure AD value changes. The duplicate user is a new, different user that has a different Cloudera workload username. This is because SAML has no way to differentiate between two different users and a user whose SAML NameID has changed.

Issue (If using SAML with SCIM):

If you are using both SAML and SCIM, you must set the SAML NameID and the SCIM userName fields to the same Azure AD field. If that Azure AD field is a mutable field, when it changes you will end up with duplicate users in Cloudera. This will cause errors in your Azure AD Enterprise Application and SCIM updates to the user will fail.

Workaround:

There is no automatic way to recover and you will likely have to delete and recreate the affected user. This includes having to delete and recreate the user's permissions, passwords, keys, and so on. If this happens, contact Cloudera support.

Known issues with Azure AD User Principal Name (UPN)

While uncommon in practice, the Azure AD UPN is actually a mutable field. Ask your Azure AD team if your organization mutates UPN before using it.

By default, when users are deleted in Azure AD they are moved to a "recently deleted" list for 30 days, before they are permanently deleted and removed from Azure AD. When a user is moved to the "recently deleted" list, their UPN is automatically changed by Azure AD. This will cause SCIM errors to show up in your Azure AD Enterprise

Application: The error is Azure AD trying to update the userName field, and Cloudera returning an error saying that operation is not supported. These errors can be ignored.

Note that even though users in Azure AD are in the "recently deleted" list, they will persist in Cloudera as active users until after the 30 day wait time when they are fully removed from Azure AD (deleted from the Azure AD system). At that point SCIM will delete them from Cloudera as well. If you permanently delete the user from Azure AD before the 30 day period (that is, delete the user from the "recently deleted" list), then that user will also be deleted in Cloudera. If neither of these options work for you, then you must delete the user manually in Cloudera.

Known issues with updating group names

Updating group names is rare - most organizations do not do this. Cloudera does not support updating group names. To fix the issue, create a new group instead.

Understanding Cloudera user accounts

User accounts identify the users who can access services, applications, and components in Cloudera.

Roles assigned to a user account determine the actions that the user can perform in Cloudera.

There are four types of user accounts in Cloudera:

- Cloudera account administrator
- Cloudera user
- Cloudera workload user
- Cloudera machine user

Cloudera account administrator

During the initial setup of the Cloudera subscription for a customer, Cloudera designates a user account as a Cloudera account administrator.

A Cloudera account administrator has administrator privileges in Cloudera. The Cloudera account administrator user account cannot be managed within Cloudera. You must contact Cloudera support to add or remove an account administrator from your Cloudera account.

As an account administrator, you have all the privileges in Cloudera and can perform any task in Cloudera. You can set up users and assign roles, services, and environments to users in Cloudera according to the tasks that they need to perform. You can set up another user as a Cloudera administrator by assigning the PowerUser role to the user. However, you cannot set up another user as a Cloudera account administrator.

A Cloudera account administrator requires a Cloudera user account. To be designated as a Cloudera account administrator, you must register for a Cloudera user account. To register for a Cloudera user account, go to the Cloudera Account Registration page and create an account.

Cloudera user

To perform tasks using Cloudera and its services, you must be a Cloudera user and roles and resources need to be assigned to this user.

Cloudera allows users within your organization to log in to Cloudera through the authentication system in your organization without registering with Cloudera or creating a Cloudera account. During the initial process of configuring the environment, the account administrator must set up identity federation and thus automatically add users.

When a Cloudera user who is not an account administrator logs in to Cloudera for the first time, the user has limited privileges. A Cloudera administrator must assign the appropriate roles to the user after the initial user login.

The Cloudera account administrator can delete the user accounts. Deleting a user removes all access keys and SSH keys associated with the user, and unassigns all roles and resource roles assigned to the user. The user is also removed from all groups that they belong to.

A user who has a valid account in Cloudera but is not assigned any role can perform a limited number of tasks. A user who logs in to the Cloudera console without an assigned role or environment can perform only the following tasks:

- Download the Cloudera client.
- View the Cloudera documentation.

Cloudera workload user

You need the workload username to access Cloudera Data Hub clusters and non-SSO interfaces, and to SSH to clusters.

Cloudera assigns workload username to all Cloudera users when the user is created in Cloudera. The workload username is generated from either the identity provider user ID (default) or the email (configurable via Generate workload usernames). If a workload user name already exists in Cloudera, which can occur when multiple identity providers are mapped, the workload username will contain a numeric suffix. Workload usernames are immutable - once set they never change.

To identify your workload username, navigate to Home > in the bottom left corner click on your user name > Profile > Workload User Name.

Alternatively, you can obtain your workload username by following these steps:

1. Sign in to the Cloudera web interface.
2. From the Cloudera home page, click Cloudera Management Console.
3. Click User Management in the left navigation panel. The Users page displays the list of all Cloudera users.
4. Search for your username.
5. In the list of users that appears, you can see the Workload User Name column. Your workload username will appear under this column.

   You can set your workload password to get access to non-SSO interfaces.

Cloudera machine user

A machine user account provides programmatic access to Cloudera. Create a machine user account if you have an application that needs to access the Cloudera services with the CLI or the Cloudera SDK for Java.

You can define the machine user account in your application to create and manage clusters and run jobs in Cloudera using the CLI or API commands.

You can create and manage a machine user account within Cloudera. You must assign an API access key to a machine user account to enable it to access the Cloudera service with the CDP CLI or Cloudera SDK. You must assign roles to a machine user account to authorize it to perform tasks in Cloudera.

A machine user account does not have an associated Cloudera user account. You cannot use a machine user to log in to the Cloudera console.

Use the following guidelines when you manage user accounts in Cloudera:

- When you create a machine user account, you assign roles and environments to the machine user account in the same way that you assign roles and environments to other user accounts.
- You can revoke permissions for a Cloudera machine user account by removing the machine user from groups or unassigning account level or resource roles.
- Deleting a machine user removes all access keys and SSH keys associated with the machine user, and unassigns all roles and resource roles assigned to the machine user. The machine user is also removed from all groups that they belong to.

Understanding account roles and resource roles

To access resources and perform tasks in Cloudera, each user requires permissions. As a Cloudera administrator, you can assign a role to a user or a machine user to give the user permission to perform the tasks either on the whole account or on a specific resource.

Each role has an attached policy that defines the permissions associated with the role. The policy attached to a role determines the operations that the role allows the user to perform. When users attempt to perform operations that are not permitted in their assigned role, they get a permission denied error message.

Cloudera has predefined roles for your use. You can assign a role or a combination of roles to give the user the appropriate permissions to complete tasks in Cloudera. You cannot modify the predefined Cloudera roles or the policies associated with the predefined roles.

The scope of predefined roles and resource roles can vary. For example, a role might grant view access only to Cloudera Data Hub clusters but not to environments in which these clusters are running. You might need to assign multiple roles to ensure that a user can perform all required tasks in Cloudera.

Cloudera provides the following types of roles:

- Account roles - An account role grants a user, machine user, or group permissions to access or perform tasks on all resources within the Cloudera tenant.
- Resource roles - A resource role grants a user, machine user, or group permissions to access or perform tasks on a specific resource (such as a specific environment or a specific Cloudera Data Hub cluster).
- Group membership administration roles - The IamGroupAdmin role can be assigned to a user to manage group membership for a specific group.

Review the following documentation to learn more about these role types:

Account roles

A Cloudera role grants permissions to perform tasks in Cloudera that are not associated with a specific resource. You explicitly assign a role to a user, machine user, or a group.

When assigning roles to users and groups, consider the following:

- Only PowerUser can assign account roles.
- A user needs the following two types of roles in order to assign access to resources to other users:
  - One of the roles that allow role assignment: EnvironmentCreator, EnvironmentAdmin, DataSteward, DatahubAdmin or another admin role for a Cloudera service.
  - One of the roles that allow listing users within the organization: IamUser or IamViewer.
- All users who need to access CDP CLI need the IamUser role.

Each role is identified by a CRN, which uses the following format:

```
crn:altus:iam:<CONTROL_PLANE_REGION>:altus:role:<ROLE_NAME>
```

For example, the following is the IamViewer role CRN:

```
crn:altus:iam:us-west-1:altus:role:IamViewer
```

You can view all available roles and their CRNs by using the cdp iam list-roles command.

Account roles can be assigned from the Cloudera Management Console> User Management > Roles tab or from CDP CLI by using the cdp iam assign-user-role or cdp iam assign-group-role commands.

The predefined account roles available in Cloudera are as follows:

Table 1: Account roles

Account role	Description	Important considerations
PowerUser	Grants permission to perform all tasks on all resources.	Unlike other users (who only see the resources that they are authorized to list), Power Users can list all resources. By default, Power Users don't not have full access to all resources but can assign themselves a resource role that grants them access to these resources.
EnvironmentCreator	Grants permission to create environments and shared resources (cluster templates, recipes, image catalogs, credentials, proxies), and sync users.	Since shared resources are managed separately from environments, in order for a user with the EnvironmentCreator role to be able to use a provisioning credential for creating an environment, that user needs to be Owner or SharedResourceUser for that credential. EnvironmentCreator is the only role that allows you to manage access to proxies that have been registered in Cloudera.
IamUser	Grants permission to create access keys and upload SSH keys for the user (but not for other users). Moreover, this role includes all permissions of the IamViewer role. It grants permission to view all users in the account and their assigned roles and access keys.	Either the IamUser or IamViewer role is required to list other users, therefore any user who needs to assign roles, such as EnvironmentCreator, EnvironmentAdmin, DataHubAdmin, and so on, should be assigned either IamUser or IamViewer.
IamViewer	Grants permission to view all users in the account and their assigned roles and access keys.	Either the IamUser or IamViewer role is required to list other users, therefore any user who needs to assign roles, such as EnvironmentCreator, EnvironmentAdmin, DataHubAdmin, and so on, should be assigned either IamUser or IamViewer.
ClassicClustersCreator	This role is required to register a new cluster. If this role is not present then the "Add Cluster" button is not visible to the user.	
DataCatalogCspRuleManager	Grants permission to perform all tasks on CSP rules in Cloudera Data Catalog.	
DataCatalogCspRuleViewer	Grants permission to list and view CSP rules in Cloudera Data Catalog.	
DFCatalogAdmin	Grants permission to perform all tasks on objects stored in the Cloudera DataFlow Catalog. This includes importing and deleting flow definitions, as well as uploading new versions of existing flow definitions.	
DFCatalogViewer	Grants permission to browse the Cloudera DataFlow Catalog and view flow definitions.	
BillingAdmin	Grants permission to monitor Cloudera credit consumption.	

Related Information

Assigning account roles to users

Assigning account roles to groups

Resource roles

A role that is associated with a specific resource is called a resource role. This type of role gives permission to perform tasks on a specific resource, such as a specific Cloudera environment, shared resource, or Cloudera Data Hub cluster.

When you assign a resource role, you must specify the resource on which to grant the resource role permissions. For example, you can assign a user a resource role that grants permission on an environment. The user assigned the resource role can list, access, and perform tasks only on that environment, but not on other environments.

A resource role determines a specific set of tasks that the user can perform on the resources. For example, the EnvironmentUser resource role assigned to a user allows the user the rights contained in the resource role only on that particular environment.

The predefined resource roles available in Cloudera that you can assign to Cloudera users, machine users, and groups are as follows:

- Environment resource roles
- Shared resource resource roles
- Cloudera Data Hub resource roles
- Classic cluster resource roles
- The Owner resource role (available on all resources)

Each role is identified by a CRN, which uses the following format:

```
crn:altus:iam:<CONTROL_PLANE_REGION>:altus:resourceRole:<RESOURCE_ROLE_NAME>
```

For example, the following is the DataHubAdmin role CRN:

```
crn:altus:iam:us-west-1:altus:resourceRole:DataHubAdmin
```

You can view all available roles and their CRNs by using the cdp iam list-resource-roles command.

Learn more about different resource role types:

- Environment resource roles on page 29
- Shared resource resource roles on page 32
- Cloudera Data Hub resource roles on page 32
- Classic cluster resource roles on page 33
- The Owner resource role on page 33

Environment resource roles

Environment resource roles can be assigned on the scope of a specific environment.

These resource roles can be assigned from the Cloudera Management Console > Environments > navigate to a specific environment > Actions > Manage Access > Access or from CDP CLI using the cdp iam assign-user-resource -role command.

Table 2: Environment resource roles

Resource role	Description	Important considerations
EnvironmentAdmin	Grants all rights to the environment and Cloudera Data Hub clusters running in it, except the ability to delete the environment.	The user who created the environment automatically gets the EnvironmentAdmin role on the scope of that environment. The EnvironmentAdmin resource role is assigned the Limited Cluster Administrator role in Cloudera Manager. Users with this role can manage the cluster lifecycle, change configurations, and manage parcels. For more information on Cloudera Manager roles, see the topic Default User Roles. The Cloudera Manager Limited Cluster Administrator role is assigned to the EnvironmentAdmin because the Cloudera Control Plane is responsible for certain tasks historically done in Cloudera Manager, for example: adding or removing hosts as part of up/down-scaling and repair operations, executing upgrades of clusters in coordination with upgrading the OS images used by clusters, and creating new clusters based on templates preconfigured to work in a Cloudera environment. In addition, only selected services and workload types are currently supported in Cloudera Data Hub, represented by the built-in cluster definitions. Finally, certain Cloudera services like encryption-at-rest infrastructure are explicitly not designed for use in the public cloud, where the cloud provider's object store encryption capabilities should be used. Because of this, Cloudera Data Hub is prescriptive in its choice of workload types and theCloudera Control Plane is best suited to manage most cluster life cycle operations. Doing so directly in Cloudera Manager could lead to unexpected operational issues. Cloudera Data Hub does, however, support fully customizable cluster templates. EnvironmentAdmin can manage access to the environment by assigning a user EnvironmentAdmin, DataSteward, or EnvironmentUser role.
EnvironmentPrivilegedUser	Grants permission to execute privileged operating system actions on Data Lake, FreeIPA, and Cloudera Data Hub virtual machines.	
EnvironmentUser	Grants permission to view Cloudera Data Hub clusters and set the workload password for the environment. The EnvironmentUser resource role is assigned the Read-Only role in Cloudera Manager. For more information on Cloudera Manager roles, see the topic Default User Roles.	This role should be used in conjunction with service-specific roles such as DataHubAdmin, DWAdmin, DWUser, MLAdmin, MLUser, and so on. When assigning one of these service-specific roles to users, make sure to also assign the EnvironmentUser role.
DataSteward	Grants permission to perform user/group management functions in Ranger and Atlas Admin, manage ID Broker mappings, and start user sync for the environment.	DataSteward can manage access to the environment by assigning a user DataSteward or EnvironmentUser role.
DataHubCreator	Grants permission to create Cloudera Data Hub clusters in the environment.	

Resource role	Description	Important considerations
DEAdmin	Grants permission to create, delete and administer Cloudera Data Engineering services for the environment.	When assigning this role, you should also assign the EnvironmentUser role.
DEUser	Grants permission to list and use Cloudera Data Engineering services for the environment.	When assigning this role, you should also assign the EnvironmentUser role.
DFAdmin	Grants permission to enable, disable and administer the Cloudera environment for Cloudera DataFlow. This includes granting and revoking the ability to access the Cloudera DataFlow Kubernetes API server.	When assigning this role, you should also assign the EnvironmentUser role.
DFFlowAdmin	Grants permission to create, terminate, administer and monitor running deployments for the environment.	When assigning this role, you should also assign the EnvironmentUser role.
DFFlowDeveloper	Grants permission to view, create, modify, or delete flow drafts; start and end test sessions in an environment.	When assigning this role, you should also assign the EnvironmentUser role.
DFFlowUser	Grants permission to view and monitor deployments for the environment.	When assigning this role, you should also assign the EnvironmentUser role.
DFProjectCreator	Grants permission to create a Cloudera DataFlow Project within a given Cloudera environment.	When assigning this role, you should also assign the EnvironmentUser role.
DWAdmin	Grants permission to activate/terminate or launch/stop/update services in Database Catalogs and Virtual Warehouses.	When assigning this role, you should also assign the EnvironmentUser role.
DWUser	Grants permission to view and use Cloudera Data Warehouse clusters within the environment.	When assigning this role, you should also assign the EnvironmentUser role.
MLAdmin	Grants permission to create and delete Cloudera AI workbenches within the environment. MLAdmins will also have Site Administrator access to all the workbenches provisioned within this environment. They can run workloads, monitor, and manage all user activity on these workbenches.	When assigning this role, you should also assign the EnvironmentUser role.
MLBusinessUser	Grants permission to view Cloudera AI workbenches for the environment. MLBusinessUsers are granted view-only access to applications that have been shared with them through projects inside a workbench.	When assigning this role, you should also assign the EnvironmentUser role.
MLUser	Grants permission to view Cloudera AI workbenches provisioned within the environment. MLUsers are also able to run workloads on all the workbenches provisioned within this environment.	When assigning this role, you should also assign the EnvironmentUser role. MLUsers currently require the SharedResourceUser role on the cloud credential used for the environment.
NotificationDistributionListAdmin	Grants permission to view, create, modify, and delete Distribution Lists for resource notifications.	
NotificationSubscriber	Grants all rights for managing individual resource subscriptions and viewing resource notifications	Ensure that you also have the required resource roles of the service to enable resource notifications.
ODAdmin	Grants permission to create, drop and administer the Cloudera Operational Databases for the environment.	When assigning this role, also assign the DataSteward or EnvironmentAdmin role.

Resource role	Description	Important considerations
ODUser	Grants permission to list and use Cloudera Operational Databases for the environment.	
Owner	Grants all permissions required to manage the environment in Cloudera including the ability to delete it.	The user who created the environment automatically gets the Owner role on the scope of that environment. The Owner role on the scope of an environment allows you to delete that environment, but to access the environment's clusters (Data Lakes, Cloudera Data Hub clusters), you need EnvironmentAdmin or EnvironmentUser.

Shared resource resource roles

Shared resources resource roles can be assigned on the scope of a specific shared resource such as a credential, cluster template, image catalog, proxy, or recipe. This does not include default shared resources (such as default cluster templates), which can be seen by everyone who is able to access the account.

These resource roles can be assigned from the Cloudera Management Console > Environments > Shared Resources > select a shared resource > navigate to a specific shared resource > Manage Access, or from CDP CLI using the cdp iam assign-user-resource-role command.

You can view all available resource roles and their CRNs by using the cdp iam list-resource-roles command.

Table 3: Shared resource resource roles

Resource role	Description	Important considerations
SharedResourceUser	This role enables shared resource sharing with other users. It grants permission to access and use the specific shared resource such as a specific cluster template, credential, image catalog, proxy, or recipe.	In order for a user to be able to use a provisioning credential for creating an environment, that user needs to be Owner or SharedResourceUser for that credential.
Owner	Grants all permissions required to manage the shared resource in Cloudera including the ability to delete it.	The user who created the shared resource automatically gets the Owner role on the scope of that shared resource. In order for a user to be able to use a provisioning credential for creating an environment, that user needs to be Owner or SharedResourceUser for that credential.

Cloudera Data Hub resource roles

Cloudera Data Hub resource roles can be assigned on the scope of a specific Cloudera Data Hub cluster.

Note: While full access to manage a Cloudera Data Hub via the Cloudera Management Console can be granted via assigning the Owner role on the scope of the Cloudera Data Hub, access to the underlying cluster can only be granted by assigning the EnvironmentAdmin or EnvironmentUser role on the scope of the environment where the Cloudera Data Hub is running.

These resource roles can be assigned from the Management Console > Data Hub clusters > click on a cluster > Actions > Manage Access, or from CDP CLI using the cdp iam assign-user-resource-role command.

You can view all available roles and their CRNs by using the cdp iam list-resource-roles command.

Table 4: Cloudera Data Hub resource roles

Resource role	Description	Important considerations
DataHubAdmin (Technical Preview)	Grants administrative rights over the Cloudera Data Hub cluster, such as start, stop, scale, repair and grant or revoke access.	When assigning this role, you should also assign the EnvironmentUser role. Granting DataHubAdmin role does not grant Cloudera Manager admin rights or Cloudera Runtime service admin rights (for example NiFi Registry Admin).
Owner	Grants all permissions required to manage the Cloudera Data Hub in Cloudera including the ability to delete it.	The user who created the Cloudera Data Hub automatically gets the Owner role on the scope of that Cloudera Data Hub. The Owner role does not grant any cluster-level permissions such as the ability to access or manage a cluster via Cloudera Manager. In order to access Cloudera Data Hub clusters running within an environment, you should assign EnvironmentUser to a user or a group on the scope of that environment.

Classic cluster resource roles

Classic cluster resource roles can be assigned on the scope of a specific classic cluster.

These resource roles can be assigned from the Cloudera Management Console > Classic clusters > context menu > Manage Access, or from CDP CLI using the cdp iam assign-user-resource-role command.

You can view all available roles and their CRNs by using the cdp iam list-resource-roles command.

Table 5: Classic cluster resource roles

Resource role	Description	Important considerations
ClassicClusterAdmin	Grants permission to perform any operation on the cluster, except deleting it. Grants permission to assign access to the cluster to other users.	
ClassicClusterUser	Grants permission to access details of the cluster.	
Owner	Grants all permissions required to manage the classic cluster in Cloudera including the ability to delete it.	The user who created the classic cluster automatically gets the Owner role on the scope of that classic cluster. The Owner role does not grant any cluster-level permissions such as the ability to access or manage a cluster via Cloudera Manager

The Owner resource role

In addition to the aforementioned resource roles, Cloudera includes the Owner resource role.

The Owner role:

- Grants full permissions on a specific resource in the Cloudera Management Console, including the ability to delete the resource. It does not grant any cluster-level permissions such as the ability to access or manage a cluster via Cloudera Manager.
- Is assigned automatically on a resource to the user who created the resource. For example, if a user creates an environment called “test”, the user is assigned the Owner role for that environment.
- Allows a user to grant a set of rights (including the Owner role) on the resource to other users and groups. This is possible only if the user also has the IamUser or IamViewer role allowing to list users within the organization.

- Can be assigned at the scope of the following resources: an environment, Data Lake, shared resource (cluster template, recipe, image catalog, credential, proxy), Cloudera Data Hub cluster, or classic cluster
- Can be assigned using the same steps as other resource roles.

Related Information

Assigning resource roles to users

Assigning resource roles to groups

Default User Roles (Cloudera Manager)

Group membership administration roles

The IamGroupAdmin role can be assigned to a user or a group on the scope of a group to allow them to manage membership of that group.

Note that:

- The IamGroupAdmin role grants a user or a group the permission to add users to or remove users from a group. The role does not grant permission to manage roles and resources for the group.
- In order for a user with the IamGroupAdmin to add or remove users from a group, the user must also have the IamUser or IamViewer role that allows listing IAM users and groups within the organization.

Related Information

Assigning a group membership administrator

Example role assignment scenario

This section outlines an example role assignment scenario.

The following illustration presents an example scenario where account and resources roles are assigned to multiple users within an organization:

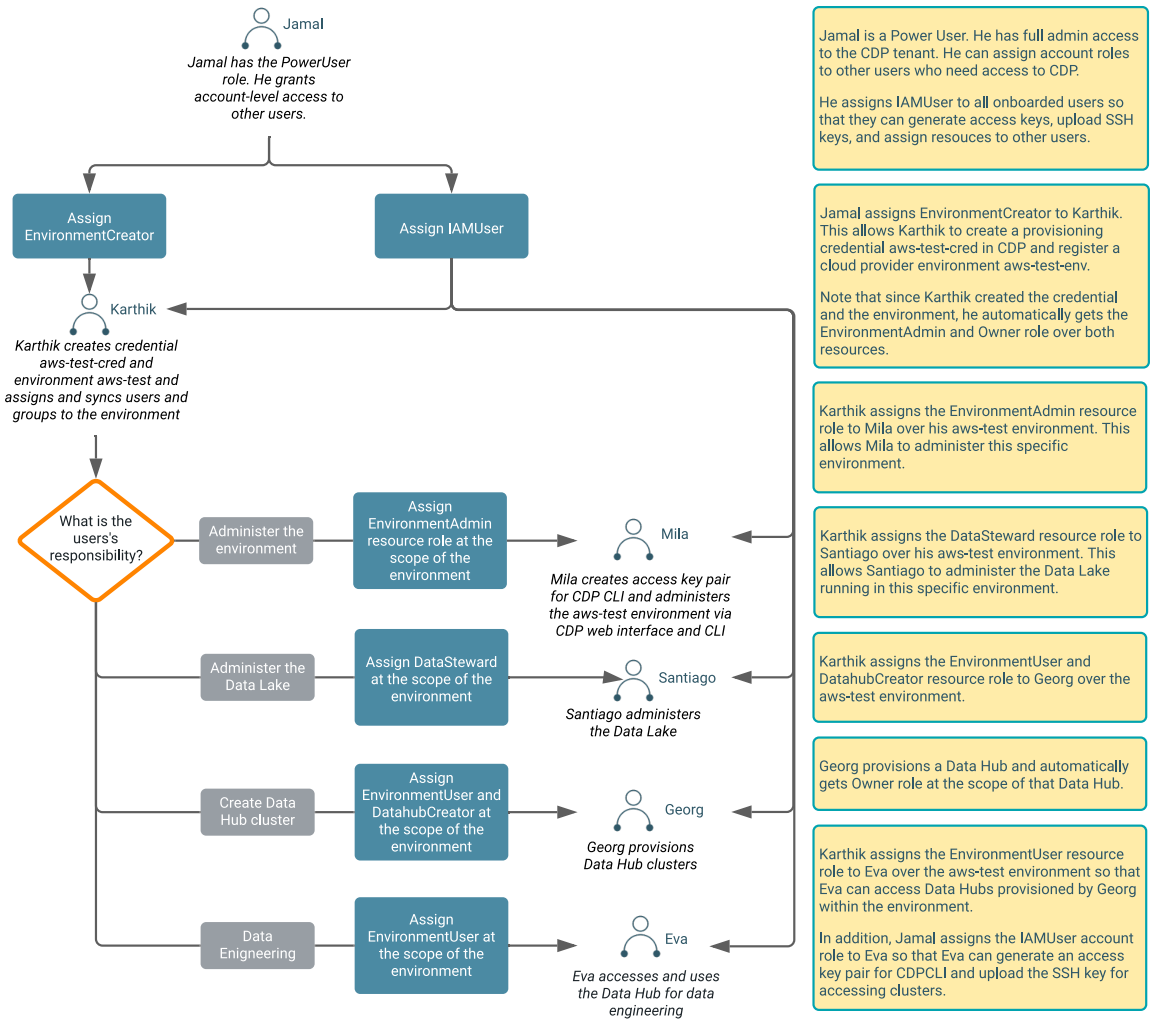

User and group limits

Cloudera has default limits in place with regard to how many users, machine users, and groups can be added per account. Customers can contact Cloudera Support to increase these limits.

The following table describes the default limit (second column) and the maximum limit that can be requested by contacting Cloudera Support (third column):

Limit name	Default limit	Maximum limit that can be requested
Maximum number of users	1,000 users	10,000 users Note: The user limit and machine user limit must total 10,000 or less. Approved limit increases may require upgrades of one or more services.
Maximum number of groups	50 groups	3,000 groups
Maximum number of machine users	100 machine users	500 machine users Note: The user limit and machine user limit must total 10,000 or less.
Maximum number of access keys	200 access keys	2,000 access keys

Maximum number of access tokens	20 access tokens	200 access tokens

Managing users and machine users in Cloudera

A PowerUser can manage Cloudera users on the Cloudera web interface or via CDP CLI.

Creating a machine user in Cloudera

You can create a machine user for programmatic access to Cloudera.

Before you begin

Machine user names cannot start with a double underscore ("__").

Required role: PowerUser

Steps

For CDP UI

1. Navigate to the Cloudera Management Console > User Management > Users.
2. From the Actions menu, select Create Machine User.
3. Provide a name and click on Create.

For CDP CLI

Use the following command:

```
cdp iam create-machine-user  \
--machine-user-name MACHINE_USER_NAME
```

Next, generate an API access key for your machine user by using the following command:

```
cdp iam create-machine-user-access-key \
    --machine-user-name MACHINE_USER_NAME
```

What to do next

- You need to perform user sync for the change to take effect. See Performing user sync.
- To generate an API access key for your machine user, see Generating an API access key.
- You should grant the newly created machine user access to resources. To do this, follow the steps for Assigning resources to users.

Note: When searching for the machine user in the "Select group or user" text box, make sure to enter machine user name, not workload user name.

Deleting users and machine users

Cloudera administrators have the ability to delete users and machine users in Cloudera through both the Cloudera user interface and the CDP CLI.

If a Cloudera user or machine user is deleted from an integrated identity provider system (for example, if a user leaves the company), the user is not automatically deleted in Cloudera. Cloudera administrators have the ability to delete a user in Cloudera through both the user interface and the CLI.

Required role: PowerUser

Steps

For CDP UI

1. From Cloudera user interface, navigate to the Cloudera Management Console > User Management.
2. Search for the user or machine user that you want to delete and click the ⋮ (context menu) at the end of the user entry row.
3. Perform one of the following:
   - Click Delete User (for a user) or Delete Machine User (for a machine user) and then OK on the confirmation screen.
   - Alternatively, you can click on the user or machine user name to enter the user's detail page. From there, click Actions > Delete User (for a user) or Delete Machine User (for a machine user).

For CDP CLI

In the CDP CLI, the command to delete a user is delete-user and the command to delete a machine user is delete-machine-user.

Run the delete-user command as shown in the following example:

```
cdp iam delete-user --user-id <value>
```

The delete-user command requires a user-id value, which can be either the user ID or the CRN (Cloudera resource name) of the given user. You can obtain the user ID from the web interface from the user's details, or from the CDP CLI from the output of the cdp iam list-users command.

Run the delete-machine-user command as shown in the following example:

```
cdp iam delete-machine-user --machine-user-name <value>
```

The delete-machine-user command requires a machine-user-name value, which can be either the user ID or the CRN (Cloudera resource name) of the given machine user. You can obtain the user ID from the web interface from the machine user's details, or from the CDP CLI from the output of the cdp iam list-machine-users command.

For a detailed description of the command properties, call the Cloudera help for the command:

```
cdp iam delete-user --help
cdp iam delete-machine-user --help
```

What to do next

Deleting a user or machine user removes all access keys and SSH keys associated with the user, and unassigns all roles and resource roles assigned to the user. The user is also removed from all groups that they belong to. Once the deletion process is completed and synced, the user will not be able to use any access keys to access Cloudera.

You must trigger user sync to ensure that the deleted user loses access to all environments. See Performing user sync. Only once the user sync is complete, the deleted user loses access to Cloudera.

It takes around 2 minutes to fully delete a user or machine user in Cloudera. During this time you will not be able to recreate the user (that is, for 2 minutes you will not be able to create a user in the same Identity Provider with the same NameID), but you can proceed to trigger user sync right away.

Assigning account roles to users

Assign account roles to a Cloudera user to manage the tasks that the user can perform in Cloudera. You can assign multiple roles to users or machine users to provide them with the permissions they need to perform their required tasks.

Required role: PowerUser

Steps

For CDP UI

1. Sign in to Cloudera.
2. From the Cloudera home page, click Cloudera Management Console.
3. Click User Management.

   The Users page displays the list of all Cloudera users.
4. Click the name of the user to whom you want to assign a role.

   The user details page displays information about the user.
5. Click the Roles tab.
6. Click Update Roles.
7. On the Update Roles window, select the roles you want to assign to the user.

   To remove a role from the user account, clear the selected role.
8. Click Update.

   The roles that you select displays in the list of roles assigned to the user.

   To remove a role from a user account, click check box next to the assigned role that you want to remove. Click Update to confirm that you want to revoke the role permissions.

For CDP CLI

You can use the following command to assign a role to a user or a machine user:

```
cdp iam assign-user-role \
--user-name <value> \
--role <value>
```

To remove a role from a user or a machine user:

```
cdp iam unassign-user-role \
--user-name <value> \
--role <value>
```

```
cdp iam unassign-machine-user-role \
--machine-user-name <value> \
--role <value>
```

The --role parameter requires the CRN of the Cloudera role. You can use the cdp iam list-roles command to list resource roles with role CRNs.

To get a list of the roles assigned to a group:

```
cdp iam list-user-assigned-roles \
--user-name <value>
```

```
cdp iam list-machine-user-assigned-roles \
--machine-user-name <value>
```

What to do next

You need to perform user sync for the change to take effect. See Performing user sync.

Related Information

Account roles

Assigning resource roles to users

To grant a user or a machine user access to a resource (such as an environment, a shared resource, or a Cloudera Data Hub cluster), assign a resource role to the user on the scope of that resource or, in some cases (Cloudera Data Hub clusters), on the scope of the parent resource.

In general, resource roles can be assigned from Cloudera user interface using the Manage Access option available from the resource details page or from CDP CLI using the cdp iam assign-user-resource-role or cdp iam assign-machine-user-resource-role commands. For detailed instructions, see the following sections:

Assign an environment resource role to a user

To assign an environment to a user or a machine user, assign a specific resource role on the scope of the specific environment.

Required roles:

- Owner or a role that allows administering the environment AND
- One of the following: IamViewer or IamUser (required for listing users).

In order to assign a role, a user must have all rights from the role that they are planning to assign to another user; That is, a user can only assign a role higher than his own.

For CDP UI

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. Navigate to the Environments page.
4. In the list of environments that appear, select an environment by clicking on it.
5. From the Actions menu select Manage Access.
6. In the Access tab, enter the name of the user in the text box.
7. In the Update Resource Roles window, select the required resource role.
8. Click Update Roles.

For CDP CLI

Use the following commands to assign a resource to a user or a machine user:

```
cdp iam assign-user-resource-role \
--user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

```
cdp iam assign-machine-user-resource-role \
--machine-user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

To remove a resource role from a user or a machine user:

```
cdp iam unassign-user-resource-role \
--user-name <value> \
```

```
--resource-role-crn <value> \
--resource-crn <value>
```

```
cdp iam unassign-machine-user-resource-role \
--machine-user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

- The resource-role-crn parameter requires the CRN of the resource role you want to assign to the user. You can use the cdp iam list-resource-roles command to list resource roles with role CRNs.
- The resource-crn parameter requires the CRN of the resource on which you want to grant the resource role permissions. You can obtain it from the details of the resource.

To get a list of the resource roles assigned to a user or a machine user:

```
cdp iam list-user-assigned-resource-role \
--user-name <value>
```

```
cdp iam list-machine-user-assigned-resource-role \
--machine-user-name <value>
```

What to do next

You need to perform user sync for the change to take effect. See Performing user sync.

Related Information

Resource roles

Assign a shared resource role to a user

You can assign shared resources such as credentials, clusters templates, recipes, image catalogs, or proxies to users and machine users. To assign a shared resource to a user or a machine user, assign a specific resource role on the scope of the specific shared resource.

Required roles:

- Owner or a role that allows administering the environment AND
- One of the following: IamViewer or IamUser (required for listing users).

In order to assign a role, a user must have all rights from the role that they are planning to assign to another user; That is, a user can only assign a role higher than his own.

For CDP UI

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. From the left pane, select Shared Resources, and then select a resource type (for example Cluster Templates) to view a summary of all resources of that type.
4. Find the specific resource (for example a specific cluster template) and click on it to navigate to its details page.
5. Click on Manage Access.
6. Enter the name of the user in the text box.
7. In the Update Resource Roles window, select the required resource role.
8. Click Update Roles.

For CDP CLI

Use the following commands to assign a resource to a user or a machine user:

```
cdp iam assign-user-resource-role \
--user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

```
cdp iam assign-machine-user-resource-role \
--machine-user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

To remove a resource role from a user or a machine user:

```
cdp iam unassign-user-resource-role \
--user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

```
cdp iam unassign-machine-user-resource-role \
--machine-user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

- The resource-role-crn parameter requires the CRN of the resource role you want to assign to the user. You can use the cdp iam list-resource-roles command to list resource roles with role CRNs.
- The resource-crn parameter requires the CRN of the resource on which you want to grant the resource role permissions. You can obtain it from the details of the resource.

To get a list of the resource roles assigned to a user or a machine user:

```
cdp iam list-user-assigned-resource-role \
--user-name <value>
```

```
cdp iam list-machine-user-assigned-resource-role \
--machine-user-name <value>
```

Related Information

Resource roles

Assign a Cloudera Data Hub resource role to a user

You can assign a Cloudera Data Hub resource role to a user or a machine user to allow them to manage a specific Cloudera Data Hub.

Note: While full access to manage a Cloudera Data Hub via the Cloudera Management Console can be granted via assigning the Owner role on the scope of the Cloudera Data Hub, access to the underlying cluster can only be granted by assigning the EnvironmentUser role on the scope of the environment where the Cloudera Data Hub is running.

Required roles:

- Owner or a role that allows administering the environment AND
- One of the following: IamViewer or IamUser (required for listing users).

In order to assign a role, a user must have all rights from the role that they are planning to assign to another user; That is, a user can only assign a role higher than his own.

For CDP UI

1. Sign in to the Cloudera console.
2. Navigate to the details page of your Cloudera Data Hub cluster. This can be done in a few ways. For example:
   a. From the Cloudera home page, click Data Hub Clusters and then click on the specific cluster.
   b. From the Cloudera home page, click on Management Console, navigate to the Data Hub Clusters page, and then click on the specific cluster.
3. From the Actions menu select Manage Access.
4. Enter the name of the user in the text box.
5. In the Update Resource Roles window, select the required resource role.
6. Click Update Roles.

For CDP CLI

Use the following commands to assign a resource to a user or a machine user:

```
cdp iam assign-user-resource-role \
--user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

```
cdp iam assign-machine-user-resource-role \
--machine-user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

To remove a resource role from a user or a machine user:

```
cdp iam unassign-user-resource-role \
--user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

```
cdp iam unassign-machine-user-resource-role \
--machine-user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

- The resource-role-crn parameter requires the CRN of the resource role you want to assign to the user. You can use the cdp iam list-resource-roles command to list resource roles with role CRNs.
- The resource-crn parameter requires the CRN of the resource on which you want to grant the resource role permissions. You can obtain it from the details of the resource.

To get a list of the resource roles assigned to a user or a machine user:

```
cdp iam list-user-assigned-resource-role \
--user-name <value>
```

```
cdp iam list-machine-user-assigned-resource-role \
--machine-user-name <value>
```

Related Information

Resource roles

Assign a classic cluster resource role to a user

You can assign a specific resource role to a user or a machine user on the scope of a specific classic cluster to allow them to manage a specific classic cluster.

Required roles:

- Owner or a role that allows administering the environment AND
- One of the following: IamViewer or IamUser (required for listing users).

In order to assign a role, a user must have all rights from the role that they are planning to assign to another user; That is, a user can only assign a role higher than his own.

For CDP UI

1. In the Cloudera Management Console navigate to the Classic Clusters dashboard.
2. Click on the ⋮ (context menu) next to the cluster that you want to update and select Manage Access.
3. Find the user that you want to update and click on Update Roles.
4. Select or deselect the roles and then click on Update Roles.

For CDP CLI

Use the following commands to assign a resource to a user or a machine user:

```
cdp iam assign-user-resource-role \
--user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

```
cdp iam assign-machine-user-resource-role \
--machine-user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

To remove a resource role from a user or a machine user:

```
cdp iam unassign-user-resource-role \
--user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

```
cdp iam unassign-machine-user-resource-role \
--machine-user-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

- The resource-role-crn parameter requires the CRN of the resource role you want to assign to the user. You can use the cdp iam list-resource-roles command to list resource roles with role CRNs.
- The resource-crn parameter requires the CRN of the resource on which you want to grant the resource role permissions. You can obtain it from the details of the resource.

To get a list of the resource roles assigned to a user or a machine user:

```
cdp iam list-user-assigned-resource-role \
--user-name <value>
```

```
cdp iam list-machine-user-assigned-resource-role \
--machine-user-name <value>
```

Managing groups in Cloudera

A PowerUser can create and manage Cloudera groups on the Cloudera web interface or via CDP CLI.

Reserved group names

There are certain group names that are reserved and therefore cannot be used in CDP. This applies to groups synchronized from your identity provider as well as groups created directly from Cloudera.

If you attempt to synchronize or register a group with a reserved name, you will get an error including the following message:

```
Invalid group name
Name cannot be a reserved group name
```

To avoid problems, review the following list and avoid synchronising or creating groups with the following names.

The following group names are reserved:

- accumulo
- admins
- atlas
- cruisecontrol
- dpprofiler
- druid
- editors
- flink
- flume
- h2o
- hbase
- hdfs
- hive
- httpfs
- hue
- impala
- ipausers
- kafka
- keytrustee
- kms
- knox
- kudu
- livy
- mapred
- nifi
- nifiregistry
- oozie
- phoenix
- ranger
- rangerraz
- schemaregistry

- sentry
- solr
- spark
- sqoop
- sqoop2
- streamsmsgmgr
- streamsrepmgr
- tez
- trust admins
- yarn
- yarn-ats
- zeppelin
- zookeeper

Understanding Cloudera groups

A Cloudera group is a collection of user accounts that have the same roles and resource roles. A group can include Cloudera user accounts and machine user accounts. A group cannot include other groups. All users in a group inherit the roles and resource roles assigned to the group.

As a Cloudera administrator, you can create a group and manage the group membership. You can also manage the roles and resources assigned to the group. If you are not a Cloudera administrator, you can add users to and remove users from a group if you have the PowerUser role.

When you create a group, you do not automatically become a member of the group. To become a member of the group, you must add your user account to the group.

You can use groups to manage user access more efficiently. If multiple users require the same roles, you can create a group, add the user accounts to the group, and assign the required roles to the group. All user accounts in the group are assigned the roles assigned to the group.

If you delete a group, users in the group lose the roles that they inherit from the group. To allow a user to retain the group roles, assign the same roles to the user separately.

Synchronizing group membership

Cloudera can synchronize the user's group membership provided by your enterprise IdP with the user's group membership in Cloudera.

When a user initially logs in to Cloudera through the identity management system in your organization, Cloudera creates a Cloudera user account for the user. However, without being assigned Cloudera roles, the user cannot perform tasks in Cloudera. Cloudera recommends that you create Cloudera groups with assigned roles and add users to the groups so that the users can take on the roles assigned to the groups.

When you create an identity provider, you can select the Sync Groups on Login option to enable Cloudera to synchronize the user group membership. By default, the Sync Groups on Login option is disabled. Clear the option selection if you do not want Cloudera to synchronize the user group membership.

Group names must be alphanumeric, may include dots (.), hyphens (-), and underscores (_), and must be fewer than 64 characters long. Additionally, names can only start with an alphabetic character or an underscore.

Note:

There are certain group names that are reserved and therefore cannot be synchronized to Cloudera. See Reserved group names.

Note: Cloudera has default limits in place with regard to how many users, machine users, and groups can be added per account. Review User and group limits and make sure that you do not exceed these limits.

Sync Groups on Login enabled

When the Sync Groups on Login option is enabled, Cloudera synchronizes a user's group in the following manner:

- The group membership that your enterprise IdP specifies for a user overrides the group membership set up in Cloudera. Each time a user logs in, Cloudera updates the user's group membership based on the groups that your enterprise IdP specifies for the user.
- If the group exists in Cloudera, Cloudera adds the user to the group. The user takes on all the roles associated with the group.
- If the group does not exist in Cloudera, Cloudera creates the group and adds the user to the group. However, no roles are assigned to the new group, so a member of the new group does not take on roles from the group.
- If the user is a member of a group in Cloudera that is not included in the list provided by your enterprise IdP, Cloudera removes the user from the group.
- If the list of groups from your enterprise IdP is empty, Cloudera removes the user from all groups in Cloudera. After login, the user will not be a member of any Cloudera group and will not have roles from any group.

To ensure that users can perform tasks in Cloudera, Cloudera recommends that you set up the groups in Cloudera with appropriate roles before you assign them to users.

Sync Groups on Login disabled

When the Sync Groups on Login option is disabled, Cloudera does not synchronize the user's group membership in CDP with the user's group membership provided by the IdP. After login, a user's group membership in Cloudera is determined by the Cloudera groups assigned to the user in Cloudera. The groups assigned to the user in your enterprise IdP are ignored.

Sync Membership option for a newly created group

Additionally, once you have synced your IdP and you create a new group in Cloudera, you have an option called Sync Membership that determines whether group membership is synced to IdP when a user logs in. By default, Sync Membership is enabled when Sync Groups on Login is enabled.

The following table describes how the global Sync Groups on Login and the per-group Sync Membership options can be used:

	IdP Sync Groups on Login on	IdP Sync Groups on Login off
Group Sync Membership on	Group membership for the specific group is reflected in IdP.	Group membership for the specific group is not reflected in IdP.
Group Sync Membership off	Group membership for the specific group is not reflected in IdP.	Group membership for the specific group is not reflected in IdP.

In other words, if Sync Groups on Login is off at the IdP level, then no groups are getting synced regardless of what the setting for Sync Membership is. But if Sync Groups on Login is turned on at the IDP level, then you have the option to override it for certain groups that you explicitly leave off.

Creating a group

Create Cloudera groups based on the tasks performed by Cloudera users in your organization.

Before you begin

Consider the following when selecting a name for your group:

- The group name must be unique. Note that there are certain group names that are reserved and therefore cannot be used in Cloudera. See Reserved group names.

- The group name can be up to 64 characters and can include only alphanumeric characters, dots (.), hyphens (-), and underscores (_). The first character in the name must be an alphabetic character or underscore.
- The group name is not case sensitive. For example, the group name AAa is equivalent to the group name aaa.
- Depending on your IdP setup in Cloudera, you may be able to manipulate the Sync Membership option. To learn more about this option, refer to Synchronizing group membership.

Required role: PowerUser

Steps

For CDP UI

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. In the User Management section of the side navigation panel, click Groups.

   The Groups page displays the list of all Cloudera groups.
4. Click Create Group.
5. On the Create Group window, enter the name of the group to create.
6. Click Create.

   Cloudera creates the group and adds it to the list of Cloudera groups on the Groups page.

For CDP CLI

You can use the following command to create a group:

```
cdp iam create-group \
--group-name <value>
```

What to do next

You need to perform user sync for the change to take effect. See Performing user sync.

Adding or removing a user from a group

You can add or remove a Cloudera user or a machine user account from a group.

Note that:

- SAML login is required to propagate group membership changes from your IdP to Cloudera. That is, the user who was added or removed from a group must log in to Cloudera in order for the group membership change to take effect in Cloudera.
- You cannot add a group to another group.
- All members of the group inherit the roles and resources assigned to the group.
- Cloudera has default limits in place with regard to how many users, machine users, and groups can be added per account. Review User and group limits and make sure that you do not exceed these limits.

Required roles: IamGroupAdmin is the minimum role required for adding or removing users from a group. In addition, in order for a user with the IamGroupAdmin to add or remove users from a group via Cloudera web interface, the user must have either the IamUser or the IamViewer role that allows listing IAM users and groups. This is not required when adding or removing users from a group via CDP CLI, as long as the admin has the CRN of the user that needs to be added or removed.

Steps

For CDP UI

To add a user to a group:

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. In the User Management section of the side navigation panel, click Groups.

   The Groups page displays the list of all Cloudera groups.
4. Click the name of the group to which you want to add a user.

   The group details page displays information about the group.
5. Click the Members tab.
6. To add a user:

   - If the group does not yet have members, click Add Member. Select the name of the user that you want to add to the group.
   - If the group already has a list of members, click in the Add a member dropdown box. Select the name of the user that you want to add to the group.

   To remove a user from a group, click Remove from Group next to the user that you want to remove. Click OK to confirm that you want to remove the user from the group.

For CDP CLI

The user-id parameter requires the CRN of the Cloudera user or machine user.

You can use the following command to add a user to a group:

```
cdp iam add-user-to-group \
--group-name <value> \
--user-id <value>
```

To remove a user from a group:

```
cdp iam remove-user-from-group \
--group-name <value> \
--user-id <value>
```

You can use the following command to add a machine user to a group:

```
cdp iam add-machine-user-to-group \
--group-name <value> \
--user-id <value>
```

To remove a machine user from a group:

```
cdp iam remove-machine-user-from-group \
--group-name <value> \
--machine-user-name <value>
```

To get a list of the users in a group:

```
cdp iam list-group-members \
--group-name <value>
```

To get the list of groups that a user or machine user is a member of:

```
cdp iam list-groups-for-user \
--user-id <value>
```

```
cdp iam list-groups-for-machine-user \
--machine-user-name <value>
```

What to do next

You need to perform user sync for the change to take effect. See Performing user sync.

Assigning account roles to groups

When you assign a role to a group, the role is also assigned to all user and machine user accounts in the group.

Required role: PowerUser

Steps

For CDP UI

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. In the User Management section of the side navigation panel, click Groups.

   The Groups page displays the list of all Cloudera groups.
4. Click the name of the group to which you want to assign a role.

   The group details page displays information about the group.
5. Click the Roles tab.
6. Click Update Roles.
7. On the Update Roles window, select the roles you want to assign to the group.
8. To view the permissions that the role grants to the group, click Policies. To remove a role from the group, clear the selected role.
9. Click Update.

   The roles that you select displays in the list of group roles.

   To remove a role from a group, click Unassign Role next to the role that you want to remove. Click OK to confirm that you want to remove the role permissions from the group.

For CDP CLI

You can use the following command to assign a role to a group:

```
cdp iam assign-group-role \
--group-name <value> \
--role <value>
```

The --role parameter requires the CRN of the Cloudera role. You can use the cdp iam list-roles command to list resource roles with role CRNs.

To get a list of the roles assigned to a group:

```
cdp iam list-group-assigned-roles \
--group-name <value>
```

What to do next

You need to perform user sync for the change to take effect. See Performing user sync.

Related Information

Account roles

Assigning resource roles to groups

When you assign a resource role to a group, the resource role is also assigned to all user and machine user accounts in the group.

To grant a group access to a resource (such as an environment, a shared resource, or a Cloudera Data Hub cluster), assign a resource role to the group on the scope of the resource and the resource role is also assigned to all user and machine user accounts in the group.

In general, resource roles can be assigned from Cloudera web interface using the Manage Access option available from the resource details page, or from CDP CLI using the cdp iam assign-group-resource-role command. Next, you need to perform user sync. For detailed instructions, see the following sections:

Assign an environment resource role to a group

To assign an environment to a group, assign a specific resource role on the scope of the specific environment.

Required roles:

- Owner or a role that allows administering the environment AND
- One of the following: IamViewer or IamUser (required for listing users).

In order to assign a role, a user must have all rights from the role that they are planning to assign to another user; That is, a user can only assign a role higher than his own.

For CDP UI

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. Navigate to the Environments page.
4. In the list of environments that appear, select an environment by clicking on it.
5. From the Actions menu select Manage Access.
6. In the Access tab, enter the name of the group in the text box.
7. In the Update Resource Roles window, select the required resource role.
8. Click Update Roles.

For CDP CLI

To assign a resource role to a group:

```
cdp iam assign-group-resource-role \
--group-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

To remove a resource role from a group:

```
cdp iam unassign-group-resource-role \
--group-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

- The resource-role-crn parameter requires the CRN of the resource role you want to assign to the group.
- The resource-crn parameter requires the CRN of the resource on which you want to grant the resource role permissions.

To get a list of the resource roles assigned to a group:

```
cdp iam list-group-assigned-resource-role \
--group-name <value>
```

What to do next

You need to perform user sync for the change to take effect. See Performing user sync.

Related Information

Resource roles

Assign a shared resource resource role to a group

You can assign shared resources such as credentials, clusters templates, recipes, image catalogs, or proxies to groups. To assign a shared resource to a group, assign a specific resource role on the scope of the specific shared resource.

Required roles:

- Owner or a role that allows administering the environment AND
- One of the following: IamViewer or IamUser (required for listing users).

In order to assign a role, a user must have all rights from the role that they are planning to assign to another user; That is, a user can only assign a role higher than his own.

For CDP UI

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. From the left pane, select Shared Resources, and then select a resource type (for example Cluster Templates) to view a summary of all resources of that type.
4. Find the specific resource (for example a specific cluster template) and click on it to navigate to its details page.
5. Click on Manage Access.
6. Enter the name of the group in the text box.
7. In the Update Resource Roles window, select the required resource role.
8. Click Update Roles.

For CDP CLI

To assign a resource role to a group:

```
cdp iam assign-group-resource-role \
--group-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

To remove a resource role from a group:

```
cdp iam unassign-group-resource-role \
--group-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

- The resource-role-crn parameter requires the CRN of the resource role you want to assign to the group.
- The resource-crn parameter requires the CRN of the resource on which you want to grant the resource role permissions.

To get a list of the resource roles assigned to a group:

```
cdp iam list-group-assigned-resource-role \
--group-name <value>
```

Related Information

Resource roles

Assign a Cloudera resource role to a group

You can assign a Cloudera Data Hub resource role to a group to allow them to manage a specific Cloudera Data Hub.

Note: While full access to manage a Data Hub via the Management Console can be granted via assigning the Owner role on the scope of the Cloudera Data Hub, access to the underlying cluster can only be granted by assigning the EnvironmentUser role on the scope of the environment where the Cloudera Data Hub cluster is running.

Required roles:

- Owner or a role that allows administering the environment AND
- One of the following: IamViewer or IamUser (required for listing users).

In order to assign a role, a user must have all rights from the role that they are planning to assign to another user; That is, a user can only assign a role higher than his own.

For CDP UI

1. Sign in to the Cloudera console.
2. Navigate to the details page of your Cloudera Data Hub cluster. This can be done in a few ways. For example:
   - From the Cloudera home page, click Data Hub Clusters and then click on the specific cluster.
   - From the Cloudera home page, click on Cloudera Management Console, navigate to the Data Hub Clusters page, and then click on the specific cluster.
3. From the Actions menu select Manage Access.
4. Enter the name of the group in the text box.
5. In the Update Resource Roles window, select the required resource role.
6. Click Update Roles.

For CDP CLI

To assign a resource role to a group:

```
cdp iam assign-group-resource-role \
```

```
--group-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

To remove a resource role from a group:

```
cdp iam unassign-group-resource-role \
--group-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

- The resource-role-crn parameter requires the CRN of the resource role you want to assign to the group.
- The resource-crn parameter requires the CRN of the resource on which you want to grant the resource role permissions.

To get a list of the resource roles assigned to a group:

```
cdp iam list-group-assigned-resource-role \
--group-name <value>
```

Related Information

Resource roles

Assign a classic cluster resource role to a group

You can assign a classic cluster resource role to a group to allow them to manage a specific classic cluster.

Required roles:

- Owner or a role that allows administering the environment AND
- One of the following: IamViewer or IamUser (required for listing users).

In order to assign a role, a user must have all rights from the role that they are planning to assign to another user; That is, a user can only assign a role higher than his own.

For CDP UI

1. In the Cloudera Management Console navigate to the Classic Clusters dashboard.
2. Click on the ⋮ (context menu) next to the cluster that you want to update and select Manage Access.
3. Find the group that you want to update and click on Update Roles.
4. Select or deselect the roles and then click on Update Roles.

For CDP CLI

To assign a resource role to a group:

```
cdp iam assign-group-resource-role \
--group-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

To remove a resource role from a group:

```
cdp iam unassign-group-resource-role \
--group-name <value> \
--resource-role-crn <value> \
--resource-crn <value>
```

- The resource-role-crn parameter requires the CRN of the resource role you want to assign to the group.

- The resource-crn parameter requires the CRN of the resource on which you want to grant the resource role permissions.

To get a list of the resource roles assigned to a group:

```
cdp iam list-group-assigned-resource-role \
--group-name <value>
```

Assigning a group membership administrator

As a Cloudera administrator, you can create a Cloudera group and manage the users, roles, and resources assigned to the group. You can also assign other users and groups the IamGroupAdmin role to allow them to manage the users in the group.

Note that:

- The IamGroupAdmin role grants a user or a group the permission to add users to or remove users from a group. The role does not grant permission to manage roles and resources for the group.
- In order for a user with the IamGroupAdmin to add or remove users from a group via Cloudera web interface, the user must have either the IamUser or the IamViewer role that allows listing IAM users and groups. This is not required when adding or removing users from a group via CDP CLI, as long as the admin has the CRN of the user that needs to be added or removed.

Required role: PowerUser

Steps

For CDP UI

To assign a group membership administrator:

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. In the User Management section of the side navigation panel, click Groups.

   The Groups page displays the list of all Cloudera groups.
4. Click the name of the group to which you want to assign a group membership administrator.

   The group details page displays information about the group.
5. Click the Admins tab.
6. Click in the Select group or user dropdown box.

   Cloudera displays the list of Cloudera groups and users that you can give group membership administrator permissions.
7. Select the name of a group or user.

   The name of the group or user you select displays in the list of group membership administrators.

   To remove group membership administrator permissions from a user or group, click Remove Resource Role next to the user or group for whom you want to revoke membership administrator permissions.

For CDP CLI

You assign the IamGroupAdmin resource role to users and groups to allow them to manage the users in a specified group.

You can use the following command to assign the IamGroupAdmin role to a user:

```
cdp iam assign-user-resource-role \
--user <value> \
--resource-role-crn <value> \
```

```
--resource-crn <value>
```

The user parameter requires the CRN of the user to whom you want to assign the IamGroupAdmin resource role.

The resource-role-crn parameter requires the CRN of the IamGroupAdmin role.

The resource-crn parameter requires the CRN of the group on which the user will have administrator permission.

To assign the IamGroupAdmin role to a group:

```
cdp iam assign-group-resource-role \
--group-name <value>e \
--resource-role-crn <value> \
--resource-crn <value>
```

The group-name parameter requires the name of the group to which you want to assign the IamGroupAdmin resource role.

The resource-role-crn parameter requires the CRN of the IamGroupAdmin role.

The resource-crn parameter requires the CRN of the group on which the group specified in the group-name parameter will have administrator permission.

For example, to assign the IamGroupAdmin to GroupABC so that GroupABC can manage the users in GroupXYZ, run a command similar to the following command:

```
cdp iam assign-group-resource-role \
--group-name groupABC \
--resource-role-crn crn:cdp:iam:us-west-1:cdp:resourceRole:IamGroupAdmin \
--resource-crn crn:cdp:iam:us-west-1:4e9d74e5-1cad-47d8-b645-7ccf9edbb73
d:group:GroupXYZ/54218ac1-187b-40f7-aadb-5ghm96c35xy4
```

To assign the users in a group to be the administrators of their own group, set the values of the group-name parameter and the resource-crn parameter to refer to the same group.

What to do next

You need to perform user sync for the change to take effect. See Performing user sync.

Updating a group

In some cases, you can enable or disable SyncMembership for a group.

Depending on your IdP setup in Cloudera, you may be able to manipulate the Sync Membership option. To learn more about this option, refer to Synchronizing group membership.

Required role: PowerUser

Steps

For CDP UI

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. In the User Management section of the side navigation panel, click Groups.
4. From the context menu to the right of the desired group, click Update Group.
5. Select or deselect the Sync Membership checkbox.
6. Click Update.

For CDP CLI

To update a group:

```
cdp iam update-group \
--group-name<value> \
--sync-membership-on-user-login
```

or

```
cdp iam update-group \
--group-name <value> \
--no-sync-membership-on-user-login
```

Removing account roles from a group

When you unassign a role to a group, the role is also unassigned to all user and machine user accounts in the group.

Required role: PowerUser

Steps

For CDP UI

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. In the User Management section of the side navigation panel, click Groups.

   The Groups page displays the list of all Cloudera groups.
4. Click the name of the group to which you want to assign a role.

   The group details page displays information about the group.
5. Click the Roles tab.
6. From the context menu to the right of a role, click Unassign role.
7. Click OK to confirm that you want to remove the role permissions from the group.

For CDP CLI

To remove a role from a group:

```
cdp iam unassign-group-role \
--group-name <value> \
--role <value>
```

The role parameter requires the CRN of the Cloudera role.

To get a list of the roles assigned to a group:

```
cdp iam list-group-assigned-roles \
--group-name <value>
```

What to do next

You need to perform user sync for the change to take effect. See Performing user sync.

Deleting a group

You can delete a group from Cloudera.

Required role: PowerUser

Steps

For CDP UI

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. In the User Management section of the side navigation panel, click Groups.
4. From the context menu to the right of the desired group, click Delete Group.
5. Click OK to confirm removal.
6. Cloudera removes the group and removes it from the list of Cloudera groups on the Groups page.

For CDP CLI

To delete a group:

```
cdp iam delete-group \
--group-name <value>
```

What to do next

You need to perform user sync for the change to take effect. See Performing user sync.

Performing user sync

When making any kind of user or group-related changes, you need to perform user sync in order for the changes to be synced to FreeIPA.

All user and group related changes with one exception of generating API access keys require user sync. For example all of the following require user sync:

- Creating machine users
- Deleting users and machine users
- Assigning Cloudera roles or environment-level resource roles to users
- Creating a group, assigning group membership, and deleting a group
- Setting workload password
- Managing user SSH keys

During user sync:

- All control plane actors (users and machine users) with the environments/accessEnvironment right are synced to the FreeIPA.
- All groups are synced to the FreeIPA.
- All users with PowerUser role are synchronized to all environments.

Syncing users to all environments

Required role: PowerUser

Steps

For CDP UI

1. From the CDP web interface, you can perform user sync from the Cloudera Management Console > User Management.
2. Click Actions > Synchronize Users.

3. Click Synchronize Users.
4. Status shown will be Running, then Completed.

For CDP CLI

From the CDP CLI, you can use the following commands:

```
cdp environments sync-all-users
```

This command synchronizes all users and groups with all Cloudera environments.

```
cdp environments sync-user
```

This commands only syncs the current user with all their Cloudera environments. You can use it if you are making changes to your own user, but you can't use it for syncing other users.

What to do next

Depending on how many users you have in Cloudera, it may take a few minutes for the user sync to complete. The sync operation times out after 30 minutes.

Syncing users to a selected environment

Required role: EnvironmentAdmin and DataSteward can perform user sync for a single environment. PowerUser can perform user sync for all environments.

Steps

For CDP UI

1. From the Cloudera web interface, you can perform user sync from the Cloudera Management Console > Environments.
2. Navigate to a specific environment.
3. Do one of the following:
   - Click on Actions > Synchronize Users to FreeIPA
   - Navigate to Summary > FreeIPA > Actions > Synchronize Users to FreeIPA
   - Click on Actions > Manage Access and then click the Synchronize Users button in the top right corner.
4. Click Synchronize Users.
5. Status shown will be Running, then Completed.

For CDP CLI

From the CDP CLI, you can use the following command:

```
cdp environments sync-all-users --environment-names <value>
```

This command synchronizes all users and groups with one or more Cloudera environments specified in --environment-names <value>.

What to do next

Depending on how many users you have in Cloudera, it may take a few minutes for the user sync to complete. The sync operation times out after 30 minutes.

Access paths to Cloudera and its components

To access the various Cloudera components, you must understand the access paths unique to the entry points that are specific to users and situations.

The typical access entry methods and their details are as follows:

- SSO access through Cloudera Management Console - After the initial identity provider configuration, users can access Cloudera and its SSO-interfaces through the Cloudera Management Console. You can access various Cloudera services including Cloudera Management Console, Cloudera AI Workbenches, Cloudera Data Catalog, Cloudera Replication Manager, Cloudera Data Hub, and Data Lakes.
- Accessing non-SSO interfaces using workload password - You must set your workload password to add the user credentials to the IPA server. Using the workload username and workload password, you can access non-SSO interfaces such as SSH to clusters, JDBC connections, REST APIs, Cloudera Data Warehouses, and so on.

  Note: To access Cloudera AI workbenches, you can enter the kerberos principal and the workload password, and gain kerberized access to components such as the HMS.

- Accessing CDP CLI using access keys - On the User Management section of the Cloudera Management Console, you can generate your access keys. Use these access keys for CDP CLI and for running jobs.
- Machine user access - To get programmatic access to Cloudera and its services, you can create and use a machine user. The process to set up the machine user for access is as follows:
  - Create a machine user in the User Management section of Cloudera Management Console.
  - Get access keys in the Cloudera Management Console for this machine user.
  - Use APIs and CDP CLI to set the workload password for the machine user.
- SSH access - There are two types of SSH access:
  - Admin users who create Environments can access the environments directly using the SSH key access.
  - All users can access workload clusters by using their SSH keys previously uploaded in Cloudera.

User Access Paths to CDP

Acess Point	Initial Configuration	Accessible Components	Limitations
SSO through Management Console	Using SSO configured as part of the initial IDP setup	Management Console, Data Lakes, Data Hub, Data Catalog, Replication Manager, Machine Learning	Exceptions: Data Analytics Studio (DAS) and Grafana
Access through workload password	Find your workload user ID and set workload password	SSH to cluster, DWX, JDBC, REST APIs	None
Access using access keys	Generate access keys in the User Management panels	CLI and for running jobs	None
Machine user access	Create machine user in the User Management panels and set IPA password through CLI	To use APIs to create applications such as ETL	Need to create access keys. Machine user password can be set from the UI by Power Users
SSH key access	SSH key for power user access is uploaded during environment creation. All users can upload their SSH keys	To access the environment and clusters directly	None

Related Information

Setting the workload password

Accessing non-SSO interfaces using workload user and password

Managing SSH keys

Generating an API access key

Setting a default identity provider in Cloudera

You can set a default identity provider (IdP) in Cloudera for workload-initiated SSO using Cloudera user interface or CDP CLI.

By default, the oldest configured identity provider is used workload-initiated SSO, but you can optionally set a default IdP using Cloudera user interface or CDP CLI.

Required roles: Account administrator or PowerUser

Steps

For CDP UI

1. In the Cloudera user interface, navigate to the Cloudera Management Console.
2. Select User Management from the navigation pane and then navigate to Identity Providers.
3. Click on the context menu next to the entry for a previously registered identity provider and select Set As Default Identity Provider from the menu.

Once the default identity provider has been updated, you will see the “Default” label next to the idP name.

For CDP CLI

To set a default IdP, use the following command:

```
cdp iam set-default-identity-provider --name-or-crn  <IDP-NAME>
```

To print the CRN of the default IdP, use the following command:

```
cdp iam get-default-identity-provider
```

This returns a CRN of the default identity provider.

Accessing non-SSO interfaces using workload user and password

You can access all SSO-based interfaces that can be accessed using the browser with your Cloudera credentials, but you cannot access SSH-based or other non-SSO connections with these credentials. Instead, you must use your workload user and set the workload password to access such interfaces.

You must import and trust the IPA's root certificate as the IPA CA will be a self-signed CA.

Note:

If you would like to access cluster nodes, you have two options: you can use your workload password or an SSH key. The advantage of the workload password is that it allows a user to interface with Kerberos. Therefore, if you would like to interface with Kerberos, use your workload password to SSH to cluster nodes.

Related Information

Setting the workload password

Setting the workload password

To access non-SSO interfaces, each user and machine user must set a workload password (also known as "FreeIPA password"). An administrator can set other users' workload passwords.

Required roles: All users can manage their workload passwords from the account management page. All users can manage their workload password from CDP CLI, but this action requires an API access key, which can only be generated by users with the IAMUser role. As a Cloudera administrator or PowerUser, you can manage the workload password for all user accounts.

Workload password requirements

Your Cloudera administrator may set a custom workload password policy for your organization. If your Cloudera administrator did not set a custom workload password policy default, Cloudera has the following workload password requirements:

- A minimum password length of 8 characters
- Must include at least 1 upper case character, lowercase character, number and special character. Supported special characters are: "#", "&", "*", "$", "%", "@", "^", ".", "_", and "!".
- All previous passwords can be reused
- The password can be changed at any time
- The password never expires

Set your own workload password

As a Cloudera user, you can see on your profile page if you have previously set your workload password and if the password is about to expire. There are two cases when you may want to set your workload password:

1. When you first start using Cloudera.
2. When your password expires. This may or may not happen depending on your company's policies. If your password does expire, you will see a banner notification on the Cloudera web interface 10 days before the expiry date. You can also see on your user's profile page the state of your workload password (if it expires soon or cannot yet be changed).

Steps

For CDP UI

1. Sign in to the Cloudera web interface.
2. Click on your user name in the bottom left corner and then select Profile.
3. Click Set Workload Password:

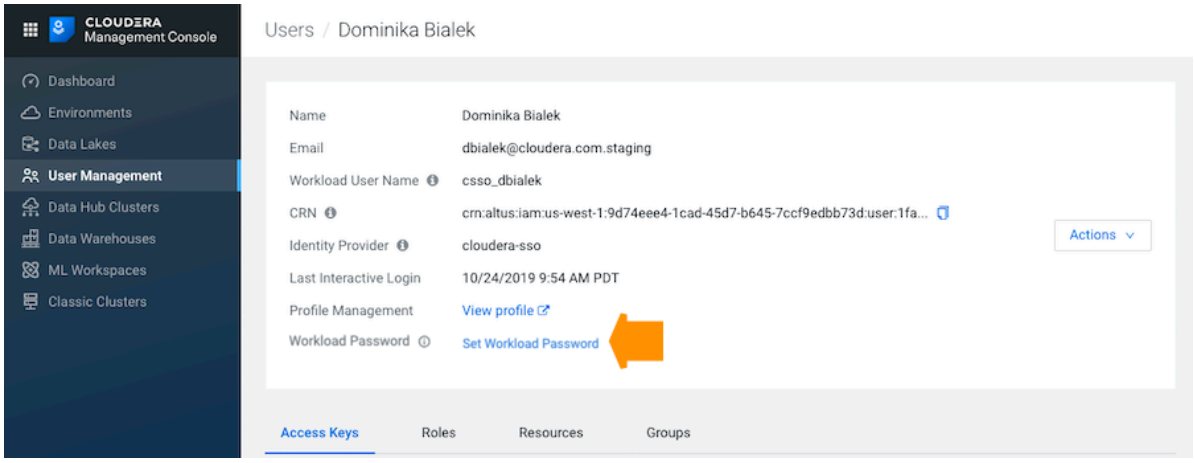

4. In the dialog box that appears, enter the new workload password twice.
5. In the Environments text box, All is pre-selected so that the workload password is synced to all environments by default.
6. Click Set Workload Password. A message appears saying that the password is set successfully.
7. Click Close.

For CDP CLI

Use the following command to set workload password:

```
cdp iam set-workload-password --password <value>
```

Set workload password for another user or machine user (admin only)

There are two cases when you may want to set workload password for a machine user:

1. When you are first onboarding the machine user to Cloudera.

2. When the machine user's password expires. This may or may not happen depending on your company's policies. A Cloudera administrator or PowerUser is able to navigate to the list of all users to see for which machine user passwords are about to expire. In the "Password expiring" column, any password that is about to expire in 10 days or less is flagged as "Expires in X days". Only machine users (and not human users) are flagged in this manner. A Cloudera administrator or PowerUser can then reset the password for each machine user whose password is about to expire.

Steps

For CDP UI

1. Sign in to the Cloudera web interface.
2. From the Cloudera home page, click Cloudera Management Console.
3. On the side navigation panel, click User Management.
4. On the Users page, enter your name in the search bar and then click on your user name:

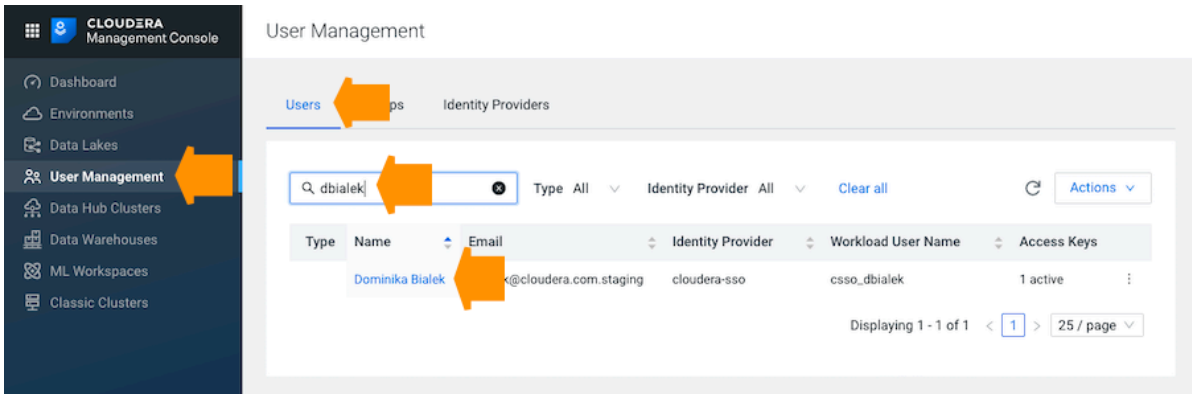

5. Click Set Workload Password:

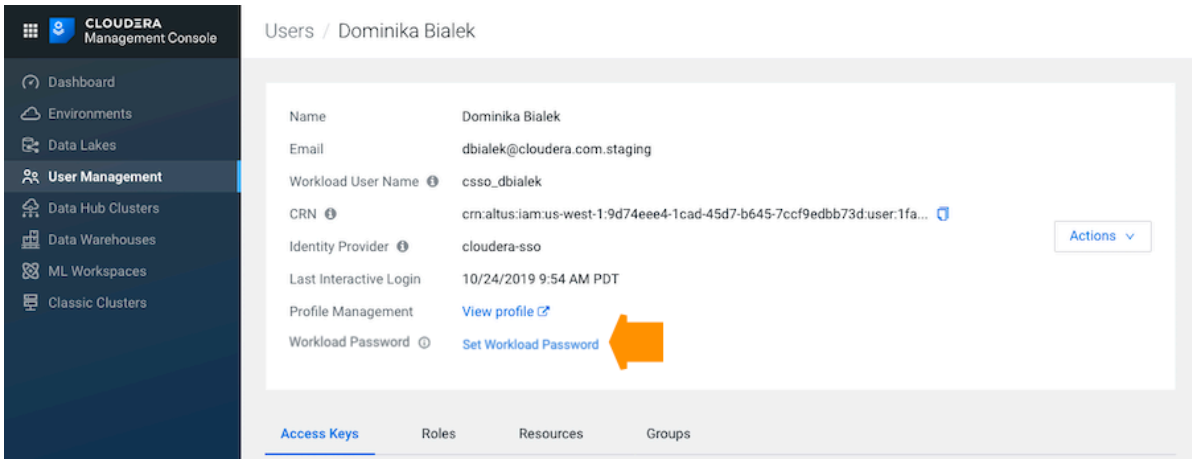

6. In the dialog box that appears, enter the new workload password twice.
7. Click Set Workload Password. A message appears saying that the password is set successfully.
8. Click Close.

For CDP CLI

Use the following command to set workload password for other users:

```
cdp iam set-workload-password  --actor-crn <value> --password <value>
```

The CRN can be obtained from Cloudera web interface from the user profile.

What to do next

Each time you reset your workload password, you must regenerate your keytab. See Retrieve keytab.

Related Information

Accessing non-SSO interfaces using workload user and password

Managing SSH keys

A Power User can add and delete SSH keys for all users and machine users, and users can add and delete their own SSH keys. Once these SSH keys are uploaded and synced, they can be used to access workload cluster nodes. RSA or ED25519 keys are supported.

Required roles: All users can manage their SSH keys from the account management page. All users can manage their SSH keys from CDP CLI, but this action requires an API access key, which can only be generated by users with the IAMUser role. As a Cloudera administrator or PowerUser, you can manage the SSH keys for all user accounts.

Manage your own SSH keys

Steps

For CDP UI

To add/delete an SSH keys via Cloudera web interface, click on your user name in the bottom left corner and then select Profile. Next, click on the SSH Keys tab.

- To add an SSH key, click on Add SSH key, then provide a description, paste your SSH public key and click Save.
- To delete click on Delete next to the SSH key that you would like to delete and then click Yes to confirm.

Once the SSH public SSH key is added and synced, the user to which the key is assigned can use a matching private SSH key to access workload cluster nodes.

For CDP CLI

You can manage your SSH keys by using the following CDP CLI commands:

```
cdp iam add-ssh-public-key
cdp iam list-ssh-publi-keys
cdp iam describe-ssh-public-key
cdp iam delete-ssh-public-key
```

Manage SSH keys for another user or machine user (admin only)

Steps

For CDP UI

To add/delete an SSH keys via Cloudera web interface, navigate to the Cloudera Management Console > User Management > Users > search for a user name > click on a user name > SSH Keys.

- To add an SSH key, click on Add SSH key, then provide a description, paste your SSH public key and click Save.
- To delete click on Delete next to the SSH key that you would like to delete and then click Yes to confirm.

Once the SSH public SSH key is added and synced, the user to which the key is assigned can use a matching private SSH key to access workload cluster nodes.

For CDP CLI

You can manage SSH keys for other users by using the following CDP CLI commands:

```
cdp iam add-ssh-public-key --actor-crn <value>
cdp iam list-ssh-publi-keys --actor-crn <value>
cdp iam describe-ssh-public-key --actor-crn <value>
cdp iam delete-ssh-public-key --actor-crn <value>
```

Generating an API access key

A Cloudera user account (a user or a machine user) must have API access credentials to access Cloudera services through the CDP CLI or API.

Required roles: Users who have the IAMUser role can generate an API access key from their own account page. As a Cloudera administrator or PowerUser, you can generate an API access key for all user accounts.

Generate your own access key

Steps

1. Sign in to the Cloudera console.
2. Click on your user name in the bottom left corner and then select Profile.
3. On the user profile page that appears, click Generate Access Key.
4. From the Cloudera home page, click Cloudera Management Console.
5. On the side navigation panel, click Users.
6. On the Users page, click the name of the user or machine user account for which you want to generate an access key.
7. On the user account page, go to the Access Keys section and click Generate Access Key.
8. Cloudera creates the key and displays the information on the screen.

   Copy the access key and private key to a text file and send it to the Cloudera user who requires it. The private key is a very long string of characters. Make sure that you copy the full string. You can optionally download the credentials file containing the access key information.
9. Click OK to exit the access key window.

   Note:

   The Cloudera console displays the API access key immediately after you create it. You must copy or download the access key ID and private key information when it is displayed. Do not exit the console without copying the private key. After you exit the console, there is no other way to view or copy the private key.

Once you've generated the access key, you can configure CDP CLI, SDK, or other utilities that require it.

Generate an access key for another user or machine user (admin only)

Steps

1. Sign in to the Cloudera console.
2. From the Cloudera home page, click Cloudera Management Console.
3. On the side navigation panel, click Users.
4. On the Users page, click the name of the user or machine user account for which you want to generate an access key.
5. On the user account page, go to the Access Keys section and click Generate Access Key.

6. Cloudera creates the key and displays the information on the screen.

   Copy the access key and private key to a text file and send it to the Cloudera user who requires it. The private key is a very long string of characters. Make sure that you copy the full string. You can optionally download the credentials file containing the access key information.
7. Click OK to exit the access key window.

   Note:

   The Cloudera console displays the API access key immediately after you create it. You must copy or download the access key ID and private key information when it is displayed. Do not exit the console without copying the private key. After you exit the console, there is no other way to view or copy the private key.

Once you've generated the access key, you can configure CDP CLI, SDK, or other utilities that require it.

Retrieve keytabs for workload users

A keytab file stores long-term keys for a principal in Kerberos. Retrieve your keytab for a specific environment.

You may need to generate a keytab for a workload user in certain Cloudera Data Hub use cases, for example long-running Spark streaming jobs, which require a keytab as a long-lived credential; or NiFi flows requiring a keytab to write data into HBase.

Note that:

- Cloudera users can retrieve their keytabs. A PowerUser can retrieve keytabs for other users.
- Each time you reset your workload password, you must regenerate your keytab.
- Keytabs are scoped to an environment, whereas workload passwords are the same for every environment. A keytab is, however, tied to the workload password. If you change the workload password, you must retrieve a new keytab. When you change a workload password, retrieve the keytab only after the user sync operation is complete.
- There are ways to generate keytabs with utilities outside of Cloudera, such as ipa-getkeytab or ktutil. You should not use these methods as they may not work as expected. For example, ipa-getkeytabcreates a keytab that may work but only temporarily.

Required roles: All users can retrieve their keytabs from the account management page. All users can retrieve their keytabs from CDP CLI, but this action requires an API access key, which can only be generated by users with the IAMUser role. As a Cloudera administrator or PowerUser, you can retrieve the keytab for all user accounts.

Before you begin

In order to retrieve a keytab for an environment, you must set workload password for that environment. See Setting the workload password.

Retrieve your own keytab

Steps

For CDP UI

1. Log in to Cloudera web interface.
2. Click on your user name in the bottom left corner and then select Profile.
3. Click on Actions > Get keytab.
4. In the pop-up window that appears, select the environment for which you would like to get the keytab.
5. Click Download.
6. Save the keytab file in a location of your choice.

For CDP CLI

To retrieve keytab for yourself, use the following command:

```
cdp environments get-keytab --environment-name <VALUE>
```

The output of the command is a base64-encoded representation of a keytab. The contents of the output must be base64 decoded and saved to a file for it to work as a keytab.

What to do next

Once you have downloaded the keytab file, you can copy it to the machine on which the cluster runs and use the keytab to authenticate as the workload user principal, or point to the keytab file when running a Spark job or other job that requires a keytab.

Retrieve keytab for another user or machine user (admin only)

Steps

For CDP UI

1. Log in to Cloudera web interface.
2. Navigate to the Cloudera Management Console > User Management, find and click on the user name of the user that you would like to retrieve a keytab for.
3. Click on Actions > Get keytab.
4. In the pop-up window that appears, select the environment for which you would like to get the keytab.
5. Click Download.
6. Save the keytab file in a location of your choice.

For CDP CLI

To retrieve keytab for another user or machine user, use the following command:

```
cdp environments get-keytab --environment-name <VALUE> --actor-crn <CRN>
```

The output of the command is a base64-encoded representation of a keytab. The contents of the output must be base64 decoded and saved to a file for it to work as a keytab.

What to do next

Once you have downloaded the keytab file, you can copy it to the machine on which the cluster runs and use the keytab to authenticate as the workload user principal, or point to the keytab file when running a Spark job or other job that requires a keytab.

Cloudera Operational Database ..

Accessing Data

Date published: 2020-08-14
Date modified: 2023-01-12

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Accessing Hue from Cloudera Operational Database..... 4

Accessing HBase REST API from Cloudera Operational Database..... 6

Accessing SQLLine from Cloudera Operational Database.....7

Accessing Hue from Cloudera Operational Database

Hue is a web-based interactive SQL editor that enables you to interact with data stored in Cloudera Operational Database. You can access the Hue user interface from the Cloudera Operational Database web user interface to create and browse HBase tables.

Procedure

1. Click Databases, and then select a database from the list.
2. Click Hue.

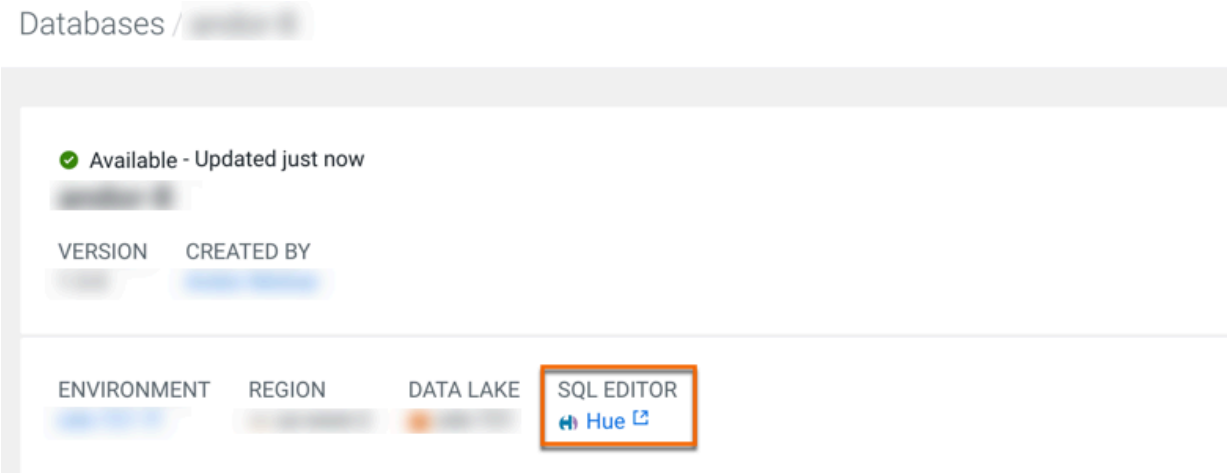

Example

You can use Hue to quickly browse large tables, create new tables, add data, modify existing cells, and also filter data using the auto-complete search.

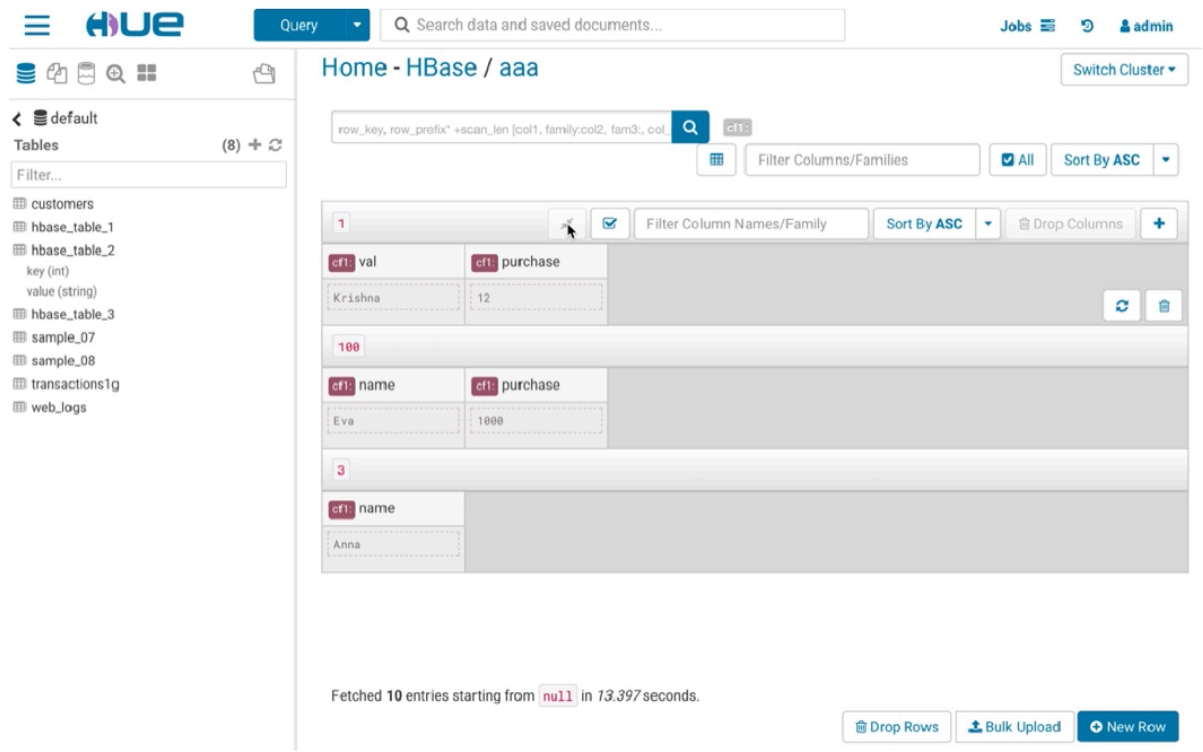


```
INVALIDATE METADATA;
SELECT * FROM my_hbase_table WHERE rowkey='abcd';
```

```
Query 47a70eda3960db:bdbef23600000000: 0% Complete (0 out of 1)
```

47a70eda3960db:bdbef23600000000

Query History | Saved Queries | Results (1)

	rowkey	value
1	abcd	my value 1

Related Information

Hue Overview

Using Hue

Use the Hue HBase app

Accessing HBase REST API from Cloudera Operational Database

You can use the Apache HBase REST server to interact with Cloudera Operational Database. Interactions happen using URLs and the REST API. REST uses HTTP to perform various actions, and this makes it easy to interface with Cloudera Operational Database using a wide range of programming languages.

Procedure

1. Click Databases, and then select a database from the list.
2. Click Connect HBase REST .
3. From the HBase REST Server URL field, copy the URL to the HBase REST Server to connect to the selected database.

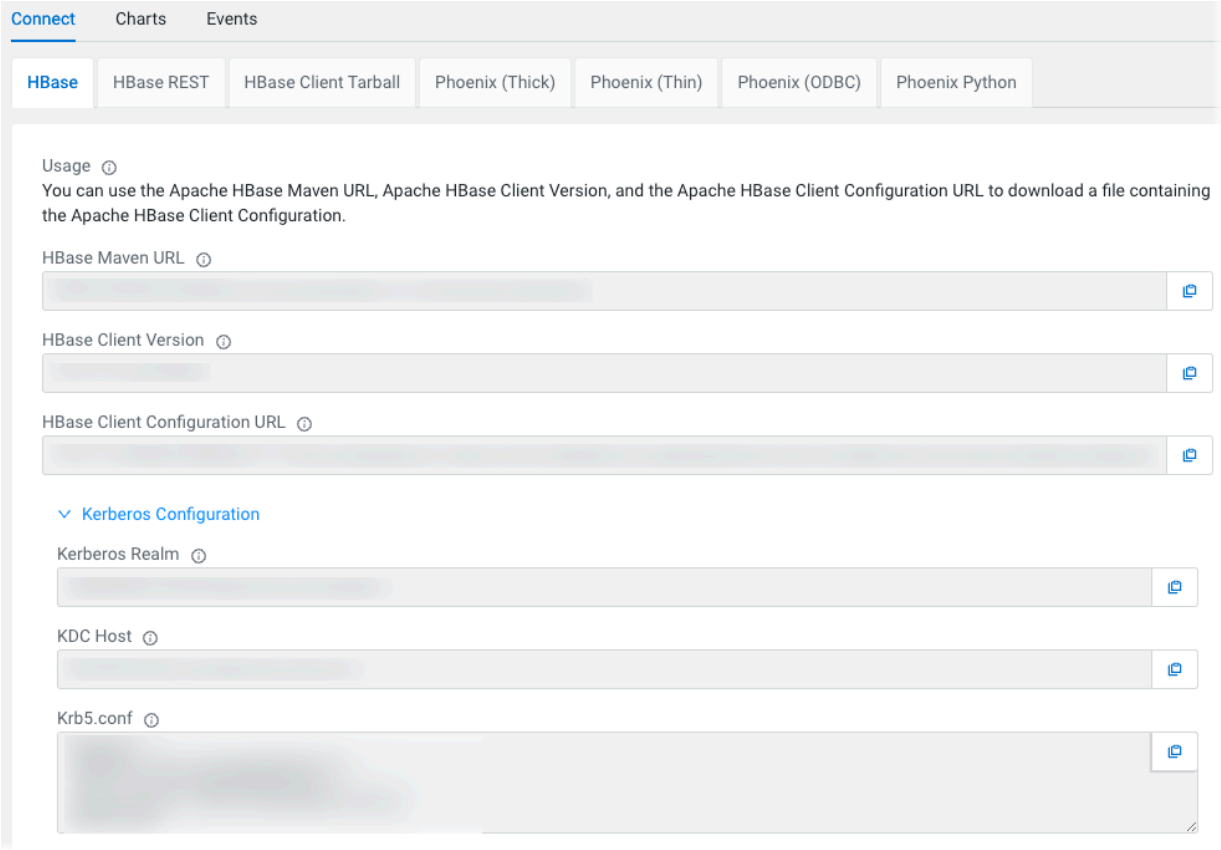

What to do next

Use the HBase REST API to interact with the HBase services, tables, and regions using HTTP endpoints. You can create tables, delete tables, and perform other operations that have the REST endpoints. For more information, see *Using the HBase REST API* using the link in the related information section.

Related Information

Use the HBase REST server

Using the REST API

Accessing SQLLine from Cloudera Operational Database

SQLLine is a command-line utility included with Cloudera Operational Database that enables you to connect and execute SQL commands using Phoenix from an edge node.

Procedure

1. Download the client JAR files for your thick or thin clients using this URL syntax.

   You can get the [****PHOENIX MAVEN URL****], [****PHOENIX_(THIN)_CLIENT_VERSION****], and the [****PHOENIX_(THICK)_CLIENT_VERSION****] information from the Database connectivity page. The URL is in the following format:

   URL for the Phoenix thick client in Cloudera Runtime 7.2.9 (environment) and higher:

   ```
   [***PHOENIX MAVEN URL***]/org/apache/phoenix/phoenix-client-hbase-2.2/
    [***PHOENIX THICK CLIENT VERSION***]/phoenix-client-hbase-2.2-[***PHOENIX
    THICK CLIENT VERSION***].jar
   ```

   URL for the Phoenix thick client in Cloudera Runtime 7.2.8 (environment) and lower:

   ```
   [***PHOENIX MAVEN URL***]/org/apache/phoenix/phoenix-client/
    [***Phoenix THICK CLIENT VERSION***]/phoenix-client-[***PHOENIX THICK
    CLIENT VERSION***].jar
   ```

   For the Phoenix thin client:

   ```
   [***PHOENIX MAVEN URL***]/org/apache/phoenix/phoenix-queryserver-client/
   [***PHOENIX THIN CLIENT VERSION***]/phoenix-queryserver-client-[***PHOENIX
    THIN CLIENT VERSION***].jar
   ```

   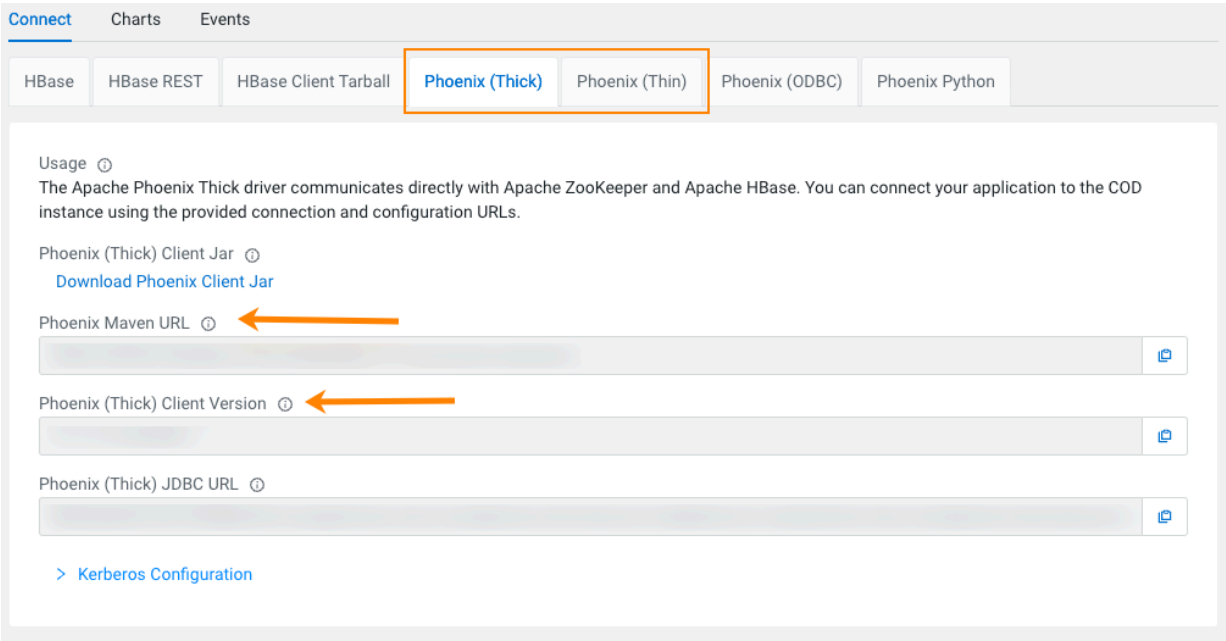

   You can use Maven to download the Phoenix client JAR files. If you only need the JAR files for SQLLine connection, you can use the curl tool to download the JAR files using the following command:

   ```
   curl -L -f -o "phoenix-client.jar" "[***PHOENIX CLIENT JAR FILE URL***]"
   ```

2. From the Databases page, download HBase client configuration Zip file using the client configuration URL .

   Note: You cannot download the client configuration file using a web browser. You must copy and paste the HBase client configuration URL to your CLI to download this file. For example, curl -f -o "hbase-config.zip" -u "csso_[***USERNAME***]" "https://[***DOWNLOAD URL***]".

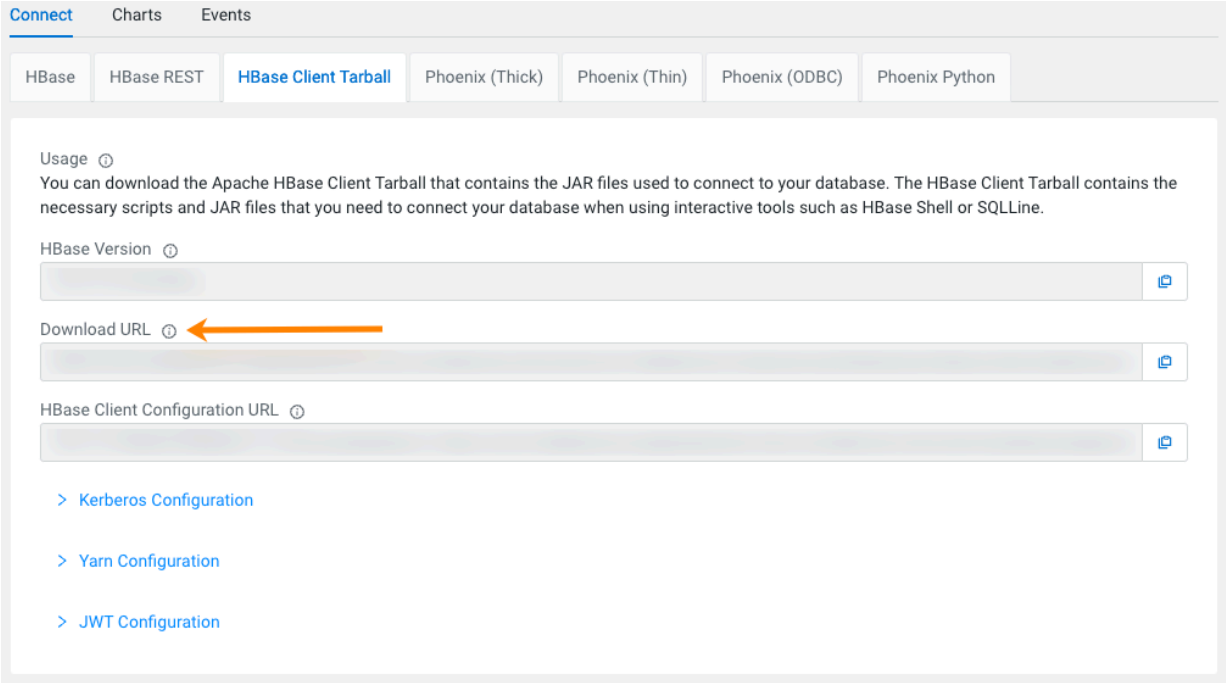

3. From the Databases page, copy the JDBC connection URL for the Phoenix (Thick) or Phoenix (Thin) client to use in the next step.
4. Run this command from your CLI:

```
java $PHOENIX_OPTS -cp "[***HBASE-CONFIGURATION***]:[***PHOENIX-CLIENT-
JAR**]]"
sqlline.SqlLine -d org.apache.phoenix.jdbc.PhoenixDriver -u
[***JDBC-CONNECTION-URL***] -n none -p none --color=true --fastConnect=
false
--verbose=true --incremental=false --isolation=TRANSACTION_READ_COMMITTED
```

Related Information

SQLLine Command Reference

Cloudera Operational Database ..

Application Support

Date published: 2020-08-14
Date modified: 2024-05-16

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Operational Database supported programming languages for application development....4

Java 2 platform and Enterprise Edition support....4

Cloudera Operational Database JDBC support....4

Cloudera Operational Database Open Database Connectivity support....5

Cloudera Operational Database .Net interface support....5

Cloudera Operational Database supported programming languages for application development

Cloudera Operational Database (COD) enables you to build applications using various programming languages. You can choose a programming language to build applications that can access data stored in your COD.

List of supported programming languages to access data stored in your COD are as follows:

- Apache Groovy
- C
- C++
- Go
- Java
- Jython
- Python
- PHP
- Scala

You can also use REST for API support.

Related Information

Operational Database Application Support - Blog post

Java 2 platform and Enterprise Edition support

You can use the Apache HBase Java API for applications to communicate with Apache HBase. Cloudera Operational Database supports the Java 2 Enterprise Edition (J2EE) along with the plugins for Eclipse.

You can perform operations on Apache HBase tables by using the Java client API. You can develop Java applications that can be used to create tables and insert data. The native Java client API offers programmatic access. For more information, see *Use the Java API* using the link in the related information section.

Related Information

Use the HBase APIs for Java

Cloudera Operational Database JDBC support

You can use the Java Database Connectivity (JDBC) interface to access the operational database. Apache Phoenix in Cloudera Operational Database provides a JDBC SQL endpoint to interact with data stored in Cloudera Operational Database.

You can create and interact with Apache HBase tables using DDL/DML statements through its standard JDBC API. Apache Phoenix JDBC driver can be easily embedded in any application that supports JDBC. The JDBC interface is provided in Cloudera Operational Database.

For more information about using the JDBC driver URL syntax and usage, see the *Client connectivity information for compiling your application for Cloudera Operational Database* and *Using JDBC API with Apache Phoenix* using the links provided in the related information section.

Related Information

Client connectivity information for compiling your application for Cloudera Operational Database

Using JDBC API with Apache Phoenix

Cloudera Operational Database Open Database Connectivity support

You can use the Open Database Connectivity (ODBC) interface to access your Cloudera Operational Database. The ODBC driver is provided by Cloudera as an additional download, but you can also use ODBC drivers from third-party providers.

The driver complies with the ODBC data standard, including important functionality such as Unicode and 32-bit and 64-bit support for high-performance computing environments on all platforms. ODBC is one of the most established and widely supported API for connecting to and working with databases.

You can download the Phoenix ODBC Driver for Cloudera Operational Database from the From the Connect tab in the Databases user interface. For more information, see *Client connectivity information for compiling your application for Cloudera Operational Database* link in the related information section.

Note: You must have a Cloudera Enterprise Support Subscription to download the ODBC driver.

Related Information

Download ODBC Driver for Apache Phoenix (v1.0.8.1011)

Cloudera Operational Database .Net interface support

You can use the C# client library to develop applications that can access data stored in the Cloudera Operational Database.

Microsoft .Net Framework is supported through multiple interfaces including ODBC, Thrift Server, and the Phoenix Query Server.

You can use the Apache Phoenix ODBC driver to connect your .Net applications to Apache Phoenix. The .Net applications that you develop interface with the Phoenix Query Server using the Avatica API and Google Protocol Buffers serialization format.

Related Information

Download Microsoft Phoenix Client

Cloudera Operational Database ..

CLI Commands

Date published: 2020-08-14
Date modified: 2024-01-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

CDP CLI commands 4

CDP CLI Beta command reference 5

--custom-user-tags in Operational Database 5

Examples for identifying resources 6

Install CDP CLI beta 7

CDP CLI commands

You can learn about the various CDP CLI commands. You can run these commands to do administration and management tasks like create or describe a Cloudera Operational Database instance. Users who have the Cloudera PowerUser role can run any command, regardless of whether they have the ODUser or ODAdmin role.

Download the CDP CLI tool

Before you can use the operataional database CLI commands, you must download the CDP CLI client for your operating system. For more information about downloading the CDP CLI client, see Installing Cloudera client.

You can see all the commands and their syntax in the *CDP CLI command reference GitHub* page.

You can see all the CDP CLI BETA commands and their syntax in the *CDP CLI BETA command reference GitHub* page.

Table 1: CDP CLI commands

Command	Description	Required Role
create-database	Create a database with the provided name in a given environment.	ODAdmin
describe-client-connectivity	List the available connectors to programmatically interact with the database and required information to use each connector.	ODUser
describe-database	Query the details of a database.	ODUser
describe-upgrade-database	Describes the upgrade availability.	ODUser
drop-database	Stop and destroy the database. Data remains in the configured cloud storage.	ODAdmin
list-databases	List all databases which the user has permission to view.	ODUser
start-database	Start an existing database.	ODAdmin
stop-database	Stop a database.	ODAdmin
collect-diagnostics	A link to the download location of the diagnostics bundle.	ODAdmin
list-diagnostics	Returns list of diagnostics collections	ODAdmin
list-supported-environments	List of environments available for the database creation.	ODAdmin

When you run the cdp opdb commands, you must provide your Cloudera Operational Database database name and the Cloudera environment name as parameters as shown in this example:

```
cdp opdb describe-database --database-name [***MY DATABASE NAME***] --envir
onment-name [***MY CLOUDERA ENVIRONMENT NAME
***]
```

Related Information

CDP CLI command reference GitHub repository

CDP CLI BETA command reference GitHub repository

CDP CLI Beta command reference

You can learn about the various CDP CLI Beta commands. You can run these commands to do administration and management tasks like create or describe a Cloudera Operational Database instance. To use any of the CDP CLI beta commands for Operational Database, you need to download and install the latest CDP CLI beta version.

Install CDP CLI Beta

To install CDP CLI Beta, see *Installing Beta CDP CLI*.

You can see all the CDP CLI Beta commands and their syntax in the *CDP CLI BETA command reference GitHub* page.

Related Information

Installing Beta CDP CLI

CDP CLI command reference GitHub repository

CDP CLI BETA command reference GitHub repository

--custom-user-tags in Operational Database

You can identify and monitor infrastructure resources that are launched in your AWS account by applying the --custom-user-tags. This is an optional tag. You can apply the --custom-user-tags tags after creating the Cloudera environment. For Cloudera Operational Database, use this feature by running the following command in the CDP CLI beta version:

```
 `cdp opdb create-database --custom-user-tags`
```

For example:

```
`cdp opdb create-database --custom-user-tags key=name1,value=value1
key=name2,value=value2 --database-name ... --environment-name ...`
```

Following is the code snippet:

```
<snip>
       --custom-user-tags  (array)  Optional  tags to apply to launched i
nfra-
       structure resources

       Shorthand Syntax:

          key=string,value=string ... (separate items with spaces)

       JSON Syntax:
          [
            {
              "key": "string",
              "value": "string"
            }
            ...
          ]
</snip>
```

Examples for identifying resources

You can identify and monitor various resources within the Cloudera Operational Database experience for AWS environments.

The following are the various resources that you can identify and monitor:

- Database names in operational database with custom-user level tags
- EC2 instances
- RDS database instances

  For example, if you run the following command through CDP CLI beta version, You can view the AWS EC2 and RDS resources launched by Cloudera Operational Database.

```
cdp.sh opdb create-database --environment-name <env-name> --database-name <d
b-name>
--custom-user-tags key=tag1,value=example key=tag2,value=value
```

Figure 1: Example of AWS EC2 resource

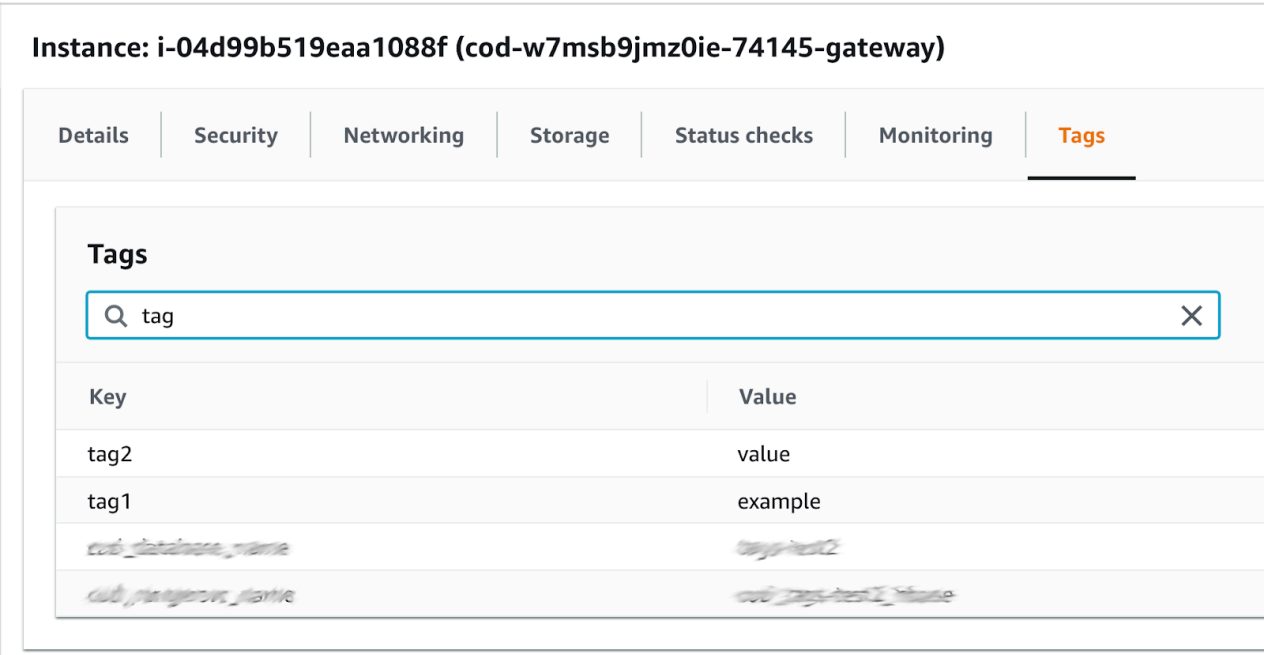

Figure 2: Example of RDS resource

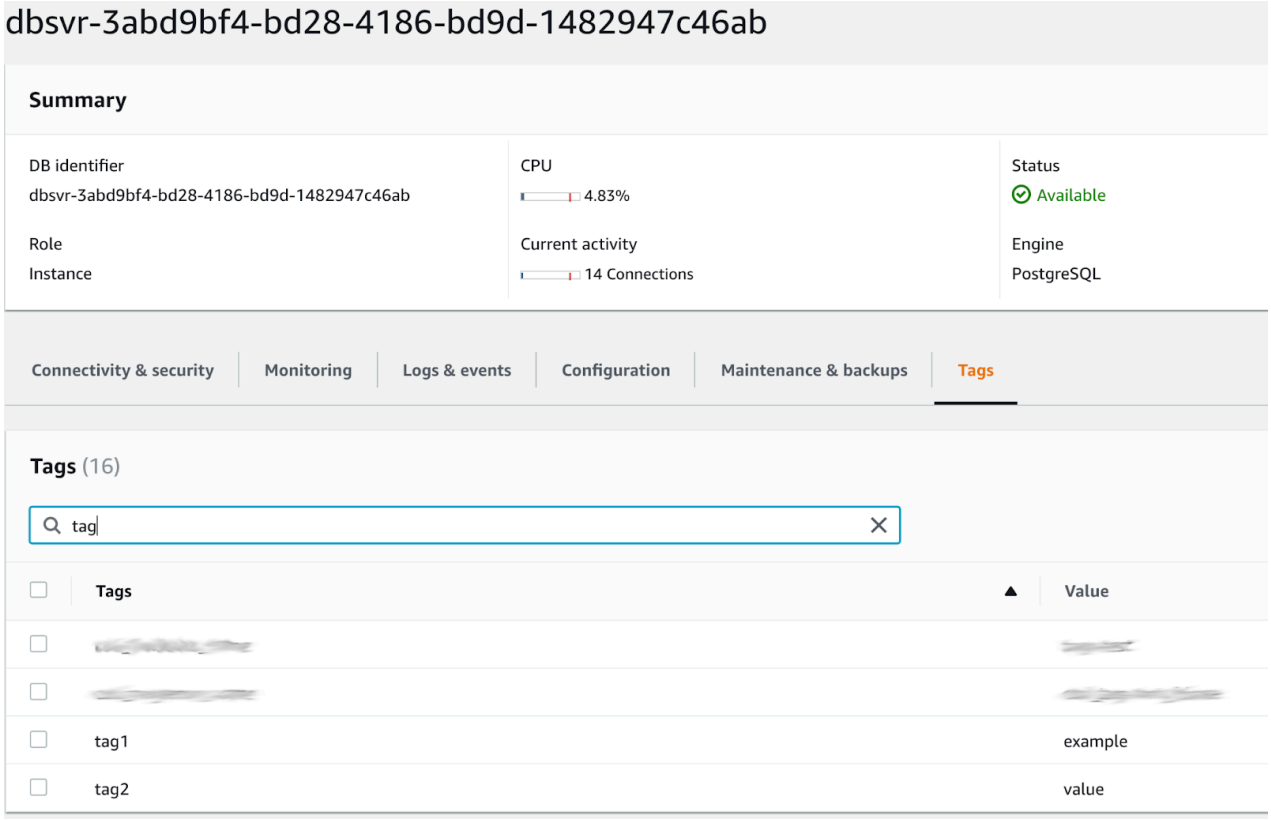

Install CDP CLI beta

If you would like to apply custom user tags on Operational Database through CDP CLI, you need to download and install CDP CLI beta version 0.9.40.

Before you begin

Do not install both the regular and beta CLIs in the same Python environment, as they use the same entry points and will therefore conflict. Either use a separate virtual environment or uninstall the cdpcli first before installing cdpcli-beta.

Procedure

1. Install Python if needed, as described in CDP CLI documentation for your platform.

   Do not install CDP CLI. Instead, proceed to step 2.
2. Run the following to install CDP CLI beta:

   ```
   pip3 install cdpcli-beta
   ```
3. Configure access keys as described in Generating an API access key

Cloudera Operational Database ..

Public Cloud Environment Checklist

Date published: 2020-08-14
Date modified: 2023-03-08

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Public cloud environment checklist for Cloudera Operational Database..... 4

AWS account requirements for Cloudera Operational Database deployment.....4

Azure account requirements for Cloudera Operational Database deployment.....5

GCP account requirements..... 5

Cloudera environments.....5

Public cloud environment checklist for Cloudera Operational Database

You must plan your cloud account requirements and resources before you create a Cloudera Operational Database. Cloudera Operational Database is deployed on the Public Cloud, and you must understand the public cloud account requirements before you deploy your Cloudera Operational Database instance.

- Currently, Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure are the supported public cloud environments. Before registering your AWS, Azure, or GCP environment in Cloudera, you must verify that your account has all the resources required by Cloudera.
- You must register your AWS, Azure, or GCP environment in Cloudera before you create a Cloudera Operational Database. You must also ensure that you read and understand the AWS or Azure account requirements so that your public cloud account has all the resources required by Cloudera and that your Cloudera administrator has adequate permissions to configure the resources and services in AWS, Azure, or GCP.

  Note:

  - You must enable an appropriate entitlement to use Cloudera Operational Database on GCP.
  - For AWS environments, Cloudera recommends you to use the EBS volumes as gp2 for better performance.
- You must also configure your public cloud environment to ensure that you have assigned the correct policies and roles. See the related information for AWS, Azure, and GCP quickstarts that will help you set up your public cloud environment for use with Cloudera Operational Database.

Related Information

AWS Onboarding Quickstart

Azure Onboarding Quickstart

GCP Onboarding Quickstart

AWS account requirements for Cloudera Operational Database deployment

When Cloudera Operational Database is deployed, it uses resources from your AWS subscription that are registered for an environment in the Cloudera Management Console.

Cloudera Operational Database uses resources from your Amazon Web Services (AWS) subscription. You must first read the Cloudera Management Console documentation to understand the basic AWS requirements before you can deploy a Cloudera Operational Database.

If you are using the HBase Java API and Phoenix Thick JDBC driver, you must configure an edge node to access data in your operational database instance. This configuration step is necessary to ensure that you can query data that is stored in your Cloudera Operational Database. See the related information section for more information.

Related Information

AWS environments

Cloudera Operational Database edge node overview

Azure account requirements for Cloudera Operational Database deployment

When Cloudera Operational Database is deployed, it uses resources from your Microsoft Azure subscription that are registered for an environment in the Cloudera Management Console.

You must first read the Cloudera Management Console documentation to understand the basic Azure requirements before you can deploy a Cloudera Operational Database.

If you are using the HBase Java API and Phoenix Thick JDBC driver, you must configure an edge node to access data in your operational database instance. This configuration step is necessary to ensure that you can query data that is stored in your Cloudera Operational Database. See the related information section for more information.

Related Information

Azure environments

Cloudera Operational Database edge node overview

GCP account requirements

When Cloudera Operational Database is deployed, it uses resources from your Google Cloud Platform (GCP) subscription that are registered for an environment in the Cloudera Management Console.

You must first read the Cloudera Management Console documentation to understand the basic GCP requirements before you can deploy a Cloudera Operational Database.

If you are using the HBase Java API and Phoenix Thick JDBC driver, you must configure an edge node to access data in your operational database instance. This configuration step is necessary to ensure that you can query data that is stored in your Cloudera Operational Database. See the related information section for more information.

Related Information

GCP environments

Cloudera Operational Database edge node overview

Cloudera environments

In Cloudera, an environment is a logical subset of your virtual private network, which is hosted on a public cloud provider. You can register as many environments as you require.

Registering an environment with Cloudera Management Console provides the Cloudera platform with access to your public cloud account. When you register an environment, resources are identified in your public cloud account that Cloudera services can access or provision.

If you are a Cloudera administrator, you must learn about how to set up your environment. For more information, see Getting started as an admin.

If you are a Cloudera user, you must learn about how to set up Cloudera and Cloudera resources. For more information, see Getting started as a user.

For more information about Cloudera environments, see AWS environments, Azure environments, and GCP environments.

Runtime versions for Cloudera Operational Database

Cloudera Operational Database supports the following Cloudera Runtime versions:

- 7.2.14
- 7.2.15
- 7.2.16
- 7.2.17
- 7.2.18

Compiling an Application against Cloudera Operational Database

Date published: 2020-08-14
Date modified: 2023-01-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Client connectivity information for compiling your application for Cloudera Operational Database.....4

Compile an application for your Cloudera Operational Database.....8

Example: run an application for the Apache HBase client.....10

Example: run an application for the Apache Phoenix thick client.....11

Example to run application using the Phoenix thin client.....12

Client connectivity information for compiling your application for Cloudera Operational Database

When compiling your application for your Cloudera Operational Database you need information about the connectivity options that helps your applications interact with Cloudera Operational Database. You can get that information either using the CLI or the Cloudera Operational Database user interface.

The following is a graphical representation of the necessary steps to compile an application against Cloudera Operational Database.

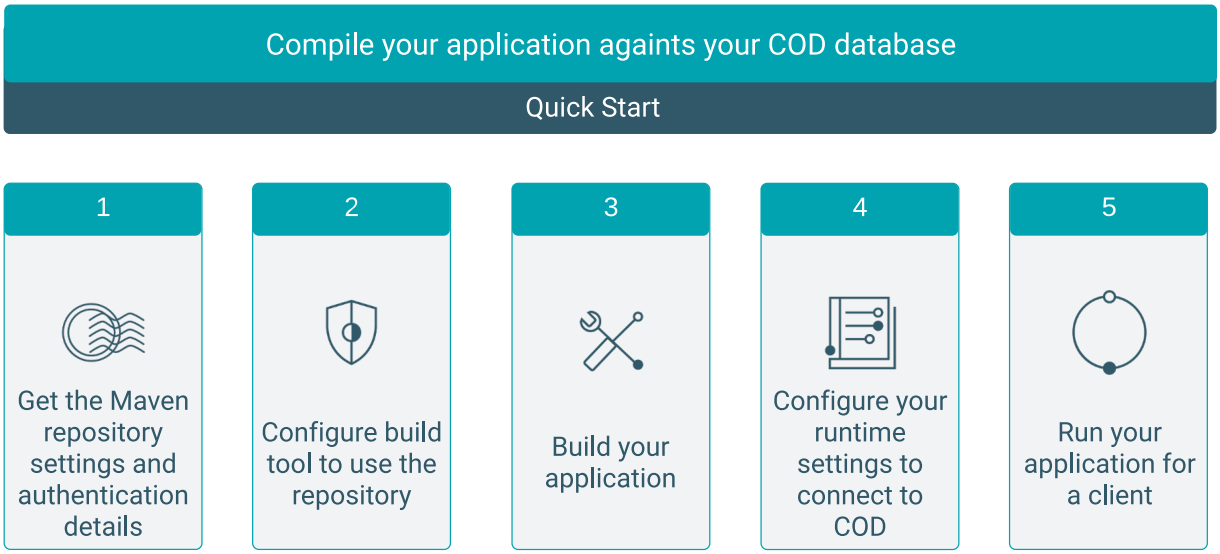

From the Connect tab in the Databases user interface, you can get information about the available client connectivity configuration options in Cloudera Operational Database:

- HBase: Apache HBase Maven URL, Apache HBase client version, and the Apache HBase client configuration URL used to download a file containing the Apache HBase client configuration
- HBase REST: Apache HBase REST Server version and Apache HBase REST Server URL
- HBase Client Tarball: Apache HBase version, download URL of the client tarball, and the Apache HBase client configuration URL used to download a file containing the Apache HBase client configuration
- Phoenix (Thick): Phoenix thick JDBC driver URL and version number of the client artifacts
- Phoenix (Thin): Phoenix thin JDBC driver URL and version number of the client artifacts
- Phoenix (ODBC): Phoenix ODBC driver URL. You need to authenticate again to access this driver

You can also download the YARN and Kerberos configuration from these tabs where applicable.

Maven repository containing the HBase client artifacts

Maven repository containing HBase client artifacts is provided in the HBase tab. You can get the following information from this tab:

- HBase Maven URL: Use this URL to access the Maven repository containing HBase client artifacts
- HBase Client Version: You can view the version of HBase client artifacts in the Maven repository
- HBase Client Configuration URL: You can get the endpoint which provides a ZIP file with HBase client configuration files

You cannot download the tarball using a web browser, you must instead use a tool like curl in the CLI to download the tarball. For example,

```
curl -O -u <username> "<download url>.tar.gz
```

Kerberos configuration information is provided in the HBase tab. Only authenticated clients can access a database in Cloudera Operational Database. You need the following Kerberos information that you can use to connect your application to Cloudera Operational Database:

- Kerberos Realm: You need the Kerberos realm information for connecting your clients requiring Kerberos authentication
- KDC Host: You need the Kerberos Key Distribution Center (KDC) hostname for your clients requiring Kerberos authentication
- Krb5.conf: You can download a sample krb5.conf file for your clients requiring Kerberos authentication

The /etc/krb5.conf file is the configuration a client uses to access a realm through its configured KDC. The krb5.conf maps the realm to the available servers supporting those realms. It also defines the host-specific configuration rules for how tickets are requested and granted. The krb5.conf file is used to determine the default Kerberos realm and KDC. Add the contents into the /etc/krb5.conf file on your edge node. For more infromation, see Configure Kerberos.

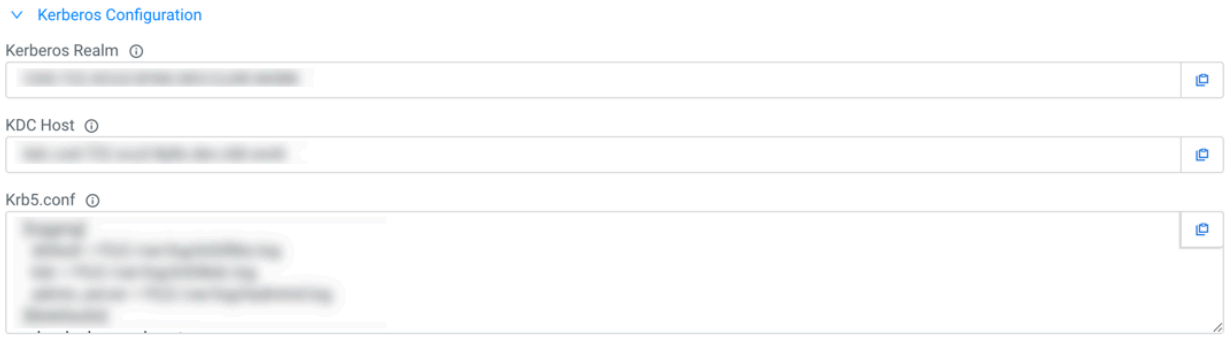

HBase REST

You can use the Apache HBase REST server to interact with Cloudera Operational Database. You can get the following information from this tab:

- HBase REST Server Version: You can see the version of HBase REST server
- HBase REST Server URL: You can get the URL to the HBase REST Server to connect to your Cloudera Operational Database

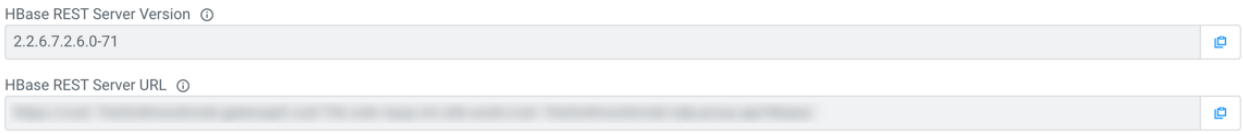

HBase Client Tarball

You can download the HBase client tarball that contains the JAR files used to connect to your database. The HBase client tarball contains the necessary scripts and JAR files that you need to connect your database when using interactive tools such as HBase Shell or SQLLine.

Click Connect HBase Client Tarball tab, and then use the Download URL to download the tarball.

You cannot download the tarball using a web browser, you must instead use a tool like curl in the CLI to download the tarball. For example,

```
curl -O -u <username> "<download url>.tar.gz"
```

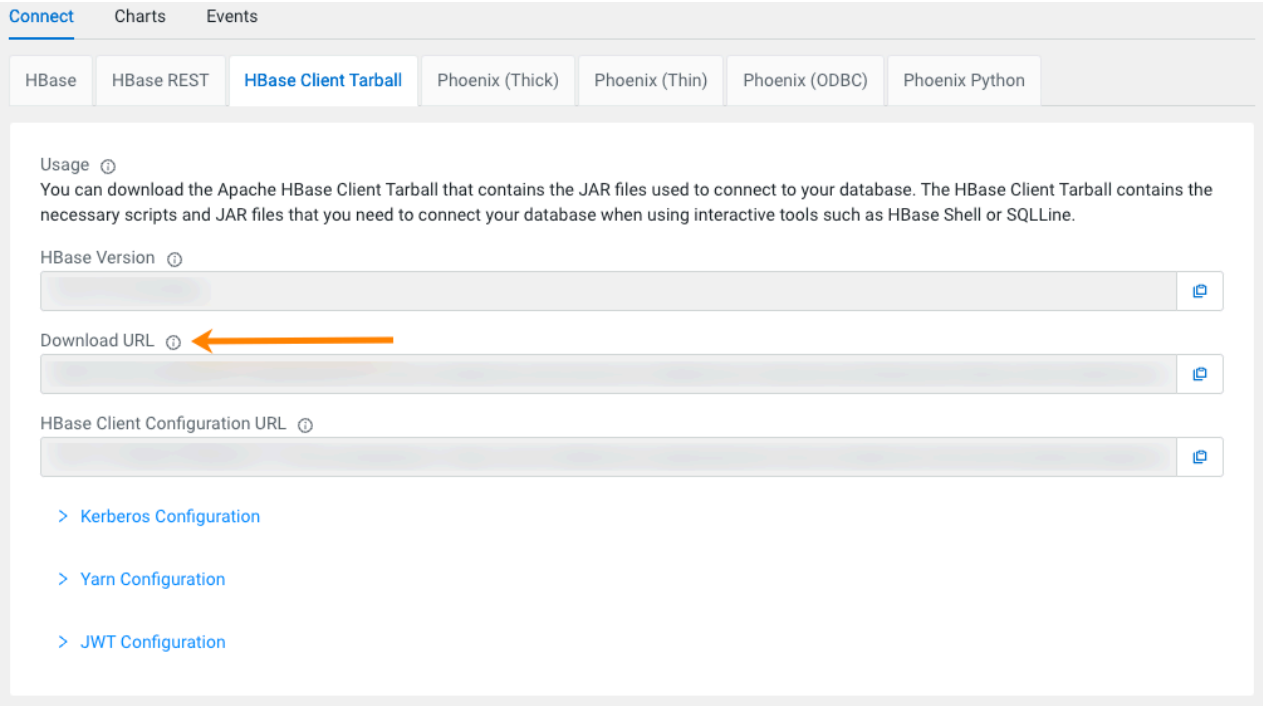

Kerberos configuration information is provided in the HBase Client Tarball tab. Only authenticated clients can access a database in Cloudera Operational Database. You need the following Kerberos information that you can use to connect your application to Cloudera Operational Database:

- Kerberos Realm: You need the Kerberos realm information for connecting your clients requiring Kerberos authentication
- KDC Host: You need the Kerberos Key Distribution Center (KDC) hostname for your clients requiring Kerberos authentication
- Krb5.conf: You can download a sample krb5.conf file for your clients requiring Kerberos authentication

YARN configuration information is provided in the HBase Client Tarball tab. You need this additional configuration information to submit YARN applications that access data stored in Cloudera Operational Database.

You must create the ssl-client.xml in the downloaded and unzipped YARN Archive if it is not present in the archive folder. You need this file to run YARN workloads against Cloudera Operational Database.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>ssl.client.truststore.type</name>
    <value>jks</value>
  </property>
  <property>
    <name>ssl.client.truststore.location</name>
```

```
    <value>[[/path/to/truststore.jks]]</value>
  </property>
  <property>
    <name>ssl.client.truststore.password</name>
    <value>[[password]]</value>
  </property>
  <property>
    <name>ssl.client.truststore.reload.interval</name>
    <value>10000</value>
  </property>
</configuration>
```

- YARN Archive: You can use this URL to download a ZIP archive that contains the necessary YARN client configuration files
- TLS Truststore: You can use this command to download a Java KeyStore file that contains the certificate authority necessary to submit YARN jobs
- TLS Truststore Password: You can use this command to download the password for the TLS truststore

Phoenix JDBC thick and thin drivers

Apache Phoenix has two kinds of JDBC drivers, thick and thin. The thick driver communicates directly with Apache ZooKeeper and Apache HBase, and the thin driver does this through the Phoenix Query Server. You can connect your application to the Cloudera Operational Database instance using the provided connection and configuration URLs.

You can get the URL for the thick and thin JDBC driver using:

- Cloudera command-line interface: Use the cdp opdb describe-client-connectivity --database-name [***YOUR DATABASE***] --environment-name [***YOUR ENVIRONMENT***] command
- Cloudera Operational Database web user interface: Click Databases *YOUR DATABASE* Connect tab, and then select the Phoenix (Thick) or Phoenix (Thin) client

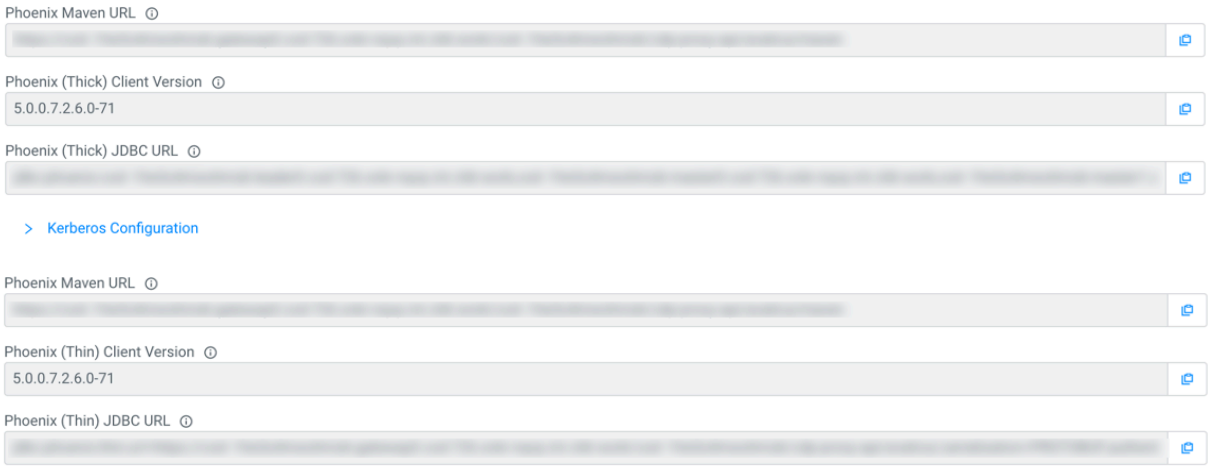

Phoenix ODBC driver

ODBC is one of the most established and widely supported API for connecting to and working with databases. Use the Database Open Database Connectivity (ODBC) interface to access the operational database. The ODBC driver is provided by Cloudera as an additional download, but you can also use ODBC drivers from third-party providers. You can download the ODBC driver for your operating system using the links in the Phoenix (ODBC) tab.

You can get the following information:

- Phoenix ODBC driver download URL: A URL where Phoenix ODBC driver is available. You may need to authenticate again to access this driver
- Phoenix ODBC URL: The basis of a ODBC URL for the Phoenix thin driver to connect to this database

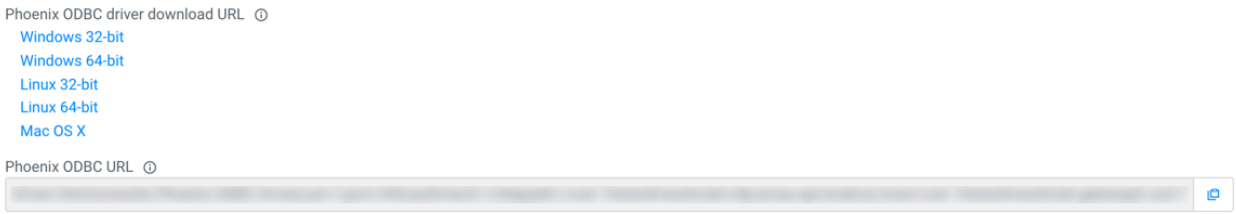

Related Information

Configuring Kerberos

Compile an application for your Cloudera Operational Database

Once you have created your application and a database using Cloudera Operational Database, you have to compile your application for your database.

Before you begin

- Set your Cloudera workload password. For more information, see the *Setting the workload password* documentation in the related information section.
- Grant the ODUser role to the machine user using the Cloudera Management Console Environments *ENVIRONMENT_NAME* Actions Manage Acces page for their Cloudera environment. By setting the ODUser role you can grant a number of rights in the system that allows you to access the Cloudera Operational Database using the machine user's workload password.
- Add synchronized users from User Management Service in the Cloudera Control Plane into the environment in which your Cloudera Operational Database database is running.

Procedure

1. Get the Maven repository location to fetch JAR files.

   There are two ways to get the necessary information:

   - In command line: Using the cdp opdb describe-client-connectivity --database-name <*YOUR_DATABASE*> --environment-name <*YOUR_ENVIRONMENT*> command.
   - In the Cloudera Operational Database user interface: Clicking the Connect bar and selecting the applicable client.

   You have to use the version and the mavenURL attributes in your Maven project and configuration.

   The following is an example about how to fetch the required HBase information to build your application:

```
$ cdp opdb describe-client-connectivity --database-name <YOUR_DATABASE> --
environment-name <YOUR_ENVIRONMENT>  | jq '.connectors[] | select(.name
 == "hbase")'
{
  "name": "hbase",
  "version": "2.2.3.7.2.0.0-219",
  "kind": "LIBRARY",
  "dependencies": {
    "mavenUrl": "HTTPS://REPOSITORY.CLOUDERA.COM/ARTIFACTORY/CLOUDERA-
REPOS"
  },
  "configuration": {
    "clientConfigurationUrl": "HTTP://CLIENT_CONFIGURATION_URL/.../
SERVICES/HBASE/CLIENTCONFIG"
  },
  "requiresKerberos": true
```

   The following example fetch the required Phoenix information to build your application:

```
 cdp opdb describe-client-connectivity --database-name <YOUR_DATABASE> --e
nvironment-name <YOUR_ENVIRONMENT> | jq ".connectors[] | select(.name == \
"phoenix-thick-jdbc\") | .dependencies.mavenUrl"
  cdp opdb describe-client-connectivity --database-name <YOUR_DATABASE> --
environment-name <YOUR_ENVIRONMENT> | jq ".connectors[] | select(.name ==
 \"phoenix-thin-jdbc\") | .version"

Phoenix-thick
"HTTPS://REPOSITORY.CLOUDERA.COM/ARTIFACTORY/CLOUDERA-REPOS"
"5.0.0.7.2.0.0-128"
Phoenix-thin
"HTTPS://REPOSITORY.CLOUDERA.COM/ARTIFACTORY/CLOUDERA-REPOS"
"5.0.0.7.2.0.0-128"
```

2. Modify your application's settings.

   Ensure your application's settings.xml file uses the correct URL and version for your Cloudera Operational Database database.

   An example when using NoSQL client:

```
<project>
  <dependencies>
  <!-- NoSQL client for Cloudera Operational Database -->
    <dependency>
      <groupId>org.apache.hbase</groupId>
      <artifactId>hbase-shaded-client</artifactId>
      <version>2.2.3.7.2.0.0-219</version>
    </dependency>
  </dependencies>
  ...
```

```
  <repositories>
  <!-- Define our Cloudera Operational Database repository; this would be
 given to us by Cloudera Operational Database itself -->
    <repository>
      <id>nosql-odx</id>
<url>HTTPS://MAVEN_URL/CDP-PROXY/HBASE/JARS</url>
         <name>Cloudera NoSQL Cloudera Operational Database Repository</
name>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>
</project>
```

An example when using SQL client:

```
<project>
  <dependencies>
    <!-- SQL client for ODX -->
    <dependency>
      <groupId>org.apache.phoenix</groupId>
      <artifactId>phoenix-client</artifactId>
      <version>5.0.0.7.2.0.0-128</version>
    </dependency>
    <dependency>
      <groupId>org.apache.phoenix</groupId>
      <artifactId>phoenix-queryserver-client</artifactId>
      <version>5.0.0.7.2.0.0-128</version>
    </dependency>
  </dependencies>
  ...
  <repositories>
  <!-- Define our Cloudera Operational Database repository -->
    <repository>
      <id>sql-cod</id>
<url>HTTPS://MAVEN_URL/CDP-PROXY-API/AVATICA/MAVEN</url>
      <name>Cloudera SQL Cloudera Operational Database Repository</name>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>
</project>
```

3. Build your application.
4. Run your application for the applicable client.

Related Information

Setting the workload password

Example: run an application for the Apache HBase client

Checking the example of how to run a Maven application for the HBase client gives you better understanding about how to run your own application for the HBase client.

Before you begin

The requiredKerberos option is set to true for the Apache HBase client. The application must be run for this client from a computer which:

- Has internal network access to the public cloud in which the database is deployed
- Can resolve the internal hostnames of the database
- Can obtain a Kerberos ticket from the database's Kerberos Key Distribution Center (KDC)

One way to run the Apache HBase client applications is to launch an edge node in your cloud provider which meets the above requirements.

Ensure that you download the directory containing the Apache HBase client configuration files from clientConfig urationUrl provided in describe-client-connectivity response. This is a protected endpoint, and you have to use your Cloudera workload credentials to access this endpoint. You can also get the Apache HBase client configuration as a client tarball from the Cloudera Operational Database web user interface Connect tab.

Procedure

1. Use clientConfigurationURL from the describe-client-connectivity response to obtain the necessary configuration details to communicate with Apache HBase:

```
$ cdp opdb describe-client-connectivity --database-name [***YOUR DATABASE
 NAME***] --environment-name [***YOUR ENVIRONMENT NAME***]  | jq '.connect
ors[] | select(.name == "hbase") | .configuration.clientConfigurationUrl'
"https://client_Configuration_URL/.../services/hbase/clientConfig"
$ curl -k -o clientConfig.zip -u '[***USERNAME***]:[***CLOUDERA WORKLOAD
 PASSWORD***]' https://client_Configuration_URL/.../services/hbase/client
Config
```

   You can build the application in your local machine and copy the JAR files to the remote node using the following commands:

```
$ scp -r target ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws
.com:
$ scp clientConfig.zip ec2-user@my-ec2-bastion-host.us-west-2.compute.amaz
onaws.com:
$ ssh ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws.com "sudo
yum install -y java-1.8.0-openjdk"
$ ssh ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws.com "unzip
 clientConfig.zip"
```

2. Ensure that you have a Kerberos ticket:

```
$ kinit [***USERNAME***] Password: [***PASSWORD***]
$ java -cp target/nosql-libs/*:target/nosql-exemplar-0.0.1-SNAPSHOT.jar
:hbase-conf com.cloudera.odx.nosql.Client
```

3. Run your application.

Example: run an application for the Apache Phoenix thick client

Checking the example of how to run a maven application for the Phoenix thick client gives you better understanding about how to run your own application for the Phoenix thick client.

Before you begin

The requiredKerberos option is set to true for the Phoenix thick client. This means that the application must be run for this client from a computer which:

- Has internal network access to the public cloud in which the database is deployed
- Can resolve the internal hostnames of the database
- Can obtain a Kerberos ticket from the database's Kerberos Key Distribution Center (KDC)

One way to run an Apache Phoenix thick client application is to launch an edge node in your cloud provider which meets the above requirements.

Procedure

1. Use the JDBC URL from the describe-client-connectivity command to run the example.

   One method is to build on your local machine and copy the JAR files to the remote node:

```
$ scp -r target ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws
.com:
$ scp clientConfig.zip ec2-user@my-ec2-bastion-host.us-west-2.compute.amaz
onaws.com:
$ ssh ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws.com "sudo
yum install -y java-1.8.0-openjdk"
```

2. Ensure that you have a Kerberos ticket and run your application for the Phoenix thick client

```
 kinit USERNAME
$ java -cp target/sql-libs/*:target/sql-exemplar-0.0.1-SNAPSHOT.jar:hbase-
conf com.cloudera.odx.sql.Client "[***PHOENIX THICK JDBC URL***]"
```

3. Run your application.

Related Information

Cloudera Operational Database edge node overview

Example to run application using the Phoenix thin client

Checking the example of how to run a maven application for the Phoenix thin client gives you better understanding about how to run your own application for the Phoenix thin client.

The requiredKerberos options set to false for the Phoenix thin client which means that it can be used from virtually any node. In this example, the client runs from the local machine.

For the Apache Phoenix thin client the describe-client-connectivity call returns a base JDBC URL . You must append the following attributes to the URL which are specific to your identity:

- avatica_user: your Cloudera username (required)
- avatica_password: your Cloudera worload password (required)
- truststore: a truststore for your Cloudera Knox gateway (optional)
- truststore_password: the password for the truststore file (optional)

You can use Maven to ease launching this application, but a standalone Java program is similarly launched:

```
$ mvn exec:exec -Dexec.executable=java -Dexec.args='-cp target/sql-libs/*:ta
rget/sql-exemplar-0.0.1-SNAPSHOT.jar com.cloudera.odx.sql.ThinClient "jdbc:p
hoenix:thin:url=[***PHOENIX THIN JDBC URL;serialization=PROTOBUF;authenti
cation=BASIC;avatica_user=[***USERNAME***];avatica_password=[***PASSWORD***]
;truststore=[***CDP-TRUSTSTORE.JKS***];truststore_password=[***TRUSTSTORE
 PASSWORD***]"'
```

Or, you can launch the application without the help of Maven:

```
$ java -cp target/sql-libs/*:target/sql-exemplar-0.0.1-SNAPSHOT.jar com.clou
dera.odx.sql.ThinClient "jdbc:phoenix:thin:url=[***PHOENIX THIN JDBC URL;ser
ialization=PROTOBUF;authentication=BASIC;avatica_user=[***USERNAME***];ava
tica_password=[***PASSWORD***];truststore=[***CDP-TRUSTSTORE.JKS***];trustst
ore_password=[***TRUSTSTORE PASSWORD***]"
```

Cloudera Operational Database ..

Concepts and Use Cases

Date published: 2022-02-17
Date modified: 2023-11-30

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Operational Database service in the Cloudera Public Cloud..... 4
Cloudera Operational Database quick start.....5
Cloudera Operational Database benefits.....5

Operational Database use cases..... 6

Autoscaling in public cloud environments.....7

Cloudera Operational Database integration with RAZ.....9
Checking RAZ for S3 and ADLS Gen2.....10

Ephemeral Storage..... 10

HBase Store File Tracking..... 11
Store File Tracking usage.....11
Cloudera Operational Database migration..... 12
Restrictions in Store File Tracking..... 12

Multi-AZ deployment on Cloudera Operational Database..... 12
Advantages of Multi-AZ deployments.....13
How to enable Multi-AZ deployment on Cloudera Operational Database.....14
Troubleshooting Multi-AZ deployments.....14
Unable to join the cluster automatically..... 14
OMID service fails to recover..... 15
Restrictions in Multi-AZ deployment on Cloudera Operational Database.....15

Cloudera Operational Database service in the Cloudera Public Cloud

Cloudera Operational Database is a service that runs on the Cloudera Public Cloud. Cloudera Operational Database enables you to create a new Cloudera Operational Database with a single click and auto-scales based on your workload.

Cloudera Operational Database delivers a real-time, always available, scalable operational database that serves traditional structured data alongside unstructured data within a unified operational and warehousing platform.

Cloudera Operational Database is powered by Apache HBase and Apache Phoenix. In Cloudera Operational Database, you use Apache HBase as a datastore with HDFS and/or S3 providing the storage infrastructure. You have the choice to either develop applications using one of the native Apache HBase API, or you can use Apache Phoenix for data access. Apache Phoenix is a SQL layer that provides a programmatic ANSI SQL interface. It works on top of Apache HBase, and it makes it possible to handle data using standard SQL queries and Apache Phoenix commands. You can use Cloudera Operational Database in the Public Cloud or Private Cloud.

You can access Cloudera Operational Database from the Cloudera console: https://console.cdp.cloudera.com/. The Cloudera Operational Database experience is a tile that you can click on your Cloudera console home screen.

Related Information

Cloudera Operational Database overview
Cloudera Operational Database quick start
Create a database using Cloudera Operational Database
Compiling applications for Cloudera Operational Database
Cloudera Operational Database

Cloudera Operational Database quick start

In this quickstart, we would like to guide you through the necessary steps to build and manage your applications using Cloudera Operational Database.

Before you start, ensure that you have created a Cloudera environment. For more information, see *Getting started as an admin*.

The following graphics represents the top tasks hierarchically that you need to perform to kick start with Cloudera Operational Database.

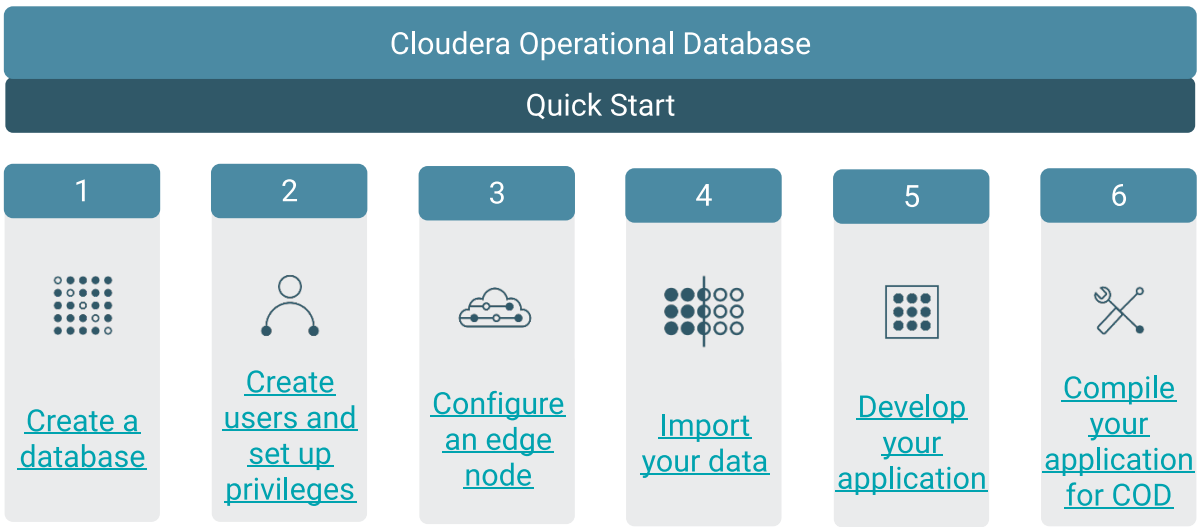

rect 26, 274, 120, 338 Creating a database using Cloudera Operational Database
rect 159, 254, 254, 353 Managing user access and authorization
rect 296, 255, 389, 349 Cloudera Operational Database edge node overview
rect 429, 273, 526, 343 Importing and restoring data into Cloudera Operational Database database
rect 563, 266, 664, 352 Cloudera Operational Database supported languages
rect 703, 254, 801, 355 Client connectivity information for compiling against Cloudera Operational Database

Related Information

Cloudera Operational Database service in the Cloudera Public Cloud

Getting started as an admin

AWS environments

Azure environments

GCP environments

Managing user access and authorization

Compiling applications for Cloudera Operational Database

Cloudera Operational Database benefits

Cloudera Operational Database benefits

You deploy Cloudera Operational Database on a public cloud infrastructure that provides you with capabilities and flexibility that your on-premises hardware sometimes cannot offer. Using an existing Cloudera environment, you can quickly create an operational database with a single click. You can launch a database with the durable and consistent storage technology you may already be familiar with if you have used CDH or HDP, but with none of the legacy complexity.

Data access

Access data stored in Cloudera Operational Database with the Apache HBase Java API, the Phoenix JDBC driver, or the Phoenix thin client JDBC driver. You can also use other Cloudera components and experiences to help you with data ingress.

Cloudera Shared Data Experience support

Cloudera Shared Data Experience Data Lake provides Cloudera Operational Database with common security, auditing, and lineage capabilities used by other Cloudera experiences. You can pause and resume your Cloudera Operational Database instances for Research and Development environments to optimize cloud cost.

Auto-scaling

Auto-scaling means that the capabilities of your database can grow to automatically handle increased load against your database, and shrink automatically to reduce your costs without sacrificing availability. Cloudera Operational Database continuously monitors the services and periodically collects all the metrices. Cloudera Operational Database meets all the latency requirements by analyzing the collected metrices. Cloudera Operational Database also fine-tunes itself to improve performance over time.

Cloudera Operational Database monitors the underlying the services and automatically scales up or down the services to ensure that the latency and RPC metrices are met. Cloudera Operational Database enables you to create a cluster quickly and has auto-scaling to help you with your different workload requirements. For more information, see Autoscaling in public cloud environments on page 7.

Auto-healing

Cloudera Operational Database continuously monitors the clusters and if any failure scenario is encountered, Cloudera Operational Database automatically heals and repairs them. For example, Cloudera Operational Database monitors the instances on the cluster in a Cloudera environment, and if any instance is broken or missing, Cloudera Operational Database recreates the instances.

Related Information

Cloudera Operational Database overview
Cloudera Operational Database quick start
HDFS overview
Introduction to Data Lakes
Introduction to Data Access
Cloudera Shared Data Experience

Operational Database use cases

You can use the Cloudera Operational Database service in online transaction processing (OLTP) use cases and other low-latency and high-throughput application use cases.

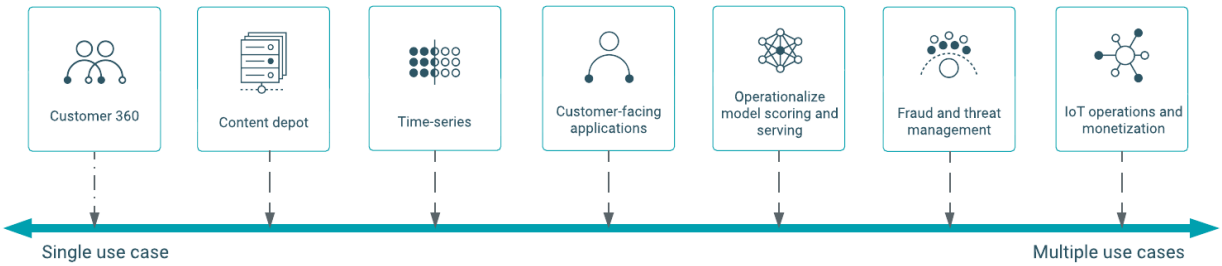

Customer 360

- Address the far-reaching effects of the shift in consumer expectations by enabling a holistic view of your business and your customers, from all products, systems, devices, and interaction channels
- Deliver a consistent, personalized, context specific, and relevant experience
- Build churn prediction models to identify at-risk customers and proactively target them with retention programs

Content Depot

- Ensure that all users can access data because of high concurrency and low latency
- Build data-based applications that distribute custom, easy-to-digest information across your organization

Time-series

Include real-time data and analysis into decision points across your organization

Customer-facing applications

- Enable serving analytics on mobile and web applications directly to end-customers
- Use as a key-value store for applications

Operationalize model scoring and serving

- Build and score models on operational data for prevention, optimization, prescription, and prediction
- Increase conversion rate of cross-sell and upsell opportunities
- Predict credit-worthiness and lifetime customer value

Fraud and threat management

Perform fraud model serving and detection

IoT - Operation and monetization

- Leverage IoT to evolve or change your business model and operations for greater efficiencies
- Provide an up-to-the-minute picture of the status of the fleet through real-time monitoring, alerting, and diagnosis
- Deliver economic value by enabling new business models
- Increase conversion rate of cross-sell and upsell opportunities

Operational excellence

- Achieve operational excellence by reducing the total cost if ownership (TCO), improving efficiency, and eliminating threats
- Decrease network downtime using predictive maintenance enabled by active collection and monitoring of network data
- Optimize equipment performance and costs using real-time IoT analytics

Autoscaling in public cloud environments

You can scale up or down the resources allocated to your database automatically by ensuring that an autoscaling event is triggered if pre-defined acceptable latency policy is not met. This ensures predictable database performance.

An autoscaling event is triggered if the rolling average value of the combined metrics reaches a pre-defined threshold value.

Autoscaling does the following to ensure that you have predictable database performance when scaling up or down:

- Gradual increase and decrease to the number of nodes (cluster does not shrink or expand drastically).

- Sufficient cooldown period between scaling up or down to avoid frequent change in cluster size.

You can see the results of scale-up or scale-down on the Databases Charts page. This page shows you graphs for concurrent clients and RPC latency that provides information about how Cloudera Operational Database is scaling up or down.

The scaling metric responds to the increase or decrease in the nodes, but there may be cases where increasing or decreasing the number of nodes does not have an effect because of the kind of workload that is running or if there are limited regions for a table.

The autoscaling framework got extended and besides the latency-based metrics, the database can scale based on HDFS disk utilization, latency, average region count per RegionServer, and bucket cache usage for cloud with ephemeral storage type of Cloudera Operational Database.

For more information, see *Fast autoscaling in Cloudera Operational Database*.

To check which autoscaling parameters are set for the Cloudera Operational Database cluster, you can use the `describe-database` CDP CLI command.

`cdp opdb describe-database --environment-name` *[CLOUDERA ENVIRONMENT NAME]* `--database-name` *[DATABASE NAME]*

To define the autoscaling, you can use the `update-database` CDP CLI command and define the required parameters in the autoScalingConfig option. Following is a sample command.

`cdp opdb update-database --environment-name` *[CLOUDERA ENVIRONMENT NAME]* `--database-name` *[DATABASE NAME]* `--auto-scaling-parameters` *"MINWORKERSFORDATABASE=<MIN_WORKER_NODES>,MAXWORKERSFORDATABASE=<MAX_WORKER_NODES>"*

The minimum and maximum nodes are limited with the minWorkersForDatabase and maxWorkersForDatabase autoscaling configurations.

Important: If the minimumBlockCacheGB is set to a non-default value and if the number of worker nodes required to support this amount of cache exceeds the maxWorkersForDatabase configured while creating the Cloudera Operational Database cluster, the maxWorkersForDatabase is automatically adjusted to 1.5 times the required number of worker nodes.

For more information, see *describe-database* CDP CLI command.

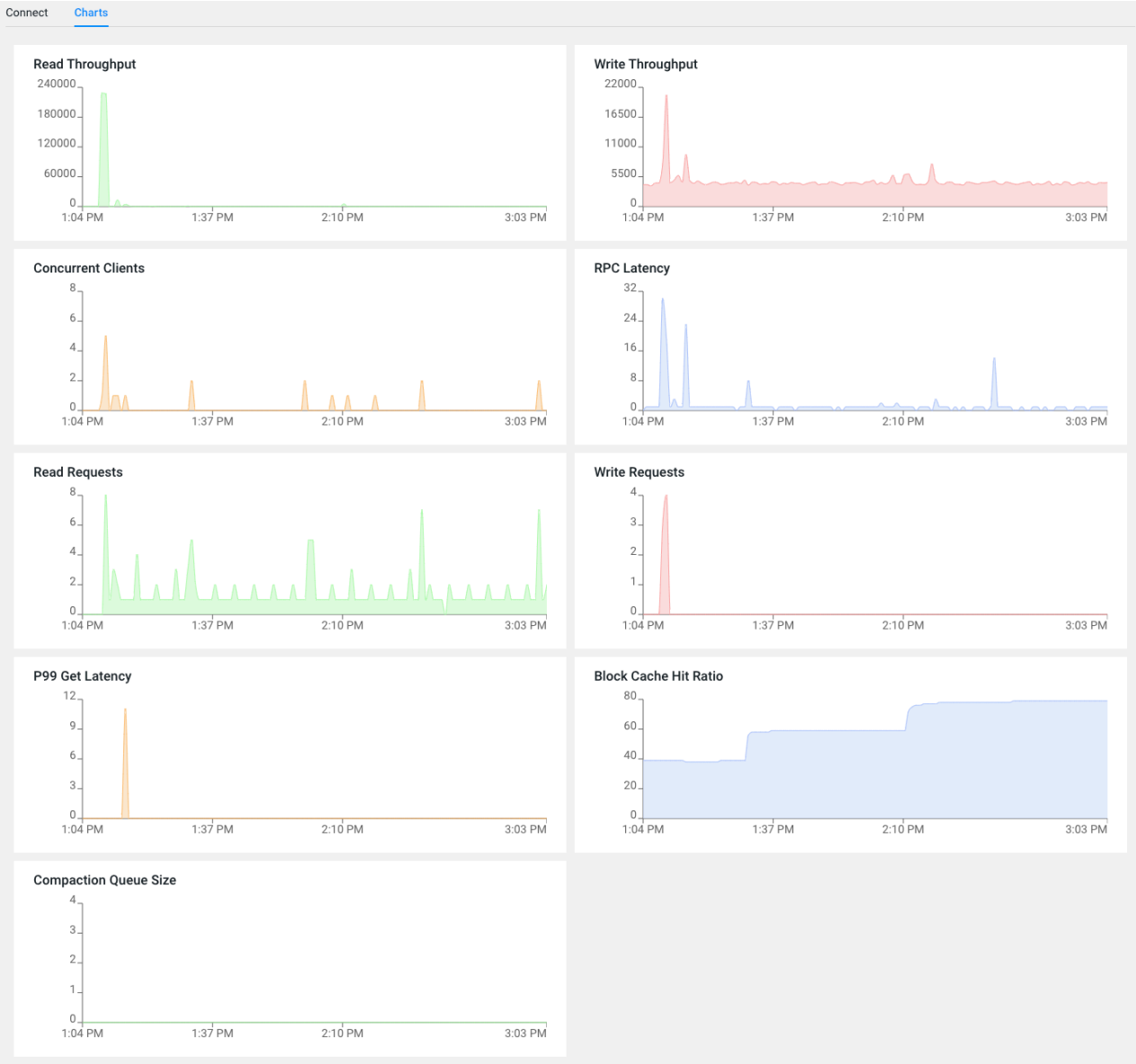

Related Information

Monitor Cloudera Operational Database
Fast autoscaling in Cloudera Operational Database
CDP CLI command
Ephemeral Storage

Cloudera Operational Database integration with RAZ

Cloudera Public Cloud defaults to using cloud storage, which introduces new challenges around managing data access across teams and individual users. The Ranger Authorization Service (RAZ) addresses these challenges, enabling users to have the equivalent fine-grained and audit capabilities in Apache Ranger they used with HDFS files in an on-prem or IaaS deployment.

The Ranger Authorization Service (RAZ) is a fine grained authorization service for cloud storage. As a regular individual user or as an HBase user, you can limit the authorization levels in the cloud storage to a directory level.

Cloudera Operational Database supports RAZ integration from the Runtime version 7.2.11.0. You can grant fine-grained access to directories. Cloudera Operational Database integration with RAZ is supported for AWS and Azure.

Note: Cloudera Operational Database integration with RAZ is not supported for GCP.

Checking RAZ for S3 and ADLS Gen2

To verify if RAZ is enabled for S3 and ADLS Gen2, perform the following steps:

1. Select the environments summary page.
2. Ensure that the Ranger Authorization for S3 is Enabled under Fine-grained access control.
3. For Azure environment, perform the following step:
   - Ensure that the Ranger Authorization for ADLS Gen2 is Enabled under Fine-grained access control

Once RAZ is enabled, all the requests for cloud storage will be routed through RAZ. RAZ checks if it should allow or deny access to objects in the object store based on the defined policies.

To know more about, RAZ requirements for AWS see, AWS requirements for RAZ-enabled AWS environment.

To register a RAZ-enabled AWS environment, see Using Cloudera UI to register RAZ-enabled AWS environment.

To register a RAZ-enabled Azure environment, see Registering a RAZ-enabled Azure environment.

Ephemeral Storage

Cloudera Operational Database now supports NVMe (Non-volatile Memory Express) based cache that significantly improves the performance when you deploy Cloudera Operational Database with object storage.

Accessing cloud storage is generally slower than block storage, as data in such deployments is not local to HBase. Our benchmarks show that Cloudera Operational Database over cloud storage performs, on average, five times slower than Cloudera Operational Database clusters using HDFS on block storage with HDD. To address this penalty, Cloudera Operational Database uses ephemeral storage (NVMe disks available with all major cloud providers) to deploy RegionServers with file base bucket cache on the ephemeral volumes. With all data cached locally in the ephemeral disks, Cloudera Operational Database on cloud storage can achieve performance parity compared to Cloudera Operational Database with HDFS on block storage, but at a lower cost.

Once you enable the entitlement for your tenant, Cloudera Operational Database automatically configures the newly created cluster for this feature. It also allows the auto-scale functionality based on the available cache space. For more information, see *Auto-scaling in public cloud environments*. No additional configuration is needed.

AWS

Cloudera Operational Database supports a 1.6 TB NVMe based cache that significantly improves the performance when you deploy Cloudera Operational Database with S3.

Azure

The NVMe-based cache is now enabled on Azure too. It uses a high-performance AMD instance type with a 2 TB NVMe disk for the worker nodes.

GCP

Cloudera Operational Database supports a 3 TB NVMe-based cache that uses local SSD for ephemeral storage on GCP. This significantly enhances the performance when you deploy Cloudera Operational Database on GCP.

Related Information

Autoscaling in public cloud environments

Cloudera Operational Database on cloud storage with ephemeral cache

HBase Store File Tracking

Cloudera Operational Database supports *Store File Tracking (SFT)* as a separate, plugable layer to handle storefile life cycle, and includes the FILE based built-in implementation that avoids internal file rename or move operations while managing the storefiles.

Cloudera has worked with the Apache HBase project to deliver the first version of this feature through HBASE-26067, and has delivered this feature as a part of Cloudera.

When using S3 for HBase data, Cloudera Operational Database can dynamically scale the number of workers based on the compute resources required, rather than the workers required to host the data in HDFS. To deliver this ability to you in a reasonable timeframe, Cloudera built HBOSS. This feature is the next evolution of HBase using S3 which no longer requires the HBOSS solution. The SFT feature for HBase with S3 prevents unwanted I/O due to renames on S3. With HDFS, a rename is a constant-time operation, but on S3 a rename requires a full copy of the file. Because of this, using S3 doubles the I/O costs for HBase operations like compactions, flushes, and snapshot-based operations. The SFT feature removes the reliance on renames from HBase internal functions that handle user data, which benefits file systems that lack atomic directory rename operation, such as S3.

You can set SFT at the HBase service level using the hbase.store.file-tracker.impl property in hbase-site.xml file or at the table or column family level by configuring TableDescriptor.

Store File Tracking usage

Learn how to use the *Store File Tracking (SFT)* on your Cloudera Operational Database cluster.

When you create a Cloudera Operational Database cluster on an AWS environment using S3 storage, the new cluster automatically has the SFT feature enabled. To use this feature, you can create a new Cloudera Operational Database database using `cdpcli-beta`.

The following is an example of the `create-database` command.

`cdp opdb create-database --environment-name` *MYENVIRONMENT* `--database-name` *MYDATABASE*

Cloudera Operational Database databases are created and the provided feature flag results in all HBase and Phoenix tables using the SFT feature. If you want to use the HDFS storage for your Cloudera Operational Database cluster and use the `--use-hdfs` option while creating the Cloudera Operational Database database, the SFT feature is not enabled because this feature is not relevant when HDFS is being used.

Note: If you try to create a Cloudera Operational Database database for a Cloudera environment on Azure or GCP, the call fails because this feature is only applicable to AWS.

You do not need to recompile or update code in your applications for Cloudera Operational Database databases which have this feature enabled. Cloudera recommends that you update applications to the same version of HBase and Phoenix client libraries to match the version of their Cloudera Operational Database database.

Related Information

Cloudera Operational Database migration

Cloudera Operational Database migration

You can migrate HBase and Phoenix tables from CDH or HDP to Cloudera Operational Database Cloudera Public Cloud where the *Store File Tracking (SFT)* is enabled.

When you are migrating from CDH or HDP to an SFT enabled Cloudera Operational Database Cloudera Public Cloud, ensure that you create an SFT enabled Cloudera Operational Database database on the target cluster. For more details on creating a Cloudera Operational Database database, see Store File Tracking usage.

To migrate from CDH or HDP to an SFT enabled Cloudera Operational Database Cloudera Public Cloud, see HBase Migration through Cloudera Replication Manager.

Even after restoring the snapshots on the SFT enabled Cloudera Operational Database Cloudera Public Cloud, the resulting tables do not load the correct SFT configurations automatically; SFT considers the DEFAULT tracker, which still relies on the temporary directories and renames.

You must convert the DEFAULT tracker to the FILE tracker in each of the restored table configurations by defining the MIGRATION tracker at the table level configurations. Consider the following example to switch the tracker from DEFAULT to FILE in a table configuration.

```
alter 'my-table', CONFIGURATION => {'hbase.store.file-tracker.impl' => 'MIGR
ATION',
'hbase.store.file-tracker.migration.src.impl' => 'DEFAULT',
'hbase.store.file-tracker.migration.dst.impl' => 'FILE'}
```

Once all table regions are online, disable the MIGRATION by setting hbase.store.file-tracker.migration.dst.impl value as hbase.store.file-tracker.impl. In the above example, this can be represented as.

```
alter 'my-table', CONFIGURATION => {'hbase.store.file-tracker.impl' => 'FILE
'}
```

Related Information

Store File Tracking usage

HBase Migration through Cloudera Replication Manager

Restrictions in Store File Tracking

Know the current restrictions and caveats while using the *Store File Tracking (SFT)* feature.

The following are the current restrictions and caveats.

- There is no automated opt-in to this feature for already existing databases. Deploy a new database using the above directions.
- The existing Cloudera Operational Database databases which are upgraded to Cloudera Runtime 7.2.14 do not use the SFT feature. These databases continue to use HBOSS to safely interact with S3.

Multi-AZ deployment on Cloudera Operational Database

Learn about Multi-AZ (Multiple Availability Zones) deployments and how Cloudera Operational Database ensures high availability (HA) and fault tolerance using Multi-AZ deployments.

How Multi-AZ deployment enables HBase high availability

Important: Multi-AZ for Cloudera Operational Database is currently supported only on Amazon Web Services (AWS) and Microsoft Azure environments. However, it is supported on an Azure environment as a technical preview and is considered under development.

To understand the Multi-AZ deployments on Cloudera Operational Database, it is important to know how a cloud provider is configured across the globe and how it provides the availability of services regardless of your location. A cloud provider is made up of a number of regions, which are physical locations around the world. Each region has multiple availability zones (AZs), which act as failure domains, preventing small outages from affecting entire regions. Most regions are home to two to three AZs, with each AZ providing adequate redundancy within a given region. An AZ is represented by a region code followed by a letter identifier; for example, *US-WEST-1A*, *US-WEST-1B*.

However, this redundancy is only applied to the storage layer (S3) and does not exist for virtual machines used to build up your workload. If the AZ where your server instances reside is having an outage, your services cease to function, because the entire compute infrastructure is offline.

A Multi-AZ deployment means that compute infrastructure for HBase's master and region servers are distributed across multiple AZs ensuring that when a single availability zone has an outage, only a portion of Region Servers is impacted and clients automatically switch over to the remaining servers in the available AZs. Similarly, the backup master (assuming the primary master was in the AZ, which is having an outage) automatically takes the role of the failing master because it is deployed in a separate AZ from the primary master server. The entire process is automatic and requires no setup and management from an user. Such a deployment works to ensure an application does not suffer an outage due to the loss of a single AZ.

The following are the results of a failover test performed using a native HDFS and HBase GUI in an HA environment.

- The HDFS NameNode and HBase Master Server on the crashed node reports in red color in Cloudera Manager between 15 and 35 seconds.
- The standby HDFS NameNode assumes an Active state between 30 and 50 seconds as indicated on the native HDFS GUI.
- The standby HBase Master node assumes an Active state between 35 and 55 seconds as indicated on the native HBase GUI.
- The HDFS NameNode and HBase Master Server on the remaining node are indicated as Active in Cloudera Manager between 1 minute and 1 minute 15 seconds.
- The HDFS NameNode and HBase Master Server on the crashed-now-restarted node (now in Standby role) reports in green color in Cloudera Manager between 1 minute and 1 minute 50 seconds.

To understand more on the Multi-AZ support on Cloudera Operational Database, see this blog, *High Availability (Multi-AZ) for Cloudera Operational Database*.

Related Information

High Availability (Multi-AZ) for Cloudera Operational Database

Advantages of Multi-AZ deployments

Understand the single and multiple availability zones that Cloudera Operational Database supports along with the advantages of deploying Multi-AZ.

Cloudera Operational Database in a single availability zone

Without multiple availability zone based deployments, Cloudera Control Plane deploys Cloudera Operational Database and Data Hubs into a single subnet, meaning that each of them has VMs spread across a single availability zone.

Note: On AWS every subnet is related to a single availability zone.

Cloudera Operational Database in a multiple availability zone

If you choose to deploy your Cloudera Operational Database Environment and Data Hubs across multiple availability zones (multi-AZ), each of these components could be spread across two or more availability zones, providing high availability, resilience, and fault tolerance.

Table 1: Key advantages

Advantages	Descriptions
High availability	Supports high availability for your application with automatic database failover. It ensures that you do not lose any data and zero manual intervention.
Resilience	Ensures that the data on your standby database instance is up-to-date with the primary.
Fault tolerance	Enhances availability by deploying a standby instance in a second AZ, and achieve fault tolerance in the event of an AZ or database instance failure.

How to enable Multi-AZ deployment on Cloudera Operational Database

Understand the entitlements that you need to access the Multi-AZ on Cloudera Operational Database and also how you can disable Multi-AZ on your Cloudera Operational Database environment.

To use this feature on an AWS or Azure environment, you must first contact Cloudera Support or your Cloudera account team to enable the following entitlements in Cloudera Public Cloud.

- On an AWS environment, you need the CDP_CB_AWS_NATIVE entitlement.
- On an Azure environment, you need both CDP_CB_AZURE_MULTIAZ and CDP_AZURE_DATABASE_FLEXIBLE_SERVER entitlements.

Once the entitlement is enabled, Cloudera Operational Database supports multiple availability while creating databases, by default.

To disable multiple availability on Cloudera Operational Database, you can provide the --disable-multi-az option while creating a new Cloudera Operational Database database using cdpcli-beta.

For example, `cdp opdb create-database --environment` *MYENVIRONMENT* `--database` *MYDATABASE* `--disable-multi-az`

If you provide this option for a Cloudera environment on GCP, the call fails because this feature is only applicable to AWS and Azure (as a technical preview).

Troubleshooting Multi-AZ deployments

Troubleshoot various scenarios which you might encounter while deploying Multi-AZ on your Cloudera Operational Database environment.

Unable to join the cluster automatically

Condition

HBase Region Servers do not join the cluster automatically after the availability zones are recovered.

Cause

The availability zones and servers are offline for too long and the Master and Region Server processes are stopped.

Solution

Procedure

1. Log in to Cloudera Manager.
2. Restart the Master and Region Server processes.

OMID service fails to recover

Condition

OMID service is failing to recover after the availability zones are down.

Cause

Root cause of this problem is yet to be identified.

Solution

Procedure

1. Log in to Cloudera Manager.
2. Restart the OMID service.

Restrictions in Multi-AZ deployment on Cloudera Operational Database

Know the restrictions while deploying multiple availability zones on your Cloudera Operational Database environment.

The following are the current restrictions:

1. Cloudera Manager currently does not support High Availability, therefore if the AZ containing the gateway node is down, the Cloudera Manager services are unavailable.
2. Cloudera Operational Database auto-scaling does not work during an AZ outage. When an AZ outage occurs, the Cloudera Data Hub identifies that multiple nodes are down and goes into an unhealthy state. Cloudera Operational Database cannot initiate auto-scaling during an unhealthy state.
3. If the node containing the YARN Resource Manager (RM) is down during an AZ outage, the MapReduce jobs related to the YARN services might be impacted, because the RM is not distributed.
4. While performing Multi-AZ failover testing, you might encounter longer failover times because of the Ranger RAZ services, which applies authorization logic in front of cloud storage systems. When this failover does not happen promptly, HBase is unable to perform any operations that require communication with the cloud storage system. Cloudera continues to evaluate this issue in later releases.

Cloudera Operational Database ..

Replicating Data

Date published: 2020-08-14
Date modified: 2022-09-29

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Operational Database data replication.....4
Configuring replication for secure clusters.....4
Configuring replication with Apache HBase in EMR.....6
Using the CldrCopyTable utility to copy data.....7
Using HashTable and SyncTable tools to copy data between Cloudera Operational Database clusters.....8

Cloudera Operational Database data replication

Cloudera Operational Database enables you to replicate your data by making a copy of the data stored in your Cloudera Operational Database instance and store them in other locations to help you in disaster recovery scenarios.

Cloudera provides a replication plugin that allows you to securely enable data replication for Apache HBase data in Cloudera Data Hub and Cloudera Operational Database experience. The replication plugin simplifies SSL authenticated data replication between Apache HBase clusters in the cloud.

Cloudera Operational Database supports the following data replication scenarios:

- CDH or HDP to Cloudera Operational Database using the replication plugin
- Cloudera Private Cloud Base to Cloudera Operational Database using Cloudera Replication Manager
- Cloudera Operational Database to Cloudera Operational Database using Cloudera Replication Manager
- External services like EMR that has Apache HBase using the replication plugin

Related Information

HBase Migration through Cloudera Replication Manager

Configuring replication for secure clusters

You can configure replication between two secure Apache HBase clusters. These clusters can be Cloudera Data Hub clusters or Cloudera Operational Database experience clusters.

Before you begin

Ensure that you have root access on your source HDP, CDH, or any other Apache HBase cluster.

Ensure that you have noted down your workload user name and password. To check your workload user name, navigate to the Cloudera Management Console > User Management > Users, find your user, and then find your Workload User Name. The workload user name is typically srv_[***WORKLOAD USER NAME***].

About this task

You need the replication plugin to enable data replication to secure clusters. A custom replication endpoint allows the Apache HBase clusters to specify a different Simple Authentication and Security Layer (SASL) ticket when establishing the connection with a remote cluster. Use the following instructions to configure secure cluster data replication.

Procedure

1. Create a machine user in your Cloudera environment that has your Cloudera Operational Database instances. This machine user is specific for your replication use case.

   Note down the machine user name. You will need this information later.
2. Add your created machine user with environmentUser role in the destination environment using the Assign environment role.
3. Do FreeIPA sync for the destination environment and wait for the sync to complete.
   The sync can take about 5 to 15 minutes depending upon the number of servers and users in the environment.
4. Verify if the credentials are correct by running the following kinit command on any node of the destination environment.

```
kinit [***WORKLOAD USER NAME***]
```

5. On any environment, generate the keystore using the following command and passing your workload password as a parameter.

```
sudo -u hbase hbase com.cloudera.hbase.security.token.CldrReplicationSec
urityTool -sharedkey cloudera -password [***WORKLOAD PASSWORD***] -keyst
ore localjceks://file/[***PATH***]/[CREDENTIALS.JCEKS]
```

6. Add the generated keystore to HDFS on both the clusters using the following commands

```
kinit -kt /var/run/cloudera-scm-agent/process/`ls -1 /var/run/cloudera-s
cm-agent/process/ | grep -i "namenode\|datanode" | sort -n | tail -1`/hd
fs.keytab hdfs/$(hostname -f)
```

```
hdfs dfs -mkdir /hbase-replication

hdfs dfs -put /[***PATH***]/[CREDENTIALS.JCEKS] /hbase-replication

hdfs dfs -chown -R hbase:hbase /hbase-replication
```

7. Configure the following properties in Cloudera Manager.
   a) Log in to Clouder Manager as an Administrator.
   b) Go to HBase Configuration and search for the property HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml.
   c) Set the values for a Keystore file on HDFS and workload username that matches the CSSO Keystore setup by FreeIPA.

```
<property>
 <name>hbase.security.replication.credential.provider.path</name>
  <value>cdprepjceks://hdfs@[***NAMENODE_HOST***]:[***NAMENODE_PORT***]/
hbase-replication/credentials.jceks</value>
</property>
<property>
 <name>hbase.security.replication.user.name</name>
  <value>srv_[***WORKLOAD USER NAME***]</value>
</property>
```

   d) Restart your cluster for the changes to take effect.
8. Using the HBase shell define replication peer at source cluster, specifying the pluggable replication endpoint implementation and zookeeper_quorum in the following format: zk1:zk2:zk3:2181:/hbase.

```
hbase shell
add_peer '1',
ENDPOINT_CLASSNAME => 'com.cloudera.hbase.replication.CldrHBaseInterClus
terReplicationEndpoint',
CLUSTER_KEY => '[***ZOOKEEPER_QUORUM***]:2181:/hbase'
```

What to do next

1. Set the REPLICATION_SCOPE for the selected table and column family. Note that enable_table_replication is not supported.

```
hbase shell alter 'my-table', {NAME=>'cf', REPLICATION_SCOPE => '1'}
```

2. Use replication plugin validation tool to make sure the setup is working. On one of the source cluster hosts, run the following command:

```
hbase com.cloudera.hbase.replication.CldrReplicationPluginValidator
[***PEER_ID***]
```

Configuring replication with Apache HBase in EMR

You can configure your Cloudera Operational Database experience for data replication with an Amazon EMR cluster with Apache HBase.

Before you begin

- Ensure that you have the replication plugin. Contact your Cloudera account team to get the replication plugin.
- Ensure that all the EC2 instances in the EMR cluster can communicate with Cloudera Operational Database. For example, you can configure this by placing the EMR cluster on the same VPC netwoek and subnets used by the Cloudera Operational Database instance.
- Ensure that your Cloudera Operational Database cluster security group allows inbound TCP connections to ports 16020, 16010, and 2181 from all the EC2 instances in the EMR cluster. You can configure this using the AWS Cloudera Management Console. The port configuration is automatically done if the EMR EC2 instances are configured with the same worker, leader, and controller (also known as master) security groups from Cloudera Operational Database.

Procedure

1. Copy the replication plugin to /usr/lib/hbase/lib/ on all EMR cluster RegionServer. (requires root access);

   Important: This step requires you to have root access to the EMR cluster.

2. Add the keystore generated on the destination Cloudera Operational Database cluster to HDFS using the following command:

```
hdfs dfs -mkdir /hbase-replication

hdfs dfs -put /[***PATH***]/[***CREDENTIALS.JCEKS***] /hbase-replication
hdfs dfs -chown -R hbase:hbase /hbase-replication
```

3. Edit the EMR /etc/hbase/conf.dist/hbase-site.xml file on each RegionServer host to add the following values:

```
<property>
   <name>[***HBASE.SECURITY.REPLICATION.CREDENTIAL.PROVIDER.PATH***]</na
me>
   <value>cdprepjceks://hdfs@[***NAMENODE_HOST***]:[***NAMENODE_PORT***]/
hbase-replication/credentials.jceks</value>
 </property>
 <property>
   <name>[***HBASE.SECURITY.REPLICATION.USER.NAME***]</name>
   <value>srv_[***WORKLOAD USER NAME***]</value>
 </property>
```

4. Restart the EMR RegionServer by stopping the running processes. RegionServers autostarts and now uses the additional JAR files in the classpath.
5. Use _ReplicationSetupTool_ to add the peer. _ReplicationSetupTool_ is a command line tool that enables you to create the replication peer and make necessary configuration to be peer specific.

Example

```
sudo -u hbase hbase org.apache.hadoop.hbase.client.replication.ReplicationSe
tupTool
-clusterKey "ZK-HOST-1,ZK-HOST-2,ZK-HOST-3:2181:/hbase"
-endpointImpl "org.apache.hadoop.hbase.replication.regionserver.CldrHBase
InterClusterReplicationEndpoint" -peerId 1
```

Using the CldrCopyTable utility to copy data

You can use the CldrCopyTable utility to copy data from one Cloudera Operational Database cluster to another. You can use it to bring data in sync for replication.

About this task

CldrCopyTable is Cloudera's version of the upstream CopyTable utility. For more information about the CopyTable utility, see *Use CopyTable*.

The --cldr.cross.domain option of CldrCopyTable is an extension of the Cloudera Operational Database replication plugin that enables you to copy data cross-realm. The replication plugin uses PAM authentication to validate the machine user credentials. Cloudera Operational Database clusters are always provisioned with PAM authentication against the Cloudera environment FreeIPA security domain. Cloudera Single Sign on (CSSO) is used as the machine user credentials (for example, csso_user).

Procedure

1. Ensure that the following properties have the correct values in the hbase-site.xml configuration file of the target cluster:

```
<property>
<name>hbase.security.replication.credential.provider.path</name>
<value>cdprepjceks://hdfs@[***NAMENODE_HOST***]:[***NAMENODE_PORT***]/hba
se-replication/credentials.jceks</value>
</property>

<property>
<name>hbase.security.replication.user.name</name>
<value>srv_[***WORKLOAD USER NAME***]</value>
</property>
```

2. Ensure that the sourse cluster can communicate with the target cluster:
   a) Get the ZooKeeper quorum address of the target cluster.
   b) Set the address as an environment parameter in the source cluster.
   c) Set a subnet that allows connection from the source cluster.

      For exmple by enabling the port 2181 for the ZooKeeper client.
3. Issue the CldrCopyTable command from the source cluster to write to the target cluster.

   Based on your target cluster setup you have to use either the --cldr.cross.domain or the --cldr.unsecure.peer option.

   For Secure target cluster

   Use the --cldr.cross.domain option:

```
hbase org.apache.hadoop.hbase.mapreduce.CldrCopyTable --cldr
.cross.domain --peer.adr=[***ZOOKEEPER QUORUM***]:[***ZOOKEEPER
 PORT***]:[***ZOOKEEPER ROOT FOR HBASE***] --new.name="[***NEW TABLE
 NAME***]" "[***SOURCE TABLE NAME***]"
```

   For Unsecure target cluster

   Use the --cldr.unsecure.peer option:

```
hbase.org.apache.hadoop.hbase.mapreduce.CldrCopyTable --cldr
.unsecure.peer --peer.adr=[***ZOOKEEPER QUORUM***]:[***ZOOKEEPER
```

```
PORT***]:[***ZOOKEEPER ROOT FOR HBASE***] --new.name="[***NEW TABLE
NAME***]" "[***SOURCE TABLE NAME***]"
```

4. Once the job is finished, check the target cluster and ensure that the copy was successful.

Related Information

Use CopyTable

Using HashTable and SyncTable tools to copy data between Cloudera Operational Database clusters

You can use the HashTable and SyncTable tools to copy data from one Cloudera Operational Database cluster to another. You can use these tools to synchronize data prior to replication.

About this task

You can use the HashTable and SyncTable CLI tools as a one way synchronization method for data in Cloudera Operational Database clusters. The CldrSyncTable job is an extension of the upstream SyncTable tool. For more information about the HashTable and SyncTable tools, see *Use HashTable and SyncTable tool*.

When you use these tools, ensure that you place the HashTable output directory and the source table at the same location where the CLI exists. This means that you cannot set the sourcezkcluster and the sourcehashdir properties to a remote cluster that the command-line executor cannot authenticate.

Before you begin

- Ensure that all RegionServers and DataNodes on the source cluster are accessible by NodeManagers on the target cluster where SyncTable job tasks are running.
- In case of secured clusters, users on the target cluster who execute the SyncTable job must be able to do the following on the HDFS and HBase services of the source cluster:
  - Authenticate: for example, using the centralized authentication or cross-realm setup.
  - Be authorized: having at least read permission.
- Ensure that the target table is created and enabled on the target cluster.

Procedure

1. Ensure that the following properties have the correct values in the hbase-site.xml configuration file of the target cluster:

```
<property>
   <name>hbase.security.replication.credential.provider.path</name>
   <value>cdprepjceks://hdfs@[***NAMENODE_HOST***]:[***NAMENODE_PORT***]/
hbase-replication/credentials.jceks</value>
 </property>
 <property>
   <name>hbase.security.replication.user.name</name>
   <value>srv_[***WORKLOAD USER NAME***]</value>
 </property>
```

2. Ensure that the source cluster can communicate with the target cluster.
   a) Get the ZooKeeper quorum address of the target cluster.
   b) Set a subnet that allows connection from the source cluster.

      For example, enabling the port 2181 for the ZooKeeper client.
3. Run the HashTable job from the source cluster to generate a hash of a table on the source cluster:

```
hbase org.apache.hadoop.hbase.mapreduce.HashTable
```

```
[***TABLE NAME***] [***HASH OUTPUT PATH***]
hdfs dfs -ls -R [***HASH OUTPUT PATH***]
```

4. Run the CldrSyncTable job from the source cluster to compare the generated hashes:

   Use the --cldr.cross.domain or the --cldr.unsecure.peer option:

   For Secure target cluster

   Use the --cldr.cross.domain option when running the CldrSyncTable job in a secure cluster:

```
hbase org.apache.hadoop.hbase.mapreduce.CldrSyncTable
--cldr.cross.domain --targetzkcluster=[***TARGET ZOOKEEPER
QUORUM***]:[***TARGET ZOOKEEPER PORT***]:[***TARGET ZOOKEEPER ROOT FOR
 HBASE***]
[***HASH OUTPUT PATH***] [***SOURCE TABLE NAME***]
[***TARGET TABLE NAME ON THE TARGET CLUSTER***]
```

   For Unsecure target cluster

   Use the --cldr.unsecure.peer option when running the CdrSyncTable job in an unsecure cluster::

```
hbase org.apache.hadoop.hbase.mapreduce.CldrSyncTable
--cldr.unsecure.peer
 --targetzkcluster=[***TARGET ZOOKEEPER QUORUM***]:[***TARGET
ZOOKEEPER PORT***]:[***TARGET ZOOKEEPER ROOT FOR HBASE***]
[***HASH OUTPUT PATH***] [***SOURCE TABLE NAME***]
[***TARGET TABLE NAME ON THE TARGET CLUSTER***]
```

5. After the job is completed, check the data copied on the target cluster.

```
scan '[***TARGET TABLE NAME***]'
```

Related Information

Use HashTable and SyncTable tool

Cloudera Operational Database on High Performance Storage

Date published: 2020-08-14
Date modified: 2023-01-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Overview 4

Using high performance storage 5

Installing CDP CLI beta 5

Autoscaling 5

Overview

Cloudera Operational Database supports both high performance storage and Amazon S3 or Azure Data Lake Gen2 storage through the abfs connector as a storage layer. While the default option is cloud storage, Cloudera Operational Database also provides you an option to choose high performance storage.

You can use high performance storage either in an on-premises or a cloud environment to maintain low-latency and high-throughput for your application. If your applications are latency sensitive, Cloudera recommends that you use high performance storage for your Cloudera Operational Database databases. The average read and write latency for Cloudera Operational Database on a high performance storage is 500% better than the Cloudera Operational Database that uses S3 storage for all the workloads.

Note: Cloudera Operational Database on GCP environment supports only high performance storage.

HDFS volume type

Hadoop Distributed File System (HDFS) is a Java-based file system for storing large volumes of data. Designed to span large clusters of commodity servers, HDFS provides scalable and reliable data storage.

An HDFS cluster contains the following main components: a NameNode and DataNodes. The NameNode manages the cluster metadata that includes file and directory structures, permissions, modifications, and disk space quotas. The file content is split into multiple data blocks, with each block replicated at multiple DataNodes.

For more information, see *HDFS Overview*.

While creating a Cloudera Operational Database database, you have the following two options for storing your data when you select HDFS as the storage type.

- HDD (Hard Disk Drives): HDD is a cost-efficient option if you want to access the data frequently. This is a preferred storage option for applications that are not very sensitive to latency.
- SSD (Solid State Drives): SSDs provide more storage volume, speed, and efficiency; however, this storage option could be expensive compared to HDD.

For more information on the difference between SSD and HDD, see Amazon documentation.

Worker nodes

Worker nodes store the data on your Cloudera Operational Database cluster. The general responsibilities of the worker nodes on your Cloudera Operational Database cluster include processing the data stored in the cluster and managing the network traffic. The worker nodes also ensure smooth operations between the applications across multiple Cloudera Operational Database clusters.

When you create a Cloudera Operational Database cluster, the minimum and maximum number of worker nodes vary for different storage types.

- Micro duty: Minimum node count: 1. Maximum node count: 5.
- Light duty: Minimum node count: 3. Maximum node count: 100.
- Heavy duty: Minimum node count: 3. Maximum node count: 800.

Related Information

HDFS Overview

Using high performance storage

You can create a Cloudera Operational Database database on high performance storage through CDP CLI beta version.

Procedure

Provide the following API request to create a Cloudera Operational Database database with high performance storage from CDP CLI beta version 0.9.40.

```
$ cdp opdb create-database --environment-name foo --database-name bar --use-hdfs
```

```
--use-hdfs | --no-use-hdfs (boolean) - Controls whether hbase root be deployed on hdfs
```

Related Information

Installing CDP CLI beta

Installing CDP CLI beta

To use Cloudera Operational Database on high performance storage through CDP CLI, you must download and install CDP CLI beta version 0.9.40.

Before you begin

Do not install both the regular and beta CLIs in the same Python environment, as they use the same entry points and will therefore conflict. Either use a separate virtual environment or uninstall the cdpcli first before installing cdpcli-beta.

Procedure

1. Install Python if needed, as described in CDP CLI documentation for your platform.

   Do not install CDP CLI. Instead, proceed to step 2.
2. Run the following to install CDP CLI beta:

   ```
   pip3 install cdpcli-beta
   ```
3. Configure access keys as described in Generating an API access key

Related Information

Using high performance storage

Autoscaling

Cloudera Operational Database with cloud storage type uses high performance storage to store Write-Ahead Log (WALs); however Cloudera Operational Database with high performance storage type uses both WALs and HFiles.

Cloudera Operational Database provides autoscaling for both the storage types. Your database automatically autoscales whenever the high performance storage reaches 60% of its buffer capacity to accommodate the growth in the data.

Cloudera Operational Database ..

Integrating Data

Date published: 2023-12-31
Date modified: 2024-04-27

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configure data engineering Spark to use with Cloudera Operational Database **4**

Configure Data Engineering Cloudera Data Hub to use with Cloudera Operational Database **6**

Running Phoenix and HBase Spark Applications using Cloudera Data Engineering **7**

Configure data engineering Spark to use with Cloudera Operational Database

You can configure Spark in your Data Engineering cluster to interact with the Cloudera Operational Database. You can use this integration to READ and WRITE to Cloudera Operational Database from Spark on Cloudera Data Engineering using the spark-hbase connector.

About this task

If you want to leverage Phoenix instead of HBase, see *Cloudera Operational Database-Cloudera Data Engineering using Phoenix*.

Before you begin

- Cloudera Operational Database is already provisioned and the database is created. For ore information, see *Onboarding Cloudera Operational Database users*.

  For this example, let us assume *TEST-COD* as the database name.
- Cloudera Data Engineering is already provisioned and the virtual cluster is already created. For more information, see *Cloudera Data Engineering service*.

Procedure

1. Download the Cloudera Operational Database client configurations.

   For the Spark in Cloudera Data Engineering to connect to Cloudera Operational Database, it requires the hbase-site.xml configuration of the Cloudera Operational Database cluster. Refer to the following steps.

   a) Go to the Cloudera Operational Database UI and click on the *TEST-COD* database.
   b) Go to the Connect HBase tab of the Cloudera Operational Database database, and copy the command under the HBase Client Configuration URL field.

   ```
   curl -f -o "hbase-config.zip" -u "<YOUR WORKLOAD USERNAME>" "https://cod
   --4wfxojpfxmwg-gateway.XXXXXXXXX.cloudera.site/clouderamanager/api/v41/c
   lusters/cod--4wfxojpfxmwg/services/hbase/clientConfig"
   ```

   - Ensure to provide the workload password for the `curl` command.
   - Explore the downloaded zip file to obtain the hbase-site.xml file.
2. Create the HBase table.

   Create a new HBase table inside the Cloudera Operational Database database using the Hue link.

   a) Go to the Cloudera Operational Database UI and click on the *TEST-COD* database.
   b) Click on the Hue link under the SQL EDITOR field.
   c) On the Hue UI, click the HBase menu item on the left navigation panel. Click New Table.
   d) Choose a table name and column families, and click on the Submit button. For example, let us consider the table name *TESTTABLE* and a single column family *TESTCF*.

3. Configure the job using Cloudera Data Engineering CLI.
   a) Configure Cloudera Data Engineering CLI to point to the virtual cluster created in the previous step. For more details, see *Configuring the CLI client*.
   b) Create resources using the following command.

```
cde resource create --name cod-spark-resource
```

   c) Upload hbase-site.xml.

```
cde resource upload --name cod-spark-resource --local-path /your/path/to
/hbase-site.xml --resource-path conf/hbase-site.xml
```

   d) Upload the demo app jar that was built earlier.

```
cde resource upload --name cod-spark-resource --local-path /path/to/your
/spark-hbase-project.jar --resource-path spark-hbase-project.jar
```

   e) Create the Cloudera Data Engineering job using a JSON definition.

```
{
"mounts": [
{
"resourceName": "cod-spark-resource"
}
],
"name": "my-cod-spark-job",
"spark": {
"className": "<YOUR MAIN CLASS>",
"conf": {
"spark.executor.extraClassPath": "/app/mount/conf",
"spark.driver.extraClassPath": "/app/mount/conf"
},
"args": [ "<YOUR ARGS IF ANY>"],
"driverCores": 1,
"driverMemory": "1g",
"executorCores": 1,
"executorMemory": "1g",
"file": "spark-hbase-project.jar",
"pyFiles": [],
"files": ["conf/hbase-site.xml"],
"numExecutors": 4
}
}
```

   f) Import the job using the following command, assuming that the above JSON is saved as *MY-JOB-DEFINITION.JSON*.

```
cde job import --file my-job-definition.json
```

What to do next

The `spark.driver.extraClassPath` and `spark.executor.extraClassPath` inside the job definition points to the same path which is used to upload the hbase-site.xml into the Cloudera Data Engineering resource.

This way the hbase-site.xml is automatically loaded from the classpath and you do not need to refer to it explicitly in your Spark code. You can define as follows.

```
val conf = HBaseConfiguration.create()
val hbaseContext = new HBaseContext(spark.sparkContext, conf)
```

Related Information

Cloudera Operational Database-Cloudera Data Engineering using Phoenix
Onboarding Cloudera Operational Database users
Cloudera Data Engineering service
Configuring the CLI client

Configure Data Engineering Cloudera Data Hub to use with Cloudera Operational Database

You can configure Hive in your Cloudera Data Engineering Cloudera Data Hub cluster to interact with the Cloudera Operational Database. You can use this integration to create and modify HBase tables using Hive. You can also READ and WRITE to existing HBase tables.

Before you begin

Ensure that you have a Cloudera Operational Database database instance and a Data Engineering template-based Cloudera Data Hub cluster in the same Cloudera environment.

Procedure

1. Create a Data Engineering Datahub cluster in the same Cloudera environment on which your Cloudera Operational Database is launched.
2. Download the HBase client configuration zip archive file from the Cloudera Operational Database client connectivity page. For more information see *Client connectivity information*. Then, extract hbase-site.xml from the zip file.
3. On each node in the Data Engineering Cloudera Data Hub cluster that runs HiveServer2, run the `mkdir /etc/hbase/cod-conf` command, and copy the hbase-site.xml file from Step 2 into /etc/hbase/cod-conf folder.

   Do not copy all the files from client configuration zip archive. Only copy the hbase-site.xml file.

   a) Get the value of ssl.client.truststore.password from the Data Engineering DataHub cluster master ssl-client.xml file (/etc/hadoop/conf.cloudera.hdfs/ssl-client.xml) and update the password for hbase.zookeeper.property.ssl.trustStore.password property in the copied hbase-site.xml file.
   b) Create a symbolic link between /opt/cloudera/parcels/CDH/lib/hbase/conf and /etc/hbase/cod-conf in the Data Engineering Datahub cluster using the following command.

      ```
      sudo ln -s /etc/hbase/cod-conf /opt/cloudera/parcels/CDH/lib/hbase/conf
      ```
4. In Cloudera Manager for the Cloudera Data Engineering Cloudera Data Hub cluster, in the Hive-on-Tez service, set the HiveServer2 Environment Advanced Configuration Snippet(hive_hs2_env_safety_valve): HADOOP_CLASSPATH to /etc/hbase/cod-conf.

   a) Click Save Configurations.
   b) Restart the Hive-on-Tez service.
5. Go to the Ranger service in the Data Lake.

   a) Find HBase Ranger policy cod_[***CLOUDERA OPERATIONAL DATABASE DATABASE NAME***]_hbase. For example, if your Cloudera Operational Database database name is CODDB, you must modify the Ranger policy cod_CODDB_hbase.
   b) Add the user hive to the existing policy. For example, all - table, column-family, column.
   c) Wait for a few minutes because Ranger policy sync is not immediate to all resources.
6. Interact with Hive using the hive command from a node in the Data Engineering Cloudera Data Hub.

   ```
   $ hive
   hive> create table test(key int, value string) stored by 'org.apache.hado
   op.hive.hbase.HBaseStorageHandler' with serdeproperties ("hbase.columns.
   mapping" = ":key,f1:val");
   ```

```
hive> insert into test values(1, 'a');
hive> select * from test;
```

7. Validate data is present in HBase as well. You can also run this from the Data Engineering Cloudera Data Hub node: the hbase executable is present to invoke. Otherwise, set up the HBase client tarball from Cloudera Operational Database client connectivity.

```
$ HBASE_CONF_DIR=/etc/hbase/cod-conf hbase shell
hbase> scan 'test'
```

Related Information

Client connectivity information

Running Phoenix and HBase Spark Applications using Cloudera Data Engineering

Cloudera Data Engineering supports running Spark applications in the Cloudera Public Cloud. You can use Cloudera Data Engineering to run Phoenix and HBase Spark applications against Cloudera Operational Database.

Before you begin

- Set your Cloudera workload password. For more information, see *Setting the workload password*.
- Synchronize users from the User Management Service in the Cloudera Control Plane into the environment in which your Cloudera Operational Database is running.
- Ensure Cloudera Data Engineering service is enabled and a virtual cluster is created in the Data Engineering Experience. For more information, see *Creating virtual clusters* and *Enabling a Cloudera Data Engineering service*.

Procedure

1. Set HBase and Phoenix versions in your Maven project.
   a) Use the `describe-client-connectivity` command for the HBase and Phoenix version information. The following code snippet shows fetching the database connectivity information and parsing the required HBase and Phoenix information to build your application.

```
echo "HBase version"
cdp opdb describe-client-connectivity --database-name my-database --env
ironment-name my-env | jq ".connectors[] | select(.name == \"hbase\") |
.version"
echo "Phoenix Connector Version"
cdp opdb describe-client-connectivity --database-name my-database --e
nvironment-name my-env | jq ".connectors[] | select(.name == \"phoenix-t
hick-jdbc\") | .version"
```

```
HBase Version
2.4.6.7.2.14.0-133
Phoenix Spark Version
"6.0.0.7.2.14.0-133"
```

   b) Update the HBase and Phoenix connector versions in our Maven project or configuration.

```
<properties>
...
    <phoenix.connector.version>6.0.0.7.2.14.0-133</phoenix.connector.ve
rsion>
    <hbase.version>2.4.6.7.2.14.0-133</hbase.version>
```

```
...
</properties>
```

2. Download hbase-site.xml and hbase-omid-client-config.yml configuration files.
   a) Use the `describe-client-connectivity` command to determine the client configuration URL.

```
cdp opdb describe-client-connectivity --database-name spark-connector --
environment-name cod-7213 | jq ".connectors[] | select(.name == \"hbase\
") | .configuration.clientConfigurationDetails[] | select(.name == \"HBA
SE\") | .url "
```

```
"https://cod--XXXXXX-gateway0..xcu2-8y8x.dev.cldr.work/clouderamanager/a
pi/v41/clusters/XXXXX/services/hbase/clientConfig"
```

   b) Use the URL gathered from the previous command and run the `curl` command to download the HBase configurations.

```
curl -f -o "hbase-config.zip" -u "<csso_user>" "https://cod--XXXXXX-gate
way0.cod-7213....xcu2-8y8x.dev.cldr.work/clouderamanager/api/v41/cluster
s/cod--XXXX/services/hbase/clientConfig"
```

   c) Unzip hbase-config.zip and copy the hbase-site.xml and hbase-omid-client-config.yml to src/main/resources path in the Maven project.

```
unzip hbase-conf.zip
cp hbase-conf/hbase-site.xml <path to src/main/resources>
cp hbase-conf/hbase-omid-client-config.yml <path to src/main/resources>
```

3. Build the project.

```
$ mvn package
```

4. Create a Cloudera Data Engineering job.
   a) Configure Cloudera Data Engineering CLI to point to the virtual cluster. For more information, see *Downloading the Cloudera Data Engineering command line interface*.
   b) Create a resource using the following command.

```
cde resource create --name phoenix-spark-app-resource
```

   c) Upload the required jars which you downloaded while building the project.

```
cde resource upload --name spark-app-resource --local-path ./target/conn
ector-libs/hbase-shaded-mapreduce-2.4.6.7.2.14.0-133.jar --resource-path
 hbase-shaded-mapreduce-2.4.6.7.2.14.0-133.jar
cde resource upload --name spark-app-resource --local-path ./target/c
onnector-libs/opentelemetry-api-0.12.0.jar --resource-path opentelemetry
-api-0.12.0.jar
cde resource upload --name spark-app-resource --local-path ./target/conn
ector-libs/opentelemetry-context-0.12.0.jar --resource-path opentelemetr
y-context-0.12.0.jar
cde resource upload --name spark-app-resource --local-path ./target/con
nector-libs/phoenix5-spark-shaded-6.0.0.7.2.14.0-133.jar --resource-path
 phoenix5-spark-shaded-6.0.0.7.2.14.0-133.jar
```

   d) Upload the Spark application app jar that you had built earlier.

```
cde resource upload --name spark-app-resource --local-path ./target/phoe
nix-spark-transactions-0.1.0.jar --resource-path phoenix-spark-transacti
ons-0.1.0.jar
```

e) Replace HBase, Phoenix, and Phoenix Spark connector versions in the spark-job.json as shown in the following sample and create a Cloudera Data Engineering job using the following JSON and import commands.

```
{
   "mounts":[
      {
         "resourceName":"phoenix-spark-app-resource"
      }
   ],
   "name":"phoenix-spark-app",
   "spark":{
      "className":"com.cloudera.cod.examples.spark.SparkApp",
      "args":[
         "{{ phoenix_jdbc_url }}"
      ],
      "driverCores":1,
      "driverMemory":"1g",
      "executorCores":1,
      "executorMemory":"1g",
      "file":"phoenix-spark-transactions-0.1.0.jar",
      "pyFiles":[

      ],
      "files":[
        "hbase-shaded-mapreduce-2.4.6.7.2.14.0-133.jar",
        "opentelemetry-api-0.12.0.jar",
        "opentelemetry-context-0.12.0.jar"
        "phoenix5-spark-shaded-6.0.0.7.2.14.0-133.jar",
      ],
      "numExecutors":4
   }
}
cde job import --file spark-job.json
```

5. Run the project.

a) Use the `describe-client-connectivity` command to determine the base JDBC URL to pass.

```
cdp opdb describe-client-connectivity --database-name my-database --envi
ronment-name my-env | jq ".connectors[] | select(.name == \"phoenix-thic
k-jdbc\") | .configuration.jdbcUrl"
```

b) Run the job by passing the JDBC URL obtained from the previous command, as an argument to the job.

```
cde job run --name phoenix-spark-app --variable phoenix_jdbc_url=<phoeni
x_jdbc_url>
```

Related Information

Enabling a Cloudera Data Engineering service
Setting the workload password
Creating virtual clusters
Downloading the Cloudera Data Engineering command line interface

Cloudera Operational Database ..

Managing Your Database

Date published: 2020-08-14
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a database using Cloudera Operational Database..... 4

Configuring a database..... 6
Configuring worker nodes..... 6
Configuring edge nodes..... 6

Upgrading a database using Cloudera Operational Database..... 7

Changing database status using Cloudera Operational Database..... 8

Launching hbase shell..... 9

Configuring strong meta servers..... 11

Importing and restoring data into Cloudera Operational Database database..... 13

Backing up a Cloudera Operational Database table..... 14

HBase REST server scaling in Cloudera Operational Database..... 17
Configuring the HBase REST server scaling..... 17
Limitations..... 18

Ensure HBase populates pre-existing object storage data..... 19

Managing custom images in Cloudera Operational Database..... 20
Creating a database using a custom image..... 20
Upgrading a database using a custom image..... 21
Switching image catalogs in Cloudera Operational Database..... 21

Enabling HBase region canary..... 22

Creating a database using Cloudera Operational Database

You can create an operational database in your registered Cloudera environment using the Cloudera Operational Database.

About this task

Required role: You must be logged into the Cloudera Operational Database as an ODAdmin.

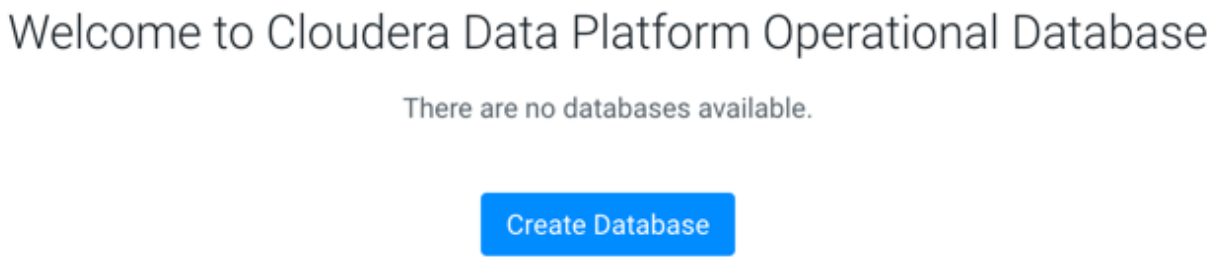

Before you begin

- Understand the Cloudera environment and user management. For more information, see *User Management* and *Cloudera Environments* topics.
- Set up an environment that gives you credential and cloud storage. For more information, see *Before you create an operational database cluster*.
- Ensure that you are authorized to create a database.

Procedure

1. In the Cloudera Operational Database web interface, click Create Database.
2. Specify the location of the database where you want to store it.
   a) Provide a name for the database in the Database Name field.
   b) Select the Cloudera environment from the list in which you want to associate the database.
   c) Click Next.

      If an environment does not exist, you can create one by clicking Create New Environment.

      For more information, see *Register your first environment*.
3. Commission your database by defining a scale for your database using a predefined Data Lake template.

   The template helps you to structure your database automatically thereby saving your time and cost. Cloudera Operational Database creates the predefined number of LITE or HEAVY gateway and master nodes, a set of worker nodes, and also adds additional functionalists into the new database. In case you need to modify the default number of nodes defined in the template, you can do so after the database creation.

   The available templates are Micro Duty, Light Duty, and Heavy Duty. By default, Light Duty is selected.

   You can create a small database using the Micro Duty template, which consists of one Gateway node and one Worker node. In a Micro database, the Gateway node carries out the processes involved in the Master or Leader nodes. You can consider using a Micro cluster for your testing and development purposes.

4. Configure your database by selecting the storage type as Cloud Storage with Caching, Cloud Storage with Caching and Data Tiering, Cloud Storage, or HDFS.
   - The storage type Cloud Storage with Caching is equivalent to using --storage-type CLOUD_WITH_EPHEMERAL option on CDP CLI while creating an operational database.
   - The storage type Cloud Storage with Caching and Data Tiering resembles cloud storage with time-based priority caching, where data within a specified time range is given a higher priority. In contrast, older data are likely to get evicted. For more information on this storage type, see *HBase Time-based Data Tiering using Persistent BucketCache*.

     You must have the COD_DATATIERING entitlement to be able to use this storage type.
   - The storage type Cloud Storage, which resembles block storage, is equivalent to using --storage-type CLOUD option on CDP CLI while creating an operational database.
   - The storage type HDFS is equivalent to using --storage-type HDFS option on CDP CLI while creating an operational database.

   By default, Cloud Storage with Caching is selected.
5. Check or update the settings for your database.
   a) Check all the default settings for your database under the Default tab.
   b) Go to the Advanced tab if you need to modify any of the default values.
      - The HDFS Volume Type option appears under the Advanced tab only if you select HDFS as the storage type in the Configuration step.
      - If you disable the Autoscaling option using the Advanced tab, the Worker Nodes and Compute Nodes options are hidden. Instead, a Node Count option appears.

      The minimum and maximum number of worker nodes vary for different storage types.
      - Micro duty: Minimum node count: 1. Maximum node count: 5.
      - Light duty: Minimum node count: 3. Maximum node count: 100.
      - Heavy duty: Minimum node count: 3. Maximum node count: 800.
6. Review the details before creating the database.

   Click Show CLI Command to get the complete command details corresponding to your settings. You can use it to create the database using CDP CLI.

   Alternatively, you can use the following sample command to create the database using CDP CLI.

```
cdp opdb create-database --environment-name cod-7218 --database-name tes
t --scale-type LIGHT --storage-type CLOUD_WITH_EPHEMERAL --auto-scaling-
parameters '{"minWorkersForDatabase":5, "maxWorkersForDatabase":100}' --
num-edge-nodes 0 --java-version 8
```

7. Click Create Database.

Results

An information page is displayed that shows the status of the database. Your new database is ready to be used once its status becomes Available.

Note: Your database starts with a fixed size; however, it scales up and down as the workload applied to the database changes. For more information, see the *Auto Scaling* topic.

Related Information

Cloudera Operational Database edge node overview
Cloudera Operational Database User Management
Cloudera Environments
Before you create an operational database cluster
Register your first environment
Cloudera Operational Database Auto Scaling

CDP CLI BETA command reference GitHub repository
HBase Time-based Data Tiering using Persistent BucketCache

Configuring a database

Learn how to configure the properties of an operational database in your Cloudera Public Cloud environment.

After you create an operational database with the specified worker and edge nodes, if you want to change the node counts, you can do so for the existing database, without needing to create a new one.

Configuring worker nodes

Know how to configure an operational database's autoscaling and worker node properties in your Cloudera Public Cloud environment.

About this task

Required role: You must be logged into the Cloudera Operational Database as an ODAdmin.

Procedure

1. In the Cloudera Operational Database web interface, select the Databases tab.
2. Find the database you want to configure.
3. In the row of the selected database, click (Actions) and choose Configure Database.
4. Modify the settings for Autoscaling and Worker Nodes.

   Consider these points while setting the worker nodes.

   - Ensure that the number of maximum worker nodes is greater than or equal to the minimum number of worker nodes; otherwise, autoscaling is disabled.

     If autoscaling is enabled, the operational database automatically adjusts the maximum and minimum number of worker nodes; however, you can set these node counts to remain in the cluster.
   - Non-micro clusters must have at least three worker nodes; while micro clusters can have one minimum worker node.
   - The supported value range for different clusters is as follows:

     Micro: 1 minimum, 5 maximum

     Light: 3 minimum, 100 maximum

     Heavy: 3 minimum, 800 maximum
5. Click Configure Database.

Results

The new settings are reflected in the existing database immediately after the modification.

Configuring edge nodes

Know how to configure the edge nodes of an existing operational database in your Cloudera Public Cloud environment.

About this task

Required role: You must be logged into the Cloudera Operational Database as an ODAdmin.

Procedure

1. In the Cloudera Operational Database web interface, select the Databases tab.
2. Find the database you want to configure.
3. In the row of the selected database, click (Actions) and choose Configure Edge Nodes.
4. You can go to each tab Add or Delete to add or delete a specific number of edge nodes.
   You can add a maximum of 20 edge nodes to the cluster.
5. Click Add Edge Nodes or Delete Edge Nodes accordingly.

Results

The new settings are reflected in the existing database immediately after the modification.

Upgrading a database using Cloudera Operational Database

Learn how to upgrade your existing Cloudera Operational Database in the Cloudera environment.

About this task

- You need to use the CDP Beta CLI and run the upgrade-database command to upgrade your database.
- Required role: You must be logged into the Cloudera Operational Database as an ODAdmin.
- To use Cloudera Operational Database on a GCP environment, you must do it through CDP CLI with --use-hdfs flag.

Important: Before upgrading the Runtime version in your existing Cloudera Operational Database clusters to 7.2.16 or higher versions, you might need to perform some additional steps before upgrading. For more information, see *Data Lake upgrade validations for Python dependency* under the Cloudera Management Console what's new topic and *Upgrading Data Hubs*.

Before you begin

- Before you perform the Cloudera Operational Database upgrade: In the Cloudera Manager properties, increase the omid_max_heap_size property for the Omid service to at least 3GB before starting the upgrade from 7.2.9/7.2.10 to 7.2.11:

- Understand Cloudera environment and user management. For more information, see *User Management* and *Cloudera Environments* topics.
- Ensure you are authorized to upgrade a database.
- You must download and install the latest CDP CLI beta version.
- Cloudera recommends that you run the `prepare-upgrade-database` command before upgrading the cluster. This command performs all validations and downloads all required parcels for the upgrade operation. For more information, see *prepare-upgrade-database*.

Procedure

1. Log in to the CDP CLI tool.
2. Run the following command to prepare for the upgrade. This step is optional.

   To reduce the chances of upgrade failures, Cloudera Operational Database supports a preparation phase for the runtime upgrades. During this phase, Cloudera Operational Database runs all the validations and downloads the required parcels for the machines.

   `cdp opdb prepare-upgrade-database --environment` <*ENVIRONMENT-NAME*> `--database` <*DATABASE-NAME*> `--runtime` <*RUNTIME-VERSION*> [OR `--image-id` <*IMAGE-ID*>]

Option	Description
--environment (string)	The name or CRN of the environment.
--database <value>	The name or CRN of the database.
--runtime <value>	The runtime version to upgrade to.
--image-id <imageId>	The image ID to upgrade to.

3. Run the following command to upgrade the database.

   This command upgrades an operational database in an environment to a given Runtime:

   `cdp opdb upgrade-database --environment` <*ENVIRONMENT-NAME*> `--database` <*DATABASE-NAME*> `--runtime` <*RUNTIME-VERSION*> `[--os-upgrade-only | --no-os-upgrade only]`

Option	Description
--environment (string)	The name or CRN of the environment.
--database <value>	The name or CRN of the database.
--runtime <value>	The runtime version to upgrade to.
[--os-upgrade-only | --no-os-upgrade-only]	Controls whether to perform only an Operating System upgrade.

Related Information

Data Lake upgrade validations for Python dependency
Service pack upgrades
Cloudera Operational Database User Management
Cloudera Environments
prepare-upgrade-database
CDP CLI BETA command reference GitHub repository

Changing database status using Cloudera Operational Database

After you create an operational database, you can start, stop, or drop the database using the Cloudera Operational Database user interface.

About this task

Required role: You must be logged into the Cloudera Operational Database as an ODAdmin

Procedure

1. In the Cloudera Operational Database web interface, select the Databases tab.
2. Find the database you want to manage.
3. In the row of the selected database, click Actions and select one of the following actions:
   - Start Database - Starting the database. When you create a new database, it is initially started by default.
   - Stop Database - Stopping the database and the underlying instances.
   - Drop Database - Dropping the database and deleting all the data that is stored in that database. However, your Cloudera Shared Data Experience Data Lake continues to exist and you can create another database using the same Data Lake.

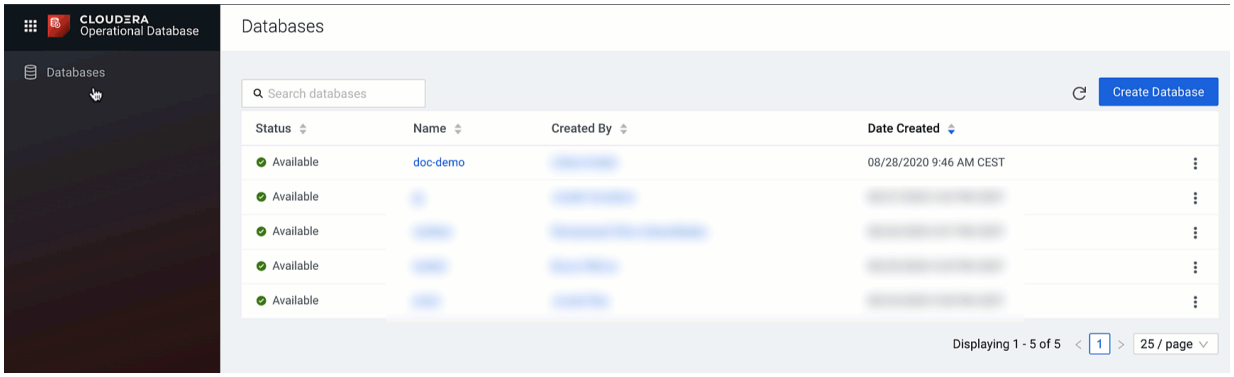

Related Information

Start database
Start database-Beta
Stop database
Start database-Beta
Drop database
Drop database-Beta

Launching hbase shell

To launch the *hbase shell* and use it with Cloudera Operational Database, you have to create an edge node with a configured HBase client tarball.

Before you begin

Configure an edge node. For more information, see *Cloudera Operational Database edge node overview*.

Procedure

1. In your Cloudera Operational Database web interface navigate to Connect HBase Client Tarball .

2. Add your Hbase Client Tarball and HBase Client Configuration.

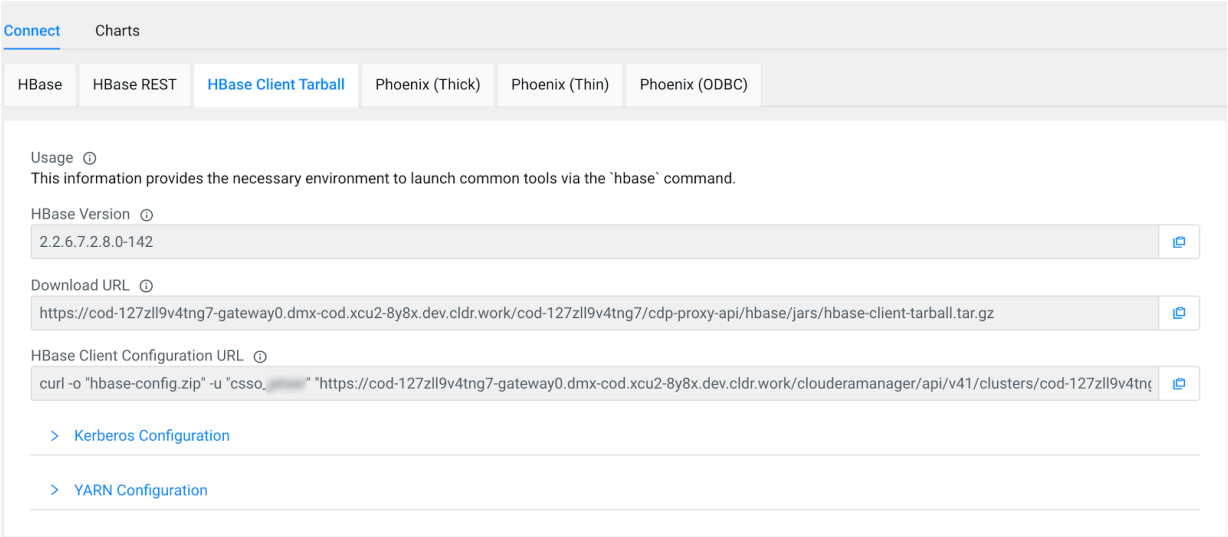

a) Download hbase-client-tarball.tar.gz and extract it to a location on the edge node. This is your *"HBASE_HOME"*.
b) Download the client configuration using the curl command.
c) Extract the client configuration zip file, and move the contained hbase-conf directory to *"HBASE_HOME"* with the name conf (instead of hbase-conf).

3. Launch the hbase shell.

```
$ kinit [***CDP_WORKLOAD_USER***]
Password: ********
$ export HBASE_HOME=hbase-2…
$ cd $HBASE_HOME
$ ./bin/hbase shell
```

4. Validate you can connect through hbase shell.

   Use the list command in hbase shell to list all tables.

5. Add yarn configuration to hbase configuration.

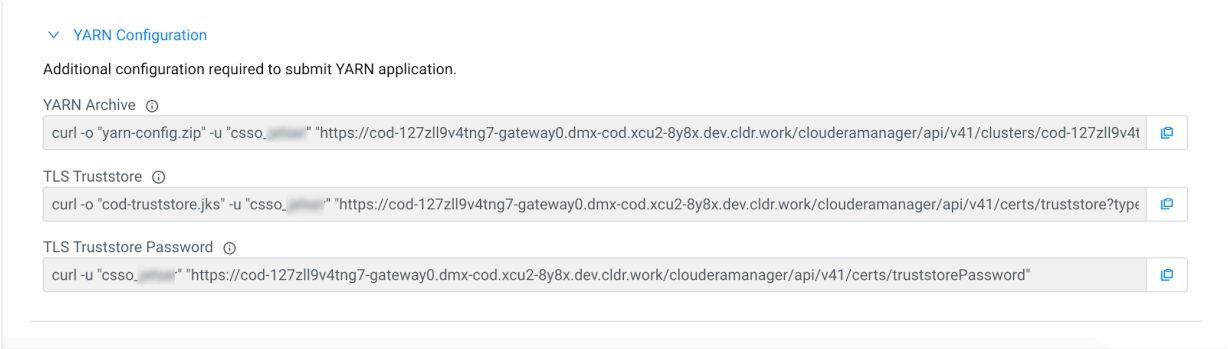

a) Navigate to Connect HBase Client Tarball and expand the YARN Configuration section..
b) Download the yarn-config.zip file.
c) Add the files from yarn-conf to *"HBASE_HOME"*/conf.
d) Download the TLS truststore and remember where you place it.
e) Update the ssl-client.xml file with the TLS TRuststore details.
f) Ensure that in the ssl-client.xml file the ssl.client.trustore.location points to the cod-truststore.jks you have downloaded in step d.
g) Ensure that the ssl.client.truststore.password matches the TLS Truststore Password.

Note: If you plan to use the hbck2 command on Cloudera Operational Database client side, set the HBASE_CLASSPATH environment variable pointing to the HBase client tarball lib directory as follows:

```
export HBASE_CLASSPATH=/<HBase_client_tarball_location>/hbase-client-tarball/lib/:
```

Results

You have an edge node with configured HBase client tarball and you can launch the hbase shell.

Related Information

Cloudera Operational Database edge node overview

Configuring strong meta servers

The strong meta feature uses an HBase RegionServerGroup (RSGroup) to provide dedicated servers to the HBase system tables. This prevents customer table regions from being hosted on the same RegionServers as the system regions and prevents the load from these custom tables from affecting system table access and initialization.

Before you begin

- You must have the COD_STRONG_META_SERVERS entitlement to use this feature.
- This feature is only supported on AWS environments.
- By default, meta replication is enabled and the replica number is set to 3. Therefore, ensure that each strong meta node contains at least 3 meta regions. This does not require any manual steps.

Procedure

1. Launch the CDP CLI tool.

2. Run the following command to add the desired number of strong meta servers.

```
cdp opdb update-database --environment-name envName --database-name dbName
 --num-desired-strong-meta-servers 3
```

Note: The minimum number of strong meta servers to be added is 3 because the meta replication is already enabled in Cloudera Operational Database and the minimum number of supported meta regions is 3.

The above command creates 3 strong meta nodes, adds them to the cluster, and puts a region server on them.

3. To verify if the nodes are healthy, ensure that the logs on these nodes have no errors or exceptions. If data on the cluster and balancer is enabled, these nodes must serve random regions because the balancer moves regions there.
4. Ensure that the RSGroup feature is enabled on the Cloudera Operational Database cluster. To enable this, use the `list_rsgroups` command in the hbase shell.

   If the command is not enabled, the following error is displayed.

```
org.apache.hadoop.hbase.exceptions.UnknownProtocolException: No registered
 Master Coprocessor Endpoint found for RSGroupAdminService. Has it been en
abled?
```

   a) To enable the RSGroup feature, check whether hbase.master.loadbalancer.class property is configured. If configured, move its value to hbase.rsgroup.grouploadbalancer.class property.
   b) Set the value of hbase.master.loadbalancer.class property to org.apache.hadoop.hbase.rsgroup.RSGroupBasedLoadBalancer in the hbase-site.xml file.
   c) Set the value of hbase.coprocessor.master.classes property to org.apache.hadoop.hbase.rsgroup.RSGroupAdminEndpoint in the hbase-site.xml file.
   d) Restart the master nodes for these changes to take effect.
   e) To verify if the setup was successful, use the `list_rsgroups` command in the hbase shell.
5. Create a RegionServerGroup and add the strong meta nodes to it. Ensure that the system tables are also added. Execute the following steps in an hbase shell.

```
// verify that the feature works and every worker/strongmeta nodes are v
isible
list_rsgroups
//create the group (please note, that a different name can be also used)
add_rsgroup "strongmeta"
//verify that the group is created
list_rsgroups
//move the strongmeta nodes to the group (node names can be copied from
the list_rsgroups output)
move_servers_rsgroup "strongmeta",["cod--xxx-strongmeta0.cod-7216.xcu2-8y8
x.dev.cldr.work:16020","cod--xxx-strongmeta1.cod-7216.xcu2-8y8x.dev.cldr
.work:16020","cod--xxx-strongmeta2.cod-7216.xcu2-8y8x.dev.cldr.work:1602
0"]
//verify that the nodes are added to the group
list_rsgroups
//move the system tables to the group
move_tables_rsgroup "strongmeta",["hbase:meta"]
move_tables_rsgroup "strongmeta",["hbase:ack"]
move_tables_rsgroup "strongmeta",["hbase:namespace"]
move_tables_rsgroup "strongmeta",["hbase:rsgroup"]
//verify that the tables were moved
list_rsgroups
```

6. If the previous steps are successful, ensure that everything is functioning as expected using the HBase UI.

What to do next

Ensure that each strong meta node now only serves system table regions.

Importing and restoring data into Cloudera Operational Database database

You can import your data into your Cloudera Operational Database by restoring your HBase table into Cloudera Operational Database.

Before you begin

- Enable HBase replication on your Cloudera Operational Database cluster. For more information, see *Cloudera Operational Database data replication*.
- Have a location in cloud storage (for example, S3 or ABFS) with an exported snapshot in it, and have the name of the snapshot.

  If you do not already have an exported HBase snapshot, you can export your data to cloud storage using the following command:

  ```
  hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snaps
  hot [***SNAPSHOT NAME***] -copy-to [***CLOUD STORAGE LOCATION***] -mappers
   10
  ```

  For example, the data-from-onprem snapshot can be exported into s3a://cod-external-bucket/hbase:

  ```
  hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot data-from-
  onprem -copy-to s3a://cod-external-bucket/hbase -mappers 10
  ```

- Have and edge node with a configured HBase client tarball and know how to launch the hbase shell from it. For more information, see *Launching HBase shell*.

Procedure

1. Get your CLOUD STORAGE LOCATION for your Cloudera Operational Database using the Cloudera Operational Database web interface.
   s3a://my_cod_bucket/cod-12345/hbase

   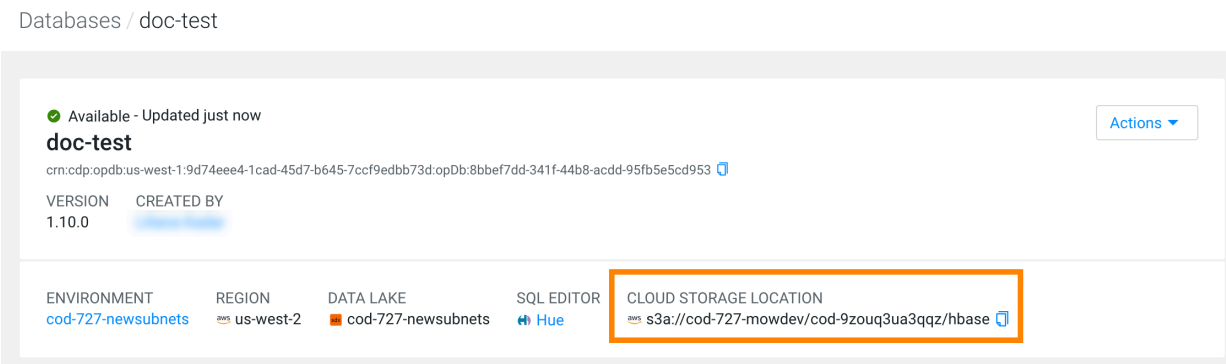

2. Add your bucket to the IAM policy used by IDBroker.

   For more information, see one of the following documentation:

   - AWS Environments: Minimal setup for cloud storage
   - Azure Environment: Minimal setup for cloud storage

3. Launch the hbase shell from the edge node.

   For more information, see *Launching HBase shell*.

4. From the edge node, run the ExportSnapshot command. Use the external bucket as the source location and the Cloudera Operational Database cloud storage as the target.

   For example:

```
$ cd $HBASE_HOME
$ ./bin/hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot
 “data-from-onprem” --copy-from s3a://cod-external-bucket/hbase --copy-to
 s3a://my_cod_bucket/cod-12345/hbase
```

5. Use the list_snapshots command and verify that your snapshot is listed.

```
$ cd $HBASE_HOME
$ ./bin/hbase shell
$ hbase> list_snapshots
[‘snapshot_name’]
```

6. Use the restore_snapshot or the clone_snapshot command to reconstitute the table.

   Warning: The restore_snapshot command overwrites an existing table.

   - restore_snapshot: Overwrites current table state with that of the snapshot. It means that any data modification applied after the snapshot was taken would be lost. If the table does not exist in the given cluster, the command automatically creates it.
   - clone_snapshot: Accepts a new table name for the table in which it restores the table schema and data.

7. Validate that all rows are present in the table using the hbase rowcounter or count command in the hbase shell.

Related Information

Launching hbase shell

Cloudera Operational Database data replication

Using the CldrCopyTable utility to copy data

Backing up a Cloudera Operational Database table

You can backup your Cloudera Operational Database table by creating a snapshot of the table and exporting it to your S3, HDFS, or ABFS bucket.

About this task

Create snapshot command takes a snapshot of the HBase table, exports the snapshot to the specified storage location, and deletes the snapshot from HBase after the export to the specified storage location is successful.

Before you begin

- Ensure that you have access to the location where you want to take the backup of the table.
- You have an S3 or ABFS bucket which you want to use to backup an HBase table in Cloudera Operational Database. For example: s3a://cod-backups/hbase
- You have an edge node configured for your Cloudera Operational Database database. For more information, see *Configure Edge Nodes*.

Procedure

1. Run the following command using cdpcli-beta.

   `cdp opdb create-snapshot --environment-name` *ENVIRONMENT_NAME* `--database-name` *DATABASE_NAME* `--table-name` *TABLE_NAME* `--snapshot-name` *SNAPSHOT_NAME* `--snapshot-location` *SNAPSHOT_LOCATION*

   For example,

   ```
   cdp opdb create-snapshot --environment-name odx-wgel6j --database-name j
   pfy --table-name table_6uf7xt --snapshot-name s1 --snapshot-location s3a
   ://cod-7215/cod--yq3p370r6qro/backupdata
   ```

   JSON output:

   ```
   {
       "environmentName": "odx-wgel6j",
       "databaseName": "jpfy",
       "status": "IN_PROGRESS",
       "creationTime": 0,
       "commandID": 1546336350
   }
   ```

2. Validate that the snapshot and data is present using the list-snapshots command.

   The following command lists the snapshots created in the environment against the Cloudera Operational Database database filters like table name, command ID, and time range.

   `cdp opdb list-snapshots --environment-name` *ENVIRONMENT_NAME* `--database-name` *DATABASE_NAME* [`--table-name` *TABLE_NAME*] [`--command-id` *COMMAND_ID*] [`--from-creation-time` *FROM_CREATION_TIME*] [`--to-creation-time` *TO_CREATION_TIME*]

   Examples

   Without filters

   ```
   cdp opdb list-snapshots  --environment-name odx-wgel6j --database-name j
   pfy
   {
       "environmentName": "odx-wgel6j",
       "databaseName": "jpfy",
       "snapshots": [
           {
               "tableName": "table_tob8vc",
               "snapshotName": "snapshot3_table_tob8vc",
               "creationTime": 1662014265011,
               "status": "SUCCESSFUL",
               "commandID": 1546336058,
               "snapshotLocation": "abfs://qedailynat-filesystem@qedailynat
   storageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase"
           },
           {
               "tableName": "table_tob8vc",
               "snapshotName": "snapshot2_table_tob8vc",
               "creationTime": 1662014023725,
               "status": "SUCCESSFUL",
               "commandID": 1546336035,
               "snapshotLocation": "abfs://qedailynat-filesystem@qedailynat
   storageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase"
           },
           {
               "tableName": "table_tob8vc",
               "snapshotName": "snapshot1_table_tob8vc",
               "creationTime": 1662013782370,
   ```

```
            "status": "DELETED",
            "commandID": 1546336077,
            "snapshotLocation": "abfs://qedailynat-filesystem@qedailyna
tstorageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase"
        },
        {
            "tableName": "table_lpirwy",
            "snapshotName": "snapshot_table_lpirwy",
            "creationTime": 1662013568312,
            "status": "DELETED",
            "commandID": 1546336004,
            "snapshotLocation": "abfs://qedailynat-filesystem@qedailyna
tstorageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase"
        }
]
}
```

With command ID filter

```
$ cdp opdb list-snapshots  --environment-name odx-wgel6j --database-name
 jpfy --command-id 1546336385
{
    "environmentName": "odx-wgel6j",
    "databaseName": "jpfy",
    "snapshots": [
        {
            "tableName": "table_6uf7xt",
            "snapshotName": "s2",
            "creationTime": 1662035967244,
            "status": "SUCCESSFUL",
            "commandID": 1546336385,
            "snapshotLocation": "abfs://qedailynat-filesystem@qedailyna
tstorageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase/"
        }
    ]
}
```

With table name filter

```
$ cdp opdb list-snapshots  --environment-name odx-wgel6j --database-name
 jpfy --table-name table_6uf7xt
{
    "environmentName": "odx-wgel6j",
    "databaseName": "jpfy",
    "snapshots": [
        {
            "tableName": "table_6uf7xt",
            "snapshotName": "s2",
            "creationTime": 0,
            "status": "SUCCESSFUL",
            "commandID": 1546336385,
            "snapshotLocation": "abfs://qedailynat-filesystem@qedailynats
torageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase/"
        }
    ]
}
```

Related Information

Launching hbase shell

Create snapshot

List snapshot

HBase REST server scaling in Cloudera Operational Database

Using Apache HBase REST API, you can scale up the HBase REST server for better connectivity to Cloudera Operational Database.

Important: To use this feature in your Cloudera environment, you must have the COD_RESTWORKERS entitlement enabled. Contact Cloudera Support or your Cloudera account team if you do not have this entitlement.

Multiple Knox Gateway servers are required to scale up the HBase REST servers and support the increased load, so you must have multiple gateway nodes. When you define multiple gateway nodes, the Apache Knox Gateway instances work in an HA mode, and the load is balanced among the multiple Knox instances. Each gateway node hosts an Apache Knox Gateway instance that provides access to the HBase REST server hosted on a REST worker node. The maximum number of gateway nodes supported is 10.

You can specify the required number of REST worker nodes using the --restworker-nodes-count option in the `create-database` command. This optional parameter can only be defined when you specify the --gateway-nodes-count option. Using the `update-database` command, you can scale up the number of the REST worker nodes after database creation. The default number of REST worker nodes supported is 0.

The recommended Cloudera Operational Database deployment type is Single-AZ to leverage the scaling of HBase REST servers.

Benefits of using a REST server

- When a direct line of sight between the client and all HBase servers cannot be established, you can connect the client to a Cloudera Operational Database-deployed load balancer backed by Knox and REST servers, allowing seamless connectivity with Cloudera Operational Database.
- This setup can leverage the HTTP authentication proxy, HTTP - HTTPS proxy, and OAuth proxy features of Knox. For example, Knox can act as an HTTP authentication proxy to support JWT for REST servers, with the flexibility to integrate other authentication providers as needed.
- This setup is language-independent, which means you can use any client, regardless of the programming language, to interact with Cloudera Operational Database.

Configuring the HBase REST server scaling

Know how to configure an HBase REST server scaling.

About this task

When you configure an HBase REST server:

- The heap of Knox is increased to 31 GiB on HEAVY and 2 GiB on LIGHT scale Cloudera Operational Database clusters respectively.
- The Knox parameter gateway.httpserver.requestHeaderBuffer is set to 1 MiB as a safety valve configuration in Cloudera Manager. For more information on this parameter, see the *Apache Knox documentation*.
- The heap size of the HBase REST server is set to 16 GiB on REST worker nodes.

Note: Cloudera recommends exercising caution while using the gateway nodes because these instance types are expensive. If you wish to save costs, use them judiciously.

Before you begin

- To utilize this functionality, you need at least two gateway nodes. The required number of gateway nodes can be specified using the --gateway-nodes-count option in the `create-database` command.
- To enable this functionality, you must create at least two gateway nodes.
- You must have the ODAdmin rights to make changes to the Cloudera Operational Database database.
- You must download and install the latest CDP CLI beta version. For more information, see *Installing Beta CDP CLI*.
- Cloudera recommends using Java 11 to create the clusters. Java 8 has a slow TLS implementation, which significantly reduces the HTTPS throughput of both the REST server and Knox.

Procedure

1. Launch the CDP CLI tool.
2. Create the database with the desired scale type and number of gateway nodes using the --gateway-nodes-count option.

   Important: In the case of the HEAVY scale type, the Gateway node is order of magnitude stronger than that of the LIGHT scale type (32 vCPU and 256 GiB memory vs 8 vCPU 32 Gib memory), and the allocated heap for the hosted Knox is 31 GiB. Thus the gateway node can handle a much higher load with fewer node numbers. You must consider this while planning the number of gateway nodes.

   `cdp opdb create-database --environment-name` <*ENV_NAME*> `--database-name` <*DATABASE_NAME*> --scale-type HEAVY --gateway-nodes-count *INTEGER* `--disable-multi-az` --java-version 11

   Optionally, you can specify the number of REST worker nodes using the --restworker-nodes-count option.

   `cdp opdb create-database --environment-name` <*ENV_NAME*> `--database-name` <*DATABASE_NAME*> --gateway-nodes-count <*INTEGER*> --restworker-nodes-count <*INTEGER*> `--disable-multi-az` --java-version 11

What to do next

If you prefer to scale up or down an HBase REST server, use the following `update-database` command.

Note: The REST server is CPU-bound, if the CPU utilization is too high, you must increase the number of nodes.

`cdp opdb update-database --environment-name` <*ENV_NAME*> `--database-name` <*DATABASE_NAME*> --num-desired-restworker-nodes <*INTEGER*>

For example,

```
cdp opdb update-database --environment-name cod-env --database-name cod-db -
-num-desired-restworker-nodes 4
```

Use the following command to scale down an HBase REST server to zero instances.

`cdp opdb update-database --environment-name` <*ENV_NAME*> `--database-name` <*DATABASE_NAME*> --remove-restworker-nodes

Related Information

Installing Beta CDP CLI

Apache Knox documentation

Limitations

Know the limitations and known issues while scaling an HBase REST server.

Limitations

- The number of gateway nodes cannot be modified after the database is created, so plan the number of requested gateway nodes based on the estimated load.
- You can scale the HBase REST servers in Cloudera Operational Database clusters deployed in Amazon Web Services (AWS) environments.
- The network address translation (NAT) could interfere with traffic distribution because the current setup relies on a Network Load Balancer for sticky sessions, which may cause all requests to route to a single Knox server. Future Cloudera Operational Database versions plan to use an Application Load Balancer for more effective distribution based on session cookies. Until then, NAT must be avoided if more than one Knox or REST server is in use.
- Knox nodes must be specified at the time of cluster creation, as scaling Knox nodes after deployment is currently not supported.
- A Multi-AZ setup can increase data transfer costs, as data might need to pass through multiple services, such as Knox, REST, and HBase region servers across different availability zones before reaching the client.
- HBase Client Tarball downloads might fail if Knox re-routes the request to the wrong gateway.
- The MICRO scale type does not support multiple Knox instances and scaling of HBase REST servers.
- Changing the Knox Provider Configurations and Descriptors using the Knox Admin UI only changes the provider configurations, descriptors, and the generated topologies of the Knox instance whose Admin UI is used.

Known issues

LB based client configs and tarball URLs in the describe-client-connectivity command output fails intermittently

Due to a known Cloudbreak issue, CB-26030, downloading an HBase client configuration intermittently fails when using multiple Gateway nodes.

Retry the download operation until it succeeds.

describe-database command returns incorrect output.

After creating the Cloudera Operational Database cluster with the required number of gateway and REST worker nodes, when you run the describe-database command, the output always shows 0 number of gateway and REST worker nodes.

Upgrade to the latest Cloudera Operational Database version.

Rolling operating system upgrade does not function correctly

When you try to perform a rolling upgrade on the Cloudera Operational Database cluster, it does not work with multiple gateway and REST worker nodes.

Upgrade to the latest Cloudera Operational Database version.

Ensure HBase populates pre-existing object storage data

Cloudera Operational Database databases store data in cloud storage. The location of the data is stored in cloud storage is defined by the data access cloud storage location in the Cloudera environment and the Cloudera Operational Database database name .

When a Cloudera Operational Database database is dropped, any data stored in the cloud storage is not dropped. You must delete the data separately if you no longer need it. If a Cloudera Operational Database database is created against a cloud storage location which already contains Cloudera Operational Database data from a previous instance of Cloudera Operational Database, Cloudera Operational Database will automatically re-create the tables with the corresponding data in the new Cloudera Operational Database instance. Multiple Cloudera Operational Database databases must not refer to the same cloud storage location as this will cause data loss and data corruption.

If a Cloudera Operational Database database is created using the same name against a cloud data access storage location, which already contains Cloudera Operational Database data from a previous instance of Cloudera

Operational Database, Cloudera Operational Database will automatically re-create the tables with the corresponding data in the new Cloudera Operational Database instance.

Managing custom images in Cloudera Operational Database

You can customize a default image for compliance or security reasons. You can then use the CDP CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create an operational database.

To know more about the custom images, see *Custom images and image catalogs*.

Related Information

Custom images and image catalogs

Creating a database using a custom image

Cloudera Operational Database allows you to create a database using a custom image for compliance or security purposes. You can inherit pre-installed packages or software libraries from the custom image while creating an operational database.

Before you begin

- You must download and install the latest CDP CLI beta version.
- You must have a working Cloudera environment.
- You must ensure that the custom image resides in your Cloudera environment.

Procedure

1. Run the following command to create a database using a custom image.

   `cdp opdb create-database --environment` <*ENV_NAME*> `--database` <*DATABASE-NAME*> `--image` < `"id"` : <*IMAGE_ID*>, `"catalog"` : <*CATALOG_NAME*>>

Parameter value	Description
<env_name>	Name of the environment that holds the new database.
<database-name>	Name of the new database for which you want to use the custom image.
<image_id>	Image ID of the database image.
<catalog_name>	Name of the database image catalog.

   For example,

   ```
   cdp opdb create-database --environment-name cod-7215 --database-name tes
   tdb --image '{"id":"78aeac7f-4a37-4395-a270-833d655c20fb", "catalog":"cdp-
   default"}'
   ```

2. Verify that the database creation is successful for the same image.

   Navigate to the Data Hub Clusters service or to the Cloudera Management Console Data Hub Clusters , and click the Image Details tab of the newly created Cloudera Data Hub cluster.

Upgrading a database using a custom image

You can upgrade an existing database using a custom image. The database inherits the packages and software libraries of the custom image, and is upgraded to the provided database image.

Before you begin

- You must download and install the latest CDP CLI beta version.
- You must have a working Cloudera environment.
- You must ensure that the custom image resides in your Cloudera environment.

Procedure

1. Run the following commands sequentially to upgrade a database using a custom image.
   a. `cdp opdb upgrade-database --environment` <*ENV_NAME*> `--database` <*DATABASE_NAME*> `--image-id`=string
   b. `cdp opdb upgrade-database --environment` <*ENV_NAME*> `--database` <*DATABASE_NAME*> `--image-id`=string `--os-upgrade-only`

Parameter value	Description
<env_name>	Name of the environment that holds the existing database.
<database_name>	Name of the existing database for which you want to use the custom image.
--image-id	Image ID of the database image to which the existing database is upgraded.
--os-upgrade-only	Only perform an Operating System upgrade.

   For example,

```
cdp opdb upgrade-database --environment cod-7216-micro3 --database tempd
b12 --image-id a188a3d7-067f-42f1-9b89-16d40faab24c
```

```
cdp opdb upgrade-database --environment cod-7216-micro3 --database tempd
b12 --image-id a188a3d7-067f-42f1-9b89-16d40faab24c --os-upgrade-only
```

2. Verify that the database upgrade is successful for the same image.

   Navigate to the Data Hub Clusters service or to the Cloudera Management Console Data Hub Clusters , and click the Image Details tab of the upgraded Cloudera Data Hub cluster.

Switching image catalogs in Cloudera Operational Database

You can switch the image catalog of an existing operational database. You may want to switch the image catalog for a database to restrict which Runtime version can be upgraded to, or to move to custom images for an existing database.

Before you begin

- You must download and install the latest CDP CLI beta version.
- You must have a working Cloudera environment.
- You must ensure that the custom image resides in your Cloudera environment.

Procedure

1. Run the following command to switch an image catalog of an existing database.

   `cdp opdb update-database --environment` *<ENV_NAME>* `--database` *<DATABASE_NAME>* `--catalog` *<CATALOG_NAME>*

Parameter value	Description
<env_name>	Name of the environment that holds the existing database.
<database_name>	Name of the existing database for which you want to switch the image catalog.
<catalog_name>	Name of the image catalog that you want to update for the database.

   For example,

   ```
   cdp opdb update-database --environment-name cod-7216-micro2 --database-n
   ame testdb --catalog v3dev-1
   ```

2. Verify that the image catalog switching is successful for the database.

   Navigate to the Data Hub Clusters service or to the Cloudera Management Console Data Hub Clusters , and click the Image Details tab of the updated Cloudera Data Hub cluster.

Enabling HBase region canary

You can enable the HBase canary, which is an optional service, to monitor the health of the HBase RegionServer.

About this task

Cloudera Operational Database provides the CLI option --enable-region-canary to add the HBase canaries to monitor the RegionServer health. You can use this option while creating an operational database using CDP CLI.

The HBase canaries are added under the HBase service in the Cloudera Manager. You can view them under Cloudera Manager HBase Service Configuration tab .

Before you begin

Ensure that you have created a Cloudera environment and is ready to use it.

Procedure

1. Log in to the terminal that has the CDP CLI client installed.
2. Use the following command to enable the HBase region canaries.

   `cdp opdb create-database --environment-name` *ENVIRONMENT_NAME* `--database-name` *DATABASE_NAME* `--enable-region-canary`

   For example,

   ```
   cdp opdb create-database --environment-name cdp_7215 --database-name cod_1
    --enable-region-canary
   ```

Results

The following HBase canaries are added on the Cloudera Manager UI.

- hbase_region_health_canary_enabled (HBase Region Health Canary)
- hbase_region_health_canary_slow_run_alert_enabled (HBase Region Health Canary Slow Run Alert Enabled)
- hbase_canary_alert_unhealthy_region_percent_threshold (HBase Canary Unhealthy Region Percentage Alert Threshold)

Related Information

Create database

Cloudera Operational Database ..

Monitoring and auditing

Date published: 2020-08-14
Date modified: 2024-12-02

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Monitor Cloudera Operational Database metrics.....4

Query Cloudera Operational Database event log and audit events.....5

Collecting operational database diagnostic bundle.....6

Monitor Cloudera Operational Database metrics

You can monitor your Cloudera Operational Database service using the charts in the Cloudera Operational Database user interface.

You can access the charts in the Cloudera Operational Database user interface. You have to navigate to Databases Charts .

The charts provide you with the following information:

- Read Throughput: Read throughput over time. The throughput per node greatly depends on table cell size and data request patterns.
- Write Throughput: Write throughput over time.
- Concurrent Clients: Number of clients currently connected to the database.
- RPC Latency: Latency when receiving or sending bytes from the RPC in the RegionServer.
- Read Requests: Total number of read requests per second.
- Write Requests: Total number of write requests per second.
- P99 Get Latency: 99th percentile of Get operation latency.
- Block Cache Hit Ratio: The percent of the time that requests with the block cache turned on hit the block cache.
- Compaction Queue Size: Size of the compaction queue.

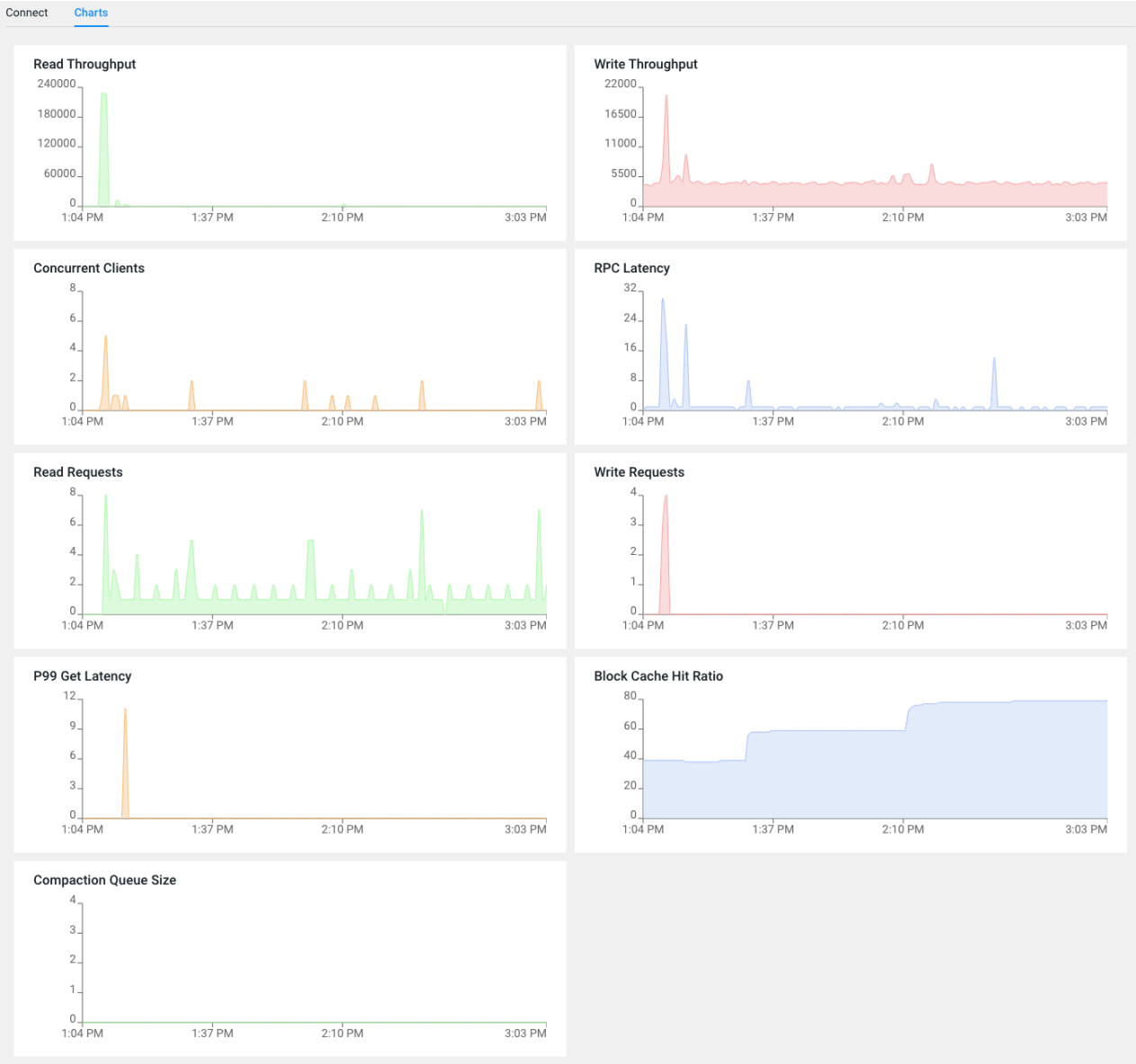

Related Information

Creating triggers and monitoring replication-related metrics in Cloudera Manager

Query Cloudera Operational Database event log and audit events

Cloudera Operational Database logs and audits important events such as the Cloudera Operational Database CRUD operations and database access. You can query these audit events for a specific time interval.

Audit events give you an understanding of the Cloudera Operational Database operations during your specified time interval.

To view the event logs for a specific time interval, run the following command in the Cloudera CLI:

```
$ clients/cdpcli/cdp.sh audit list-events --from-timestamp YYYY-MM-DDT00:00:00Z --to-timestamp YYYY-MM-DDT00:00:00Z --event-source opdb
```

Note that you have to use the ISO-8601 standard timestamps format.

To view the command information and on-screen help, run the following command in the Cloudera CLI:

```
$ clients/cdpcli/cdp.sh audit list-events
```

The audit query result displays event details such as event name, time stamp, accountID, requestID, and result code. Some events do not display a success or failure result codes; these are simple events that indicate that an action succeeded.

Collecting operational database diagnostic bundle

Learn how to download or upload diagnostic bundles to troubleshoot issues in an operational database in Cloudera Public Cloud.

About this task

To troubleshoot issues in your Cloudera Operational Database, you can download diagnostic bundles of log files. These log files are generated when you run some operations in your Cloudera Operational Database environment.

Before you begin

You must be logged into the Cloudera Operational Database as an ODAdmin.

Procedure

1. Log in to the Cloudera Operational Database web interface as an ODAdmin.
2. Find the database for which you want to collect the diagnostic bundle and click ⋮ Collect Diagnostic Bundle .

   The options for generating the diagnostic bundles are displayed as shown in the following image:

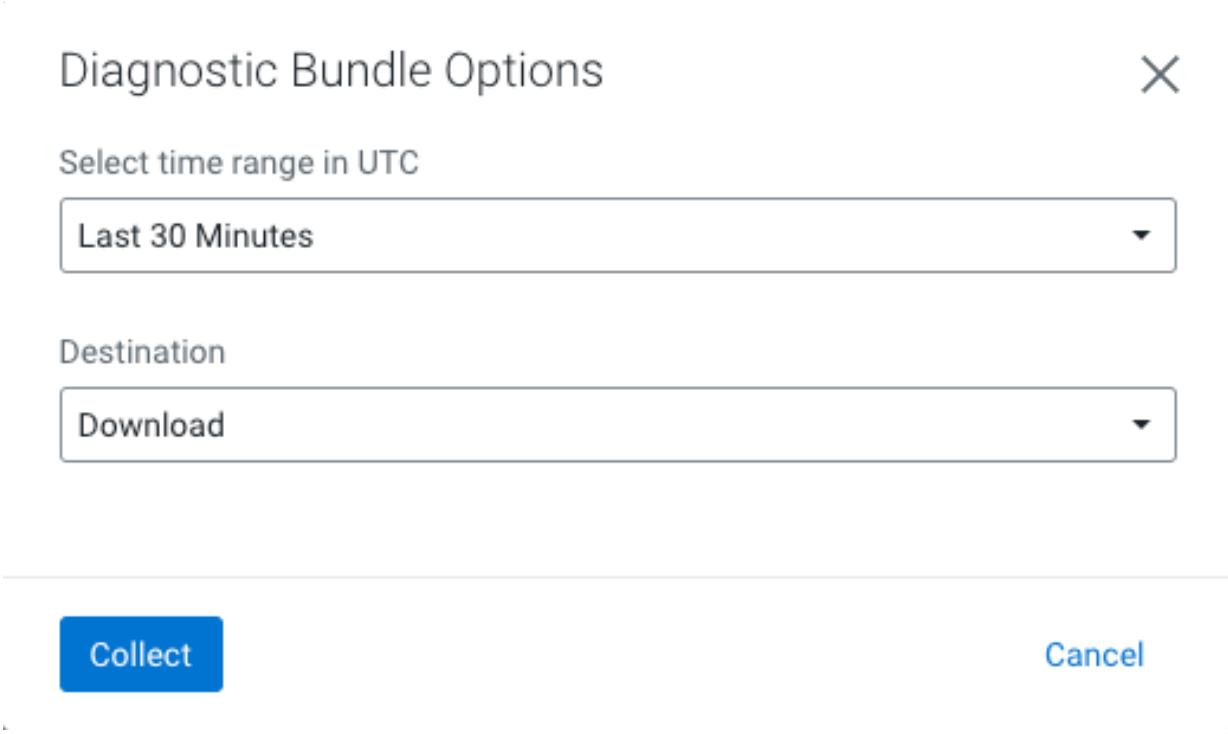

3. Select the time range for which you want to download the diagnostic bundle.
4. Select the destination where you want to collect the diagnostic bundle.
   - Download: You can download a diagnostic bundle to your local computer.
   - Upload: You can upload a diagnostic bundle directly to the Cloudera support incident portal. You need to specify the support case number while uploading it.

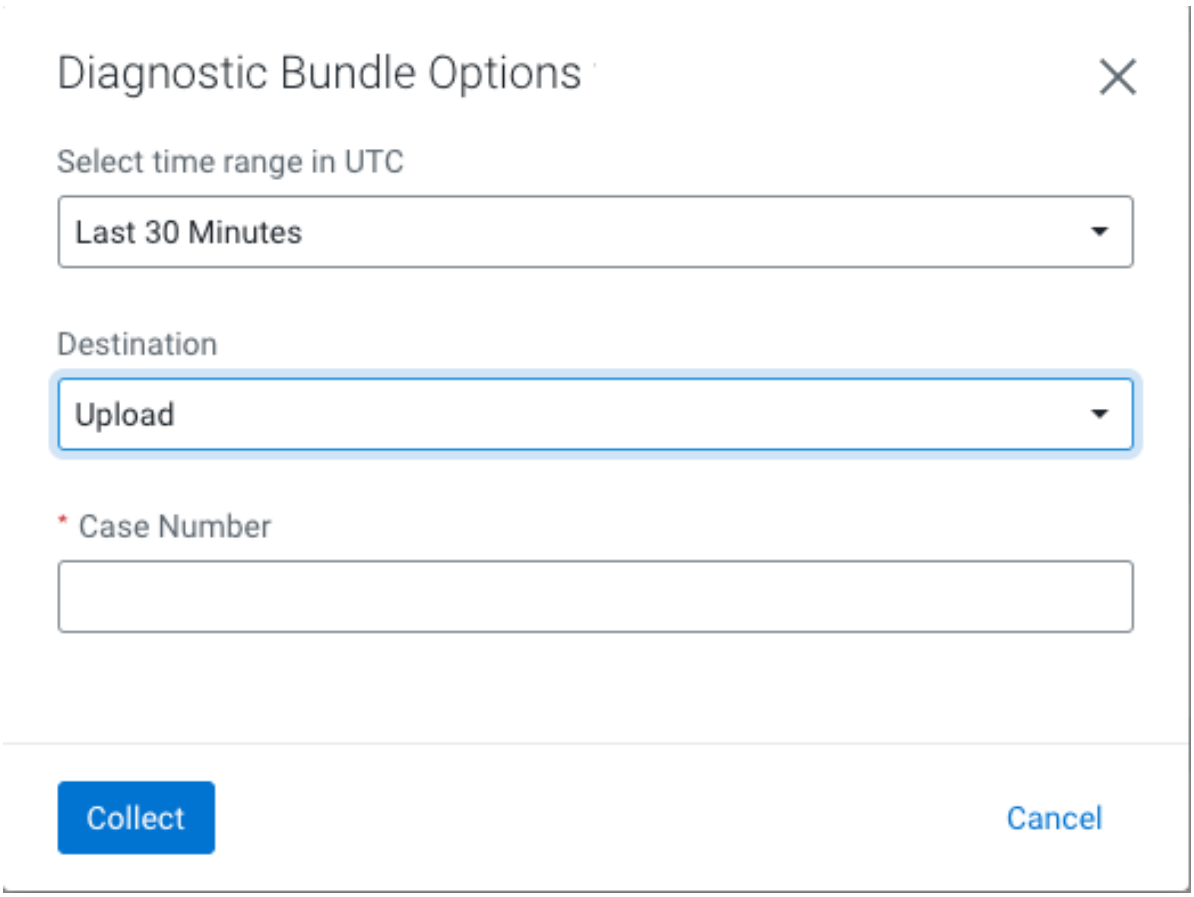

5. Click Collect.

   The following message is displayed:

   “Collection of Diagnostic Bundle for <db_name> initiated. Please go to details page for more information .”
6. Go to Databases and select the database for which you are collecting the diagnostic data.

7. Go to the Diagnostic Bundles tab on the database details page.

   The jobs that have been triggered for generating the diagnostic bundles are displayed, as shown in the following image:

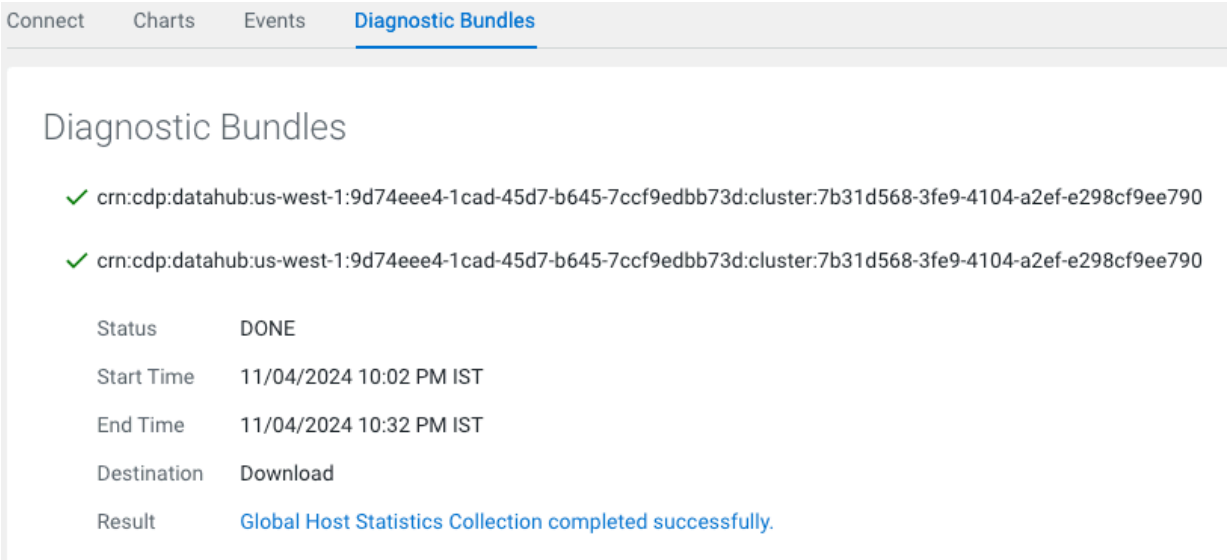

8. Click on the link next to the Result to download the diagnostic bundle to your computer.

Results

The Diagnostic Bundles tab shows the link to download the diagnostic data.

Related Information

Send a diagnostic bundle to Cloudera Support

Onboarding Cloudera Operational Database Users

Date published: 2022-09-20
Date modified: 2023-12-22

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Onboarding Cloudera Operational Database users.....4
Onboarding to Apache components.....4

Migrating HBase data to Cloudera Operational Database.....6

Migrating Phoenix data to Cloudera Operational Database.....6

Onboarding Cloudera Operational Database users

To enable users to work on Cloudera Operational Database and various Cloudera components, like HBase and Phoenix, you can onboard them to Cloudera Operational Database.

Before you begin

1. You have created a Cloudera environment. See *Cloudera environments*.
2. You have created the IDBroker mapping. See *Create IDBroker mapping*.
3. You have a prior knowledge of HBase and Phoenix.

Procedure

1. Set up a workload password in Cloudera to compile any application against Cloudera Operational Database. See *Setting the workload password*.
2. Create an Operational Database on your Cloudera environment. See *Creating a database using Cloudera Operational Database*.
3. Assign the appropriate roles to the Cloudera Operational Database users. See *User management in Cloudera Operational Database*.
4. Configure and deploy an edge node to work with HBase and Phoenix thick clients. See *Configuring edge nodes*.

Related Information

Cloudera environments

Create IDBroker mapping

Setting the workload password

Creating a database using Cloudera Operational Database

User management in Cloudera Operational Database

Cloudera Operational Database edge node overview

How to setup a user in Cloudera

(Admin user) How to create Cloudera Operational Database

How to Deploy an edge node (for use with HBase/Phoenix thick clients)

Onboarding to Apache components

You can create an Operational Database Docker Container to experiment with a simplified setup similar to Cloudera Operational Database, built using Apache HBase, Apache Zookeeper, Apache Omid, and Apache Phoenix.

About this task

Please know that the setup provided in the Docker image is:

- a highly simplified and single node setup, and not comparable to an actual Cloudera Operational Database cluster in performance or management.
- not using the same builds used in Cloudera Operational Database. Docker only contains the freely available upstream builds of some of the components.
- to provide an easy to use and setup environment for learning the basics and experimenting with the technology and not for testing with any load.

Before you begin

1. Launch Docker container and add *OPDB-DOCKER* as localhost in the host file. Run the following command.

```
$ sudo vim /etc/hosts
```

Add the following entry to the host file.

```
127.0.0.1 localhost opdb-docker
```

2. The *OPDB-DOCKER* runs many services in the same container so it is recommended to increase the Docker resource configurations under Preferences > Resources to the following:
   - CPUs: 6
   - Memory: 10 GB
   - Swap: 1.5 GB
   - Disk image size: 64 GB

   Note: It is recommended to set higher values to avoid issues with memory allocation.

Procedure

1. Pull the docker image from Docker Hub.

```
$ docker pull cloudera/opdb-docker
```

2. Run the *OPDB-DOCKER* container.

```
$ docker run -p 8765:8765 -p 8080:8080 -p 8085:8085 -p 9090:9090 \
    -p 9095:9095 -p 2181:2181 -p 16010:16010 -p 16020:16020 -p 16000:16000
 \
    -p 16030:16030 -d -h "opdb-docker"  --name opdb-docker opdb-docker
```

3. Log in to the Docker and run Apache Phoenix or Apache HBase.

```
$ docker exec -it opdb-docker /bin/bash
$ phoenix-sqlline
$ hbase shell
```

4. Run the following commands to start the HBase Thrift and HBase REST servers as they are not started automatically.

```
$ docker exec opdb-docker /opt/hbase/bin/hbase-daemon.sh start thrift
$ docker exec opdb-docker /opt/hbase/bin/hbase-daemon.sh start rest
```

5. Run the following commands to stop the HBase Thrift and HBase REST servers.

```
$ docker exec opdb-docker /opt/hbase/bin/hbase-daemon.sh stop thrift
$ docker exec opdb-docker /opt/hbase/bin/hbase-daemon.sh stop rest
```

Related Information

GitHub Operational Database repository

Operational Database Docker image

Sample applications for Cloudera Operational Database

Migrating HBase data to Cloudera Operational Database

If you have an existing HBase application running on a CDH or HDP environment, you can migrate your data to a Cloudera Operational Database Cloudera Public Cloud environment. You can launch a database with the durable and consistent storage technology you may already be familiar with while using CDH or HDP, but with none of the legacy complexity.

About this task

To migrate your HBase data into a Cloudera Operational Database Cloudera Public Cloud environment, see *HBase Migration through Cloudera Replication Manager*.

Related Information

HBase Migration through Cloudera Replication Manager

Migrating Phoenix data to Cloudera Operational Database

Cloudera Operational Database Replication plugin enables HBase replication from a number of products which also include Phoenix to Cloudera Operational Database, such as CDH 5, CDH 6, HDP 2.6, and HDP 3.1. You can replicate Phoenix tables to Cloudera Operational Database using the Replication plugin.

About this task

Currently, Cloudera Operational Database includes Apache Phoenix 5.1.1 while other products include a range of versions of Phoenix from 4.7.0 to 5.0.0.

To migrate your Phoenix data into a Cloudera Operational Database Cloudera Public Cloud environment, see *Phoenix Replication to Cloudera Operational Database*.

Related Information

Phoenix Replication to Cloudera Operational Database

Cloudera Operational Database ..

Release Notes

Date published: 2020-08-14
Date modified: 2024-12-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What's New 4
- January 15, 2025 4
- December 11, 2024 4
- November 5, 2024 5
- September 3, 2024 7
- May 30, 2024 8
- April 28, 2024 9
- March 28, 2024 10
- February 28, 2024 10
- February 7, 2024 11
- Older releases 11
  - December 18, 2023 11
  - October 26, 2023 12
  - September 28, 2023 13
  - August 30, 2023 13
  - August 10, 2023 14
  - June 19, 2023 15
  - May 31, 2023 16
  - May 10, 2023 16
  - March 10, 2023 17
  - January 10, 2023 17
  - December 10, 2022 18
  - November 10, 2022 19
  - October 10, 2022 19
  - August 10, 2022 19
  - June 10, 2022 20
  - April 10, 2022 20
  - March 10, 2022 21
  - February 10, 2022 21
  - December 10, 2021 22
  - November 10, 2021 22
  - October 10, 2021 22
  - September 10, 2021 23
  - August 10, 2021 23
  - July 10, 2021 24
  - May 10, 2021 24
  - February 10, 2021 25
  - November 10, 2020 25

Fixed issues 25

Known issues 26

What's New

This section lists major features and updates for the Cloudera Operational Database experience in Cloudera Public Cloud.

January 15, 2025

The Cloudera Operational Database 1.48 version supports the SELinux enforcement and enhancement to the database user management.

Cloudera Operational Database supports Security-Enhanced Linux (SELinux) enforcement

Cloudera Operational Database supports creating a database with SELinux enforcement using the CDP CLI. You must have the CDP_SECURITY_ENFORCING_SELINUX entitlement to be able to use the SELinux support. Please contact Cloudera support if you do not have this entitlement.

The SELinux allows you to set access control through policies. You can set the SELinux mode while creating a new operational database. You can define the SELinux mode using the seLinux parameter in the `create-database` command. The supported SELinux modes are:

- ENFORCING: Enables SELinux in enforced mode, actively applying security policies.
- PERMISSIVE (default): Sets SELinux to permissive mode, logging any security violations without enforcing policies.

If you do not define the seLinux parameter, by default, the PERMISSIVE mode is applied.

The following example shows usage of the seLinux parameter.

`opdb create-database --environment-name` [***ENVIRONMENT_NAME***] `--database-name` [***DATABASE_NAME***] `--security-request` '{"seLinux": string}'

```
opdb create-database --environment-name cod-7218-micro1 --database-name test
DB --security-request '{"seLinux": "ENFORCING"}'
```

```
opdb create-database --environment-name cod-7218-micro1 --database-name test
DB --security-request '{"seLinux": "PERMISSIVE"}'
```

For more information, see *CDP CLI documentation* and *Setting SELinux Mode*.

Assign ODAdmin role at the database level

Cloudera Operational Database supports setting a user as an ODAdmin at the database level. In earlier versions of the Cloudera Operational Database, you could set the ODAdmin at the environment level only, however, for better usability and enhanced security, now you can set it at the database level too.

Related Information

CDP CLI documentation

Setting SELinux Mode

December 11, 2024

The Cloudera Operational Database 1.47 version supports enhancements to the CDP CLI and adding a new storage type while creating a new operational database.

Cloudera Operational Database supports JDK17

Cloudera Operational Database only supports creating a database with Cloudera Runtime 7.3.1 using JDK17. If you use an earlier runtime version, only JDK8 and JDK11 are supported for database creation.

New CLI commands to list and upload certificates

Cloudera Operational Database supports two new CLI commands, list-certificates and upload-certificate.

In an Auto-TLS setup, Cloudera Runtime maintains a global certificate trust store across the cluster to ensure a mutual trust relationship between cluster nodes in secure TLS connections.

You can now upload custom certificates into the global certificate store and distribute them across all nodes, enabling secure Cloudera Operational Database connections from your infrastructure without changing the existing PKI infrastructure, certificates, or Root CA.

The feature is designed to support mTLS authentication from outside of the Cloudera Operational Database network, but it could also be useful for TLS connections from Cloudera Operational Database to other networks in general. The command details are as follows.

- list-certificates: This command lists SHA-1 fingerprints of certificates listed in the Global Trust Store.

  Following is an example,

  ```
  cdp opdb list-certificates --environment <environment_name> --database <
  database_name>
  ```

- upload-certificate: This command uploads a single, PEM-encoded certificate to the Global Trust Store and refreshes all the nodes in the cluster.

  Following is an example,

  ```
  cdp opdb upload-certificate --environment <environment_name> --database
  <database_name> --certificate <custom_certificate_in_PEM_format>
  ```

For more information, see *CDP CLI documentation*.

New storage type support during database creation

The Cloudera Operational Database UI supports a new storage type, Cloud Storage with Caching and Data Tiering while creating an operational database. This storage type is equivalent to a cloud storage that supports time-based priority caching, where data within a specified time range is given a higher priority.

You must have the COD_DATATIERING entitlement to be able to use this storage type.

For more information, see *Creating a database using Cloudera Operational Database* and *HBase Time-based Data Tiering using Persistent BucketCache*.

Related Information

CDP CLI documentation

Creating a database using Cloudera Operational Database

HBase Time-based Data Tiering using Persistent BucketCache

November 5, 2024

The Cloudera Operational Database 1.46 version supports updates to the supported instance types and enhancements to the Operational Database UI.

A new CLI command to get the list of supported instance types

Cloudera Operational Database adds a new CLI command, `list-supported-instance-types`, allowing you to list the supported instance types for a combination of cloud platform, scale type, and storage type. You can filter using the instance group and architecture as well.

The --custom-instance-types option in the `create-database` command allows you to pick and choose instance types for the Cloudera Operational Database. However, it is essential that these instance types are included in the Cloudera Operational Database allowlist. The `list-supported-instance-types` command is useful to fetch the list of supported instance types.

Following is an example of the command.

```
cdp opdb list-supported-instance-types --cloud-platform AZURE --storage-type
 CLOUD_WITH_EPHEMERAL --scale-type MICRO --instance-group WORKER --architect
ure X86_64
```

For more information, see *CDP CLI documentation*.

Enhancements to the update-database command

A new CLI option --switch-instance-type is added to the `update-database` command to switch the instance types to custom instance types for the Cloudera Operational Database clusters after the database creation. You must have the COD_USE_CUSTOM_INSTANCE_TYPES entitlement to use this option.

If you plan to scale up the cluster vertically or change the existing instance type to any custom instance type, you can use the --switch-instance-type option. However, the custom instance types must be included in the allowlist by the Cloudera Operational Database. Use the `list-supported-instance-types` command to fetch the list of supported instance types.

Following is an example of the usage.

```
cdp opdb update-database --environment-name <env_name> --database-name <data
base_name> --switch-instance-type instanceGroup=INSTANCEGROUP,instanceType=i
nstancetype
```

For more information, see *CDP CLI documentation*.

Enhancements to the Cloudera Operational Database UI

The Cloudera Operational Database UI provides two new UI options Configure Database and Configure Edge Nodes to configure the worker and edge node properties in your existing operational database.

You can access these menu items using the Actions menu item inside the database details page.

For more information, see *Configuring a database*.

GA support for the HBase REST server scaling

The feature to scale up the HBase REST server using the Apache HBase REST API is no longer a technical preview and is now generally available (GA).

For more information, see *HBase REST server scaling in Cloudera Operational Database*.

Related Information

CDP CLI documentation

Configuring a database

HBase REST server scaling in Cloudera Operational Database

September 3, 2024

The Cloudera Operational Database 1.45 version supports updates to the HDFS instance types and enhancements to the Operational Database UI.

Enhancements to the create-database command

- The CDP CLI adds a new option --custom-instance-types to the `create-database` command. Using this option, you can define custom instance types; however, the instance types must be included in the allowlist by Operational Database.

  Currently, the allowlist is not accessible, however, the following are the new custom instance types supported by Operational Database.

  AWS, HEAVY, HDFS:

  ```
  Worker: m6i.4xlarge, m7i.4xlarge
  Master: m6i.8xlarge, m7i.8xlarge
  Compute/edge/leader: m6i.2xlarge, m7i.2xlarge
  Gateway: r6i.8xlarge, r7i.8xlarge
  ```

  AWS, LIGHT, HDFS:

  ```
  Worker: m6i.4xlarge, m7i.4xlarge
  Compute/edge/leader/gateway: m6i.2xlarge, m7i.2xlarge
  Master: m6i.4xlarge, m7i.4xlarge
  ```

  The following is a sample output of the `create-database` command highlighting the usage of the --custom-instance-types option.

  ```
  cdp opdb create-database
  --environment-name cod_env
  --database-name cod_db
  --custom-instance-types masterType=m7i.4xlarge,workerType=m7i.4xlarge,lead
  erType=m7i.2xlarge,gatewayType=m7i.2xlarge
  --storage-type=HDFS
  --scale-type LIGHT
  ```

- The --storage-type option is now optional. If you do not define the --storage-type option, Operational Database considers the default storage type.

  The default storage type is blob storage. If the ephemeral storage is enabled, Operational Database considers the storage type as CLOUD_WITH_EPHEMERAL; otherwise, CLOUD is considered.

  For more information, see *CDP CLI documentation*.

Updates to the HDFS clusters on AWS environments to add support for m6i and m7i instance types

When you create an operational database with HDFS storage type in an AWS environment, Operational Database on HDFS clusters now also supports m6i and m7i instances for the applicable nodes. The Operational Database clusters with HDFS storage type are upgraded to enhance the operational database's performance and usability.

The following are the new custom instance types supported by Operational Database.

AWS, HEAVY, HDFS:

```
Worker: m6i.4xlarge, m7i.4xlarge
Master: m6i.8xlarge, m7i.8xlarge
Compute/edge/leader: m6i.2xlarge, m7i.2xlarge
Gateway: r6i.8xlarge, r7i.8xlarge
```

AWS, LIGHT, HDFS:

```
Worker: m6i.4xlarge, m7i.4xlarge
Compute/edge/leader/gateway: m6i.2xlarge, m7i.2xlarge
Master: m6i.4xlarge, m7i.4xlarge
```

Enhancements to the Cloudera Operational Database UI

The Cloudera Operational Database UI is updated for better usability and performance. The following are the key enhancements.

- The cloud form factor on which the database is deployed is shown as a logo under the Environment column.
- The Version column is renamed to the Runtime Version in the Databases screen.
- A new column Node Count is added to the Databases screen.
- The Date Created column name is changed to Uptime in the Databases screen.
- The SQL Editor is renamed HUE and displayed as a link under the new column Apps.
- A new action menu item Collect diagnostic bundle is added to the Databases *DATABASE_NAME* Actions . For more information, see *Collecting operational database diagnostic bundle*.

Related Information

CDP CLI documentation

Collecting operational database diagnostic bundle

May 30, 2024

Cloudera Operational Database 1.42 version supports HBase REST server scaling and CDP CLI enhancements.

The HBase REST server scaling for better performance [Technical Preview]

You can scale up the HBase REST server using the Apache HBase REST API, for better connectivity to Cloudera Operational Database. You need a minimum of two Gateway nodes to utilize this functionality. The required number of Gateway nodes can be specified using the --gateway-nodes-count option in the `create-database` command using CDP CLI.

This feature is under technical preview. To use this feature, you must have the COD_RESTWORKERS entitlement enabled in your Cloudera environment.

Following is a sample command.

`cdp opdb create-database --environment-name` *ENV_NAME* `--database-name` *DATABASE_NAME* --gateway-nodes-count *INTEGER*

For more information, see *Scaling the HBase REST server in Cloudera Operational Database*.

Enhancements to the `describe-database` command

In CDP CLI, the output of the `describe-database` command shows the JDK version of the Cloudera Operational Database cluster if the cluster was created using a specific JDK version; otherwise, the output shows the JDK version as "Not Available".

The following is a sample output of the describe-database command that shows the Java version used to create the cluster.

```
"dbEdgeNodeCount": 0,
"scaleType": "MICRO",
"type": "COD",
"computeNodesCount": 0,
"totalComputeNodesCount": 0,
"isJwtEnabled": true,
```

```
"cloudPlatform": "AWS",
"javaVersion": "11"
```

For more information, see *CDP CLI documentation*.

Related Information

Scaling the HBase REST server in Cloudera Operational Database

CDP CLI documentation

April 28, 2024

Cloudera Operational Database 1.41 version supports CDP CLI changes and upgrade to higher instance types for HDFS storage type and Azure deployments.

Cloudera Operational Database has updated the HDFS instance type to 16 core instances

When you create an operational database with HDFS storage type, the Cloudera Operational Database clusters now use 16 core instances on AWS, Azure, and GCP environments for worker nodes. The Cloudera Operational Database clusters with HDFS storage type are upgraded to enhance the performance and usability of the Cloudera Operational Database.

The new worker instances for HDFS storage type are as follows:

- AWS: m5.4xlarge
- Azure: Standard_D16_v3
- GCP: n2-standard-16

Cloudera Operational Database supports configuring root volume size for available instances in a Cloudera Operational Database cluster

In CDP CLI, while creating an operational database, you can set the default root volume size with the --root-volume-size (integer) option in GiB for all the instances in the cluster.

Following is a sample command.

```
cdp opdb create-database --environment-name test-env --database-name test-db
 --root-volume-size 300
```

For more information, see *CDP CLI documentation*.

Cloudera Operational Database has updated the instance type for Azure deployments

When you create and deploy an operational database in an Azure environment, by default, Cloudera Operational Database clusters now use Standard_D8s_v3 instance type instead of Standard_D8_v3. The instance type is upgraded to support encryption at the host level.

If you want to retain the Standard_D8_v3 instance type, you must have the COD_USE_DV3_INSTANCE_TYPE entitlement on your account.

Cloudera Operational Database supports instance group encryption in AWS environments

In CDP CLI, while creating an operational database, you can specify the encryption key to encrypt the volume for instance groups using the --volume-encryptions (array) option. You can specify this option only in AWS environments.

Following is a sample command.

```
opdb create-database --environment-name <environment-name> --database-name <
database-name> --disable-external-db --scale-type MICRO --attached-storage-f
```

```
or-workers '{"volumeCount":1,"volumeType":"SSD","volumeSize":100}' --endpoin
t-url http://localhost:8988 \
--volume-encryptions '[
  {
    "encryptionKey": "<aws-key-arn>",
    "instanceGroup": "GATEWAY"
  }
]'
```

Shorthand syntax: encryptionKey=string,instanceGroup=string ... (separate items with spaces)

JSON syntax:

```
[
            {
              "encryptionKey": "string",
              "instanceGroup": "WORKER"|"LEADER"|"MASTER"|"GATEWAY"|"STRONG
META"|"EDGE"
            }
            ...
          ]
```

For more information, see *CDP CLI documentation*.

Related Information

CDP CLI documentation

March 28, 2024

Cloudera Operational Database 1.40 version provides enhancements to the create database UI wizard.

Cloudera Operational Database supports enhancements to the Create Database UI wizard

When you create an operational database using the Cloudera Operational Database UI, the Create Database option provides enhancements to the database creation workflow. In addition to the enhanced look and feel for better usability, the workflow also provides two additional steps called, Settings and Review. In the Settings step, you can view the default settings for your database and can also modify them. In the Review step, you can view, and finalize your changes, and also obtain a complete CLI command, in case you want to create the database using CDP CLI.

For more information, see *Creating a database using Cloudera Operational Database*.

Related Information

Creating a database using Cloudera Operational Database

CDP CLI Beta

February 28, 2024

Cloudera Operational Database 1.39 version removes a CDP CLI command and provides support for GP3 for attached storage.

Cloudera Operational Database has removed the CDP CLI command, `disengage-auto-admin`

Cloudera Operational Database has removed the support for `disengage-auto-admin` command, which allowed users to disable the autonomous functions of the database and use the underlying DataHub cluster instead.

Cloudera Operational Database supports GP3 for attached storage disks

Cloudera Operational Database now supports GP3 (SSD) volume types for attached storage. GP3 volumes allow you to increase performance (independently provisioning IOPS and throughput) without increasing storage size. GP3

volumes deliver similar performance as similar GP2 volumes at a lower cost. GP3 is now the default attached storage type for Cloudera Operational Database instances that previously used GP2 storage.

February 7, 2024

Cloudera Operational Database 1.38 version introduces a new CDP CLI command.

Cloudera Operational Database has introduced a new CDP CLI command `prepare-upgrade-database`

The new CDP CLI command `prepare-upgrade-database` is used during the preparation phase of the runtime upgrades. This command performs all validations and downloads all required parcels for the upgrade operation. Following is a sample command.

`cdp opdb prepare-upgrade-database --environment` <*ENVIRONMENT-NAME*> `--database` <*DATABASE-NAME*> [`--runtime` <*RUNTIME-VERSION*>] [`--imageId` <*IMAGEID*>]

You can either use the runtime version [`--runtime`] or the image ID [`--imageId`] of the cluster while using this command.

For more information, see *CDP CLI Beta*.

Related Information

CDP CLI Beta

Older releases

Overview of new features, enhancements, and changed behavior introduced in earlier releases of Cloudera Operational Database.

December 18, 2023

Cloudera Operational Database 1.37 version supports modifications to the entitlements and Multi-AZ deployment on an Azure environment.

Cloudera Operational Database has enabled the COD_ON_GCS entitlement

You can deploy Cloudera Operational Database on a Google Cloud Platform (GCP) by using Google Cloud Storage (GCS) similar to what is available for Amazon Web Services (AWS) S3 storage and Microsoft Azure blob storage. Now, Cloudera Operational Database has enabled the COD_ON_GCS entitlement, by default, for such a deployment.

Cloudera Operational Database has removed the COD_EDGE_NODE entitlement

Cloudera Operational Database has removed the COD_EDGE_NODE entitlement now because it is not needed anymore. Cloudera Operational Database edge node functionality is enabled for all Cloudera Operational Database customers now.

Cloudera Operational Database has removed the COD_STOREFILE_TRACKING entitlement

Cloudera Operational Database has removed the COD_STOREFILE_TRACKING entitlement because it is not needed anymore. The Store File Tracking (SFT) functionality is enabled on all new Cloudera Operational Database clusters created with cloud storage.

Cloudera Operational Database has removed the OPDB_USE_EPHEMERAL_STORAGE entitlement

Cloudera Operational Database has removed the OPDB_USE_EPHEMERAL_STORAGE entitlement because it is not needed anymore. The use of Cloudera Operational Database on a cloud storage with ephemeral cache is enabled without an entitlement depending on the cluster creation parameters.

Cloudera Operational Database supports Multiple Availability Zones (Multi-AZ) on Azure [Technical Preview]

Cloudera Operational Database ensures high availability and fault tolerance using Multi-AZ deployments. A Multi-AZ deployment means that compute infrastructure for HBase's master and region servers are distributed across multiple AZs ensuring that when a single availability zone has an outage, only a portion of Region Servers is impacted and clients automatically switch over to the remaining servers in the available AZs.

Multi-AZ for Cloudera Operational Database is now supported on Microsoft Azure environments as a technical preview and is considered under development. For more information, see *Multi-AZ deployment on Cloudera Operational Database*.

Cloudera Operational Database supports a new instance type I4i for Cloud With Ephemeral Storage type databases on AWS environments

When you create a new operational database with Cloud With Ephemeral Storage as the storage type on an AWS environment, Cloudera Operational Database creates the database with an I4i instance type for the worker nodes.

Cloudera Operational Database supports fast autoscaling for higher computing requirements

When you create a new operational database using CDP CLI, you can enable fast autoscaling by defining the required parameters using the --auto-scaling-parameters option. Cloudera Operational Database now supports a new instance group called, Compute. The nodes under this instance group are automatically scaled up or scaled down based on the CPU utilization and RPC latency.

To use fast autoscaling, you must have the COD_USE_COMPUTE_ONLY_NODES entitlement.

Following is a sample command.

`cdp opdb create-database –environment-name` <*ENV_NAME*> `–database-name` <*DB_NAME*> `--auto-scaling-parameters` '{"minComputeNodesForDatabase":<min_compute_nodes>, "maxComputeNodesForDatabase": <max_compute_nodes>}'

For more information, see *The fast autoscaling in Cloudera Operational Database*.

Related Information

Multi-AZ deployment on Cloudera Operational Database

Fast autoscaling in Cloudera Operational Database

October 26, 2023

Cloudera Operational Database 1.36 version supports an UI enhancement and the enabling of multiple entitlements.

Cloudera Operational Database has enabled the OPDB_USE_EPHEMERAL_STORAGE entitlement

Cloudera Operational Database supports large ephemeral block cache while deploying on cloud storage. The entitlement OPDB_USE_EPHEMERAL_STORAGE is enabled by default while using a large ephemeral block cache on any cloud storage.

Cloudera Operational Database introduces a new storage type UI option while creating an operational database

On the Cloudera Operational Database UI, a new storage type option Cloud With Ephemeral Storage is added. This option is equivalent to using the `--storage-type CLOUD_WITH_EPHEMERAL` option on CDP CLI while creating an operational database.

For more information, see *Creating a database using Cloudera Operational Database* and *CDP CLI Beta*.

Cloudera Operational Database has enabled the COD_EDGE_NODE entitlement

Earlier, you were required to have the COD_EDGE_NODE entitlement to create an edge node on your Cloudera Operational Database cluster. Now the entitlement is enabled by default.

Cloudera Operational Database has enabled the COD_STOREFILE_TRACKING entitlement

Earlier, you were required to have the COD_STOREFILE_TRACKING entitlement to use the Store File Tracking (SFT) on your Cloudera Operational Database cluster. Now the entitlement is enabled by default.

Related Information

Creating a database using Cloudera Operational Database

CDP CLI Beta

September 28, 2023

Cloudera Operational Database 1.35 version supports rolling operating system upgrade and enhancements to the CDP CLI.

Cloudera Operational Database supports rolling operating system upgrades of a Cloudera Operational Database cluster

Cloudera Operational Database using *HDFS* and *Object Store without Ephemeral Storage*, now supports upgrading the operating system version of the database using the rolling restart mode. This ensures continuous service availability during an upgrade operation.

In CDP CLI, you need to use the --os-upgrade-only option along with the --rolling-upgrade option in the `upgrade-database` command.

Following is a sample command.

`cdp opdb upgrade-database --environment` <*ENVIRONMENT-NAME*> `--database` <*DATABASE-NAME*> [`--runtime` <*RUNTIME-VERSION*> | `--image` <*IMAGEID*>] `--os-upgrade-only --rolling-upgrade`

For more information, see *Rolling upgrade in Cloudera Operational Database*.

Cloudera Operational Database introduces a new CDP CLI option --storage-type in the `create-database` command

In CDP CLI, a new option --storage-type in the `create-database` command is introduced that replaces the [--use-hdfs | --no-use-hdfs] and [--disable-ephemeral-storage | --no-disable-ephemeral-storage] options.

The --storage-type option supports these values, CLOUD_WITH_EPHEMERAL, CLOUD, and HDFS.

Following is a sample command.

`cdp opdb create-database --environment` <*ENVIRONMENT-NAME*> `--database` <*DATABASE-NAME*> `--storage-type CLOUD`

For more information, see *CDP CLI Beta*.

Cloudera Operational Database drops support of the Cloudera Runtime versions Cloudera Runtime 7.2.12 and earlier

Cloudera Operational Database has stopped supporting the Cloudera Runtime versions Cloudera Runtime 7.2.12 and earlier because they have reached the end of life.

Related Information

CDP CLI Beta

Rolling upgrade in Cloudera Operational Database

August 30, 2023

Cloudera Operational Database 1.34 version supports different JDK versions during Cloudera Operational Database creation and deploying Cloudera Operational Database on GCS.

Cloudera Operational Database supports creating an operational database using JDK8 and JDK11

Cloudera Operational Database now added a new CLI option, –java-version which can be used to configure a major Java version on your Cloudera Operational Database cluster. The new CLI option can be used along with the create-database command to specify the Java version. The supported Java versions are JDK8 and JDK11. In case the parameter is not specified, JDK8 is used. Following is a sample command.

`cdp opdb create-database --environment-name` *<ENVIRONMENT_NAME>* `--database-name` *<DATABASE_NAME>* `--java-version` *<VALUE>*

```
cdp opdb create-database --environment-name cod7215 --database-name testenv
--java-version 11
```

For more information, see *CDP CLI beta*.

Cloudera Operational Database is available as a Technical Preview feature on Google Cloud Storage (GCS)

Cloudera Operational Database on Google Cloud Platform (GCP) can now be deployed by using Google Cloud Storage (GCS) easily, similar to what is available for Amazon Web Services (AWS) S3 storage and Microsoft Azure blob storage. The use of GCS for such a setup requires the COD_ON_GCS entitlement.

Cloudera Operational Database also now supports a large ephemeral block cache while deploying on GCP. The use of ephemeral storage along with any cloud storage still requires the OPDB_USE_EPHEMERAL_STORAGE entitlement.

Cloudera Operational Database has removed the COD_ON_GCP entitlement

COD_ON_GCP entitlement has been removed from Cloudera Operational Database because it is not needed anymore. From this version onwards, customers can create Cloudera Operational Database clusters on Google Cloud Platform (GCP) without it.

Related Information

CDP CLI Beta

August 10, 2023

Cloudera Operational Database 1.33 version provides enhancements to the CDP CLI as well as on Cloudera Operational Database UI.

Cloudera Operational Database drops support of the Cloudera Runtime versions Cloudera Runtime 7.2.8 and Cloudera Runtime 7.2.9

Cloudera Operational Database has stopped supporting the Cloudera Runtime versions Cloudera Runtime 7.2.8 and Cloudera Runtime 7.2.9 because they have reached the end of life.

Cloudera Operational Database supports faster rolling restarts on Cloudera Operational Database clusters

The default value of Cloudera Manager HBase Configuration Region Mover Threads is changed to 30. This speeds up the rolling restart functionality for HBase.

For more information see *Rolling Restart*.

Cloudera Operational Database supports rolling runtime upgrades of a Cloudera Operational Database cluster

Cloudera Operational Database now supports upgrading the Cloudera Runtime version of the database using the rolling restart mode. This ensures continuous service availability during an upgrade operation. A new CLI parameter

--rolling-upgrade | --no-rolling-upgrade is added to the `upgrade-database` command. Following is a sample command:

```
cdp opdb upgrade-database --environment <environment-name> --database <datab
ase-name> --runtime <runtime-version> [--rolling-upgrade | --no-rolling-upgr
ade]
```

For more information, see *PRolling upgrade in Cloudera Operational Database*.

Cloudera Operational Database provides enhancements to the --scale-type CDP CLI option in the `create-database` command

In CDP CLI, the --scale-type option now supports all three options --scale-type (string) <MICRO, LIGHT, HEAVY> for both the --master-node-type and --gateway-node-type.

- --scale-type LIGHT (--master-node-type LITE, --gateway-node-type LITE)
- --scale-type HEAVY (--master-node-type HEAVY, --gateway-node-type HEAVY)

If the --scale-type option is not defined, by default --scale-type LIGHT is considered for both the --master-node-type and --gateway-node-type. However, you can overwrite the --scale-type for a --gateway-node-type using the --gatewa y-node-type <value> option.

For more information, see *CDP CLI Beta*.

Cloudera Operational Database supports enabling a consolidated view of Cloudera Operational Database metrics using Grafana dashboards

In CDP CLI, the `create-database` command now provides a new option --enable-grafana which allows you to enable the Grafana URL under the GRAFANA DASHBOARD option inside your Cloudera Operational Database database. When you click on the Grafana URL, it takes you to the Grafana dashboard which provides a consolidated view of the Cloudera Operational Database metrics.

Following is an example of the `create-database` command.

```
cdp opdb create-database --environment <environment_name> --database <databa
se_name> --enable-grafana
```

For more information, see *Monitoring metrics in Cloudera Operational Database with Grafana*.

Related Information

Rolling Restart

Rolling upgrade in Cloudera Operational Database

CDP CLI Beta

Monitoring metrics in Cloudera Operational Database with Grafana

June 19, 2023

Cloudera Operational Database 1.32 version provides enhancements to the CDP CLI as well as on Cloudera Operational Database UI.

Cloudera Operational Database provides UI enhancements to the Scale option on the database creation page

On the Cloudera Operational Database UI, when you create an operational database the Medium Duty is renamed to Heavy Duty under Create Database Scale . This ensures that the options on Cloudera Operational Database UI and CDP CLI Beta are symmetrical.

For more information, see *Creating a database using Cloudera Operational Database*.

Cloudera Operational Database provides enhancements to the CDP CLI option --scale-type <HEAVY>

In CDP CLI, when you select the --scale-type option as HEAVY, Cloudera Operational Database allocates larger SSD storage (for example, gp2 on AWS, StandardSSD_LRS on Azure, or pd-ssd on GCP) for both master and leader node types. This ensures the higher loads on Zookeeper and provides a better performance for Cloudera Operational Database.

For more information, see *CDP CLI Beta*.

Cloudera Operational Database supports enabling custom recipes using CDP CLI Beta

You can now define custom recipes while creating an operational database using the --recipes option in CDP CLI Beta. You can register pre-created recipes during database creation based on the instance groups in your database. The recipes are executed automatically for the specified nodes based on the recipes' type.

Use the following example command to define custom recipes for your operational database.

```
cdp opdb create-database --environment-name <ENVIRONMENT_NAME> --database-na
me <DATABASE_NAME> --recipe names=<rec1,rec2>,instanceGroup=<MASTER> names=<
rec2>,instanceGroup=WORKER names=<rec3,rec4>,instanceGroup=<GATEWAY>
```

To know more about the recipes, see *Recipes*.

To know more about the --recipes option, see *CDP CLI Beta*.

Related Information

Creating a database using Cloudera Operational Database

CDP CLI Beta

Recipes

May 31, 2023

Cloudera Operational Database 1.31 version provides enhancements to the CDP CLI options.

Cloudera Operational Database provides enhancements to the --scale-type CDP CLI option

In CDP CLI, the --scale-type option now supports all the three options --scale-type (string) <MICRO, LIGHT, HEAVY>. Cloudera Operational Database has added the support for additional parameters, LIGHT and HEAVY.

- --scale-type LIGHT (master-node-type LITE)
- --scale-type HEAVY (master-node-type HEAVY)

Additionally, Cloudera Operational Database has removed the --master-node-type (string) <LITE,HEAVY> CDP CLI option because this option is not needed anymore. For more information, see *CDP CLI Beta*.

Related Information

CDP CLI Beta

May 10, 2023

Cloudera Operational Database 1.30 version supports scaling up the Cloudera Operational Database clusters vertically and also provides an UI option to create smaller Cloudera Operational Database clusters.

Cloudera Operational Database supports scaling up the clusters vertically

Cloudera Operational Database now allows you to vertically scale up the Cloudera Operational Database clusters from a Light Duty to a Medium Duty instance type. You can upgrade the instance type of a Cloudera Operational Database cluster that belongs to a Master or Gateway node.

To know more about the vertical scaling, see *Scaling Cloudera Operational Database instances vertically*.

Cloudera Operational Database UI supports creating a smaller cluster using a predefined Data Lake template

Cloudera Operational Database now allows you to create a smaller cluster with one Gateway node and one Worker node using a new scale type Micro Duty while creating an operational database through Cloudera Operational Database UI. The Micro database is a two node cluster in which the Gateway node processes the activities of the Master and Leader nodes, thereby removing the need of these nodes. You can use a Micro database for testing and development purposes.

For more information, see *Creating a database using Cloudera Operational Database*.

Related Information

Scaling Cloudera Operational Database instances vertically

Creating a database using Cloudera Operational Database

March 10, 2023

Cloudera Operational Database 1.28 version provides UI enhancements that include storage type selection when you create a database and JWT configurations to connect to your HBase client.

Cloudera Operational Database UI allows storage type selection when creating an operational database

Cloudera Operational Database UI now allows you to select the storage type when creating an operational database. You can either select Cloud Storage or HDFS.

Earlier if you had to use the HDFS as the storage type, you were required to use the --use-hdfs option on CDP CLI while creating the operational database. Now Cloudera Operational Database UI is enhanced to let you select the storage type when you create an operational database.

For more information, see *Creating a database using Cloudera Operational Database*.

Cloudera Operational Database UI provides the JWT configuration details to connect to your HBase client

Now you can find the JWT configuration details On the Cloudera Operational Database UI. Click on a database and go to Connect HBase Client Tarball JWT Configuration . You can refer to these configurations to set up a connection to HBase with a JWT token and build your own truststore JKS file.

For more information, see *Configuring JWT authentication for HBase client*.

Related Information

Creating a database using Cloudera Operational Database

Configuring JWT authentication for HBase client

January 10, 2023

Cloudera Operational Database 1.27 version supports JWT authentication, provides Data Lake templates while creating a database, and a CLI option to enable HBase region canaries.

Cloudera Operational Database supports configuring JWT authentication for your HBase clients

Cloudera Operational Database now allows you to configure JWT (JSON Web Token)-based authentication for your HBase clients, which uses an unique identifier and is a standard way of securely transmitting signed information between two parties. To know more about the JWT authentication, see *Configuring JWT authentication for HBase client*.

Cloudera Operational Database supports creating an operational database using a predefined Data Lake template

When you create an operational database, you can now define the structure of your database based on a predefined Data Lake template. The template defines the number of gateway, master, and worker nodes to be added while creating a database.

You can select a template and accordingly the nodes are added into the Cloudera Operational Database cluster after the database is successfully created.

To know more about this, see *Creating a database using Cloudera Operational Database*.

Cloudera Operational Database provides a CLI option to enable HBase region canaries

Cloudera Operational Database now provides a CLI option, --enable-region-canary to enable the HBase region canaries while creating an operational database.

Use the following command to enable the HBase region canaries.

```
cdp opdb create-database --environment-name ENVIRONMENT_NAME --database-name
 DATABASE_NAME --enable-region-canary
```

- hbase_region_health_canary_enabled
- hbase_region_health_canary_slow_run_alert_enabled
- hbase_canary_alert_unhealthy_region_percent_threshold

For more information, see *Enabling HBase region canary*.

Related Information

Configuring JWT authentication for HBase client

Creating a database using Cloudera Operational Database

Enabling HBase region canary

December 10, 2022

Cloudera Operational Database 1.26 version supports managing the edge nodes using CDP CLI and disabling the Kerberos authentication while creating an operational database using CDP CLI.

Cloudera Operational Database supports creating edge nodes while creating an operational database

Cloudera Operational Database now allows you to create edge nodes while creating an operational database. You can define the number of edge nodes to be created, and Cloudera Operational Database automatically creates and configures the nodes in your Cloudera Operational Database cluster.

You can also add additional nodes into your Cloudera Operational Database cluster or delete an existing one using the edge node instance ID.

For more information, see *Managing edge nodes*.

Cloudera Operational Database supports disabling the Kerberos authentication while creating an operational database using CDP CLI

Cloudera Operational Database now allows you to disable the Kerberos authentication while creating an operational database using CDP CLI. You can use the --disable-kerberos option while running the `create-database` command to disable the Kerberos authentication.

`cdp opdb create-database --environment-name` *ENVIRONMENT_NAME* `--database-name` *DATABASE_NAME* `--disable-kerberos`

Related Information

Managing edge nodes

November 10, 2022

Cloudera Operational Database 1.25 version supports creating and updating an operational database using a custom image.

Cloudera Operational Database supports custom images for deploying Cloudera Operational Database clusters

Cloudera Operational Database now allows you to create or update a database using a custom image. Custom images can be used for various purposes, such as compliance or security requirements. An image catalog is used to hold one or more custom images. You can inherit pre-installed packages or software from the custom image while creating or updating an operational database.

You can also switch an image catalog of an existing operational database. For more information, see *Managing custom images in Cloudera Operational Database*.

Related Information

Managing custom images in Cloudera Operational Database

October 10, 2022

Cloudera Operational Database 1.24 version supports fast SSD volume types, deploys strong meta servers for multiple regions, and provides two new CDL CLI commands.

Cloudera Operational Database supports fast SSD based volume types for gateway nodes of HEAVY types

Cloudera Manager and monitoring systems require more resources than a regular deployment for big clusters with a large number of nodes. To support this, now Cloudera Operational Database supports a fast gp2 storage when you choose a gateway node of HEAVY type.

Cloudera Operational Database deploys strong meta servers for multiple regions for Multi-AZ

For Multi-AZ deployments, Cloudera Operational Database assigns the nodes to multiple regions when multiple strong meta servers exist in the deployment.

Cloudera Operational Database provides CDP CLI commands to set the HBase configuration values

Cloudera Operational Database now supports the CDP CLI commands `update-hbase-configuration` and `describe-hbase-configuration` to update and retrieve the HBase configuration values. For more information, see *describe-hbase-configuration* and *update-hbase-configuration*.

Related Information

describe-hbase-configuration

update-hbase-configuration

August 10, 2022

Cloudera Operational Database 1.23 version supports custom EBS volumes for HDFS clusters while creating a database and displays UI notifications on the Cloudera Operational Database UI.

Cloudera Operational Database supports custom EBS volumes for HDFS clusters

Cloudera Operational Database allows you to customize the attached storage worker node for an HDFS cluster using the --attached-storage-for-workers option while creating a database. You can define the EBS volumes for HDFS clusters.

You can also view the customized attached storage worker nodes using the `describe-database` and `list-database` commands. For more information, see create-database.

Cloudera Operational Database displays notifications on the Cloudera Operational Database UI for all the major events

Cloudera Operational Database now displays notifications about all the major events on the Cloudera Operational Database UI. For example, auto-scaling events, auto-healing events, and alerting notifications.

The UI notifications is a major step in providing enough information on Cloudera Operational Database UI so that you do not need to navigate to multiple internal systems used by Cloudera Operational Database to get the required information as well as be aware of the events happening in the system on an immediate basis.

June 10, 2022

Cloudera Operational Database 1.22 version supports HBase Store File Tracking, Multiple Availability zones on AWS, and Cloudera Control Plane for multiple regions.

Cloudera Operational Database now supports the Store File Tracking (SFT) as a limited availability feature

Store File Tracking (SFT) defines a separate, pluggable layer to handle storefile life cycle, and includes the FILE based built-in implementation that avoids internal file rename or move operations while managing the storefiles. This is a critical enablement to deploy HBase over S3 object store, which is known for the lack of atomic renames. Cloudera Operational Database enables this feature by default for databases deployed on AWS with S3, to mitigate the aforementioned S3 limitation that could cause critical issues for HBase. For more information, see HBase Store File Tracking.

Cloudera Operational Database supports Multiple Availability Zones (Multi-AZ) on AWS

Cloudera Operational Database ensures high availability and fault tolerance using Multi-AZ deployments. A Multi-AZ deployment means that compute infrastructure for HBase's master and region servers are distributed across multiple AZs ensuring that when a single availability zone has an outage, only a portion of Region Servers is impacted and clients automatically switch over to the remaining servers in the available AZs.

Multi-AZ for Cloudera Operational Database is currently supported only on Amazon Web Services (AWS) environments. For more information, see Multi-AZ deployment on Cloudera Operational Database.

Cloudera Operational Database supports Cloudera Control Plane for multiple regions

Cloudera Operational Database now supports Cloudera Control Plane for eu-1 (Germany) and ap-1 (Australia) regions. Certain countries may have regulatory requirements that limit or forbid specific types of data from being sent or stored outside of the country or region where an organization operates. For this reason Cloudera has introduced new regions in which the Control Plane can run. By choosing a region other than us-west-1, you can ensure that sensitive metadata (such as information about Cloudera users) does not leave its region or country of origin. For more information, see Cloudera Control Plane regions.

April 10, 2022

Cloudera Operational Database 1.21 version supports enabling the replications for META regions, by default.

Cloudera Operational Database now supports replications for META regions enabled by default

Cloudera Operational Database now supports replication for the META regions, which means clients can read the META replicas first before connecting to the primary region server. This reduces the load on the META table because the read META load is distributed across multiple replicas.

Cloudera Operational Database supports configuring three META replicas which can be deployed in different RegionServers. Client needs to use the Cloudera Operational Database provided configuration to get the META calls that are distributed across these replicas. This is useful when an application has multiple clients accessing spark, yarn, or phoenix-spark.

March 10, 2022

Cloudera Operational Database 1.20 version provides UI enhancements to download the Phoenix client jars. Refer to the following section for more details.

Cloudera Operational Database provides enhanced user interface to download the Phoenix client jar

Cloudera Operational Database now provides an enhanced user interface to download the Phoenix client jar from the Phoenix Thick and Phoenix Thin client tabs through a single click.

You can now download the Phoenix client jars with a single click directly from the Phoenix Thick client and Phoenix Thin client tabs in the UI.

February 10, 2022

Cloudera Operational Database 1.19 version supports Spark transactional tables using Apache OMID and is also bundled with the HBase version 2.4.6. Refer to the detailed section for more features.

Cloudera Operational Database through Phoenix-Spark connector supports Spark transactional tables using Apache OMID

Cloudera Operational Database supports Apache OMID transactional framework that allows Big Data applications to execute ACID transactions on top of Phoenix tables.

The transaction support in Cloudera Operational Database enables you to perform complex distributed transactions and run atomic database operations, meaning your database operations must either be completed or terminated. A transaction ensures adhering to the ACID properties.

Cloudera Operational Database is now bundled with the HBase version 2.4.6

Cloudera Operational Database is now bundled and shipped along with the HBase version 2.4.6 when the Cloudera Runtime version is 7.2.14.

For a smooth and better functionality, Cloudera Operational Database is now bundled with the HBase version 2.4.6. You need to upgrade the HBase client version for seamless connectivity.

Cloudera Operational Database supports custom table coprocessors

Cloudera Operational Database supports custom table coprocessors, which you can implement and extend from HBase coprocessors' interfaces.

You can add table coprocessors so that HBase can run custom code on the server side against the stored data and filter local minimum or maximum value during ingestion without scanning the entire table. You can use built-in table coprocessors from the upstream HBase releases. For more information, see Custom Table Coprocessors in Cloudera Operational Database.

Cloudera Operational Database supports RAZ integration from the Runtime version 7.2.11.0

Cloudera Operational Database supports RAZ integration from the Runtime version 7.2.11.0. You can grant fine-grained access to directories.

The Ranger Authorization Service (RAZ) is a fine grained authorization service for cloud storage. As a regular individual user or as an HBase user, you can limit the authorization levels in the cloud storage to a directory level. For more information, see Cloudera Operational Database integration with RAZ.

Storefile Tracking (SFT) is available as an optional feature delivered through the Cloudera Operational Database service

Cloudera Operational Database now supports the "Storefile Tracking" (SFT) as an optional feature in Runtime 7.2.14.0.

Storefile Tracking (SFT) changes how HBase manages its files to avoid operations which are known to be suboptimal when using object stores. Cloudera Operational Database enables this feature for Cloudera Operational Database databases deployed on AWS which use S3 for HBase storage which will address performance issues known around flushes, compactions, and other HBase operations. For more information, see HBase Storefile Tracking.

Cloudera Operational Database allows to disable the Kerberos authentication temporarily for HBase clients

Cloudera Operational Database allows to disable the Kerberos authentication temporarily for HBase clients that run on Cloudera legacy products.

If your client applications are running on Cloudera legacy products, they usually do not have Kerberos authentication enabled. When you try to connect to any Cloudera Operational Database instance, the connection fails because the Cloudera Operational Database instances have Kerberos enabled, by default. Now, you can disable Kerberos authentication in your Cloudera Operational Database instances so that HBase or Phoenix clients can connect seamlessly. For more information, see Disabling Kerberos authentication for HBase clientsDisabling Kerberos authentication for HBase clients.

December 10, 2021

Cloudera Operational Database 1.18 version supports ephemeral storage on Azure.

Cloudera Operational Database supports ephemeral storage on Azure

Cloudera Operational Database now supports the configuration of instance storage to cache HBase data stored in block storage. This is only available on AMD instance types.

November 10, 2021

Cloudera Operational Database 1.17 version is now available as a technical preview on GCP (Google Cloud Platform) and with various performance improvements.

Cloudera Operational Database is available as a Technical Preview feature on GCP

You can now deploy Cloudera Operational Database on GCP easily similar to what is available for Amazon Web Services (AWS) and Microsoft Azure.

Cloudera Operational Database automatically improves the performance by 80% when you use AWS S3

Cloudera Operational Database now delivers a better performance in S3 because the data loading behaviour from S3 into cache is tuned. This improvement minimizes the cost associated with the S3's high latency to read data.

Cloudera Operational Database improvises scalability when using block storage on AWS

Cloudera Operational Database now uses larger EBS volumes for the underlying master nodes to provide better scalability.

October 10, 2021

Cloudera Operational Database 1.16 version supports modified auto-scaling criteria and a built-in coprocessor *AGGREGATEIMPLEMENTATION*.

Cloudera Operational Database supports modified auto-scaling criteria

Cloudera Operational Database now supports an improved auto-scaling algorithm that considers the latency of the user operations. Cloudera Operational Database now prioritises user operations over system operations that results in a reduced cost of infrastructure with a minor increase in replication.

Cloudera Operational Database supports a built-in coprocessor *AGGREGATEIMPLEMENTATION*

Cloudera Operational Database now supports a built-in coprocessor called *AGGREGATEIMPLEMENTATION* that facilitates aggregation function computations (min, max, sum,avg, median, std) at the region level. This yields better performance because you need not get all the data to perform these calculations. Cloudera Operational Database enables *AGGREGATEIMPLEMENTATION* by default, and you can use the *AGGREGATIONCLIENT* service in HBase to perform RegionServer side aggregation. For example, row count.

September 10, 2021

Cloudera Operational Database 1.15 version provides a new option for the upgrade-database command and supports DataHub/OpDB deployment across multiple availability zones.

The upgrade-database command supports a new option to upgrade only the operating system

Cloudera Operational Database provides a new option `--os-upgrade-only` for the `upgrade-database` command, which you can use to upgrade only the operating system to the latest supported version addressing any CVEs that were patched in the operating system. Running the `upgrade-database` command with this option does not have an effect on the Cloudera runtime version running on the given cluster.

Cloudera Operational Database supports DataHub/OpDB deployment across multiple availability zones (Multi-AZ) to ensure high availability

- Cloudera Operational Database introduces a technical preview version of Multi-AZ deployment capability and is available by special request through your account team.
- When you enable Cloudera Operational Database, by default it deploys all the databases across three availability zones identified in the Cloudera Shared Data Experience by the provided network subnets. CDP CLI beta provides a new `--disable-multi-az` to limit deployments to a single availability zone.

Cloudera Operational Database can disable ephemeral storage in CLI

Cloudera Operational Database provides the ability to disable the ephemeral storage using a new command `--disable-ephemeral-storage`.

Cloudera Operational Database provides the ability to list all the HBase snapshots using different filters

Cloudera Operational Database supports listing all the HBase snapshots created within a database of an environment using these filters, time range, snapshot name, and command ID. For example, you can use the time range filter to obtain the snapshots created within a particular time, the command ID filter to know the status of a snapshot operation.

August 10, 2021

Cloudera Operational Database 1.14 version offers multiple new features that includes disengaging autoadmin database function, runtime upgrade, copy tables between Cloudera Operational Database environments and many more.

Disengage autoadmin database function

Cloudera Operational Database can disable the autonomous database functions thereby stops managing a database. When Cloudera Operational Database disables the autonomous functions, the database cannot leverage it; however the functions are available and accessible as a datahub cluster. Once you disable the autonomous database functions, it is irreversible.

Cloudera Runtime upgrade with downtime

You can upgrade the Cloudera Runtime installed on the cluster running Cloudera Operational Database to an advanced one with downtime.

Copy tables between two Cloudera Operational Database environments

Cloudera Operational Database provides a CopyTable utility to copy tables from one Cloudera Operational Database cluster to another.

Cloudera Operational Database supports ephemeral storage on AWS

Cloudera Operational Database now supports ephemeral storage on AWS for HBase buckets.

Cloudera Operational Database provides HDFS as a storage option

Cloudera Operational Database now provides HDFS as a storage option for Cloudera Operational Database deployment instead of cloud storage.

Cloudera Operational Database uses instance storage for HBase bucket cache

Cloudera Operational Database now deploys clusters with large block cache on ephemeral storage thereby enhancing the performance of the clusters. This is available only for AWS-based clusters with S3 blob storage.

July 10, 2021

Cloudera Operational Database 1.13 version offers multiple new features that includes disengaging autoadmin database function, runtime upgrade, copy tables between Cloudera Operational Database environments and many more.

Cloudera Operational Database supports HDFS for storage on Cloud Block Storage in addition to Cloud Object Storage on AWS & Azure

1. Cloudera Operational Database now supports HDFS for fast and consistent performance.
2. Cloud Object Storage has a high variance in latency, and sometimes the delay in accessing the data is high. The operational cost in managing such an operational database is expensive, as it reads millions of data per second. Cloudera Operational Database now deploys HDFS on Cloud Block Storage, which ensures that the performance characteristics are similar to on-prem. This feature simplifies the lift-and-shift efforts to the cloud.
3. This capability includes an updated `update-database` command in CDP CLI Beta to enable tuning of auto-scaling based on HDFS utilization.

Cloudera Operational Database promotes custom user tags to CDP CLI Beta

When you create a Cloudera Operational Database database, Cloudera Operational Database enables you to supply custom tags for launched hardwares while using CDP CLI Beta.

Cloudera Operational Database delivers utility to manually copy tables across Kerberos realms without cross-realm Kerberos trust

Cloudera Operational Database introduces a CldrCopyTable utility, which is Cloudera's version of the upstream CopyTable utility. This utility is an extension of the Cloudera Operational Database replication plugin that enables you to copy data cross-realm. This utility enables you to close any replication gaps that might arise depending on how you start or stop replication.

Cloudera Operational Database streamlines on-prem replication plugins

When you create a Cloudera Operational Database database, Cloudera Operational Database automatically sets configuration properties for the Cloudera replication plugin so that on-prem clusters can replicate into Cloudera Operational Database databases. Cloudera Operational Database now automatically configures the OpDB replication plugin and reduces the time needed to move the application into production.

May 10, 2021

Cloudera Operational Database 1.11 version now supports Phoenix through Hue web UI.

Cloudera supports Phoenix when using Hue web UI

Cloudera now supports Phoenix access through the Apache Hue SQL assistant. You can now access Phoenix and query the data from the Hue web UI out of the box and leverage Hue's rich SQL support (for example, type ahead) to query Phoenix for operational insights or developer testing of SQL statements.

February 10, 2021

Cloudera Operational Database now supports deploying a Cloudera Data Hub cluster as an edge node.

Cloudera Operational Database now supports deploying a Cloudera Data Hub cluster as an edge node

You can deploy a Cloudera Data Hub cluster that works as an edge node to access your Cloudera Operational Database instance. Use the cluster template [***RUNTIME VERSION***] Cloudera Operational Database Edge Node cluster template. For example, 7.2.7 COD Edge Node. For more information see Configure Cloudera Data Hub edge node.

November 10, 2020

Cloudera Operational Database is now available as a technical preview.

Cloudera Operational Database is available as a technical preview

Cloudera Operational Database is now available as a technical preview and is considered under development. For an overview about this service, see Cloudera Operational Database service in the public cloud.

Important: Do not use this in your production systems. If you have feedback, contact Support by logging a case on the Cloudera Support Portal at https://my.cloudera.com/support.html. Technical preview services and features are not guaranteed troubleshooting and fixes.

Fixed issues

Learn about the fixed issues and the enhancements in Cloudera Operational Database.

COD-3836 (Cloudera Operational Database 1.41) : Cloudera Operational Database supports deployments to specific subnets in Azure and GCP environments

When you create an operational database using the CDP CLI, the subnet-id parameter in the `create-database` command is fixed and can be used to deploy the cluster into specific subnets in Azure and GCP environments.

Following is an example of subnet-id.

```
cdp opdb create-database --environment-name <env> --database-name <db> --sub
net-id DataLakeHubSubnet1
```

For more information, see *CDP CLI documentation*.

ODX-2928 (Cloudera Operational Database 1.31) : Minor upgrade is not available

Minor upgrades are available now while using the `describe-upgrade-database` CDP CLI command. You can now list the available hotfix versions using the `describe-upgrade-database` CDP CLI command.

ODX-1716 (Cloudera Operational Database 1.21) : Increase storage density for HDFS form factor

Storage density for HDFS form factor is increased. Now, Cloudera Operational Database deploys 8 TB of disks for each worker instead of 6 TB during any new HDFS deployment.

ODX-1811 (Cloudera Operational Database 1.21) : Increase handler count

The number of RegionServer handlers is increased to process 50% more read or write requests.

Related Information

CDP CLI documentation

Known issues

Learn about the known issues in the Cloudera Operational Database, the impact or changes to the functionality, and the workaround.

COD-3987 (Cloudera Operational Database 1.46): LB based client configs and tarball URLs in the describe-client-connectivity command output fails intermittently

Problem: Due to a known Cloudbreak issue, CB-26030, downloading an HBase client configuration intermittently fails when using multiple Gateway nodes

Workaround: Retry the download operation until it succeeds.

COD-3611 (Cloudera Operational Database 1.38): [Azure] Import TSV job during AZ outage fails to write data to the HBase table

Problem: Cloudera Operational Database is unable to write data on an HBase table and also fails to import a TSV file during a Multi-AZ outage on an Azure environment.

Workaround: None

ODX-1603 (Cloudera Operational Database 1.22) : Cloudera Operational Database throws a "Node Failure" issue

Problem: Cloudera Operational Database reports "Node Failure" state whenever OMID service goes down and retains the state until the OMID service is restarted from Cloudera Manager (CM).

Workaround:

- If you are using OMID, restart the OMID service through CM and wait for a few minutes until Cloudera Operational Database starts displaying Available again.
- If you are not using OMID service and your Cloudera Operational Database status is not affected by the OMID service status, you can suppress the OMID alerts directly in the CM.

Technical Service Bulletins

TSB 2022-568: HBase normalizer must be disabled for Salted Phoenix tables

When Apache Phoenix (“Phoenix”) creates a salted table, it pre-splits the table according to the number of salt regions. These regions must always be kept separate, otherwise Phoenix does not work correctly.

The HBase normalizer is not aware of this requirement, and in some cases the pre-split regions are merged automatically. This causes failure in Phoenix.

The same requirement applies when merging regions of salted tables manually: regions containing different salt keys (the first byte of the rowkey) must never be merged.

Note that either automatic or manual splitting of the regions for a salted table does not cause a problem. The problem only occurs when adjacent regions containing different salt keys are merged.

Upstream JIRA

PHOENIX-4906

Knowledge article

For the latest update on this issue, see the corresponding Knowledge article: TSB 2022-568: Hbase normalizer must be disabled for Salted Phoenix tables

Cloudera Operational Database ..

Database Application Access Examples

Date published: 2020-08-14
Date modified: 2023-01-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Connect your Go applications to Cloudera Operational Database.....4

Connect your Python applications to Cloudera Operational Database.....5

Connect your Go applications to Cloudera Operational Database

You can connect your applications written in the Go programming language to a Cloudera Operational Database through Phoenix.

Go application to connect to Cloudera Operational Database

Before you write an application to connect to Cloudera Operational Database, do the following:

- Create a Cloudera Operational Database instance
- Get the Phoenix (Thin) client connectivity information
- Set your Cloudera wokload password

This application code is an example for connecting to a Cloudera Operational Database instance, creating a table, inserting a row, and then reading from the Cloudera Operational Database instance.

```
package main

import (
 "database/sql"
 "log"

 _ "github.com/apache/calcite-avatica-go"
)

func main() {
 // Connections are defined by a DSN
 // The format is http://address:port[/schema][?parameter1=value&...param
eterN=value]
 // For Cloudera Operational Database, BASIC authentication is used.
 // The workload username and password are passed as parameters avaticaUser
and avaticaPassword
 //
 // For example:
 // Cloudera Operational Database URL: 'https://gateway.env.cloudera.site/
cdp-proxy-api/avatica/'
 // Workload username: jgoodson
 // Workload password: Secret1!
 // Would result in this DSN:
 dsn := "https://gateway.env.cloudera.site/cdp-proxy-api/avatica/?&authen
tication=BASIC&avaticaUser=jgoodson&avaticaPassword=Secret1!"

 log.Println("Connecting...")
 db, err := sql.Open("avatica", dsn)
 if err != nil {
  log.Fatal("Connection: ", err)
 }
 defer db.Close()

 log.Println("Create table if not exists...")
 _, err = db.Exec("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY,
 username VARCHAR)")
 if err != nil {
  log.Fatal("Create: ", err)
 }

 log.Println("Insert a row...")
 _, err = db.Exec("UPSERT INTO users VALUES (?, ?)", 1, "admin")
```

```
 if err != nil {
  log.Println("Insert: ", err)
 }

 log.Println("Reading and printing rows...")
 var (
  id       int
  username string
 )
 rows, err := db.Query("SELECT id, username from users")
 if err != nil {
  log.Fatal("Query: ", err)
 }
 defer rows.Close()
 for rows.Next() {
  err := rows.Scan(&id, &username)
  if err != nil {
   log.Fatal(err)
  }
  log.Println(id, username)
 }
}
```

Related Information

How to connect Go Applications to Cloudera Operational Database

Connect your Python applications to Cloudera Operational Database

You can connect your applications written in Python programming language to a Cloudera Operational Database through Phoenix.

Python application to connect to Cloudera Operational Database

Before you write an application to connect to Cloudera Operational Database, do the following:

- Create a Cloudera Operational Database instance
- Get the Phoenix (Thin) client connectivity information
- Set your Cloudera wokload password

This application code is an example of validating your connection to Cloudera Operational Database.

```
import phoenixdb
import configparser

class Database:

    def connect(self):
        REQUIRED_OPTS = ['Username', 'Password', 'Url']
        config = configparser.ConfigParser()
        config.read('config.ini')
        if not 'COD' in config:
            raise Exception("Could not find section for COD in config.ini")
        cod_config = config['COD']
        opts = {}

        # Validate the configuration
        for required_opt in REQUIRED_OPTS:
            if not required_opt in cod_config:
```

```
                raise Exception("Did not find %s in configuration" % (requ
ired_opt))

      # Provide non-required options
        if 'Truststore' in cod_config:
            opts['verify'] = cod_config['Truststore']
        if 'Authentication' in cod_config:
            opts['authentication'] = cod_config['Authentication']
        else:
            opts['authentication'] = 'BASIC'
        # Read required options
        opts['avatica_user'] = cod_config['Username']
        opts['avatica_password'] = cod_config['Password']
        return phoenixdb.connect(cod_config['Url'], autocommit=True, **opts)
```

This code is an example of creating a table and adding data to a Cloudera Operational Database instance.

```
import phoenixdb
from database import Database

def load():
    db = Database()
    conn = db.connect()
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY
, username VARCHAR)")
    cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin'))
    cursor.execute("SELECT * FROM users")
    print(cursor.fetchall())
    conn.close()

if __name__ == '__main__':
    load()
```

Related Information

Building a Simple CRUD web application and image store using Cloudera Operational Database and Flask

Cloudera Operational Database Top Tasks

Date published: 2022-09-27
Date modified: 2023-01-12

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating a database using Cloudera Operational Database..... 4
Creating a database using CDP CLI.....6

Deploying applications on Cloudera Operational Database.....6

Importing and restoring data into Cloudera Operational Database database.....7

Example: run an application for the Apache HBase client.....9
Example: run an application for the Apache Phoenix thick client.....10
Example to run application using the Phoenix thin client.....10

Compile an application for your Cloudera Operational Database.....11

Creating a database using Cloudera Operational Database

You can create an operational database in your registered Cloudera environment using the Cloudera Operational Database.

About this task

Required role: You must be logged into the Cloudera Operational Database as an ODAdmin.

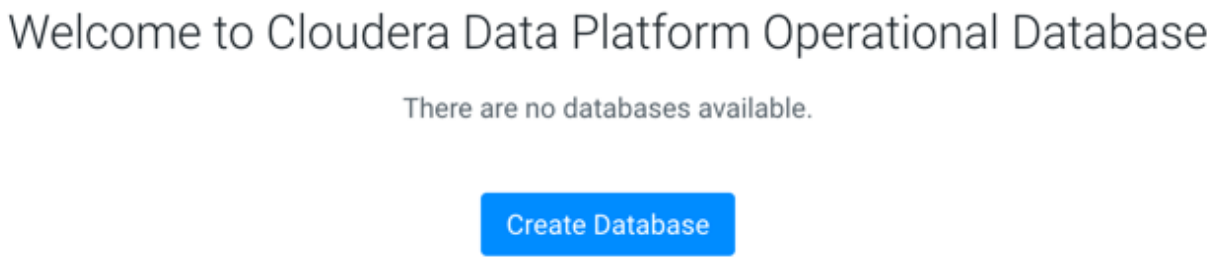

Before you begin

- Understand the Cloudera environment and user management. For more information, see *User Management* and *Cloudera Environments* topics.
- Set up an environment that gives you credential and cloud storage. For more information, see *Before you create an operational database cluster*.
- Ensure that you are authorized to create a database.

Procedure

1. In the Cloudera Operational Database web interface, click Create Database.
2. Specify the location of the database where you want to store it.
   a) Provide a name for the database in the Database Name field.
   b) Select the Cloudera environment from the list in which you want to associate the database.
   c) Click Next.

      If an environment does not exist, you can create one by clicking Create New Environment.

      For more information, see *Register your first environment*.
3. Commission your database by defining a scale for your database using a predefined Data Lake template.

   The template helps you to structure your database automatically thereby saving your time and cost. Cloudera Operational Database creates the predefined number of LITE or HEAVY gateway and master nodes, a set of worker nodes, and also adds additional functionalists into the new database. In case you need to modify the default number of nodes defined in the template, you can do so after the database creation.

   The available templates are Micro Duty, Light Duty, and Heavy Duty. By default, Light Duty is selected.

   You can create a small database using the Micro Duty template, which consists of one Gateway node and one Worker node. In a Micro database, the Gateway node carries out the processes involved in the Master or Leader nodes. You can consider using a Micro cluster for your testing and development purposes.

4. Configure your database by selecting the storage type as Cloud Storage with Caching, Cloud Storage with Caching and Data Tiering, Cloud Storage, or HDFS.
   - The storage type Cloud Storage with Caching is equivalent to using --storage-type CLOUD_WITH_EPHEMERAL option on CDP CLI while creating an operational database.
   - The storage type Cloud Storage with Caching and Data Tiering resembles cloud storage with time-based priority caching, where data within a specified time range is given a higher priority. In contrast, older data are likely to get evicted. For more information on this storage type, see *HBase Time-based Data Tiering using Persistent BucketCache*.

     You must have the COD_DATATIERING entitlement to be able to use this storage type.
   - The storage type Cloud Storage, which resembles block storage, is equivalent to using --storage-type CLOUD option on CDP CLI while creating an operational database.
   - The storage type HDFS is equivalent to using --storage-type HDFS option on CDP CLI while creating an operational database.

   By default, Cloud Storage with Caching is selected.
5. Check or update the settings for your database.
   a) Check all the default settings for your database under the Default tab.
   b) Go to the Advanced tab if you need to modify any of the default values.
      - The HDFS Volume Type option appears under the Advanced tab only if you select HDFS as the storage type in the Configuration step.
      - If you disable the Autoscaling option using the Advanced tab, the Worker Nodes and Compute Nodes options are hidden. Instead, a Node Count option appears.

      The minimum and maximum number of worker nodes vary for different storage types.
      - Micro duty: Minimum node count: 1. Maximum node count: 5.
      - Light duty: Minimum node count: 3. Maximum node count: 100.
      - Heavy duty: Minimum node count: 3. Maximum node count: 800.
6. Review the details before creating the database.

   Click Show CLI Command to get the complete command details corresponding to your settings. You can use it to create the database using CDP CLI.

   Alternatively, you can use the following sample command to create the database using CDP CLI.

```
cdp opdb create-database --environment-name cod-7218 --database-name tes
t --scale-type LIGHT --storage-type CLOUD_WITH_EPHEMERAL --auto-scaling-
parameters '{"minWorkersForDatabase":5, "maxWorkersForDatabase":100}' --
num-edge-nodes 0 --java-version 8
```

7. Click Create Database.

Results

An information page is displayed that shows the status of the database. Your new database is ready to be used once its status becomes Available.

Note: Your database starts with a fixed size; however, it scales up and down as the workload applied to the database changes. For more information, see the *Auto Scaling* topic.

Related Information

Cloudera Operational Database Edge Node Overview
Cloudera Operational Database User Management
Cloudera Environments
Cloudera Operational Database Auto Scaling
Cloudera Operational Database CLI command reference GitHub repository
Cloudera CLI BETA command reference GitHub repository

Creating a database using CDP CLI

You can create an operational database in your registered environment using CDP CLI Beta.

Before you begin

- You must download and install the latest CDP CLI beta version.
- Understand Cloudera environment and user management. For more information, see *User Management* and *Cloudera Environments* topics.
- Set up an environment that gives you credential and cloud storage.
- Ensure you are authorized to create a database.

About this task

- Required role: You must be logged into the Cloudera Operational Database as an ODAdmin.
- You can create a Cloudera Operational Database in an AWS, GCP, or Azure environment.

Procedure

1. Log in to the CDP CLI beta tool.
2. Enter the following command:

   cdp opdb create-database --environment-name cod7215 --database-name test123

   Pass the --use-hdfs flag to create a database for Cloudera Operational Database using HDFS storage.

   If you pass --no-use-hdfs flag or do not pass any flag, the database for Cloudera Operational Database is created using cloud storage. For example, Amazon S3, GCS, or ABFS.

Results

Your new database is ready for use.

Related Information

Cloudera Operational Database User Management

Cloudera Environments

Deploying applications on Cloudera Operational Database

The edge node is a dedicated Cloudera Data Hub cluster that enables you to communicate with your Cloudera Operational Database instance and your applications. You can deploy a cluster that works as an edge node to access your Cloudera Operational Database instance. Deploy the edge node cluster in the same environment as the Cloudera Operational Database instance to ensure that the security groups and data ingress rules that apply to the Cloudera Operational Database instance must also apply to the edge node cluster.

Procedure

1. From the Cloudera Management Console, click Data Hub Clusters.
2. Click Create Data Hub.
3. In the Selected Environment with running Data Lake drop-down list, select the same environment used by your Cloudera Operational Database instance.
4. Select the Cluster Definition.

5. In the Cluster Definition drop-down list, select the [***RUNTIME VERSION***] COD Edge Node for [***CLOUD PROVIDER NAME***].

   For example, select the 7.2.10 COD Edge Node for AWS cluster template.

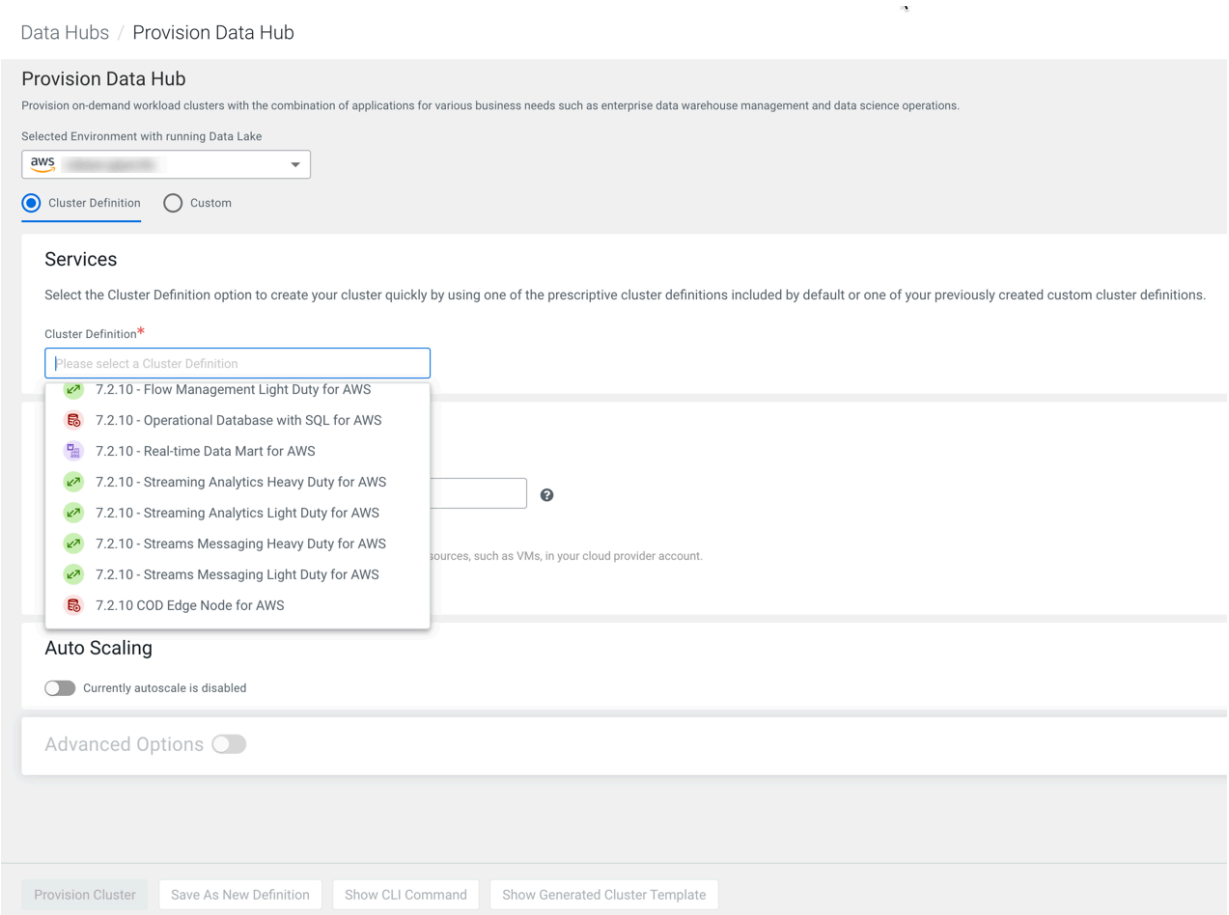

6. In the Cluster Name field, provide a cluster name that you can identify later as an edge node of a specific Cloudera Operational Database instance.
7. Click Provision Cluster.

What to do next

After you deploy the edge node, you can run your applications on this edge node using the Client connectivity information. See how to compile applications for Cloudera Operational Database in Compile an application for your database.

Importing and restoring data into Cloudera Operational Database database

You can import your data into your Cloudera Operational Database by restoring your HBase table into Cloudera Operational Database.

Before you begin

- Enable HBase replication on your Cloudera Operational Database cluster. For more information, see *Cloudera Operational Database data replication*.

- Have a location in cloud storage (for example, S3 or ABFS) with an exported snapshot in it, and have the name of the snapshot.

  If you do not already have an exported HBase snapshot, you can export your data to cloud storage using the following command:

  ```
  hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snaps
  hot [***SNAPSHOT NAME***] -copy-to [***CLOUD STORAGE LOCATION***] -mappers
   10
  ```

  For example, the data-from-onprem snapshot can be exported into s3a://cod-external-bucket/hbase:

  ```
  hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot data-from-
  onprem -copy-to s3a://cod-external-bucket/hbase -mappers 10
  ```

- Have and edge node with a configured HBase client tarball and know how to launch the hbase shell from it. For more information, see *Launching HBase shell*.

Procedure

1. Get your CLOUD STORAGE LOCATION for your Cloudera Operational Database using the Cloudera Operational Database web interface.
   s3a://my_cod_bucket/cod-12345/hbase

   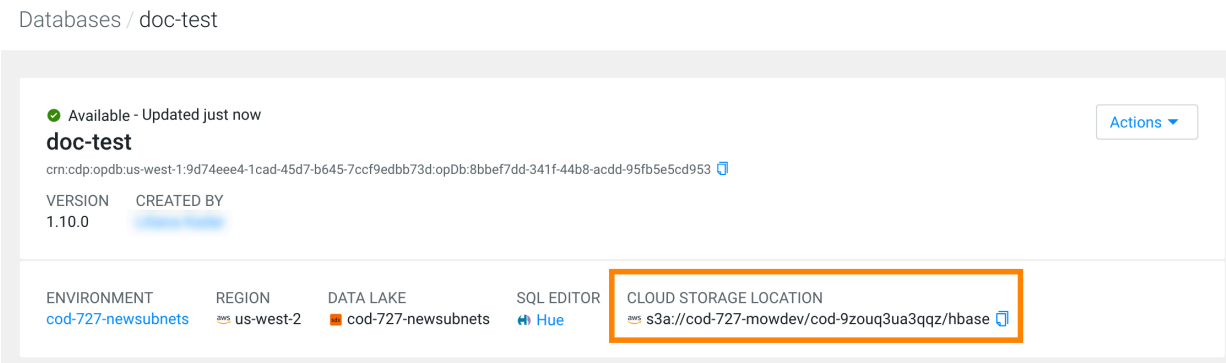

2. Add your bucket to the IAM policy used by IDBroker.

   For more information, see one of the following documentation:

   - AWS Environments: Minimal setup for cloud storage
   - Azure Environment: Minimal setup for cloud storage

3. Launch the hbase shell from the edge node.

   For more information, see *Launching HBase shell*.

4. From the edge node, run the ExportSnapshot command. Use the external bucket as the source location and the Cloudera Operational Database cloud storage as the target.

   For example:

   ```
   $ cd $HBASE_HOME
   $ ./bin/hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot
    “data-from-onprem” --copy-from s3a://cod-external-bucket/hbase --copy-to
    s3a://my_cod_bucket/cod-12345/hbase
   ```

5. Use the list_snapshots command and verify that your snapshot is listed.

   ```
   $ cd $HBASE_HOME
   $ ./bin/hbase shell
   $ hbase> list_snapshots
   [‘snapshot_name’]
   ```

6. Use the restore_snapshot or the clone_snapshot command to reconstitute the table.

   Warning: The restore_snapshot command overwrites an existing table.

   - restore_snapshot: Overwrites current table state with that of the snapshot. It means that any data modification applied after the snapshot was taken would be lost. If the table does not exist in the given cluster, the command automatically creates it.
   - clone_snapshot: Accepts a new table name for the table in which it restores the table schema and data.
7. Validate that all rows are present in the table using the hbase rowcounter or count command in the hbase shell.

Related Information

Launching hbase shell
Cloudera Operational Database data replication
Using the CldrCopyTable utility to copy data

Example: run an application for the Apache HBase client

Checking the example of how to run a Maven application for the HBase client gives you better understanding about how to run your own application for the HBase client.

Before you begin

The requiredKerberos option is set to true for the Apache HBase client. The application must be run for this client from a computer which:

- Has internal network access to the public cloud in which the database is deployed
- Can resolve the internal hostnames of the database
- Can obtain a Kerberos ticket from the database's Kerberos Key Distribution Center (KDC)

One way to run the Apache HBase client applications is to launch an edge node in your cloud provider which meets the above requirements.

Ensure that you download the directory containing the Apache HBase client configuration files from clientConfig urationUrl provided in describe-client-connectivity response. This is a protected endpoint, and you have to use your Cloudera workload credentials to access this endpoint. You can also get the Apache HBase client configuration as a client tarball from the Cloudera Operational Database web user interface Connect tab.

Procedure

1. Use clientConfigurationURL from the describe-client-connectivity response to obtain the necessary configuration details to communicate with Apache HBase:

```
$ cdp opdb describe-client-connectivity --database-name [***YOUR DATABASE
 NAME***] --environment-name [***YOUR ENVIRONMENT NAME***]  | jq '.connect
ors[] | select(.name == "hbase") | .configuration.clientConfigurationUrl'
"https://client_Configuration_URL/.../services/hbase/clientConfig"
$ curl -k -o clientConfig.zip -u '[***USERNAME***]:[***CLOUDERA WORKLOAD
 PASSWORD***]' https://client_Configuration_URL/.../services/hbase/client
Config
```

   You can build the application in your local machine and copy the JAR files to the remote node using the following commands:

```
$ scp -r target ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws
.com:
```

```
$ scp clientConfig.zip ec2-user@my-ec2-bastion-host.us-west-2.compute.amaz
onaws.com:
$ ssh ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws.com "sudo
yum install -y java-1.8.0-openjdk"
$ ssh ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws.com "unzip
 clientConfig.zip"
```

2. Ensure that you have a Kerberos ticket:

```
$ kinit [***USERNAME***] Password: [***PASSWORD***]
$ java -cp target/nosql-libs/*:target/nosql-exemplar-0.0.1-SNAPSHOT.jar
:hbase-conf com.cloudera.odx.nosql.Client
```

3. Run your application.

Example: run an application for the Apache Phoenix thick client

Checking the example of how to run a maven application for the Phoenix thick client gives you better understanding about how to run your own application for the Phoenix thick client.

Before you begin

The requiredKerberos option is set to true for the Phoenix thick client. This means that the application must be run for this client from a computer which:

- Has internal network access to the public cloud in which the database is deployed
- Can resolve the internal hostnames of the database
- Can obtain a Kerberos ticket from the database's Kerberos Key Distribution Center (KDC)

One way to run an Apache Phoenix thick client application is to launch an edge node in your cloud provider which meets the above requirements.

Procedure

1. Use the JDBC URL from the describe-client-connectivity command to run the example.

   One method is to build on your local machine and copy the JAR files to the remote node:

```
$ scp -r target ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws
.com:
$ scp clientConfig.zip ec2-user@my-ec2-bastion-host.us-west-2.compute.amaz
onaws.com:
$ ssh ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws.com "sudo
yum install -y java-1.8.0-openjdk"
```

2. Ensure that you have a Kerberos ticket and run your application for the Phoenix thick client

```
 kinit USERNAME
$ java -cp target/sql-libs/*:target/sql-exemplar-0.0.1-SNAPSHOT.jar:hbase-
conf com.cloudera.odx.sql.Client "[***PHOENIX THICK JDBC URL***]"
```

3. Run your application.

Related Information

COD edge node overview

Example to run application using the Phoenix thin client

Checking the example of how to run a maven application for the Phoenix thin client gives you better understanding about how to run your own application for the Phoenix thin client.

The requiredKerberos options set to false for the Phoenix thin client which means that it can be used from virtually any node. In this example, the client runs from the local machine.

For the Apache Phoenix thin client the describe-client-connectivity call returns a base JDBC URL . You must append the following attributes to the URL which are specific to your identity:

- avatica_user: your Cloudera username (required)
- avatica_password: your Cloudera worload password (required)
- truststore: a truststore for your Cloudera Knox gateway (optional)
- truststore_password: the password for the truststore file (optional)

You can use Maven to ease launching this application, but a standalone Java program is similarly launched:

```
$ mvn exec:exec -Dexec.executable=java -Dexec.args='-cp target/sql-libs/*:ta
rget/sql-exemplar-0.0.1-SNAPSHOT.jar com.cloudera.odx.sql.ThinClient "jdbc:p
hoenix:thin:url=[***PHOENIX THIN JDBC URL;serialization=PROTOBUF;authenti
cation=BASIC;avatica_user=[***USERNAME***];avatica_password=[***PASSWORD***]
;truststore=[***CDP-TRUSTSTORE.JKS***];truststore_password=[***TRUSTSTORE
 PASSWORD***]"'
```

Or, you can launch the application without the help of Maven:

```
$ java -cp target/sql-libs/*:target/sql-exemplar-0.0.1-SNAPSHOT.jar com.clou
dera.odx.sql.ThinClient "jdbc:phoenix:thin:url=[***PHOENIX THIN JDBC URL;ser
ialization=PROTOBUF;authentication=BASIC;avatica_user=[***USERNAME***];ava
tica_password=[***PASSWORD***];truststore=[***CDP-TRUSTSTORE.JKS***];trustst
ore_password=[***TRUSTSTORE PASSWORD***]"
```

Compile an application for your Cloudera Operational Database

Once you have created your application and a database using Cloudera Operational Database, you have to compile your application for your database.

Before you begin

- Set your Cloudera workload password. For more information, see the *Setting the workload password* documentation in the related information section.
- Grant the ODUser role to the machine user using the Cloudera Management Console Environments *ENVIRONMENT_NAME* Actions Manage Acces page for their Cloudera environment. By setting the ODUser role you can grant a number of rights in the system that allows you to access the Cloudera Operational Database using the machine user's workload password.
- Add synchronized users from User Management Service in the Cloudera Control Plane into the environment in which your Cloudera Operational Database database is running.

Procedure

1. Get the Maven repository location to fetch JAR files.

   There are two ways to get the necessary information:

   - In command line: Using the cdp opdb describe-client-connectivity --database-name <*YOUR_DATABASE*> --environment-name <*YOUR_ENVIRONMENT*> command.
   - In the Cloudera Operational Database user interface: Clicking the Connect bar and selecting the applicable client.

   You have to use the version and the mavenURL attributes in your Maven project and configuration.

   The following is an example about how to fetch the required HBase information to build your application:

```
$ cdp opdb describe-client-connectivity --database-name <YOUR_DATABASE> --
environment-name <YOUR_ENVIRONMENT>  | jq '.connectors[] | select(.name
 == "hbase")'
{
  "name": "hbase",
  "version": "2.2.3.7.2.0.0-219",
  "kind": "LIBRARY",
  "dependencies": {
    "mavenUrl": "HTTPS://REPOSITORY.CLOUDERA.COM/ARTIFACTORY/CLOUDERA-
REPOS"
  },
  "configuration": {
    "clientConfigurationUrl": "HTTP://CLIENT_CONFIGURATION_URL/.../
SERVICES/HBASE/CLIENTCONFIG"
  },
  "requiresKerberos": true
```

   The following example fetch the required Phoenix information to build your application:

```
 cdp opdb describe-client-connectivity --database-name <YOUR_DATABASE> --e
nvironment-name <YOUR_ENVIRONMENT> | jq ".connectors[] | select(.name == \
"phoenix-thick-jdbc\") | .dependencies.mavenUrl"
  cdp opdb describe-client-connectivity --database-name <YOUR_DATABASE> --
environment-name <YOUR_ENVIRONMENT> | jq ".connectors[] | select(.name ==
 \"phoenix-thin-jdbc\") | .version"

Phoenix-thick
"HTTPS://REPOSITORY.CLOUDERA.COM/ARTIFACTORY/CLOUDERA-REPOS"
"5.0.0.7.2.0.0-128"
Phoenix-thin
"HTTPS://REPOSITORY.CLOUDERA.COM/ARTIFACTORY/CLOUDERA-REPOS"
"5.0.0.7.2.0.0-128"
```

2. Modify your application's settings.

   Ensure your application's settings.xml file uses the correct URL and version for your Cloudera Operational Database database.

   An example when using NoSQL client:

```
<project>
  <dependencies>
  <!-- NoSQL client for Cloudera Operational Database -->
    <dependency>
      <groupId>org.apache.hbase</groupId>
      <artifactId>hbase-shaded-client</artifactId>
      <version>2.2.3.7.2.0.0-219</version>
    </dependency>
  </dependencies>
  ...
```

```
  <repositories>
  <!-- Define our Cloudera Operational Database repository; this would be
 given to us by Cloudera Operational Database itself -->
    <repository>
      <id>nosql-odx</id>
<url>HTTPS://MAVEN_URL/CDP-PROXY/HBASE/JARS</url>
         <name>Cloudera NoSQL Cloudera Operational Database Repository</
name>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>
</project>
```

An example when using SQL client:

```
<project>
  <dependencies>
    <!-- SQL client for ODX -->
    <dependency>
      <groupId>org.apache.phoenix</groupId>
      <artifactId>phoenix-client</artifactId>
      <version>5.0.0.7.2.0.0-128</version>
    </dependency>
    <dependency>
      <groupId>org.apache.phoenix</groupId>
      <artifactId>phoenix-queryserver-client</artifactId>
      <version>5.0.0.7.2.0.0-128</version>
    </dependency>
  </dependencies>
  ...
  <repositories>
  <!-- Define our Cloudera Operational Database repository -->
    <repository>
      <id>sql-cod</id>
<url>HTTPS://MAVEN_URL/CDP-PROXY-API/AVATICA/MAVEN</url>
      <name>Cloudera SQL Cloudera Operational Database Repository</name>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>
</project>
```

3. Build your application.
4. Run your application for the applicable client.

Related Information

Setting the workload password

Cloudera Operational Database ..

User Management

Date published: 2020-08-14
Date modified: 2023-01-12

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

User management in Cloudera Operational Database..... 4

User management in Cloudera Operational Database

You can assign roles to the users to ensure that they are given the required permissions when accessing and using Cloudera Operational Database. To provide access to Cloudera Operational Database resources, you must add users and groups, and assign roles and resources to them using the Cloudera Management Console.

A Cloudera *POWERUSER* must assign roles to users who require access to the operational databases that are associated with specific environments. After you grant these roles to users and groups, they acquire a Sign-On (SSO) access to the operational database that is associated with an environment. For more information about configuring user roles, see Managing user access and authorization.

The following Cloudera resource roles are associated with the Cloudera Operational Database service:

- *ODAdmin*- This role enables users or groups to grant a Cloudera user or a group the ability to create, start, stop, and drop a database.
- *ODUser*- This role enables users or groups to view and use operational databases that are associated with specific environments.

ODAdmin

Grants permission to create, drop, and administer the Cloudera Operational Database for a Cloudera environment.

Important: You must be an *ENVIRONMENTADMIN* of the account to create a database.

Action	Right	Resource
HBase Admin user interface access	opdb/adminDatabase	Environment
Create a database	opdb/createDatabase	Environment
Describe a database	opdb/describeDatabase	Environment
List all databases	opdb/listDatabases	Environment
Fetch database metrics	opdb/fetchDatabaseMetrics	Environment
Drop a database	opdb/dropDatabase	Environment
Start a database	opdb/startDatabase	Environment
Stop a database	opdb/stopDatabase	Environment
Environment read	environments/read	Environment
Nodes ssh and Knox access	environments/accessEnvironment	Environment
Get keytab	environments/getKeytab	Environment
Set workload password	environments/setPassword	Environment

ODUser

Grants permission to list and use Cloudera Operational Database for a Cloudera environment.

Important: You must be an *ENVIRONMENTUSER* of the environment to utilize Cloudera Operational Database.

Action	Right	Resource
Describe a database	opdb/describeDatabase	Environment
List all databases in an environment	opdb/listDatabases	Environment
Fetch database metrics	opdb/fetchDatabaseMetrics	Environment

Environment read	environments/read	Environment
Nodes SSH and Knox access	environments/accessEnvironment	Environment
Set workload password	environments/setPassword	Environment

Related Information

Managing user access and authorization using the Cloudera Management Console

Cloudera Operational Database ..

Working with Edge Nodes

Date published: 2020-08-14
Date modified: 2024-05-15

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Operational Database edge node overview..... 4

Managing edge nodes..... 4

Creating an edge node.....4
Viewing edge nodes..... 5
Adding an edge node.....6
Deleting an edge node.....6

Deploying applications on Cloudera Operational Database.....6

Configuring an on-prem instance to connect to Cloudera Operational Database.....7
Connecting an on-prem instance to the Cloudera Operational Database on AWS.....8
Connecting an on-prem instance to the Cloudera Operational Database on Azure.....8
Connecting an on-prem instance to the Cloudera Operational Database on GCP.....8
Configuring DNS.....8
Verifying the DNS configuration.....9
Configuring Kerberos..... 9
Configuring JWT authentication for HBase client..... 10

Cloudera Operational Database edge node overview

An edge node is a resource dedicated to access private computing resources on the public cloud. You must configure an edge node in your public cloud environment if you use the Apache HBase Java API or the Apache Phoenix thick JDBC client.

Cloudera Operational Database cannot be accessed directly by clients and resources on the public internet. The subnet security group and ingress rules of your public cloud providers prevent you from accessing your database from a public network.

If you have enabled a public endpoint access gateway while creating your environment, you can access your Cloudera Operational Database instance from outside the public cloud through Apache Knox. But, if you use a private subnet in your environment, you must configure a VPN for your client applications to access your Cloudera Operational Database instance using Apache Knox or use an edge node.

Clients that use HTTP interfaces such as the HBase REST server, Thrift client and server, Phoenix Query Server, SQL over HTTP using Apache Phoenix thin JDBC driver, ODBC driver, Go driver, and Python phoenixdb library, need not use the edge node and can be proxied through the Apache Knox gateway.

However, you must create an edge node to use the Apache HBase Java API or the Apache Phoenix thick JDBC driver.

You can create an edge node in your Cloudera Operational Database cluster that acts as an individual node type and not as a separate Cloudera Data Hub cluster. The edge node automatically synchronizes with the Cloudera Operational Database cluster, which means you do not need to manually configure the node.

Related Information

Configure edge node on AWS

Configure edge node on Azure

Configure edge node on GCP

Managing edge nodes

Cloudera Operational Database supports creation of an edge node while creating an operational database. You can also modify the number of instances in the edge node type.

You must create an edge node if you plan to use Apache HBase Java API or the Apache Phoenix thick JDBC driver. You can use the edge node either as a gateway node or use it to deploy your applications while working with a Cloudera Operational Database cluster.

When you create an edge node in the Cloudera Operational Database cluster, it acts as an edge node type and not as a separate edge node Cloudera Data Hub cluster.

Related Information

Create database

List edge nodes

Update edge nodes

Creating an edge node

You can create an edge node while creating an operational database in your Cloudera environment. You can also define the number of nodes to be created.

Procedure

1. Launch the CDP CLI tool.
2. Use the following command to create an edge node.

   `cdp opdb create-database --environment-name` *<CDP_ENVIRONMENT_NAME>* `--database-name` *<DATABASE_NAME>* `--num-edge-nodes` *<NUMBER_OF_EDGE_NODES>*

   For example,

   ```
   cdp opdb create-database --environment-name cdp1 --database-name test1 --num-edge-nodes 1
   ```

Viewing edge nodes

You can view the list of available edge nodes on a Cloudera Operational Database cluster.

Procedure

1. Launch the CDP CLI tool.
2. Use the following command to list the available edge nodes.

   `cdp opdb list-edge-nodes --environment` *<CDP_ENVIRONMENT_NAME>* `--database` *<DATABASE_NAME>*

   For example,

   ```
   cdp opdb list-edge-nodes --environment odx-i2dr46 --database edge2
   ```

   Sample output:

   ```
   "database": "edge2",
       "edgeNodes": [
           {
               "instanceId": "cod--1qs4ckni9gwnj112955e22",
               "discoveryFQDN": "cod--1qs4ckni9gwnj-edge1.odx-i2dr.xcu2-8y8x
   .dev.cldr.work",
               "privateIp": "10.124.64.24",
               "publicIp": "N/A"
           },
           {
               "instanceId": "cod--1qs4ckni9gwnj112955e21",
               "discoveryFQDN": "cod--1qs4ckni9gwnj-edge2.odx-i2dr.xcu2-8y8x.
   dev.cldr.work",
               "privateIp": "10.124.64.26",
               "publicIp": "N/A"
           },
           {
               "instanceId": "cod--1qs4ckni9gwnj112955e20",
               "discoveryFQDN": "cod--1qs4ckni9gwnj-edge0.odx-i2dr.xcu2-8y
   8x.dev.cldr.work",
               "privateIp": "10.124.64.23",
               "publicIp": "N/A"
           },
           {
               "instanceId": "cod--1qs4ckni9gwnj112955e23",
               "discoveryFQDN": "cod--1qs4ckni9gwnj-edge3.odx-i2dr.xcu2-8y8
   x.dev.cldr.work",
               "privateIp": "10.124.64.25",
               "publicIp": "N/A"
           }
   ```

```
    ],
    "environmentName": "odx-i2dr46"
}
```

Adding an edge node

You can add additional edge nodes into your existing Cloudera Operational Database cluster. You can define the number of edge nodes to be added, Cloudera Operational Database automatically adds the additional nodes into the Cloudera Operational Database cluster.

Procedure

1. Launch the CDP CLI tool.
2. Use the following command to add edge nodes into the Cloudera Operational Database cluster.

   `cdp opdb update-edge-nodes --environment` *<CDP_ENVIRONMENT_NAME>* `--database` *<DATABASE_NAME>* `--num-add-edge-nodes` *<NUMBER_OF_EDGE_NODES>*

   For example,

   ```
   cdp opdb update-edge-nodes --environment cdp1 --database cod-7215 --num-add-edge-nodes 2
   ```

Deleting an edge node

You can delete single or multiple existing edge nodes from a Cloudera Operational Database cluster. You can use the instance ID of an edge node to delete it from the Cloudera Operational Database cluster.

Procedure

1. Launch the CDP CLI tool.
2. Use the following command to delete an existing node from a Cloudera Operational Database cluster.

   `cdp opdb update-edge-nodes --environment` *<CDP_ENVIRONMENT_NAME>* `--database` *<DATABASE_NAME>* `--delete-edge-instances` *<EDGE_NODE_INSTANCE_ID_1>* *<EDGE_NODE_INSTANCE_ID_2>*

   For example,

   ```
   cdp opdb update-edge-nodes --environment cdp1 --database cod-7215 --delete-edge-instances cod--1msrexj6oumro108509e10 cod--1msrexj6oumro108509e12
   ```

Deploying applications on Cloudera Operational Database

The edge node is a dedicated Cloudera Data Hub cluster that enables you to communicate with your Cloudera Operational Database instance and your applications. You can deploy a cluster that works as an edge node to access your Cloudera Operational Database instance. Deploy the edge node cluster in the same environment as the Cloudera Operational Database instance to ensure that the security groups and data ingress rules that apply to the Cloudera Operational Database instance must also apply to the edge node cluster.

Procedure

1. From the Cloudera Management Console, click Data Hub Clusters.
2. Click Create Data Hub.

3. In the Selected Environment with running Data Lake drop-down list, select the same environment used by your Cloudera Operational Database instance.
4. Select the Cluster Definition.
5. In the Cluster Definition drop-down list, select the [***RUNTIME VERSION***] COD Edge Node for [***CLOUD PROVIDER NAME***].

   For example, select the 7.2.10 COD Edge Node for AWS cluster template.

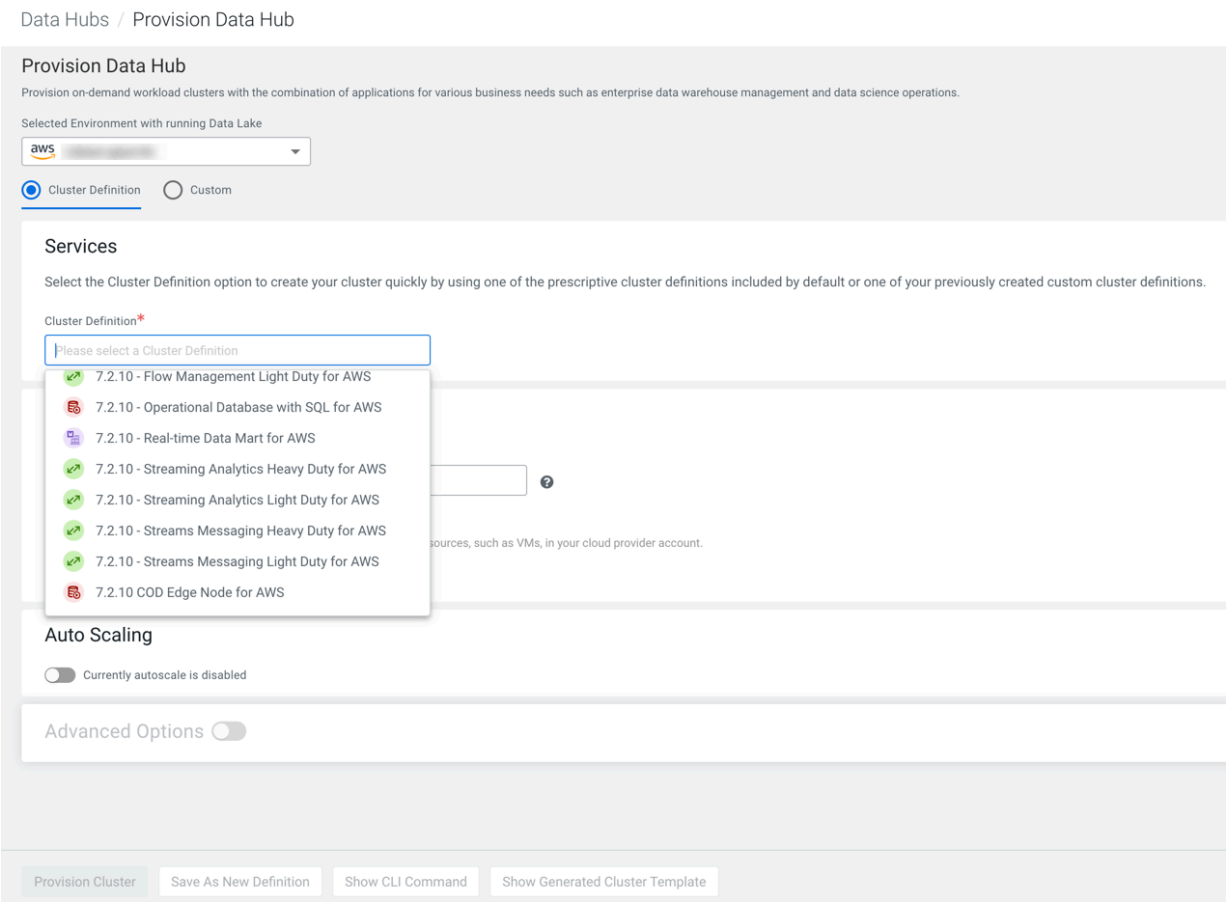

6. In the Cluster Name field, provide a cluster name that you can identify later as an edge node of a specific Cloudera Operational Database instance.
7. Click Provision Cluster.

What to do next

After you deploy the edge node, you can run your applications on this edge node using the Client connectivity information. See how to compile applications for Cloudera Operational Database in Compile an application for your database.

Configuring an on-prem instance to connect to Cloudera Operational Database

You can configure an on-prem instance or node to connect to the Cloudera Operational Database deployed on an AWS, Azure, or GCP environment to run applications using Apache HBase Java API or the Apache Phoenix thick JDBC driver. When configuring a node, you have to configure a DNS and Kerberos

In Cloudera Operational Database, by default, your database is not accessible to the public internet due to security groups on the subnets of the VPC in which the database is deployed and the ingress rules of the VPC itself.

Configuring an on-prem instance correctly can run applications using the Apache HBase Java API or the Apache Phoenix thick JDBC driver.

Ensure that the following prerequisites are met before connecting the on-prem instance to the Cloudera Operational Database.

- You have the required permission to log into the Cloudera Operational Database on AWS, Azure, or GCP configured with Cloudera.
- You have the required permission to launch EC2 nodes in your AWS account.
- You have a basic understanding of the VPC and subnets created for use with Cloudera.

Connecting an on-prem instance to the Cloudera Operational Database on AWS

Learn how to connect an on-prem instance to the Cloudera Operational Database deployed on an AWS environment.

To connect an on-prem instance to the Cloudera Operational Database on AWS, see the AWS documentation, *Connect your VPC to other networks*.

Related Information

Connect your VPC to other networks

Connecting an on-prem instance to the Cloudera Operational Database on Azure

Learn how to connect an on-prem instance to the Cloudera Operational Database deployed on an Azure environment.

To connect an on-prem instance to the Cloudera Operational Database on Azure, see the Microsoft documentation, *Connect an on-premises network to a Microsoft Azure virtual network*.

Related Information

Connect an on-premises network to a Microsoft Azure virtual network

Connecting an on-prem instance to the Cloudera Operational Database on GCP

Learn how to connect an on-prem instance to the Cloudera Operational Database deployed on a GCP environment.

To connect an on-prem instance to the Cloudera Operational Database on GCP, see the Google documentation, *Cloud Interconnect overview*.

Related Information

Cloud Interconnect overview

Configuring DNS

You must configure your instance or node to perform forward and reverse DNS lookups with your Cloudera Operational Database. Obtain the private IP address of your Cloudera environment and configure your instance to resolve hostnames from your Cloudera Operational Database. An instance must resolve the hostnames from your Cloudera Operational Database.

About this task

Each Cloudera environment acts as its own DNS nameserver. Obtain your environment's private IP address information and configure resov.conf on your instance to list your Cloudera environment's private IP address as a nameserver.

Procedure

1. Go to the Cloudera Management Console.
2. Click Environments and select your environment from the list.
3. Click Summary.
4. Find the FreeIPA section and copy the contents of the Private IP field.

5. Add the private IP address as a nameserver to the /etc/resolv.conf file in your instance. You can do this using the cat resolv.conf nameserver [****NAMESERVER IP ADDRESS****] command.

> **Note:** When an HBase client application running on the server cannot be set up with reverse DNS lookup and can only perform forward DNS, then the hbase.unsafe.client.kerberos.hostname.disable.reversedns property can be set to true at the client configuration so that the HBase client connects to the HBase cluster through SASL Kerberos using the hostname of the principal and skips the reverse DNS lookup.

Verifying the DNS configuration

After configuring the nameserver you must verify that your instance can now resolve DNS names.

Procedure

1. Navigate to Cloudera Control Plane Cloudera Management Console .
2. Click Environments and select your environment from the list.
3. Click Summary.
4. Find the FreeIPA section and copy the content of the Private IP field.
5. Open a terminal.
6. Run the following command: $ nslookup [***FULLY QUALIFIED DOMAIN NAME***]
   If your DNS is set up correctly, this command returns an address for the name you provided.

Configuring Kerberos

All Cloudera Operational Databases (CODs) are secured with Kerberos-based authentication, which means only authorized users can connect to your database. All HBase and Phoenix Thick JDBC clients must have a Kerberos configuration on the host where they run a client.

Procedure

1. Run the following command to obtain the necessary Kerberos information and a sufficient krb5.conf file encoded with Base64.

```
$ cdp opdb describe-client-connectivity --environment-name [***YOUR
 ENVIRONMENT***] \
  --database-name [***YOUR DATABASE NAME***] | jq -r \
  '.kerberosConfiguration.krb5Conf' | base64 --decode
```

2. Copy the output of the command.
3. Add the contents into the /etc/krb5.conf file on your instance.

What to do next

Validate that Kerberos is correctly set up. Use the kinit command to validate that you are able to obtain a Kerberos ticket.

```
$ kinit [***CLOUDERA WORKLOAD NAME***]
                    Password: [***CLOUDERA WORKLOAD PASSWORD***]
```

For more information, see *Cloudera workload user* and *Setting the workload password*.

If you successfully authenticate, you do not receive an error and can validate that you have a ticket using the `klist` command. For more information, see *Installing CDP CLI beta*.

After configuring the Kerberos successfully, you must compile your instance or application against the Cloudera Operational Database database. For more information, see *Compiling Applications*.

Related Information

Setting the workload password

Cloudera workload user
Installing CDP CLI beta
Compile an application for your COD database

Configuring JWT authentication for HBase client

JWT (JSON Web Token)-based authentication uses an unique identifier and is a standard way of securely transmitting signed information between two parties. Learn how to configure JWT-based authentication for your HBase client.

About this task

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted. The structure of JWT allows you to verify whether the content is tampered.

To disable JWT authentication for HBase clients, you can use the --disable-jwt-auth option while creating an operational database using CDP CLI. Ensure that the COD_JWT_AUTH entitlement is enabled for the HBase client.

For example,

`cdp opdb create-database --environment-name` *MYENVIRONMENT* `--database-name` *MYDATABASE* `--disable-jwt-auth`

Important:

- HBase client tarball contains all the binary dependencies that are required for JWT to function smoothly.
- You cannot perform MapReduce operations such as RowCounter and bulk-load while using JWT authentication.

Before you begin

- CDP CLI must have been configured to access Cloudera environments.
- Ensure that you have COD_JWT_AUTH entitlement enabled for your HBase client.

Procedure

1. On the Cloudera Operational Database UI, click on the database and go to Connect HBase Client Tarball JWT Configuration .

   This section provides you the necessary details in setting up a connection to HBase with a JWT token.

Connect Charts Events

HBase | HBase REST | HBase Client Tarball | Phoenix (Thick)

Usage

You can download the Apache HBase Client Tarball that contains the
such as HBase Shell or SQLLine.

HBase Version

x.x.x.x.x.x.x.x.x.x.x.x

Download URL

https://cod--xnz769xcwvk3-gateway0.cod-7216.xcu2-8y8x.dev.cldr.w

HBase Client Configuration URL

curl -f -o "hbase-config.zip" -u "csso_l xxxxxxxxxxxx "https://cod--xnz7

> Kerberos Configuration

> Yarn Configuration

JWT Configuration

Download Environment Certificate

Run the following command to create the truststore:

```
keytool -importcert -noprompt -storetype JKS -keystore truststore j
```

hbase-site.xml

```
<!-- TLS -->

<property>
  <name>hbase.client.netty.tls.enabled</name>
```

2. Download the Environment Certificate and run the command as mentioned on the UI to build your own truststore JKS file.
3. Open the HBase client's hbase-site.xml file. The file is usually located in /etc/hbase/conf.

   Download the configuration snippet from the UI or add the following TLS and JWT properties to the hbase-site.xml file filling in the template based on your local configuration.

```
<!-- TLS -->

<property>
    <name>hbase.client.netty.tls.enabled</name>
    <value>true</value>
</property>
<property>
   <name>hbase.rpc.tls.truststore.location</name>
   <value>/path/to/truststore.jks</value>
</property>
<property>
    <name>hbase.rpc.tls.truststore.password</name>
    <value>...</value>
</property>
<property>
    <name>hbase.rpc.tls.truststore.type</name>
    <value>jks</value>
</property>

<!-- JWT -->
<property>
  <name>hbase.client.sasl.provider.extras</name>
  <value>com.cloudera.hbase.security.provider.OAuthBearerSaslClientAuth
enticationProvider</value>
</property>
<property>
  <name>hbase.client.sasl.provider.class</name>
  <value>com.cloudera.hbase.security.provider.OAuthBearerSaslProviderSelec
tor</value>
</property>
<property>
  <name>hbase.client.sasl.oauth.tokenprovider</name>
  <value>com.cloudera.hbase.security.token.FileOAuthBearerTokenProvider</
value>
</property>
<property>
  <name>hbase.client.sasl.oauth.tokenfile</name>
  <value>/path/to/token.txt</value>
</property>
```

   Note: Cloudera recommends that you set this client configurations in hbase-site.xml file using the CDP CLI client instead of Cloudera Manager. If you set using Cloudera Manager, you might encounter problems.

4. Ensure that the following JWT libraries (included in the HBase client tarball) are added on the classpath.

```
cloudera-opdb-jwtauth-client-1.0.0.7.2.16.0-SNAPSHOT.jar
cloudera-opdb-jwtauth-common-1.0.0.7.2.16.0-SNAPSHOT.jar
nimbus-jose-jwt-9.15.2.jar
```

5. Obtain the JWT token from the IAM service or the console authentication service.

```
cdp iam generate-workload-auth-token --workload-name OPDB
```

The command returns a JWT token in a JSON format. For example,

```
{
    "token": "eyJraWQiOiJjMDBjNmRlNGE1MjIyYTk1IiwidHlwIjo...",
    "expireAt": "2022-03-17T17:10:32.472000+00:00"
}
```

6. Copy and paste the base64 encoded token into a TXT file (token.txt) with username. For example,

```
<username>,<token>
cloudbreak,eyJraWQiOiJjMDBjNmRlNGE1MjIyYTk1Iiwid…
```

What to do next

Validate that JWT is correctly set up. Use the following `list` command to validate that you are able to run commands on HBase.

```
bin/hbase shell
hbase> list
```

After successful authentication, you can see the list of available tables in the database.

Related Information

Create database

Cloudera Replication Manager

Replication Manager Overview

Date published: 2019-11-15
Date modified: 2024-07-08

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Replication Manager in Cloudera Public Cloud 4

Replication Manager terminology 5

Fine-grained permission to access Cloudera Replication Manager 6
- Providing role-based access control (RBAC) to Replication Manager users 6

Access Replication Manager in Cloudera Public Cloud 7
- Overview 7
  - Classic Clusters 8
  - Policies 8
  - Jobs 9
  - Issues & Updates 9
- Classic Clusters page 10
- Cloud Credentials page 11
- Replication Policies page 12

How replication policies work 13
- Replication policy considerations 14

How temporary AWS credentials for replication policies works 14
- Authentication methods to use AWS credentials in replication policies 15

HDFS replication policy 16
- HDFS snapshots 16
  - Requirements and benefits of HDFS snapshots 17
  - Enabling and taking snapshots in Cloudera Manager 18

Hive replication policy 18
- Hive replication 19
- Hive tables 20
- Hive cloud replication 21
- Table-level replication 22
- Migrate Sentry authorization policies into Ranger 22
  - Sentry to Ranger permissions 23

HBase replication policy 24
- Supported clusters for HBase replication policies 24
- How HBase replication policies work 25
- Methods to replicate HBase data 26
- Replicate HBase data simultaneously between multiple clusters 27

Replication Manager in Cloudera Public Cloud

Replication Manager is a service in Cloudera Public Cloud. You can create replication policies in Replication Manager to copy and migrate data from CDH (version 5.13 and higher) clusters (HDFS, Hive, and HBase data) and Cloudera Private Cloud Base (version 7.1.4 and higher) clusters (HDFS, Hive external tables, and HBase data) to Cloudera Public Cloud clusters. You can also replicate HDFS data from cloud storage to classic clusters (CDH or Cloudera Private Cloud Base clusters), and Hive external tables to Data Hubs. The supported Cloudera Public Cloud services include Amazon S3 and Microsoft Azure ADLS Gen2 (ABFS). Replicating Hive managed tables using Replication Manager from HDP clusters to Cloudera Public Cloud is a beta feature and is not available for general use.

Replication Manager can replicate between services where at least the source service or the destination service is a local service. The term *local service* refers to a service of a cluster managed by Cloudera Manager. In the on-premises to on-premises replication scenarios, you create the replication policy in the target Cloudera Manager. It is not possible to create replication policies between two non-local services, services on separate peer clusters, or between cloud and a peer cluster's service. In these unsupported scenarios, you can create replication policies on the Cloudera Manager that has the service that you are replicating to/from.

Before you create replication policies, you must ensure that the clusters are supported by Replication Manager. For more information, see Support matrix for Replication Manager on Cloudera Public Cloud.

You can access the Replication Manager service on the Cloudera Public Cloud web interface. To replicate data between clusters, add the source on-premises clusters as classic clusters on the Cloudera Management Console Clusters page, add/create one or more Cloudera Public Cloud SDX Data Lakes and/or Data Hubs, and then create the replication policies in Replication Manager. The **Replication Policies** page shows the progress and status of replication policy jobs. You can also use CDP CLI to create HDFS and Hive replication policies.

Replication Manager provides the following functionalities that you can use to accomplish your data replication goals:

HDFS replication policies

These policies replicate HDFS data and metadata from on-premises clusters (CDH, Cloudera Private Cloud Base, and HDP) to Public Cloud storage buckets such as S3 and ABFS, and from cloud storage to classic clusters (CDH or Cloudera Private Cloud Base clusters). You can choose the frequency of replicating data.

Some use cases where you can use HDFS replication policies include:

- Moving legacy data (from CDH clusters) to cloud deployments (AWS or Azure on Cloudera Public Cloud).
- Archiving cold data.
- Replicating the required data to another cluster to run analytics on it.

Hive replication policies

These policies support table-level replication and can replicate Hive external tables from on-premises clusters (CDH and Cloudera Private Cloud Base) to cloud storage such as S3 and ABFS and to Data Hubs. They also can:

- replicate data stored in Hive tables, Hive metadata, data in Hive metastore, and Impala metadata (catalog server metadata) associated with Impala tables registered in the Hive metastore, and

  Note: Hive2 managed tables are converted to external tables after replication.

- migrate Sentry permissions to Ranger.

  Note: To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.

Some use cases where you can use Hive replication policies include:

- Backing up data periodically.
- Performing a recovery operation when necessary.
- Creating a development and test system for engineers to run quality checks.

HBase replication policies

You can create these policies to replicate HBase data from a source classic cluster (CDH or Cloudera Private Cloud Base cluster), COD, or Data Hub to a target Data Hub or COD cluster. You can also copy or replicate HBase data between different environments within a Virtual Private Cloud (VPC) using these policies. Any future data change in the source cluster is pushed to the target cluster automatically without user intervention.

Some use cases where you can use HBase replication policies include:

- Performing an active-active disaster recovery with conflict resolution (enabling other disaster recovery use cases which provides an efficient utilization of resources).
- Copying required data to the cloud clusters for heavy-duty analytics workloads which helps to optimize on-premises cluster performance.
- Utilizing the continuous data synchronize feature to implement a hybrid cloud that in turn helps you to use it in various other use cases.

CDP CLI for HDFS and Hive replication policies

You can also use CDP CLI commands to create HDFS and Hive replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see *CDP CLI for Replication Manager*.

CDP CLI is a unified tool to manage all the Cloudera Public Cloud services. This gives you the flexibility to use it across services from a single pane, view required information in a single scroll (for example, you can view all available clusters' service status in a single page), and collaborate to troubleshoot issues.

Related Information

Support matrix for Cloudera Replication Manager

HBase replication policy

CDP CLI for Cloudera Replication Manager

Replication Manager terminology

Replication Manager is a service that can be accessed through the Cloudera Public Cloud web interface in Cloudera Data Platform. You can create replication policies in Replication Manager. You can also use CDP CLI commands to create replication policies.

Term	Description
Replication Manager Service	The web UI that runs on the Cloudera Data Platform host.
Data center	The facility that contains the computer, server, and storage systems and associated infrastructure, such as routers and switches. Corporate data is stored, managed, and distributed from the data center. In an on-premises environment, a data center is often composed of a CDH cluster or Cloudera Private Cloud Base cluster. However, a single data center can contain multiple on-premises clusters.
Cloud data lake or data lake	A Cloudera Public Cloud cluster on the cloud, using virtual machines, with data retained on cloud storage. A cloud data lake requires minimal services for metadata and governance, such as Hive metastore, Ranger, and Atlas.
Cloud storage	A storage retained in a cloud account, such as Amazon S3 web service or Microsoft Azure.

Term	Description
On-premises cluster	A CDH cluster in a data center or a Cloudera Private Cloud Base cluster, with Apache services running, such as HDFS, Yarn, HMS, Hiveserver2, Ranger, and Atlas. Replication behavior is similar to IaaS cluster replication. The data is on local HDFS.
Replication policy	A set of rules applied to a replication relationship. The rules include which clusters serve as source and destination, the type of data to replicate, the schedule for replicating data, and so on.
Job	An instance of a replication policy that is running or is completed
Source cluster	The cluster that contains the source data that is replicated to a destination cluster. Source data could be an HDFS dataset, Hive database, or HBase tables.
Target cluster	The cluster to which the data is replicated.

Related Information

How replication policies work
HDFS replication policy
Hive replication policy
HBase replication policy
Support matrix for Cloudera Replication Manager

Fine-grained permission to access Cloudera Replication Manager

You can restrict access to specific users to view and use Cloudera Replication Manager in a Cloudera Public Cloud environment so that you can govern the access to critical replication functionalities.

Currently, any user in a Cloudera Public Cloud environment can view and use Cloudera Replication Manager to create, run, and manage replication policies. However, in some deployments, it is essential that only a few authorized users have access to Replication Manager. This requirement arises when you want to provide an added layer of control which aligns with the best practices for data management and security, and also to enhance security and control over replication management which includes monitoring the replication jobs, and troubleshooting issues efficiently.

Providing role-based access control (RBAC) to Replication Manager users

You can provide fine-grained permissions to specific users to view and use Cloudera Replication Manager

Procedure

1. Enable the RBAC entitlement. Contact your Cloudera account team to accomplish this task.
2. Identify the users that require access to Cloudera Replication Manager.
3. Go to the Cloudera Management Console User Management page. You can manage the role assignments on this page.

   Important: Administrators with the PowerUser role can add, modify, or delete users and groups in the Cloudera Public Cloud environment.

4. Assign the ReplicationAdmin role to one or more users, and Save the changes.

   Note: You must have the ReplicationAdmin or PowerUser role to use Replication Manager if the entitlement is enabled.

5. Optionally, create a group to manage the Replication Manager users and their roles. For example, replicationusers.

Access Replication Manager in Cloudera Public Cloud

You can access the Replication Manager service by logging into Cloudera Data Platform.

When you log into Cloudera Data Platform, the Cloudera Public Cloud web interface appears. Click Replication Manager to view the **Overview** page of the Replication Manager.

Replication Manager has the following pages:

- Overview
- Classic Clusters
- Cloud Credentials
- Replication Policies

The following image shows the Cloudera Public Cloud web interface:

You can also access the Replication Manager service by logging into the Management Console. In the Management Console, click and select Replication Manager.

Overview

When you click Replication Manager on the Cloudera Public Cloud web interface, the Overview page appears. The page provides a snapshot of the Replication Manager service. It provides insights into issues and updates related to various entities and resources through dashboards like Classic Clusters, Replication Policies, Notifications, and so on.

The following panels appear on the Overview page:

- Classic Clusters
- Policies
- Jobs
- Issues & Updates

Click Create Policy to create a replication policy.

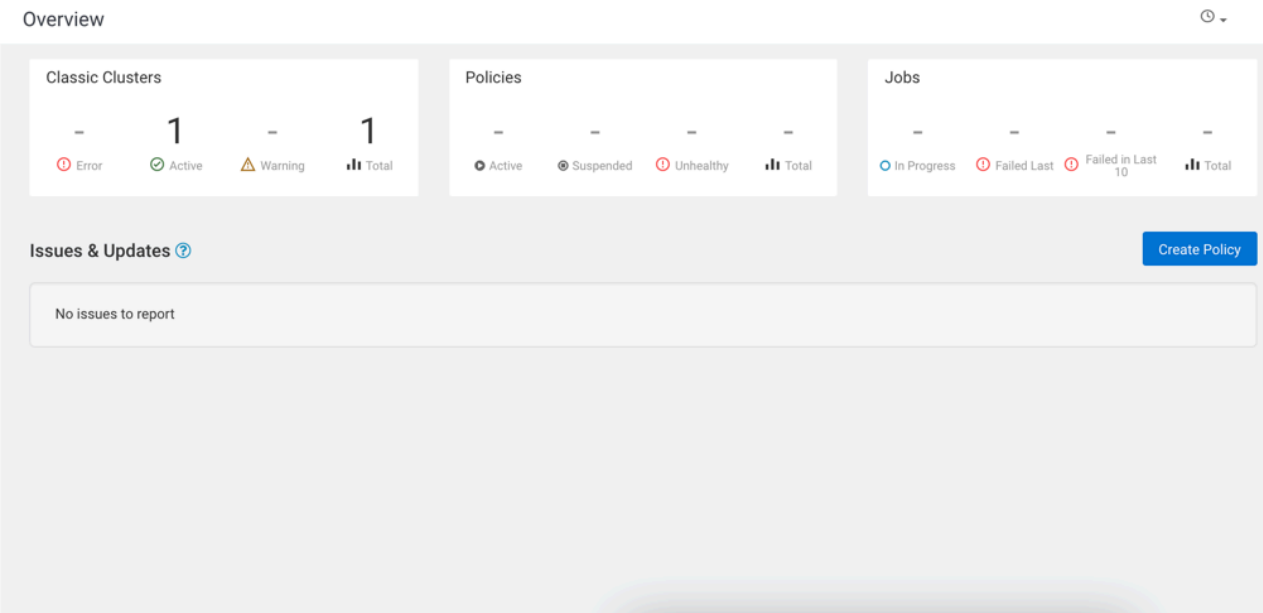

Classic Clusters

The Classic Clusters panel on the Overview page tracks the total number of clusters enabled for Replication Manager, the number of clusters that are in an error state, the number of clusters that are active, and the number of clusters for which a warning is issued.

You must register your existing on-premises Cloudera Distribution of Hadoop (CDH) and Cloudera Private Cloud Base clusters on the Cloudera Management Console, after which you can copy or move your data to the cloud. In Cloudera Public Cloud, these clusters are called *classic clusters*.

The Classic Clusters panel shows the following cluster status:

- **Active** clusters that are currently available to run the replication jobs.
- Clusters having less than 10% disk capacity appears as **Warning**. Click the number to open a table to track the cluster name and the exact disk capacity remaining.
- **Total** number of clusters that are in use.
- Number of clusters that are currently not running as expected or in **Error** state.

To investigate the issues associated with clusters that have an error or warning status, use Cloudera Manager for on-premises clusters.

Policies

The Policies panel on the Overview page tracks the number of replication policies that are in use and their status.

The **Policies** panel shows the following status for the replication policies:

- **Active** replication policies in Submitted state or Running state. This item is not actionable.
- Replication policies that are **Suspended** by the administrator. This item is not actionable.
- **Unhealthy** replication policies that are associated with a cluster designated as Error on the **Classic Clusters** panel. Click the number to find out the policy names, the names of its associated source clusters and destination clusters, and the services that have stopped on the source cluster or destination cluster.
- **Total** number of running policies.

When you click **Policies** on the **Overview** page, the **Replication Policies** page appears. On this page, you can view the replication policy details.

Jobs

The Jobs panel on the Overview page tracks the total number of running and failed jobs and their status in Replication Manager.

The **Jobs** panel shows the following details:

- **In Progress** jobs or jobs in running state.
- **Failed Last** replication policies are the policies for which the last job failed to complete.
- **Failed in Last 10** replication policies are the policies for which at least one of the last ten jobs failed.
- **Total** number of available jobs.

You can click the number in the panel to apply a filter to the **Issues & Updates** table to view the required policies.

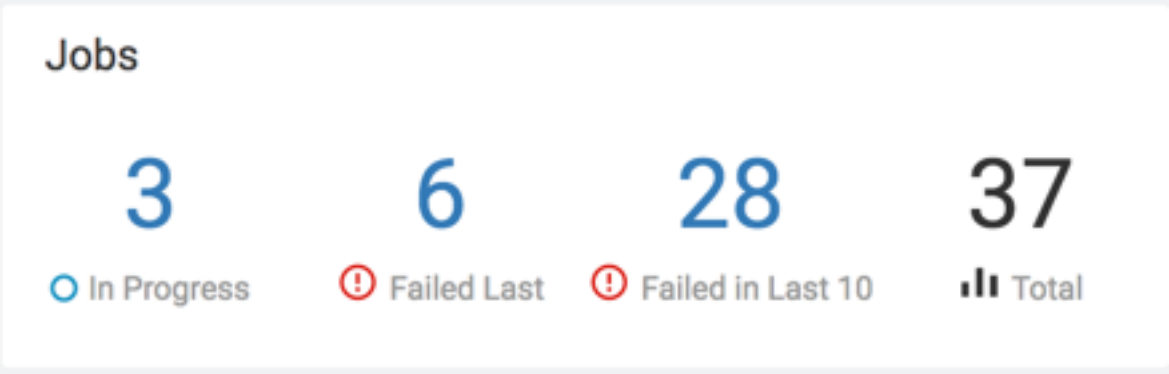

Issues & Updates

The Issues & Updates panel lists the replication policies that have running jobs with at least one job in Failed status in the most recent ten jobs. If you do not see any policy, it indicates that the last ten jobs of all the replication policies were successful.

The **Issues & Updates** panel has the following columns:

- Current **Job Status** of the job. When the job status is running, the status circle icon and a progress bar appear. For jobs that are not running, the status circle icon appears with Success, Failed, or Ignored text. Hover over the Failed status and click View Log to view the job log.
- **Source** cluster associated with the replication policy.
- **Destination** cluster associated with the policy.
- **Service** indicates whether the data being replicated is HDFS, Hive, or HBase.
- Replication **Policy** name.
- Policy History of the last ten job statuses. The column also shows the status as colored dots which you can click to view the policy details on the **Policies** page:
  - Green indicates that the job completed successfully.
  - Red indicates that the job did not complete.
  - Gray indicates that the job did not start because a previous instance of the policy is still in progress. Only one run of a job can be in progress at one time. If a job is consistently ignored, edit the replication policy to modify its frequency.

  When you click the colored dots, the page appears with the filter preset to show the information about the specified policy.
- **Transferred/Files** is the amount of data transferred, in gigabytes, and the number of objects transferred, if available. When a job is running, the column shows In Progress.
- **Runtime** or time taken to complete the most recent job.
- Most recent job that **Started**.
- Most recent job that **Ended**.

The **Actions** menu shows the following options:

- **Abort Job** aborts a running job.
- **Re-run Job** starts another instance of the policy. This option is not available for running jobs.
- **Edit Policy** settings. This option is not available for expired policies.
- **Delete Policy** removes the replication policy permanently. The delete operation cannot be undone.
- **Suspend Policy** only if the job is running.
- **Activate Policy** resumes a suspended replication policy.

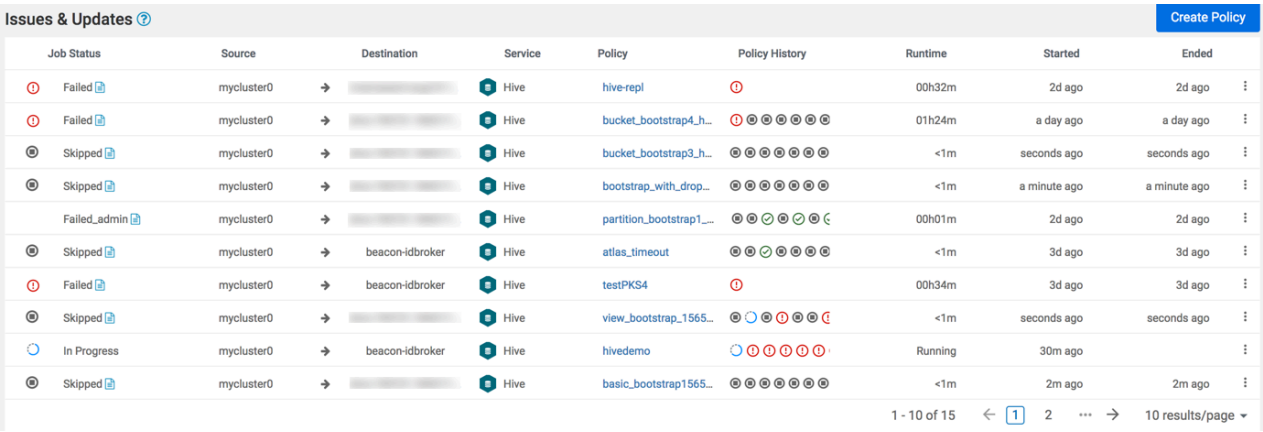

Classic Clusters page

The Classic Clusters page specifies the total number of clusters enabled for Replication Manager, the number of clusters that are in an error state, the number of clusters that are active, and the number of clusters for which a warning is issued.

The **Classic Clusters** page shows the cluster health status, cluster name, cluster version, number of nodes in the cluster, number of replication policies in the cluster, and the location of the cluster. Use the Actions menu to create a replication policy for the cluster, launch Cloudera Manager, or sync the cluster configuration. Click Add to create a replication policy.

The Classic Clusters map panel shows the geolocation of each cluster and helps you to easily identify the status of cluster services, using the following interactive markers on the map:

- Red indicates that at least one required service has stopped on the cluster.
- Orange indicates that all the required services are running on the cluster but the remaining disk capacity on the cluster is less than 10%.
- Green indicates that all the required services are running on the cluster and the remaining disk capacity is greater than 10%.

Hover over a marker on the map to view the data center associated with the cluster, the cluster name, and the number of Replication Manager policies that are associated with that cluster.

To investigate the issues associated with clusters that have an error or warning status, launch Cloudera Manager.

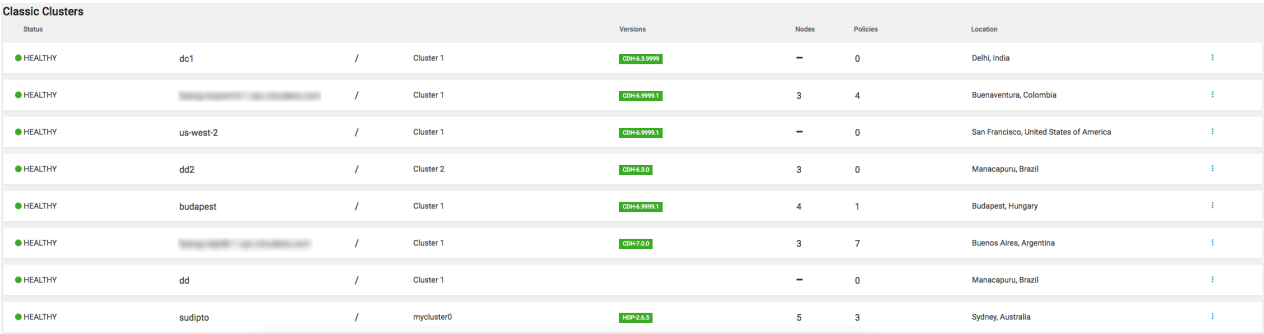

Cloud Credentials page

The Cloud Credentials page shows the registered cloud credentials for Replication Manager. To replicate data to or from a storage cloud account, you must register the cloud credentials, so that the Replication Manager can access your cloud account. The supported cloud storage accounts are Amazon S3 and Azure Blob Filesystem (ABFS). On the Cloud Credentials page, you can add cloud credentials. You can also update or delete the credentials when necessary.

When you add cloud credentials for your Amazon S3 account, you can choose one of the following authentication methods:

- Access secret key. To use this authentication type, you require an AWS Access Key and an AWS Secret key that you obtain from Amazon. Cloudera Manager stores these values securely and does not store them in world-readable locations. The credentials are masked and encrypted in the configurations passed to processes managed by Cloudera Manager, and redacted from the logs.
- IAM role. Amazon Identity and Access Management (IAM) can be used to create users, groups, and roles for use with Amazon Web Services, such as EC2 and Amazon S3. IAM role-based access provides the same level of access to all clients that use the role.

Important: You can choose the IAM role authentication type only when the following conditions are met:

- The source cluster is hosted on an AWS EC2 infrastructure.
- The source cluster Cloudera Manager and all the nodes in the cluster are running on an EC2 instance.
- The source cluster Cloudera Manager has the same IAM role.

For information about configuring AWS credentials, see Introduction to role based provisioning credential in AWS.

You can perform the following tasks on the Cloud Credentials page to manage cloud credentials:

Add cloud credentials

You can add cloud credentials for your S3 or ABFS account. For information about adding cloud credentials, see Working with Cloud Credentials.

Note: Unregistered credentials can impact the replication process. Credentials associated with a cluster node that do not have updated credentials are called unregistered credentials. For example, if a node is down when the credentials are changed on a bucket or when the node is brought up that has the old credentials.

Update cloud credentials

You can update the cloud credentials based on various factors. When the bucket configuration such as secret or access keys, bucket name or endpoint, and encryption type is changed, it can affect the Replication Manager replication policy run and might require an update to the Replication Manager cloud credentials.

Credential changes are picked up by the next run of the policy. When you change the credentials, the in-progress policy runs might fail but the succeeding runs pick up the changes.

To update a cloud credential, click Actions Update .

Delete cloud credentials

You can delete unwanted credentials from the Replication Manager. When you delete cloud credentials, the replication policies that use the deleted cloud credentials might fail. To avoid failures, delete the Replication Manager cloud policies associated with the deleted credentials and recreate the policies with the new credentials. You can view a list of policies associated with specific credentials on the **Cloud Credentials** page.

To delete a cloud credential, click Actions Delete .

Replication Policies page

The "Replication Policies" page shows the number of replication policies that are active, the number of policies that have been suspended, the number of policies that are in error state, and the total number of replication policies available in Replication Manager. The page also provides a detailed view about the replication policies.

The Replication Policies page shows a replication policy status dashboard which shows the following panels and the number of policies with the status:

- **Error** shows the number of replication policies associated with a cluster designated as Error on the Classic Clusters map. Click the number to understand the policy names, the names of the source and destination clusters, and which services are stopped on the source or destination cluster.
- Active replication policies that are in Submitted or Running state. This item is not actionable.
- **Suspended** replication policies that have been suspended by the administrator. This item is not actionable.
- **Total** number of running policies.

Click a replication policy to view more details about the policy. Click Actions ( ) to perform more actions on a replication policy.

Additionally, you can perform the following tasks on the **Replication Policies** page:

- Change the timezone, if required, using .
- View the list of unreachable clusters using .

Replication policy details

You can also view the following policy details on the Replication Policies page:

- Current policy Status .
- Policy Type shows HDFS, Hive, or HBase.
- Replication policy Name .
- Source cluster name.
- Destination cluster name.
- Jobs that were run for the replication policy and its current status.
- Duration or time taken to run the policy.
- Last Success timestamp of the last successful run.
- Next Run timestamp of the next scheduled run.

Optimize Replication Policies page performance

By default, the replication policies are loaded only partially on the **Replication Policies** page, therefore the page might display incomplete statistics about a job status. This is because the job history is necessary to decide whether a policy failed or succeeded. The replication policies with failed jobs might take a longer time to load.

You can change the page load behavior depending on your requirements using . Choose one of the following options to load the **Replication Policies** page faster by delaying to load the job history:

- Delay loading job history when it takes too long attempts to load the job history, but omits the load operation above a certain threshold. By default, Replication Manager uses this option.
- Never load job history minimizes the load on Cloudera Manager and maximizes Replication Manager performance.
- Always load job history ensures that the job history is always loaded for all the displayed replication policies.

Use Case

Sometimes, Replication Manager fails to reach a healthy Cloudera Manager when there is a temporary networking blip or when there is a load spike on Cloudera Manager. When a cluster becomes unreachable for Replication Manager, the cluster is placed in the list of unreachable clusters (the list appears when you click). Replication Manager retries to reach the cluster again after 20 minutes. After you confirm that the Cloudera Manager is healthy and expect it to be reachable by Replication Manager, you can force reload the **Replication Policies** page using to reconnect every cluster.

For more information, see Replication Policies page does not display all the replication policies.

How replication policies work

In Replication Manager, you create replication policies to establish the rules you want applied to your replication jobs. The policy rules you set can include which cluster is the source and which is the destination, what data is replicated, what day and time the replication job occurs, the frequency of job runs, and bandwidth restrictions.

The first time you run a job (an instance of a policy) with data that has not been previously replicated, Replication Manager creates a new folder or database and bootstraps the data. During a bootstrap operation, all data is replicated from the source cluster to the destination. As a result, the initial execution of a job can take a significant amount of time, depending on how much data is being replicated, network bandwidth, and so on. So you should plan the bootstrap operation accordingly.

After the bootstrap operation succeeds, an incremental data replication is automatically performed. This job synchronizes, between the source and destination clusters, any events that occurred during the bootstrap process. After the data is synchronized, the replicated data is ready for use on the destination. Data is in a consistent state only after incremental replication has captured any new changes that occurred during bootstrap.

Subsequent replication jobs from the same source location to the same target on the destination are incremental, so only the changed data is copied.

When a bootstrap operation is interrupted, such as due to a network failure or an unrecoverable error, the Replication Manager does not retry the job instead it runs the job at the next scheduled interval, if available. Therefore, if the bootstrap operation is interrupted, you must manually correct the issue and then run the policy.

When scheduling how often you want a replication job to run, you should consider the recovery point objective (RPO) of the data being replicated; that is, what is the acceptable lag time between the active site and the replicated data on the destination.

Related Information

HDFS replication policy

Hive replication policy

HBase replication policy

Replication policy considerations

You should take into consideration certain guidelines when creating or modifying a replication policy. It is important for you to understand the security restrictions and encryption policies within Replication Manager.

The guidelines you need to consider before you create or modify a replication policy includes:

Data security

To use an S3 or ABFS cluster for your policy, register your credentials on the Cloud Credentials page.

A user with access to the Replication Manager user interface has the ability to browse, within the Replication Manager UI, the folder structure of any clusters enabled for Replication Manager.

Therefore, users can view folders, files, and databases in the Replication Manager user interface, that they might not have access to in HDFS. Users cannot view from the Replication Manager UI the content of files on the source or destination clusters. Nor do these administrators have the ability to modify or delete folders or files that are viewable from the Replication Manager UI.

Policy properties and settings

Consider the recovery point objective (RPO) of the data being replicated when you schedule a replication policy. The RPO is the acceptable lag time between the active site and the replicated data on the destination. Ensure that the frequency is set so that a job finishes before the next job starts.

Jobs based on the same policy cannot overlap. If a job is not completed before another job starts, the second job does not execute and is given the status Skipped. If a job is consistently skipped, you might need to modify the frequency of the job.

Specify bandwidth per map, in MBps. Each map is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value.

Cluster requirements

- Pair the clusters before you include them in a replication policy.
- With a single cluster, you can replicate data on-premises to cloud.
- With a single cluster, you cannot replicate data on-premises to on-premises.
- If the clusters are Replication Manager-enabled, it appears in the Source Cluster or Destination or Data Lake Cluster fields in the Create Policy wizard. You must ensure that the clusters you select are healthy before you start a policy instance (job).

Hive restrictions

- When creating a schedule for a Hive replication policy, you should set the frequency so that changes are replicated often enough to avoid overly large copies.
- ACID tables, managed tables, storage handler-based tables such as Apache HBase, and column statistics are not replicated. Hive2 managed tables are converted to external tables after replication.

How temporary AWS credentials for replication policies works

Some deployments require temporary AWS session credentials to provide just-in-time, minimum required access to replicate data using replication policies. You can achieve this task using IDBroker. You can use temporary AWS credentials, through the IDBroker service, to replicate HDFS data, Hive external tables, and HBase data from Kerberized Cloudera Private Cloud Base 7.1.9 SP1 clusters or higher using Cloudera Manager 7.11.3 CHF7 or higher versions to S3 buckets using Cloudera Replication Manager.

You can also use the temporary AWS credentials to replicate the HDFS data from S3 buckets to Kerberized Cloudera Private Cloud Base 7.1.9 SP1 clusters or higher using Cloudera Manager 7.11.3 CHF7 or higher versions.

IDBroker is a REST API built as an extension of Apache Knox's authentication services. It allows an authenticated and authorized user to exchange a set of credentials or a token for short-lived cloud vendor access tokens.

To acquire the temporary AWS credentials, you create an IDBroker topology and then map the Kerberos users (or groups) to an AWS IAM Role. During the replication policy run, Replication Manager invokes IDBroker, and the IDBroker then uses the mapping between the on-premises Kerberized user and the IAM Role to request an AWS session token for that role.

Use case

An organization uses the same on-premises cluster across all their departments, and each department has its own AWS account so that it can replicate its required data from the on-premises cluster to its own AWS account when necessary. Each department depending on their requirements might either want to leverage the cloud storage capabilities to store data, or use the cloud processing capabilities to run workloads, analyze the data, or any other purposes.

Authentication methods to use AWS credentials in replication policies

You can choose long-term AWS cloud credentials or temporary AWS session credentials when you want to replicate HDFS data, Hive external tables, and HBase data from Cloudera Private Cloud Base clusters to S3 buckets on Cloudera Public Cloud.

Long-term cloud credentials

You can use long-term credentials to replicate data to the cloud using replication policies. To use long-term cloud credentials in a replication policy, you must:

- have an AWS account, and access key and secret key for it.
- register an external account in Cloudera Manager using AWS access key and AWS secret key.

  You can add an external account on the Cloudera Manager Administration External Accounts page. The external account serves as an authentication method during data replication, using replication policies, from Cloudera Private Cloud Base clusters to cloud.
- add the cloud credential in Cloudera Public Cloud Replication Manager.

The following use cases illustrate scenarios where you can use long-term AWS credentials:

- Environments where you have multiple users and multi-tenancy – In this instance, you can add an **Add Access Key Credentials** external account in Cloudera Manager for Cloudera Private Cloud Base cluster, add the cloud credentials in the Cloudera Replication Manager, and then create a replication policy.
- Single user cluster, or where all the users of the cluster have the same privileges to the data in Amazon S3 – In this instance, you can add **IAM role-based authentication** in Cloudera Manager for Cloudera Private Cloud Base cluster, add the cloud credentials in the Cloudera Replication Manager, and then create a replication policy.

Temporary AWS session credentials

You can use temporary AWS session credentials to provide just-in-time, minimum required access to replicate data using replication policies. Before you use temporary AWS session credentials in a replication policy, you must:

1. have an AWS account with an IAM role that has the required permissions to access the target S3 bucket and has the necessary trust relationships set up.
2. install and configure IDBroker on the Cloudera Private Cloud Base cluster.
3. add the cloud credential in Cloudera Replication Manager.

   Alternatively, you can add an external account for the IDBroker topology in Cloudera Manager.

HDFS replication policy

You can use the HDFS replication policies in Cloudera Replication Manager to replicate HDFS data. The HDFS replication policies can replicate HDFS data and metadata from classic clusters (CDH, Cloudera Private Cloud Base, and HDP) to Cloudera Public Cloud storage buckets such as S3 and ABFS, and from cloud storage to classic clusters (CDH or Cloudera Private Cloud Base clusters). To use an on-premises cluster (CDH or Cloudera Private Cloud Base cluster) in the replication policy, you must register it as a classic cluster in the Cloudera Management Console. To use the cloud storage for data replication, you must register the cloud credentials in Replication Manager so that the Replication Manager service can access the cloud storage. You must also verify cluster access and configure minimum ports for replication before you create HDFS replication policies.

Important: Before you create replication policies, see *Support matrix for Replication Manager on Cloudera Public Cloud* to verify whether your clusters are supported by Replication Manager.

You can also use CDP CLI commands to create HDFS replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see *CDP CLI for Replication Manager*.

Related Information

Support matrix for Cloudera Replication Manager

Using HDFS replication policies

CDP CLI for Cloudera Replication Manager

HDFS snapshots

You can schedule taking HDFS snapshots for replication in the Replication Manager. HDFS snapshots are read-only point-in-time copies of the filesystem. You can enable snapshots on the entire filesystem, or on a subtree of the filesystem. In Replication Manager, you take snapshots at a dataset level. Understanding how snapshots work and some of the benefits and costs involved can help you to decide whether or not to enable snapshots.

To improve the performance and consistency of HDFS replications, enable the HDFS replication source directories for snapshots, and for Hive replications, enable the Hive warehouse directory for snapshots. For more information, see HDFS snapshots.

Enabling snapshots on a folder requires HDFS admin permissions because it impacts the NameNode. When you enable snapshots, all the subdirectories are automatically enabled for snapshots as well. So when you create a snapshot copy of a directory, all content in that directory including the subdirectories is included as part of the copy. If a directory contains snapshots but the directory is no longer snapshot-enabled, you must delete the snapshots before you enable the snapshot capability on the directory.

Take snapshots on the highest-level parent directory that is snapshot-enabled. Snapshot operations are not allowed on a directory if one of its parent directories is already snapshot-enabled (snapshottable) or if descendants already contain snapshots.

For example, in the following directory tree image, if directory-1 is snapshot-enabled but you want to replicate subdirectory-2, you cannot select only subdirectory-2 for replication. You must select directory-1 for your replication policy.

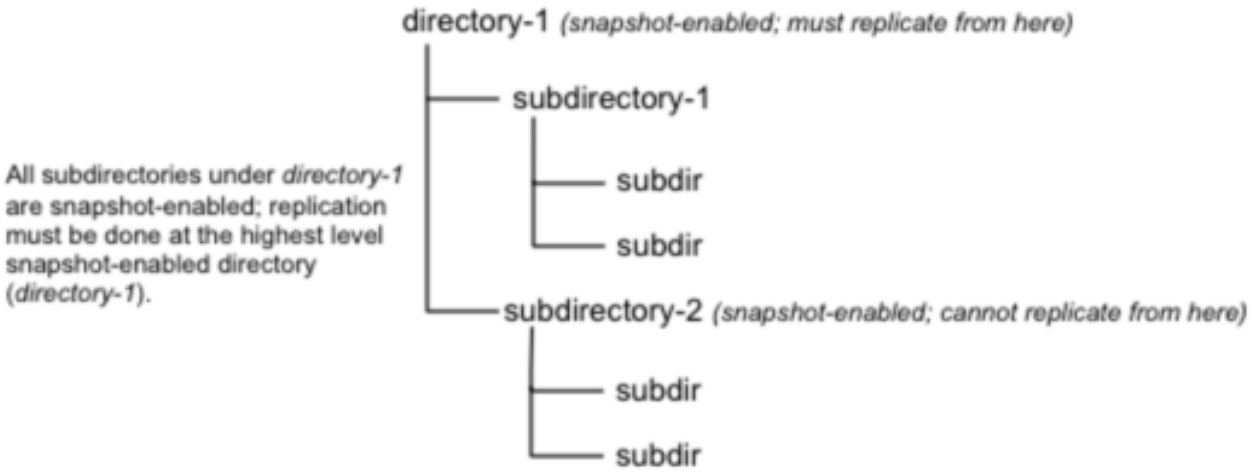

There is no limit to the number of snapshot-enabled directories you can have. A snapshot-enabled directory can accommodate 65,536 simultaneous snapshots. Blocks in datanodes are not copied during snapshot replication. The snapshot files record the block list and the file size. There is no data copying.

When snapshots are initially created, a directory named .snapshot is created on the source and destination clusters under the directory being copied. All snapshots are retained within the .snapshot directories. By default, the last snapshot of a file or directory is retained. Snapshots older than this are automatically deleted. You can configure the number of snapshots to retain when you create or edit an HDFS Snapshot Policy in the target Cloudera Manager using Cloudera Private Cloud Base Replication Manager 7.1.1 or higher or in Backup and Disaster Recovery 5.10 or higher depending on your on-premises target cluster.

Requirements and benefits of HDFS snapshots

You might want to consider the benefits and memory cost of using snapshots. Verify the requirements before you enable snapshots.

Requirements

You must have HDFS superuser privilege to enable or disable snapshot operations. Replication using snapshots requires that the target filesystem data being replicated is identical to the source data for a given snapshot. There must be no modification to the data on the target. Otherwise, the integrity of the snapshot cannot be guaranteed on the target and replication can fail in various ways.

Benefits

Snapshot-based replication helps you to avoid unnecessary copying of renamed files and directories. If a large directory is renamed on the source side, a regular DistCp update operation sees the renamed directory as a new one and copies the entire directory.

Generating copy lists during incremental synchronization is more efficient with snapshots than using a regular DistCp update, which can take a long time to scan the whole directory and detect identical files. And because snapshots are read-only point-in-time copies between the source and destination, modification of source files during replication is not an issue, as it can be using other replication methods.

A snapshot cannot be modified. This protects the data against accidental or intentional modification, which is helpful in governance.

Memory cost

There is a memory cost to enable and maintain snapshots. Tracking the modifications that are made relative to a snapshot increases the memory footprint on the NameNode and can therefore stress NameNode memory. Because of the additional memory requirements, snapshot replication is recommended for situations where you expect to do a

lot of directory renaming, if the directory tree is very large, or if you expect changes to be made to source files while replication jobs run.

Enabling and taking snapshots in Cloudera Manager

Before you take snapshots (in Cloudera Manager) or create snapshot policies (in Replication Manager) for HDFS directories, you must enable snapshots for the directories in Cloudera Manager.

Procedure

1. To enable snapshots for HDFS directories, navigate to the directory on the Cloudera Manager Clusters HDFS service File Browser tab, and click Enable Snapshots.

   Note: If you enable snapshots for a directory, you cannot enable snapshots for its parent directory. Snapshots can be taken only on directories that have snapshots enabled.

   Tip: To disable snapshots for a directory that has snapshots enabled, click Disable Snapshots. Ensure that the snapshots of the directory are deleted before you disable snapshots for the directory.

2. To take a snapshot of a directory or table, perform the following steps:
   a) Navigate to the directory or folder.
   b) Click Take Snapshot in the drop-down menu next to the full file path.
   c) Specify a unique name for the snapshot.

   The snapshot name with the timestamp of its creation appears in the snapshot list.

3. Click Actions Delete to delete a snapshot.

   Note: After you delete a snapshot, you can restore it, if required.

Hive replication policy

You can create a Hive replication policy in Cloudera Replication Manager after you configure the required Ranger policy in Ranger, register the on-premises cluster (CDH or Cloudera Private Cloud Base) as a classic cluster in Management Console, register cloud account credentials in the Replication Manager service, verify cluster access, and configure minimum ports for replication. The replication load happens on the source on-premises cluster. You can replicate data on-premises to the cloud with a single cluster if the Metastore is running on the cloud.

These policies support table-level replication and can replicate Hive external tables from on-premises clusters (CDH and Cloudera Private Cloud Base) to cloud storage such as S3 and ABFS and to Data Hubs. They also can:

- replicate data stored in Hive tables, Hive metadata, data in Hive metastore, and Impala metadata (catalog server metadata) associated with Impala tables registered in the Hive metastore, and

  Note: Hive2 managed tables are converted to external tables after replication.

- migrate Sentry permissions to Ranger.

  Note: To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.

Hive metadata replication involves multiple entities. Replication Manager supports replication of external tables in Hive. Hive supports replication of external tables to the target cluster and it retains all the properties of external tables. The data files permission and ownership are preserved so that the relevant external processes can continue to write in it even after failover.

Important: Before you create Hive replication policies, you must ensure that the required Ranger policy is set in Ranger and see Support matrix for Replication Manager on Cloudera Public Cloud to verify whether your clusters are supported by Replication Manager.

You can also use CDP CLI commands to create Hive replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see *CDP CLI for Replication Manager*.

The Apache Ranger access policy model consists of the following components:

- Specification of the resources that you can apply to a replication policy which includes the HDFS files and directories; Hive databases, tables, and columns; and HBase tables, column-families, and columns.
- Specification of access conditions for specific users and groups.

You must set the Ranger policy for the hdfs user on the target cluster to perform all operations on all databases and tables. The same user role is used to import Hive Metastore. The hdfs user should have access to all Hive datasets, including all operations. Otherwise, Hive import fails during the replication process.

On the target cluster, the hive user must have Ranger admin privileges. The same hive user performs the metadata import operation.

For more information about Hive replication policies to replicate data from CDH clusters to Cloudera Public Cloud, see Migrate Hive data from CDH to Cloudera Public Cloud blog.

Related Information

Support matrix for Cloudera Replication Manager

Using Hive replication policy

CDP CLI for Cloudera Replication Manager

Hive replication

Replication Manager allows you to replicate Hive databases from a source cluster to a target location on a destination cluster. The first time you run a job with data that has not been previously replicated, the Replication Manager creates a new folder or database and bootstraps the data. To replicate Hive metadata, Replication Manager performs a full replication. To replicate the data stored in Hive tables, Replication Manager uses snapshot diff-based replication to perform incremental replication.

During bootstrap operation, all of the data from the source location is copied to the destination. This bootstrapping of data can take hours to days, depending on factors such as the amount of data being copied and available network bandwidth.

After the bootstrap operation succeeds, an incremental replication is automatically performed for data replication using snapshot diff-based replication. The job synchronizes, between the source and destination clusters, any events that occurred during the bootstrap process. After the data is synchronized, the replicated data is ready for use on the destination. Data is in a consistent state only after incremental replication has captured any new changes that occurred during bootstrap.

Subsequent replication jobs from the same source location to the same target on the destination are incremental, so only the changed data is copied.

If a bootstrap operation is interrupted, such as due to a network failure or an unrecoverable error, the Replication Manager automatically retries the job. If a retry succeeds, the replication job continues from the point at which it was interrupted. If the automatic retries are not successful, you must manually correct the issue before running the policy again. When you activate the policy again, the replication job resumes from the point at which it was suspended.

Functions such as User Defined Functions (UDF) in Hive can be replicated. To enable this, you can create UDFs using a syntax. For example, the following sample code shows an UDF creation syntax:

```
CREATE FUNCTION [db_name.]function_name AS class_name  USING JAR|FILE|ARCHIV
E 'file_uri' [, JAR|FILE|ARCHIVE 'file_uri'] ;
```

Snapshot diff-based replication

By default, Replication Manager uses snapshot differences ("diff") to improve performance by comparing HDFS snapshots and only replicating the files that are changed in the source directory.

While Hive metadata requires a full replication, the data stored in Hive tables takes advantage of snapshot diff-based replication. To replicate a database using a Hive replication policy, ensure that all the HDFS paths for the tables in that database are either snapshottable or under a snapshottable root. For example, if the database that is being replicated has external tables, all the external table HDFS data locations should be snapshottable too. Failing to do so can cause the Replication Manager to fail to generate a diff report. Without a diff report, Replication Manager will not use snapshot diff.

An HDFS directory is referred to as snapshottable if an administrator - having superuser privilege or having owner access to the directory - has enabled snapshots for the directory in Cloudera Manager.

You must ensure that the following guidelines are met for efficient incremental replication:

- HDFS snapshots are immutable.

  Tip: In the source Cloudera Manager, go to Clusters *HDFS SERVICE* Configuration section, and search for Enable Immutable Snapshots.

- Snapshot root directory is set to as low in the hierarchy as possible.
- Replication Manager user is a super user or the owner of the snapshottable root. This is because the user specified in the Run-as-username field in the replication policy must have the permission to list the snapshots.
- Paths from both source and destination clusters in the replication policy are under a snapshottable root or are snapshottable for the replication policy to run using snapshot diff.

Replication Manager performs a complete replication when one or more of the following change: Delete Policy, Preserve Policy, Target Path, or Exclusion Path.

Note: Ensure the source data does not contain an encrypted subdirectory. This is because snapshot diff-based replication might fail if an encrypted subdirectory exists in the source data.

Hive tables

Managed tables are Hive owned tables where the entire lifecycle of the tables' data are managed and controlled by Hive. External tables are tables where Hive has loose coupling with the data. Replication Manager replicates external tables successfully to a target cluster, and the Hive2 managed tables are converted to external tables after replication.

Hive supports replication of external tables with data to target cluster and it retains all the properties of external tables. The data files' permission and ownership are preserved so that the relevant external processes can continue to write in it even after failover.

The writes on external tables are performed using the Hive SQL commands and the data files can also be accessed and managed by processes outside of Hive. If an external table or partition is dropped, only the metadata associated with the table or partition is deleted but the underlying data files stay intact. A typical example for an external table is to run analytical queries on HBase or Druid owned data using Hive, where the data files are written by HBase or Druid and Hive reads them for analytics.

Important: Hive Materialized Views replication is not supported. However, Replication Manager does not skip it during replication and the replicated data might not work as expected in the target cluster.

When you create a schedule for a Hive replication policy, set the frequency so that changes are replicated often enough to avoid overly large copies.

You might come across the following use cases during Hive replication:

Replication Manager upgrade use case

In a normal scenario, if you have external tables that are replicated as managed tables, after the upgrade process, you must drop those tables from the target cluster and set the base directory. In the next instance, they get replicated as external tables.

Conflicts in external tables' data location for multiple source clusters replication to the same target cluster

To handle the conflicts in external tables' data location for multiple source clusters replication to the same target cluster, the Replication Manager assigns an unique base directory for each source cluster under which the external tables' data from the corresponding source cluster is copied.

For example, if the external table location in a source cluster is /ext/hbase_data, then the location in the target cluster after replication is <base_dir>/ext/hbase_data. You can use the DESCRIBE TABLE command to track the new location of external tables.

Replication conflicts between HDFS and Hive external table location

When you run the Hive replication policy on an external table, the data is stored on the target directory at a specific location. Next, when you run the HDFS replication policy which tries to copy data at the same external table location, Replication Manager ensures that the Hive data is not overridden by HDFS.

For example, when you run a Hive replication policy on an external table, the policy creates a target directory /tmp/db1/ext1. When you run an HDFS replication policy, the policy should not override the data by replicating on the /tmp/db1/ext1 directory.

Conflicts during external tables replication process

Conflicts appear when two Hive replication policies on DB1 and DB2 (either from the same source cluster or different source clusters) have external tables that point to the same data location (for example, /abc) and are replicated to the same target cluster. To avoid such conflicts, you must set different paths for the external table base directory configuration, for both the policies.

For example, set /db1 for DB1 and /db2 for DB2. This ensures that the target external table data location is different for both databases. For example, /db1/abcd and /db2/abcd.

Note: Replication conflicts are not supported for on-premises to cloud scenario.

Hive cloud replication

Replication Manager supports replication of the Hive database from a cluster with underlying HDFS to another cluster with cloud storage. It uses push-based replication, with the replication job running on the cluster with HDFS.

Hive stores its metadata in Hive Metastore, but the underlying data is stored in HDFS or cloud storage. In a Hadoop cluster with Hive service, the Hive warehouse directory can be configured with either HDFS or cloud storage.

You can perform the following tasks with Hive replication:

- Rename the dataset in the policy that is replicated.
- Create a pull-based policy on the source cluster to move data from the target back into the source cluster Hive database.

Hive replication from an HDFS-based cluster to a cloud storage-based cluster requires the following components:

- Source cluster - The cluster with a Hive warehouse directory on local HDFS. This can be an on-premises cluster or an IaaS cluster with data on local HDFS. The required services are HDFS, YARN, Hive, Ranger, and Knox.
- Destination cluster - The cluster with data on cloud storage. The cluster minimally requires Hive Metastore, Ranger, and Knox.

Replication Manager does not manage Ranger policies, and Personally Identifiable Information (PII) or any other secure data that gets replicated from on-premises to Amazon S3. You must manage these items outside of Replication Manager.

Table-level replication

To enable table-level replication, you must specify the list of tables to be replicated in a given replication policy. Table-level replication enables you to replicate just the critical tables. It also helps you to speed-up the replication process and also reduces network bandwidth utilization.

You can define table-level replication using regular expressions. You can include or exclude tables in a database for Hive replication during the Hive replication policy creation process.

The following examples illustrate how you can include or exclude Hive tables in the Hive replication policy:

- To include only table1, table2, and table3 in a database for replication, enter the database name in the Database field, and then enter table1|table2|table3 in the Tables field.
- To exclude table5, table7, and table9 and include the rest of the tables in the database, enter the database name in the Database field, and then enter (?!table5|table7|table9).+ in the Tables field.

Limitations using table-level replication

- If a table is dynamically added for replication due to changes in regular expression or added to the include list, the tables' data may not be point-in-time consistent with other tables which are already replicated incrementally. However, this inconsistency is seen for a very small duration until the completion of the next incremental replication after tables are added in the bootstrapped manner.
- Hive does not support overlapping replication policies such as db., db.[t1], and *. to the same target database but works as expected if the target databases are different.

Migrate Sentry authorization policies into Ranger

During Hive replication, Replication Manager migrates Sentry authorization policies into Ranger as part of the replication policy.

The Sentry service serves authorization metadata from the database-backed storage but does not handle actual privilege validation. The Hive and Impala services are clients of this service and it enforces Sentry privileges when the services are configured to use Sentry. Replication Manager allows administrators to migrate the existing Sentry permissions from the source CDH cluster to the Ranger policies in Cloudera Public Cloud.

When you create a replication policy, you can choose to migrate the Sentry policies for the resources that you want to migrate. During replication policy job run, the resources and its Sentry policies are migrated to the destination cluster. To migrate the Sentry policies for the resources, select the Include Sentry Permissions with Metadata option in the Additional Settings page of the Create Replication Policy wizard.

Note:

To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.

The Sentry Permissions section of the Create Replication Policy wizard has the following options:

- Include Sentry Permissions with Metadata migrates the Sentry permissions during the replication job run.
- Exclude Sentry Permissions with Metadata ensures that the Sentry permissions are not migrated during the replication job.
- Choose Skip URI Privileges if you do not want to include URI privileges when you migrate Sentry permissions. During migration, the URI privileges are translated to point to an equivalent location in S3. If the resources have a different location in Amazon S3, do not migrate the URI privileges because the URI privileges might not be valid.

The following image shows the Sentry Permissions section in the Create Replication Policy wizard:

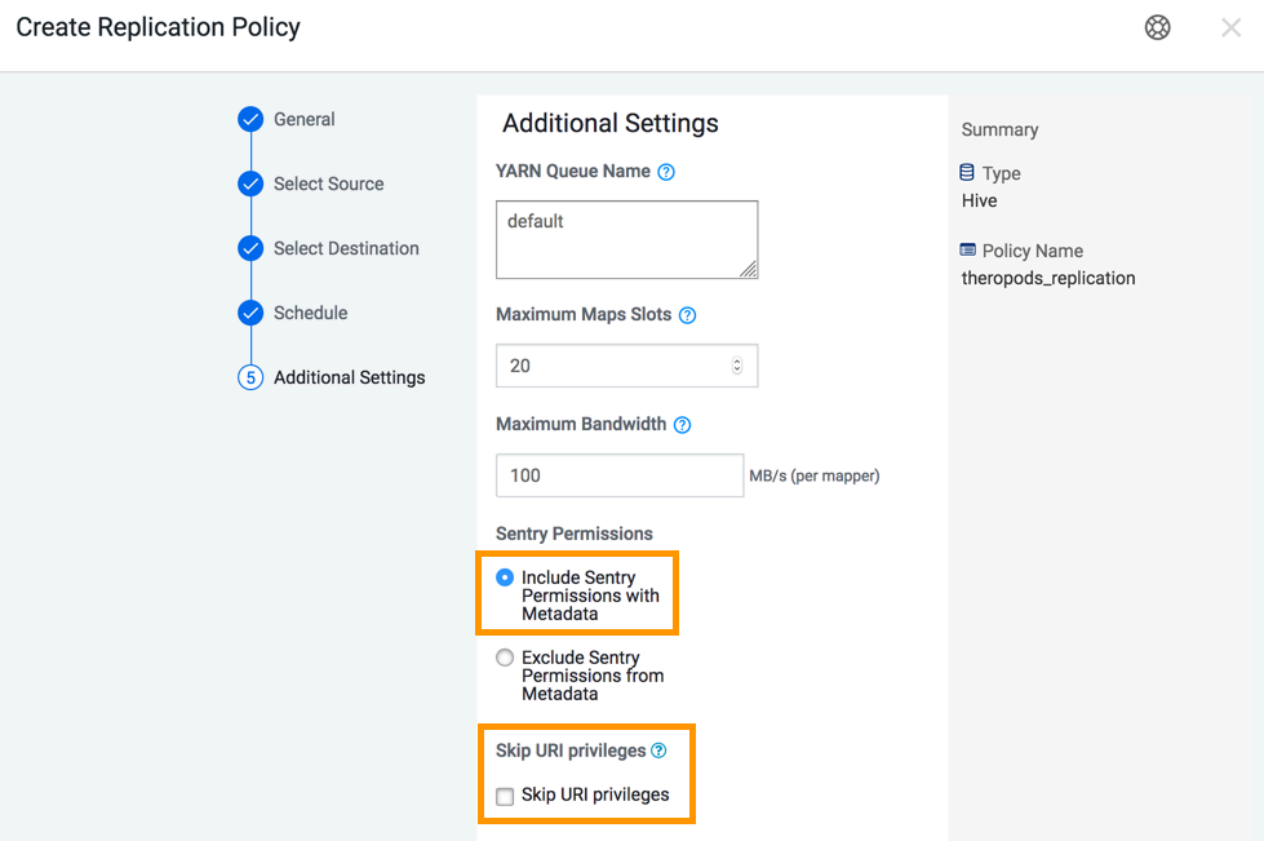

The following steps are completed during the migration of Sentry policies into Ranger:

- The Export operation runs in the source cluster. During this operation, the Sentry permissions are fetched and exported to a JSON file. This file might be in a local file system or HDFS or S3, based on the configuration.
- The Translate and Ingest operations take place on the target cluster. In the translate operation, Sentry permissions are translated into a format that can be read by Ranger. The permissions are then imported into Ranger. When the permissions are imported, they are tagged with the source cluster name and the time that the ingest took place. After the import, the file containing the permissions is deleted.

Note: During Hive replication from an on-premises CDH cluster to a cloud cluster, the Replication Manager migrates Sentry authorization policies into Ranger as part of the replication policy. However, no import operation is initiated if the end service in the cloud cluster (AWS) is Sentry.

A Ranger policy is created for each resource, such as a database, table, or column. The policy name is derived from the resource name. For example, if the resource is Database:dinosaurs, table= theropods, then the derived policy name is database=dinosarus->table=theropods.

The priority for migrated policies is set to normal in Ranger. The normal priority allows you to create another policy for the same resource that overrides the policy that is imported from Sentry.

Sentry to Ranger permissions

There are no one-to-one mapping between Sentry privileges and Ranger service policies, therefore the Sentry privileges are translated to their equivalents within the Ranger service policies.

The following list illustrates how the Sentry privileges appear in Ranger after the migration:

- Sentry permissions that are granted to roles are granted to groups in Ranger.
- Sentry permissions that are granted to a parent object are granted to the child object as well. The migration process preserves the permissions that are applied to child objects. For example, a permission that is applied at the database level also applies to the tables within that database.
- Sentry OWNER privileges are translated to the Ranger ALL privilege.

- Sentry OWNER WITH GRANT OPTION privileges are translated to Ranger ALL with Delegated Admin checked.
- Sentry does not differentiate between tables and views. When view permissions are migrated, they are treated as table names.
- Sentry privileges on URIs uses the object store location as the base location.

The table below shows how actions in Sentry are applied to the corresponding action in Ranger:

Table 1: Sentry Actions to Ranger Actions

Sentry Action	Ranger Action
SELECT	SELECT
INSERT	UPDATE
CREATE	CREATE
REFRESH	REFRESH
ALL	ALL
SELECT with Grant	INSERT
INSERT with Grant	INSERT
CREATE with Grant	CREATE
ALL with Grant	ALL with Delegated Admin Checked

HBase replication policy

You can replicate HBase and Phoenix tables using HBase replication policies in Cloudera Replication Manager. An HBase replication policy replicates the data at table-level granularity. After you create an HBase replication policy, you can delete one or more tables from the policy.

The replication policy replicates the data in the specified tables and continues to replicate the generated data (that is, future changes in data) unless you suspend the policy or delete the tables. You can replicate only existing HBase data, generated HBase data, or both depending on your requirements. You also can choose to replicate all the HBase tables or only the required tables in a database.

Before you create an HBase replication policy, you must:

- verify whether your clusters are supported by Replication Manager.
- understand how first-time setup configuration works.
- understand how cluster pairing works.
- understand the available methods to replicate HBase data.

You can also replicate HBase data simultaneously between multiple clusters.

Supported clusters for HBase replication policies

Before you create HBase replication policies, you must verify whether your clusters are supported by Replication Manager.

The following image shows a high-level view of the support matrix for HBase replication policies. You must consult the Support matrix for Replication Manager on Cloudera Public Cloud for the complete list of supporting clusters and scenarios:

Figure 1: High-level replication scenarios supported by HBase replication policies

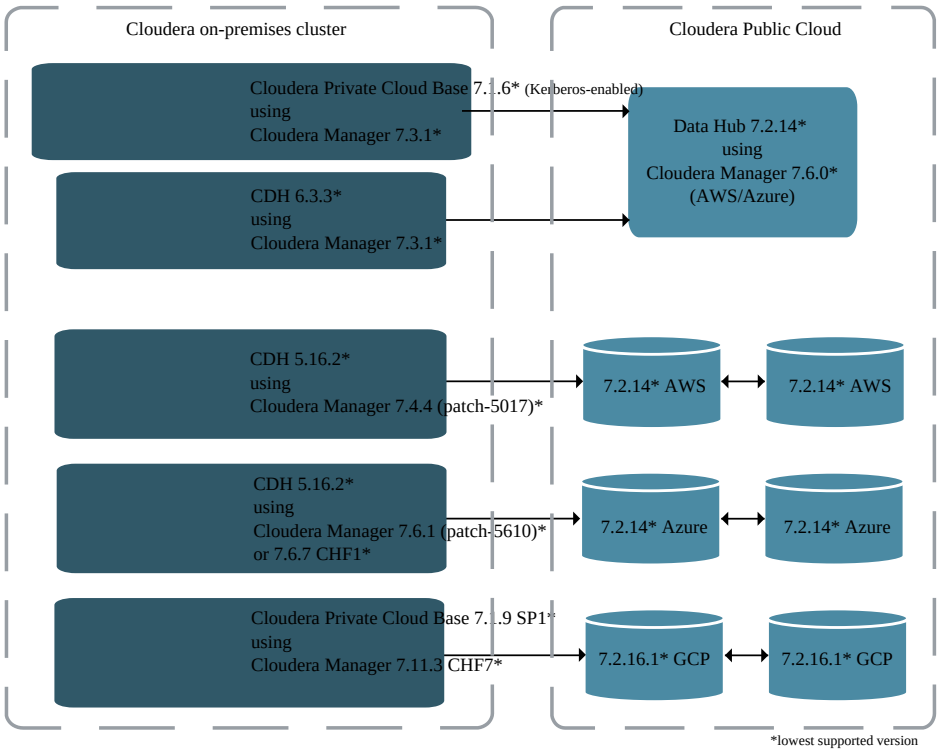

How HBase replication policies work

After you create the first HBase replication policy between a source cluster and target cluster, the Replication Manager service starts several background processes. After the processes complete, the service initiates the HBase data replication. One of the main background process is the first-time setup configuration.

- Understanding first-time setup configuration process
- What is a cluster pair

Steps in first-time setup configuration process

When you create the first HBase replication policy to replicate HBase data from a source cluster to a target cluster, the Replication Manager completes the first-time setup configuration steps and then replicates the data. The first-time setup configuration between a cluster pair is a one-time process, therefore subsequent HBase replication policies for the same cluster pair (ClusterA and ClusterB) do not require a first-time setup.

The first-time setup configuration completes the following steps:

1. Creates peers between the source (ClusterA) and target (ClusterB), that is, creates a cluster pair between ClusterA and ClusterB.
2. Initiates the required configuration changes. One of the steps ensures that both the clusters use the same credentials.jceks file.
3. Restarts the HBase service on both the clusters.

Important: If you are replicating HBase data from on-premises cluster to Cloudera Operational Database (COD) or Data Hub, you must manually restart HBase service on the source cluster.

When HBase replication policies are created, modified, or deleted simultaneously, Replication Manager processes each operation independent of each other.

Note: When the first-time setup between two clusters is in progress, you can create HBase replication policies between them. However, you cannot use one of these clusters with another cluster to create an HBase replication policy (in multi-cluster replication scenario) until the first-time setup is complete.

What is a cluster pair

The first-time setup configuration creates peers between the source (ClusterA) and target (ClusterB), that is, creates a cluster pair between ClusterA and ClusterB. If a cluster pair (ClusterA and ClusterB) has one or more active/suspended HBase replication policies between them, you cannot pair either of the clusters with another cluster.

You can use ClusterA or ClusterB with another cluster in an HBase replication policy only if the following conditions are true:

- ClusterA and ClusterB do not have existing HBase replication policies.
- All existing active/suspended HBase replication policies are deleted.

A warning appears on the **Select Destination** page during HBase replication policy creation if you choose ClusterA or ClusterB as source or target (when no HBase replication policies exist between them) with another cluster or if the other cluster in the cluster pair is unreachable. To continue HBase replication policy creation, you must acknowledge the forced re-pairing of the clusters.

Related Information

Support matrix for Cloudera Replication Manager
Using HBase replication policy

Methods to replicate HBase data

You can replicate only existing HBase data, generated HBase data, or both depending on your requirements. You also can choose to replicate all the HBase tables or only the required tables in a database.

When you create an HBase replication policy, you can choose one or more of the following methods to replicate HBase data depending on your requirements:

- Replicate only the generated data from chosen tables.
- Replicate existing data and generated data from chosen tables.
- Replicate existing tables and future tables in the database.
- Replicate existing data and generated data from chosen tables and future tables

Replicate only the generated data from chosen tables

In this method, you choose one or more tables during the replication policy creation process. Replication Manager replicates only the data that is generated after policy creation.

Replicate existing data and generated data from chosen tables

In this method, you choose one or more tables, and also choose the Select Source Perform Initial Snapshot option during the HBase replication policy creation process. Replication Manager replicates the existing data and the data that is generated after policy creation.

For example, you have two tables named 'Orders' and 'Customers' in the source cluster and you want to copy the data from these tables from March 1, 2021 onwards. To accomplish this task, you create an HBase replication policy without choosing the Perform Initial Snapshot option in the **Create Replication Policy** wizard on March 1, 2021. The data that you create, update, or delete in the source cluster after you created the HBase replication policy is automatically replicated to the target cluster.

Replicate existing tables and future tables in the database

In this method, you choose the Select Source Replicate Database option during the HBase replication policy creation process. Replication Manager replicates the generated data from the existing tables, and it replicates data from the future tables that are created after the HBase replication policy creation process is complete.

To replicate data from the future tables successfully, you must create similar empty tables on the target cluster. You can perform this action when you create or add a table to the database on the source cluster.

You can choose the Replicate Database option only if the following conditions are true:

- Target Cloudera Manager version is 7.11.0 or higher.
- Source cluster's CDH version is 6.x or higher.

  CDH 5.16.2 and higher versions also support the Replicate Database option after you upgrade the source cluster Cloudera Manager.
- No existing HBase replication policies exist between the source and target clusters.

  Tip: If you want to replicate the new tables that are created after the replication policy creation is complete, you must configure the replication scope to "1" for those tables on the source cluster.

  To configure the replication scope for a table on the master cluster, run the alter *[***TABLENAME***]*, {NAME => *[***COLUMN FAMILY***]*, REPLICATION_SCOPE => 1} command for each column family that must be replicated. *REPLICATION_SCOPE* is a column-family level attribute, where the value '0' means replication is disabled, and '1' means replication is enabled.

After you select the Select Source Replicate Database option in the HBase replication policy wizard, you can choose one of the following options to determine the tables in the database to replicate:

- Replicate all user tables - Replicates all the HBase tables in the database after the replication scope of the tables are set to 1.
- Replicate only tables where replication is already enabled - Replicates only those tables for which the replication scope is already set to 1.

  This option is supported only if the target cluster Cloudera version is 7.2.17.300 using Cloudera Manager 7.11.0-h3 or higher versions or Cloudera version 7.2.16.500 using Cloudera Manager 7.9.0-h7 or higher versions, or Cloudera version 7.12.0.0.

Replicate existing data and generated data from chosen tables and future tables

In this method, you choose the Perform Initial Snapshot and Replicate Database options on the **Select Source** page during the HBase replication policy creation process. You can also choose to replicate all the tables in the database or only those tables for which the replication scope is already set to 1. Replication Manager replicates the existing and generated data from the existing tables in addition to the data in future tables.

Replicate HBase data simultaneously between multiple clusters

Starting from Cloudera Public Cloud version 7.2.16.500, 7.2.17.200, and 7.2.18, you can create multiple HBase replication policies between multiple clusters to replicate HBase data. You must consider the limitations before you create a multi-cluster replication scenario. You can use the multi-cluster replication scenario for various use cases.

- How multi-cluster HBase replication works
- Limitations
- Sample use cases

How multi-cluster HBase replication works

The first-time setup configuration consists of several steps of which one step is to ensure that the source and target cluster use the same credentials.jceks file. Therefore, if multiple supported clusters share the same credentials.jceks file, you can replicate HBase data between them seamlessly using HBase replication policies.

The following image shows a sample multi-cluster HBase data replication scenario and a few possible directions of replication:

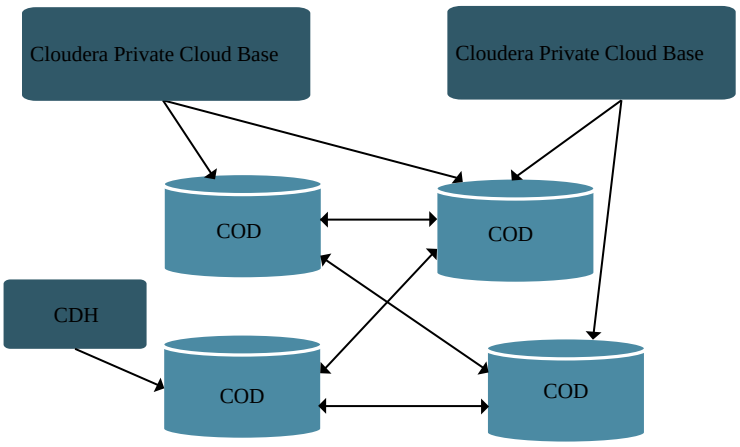

It is recommended that you do not replace the credentials.jceks file manually to create a multi-cluster HBase replication scenario. This is because when you create the first HBase replication policy between a pair of clusters, Replication Manager triggers the first-time setup process during which the credentials.jceks file in both the clusters get synchronized as required for HBase data replication.

Limitations

Consider the following limitations before you replicate the HBase data between multiple clusters using HBase replication policies:

- An HBase replication policy in a multi-cluster HBase replication setup fails when you use clusters that are part of another independent replication setup. This is because the clusters use a different credentials.jceks file. To use these clusters, you must break the cluster pairing and then create the required HBase replication policies.

  Tip: To break the cluster pairing, use the `POST /dmx/api/clusters/<target cluster crn>/hbase/resetFirstTimeSetup` API with the `{"sourceCluster": "<source cluster data center>$<source cluster name>"}` payload.

  Monitor the growing multi-cluster replication network so that it does not get disconnected. This ensures that the credentials.jceks file is the same on all clusters, the replication setup is always consistent, and no existing replication scenarios have to be reset.
- The Replication Manager UI does not allow the HBase replication policy creation to proceed if you choose a cluster (as source or target) that is in another first-time setup process. In this instance, you can wait for a few minutes to allow the first-time setup to complete and then create the HBase replication policy.

  When you create the first HBase replication policy between two clusters, the first-time setup configuration is initiated. After the configuration completes, the HBase data replication is initiated.

  Important: In a multi-cluster replication scenario, when you use a source on-premises cluster, you must manually restart the HBase service on the cluster after the first-time setup configuration completes. However, the next time you choose the same source cluster with another target cluster, the manual restart is not required. The same is true for the automatic HBase service restart in Data Hubs where the restart is performed only when the first replication is created with a particular cluster.

- The following conditions must be met to use the IDBroker credentials to create multiple HBase replication policies between multiple clusters when the target COD clusters are in separate AWS accounts or when a single AWS Role does not have access to all the required S3 buckets for all HBase target clusters:
  - Use Cloudera Public Cloud 7.2.18.200 or higher versions.
  - Choose the Perform Initial Snapshot option, and then specify the custom username in the Export snapshot user field in the **Select Source** page during the HBase replication policy creation process.

Use cases

Some use cases where you can use the multi-cluster HBase replication scenarios are illustrated below:

- Multiple source clusters and a single target cluster. You might have a disaster-recovery use case where you want to use a single COD to back up all the HBase data. The following image illustrates this scenario:

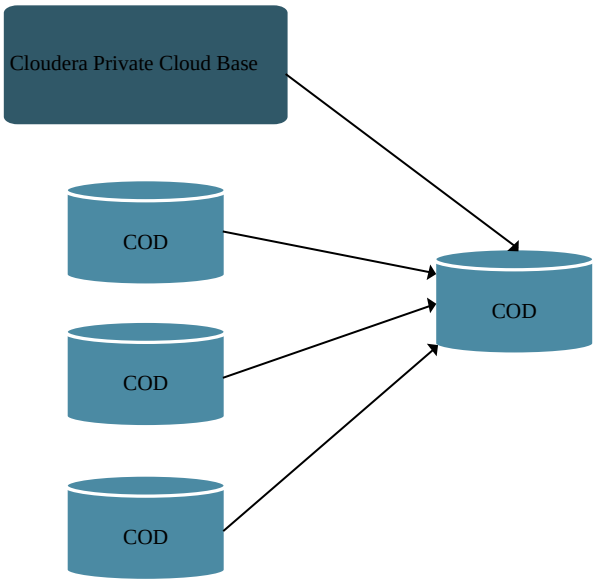

- Single source cluster and multiple target clusters. You might have a use case where all the HBase data is located in a cluster and you want to replicate only specific HBase tables to different environments to fulfill specific requirements. For example, QE environments and/or experimentation use case. The following image illustrates this scenario:

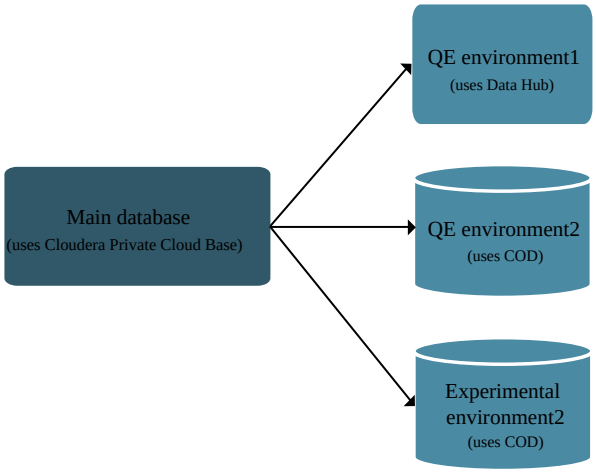

Cloudera Replication Manager

Replication Manager Operations

Date published: 2019-11-07
Date modified: 2024-07-08

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Replication Manager 4
- Support matrix for Cloudera Replication Manager 4
  - List of features supported by Cloudera Replication Manager 4
  - Replicate data from Cloudera Private Cloud Base and Cloudera Public Cloud source clusters 7
  - Replicate data from CDH and HDP source clusters 9
- Cloud credentials to use in Cloudera Replication Manager 10
  - Registering Amazon S3 cloud account in Replication Manager 11
  - Register Azure cloud credentials in Replication Manager 13
  - Registering GCP credentials to use in Replication Manager 15
  - Add IDBroker to use temporary AWS session credentials 20
- Ports for Cloudera Replication Manager 24
  - Ports required for HBase replication policies 29
- Verify cluster requirements for replication policies 29

Using HDFS replication policies 30
- Preparing to create an HDFS replication policy 30
- Creating HDFS replication policy 34
- Manage and monitor HDFS replication policies 39

Using Hive replication policies 42
- Preparing to create a Hive replication policy 43
- Creating Hive replication policy 47
- Manage and monitor Hive replication policies 53

Using HBase replication policies 55
- Preparing to create an HBase replication policy 56
- Creating HBase replication policy 60
- Manage and monitor HBase replication policies 68
  - Monitor HBase replication policy job details 68
  - Creating triggers and monitoring replication-related metrics in Cloudera Manager 70
  - Monitor HBase RegionServer replication peer metrics in Replication Manager 72
  - Viewing HBase RegionServer replication peer metrics 72

Troubleshooting replication policies in Cloudera Replication Manager 73
- Different methods to identify errors related to failed replication policy 73
- Replication Policies page does not display all the replication policies 75
- HDFS replication policy fails due to export HTTPS_PROXY environment variable 77
- Cannot find destination clusters for HBase replication policies 77
- HBase replication policy fails when Perform Initial Snapshot is chosen 78
- Optimize HBase replication policy performance when replicating HBase tables with several TB data 78
- Partition metadata replication takes a long time to complete 79
- Replicating Hive nested tables 79
- Target HBase folder is deleted when HBase replication policy fails 80
- Replicate HBase data in existing and future tables 83
- Sentry to Ranger migration fails 83

Cloudera Replication Manager

Replication Manager is a service in Cloudera Public Cloud. You can create replication policies in Replication Manager to copy or migrate HDFS, Hive external tables, and HBase data between on-premises clusters (CDH, Cloudera Private Cloud Base, or HDP) and Cloudera Public Cloud(Amazon S3 and Microsoft Azure ADLS Gen2 (ABFS)). Replicating Hive managed tables using Replication Manager from HDP clusters to Cloudera Public Cloud is a beta feature and is not available for general use.

Important: Before you create replication policies, see Support matrix for Replication Manager on Cloudera Public Cloud to verify whether your clusters are supported by Replication Manager.

Before you create a replication policy, you must verify whether the clusters are supported, register the on-premises clusters (CDH or Cloudera Private Cloud Base) as classic clusters in Cloudera Management Console, register the cloud credentials in Replication Manager, and verify whether the minimum port configuration is configured.

After you create a replication policy, you can perform various actions on the replication policy. You can monitor the replication jobs, view job history, and perform actions on the replication job.

Related Information

Introduction to Cloudera Replication Manager
Support matrix for Cloudera Replication Manager
Using HDFS replication policy
Using Hive replication policy
Using HBase replication policy
Troubleshooting replication policies in Cloudera Replication Manager
CDP CLI for Replication Manager

Support matrix for Cloudera Replication Manager

You can use Replication Manager or other alternate replication methods to replicate HDFS, Hive external tables, and HBase data between on-premises clusters (CDH clusters, Cloudera Private Cloud Base clusters, HDP clusters) and Cloudera Public Cloud (Amazon S3 (AWS), Microsoft Azure ADLS Gen2 (ABFS), and Google Cloud Platform (GCP)) clusters. Replication Manager from HDP clusters to Cloudera Public Cloud Azure is a beta feature and is not available for general use.

Note: Before you create replication policies, you must verify whether the on-premises cluster versions are supported by Replication Manager, register the on-premises clusters as Classic Clusters in the Cloudera Management Console, register the cloud account credentials in the Replication Manager service, verify cluster access, and configure minimum ports for replication.

See the other sections in this topic for the supported cluster and runtime versions.

- Replication policies and features.
- Cloudera Private Cloud Base and Cloudera Public Cloud source clusters that Replication Manager supports.
- CDH and HDP source clusters that Replication Manager supports.

Related Information

Introduction to Cloudera Replication Manager
Cloudera Operational Database data replication

List of features supported by Cloudera Replication Manager

Replication Manager provides replication policies that you can create, edit, and manage to accomplish your data replication goals. You can use other alternate replication methods for scenarios that Replication Manager does not support. Certain features in Cloudera Replication Manager are available only if the source and target clusters'

Cloudera Manager versions support the feature. Verify whether your source and target cluster's Cloudera Manager version support the required feature.

Supported replication policies

You can use the following replication policies in Cloudera Replication Manager:

- HDFS replication policies
- Hive replication policies
- HBase replication policies

HDFS replication policies

Replicate HDFS data and metadata from:

- on-premises clusters (CDH, Cloudera Private Cloud Base, and HDP) to cloud storage.
- cloud storage to Classic Clusters (CDH or Cloudera Private Cloud Base clusters).

You can choose the frequency during policy creation to replicate the data.

Hive replication policies

Support table-level replication, and can replicate Hive external tables from on-premises clusters (CDH and Cloudera Private Cloud Base) to cloud storage and to Data Hubs. The replication policies can also:

- replicate data stored in Hive tables, Hive metadata, data in Hive metastore, and Impala metadata (catalog server metadata) associated with Impala tables registered in the Hive metastore, and

  Note: Hive2 managed tables are converted to external tables after replication.

- migrate Sentry permissions to Ranger.

  Note: To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.

You can choose the frequency during policy creation to replicate the data.

HBase replication policies

Replicate HBase data from a source classic cluster (CDH or Cloudera Private Cloud Basecluster), COD, or Data Hub to a target Data Hub or COD cluster. You can also copy or replicate HBase data between different environments within a Virtual Private Cloud (VPC) using these policies.

Table 1: Supported cluster and runtime versions for HBase replication policies

Source Cluster Type	Lowest Supported Source CDH/ Cloudera Version	Lowest Supported Source Cloudera Manager Version	Target Cluster Type	Lowest Supported Target Cloudera Version	Lowest Supported Target Cloudera Manager Version
CDH	5.16.2	7.4.4 (patch-5017)	COD (AWS)	7.2.14	-
CDH	5.16.2	• 7.6.1 (patch-5610) • 7.6.7 CHF1 and higher	COD (Azure)	7.2.14	-
CDH	6.3.3	7.3.1	Data Hub in Cloudera Public Cloud AWS/ Azure	7.2.14	7.6.0

Source Cluster Type	Lowest Supported Source CDH/ Cloudera Version	Lowest Supported Source Cloudera Manager Version	Target Cluster Type	Lowest Supported Target Cloudera Version	Lowest Supported Target Cloudera Manager Version
Cloudera Private Cloud Base	7.1.6*	7.3.1	Data Hub in Cloudera Public Cloud AWS/ Azure	7.2.14	7.6.0
Cloudera Private Cloud Base	7.1.9 SP1	7.11.3 CHF7	GCP	7.2.16.1	-
COD (AWS/ Azure)	7.2.14	-	COD (AWS/ Azure)	7.2.14	-
COD (GCP)	7.2.16.1**	-	COD (GCP)	7.2.16.1**	-
COD (GCP)	7.2.16.500 7.2.17.300 7.2.18.0	-	COD (GCP)	7.2.16.500 7.2.17.300 7.2.18.0	-

*Cloudera Private Cloud Base 7.1.6 and higher clusters must be Kerberos enabled to use them as source classic clusters in an HBase replication policy.

**You must add key-value pairs to register a Google account to use in Replication Manager. For more information about the key-value pairs, see Preparing to create an HBase replication policy.

Important:

- You can replicate HBase data in SFT-enabled clusters for target clusters with version Cloudera Public Cloud 7.2.16 or higher using HBase replication policies.
- You can replicate Phoenix tables from CDH 5.16.2 and higher using Cloudera Manager 7.4.4 and higher to COD using HBase replication policies.
- You cannot create HBase replication policies if the target Cloudera version is 7.2.16, 7.2.16.1, 7.2.16.2, 7.2.16.3, or 7.2.16.200, the source Cloudera Manager version is 7.7.3 or lower, and the source Cloudera Manager API version is v50 or higher.

  Tip: The endpoint http://*[***CM HOST***]*:*[***CM PORT***]*/api/version shows the API version of the Cloudera Manager.

- HBase replication policies do not support source or destination Cloudera Public Cloud clusters created using the Micro Duty template.

HBase replication policies replicate all the data from the specified tables and then continue to replicate the changed data automatically without user intervention.

Note: Alternate replication methods:

- Use the replication plugin for HBase data. For more information, see COD replication in a Nutshell, Cloudera replication plugin, and HBase data replication.
- Contact Cloudera Support for Hive external tables.

Supported features

The following table lists the features and the Cloudera Manager instances that are required for source clusters and target clusters to use the features:

Feature	Lowest supported source Cloudera Manager version	Lowest supported target Cloudera Manager version
Register the GCP credentials to use in Replication Manager on the **Cloud Credentials** page.	• 7.9.0-h7 and higher • 7.11.0-h3 and higher • 7.12.0.0 and higher	Supports all Cloudera Public Cloud Cloudera Manager versions.

Feature	Lowest supported source Cloudera Manager version	Lowest supported target Cloudera Manager version
Replicate HBase data simultaneously between multiple clusters*.	• 7.9.0-h7 and higher • 7.11.0-h2 and higher • 7.12.0.0 and higher	• 7.9.0-h7 and higher • 7.11.0-h2 and higher • 7.12.0.0 and higher
Replicate only those HBase tables where the replication scope is already enabled using the Select Source Replicate only tables where replication is already enabled * option during the HBase replication policy creation process.	Supports all Cloudera Public Cloud Cloudera Manager versions.	• 7.9.0-h7 and higher • 7.11.0-h3 and higher • 7.12.0.0 and higher
Specify the network load balancer (NLB) Endpoint after you enable the Select Destination Replicate via a Network Load Balancer* option during the HBase replication policy creation process if the on-premises cluster uses NLB to communicate with the COD clusters.	CDH 5.16.2	7.12.0.100
Specify the YARN queue bandwidth using the Initial Snapshot Settings Maximum Bandwidth * option during the HBase replication policy creation process to export the HBase initial snapshot.	• 7.9.0-h7 and higher • 7.11.0-h3 and higher • 7.12.0.0 and higher	• 7.9.0-h7 and higher • 7.11.0-h3 and higher • 7.12.0.0 and higher
Enter Initial Snapshot Settings Maximum parallel snapshots* to specify the maximum number of tables to process in parallel during the initial snapshot export and import step for an HBase replication policy. If you do not enter any value, Replication Manager chooses an appropriate value, depending on the resources in the source and target cluster, to optimize the performance.	Supports all Cloudera Public Cloud Cloudera Manager versions.	• 7.9.0-h7 and higher • 7.11.0-h3 and higher • 7.12.0.0 and higher
Add IDBroker credentials* to use in Replication Manager on the **Cloud Credentials** page.	7.11.3 CHF7	7.11.3 CHF7
Enter the Select Source Export snapshot user * field during the HBase replication policy creation process to specify the username to export the initial snapshot to the target.	7.11.3 CHF7	7.11.3 CHF7
*To enable this feature, contact your Cloudera Account team.		

Related Information

Introduction to Cloudera Replication Manager
HDFS replication policy
Using HDFS replication policy
Hive replication policy
Using Hive replication policy
HBase replication policy
Using HBase replication policy
Working with cloud credentials

Replicate data from Cloudera Private Cloud Base and Cloudera Public Cloud source clusters

Replication Manager replicates HDFS (Cloudera Private Cloud Base source clusters and Cloudera Public Cloud storage on AWS and Azure), Hive external tables (Cloudera Private Cloud Base source clusters), and HBase (Cloudera Private Cloud Base source clusters) data to Cloudera Public Cloud (Amazon S3 and Microsoft Azure

ADLS Gen2 (ABFS)) clusters. You can use the replication plugin as an alternate replication method to replicate HBase data for scenarios that are not supported by Replication Manager.

The following tables list the minimum source and destination cluster versions, minimum Cloudera Manager versions, supported cloud providers, and supported scenarios:

Replicate data from Cloudera Private Cloud Base source clusters

Source cluster	Lowest supported source Cloudera Manager version	Lowest supported source Cloudera Runtime version	Cloud provider	Supported services on Replication Manager	Services that require alternate replication methods
Cloudera Private Cloud Base	7.1.1	7.1.1	Cloudera Public Cloud AWS/Azure	HDFS	HBase To replicate HBase data, see COD replication in a Nutshell and HBase data replication.
Cloudera Private Cloud Base	7.1.1	7.1.1	Data Lake in Cloudera Public Cloud AWS/Azure	Hive external tables	
Cloudera Private Cloud Base	7.9.0	7.1.1	Data Hub in Cloudera Public Cloud AWS/ Azure	Hive external tables	None
Cloudera Private Cloud Base	7.3.1	7.1.6	Data Hub in Cloudera Public CloudAWS/ Azure	HBase	None
Cloudera Private Cloud Base	7.11.3 CHF7	7.1.9 SP1	Cloudera Public Cloud GCP	HDFS, Hive external tables, HBase	None

Note: Replication Manager converts Hive2 managed tables to external tables after replication.

Replicate data from Cloudera Public Cloud source clusters

Consider the following limitations while using Cloudera Public Cloud source and Cloudera Public Cloud target clusters:

- Replication across cross-cloud providers, that is from AWS to Azure and vice-versa is not supported.
- The source and target clusters must use the same account.

Source cluster	Destination cluster	Supported services on Replication Manager	Services that require alternate replication methods
Cloudera Public Cloud AWS* / Azure	CDH 5.x CDH 6.x HDP 2.x HDP 3.x	Not applicable	HBase To replicate HBase data, see COD replication in a Nutshell and HBase data replication.
Cloudera Public CloudAWS*	CDH 5.9.0 and higher Cloudera Private Cloud Base 7.1.7 SP1 and higher	HDFS	None
Cloudera Public Cloud Azure	CDH 6.1.0 and higher Cloudera Private Cloud Base 7.1.7 SP1 and higher	HDFS	None
Cloudera Public Cloud GCP 7.2.18 and higher	Cloudera Private Cloud Base 7.1.9 SP1 and higher	HDFS	None
COD version 7.2.14 and higher - Cloudera Public Cloud AWS	AWS	HBase	None
COD version 7.2.14 and higher - Cloudera Public Cloud Azure	Azure	HBase	None

Source cluster	Destination cluster	Supported services on Replication Manager	Services that require alternate replication methods
COD version 7.2.16.1 and higher - Cloudera Public Cloud GCP	GCP	HBase	None
*Replication Manager does not support S3 as a source or destination when S3 is configured to use SSE-KMS.			

Replicate data from CDH and HDP source clusters

Replication Manager replicates HDFS data (CDH source clusters and HDP source clusters), Hive external tables (CDH source clusters), and HBase data (CDH 6 source clusters) to Cloudera Public Cloud (Amazon S3 and Microsoft Azure ADLS Gen2 (ABFS)) clusters. Replication Manager from HDP clusters toCloudera Public Cloud Azure is a beta feature and is not available for general use. You can use alternate methods to replicate Hive external tables and HBase data for scenarios that are not supported by Replication Manager.

The following tables list the minimum CDH and HDP source cluster versions, minimum Cloudera Manager versions, supported cloud providers, and supported scenarios:

Table 2: Replicate data from CDH 5 source clusters

Source cluster	Lowest supported source Cloudera Runtime version	Lowest supported source Cloudera Manager version	Cloud provider	Supported services on Replication Manager	Services that require alternate replication methods
CDH 5	5.10	6.3.0	Cloud storage in Cloudera Public Cloud AWS	HDFS	HBase To replicate HBase data, see COD replication in a Nutshell, Migrating HBase data, and HBase data replication.
CDH 5	5.10	6.3.0	Data Lake in Cloudera Public Cloud AWS	• Sentry to Ranger* • Hive external tables	
CDH 5	5.10	6.3.4	Cloud storage in Cloudera Public Cloud Azure	HDFS	
CDH 5	5.10	6.3.4	Data Lake in Cloudera Public Cloud Azure	• Sentry to Ranger* • Hive external tables	
CDH 5	5.10	7.9.0	Data Hub in Cloudera Public Cloud AWS/ Azure	Hive external tables	None
*To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.					

Table 3: Replicate data from CDH 6 source clusters

Source cluster	Lowest supported source Cloudera Runtime version	Lowest supported source Cloudera Manager version	Cloud provider	Supported services on Replication Manager	Services that require alternate replication methods
CDH 6	6.1	6.3.0	Cloud storage in Cloudera Public Cloud AWS	HDFS	HBase To replicate HBase data, see COD replication in a Nutshell, Migrating HBase data, and HBase data replication.
CDH 6	6.1	6.3.0	Data Lake in Cloudera Public Cloud AWS	• Sentry to Ranger* • Hive external tables	
CDH 6	6.1	7.1.1 / 6.3.4	Cloud storage in Cloudera Public Cloud Azure	HDFS	

Source cluster	Lowest supported source Cloudera Runtime version	Lowest supported source Cloudera Manager version	Cloud provider	Supported services on Replication Manager	Services that require alternate replication methods
CDH 6	6.1	7.1.1 / 6.3.4	Data Lake in Cloudera Public Cloud Azure	• Sentry to Ranger* • Hive external tables	
CDH 6	6.1	7.9.0	Data Hub in Cloudera Public Cloud AWS/ Azure	• Sentry to Ranger* • Hive external tables	
CDH 6	6.3.3	7.3.1	Data Hub in Cloudera Public Cloud AWS/ Azure	HBase	None
*To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.					

Note: Replication from HDP clusters to Cloudera Public Cloud clusters is a technical preview feature and is not available for general use. Before you replicate data from HDP, contact your Cloudera account team.

Table 4: Replicate data from HDP 2 and HDP 3 source clusters

Lowest supported source HDP version	Cloud provider	Supported services on Replication Manager	Services that require alternate replication methods
HDP 2.6.5*	AWS	HDFS	• HBase To replicate HBase data, see COD replication in a Nutshell and HBase data replication. • Hive external tables For more information, contact Cloudera Support.
HDP 2.6.5*	Azure	HDFS	HBase To replicate HBase data, see COD replication in a Nutshell and HBase data replication.
HDP 3.1.1*	AWS Azure	HDFS	• HBase To replicate HBase data, see COD replication in a Nutshell and HBase data replication. • Hive external tables For more information, contact Cloudera Support.
*No alternate replication methods are available for HDFS, Ranger, and Atlas replication.			

Cloud credentials to use in Cloudera Replication Manager

The Cloud Credentials page shows the registered cloud credentials for Replication Manager. To replicate data to a storage cloud account, you must register the cloud credentials, so that the Replication Manager can access your cloud account. The supported cloud storage accounts are Amazon S3 and Azure Blob Filesystem (ABFS). On the Cloud Credentials page, you can add, update, or delete cloud credentials. Before you register an Amazon S3 cloud account, ensure the cloud bucket requirements are met. Before you add Azure Cloud Credentials in Cloudera Manager, ensure the Blob container requirements are met.

You can perform the following tasks on the Cloud Credentials page to manage cloud credentials:

Add cloud credentials

You can add cloud credentials for your S3 or ABFS account. For information about adding cloud credentials, see Working with Cloud Credentials.

Note: Unregistered credentials can impact the replication process. Credentials associated with a cluster node that do not have updated credentials are called unregistered credentials. For example, if a node is down when the credentials are changed on a bucket or when the node is brought up that has the old credentials.

Update cloud credentials

You can update the cloud credentials based on various factors. When the bucket configuration such as secret or access keys, bucket name or endpoint, and encryption type is changed, it can affect the Replication Manager replication policy run and might require an update to the Replication Manager cloud credentials.

Credential changes are picked up by the next run of the policy. When you change the credentials, the in-progress policy runs might fail but the succeeding runs pick up the changes.

To update a cloud credential, click Actions Update option.

Delete cloud credentials

You can delete unwanted credentials from the Replication Manager. When you delete cloud credentials, the replication policies that use the deleted cloud credentials might fail. To avoid failures, delete the Replication Manager cloud policies associated with the deleted credentials and recreate the policies with the new credentials. You can view a list of policies associated with specific credentials on the **Cloud Credentials** page.

To delete a cloud credential, click Actions Delete option.

Registering Amazon S3 cloud account in Replication Manager

You must have valid Amazon S3 credentials to register the cloud account with Cloudera Replication Manager.

Before you begin

Consider the following requirements before you register an Amazon S3 cloud account in Replication Manager:

- You need a cloud bucket with user credentials that you can enter in Replication Manager, so Replication Manager can access the bucket.
- The bucket has to have enough space for the replicated data, and write permissions to copy the data.
- The bucket needs to support cloud storage encryption types supported by Replication Manager (SSE-S3 & SSE-KMS).

About this task

When you add cloud credentials for your Amazon S3 account, you can choose one of the following authentication methods:

- Access secret key. To use this authentication type, you require an AWS Access Key and an AWS Secret key that you obtain from Amazon. Cloudera Manager stores these values securely and does not store them in world-readable locations. The credentials are masked and encrypted in the configurations passed to processes managed by Cloudera Manager, and redacted from the logs.

- IAM role. Amazon Identity and Access Management (IAM) can be used to create users, groups, and roles for use with Amazon Web Services, such as EC2 and Amazon S3. IAM role-based access provides the same level of access to all clients that use the role.

  Important: IAM role conditions

  You can choose the IAM role authentication type only when the following conditions are met:

  - The source cluster is hosted on an AWS EC2 infrastructure.
  - The source cluster Cloudera Manager and all the nodes in the cluster are running on an EC2 instance.
  - The source cluster Cloudera Manager has the same IAM role.

  For information about configuring AWS credentials, see Introduction to role based provisioning credential in AWS.

Procedure

1. Go to the Replication Manager Cloud Credentials page, and click Add.
2. In the **Add Cloud Credential** window, perform the following steps:
   a) Select the Cluster.
   b) Select S3 as the Cloud Storage Type.
   c) Name - Provide a unique cloud credential name.
   d) Authentication Type - Select one of the following authentication types:
      - Select the authentication type as Access Secret Key from the drop-down.
        - Access Key - Enter the valid access key.
        - Secret Key - Enter the valid secret key.
      - Select IAM Role if the *IAM role conditions* mentioned in the **About this task** section are met, and click Save

3. Click Validate.

Note: Using the validation feature is recommended to ensure that the Amazon S3 bucket keys are valid. If the keys are not valid, the Replication Manager policy cannot copy data to the target Amazon S3 bucket.

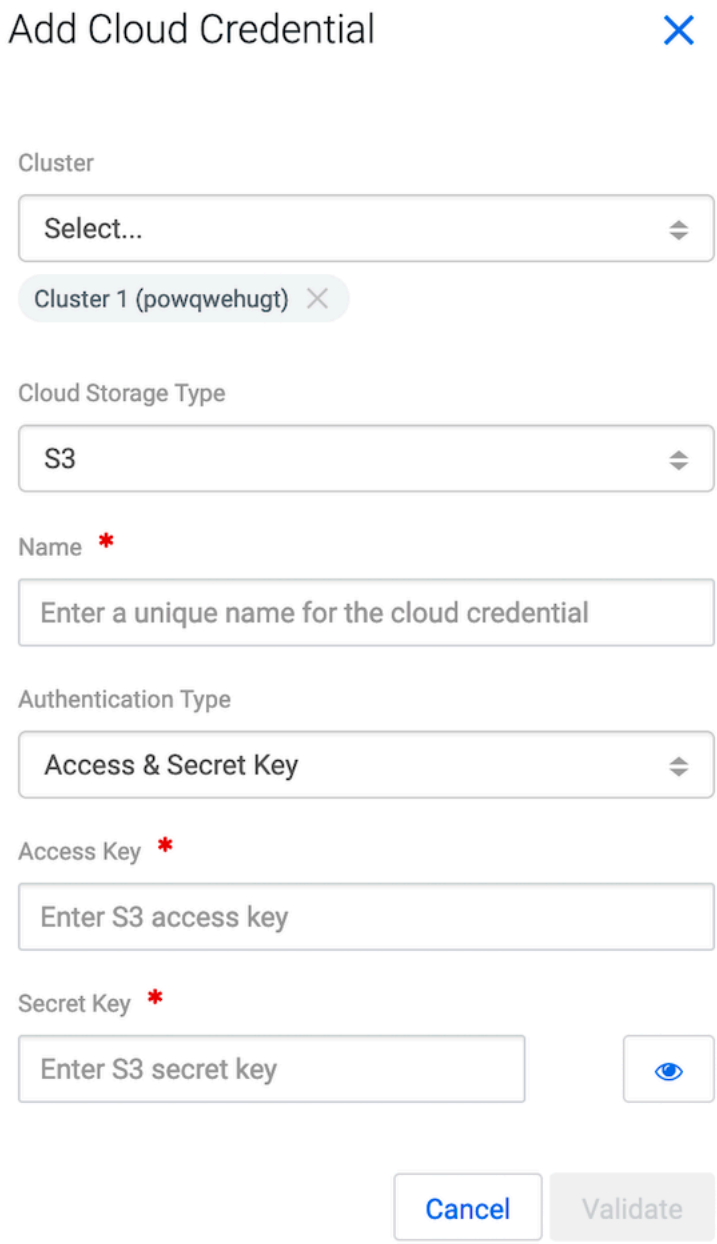

What to do next

Verify whether your credentials are listed on the Cloud Credentials page.

Register Azure cloud credentials in Replication Manager

You require an ADLS Gen2 storage account that has a cloud Blob container with user credentials to use in the Replication Manager service. The container must have enough space for the replicated data, and write permissions to copy the data.

Currently, registering Azure cloud credentials using the Replication Manager UI does not automatically create the same authorisation rules on the source Cloudera Manager cluster. When you plan to submit the replication policies

with Azure as your cloud storage, Cloudera recommends that you update the cloud credentials in the source Cloudera Manager.

Updating Azure Cloud Credentials in Cloudera Manager

Before you register an ABFS cloud account in Replication Manager, Cloudera recommends that you update the cloud credentials in the source cluster Cloudera Manager. This is because registering Azure cloud credentials using the Replication Manager UI does not automatically create the same authorisation rules on the source Cloudera Manager cluster.

Procedure

1. Click Add AD Service Principal on the Cloudera Manager Administration Azure Credentials page for the source cluster instance.

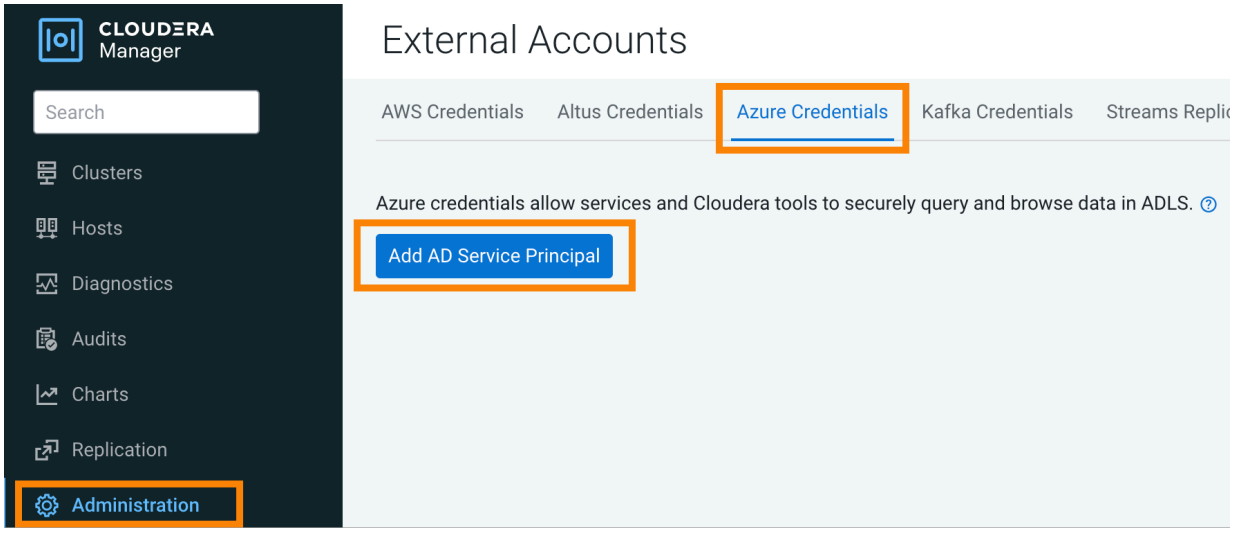

2. Enter the following details on the **Add AD Service Principal** modal window:
   a) Name of the credential.
   b) Client ID for the credential.
   c) Client Secret Key for the credential.
   d) Tenant ID for the credential.

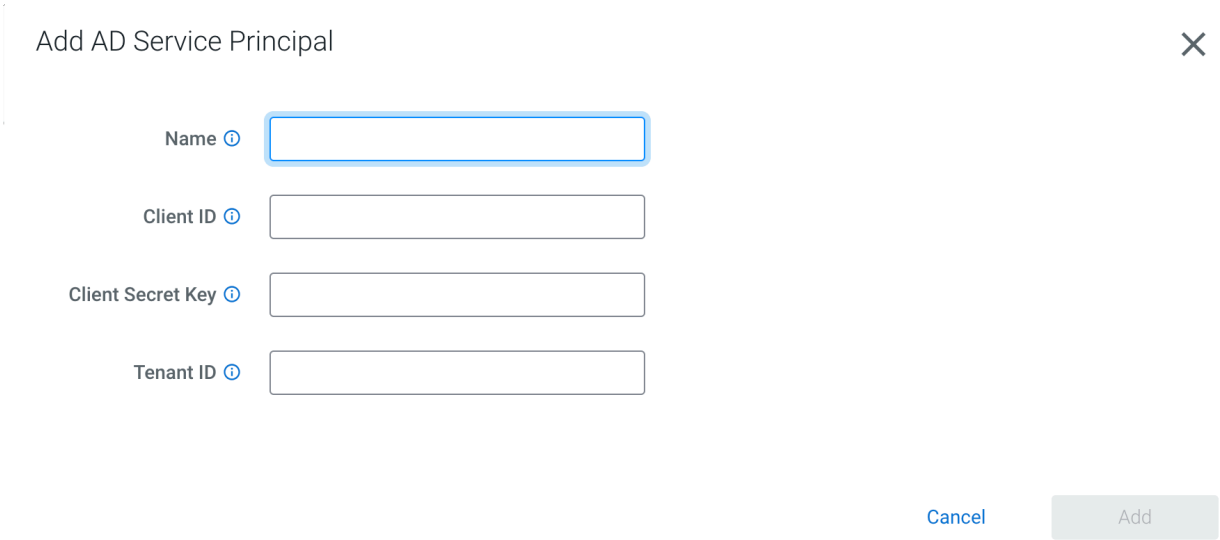

3. Click Add.
   The Azure Cloud credentials are successfully registered with the Cloudera Manager instance.

What to do next
Register the ABFS cloud account in Cloudera Replication Manager.

Registering ABFS cloud account in Replication Manager
You must have valid ABFS credentials in Cloudera Replication Manager to register the cloud account.

Before you begin
Consider the following requirements before you register an ABFS cloud account in Replication Manager:

- You require an ADLS Gen2 storage account that has a cloud Blob container with user credentials to use in the Replication Manager service.
- The container must have enough space for the replicated data, and write permissions to copy the data.
- The container must support cloud storage encryption types supported by Replication Manager (SSE-S3 & SSE-KMS).

Procedure

1. Click Cloudera Replication Manager Cloud Credentials Add .
2. Perform the following steps on the Add Cloud Credential modal window:
   a) Select the Cluster.
   b) Select ABFS as the Cloud Storage Type.
   c) Enter the cloud credential Name.
   d) Enter your ABFS Storage Client Id.
   e) Enter your ABFS Storage Client Secret Key.
   f) Enter your ABFS Storage Tenant Id.
3. Click Validate.

After you add the ABFS cloud credentials and you create a replication policy with ABFS as your selected cloud storage for your target cluster, the following error message might appear:

To resolve this issue, update the ABFS cloud credential values in the source Cloudera Manager instance. For more information, see *Updating Azure Cloud Credentials in Cloudera Manager*.

What to do next
Verify whether the credentials appear on the **Cloud Credentials** page.

Registering GCP credentials to use in Replication Manager
You can register Google Cloud Storage (GCS) credentials to use in Replication Manager after you provide access to GCP for the source cluster in the source Cloudera Manager.

Before you begin

Ensure that the following details are available before you register the GCS credentials to use in Replication Manager.

- A valid Google Cloud Platform service account. For more information, see Configuring access to Google Cloud Storage.
- Enough space for the replicated data and the required write permissions to copy the data in the storage of the destination cluster.

Procedure

1. Click Cloud Credentials Add in Cloudera Replication Manager.
2. Select the required source Cluster.
3. Choose GCS as the Cloud Storage Type.
4. Enter a unique Name for the cloud credential.

5. Select one of the following Input Type, and then complete the required steps:
   a) Select Manual if you have the following details about the GCP service account:
      - Enter your GCS service account Client Email address.
      - Enter your GCS service account Private Key.
      - Enter your GCS service account Private Key ID.

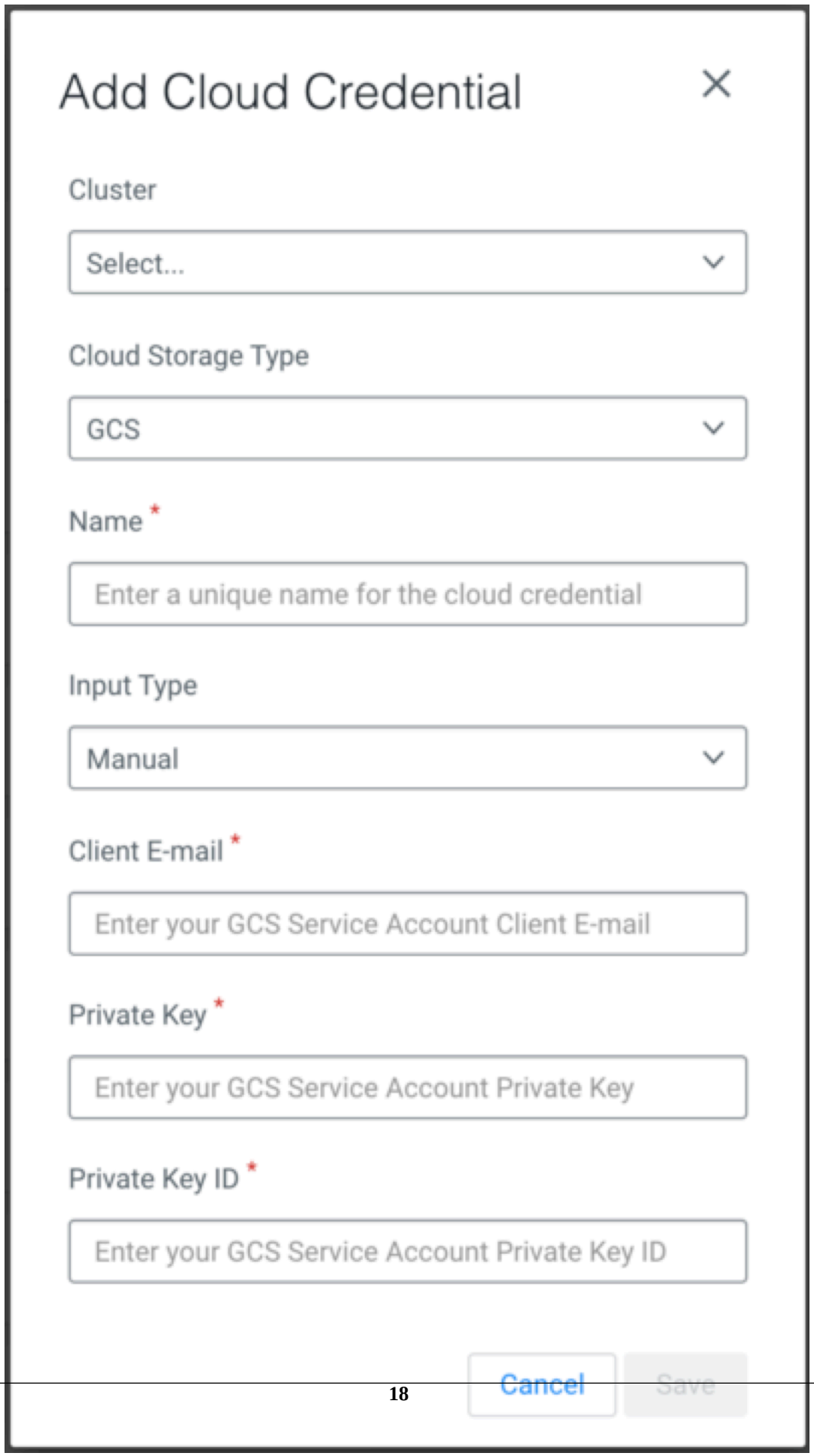
Add Cloud Credential
Cluster
Select...
Cloud Storage Type
GCS
Name *
Enter a unique name for the cloud credential
Input Type
Manual
Client E-mail *
Enter your GCS Service Account Client E-mail
Private Key *
Enter your GCS Service Account Private Key
Private Key ID *
Enter your GCS Service Account Private Key ID
Cancel
Save

b) Select File if you chose to save the service account private key in a file in JSON format. Ensure that you have the required permissions to access and use the file.

Upload File for Replication Manager to use the credentials in it to access data.

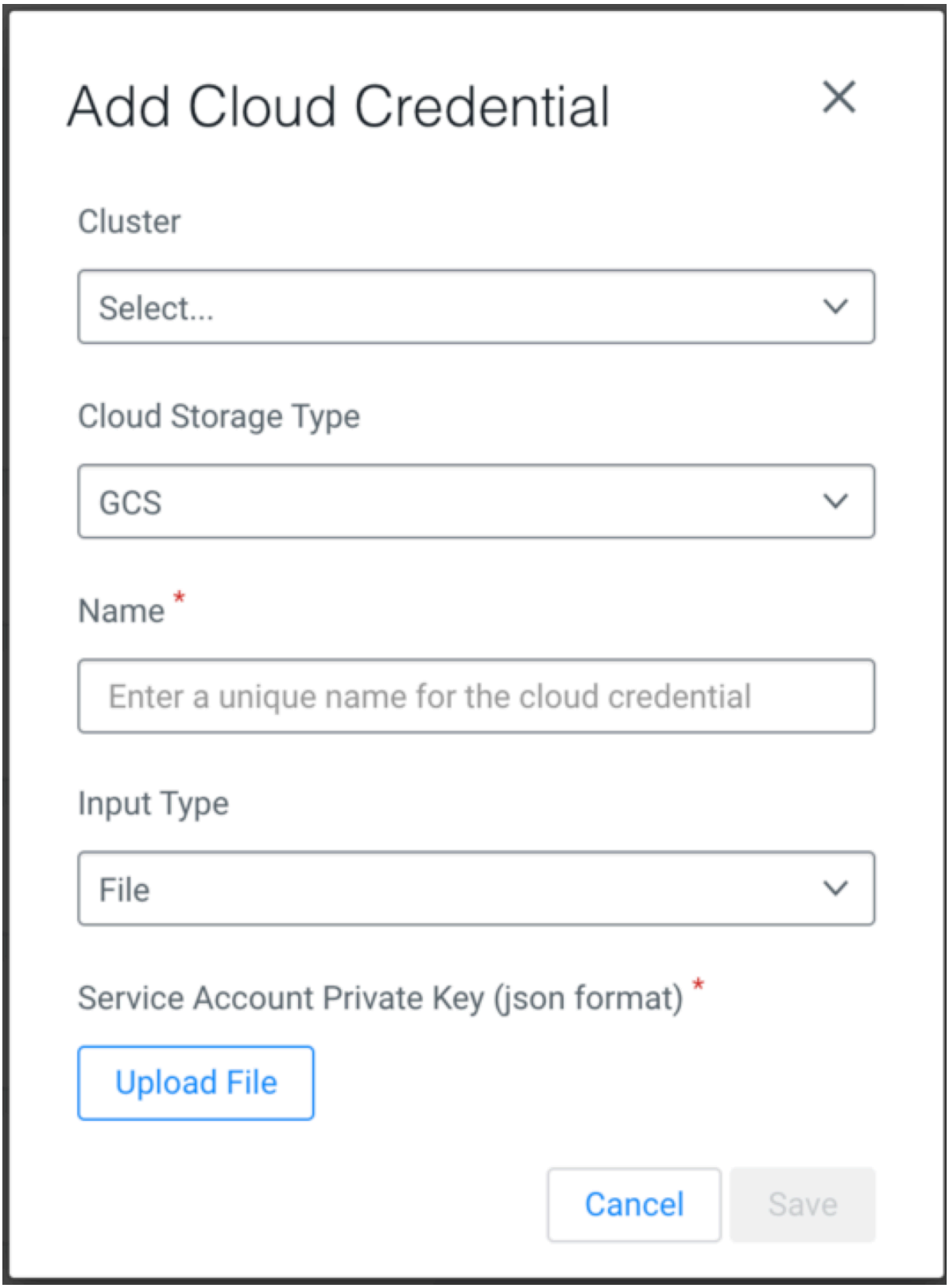

6. Click Save.

Results

The GCP cloud credentials appear on the **Cloud Credentials** page. You can use these credentials when you create the replication policies. The credentials allow Replication Manager to access the cloud data in the source cluster.

Add IDBroker to use temporary AWS session credentials

You can use temporary AWS session credentials through IDBroker to provide just-in-time, minimum required access to replicate data using replication policies. You must complete a few prerequisites before you configure IDBroker to use in replication policies. You can then add the credentials in Cloudera Replication Manager. Alternatively, you can add an external account for the IDBroker topology in Cloudera Manager.

Before you use temporary AWS session credentials in a replication policy, you must:

1. have an AWS account with an IAM role that has the required permissions to access the target S3 bucket and has the necessary trust relationships set up.
2. install a role instance for IDBroker
3. configure non-HA IDBroker on the Cloudera Private Cloud Base cluster.
4. add the cloud credential in Cloudera Replication Manager.

   Alternatively, you can add an external account for the IDBroker topology in Cloudera Manager.

Adding a role instance to IDBroker in Cloudera Manager

To use IDBroker to access the cloud credentials, you must add a role instance to IDBroker, and then you configure the required properties for it in Cloudera Manager.

Before you begin

You must have an AWS user account that has:

- an IAM policy which allows you to access the S3 bucket to which you want to replicate the data.
- an IAM role, which you can assume, that has the above policy attached to it.

If you have upgraded or installed the Cloudera Private Cloud Base cluster, ensure that IDBroker is available. IDBroker is automatically configured by Cloudera Manager in Cloudera deployments, where Knox is installed. For more information about IDBroker, see Configuring access to S3.

Procedure

1. Go to the source Cloudera Manager Clusters *KNOX SERVICE* Instances page.

   Important: If you are replicating HDFS data from cloud to Cloudera Private Cloud Base cluster, add the role instance on the target Cloudera Manager.

2. Click Actions Add Role Instances .
3. Select the required **Knox IDBroker** host on the **Add Role Instance to Knox** modal window to install it to the required host, and then click Continue.
4. Specify an IDBroker Master Secret, and click Finish.
   The **Instances** tab shows the added role instance.

What to do next

Configure the required properties for IDBroker to use in replication policies.

Configuring IDBroker to use in replication policies

After you add a role instance to install IDBroker, you configure the required properties for it in Cloudera Manager for the Cloudera Private Cloud Base cluster.

Procedure

1. Go to the source Cloudera Manager Clusters *KNOX SERVICE* Instances page.

   Important: If you are replicating HDFS data from cloud to Cloudera Private Cloud Base cluster, add the role instance on the target Cloudera Manager.

2. Click the role instance on the **Instances** tab to open the Knox IDBroker service.
3. Click the **Configuration** tab.
4. Click **Continue Editing Role Instance** if an alert appears.
5. Add the Kerberos username and ARN in the *[***USERNAME***]=[***ARN***]* format in the Knox IDBroker AWS User Mapping property. This maps the Kerberos user to the AWS role.

   If you have multiple usernames, you can map it as shown in the following sample Knox IDBroker AWS User Mapping value. The sample shows Kerberos users *repl*, *hdfs*, and *hbase* mapped to an AWS IAM role:

   repl=arn:aws:iam::134232123254:role/cldr-demo-role;hdfs=arn:aws:iam::134232123254:role/cldr-demo-role;hbase=arn:aws:iam::134232123254:role/cldr-demo-role

   Note: You can map multiple Kerberos users and groups to the same AWS IAM role, but one Kerberos user cannot have multiple AWS IAM roles mapped to it.

6. Add the following in the Kerberos Proxy Block property for HDFS replication policies. This allows the hdfs user to impersonate the Kerberos user during the replication policy run:
   - “hadoop.proxyuser.hdfs.groups”: “*[***KERBEROS USER***]*
   - “hadoop.proxyuser.hdfs.hosts”: “*”
7. Perform the following steps for HBase replication policies:
   a) Add the following details in the Kerberos Proxy Block property. This allows the hbase users to impersonate the Kerberos users during the replication policy run.
      - “hadoop.proxyuser.hbase.groups”: “*[***KERBEROS USER***]*
      - “hadoop.proxyuser.hbase.hosts”: “*”

   b) Add the following details in the Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml in HDFS configuration. This allows the hbase users to impersonate the Kerberos users when exporting the initial snapshot from HDFS.
      - “hadoop.proxyuser.hbase.groups”: “*[***KERBEROS USER***]*
      - “hadoop.proxyuser.hbase.hosts”: “*”
8. Configure the IDBroker Knox Token TTL property to ensure that the configured Knox session token time is greater than the time required to complete a replication policy run, and Save Changes.
9. Use the default aws-cab topology, or create a custom topology, if required, using the Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property. You can also create multiple topologies depending on your use case requirements.

   The following sample code shows a custom topology added to the Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property:

```
<property>
    <name>[***TOPOLOGY1***]</name>
    <value>
         providerConfigRef=cab-providers#IDBROKER:cloud.policy.config.prov
ider=default#IDBROKER:cloud.client.provider=AWS
    </value>
</property>
```

10. Complete the following steps to create the aws.credentials.key and aws.credentials.secret aliases in the topology.
    a) Search for the Save Alias Command Input property:
    b) Enter *[***TOPOLOGY1***]*.aws.credentials.secret=*[***SECRET***]*, and click Save Changes.
    c) Click Actions Save Alias - IDBroker .
    d) Enter *[***TOPOLOGY1***]*.aws.credentials.key=*[***ACCESS KEY***]*, and click Save Changes.
    e) Click Actions Save Alias - IDBroker .
11. Add the following credential details to use the default AWS topology in IDBroker if all the required IAM roles are assumed by a single set of long-term AWS keys. IDBroker uses these credentials to authenticate and to request session tokens from AWS Session Token Service (AWS STS). These credentials are used by IDBroker only to request session tokens and are not used during replication.
    - Knox IDBroker AWS Credentials Key
    - Knox IDBroker AWS Credentials Secret
12. Save the changes.
13. Restart Stale Services, if any.

What to do next

You can add the cloud credential in Cloudera Replication Manager. Alternatively, you can add an external account for the IDBroker topology in Cloudera Manager to use in replication policies.

Adding IDBroker credentials in Cloudera Replication Manager

After you install and configure non-HA IDBroker in Cloudera Manager, you add the cloud credentials in Cloudera Replication Manager. Alternatively, you can create an IDBroker-based external account in Cloudera Manager to use AWS temporary credentials for data replication using replication policies.

Before you begin

Ensure that you have completed the steps to install and configure IDBroker in Adding a role instance to IDBroker in Cloudera Manager on page 20.

Procedure

1. Go to the Cloudera Replication Manager Cloud Credentials page.
2. Click Add.
3. Enter the following details on the **Add Cloud Credential** modal window:

Option	Description
Cluster	Choose the source Cloudera Private Cloud Base cluster where you have configured the IDBroker mapping. **Important:** If you are replicating HDFS data from cloud to Cloudera Private Cloud Base cluster, the IDBroker must be configured on the target Cloudera Manager.
Cloud Storage Type	Choose S3.
Name	Provide a name for the IDBroker topology.
Authentication Type	Choose IDBroker Topology.
IDBroker Address	Enter the IDBroker host and port details in the https://*[***IDBROKER HOST***]*:*[***IDBROKER PORT***]*/gateway format. To identify the required IDBroker address to add in this field, go to the source Cloudera Manager Clusters *KNOX SERVICE* Instances page. The Hostname for the Knox IDBroker Role Type is the IDBroker address. For example, `https://cldrcld-1.cld.root.site:8444/gateway`.

Option	Description
Cloud Topology	Enter the topology name. Enter the topology name you added in the Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property on the source Cloudera Manager Clusters Knox service Configuration tab.

Results

The credentials appear on the **Cloud Credentials** page.

What to do next

You can now use the temporary AWS session credentials through the IDBroker credentials when you create HDFS, Hive, or HBase replication policies in Cloudera Replication Manager.

Adding and managing an IDBroker-based external account in Cloudera Manager

You can create an IDBroker-based external account in Cloudera Manager to use AWS temporary credentials for data replication using replication policies. Ensure that you have configured the required properties for it in Cloudera Manager for the Cloudera Private Cloud Base cluster. Alternatively, you can add the IDBroker credentials in Cloudera Replication Manager.

Procedure

1. Go to the source Cloudera Manager Administration Settings AWS Credentials page.

   Important: If you are replicating HDFS data from cloud to Cloudera Private Cloud Base cluster, add the role instance on the target Cloudera Manager.

2. Complete the following steps to add an IDBroker-based external account in Cloudera Manager:
   a) Click Add IDBroker Topology for Authentication.

      The **Add IDBroker Topology for Authentication** modal window appears.
   b) Provide a Name for the IDBroker topology.
   c) Enter one or more comma-separated list of IDBroker addresses. The IDBroker address includes the IDBroker host and port details in the https://*[***IDBROKER HOST***]*:*[***IDBROKER PORT***]*/gateway format.

      To identify the required IDBroker address to add in this field, go to the source Cloudera Manager Clusters *KNOX SERVICE* Instances page. The Hostname for the Knox IDBroker Role Type is the IDBroker address.

      For example, https://cldrcld-1.cld.root.site:8444/gateway.
   d) Enter the Cloud topology or IDBroker topology to use for cloud connections.

      Tip: Enter the topology name that you added in the source Cloudera Manager Clusters Knox service Configuration Knox IDBroker Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property.

3. Perform one or more of the following steps to manage an IDBroker-based external account:
   a) Go to the source Cloudera Manager Administration Settings AWS Credentials page.

      Important: If you are replicating HDFS data from cloud to Cloudera Private Cloud Base cluster, add the role instance on the target Cloudera Manager.
   b) Click Actions Edit Credential to edit the credentials for the required **IDBroker Topology**.
   c) Click Actions Remove the credentials to remove the credentials for the required **IDBroker Topology**.
   d) Click Actions Edit Connectivity to open the **Connect to Amazon Web Services** modal window to view more details.

Ports for Cloudera Replication Manager

Before you create replication policies, you must ensure that the required ports are open and available for data replication. You can verify the mandatory ports using the Replication Manager network security diagram.

HDFS replication polices

The following ports must be open and available for Replication Manager for HDFS replication policies:

Table 5: Minimum ports required for HDFS replication policies

Connectivity required	Default Port	Type	Description
Data transfer from classic cluster hosts to cloud storage	80 or 443 (TLS)	Outbound	Outgoing port. All classic cluster nodes must be able to access S3/ADLS Gen2 endpoint. **Note:** Before you create replication policies, ensure that the best practices for Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS) are complete.
Classic cluster	6000-6049 for CCMv1 443 for CCMv2	Outbound	Connecting source classic cluster to the Cloudera Management Console through Cluster Connectivity Manager (CCM). For more information, see Outbound network access for CCM, and CCM overview.

The following system architecture diagram shows the interaction between components during HDFS replication using HDFS replication policies:

Figure 1: System architecture diagram for HDFS replication in Cloudera Replication Manager

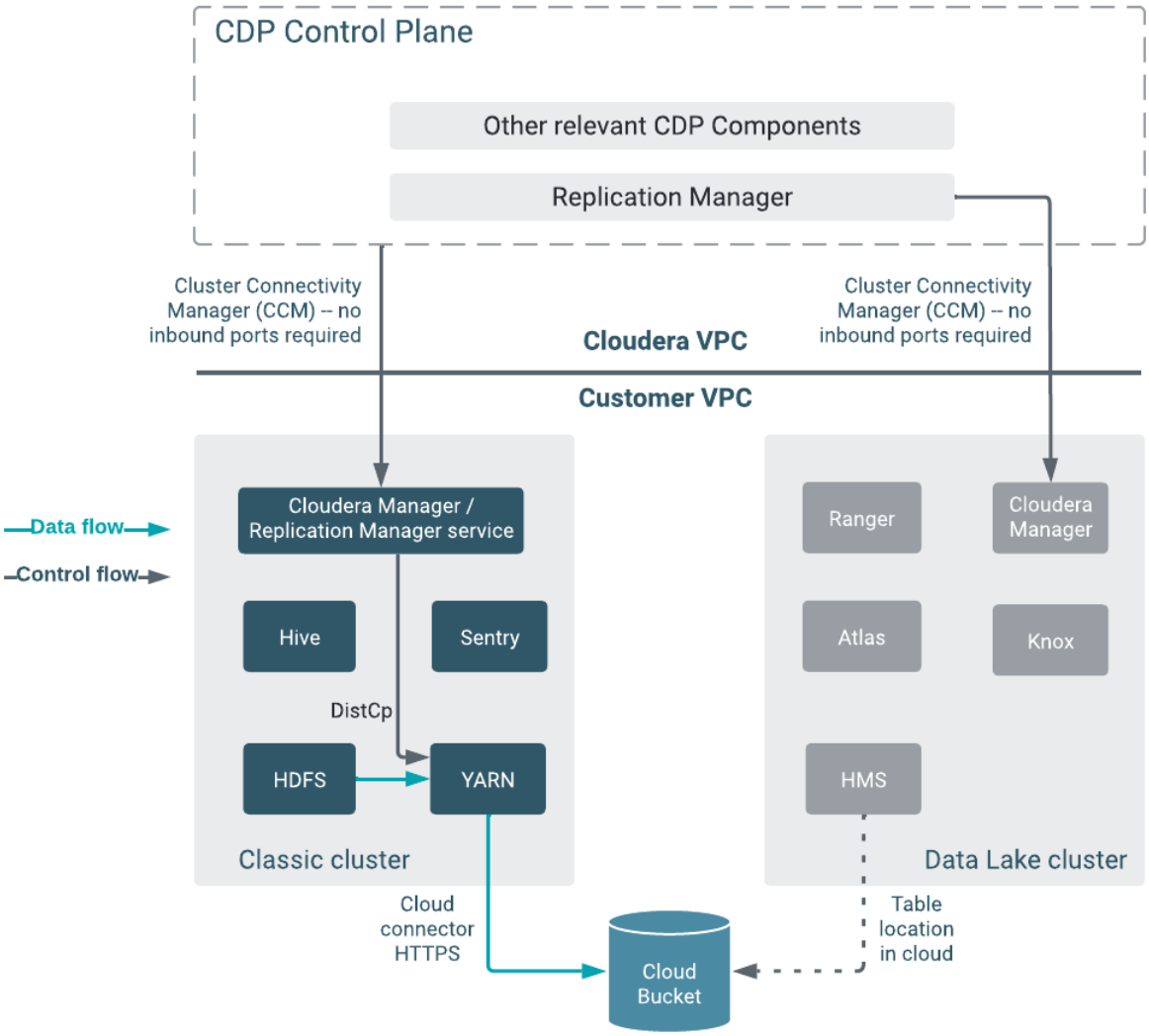

Hive replication polices

The following ports must be open and available for Replication Manager for Hive replication policies:

Table 6: Minimum ports required for Hive replication policies

Connectivity required for	Default Port	Port opened toward	Description
Data transfer from classic cluster hosts to cloud storage	80 or 443 (TLS)	Outbound towards cloud provider public hosts (For example, AWS)	Outgoing port. All classic cluster nodes must be able to access S3/ ADLS Gen2 endpoint. **Note:** Before you create replication policies, ensure that the best practices for Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS) are complete.

Connectivity required for	Default Port	Port opened toward	Description
Cloudera Manager Admin Console HTTP	7180 or 7183 (when TLS enabled)	Inbound towards destination Cloudera Public Cloud Cloudera Manager host	Incoming port. Open on the source cluster to enable the target Cloudera Manager in cloud to communicate to the on-premises Cloudera Manager.
Classic cluster	6000-6049 for CCMv1 443 for CCMv2	Outbound towards Cloudera Public Cloud Control Plane (CCM and cluster proxy)	Connecting the source classic cluster to the Cloudera Management Console through Cluster Connectivity Manager (CCM) For more information, see Outbound network access for CCM, and CCM overview.

The following system architecture diagram shows the interaction between components during Hive replication using Hive replication policies:

Figure 2: System architecture diagram for Hive replication in Cloudera Replication Manager

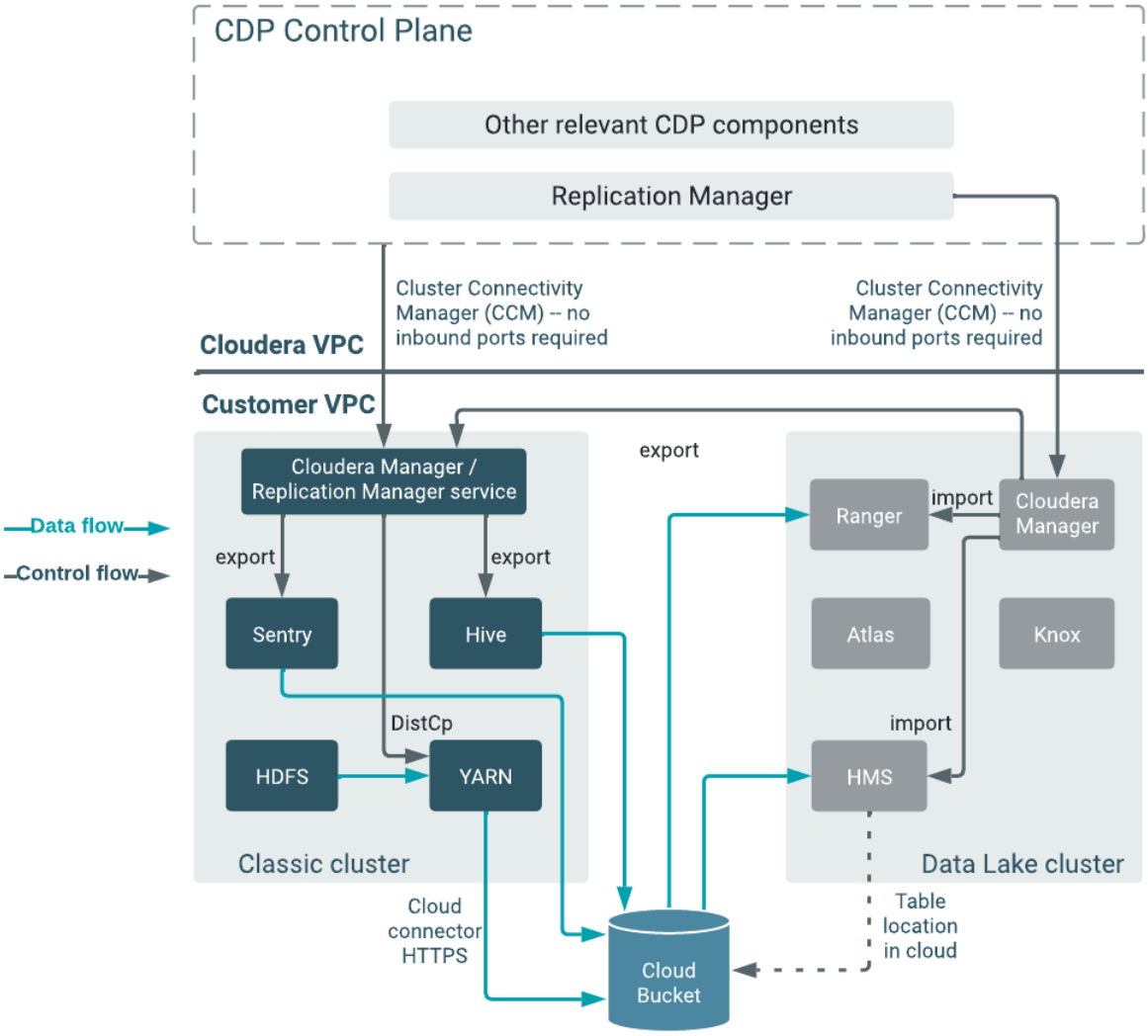

HBase replication policies

The following ports must be open and available for Replication Manager for HBase replication policies:

Table 7: Minimum ports required for HBase replication policies

Service	Ports	Ports opened toward	Description
Destination hosts of the AWS cluster or ADLS cluster (target cluster), and the Cloudera Manager server port on the source cluster	2181 and 16020	Outbound in destination Zookeeper and HBase RegionServer host(s)	Verify whether the ports 16020 for worker security group and 2181 for worker, master, and leader groups are open for connection from the source cluster to the destination cluster on AWS or Azure. This ensures that the source HBase service can communicate with Zookeeper and HBase services on the destination hosts uninterruptedly. For more information, see Ports for HBase replication.
HMaster	16000	Outbound in destination HBase Master host(s)	Open the port on the Master Nodes (HBase Master Node and any back-up HBase Master node). Before you select the Validate Replication option during the first HBase replication policy creation between two specific clusters, you must ensure that the port is open on the target cluster.
Cloudera Manager Admin Console HTTP	7180 or 7183	Inbound towards destination Cloudera Public Cloud Cloudera Manager host	Open on the source cluster to enable Data lake Cloudera Manager to communicate to the on-premises Cloudera Manager. Connects to destination SDX Data Lake Cloudera Manager.
Cluster Connectivity Manager (CCM) for CCMv1	6000-6049	For Cloudera Public Cloud Control Plane (CCM and Cluster Proxy)	Required for SSL connections to the Control Plane via CCM to communicate with Replication Manager.
Data transfer from secondary node for AWS / ADLS Gen2 for CCMv2	80 or 443	Outbound towards data transfer from secondary node for AWS / ADLS Gen2	Outgoing port. Open on all the HDFS nodes for AWS and ADLS Gen2. **Note:** Before you create replication policies, ensure that the best practices for Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS) are complete.
Data Lake cluster	8443	On destination Cloudera Public Cloud Cloudera Manager/Knox host. (applicable when Knox is available on the on-premises source cluster)	Outgoing port. Configure the port on the Data Lake cluster as the outgoing port for Cloudera Management Console to communicate with Cloudera Manager and Knox.

Figure 3: System architecture diagram for HBase replication in Cloudera Replication Manager

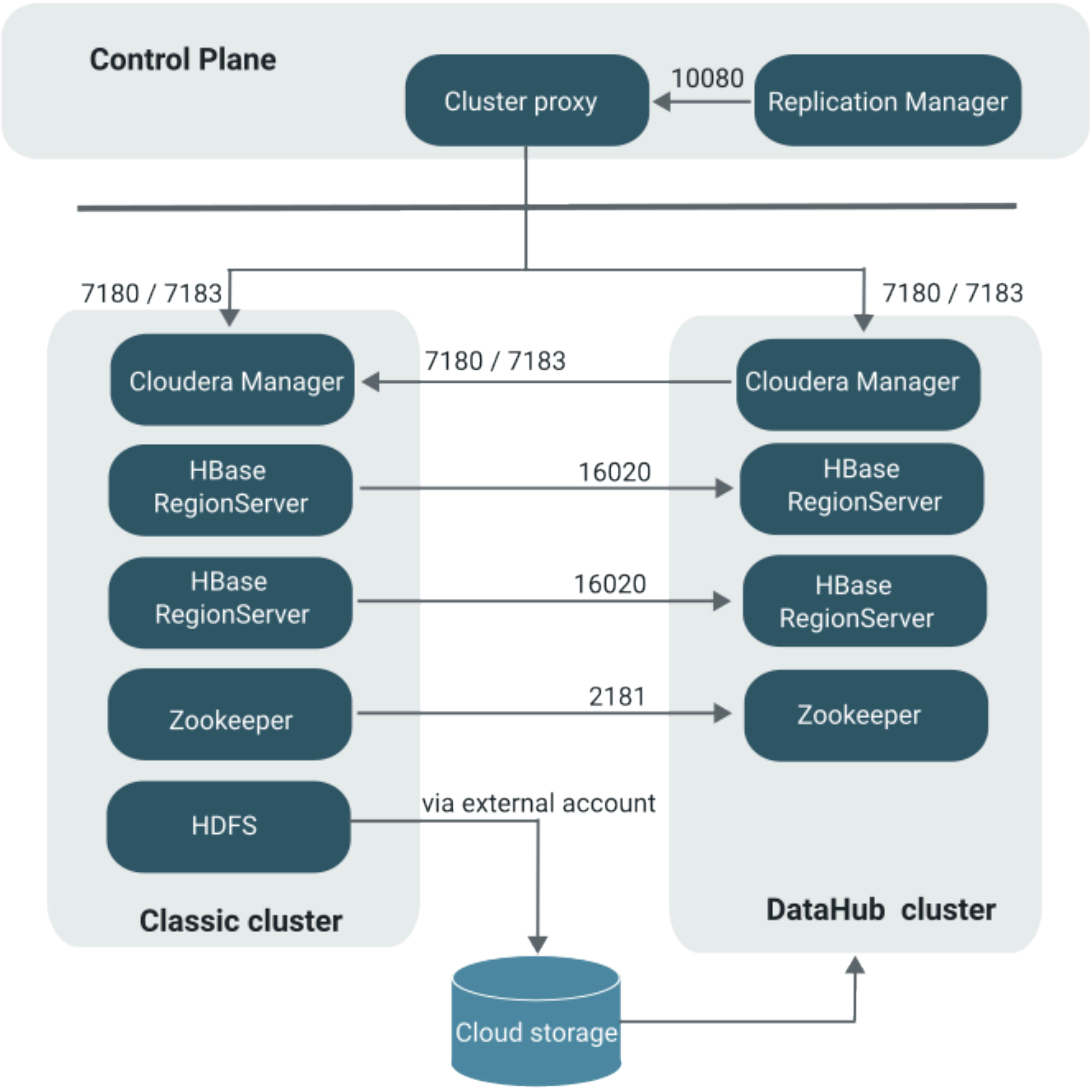

Best practices

Consider the following best practices while using Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS):

- Ensure that the on-premises cluster (port 443) can access the *https://login.microsoftonline.com* endpoint. This is because the Hadoop client in the on-premises cluster (CDH/Cloudera Private Cloud Base) connects to the endpoint to acquire the access tokens before it connects to Azure ADLS storage. For more information, see the *General Azure guidelines* row in the Azure-specific endpoints table.
- Ensure that the steps mentioned in the *General Azure guidelines* and *Azure Data Lake Storage Gen 2* rows in the Azure-specific endpoints table are complete so that the endpoint connects to the target path successfully.

Related Information

Preparing to create an HDFS replication policy
Preparing to create a Hive replication policy
Preparing to create an HBase replication policy
Support matrix for Cloudera Replication Manager

Ports required for HBase replication policies

Open the ports 2181 and 16020 on the source and destination secondary nodes to ensure that the source HBase service can reach Zookeeper and HBase services on the destination hosts. You must also ensure that the Cloudera Manager server port is open on the source cluster.

Use one of the following methods to open the required ports for HBase replication:

- Choose a security group for your environment and open the ports manually. In this method, you choose the security groups that are automatically created for the environment. By default, the security groups do not have any rules for Zookeeper and HBase ports, therefore, you must open the required ports manually after you create a Data Hub.

  After you open the ports, the required security groups are assigned to the nodes when the nodes are autoscaled. This is a one-time process that you must perform when you create a Data Hub.
- Define a security group with the required ports open, and assign it to the new Data Hub environment. In this method, you define a security group for a VPC that contains inbound rules to open the required ports which include Zookeeper and HBase ports. When you create an environment, you assign this security group to it. If required, you can assign different security groups to the gateway node and other nodes.

  This method allows you to reuse the security groups in other new Data Hubs. Security issues do not appear because the nodes in the same security group do not access each other by default. However, if required, you can add a separate rule to impose this restriction. Sharing the same security group for inbound and outbound network access rules remains as strict as having separate security groups for each environment, but the extra rules for Zookeeper and HBase ports do not need to be added at each environment creation.

The following use cases illustrate the situations where a requirement for autoscaling nodes during HBase replication might appear:

- You replicate HBase data to another Cloudera account or region in the same cloud provider. In this use case, ensure that the VPC/VNET peering is complete before you open the ports to establish connection over private networks.
- You replicate HBase data to COD or Data Hub using a direct connection. In this use case, you ensure that public IPs and Zookeeper ports are not open to the internet.

Verify cluster requirements for replication policies

You must complete the prerequisites before you create a replication policy in Cloudera Replication Manager.

The following topics list the prerequisites that must be completed before you create a replication policy:

- HDFS replication policy prerequisites
- Hive replication policy prerequisites
- HBase replication policy prerequisites

Related Information

Adding a CDH cluster
Introduction to Replication Manager
Role-based credential on AWS
App-based credential on Azure
Management console
How to configure AWS credentials
IAM role-based authentication
Support matrix for Cloudera Replication Manager

Using HDFS replication policies

You can use the HDFS replication policies in Cloudera Replication Manager to replicate HDFS data. The HDFS replication policies can replicate HDFS data and metadata from Classic Clusters (CDH, Cloudera Private Cloud Base, and HDP) to Cloudera Public Cloud storage buckets such as S3 and ABFS, and from cloud storage to Classic Clusters (CDH or Cloudera Private Cloud Base clusters). To use an on-premises cluster (CDH or Cloudera Private Cloud Base) in the replication policy, you must register it as a classic cluster in the Management Console. To use the cloud storage for data replication, you must register the cloud credentials in Replication Manager so that the Replication Manager service can access the cloud storage. You must also verify cluster access and configure minimum ports for replication before you create HDFS replication policies.

Important: Before you create replication policies, see *Support matrix for Replication Manager on Cloudera Public Cloud* to verify whether your clusters are supported by Replication Manager.

You can also use CDP CLI commands to create HDFS replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see *CDP CLI for Replication Manager*.

Related Information

HDFS replication policy

Support matrix for Cloudera Replication Manager

CDP CLI for Replication Manager

HDFS replication between on-premises and cloud

Preparing to create an HDFS replication policy

Before you create the HDFS replication policies in Cloudera Replication Manager to replicate HDFS data, register the on-premises cluster (CDH or Cloudera Private Cloud Base) as a classic cluster in Cloudera Management Console, register cloud account credentials in the Replication Manager service, verify cluster access, and configure minimum ports for replication.

Procedure

- Do the source cluster and target cluster meet the requirements to create an HDFS replication policy?

  The following image shows a high-level view of the support matrix for HDFS replication policies, you must consult the Support matrix for Cloudera Replication Manager for the complete list of supporting clusters and scenarios:

  Figure 4: High-level replication scenarios supported by HDFS replication policies

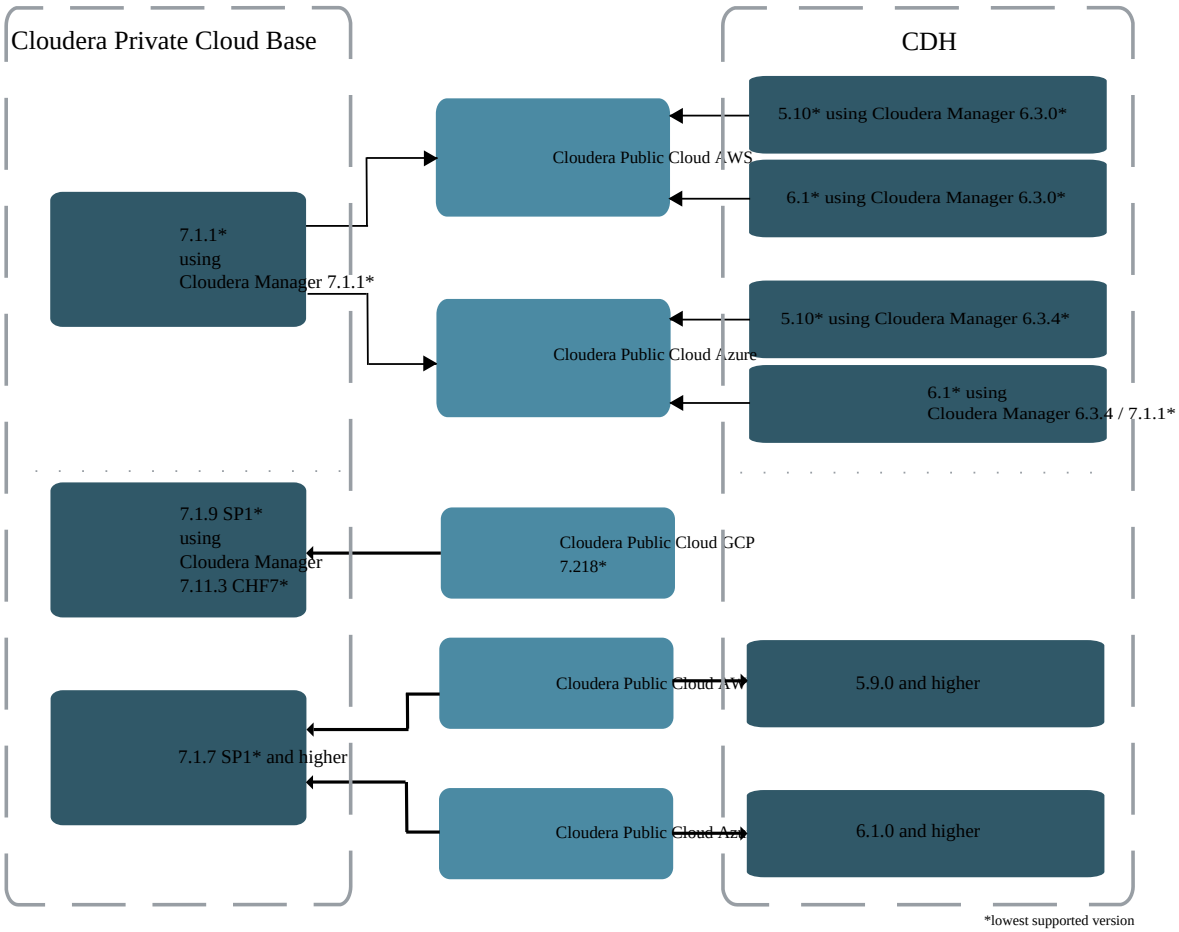

- Is the required on-premises cluster (CDH cluster or Cloudera Private Cloud Base cluster) registered as a classic cluster on the Management Console?

  CDH clusters and Cloudera Private Cloud Base clusters are managed by Cloudera Manager. To enable these on-premises clusters for Replication Manager, you must register them as Classic Clusters on the Management Console. After registration, you can use them for data migration purposes.

  Important: When you register a Cloudera Private Cloud Base cluster as a classic cluster, ensure that you use only the Cloudera Manager IP Address and Cloudera Manager Port options and not the Register KNOX endpoint (Optional) option.

  For information about registering an on-premises cluster as a classic cluster, see Adding a CDH cluster and Adding a Cloudera Private Cloud Base cluster.

- Is an external account available in the Cloudera Manager instance that has access to the bucket or container that you are using in the HDFS replication policy?

  For more information, see Role-based credential on AWS, App-based credential on Azure, and Cloudera Manager documentation.

- Do you have the required cluster access to create replication policies?

  Power users, the user who onboarded the source and target clusters, and users with ClassicClusterAdmin or ClassicClusterUser resource roles can create replication policies on clusters for which they have access. For more information, see Understanding account roles and resource roles.

  Tip: Ensure that you have *Replication Administrator* or Full Administrator role on the source on-premises cluster.

- Do you have the required cluster access to view the replication policies?

  Existing HDFS replication policies are visible to users who have access to the source cluster in the replication policy. A warning appears if you do not have access to the source cluster.

  If you can view the policies, you can perform other actions on the policy including policy update and policy delete operations.

  Note: A Replication Manager user can browse, within Replication Manager, the folder structure of any cluster enabled for Replication Manager. Therefore, permitted users can view folders, files, and databases in Replication Manager that they might not have access to in HDFS. The users cannot view the content of files on the source or destination clusters, nor can they modify or delete folders or files that are viewable from Replication Manager.

- Is the required cloud credential that you want to use in the replication policy registered with the Replication Manager service?

  For more information, see Working with cloud credentials.

- Are the following ports open and available for Replication Manager?

Table 8: Minimum ports required for HDFS replication policies

Connectivity required	Default Port	Type	Description
Data transfer from classic cluster hosts to cloud storage	80 or 443 (TLS)	Outbound	Outgoing port. All classic cluster nodes must be able to access S3/ ADLS Gen2 endpoint.
Classic cluster	6000-6049 for CCMv1 443 for CCMv2	Outbound	Connecting source classic cluster to the Cloudera Management Console through Cluster Connectivity Manager (CCM). For more information, see Outbound network access for CCM, and CCM overview.

Consider the following best practices while using Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS):

- Ensure that the on-premises cluster (port 443) can access the *https://login.microsoftonline.com* endpoint. This is because the Hadoop client in the on-premises cluster (CDH/Cloudera Private Cloud Base) connects to the

endpoint to acquire the access tokens before it connects to Azure ADLS storage. For more information, see the *General Azure guidelines* row in the Azure-specific endpoints table.

- Ensure that the steps mentioned in the *General Azure guidelines* and *Azure Data Lake Storage Gen 2* rows in the Azure-specific endpoints table are complete so that the endpoint connects to the target path successfully.

The following system architecture diagram shows the interaction between components during HDFS replication using HDFS replication policies:

Figure 5: System architecture diagram for HDFS replication in Cloudera Replication Manager

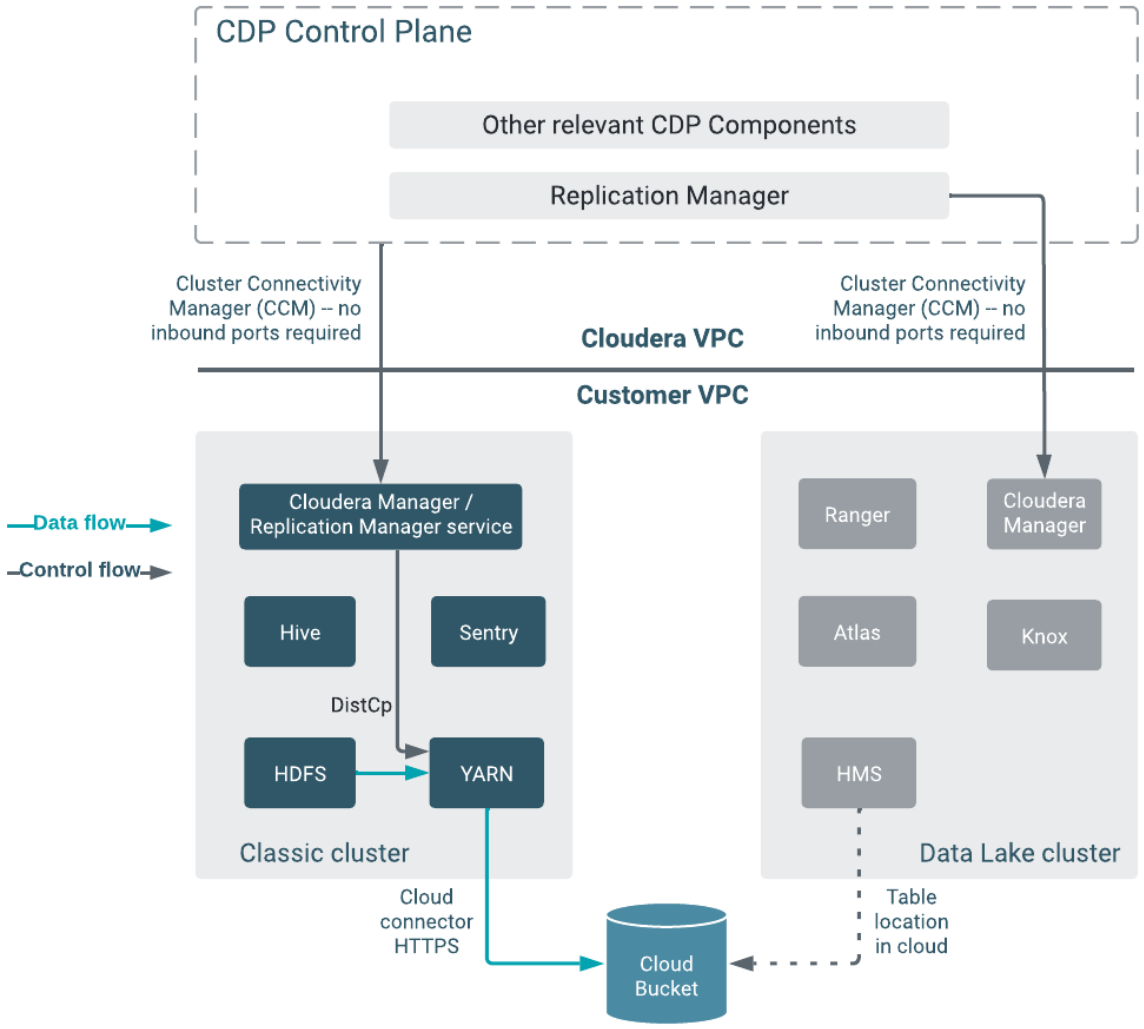

What to do next

After the clusters and cloud storage requirements are met, you can create an HDFS replication policy.

Related Information

Support matrix for Cloudera Replication Manager
Adding a CDH cluster
Adding a Cloudera Private Cloud Base cluster
Accessing Data Lake services
Role-based credential on AWS
App-based credential on Azure
Understanding account roles and resource roles

Working with cloud credentials
Ports for Cloudera Replication Manager

Creating HDFS replication policy

After you register the Classic Clusters in Management Console and register the cloud credentials in Cloudera Replication Manager, you can create an HDFS replication policy to replicate HDFS data. An HDFS replication policy can replicate HDFS data from a classic cluster (CDH, Cloudera Private Cloud Base, and HDP clusters) to cloud storage, and from cloud storage to Classic Clusters (CDH or Cloudera Private Cloud Base clusters). You can replicate data on-premises to cloud storage account with a single cluster.

Before you begin

Ensure that the HDFS replication policy prerequisites are complete.

Procedure

1. On the Cloudera Management Console Replication Manager Replication Policies page, click Add Policy.
   The **Create Replication Policy** wizard appears.
2. On the General page, choose or enter the following information:

Option	Description
HDFS	Creates an HDFS replication policy.
Policy Name	Enter a unique name for the replication policy.
Description	Optional. Enter a brief description about the replication policy.

The following image shows a sample General page in the **Create Replication Policy** wizard:

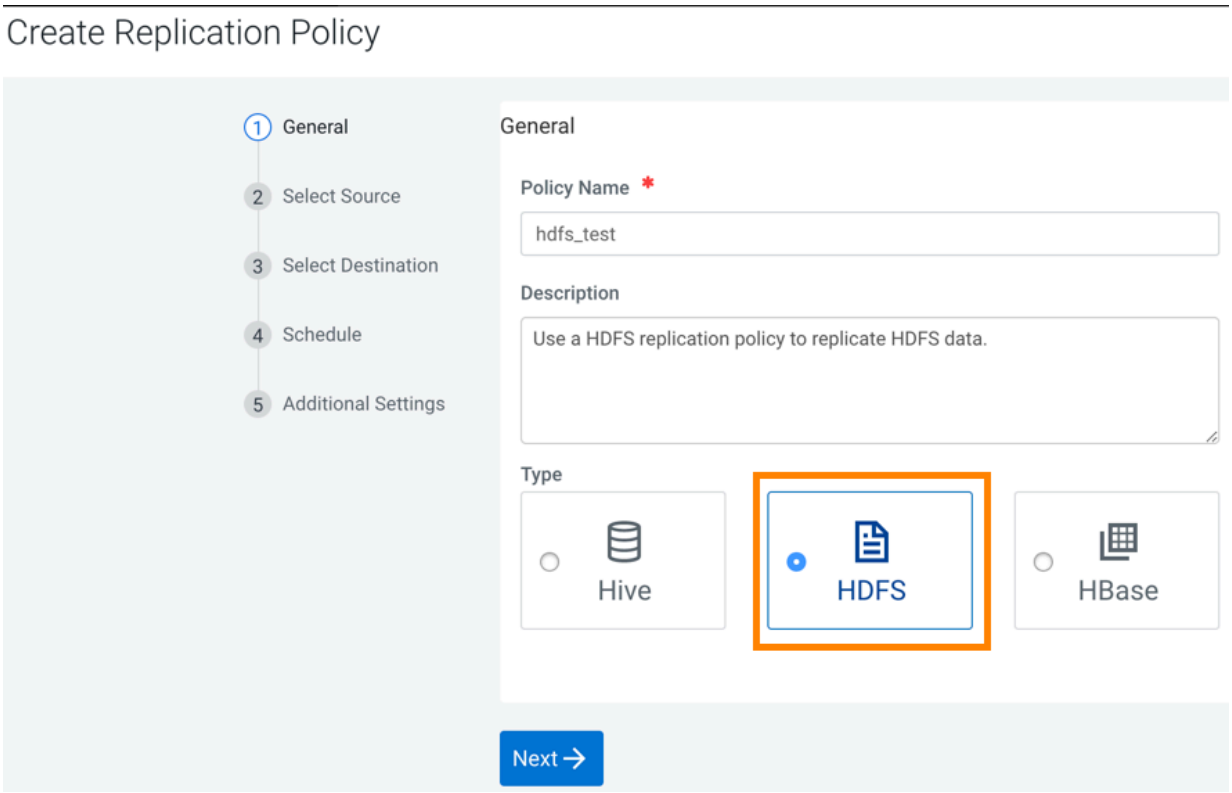

3. Click Next.

4. On the Select Source page, the options change depending on whether you choose a classic cluster or cloud storage as the source for data to be replicated.
   a) If you are replicating from a classic cluster, choose or enter the following information:

Option	Description
Type	Choose CLUSTER to select an on-premises cluster as the source cluster.
Source Cluster	Select a classic cluster.
Source path	Choose one of the following methods to determine the directory where source data resides on the source cluster: • Enter the complete directory path. • Click File Browser to view and navigate the existing directory list on the selected cluster. Select the required directory that you want to replicate.
Run As Username (on source)	Optional. The replication policy uses the Default username to replicate HDFS data. If you are using a kerberized cluster, enter the required username. The replication policy uses this username to replicate the data in the kerberized cluster.

   b) If the source of your replication is cloud storage, choose or enter the following information:

Option	Description
Type	Choose S3 or ABFS to select your cloud provider.
Cloud Credential on Source	Choose the required cloud credential. The cloud credentials that you register for Replication Manager on the Cloud Credentials page appear in this field.
Path	Based on the cloud provider, enter the path to the source data in the required format. • For an S3 bucket, provide the path of the directory in the *[***BUCKET_NAME***]/[***PATH***]* format. • For an ABFS container, provide the path of the directory in the *[***FILESYSTEM***]@[***STORAGE_ACCOUNT***]/[***LOCATION***]* format.

5. Click Next.
6. On the Select Destination page, the options change depending on whether you choose a classic cluster or cloud storage as the destination for the replicated data:
   a) If you are replicating from a classic cluster to cloud storage, choose or enter the following information:

Option	Description
Type	Choose S3 or ABFS to select your cloud provider.
Cloud Credential on Source	Choose the required cloud credential. The cloud credentials that you register for the selected source cluster on the **Cloud Credentials** page appear in this field. You can also add cloud credentials using the Add Cloud Credential link.
Path	Based on the cloud provider, enter the target path, where data is replicated to, in the required format. • For an S3 bucket, provide the path of the directory in the *[***BUCKET_NAME***]/[***PATH***]* format. • For an ABFS container, provide the path of the directory in the *[***FILESYSTEM***]@[***STORAGE_ACCOUNT***]/[***LOCATION***]* format.

b) If you are replicating from cloud storage to a classic cluster, choose or enter the following information:

Option	Description
Type	Choose Classic Cluster.
Destination Cluster	Choose a classic cluster as the target cluster.
Destination Path	Enter the directory on the target cluster to which the replication policy replicates HDFS data.
Run As Username	Optional. The replication policy uses the Default username to replicate HDFS data. Enter another username if you want the replication policy to use it to replicate data.

7. Click Validate Policy.

   Replication Manager verifies whether the details provided are correct.

8. Click Next.

9. On the **Schedule** page, choose or enter the following information:

Option	Description
Run Now	Starts to replicate the existing HDFS data after the replication policy creation is complete. Choose the frequency to replicate data periodically.
Schedule Run	Runs the replication policy to replicate data at a later time. Choose the date and time for the first run, and then choose the frequency to replicate data periodically. **Tip:** On the **Replication Policies** page, click to change the timezone.
Frequency	Choose one of the following options: • Does Not Repeat • Custom - In the Custom Recurrence dialog box, choose the time, date, and the frequency to run the policy. Replication Manager ensures that the same number of seconds elapse between the runs. For example, if you choose the Start Time as January 19, 2022 11.06 AM and Interval as 1 day, Replication Manager runs the replication policy for the first time at the specified time in the timezone the replication policy was created in, and then runs it exactly after 1 day that is, after 24 hours or 86400 seconds. **Note:** Ensure that the frequency in a schedule enables a job to finish before the next job starts. Also, ensure that the jobs based on the same policy do not overlap. If a job is not completed before another job starts, the second job does not run and the job status appears as Skipped. If a job is consistently skipped, you might need to modify the frequency of the job.

10. Click Next.

11. On the Additional Settings page, enter or choose the values as necessary:

Option	Description
YARN Queue Name	Enter the name of the YARN queue for the cluster to which the replication job is submitted if you are using Capacity Scheduler queues to limit resource consumption. The default value for this field is default.

Option	Description
Maximum Maps Slots	Set the maximum number of map tasks (simultaneous copies) per replication job. The default value is 20.
Maximum Bandwidth	Adjust this setting so that each map task is throttled to consume only the specified bandwidth. Each map task ((simultaneous copy) is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value. You can adjust this setting so that each map task is throttled to consume only the specified bandwidth so that the net bandwidth used tends towards the specified value. The default value for the bandwidth is 100MB per second for each mapper.
Path Exclusion	Enter one or more regular expressions separated by comma. Replication Manager does not copy the subdirectories or files from the source that matches one of the specified regular expressions to the target cluster.
Replication Strategy	Choose one of the following replication strategies to determine whether the file replication tasks should be distributed among the mappers statically or dynamically. • Static distributes file replication tasks among the mappers up front to achieve an uniform distribution based on the file sizes. • Dynamic distributes the file replication tasks in small sets to the mappers, and as each mapper completes its tasks, it dynamically acquires and processes the next unallocated set of tasks. The default replication strategy is Dynamic.
MapReduce Service	Choose the MapReduce or YARN service to use.
Log Path	Enter an alternate path for the logs, if required.

Option	Description
Error Handling	Select the following options as necessary: • Skip Checksum Checks - Determines whether to skip checksum checks on the copied files. If selected, checksums are not validated. Checksums are checked by default. **Note:** You must skip checksum checks to prevent replication failure due to non-matching checksums in the following cases: • Replications from an encrypted zone on the source cluster to an encrypted zone on a destination cluster. • Replications from an encryption zone on the source cluster to an unencrypted zone on the destination cluster. • Replications from an unencrypted zone on the source cluster to an encrypted zone on the destination cluster. Checksums are used for two purposes: • To skip replication of files that have already been copied. If Skip Checksum Checks is selected, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination. • To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data. • Skip Listing Checksum Checks - Whether to skip checksum check when comparing two files to determine whether they are same or not. If skipped, the file size and last modified time are used to determine if files are the same or not. Skipping the check improves performance during the mapper phase. Note that if you select the Skip Checksum Checks option, this check is also skipped. • Abort on Error - Whether to abort the job on an error. If selected, files copied up to that point remain on the destination, but no additional files are copied. Abort on Error is not selected by default. • Abort on Snapshot Diff Failures - If a snapshot diff fails during replication, the replication policy uses a complete copy to replicate data. If you select this option, the policy aborts the replication when it encounters an error instead.

Option	Description
Preserve	Choose the required options to preserve the block size, replication count, permissions (including ACLs), and extended attributes (XAttrs) as they exist on the source file system, or to use the settings as configured on the destination file system. By default source system settings are preserved. When Permission is selected, and both the source and destination clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. When Extended attributes is selected, and both the source and destination clusters support extended attributes, replication preserves them. (This option only displays when both source and destination clusters support extended attributes.) If you select one or more of the Preserve options and you are replicating to S3 or ADLS, the values all of these items are saved in metadata files on S3 or ADLS. When you replicate from S3 or ADLS to HDFS, you can select which of these options you want to preserve. **Note:** To preserve permissions to HDFS, you must be running as a superuser on the destination cluster. Use the Run As Username option to set the username.
Delete Policy	Choose the required options to determine whether the files that were deleted on the source should also be deleted from the destination directory. This policy also determines the handling of files in the destination location that are unrelated to the source. Options include: • Keep Deleted Files - Retains the destination files even when they no longer exist at the source. This is the default option. • Delete to Trash - If the HDFS trash is enabled, files are moved to the trash folder. This is not supported when replicating to S3 or ADLS. • Delete Permanently - Uses the least amount of space; use with caution.
Alerts	Choose to generate alerts for various state changes in the replication workflow. You can choose to generate an alert On Failure, On Start, On Success, or On Abort of the replication job. You can configure alerts to be delivered by email or sent as SNMP traps. If alerts are enabled for events, you can search for and view the alerts on the **Events** tab, even if you do not have email notification configured. For example, if you choose Command Result that contains the Failed filter on the Diagnostics Events page, the alerts related to the On Failure alert for all the replication policies for which you have set the alert appear. For more information, see Managing Alerts and Configuring Alert Delivery.

12. Click Create.

What to do next

You can track the replication policy job status on the **Replication Policies** page.

Manage and monitor HDFS replication policies

After you create an HDFS replication policy in Cloudera Replication Manager, you can perform and monitor various tasks related to the replication policy. You can view the job progress and replication logs. You can edit the advanced options to optimize a job run. You can suspend a job and also activate a suspended job. You can edit the replication policy as necessary.

About this task

On the **Replication Policies** page, you can perform the following actions and tasks on a replication policy and its jobs:

Procedure

- When you click Actions for an HDFS replication policy, the following actions appear:

Action	Description
Edit*	Change the replication policy options as required for non-expired policies that are in active or suspended state. Based on the schedule you choose, the replication policy replicates data. You can edit the replication policies to better align with changing requirements. For example, you might want to change the frequency of a policy depending on the data size and importance of the data being replicated. **Note:** A replication policy is associated with a cluster or a cluster pair, therefore you cannot change the clusters in the policy. Optionally, expand a replication policy on the **Replication Policies** page to edit the replication policy options which include frequency (start time cannot be modified if the policy has already started), queue name, maximum bandwidth, and maximum map slots. **Tip:** To optimize the replication policy performance, you can configure the queue name, maximum bandwidth, and maximum map slots as necessary.
Delete	Deletes the replication policy permanently.
Suspend	Suspends a running replication policy. Activate the replication policy, if required.
View Log	Download, copy, or open the log. The log shows a brief output of the stdout and stderr logs of a single step of the latest replication policy job run. You can also view the current job status in the Replication Manager Overview Issues & Updates Job Status column. If the job failed, click **Failed** to view the log details about the job.
Collect diagnostic bundle	Generates a diagnostic bundle for the replication policy. You can download the bundle as a ZIP file to your machine. Ensure that you are logged into the Cloudera Manager instances for both the source and target clusters before you download the bundle in Replication Manager.

*

To view and use the replication policies with an empty name in Replication Manager, you must understand the following implementation:

- If the API version is lower than 51, an existing replication policy with an empty name can be used and updated. However, if you edit the replication policy and provide a name for the replication policy in versions higher or equal to 51, you must ensure that the name conforms to the validation rules.
- If the API version is higher or equal to 51, it is mandatory that you provide a unique name to the replication policy to continue using it. This is because API version 51 and higher enforces the validation rules on all the replication policies.

To pass the replication policy name validation, you must ensure that the replication policy name is unique. The name can contain letters, numbers, and the _ / - characters. You must also ensure that it does not contain the characters % . ; \ nor any character that is not ASCII printable, which includes the ASCII characters less than 32 and the ASCII characters that are greater than or equal to 127.

- When you expand the policy details, the **Job History** panel appears.

  You can view the following details on the panel:

  a) Previous jobs, current job, and one future scheduled job if any.

  b) Job details which include:

Job details	Description
Started	Timestamp when the job started.
Ended	Timestamp when the job ended.
Duration	Time taken to complete the job.

Job details	Description
Progress	Current status of a running job.
Expected	Remaining number of files and bytes expected to be copied for a running job.
Copied	Number of files and bytes copied for a running job and completed job.
Failed	Number of files and bytes that failed to be copied for a completed job.
Deleted	Number of files deleted for a completed job.
Skipped	Remaining number of files and bytes skipped from copying for a running job and complete job.

c) Click Actions to:

- Abort the job.
- Re-run an aborted or failed job.
- View Log for the job. You can download, copy, or open it to track the job and to troubleshoot any issues for the job.

- When you click a job on the **Job History** panel, the following tabs appear:

Tab	Description
General	Shows the following job details: • **Started at** timestamp • **Duration** to complete the job • **HDFS Replication Report** to download the job statistics in CSV format • Job status **Message**
Command Details	Shows the steps that Replication Manager ran for the job along with the timestamp.

- You can download the following CSV reports from the General HDFS Replication Report field to track the replication jobs and to troubleshoot issues:

Report	Description
Listing	Lists all the files and directories copied during the replication job.
Status	Shows the complete status report of each file as: • an **Error** occurred and the file was not copied. • a **Deleted** file. • an up-to-date file for which the replication was **Skipped**.
Error Status	Status report of all the copied files with errors. Each file shows the status, path, and message for the copied files with errors.
Skipped Status	Status report of all skipped files. Each file lists the status, path, and message for the databases and tables that were skipped.
Deleted Status	Status report of all deleted files. Each file lists the status, path, and message for the databases and tables that were deleted.
Performance	Summary report about the performance of the running replication job which includes the last performance sample for each mapper that is working on the replication job.
Full Performance	Performance report of the job which includes the samples taken for all mappers during the replication job.

Note: The reports are generated based on the source Cloudera Manager response. If the Cloudera Manager response is interrupted or is not handled as expected, corresponding error messages appear in HTML format in the reports.

Using Hive replication policies

To create a Hive replication policy in Cloudera Replication Manager, you must configure the required Ranger policy in Ranger, register the on-premises cluster (CDH or Cloudera Private Cloud Base) as a classic cluster in Management Console, register cloud account credentials in the Replication Manager service, verify cluster access, and configure minimum ports for replication. The replication load happens on the source on-premises cluster. You can replicate data on-premises to the cloud with a single cluster if the Metastore is running on the cloud.

These policies support table-level replication and can replicate Hive external tables from on-premises clusters (CDH and Cloudera Private Cloud Base) to cloud storage such as S3 and ABFS and to Data Hubs. They also can:

- replicate data stored in Hive tables, Hive metadata, data in Hive metastore, and Impala metadata (catalog server metadata) associated with Impala tables registered in the Hive metastore, and

  Note: Hive2 managed tables are converted to external tables after replication.

- migrate Sentry permissions to Ranger.

  Note: To perform the Sentry policy replication, you must be running the Sentry service on CDH 5.12 or higher, or any CDH 6.x version.

Hive metadata replication involves multiple entities. Replication Manager supports replication of external tables in Hive. Hive supports replication of external tables to the target cluster and it retains all the properties of external tables. The data files permission and ownership are preserved so that the relevant external processes can continue to write in it even after failover.

You can also use CDP CLI commands to create Hive replication policies. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option. For more information, see *CDP CLI for Replication Manager*.

Important: Before you create Hive replication policies, you must ensure that the required Ranger policy is set in Ranger and see Support matrix for Replication Manager on Cloudera Public Cloud to verify whether your clusters are supported by Replication Manager.

The Apache Ranger access policy model consists of the following components:

- Specification of the resources that you can apply to a replication policy which includes the HDFS files and directories; Hive databases, tables, and columns; and HBase tables, column-families, and columns.
- Specification of access conditions for specific users and groups.

You must set the Ranger policy for the hdfs user on the target cluster to perform all operations on all databases and tables. The same user role is used to import Hive Metastore. The hdfs user should have access to all Hive datasets, including all operations. Otherwise, Hive import fails during the replication process.

On the target cluster, the hive user must have Ranger admin privileges. The same hive user performs the metadata import operation.

For more information about Hive replication policies to replicate data from CDH clusters to Cloudera Public Cloud, see Migrate Hive data from CDH to Cloudera Public Cloud blog.

Related Information

Hive replication policy

Support matrix for Cloudera Replication Manager

CDP CLI for Replication Manager

Preparing to create a Hive replication policy

Before you create the Hive replication policies in Cloudera Replication Manager, you must prepare the clusters and verify cluster access and cloud credentials.

Procedure

- Do the source cluster and target cluster meet the requirements to create an Hive replication policy?

  The following image shows a high-level view of the support matrix for Hive replication policies, you must consult the Support matrix for Replication Manager on Cloudera Public Cloud for the complete list of supporting clusters and scenarios:

  Figure 6: High-level replication scenarios supported by Hive replication policies

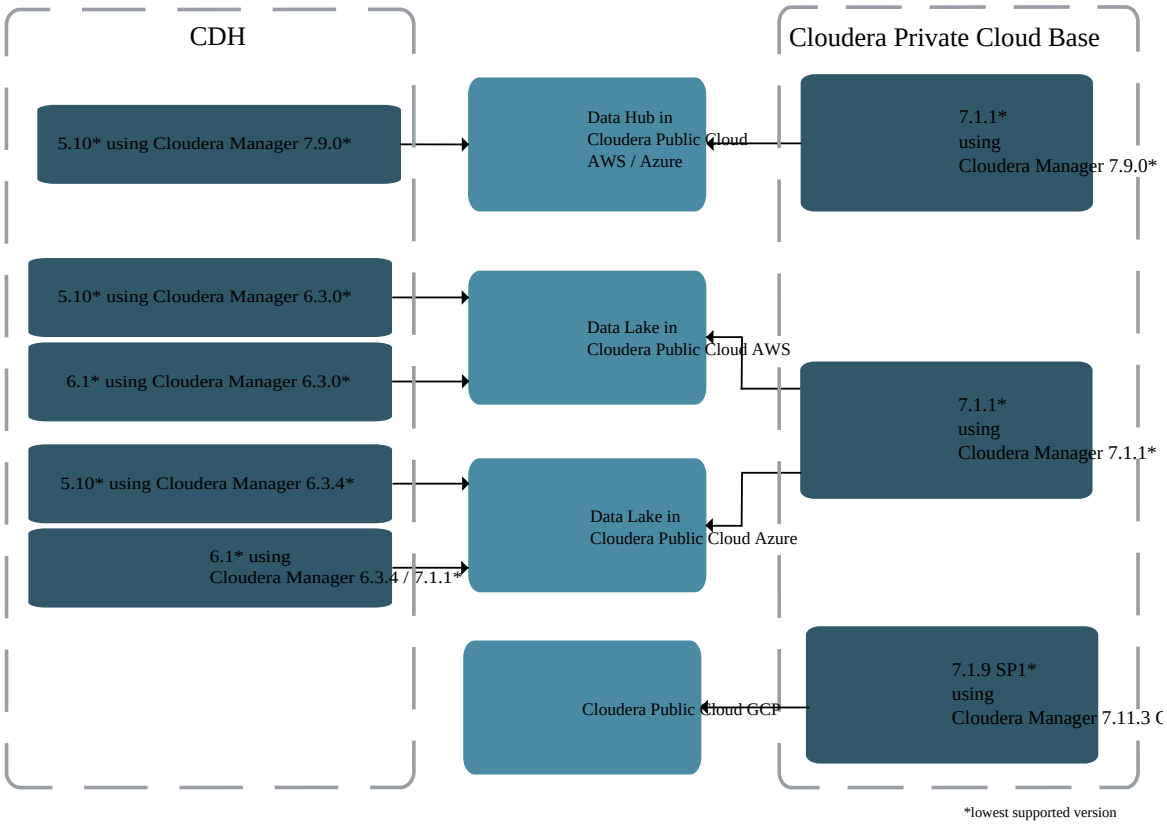

- Is the source CDH cluster or source Cloudera Private Cloud Base cluster registered as a classic cluster on the Management Console?

  CDH clusters and Cloudera Private Cloud Base clusters are managed by Cloudera Manager. To enable these on-premises clusters for Replication Manager, you must register them as Classic Clusters on the Management Console. After registration, you can use them for data migration purposes.

  Important: When you register a Cloudera Private Cloud Base cluster as a classic cluster, ensure that you use only the Cloudera Manager IP Address and Cloudera Manager Port options and not the Register KNOX endpoint (Optional) option.

  For information about registering an on-premises cluster as a classic cluster, see Adding a CDH cluster and Adding a Cloudera Private Cloud Base cluster.

- Does the target Data Hub use Cloudera Manager 7.9.0 or higher? If not, upgrade Cloudera Manager to version 7.9.0 or higher.
- Have you configured the **all-database, table, column** Ranger policy for the hdfs user on the source and target cluster to perform all the operations on all databases and tables?

  The hdfs user role is used to import Hive Metastore and must have access to all Hive datasets, including all operations. Otherwise, Hive import fails during the replication process. On the target cluster, the hive user must have Ranger admin privileges. The same hive user performs the metadata import operation.

  To provide access, navigate to the Ranger Admin UI Service Manager Hadoop_SQL Policies Access section, and provide hdfs user permission to the **all-database, table, column** policy name.

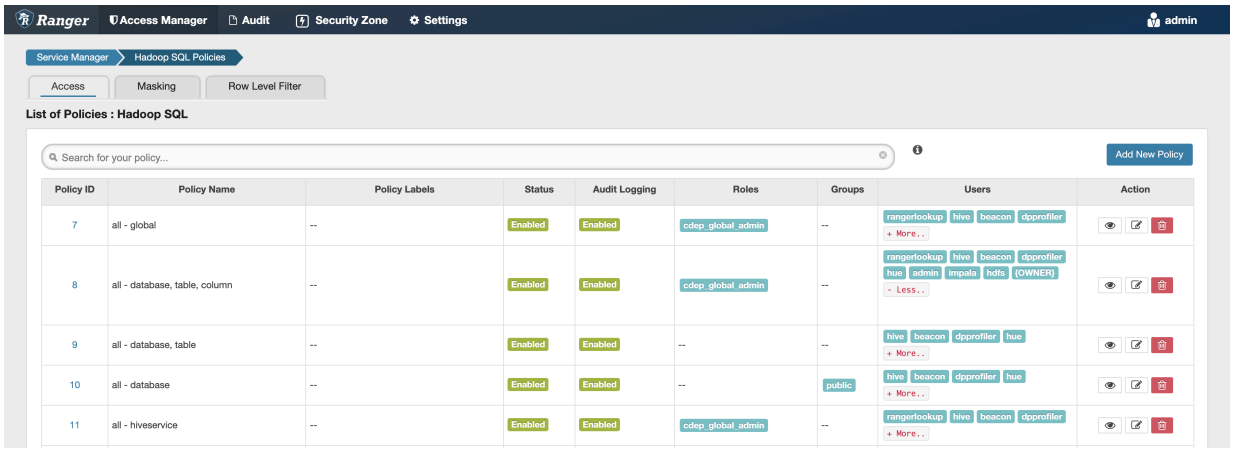

- Is an external account configured on the source CDH cluster's Cloudera Manager which allows the CDH cluster to access Cloudera cloud storage?

  Tip: The external account can be configured on the Cloudera Manager Administration page and the account has the access key / secret key pair that you can use to access Cloudera cloud storage.

- Do you have the required cluster access to create replication policies?

  Power users, the user who onboarded the source and target clusters, and users with ClassicClusterAdmin or ClassicClusterUser resource roles can create replication policies on clusters for which they have access. For more information, see Understanding account roles and resource roles.

  Tip: Ensure that you have *Replication Administrator* or Full Administrator role on the source on-premises cluster.

- Do you have the required cluster access to view the replication policies?

  Existing Hive replication policies are visible to users who have access to the source cluster in the replication policy. A warning appears if you do not have access to the source cluster.

  If you can view the policies, you can perform other actions on the policy including policy update and policy delete operations.

  Note: A Replication Manager user can browse, within Replication Manager, the folder structure of any cluster enabled for Replication Manager. Therefore, permitted users can view folders, files, and databases in Replication Manager that they might not have access to in HDFS. The users cannot view the content of files on the source or destination clusters, nor can they modify or delete folders or files that are viewable from Replication Manager.

- Is the required cloud credential that you want to use in the replication policy registered with the Replication Manager service?

  For more information, see Working with cloud credentials.

- Are the following ports open and available for Replication Manager?

Table 9: Minimum ports required for Hive replication policies

Connectivity required for	Default Port	Type	Description
Data transfer from classic cluster hosts to cloud storage	80 or 443 (TLS)	Outbound	Outgoing port. All classic cluster nodes must be able to access S3/ ADLS Gen2 endpoint.
Cloudera Manager Admin Console HTTP	7180 or 7183 (when TLS enabled)	Inbound	Incoming port. Open on the source cluster to enable the target Cloudera Manager in cloud to communicate to the on-premises Cloudera Manager.
Classic cluster	6000-6049 for CCMv1 443 for CCMv2	Outbound	Connecting the source classic cluster to the Cloudera Management Console through Cluster Connectivity Manager (CCM) For more information, see Outbound network access for CCM,and CCM overview.

Consider the following best practices while using Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS):

- Ensure that the on-premises cluster (port 443) can access the *https://login.microsoftonline.com* endpoint. This is because the Hadoop client in the on-premises cluster (CDH/Cloudera Private Cloud Base) connects to the endpoint to acquire the access tokens before it connects to Azure ADLS storage. For more information, see the *General Azure guidelines* row in the Azure-specific endpoints table.

- Ensure that the steps mentioned in the *General Azure guidelines* and *Azure Data Lake Storage Gen 2* rows in the Azure-specific endpoints table are complete so that the endpoint connects to the target path successfully.

The following system architecture diagram shows the interaction between components during Hive replication using Hive replication policies:

Figure 7: System architecture diagram for Hive replication in Cloudera Replication Manager

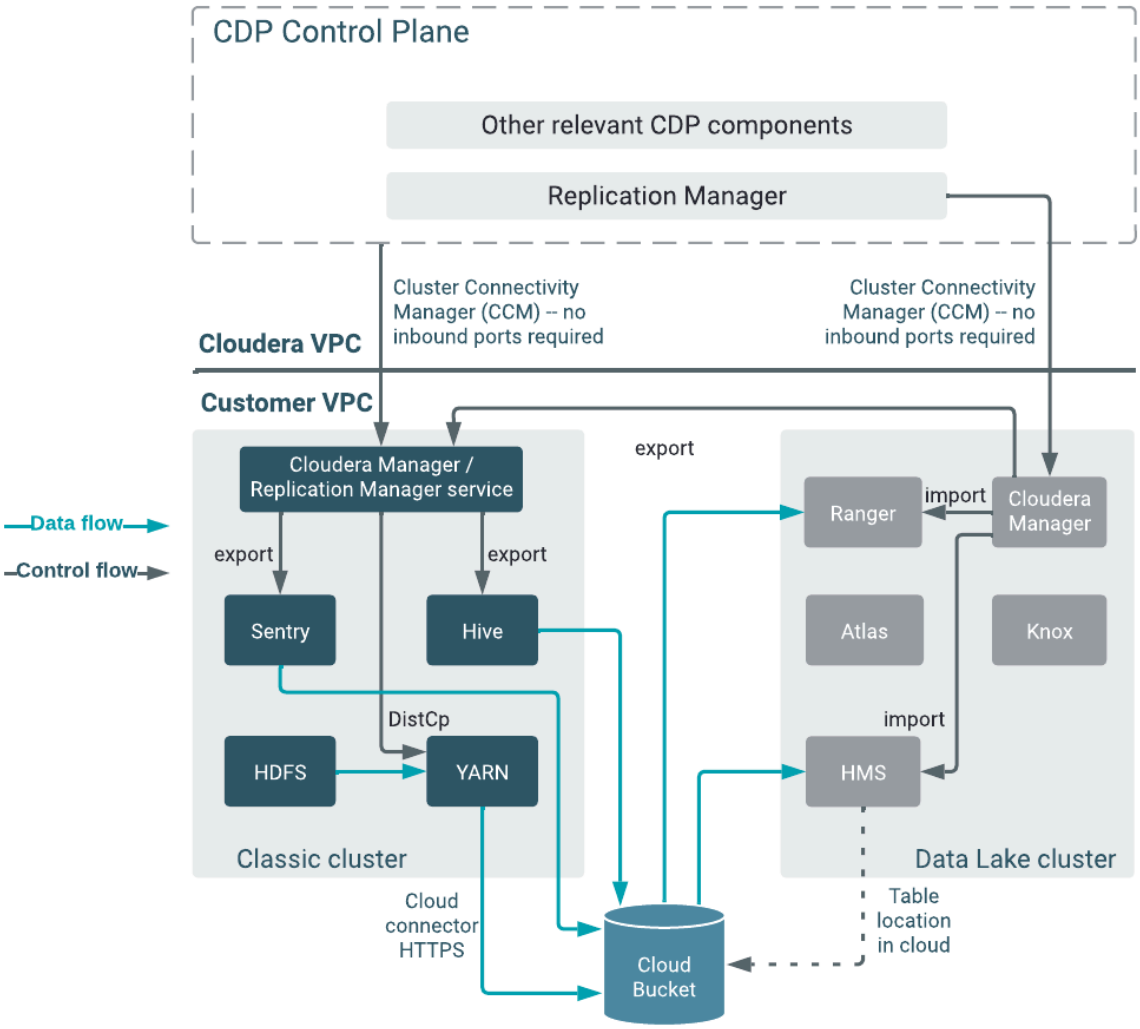

What to do next

After the clusters and cloud storage requirements are met, you can create a Hive replication policy.

Related Information

Support matrix for Cloudera Replication Manager
Adding a CDH cluster
Adding a Cloudera Private Cloud Base cluster
Accessing Data Lake services
Understanding account roles and resource roles
Working with cloud credentials
Ports for Cloudera Replication Manager

Creating Hive replication policy

Before you create a Hive replication policy in Cloudera Replication Manager, you must ensure that the prerequisites are complete and the required Ranger policy is set in Ranger.

Before you begin

Ensure that the Hive replication policy prerequisites are complete.

Procedure

1. On the Management Console Replication Manager Replication Policies page, click Add Policy.
   The **Create Replication Policy** wizard appears.
2. On the General page, choose or enter the following information:

Option	Description
Hive	Creates a Hive replication policy.
Policy Name	Enter a unique name for the replication policy.
Description	Optional. Enter a brief description about the replication policy.

The following image shows a sample General page in the **Create Replication Policy** wizard:

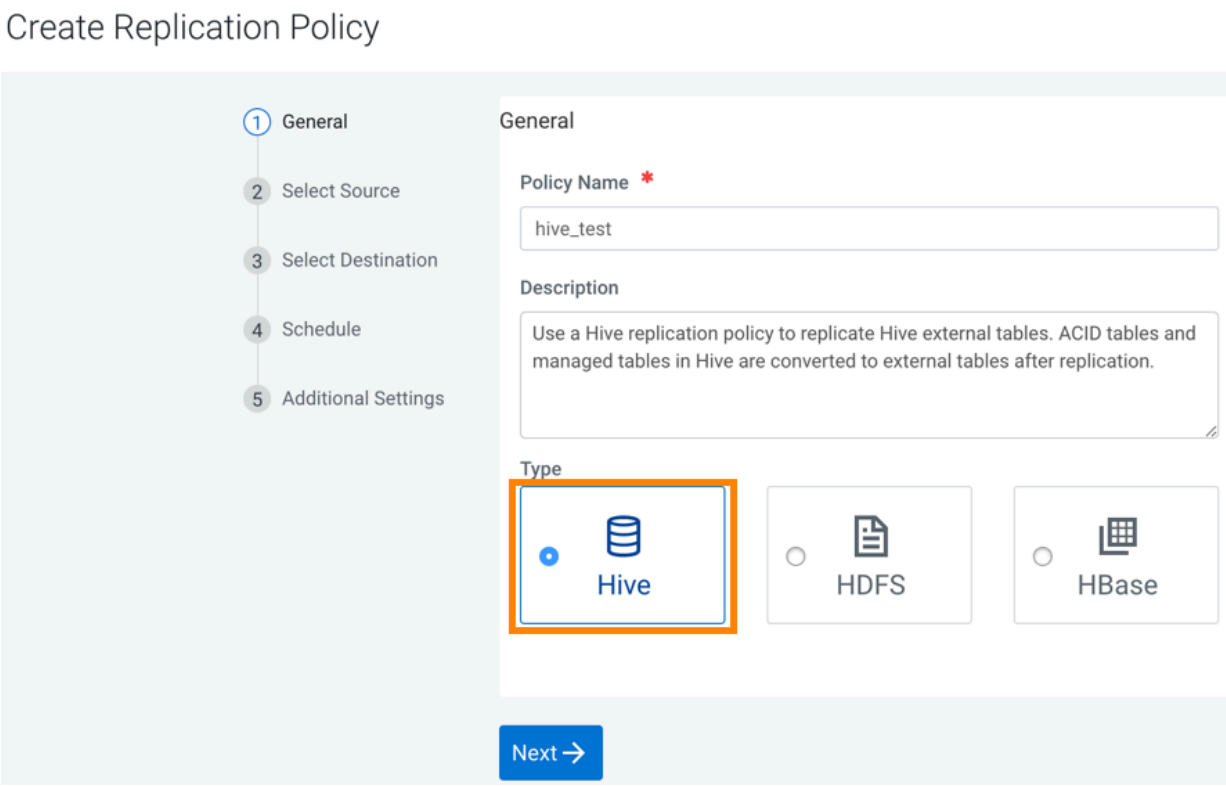

3. Click Next.
4. On the Select Source page, enter or choose the options as required:

Option	Description
Source cluster	Choose a classic cluster as a source cluster.

Option	Description
Source Databases and Tables	Enter the database name and table name. Click Add to enter more databases and tables as necessary. **Note:** Ensure that you do not add the sys and information_schema databases for replication if you want to choose the Additional Settings Invalidate Impala Metadata on Destination option for the Hive replication policy.
Run As Username (on source)	Optional. The replication policy uses the Default username to replicate data. Enter another username if you want the replication policy to use it to replicate data. Ensure that the user has the necessary permissions to replicate data. **Note:** You must provide a username for Kerberized clusters.

5. Click Next.
6. On the Select Destination page, tenter or choose the options as required:

Option	Description
Destination Data Lake or Data Hub	Choose a Data Lake or Data Hub as the destination cluster. The Managed Warehouse Path and the Hive External Table Base Directory Path for the Data Lake appears. **Note:** Administrators can edit the Hive External Table Base Directory field to add another path to override the default storage location for replicated Hive external tables. Before you add another path to override the default storage location, ensure that the following steps are complete in the Ranger UI: • Alter the ranger policy Default: Hive warehouse locations in cm_s3 service to allow the Hive service to access the updated locations of S3 bucket path. • Manually update the Ranger and Sentry permissions. **Note:** The Data Hubs, using Cloudera Manager version 7.9.0 and higher, for which you have access permissions appear automatically. If you want to use a Cloudera Data Hub that appears disabled, you must upgrade the Cloudera Manager version to 7.9.0 or higher for the Data Hub.
Cloud Credential on Source	Choose the required cloud credential. The cloud credentials that you register for the selected source cluster on the Cloud Credentials page appear in this field. You can also add cloud credentials using the Add Cloud Credential link.
Run as Username	Optional. The replication policy uses the Default username to replicate data. Enter another username if you want the replication policy to use it to replicate data.

7. Click Validate Policy.

   Replication Manager verifies whether the details provided are correct.
8. Click Next.
9. On the **Schedule** page, choose or enter the following information:

Option	Description
Run Now	Starts to replicate the existing HDFS data after the replication policy creation is complete. Choose the frequency to replicate data periodically.

Option	Description
Schedule Run	Runs the replication policy to replicate data at a later time. Choose the date and time for the first run, and then choose the frequency to replicate data periodically. **Tip:** On the **Replication Policies** page, click to change the timezone.
Frequency	Choose one of the following options: • Does Not Repeat • Custom - In the Custom Recurrence dialog box, choose the time, date, and the frequency to run the policy. Replication Manager ensures that the same number of seconds elapse between the runs. For example, if you choose the Start Time as January 19, 2022 11.06 AM and Interval as 1 day, Replication Manager runs the replication policy for the first time at the specified time in the timezone the replication policy was created in, and then runs it exactly after 1 day that is, after 24 hours or 86400 seconds. **Note:** Ensure that the frequency in a schedule enables a job to finish before the next job starts. Also, ensure that the jobs based on the same policy do not overlap. If a job is not completed before another job starts, the second job does not run and the job status appears as Skipped. If a job is consistently skipped, you might need to modify the frequency of the job.

10. Click Next.

11. On the Additional Settings page, enter or choose the values as necessary:

Option	Description
YARN Queue Name	Enter the name of the YARN queue for the cluster to which the replication job is submitted if you are using Capacity Scheduler queues to limit resource consumption. The default value for this field is default.
Maximum Maps Slots	Set the maximum number of map tasks (simultaneous copies) per replication job. The default value is 20.
Maximum Bandwidth	Adjust this setting so that each map task is throttled to consume only the specified bandwidth. Each map task ((simultaneous copy) is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value. You can adjust this setting so that each map task is throttled to consume only the specified bandwidth so that the net bandwidth used tends towards the specified value. The default value for the bandwidth is 100MB per second for each mapper.
Number of concurrent HMS connections	Enter the number of concurrent Hive Metastore connections. The connections are used to concurrently import and export metadata from Hive. Increase the number of threads to improve Replication Manager performance. By default, a new replication policy uses 4 connections. • If you set the value to 1 or more, Replication Manager uses multi-threading with the number of connections specified. • If you set the value to 0 or fewer, Replication Manager uses single threading and a single connection.

Option	Description
Sentry Permissions	Choose Include Sentry Permissions with Metadata to migrate Sentry permissions during the replication job. Otherwise, choose Exclude Sentry Permissions from Metadata to not migrate Sentry permissions during the replication job.
Skip URI Privileges	Select this option if you do not want to include URI privileges when you migrate Sentry permissions. During migration, the URI privileges are translated to point to an equivalent location in S3. If the resources have a different location in Amazon S3, do not migrate the URI privileges because the URI privileges might not be valid.
Replication Option	Choose Metadata and Data to replicate metadata and data in files and directories. Otherwise, choose Metadata only to replicate the metadata of files and directories
Directory for metadata file	Enter / or a valid folder path in the target cluster to save the metadata file. If the field is empty or if the specified folder does not exist, Replication Manager creates a new folder.
Force Overwrite	Select to overwrite data in the destination metastore if incompatible changes are detected. For example, if the destination metastore was modified, and a new partition was added to a table, this option forces deletion of that partition, overwriting the table with the version found on the source. If you do not choose the option and the Hive replication process detects incompatible changes on the source cluster, Hive replication fails. This sometimes occurs with recurring replications, where the metadata associated with an existing database or table on the source cluster changes over time.
Invalidate Impala Metadata on Destination	Choose the option to run the Impala INVALIDATE METADATA statement per table on the destination cluster after completing the replication. The statement purges the metadata of the replicated tables and views within the destination cluster's Impala upon completion of replication, allowing other Impala clients at the destination to query these tables successfully with accurate results. You must run the INVALIDATE METADATA statement manually for the following scenarios: • If the destination Cloudera Manager does not have an Impala service. • If the source contains User Defined Functions (UDF). **Warning:** However, this operation is potentially unsafe if DDL operations are being performed on any of the replicated tables or views while the replication is running. In general, directly modifying replicated data/metadata on the destination is not recommended. Ignoring this can lead to unexpected or incorrect behavior of applications and queries using these tables or views.
Replication Strategy	Choose one of the following replication strategies to determine whether the file replication tasks should be distributed among the mappers statically or dynamically. • Static distributes file replication tasks among the mappers up front to achieve an uniform distribution based on the file sizes. • Dynamic distributes the file replication tasks in small sets to the mappers, and as each mapper completes its tasks, it dynamically acquires and processes the next unallocated set of tasks. The default replication strategy is Dynamic.
MapReduce Service	Choose the MapReduce or YARN service to use.
Log Path	Enter an alternate path for the logs, if required.

Option	Description
Error Handling	Select the following options as necessary: • Skip Checksum Checks - Determines whether to skip checksum checks on the copied files. If selected, checksums are not validated. Checksums are checked by default. **Note:** You must skip checksum checks to prevent replication failure due to non-matching checksums in the following cases: • Replications from an encrypted zone on the source cluster to an encrypted zone on a destination cluster. • Replications from an encryption zone on the source cluster to an unencrypted zone on the destination cluster. • Replications from an unencrypted zone on the source cluster to an encrypted zone on the destination cluster. Checksums are used for two purposes: • To skip replication of files that have already been copied. If Skip Checksum Checks is selected, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination. • To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data. • Skip Listing Checksum Checks - Whether to skip checksum check when comparing two files to determine whether they are same or not. If skipped, the file size and last modified time are used to determine if files are the same or not. Skipping the check improves performance during the mapper phase. Note that if you select the Skip Checksum Checks option, this check is also skipped. • Abort on Error - Whether to abort the job on an error. If selected, files copied up to that point remain on the destination, but no additional files are copied. Abort on Error is not selected by default. • Abort on Snapshot Diff Failures - If a snapshot diff fails during replication, the replication policy uses a complete copy to replicate data. If you select this option, the policy aborts the replication when it encounters an error instead.

Option	Description
Preserve	Choose the required options to preserve the block size, replication count, permissions (including ACLs), and extended attributes (XAttrs) as they exist on the source file system, or to use the settings as configured on the destination file system. By default source system settings are preserved. When Permission is selected, and both the source and destination clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. When Extended attributes is selected, and both the source and destination clusters support extended attributes, replication preserves them. (This option only displays when both source and destination clusters support extended attributes.) If you select one or more of the Preserve options and you are replicating to S3 or ADLS, the values all of these items are saved in metadata files on S3 or ADLS. When you replicate from S3 or ADLS to HDFS, you can select which of these options you want to preserve. **Note:** To preserve permissions to HDFS, you must be running as a superuser on the destination cluster. Use the Run As Username option to set the username.
Delete Policy	Choose the required options to determine whether the files that were deleted on the source should also be deleted from the destination directory. This policy also determines the handling of files in the destination location that are unrelated to the source. Options include: • Keep Deleted Files - Retains the destination files even when they no longer exist at the source. This is the default option. • Delete to Trash - If the HDFS trash is enabled, files are moved to the trash folder. This is not supported when replicating to S3 or ADLS. • Delete Permanently - Uses the least amount of space; use with caution.
Alerts	Choose to generate alerts for various state changes in the replication workflow. You can choose to generate an alert On Failure, On Start, On Success, or On Abort of the replication job. You can configure alerts to be delivered by email or sent as SNMP traps. If alerts are enabled for events, you can search for and view the alerts on the **Events** tab, even if you do not have email notification configured. For example, if you choose Command Result that contains the Failed filter on the Diagnostics Events page, the alerts related to the On Failure alert for all the replication policies for which you have set the alert appear. For more information, see Managing Alerts and Configuring Alert Delivery.

12. Click Create.

What to do next

After the replication policy runs successfully, you can view the replication job status on the **Replication Policies** page. Verify whether the job starts and runs as expected.

The administrator can verify whether the job starts and runs as expected using the following steps:

- For HDFS data replication, check the cloud storage path (for example, S3 bucket path) to verify whether the data was successfully copied in the specified bucket.
- For Hive metadata replication, verify whether the specified source database, along with tables, partitions, UDFs and column stats are available in the Data Lake HMS instance. For this, the administrator can use a Data Hub cluster and run the corresponding queries using Hue or beeline.
- For Ranger policies, query the Ranger policies to ensure that the Sentry policies are properly mapped as Ranger policies for the right users and groups.

Note: If the CDH source database contains functions, you must explicitly run the `reload function` command to view the migrated replication functions in the target location.

Manage and monitor Hive replication policies

After you create a Hive replication policy in Cloudera Replication Manager, you can perform and monitor various tasks related to the replication policy. You can view the job progress and replication logs. You can edit the advanced options to optimize a job run. You can suspend a job and also activate a suspended job. You can edit the replication policy as necessary.

About this task

On the **Replication Policies** page, you can perform the following actions and tasks on a replication policy and its jobs.

Procedure

- When you click Actions for a Hive replication policy, the following actions appear:

Action	Description
Edit*	Change the replication policy options as required for non-expired policies that are in active or suspended state. Based on the schedule you choose, the replication policy replicates data. You can edit the replication policies to better align with changing requirements. For example, you might want to change the frequency of a policy depending on the data size and importance of the data being replicated. **Note:** A replication policy is associated with a cluster or a cluster pair, therefore you cannot change the clusters in the policy. Optionally, expand a replication policy on the **Replication Policies** page to edit the replication policy options which include frequency (start time cannot be modified if the policy has already started), queue name, maximum bandwidth, and maximum map slots. **Tip:** To optimize the replication policy performance, you can configure the queue name, maximum bandwidth, and maximum map slots as necessary.
Delete	Deletes the replication policy permanently.
Suspend	Suspends a running replication policy. Activate the replication policy, if required.
View Log	Download, copy, or open the log. The log shows a brief output of the stdout and stderr logs of a single step of the latest replication policy job run. You can also view the current job status in the Replication Manager Overview Issues & Updates Job Status column. If the job failed, click **Failed** to view the log details about the job.
View Command Details	Opens the latest Hive replication policy job page. The steps and substeps appear in a tree view. The failed steps are expanded by default, showing the last 15 lines of the log. You can also view the command details for a Hive replication policy on the Overview Issues & Updates panel. **Tip:** To view the complete log for all the jobs, go to the target cluster Cloudera Manager Running Commands page.
Collect diagnostic bundle	Generates a diagnostic bundle for the replication policy. You can download the bundle as a ZIP file to your machine. Ensure that you are logged into the Cloudera Manager instances for both the source and target clusters before you download the bundle in Replication Manager.

Action	Description
*	
To view and use the replication policies with an empty name in Replication Manager, you must understand the following implementation: • If the API version is lower than 51, an existing replication policy with an empty name can be used and updated. However, if you edit the replication policy and provide a name for the replication policy in versions higher or equal to 51, you must ensure that the name conforms to the validation rules. • If the API version is higher or equal to 51, it is mandatory that you provide a unique name to the replication policy to continue using it. This is because API version 51 and higher enforces the validation rules on all the replication policies. To pass the replication policy name validation, you must ensure that the replication policy name is unique. The name can contain letters, numbers, and the _ / - characters. You must also ensure that it does not contain the characters % . ; \ nor any character that is not ASCII printable, which includes the ASCII characters less than 32 and the ASCII characters that are greater than or equal to 127.	

- When you expand the policy details, the **Job History** panel appears.

  You can view the following details on the panel:

  a) Previous jobs, current job, and one future scheduled job if any.

  b) Job details which include:

Job details	Description
Started	Timestamp when the job started.
Ended	Timestamp when the job ended.
Duration	Time taken to complete the job.
Tables	Number of imported or exported tables.
Progress	Current status of a running job.
Expected	Remaining number of files and bytes expected to be copied for a running job.
Copied	Number of files and bytes copied for a running job and completed job.
Failed	Number of files and bytes that failed to be copied for a completed job.
Deleted	Number of files deleted for a completed job.
Skipped	Remaining number of files and bytes skipped from copying for a running job and complete job.

  c) Click Actions to:

  - Abort the job.
  - Re-run an aborted or failed job.
  - View Log for the job. You can download, copy, or open it to track the job and to troubleshoot any issues for the job.

- When you click a job on the **Job History** panel, the following tabs appear:

Tab	Description
General	Shows the following job details: • **Started at** timestamp • **Duration** taken to complete the job • **HDFS Replication Report** to download the job statistics in CSV format • **Hive Replication Report** to download the job statistics in CSV format • **Hive Export/Import** is the number of external tables exported or imported using Hive replication. • Number of **Errors** encountered during the replication job. • **Impala UDFs** is the number of tables exported or imported using Impala. • Job status **Message**.

Tab	Description
Command Details	Shows the details about the commands that ran on the source Cloudera Manager for the job, along with the timestamp.
Setup Error	Shows the stack trace for the commands that ran on the source Cloudera Manager for the failed job.

- You can download the following CSV reports from the General HDFS Replication Report field to track the replication jobs and to troubleshoot issues:

Report	Description
Listing	Lists all the files and directories copied during the replication job.
Status	Shows the complete status report of each file as: • an **Error** occurred and the file was not copied. • a **Deleted** file. • an up-to-date file for which the replication was **Skipped**.
Error Status	Status report of all the copied files with errors. Each file shows the status, path, and message for the copied files with errors.
Skipped Status	Status report of all skipped files. Each file lists the status, path, and message for the databases and tables that were skipped.
Deleted Status	Status report of all deleted files. Each file lists the status, path, and message for the databases and tables that were deleted.
Performance	Summary report about the performance of the running replication job which includes the last performance sample for each mapper that is working on the replication job.
Full Performance	Performance report of the job which includes the samples taken for all mappers during the replication job.

Note: The reports are generated based on the source Cloudera Manager response. If the Cloudera Manager response is interrupted or is not handled as expected, corresponding error messages appear in HTML format in the reports.

- You can download the following CSV reports from the General Hive Replication Report field to track the replication jobs and to troubleshoot issues:

Report	Description
Hive Result	List of replicated tables.
Hive Performance	Performance report for Hive replication.

Using HBase replication policies

To create an HBase replication policy in Cloudera Replication Manager, you must register the on-premises cluster (CDH or Cloudera Private Cloud Base) as a classic cluster in Cloudera Management Console, register cloud account credentials in the Replication Manager service, verify cluster access, and configure minimum ports for replication.

Related Information

HBase replication policy

Support matrix for Cloudera Replication Manager

CDP CLI for Replication Manager

Cloud-to-Cloud HBase Replication on AWS

HBase replication from on-premises to Cloud

HBase cross-region replication for disaster recovery on Azure

Preparing to create an HBase replication policy

Before you create HBase replication policies in Cloudera Replication Manager, you must prepare the clusters, register cloud storage in Replication Manager, and verify cluster access.

Procedure

- Do the source cluster and target cluster meet the requirements to create an HBase replication policy?

  The following image shows a high-level view of the support matrix for HBase replication policies, you must consult the Support matrix for Cloudera Replication Manager for the complete list of supporting clusters and scenarios:

  Figure 8: High-level replication scenarios supported by HBase replication policies

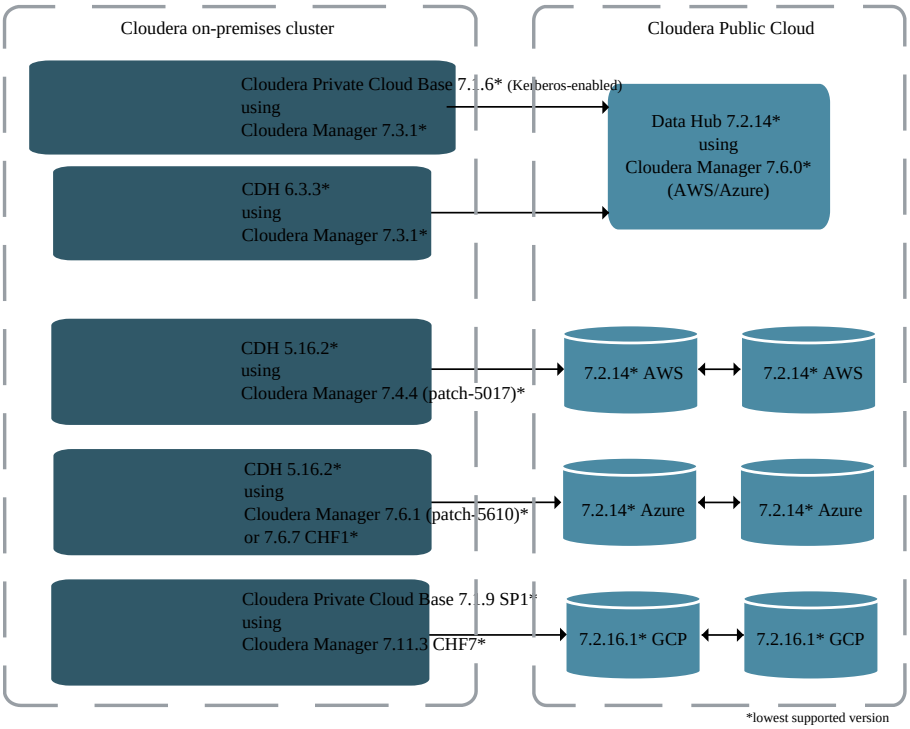

- Is the source CDH cluster or source Cloudera Private Cloud Base cluster registered as a classic cluster on the Management Console?

  CDH clusters and Cloudera Private Cloud Base clusters are managed by Cloudera Manager. To enable these on-premises clusters for Replication Manager, you must register them as Classic Clusters on the Management Console. After registration, you can use them for data migration purposes.

  Important: When you register a Cloudera Private Cloud Base cluster as a classic cluster, ensure that you use only the Cloudera Manager IP Address and Cloudera Manager Port options and not the Register KNOX endpoint (Optional) option.

  For information about registering an on-premises cluster as a classic cluster, see Adding a CDH cluster and Adding a Cloudera Private Cloud Base cluster.

  Note:

  - Cloudera Private Cloud Base 7.1.6 and higher clusters must be Kerberos enabled to use them as source Classic Clusters.
  - You must have *Full Administrator* role on the source on-premises cluster.

- Are the following steps complete on the source Cloudera Private Cloud Base cluster or source CDH cluster (these steps are not required for COD sources)?

  1. Complete Step 1 of Migrating HBase data from CDH/HDP to COD to install the HBase replication plugin parcel in the CDH source clusters.

     This step is applicable for CDH versions 7.2.x that are lower than 7.2.2, versions 7.1.x that are lower than 7.1.5, and for versions lower than 7.x.
  2. Create the /user/hbase folder for the hbase user in HDFS in the source cluster using the following commands:

     ```
     sudo -u hdfs hdfs dfs -mkdir /user/hbase
     sudo -u hdfs hdfs dfs -chown hbase:hbase /user/hbase
     ```

     These commands allow the HBase replication policy to replicate the existing data in the source cluster.

     This step is applicable for Cloudera Manager versions 7.4.3 or lower; Cloudera Manager 7.4.4 only if the API version is lower than v45. The endpoint http://*[***CM_HOST***]*:*[***CM_PORT***]*/api/version shows the API version of the Cloudera Manager.
- Is the required target cluster (Data Hub or COD) available and healthy?

  Note: HBase replication policies do not support source or destination Cloudera Public Cloud clusters created using the Micro Duty template.
- Do you have the necessary permission to run the HBase replication jobs on YARN?

  To verify whether you have the necessary permission, perform the following steps:

  1. Go to the source Cloudera Manager *YARN SERVICE* Configuration tab.
  2. Ensure one of the following conditions is met:

     - The Allowed System Users property must have *hbase*. Otherwise, add hbase to the existing property value.

       This property lists the users permitted (or allowed) to run containers. Note that the users with IDs lower than the Minimum User ID property might be permitted (or allowed) to run containers.
     - The Minimum User ID property is set to a value that is lower than the HBase user's ID.

       To check the HBase user's ID, SSH into a cluster node and run the id hbase command.

       The following sample snippet shows the HBase user's ID when you run the id hbase command:

       ```
       # id hbase
       uid=39993(hbase) gid=39993(hbase) groups=39993(hbase)
       ```

- Is the required cloud credential that you want to use in the replication policy registered with the Replication Manager service?

  For more information, see Working with cloud credentials.

  You can also add the following advanced configuration settings to use Google Cloud, Amazon S3, and ADLS accounts in Replication Manager:

  1. Go to the source Cloudera Manager Clusters *HDFS SERVICE* Configuration tab.
  2. Locate the HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml property.
  3. Add the following key-value pairs to register a Google account to use in Replication Manager:
     - fs.gs.impl=com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystem
     - fs.gs.project.id=*[***ENTER THE PROJECT ID***]*
     - fs.gs.system.bucket=*[***ENTER THE BUCKET NAME***]*
     - fs.gs.working.dir=/
     - fs.gs.auth.service.account.enable=true
     - fs.gs.auth.service.account.email=*[***ENTER THE SERVICE PRINCIPAL EMAIL ID***]*
     - fs.gs.auth.service.account.keyfile=/*[***ENTER THE LOCAL PATH OF THE P12 FILE***]*

     You can add these key-value pairs to register a Google account to use in Replication Manager from Cloudera Public Cloud 7.2.16.1 and higher versions.
  4. Add the following key-value pairs to register an S3 account to use in Replication Manager:
     - fs.s3a.access.key=*[***ENTER THE SESSION ACCESS KEY***]*
     - fs.s3a.secret.key=*[***ENTER THE SESSION SECRET KEY***]*
  5. Add the following key-value pairs to register an ADLS account to use in Replication Manager:
     - fs.azure.account.oauth2.client.id=*[***ENTER THE ABFS STORAGE CLIENT ID***]*
     - fs.azure.account.oauth2.client.secret=*[***ENTER THE ABFS STORAGE CLIENT SECRET KEY***]*
  6. Save and restart the HDFS service for the changes to take effect.
- Do you have the required cluster access to create or view replication policies?
- Have you assigned the managed identity of source roles, Storage Blob Data Owner or Storage Blob Data Contributor, to the destination storage data container and vice versa for bidirectional replication when you are using COD on Microsoft Azure?

  The roles allow writing a snapshot in the destination cluster container.
- Does DNS resolution work as expected between the source and destination clusters?

  Tip: If the destination cluster is not reachable from the source RegionServer hosts, add the hostname and IP address of the destination hosts to the /etc/hosts file on the RegionServers of the source cluster.
- Is the outgoing SSH port open on the Cloudera Manager host?
- Are the following ports open and available for Replication Manager?

Table 10: Minimum ports required for HBase replication policies

Ports	Service	Description
2181 and 16020	Destination hosts of the AWS cluster or ADLS cluster (target cluster), and the Cloudera Manager server port on the source cluster	Verify whether the ports 16020 for worker security group and 2181 for worker, master, and leader groups are open for connection from the source cluster to the destination cluster on AWS or Azure. This ensures that the source HBase service can communicate with Zookeeper and HBase services on the destination hosts uninterruptedly. For more information, see Ports for HBase replication.

Ports	Service	Description
16000	HMaster	Open the port on the Master Nodes (HBase Master Node and any back-up HBase Master node). Before you select the Validate Replication option during the first HBase replication policy creation between two specific clusters, you must ensure that the port is open on the target cluster. **Note:** Irrespective of whether this port is open or not on the Master nodes, Replication Manager displays a warning message to inform you that this port should be open on the target cluster (to communicate with the source cluster) when you choose Validate Replication on the **Select Destination** page during the HBase replication policy creation process.
7180 or 7183	Cloudera Manager Admin Console HTTP	Open on the source cluster to enable Data lake Cloudera Manager to communicate to the on-premises Cloudera Manager. Connects to destination SDX Data Lake Cloudera Manager.
9000	Cloudera Manager Agent	Open on the source and target cluster to retrieve diagnostic and log information.
6000-6049	Cluster Connectivity Manager (CCM)	Required for SSL connections to the Control Plane via CCM to communicate with Replication Manager.
80 or 443	Data transfer from secondary node for AWS / ADLS Gen2	Outgoing port. Open on all the HDFS nodes for AWS and ADLS Gen2.
8443	Data Lake cluster	Outgoing port. Configure the port on the Data Lake cluster as the outgoing port for Cloudera Management Console to communicate with Cloudera Manager and Knox.
8032	YARN Resource Manager	Open on the source and target cluster to access the YARN ResourceManager.

Consider the following best practices while using Cloudera Public Cloud on Microsoft Azure ADLS Gen2 (ABFS):

- Ensure that the on-premises cluster (port 443) can access the *https://login.microsoftonline.com* endpoint. This is because the Hadoop client in the on-premises cluster (CDH/Cloudera Private Cloud Base) connects to the endpoint to acquire the access tokens before it connects to Azure ADLS storage. For more information, see the *General Azure guidelines* row in the Azure-specific endpoints table.
- Ensure that the steps mentioned in the *General Azure guidelines* and *Azure Data Lake Storage Gen 2* rows in the Azure-specific endpoints table are complete so that the endpoint connects to the target path successfully.

- Are the following steps complete to use the Export snapshot user field during the HBase replication policy creation process?

  Replication Manager uses the specified username on the source cluster to export the initial snapshots to the target. The *hbase* user impersonates the specified *Export snapshot user* credentials and permissions to submit the HBase YARN application to export the HBase snapshot.

  To ensure the YARN application succeeds, the *hbase* user must be allowed to impersonate the *exportSnapshotUser* in HDFS by configuring the required properties and values in the HDFS core_site_safety_valve of the source cluster.

  For example, if you want to allow the impersonation from any host and if the *exportSnapshotUser* is in the *repl* user group, you can set the following key-value pairs in source Cloudera Manager Clusters *CORE SETTINGS* Configuration Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml :

  - hadoop.proxyuser.hbase.groups = repl
  - hadoop.proxyuser.hbase.hosts = *

  The Export snapshot user field appears in the **Create HBase replication policy** wizard when the following conditions are true:

  - The target cluster version is 7.2.18.200 and higher.
  - The source cluster version is 7.1.9 SP1 using Cloudera Manager 7.11.3 CHF7 and higher.
  - You choose the Select Source Perform Initial Snapshot option in the **Create HBase replication policy** wizard.

  When you use IDBroker credentials to replicate from Cloudera Private Cloud Base clusters, you must map the Kerberos username to AWS in the *[***USERNAME***]=[***ARN***]* format in the source Cloudera Manager Clusters *KNOX SERVICE* Instances Configuration Knox IDBroker AWS User Mapping property, and allow the *hbase* user to impersonate this user in IDBroker.
- (Optional) Complete the steps mentioned in the Optimize HBase replication policy performance when replicating HBase tables with several TB data FAQ if you choose Perform Initial Snapshot during HBase replication policy creation to replicate HBase tables with several TB data.

What to do next

After the clusters and cloud storage requirements are met, you can create an HBase replication policy.

Related Information

Support matrix for Cloudera Replication Manager
Adding a CDH cluster
Adding a Cloudera Private Cloud Base cluster
Accessing Data Lake services
Migrating HBase data from CDH/HDP to COD
Working with cloud credentials
Ports for Cloudera Replication Manager

Creating HBase replication policy

You can replicate HBase data from a source classic cluster (CDH or Cloudera Private Cloud Base cluster), COD, or Data Hub to a target Data Hub or COD cluster in Cloudera Replication Manager.

Before you begin

Ensure that the HBase replication policy prerequisites are complete.

Procedure

1. On the Management Console Replication Manager Replication Policies page, click Add Policy.

   The **Create Replication Policy** wizard appears.

2. On the General page, choose or enter the following information:

Option	Description
HBase	Creates an HBase replication policy.
Policy Name	Enter a unique name for the replication policy.
Description	Optional. Enter a brief description about the replication policy.

The following image shows a sample General page in the **Create Replication Policy** wizard:

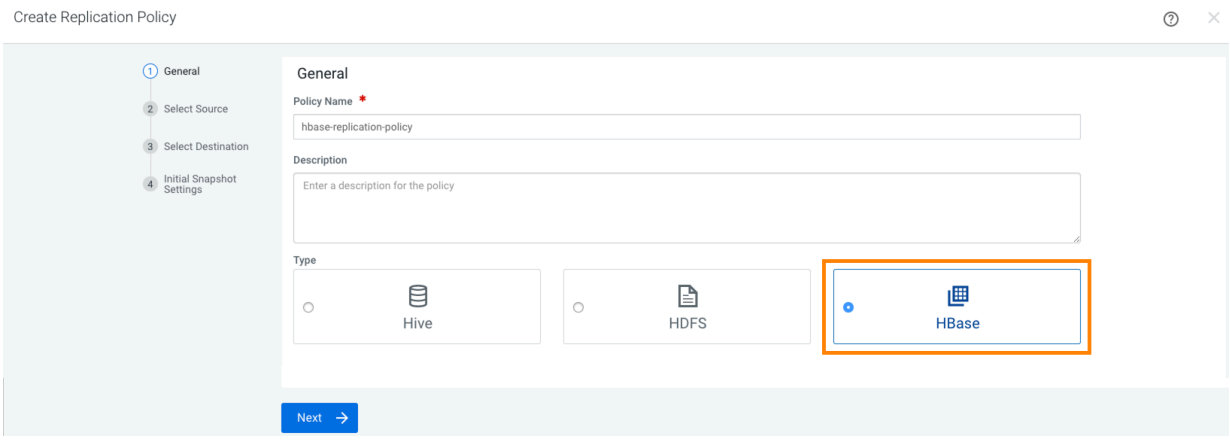

3. Click Next.
4. On the Select Source page, enter or choose the options as required:

Option	Action
Source Cluster or Database	Choose a source cluster. **Note:** HBase replication policies do not support source Cloudera Public Cloud clusters created using the Micro Duty template.
Source Tables	Enter a table name that you want to replicate. Click the Add icon to add more table names.
Perform Initial Snapshot	Select the option to replicate existing data.
Credentials are available in source cluster HDFS service configuration setting	You can choose this option when you want to use a Google Cloud account. You can use this option for S3 and ADLS accounts as well. Before you use this option, ensure that the advanced configuration settings in Preparing to create an HBase replication policy are configured.
Credentials from External Account	Choose this option for S3 and ADLS storage options. This option appears when you choose a Cloudera Private Cloud Base cluster or CDH cluster as the source cluster and you choose Perform Initial Snapshot. Click Add Cloud Credential. In the Add Cloud Credential dialog box, enter a unique name for the cloud credential. Click Save after you choose one of the following cloud storage types and enter the required options: • S3 - Choose an authentication type, enter an access key and secret key. • ADLS - Enter the client ID, tenant ID, and secret key.

Option	Action
Replicate Database	Replication Manager replicates: • Current and future data from the existing tables if you choose the "Perform Initial Snapshot" option. Otherwise, only the future data is replicated. • data from the future tables that are created after policy creation. To replicate data from the future tables successfully, you must create similar empty tables on the target cluster. You can perform this action when you create or add a table to the database on the source cluster. You can choose the Replicate Database option only if the following conditions are true: • Target Cloudera Manager version is 7.11.0 or higher. • Source cluster version is CDH 6.x or higher. CDH 5.16.2 and higher versions also support the Replicate Database option after you upgrade the source cluster Cloudera Manager. • No existing HBase replication policies exist between the source and target clusters. **Tip:** If you want to replicate the new tables that are created after the replication policy creation is complete, you must configure the replication scope to "1" for those tables on the source cluster. To configure the replication scope for a table on the master cluster, run the alter *[***TABLE NAME***]*, {NAME => *[***COLUMN FAMILY***]*, REPLICATION_SCOPE => 1} command for each column family that must be replicated. REPLICATION_SCOPE is a column-family level attribute, where the value '0' means replication is disabled, and '1' means replication is enabled.
Replicate all user tables	Appears after you choose the Replicate Database option. Choose this option to replicate all the HBase tables in the database. This action sets the replication scope to 1 for all the tables and then replicates the tables to the target cluster.
Replicate only tables where replication is already enabled	Appears after you choose the Replicate Database option. Choose this option to replicate only those tables for which the replication scope is already set to 1. This provides you a choice to replicate only the required tables in a database. This option is supported only if the target cluster Cloudera version is 7.2.17.300 using Cloudera Manager 7.11.0-h3 or higher versions, Cloudera 7.2.16.500 using Cloudera Manager 7.9.0-h7 or higher versions, or Cloudera 7.12.0.0. To enable this feature, contact your Cloudera Account team.

Option	Action
Export snapshot user	Enter the custom username. Replication Manager uses the specified username to export the initial snapshots to the target. The option appears after you choose the Select Source Perform Initial Snapshot option. **Note:** Before you specify the username, complete the prerequisites for this field as shown in HBase replication policy prerequisites. To enable this feature, contact your Cloudera Account team.

The following sample image shows the Select Source page in the Create Replication Policy wizard when you choose a COD as source cluster:

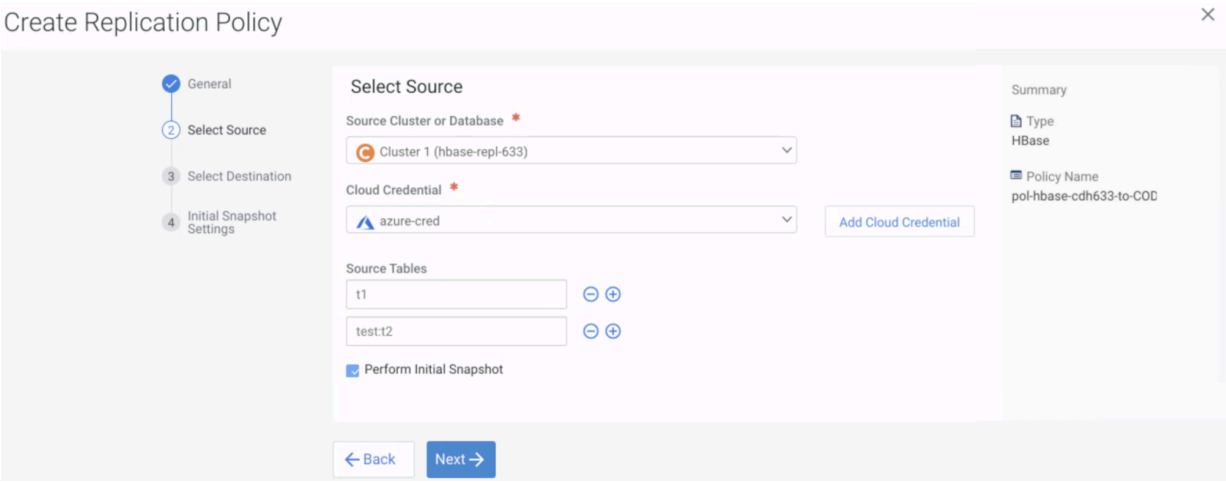

5. Click Next.
6. On the Select Destination page, enter or choose the options as required:

Option	Description
Destination Data Hub or COD	Choose a Data Hub cluster or COD. **Note:** HBase replication policies do not support destination Cloudera Public Cloud clusters created using the Micro Duty template.
Set HBase Replication Machine User	Optional. Choose the option and then enter the username and password. Ensure that you enter the correct password for an existing user because if the password is incorrect, the data is not replicated even though the policy is created successfully. Based on the username and password that you enter, one of the following possible scenarios is implemented by Replication Manager: • If Set HBase Replication Machine User is not selected, an HBase replication machine user is created automatically with an auto-generated username. • If Set HBase Replication Machine User is selected and Create User If Does Not Exist is not selected, ensure that the username you enter exists in the Cloudera User Management System (UMS), otherwise an error message appears. • If Set HBase Replication Machine User is selected and Create User If Does Not Exist is selected and the username does not exist in UMS, the username is created.

Option	Description
Sync Replication User	Optional. Replication Manager validates the existing username with the UMS and synchronizes the new username and password to the destination cluster's environment (and to the source's as well if the source is COD). **Note:** Error appears after you click Sync Replication User if you entered a non-existent username and did not choose Create User If Does Not Exist.
Replicate via a Network Load Balancer	Enable if the source on-premises cluster uses a network load balancer (NLB) to communicate with ZooKeeper and RegionServers of the destination Cloudera Manager of Cloudera Operational Database clusters. You can use this option when the Cloudera Operational Database clusters are isolated and the on-premises clusters can only use NLB to communicate with them. The option is disabled by default if the chosen source and target clusters already have HBase replication policies between them. To enable this feature, contact your Cloudera Account team.
Endpoint	Enter the endpoint details in the *[***NLB ADDRESS***]:[***ZOOKEEPER LISTENER PORT (2181)***]:[***TARGET HBASE ZNODE PATH***]* format. For example, my.nlb.us-west-2.amazonaws.com:2181:/hbase.

The following options appear if the source Cloudera Manager is 7.6.0 or higher:

Option	Description
Rolling HBase Service Restart on Source	[Appears if you select COD or Data Hub as the source cluster] Select to enable automatic rolling restart* of HBase service on the source cluster after the HBase replication policy first-time setup steps are complete. Otherwise, Cloudera Manager performs an automatic full restart* of the service.
Rolling HBase Service Restart on Destination	Select to enable automatic rolling restart* of HBase service on the target cluster as a rolling restart* after the HBase replication policy first-time setup steps are complete. Otherwise, Cloudera Manager performs an automatic full restart of the service.
I want to force the setup of this HBase replication policy	Choose to run the first-time setup configuration between the selected source and destination clusters. This option appears when the selected source or target cluster is part of an existing cluster pair, and one of the following is true about the cluster pair: • No HBase replication policies exist between them. • The other cluster in the pair is currently unreachable. When you select the option, you acknowledge that the existing pairing for the selected source or target cluster will be cleared and the first-time setup will be initiated with the chosen new source or destination cluster.

Option	Description
Validate Replication	Select the option to notify Replication Manager to verify the provided details so that the replication is initiated after the policy creation is complete. **Note:** This option is available for target clusters using Cloudera Manager version 7.6.2 and higher. Before you select the Validate Replication option during the first HBase replication policy creation between two specific clusters, you must ensure that the 16000 port is open on the target cluster. **Note:** Irrespective of whether this port is open or not on the Master nodes, Replication Manager displays a warning message to inform you that this port should be open on the target cluster (to communicate with the source cluster) when you choose Validate Replication on the **Select Destination** page during the HBase replication policy creation process.
*During rolling restart, one node is restarted at a time and this continues until all the nodes in the cluster are restarted. This type of restart ensures that there is no disruption of service. During full restart, all the nodes are shut down at once and restarted simultaneously.	

The following sample image shows the Select Destination page in the Create Replication Policy wizard:

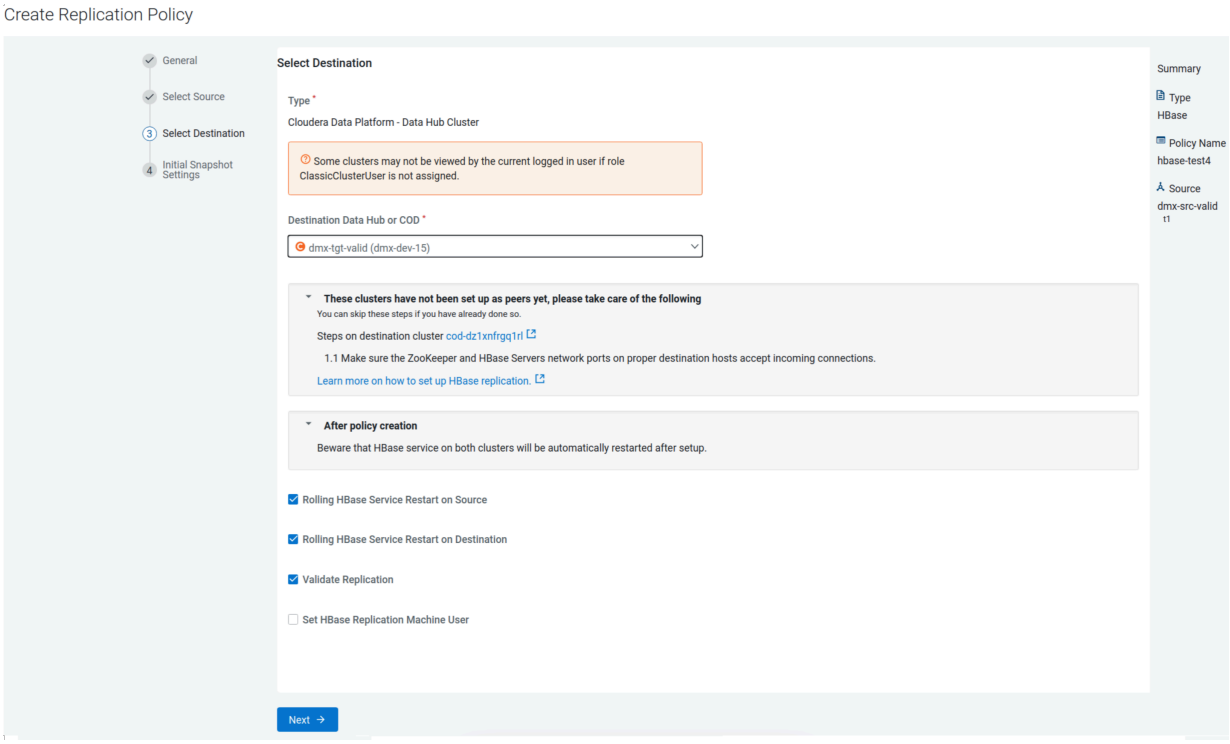

7. Click Next.
8. On the Initial Snapshot Settings page, configure the following options for the source cluster:

Option	Description
YARN Queue Name	Enter the name of the YARN queue for the cluster to which the replication job is submitted only if you are using Capacity Scheduler queues to limit resource consumption. The default value for this field is default.
Maximum Maps Slots	Configure the maximum number of map tasks (simultaneous copies) per replication job. The default value is 20.

Option	Description
Maximum Bandwidth	Adjust this setting so that each map task is throttled to consume only the specified bandwidth. Each map task ((simultaneous copy) is restricted to consume only the specified bandwidth. This is not always exact. The map throttles back its bandwidth consumption during a copy in such a way that the net bandwidth used tends towards the specified value. You can adjust this setting so that each map task is throttled to consume only the specified bandwidth so that the net bandwidth used tends towards the specified value. The default value for the bandwidth is 100MB per second for each mapper. You can adjust the setting only if the source and destination Cloudera Manager instances support this option. To enable this feature, contact your Cloudera Account team.
Maximum parallel snapshots	Specify the maximum number of tables to process in parallel during the initial snapshot export and import step for the HBase replication policy. If you do not enter any value, Replication Manager chooses an appropriate value, depending on the resources in the source and target cluster, to optimize the performance. To enable this feature, contact your Cloudera Account team.

The following sample image shows the Initial Snapshot Settings page in the Create Replication Policy wizard:

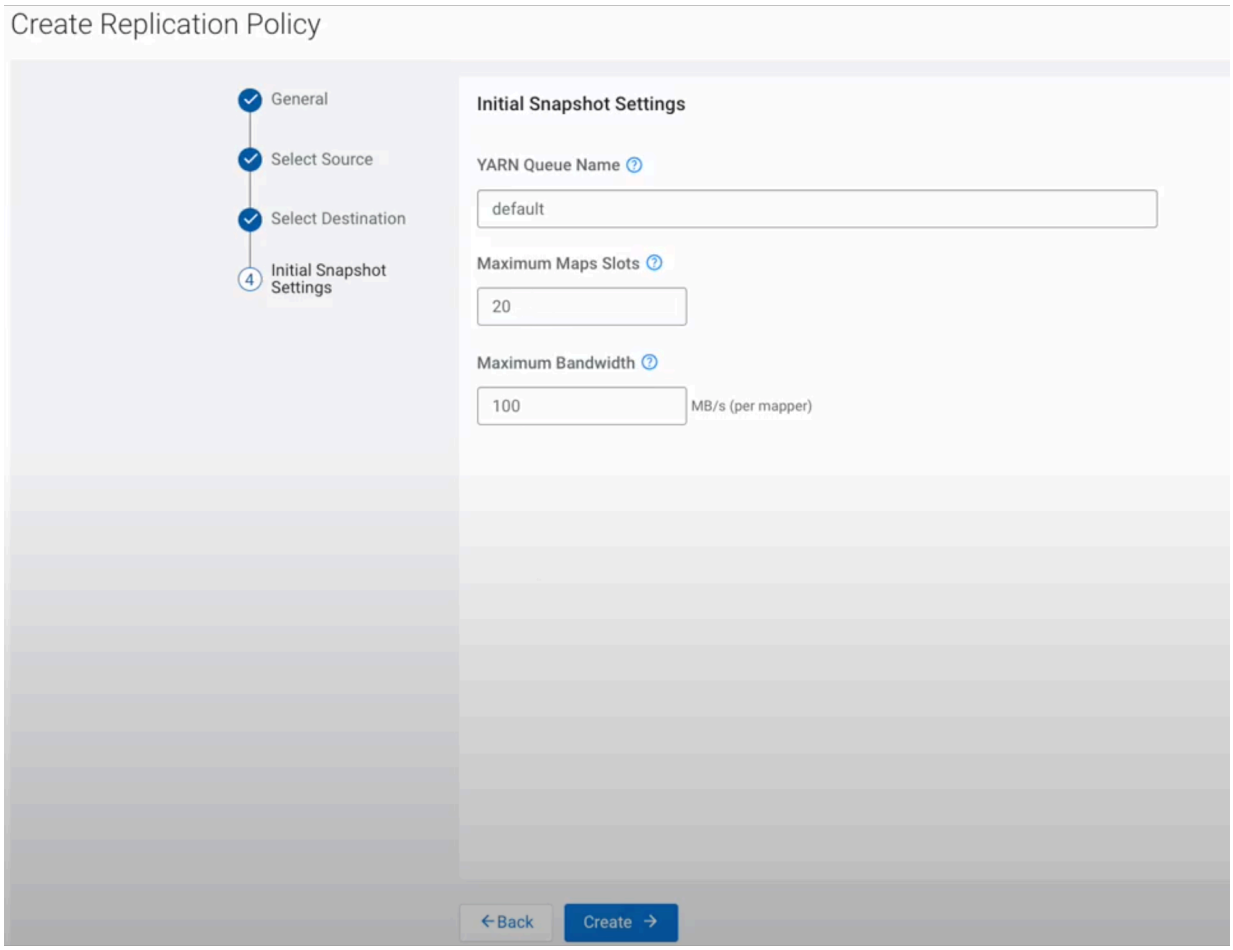

9. Click Create.

10. Restart the HBase service on the on-premises source cluster when the policy status on the **Replication Policies** page shows Manual restart (src) / restarting (dest) or Manual HBase restart needed on source. After the service restart is complete, the setup continues automatically for the replication policy. You do not need to restart the HBase service if the source is COD or Data Hub.

    Important: If the source cluster Cloudera Manager version is 7.6.0 or lower and you are using an on-premises source cluster, you must perform the following steps to complete the HBase replication policy setup:

    a. Restart the HBase service on the on-premises source cluster when the policy status on the **Replication Policies** page shows **Waiting for 'Continue Setup' action call**.
    b. Click Continue setup for the policy on the **Replication Policies** page after the service restart is complete. This action informs Replication Manager to continue the replication policy setup.

    The following image shows the Continue setup option for the HBase replication policy on the Replication Policies page:

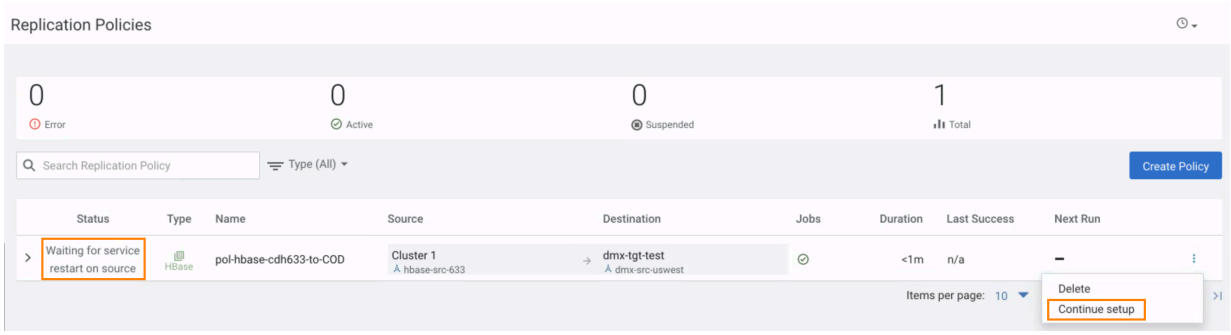

Results

After you create the first replication policy between a source cluster and target cluster (policy that is in setup/service restart state), Replication Manager creates and runs two schedules or jobs. The first schedule shows the service configuration and service restart progress, and the second schedule shows the policy creation progress. Subsequent replication policies between the same source cluster and target cluster creates only one job. Replication Manager restarts the HBase services on both the clusters if they are COD clusters.

Note: The first HBase replication policy between a source cluster and target cluster takes more time to complete because of the first-time setup configuration process that runs in the background. You can continue creating more HBase replication policies while the setup process of the in-progress policies is running in the background. The subsequent replication policies for the same source and target cluster do not run the first-time setup configuration.

What to do next

To verify whether a replication policy is running, you can either click the replication policy on the Replication Policies page, or click Running Commands in Cloudera Manager.

The following sample image shows the Job History page of a HBase replication policy between COD clusters:

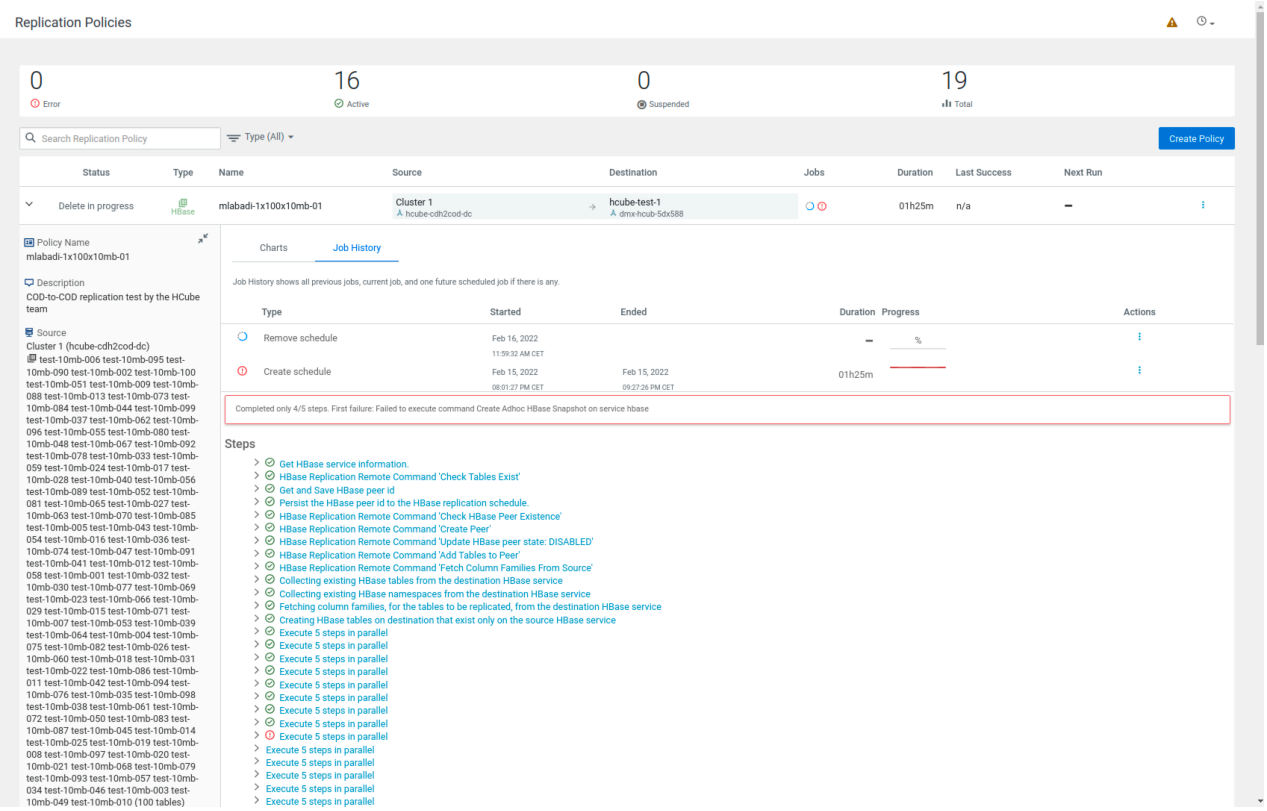

Manage and monitor HBase replication policies

After you create an HBase replication policy in Cloudera Replication Manager, you can perform and monitor various tasks related to the replication policy. You can also view the job progress and replication logs.

About this task

You can perform the following actions and tasks on the replication policy and its jobs on the **Replication Policies** page:

Procedure

- You can edit the policy name and description; view the source cluster name and tables selected in it to be replicated; and view the destination Data Hub or COD and the tables that have been replicated. You cannot remove the tables if you choose Select Source Replicate Database option during the HBase replication policy creation process.
- On the **Job History** tab, you can view the previous replication jobs, current running job, and one future scheduled job for the replication policy. You can perform **Actions** on the HBase replication policy.
- On the **Charts** tab, you can view the HBase RegionServer metrics for a specific replication peer.

Monitor HBase replication policy job details

When you click a replication policy on the "Replication Policies" page, the "Job History" tab appears. On this tab, you can view the previous replication jobs, current running job, and one future scheduled job for the replication policy.

Replication policy job details

The following columns appear for each replication policy:

- **Type** of job. Click the job to view the completed and in-progress steps that Replication Manager takes to complete the job. Here, you can verify whether the first-time setup configuration between the source and destination clusters was successful.

  Tip: You can also view the progress and results of the first-time setup on the Cloudera Manager Running Commands page for the source and destination clusters.

- Timestamp when the job **Started**.
- Timestamp when the job **Ended**.
- **Duration** taken to complete the job.
- **Progress** of the job run.

Actions menu

On the **Actions** menu, you can perform the following tasks on the HBase replication policy:

Option	Description
Edit*	Enables you to edit the replication policy name and description, or delete one or more tables for the HBase replication policy. You can also edit **Source** to delete the tables in the HBase replication policy.
Delete	Removes the HBase replication policy permanently. You can Force Delete an HBase replication policy after Replication Manager fails to delete the replication policy.
Suspend	Pauses an active HBase replication policy. This option appears for target clusters using Cloudera Manager 7.9.0 (Cloudera Manager API** version 51) and higher. **Warning:** When you suspend an HBase replication policy, Replication Manager suspends all the HBase replication policies between the same source and destination cluster. You can suspend only the specified replication policy for target clusters using Cloudera Manager with API version lower than 45. However, the Suspend option is not available for clusters using Cloudera Manager with API versions 45 through 50.
Activate	Resumes a suspended HBase replication policy. **Warning:** When you activate a suspended HBase replication policy, Replication Manager activates all the suspended HBase replication policies between the same source and destination cluster. You can activate only the specified suspended policy for target clusters using Cloudera Manager with API** version lower than 51.
View command details	Opens the latest HBase replication policy job page. The steps and substeps appear in a tree view. The failed steps are expanded by default, showing the last 15 lines of the log. You can also view the command details for a Hive replication policy on the Overview Issues & Updates panel. To view the complete log for all the jobs, go to the target cluster Cloudera Manager Running Commands page.

Option	Description
Retry First Time Setup	Runs the first-time setup configuration between the source and destination clusters if the first-time setup has failed. This option is available only if the first-time setup configuration fails. After you click Retry Create, you can choose: • to set an Set HBase Replication Machine User and Sync Replication User to run the first-time setup. • to opt for Rolling HBase Service Restart on Source and Rolling HBase Service Restart On Destination after the first-time setup is complete. For more information about these options, see *Creating HBase replication policy*.
Retry Failed Snapshots	Reruns the failed initial snapshots (and only the failed ones) in the replication policy if the Replication Manager failed to replicate the existing data of some tables. This option appears if you selected Select Source Perform Initial Snapshot during policy creation and Replication Manager failed to replicate the existing data. In this scenario, the policy status shows **Active** with a *Snapshot Failure* warning message on the **Replication Policies** page for the HBase replication policy.
Collect diagnostic bundle	Generates a diagnostic bundle for the replication policy. You can download the bundle as a ZIP file to your machine. Ensure that you are logged into the Cloudera Manager instances for both the source and target clusters before you download the bundle in Replication Manager.

*

To view and use the replication policies with an empty name in Replication Manager, you must understand the following implementation:

- If the API version is lower than 51, an existing replication policy with an empty name can be used and updated. However, if you edit the replication policy and provide a name for the replication policy in versions higher or equal to 51, you must ensure that the name conforms to the validation rules.
- If the API version is higher or equal to 51, it is mandatory that you provide a unique name to the replication policy to continue using it. This is because API version 51 and higher enforces the validation rules on all the replication policies.

To pass the replication policy name validation, you must ensure that the replication policy name is unique. The name can contain letters, numbers, and the _ / - characters. You must also ensure that it does not contain the characters % . ; \ nor any character that is not ASCII printable, which includes the ASCII characters less than 32 and the ASCII characters that are greater than or equal to 127.

**

The endpoint http://*[***CM_HOST***]*:*[***CM_PORT***]*/api/version shows the API version of the .

Creating triggers and monitoring replication-related metrics in Cloudera Manager

After you create an HBase replication policy between two Cloudera Operational Database (COD) clusters in Cloudera Replication Manager, you can set up the triggers in Cloudera Manager to monitor specific replication-related metrics for the HBase replication policy. When the condition specified in the trigger is met, Cloudera Manager triggers an alert, and you can view the metrics for the replication policy in Cloudera Replication Manager and in Cloudera Manager.

Procedure

1. Go to the source Cloudera Manager Hosts All Hosts page.
2. Click a link in the **Name** column to open the host status page.
3. Click Create Trigger in the **Health Tests** section.

   For information about creating a trigger, see Cloudera Manager Trigger Use Cases. For information about the list of supported HBase RegionServer replication peer metrics, see HBase RegionServer Replication Peer Metrics.

4. Enter the required values and the following expression to create a trigger for a metric with a certain threshold value on the **Create New Trigger** page:

   IF (SELECT *[***ENTER THE METRIC NAME***]* WHERE serviceName=$SERVICENAME AND max(*[***ENTER THE METRIC NAME***]*) > *[***ENTER THE THRESHOLD VALUE***]* AND roleType=RegionServer) DO health:concerning

   The following examples illustrate how you can use triggers to monitor the replication metrics in Cloudera Manager:

   a) The `IF (SELECT log_queue_size WHERE serviceName=$SERVICENAME AND max(log_queue_size) > 10 AND roleType=RegionServer) DO health:concerning` trigger statement triggers the health alert when the value of the log_queue_size is greater than 2.

      This trigger condition is met when the processing of the queue becomes slow which results in the logs being queued or the replication process being halted.

   b) The following sample image shows the chart that appears on the source Cloudera Manager Charts page when the health alert is triggered for the following trigger expression: `IF (SELECT shipped_batches_rate WHERE serviceName=$SERVICENAME AND max(shipped_batches_rate) > 0.01 AND roleType=RegionServer) DO health:concerning`

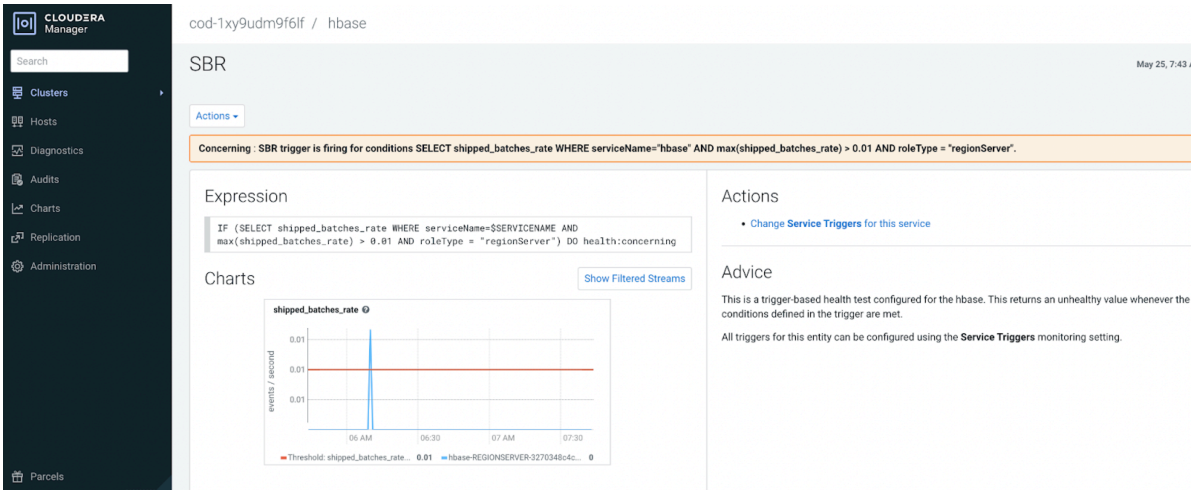

   c) The following image shows the **Chart Builder** where you can configure the options as required and save it for troubleshooting purposes or for future references.

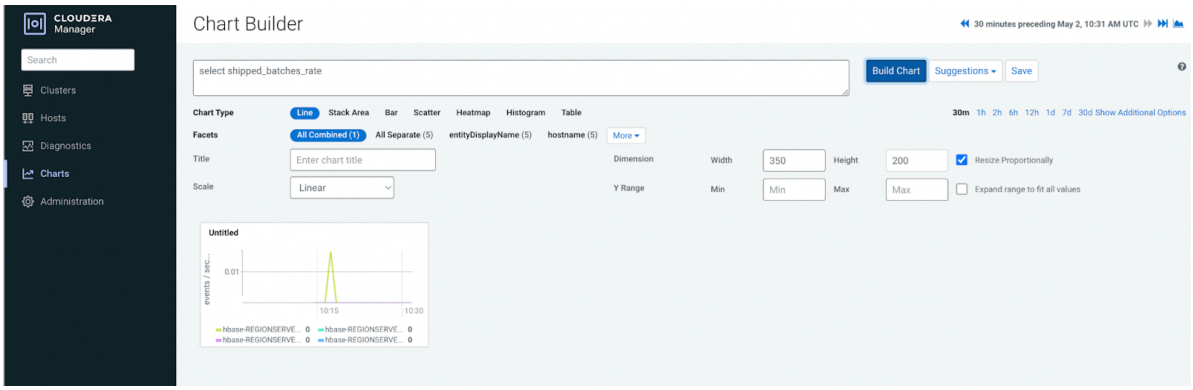

   For information about monitoring the metrics related to replication and HBase health on your COD environment, see Monitor COD metrics.

What to do next

You can view and monitor the metrics in Cloudera Manager and on the **Charts** page in Cloudera Replication Manager.

Monitor HBase RegionServer replication peer metrics in Replication Manager

When you click a replication policy on the "Replication Policies" page, the "Charts" tab appears. On this tab, when you choose the HBase replication peer metric, aggregate rollup level, start time, and end time, a graphical representation of HBase RegionServer metrics specific to the replication peer appears.

Cloudera Manager monitors several metrics which includes performance metrics for the HBase RegionServer. Replication Manager leverages this capability to display the HBase RegionServer metrics specific to a replication peer as a graph on the **Charts** tab for a replication policy, and stores it in the source cluster of the replication policy. You can use these metrics to monitor a HBase replication job and to find and diagnose issues related to the HBase replication peer.

You can view these metrics for a replication peer only if the following conditions are true:

- The source Cloudera Manager API version is 42 or higher and the target Cloudera Manager API version is 53 or higher.

  Tip: The endpoint http://*[***CM HOST***]*:*[***CM PORT***]*/api/version shows the API version of the Cloudera Manager.

- The HBase replication policy has been successfully created and is not in an erroneous state.

Note: If the target Cloudera Manager version is 45 or higher, the graphs generated for HBase replication policies are the same if they share the same source and target clusters. If the target Cloudera Manager version is lower than 45, the graph differs from one HBase replication policy to another because one replication peer is created for each HBase replication policy in these API versions.

For more information about the available metrics, see HBase RegionServer Replication Peer Metrics.

Viewing HBase RegionServer replication peer metrics

You can view HBase RegionServer metrics specific to a replication peer as a graph on the "Charts" tab for a replication policy on the "Replication Policies" page.

Procedure

1. Go to the **Replication Policies** page in Cloudera Replication Manager.
2. Click a successful HBase replication policy that is not in an erroneous state.
3. Go to the **Charts** tab.

4. Choose the following options depending on your requirement:
   a) Choose one of the following HBase replication peer metric:
      - Age of last shipped operation
      - Age of last shipped operation rate
      - Log edits filtered rate
      - Log edits read rate
      - Log queue size
      - Log read in bytes rate
      - Shipped batches rate
      - Shipped ops rate
      - Shipped size in kb rate
      - Shipped hfiles rate
      - Size of hfile refs queue
   b) Choose an aggregate rollup level for the metrics:
      - Raw
      - 10 minutes
      - 1 hour
      - 6 hours
      - Daily
      - Weekly
   c) Choose a Start Time.
   d) Choose an End Time.
5. Click Load Metrics to view the graphs.

Troubleshooting replication policies in Cloudera Replication Manager

The troubleshooting scenarios in this topic help you to troubleshoot issues in Cloudera Replication Manager.

Related Information

Using HDFS replication policy
Using Hive replication policy
Using HBase replication policy
Support matrix for Cloudera Replication Manager

Different methods to identify errors related to failed replication policy

What are the different methods to identify errors while troubleshooting a failed replication policy?

Procedure

You can choose one of the following methods to identify the errors to troubleshoot a job failure:

- On the **Replication Policies** page, click the failed job in the **Job History** pane. The errors for the failed job appear.

  The following sample image shows the **Job History** pane for a replication policy job:

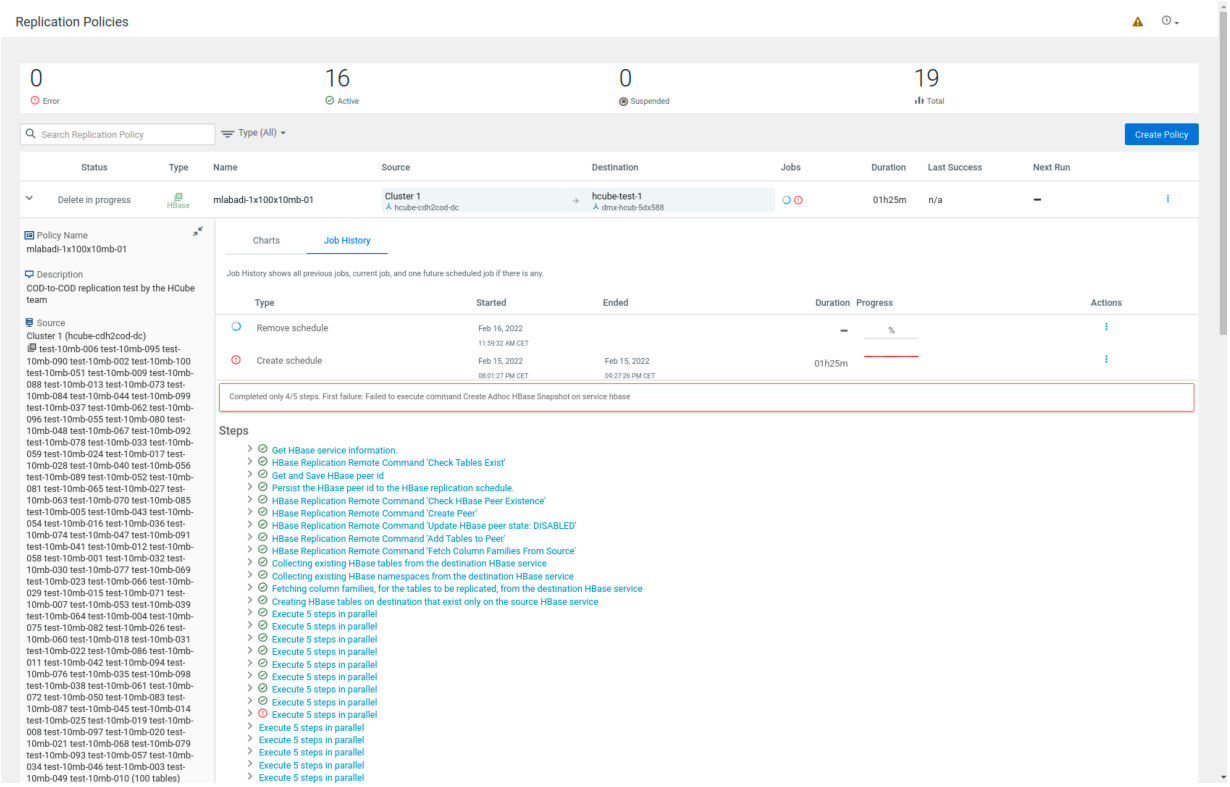

- In the source and target Cloudera Manager, click **Running Commands** on the left navigation bar. The recent command history shows the failed commands.

  The following sample image shows the **Running Commands** page for an HBase replication policy:

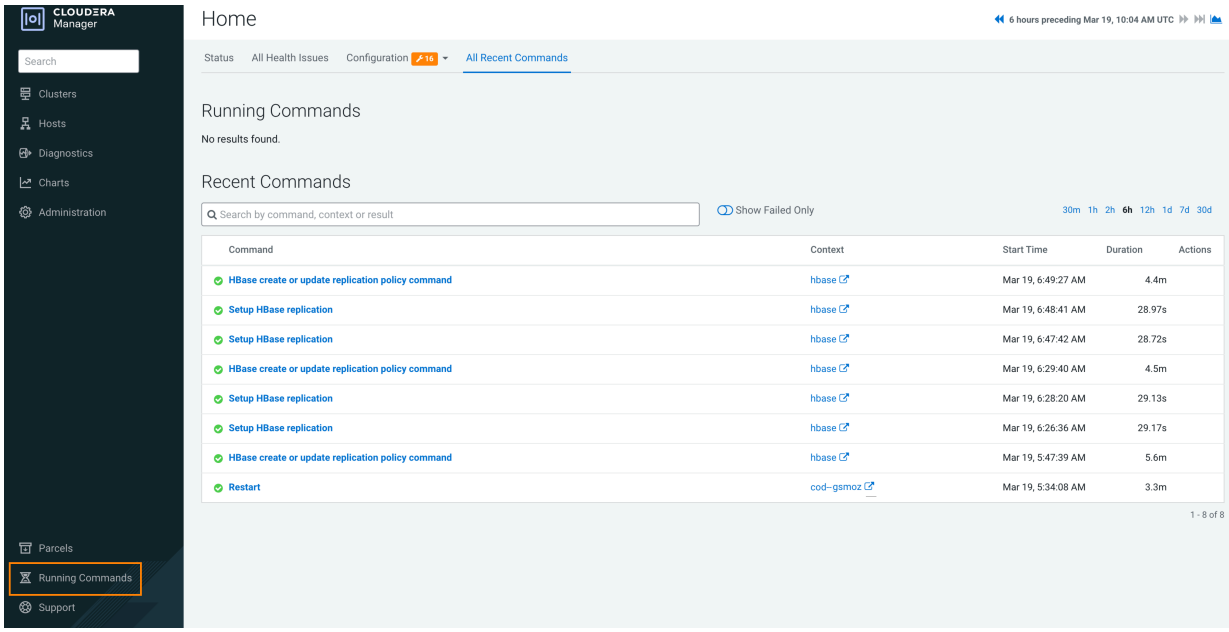

- On the source cluster and target cluster, open the service logs to track the errors (For example, HBase service logs).

  You can also search on the Cloudera Manager Diagnostics Logs page to view the logs.

Replication Policies page does not display all the replication policies

The "Replication Policies" page might not display all the replication policies depending on various factors. In such scenarios, you can choose to reload the page, choose a load page option, or use CDP CLIs to view and monitor the replication policies and its statistics.

Problem

When a Cloudera Manager instance is slow, that is while handling more than 650 replication policies or when it is generally under heavy load, it might slow down the ‘policy list request’ operation. In such scenarios, the replication policies take more time than expected to appear, or might not get displayed on the **Replication Policies** page.

Solution

Procedure

- Force reload page using the option on the **Replication Policies** page.

  You can use this option if the ‘policy list request’ operation has timed out on the Cloudera Manager. You can identify this scenario when the cluster that stores the replication policies shows up in the Error list. Click

  to see the error list and the list of unreachable clusters.

  Tip: Sometimes, Replication Manager fails to reach a healthy Cloudera Manager when there is a temporary networking blip or when there is a load spike on Cloudera Manager. When a cluster becomes unreachable for Replication Manager, the cluster is placed in the list of unreachable clusters. Replication Manager retries to reach the cluster again after 20 minutes. After you confirm that the Cloudera Manager is healthy and expect it to be reachable by Replication Manager, you can force reload the **Replication Policies** page to reconnect every cluster.

- Omit the job history of the policies to speed up the ‘policy list query’ operation using the Never load history option.

  Tip: Depending on your requirements, you can choose one of the following options in

  to load the **Replication Policies** page faster by delaying to load the job history:

  - Delay loading job history when it takes too long attempts to load the job history, but omits the load operation above a certain threshold. By default, Replication Manager uses this option.
  - Never load job history minimizes the load on Cloudera Manager and maximizes Replication Manager performance.
  - Always load job history ensures that the job history is always loaded for all the displayed replication policies.

  The following sample image shows the options for

  :

  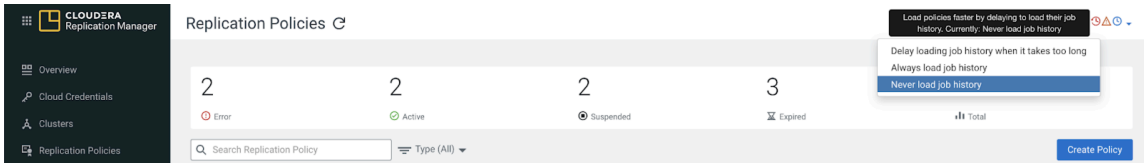

- If the Replication Manager UI still performs slowly and you are not able to view the job history and monitor the dashboards, Cloudera recommends that you use CDP CLIs for Replication Manager. The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option.

  For more information, see CDP CLI for Replication Manager.

  Important: Policy statistics and job details are incorrect if the job history for a replication policy does not load completely. This is because the job history is necessary to decide whether a policy failed or succeeded.

  You can identify this scenario when warnings appear on the **Overview** or **Replication Policies** pages. The following sample images shows the warnings that appear on the **Overview** and **Replication Policies** pages respectively:

  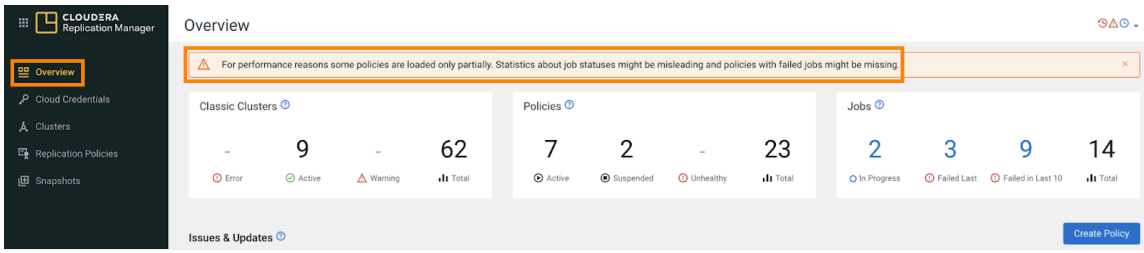

  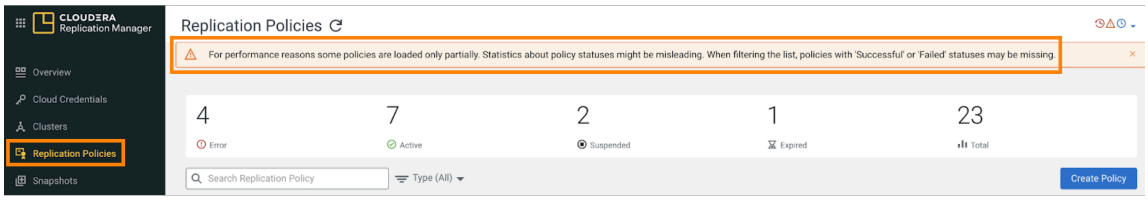

HDFS replication policy fails due to export HTTPS_PROXY environment variable

HDFS replication policies fail when the export HTTPS_PROXY environment variable is set to access AWS through proxy servers. How to resolve this issue?

Remedy

Procedure

To resolve this issue, perform the following steps:

1. Open the core-site.xml file on the source cluster.
2. Enter the following properties in the file:

```
<property>
  <name>fs.s3a.proxy.host</name>
  <description>Hostname of the (optional) proxy server for S3 connection
s.</description>
</property>

<property>
  <name>fs.s3a.proxy.port</name>
  <description>Proxy server port. If this property is not set
    but fs.s3a.proxy.host is, port 80 or 443 is assumed (consistent with
    the value of fs.s3a.connection.ssl.enabled).</description>
</property>
```

3. Save and close the file.
4. Restart the source Cloudera Manager.
5. Run the failed HDFS replication policies in Replication Manager.

   Replication Manager completes the replication successfully.

Cannot find destination clusters for HBase replication policies

When you ping destination clusters using their host names, the source cluster hosts for HBase replication policies do not find the destination clusters. How to resolve this issue?

Cause

This might occur for on-premises clusters such as Cloudera Private Cloud Base clusters or CDH clusters because the source clusters are not on the same network as the destination Data Hub. Therefore, hostnames cannot be resolved by the DNS service on the source cluster.

Remedy

Procedure

- Add the destination Region Server and Zookeeper IP to host name mappings in the /etc/hosts files of all the Region Servers on the source cluster.

  The following snippet shows the contents in a sample /etc/hosts file:

```
10.115.74.181 dx-7548-worker2.dx-hbas.x2-8y.dev.dr.work
10.115.72.28 dx-7548-worker1.dx-hbas.x2-8y.dev.dr.work
10.115.73.231 dx-7548-worker0.dx-hbas.x2-8y.dev.dr.work
10.115.72.20 dx-7548-master1.dx-hbas.x2-8y.dev.dr.work
```

```
10.115.74.156 dx-7548-master0.dx-hbas.x2-8y.dev.dr.work
10.115.72.70 dx-7548-leader0.dx-hbas.x2-8y.dev.dr.work
```

HBase replication policy fails when Perform Initial Snapshot is chosen

An HBase replication policy fails for COD on Microsoft Azure when the "Perform Initial Snapshot" option is chosen but data replication is successful when the option is not chosen. How to resolve this issue?

Cause

This issue appears when the required managed identity of source roles are not assigned.

Remedy

Procedure

Assign the managed identity of source roles, Storage Blob Data Owner or Storage Blob Data Contributor, to the destination storage data container and vice versa for bidirectional replication.

The roles allow writing a snapshot in the destination cluster container.

Optimize HBase replication policy performance when replicating HBase tables with several TB data

Can HBase replication policy performance be optimized when replicating HBase tables with several TB of data if the "Perform Initial Snapshot" option is chosen during HBase replication policy creation?

Complete the following manual steps to optimize HBase replication policy performance when replicating several TB of HBase data if you choose the Perform Initial Snapshot option during the HBase replication policy creation process.

Remedy

Procedure

1. Before you create the HBase replication policy, perform the following steps:
   a) Navigate to the source Cloudera Manager YARN service Configuration tab.
   b) Search for the mapreduce.task.timeout parameter.
   c) Increase the value or set it to 0 to switch off the timeout.
   d) Restart the YARN service.
   e) Navigate to the source Cloudera Manager HBase service Configuration tab.
   f) Search and configure the following key-value pairs:
      - hbase.snapshot.master.timeout.millis = 840000
      - hbase.client.sync.wait.timeout.msec = 180000
      - hbase.client.operation.timeout = 2400000
      - hbase.client.procedure.future.get.timeout.msec = 3000000
      - hbase.hfilearchiver.thread.pool.max=100
      - hbase.snapshot.thread.pool.max=24
   g) Restart the HBase service.
   h) Perform steps e through g on the target Cloudera Manager.

2. When you create the HBase replication policy for the first time using the above configured source cluster, you must increase the Maximum Map Slots value to a higher number on the Advanced Settings page.

   Tip: Calculate this number by multiplying the number of nodes on the source cluster and the number of cores in the node.

3. If Store File Tracking (SFT) is enabled in the target COD, perform the steps mentioned in the COD migration topic after the replication policy creation is complete.

   Note: SFT is enabled by default on Cloudera clusters with COD version 7.2.14.2 and higher using Cloudera Manager versions 7.2.16 and higher.

Partition metadata replication takes a long time to complete

How can partition metadata replication be improved when the Hive tables use several Hive partitions?

Hive metadata replication process takes a long time to complete when the Hive tables use several Hive partitions. This is because the Hive partition parameters are compared during the import stage of the partition metadata replication process and if the exported and existing partition parameters do not match, the partition is dropped and recreated. You can configure a key-value pair to support partition metadata replication.

Procedure

1. Go to the Cloudera Manager Clusters *HIVE SERVICE* Configuration tab.
2. Search for the Hive Replication Environment Advanced Configuration Snippet (Safety Valve) property.
3. Enter the HIVE_IGNORED_PARTITION_PARAMETERS=*[***COMMA SEPARATED LIST OF HIVE PARTITION PARAMETERS***]* key-value pair.

   For example,

   ```
   HIVE_IGNORED_PARTITION_PARAMETERS=transient_lastDdlTime,totalSize,numRow
   s,COLUMN_STATS_ACCURATE,numFiles
   ```

   The partition parameter names you provide are not compared during the import stage of the partition metadata replication process. Therefore, even if the partition parameters do not match between the exported and existing partitions, the partition is not dropped or recreated. After you configure this key-value pair, the import stage of the partition metadata replication process completes faster.
4. Save the changes, and restart the Hive service.

Replicating Hive nested tables

Cloudera Replication Manager does not support Hive nested tables. What do I do if there are Hive nested tables in the source cluster?

Replication Manager does not support Hive nested tables for replication. Therefore, it is recommended that you move the nested tables to a different location in HDFS and then replicate Hive external tables. However, if this is not possible, you can perform the following steps in the given order as a workaround.

Solution

Procedure

1. Create a Hive replication policy on the target cluster. Ensure that the Additional Settings Replication Option Metadata only option is selected to replicate the metadata of required files and directories.

   Note: If you are using Cloudera Public Cloud 7.2.15 or lower and if you are using Amazon S3 as the disaster recovery cluster, ensure that you (the administrator) run the following command after the Hive metadata replication is complete:

   ALTER TABLE table SET LOCATION "*[***[S3A://S3_BUCKET/WAREHOUSE_PATH]***]*";

2. Create a HDFS replication policy on the source cluster to replicate the table data.

Target HBase folder is deleted when HBase replication policy fails

When the snapshot export fails during the HBase replication policy job run, the target HBase folder in the destination Data Hub or COD gets deleted.

Note: This scenario appears if you are using Cloudera Manager versions that are lower than 7.6.7 CHF8, 7.11.0, or 7.9.0-h6 on the source cluster.

You can either revoke the delete permission for the user, or ensure that you use an access key/role that does not have delete permissions to the required storage component.

The following steps show how to create an access key in AWS and an Azure service principal, which do not have delete permission for the storage component.

Solution in AWS

Procedure

If the Data Hub or COD is on Amazon S3, you can perform the following steps to create an access key that does not have delete permission for the storage component:

1. Login to AWS.
2. Click Create policy on the IAM Policies Create policy page.

   The following sample image shows the **Policies** page in the AWS IAM section to create a policy.

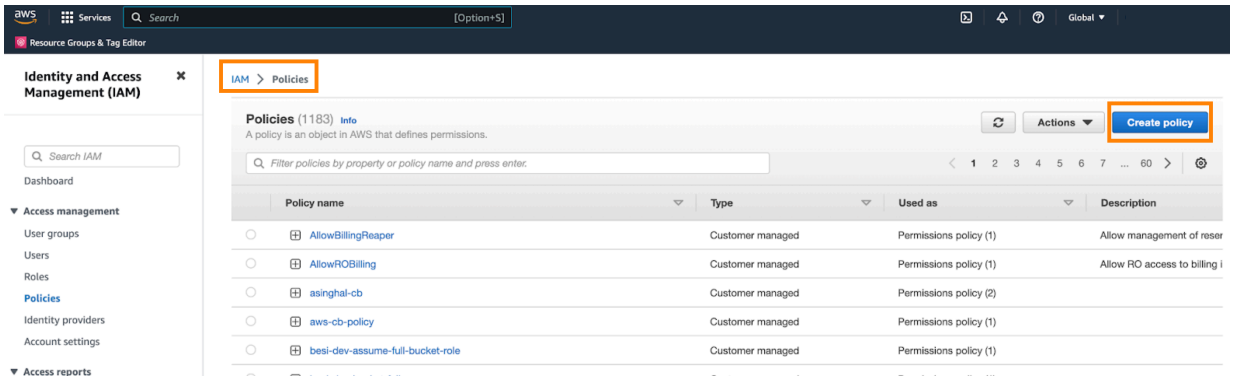

3. Choose the S3 Service, and then choose the following permissions for generic access (assign resources as required):
   - *List/ListBucket*
   - *Read/GetBucketLocation*
   - *Read/GetObject*
   - *Write/AbortMultipartUpload*
   - *Write/PutObject*
   - *Permissions management/PutBucketPublicAccessBlock*
4. Add *Delete/DeleteObject* permission to the target COD cluster's snapshot temporary folder.
   For example, the target COD cluster's snapshot temporary folder might be located in *[***TARGET COD S3 PATH***]*/hbase/.hbase-snapshot/.tmp/*.
5. Enter a Name for the policy, add tags, and click Create policy.
6. Click Add Users to create a user on the IAM Users page.
7. Enter a Name, and click Next.
8. Choose the Attach policies directly option on the **Set permission** page, and then assign the previously created policy to the user.
9. Optionally, add tags and create the user.
10. Click Create access key to create an access key for the user on the IAM Users *[***NEW USER***]* page.
11. On the Security Credentials Access keys page, choose Application running outside AWS. Click Next.
12. Optionally, attach the tags, create and save the access key. This access key is used as an external account for replication.

    How do I verify whether the target HBase folder in the destination Data Hub or COD does not get deleted if the snapshot export fails during the HBase replication policy job run?

    Perform the following steps to verify if the delete operation is allowed for the access key that you previously created:

    a. Run the aws configure --profile delete-test command to setup the credentials in AWS CLI.
    b. Delete an arbitrarily created temporary file from the account using the aws s3 --profile delete-test rm --recursive s3://*[***ACCOUNT NAME***]*/delete-testing/ command.

    The delete operation is not allowed.

Solution in Microsoft Azure

Procedure

If the Data Hub or COD is on ABFS, you can perform the following steps to create an Azure service principal that does not have delete permission for the storage component:

1. Login to Microsoft Azure.

2. Click Add Add a custom role on the Subscriptions Select the subscription Access Control (IAM) page in Microsoft Azure, and complete the following steps:
   a) On the **Basics** tab, provide a name for the role, select Clone a role for the **Baseline permissions** field, and choose Storage Blob Data Contributor for the **Role to clone** field. Click Next.
   b) On the **Permissions** tab, remove the *Microsoft.Storage/storageAccounts/blobServices/containers/delete* and *Microsoft.Storage/storageAccounts/blobServices/containers/blobs/delete* permissions.
   c) Click Review + create.

   The following sample image shows the Access control (IAM) page in Microsoft Azure:

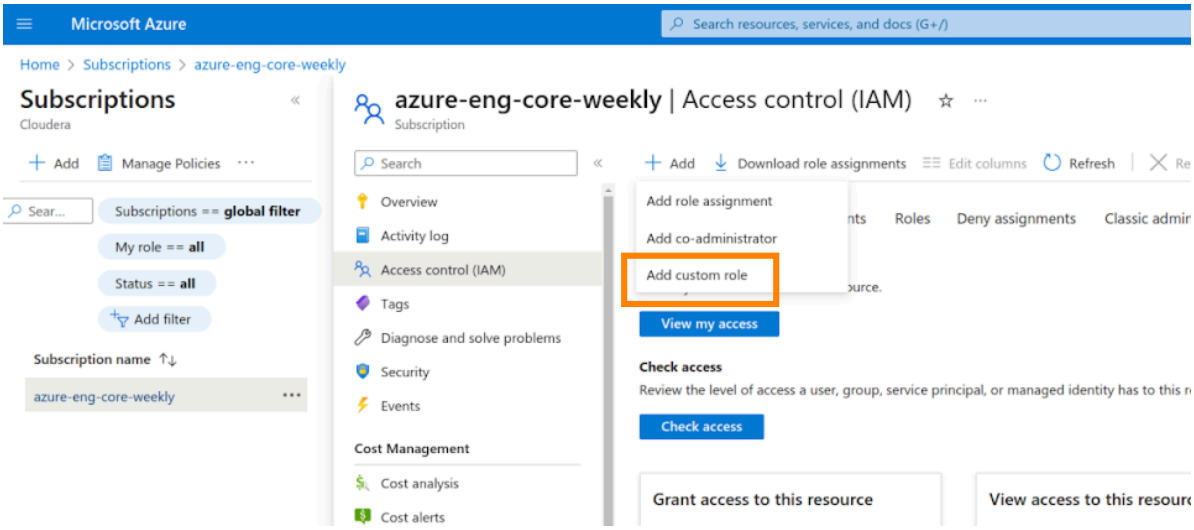

3. Click Add Add role assignment on the Storage accounts Select the account where snapshots will be written Access Control (IAM) page, and complete the following steps:
   a) On the **Role** tab, select the custom role previously created. Click Next.
   b) On the **Members** tab, select User, group, or service principal for **Assign access to** field, and select the required service principal.
   c) Click Review + assign.
   d) Click Review + assign on the **Conditions (optional)** tab.
4. Click Add principal on the Storage accounts *[***ACCOUNT NAME***] [***CONTAINER WHERE SNAPSHOTS ARE TO BE WRITTEN***]* Manage ACL page.
   a) Select the required service principal.
   b) Choose the Execute permission for the required container, and click Save.

   Important: Ensure that no other role containing delete access to the given storage account is assigned to the new service principal on the Access Control (IAM) page.

   How do I verify whether the target HBase folder in the destination Data Hub or COD does not get deleted if the snapshot export fails during the HBase replication policy job run?

   To verify if the delete operation is allowed on the service principal that you previously created, perform the following steps:

   a. Open the Azure Cloud Shell terminal.
   b. Login using the service principal that you created previously using the az login --service-principal -u *[***CLIENT ID***]* -p *[***CLIENT SECRET***]* --tenant *[***TENANT ID***]* command.
   c. Delete an arbitrarily created temporary file from the account using the az storage fs file delete --path *[***TEMPORARY FILE***]* -f data --account-name *[***ACCOUNT NAME***]* --auth-mode login command.

   The delete operation is not allowed.

Replicate HBase data in existing and future tables

Errors might appear when you try to replicate HBase data from existing tables and future tables in a database using the “Replicate Database” option during the HBase replication policy creation process. These errors appear when there are compatibility issues.

The following list shows a few errors that might appear and how to mitigate these issues:

Procedure

- The error *Destination cluster doesn't support replication of all HBase tables. Please change the destination cluster or specify the tables that should be replicated.* appears during HBase replication policy creation process. How to mitigate this issue?

  This error appears if you choose the Select Source Replicate Database option during the HBase replication policy creation process for unsupported cluster versions.

  To mitigate this issue, ensure that the target Cloudera Manager version is 7.11.0 or higher and the source cluster version is CDH 6.x or higher before you choose the Select Source Replicate Database option during the HBase replication policy creation process.
- The error *Exception in thread "main" java.lang.IllegalArgumentException: Cannot add a peer with id=_repl__f1907207cdbc4a528777bb4a316_ba23f09f7328494bbcbf81f40b because that id already exists* appears after creating subsequent HBase replication policies.

  This error appears if you created an HBase peer manually using the *hbase shell* to replicate all column families on a source cluster using CDH 5.x.

  You can replicate the HBase data (existing tables and future tables) in a database only if the target Cloudera Manager version is 7.11.0 or higher and the source cluster version is CDH 6.x or higher. Therefore, even if you create a peer using *hbase shell* in the source cluster version CDH 5.16.2 or any other unsupported version, errors might appear.

Sentry to Ranger migration fails

During the Sentry to Ranger migration process from CDH. 6.3.x to Cloudera Public Cloud 7.3.0.1, the Hive replication command fails at the **Import Ranger permissions** step for Hive replication policies.

Problem

Hive replication command during the Sentry to Ranger migration process from CDH. 6.3.x to Cloudera Public Cloud 7.3.0.1 fails during the “Import Ranger permissions” step for Hive replication policies.

Solution

To solve this issue, you can upgrade the source Cloudera Manager version to 7.11.0.x, 7.12.0.400, or to a higher cluster version.

If you want to continue using CDH 6.3.x as the source cluster, you can upgrade the target Cloudera Manager version to 7.11.0.x, 7.12.0.400, or to a higher cluster version to mitigate the issue.

Cloudera Replication Manager

Replication Manager Reference

Date published: 2019-11-07
Date modified: 2024-07-08

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

CDP CLI for Cloudera Replication Manager..... 4
CDP CLI options for Replication Manager..... 4

Adding cloud credentials in Replication Manager using CDP CLI.....9

Creating HDFS replication policy using CDP CLI.....10
HDFS replication policy definition JSON file.....11
Managing HDFS replication policies using CDP CLI..... 15

Creating Hive replication policy using CDP CLI..... 17
Hive replication policy definition JSON file..... 18
Managing Hive replication policies using CDP CLI..... 22

Creating HBase replication policy using CDP CLI..... 24
HBase replication policy definition JSON file..... 26
Managing HBase replication policies using CDP CLI..... 29

CDP CLI for Cloudera Replication Manager

You can use CDP CLI commands to create and manage HDFS, Hive, and HBase replication policies in Cloudera Replication Manager. You can also register the ABFS and AWS cloud credentials to use in Replication Manager. The CDP CLI commands for Replication Manager are under the "replicationmanager" CDP CLI option.

Prerequisites

To use CDP CLI commands for Replication Manager, ensure that the following are available:

1. CDP CLI client.

   For information about installing a CDP CLI client, see Installing CDP CLI client.
2. Access to CDP CLI.

   Choose one of the following methods to log into CDP CLI:

   - Interactive method. This login method grants a 12-hour access key to the CLI. For more information, see Logging into CLI/SDK.
   - Traditional method. In this method, you generate access credentials and configure the ~/.cdp/credentials file with the key pair. This login method allows you to withdraw the access permission by removing the access credentials from the ~/.cdp/credentials file. For more information, see Generating an API access key and Configuring CDP client with the API access key.

Access CLI help

CDP CLI includes help that can be accessed by using the cdp help command. For information about a certain CDP CLI, use the cdp *[***MODULE-NAME***]* *[***COMMAND-NAME***]* help command.

You can also find all of the CDP CLI commands in the CDP CLI Reference documentation.

Related Information

Introduction to Replication Manager

Support matrix for Cloudera Replication Manager

Using HDFS replication policy

Using Hive replication policy

CDP CLI options for Replication Manager

You can use the CDP CLI commands to create, activate, and delete HDFS, Hive, and HBase replication policies.

CDP CLI options

You can use the following CDP CLI options to perform tasks in Replication Manager:

CDP CLI option	Description
abort-policy*	Aborts all running instances of the specified replication policy. Provide the CRN of the cluster where the replication policy is created and stored.
activate-hbase-policy	Resumes the specified suspended HBase replication policy. If the target Cloudera Manager API version is higher than 50, the command resumes all the HBase replication policies between the same source and destination cluster. However, if the target Cloudera Manager API version is lower than 45, the command resumes only the specified HBase replication policy. This command is not available for target Cloudera Manager API versions between 45 and 50.

CDP CLI option	Description
activate-policy	Resumes the specified suspended HDFS or Hive replication policy. Provide source CRN for HDFS replication policies and target CRN for Hive replication policies.
activate-snapshot-policy*	Resumes a suspended snapshot policy run.
collect-diagnostic-bundle	Triggers the diagnostic bundle collection for the specified replication policy in the target Cloudera Manager. Use the commandId in the command output to download the diagnostic bundle.
continue-hbase-setup	Continues the first-time setup on the target cluster. **Note:** If an HBase replication policy has a Classic Cluster source, the HBase service on the source is not restarted automatically. You must restart the source manually, and then run the continue-hbase-setup command. This command is not required if the source cluster is not a Classic Cluster.
create-abfs-access-key-credential*	Creates an ABFS access key-based account in Replication Manager.
create-abfs-client-key-credential*	Creates an Azure active directory service principal account in source Cloudera Manager.
create-abfs-credential	Adds a ABFS cloud credentials to use in Replication Manager. **Important:** Before you register the ABFS cloud credentials in Replication Manager, ensure that you update cloud credentials in the source cluster Cloudera Manager UI. For more information, see Registering ABFS Cloud account in Replication Manager.
create-aws-access-key-credential*	Creates an AWS access key cloud credential.
create-aws-credential	Adds AWS cloud credentials to use in Replication Manager. **Important:** Before you add AWS credentials, see Registering Amazon S3 cloud account.
create-aws-iam-credential*	Creates an AWS IAM-role based cloud credential.
create-aws-idbroker-credential*	Creates an AWS IDBroker cloud credential.
create-gcs-private-key-credential*	Creates a GCS private key cloud credential.
create-hbase-policy	Creates an HBase replication policy with the given name on the specified cluster.
create-policy	Creates an HDFS or Hive replication policy based on the provided parameters. Provide source CRN for HDFS replication policies and target CRN for Hive replication policies. **Important:** Only a non-machine user can run this CDP CLI command. Otherwise, an HTTP 500 error appears.
create-snapshot-policy*	Creates a HDFS or HBase snapshot policy depending on the details you provide in the snapshot definition.
delete-credential	Deletes the registered credentials from Replication Manager.
delete-hbase-policy	Deletes the specified HBase replication policy permanently. Enter --force if a normal delete fails. For example, when the source cluster is unreachable. Otherwise, enter --no-force.
delete-policy	Deletes the specified HDFS or Hive replication policy. Provide source CRN for HDFS replication policies and target CRN for Hive replication policies.
delete-snapshot-policy*	Deletes the specified snapshot policy.

CDP CLI option	Description
download-diagnostic-bundle	Generates a bundleFile, a binary string in base64 encoded format, as a ZIP file. Run a script to save the response as a file to your machine. For example, cat response.json | jq -r '.bundleFile' | base64 -D > bundle.zip. The file is generated only if the bundleStatus in the get-command-status command shows *DOWNLOADABLE WITH CLI*.
get-cluster-config	Retrieves the configuration of a specified cluster.
get-command-status	Returns the current status of the collect-diagnostic-bundle command. You can also view the status of any Cloudera Manager command using the relevant input command ID.
get-credentials	Returns the cloud credentials that are available on the specified cluster. If you provide the credential name and credential ID, the ID is considered by the CDP CLI command.
get-hbase-time-series*	Returns the time series data for an HBase replication peer depending on the provided parameters. You can enter one of the following HBase replication peer metric: • age_of_last_shipped_operation • log_edits_filtered_rate • log_edits_read_rate • log_queue_size • log_read_in_bytes_rate • shipped_batches_rate • shipped_ops_rate • shipped_size_in_kb_rate • age_of_last_shipped_operation_25th_percentile • age_of_last_shipped_operation_75th_percentile • age_of_last_shipped_operation_90th_percentile • age_of_last_shipped_operation_98th_percentile • age_of_last_shipped_operation_99th_percentile • age_of_last_shipped_operation_999th_percentile • age_of_last_shipped_operation_max • age_of_last_shipped_operation_mean • age_of_last_shipped_operation_median • age_of_last_shipped_operation_min • age_of_last_shipped_operation_rate • shipped_hfiles_rate • size_of_hfile_refs_queue You can enter one of the following aggregate rollup level for time series data: • RAW • TEN_MINUTELY • HOURLY • SIX_HOURLY • DAILY • WEEKLY
get-snapshot-policy*	Retrieves the details about the specified snapshot policy.
list-all-credentials	Provides a detailed list of cloud credentials across all clusters that are available for Replication Manager.
list-cluster-service-statuses	Provides the current status of the services on all the clusters that are available for Replication Manager.
list-clusters	Provides a detailed list of all the clusters that are available for Replication Manager.
list-paired-hbase-clusters	Lists the paired clusters to use for HBase replication policies. The first-time setup is complete for paired clusters.

CDP CLI option	Description
list-policies	Lists the available replication policies across all the clusters. Enter the source cluster CRN to view the HDFS replication policies, and target cluster CRN to view all the Hive and HBase replication policies.
list-policy-jobs*	Returns the list of jobs triggered by the replication policy. This command is a paginated operation, therefore multiple API calls might be required to retrieve the entire dataset of results. You can use one of the following methods to retrieve the results: • Use --no-paginate argument to disable pagination. • Use --max-items *[***TOTAL NUMBER OF ITEMS TO RETURN***]* --starting-token *[***TOKEN TO START PAGINATING***]* --page-size *[***PAGE SIZE***]* to return a specific number of jobs at a time.
list-snapshot-policies*	Returns a list of all the snapshot policies across all available clusters.
list-snapshot-policy-jobs*	Retrieves the snapshot history of the specified snapshot policy.
repair-hbase-policy*	Runs the last failed command for a failed HBase replication policy.
rerun-policy*	Runs the specified HDFS or Hive replication policy.
restore-snapshot*	Restores the specified snapshot.
retry-failed-hbase-first-time-setup*	Runs the first time setup configuration between the clusters in the specified HBase replication policy if the first time setup fails for the replication policy. You can use the following arguments as required: • `--machine-user [***ENTER USERNAME AND PASSWORD OF MACHINE USER***]` Syntax: user=*[***STRING***]*, password=*[***STRING***]*, createUser=*[***ENTER 'TRUE' TO CREATE A NEW MACHINE USER, OR 'FALSE' TO RETURN AN ERROR IF THE USER DOES NOT EXIST***]* • `--source-restart-type [***ENTER RESTART OR ROLLING_RESTART TO SPECIFY THE RESTART TYPE***]` • `--target-restart-type [***ENTER RESTART OR ROLLING_RESTART TO SPECIFY THE RESTART TYPE***]`
retry-failed-hbase-snapshots*	Reruns only the failed initial snapshots in the replication policy if the policy failed to replicate existing data in some or all the tables
suspend-hbase-policy	Pauses an active HBase replication policy.
suspend-policy	Stops all the replication tasks defined for the HDFS or Hive replication policy. Provide source CRN for HDFS replication policies and target CRN for Hive replication policies.
suspend-snapshot-policy*	Suspends the specified snapshot policy.
update-abfs-access-key-credential*	Updates an existing ABFS access key-based account in Replication Manager.
update-abfs-client-key-credential*	Updates an existing Azure active directory service principal account in source Cloudera Manager.
update-aws-access-key-credential*	Updates an existing AWS access key cloud credential.
update-aws-iam-credential*	Updates an existing AWS IAM-role based cloud credential
update-aws-idbroker-credential*	Updates an existing AWS IDBroker cloud credential.
update-gcs-private-key-credential*	Updates and existing GCS private key cloud credential.

CDP CLI option	Description
update-hbase-policy	Modifies an existing HBase replication policy. You can change the name and description of the replication policy. You can also delete one or more tables in the replication policy.
update-policy*	Modifies the HDFS or Hive replication policy. To modify the replication policy, make appropriate changes in the JSON file, save the file, and use it in the update-policy command. Use the rerun-policy command to run the policy. **Note:** Some elements such as cloud credential, source or target cluster, or source dataset cannot be modified. Create another replication policy to use the new values.
update-snapshot-policy*	Modifies an existing snapshot policy depending on the arguments you choose.
verify-hbase-cluster-pair	Verifies whether the specified pair of clusters are paired, not paired, or wrongly paired. The wrongly paired status indicates that one or both of the specified clusters are paired to other clusters.
*The option is a technical preview feature and is not ready for production deployment. The components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the use of preview components, which should be used by customers at their own risk. For more information, contact your Cloudera account team.	

CDP CLI options to create a replication policy

Note: Only a non-machine user can run the replicationmanager create-policy CDP CLI command to create a replication policy.

The following parameters are available in the replicationmanager create-policy CDP CLI option:

Parameter	Description
--cluster-crn	Enter the cluster CRN. To determine the cluster CRN, run the list-clusters command. Provide the following CRN when you create a replication policy: • Source cluster CRN for HDFS replication policy. • Target cluster CRN for Hive replication policy. • Target cluster CRN for HBase replication policy.
--policy-name	Enter a unique name for the replication policy.
--policy-definition	Enter the policy definition in the $(cat *[***POLICY DEFINITION FILE NAME***]*) format, and then enter the cluster CRN and policy name as command arguments.
--cli-input-json	Enter the policy definition JSON file path using the cat command to read the data from the file to create, and run the replication policy.
--generate-cli-skeleton	Shows a policy definition template in JSON format. You can copy the output of this command to a file, add the required parameters, and save it as a JSON file. You can use the filename while creating a replication policy.

Related Information

CDP CLI options for Replication Manager

Technical preview CDP CLI options for Replication Manager

Adding cloud credentials in Replication Manager using CDP CLI

To replicate data to a storage cloud account, you must register the cloud credentials, so that the Replication Manager can access your cloud account. The supported cloud storage accounts are Amazon S3 and Azure Blob Filesystem (ABFS). You can add, update, or delete AWS or ABFS cloud credentials to use in Replication Manager using CDP CLI.

About this task

Perform one of the following steps, as necessary, to manage cloud credentials in Replication Manager using CDP CLI:

Procedure

- Add ABFS credentials:

```
replicationmanager create-abfs-credential --name [***CREDENTIAL NAME***]
 --clusters [***CLUSTER CRNS SEPARATED BY SPACE***] --type [***ACCESSKEY
 OR CLIENTKEY***] --access-key [***ABFS ACCESS KEY***] --storage-accou
nt-name [***ABFS STORAGE ACCOUNT NAME***] --client-id [***CLIENT ID
 OF ACTIVE DIRECTORY SERVICE PRINCIPAL ACCOUNT***] --client-secret-
key [***CLIENT KEY OF ACTIVE DIRECTORY SERVICE PRINCIPAL ACCOUNT***] --t
enant-id [***TENANT ID OF ACTIVE DIRECTORY SERVICE PRINCIPAL ACCOUNT***]
```

Note:

- Provide the access key and storage account name if you choose ACCESSKEY type.
- Provide the client ID, client secret key, and tenant ID if you choose CLIENTKEY type.

Before you add ABFS credentials, see Registering ABFS Cloud account in Replication Manager.

- Add AWS credentials:

```
replicationmanager create-aws-credential --name [***CREDENTIAL NAME***]
 --clusters [***CLUSTER CRNS SEPARATED BY SPACE***] --type [***IAM OR
 ACCESSKEY***] --access-key [***AWS ACCESS KEY***] --secret-key [***AWS
 SECRET KEY***]
```

Important: Before you add AWS credentials, see Registering Amazon S3 cloud account.

- Update ABFS credentials:

```
replicationmanager update-abfs-credential --name [***CREDENTIAL NAME***]
 --type [***ACCESSKEY OR CLIENTKEY***] --access-key [***ABFS ACCESS
 KEY***] --storage-account-name [***ABFS STORAGE ACCOUNT NAME***] --clien
t-id [***ABFS CLIENT ID***] --client-secret-key [***ABFS CLIENT SECRET
 KEY***] --tenant-id [***ABFS TENANT ID]
```

- Update AWS credentials:

```
replicationmanager update-aws-credential --name [***CREDENTIAL NAME***] --
type [***IAM OR ACCESSKEY***] --access-key [***ABFS ACCESS KEY***] --secre
t-key [***AWS SECRET KEY***]
```

Note:

- No parameters are required for the IAM type.
- Provide the access key and secret key if you choose the ACCESSKEY type.

- View all the available registered credentials in a cluster:

```
replicationmanager get-credentials --cluster-crn [***CLUSTER CRN***]
```

- Delete a registered credential from Replication Manager:

```
replicationmanager delete-credential --name [***CREDENTIAL NAME***]
```

Creating HDFS replication policy using CDP CLI

You can use CDP CLI to create an HDFS replication policy. Only a non-machine user can run the "replicationmanager create-policy" CDP CLI command to create a replication policy.

Procedure

1. Log into Replication Manager CDP CLI setup:

```
cdp --profile [***PROFILE NAME***] replicationmanager
```

2. List the clusters to verify whether the required clusters are available:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-clusters
```

3. Verify whether the required services are running on the source cluster:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-cluster-service
-statuses
```

4. Ensure that the cloud credentials are available:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-all-credentials
```

5. Perform the following steps to create a policy definition JSON file:
   a) Open a policy definition JSON file, or copy the output of the following command to a JSON file to generate a policy definition JSON file.

```
cdp --profile [***PROFILE NAME***] replicationmanager create-policy --
generate-cli-skeleton
```

```
cdp --profile hdfs1 replicationmanager create-policy --generate-cli-
skeleton > rm_hdfs1.json
```

   b) Enter the required parameters. Remove the key-value pairs that are not required in the policy definition JSON file for the specific policy. For example, you can remove the hiveArguments key-value pairs when you create an HDFS replication policy.
   c) Save the file.

6. Create the HDFS replication policy.

```
cdp --profile [***PROFILE NAME***] replicationmanager create-policy --cli-input-json [****POLICY DEFINITION JSON FILE PATH USING CAT***]
```

The `cat` command reads the data from the policy definition JSON file.

cdp --profile local-dev replicationmanager create-policy --cli-input-json “$(cat temp/rm_hdfs1.json)”

Results

Replication Manager creates the replication policy and initiates data replication.

What to do next

Verify whether the replication policy is running as expected on the Cloudera Replication Manager Replication Policies page, or run the following command:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-policies --cluster-crn [***CRN OF THE CLUSTER WHERE THE REPLICATION POLICIES ARE STORED***]
```

Related Information

Creating HDFS replication policy using Replication Manager

HDFS replication policy definition JSON file

The policy definition JSON file contains all the parameters required to create an HDFS replication policy. When you edit the file to define an HDFS replication policy, remove the parameters that are not required for the replication policy.

Parameters in HDFS replication policy definition JSON file

The following table lists the parameters in the policy definition JSON file that are required for an HDFS replication policy:

Parameter	Description
clusterCrn	Provide the source cluster CRN for the HDFS replication policy. Replication Manager saves the replication policy in the specified cluster CRN.
policyName	Provide a unique name for the replication policy.
name	Provide the unique name for the policy.
type	Provide FS to create an HDFS replication policy.
path	Provide the HDFS file path in the source cluster.
mapReduceService	Provide the MapReduce or YARN service for the replication policy to use.
logPath	Provide an alternate path for the logs, if required.
replicationStrategy	Provide one of the following options to determine whether the file replication tasks must be distributed among the mappers statically or dynamically: • STATIC - Static replication distributes file replication tasks among the mappers up front to achieve an uniform distribution based on the file sizes. • DYNAMIC - Dynamic replication distributes the file replication tasks in small sets to the mappers, and as each mapper completes its tasks, it dynamically acquires and processes the next unallocated set of tasks. Default is DYNAMIC.

Parameter	Description
skipChecksumChecks	Provide true to skip checksum checks. Default is true. Checksums are used to perform the following tasks: • To skip replication of files that have already been copied. When set to true, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination. • To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data.
skipListingChecksumChecks	Provide true to skip checksum check while comparing two files to determine whether they are the same or not. Otherwise, the file size and last modified time are used to determine if files are the same or not. Skipping the check improves performance during the mapper phase. **Note:** If you set skipChecksumChecks to false, the skipListingChecksumChecks is also set to false by default.
abortOnError	Provide true to stop the policy job when an error occurs. This ensures that the files copied up to that point remain on the destination, but no additional files are copied. Default is false.
abortOnSnapshtDiffFailures	Provide true to stop the replication job if a snapshot diff fails during replication.
preserve	Provide true to preserve the block size, replication count, permissions (including ACLs), and extended attributes (XAttrs) as they exist on the source file system. • blockSize • replicationCount • permissions • extendedAtrributes Provide false to use the settings as configured on the destination file system. By default, the source system settings are preserved. In an HDFS replication policy: • when the permissions parameter is set to true and both the source and destination clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. • when extendedAtrributes is set to true and both the source and destination clusters support extended attributes, the replication process preserves them. • when you select one or more of the Preserve options, and you are replicating *to* S3 or ADLS, the values of all of these options are saved in metadata files on S3 or ADLS. When you replicate *from* S3 or ADLS to HDFS, you can set the options you want to preserve.

Parameter	Description
deletePolicy	Provide one of the following options: • KEEP_DELETED_FILES - Retains the destination files even when they no longer exist at the source. • DELETE_TO_TRASH - Moves files to the trash folder if the HDFS trash is enabled. (Not supported when replicating to S3 or ADLS.) • DELETE_PERMANENTLY - Uses the least amount of space; use with caution. Default is KEEP_DELETED_FILES.
alerts	Configure the following parameters as required: • onFailure - Provide true to generate alerts when the replication job fails. • onStart - Provide true to generate alerts when the replication job starts. • onSuccess - Provide true to generate alerts when the replication job completes successfully. • onAbort - Provide true to generate alerts when the replication job is aborted.
exclusionFilters	Provide one or more directory paths to exclude from replication.
frequencyInSec	Auto-populated after the policy runs successfully. Shows the time duration between two replication jobs in seconds.
targetDataset	Auto-populated after the policy runs successfully. Shows the target location where the replicated files are available on the target cluster.
cloudCredentials	Provide the cloud credentials.
sourceCluster	Shows the source cluster name.
targetCluster	Shows the target cluster name in the dataCProvideName$cluster name format. For example, "DC-US$My Destination 17".
startTime	Shows the start time of the replication job in the YYYY-MM-DDTH H:MM:SSZ format.
endTime	Shows the end time of the replication job in the YYYY-MM-DDTH H:MM:SSZ format.
distcpMaxMaps	Provide the maximum map slots to limit the number of map slots per mapper. Default is 20.
distcpMapBandwidth	Provide the maximum bandwidth to limit the bandwidth per mapper. Default is 100 MB.
queueName	Provide a YARN queue name, if necessary. Default queue name is Default.
tdeSameKey	Provide true if the source and destination are encrypted with the same TDE key.
description	Provide a description for the policy.
enableSnapshotBasedReplication	Provide true to enable snapshot-based replication.
cloudEncryptionAlgorithm	Provide the cloud encryption algorithm.
cloudEncryptionKey	Provide the cloud encryption key.
plugins	Provide the plugins to deploy on all the nodes in the cluster if you have multiple repositories configured in your environment.

Parameter	Description
cmPolicySubmitUser	Provide the following options: • userName - Provide the user name that you are using to run the policy. • sourceUser - Provide the source cluster username, if any.

Sample HDFS replication policy definition JSON file

The following snippet shows the contents of the HDFS replication policy definition JSON file.

```
{
 "name": "string",
 "type": "FS"|"HIVE",
 "sourceDataset": {
  "hdfsArguments": {
   "path": "string",
                      "mapReduceService": "string",
                      "logPath": "string",
   "replicationStrategy": "DYNAMIC"|"STATIC",
   "errorHandling": {
    "skipChecksumChecks": true|false,
    "skipListingChecksumChecks": true|false,
    "abortOnError": true|false,
    "abortOnSnapshotDiffFailures": true|false
   },
   "preserve": {
    "blockSize": true|false,
    "replicationCount": true|false
    "permissions": true|false,
    "extendedAttributes": true|false
   },
   "deletePolicy": "KEEP_DELETED_FILES"|"DELETE_TO_TRASH"|"DELETE_PERMANENTL
Y",
   "alerts": {
    "onFailure": true|false,
    "onStart": true|false,
    "onSuccess": true|false,
    "onAbort": true|false
   },
   "exclusionFilters": ["string", ...]
   },
   "hiveArguments": {
    "databasesAndTables": [
     {
      "database": "string",
      "tablesIncludeRegex": "string",
      "tablesExcludeRegex": "string",
     }
     ...
    ],
    "sentryPermissions": "INCLUDE"|"EXCLUDE",
    "skipUrlPermissions": true|false,
    "numThreads": integer
   }
  },
  "frequencyInSec": integer,
  "targetDataset": "string",
  "cloudCredentials": "string",
  "sourceCluster": "string",
  "targetCluster": "string",
  "startTime": "string",
  "endTime": "string",
```

```
  "distcpMaxMaps": integer,
  "distcpMapBandwidth": integer,
  "queueName": "string",
  "tdeSameKey": true|false,
  "description": "string",
  "enableSnapshotBasedReplication": true|false
  "cloudEncryptionAlgorithm": "string",
  "cloudEncryptionKey": "string",
  "plugins": ["string", ...],
  "hiveExternalTableBaseDirectory": "string",
  "cmPolicySubmitUser": {
   "userName": "string",
   "sourceUser": "string"
  }
 }
```

Important: When you edit the file, ensure that you remove the key-value pairs that are not required for the policy. For example, you can remove queueName if you do not want to configure it for the replication policy.

Managing HDFS replication policies using CDP CLI

You can use CDP CLI to perform various actions on a replication policy.

About this task

You can perform the following actions to manage an HDFS replication policy:

Procedure

- Suspend a running policy job:

```
cdp --profile [***PROFILE NAME***] replicationmanager suspend-policy --clu
ster-crn  [***SOURCE CLUSTER CRN***] --policy-name [***POLICY NAME***]
```

- Activate a suspended policy job:

```
cdp --profile [***PROFILE NAME***] replicationmanager activate-policy --cl
uster-crn [***SOURCE CLUSTER CRN***] --policy-name [***POLICY NAME***]
```

- Download the diagnostic bundle for the specified replication policy:
  a) Initiate the diagnostic bundle collection operation for the specified replication policy on the target Cloudera Manager.

```
cdp --profile [***PROFILE NAME***] replicationmanager collect-diag
nostic-bundle --cluster-crn [***TARGET CLUSTER CRN***] --policy-na
me [***POLICY NAME***]
```

  The following sample snippet shows the command output:

```
{
    "commandId": 58747,
    "name": "Replication Diagnostics Collection",
    "active": true,
    "startTime": "2022-11-07T12:27:25.872Z"
}
```

b) View the diagnostic bundle collection status. Optionally, you can use the following command to get the current status of any Cloudera Manager command.

```
cdp --profile [***PROFILE NAME***] replicationmanager get-command-stat
us --cluster-crn [***TARGET CLUSTER CRN***] --policy-name [***POLICY
 NAME***]
```

The command returns the following responses:

- The diagnostic bundle collection is **IN PROGRESS** on the Cloudera Manager server.
- The diagnostic bundle collection is complete, and is **DOWNLOADABLE WITH URL** using the URL specified in the resultDataUrl field in the command output.
- The diagnostic bundle collection is complete, and is **DOWNLOADABLE WITH CLI** using the download -diagnostic-bundle CDP CLI operation in Step 3.
- The diagnostic bundle collection has **FAILED** on the Cloudera Manager server.

Tip: The commandId in the output is used in Step 3 to download the bundle.

The following sample snippet shows the command output when the bundleStatus is **DOWNLOADABLE WITH CLI**:

```
{
    "commandId": 58741,
    "success": true,
    "active": false,
    "name": "Replication Diagnostics Collection",
    "resultDataUrl": "http://[***CM HOST***]:[***CM PORT***]/cmf/command
/58741/download",
    "resultMessage": "Replication diagnostics collection succeeded.",
    "bundleStatus": "DOWNLOADABLE WITH CLI",
    "bundleStatusMessage": "The bundle can be downloaded with the down
load-diagnostic-bundle operation."
}
```

c) Run the following command only if the bundleStatus shows **DOWNLOADABLE WITH CLI** in the Step 2 command output. The command output appears as a binary string in base64 encoded format on the screen.

```
cdp --profile [***PROFILE NAME***] replicationmanager download-dia
gnostic-bundle --cluster-crn [***TARGET CLUSTER CRN***] --command-i
d [***COMMAND ID***]
```

You can use any method to parse the response. Alternatively, you can also use one of the following methods to parse the response:

Method	Command
Save the diagnostic bundle in the specified file in JSON format and download to your machine	`cdp --profile [***PROFILE NAME***] replicationmanager download-diagnostic-bundle --cluster-crn [***TARGET CLUSTER CRN***] --command-id [***COMMAND ID***] > [***<FILE>.JSON***]`
Save the diagnostic bundle to the specified ZIP file and download to your machine.	`cdp --profile [***PROFILE NAME***]replicationmanager download-diagnostic-bundle --cluster-crn [***TARGET CLUSTER CRN***] --command-id [***COMMAND ID***] > [***<FILENAME>.JSON***] | j`

Method	Command
	`q -r '.bundleFile' | base64 -D > [***<FILENAME>.ZIP***]`
Save the diagnostic bundle as a ZIP file and extract it to the specified location on your machine automatically.	`cdp --profile [***PROFILE NAME***] replicationmanager download-diagnostic-bundle --cluster-crn [***TARGET CLUSTER CRN***] --command-id [***COMMAND ID***] > [***<FILENAME>.JSON***] | jq -r '.bundleFile' | base64 -D >| bsdtar -xcf -f [***LOCATION***]`

- Delete the replication policy:

```
cdp --profile [***PROFILE NAME***] replicationmanager delete-policy --cluster-crn [***SOURCE CLUSTER CRN***] --policy-name [***POLICY NAME***]
```

Creating Hive replication policy using CDP CLI

You can use CDP CLI to create an Hive replication policy. Only a non-machine user can run the "replicationmanager create-policy" CDP CLI command to create a replication policy.

Procedure

1. Log into Replication Manager CDP CLI setup:

```
cdp --profile [***PROFILE NAME***] replicationmanager
```

2. List the clusters to verify whether the required clusters are available:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-clusters
```

3. Verify whether the required services are running on the source cluster:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-cluster-service-statuses
```

4. Ensure that the cloud credentials are available:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-all-credentials
```

5. Perform the following steps to create a policy definition JSON file.
   a) Open a policy definition JSON file, or copy the output of the command to a JSON file to generate a policy definition JSON file.

   ```
   cdp --profile [***PROFILE NAME***] replicationmanager create-policy --generate-cli-skeleton
   ```

   ```
   cdp --profile hive1 replicationmanager create-policy --generate-cli-skeleton > rm_hive1.json
   ```

   b) Enter the required parameters. Remove the key-value pairs that are not required in the policy definition JSON file for the specific policy.
   c) Save the file.

6. Create the Hive replication policy.

```
cdp --profile [***PROFILE NAME***] replicationmanager create-policy --cli-
input-json [****POLICY DEFINITION JSON FILE PATH USING CAT***]
```

The `cat` command reads the data from the policy definition JSON file.

cdp --profile local-dev replicationmanager create-policy --cli-input-json “$(cat temp/rm_hive1.json)”

Results

Replication Manager creates the replication policy and initiates data replication.

What to do next

Verify whether the replication policy is running as expected on the Cloudera Replication Manager Replication Policies page, or run the following command:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-policies --cluste
r-crn [***CRN OF THE CLUSTER WHERE THE REPLICATION POLICIES ARE STORED***]
```

Related Information

Creating Hive replication policy using Replication Manager

Hive replication policy definition JSON file

The policy definition JSON file contains all the parameters required to create a Hive replication policy. When you edit the file to define a Hive replication policy, remove the parameters that are not required for the replication policy.

Parameters in Hive replication policy definition JSON file

The following table lists the parameters in the policy definition JSON file that are required for a Hive replication policy:

Parameter	Description
name	Provide the unique name for the policy.
type	Provide HIVE to create a Hive replication policy.
mapReduceService	Provide the MapReduce or YARN service for the replication policy to use.
logPath	Provide an alternate path for the logs, if required.
replicationStrategy	Provide one of the following options to determine whether the file replication tasks must be distributed among the mappers statically or dynamically: • STATIC - Static replication distributes file replication tasks among the mappers up front to achieve an uniform distribution based on the file sizes. • DYNAMIC - Dynamic replication distributes the file replication tasks in small sets to the mappers, and as each mapper completes its tasks, it dynamically acquires and processes the next unallocated set of tasks. Default is DYNAMIC.

Parameter	Description
skipChecksumChecks	Provide true to skip checksum checks. Default is true. Checksums are used to perform the following tasks: • To skip replication of files that have already been copied. When set to true, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination. • To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data.
skipListingChecksumChecks	Provide true to skip checksum check while comparing two files to determine whether they are the same or not. Otherwise, the file size and last modified time are used to determine if files are the same or not. Skipping the check improves performance during the mapper phase. **Note:** If you set skipChecksumChecks to false, the skipListingChecksumChecks is also set to false by default.
abortOnError	Provide true to stop the policy job when an error occurs. This ensures that the files copied up to that point remain on the destination, but no additional files are copied. Default is false.
abortOnSnapshtDiffFailures	Provide true to stop the replication job if a snapshot diff fails during replication.
preserve	Provide true to preserve the block size, replication count, permissions (including ACLs), and extended attributes (XAttrs) as they exist on the source file system. • blockSize • replicationCount • permissions • extendedAtrributes Provide false to use the settings as configured on the destination file system. By default, the source system settings are preserved.
deletePolicy	Provide one of the following options: • KEEP_DELETED_FILES - Retains the destination files even when they no longer exist at the source. • DELETE_TO_TRASH - Moves files to the trash folder if the HDFS trash is enabled. (Not supported when replicating to S3 or ADLS.) • DELETE_PERMANENTLY - Uses the least amount of space; use with caution. Default is KEEP_DELETED_FILES.
alert	Configure the following parameters as required: • onFailure - Provide true to generate alerts when the replication job fails. • onStart - Provide true to generate alerts when the replication job starts. • onSuccess - Provide true to generate alerts when the replication job completes successfully. • onAbort - Provide true to generate alerts when the replication job is aborted.

Parameter	Description
exclusionFilters	Provide one or more directory paths to exclude from replication.

databasesAndTables	Configure the parameter as required:
	• database - Provide one or more database names to include from replication.
	• tablesIncludeRegex - Provide one or more regular expression-based paths to tables to include in replication.
	For example, if you Provide
	`table1|table2|table3`
	, Replication Manager includes the specified tables for replication. If you Provide
	`DB :db_name`
	`Table : (?!table1|table2|table3).+`
	, Replication Manager includes all the tables in the 'db_name' database and excludes 'table1', 'table2', and 'table3' from replication.
	tablesExcludeRegex is a legacy option. You can provide one or more regular expression-based paths of tables to exclude in replication.

sentryPermissions	Provide INCLUDE to import both Hive object and URL permissions.
skipUrlPermissions	Provide true to import only the Hive object permissions.
numThreads	Provide the number of threads to use during replication.
frequencyInSec	Auto-populated after the policy runs successfully. Shows the time duration between two replication jobs in seconds.
targetDataset	Auto-populated after the policy runs successfully. Shows the target location where the replicated files are available on the target cluster.
cloudCredential	Provide the cloud credentials.
sourceCluster	Shows the source cluster name.
targetCluster	Shows the target cluster name in the dataCProvideName$clustername format. For example, "DC-US$My Destination 17".
startTime	Shows the start time of the replication job in the YYYY-MM-DDTH H:MM:SSZ format.
endTime	Shows the end time of the replication job in the YYYY-MM-DDTH H:MM:SSZ format.
distcpMaxMaps	Provide the maximum map slots to limit the number of map slots per mapper. Default is 20.
distcpMapBandwidth	Provide the maximum bandwidth to limit the bandwidth per mapper. Default is 100 MB.
queueName	Provide a YARN queue name, if necessary. Default queue name is Default.
tdeSameKey	Provide true if the source and destination are encrypted with the same TDE key.
description	Provide a description for the policy.
enableSnapshotBasedReplication	Provide true to enable snapshot-based replication.
cloudEncryptionAlgorithm	Provide the cloud encryption algorithm.

Parameter	Description
cloudEncryptionKey	Provide the cloud encryption key.
plugins	Provide the plugins to deploy on all the nodes in the cluster if you have multiple repositories configured in your environment.
hiveExternalTableBaseDirectory	Provide the Hive external table base directory path.
cmPolicySubmitUser	Provide the following options: • userName - Provide the user name that you are using to run the policy. • sourceUser - Provide the source cluster username, if any.

Sample Hive replication policy definition JSON file

The following snippet shows the contents of the Hive replication policy definition JSON file. While editing the file, ensure that you remove the key-value pairs that are not required for the Hive replication policy.

```
{
 "name": "string",
 "type": "HIVE",
 "sourceDataset": {
  "hdfsArguments": {
   "path": "string",
   "mapReduceService": "string",
                      "logPath": "string",
                      "replicationStrategy": "DYNAMIC"|"STATIC",
   "errorHandling": {
    "skipChecksumChecks": true|false,
    "skipListingChecksumChecks": true|false,
    "abortOnError": true|false,
    "abortOnSnapshotDiffFailures": true|false
   },
   "preserve": {
    "blockSize": true|false,
    "replicationCount": true|false
    "permissions": true|false,
    "extendedAttributes": true|false
   },
   "deletePolicy": "KEEP_DELETED_FILES"|"DELETE_TO_TRASH"|"DELETE_PERMANE
NTLY",
   "alert": {
    "onFailure": true|false,
    "onStart": true|false,
    "onSuccess": true|false,
    "onAbort": true|false
   },
   "exclusionFilters": ["string", ...]
   },
   "hiveArguments": {
    "databasesAndTables": [
     {
      "database": "string",
      "tablesIncludeRegex": "string",
      "tablesExcludeRegex": "string",
     }
     ...
    ],
    "sentryPermissions": "INCLUDE"|"EXCLUDE",
    "skipUrlPermissions": true|false,
    "numThreads": integer
   }
  },
```

```
  "frequencyInSec": integer,
  "targetDataset": "string",
  "cloudCredential": "string",
  "sourceCluster": "string",
  "targetCluster": "string",
  "startTime": "string",
  "endTime": "string",
  "distcpMaxMaps": integer,
  "distcpMapBandwidth": integer,
  "queueName": "string",
  "tdeSameKey": true|false,
  "description": "string",
  "enableSnapshotBasedReplication": true|false
  "cloudEncryptionAlgorithm": "string",
  "cloudEncryptionKey": "string",
  "plugins": ["string", ...],
  "hiveExternalTableBaseDirectory": "string",
  "cmPolicySubmitUser": {
   "userName": "string",
   "sourceUser": "string"
  }
 }
```

Important: When you edit the file, ensure that you remove the key-value pairs that are not required for the policy.

Managing Hive replication policies using CDP CLI

You can use CDP CLI to perform various actions on a replication policy. You can suspend a running Hive replication policy job and then activate it. You can also delete a replication policy.

About this task

You can perform the following actions to manage a Hive replication policy:

Procedure

- Suspend a running policy job:

```
cdp --profile [***PROFILE NAME***] replicationmanager suspend-policy --clu
ster-crn  [***SOURCE CLUSTER CRN***] --policy-name [***PROFILE NAME***]
```

- Activate a suspended policy job:

```
cdp --profile [***PROFILE NAME***] replicationmanager activate-policy --cl
uster-crn [***TARGET CLUSTER CRN***] --policy-name [***PROFILE NAME***]
```

- Download the diagnostic bundle for the specified replication policy:
  a) Initiate the diagnostic bundle collection operation for the specified replication policy on the target Cloudera Manager.

```
cdp --profile [***PROFILE NAME***] replicationmanager collect-diag
nostic-bundle --cluster-crn [***TARGET CLUSTER CRN***] --policy-na
me [***POLICY NAME***]
```

  The following sample snippet shows the command output:

```
{
    "commandId": 58747,
    "name": "Replication Diagnostics Collection",
```

```
    "active": true,
    "startTime": "2022-11-07T12:27:25.872Z"
}
```

b) View the diagnostic bundle collection status. Optionally, you can use the following command to get the current status of any Cloudera Manager command.

```
cdp --profile [***PROFILE NAME***] replicationmanager get-command-stat
us --cluster-crn [***TARGET CLUSTER CRN***] --policy-name [***POLICY
 NAME***]
```

The command returns the following responses:

- The diagnostic bundle collection is **IN PROGRESS** on the Cloudera Manager server.
- The diagnostic bundle collection is complete, and is **DOWNLOADABLE WITH URL** using the URL specified in the resultDataUrl field in the command output.
- The diagnostic bundle collection is complete, and is **DOWNLOADABLE WITH CLI** using the download -diagnostic-bundle CDP CLI operation in Step 3.
- The diagnostic bundle collection has **FAILED** on the Cloudera Manager server.

Tip: The commandId in the output is used in Step 3 to download the bundle.

The following sample snippet shows the command output when the bundleStatus is **DOWNLOADABLE WITH CLI**:

```
{
    "commandId": 58741,
    "success": true,
    "active": false,
    "name": "Replication Diagnostics Collection",
    "resultDataUrl": "http://[***CM HOST***]:[***CM PORT***]/cmf/command
/58741/download",
    "resultMessage": "Replication diagnostics collection succeeded.",
    "bundleStatus": "DOWNLOADABLE WITH CLI",
    "bundleStatusMessage": "The bundle can be downloaded with the down
load-diagnostic-bundle operation."
}
```

c) Run the following command only if the bundleStatus shows **DOWNLOADABLE WITH CLI** in the Step 2 command output. The command output appears as a binary string in base64 encoded format on the screen.

```
cdp --profile [***PROFILE NAME***] replicationmanager download-dia
gnostic-bundle --cluster-crn [***TARGET CLUSTER CRN***] --command-i
d [***COMMAND ID***]
```

You can use any method to parse the response. Alternatively, you can also use one of the following methods to parse the response:

Method	Command
Save the diagnostic bundle in the specified file in JSON format and download to your machine	`cdp --profile [***PROFILE NAME***] replicationmanager download-diagnostic-bundle --cluster-crn [***TARGET CLUSTER CRN***] --command-id [***COMMAND ID***] > [***<FILE>.JSON***]`
Save the diagnostic bundle to the specified ZIP file and download to your machine.	`cdp --profile [***PROFILE NAME***]replicationmanager downl`

Method	Command
	`oad-diagnostic-bundle --cluster-crn [***TARGET CLUSTER CRN***] --command-id [***COMMAND ID***] > [***<FILENAME>.JSON***] | jq -r '.bundleFile' | base64 -D > [***<FILENAME>.ZIP***]`
Save the diagnostic bundle as a ZIP file and extract it to the specified location on your machine automatically.	`cdp --profile [***PROFILE NAME***] replicationmanager download-diagnostic-bundle --cluster-crn [***TARGET CLUSTER CRN***] --command-id [***COMMAND ID***] > [***<FILENAME>.JSON***] | jq -r '.bundleFile' | base64 -D >| bsdtar -xcf -f [***LOCATION***]`

- Delete the replication policy:

```
cdp --profile [***PROFILE NAME***] replicationmanager delete-policy --cluster-crn [***SOURCE CLUSTER CRN***] --policy-name [***POLICY NAME***]
```

Creating HBase replication policy using CDP CLI

You can use CDP CLI commands to create an HBase replication policy.

Before you begin

Ensure that the prerequisites are complete before you create an HBase replication policy. For more information about the prerequisites, see Using HBase replication policies.

Procedure

1. Log into Replication Manager CDP CLI setup:

```
cdp --profile [***PROFILE NAME***] replicationmanager
```

2. List the clusters to verify whether the required clusters are available:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-clusters
```

3. Verify whether the required services are running on the required clusters:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-cluster-service-statuses
```

4. Ensure that the cloud credentials are available:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-all-credentials
```

5. Verify whether the source cluster and target cluster are paired:

```
cdp --profile [***PROFILE NAME***] replicationmanager verify-hbase-cluster
-pair
```

The command returns the following cluster pairing status:

pairingStatus	Description
PAIRED	Indicates that the clusters are paired and Replication Manager attempts to initiate the first-time setup between the clusters. The following status appear during the pairing process: • ERROR - Indicates that the policy creation process cannot be continued unless the error is fixed and the pairing is reset. • UNDER_SETUP - Indicates that you must wait for the process to complete before you continue to create the replication policy. • RESETTABLE - Indicates that the pairing process completed successfully and that there are no existing replication policies using this pairing. You can continue to create the HBase replication policy. • NOT_RESETTABLE - Indicates that the pairing process completed successfully and there are existing policies using this pairing. You can continue to create the HBase replication policy.
CAN_BE_FORCED	Indicates that one or both of the specified clusters are part of other pairings that are in ERROR or RESETTABLE state. You can set up the pairing between the clusters by setting the force* parameter to true. This action removes the existing pairings.
WRONG_PAIRING	Indicates that one or both of the specified clusters are part of other pairings that are in NOT_RESETTABLE state. In this scenario, you cannot continue to create HBase replication policies.
ON_HOLD	Indicates that one or both of the specified clusters are part of other pairings that are in UNDER_SETUP state. In this scenario, you can retry the pairing process after some time.
*The option is a technical preview feature and is not ready for production deployment. The components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the use of preview components, which should be used by customers at their own risk. For more information, contact your Cloudera account team.	

Tip: View the previously paired clusters:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-paired-
hbase-clusters
```

6. Create a policy definition JSON file.
   a) Open a policy definition JSON file, or copy the output of the following command to a JSON file to generate a policy definition JSON file:

```
cdp --profile [***PROFILE NAME***] replicationmanager create-hbase-pol
icy --generate-cli-skeleton
```

```
cdp --profile hbase1 replicationmanager create-hbase-policy --generate-
cli-skeleton > rm_hbase1.json
```

   b) Enter the required parameters. Remove the key-value pairs that are not required in the policy definition JSON file for the specific policy.
   c) Save the file.

7. Create the HBase replication policy:

```
cdp --profile [***PROFILE NAME***] replicationmanager create-hbase-policy
--cli-input-json [***POLICY DEFINITION JSON***]
```

Optionally, you can use the cat command to read the data from the policy definition JSON file. For example, cdp --profile local-dev replicationmanager create-policy --cli-input-json “$(cat temp/rm_hbase1.json)”

Important: If the source cluster is a classic cluster and the clusters have not been paired before, manual intervention is required for the initial setup.

Perform the following steps after you create the HBase replication policy to complete the first-time setup between the clusters:

a. View the policy status:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-policie
s | less
```

b. Restart the HBase service on the source Cloudera Manager when the policy status shows “WAITING _RESTART_ON_SRC” or WAITING_ON_SRC_RESTARTING_ON_DEST.

c. Ensure that the HBase service restart is complete.

d. Continue the first time setup process when the policy status shows WAITING_FOR_CONTINUE _SETUP_CALL:

```
cdp --profile [***PROFILE NAME***] replicationmanager continue
-hbase-setup --cluster-crn [***TARGET CLUSTER CRN***] --policy-
id [***POLICY ID***]
```

Results

Replication Manager creates the replication policy and initiates data replication.

What to do next

Verify whether the replication policy is running as expected on the Cloudera Replication Manager Replication Policies page, or run the following command:

```
cdp --profile [***PROFILE NAME***] replicationmanager list-policies --cluste
r-crn [***CRN OF THE CLUSTER WHERE THE REPLICATION POLICIES ARE STORED***]
```

HBase replication policy definition JSON file

The HBase replication policy definition JSON file contains all the parameters required to create an HBase replication policy.

Parameters in HBase replication policy definition JSON file

The following table lists the parameters in the policy definition JSON file that are required to create an HBase replication policy:

Parameter	Description	Required?
clusterCrn	Provide the target cluster CRN. Replication Manager saves the replication policy in the specified cluster CRN.	Required
policyName	Provide a unique name for the replication policy.	Required

Parameter	Description	Required?
policyDefinition	Provide the policy definition parameters as required.	Required
hbasePolicyArguments	Provide the HBase replication parameters.	Required
tables	Provide the tables to be replicated where the key must be in the "namespace:tablename" format. **Note:** If the tables parameter has no value, Replication Manager replicates the entire database.	Optional
credentialLocation*	Provide SAFETY_VALVE if the credentials are available in the advanced configuration snippets (safety valves) on the source cluster, or provide EXTERNAL_ACCOUNT if the credentials are to be defined in the cloudCre dential property. Default is EXTERNAL_ACCOUNT.	Optional
cloudCredential	Provide the cloud credentials to use to replicate the initial snapshot if the credentials are defined in an external account. **Note:** The cloudCredential parameter is mandatory if init ialSnapshot=true and the cloud credentials are not available in the advanced configuration snippets.	Optional
validateReplicationSetup*	Provide true to validate the replication setup after policy creation. Otherwise, provide false.	Optional
forceSetup*	Provide true to force the first-time setup when one of the clusters is already paired with another cluster. Forced setup is only possible if there are no HBase replication policies between the pair of clusters or if the other cluster in the pair is currently unreachable. During the force setup, Replication Manager clears the existing pairing for the selected source or target cluster and initiates the first-time setup with the chosen new source or destination cluster.	Optional
databaseArguments	Provide one of the following values for the replicationStrategy parameter: • ALL_TABLES • TABLES_WITH_REPLICATION_SCOP E_SET For more information about replication scope, see Creating HBase replication policy.	Optional
sourceCluster	Provide the name of the source cluster in the "dataCenterName$cluster name" format. For example, "DC-Europe$My Source 42”	Required
targetCluster	Provide the name of the target cluster in the "dataCenterName$cluster_name" format. For example, "DC-US$My Destination 17"	Required
initialSnapshot	Provide true to replicate the existing data in the table. When you provide false, Replication Manager replicates only the data generated after the policy creation.	Required

Parameter	Description	Required?
exportSnapshotUser	Enter the custom username. Replication Manager uses the specified username to export the initial snapshots to the target. **Note:** Use this option only if initialSnapshot=true. You must map the Kerberos username to an AWS role if you use an IDBroker topology based credential. For more information about user mapping, see Creating HBase replication policy.	Required
description	Provide a brief description of the replication policy.	Optional
machineUser	The credentials of the machine user that Replication Manager uses to run HBase replication policies. Provide the following parameters for the machine user: • user • password • createUser - Provide true to create a new machine user. If you provide false, ensure that the username you provide exists in the Cloudera User Management System (UMS), otherwise an error message appears. If you do not provide the machine user details, an HBase replication machine user is created automatically with an auto-generated username.	Optional
queueName	Provide a YARN queue name to use for the initial snapshot operation. Default is default.	Optional
distcpMaxMaps	Provide the maximum map jobs to use for initial snapshot operation.	Optional
distcpMapBandwidth*	Adjust the setting so that each map task is throttled to consume only the specified bandwidth. Default is 100 MB.	Optional
sourceRestartType	Provide RESTART or ROLLING_RESTART to restart the HBase service on the source cluster. **Note:** The sourceRestartType parameter is accepted only when the source cluster is not a classic cluster and the destination cluster has the restart-type support. Default is ROLLING_RESTART. **Tip:** During rolling restart, one node is restarted at a time and this continues until all the nodes in the cluster are restarted. This type of restart ensures that there is no disruption of service. During full restart, all the nodes are shut down at once and restarted simultaneously.	Optional

Parameter	Description	Required?
targetRestartType	Provide RESTART or ROLLING_RESTART to restart the HBase service on the destination cluster. **Note:** The targetRestartType parameter is accepted only if the cluster has restart-type support. Default is ROLLING_RESTART.	Optional
*The option is a technical preview feature and is not ready for production deployment. The components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the use of preview components, which should be used by customers at their own risk. For more information, contact your Cloudera account team.		

Sample HBase replication policy definition JSON file

The following snippet shows the contents of an HBase replication policy definition JSON file:

```
{
    "clusterCrn": "string",
    "policyName": "string",
    "policyDefinition": {
        "hbasePolicyArguments": {
            "tables": [
                "string"
            ],
            "credentialLocation": "EXTERNAL_ACCOUNT"|"SAFETY_VALVE", (tech
nical preview)
            "cloudCredential": "string",
            "validateReplicationSetup": true|false, (technical preview only)
            "forceSetup": true|false, (technical preview only)
            "databaseArguments": {
                "replicationStrategy": "ALL_TABLES"|"TABLES_WITH_REPLICAT
ION_SCOPE_SET"
            }
    },
            "sourceCluster": "string",
            "targetCluster": "string",
            "initialSnapshot": true|false,
            "exportSnapshotUser": "string",
            "description": "string",
            "machineUser": {
                "user": "string",
                "password": "string",
                "createUser": true|false
            },
            "queueName": "string",
            "distcpMaxMaps": 0,
            "distcpMapBandwidth": 0, (technical preview only)
            "sourceRestartType": "RESTART"|"ROLLING_RESTART",
            "targetRestartType": "RESTART"|"ROLLING_RESTART"
    }
}
```

Important: When you edit the file, ensure that you remove the key-value pairs that are not required for the policy. For example, you can remove queueName if you do not want to configure it for the replication policy.

Managing HBase replication policies using CDP CLI

You can use CDP CLI commands to manage HBase replication policies.

Procedure

- Pause an active HBase replication policy:

```
cdp --profile [***PROFILE NAME***] replicationmanager suspend-hbase-policy
 --cluster-crn [***TARGET CLUSTER CRN***] --policy-id [***POLICY ID***]
```

- Change the name and description, and delete one or more tables in the HBase replication policy:

```
cdp --profile [***PROFILE NAME***] replicationmanager update-hbase-pol
icy --cluster-crn [***TARGET CLUSTER CRN***] --policy-id [***POLICY
 ID***] --update-hbase-policy-definition name=[***STRING***],descrip
tion=[***STRING***],tables=[***STRING,STRING...***]
```

- Delete an HBase replication policy:

```
cdp --profile [***PROFILE NAME***] replicationmanager delete-hbase-policy
--cluster-crn [***TARGET CLUSTER CRN***] --policy-id [***POLICY ID***] [--
force] [--no-force]
```

  Default is --no-force.

  When the source cluster is not reachable, the error HBase peer not found appears. In this instance, you can append the --force option to delete the HBase replication policy.

Cloudera Replication Manager

Replication Manager Release Notes

Date published: 2019-11-15
Date modified: 2024-07-08

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What's new 4
November 2024 4
July 2024 4
June 2024 5

Older releases 5
April 2024 5
January 2024 6
August 2023 6
June 2023 7
Mar 07, 2023 7
Jan 11, 2023 8
May 16, 2022 9
May 12, 2022 10
Mar 1, 2022 11
Feb 24, 2022 11
Sept 9, 2021 12
April 30, 2021 12
March 25, 2021 12
June 19, 2020 12
November 15, 2019 13

Known Issues 13

Fixed Issues 15

What's new

This section lists major features and updates for Cloudera Replication Manager service.

Attention: Support of Replication Manager with Hortonworks Data Platform (HDP) clusters is Technical Preview and not available by default. For more information, contact your Cloudera account team. If you have questions regarding this service, contact support by logging a case on our Cloudera Support Portal.

November 2024

This release of the Replication Manager service introduces the following new features.

CDP CLI commands for HBase replication policies

You can create, activate, or delete HBase replication policies using the following CDP CLI commands:

- activate-hbase-policy
- create-hbase-policy
- continue-hbase-setup
- delete-hbase-policy
- get-cluster-config
- list-paired-hbase-clusters
- suspend-hbase-policy
- update-hbase-policy
- verify-hbase-cluster-pair

For more information, see Creating HBase replication policy.

July 2024

This release of the Replication Manager service introduces the following new features.

Fine-grained permission to access CDP Public Cloud Replication Manager

You can choose to restrict user access to view and use CDP Public Cloud Replication Manager with RBAC entitlement. To enable or disable the role-based access control (RBAC) entitlement, contact your Cloudera account team.

For more information, see Fine-grained permission to access CDP Public Cloud Replication Manager.

Replication Manager support for AWS temporary credentials (technical preview)

You can use temporary AWS credentials, through the IDBroker service, to replicate HDFS data, Hive external tables, and HBase data from 7.1.9 SP1 Kerberized CDP Private Cloud Base clusters using Cloudera Manager 7.11.3 CHF7 or higher versions to CDP Public Cloud S3 clusters. You can also use the temporary AWS credentials to replicate HDFS data from S3 buckets to 7.1.9 SP1 Kerberized CDP Private Cloud Base clusters or higher using Cloudera Manager 7.11.3 CHF7 or higher versions.

This is a technical preview feature. It is not recommended for production deployments. Cloudera recommends that you try this feature in development or test environments. To enable this feature, contact your Cloudera account team.

For more information, see Add IDBroker to use temporary AWS session credentials

HBase replication policy enhancements

Specify custom username

Starting from CDP Public Cloud 7.2.18.200, you can specify a custom username in the Select Source Export snapshot user field during the HBase replication policy creation process. The option appears after you choose the Select Source Perform Initial Snapshot option. Replication Manager uses the specified username on the source cluster to export the initial snapshot to the target.

For more information, see Creating HBase replication policies.

June 2024

This release of the Replication Manager service introduces the following new features.

HBase replication policy enhancements

Replicate data through an network load balancer (NLB)

During the HBase replication policy creation process, you can choose to specify an NLB if the source CDH 5.16.2 cluster uses the NLB to communicate with ZooKeeper and RegionServers of the destination Cloudera Manager of COD clusters.

Specify the NLB details after you enable the Select Destination Replicate via a Network Load Balancer option during the HBase replication policy creation process.

For more information, see Support matrix for CDP Replication Manager and Creating HBase replication policies.

Specify maximum number of tables to process in parallel

You can specify the maximum number of tables to process in parallel during the initial snapshot export and import step during an HBase replication policy run. You specify the number of tables using the Initial Snapshot Settings Maximum parallel snapshots option during the HBase replication policy creation process. If you do not enter any value, Replication Manager chooses an appropriate value, depending on the resources in the source and target cluster, to optimize the performance.

For more information, see Support matrix for CDP Replication Manager and Creating HBase replication policies.

Older releases

Overview of new features, enhancements, and changed behavior introduced in earlier releases of Replication Manager.

April 2024

CDP version 7.2.18 introduces the following new features in the Replication Manager service.

Register GCP credentials

You can add the GCP credentials on the CDP Public Cloud Replication Manager Cloud Credentials page to use in Replication Manager.

For more information, see Registering GCS credentials to use in Replication Manager.

HBase replication policy enhancements

HBase replication policies support the following enhancements:

- You can create multiple HBase replication policies between multiple clusters to replicate HBase data.

  For more information, see Replicate HBase data simultaneously between multiple clusters.
- You can choose to Replicate all user tables or Replicate only tables where replication is already enabled after you choose the Select Source Replicate Database option during the HBase replication policy creation process.

  The Replicate only tables where replication is already enabled option is supported only if the target cluster uses Cloudera Manager version 7.12.0.0 and higher, 7.11.0-h3 and higher, or 7.9.0-h7 and higher.

  For more information, see Methods to replicate HBase data and Creating HBase replication policies.
- You can enter a YARN Queue Name to submit the replication job during the HBase replication policy creation or choose to retain the "default" value.

  For more information, see Creating HBase replication policies.
- You can click Actions on the **Replication Policies** page to Collect diagnostic bundle for the required HBase replication policy.

  For more information, see Monitor HBase replication policy job details.

For more information about feature support in CDP Public Cloud Replication Manager, see Supported features.

January 2024

This release of the Replication Manager service introduces the following new features.

Replicate HBase tables in a database depending on the replication scope

During the HBase replication policy creation process, after you select the Select Source Replicate Database , you can choose the Select Source Replicate all user tables option to replicate all the HBase tables in the database, or choose the Select Source Replicate only tables where replication is already enabled option to replicate the tables for which the replication scope is already set to 1. This provides a choice to replicate the required tables in a database.

The Replicate only tables where replication is already enabled option is supported only if the target cluster CDP version is CDP version 7.2.16.500 using Cloudera Manager 7.9.0-h7 or higher versions, CDP version 7.2.17.300 using Cloudera Manager 7.11.0-h3 or higher versions, or CDP version 7.2.18 using Cloudera Manager 7.12.0.0.

For more information, see Methods to replicate HBase data and Creating HBase replication policies.

Replicate HBase data between multiple clusters simultaneously

Starting from CDP Public Cloud version 7.2.16.500, 7.2.17.200, and 7.2.18 you can create multiple HBase replication policies between multiple clusters to replicate HBase data.

For more information, see Replicate HBase data simultaneously between multiple clusters.

August 2023

This release of the Replication Manager service introduces the following new features.

Replicate HBase data in all the tables in a database

You can replicate HBase data (existing tables and future tables in a database) using the Select Source Replicate Databases option during the HBase replication policy creation process. You can use this option only if the minimum target Cloudera Manager version is 7.11.0 and the minimum source cluster versions are CDH 5.16.2 (after you upgrade the source cluster Cloudera Manager), CDP Private Cloud Base 7.1.6, or COD 7.2.17.

For more information, see HBase replication policy and Creating HBase replication policies.

Provide a unique replication policy name

You must provide a unique name to the replication policy during the replication policy creation process if the Cloudera Manager API version is 51 or higher.

For other Cloudera Manager API versions, you can continue to use an existing replication policy with an empty name. However, if you edit the replication policy and provide a name for the replication policy, ensure that the name conforms to the validation rules.

For more information, see Managing HDFS replication policy, Managing Hive replication policy, or Managing HBase replication policy.

Load replication policies and their job history on the Replication Policies page

You can choose one of the following options to **Load policies faster by delaying to load their job history**:

- Delay loading job history when history is too long
- Always load job history
- Never load job history

By default, the replication policies are loaded only partially on the **Replication Policies** page, therefore the page might display incomplete statistics about a job status and replication policies with failed jobs might take a longer time to load. You can change the behavior depending on your requirements.

For more information, see Replication Policies page

June 2023

This release of the Replication Manager service introduces the following new features.

HBase replication policy enhancements

You can view the HBase RegionServer metrics for a specific replication peer in Replication Manager. You can use these metrics to monitor HBase replication jobs and to find and diagnose issues with the HBase replication peer.

For more information, see Graphical representation of HBase RegionServer replication peer metrics

CDP CLI commands for Replication Manager

You can collect and download the diagnostic bundle, add ABFS and AWS cloud credentials to use in Replication Manager and delete credentials using CDP CLI commands.

For more information, see Adding cloud credentials in Replication Manager using CDP CLI.

Mar 07, 2023

This release of the Replication Manager service introduces the following new features.

HBase replication policy enhancements

The following enhancements are available for HBase replication policies:

- During HBase replication policy creation, you can choose one of the following cloud credential options:
  - Credentials are available in source cluster HDFS service configuration setting- You can choose this option when you want to use a Google Cloud account. You can use this option for S3 and ADLS accounts as well.

    To add the advanced configuration settings for Google Cloud, Amazon S3, and ADLS accounts, perform the following steps:

    1. Go to the source Cloudera Manager Clusters *HDFS SERVICE* Configuration tab.
    2. Locate the HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml property.
    3. Add the following key-value pairs to register a Google account to use in Replication Manager:
       - fs.gs.impl=com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystem
       - fs.gs.project.id=*[***ENTER THE PROJECT ID***]*
       - fs.gs.system.bucket=*[***ENTER THE BUCKET NAME***]*
       - fs.gs.working.dir=/
       - fs.gs.auth.service.account.enable=true
       - fs.gs.auth.service.account.email=*[***ENTER THE SERVICE PRINCIPAL EMAIL ID***]*
       - fs.gs.auth.service.account.keyfile=/*[***ENTER THE LOCAL PATH OF THE P12 FILE***]*

       You can add these key-value pairs to register a Google account to use in Replication Manager from CDP Public Cloud 7.2.16.1 and higher versions.
    4. Add the following key-value pairs to register an S3 account to use in Replication Manager:
       - fs.s3a.access.key=*[***ENTER THE SESSION ACCESS KEY***]*
       - fs.s3a.secret.key=*[***ENTER THE SESSION SECRET KEY***]*
    5. Add the following key-value pairs to register an ADLS account to use in Replication Manager:
       - fs.azure.account.oauth2.client.id=*[***ENTER THE ABFS STORAGE CLIENT ID***]*
       - fs.azure.account.oauth2.client.secret=*[***ENTER THE ABFS STORAGE CLIENT SECRET KEY***]*
    6. Save and restart the HDFS service for the changes to take effect.
  - Credentials from External Account - You can choose this option for S3 and ADLS storage options.

  For more information, see Creating an HBase replication policy.
- You can Force Delete an HBase replication policy after Replication Manager fails to delete the replication policy (using the Actions Delete option on the **Replication Policies** page for the replication policy). For more information, see Managing HBase replication policy.

Jan 11, 2023

This release of the Replication Manager service introduces the following new features.

HBase replication policy enhancements

The following enhancements are available for HBase replication policies:

- HBase data replication using HBase replication policies is supported when SFT setting is automatically set on SFT-enabled clusters.
- You must choose the Select Destination I want to force the setup of this HBase replication policy option in the HBase policy creation wizard to acknowledge that the first-time setup between the selected source and destination clusters should be initiated after the existing pairing of the source and/or target cluster gets cleared.

  For more information, see Creating HBase replication policy.

- You can also use the following options to manage an HBase replication policy:
  - The Replication Policies HBase replication policy Actions Suspend action suspends all the active HBase replication policies between the source and target clusters selected in the replication policy.
  - The Replication Policies HBase replication policy Actions Activate action resumes data replication for all the HBase replication policies between the source and target clusters selected in the replication policy.
  - The Replication Policies HBase replication policy Actions Retry Create action retries the first-time setup between the source and target clusters selected in the replication policy. This option is available only if the first-time setup configuration fails.
  - The Replication Policies HBase replication policy Actions Retry Failed Snapshots action reruns the failed initial snapshots (and only the failed ones) in the replication policy if the Replication Manager failed to replicate the existing data of some tables.
  - The Replication Policies HBase replication policy Actions View Command Details action opens the latest replication policy job and shows the last 15 steps of the log for the policy job run. The steps and substeps appear in a tree view. The failed steps are expanded by default.

  For more information, see Managing HBase replication policy.
- You can now monitor the first-time setup configuration steps and its progress on the Replication Policies HBase replication policy Job History tab.

Hive replication policy enhancements

The following enhancements are available for Hive replication policies:

- You can use Hive replication policies to replicate Hive external tables and metadata from CDH 5.10 and higher clusters and CDP Private Cloud Base 7.1.1 and higher clusters to Data Hubs using Cloudera Manager 7.9.0 or higher.

  For more information, see Creating Hive replication policy.
- The Replication Policies Hive replication policy Actions View Command Details action opens the latest replication policy job and shows the last 15 steps of the log for the policy job run. The steps and substeps appear in a tree view. The failed steps are expanded by default.

  For more information, see Managing Hive replication policy.

May 16, 2022

This release of the Replication Manager service introduces the following new features.

Updates to HBase replication policy

- If you are using Cloudera Manager version 7.6.0 or higher, you can use the following options while creating an HBase replication policy:
  - Rolling HBase Service Restart on Source - This option appears if you select COD or Data Hub as the source cluster. Select this option to enable automatic rolling restart of HBase service on the source cluster after the HBase replication policy first-time setup steps are complete. Otherwise, Cloudera Manager performs a full restart of the service.
  - Rolling HBase Service Restart on Destination - Select this option to enable automatic rolling restart of HBase service on the target cluster as a rolling restart after the HBase replication policy first-time setup steps are complete. Otherwise, Cloudera Manager performs a full restart of the service.
  - Validate Policy - Select to notify Replication Manager to verify the policy details after the policy creation is complete.

  For more information, see Creating HBase replication policy.

- A warning message appears when you choose a cluster that is part of an existing cluster pairing. If there are HBase replication policies for the existing pairing, policy creation is not allowed to continue. If there are no policies, the existing pairing is removed and policy creation continues.

  For more information, see Using HBase replication policy.

Support for Replication Manager in ap-1 and eu-1 regional Control Planes

Replication Manager is now supported in the ap-1 (Australia) and eu-1 (Germany) regional Control Planes. For the list of all supported services in the supported Control Plane regions, see CDP Control Plane regions.

May 12, 2022

This release of the Replication Manager service introduces the following new features.

Migrating HDFS data from cloud storage to CDH clusters

After you register a CDH cluster as a classic cluster in Management Console, and register the cloud credentials (AWS S3 and Azure) in Replication Manager, you can migrate HDFS data from the cloud storage to the registered classic cluster (CDH cluster) using HDFS replication policies.

For more information about CDP CLI for HDFS and Hive replication policies, see Creating HDFS replication policy.

Tracking and monitoring the performance of HDFS and Hive replication policies

You can monitor and track the performance of the following replication policies:

- HDFS replication policies - Download the CSV reports in the HDFS Replication Report field on the Replication Policies *[***HDFS REPLICATION POLICY***]* Job History panel.
- Hive replication policies - Download the CSV reports in the HDFS Replication Report field and Hive Replication Report field on the Replication Policies *[***HIVE REPLICATION POLICY***]* Job History panel.

For more information, see Managing HDFS replication policy and Managing Hive replication policy.

Modifying HDFS and Hive replication policy options

After you create an HDFS replication policy or Hive replication policy, you can change the policy options as required to meet any changing requirements. Optionally, on the **Replication Policies** page, you can expand an HDFS or Hive replication policy to edit the policy description, frequency (start time cannot be modified if the policy has already started) to run the policy, YARN queue name to submit the replication job, maximum bandwidth for each map task, and maximum map slots or tasks per replication job. To optimize the replication policy performance, you can configure the queue name, maximum bandwidth, and maximum map slots as necessary.

For more information, see Managing HDFS replication policy and Managing Hive replication policy.

Restart HBase service when you use on-premises cluster as source cluster in HBase replication policy

After you create an HBase replication policy, you must restart the HBase service in the on-premises source cluster when the policy status on the **Replication Policies** page shows Manual restart (src) / restarting (dest) or Manual HBase restart needed on source. After the service restart is complete, the setup continues automatically for the replication policy.

However, if the source cluster Cloudera Manager version is 7.6.0 or lower and you are using an on-premises cluster as the source cluster, you must perform the following steps to complete the HBase replication policy setup:

1. Restart the HBase service on the on-premises source cluster when the policy status on the **Replication Policies** page shows **Waiting for 'Continue Setup' action call**.
2. Click Continue setup for the policy on the **Replication Policies** page after the service restart is complete. This action informs Replication Manager to continue the replication policy setup.

For more information, see Managing HBase replication policy.

Mar 1, 2022

This release of the Replication Manager service introduces the following new features.

Using CDP CLI for HDFS and Hive replication policies

The CDP CLI commands for Replication Manager are under the replicationmanager CDP CLI option.

You can create, suspend, activate, or delete HDFS and Hive replication policies using create-policy, suspend-policy, activate-policy, and delete-policy CDP CLI commands. You can also list the clusters, replication policies, service statuses, credentials, and get the credentials for a specific cluster using the list-clusters, list-policies, list-cluster-service-statuses, list-all-credentials, and get-credentials CDP CLI commands.

For more information about CDP CLI for HDFS and Hive replication policies, see CDP CLI for Replication Manager.

Feb 24, 2022

This release of the Replication Manager service introduces the following new features.

Creating HDFS and HBase snapshot policies (Tech preview)

A snapshot is a set of metadata information, a point-in-time backup of HDFS files and HBase tables. You can create snapshot policies for HDFS directories and HBase tables in registered classic clusters and SDX Data Lake clusters to take snapshots at regular intervals. Before you create an HDFS snapshot policy for an HDFS directory, you must enable snapshots for the directory in Cloudera Manager.

After a snapshot policy takes a snapshot of the HDFS directory or HBase table, you can perform the following tasks:

- Restore the snapshot to the original directory or table using the Restore Snapshot option.
- Restore a directory or table to a different directory or table using the Restore Snapshot as option.

For more information about snapshot policies, see Snapshot policies in Replication Manager.

Important: Creating an HDFS or HBase snapshot policy is a technical preview feature. Access to preview features is provided upon request to customers for trial and evaluation. The components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the use of preview components, which should be used by customers at their own risk. Contact your Cloudera account team to have this preview feature enabled in your CDP account.

Override default storage location for replicated Hive external tables in the target cluster

Administrators can override the default storage location for replicated Hive external tables in the target cluster when they create a Hive replication policy.

Before you add another path to override the default storage location, ensure that the following steps are complete in the Ranger UI:

1. Alter the ranger policy Default: Hive warehouse locations in cm_s3 service to allow the Hive service to access the updated locations of S3 bucket path.
2. Manually update the Ranger and Sentry permissions.

For more information about editing the Hive External Table Base Directory option in Hive replication policies, see Creating Hive replication policy.

Cannot suspend HBase replication policies

If you create an HBase replication policy, you can no longer suspend the policy. However, you can resume any existing suspended HBase replication policy.

Generate and download diagnostic bundles for replication policies

You can generate and download diagnostic bundles for HDFS and Hive replication policies. You can use the bundle to troubleshoot failed replication jobs or to view replication-specific diagnostic data for a replication policy.

For more information about CDP CLI for HDFS and Hive replication policies, see Managing HDFS replication policies and Managing Hive replication policies.

Sept 9, 2021

This release of the Replication Manager service introduces the following new features.

Enhancements to HBase replication policies

The following enhancements are available for an HBase replication policy:

- You do not need to create a schema similar to the source cluster on the destination cluster.
- Replication Manager performs the first-time setup configuration steps which includes HBase service restart on both the clusters automatically.
- Optionally, you can also create or use an existing HBase replication machine user during policy creation and then validate the existing username with the UMS. The username and password is automatically synchronized to the destination cluster's environment (and to the source's as well if the source is Cloudera Operational Database (COD)).

April 30, 2021

This release of the Replication Manager service introduces the following new features.

Registering CDP Private Cloud Base clusters in CDP Public Cloud

You can register the CDP Private Cloud Base as a classic cluster in CDP. After registration, you can replicate the data in HDFS and Hive external tables in the classic cluster to CDP Public Cloud.

March 25, 2021

This release of the Replication Manager service introduces the following new features.

Replication Manager introduces HBase replication policy which supports the following use cases:

- From CDP Private Cloud Base cluster to Data Hub cluster.
- From CDH cluster to Data Hub cluster.
- From CDH cluster to COD cluster.
- From COD cluster to COD cluster.

Important: HBase replication policy is a technical preview feature. If you require access, contact your Cloudera account team.

June 19, 2020

This release of the Replication Manager service introduces the following new features.

Replication Manager is a service for copying and migrating data between environments within the enterprise data cloud. It is a simple, easy-to-use, and feature-rich data movement capability to move existing data and metadata to the cloud to fuel new workloads.

Replication Manager introduces the following new features:

- Availability of Azure Blob Filesystem (ABFS) cloud storage
- Ability to use Cloudera Data Platform Data Center clusters

November 15, 2019

This release of the Replication Manager service introduces the following new features.

Replication Manager is a service for copying and migrating data between environments within the enterprise data cloud. It is a simple, easy-to-use, and feature-rich data movement capability to move existing data and metadata to the cloud to fuel new workloads.

Replication Manager introduces the following new features:

- Full support for CDH On-Premise cluster (Classic Cluster)
- AutoTLS enablement
- HDFS-less compute clusters
- Table Level Replication

Known Issues

This section lists known issues that you might run into while using the Replication Manager service.

DOCS-13504

When you create an HBase replication policy between a source cluster using Cloudera Manager version is 7.6.0-patch5366 or higher and a target cluster using Cloudera Manager version is 7.6.0 or lower, the first-time setup is not initiated and the following misleading message appears during policy creation:

Skipping Replication Setup because it has already been done.

To resolve this issue, upgrade the target cluster to Cloudera Manager version to 7.6.0-patch5366 or higher.

DMX-518

Hive replication policy fails when a table is dropped in the source database (export tables) when the replication job is running.

DMX-519

If snapshots on the /user/hive/warehouse directory is not enabled, Hive replication policy fails when inserts are done on a source table during replication job run.

DMX-521

While running Hive replication policy, if you drop a table, it is not dropped on the target cluster. The data still remains and 'show tables' displays the dropped table after successful replication instances.

DMX-1455

An HBase replication policy from Cloudera Operational Database COD to COD cluster fails if you select the Perform Initial Snapshot option in the Create Replication Policy wizard, and the source and destination COD clusters use different AWS accounts.

DMX-2923

When you create an HBase replication policy with the Perform Initial Snapshot option in a bi-directional replication setup from the disaster recovery cluster to the production cluster, the original table gets dropped and it is recreated with the content in the disaster recovery cluster. Therefore, the original content of the table is lost.

When you create the HBase replication policy in a bi-directional replication setup from the disaster recovery cluster to the production cluster, do not select the Perform Initial Snapshot option.

OPSAPS-66327

The HBase peer can be created in the disabled state in CDH 6.0 or higher clusters. However, HBase peer cannot be created in disabled state in CDH 5 clusters and the peer remains enabled throughout the replication setup process. As a workaround, you can create a ‘test’ HBase replication policy with a small table in CDH 5 clusters. This creates the HBase peer.

OPSAPS-69210

HBase replication policies do not support source or destination CDP Public Cloud clusters created using the Micro Duty template.

OPSAPS-69782

HBase replication using HBase replication policies between two Data Hubs/COD clusters fail if all the following conditions are true:

- The destination Data Hub/COD cluster’s Cloudera Manager version is 7.9.0-h7 through 7.9.0-h9 or 7.11.0-h2 through 7.11.0-h4, or 7.12.0.0.
- The source Data Hub/COD cluster's Cloudera Manager major version is higher than the destination cluster's Cloudera Manager major version.
- The Initial Snapshot option is chosen during the HBase replication policy creation process and/or the source cluster is already participating in another HBase replication setup as a source or destination with a third cluster.

Upgrade the destination cluster's Cloudera Manager to a major version that is equal to or greater than the source cluster's Cloudera Manager major version.

OPSAPS-62230

HBase replication command steps for an HBase replication policy might fail if you have configured an automatic delete of /tmp folder content on the destination cluster’s Cloudera Manager server node.

OPSAPS-70282

You cannot create HBase replication policies if the target Cloudera Manager Clusters HBase service Configuration HBase ZooKeeper Secure Client Enabled property is enabled.

Disable the HBase ZooKeeper Secure Client Enabled property, and then create HBase replication policies.

OPSAPS-71317, OPSAPS-71318

When you delete an HBase replication policy, its peer is not removed if a table name in the replication policy contains a period. This issue is observed when the source Cloudera Manager is 7.4.4, 7.6.1, 7.6.7, 7.7.1, 7.7.3 CHFx, or CDP Public Cloud 7.2.16 using source Cloudera Manager 7.9.2, or CDP Public Cloud 7.2.17 using source Cloudera Manager 7.11.0, and unsupported target Cloudera Manager versions.

To resolve the issue:

- Upgrade to a supported target Cloudera Manager.
- Upgrade the source Cloudera Manager to 7.11.3 CHFx or higher.
- Remove the period from the table name on the source cluster.
- Perform the following steps to remove the peer associated with the deleted replication policy manually:
  1. Confirm that the HBase peer in the source HBase cluster does not contain any tables. If there are any tables, this indicates that there might be other HBase replication policies between these clusters. In such instances, you can contact your Cloudera account team.
  2. Manually remove the peer using the remove_peer *[***PEER ID***]* HBase command.
  3. Create an HBase replication policy in Replication Manager.

DMX-3888

The existing Hive replication policies do not appear on the **Replication Policies** page after you upgrade the target cluster from CDP Public Cloud 7.2.16 using Cloudera Manager 7.9.0 to CDP Public Cloud 7.2.17 using Cloudera Manager 7.11.0-h2. This issue appears if the existing replication policies were created using external APIs and the API payload included the rangerImport Properties and sentryExportProperties attributes.

Before you upgrade the target cluster, verify whether the HIVE_REPLICATION_SENTRY_EXPORT_PROPERTIES feature flag is enabled. If the feature flag is not enabled, you must remove the rangerImportProperties and sentryExportProperties attributes from the existing Hive replication policies, and then upgrade the target cluster.

Related Information

Replication Manager Overview

Fixed Issues

This section lists fixed issues in the Replication Manager service.

CDPSDX-2879: Ranger import fails when you create a Hive replication policy for a medium duty Data Lake cluster

When you create a Hive replication policy with the Include Sentry Permissions with Metadata or Skip URI Privileges option for a medium duty Data Lake cluster, Ranger import fails. Before you choose the Include Sentry Permissions with Metadata option for a Hive replication policy for a medium duty Data Lake cluster, contact Cloudera Support.

DMX-348: Export remote Hive Metastore step failed while using Replication Manager

While creating a replication policy with no Sentry permissions for the source database/tables, with latest Data Lake clusters, an error message appears The remote command failed with error message : Command (HDFS replication (433)) has failed.

Make sure that while using CDH cluster(s), you do not use MapReduce 1 service.

DMX-355: Currently, Replication Manager does not work efficiently with high-frequency policies

Recommend using policy frequency which is greater than 30 minutes.

DMX-364: Multiple instance of Replication Manager UI displays as "In Progress"

When the replication policy is scheduled with a specific interval, it is seen from the UI that all the instances are "In Progress" and one of the instances is failed.

Refresh the web browser and try again.

DMX-391: Schedule Run option displays only "Does not repeat" and "Custom" options

Recommend to use "Does not repeat" or "Run now" option to schedule the replication policy.

DMX-489

With default replication configurations, small files consume more time to complete the replication process. You can increase the number of mappers to improve the replication performance.

DMX-636: Inconsistency in the value of Timestamp Type column when source ORC is replicated, and source and target clusters are in different time zones

From an Auto TLS source cluster, when a ORC table with static data is replicated, the data in the Timestamp Type column does not match in the target cluster.

DMX-666: Replication fails when the exception "connection timed out" is not handled in Cloudera Manager

Ensure there is connectivity between Source Cloudera Manager and SDX CM. If the source hostname is not resolved to IP, add the host mapping of Cloudera Manager host to /etc/hosts entries of SDX CM.

OPSAPS-61288: Replication setup fails when a non-default namespace is not created on the destination cluster

Create a namespace on the destination cluster before you create an HBase replication policy.

OPSAPS-61596: HBase policy returns "different schema" when tables on source and destination clusters have the same column families

This issue appears because HBase replication policies handle tables that have table attributes incorrectly. Tables with table attributes in replicated tables might lead to other errors as well.

Remove the table attributes for the HBase tables.

OPSAPS-62836

When the first HBase replication policy is created between two clusters where the source cluster is an on-premises cluster, sometimes the policy's status shows Waiting for 'Continue Setup' action call for the first few seconds. No user action is required. The status automatically is updated to Configuring clusters after a few seconds.

OPSAPS-62910, OPSAPS-62948, OPSAPS-62956, and OPSAPS-62998

When you perform HBase replication policy creation, policy update, or policy delete operations on multiple policies between the same cluster pair at once, different failures appear. This is because the HBase peer does not get synchronized during these operations as expected.

Perform HBase replication policy creation, modification, or deletion action one at a time on each HBase replication policy of a cluster pair.

OPSAPS-62995

The HBase replication policy first time setup fails when the destination cluster is a Data Hub.

This issue appears when the HBase classpath is not configured automatically as expected.

To resolve this issue, perform the following steps in Cloudera Manager of the Data Hub before setting up an HBase replication policy:

1. Navigate to the HBase service.
2. Search for the following `Advanced Configuration Snippet (Safety Valve)` values, and add the key and value as shown below:
   - `HBase Service Environment Advanced Configuration Snippet (Safety Valve)`

     Key: HBASE_CLASSPATH

     Value: HBASE_CLASSPATH="$HBASE_CLASSPATH:/opt/cloudera/parcels/CDH/lib/hbase-cdp-jars/"
   - `HBase Client Environment Advanced Configuration Snippet (Safety Valve)`

     Key: HBASE_CLASSPATH

     Value: HBASE_CLASSPATH="$HBASE_CLASSPATH:/opt/cloudera/parcels/CDH/lib/hbase-cdp-jars/"
3. Restart HBase service.

OPSAPS-63071

HBase replication policies from on-premises (CDH5 and CDH6) clusters fail when the source cluster Cloudera Manager version is 7.6.0.

To resolve this issue, perform the following steps in the source cluster Cloudera Manager before you create an HBase replication policy:

1. Navigate to the HBase service.

2. Search for the following `Advanced Configuration Snippet (Safety Valve)` values, and add the key-value pairs as shown below:
   - `HBase Service Environment Advanced Configuration Snippet (Safety Valve)`

     Key: HBASE_CLASSPATH

     Value: HBASE_CLASSPATH="$HBASE_CLASSPATH:*[***LOCATION OF HBASE REPLICATION PARCEL JAR FILE***]*"
   - `HBase Client Environment Advanced Configuration Snippet (Safety Valve)`

     Key: HBASE_CLASSPATH

     Value: HBASE_CLASSPATH="$HBASE_CLASSPATH:*[***LOCATION OF HBASE REPLICATION PARCEL JAR FILE***]*"

   For example, the location of the HBase replication parcel jar in your CDH source cluster might be /opt/cloudera/parcels/CLOUDERA_OPDB_REPLICATION-1.0-1.CLOUDERA_OPDB_REPLICATION5.14.4.p0.9261092/lib/
3. Restart HBase service.

OPSAPS-63138

The `HBase Replication First Time Setup` command runs successfully in the destination cluster though the `Admin Setup HBase replication` subcommand fails on the source cluster.

OPSAPS-63905

Replication Manager runs the clone_snasphot command when restoring snapshots on SFT-enabled clusters without setting the SFT attributes on the table.

OPSAPS-64034

When you delete an HBase replication policy, the HBase peer used in the policy is also deleted if a replicated table name has the - character. Note that the HBase peer is deleted even when there are existing HBase replication policies using this peer. This issue also occurs if a replicated table in any of the policies that uses this peer has the - character in its name.

OPSAPS-64879

Replication policies with an empty name do not appear on the Replication Policies page.Provide a unique replication policy name during replication policy creation.

OPSAPS-65572

When you create an HBase replication policy between a cluster pair where HBase policies between them are suspended (which also means that the corresponding HBase peer is disabled), the HBase peer is enabled at the end of the policy creation process which results in an inconsistent suspend state for the suspended policies.

OPSAPS-65573

HBase replication policy creation and the Retry Failed Snapshots option can be run concurrently.

OPSAPS-66303

A NullPointerException appears when you delete an HBase replication policy for which the policy creation failed because a table in the policy did not exist on the source cluster. After the failed delete action (using the Actions Delete option on the **Replication Policies** page), select Actions Force Delete to get the policy deleted.

OPSAPS-66305

If the snapshot export/import fails during HBase replication policy creation, and then you choose the Actions Retry Failed Snapshots option for the replication policy on the **Replication Policies**

page, the HBase peer is not enabled even if there were no export/import failures during the retry action. In this scenario, the replication peer in the HBase service remains disabled and the Retry Failed Snapshots action fails.

Delete the replication policy and re-create it.

OPSAPS-66327

The HBase peer can be created in the disabled state in CDH 6.0 or higher clusters. However, HBase peer cannot be created in disabled state in CDH 5 clusters and the peer remains enabled throughout the replication setup process. As a workaround, you can create a 'test' HBase replication policy with a small table in CDH 5 clusters. This creates the HBase peer. This issue is fixed.

OPSAPS-66924

When the snapshot export fails during the HBase replication policy job run, the target HBase folder in the destination Data Hub or COD gets deleted if you are using Cloudera Manager versions that are lower than 7.6.7 CHF8, 7.11.0, or 7.9.0-h6 on the source cluster.

As a workaround, you can either revoke the delete permission for the user, or ensure that you use an access key/role that does not have delete permissions for the required storage component. For more information on how to create an access key in AWS or Azure service principal, see Target HBase folder is deleted when HBase replication policy fails.

OPSAPS-66988

You cannot create HBase replication policies if the target CDP version is 7.2.16, 7.2.16.1, 7.2.16.2, or 7.2.16.3 and the source Cloudera Manager version is 7.7.3 or lower, and the source Cloudera Manager API version is v50 or higher.

Tip: The endpoint http://*[***CM_HOST***]*:*[***CM_PORT***]*/api/version shows the API version of the Cloudera Manager.

OPSAPS-67042

Currently, when you create an HBase replication policy for a table, a snapshot is created and then the replication scope is set to "1". Because of the order of steps, the data that is generated between snapshot creation and setting the replication scope is not replicated.

As a workaround, you can set the replication scope of the tables on the source cluster that you want to replicate to "1" and then create the HBase replication policy.

To configure the replication scope for a table on the master cluster, run the "alter *[***TABLE_NAME***]*, {NAME => *[***COLUMN-FAMILY***]*, REPLICATION_SCOPE => 1} command for each column family that must be replicated. `REPLICATION_SCOPE` is a column-family level attribute, where the value '0' means replication is disabled, and '1' means replication is enabled.

OPSAPS-70074

When you choose the Advanced Force Overwrite option during the Hive replication policy creation process to replicate data from CDP Private Cloud Base cluster to CDP Public Cloud, the `Check Hive Metadata` step is skipped during the replication policy run.

OPSAPS-62230

HBase replication no longer fails if the content of the /tmp folder gets deleted while Cloudera Manager is running.

OPSAPS-70297

You can specify a custom username in the Select Source Export snapshot user field during the HBase replication policy creation process. The option appears after you choose the Select Source Perform Initial Snapshot option. Replication Manager uses the specified username on the source cluster to export the initial snapshot to the target.

Cloudera Replication Manager use cases

Date published: 2022-08-11
Date modified: 2024-02-26

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Use cases for Cloudera Replication Manager.....4
Replicating multiple large tables using Cloudera Replication Manager.....4
Tracking, monitoring, and troubleshooting replication policies..... 5

Use cases for Cloudera Replication Manager

Replication Manager is a service in Cloudera Public Cloud. You can create replication policies in Replication Manager to copy and migrate data from CDH (version 5.13 and higher) clusters (HDFS, Hive, and HBase data) and Cloudera Private Cloud Base (version 7.1.4 and higher) clusters (HDFS, Hive external tables, and HBase data) to Cloudera Public Cloud clusters. You can also replicate HDFS data from classic clusters (CDH, Cloudera Private Cloud Base, and HDP clusters) to Cloudera Public Cloud buckets. The supported Cloudera Replication Manager services include Amazon S3 and Microsoft Azure ADLS Gen2 (ABFS).

You can use Replication Manager for a variety of use cases. Some major use cases include:

- Implementing a complete backup and restore solution.

  You might want to implement a backup and restore solution for HDFS data or Hive external tables. You create the replication policy based on the type of data you want to backup and restore. To implement this use case, you back up data in ClusterA to ClusterB. When the need arises, you can create another replication policy to restore the data from ClusterB to ClusterA.
- Migrating legacy data (for example, from CDH clusters) to Cloudera Public Cloud clusters on Amazon S3 and Microsoft Azure ADLS Gen2 (ABFS).

  Replication Manager supports replicating HDFS data, Hive external tables, and HBase data from CDH (version 5.13 and higher) clusters (HDFS, Hive, and HBase data) to Cloudera Public Cloud clusters. Before you create replication policies, you must prepare the clusters and complete the prerequisites.

  Alternatively, you can use CDP CLI to create replication policies to replicate HDFS data and Hive external tables.
- Replicating the required data to another cluster to run workloads or analytics.

  Sometimes, you might want to move workloads, especially heavy-duty workloads to another cluster to reduce the load and optimize the performance of the primary cluster, or run analytics on the required data on another cluster because you do not want to overload the primary cluster. In such scenarios, you can create replication policies in Replication Manager to replicate the required data at regular intervals. After replication, you can use the required tools to analyze the data.

Replicating multiple large tables using Cloudera Replication Manager

You can create multiple replication policies to replicate multiple large HDFS directories, Hive external tables, or HBase tables using Cloudera Replication Manager.

When you have multiple tables or directories to replicate, you might want to create more than one replication policy to replicate the data. This is because even if one replication policy fails, it would not stop the replication process. You can troubleshoot and restart the failed replication policy or create a new replication policy for the tables or directories in the failed replication policy.

Scenario and solution

You want to replicate multiple large HBase tables (around 6000 tables) using the incremental approach. In this approach, you replicate data in batches. For example, 500 tables at a time. This approach ensures that the source cluster is healthy because you replicate data in small batches.

The following steps explain the incremental approach in detail:

1. You create an HBase replication policy for the first 500 tables.

   Internally, Replication Manager performs the following steps:

   a. Creates a disabled HBase peer on the source cluster or disables if the HBase peer existed.
   b. Creates a snapshot and copies it to the target cluster.

      HBase replication policies use snapshots to replicate existing HBase data; this step ensures that all the existing data existing is replicated.
   c. Restores the snapshot to appear as the table on the target.

      This step ensures that the existing data is replicated to the target cluster.
   d. Deletes the snapshot.

      The Replication Manager performs this step after the replication is successfully complete.
   e. Enables table's replication scope for subsequent replication.
   f. Enables the peer.

      This step ensures that the accumulated data is completely replicated.

   After all the accumulated data is migrated, the HBase service continues to replicate new/changed data in this batch of tables automatically.
2. Create another HBase replication policy to replicate the next batch of 500 tables after all the existing data and accumulated data of the first batch of tables is migrated successfully.
3. You can continue this process until all the tables are replicated successfully.

In an ideal scenario, the time taken to replicate 500 tables of 6 TB size might take around four to five hours, and the time taken to replicate the accumulated data might be another 30 minutes to one and a half hours, depending on the speed at which the data is being generated on the source cluster. Therefore, this approach uses 12 batches and around four to five days to replicate all the 6000+ tables to Cloudera Operational Database (COD).

For more information about the cluster versions supported by Replication Manager, clusters preparation process to complete before you create the HBase replication policy, HBase replication policy creation, and for an in depth analysis of the use case, see related information.

Related Information

Using HBase replication policy

Preparing to create an HBase replication policy

Creating an HBase replication policy

Tracking, monitoring, and troubleshooting replication policies

After you create a replication policy, you can track the replication policy job progress, monitor the status of the replication policies, and troubleshoot the failed replication policies in Cloudera Replication Manager.

Tracking and monitoring replication policies

You can view, track, and monitor the available replication policies on the Replication Manager UI:

- **Overview** page. This page shows the statistics in the Policies, Jobs, and Issues & Updates panels.

  Tip: The **Clusters** panel and page shows information about the available classic clusters, Data Lakes, and Data Hubs to use in Replication Manager

- **Replication Policies** page. This page lists all the replication policies and information about each replication policy.

Troubleshooting replication policies

You can use one of the following methods to troubleshoot a failed replication policy:

- Click the failed job on the Replication Manager Replication Policies Job History panel. The errors for the failed job appear.
- Click Cloudera Manager Running Commands for the source or target cluster. The recent command history shows the failed commands.
- Open the service logs in Cloudera Manager to track the errors on the source cluster and target cluster. For example, HBase service logs.

Tip: You can also search on the Cloudera Manager Diagnostics Logs page to view the logs.

Related Information

Overview page
Replication Policies page
Using HDFS replication policies
Using Hive replication policy
Using HBase replication policy
Troubleshooting replication policies in Cloudera Replication Manager

Cloudera Runtime 7.1.1

Introduction to HBase MCC

Date published: 2022-08-10
Date modified: 2022-08-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to HBase Multi-cluster Client.....4
HBase MCC Usage with Kerberos.....5
HBase MCC Usage in Spark with Scala.....7
HBase MCC Usage in Spark with Java.....13
Zookeeper Configurations.....14
HBase MCC Configurations.....14
HBase MCC Restrictions.....15

Introduction to HBase Multi-cluster Client

Learn about the HBase Multi-cluster Client (MCC) and ways to switch between the single HBase clusters and Multi-HBase clients. Explore HBase MCC usage with Kerberos with examples. Implement your HBase clients in Spark using Scala or Java.

Cloudera Operational Database supports HBase MCC that ensures high availability and better performance. HBase MCC relies on the existing HBase replication functionality to provide an eventually consistent solution in cases of primary cluster downtime.

HBase MCC offers the following benefits:

- Switch between single HBase clusters and Multi-HBase Client with limited code changes.
- Support multiple number of linked HBase clusters.
- Continue to keep applications running without disrupting a cluster during an upgrade or outage.

Figure 1: HBase MCC architecture

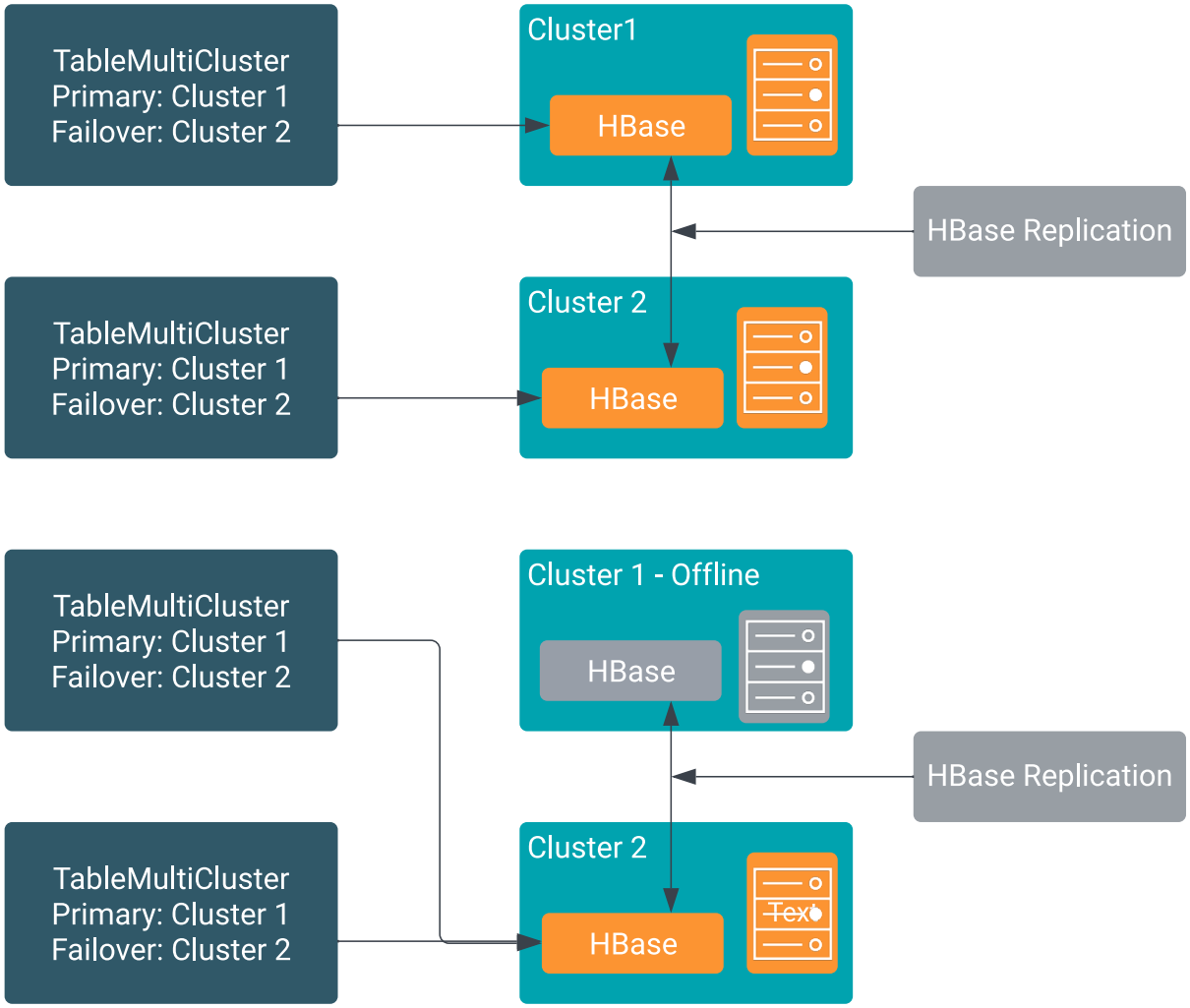

HBase MCC Usage with Kerberos

You can configure HBase MCC with Kerberos authentication for enhanced security and better performance.

When you build a client application and want to add the configurations as resources, you must configure the client resources as follows:

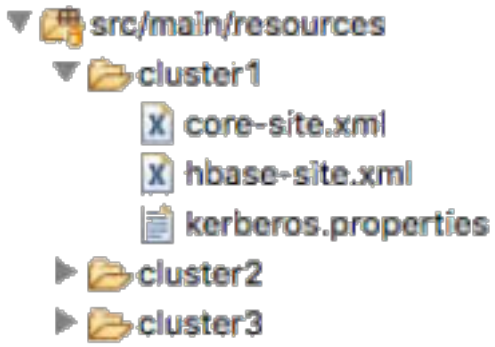

- Each cluster folder inside the client application (src/main/resources) must be named with a lowercase "cluster" followed by a sequence starting with 1 and through the “n” number of clusters.
- The first cluster directory is configured as primary, each additional as a failover.
- There must be at least one folder available in the classpath.

Each folder must contain the following three files:

1. hbase-site.xml
2. core-site.xml
3. kerberos.properties

The kerberos.properties file must be configured as a key value property using the following two configurations:

1. principal
2. keytab

For example,

```
principal=youruser/yourhost@EXAMPLE.COM
keytab=/path/to/youruser.keytab
```

The following Java example of a multi-cluster client MultiClusterConf multiClusterConf = new MultiClusterConf (true) shows that the default constructor is passed as true that creates an instance of an HBase configuration for each cluster assuming default configurations and no override.

```
package org.your.domain.testpackage;

import org.apache.commons.lang.StringUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;

public class AutoConfigClassPath {
 static final Log log = LogFactory.getLog(AutoConfigClassPath.class);
 public static void main(String[] args) throws Exception{
  MultiClusterConf multiClusterConf = new MultiClusterConf(true);  //autoCo
nfigure = true
  Connection connection = ConnectionFactoryMultiClusterWrapper.createConn
ectionMultiUgi(multiClusterConf);
```

```
  String tableName = "test_table";
  String familyName = "cf1";
  int numberOfPuts = 10;
  int milliSecondsOfWait = 200;
  Table table = connection.getTable(TableName.valueOf(tableName));
  for (int i = 1; i <= numberOfPuts; i++) {
   log.info("PUT");
   Put put = new Put(Bytes.toBytes(i % 10 + ".key." + StringUtils.leftPad(
String.valueOf(i), 12)));
   put.addColumn(Bytes.toBytes(familyName), Bytes.toBytes("C"), Bytes.toBy
tes("Value:" + i));
   table.put(put);
   Thread.sleep(milliSecondsOfWait);
  }
  log.info("Closing Connection");
  connection.close();
  log.info(" - Connection Closed");
  System.exit(0);
 }
}

java -cp "hbase-mcc-0.2.0-SNAPSHOT.jar:/path/to/clusters/:/opt/cloudera/parc
els/CDH/jars/*" org.your.domain.testpackage.AutoConfigClassPath
```

Note: You can also build and execute with src/main/resources configured with cluster directories.

You can use autoConfigure to override any parameter in the configuration file, and if the parameter impacts MCC, you must configure with "cluster1".

You can use the multiClusterConf method to change the parameter in the Java program directly. The following example shows setting the parameter and referencing the index of "cluster1" to impact MCC.

```
MultiClusterConf multiClusterConf = new MultiClusterConf(true, false);  //au
toConfigure = true, addResourceOnStart = false
multiClusterConf.set("hbase.mcc.failover.mode", "false", 0);
```

However, it is recommended not to use the previous method; instead update the Java configuration files. Refer to the following code snippet.

```
package com.cloudera.hbase.mcc.examples;

import com.cloudera.hbase.mcc.*;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.util.Bytes;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class BuilderPatternExample {
    private static final Logger LOG = LoggerFactory.getLogger(BuilderPattern
Example.class);
    public static void main(String[] args) throws Exception {
        MultiClusterConf multiClusterConf = new MultiClusterConf.MultiClu
sterConfBuilder()
                .addClusterConfig(new ClusterConfig.ClusterConfigBuilder()
                        .coreSite("/Users/admin/Documents/hbase/cluster1/co
re-site.xml")
                        .hbaseSite("/Users/admin/Documents/hbase/cluster1/
hbase-site.xml")
                        .principal("cluster1tls/server1.example.site@TLS1.C
OM")
```

```
                        .keytab("/Users/admin/Documents/hbase/cluster1/clus
ter1tls_host.keytab")
                        .build()
                        .set("hbase.client.retries.number", "1")
                        .set("hbase.client.pause", "1"))
                        .set("zookeeper.recovery.retry", "0"))
                .addClusterConfig(new ClusterConfig.ClusterConfigBuilder()
                        .coreSite("/Users/admin/Documents/hbase/cluster2/
core-site.xml")
                        .hbaseSite("/Users/admin/Documents/hbase/cluster2/hb
ase-site.xml")
                        .principal("cluster2tls/server2.example.site@TLS2
.COM")
                        .keytab("/Users/admin/Documents/hbase/cluster2/cl
uster2tls_host.keytab")
                        .build()
                        .set("hbase.client.retries.number", "1")
                        .set("hbase.client.pause", "1"))
                .build();

        Connection connection = ConnectionFactoryMultiClusterWrapper.creat
eConnectionMultiUgi(multiClusterConf);
        String tableName = "test_table";
        String familyName = "cf1";
        int numberOfPuts = 3000;
        int milliSecondsOfWait = 200;
        Table table = connection.getTable(TableName.valueOf(tableName));
        for (int i = 1; i <= numberOfPuts; i++) {
            LOG.info("PUT");
            Put put = new Put(Bytes.toBytes("key" + i));
            put.addColumn(Bytes.toBytes(familyName), Bytes.toBytes("C"),
Bytes.toBytes("Value:" + i));
            table.put(put);
            Thread.sleep(milliSecondsOfWait);
        }
        LOG.info("Closing Connection");
        connection.close();
        LOG.info(" - Connection Closed");
        System.exit(0);
    }

}
```

HBase MCC Usage in Spark with Scala

The Spark implementation does not support multiple keytabs. You must configure and enable the cross-realm kerberos for Spark to operate properly.

You do not need to distribute the keytabs or configuration files on the cluster, instead make them available on the edge node from which you start the Spark application. The principal and keytab are referenced twice. First, the credentials must be passed to start the application on the cluster and second, they must be made available to the code responsible for creating credentials for each cluster.

Spark driver writes the credentials to HDFS and later executors obtain and apply them before any code is executed. A process for both the driver and executor is started to ensure the credentials are refreshed in a timely manner.

The following example uses a constructor method for the HBase MCC configuration. It assumes that you are passing hbase-site.xml and core-site.xml into your executable main method. While using this utility, you must modify the configurations using the configuration files only. This utility does not support Spark streaming.

To start the Spark application when the primary cluster's HBase is down, use the following parameter:

```
--conf "spark.security.credentials.hbase.enabled=false"
```

You must override the HBase `UserProvider` class with the `UserProviderMultiCluster` class. If you are using the constructor method, this is set automatically.

```
val HBASE_CLIENT_USER_PROVIDER_CLASS = "hbase.client.userprovider.class"
mccConf.set(HBASE_CLIENT_USER_PROVIDER_CLASS, ConfigConst.HBASE_MCC_USER_PR
OVIDER)
```

Example on writing to HBase with Spark DataFrame API

Example command:

```
spark-submit --class com.cloudera.hbase.mcc.scala.test.SparkToHbase \
--jars "/tmp/hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-class-path "hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.executor.extraClassPath=hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.hbase.connector.security.credentials.enabled=true" \
--conf "spark.yarn.principal=exampleuser/example.kdc.server.com@EXAMPLE.COM"
 \
--conf "spark.yarn.keytab=/tmp/configs/exampleuser.keytab" \
--conf "spark.security.credentials.hbase.enabled=false" \
--files /tmp/configs/hbase-site1.xml,/tmp/configs/core-site1.xml,/tmp/con
figs/hbase-site2.xml,/tmp/configs/core-site2.xml  \
--principal exampleuser/example.kdc.server.com@EXAMPLE.COM \
--keytab /tmp/configs/exampleuser.keytab \
--master yarn --deploy-mode cluster /tmp/ScalaHbaseTest.jar hbase-site1.xml
core-site1.xml hbase-site2.xml core-site2.xml
```

Example code: you can view the example code on this GitHub repository.

Breakdown of the code execution:

- The first part of the code sets up HBase to override the connection implementation in hbase-spark along with setting up variables containing configuration information for the two clusters. A recent addition is `HBASE_CLIENT_USER_PROVIDER_CLASS` used in MCC to allow Spark to start jobs where an HBase cluster may be down for maintenance.

  ```
  val HBASE_CLIENT_CONNECTION_IMPL = "hbase.client.connection.impl"
      val HBASE_CLIENT_USER_PROVIDER_CLASS = "hbase.client.userprovider.clas
  s"
      val CONNECTION_IMPL = "com.cloudera.hbase.mcc.ConnectionMultiCluster"
      val primaryHBaseSite = args(0)
      val primaryCoreSite = args(1)
      val failoverHBaseSite = args(2)
      val failoverCoreSite = args(3)
  ```

- HBase MCC provides a base configuration that stores the configurations for the MCC applications. The first step is to create an instance of the configuration and set the override from the variables above. For spark, it is also required to set the `hbase.mcc.userName` and `hbase.mcc.applicationId`. These are used to write the delegation tokens into HDFS; in addition, it is required to set `fs.defaultFS` from the current Spark execution.

  ```
  val mccConf = new MultiClusterConf
      mccConf.set(HBASE_CLIENT_CONNECTION_IMPL, CONNECTION_IMPL)
      //Sets the default FS to that of the cluster submitting the spark job
      mccConf.set(CommonConfigurationKeysPublic.FS_DEFAULT_NAME_KEY, sc.ha
  doopConfiguration.get(CommonConfigurationKeysPublic.FS_DEFAULT_NAME_KEY))
      mccConf.set("hbase.mmc.userName", sc.sparkUser)
      mccConf.set("hbase.mcc.applicationId", sc.applicationId)
  ```

- The HBase configurations are stored independently and prefixed when loaded into the application to maintain isolation between each cluster configuration and the base MCC configuration. The following code segment shows HBase execution related parameters, which you plan to override.

```
val primary = HBaseConfiguration.create()
    primary.addResource(new Path(primaryHBaseSite))
    primary.addResource(new Path(primaryCoreSite))
    primary.set("hbase.client.retries.number", "1"); //Override Default
 Parameters
    primary.set("hbase.client.pause", "1"); //Override Default Parameters
    primary.set("zookeeper.recovery.retry", "0"); //Override Default Param
eters

    val failover = HBaseConfiguration.create()
    failover.addResource(new Path(failoverHBaseSite))
    failover.addResource(new Path(failoverCoreSite))
    failover.set("hbase.client.retries.number", "1"); //Override Default
 Parameters
    failover.set("hbase.client.pause", "1"); //Override Default Parameters
```

- Once the configurations for each HBase cluster have been created, it is time to call the code that manages the delegation tokens for the clusters. This process returns the token name to the configuration and ensures that the executor is reading the proper tokens and applying the credentials.

```
val credentialsManager = CredentialsManager.getInstance
    primary.set(ConfigConst.HBASE_MCC_TOKEN_FILE_NAME, credentialsManage
r.confTokenForCluster(primaryHBaseSite, primaryCoreSite, sc))
    failover.set(ConfigConst.HBASE_MCC_TOKEN_FILE_NAME, credentialsManager
.confTokenForCluster(failoverHBaseSite, failoverCoreSite, sc))
```

- Each of the final cluster configurations must be added to the MCC configuration and the following function prefixes all of the parameters for each cluster. As the hbase-spark implementation only takes a single configuration, everything is merged into an Uber configuration.

```
mccConf.addClusterConfig(primary)
    mccConf.addClusterConfig(failover)
```

- The `HBaseContext` from hbase-spark is created and the dataframe API is used as in any other case for Spark.

```
import spark.implicits._

    new HBaseContext(sc, mccConf.getConfiguration())

    val rdd = sc.parallelize(Seq(("rowkey","SparkToHbase","0","","1")))
    val df_withcol = rdd.toDF("rowKey", "application", "batchId", "timeS
tamp", "loaded_events")
    logger.info("Data frame to Hbase : " + df_withcol.show())
    val hbaseTableName = "test_table"
    val hbaseTableSchema =""""rowKey STRING :key, application STRING cf1:
APP, batchId STRING cf1:BID, timeStamp STRING cf1:TS, loaded_events STRING
 cf1:PR"""
    logger.info("Started writing to Hbase")
    df_withcol.write.format("org.apache.hadoop.hbase.spark")
      .options(Map("hbase.columns.mapping" -> hbaseTableSchema, "hbase.tab
le" -> hbaseTableName))
      .save()
    logger.info("Completed writing to hbase")
```

Example on writing to HBase with Spark Streaming

The following example shows an application to an HDFS directory for new files. When a file is added, it counts each of the words, keeping record of the count for any previously read file and then updates HBase with the new count.

Example command:

```
spark-submit --class com.cloudera.hbase.mcc.scala.test.SparkStreamingExample
 \
--jars "/tmp/hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-class-path "hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.executor.extraClassPath=hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.hbase.connector.security.credentials.enabled=true" \
--conf "spark.yarn.principal=exampleuser/example.kdc.server.com@EXAMPLE.COM"
 \
--conf "spark.yarn.keytab=/tmp/configs/exampleuser.keytab" \
--conf "spark.security.credentials.hbase.enabled=false" \
--files /tmp/configs/hbase-site1.xml,/tmp/configs/core-site1.xml,/tmp/config
s/hbase-site2.xml,/tmp/configs/core-site2.xml  \
--principal exampleuser/example.kdc.server.com@EXAMPLE.COM \
--keytab /tmp/configs/exampleuser.keytab \
--master yarn --deploy-mode cluster /tmp/ScalaHbaseTest.jar hbase-site1.xml
 core-site1.xml hbase-site2.xml core-site2.xml
```

Example code: you can view the example code on this GitHub repository.

In this example the configuration must be executed inside the `putHBase` function. That is why the variables for the token file names must be set outside of the function. If you try to use the Spark Streaming Context inside that function, it throws an error as MCC is not serializable. This does not happen in structured streaming which is shown in the following examples.

Example on reading from HDFS and writing to HBase with Spark structured streaming

In this example an application is created that will listen to an HDFS directory for new files. When a file is added, it will read the file into a micro-batch using structured streaming and push the results into HBase.

Example command:

```
spark-submit --class com.cloudera.hbase.mcc.scala.test.StructuredStreamingHD
FSToHBase \
--jars "/tmp/hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-class-path "hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-memory 2g \
--executor-memory 2g \
--conf "spark.executor.extraClassPath=hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.hbase.connector.security.credentials.enabled=true" \
--conf "spark.yarn.principal=exampleuser/example.kdc.server.com@EXAMPLE.C
OM" \
--conf "spark.yarn.keytab=/tmp/configs/exampleuser.keytab" \
--conf "spark.security.credentials.hbase.enabled=false" \
--files /tmp/cluster1/hbase-site1.xml,/tmp/cluster1/core-site1.xml,/tmp/clu
ster2/hbase-site2.xml,/tmp/cluster2/core-site2.xml \
--principal exampleuser/example.kdc.server.com@EXAMPLE.COM \
--keytab /tmp/configs/exampleuser.keytab \
--master yarn \
--deploy-mode cluster /tmp/ScalaHbaseTest.jar hbase-site1.xml core-site1.xml
 hbase-site2.xml core-site2.xml \
/user/exampleuser/StructuredStreamingExampleSource/ /user/exampleuser/Spa
rkStreamingExampleCheckpoint/
```

Example code: you can view the example code on this GitHub repository.

Example on reading from Kafka and writing to HBase with Spark structured streaming

The following example shows how to create an application that pulls messages from Kafka. When you push a set of messages to Kafka, this process creates a micro-batch using the structured streaming and push the results into HBase.

For this use case, a properties file is set up to pass into the application for the required Kafka configurations. Example properties file:

```
spark.kafkatohbase.target.hbase.table test_table
spark.kafkatohbase.target.hbase.cf cf1
spark.kafkatohbase.source.kafka.topic spark_2
spark.kafkatohbase.checkpoint.dir /user/exampleuser/SparkStreamingExampleC
heckpoint/
spark.kafkatohbase.kafka.ssl.truststore.location ./cm-auto-global_trustst
ore.jks
spark.kafkatohbase.kafka.bootstrap.servers yourzookeeper1:9093,yourzookeeper
2:9093,yourzookeeper3:9093
spark.kafkatohbase.kafka.security.protocol SASL_SSL
spark.kafkatohbase.kafka.ssl.truststore.password trustorepasswordhere
spark.kafkatohbase.startingOffsets earliest
spark.kafkatohbase.keytab kafka.keytab
spark.kafkatohbase.principal kafka/example.server.com@EXAMPLE.COM
```

A jaas.conf file also need to be set.

```
KafkaClient {
    com.sun.security.auth.module.Krb5LoginModule required
    debug =true
    useKeyTab=true
    storeKey=true
    keyTab="/tmp/kafka.keytab"
    useTicketCache=false
    serviceName="kafka"
    principal="kafka/example.server.com@EXAMPLE.COM";
};
```

Example command:

```
hdfs dfs -rm -r /user/cluster1tls/SparkStreamingExampleCheckpoint/
hdfs dfs -mkdir /user/cluster1tls/SparkStreamingExampleCheckpoint/
spark-submit --class com.cloudera.hbase.mcc.scala.test.StructuredStreaming
ExampleKafkaToHBase \
--jars "/tmp/hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-class-path "hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-memory 2g \
--executor-memory 2g \
--conf spark.streaming.receiver.maxRate=20 \
--conf spark.dynamicAllocation.minExecutors=5 \
--conf spark.dynamicAllocation.initialExecutors=5 \
--conf spark.dynamicAllocation.maxExecutors=10 \
--conf "spark.executor.extraClassPath=hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.hbase.connector.security.credentials.enabled=true" \
--conf "spark.yarn.principal=exampleuser/example.server.com@EXAMPLE.COM" \
--conf "spark.yarn.keytab=/tmp/configs/exampleuser.keytab" \
--conf "spark.security.credentials.hbase.enabled=false" \
--conf "spark.security.credentials.hive.enabled=false" \
--conf "spark.security.credentials.hdfs.enabled=false" \
--driver-java-options "-Djava.security.auth.login.config=./producer_jaas2
.conf -Djava.io.tmpdir=/tmp"  \
--conf "spark.executor.extraJavaOptions=-Djava.security.auth.login.config=./
producer_jaas2.conf -Djava.io.tmpdir=/tmp" \
--files /tmp/cluster1/hbase-site1.xml,/tmp/cluster1/core-site1.xml,/tmp/clu
ster2/hbase-site2.xml,/tmp/cluster2/core-site2.xml,/tmp/producer_jaas.conf#p
roducer_jaas.conf,/var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_trus
tstore.jks,/tmp/kafka.keytab \
--principal exampleuser/example.server.com@EXAMPLE.COM" \
--keytab /tmp/cluster1/exampleuser.keytab \
```

```
--master yarn \
--deploy-mode cluster \
--properties-file  /tmp/properties.conf \
/tmp/ScalaHbaseTest.jar hbase-site1.xml core-site1.xml hbase-site2.xml core-
site2.xml
```

Example code: you can view the example code on this GitHub repository.

Example on writing to HBase bulk put and hbase-spark

Example command:

```
spark-submit --class com.cloudera.hbase.mcc.scala.test.HBaseMCCBulkPutExampl
e \
--jars "/tmp/hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-class-path "hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.executor.extraClassPath=hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.hbase.connector.security.credentials.enabled=true" \
--conf "spark.yarn.principal=exampleuser/example.kdc.server.com@EXAMPLE.C
OM" \
--conf "spark.yarn.keytab=/tmp/configs/exampleuser.keytab" \
--conf "spark.security.credentials.hbase.enabled=false" \
--files /tmp/configs/hbase-site1.xml,/tmp/configs/core-site1.xml,/tmp/confi
gs/hbase-site2.xml,/tmp/configs/core-site2.xml  \
--principal exampleuser/example.kdc.server.com@EXAMPLE.COM \
--keytab /tmp/configs/exampleuser.keytab \
--master yarn --deploy-mode cluster /tmp/ScalaHbaseTest.jar hbase-site1.xml
 core-site1.xml hbase-site2.xml core-site2.xml
```

Note: This example is using the MCC configurations constructor to eliminate the boilerplate configurations.

Example code: you can view the example code on this GitHub repository.

Example on writing to HBase bulk get and hbase-spark

Example command:

```
spark-submit --class com.cloudera.hbase.mcc.scala.test.HBaseMCCBulkGetExampl
e \
--jars "/tmp/hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-class-path "hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.executor.extraClassPath=hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.hbase.connector.security.credentials.enabled=true" \
--conf "spark.yarn.principal=exampleuser/example.kdc.server.com@EXAMPLE.C
OM" \
--conf "spark.yarn.keytab=/tmp/configs/exampleuser.keytab" \
--conf "spark.security.credentials.hbase.enabled=false" \
--files /tmp/configs/hbase-site1.xml,/tmp/configs/core-site1.xml,/tmp/confi
gs/hbase-site2.xml,/tmp/configs/core-site2.xml  \
--principal exampleuser/example.kdc.server.com@EXAMPLE.COM \
--keytab /tmp/configs/exampleuser.keytab \
--master yarn --deploy-mode cluster /tmp/ScalaHbaseTest.jar hbase-site1.xml
 core-site1.xml hbase-site2.xml core-site2.xml
```

Note: This example is using the MCC configurations constructor to eliminate the boilerplate configurations.

Example code: you can view the example code on this GitHub repository.

Example on writing to HBase bulk delete and hbase-spark

Example command:

```
spark-submit --class com.cloudera.hbase.mcc.scala.test.HBaseMCCBulkDeleteExa
mple \
--jars "/tmp/hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-class-path "hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.executor.extraClassPath=hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.hbase.connector.security.credentials.enabled=true" \
--conf "spark.yarn.principal=exampleuser/example.kdc.server.com@EXAMPLE.COM"
 \
--conf "spark.yarn.keytab=/tmp/configs/exampleuser.keytab" \
--conf "spark.security.credentials.hbase.enabled=false" \
--files /tmp/configs/hbase-site1.xml,/tmp/configs/core-site1.xml,/tmp/config
s/hbase-site2.xml,/tmp/configs/core-site2.xml  \
--principal exampleuser/example.kdc.server.com@EXAMPLE.COM \
--keytab /tmp/configs/exampleuser.keytab \
--master yarn --deploy-mode cluster /tmp/ScalaHbaseTest.jar hbase-site1.xml
 core-site1.xml hbase-site2.xml core-site2.xml
```

Example code: you can view the example code on this GitHub repository.

HBase MCC Usage in Spark with Java

Example on writing to HBase bulk put and hbase-spark

Example command:

```
spark-submit --class com.cloudeara.hbase.mcc.java.HBaseMCCBulkPutExample \
--jars "/tmp/hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-class-path "hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.executor.extraClassPath=hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.hbase.connector.security.credentials.enabled=true" \
--conf "spark.yarn.principal=exampleuser/example.kdc.server.com@EXAMPLE.COM"
 \
--conf "spark.yarn.keytab=/tmp/configs/exampleuser.keytab" \
--conf "spark.security.credentials.hbase.enabled=false" \
--files /tmp/configs/hbase-site1.xml,/tmp/configs/core-site1.xml,/tmp/config
s/hbase-site2.xml,/tmp/configs/core-site2.xml  \
--principal exampleuser/example.kdc.server.com@EXAMPLE.COM \
--keytab /tmp/configs/exampleuser.keytab \
--master yarn --deploy-mode cluster /tmp/JavaMCCExamples-0.0.1-SNAPSHOT.jar
 hbase-site1.xml core-site1.xml hbase-site2.xml core-site2.xml
```

Example code: you can view the example code on this GitHub repository.

Example on writing to HBase bulk get and hbase-spark

Example command:

```
spark-submit --class com.cloudeara.hbase.mcc.java.HBaseMCCBulkGetExample \
--jars "/tmp/hbase-mcc-0.2.0-SNAPSHOT.jar" \
--driver-class-path "hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.executor.extraClassPath=hbase-mcc-0.2.0-SNAPSHOT.jar" \
--conf "spark.hbase.connector.security.credentials.enabled=true" \
--conf "spark.yarn.principal=exampleuser/example.kdc.server.com@EXAMPLE.COM"
 \
--conf "spark.yarn.keytab=/tmp/configs/exampleuser.keytab" \
--conf "spark.security.credentials.hbase.enabled=false" \
```

```
--files /tmp/configs/hbase-site1.xml,/tmp/configs/core-site1.xml,/tmp/config
s/hbase-site2.xml,/tmp/configs/core-site2.xml  \
--principal exampleuser/example.kdc.server.com@EXAMPLE.COM \
--keytab /tmp/configs/exampleuser.keytab \
--master yarn --deploy-mode cluster /tmp/JavaMCCExamples-0.0.1-SNAPSHOT.jar
 hbase-site1.xml core-site1.xml hbase-site2.xml core-site2.xml
```

Example code: you can view the example code on this GitHub repository.

Zookeeper Configurations

HBase MCC uses the zookeeper.recovery.retry parameter to monitor the number of retries when Zookeeper is down.

zookeeper.recovery.retry

When a cluster is online and an application is started, the HBase client communicates to the Zookeeper quorum. When you perform a read or write operation on region servers and a failover happens, MCC runs a background thread to check when HBase becomes available again. If an application is started and the cluster is down, MCC uses the zookeeper.recovery.retry parameter to check Zookeeper multiple times. This might not be desired, especially for Spark streaming which operates with a microbatch. Each batch needs to iterate over the default before it finally fails over and hit the second cluster.

You can set this parameter to either 1 or 0. You can set the parameters differently for each cluster; set it as default for the second cluster and 0 for the primary so that it may fail over quickly. In addition, when you set zookeeper.recovery.retry to 0 with HBase, the number of client connections to Zookeeper increases and you might want to monitor `maxclientcnxns`. This is applicable for a failure scenario, when an application is starting and Zookeeper is down.

HBase MCC Configurations

HBase Multi-cluster Client (MCC) provides various configuration parameters to set up clusters, mode, and different connection related properties.

hbase.mcc.failover.clusters

This is a comma separated list of cluster names that represent the clusters available when running MCC. The first cluster in the list is the primary cluster. If this cluster fails to respond, the requests are routed to the next cluster in line. This configuration is automatically generated based on N+ number of clusters in the submission.

The following example shows an uber configuration.

```
hbase.failover.clusters=hbase.mcc.cluster0,hbase.mcc.cluster1
hbase.mcc.cluster0.hbase.zookeeper.quorum=1.0.0.2
hbase.mcc.cluster1.hbase.zookeeper.quorum=1.0.0.3
```

Note: Auto-configuration does not support building uber configuration.

hbase.mcc.failover.mode

The default value is set to true, this means that if the primary cluster fails, the code automatically attempts to execute on failover clusters until all failover clusters are exhausted. If set to false, the only attempt is on the primary or first cluster. If you are executing a put or delete when set to true and culter failover happens, HBase MCC executes the put or delete on the failover cluster.

This configuration is not applicable for the read or scan methods.

Note: This configuration does not apply to reads or scans. This is applied to puts or deletes.

hbase.mcc.connection.pool.size

The default value is set as 20. Part of `java.util.concurrent.Executors` creates a thread pool that reuses a fixed number of threads operating from a shared unbounded queue.

hbase.mcc.allow.check.and.mutate

The default value is false that disables the `checkAndMutate` function on the Table class.

hbase.mcc.wait.time.before.trying.primary.after.failure

The default value is 30000 milliseconds (30 seconds). This is the default amount of time in milliseconds that the application waits before re-attempting the primary cluster after a failure and executing against a failover cluster. The cluster tracker uses this thread to check and ensure that a cluster is available before calls are routed to that cluster.

hbase.mcc.wait.time.before.enable.cluster

The default value is 10000 milliseconds (10 seconds). This represents the wait time to read the metadata for a cluster before the region server is actually available to read. If this number is too low, the system detects that the cluster is available from the metastore, tries to read or write from the cluster because the region server is decoupled from the meta and is not yet available. This additional buffer allows some time before the metadata is available for the region to become fully online.

hbase.mcc.buffered.mutator.behavior

The supported values are failover, replicate, and replay. The default value is failover. When `BufferedMutator` is used, there might be a data loss. When set to failover, HBase MCC writes to a single cluster connection. When a flush fails for that connection, it starts writing new mutations to the new connection. Any mutations that are part of the failure flush are lost. Data loss in `BufferedMutator` is caused when HBase puts are stored in client memory up until the point that a flush takes place. If that flush fails when a region server is down, data is not be persisted.

The second option is to configure `BufferedMutator` with replicate, which replicates all calls for mutations to all connections. When a connection fails, data is not lost because we are writing to another connection automatically.

The third option is to configure replay, this operation stores data in a cache until flush is called. When flush is called, HBase MCC attempts to write all data from the cache into the first available and active connection’s `BufferedMutator` followed by an immediate flush. If the flush is a success, the cache is cleared. If the flush fails, HBase MCC pushes the data to the next connection's `BufferedMutator`. This also helps to prevent data loss with `BufferedMutator` while not pushing to each cluster by default as an alternative to the replicate option.

In all cases with `BufferedMutator`, you could still have data loss on the cluster when services go down or if no cluster is available to write the data. You need to ensure that your application is ready to handle the failures as they happen.

HBase MCC Restrictions

HBase MCC contains a few restrictions when you are executing it in Spark compared to executing it in a standard Java application.

Only a single keytab can be used with Spark. In the Java implementation multiple UGIs can be called as long as a proper krb5.conf can connect easily to different clusters in different realms with multiple keytabs.

Cloudera Runtime 7.2.11

Accessing Apache HBase

Date published: 2020-02-29
Date modified: 2023-05-04

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Use the HBase shell 4
Virtual machine options for HBase Shell 4
Script with HBase Shell 4

Use the HBase command-line utilities 5

Use the HBase APIs for Java 11

Use the HBase REST server 12
Installing the REST Server using Cloudera Manager 12
Using the REST API 12
Using the REST proxy API 18

Using the Apache Thrift Proxy API 19
Preparing a thrift server and client 19
List of Thrift API and HBase configurations 20
Example for using THttpClient API in secure cluster 21
Example for using THttpClient API in unsecure cluster 23
Example for using TSaslClientTransport API in secure cluster without HTTP 24

Using Apache HBase Hive integration 25
Configure Hive to use with HBase 25
Using HBase Hive integration 25

Using the HBase-Spark connector 27
Configuring HBase-Spark connector using Cloudera Manager when HBase and Spark are on the same cluster 27
Configuring HBase-Spark connector using Cloudera Manager when HBase is on a remote cluster 29
Example: Using the HBase-Spark connector 32

Use the Hue HBase app 37
Configure the HBase thrift server role 38

Use the HBase shell

You can use the HBase Shell from the command line interface to communicate with HBase. In Cloudera, you can create a namespace and manage it using the HBase shell. Namespaces contain collections of tables and permissions, replication settings, and resource isolation.

In Cloudera, you need to SSH into an HBase node before you can use the HBase Shell. For example, to SSH into an HBase node with the IP address 10.10.10.10, you must use the command:

```
ssh <USERNAME>@10.10.10.10
```

Note: You must use your IPA password for authentication.

After you have started HBase, you can access the database in an interactive way by using the HBase Shell, which is a command interpreter for HBase which is written in Ruby. Always run HBase administrative commands such as the HBase Shell, hbck, or bulk-load commands as the HBase user (typically hbase).

```
hbase shell
```

You can use the following commands to get started with the HBase shell:

- To get help and to see all available commands, use the help command.
- To get help on a specific command, use help "command". For example:

  ```
  hbase> help "create"
  ```

- To remove an attribute from a table or column family or reset it to its default value, set its value to nil. For example, use the following command to remove the KEEP_DELETED_CELLS attribute from the f1 column of the users table:

  ```
  hbase> alter 'users', { NAME => 'f1', KEEP_DELETED_CELLS => nil }
  ```

- To exit the HBase Shell, type quit.

Virtual machine options for HBase Shell

You can set variables for the virtual machine running HBase Shell, by using the HBASE_SHELL_OPTS environment variable. This example sets several options in the virtual machine.

This example sets several options in the virtual machine.

```
$ HBASE_SHELL_OPTS="-verbose:gc -XX:+PrintGCApplicationStoppedTime -XX:+Prin
tGCDateStamps
      -XX:+PrintGCDetails -Xloggc:$HBASE_HOME/logs/gc-hbase.log" ./bin/hbase
 shell
```

Script with HBase Shell

You can use HBase shell in your scripts. You can also write Ruby scripts for use with HBase Shell. Example Ruby scripts are included in the hbase-examples/src/main/ruby/ directory.

The non-interactive mode allows you to use HBase Shell in scripts, and allow the script to access the exit status of the HBase Shell commands. To invoke non-interactive mode, use the -n or --non-interactive switch. This small example script shows how to use HBase Shell in a Bash script.

```
#!/bin/bash
echo 'list' | hbase shell -n
status=$?
if [$status -ne 0]; then
  echo "The command may have failed."
fi
```

Successful HBase Shell commands return an exit status of 0. However, an exit status other than 0 does not necessarily indicate a failure, but should be interpreted as unknown. For example, a command may succeed, but while waiting for the response, the client may lose connectivity. In that case, the client has no way to know the outcome of the command. In the case of a non-zero exit status, your script should check to be sure the command actually failed before taking further action.

You can use the get_splits command, which returns the split points for a given table:

```
hbase> get_splits 't2'
Total number of splits = 5

=> ["", "10", "20", "30", "40"]
```

Use the HBase command-line utilities

Besides the HBase Shell, HBase includes several other command-line utilities, which are available in the hbase/bin/ directory of each HBase host. This topic provides basic usage instructions for the most commonly used utilities.

PerformanceEvaluation

The PerformanceEvaluation utility allows you to run several preconfigured tests on your cluster and reports its performance. To run the PerformanceEvaluation tool, use the bin/hbase pecommand.

```
$ hbase pe

Usage: java org.apache.hadoop.hbase.PerformanceEvaluation \
  <OPTIONS> [-D<property=value>]* <command> <nclients>

Options:
 nomapred        Run multiple clients using threads (rather than use mapred
uce)
 rows            Rows each client runs. Default: One million
 size            Total size in GiB. Mutually exclusive with --rows. Default:
 1.0.
 sampleRate      Execute test on a sample of total rows. Only supported by r
andomRead.
                 Default: 1.0
 traceRate       Enable HTrace spans. Initiate tracing every N rows. Defaul
t: 0
 table           Alternate table name. Default: 'TestTable'
 multiGet        If >0, when doing RandomRead, perform multiple gets instead
 of single
                 gets.
                 Default: 0
 compress        Compression type to use (GZ, LZO, ...). Default: 'NONE'
 flushCommits    Used to determine if the test should flush the table. Defau
lt: false
 writeToWAL      Set writeToWAL on puts. Default: True
```

```
 autoFlush       Set autoFlush on htable. Default: False
 oneCon          all the threads share the same connection. Default: False
 presplit        Create presplit table. Recommended for accurate perf analy
sis (see
                 guide). Default: disabled
 inmemory        Tries to keep the HFiles of the CF inmemory as far as possi
ble. Not
                 guaranteed that reads are always served from memory.  Defa
ult: false
 usetags         Writes tags along with KVs. Use with HFile V3. Default:
false
 numoftags       Specify the no of tags that would be needed. This works o
nly if usetags
                 is true.
 filterAll       Helps to filter out all the rows on the server side there
by not returning
                 anything back to the client.  Helps to check the server si
de performance.
                 Uses FilterAllFilter internally.
 latency         Set to report operation latencies. Default: False
 bloomFilter     Bloom filter type, one of [NONE, ROW, ROWCOL]
 valueSize       Pass value size to use: Default: 1024
 valueRandom     Set if we should vary value size between 0 and 'valueSiz
e'; set on read
                 for stats on size: Default: Not set.
 valueZipf       Set if we should vary value size between 0 and 'valueSize'
 in zipf form:
                 Default: Not set.
 period          Report every 'period' rows: Default: opts.perClientRunRo
ws / 10
 multiGet        Batch gets together into groups of N. Only supported by ran
domRead.
                 Default: disabled
 addColumns      Adds columns to scans/gets explicitly. Default: true
 replicas        Enable region replica testing. Defaults: 1.
 splitPolicy     Specify a custom RegionSplitPolicy for the table.
 randomSleep     Do a random sleep before each get between 0 and entered v
alue. Defaults: 0
 columns         Columns to write per row. Default: 1
 caching         Scan caching to use. Default: 30

 Note: -D properties will be applied to the conf used.
  For example:
   -Dmapreduce.output.fileoutputformat.compress=true
   -Dmapreduce.task.timeout=60000
Command:
 append          Append on each row; clients overlap on keyspace so some c
oncurrent
                 operations
 checkAndDelete  CheckAndDelete on each row; clients overlap on keyspace so
 some concurrent
                 operations
 checkAndMutate  CheckAndMutate on each row; clients overlap on keyspace so
 some concurrent
                 operations
 checkAndPut     CheckAndPut on each row; clients overlap on keyspace so s
ome concurrent
                 operations
 filterScan      Run scan test using a filter to find a specific row based
 on it's value
                 (make sure to use --rows=20)
 increment       Increment on each row; clients overlap on keyspace so some
 concurrent
                 operations
```

```
 randomRead      Run random read test
 randomSeekScan  Run random seek and scan 100 test
 randomWrite     Run random write test
 scan            Run scan test (read every row)
 scanRange10     Run random seek scan with both start and stop row (max 10
rows)
 scanRange100    Run random seek scan with both start and stop row (max 100
 rows)
 scanRange1000   Run random seek scan with both start and stop row (max 1000
 rows)
 scanRange10000  Run random seek scan with both start and stop row (max 1
0000 rows)
 sequentialRead  Run sequential read test
 sequentialWrite Run sequential write test
Args:
 nclients        Integer. Required. Total number of clients (and HRegionS
ervers)
                 running: 1 <= value <= 500
Examples:
 To run a single client doing the default 1M sequentialWrites:
 $ bin/hbase org.apache.hadoop.hbase.PerformanceEvaluation sequentialWrite 1
 To run 10 clients doing increments over ten rows:
 $ bin/hbase org.apache.hadoop.hbase.PerformanceEvaluation --rows=10 --noma
pred increment 10
```

LoadTestTool

The LoadTestTool utility load-tests your cluster by performing writes, updates, or reads on it. To run the LoadTest Tool, use the bin/hbase ltt command. To print general usage information, use the -h option.

```
$ bin/hbase ltt -h

Options:
 -batchupdate                    Whether to use batch as opposed to separate
 updates for every column
                                 in a row
 -bloom <arg>                    Bloom filter type, one of [NONE, ROW, ROWC
OL]
 -compression <arg>              Compression type, one of [LZO, GZ, NONE, SN
APPY, LZ4]
 -data_block_encoding <arg>      Encoding algorithm (e.g. prefix compress
ion) to use for data blocks
                                 in the test column family, one of
                                 [NONE, PREFIX, DIFF, FAST_DIFF, PREFIX_T
REE].
 -deferredlogflush               Enable deferred log flush.
 -encryption <arg>               Enables transparent encryption on the test
 table, one of [AES]
 -families <arg>                 The name of the column families to use se
parated by comma
 -generator <arg>                The class which generates load for the too
l. Any args for this class
                                 can be passed as colon separated after c
lass name
 -h,--help                       Show usage
 -in_memory                      Tries to keep the HFiles of the CF inmemory
 as far as possible.  Not
                                 guaranteed that reads are always served fro
m inmemory
 -init_only                      Initialize the test table only, don't do
 any loading
```

```
 -key_window <arg>                  The 'key window' to maintain between reads
 and writes for concurrent
                                    write/read workload. The default is 0.
 -max_read_errors <arg>             The maximum number of read errors to tol
erate before terminating all
                                    reader threads. The default is 10.
 -mob_threshold <arg>               Desired cell size to exceed in bytes that
will use the MOB write path
 -multiget_batchsize <arg>          Whether to use multi-gets as opposed to sep
arate gets for every
                                    column in a row
 -multiput                          Whether to use multi-puts as opposed to s
eparate puts for every
                                    column in a row
 -num_keys <arg>                    The number of keys to read/write
 -num_regions_per_server <arg>      Desired number of regions per region serv
er. Defaults to 5.
 -num_tables <arg>                  A positive integer number. When a number n
 is specified, load test tool
                                    will load n table parallely. -tn parameter
value becomes table name prefix.
                                    Each table name is in format <tn>_1...<tn>
_n
 -read <arg>                        <verify_percent>[:<#threads=20>]
 -reader <arg>                      The class for executing the read requests
 -region_replica_id <arg>           Region replica id to do the reads from
 -region_replication <arg>          Desired number of replicas per region
 -regions_per_server <arg>          A positive integer number. When a number n
 is specified, load test tool
                                    will create the test table with n regions p
er server
 -skip_init                         Skip the initialization; assume test table
 already exists
 -start_key <arg>                   The first key to read/write (a 0-based ind
ex). The default value is 0.
 -tn <arg>                          The name of the table to read or write
 -update <arg>                      <update_percent>[:<#threads=20>][:<#whether
 to ignore nonce collisions=0>]
 -updater <arg>                     The class for executing the update requests
 -write <arg>                       <avg_cols_per_key>:<avg_data_size>[:<#thr
eads=20>]
 -writer <arg>                      The class for executing the write requests
 -zk <arg>                          ZK quorum as comma-separated host names w
ithout port numbers
 -zk_root <arg>                     name of parent znode in zookeeper
```

wal

The wal utility prints information about the contents of a specified WAL file. To get a list of all WAL files, use the HDFS command hadoop fs -ls -R /hbase/WALs. To run the wal utility, use the bin/hbase wal command. Run it without options to get usage information.

```
hbase wal
usage: WAL <filename...> [-h] [-j] [-p] [-r <arg>] [-s <arg>] [-w <arg>]
 -h,--help             Output help message
 -j,--json             Output JSON
 -p,--printvals        Print values
 -r,--region <arg>     Region to filter by. Pass encoded region name; e.g.
                       '9192caead6a5a20acb4454ffbc79fa14'
 -s,--sequence <arg>   Sequence to filter by. Pass sequence number.
 -w,--row <arg>        Row to filter by. Pass row name.
```

hfile

The hfile utility prints diagnostic information about a specified hfile, such as block headers or statistics. To get a list of all hfiles, use the HDFS command hadoop fs -ls -R /hbase/data. To run the hfile utility, use the bin/hbase hf ilecommand. Run it without options to get usage information.

```
$ hbase hfile

usage: HFile [-a] [-b] [-e] [-f <arg> | -r <arg>] [-h] [-i] [-k] [-m] [-p]
       [-s] [-v] [-w <arg>]
 -a,--checkfamily         Enable family check
 -b,--printblocks         Print block index meta data
 -e,--printkey            Print keys
 -f,--file <arg>          File to scan. Pass full-path; e.g.
                          hdfs://a:9000/hbase/hbase:meta/12/34
 -h,--printblockheaders   Print block headers for each block.
 -i,--checkMobIntegrity   Print all cells whose mob files are missing
 -k,--checkrow            Enable row order check; looks for out-of-order
                          keys
 -m,--printmeta           Print meta data of file
 -p,--printkv             Print key/value pairs
 -r,--region <arg>        Region to scan. Pass region name; e.g.
                          'hbase:meta,,1'
 -s,--stats               Print statistics
 -v,--verbose             Verbose output; emits file and meta data
                          delimiters
 -w,--seekToRow <arg>     Seek to this row and print all the kvs for this
                          row only
```

hbck

The hbck utility checks and optionally repairs errors in HFiles.

Warning: Running hbck with any of the -fix or -repair commands is dangerous and can lead to data loss. Contact Cloudera support before running it.

To run hbck, use the bin/hbase hbck command. Run it with the -h option to get more usage information.

```
---------------------------------------------------------------------------
NOTE: As of HBase version 2.0, the hbck tool is significantly changed.
In general, all Read-Only options are supported and can be be used
safely. Most -fix/ -repair options are NOT supported. Please see usage
below for details on which options are not supported.
---------------------------------------------------------------------------

Usage: fsck [opts] {only tables}
 where [opts] are:
   -help Display help options (this)
   -details Display full report of all regions.
   -timelag <timeInSeconds>  Process only regions that  have not experienced
 any metadata updates in the last  <timeInSeconds> seconds.
   -sleepBeforeRerun <timeInSeconds> Sleep this many seconds before checking
 if the fix worked if run with -fix
   -summary Print only summary of the tables and status.
   -metaonly Only check the state of the hbase:meta table.
   -sidelineDir <hdfs://> HDFS path to backup existing meta.
   -boundaries Verify that regions boundaries are the same between META and
 store files.
   -exclusive Abort if another hbck is exclusive or fixing.

  Datafile Repair options: (expert features, use with caution!)
```

```
   -checkCorruptHFiles     Check all Hfiles by opening them to make sure the
y are valid
   -sidelineCorruptHFiles  Quarantine corrupted HFiles.  implies -checkCorru
ptHFiles

 Replication options
   -fixReplication   Deletes replication queues for removed peers

  Metadata Repair options supported as of version 2.0: (expert features, use
 with caution!)
   -fixVersionFile   Try to fix missing hbase.version file in hdfs.
   -fixReferenceFiles  Try to offline lingering reference store files
   -fixHFileLinks  Try to offline lingering HFileLinks
   -noHdfsChecking   Don't load/check region info from HDFS. Assumes hbas
e:meta region info is good. Won't check/fix any HDFS issue, e.g. hole, orpha
n, or overlap
   -ignorePreCheckPermission  ignore filesystem permission pre-check

NOTE: Following options are NOT supported as of HBase version 2.0+.
  UNSUPPORTED Metadata Repair options: (expert features, use with caution!)
   -fix              Try to fix region assignments.  This is for backwards
compatiblity
   -fixAssignments   Try to fix region assignments.  Replaces the old -fix
   -fixMeta          Try to fix meta problems.  This assumes HDFS region inf
o is good.
   -fixHdfsHoles     Try to fix region holes in hdfs.
   -fixHdfsOrphans   Try to fix region dirs with no .regioninfo file in hdfs
   -fixTableOrphans  Try to fix table dirs with no .tableinfo file in hdfs
 (online mode only)
   -fixHdfsOverlaps  Try to fix region overlaps in hdfs.
   -maxMerge <n>     When fixing region overlaps, allow at most <n> regions
 to merge. (n=5 by default)
   -sidelineBigOverlaps  When fixing region overlaps, allow to sideline big
overlaps
   -maxOverlapsToSideline <n>  When fixing region overlaps, allow at most <
n> regions to sideline per group. (n=2 by default)
   -fixSplitParents  Try to force offline split parents to be online.
   -removeParents    Try to offline and sideline lingering parents and keep
 daughter regions.
   -fixEmptyMetaCells  Try to fix hbase:meta entries not referencing any
region (empty REGIONINFO_QUALIFIER rows)

  UNSUPPORTED Metadata Repair shortcuts
   -repair           Shortcut for -fixAssignments -fixMeta -fixHdfsHoles -
fixHdfsOrphans -fixHdfsOverlaps -fixVersionFile -sidelineBigOverlaps -fixRef
erenceFiles-fixHFileLinks
   -repairHoles      Shortcut for -fixAssignments -fixMeta -fixHdfsHoles
```

clean

After you have finished using a test or proof-of-concept cluster, the hbase clean utility can remove all HBase-related data from ZooKeeper and HDFS.

Warning: The hbase clean command destroys data. Do not run it on production clusters, or unless you are absolutely sure you want to destroy the data.

To run the hbase clean utility, use the bin/hbase clean command. Run it with no options for usage information.

```
$ bin/hbase clean

Usage: hbase clean (--cleanZk|--cleanHdfs|--cleanAll)
Options:
```

```
        --cleanZk   cleans hbase related data from zookeeper.
        --cleanHdfs cleans hbase related data from hdfs.
        --cleanAll  cleans hbase related data from both zookeeper and hdfs.
```

Use the HBase APIs for Java

You can use the Apache HBase Java API to communicate with Apache HBase. The Java API is one of the most common ways to communicate with HBase.

The following sample uses Apache HBase APIs to create a table and put a row into that table. The table name, column family name, qualifier (or column) name, and a unique ID for the row are defined. Together, these define a specific cell. Next, the table is created and the text “Hello, World!” is inserted into this cell.

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;


public class CreateAndPut {
    private static final TableName TABLE_NAME = TableName.valueOf("test_tabl
e_example");
    private static final byte[] CF_NAME = Bytes.toBytes("test_cf");
    private static final byte[] QUALIFIER = Bytes.toBytes("test_column");
    private static final byte[] ROW_ID = Bytes.toBytes("row01");
    public static void createTable(final Admin admin) throws IOException {
        if(!admin.tableExists(TABLE_NAME)) {
            TableDescriptor desc = TableDescriptorBuilder.newBuilder(TABLE_
NAME)
                    .setColumnFamily(ColumnFamilyDescriptorBuilder.of(CF_N
AME))
                    .build();
            admin.createTable(desc);
        }
    }

    public static void putRow(final Table table) throws IOException {
        table.put(new Put(ROW_ID).addColumn(CF_NAME, QUALIFIER, Bytes.toByte
s("Hello, World!")));
    }
    public static void main(String[] args) throws IOException {
        Configuration config = HBaseConfiguration.create();

        try (Connection connection = ConnectionFactory.createConnection(co
nfig); Admin admin = connection.getAdmin()) {
            createTable(admin);

            try(Table table = connection.getTable(TABLE_NAME)) {
                putRow(table);
            }
        }
    }
}
```

Related Information

HBase API reference documentation

Use the HBase REST server

You can use the Apache HBase REST server to interact with the Apache HBase. This is a very good alternative if you do not want to use the Java API. Interactions happen using URLs and the REST API. REST uses HTTP to perform various actions, and this makes it easy to interface with the operational database using a wide array of programming languages.

You can use the REST server to create, delete tables, and perform other operations that have the REST end-points. You can configure SSL for encryption between the client and the REST server. This helps you to ensure that your operations are secure during data transmission.

Using the REST server enables you access your data across different network boundaries. For example, if you have an Cloudera Operational Database Cloudera Data Hub cluster running inside a private network and don't want to expose it to your company's public network, the REST server can work as a gateway between the private and public networks.

Installing the REST Server using Cloudera Manager

You can use the HBase REST API to interact with HBase services, tables, and regions using HTTP endpoints. You must manually install the REST Server only in a Cloudera Private Cloud Base deployment. The REST Server service is automatically added to the Cloudera Data Hub cluster in a Cloudera Public Cloud deployment.

About this task

Install the REST Server using Cloudera Manager in your Cloudera Private Cloud Base deployment.

Procedure

1. Click the Clusters tab.
2. Select Clusters*HBASE*.
3. Click the Instances tab.
4. Click Add Role Instance.
5. Under HBase REST Server, click Select Hosts.
6. Select one or more hosts to serve the HBase Rest Server role. Click Continue.
7. Select the HBase Rest Server roles. Click Actions For Selected Start.

Using the REST API

The HBase REST Server exposes endpoints that provide CRUD (create, read, update, delete) operations for each HBase process, as well as tables, regions, and namespaces.

Background

For a given endpoint, the HTTP verb controls the following type of operations (create, read, update, or delete).

Note: curl Command Examples

The examples in these tables use the curl command, and follow these guidelines:

- The HTTP verb is specified using the -X parameter.
- For GET queries, the Accept header is set to text/xml, which indicates that the client (curl) expects to receive responses formatted in XML. You can set it to application/json to receive JSON responses instead.
- For PUT, POST, and DELETE queries, the Content-Type header should be set only if data is also being sent with the -d parameter. If set, it should match the format of the data being sent, to enable the REST server to deserialize the data correctly.
- If you are using a Cloudera Data Hub cluster, you must provide the basic authentication parameters in your REST query string to access the REST server end-point. For example, curl -vi -X GET \

  -H "Accept: text/xml" -u "<USER>:<MY_WORKLOAD_PASSWORD>" \

For more details about the curl command, see the documentation for the curl version that ships with your operating system.

In Cloudera, all REST queries are routed through the Apache Knox gateway. In your REST query, ensure that the hostname points to the gateway node and cdp-proxy-api endpoint as shown in these examples.

Table 1: Cluster-Wide Endpoints

Endpoint	HTTP Verb	Description	Example
/version/cluster	GET	Version of HBase running on this cluster	`curl -L -v \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/version/cluster"`
/status/cluster	GET	Cluster status	`curl -vi -X GET \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/status/cluster"`
/	GET	List of all nonsystem tables	`curl -vi -X GET \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase"`

Table 2: Namespace Endpoints

Endpoint	HTTP Verb	Description	Example
/namespaces	GET	List all namespaces.	`curl -vi -X GET \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/namespaces/"`
/namespaces/namespace	GET	Describe a specific namespace.	`curl -vi -X GET \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/namespaces/special_ns"`
/namespaces/namespace	POST	Create a new namespace.	`curl -vi -X POST \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/namespaces/special_ns"`
/namespaces/namespace/tables	GET	List all tables in a specific namespace.	`curl -vi -X GET \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/namespaces/special_ns/tables"`
/namespaces/namespace	PUT	Alter an existing namespace. Currently not used.	`curl -vi -X PUT \ -H "Accept: text/xml" \`

Table 3: Table Endpoints

Endpoint	HTTP Verb	Description	Example
/table/schema	GET	Describe the schema of the specified table.	`curl -vi -X GET \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/users/schema"`
/table/schema	POST	Create a new table, or replace an existing table's schema with the provided schema.	`curl -vi -X POST \ -H "Accept: text/xml" \ -H "Content-Type: text/xml" \ -d '<?xml version="1.0" encoding="UTF-8"?><TableSchema name="users"><ColumnSchema name="cf" /></TableSchema>' \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/users/schema"`
/table/schema	UPDATE	Update an existing table with the provided schema fragment.	`curl -vi -X PUT \ -H "Accept: text/xml" \ -H "Content-Type: text/xml" \ -d '<?xml version="1.0" encoding="UTF-8"?><TableSchema name="users"><ColumnSchema name="cf" KEEP_DELETED_CELLS="true" /></TableSchema>' \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/users/schema"`
/table/schema	DELETE	Delete the table. You must use the table/schema endpoint, not just table/	`curl -vi -X DELE`

Table 4: Endpoints for Get Operations

Endpoint	HTTP Verb	Description	Example
/table/row	GET	Get all columns of a single row. Values are Base-64 encoded. This requires the Accept request header with a type that can hold multiple columns (like xml, json or protobuf)	`curl -vi -X GET \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/users/row1"`
/table/row/column:qualifier/timestamp	GET	Get the value of a single row. Values are Base-64 encoded.	`curl -vi -X GET \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/users/row1/cf:a/1458586888395"`
/table/row/column:qualifier	GET	Get the value of a single column. Values are Base-64 encoded.	`curl -vi -X GET \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/users/row1/cf:a"`
/table/row/column:qualifier/?v=number_of_versions		Multi-Get a specified number of versions of a given cell. Values are Base-64 encoded.	`curl -vi -X GET \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/users/row1/cf:a/?v=2"`

Table 5: Endpoints for Scan Operations

Endpoint	HTTP Verb	Description	Example
/table/scanner/	PUT	Get a Scanner object. Required by all other Scan operations. Adjust the batch parameter to the number of rows the scan should return in a batch. See the next example for adding filters to your Scanner. The scanner endpoint URL is returned as the Location in the HTTP response. The other examples in this table assume that the Scanner endpoint is http://example.com:20550/users/scanner/14586907282437552207.	`curl -vi -X PUT \ -H "Accept: text/xml" \ -H "Content-Type: text/xml" \ -d '<Scanner batch="1"/>' \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/users/scanner/"`

/table/scanner/	PUT	To supply filters to the Scanner object or configure the Scanner in any other way, you can create a text file and add your filter to the file. For example, to return only rows for which keys start with u123 and use a batch size of 100:
		`<Scanner batch="100">`
		`  <filter>`
		`  { "type": "PrefixFilter", "value": "u123" }`
		`  </filter>`
		`</Scanner>`
		Pass the file to the -d argument of the curl request.
			`curl -vi -X PUT \ -H "Accept: text/xml" \ -H "Content-Type:text/xml" \ -d @filter.txt \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/users/scanner/"`

/table/scanner/scanner_id	GET	Get the next batch from the scanner. Cell values are byte-encoded. If the scanner is exhausted, HTTP status 204 is returned.	`curl -vi -X GET \ -H "Accept: text/xml" \ "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ -u "https://<gateway_node>/cdp-proxy-api/hbase/users/scanner/14586907282437552207"`
/table/scanner/scanner_id	DELETE	Deletes the scanner and frees the resources it was using.	`curl -vi -X DELETE \ -H "Accept: text/xml" \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \`

Table 6: Endpoints for Put Operations

Endpoint	HTTP Verb	Description	Example
/table/row_key/	PUT	Write a row to a table. The row, column qualifier, and value must each be Base-64 encoded. To encode a string, you can use the base64 command-line utility. To decode the string, use base 64 -d. The payload is in the --data argument, so the /users/fakerow value is a placeholder. Insert multiple rows by adding them to the <CellSet> element. You can also save the data to be inserted to a file and pass it to the -d parameter with the syntax -d @ filename.txt.	XML: `curl -vi -X PUT \ -H "Accept: text/xml" \ -H "Content-Type: text/xml" \ -d '<?xml version="1.0" encoding="UTF-8" standalone="yes"?><CellSet><Row key="cm93NQo=""><Cell column="Y2Y6ZQo=">dmFsdWU1Cg==</Cell></Row></CellSet>' \ -u "<MY_WORKLOAD_USERNAME:MY_WORKLOAD_PASSWORD>" \ "https://<gateway_node>/cdp-proxy-api/hbase/users/FAKEROW"` JSON: `curl -vi -X PUT \ -H "Accept: application/json" \ -H "Content-Type: application/json" \ -d '{"Row":[{"key":"cm93NQo=", "Cell": [{"column":"Y2Y6ZQo=", "$":"dmFsdWU1Cg=="}]}]}'' \`

Using the REST proxy API

After configuring and starting HBase on your cluster, you can use the HBase REST Proxy API to stream data into HBase, from within another application or shell script, or by using an HTTP client such as wget or curl.

The REST Proxy API is slower than the Java API and may have fewer features. This approach is simple and does not require advanced development experience to implement. However, like the Java and Thrift Proxy APIs, it uses the full write path and can cause compactions and region splits.

Specified addresses without existing data create new values. Specified addresses with existing data create new versions, overwriting an existing version if the row, column:qualifier, and timestamp all match that of the existing value.

```
curl -H "Content-Type: text/xml" http://localhost:8000/test/testrow/test:tes
tcolumn
```

The REST Proxy API does not support writing to HBase clusters that are secured using Kerberos.

For full documentation and more examples, see the REST Proxy API documentation.

Using the Apache Thrift Proxy API

The Apache Thrift library provides cross-language client-server remote procedure calls (RPCs), using Thrift bindings.

A *Thrift binding* that uses the Apache Thrift Proxy API, is a client code generated by the Apache Thrift compiler for a target language (such as Python) that allows communication between the Thrift server and clients using that client code. HBase includes an Apache Thrift Proxy API, which allows you to write HBase applications in Python, C, C++, or another language that Thrift supports. The Thrift Proxy API is slower than the Java API and may have fewer features. To use the Thrift Proxy API, you need to configure and run the HBase Thrift server on your cluster. You also need to install the Apache Thrift compiler on your development system.

Preparing a thrift server and client

Learn how to prepare a Thrift server and client before using a Thrift Proxy API.

Before you begin

Ensure that the thrift server is configured and running.

Procedure

1. Generate *Thrift bindings* for the language of your choice, using an HBase IDL file, named HBase.thrift that is included as part of HBase.
2. Copy the Thrift libraries for your language into the same directory as the generated bindings.
3. Verify that the Thrift compiler version is newer than 0.9.0 by running the `thrift -version` command. You need to find the Hbase.thrift file from the HBase node or copy it to co-locate with the Thrift compiler.
   In the following Python example, these libraries provide the thrift.transport and thrift.protocol libraries. These commands show how you might generate the *Thrift bindings* for Python and copy the libraries on a Linux system.

   ```
   mkdir HBaseThrift
   cd HBaseThrift/
   thrift -gen py /path/to/Hbase.thrift
   mv gen-py/* .
   rm -rf gen-py/
   mkdir thrift
   cp -rp ~/Downloads/thrift/lib/py/src/* ./thrift/
   ```

Results

As a result, the HBase thrift Python bindings appears as follows:

```
HbaseThrift/
|-- hbased
|   |-- constants.py
|   |-- Hbase.py
```

```
|    |-- Hbase-remote
|    |-- __init__.py
|    `-- ttypes.py
|-- __init__.py
`-- thrift
    |-- compat.py
    |-- ext
    |    |-- binary.cpp
    |    |-- binary.h
    |    |-- compact.cpp
    |    |-- compact.h
    |    |-- endian.h
    |    |-- module.cpp
    |    |-- protocol.h
    |    |-- protocol.tcc
    |    |-- types.cpp
    |    `-- types.h
    |-- __init__.py
    |-- protocol
    |    |-- __init__.py
    |    |-- TBase.py
    |    |-- TBinaryProtocol.py
    |    |-- TCompactProtocol.py
    |    |-- THeaderProtocol.py
    |    |-- TJSONProtocol.py
    |    |-- TMultiplexedProtocol.py
    |    |-- TProtocolDecorator.py
    |    `-- TProtocol.py
    |-- server
    |    |-- __init__.py
    |    |-- THttpServer.py
    |    |-- TNonblockingServer.py
    |    |-- TProcessPoolServer.py
    |    `-- TServer.py
    |-- Thrift.py
    |-- TMultiplexedProcessor.py
    |-- transport
    |    |-- __init__.py
    |    |-- sslcompat.py
    |    |-- THeaderTransport.py
    |    |-- THttpClient.py
    |    |-- TSocket.py
    |    |-- TSSLSocket.py
    |    |-- TTransport.py
    |    |-- TTwisted.py
    |    `-- TZlibTransport.py
    |-- TRecursive.py
    |-- TSCons.py
    |-- TSerialization.py
    `-- TTornado.py
```

What to do next

List of Thrift API and HBase configurations

References for the right classes and functions along with the right configurations for HBase.

Classes and functions

Transport level

`TBufferedTransport`, `TFramedTransport`, `TSaslTransport`, and `THttpClient`

Protocol level

`TBinaryProtocol` and `TCompactProtocol`

HBase Thrift configurations

HBase thrift configurations

Property	Default value (secured)	Default value (unsecured)	Description
hbase.thrift.support.proxyuser	true	true	Use this to allow proxy users on the thrift gateway, which is mainly needed for `doAs` functionality.
hbase.regionserver.thrift.framed	true	true	Use framed transport. When using the `THsHaServer` or `TNonblockingServer`, framed transport is always used irrespective of this configuration value.
hbase.regionserver.thrift.compact	true	true	Use the `TCompactProtocol` instead of the default `TBinaryProtocol`. `TCompactProtocol` is a binary protocol that is more compact than the default and typically more efficient.
hbase.regionserver.thrift.http	true	true	Use this to enable HTTP server usage on thrift, which is mainly needed for `doAs` functionality.
hbase.thrift.security.qop	auth_conf	none	If this is set, HBase Thrift Server authenticates its clients. HBase Proxy User Hosts and Groups must be configured to allow specific users to access HBase through Thrift Server.
hbase.thrift.ssl.enabled	true	false	Encrypt communication between clients and HBase Thrift Server over HTTP using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).

Related Information

Example for using THttpClient API in secure cluster
Example for using THttpClient API in unsecure cluster
Example for using TSaslClientTransport API in secure cluster without HTTP

Example for using THttpClient API in secure cluster

Refer to this example of using the THttpClient API in secure cluster.

THttpClient API in secure cluster

Let us consider that the cluster is secured with the configuration properties mentioned in the *HBase thrift configurations* table under the *Default value (secured)* column.

Before proceeding, ensure that the following applications are installed on your system.

- python 3.6.8 and python 3-devel
- pip 21.3.1
- virtualenv 20.17.1

Perform the following steps:

1. Install virtualenv using pip3.

```
pip3 install virtualenv
```

2. Create a new virtual environment named *PY3ENV*.

```
virtualenv py3env
```

3. Activate the virtual environment.

```
source py3env/bin/activate
```

4. Install the required Python packages and their specific versions. Consider you are inside the python3 virtual environment.

```
pip3 install kerberos==1.3.1 pure-sasl==0.6.2 setuptools==59.6.0 six==1.
16.0 wheel==0.37.1
```

This ensures that you have all the necessary dependencies and packages installed to proceed with your project.

```
from thrift.transport import THttpClient
from thrift.protocol import TBinaryProtocol
from hbase.Hbase import Client
from subprocess import call
import ssl
import kerberos
import os

# Get the env parameters
def get_env_params():
    # Replace with your own parameters
    hostname='your_hbase_thrift_hostname'
    cert_file="your_cert_file"
    key_file="your_key_file"
    ca_file="your_ca_file"
    key_pw='your_key_pw'
    keytab_file='your_keytab'
    principal = 'your_principal'
    return hostname,cert_file,key_file,ca_file,keytab_file,principal,key_pw

#Check if a valid Kerberos ticket is already present in the cache
def check_kerberos_ticket():
    ccache_file = os.getenv('KRB5CCNAME')
    if ccache_file:
        ccache = CCache.load_ccache(ccache_file)
        if ccache.get_principal() and not ccache.get_principal().is_anonymou
s():
            return True
    return False

# Obtain a Kerberos ticket by running kinit from keytab
def kinit(keytab_file,principal):
    call(['kinit', '-kt', keytab_file, principal])
# Authenticate with Kerberos
def kerberos_auth():
    __, krb_context = kerberos.authGSSClientInit("HTTP")
    kerberos.authGSSClientStep(krb_context, "")
    negotiate_details = kerberos.authGSSClientResponse(krb_context)
    headers = {'Authorization': 'Negotiate ' + negotiate_details, 'Content-T
ype': 'application/binary'}
    return headers
```

```
# Initializete an SSL context with certificate verification enabled
def get_ssl_context():
    ssl_context = ssl.create_default_context()
    ssl_context.load_cert_chain(certfile=cert_file,keyfile=key_file,passwo
rd=key_pw)
    ssl_context.load_verify_locations(cafile=ca_file)
    return ssl_context
if __name__ == '__main__':
    hostname,cert_file,key_file,ca_file,keytab_file,principal,key_pw=get_env
_params()
    # Check if a valid Kerberos ticket is not in the cache, then kinit.
    if not check_kerberos_ticket():
        kinit(keytab_file,principal)

# create a THttpClient instance with the SSL context and custom headers
    httpClient = THttpClient.THttpClient('https://' + hostname + ':9090/', s
sl_context=get_ssl_context())
    httpClient.setCustomHeaders(headers=kerberos_auth())

# Initialize TBinaryProtocol with THttpClient
    protocol = TBinaryProtocol.TBinaryProtocol(httpClient)

# Create HBase client
    client = Client(protocol)
# Retrieve list of HBase tables
    tables = client.getTableNames()
    print(tables)
# Close connection
    httpClient.close()
```

Related Information

List of Thrift API and HBase configurations

Example for using THttpClient API in unsecure cluster

Refer to this example of using the THttpClient API in unsecure cluster.

THttpClient API in unsecure cluster

Let us consider that the cluster is unsecured with the configuration properties mentioned in the *HBase thrift configurations* table under the *Default value (unsecured)* column.

```
from thrift.transport import THttpClient
from thrift.protocol import TBinaryProtocol
from hbase.Hbase import Client
# Replace with your own parameters
hostname = 'your_hbase_thrift_server_hostname'

# Initialize THttpClient
httpClient = THttpClient.THttpClient('http://' + hostname + ':9090/')

# Initialize TBinaryProtocol with THttpClient
protocol = TBinaryProtocol.TBinaryProtocol(httpClient)

# Create HBase client
client = Client(protocol)

# Retrieve list of HBase tables
tables = client.getTableNames()
print(tables)
```

```
# Close connection
httpClient.close()
```

Related Information

List of Thrift API and HBase configurations

Example for using TSaslClientTransport API in secure cluster without HTTP

Refer to this example of using the TSaslClientTransport API in secure cluster without HTTP.

TSaslClientTransport API in secure cluster without HTTP

If you do not use `THttpClient` and want to use `TSaslClientTransport` for legacy compatibility reasons, ensure that you set hbase.regionserver.thrift.http property to false. The other settings could be same as the configuration properties mentioned in the *HBase thrift configurations* table under the *Default value (secured)* column.

```
from thrift.transport import TSocket
from thrift.transport import TTransport
from thrift.protocol import TBinaryProtocol
from thrift.protocol import TCompactProtocol
from hbase import Hbase

'''
Assume you already kinit the hbase principal, or you can use the function
 in example-1 to kinit.
'''
# Replace with your own parameters
thrift_host = 'your_hbase_thrift_server_hostname'
thrift_port = 9090

# Initialize TSocket and TTransport
socket = TSocket.TSocket(thrift_host, thrift_port)
transport=TTransport.TSaslClientTransport(socket,host=thrift_host,service='
hbase',mechanism='GSSAPI')

# Initialize TCompactProtocol with TTransport
protocol = TCompactProtocol.TCompactProtocol(transport)

# Create HBase client
client = Hbase.Client(protocol)

# Open connection and retrieve list of HBase tables
transport.open()
tables = client.getTableNames()
print(tables)

# Close connection
transport.close()
```

Cloudera recommends you to use the HTTP options (Example-1 and Example-2). You can consider the Example-3 for legacy compatibility issues where some old applications might not rewrite the codes. This is because Hue is using HTTP mode to interact with HBase thrift, and if you disable the HTTP mode, Hue might not work properly with HBase.

Known bugs while using TSaslClientTransport with Kerberos enabled Cloudera versions

Upstream JIRA HBASE-21652, where a bug is introduced related to Kerberos principal handling. The affected versions are Cloudera 7.1.6 and earlier. The versions containing the fix are 7.1.7, 7.2.11, and later.

Related Information

List of Thrift API and HBase configurations

Using Apache HBase Hive integration

You can use the HBase Hive integration to create HBase tables and modify Apache HBase tables from Apache Hive.

HBase Hive integration enables you to READ and WRITE to existing HBase tables. Before you can access your data stored in HBase using Hive, ensure that you have completed the configuration that enables Apache Hive to interact with Apache HBase.

Configure Hive to use with HBase

To use Hive with HBase, you must add the HBase service as a dependency to the Hive service in Cloudera Manager.

About this task

The following steps are applicable when both HBase and Hive are in the same cluster.

Procedure

1. Go to Cloudera Manager.
2. Go to the Hive service.
3. Click the Configuration tab.
4. Select the intended HBase service. For example, HBASE-1 as dependency under HBase Service.
5. Click Save Changes.

   Repeat steps 2 to 5 for the HBase on Tez service.
6. Restart the Hive service.

   Restart the Hive on Tez service, when applicable.

Using HBase Hive integration

After you have configured HBase Hive integration, you can create an HBase table from Hive, and access that table from HBase.

About this task

Procedure

1. From the Hive shell, create an HBase table:

```
CREATE EXTERNAL TABLE hbase_hive_table (key int, value string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,cf1:val")
```

```
TBLPROPERTIES ("hbase.table.name" = "hbase_hive_table", "hbase.mapred.o
utput.outputtable" = "hbase_hive_table");
```

The hbase.columns.mapping property is mandatory. The hbase.table.name property is optional. The hbase.mapred .output.outputtable property is optional; it is needed, if you plan to insert data to the table.

If hbase.columns.mapping values contain special characters like '#' or '%', they have to be encoded because the values are used to form the URI for Ranger based authentication. To enable URL encoding, set the hive.sec urity.hbase.urlencode.authorization.uri property to "true" in the Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml and restart the Hive on Tez service. Also, update the corresponding Ranger policies for the table so that they are in URL encoded format.

2. From the HBase shell, access the hbase_hive_table:

```
$ hbase shell
hbase(main):001:0> list 'hbase_hive_table'

 1 row(s) in 0.0530 seconds
hbase(main):002:0> describe 'hbase_hive_table'
Table hbase_hive_table is ENABLED
hbase_hive_table COLUMN FAMILIES DESCRIPTION{NAME => 'cf', DATA_BLOCK_ENCO
DING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS
 => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', K
EEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false',
 BLOCKCACHE => 'true'} 1 row(s) in 0.2860 seconds

hbase(main):003:0> scan 'hbase_hive_table'
ROW           COLUMN+CELL

0 row(s) in 0.0060 seconds
```

3. Insert the data into the HBase table through Hive:

```
INSERT OVERWRITE TABLE HBASE_HIVE_TABLE values (98, 'val_98');
```

4. From the HBase shell, verify that the data got loaded:

```
hbase(main):009:0> scan "hbase_hive_table"
ROW                        COLUMN+CELL

 98                        column=cf1:val, timestamp=1267737987733, valu
e=val_98
1 row(s) in 0.0110 seconds
```

5. From Hive, query the HBase data to view the data that is inserted in the hbase_hive_table:

```
hive> select * from HBASE_HIVE_TABLE;
Total MapReduce jobs = 1
Launching Job 1 out of 1
...
OK
98 val_98
Time taken: 4.582 seconds
```

Example

Use the following steps to access the existing HBase table through Hive.

- You can access the existing HBase table through Hive using the CREATE EXTERNAL TABLE:

```
CREATE EXTERNAL TABLE hbase_table_2(key int, value string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEP
ROPERTIES
```

```
("hbase.columns.mapping" = ":key,cf1:val")
TBLPROPERTIES("hbase.table.name" = "some_existing_table",
"hbase.mapred.output.outputtable" = "some_existing_table");
```

- You can use different type of column mapping to map the HBase columns to Hive:
  - Multiple Columns and Families To define four columns, the first being the rowkey: “:key,cf:a,cf:b,cf:c”
  - Hive MAP to HBase Column Family When the Hive data type is a Map, a column family with no qualifier might be used. This will use the keys of the Map as the column qualifier in HBase: “cf:”
  - Hive MAP to HBase Column Prefix When the Hive data type is a Map, a prefix for the column qualifier can be provided which will be prepended to the Map keys: “cf:prefix_.*”

    Note: The prefix is removed from the column qualifier as compared to the key in the Hive Map. For example, for the above column mapping, a column of “cf:prefix_a” would result in a key in the Map of “a”.
- You can also define composite row keys. Composite row keys use multiple Hive columns to generate the HBase row key.
  - Simple Composite Row Keys. A Hive column with a datatype of Struct will automatically concatenate all elements in the struct with the termination character specified in the DDL.
  - Complex Composite Row Keys and HBaseKeyFactory Custom logic can be implemented by writing Java code to implement a KeyFactory and provide it to the DDL using the table property key “hbase.composite.key.factory”.

Using the HBase-Spark connector

You can use HBase-Spark connector on your secure cluster to perform READ and WRITE operations. The HBase-Spark Connector bridges the gap between the simple HBase Key Value store and complex relational SQL queries and enables users to perform complex data analytics on top of HBase using Spark.

An HBase DataFrame is a standard Spark DataFrame, and is able to interact with any other data sources such as Hive, ORC, Parquet, or JSON.

The following blog post provides additional information about Spark and HBase usage in Cloudera Public Cloud: How to use Apache Spark with Cloudera Operational Database Experience.

Configuring HBase-Spark connector using Cloudera Manager when HBase and Spark are on the same cluster

Learn how to configure the HBase-Spark connector when both the HBase and Spark are on the same cluster.

About this task

- Ensure that every Spark node has the HBase Master, Region Server, or Gateway role assigned to it. If no HBase role is assigned to a Spark node, add the HBase Gateway role to it, which ensures that the HBase configuration files are available on the Spark node. For more information, see Managing Roles.

Note: If an application needs to interact with other secure Hadoop filesystems, their URIs need to be explicitly provided to Spark at launch time.

Spark configuration property: spark.kerberos.access.hadoopFileSystems

A comma-separated list of secure Hadoop filesystems your Spark application is going to access. For example:

```
spark.kerberos.access.hadoopFileSystems=hdfs://nn1.com:8032,hdfs://n
n2.com:8032, abfs://test1@example1.dfs.core.windows.net,abfs://test2
@example2.dfs.core.windows.net
```

For more information see *Spark 3 documentation*.

Procedure

1. Go to the Spark service.
2. Click the Configuration tab.
3. Ensure that the HBase service is selected in Spark Service as a dependency.
4. Select Scope Gateway .
5. Select Category Advanced .
6. Locate the spark-defaults.conf.

   Locate the Spark 3 Client Advanced Configuration Snippet (Safety Valve) for spark3-conf/spark-defaults.conf property or search for it by typing its name in the Search box.
7. Add the required properties to ensure that all required Phoenix and HBase platform dependencies are available on the classpath for the Spark executors and drivers.
   a) Upload all necessary jar files to the distributed filesystem, for example HDFS (it can be GS, ABFS, or S3A). If the CDH version is different on the remote HBase cluster, run the `hbase mapredcp` command on the HBase cluster and copy them to /path/hbase_jars_common location so that the Spark applications can use them.
      - Spark3 related files:

        ```
        hdfs dfs -mkdir /path/hbase_jars_spark3
        ```
      - Common files for Spark:

        ```
        hdfs dfs -mkdir /path/hbase_jars_common
        hdfs dfs -put `hbase mapredcp | tr : " "` /path/hbase_jars_common
        ```
   b) Download the /etc/hbase/conf/hbase-site.xml from the remote HBase cluster and update the truststore password in the hbase-site.xml file with the Data Engineering DataHub truststore password.
   c) Create the hbase-site.xml.jar file. The hbase-site.xml is added to the classpath with the spark.jars parameter because it is part of the jar file's root path.

      ```
      jar cf hbase-site.xml.jar hbase-site.xml
      hdfs dfs -put hbase-site.xml.jar /path/hbase_jars_common
      ```
   d) Download the truststore JKS file from the remote HBase cluster.
   e) Upload the Spark related files:

      ```
      hdfs dfs -put /opt/cloudera/parcels/CDH/lib/hbase_connectors_for_spark3/
      lib/hbase-spark3.jar /path/hbase_jars_spark3
      hdfs dfs -put /opt/cloudera/parcels/CDH/lib/hbase_connectors_for_spark3/
      lib/hbase-spark3-protocol-shaded.jar /path/hbase_jars_spark3
      ```
   f) Add all the Spark version related files and the `hbase mapredcp` files to the spark.jars parameter:
      -
        ```
        spark.jars=hdfs:///path/hbase_jars_common/hbase-site.xml.jar,hdfs://
        /path/hbase_jars_spark3/hbase-spark3.jar,hdfs:///path/hbase_jars_spa
        ```

```
rk3/hbase-spark3-protocol-shaded.jar,/path/hbase_jars_common(other c
ommon files)...
```

8. Enter a Reason for change, and then click Save Changes to commit the changes.
9. Restart the role and service when Cloudera Manager prompts you to restart.

   Perform the following steps while using HBase RegionServer:

   Edit the HBase RegionServer configuration for running Spark Filter. Spark Filter is used when Spark SQL Where clauses are in use.

   a. In Cloudera Manager, select the HBase service.
   b. Click the Configuration tab.
   c. Search for regionserver environment.
   d. Find the RegionServer Environment Advanced Configuration Snippet (Safety Valve).
   e. Click the plus icon to add the following property:

      Key: HBASE_CLASSPATH

      Value:

      ```
      /opt/cloudera/parcels/CDH/lib/hbase_connectors_for_spark3/lib/hbase-spar
      k3.jar:/opt/cloudera/parcels/CDH/lib/hbase_connectors_for_spark3/lib/hba
      se-spark3-protocol-shaded.jar:/opt/cloudera/parcels/CDH/lib/hbase_connec
      tors_for_spark3/lib/scala-library.jar
      ```
   f. Ensure that the listed jars have the correct version number in their name.
   g. Click Save Changes.
   h. Restart the Region Server.

What to do next

Build a Spark application using the dependencies that you provide when you run your application. If you follow the previous instructions, Cloudera Manager automatically configures the connector for Spark. If you have not:

- Consider the following example for the Spark application:

  ```
  spark-shell --conf spark.jars=hdfs:///path/hbase_jars_common/hbase-site.
  xml.jar,hdfs:///path/hbase_jars_spark3/hbase-spark3-protocol-shaded.jar,
  hdfs:///path/hbase_jars_spark3/hbase-spark3.jar,hdfs:///path/hbase_jars_
  common/hbase-shaded-mapreduce-***VERSION NUMBER***.jar,hdfs:///path/hbas
  e_jars_common/opentelemetry-api-***VERSION NUMBER***.jar,hdfs:///path/hb
  ase_jars_common/opentelemetry-context-***VERSION NUMBER***.jar
  ```

Related Information

Example: Using the HBase-Spark connector

Configure Phoenix-Spark connector using Cloudera Manager

Spark 3 documentation

Configuring HBase-Spark connector using Cloudera Manager when HBase is on a remote cluster

Learn how to configure the HBase-Spark connector when HBase is residing on a remote cluster.

About this task

Note: If an application needs to interact with other secure Hadoop filesystems, their URIs need to be explicitly provided to Spark at launch time.

- Spark configuration property: spark.kerberos.access.hadoopFileSystems

  A comma-separated list of secure Hadoop filesystems your Spark application is going to access. For example:

  ```
  spark.kerberos.access.hadoopFileSystems=hdfs://nn1.com:8032,hdfs://n
  n2.com:8032, abfs://test1@example1.dfs.core.windows.net,abfs://test2
  @example2.dfs.core.windows.net
  ```

  For more information see *Spark 3 documentation*.

Procedure

1. Go to the Spark service.
2. Click the Configuration tab.
3. Select Scope Gateway .
4. Select Category Advanced .
5. Locate the spark-defaults.conf property or search for it by typing its name in the Search box.
   - Locate the Spark 3 Client Advanced Configuration Snippet (Safety Valve) for spark3-conf/spark-defaults.conf property or search for it by typing its name in the Search box.

6. Add the required properties to ensure that all required Phoenix and HBase platform dependencies are available on the classpath for the Spark executors and drivers.

   Note:

   - Upload all necessary jar files to the distributed filesystem, for example HDFS (it can be GS, ABFS, or S3A). If the CDH version is different on the remote HBase cluster, run the `hbase mapredcp` command on the HBase cluster and copy them to /path/hbase_jars_common location so that the Spark applications can use them.
     - Spark3 related files:

       ```
       hdfs dfs -mkdir /path/hbase_jars_spark3
       ```

     - Common files for Spark:

       ```
       hdfs dfs -mkdir /path/hbase_jars_common
       hdfs dfs -put `hbase mapredcp | tr : " "` /path/hbase_jars_c
       ommon
       ```

   - Download the /etc/hbase/conf/hbase-site.xml from the remote HBase cluster. Create the hbase-site.xml.jar file. The hbase-site.xml is added to the classpath with the spark.jars parameter because it is part of the jar file's root path.

     ```
     jar cf hbase-site.xml.jar hbase-site.xml
     hdfs dfs -put hbase-site.xml.jar /path/hbase_jars_common
     ```

   - Upload the Spark related files:

     ```
     hdfs dfs -put /opt/cloudera/parcels/CDH/lib/hbase_connectors_for
     _spark3/lib/hbase-spark3-***VERSION /path/hbase_jars_spark3
     hdfs dfs -put /opt/cloudera/parcels/CDH/lib/hbase_connectors_for
     _spark3/lib/hbase-spark3-protocol-shaded-***VERSION NUMBER***.ja
     r /path/hbase_jars_spark3
     ```

   - Add all the Spark version related files and the `hbase mapredcp` files to the spark.jars parameter:
     - ```
       spark.jars=hdfs:///path/hbase_jars_common/hbase-site.xml.jar,hdf
       s:///path/hbase_jars_spark3/hbase-spark3.jar,hdfs:///path/hbase_
       jars_spark3/hbase-spark3-protocol-shaded.jar,/path/hbase_jars_co
       mmon(other common files)...
       ```

7. Enter a Reason for change, and then click Save Changes to commit the changes.
8. Restart the role and service when Cloudera Manager prompts you to restart.

   Perform the following steps while using HBase RegionServer:

   Edit the HBase RegionServer configuration for running Spark Filter. Spark Filter is used when Spark SQL Where clauses are in use.

   a. In Cloudera Manager, select the HBase service.
   b. Click the Configuration tab.
   c. Search for regionserver environment.
   d. Find the RegionServer Environment Advanced Configuration Snippet (Safety Valve).
   e. Click the plus icon to add the following property:

      Key: HBASE_CLASSPATH

      Value:

      ```
      /opt/cloudera/parcels/CDH/lib/hbase_connectors_for_spark3/lib/hbase-spar
      k3.jar:/opt/cloudera/parcels/CDH/lib/hbase_connectors_for_spark3/lib/hba
      ```
      ```

```
se-spark3-protocol-shaded.jar:/opt/cloudera/parcels/CDH/lib/hbase_connec
tors_for_spark3/lib/scala-library.jar
```

f. Ensure that the listed jars have the correct version number in their name.
g. Click Save Changes.
h. Restart the Region Server.

What to do next

Build a Spark application using the dependencies that you provide when you run your application. If you follow the previous instructions, Cloudera Manager automatically configures the connector for Spark. If you have not:

- Consider the following example while using a Spark application:

```
spark3-shell --conf spark.jars=hdfs:///path/hbase_jars_common/hbase-site
.xml.jar,hdfs:///path/hbase_jars_spark3/hbase-spark3-protocol-shaded.jar
,hdfs:///path/hbase_jars_spark3/hbase-spark3.jar,hdfs:///path/hbase_jars
_common/hbase-shaded-mapreduce-***VERSION NUMBER***.jar,hdfs:///path/hba
se_jars_common/opentelemetry-api-***VERSION NUMBER***.jar,hdfs:///path/h
base_jars_common/opentelemetry-context-***VERSION NUMBER***.jar
```

Related Information

Spark 3 documentation

Example: Using the HBase-Spark connector

Learn how to use the HBase-Spark connector by following an example scenario.

Note: If an application needs to interact with other secure Hadoop filesystems, their URIs need to be explicitly provided to Spark at launch time.

- Spark configuration property: spark.kerberos.access.hadoopFileSystems

  A comma-separated list of secure Hadoop filesystems your Spark application is going to access. For example:

  ```
  spark.kerberos.access.hadoopFileSystems=hdfs://nn1.com:8032,hdfs://n
  n2.com:8032, abfs://test1@example1.dfs.core.windows.net,abfs://test2
  @example2.dfs.core.windows.net
  ```

  For more information see *Spark 3 documentation*.

If you follow the instructions mentioned in *Configure HBase-Spark connector using Cloudera Manager* topic, Cloudera Manager automatically configures the connector for Spark. If you have not, add the following parameters to the command line while running `spark-submit`, `spark-shell`, or `pyspark` commands.

- ```
  --conf spark.jars=/path/to/hbase-site.xml.jar,/opt/cloudera/parcels/CDH/
  lib/hbase_connectors_for_spark3/lib/hbase-spark3.jar,/opt/cloudera/parce
  ls/CDH/lib/hbase_connectors_for_spark3/lib/hbase-spark3-protocol-shaded.
  jar,`hbase mapredcp | tr : ,`
  ```

You can use the following command to create the hbase-site.xml.jar file. The hbase-site.xml is added to the classpath with the spark.jars parameter because it is part of the jar file's root path.

```
jar cf hbase-site.xml.jar hbase-site.xml
```
```

Schema

In this example we want to store personal data in an HBase table. We want to store name, email address, birth date and height as a floating point number. The contact information (email) is stored in the c column family and personal information (birth date, height) is stored in the p column family. The key in HBase table will be the name attribute.

	Spark	HBase
Type/Table	Person	person
Name	name: String	key
Email address	email: String	c:email
Birth date	birthDate: Date	p:birthDate
Height	height: Float	p:height

Create HBase table

Use the following command to create the HBase table:

```
shell> create 'person', 'p', 'c'
```

Insert data (Scala)

Use the following spark code in spark-shell to insert data into our HBase table:

```
val sql = spark.sqlContext

import java.sql.Date

case class Person(name: String,
                  email: String,
                  birthDate: Date,
                  height: Float)
var personDS = Seq(
 Person("alice", "alice@alice.com", Date.valueOf("2000-01-01"), 4.5f),
 Person("bob", "bob@bob.com", Date.valueOf("2001-10-17"), 5.1f)
).toDS

if (true) {
  personDS.write.format("org.apache.hadoop.hbase.spark")
   .option("hbase.columns.mapping",
     "name STRING :key, email STRING c:email, " +
       "birthDate DATE p:birthDate, height FLOAT p:height")
   .option("hbase.table", "person")
   .option("hbase.spark.use.hbasecontext", false)
   .save()
}
```

Insert data (Python)

Use the following spark code in pyspark to insert data into our HBase table:

```
from datetime import datetime
from pyspark.sql.types import StructType, StructField, StringType, DateType,
 FloatType

data = [("alice","alice@alice.com", datetime.strptime("2000-01-01",'%Y-%m-
%d'), 4.5),
    ("bob","bob@bob.com", datetime.strptime("2001-10-17",'%Y-%m-%d'), 5.1)
  ]
```

```
schema = StructType([ \
    StructField("name",StringType(),True), \
    StructField("email",StringType(),True), \
    StructField("birthDate", DateType(),True), \
    StructField("height", FloatType(), True)
  ])

personDS = spark.createDataFrame(data=data,schema=schema)

personDS.write.format("org.apache.hadoop.hbase.spark").option("hbase.colu
mns.mapping", "name STRING :key, email STRING c:email, birthDate DATE p:birt
hDate, height FLOAT p:height").option("hbase.table", "person").option("hbase
.spark.use.hbasecontext", False).save()
```

Scan data

The previously inserted data can be tested with a simple scan:

```
shell> scan ‘person’
ROW  COLUMN+CELL
 alice  column=c:email, timestamp=1568723598292, value=alice@alice.com
 alice  column=p:birthDate, timestamp=1568723598292, value=\x00\x00\x00\xDCl
\x87 \x00
 alice  column=p:height, timestamp=1568723598292, value=@\x90\x00\x00
 bob  column=c:email, timestamp=1568723598521, value=bob@bob.com
 bob  column=p:birthDate, timestamp=1568723598521, value=\x00\x00\x00\xE9\
x99u\x95\x80
 bob  column=p:height, timestamp=1568723598521, value=@\xA333
2 row(s)
```

Read data back (Scala)

Use the following snippet in spark-shell to read the data back:

```
val sql = spark.sqlContext
var df = spark.emptyDataFrame

if (true) {
  df = sql.read.format("org.apache.hadoop.hbase.spark")
 .option("hbase.columns.mapping",
   "name STRING :key, email STRING c:email, " +
     "birthDate DATE p:birthDate, height FLOAT p:height")
 .option("hbase.table", "person")
 .option("hbase.spark.use.hbasecontext", false)
 .load()
 }

df.createOrReplaceTempView("personView")val results = sql.sql("SELECT * FROM
 personView WHERE name = 'alice'")
results.show()
```

The result of this snippet is the following Data Frame:

```
+-----+------+---------------+----------+
| name|height|          email| birthDate|
+-----+------+---------------+----------+
|alice|   4.5|alice@alice.com|2000-01-01|
+-----+------+---------------+----------+
```

Read data back (Python)

Use the following snippet in pyspark to read the data back:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Test HBase Connector from Python").ge
tOrCreate()
df = spark.read.format("org.apache.hadoop.hbase.spark").option("hbase.colum
ns.mapping", "name STRING :key, email STRING c:email, birthDate DATE p:birth
Date, height FLOAT p:height").option("hbase.table", "person").option("hbase.
spark.use.hbasecontext", False).load()
df.createOrReplaceTempView("personView")
results = spark.sql("SELECT * FROM personView WHERE name = 'alice'")
results.show()
```

Test spark-submit

Use the following snippet to test spark-submit commands in Spark cluster mode.

pyspark_app.py:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Test HBase Connector from Python").ge
tOrCreate()
df = spark.read.format("org.apache.hadoop.hbase.spark").option("hbase.colum
ns.mapping", "name STRING :key, email STRING c:email, birthDate DATE p:birth
Date, height FLOAT p:height").option("hbase.table", "person").option("hbase.
spark.use.hbasecontext", False).load()
df.createOrReplaceTempView("personView")
results = spark.sql("SELECT * FROM personView WHERE name = 'alice'")
results.show()
spark.stop()
```

Test commands:

- Spark3:

  ```
  spark3-submit --deploy-mode cluster --conf spark.jars=/path/to/hbase-sit
  e.xml.jar,/opt/cloudera/parcels/CDH/lib/hbase_connectors_for_spark3/lib/
  hbase-spark3.jar,/opt/cloudera/parcels/CDH/lib/hbase_connectors_for_spar
  k3/lib/hbase-spark3-protocol-shaded.jar,`hbase mapredcp | tr : ,` pyspar
  k_app.py
  ```

SparkSQL or DataFrames

You need to define the Catalog for the schema mapping between the HBase and Spark tables, prepare the data, populate the HBase table, and then load the HBase DataFrame. Afterward, you can run an integrated query and access records in HBase tables with SQL query. The following illustrates the basic procedure. For more example, see the Apache upstream documentation, *SparkSQL/DataFrames*.

Define catalog

```
def catalog = s"""{
       |"table":{"namespace":"default", "name":"table1"},
       |"rowkey":"key",
       |"columns":{
         |"col0":{"cf":"rowkey", "col":"key", "type":"string"},
         |"col1":{"cf":"cf1", "col":"col1", "type":"boolean"},
         |"col2":{"cf":"cf2", "col":"col2", "type":"double"},
         |"col3":{"cf":"cf3", "col":"col3", "type":"float"},
         |"col4":{"cf":"cf4", "col":"col4", "type":"int"},
         |"col5":{"cf":"cf5", "col":"col5", "type":"bigint"},
```

```
          |"col6":{"cf":"cf6", "col":"col6", "type":"smallint"},
          |"col7":{"cf":"cf7", "col":"col7", "type":"string"},
          |"col8":{"cf":"cf8", "col":"col8", "type":"tinyint"}
        |}
      |}""".stripMargin
```

Catalog defines a mapping between HBase and Spark tables. There are two critical parts of this catalog. One is the rowkey definition and the other is the mapping between the table column in Spark and the column family and column qualifier in HBase. The above defines a schema for an HBase table with the name *TABLE1*, *ROWKEY* as key, and several columns (*COL1* - *COL8*). Note that the *ROWKEY* also has to be defined in detail as a column (*COL0*), which has a specific `cf (rowkey)`.

Save the DataFrame

```
case class HBaseRecord(
   col0: String,
   col1: Boolean,
   col2: Double,
   col3: Float,
   col4: Int,
   col5: Long,
   col6: Short,
   col7: String,
   col8: Byte)
object HBaseRecord
{

   def apply(i: Int, t: String): HBaseRecord = {
      val s = s"""row${"%03d".format(i)}"""
      HBaseRecord(s,
      i % 2 == 0,
      i.toDouble,
      i.toFloat,
      i,
      i.toLong,
      i.toShort,
      s"String$i: $t",
      i.toByte)
  }
}

val data = (0 to 255).map { i =>  HBaseRecord(i, "extra")}

sc.parallelize(data).toDF.write.options(
 Map(HBaseTableCatalog.tableCatalog -> catalog, HBaseTableCatalog.newTable -
> "5"))
 .format("org.apache.hadoop.hbase.spark ")
 .save()
```

The data represents a local Scala collection that has 256 HBaseRecord objects. The `sc.parallelize(data)` function distributes data to form an RDD. The `toDF` returns a DataFrame. The write function returns a `DataFrameWriter` used to write the DataFrame to external storage systems (for example, HBase here). In the DataFrame with a specified schema catalog, the save function creates an HBase table with 5 regions and saves the DataFrame inside.

Load the DataFrame

```
def withCatalog(cat: String): DataFrame = {
  sqlContext
  .read
  .options(Map(HBaseTableCatalog.tableCatalog->cat))
  .format("org.apache.hadoop.hbase.spark")
```

```
  .load()
}
val df = withCatalog(catalog)
```

In `withCatalog` function, *SQLCONTEXT* is a variable of `SQLContext`, which is the entry point for working with structured data (rows and columns) in Spark. The read function returns a `DataFrameReader` that can be used to read data in as a DataFrame. The option function adds input options for the underlying data source to the `DataFrameReader`, and the format function specifies the input data source format for the `DataFrameReader`. The `load()` function loads input in as a DataFrame. The data frame `df` returned by `withCatalog` function could be used to access the HBase table, such as 4.4 and 4.5.

Language integrated query

```
val s = df.filter(($"col0" <= "row050" && $"col0" > "row040") ||
  $"col0" === "row005" ||
  $"col0" <= "row005")
  .select("col0", "col1", "col4")
s.show
```

The DataFrame can do various operations, such as join, sort, select, filter, orderBy and so on. The `df.filter` function above filters rows using the given SQL expression. The select function selects a set of columns: *COL0*, *COL1* and *COL4*.

SQL query

```
df.registerTempTable("table1")
sqlContext.sql("select count(col1) from table1").show
```

The `registerTempTable` function registers `df` DataFrame as a temporary table using the table name *TABLE1*. The lifetime of this temporary table is tied to the `SQLContext` that was used to create the data frame `df`. The `sqlContext.sql` function allows the user to execute SQL queries.

Related Information

Configuring HBase-Spark connector using Cloudera Manager when HBase and Spark are on the same cluster

SparkSQL/DataFrames

Spark 3 documentation

Use the Hue HBase app

Hue is a web-based interactive query editor that enables you to interact with data warehouses. You can use the HBase Browser application in Hue to create and browse HBase tables.

The HBase Hue app enables you to insert a new row or bulk upload CSV files, TSV files, and type data into your table. You can also insert columns into your row. If you need more control or data about your cell, you can use the full editor to edit a cell.

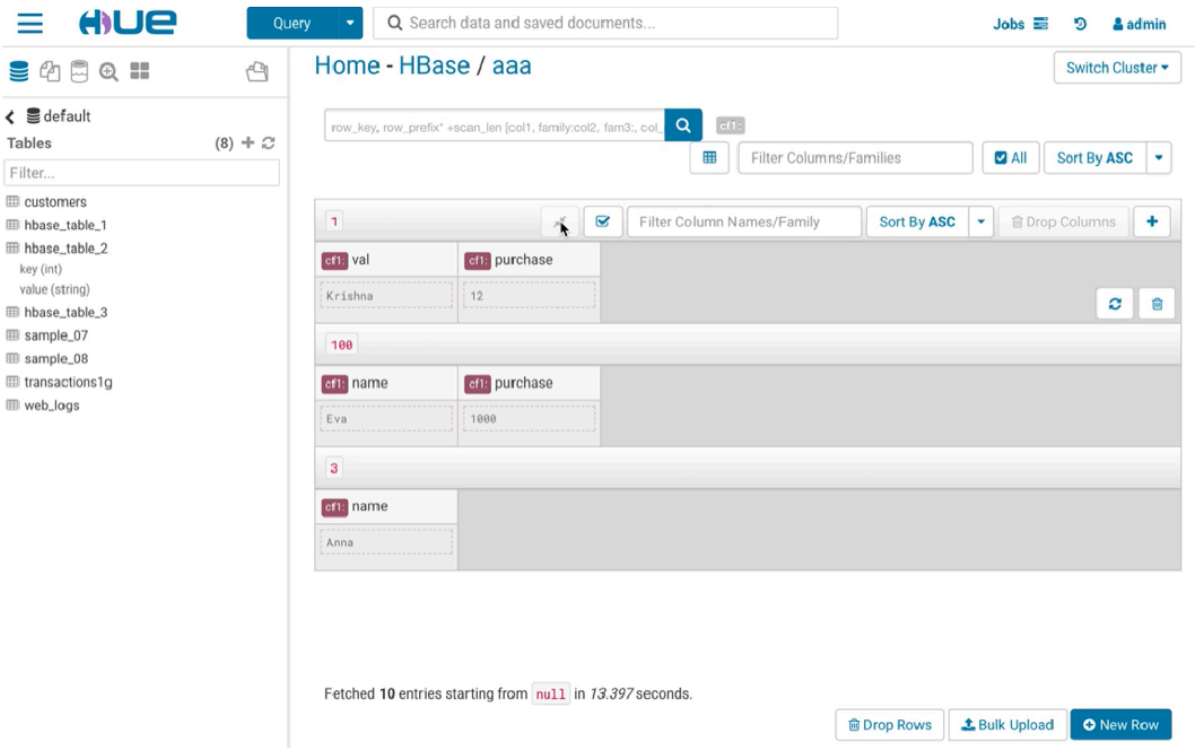

If you are using the HBase Thrift interface, Hue fits in between the Thrift Server and the HBase client, and the Thrift Server assumes that all HBase operations come from the hue user and not the client. To ensure that users in Hue are only allowed to perform HBase operations assigned to their own credentials, and not those of the hue user, you must enable doAs Impersonation for the HBase Browser Application.

Related Information

Hue

Configure the HBase thrift server role

You must configure the Thrift Server Role to access certain features such as the Hue HBase browser.

About this task

The Thrift Server role is not added by default when you install HBase, but it is required before you can use certain other features such as the Hue HBase browser. To add the Thrift Server role:

Procedure

1. Go to the HBase service.
2. Click the Instances tab.
3. Click the Add Role Instances button.
4. Select the host(s) where you want to add the Thrift Server role (you only need one for Hue) and click Continue. The Thrift Server role should appear in the instances list for the HBase server.
5. Select the Thrift Server role instance.
6. Select Actions for Selected > Start.

Cloudera Runtime 7.3.1

Administering Hue

Date published: 2020-08-10
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Reference architecture 5

Hue configuration files 6

Hue configurations in Cloudera Runtime 7

Hue Advanced Configuration Snippet 9
- Advanced safety valves 9
- Environment safety valves 9

Set up SQL AI Assistant 10
- Prerequisites for configuring Hue SQL AI Assistant 10
  - Open approach for passing a token to configure Hue SQL AI Assistant 10
- Configure SQL AI Assistant using the Microsoft Azure OpenAI service 11
- Configure SQL AI Assistant using the Amazon Bedrock Service 12
- Configure SQL AI Assistant using the OpenAI platform 12
- Complete list of model-related configurations for setting up the Hue SQL AI Assistant 13

Hue logs 14
- Standard stream logs 14
- Hue service Django logs 15
- Enabling DEBUG logging for Hue logs 15
- Enabling httpd log rotation for Hue 16

Hue supported browsers 17

Enabling cache-control HTTP headers when using Hue 17

Setting up a Hue service account with a custom name 18
- To validate newly configured custom user and group 19

Options to restart the Hue service 20

Customizing the Hue web interface 21
- Adding a custom banner in Hue 21
- Changing the page logo in Hue 21
- Adding a splash screen in Hue 23
- Setting the cache timeout 24
- Enabling or disabling anonymous usage date collection 24

Disabling the share option in Hue..... 25

Enabling Hue applications with Cloudera Manager.....25

Running shell commands..... 26

Downloading and exporting data from Hue..... 28

Enabling a multi-threaded environment for Hue.....29

Adding Query Processor service to a cluster.....29

Removing Query Processor service from cluster..... 31

Enabling the Query Processor service in Hue.....31

Adding Query Processor admin users and groups.....32

Cleaning up old queries..... 32

Downloading debug bundles.....33

Configuring Hue to handle HS2 failover..... 34

Enabling Spark 3 engine in Hue.....35

Using Hue scripts.....36

Configurations for submitting a Hive query to a dedicated queue.....37

Configuring timezone for Hue..... 38

Reference architecture

Hue server can support approximately 25 concurrent users, depending on what tasks the users are performing. Most scaling issues occur as a result of users performing resource-intensive operations and not from the number of users. For example, large downloads of query results can impact resource availability for the other users who are using the same Hue instance during the download operation. During that time, the users can experience slow performance. Another common cause of noticeable performance changes are slow RPC calls between Hue and another service. When this happens, queries may appear to suddenly "stop responding" after they are submitted.

As a guide, 2 Hue servers can support up to:

- 100 unique users per week
- 50 users per hour at peak times executing up to 100 queries

A typical setup is 2 Hue servers.

General guidelines

- Deploy a load balancer in front of Hue.
- Use a production-quality database.
- Ensure that other services, such as Impala, Hive, and Oozie, are healthy and not impacted by too few resources. If these services are hanging, it adversely affects Hue performance.
- Consider moving workloads that are subject to SLAs (service-level agreements) or considered "noisy neighbors" to their own compute cluster. Noisy neighbors are workloads that use the majority of available resources and cause performance issues.
- Limit the number of rows that are returned for queries.

  One way to limit the number of rows returned is to specify a value for the download_row_limit configuration property for the Hue Beeswax application. This property can be set in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini property in Cloudera Manager:

  1. In Cloudera Manager, click Hue Configuration , and enter Hue Service Advanced Configuration Snippet in the search text box.
  2. In the text box for the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini, add the following configuration information:

     ```
     [beeswax]
     download_row_limit=NUMBER_OF_ROWS
     ```

3. Click Save Changes and click the restart icon at the top of the page to restart the Hue service:

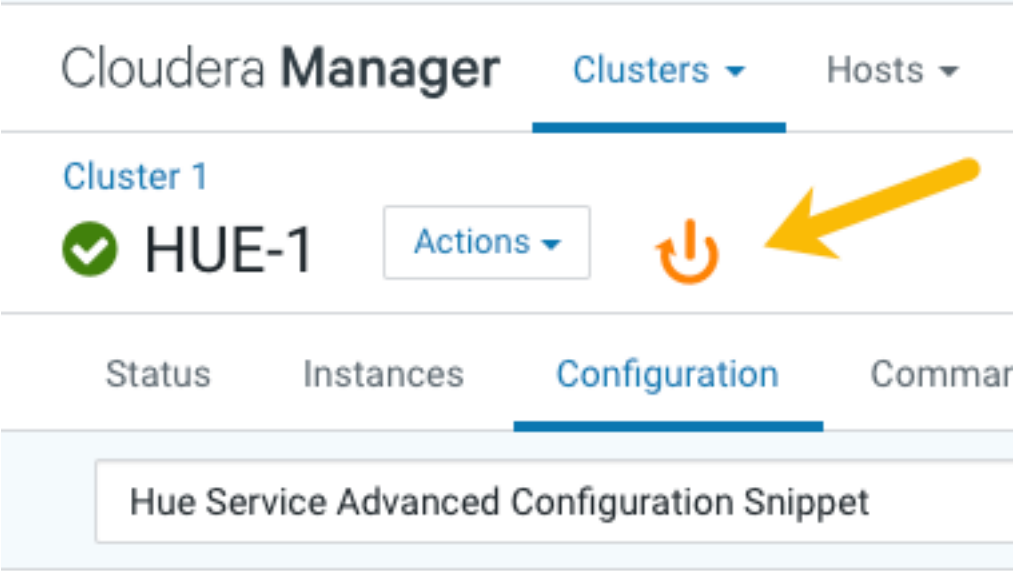

Hue configuration files

Hue roles are configured with the following three configuration files: hue.ini, hue_safety_valve.ini, and hue_safety_valve_server.ini.

The hue.ini file is the first file that is auto-generated when you add the Hue service to your Cloudera cluster using Cloudera Manager. The hue_safety_valve.ini file is used to override bugs or configure properties that are missing from the Cloudera Manager UI. The hue_safety_valve_server.ini file is used to configure only the Hue role, and not the full Hue service (which includes the Hue Load Balancer). The hue_safety_valve_server.ini file is not used in practice.

Cloudera recommends that you do not edit the .ini files from the command line because they are stored in dynamic directories named by process ID and populated from the Cloudera Manager database. To add configurations that you cannot add directly from the Cloudera Manager UI, such as Authentication Backend for SAML, use the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field under the Hue service configurations in Cloudera Manager.

Run the following command to view the .ini files per process ID:

```
ls -ltr /var/run/cloudera-scm-agent/process/`ls -valrt /var/run/cloudera-scm
-agent/process | grep HUE_SERVER | tail -1 | awk '{print $9}'`
```

Figure 1: Terminal showing Hue configuration files

```
[root@hue4-cdh512-1 257-hue-HUE_SERVER]# cd /var/run/cloudera-scm-agent/process/`ls -valrt /var/run/cloudera-scm-agent/process
 | grep HUE_SERVER | tail -1 | awk '{print $9}'`
[root@hue4-cdh512-1 257-hue-HUE_SERVER]# pwd
/var/run/cloudera-scm-agent/process/257-hue-HUE_SERVER
[root@hue4-cdh512-1 257-hue-HUE_SERVER]# ll
total 76
-rwxr----- 1 hue  hue    393 Aug  4 09:17 altscript.sh
-rw-r----- 1 hue  hue    359 Aug  4 09:17 cloudera-monitor.properties
-rw------- 1 root root 21330 Aug  4 09:17 config.zip
-rw-r----- 1 hue  hue   1987 Aug  4 09:17 creds.localjceks
drwxr-xr-x 2 hue  hue    300 Aug  4 09:17 hive-conf
-rw-r----- 1 hue  hue   4267 Aug  4 09:17 hue.ini
-rw------- 1 hue  hue     99 Aug  4 09:17 hue.keytab
-rw-r----- 1 hue  hue      0 Aug  4 09:17 hue_safety_valve.ini
-rw-r----- 1 hue  hue      0 Aug  4 09:17 hue_safety_valve_server.ini
drwxr-x--x 2 hue  hue     60 Aug  4 09:17 impala-conf
drwxr-x--x 2 hue  hue     80 Aug  4 09:17 logs
-rw-r----- 1 hue  hue    541 Aug  4 09:17 navigator.client.properties
-rw-r----- 1 hue  hue    540 Aug  4 09:17 navigator.lineage.client.properties
-rw------- 1 root root  2415 Aug  4 09:17 proc.json
-rw-r----- 1 hue  hue      0 Aug  4 09:17 redaction-rules.json
drwxr-x--x 2 hue  hue     60 Aug  4 09:17 sentry-conf
-rw-r----- 1 hue  hue   8328 Aug  4 09:17 service-metrics.properties
drwxr-x--x 2 hue  hue     60 Aug  4 09:17 sqoop2-conf
-rw------- 1 root root  3000 Aug  4 09:17 supervisor.conf
drwxr-xr-x 2 hue  hue    220 Aug  4 09:17 yarn-conf
```

The process directory for any given role is mirrored in Cloudera Manager. Go to Hue Instances , select a role such as Hue Server, and then click the Processes tab.

Related Information

hue.ini

Hue configurations in Cloudera Runtime

Cloudera Manager exposes properties that allow you to insert custom configuration text into XML configuration, property, and text files, or into an environment. The configuration snippet mechanism is intended for use in cases where there is a configuration setting that is not exposed as a configuration property in the Cloudera Managerweb interface. Configuration snippets generally override normal configuration. You can configure custom properties for Hue by specifying them in the Advanced Configuration Snippet (Safety valve) field in Cloudera Manager.

The following table lists the safety valve parameters supported by Hue in Cloudera Runtime:

Parameter	Description
`[[desktop]]` `app_blacklist`	Used to add or remove applications, such as the File Browser, Impala, Hive, Oozie, and so on from the blocked list of applications.

`[desktop]`
`  [[session]]`
`  ttl=[***NUMBER-OF-SECONDS***]`
	Used to configure the duration of a user session. The ttl property determines the length of time that the cookie with the user's session ID lives before expiring. After the ttl setting is reached, the user's session expires whether it is active or not.
	The default setting for ttl is 1,209,600 seconds, which equals two weeks.

`[jobbrowser]`
`[[query_store]]`
`server_url=[***QUERY-PROCESSOR-URL***]`
	Used to display the **Queries** tab for Hive and Impala on the **Job Browser** page. This configuration is enabled by default and is not exposed in the Hue safety valve.
	However, to hide the **Queries** tab, you can override the configuration by adding the following lines in Cloudera Manager Clusters Hue Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini :
	`[jobbrowser]`
	`[[query_store]]`
	`is_enabled=false`
	To enable the **Queries** tab, set is_enabled to true.

`[aws]`
`[[aws_accounts]]`
`[[[default]]]`
`access_key_id=[***AWS-ACCESS-KEY***]`
`secret_access_key=[***SECRET-ACCESS-KEY***]`
`region=[***AWS-REGION***]`
	Used to enable the S3 File Browser for Hue without IDBroker.

`[aws]`
`has_iam_detection=true`
`[[aws_accounts]]`
`[[[default]]]`
`region=[***AWS-REGION***]`
	Used to enable the S3 File Browser for Hue with IDBroker.

`[desktop]`
`[azure]`
`  [[azure_accounts]]`
`  [[[default]]]`
`  client_id=[***AZURE-ACCOUNT-CLIENT-ID***]`
`  client_secret=[***AZURE-ACCOUNT-CLIENT-SECRET***]`
`  tenant_id=[***AZURE-ACCOUNT-TENANT-ID***]`
`  [[abfs_clusters]]`
`  [[[default]]]`
`  fs_defaultfs=abfs://[***CONTAINER-NAME***]@[***AZURE-STORAGE-ACCOUNT-NAME***]>.dfs.core.windows.net`
`  webhdfs_url=https://[***AZURE-STORAGE-ACCOUNT-NAME***].dfs.core.windows.net/`
	Used to enable the ABFS File Browser for Hue without IDBroker.

`[desktop]` `[azure]`	Used to enable the ABFS File Browser for Hue with IDBroker.

Contact Cloudera Support if you are required to use a configuration snippet that is not explicitly documented.

Introduction to Hue Advanced Configuration Snippet (Safety valves)

To customize and configure Hue properties, such as authentication or enabling S3 or ABFS browsers, and so on that you cannot directly configure from the Cloudera Manager UI, you can use the Advanced Configuration Snippet field under the Hue cluster configurations in Cloudera Manager.

Advanced safety valves

The field for hue_safety_valve.ini is service-wide and affects all Hue roles listed on the Instances tab. The field for hue_safety_valve_server.ini only affects Hue Server role instances.

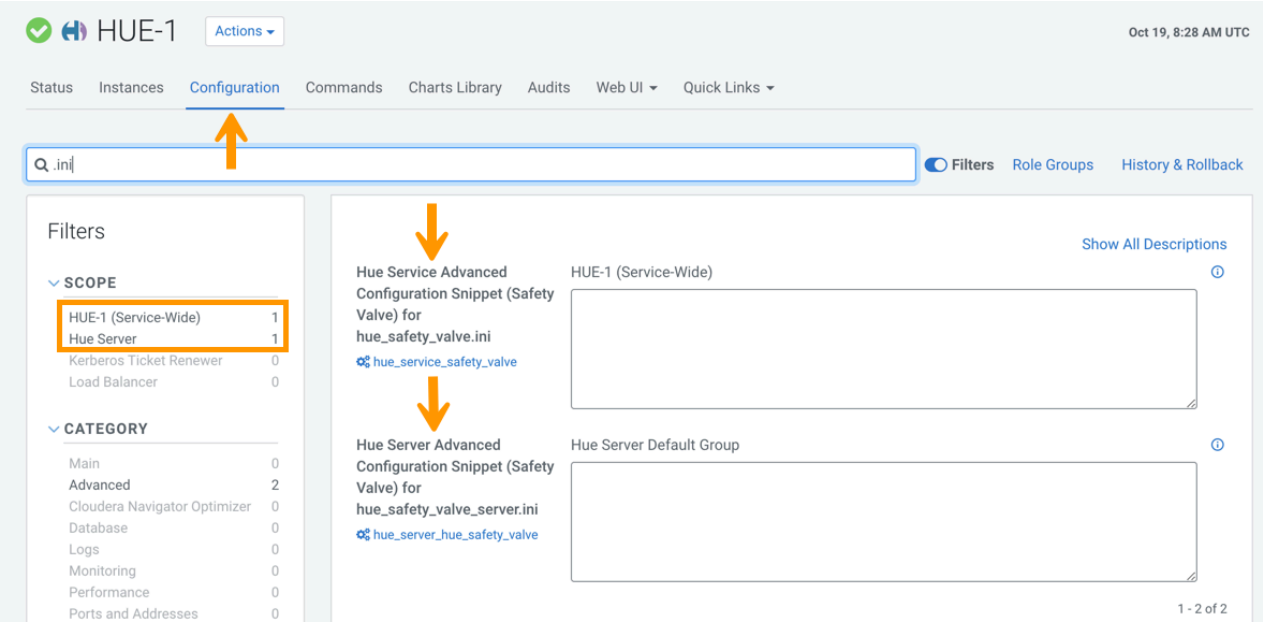

Environment safety valves

Environment safety-valves let you configure environment variables across the service or for specific role instances. These safety-valves lack headers.

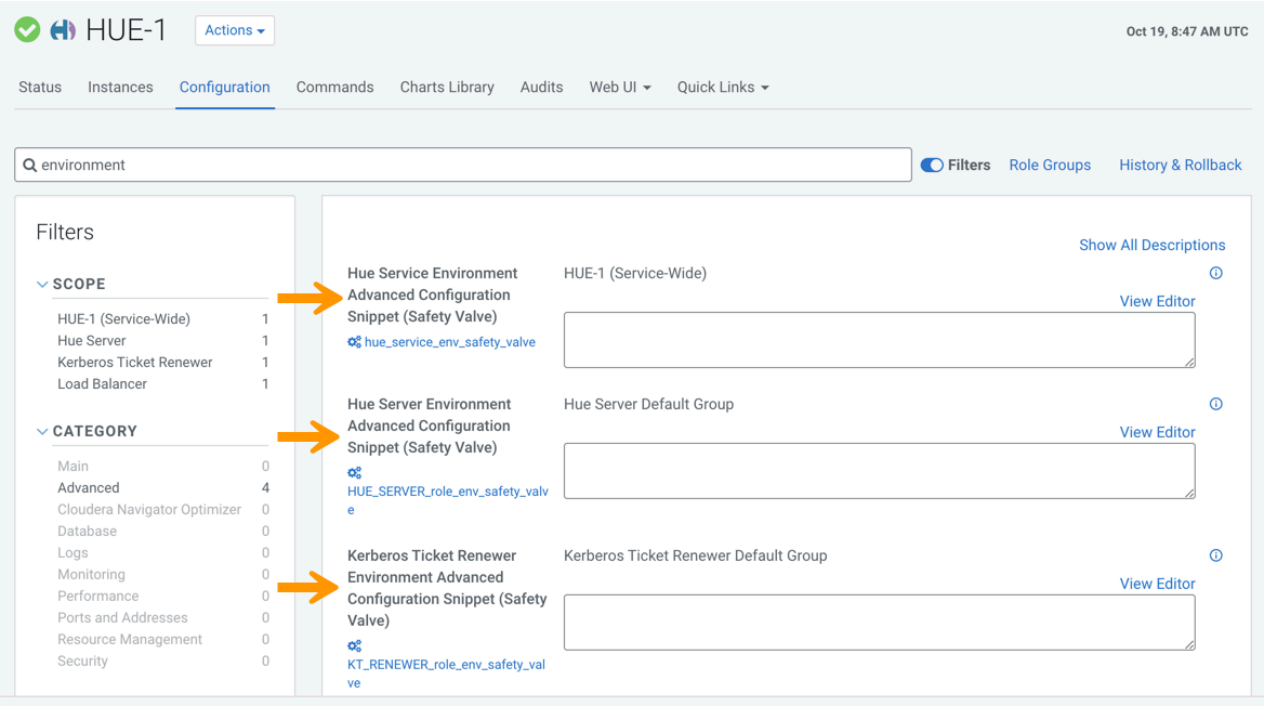

About setting up the Hue SQL AI Assistant

Administrators are required to set up and enable the SQL AI Assistant before analysts can use it to generate, edit, optimize, explain, and fix queries using natural language in Hue.

First, you must obtain clearance from your organization's infosec team to ensure it is safe to use the SQL AI Assistant because some of the table metadata and data, as mentioned in the previous section, is shared with the LLM.

Next, select and prepare one of the following AI services of your choice for hosting an LLM, and then configure the SQL AI Assistant in Hue:

- Microsoft Azure OpenAI service
- Amazon Bedrock service
- OpenAI platform

Prerequisites for configuring Hue SQL AI Assistant

To configure the SQL AI Assistant in Hue, you must pass the token required for connecting to the LLM service. Learn about the open and secure approaches to pass the tokens, and use the one that fits your organization policy.

Open approach for passing a token to configure Hue SQL AI Assistant

In this approach, you specify the token value in the Hue Advanced Configuration Snippet field in Cloudera Manager.The credentials are saved in a configuration file in the plain text format.

Note: Cloudera recommends that you use the open approach to pass tokens in test deployments, for proof of concept use cases.

Here's a list of the open token values in the hue-safety-valve field to configure the SQL AI Assistant:

For Open token

Microsoft Azure OpenAI

```
[desktop]
[[ai_interface]]
    service='azure'
    model_name='[***DEPLOYMENT-NAME***]'
    base_url="https://[***RESOURCE***].openai.azure.com/"
    token="[***RESOURCE-KEY***]"
```

AWS

```
[aws]
[[bedrock_account]]
 access_key_id='[***ACCESS-KEY***]'
 secret_access_key='[***SECRET-KEY***]'
 region='us-east-1'
[desktop]
[[ai_interface]]
 service='bedrock'
 model='claude'
```

OpenAI

```
[desktop]
[[ai_interface]]
 service='openai'
 token='[***API-KEY***]'
```

Configure SQL AI Assistant using the Microsoft Azure OpenAI service

Microsoft Azure allows for dedicated deployments of OpenAI GPT models. You can use Azure's OpenAI service instead of the publicly hosted OpenAI APIs, as it enables data processing within your Azure Virtual Network (VNet) network. GPT models can also be integrated with the Hue SQL AI Assistant using Azure's OpenAI service.

Before you begin

Obtain a Microsoft Azure subscription by working with your organization's IT team and registering for access to the Azure OpenAI service. For more information, see Create and deploy an Azure OpenAI Service resource.

Procedure

1. Log in to the Cloudera Management Console as an Administrator.
2. Go to Environments and select your environment.
3. Go to the Data Lake tab and click on the CM URL to open Cloudera Manager.
4. Go to Clusters Hue service Configuration and select add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[desktop]
[[ai_interface]]
    service='azure'
    model_name='[***DEPLOYMENT-NAME***]'
    base_url="https://[***RESOURCE***].openai.azure.com/"
```

5. Click Save Changes.

Results

You see ✨ Assistant on the Hue SQL editor, and the SQL AI Assistant will connect to the specified model on the Microsoft Azure OpenAI service.

Configure SQL AI Assistant using the Amazon Bedrock Service

This topic describes how to configure the SQL AI Assistant using the Amazon Bedrock Service.

Before you begin

You must have an AWS account with Bedrock access. For more information on accessing keys, see Amazon Bedrock.

Procedure

1. Log in to the Cloudera Management Console as an Administrator.
2. Go to Environments and select your environment.
3. Go to the Data Lake tab and click on the CM URL to open Cloudera Manager.
4. Go to Clusters Hue service Configuration and select add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

   ```
   [aws]
     [[bedrock_account]]
       access_key_id_script='echo $AWS_BEDROCK_ACCESS_KEY_ID'
       secret_access_key_script='echo $AWS_BEDROCK_SECRET_ACCESS_KEY'
       region='us-east-1'
   [desktop]
     [[ai_interface]]
       service='bedrock'
       model='claude'
   ```

   AWS_BEDROCK_ACCESS_KEY_ID and AWS_BEDROCK_SECRET_ACCESS_KEY must be added as encoded values under hue-secret.
5. Click Save Changes.

Results

You see ✨ Assistant on the Hue SQL editor, and the SQL AI Assistant will connect to the specified model in the Amazon Bedrock service.

Configure SQL AI Assistant using the OpenAI platform

This topic describes how to set up SQL AI Assistant and connect to a model on the OpenAI platform.

Before you begin

You must have created an account with the OpenAI platform.

Procedure

1. Log in to the Cloudera Management Console as an Administrator.
2. Go to Environments and select your environment.
3. Go to the Data Lake tab and click on the CM URL to open Cloudera Manager.

4. Go to Clusters Hue service Configuration and select add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[desktop]
  [[ai_interface]]
    service='openai'
```

   You can specify the model_name (optional) and define the model. If no model is defined, the default model (gpt-3.5-turbo-16k) will be used.
5. Click Save Changes.

Results

You see ✨ Assistant on the Hue SQL editor, and the SQL AI Assistant will connect to the specified model on the OpenAI platform.

Complete list of model-related configurations for setting up the Hue SQL AI Assistant

Review the list of service, model, and semantic search-related configurations used for custom configuring the AI services and models you want to use with the SQL AI Assistant and how to specify them in the Hue Advanced Configuration Snippet in the Cloudera Manager.

List of service and model-related configurations

You can configure the AI services and models you want to use by going to Cloudera Manager Clusters Hue service Configurations Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini and adding the following lines:

```
[desktop]
          [[ai_interface]]
          [***CONFIG-KEY1***]='[***VALUE***]'
          [***CONFIG-KEY2***]='[***VALUE***]'
          [[semantic_search]]
          [***CONFIG-KEY1***]='[***VALUE***]'
          [***CONFIG-KEY2***]='[***VALUE***]'
```

AI interface-related configurations

Here is the complete list of configurations under [[ai_interface]], which allows you to specify the service and model to be used:

AI interface config key	Description
service	API service to be used for AI tasks. AI is disabled when a service is not configured. For exam Workbench and Cloudera AI Inference service are API services.
service_version	API service version to be used for AI tasks
trusted_service	Indicates whether the LLM is trusted or not. Turn on to disable the warning. The default value
model	The AI model you want to use for AI tasks. For example, gptand llama.
model_name	The fully qualified name of the model to be used. For example, gpt-3.5-turbo-16k.
model_ref	The `model_ref` is a placeholder for adding the access key of the specific model you want to u
base_url	Service API base URL.
add_table_data	When enabled, sample rows from the table are added to the prompt. The default value is True.
table_data_cache_size	Size of the LRU cache used for storing table sample data.

AI interface config key	Description
auto_fetch_table_meta_limit	Number of tables to load from a database, initially.
token	Service API secret token.
token_script	Provides a secure way to get the service API secret token.
enabled_sql_tasks	A comma-separated list of SQL-related AI tasks available in the Editor.

Semantic search-related configurations

Specify the semantic search-related configurations used for RAG under the [[semantic_search]] section, as listed in the following table:

Semantic search config key	Description
relevancy	The technology you want to use for semantic search. Acceptable values are vector_search or v
embedding_model	The model you want to use for data-embedding. This must be compatible with SentenceTransf
top_k	Number of top-ranking items returned by semantic search.
cache_size	Size of the LRU cache used for storing embedding.

Hue logs

Cloudera Manager generates standard stream logs when each Hue role instance starts and stops. The Hue service, which is built on the Django framework, generates log4j logs when the service is running.

Standard stream logs

Cloudera Manager logs the start and stop of each supervised Hue process in standard stream logs (stdout.log, stderr.log).

When the Hue service restarts, Cloudera Manager generates a new directory for each supervised process of a Hue role instance:

```
ls -vrl /var/run/cloudera-scm-agent/process | grep HUE
```

```
[root@hue4-cdh512-1 ~]# ls -vrl /var/run/cloudera-scm-agent/process | grep HUE
drwxr-x--x 4 hue          hue          320 Aug  6 14:33 289-hue-HUE_LOAD_BALANCER
drwxr-x--x 8 hue          hue          440 Aug  6 14:33 288-hue-HUE_SERVER
drwxr-x--x 4 hue          hue          280 Aug  6 14:33 258-hue-HUE_LOAD_BALANCER
drwxr-x--x 8 hue          hue          460 Aug  6 14:33 257-hue-HUE_SERVER
drwxr-x--x 4 hue          hue          280 Aug  4 09:17 242-hue-HUE_LOAD_BALANCER
drwxr-x--x 8 hue          hue          460 Aug  4 09:17 241-hue-HUE_SERVER
drwxr-x--x 4 hue          hue          280 Aug  4 08:41 239-hue-HUE_LOAD_BALANCER
drwxr-x--x 8 hue          hue          460 Aug  4 08:41 238-hue-HUE_SERVER
drwxr-x--x 4 hue          hue          280 Aug  4 08:38 236-hue-HUE_LOAD_BALANCER
drwxr-x--x 8 hue          hue          460 Aug  4 08:38 235-hue-HUE_SERVER
```

It writes to a nested logs directory for each running instance:

```
[root@hue4-cdh512-1 logs]# pwd
/var/run/cloudera-scm-agent/process/289-hue-HUE_LOAD_BALANCER/logs
[root@hue4-cdh512-1 logs]# ll
total 16
-rw-r--r-- 1 root root 11148 Aug  6 14:33 stderr.log
-rw-r--r-- 1 root root   447 Aug  6 14:33 stdout.log
```

Configuration errors are written here because they prevent Hue servers and load balancers from starting properly.

Tip: Testing the LDAP configuration from Cloudera Manager (Clusters Hue service Test LDAP Configuration) also writes to standard stream logs which you can search using the following command: ls -vrl /var/run/cloudera-scm-agent/process | grep ldaptest

Hue service Django logs

When the Hue service is running, Hue generates logs in /var/log/hue using log4j. Load balancer logs are in /var/run/httpd. You can view these logs in Hue at http://hueserver:port/logs.

Table 1: Hue service logs

Log Name	Description
access.log	Filtered list of successful attempts to access Hue Web UI.
audit.log	Captures user operations, such as: • USER_LOGIN • USER_LOGOUT • EXPORT • DOWNLOAD • DELETE_USER • DELETE_GROUP • EDIT_USER • CREATE_USER • EDIT_GROUP • CREATE_GROUP • EDIT_PERMISSION • ADD_LDAP_USERS • ADD_LDAP_GROUPS • SYNC_LDAP_USERS_GROUPS
audit/hue_server_audit_wal.log	Audit log visible in Apache Atlas.
error.log	Filtered list of all nontrivial errors
kt_renewer.log	Kerberos ticket renews
metrics-hue_server/metrics.log	Populates charts in Cloudera Manager
migrate.log	Database and table migrations + First Run of Hue server
rungunicornserver.log	Hue (Gunicorn) web server information.
hue_install.log	Contains the log produced during installation

Enabling DEBUG logging for Hue logs

DEBUG is available for the Hue server logs in /var/log/hue. By default, the Hue service writes INFO level messages and keeps a small buffer of log messages at all levels in memory. If you want detailed, debug-level information, then you can enable this feature from Hue configuration in Cloudera Manager.

About this task

In the debug mode, Hue displays a detailed traceback when an exception occurs. Debugging information may contain sensitive data.

Procedure

1. Log in to the Cloudera web interface as an Administrator.
2. Go to Cloudera Management Console Environments , select your environment and go to the Data Lake tab.
3. Click on the CM URL to open Cloudera Manager.
4. Go to Clusters Hue Configuration and select the Enable Django Debug Mode option.

   The Enable Django Debug Mode option is used to toggle between INFO and DEBUG-level logging.
5. Restart the Hue service.

Enabling httpd log rotation for Hue

Several log files, such as access logs and error logs, are produced by the Apache HTTP server. Over time, these files can grow in size and can cause disk issues. You can prevent this problem by using Apache's rotatelogs utility and defining when to rotate logs – either based on a time interval or maximum size of the log.

About this task

You can configure the Apache server to rotate the logs when a certain filesize is reached (for example, 5 Bytes or 10 KBytes) or at a set time interval (for example, daily or weekly), or both. If you want to specify both the time and size, the size must be specified after the time. Log rotation occurs whenever time or size limits are reached.

The log files are generated as per the configurations defined in the httpd.conf file.

Before you begin

The rotatelogs utility must be present on your Operating System, and you must know its location so that you can use it. Run the following command from the Hue load balancer host system to get the path of the rotatelogs utility:

```
[root@example-domain-hue-HUE_LOAD_BALANCER]# which rotatelogs
/usr/sbin/rotatelogs
```

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration .
3. Configure the log rotation policy based on your criteria in the Load Balancer Advanced Configuration Snippet (Safety Valve) for httpd.conf field.

   To rotate logs whenever the specific file size is reached, add the following lines:

   ```
   CustomLog "|/usr/sbin/rotatelogs /var/log/hue-httpd/access_log.%Y-%m-%d-
   %H_%M_%S [***FILE-SIZE***](B|K|M|G)" common
   ErrorLog "|/usr/sbin/rotatelogs /var/log/hue-httpd/error_log.%Y-%m-%d-%H_
   %M_%S [***FILE-SIZE***](B|K|M|G)"
   ```

   For example:

   ```
   CustomLog "|/usr/sbin/rotatelogs /var/log/hue-httpd/access_log.%Y-%m-%d-
   %H_%M_%S 5B" common
   ```

```
ErrorLog "|/usr/sbin/rotatelogs /var/log/hue-httpd/error_log.%Y-%m-%d-%H_
%M_%S 5B"
```

This configuration rotates the log file whenever it reaches a size of 5 bytes, and the suffix to the log file name will be created with the format access_log. YYYY-mm-dd-HH_MM_SS and error_log.YYYY-mm-dd-HH_MM_SS.

To rotate logs daily, add the following lines:

```
CustomLog "|/usr/sbin/rotatelogs -l /var/log/hue-httpd/access_log.%Y-%m-
%d-%H_%M_%S 86400" common
ErrorLog "|/usr/sbin/rotatelogs -l /var/log/hue-httpd/error_log.%Y-%m-%d-
%H_%M_%S  86400"
```

This creates the files /var/log/access_log.yyyy.mm.dd and /var/log/error_log.yyyy.mm.dd where yyyy is the year, mm is the month, and dd is the day of the month. The log operation switches to a new file every day at midnight, local time.

For more information, see *rotatelogs - Piped logging program to rotate Apache logs*.

4. Click Save Changes.
5. Restart the affected services.

Related Information

rotatelogs - Piped logging program to rotate Apache logs

Hue supported browsers

Hue works with the two most recent LTS (long term support) or ESR (extended support release) browsers. Cookies and JavaScript must be enabled.

The lists the minimum tested versions of the most common browsers:

- Chrome: (Version history)
- Firefox: (Version history)
- Safari (Mac only): Version history
- Microsoft Edge: (Version history)

Hue can display in other browsers and in older versions of the common browsers, but you might not have access to all features.

Enabling cache-control HTTP headers when using Hue

You can enable Hue to use HTTP headers such as Cache-Control, Pragma, and Expires to ensure that your broswer always uses the fresh version of the resource. You must set the value of the custom_cache_control property to true in Hue's Advanced Configuration Snippet.

About this task

Note: Enabling the use of these cache-control HTTP headers can affect performance because your broswer tries to fetch the latest resources instead of using cached data. To disable cache-control HTTP headers, set the value of the custom_cache_control property to false.

Procedure

1. Log in to the Cloudera web interface as an Administrator.
2. Go to Cloudera Management Console Environments , select your environment and go to the Data Lake tab.

3. Click on the CM URL to open Cloudera Manager.
4. Go to Clusters Hue Configuration and add the following line in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[desktop]
custom_cache_control=true
```

5. Click Save Changes.
6. Restart the Hue service.

Setting up a Hue service account with a custom name

In Cloudera, “hue” username is the default machine user and group name for Hue. You can create a custom machine user and group name on the Hue cluster and then specify them in the Hue configurations inCloudera Manager.

Procedure

1. SSH into the Hue server host as a user with sudo privileges
2. Create a new group for your custom Hue user account by running the following command:

```
sudo groupadd[***NEW-HUE-GROUP-NAME***]
```

   Replace *[***NEW-HUE-GROUP-NAME***]* with the custom group name you want to use.
3. Create a new user and assign it to the newly created group by running the following command:

```
sudo useradd -G [***NEW-HUE-GROUP-NAME***] -m [***NEW-HUE-USER-NAME***] -
s /usr/sbin/nologin
```

   The new user is stored in the /etc/passwd file, and group is stored in the /etc/group file.
4. Verify that the new user exists within the newly created group by running the following command:

```
grep [***NEW-HUE-USER-NAME***] /etc/passwd
```

5. Change the ownership of the Hue directories as follows:

```
sudo chown -R [***NEW-HUE-USER-NAME***]:[***NEW-HUE-GROUP-NAME***] /var/lo
g/hue
sudo chown -R [***NEW-HUE-USER-NAME***]:[***NEW-HUE-GROUP-NAME***] /var/lo
g/hue-httpd
sudo chown -R [***NEW-HUE-USER-NAME***]:[***NEW-HUE-GROUP-NAME***] /tmp/hu
e_*
```

6. Modify the Hue configuration in Cloudera Manager as follows:
   a) Log in to Cloudera Manager as an Administrator.
   b) Go to Clusters Hue Configuration .
   c) Specify the custom group name you created earlier*[***NEW-HUE-GROUP-NAME***]* in the Default User Group field.
   d) Specify the custom user name you created earlier*[***NEW-HUE-USER-NAME***]* in the System User field.
   e) Specify the custom group name you created earlier*[***NEW-HUE-GROUP-NAME***]* in the System Group field.
   f) Add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field.

```
[desktop]
server_user=[***NEW-HUE-USER-NAME***]
```

```
server_group=[***NEW-HUE-GROUP-NAME***]
default_user=[***NEW-HUE-USER-NAME***]
```

g) Click Save Changes.

h) Restart the Hue service.

7. Tail the /var/log/hue/rungunicornserver.log log file for any issues.
8. Change the permissions of the Hue home directory as follows:

```
sudo cd /opt/cloudera/parcels/CDH/lib/
sudo chown -R [***NEW-HUE-USER-NAME***]:[***NEW-HUE-GROUP-NAME***]hue
```

9. Change the permissions of the process directory as follows:

```
sudo cd /var/run/cloudera-scm-agent/
sudo setfacl -Rm user:[***NEW-HUE-USER-NAME***]:rwx process
```

10. Verify Access Control List (ACL) changes by running the following command:

```
getfacl process
```

To validate newly configured custom user and group

Run the following Python script on your Hue hosts to validate that the new user and group names are valid and configured correctly.

This script requires you to input the user (uid) and group (gid) IDs corresponding to the custom user and group names you created and displays the effective user and group IDs that are in use. You must ensure that the user belongs to the same group while specifying the uid and gid. The script displays the “Permission denied: Operation not permitted” error if there is a mismatch between the user ID and the group ID :

```
import os
def change_user_and_group(euid, egid):
    try:
        # Change the effective group ID
        os.setegid(egid)
        print(f"Effective Group ID changed to: {egid}")
        # Change the effective user ID
        os.seteuid(euid)
        print(f"Effective User ID changed to: {euid}")
        # Verify changes
        print(f"Current Effective User ID: {os.geteuid()}")
        print(f"Current Effective Group ID: {os.getegid()}")
    except PermissionError as e:
        print(f"Permission denied: {e}")
    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    # Replace with actual user and group IDs that you want to switch to
    # Change `new_euid` to the user ID (UID) of the custom user you created.
    # Change `new_egid` to the group ID (GID) of the custom group you create
d.
    # Example: new_euid = 1001  # Non-root user ID
    #          new_egid = 1001  # Non-root group ID
    new_euid = 1001  # Replace with the new user ID (UID)
    new_egid = 1001  # Replace with the new group ID (GID)
    change_user_and_group(new_euid, new_egid)
```

Options to restart the Hue service

Restarting the Hue service is necessary after upgrading Cloudera or making configuration changes to the Hue service. You can restart Hue in regular or rolling restart mode. Rolling restart reduces the service downtime to approximately 80 to 90 seconds.

You can restart the Hue service independently or as a part of the cluster restart. When you restart the Hue service normally, Hue can become unavailable for an average of 30 minutes from the restart time (this could vary depending on the number of services in your cluster). When you restart the cluster containg the Hue service, Hue is stopped first among other services in the cluster and restarted at the end. To reduce the service downtime, Cloudera recommends that you use the “Rolling Restart” option.

Note: Cluster-level rolling restart is supported only in High Availability (HA) clusters. However, you can restart the Hue service and its roles independently in the rolling restart mode in both HA and non-HA clusters.

Restarting the Hue service independently in rolling restart mode

To restart Hue service in the rolling restart mode, use one of the following steps:

- Go to Cloudera Manager Clusters and click ⋮ corresponding to the Hue service and then click Rolling Restart, or
- Go to Cloudera Manager Clusters Hue and click Actions Rolling Restart .

Cloudera Manager shows the following options to restart the Hue service in the rolling restart mode:

Category	Restart options	Description
Restart filters	Restart roles with stale configuration only	Select this option to restart Hue if you have updated the Hue configuration.
	Restart roles with the old software version only	Select this option to restart Hue if you have upgraded the cluster.
Role types to restart	Kerberos Ticket Renewer	Selected by default. Restarts the Kerberos Ticket Renewer role.
	Load Balancer	Selected by default. Restarts the Load Balancer role after the Kerberos Ticket Renewer.
	Hue Server	Selected by default. Restarts the Hue Server role after the load balancer.

When you restart Hue, Hue’s non-worker roles, such as the load balancer, Kerberos ticket renewer, and Hue server restart one after the other.

Restarting a CDP cluster containing the Hue service in the rolling restart mode

To restart an HA cluster containing the Hue service in the rolling restart mode, go to Cloudera Manager Clusters and click ⋮ corresponding to your cluster name and select Rolling Restart. You see the **Rolling Restart** modal with the list of services that support rolling restart.

Note: All Hue roles (namely the load balancer, Kerberos ticket renewer, and Hue server) are of the non-worker type. You must select one of the following options under the Roles to include section for ensuring that the Hue service is restarted in the rolling restart mode, depending on your requierments:

- Non-Workers Only, or
- All Roles

Related Information

Rolling Restart

Customizing the Hue web interface

To customize the Hue web interface, add configuration properties in Cloudera Manager. You can customize the banner, the page logo, the cache timeout setting, and you can enable or disable anonymous usage data collection.

Adding a custom banner in Hue

You can add a custom banner to the Hue web interface by adding your custom HTML to the Top Banner Custom HTML property in Cloudera Manager.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration and add the banner test in the Top Banner Custom HTML field as follows:
   <H1>Your company's custom Hue Web UI banner</H1>
3. Click Save Changes.
4. Restart the Hue service.

Changing the page logo in Hue

You can replace the Hue web interface logo with a custom log that is created with an SVG code. Add any type of logo you want, but your custom logo should be designed to fit into a 160 x 40 pixel space.

About this task

For example, here is the Hue logo shown in the following image:

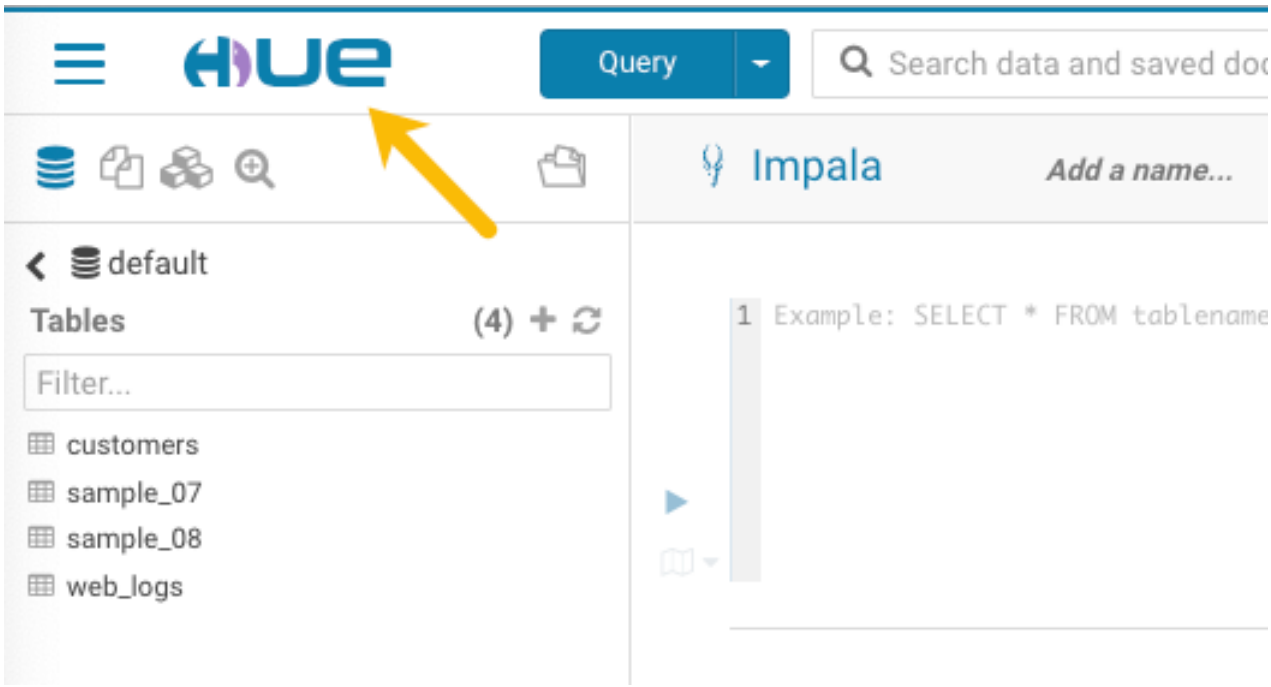

You can change this Hue logo by adding the appropriate SVG code to the logo_svg property under the [desktop] [[custom]] section in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini configuration parameter in Cloudera Manager.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue Configuration
3. In the Search text box, type hue_safety_valve.ini to locate the configuration parameter:

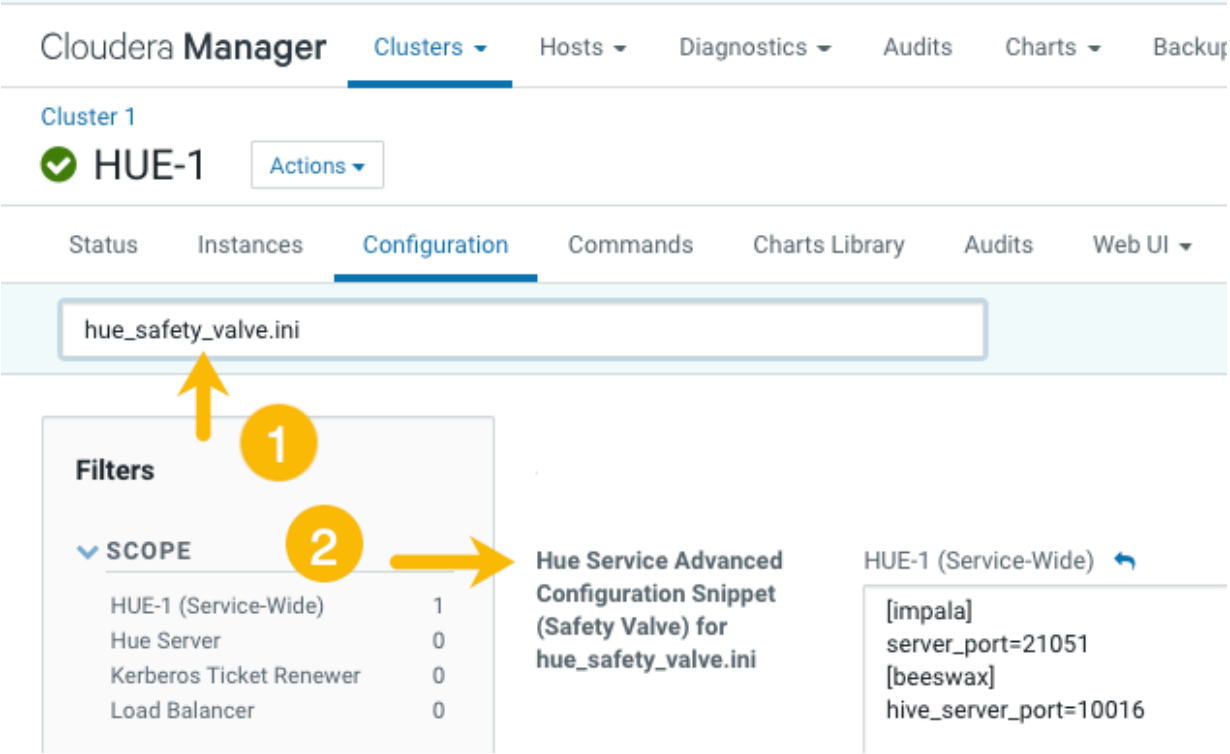

4. Add the following parameters with your custom logo SVG code to the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini configuration parameter:

```
[desktop]
[[custom]]
logo_svg='[***SVG-CODE-FOR-CUSTOM-LOGO***]'
```

For example, the following SVG code replaces the Hue logo with a red heart:

```
[desktop]
[[custom]]
logo_svg='<g><path stroke="null" id="svg_1" d="m44.41215,11.43463c-4.05
017,-10.71473
-17.19753,-5.90773 -18.41353,-0.5567c-1.672,-5.70253 -14.497,-9.95663
-18.411,0.5643c-4.35797,11.71793 16.891,22.23443 18.41163,23.95773c1.518
1,-1.36927 22.7696,-12.43803
18.4129,-23.96533z" fill="#ffffff"/> <path stroke="null" id="svg_2"
d="m98.41246,10.43463c-4.05016,-10.71473 -17.19753,-5.90773 -18.41353,-
0.5567c-1.672,-5.70253
-14.497,-9.95663 -18.411,0.5643c-4.35796,11.71793 16.891,22.23443 18.4116
4,23.95773c1.5181,-1.36927
22.76959,-12.43803 18.41289,-23.96533z" fill="#FF5A79"/> <path stroke="nu
ll" id="svg_3"
d="m154.41215,11.43463c-4.05016,-10.71473 -17.19753,-5.90773 -18.41353,-0
.5567c-1.672,-5.70253
```

```
-14.497,-9.95663 -18.411,0.5643c-4.35796,11.71793 16.891,22.23443 18.41164
,23.95773c1.5181,-1.36927 22.76959,-12.43803 18.41289,-23.96533z" fill="
#ffffff"/> </g>'
```

5. Click Save Changes.
6. Restart the Hue service.
7. In the Hue configuration page of Cloudera Manager, select Web UI Hue Load Balanced to load Hue and view your custom logo.

   If you added the sample SVG code that defines a red heart as the logo, your Hue web UI looks like this:

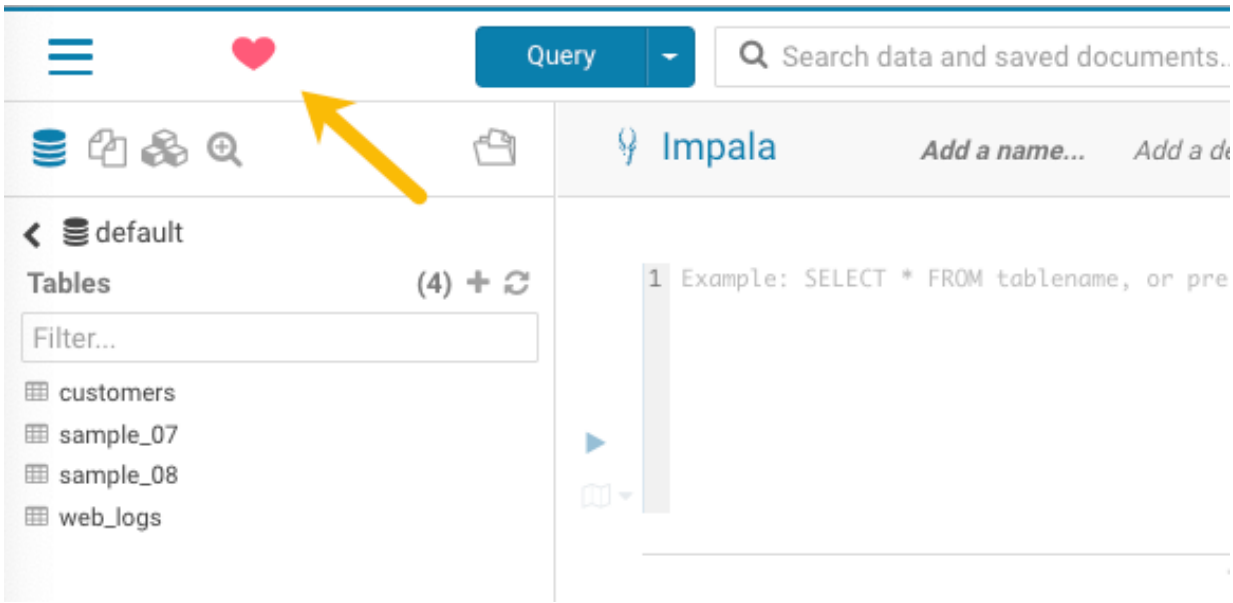

8.
9.
10.

Related Information

Scalable Vector Graphics

Adding a splash screen in Hue

You can add a custom splash screen to the Hue web interface by adding your custom HTML to the Advanced Configuration Snippet in Cloudera Manager.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration and add the banner test in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field as follows:

```
[desktop]
[[custom]]
login_splash_html=[***CUSTOM-HTML***]
```

```
[desktop]
[[custom]]
login_splash_html=<h1>Hue, the next-gen SQL Assistant</h1>
```

3. Click Save Changes.
4. Restart the Hue service.

Setting the cache timeout

Enable Hue UI caching by setting a timeout value in milliseconds. The default is 10 days or 864000000 milliseconds. Set the timeout to 0 to disable caching.

About this task

When you browse tables using the left assist panel or run queries, Hue caches this information for fetching information faster and query autocompletion. You can configure the time for which you want to Hue to cache this information by setting the value of the cacheable_ttl property under the [desktop][[custom]] section in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini configuration property in Cloudera Manager.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue Configuration .
3. In the Search text box, type hue_safety_valve.ini to locate the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini configuration parameter.
4. Add the following parameters with the cache timeout value to the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini configuration parameter:

   ```
   [desktop]
   [[custom]]
   cacheable_ttl=[***VALUE-IN-MILLISECONDS***]
   ```

   For example, the following configuration sets the cache timeout to the default value of 86400000 milliseconds:

   ```
   [desktop]
   [[custom]]
   cacheable_ttl=86400000
   ```

5. Click Save Changes.
6. Restart the Hue service.

Enabling or disabling anonymous usage date collection

Hue tracks anonymized pages and application versions to gather information about application usage levels. The data collected does not include hostnames or IDs. For example, the data collected has the format /2.3.0/pig or /2.5.0/beeswax/execute.

About this task

To enable or disable anonymous usage data collection:

Procedure

1. In the Cloudera Manager Admin Console, select Clusters Hue Configuration to navigate to the configuration page for Hue.
2. In the Search text box, type usage to locate the Enable Usage Data Collection check box:
   - To enable anonymous data collection, check the box, which is the default setting.
   - To disable anonymous data collection, clear the check box.

3. Enter a Reason for change..., and then click Save Changes at the bottom of the page to save the configuration change.
4. Refresh the browser page and click the restart icon at the top of the page so the new configuration changes can be read by the server and the new data collection setting takes effect.

Disabling the share option in Hue

Hue allows you to share documents, queries, and workflows with other users, either for viewing only or viewing and modifying. Sharing is enabled by default in the Hue UI. For added privacy and control, you can disable sharing by setting the enable_sharing property to false in the Advanced Configuration Snippet.

About this task

The sharing option is always available to the admin users. To disable the share option:

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini and set the value of the enable_sharing property to false in the [desktop] section:

```
[desktop]
enable_sharing=false
```

3. Click Save Changes.
4. Restart the Hue service.

Enabling Hue applications with Cloudera Manager

Most Hue applications are configured by default, based on the services you have installed.Cloudera Manager selects the service instance that Hue depends on. If you have more than one service, you may want to verify or change the service dependency for Hue. If you add a service such as Oozie after you have set up Hue, you must set the dependency because it is not done automatically.

About this task

To add a dependency in Hue:

Procedure

1. In the Cloudera Manager Admin Console, select Clusters Hue Configuration to navigate to the configuration page for Hue.
2. Filter by ScopeHue (Service-Wide) and CategoryMain .
3. Select the <*SERVICE_NAME*> Service property that you want to set a dependency for. Select none to remove a dependency.
4. Enter a Reason for change..., and then click Save Changes at the bottom of the page to save the configuration change.
5. Refresh the browser page and click the restart icon at the top of the page so the new configuration changes can be read by the server and the new setting takes effect.

Running shell commands

You can run shell commands to administer Hue programmatically. For example, to reset the superuser password or to assign an LDAP user superuser permissions.

About this task

To run Hue shell commands:

Procedure

1. Set HUE_CONF_DIR to the latest Hue process directory:

```
export HUE_CONF_DIR="/var/run/cloudera-scm-agent/process/`ls -alrt /var/
run/cloudera-scm-agent/process | grep HUE_SERVER | tail -1 | awk '{print
 $9}'`"
echo $HUE_CONF_DIR
```

2. Set environment variables used to run the Hue webserver:
   - CentOS/RHEL:

```
for line in `strings /proc/$(lsof -i :8888|grep -m1 python|awk '{ print
 $2 }')/environ|egrep -v "^HOME=|^TERM=|^PWD=|^CLDR_JAVA_OPTS="`;do expo
rt $line;done
```

   - Ubuntu:

```
for line in `strings /proc/$(lsof -i :8888|grep -m1 hue|awk '{ print $2
 }')/environ|egrep -v "^HOME=|^TERM=|^PWD=|^CLDR_JAVA_OPTS="`;do export
$line;done
```

Note: Environment variables separated by spaces or underscores (_) are not parsed correctely, and must be excluded by using a caret symbol (^) to prevent errors.

3. Run shell subcommands

   When true, HUE_IGNORE_PASSWORD_SCRIPT_ERRORS runs the Hue shell even if hue.ini contains passwords generated by Cloudera Manager (such as bind_password and ssl_password).

   Note: Do not export HUE_IGNORE_PASSWORD_SCRIPT_ERRORS or HUE_DATABASE_PASSWORD to ensure that they are not stored and only apply to this command.

   For CDH parcel deployments:

   - Run the interactive Hue Python shell (Ctrl+D to quit)

     ```
     HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 /opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue shell
     ```

     Or with the database password:

     ```
     HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 HUE_DATABASE_PASSWORD=<your db password> /opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue shell
     ```

   - Change a user password

     ```
     HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 /opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue changepassword admin
     ```

   - Promote Hue user to superuser

     ```
     HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 /opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue shell
     ```

     ```
     from django.contrib.auth.models import User
     a = User.objects.get(username='gwen')
     a.is_superuser = True
     a.save()
     ```

   - Count all of the documents of a certain user:

     ```
     from django.contrib.auth.models import User
     from desktop.models import Document2

     user=User.objects.get(username='demo')
     Document2.objects.documents(user=user).count()

     Out[X]: 1167
     ```

   - List available subcommands

     ```
     HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 /opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue
     ```

   For CDH package deployments:

   - ```
     HUE_IGNORE_PASSWORD_SCRIPT_ERRORS=1 /usr/lib/hue/build/env/bin/hue shell
     ```
     ```

Downloading and exporting data from Hue

Hue enables you to download or export data from Hue to HDFS or to an external storage location from Hue Editor, Hue Dashboard, and the Hue File browser. You can limit the number of rows or bytes that are downloaded or disable the export feature altogether so that you do not run out of storage space.

For a service-wide change, go to Cloudera Manager Clusters Hue service Configuration and specify the configurations in the Hue Service Advanced Configuration Snippet (Safety valve) for hue_safety_valve.ini field.

By default, Hue users can download the query results from the Hue Editor, the Hue Dashboard, and the File browser.

Limiting the number of rows to download

Specify the following in the Hue Service Advanced Configuration Snippet (Safety valve) for hue_safety_valve.ini to limit the number of rows that can be downloaded from a query before it is truncated:

```
[beeswax]
download_row_limit=X
```

X represents the number of rows that you can download.

By default, there is no download limit, and you can configure this by setting the value to “-1”:

```
[beeswax]
download_row_limit=-1
```

Limiting the number of bytes to download

Specify the following in the Hue Service Advanced Configuration Snippet (Safety valve) for hue_safety_valve.ini to limit the number of bytes that can be downloaded from a query before it is truncated:

```
[beeswax]
download_bytes_limit=X
```

X represents the number of bytes that you can download.

By default, there is no download limit, and you can configure this by setting the value to “-1”:

```
[beeswax]
download_bytes_limit=-1
```

Disabling the data download feature

Specify the following in the Hue Service Advanced Configuration Snippet (Safety valve) for hue_safety_valve.ini field to disable your users from downloading query results:

```
[desktop]
enable_download=false
```

Specify the following in the Hue Service Advanced Configuration Snippet (Safety valve) for hue_safety_valve.ini field to hide the Download button from the Hue File browser:

```
[filebrowser]
show_download_button=false
```

Enabling a multi-threaded environment for Hue

A multi-threaded environment can help reduce the time it takes to migrate data from one database to other. By default, operations such as migrating data run on a single thread. For example, if you are switching from MySQL as the backend database for Hue to Oracle, then enabling a multi-threaded environment significantly reduces the data transfer time.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini .
3. Locate the [[database]] section under [desktop] and set threaded to true:

```
[desktop]
[[database]]
options={"threaded":true}
```

4. Click Save Changes.
5. Restart the Hue service.

Adding Query Processor service to a cluster

The Query Processor service indexes Hive and Tez events and provides APIs to access them. It is required if you want to view the Queries tab (query history and query details) on the Hue Job Browser.

About this task

Attention: This task is applicable only if you are upgrading to Cloudera 7.1.8 and higher. If you are upgrading to Cloudera 7.1.7 SP2, 7.1.7 SP1, 7.1.7, or lower, then you can skip this task.

Note: The Query Processor service does not support MariaDB.

You can either install the Query Processor service as a dependency while adding the Hue service or after adding the Hue service.

Before you begin

Note: You must be assigned the DataHubCreator role to perform this task.

This task assumes that you already have a database installed on a host in your cluster. On Cloudera, Hue Query Processor supports PostgreSQL, MySQL, and Oracle databases. The supported PostgreSQL database version for Hue Query Processor is 9.6 and higher.

Next, you need to create a database for the Query Processor service with the required roles. To create the Query Processor database:

1. SSH into your database host as a root user.

2. Start the database terminal by running the following commands:

   (PostgreSQL)

   ```
   sudo -u postgres psql
   ```

   (MySQL)

   ```
   systemctl start mysqld
   ```

   (Oracle)

   ```
   sqlplus / as sysdba
   ```

3. Run the following statement while specifying the username, password, and a database name for the Query Processor:

   (PostgreSQL)

   ```
   CREATE ROLE [***QP-USER***] WITH LOGIN PASSWORD '[***QP-PASSWORD***]';
   ALTER ROLE [***QP-USER***] CREATEDB;
   CREATE DATABASE [***QP-DATABASE***];
   GRANT ALL PRIVILEGES ON DATABASE [***QP-DATABASE***] TO [***QP-USER***];
   ```

   (MySQL)

   ```
   CREATE USER [***QP-USER***] IDENTIFIED BY '[***QP-PASSWORD***]';
   CREATE DATABASE [***QP-DATABASE***];
   USE [***QP-DATABASE***];
   GRANT ALL PRIVILEGES ON [***QP-DATABASE***].* TO [***QP-USER***];
   ```

   (Oracle)

   ```
   CREATE USER [***QP-USER***] IDENTIFIED BY [***QP-PASSWORD***];
   GRANT ALL PRIVILEGES TO [***QP-USER***];
   ```

   Note: When you create a user in the Oracle database, the associated schema is created automatically. This schema becomes the default schema for that user.

Procedure

1. Log in to the Cloudera Management Console as an Administrator.
2. Go to your environment and click on CM UI to open Cloudera Manager.
3. Go to Clusters ⋮ Add Service .
4. Select Query Processor on the **Add Service to Cluster** page and click Continue.

   The required dependencies are automatically selected.
5. Select the host on which you want to install the Query Processor by clicking View By Host. Then click Continue.
6. Select the database type, and specify the database hostname, database name, and username and password to access the database on the **Setup Database** page and click Test Connection.

   Attention:

   - (For MySQL and Oracle databases) Ensure that you specify the same name in the Database Name and Database Schemas fields.
   - (For Oracle database) You must specify the value of the query_processor_database_schema property (Database Schema) in upper case.

   After the connection is verified, click Continue.

7. On the **Review Changes** page, accept the default settings and click Continue.

   If Kerberos or Auto-TLS are set up on your Cloudera Data Hub cluster, then the corresponding settings are automatically configured.

   The **Command Details** page is displayed, which shows the status of the installation and configuration.
8. Click Continue if the operation is successful.
9. Click Finish on the **Summary** page.

Results

You are redirected to the Cloudera Manager home page. You can now see the Query Processor service listed within your cluster.

What to do next

(Optional) To view the **Queries** tab on the **Job Browser** page in Hue, you must enable the Query Processor Service option in Hue configurations using Cloudera Manager.

Removing Query Processor service from cluster

You can remove the Query Processor service using Cloudera Manager when you no longer need it.

Before you begin

Because Hue has a dependency on the Query Processor service, deselect the Query Processor Service option in the Hue configurations before proceeding to remove service dependencies.

Procedure

1. Log in to the Cloudera Management Console as an Administrator.
2. Go to your environment and click on CM UI to open Cloudera Manager.
3. Go to Clusters Query Processor service ⋮ Delete .

   If you have not removed the service dependencies before deleting the Query Processor, then a **Remove Service Dependencies** dialog box is displayed.

   a) Click Configure Service Dependency.
   b) Deselect the Query Processor Service option on the **Configuration** page and click Save Changes.
   c) Go to Clusters Query Processor service ⋮ Delete .

Enabling the Query Processor service in Hue

The Queries tab on Hue's Job Browser page allows you to view only Hive query history and details. You need to enable the Query Processor service from the Hue configurations to enable the Queries tab. Hue does not display Impala query history and query details. Currently, they are not supported and will result in an error if accessed.

About this task

The Query Processor service is automatically enabled in Hue if you select it as a dependency on the **Select Dependencies** page while adding the Hue service to your cluster. But if you did not select it as a dependency while adding the Hue service, you can add the Query Processor service to your cluster (if not installed already) and then enable the Query Processor Service option in Hue configurations.

Procedure

1. Log in to the Cloudera Management Console as an Administrator.
2. Go to your environment and click on CM UI to open Cloudera Manager.
3. Go to Clusters Hue service Configuration and select the Query Processor Service option.
4. Click Save Changes.
5. Restart the Hue service.

Results

You can now view Hive query history and query details under the **Queries** tab on Hue's **Job Browser** page.

Adding Query Processor Administrator users and groups on Cloudera Data Hub clusters

Query Processor Administrators have special privileges that enable them to view and monitor queries from all users, including the ones that are submitted from query interfaces, such as Beeline, Hive Warehouse Connector (HWC), Tableau, and so on.

About this task

This task is applicable only if you are running Hive queries.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Query Processor Configuration and enter the list of users in the Extra Auth Options field as follows:

   "adminUsers": "hive, [***USER-1***], [***USER-2***]"

   You can also add a list of admin groups as follows:

   "adminGroups": "admin-group, [***GROUP-1***], [***GROUP-2***]"
3. Click Save Changes.
4. Restart the Query Processor service.

Ways to clean up old queries from the Query Processor tables

Learn how to schedule a query clean-up and how to use the API to manually clean up queries from the following Query Processor tables: vertex_info, dag_details, dag_info, query_details, hive_query, tez_app_info.

Scheduling query clean-up

By default, the Hue Query Processor automatically clears queries from the Query Processor tables that are 30 days old at 2 AM every day as per the timezone set on your server. You can configure the following parameters in Cloudera Manager Query Processor service Configurations Query Processor Extra Configurations to set up a schedule based on your requirements:

- hue.query-processor.event-pipeline.cleanup.cron.expression
- hue.query-processor.event-pipeline.cleanup-interval-secs

For example:

```
“hue.query-processor.event-pipeline.cleanup.cron.expression” : “0 0 2 * * ?”
,
“hue.query-processor.event-pipeline.cleanup-interval-secs” : “2592000”
```

Manually cleaning up queries using an API

The ability to clean up queries manually in addition to the scheduled clean-up routines is useful when you have a high load of queries in a particular week that are hogging resources that you must free up. The API also runs a VACUUM command on the Query Processor table to reclaim storage that is occupied by dead tuples.

You can send an API request using tools such as cURL or Postman.

API format: `[***QUERY-PROCESSOR-ADDRESS***]/api/admin/cleanup/[***EPOCH-TIME***]`

Where,

- [***QUERY-PROCESSOR-ADDRESS***] is the query processor host address
- [***EPOCH-TIME***] is the Unix epoch time in seconds

Queries that were run before the specified epoch time are purged.

For example:

```
curl "http://machine1.company.com:30700/api/admin/cleanup/1670006742"
```

Downloading debug bundles

The debug bundle is a ZIP file that contains the query details in JSON format and an error-reports.json file, which is created only if an error occurs while the query is run.

About this task

Note: This feature is available only for Hive queries.

If Tez is used to run a query, then the debug bundle also contains DAG and Tez JSON files, as shown in the following image:

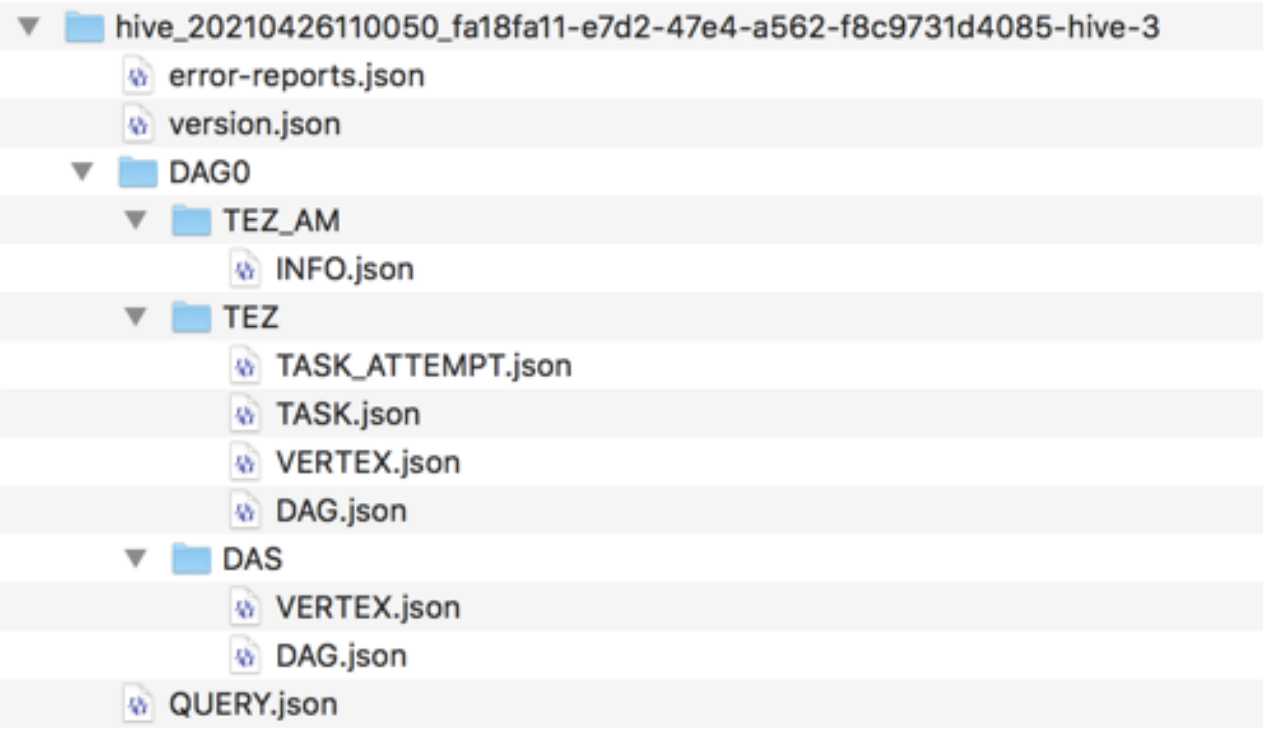

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.
   The **Job Browser** page is displayed.
3. Click Queries.
   The Hive queries that were run are displayed.
4. Select a query for which you want to download the debug bundle.
5. Click Download and save the ZIP file on your computer.
   The filename is in the following format:

```
hive_[***HIVE-QUERY-ID***]_[***USER-ID***]_[***UNIQUE-INDEX***]
```

Configuring Hue to handle HS2 failover

Hue can handle HiveServer2 (HS2) failovers using ZooKeeper without the need to set up a load balancer.

About this task

Note: Zookeeper configuration only supports failover situations. It cannot be used to balance loads between the active HS2 servers.

Procedure

1. Log in to Cloudera Manager as an Administrator.

2. Go to Clusters Hue Configuration and add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[beeswax]
hive_discovery_hs2=true
hive_discovery_hiveserver2_znode=/hiveserver2
```

Note: hiveserver2 is the default value of the hive_discovery_hiveserver2_znode property. If this value is different from the Hive-On-Tez configuration, then you must obtain the information from the hive.server2 .zookeeper.namespace property from Hive-On-Tez configuration.

3. Click Save Changes.
4. Restart the Hue service.

Enabling Spark 3 engine in Hue

Hue leverages Apache Livy 3 to support Spark SQL queries in Hue on the Apache Spark 3 engine. To enable the Spark 3 engine, specify the Livy server URL in the Hue Advanced Configuration Snippet using Cloudera Manager, and enable the Spark SQL notebook. Livy for Spark 3 and Spark 3 services are installed when you create a Cloudera Data Hub cluster with the Cloudera Data Engineering cluster template.

Before you begin

Note: Livy v0.6 supports Python versions upto Python 3.7. If you install Python 3.8 or higher, then you may see the following error: TypeError: required field "type_ignores" missing from Module.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters HDFS Configuration and add the following lines in the Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml field:

```
<property>
<name>hadoop.proxyuser.hue.groups</name>
<value>*</value>
</property>
<property>
<name>hadoop.proxyuser.hue.hosts</name>
<value>*</value>
</property>
<property>
<name>hadoop.proxyuser.spark.groups</name>
<value>*</value>
</property>
<property>
<name>hadoop.proxyuser.spark.hosts</name>
<value>*</value>
</property>
<property>
<name>hadoop.proxyuser.livy.groups</name>
<value>*</value>
</property>
<property>
<name>hadoop.proxyuser.livy.hosts</name>
<value>*</value>
</property>
```

3. Click Save Changes.
4. Go to Clusters Livy for Spark 3 service Configuration and add the following configurations:
   a) Add the hue user in the Admin Users (livy.superusers) field.
   b) Go to the HMS Service field and select Hive.
   c) Click Save Changes.
5. Go to Clusters SPARK_ON_YARN Configuration Admin Users , add hue to the list of admin users (spark.history.ui.admin.acls) and click Save Changes.
6. Go to Clusters SPARK Configuration Admin Users , add hue to the list of admin users (spark.history.ui.admin.acls) and click Save Changes.
7. Go to Clusters SPARK 3 Configuration Admin Users , add hue to the list of admin users (spark.history.ui.admin.acls) and click Save Changes.
8. Go to Clusters Hue Configuration and enter the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field and click Save Changes:

```
[desktop]
app_blacklist=zookeeper, pig #custom list of blocked apps
[spark]
#This is not a thrift server port
#If this TLS/SSL is enabled then check to see whether the livy url is on h
ttps or http and modify the url accordingly.
livy_server_url=http(s)://[***LIVY-FOR-SPARK3-SERVER-HOST***]:[***LIVY-
FOR-SPARK3-SERVER-PORT***]
ssl_cert_ca_verify=false
security_enabled=true
[notebook]
[[interpreters]]
[[[sparksql]]]
name=Spark SQL
interface=livy
```

   Attention:
   - Ensure that the Spark application is not on the blocked list.
   - Set ssl_cert_ca_verify=false if an SSL certificate is not present in Hue's truststore. ssl_cert_ca_verify=true if your environment is a secured environment.
   - Set security_enabled=true for Kerberized clusters.
9. Restart the affected services.

Results

You can now select the “Spark SQL” dialect on the Hue editor and run Spark queries from Hue.

Note: Starting a Livy session can take 30-45 seconds. If you cancel a running Spark SQL query and rerun it, the Livy session expires, and a new session is created.

Using Hue scripts

Hue scripts help you to share workflows, set up default editors, change ownership of documents, and test the backend processes using cURL commands. The Hue scripts are included with Cloudera.

Sharing workflows with users and groups

Use the following command to share workflows with users and groups:

```
./build/env/bin/hue share_all_workflows --owner [***OWNER-NAME***] --sharegr
oups [***GROUP1/USER1***],[***GROUP2/USER2***] --permissions read,write
```

To share users, use the --shareusers option instead of the --sharegroups option.

Changing the document owner

Use the following command to change the owner of the Hue documents:

```
./build/env/bin/hue change_owner_of_docs --olduser [***OLD-USER-NAME***] --n
ewuser [***NEW-USER-NAME***]
```

Testing the backend using cURL

Use the following command to get the syntax for the backend cURL commands or to test the endpoints:

```
./build/env/bin/hue get_backend_curl --showcurl --testname [***SERVICE-NAME-
WHERE-TEST-NEEDS-TO-BE-PERFORMED***] --service [***COMMA-SEPARATED-SERVICES-
TO-TEST]
```

Replace [***SERVICE-NAME***] with the name of the service where you want to perform the test. Replace [***COMMA-SEPARATED-SERVICES-TO-TEST***] with options or services such as all, httpfs, solr, oozie, rm, jhs, sparkhs, and so on.

Setting the default editor in Hue

Use the following command to set the default editor in Hue:

```
./build/env/bin/hue set_default_editor --hive/impala --username [***USERNAME
***]
```

The default value for the username is all.

Configurations for submitting a Hive query to a dedicated queue

When you run a Hive query from Hue or Beeline, HiveServer2 submits the query in YARN in the queue mapped from an end user instead of a hive user. If you run a query while specifying doAs=false, then the queries are submitted and run as a hive user, using impersonation. This is the default behaviour. You can change this behaviour by configuring the queue using "yarn_queue_manager" and "application tag" parameters in YARN.

If you see go to the **Job Browser** page in Hue and see that the Hive queries are appearing under the default queue, and not the actual user who ran the query, then this means that the queries are not being run in their dedicated queue in YARN. To submit a Hive query in its dedicated queue, you must enable application-tag based scheduling by configuring queue mapping and per queue properties.

Related Information

Configuring queue mapping to use the user name from the application tag using Cloudera Manager

Configure Per Queue Properties

Configuring timezone for Hue

You can configure the timezone for Hue using Cloudera Manager. The default value is America/Los_Angeles.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue Configuration Time Zone and set a desired value.

   The default value is “America/Los_Angeles”. You can refer to https://timezonedb.com/time-zones for information on the timezones.
3. Click Save Changes.
4. Restart the Hue service.

Cloudera Runtime 7.3.1

Atlas Audits

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Audit Operations....4
Atlas Type Definitions.... 6
Atlas Export and Import operations....7
Exporting data using Connected type.... 8
Atlas Server Operations....9
Audit enhancements....10
Examples of Audit Operations.... 12

Storage reduction for Atlas.... 16
Using audit aging....17
Enabling audit aging....17
Using default audit aging.... 17
Using Sweep out configurations.... 18
Using custom audit aging....19
Aging patterns....20
Audit aging reference configurations....20
Audit aging using REST API....21
Using custom audit filters.... 22
Supported operators.... 22
Rule configurations....23
Use cases and sample payloads....24

Audit Operations

As an Atlas administrator you can view the audit operations listed in the Audits tab in the Atlas UI. The audit data is captured implicitly by Atlas for various operations.

The audit operations include:

- Import
- Export
- Server start
- Server state active (In case of HA environment)
- Type Definition create
- Type Definition update
- Type Definition delete

The Type Definition can be of any type category listed:

- Business Metadata
- Classification
- Enum
- Entity
- Relationship
- Struct

Audit is collected for every create, update and delete operations.

Note: A user or a group must have the Admin Audits Ranger permission to access audits and related information.

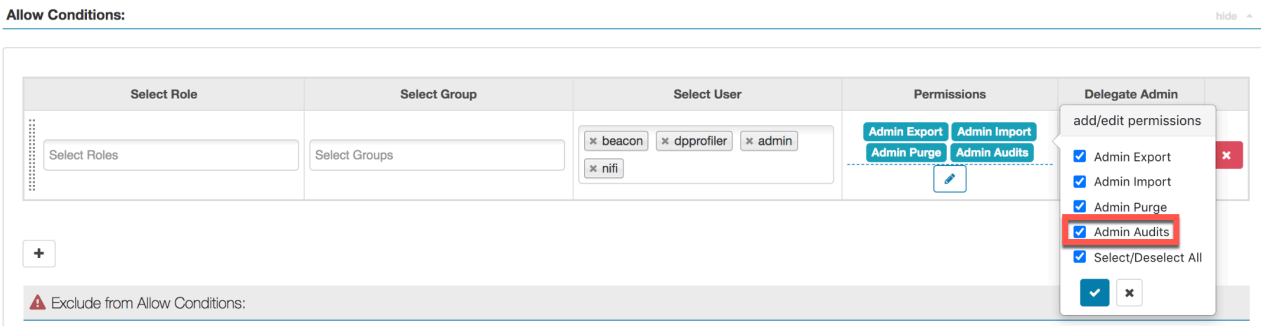

The JSON data is the payload which contains the actual data. You can submit the payload using the appropriate REST API tool.

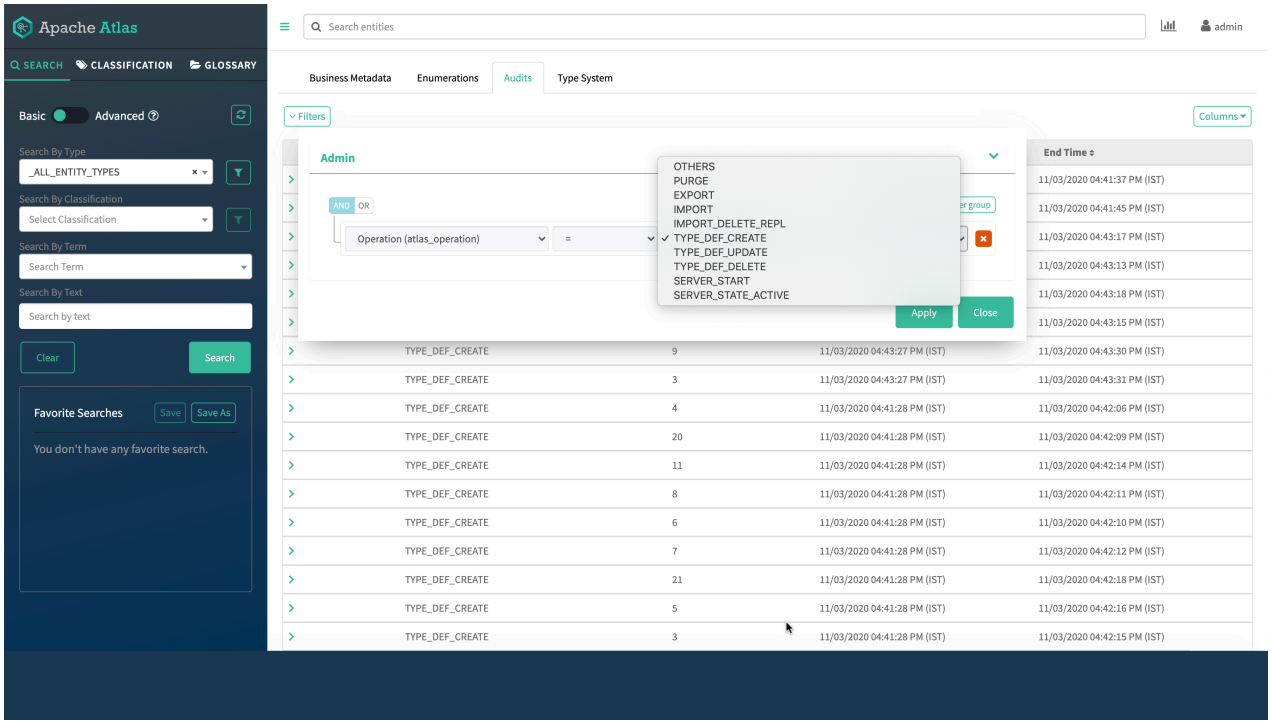

An audit entry logs the total number of Type Definitions that are created for create, update, and delete operations. Type Definitions are categorized according to entity types, struct types, Enum types, relationships, classification, and Business Metadata. For every Type Definition, the JSON data is stored in the audit entry.

Each audit entry logs the following details:

- Users - Indicates the user name of the user who performed the audit operation..
- Operation - Indicates an operation enum; can be used for searching audit entities as well
- Client ID - Indicates the IP address of the machine from which the request was generated.
- Result Count - Provides the total number of artifacts on which the operation was performed.
- Start Time - Indicates the actual time when the request was generated.
- End Time - Indicates the actual time when the requested operation was completed.
- Duration - Indicates the time taken by a request to complete the intended operation.

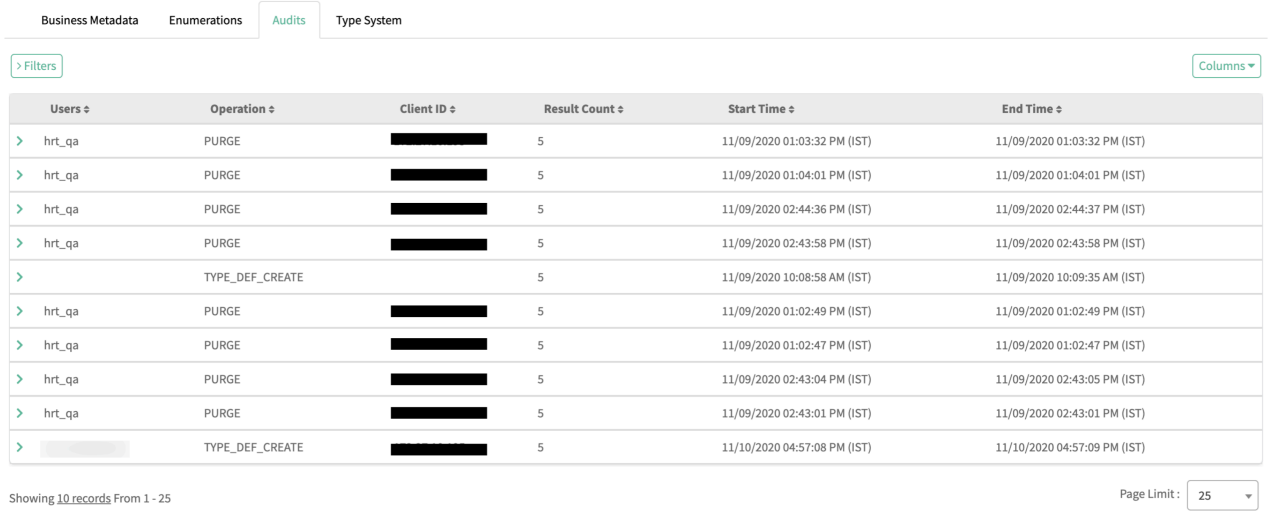

Atlas Type Definitions

Using Type Definitions, you can create, update, and delete entities of various types.

An example of Type Definition - Create

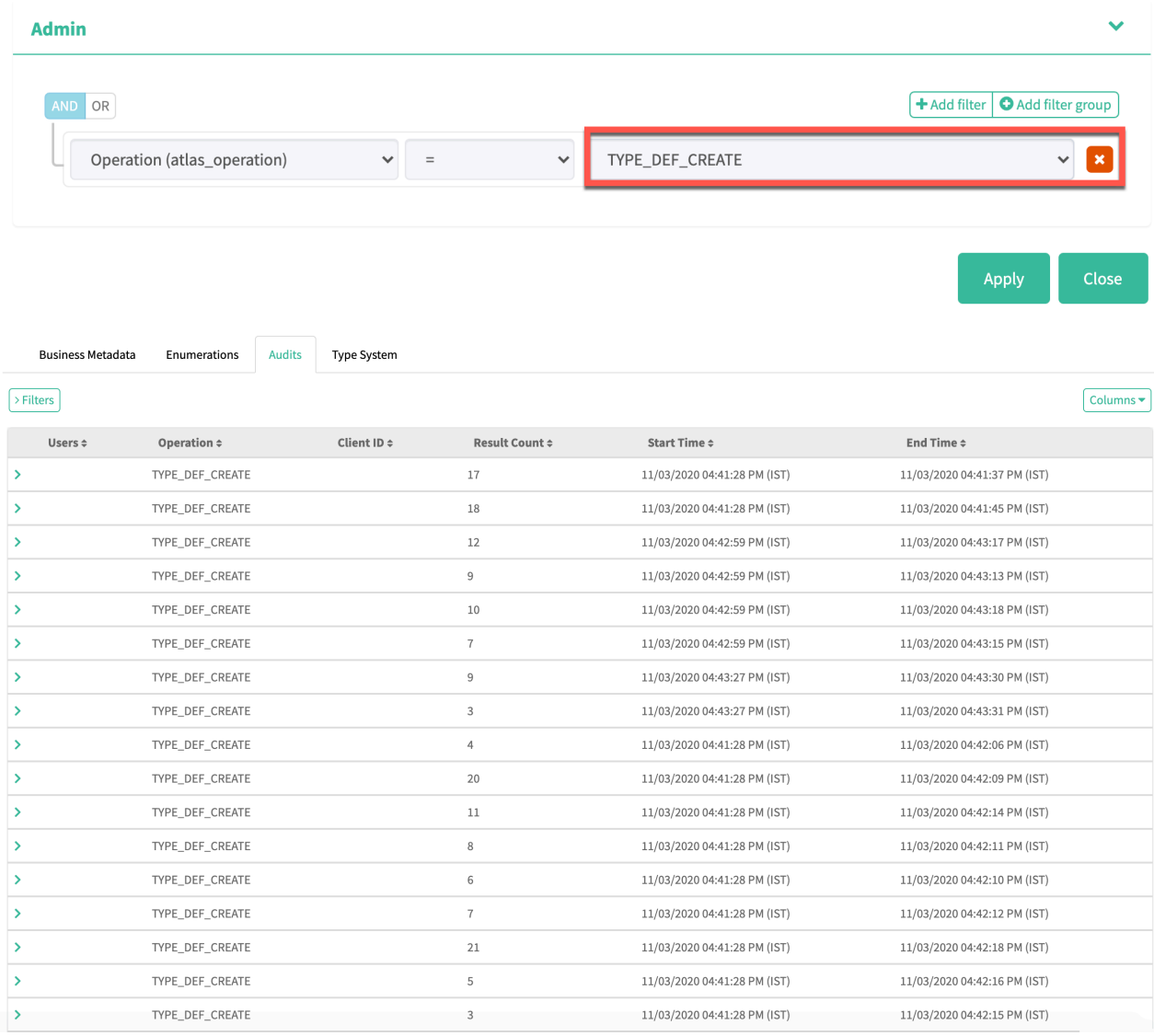

Users	Operation	Client ID	Result Count	Start Time	End Time
	TYPE_DEF_CREATE		17	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:37 PM (IST)
	TYPE_DEF_CREATE		18	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:45 PM (IST)
	TYPE_DEF_CREATE		12	11/03/2020 04:42:59 PM (IST)	11/03/2020 04:43:17 PM (IST)
	TYPE_DEF_CREATE		9	11/03/2020 04:42:59 PM (IST)	11/03/2020 04:43:13 PM (IST)
	TYPE_DEF_CREATE		10	11/03/2020 04:42:59 PM (IST)	11/03/2020 04:43:18 PM (IST)
	TYPE_DEF_CREATE		7	11/03/2020 04:42:59 PM (IST)	11/03/2020 04:43:15 PM (IST)
	TYPE_DEF_CREATE		9	11/03/2020 04:43:27 PM (IST)	11/03/2020 04:43:30 PM (IST)
	TYPE_DEF_CREATE		3	11/03/2020 04:43:27 PM (IST)	11/03/2020 04:43:31 PM (IST)
	TYPE_DEF_CREATE		4	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:42:06 PM (IST)
	TYPE_DEF_CREATE		20	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:42:09 PM (IST)
	TYPE_DEF_CREATE		11	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:42:14 PM (IST)
	TYPE_DEF_CREATE		8	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:42:11 PM (IST)
	TYPE_DEF_CREATE		6	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:42:10 PM (IST)
	TYPE_DEF_CREATE		7	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:42:12 PM (IST)
	TYPE_DEF_CREATE		21	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:42:18 PM (IST)
	TYPE_DEF_CREATE		5	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:42:16 PM (IST)
	TYPE_DEF_CREATE		3	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:42:15 PM (IST)

An example of Type Definition - Update

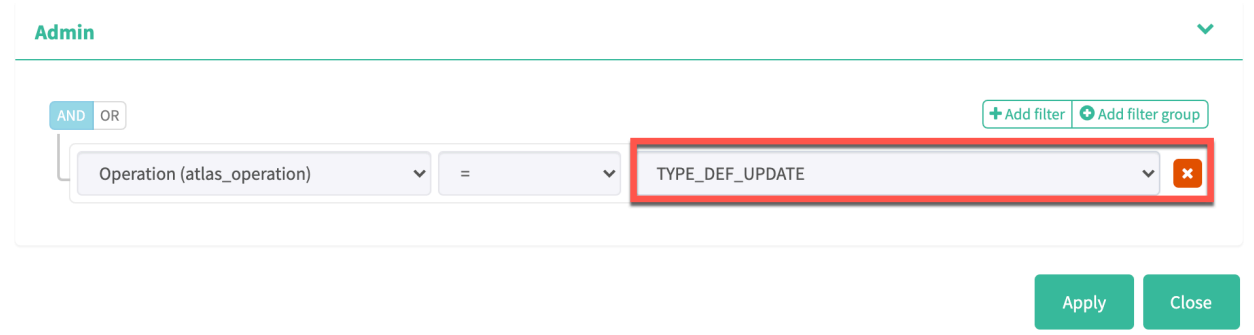

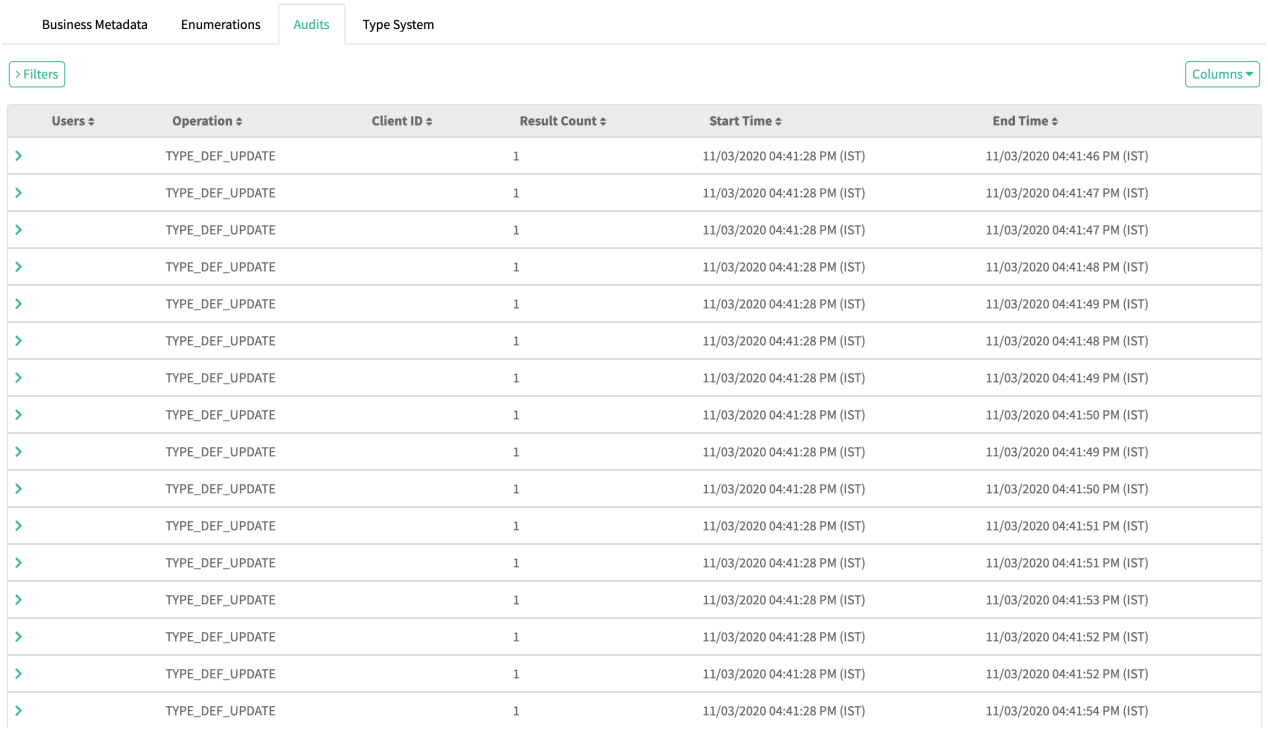

Users	Operation	Client ID	Result Count	Start Time	End Time
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:46 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:47 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:47 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:48 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:49 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:48 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:49 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:50 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:49 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:50 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:51 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:51 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:53 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:52 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:52 PM (IST)
	TYPE_DEF_UPDATE		1	11/03/2020 04:41:28 PM (IST)	11/03/2020 04:41:54 PM (IST)

An example of Type Definition - Delete

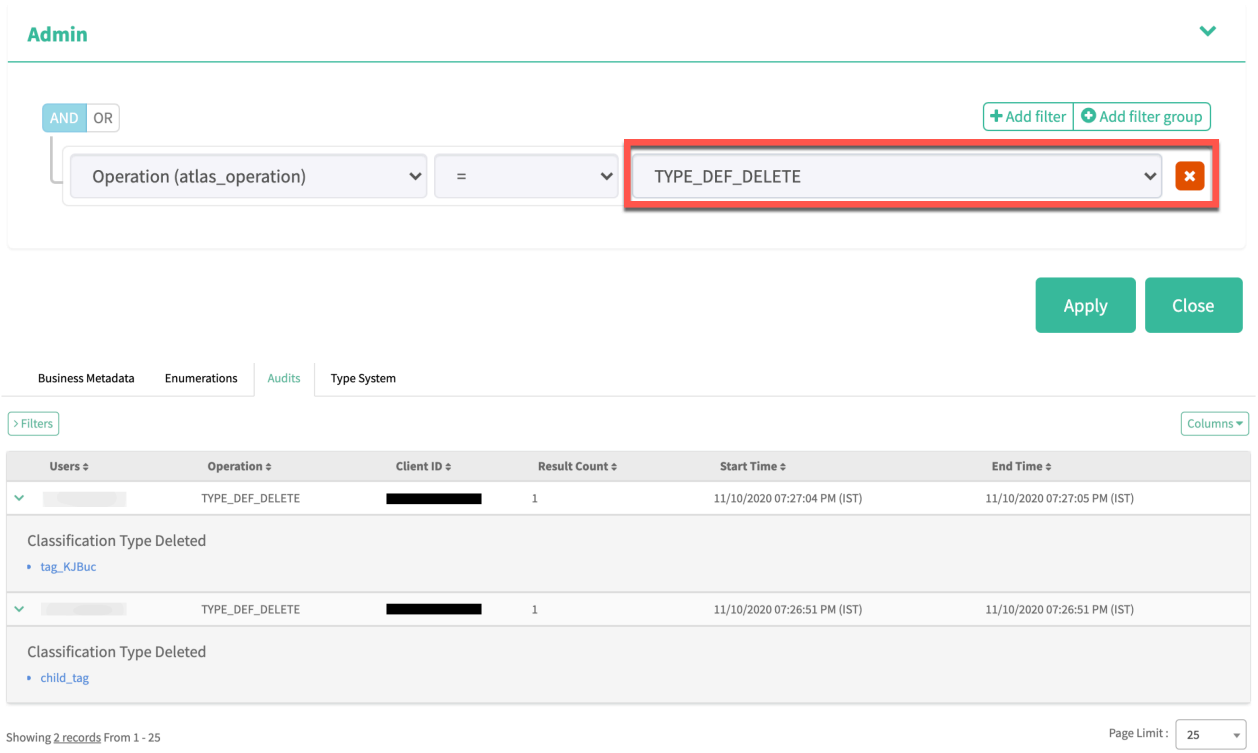

Atlas Export and Import operations

An audit entry is created for every export and import operation.

An example of - Export operation

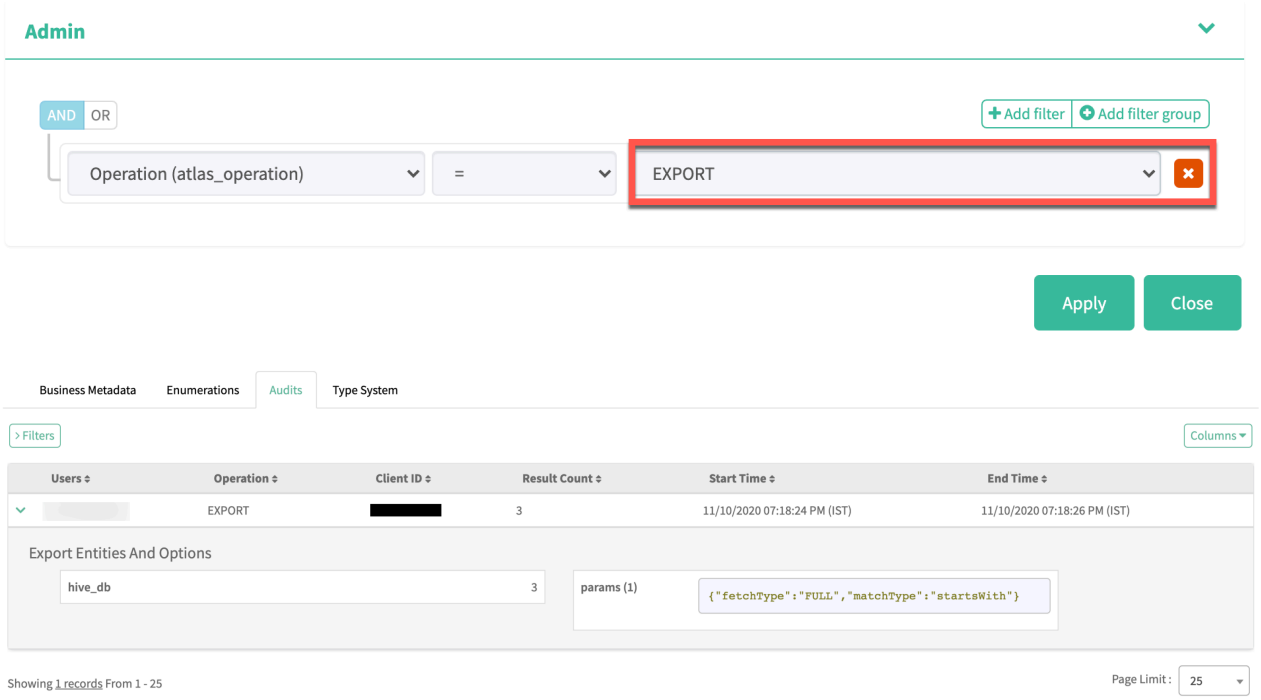

Each import audit entry provides information about the total number of entities imported, along with the number of entities imported with each Type Definition.

An example of - Import operation

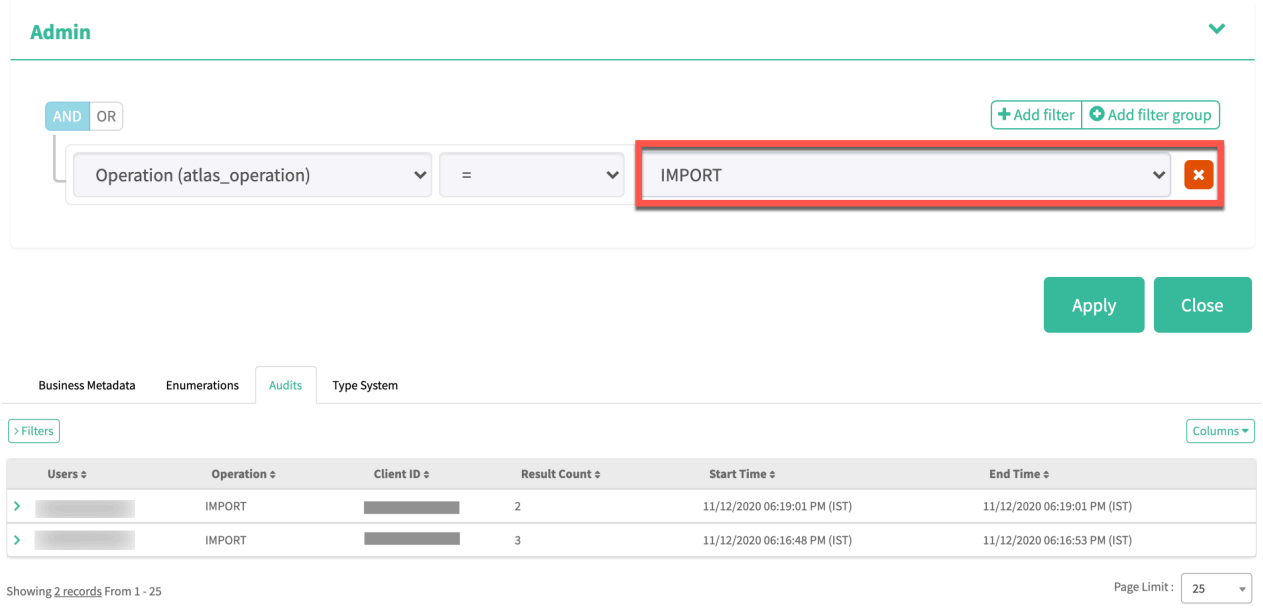

Exporting data using Connected type

As per Apache Atlas Software Foundation notes, only directly connected entities must be exported and when the data is exported with the starting entity as Hive table and the fetch type is "CONNECTED", the exported entities must not include the external and managed locations.

But the expected behavior is that all the entities which are directly connected entities get exported. Additionally, other dependent entities will be updated like the managed location of a database or associated database of a table which also gets exported.

For example:

```
db1.table1 --> p1 ---> db2.table2 ---> p2 ---> db3.table3 ---> p3 ---> db4.t
able4 --> p4 ---> db5.table5

Export db3.table3 with options
{
"itemsToExport": [{
"typeName": "hive_table", "uniqueAttributes":
{ "qualifiedName": "db3.table3@cluster0" }
}],
"options":
{ "fetchType": "connected" }
}
```

Result: The exported ZIP file must contain entities: db2, db3, db4, table2, table3, table4, p2, and p3.

Atlas Server Operations

When you perform server related tasks, audit entries are logged.

When the Apache Atlas server is started, an audit entry is logged. Also, when the server is started in the Active mode using High Availability (HA), an audit entry is logged. For more information, see About Atlas High Availability.

An example of - Server Start operation

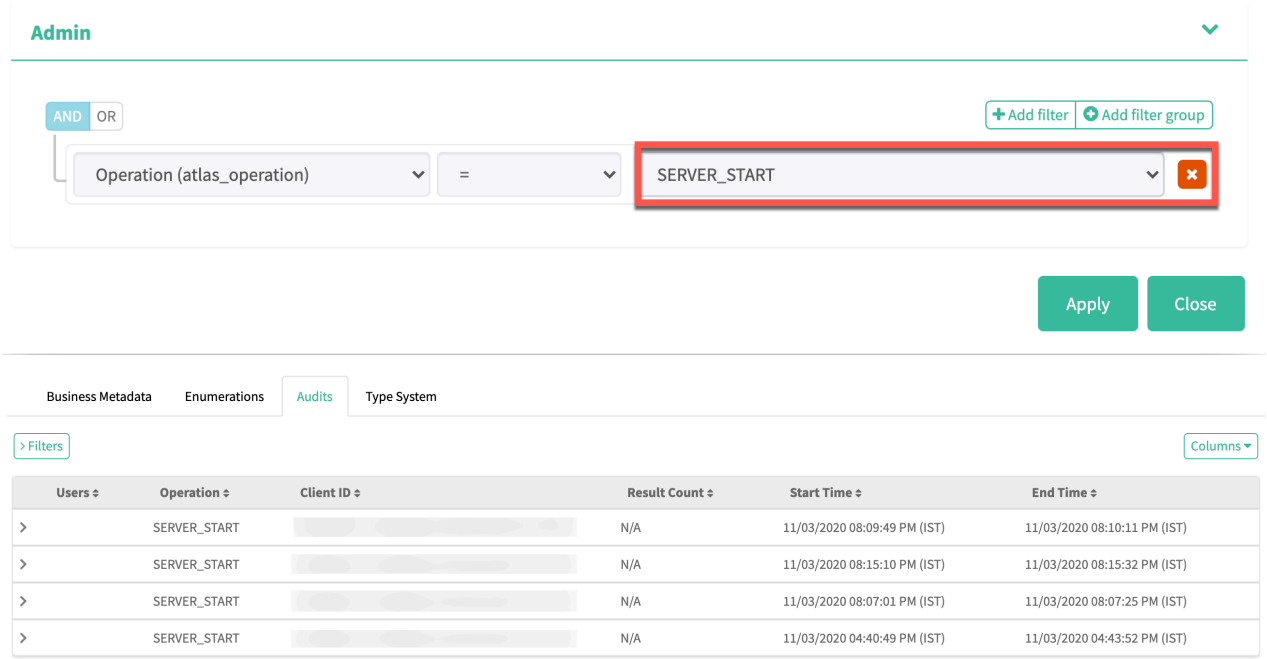

An example of - Server State Active operation

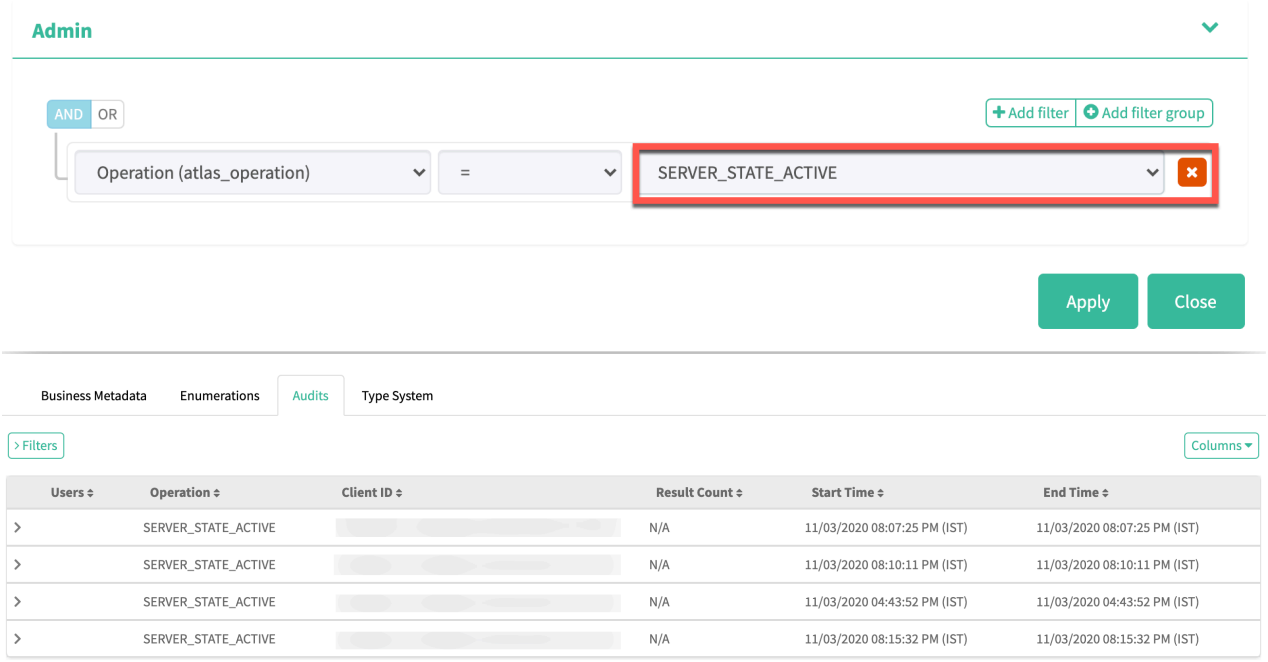

Related Information

About Atlas High Availability

Audit enhancements

When any entity is created in Apache Atlas, the created entity is stored in Apache HBase tables.

Currently, when the created entity is modified or updated, for example, entity core attributes, relationship attributes, custom attributes, or associated classifications, the changed entity is captured by Atlas either as a full entity object or partial object (with only updated attributes or relations) and stored in HBase tables.

While processing update requests, Atlas generates entity audit events and stores them in the HBase table. These audit events have complete entity information in the JSON format. Multiple updates on a single entity results in generating multiple audit events, each of which has a complete entity instance duplicated with minimal changes.

For example, if entity A1 is updated or modified for about five times, for every update, along with the changes (minimal), the entire entity is stored in the HBase tables. This process consumes additional storage space in HBase tables. In simple terms, A1 + the number of times the changes made is the resultant output that is saved in the HBase tables. Even if the changes are minimal (updating an attribute or relations or something that is not significant), Atlas captures the complete entity.

The Atlas audit enhancement optimizes storage space by not capturing the entire entity instance in each audit event. You can configure Atlas to store only differential information in audit events.

You must enable the application flag by setting the parameter using:

Cloudera Manager UI > Atlas > Configuration > Click Advanced (Under Category) > Enter the following parameter and the value under Atlas Server Advanced Configuration Snippet (Safety Valve) for confi/atlas-application.properties text field.

atlas.entity.audit.differential=true

To have significant savings in the HBase table memory footprint, only the difference between the original and updated entity state is captured in Atlas.

Previously, Atlas exhibited full entity as shown in the image:

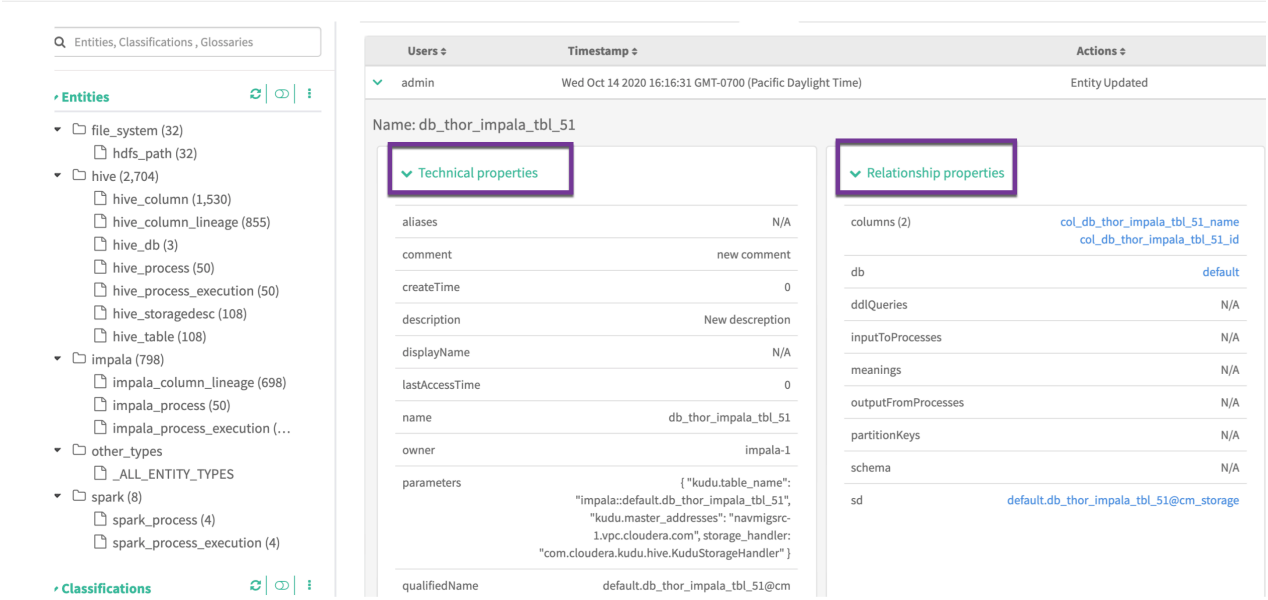

Currently, Atlas displays only the differences between the original and updated entity state as shown in the image

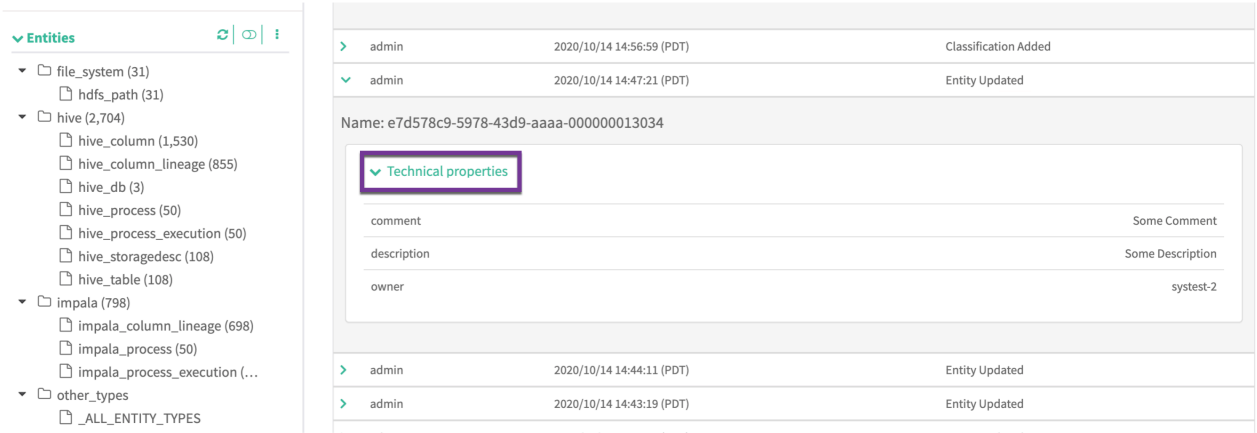

As a use case, previously when you add data under user-defined-properties, for example, key_1 and val_1, Atlas displayed the same as seen in the image.

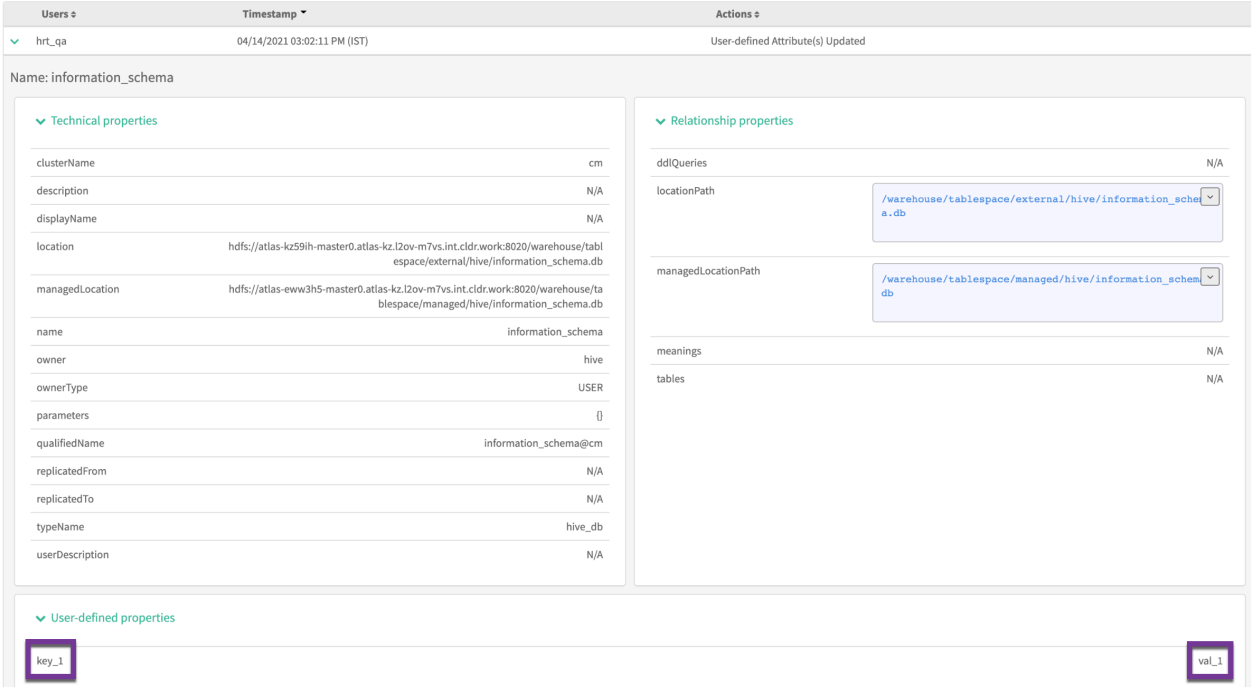

Currently Atlas displays only the changed or updated entity values.

Examples of Audit Operations

Some examples of payload submission for audit operations.

An example of creating a Type Definition:

```
enumDefs:
1.days_of_week
entityDefs:
1.Country
2.State
3.Vehicle

relationshipDefs:
```

```
1.country_state_rel
curl --location --request POST -u admin:admin 'http://car123.test1234.root
.hwx.site:23400/api/atlas/v2/types/typedefs' \
--header 'Content-Type: application/json' \
--data-raw '{
"enumDefs": [
{
"name": "days_of_week",
"typeVersion": "1.0",
"elementDefs": [
{
"ordinal": 1,
"value": "MONDAY"
},
{
"ordinal": 2,
"value": "TUESDAY"
},
{
"ordinal": 3,
"value": "WEDNESDAY"
},
{
"ordinal": 4,
"value": "THURSDAY"
},
{
"ordinal": 5,
"value": "FRIDAY"
},
{
"ordinal": 6,
"value": "SATURDAY"
},
{
"ordinal": 7,
"value": "SUNDAY"
}
]
}
],
"entityDefs": [
{
"category": "ENTITY",
"createdBy": "admin",
"updatedBy": "admin",
"createTime": 1537261952180,
"updateTime": 1537262097732,
"version": 1,
"name": "Vehicle",
"description": "desc Vehicle",
"typeVersion": "1.1",
"attributeDefs": [
{
"name": "no_of_wheels",
"typeName": "int",
"isOptional": true,
"cardinality": "SINGLE",
"valuesMinCount": 1,
"valuesMaxCount": 1,
"isUnique": false,
"isIndexable": false,
"includeInNotification": false
}
```

```
]
},
{
"category": "ENTITY",
"createdBy": "admin",
"updatedBy": "admin",
"createTime": 1537261952180,
"updateTime": 1537262097732,
"version": 1,
"name": "Country",
"description": "desc Country",
"typeVersion": "1.1",
"attributeDefs": [
{
"name": "ISD_CODE",
"typeName": "string",
"isOptional": false,
"cardinality": "SINGLE",
"valuesMinCount": 1,
"valuesMaxCount": 1,
"isUnique": false,
"isIndexable": false,
"includeInNotification": false
}
]
},
{
"category": "ENTITY",
"createdBy": "admin",
"updatedBy": "admin",
"createTime": 1537261952180,
"updateTime": 1537262097732,
"version": 1,
"name": "State",
"description": "desc State",
"typeVersion": "1.1",
"attributeDefs": [
{
"name": "STD_CODE",
"typeName": "string",
"isOptional": false,
"cardinality": "SINGLE",
"valuesMinCount": 1,
"valuesMaxCount": 1,
"isUnique": false,
"isIndexable": false,
"includeInNotification": false
}
]
}
],
"relationshipDefs": [
{
"name": "country_state_rel",
"typeVersion": "1.1",
"relationshipCategory": "AGGREGATION",
"endDef1": {
"type": "Country",
"name": "state_st",
"isContainer": false,
"cardinality": "SINGLE",
"isLegacyAttribute": true
},
"endDef2": {
```

```
"type": "State",
"name": "country_ct",
"isContainer": true,
"cardinality": "SET"
},
"propagateTags": "NONE"
}
]
}'
```

An example of updating a Type Definition:

```
enumDefs:days_of_week
entityDefs:Country
curl --location --request PUT -u admin:admin 'http://car123.car123-1.root.
hwx.site:31000/api/atlas/v2/types/typedefs' \
--header 'Content-Type: application/json' \
--data-raw '{
"enumDefs": [
{
"name": "days_of_week",
"typeVersion": "1.0",
"elementDefs": [
{
"ordinal": 1,
"value": "MONDAY"
},
{
"ordinal": 2,
"value": "TUESDAY"
},
{
"ordinal": 3,
"value": "WEDNESDAY"
},
{
"ordinal": 4,
"value": "THURSDAY"
},
{
"ordinal": 5,
"value": "FRIDAY"
},
{
"ordinal": 6,
"value": "SATURDAY"
},
{
"ordinal": 7,
"value": "SUNDAY"
},
{
"ordinal": 8,
"value": "HOLIDAY"
}
]
}
],
"entityDefs": [
{
"category": "ENTITY",
"createdBy": "admin",
```

```
"updatedBy": "admin",
"createTime": 1537261952180,
"updateTime": 1537262097732,
"version": 1,
"name": "Country",
"description": "desc Country Updated",
"typeVersion": "1.1",
"attributeDefs": [
{
"name": "ISD_CODE",
"typeName": "string",
"isOptional": false,
"cardinality": "SINGLE",
"valuesMinCount": 1,
"valuesMaxCount": 1,
"isUnique": false,
"isIndexable": false,
"includeInNotification": false
}
]
}
]
}'
```

An example of deleting a Type Definition:

```
curl --location --request DELETE -u admin:admin
          'http://car123.car123-1.root.hwx.site:31000/api/atlas/v2/types/typ
edef/name/Vehicle'
```

An example of exporting an Atlas entity:

```
'{
    "itemsToExport": [
            { "typeName": "DB", "uniqueAttributes": { "name": "Sales" }},
            { "typeName": "DB", "uniqueAttributes": { "name": "Reporting" }}
,
            { "typeName": "DB", "uniqueAttributes": { "name": "Logging" }}
    ],
        "options": { "fetchType": "full" }
    }'
```

For additional reference related to exporting entities, see https://atlas.apache.org/#/ExportAPI.

An example of importing an Atlas entity:

Performing an import operation should create an entry in the audits tab.

For additional reference related to exporting entities, see https://atlas.apache.org/#/ImportAPI.

Storage reduction for Atlas

Audit aging reduces the existing audit data in the Atlas system which is based on the end user criteria and configuration changes that users can manage. With this configuration, users can control the type of DDL events pertaining to schema changes that will be retained for a period of time determined by their compliance requirements.

Atlas creates audit events for creation, update, and deletion (CUD) of entities in Cloudera, including TAG/Classification and DML events. Notification processing with less significant metadata, exponential growth of audit events with no constraints, and retaining of audit data for long periods of time causes overhead on the application processing and consumes additional storage space, thereby causing operational inefficiencies.

Atlas supports various audit reduction strategies, pertaining to the type of audit data the users can store and the period of time that they can retain the data in the Atlas systems.

Some of the audit reduction options include:

- DML Audit Filter - A hook based filter that skips processing of DML events.
- Custom Audit Filters - Users can decide which audit events will be ignored.
- Audit Aging - Users can purge existing audit data based on the criteria they set.

Note: Audit reduction is targeted only for the entity audit data which persists in the ATLAS_ENTITY_AUDIT_EVENTS table. Filtering and aging data is exclusive to Admin Audit data which typically persists in the atlas_janus table.

Attention: Audit aging is employed for optimizing the existing audit events and Audit filters are used to prevent unnecessary audit events.

Using audit aging

You can employ various options to manage the volume of audit data stored and reduce the storage limits on your Atlas instance.

Audit aging overview

Using the audit aging feature helps you to manage the Atlas storage in an efficient manner. Based on certain conditions the Atlas audit data can be deleted or aged out.

Attention:

- Audit aging is allowed only for users with **admin-audits** policy.
- Solr should be up and healthy to perform Audit aging as it depends on Basic search (Solr based) to fetch the entity details.
- By default Audit Aging is disabled through REST API. Default aging is disabled by default in REST API, as the sole purpose of REST API is to trigger custom scenarios than default aging and also default aging option can be time consuming in case of larger or huge data.

Enabling audit aging

You can initiate and configure Audit aging feature in Atlas either though application-properties in Cloudera Manager and the REST API route.

For all the audit aging operations, the configuration can be enabled through application properties.

From Cloudera Manager > Configuration tab > Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties

Once the configuration is updated in Cloudera Manager, restart Atlas to enable audit aging and also follow the audit aging process based on the scheduled frequency.

You can specify the audit aging cases based on the configuration property named atlas.audit.aging.enabled=true. By default, this property is disabled. If needed, you must explicitly enable the property.

The following types of audit aging are supported:

- Default Audit Aging
- Sweep out Audits
- Custom Audit Aging

Using default audit aging

You can use the default audit aging mechanism to configure Time-to-Live (TTL), audit count limit, and disable default audit aging processes.

Using Time-to-Live (TTL) configuration

(TTL) is a value for the period of time that a packet, or data, should exist on a computer or network before being discarded.

Configuration to set TTL: atlas.audit.default.ageout.ttl.in.days

This configuration makes a final decision on how long the audit data can be retained in the database. This configuration is used by the Audit aging scheduler to delete all audit data when the audit data lifetime crosses the configured TTL. This configuration is applicable for all entity and audit action types.

By default only TTL will be configured for 90 days and no audit count will be considered.

Note: Using the REST API route, when default aging is enabled without any ttl configuration, default aging will be triggered with default ttl value which is 90 days.

As an example for TTL configuration usage, consider the following scenario::

You must maintain the entire audit data only for 40 days. The following configuration deletes audit data older than 40 days.

atlas.audit.default.ageout.ttl.in.day=40

Note: The parameter atlas.audit.default.ageout.ignore.ttl to be configured in case ttl is ignored and only audit count is to be considered for default aging.

Using Audit count limit parameter

Configuration to set allowed: atlas.audit.default.ageout.count

Using this configuration limits audit data for each entity. Atlas deletes all old audit data exceeding the configured audit count for all entities. This configuration is applicable for all entity and audit action types.

As an example for Audit limit count configuration usage, consider the following scenario:

You must maintain only the latest 20 audits for any entity.

atlas.audit.default.ageout.count=20

Using Disable Default Audit Aging parameter

Configuration to disable default audit aging: atlas.audit.default.ageout.enabled

For all entities, the process is default aging.This process consumes more time and resources. Under certain circumstances, if you want to execute only custom aging or sweep out features for minimal data, using this property default aging can be disabled.

By default, default audit aging is enabled.

Related Information

Using Sweep out configurations

Using custom audit aging

Using Sweep out configurations

You can employ the sweep out option for the entire audit data without any limitations or restrictions of TTL or minimum data count.

Caution: Use the sweep out feature carefully as any wrong usage can wipe out the entire audit data.

Sweep out option supports the following configurations to manage your audit data:

- The default Sweep out configuration is atlas.audit.sweep.out.enabled=true. The default value is false (disabled).

  To delete the entire audit data for specific entity and audit action types, you must configure at least one of the following properties:

  - atlas.audit.sweep.out.entity.types=<List of entity types> With sweep out option enabled, by default it is NOT applicable for any entity or action types. You can override the configuration with specific entity or action types.
  - atlas.audit.sweep.out.action.types=<List of audit action types> With sweep out option enabled, by default it is NOT applicable for any entity or action types. You can override the configuration with specific entity or action types.

Use cases detailing the usage of the Sweep out option using specific configurations.

Sweep out action	Applicable configuration
Delete entire audit data for hive_column and hive_storagedesc	atlas.audit.sweep.out.enabled=trueatlas.audit.sweep.out.entity.types
Delete entity update events for all entity types	atlas.audit.sweep.out.enabled=true atlas.audit.sweep.out.action.types=ENTITY_UPDATE
Delete all entity update events for hive_table	atlas.audit.sweep.out.enabled=true atlas.audit.sweep.out.entity.types=hive_table atlas.audit.sweep.out.action.types=ENTITY_UPDATE

Related Information

Using default audit aging

Using custom audit aging

Using custom audit aging

For additional flexibility to manage audit data, the custom audit aging mechanism is used to contrive the configuration of TTL and limit audit event count which is based on entity or audit action type.

Attention: Default configuration for custom aging is not supported. For custom aging to be in effect, users must specifically configure ttl/audit count and entityTypes/actionTypes.

Supported custom aging configurations:

- atlas.audit.custom.ageout.count=20
- atlas.audit.custom.ageout.ttl.in.days=30
- atlas.audit.custom.ageout.entity.types=<List of entity types>
- atlas.audit.custom.ageout.action.types=<List of audit action types>

Using these configurations, Atlas limits (with audit count) or age-out (with TTL) audit data for the configured entity and audit action types.

Note: Custom audit count and custom TTL configurations are applicable only when at least one of the custom entity or action types is configured.

Custom audit aging configurations are categorized using the following use cases:

Actions by	Description
Entity Type	Example: Limit to five latest audits for hive_column. atlas.audit.custom.ageout.count=5 atlas.audit.custom.ageout.entity.types=hive_column

By Action Type	Example: Delete all ENTITY_UPDATE audit events older than ten days. atlas.audit.custom.ageout.ttl.in.days=10 atlas.audit.custom.ageout.action.types=ENTITY_UPDATE
Limited audit by Action type for specific entity type	Example: Limit to five latest ENTITY_UPDATE audits for hive_storagedesc entities. • atlas.audit.custom.ageout.count=5 • atlas.audit.custom.ageout.entity.types=hive_storagedesc • atlas.audit.custom.ageout.action.types=ENTITY_UPDATE Example: Limit to five latest audits created in the last 1 week for hive_db entities • atlas.audit.custom.ageout.count=5 • atlas.audit.custom.ageout.ttl.in.days=7 • atlas.audit.custom.ageout.entity.types=hive_db

Attention: Regular expressions patterns are allowed patterns for action types and entity types.

The supported regex for entity types is entity type suffixed with '*'

For Example: hive_* considers all the entity types that start with hive_

The supported regex for action types is action type prefixed/suffixed with '*'.

For Example: CLASSIFICATION* or *CLASSIFICATION => Consider all the action types that contain CLASSIFICATION

Related Information

Using default audit aging

Using Sweep out configurations

Aging patterns

When you simultaneously perform multiple aging operations, certain features override the order in which the aging process takes place.

You must note about the following aging options and how they take effect.

- If the sweep out option is configured, sweep out entity types action types (if any) gets skipped from Custom aging and Default aging operations.
- If Custom aging is configured, custom entity types and action types (if any) get skipped from Default aging operations.
- Default aging skips the aging process for the entity and action types configured under custom and sweep out options and shall continue to process for all the remaining entity and action types.

Finally, if all the three aging options are simultaneously configured during the audit reduction cycle, Atlas follows the process in the following order of occurrence:

Sweep out -> Custom aging -> Default aging

Related Information

Audit aging reference configurations

Audit aging reference configurations

You must be aware about other related audit aging configurations that are required while working with optimizing the storage requirements and retaining the required audit data in your Atlas application.

The following configuration options provide you flexibility to manage your audit aging operations.

- By default all the aging options are applicable to only non **ENTITY_CREATE** events. Atlas ignores deleting all **ENTITY_CREATE** / **ENTITY_CREATED_BY_IMPORT** events by default. If users need

to delete audit data, including entity creation events, they can configure this property to true. By default it is configured as false. If you set the value of atlas.audit.create.events.ageout.allowed=true , **ENTITY_CREATE / ENTITY_IMPORT_CREATE** events will be considered as any other non-create events and will be deleted.

- The frequency for performing the audit aging process is moderated by using the atlas.audit.aging.scheduler.frequency.in.days=10 parameter. By default the audit aging process is scheduled for every 30 days.
- You cannot directly configure the custom or default TTL value less than the minimum TTL configured for this property - min.ttl.to.maintain. If the TTL value is configured for less than the minimum TTL, Atlas considers TTL as the value configured for the parameter. The default value is 7.
- You cannot directly configure custom or default audit count less than the minimum audit count configured for this property - min.audit.count.to.maintain. If you configure a lesser value, Atlas considers audit count as the value configured for the parameter. The default value is 50.
- By default audit aging will be executed for all the subtypes of the entity types configured using the property atlas.audit.aging.subtypes.included. By default it is configured as true. To limit audit aging operation for configured entity types but not sub types, this configuration can be used to exclude the subtypes. For example, the iceberg_table is a subtype of hive_table and if the user wants to purge audit data for only hive_table, in that case this configuration can be set to false.

Related Information

Aging patterns

Audit aging using REST API

Optionally, Atlas triggers Audit aging through REST API configuration. Using REST API to enable Audit aging features can be less time consuming and beneficial. By default Audit Aging is disabled through REST API.

Examples for employing REST APIs in specific scenarios:

- For a single instance, when you can perform an audit aging process for any specific scenario without changing any scheduled audit aging configuration, you can trigger audit aging using the REST API by passing the specified audit criteria as a payload.
- When regular configuration based Audit aging is scheduled for every 30 days, and user wants to trigger it before the next scheduled time, it can be done through REST API by passing the query parameter

```
useAuditConfig
```

Note: Default aging option is disabled by default while using the REST API option.

An example usage of the REST API feature:

```
POST /api/atlas/admin/audits/ageout?useAuditConfig=false
Payload options:
{
    “auditAgingEnabled”:false,
    “defaultAgeoutEnabled”:false,
    “createEventsAgeoutAllowed”:false,
    “subTypesIncluded”:true,
    “defaultAgeoutAuditCount”:10,
    “defaultAgeoutTTLInDays”:30,
    “customAgeoutAuditCount”:10,
    “customAgeoutTTLInDays”:30,
    “customAgeoutEntityTypes”:”hive_table”,
    “customAgeoutActionTypes”:”ENTITY_UPDATE”,
    “auditSweepoutEnabled”:true,
    “sweepoutEntityTypes”:”hive_column”,
    sweepoutActionTypes:”CLASSIFICATION_ADD”
}
```

Using custom audit filters

Customizing Atlas audit filters using the users' criteria provides you with the flexibility to finalize which audit events can be persisted with and which of the events can be ignored.

Custom audit filters are configured in the Atlas server which improves the performance of audit trail and an enhancement towards audit reduction. Using this feature, you can set some predefined rules or conditions that enable you to filter certain events that queues up the storage space in Atlas.

Common example use-cases:

- User might want to skip audit for all temporary tables.
- User might want to skip all entities name starts with "test".
- User might want to skip all update events for all entities.
- All types of operation can be allowed for auditing.
- All types of operation can be ignored from auditing.
- You can accept/discard audits by one specific attribute or combination of multiple attributes. Example attributes include, typeName, isIncomplete, temporary, and attributes (like owner, qualifiedName, name).

The audit filters operate based on a logic that transforms into a rule that results in a single action. The logic executes rules stored in the database to perform the following actions:

- Retrieve appropriate rules stored in the database
- Evaluate one or more rules.
- Match the rule conditions.
- Perform one or more actions that match the conditions.
- Result orientation.
- Repeat the process for the remaining rules.

Custom audit filters are supported using the REST API method.

The following REST API calls are used to perform various custom audit filter operations.

- POST: /api/atlas/admin/audits/rules
- GET: /api/atlas/admin/audits/rules
- PUT: /api/atlas/admin/audits/rules/{guid}
- DELETE: /api/atlas/admin/audits/rules/guid/{guid}
- DELETE: /api/atlas/admin/audits/rules
- DELETE: /api/atlas/admin/audits/rules/all

Supported operators

Operators currently supported for custom audit filters are numeric, boolean, and string.

Table 1: Supported operations types

Numeric operations	Boolean operations	String operations
“<”: less than “>”: greater than “<= : less than or equal to “>=: greater than or equal to	“==”: equals to “!=”: not equals to	“startsWith”: starts with “endsWith”: ends with “contains”: contains (case-sensitive) “notContains”: does not contain (case-sensitive) “isNull”: is null “notNull”: is not null “containsIgnoreCase”: contains (case-insensitive) “notContainsIgnoreCase”: does not contain (case-insensitive)

A typical rule is defined in the following manner:

```
{
  "desc": "Discard all hive_table audits with name containing test",
  "action": "DISCARD",
  "ruleName": "test_rule_1",
  "ruleExpr": {
      "ruleExprObjList": [
          {
              "typeName": "hive_table",
              "attributeName": "name",
              "operator": "contains",
              "attributeValue": "test"
          }
      ]
  }
 }
```

Related Information

Rule configurations

Rule configurations

You must use the following configurations to set up the custom audit filters.

Use the following configurations to control the usage:

From Cloudera Manager > Configuration tab > Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties

Enabling custom filters:

atlas.entity.audit.filter.enabled=false

By default this feature is disabled.

When no rules are created or defined, the default action is to accept the rule. The value can be changed to discard using the following parameter.

atlas.entity.audit.filter.default.action=DISCARD

Note: Permitted values are ACCEPT and DISCARD

Related Information

Supported operators

Use cases and sample payloads

Use cases and sample payloads

Assuming the default action is to ACCEPT an audit and the user wants to discard the audits conditionally, you must understand the rules payload for some of the common use case scenarios.

Discard temporary and test hive_table audits (Nested rules example)

```
"action":"DISCARD",
"ruleName":"test_rule_1",
"ruleExpr":{
   "ruleExprObjList":[
      {
         "typeName":"hive_table",
         "condition":"AND",
         "criterion":[
            {
               "operator":"==",
               "attributeName":"temporary",
               "attributeValue":"false"
            },
            {
               "condition":"OR",
               "criterion":[
                  {
                     "operator":"==",
                     "attributeName":"name",
                     "attributeValue":"tmp"
                  },
                  {
                     "operator":"==",
                     "attributeName":"qualifiedName",
                     "attributeValue":"tmp"
                  }
               ]
            }
         ]
      }
   ]
}
```

Discard all audits of a type

```
"action":"DISCARD",
"ruleName":"test_rule_1",
"ruleExpr":{
   "ruleExprObjList":[
      {
         "typeName":"hive_table"
      }
   ]
}
```

Discard all update audits for all entities

```
"action":"DISCARD",
"ruleName":"test_rule_1",
"ruleExpr":{
   "ruleExprObjList":[
      {
         "typeName":"”_ALL_ENTITY_TYPES”",
         "operator":"==",
```

```
            "attributeName":"operationType",
            "attributeValue":"ENTITY_UPDATE"
         }
      ]
   }
```

Discard all CLASSIFICATION_ADD audits for all entities

```
"action":"DISCARD",
"ruleName":"test_rule_1",
"ruleExpr":{
   "ruleExprObjList":[
      {
         "typeName":"_ALL_ENTITY_TYPES",
         "operator":"==",
         "attributeName":"operationType",
         "attributeValue":"CLASSIFICATION_ADD"
      }
   ]
}
```

Discard audits for entity DELETE operation types such as ENTITY_DELETE, ENTITY_IMPORT_DELETE, CLASSIFICATION_DELETE, PROPAGATED_CLASSIFICATION_DELETE, and LABEL_DELETE.

```
{
  "desc":"test3",
  "action":"DISCARD",
  "ruleName":"rule123",
  "ruleExpr":{
     "ruleExprObjList":[
        {
           "typeName":"hive_table",
           "condition":"AND",
           "criterion":[
              {
                 "operator":"contains",
                 "attributeName":"operationType",
                 "attributeValue":"DELETE"
              }
           ]
        }
     ]
  }
```

Usage of DELETE API for multiple rules: specify array of guids using comma separator in payload

```
URL: api/atlas/admin/audits/rules
Payload : ["477d8fcd-3d89-4c4c-bb91-9586c49fbd19","8b2b37a8-30eb-4510-b19a
-589816e45b80"]
```

Usage of DELETE API to delete all rules

```
URL: api/atlas/admin/audits/rules/all
Payload : not required
```

Discard audits for hive tables where description is null

```
Payload:
{
   "action": "DISCARD",
```

```
    "ruleName": "hiverule3",
    "ruleExpr": {
        "ruleExprObjList": [
            {
                "typeName": "hive_table",
                "includeSubTypes": "false",
                "attributeName": "description",
                "operator": "isNull"
            }
        ]
    }
```

Discard audits of event for a specific type based on attribute value

```
"action":"DISCARD",
"ruleName":"test_rule_1",
"ruleExpr":{
   "ruleExprObjList":[
      {
         "typeName":"hive_table",
         "condition":"AND",
         "criterion":[
            {
               "operator":"==",
               "attributeName":"operationType",
               "attributeValue":"ENTITY_UPDATE"
            },
            {
               "operator":"==",
               "attributeName":"name",
               "attributeValue":"employee"
            }
         ]
      }
   ]
}
```

Discard audits for all types under a hook type (Regex supported with wildcard character *)

```
"action":"DISCARD",
"ruleName":"test_rule_1",
"ruleExpr":{
   "ruleExprObjList":[
      {
         "typeName":"hive*",
         "operator":"==",
         "attributeName":"operationType",
         "attributeValue":"CLASSIFICATION_ADD"
      }
   ]
}
```

Note: All audits of Hive table, Hive column, DB, storage desc, and process are discarded for CLASSIFICATION_ADD event.

Discard all audits of a type and its sub types

```
"action":"DISCARD",
"ruleName":"test_rule_1",
"ruleExpr":{
   "ruleExprObjList":[
```

```
        {
            "typeName":"Asset",
            "includeSubTypes":true
        }
    ]
}
```

Note: All asset type audits are discarded.

CSV of type-names is supported

```
"action":"DISCARD",
"ruleName":"test_rule_1",
"ruleExpr":{
   "ruleExprObjList":[
      {
         "typeName":"hive_table,hbase_table",
         "attributeName":"name",
         "operator":"contains",
         "attributeValue":"test1"
      }
   ]
}
```

Related Information

Rule configurations

Cloudera Runtime 7.3.1

Configuring Apache Atlas

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring and Monitoring Atlas..... 4
Showing Atlas Server status.....4
Accessing Atlas logs.....5

Configuring and Monitoring Atlas

Cloudera Manager manages Atlas as a service, including monitoring its health and collecting its logs. Use Cloudera Manager to configure additional health checks, set up extractors, and set other Atlas properties.

Typically, you would need to make changes to Atlas configuration for the following reasons:

- Expanding resources for the Atlas server. For example in Cloudera Data Center, the initial memory allocation for Atlas is 2 MB.
- Changing Atlas security settings to add security levels or update parameters for Knox, LDAP, TLS, and Kerberos.
- Updating Atlas for other environment changes such as log directory locations.

To start or stop metadata collection from a service, make the configuration change in the properties for that service.

Atlas collects metadata for services in a given cluster managed by Cloudera Manager. If Cloudera Manager manages more than one cluster, you may have multiple instances of Atlas, one in each cluster. When you check the health of Atlas or configure its properties, make sure you are working in the correct cluster.

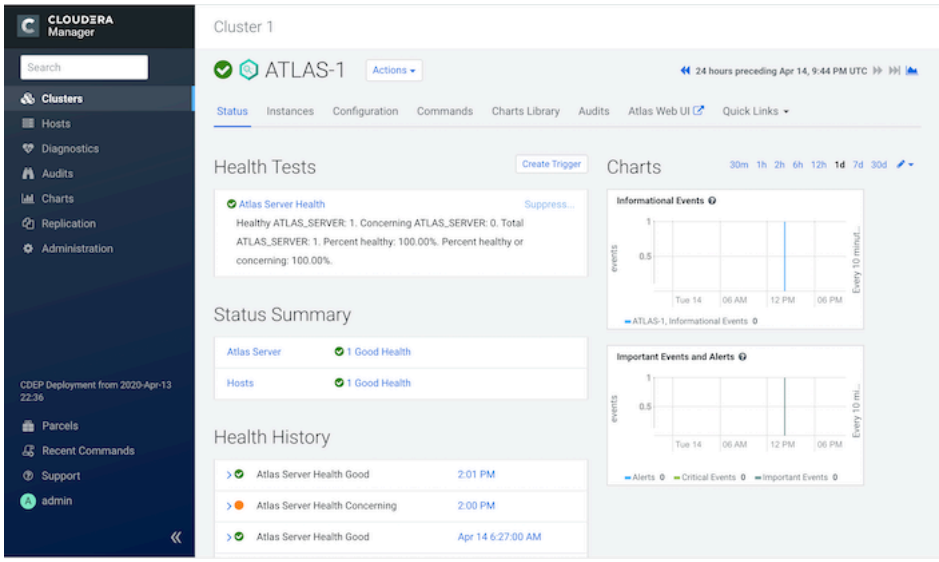

Here are some of the Atlas operations you can perform from Cloudera Manager:

Related Information

Changing the Configuration of a Service or Role Instance

Showing Atlas Server status

Cloudera Manager shows the status of the Apache Atlas Server role running in a given cluster. To understand if the Atlas UI is up and working for users, check the health of the other services on which Atlas depends: HBase, Kafka, and Solr.

The high-level status that Cloudera Manager shows for Atlas indicates the health of the Atlas Service. If you are experiencing problems with the Atlas UI or with metadata not showing up as expected, check to make sure the dependent services are also healthy.

To view the high-level status of Atlas and its dependent services:

1. Open the Cloudera Admin Console.
   - public cloud From Cloudera Public Cloud, access Cloudera Manager through your Data Lake.

2. Choose Clusters <*CLUSTER NAME*> . The cluster detail page shows the high-level status of each service. Check Atlas, HBase, Kafka, and Solr.

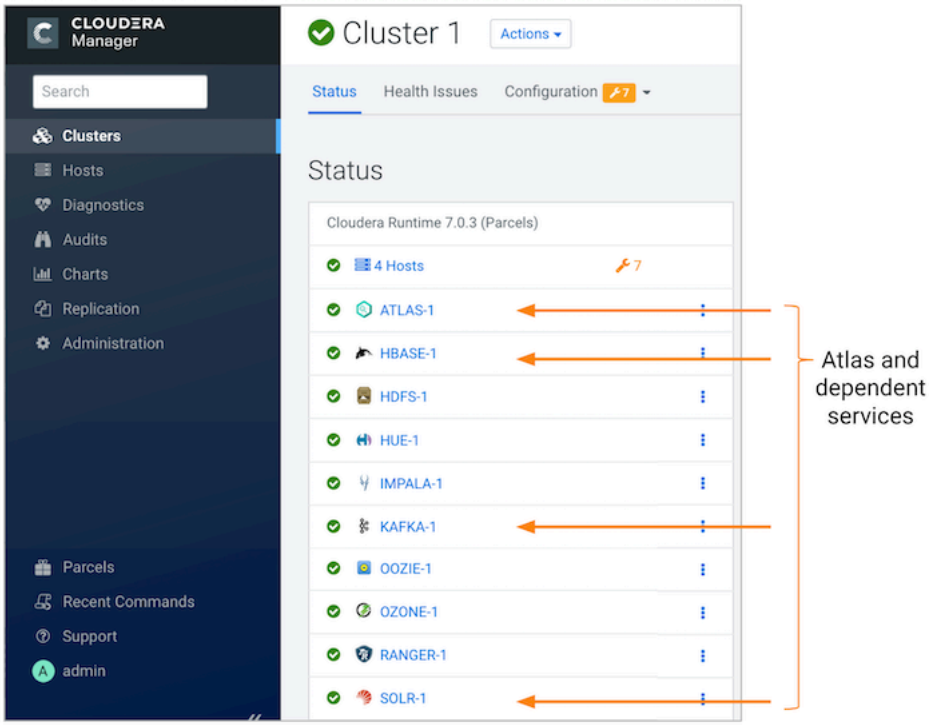

Understanding Atlas status

The Atlas status bubbles up to a single health indicator, which appears in many places including the service name

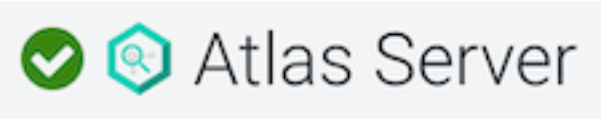

.

To see more detail of the Atlas status:

1. Click the links in the top-level status to drill into the recent health tests and events.
2. Drill into specific health test to open the detail page for the test.

If the Atlas status is anything but a green check for "healthy", you can use the content of the page to identify the problem. Check the following:

- If Atlas is stopped, was it stopped deliberately or did it shutdown on its own? To restart the service, see Restarting a Cloudera Runtime Service. To review what may have caused Atlas to shutdown, click the status icon to open a more detailed list of recent events.
- If Atlas shows a red exclamation point for "bad" health, check to see if one of the services it depends upon is stopped: HBase, Kafka, and Solr all run in the same cluster to support Atlas.

Related Information

Cloudera Manager Health Tests

Viewing Service Status in Cloudera Manager

Accessing Atlas logs

You can view and download Apache Atlas logs through Cloudera Manager.

To access the Atlas Server log:

1. Open the Cloudera Admin Console.
   - public cloud From Cloudera Public Cloud, access Cloudera Manager through your Data Lake.
2. Choose Clusters<*CLUSTER NAME*>.
3. Open the Atlas service page by clicking the Atlas Server instance, typically "ATLAS-1".
4. In the Atlas Server detail page, click the Atlas Service role link.
5. In the **Log Files** menu, select Role Log File.

   This option opens the Atlas application.log file, showing the latest contents of the log. You can filter the entries to a particular date range and save the log locally (Download button).

Cloudera Runtime 7.3.1

Atlas Lineage

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Lineage overview....4

Viewing lineage.... 4

Lineage lifecycle.... 7

Support for On-Demand lineage....7

HDFS lineage data extraction in Atlas....9

Prerequisites for HDFS lineage extraction....9
HDFS lineage commands.... 11
Running HDFS lineage commands.... 11
Inclusion and exclusion operation for HDFS files.... 15
Supported HDFS entities and their hierarchies....16

Lineage overview

Atlas lineage helps you understand the source and impact of data and changes to data over time and across all your data.

Lineage information helps you understand the origin of data and the transformations it may have gone through before arriving in a file or table. In Atlas, if transformations occurred in services that provide process metadata, a lineage graph shows how data in a given column was generated. When a column appears in the output of a process, Atlas reads the content of the process and links the input column or columns to the output asset. This relationship is stored as a vertex in Atlas's graph database. It is displayed as a lineage graph in the details of each entity.

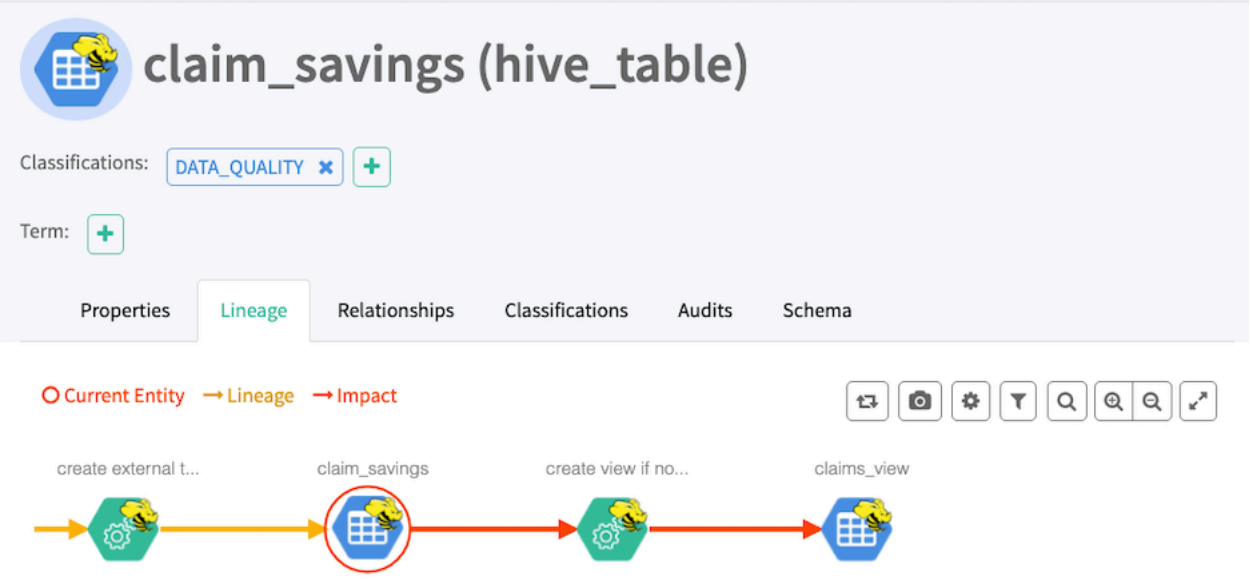

By default, Atlas can collect lineage information from the following sources:

- HiveServer
- Impala
- Spark

The lineage metadata produced by these sources may refer to additional asset entities. For example, when a Hive operation moves data from a Hive table to an HBase table or an HDFS file, Atlas includes an entity to represent the HBase table or HDFS file, and the entity is connected to the Hive table through lineage. The following sources may appear in lineage graphs when referenced:

- HBase
- HDFS
- S3

Data flow lineage from Cloudera Flow Management (NiFi) can be included as well by configuring the appropriate reporting task.

Viewing lineage

Atlas lineage graphs include lots of detail that you can reveal and configure.

Use your cursor to explore a lineage graph:

- Click to show details for an entity

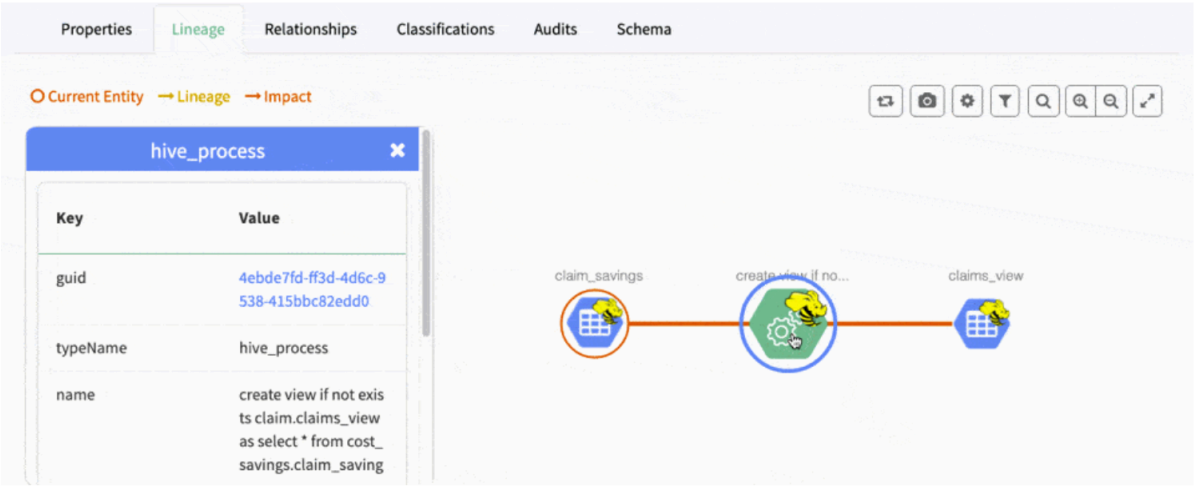

- Hover over an entity to show only one ancestor and descendant

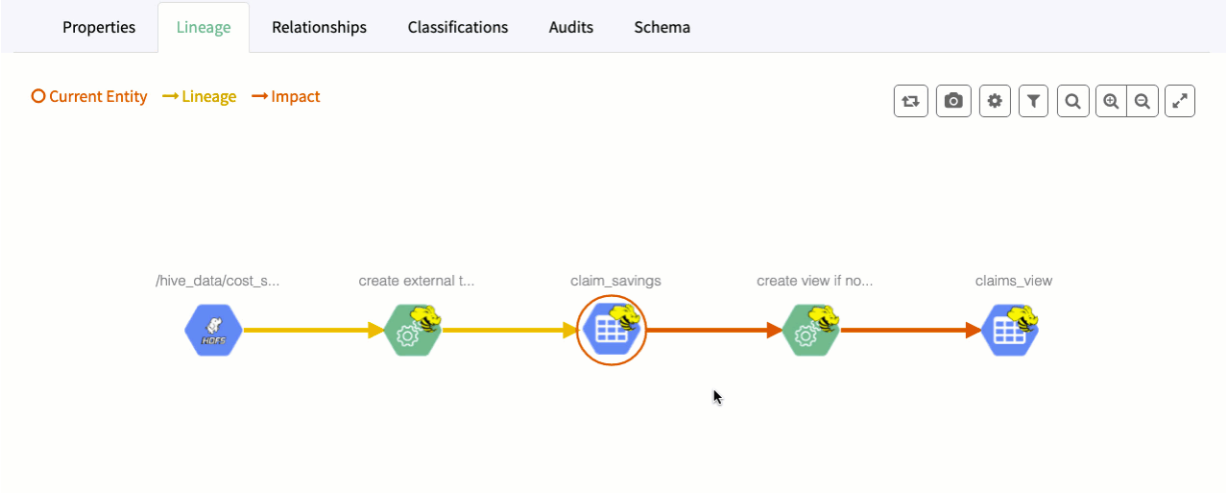

The following symbols can appear in lineage graphs:

Symbols	Name	Description and Actions
	Data Set entity	Represents a column, table, view, file, database, or other physical or logical data asset. While all data set entities are shown as a blue hexagon, the icon may vary based on the type of data asset and the source system.
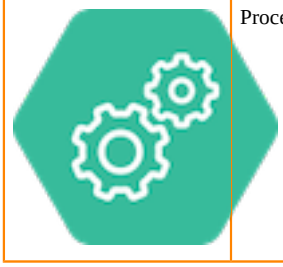	Process entity	Represents a query, job, or other process that applied to data assets. While all process entities are shown as a green hexagon, the icon may vary based on the type of process and the source system.

Symbols	Name	Description and Actions
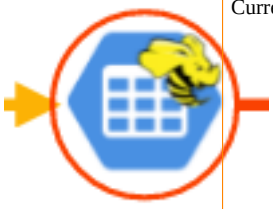	Current entity	A red circle indicates the current entity in the lineage graph.
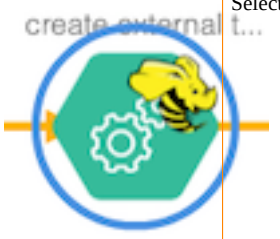	Selected entity	A blue circle indicates the entity selected in the lineage graph.
	Lineage	Connects the current entity with entities that provided input data. Entities connected with yellow lines are ancestors of the current entity.
	Impact	Connects the current entity with entities that could have an impact if data or schema information is changed in the current entity. Entities connected with red lines are descendants of the current entity.

The upper right corner of the lineage picture includes the following controls:

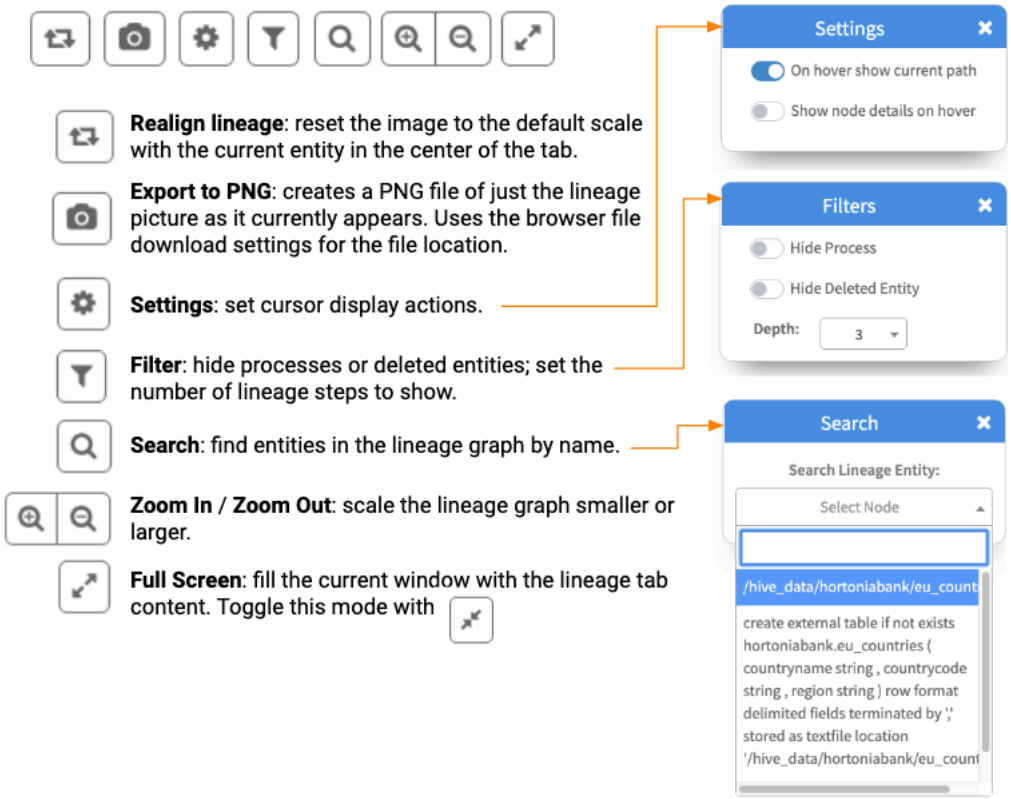

Related Information

Propagating classifications through lineage

Lineage lifecycle

Tables are dropped, schemas change, views are created: lineage tracks these changes over time.

Atlas reads the content of the metadata it collects to build relationships among data assets. When Atlas receives query information, it notes the input and output of the query at the column level: Atlas generates a lineage map that traces how data is used and transformed over time. This visualization of data transformations allows governance teams to quickly identify the source of data and to understand the impact of data changes.

Atlas processes contain lineage info; data assets by themselves do not. Impala queries are represented as processes and have lineage information; the data asset affected by Impala queries appear as Hive entities.

HDFS, S3, ADLS files appear when they are referenced by Hive, Impala, or Spark queries; operations that occur on the file system are not reflected in Atlas lineage.

The contents of a lineage graph are determined by what metadata is collected from services. If a process refers to a data asset but Atlas doesn't have an entity for that data asset, Atlas isn’t able to create an entity for the process and the lineage defined by that process won’t appear in Atlas.

Deleted data assets

Entities that represent data assets that have been deleted (such as after a DROP TABLE command in Hive) are marked as deleted. They show up in search results only if the checkbox to Show historical entities is checked. Deleted entities appear in lineage graph dimmed-out.

Historical entities are never automatically removed or archived from Atlas' metadata. If you find you need to remove specific deleted entities, you can purge specific entities by their GUIDs through REST API calls.

Temporary data assets

Sometimes operations include data assets that are created and then deleted as part of the operation (or as part of a series of operations that occur close together in time). Atlas collects metadata for these temporary objects. The technical metadata for the operation, such as query text, includes a reference to the temporary object; the object itself will show in the Atlas lineage diagrams.

For example, consider a Hive pipeline that writes data to a table, transforms the data and writes it to a second table, then removes the first table. The lineage graph shows the source file, the process that creates the first table, the first table, the process that transforms the data and loads it into the second table, and the second table. Atlas also collects the process where the first table is dropped. When you view the lineage graph, you can choose to show the first table or to exclude it by setting the filter option Hide Deleted Entity.

Support for On-Demand lineage

The on-demand lineage provides improved end user experience to handle data flow and related entities.

Currently, the lineage works with minimal basic data to return all the connected entities for any single request. The Atlas UI consumes additional time to load when the lineage response is larger than usual and also when the lineage is multi-level, the cost of processing the data could be higher.

To handle these conditions, the concept of on-demand lineage is introduced. Using the on-demand lineage, you can provide more control to the user for depth and height of specific lineage, apart from increasing the number of nodes from three (which is default with three input and output nodes respectively) and finally displaying the maximum number of nodes. On-demand lineage also processes limited lineage data that helps in the optimization of the UI.

In hindsight, the lineage data is broken down into parts and when the user clicks on the specific “expandable” button in the lineage, only then the lineage shows up fully (or as the lineage data is broken) to provide additional information.

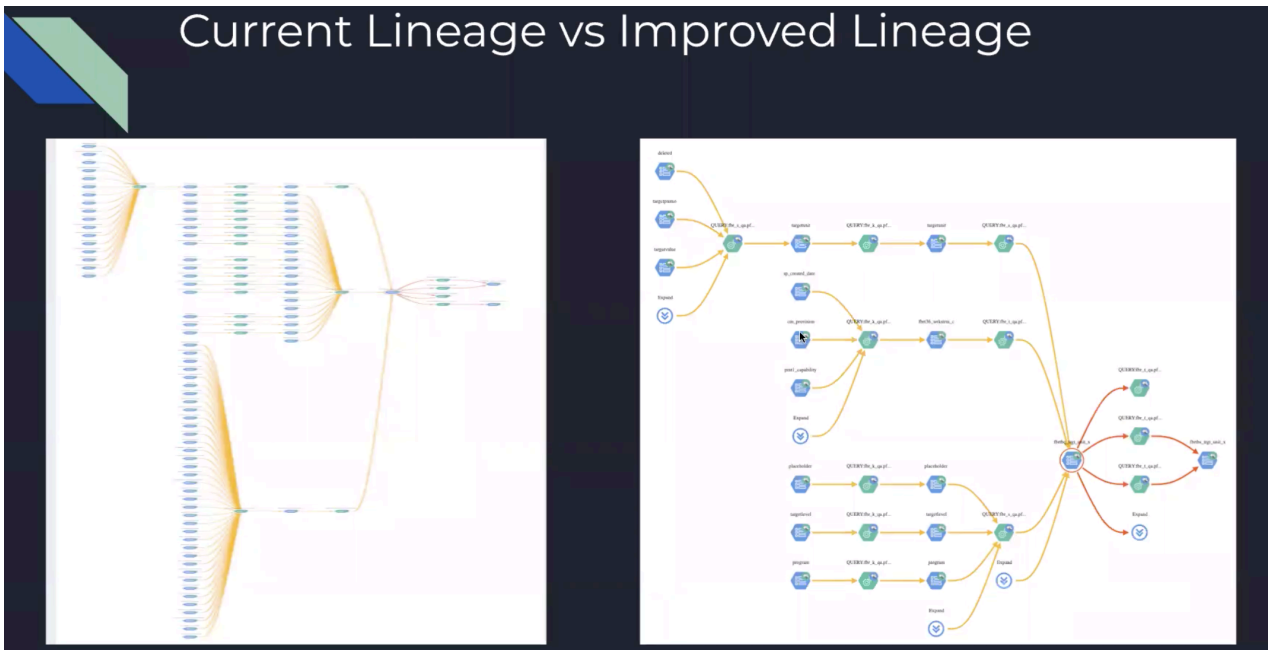

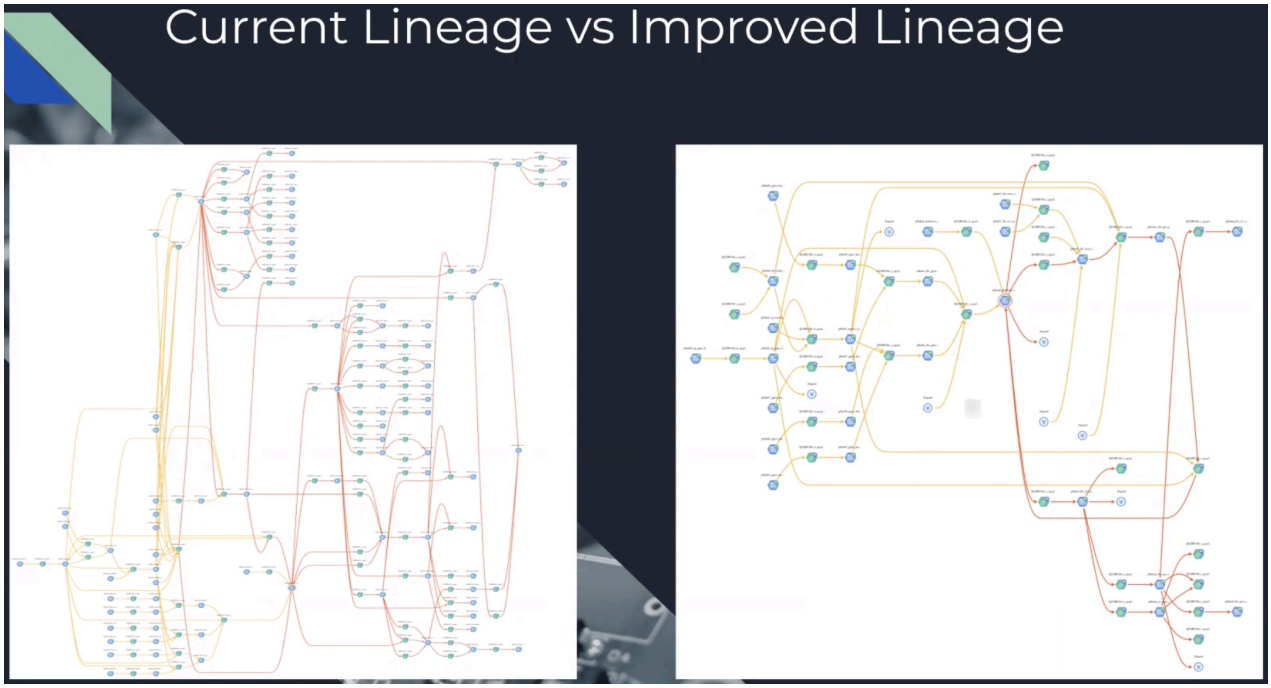

Attention: The existing lineage setup is also supported but the user has to configure additional settings to bring up the on-demand lineage feature.

You can set the following parameters to use the on-demand lineage:

- atlas.lineage.on.demand.enabled = true
- atlas.lineage.on.demand.default.node.count = 3 (Default value is 3)
- atlas.lineage.max.node.count = 5k (Atlas supports upto 9k nodes)

Note the following:

- If you pass an inputRelationsLimit/ outputRelationsLimit in the payload, the entered values are applicable only for that specific GUID, and for other nodes Atlas considers the default value as input/output Relations Limit.
- If you do not provide any limits or if you have 0 as the limit, Atlas considers default configurations.

- Order nodes in lineage are based on how the nodes are stored in JanusGraph.

The lineage data is retrieved from JanusGraph (retrieving the edges and vertices). The order in which the data availability is provided solely depends on the order JanusGraph stores and returns vertices. And the order of vertices in lineage is generally consistent all the time. In other words, every time a request is initiated for lineage data, the result is in the same order and the lineage on-demand feature is designed based on the consistent ordering as available in the JanusGraph database.

- You can update the node count value using the Settings > Nodes On Demand option in Atlas UI. This, apart from updating the same using Cloudera Manager.

HDFS lineage data extraction in Atlas

Atlas supports the HDFS lineage data extraction mechanism. Learn how it works, which HDFS shell script commands are supported, how to use these commands, and what the supported HDFS entities and hierarchies are.

Previously, HDFS data extraction ensured that only partial data was captured. Data stewards and platform owners track lineage of their ingest and ETL processes that depend on custom programs that convert copy and convert data when moving from one directory in HDFS to another.

The data descends into a staging directory in HDFS and later, periodically moves to an input directory in HDFS. Hive External tables are constructed on top of the input directory and Atlas does not support capturing HDFS locations without the Hive External table being created on top of the HDFS location(s). Typically, the move from staging to input directory is captured in Atlas lineage. The connectivity between staging and input directories does not generate the lineage automatically.

With the enhancement, lineage would automatically be captured and connected for the data ingest processes. This development allows the data architect and data steward to understand and govern data from ingest, ETL to consumption.

How HDFS lineage extraction works

The enhanced implementation consists of a wrapper, which is based around the HDFS commands that creates a new lineage for filesystem operations.

The lineage captures the directory / file specified in the scripting tool. The lineage also creates a parent-child relationship of directories / files and its contents upto the NameNode.

A filtering mechanism is also built to exclude or include paths and directories.

Currently, the following HDFS shell script commands are supported:

- -put and -copyFromLocal - Copy the contents from the local machine into HDFS.
- -cp - Copies the contents from one path to another HDFS path.
- -mv - Moves the contents from one path to another HDFS path.
- -rm - Removes the contents from HDFS.

Related Information

HDFS lineage commands

Prerequisites for HDFS lineage extraction

Before you start the extraction mechanism ensure you review the activities that need to be performed in Cloudera Manager.

You must enable the gateway role in Cloudera Manager. The following properties are visible in the Apache Atlas configuration.

- atlas.hdfs.lineage.blacklist.paths

- atlas.hdfs.lineage.whitelist.paths

For example:

atlas.hdfs.lineage.blacklist.paths=/tmp/blacklist/./tmp/whitelist/blacklist/

atlas.hdfs.lineage.whitelist.paths=/tmp/whitelist/

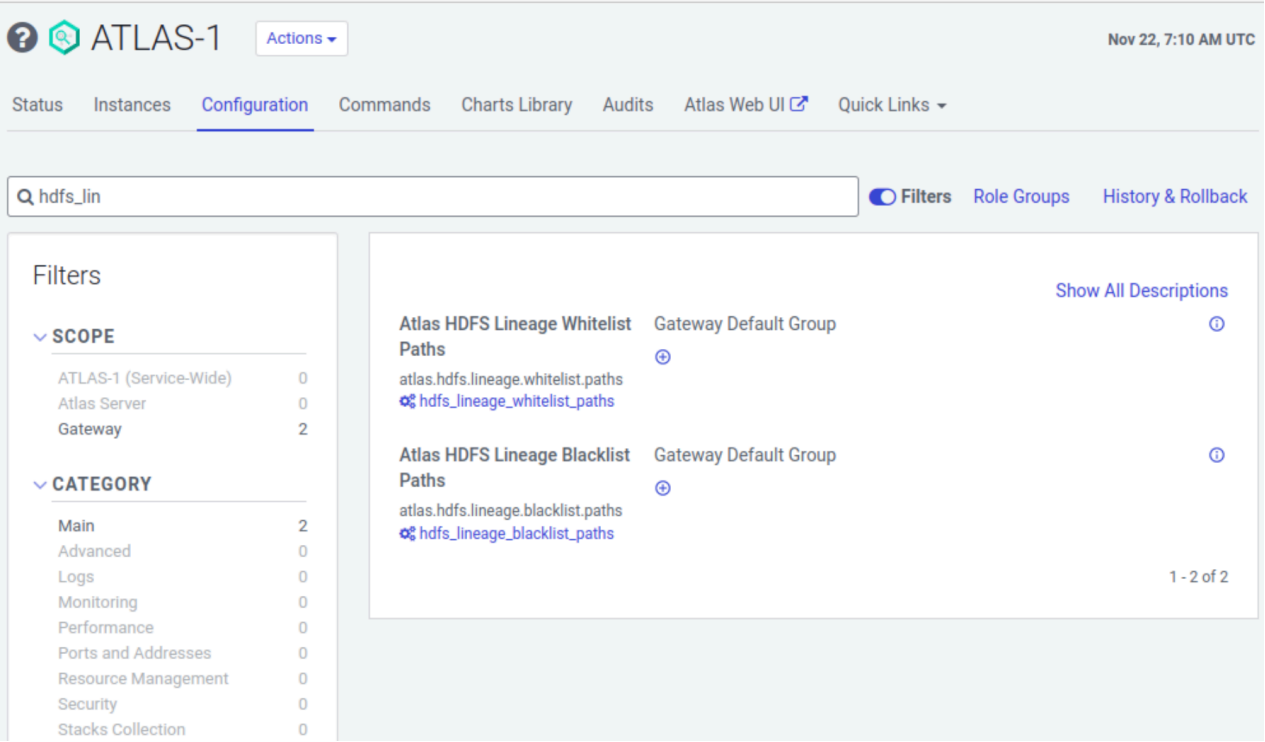

Adding additional configuration

Before you commence the HDFS extraction, you must include the following properties in your Atlas setup to ensure that the HDFS entities reflect at the Atlas endpoint.

- Set the following to atlas-application.properties manually (NOT in Cloudera Manager) using the path: /etc/atlas/conf/atlas-application.properties

  Note: The user has to update the Atlas properties (/etc/atlas/conf/atlas-application.properties) in the Gateway node and run the extractor script from there. Any user who is allowed or permitted or has the required access to publish messages to Kafka can be used. For example: "atlas"

  - atlas.jaas.KafkaClient.option.keyTab
  - atlas.jaas.KafkaClient.option.principal

If you do not add these properties, the HDFS lineage entities do not reflect at the Atlas endpoint.

Attention: Entities created through Hook do not get consumed by Atlas and are specifically observed while running the HDFS Lineage script. Once the process to run the hdfs-lineage.sh script is completed, it is seen that in a few instances the entity is not created in Atlas. This scenario is observed on an intermittent basis and few entities are not viewed in Atlas. In the case where this issue is observed, the publishing of messages to Kafka topics consumes more than three seconds. This additional time consumption could be because of:

- Logging into Kerberos took more time to complete
- Connecting to Kafka topic took more than three seconds

To overcome this scenario, the Async message processing (atlas.notification.hook.asynchronous) must be disabled. You must manually set this flag in /etc/atlas/conf/atlas-application.properties to false

HDFS lineage commands

You must run the following commands to complete the HDFS lineage extraction into Atlas.

The name of the HDFS lineage capture script is called hdfs-lineage.sh. The extractor script is located in the extractors/bin directory in $ATLAS_HOME. The default location path is: /opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/hdfs-lineage.sh

Attention: Any HDFS lineage commands that are run outside of the script does not capture the lineage in Atlas.

As noted previously in earlier about the way HDFS lineage extraction is performed, the HDFS shell script commands are used to extract lineage in Atlas. The commands are run in the order of extraction.

Related Information

Prerequisites for HDFS lineage extraction

Running HDFS lineage commands

To support HDFS lineage extraction, you must execute the lineage shell script commands.

About this task

Running the HDFS commands:

Running the -put command:

./hdfs-lineage.sh -put ~/csvfolder/ /tmp/whitelist/demo/cpfol/cp1

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

```
Hdfs dfs -put /root/csvfolder/ /tmp/whitelist/demo/cpfol/cp1 command was
          successful
```

HDFS Meta Data extracted successfully!!!

You must verify if the specific folder is moved to Atlas.

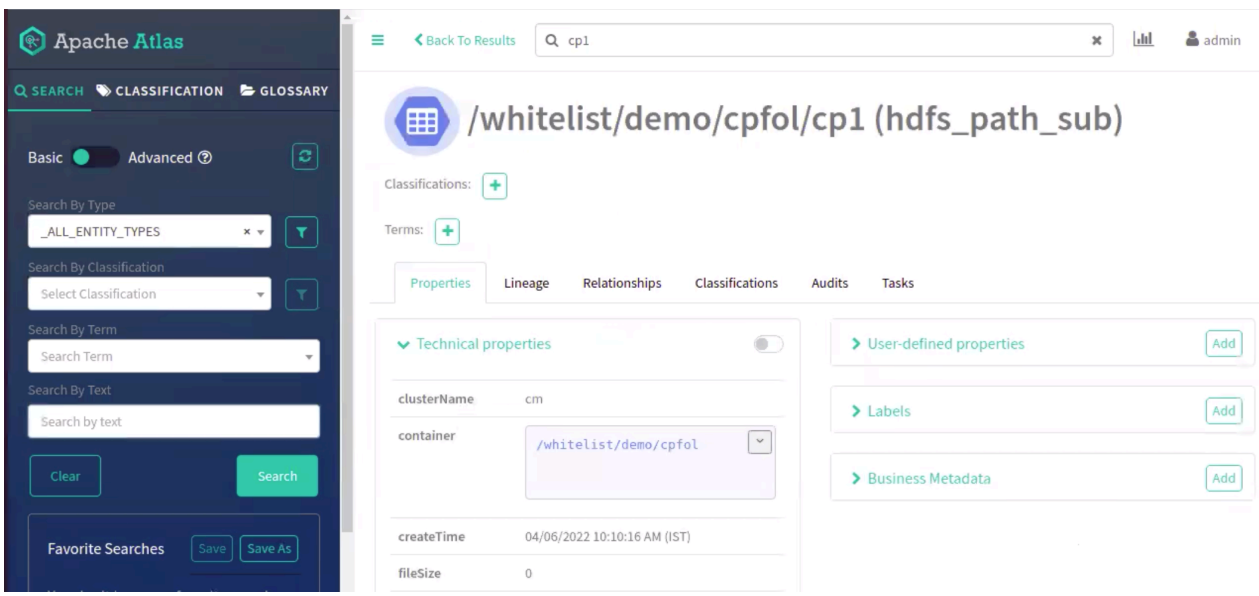

Running the -cp command:

./hdfs-lineage.sh -cp /tmp/whitelist/demo/cpfol/cp1 /tmp/whitelist/demo/cpfol/cp2

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

Hdfs dfs -cp /tmp/whitelist/demo/cpfol/cp1 /tmp/whitelist/demo/cpfol/cp2 command was successful

HDFS Meta Data extracted successfully!!!

You must verify if the contents of the folder is moved to another folder.

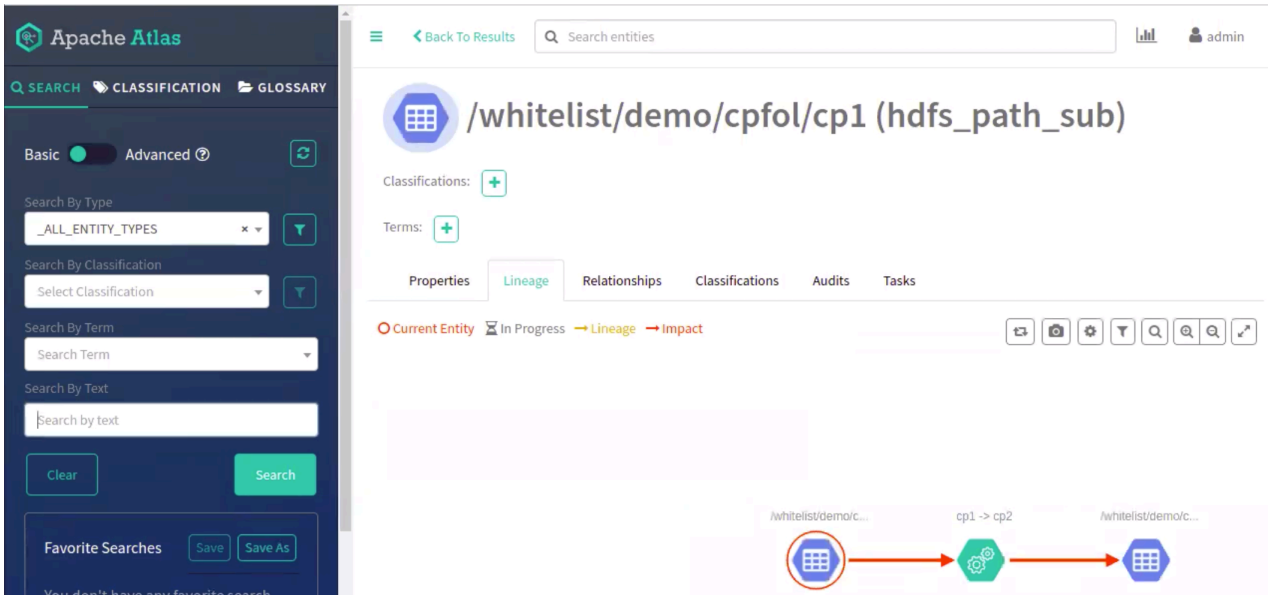

Similarly, you can move the contents of the copied folder to another folder.

./hdfs-lineage.sh -cp /tmp/whitelist/demo/cpfol/cp2 /tmp/whitelist/demo/cpfol/cp3

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

Hdfs dfs -cp /tmp/whitelist/demo/cpfol/cp2 /tmp/whitelist/demo/cpfol/cp3 command was successful

HDFS Meta Data extracted successfully!!!

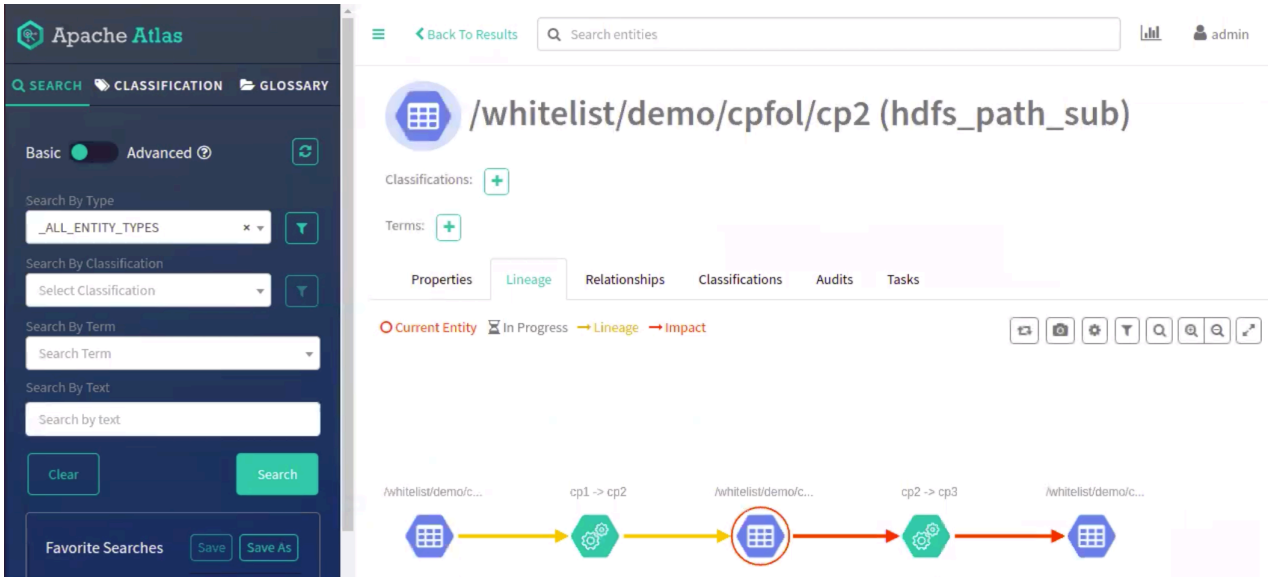

You must create an External Hive table on top of the HDFS location. For example, a Hive table on top of the HDFS location: cp3

create external table if not exists countries_table_11 (

CountryID int, CountryName string, Capital string, Population string)

comment 'List of Countries'

row format delimited

fields terminated by ','

stored as textfile

location '/tmp/whitelist/demo/cpfol/cp3';

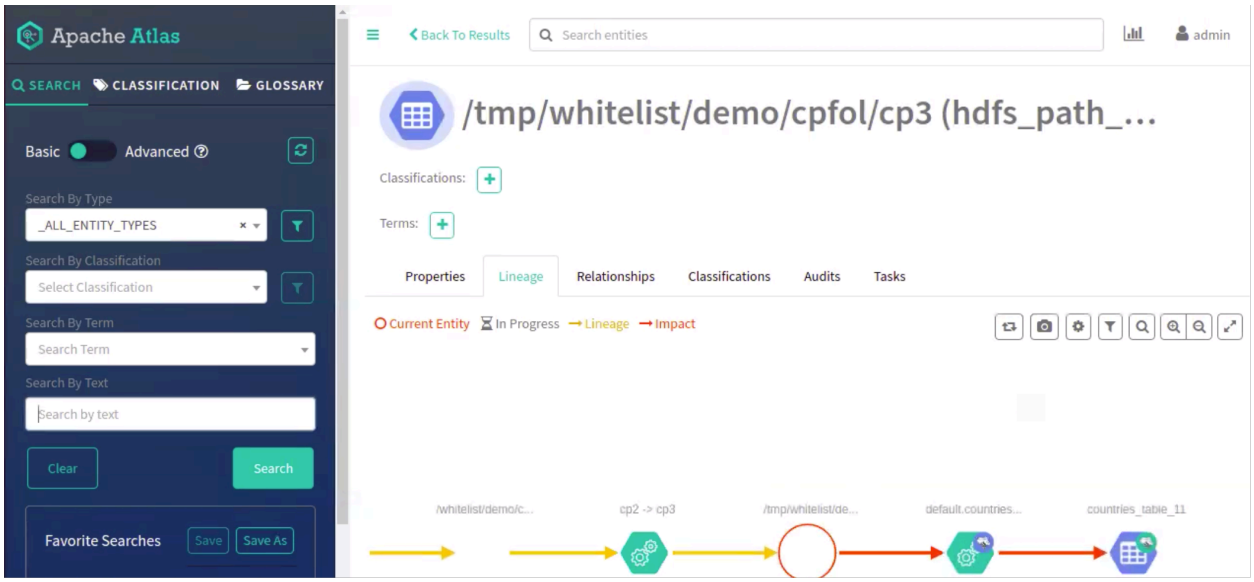

Running the mv command:

./hdfs-lineage.sh -put ~/csvfolder/ /tmp/whitelist/demo/mvfol/mv1

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

```
Hdfs dfs -put /root/csvfolder/ /tmp/whitelist/demo/mvfol/mv1 command was
          successful
```

HDFS Meta Data extracted successfully!!!

A single node is generated in the Atlas UI

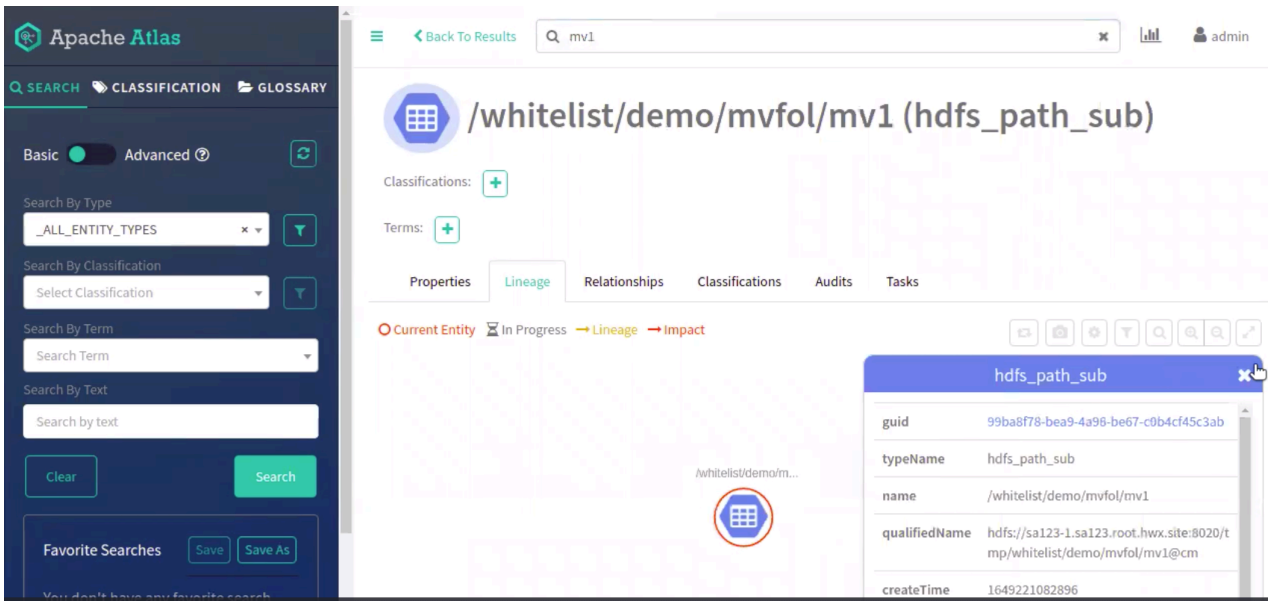

You can move the contents of the folder mv1 to mv2

./hdfs-lineage.sh -mv /tmp/whitelist/demo/mvfol/mv1 /tmp/whitelist/demo/mvfol/mv2

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

Hdfs dfs -mv /tmp/whitelist/demo/mvfol/mv1 /tmp/whitelist/demo/mvfol/mv2 command was successful

HDFS Meta Data extracted successfully!!!

In Atlas UI, you can view the contents of mv1 being moved to mv2 and the old entity is deleted (grayed out in the image) and likewise it continues for every operation where the previously created entity is deleted.

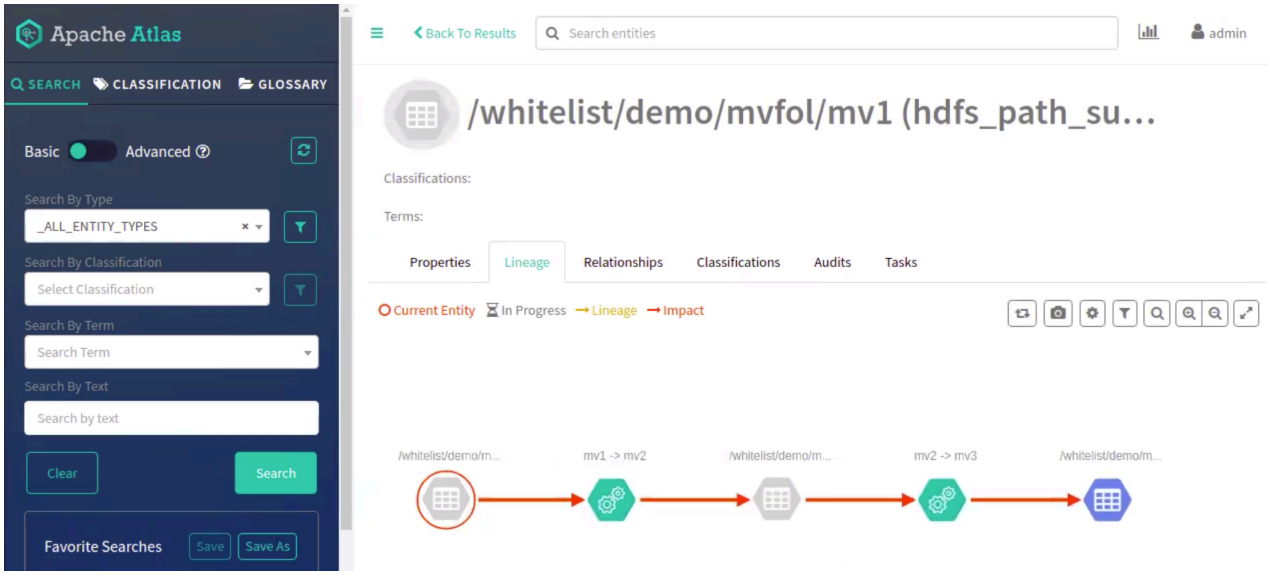

You must create an External Hive table on top of the HDFS location. For example, a Hive table on top of the HDFS location: mv3

create external table if not exists countries_table_12 (

CountryID int, CountryName string, Capital string, Population string)

comment 'List of Countries'

row format delimited

fields terminated by ','

stored as textfile

location '/tmp/whitelist/demo/mvfol/mv3';

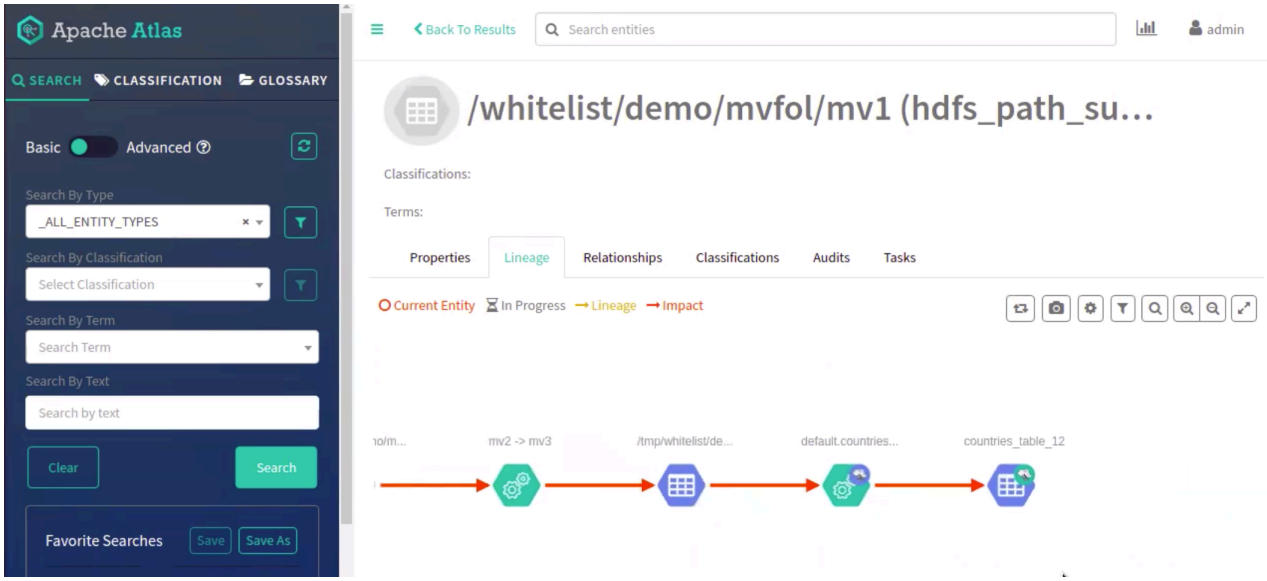

Running the rm command:

Create a sample file in HDFS

./hdfs-lineage.sh -put ~/sample.txt /tmp/whitelist/rmtest.txt

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

Hdfs dfs -put /root/sample.txt /tmp/whitelist/rmtest.txt command was successful

HDFS Meta Data extracted successfully!!!

Use the rm command to delete the rmtest.txt file.

./hdfs-lineage.sh -rm -skipTrash /tmp/whitelist/rmtest.txt

Log file for HDFS Extraction is /var/log/atlas/hdfs-extractor/log

Hdfs dfs -rm -skipTrash /tmp/whitelist/rmtest.txt command was successful

HDFS Meta Data extracted successfully!!!

Note: If a folder contains multiple sub-folders and if the parent folder is deleted, all the contents of the parent folder are deleted.

Example

./hdfs-lineage.sh -rm -r -skipTrash /tmp/whitelist/demo/delfol

Related Information

Supported HDFS entities and their hierarchies

Inclusion and exclusion operation for HDFS files

The content or the files available in HDFS are either captured or ignored as lineage in Atlas. This scenario is achieved by including and excluding files and folders that Atlas checks for creating a lineage.

The hdfs.conf file contains the configuration for inclusion and exclusion options as described in Prerequisites for HDFS lineage extraction on page 9.

Some use cases where the including and excluding of HDFS files can be achieved are as follows

- When a simple file is created in the “whitelist” folder, a lineage is created in Atlas.

./hdfs-lineage.sh -put ~/sample.txt /tmp/whitelist/sample2.txt

- Similarly, when you try to create a file in the “blacklist” folder, Atlas does not create a lineage.

./hdfs-lineage.sh -put ~/sample.txt /tmp/blacklist/sample3.txt

- If a folder is not listed for any specific operation (Whitelist or Blacklist), Atlas does not capture the lineage.

./hdfs-lineage.sh -put ~/sample.txt /tmp/non/folder1/folder1file.txt

Note: Atlas does not capture the lineage even if the parent folder is included but the sub-folder / directories are excluded.

For example:

./hdfs-lineage.sh -put ~/sample.txt /tmp/whitelist/blacklist/blacklistfile.txt

- Atlas captures the lineage of the sub-folder if the parent directory or folder is included.

./hdfs-lineage.sh -put ~/sample.txt /tmp/whitelist/whitesub/whitelistsubfile.txt

- Atlas does not create a lineage to an excluded folder but a lineage is created to the same excluded folder if there is an included path available.

./hdfs-lineage.sh -put ~/sample.txt /tmp/blacklist/blToWl.txt

./hdfs-lineage.sh -cp /tmp/blacklist/blToWl.txt /tmp/whitelist/blToWl.txt

Atlas captures the lineage when folder(s) are included and later the lineage is captured from the excluded entity.

Important: The included folder takes precedence over the excluded folder, always.

The included folder takes precedence over the excluded folder, always.

./hdfs-lineage.sh -put ~/sample.txt /tmp/whitelist/wlToBl.txt

./hdfs-lineage.sh -cp /tmp/whitelist/wlToBl.txt /tmp/blacklist/wlToBl.txt

Supported HDFS entities and their hierarchies

When you perform basic search with the sub-type "hdfs_path" through Atlas UI or API, the search results include other HDFS types like "hdfs_path_v2", "hdfs_path_namenode", and other related entities.

The following HDFS lineage types are available when you perform the search using hdfs_path:

- hdfs_path_v2

  - hdfs_container

  - hdfs_contained

  - hdfs_path_namenode

  - hdfs_process
- hdfs_path_v2 is subtype of - hdfs_path - hdfs_container - hdfs_contained
- hdfs_path_namenode is subtype of - hdfs_path - hdfs_container

Note: Searching for the entity hdfs_path renders results for hdfs_path_v2 and hdfs_path_namenode. When the 'Exclude subtypes' option is selected, only the hdfs_path results are available.

Cloudera Runtime 7.3.1

atlas-extract-adls

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Before you start 4

Introduction to Atlas ADLS Extractor 4

Terminologies 4

Extraction Prerequisites 4

Updating Extractor Configuration with ADLS Authentication 5

Configuring ADLS Gen2 Storage Queue 5

Setting up Azure managed Identity for Extraction 12

Creating Managed Identity 12
Assigning Roles for the Managed Identities 12
Mapping Atlas Identity to Cloudera users 13

Running ADLS Metadata Extractor 13

Running Bulk Extraction 14

Running Incremental Extraction 14

Command-line options to run Extraction 15

Extraction Configuration 15

Verifying Atlas for the extracted data 17

Resources for on-boarding Azure for Cloudera users 17

Before you start

It is assumed that you are familiar with Cloudera, Azure Environments, and Azure Data Lake Storage Service.

Refer to Resources for Onboarding Azure for Cloudera to familiarize yourself with the terms used for this feature.

It is assumed that you have performed the steps mentioned in Azure quick start and are familiar with the functioning of Atlas.

Follow the Azure environment documentation to register your Azure environment to Cloudera and create required resources.

Introduction to Atlas ADLS Extractor

The Atlas metadata extractor, adls-extractor, for Azure Data Lake Storage is a stand-alone application that you can run on the Atlas host as needed to provide comprehensive metadata for data assets stored in ADLS Gen2.

The adls-extractor reads metadata relevant for Atlas from ADLS artifacts like blobs, directories, containers, storage accounts and publishes them to Atlas.

In order for extraction to function, the following prerequisites must be met:

- Determine the Type of Extraction on page 5
- Updating Extractor Configuration with ADLS Authentication on page 5
- Setting up Azure managed Identity for Extraction on page 12

To perform the extraction follow Running ADLS Metadata Extractor on page 13.

Once the extraction process is completed, you must verify the data that has been published to Atlas. For more information, see Verifying Atlas for the extracted data on page 17.

Terminologies

When you use this document for performing ADLS metadata extraction, there are terminologies and usage of technical terms that you could come across. Use the following information for more information.

ADLS

Azure Data Lake Storage. For more information, see Azure Data Lake Storage.

ADLS

Storage and Blob: Definitions are available in Introduction to Azure Blob Storage.

Atlas

Metadata management Service on Cloudera. For more information, see Using metadata for cluster governance.

IDBroker

Authentication services within Cloudera Public Cloud. For more information, see Cloud identity federation

Extraction Prerequisites

Before you perform Atlas metadata extraction, you must understand the prerequisites of this feature, that includes the types of extraction.

Determine the Type of Extraction

Choose Running Bulk Extraction on page 14 if you need to get the complete snapshot of the storage account, container or specific path (configured at allow list in extractor configuration file) metadata at Atlas. Normally recommended for the very first time.

Choose Running Incremental Extraction on page 14 if you need to keep Atlas synchronized with the changes happening in ADLS. This takes more time to process the individual changes compared to bulk extraction. It needs to be run periodically if lineage has to be kept up to date.

If incremental extractions need to be set up to run periodically, it can be done using your own scheduling tools or using crontabs. As you get familiar with the nature of changes within ADLS, viz. volume of changes and the speed of the extraction, you can determine the frequency with which to run the extractor.

For incremental extractions to function, you will need to set up Azure storage queues. Please see Configuring ADLS Gen2 Storage Queue on page 5 for more details.

Note: The success of this operation is dependent on ensuring that the extraction is done before the events are deleted from the ADLS queue.

Updating Extractor Configuration with ADLS Authentication

In the Cloudera environment Knox IDBroker is used for federated authentication.

The extractor employs the Atlas' kerberized session to authenticate to Knox IDBroker. It receives the delegation token from IDBroker and uses it to request cloud credentials. No configuration change is necessary in this case.

To override this behaviour, the ADLS secret keys can be configured at:

/opt/cloudera/parcels/CDH/lib/atlas/extractors/conf/adls.conf Default

Example: atlas.adls.extraction.account.key=<key>

Configuring ADLS Gen2 Storage Queue

You must create and configure a storage queue for the ADLS Gen2 storage account which was created as part of Extraction Prerequisites on page 4 for storing blob and directory event notifications.

Note: The ADLS storage queue is created only for Incremental Extraction.

Create or Identify a Storage queue

1. On Azure Portal > Storage Accounts > Queue service > Queues > Queue.
2. Provide a name for your queue and click OK. For example, testqueue.

Configuring the Storage queue for Storage events

Configure the above storage queue for the ADLS Gen2 storage account so that it gets all the Blob and Directory events like create, delete, and rename operations.

1. On Azure Portal, navigate to Storage Accounts > Events > Get Started > When a new blob is uploaded > Create. The Logic App Designer window is displayed.

2. Click Create Azure Blob Storage and Azure Event Grid > Continue.

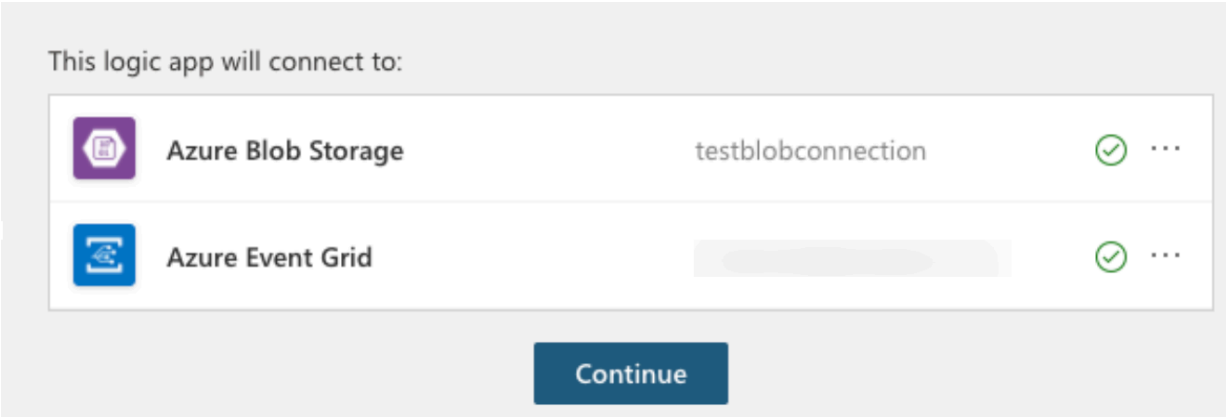

3. Choose all the events for When a resource event occurs:
   - Microsoft.Storage.BlobCreated
   - Microsoft.Storage.BlobDeleted
   - Microsoft.Storage.BlobRenamed
   - Microsoft.Storage.DirectoryCreated
   - Microsoft.Storage.DirectoryDeleted
   - Microsoft.Storage.DirectoryRenamed

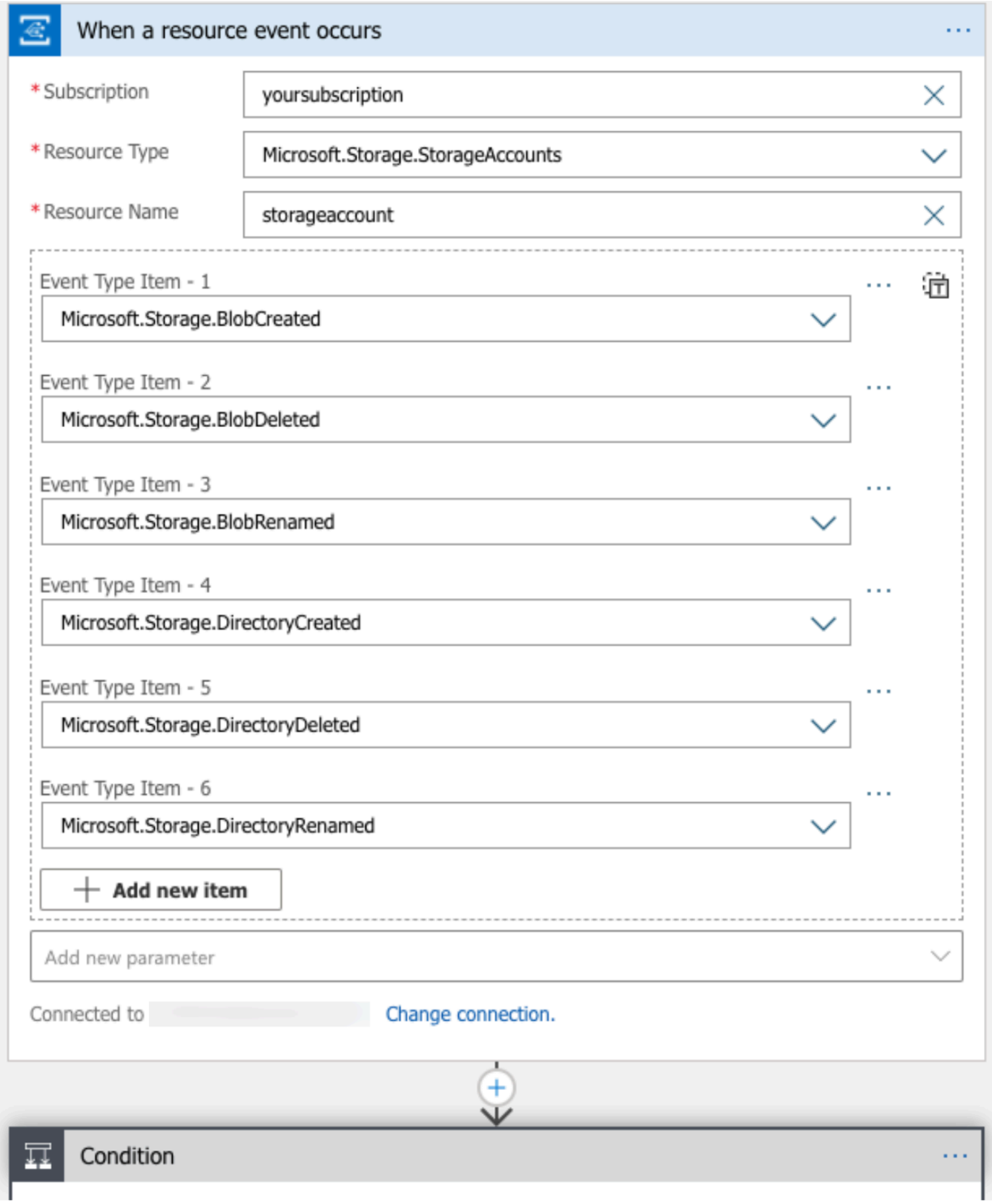

4. Delete the Condition block and click New step.

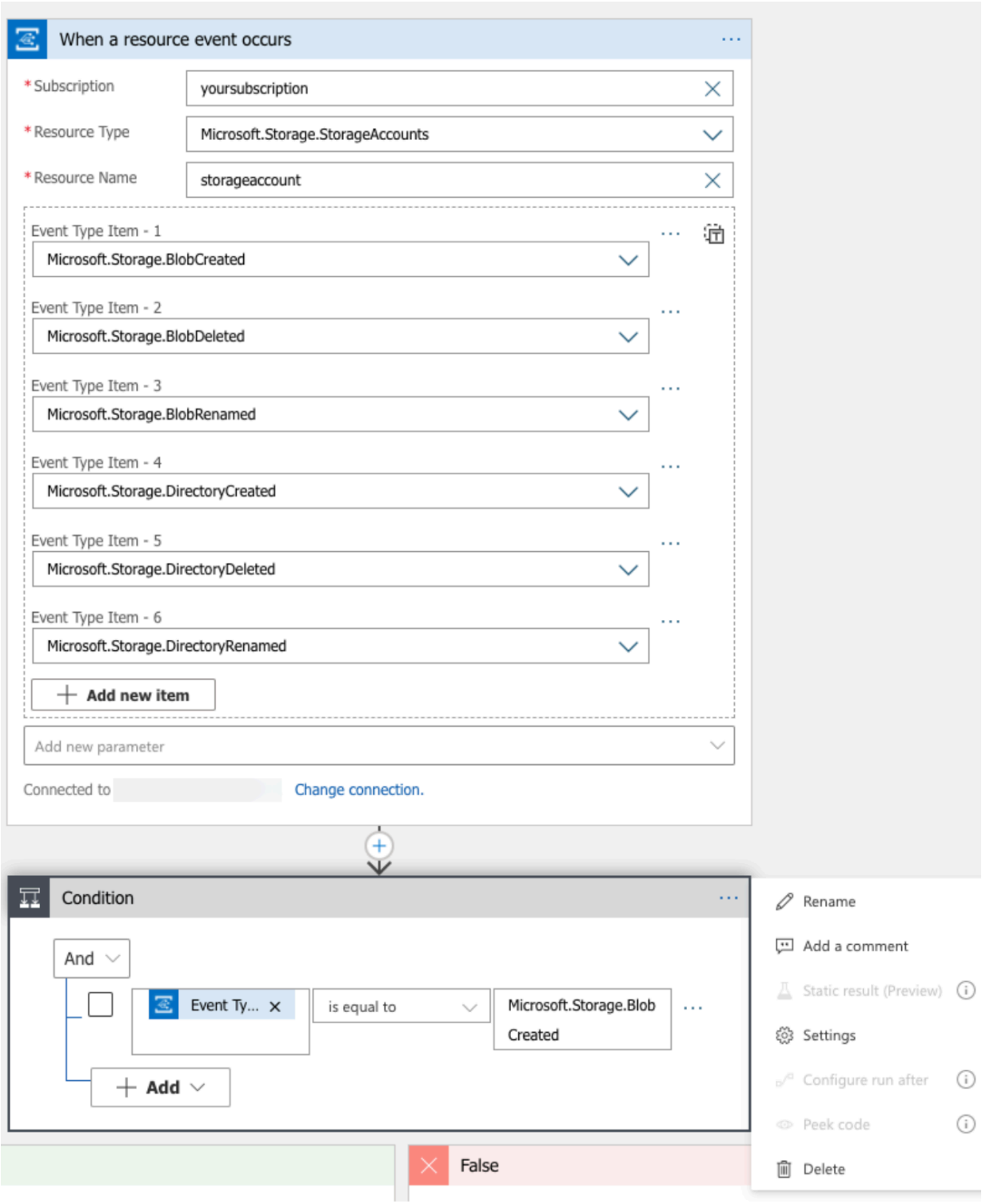

5. Under Choose an operation window search Put a message on a queue and choose the Action > Azure Queues > Put a message on a queue.

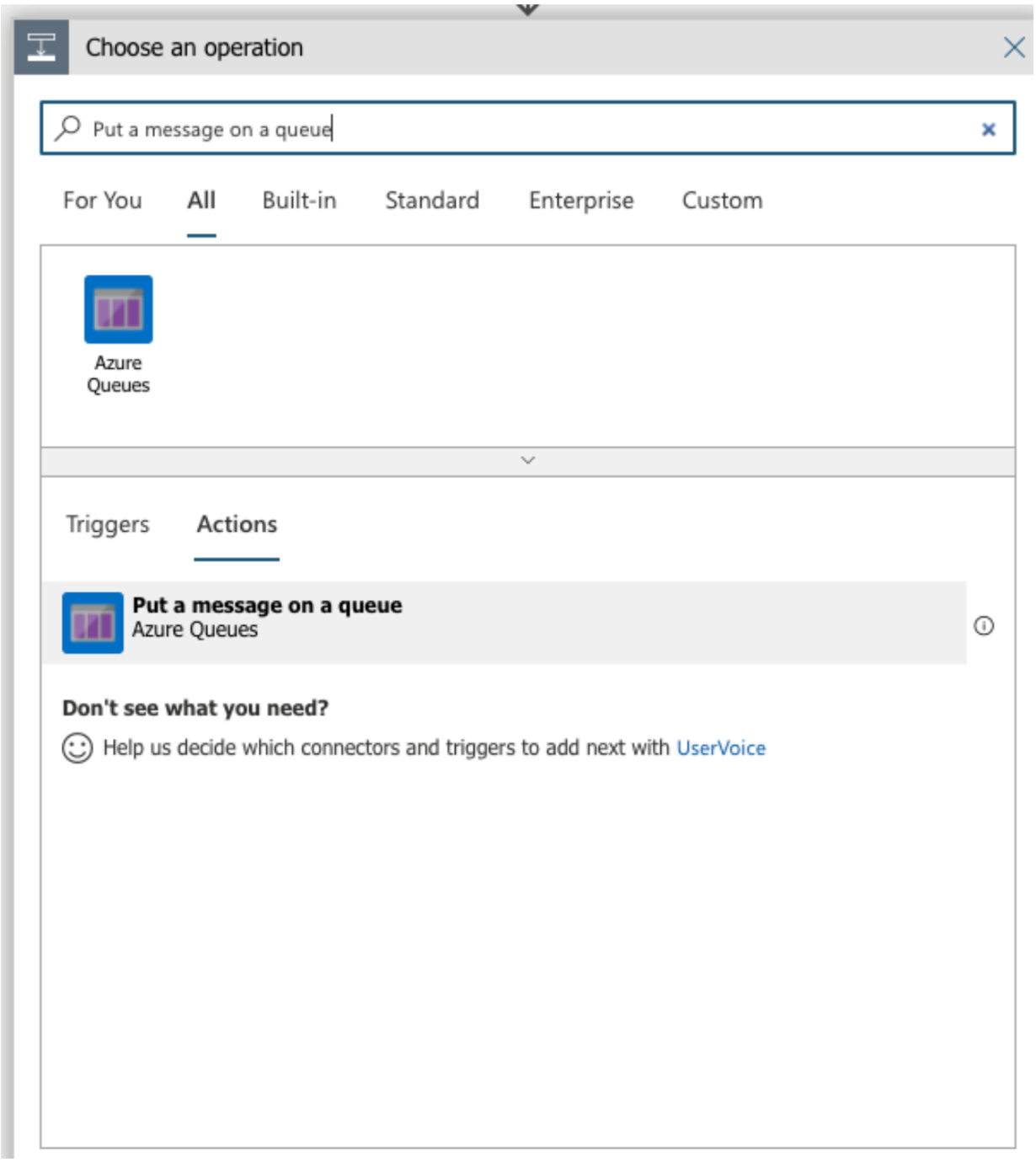

6. Choose the queue name and Messages format:
   - Queue Name: testqueue. (Example)
   - Message format: Body.

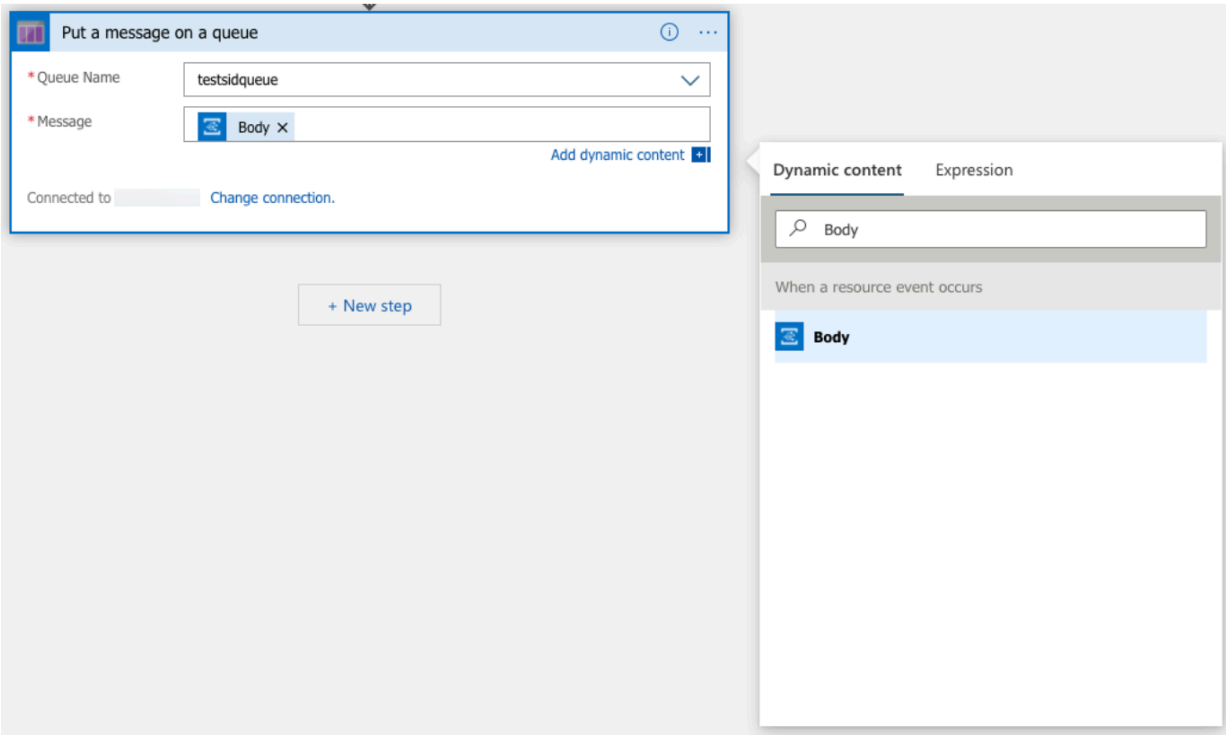

7. Click Save As and provide an appropriate name to Logic App. For example: newLogicAppForEventQueuetest.

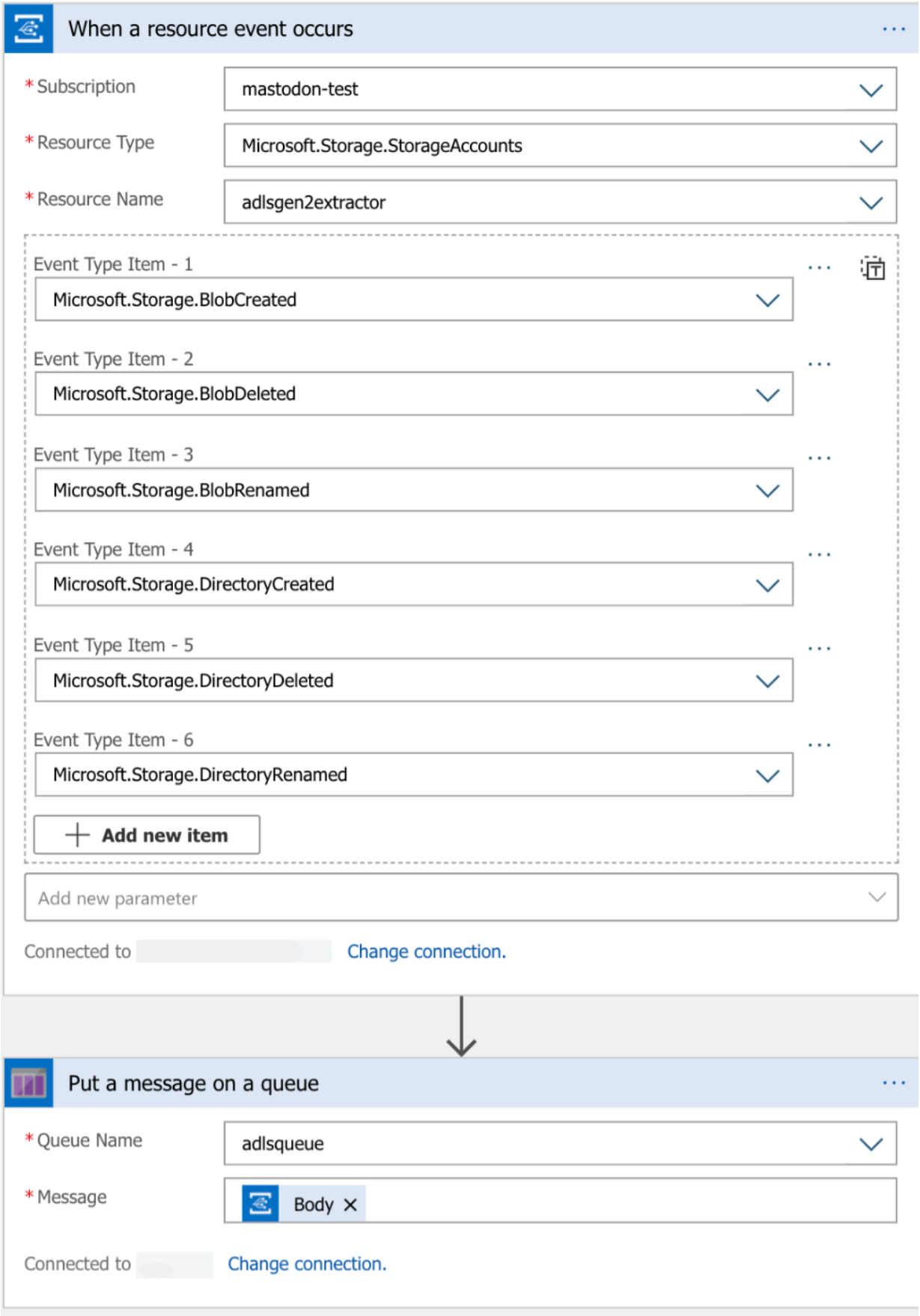

8. Click Create.

Setting up Azure managed Identity for Extraction

Once you have created and configured the storage account and setup the queue (for incremental extraction only), you must create the managed identities and later assign roles with specific scopes to these identities.

You can reuse the same resource group that you created for the storage account or you can create a new resource group that can act as a logical grouping of managed identities.

Note: It is assumed that the minimal secure setup, where the managed identity will have at least one role assigned.

Creating Managed Identity

You must create a new managed identity or use the existing one based on the requirement.

1. On the Azure portal > Azure services homepage select Managed Identities > Create.

   The Create User Assigned Managed Identity page is displayed.
2. Under the Basics tab, provide information for the following:
   - Subscription
   - Resource Group (You can optionally create a new Resource Group as well)
     - Region
     - Name (Specify the name of a managed identity). For example, testatlasmanagedidentity
3. Click Review + Create.
4. Review the entered information and click Create.

   The deployment is initialized and submitted. Later, it is successfully deployed to the selected Resource Group.

Assigning Roles for the Managed Identities

Once you have created the Managed Identities, assign the roles with specific scoopes to the identity.

Procedure

1. In ADLS > Storage Accounts > Your storage account Access Control (IAM) > Add role assignments.
2. Based on the type of extraction selected, you can add the type of role:

   For Bulk extraction, you can add the following:
   - Click Add > Add role assignment
   -
     - Role > Select Storage Blob Data Contributor.
     - Assign Access to > Select User assigned managed identity.

  - Subscription > As applicable.
- Click Save.

For Incremental extraction, you can add the following:

- Click Add > Add role assignment
  - Role > Storage Queue Data Contributor.
  - Assign Access to > User assigned managed identity.
  - Subscription > As applicable.
- Click Save.

Mapping Atlas Identity to Cloudera users

In order to use the created Managed Identity, a Apache Atlas user must be mapped to the managed identity.

The option to add or modify these mappings is available in the Cloudera Manager UI.

1. Select Management Console under Environments > click on an environment > Actions > Manage Access > IDBroker Mappings > Edit.
2. Under IDBroker Mappings, you can change the mappings of the Atlas user to managed identity. The user or group dropdown is prepopulated with Cloudera users and groups. On the right hand side, specify the Resource ID (copied from the Azure Portal) for the Atlas user.

For example:

1. Cloudera Manager > Configuration > Search for the user role and add Knox IDBroker Azure User Mapping information.
2. Restart the Knox service.

Note: If a user is mapped to multiple roles via group membership, the specific role to be used needs to be provided at runtime. If the user is mapped directly to a role, the direct mapping takes precedence over mapping via group membership. For information on how to specify the role, refer to Specifying a group when user belongs to multiple groups .

Running ADLS Metadata Extractor

You must note about how you can run the ADLS Metadata Extractor.

The extractor script is located at:

/opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/adls-extractor.sh

The configuration is located at:

/opt/cloudera/parcels/CDH/lib/atlas/extractors/conf/adls.conf

Note: The extractor configuration file includes properties that require ADLS Gen2 configuration values. Be sure to set those values in the configuration file before running the command.

Before running the extraction script, you must log in to Atlas using “atlas” user or a “root” user.

Attention: Ensure that a valid Atlas Kerberos session is in place before running the script.

For more information, see Extraction Configuration on page 15.

Running Bulk Extraction

Bulk extraction mechanism employs direct calls on Azure API calls to fetch all blobs and containers in ADLS Gen2 storage account.

If there is a failure while extracting the complete Azure metadata, the bulk extraction must be resumed from the last checkpoint by changing atlas.adls.extraction.resume.from.progress.file=true configuration at adls.conf.

The following command line example runs the bulk extraction. Assuming the mandatory properties are set in the default configuration file, only the parameter to enable bulk mode is required:

/opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/adls-extractor.sh

Or

/opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/adls-extractor.sh -e BULK

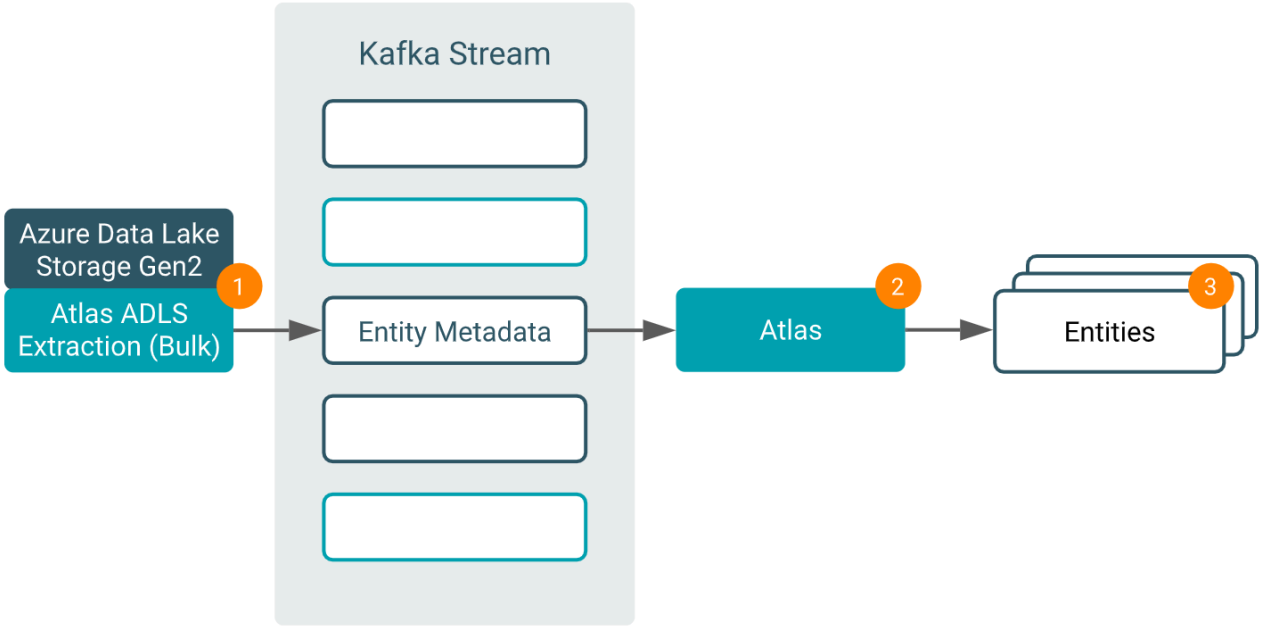

Refer to Extraction Configuration on page 15for more details on different optional configurations.

Running Incremental Extraction

With incremental extraction, you can keep Atlas updated with changes happening within ADLS.

You must configure the queue for the storage account so that extraction receives all blob, create, delete, and rename events.

In addition, you can optionally set the timeout or retry time so the extractor does not keep running even when it is not able to connect to the queue.

The following command line example runs the incremental extraction. Assuming the mandatory properties are set in the default configuration file, only the parameter to enable incremental mode is required:

```
/opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/adls-extractor.sh -e INCR
EMENTAL
```

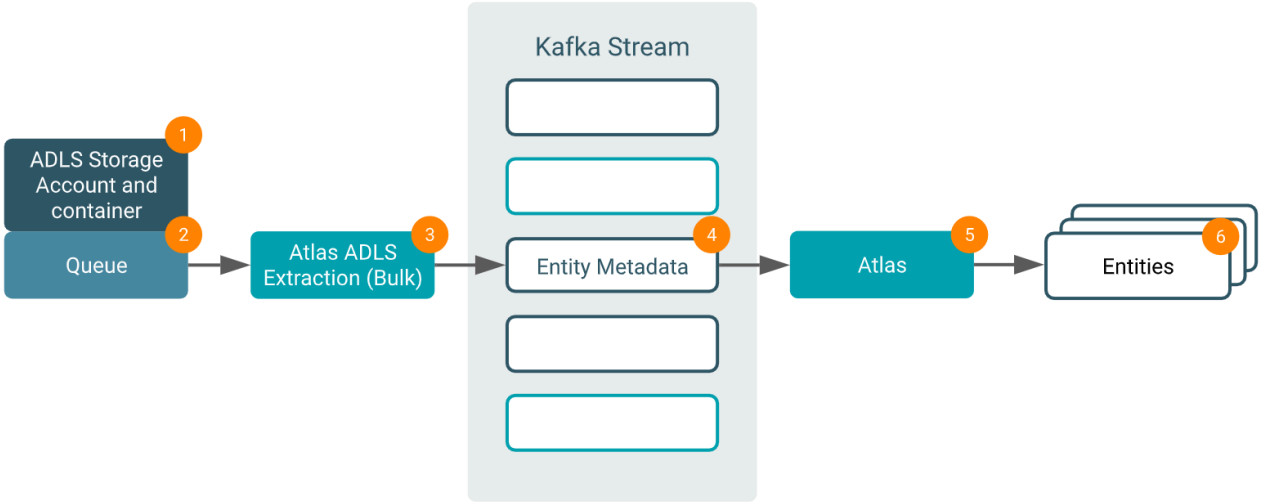

You must plan whether to set up a repeating job to run incremental extraction. Assuming you have a lot of changes that you want to track.

Command-line options to run Extraction

You can use the command-line options to run the Atlas ADLS metadata extraction.

Command-line option	Parameters	Default Value
-c, or--config	The path of the configuration file containing the extraction type specific configurations parameters.	By default it considers $ATLAS_HOME/extractors/conf/adls.conf. This can be changed using this parameter.
-e,--extraction <arg>	The type of extraction to be done. Two values allowed Incremental or Bulk.	Bulk
-f, -failOnError	Setting this option causes extraction to stop when an error occurs. Not setting this option, will cause the exception to be logged and operation to proceed.	false

Note: Progress of the extraction can be monitored via the log file /var/log/atlas/azure-adls-extractor.log

Extraction Configuration

Some of the configurations that must be set up before you perform the metadata extraction.

The configuration file is located at: /opt/cloudera/parcels/CDH/lib/atlas/extractor/adls.conf

Configuration Parameter	Purpoose	Default Value
atlas.adls.extraction.account.name	ADLS Gen2 storage account which was created as part of Extraction Prerequisites on page 4.	Mandatory.
atlas.adls.extraction.account.key	ADLS account key if IDBroker is not configured.	To be specified if Knox IDBroker is not configured at Cloudera.

Configuration Parameter	Purpoose	Default Value
atlas.adls.extraction.access.token	Access token for token based authentication.	If Knox IDBroker is not configured at , token based authentication is required. It must be configured.
atlas.adls.extraction.allowlist.paths=abfs://<containername>@<accountname>.dfs.core.windows.net/<path>	Comma separated ABFS paths or patterns from which ADLS metadata (directory, blob) needs to be extracted. Multiple values can be configured by ',' separated. Example: abfs://testcontainer@teststorageaccount.dfs.core.windows.net/testdir1/	
atlas.adls.extraction.denylist.paths=abfs://<containername>@<accountname>.dfs.core.windows.net/<path>	Comma separated ABFS paths or patterns from which ADLS metadata should be excluded from extraction. Multiple values can be configured by ',' separated. Example: abfs://testcontainer@teststorageaccount.dfs.core.windows.net/testdir2/	
atlas.adls.extraction.max.blob.per.call	Number of blob storage to be fetched in one call to Azure ADLS by bulk extraction.	1000
atlas.adls.extraction.timeout.per.call.in.sec	The timeout (seconds), used for each ADLS SDK call wherever it is required.	30
atlas.adls.extraction.resume.from.progress.file	Resume from the last run in case of failure feature.	Set to false by default. Set it to true if resuming an extract.
atlas.adls.extraction.progress.file	Progress file used for extraction in case the user wants to resume.	adls_extractor_progress_file.props
atlas.adls.extraction.max.reconnect.count	Specify the maximum number of retries to: • IDBroker in case of credentials expiry. • Retryable exception for Azure ADLS.	
atlas.adls.extraction.fs.system	File System used in Azure ADLS.	Default set to: abfs
atlas.adls.extraction.incremental.queueNames	Azure list of Account:QueueName which is configured as part of Configuring ADLS Gen2 Storage Queue on page 5 to get the blob/directory create, delete events. Example: teststorageaccount:testqueue	
atlas.adls.extraction.incremental.messagesPerRequest	The number of messages Incremental Extractor tries to fetch from ADLS Queue in a single call.	Default is 10. It ranges from 1 to 32.
atlas.adls.extraction.incremental.requestWaitTime	The wait time in seconds in a single call to ADLS Queue to fetch atlas.adls.extraction.incremental.messagesPerRequest messages.	20
atlas.adls.extraction.incremental.max.retry	Maximum retry count in case of Idle while reading Queue Messages in Incremental Extraction.	20
atlas.adls.extraction.incremental.delete.needed.for.rename	Does an entity need deletion if it has been renamed to something which should not be created at Atlas due to allow and deny list.	false
atlas.notification.hook.asynchronous	This setting should be set to "true" only when extracting a large number of adls metadata (directory, blob) where there is a possibility of a lag when publishing messages to ATLAS_HOOK Kafka topic.	Defaults to asynchronous sending of events: true To set synchronous sending of events: false (Synchronous)

Verifying Atlas for the extracted data

Once the metadata extraction is completed, you can verify the new types of data that are added to Atlas.

Data Type	Description
adls_gen2_account	Represents a StorageAccount (not the Azure Account).
adls_gen2_container	Represents the container
adls_gen2_directory	Represents the directory.
adls_gen2_blob	Represents a blob or file with the name.

Note: From Atlas' web UI, use Basic Search to search for entities added to Atlas as part of the extraction.

Resources for on-boarding Azure for Cloudera users

Refer to the following sections before you commence the Atlas ADLS Extraction feature.

- Introduction to Azure Data Lake Storage Gen2
- Azure quick start
- Azure Subscription Requirements
- Register an Azure Environment
- Creating ADLS Gen2 storage account and containers
- Setting up ADLS Gen2 and Managed Identities

Cloudera Runtime 7.3.1

Amazon S3 Extract

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Amazon S3 metadata collection **4**
Accessing AWS 6
AWS objects and inferred hierarchy 6
AWS object lifecycle 6

AWS configuration **6**
To configure an SQS queue suitable for Atlas extraction 7
To configure an S3 bucket to publish events 8

S3 Extractor configuration **8**
Prerequisites 8
Configure credentials for Atlas extraction 11
Extraction Command 12
Extractor configuration properties 12
Defining what assets to extract metadata for 13
Running bulk extraction 14
Running incremental extraction 14
Logging Extractor Activity 15

S3 actions that produce or update Atlas entities **16**

S3 entities created in Atlas **17**
AWS S3 Base 19
AWS S3 Container 19
AWS S3 Contained 19
AWS S3 Bucket 20
AWS S3 Object 20
AWS S3 Directory 21
S3 relationships 22
S3 entity audit entries 24

Amazon S3 metadata collection

Amazon Simple Storage Service (S3) is a storage solution offered by Amazon Web Services (AWS) that provides highly available storage in the cloud. Clusters deployed in the AWS cloud and in on-premise data centers are using Amazon S3 as persistent storage. Common use cases include BDR (backup and disaster recovery) and persistent storage for transient clusters deployed to the cloud, such as storage for ETL workload input and output.

As with data stored on HDFS and processed using compute engines like Hive and Impala, Atlas can obtain metadata and lineage from Amazon S3 storage. The S3 extraction logic is provided as stand-alone scripts for bulk and incremental metadata collection that you can configure to run through a scheduler.

Typically, you would run the extractor in bulk mode once, then run the extractor in incremental mode on a regular schedule. This method allows Atlas to track changes that occur to data assets; it requires that the bucket publish events to an SQS Queue so the extractor can process all events relevant to the data assets being tracked.

Alternatively, you can run the bulk extraction periodically. With this method, Atlas may miss updates that occur between extractions and will not acknowledge when data assets are deleted.

When running the bulk extractor, Atlas collects metadata for current buckets and objects.

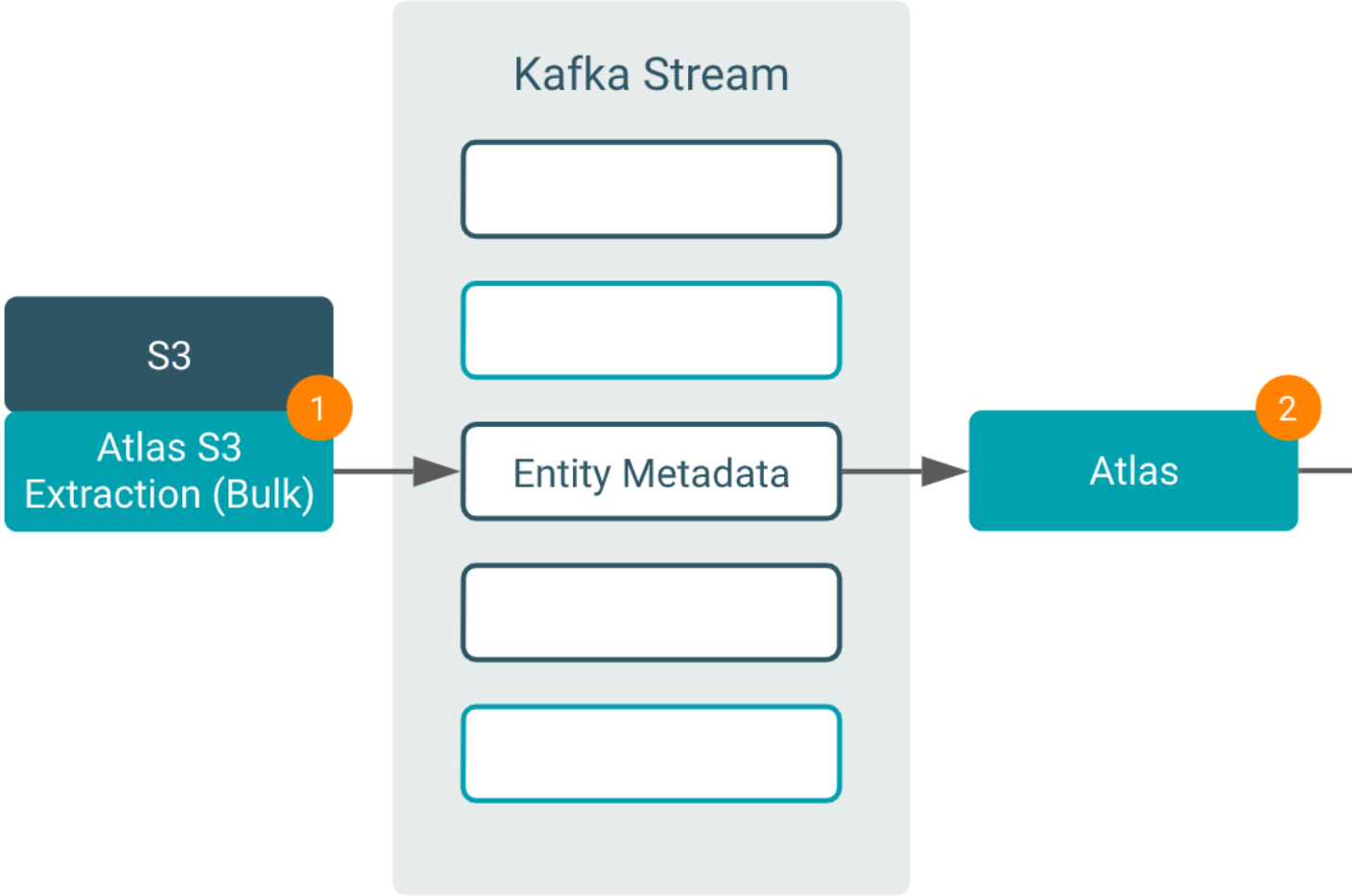

1. The bulk S3 extractor collects metadata from S3 and publishes it on a Kafka topic.
2. Atlas reads the message from the topic and compares it to existing metadata and lineage information.
3. Atlas creates and updates the appropriate entities.

When running the incremental extractor, events in S3 are communicated to Atlas through an SQS Queue:

1. When an action occurs in the S3 bucket.
2. The bucket publishes an event to an SQS queue.
3. The incremental S3 extractor runs and collects the events from the SQS queue and converts them into Atlas entity metadata.
4. The extractor publishes the metadata on a Kafka topic.
5. Atlas reads the message from the topic and compares it to existing metadata and lineage information.
6. Atlas creates and updates the appropriate entities.

The following table lists some differences between object types and supported features offered by Amazon S3 and how those are supported by Atlas:

Feature	Amazon S3	Atlas
System-defined metadata includes properties such as Date, Content-Length, Last-Modified. Some system-defined properties comprise the technical attributes for the object in Atlas.	✓	✓
User-defined metadata consists of custom key-value pairs (in which each key is prefixed with x-amz-meta-) that can be used to describe objects on Amazon S3. Atlas does not collect this kind of metadata; Atlas collects system-defined metadata and tags.	✓	⊘
Versioning. Atlas maintains a single version of a S3 object regardless of how many versions are created in S3. The extractor takes the latest version of an object and replaces previous versions in Atlas. The Atlas audit log will show the updates. If, after metadata is extracted, the latest object's version is deleted, the next extraction will update the Atlas entity metadata with that of the earlier version (which is now the latest).	✓	⊘
Unnamed directories. Unnamed directories are ignored by Atlas.	✓	⊘
Object lifecycle rules. See Object Lifecycle Rules Constraints for more information. Atlas does not support lifecycle rules that remove objects from Amazon S3. For example, an object lifecycle rule that removes objects older than n days deletes the object from Amazon S3 but the event is not tracked by Atlas. This limitation applies to removing objects only.	✓	✓
Amazon Simple Queue Service (SQS). If an SQS queue is configured, Atlas uses Amazon SQS queue for incremental extraction. The queue holds event messages such as notifications of updates or deletes that allow Atlas to track changes for a given object, such as uploading a new version of an object.	✓	✓
Hierarchy in object storage. Atlas uses metadata from S3 to produce a file-system hierarchy.	✓	✓

Feature	Amazon S3	Atlas
Encryption. S3 supports bucket-level and object level encryption. Atlas collects encryption information as metadata. For reading, assuming the AWS credentials are configured such that Atlas can access buckets to collect metadata, the encryption choices are transparent to Atlas.	✓	✓

Accessing AWS

Atlas is integrated with IDBroker for access to the AWS session credential using Kerberos credentials for the atlas service user.

The Atlas S3 extractor authenticates with the service user and receives a delegation token fom IDBroker. The extractor uses the token to request AWS credentials. By default, this credential has a lifetime of seven days, which can be adjusted in Cloudera Manager by modifying the Knox configuration property IDBroker Knox Token TTL.

If IDBroker is not configured, you can provide AWS credentials (AWS access key and secret key or temporary security credential) on the extractor command line or in a configuration file.

AWS objects and inferred hierarchy

Atlas recreates the inferred hierarchy of objects in S3 by creating relationships among objects.

Atlas creates entities for S3 objects following a file system path model. Virtual objects for intermediate directory names are ignored. For example, s3://sampleBucket/testDb/testTable/data.orc and s3://sampleBucket//testDb/testTable/data.orc are treated as the same.

AWS object lifecycle

Atlas can represent the lifecycle of a given S3 object when you use the incremental extractor to collect creation and deletion events from S3.

To collect these events, you need to provide an existing SQS queue or create a new one and to configure your S3 bucket to publish the events to the queue.

Note: Note that when you configure Amazon S3 Lifecycle policies to automatically delete versions of objects or objects after an interval, S3 does not send event notifications and therefore Atlas will not reflect that these objects have been deleted.

AWS configuration

You need the following items to be set up for AWS before you can extract metadata from an S3 bucket:

- AWS credentials
  - The account for this AWS credential must have appropriate permissions to read from S3 buckets and the configured SQS Queue.
  - If you are using Cloudera Public Cloud in the cloud, these credentials are configured as part of SDX and there isn't anything extra you need to do.
  - If you are using Cloudera in a data center (installed on-premise or on virtual hardware in the cloud), you need to supply the extractor with AWS credentials. See Configure credentials for Atlas extraction.

- S3 Bucket access policies
  - If you use a bucket other than the one specified by default when creating a Cloudera environment, you need to update the aws-cdp-bucket-access-policy with the additional bucket as a resource. See Minimal setup for cloud storage.
- S3 Bucket encryption
  - See AWS documentation "Amazon S3 default encryption for S3 buckets". Your choices for bucket and object encryption are transparent to Apache Atlas as Atlas accesses the bucket through the authorized credentials you provide.
- S3 Bucket event publication
  - If you intend to collect all activity that occurs on the data assets you are tracking, you should configure the buckets to publish, update and delete events. See To configure an S3 bucket to publish events.
- SQS queue
  - Update an existing queue or configure a new one to pass S3 event information to Atlas. See To configure an SQS queue suitable for Atlas extraction.

To configure an SQS queue suitable for Atlas extraction

Use the following information to configure an SQS queue for Atlas extraction.

Prerequisite: You must add Amazon Resource Names (ARN) of the IAM role for the user / group configured under Cloudera. For more information, see Onboarding Cloudera users and groups for cloud storage.

1. Log in to the AWS Management Console (https://aws.amazon.com/console/) with the AWS account (IAM user).

   Make sure that there are adequate privileges to administer the queue and the buckets involved in metadata extraction.
2. Open the Simple Queue Service setup page and select Services > Application Integration > Simple Queue Service.
3. Click Create New Queue or Get Started Now, if the region has no configured queues.
4. For each region that has Amazon S3 buckets, create a queue as follows:

   a. Click Create Queue. Enter a Queue Name, name it, for example: aws-s3-cdp-atlas-events, click the Standard Queue (not FIFO), and next under configuration you must configure the queue using the following settings:

   Default Visibility Timeout =10 minutes

   Message Retention Period =14 days

   Delivery Delay = 0 seconds

   Receive Message Wait Time = 0 seconds

   Maximum message size = 256 KB

   b. Under Access Policy, choose method Advanced and insert the access policy similar to below:

```
{
"Version": "2008-10-17",
"Id": "__default_policy_ID",
"Statement": [
{
“Xxxx”: "__owner_statement",
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam::00000000000:root"
},
"Action": "SQS:ReceiveMessage",
"Resource": "arn:aws:sqs:us-west-1:00000000000:aws-s3-cdp-atlas-events"
},
{
"xxxx": "__sender_statement",
```

```
"Effect": "Allow",
"Principal": {
"Service": "s3.amazonaws.com"
},
"Action": "SQS:SendMessage",
"Resource": "arn:aws:sqs:us-west-1:00000000000:aws-s3-cdp-atlas-events"
}
]
}
```

Note: The value above in the snippet arn:aws:iam::00000000000:root is the ARN for the AWS role mentioned as indicated in the prerequisite.

c. Configure at rest Encryption to meet your organization standards. In-transit encryption is configured automatically.

Caution: Do NOT change other default configuration settings.

d. Click Create Queue.

5. Repeat this process for each region that has Amazon S3 buckets.

For more information about creating an SQS queue, see Creating an Amazon SQS queue.

To configure an S3 bucket to publish events

Before you configure an S3 bucket, you must make sure that Cloudera can access the S3 bucket.

For more information, see Minimal setup for cloud storage.

1. Log into AWS.
2. Go to S3 Service.
3. Select the bucket you want to update.
4. Go to Properties > Advanced Settings > Events.
5. In the Events dialog box, click Add notification.
6. Enter a name for the event notification, such as Atlas Create/Delete Events.
7. In the Events list, select All object delete events and All object create events.
8. In Send to, choose SQS Queue.
9. In SQS, enter the name of the existing queue or choose "Add SQS queue ARN" to specify an existing queue with its ARN.
10. Click Save.

S3 Extractor configuration

The Atlas metadata extractor for AWS S3 is a stand-alone application that you can run on the Atlas host as needed to provide comprehensive metadata for data assets stored in S3.

Prerequisites

To setup the extractor follow these steps as part of the prerequisite.

- In Apache Ranger, in the Kafka service policies, add the "Publish" permission for the atlas user in the Ranger Atlas_Hook policy.
  1. Contact an administrator with privileges to update Ranger policies.
  2. Open **Service Manager** > **Kafka policies**.

3. Open the ATLAS_HOOK

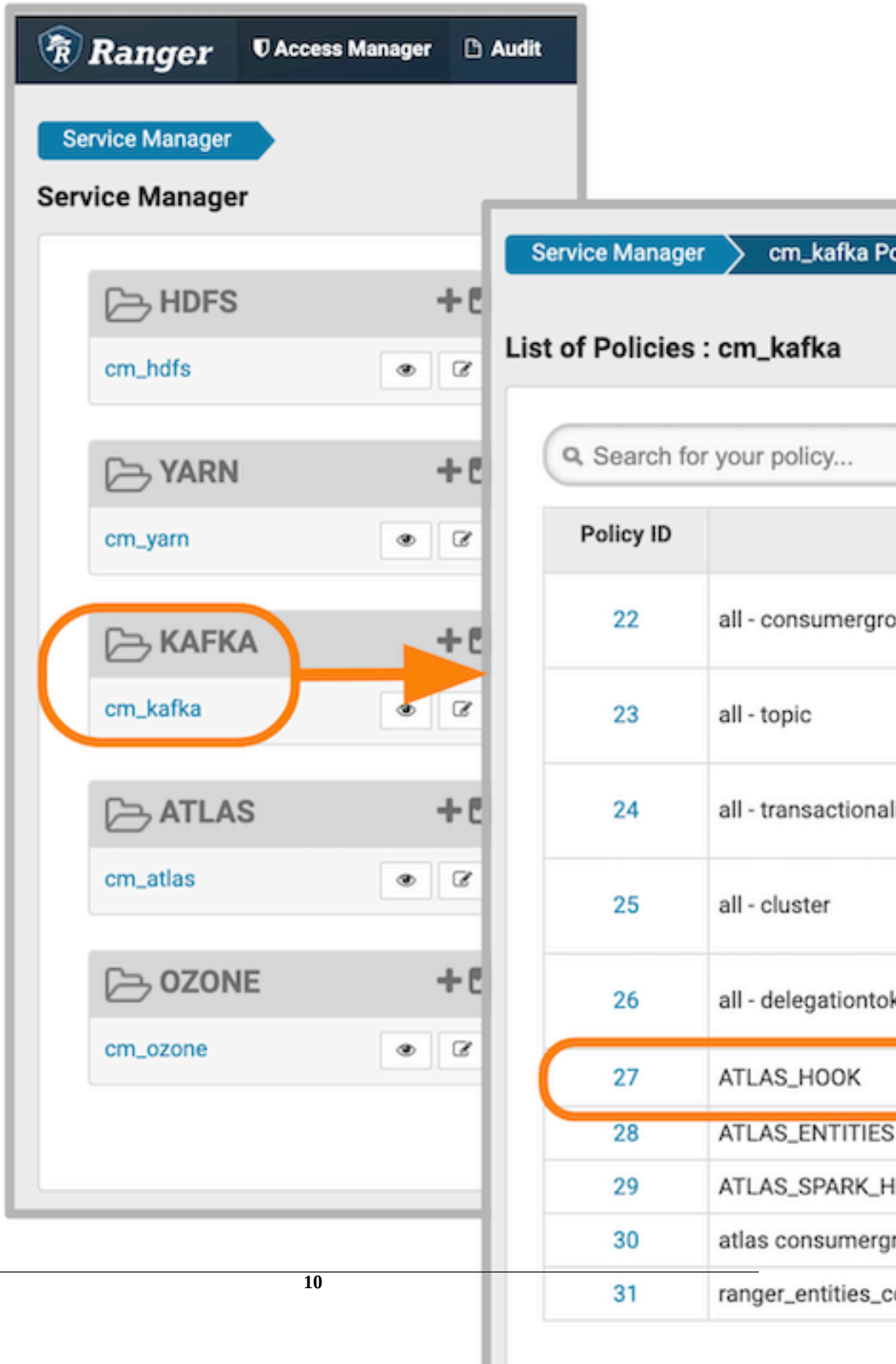

4. In the policy details, look for the atlas user and add the Publish permission to that condition.

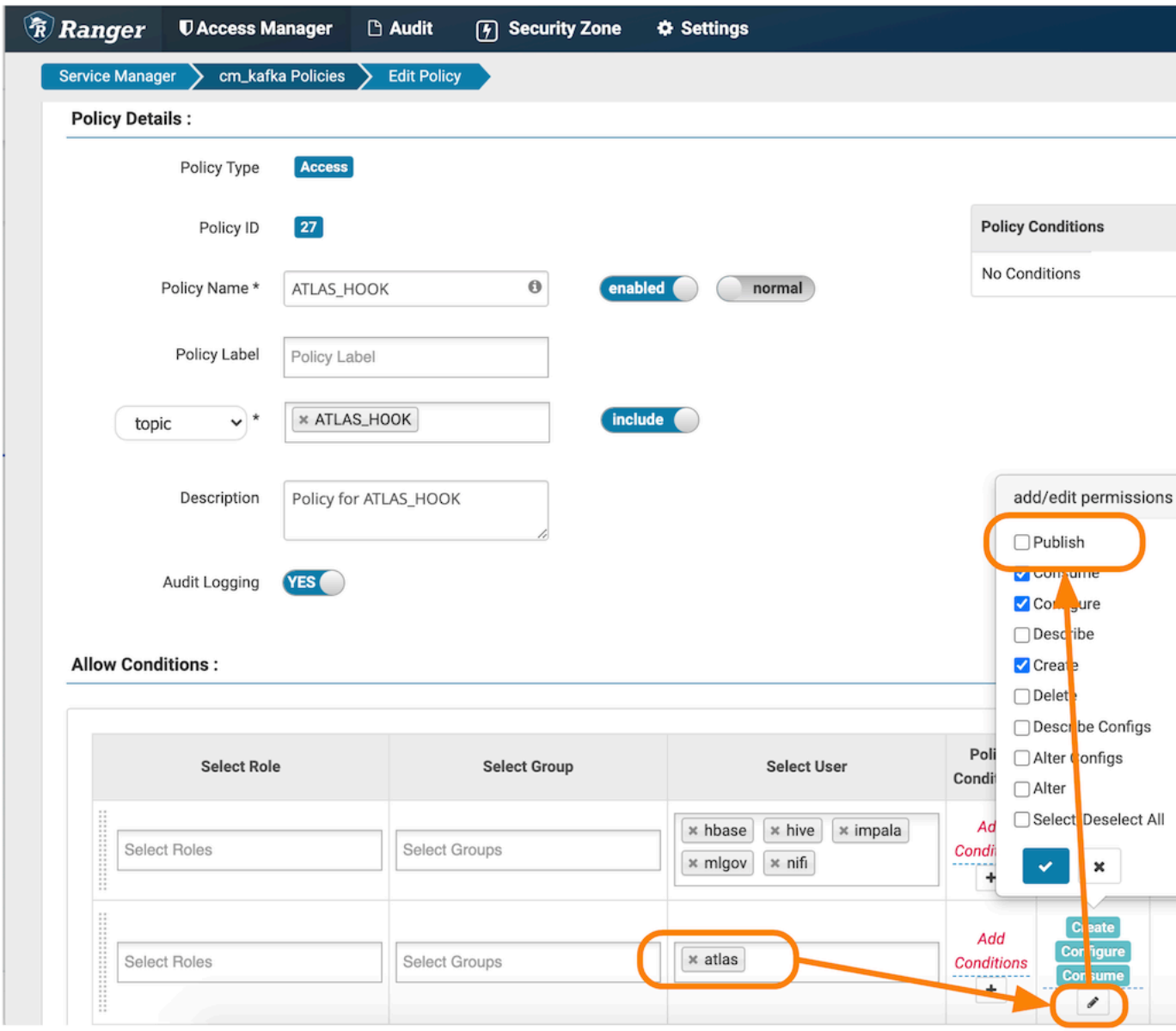

- SSH into the Atlas host. You can locate the Atlas host URL through Cloudera Manager.
- Set JAVA_HOME if it is not already set.

Configure credentials for Atlas extraction

When Atlas is running as part of SDX, you can skip this step but follow other instructions.

The extractor is authenticated with AWS through integration with the IDBroker. To authenticate the extractor otherwise, you'll need to provide AWS credentials to the extractor, either through a configuration file or on the command line.

Note: Any changes to the AWS credentials (for example, if you rotate credentials on a regular basis) must be for the same AWS account (IAM user). Changing the AWS credentials to those of a different IAM user results in errors from the Amazon Simple Queue Service (used transparently by Atlas).

Atlas can use either of the following authentication methods:

1. Basic credentials: AWS access key and secret key.
2. Temporary credentials: AWS session token along with access and secret key. See Getting Temporary Credentials with AWS STS.

Both methods require you to specify the region where the authentication is valid.

- If the connection to AWS S3 expires, Atlas will retry X times to re-establish the connection. This property atlas.s3.extraction.max.reconnect.count controls the number of retries, which defaults to 2.

Extraction Command

The extractor script is located in the extractors/bin directory in $ATLAS_HOME.

By default, the location is /opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/aws-s3-extractor.sh

Note: The extractor configuration file includes properties that require AWS S3 configuration values. Be sure to set those values in the configuration file before running the command.

Using the following command to start S3 extraction:

```
$ aws-s3-extractor.sh [--config <config file>]
                      [--extraction <mode>]
                      [--failOnError]
                      [--logdir <log directory>]
```

Command Line Option	Description
`-c <config file>` `--config <config file>`	The path of the configuration file containing extraction configuration properties. See Extractor configuration properties on page 12. Defaults to $ATLAS_HOME/extractors/conf/aws-s3.conf
`-e <mode>` `--extraction <mode>`	The type of extraction to be done, “Incremental” or “Bulk”. If not specified, the extraction mode is "Bulk".
-f --failOnError	Flag to set the extraction to stop when it encounters an exception. Without the flag, the extraction continues after exceptions.

Extractor configuration properties

Some of the extractor configuration properties.

Configuration Parameter (prefixed with atlas.s3.extraction)	Purpose
.aws.region	Required. AWS S3 region for extraction. Cloudera Public Cloud: provided by IDBroker integration. Cloudera Data Center: the region must be provided.
.access.key .secret.key .session.token	AWS S3 credentials. Cloudera Public Cloud: provided by IDBroker integration. Cloudera Data Center: • For basic authentication, provide access and secret keys. • For session based authentication, provide all three credentials. Configure credentials for Atlas extraction on page 11.

Configuration Parameter (prefixed with atlas.s3.extraction)	Purpose
.whitelist.paths	Set of data assets to include in metadata extraction. If no paths are included, the extractor extracts metadata for all buckets. Buckets inside the region and accessible by the credential used to run the extractor. For public cloud, this is defined in aws-cdp-bucket-access-policy. The list can include buckets or paths; separate multiple values with commas. See Defining what assets to extract metadata for on page 13.
.blacklist.paths	Set of data assets to exclude from metadata extraction. The list can include buckets or paths; separate multiple values with commas. See Defining what assets to extract metadata for on page 13.
.max.object.per.call	Number of objects returned in one call to AWS S3 in bulk extraction. Defaults to 1000.
.resume.from.progress.file	Enable bulk extraction resume. Disabled by default.
.progress.file	Name of the progress file that tracks bulk extraction that will be used if the extraction has to be resumed mid-way after a failure. This file is created in the Atlas log directory. Defaults to extractor_progress_file.log
.max.reconnect.count	Number of times the extractor will try to reconnect to AWS in case of credentials expiring or retryable exception from AWS S3. Defaults to 2.
.fs.scheme	File system scheme used in case of AWS S3. At this time, S3a is the only option.
.incremental.queueName	Required for incremental mode. AWS SQS Queue name configured to get the AWS S3 bucket events.
.incremental.messagesPerRequest	The number of messages the extractor will try to get from the SQS queue in one call. Defaults to 10.
.incremental.requestWaitTime	The interval that the extractor will wait to collect messages so long as the limit set in .incremental.messagesPerRequest hasn't been met. Defaults to 20 seconds.
.incremental.max.retry	Maximum retry count if the SQS queue doesn't respond during an extractor request. Defaults to 20.

Defining what assets to extract metadata for

By default, the Atlas S3 extractor collects metadata for all buckets that are visible through the active IAM policy.

You can specify the objects to include using configuration properties to include buckets, directory trees, and specific objects (whitelist.paths) or exclude buckets, directory trees, and specific objects (blacklist.paths). When you include a path in the whitelist, Atlas extracts metadata for that directory object and all objects underneath it.

The include and exclude path lists are comma-separated lists of paths for any given bucket. Each item in the list can be as general as a bucket name or a directory path, or as specific as a low level directory or even a single object. The path names in the lists are formatted as S3 URIs, so it may be that they include repeated slash characters to represent virtual objects in S3.

If the same path ends up on both lists, the exclude list wins.

The following examples show how to include objects in or exclude objects from metadata collection.

This example includes a single file from the 2018 directory and all files in the data subdirectory for 2019 and 2020:

atlas.s3.extraction.whitelist.paths=s3a://acme-finance-set4/year-end/2018/data/storage/only-this-file.json,acme-finance-set4/year-end/2019/data,acme-finance-set4/year-end/2020/data

This example includes a single file from one bucket and all contents of a second bucket:

atlas.s3.extraction.whitelist.paths=s3a://demo-bucket/demo.txt, s3a://acme-finance-set4/

This example excludes specific files from a directory. If no include paths were given, only these files would be excluded from all the buckets available.

atlas.s3.extraction.blacklist.paths=s3a://acme-finance-set4/year-end/admin/tools-archive.zip,atlas.s3.extraction.blacklist.paths=s3a://acme-finance-set4/year-end/reporting/template.xls

Running bulk extraction

Bulk extraction takes a snapshot of the metadata for the current version of all objects defined by the allowlist as of the initial start time of the extraction.

If the bulk extraction process runs for a few hours, there may be additions or modifications to AWS S3 content after the start of the extraction. The changes after the extraction start time are not included in the extraction.

The first time bulk extraction runs, it creates entities in Atlas for each S3 bucket and object. If you run bulk extraction again, it will:

- Update Atlas entities with new information it finds for buckets or objects whose qualifiedNames match existing entities.
- Create new entities for any new buckets or objects that are not found in Atlas.

The second extractions will NOT:

- Mark entities deleted in Atlas if the object from the first extraction does not exist at the time of the second extraction.
- Collect version information for objects other than the current version.

The following command line example runs the bulk extraction. Assuming the mandatory properties are set in the default configuration file, no parameters are required:

```
/opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/aws-s3-extractor.sh
```

Running incremental extraction

Any object added to S3, including new buckets and new versions of existing objects, publishes an event to AWS SQS (as configured by the user).

Incremental extraction consumes the events from SQS queue and creates corresponding Atlas entities.

After running the bulk extraction once, you can run incremental extraction to update Atlas metadata based on the events tracked in S3. This method has the benefit of tracking changes that may not be captured in successive bulk extraction, such as when more than one object version is created between extractions or when objects are deleted.

Note: Before running incremental extraction, you must set up event tracking in the AWS S3 buckets and configure an SQS queue to collect the S3 events.

The incremental extraction operation reads all events produced by S3, then filters the list based on the include and exclude configurations for the extractor. The appropriate messages are passed into the Kafka queue and consumed by Atlas. Because the messages sit in the SQS queue, you don't need to run the incremental extraction operation constantly: when it runs, it will go through all the messages waiting in the queue. Run the extractor often enough that the metadata is not stale by the time it processes, but not so often that it takes longer to run than the time between runs. In addition, note that the SQS Queue does not maintain the events forever. By default, the messages stay for 4 days; you can configure this value to hold the events longer to protect against the chance that the extractor isn't able to process events in time.

In addition to the include and exclude lists described in Defining what assets to extract metadata for on page 13, the configuration properties relevant to incremental extraction control the connection to the SQS Queue and how long the extraction actively waits for events from S3. You must configure the name of the queue that the events are published on. You can control how long the extractor waits to receive messages from the queue and how many

messages to wait for. When configuring the number of messages to collect, remember that the extractor collects all events from the queue, not just the events corresponding to objects in the allowlist.

In addition, you can set the timeout or retry time so the extractor doesn't keep running even when it isn't able to connect to the queue.

You must plan whether to set up a repeating job to run incremental extraction. Assuming you have a lot of changes that you want to track:

- Incremental extraction takes more time to process the individual changes than running bulk extraction.
- Incremental extraction maintains a complete history of objects (shows updates and deletes)
- Batch is faster, but only captures current state and leaves deleted entities in Atlas

In conclusion, if you want to capture the history of S3 objects, run bulk extraction once then run incremental extraction on a regular basis. As you have more experience with the volume of changes and the speed of the extraction, you can determine if you should run the incremental extraction more frequently.

The following command line example runs the incremental extraction. Assuming the mandatory properties are set in the default configuration file, only the parameter to enable incremental mode is required:

```
/opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/aws-s3-extractor.sh -e IN
CREMENTAL
```

Logging Extractor Activity

The extractor produces a log file in the same location as the Atlas server log file. While the server log is available through Cloudera Manager, the extractor file is not.

About this task

To locate the extractor log file.

Procedure

1. SSH into the Atlas host.
2. Find the active Atlas configuration:

   $ ps -ef | grep atlas.conf

   In the output, look for the string

   ```
   atlas.conf
   ```

   For example:

   atlas.conf=/var/run/cloudera-scm-agent/process/99-atlas-ATLAS_SERVER/conf
3. In the configuration directory you found in the previous step, read the atlas-log4j.properties file.

   Look for the log.dir entry. For example:

   log.dir=/var/log/atlas
4. In the log directory, you'll see the extractor log file and the progress file:
   - aws-s3-extractor.log
   - extractor_progress_file.log

S3 actions that produce or update Atlas entities

As objects are created in S3, Atlas generates entities to represent those objects. Atlas does not create processes to represent S3 operations.

The following table lists the S3 actions that produce or update metadata in Atlas.

S3 Evemt	Bult Extraction	Incremental Extraction
Bucket		
• creation	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current bucket metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete (No S3 event is produced to indicate that a bucket has been deleted.)	No	No
Object (directory)		
• creation	Yes	Yes
• upload	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current object metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete	No	Yes#
Object (file)		
• creation	Yes	Yes
• upload (When you upload a "folder" to S3, the objects in the folder are prefixed with the folder name.)	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current object metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete	No	Yes#

Notable actions in S3 that do NOT produce metadata entities include any actions that affect only data and not metadata. Actions that do not create Atlas entities include:

- Bucket deletion does not produce an event and is ignored by Atlas.
- Events associated with subfolders when a folder is uploaded. When you upload a "folder" to S3, the objects in the folder are prefixed with the folder name, so the folder and any subfolders are not created as objects in S3. Atlas ignores metadata or events for subdirectories included in the uploaded folder; Atlas tracks metadata for the uploaded folder and the data asset objects in the uploaded folder.
- Event notifications produced when a version other than the latest version is deleted.

S3 entities created in Atlas

Note that previous versions of Atlas included other metadata models for S3 data. The version 2 (V2) models are supported as of Cloudera 7.2.0.

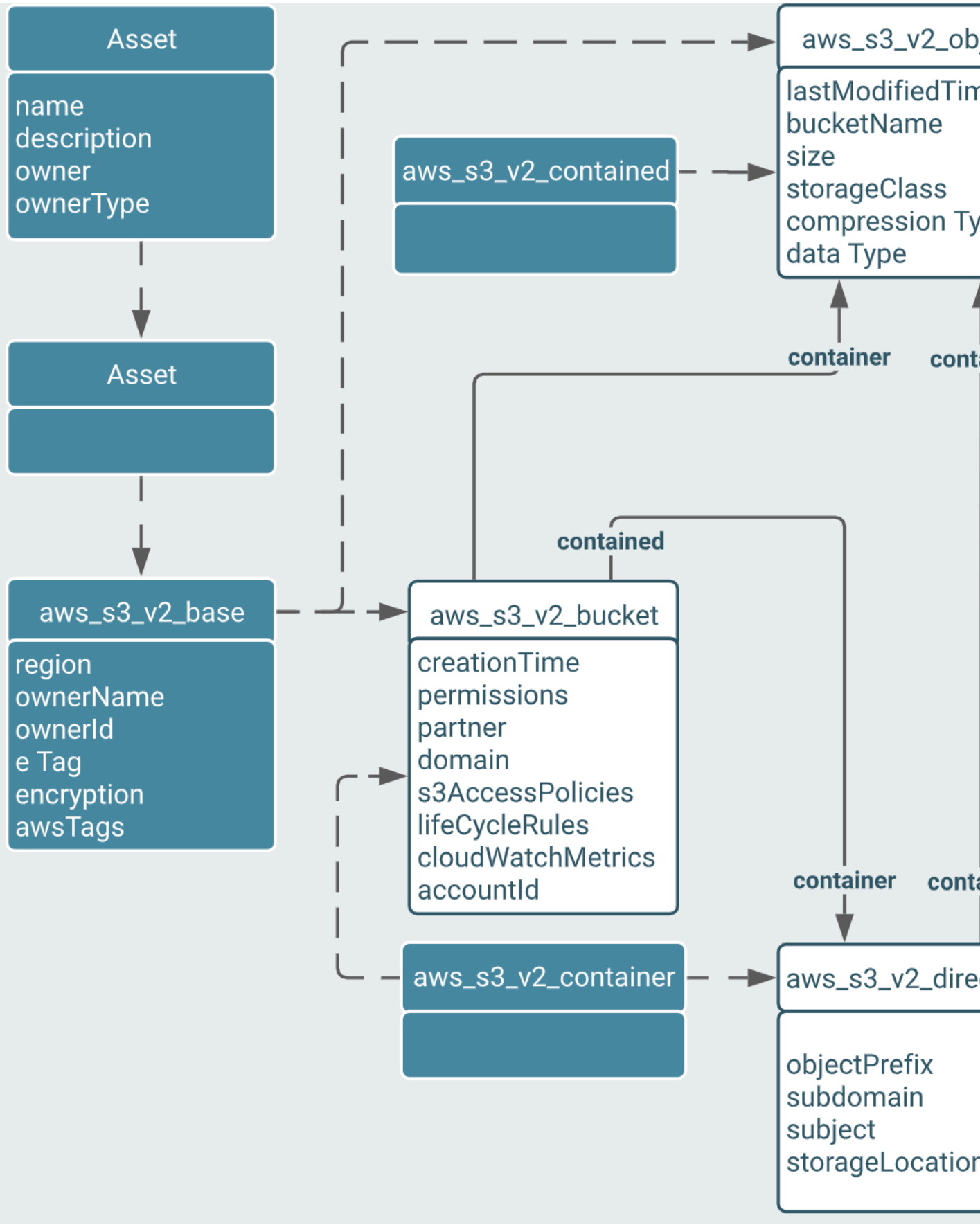
Asset
name
description
owner
ownerType
Asset
aws_s3_v2_base
region
ownerName
ownerId
e Tag
encryption
awsTags
aws_s3_v2_ob
lastModifiedTim
bucketName
size
storageClass
compression Ty
data Type
aws_s3_v2_contained
container
cont
contained
aws_s3_v2_bucket
creationTime
permissions
partner
domain
s3AccessPolicies
lifeCycleRules
cloudWatchMetrics
accountId
container
conta
aws_s3_v2_container
aws_s3_v2_dire
objectPrefix
subdomain
subject
storageLocatior

AWS S3 Base

S3 entities that represent S3 buckets and objects inherit attributes from the S3 base entity type. Atlas does not generate entities of this type; it is used only as a supertype for other entity definitions.

Identifier	Description
typeName	AWS_S3_V2_BASE
name	Inherited from Asset supertype.
description	Inherited from Asset supertype.
owner	Inherited from Asset supertype.
ownerType	Inherited from Asset supertype.
qualifiedName	Fully qualified name for each entity at Atlas which is used to create lineage. See the child entity types for details on how the qualified name is generated.
region	AWS S3 region as reported by S3.
ownerName	Owner as reported by S3.
ownerID	Owner ID as reported by S3.
eTag	A hash of an object as reported by S3. This value is updated when a new version of an object is added to a bucket.
encryption	Encryption algorithm as reported by S3.
awsTags	Object tags defined in S3. Atlas stores S3 tags as an array of key-value pairs.

AWS S3 Container

The S3 model includes two entity definitions that generalize the hierarchical relationship among the other entity types. Atlas does not generate entities of this type; it is used only as a supertype for other entity definitions. The "container" entity type includes a relationship that represents the one-to-many relationship between a directory and objects in the directory. The "contained" relationship looks the other way to represent the one-to-one relationship between an object and its parent, either a directory or a bucket.

The entity definitions for aws_s3_v2_bucket and aws_s3_v2_directory inherit the attributes from the container entity definition.

name	AWS_S3_V2_CONTAINER
relationship: contained	The relationship between a bucket or directory and an array of one or more objects, including subdirectories.

AWS S3 Contained

The S3 model includes two entity definitions that generalize the hierarchical relationship among the other entity types. Atlas does not generate entities of this type; it is used only as a supertype for other entity definitions. The "contained" relationship represents the one-to-one relationship between an object and its parent, either a directory or a bucket. The "container" entity type provides the opposite relationship: the one-to-many relationship between a bucket or directory and objects in the directory.

The entity definition for aws_s3_v2_object inherits the attributes from the contained entity definition.

Identifier	Description
name	AWS_S3_V2_CONTAINED

Identifier	Description
relationship: container	The one-to-one relationship between an object and its parent directory or bucket.

AWS S3 Bucket

The Atlas entity type for bucket metadata inherits attributes from the AWS_S3_V2_BASE entity type and includes additional attributes specific to bucket metadata.

Identifier	Description
typeName	AWS_S3_V2_BUCKET
name	Name of the object as reported by S3.
description	Description provided in Atlas metadata. No value is collected from S3.
owner	Atlas user who ran the metadata extractor.
ownerType	Not used.
qualifiedName	Fully qualified name of the bucket. The name includes Atlas's version of the S3 URI followed by @cluster_name. For example: s3a://acme-finance-bucket@cm
region	AWS S3 region as reported by S3.
ownerName	Owner as reported by S3.
ownerID	Owner ID as reported by S3.
eTag	A hash of the bucket as reported by S3.
encryption	Bucket-level encryption algorithm as reported by S3.
awsTags	Object tags defined in S3. Atlas stores S3 tags as an array of key-value pairs.
permission	Permissions at AWS S3
creationTime	Creation data as reported by S3. DIstinct from Atlas entity create time.
partner	Not used. Supports backward compatibility.
domain	Not used. Supports backward compatibility.
s3AccessPolicies	Not used. Supports backward compatibility.
lifeCycleRules	Not used. Supports backward compatibility.
cloudWatchMetrics	Not used. Supports backward compatibility.
accountId	Account ID as reported by S3.
relationship: contained	The relationship between a bucket and an array of one or more objects, including directories.

AWS S3 Object

Though AWS S3 includes both files and directories as objects, the Atlas model creates separate entity types for objects from directories.

The Atlas entity type for object metadata inherits attributes from the AWS_S3_V2_BASE entity type and includes additional attributes specific to object metadata. Note that Atlas makes a distinction between files and directories. Atlas creates entities of this type for files; it creates entity types of AWS_S3_V2_DIRECTORY for directories.

Table 1:

Identifier	Description
typeName	AWS_S3_V2_OBJECT
name	Name of the object as reported by S3.
description	Description provided in Atlas metadata. No value is collected from S3.
owner	
ownerType	Not used.
qualifiedName	Fully qualified name of the object. The name includes Atlas's version of the S3 URI followed by @cluster_name. For example: s3a://acme-finance-bucket/2018directory/specific_file.json@cm
region	AWS S3 region as reported by S3.
ownerName	Owner as reported by S3.
ownerID	Owner ID as reported by S3.
eTag	A hash of an object as reported by S3. This value is updated when a new version of an object is added to a bucket.
encryption	Encryption algorithm as reported by S3. If no object-level encryption is specified, this value is the same as the bucket-level encryption.
awsTags	Object tags defined in S3. Atlas stores S3 tags as an array of key-value pairs.
lastModifiedTime	The last modified time of the object as reported from S3.
bucketName	The name of the bucket containing this object.
size	The size of the object as reported from S3.
storageClass	Storage class used for storing the object as reported from S3.
compressionType	Compression type used for the object as reported from S3.
data Type	Owner of the object
relationship: container	The one-to-one relationship between an object and its parent directory or bucket.

AWS S3 Directory

The Atlas entity type for "directory" metadata inherits attributes from the AWS_S3_V2_OBJECT entity type and includes additional attributes specific to object metadata.

Identifier	Description
typeName	AWS_S3_V2_DIRECTORY
name	Name of the object as reported by S3.
description	Description provided in Atlas metadata. No value is collected from S3.
owner	
ownerType	Not used.
qualifiedName	Fully qualified name of the directory object. The name includes Atlas's version of the S3 URI ending with a slash, followed by @cluster_name. For example: s3a://acme-finance-bucket/2018directory/@cm
region	AWS S3 region as reported by S3.
ownerName	Owner as reported by S3.

Identifier	Description
ownerID	Owner ID as reported by S3.
eTag	A hash of an object as reported by S3. This value is updated when a new version of an object is added to a bucket.
encryption	Encryption algorithm as reported by S3. If no directory-level encryption is specified, this value is the same as the bucket-level encryption.
awsTags	Object tags defined in S3. Atlas stores S3 tags as an array of key-value pairs.
storageLocation	Not used. Supports backward compatibility.
objectPrefix	Not used. Supports backward compatibility.
subdomain	Not used. Supports backward compatibility.
subject	Not used. Supports backward compatibility.
relationship: contained	The relationship between a directory and an array of one or more objects, including other directories.
relationship: container	The one-to-one relationship between a directory and its parent directory or bucket.

S3 relationships

The S3 model includes relationships that describe the perceived hierarchy among objects in a bucket. These relationships are created through "container" and "contained" relationship attributes.

Example of Atlas S3 Lineage

An example of S3 lineage that includes S3 objects connected to a Hive table.

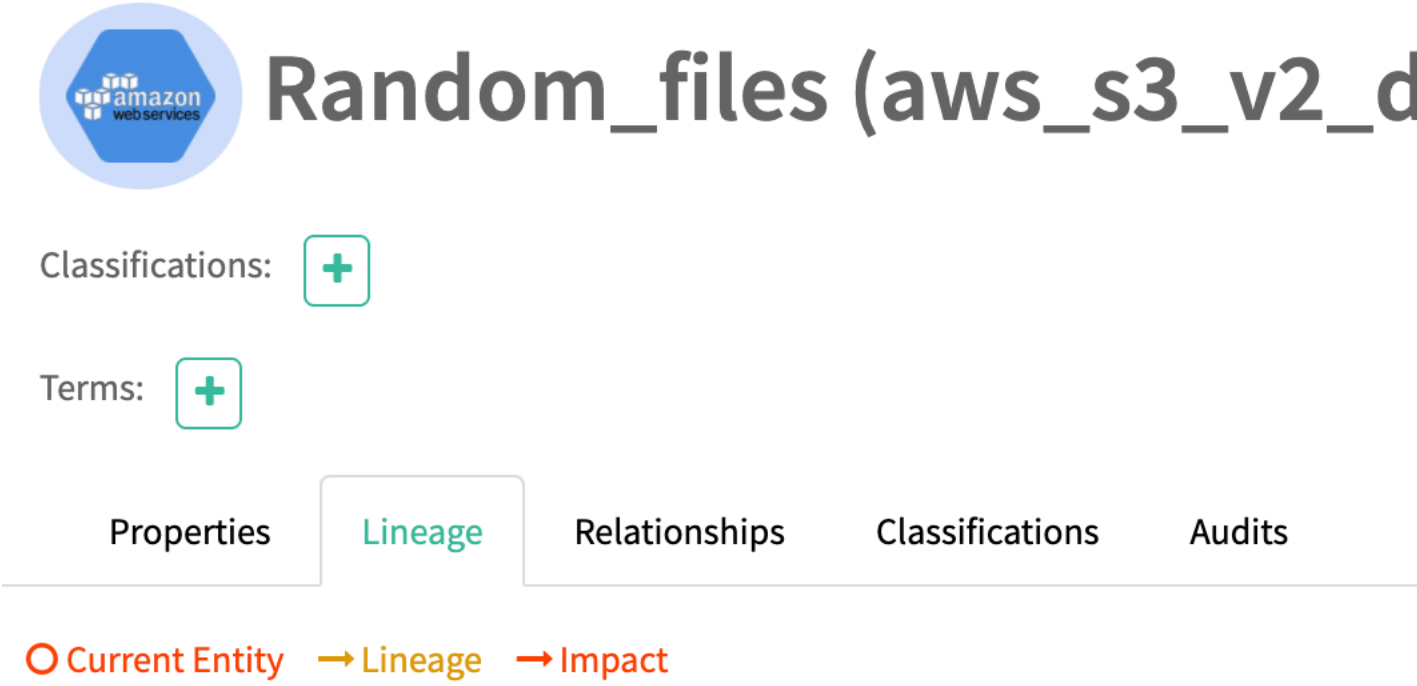
Random_files (aws_s3_v2_d
Classifications:
Terms:
Properties
Lineage
Relationships
Classifications
Audits
Current Entity
Lineage
Impact

S3 entity audit entries

The number of Atlas audits for S3 entities may vary based on whether or not you are using incremental extraction to collect metadata from S3. For example, if you run bulk extraction against a bucket, then objects in the bucket are updated twice before bulk extraction runs again, Atlas will show the creation audit event followed by a single update audit event. If instead you run bulk extraction against a bucket, then trigger incremental extraction, the Atlas audit will show the creation audit event followed by an update audit event for the first update and an update audit event for the second update.

S3 Evemt	Bult Extraction	Incremental Extraction
Bucket		
• creation	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current bucket metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete (No S3 event is produced to indicate that a bucket has been deleted.)	No	No
Object (directory)		
• creation	Yes	Yes
• upload	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current object metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete	No	Yes#
Object (file)		
• creation	Yes	Yes
• upload (When you upload a "folder" to S3, the objects in the folder are prefixed with the folder name.)	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current object metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete	No	Yes#

Cloudera Runtime 7.3.1

Apache Atlas High Availability

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

About Atlas High Availability.....4

Prerequisites for setting up Atlas HA.....4

Installing Atlas in HA using Cloudera Private Cloud BaseCloudera Public Cloud cluster.....4

About Atlas High Availability

Atlas supports multiple instances of the service in an active-passive configuration.

You can deploy multiple instances of the service on different physical hosts in a cluster. One of these instances is automatically selected as an 'active' instance to respond to user requests. All other instances are deemed 'passive'. If the 'active' instance is unavailable either because it is deliberately stopped, or due to an unexpected failure, one of the other instances automatically changes to an 'active' mode and starts servicing requests.

An 'active' instance is the only instance that can respond to user requests. A 'passive' instance redirects requests coming to it directly (using HTTP redirect) to the current 'active' instance. A passive instance does not service any requests. However, all instances (both active and passive), respond to admin status requests that return heartbeat information about that instance.

How Atlas High Availability works

Atlas uses Zookeeper to know about all available Atlas service instances within the cluster. It uses Zookeeper's electoral election to determine the instance that needs to be active. Once the leader is elected, other instances set themselves as passive and do not participate in servicing requests. Atlas thus achieves active-passive configuration for high availability (HA). The active instance is also the one that processes the notification messages from the hook.

When an instance goes down or is turned off, Zookeeper's electoral election starts and elects a new leader. ZooKeeper ensures that one instance is always available for processing requests.

The other server performs the real (active) work only when the "active" server goes down.

Currently, the only way to change the mode from active to passive is to shutdown the active mode; the other server instance receives a message (through ZooKeeper) to make itself active.

Note:

You cannot change an instance from active to passive mode using Cloudera Manager.

Prerequisites for setting up Atlas HA

You must ensure that the following prerequisites are met before you set up High Availability (HA) on Apache Atlas:

- You must have a cluster with more than one node.
- You must have a Cloudera Private Cloud BaseCloudera Public Cloud on-premise cluster.
- Atlas service must be up and running on that cluster.
- Apache ZooKeeper must be running on a cluster of machines. A minimum of three servers is recommended for the production environment.
- You must select two or more physical nodes for running the Atlas Web Service instances. These nodes define a server ensemble for Atlas.

Installing Atlas in HA using Cloudera Private Cloud BaseCloudera Public Cloud cluster

When you are using the Cloudera Private Cloud BaseCloudera Public Cloud cluster, you must install Apache Atlas in HA using the Cloudera Manager UI.

Procedure

1. Log into Cloudera Manager and verify that Atlas service is running.

2. Stop the Atlas service.
3. Go to Clusters > Select Atlas >Instances > Add Role Instances > Add Atlas Service role to the required host and proceed with the wizard to complete the process.
4. Restart services for stale configurations.

   Note: Cloudera recommends that the selected cluster in the Cloudera Manager must be Kerberos enabled but it is not a mandatory requirement.

5. Using the Cloudera Manager UI, verify that both Atlas services are running (services are green).

   The active Atlas instance can be found using a HTTP status request to each Atlas instance:

   - http://<atlas_host_1>:3100/api/atlas/admin/status
   - http://<atlas_host_2>:3100/api/atlas/admin/status

   The response for ACTIVE instance will be:{"Status":"ACTIVE"} The response for PASSIVE instance will be: {"Status":"PASSIVE"}

   Atlas is currently running in the HA mode (active-passive mode)

Cloudera Runtime 7.3.1

Iceberg support for Atlas

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Iceberg support for Atlas.....4
How Atlas works with Iceberg.....4
Using the Spark shell.....5
Using the Hive shell.....20
Using the Impala shell.....27

Iceberg support for Atlas

Atlas integration with Iceberg helps you identify the Iceberg tables to scan data and provide lineage support. Learn how Atlas works with Iceberg and what schema evolution, partition specification, partition evolution are with examples.

How Atlas works with Iceberg

You can use Atlas to find, organize, and manage different aspects of data about your Iceberg tables and how they relate to each other. This enables a range of data stewardship and regulatory compliance use cases.

The Atlas connectors distinguish between Hive and Iceberg tables. The Iceberg table is available in a typedef format which implies that the underlying data can be retrieved by querying the Iceberg table. All attributes of the Hive table are available in the Iceberg table and this equivalence is achieved by creating the Iceberg table as a sub-type of the underlying Hive table. Optionally, the Iceberg table can also be queried by Hive or Impala engine. For more information about Iceberg and related concepts, see Apache Iceberg features and Apache Iceberg in Cloudera.

Both Iceberg and Hive tables have equality in Atlas in terms of data tagging. Data evolution and transformation are features unique to Iceberg tables. Iceberg adds tables to compute engines including Spark, Hive and Impala using a high-performance table format that works just like a SQL table. Also, the lineage support for Iceberg table is available. For example, when a Hive table is converted to Iceberg format, the lineage is displayed for the conversion process in Atlas UI.

Attention: Whenever a classification is applied on Iceberg entities,Tag-based policies are supported for Iceberg entities.

- Migration of Hive tables to Iceberg is achieved with the following:
  - Using in-place migration by running a Hive query with the ALTER TABLE statement and setting the table properties.
  - Executing CTAS command from Hive table to the Iceberg table.
- Schema evolution allows you to easily change a table's current schema to accommodate data that changes over time. Schema evolution enables you to update the schema that is used to write new data while maintaining backward compatibility with the schemas of your old data. Later the data can be read together assuming all of the data has one schema.
  - Iceberg tables supports the following schema evolution changes:

    Add – add a new column to the table or to a nested struct

    Drop– remove an existing column from the table or a nested struct

    Rename– rename an existing column or field in a nested struct

    Update– widen the type of a column, struct field, map key, map value, or list element

    Reorder – change the order of columns or fields in a nested struct
  - Partition specification allows you to initiate queries faster by grouping similar rows together when writing.

  As an example, queries for log entries from a logs table usually include a time range, like the following query for logs between 10 A.M. and 12 A.M.

  SELECT level, message FROM logs

  WHERE event_time BETWEEN '2018-12-01 10:00:00' AND '2018-12-0112:00:00'

Configuring the logs table to partition by the date of event_time groups log events into files with the same event date. Iceberg keeps track of that date and uses it to skip files for other dates that do not have useful data.

- Partition evolution across Iceberg table partitioning can be updated in an existing table because queries do not reference partition values directly.

When you evolve a partition specification, the old data written with an earlier specification remains unchanged. New data is written using the new specification in a new layout. The metadata for each of the partition versions is stored separately.

Due to this nature of partition evolution, when you start writing queries, you get split planning. This is where each partition layout plans files separately using the filter it derives for that specific partition layout.

Related Information

Using the Spark shell

Using the Hive shell

Using the Impala shell

Apache Iceberg features

Using the Spark shell

Using Spark, you can create an Iceberg table followed by schema evolution, partition specification, and partition evolution.

Before you begin

You must configure the Spark shell as such you have included the valid Spark runtime version.

Run the following command in your Spark shell to create a new Iceberg table

Procedure

1. `spark.sql("CREATE TABLE spark_catalog.default.sample_1 ( id bigint COMMENT 'unique id', data string) USING iceberg");`

2. Navigate accordingly in the Atlas UI to view the changes.
   The following images provide information about Iceberg table creation process.

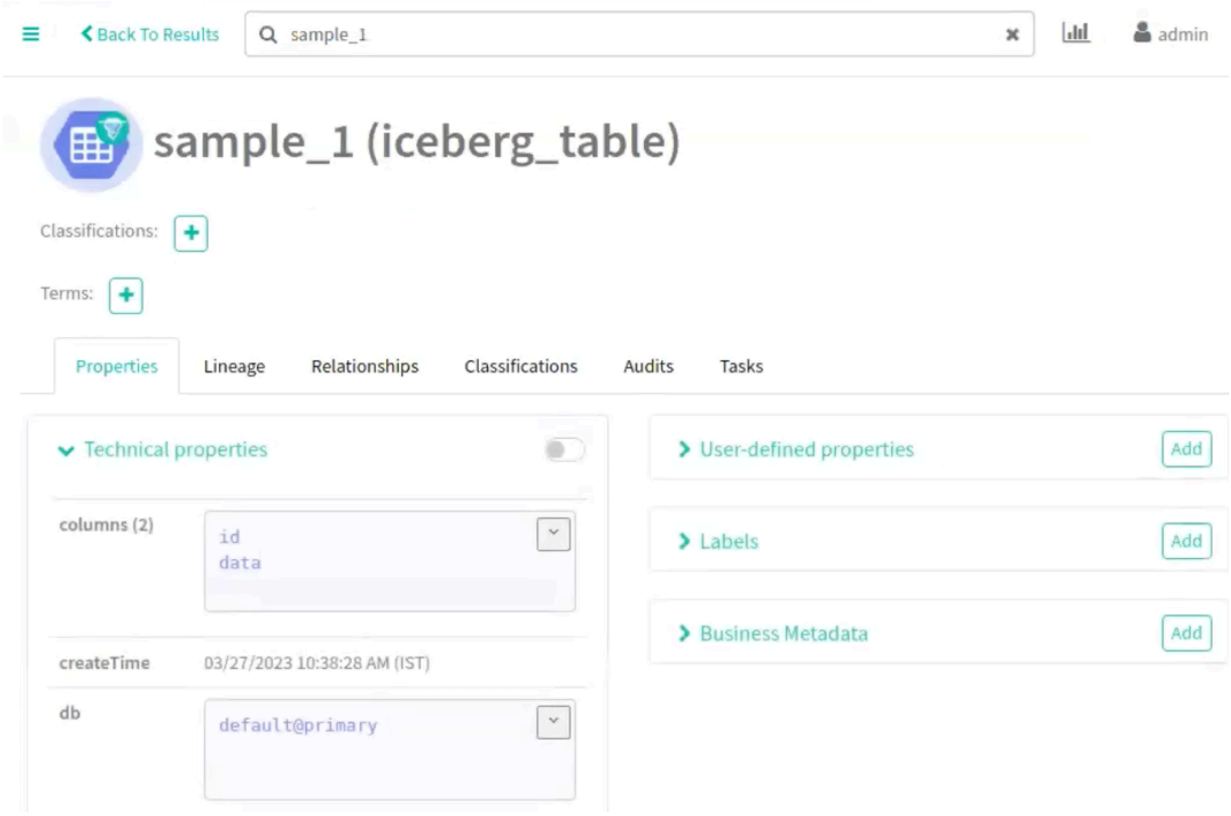

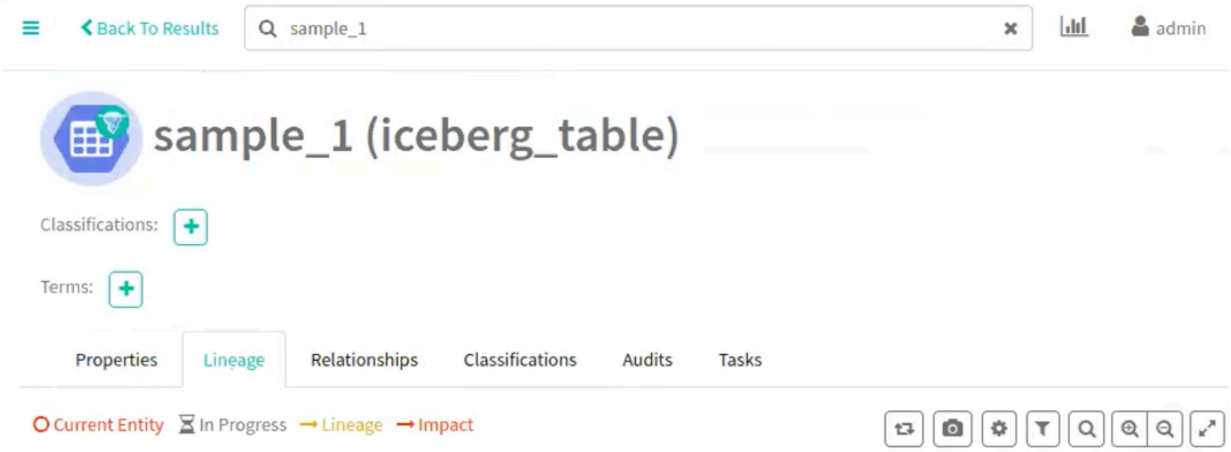

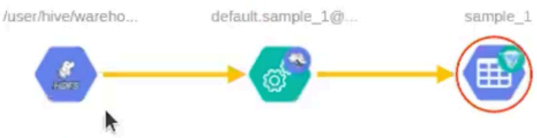

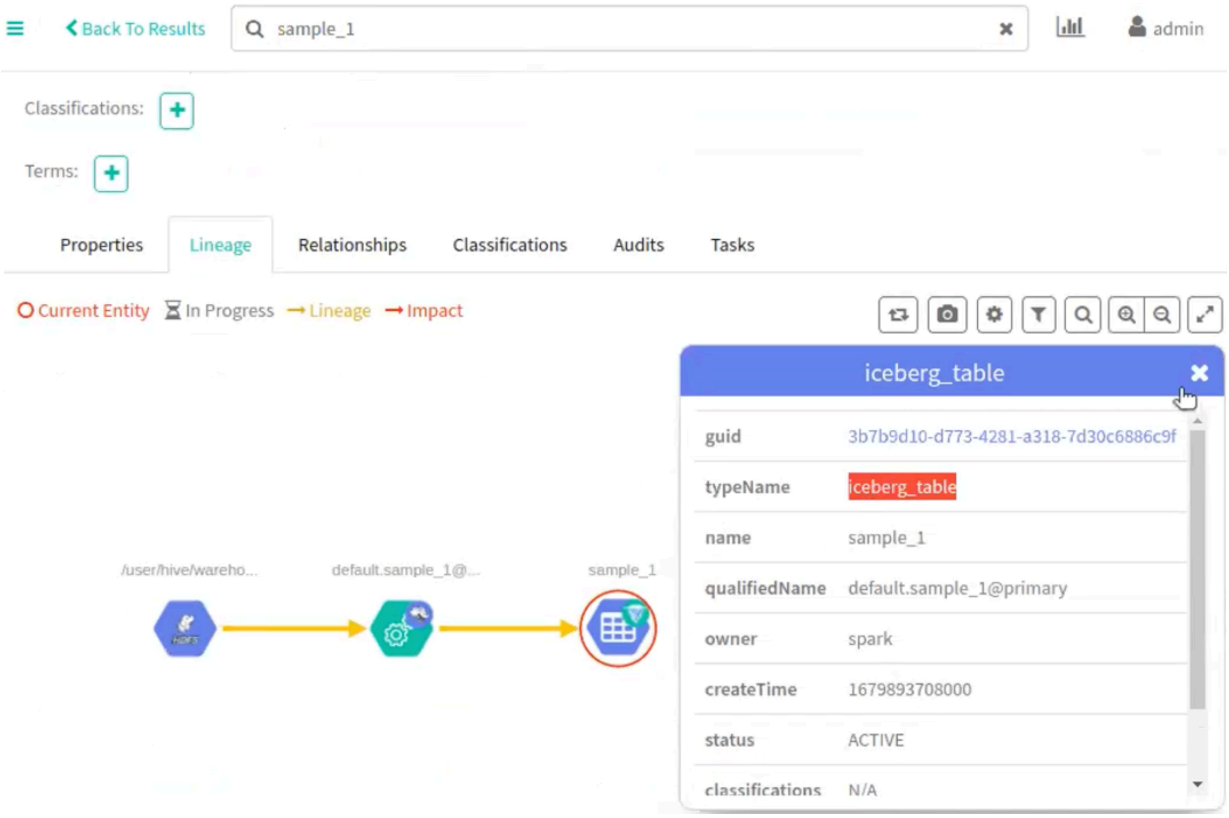

Run the following command in your Spark shell to create a Schema Evolution in a new table. For example - sample_2.

3. spark.sql("CREATE TABLE spark_catalog.default.sample_2 (id bigint COMMENT 'unique id', data string) USING iceberg");

4. Navigate accordingly in the Atlas UI to view the changes.

   The following image provide information about Iceberg schema evolution process.

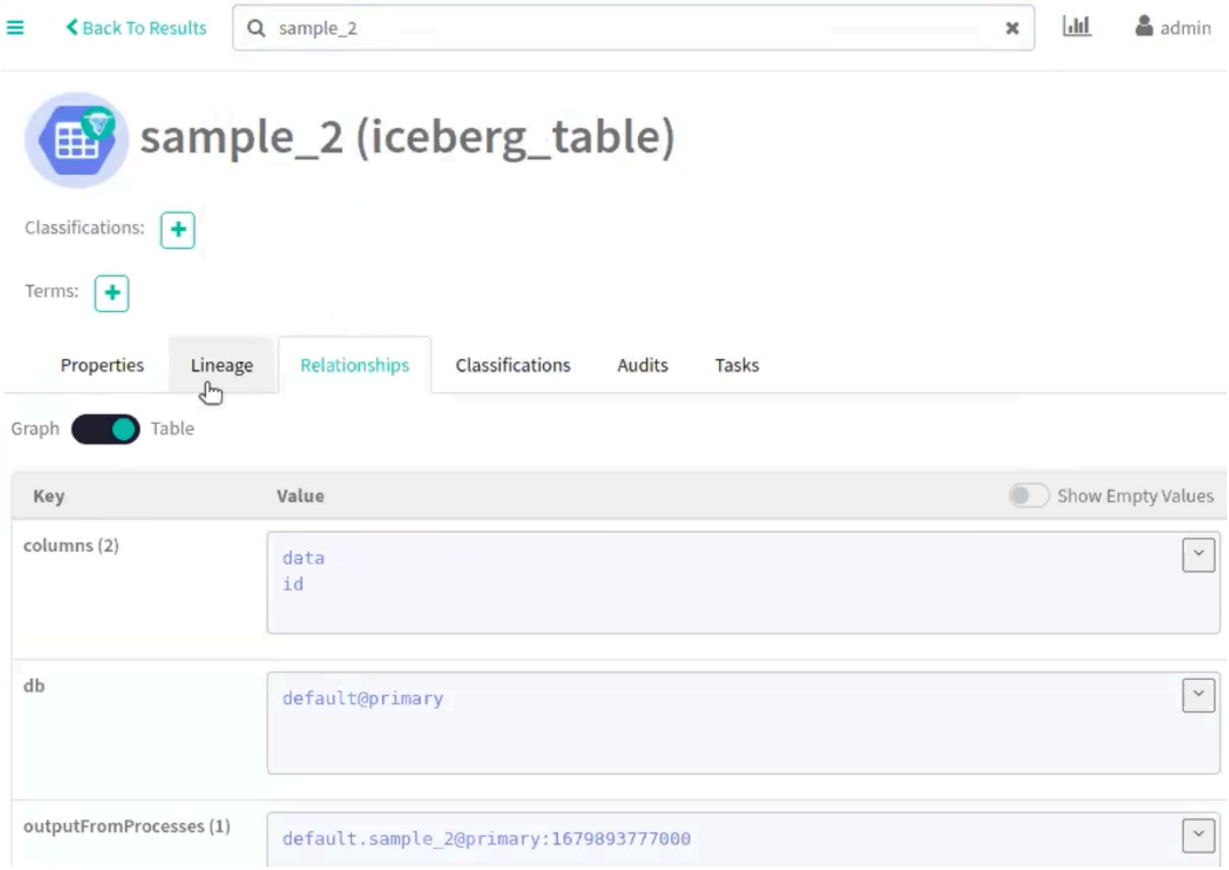

   Run the following command in your Spark shell to include a column:

5. spark.sql("ALTER TABLE spark_catalog.default.sample_2 ADD COLUMN (add_col_1 string)");

6. Navigate accordingly in the Atlas UI to view the changes.

   The following images provide information about Iceberg schema creation process.

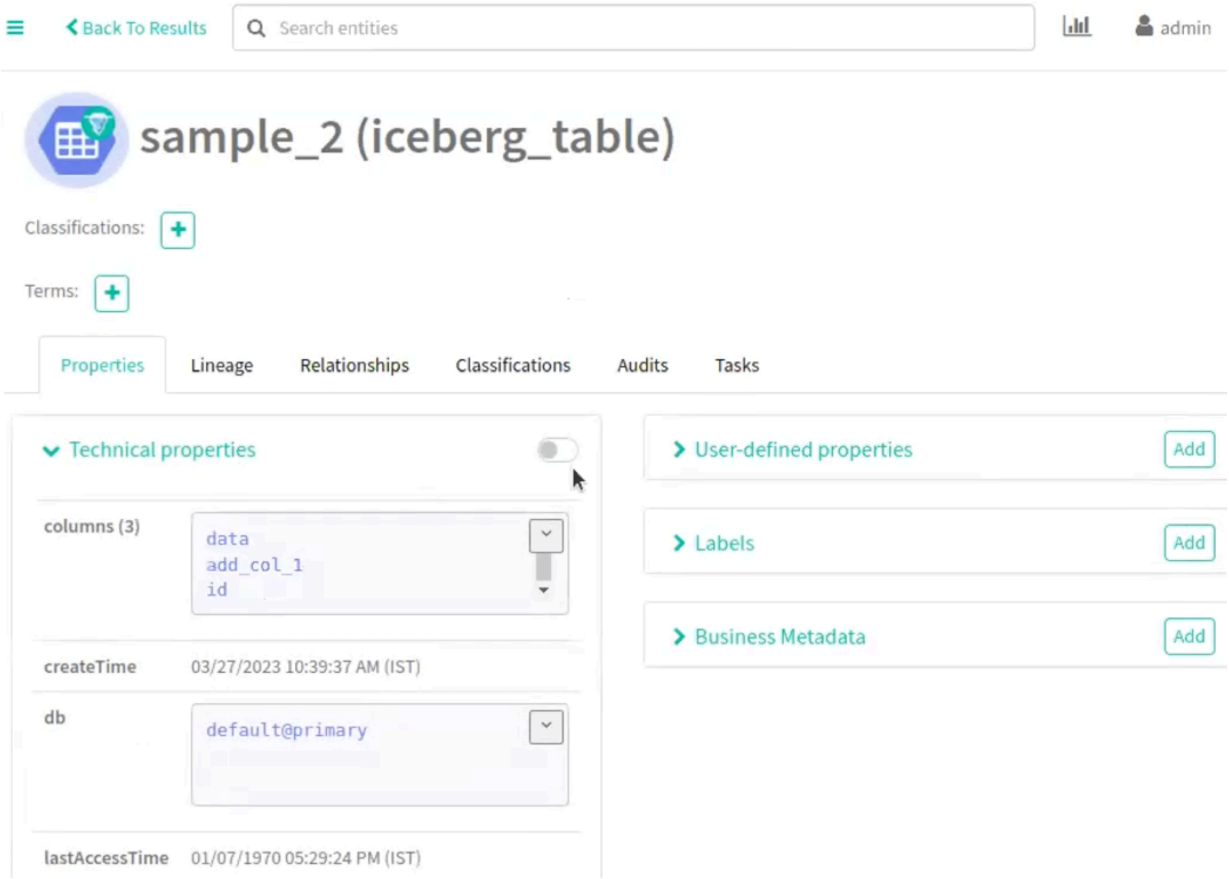

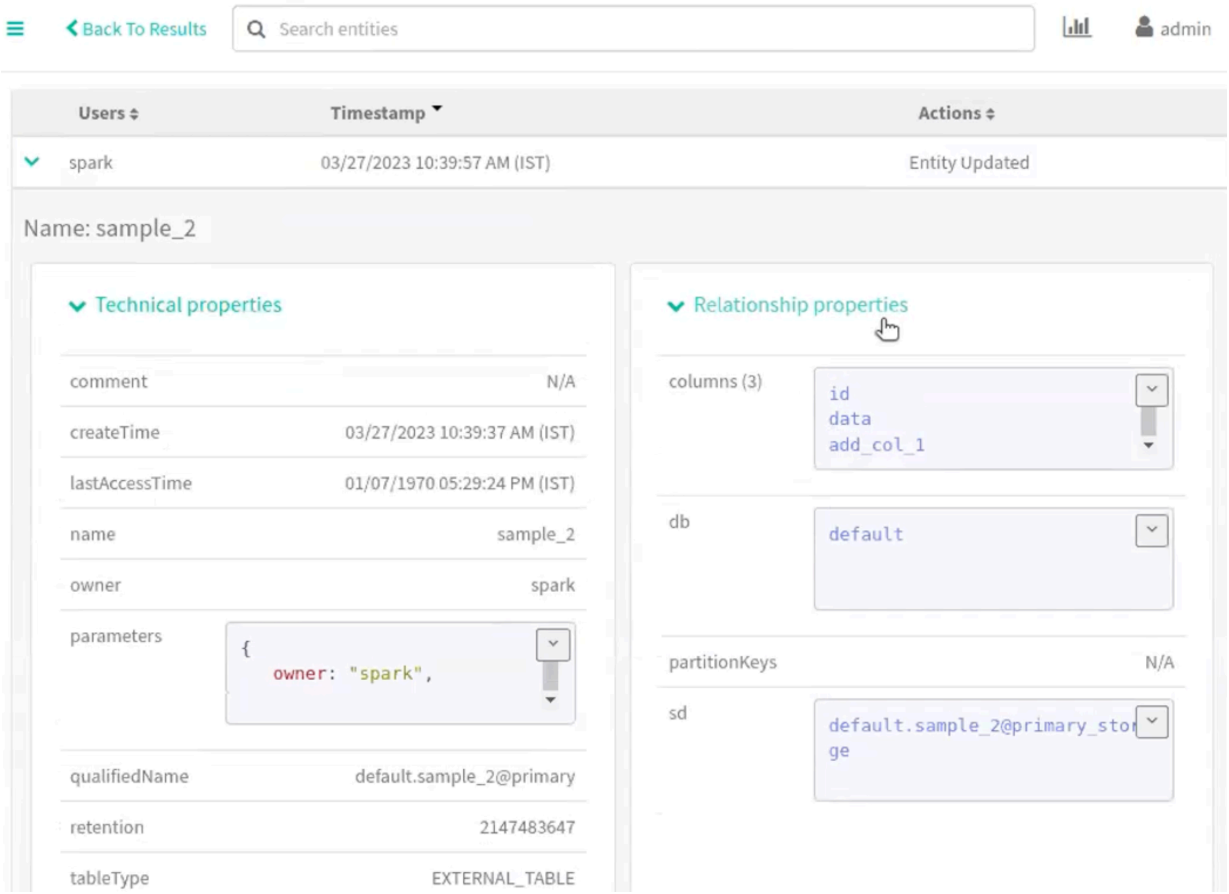

Run the following command in your Spark shell to include the second column:

7. spark.sql("ALTER TABLE spark_catalog.default.sample_2 ADD COLUMN (add_col_2 string)");

8. Navigate accordingly in the Atlas UI to view the changes.

   The following image provide information about Iceberg schema creation process.

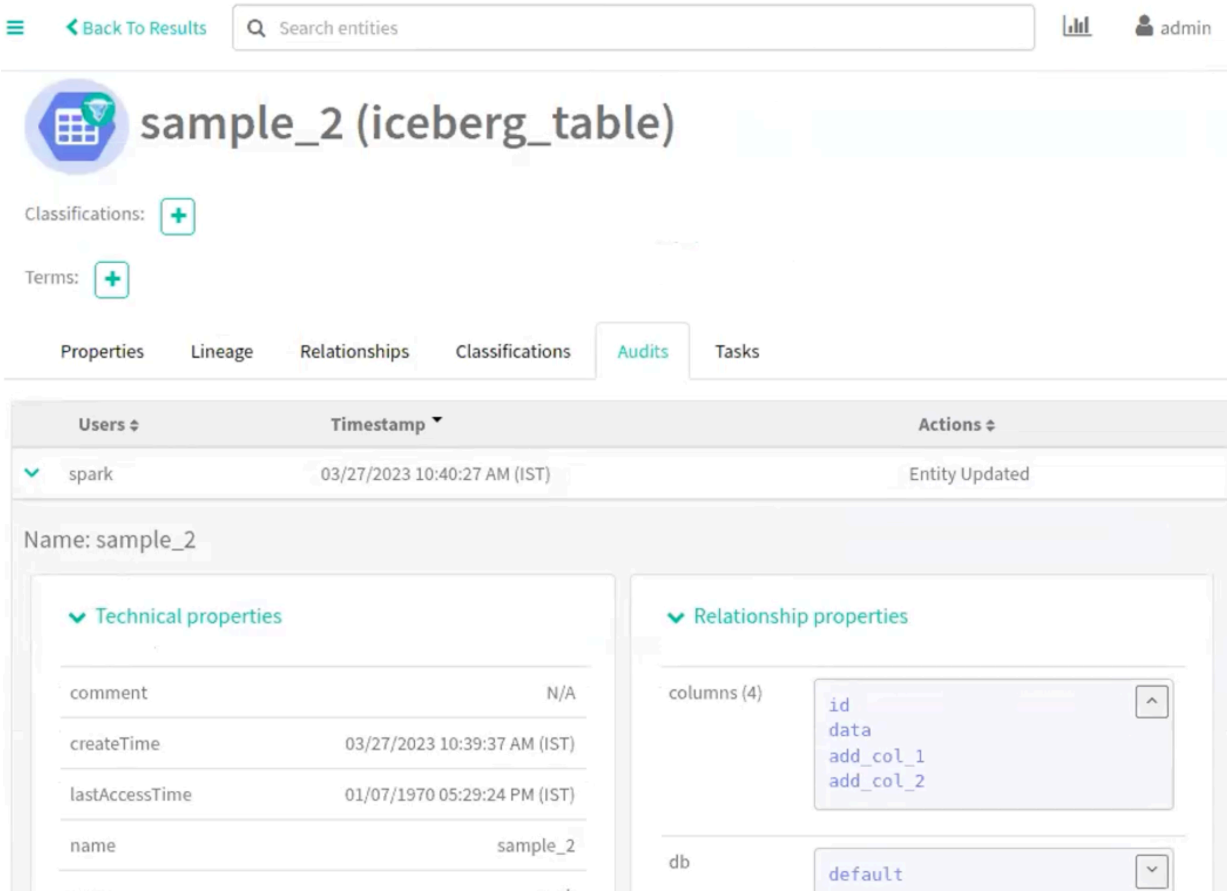

   Run the following command in your Spark shell to create a Partition Specification in a new table (sample_3):

9. spark.sql("CREATE TABLE spark_catalog.default.sample_3 (id bigint,data string,category string,ts timestamp) USING iceberg PARTITIONED BY (bucket(16, id), days(ts), category)");

10. Navigate accordingly in the Atlas UI to view the changes.

The following images provide information about Iceberg partition specification process.

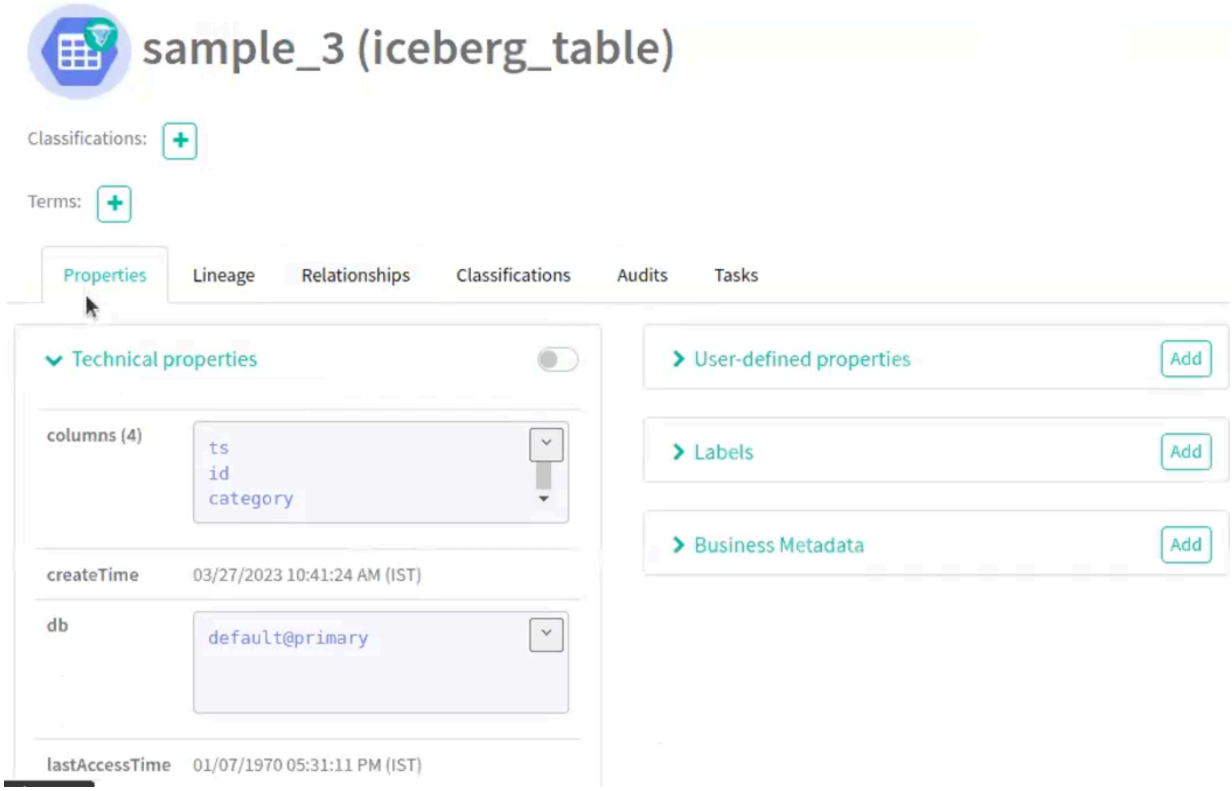

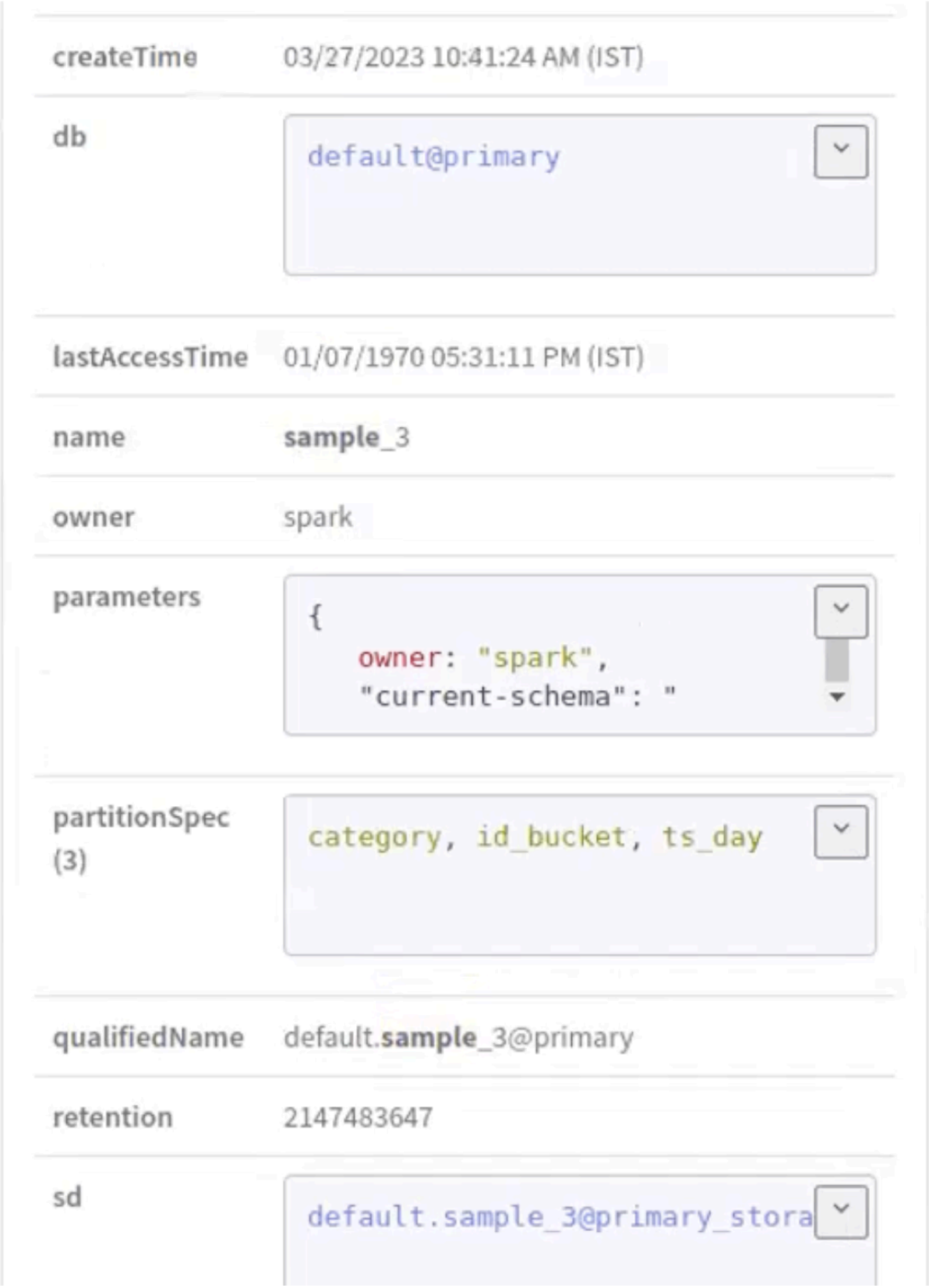

Run the following command in your Spark shell to create a Partition Evolution in a new table (sample_3):

11. spark.sql("ALTER TABLE spark_catalog.default.sample_3 ADD PARTITION FIELD years(ts)");

12. Navigate accordingly in the Atlas UI to view the changes.

The following images provide information about Iceberg partition evolution process.

Displaying Snapshot attributes

Run the following command to display relevant snapshot table parameters as attributes in Atlas. For example: Existing snapshot ID, current snapshot timestamp, snapshot count, number of files and related attributes.

```
spark.sql("CREATE TABLE spark_catalog.default.sample3 ( id int, data string) USING iceberg");
spark.sql("INSERT INTO default.sample3 VALUES (1, 'TEST')");
```

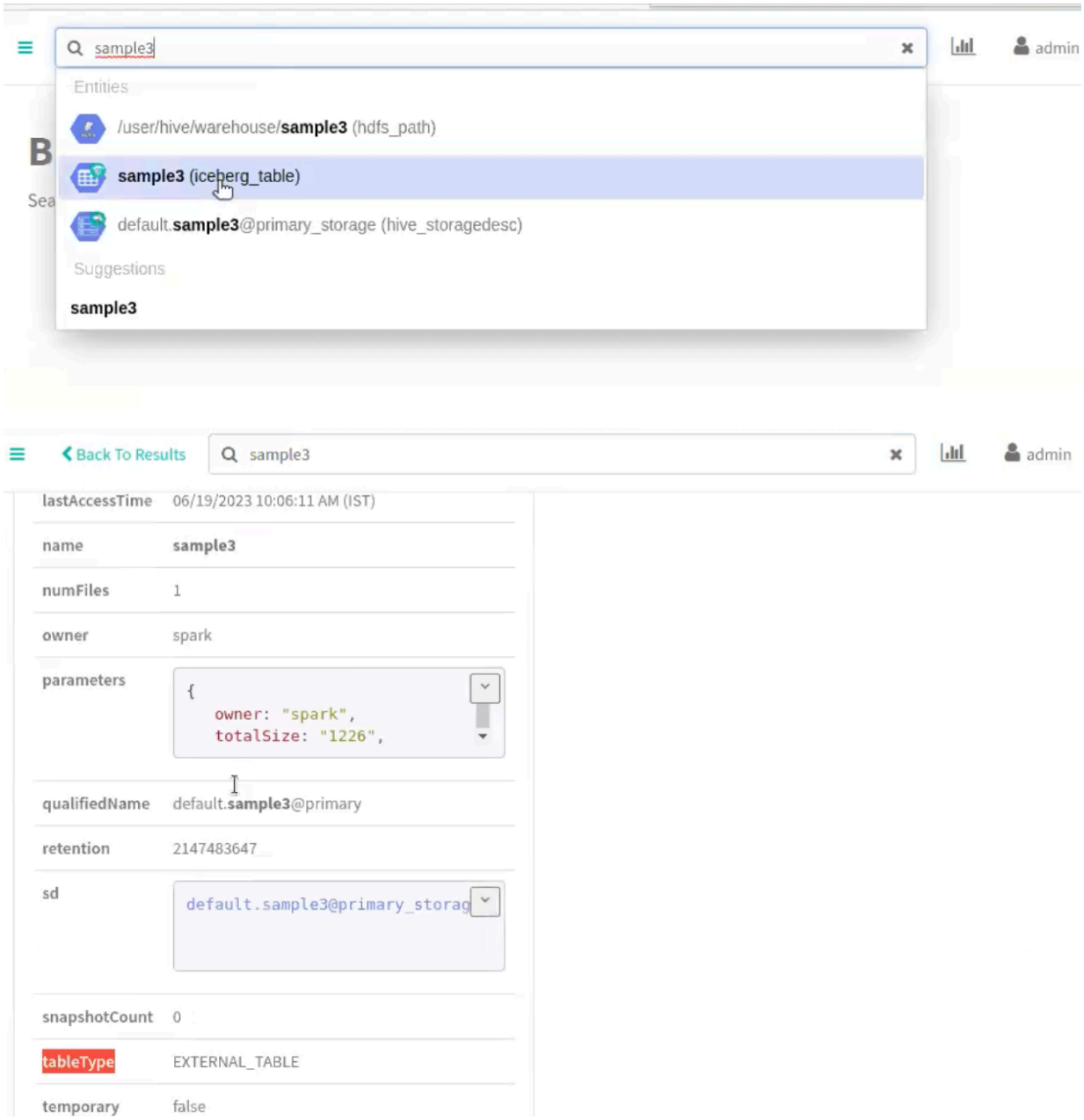

Run the following command to scale up the snapshot count value.

```
spark.sql("INSERT INTO default.sample3 VALUES (1, 'TEST')");
```

The latest Snapshot ID that Iceberg points to along with the Snapshot timestamp and Snapshot count are updated respectively.

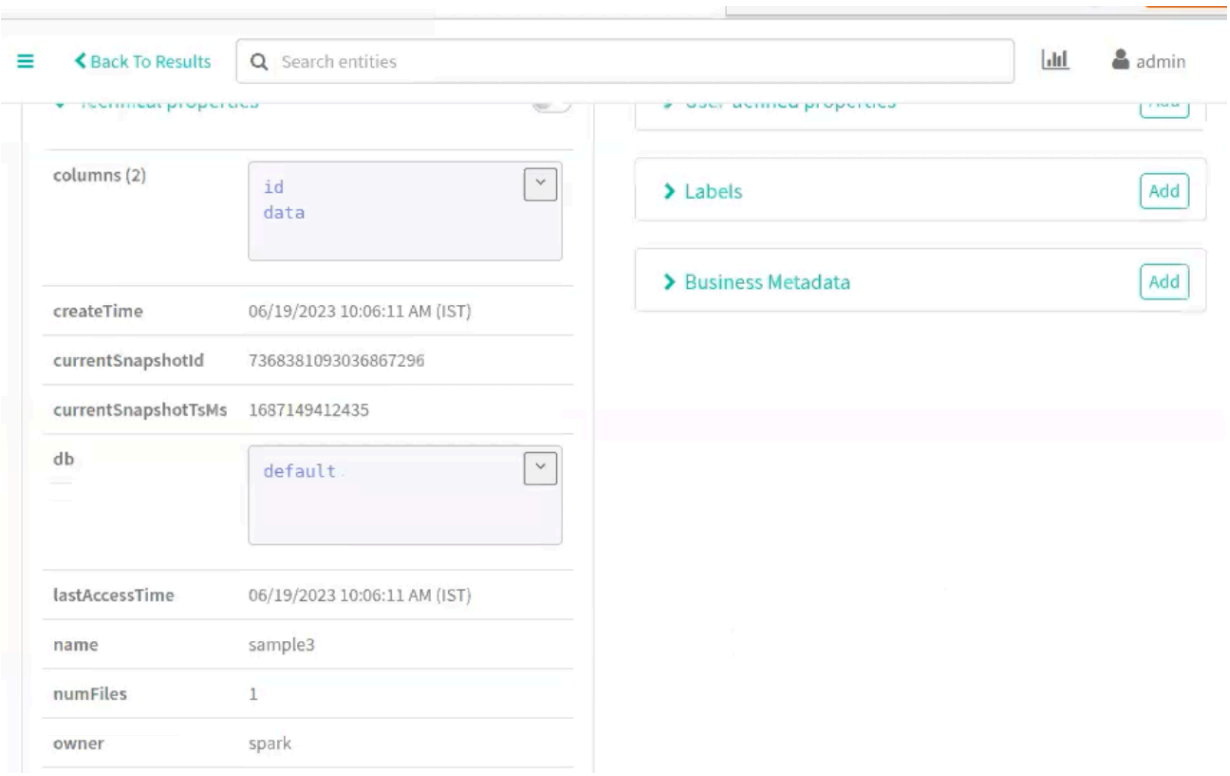

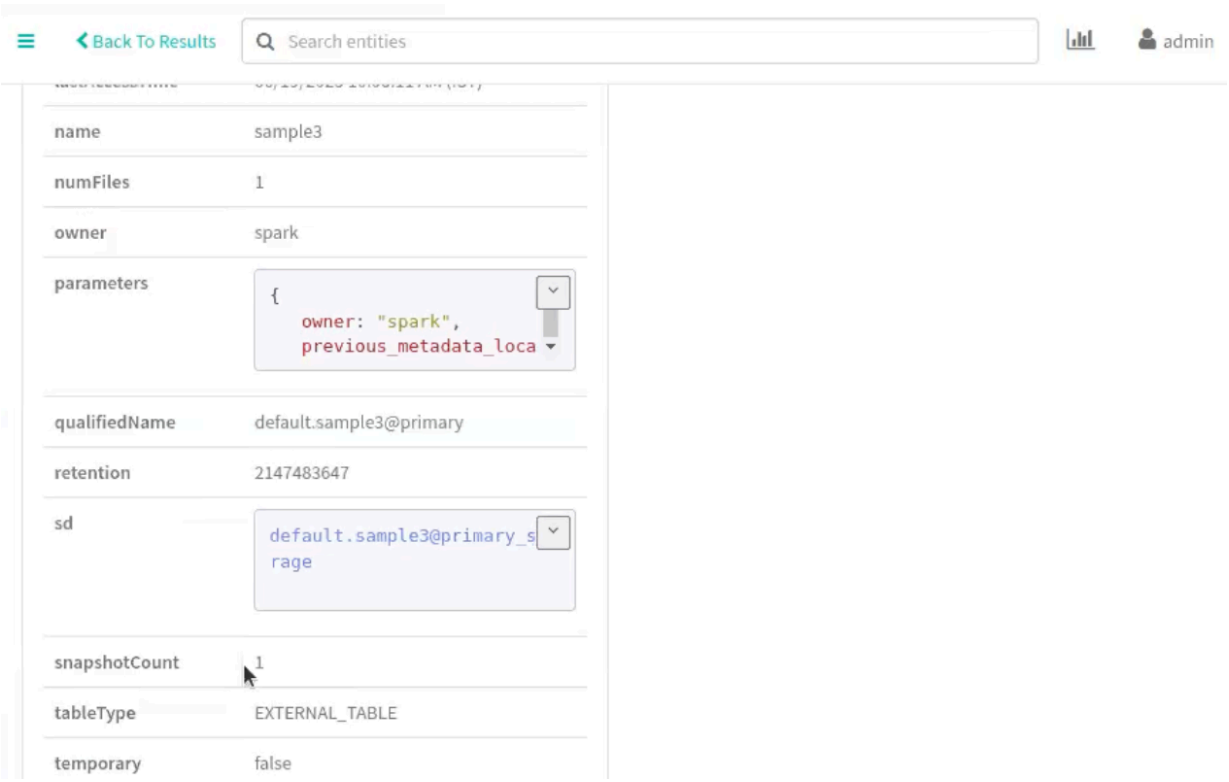

Click on the parameters field to display the details pertaining to the snapshot attributes.

Support for data compaction

Data Compaction is the process of taking several small files and rewriting them into fewer larger files to speed up queries. When performing compaction on an Iceberg table, execute the rewriteDataFiles procedure, optionally specifying a filter of which files to rewrite and the desired size of the resulting files.

As an example, in an Atlas instance consider that the number of files and snapshot count are 9.

Run the following command to perform data compaction

```
spark.sql("CALL spark_catalog.system.rewrite_data_files('default.sample3')"),show();
```

```
+-------------------------+----------------------+
|rewritten_data_files_count|added_data_files_count|
+-------------------------+----------------------+
|                        9|                     1|
+-------------------------+----------------------+
```

The count for the number of rewritten data files is compacted from a total count of 9 to 1.

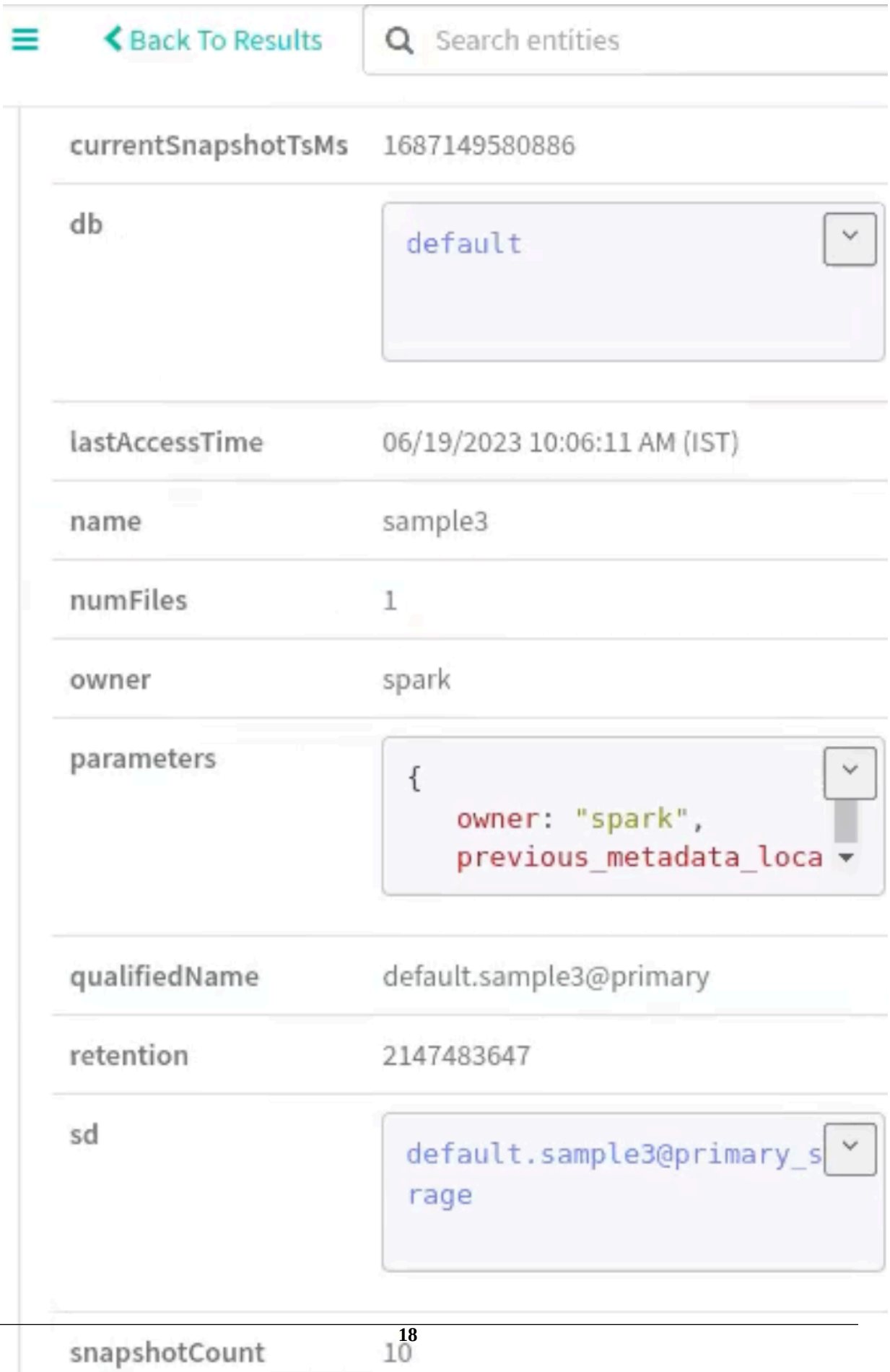
Back To Results
Search entities
currentSnapshotTsMs
1687149580886
db
default
lastAccessTime
06/19/2023 10:06:11 AM (IST)
name
sample3
numFiles
1
owner
spark
parameters
{
owner: "spark",
previous_metadata_loca
qualifiedName
default.sample3@primary
retention
2147483647
sd
default.sample3@primary_s
rage
snapshotCount
10

Support for metadata rewrite attributes

Iceberg uses metadata in its manifest list and manifest files speed up query planning and to prune unnecessary data files. You can rewrite manifests for a table to optimize the plan to scan the data.

Run the following command to rewrite manifests on a sample table 3.

spark.sql("CALL spark_catalog.system.rewrite_manifests('default.sample3')").show();

```
+-------------------------+---------------------+
|rewritten_manifests_count|added_manifests_count|
+-------------------------+---------------------+
|                       10|                    1|
+-------------------------+---------------------+
```

The count for table manifested data is rewritten from 10 to 1.

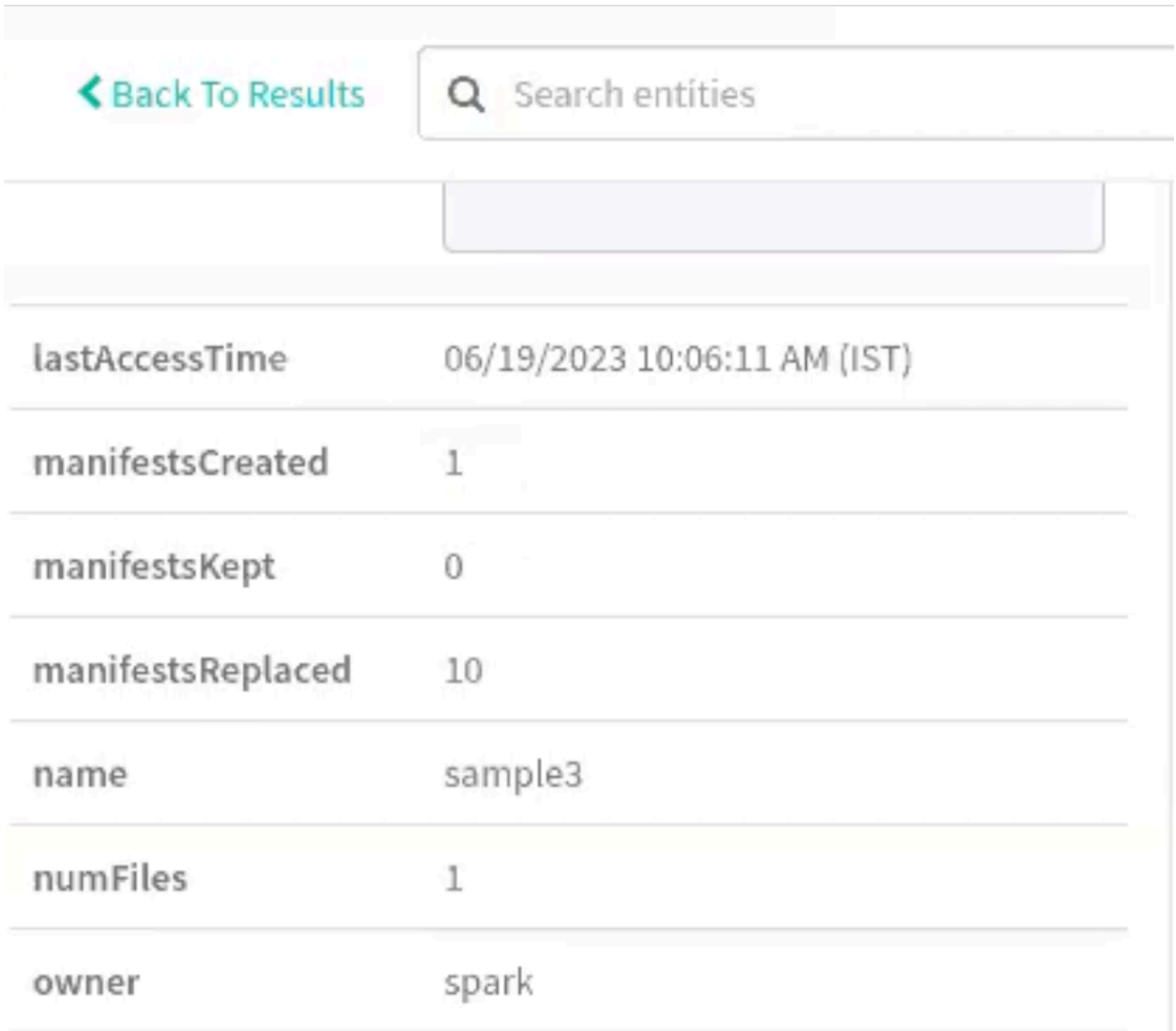

The process is completed.

Related Information

Using the Hive shell

Using the Impala shell

Using the Hive shell

Using Hive, you can create an Iceberg table followed by using the CTAS command to alter or copy the existing Hive table and its properties into the Iceberg table.

Before you begin

In this case, you create an external table and alter an existing Hive table to Iceberg table using the Hive engine.

Run the following command in your Hive shell to create an Iceberg table.

Procedure

1. create external table if not exists hive_ice_1 (CountryID int, CountryName string, Capital string, Population string) STORED BY ICEBERG STORED AS PARQUET;

2. Navigate accordingly in the Atlas UI to view the changes.
   The following images provide information about Iceberg table creation process.

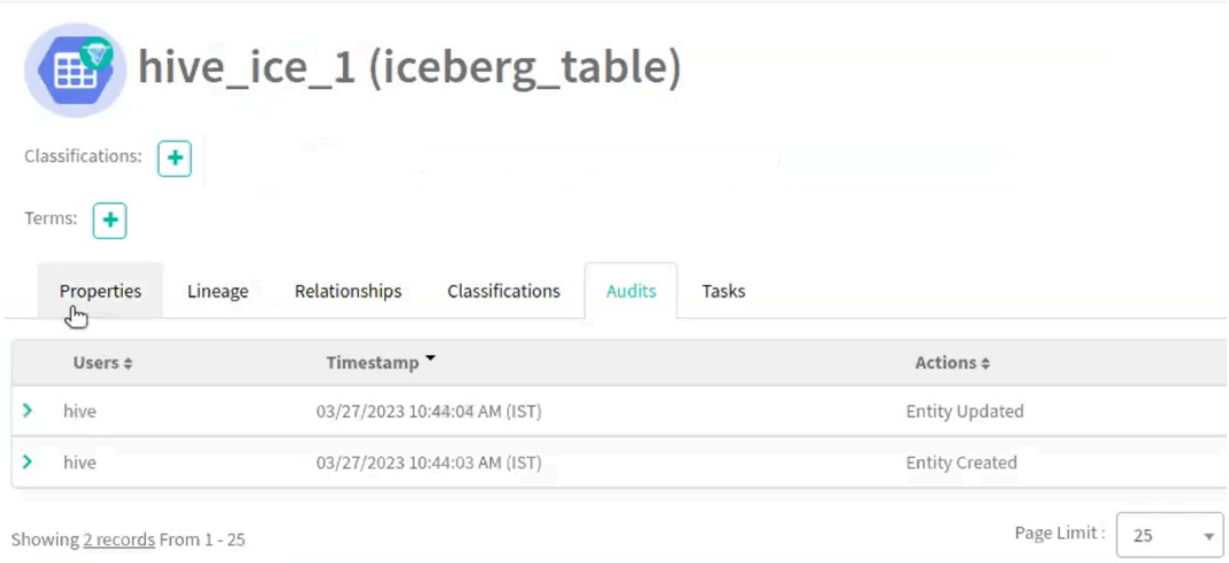

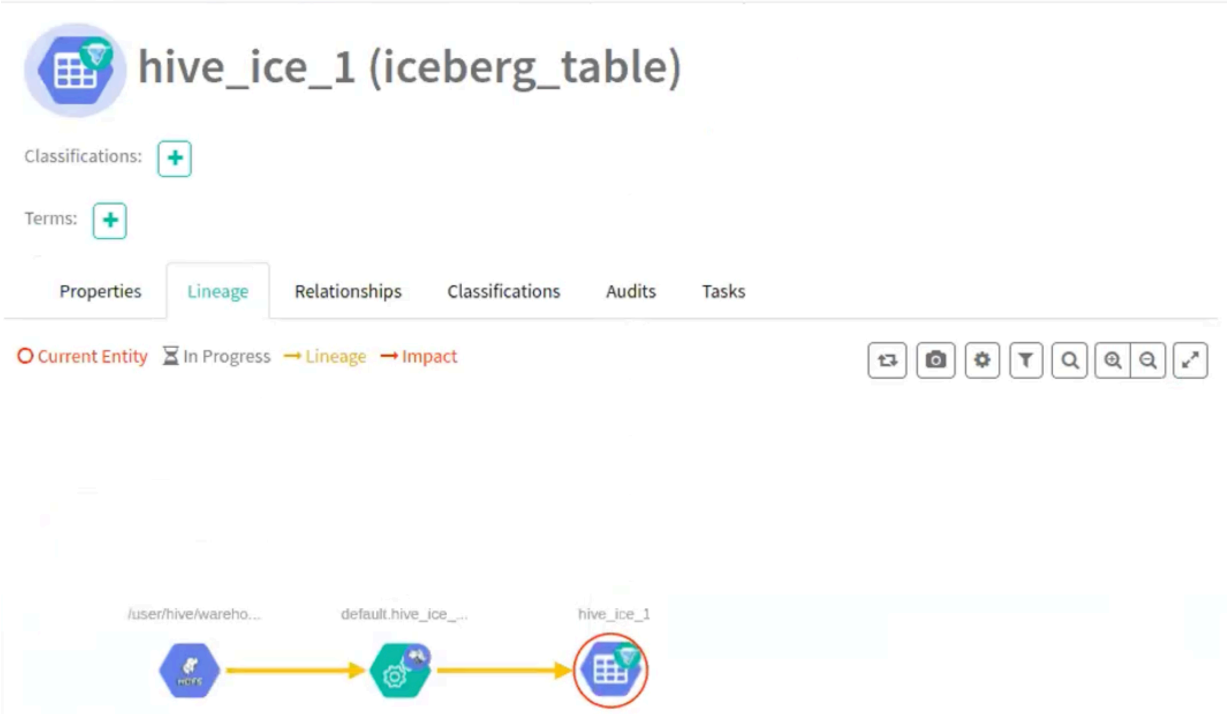

Run the following commands in your Hive shell to copy the contents of one table (hive_ice_3) to another newly created table (hive_ice_4).

3. create external table if not exists hive_ice_3 (CountryID int, CountryName string, Capital string, Population string) STORED BY ICEBERG STORED AS PARQUET;

4. create external table if not exists hive_ice_4 STORED BY ICEBERG STORED AS PARQUET as select * from hive_ice_3;

   The following images provide information about copying contents from one table to another.

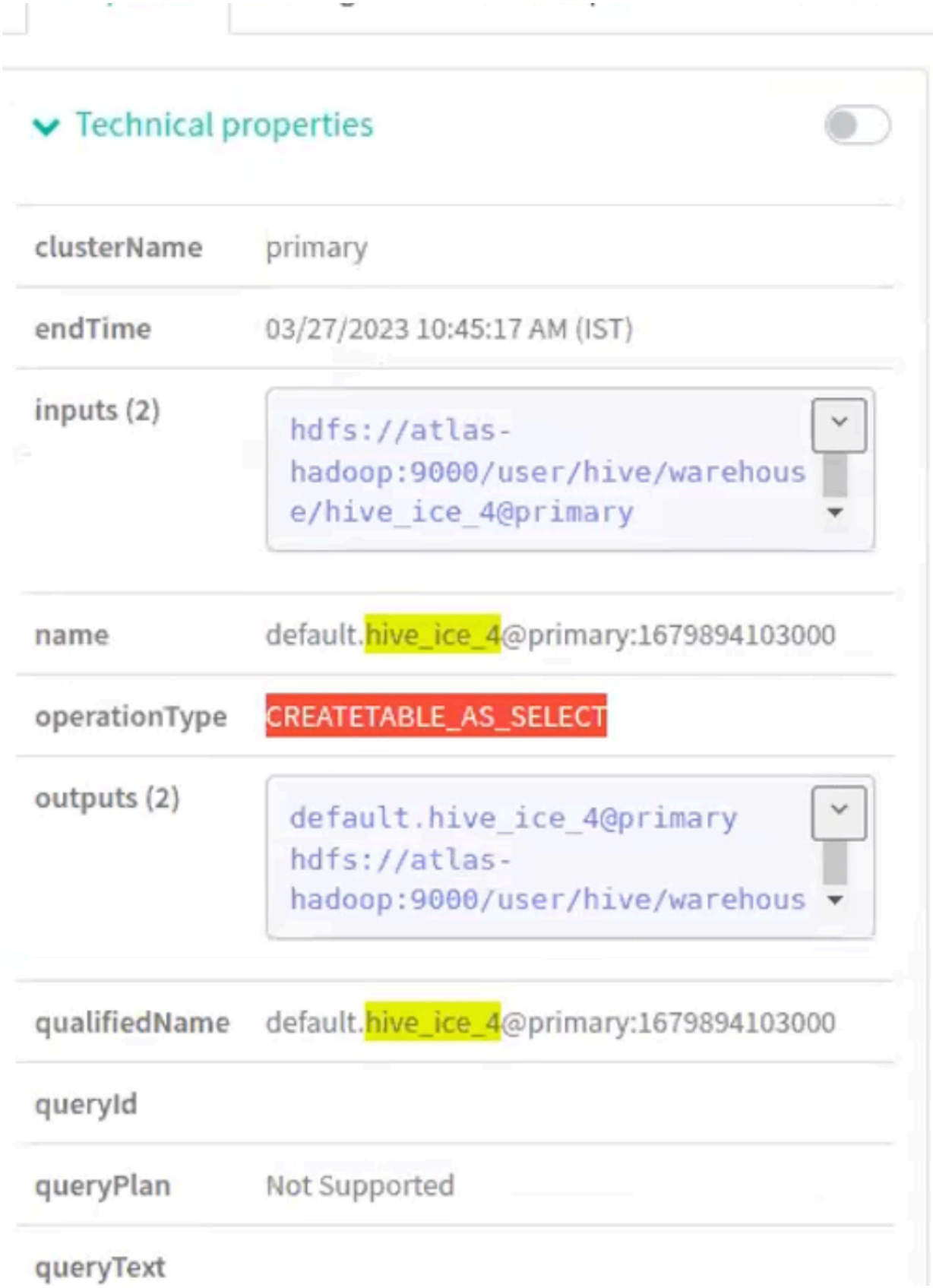

You can alter an existing Hive table to Iceberg table.

5. create external table if not exists hive_ice_5 (CountryID int, CountryName string, Capital string, Population st ring)STORED AS PARQUET;

6. ALTER TABLE hive_ice_5 SET TBLPROPERTIES ('storage_handler'='org.apache.iceberg.mr.hive.HiveIcebergStorageHandler');

   The following images provide information about alter tables operations.

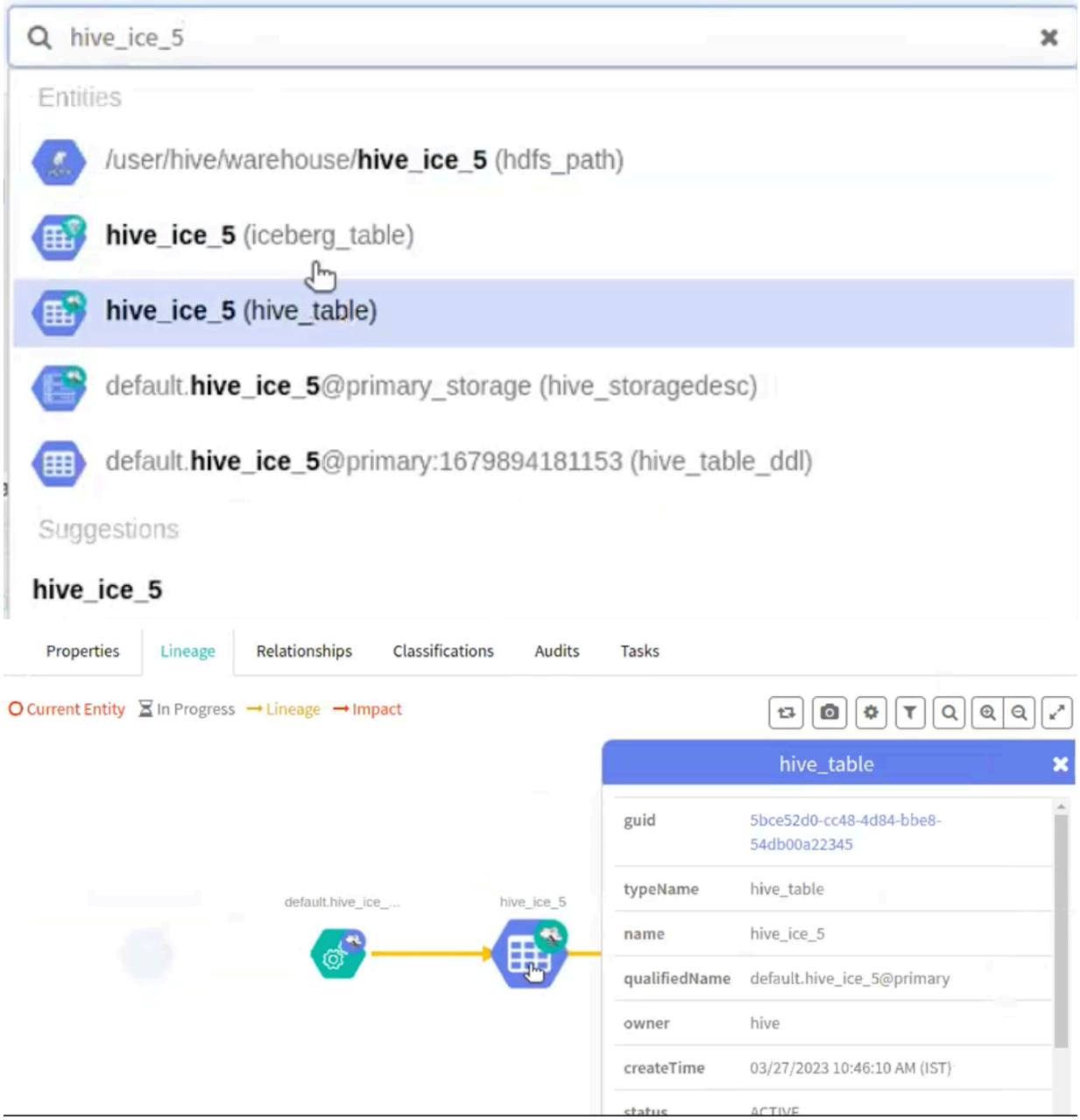

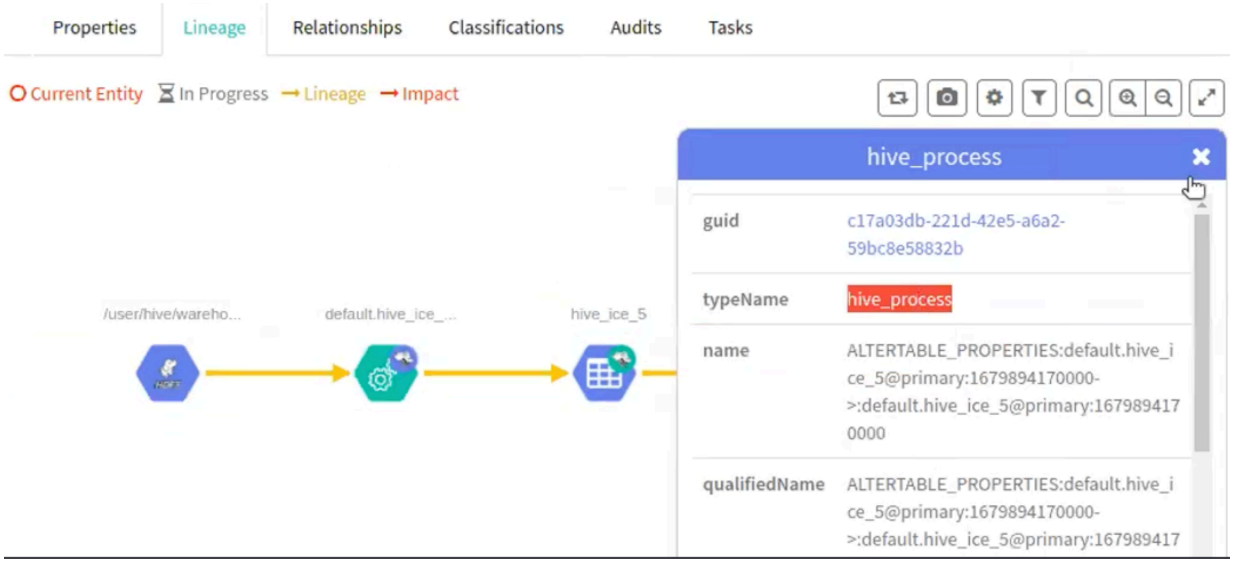

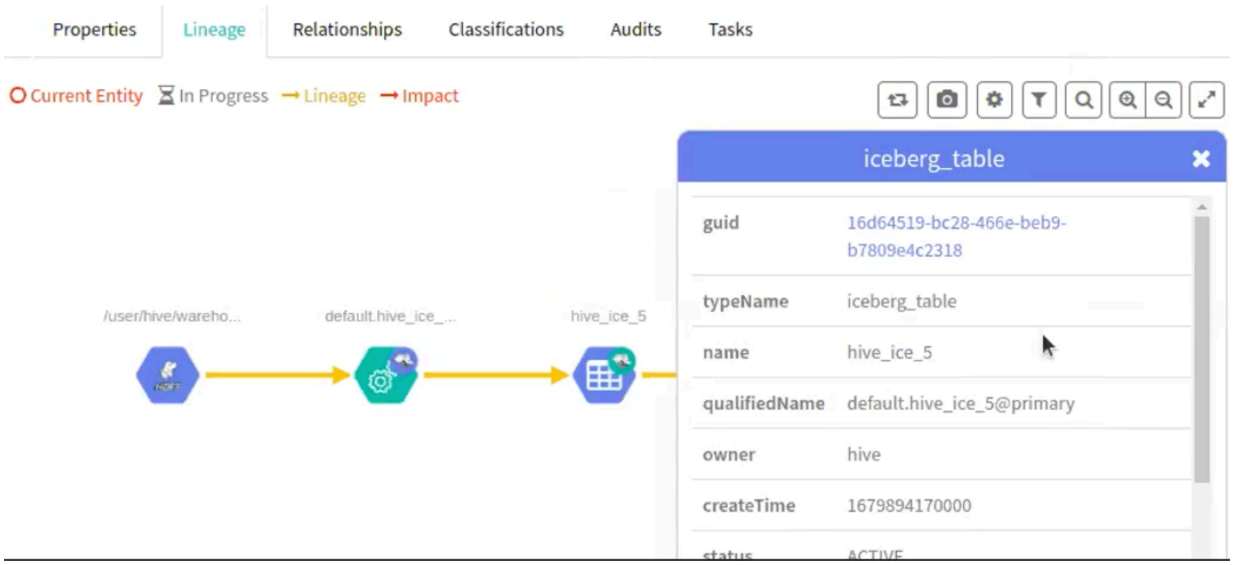

Related Information

Using the Spark shell

Using the Impala shell

Using the Impala shell

Using Impala, you can create an Iceberg table followed by Schema evolution, partition specification, partition evolution and CTAS operation.

Before you begin

Run the following command in your Impala shell to create a new Iceberg table

Procedure

1. CREATE TABLE ice_t (i INT) STORED AS ICEBERG;

2. Navigate accordingly in the Atlas UI to view the changes.
   The following images provide information about Iceberg table creation process.

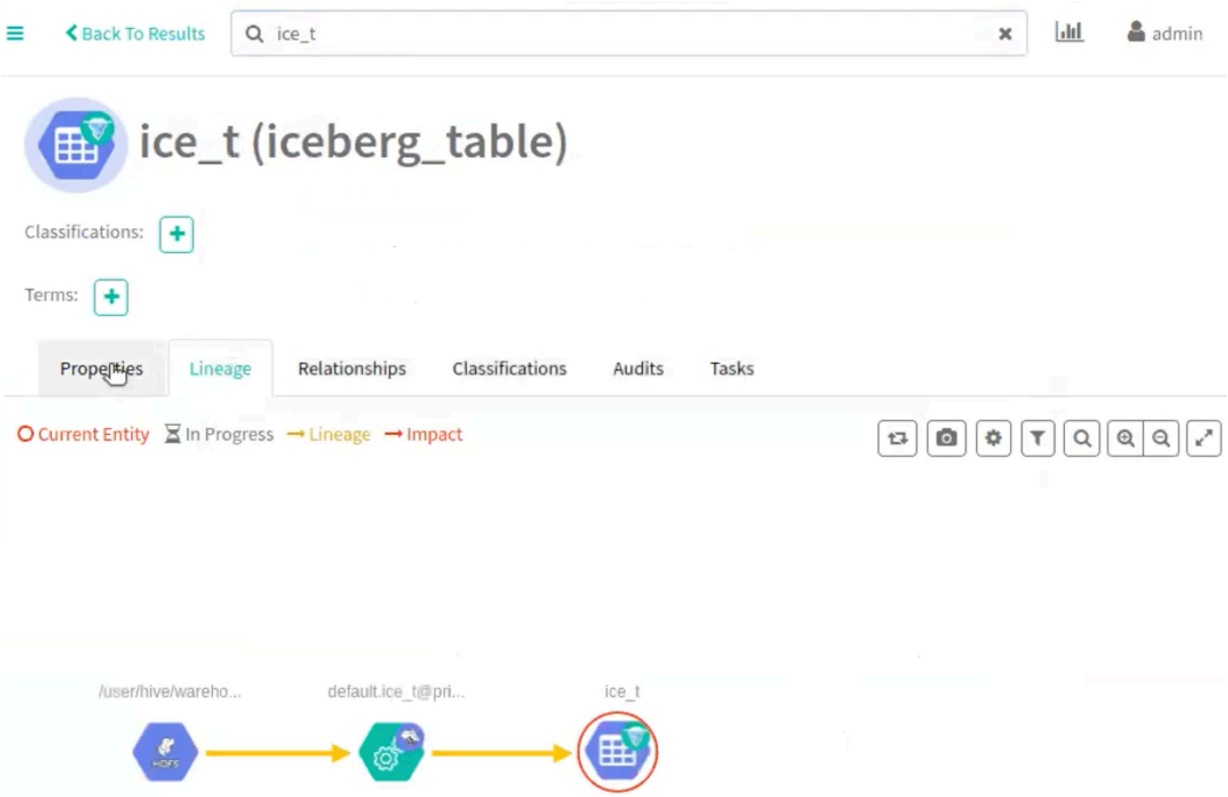

Run the following command in your Impala shell to create a scheme evolution:

3. CREATE TABLE ice_t_2 (i INT) STORED AS ICEBERG;

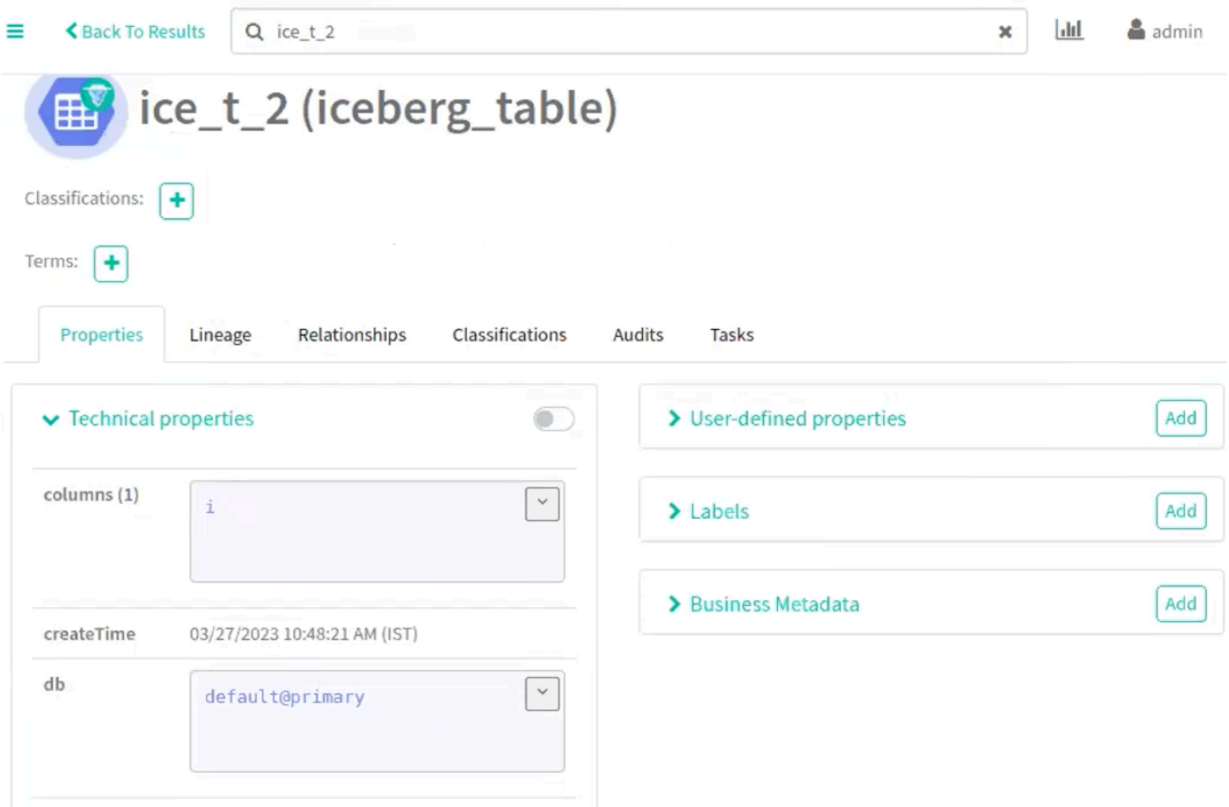

Run the following command in your Impala shell to add a column to the existing table (ice_t_2):

4. alter table ice_t_2 ADD COLUMNS (add_col_1 string);

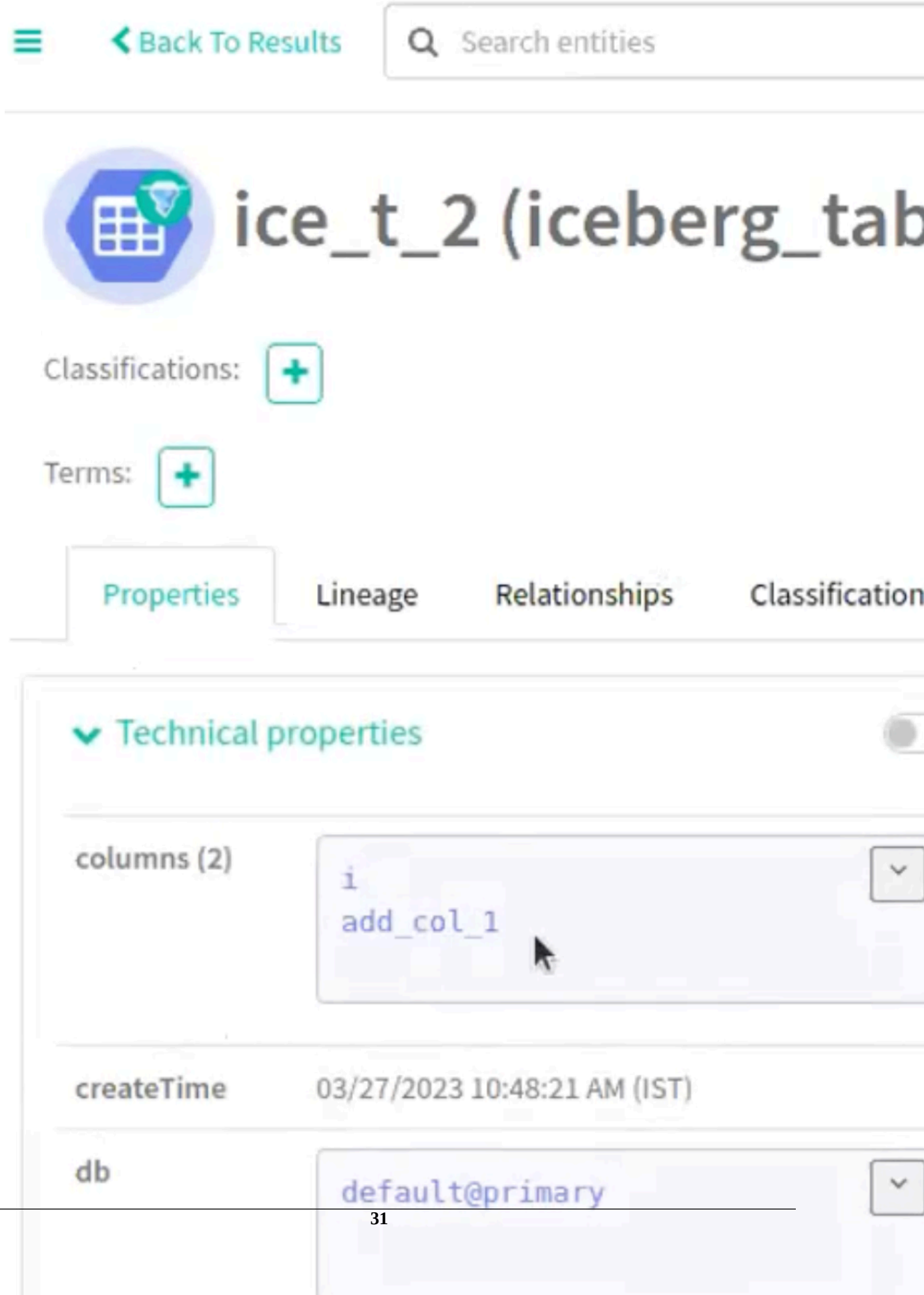
Back To Results
Search entities
ice_t_2 (iceberg_tab
Classifications:
Terms:
Properties
Lineage
Relationships
Classification
Technical properties
columns (2)
i
add_col_1
createTime
03/27/2023 10:48:21 AM (IST)
db
default@primary

Run the following command in your Impala shell to create a partition specification.

5. CREATE TABLE ice_part_spec (s string , b string) PARTITIONED BY SPEC (truncate(3, s)) STORED AS ICEBERG ;

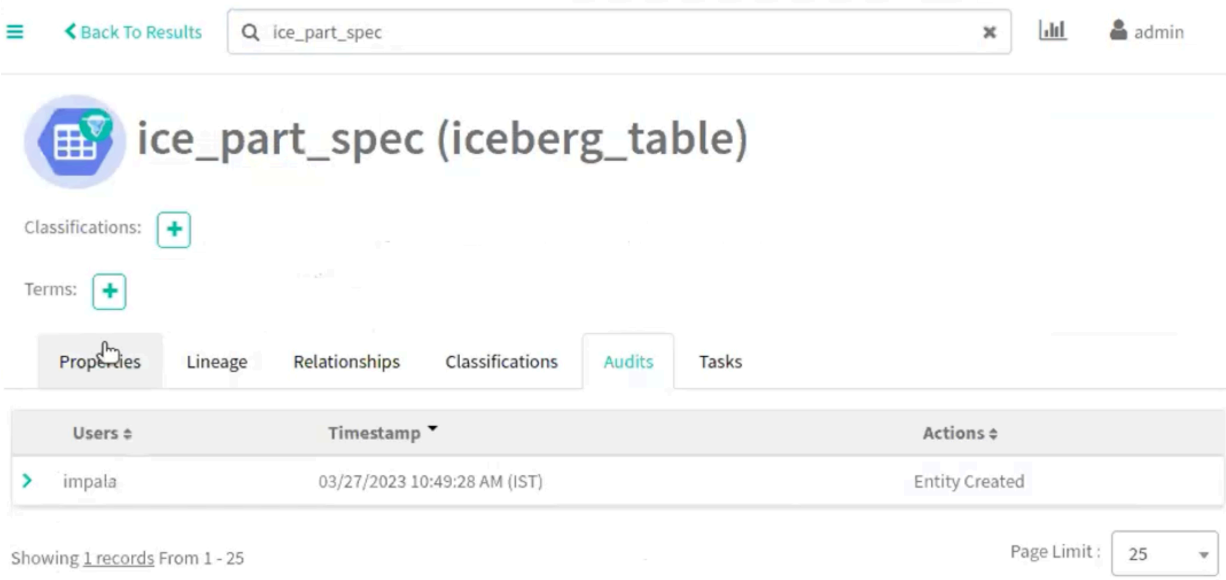

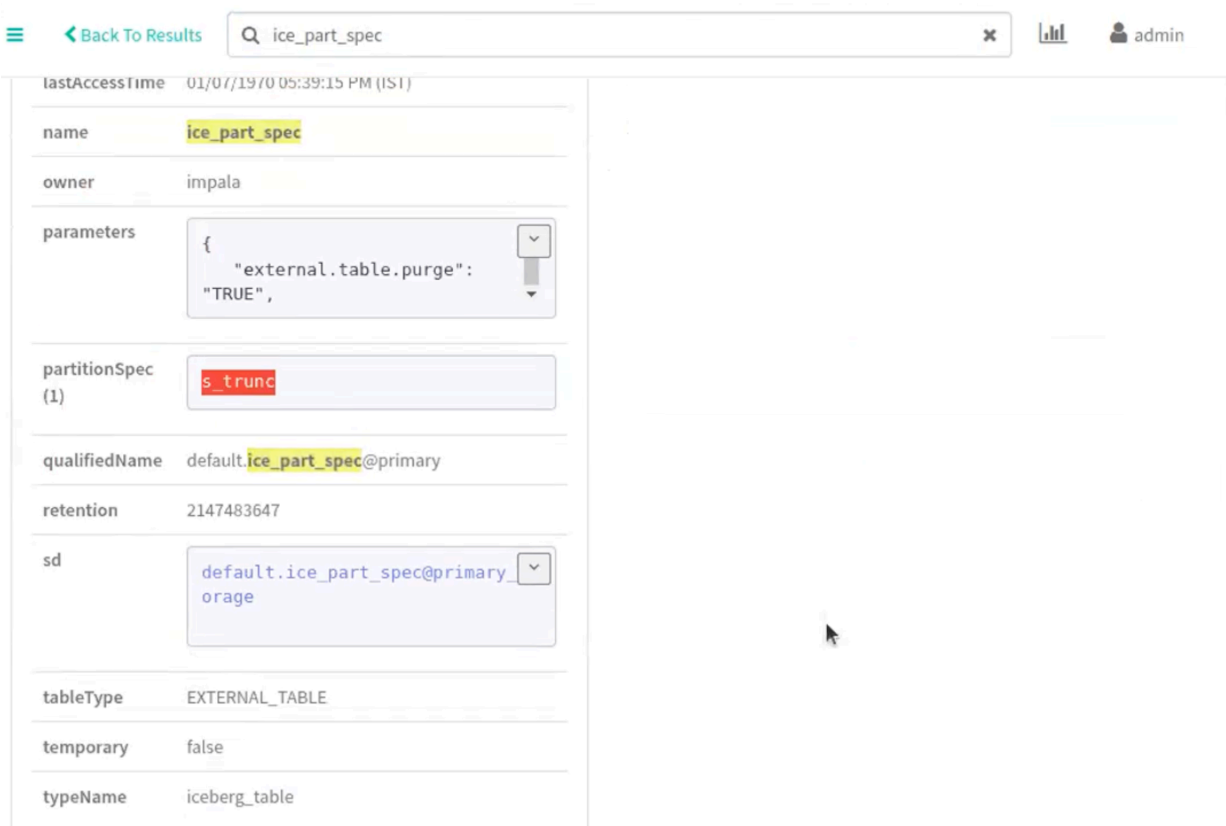

Run the following command in your Impala shell to create a partition evolution.

6. CREATE TABLE ice_p (i INT, d DATE, s STRING, t TIMESTAMP) PARTITIONED BY SPEC (BUCKET(5, i), MONTH(d), TRUNCATE(3, s), HOUR(t))STORED AS ICEBERG;

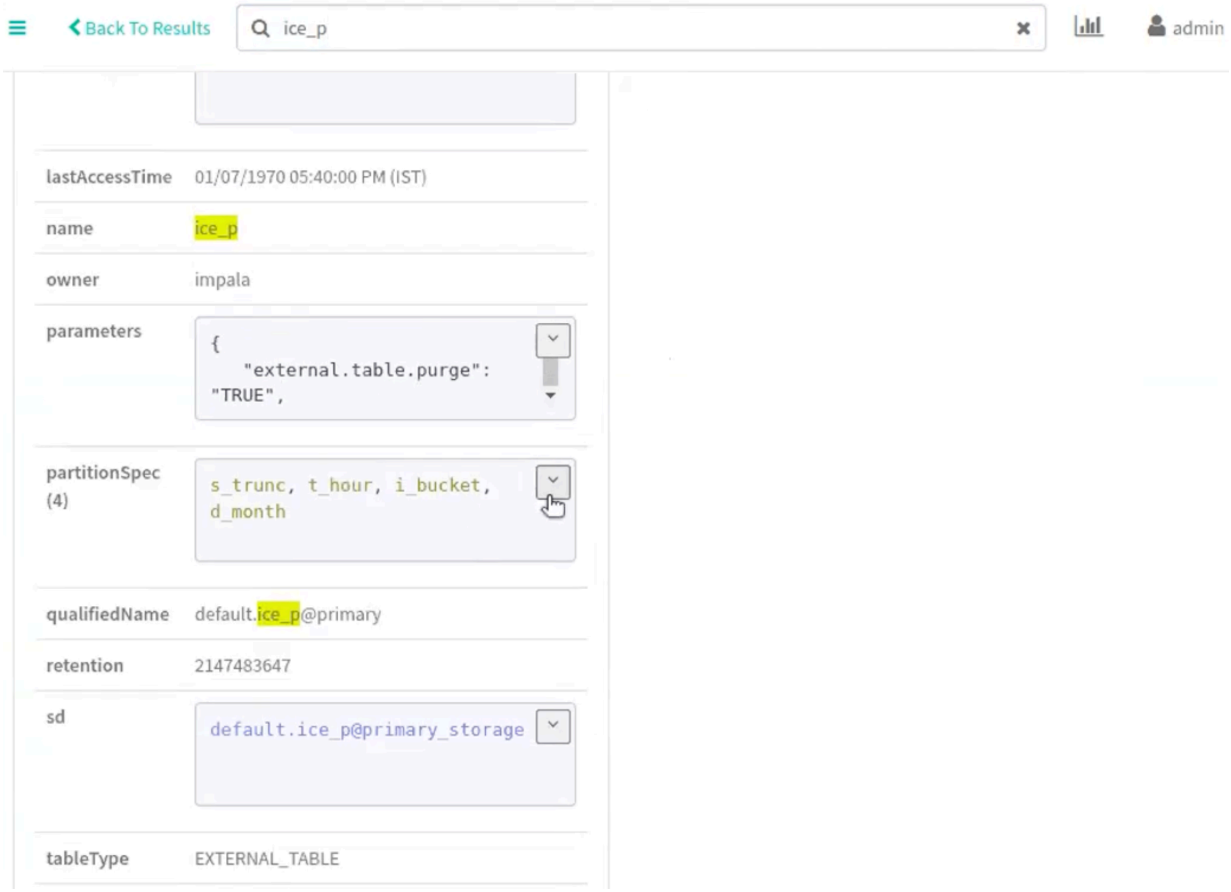

Run the following command in your Impala shell to modify the partition specification

7. ALTER TABLE ice_p SET PARTITION SPEC (VOID(i), VOID(d), TRUNCATE(3, s), HOUR(t), i);

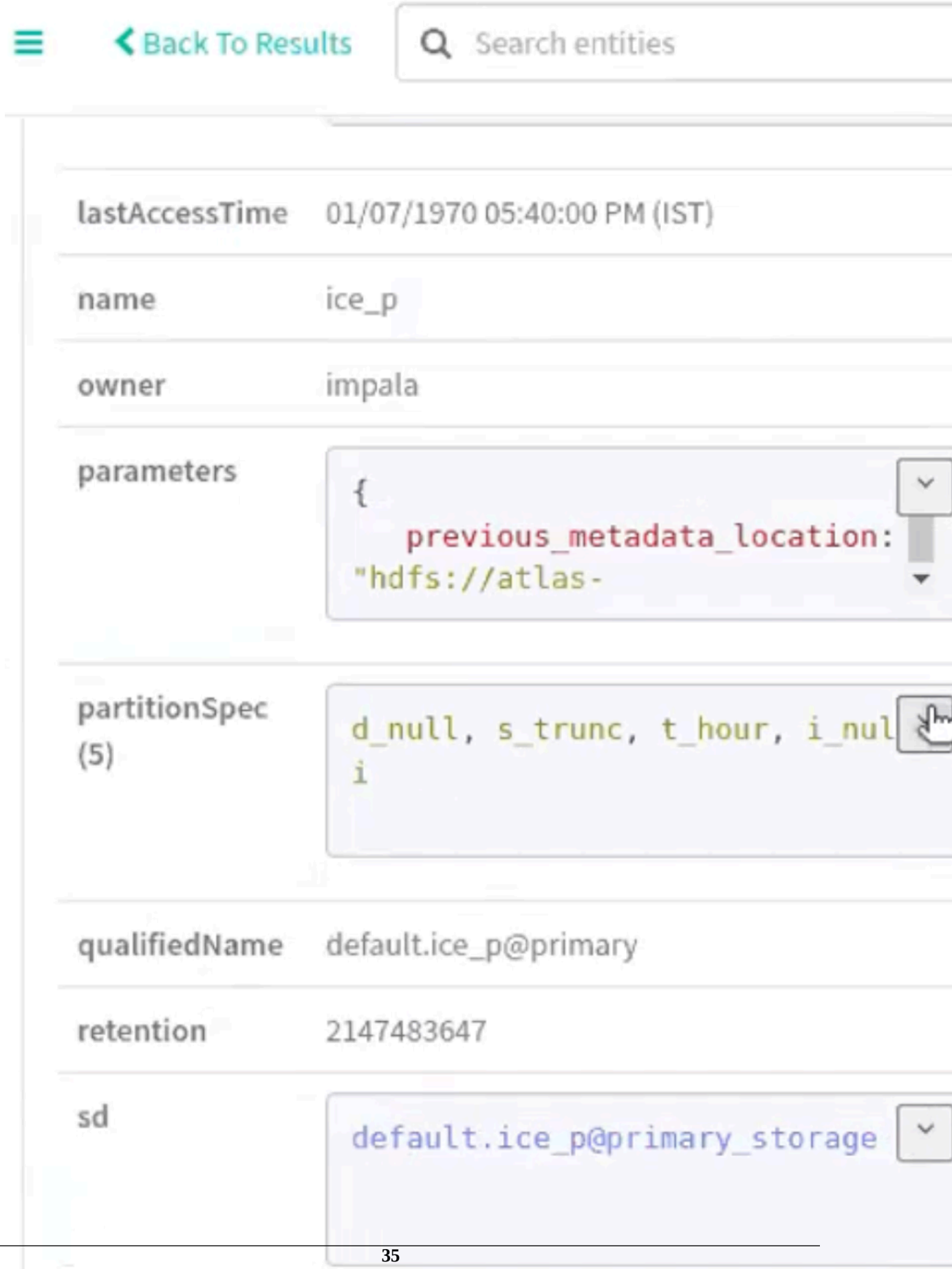
Back To Results
Search entities
lastAccessTime
01/07/1970 05:40:00 PM (IST)
name
ice_p
owner
impala
parameters
{
previous_metadata_location:
"hdfs://atlas-
partitionSpec
(5)
d_null, s_trunc, t_hour, i_nul
i
qualifiedName
default.ice_p@primary
retention
2147483647
sd
default.ice_p@primary_storage

Run the following commands in your Impala shell to create the contents of one table (ice_t_3) to another table (ice_t_4).

8. CREATE TABLE ice_t_3 (i INT) STORED AS ICEBERG;

9. CREATE TABLE ice_t_4 STORED AS ICEBERG as select * from ice_t_3;

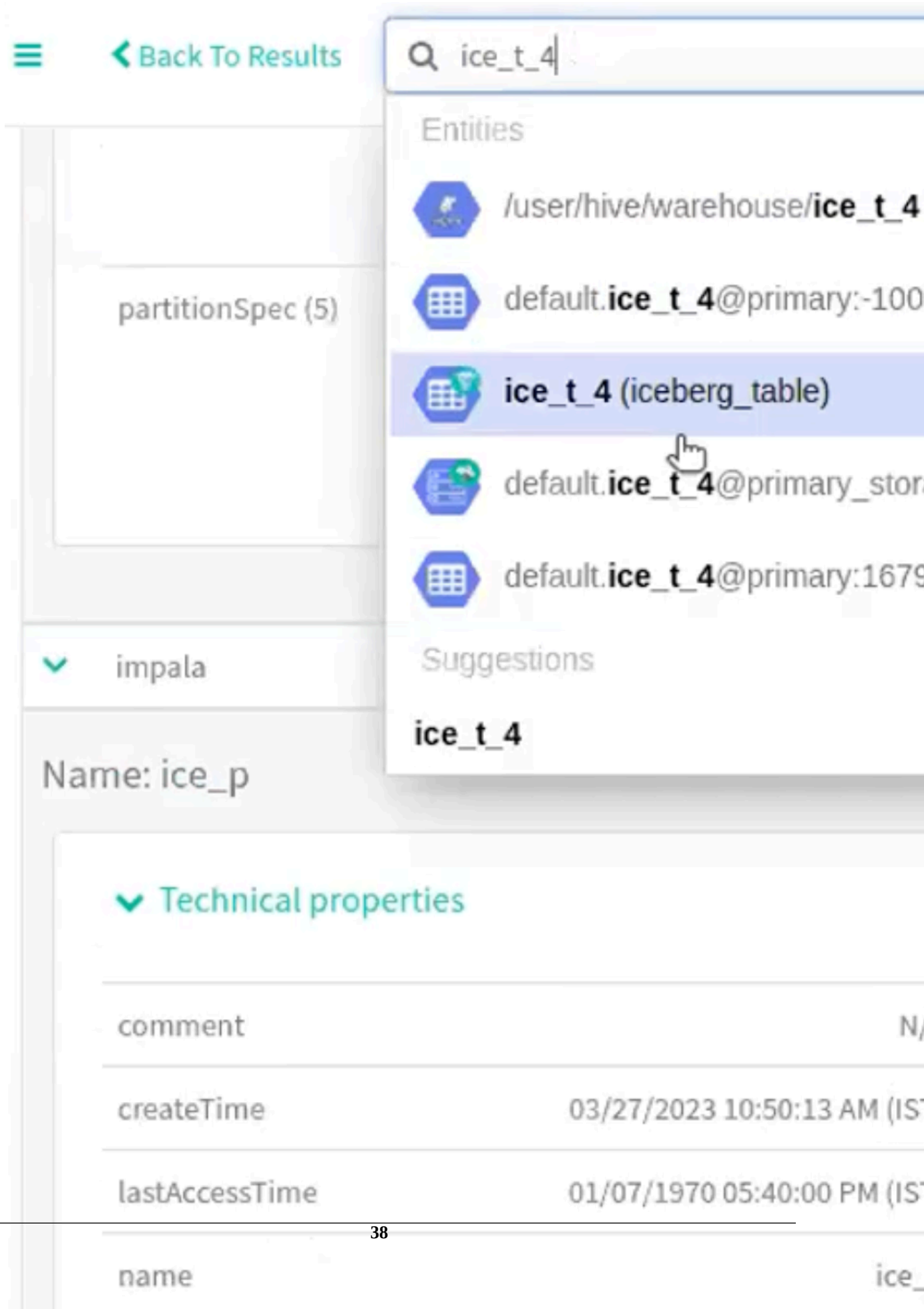
Back To Results
ice_t_4
Entities
/user/hive/warehouse/ice_t_4
default.ice_t_4@primary:-100
ice_t_4 (iceberg_table)
default.ice_t_4@primary_stor
default.ice_t_4@primary:1679
Suggestions
ice_t_4
partitionSpec (5)
impala
Name: ice_p
Technical properties
comment
N
createTime
03/27/2023 10:50:13 AM (IS
lastAccessTime
01/07/1970 05:40:00 PM (IS
name
ice_

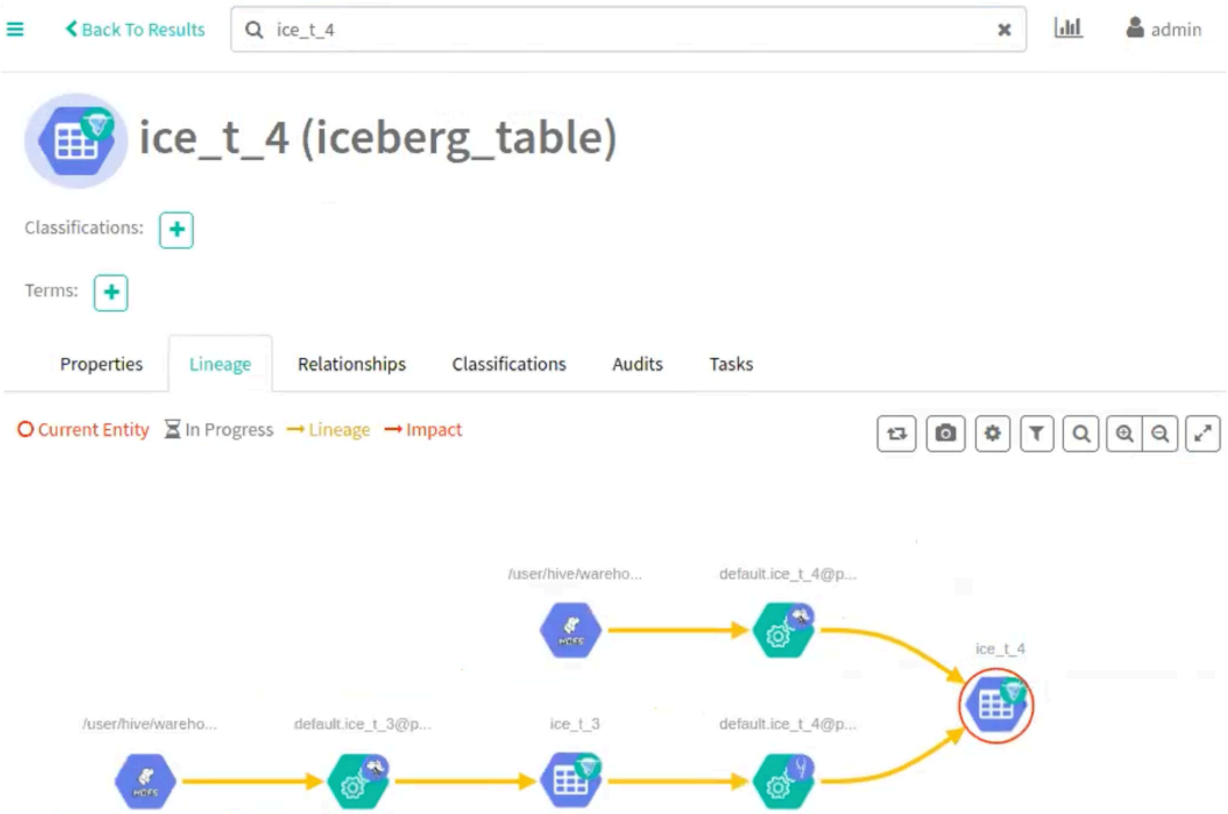

The process is completed.

Related Information

Using the Spark shell

Using the Hive shell

Cloudera Runtime 7.3.1

Atlas Import Utility

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Importing Hive Metadata using Command-Line (CLI) utility.....4
Bulk and migration import of Hive metadata.....5
Using Atlas-Hive import utility with Ozone entities.....6

Setting up Atlas Kafka import tool.....6

Importing Hive Metadata using Command-Line (CLI) utility

You can use the Atlas-Hive import command-line utility to load Atlas with databases and tables present in Hive Metastore.

This utility supports importing metadata of a specific table, tables from a specific database or all databases and tables.

Important: You must add Atlas Gateway role to all hosts which have the Hook-based services like Hive / Hive on Tez / Impala / HBase / Spark / Kafka. Additionally, when Atlas hooks are disabled, atlas-application property file is removed. Therefore, while running import-hive/hbase/kafka scripts, make sure that the Atlas service-hook is **ON**.

Consider a scenario where Hive has databases and tables prior to enabling Hive hook for Atlas. In such a situation, the Atlas-Hive import utility can be employed to ensure Hive and Atlas are in sync.

Also, the utility can be used in a scenario where Atlas is unable to process specific messages due to some errors that could possibly occur with Kafka.

Important: You must run the tool on the Hive node.

Attention: Ensure that the end user (e.g. hive) running the import-hive script has permissions in Ranger to create, update, and delete Atlas entities.The user must also have sufficient permissions to access Hive databases, tables, and columns that need to be imported into Atlas.

Supported Hive Metadata import options:

Atlas-Hive utility supports various options which can be used while importing Hive Metadata:

- -d <database regex> OR --database <database regex>

Specify the database name pattern which has to be synced with Atlas.

- -t <table regex> OR --table <table regex>

Specify the table name pattern which has to be synced with Atlas. It must be used along with -d.

- -f <filename>

Imports all databases and tables in the specified file. The file must have one entry on each line where the entry is in the form of <database_name>:<table_name>.

For example:

A scenario where the user wants to import two tables named t11 from database db1 and t21 from db2 and all tables from db3. The file content must be:

- db1:t11

- db2:t21

- db3

- No options

  Does not specify any option to import all databases from Hive into Atlas.

  A sample usage of the script Atlas hook in Hive is not configured and hence no “Live” data gets reflected into Atlas.

Later, you configure the Atlas hook but it is observed that the Hive database already contains entities that need to reflect in Atlas. In such cases, an Atlas-hive import script reads the database and tables from Hive Metadata and creates entities in Atlas.

An example of Atlas-hive script:

Usage 1: <atlas bundle>/hook-bin/import-hive.sh

Usage 2: <atlas bundle>/hook-bin/import-hive.sh [-d <database regex> OR --database <database regex>] [-t <table regex> OR --table <table regex>]

Usage 3: <atlas bundle>/hook-bin/import-hive.sh [-f <filename>]

File Format:

database1:tbl1

database1:tbl2

database2:tbl1

Limitations of using Atlas-Hive import script

The Atlas-Hive import utility has the following limitations:

- Cannot delete entities which are dropped from Hive but do exist in Atlas.
- Cannot create lineages.

Note: If a specified entity is not available in Atlas, the entry gets created in Atlas. If a specified entity exists in Atlas, it gets updated with the latest metadata state present in Hive.

Bulk and migration import of Hive metadata

The existing sequential import of Hive metadata into Apache Atlas is time consuming. The new bulk and migration import process which creates a zip file first and then runs the Atlas bulk import using the zip file, improves performance and is faster. An improvised method to import Hive metadata into Atlas is now available.

The previous version of Hive Metadata import was a sequential import into Atlas. The time taken for manual import of databases to the Atlas system consumes more time. With the need to import more databases, additional time is consumed in completing the import process and the process takes longer to complete.

Instead of Atlas sequential import, this update enables you to import without the Atlas interaction, where the existing Hive import is updated to import the Hive databases and/or tables into zip files. And later, you can import the zip file into Atlas using a bulk or migration import process.

By default, the enhanced Hive metadata import creates the zip file and then automatically runs Atlas bulk import using the same zip file.

As an example, use the following commands to perform the enhanced new Hive metadata import.

./import-hive.sh -t hive_db_vxofz.hive_table_tmwos -o /tmp/import.zip

Optionally, you can perform Hive metadata import using the following two phases separately to use either the bulk or migration import process:

1. Create the Hive metadata export zip file.
2. Use the exported zip file to run the bulk or migration import process for Atlas.

As an example, use the following commands to perform the new Hive metadata import in phases.

1. To create the Hive metadata export zip file, run the following command:

   ./import-hive.sh -i -t hive_db_vxofz.hive_table_tmwos -o /tmp/import.zip

Option	Description

-o or --output	Used to create the zip file. • If you specify a zip file followed by the -o option, it creates the zip file. • If you provide only a path, then by default this option tries to create a import-hive-output.zip file inside the path. If the zip file already exists, the import process fails.
-i or --ignorebulkImport	Used to ignore the second phase, that is the bulk import execution (the default behavior is to invoke the bulk import process)

2. To Import the create a zip file using bulk import, use the CURL command:

   cat > /tmp/importOptions.json

   {"options":{"updateTypeDefinition":"false", "format":"zipDirect", "size":"1"}} kinit -kt /cdep/keytabs/atlas .keytab atlas

   curl -k --negotiate -u : -X POST

   'https://quasar-cgfwjs-1.quasar-cgfwjs.root.hwx.site:31443/api/atlas/admin/import' --form 'request=@/tmp/importOptions.json' --form 'data=@/tmp/import.zip'

   Note: While performing the import operation using the API with this enhancement, the deleteNonExisting option is not supported. For example, when you create tables in a database, perform an import, and drop the table, and later perform an import with deleteNonExisting, the dropped table status shows as Active in Atlas. You must perform this operation using the previous version of Hive Metadata import.

Using Atlas-Hive import utility with Ozone entities

The Atlas-Hive import script also supports creating the corresponding Ozone entities into Atlas, if the Hive entity has an Ozone path associated with it.

For example, if a table was created using an Ozone path, the Atlas-Hive import script creates Ozone related entities into Atlas.

Attention: If a table was loaded using a file located into an Ozone path, Ozone entities will not be created into Atlas as HMS does not furnish the details.

Setting up Atlas Kafka import tool

You can use the Atlas-Kafka import tool or an Apache Kafka import action to load Apache Atlas with Kafka topics.

You must first set up the tool and later run the tool manually to create or update the Kafka topic entities in Atlas. The tool uses client configuration to fetch the required configuration, like the Atlas endpoint and Zookeeper.

Important: You must run the tool on the Kafka node.

The Kafka type definition in Atlas contains multiple fields. The import tool fills out the following field values:

- clusterName - Contains the value provided by the atlas.metadata.namespace property in the application configuration file. The default value is cm.
- topic, name, description, URI - Contains the topic name.
- qualifiedName - Contains the topic name and the cluserName which is joined by '@'. For instance, my-topic @cm. This serves as the unique identifier for topics.
- partitionCount - Displays the number of partitions of the topic.

To set up the Atlas - Kafka import tool, follow these steps:

1. Select the Atlas service in Cloudera Manager.

2. Deploy client configs: Actions > Deploy Client Configuration.

   Note: If Kerberos authentication is enabled in Apache Zookeeper, you must run the tool as the Kafka Kerberos user and set up the Ranger policies for the Kafka user.

3. If Apache Ranger is enabled, create a Ranger policy with the user running the tool in cm_atlas to allow the <user> to create, update, delete, and read Kafka entities:
   - Log into Ranger UI.
   - Navigate to cm_atlas policies.
   - Select the policy which has create_entity and delete_entity permissions.
   - Add the Kerberos user that is used with the import-kafka.sh tool.
4. SSH into a host where the client configuration is deployed.
5. Run `kinit` as Kafka if Kerberos is used to set up the Ticket Granting Ticket (TGT).
6. Set JAVA_HOME.

   Note: Use export JAVA_HOME=/usr/java/default in this scenario.

7. Run the command /opt/cloudera/parcels/CDH/lib/atlas/hook-bin/import-kafka.sh to import Kafka topics into Atlas.

Note: Existing Kafka topics and clients can also be imported into Apache Atlas as entities using the Import Kafka Topics Into Atlas action available in Cloudera Manager. See the Related information.

Related Information

Importing Kafka entities into Atlas

Cloudera Runtime 7.3.1

Atlas Business Metadata

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Business Metadata overview.....4

Creating Business Metadata.....4

Adding attributes to Business Metadata.....5

Associating Business Metadata attributes with entities.....7

Importing Business Metadata associations in bulk.....9

Searching for entities using Business Metadata attributes.....9

Business Metadata overview

Atlas allows you to define your own attributes as key-value pairs to add to the technical metadata Atlas stores for data assets. These attributes are grouped in Business Metadata collections so you can control which users have access to create, update, and assign these attributes.

There's a fixed set of attributes that Atlas collects for each entity type from services on the cluster. You can augment these attributes by updating the entity model, but this requires changing the code in the plugin that manages metadata collection for the service. Defining business metadata attributes allows you to add to the technical metadata for an entity type, but also to control access to those attributes in groups. You can use business metadata groupings or collections in Ranger policies to determine who can view business metadata attributes, who can set their values, and who can create additional attributes. For example, you could create a business metadata collection called "Operations," that included attributes such as IT-owner, Operational Phase, Processing Strategy, and Processed Date. In a Ranger policy, you could expose these attributes only to the IT staff or expose them to all users but only allow the IT staff to set their values. You could define attributes that had meaning for a specific entity type or applied to any entity types.

In Atlas, you define a business metadata collection—simply the label for a group of attributes. You can than define attributes in that business metadata collection. After you've defined business metadata attributes, you can add values through the Atlas UI or apply them in bulk to entities using a bulk import process.

Related Information

Creating Business Metadata

Adding attributes to Business Metadata

Associating Business Metadata attributes with entities

Importing Business Metadata associations in bulk

Searching for entities using Business Metadata attributes

Creating Business Metadata

Create Business Metadata collections from the Administration page.

1. To access the Atlas Administration features, choose Administration from the user menu in the top right of the Atlas window.

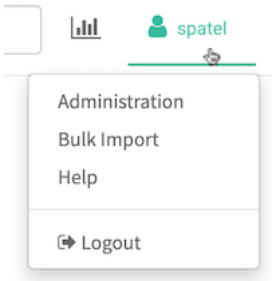

Users need administrator privileges to access the Administration panel features.

2. In the Business Metadata tab, click Create Business Metadata:

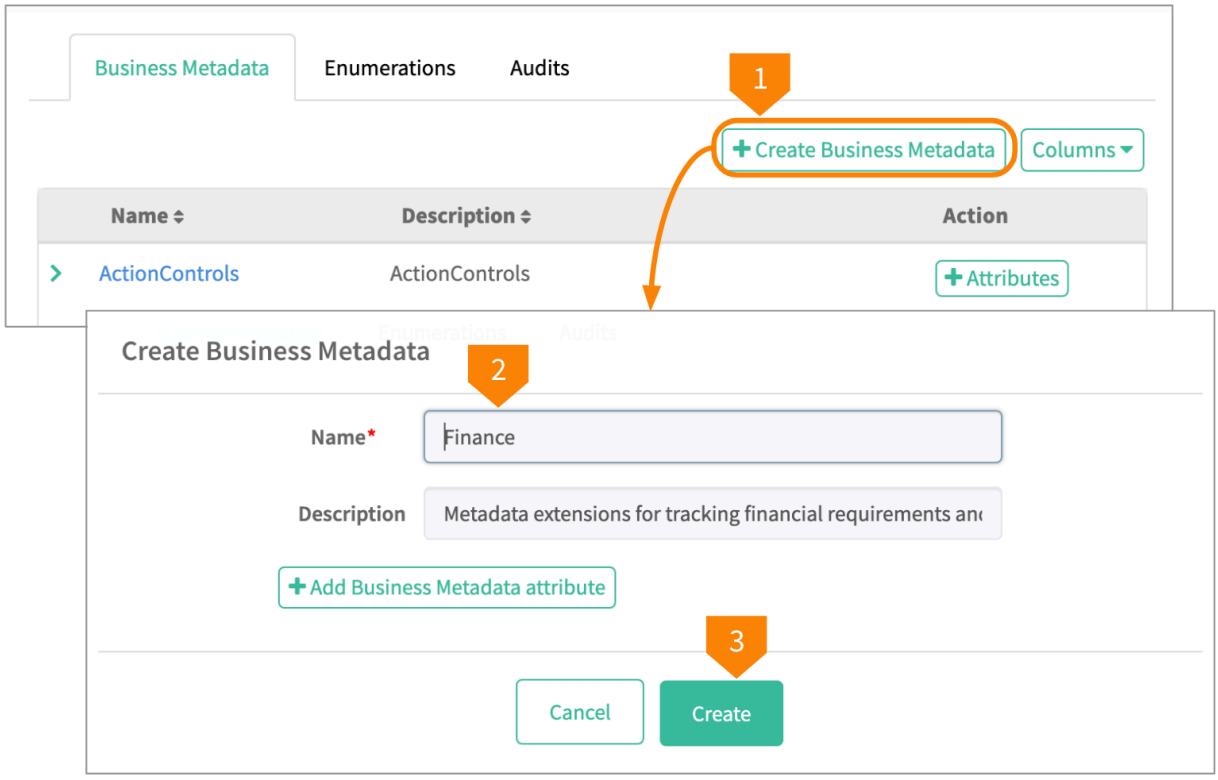

Business Metadata names can be 255 characters. Names must start with a letter and can contain letters, numbers, underscores, and spaces. Descriptions can be 1024 characters and contain any characters.

Note that, once the Business Metadata collections are created, the following scenarios are applicable:

- You can add new attributes to the existing business metadata.
- You cannot remove or delete any existing attribute associated with the business metadata.
- You cannot change or rename the business metadata or the associated attribute.

Adding attributes to Business Metadata

Business Metadata attributes allow you to extend the metadata for a given entity type. Create them in a Business Metadata collection from the Administration page.

Important: At this time, Atlas doesn't support searching on business metadata attributes that allow multiple entries (arrays).

1. To access the Atlas Administration features, choose Administration from the user menu in the top right of the Atlas window.

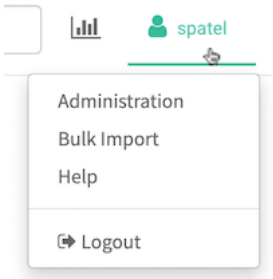

   Users need administrator privileges to access the Administration panel features.
2. Open the Business Metadata tab.

For a new Business Metadata collection, you can add attributes when you define the Business Metadata collection.

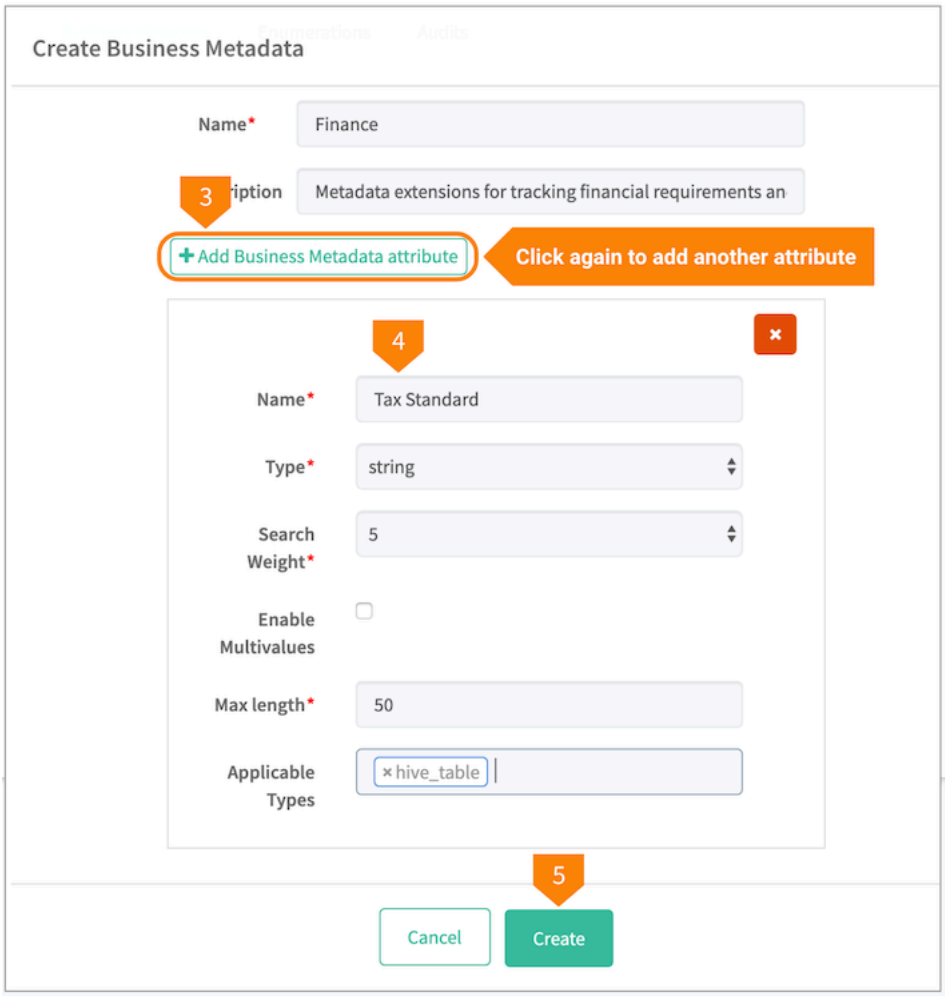

You can also add attributes to an existing Business Metadata collection.

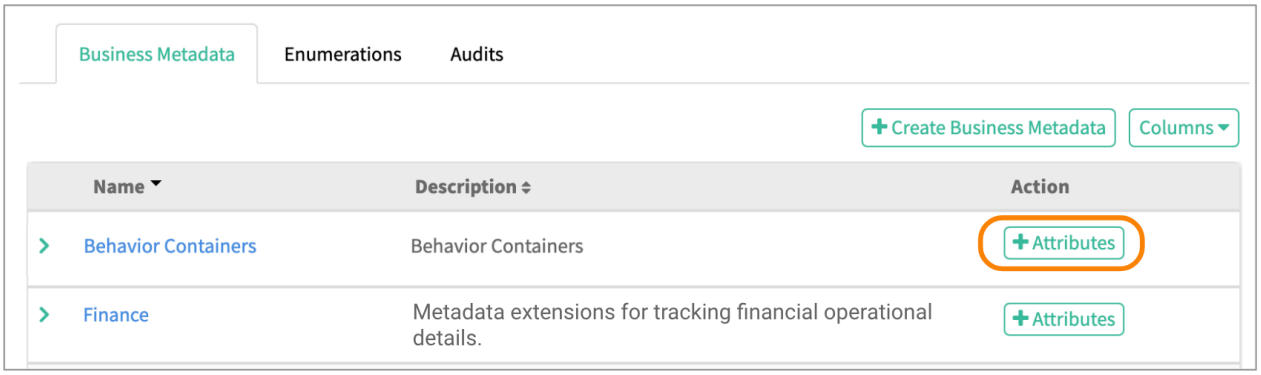

Related Information

Defining Apache Atlas enumerations

Associating Business Metadata attributes with entities

Assign Business Metadata attributes to entities in the entity detail page.

To add a Business Metadata attribute and value in the detail page for an entity:

1. Click Add in the Business Metadata section.
2. Select the Business Metadata collection and attribute from the drop-down menu.
3. Set the attribute value associated with this entity.
4. Click Save.

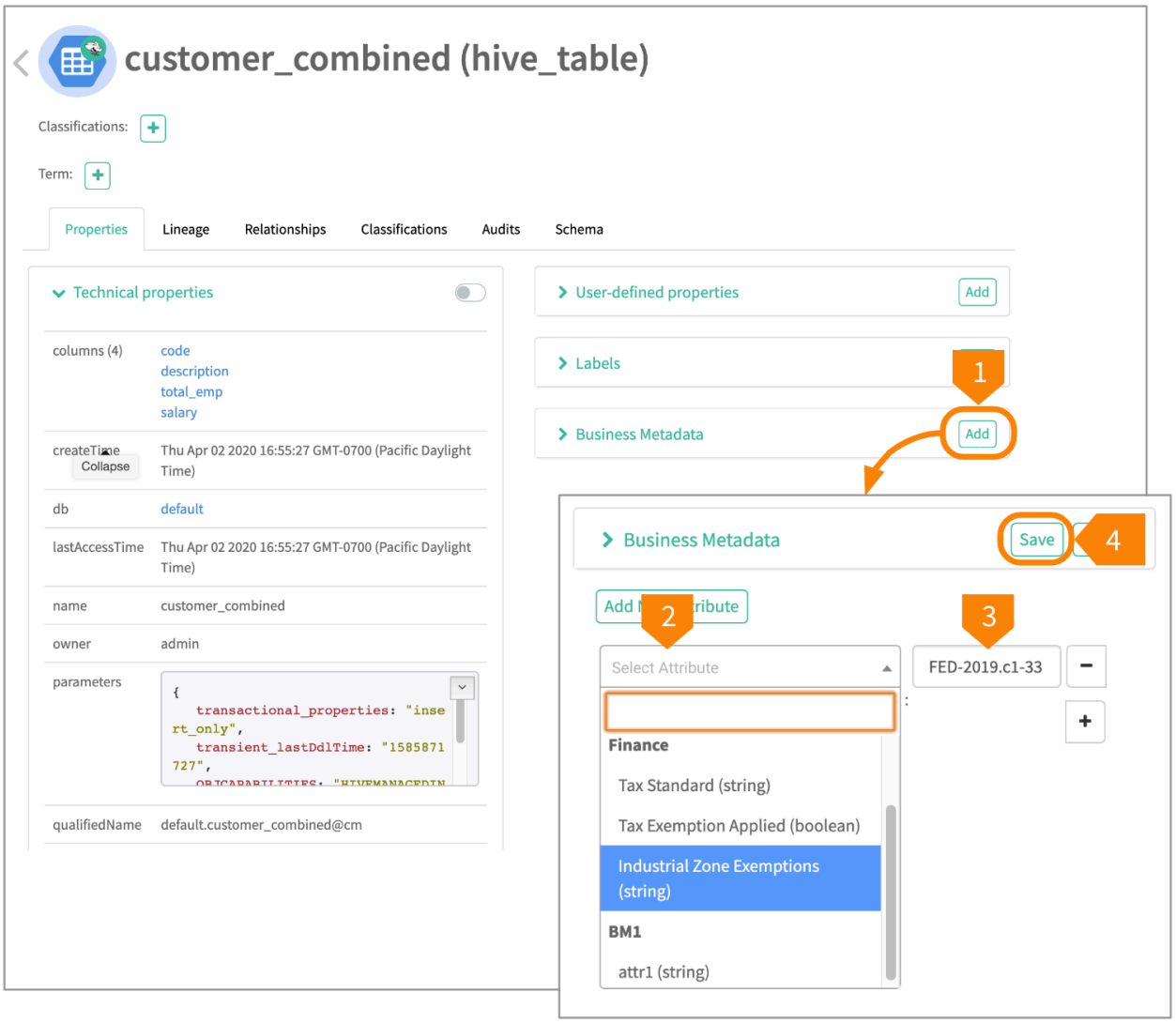

The final result appears in the Business Metadata section:

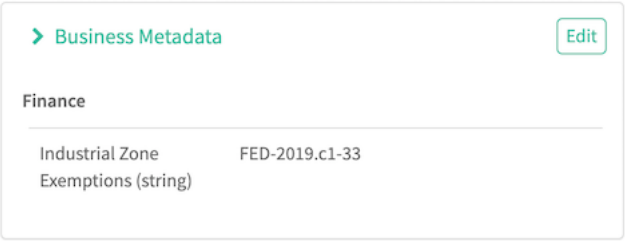

You can update the value of Business Metadata attributes from an entity by clicking Edit, then entering a new value. Be sure to click Save.

You can remove Business Metadata attributes from an entity by clicking Edit, then using the X button next to the attribute.

Related Information

Importing Business Metadata associations in bulk

Importing Business Metadata associations in bulk

After defining Business Metadata attributes, you can assign values to these attributes for individual entities in bulk by creating and importing a comma-separated values file, one row for each assigned attribute value.

To import business metadata attribute values:

1. Log into Atlas.
2. From the user menu, chose Bulk Import Business Metadata Download Import template to download a template for the bulk import file.
3. Fill in the template with the entity names and the attribute values you want to assign to the entities.

   Each line in the file assigns a value to one business metadata attribute on one entity. To assign the value for more than one business metadata attribute to an entity, include multiple lines in the file that refer to the same entity.

Field	Description	Example Value
EntityType	The model type definition for the entity.	hive_table or aws_s3_object.
EntityUniqueAttributeValue	The value of the unique attribute that identifies the entity. By default, the unique-attribute is qualifiedName; however you can specify a different unique attribute name in the optional field EntityUniqueAttributeName (see below).	financedb.annualized_earnings@uswest1 if EntityUniqueAttributeName is not set. annualized_earnings if EntityUniqueAttributeName is set to "name"
BusinessAttributeName	The name of the business metadata attribute to assign to the entity. The name is in the form: <business metadata>.<attribute>	operations.itowner
BusinessAttributeValue	The business metadata attribute value to set for this entity. The value must conform to the data type defined for the attribute. Enclose the value in quotes if it includes spaces or more than one value; use | to separate values.	spatel|nyork
EntityUniqueAttributeName	The name of the technical attribute to use to identify the entity if not qualifiedName.	location name

4. From the user menu, chose Bulk Import Business Metadata Import Business Metadata and select the file to upload.

   If a row in the input file does not import successfully, you will see an error in the UI; additional errors in the same import are not shown. Rows that are correct import even if you see an error for a row that does not import. If you see an error, fix the error and reimport the file. Rows that have already been successfully imported are reapplied without error. Note that if you provide a single entry for an array, the new value replaces all the values of the array. To see all the errors for a given import, look in the Atlas Server log.

Related Information

Accessing Atlas logs

Searching for entities using Business Metadata attributes

Because business metadata attributes are centrally defined and consistently applied across entities, they provide an excellent framework for building an index of search terms that reflect the terminology of your business users. You can add attribute values as filters in Atlas searches.

About this task

After your organization has defined business metadata attributes and data stewards have applied values to the attributes in Atlas entities, users can use the attribute values for searching.

Before you begin

Important: At this time, the free-text search includes only string type Business Metadata attributes. To search on business metadata attributes of other data types, use the attribute filter in Basic Search as shown here. Attributes that allow multiple string values (arrays) are not searchable in free-text or Basic search.

Procedure

1. Select the type of entity you want to search for in the left navigation panel.
2. Click the filter icon.
3. Select the Business Metadata attribute to search for.
4. Form the filter criteria, using operators such as equal, contains, not null, etc.
5. Click Apply if you want to add more filters from the left navigator panel.
6. Click Search to apply this filter and run the search.

Results

Entities with the attribute value appear in the search results.

Related Information

Using Basic Search

Cloudera Runtime 7.3.1

Atlas Glossaries

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Glossaries overview **4**

Creating glossaries **4**

Creating terms **5**

Associating terms with entities **6**

Defining related terms **7**

Creating categories **8**

Assigning terms to categories **8**

Searching using terms **8**

Importing Glossary terms in bulk **9**

Enhancements related to bulk glossary terms import 9

Glossary performance improvements **13**

Glossaries overview

Use glossaries to define a common set of search terms that data users across your organization use to describe their data.

“Data” can describe a wide variety of content: lists of names or text or columns full of numbers. You can use algorithms to describe data as having a specific pattern, of being within a range or having wide variation, but what’s missing from these descriptions is what does the data mean in a given business context and what is it used for? Is this column of integers the count of pallets that entered a warehouse on a given day or number of visitors for each room in a conference center? An Atlas glossary is a way to organize the context information that your business uses to make sense of your data beyond what can be figured out just by looking at the content. The glossary holds the terms you’ve agreed upon across your organization so business users can use familiar terms to find what they are looking for.

Apache Atlas glossaries enable you to define a hierarchical set of business terms that represents your business domain. You can then associate these glossary terms with the metadata entities that Atlas manages.

Glossaries

You can create any number of separate glossaries to group terms used in different domains. Define glossaries to correspond to how your users think about their data. For example, you might want to put terms specific to Sales in one glossary, while your Manufacturing terms might go in a separate glossary. Terms from multiple glossaries can still be associated with the same data; the same term (“Shipping Address City”) can appear in more than one glossary.

Terms

Glossary terms can be thought of as of a flat (but searchable) list of business terms organized by glossaries. Unlike classifications, terms are not propagated through lineage relationships: the context of the term is what’s important, so propagation may or may not make sense.

Categories

Glossary categories enable you to create a hierarchy of categories and subcategories within each glossary. You can then associate terms with these categories, thereby establishing an organizational hierarchy of business terms.

Creating glossaries

Create a new glossary when you need to create a new context for terms.

Typically you would want to create new glossaries when terms are used by significantly different audiences or when the same term has significantly different meanings depending on the context. You can create any number of glossaries; glossary names can include letters, numbers, spaces, and underscores.

Create a new glossary from the Glossary tab in the left navigation pane:

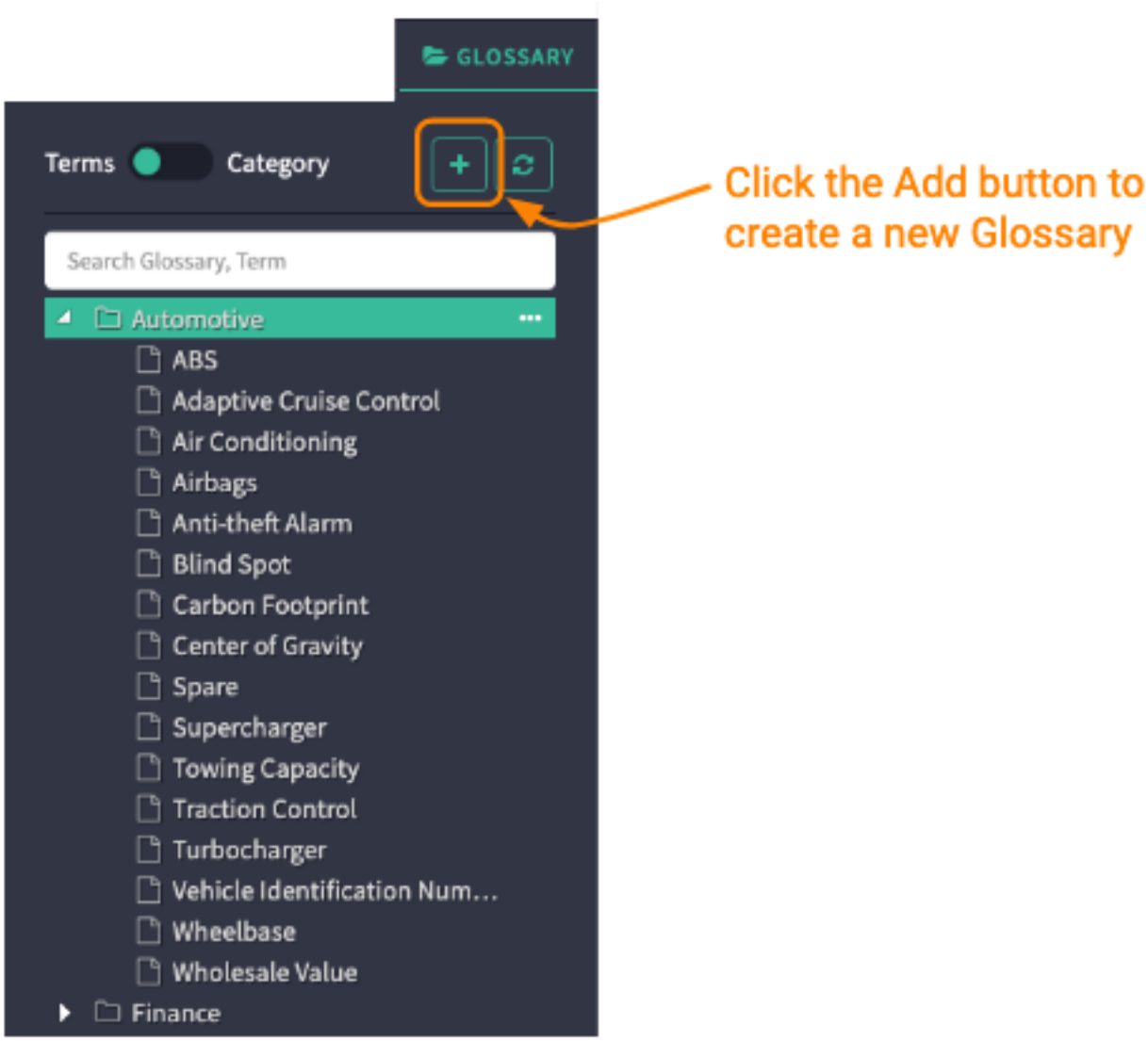

Creating terms

After you've created a glossary, you can add terms to it.

Glossary terms describe business concepts or information.

To create a glossary term, go to the Glossary tab in the left navigation panel. With the toggle switch set to "Terms", click the plus button.

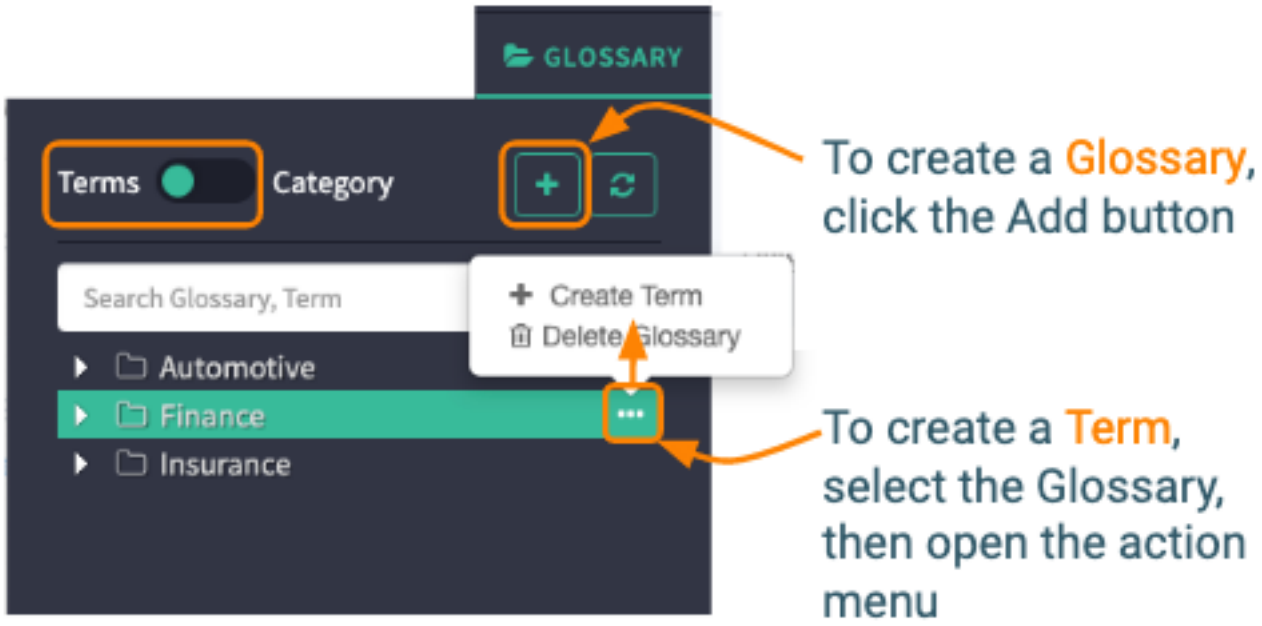

Associating terms with entities

The value of terms comes with using them to identify data and data sets. You can associate terms with entities from search results and within an entity detail page.

When you associate a term with an entity, that entity is part of the group of entities returned when you search on the term.

You can associate a term with an entity in the dashboard:

- From an entity detail page:

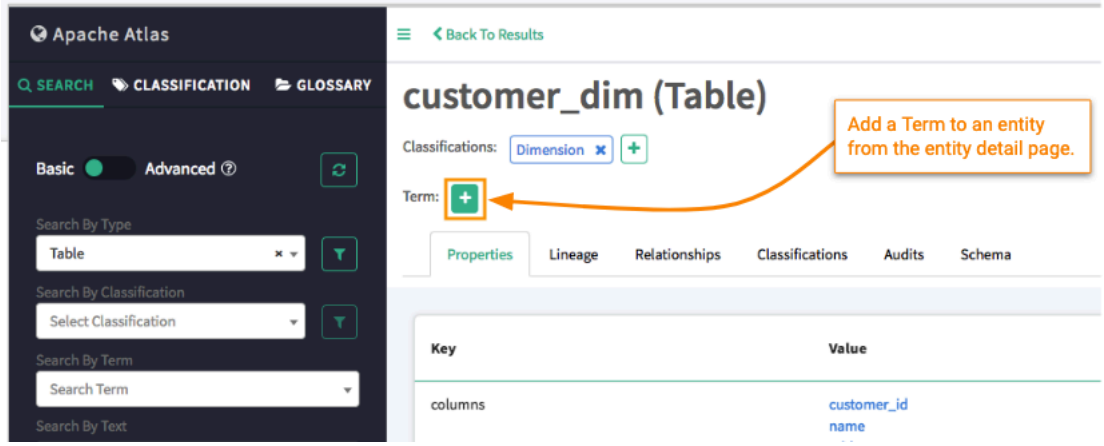

- From search results:

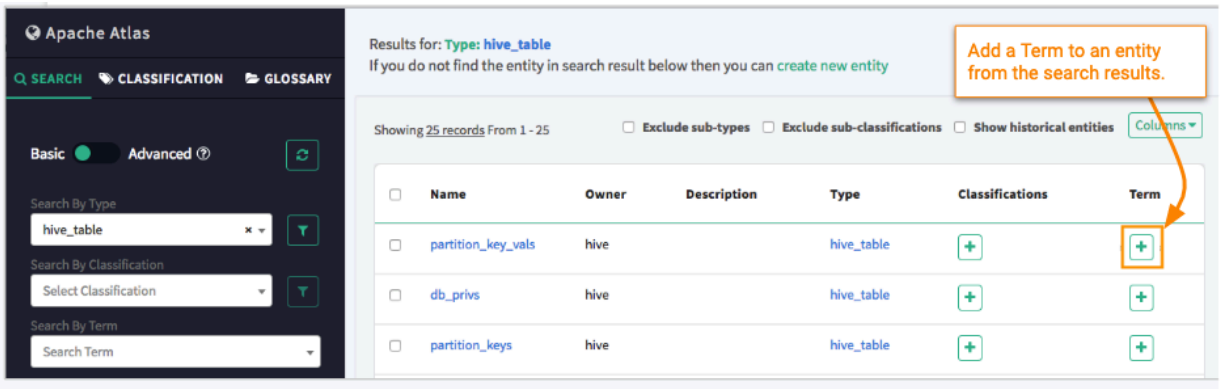

Both methods open a dialog box where you can search for the term or browse through the glossary hierarchy.

Related Information

Searching using terms

Defining related terms

Can't agree on a single term among multiple organizations? Define two terms as synonyms so you can use either term to search for entities. Atlas provides many types of relationships to help data stewards build useful glossaries.

The glossary allows you to indicate relationships among terms. When you indicate a relationship, it is automatically added to the definition of the related term.

To create a relationship between two glossary terms, go to the detail page for one term, open the Related Terms tab, and add a term to the appropriate relationship type.

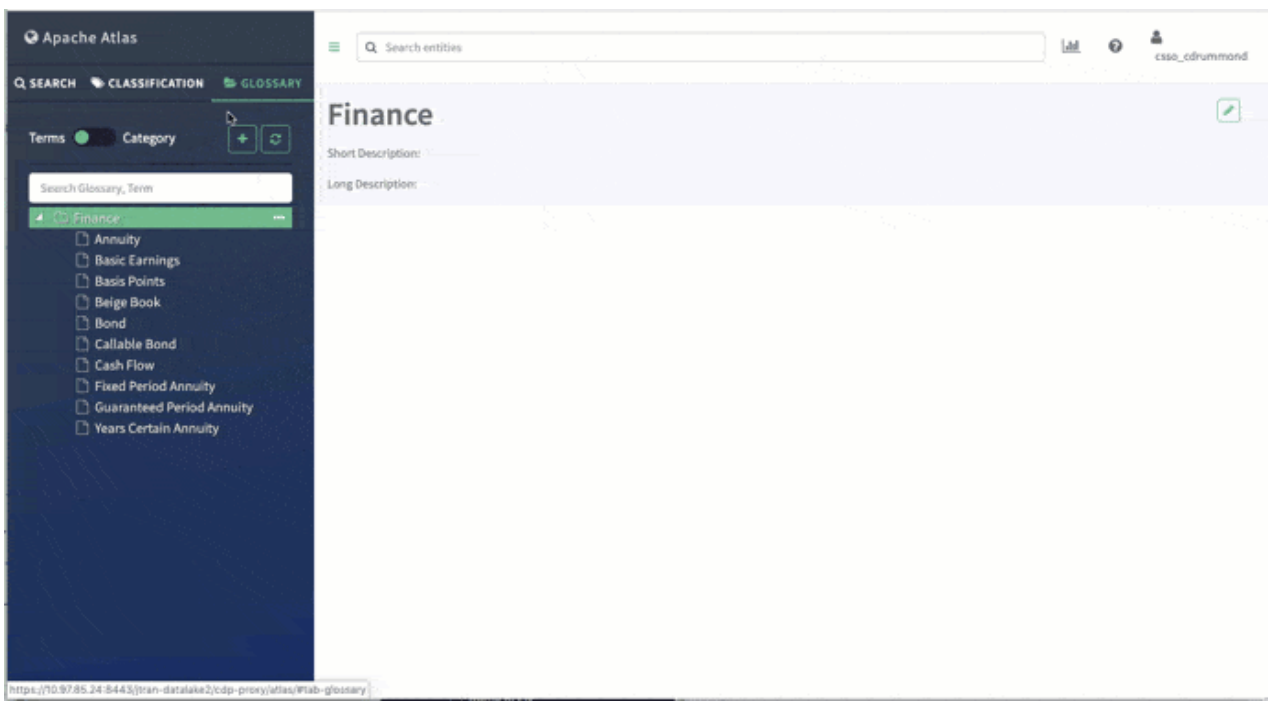

The relationship types are as follows:

Antonym

Glossary terms that have the opposite (or near opposite) meaning, in the same language.

Classifies

The current term is a generic name for the indicated more specific term or terms. "Has Types"

isA

The current term is an example of or type of the identified term. "Is A Type Of"

Preferred Terms

The identified term is the preferred synonym and should be used instead of the current term. "Preferred Synonym"

Preferred To Terms

The current term is the preferred term among a group of synonyms.

Replaced By

The current term is replaced by the indicated term.

Replacement Terms

The indicated term or terms replace the current term.

See Also

The indicated term or terms provide additional information related to the current term.

Synonyms

A relationship that indicates that the current term is the same or similar to the indicated term.

Translated Term

The current term is translated from the indicated term.

Translation Term

The current term is translated into one or more languages in the indicated terms.

Valid Values

Values that are meaningful for this term, such as a list of types that could apply to the term.

Valid Values For

The current term is a valid value for the indicated term.

Creating categories

Categories form the layers in a hierarchy so you can organize glossary terms to make them intuitive to find.

Assigning terms to categories

Group terms into categories to create meaningful contexts in a glossary.

Searching using terms

Now that you have terms defined and associated with entities, you have a search index that lets you quickly find data assets based on their value in your organization.

Importing Glossary terms in bulk

You can import a comma-separated list of Glossary terms into Atlas.

Create the Glossary or Glossaries before you begin the import. Then, download a template CSV file to organize your term list so Atlas can consume it.

Note: The mechanism to import Glossary terms can be used to import new terms only. If you want to update existing terms, use the Atlas UI or API.

To import terms:

1. Log into Atlas.
2. From the user menu, chose Bulk Import Glossary Download Import template to download a template for the bulk import file.
3. Fill in the template with the Glossary term details.

   Consider the following guidelines for the import file:

   - Each line in the file produces a term in a pre-defined Glossary.
   - Only the GlossaryName and TermName are required.
   - Only new terms can be added; if your list contains an existing term, it will fail to import any terms.
   - If you provide other metadata for a term, be sure to include the correct separators so the value is applied to the correct relationship type.
4. When you have a term list to import, from the user menu, chose Bulk Import Glossary Import Glossary Term and select the file to upload.

   If a row in the input file does not import successfully, you will see an error in the UI; additional errors in the same import are not shown. Rows that are correct import even if you see an error for a row that does not import. If you see an error, fix the error, remove the rows previous to the error row, and reimport the file. Note that if you provide a single entry for an array, the new value replaces all the values of the array. To see all the errors for a given import, look in the Atlas Server log.

Related Information

Defining related terms

Accessing Atlas logs

Enhancements related to bulk glossary terms import

After you import Glossary terms into Atlas, you can now list the terms linked to the specific term (Related Terms) in Atlas UI that are provided by the user using the CSV file.

Additionally, you can view the error messages that contain information about the terms that are not in accordance with the import definitions, and also view the status of import from the Response after the import operation is complete. Before this enhancement, the relationship dependency was not in place to import a related term that was created as a part of the same file.

You can now make changes to the CSV file as shown in the example:

```
GlossaryName, TermName, ShortDescription, LongDescription, Examples, Abbrevi
ation, Usage,
          AdditionalAttributes, TranslationTerms, ValidValuesFor, Synonyms,
 ReplacedBy, ValidValues,
          ReplacementTerms, SeeAlso, TranslatedTerms, IsA, Antonyms, Clas
sifies, PreferredToTerms,
```

```
          PreferredTerms
```

```
glossary_1,term_1,"short desc","long desc", "eg", "G1", "Usage",
          "val:100%",,,,,,,glossary_1:term_2,,glossary_1:term_3,,,,"glossar
y_1:term_2|glossary_1:term_3"
```

```
glossary_1,term_2,"short desc","long desc", "eg", "G1", "Usage",
          "val:100%",,,,,,,,,,"glossary_1:term_3|glossary_1:term_2",,,
```

```
glossary_1,term_3,"short desc","long desc", "eg", "G1", "Usage",
          "val:100%",,,,,,,"glossary_1:term_a|glossary_1:term_3",,,,,,
```

And later when you import the CSV file into Atlas, the glossary terms are imported.

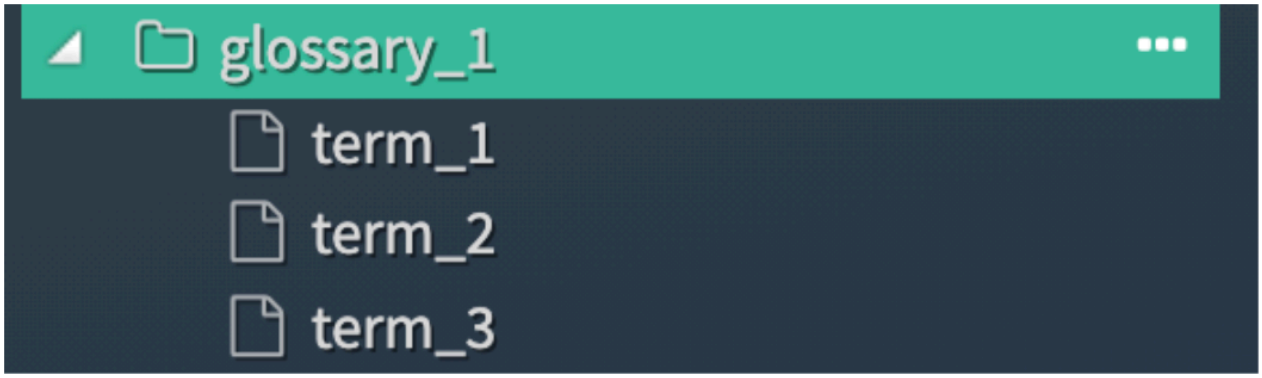

You can view the glossary-associated Related Terms in Atlas, which are displayed as seen in the following example images:

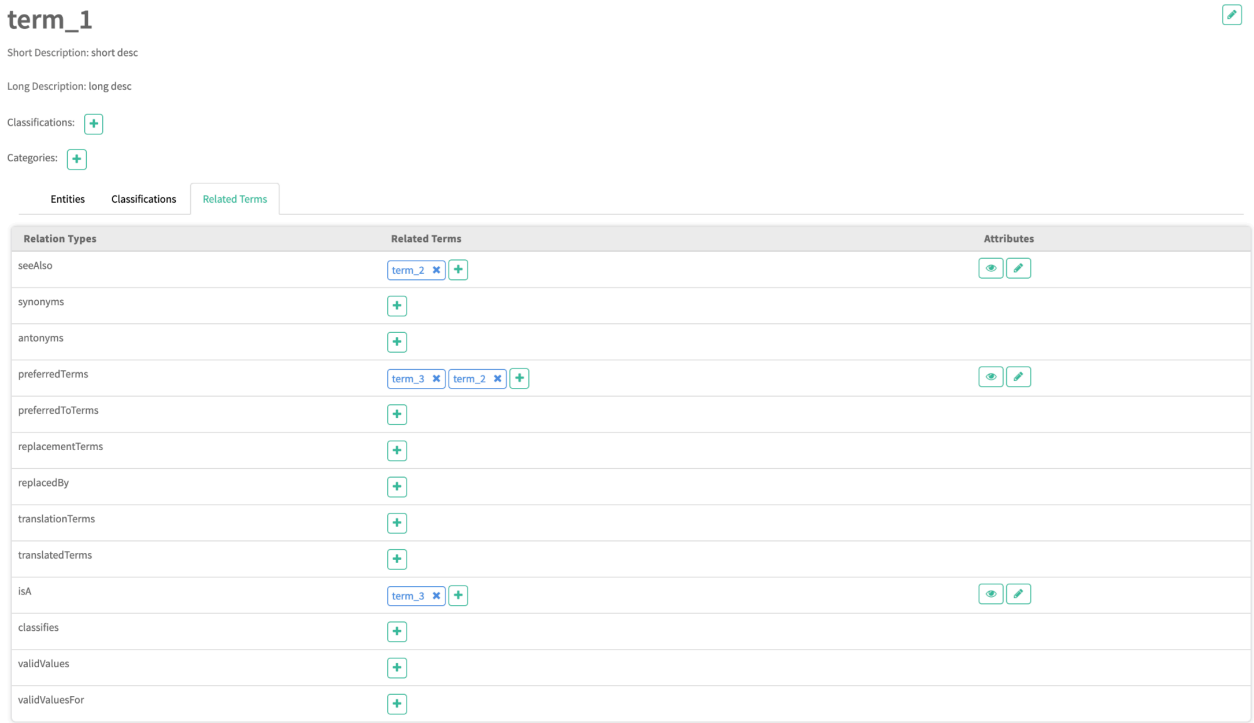

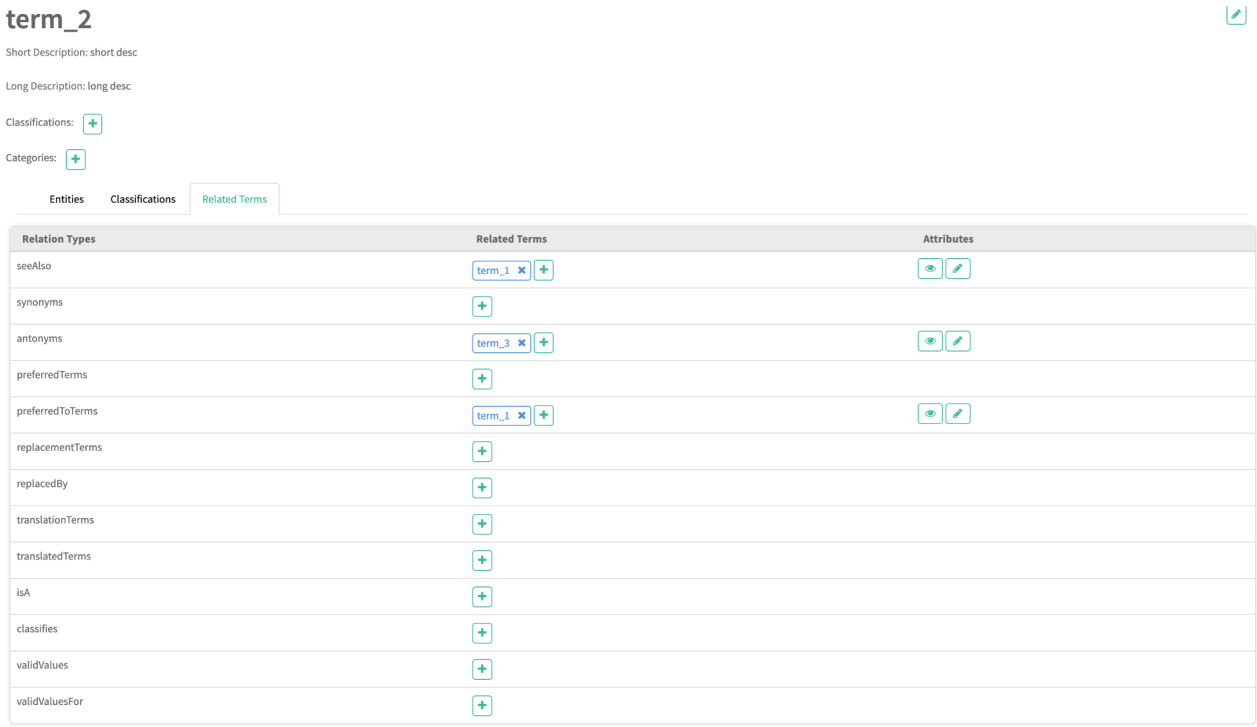

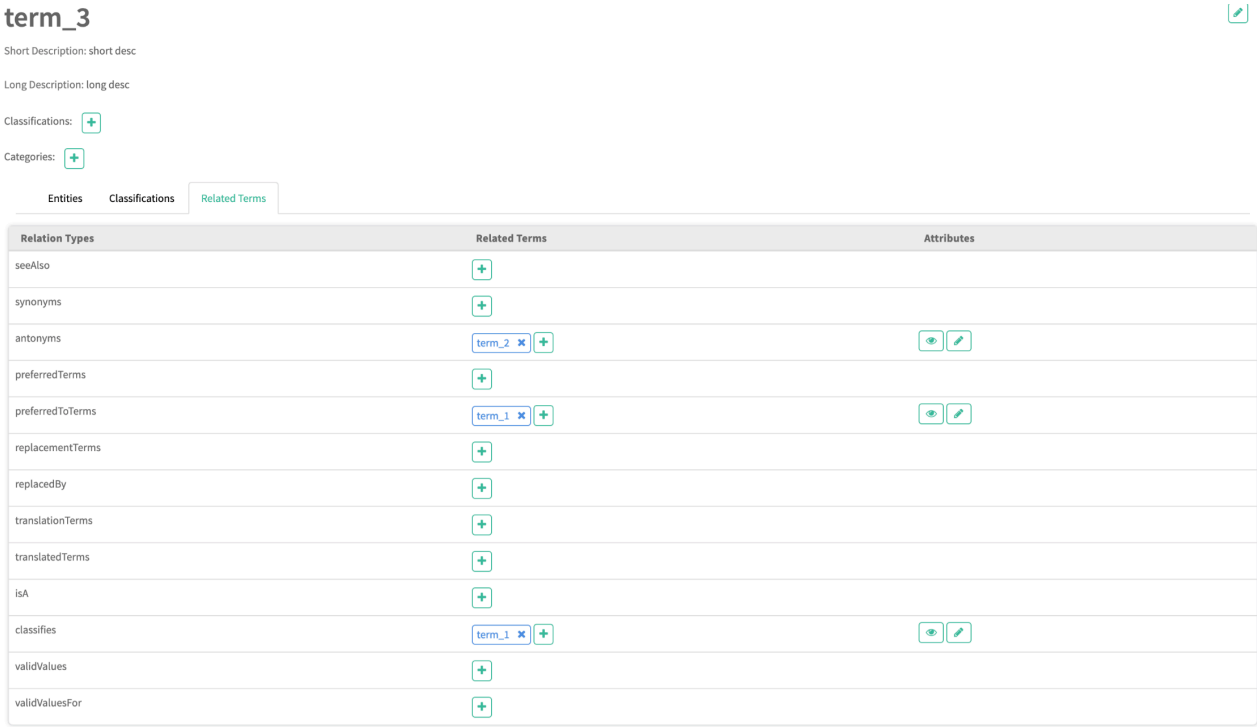

In the example use case, the terms associated with the Glossary named “glossary_1” are listed. And the Related Terms tab displays the list of associated terms.

A new error message is displayed pertaining to the imported CSV file containing specific entries that are not in agreement with the conformed standards with which Atlas operates. The error might be due to a wrong format for including the terms or not adhering to the defined standards. The error message that gets displayed provides all the changes that need to be made.

The following image details an error message:

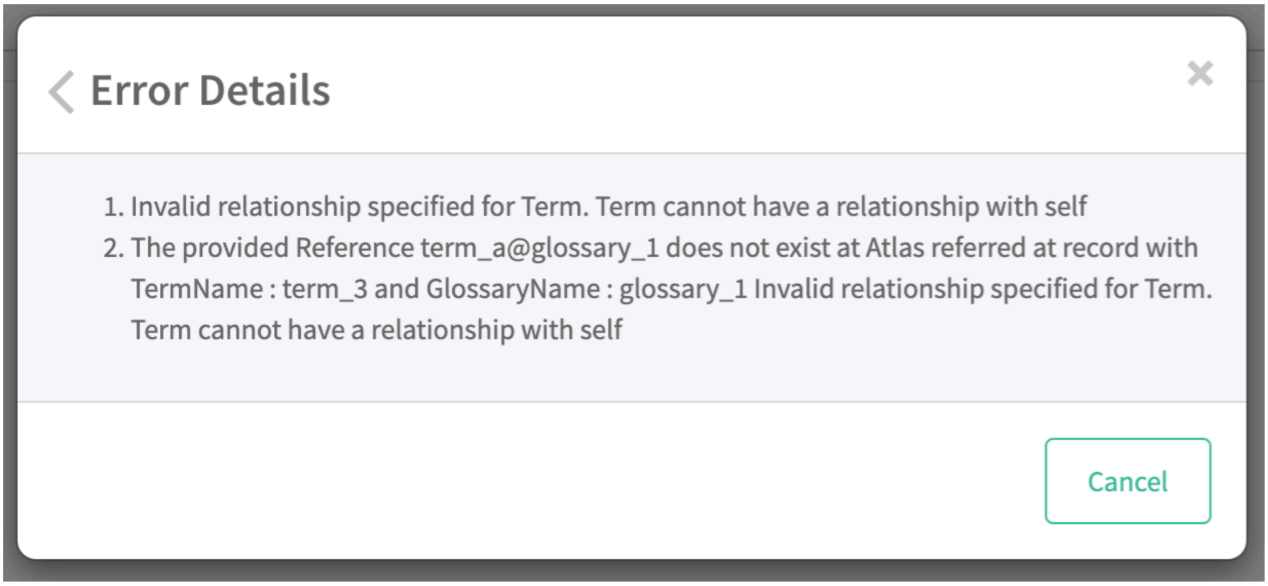

The new response format provides extensive information about the bulk import of glossary terms. This response file contains failedImportInfoList and successImportInfoList.

The messages that are listed in the error message in the Atlas UI are available in this response format.

An example response format:

{

"failedImportInfoList": [

{

"parentObjectName": "glossary_1",

"childObjectName": "term_2",

"importStatus": "FAILED",

```
     "remarks": "Invalid relationship specified for Term. Term cannot have a
          relationship with self"
```

},

{

"parentObjectName": "glossary_1",

"childObjectName": "term_3",

"importStatus": "FAILED",

```
     "remarks": "The provided Reference term_a@glossary_1 does not exist at
Atlas
          referred at record with TermName  : term_3 and GlossaryName : gl
ossary_1\nInvalid
          relationship specified for Term. Term cannot have a relationship
 with self"
```

}

],

"successImportInfoList": [

{

"parentObjectName": "glossary_1",

"childObjectName": "term_1",

"importStatus": "SUCCESS",

```
     "remarks":
          "{\"termGuid\":\"650e4f8e-cebe-43c7-a312-72216ab975f6\",\"qualifie
dName\":\"term_1@glossary_1\"}"
```

},

{

"parentObjectName": "glossary_1",

"childObjectName": "term_2",

"importStatus": "SUCCESS",

```
     "remarks":
          "{\"termGuid\":\"874392b7-8dff-4c3a-881e-c57afe5b2919\",\"qualifie
dName\":\"term_2@glossary_1\"}"
```

},

{

"parentObjectName": "glossary_1",

"childObjectName": "term_3",

"importStatus": "SUCCESS",

```
     "remarks":
          "{\"termGuid\":\"d2b9b591-5912-4b14-8793-58ee04d06d17\",\"qualifie
dName\":\"term_3@glossary_1\"}"
```

}

]

}

Glossary performance improvements

Atlas glossary peformance improvement information is available on this page.

Some of the performance related use cases are listed below:

- Insignificant database calls made when there is a large number of data associated with the Glossary (say about 10K terms or categories)

Earlier, for Glossary GET API it took about 180 seconds, 168.814 seconds, and 169.237 seconds respectively. AVERAGE: 172.683 seconds.

Currently, for Glossary GET API with performance changes it takes about: 45 seconds, 45 seconds, and 43.626 seconds respectively. AVERAGE: 44.542 seconds

- The Bulk Term creation process took about a day to complete the creation process of about 2222 terms.

A patch is now provided which stores the optimum information of Glossaries and Categoies in the cache against their respective GUIDs. With the performance optimization, the Bulk Term creation takes about 5-6 minutes to complete the process.

Cloudera Runtime 7.3.1

Apache Atlas Reference

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Atlas Advanced Search language reference.....5

Apache Atlas Statistics reference.....8

Apache Atlas metadata attributes.....11

Dynamic handling of failure in updating index.....14
Configurations used for index recovery.....14
List of REST API for dynamic index recovery.....15

Defining Apache Atlas enumerations.....16

Purging deleted entities.....17
Auditing purged entities.....18
PUT /admin/purge/ API.....19
POST /admin/audits/ API.....22

Apache Atlas technical metadata migration reference.....24
System metadata migration.....24
HDFS entity metadata migration.....25
Hive entity metadata migration.....26
Impala entity metadata migration.....30
Spark entity metadata migration.....32
AWS S3 entity metadata migration.....33

NiFi metadata collection.....35
How Lineage strategy works.....36
Understanding the data that flow into Atlas.....38
NiFi lineage.....38
Atlas NiFi relationships.....39
Atlas NiFi audit entries.....41
How the reporting task runs in a NiFi cluster.....41
Analysing event analysis.....42
Limitations of Atlas-NiFi integration.....43

HiveServer metadata collection.....44
HiveServer actions that produce Atlas entities.....44
HiveServer entities created in Atlas.....45
HiveServer relationships.....50
HiveServer lineage.....51
HiveServer audit entries.....52

HBase metadata collection.....53
HBase actions that produce Atlas entities..... 54
HBase entities created in Atlas..... 54
Changing the column family compression type..... 58
Hbase lineage..... 58
HBase audit entries..... 59

Schema Registry metadata collection.....59
Configuring Atlas and Schema Registry..... 61
Schema Registry actions that produce Atlas entities..... 62
Schema replationships..... 62
Schema Registry audit entries..... 64
Troubleshooting Schema Registry..... 64

Impala metadata collection.....64
Impala actions that produce Atlas entities..... 65
Impala entities created in Atlas..... 66
Impala lineage..... 68
Impala audit entries..... 69

Kafka metadata collection..... 69
Kafka actions that produce Atlas entities..... 70
Kafka relationships..... 70
Kafka lineage..... 71
Kafka audit entries..... 71

Spark metadata collection..... 72
Spark actions that produce Atlas entities..... 73
Spark entities created in Apache Atlas..... 73
Spark lineage..... 76
Spark relationships..... 76
Spark audit entries..... 77
Spark Connector configuration in Apache Atlas..... 77
Spark troubleshooting..... 78

Apache Atlas Advanced Search language reference

Atlas lets you search for metadata using a domain-specific language with a SQL-like format.

If you find that the Basic Search or Free-text Search doesn't allow you to search as precisely as you would like, you can create a query in the Advanced Search interface to return exactly the results you are looking for. Advanced Search queries use a domain-specific language that is intentionally SQL-like.

Each Advanced Search query is in the form of three clauses:

```
FROM WHERE SELECT
```

Additional keywords such as GROUPBY, ORDERBY, and LIMIT can be used to affect the output.

FROM clause

The value specified in the FROM clause acts as the scope of the query. You can specify any entity type in the FROM clause. The possible entity types are the same list as in the Type search; the names are case-sensitive.

The FROM clause is required and also assumed: the first item included in the query (if not literally the word "from") is assumed to be the object of the FROM clause.

Examples

With or without FROM: To retrieve all entities of type "hive_db" use one of the following queries:

```
hive_db
from hive_db
```

If you only specify a FROM clause, Atlas returns all entities of that type.

Note: To avoid unintentional load on the server because of an overly broad search, Atlas returns a maximum of 100 results when no limit is set.

Where Clause

The WHERE clause allows for filtering over the result set identified in the FROM clause by specifying a condition of the form:

```
identifier operator 'literal'
```

The identifier is the name of a property of the entity type specified in the FROM clause. The properties for a given entity type are those shown in the Properties tab of an entity detail page. The names are case-sensitive.

Operators vary by the data type of the literal and include the following:

String: = LIKE

Numeric, Date: = < >

Boolean: =

The LIKE operator allows you to use wildcards in the literal. Asterisk (*) replaces zero to multiple values; question mark (?) replaces a single value.

The literal must be enclosed in single or double quotes. Matches are case-sensitive. Literals can be lists of values. If you specify comma-separated values in square brackets, they act as an OR operation.

Dates used in literals need to be specified using the ISO 8601 format and in single or double quotes.

Boolean values used in literals are lower case "true" and "false" without quotation marks.

You can specify multiple conditions using AND or OR operators. Note that making a list of values is more efficient than using the same identifier in multiple conditions.

Examples:

Exact string: To retrieve all entities of type hive_table with a specific name "time_dim", use:

```
from hive_table where name = 'time_dim'
```

Multiple conditions: To retrieve entity of type hive_table with name that can be either "time_dim" or "customer_dim":

```
from hive_table where name 'time_dim' or name = 'customer_dim'
```

List of values: The query in the example above can be written using a value array:

```
from hive_table where name = ["customer_dim", "time_dim"]
```

Wildcard filtering: To retrieve entity of type hive_table whose name ends with '_dim':

```
from hive_table where name LIKE '*_dim'
```

To retrieve a hive_db whose name starts with R followed by any 3 characters, followed by rt followed by at least 1 character, followed by none or any number of characters:

```
DB where name like "R???rt?*"
```

Date Literal: To retrieve entity of type hive_table created within 2019 and 2020, use the date portion of the time value and specify a range using two phrases connected by AND:

```
from hive_table where createTime > '2019-01-01' and
        createTime < '2019-01-03'
```

Boolean Literal: To retrieve entity of type hdfs_path whose attribute isFile is set to true and whose name is Invoice:

```
from hdfs_path where isFile = true and name = "Invoice"
```

Select Clause

The select clause allows you to specify the properties you want returned in the search results. Properties with simple values can be returned; properties that contain other entities are not available. The property names are case sensitive.

To display column headers that are more meaningful that the system property names, you can use aliases using 'as.'

Examples

Select clause only: To retrieve entities of type "hive_table" with some of its properties:

```
from hive_table select owner, name, qualifiedName
```

WHERE and SELECT clauses: To retrieve entity of type hive_table for a specific table with some properties:

```
from hive_table where name = 'customer_dim' select owner, name, qualifie
dName
```

Change output names using AS: To display column headers as 'Owner', 'Name' and 'FullName'.

```
from hive_table select owner as Owner, name as Name, qualifiedName as FullNa
me
```

Searches with system attributes

In the attribute filter lists, system attributes appear with normal text names. When you use them in advanced searches, use the corresponding field name, which is prefixed with two underscores.

System Attribute	Description	Identifier in Advanced Search
Type	The Atlas entity type.	__typeName
Status	The entity status in Atlas: this field indicates if a data asset has been deleted; Atlas maintains the entity information after the asset no longer exists on the cluster.	__state
Created By User	The Atlas user who created this entity. Typically this is the Atlas system user. If an entity was created by an API call or created manually by users, the active user account would be included in this attribute.	__createdBy
Last Modified User	The Atlas user who last updated the entity, whether through Atlas metadata collection from a cluster service, an Atlas API, or a change through the Atlas UI.	__modifiedBy
Created timestamp	The date Atlas created the entity. Note that this field is different from the technical attribute for the creation date of the original data asset or operation.	__timestamp
Last Modified timestamp	The date when an entity was last updated in Atlas. Note that this field is different from the technical attribute for the last modification date of the actual data asset or operation on the cluster.	__modificationTimestamp
GUID	A unique identifier generated by Atlas. This is the 32-digit code found in the browser URL for an entity.	__guid
Labels	Label metadata added to an entity.	__labels
User-Defined Properties	Key-value pair metadata added to an entity.	__customAttributes
Classifications	Classifications added to an entity.	__classificationNames
Propagated Classifications	Classifications added to entities downstream from an entity where the classification was added by a user.	__propagatedClassificationNames
—	A concatenated string of classification names and attributes for an entity. This attribute is not available through the Atlas UI.	__classificationsText
IsIncomplete	A system indicator that entities were created because they were referenced in the metadata collected by a service other than the source type associated with the entity type. An entity is typically marked "isIncomplete" when Atlas receives metadata out of order from when the events occurred. If IsIncomplete entities remain “incomplete” for a long time, it may indicate that the original messages for entity metadata have not arrived.	__isIncomplete

Advanced Searches using Classifications

You can search for entities that are tagged with a specific classification using "is" or "isa" keywords in either the From or Where clauses. Is and Isa are interchangeable.

Examples

FROM or WHERE clause: To retrieve all entities of type "hive_table" that are tagged with the "Dimension" classification, you could use the following query:

```
hive_table is Dimension
from hive_table where hive_table isa Dimension
```

Related Information

Apache Atlas metadata attributes

Apache Atlas Advanced Search

Apache Atlas Statistics reference

Atlas collects statistics on the metadata it processes. Use this information to help troubleshoot problems and to gauge performance.

To view statistics, click the graph button in the top right corner:

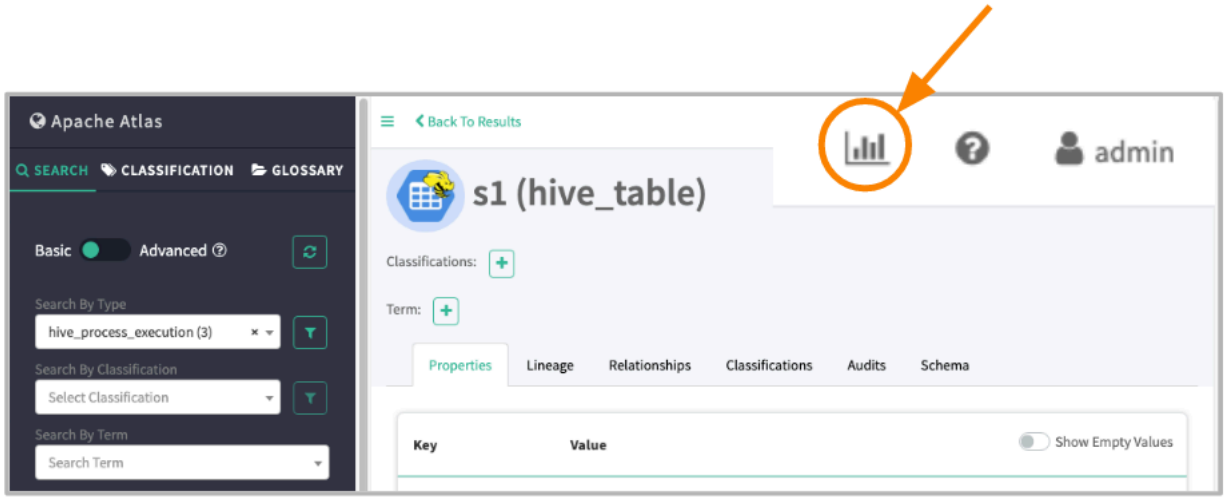

The statistics available are categorized into Entity Statistics and Server Statistics:

Entity Statistics

The distribution of entity across their types. A second column gives the number of these entities that have been marked as deleted.

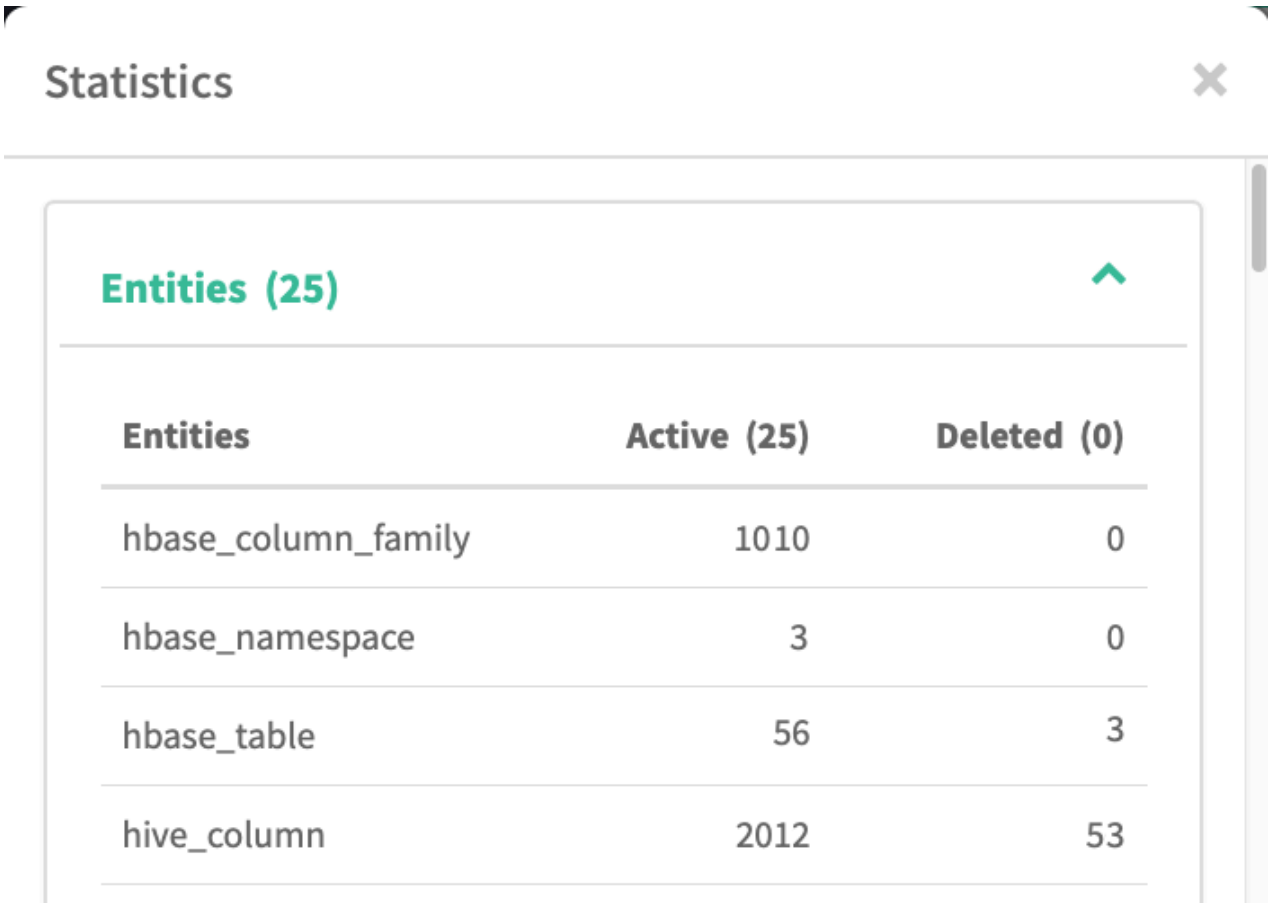

Classification Statistics

A list of classifications assigned to entities and the count of entities marked with that classification. The count for each classification is a hyperlink that runs a search for entities marked with the classification.

Server Statistics

Server statistics reflect the current server session and the metadata collection messages that Atlas reads from a dedicated Kafka topic.

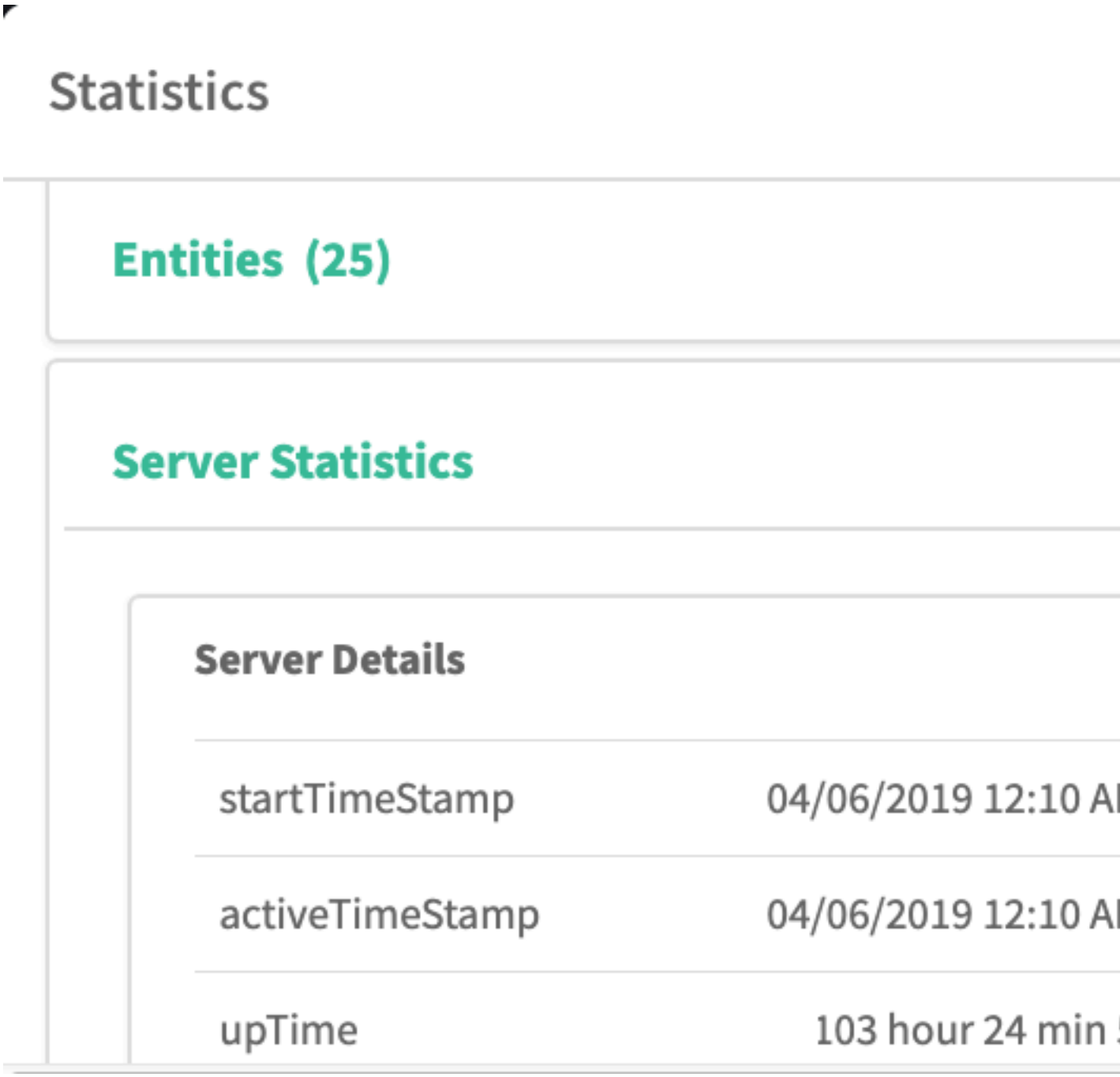

Server Details

startTimeStampThe

The timestamp of the most recent start of the Atlas server.

activeTimeStamp

Same as the startTimeStamp unless Atlas was disabled.

upTime

The amount of time between startTimeStamp and the current time when the server was running.

statusBackendStore

The status of the Atlas server connection to the HBase namespace where entity metadata is stored.

statusIndexStore

The status of the Atlas server connection to the Solr collection where entity metadata is indexed.

collectionTime

The last time metrics were calculated.

lastMessageProcessedTime

The timestamp of the last message Atlas recorded from the Kafka topic where services publish metadata.

offsetCurrent

The index in the Kafka partition that was most recently read.

offsetStart

The index in the Kafka partition that was first read.

Notification Details: Kafka Topic-Partition

Atlas Hook

The primary topic through which services send metadata to Atlas and Atlas sends metadata to Ranger.

Spark-Atlas Hook Topic

A supplementary topic provided for Spark communication to Atlas.

Notification Details: Message Statistics

Period

The interval that the statistic applies to, including the total lifetime of Atlas. Each period indicated includes a timestamp for when the period started.

Count

The number of messages processed by Atlas during the period.

Avg Time (ms)

The average duration between the time that a hook published a message to the Kafka topic to the time entities where successfully created or updated.

Creates

The number of entities produced from the messages processed during the period.

Updates

The number of entities updated based on the messages processed during the period.

Deletes

The number of entities updated based on the messages processed during the period.

Failed

The number of messages that were received but not processed. For more information on what might have prevented these messages from being processed.

Apache Atlas metadata attributes

Attributes are the key-value pairs that hold metadata details for entities and classifications.

Different types of attributes are populated with values differently.

Technical Attributes

These attributes are the entity fields that contain technical metadata defined in entity models. For the built-in entity types, Atlas collects this information from services on the cluster. These attributes are read-only in the UI but can be updated using the Atlas API. All entity types share basic metadata such as names and qualified names; however, the rest of the technical metadata is specific to the entity type.

System Attributes

These attributes are populated by Atlas when it creates an entity instance. They include:

System Attribute	Description	Identifier in Advanced Search
Type	The Atlas entity type.	__typeName
Status	The entity status in Atlas: this field indicates if a data asset has been deleted; Atlas maintains the entity information after the asset no longer exists on the cluster.	__state
Created By User	The Atlas user who created this entity. Typically this is the Atlas system user. If an entity was created by an API call or created manually by users, the active user account would be included in this attribute.	__createdBy
Last Modified User	The Atlas user who last updated the entity, whether through Atlas metadata collection from a cluster service, an Atlas API, or a change through the Atlas UI.	__modifiedBy
Created timestamp	The date Atlas created the entity. Note that this field is different from the technical attribute for the creation date of the original data asset or operation.	__timestamp
Last Modified timestamp	The date when an entity was last updated in Atlas. Note that this field is different from the technical attribute for the last modification date of the actual data asset or operation on the cluster.	__modificationTimestamp
GUID	A unique identifier generated by Atlas. This is the 32-digit code found in the browser URL for an entity.	__guid
Labels	Label metadata added to an entity.	__labels
User-Defined Properties	Key-value pair metadata added to an entity.	__customAttributes
Classifications	Classifications added to an entity.	__classificationNames
Propagated Classifications	Classifications added to entities downstream from an entity where the classification was added by a user.	__propagatedClassificationNames
—	A concatenated string of classification names and attributes for an entity. This attribute is not available through the Atlas UI.	__classificationsText
IsIncomplete	A system indicator that entities were created because they were referenced in the metadata collected by a service other than the source type associated with the entity type. An entity is typically marked "isIncomplete" when Atlas receives metadata out of order from when the events occurred. If IsIncomplete entities remain “incomplete” for a long time, it may indicate that the original messages for entity metadata have not arrived.	__isIncomplete

When defining new models, you can take advantage of the isAppendOnPartialUpdate option in attribute definitions. This option allows array or map type attribute values to be updated by appending rather than replacing the entire set. For example, to represent a list of key-value pairs that can be augmented over time, you might define an attribute metadata as a map with the option isAppendOnPartialUpdate set to true:

```
{
   "name": "metadata",
   "typeName": "map<string,string>",
   "isOptional": true,
   "cardinality": "SINGLE",
   "valuesMinCount": 0,
   "valuesMaxCount": 1,
   "isUnique": false,
   "isIndexable": false,
   "includeInNotification": false,
```

```
   "description": "Contains key-value pairs that provide metadata
",
   "searchWeight": -1,
   "options": {
   "isAppendOnPartialUpdate": "true"
}
```

Classifications, labels, and user-defined properties are included as system attributes in the context of search. They are modeled as entity attributes so that when you access an entity (through the UI or API), you get all these entity-specific metadata.

Business Metadata Attributes

These attributes are populated in the Atlas UI or through API calls. They provide a way to extend the metadata stored for entity instances. You can define Business Metadata attributes to apply to a specific entity type or to many entity types. Administrators can control the users or groups who can set values for these attributes by creating a Ranger policy against the Business Metadata collection that contains the attribute.

Classification Attributes

These attributes are populated in the Atlas UI or through API calls. They provide a way to enrich the worth of a classification for searching, for access policies in Ranger, and for organizing cluster data assets.

Classifications can also be assigned to entities through lineage: if the classification is defined to allow lineage propagation, a classification assigned to an entity is also assigned to all entities that have output relationships to the classified entity. The propagation applies to all further generations of the lineage. Note that Atlas distinguishes between classifications that were specifically assigned to an entity and classifications that were assigned through lineage propagation.

User-defined Properties

These attributes are populated in the Atlas UI or through API calls. They allow users to add metadata in the form of key-value pairs to any entity instance. Values are limited to strings. Both key and value are included in searches. They are not centrally managed like classifications or Business Metadata attributes. They are not accessible through Ranger for specifying access policies.

Defining attributes

Attribute names can include letters, numbers, underscores, and hyphens; they must start with a letter or number. All attributes can have values one of the following Java data types:

- string
- Boolean
- byte
- short
- int
- float
- double
- long
- date
- enumeration

Where enumeration type values are strings from pre-defined enumeration defined using the Atlas API.

Note: The Atlas Free-text search only works with attributes with string values.

When you define an attribute, you an indicate that the value can include more than one entry. Atlas records multiple values in a comma-separated list. Thus, when searching on attributes with multiple values, users should use the logical operator "Contains" rather than "=" so the search matches on a single value rather than the whole list.

Related Information

Atlas Business Metadata overview
Working with Atlas Classifications and Labels
Configuring Atlas Authorization using Ranger
Defining Apache Atlas enumerations

Dynamic handling of failure in updating index

The JanusGraph database transaction might fail in certain scenarios and this failure can be handled dynamically using a specific configuration.

There are certain scenarios in which JanusGraph transactions might fail, including, while indexing into Solr. When you create HIVE entities, the first level of storage includes persisting into HBase and the corresponding indexes are logged in Solr.

The index updation can be partially successful when data gets stored in HBase but Solr indexing fails. There could be inconsistencies with logged indexes for the failed transactions in Solr. These scenarios might lead to a mismatch between the basic and advanced search results in Atlas. Re-indexing the data is an option which can reindex the entire data; however, it is time consuming.

To have a better system resilience and evolution of data tracking, a transaction log processor option named write-ahead is implemented. When enabled, JanusGraph maintains all the transaction log information which can be used to recover indices in the event of failures as described earlier. The log data information consumes more storage space but having a robust system coupled with maintaining data movement overrides the need to have additional storage systems.

The write-ahead configuration inspects the Solr health and registers the time in the event of any issues with Solr. Once Solr is up and running, the indices can be recovered from the previously noted time.

About JanusGraph

Learn what JanusGraph is and how Apache Atlas uses the JanusGraph database to support search index capabilities.

JanusGraph is a scalable graph database optimized for storing and querying graphs containing hundreds of billions of vertices and edges distributed across a multi-machine cluster.

Apache Atlas uses the JanusGraph database at the heart of its metadata repository. This graph is used to show the interconnected relationships between data sources; the data sets they host; the business meaning of the data elements within each data set; the classification of these elements in terms of quality, confidentiality, retention; who (people and processes) are using them and for which purposes.

JanusGraph uses a pluggable persistence store to save the metadata content and a search index for its search API. Apache Atlas takes advantage of this configurability to support a range of size, scalability, and performance requirements.

Configurations used for index recovery

To enable index recovery, you must configure some of the underlying entities.

Before you begin

Configuration properties that must be set:

- atlas.index.recovery.enable: To enable or disable index recovery.

Example: atlas.index.recovery.enable=true.

By default index recovery is enabled on Atlas startup. To disable index recovery, set the value to false. When no index recovery data is available by default, Atlas considers recovery start time based on TTL - (current time minus tx.log TTL). By default SOLR tx.log default TTL is configured as 10 days.

- atlas.graph.index.status.check.frequency: To configure the frequency of SOLR health check. It takes input as time in milliseconds. By default SOLR health checkup takes effect every 30 seconds.

Example: atlas.graph.index.status.check.frequency=5000.

- atlas.index.recovery.start.time: To start index recovery by custom recovery time as provided by the user. This property must be set to a timestamp in the yyyy-MM-dd'T'HH:mm:ssformat

If there are frequent SOLR restart operations or if SOLR is down, setting the start time configuration provides additional clarity.

Example: atlas.index.recovery.start.time=2023-03-24T09:17:23.656Z.

- write.ahead.log.ttl.in.hours: To configure time duration for which the write-ahead transaction log is available. This property value must be set in hours.

Note: The write-ahead logs live up to 10 days as per JanusGraph default settings.

Example: write.ahead.log.ttl.in.hours=36.

List of REST API for dynamic index recovery

You can use Atlas REST APIs that provide flexibility to start index recovery on demand, for index recovery. POST API and GET API are the supported APIs to handle index recovery.

POST API - Start on-demand index recovery

With this API, you can commence index recovery on-demand operation providing the start time for index recovery.

API: POST api/atlas/v2/indexrecovery/start?startTime=2023-04-10T06:17:23.656Z

This API recovers indices starting from earliest of 2023-04-10T06:17:23.656Z or from the transaction log available time.

Note: You can initiate the on-demand index recovery feature only as a user with administrator privileges.

GET API - Get index recovery timing details

Without REST API, if youwanted to retrieve the details about previous or upcoming index recovery, the only option available is to search through Atlas logs. But with this API, you can directly request for index recovery details.

API: GET api/atlas/v2/indexrecovery/

Response to GET API provides recovery time of previous and recent/upcoming index recovery and also provides the customTime which indicates on-demand index recovery started by you through the REST API.

```
"customTime": "2023-04-10T06:17:23.656Z", // Start time of index recovery requested through REST API
"startTime": "2023-04-18T04:04:24.411Z", // Start time of recent/upcoming Index recovery
"prevTime": "2023-04-17T03:54:54.084Z" // Start time of previous index recovery
```

Index recovery time

Learn about the Index recovery vertex properties and how Index recovery time persists in a single graph vertex.

Index recovery vertex has the following properties:

- Property-1: Key as “__idxRecovery_name” and with constant value as ”__solrIndexRecoveryInfo”.

- Property-2: Key as “__idxrecovery_startTime” and it holds the current recovery time.
- Property-3: Key as “__idxrecovery_prevTime” and it holds the previous recovery time.
- Property-4 Key as “__idxRecovery_customTime” and it holds the previous recovery time started through REST API.

  Whenever Apache Solr moves to an unhealthy state the upcoming recovery time is logged in Atlas application log as Index Recovery: Stopped! Recovery time: <timestamp>

  Every time Solr moves to an Active state from an unhealthy state it is logged in Atlas application log as Index Recovery: Started! Recovery time: <timestamp> and current recovery time is reset to null by assigning that time to previous recovery time.

Defining Apache Atlas enumerations

Atlas lets you define enumerations to use as attribute values.

Enumerations are a top-level objects in the Atlas data model. They can be used to standardize values available for users to select when assigning Business Metadata attributes to entities.

Users need administrator privileges to create or update enumerations.

To define enumerations:

1. Log in to Atlas.
2. To access the Atlas Administration features, choose Administration from the user menu in the top right of the Atlas window.

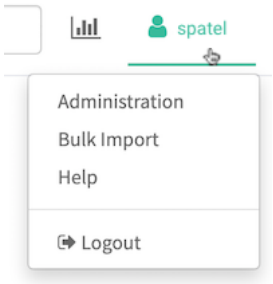

   Users need administrator privileges to access the Administration panel features.
3. Go to the Enumerations tab.
4. Enter the name of the new enumeration and select it in the list.

   Enumeration names must start with a letter and can include letters, numbers, spaces, and underscores.
5. Enter the enumeration values, separating each value by pressing Enter.

   Enumeration values are stored as strings and can include UTF-8 characters including spaces.
6. Click Update.

The enumeration is now available to be used as the type for Business Metadata attributes.

If the enumeration name or values don't meet the requirements, you'll see an error in the top right corner of the Atlas UI.

Related Information

Adding attributes to Business Metadata

Purging deleted entities

You can use Atlas REST API calls to remove entities from Atlas. Only entities that have been deleted in the source system and marked as deleted in Atlas can be purged.

When a data asset is deleted, such as after a DROP TABLE command in Hive, Atlas continues to retain the asset's entity, including metadata, lineage, and audit record. The status of the entity is set to "deleted"; deleted entities show up in search results when the checkbox to Show historical entities is checked. Deleted entities appear in lineage graph dimmed-out.

In some cases, it may be appropriate to remove entities for deleted assets from Atlas. For example, in a development or test environment, you may choose to clean out specific entities rather than clearing the entire Atlas database. Be careful not to purge entities in a production environment without understanding the impact of removing entities on compliance processes in your organization.

Deleted entities can be removed completely from Atlas by using the REST API call PUT /admin/purge.

When you purge a deleted entity:

- The entity is removed from Atlas.
- Related, dependent entities are also removed. For example, when purging a deleted Hive table, the deleted entities for the table columns, DLL, and storage description are also purged.
- The entity is no longer available in search results, even with Show historical entities enabled.
- Lineage relationships that include the purged entities are removed, which breaks lineages that depend upon a purged entity to show connections between ancestors and descendents.
- Classifications propagated across the purged entities are removed in all descendent entities.
- Classifications assigned to the purged entities and set to propagate are removed from all descendent entities.

Note that classifications can propagate to an entity from more than one source; if one source is purged, the classification will remain on the entity as propagated from the other source.

Purged entities cannot be restored.

Atlas retains an audit record of the purge operations, which is available through the REST API call POST /admin/audit. This call allows you to retrieve a list of entities purged in a given time interval. In addition, the Administration Audit tab in the Atlas UI records entities that were successfully purged.

Auditing purged entities

The results of a successful entity purge appear in the Audits tab of the Administration page in Atlas.

To see an audit of successful purges:

1. Log in to Atlas.
2. To access the Atlas Administration features, choose Administration from the user menu in the top right of the Atlas window.

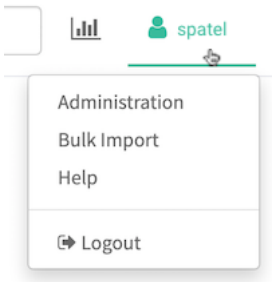

   Users need administrator privileges to access the Administration panel features.
3. Go to the Audits tab.
4. Open the Filters to set one or more filters to reduce the volume of entries.

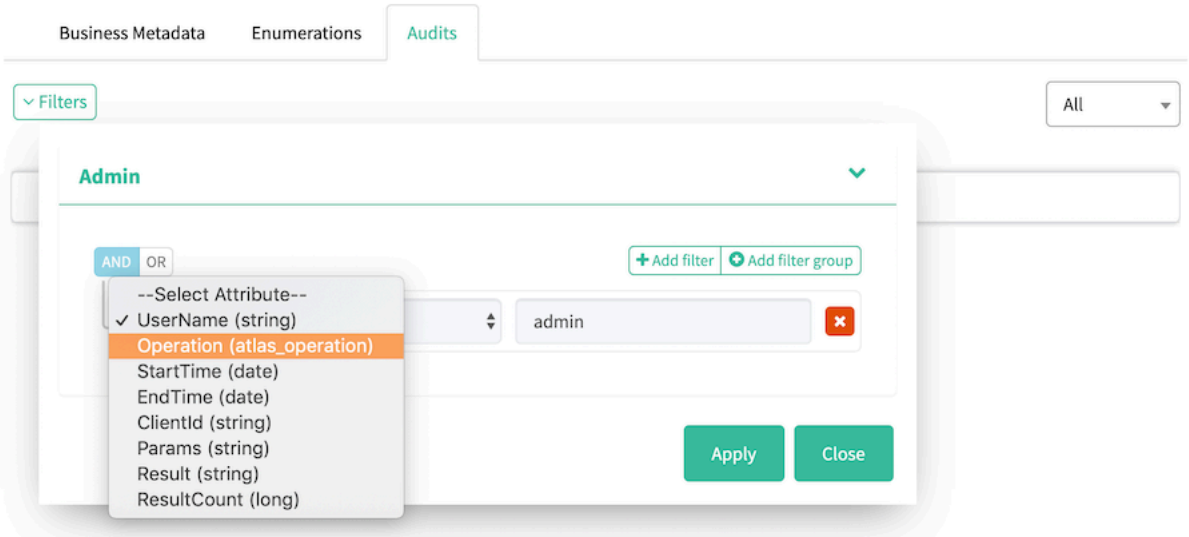

   For example:

   - Set Operation = PURGE to show only purge audits.
   - Set Start Time and End Time to reduce the range of audits.
5. Open a purge audit entry to show a list of GUIDs that were purged.

   Use the arrow on the left end of the row to show the details of the audit entry.

6. Click a GUID to show details about that purged entity.

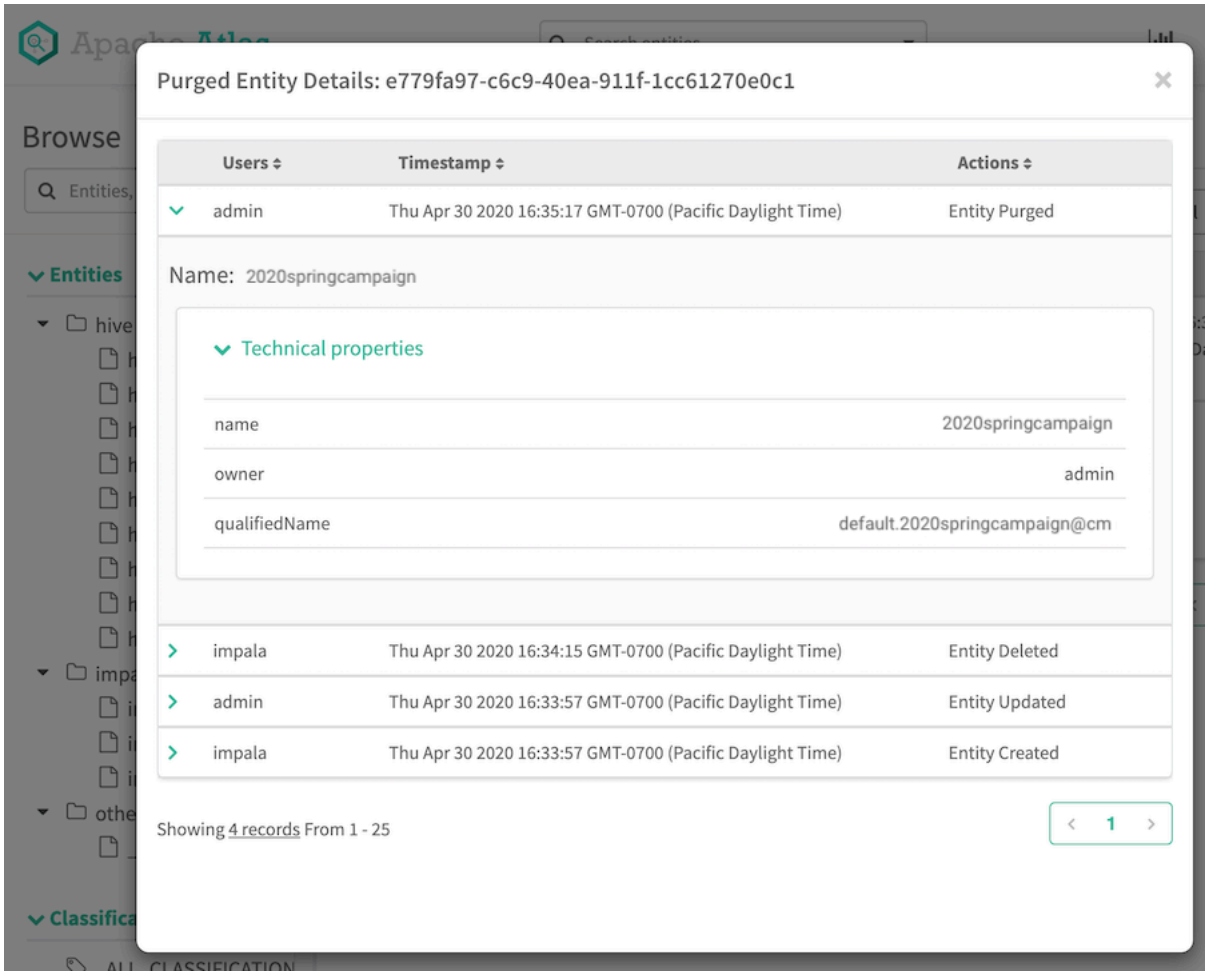

PUT /admin/purge/ API

The PUT /admin/purge/ API endpoint allows you to remove a list of deleted entities from Atlas.

To purge deleted entities, use the PUT method on the /admin/purge/ endpoint with a payload containing a JSON list of Atlas GUIDs:

```
PUT /api/atlas/admin/purge/
```

This call takes a list of GUIDs for Atlas entities; each entity in the list is purged from Atlas if the entity is already marked as deleted. This call requires a user account with Atlas administrator privileges. The successfully purged entities are listed in the audit log, referenced by their GUIDs.

The parameters include:

Parameter	Format / Value	Description
body	JSON list: ["GUID", "GUID", ...]	One or more Atlas entity GUIDs for deleted entities that you want to purge from Atlas. If the list includes an entity that is not already marked as deleted, the entity will be ignored and Atlas will process the remaining entities in the list. Entity GUIDs appear on the URL for the entity detail page and in the metadata returned from an API search call such as GET /search/dsl or POST /search/basic.
header: content type	application/json	Atlas expects the payload of the call to be in JSON format.

Parameter	Format / Value	Description
header: authorization	Per your environment	Atlas requires authentication, which you can provide with username and password as a parameter in a cURL call (-u username:password) or using cookies or other generated methods to pass appropriate credentials. The user must have Atlas administrator privileges.

The response from a PUT /admin/purge call is a JSON-formatted list of the entities that were successfully purged. The content includes the GUID and qualified name of each entity, its status ("DELETED"), and lists of classifications, terms, labels, and related entities that were associated with the purged entity. If no entities are purged, the response is an empty list {}.

Note that the number of entities provided in the request may not match the number of entities included in the response. When Atlas purges an entity, it also purges all additional entities dependent on the indicated entity. Thus if you request to purge a single Hive table, the response includes an entry for the Hive table and entries for each of the table's columns, its DDL, and its storage description.

For example, a cURL command to purge two Hive tables might look like the following, where the authorization is passed in the call as an encrypted string:

```
curl -X PUT 'http://host3.acme.com:31000/api/atlas/admin/purge/' \
-H 'Content-Type: application/json' \
-H 'Authorization: Basic YWRtaW46YWRtaW4=' \
-d '["b9355eab-bbf5-4cd6-b711-12f85a3e9d01", "9fed31f5-0a27-40dc-ba97-96d1
53fc297b"]'
```

The response would include each table, its DDL, its storage description, and all its columns:

```
{
    "mutatedEntities": {
        "PURGE": [
            {
                "typeName": "hive_table_ddl",
                "attributes": {
                    "qualifiedName": "default.2020springcampaign@cm:157896
8155000"
                },
                "guid": "6cfb43ba-d6ec-4628-b98c-da13a7fe35a0",
                "status": "DELETED",
                "displayText": "default.2020springcampaign@cm:1578968155000
",
                "classificationNames": [],
                "meaningNames": [],
                "meanings": [],
                "isIncomplete": false,
                "labels": []
            },
            {
                "typeName": "hive_table",
                "attributes": {
                    "owner": "admin",
                    "createTime": 1578968155000,
                    "qualifiedName": "default.2020springcampaign@cm",
                    "name": "2020springcampaign"
                },
                "guid": "9fed31f5-0a27-40dc-ba97-96d153fc297b",
                "status": "DELETED",
                "displayText": "2020springcampaign",
                "classificationNames": [
                    "Fact"
                ],
                "meaningNames": [],
                "meanings": [],
```

```
                "isIncomplete": false,
                "labels": [
                    "ReviewComplete"
                ]
            },
            {
                "typeName": "hive_storagedesc",
                "attributes": {
                    "qualifiedName": "default.2020springcampaign@cm_storage"
                },
                "guid": "ed59c502-64d9-485a-b5b7-fd2f3d41e2b8",
                "status": "DELETED",
                "displayText": "default.2020springcampaign@cm_storage",
                "classificationNames": [],
                "meaningNames": [],
                "meanings": [],
                "isIncomplete": false,
                "labels": []
            },
            {
                "typeName": "hive_column",
                "attributes": {
                    "owner": "admin",
                    "qualifiedName": "default.2020springcampaign.id@cm",
                    "name": "id"
                },
                "guid": "1f8c8d86-f9d8-4810-889b-0dbfee2c73ff",
                "status": "DELETED",
                "displayText": "id",
                "classificationNames": [],
                "meaningNames": [],
                "meanings": [],
                "isIncomplete": false,
                "labels": []
            },
            {
                "typeName": "hive_column",
                "attributes": {
                    "owner": "admin",
                    "qualifiedName": "default.2020springcampaign.name@cm",
                    "name": "name"
                },
                "guid": "72d689d2-6fae-4de3-bb75-27ab764e1083",
                "status": "DELETED",
                "displayText": "name",
                "classificationNames": [],
                "meaningNames": [],
                "meanings": [],
                "isIncomplete": false,
                "labels": []
            },
            {
                "typeName": "hive_table_ddl",
                "attributes": {
                    "qualifiedName": "default.2019wintercampaign@cm:14889
68156001"
                },
                "guid": "6cfb43ba-d6ec-4628-c98c-bc13a7fe3982",
                "status": "DELETED",
                "displayText": "default.2019wintercampaign@cm:148896815600
1",
                "classificationNames": [],
                "meaningNames": [],
                "meanings": [],
```

```
                "isIncomplete": false,
                "labels": []
            },
            {
                "typeName": "hive_table",
                "attributes": {
                    "owner": "admin",
                    "createTime": 1488968156001,
                    "qualifiedName": "default.2019wintercampaign@cm",
                    "name": "2019wintercampaign"
                },
                "guid": "9fed31f5-0a27-40dc-ba98-bc13a7fe3983",
                "status": "DELETED",
                "displayText": "2019wintercampaign",
                "classificationNames": [
                    "Fact"
                ],
                "meaningNames": [],
                "meanings": [],
                "isIncomplete": false,
                "labels": [
                    "ReviewComplete"
                ]
            }, <additional entries for Hive table columns, ddl, storage de
scription>
        ]
    }
}
```

POST /admin/audits/ API

The POST /admin/audits/ API endpoint can be used to retrieve the entity purge operations that have occurred in a given time period.

To report what entities were purged, use the POST method on the /admin/audits/ endpoint with a payload containing a JSON-formated query for purged entities:

```
POST /api/atlas/admin/audits/
```

where the parameters include:

Parameter	Format / Value	Description
body: auditFilters	JSON-formatted query including: • username (required) • purge operation (required) • start time • end time The results can further be controlled with: • limit • offset • sort by (required) • sort order	The payload describes a query that returns PURGE operations performed by a specific user. The query can include a filter for a specific time range, where the start and end times are specified in UNIX epoch time. The query results can also be paged using a result count (limit) and offset so multiple calls can be made to retrieve unique result sets.
header: content type	application/json	Atlas expects the payload of the call to be in JSON format.
header: authorization	Per your environment	Atlas requires authentication, which you can provide with a username and password as a parameter in a cURL call (-u username:password) or using cookies or other generated methods to pass appropriate credentials. The user must have Atlas administrator privileges.

The audit filter in this call uses the same syntax as Atlas' search filters. Here are some guidelines that are useful for using this interface for purge auditing:

- Valid operators for the time criteria include less than (lt), greater than (gt), less than or equal to (lte), greater than or equal to (gte), equal to (eq), and not equal to (neq).
- Valid operators for the string criteria include like, startsWith, endsWith, contains, isNull, notNull, and equals (eq).
- The sortOrder can be ASCENDING or DESCENDING.

The response from a POST /admin/audits call is a JSON-formatted list of the purge operations that occurred in the specified time range. The content includes the parameters passed in the purge call and the list of GUIDs for the entities that were successfully purged. If no entities are purged, the response is an empty list [].

For example, a cURL command to return the most recent 10 purge operations might look like the following where the authorization is passed in the call as a username and clear-text password:

```
curl -X POST 'http://host3.acme.com:31000/api/atlas/admin/audits/' \
-H 'Content-Type: application/json' \
-d '{
    "auditFilters": {
        "condition": "AND",
        "criterion": [
            {
                "attributeName": "userName",
                "operator": "like",
                "attributeValue": "admin"
            },
            {
                "attributeName": "operation",
                "operator": "like",
                "attributeValue": "PURGE"
            }
        ]
    },
    "limit": 10,
    "offset": 0,
    "sortBy": "endTime",
    "sortOrder": "DESCENDING"
}
' -u username:password
```

The response includes a list of purge operations (this example is shortened to only two entries):

```
[
    {
        "guid": "d93c7664-6e41-4aa9-aed8-b740b985c9a0",
        "userName": "admin",
        "operation": "PURGE",
        "params": "[ac2772e8-984d-4ab6-9e99-323f1be2d3c0, 90231026-6581-4
168-8828-f010aa9b097c]",
        "startTime": 1576261685009,
        "endTime": 1576261685197,
        "clientId": "10.16.1.255",
        "result": "ac2772e8-984d-4ab6-9e99-323f1be2d3c0,ae143e74-48d4-4f4b
-8164-192bc842ed3b,dd742369-2a2c-44ee-902c-4d78a55b7100,90231026-6581-4168-8
828-f010aa9b097c"
    },
    {
        "guid": "b964079d-1f55-43f3-af7b-3a3701378826",
        "userName": "admin",
        "operation": "PURGE",
        "params": "[35b7aaad-2aaf-4af8-a043-e7b524e1314e]",
        "startTime": 1576028426951,
        "endTime": 1576028427165,
```

```
        "clientId": "10.16.1.255",
        "result": "35b7aaad-2aaf-4af8-a043-e7b524e1314e,add45b8a-4bef-4ebb-
a8d0-0b8b920f068d"
    },...
]
```

Related Information

Auditing purged entities

Apache Atlas technical metadata migration reference

This documentation includes an exhaustive reference of how Cloudera Navigator technical metadata is migrated into Atlas entities.

The migration process moves technical metadata from Navigator to Atlas in one of these ways:

- One-to-one mapping. There are no field mapping notes.
- Type conversion. The field mapping notes indicate the new type applied.
- Data conversion. The field mapping notes indicate how the data was converted, such as from a string to a Boolean value (for example, type=FILE to isFile=True).
- No reason to migrate. Navigator stored a value that has no use in Atlas, such as the system ID. The field mapping notes indicate that the value is not used in Atlas.
- No value in Navigator. In most cases, if an Atlas field does not have an equivalent in Navigator, the Atlas field is left as null. The migration notes indicate if a value is filled in by default.
- Not migrated. There is one case where metadata in Navigator is not migrated and potentially the information is lost: the Spark operation (spark_process) metadata for principal is not migrated to Atlas. The principal is migrated at the operation execution (spark_process_execution) entity level. The field mapping notes indicate this case.

All Atlas entities share "system" attributes. The mapping for these attributes is described once but apply to all entities.

Related Information

Mapping Navigator business metadata to Atlas

System metadata migration

All migrated entities in Atlas include the same top-level metadata attributes, such as name, description, and creation time.

The following sections describe how Navigator "common" entity metadata is mapped to Atlas "system" metadata. If Atlas requires metadata that wasn't available in Navigator, the migration notes describe how the Atlas metadata values are generated.

Navigator Metadata	Atlas Metadata	Migration Notes
created	createTime	Converted to date type.
deleted	status	If True in Navigator, Atlas status is set to "DELETED"; otherwise status is set to "ACTIVE".
description	attributes.userDescription	
extractorRunId		No equivalent in Atlas.
identity	guid	Converted to Atlas value
internalType	typeName	Converted to Atlas values.
lastModified	updateTime	Converted to date type.
lastModifiedBy	updatedBy	

Navigator Metadata	Atlas Metadata	Migration Notes
name	attributes.displayName	
originalName	attributes.name	
originalDescription	attributes.description	
owner	attributes.owner	
packageName		No equivalent in Atlas.
properties	customAttribues	Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
sourceId		Inferred rather than migrated.
technicalProperties	customAttribues	Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
tags	labels	
	attributes.qualifiedName	Atlas uniquely identifies entity instances with the qualified name. See the entity-specific migration reference to see how these values are generated.
	clusterName	Supplied in nav2atlas migration command.
	homeId	Not used currently in Atlas.
	isProxy	Not used currently in Atlas.
	provenanceType	Not used currently in Atlas.
	version	Not used currently in Atlas.

HDFS entity metadata migration

HDFS metadata entities are migrated from Navigator to Atlas when they appear in a lineage relationship from Hive, Impala, or Spark processes.

The following sections describe how metadata is mapped from Navigator to Atlas; if Atlas requires metadata that wasn't available in Navigator, the migration notes describe how the Atlas metadata values are generated.

Migrated entities include:

- HDFS Directory on page 25
- HDFS File on page 26

For entity metadata that is common to all entities, see System metadata migration on page 24.

HDFS Directory

Navigator fselement entities of type=DIRECTORY are migrated to Atlas hdfs_path entities with the isFile attribute set to false.

Navigator Metadata	Atlas Metadata	Migration Notes
blockSize		Null for directories.
created	attributes.createTime	Converted to date type.
ezKeyName		Null for directories.
fileSystemPath	attributes.path	
group	attributes.group	
lastAccessed	attributes.modifiedTime	Converted to date type.
mimeType		Null for directories.

Navigator Metadata	Atlas Metadata	Migration Notes
owner	attributes.owner	
permissions	attributes.posixPermissions	Converted to Atlas values.
replication	attributes.numberOfReplicas	Null for directories.
size	attributes.fileSize	Null for directories.
type	attributes.isFile	The Navigator type=DIRECTORY property is converted to isFile=FALSE.
	attributes.isSymLink	Defaults to FALSE.
	attributes.nameServiceId	Optional in Atlas.
	attributes.qualifiedName	Generated as a string in the format <path>@clustername.

HDFS File

Navigator fselement entities of type=FILE are migrated to Atlas hdfs_path entities with the isFile attribute set to true.

Navigator Metadata	Atlas Metadata	Migration Notes
blockSize		Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
created	attributes.createTime	Converted to date type.
ezKeyName		Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
fileSystemPath	attributes.path	
group	attributes.group	
lastAccessed	attributes.modifiedTime	Converted to date type.
mimeType		Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
owner	attributes.owner	
parentPath	attributes.extendedAttributes	
permissions	attributes.posixPermissions	Converted to Atlas values.
replication	attributes.numberOfReplicas	
size	attributes.fileSize	
type	attributes.isFile	The Navigator type=FILE property is converted to isFile=TRUE.
	attributes.isSymLink	Defaults to FALSE.
	attributes.nameServiceId	Optional in Atlas.
	attributes.qualifiedName	Generated as a string in the format <path>@clustername.

Hive entity metadata migration

Hive metadata entities are fully migrated from Navigator to Atlas.

The following sections describe how metadata is mapped from Navigator to Atlas; if Atlas requires metadata that wasn't available in Navigator, the migration notes describe how the Atlas metadata values are generated.

Migrated entities include:

- Hive Database on page 27
- Hive Table on page 27
- Hive View on page 28

- Hive Storage Description on page 28
- Hive Column on page 28
- Hive Process on page 28
- Hive Column Lineage on page 29
- Hive Process Execution on page 29

Hive Database

Navigator hv_database entities are migrated to Atlas hive_db entities.

Navigator Metadata	Atlas Metadata	Migration Notes
fileSystemPath	attributes.location	
firstClassParentId	Not needed in Atlas	
params	attributes.parameters	
parentPath		Not needed in Atlas.
technicalProperties	customAttributes	
type		Inferred rather than migrated.
	attributes.ownerType	
	attributes.parameters	
	attributes.qualifiedName	Generated as a string in the format dbname@clustername.

Hive Table

Navigator hv_table entities are migrated to Atlas hive_table entities.

Navigator Metadata	Atlas Metadata	Migration Notes
clusterByColNames	bucketCols	Not needed in Atlas.
group	attributes.parameters	Added to Atlas entity as a key value pair with the Navigator name as the key.
params	attributes.parameters	Added to Atlas entity as a key value pair with the Navigator name as the key.
partColNames	relationshipAttributes.partitionKeys	
sortByColName	attributes.sortCols	Converted from string to array type.
technicalProperties	attributes.parameters	Added to the Atlas entity attributes as a key value pair with the Navigator name as the key.
	attributes.aliases	Defaults to null.
	attributes.comment	Defaults to null.
	attributes.lastAccessTime	Defaults to null.
	attributes.qualifiedName	Generated as a string in the format <parent_db>.<tablename>@<clustername>.
	attributes.retention	Defaults to null.
	attributes.tableType	Defaults to null.
	attributes.temporary	Defaults to null.
	attributes.viewOriginalText	Defaults to null.
	attributes.viewExpandedText	Defaults to null.

Hive View

Navigator hv_view entities are migrated to Atlas hive_table entities. Atlas does not distinguish between Hive tables and Hive views.

Hive Storage Description

Atlas includes a separate entity that represents how Hive table data is stored. Navigator included this metadata as part of its hv_table entity and the logical-physical lineage relationship. The migration creates the Atlas hive_storagedesc entity using metadata from the HMS table information.

Navigator Metadata	Atlas Metadata	Migration Notes
compressed	attributes.compressed	
fileSystemPath	attributes.location	
inputFormat	attributes.inputFormat	
outputFormat	attributes.outputFormat	
partColNames	attributes.bucketColNames	
serdeLibName	attributes.serdeInfo.serializationLib	
serdeProps	attributes.serdeInfo	
sortByColNames	attributes.sortCols	Converted from string to array type.
	attributes.numBuckets	
	attributes.parameters	
	attributes.qualifiedName	Generated as a string in the format <parent_db>.<tablename>@<clustername>_storage.
	attributes.sortedAsSubDirectories	

Hive Column

Navigator hv_column entities are migrated to Atlas hive_column entities. Note that the Atlas owner value is not available from Navigator and remains blank.

Navigator Metadata	Atlas Metadata	Migration Notes
dataType	attributes.type	
firstClassParentId		Not used in Atlas.
fieldIndex	attributes.position	
parentPath		Not used in Atlas.
	attributes.comment	Defaults to null.
	attributes.owner	Defaults to null.
	attributes.qualifiedName	Generated as a string in the format <parent_db>.<tablename>.<columnname>@<clustergname>.

Hive Process

Navigator hv_query entities are migrated to Atlas hive_process entities.

Navigator Metadata	Atlas Metadata	Migration Notes
inputs	inputs	Points to the input entities as relationship attributes.
outputs	outputs	Points to the output entities as relationship attributes.

Navigator Metadata	Atlas Metadata	Migration Notes
queryHash		Not used in Atlas.
queryText	attributes.queryText	Not used currently in Atlas.
sourceId		Not used in Atlas.
unparsed		Not used in Atlas.
wfIds		Not used in Atlas.
	attributes.startTime	Not used currently in Atlas.
	attributes.endTime	Not used currently in Atlas.
	attributes.userName	Not used currently in Atlas.
	attributes.operationType	Defaults to null.
	attributes.qualifiedName	Generated as a string with the operation, input entities, and output entities, where each entity is noted by <asset_qualifiedName>:<createTime> and entries are separated by colons, and an arrow shows the break between input and output entities. For example: `dbblue.table_aqua@clustercolor:1589373411000->dbblue.table_teal@clustercolor:1589394039000`
	attributes.queryId	Defaults to null.
	attributes.queryGraph	Defaults to null.
	attributes.recentQueries	Defaults to null.

Hive Column Lineage

Navigator hv_query_part entities are migrated to Atlas hive_column_lineage entities.

Navigator Metadata	Atlas Metadata	Migration Notes
inputs	attributes.inputs	Points to the input column entities as relationship attributes.
outputs	attributes.outputs	Points to the output column entities as relationship attributes.
firstClassParentId	attributes.query	Points to the parent hive_process entity as a relationship attribute.
originalName	attributes.qualifiedName	Generated as a string with the operation, input entities, output entities, and target column name, where each entity is noted by <column_qualifiedName>:<createTime> and entries are separated by colons, and an arrow shows the break between input and output entities. For example: `dbblue.table_aqua@clustercolor:1589373411000->dbblue.table_teal@cm:1589390050000:column_beach`
	attributes.dependencyType	Set to "SIMPLE".
	attributes.expression	Defaults to null.

Hive Process Execution

Navigator hv_query_execution entities are migrated to Atlas hive_process_execution entities.

Navigator Metadata	Atlas Metadata	Migration Notes
inputs	inputs	Points to the input entities as relationship attributes.
outputs	outputs	Points to the output entities as relationship attributes.

Navigator Metadata	Atlas Metadata	Migration Notes
ended	attributes.endTime	
operation	attributes.process	Points to the parent hive_process entity as a relationship attribute.
originalName	attributes.queryText	
principal	attributes.userName	
started	attributes.startTime	
	attributes.hostName	Defaults to null.
	attributes.qualifiedName	Generated as a string with the operation, input entities, output entities, and execution start and end timestamps, where each entity is noted by <asset_qualifiedName>:<creat eTime> and entries are separated by colons, and an arrow shows the break between input and output entities. For example: `dbblue.table_aqua@clustercolor:1589373411000 ->dbblue.table_teal@clustercolor:15893900500 00:1589394047386:1589394064097`
	attributes.queryGraph	Defaults to null.
	attributes.queryId	Defaults to null.
	attributes.queryPlan	Set to "Not Supported".

Impala entity metadata migration

Impala metadata entities are fully migrated from Navigator to Atlas.

The following sections describe how metadata is mapped from Navigator to Atlas; if Atlas requires metadata that wasn't available in Navigator, the migration notes describe how the Atlas metadata values are generated.

Migrated entities include:

- Impala Process on page 30
- Impala Column Lineage on page 31
- Impala Process Execution on page 31

For entity metadata that is common to all entities, see System metadata migration on page 24.

Impala Process

Navigator impala_operation entities are migrated to Atlas impala_process entities.

Navigator Metadata	Atlas Metadata	Migration Notes
inputs	inputs	Points to the input entities as relationship attributes.
outputs	outputs	Points to the output entities as relationship attributes.
queryHash		Not used in Atlas.
queryText	attributes.queryText	Not used currently in Atlas (see Impala Process Execution on page 31).
sourceId		Not used in Atlas.
unparsed		Not used in Atlas.
wfIds		Not used in Atlas.
	attributes.endTime	Not used currently in Atlas.
	attributes.operationType	Defaults to null.

Navigator Metadata	Atlas Metadata	Migration Notes
	attributes.qualifiedName	Generated as a string with the input entities and output entities, where each entity is noted by <asset_qualifiedName>:<createTime> and entries are separated by colons, and an arrow shows the break between input and output entities. For example: `dbblue.table_aqua@clustercolor:1589373411000->dbblue.table_teal@clustercolor:1589390050000`
	attributes.queryGraph	Defaults to null.
	attributes.queryId	Defaults to null.
	attributes.recentQueries	Defaults to null.
	attributes.startTime	Not used currently in Atlas.
	attributes.userName	Not used currently in Atlas.

Impala Column Lineage

Navigator impala_sub_operation entities are migrated to Atlas impala_column_lineage entities.

Navigator Metadata	Atlas Metadata	Migration Notes
inputs	attributes.inputs	Points to the input column entities as relationship attributes.
outputs	attributes.outputs	Points to the output column entities as relationship attributes.
firstClassParentId	attributes.query	Points to the parent impala_process entity as a relationship attribute.
originalName	attributes.qualifiedName	Generated as a string with the input entities, output entities, and target column name, where each entity is noted by <column_qualifiedName>:<createTime> and entries are separated by colons, and an arrow shows the break between input and output entities. For example: `dbblue.table_aqua@clustercolor:1589373411000->dbblue.table_teal@cm:1589390050000:column_beach`
	attributes.dependencyType	Set to "SIMPLE".
	attributes.expression	Defaults to null.

Impala Process Execution

Navigator impala_operation_execution entities are migrated to Atlas impala_process_execution entities.

Navigator Metadata	Atlas Metadata	Migration Notes
inputs	inputs	Points to the input entities as relationship attributes.
outputs	outputs	Points to the output entities as relationship attributes.
ended	attributes.endTime	
operation	attributes.process	Points to the parent impala_process entity as a relationship attribute.
originalName	attributes.queryText	
principal	attributes.userName	
started	attributes.startTime	
	attributes.hostName	Defaults to null.

Navigator Metadata	Atlas Metadata	Migration Notes
	attributes.qualifiedName	Generated as a string with the input entities, output entities, and execution start and end timestamps, where each entity is noted by <asset_qualifiedName>:<createTime> and entries are separated by colons, and an arrow shows the break between input and output entities. For example: `dbblue.table_aqua@clustercolor:1589373411000 ->dbblue.table_teal@clustercolor:1589390050000:1589390056000:1589390058000`
	attributes.queryGraph	Defaults to null.
	attributes.queryId	Defaults to null.
	attributes.queryPlan	Set to "Not Supported".

Spark entity metadata migration

Spark metadata entities are fully migrated from Navigator to Atlas.

The following sections describe how metadata is mapped from Navigator to Atlas; if Atlas requires metadata that wasn't available in Navigator, the migration notes describe how the Atlas metadata values are generated.

Migrated entities include:

- Spark Process on page 32
- Spark Process Execution on page 32

For entity metadata that is common to all entities, see System metadata migration on page 24.

Spark Process

Navigator spark_operation entities are migrated to Atlas spark_process entities.

Notice: The metadata for the principal from Navigator is not migrated to Atlas.

Navigator Metadata	Atlas Metadata	Migration Notes
inputs	relationshipAttributes.inputs	Points to the input directories as relationship attributes.
outputs	relationshipAttributes.outputs	Points to the output directory as a relationship attribute.
principal		Not mapped.
	attributes.currUser	Defaults to null.
	attributes.details	Defaults to null.
	attributes.executionId	Defaults to null.
	attributes.remoteUser	Defaults to null.
	attributes.sparkPlanDescription	
	attributes.qualifiedName	Generated as a string in the format process_id@clustername. For example, application_1589303388872_0001@clustercolor.20324.

Spark Process Execution

Navigator spark_operation_execution entities are migrated to Atlas spark_process_execution entities.

Navigator Metadata	Atlas Metadata	Migration Notes
inputs	relationshipAttributes.inputs	Points to the input files as relationship attributes.

Navigator Metadata	Atlas Metadata	Migration Notes
outputs	relationshipAttributes.outputs	Points to the output files as relationship attributes.
ended	attributes.endTime	
operation	attributes.process	Points to the parent spark_process entity as a relationship attribute.
originalName	attributes.queryText	
principal	attributes.userName	
started	attributes.startTime	
	attributes.hostName	Defaults to null.
	attributes.qualifiedName	Generated as a string in the format process_id-exec.timestamp@clustermame. For example, application_1589303388872_0001-exec.1589331800580@clustercolor.

AWS S3 entity metadata migration

S3 metadata entities are migrated from Navigator to Atlas.

The following sections describe how metadata is mapped from Navigator to Atlas; if Atlas requires metadata that wasn't available in Navigator, the migration notes describe how the Atlas metadata values are generated.

Migrated entities include:

- S3 Bucket on page 33
- S3 Object: Directory on page 33
- S3 Object: File on page 34

For entity metadata that is common to all entities, see System metadata migration on page 24.

S3 Bucket

Navigator s3_bucket entities are migrated to Atlas aws_s3_v2_bucket entities.

Navigator Metadata	Atlas Metadata	Migration Notes
encryption	attributes.encryption	
eTag	attributes.eTag	
owner	attributes.owner	
ownerId	attributes.ownerId	
region	attributes.region	
	attributes.qualifiedName	Generated as a string in the format <bucket_name>@<cluster_name>. For example yellow_bucket@clustercolor.

S3 Object: Directory

Navigator s3_object entities of type=DIRECTORY is converted to aws_s3_v2_directory entities.

Navigator Metadata	Atlas Metadata	Migration Notes
bucketName	attributes.bucketName	
depth	customAttributes	Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
encryption	attributes.encryption	
eTag	attributes.eTag	

Navigator Metadata	Atlas Metadata	Migration Notes
fileSystemPath	attributes.storageLocation	
firstClassParentId	relationshipAttributes.container	Points to the parent directory or bucket as a relationship attribute.
implicit	attributes.implicit	
newObject	customAttributes	Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
owner	attributes.owner	
ownerId	attributes.ownerId	
parentPath	attributes.objectPrefix	The Atlas value is derived from the Navigator value (no one-to-one migration).
region	attributes.region	
sequencer	customAttributes	Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
size	attributes.size	
storageClass	attributes.storageClass	.
type		Used to determine the Atlas entity type.
	attributes.qualifiedName	Generated as a string in the format <object_prefix>://<bucket_name>/<directory_name>@<cluster_name>. For example s3a://yellow_bucket/hive_storage_color_table_dir@clustercolor.

S3 Object: File

Navigator s3_object entities of type=file is converted to aws_s3_v2_object entities.

Navigator Metadata	Atlas Metadata	Migration Notes
bucketName	attributes.bucketName	
depth	customAttributes	Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
encryption	attributes.encryption	
eTag	attributes.eTag	
fileSystemPath	customAttributes	Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
firstClassParentId	relationshipAttributes.container	Points to the parent directory or bucket as a relationship attribute.
implicit	attributes.implicit	
newObject	customAttributes	Added to the Atlas entity custom attributes as a key value pair with the Navigator name as the key.
owner	attributes.owner	
ownerId	attributes.ownerId	
parentPath	attributes.objectPrefix	The Atlas value is derived from the Navigator value (no one-to-one migration).
region	attributes.region	
size	attributes.size	
storageClass	attributes.storageClass	
type		Used to determine the Atlas entity type.

Navigator Metadata	Atlas Metadata	Migration Notes
	attributes.qualifiedName	Generated as a string in the format <object_prefix>://<bucket_name>/<directory_name>/<file_name>@<cluster_name>. For example s3a://yellow_bucket/hive_storage_color_table_dir/.hive-staging_hive_2020-03-20_01-43-14_862_995309273321986325-1@clustercolor.

NiFi metadata collection

Atlas collects metadata of events from NiFi clusters to illustrate the high-level relationships among processes and data sets and potentially detailed NiFi event-level lineage.

The metadata connector between NiFi and Atlas is implemented as the NiFi reporting task ReportLineageToAtlas.

This reporting task stores two types of NiFi flow information:

- NiFi flow structure describes the components running within a NiFi flow and how they are connected.
- NiFi data lineage describes how NiFi flows interact with different data assets such as HDFS files or Hive tables. Lineage is determined by analyzing NiFi provenance events.

When the Atlas reporting task runs in a NiFi cluster, the primary node performs the one-time task of inserting the NiFi-specific entity types in Atlas. The primary node sends metadata to Atlas as and when an event occurs in the NiFi cluster. Every node (including primary node) analyzes NiFi provenance events stored in a provenance event repository to create lineage between NiFi and other data assets such as Hive tables and HDFS paths.

The NiFi reporting tasks provide a number of properties to control what metadata is collected and at what level of NiFi events are described in lineage. The implementation details to be aware of are:

1. Mapping the NiFi cluster to a cluster name identifier in the Atlas environment.

   You must specifically note the following:

   - Which cluster name should be used to represent the current NiFi cluster?
   - When NiFi components perform operations on data assets, which cluster does the data asset reside in?

   The mapping can be defined by Dynamic Properties with a name in the 'hostnamePattern.ClusterName' format, having its value as a set of Regular Expression Patterns to match IP addresses or host names to a particular cluster name.

   This mapping is described in detail in Cluster Hostname Resolution in the Apache NiFi documentation.
2. Level of lineage graph detail.

   NiFi Lineage Strategy

Choose a strategy for showing lineage information in Atlas by using one of the following:

- Simple Path

A high-level view of NiFi processes that summarizes how each data asset and NiFi process are related when the data flows among services. Specifically, this strategy maps data I/O provenance events such as SEND/RECEIVE to 'nifi_flow_path' created by NiFi flow structure analysis. The result is that if different data assets go through the same 'nifi_flow_path', the lineage picture gives the impression that there is a relationship among the data assets because they are processed in the same flow. No information is lost; however, as you can use the NiFi provenance events to investigate the details.

This strategy generates the least amount of data in Atlas.

- Complete Path

Lineage is created by traversing provenance events backwards from a DROP event, reporting the entire lineage for a given FlowFile including where it is created, and where it goes. This strategy generates more detail and shows separate lineage pictures for data sets that are not related but happen to be processed through the same flow path.

However, reporting complete flow paths for every single FlowFile produces too many entities in Atlas and needs more computing resources to collect the relevant metadata.

Choose the strategy before implementing lineage collection in a production environment because switching among strategies can leave you with residual entities shown in lineage pictures and the lineage pictures may show different entities for otherwise equivalent processes.

About Namespaces

An entity in Atlas can be identified either by its GUID for any existing objects, or type name and unique attribute if GUID is not known. A qualified name is commonly used as the unique attribute.

One Atlas instance can be used to manage multiple environments, and objects in different environments may have the same name. For example, a Hive table 'request_logs' in two different clusters, 'cluster-A' and 'cluster-B'. For this reason the qualified names contain a so-called metadata namespace.

It is common practice to provide the cluster name as the namespace, but the cluster name it can be any arbitrary string.

With this, a qualified name has a 'componentId@namespace' format. For example, a Hive table qualified name is dbName.tableName@namespace (default.request_logs@cluster-A).

From this NiFi reporting task standpoint, a namespace is needed to be resolved in the following situations:

- To register NiFi component entities. Which namespace should be used to represent the current NiFi environment?
- To create lineages from NiFi components to other datasets. Which environment does the dataset reside in?

To answer such questions, ReportLineageToAtlas reporting task provides a way to define mappings from IP address or hostname to a namespace. The mapping can be defined by Dynamic Properties with a name in the 'hostnamePattern.namespace' format, having its value as a set of Regular Expression Patterns to match IP addresses or host names to a particular namespace.

As an example, following mapping definition resolves namespace 'namespace-A' for IP address such as '192.168.30.123' or hostname 'namenode1.a.example.com', and 'namespace-B' for '192.168.40.223' or 'nifi3.b.example.com'.

```
# Dynamic Property Name for namespace-A

hostnamePattern.namespace-A

# Value can have multiple Regular Expression patterns separated by new    line

192\.168\.30\.\d+

[^\.]+\.a\.example\.com

# Dynamic Property Name for namespace-B

hostnamePattern.namespace-B

# Values

192\.168\.40\.\d+

[^\.]+\.b\.example\.com
```

If no namespace mapping matches, then a name defined at 'Atlas Default Metadata Namespace' is used.

How Lineage strategy works

To illustrate the difference between lineage strategies, consider a sample NiFi flow.

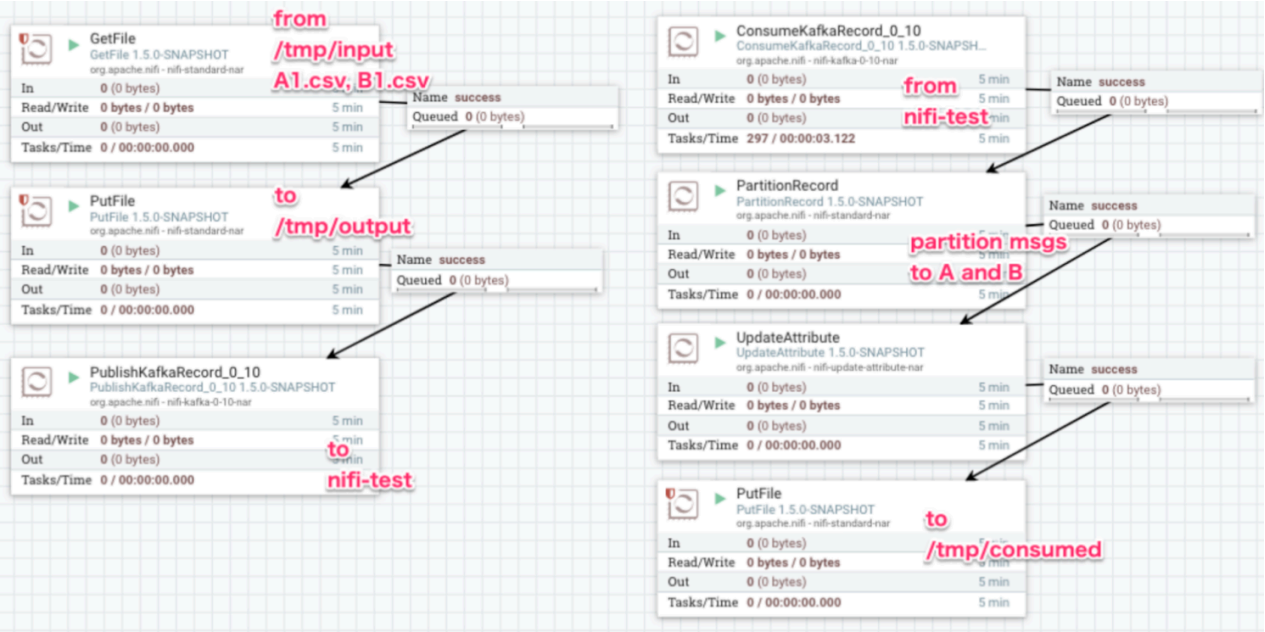

With 'Simple Path', Atlas lineage is reported as seen in the following image, when '/tmp/input/A1.csv' is selected.

Because 'Simple Path' maps I/O events to a 'nifi_flow_path', '/tmp/output/B1.csv' is shown in the lineage graph as the file that is written by the 'GetFile, PutFile...' process.

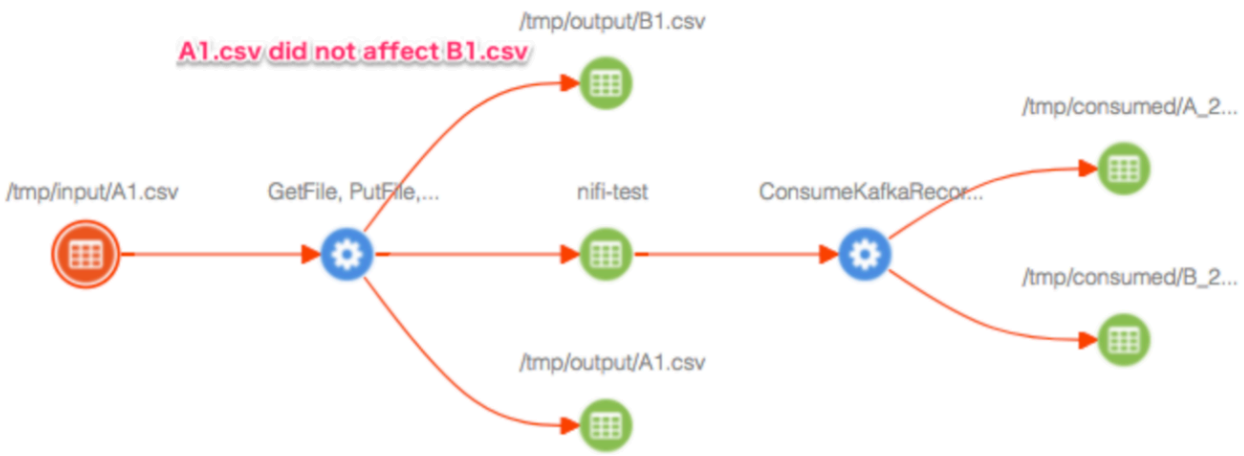

With 'Complete Path', Atlas lineage is reported as seen in the following image. In this use case, 'GetFile, PutFile...' process is not linked to '/tmp/output/B1.csv'.

The 'Complete Path' strategy creates two different 'nifi_flow_path' entities. One for '/tmp/input/A1.csv -> /tmp/output/A1.csv' and another for '/tmp/input/B1.csv -> /tmp/output/B1.csv'.

However, after the data records ingest from A.csv and B.csv into a bigger dataset, 'nifi-test' Kafka topic in the following example (or whatever dataset such as a database table or a concatenated file ... and so on), record level lineage tracking about where it came from can no longer be tracked.

The resulting '/tmp/consumed/B_2..' is shown in the same lineage graph, although the file does not contain any data that came from '/tmp/input/A1.csv'.

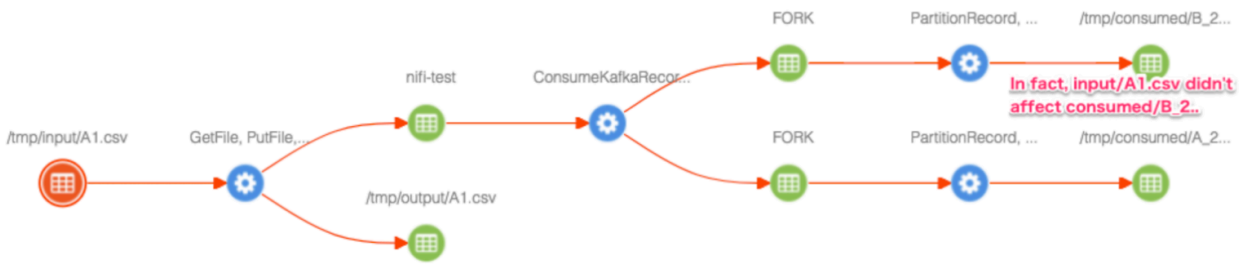

Understanding the data that flow into Atlas

The reporting task stores two types of NiFi flow information.

- NiFi flow structure
- NiFi data lineage

'NiFi flow structure' indicates what components are running within a NiFi flow and how these are connected. It is reported by analyzing current NiFi flow structure, specifically NiFi component relationships.

'NiFi data lineage' indicates what part of NiFi flow interacts with different datasets such as HDFS files or Hive tables. It is reported by analyzing NiFi provenance events.

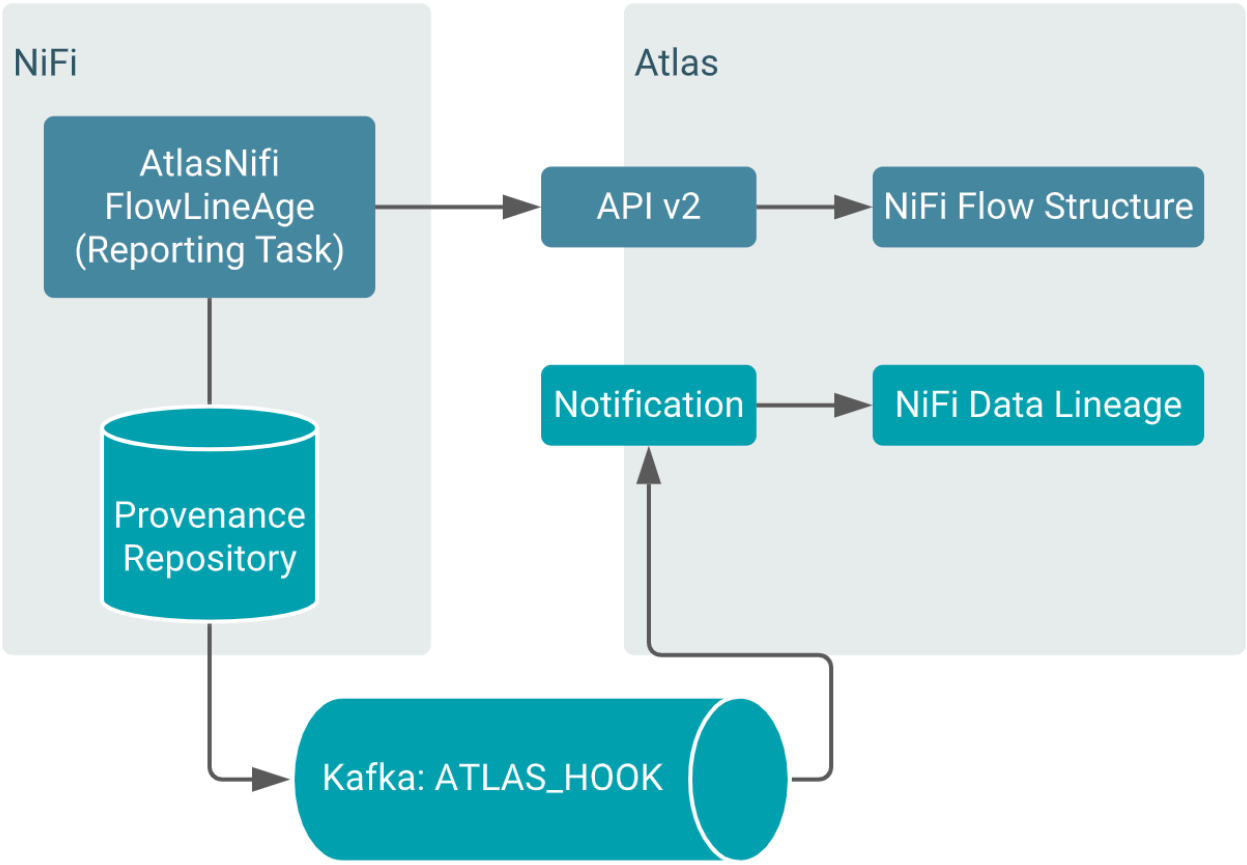

NiFi lineage

Atlas collects metadata from NiFi to represent the lineage among data assets in the Lineage tab.

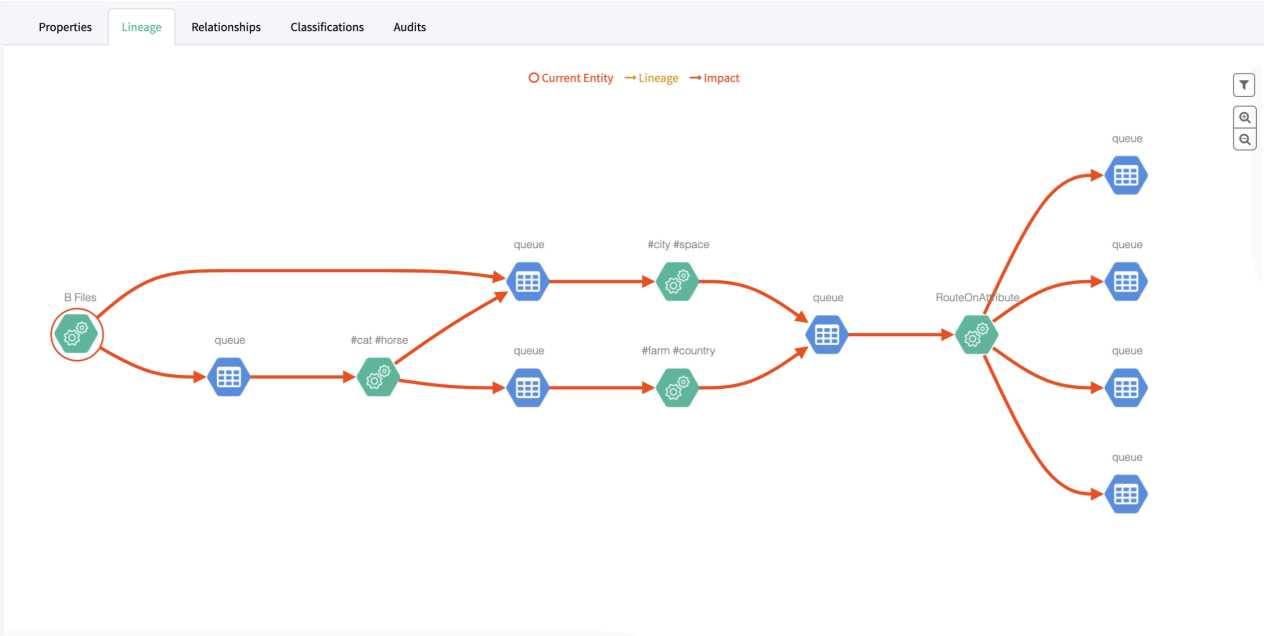

The Atlas lineage graph shows the input and output processes that the current entity participated in, specifically those relationships modeled as “inputToProcesses” and “outputFromProcesses.” Entities are included if they are provided as an input to processes that lead to the current entity or they are output from processes for which the current entity is an input. NiFi processes follow this pattern.

Atlas NiFi relationships

Atlas shows the related entities in the Relationships tab in the Dashboard.

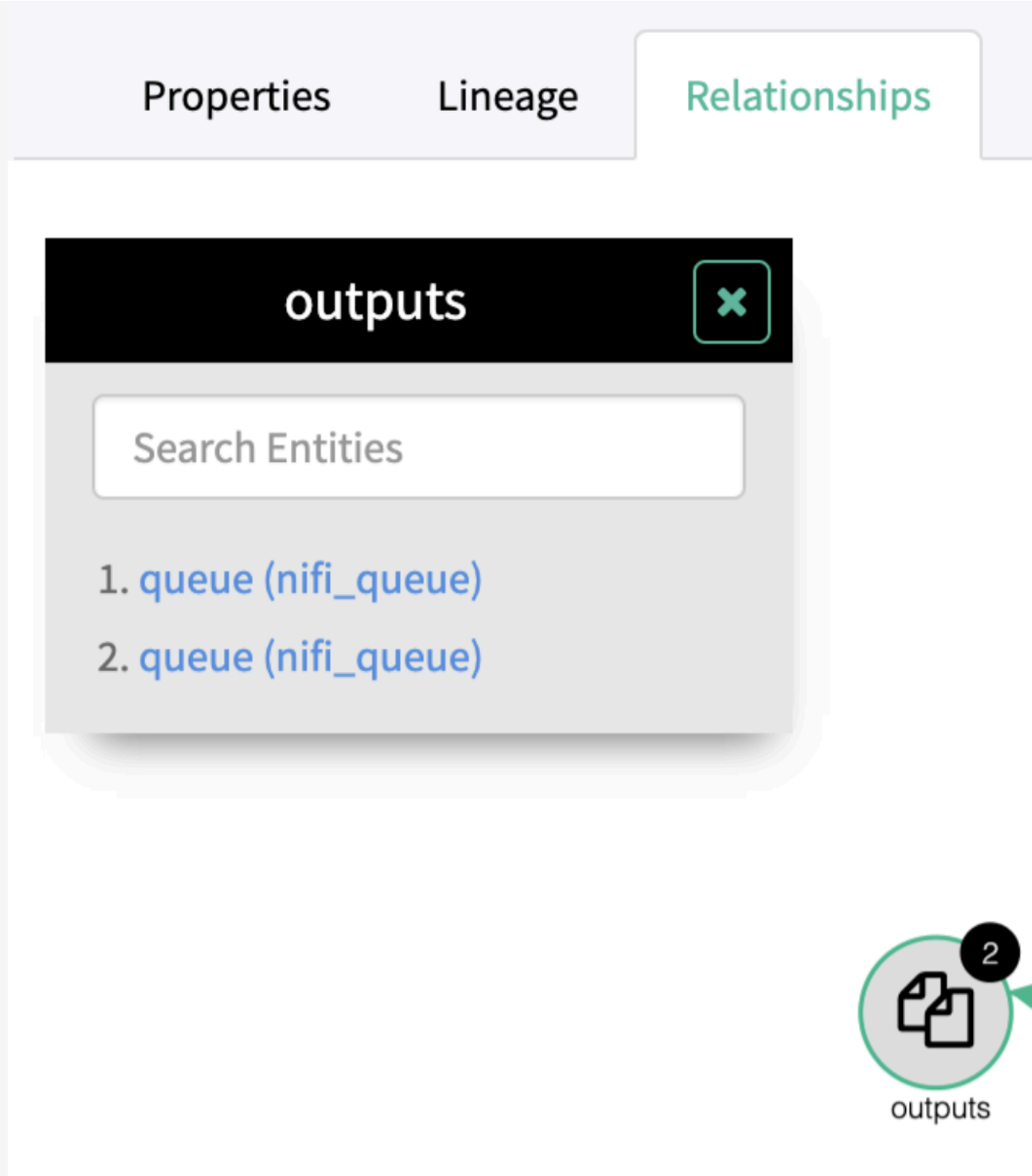

The Relationship tab shows the relationships that exist for an entity. Use this view to navigate among related entities.

Atlas NiFi audit entries

Atlas lists changes to metadata entities in the Audits tab in the Dashboard.

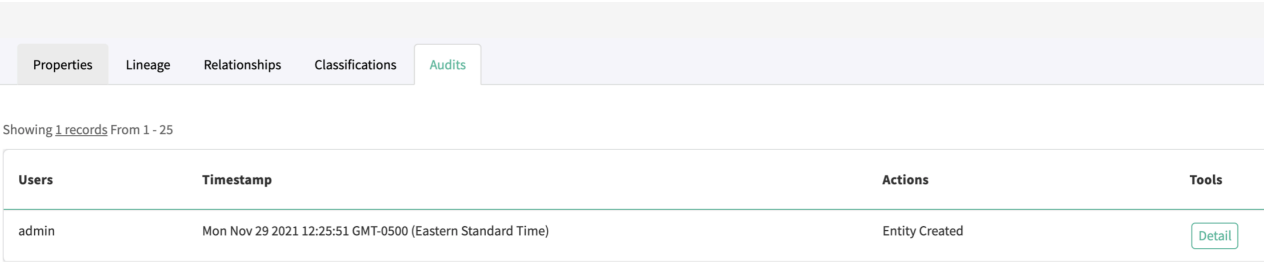

Atlas tracks the lifecycle of each NiFi entity, including its creation and updation. User access and actions that affect the data content of the source asset are not included in the audit.

Note: If the data is active while the entity is created, it displays some additional information. You must be aware that after the entity is created, the flow files do not create an event, unless the processor is listed.

How the reporting task runs in a NiFi cluster

When the reporting task runs in a NiFi cluster, the following tasks are executed only by the primary node:

- Create NiFi Atlas types in the Atlas type system.
- Maintain NiFi flow structure and metadata in Atlas both which consists of NiFi component entities such as 'nifi_flow', 'nifi_flow_path', and 'nifi_input(output)_port'.

Every node (including primary node) analyzes NiFi provenance events stored in a provenance event repository, to create lineage between 'nifi_flow_path' and other dataset (for example: Hive tables and HDFS path).

NiFi Atlas Types

The reporting task creates the following NiFi specific types in Atlas Type system if these type definitions are not found.

Green boxes represent sub-types of dataset and blue ones are sub-types of Process. Gray lines represent entity ownership. Red lines represent lineage.

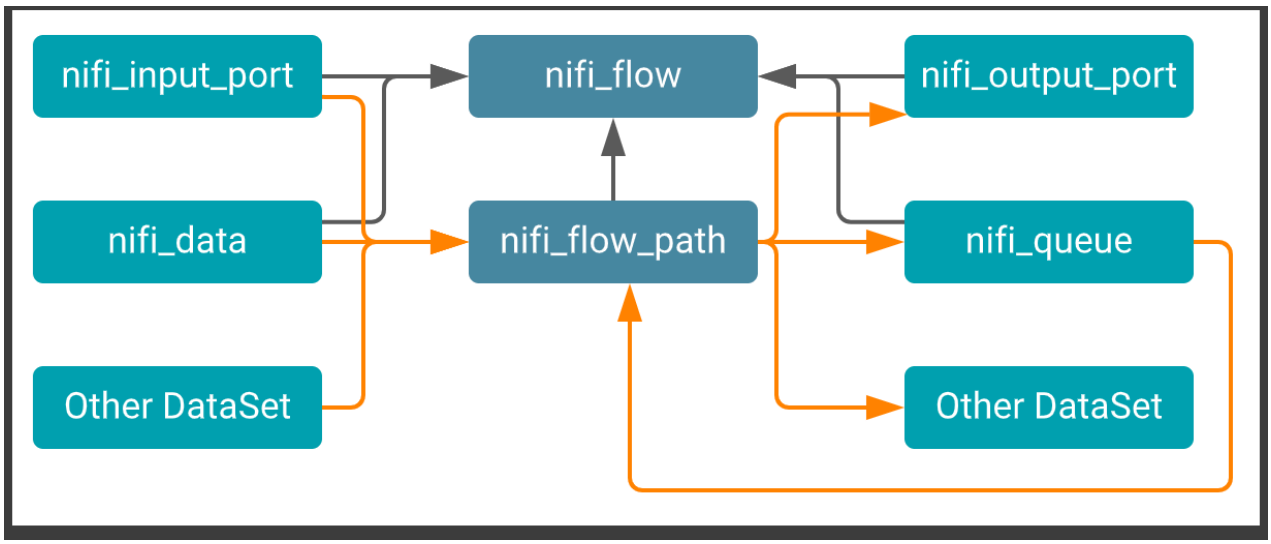

Nifi_flow: Represents a NiFi data flow. As shown in the diagram, nifi_flow owns other nifi_component types. This owning relationship is defined by Atlas 'owned' constraint so that when a 'nifi_flow' entity is removed, all owned NiFi component entities are removed in a cascading manner.

When the reporting task runs, it analyzes and traverses the entire flow structure, and creates NiFi component entities in Atlas. During later runs, it compares the current flow structure with the one stored in Atlas to figure out if any changes have been made since the last time the flow was reported. The reporting task updates NiFi component entities in Atlas if needed.

NiFi components that are removed from a NiFi flow also get deleted from Atlas. However those entities can be seen in Atlas search results or lineage graphs because Atlas uses 'Soft Delete' by default.

- Attributes:
  - qualifiedName: Root ProcessGroup ID@namespace (For example, 86420a14-2fab-3e1e-4331-fb6ab42f58e0@ns1)
  - name: Name of the Root ProcessGroup.
  - url: URL of the NiFi instance. This can be specified using the reporting task 'NiFi URL for Atlas' property.

Nifi_flow_path: Part of a NiFi data flow containing one or more processing NiFi components such as Processors and RemoteGroupPorts. The reporting task divides a NiFi flow into multiple flow paths.

- Attributes:
  - qualifiedName: The first NiFi component Id in a path@namespace (for example: 529e6722-9b49-3b66-9c94-00da9863ca2d@ns1)
  - name: NiFi component names within a path are concatenated (for example: GenerateFlowFile, PutFile, LogAttribute)
  - url: A deep link to the first NiFi component in corresponding NiFi UI

nifi_input/output_port: Represents a RootGroupPort which can be accessed by RemoteProcessGroup through Site-to-Site protocol.

- Attributes:
  - qualifiedName: Port ID@namespace (for example: 3f6d405e-6e3d-38c9-c5af-ce158f8e593d@ns1)
  - name: Name of the Port.

Nifi_data: Represents unknown datasets created by CREATE/SEND/RECEIVE NiFi provenance events that do not have a particular provenance event analyzer.

- Attributes:
  - qualifiedName: ID of a Processor which generated the provenance event@namespace (for example: db8bb12c-5cd3-3011-c971-579f460ebedf@ns1)
  - name: Name of the Processor.

Nifi_queue: An internal dataset of NiFi flows which connects nifi_flow_paths. Atlas lineage graph requires a dataset in between Process entities.

- Attributes:
  - qualifiedName: ID of the first Processor in the destination nifi_flow_path.
  - name: Name of the Processor.

Analysing event analysis

To create lineage describing which NiFi component interacts with what datasets, dataset entity, and Process entity need to be created in Atlas. Specifically, at least three entities are required to draw a lineage graph on Atlas UI.

A Process entity, and a dataset which is referred by a Process 'inputs' attribute, and a dataset referred from an 'outputs' attribute. For example:

```
# With following entities
            guid: 1
            typeName: fs_path (extends dataset)
            qualifiedName: /data/A1.csv@BranchOffice1
            guid: 2
            typeName: nifi_flow_path (extends Process)
            name: GetFile, PutHDFS
            qualifiedName: 529e6722-9b49-3b66-9c94-00da9863ca2d@BranchOff
ice1
            inputs: refer guid(1)
            outputs: refer guid(3)

            guid: 3
            typeName: hdfs_path (extends dataset)
            qualifiedName: /data/input/A1.csv@Analytics
            # Atlas draws lineage graph
            /data/A1.csv -> GetFile, PutHDFS -> /data/input/A1.csv
```

To identify such Process and dataset Atlas entities, this reporting task uses NiFi Provenance Events. At the minimum, the reporting task needs to derive the following details from a NiFi Provenance event record:

- typeName (for example: fs_path, hive_table)
- qualifiedName in uniqueId@namespace (for example: /data/A1.csv@ns1)

'namespace' in 'qualifiedName' attribute is resolved by mapping an IP address or hostname available at the NiFi Provenance event 'transitUri' to a namespace.

For 'typeName' and 'qualifiedName', different analysis rules are needed for different datasets. ReportLineageToAtlas provides an extension point called 'NiFiProvenanceEventAnalyzer' to implement such analysis logic for particular datasets.

When a Provenance event is analyzed, registered NiFiProvenanceEventAnalyzer implementations are searched in the following order to find a best matching analyzer implementation:

1. By component type (for example: KafkaTopic)
2. By transit URI protocol (for example: HDFSPath)
3. By event type, if none of above analyzers matches (for example: Create)

Limitations of Atlas-NiFi integration

Some limitations of this integration.

- Requires Atlas 0.8-incubating or later. The reporting task requires Atlas REST API version 2, which is introduced in Atlas 0.8-incubating. Older versions of Atlas are not supported.
- Supports limited datasets and Processors. To report lineage to Atlas, the reporting task should be familiar with what a given processor does with a certain dataset. Later, create an 'Atlas Object Id' for a dataset which uniquely identifies an entity in Atlas. Atlas Object ID has a unique properties map, and mostly 'qualifiedName' is set in the unique properties map to identify an entity. The format of a qualifiedName depends on each dataset. To create this Atlas Object ID, you must implement Processor-specific code that analyzes configured properties.
- Requires restart of NiFi to update some ReportingTask properties. As the underlying Atlas client library caches configurations when it runs the first time, some properties of this reporting task can not be updated by stopping, configuring, and restarting the reporting task. The NiFi process needs to be restarted in such cases.

HiveServer metadata collection

Atlas can collect metadata from HiveServer, including queries and the data assets the queries affect.

An Atlas hook runs in each HiveServer instance. This hook sends metadata to Atlas for both Hive operations and Hive data assets. Operations are represented by process and process execution entities in Atlas. Hive databases, tables, views, and columns are represented by entities in Atlas. When a Hive operation involves files, the metadata for the file system and files are represented in Atlas as file system paths.

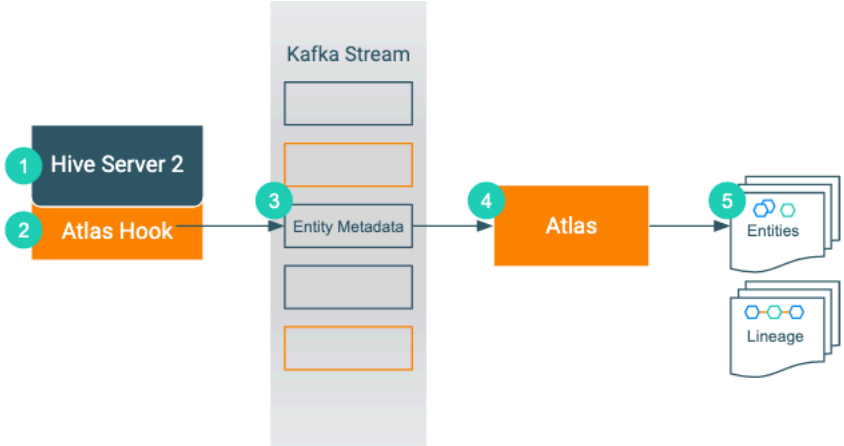

1. When an action occurs in the HiveServer instance...
2. The corresponding Atlas hook collects information for the action into metadata entities.
3. The hook publishes the metadata on a Kafka topic.
4. Atlas reads the message from the topic and determines what information will create new entities and what information updates existing entities.
5. Atlas creates and updates the appropriate entities and determines lineage from existing entities to the new entities.

The Atlas bridge for HBase pulls the same metadata as the hook, but instead of sending the metadata through Kafka, it passes message in bulk in an API call. The bridge creates entities in Atlas for all of the existing HBase namespaces, tables, columns, and column families.

HiveServer actions that produce Atlas entities

Operations that create, update, or delete Hive metadata will affect Atlas entities; operations that only affect data do not show up in Atlas.

The following table lists the HiveServer actions that produce or update metadata in Atlas.

This Action in HiveServer...	...Produces metadata for these Atlas entities
ALTER DATABASE, CREATE DATABASE, DROP DATABASE	hive_db, hive_db_ddl
ALTER TABLE, CREATE TABLE, CREATE TABLE AS SELECT, DROP TABLE	hive_process, hive_process_execution, hive_table, hive_table_ddl, hive_column, hive_column_lineage, hive_storagedesc, hdfs_path

This Action in HiveServer...	...Produces metadata for these Atlas entities
ALTER VIEW, ALTERVIEW_AS_SELECT, CREATE VIEW, CREATE VIEW AS SELECT, DROP VIEW	hive_process, hive_process_execution, hive_table, hive_column, hive_column_lineage, hive_table_ddl
INSERT INTO (SELECT), INSERT OVERWRITE	hive_process, hive_process_execution

Notable actions in HiveServer that do NOT produce process or process execution entities in Atlas, meaning that no lineage is produced for these operations:

- SELECT

HiveServer entities created in Atlas

Each HiveServer entity in Atlas includes detailed metadata collected from Hive.

The following diagrams show a summary of the entities created in Atlas for Hive operations and assets. The supertypes that contribute attributes to the entity types are shaded.

Figure 1: Atlas Entity Types for HiveServer Data Sets

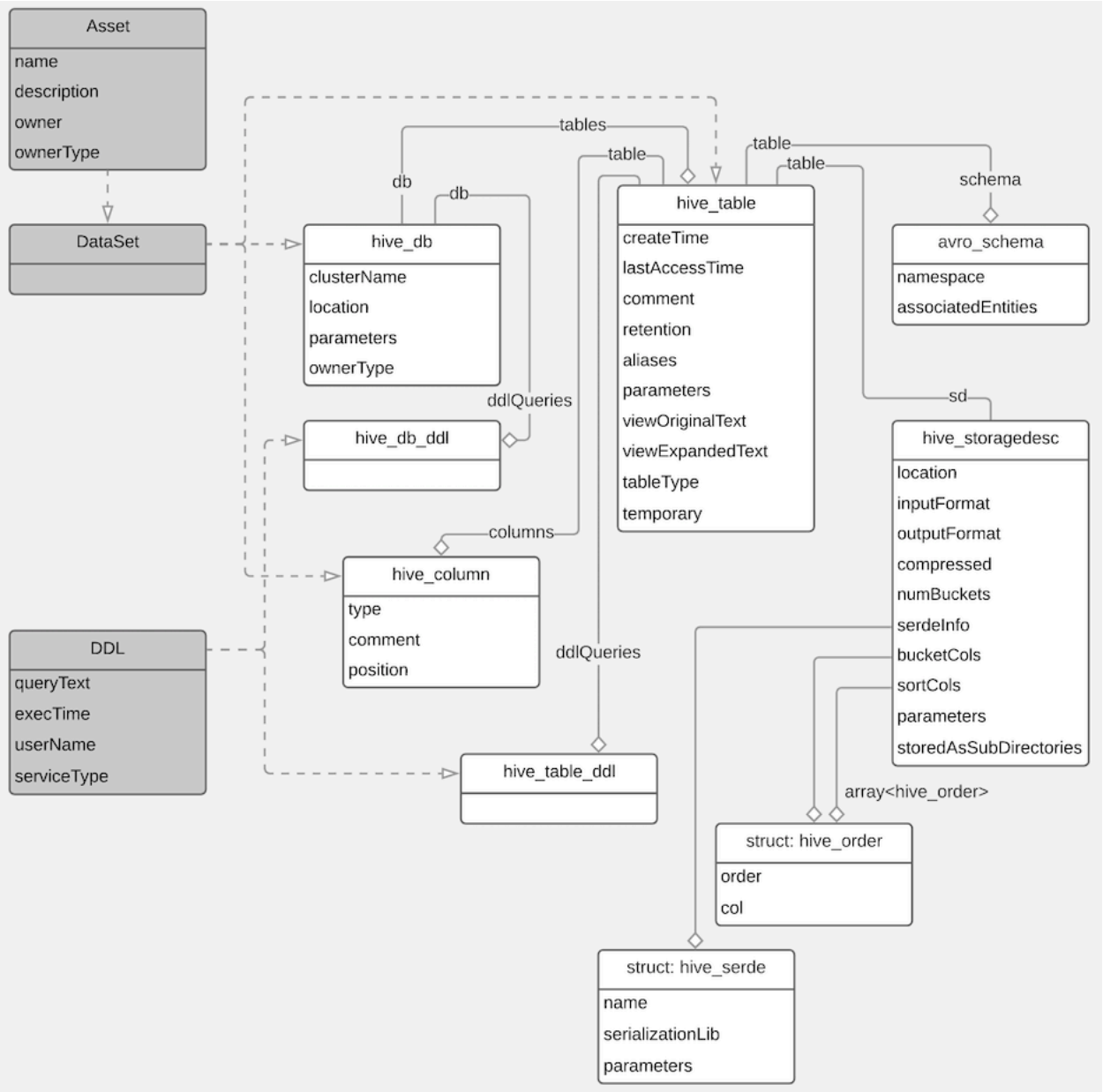

Figure 2: Atlas Entity Types for HiveServer Processes

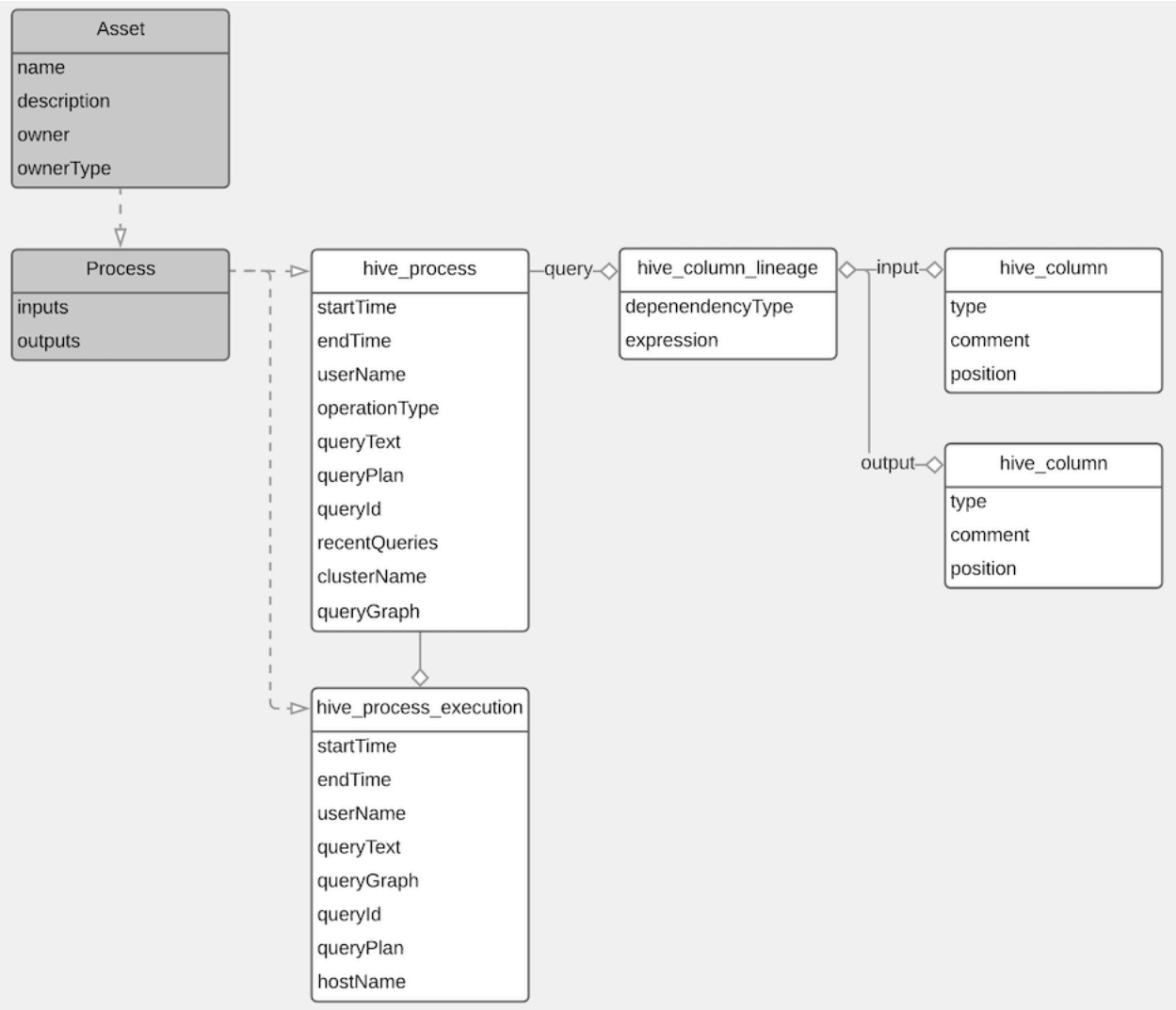

The metadata collected for each entity type is as follows:

Hive Process

Identifier	Example content
typeName	hive_process
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	<database>.<target table>@<clustername>:<generated ID> The generated ID is distinct from the GUID.
name	Text of the query.
inputs	List of the input tables or views, including each entity's type name and the qualified name.
outputs	List of the output objects, including each entity's type name and the qualified name.
recentQueries	Last query executed (duplicated in process_execution).
operationType	One of the operations that triggers metadata collection.
queryPlan	Reserved for future use.

Hive Process Execution

Identifier	Example Content
typeName	hive_process_execution
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	<database>.<target table>@<clustername>:<ID from process qualified name>:<ID from the process execution name>:<generated ID for this process execution>
name	Text of the query with a system-generated ID added to the end.
queryText	Text of the query.
queryPlan	Reserved for future use.
queryId	impala_<date as yyyymmddhhmmss>_<generated id>
startTime	Query start time.
endTime	Query end time.
userName	The user who ran the query.
Relationship: Process	One process to one or more process executions. hive_process_process_execution

Hive Database

Identifier	Example Content
typeName	hive_db
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	<database>@<clustername>
name	Database name as reported from Hive.
clusterName	Cluster name.
location	The file system path where the backing files for the database are stored. This could be an HDFS path, an AWS S3 object, or an Azure data storage location.
owner	The user who initially created the database.
ownerType	The principal type of the database owner. Could be USER, ROLE, or GROUP.
parameters	Additional key-value pair metadata that comes from Hive such as table size, number of rows, and number of storage files.
Relationship: Table	One database to many tables. hive_table_db
Relationship: Database DDL	One database to many database DDL entities. hive_db_ddl_queries

Hive Table

Identifier	Example Content
typeName	hive_table
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	<database>.<tablename>@<clustername>
name	Table name.
columns	List of the columns defined in the table. The Atlas Dashboard shows these as links to the column entity details.
owner	The user who created the table.

Identifier	Example Content
parameters	Table details from HiveServer such as: • totalSize • External • numFiles • transient_lastDdlTime • bucketing_version
retention	Provided by HS2. Integer value
sd	The location of the table data, the storage description. <database>.<table>@<clustername>_storage
tableType	How the table was created: one of EXTERNAL_TABLE, VIRTUAL_VIEW, or MANAGED_TABLE.
Relationship: Database	One database to many tables. hive_table_db
Relationship: Columns	One table to one or more columns. hive_table_columns
Relationship: Partition Key Column	One table to one or more columns that are partition keys. hive_table_partitionkeys
Relationship: Storage Description	One table to one storage description. hive_table_storagedesc
Relationship: DDL	One table to many DDL entities. hive_table_ddl_queries

Hive Column

Identifier	Example Content
typeName	hive_column
comment	Metadata from Hive from the column description.
name	Column name as reported by HMS.
owner	Table owner name as reported by HMS.
position	This column's position in the list of columns in a zero-based index.
qualifiedName	<database>.<table>.<column>@<clustername>
table	Table name. Also modeled as relationship.
type	Column data type as reported by HMS.
Relationship: table	One table to one or more columns. hive_table_columns
Relationship: inputToProcesses	The hive_column_lineage entities that include this column in the input to a transformation. The relationship type is dataset_process_inputs.
Relationship: outputFromProcesses	The hive_column_lineage entities that include this column in the output to a transformation. The relationship type is process_dataset_outputs.
Relationship: Table	One table to one or more columns. hive_table_columns
Relationship: Partition Key Column	One table to one or more columns that are partition keys. hive_table_partitionkeys

Hive Column Lineage

Identifier	Example Content
typeName	hive_column_lineage
dependencyType	The type of relationship between the input and output columns; one of SIMPLE, EXPRESSION, or SCRIPT.
name	<database>.<table>@<clustername>:<generated ID>:<output_column>

Identifier	Example Content
inputs	List of 0 or more hive_column entities that contributed to the output columns. This is a legacy model component: the more current model uses a relationship attribute.
outputs	This is a legacy model component: the more current model uses a relationship attribute.
qualifiedName	Same as name.
query	Name of the hive_process entity that produced this lineage. This is a legacy model component: the more current model uses a relationship attribute.
Relationship: Process	Name of the hive_process entity that produced this lineage. hive_process_column_lineage
Relationship: inputToProcesses	List of 0 or more hive_column entities that contributed to the output columns.
Relationship: outputFromProcesses	List of 0 or more hive_column entities that were produced in the process.

Hive Storage Description

Identifier	Example Content
typeName	hive_storagedesc
compressed	Metadata from Hive indicating whether the table is stored compressed.
inputFormat	Metadata from Hive indicating the storage input format.
outputFormat	Metadata from Hive indicating the storage output format.
parameters	Additional metadata from Hive in the form of key-value pairs.
qualifiedName	<database>.<table>@<clustername>_storage
serdeInfo	Metadata from Hive indicating the serialization/deserialization implementation used to write/read table data.
sortCols	Metadata from Hive listing the column or columns used to sort the table data.
storedAsSubDirectories	Metadata from Hive indicating whether a skewed table uses the list bucketing feature, which creates subdirectories for skewed values.
numBuckets	Metadata from Hive indicating the number of buckets for bucketed tables. Non-bucketed tables are indicated by -1.
table	The table that this storage description holds data for. Also represented as a relationship.
Relationship: table	The table that this storage description holds data for.

HiveServer relationships

Atlas shows the entities related to this entity in the Relationships tab in the Dashboard.

The Relationship tab shows the relationships that exist for an entity. Use this view to navigate among related entities.

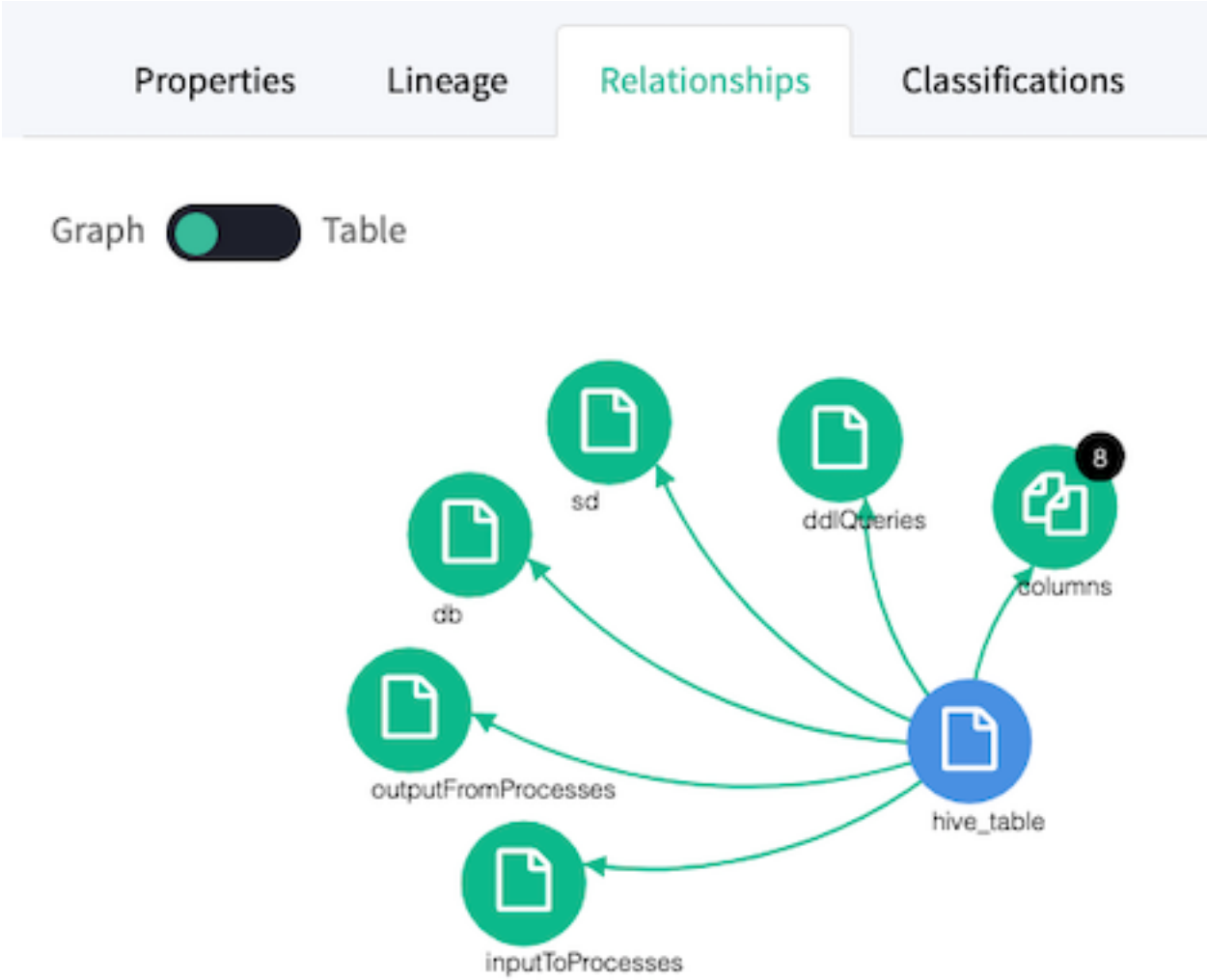

HiveServer lineage

Atlas collects metadata from HiveServer to represent the lineage among data assets.

The Atlas lineage graph shows the input and output processes that the current entity participated in, specifically those relationships modeled as “inputToProcesses” and “outputFromProcesses.” Entities are included if they were inputs to processes that lead to the current entity or they are output from processes for which the current entity was an input. HiveServer processes follow this pattern.

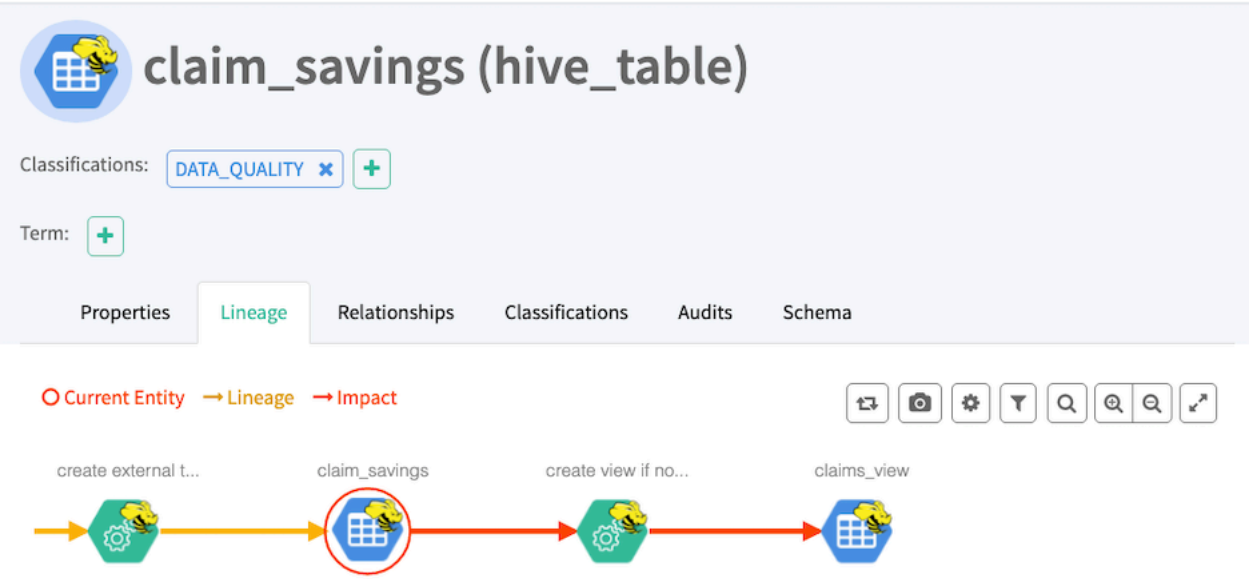

Related Information

Viewing lineage

HiveServer audit entries

Atlas lists changes to metadata entities in the Audit tab in the Dashboard.

Atlas tracks the lifecycle of each Hive entity, including its creation, update, and deletion. User access and actions that affect the data content of the source asset are not included in the audit.

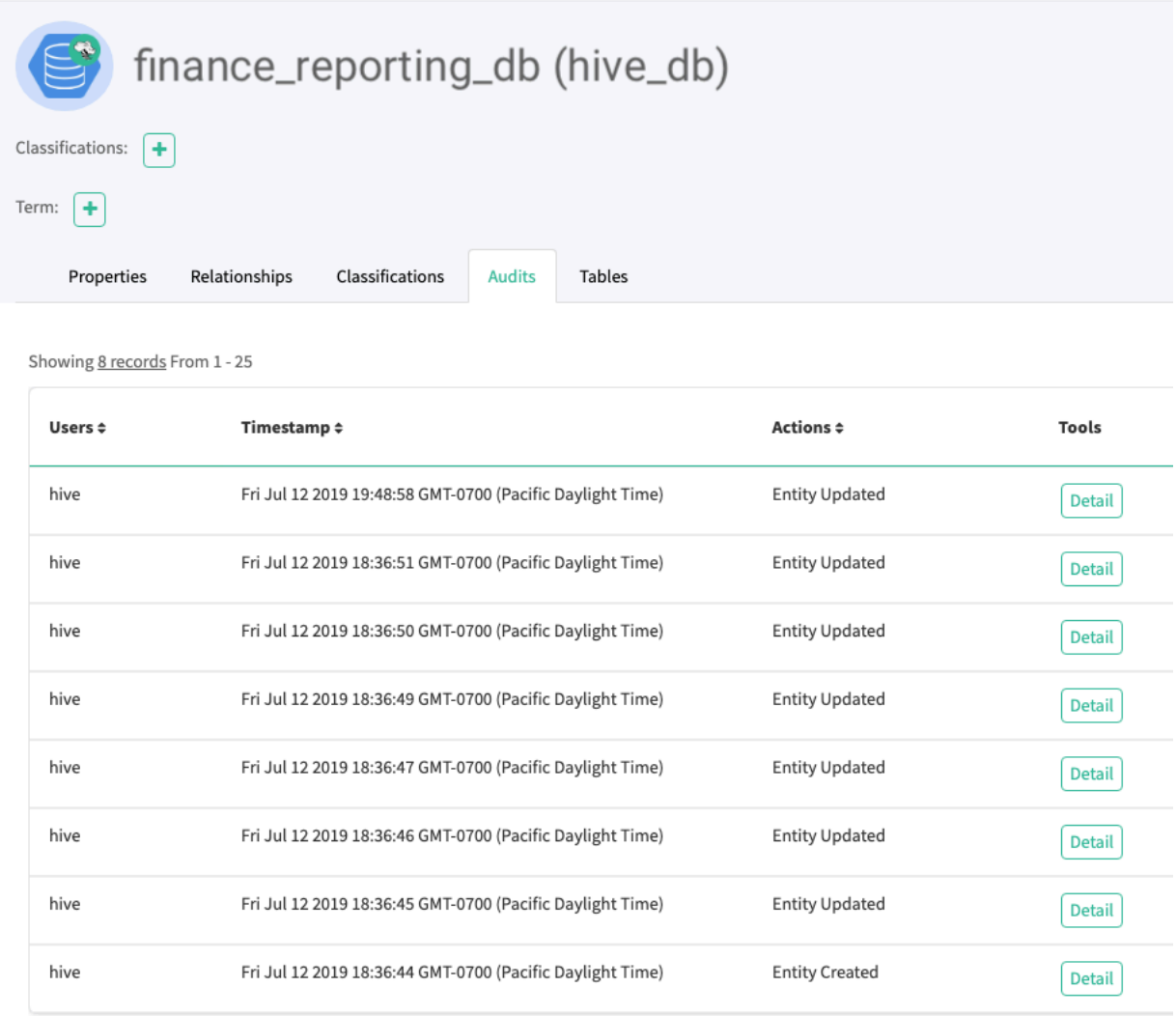

HBase metadata collection

Atlas can collect metadata from HBase that describes the data assets HBase manages.

An Atlas hook runs in each HBase instance. This hook sends metadata to Atlas for HBase data assets. HBase namespaces, tables, columns, and column families are represented by entities in Atlas.

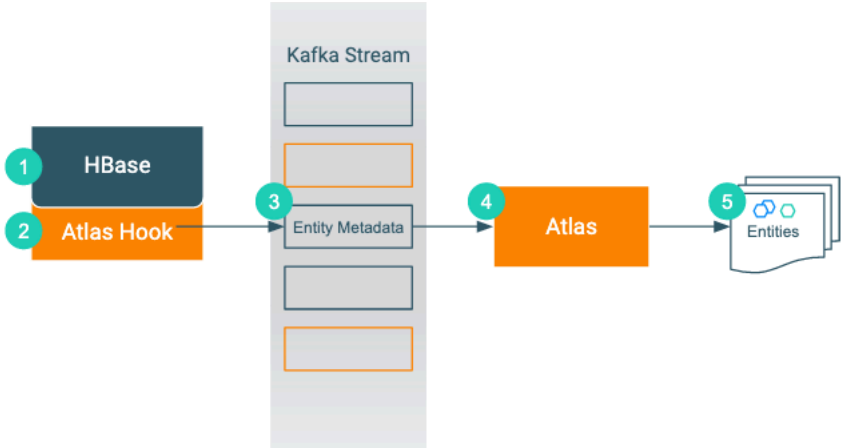

1. When an action occurs in the HBase instance...
2. The corresponding Atlas hook collects information for the action into metadata entities.
3. The hook publishes the metadata on a Kafka topic.
4. Atlas reads the message from the topic and determines what information will create new entities and what information updates existing entities.
5. Atlas creates and updates the appropriate entities.

The Atlas bridge for HBase pulls the same metadata as the hook, but instead of sending the metadata through Kafka, it passes message in bulk in an API call. The bridge creates entities in Atlas for all of the existing HBase namespaces, tables, columns, and column families.

HBase actions that produce Atlas entities

As data assets are created in HBase, Atlas generates entities to represent those assets. Atlas does not create processes to represent HBase operations.

The following table lists the HBase actions that produce or update metadata in Atlas.

This Action in HBase...	...Produces metadata for these Atlas entities
alter_async	hbase_namespace, hbase_table, hbase_column_family
create_namespace, alter_namespace, drop_namespace	hbase_namespace
create table, alter table, drop table, drop_all tables	alter table (create column family), alter table (alter column family), alter table (delete column family)
alter table (create column family), alter table (alter column family), alter table (delete column family)	hive_process, hive_process_execution

Notable actions in HBase that do NOT produce metadata entities include any actions that affect only data and not metadata. In addition, Atlas does not collect metadata for HBase columns. Actions that do not create Atlas entities include:

- Truncate table
- Put (cell value)]
- Disable/enable table

HBase entities created in Atlas

Each HBase data set entity in Atlas includes detailed metadata collected from HBase.

The following diagrams show a summary of the entities created in Atlas for Hive operations and assets. The supertypes that contribute attributes to the entity types are shaded.

Figure 3: Atlas Entity Types for HBase Data Sets

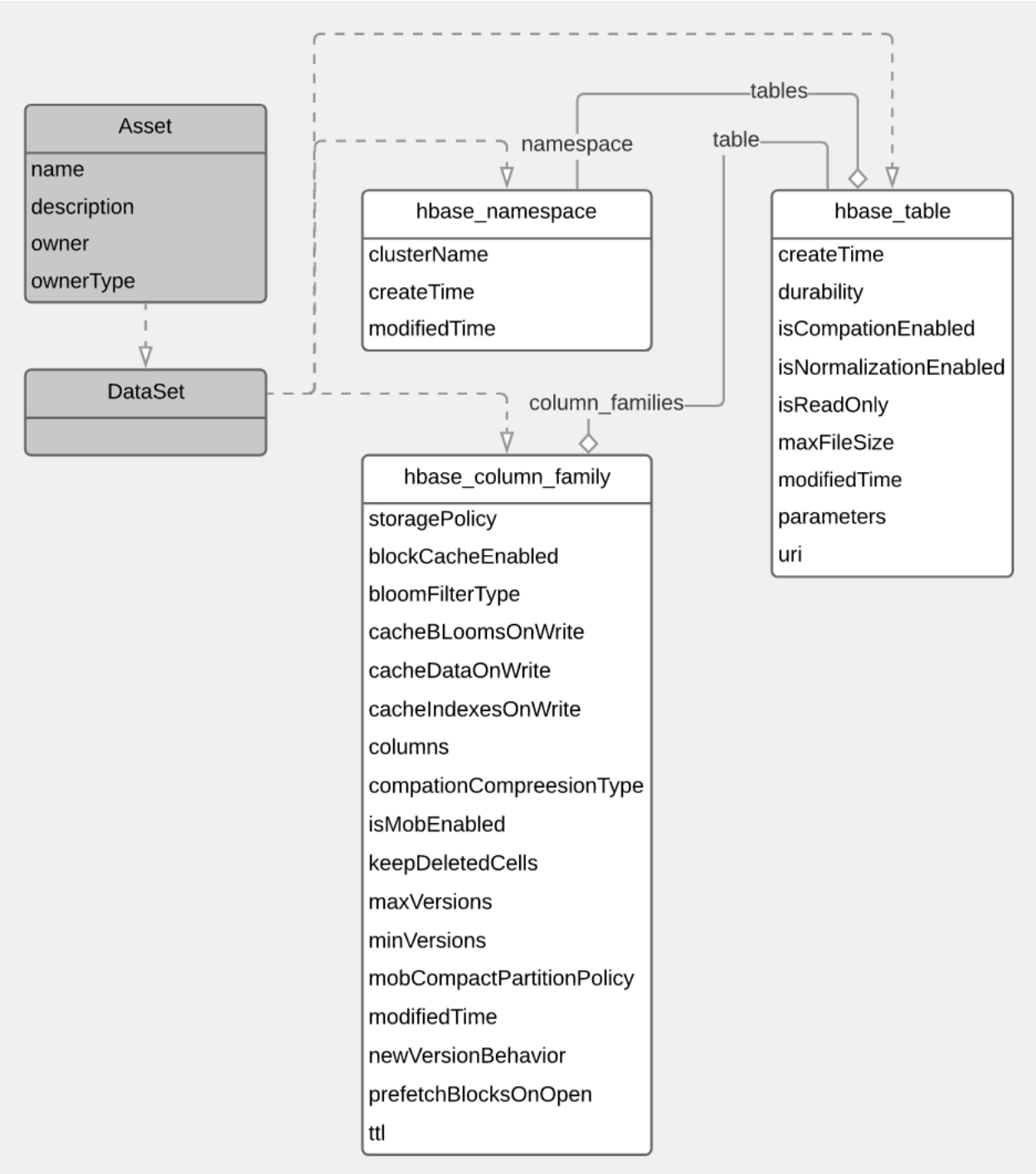

The metadata collected for each entity type is as follows:

HBase Namespace

Identifier	Example content
typeName	hbase_namespace
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	<name>@<clustername>
name	Namespace name as reported from HBase.

Identifier	Example content
description	String description metadata from HBase.
modifiedTime	Time from HBase indicating a change to the namespace. Formatted as in this example: “Wed Apr 17 2019 18:32:14 GMT-0700 (Pacific Daylight Time)”
owner	Owner as reported from HBase.
parameters	Reserved for future use.
replicatedFrom	Reserved for future use.
replicatedTo	Reserved for future use.
Relationship: tables	One namespace to many tables. hbase_table_namespace

HBase Table

Identifier	Example content
typeName	hbase_table
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	<namespace>:<tablename>@<clustername>
name	Table name as reported from HBase.
description	String description metadata from HBase.
modifiedTime	Time from HBase indicating a change to the table. Formatted as in this example: “Wed Apr 17 2019 18:32:14 GMT-0700 (Pacific Daylight Time)”
owner	Owner as reported from HBase.
parameters	Reserved for future use.
replicatedFrom	Reserved for future use.
replicatedTo	Reserved for future use.
durability	Storage property as reported from HBase. Values include true or false.
isCompactionEnabled	Storage property as reported from HBase. Values include true or false.
isNormalizationEnabled	Storage property as reported from HBase. Values include true or false.
isReadOnly	
maxFileSize	Storage property as reported from HBase. -1 indicates that no maximum was set.
uri	Table name.
Relationship: namespace	One namespace to many tables. hbase_table_namespace
Relationship: column families	Column families associated with this table. hbase_table_column_families

HBase Column Family

Identifier	Example content
typeName	hbase_column_family
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	<namespace>:<tablename>.<columnfamily>@<clustername>
name	Column family name as reported from HBase.
description	String description metadata from HBase.
modifiedTime	Time from HBase indicating a change to the column family. Formatted as in this example: “Wed Apr 17 2019 18:32:14 GMT-0700 (Pacific Daylight Time)”

Identifier	Example content
owner	Owner as reported from HBase.
StoragePolicy	Value for the storagePolicy property for the column family. Values include N/A, ALL_SSD, ONE_SSD, HOT, WARM, COLD.
blockCacheEnabled	Storage property as reported from HBase. Values include true or false.
bloomFilterType	Value for the BLOOM_FILTER_TYPE property for the column family. Values include NONE, ROW, or ROWCOL.
cacheBloomsOnWrite	Boolean value for the CACHE_BLOOMS_ON_WRITE property for the column family.
cacheDataOnWrite	Boolean value for the CACHE_DATA_ON_WRITE property for the column family.
cacheIndexesOnWrite	Boolean value for the CACHE_INDEX_ON_WRITE property for the column family.
columns	List of columns included in the column family.
compactionCompressionType	Storage property as reported from HBase.
compressionType	Value for the COMPRESSION property for the column family. Values include NONE, SNAPPY, LZO, LZ4, GZ. The default value is SNAPPY. **Note:** Atlas is only certified to use Snappy and Gzip for HBase table compaction.
createTime	Time from HBase indicating when the column family was created. Formatted as in this example: “Wed Apr 17 2019 18:32:14 GMT-0700 (Pacific Daylight Time)”
dataBlockEncoding	The DATA_BLOCK_ENCODING property for the column family. Values include NONE, PREFIX, DIFF, FAST_DIFF, ROW_INDEX_V1.
encryptionType	Column family encryption property. Values include “N/A”, and AES.
evictBlocksOnClose	Boolean value for the EVICT_BLOCKS_ON_CLOSE property for the column family.
inMemoryCompactionPolicy	In memory compaction behavior (IN_MEMORY_COMPACTION) set for the column family. Values include NONE, BASIC, EAGER, ADAPTIVE, or “N/A”.
isMobEnabled	Boolean value for Medium OBject (MOB) properties for the column family (IS_MOB).
keepDeletedCells	Boolean value for the KEEP_DELETED_CELLS property of the column family.
maxVersions	The maximum number of row versions this column family is configured to store.
minVersions	The minimum number of row versions this column family is configured to store.
mobCompactPartitionPolicy	The MOB_COMPACT_PARTITION_POLICY for this column family. Values include DAILY, WEEKLY, MONTHLY.
modifiedTime	Time from HBase indicating a change to the column family. Formatted as in this example: “Wed Apr 17 2019 18:32:14 GMT-0700 (Pacific Daylight Time)”
newVersionBehavior	Boolean value for the NEW_VERSION_BEHAVIOR property for this column family.
prefetchBlocksOnOpen	Boolean value for PREFETCH_BLOCKS_ON_OPEN property of the column family.
replicatedFrom	Not used.
replicatedTo	Not used.
table	The table that the column family corresponds to. Also modeled as a relationship.
ttl	Time to live (TTL) length in seconds. The TTL time encoded in the HBase for the row is specified in UTC.
Relationship: columns	Not used.
Relationship: table	One table to many column families. hbase_table_column_families

Changing the HBase column family compression type

Learn how to configure compression types. Changing the default Snappy algorithm to Gzip might be required depending on your use case and operational requirements.

About this task

Note: Atlas is only certified to use Snappy and Gzip for HBase table compaction.

Procedure

1. Add the atlas.graph.storage.hbase.compression-algorithm property in Atlas Server Advanced Configuration Snippet for conf/atlas-application.properties.
2. Enter the desired compression type in the property.
   atlas.graph.storage.hbase.compression-algorithm=*GZ*
3. Select the Master-(Active) Role Type for the HBase Instance from Cloudera Manager. Then, log in by using SSH in the terminal.
4. Run kinit -kt ****PATH TO HBASE.KEYTAB**** hbase to authenticate into HBase.
5. Enter the HBase shell.
   a) Run list to see the available HBase tables.
6. Run 'desc 'atlas_janus'' to see all column descriptions.
7. Change the compression type for each required table:

```
alter 'atlas_janus', {NAME => 'e' , COMPRESSION => ''GZ''}
alter 'atlas_janus', {NAME => 'f' , COMPRESSION => 'GZ'}
alter 'atlas_janus', {NAME => 'g' , COMPRESSION => 'GZ'}
alter 'atlas_janus', {NAME => 'h' , COMPRESSION => 'GZ'}
alter 'atlas_janus', {NAME => 'i' , COMPRESSION => 'GZ'}
alter 'atlas_janus', {NAME => 'l' , COMPRESSION => 'GZ'}
alter 'atlas_janus', {NAME => 'm' , COMPRESSION => 'GZ'}
alter 'atlas_janus', {NAME => 's' , COMPRESSION => 'GZ'}
alter 'atlas_janus', {NAME => 't' , COMPRESSION => 'GZ'}
```

8. Run major_compact 'atlas_janus' only in HBase Shell to start compacting your tables.

Results

Your HBase column family uses the newly set compression type.

Hbase lineage

Atlas collects lineage information for HBase data assets when HBase tables are referenced in HiveServer or Impala operations.

The Atlas lineage graph shows the input and output processes that the current entity participated in, specifically those relationships modeled as “inputToProcesses” and “outputFromProcesses.” Entities are included if they were inputs to processes that lead to the current entity or they are output from processes for which the current entity was an input.

No lineage metadata is collected directly from HBase.

Related Information

Viewing lineage

HBase audit entries

Atlas lists changes to metadata entities in the Audit tab in the Dashboard.

Atlas tracks the lifecycle of each HBase entity, including its creation, update, and deletion. User access and actions that affect the data content of the source asset are not included in the audit.

Schema Registry metadata collection

Integrating Schema Registry with Atlas will enable you to persist and view the schemas in Atlas.

Atlas serves as a metadata catalogue that shows relationships between entities. Additionally, when Kafka is also integrated with Atlas you can connect Kafka topics and schemas.

As an example, an ETL job which reads encoded data from Hadoop, performs some operations on it and writes it into Kafka. In order to read the encoded data, the ETL job needs access to Schema Registry to obtain the schema. To write the data into Kafka, it needs to be encoded again, perhaps with a different schema. A relationship graph for this example is detailed in the following diagram.

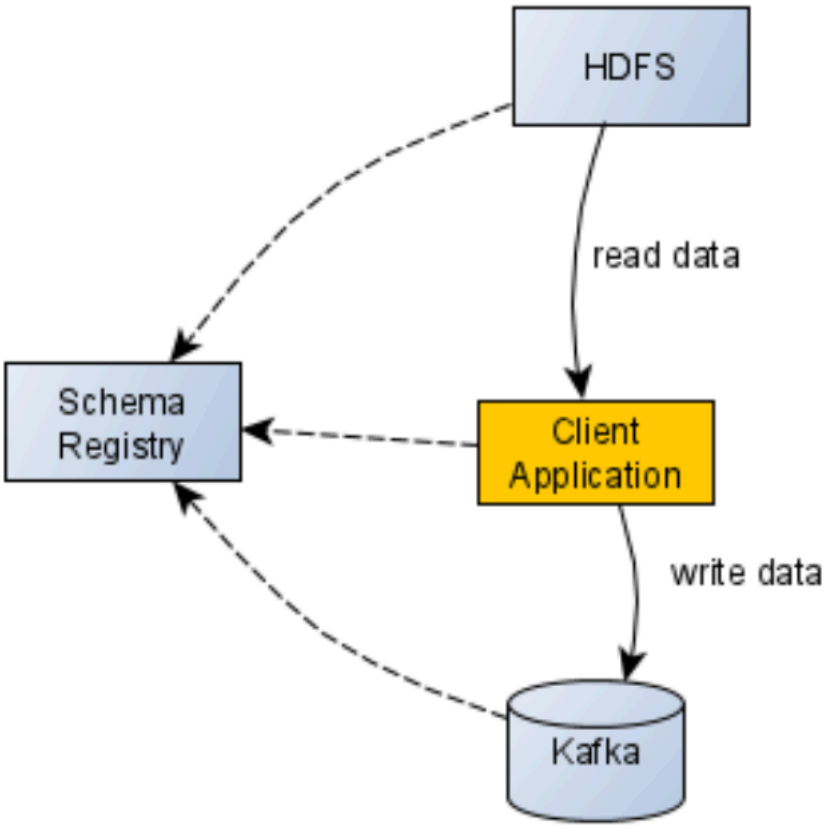

In a real-world environment, however, we can have many components communicating with each other and it might become hard to track the relations and data flows.

Atlas in its core is a metadata catalog which uses a graph database to store entities and its relationships, which lets users track the data flows as discussed earlier in this section. The graph would be built automatically for the end-user. The corresponding services (Hadoop, Schema Registry, Kafka, Hive (HiveServer2 and HMS), Impala, Spark, HBase, Sqoop, Storm, NiFi) send updates to Atlas and build the graph on the go.

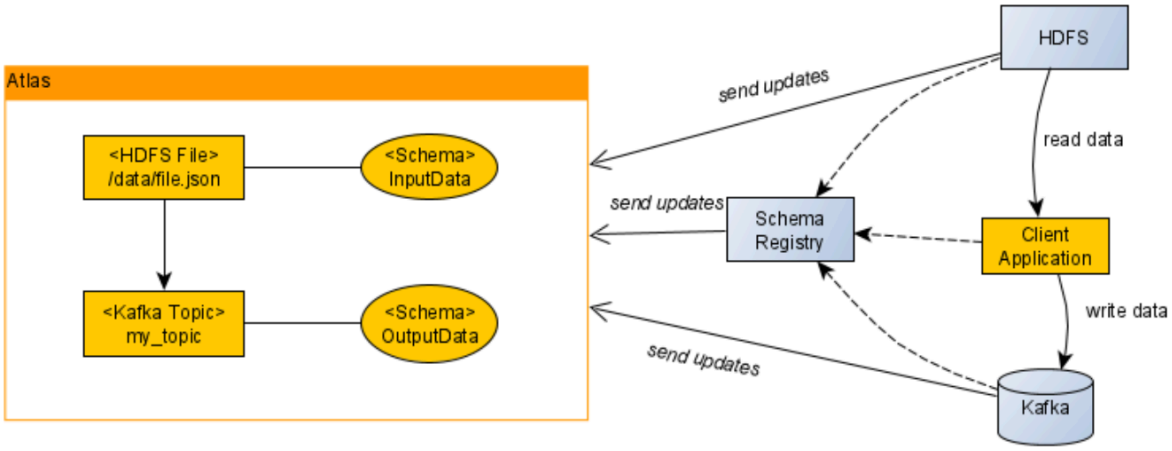

Note: The implementation of how those updates are sent varies between services.

There are two scenarios with which this integration works:

- When Kafka topic is created:

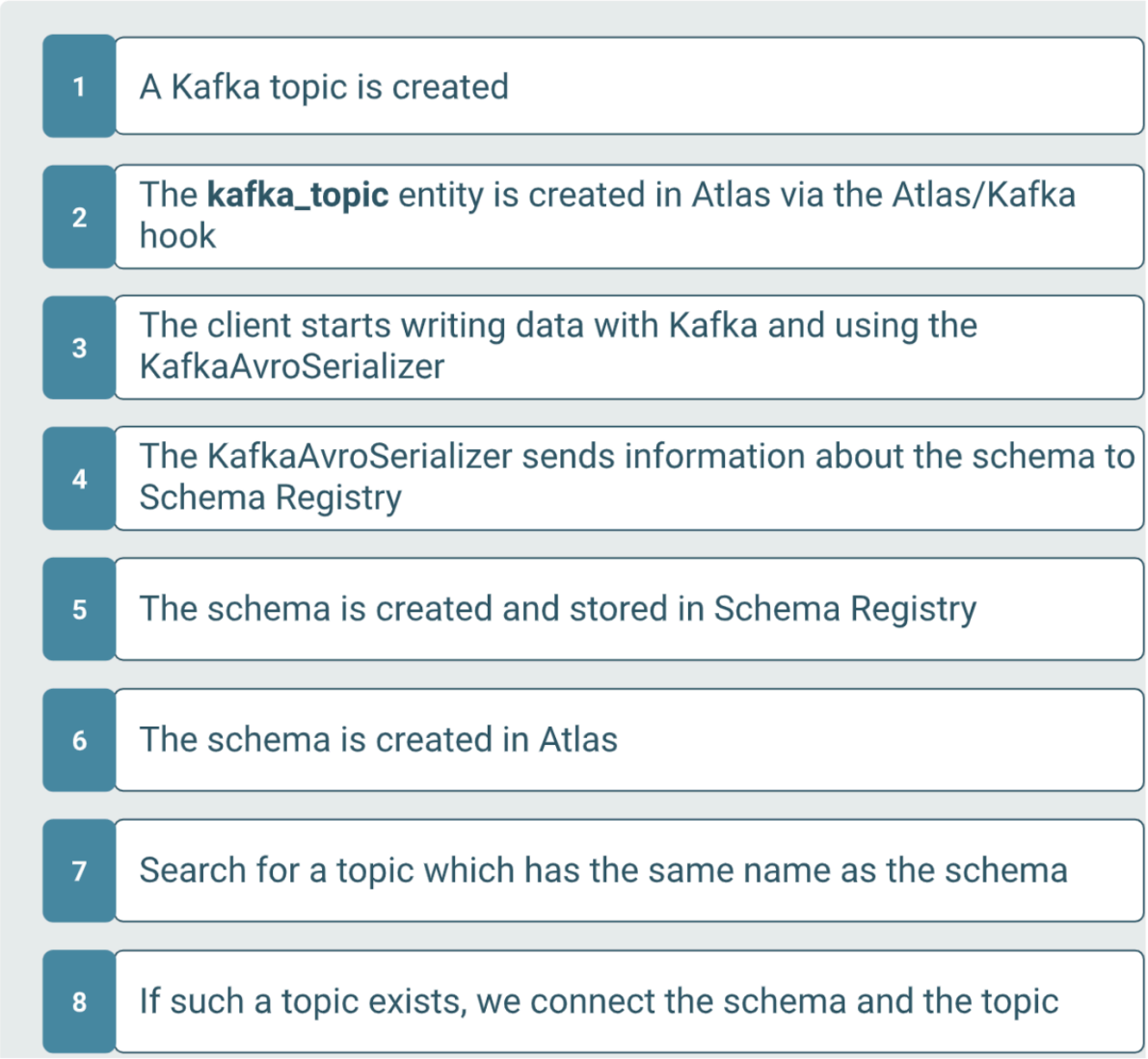

- When Schema data is created in Atlas:

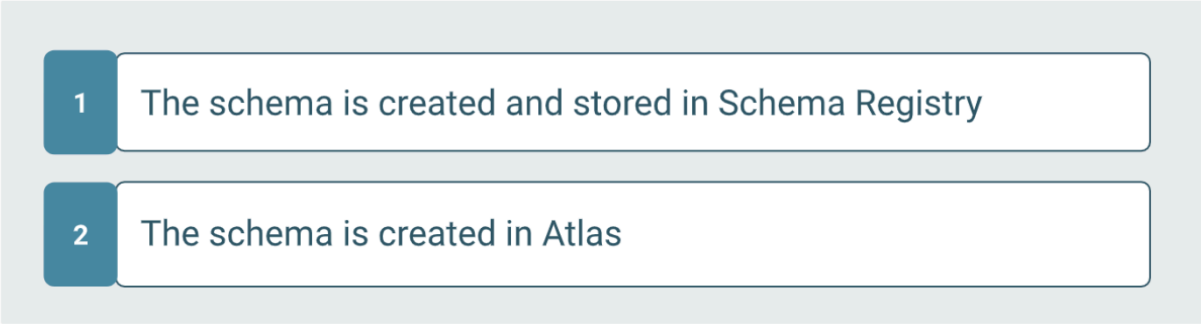

Configuring Atlas and Schema Registry

You must configure the settings in your Cloudera Manager instance

Procedure

1. Navigate to the Configuration tab of the Schema Registry in your Cloudera Manager instance and enter “Atlas”.
2. Check the Atlas Service checkbox.
3. Click Save Changes and restart Schema Registry.

What to do next

Once Apache Atlas integration is enabled, you can create new schemas which show up in Atlas. Note that the process which mirrors schemas from Schema Registry to Atlas is asynchronous and runs once every minute. A schema which is created does not show up immediately, but only after one minute.

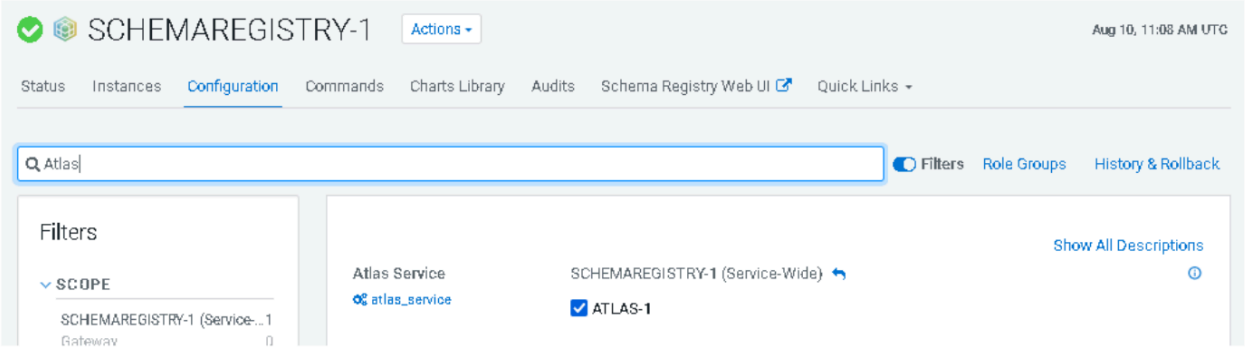

Note: If a Kafka topic exists with the same name as the schema, then a relationship is created between them in Atlas. If no such topic exists, only the schema is created.

Schema Registry actions that produce Atlas entities

Once the Schema Registry application is integrated with Atlas, entities are created in Atlas to represent the metadata.

The following table shows a summary of the entities created in Atlas to represent the metadata.

Atlas Entity	Description
Schema metadata	Contains the properties of a schema.
Schema version	A schema can have one or more versions. They are all connected to one metadata.
Kafka topic	If a Kafka topic exists and the Atlas-Kafka integration is enabled, then a topic entity is created. When a schema is created with the same name as the topic then a relationship is created between them.

Schema replationships

Atlas shows the entities related to this entity in the Relationships tab in the Dashboard.

The Relationship tab shows the relationships that exist for an entity. Use this view to navigate among related entities.

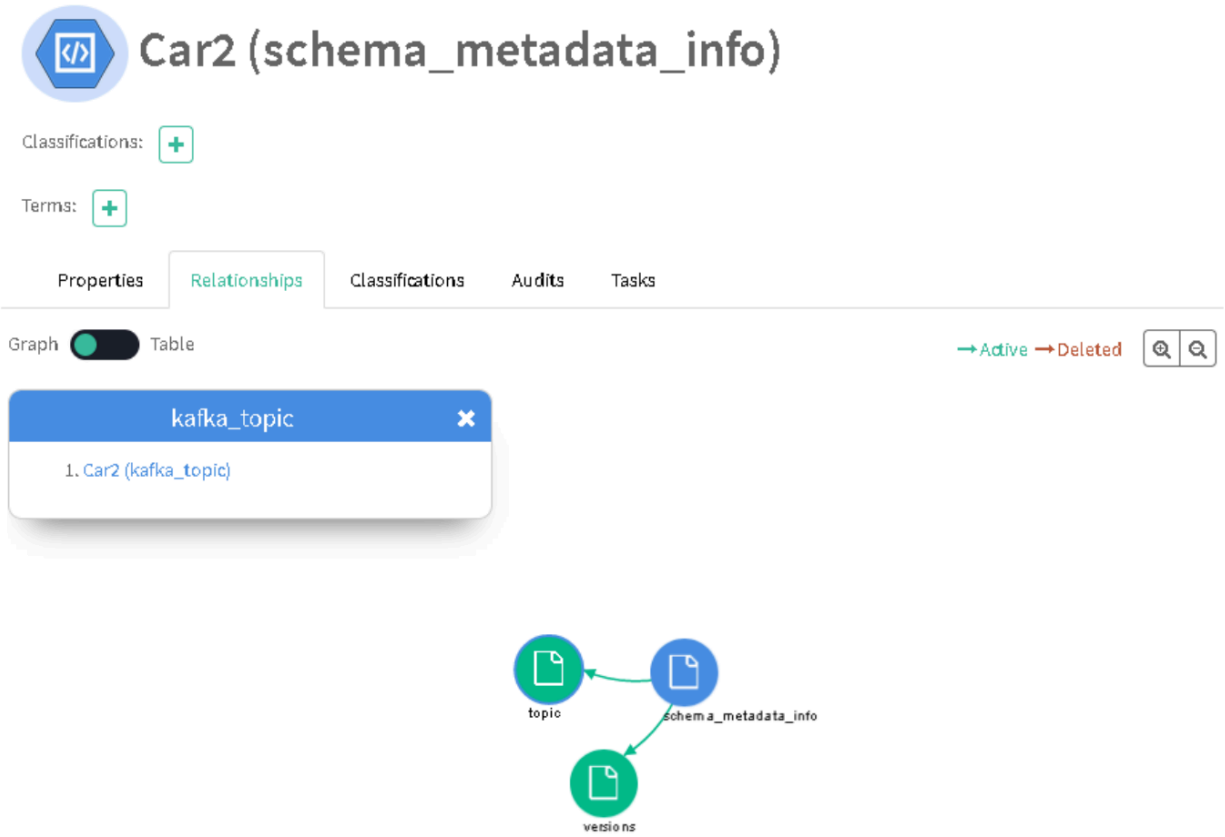
Car2 (schema_metadata_info)
Classifications:
Terms:
Properties
Relationships
Classifications
Audits
Tasks
Graph
Table
Active
Deleted
kafka_topic
1. Car2 (kafka_topic)
topic
schema_metadata_info
versions

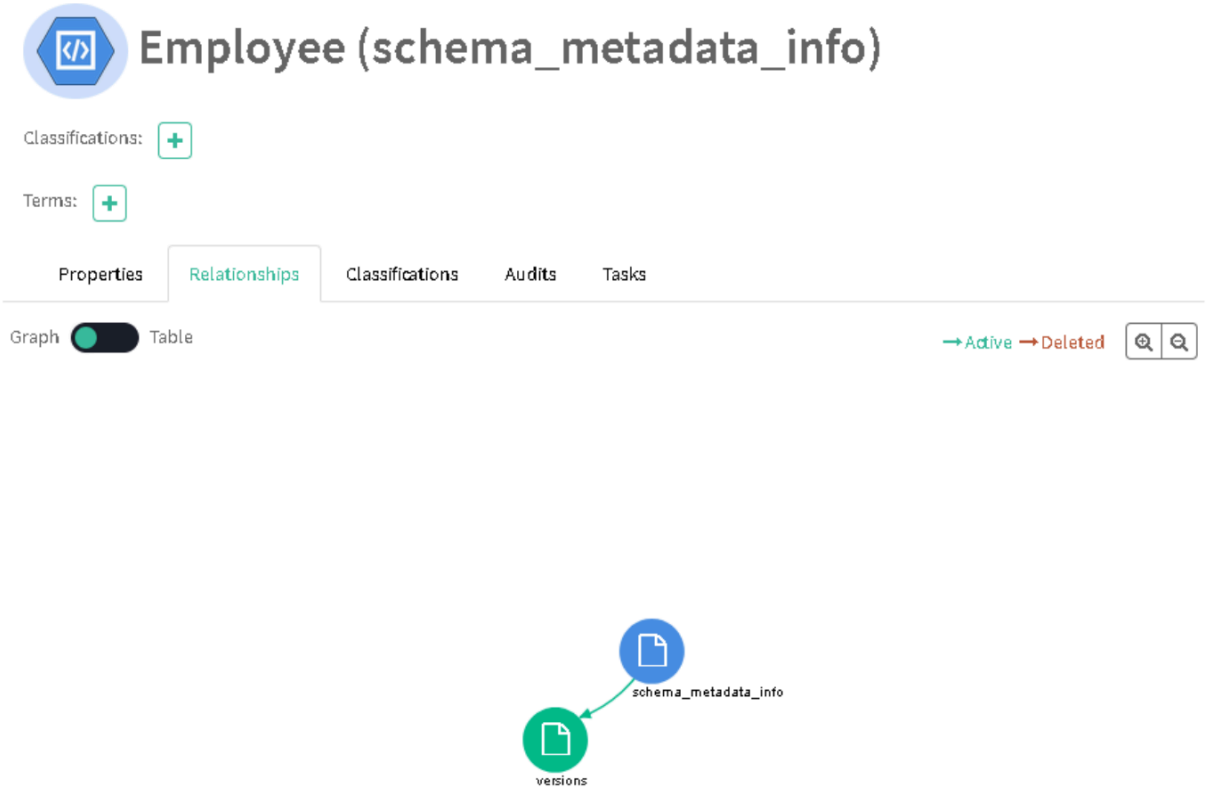
Employee (schema_metadata_info)
Classifications:
Terms:
Properties
Relationships
Classifications
Audits
Tasks
Graph
Table
Active
Deleted
schema_metadata_info
versions

Schema Registry audit entries

Atlas lists changes to metadata entities in the Audits tab in the Dashboard.

Atlas tracks the lifecycle of each Schema Registry entity, including its creation and update. User access and actions that affect the data content of the source asset are not included in the audit.

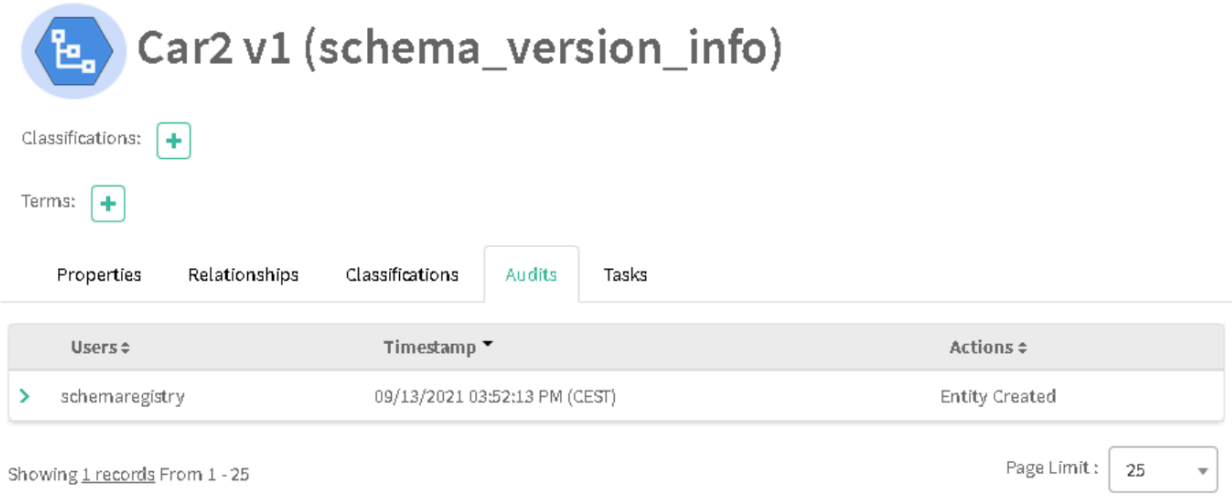

Troubleshooting Schema Registry

What do you do if you are not able to view Atlas metadata from Schema Registry?

Mirroring schemas from Schema Registry to Atlas is done by an asynchronous process. When a schema or schema version is created, the information about this event is generally stored in the database table atlas_events. When an event has been processed, its status will be updated to either success or failure.

Mirroring a schema to Atlas can fail for multiple reasons but not limited to the following:

- Atlas may be down
- Network connection failure

If you have analyzed the problem and can fix it by “replaying” the event, you must update the table and set the failure flag to false. In this case, the event will be picked up again and processed once again.

The SQL format to perform the above:

update atlas_events set processed=0, failed=0 where id = <event id>

Impala metadata collection

Atlas can collect metadata for queries from Impala. It collects metadata for affected data assets from Hive Metastore (HMS).

An Atlas hook runs in each Impalad instance. This hook sends metadata to Atlas for Impala operations, which are represented by process and process execution entities in Atlas.

In addition, an Atlas hook runs in Hive Metastore (HMS). Before sending metadata to Atlas, Impala synchronizes its metadata with HMS. This synchronization makes sure that Impala uses the same names and IDs as HMS.

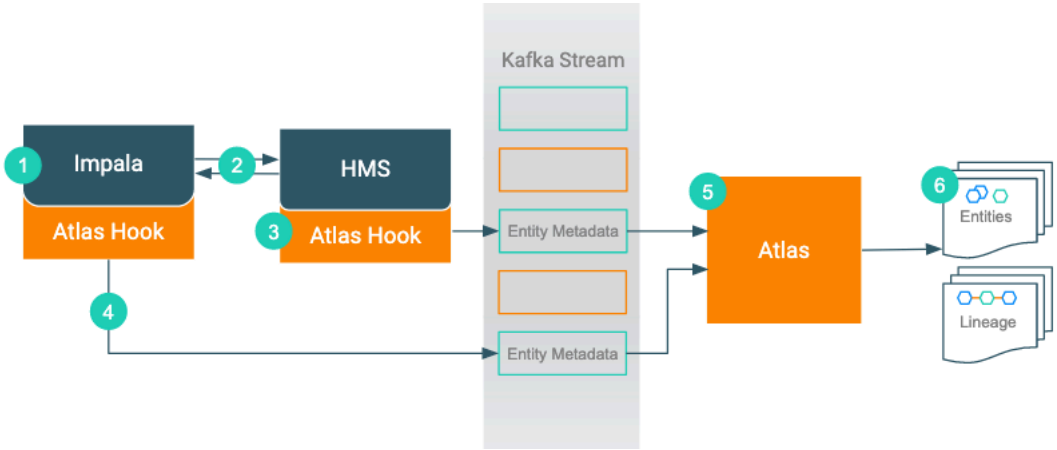

1. When an action occurs in the Impala instance...
2. Impala updates HMS with information about the assets affected by the action.
3. The Atlas hook for HMS collects information for the changed and new assets and forms it into metadata entities. It publishes the metadata to a Kafka topic.
4. The Atlas hook for the Impala instance collects information for the action and forms it into metadata entities. It publishes the metadata to a Kafka topic.
5. Atlas reads the messages from the topic and determines what information will create new entities and what information updates existing entities. Atlas is able to determine the correct entities regardless of the order in which Atlas receives messages from the Kafka topic.
6. Atlas creates the appropriate entities and determines lineage from existing entities to the new entities.

Impala actions that produce Atlas entities

Impala operations that create, update, or delete Hive metadata will affect Atlas entities; operations that only affect data do not show up in Atlas.

The following table lists the Impala actions that produce or update metadata in Atlas.

This Action in Impala...	...Produces metadata for these Atlas entities	...Triggers HMS to produce metadata for these Atlas entities	...Produces metadata for these Atlas relationships
CREATETABLE_AS_SELECT	impala_process, impala_process_execution, impala_column_lineage, hive_db hive_table_ddl	hive_table, hive_column(s), hive_storagedesc, hive_db hive_table_ddl	hive_table_db, hive_table_columns, hive_table_partitionkeys, hive_table_storagedesc, hive_process_process_execution, hive_process_columnlineage, hive_table_ddl_queries, hive_db_ddl_queries
CREATEVIEW	impala_process, impala_process_execution, impala_column_lineage, hive_table_ddl	hive_table, hive_column(s), hive_db	hive_table_db, hive_table_columns, hive_table_partitionkeys, hive_process_process_execution, hive_process_columnlineage, hive_table_ddl_queries

This Action in Impala...	...Produces metadata for these Atlas entities	...Triggers HMS to produce metadata for these Atlas entities	...Produces metadata for these Atlas relationships
ALTERVIEW_AS_SELECT	impala_process, impala_process_execution, impala_column_lineage, hive_table_ddl	Updates to: hive_table, hive_column(s)	hive_process_process_execution, hive_process_columnlineage, hive_table_ddl_queries
INSERT INTO, INSERT, OVERWRITE	impala_process, impala_process_execution	If not already in Atlas, HMS sends metadata for data assets indicated in the query: hive_table, hive_column(s), hive_storagedesc, hive_db	hive_process_process_execution

Notable actions in Impala that do NOT produce process or process execution entities in Atlas, meaning that no lineage is produced for these operations:

- LOAD DATA INPATH
- CREATE TABLE (table metadata produced by HMS)
- ALTER VIEW (table metadata produced by HMS)
- SELECT or other queries that don't produce output

Impala entities created in Atlas

Each Impala entity in Atlas includes detailed metadata for Impala queries.

The following diagrams show a summary of the entities created in Atlas for Impala operations. The supertypes that contribute attributes to the entity types are shaded.

Figure 4: Atlas Entity Types for Impala Operations

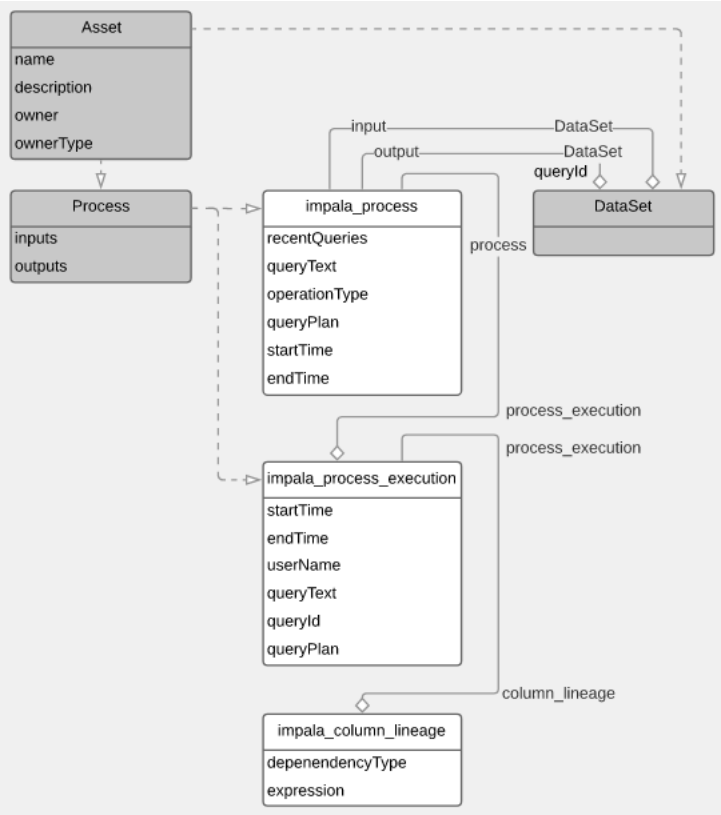

The metadata collected for each entity type is as follows:

Impala Process

Identifier	Example content
typeName	impala_process
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	<database>.<target table>@<clustername>:<generated ID> The generated ID is distinct from the GUID.
name	Text of the query.
inputs	List of the input tables or views, including each entity's type name and the qualified name.
outputs	List of the output objects, including each entity's type name and the qualified name.
recentQueries	Last query executed (duplicated in process_execution).
operationType	One of the operations that triggers metadata collection.
queryPlan	Reserved for future use.
startTime	Most recent query start time.
endTime	Most recent query end time.
Relationship: Process Execution	One process to one or more process executions. impala_process_process_execution
Relationship: Column Lineage	One process to one or more column lineages. impala_process_column_lineage

Impala Process Execution

Identifier	Example Content
typeName	impala_process_execution
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	<database>.<target table>@<clustername>:<ID from process qualified name>:<ID from the process execution name>:<generated ID for this process execution>
name	Text of the query with a system-generated ID added to the end.
queryText	Text of the query.
queryPlan	Reserved for future use.
queryId	impala_<date as yyyymmddhhmmss>_<generated id>
startTime	Query start time.
endTime	Query end time.
userName	The user who ran the query.
Relationship: Process	One process to one or more process executions. impala_process_process_execution

Impala Column Lineage

Identifier	Example Content
typeName	impala_column_lineage
dependencyType	The type of relationship between the input and output columns; one of SIMPLE, EXPRESSION, or SCRIPT.
name	<database>.<table>@<clustername>:<generated ID>:<output_column>
inputs	List of 0 or more hive_column entities that contributed to the output columns. This is a legacy model component: the more current model uses a relationship attribute.
outputs	This is a legacy model component: the more current model uses a relationship attribute.
qualifiedName	Same as name.
query	Name of the impala_process entity that produced this lineage. This is a legacy model component: the more current model uses a relationship attribute.
Relationship: Process	Name of the impala_process entity that produced this lineage. impala_process_column_lineage
Relationship: inputToProcesses	List of 0 or more hive_column entities that contributed to the output columns.
Relationship: outputFromProcesses	List of 0 or more hive_column entities that were produced in the process.

Impala lineage

You can use the Atlas lineage graph to understand the source and impact of data and changes to data over time and across all your data.

Atlas collects metadata from Impala to represent the lineage among data assets. The Atlas lineage graph shows the input and output processes that the current entity participated in. Entities are included if they were inputs to processes that lead to the current entity or they are output from processes for which the current entity was an input. Impala processes follow this pattern.

Note that lineage is not updated between a table and views that the table is a part of when an Impala ALTER TABLE operation runs on the table.

Related Information

Viewing lineage

Impala audit entries

Atlas lists changes to metadata entities in the Audit tab in the Dashboard.

Atlas tracks the lifecycle of each Impala entity, including its creation, update, and deletion.

Kafka metadata collection

Atlas can collect metadata from Kafka using the concept called metadata namespace.

When the Kafka cluster is configured to audit using Atlas, the Kafka brokers start notifying Atlas about metadata changes in the Kafka cluster. Clients do not have to be integrated with Atlas.

Note: Atlas auditing of Kafka topics is not supported for Kafka consumers using the version lower than 2.5.

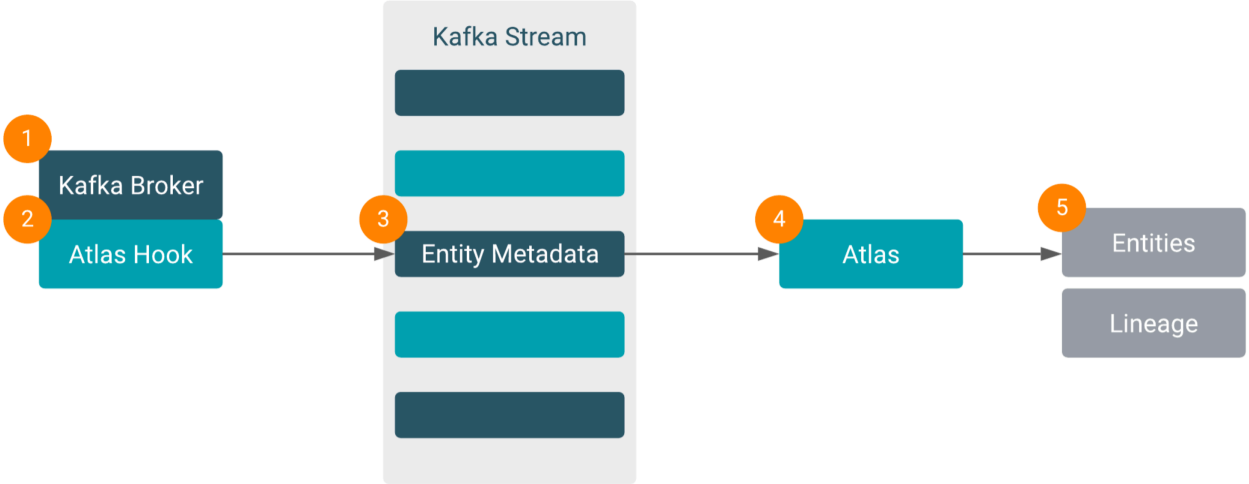

Atlas is used to track the metadata of multiple clusters. Metadata namespace in Atlas is used to categorise entities into groups. To avoid entity collision between clusters, entities belonging to different clusters can be grouped together, into separate metadata namespaces.

An Atlas hook runs in each Kafka instance. This hook transfers the metadata of Kafka assets to Atlas. The Kafka-Atlas auditing involves two metadata namespaces, one each for topics and clients.

The topic entities are created in the topic metadata namespace. Producer, Consumer and ConsumerGroup entities are created in the client metadata namespace.

For a simple use case, as a default configuration, the auditor can be configured to use the same namespace for both types.

Client metadata namespace can be used to insert entities representing an application spanning multiple Kafka clusters in the same namespace.

For example, if an application is connecting to a couple of Kafka clusters, and they have separate namespaces, the client entities are created twice, because each Kafka cluster will create them in its own client namespace. If Kafka clusters use the same client namespace, they will create and update the same client entity, so the application will be represented by a single producer and/or consumer entity.

Important: If Atlas auditing is enabled for an existing Kafka cluster with multiple topics that are not pushed into Atlas, the topics can be imported using the import-kafka tool manually or using the Import Kafka Topics Into Atlas action available in Cloudera Manager. Next, the auditor within the Kafka brokers retains the metadata of the Kafka cluster in sync with Atlas.

Related Information

Configuring the Atlas hook in Kafka

Importing Kafka entities into Atlas

Kafka actions that produce Atlas entities

After the Kafka application is integrated with Atlas, entities are created in Atlas to represent the metadata.

The following table show a summary of the entities created in Atlas to represent the metadata.

Atlas Entity	Description
Topic entities	Represents a Kafka topic
Producer entity	Represents a Kafka producer client. Producers are identified by their client.id.
Consumer entity	Represents a Kafka consumer client. Consumers are identified by their client.id.
ConsumerGroup entity	Represents a consumer group. Consumer groups are identified by their consumer group ID.
Producer lineage entity	Artificial process type entity to support the lineage of producers.
Consumer lineage entity	Artificial process type entity to support the lineage of consumers

Kafka relationships

Atlas shows the entities related to this entity in the Relationships tab in the Dashboard.

The Relationship tab shows the relationships that exist for an entity. Use this view to navigate among related entities.

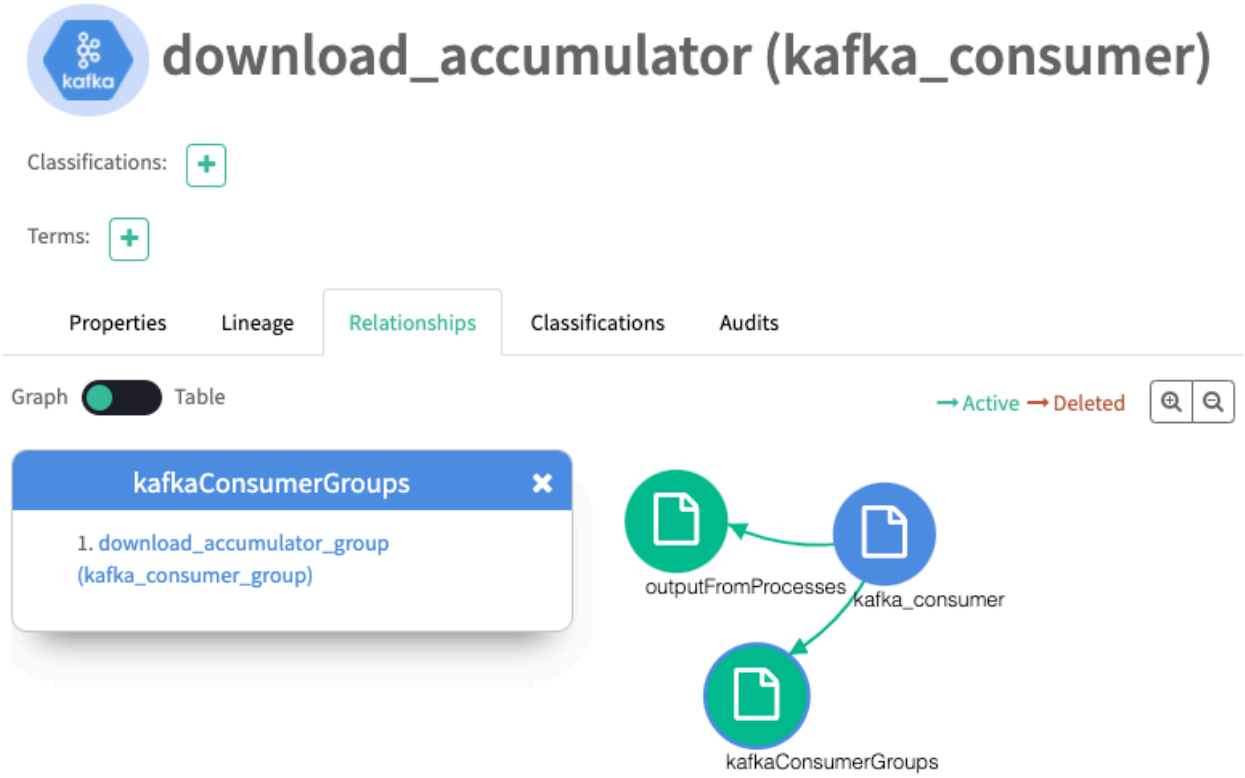

Kafka lineage

Using the Lineage tab, Atlas collects metadata from Kafka to represent the lineage among data assets.

The Atlas lineage graph shows the input and output processes that the current entity participated in, specifically those relationships modeled as “inputToProcesses” and “outputFromProcesses.” Entities are included if they were inputs to processes that lead to the current entity or they are output from processes for which the current entity was an input. Kafka processes follow this pattern.

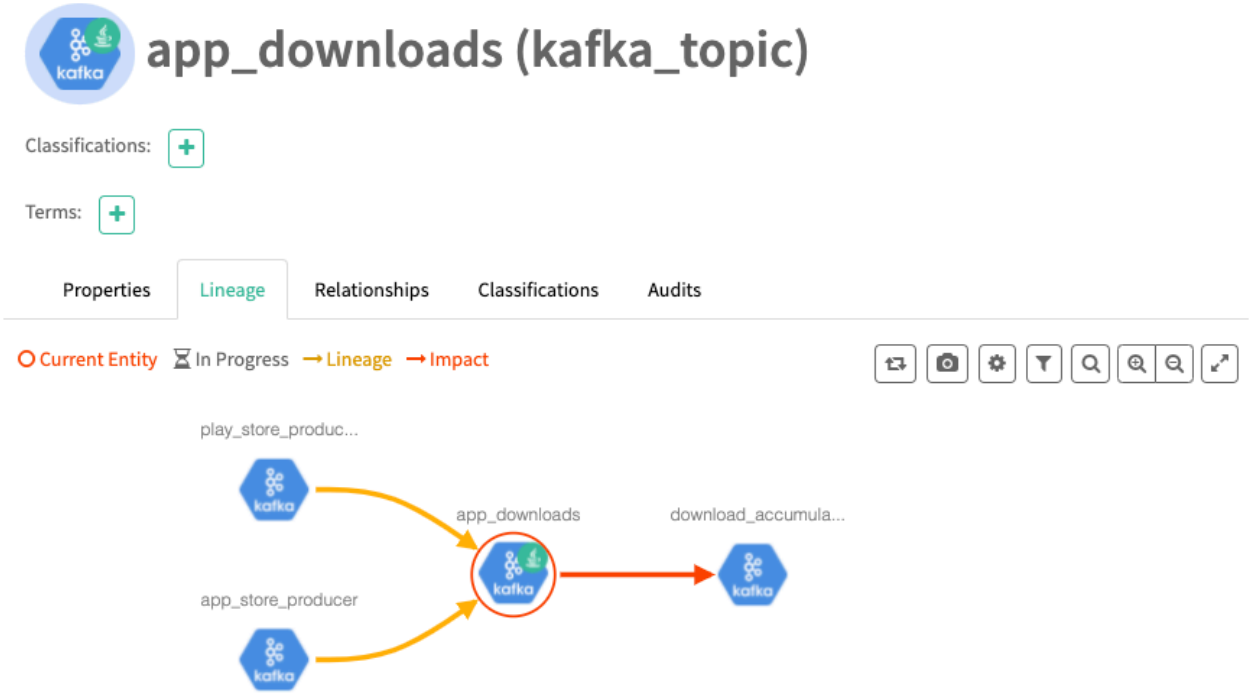

Kafka audit entries

Atlas lists changes to metadata entities in the Audits tab in the Dashboard.

Atlas tracks the lifecycle of each Kafka entity, including its creation, update, and deletion. User access and actions that affect the data content of the source asset are not included in the audit.

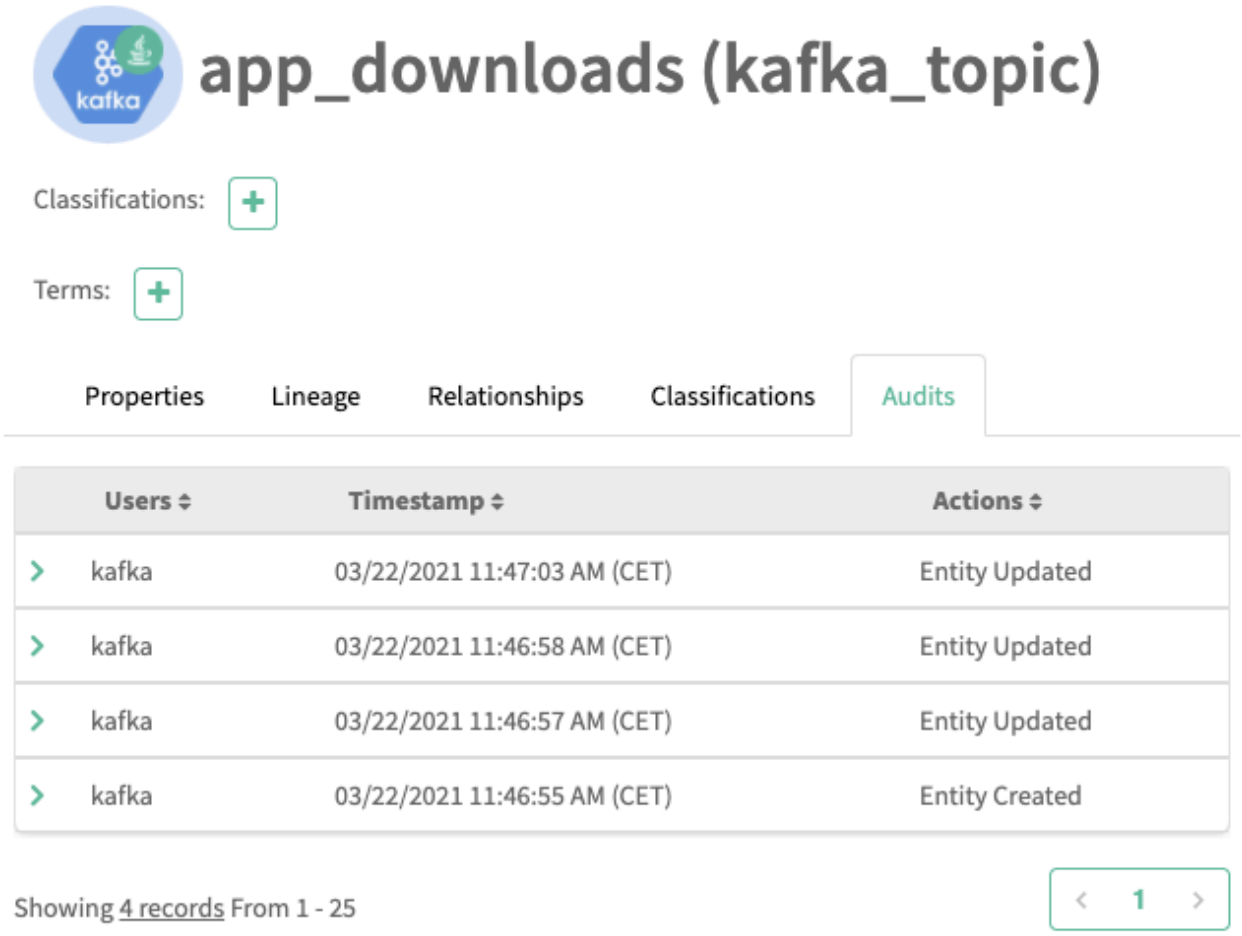

Spark metadata collection

Atlas can collect metadata from Spark, including queries on Hive tables. The Spark Atlas Connector (SAC) is available as of Spark 2.4 and Atlas 2.1 and Spark 3.

An Atlas hook runs in each Spark instance. This hook sends metadata to Atlas for Spark operations. Operations are represented by process entities in Atlas. Hive databases, tables, views, and columns that are referenced in the Spark operations are also represented in Atlas, but the metadata for these entities is collected from HMS. When a Spark operation involves files, the metadata for the file system and files are represented in Atlas as file system paths.

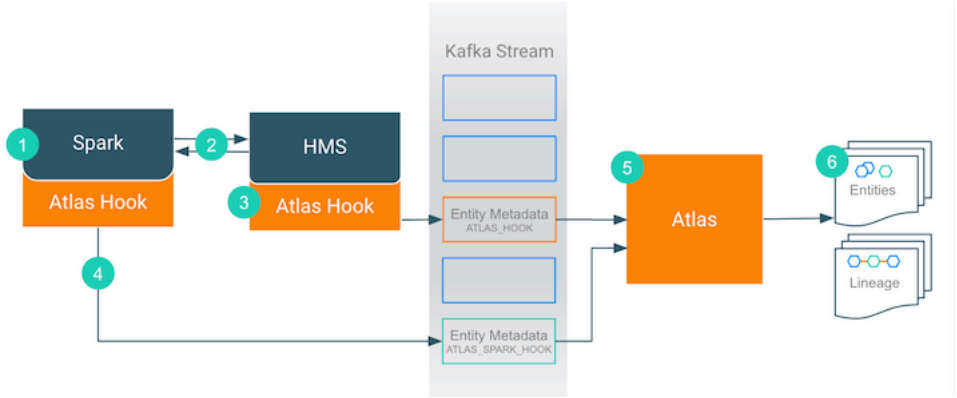

1. When an action occurs in the Spark instance...
2. It updates HMS with information about the assets affected by the action.
3. The Atlas hook corresponding to HMS collects information for the changed and new assets and forms it into metadata entities. It publishes the metadata to the Kafka topic named ATLAS_HOOK.
4. The Atlas hook corresponding to the Spark instance collects information for the action and forms it into metadata entities. It publishes the metadata to a different Kafka topic named ATLAS_SPARK_HOOK.
5. Atlas reads the messages from the topics and determines what information will create new entities and what information updates existing entities. Atlas is able to determine the correct entities regardless of the order in which Atlas receives messages from the Kafka topics.
6. Atlas creates the appropriate entities and the relationships among them and determines lineage from existing entities to the new entities.

Spark actions that produce Atlas entities

Spark jobs create Spark application and process entities and create, update, or delete the data assets affected by those operations will affect Atlas entities; operations that only affect data do not show up in Atlas.

The following table lists the Spark actions that produce or update metadata in Atlas.

This Action in Spark...	...Produces metadata for these Atlas entities
CREATE TABLE USING CREATE TABLE AS SELECT, CREATE TABLE USING ... AS SELECT	spark_application, spark_column_lineage, spark_process, hive_table, hive_column, hive_storagedesc
CREATE VIEW AS SELECT,	spark_application, spark_process, hive_table, hive_column, hive_storagedesc
INSERT INTO (SELECT), LOAD DATA [LOCAL] INPATH	spark_application, spark_process

Notable actions in Spark that do NOT produce process entities in Atlas, meaning that no lineage is produced for these operations:

- LOAD DATA INPATH (when not coming from a local file source)
- CREATE TABLE (hive_table metadata produced by HMS)
- ALTER VIEW (hive_table metadata produced by HMS)
- SELECT or other queries that don’t change table metadata

Spark entities created in Apache Atlas

Each Spark entity in Atlas includes detailed metadata collected from Spark.

The following diagrams show a summary of the entities created in Atlas for Spark operations. The data assets that Spark operations act upon are collected through HMS. The supertypes that contribute attributes to the entity types are shaded.

Figure 5: Atlas Entity Types for Spark Data Sets

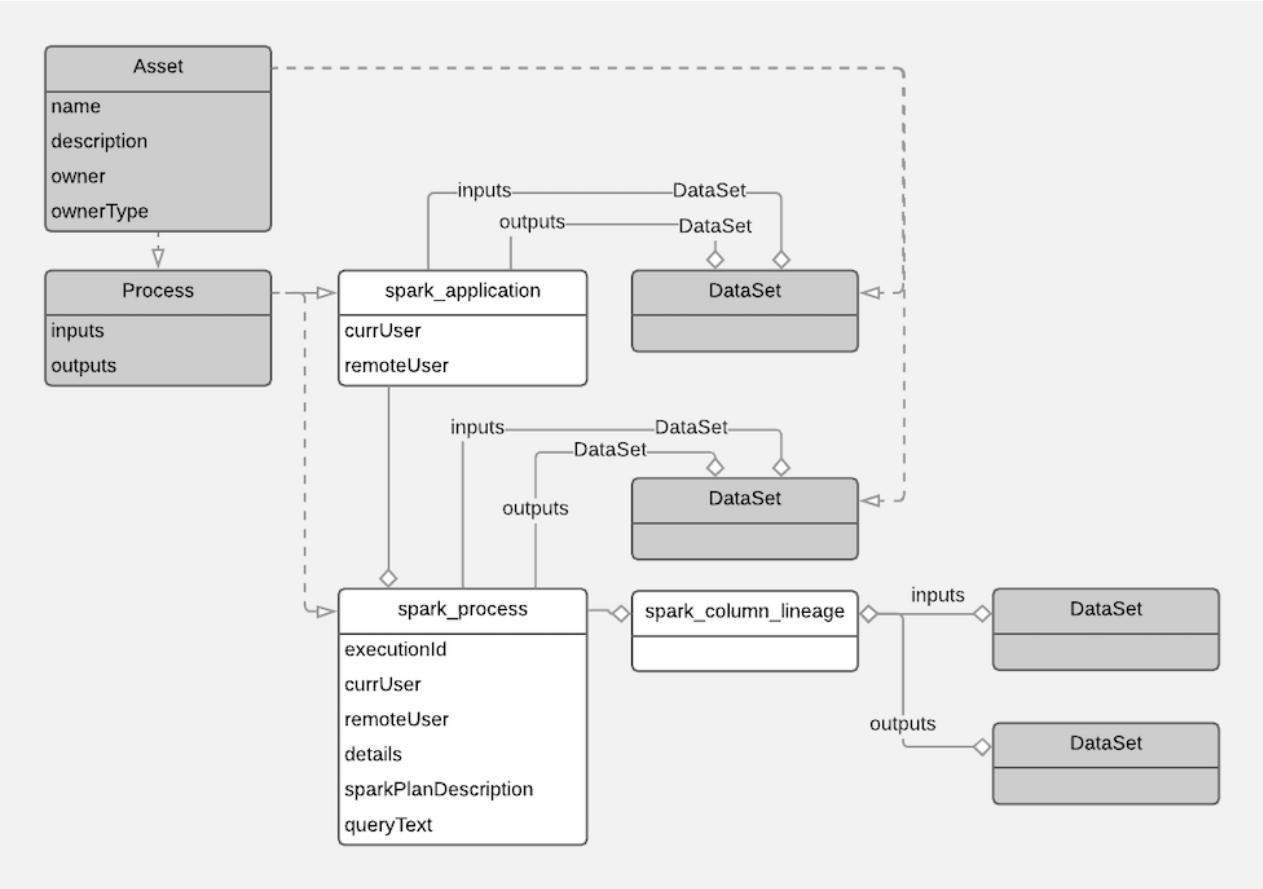

The metadata collected for each entity type is as follows:

Spark Application

Identifier	Example content
typeName	spark_application
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	<Spark application ID>
name	Spark Job + <Spark application ID>
description	Metadata from Spark. Reserved for future use.
displayName	Reserved for future use.
owner	Metadata from Spark. Reserved for future use.
currentUser	Metadata from Spark. In a Kerberized environment, this value contains the principal name.
remoteUser	Metadata from Spark. In a Kerberized environment, this value contains the principal name.
userDescription	Metadata from Spark. Reserved for future use.
replicatedFrom	Reserved for future use.
replicatedTo	Reserved for future use.
Relationship: inputs	Reserved for future use.
Relationship: outputs	Reserved for future use.
Relationship: processes	List of Spark process entities created as part of the processing accomplished in this Spark job.

Spark Process

Identifier	Example content
typeName	spark_process
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	*APPLICATION-ID*-execution-*N* where *N* is a sequential integer assigned by the Spark engine and the application ID is for the parent Spark job.
name	execution-*N* where *N* is a sequential integer assigned by the Spark engine. The number is unique only for the job, so it is possible to have Spark processes with duplicate names in Atlas.
description	Metadata from Spark. Reserved for future use.
owner	Metadata from Spark. Reserved for future use.
details	Metadata from Spark describing the logical plan.
displayName	Reserved for future use.
executionId	Metadata from Spark.
inputs	List of the input tables or views, including each entity's type name and the qualified name.
outputs	List of the output objects, including each entity's type name and the qualified name.
queryText	Metadata from Spark. Reserved for future use.
currUser	Metadata from Spark. In a Kerberized environment, this value contains the principal name.
remoteUser	Metadata from Spark. In a Kerberized environment, this value contains the principal name.
executionTime	Metadata from Spark.
details	Query plan text, including parsed logical plan, analyzed logical plan, optimized logical plan, and physical plan.
sparkPlanDescription	Physical plan text.
replicatedFrom	Reserved for future use.
replicatedTo	Reserved for future use.
userDescription	Metadata from Spark. Reserved for future use.
Relationship: inputs	List of the input tables or views, including each entity's type name and the qualified name.
Relationship: outputs	List of the output objects, including each entity's type name and the qualified name.
Relationship: application	The Spark application entity that describes the Spark job in which this process was created.

Spark Column Lineage

Identifier	Example Content
typeName	spark_column_lineage
guid	System generated ID. This value is used to identify the entity in the Atlas Dashboard URL.
qualifiedName	*<OUTPUT_TABLE_NAME>-<TIMESTAMP>-<OUTPUT_COLUMN_NAME>*
name	Same as qualifiedName.
Relationship: Process	Name of the spark_process entity that produced this lineage. spark_process_column_lineages
Relationship: inputs	List of the input columns, including each entity's type name and the qualified name.
Relationship: outputs	Output column, including each entity type name and the qualified name.

Spark lineage

Atlas collects metadata from Spark to represent the lineage among data assets.

The Atlas lineage graph shows the input and output processes that the current entity participated in, specifically those relationships modeled as “inputToProcesses” and “outputFromProcesses.” Entities are included if they were inputs to processes that lead to the current entity or they are output from processes for which the current entity was an input. In the context of Spark, a Spark job is modeled as a spark_application entity. Each application entity includes relationships to one or more processes that were executed in the job. The spark_process entities are automatically named "execution-*N*" where *N* is an integer incremented sequentially.

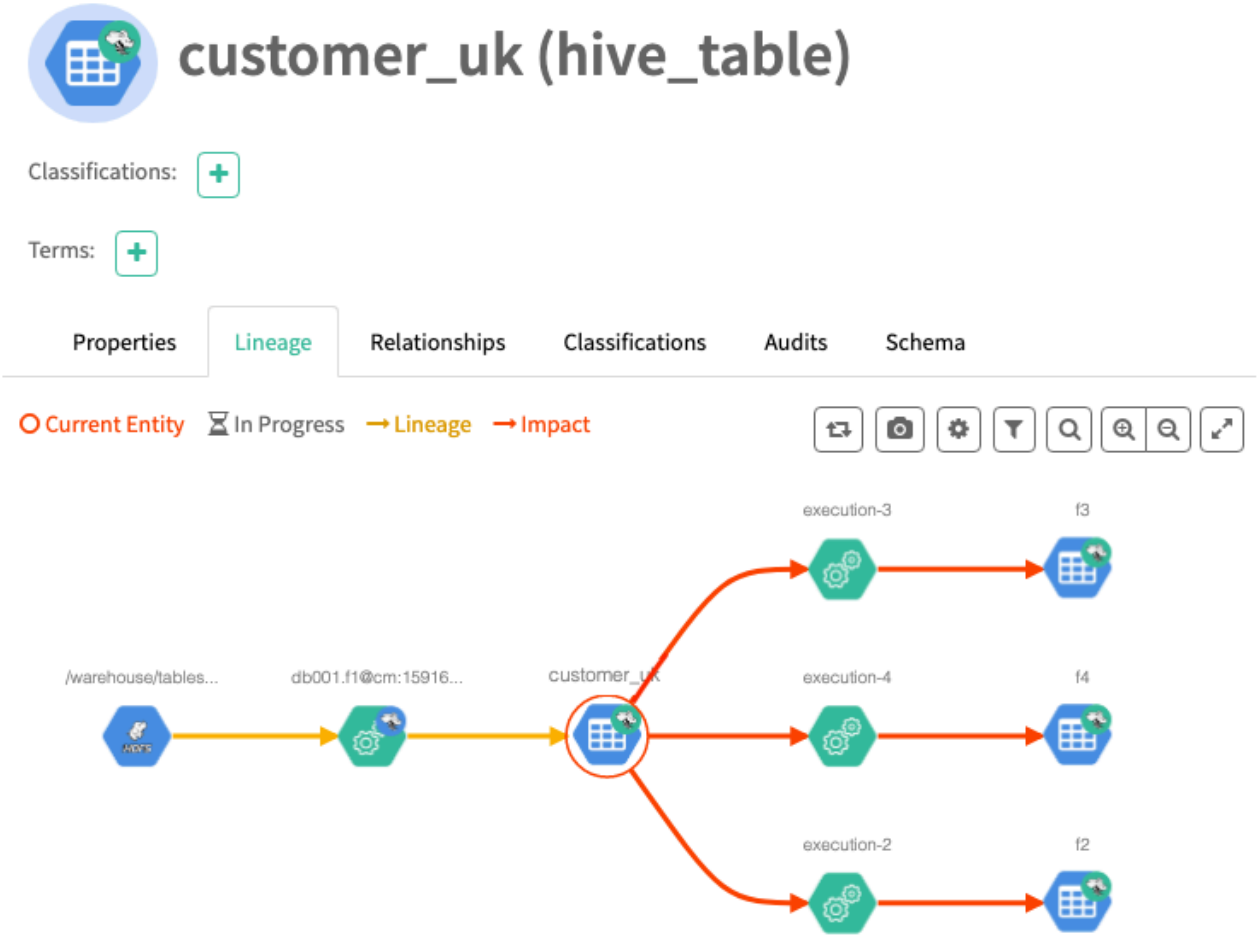

It is possible to have two spark process entities with the same name in a lineage graph; be sure to check the qualified name to make sure you are looking at the appropriate process.

Related Information

Viewing lineage

Spark relationships

Atlas shows the entities related to this entity in the Relationships tab in the Dashboard.

The Relationship tab shows the relationships that exist for an entity. Use this view to navigate among related entities.

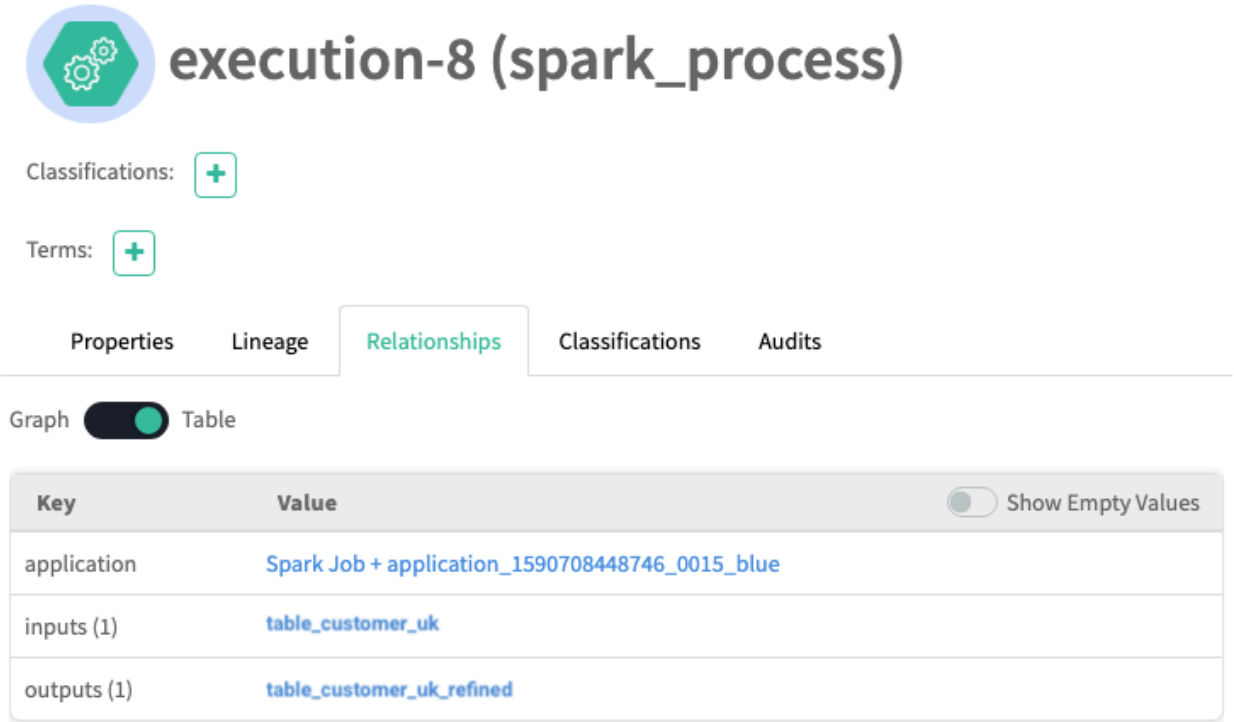

Spark audit entries

Atlas lists changes to metadata entities in the Audit tab in the Dashboard.

Atlas tracks the lifecycle of each Spark entity, including its creation, update, and deletion. Note that if you change the name of an application in Atlas, the change will cause each related Spark process entity to be updated. User access and actions that affect the data content of the source asset are not included in the audit.

Spark Connector configuration in Apache Atlas

Learn to configure the Spark Atlas Connector so that Spark jobs can run even when Kafka brokers are down. This ensures that your job submissions do not fail.

Note: The Spark Atlas Connector is available in Cloudera Public Cloud from version 7.2.18 and in CDS Powered by Apache Spark from version 3.4.1 from 7.2.18.

Before you begin the configuration, you must enable the following properties in Cloudera Manager:

Atlas Service:

1. Log in to Cloudera Manager.
2. Select the Spark service.
3. Select the Configurations tab.
4. Search for and select the Atlas Service (atlas_service) parameter.
5. Click Save Changes.

Spark Lineage:

1. Log in to Cloudera Manager.
2. Select the Spark service.
3. Select the Configurations tab.

4. Search for and select the Spark Lineage (spark.lineage.enabled) parameter.
5. Click Save Changes.

Using the spark.lineage.kafka.fault-tolerant.timeout.ms parameter

If all Kafka brokers are down, the deploy mode is cluster mode without the keytab/principal. Spark Atlas lineage is also enabled which does not allow a delegation token from the Kafka broker. Therefore, the job submission fails. As a result, --deploy-mode is set but --principal PRINCIPAL and --keytab KEYTAB are not set. Use spark.lineage.kafka.fault-tolerant.timeout.ms parameter to resolve this case.

The default timeout value for the Kafka delegation token creation is 0. This means that the fault tolerant mode is disabled. If the value is greater than 0, the fault tolerant mode is enabled with the configured timeout value.

See the example PySpark code below:

```
from pyspark.sql import SparkSession
import time
import sys
id_string = str(time.time()).split(".")[0]
spark = SparkSession.builder.appName("Python CTAS example").getOr
Create()
spark.sql("CREATE TABLE parquet_spark1_" + id_string +" ( id bi
gint, data string, category string) using parquet")
spark.sql("CREATE TABLE parquet_spark2_" + id_string +" USING pa
rquet AS SELECT * from parquet_spark1_" + id_string)
```

Example commands:

```
spark-submit --conf spark.lineage.kafka.fault-tolerant.timeout.m
s=8000 --deploy-mode cluster test.py
```

```
spark3-submit --conf spark.lineage.kafka.fault-tolerant.timeout.
ms=8000 --deploy-mode cluster test.py
```

Limitation:

This timeout will permit the job submission in cluster mode if all Kafka Brokers are down. There will be no Spark Atlas Connector lineage because we do not have Kafka delegation token which is required for the lineage creation.

Spark troubleshooting

What do you do if you do not see Apache Atlas metadata from Spark or face any related issues.

Spark runs an Atlas "hook" or plugin called Spark Atlas Connector (SAC) on every host where Spark runs. To troubleshoot problems, consider the following methods for narrowing down where the problem is:

- Are you missing all metadata?

  Make sure that all the services supporting Atlas are configured and running. For Cloudera, the configuration is done for you; look in Cloudera Manager to see that Kafka, Solr, and Atlas services are running in the data lake.
- Are you missing all Spark process metadata?

  By default, Spark operations are configured to send metadata to Atlas. To check that these settings have not been rolled back, look at the Spark On YARN service configuration page in Cloudera Manager to ensure that Spark is configured to send metadata to Atlas (Atlas Service property). Assuming this configuration is enabled, you can next check the Kafka topic queue to make sure that metadata messages are being produced in Spark and making it to the Kafka topic.

- Missing only some Spark metadata?

  Because each instance of Spark collects metadata independently of other instances, it is possible that one instance failed to send metadata to Atlas. To determine if this is the problem, check the Kafka topic queue to see if one of the Spark hosts is not sending metadata.
- When customer is running a standard Spark structured streaming application in DataHub Data Engineering cluster, the Spark job might throw exceptions in the YARN App logs.

  For example:

  WARN clients.NetworkClient: [Producer clientId=producer-1] Error while fetching metadata with correlation id 1 : {ATLAS_HOOK=TOPIC_AUTHORIZATION_FAILED} ERROR hook.AtlasHook: Giving up after 3 failed attempts to send notification to Atlas

  The Spark application tries to publish the metadata to the Kafka topic named ATLAS_HOOK. At that stage it was failing because of the permission issue for the spark application user. For example: ‘csso_jonathan’.

  To overcome this permission issue, you must provide the Ranger permission for the application user 'csso_jonathan’ using following steps:

  Go to Ranger Admin> Kafka policies> Provide "publish" permission to "csso_jonathan” user for the following:
- ATLAS_HOOK
- ATLAS_ENTITIES
- ATLAS_SPARK_HOOK policies.

Later, re-run the Spark job and verify if the Atlas related errors are still visible in the application logs.

Cloudera Runtime 7.3.1

Atlas Search

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Basic search **4**
Basic search enhancement 5

Using Relationship Search **6**

Using Search filters **12**

Ability to download search results from Atlas UI **14**
How to download results using Basic and Advanced search options 15

Using Free-text Search **17**

Enhancements with search query **19**

Ignore or Prune pattern to filter Hive metadata entities **21**
How Ignore and Prune feature works 22
Using Ignore and Prune patterns 23

Saving searches **24**

Using advanced search **25**

Atlas index repair configuration **28**

Using Basic search

Search using an entity type in Basic Search.

With **Basic Search**, Atlas returns all of the entities of the type you select.

There are many ways you can define a **Basic Search**. Setting a value in more than one search field builds a logical AND condition for the search. To repeat the same search, click the Refresh button.

Search By Type

- Choose an entity type to limit the search.
- Choose _ALL_ENTITY_TYPES to apply an attribute filter across all entity types.

Search By Type and specify attribute values using the Entity Attribute Filter

The **Attribute Filter** dialog box lists all the attributes that correspond to the selected entity type, including:

- Technical attributes specific to the entity type
- System attributes, including classifications, labels, and user-defined properties
- Business Metadata attributes
- Terms

Note: When searching on classifications in the Search By Type filter, use "contains" rather than "=" for the filter operator. If there is more than one classification assigned to an entity, "contains" matches a single classification name; equal only matches the entire list of classifications names.

Search By Classification

- Choose an existing classification; the search returns all entities that have that classification assigned to them.
- Choose _ALL_CLASSIFICATION_TYPES to apply an attribute filter across all classifications.
- Choose _CLASSIFIED or _NOT_CLASSIFIED with an entity type selected to find entities of that type with any or no classifications assigned.

Search By Classification and specify attribute values using the Filter

The **Attribute Filter** dialog box lists all the attributes for the selected classification; set a value to one or more attributes to define the search. You can choose to match partial strings using the "contains", "begins with", and "ends with" operators.

Search by Term

Choose an existing glossary term. You can enter the first few letters to select a term from a list of matching terms. This filter is case-sensitive.

Search by Text

Search on string values for technical, system, Business Metadata, and classification attribute values. Labels and terms are also included. This search is the same as the Free-Text search; note that when you enter text in the Free-Text search box, it fills in this Search By Text field also.

You can also save these searches when they are useful to run more than once.

Related Information

Using Free-text Search
Searching for entities using Business Metadata attributes
Searching for entities using terms
Searching for entities using classifications
Saving searches
Apache Atlas metadata attributes
Using Search filters

Basic search enhancement

While performing basic search in Atlas, you can exclude header attributes of entities from the response to reduce latency.

The Basic Search feature in Atlas has AtlasEntityHeader data type of each entity in the response.

The AtlasEntityHeader data type has multiple attributes including classification and terms. AtlasEntityHeader requests the Janusgraph database to provide the information for each attribute. This process can be time consuming increasing the response latency.

To overcome this situation, you can add a flag to exclude generic attributes and add only the selected attributes from the attributes field in the response.

In the request payload, including the following improves the search experience:

- Attributes having entityTypes
- excludeHeaderAttributes=true
- Valid entity attributes (not relationship) in the attributes field

Note: The excludeHeaderAttributes attribute overrides other attributes' fields such as the includeClassificatio nAttributes in the request payload.

An example payload request:

Request

```
{
"excludeDeletedEntities": true,
"includeSubClassifications": true,
"includeSubTypes": true,
"includeClassificationAttributes": true,
"limit": 25,
"offset": 0,
"typeName": "hdfs_path",
"attributes": ["path", "name"],
"excludeHeaderAttributes": "true"
}
```

Response

```
{
"queryType": "BASIC",
"searchParameters":
{ "typeName": "hdfs_path", "excludeDeletedEntities": true, "includeClassific
ationAttributes": true, "includeSubTypes": true, "includeSubClassifications"
: true, "limit": 25, "offset": 0, "attributes": ["path", "name"] }
,
"attributes": {
```

```
"name": ["path", "name"],
"values": [
["/data/warehouse/customer", "customer"],
["/data/warehouse/sales", "sales"]
]
},
"approximateCount": 2
}
```

Using Relationship Search

Entities in Atlas can be searched based on the relationships that describe various metadata between a couple of entity end-points.

Important: By default, the Relationship Search feature is disabled in Atlas to reduce the start and restart time of the service.

As an example, a relationship between entities hive_table and hive_db can be attributed or defined as hive_table_db, which has a standalone metadata that can be added as an attribute to this format. By searching for hive_table_db, you can retrieve the relationship between hive_table and hive_db entities. This enhancement ensures that those relationships which are tied to the entities and that match the filter criteria on attributes of the relationships, can be searched.

Enabling relationship search

You can enable Relationship Search in Cloudera Manager under Atlas Server Advanced Configuration Snippet for conf/atlas-application.properties by setting the atlas.relationship.search.enabled property to true.

Until the property is enabled, Relationship Search is not visible in the UI:

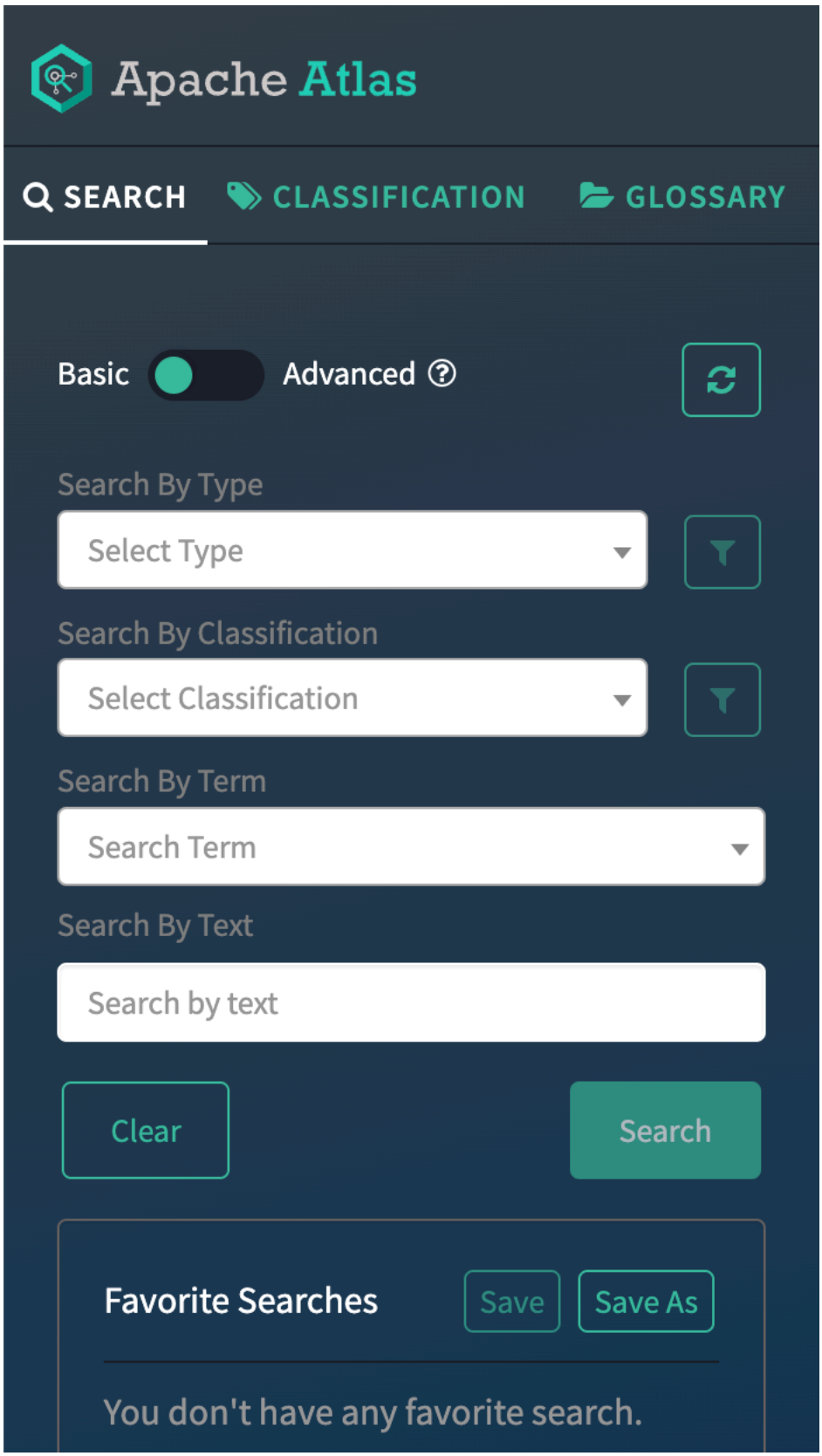
Apache Atlas
SEARCH
CLASSIFICATION
GLOSSARY
Basic
Advanced
Search By Type
Select Type
Search By Classification
Select Classification
Search By Term
Search Term
Search By Text
Search by text
Clear
Search
Favorite Searches
Save
Save As
You don't have any favorite search.

Configuring Relationship Search

For the entities to be searchable in the relationship definition model, the attributes must be added and marked as indexable before starting the Atlas service.

Use the POST API typedef for the following configuration:

As an example of the configuration setup is as follows:

```
{
“attributeDefs”: [{
“Name”: “edge_property1”,
“isIndexable”: true
}]
```

Note:

The following API calls are enabled if the isIndexable flag is set to true:

- GET/POST /v2/search/relations
- GET /v2/search/relationship

These API calls are not available by default.

In Atlas, once the atlas.relationship.search.enabled property is set to true, you can switch between Entity Search and Relationship Search:

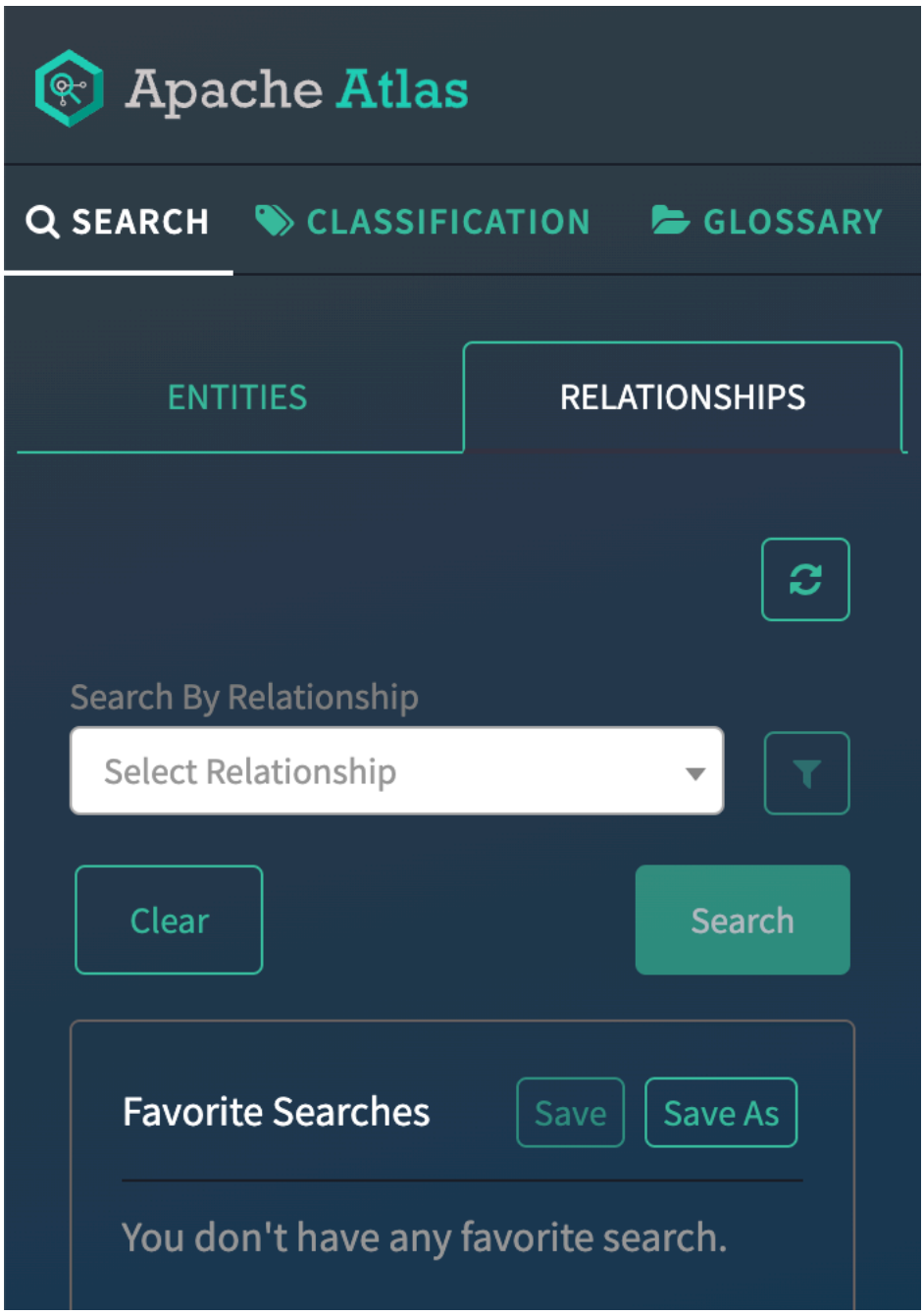

You can search for the relationship using the drop-down list to select a relationship between entities and explore them in Atlas.

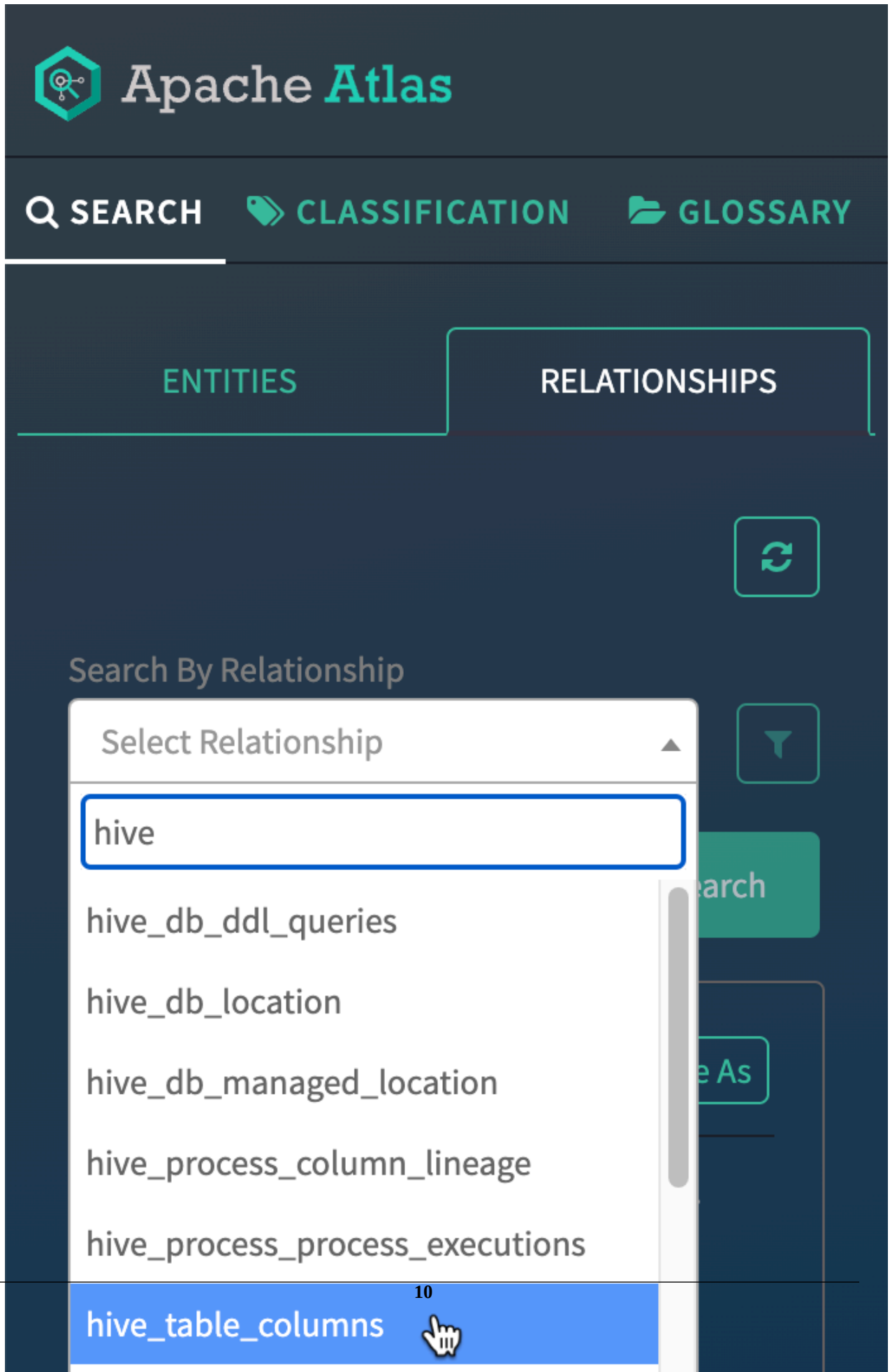
Apache Atlas
SEARCH
CLASSIFICATION
GLOSSARY
ENTITIES
RELATIONSHIPS
Search By Relationship
Select Relationship
hive
hive_db_ddl_queries
hive_db_location
hive_db_managed_location
hive_process_column_lineage
hive_process_process_executions
hive_table_columns

The list of entities with the type hive_table_columns contains additional information about the connected entities.

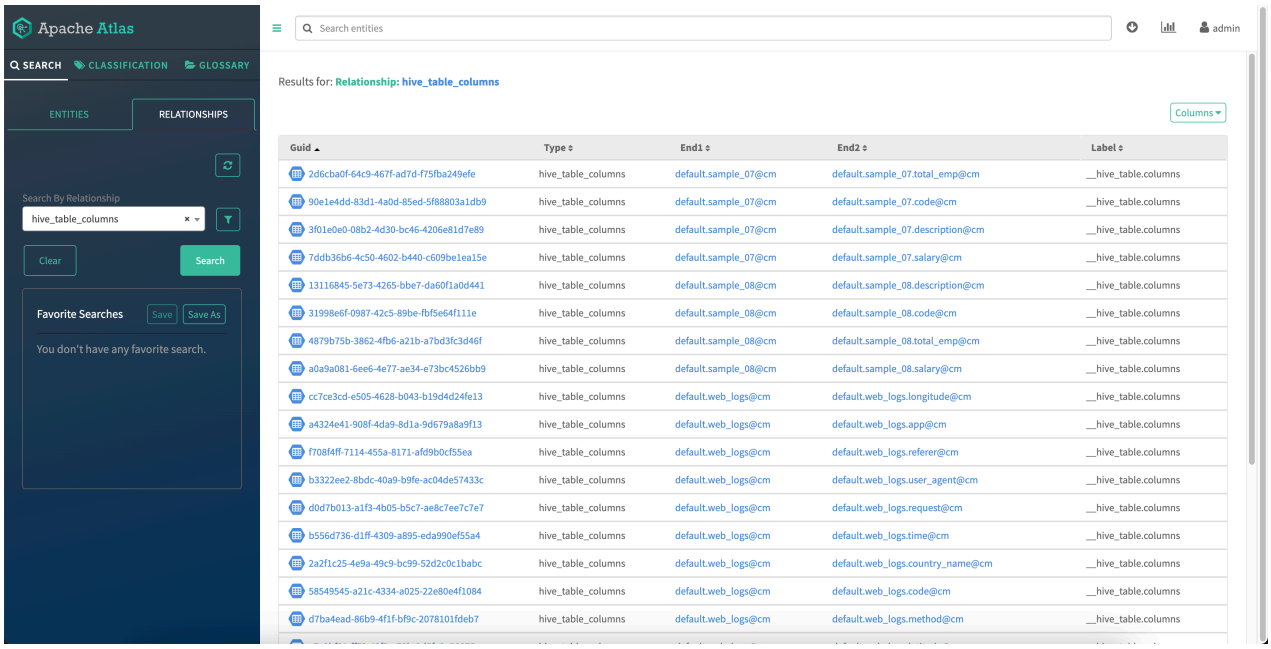

You can add specific filters to search for relationship entities.

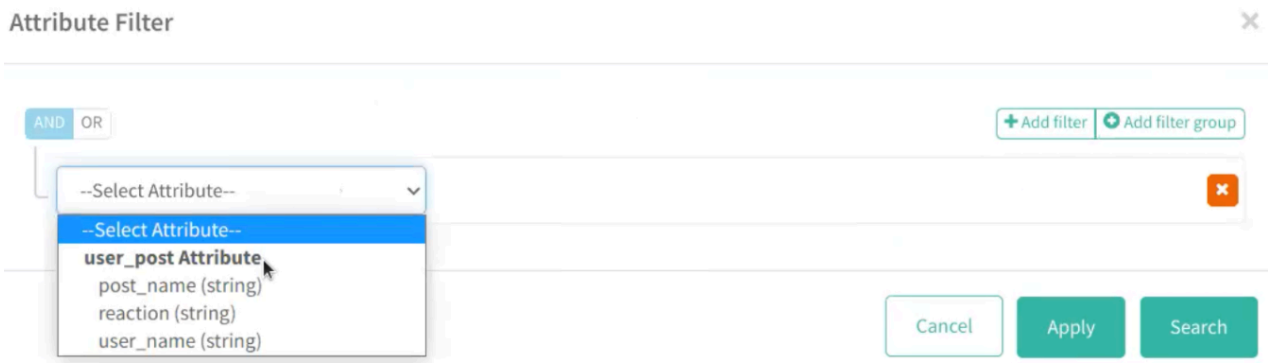

In the example image, end 1 and end 2 are the entities with which the relationship is created.

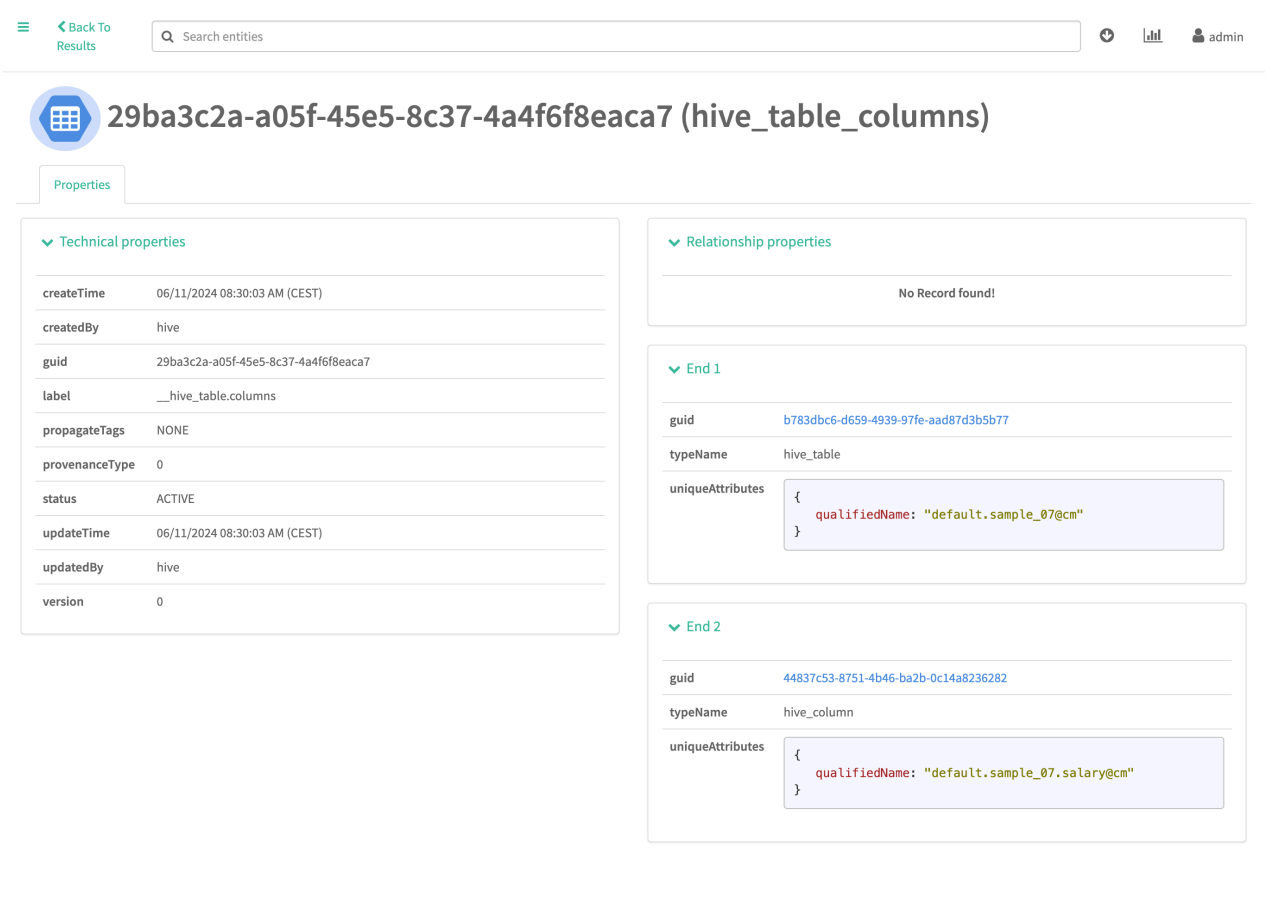

By clicking on each search result, you can navigate to the details page to view additional details and pertinent information.

Using Search filters

The Basic Search panel includes filter icons that allow you to search for entities based on one or more attribute values.

In a filter row, the attribute data type determines which of the following operators can be used to define your search criteria:

Strings	Dates	Enumerations Boolean	Numerics
=		=	=
!=		!=	!=
	>	>	>
	<	<	<
is null	is null	is null	is null
is not null	is not null	is not null	is not null
contains			
begins with			

Strings	Dates	Enumerations Boolean	Numerics
ends with			

All classification attributes are string values; numerics include byte, short, int, float, double, and long attribute data types.

Note: If the attribute you are searching for could include multiple values, use "contains" rather than "=" to make sure the search finds the individual value out of the list.

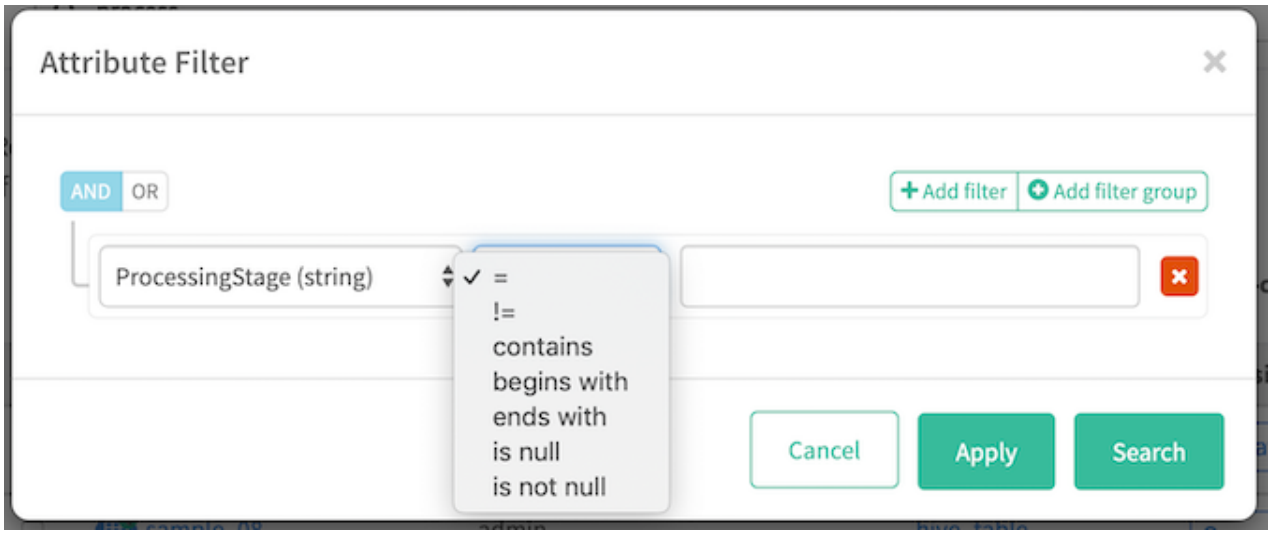

To search on values for more than one attribute, add another filter row to the search filter (click Add filter). The search can find entities matching either filter criteria (logical OR) or matching both criteria (logical AND). Set the logic using the AND / OR buttons at the top-left of the filter rows.

You can combine logical AND and OR criteria using filter groups. The logic is the same within a filter group; use more than one filter group to produce both AND and OR logic. For example, the following Classification attribute filter searches for entities that are at "new" or "acknowledged" stages in their processing and are owned by the Finance business team.

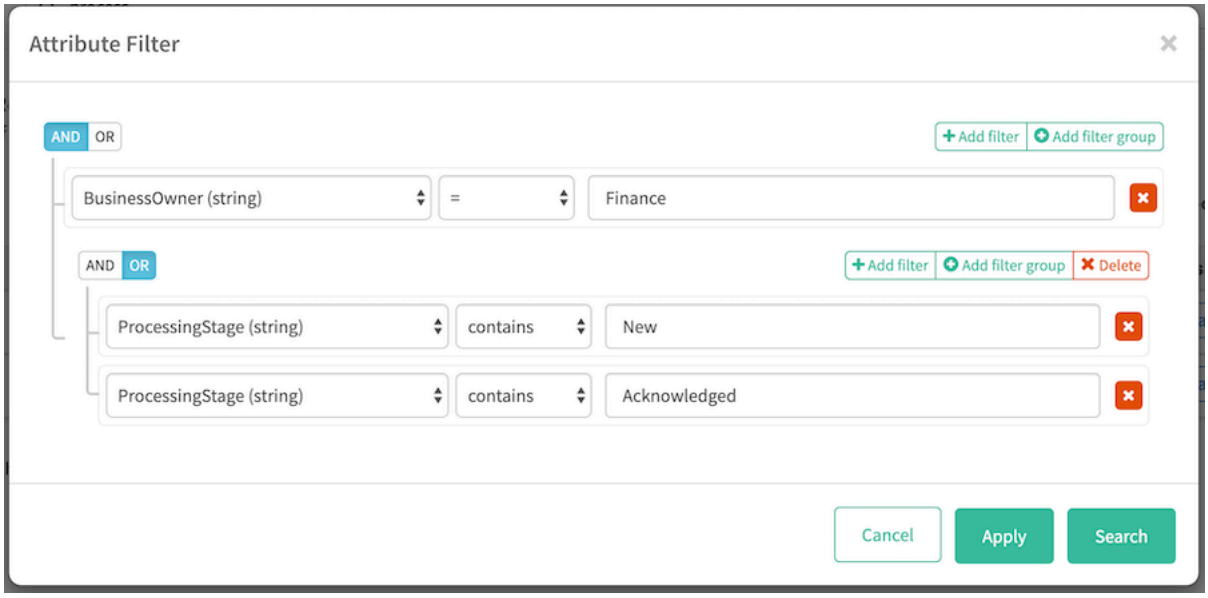

If you wanted to further limit the search results to the entities that were created in Atlas in the last 24 hours, you would open the attribute filter for Search by Entity Type and set the system attribute "Created Timestamp" less than 24 hours. To open the Search by Entity Type filter, you would need to select an entity type or "_ALL_ENTITY_TYPES".

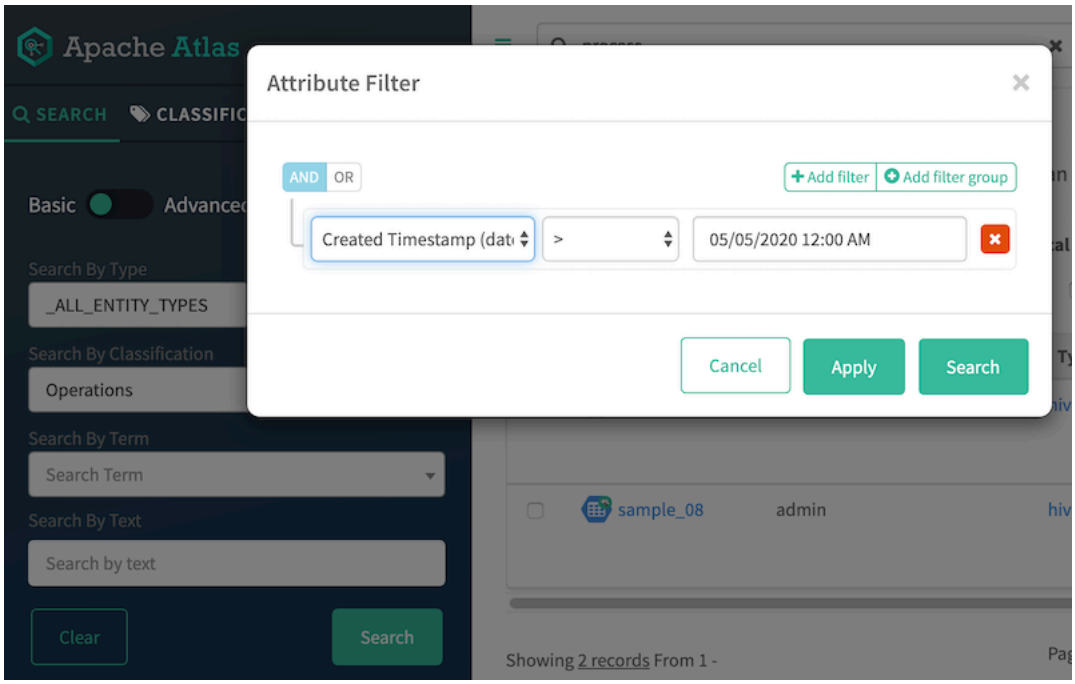

Ability to download search results from Atlas UI

Atlas supports improved search results capabilities. You can download search results for both basic and advanced search options. You can configure the parameters to specify a directory path and configure automatic cleaning up of files to manage space efficiently.

You can use the new Download option to capture the search results of both Basic and Advanced (DSL) searches, in CSV file format. The CSV files are stored on the local file system. This download option queues up the search results and downloads the files in the following format: User type_type_of_search_timestamp. For example: admin_DSL_2023-04-29_04-56.csv indicates that the file contains results of search performed using the advanced search option.

Atlas employs the API path to route the requests to queue up the search results and later convert them to CSV files. The following are commonly used for the search and download operations.

- /atlas/v2/search/<basic/dsl>/download/create_file
- /atlas/v2/search/download/status - Status API
- /atlas/v2/search/download/{filename} - Download request API

Search results configuration

You can configure the search result size The default value of search result size is 10000 and only 50 simultaneous requests can be done. You can configure the search request using the atlas.download.search.max.pending.tasks property.

You can configure the parameters in such a way that the downloaded CSV files are available in a specified directory path for a fixed time period and they are automatically cleaned up to free up the storage space. You can search for specific type definitions in Atlas and download them as appropriate.

The following APIs can be used to manage the download options and can be included in the **Configuration** tab in the Cloudera Manager instance under Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties.

- atlas.download.search.dir.path - Use this parameter to define the path for storing the downloaded CSVs.
- atlas.download.search.file.expiry.millis - Use this parameter to set the time limit for the downloaded file to live in the storage. As an example: 180000 milliseconds # 3 minutes.

Note: For every logged in user, a separate directory is created automatically under the path defined by the atlas.download.search.dir.path property and the user's requested search files are stored in that directory. Users can only download files created by them using UI / API.

How to download results using Basic and Advanced search options

Using the Basic search option you can search and download the results. You must select the type of query and search for the results and later click the Download option.

The search query queues up the results. Click to view the resultant queue.

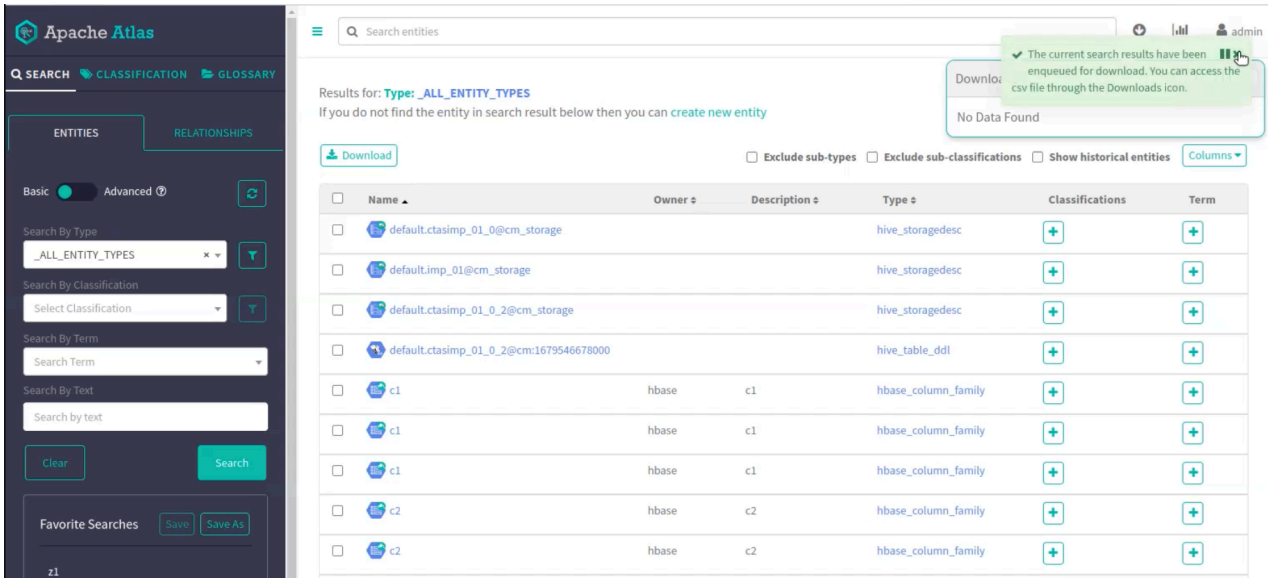

You can expand the scope of the downloaded search results by using the Column drop-down menu to select the columns that you want to get include in the CSV files.

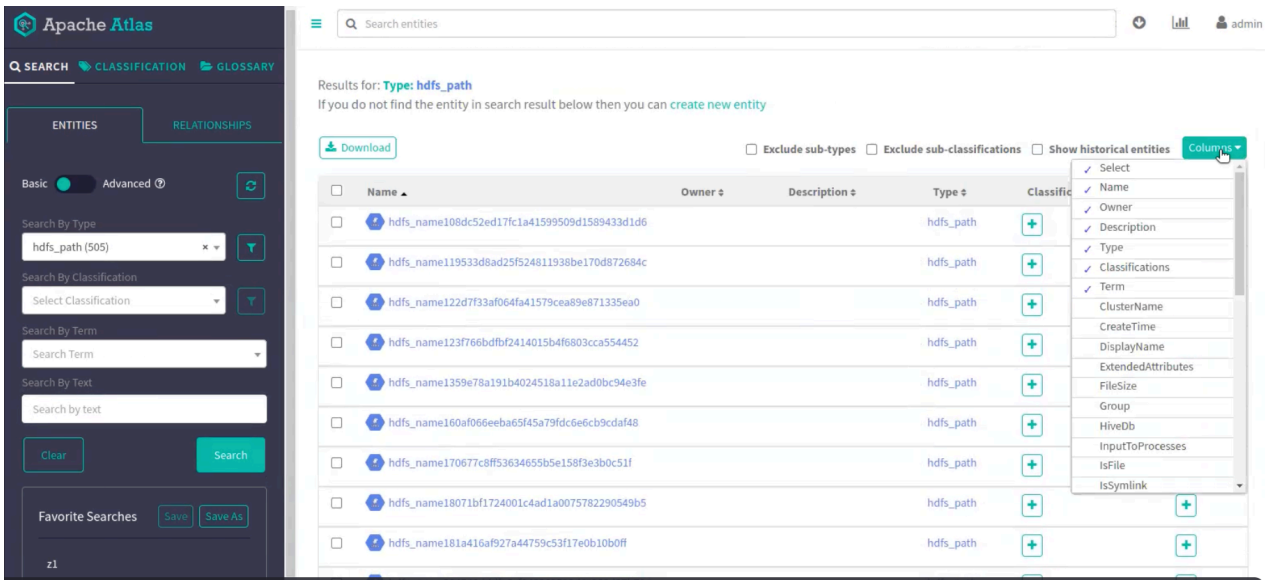

You can search using the Advanced option and download the search results. For example, you can search by using the name and owner criteria. Click Download and follow the same process as described while performing the Basic search operations.

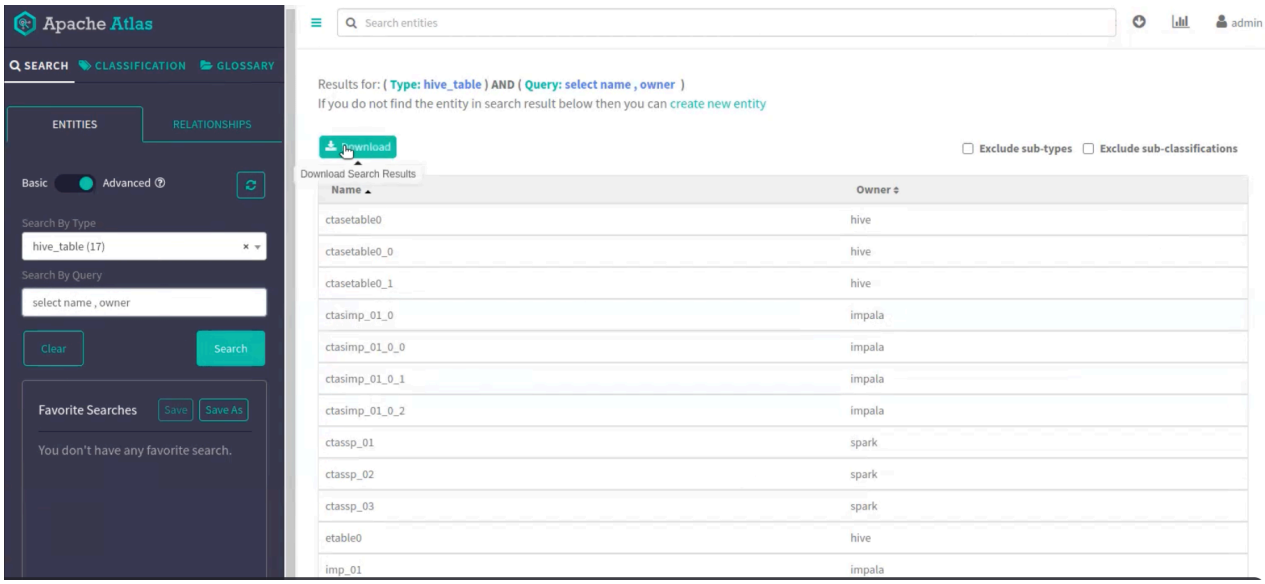

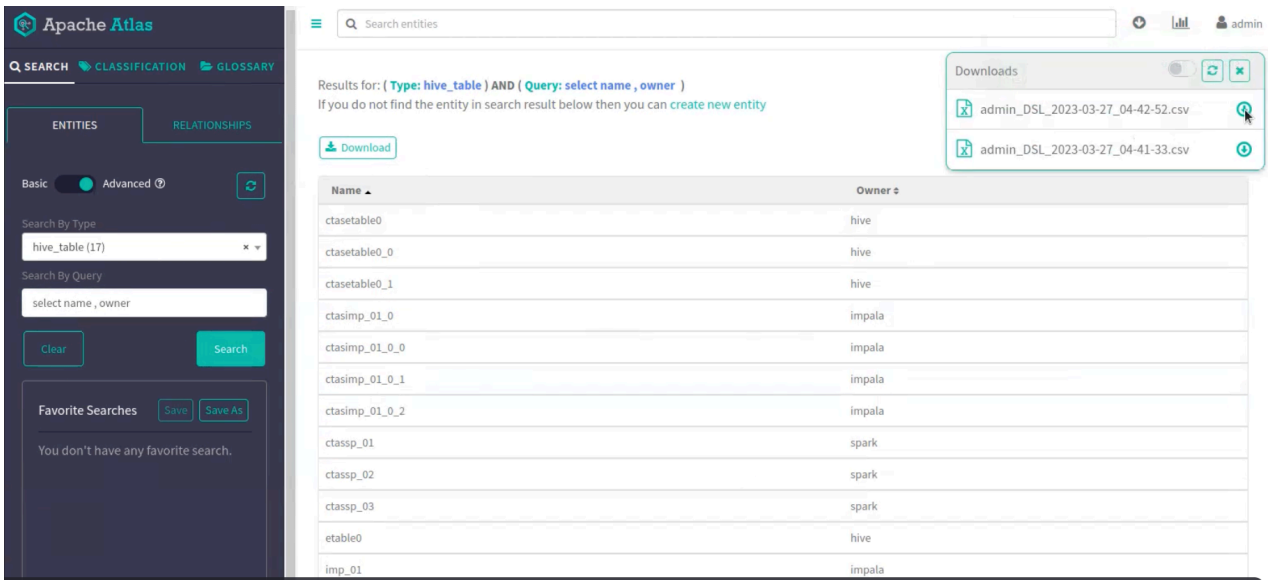

Using Free-text Search

Apache Atlas builds a ranked index of metadata values so you can search for values across all metadata.

The search field in the top of the dashboard lets you search for entities, classifications, or terms by entering any full or partial text to match any entity metadata values. Atlas searches all metadata fields that have string data types, so you can use this search tool to find entities by their labels, descriptions, locations, or other metadata.

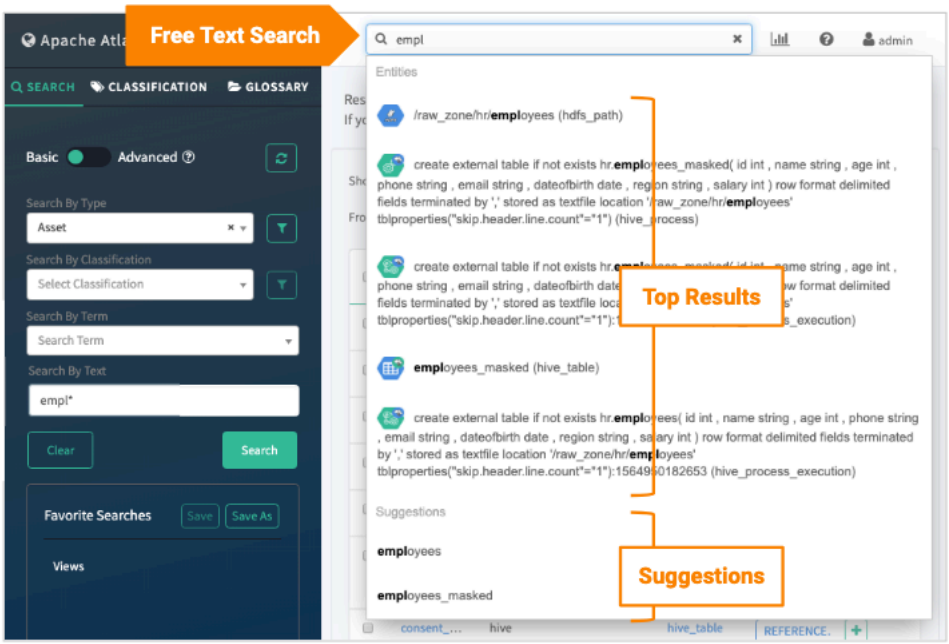

Searches are case insensitive. You can add the asterisk (*) wildcard to the search term's start or end or terms to find partial strings anywhere they occur in the metadata value.

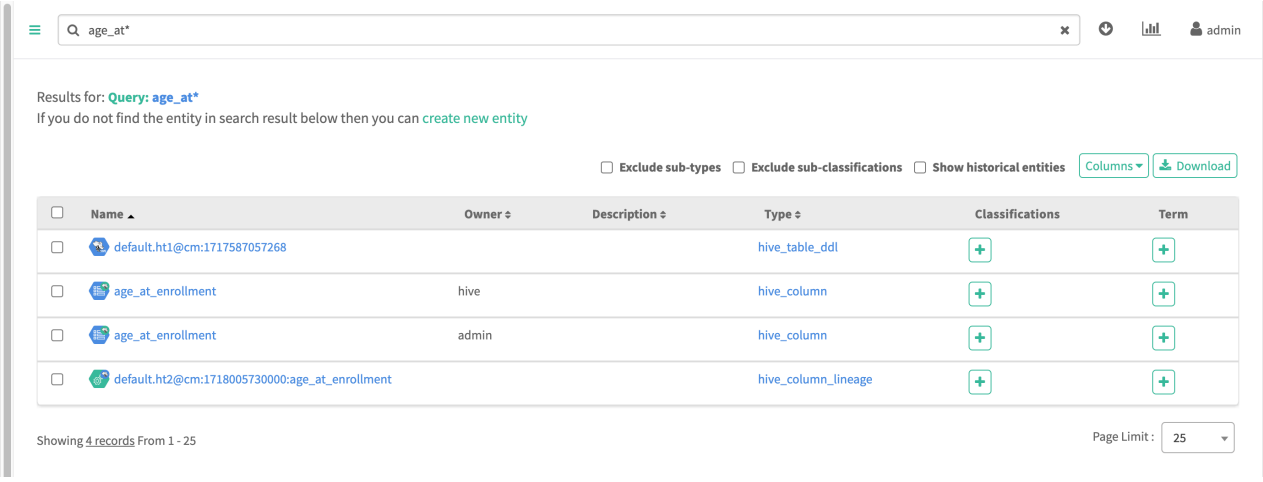

The following single and double characters have special meaning:

```
+ - && || ! ( ) { } [ ] ^ " ~ * ? : \
```

If your search string includes one of these characters, surround the string in double quotation marks or escape the special character with a backslash.

You can see that the search terms you use in the search at the top of the dashboard are also inserted into the free text search field in the left Search panel: you can combine the free text search with other selections to narrow the search results. The combination acts as an “AND” with other search criteria.

Search result ordering: The search results are ranked by how well they match the search terms, with entities that match on more than one metadata value being ranked higher.

Different metadata have different scores, where the highest scoring metadata fields are entity names, including Kafka topic names. Descriptions, users/owners, query text, and comments rank next. Locations, namespaces, domains, etc. come next. Search results are not ordered in any specific way among results that have the same search ranking.

Suggestions: As you enter your search text, you see the five highest-ranked matching items and as many as five suggestions.

The matching items are ranked in the same way as the general search results, case-sensitive (at the moment) terms that “contain” the search terms; If there are more than five search results with the same search ranking, the five shown are randomly ordered from the highest scoring results.

The suggested items are chosen from search results that match with a “starts with” behavior.

Limitations

The suggestions under the Free-text Search field work independently from the search results. Also, certain languages have different behaviors when they are used for searching:

Language or Alphabet	Limitations

• English and languages with Latin alphabets
• Chinese
• Japanese
• Japanese Kana
	• Searching entity names (indexable attributes)
	  • Suggestions are provided for multicharacter searches.
	  • Partial search terms return results.
	• Searching descriptions or comments and other (non-indexable attributes)
	  • Suggestions are not provided for multiword searches.
	  • Partial search terms return results.

Language or Alphabet	Limitations

• Korean	• Searching entity names (indexable attributes)
	  • Suggestions are provided for multicharacter searches.
	  • Partial search terms do not return results. Only partial suggestions return search results.
	• Searching descriptions or comments and other (non-indexable attributes)
	  • Suggestions are provided for multicharacter searches.
	  • Partial search terms do not return results. Only partial suggestions return search results.
	Note: For Korean, you need to add manually the asterisk character to enable partial search results. However, suggestions are provided for partial searches if your search matches the start of an existing term.

Enhancements with search query

When you perform search operations in Atlas, note some of the new changes that are in effect.

First use case

Performing freetext / quick search does not require pre-fixing the attribute name in the search text query.

You can directly search for the entities by searching with its value.

For example:

To search entity having:

qualifiedName="default.avbzi_random_table@cm"

You can directly add "default.avbzi_random_table@cm" in the search bar.

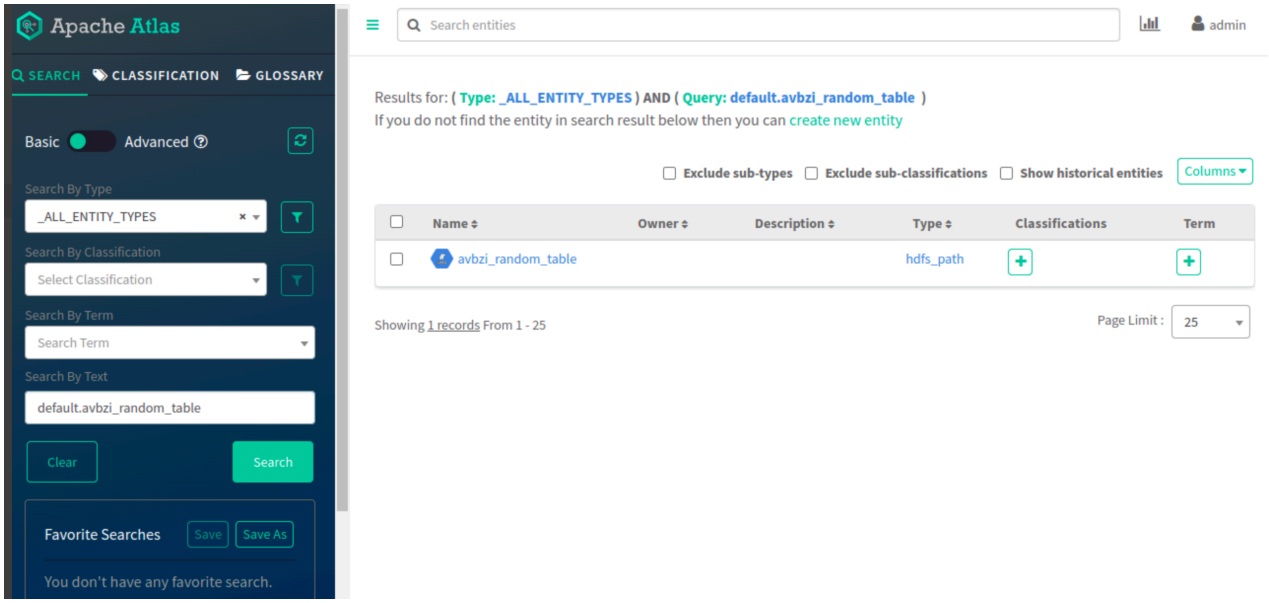

Second use case

While using the basic / quick search in Atlas, the characters which are not alphabetic and numeric are considered as special characters except for the following characters:

_, ., :, and ‘

Note: When using ., :, and ‘ characters, if the prefix or suffix does not contain alphabet character, it is considered a special character.

For example:

default.1marketing > . will be considered as a special character because suffix is numeric not alphabetic

default.marketing > . will not be considered as a special character because prefix and suffix are alphabetic

Additionally, while performing the search operation, when the search string has special characters, SOLR tokenizes the string enabling the search result query to provide OR condition of each tokenized string.

For example: if search string is provided as default.avbzi_random_table@cm

results in, picking up all the matching documents or attributes, where, attribute contains default.avbzi_random_table OR cm.

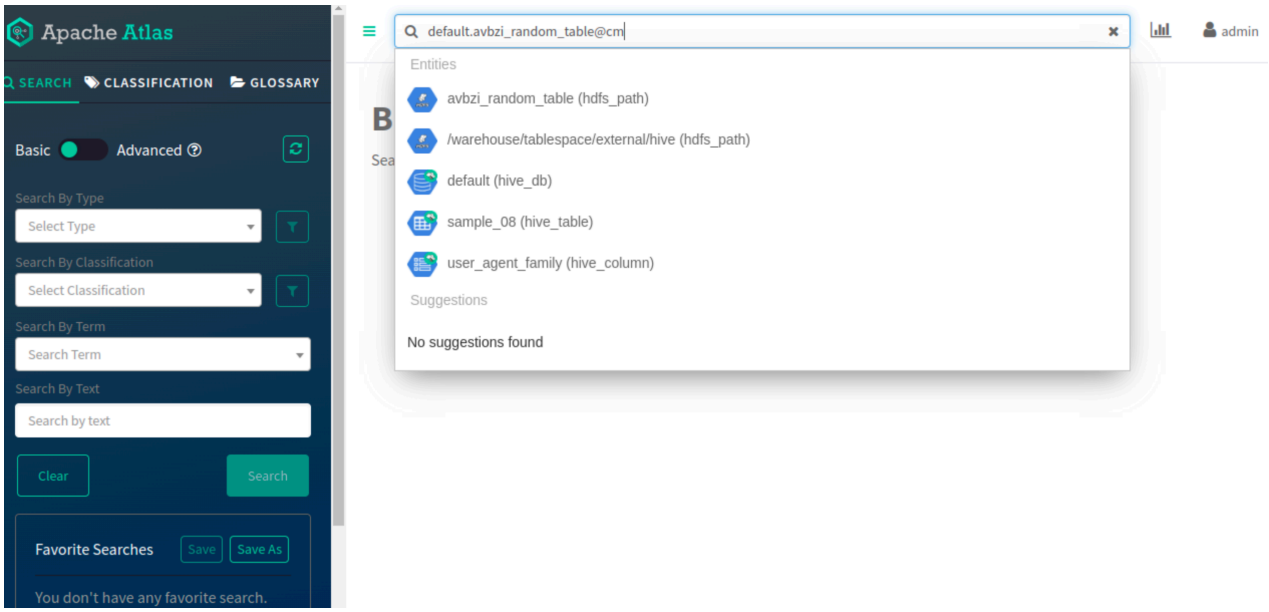

If a string is enclosed with double quotes, SOLR does not tokenize and behave as a single string.

For example: if search string is like "default.avbzi_random_table@cm"

results in, picking up all the matching documents or attributes, where, attribute contains default.avbzi_random_table AND cm.

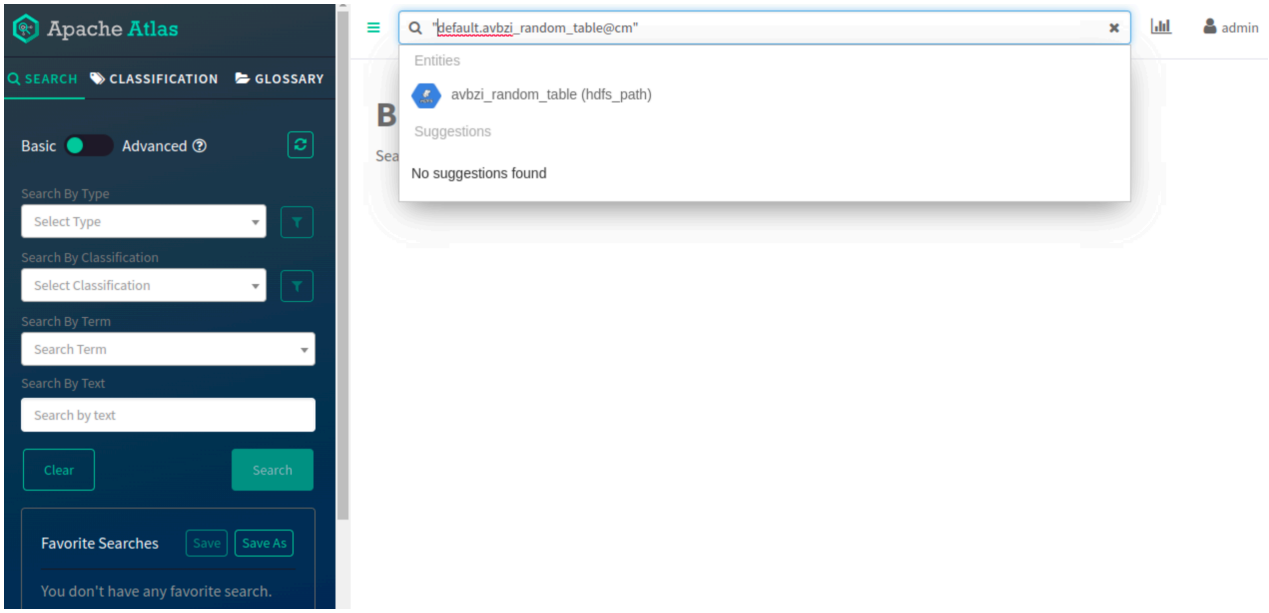

Third use case

In Atlas, name and qualifiedName attributes are different in the way they operate in Atlas Properties. The qualifiedName is a text-based attribute while the name is a string-based attribute. SOLR does not tokenize any string type attribute and does tokenize the text-based attribute.

When you search for 'STRING' type attribute called: 'name', the following conditions must be met:

- Client has to escape 'space'
- For the partial search, client has to append "{*}

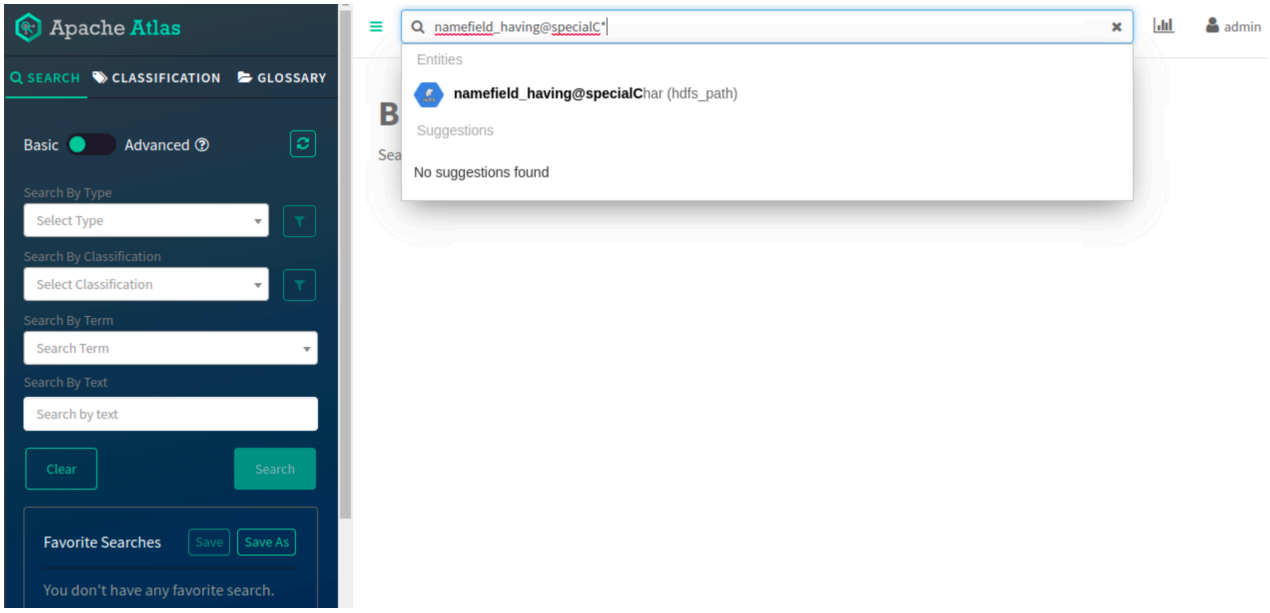

For example: While searching for string with name = finance department@axis, user has to search like finance\ department*

Whenever you search for a 'TEXT' type attribute called: 'qualifiedName', the following conditions must be met:

- No Need of escape 'space'. If it is escaped partial search will not work
- For partial search, no need of appending "{*}

For example: While searching for string with qualifiedName = finance department@axis, user has to search like finance department or finance\ department@axis

Ignore or Prune pattern to filter Hive metadata entities

Atlas supports metadata and lineage updates from services like HBase, Hive, Impala, and Spark.

These updates are in the form of messages that are posted by these services. The messages contain Atlas entities specific to the service. The notification processing module within Atlas processes these messages.

Typically, most of the metadata is tracked. Sometimes, a part of the schema changes more often than not and tracking these frequent changes creates metadata that is insignificant. The Atlas notification processing system gets overloaded with the frequently changing schema updates. The resultant outcome might be that the low-value messages are processed at the expense of messages that contain critical schema updates.

To overcome such a pattern within a data processing pipeline, you can employ a couple of options:

- Ignore schema updates.
- Preserve an abbreviated form of the entity.

The Ignore and Prune feature within Atlas addresses this scenario for Hive Metastore and Hive Server2 (HS2) hooks. This feature is a mechanism to specify which Hive tables should be ignored and which ones should be pruned. This feature helps regulate data that is posted to Atlas. The user is able to choose data that is important for metadata management and lineage capture.

Note: This mechanism does not exist for other hooks.

Attention: The Ignore / Prune configurations feature is not supported when the configurations are provided in upper case or mixed case. You must use the lower case while setting up the Ignore / Prune configurations.

Tables whose lifecycle is of no consequence are targeted for being ignored. Tables whose lifecycle need not be tracked closely or for garnering minute details are targeted for pruning.

Attention: Atlas tracks the table and table-level lineage; however, columns of pruned table and their column level lineage are not tracked in Atlas.

Use case

As a part of the Extract/Transform/Load (ETL) data pipeline, services such as Hive use a number of temporary and/or staging tables that are short-lived. These temporary and/or staging tables are generally employed during the extract or transform phase before the data is loaded. Once the processing is complete, these tables are not used anymore and are deleted.

With Atlas Hive Hook enabled, Atlas captures metadata events, lifecycle, and lineage of all the Hive entities.

Temporary tables that are created only to aid the development process are safe to be ignored. Metadata for these tables are not generated or reported into Atlas.

For staging tables, tracking details like columns and column-lineage in Atlas may not be useful. By not tracking the information in Atlas, it can significantly reduce the time it takes to process notification and can help the overall performance of Atlas.

You can ignore temporary tables completely. Just the minimum details of these staging tables can be stored in Atlas, to capture data lineage from source to target table through all the intermediate staging tables.

Setting Ignore and Prune Properties

The ignore and prune configuration properties can be set both at Atlas server-side and Hive hooks configuration.

Setting it at Hive Hook side prevents Atlas' metadata from being generated.

If the metadata for ignored and pruned elements is generated and posted on Atlas' Kafka topic, setting this property on Atlas' server side handles these elements before they get stored within Atlas.

Both these properties accept Java regex expressions. For more information, see documentation.

How Ignore and Prune feature works

The configurations are matched against the Hive table's qualifiedName attribute.

Within the Hive hook, qualifiedName attribute value has this format: database.table@namespace

The namespace is the value specified by the atlas.metadata.namespace property.

For example, for a Hive hook, the property atlas.metadata.namespace is set to glv.

On that server, for a table t1 which is a part of database db1, the qualifiedName value is: db1.tb1@glv

Ignore Pattern

Hook-side

atlas.hook.hive.hive_table.ignore.pattern

Atlas server side

atlas.notification.consumer.preprocess.hive_table.ignore.pattern

Prune pattern

Hook-side

atlas.hook.hive.hive_table.prune.pattern

Atlas server side

atlas.notification.consumer.preprocess.hive_table.prune.pattern

Using Ignore and Prune patterns

You can configure both Ignore and Prune patterns to manage your data.

Using the Ignore pattern

Atlas ignores temporary managed tables by default. But an external temporary table is captured because the table uses the HDFS path for storage and Atlas creates a lineage in between.

To disregard the temporary table and avoid Atlas processing it, you can set up appropriate configurations in Hive and Atlas and later restart the services.

For example, if all tables in the ‘sales’ database and the tables that contain ‘_tmp’ in the ‘finance’ database should be ignored, the property can be set as follows in your Cloudera Manager instance.

Hive Metastore Server and Hive settings:

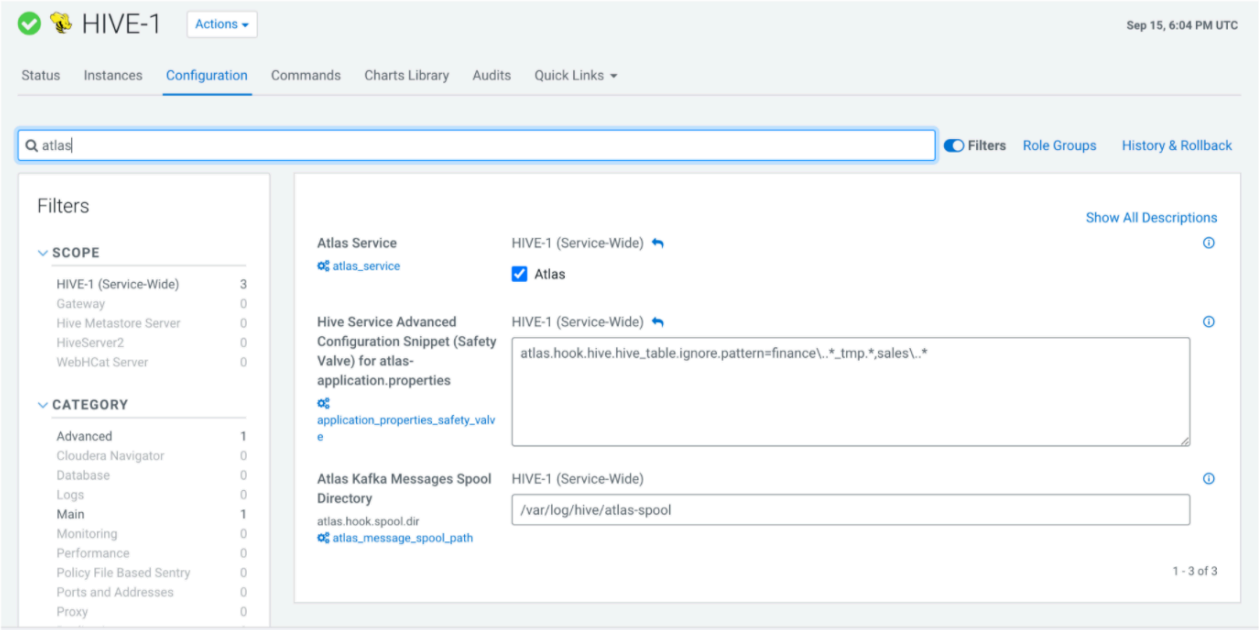

atlas.hook.hive.hive_table.ignore.pattern=finance\..*_tmp.*,sales\..* is set in Cloudera Manager Hive Service Advanced Configuration Snippet (Safety Valve) for atlas-application properties in Hive(HMS) and Hiveserver2.

Atlas server

atlas.notification.consumer.preprocess.hive_table.ignore.pattern=finance\..*_tmp.*

With the above configurations, tables having _tmp in their names, in the finance database are ignored.

Note: The "." is a special regular expression character, hence had to be escaped with a backslash (\).

Using the Prune pattern

Staging tables are created to hold data temporarily during a query execution and are manually dropped once the processing is completed. It might be insignificant to track the details of the staging tables.

For example, in the below images, the finance table contains 333 columns (column names blurred) and the staging tables are created frequently by running an “INSERT OVERWRITE TABLE” query on the finance table. Processing is executed on the data in staging tables and later the staging tables are deleted as observed in the table level lineage diagram.

Table-level Lineage

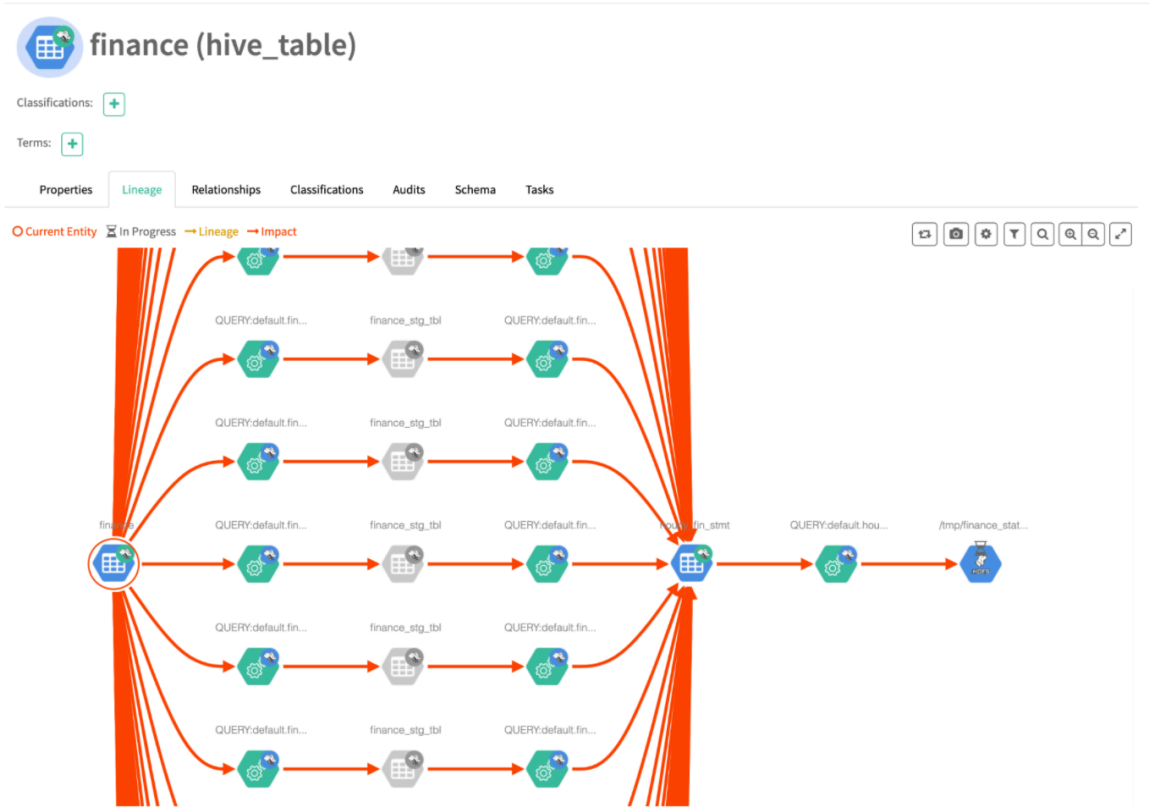

Every time you run a query to create lineage between tables, column-level lineage is also created along with table-level lineage.

Note: If the query involves all the columns, 333 hive_column_lineage entities are created and pushed to the ATLAS_HOOK Kafka topic.

Saving searches

Saving a search saves the search criteria with a name that will help you remember what the search returns.

After you run a search, you can save it under a name in the list of Favorites. Here's what you can do to save a search and to use a search you've already saved:

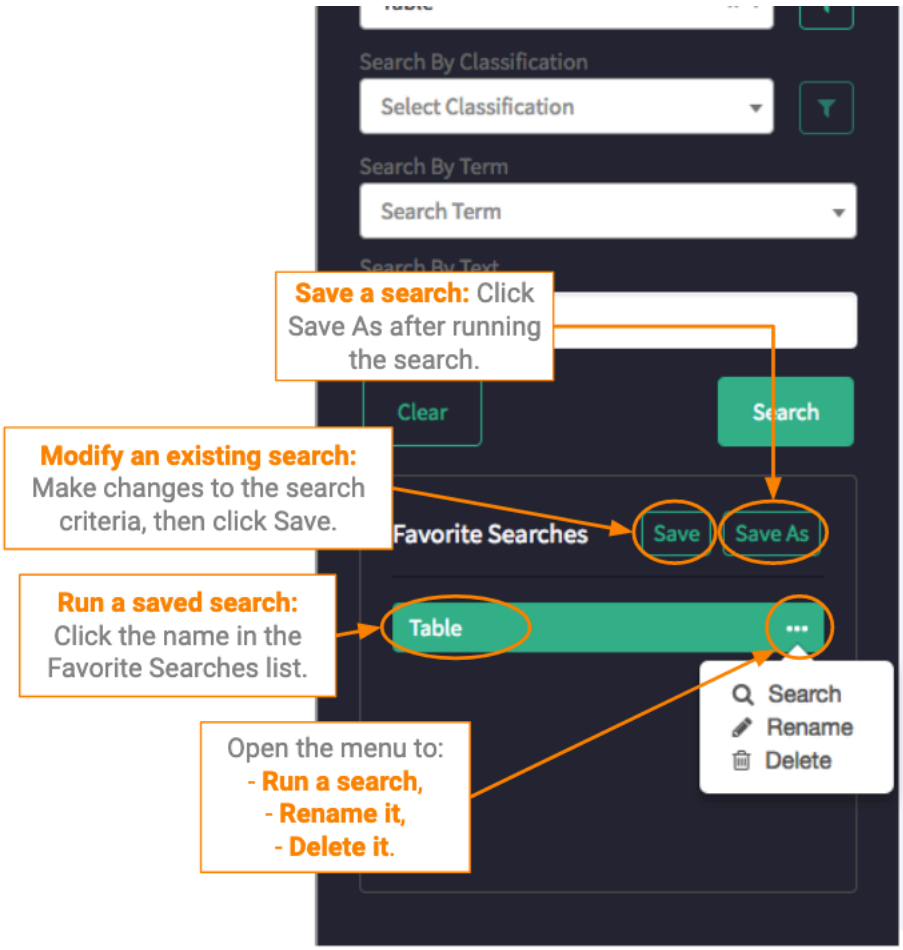

Using advanced search

Apache Atlas advanced search lets you use a query language to combine criteria and refine search results.

Important: In Atlas, the Basic Search feature is the recommended search method because of its more efficient use of resources. Advanced Search is recommended to be used in the following scenarios:

- Querying relationship attributes, for example: hive_table select outputFromProcesses.name
- Using aggregate functions, such as sum, min, max, count, groupby

Advanced search gives you more control over search criteria through the Atlas domain-specific query language.

In the left navigator pane, Search tab, switch to Advanced Search mode by sliding the green toggle button from Basic to Advanced.

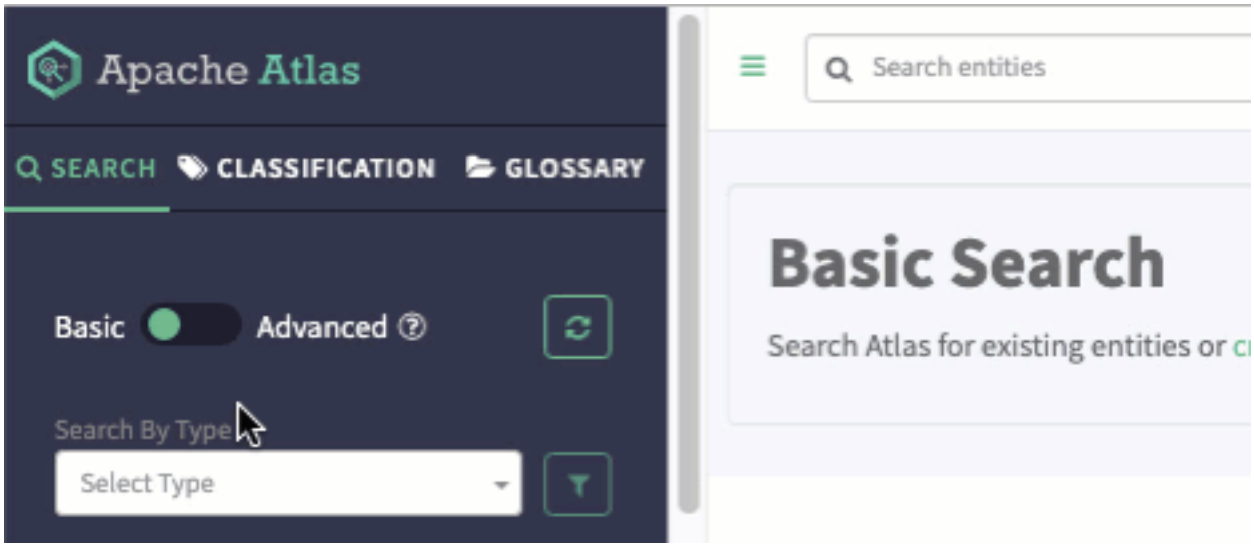

Select an entity type if appropriate, then add your query to refine the search results. Here are some examples of advanced search queries:

- Search for partial names

```
from hive_table where name LIKE '*_dim'
```

- Search in date ranges

  Note that the entity attributes may contain date fields that are populated from the source while the system attributes contain date fields that are populated when the entity is created in Atlas. The values can be different.

Entity attribute createTime	`from hive_table where createTime > '2019-01-01' and createTime < '2019-01-03'`
System attribute Create Timestamp	`from hive_table where __timestamp > '2019-01-01' and __timestamp < '2019-01-03'`

- Search for deleted entitiesSystem attributes (with two underscores before the name) are available on all entity types.

```
from hive_table where __state = DELETED
```

- Search for multiple criteria

```
from hdfs_path where isFile = true and name = "Invoice"
```

- Return specific metadata

```
from hive_table where name = 'customer_dim' select owner, name,
        qualifiedName, createTime
```

- Search for entities with classifications

```
from hive_table where hive_table isa Dimension select owner, name, quali
fiedName
```

See the advanced search reference for information about the query language and for more examples.

Improved search capabilities for Glossary terms and Relationships

In Atlas, while using the Advanced Search feature, you can now search for entities based on the glossary term, by using the newly introduced hasTerm keyword that allows users to search the entities which are tagged with them. You can also search the entities based on relationship attributes using the where clause.

In order to search for those entities having a specific glossary term, you must add a fully qualified name. For example: {termName}@{glossaryName}. This term gets compared with the qualifiedName attribute of glossary type.

Where as, when you add only the term name, the resultant output will be the available entities with the specific term name. This is irrespective of what type of glossary it is in and would compare with the name attribute of the glossary type.

Additionally, to search for entities related to the referenced entities, you must add the relationship attribute and value to search for in the where clause. For example: To search for tables under a specific database. For example: {relatio nshipName}.{attributeName} = {value}

Examples of Glossary term filtering:

- Table hasTerm savingAccount1234
- Table hasTerm “savingAccount1234@Banking"
- Table hasTerm “savingAccount1234@Banking" where Table.name = "customer_dim" and tableType = "externa l"
- Table hasTerm “savingAccount1234@Banking" select name orderby name desc
- Table hasTerm “savingAccount1234@Banking" limit 2
- Table hasTerm “savingAccount1234@Banking" or Table hasTerm "salesTerm@salesGlossary"
- Table hasTerm “savingAccount1234@Banking" and Table isA Dimension
- Table hasTerm “savingAccoun1234t@Banking" and db.name = “Sales" or (Table.qualifiedName like "custom er")
- Table where Table hasTerm “savingAccount1234@Banking"
- Table where (name = "customer_dim" and Table hasTerm “savingAccount1234@Banking")
- Table hasTerm "savingAccount1234@Banking" select count() as terms

Examples of Relationship attributes filtering:

- ```
  Table where db.name = “Sales4321”
  ```
- Table where name = “customer_dim” select columns
- Table where columns.name like “sales” and Table isA Dimension
- Table where db.name = “Sales4321” limit 2
- Table where db.name = “Sales4321” orderby name asc
- Table where db.name = “Sales4321” and columns.name like “sales” and Table hasTerm “salesTerm@salesGl ossary” - (Combination of both where and hasTerm attribute and keyword respectively.)
  ```

Related Information

Atlas Advanced Search language reference

Apache Atlas Advanced Search (atlas.apache.org)

Atlas index repair configuration

You can use reindexing to troubleshoot Apache Atlas basic search inconsistency.

In your Cloudera Manager instance running the Atlas service, add the following in Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties.

atlas.rebuild.index=true

atlas.patch.numWorkers=3

atlas.patch.batchSize=300

Attention:

- You must revert back this configuration once the reindexing is completed, else the reindexing takes place on every restart.
- The reindexing process will be done during Atlas restart, so Atlas will not be reachable till reindexing process is completed.
- The time taken for reindexing depends upon the amount of data.

Later, restart the Atlas Service.

Cloudera Runtime 7.3.1

Securing Apache Atlas

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Securing Atlas 4

Configuring TLS/SSL for Apache Atlas 5

Configuring Atlas Authentication 6

Configure Kerberos authentication for Apache Atlas 7
Configure Atlas authentication for AD 7
Configure Atlas authentication for LDAP 9
Configure Atlas PAM authentication 11
Configure Atlas file-based authentication 12

Configuring Atlas Authorization 15

Restricting classifications based on user permission 16
Configuring Ranger Authorization for Atlas 17
Configuring Atlas Authorization using Ranger 18
Configuring Simple Authorization in Atlas 22

Securing Atlas

Cloudera Manager manages Apache Atlas as a service, automatically ensuring Atlas can communicate securely with its clients and the services it depends on. Use Cloudera Manager to manage additional Atlas settings; use Apache Ranger to control user access in Atlas.

When you include Atlas as a service in a cluster, Cloudera Manager automatically configures the following settings:

- Ranger plugin. Atlas uses Ranger to determine which users have access to perform actions in Atlas.
- TagSync with Ranger. Atlas passes entity metadata for classifications to Ranger using a Kafka topic.
- Access Policies in Ranger. Default policies are configured for the following users:
  - admin: the initial Atlas administrator user has full access to all Atlas actions, including full access to entity metadata, classifications, business metadata attributes, labels and relationship creation, the ability to create new entity, enumeration, structure, and relationship types, the ability to import, export, and purge metadata from Atlas, and the ability to save searches.
  - dpprofiler: the Data Plane service user has the same extensive privileges as the admin user. These privileges allow integration between Cloudera Data Catalog (Data Steward Studio) and Atlas.
  - beacon: the Cloudera Replication Manager service user has the same extensive privileges as the admin user. These privileges allow Atlas to participate in cluster-level disaster recovery operations.
  - rangertagsync: the TagSync service user has read access to entity metadata, specifically to entity classifications, business metadata, and labels to be used in Ranger tag-based policies.
  - rangerlookup: the Ranger lookup service user has read access to entity metadata, specifically to entity classifications, business metadata, and labels to be used in enforcing Ranger policies.
  - public: all users are granted access to read Atlas entity metadata, classifications, labels, and relationships (such as lineage).
  - {USER}: any user who successfully logs in to Atlas can save searches so they are available in subsequent Atlas sessions.

  You will probably want to update and add to these policies to include users and groups in your organization who will need access to Atlas actions.
- TLS-enabled clusters. Cloudera Manager configures:
  - The option to enable TLS for Atlas (atlas.enableTLS)
  - Keystore file locations and passwords for encrypting client-server communication
  - Trust store location and password for the Atlas server to communicate as a client to other services such as Apache HBase and Apache Solr
  - Trust store location and password for the Atlas gateway role that passes information through Kafka topics.

  Note: In Cloudera Public Cloud, communication among services within the cluster is configured to use TLS by default. There is no need to manually configure TLS for Atlas in an SDX cluster.
- Kerberos-enabled clusters. Cloudera Manager configures:
  - Principals for Atlas service users
  - Ranger policies to support authentication for Atlas server and hook communication to Apache Kafka
  - Ranger policies to support authentication for the Atlas server to communicate with Solr and HBase

  Note: In Cloudera Public Cloud, user authentication is managed using Free IPA and Kerberos. There is no need to manually configure authentication settings for Atlas.

Note: Governance through Apache Atlas is just one element of a secure production cluster: Cloudera supports Atlas when it runs on a cluster where enabling Kerberos is optional to authenticate users. Atlas works even without Kerberos enabled.

Setting Atlas session inactivity timeout

For Atlas, you can configure the session inactivity timeout.

In Cloudera Manager for Atlas service, search for the property atlas.session.timeout.secs. By default, the value is -1. You can set the value of this property to 900 (15 seconds).

For example: atlas.session.timeout.secs=900

Related Information

FreeIPA identity management

Configuring TLS/SSL for Apache Atlas

How to configure TLS/SSL for Apache Atlas if you don't choose to use Cloudera Manager's Auto-TLS.

About this task

Typically you would configure TLS for Atlas by using Cloudera Manager's Auto-TLS option. If you need to change these settings or want to understand more about the values set by Cloudera Manager, follow these steps.

Note: In Public Cloud, communication among services within the cluster is configured to use TLS by default. There is no need to manually configure TLS for Atlas in an SDX cluster.

Procedure

1. In Cloudera Manager, select the Atlas service, then click the Configuration tab.
2. Under Category, select Security.
3. Set or update the following properties.

Table 1: Apache Atlas TLS/SSL Settings

Grouping	Configuration Property	Description
Turn on TLS	Enable TLS/SSL for Atlas Server atlas.enableTLS	Select this check box to encrypt Atlas server communication using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)). This option enables TLS for communication between clients and Atlas Server, between the Atlas Server as a client and services it depends on, such as HBase, and between the Atlas Gateway server and Kafka messaging topics. **Attention:** You must set the value of atlas.ssl.exclude.protocols configuration as TLSv1,TLSv1.1
Private Key Management	Atlas Server TLS/SSL Server JKS Keystore File Location keystore.file	The path to the TLS/SSL keystore file containing the server certificate and private key Atlas uses to prove its own identity when it receives communication from other applications using the public key identified in Atlas Server TLS/SSL Client Trust Store File. The keystore must be in JKS format.
	Atlas Server TLS/SSL Server JKS Keystore File Password keystore.password	The password that allows access to the Atlas Server JKS keystore file.

Grouping	Configuration Property	Description
	Atlas Server TLS/SSL Server JKS Keystore File Password password	The password that protects the Atlas Server private key contained in the JKS keystore.
Public Key Management	Atlas Server TLS/SSL Client Trust Store File truststore.file	The path to a TLS/SSL public key trust store file, in .jks format, that contains Atlas server public key. This trust store must contain the certificate(s) used to sign the connected service(s). If this parameter is not provided, the default list of well-known certificate authorities is used instead.
	Atlas Server TLS/SSL Client Trust Store Password truststore.password	(Optional) The password for the Atlas Server TLS/SSL Certificate trust store file. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.
	Gateway TLS/SSL Client Trust Store File atlas.kafka.ssl.truststore.location	The path to the trust store file, in .jks format, used to confirm the authenticity of TLS/SSL servers that the Atlas Gateway might connect to. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead. Typically, this file is the same file listed in the truststore.file.
	Gateway TLS/SSL Client Trust Store Password atlas.kafka.ssl.truststore.password	The password for the Gateway TLS/SSL Certificate trust store file. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.

4. Click Save Changes.
5. Restart the Atlas service.

Configuring Atlas Authentication

This section describes how to configure the authentication methods that determine who is allowed to log in to the Apache Atlas web UI. The authentication options are Kerberos, LDAP—including AD, PAM, or file-based.

Atlas allows more than one authentication method to be enabled at one time. If more than one authentication method is enabled, users failing the first method are authenticated against the second method. The priority order of the methods is Kerberos, LDAP, then file-based authentication. For example if both Kerberos and LDAP authentication are enabled, a request without a Kerberos principal and keytab are authenticated using LDAP.

Specifying more than one authentication method allows you to setup useful production and development scenarios:

- In a Production environment, you might configure Kerberos for service account access to the Atlas server while also supporting LDAP authentication for users logging in through the UI.
- In a Development environment, you might configure Kerberos for service account access while leaving file-based authentication enabled to allow a limited number of administrator to access the Atlas UI.

Note: By default, Cloudera Manager installs Atlas with PAM authentication with full Atlas access given to the username you configure for this authentication method. This configuration ensures a smooth end-to-end installation experience. Be sure to disable PAM authentication when you configure your production authentication method.

Configure Kerberos authentication for Apache Atlas

How to configure Kerberos Authentication for Apache Atlas

Note: In Cloudera Public Cloud, authentication is configured through Free IPA, which uses Kerberos to secure user identities in the environment. There is no need to manually configure authentication settings for Atlas. For more information, see "Managing user access and authorization" in the Cloudera Management Console documentation.

Kerberos authentication for Apache Atlas is automatically configured when you use Cloudera Manager to enable Kerberos authentication for the cluster (typically using the Cloudera Manager Kerberos Wizard).

A prerequisite for using Kerberos authentication is that your cluster is configured for TLS/SSL encryption. There are Atlas-specific steps to complete to enable TLS for Atlas.

Enabling Atlas log in via browser using kerberos keytab

To enable Atlas to log in using the kerberos keytab, you need to set the value of the property atlas.authentication.method.kerberos.support.keytab.browser.login=true in the Safety Valve on Cloudera Manager.

Note: It is not recommended to use this authentication to log in through the browser. Instead use the Knox trusted proxy to access Atlas UI with Kerberos identity.

Related Information

Configuring TLS/SSL for Apache Atlas

FreeIPA identity management

Configure Atlas authentication for AD

How to configure Apache Atlas to use AD for user authentication when using AD cluster-wide.

About this task

Note: In Cloudera Public Cloud, identity management is provided by FreeIPA and configured using the Cloudera Management Console. There is no need to manually configure authentication settings for Atlas. For more information on FreeIPA, see the Cloudera Management Console documentation.

The settings indicated in these steps apply to Atlas authentication and it is likely that the values will be the same as you use to configure other services on the cluster.

Procedure

1. In Cloudera Manager, select the Atlas service, then open the Configuration tab.
2. To display the authentication settings, type "authentication" in the Search box. You may need to scroll down to see all of the AD settings.
3. Configure the following settings for AD authentication:

Property	Description	Sample values
Enable LDAP Authentication atlas.authentication.method.ldap	Determines whether LDAP is used for authentication.	true
LDAP Authentication Type atlas.authentication.method.ldap.type	The LDAP type (ldap, ad, or none).	ad

Property	Description	Sample values
AD URL atlas.authentication.method.ldap.ad.url	The AD server URL.	
AD Bind DN Username atlas.authentication.method.ldap.ad.bind.dn	Full distinguished name (DN), including common name (CN), of an AD user account that has privileges to search.	cn=admin,dc=example,dc=com
AD Bind DN Password atlas.authentication.method.ldap.ad.bind.password	Password for the account that can search in AD.	Secret123!
AD Domain Name (Only for AD) atlas.authentication.method.ldap.ad.domain	AD domain, only used if Authentication method is AD.	example.com
AD User Search Filter atlas.authentication.method.ldap.ad.user.searchfilter	The AD user search filter.	(&(sAMAccountName={0})(objectClass=user)(memberOf=cn=PTTOR-Contract-Big data Admin,OU=OR Data Platform,OU=PTTOR Distribution Manual Group,DC=pttor,DC=corp)) **Note:** The {0} place holder is required in configuration to substitute to fill in with a valid user name to build the search filter.
AD Base DN atlas.authentication.method.ldap.ad.base.dn	The Distinguished Name (DN) of the starting point for directory server searches.	dc=example,dc=com
AD User Default Role atlas.authentication.method.ldap.ad.default.role	AD User default Role.	
AD Referral atlas.authentication.method.ldap.ad.referral*	See below. Defaults to ignore.	follow

* There are three possible values for atlas.authentication.method.ldap.ad.referral: follow, throw, and ignore. The recommended setting is follow.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to follow, the AD service provider processes all of the normal entries first, and then follows the continuation references.
- When this property is set to throw, all of the normal entries are returned in the enumeration first, before the ReferralException is thrown. By contrast, a "referral" error response is processed immediately when this property is set to follow or throw.
- When this property is set to ignore, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search. In the case of AD, a PartialResultException is returned when referrals are encountered while search results are processed.

4. Click Save Changes.
5. Restart the Atlas service.

Related Information

Cloudera Management Console onPublic Cloud: Managing user access and authorization

Cloudera Management Console onPrivate Cloud: Managing user access and authorization

Configure Atlas authentication for LDAP

How to configure Apache Atlas to use LDAP for user authentication.

About this task

Note: In Cloudera Public Cloud, identity management is provided by FreeIPA and configured using the Cloudera Management Console. There is no need to manually configure authentication settings for Atlas. For more information on FreeIPA, see the Cloudera Management Console documentation.

The settings indicated in these steps apply to Atlas authentication and it is likely that the values will be the same as you use to configure other services on the cluster.

Procedure

1. In Cloudera Manager, select the Atlas service, then open the Configuration tab.
2. To display the authentication settings, type "authentication" in the Search box. You may need to scroll down to see all of the LDAP settings.
3. **Note:** Hadoop groups were not included during user/group lookup, because Atlas queries the UserGroupInformation for that specific user at the Operating System level (org.apache.hadoop.security.ShellBasedUnixGroupsMapping package). At the time of logging into the Atlas UI, it was seen that "no such user as Gen.tleman" exception was thrown. When user selects the LDAP configuration fromCloudera Manager, they must configure (org.apache.hadoop.security.LdapGroupsMapping package) in core-site.xml file and also add atlas.authentication.ugi-groups.include-hadoop-groups=true property.

   Later, the user can sync the Hadoop group and create entity and classification.

   Include the property atlas.authentication.ugi-groups.include-hadoop-groups=true in Atlas safety valve (atlas-application.properties file), as it takes groups from Hadoop LDAP-based group mapping.

   Configure the following settings for LDAP authentication:

Grouping	Property	Description	Sample values
Enable LDAP Authentication	Enable LDAP Authentication atlas.authentication.method.ldap	Determines whether LDAP is used for authentication.	true
	LDAP Authentication Type atlas.authentication.method.ldap.type	The LDAP type (ldap, ad, or none).	ldap
LDAP Server Location	LDAP Server URL atlas.authentication.method.ldap.url	The LDAP server URL.	ldap://localhost:389 or ldaps://localhost:636
Bind Credentials	LDAP Bind Username atlas.authentication.method.ldap.bind.dn	Full distinguished name (DN), including common name (CN), of an LDAP user account that has privileges to query the LDAP database of user accounts on behalf of Atlas. This could be a read-only LDAP user.	cn=admin,ou=people,dc=example,dc=com
	LDAP Bind DN Password atlas.authentication.method.ldap.bind.password	Password for the account that can search for users.	Secret123!

Grouping	Property	Description	Sample values
Group Lookup	LDAP Group-Search Base atlas.authentication.method.ldap.groupSearchBase	The organizational unit (OU) and domain component (DC) properties for the LDAP search tree where Atlas searches for groups.	(|(CN=Hdp_users)(CN=Hdp_admins))
	LDAP Group-Search Filter atlas.authentication.method.ldap.groupSearchFilter	(Optional) Refine the scope of LDAP group search. The Groups-Search Filter is combined with the Group-Search Base to define the group lookup.	
Role Assignment	LDAP Group-Role Attribute atlas.authentication.method.ldap.groupRoleAttribute	The attribute stored in the LDAP Group object to use to map LDAP groups to Atlas roles.	cn
	LDAP User Default Role atlas.authentication.method.ldap.default.role	Atlas role to assign to LDAP users.	
LDAP Search-Bind Authentication Mode	LDAP DN atlas.authentication.method.ldap.base.dn	The Distinguished Name (DN) of the starting point of the LDAP search tree for directory server searches. You can also specify a User Search Filter to further reduce the scope of the search.	dc=example,dc=com
	LDAP User Search Filter atlas.authentication.method.ldap.user.searchfilter	The LDAP user search filter. Used with the User Search Base to further limit the scope of the search for a directory entry that matches the credentials of the user logging into Atlas. Use a user search filter along with a DN pattern so that the search filter provides a fallback if the DN pattern search fails.	
LDAP Direct-Bind Authentication Mode	LDAP User DN Pattern atlas.authentication.method.ldap.userDNpattern	Direct-bind authentication can be used if search is not required to determine the DN needed to bind to the LDAP server. Leave this property blank if LDAP DN is set. To use this authentication mode, all users must be under a single branch in the LDAP directory.	To search for a distinguished name where the uid attribute is the username at login, you might provide a pattern such as: uid={0},ou=users,dc=xasecure,dc=net where {0} indicates the username of the authenticating user. If a user provides the username "foo" at the login page, Atlas searches for the DN: uid=foo,ou=People,dc=corp,dc=com

Grouping	Property	Description	Sample values
LDAP Referral	LDAP Referral atlas.authentication.method.ldap.referral*	See below. Defaults to ignore.	follow

* There are three possible values for atlas.authentication.method.ldap.referral: follow, throw, and ignore. The recommended setting is follow.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to follow, the LDAP service provider processes all of the normal entries first, and then follows the continuation references.
- When this property is set to throw, all of the normal entries are returned in the enumeration first, before the ReferralException is thrown. By contrast, a "referral" error response is processed immediately when this property is set to follow or throw.
- When this property is set to ignore, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search. In the case of LDAP, a PartialResultException is returned when referrals are encountered while search results are processed.

4. Click Save Changes.
5. Restart the Atlas service.

Related Information

Cloudera Management Console on public cloud: Managing user access and authorization
Cloudera Management Console on Private Cloud: Managing user access and authorization

Configure Atlas PAM authentication

By default in Data Center installations, Apache Atlas uses PAM authentication, which means valid Atlas users correspond to the users configured for the operating system on the host where Atlas runs. Cloud installations do not use PAM authentication.

About this task

private cloud These steps describe setting properties to enable PAM authentication. To disable PAM, you can remove the properties or set the PAM authentication method property to false. If you leave the PAM authentication method to true and set another authentication method to true, Atlas uses both methods to authenticate users.

Note: In Cloudera Public Cloud, authentication is configured through Free IPA, which uses Kerberos to secure user identities in the environment. There is no need to manually configure authentication settings for Atlas. For more information, see "Managing user access and authorization" in the Cloudera Management Console documentation.

Before you begin

Minimum Required Role in Cloudera Manager: Full Administrator.

Procedure

1. In Cloudera Manager, select the Atlas service, then open the Configuration tab.
2. To display the appropriate property, type "safety" in the Search box.
3. Find the Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties.

4. In the safety valve, set the following properties:

```
atlas.authentication.method.pam=true
atlas.authentication.method.pam.service=LOGIN SERVICE
```

   where *LOGIN SERVICE* indicates the desired PAM login service. For example, set atlas.authentication.method.pam.service=login to use /etc/pam.d/login.
5. Click Save Changes.
6. Restart the Atlas service.

   Once the PAM authentication is enabled for Atlas on a Private Cloud Base cluster, to access Atlas, note the following:

   - If Atlas was configured using LDAP or Active Directory, the same user role which was set-up can be authenticated and used for accessing Atlas.
   - For an Operating System user, you can add the user using useradd or adduser commands and later access Atlas.

   Note: If Ranger is enabled, you must have appropriate permissions for the user (both the scenarios above) in Ranger policies. Alternatively, if Ranger is not enabled, the user must manually update the JSON file - atlas-simple-authz-policy.json

Related Information

Cloudera Management Console on Public Cloud: Managing user access and authorization

Cloudera Management Console on Private Cloud: Managing user access and authorization

Configure Atlas file-based authentication

How to manage Apache Atlas user authentication when using user credentials from a file.

Note: In Cloudera Cloud, identity management is provided by FreeIPA and configured using the Cloudera Management Console. Therefore for Cloudera Public Cloud, you should leave the Admin Authentication Method set to Kerberos authentication settings. For more information, see "Managing user access and authorization" in the Cloudera Management Console documentation.

Warning: Atlas file-based authentication is intended as a convenience for managing authentication in a development environment. Do not use file-based authentication in a production environment.

Two of the most likely tasks you would perform related to file-based authentication for Atlas:

- Enable or disable file-based authorization when transitioning among authentication methods
- Add to the list of authorized users for use in a non-production environment

To enable or disable file-based authentication

1. In Cloudera Manager, select the Atlas service, then open the Configuration tab.
2. Display the authentication settings by typing "file" in the Search box.

3. Check or uncheck the option Enable File Authentication.

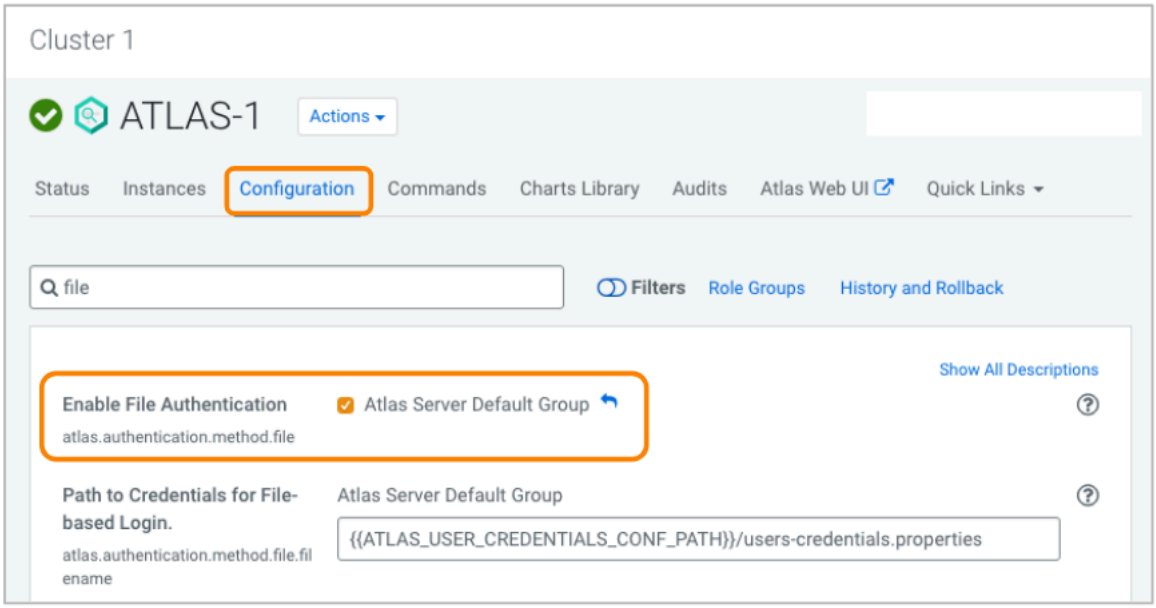

4. If you are enabling file-based authentication, review the location of the file that contains user credentials as specified in Path to Credentials for File-based Login.

   The default directory location indicated by ATLAS_USER_CREDENTIALS_CONF_PATH is the Atlas configuration directory in the Cloudera Runtime installation location, typically

   ```
   /opt/cloudera/parcels/CDH-VERSION/etc/atlas/conf.dist
   ```

   If you need to change the location, specify an absolute path on the Atlas host. Alternatively you can reset the location for users-credentials.properties by setting the atlas.authentication.method.file.filename property in the Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties. You can find the hostname where the Atlas service is running on the Instances tab in Cloudera Manager.
5. Configure the list of users as described in step 6 on page 14 in To add to the users list: on page 13.
6. Click Save Changes.
7. Restart the Atlas service.

To add to the users list:

This procedure requires that you restart the Atlas service.

1. In Cloudera Manager, select the Atlas service, then open the Instances tab.

2. Note the Hostname where the Atlas service is running. You'll need to be able to SSH into that host to update the user list.

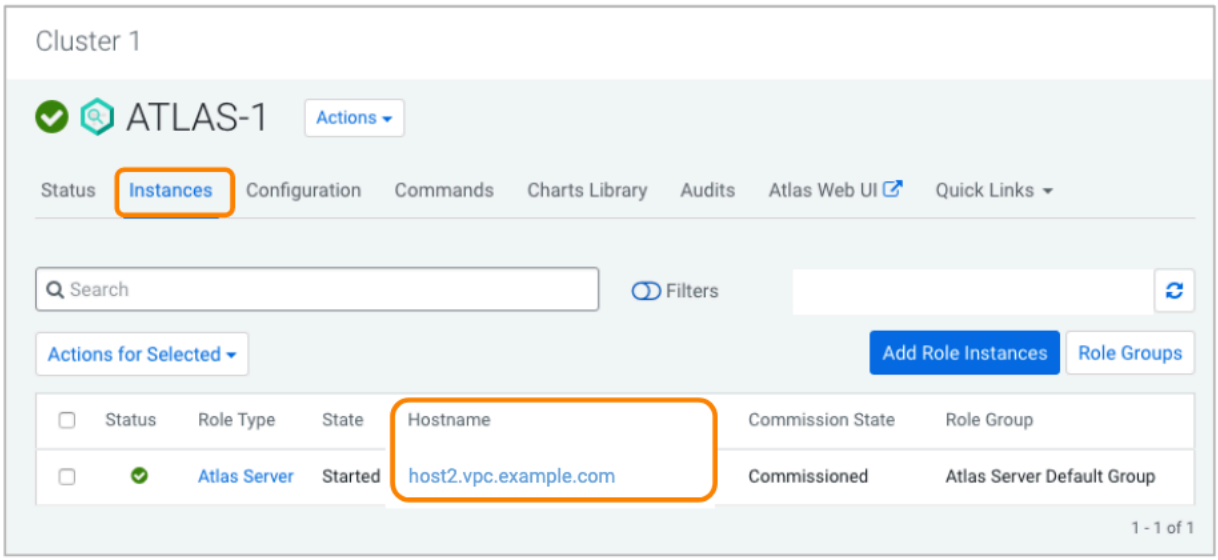

3. Open the Configuration tab.
4. Type "file" in the search box to filter the property list.
5. Find the user file.

   Look for the location of the users-credentials.properties file as set in the Path to Credentials for File-based Login property.

   The default directory location indicated by ATLAS_USER_CREDENTIALS_CONF_PATH is the Atlas configuration directory in the Cloudera Runtime installation location, typically

   ```
   /opt/cloudera/parcels/CDH-VERSION/etc/atlas/conf.dist
   ```

   If you need to reset the location of the users-credentials.properties file, see Moving the Atlas user credentials file on page 15.
6. Update the user list.

   SSH into the Atlas host. Edit the users file to include additional users in the following format:

   ```
   USERNAME=GROUP::PASSWORD-HASH
   ```

   where the *USERNAME* is the string used in the login page, the *GROUP* is one of ADMIN, DATA_STEWARD, or DATA_SCIENTIST. The *PASSWORD-HASH* is the password encoded using salting.

   To generate an encoded password, use the python cputil.py script provided in the Atlas installation, typically:

   ```
   /opt/cloudera/parcels/CDH-version/lib/atlas/bin
   ```

   Run the command as follows, making sure that variables used in the script are available:

   ```
   $ export ATLAS_HOME=/opt/cloudera/parcels/CDH-VERSION/lib/atlas
   $ export ATLAS_CONF=<Atlas installation location>
   $ /opt/cloudera/parcels/CDH-version/lib/atlas/bin/cputil.py -g -u <user
   name> -p <new secure password> -s
   ```

   The -s option ensures that the output includes only the hash value. The Atlas installation location is typically /opt/cloudera/parcels/CDH-*VERSION*/etc/atlas/.
7. Restart the Atlas service.

Moving the Atlas user credentials file

If you find you need to point Atlas to a different location for the users-credentials.properties file, you can reset the location using a property in an Atlas advanced configuration snippet.

1. In Cloudera Manager, select the Atlas service, then open the Configuration tab.
2. Display the advanced configuration settings by typing "safety" in the Search box.
3. In the Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties property, add the following property and set it to the absolute path to the users-credentials.properties file on the Atlas host. This path must be accessible to the Atlas service user (defaults to atlas).

```
atlas.authentication.method.file.filename=/some/secure/location
```

4. Restart the Atlas service.

Related Information

Cloudera Management Console on Public Cloud: Managing user access and authorization

Cloudera Management Console on Private Cloud: Managing user access and authorization

Configuring Atlas Authorization

Apache Atlas in Cloudera uses Apache Ranger policies to control access to metadata that are managed by Atlas. Ranger policies also control access to Atlas administrative tasks.

Ranger provides authorization to access the following metadata and operations:

Types

Atlas "types" are the entity model definitions, whether provided in Atlas or added in your environment. Types include these "categories":

- Entity
- Classification
- Relationship
- Business Metadata
- Struct
- Enum

Ranger authorization allows you to configure access for users and groups to perform the following operations on types:

- Create
- Update
- Delete
- Read

The policies can be configured to apply to one or more types or all types. For example, the Atlas administrator user has access to create, update, and delete all type categories (type-category *).

Entities

Atlas "entities" are instances of entity types: entities represent assets and processes on your cluster. Ranger authorization allows you to configure access to users and groups to perform the following operations on entities:

- Read
- Create
- Update
- Delete

- Read classification
- Add classification
- Update classification
- Remove classification
- Add label
- Remove label
- Update Business Metadata

Note that the classification operations are those that involve associating a classification to an entity; operations on a classification definition are controlled by authorization on the classification category of type described previously. Use the entity authorization to give a user the ability to associate an existing classification with any entity (entity-type *); use the type authorization to give a user the ability to create new classifications (type-category classification).

Policies for labels and business metadata work similarly to classifications: you can control whether users can add labels or business metadata to specific entity types, individual entities, or entities marked with specific classifications. For example, a default policy allows any authenticated user to update all business metadata for any entity types with any classifications and on any instances of entities (entity-type *, entity-classification *, entity-id *, entity-business-metadata *).

Some Atlas features, such as saved searches, are modeled as entities. You can control access to these features using entity policies. For example, a default policy allows any authenticated user to save Atlas searches (entity-type __AtlasUserProfile, __AtlasUserSavedSearch).

Relationships

Atlas "relationships" describe connections between two entities, including, but not limited to, the input and output relationships that are used to build lineage graphs. Ranger authorization allows you to configure access to users and groups to perform the following operations on relationships:

- Add relationship
- Update relationship
- Remove relationship

These operations are required to build rich models among entities and are granted to administrative users and system users. Relationships cannot be updated by users through the Atlas UI.

Admin operations

Atlas administrative operations include:

- Import entities
- Export entities

These operations encompass all the privileges needed to create new and update existing entities. Typically, this access is granted to administrative users and system users such as RangerLookup and the Data Plane profiler user (DPProfiler).

Restricting classifications based on user permission

The Apache Atlas authorisation model has been updated with a new permission, type-read, to enable restricting visibility of types to specific users, groups, and roles.

Altas authorization model supports permissions to create, update, delete types, which enables setting up policies to restrict who can perform create, update, delete on types (like classifications, business metadata). Note that all authenticated users were allowed to perform read operation on types.

In the updated model, users with type-create or type-update or type-delete permission will implicitly be granted type-read permission. Also, to be compliant with earlier user experience, a default policy is created to allow type-read operation to all authenticated users. Security administrators can update the default policy and add new policies to restrict type-read operation to specific users.

Prior to this update, new permission, the type-read access was not available and read access for typedefs was public. Now the objective for Read Type permission is to have the ability to set access control permission on visibility of classifications and other typedefs (entitydef, enumdef, structdef, and business_metadatadef) in Atlas.

Previously, policy permission types only included Create Type, Update Type, and Delete Type.

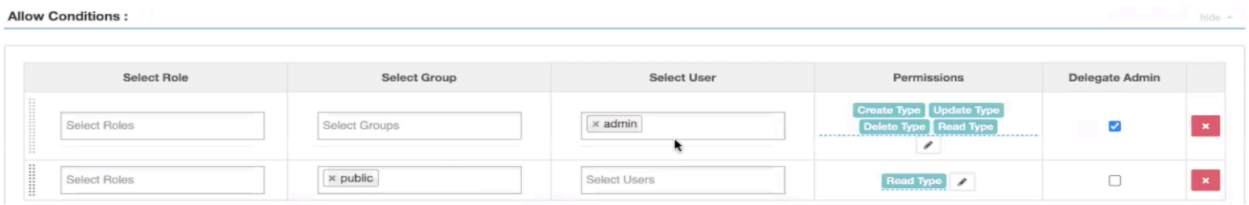

Note: With the introduction of “Read Type” permission, when you select any one of the types - Create Type or Update Type or Delete Type, the Read Type permission is implicitly enabled.

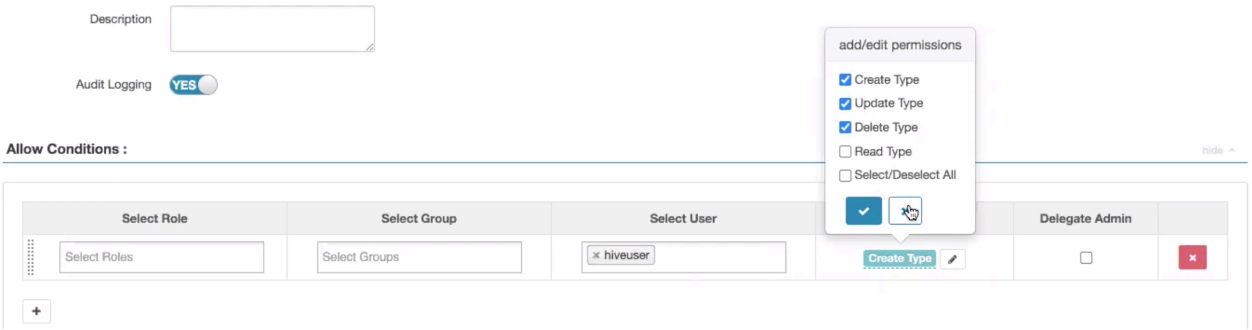

Note: Upgrade scenario: The new Read Type permission is available as part of the upgrade scenario.

For example: If you are upgrading from Cloudera Private Cloud Base 7.1.x to 7.1.4, you must be able to make use of the new read type permission.

Configuring Ranger Authorization for Atlas

Apache Atlas is configured to use Apache Ranger for authorization by default. You might need to change configuration settings to disable Ranger as the source of authorization in a development environment; Ranger authorization is highly recommended in a production environment. In addition, there are some configuration values that you might need to change should you make significant changes to how Atlas and Ranger are installed in your cluster.

About this task

Atlas behaves like any other service when it comes to integrating with Ranger for access control: turn on Ranger authorization from the Cloudera Manager configuration page for the Atlas service. This integration allows Atlas to use Ranger policies to determine authorization for user actions in Atlas; Atlas also reports success or failure against the policies back to Ranger. In addition to this standard integration for authorization, Atlas integrates with Ranger to send metadata updates to Ranger using an Apache Kafka topic. The configuration properties that support the two integration paths include specifying HDFS locations for caching Ranger policies, storing Atlas audits, and storing other metadata exchanged between the two services. In rare circumstances, you may need to relocate these storage locations.

Important: Native Hadoop Library must be added to the ATLAS classpath for Ranger group authorization to work. If this is missing from the Atlas class path, for users affected who need Hadoop Group based authorization to work, the below details must be added to the Environment Safety Valve for Atlas Server:

Key : ATLAS_CUSTOM_OPTS

Value : -Djava.library.path=${CDH_HADOOP_HOME}/lib/native

Before you begin

Minimum Required Role in Cloudera Manager: Full Administrator.

Procedure

1. In Cloudera Manager, select the Atlas service, then open the **Configuration** tab.
2. To display the Ranger configuration settings, type "ranger" in the Search box.
3. To modify the behavior of the Atlas to Ranger integration, update the following properties:

   To disable Atlas authentication through Ranger

   Uncheck the RANGER Service property. Leave the supporting properties as is so you can re-instate Ranger easily if needed.

   To set where Atlas stores intermediate data for Ranger audits and cached authorization policies

   Review and potentially change the path value for these properties:

   - Ranger Atlas Plugin Hdfs Audit Directory (HDFS location)
   - Ranger Atlas Plugin Audit Hdfs Spool Directory Path (local file system location)
   - Ranger Atlas Plugin Audit Solr Spool Directory Path (local file system location)
   - Ranger Atlas Plugin Policy Cache Directory Path (local file system location)

   To re-instate Ranger after disabling it

   Enable the RANGER Service property. The supporting properties should still be available.
4. Restart the Atlas service.

Configuring Atlas Authorization using Ranger

Use the Ranger Admin Web UI to add or update policies to control Apache Atlas access.

About this task

You can use Apache Ranger policies to control user-access to Atlas metadata and to actions that users can perform in Atlas.

Attention: To manage the Atlas Web UI authorisation using Ranger, **LDAP UGI Groups** (atlas.authentication.method.ldap.ugi-groups) property must be enabled.

The following policies are defined by default:

- admin: the initial Atlas administrator user has full access to all Atlas actions, including full access to entity metadata, classifications, business metadata attributes, labels and relationship creation, the ability to create new entity, enumeration, structure, and relationship types, the ability to import, export, and purge metadata from Atlas, and the ability to save searches.
- dpprofiler: the Data Plane service user has the same extensive privileges as the admin user. These privileges allow integration between Cloudera Data Catalog (Data Steward Studio) and Atlas.
- beacon: the Cloudera Replication Manager service user has the same extensive privileges as the admin user. These privileges allow Atlas to participate in cluster-level disaster recovery operations.
- rangertagsync: the TagSync service user has read access to entity metadata, specifically to entity classifications, business metadata, and labels to be used in Ranger tag-based policies.
- rangerlookup: the Ranger lookup service user has read access to entity metadata, specifically to entity classifications, business metadata, and labels to be used in enforcing Ranger policies.
- public: all users are granted access to read Atlas entity metadata, classifications, labels, and relationships (such as lineage).
- {USER}: any user who successfully logs in to Atlas can save searches so they are available in subsequent Atlas sessions.

Before you begin

To change Ranger policies for Atlas, your user needs privileges in Ranger to change Resource Based Policies.

Procedure

1. Open the Ranger service that is running in the same cluster as Atlas.
   One way to do this is to open the Ranger Admin Web UI from Cloudera Manager.

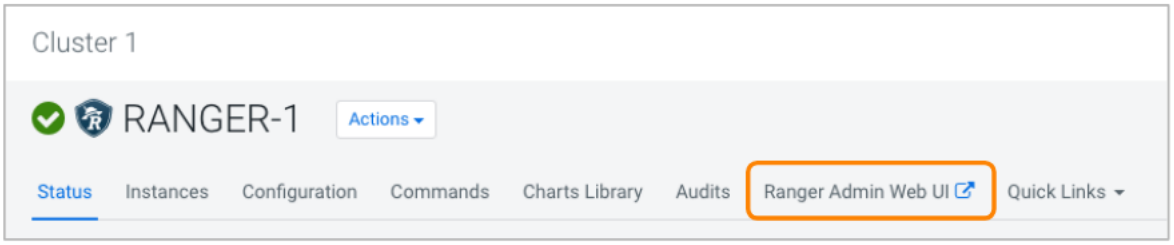

2. Open Access Manager Resource Based Policies and select Atlas policies (cm_atlas).

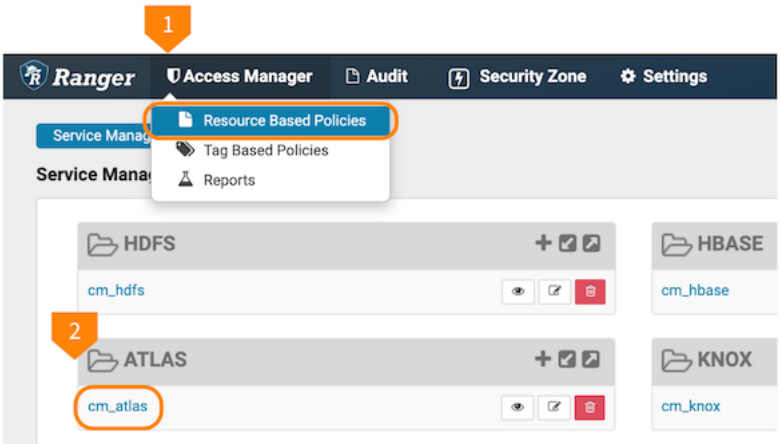

3. On the List of Policies page, click Add New Policy.

4. Use the Create Policy page to specify the Atlas authorization policy.

   Ranger fields support ? and * wildcards for single and multiple character replacement. To apply the policy to all types of a given entry, use *.

   These selections can be set to "include" the selected resources or "exclude" the selected resources. An include policy applies only to the named types. An exclude policy for the same type would apply to all metadata types other than named types.

   Ranger Authorization with classifications

   Currently in Atlas we have the following authorization options for using classification:

   - Authorization on types: Where user can create a new classification def (not the association)

     For example: Only admin_user is allowed to create tags - PII, PHI..etc.
   - Authorization on Entity: For entities with typeName hive_table with qualifiedName (abc@cl1) and having classification (already associated) can add, update, and remove classifications.

     For example, if any Hive table with qualifiedName 'emp@cl1 and having the tag PII, the admin_user can add the SENSITIVE tag.

   Authorization Enhancement

   The enhancement is provided to authorize as to who can add, remove, and update classification for an entity. Even if the entities on which classification have to be applied, that do not have classifications already tagged to it, provided the entity-type, Entity-ID and classification on it matches the specified policy.

   For example, any users belonging to the user group - finance_group can add classification - FINANCE_PII to any entities (these entities may or may not have any classifications associated) if the policy has provided access to the user or user-group.

   To achieve this, a new field named classification is introduced in the Ranger UI in which tags to be added can be specified.

   Entity Classification denotes the classifications already associated with the entity.

   For example:

   Entity Classification - Finance_*

   Classification: Finance_PII

   Access: Group: finance_group, Permission: Add/Update/Delete Classifications.

Policy Type	Access. There are no other policy types available for an Atlas service.
Policy Name	255 character name that appears in the list of policies. Roles, users, and groups also show up in the list, so it helps if your name includes the operations or metadata that the policy controls.
Policy Label	Metadata you can include in the policy definition to help organize the policies for a given service. The same label can be added to any number of policies for the service. There is no limit to the number of characters in a label, but only 28 characters display in the policy list.
type-category menu	The metadata or operation type ("resources" in Ranger terms) that the policy applies to, including: • type-category • entity-type • atlas-service • relationship-type
type-category option	Choose this option to authorize actions generally against Atlas resource types, including business metadata, classifications, enumerations, entities, relationships, structures. With type-category selected, options include:

	Type Name	Refine the authorization to specific types within the named type category. For example, to give users authorization to create Atlas Business Metadata, choose type-category and the category Business Metadata; then set the Type Name to *. For example, to authorize users to add values to an existing enum, such as AtlasGlossaryTermRelationshipStatus, add this enum to the Type Name and include the permission for "Update Type" in the Allow Condition. To allow users to update any types within the type category, use *. To determine the supported values, use the Atlas UI or API to show the defined types.
entity-type option		Authorizes actions against specific entity types, individual entities, entities identified by associated classifications, or entities identified by associated metadata. For example, to authorize users to add classifications or metadata to any Hive table entities, set the entity-type to hive_table and set additional options to *. With entity-type selected, options include:
	Entity Classification	Refines the list of entities in entity-type to those associated with a specified classification. For example, to restrict authorization to Hive tables that were marked with some classification that indicates their readiness for use, set entity-type to hive_table and include the identifying classification name (e.g., Avai lable) in Entity Classification.
	Entity ID	Refines the list of entities in entity-type to those associated with a specified ID. When the detail page for an entity is open in the Atlas UI, the last element of the browser URL indicates the entity ID.
	classification	Provides the option to authorize as to who can add, remove, and update classification for an entity, even if the entities on which classification have to be applied, which do not have classifications already tagged to it, provided the entity-type, Entity-ID and classification on it matches the specified policy.
	Metadata types selection	Refines the list of entities in entity-type to those associated with specific user-defined metadata, including: • entity-label • entity-business-metadata • classification • none Set label names in the type entity-label to limit the authorization policy to entities marked with any of those labels. Use * to indicate any label. Set business metadata collection names in the type entity-business-metadata to limit the authorization policy to entities marked with metadata attributes from that business metadata collection. Use * to indicate any business metadata collections.
atlas-service option		Authorizes the import and export Atlas entities and purge deleted entities through the API. This privilege overrides specific privileges for entity-types. Typically the users with this privilege are service users creating entities in Atlas.
relationship-type option		Authorizes the creation and update of Atlas relationships. You can identify specific relationship types or use * to indicate any relationship type. Typically the users with this privilege are service users creating entities in Atlas.
	End1 Entity Type End1 Entity Classification End1 Entity ID End2 Entity Type End2 Entity Classification End2 Entity ID	Refines the relationship authorization to specific attributes of relationships. "End1" and "End2" indicate the entities on each side of the relationship. For example, you could use the End1 and End2 Entity Type options to allow modification of relationships when one side of the relationship are Hive tables and the other side Hive columns.
Description		Information that you add to help you remember the value of this policy. The description can be up to 1000 characters.
Audit Logging		Enables Ranger's audit logging for this policy. There are other options in Ranger's configuration that can conflict with this option, but generally if you turn off this setting, Ranger enforces the policy but does not audit success or failed actions against the policy.

Allow Conditions	Choose the roles, users, and/or groups and the permissions they can access for the resources defined in the policy. If you need to include parts of overlapping groups, add an exclude condition in addition to the allow condition. For more information, see Ranger access conditions.
Deny Conditions	Choose the roles, users, and/or groups and the permissions they cannot access for the resources defined in the policy.

5. Click Add.

Results

You should be able to validate the policy almost immediately after saving a valid policy.

Related Information

Configure a resource-based service: Atlas

Add or edit permissions in Ranger

Configuring Simple Authorization in Atlas

Apache Atlas in Cloudera uses the simple authorization, which is the default authorizer to manage different users, user groups, and group roles.

This type of authorization does not involve creating policies in Ranger UI. This is NOT a recommended authorization method to grant permissions in Atlas but can still be used to manage user authorization. The simple authorizer employs policies that are defined in a JSON file.

The following properties must be configured in Atlas properties. When you uncheck Ranger authorizer, these properties are automatically added by the Cloudera Manager.

- atlas.authorizer.impl=simple
- atlas.authorizer.simple.authz.policy.file=atlas-simple-authz-policy.json

Simple Authorization provides certain roles for managing Atlas metadata and related operations. Under each role, you could define the permission for each resource, based on the entity type for which you want to grant access.

There are three supported roles:

- ROLE_ADMIN - Has all permissions to perform any action.
- DATA_SCIENTIST - Has permission for reading and classifying entities.
- DATA_STEWARD - Has permission to read, classify, create, update, add, and remove classifications.

An example JSON file that displays the above scenarios:

```
{

 "roles": {
   "ROLE_ADMIN": {
     "adminPermissions": [
       {
         "privileges": [ ".*" ]
       }
     ],

     "typePermissions": [

       {
         "privileges":     [ ".*" ],
         "typeCategories": [ ".*" ],
         "typeNames":      [ ".*" ]
       }
     ],

     "entityPermissions": [
```

```
        {

           "privileges":      [ ".*" ],
           "entityTypes":     [ ".*" ],
           "entityIds":       [ ".*" ],
           "classifications": [ ".*" ],
           "labels" :         [ ".*" ],
           "businessMetadata" : [ ".*" ],
           "attributes" :     [ ".*" ]

        }
      ],

      "relationshipPermissions": [

        {
          "privileges":     [ ".*" ],
          "relationshipTypes": [ ".*" ],
          "end1EntityType":           [ ".*" ],
          "end1EntityId":             [ ".*" ],
          "end1EntityClassification": [ ".*" ],
          "end2EntityType":           [ ".*" ],
          "end2EntityId":             [ ".*" ],
          "end2EntityClassification": [ ".*" ]

        }
      ]
    },

    "DATA_SCIENTIST": {

      "entityPermissions": [

        {
          "privileges":      [ "entity-read", "entity-read-classification" ],
          "entityTypes":     [ ".*" ],
          "entityIds":       [ ".*" ],
          "classifications": [ ".*" ]
        }
      ]
    },

    "DATA_STEWARD": {

      "entityPermissions": [
        {
          "privileges":      [ "entity-read", "entity-create", "entity-upda
te", "entity-read-classification", "entity-add-classification", "entity-upda
te-classification", "entity-remove-classification" ],
          "entityTypes":     [ ".*" ],
          "entityIds":       [ ".*" ],
          "classifications": [ ".*" ]=
        }
      ],

      "relationshipPermissions": [
        {
          "privileges":               [ "add-relationship", "update-relatio
nship", "remove-relationship" ],
          "relationshipTypes":        [ ".*" ],
          "end1EntityType":           [ ".*" ],
          "end1EntityId":             [ ".*" ],
          "end1EntityClassification": [ ".*" ],
          "end2EntityType":           [ ".*" ],
```

```
          "end2EntityId":              [ ".*" ],
          "end2EntityClassification": [ ".*" ]

        }
      ]
    }
  },
```

The simple authorization roles that are defined as explained earlier can be assigned to users:

```
"userRoles": {
"admin":[ "ROLE_ADMIN" ],
"rangertagsync":[ "DATA_SCIENTIST" ]
```

Note: The rangertagsync user role is pre-defined in the simple authorizer JSON.

Roles can be granted to user-groups. An user can belong to multiple groups; roles assigned to all groups the user belongs to will be used to authorize the access.

```
"groupRoles": {

   "ROLE_ADMIN":      [ "ROLE_ADMIN"],
   "hadoop":          [ "DATA_STEWARD" ],
   "DATA_STEWARD":    [ "DATA_STEWARD" ],
   "RANGER_TAG_SYNC": [ "DATA_SCIENTIST" ]
 }

}
```

Note: The RANGER_TAG_SYNC group role is pre-defined in the simple authorizer JSON.

Cloudera Runtime 7.3.1

Troubleshooting Apache Atlas

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Atlas index repair configuration.....4

Atlas index repair configuration

You can use reindexing to troubleshoot Apache Atlas basic search inconsistency.

In your Cloudera Manager instance running the Atlas service, add the following in Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties.

atlas.rebuild.index=true

atlas.patch.numWorkers=3

atlas.patch.batchSize=300

Attention:

- You must revert back this configuration once the reindexing is completed, else the reindexing takes place on every restart.
- The reindexing process will be done during Atlas restart, so Atlas will not be reachable till reindexing process is completed.
- The time taken for reindexing depends upon the amount of data.

Later, restart the Atlas Service.

Cloudera Runtime 7.3.1

Atlas Classifications

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Working with Atlas classifications and labels **4**
Text-editor for Atlas parameters 5

Creating classifications **7**
Example for finding parent object for assigned classification or term 8

Creating labels **13**

Adding attributes to classifications **14**

Support for validating the AttributeName in parent and child TypeDef **16**
Validations for parent types 16
Case for implementing backward compatibility 17

Associating classifications with entities **18**

Propagating classifications through lineage **19**

Searching for entities using classifications **21**

Working with Atlas classifications and labels

Add metadata to Atlas entities using labels and classifications.

You can add metadata to Atlas entities to help your organization find, organize, and share your understanding of the data assets that drive business processes. Atlas provides two mechanisms for adding metadata to entities: labels and classifications. Both labels and classifications can be applied to entities to help describe the entity's content, status, or other business-driven value.

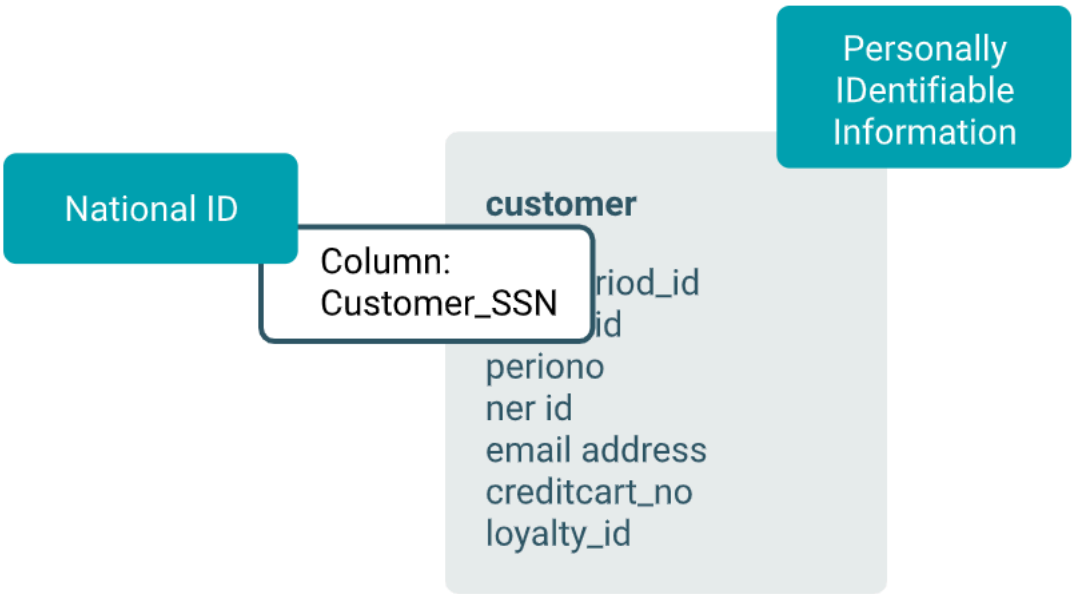

Labels

Labels are words or phrases (strings) that you can associate with an entity and reuse for other entities. They are a light-weight way to add information to an entity so you can find it easily and share your knowledge about the entity with others.

Anyone can create labels and associate labels with entities.

Classifications

Classifications are strings like labels, with added complexity and structure:

- Atlas includes precise search tools for finding entities using classifications.
- Classifications can automatically propagate to additional entities through lineage relationships.
- You can use classifications to drive access control policies in Ranger.
- You can enrich a classification with attributes in the form of key-value pairs and set the value to describe a particular entity.

Administrators can control who can create and assign classifications through Atlas policies in Ranger.

In short, use labels for annotating entities; use classifications to involve entities in processes inside and outside Atlas.

More about classifications: attributes

You can add key-value pairs, or *attributes* to the definition of a classification. A typical use for classification attributes would be to refine the meaning of a general category. Data assets identified with a classification of “PII” or Personally Identifiable Information can have classification attributes that indicate the nature of the information to drive data masking or expiration policies. Columns tagged with “PII” might be further separated into phone numbers, credit card numbers, and “other.” A Ranger policy based on the classification can use the attribute values to identify masks for the phone and credit card numbers and to block columns tagged as “PII” with attribute “Other.”

Planning for classifications and labels

Here are some questions to help you think about how you define classifications and labels in your system:

- Will you use the metadata to drive a workflow outside of Atlas? If so, use a classification. If not, a label may work fine.
- Do you need the text of the metadata to be exact values? You can limit who has the ability to create classifications: you can manage a single list for an organization. The ability to apply a classification to an entity can be controlled separately from defining classifications, which let's you potentially allow more users to applying the "official" classifications. Any users can create and add labels to entities, so it may be more difficult to standardize the content of labels.
- What types of entities do you expect the metadata to apply to? Labels can be added to any entity type; classifications can be defined to apply to a specific entity type. If you define a classification to only to apply to certain entity types, such as table columns, make sure that the name and description helps data stewards use the classification correctly.
- Are you adding metadata to make Atlas searches easier? Consider using classifications with attributes to allow refinement of search results. If your Atlas users are more likely to search using *terms*, you might consider connecting the classification to a glossary term so searches from either mechanism return the correct results. (Assigning the term to an entity automatically assigns the classification to the same entity.) For both labels and classifications, consider setting guidelines so everyone creating metadata uses the same conventions: underscores or hyphens? detail first or last? table-level metadata or column-level?
- Do you want a classification to follow lineage relationships and be assigned automatically as data is used in other entities? If not, you can use the classification description to help data stewards understand that they should turn off propagation when assigning this classification to entities or terms. Labels do not propagate.

Text-editor for Atlas parameters

In Atlas, while creating Classification, Glossary, and Business metadata, a new text editor is available, which can also be used for adding and customizing string attribute values of Business metadata on the Entity details page.

On the Atlas entities details page, when you add the business metadata and select the attribute type as “String”, you can view a text editor with which you can customize content / description that is entered.

Important: While adding a Business metadata attribute, you MUST select the “Type” as “String”. If you do not fulfill this criteria, you cannot add formatting to the value of the attribute using the text-editor on the Entity details page under the Business metadata tab.

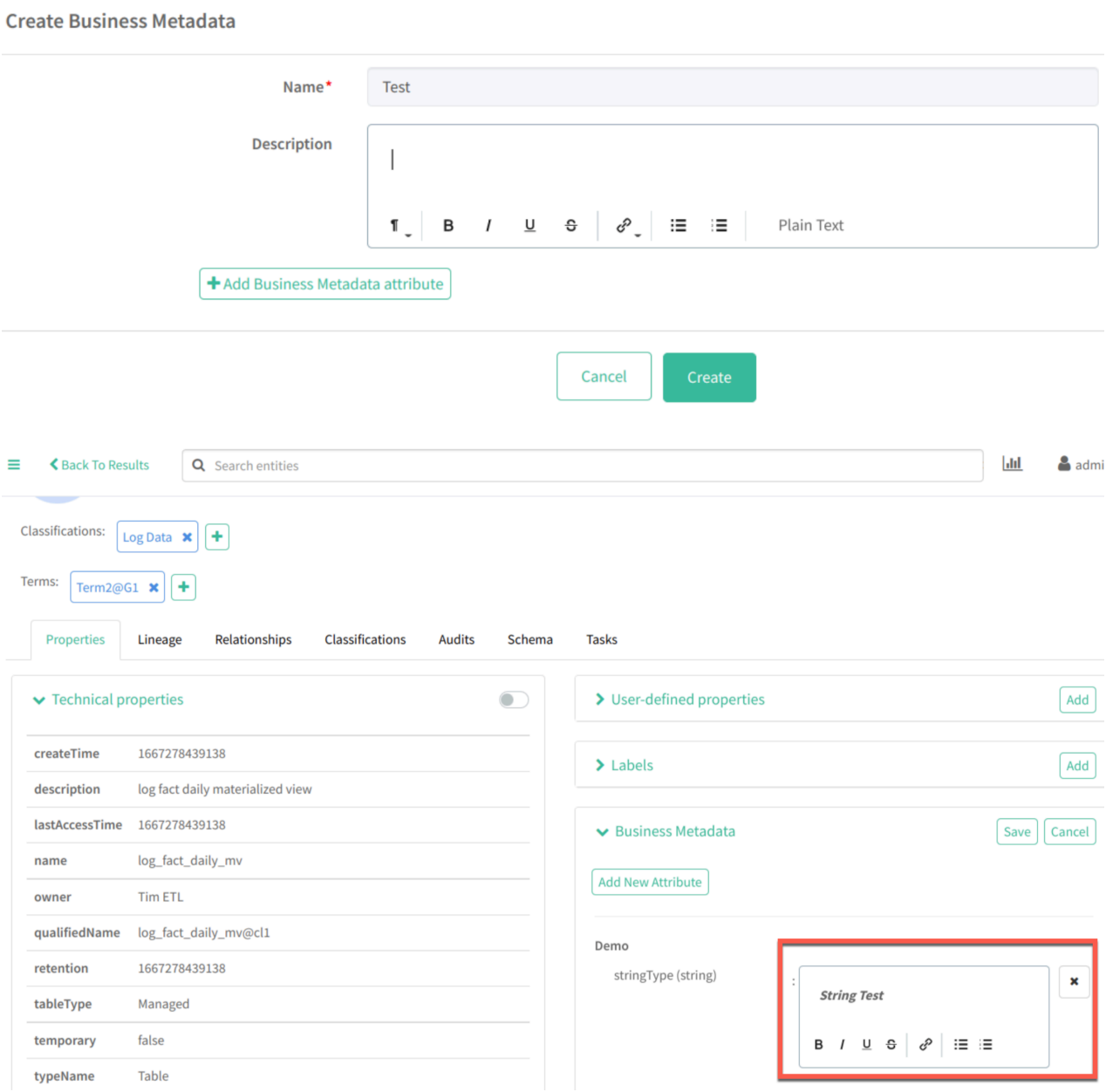

Note that while creating and editing Glossary, Term, and Category, the text editor is available only for the Long Description field. For creating and editing Business Metadata and Classification, text editor is available only for the Description field.

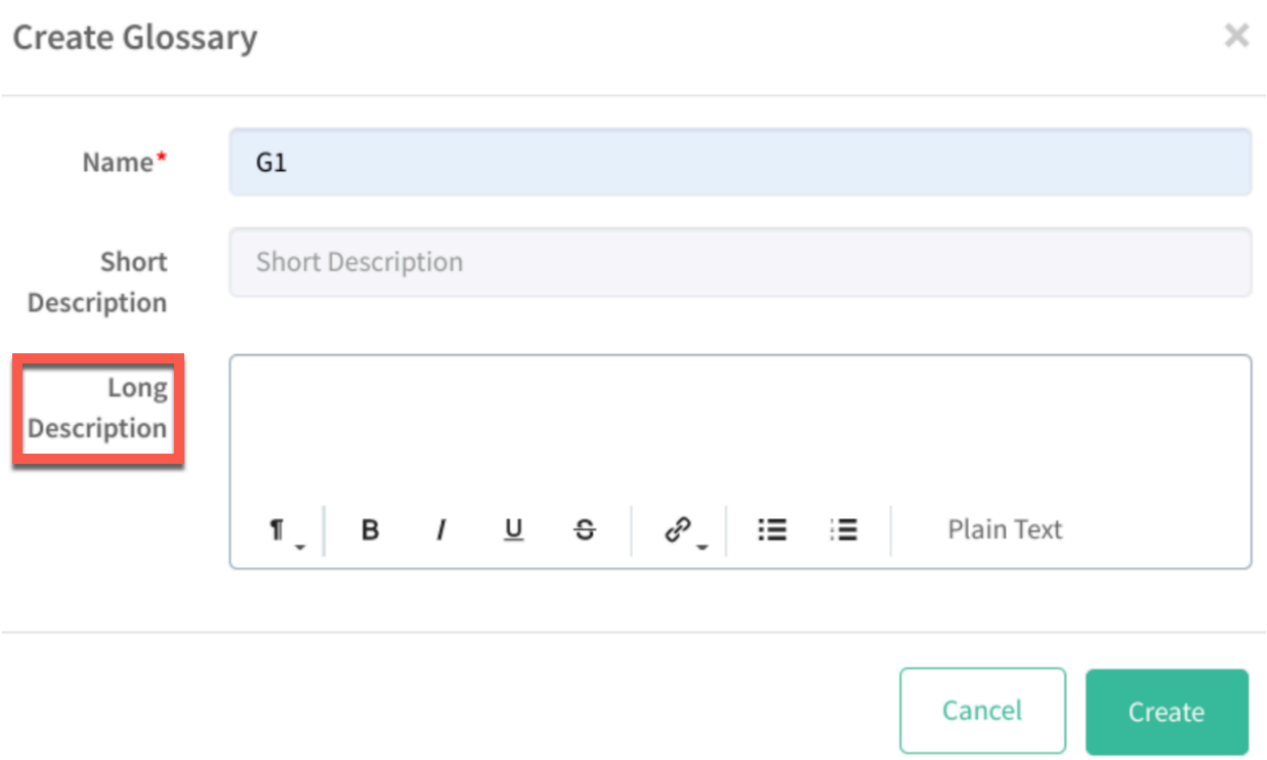

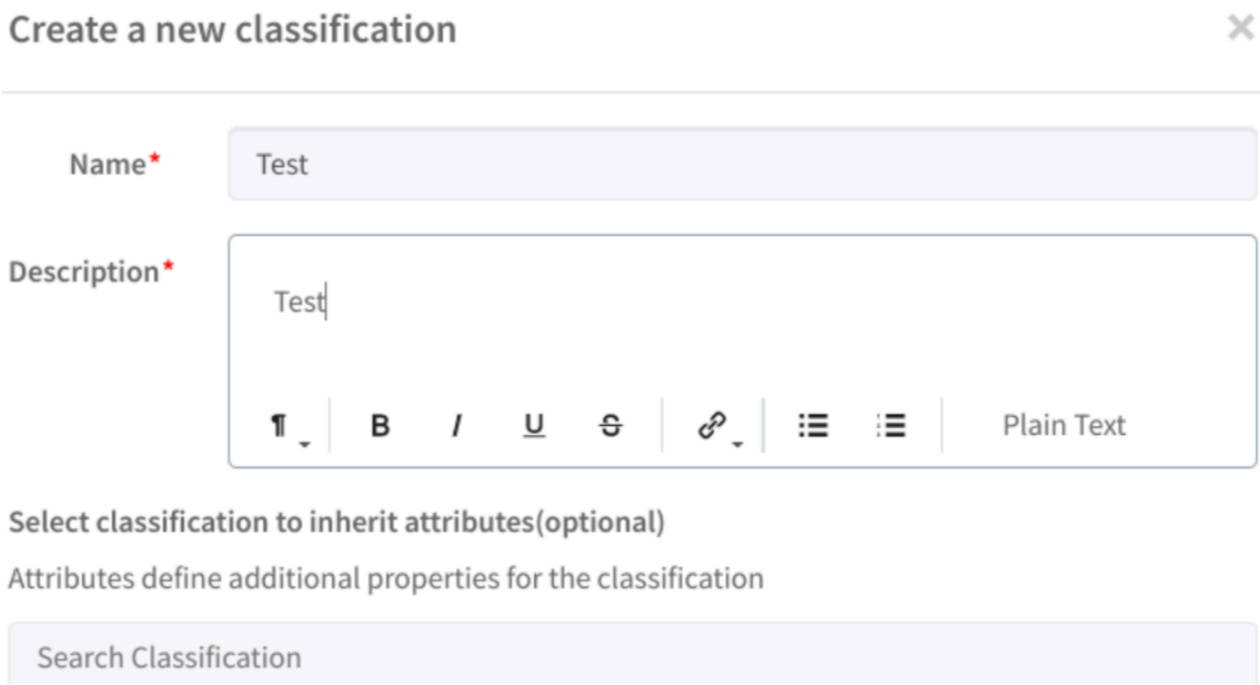

Creating classifications

Create classifications from the tab in the left navigation bar.

Create classifications from the Classifications tab in the left navigation pane.

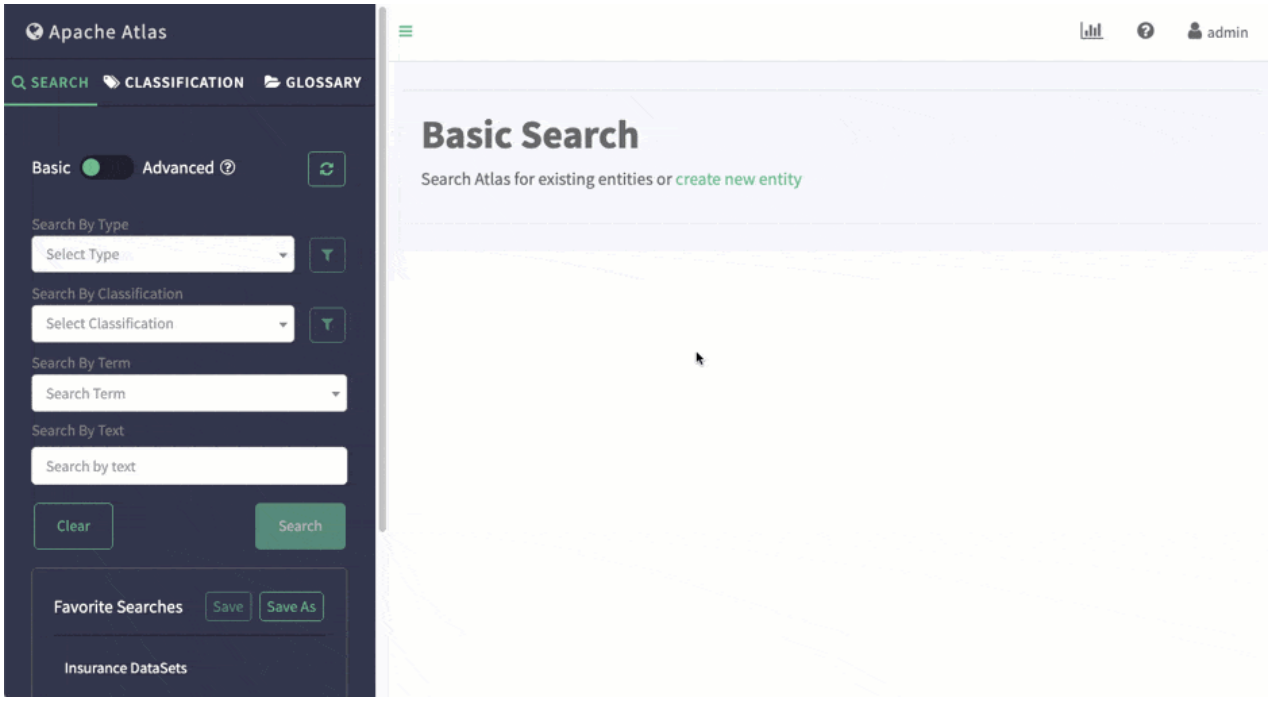

Names can be 255 characters and contain ASCII characters and spaces. Descriptions can be 1024 characters and contain any characters.

When creating a classification, you can choose an existing classification to reuse its attribute definitions.

Example for finding parent object for assigned classification or term

You can view and search for Atlas entities by using the assigned name for the classification / term.

When you create an entity and assign a classification or term to the entity, the immediate parent of the assigned classification / term is displayed when you hover over the classification/term. The parent-child relationship lineage appears in the “assigned @parent1, parent2, parent3” format, depending upon the number of parent-child relationships. Click the assigned classification and term to view the details.

As an example, assume that there is an entity called hdfs_path for which a classification named “Classification1_1” is assigned. Under the “Classification” tab on the Atlas UI, you can search for the list of classifications > child classifications > child classifications > and so on.

The following image provides an example of how a classification is listed in Atlas.

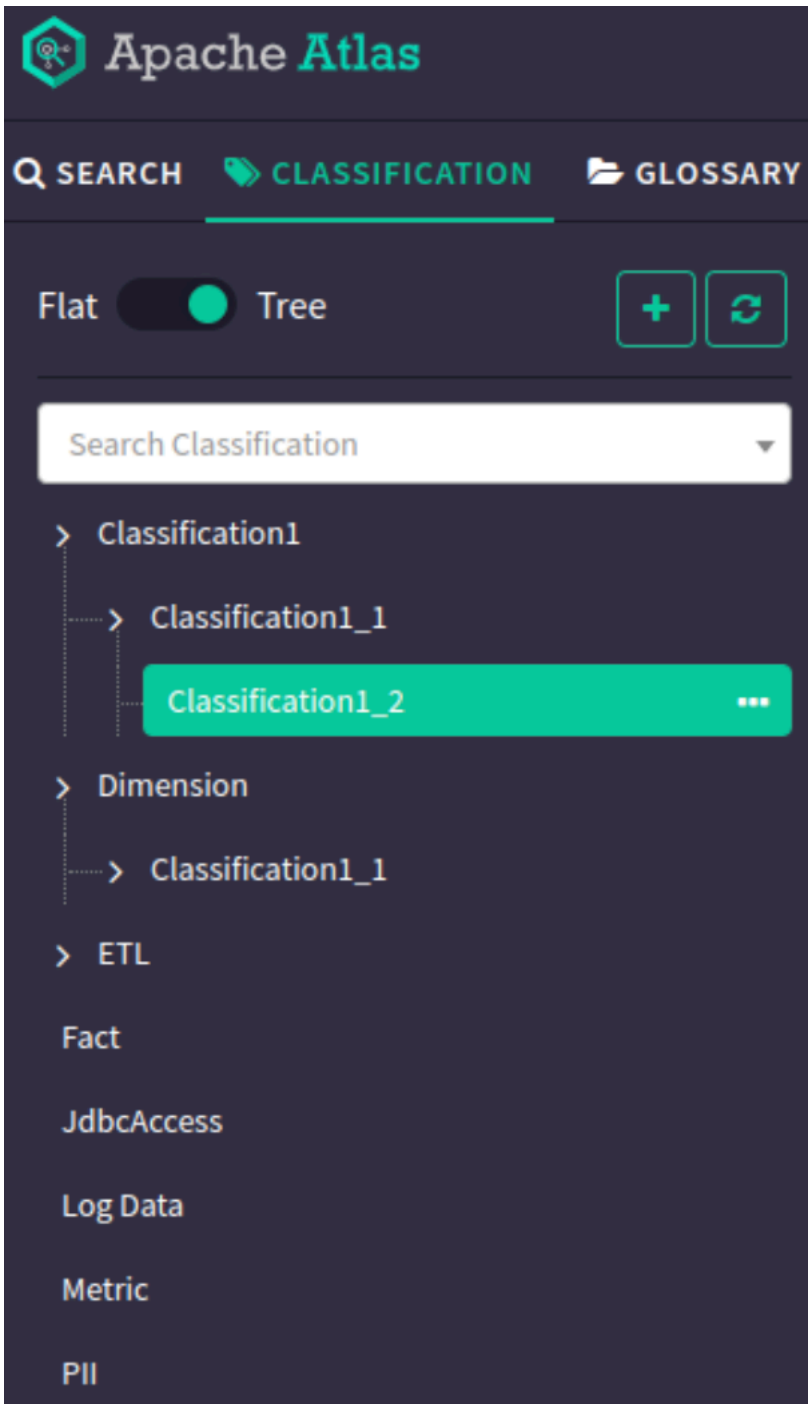

The classification named Classification1_1 has parent classifications named Classification1 and Dimension respectively.

In the following image, the parent classifications are displayed on an entity for which Classification1_1 is assigned.

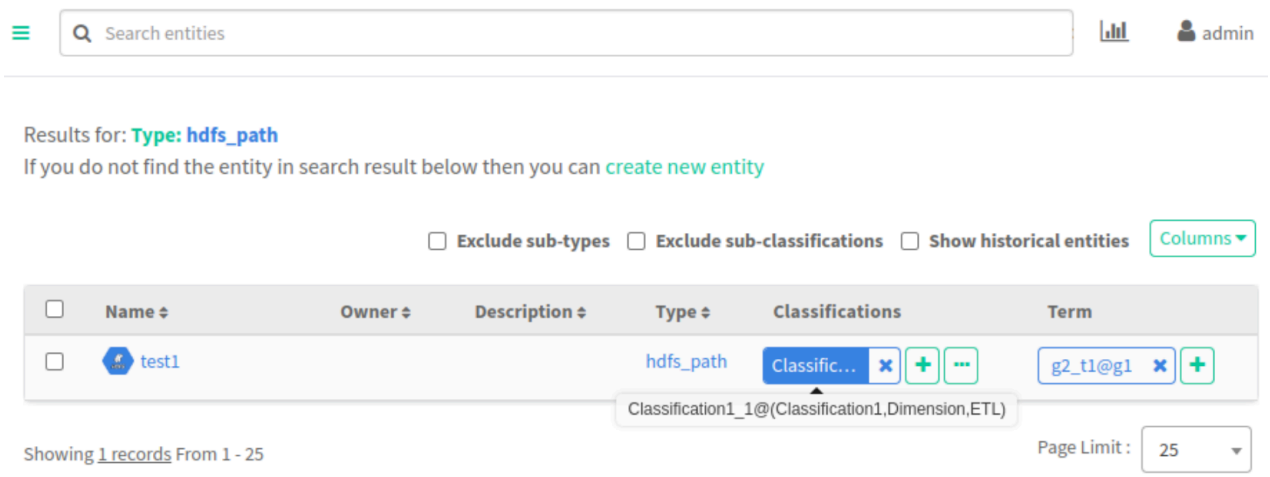

Note that hovering over the assigned classification displays the parent-child relationship. In this example: Classification1_1@(Classification1,Dimension,ETL)

Clicking on the assigned classification navigates to the classification details page.

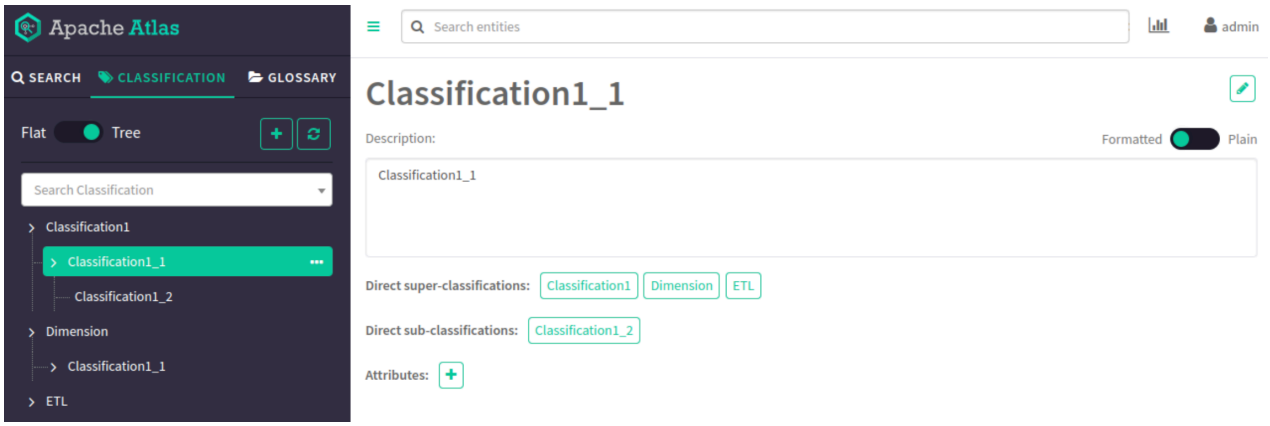

The following image provides an example of how Glossary > Terms are listed in Atlas.

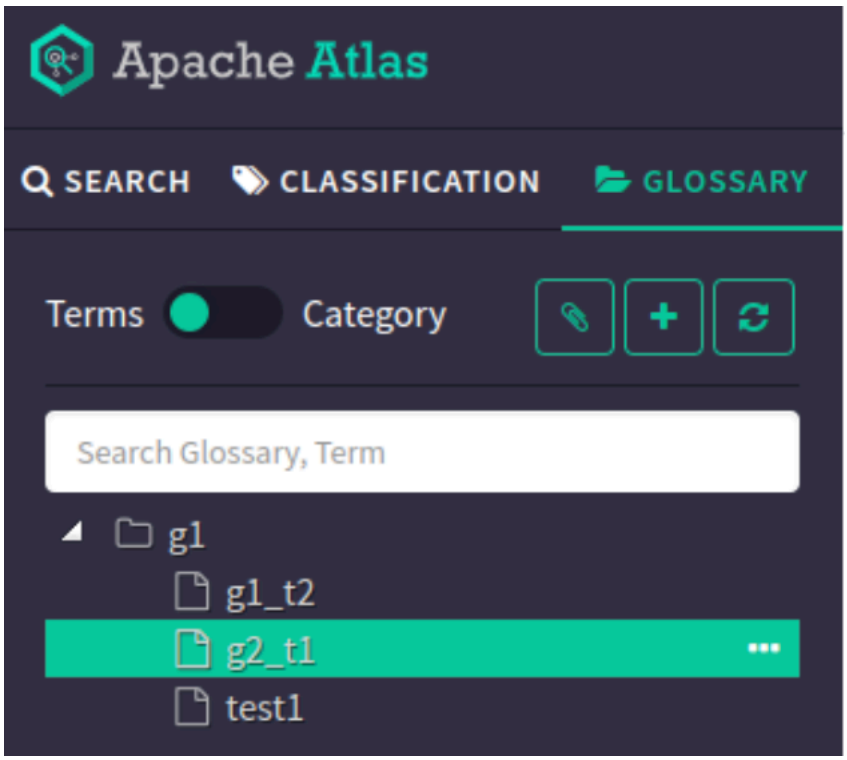

The term named g2_t1 has a parent term named g1, which is displayed on an entity for which g2_t1 is assigned. The parent term is visible for the assigned term in the “assigned term@parent Glossary” format. In this example: g2_t1@g1. Clicking on the assigned term navigates to the Term details page.

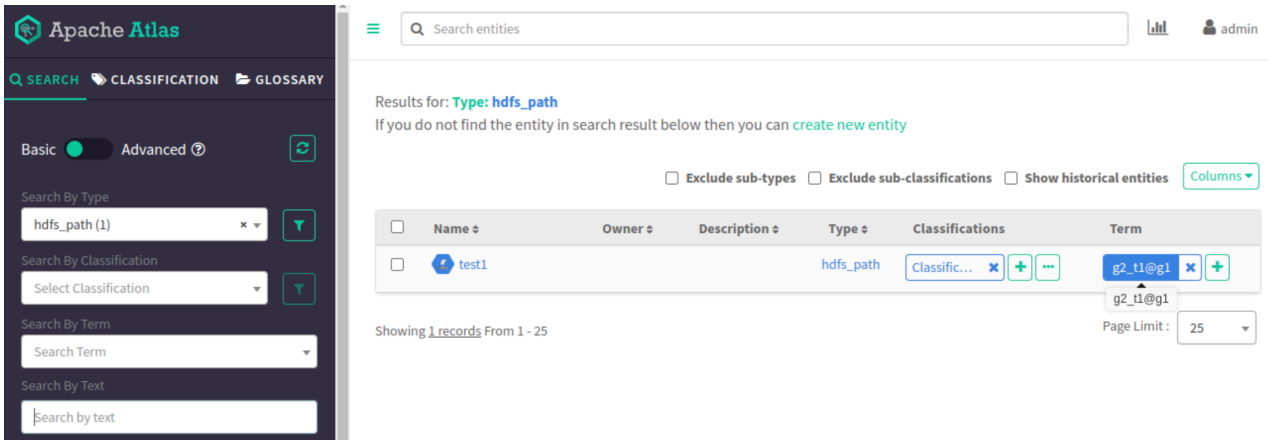

The respective entity pages for classification and term are as shown in the following images:

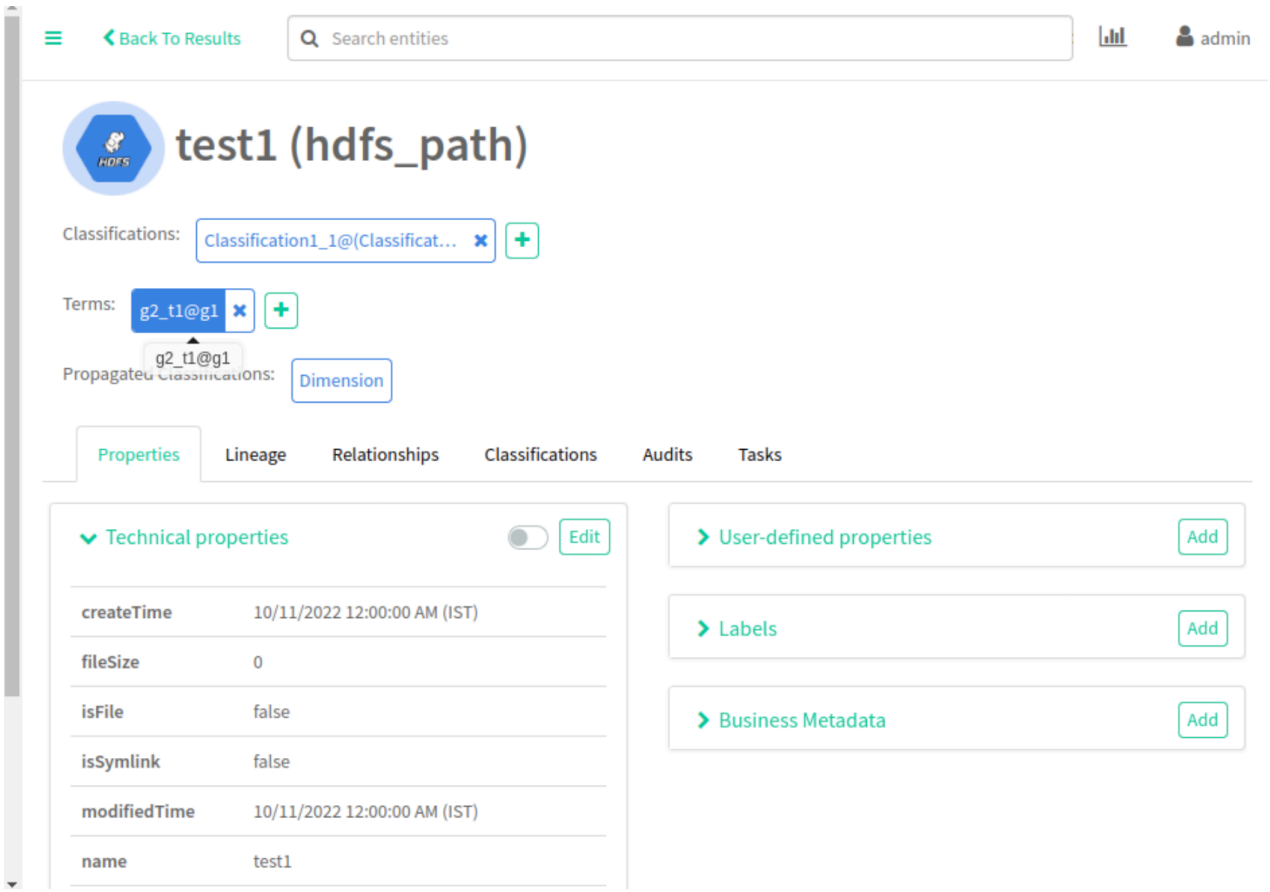
Back To Results
Search entities
admin
test1 (hdfs_path)
Classifications:
Classification1_1@(Classificat...
Terms:
g2_t1@g1
g2_t1@g1
Propagated Classifications:
Dimension
Properties
Lineage
Relationships
Classifications
Audits
Tasks
Technical properties
Edit
User-defined properties
Add
Labels
Add
Business Metadata
Add
createTime
10/11/2022 12:00:00 AM (IST)
fileSize
0
isFile
false
isSymlink
false
modifiedTime
10/11/2022 12:00:00 AM (IST)
name
test1

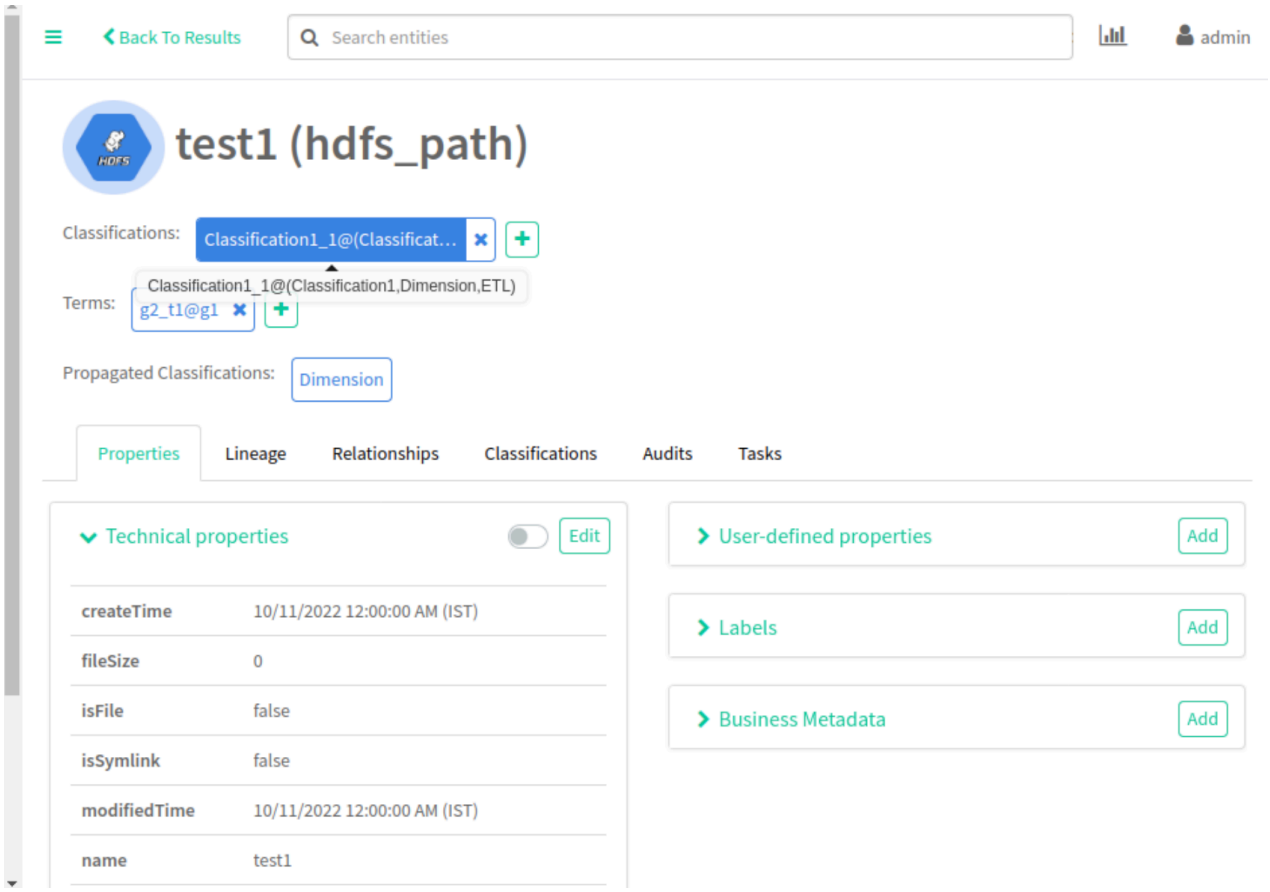
Back To Results
Search entities
admin
test1 (hdfs_path)
Classifications:
Classification1_1@(Classificat...
Classification1_1@(Classification1,Dimension,ETL)
Terms:
g2_t1@g1
Propagated Classifications:
Dimension
Properties
Lineage
Relationships
Classifications
Audits
Tasks
Technical properties
Edit
User-defined properties
Add
Labels
Add
Business Metadata
Add
createTime
10/11/2022 12:00:00 AM (IST)
fileSize
0
isFile
false
isSymlink
false
modifiedTime
10/11/2022 12:00:00 AM (IST)
name
test1

Note: On the Term details page, you can assign terms to “Relation Types” on the “Related Terms” tab. The assigned term can be related to other terms. When you hover over the term, the assigned term displays in the “termname@glossaryname” format.

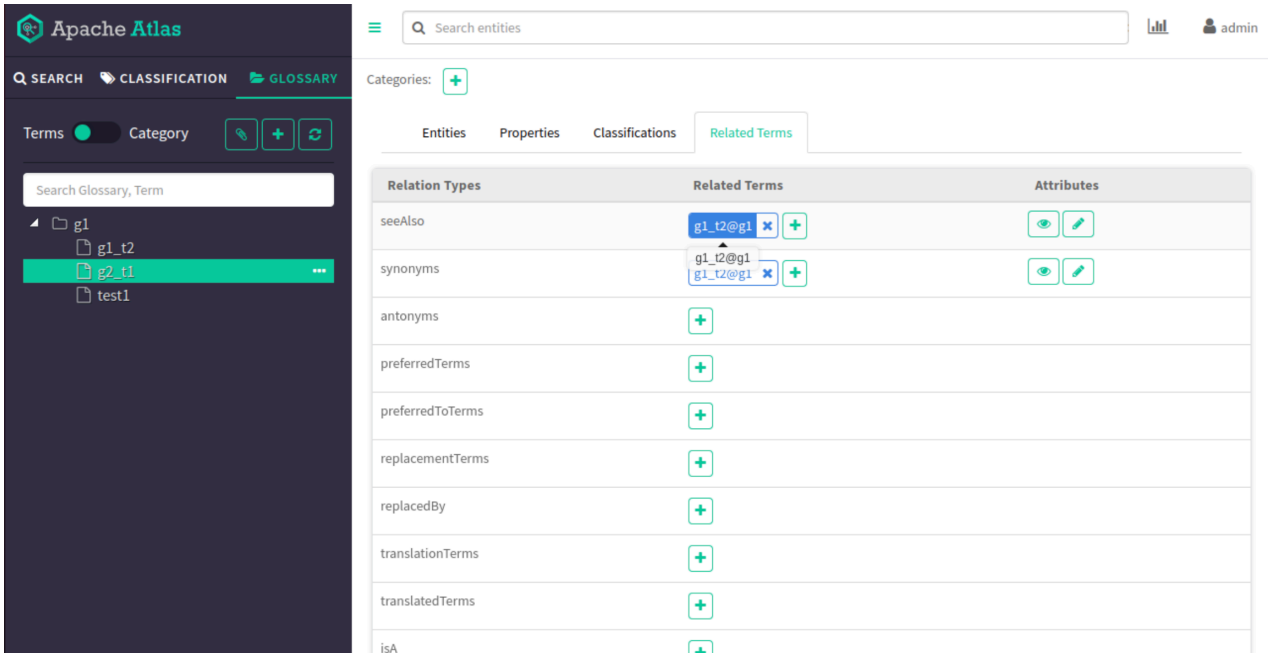

Creating labels

Create and assign a label from the detail page for an entity.

From the detail page for an entity, the Properties tab includes a section for labels:

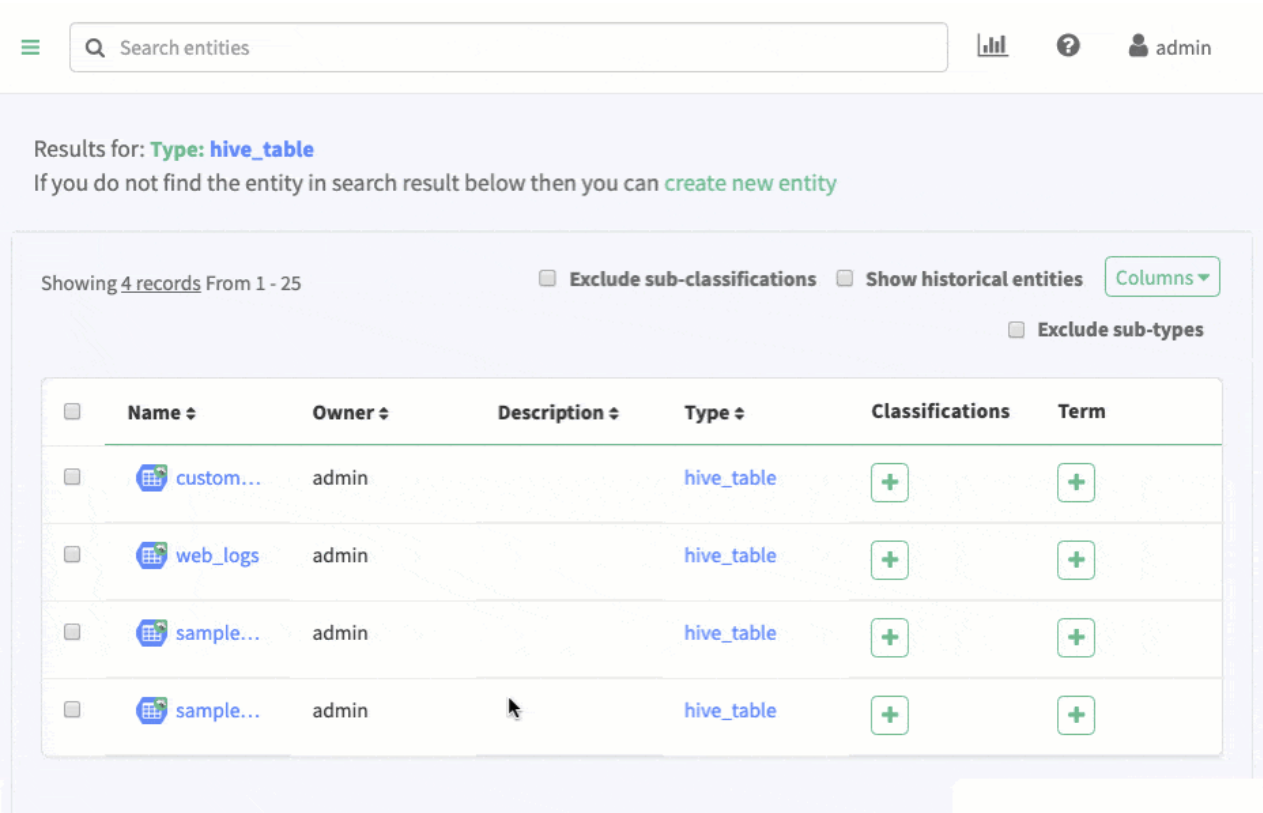

Labels can be 50 characters and contain ASCII characters, hyphens (-) and underscores (_). You can search for entities using the text from labels assigned to the entities.

Related Information

Using Free-text Search

Adding attributes to classifications

Attributes are key-value pairs defined for a classification that you can set to a value when you assign the classification to an entity

For a new classification, you can add attributes as part of the classification definition.

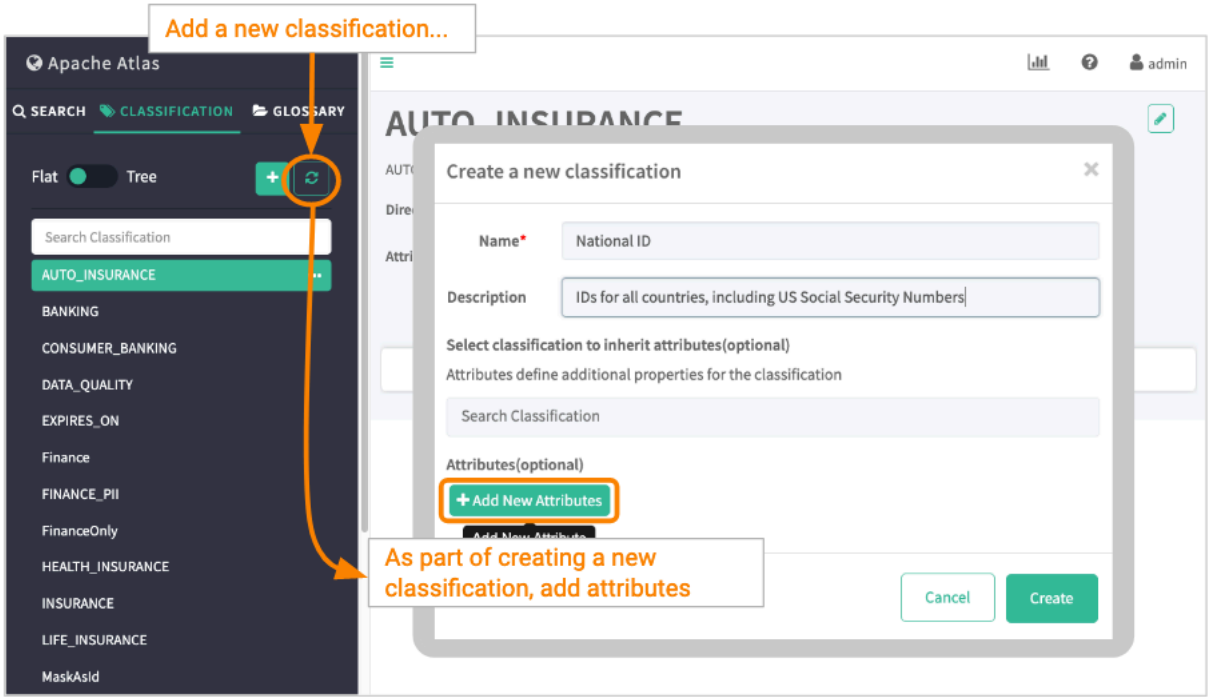

You can also add attributes to an existing classification. Select the classification in the list, then click the plus button next to Attributes.

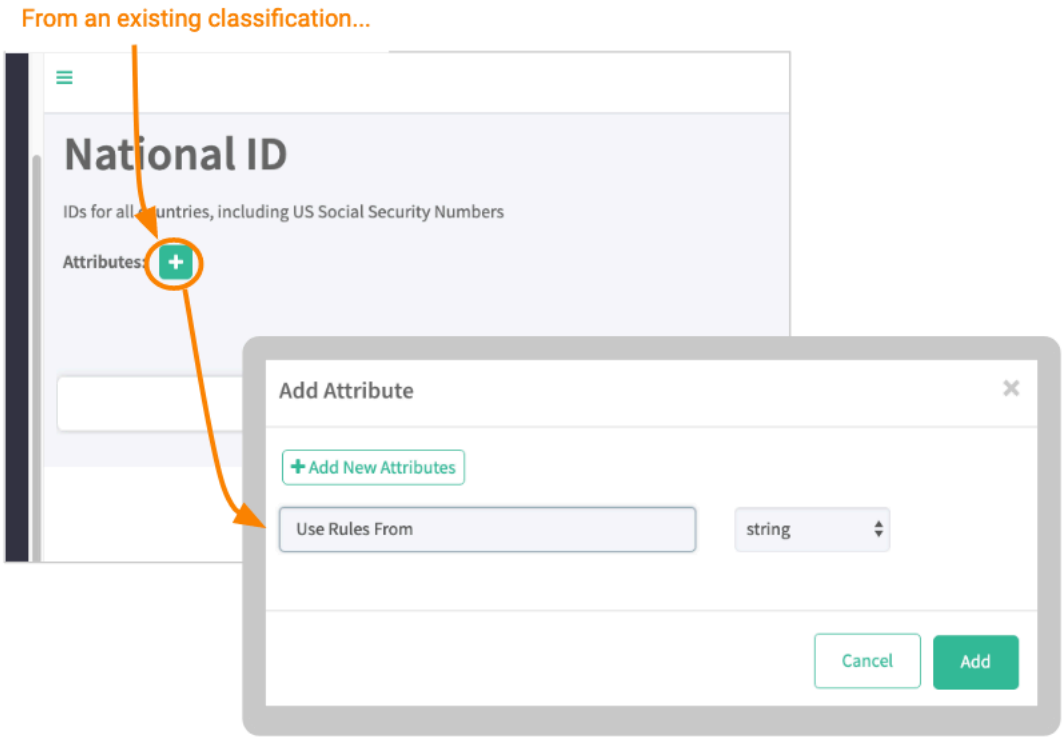

Support for validating the AttributeName in parent and child TypeDef

The Apache Atlas service validates the attribute names of the entity types for those attributes having identical names as their parents' attributes.

Previously, until the Cloudera Runtime 7.1.6 release, Apache Atlas service did not validate the attributes of entity types which led to unpredictable situations, such as being unable to update attribute values or to assign valid values to those attributes in child types.

The checks or the validation of the attributes are performed for multiple use cases pertaining to

Entity and Classification type attribute names. Some of the common scenarios include:

1. Creating Entity or Classification type having attribute name which is same as one of the attributes from its Parent type
2. Updating the existing type to make it as a child of some other type, Atlas validates for the attribute names in both the types.
3. Updating the existing child type by adding an attribute , enables Atlas to validate if that attribute name is already present in its parent-type attributes.

Attention: These validations are also applicable for the Classification types.

Use case 1: If the user attempts to create an Entity or Classification type having an attribute name same as one of the attributes from its parent type, the Atlas service does not allow to create a child type and throws an exception.

As an example:

Consider there is a Type "ParentType" which has attributes "attributeA", "attributeB", and "attributeC". The user wants to create a child type of this "ParentType", say "ChildType". Note that the child type should not have any of the above attributes that are indicated. If any of these attributes are included at the time of "ChildType" creation, the Atlas service does not allow the creation of "ChildType".

Use case 2: If the user attempts to update the existing type and tries to change this type as a child to another type, Atlas service verifies the attribute names in both the types. If Atlas reports that the parent and child type have the same name attributes, Atlas does not allow updating the type and raises an exception.

As an example:

Consider there are 2 types "MyType01" having attribute "attributeA" and "MyType02" also having "attributeA". Atlas does not allow updating "MyType02" to become the child type of "MyType01" (and vice versa).

Use case 3: If a user attempts to update the existing child type and tries to add any attribute in child type, the Atlas service checks if that attribute name is already present in its parent-type attributes. If the attribute is present, Atlas service does not allow adding the attribute in child type and raises an exception.

As an example:

Suppose there are 2 types: "ParentType" having attribute "attributeA" and "ChildType" which is a child of "ParentType" and having attribute "attributeB". If a user tries to update the "ChildType" by adding an attribute "attributeA" in it, Atlas service does not allow adding the attribute in the "ChildType".

Validations for parent types

In Atlas, validations for scenarios where multiple parent types are involved for both entity types as well as Classifications.

Note: These validations are also applicable for the Classification types.

Use case 1: When there are two types having same name attributes, these two types cannot be assigned as parent-types to a child type. If a user tries to create such a child type, Atlas service does not allow it to create child type and raises an exception.

As an example:

Suppose two types "MyType01" and "MyType02" have a common attribute "attributeA".

A user tries to create a "ChildType" type by assigning both types "MyType01" and "MyType02" as its parents. Atlas service does not allow creation of "ChildType" because both parent types have the same name attribute.

Use case 2: A child type cannot be updated as a child of two parent types which have the same name attribute. If a user tries to update such a child type, Atlas service does not allow that and raises an exception.

As an example:

Suppose a type "ParentType01" has attribute "attributeA" and its child type is "ChildType". Also, another independent type "MyType02" which has the attribute "attributeA". The user wants to update the "ChildType" and tries to make it a child of "MyType02". Atlas service does not allow updating "ChildType", because both parent types have the same name attribute.

Case for implementing backward compatibility

When users of older versions want to upgrade their Apache Atlas versions to the latest one, they might face an issue because of certain validation checks in Atlas.

Existing types with attributes violating the attribute name rules (which were added in later Atlas versions) may have been added to the system.

All the existing types get populated at the time of Atlas startup and they all get validated.

If the users have Entity types or Classifications in the system that violate the naming rule, then Atlas raises an exception at the time of the service startup if the users have upgraded their environment to the latest version.

As a result of this exception that results at the time of Atlas startup, the typedef registry does not get populated. Even though users can log in to Atlas UI, they cannot view or use any types or Entities.

When users having data with such types try to upgrade their environment to the latest bits, they encounter issues as discussed earlier in this section.

Also, in their application.log they see an exception with a system output:

Error code: "ATLAS-400-00-09E" and Error message: "Invalid attribute name: <type name>.<attribute name>. Attribute already exists in another parent type: <parent type name>"

The backward compatibility feature is built to avoid these scenarios for such Entity types and Classifications.

How backward compatibility works

The backward compatibility feature is provided to skip exceptions coming from the attribute name rule.

From your Cloudera Manager instance > Configuration tab, a property must be added in your CM Safety Valve configuration.

atlas.skip.check.for.parent.child.attribute.name=true

Later, restart the Atlas service to apply this configuration.

When Atlas starts, no exceptions are raised and all the typedefs and Entities are displayed and available for use.

What Next:

After adding the backward compatibility configuration, all the validations and checks are disabled. You can create / update the types which have the same name attributes as their parent type attributes. The Atlas service operates in the same way as it was behaving prior to the Cloudera Runtime 7.1.7 version.

Note: When this feature is disabled, there will be corresponding Warning logs generated against those entity types, which may contain the information like this example.

Error code: "ATLAS-400-00-09E" and Error message: "Invalid attribute name: <type name>.<attribute name>. Attribute already exists in another parent type: <parent type name>"

If the user does not want any of their migrated existing types (which violate the name rule) to be present in the current environment, the user can delete these types using the Atlas REST APIs and later remove the configuration from the Cloudera Manager Safety Valve. Finally, restart the Atlas service.

Caution: Before deleting the typedefs, if there are any entities belonging to those types in the system, then the user must first delete and then purge those entities.

Associating classifications with entities

Assign classifications to entities in the entity detail page or in the search results.

There are two ways to associate a classification with an entity:

- Add the classification to the entity in the entity detail page.

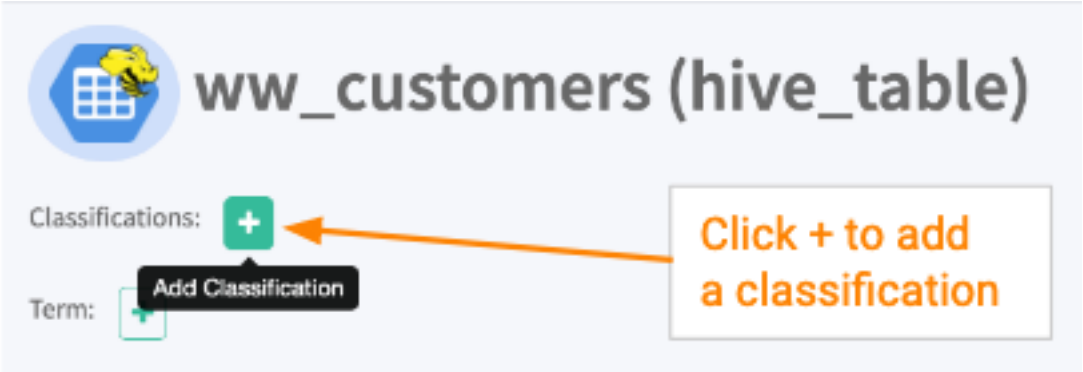

- Add the classification to the entity in the search results.

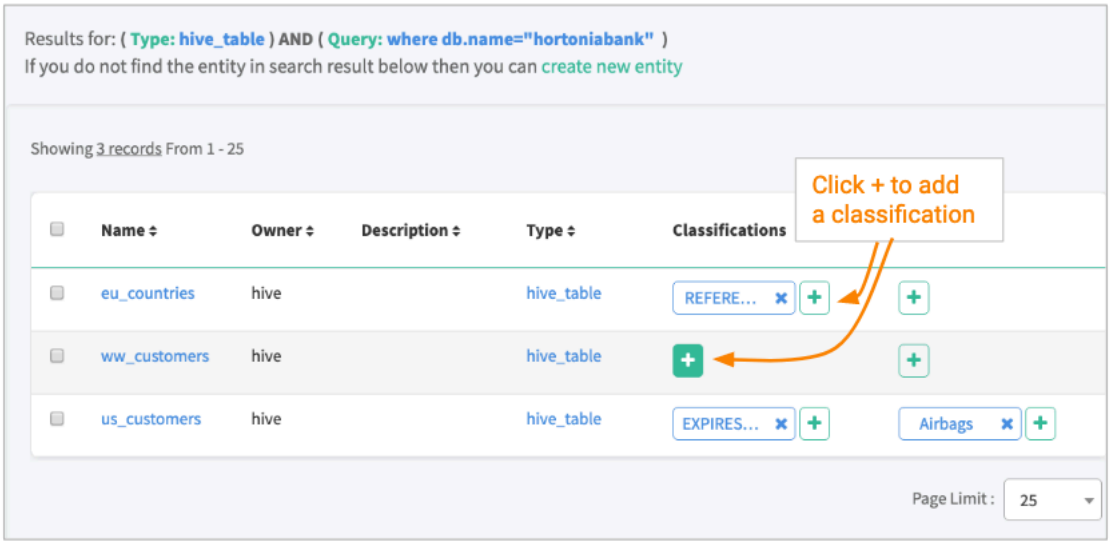

When you associate a classification with an entity, you can set the classification's behavior with these controls:

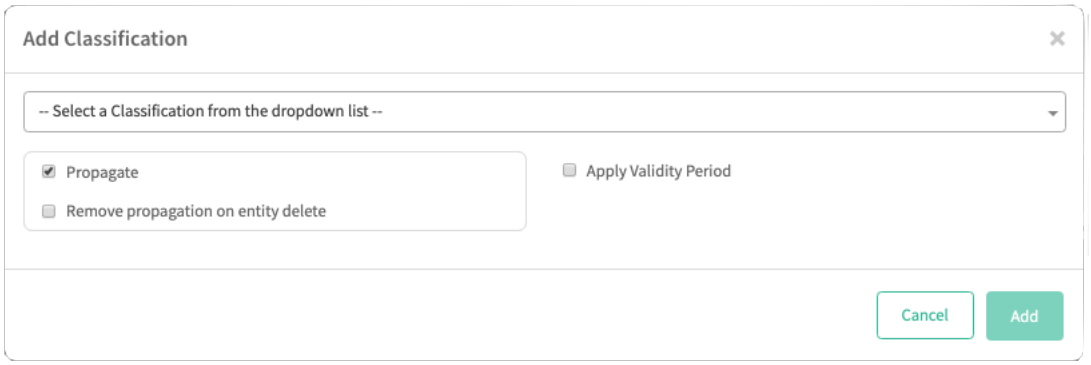

- Propagate: Any entities in the impact chain of this entity will automatically be tagged with this classification. The classification can't be removed from the "downstream" entities without being removed from this source entity. See more at Propagate classifications to derived entities.
- Apply validity period: You can set a time period after which this classification is removed from the entity. Set the start time or end time or both. If you do not choose a time zone, the local time of the Atlas server is used.
- Remove propagation on entity delete: When an entity is deleted in the source and marked as deleted in Atlas, you can choose to maintain the propagation path, or to break propagation at the deleted entity.

You can change the behavior of classifications after they are assigned: go to the Classifications tab for an entity.

You can remove classifications from an entity using the X button on search results or entity detail page. Don't see an X on the classification name? That classification was assigned through propagation; to remove it, you have to turn off propagation for the source entity or term.

Related Information

Propagating classifications through lineage

Propagating classifications through lineage

Classifications on table columns can automatically apply to other tables where the same data is found when the data transformations are captured in lineage.

Viewing propagated classifications

When you look at the classifications assign to an entity, you can see which classifications where originally assigned to this entity and which were propagated here through lineage. From the Classifications tab, you can:

- Show or hide propagated classifications (Show Propagated Classifications button)
- Navigate to the original entity where a classification was assigned (Propagated From button)

Navigate to the source of the classification

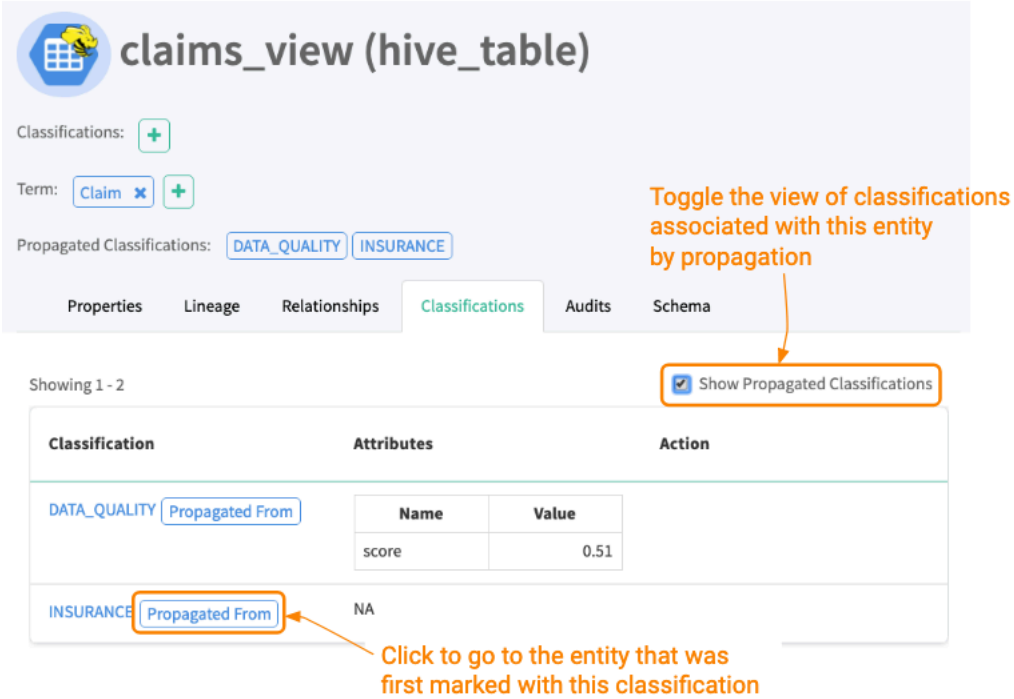

Setting propagation behavior

You can set propagation for a classification when you create the classification or in the lineage for an entity where the classification is assigned:

- Set propagation at the time when the classification is associated with an entity or term.

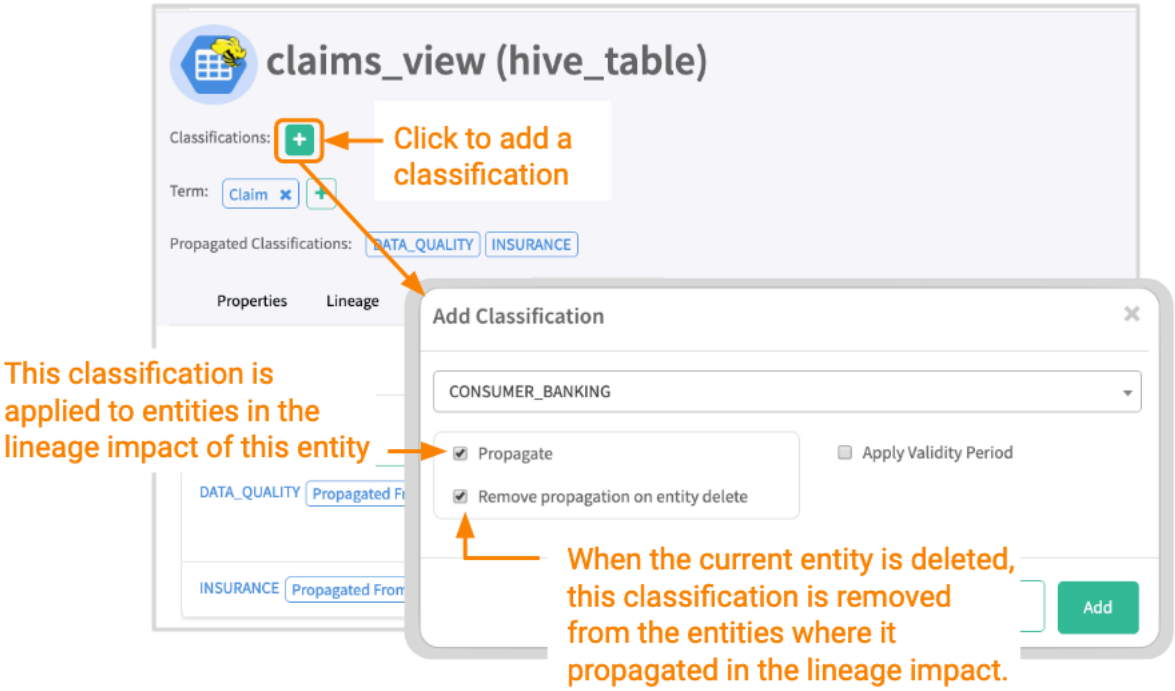

- Set propagation at a specific point in the lineage.

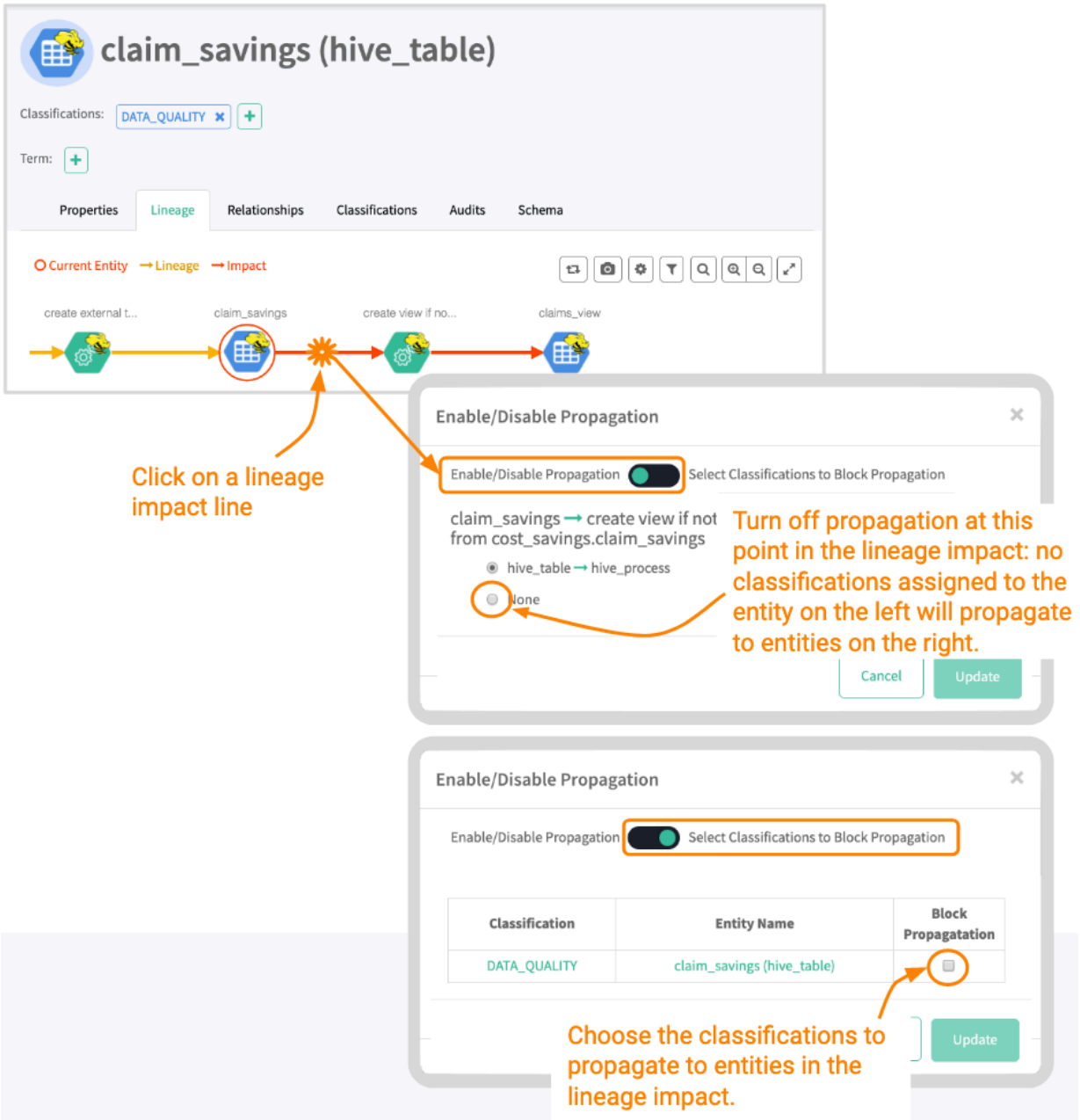

Related Information

Associating classifications with entities

Controlling data access using tags

Searching for entities using classifications

After you tag entities with classifications, it's very easy to search for the entities.

The Atlas dashboard makes it easy to find entities that you've tagged with classifications: in the left navigation pane, go to the Classification tab and select the classification you want. You can refine your search results by adding criteria on the Search tab (Basic search).

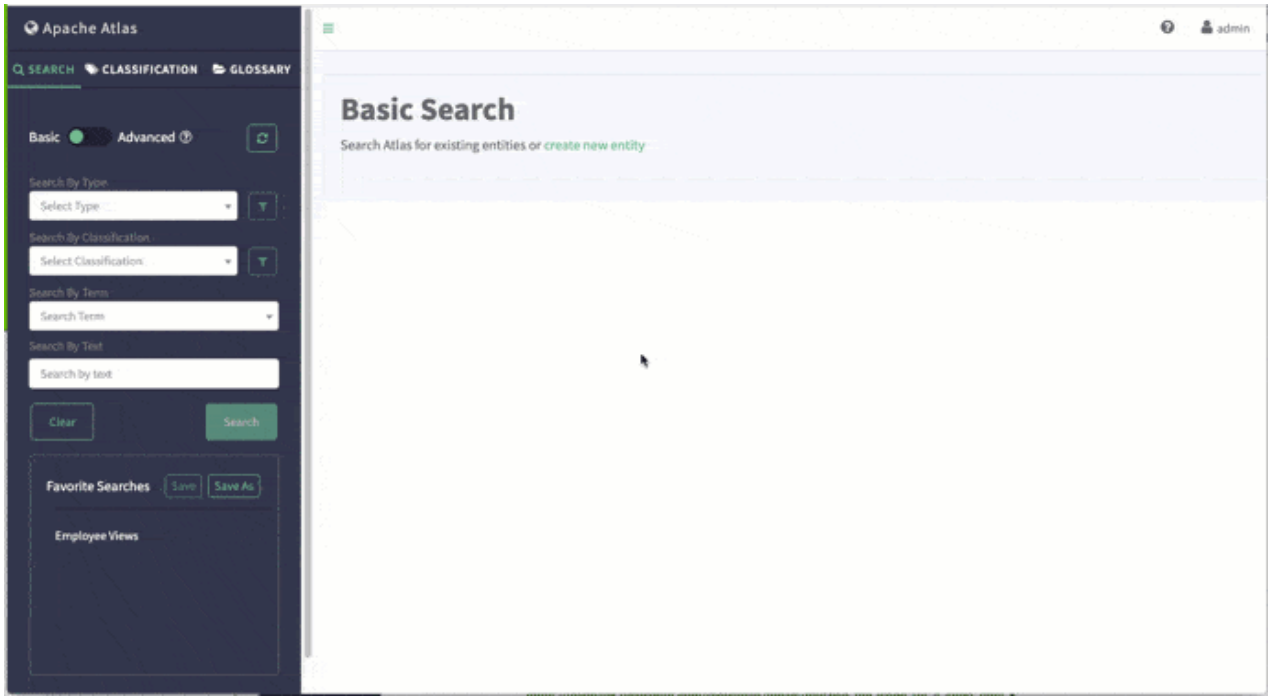

Related Information

Using Basic Search

Cloudera Runtime 7.3.1

Configuring Cruise Control

Date published: 2019-08-22
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Setting capacity estimations and goals.....4
Configuring capacity estimations.....4
Configuring goals.....5
Example of Cruise Control goal configuration.....6
Multi-level rack-aware distribution goal.....7

Configuring Metrics Reporter in Cruise Control.....7

Adding self-healing goals to Cruise Control in Cloudera Manager.....8

Setting capacity estimations and goals

Cruise Control rebalancing works using capacity estimations and goals. You need to configure the capacity estimates based on your resources, and set the goals for Cruise Control to achieve the Kafka partition rebalancing that meets your requirements.

When configuring Cruise Control, you need to make sure that the Kafka topics and partitions, the capacity estimates, and the proper goals are provided so the rebalancing process works as expected.

You can find the capacity estimate and goal configurations at the following location in Cloudera Manager:

1. Access Cloudera Manager for the Cruise Control configurations.
   a. Navigate to Cloudera Management Console Environments , and select the environment where you have created your cluster.
   b. Select Cloudera Manager from the services.
   c. Select Clusters Cruise Control .
2. Click Configuration.
3. Select Main from the Filters.

Configuring capacity estimations

The values for capacity estimation needs to be provided based on your available resources for CPU and network. Beside the capacity estimation, you also need to provide information about the broker and partition metrics. You can set the capacity estimations and Kafka properties in Cloudera Manager.

For the rebalancing, you need to provide the capacity values of your resources. These values are used for specifying the rebalancing criteria for your deployment. The following capacity values must be set:

Capacity	Description
capacity.default.cpu	100 by default
capacity.default.network-in	Given by the internet provider
capacity.default.network-out	

Note: For the capacity estimates, the disk capacity value is also needed. However, Cruise Control automatically retrieves the disk capacity value from the kafka_log_directory_disk_total_space Kafka metric.

The optimizers in Cruise Control use the network incoming and outgoing capacities to define a boundary for optimization. The capacity estimates are generated and read by Cruise Control. A capacity.json file is generated when Cruise Control is started. When a new broker is added, Cruise Control uses the default broker capacity values. However, in case disk related goals are used, Cruise Control must be restarted to load the actual disk capacity metrics of the new broker.

The following table lists all the configurations that are needed to configure Cruise Control specifically to your environment:

Note: The security settings are not listed in the table below.

Configuration	Description
num.metric.fetchers	Parallel threads for fetching metrics from the Cloudera Manager database
partition.metric.sample.store.topic	Storing Cruise Control metrics
broker.metric.sample.store.topic	Storing Cruise Control metircs

Configuration	Description
partition.metrics.window.ms	Time window size for partition metrics
broker.metrics.window.ms	Time window size for broker metrics
num.partition.metrics.windows	Number of stored partition windows
num.broker.metrics.windows	Number of stored broker windows

Configuring goals

After setting the capacity estimates, you can specify which goals need to be used for the rebalancing process in Cloudera Manager. The provided goals are used for the optimization proposal of your Kafka cluster.

Procedure

1. Access Cloudera Manager for the Cruise Control configurations.
   a) Navigate to Cloudera Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streams Messaging cluster from the list of Cloudera Data Hub clusters.
   c) Select Cloudera Manager from the services.
   d) Select Clusters Cruise Control .
   e) Click Configuration.
2. Search for goals using the search bar.
   The list of goals are displayed based on the goal sets.
3. Add goals using the property name to the Default, Supported, Hard, Self-healing and Anomaly detection lists based on your requirements, and click Save Changes.

   The following table lists the goals that can be used:

Goal	Property name	Description
`RackAwareDistributionGoal`	com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareDistributionGoal	As long as replicas of each partition can achieve a perfectly even distribution across the racks, this goal lets placement of multiple replicas of a partition into a single rack.
`ReplicaCapacityGoal`	com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaCapacityGoal	Attempt to make all the brokers in a cluster to have less than a given number of replicas.
`CapacityGoals`	com.linkedin.kafka.cruisecontrol.analyzer.goals.DiskCapacityGoal	Goals that ensure the broker resource utilization is below a given threshold for the corresponding resource.
	com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkInboundCapacityGoal	
	com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkOutboundCapacityGoal	
	com.linkedin.kafka.cruisecontrol.analyzer.goals.CpuCapacityGoal	
`ReplicaDistributionGoal`	com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaDistributionGoal	Attempt to make all the brokers in a cluster to have a similar number of replicas.
`PotentialNwOutGoal`	com.linkedin.kafka.cruisecontrol.analyzer.goals.PotentialNwOutGoal	A goal that ensures the potential network output (when all the replicas become leaders) on each of the brokers do not exceed the broker’s network outbound bandwidth capacity.

Goal	Property name	Description
`ResourceDistributionGoals`	com.linkedin.kafka.cruisecontrol.analyzer.goals.DiskUsageDistributionGoal	Attempt to make the resource utilization variance among all the brokers are within a certain range. This goal does not do anything if the cluster is in a low utilization mode (when all the resource utilization of each broker is below a configured percentage.) This is not a single goal, but consists of the following separate goals for each of the resources.
	com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkInboundUsageDistributionGoal	
	com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkOutboundUsageDistributionGoal	
	com.linkedin.kafka.cruisecontrol.analyzer.goals.CpuUsageDistributionGoal	
`TopicReplicaDistributionGoal`	com.linkedin.kafka.cruisecontrol.analyzer.goals.TopicReplicaDistributionGoal	Attempt to make the replicas of the same topic evenly distributed across the entire cluster.
`LeaderReplicaDistributionGoal`	com.linkedin.kafka.cruisecontrol.analyzer.goals.LeaderReplicaDistributionGoal	Attempt to make all the brokers in a cluster to have the similar number of leader replicas.
`LeaderBytesInDistributionGoal`	com.linkedin.kafka.cruisecontrol.analyzer.goals.LeaderBytesInDistributionGoal	Attempt to make the leader bytes in rate on each host to be balanced.
`PreferredLeaderElectionGoal`	com.linkedin.kafka.cruisecontrol.analyzer.goals.PreferredLeaderElectionGoal	Attempt to make the first replica in the replica list leader replica of the partition for all topic partitions.
`MinTopicLeadersPerBrokerGoal`	com.linkedin.kafka.cruisecontrol.analyzer.goals.MinTopicLeadersPerBrokerGoal	Ensures that each alive broker has at least a certain number of leader replica of each topic in a configured set of topics
`KafkaAssignerGoals`[1]	com.linkedin.kafka.cruisecontrol.analyzer.kafkaassigner.KafkaAssignerDiskUsageDistributionGoal	A goal that ensures all the replicas of each partition are assigned in a rack aware manner.
	com.linkedin.kafka.cruisecontrol.analyzer.kafkaassigner.KafkaAssignerEvenRackAwareGoal	Attempt to make all the brokers in a cluster to have the similar number of replicas

Example of Cruise Control goal configuration

By default, Cruise Control is configured with a set of Default, Supported, Hard, Self-healing and Anomaly detection goals in Cloudera Manager. The default configurations can be changed based on what you would like to achieve with the rebalancing.

The following example details how to configure Cruise Control to achieve the following:

- Find dead/failed brokers and create an anomaly to remove load from them (self.healing.broker.failure.enabled)
- Move load back to the brokers when the brokers are available again (self.healing.goal.violation.enabled and added goals)
- Prevent too frequent rebalances to reduce cluster costs (incremented thresholds, reduced self.healing.goals set)
- Have an always balanced cluster from the replicas and leader replicas point of view
- Not enable every type of self-healing methods if it is not required (only two type of self-healing is enabled)

Configurations that need to be added to the Cruise Control Server Advanced Configuration Snippet (Safety Valve) for cruisecontrol.properties property:

- self.healing.goal.violation.enabled=true
- self.healing.broker.failure.enabled=true
- self.healing.exclude.recently.removed.brokers=false

Configurations that need to be set (and available explicitly among properties):

- anomaly.notifier.class=com.linkedin.kafka.cruisecontrol.detector.notifier.SelfHealingNotifier

[1] These goals are used to make Cruise Control behave like a Kafka assigner tool. These goals will be picked up if kafka_assigner parameter is set to true in the corresponding request (for example, with the rebalance request as shown in the Cruise Control documentation).

- replica.count.balance.threshold=1.25
- leader.replica.count.balance.threshold=1.25

Goals that need to be added to Hard goals:

- com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaDistributionGoal
- com.linkedin.kafka.cruisecontrol.analyzer.goals.LeaderReplicaDistributionGoal

Goals that need to be added to Self-healing goals:

- com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaDistributionGoal
- com.linkedin.kafka.cruisecontrol.analyzer.goals.LeaderReplicaDistributionGoal

Goals that need to be added to Anomaly detection goals:

- com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaDistributionGoal
- com.linkedin.kafka.cruisecontrol.analyzer.goals.LeaderReplicaDistributionGoal

Other configurations can remain as set by default.

Multi-level rack-aware distribution goal

You can use the MultiLevelRackAwareDistributionGoal to ensure rack awareness on a higher level than for the standard rack aware goal for Kafka clusters using Cruise Control.

The MultiLevelRackAwareDistributionGoal behaves differently than the default RackAwareGoal or RackAwareDistributionGoal in Cruise Control. The standard goals have lighter requirements on rack awareness, and always optimize based on the current state of the cluster and with the priority on making all replicas come back online.

This means that in case a network partition failure occurs, and a data center goes offline, a Cruise Control rebalance operation using a standard rack-aware goal ignores the data center that is not working, and moves replicas around as if there were one fewer data center in the cluster. For example, if a Kafka cluster has three data centers and one goes offline, the standard goals are not aware of the existence of the third data center, and act as if only two data centers are used in the cluster.

The MultiLevelRackAwareDistributionGoal acts differently in the following aspects:

- Handles rack IDs as multi-level rack IDs, respecting the hierarchy of racks when distributing replicas
- Keeps track of the whole state of the cluster with caching previous states to make sure that all racks are visible
- Prioritizes multi-level rack awareness guarantees over bringing all replicas back online

In the same failure situation, where one data center is offline out of three, the multi-level rack-aware goal is still aware of the existence of the third data center. This means that the offline replicas are not moved from the third data center if the migration violates the multi-level rack awareness guarantees. The goal allows optimizations to pass even in the presence of offline replicas, which can be configured with cloudera.multi.level.rack.awareness.ensure.no.offline.replicas property. If the cloudera.multi.level.rack.awareness.ensure.no.offline.replicas is set to true, the goal causes the rebalance operation to fail if the replicas would stay offline after the optimizations are implemented.

Configuring Metrics Reporter in Cruise Control

You can choose between using the default Cruise Control Metrics Reporter or using the Cloudera Manager Metrics Reporter for fetching metrics in Cruise Control. Cloudera recommends using the Cloudera Manager solution with light installation, and the default solution with heavy installations of Kafka deployments.

Procedure

1. Access Cloudera Manager for the Cruise Control configurations.
   a) Navigate to Cloudera Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streams Messaging cluster from the list of Cloudera Data Hub clusters.
   c) Select Cloudera Manager from the services.
   d) Select Clusters Cruise Control .
   e) Click Configuration.
2. Search for Metrics Reporter.
3. Select Cloudera Manager metrics reporter or Cruise Control metrics reporter based on your requirements.
4. Click Save changes.
5. Click on Action Restart next to the Cruise Control service name to restart Cruise Control.

Adding self-healing goals to Cruise Control in Cloudera Manager

As self-healing is enabled by default for Cruise Control, you only need to specify the actions Cruise Control should take when detecting anomaly types by providing self-healing goals in Cloudera Manager.

Procedure

1. Access Cloudera Manager for the Cruise Control configurations.
   a) Navigate to Cloudera Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streams Messaging cluster from the list of Cloudera Data Hub clusters.
   c) Select Cloudera Manager from the services.
   d) Select Clusters Cruise Control .
   e) Click Configuration.
2. Search for Self-Healing Goals.
3. Add the required self-healing goals to the corresponding field.
4. Click Save changes.
5. Click on Action > Restart next to the Cruise Control service name to restart Cruise Control.

Cloudera Runtime 7.3.1

Managing Cruise Control

Date published: 2019-08-22
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Rebalancing with Cruise Control..... 4

Cruise Control REST API endpoints..... 4

Rebalancing with Cruise Control

In Cloudera Public Cloud, when you scale your Kafka nodes that are running in your Cloudera Data Hub clusters, Cruise Control is responsible for rebalancing the partitions based on the goals previously set.

When using the scaling functionality of Cloudera Data Hub, you can set the number of Kafka nodes that you want to add or you want to remove, and the partition rebalancing is automatically completed in the background with Cruise Control. You do not need to manually initiate the rebalancing process after managing the Kafka brokers, as the Resize action in Cloudera Data Hub does this automatically.

For more information, see the *Scaling Kafka broker nodes in a Streams Messaging cluster* documentation.

Related Information

Scaling Kafka broker nodes in a Streams Messaging cluster

Cruise Control REST API endpoints

You can configure and check the status of Cruise Control using the REST API with GET and POST endpoints. You must be aware of the required action or query, and submit a curl command in the command line to use the Cruise Control REST API.

There are two types of REST endpoints for Cruise Control: GET and POST.

For GET

With GET endpoints, you can query information about the rebalancing process, the status of Cruise Control and Kafka brokers. The GET endpoints are read-only operations, and do not have any impact on Cruise Control, Kafka, or the rebalancing process.

For POST

You can use the POST endpoints to change the rebalancing process, modify the number of Kafka brokers, configure Cruise Control, and customize certain sampling and proposal tasks.

Prerequisite for Cloudera Public Cloud

The ADMIN, USER and VIEWER levels define which endpoint a user can call when managing Cruise Control. After provisioning your Streams Messaging cluster, you must set an access level for user authentication in Cloudera Manager to be able to call the Cruise Control API endpoint.

1. Navigate to Cloudera Management Console Environments , and select the environment where you have created your cluster.
2. Select Cloudera Manager from the services.
3. Select Clusters Cruise Control .
4. Click Configuration.
5. Search for one of the following access levels based on which level you want to add the user:

Security property		Description
ADMIN Level Users	auth_admins	The list of ADMIN level users to have access to all endpoints. The ADMIN level also contains the permissions of the USER and VIEWER level.

Security property		Description
USER Level Users	auth_users	The list of USER level users to have access to all the GET endpoints except bootstrap and train. The USER level also contains the permissions of the VIEWER level.
VIEWER Level Users	auth_viewers	The list of VIEWER level users to have access to the most lightweight kafka_cl uster_state, user_tasks and review_board endpoints.

6. Click Save Changes.
7. Restart the Cruise Control service.

Using the Cruise Control REST API endpoints

In the command line, you need to create the POST or GET commands with curl, and also include the Cruise Control hostname, port, and the required endpoint.

For GET

```
curl -X -u <workload_username>:<workload_password> GET "https://[***MY-D
ATA-HUB- CLUSTER.COM***]/cdp-proxy-api/cruise-control/kafkacruisecontrol/
kafka_cluster_state”
```

For POST

```
curl -X -u <workload_username>:<workload_password> POST “https://[***MY-
DATA-HUB- CLUSTER.COM***]/cdp-proxy-api/cruise-control/kafkacruisecontrol/
add_broker?brokerid=24”
```

You can find the API base URL for Cruise Control on the Endpoints tab of the **Cluster Details** page: Cloudera Management Console Data Hub Clusters Your Cluster Endpoints .

The following table summarizes the available operations for GET and POST endpoints. The description column only provides a brief summary of the endpoints. For the detailed description of the Cruise Control REST API, see the official REST API documentation.

Note: Most of the POST actions has a dry-run mode. In dry-run mode, the actions only generate the proposals and estimated results, and not trigger the execution of the proposals. To avoid accidentally triggering the data movement, all of the POST actions are in dry run mode by default. To let Cruise Control actually move the data, you need to explicitly set dryrun=false.

For GET

Endpoint	Action	Description
/kafkacruisecontrol/bootstrap	You can start the startup process of Cruise Control and reload all data from Kafka.	This feature can only be used in development mode. The bootstrap endpoint clears the internal state. You can run the bootstrap endpoint in the following modes: • range - an interval is specified • since - a starting point is specified • recent - most recent metric samples are used
/kafkacruisecontrol/kafka_cluster_state	You can query partition and replica state.	This includes information for each broker and each partition.
/kafkacruisecontrol/load	You can query the current cluster load.	The Load Monitor needs to be in RUNNING state. This includes the load-per-broker and load-per-host information. Only shows valid partitions with enough metric samples.

Endpoint	Action	Description
/kafkacruisecontrol/partition_load	You can query partition resource utilization.	The partition load is sorted based on the given resource.
/kafkacruisecontrol/permissions	You can query the list of permissions set for the current user	This endpoint can be used to review the configured permissions of the current user that include ADMIN, USER and VIEWER levels. Permissions are not used when security is enabled for Cruise Control.
/kafkacruisecontrol/proposals	You can query the optimization proposals generated on the workload model.	The proposals can be generated based on either the snapshot window or the valid partitions.
/kafkacruisecontrol/review_board	You can check the review processes of Cruise Control.	This endpoint shows the status of ongoing review processes in Cruise Control.
/kafkacruisecontrol/state	You can query the state of Kafka Cruise Control.	This includes the state of monitor, executor, analyzer, and anomaly detector.
/kafkacruisecontrol/train	You can use the endpoint to train the partition of Cruise Control for CPU estimation.	This endpoint uses a regression algorithm to train the CPU estimation of the Cruise Control partition. This is an experimental feature.
/kafkacruisecontrol/user_tasks	You can query the user request result.	This includes a full list of all the active and completed tasks in Cruise Control.

For POST

Endpoint	Action	Description
/kafkacruisecontrol/add_broker?brokerid=[id1,id2...]	You can add a list of brokers from a Kafka cluster.	Only moves replicas from existing brokers to the newly added brokers. You can choose to throttle the movement of the replica to the newly added broker. The replica movement is throttled on the existing broker. Dry-run mode is activated by default, set dryrun=false to trigger adding brokers.
/kafkacruisecontrol/admin?concurrent_partition_movements_per_broker=[integer]	You can increase or decrease the execution concurrency.	The configuration is only possible with administrator rights.
/kafkacruisecontrol/admin?disable_self_healing_for=[anomaly_type]	You can enable or disable self-healing for Cruise Control.	The configuration is only possible with administrator rights.
/kafkacruisecontrol/admin?drop_recently_removed_brokers=[broker_ids]	You can drop recently removed or demoted Kafka brokers.	The configuration is only possible with administrator rights.
/kafkacruisecontrol/demote_broker?brokerid=[id1, id2...]	You can move all the leader replicas from a list of brokers.	This makes all the replicas on a given broker to be preferred the least, and triggers a preferred leader replica election to migrate the leader replicas off the broker. Dry-run mode is activated by default, set dryrun=false to trigger demoting brokers.
/kafkacruisecontrol/demote_broker?brokerid_and_logdirs=[id1-logdir1, id2-logdir2...]	You can move all the leader replicas from a list of disks.	This makes all the replicas on a given disk to be preferred the least, and triggers a preferred leader replica election to migrate the leader replicas off the disk.
/kafkacruisecontrol/fix_offline_replicas	You can fix offline replicas in a Kafka cluster.	If the specified topic has offline replicas, they are still moved to healthy brokers. Dry-run mode is activated by default, set dryrun=false to trigger fixing offline replicas.
/kafkacruisecontrol/pause_sampling	You can pause an ongoing metrics sampling process.	The cause of the pause is recorded and shows up when querying the state of Cruise Control.

Endpoint	Action	Description
/kafkacruisecontrol/rebalance	You can trigger a workload balance for a Kafka cluster.	Can be based on valid windows or valid partitions. You can specify the goal you want to use. Dry-run mode is activated by default, set dryrun=false to trigger rebalancing.
/kafkacruisecontrol/remove_broker?brokerid=[id1,id2...]	You can remove a list of brokers from a Kafka cluster.	Only moves partitions from the brokers to be removed to the other existing brokers. You can also specify the destination broker. You can choose to throttle the removed broker during the partition movement. Dry-run mode is activated by default, set dryrun=false to trigger removing brokers.
/kafkacruisecontrol/resume_sampling	You can resume a paused metrics load sampling.	The cause of the resumption is recorded and shows up when querying the state of Cruise Control.
/kafkacruisecontrol/review	You can review the pending Cruise Control requests.	You can check the previously submitted requests in Cruise Control.
/kafkacruisecontrol/stop_proposal_execution	You can stop an ongoing proposal execution task.	You can stop a proposal when the execution of a task is still ongoing.
/kafkacruisecontrol/topic_configuration?topic=[topic_regex]&replication_factor=[target_replication_factor]	You can change the replication factor of a topic.	Existing replicas are not moved. You can determine which replica should be deleted and which broker should be assigned to the new replica. Dry-run mode is activated by default, set dryrun=false to trigger changing the replication factor of a topic.

Cloudera Runtime 7.3.1

Cruise Control

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Kafka cluster load balancing using Cruise Control.....4
How Cruise Control rebalancing works.....5
How Cruise Control retrieves metrics.....6
How Cruise Control self-healing works..... 6

Kafka cluster load balancing using Cruise Control

You can use Cruise Control as a load balancing component in large Kafka installations to automatically balance the partitions based on specific conditions for your deployment. The elements in the Cruise Control architecture are responsible for different parts of the rebalancing process that uses Kafka metrics and optimization goals.

The following illustration shows the architecture of Cruise Control.

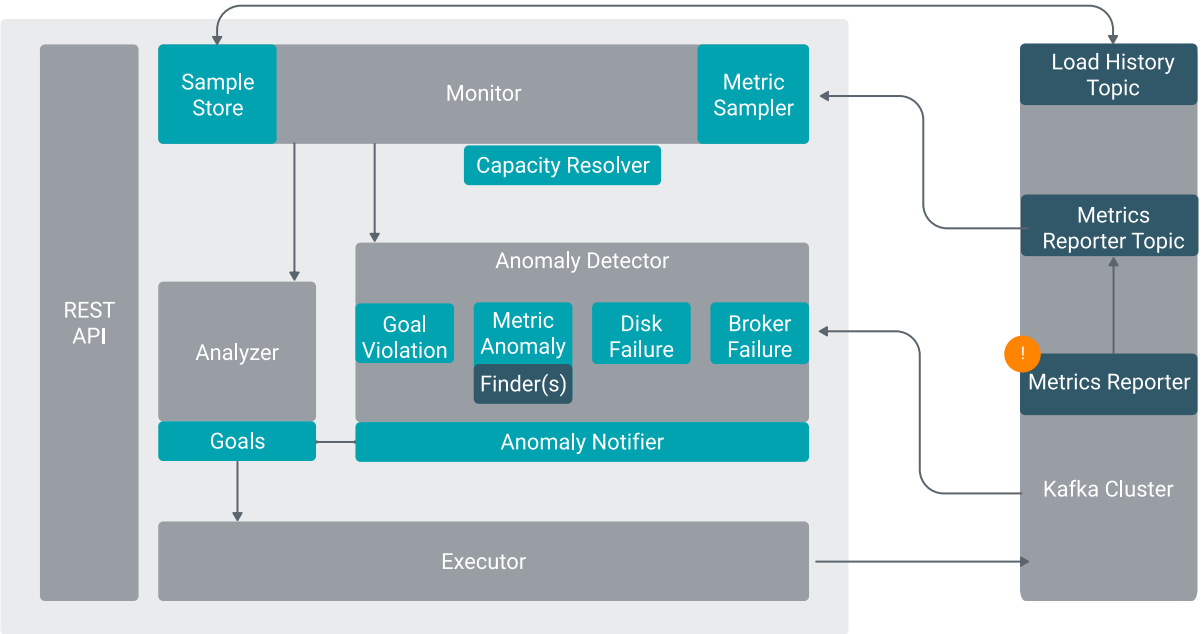

Load Monitor

Generates a cluster workload model based on standard Kafka metrics and resource metrics to utilize disk, CPU, bytes-in rate, and bytes-out rate. Feeds the cluster model into Anomaly Detector and Analyzer.

Analyzer

Generates optimization proposals based on optimization goals provided by the user, and cluster workload model from Load Monitor. Hard goals and soft goals can be set. Hard goals must be fulfilled, while soft goals can be left unfulfilled if hard goals are reached. The optimization fails if the hard goal is violated by optimization results.

Anomaly Detector

Responsible for detecting anomalies that can happen during the rebalancing process. The following anomaly detections are supported in Cruise Control:

- Broker failure
- Goal violations
- Disk failure
- Slow broker as Metric Anomaly
- Topic replication factor

The detected anomalies can be resolved by the self-healing feature of Cruise Control. For more information, see the *How Cruise Control self-healing works* documentation.

Executor

Carries out the optimization proposals and it can be safely interrupted when executing proposals. The executions are always resource-aware processes.

Related Information

How Cruise Control self-healing works

How Cruise Control rebalancing works

For the rebalancing process, Cruise Control creates a workload model based on the resources of the cluster, such as CPU, disk and network load, also known as capacity estimates. This workload model will be the foundation of the optimization proposal to rebalance the Kafka cluster, which can be further customized for your requirements using optimization goals.

During the rebalancing operation, the Kafka brokers are checked to determine if they met the requirements set by the selected goals. If the goals are fulfilled, the rebalancing process is not triggered. If a goal is not fulfilled, the rebalancing process is triggered, and partitions, replicas and topics are reassigned between the brokers until the requirements set by the goal are met. The rebalancing fails in case the hard goals are not met at the end of the reassignment process.

The following goals can be defined for the rebalancing process:

Hard goals

- List of goals that any optimization proposal must fulfill. If the selected hard goals are not met, the rebalancing process fails.

Default goals

- Default goals are used to pre-compute optimization proposals that can be applied regardless of any anomalies. These default goal settings on a healthy cluster can optimize resource utilization if self-healing goals are not specified.
- The value of the Default goals must be a subset of the Supported goals, and a superset of the Hard and Self-healing goals.
- If there are no goals specified as query parameters, the Default goals will be used for the rebalancing process.

Supported goals

- List of supported goals to assist the optimized rebalancing process.

When the hard, default and supported goals are fulfilled, the rebalancing is successful. If there are any goal violations, self-healing can be used.

Self-healing goals

- List of goals to be used for self-healing relevant anomalies. If there are no self-healing goals defined, the Default goals are used for self-healing.
- If the rebalancing process is triggered by self-healing and the Self-healing goals list is not empty, then the Self-healing goals will be used to create the optimization proposal instead of the Default goals.
- The value of the Self-healing goals must be a subset of the Supported goals and a superset of the Hard goals.

Anomaly detection goals

- List of goals that the Anomaly detector should detect if violated. It must be a subset of the self-healing goals and thus also of Default goals.
- The value of the Anomaly detection goals must be a subset of the Self-healing goals. Otherwise, the self-healing process is not able to resolve the goal violation anomaly.

Cruise Control has an anomaly detection feature where goal violations can also be set. The Anomaly detection goals configuration defines when the goals are not met, thus causing a violation. These anomalies can be fixed by the proposal generated from the Self-healing goal configuration. In case there is no self-healing goal specified, Cruise Control uses the Default goals setting. Hard goals can also be set to guarantee the fulfillment of any optimization or self-healing process.

How Cruise Control retrieves metrics

Cruise Control creates metric samples using the retrieved raw metrics from Kafka. The metric samples are used to set up the cluster workload model for the Load Monitor.

In Load Monitor, the Metric Fetcher Manager is responsible for coordinating all the sampling tasks: the Metric Sampling Task, the Bootstrap Task, and the Linear Model Training Task.

Each sampling task is carried out by a configured number of Metric Fetcher threads. Each Metric Fetcher thread uses a pluggable Metric Sampler to fetch samples. Each Metric Fetcher is assigned with a few partitions in the cluster to get the samples. The metric samples are organized by the Metric Sample Aggregator that puts each metric sample into a workload snapshot according to the timestamp of a metric sample.

The cluster workload model is the primary output of the Load Monitor. The cluster workload model reflects the current replica assignment of the cluster and provides interfaces to move partitions or replicas. These interfaces are used by the Analyzer to generate optimization solutions.

The Sample Store stores the metric and training samples for future use.

With the metric sampler, you can deploy Cruise Control to various environments and work with the existing metric system.

Cruise Control Metrics Reporter produces raw metrics directly to a Kafka topic by CruiseControlMetricsReporter. Then these metrics are fetched by Cruise Control and metric samples are created and stored back to Kafka. The samples are used as the building blocks of cluster models.

How Cruise Control self-healing works

The Anomaly detector is responsible for the self-healing feature of Cruise Control. When self-healing is enabled in Cruise Control, the detected anomalies can trigger the attempt to automatically fix certain types of failure such as broker failure, disk failure, goal violations and other anomalies.

Broker failures

The anomaly is detected when a non-empty broker crashes or when a broker is removed from a cluster. This results in offline replicas and under-replicated partitions.

When the broker failure is detected, Cruise Control receives an internal notification. If self-healing for this anomaly type is enabled, Cruise Control will trigger an operation to move all the offline replicas to other healthy brokers in the cluster. Because brokers can be removed in general during cluster management, the anomaly detector provides a configurable period of time before the notifier is triggered and the self-healing process starts.

If a broker disappears from a cluster at a timestamp specified as T, the detector will start a countdown. If the broker did not rejoin the cluster within the broker.failure.alert.threshold.ms since the specified T, the broker is defined as dead and an alert is triggered. Within the defined time of broker.failure.self.healing.threshold.ms, the self-healing process is activated and the failed broker is decommissioned.

Goal violations

The anomaly is detected when an optimization goal is violated.

When an optimization goal is violated, Cruise Control receives an internal notification. If self-healing for this anomaly type is enabled, Cruise Control will proactively attempt to address the goal violation by automatically analyzing the workload, and executing optimization proposals.

You need to configure the anomaly.detection.goals property if you enable self-healing for goal violations to choose which type of goal violations should be considered. By default, the following goals are set for anomaly.detection.goals:

- com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareGoal
- com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaCapacityGoal
- com.linkedin.kafka.cruisecontrol.analyzer.goals.DiskCapacityGoal

Disk failure

The anomaly is detected when one of the non-empty disks dies.

Note: This is the only anomaly that is related to Kafka broker running on a JBOD disk.

When a disk fails, Cruise Control will send out a notification. If self-healing for this anomaly type is enabled, Cruise Control will trigger an operation to move all the offline replicas by replicating alive replicas to other healthy brokers in the cluster.

Slow broker

The anomaly is detected when based on the configured thresholds, a broker is identified as a slow broker.

Within the Metric anomaly, you can set the threshold to identify slow brokers. In case a broker is identified as slow, the broker will be decommissioned or demoted based on the configuration you set for the remove.slow.broker. The remove.slow.broker configuration can be set to true, this means the slow broker will be removed. When setting the remove.slow.broker configuration to false, the slow broker will be demoted.

For more information about configuring the slow broker finder, see the official Cruise Control documentation.

Topic replication factor

The anomaly is detected when a topic partition does not have the desired replication factor.

The topic partition will be reconfigured to match the desired replication factor.

Cloudera Runtime 7.3.1

Securing Cruise Control

Date published: 2019-08-22
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enable security for Cruise Control..... 4

Enable security for Cruise Control

When AutoTLS is disabled, you need to configure the security properties in Cloudera Manager to use Cruise Control in a secure environment. You can also choose between SPENGO and Trusted Proxy as an authentication method, and can assign admin, user and viewer roles to users to achieve further authorization over Cruise Control tasks.

About this task

You can use TLS/SSL security protocols for securing Cruise Control. You must set the TLS/SSL security protocol in Cruise Control just as it is set for Kafka. When TLS/SSL security is enabled, the secure connection is automatically set for Zookeeper as well for secure communication and the non-secure ports are cannot be used.

There are two authentication methods for Cruise Control: SPENGO and Trusted Proxy. SPENGO uses Kerberos over HTTP. Trusted Proxy uses Knox through a gateway mechanism where Knox authenticates with Cruise Control over SPENGO and forwards the real user ID.

Before you begin

Ensure that you have set up TLS for Cloudera Manager:

- Generate TLS certificates
- Configure TLS for Admin Console and Agents
- Enable server certificate verification on Agents
- Configure agent certificate authentication

Procedure

1. Access Cloudera Manager for the Cruise Control configurations.
   a) Navigate to Cloudera Management Console Environments , and select the environment where you have created your cluster.
   b) Select the Streams Messaging cluster from the list of Cloudera Data Hub clusters.
   c) Select Cloudera Manager from the services.
   d) Select Clusters Cruise Control .
   e) Click Configuration.
2. Select Category > Main.
3. Edit the authorization and Kafka security properties according to the cluster configuration.

   You need the following authorization and security properties from the Main category.

Security property		Description
Kafka Client Security Protocol	security.protocol	Protocol to be used for communicating with Kafka. Select the same protocol as you use for Kafka.
Authentication Method	auth_method	Authentication method that Cruise Control uses to authenticate clients.
Trusted Proxy Authentication Service	trusted_auth_service_user	The username part of the trusted proxy authentication service's principal. The default service is Knox for Cruise Control.
ADMIN Level Users	auth_admins	The list of ADMIN level users to have access to all endpoints.
USER Level Users	auth_users	The list of USER level users to have access to all the GET endpoints except bootstrap and train.

Security property		Description
VIEWER Level Users	auth_viewers	The list of VIEWER level users to have access to the most lightweight kafka_cl uster_state, user_tasks and review_board endpoints.

4. Click Save Changes.
5. Click Clear next to the Category Filter.
6. Select Category > Security.
   All the security related properties are displayed.
7. Edit the security properties according to the cluster configuration.

Security property		Description
Enable TLS/SSL for Cruise Control Server	webserver.ssl.enable	Encrypting communication between clients and Cruise Control Server using TLS.
Cruise Control Server TLS/SSL Server Keystore File Location	webserver.ssl.keystore.location	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when Cruise Control Server is acting as a TLS/SSL server.
Cruise Control Server TLS/SSL Server Keystore File Password	webserver.ssl.keystore.password	The password for the Cruise Control Server keystore file.
Cruise Control Server TLS/SSL Server Keystore Key Password	webserver.ssl.key.password	The password that protects the private key contained in the keystore used when Cruise Control Server is acting as a TLS/SSL server.
HTTP Strict Transport Security Enabled	webserver.ssl.sts.enabled	Enables the Strict Transport Security header in the web server responses. By default the configuration is enabled.
Cruise Control Server TLS/SSL Client Trust Store File	ssl.truststore.location	The location on disk of the trust store, used to confirm the authenticity of TLS/SSL servers that Cruise Control Server might connect to. This is used when Cruise Control Server is the client in a TLS/SSL connection. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
Cruise Control Server TLS/SSL Client Trust Store Password	ssl.truststore.password	The password for the Cruise Control Server TLS/SSL Certificate Trust Store File. This password is not required to access the trust store, the field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.

8. Click Save Changes.
9. Click Clear next to the Category Filter.
10. Type ssl.properties to the Search field.
    The Cruise Control Server Advanced Configuration Snippet (Safety Valve) for ssl.properties field is displayed. You can define additional confiuration parameters for Cruise Control in the Safety Valve field. You also need to

add properties that are inherited from Kafka clients, if needed for the secured communication. The values for the inherited properties should match with the values defined in Kafka.

a) Add the webserver keystore type, if needed, by adding the following property and its value to the Advanced Configuration Snippetfield.

Security property	Description
webserver.ssl.keystore.type	The file format of the key store file. This configuration is optional. If the keystore type for the webserver is not set, it falls back to Jetty's default behavior.

b) Add the following additional security properties to the Advanced Configuration Snippet.

Security property	Description
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.
ssl.cipher.suites	A cipher suite is a named combination of authentication, encryption, MAC, and a key exchange algorithm used to negotiate the security settings for a network connection using TLS or SSL network protocol.
ssl.enabled.protocols	This property should list at least one of the protocols (TLSv1.2, TLSv1.1,TLSv1) configured on the broker side.
ssl.truststore.type	The file format of the trust store file.
ssl.keystore.type	The file format of the key store file.

11. Adding the following properties to the Advanced Configuration Snippet helps customizing the default TLS configurations defined for the Cruise Control web server:

Security property	Description
webserver.ssl.include.ciphers	Sets the included ciphers for the webserver.
webserver.ssl.exclude.ciphers	Sets the excluded ciphers for the webserver.
webserver.ssl.include.protocols	Sets the included protocols for the webserver.
webserver.ssl.exclude.protocols	Sets the excluded protocols for the webserver.

12. Click Save Changes.

Cloudera Runtime 7.3.1

Atlas Use Case: Controlling Data Access Using Tags

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Atlas classifications drive Ranger policies **4**

When to use Atlas classifications for access control **5**

How tag-based access control works **5**

Propagation of tags as deferred actions 6

Examples of controlling data access using classifications **7**

Atlas classifications drive Ranger policies

Ranger policies can use tags to identify data; Atlas classifications are pulled into Ranger as tags.

You can use Atlas classifications to drive data access control through Ranger. Ranger offers both resource-based and tag-based access control policies. Using metadata tags rather than specific resource names gives you flexibility and allows access controls to apply immediately to new data assets without requiring administrator intervention.

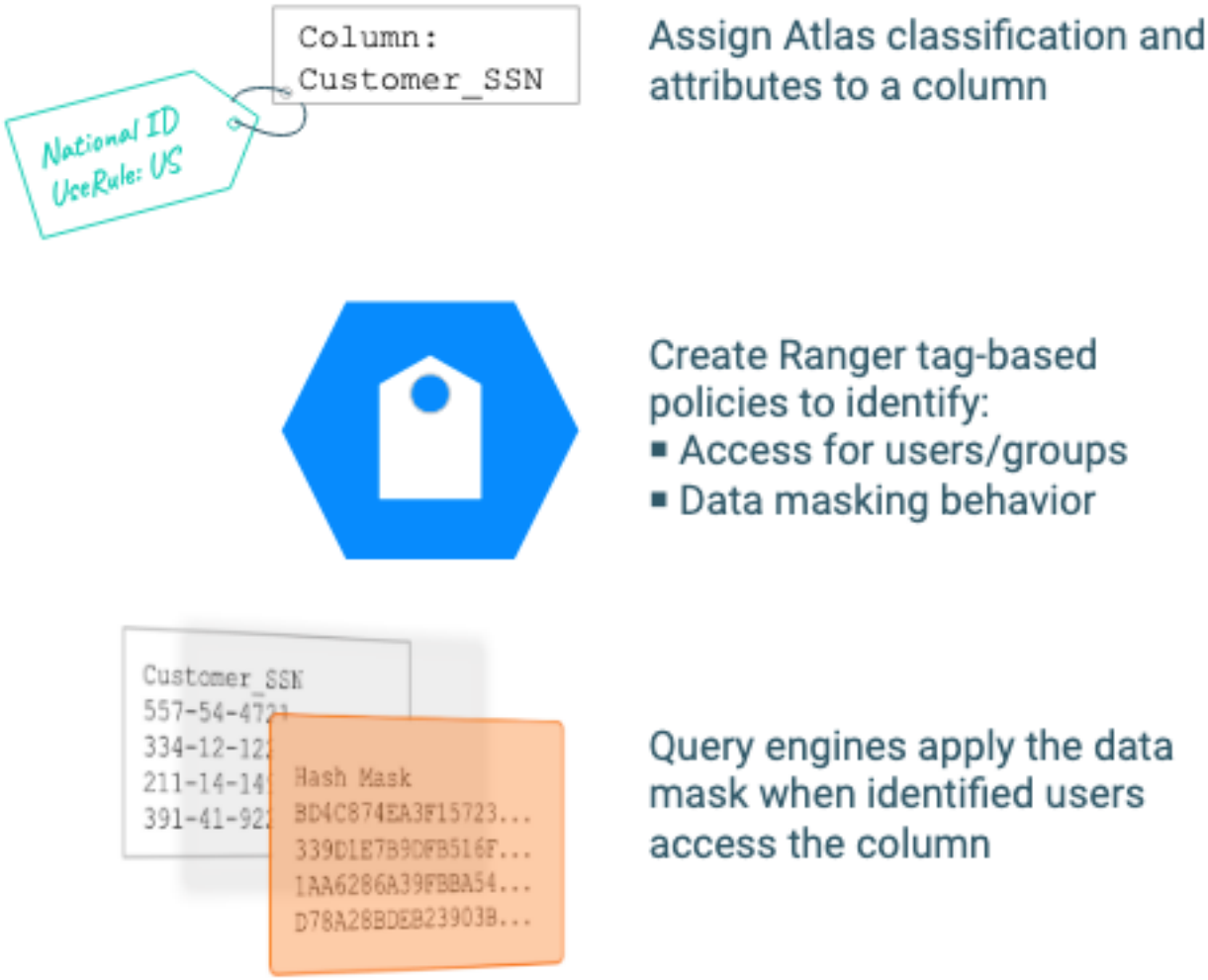

You can use Atlas classifications to control user access to data assets by using Atlas classifications to define Ranger tag-based access-control policies. Ranger tag-based policies ensure that services such as Hive and Impala control user and group access to specific data assets. Ranger policies can have services apply masks to column data, so users see results such as partial data or anonymized data. To make this work, you define classifications in Atlas; associate the classifications with data assets, including databases, tables, views, and columns; then define policies that act on the data assets tagged with the Atlas classifications.

Some of the ways you can use classifications include:

- Add attributes to Atlas classifications to define separate behaviors for separate contexts. For example, you could mark columns as “National ID” and apply a policy based on that information. You could add an attribute to the “National ID” classification that describes which the rules to apply to the display of the national ID, such as “Apply Rules From”:“EU” or “Apply Rules From”:“JPN”). Ranger policies can use the attribute value to apply different mask patterns to the data.

- Atlas lineage can propagate classifications from one column to columns created later from the same data. When classifications are propagated, the Ranger policies built on these classifications apply to the new location of the data. No intervention is required to ensure that the access controls on the original data are applied to the new copy.
- If Ranger is set up to deny access to new data except for the owner, you can use tags to reveal this data (access only to classified data).

Group level policy changes for Atlas

Sometimes, group level policy does not function normally due to multiple gid for a group if SSSD is configured by the end user.

The group level policy occurence could be because of an intermittent environmental issue. This behaviour can be addressed by adding hadoop native lib to Atlas process: - Djava.library.path=/opt/cloudera/parcels/CDH/lib/hadoop/lib/native

For more information, see Hadoop Groups Mapping.

Related Information

When to use Atlas classifications for access control

How tag-based access control works

Examples of controlling data access using classifications

When to use Atlas classifications for access control

Resource-based and tag-based policies are useful in different ways.

Ranger provides resource-based policies and tag-based policies. The following table provides some examples of when you would choose one type of policy over the other:

Resource-based Policies	Tag-based Policies
Control access to data assets per service type (multiple policies for each data asset)	Control access to data assets across all service types
Control access to entire databases	Control access to columns in source tables that users can copy or transform to other tables
Control access to long-lived tables	Control access to data until it is reviewed/classified by setting an validity date
Control access to well-known columns in specific tables, which don't change over time	

Related Information

Ranger tag-based policies

Resource-based services and policies

How tag-based access control works

Make the necessary preparations in Atlas to make tags available for creating Ranger policies.

Follow these steps to set up tag-based access control in your environment:

1. Determine what data to control, who it's controlled for, and how you want to control it.

   If you know the data characteristics but there isn't a reliable column name for the data or if you want to reveal parts of the data to some users, assign a classification to the column and set a tag-based policy in Ranger to apply masks to the data.

   - Same resource across multiple services. Set tag-based policies in Ranger. Note that resource-based policies apply to a single service.
   - Entire databases. Set resource-based policies in Ranger.
   - Tables. Set resource-based policies in Ranger.
   - Columns. Tagging columns in Atlas and then creating tag-based policies in Ranger allows you to control access to this data even as it is transformed into other tables.

   Note: Ranger tag-based policies cannot be applied to Hive views.

2. Create classifications in Atlas that describe triggers for when data should be controlled.
3. Assign the classifications to the Atlas data assets.
4. Create "tag based policies" in Ranger.
5. Use Hue or Zeppelin to validate that the policies work as expected.

Related Information

Ranger tag-based policies

Resource-based services and policies

When to use Atlas classifications for access control

Examples of controlling data access using classifications

Propagation of tags as deferred actions

By default, Atlas asynchronously propagates tags along a lineage while processing other tasks. Depending on your scenario, you can disable the propagation of tags as deferred action. This might help you to save compute resources and time for other Atlas processes at the cost that the tag propagation finishes later.

To control tag propagation, you can set the value of the atlas.tasks.enabled property. By default, this is set to true. You can set this property in Cloudera Manager under Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties.

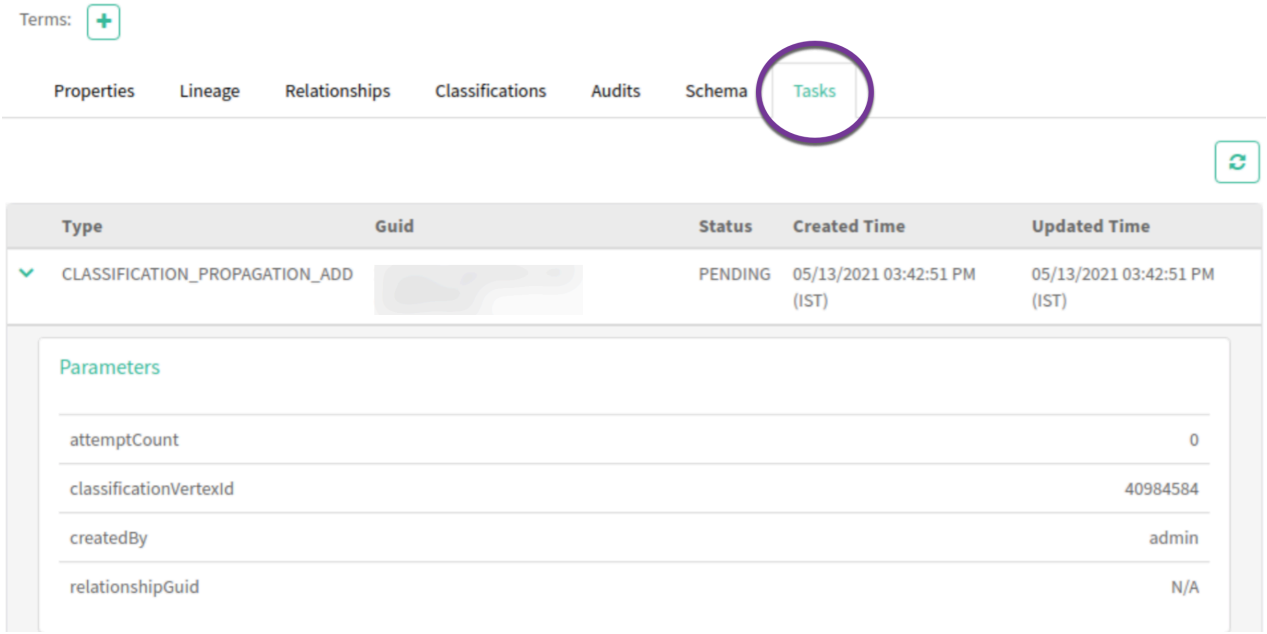

While the background task is processing the tag propagation request, you can view the notification of propagation of tags in the Atlas UI on the Entities details page. The Task tab displays the progress of this request.

Examples of controlling data access using classifications

Some of the ways you can use classifications to control access to data.

Use classifications to control data:

- Validity period or expiration date.
- Sensitive data masking
- Segregating access privileges by department or region

Related Information

Adding a tag-based PII policy

Default EXPIRES ON tag policy

Dynamic tag-based column masking in Hive with Ranger policies

Cloudera Runtime 7.3.1

Atlas Use Case: Extending Metadata

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Top-down process for adding a new metadata source.....4

Top-down process for adding a new metadata source

You can create entity types in Atlas to represent data assets, operations, or other types of artifacts from sources other than those supported by default.

Atlas' data model is designed to be flexible enough to represent a wide variety of data assets and the processes that generate them. You can design your own entities to collect metadata from sources beyond what comes predefined in Atlas. The high-level process for creating entities is as follows:

1	Think about how you want to model the core assets from your source. The predefined entity types inherit basic attributes from the DataSet and Process entity types. Using these general types as your starting point ensures that your entities can display lineage in the Atlas Dashboard and take advantage of other predefined rules and hierarchies built into the models. That said, new entity types are not limited to this data/process model.
2	Define the relationships you want to track among the data assets. To have your entities appear in lineage graphs in the Atlas Dashboard, include relationship attributes for "inputToProcesses" and "outputFromProcesses".
3	List all the metadata you want to track for each entity type. If necessary or convenient, define enumerations or structures to describe metadata specifically for your entities.
4	Use REST API or create an Atlas client to build the entity type definitions in Atlas. (build JSON)
5	Validate the model by manually filling in metadata in the types to validate that they behave the way you expect. You can do this using the Atlas Dashboard or REST API and then test that lineage, search, and Ranger tag-based-policies do what you expect.
6	Write a hook, bridge, or both to collect metadata automatically from your source.

Cloudera Runtime 7.3.1

Atlas Use Cases

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using metadata for cluster governance.....4

Data Stewardship with Apache Atlas.....5

Apache Atlas dashboard tour.....6

Apache Atlas metadata collection overview.....9

Atlas metadata model overview..... 10

Using metadata for cluster governance

Concepts for collecting, creating, and using metadata.

What is Apache Atlas?

Atlas is a metadata management and governance system designed to help you find, organize, and manage data assets. Atlas creates “entities” or metadata representations of objects and operations in your data lake. You can add business metadata to these entities so you can use business vocabulary to make it easier to search for specific assets.

Apache Atlas uses metadata to create lineage relationships

Atlas reads the content of the metadata it collects to build relationships among data assets. When Atlas receives query information, it notes the input and output of the query and generates a lineage map that traces how data is used and transformed over time. This visualization of data transformations allows governance teams to quickly identify the source of data and to understand the impact of data and schema changes.

Adding to entity metadata makes searching easier

Atlas manages *classifications* and *labels* that you create and use to enhance the metadata for data assets. You can create and organize classifications and labels to use for anything from identifying data cleansing stages to recording user comments and insights on specific data assets. When you use classifications, the Atlas Dashboard makes it easy to search, group, report, and further annotate the entities you label. Classifications themselves can be organized into hierarchies to make them easier to manage.

Atlas also provides an infrastructure to create and maintain business ontologies to label your data assets. Atlas’ “glossaries” include “terms” so you can build agreed-upon lists for department- or organization-wide vocabulary to identify and manage data. Adding a term gives you a single-click report of entities identified by that term.

Apache Atlas architecture

Atlas runs as an independent service in a Hadoop environment. Many Hadoop data processing and storage services include Atlas add-ons that publish metadata for the services’ activities to a Kafka message topic. Atlas reads the messages and stores them in JanusGraph to model the relationships among entities. The datastore behind JanusGraph is HBase. Atlas stores a search index in Solr to take advantage of Solr’s search functionality.

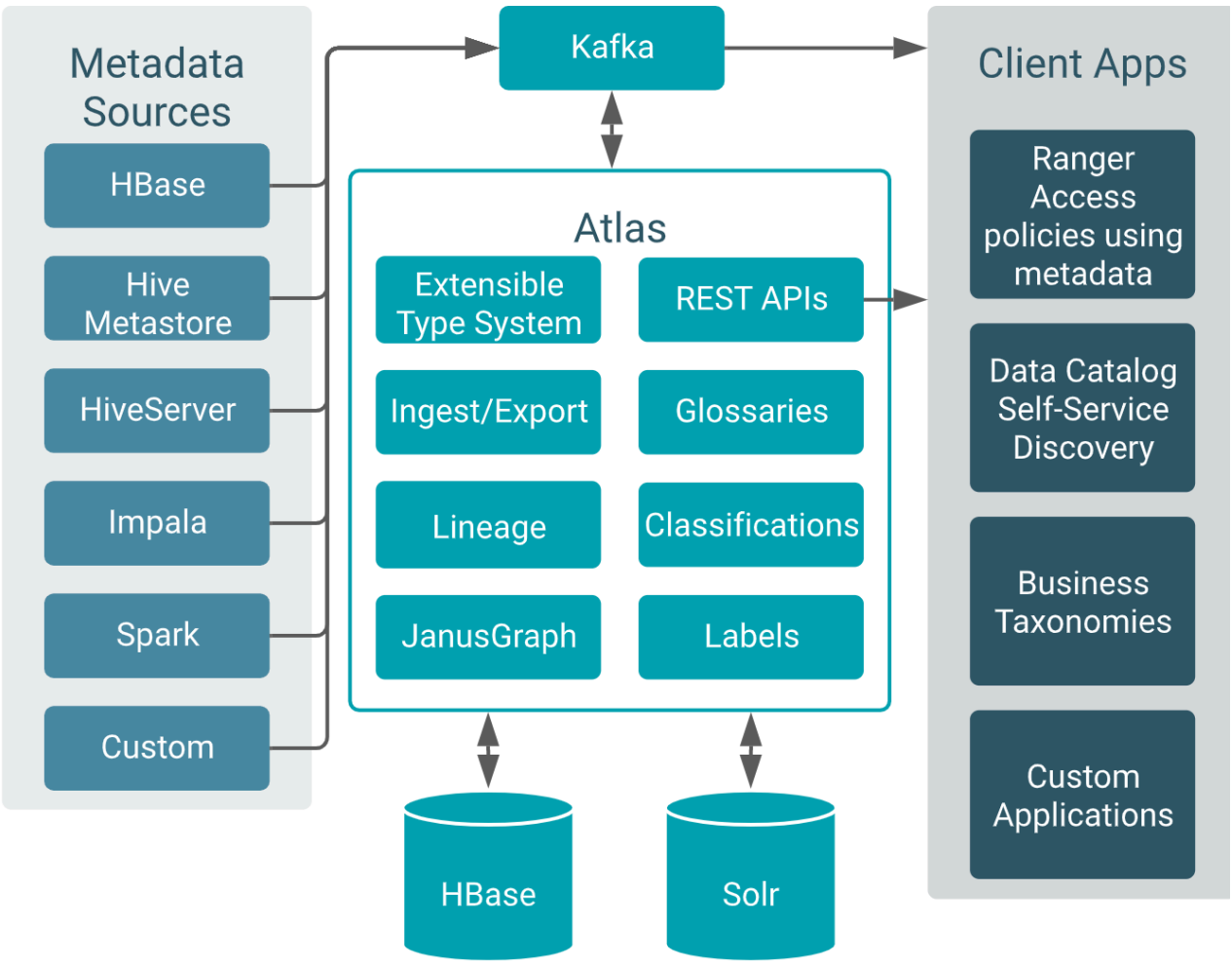

Pre-defined hooks exist for Hive, Impala, Kafka, NiFi, Spark, and Sqoop.

Atlas also provides “bridges” that import metadata for all of the existing data assets in a given source. For example, if you start Atlas after you’ve already created databases and tables in Hive, you can import metadata for the existing data assets using the Hive bridge. Bridges use the Atlas API to import the metadata rather than publishing messages to Kafka.

If you need a hook or bridge to automate collecting metadata from another source, use the Atlas Java API to create a custom Atlas addon.

Note: Governance through Apache Atlas is just one element of a secure production cluster: Cloudera supports Atlas only when it runs on a cluster where Kerberos is enabled to authenticate users.

Data Stewardship with Apache Atlas

Concepts for collecting, creating, and using metadata.

The value of Atlas metadata for organizing and finding data increases when you augment the generated “technical” metadata by using your organization's business vocabulary. Here’s how you would do that:

- Tools. You can add metadata to entities using labels, classifications, attributes, and a glossary of terms. The glossary allows you to identify synonyms so that the vocabulary from different teams does not get in the way of identifying the same data.

- Planning. Figure out who and how to apply the tools; set up an overall plan for what kinds of metadata you want to apply, design some conventions for how to apply them and who can apply them. Design some processes to oversee metadata as it collects to make sure the results are consistent and useful; identify synonyms and antonyms.
- Examples. This document includes examples of ways you can organize your metadata; strategies that describe how to optimize for specific use cases.

Apache Atlas dashboard tour

Quick introduction to the Atlas user interface and terms.

The Atlas user interface is a metadata dashboard with two parts:

- Search panel on the left.
- Detail panel on the right where search results appear, and, when you drill into a specific entity, the details for that entity are displayed. Each detail page has a header section and a series of tabbed panels, all of which are oriented to the metadata for that entity type.

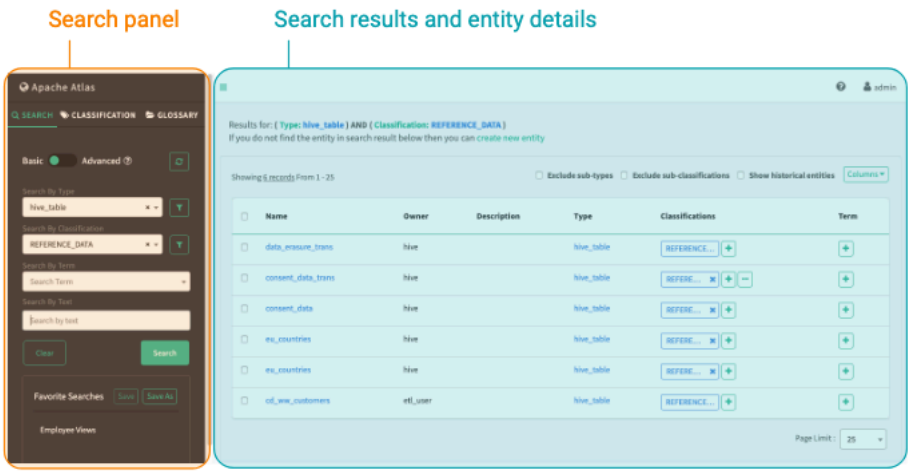

Searching

The search panel has three tabs for searching: the general Search tab, and predefined searches based on Classifications and Glossary terms. In the general Search tab, you choose from existing lists of metadata types to narrow the search results. Switching to the Advanced search lets you enter specific search queries; both basic and advanced searches can be saved for easy reuse.

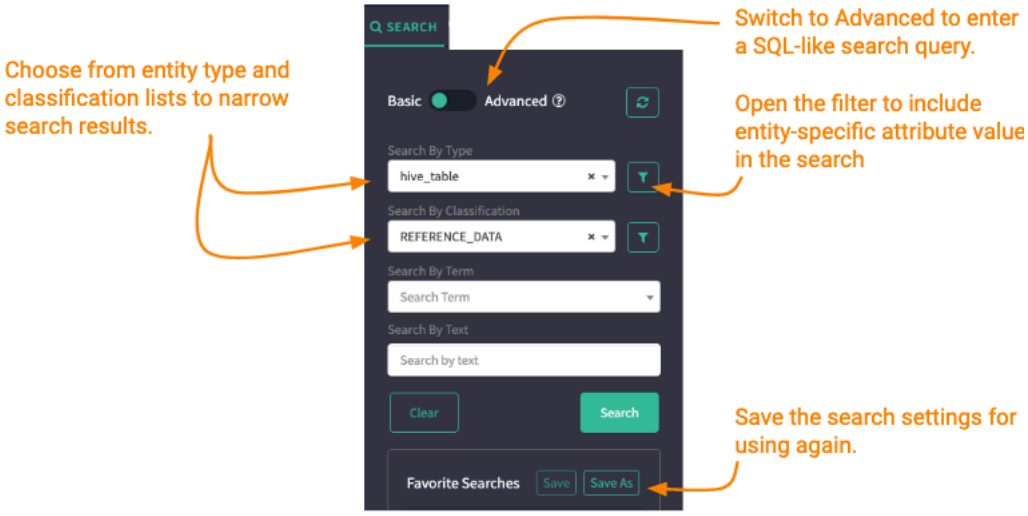

In the Classification tab, selecting a classification displays all the entities that are marked with that classification. Find a specific classification using the search box or browse through the classification hierarchy that you define when you create classifications.

In the Glossary tab, selecting a term displays all the entities that are marked with that term. Find a specific term using the search box or browse through terms by glossary. You can also find specific terms using the category view: browse through the hierarchy you build of your organization's business glossary; when you select a category, the display pane shows the terms assigned to the category. When you select one of those terms, Atlas displays the entities associated with that term.

When you run a search and Atlas returns results, you see a paged-list of entities that match the search criteria. From here, you can go back to the search options and further refine your search or use controls to change how the search results are presented.

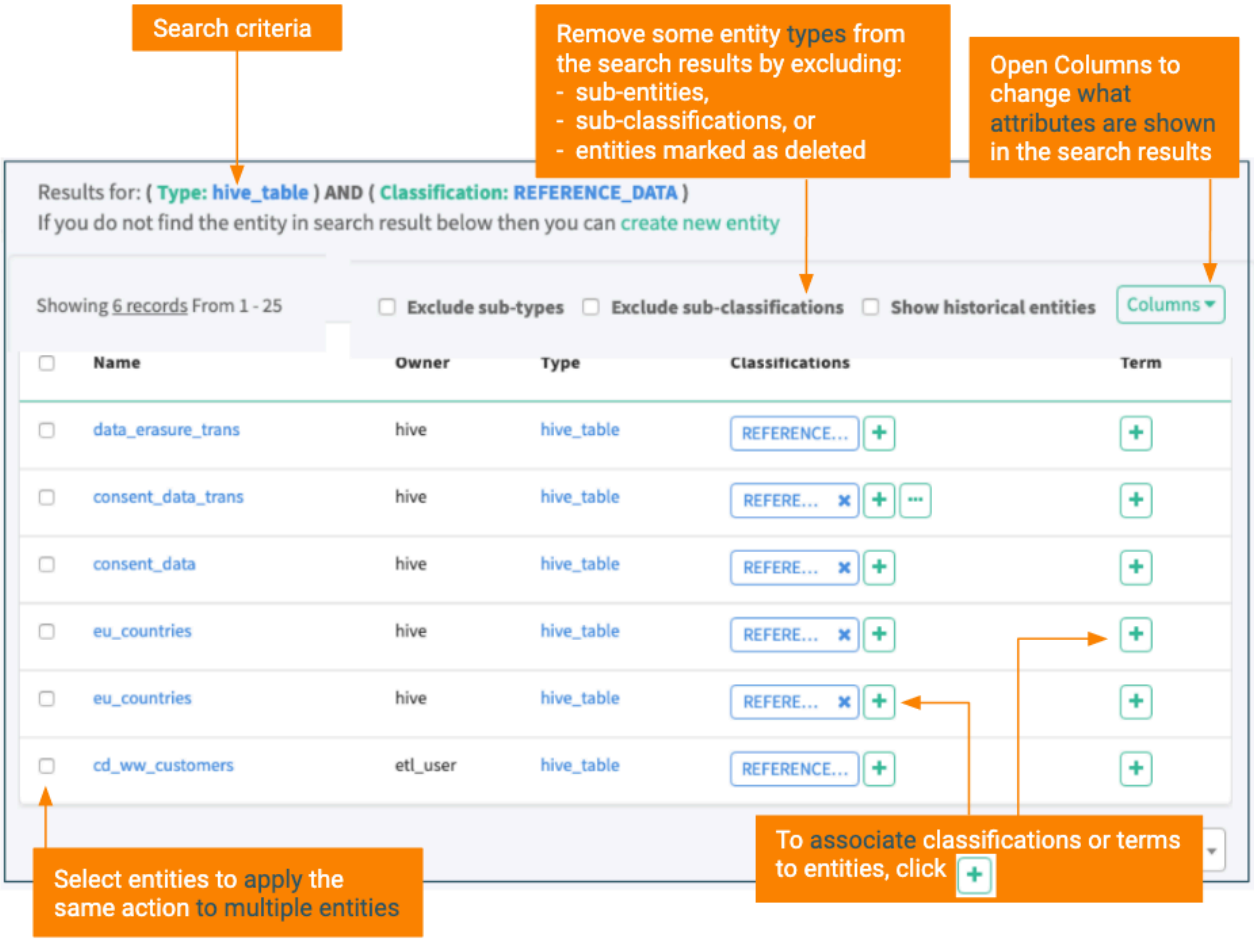

Viewing entity details

When you click a link for an entity in the search results, Atlas opens an entity detail page that includes the metadata collected for the entity. The detail page organizes the entity content in tabs:

Properties

The Properties tab includes the system metadata collected for this entity and any user-defined properties added. It also contains a list of labels applied to the entity. You can use values from any of the "string" data type properties to find this entity using free-text search.

Lineage

Atlas UI displays a lineage graph for each entity. The graph appears in the Lineage tab in the entity details page. For data asset entities, the lineage graph shows when the entity was an input or an output from an operation. For process entities, the lineage graph shows all input and output entities used by or produced by the operation.

Relationships

This tab lists the other entities that are associated with this entity as "relationships". You can show the list of related entities as a list or as a graph. Use this tab to navigate among entities. The special relationship types "input" and "output" include the entities that make up lineage.

Classifications

The Classifications tab shows the classifications associated with this entity (which are also shown in the top section of the detail page). It allows you to add, update, or remove classifications from the entity. Note that to manage classifications, you need to be granted privileges to perform classification actions.

Audits

Atlas records the changes that occur to entity metadata. The changes are listed in the Audit tab in the entity details page. Audits show when Atlas updates the entity's metadata, including the following changes:

- Classifications added or removed
- Entity attributes updated
- Labels added, updated, or removed
- Relationships added, updated, or removed
- Glossary terms added or removed

Schema

When the current entity is a table, the Schema tab appears and lists columns in the table. Use this tab to drill into a specific column or to add classifications to columns (no need to open the detail page for the column to add a classification).

Apache Atlas metadata collection overview

Actions performed in cluster services create metadata in Atlas.

Atlas provides addons to many Hadoop cluster services to collect metadata when the service performs certain operations. The Atlas addon or “hook” assembles a predefined set of information and sends it to the Atlas server. The Atlas server reads through the metadata and creates entities to represent the data sets and processes described by the metadata. Atlas may create one or many entities for each event it processes. For example, when a user creates a namespace in HBase, Atlas creates a single entity to represent the new HBase namespace. When a user runs a query in HiveServer, Atlas may create many entities, including entities to describe the query itself, any tables involved in the query, entities for each column for each table involved in the query, and so on.

The following table lists the services that are integrated with Atlas by default. For each service, the table lists the events produced by the service that Atlas acknowledges and the entities Atlas produces in response to each event. Note that there isn’t always a one-to-one relationship between the event and an entity: the entities produced from a single event depend on the event itself.

Source	Actions Acknowledged	Entities Created/Updated
HiveServer	ALTER DATABASE CREATE DATABASE DROP DATABASE	hive_db, hive_db_ddl
	ALTER TABLE CREATE TABLE CREATE TABLE AS SELECT DROP TABLE	hive_process, hive_process_execution, hive_table, hive_table_ddl, hive_column, hive_column_lineage, hive_storagedesc, hdfs_path
	ALTER VIEW ALTERVIEW_AS_SELECT CREATE VIEW CREATE VIEW AS SELECT DROP VIEW	hive_process, hive_process_execution, hive_table, hive_column, hive_column_lineage, hive_table_ddl

Source	Actions Acknowledged	Entities Created/Updated
	INSERT INTO (SELECT) INSERT OVERWRITE	hive_process, hive_process_execution
HBase	alter_async	hbase_namespace, hbase_table, hbase_column_family
	create_namespace alter_namespace drop_namespace	hbase_namespace
	create table alter table drop table drop_all tables	hbase_table, hbase_column_family
	alter table (create column family) alter table (alter column family) alter table (delete column family)	hbase_table, hbase_column_family
Impala*	CREATETABLE_AS_SELECT	impala_process, impala_process_execution, impala_column_lineage, hive_db hive_table_ddl
	CREATEVIEW	impala_process, impala_process_execution, impala_column_lineage, hive_table_ddl
	ALTERVIEW_AS_SELECT	impala_process, impala_process_execution, impala_column_lineage, hive_table_ddl
	INSERT INTO INSERT OVERWRITE	impala_process, impala_process_execution
Spark*	CREATE TABLE USING CREATE TABLE AS SELECT, CREATE TABLE USING ... AS SELECT	spark_process
	CREATE VIEW AS SELECT,	spark_process
	INSERT INTO (SELECT), LOAD DATA [LOCAL] INPATH	spark_process

*For these sources, Atlas collects the corresponding asset metadata from HMS. Atlas reconciles the entity metadata received from Kafka messages from each source.

Related Information

Hive Server 2 metadata collection

HBase metadata collection

Impala metadata collection

Spark Metadata Collection

Atlas metadata model overview

Atlas' model represents cluster data assets and operations, and is flexible enough to let you represent objects from other sources.

The flexibility Atlas' metadata model lets you represent the objects and relationships among them so you can produce a map of your data lake. Atlas lets you create new instances of predefined entity types and lets you define new types of entities so you can represent data assets and actions from additional data sources or even services that do not reside in Hadoop. Atlas' building blocks are entities, relationships, classifications, enumerations, and structures.

Entities are a collection of attributes that model or represent a data asset or data action. Entities are the unit that Atlas returns in search results or shows as nodes in a lineage diagram. Labels are modeled as attributes on a given entity instance; you can add user-defined properties to individual entity instances (without affecting the entity type definition).

Relationships describe connections between two entities. You can create relationship definitions with custom attributes to represent behaviors that are specific to your processes. Changes to relationship definitions require changing the model through the Atlas API.

Classifications are named sets of key-value pairs that can be associated with entities. Classifications are distinct from entity attributes:

- Classifications are not part of entity metadata so they are a way to add metadata to entities without updating entity type definitions.
- Classifications can be added to any entity type.
- Atlas can propagate classifications through lineage relationships.
- Classifications can be used in Ranger to drive access policies.

Business Metadata are a set of custom attributes that an administrator can use to extend the metadata stored for each entity type. User access to each set of attributes can be managed using Ranger policies.

Atlas supports defining custom enumerations and data structures as well, similar to those constructs in structured programming languages. Enums can be used in attribute definitions to store lists of predetermined values; structs can be used in attribute definitions to identify more complex data types.

Cloudera Runtime 7.3.1

Cloudera Security Overview

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction **4**
Importance of a Secure Cluster 4
Secure by Design 4

Pillars of Security **4**
Authentication 5
Authorization 5
Encryption 5
Identity Management 6

Security Management Model **6**

Security Levels **7**
Choosing the Sufficient Security Level for Your Environment 8
Logical Architecture 8

SDX **9**

Security Terms **10**

Introduction

Securing data is important to every aspect of your business, including your customers and personnel. Before you can confidently and effectively secure your data, you must understand data security fundamentals.

Cloudera Private Cloud, and the components that make up Cloudera, rely on these fundamentals to ensure your data is secure by design.

This document provides a high-level overview of security basics including the importance of securing a cluster, the Four Pillars of Security, and Cloudera Shared Data Experience, as well as a comprehensive glossary of security terms.

Importance of a Secure Cluster

Threats to your cluster can come from a variety of sources, so securing your cluster against all of these sources is important.

External attackers might gain access to your cluster and compromise sensitive data, malicious software might be implemented through known vulnerabilities, and insiders or third party vendors might misuse legitimate permissions if authorization is not appropriately implemented.

Governing bodies have implemented various data protection and privacy laws which you must implement into your cluster's security. For example, the Health Insurance Portability and Accountability Act of 1996 (HIPAA) is a United States federal law that requires protecting sensitive patient health information from being disclosed without the patient's consent or knowledge. The General Data Protection Regulation (GDPR) is a regulation in European Union law aimed at enhancing individuals' control and rights over their personal data both inside and outside the EU.

Additionally, if your business accepts credit card payments, you must comply with the Payment Card Industry Data Security Standard (PCI DSS). This global standard applies a series of security measures to ensure the safety of your customers' credit card information.

It is therefore crucial to have a secure cluster that addresses all potential threats and vulnerabilities, and complies with all data protection and privacy laws. By actively reducing the threat surface on your environment, you actively manage the risk, likelihood, and impact of a security event.

Secure by Design

Our Cloudera is secure by design.

We treat security as the top-tier design requirement to manage risk, reduce attack surface and vulnerabilities, and develop design concepts and patterns in an industry-preferred way.

We have designed Cloudera to meet technical audit requirements out-of-the-box. In fact, many Cloudera services require security components prior to their installation.

Cloudera gives you the ability to manage your cluster securely, allowing you to audit and understand how and what has happened on your cluster, and manage data policy and governance.

Pillars of Security

To ensure the security of your data, Cloudera follows the Pillars of Security: authentication, authorization, encryption, and identity management.

These pillars work together to support our platform.

Authentication

Authentication is a basic security requirement in a digital environment.

Users, processes, and services must verify their identity using credentials known by the authenticator in order to access a system. Credentials can include something you know such as a password or PIN; something you have such as a badge or RSA token; or something you are such as biometrics including fingerprints, facial features, or iris patterns.

The components of Cloudera that provide authentication solutions to your cluster are Apache Knox and Kerberos. Apache Knox provides perimeter security so that the enterprise can extend access to new users while also maintaining compliance with security policies. Kerberos is a network authentication protocol designed to provide strong authentication for client/server applications by using secret-key cryptography.

An iPhone user can unlock their device by using the facial recognition scan or inputting their PIN.

Authorization

Authorization is a basic security requirement in a digital environment.

Authorization controls what users can and cannot access within a system. These permissions include viewing, editing, using, or controlling information. In a physical environment, this can also include accessing servers, rooms, and devices.

Apache Ranger is a component of Cloudera that provides authorization solutions to your cluster. Ranger defines the policies that determine what users can and cannot access.

Note: Authorization through Apache Ranger is just one element of a secure production cluster: Cloudera supports Ranger only when it runs on a cluster where Kerberos is enabled to authenticate users.

The owner of a Google Doc can share their document with others. When sharing, they can set the access permissions of others as either editors, viewers, or commenters to determine their level of interaction with the file.

Encryption

Encryption increases the level of security in a digital environment.

Encryption protects sensitive data through encoding, so the data can only be accessed or decoded with a digital key. Encryption applies to both data at-rest and data in-transit.

Data at-rest, or data that is stored physically on a storage device such as a hard drive or cloud storage and not actively moving, is encrypted and digital keys are distributed to authorized users to decrypt it when needed.

Data en route between its source and its destination is known as data in-transit. End-to-end encryption utilizes protocols like Transparent Layer Security (TLS) to protect data in-transit. Each data transmitting session has a new digital key, which relies on proper authentication and can reduce the risk of unauthorized access.

Cloudera provides the following components for encryption solutions: Ranger KMS and Navigator Encrypt.

Ranger extends the Hadoop KMS functionality by allowing you to store keys in a secure database. Navigator Encrypt transparently encrypts and secures data at rest without requiring changes to your applications.

Figure 1: Encryption model demonstrating encryption at rest and in transit

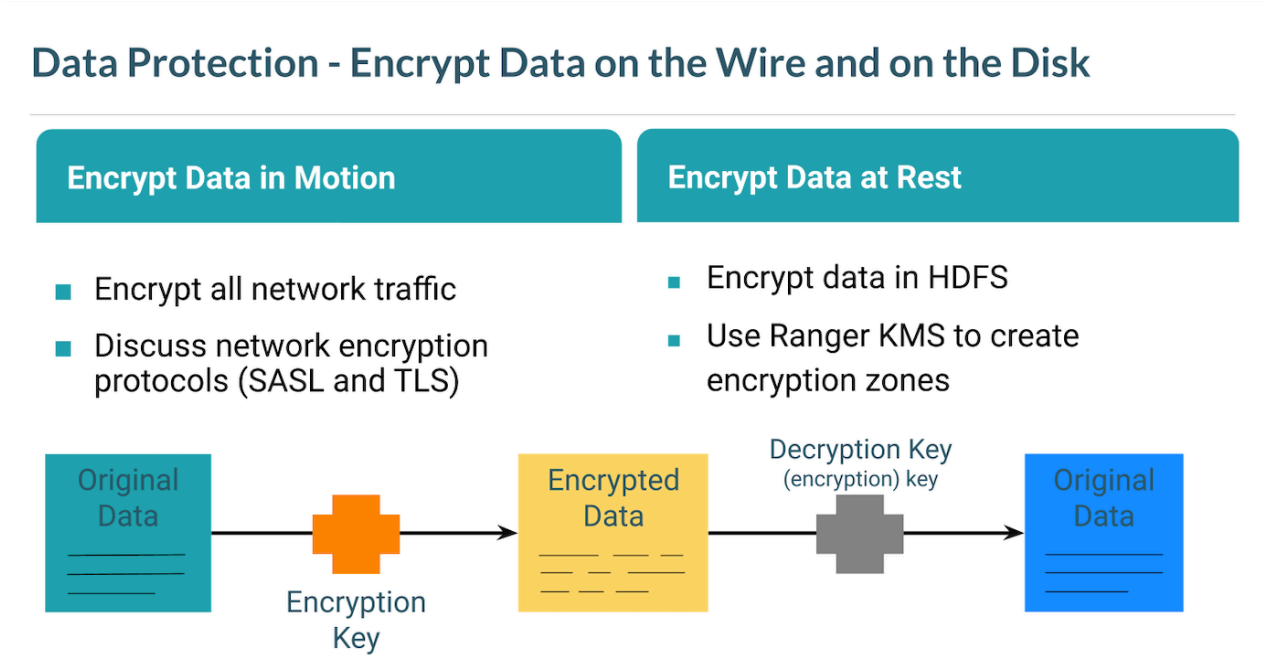

Identity Management

Identity management is how user and group identities are centrally stored.

Identity Management functions in conjunction with authentication and authorization as a form of security that ensures everyone has appropriate access to the resources needed to perform their roles. Managing identities of users and groups involves implementing policies and allowing or denying permissions.

Security Management Model

The Pillars of Security combined with related technical concepts and Cloudera components make up the Security Management Model.

The combination of Cloudera Manager, Ranger, and Atlas allows you to consolidate security configurations in your clusters. The key security principles of our layered approach to data defense are as follows:

Figure 2: Security Management Model

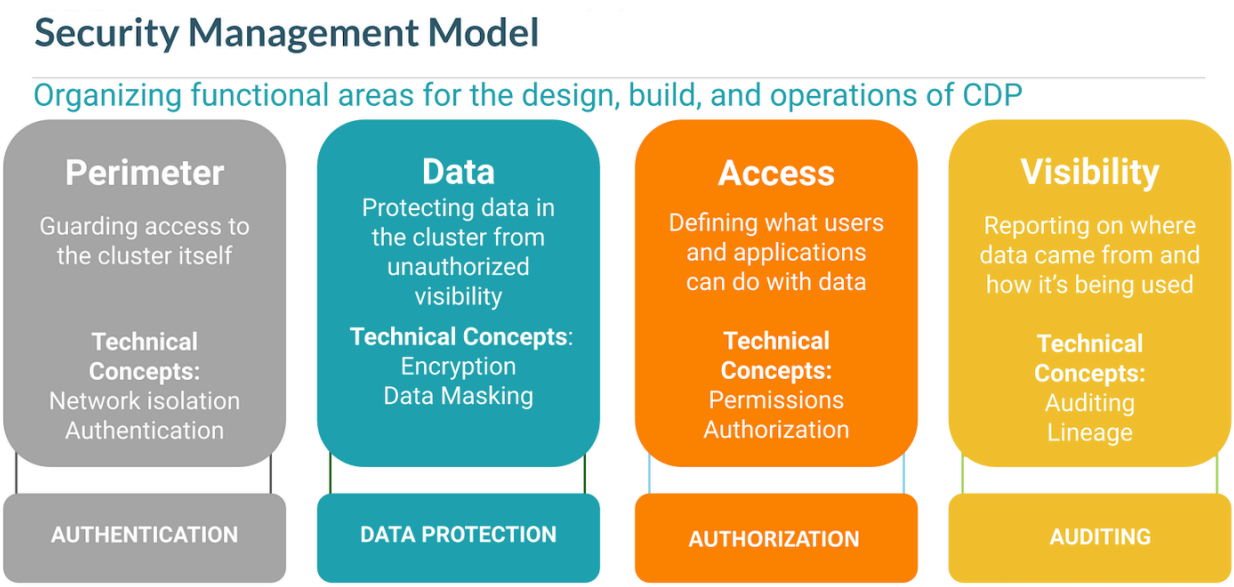

Security Levels

Securing a cluser requires a layered approach, applying the Pillars of Security to ensure data availability, integrity, and confidentiality. Cloudera Private Cloud offers four security levels, as security is no longer optional.

0 - Non-Secure

The cluster has no configured security. Clusters without security must not be used in production environments because they are highly vulnerable to all attacks and exploits. Cloudera does not support running an insecure environment.

1 - Minimally Secure

Minimally secure clusters are configured for authentication, authorization, and auditing. Authentication is first configured to ensure that users and services can only access the cluster after proving their identities. Authorization mechanisms are then applied to assign privileges to users and user groups. Auditing procedures keep track of who accesses the cluster.

2 - More Secure

Clusters that encrypt their sensitive data are more secure. These systems include a variety of steps, including key management, to ensure data is encrypted while in-transit.

Auditing has been set up for data governance to trace any data object's lineage and verify its authenticity. It determines what people are doing with your data, and when they are doing it.

3 - Most Secure

The most secure clusters ensure that all data, both at-rest and in-transit, is encrypted and the key management system is fault-tolerant. Auditing mechanisms comply with industry, government, and regulatory standards such as PCI, HIPAA, GDPR, and NIST, and extend from the cluster to any integrated systems. Cluster administrators are well-trained, the security procedures are certified by an expert, and the cluster passes technical reviews.

Figure 3: Cloudera Security Level Model

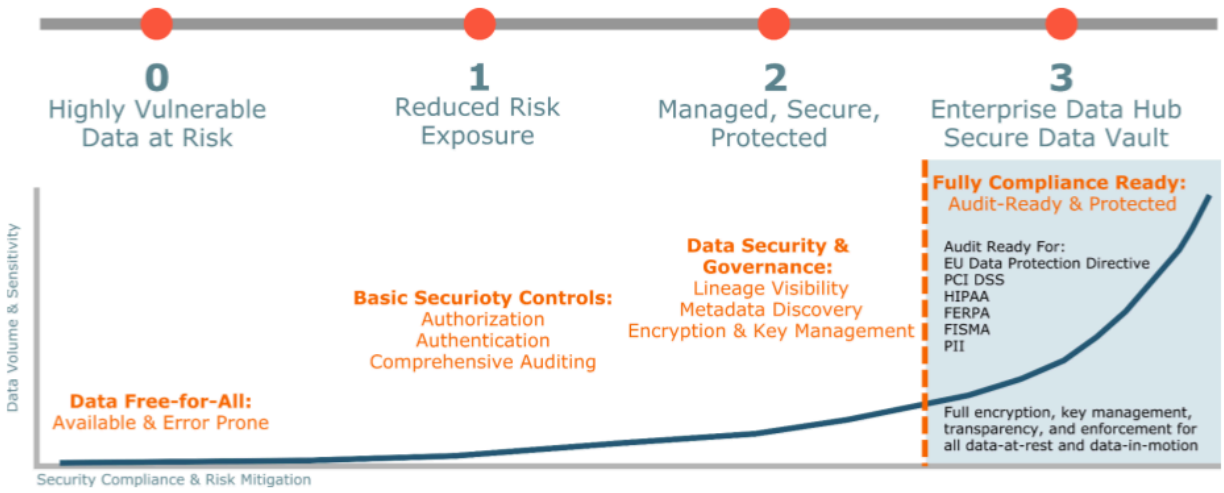

Choosing the Sufficient Security Level for Your Environment

Cloudera will assist you in deciding which security level is best for your environment.

Regardless of the security level each major organization chooses, they must complete a penetration test, also known as a “pen test.” This test simulates a cyberattack through ethical hacking which evaluates the system security, identifying its strengths and weaknesses.

Related Information

Cloudera Security

Logical Architecture

The components that embody the Pillars of Security and make up the Cloudera Private Cloud security architecture are designed to work together logically.

The following diagram visualizes this relationship:

Figure 4: Cloudera Private Cloud Security Architecture

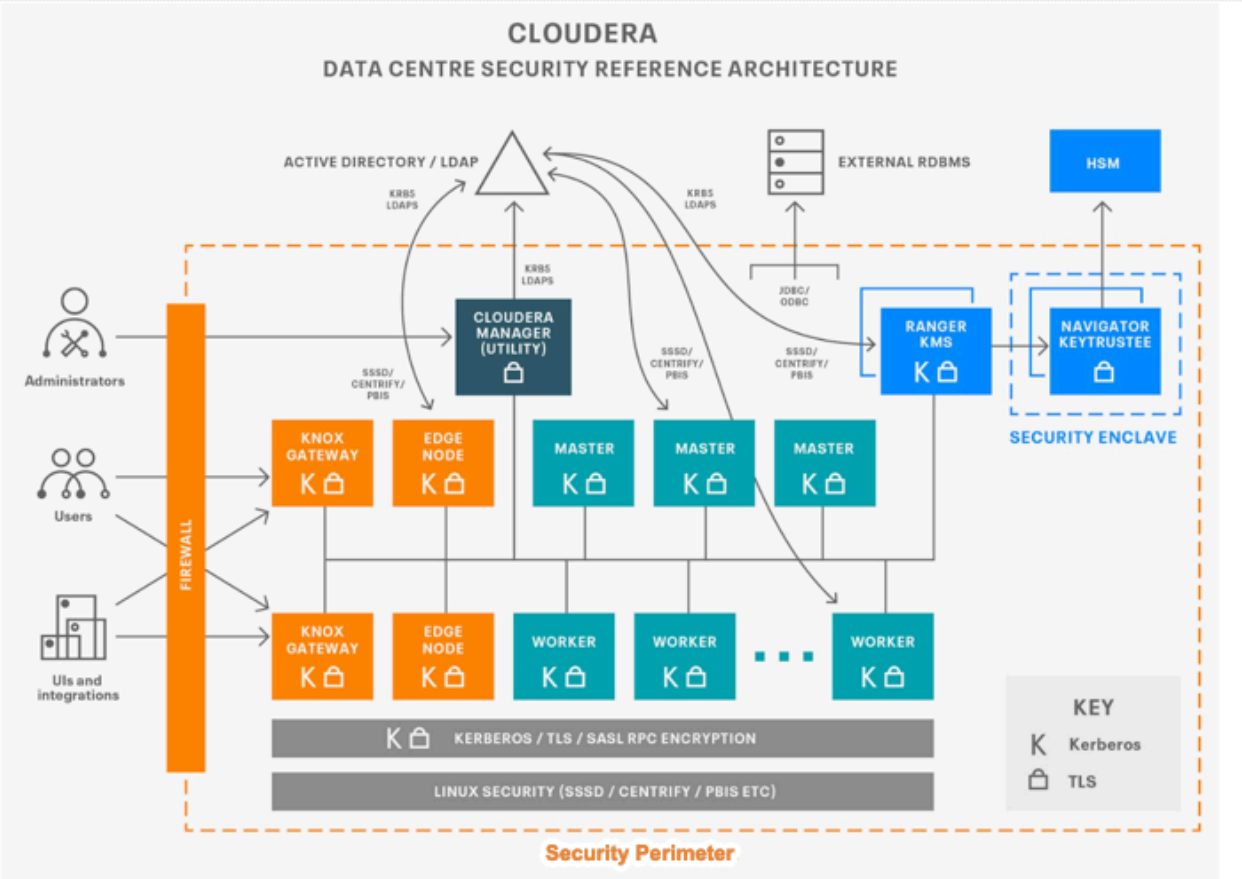

SDX

Cloudera Shared Data Experience (SDX) ensures your information is secure by design.

What is SDX?

SDX is Cloudera's design architecture that we incorporate into all of our products. It ensures your information is secure by design, so an integrated set of security and governance technologies protect your data. These technologies are built using metadata, which SDX collects to implement security policies.

SDX contains a combination of tools such as Ranger, Atlas, Knox, Hive Metastore, Cloudera Data Catalog, Cloudera Replication Manager, and Workload Manager.

SDX and Security

SDX provides consistent policy, schema, and metadata on the backend. It is effectively a single pane of glass for viewing metadata, schema, and policy within your digital environment to reduce security risks and operational costs.

Related Information

Cloudera SDX: Cloudera Shared Data Experience
Apache Ranger
Apache Atlas
Apache Knox
Cloudera Data Catalog

Security Terms

A glossary of commonly used data security terms.

Access Management

A framework for controlling what users can and cannot access.

Access Control List (ACL)

A list of associated permissions that specifies which users or system processes are granted access to a digital environment.

Apache

The Apache Software Foundation is an American nonprofit corporation that supports open source software projects. Cloudera utilizes Apache software.

Apache Atlas

Apache Atlas is a scalable and extensible set of core functional governance services that provides open metadata management for organizations. Atlas enables enterprises to effectively and efficiently meet their compliance requirements within Cloudera.

Apache Hadoop

The Apache Hadoop software library is a scalable framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.

Apache Knox

Apache Knox provides perimeter security so that the enterprise can confidently extend Hadoop access to new users while also maintaining compliance with enterprise security policies. Knox also simplifies Hadoop security for users who access the cluster data and execute jobs.

Apache Ranger

Apache Ranger is a framework to enable, monitor, and manage comprehensive data security across the platform. It is used for creating and managing policies to access data and other related objects for all services in the Cloudera stack.

Auditing

The process of assessing the quality of data to ensure its authenticity and integrity.

Authentication

The process of proving an individual is who they claim to be.

Authorization

The protection that allows or denies user access to certain resources through specific established rules.

Certificate

Digital certificates are small data files that digitally bind a cryptographic key to an organization's details. It is used in electronic documents to verify the identity of an individual, a server, or an organization.

Data At-Rest

Data that is not actively moving and is stored in a physical location such as a hard drive or a cloud.

Data In-Transit

Data that is actively moving from one location to another, whether across the internet or between devices.

Data Integrity

Maintaining and assuring the accuracy and validity of data.

Data Governance

The prcess of managing the availability, usability, integriy, and security of data based on established policies.

Data Lineage

Lineage informatin helps you understand the origin of your data and the transformations it may have gone through before arriving in a file or table.

Data Masking

Protect sensitive data from being viewed by unauthorized users through methods such as redacting, hasing, nullifying, or implementing a partial mask.

Data Protection

Prevent the accidental deletion of data files through various protective methods.

Encryption

A security method where information is translated from plaintext to ciphertext and can only be accessed or decrypted by a user with the correct encryption key. Encrypted data appears as unreadable to anyone accessing it without permission.

Kerberos

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography.

Key

Within cryptography, a key is needed to encrypt and decrypt a message. Keys determine the output of the cipher algorithm.

Key Management

The management of cryptographic keys in a cryptosystem. This includes dealing with the generation, exchange, storage, use, destruction, and replacement of keys.

Identity Management

A framework of policies that verifies user identities and ensures only authorized users have access to certain data or technology.

Lightweight Directory Access Protocol (LDAP)

Lightweight Directory Access Protocol (LDAP) is a software protocol for enabling anyone to locate organizations, individuals, and other resources such as files and devices in a network, whether on the public Internet or on a corporate intranet.

Metadata

Data about data.

Permissions

The authorization that enables users to access specific resources such as files, applications, and devices.

Personal Identifiable Information (PII)

Personal Identifiable Information (PII) includes any information that could be used to identify an individual.

Policies

A set of guiding principles or rules established to enforce data governance which determines how data is collected, used, and accessed.

Secure by Design

Cloudera treats security as a tier-1 design requirement to manage risk, reduce attack surface and vulnerabilities, and develop design concepts and patterns in an industry-preferred way

Transport Layer Security (TLS)

Transport Layer Security (TLS) is the most widely used security protocol for wire encryption. TLS provides authentication, privacy, and data integrity between applications communicating over a network by encrypting the packets transmitted between endpoints.

Cloudera Runtime 7.3.1

Accessing Cloud Data

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloud storage connectors overview 5

The Cloud Storage Connectors 5

Working with Amazon S3 6

Limitations of Amazon S3 7
Configuring Access to S3 7
Configuring Access to S3 on Cloudera Public Cloud 8
Configuring Access to S3 on Cloudera Private Cloud Base 9
Using EC2 Instance Metadata to Authenticate 10
Referencing S3 Data in Applications 11
Configuring Per-Bucket Settings 11
Customizing Per-Bucket Secrets Held in Credential Files 12
Configuring Per-Bucket Settings to Access Data Around the World 12
Encrypting Data on S3 13
SSE-S3: Amazon S3-Managed Encryption Keys 14
SSE-KMS: Amazon S3-KMS Managed Encryption Keys 15
SSE-C: Server-Side Encryption with Customer-Provided Encryption Keys 16
CSE-KMS: Amazon S3-KMS managed encryption keys 16
Configuring Encryption for Specific Buckets 18
Encrypting an S3 Bucket with Amazon S3 Default Encryption 18
Performance Impact of Encryption 18
Safely Writing to S3 Through the S3A Committers 18
Introducing the S3A Committers 18
Configuring Directories for Intermediate Data 20
Using the Directory Committer in MapReduce 20
Verifying That an S3A Committer Was Used 20
Cleaning up after failed jobs 21
Advanced Committer Configuration 21
Securing the S3A Committers 22
The S3A Committers and Third-Party Object Stores 22
Limitations of the S3A Committers 23
Troubleshooting the S3A Committers 23
Security Model and Operations on S3 24
S3A and Checksums (Advanced Feature) 25
A List of S3A Configuration Properties 25
Working with versioned S3 buckets 25
Working with Third-party S3-compatible Object Stores 27
Improving Performance for S3A 28
Working with S3 buckets in the same AWS region 28
Configuring and tuning S3A block upload 28
Optimizing S3A read performance for different file types 31
S3 Performance Checklist 32
Troubleshooting S3 33

Working with Google Cloud Storage 33

Configuring Access to Google Cloud Storage....33
Create a GCP Service Account....34
Create a Custom Role....35
Modify GCS Bucket Permissions....36
Configure Access to GCS from Your Cluster....37
Manifest committer for ABFS and GCS....38
Using the manifest committer....39
Spark Dynamic Partition overwriting....42
Job summaries in _SUCCESS files....43
Job cleanup....44
Working with Google Cloud Storage....44
Advanced topics....45
Additional Configuration Options for GCS....46

Working with the ABFS Connector....46
Introduction to Azure Storage and the ABFS Connector....46
Feature Comparisons....47
Setting up and configuring the ABFS connector....47
Configuring the ABFS Connector....48
Authenticating with ADLS Gen2....48
ADLS Proxy Setup....50
Manifest committer for ABFS and GCS....50
Using the manifest committer....51
Spark Dynamic Partition overwriting....54
Job summaries in _SUCCESS files....55
Job cleanup....56
Working with Azure ADLS Gen2 storage....56
Advanced topics....58
Performance and Scalability....59
Hierarchical namespaces vs. non-namespaces....59
Flush options....60
Using ABFS using CLI....60
Hadoop File System commands....60
Create a table in Hive....61
Accessing Azure Storage account container from spark-shell....62
Copying data with Hadoop DistCp....63
DistCp and Proxy Settings....64
ADLS Trash Folder Behavior....64
Troubleshooting ABFS....64

Cloud storage connectors overview

This content provides information and steps required for, using, securing, tuning performance, and troubleshooting access to the cloud storage services using Cloudera cloud storage connectors available for Amazon Web Services (AWS), Microsoft Azure (Azure) and Google Cloud Platform (GCP).

The primary audience of this content are the administrators and users of Cloudera deployed on cloud Infrastructure-as-a-Service (IaaS) such as AWS, Azure and GCP.

You may also use this content if your Cloudera is deployed in your own data center and you plan to access cloud storage through the connectors; however, your experience and performance may vary based on the network bandwidth between your data center and the cloud storage service. In addition, you can optionally configure other features where available.

The Cloud Storage Connectors

When deploying Cloudera clusters on cloud Infrastructure-as-a-Service (IaaS), you can take advantage of the native integration with the object storage services available on Amazon Simple Storage Service (S3) in AWS, Azure Data Lake Storage (ADLS) Gen2 in Azure, and Google Cloud Storage (GCS) in Google Cloud. This integration is through cloud storage connectors included with Cloudera. Their primary function is to help you connect to, access, and work with data the cloud storage services.

The Cloud Connectors allow you to access and work with data stored in Amazon S3, ADLS Gen2, and GCS storage services including but not limited to the following use cases:

- Collect data for analysis and then load it into Hadoop ecosystem applications such as Hive or Spark directly from cloud storage services.
- Persist data to cloud storage services for use outside of Cloudera clusters.
- Copy data stored in cloud storage services to HDFS for analysis and then copy back to the cloud when done.
- Share data between multiple Cloudera clusters, and between various external non-Cloudera systems.
- Back up Cloudera clusters using distcp.

The cloud object store connectors are implemented as individual Hadoop modules. The libraries and their dependencies are automatically placed on the classpath.

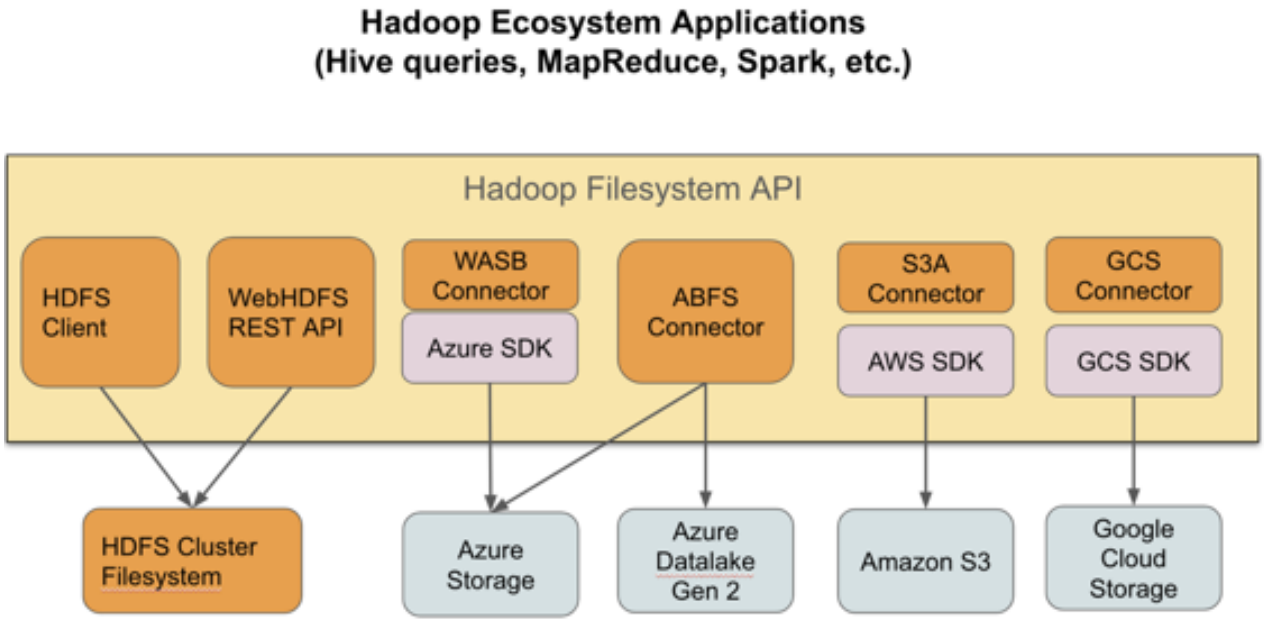

Amazon S3 is an object storage service. The S3A connector implements the Hadoop filesystem interface using AWS Java SDK to access the web service, and provides Hadoop applications with a filesystem view of the buckets. Applications can manipulate data stored in Amazon S3 buckets with an URL starting with the s3a:// prefix.

ADLS Gen2 is an object storage service that combines the features of Azure Blob Storage and ADLS Gen1. ADLS Gen2 is an object store designed for large scale big-data applications, which can be treated as a hierarchical file system, and has a security model which matches that of HDFS. Applications can manipulate data store in ADLS Gen2 with with URLs starting with the abfs:// prefix.

GCS is an object storage service for unstructured data, which can be accessed through URLs beginning with the gcs:// prefix.

Table 1: Summary of Cloud Storages and Connectors

Cloud Storage Service	Connector Description	URL Prefix
Amazon S3	The S3A connector enables reading and writing files stored in the Amazon S3 object store.	s3a://
ADLS Gen2	The ABFS connector enables reading and writing files stored in the ADLS Gen2 object store.	abfs://
GCS	The GCS connector supports reading and writing data in GCS.	gcs://

Amazon S3 can not be used as a replacement for HDFS as the cluster filesystem in Cloudera. Amazon S3 can be used as a source and destination of work.

Working with Amazon S3

The Amazon S3 object store is the standard mechanism to store, retrieve, and share large quantities of data in AWS.

Cloudera recommends that you use unique S3 bucket names across all regions.

The features of Amazon S3 include:

- Object store model for storing, listing, and retrieving data.
- Support for objects up to 5 terabytes, with many petabytes of data allowed in a single "bucket".
- Data is stored in Amazon S3 in buckets which are stored in different AWS regions.
- Buckets can be restricted to different users or IAM roles.
- Data stored in an Amazon S3 bucket is billed based on the size of data how long it is stored, and on operations accessing this data. In addition, you are billed when you transfer data between regions:
  - Data transfers between an Amazon S3 bucket and a cluster running in the same region are free of download charges (except in the special case of buckets in which data is served on a user-pays basis).
  - Data downloaded from an Amazon S3 bucket located outside the region in which the bucket is hosted is billed per megabyte.
  - Data downloaded from an Amazon S3 bucket to any host over the internet is also billed per-Megabyte.
- Data stored in Amazon S3 can be backed up with Amazon Glacier.

The Hadoop client to S3, called "S3A", makes the contents of a bucket appear like a filesystem, with directories, files in the directories, and operations on directories and files. As a result, applications which can work with data stored in HDFS can also work with data stored in S3. However, since S3 is an object store, it has certain limitations that you should be aware of.

Limitations of Amazon S3

Even though Hadoop's S3A client can make an S3 bucket appear to be a Hadoop-compatible filesystem, it is still an object store, and has some limitations when acting as a Hadoop-compatible filesystem.

The key things to be aware of are:

- Operations on directories are potentially slow and non-atomic.
- Not all file operations are supported. In particular, some file operations needed by Apache HBase are not available — so HBase cannot be run on top of Amazon S3 without additional features provided by a service. HBase Object Store Semantics (HBOSS) adapter bridges the gap between HBase and the S3A filesystem.
- Except for versions of HBase specifically designed to work with S3 storage, HBase must not use s3a:// paths for HBase storage.
- S3 can not be used as a replacement for HDFS as the cluster filesystem in Cloudera. S3 can be used as a source and destination of work.
- Data is not visible in the object store until the entire output stream has been written.
- Neither the per-file and per-directory permissions supported by HDFS nor its more sophisticated ACL mechanism are supported.
- Bandwidth between your workload clusters and Amazon S3 is limited and can vary significantly depending on network and VM load.

For these reasons, while Amazon S3 can be used as the source and store for persistent data, it cannot be used as a direct replacement for a cluster-wide filesystem such as HDFS, or be used as defaultFS.

Configuring Access to S3

IDBroker is a REST API built as part of Apache Knox’s authentication services. It allows an authenticated user to exchange a set of credentials or a token for cloud vendor access tokens. It manages mapping LDAP users to FreeIPA cloud identities for data access. It performs identity mapping for access to object stores.

This section provides information about the options that are enabled by default. For information on how IDBroker works in Cloudera, see Cloud identity federation in the *Cloudera Management Console* documentation.

For Apache Hadoop applications to be able to interact with Amazon S3, they must know the AWS access key and the secret key. This can be achieved in multiple ways, including configuration properties, environment variables, and EC2 instance metadata. While the first two options can be used when accessing S3 from a cluster running in your

own data center. IAM roles, which use instance metadata should be used to control access to AWS resources if your cluster is running on EC2.

Deployment Scenario	Authentication Options
Cluster runs on EC2	Use IAM roles to bind your EC2 VMs to a specific role, and so manage its access to AWS resources. IDBroker will provide authenticated users with the AWS credentials they need to access data in S3. There is no need to manually provide any AWS login credentials directly to users
Cluster runs in your own data center	Use configuration properties to authenticate. You can set the configuration properties globally or per-bucket.

Temporary security credentials, also known as "session credentials", can be issued. These consist of a secret key with a limited lifespan, along with a session token, another secret which must be known and used alongside the access key. The secret key is never passed to AWS services directly. Instead it is used tosign the URL and headers of the HTTP request.

By default, the S3A filesystem client follows the following authentication chain:

1. The AWS login details are looked for in the job configuration settings.
2. The AWS environment variables are then looked for.
3. An attempt is made to query the Amazon EC2 Instance Metadata Service to retrieve credentials published to EC2 VMs.

Configuring Access to S3 on Cloudera Public Cloud

IDBroker is a REST API built as part of Apache Knox's authentication services. It allows an authenticated user to exchange a set of credentials or a token for cloud vendor access tokens. It manages mapping LDAP users to FreeIPA cloud identities for data access. It performs identity mapping for access to object stores.

This section provides information about the options that are enabled by default. For information on how IDBroker works in Cloudera, see Cloud identity federation in the *Cloudera Management Console* documentation.

For Apache Hadoop applications to be able to interact with Amazon S3, they must know the AWS access key and the secret key. This can be achieved in multiple ways, including configuration properties, environment variables, and EC2 instance metadata. While the first two options can be used when accessing S3 from a cluster running in your own data center. IAM roles, which use instance metadata should be used to control access to AWS resources if your cluster is running on EC2.

Deployment Scenario	Authentication Options
Cluster runs on EC2	Use IAM roles to bind your EC2 VMs to a specific role, and so manage its access to AWS resources. IDBroker will provide authenticated users with the AWS credentials they need to access data in S3. There is no need to manually provide any AWS login credentials directly to users
Cluster runs in your own data center	Use configuration properties to authenticate. You can set the configuration properties globally or per-bucket.

Temporary security credentials, also known as "session credentials", can be issued. These consist of a secret key with a limited lifespan, along with a session token, another secret which must be known and used alongside the access key. The secret key is never passed to AWS services directly. Instead it is used tosign the URL and headers of the HTTP request.

By default, the S3A filesystem client follows the following authentication chain:

1. The AWS login details are looked for in the job configuration settings.
2. The AWS environment variables are then looked for.
3. An attempt is made to query the Amazon EC2 Instance Metadata Service to retrieve credentials published to EC2 VMs.

Configuring Access to S3 on Cloudera Private Cloud Base

For Apache Hadoop applications to be able to interact with Amazon S3, they must know the AWS access key and the secret key. This can be achieved in three different ways: through configuration properties , environment variables, or instance metadata. While the first two options can be used when accessing S3 from a cluster running in your own data center. IAM roles, which use instance metadata should be used to control access to AWS resources if your cluster is running on EC2.

Table 2: Authentication Options for Different Deployment Scenarios

Deployment Scenario	Authentication Options
Cluster runs on EC2	Use IAM roles to control access to your AWS resources. If you configure role-based access, instance metadata will automatically be used to authenticate.
Cluster runs in your own data center	Use the configuration properties to authenticate. You can set the configuration properties globally or per-bucket.

Temporary security credentials, also known as session credentials, can be issued. These consist of a secret key with a limited lifespan, along with a session token, another secret which must be known and used alongside the access key. The secret key is never passed to AWS services directly. Instead it is used to sign the URL and headers of the HTTP request.

By default, the S3A filesystem client follows the following authentication chain:

1. The fs.s3a.access.key and fs.s3a.secret.key are looked for in the Hadoop configuration properties.
2. The AWS environment variables are then looked for.
3. An attempt is made to query the Amazon EC2 Instance Metadata Service to retrieve credentials published to EC2 VMs.

Using Configuration Properties to Authenticate

To configure authentication with S3, explicitly declare the credentials in a configuration file such as core-site.xml.

```
<property>
        <name>fs.s3a.access.key</name>
        <value>ACCESS-KEY</value>
        </property>

        <property>
        <name>fs.s3a.secret.key</name>
        <value>SECRET-KEY</value>
        </property>
```

If using AWS session credentials for authentication, the secret key must be that of the session, and the fs.s3a.session.t oken option set to your session token.

```
<property>
          <name>fs.s3a.session.token</name>
          <value>SESSION-TOKEN</value>
          </property>
```

This configuration can be added for a specific bucket. For more information, see Using Per-Bucket Credentials to Authenticate.

To protect these credentials, we recommend that you use the credential provider framework to securely store and access your credentials.

To validate that you can successfully authenticate with S3, try referencing S3 in a URL.

Using Per-Bucket Credentials to Authenticate

S3A supports per-bucket configuration, which can be used to declare different authentication credentials and authentication mechanisms for different buckets. For example, a bucket s3a://nightly/ used for nightly data can be configured with a session key.

```
<property>
          <name>fs.s3a.bucket.nightly.access.key</name>
          <value>AKAACCESSKEY-2</value>
          </property>

          <property>
          <name>fs.s3a.bucket.nightly.secret.key</name>
          <value>SESSIONSECRETKEY</value>
          </property>
```

Similarly, you can set a session token for a specific bucket:

```
<property>
          <name>fs.s3a.bucket.nightly.session.token</name>
          <value>SESSION-TOKEN</value>
          </property>
```

This technique is useful for working with external sources of data, or when copying data between buckets belonging to different accounts.

Using Environment Variables to Authenticate

AWS CLI supports authentication through environment variables. These same environment variables will be used by Hadoop if no configuration properties are set.

The environment variables are:

Environment Variable	Description
AWS_ACCESS_KEY_ID	Access key
AWS_SECRET_ACCESS_KEY	Secret key
AWS_SESSION_TOKEN	Session token (only if using session authentication)

Using EC2 Instance Metadata to Authenticate

If your cluster is running on EC2, the standard way to manage access is via Amazon Identity and Access Management (IAM),which allows you to create users, groups, and roles to control access to services such as Amazon S3 via attached policies.

A role does not have any credentials such as password or access keys associated with it. Instead, if a user is assigned to a role, access keys are generated dynamically and provided to the user when needed. For more information, refer to IAM Roles for Amazon EC2 in Amazon documentation.

When launching your cluster on EC2, specify an IAM role that you want to use; if you are planning to use S3 with your cluster, make sure that the role associated with the cluster includes a policy that grants access to S3. For more information, refer to Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances in Amazon documentation. No additional configuration is required.

Note: You can use IAM Roles to control access to keys stored in Amazon's KMS Key Management service. For more information, refer to Overview of Managing Access to Your AWS KMS Resources in Amazon documentation.

Referencing S3 Data in Applications

You can reference data in Amazon S3 using a URL starting with the s3a:// prefix followed by bucket name and path to file or directory.

The URL structure is:

```
s3a://<bucket>/<dir>/<file>
```

For example, to access a file called "mytestfile" in a directory called "mytestdir", which is stored in a bucket called "mytestbucket", the URL is:

```
s3a://mytestbucket/mytestdir/mytestfile
```

The following FileSystem shell commands demonstrate access to a bucket named mytestbucket:

```
hadoop fs -ls s3a://mytestbucket/

hadoop fs -mkdir s3a://mytestbucket/testDir

hadoop fs -put testFile s3a://mytestbucket/testFile
hadoop fs -cat s3a://mytestbucket/testFile
```

Configuring Per-Bucket Settings

You can specify bucket-specific configuration values which override the common configuration values.

1. Authentication mechanisms and credentials
2. Encryption settings
3. The specific S3 endpoints to send HTTP requests to. This is essential when working with S3 regions which only support the "V4 authentication API", in case of which callers must always declare the explicit region.

All fs.s3a options other than a small set of unmodifiable values (currently fs.s3a.impl) can be set on a per-bucket basis.

To set a bucket-specific option:

1. Add a new configuration, replacing the fs.s3a. prefix on an option with fs.s3a.bucket.BUCKETNAME, where BUCKETNAME is the name of the bucket.

   For example, if you are configuring access key for a bucket called "nightly", instead of using fs.s3a.access.key property name, use fs.s3a.bucket.nightly.access.key.
2. When connecting to a bucket, all options explicitly set for that bucket will override the base fs.s3a. values, but they will not be picked up by other buckets.

Example

You may have a base configuration to use the IAM role information available when deployed in Amazon EC2 VM or an AWS container service.

```
<property>
  <name>fs.s3a.aws.credentials.provider</name>
  <value>org.apache.hadoop.fs.s3a.auth.IAMInstanceCredentialsProvider</val
ue>
</property>
```

This will be the default authentication mechanism for S3A buckets.

A bucket s3a://nightly/ used for nightly data uses a session key, so its bucket-specific configuration is:

```
<property>
  <name>fs.s3a.bucket.nightly.access.key</name>
  <value>AKAACCES-SKEY-2</value>
</property>
<property>
  <name>fs.s3a.bucket.nightly.secret.key</name>
  <value>SESSION-SECRET-KEY</value>
</property>
<property>
  <name>fs.s3a.bucket.nightly.session.token</name>
  <value>SHORT-LIVED-SESSION-TOKEN</value>
</property>

<property>
  <name>fs.s3a.bucket.nightly.aws.credentials.provider</name>
  <value>org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider</value>
</property>
```

Finally, the public s3a://landsat-pds/ bucket could be accessed anonymously, so its bucket-specific configuration is:

```
<property>
   <name>fs.s3a.bucket.landsat-pds.aws.credentials.provider</name>
   <value>org.apache.hadoop.fs.s3a.AnonymousAWSCredentialsProvider</value>
</property>
```

For all other buckets, the base configuration is used.

Customizing Per-Bucket Secrets Held in Credential Files

JCEKs credential files support the same per-bucket settings as those in XML files. To provide different credentials for different buckets, simply create par-bucket entries in the JCEKS file with the relevant secrets.

Example

1. Set base properties for fs.s3a.secret.key and fs.s3a.access.key in the JCEKS file. These will be the default.
2. Set all non-security properties per-bucket for a bucket called "frankfurt-1" in the core-site.xml. These will override the base properties when talking to the bucket "frankfurt-1".
3. For the AWS authentication secrets, set the fs.s3a.frankfurt-1.access.key and fs.s3a.frankfurt-1.secret.key properties in the JCEKS file.

Related Information

Credential Provider API Guide

Configuring Per-Bucket Settings to Access Data Around the World

S3 buckets are hosted in different AWS regions, the default being "US-East". The S3A client talks to this region by default, issuing HTTP requests to the server s3.amazonaws.com. This central endpoint can be used for accessing any bucket in any region which supports using the V2 Authentication API, albeit possibly at a reduced performance.

Each region has its own S3 endpoint, documented by Amazon. The S3A client supports these endpoints. While it is generally simpler to use the default endpoint, direct connections to specific regions (i.e. connections via region's own endpoint) may deliver performance and availability improvements, and are mandatory when working with the most recently deployed regions, such as Frankfurt and Seoul. You can use `fs.s3a.endpoint.region` to explicitly set the region for the bucket.

When deciding which endpoint to use, consider the following:

1. Applications running in EC2 infrastructure do not pay for data transfers to or from local S3 buckets. In contrast, they will be billed for access to remote buckets. Therefore, wherever possible, always use local buckets and local copies of data.

2. When the V1 request signing protocol is used, the default S3 endpoint can support data transfer with any bucket.
3. When the V4 request signing protocol is used, AWS requires the explicit region endpoint to be used — hence S3A must be configured to use the specific endpoint. This is done in the configuration option fs.s3a.endpoint.
4. All endpoints other than the default endpoint only support interaction with buckets local to that S3 instance.

If the wrong endpoint is used, the request may fail. This may be reported as a 301 redirect error, or as a 400 Bad Request. Take these failures as cues to check the endpoint setting of a bucket.

The up to date list of endpoints is provided by Amazon: https://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region

Example

The following examples show per-bucket endpoints set for the "landsat-pds" and "eu-dataset" buckets, with the endpoints set to the default central endpoint and EU/Ireland, respectively:

```
<property>
  <name>fs.s3a.bucket.landsat-pds.endpoint</name>
  <value>s3.amazonaws.com</value>
  <description>The endpoint for s3a://landsat-pds URLs</description>
</property>
<property>
  <name>fs.s3a.bucket.eu-dataset.endpoint</name>
  <value>s3-eu-west-1.amazonaws.com</value>
  <description>The endpoint for s3a://eu-dataset URLs</description>
</property>
```

The following example shows how to set the region for a bucket as ap-south-1:

```
<property>
  <name>fs.s3a.endpoint.region</name>
  <value>ap-south-1</value>
  <description>AWS S3 region for a bucket, which bypasses the parsing o
f fs.s3a.endpoint to know the region. This helps in avoiding errors while
 using privateLink URL and explicitly setting the bucket region.
  </description>
</property>
```

Explicitly declaring a bucket bound to the central endpoint ensures that if the default endpoint is changed to a new region, data stored in US-east is still reachable.

Encrypting Data on S3

The S3A filesystem client supports Amazon S3's server-side and client-side encryption mechanisms to better secure the data in S3.

Encryption on S3

- In Server-Side Encryption (SSE), the data is encrypted before it is saved to disk in S3, and decrypted when it is read. This encryption and decryption takes place in the S3 infrastructure, and is transparent to (authenticated) clients.
- In Client-Side Encryption (CSE), the data is encrypted and decrypted on the client, that is, inside the AWS S3 SDK.

In general, the specific configuration mechanism can be set via the property fs.s3a.server-side-encryption-algorithm in core-site.xml. However, some encryption options require extra settings. The encryption method configured in core-site.xml applies cluster wide. Any new file written will be encrypted with this encryption configuration. When the S3A client reads a file, S3 will attempt to decrypt it using the mechanism and keys with which the file was encrypted.

It is also possible to configure encryption for specific buckets and to mandate encryption for a specific S3 bucket.

Note the following:

- It is NOT advised to mix and match encryption types in a bucket.
- It is much simpler and safer to encrypt with just one type and key per bucket.
- You can use AWS bucket policies to mandate encryption rules for a bucket.
- You can use S3A per-bucket configuration to ensure that S3A clients use encryption policies consistent with the mandated rules.
- You can use S3 Default Encryption in the AWS console to encrypt data without needing to set anything in the client.
- Changing the encryption options on the client does not change how existing files were encrypted, except when the files are renamed.
- For all mechanisms other than SSE-C and CSE-KMS, clients do not need any configuration options set in order to read encrypted data: it is all automatically handled in S3 itself.
- Encryption options and secrets are collected by S3A Delegation Tokens and passed to workers during job submission.
- Encryption options and secrets MAY be stored in JCEKS files or any other Hadoop credential provider service. This allows for more secure storage than XML files, including password protection of the secrets.

Server-side encryption

AWS S3 supports server-side encryption inside the storage system itself. When an S3 client uploading data requests data to be encrypted, then an encryption key is used to encrypt the data as it saved to S3. It remains encrypted on S3 until deleted and clients cannot change the encryption attributes of an object once uploaded.

The server-side "SSE" encryption is performed with symmetric AES256 encryption; S3 offers different mechanisms for defining the key to use.

For server-side encryption to work, the S3 servers require secret keys to encrypt data, and the same secret keys to decrypt it. These keys can be managed in three ways:

- SSE-S3: By using Amazon S3-Managed Keys
- SSE-KMS: By using AWS Key Management Service
- SSE-C: By using customer-supplied keys

Client-side encryption

Caution: Client-side encryption (CSE-KMS) is an experimental feature. Do not mix encrypted and unencrypted data in a CSE-enabled configuration; instead maintain separate buckets. CSE will slow down performance due to the linear nature of multi-part uploads.

Client-side encryption encrypts the data on the client, before transmitting to S3, where it is stored encrypted. The data is unencrypted after downloading, when it is being read back.

In CSE-KMS, the ID of an AWS-KMS key is provided to the S3A client; the client communicates with AWS-KMS to request a new encryption key, which KMS returns along with the same key encrypted with the KMS key. The S3 client encrypts the payload and attaches the KMS-encrypted version of the key as a header to the object.

When downloading data, this header is extracted, passed to AWS KMS, and, if the client has the appropriate permissions, the symmetric key is retrieved and returned. This key is then used to decode the data.

Related Information

Troubleshooting S3

SSE-S3: Amazon S3-Managed Encryption Keys

In SSE-S3, all keys and secrets are managed inside S3. This is the simplest encryption mechanism.

Enabling SSE-S3

To write S3-SSE encrypted files, the value of fs.s3a.server-side-encryption-algorithm must be set to that of the encryption mechanism used in core-site.xml; currently only AES256 is supported.

```
<property>
  <name>fs.s3a.server-side-encryption-algorithm</name>
  <value>AES256</value>
</property>
```

Once set, all new data will be uploaded encrypted. There is no need to set this property when downloading data — the data will be automatically decrypted when read using the Amazon S3-managed key.

To learn more, refer to Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3) in AWS documentation.

Note: When encrypted files are renamed, they are de-encrypted and then re-encrypted with the encryption settings algorithm of the client application performing the rename. If a client with encryption disabled renames an encrypted file, the new file will be unencrypted.

SSE-KMS: Amazon S3-KMS Managed Encryption Keys

Amazon offers a pay-per-use key management service, AWS KMS. This service can be used to encrypt data on S3 using keys which can be centrally managed and assigned to specific roles and IAM accounts.

The AWS KMS can be used by S3 to encrypt uploaded data. When uploading data encrypted with SSE-KMS, the named key that was used to encrypt the data is retrieved from the KMS service, and used to encode the per-object secret which encrypts the uploaded data. To decode the data, the same key must be retrieved from KMS and used to unencrypt the per-object secret key, which is then used to decode the actual file.

KMS keys can be managed by an organization's administrators in AWS, including having access permissions assigned and removed from specific users, groups, and IAM roles. Only those "principals" with granted rights to a key may access it, hence only they may encrypt data with the key, and decrypt data encrypted with it. This allows KMS to be used to provide a cryptographically secure access control mechanism for data stores on S3.

Note:

AWS KMS service is not related to the Key Management Service built into Hadoop (Hadoop KMS). The Hadoop KMS primarily focuses on managing keys for HDFS Transparent Encryption. Similarly, HDFS encryption is unrelated to S3 data encryption.

Enabling SSE-KMS

To enable SSE-KMS, the property fs.s3a.server-side-encryption-algorithm must be set to SSE-KMS in core-site.xml.

```
<property>
  <name>fs.s3a.server-side-encryption-algorithm</name>
  <value>SSE-KMS</value>
</property>
```

The ID of the specific key used to encrypt the data should also be set in the property fs.s3a.server-side-encryptio n.key:

```
<property>
<name>fs.s3a.server-side-encryption.key</name>
<value>arn:aws:kms:us-west-2:360379543683:key/071a86ff-8881-4ba0-9230-95af
6d01ca01</value>
</property>
```

If your account is set up set up with a default KMS key and fs.s3a.server-side-encryption.key is unset, the default key will be used.

Alternatively, organizations may define a default key in the Amazon KMS; if a default key is set, then it will be used whenever SSE-KMS encryption is chosen and the value of fs.s3a.server-side-encryption.key is empty.

Note: AWS Key Management Service (KMS) is pay-per-use, working with data encrypted via KMS keys incurs extra charges during data I/O.

To learn more, refer to Protecting Data Using Server-Side Encryption with AWS KMS-Managed Keys (SSE-KMS) in the AWS documentation.

Note: The AWS KMS key used must be in the same region as the S3 Bucket where the data is being encrypted.

Note: The KMS key specified is only used when writing new data, or when renaming objects. When reading an existing object, whichever KMS key was used to encrypt the data is reused to decrypt it.

IAM Role permissions for working with SSE-KMS

All IAM roles which need to read data encrypted with SSE-KMS must have the permissions to decrypt using the specific key the data was encrypted with: kms:Decrypt

All IAM roles which need to both read and write data need the encrypt and decrypt permissions (encrypt-only permission is not supported).

```
kms:Decrypt
kms:GenerateDatakey
```

If a role does not have the permissions to read data, it will fail with an java.nio.AccessDeniedException.

Note: The renaming files require the permission to decrypt the data, as it is decrypted and then re-encrypted as it is copied. See AWS KMS API Permissions: Actions and Resources Reference for more details on KMS permissions.

SSE-C: Server-Side Encryption with Customer-Provided Encryption Keys

In SSE-C, the client supplies the secret key needed to read and write data.

The same SSE-C key must be used on all clients reading or writing the data; this must be set client-side, in the hadoop configuration.

For hadoop applications to work reliably, SSE-C with a common key *must* be used exclusively within a bucket if it is to be used at all. This is the only way to ensure that path and directory listings do not fail with "Bad Request" errors.

Enabling SSE-C

To use SSE-C, the configuration option fs.s3a.server-side-encryption-algorithm must be set to SSE-C, and a base-64 encoding of the key placed in fs.s3a.server-side-encryption.key.

```
<property>
  <name>fs.s3a.server-side-encryption-algorithm</name>
  <value>SSE-C</value>
</property>

<property>
  <name>fs.s3a.server-side-encryption.key</name>
  <value>RG8gbm90IGV2ZXIgbG9nIHRoaXMga2V5IG9yIG90aGVyd2lzZSBzaGFyZSBpdA==</
value>
</property>
```

This property can be set in a Hadoop JCEKS credential file, which is significantly more secure than embedding secrets in the XML configuration file.

CSE-KMS: Amazon S3-KMS managed encryption keys

Amazon S3 Client Side Encryption (S3-CSE) is used to encrypt data on the client side and then transmit it to S3 storage. The same encrypted data is transmitted to the client while reading and then decrypted on the client side.

Introduction

S3-CSE uses AmazonS3EncryptionClientV2.java as the Amazon S3 client. The encryption and decryption is done by AWS SDK. Currently only the CSE-KMS method of client-side encryption is supported.

Previously, client-side encryption was unavailable due to the AWS S3 client padding uploaded objects with a 16 byte footer. This meant that files were shorter when being read than when are listed them through any of the list API calls/ getFileStatus(). This broke many applications, including anything seeking near the end of a file to read a footer, as ORC and Parquet do.

There is now a workaround: compensate for the footer in listings when CSE is enabled.

- When listing files and directories, 16 bytes are subtracted from the length of all non-empty objects (greater than or equal to 16 bytes).
- Directory markers MAY be longer than 0 bytes long.

The length of files when listed through the S3A client is now going to be shorter than the length of files listed with other clients--including S3A clients where S3-CSE has not been enabled.

Features

- Supports client-side encryption with keys managed in AWS KMS.
- Encryption settings propagated into jobs through any issued delegation tokens.
- Encryption information stored as headers in the uploaded object.

Limitations

- Performance will be reduced. All encrypt/decrypt is now being done on the client.
- Writing files may be slower, as only a single block can be encrypted and uploaded at a time.
- Multipart Uploader API is disabled.
- S3 Select is not supported.
- Multipart uploads would be serial, and partSize must be a multiple of 16 bytes.
- Maximum message size in bytes that can be encrypted under this mode is 2^36-32, or ~64G, due to the security limitation of AES/GCM as recommended by NIST.

Enabling CSE-KMS

To enable CSE-KMS, the property fs.s3a.server-side-encryption-algorithm must be set to CSE-KMS in core-site.xml.

- Generate an AWS KMS Key ID from the AWS console for your bucket, with the same region as the storage bucket.
- If already created, view the KMS key ID following these steps.
- Set fs.s3a.server-side-encryption-algorithm=CSE-KMS.
- Set fs.s3a.server-side-encryption.key=<KMS_KEY_ID>.

Note: If fs.s3a.server-side-encryption-algorithm=CSE-KMS is set, the fs.s3a.server-side-encryption.key=<KMS_KEY_ID> property must be set for S3-CSE to work.

```
<property>
     <name>fs.s3a.server-side-encryption-algorithm</name>
     <value>CSE-KMS</value>
 </property>

 <property>
     <name>fs.s3a.server-side-encryption.key</name>
     <value>${KMS_KEY_ID}</value>
 </property>
```

Configuring Encryption for Specific Buckets

S3A's per-bucket configuration mechanism can be used to configure the encryption mechanism and credentials for specific buckets.

For example, to access the bucket called "production" using SSE-KMS with the key ID arn:aws:kms:us-west-2:360379543683:key/071a86ff-8881-4ba0-9230-95af6d01ca01, the settings are as follows:

```
<property>
  <name>fs.s3a.bucket.production.server-side-encryption-algorithm</name>
  <value>SSE-KMS</value>
</property>
```

```
<property>
  <name>fs.s3a.bucket.production.server-side-encryption.key</name>
  <value>arn:aws:kms:us-west-2:360379543683:key/071a86ff-8881-4ba0-9230-95
af6d01ca01</value>
</property>
```

Encrypting an S3 Bucket with Amazon S3 Default Encryption

To guarantee that all data uploaded to a bucket is encrypted, it is possible to set a default encryption option for a bucket in the AWS management console.

For more information, see Amazon S3 Default Encryption for S3 Buckets.

- This does not encrypt any data already stored in the bucket.
- S3A clients can still be configured to use a different encryption option if desired; this is the default value to use if no other policy was set.

A default encryption across a bucket offers significant benefits:

- It guarantees that all clients uploading data have encryption enabled.
- It guarantees that when a file is renamed, it will be re-encrypted, even if the client does not explicitly request encryption.
- If applied to an empty bucket, it guarantees that all future uploaded data in the bucket is encrypted.

We recommend selecting an encryption policy for a bucket when the bucket is created, and setting it in the bucket policy. This stops misconfigured clients from unintentionally uploading unencrypted data, or decrypting data when renaming files.

Performance Impact of Encryption

Server Side encryption slightly slows down performance when reading data from S3, both in the reading of data during the execution of a query, and in scanning the files prior to the actual scheduling of work.

Amazon throttles reads and writes of S3-SSE data, which results in a significantly lower throughput than normal S3 IO requests. The default rate, 600 requests/minute, means that at most ten objects per second can be read or written using SSE-KMS per second — across an entire hadoop cluster (or the entire customer account). The default limits may be suitable during development — but in large scale production applications the limits may rapidly be reached. Contact Amazon to increase capacity.

Safely Writing to S3 Through the S3A Committers

The S3A committers are three different committers used to commit work directly to Mapreduce and Spark. The committers are enabled by default for Spark in Cloudera.

Introducing the S3A Committers

Amazon's S3 Object Store is not a filesystem: some expected behaviors of a filesystem are missing.

Some of the following aspects of S3 that are different from a filesystem are as follows:

- Directory listings are only eventually consistent.
- File overwrites and deletes are only eventually consistent: readers may get old data.
- There is no rename operation; it is mimicked in the S3A client by listing a directory and copying each file underneath, one-by-one.

Because directory rename is mimicked by listing and then copying files the eventual consistency of both listing and reading fails may result in incorrect data. And, because of the copying: it is slow.

The normal means by which Hadoop MapReduce and Apache Spark commit work from multiple tasks is through renaming the output. Each task attempt writes to a private task attempt directory; when the task is given permission to commit by the MapReduce Application Master or Spark Driver, this task attempt directory is renamed into the job attempt directory. When the job is ready to commit, the output of all the tasks is merged into the final output directory, again by renaming files and directories.

This is fast and safe on HDFS and similar filesystems, and on object stores with rename operations, such as Azure WASB. The time to commit work to S3 is proportional to the amount of data written. An operation which may work well during development can turn out to be unusable in production.

To address this, S3A committers were developed. They allow the output of MapReduce and Spark jobs to be written directly to S3, with a time to commit the job independent of the amount of data created.

What Are the S3A Committers?

The S3A committers are three different committers which can be used to commit work directly to Map-reduce and Spark. They differ in how they deal with conflict and how they upload data to the destination bucket —but underneath they all share much of the same code.

They rely on a specific S3 feature: multipart upload of large files.

When writing a large object to S3, S3A and other S3 clients use a mechanism called “Multipart Upload”.

The caller initiates a “multipart upload request”, listing the destination path and receiving an upload ID to use in the upload operations.

The caller then uploads the data in a series of HTTP POST requests, closing with a final POST listing the blocks uploaded.

The uploaded data is not visible until that final POST request is made, at which point it is published in a single atomic operation.

This mechanism is always used by S3A whenever it writes large files; the size of each part is set to the value of fs.s3a.multipart.size

The S3A Committers use the same multipart upload process, but convert it into a mechanism for committing the work of tasks because of a special feature of the mechanism: The final POST request does not need to be issued by the same process or host which uploaded the data.

The output of each worker task can be uploaded to S3 as a multipart upload, but without issuing the final POST request to complete the upload. Instead, all the information needed to issue that request is saved to a cluster-wide filesystem (HDFS or potentially S3 itself)

When a job is committed, this information is loaded, and the upload completed. If a a task is aborted of fails: the upload is not completed, so the output does not become visible. If the entire job fails, none of its output becomes visible.

For further reading, see:

- HADOOP-13786: Add S3A committers for zero-rename commits to S3 endpoints.
- S3A Committers: Architecture and Implementation.
- A Zero-Rename Committer: Object-storage as a Destination for Apache Hadoop and Spark.

The Three Committers

The three different S3A committers are directory committer, partitioned committer, magic committer.

- Directory Committer: Buffers working data to the local disk, uses HDFS to propagate commit information from tasks to job committer, and manages conflict across the entire destination directory tree.
- Partitioned Committer: Identical to the Directory committer except that conflict is managed on a partition-by-partition basis. This allows it to be used for in-place updates of existing datasets. It is only suitable for use with Spark.
- Magic Committer: Data is written directly to S3, but “magically” re-targeted at the final destination. Conflict is managed across the directory tree. It requires a consistent S3 object store.

We currently recommend use of the “Directory” committer: it is the simplest of the set, and by using HDFS to propagate data from workers to the job committer, this makes it easier to set up.

The rest of the documentation only covers the Directory Committer: to explore the other options, consult the Apache documentation.

Configuring Directories for Intermediate Data

In addition to fs.s3a.committer.name, two other core-site.xml configuration options are used to control where intermediate is stored.

A location is in the local filesystem for buffering data

```
<property>
  <name>fs.s3a.buffer.dir</name>
  <value>${hadoop.tmp.dir}/s3a</value>
  <description>Comma separated list of directories that will be used to b
uffer file
  uploads to.</description>
</property>
```

These directories will store the output created by all active tasks until each task is committed; the more worker processes/spark worker threads a host can support, the more disk space will be needed. Multiple disks can be listed to help spread the load, and recover from disk failure.

A location in the cluster's HDFS filesystem to share summary data about pending uploads.

```
<property>
  <name>fs.s3a.committer.staging.tmp.path</name>
  <value>tmp/staging</value>
</property>
```

These files are generally quite small: a few kilobytes per task.

Using the Directory Committer in MapReduce

Once the propertyfs.s3a.committer.name is set, Hadoop MapReduce jobs writing to paths using the s3a:// schema will automatically switch to the new committers.

Jobs using any other filesystem as a destination will still use the normal file committer.

Verifying That an S3A Committer Was Used

When working correctly, the only sign the new committers are in use is that it should be faster to use S3 as a destination of work.

There is a straightforward way to determine if a new committer was used: examine the _SUCCESS file created in the destination directory of a query. With the original file committer, this is a zero-byte file. The new S3A committers all write a JSON file describing the committer used, the files created and various diagnostics information.

Listing this file is enough to show whether an S3A committer was used:

```
hadoop fs -ls s3a://guarded-bucket/datasets/orc/_SUCCESS
```

If this file is of size 0. then no S3A committer was used. If the file length is greater than zero, then an S3A committer was used.

To see more details about the job commit operation, the file's contents can be printed.

```
hadoop fs -cat s3a://guarded-bucket/datasets/orc/_SUCCESS
```

Cleaning up after failed jobs

The S3A committers upload data in the tasks, completing the uploads when the job is committed.

About this task

Amazon AWS still bill you for all data held in this “pending” state. You can use bucket lifecycle rules to automate the cleanup of these jobs.

Procedure

1. Go to the AWS S3 console: https://s3.console.aws.amazon.com/.
2. Find the bucket you are using as a destination of work.
3. Select the Management tab.
4. Select Add a new lifecycle rule.
5. Create a rule “cleanup uploads” with no filter, and without any “transitions”.

   Configure an “Expiration” action of Clean up incomplete multipart uploads.
6. Select a time limit for outstanding uploads, such as 1 Day.
7. Review and confirm the lifecycle rule

   You need to select a limit of how long uploads can be outstanding. For Hadoop applications, this is the maximum time that either an application can write to the same file and the maximum time which a job may take. If the timeout is shorter than either of these, then programs are likely to fail.

   Once the rule is set, the cleanup is automatic.

Advanced Committer Configuration

The Apache documentation covers the full set of configuration options for the committers.

Enabling Speculative Execution

The S3A committers all support speculative execution.

For MapReduce, enable the following properties in the job:

```
mapreduce.map.speculative true
mapreduce.reduce.speculative true
```

For Spark, set the following configuration option:

```
spark.speculation true
```

Using Unique Filenames to Avoid File Update Inconsistency

When updating existing datasets, if a new file overwrites an existing file of the same name, S3's eventual consistency on file updates means that the old data may still be returned.

To avoid this, unique filenames should be used when creating files. The property fs.s3a.committer.staging.unique-file names enables this.

```
<property>
  <name>fs.s3a.committer.staging.unique-filename</name>
  <value>true</value>
```

```
</property>
```

It is set to true by default; you only need to disable it if you explicitly want newly created files to have the same name as any predecessors.

Speeding up Job Commits by Increasing the Number of Threads

When an individual job writes many files to S3, the time to commit the job can increase.

It can be speeded up by increasing the value of fs.s3a.committer.threads>. However, the value of fs.s3a.connection.maximum must be at least this size otherwise the limit on the number of parallel uploads is still limited.

```
<property>
  <name>fs.s3a.committer.threads</name>
  <value>8</value>
</property>
<property>
  <name>fs.s3a.connection.maximum</name>
  <value>15</value>
</property>
```

Securing the S3A Committers

There are a few security considerations when using the S3A committers.

S3 Bucket Permissions

To use an S3A committer, the account/role must have the same AWS permissions as needed to write to the destination bucket.

The multipart upload list/abort operations may be a new addition to the permissions for the active role.

Local Filesystem Security

All the committers use the directories listed in fs.s3a.buffer.dir to store staged/buffered output before uploading it to S3.

To ensure that other processes running on the same host do not have access to this data, unique paths should be used per-account.

This requirement holds for all applications working with S3 through the S3A connector.

HDFS Security

The directory and partitioned committers use HDFS to propagate commit information between workers and the job committer.

These intermediate files are saved into a job-specific directory under the path ${fs.s3a.committer.staging.tmp.path}/${user} where ${user} is the name of the user running the job.

The default value of fs.s3a.committer.staging.tmp.path is tmp/staging, Which will be converted at run time to a path under the current user's home directory, essentially /user/${user}/tmp/staging/${user}/.

Provided the user's home directory has access restricted to trusted accounts, this intermediate data will be secure.

If the property fs.s3a.committer.staging.tmp.path is changed to a different location, then this path will need to be secured to protect pending jobs from tampering.

Note: This intermediate data does not contain the output, merely the listings of all pending files and the MD5 checksums of each block.

The S3A Committers and Third-Party Object Stores

The S3A committers will work with any object store which implements the AWS S3 protocols.

The directory committer requires a consistent cluster filesystem to propagate commit information from the worker processes to the job committer. This is normally done in HDFS.

If the third-party object store is consistent, then it may also be used as the cluster filesystem. Set fs.s3a.committer.staging.tmp.path to a chosen path in the store.

Limitations of the S3A Committers

There are limitations of the S3A committers associated with custom file output formats, MapReduce API output format, and non-availability of Hive support.

Custom file output formats and their committers

Output formats which implement their own committers do not automatically switch to the new committers. If such a custom committer relies on renaming files to commit output, then it will depend on a consistent view of the object store, and take time to commit output proportional to the amount of data and the number of files.

To determine if this is the case, find the subclass of org.apache.hadoop.mapreduce.lib.output.FileOutputFormat which implements the custom format, to see if it subclasses the getOutputCommitter() to return its own committer, or has a custom subclass oforg.apache.hadoop.mapreduce.lib.output.FileOutputCommitter.

It may be possible to migrate such a committer to support store-specific committers, as was done for Apache Parquet support in Spark. Here a subclass of Parquet's ParquetOutputCommitter was implemented to delegates all operations to the real committer.

MapReduce V1 API Output Format and Committers

Only the MapReduce V2 APIs underorg.apache.hadoop.mapreduce support the new committer binding mechanism. The V1 APIs under the org.apache.hadoop.mapred package only bind to the file committer and subclasses. The v1 APIs date from Hadoop 1.0 and should be considered obsolete. Please migrate to the v2 APIs, not just for the new committers, but because the V2 APIs are still being actively developed and maintained.

No Hive Support

There is currently no Hive support for the S3A committers.

Troubleshooting the S3A Committers

The Apache documentation contains information about troubleshooting the committers.

The primary issue which surfaces is actually programs not switching to the new committers. There are three common reasons for this.

- The configuration settings to switch to the new committer are not being picked up. This is particularly common in Spark, which is harder to set up.
- The program is using the older V1 MapReduce APIs. Fix: switch to the V2 API.
- The output format the program uses is explicitly creating its own committer. This can only be fixed by modifying the program.

To help debug Spark's configuration, there is an option which can be set to forcibly fail the spark query if the path output committer is used. However, the file committer is being returned.

```
spark.hadoop.pathoutputcommit.reject.fileoutput true
```

There is also the blunt-instrument approach of causing the original output committer to crash with an invalid configuration.

```
spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version 3
```

This (invalid) option ensures that if the original file committer is used, it will raise an exception.

To enable low-level logging of the committers, set the log-level of the package org.apache.hadoop.fs.s3a.commit to DEBUG. With Log4J, this can be one in log4j.properties:

```
log4j.logger.org.apache.hadoop.fs.s3a.commit=DEBUG
```

Security Model and Operations on S3

The security and permissions model of Amazon S3 is very different from this of a UNIX-style filesystem: on Amazon S3, operations which query or manipulate permissions are generally unsupported. Operations to which this applies include: chgrp, chmod, chown, getfacl, and setfacl. The related attribute commands getfattr andsetfattr are also unavailable.

In addition, operations which try to preserve permissions (for example fs -put -p) do not preserve permissions.

Although these operations are unsupported, filesystem commands which list permission and user/group details usually simulate these details. As a consequence, when interacting with read-only object stores, the permissions found in “list” and “stat” commands may indicate that the user has write access — when they do not.

Amazon S3 has a permissions model of its own. This model can be manipulated through store-specific tooling. Be aware that some of the permissions which can be set — such as write-only paths, or various permissions on the root path — may be incompatible with the S3A client. It expects full read and write access to the entire bucket with trying to write data, and may fail if it does not have these permissions.

As an example of how permissions are simulated, here is a listing of Amazon's public, read-only bucket of Landsat images:

```
$ hadoop fs -ls s3a://landsat-pds/
Found 10 items
drwxrwxrwx - mapred 0 2016-09-26 12:16 s3a://landsat-pds/L8
-rw-rw-rw- 1 mapred 23764 2015-01-28 18:13 s3a://landsat-pds/index.html
drwxrwxrwx - mapred 0 2016-09-26 12:16 s3a://landsat-pds/landsat-pds_stats
-rw-rw-rw- 1 mapred 105 2016-08-19 18:12 s3a://landsat-pds/robots.txt
-rw-rw-rw- 1 mapred 38 2016-09-26 12:16 s3a://landsat-pds/run_info.json
drwxrwxrwx - mapred 0 2016-09-26 12:16 s3a://landsat-pds/runs
-rw-rw-rw- 1 mapred 27458808 2016-09-26 12:16 s3a://landsat-pds/scene_list.
gz
drwxrwxrwx - mapred 0 2016-09-26 12:16 s3a://landsat-pds/tarq
drwxrwxrwx - mapred 0 2016-09-26 12:16 s3a://landsat-pds/tarq_corrupt
drwxrwxrwx - mapred 0 2016-09-26 12:16 s3a://landsat-pds/test
```

The following is evident from the examples:

- All files are listed as having full read/write permissions.
- All directories appear to have full rwx permissions.
- The replication count of all files is "1".
- The owner of all files and directories is declared to be the current user (mapred).
- The timestamp of all directories is actually that of the time the -ls operation was executed. This is because these directories are not actual objects in the store; they are simulated directories based on the existence of objects under their paths.

When an attempt is made to delete one of the files, the operation fails — despite the permissions shown by the ls command:

```
$ hadoop fs -rm s3a://landsat-pds/scene_list.gz
rm: s3a://landsat-pds/scene_list.gz: delete on s3a://landsat-pds/scene_list.
gz:
com.amazonaws.services.s3.model.AmazonS3Exception: Access Denied (Service: A
mazon S3;
Status Code: 403; Error Code: AccessDenied; Request ID: 1EF98D5957BCAB3D),
```

```
S3 Extended Request ID: wi3veOXFuFqWBUCJgV3Z+NQVj9gWgZVdXlPU4KBbYMsw/gA+hyh
RXcaQ+PogOsDgHh31HlTCebQ=
```

This demonstrates that the listed permissions cannot be taken as evidence of write access. Only object manipulation can determine this.

S3A and Checksums (Advanced Feature)

The S3A connector can be configured to export the HTTP etag of an object as a checksum, by setting the option fs.s3a.etag.checksum.enabled to true. When unset (the defaut), S3A objects have no checksum.

```
$ hadoop fs -touchz s3a://hwdev-bucket/src/something.txt
$ hadoop fs -checksum s3a://hwdev-bucket/src/something.txt
s3a://hwdev-bucket/src/something.txt NONE
```

Once set, S3A objects have a checksum which is created on upload.

```
$ hadoop fs -Dfs.s3a.etag.checksum.enabled=true -checksum s3a://hwdev-bucket
/src/something.txt
s3a://hwdev-bucket/src/something.txt etag 6434316438636439386630306232303
4653938303039393865636638343237 65
```

This checksum is not compatible with that or HDFS, so cannot be used to compare file versions when using the -update option on DistCp between S3 and HDFS. More specifically, unless -skipcrccheck is set, the DistCP operation will fail with a checksum mismatch. However, it can be used for incremental updates within and across S3A buckets.

```
$ hadoop distcp -Dfs.s3a.etag.checksum.enabled=true --update s3a://hwdev-buc
ket/src s3a://hwdev-bucket/dest
$ hadoop fs -Dfs.s3a.etag.checksum.enabled=true -checksum s3a://hwdev-bucke
t/dest/something.txt
s3a://hwdev-bucket/src/something.txt etag 6434316438636439386630306232303 46
539383030393938656366383432376 5
```

As the checksums match small files created as a single block, incremental updates will not copy unchanged files. For large files uploaded using multiple blocks, the checksum values may differ in which case the source file will be copied again.

A List of S3A Configuration Properties

All S3A client options are configured with options with the prefix fs.s3a.

For information about the S3A configuration properties, see General S3A client configuration.

Working with versioned S3 buckets

AWS S3 supports "Versioning" in buckets, where the bucket is configured to save older versions of objects. When an object is overwritten (or even deleted), the old version can be accessed when a request is made for the object using the original version ID.

For more information, see Using Versioning.

The S3A connector supports versioning in the following ways:

- When a file is opened for reading, the version ID of that file is recorded, and then for the duration of the file read (seconds, minutes, hours...) only that version of the data is read. Even if the file is overwritten, the single ongoing file read will always read the original data.

- When a file is copied (as in a rename operation), the version ID is used to guarantee that even if the source file is overwritten, the copied file will be the original version.
- It can be used as an alternative to moving deleted files to a trash location: simply delete the files and then recover them later. Note: The S3A connector does not provide a recovery tool.

The following are some issues you must be aware of when using versioning:

- Too many S3 Tombstone markers from deleted objects will slow down directory listings, and can result in clients being throttled (https://docs.aws.amazon.com/AmazonS3/latest/dev/troubleshooting.html#troubleshooting-by-symptom-increase-503-reponses)
- Old versions of files are still billed for. That means files overwritten by new versions are still billed by the megabyte until the previous versions are permanently deleted.
- While working with Hive managed or external table directories, staging directories are created for each table or partition every time they are queried. These staging directories are deleted after the query is complete, but copies are kept. Cloudera recommends to add appropriate S3 lifecycle rule to remove the copies (created due to versioning) after a period of time.

To keep costs down and minimize performance problems, we recommend having a lifecycle rule on the bucket which deletes older versions of files after a number of days. This can be done from the Management tab of the AWS S3 console.

Here is an example policy which deletes all versions of objects which were overwritten more than a week previously, while also cleaning up tombstone markers and incomplete file uploads:

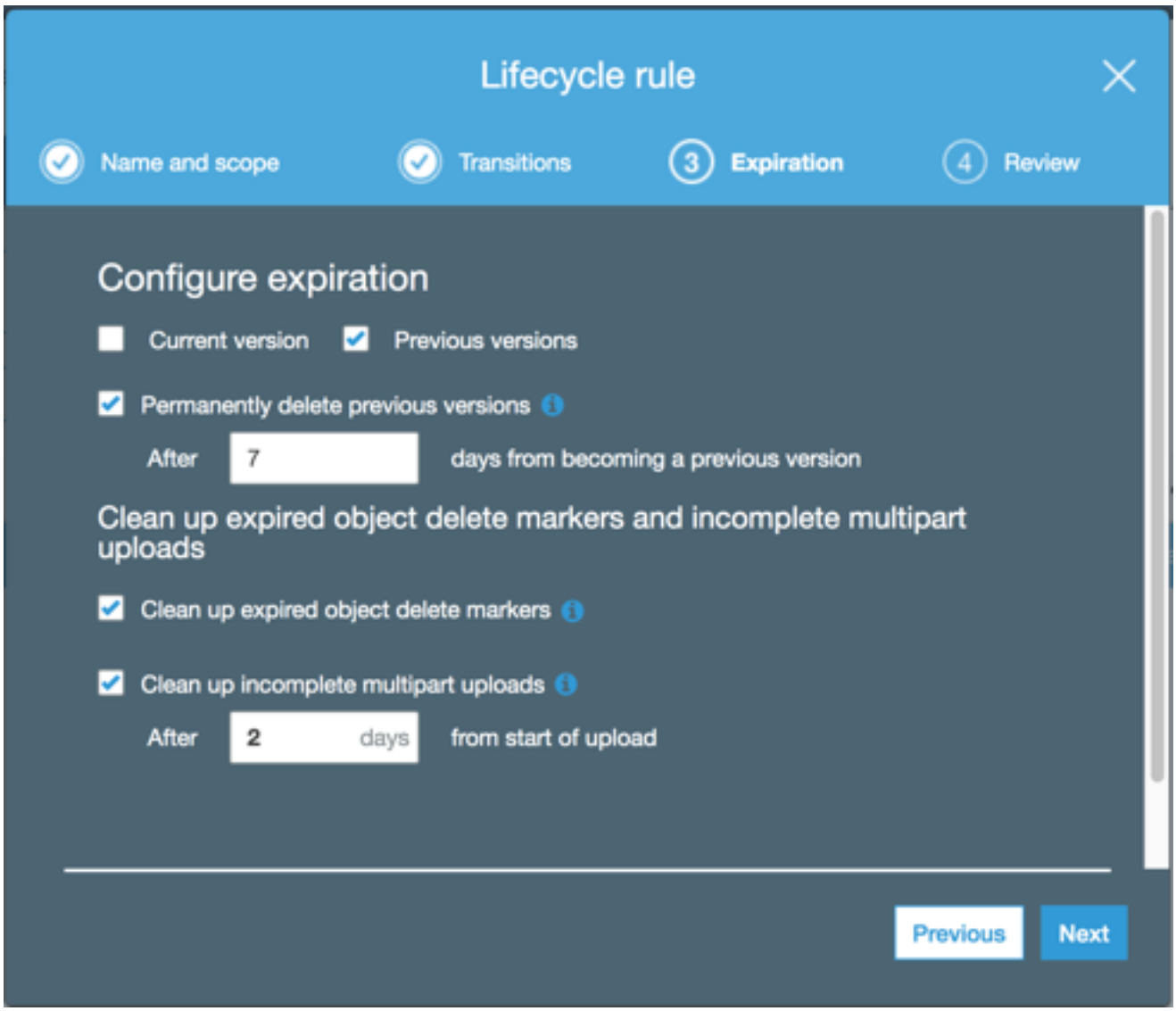

Working with Third-party S3-compatible Object Stores

The S3A Connector can work with third-party object stores; some vendors test the connector against their stores — and even actively collaborate in developing the connector in the open source community.

You need to configure the S3A connector in Cloudera Manager to be able to use it with third-party objects stores.

1. Log in to Cloudera Manager, and select your cluster.
2. Select the Configuration tab.
3. Search for Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml.
4. Use the + button to add each of the following configuration parameter and provide its value.
5. Click Save, and restart the service.

Option	Change to
fs.s3a.endpoint	(the hostname of the S3 Store)
fs.s3a.endpoint.region	Specify the AWS region for the region.
fs.s3a.path.style.access	true

Option	Change to
fs.s3a.signing-algorithm	If the default signing mechanism is rejected, another mechanism may work from: "QueryStringSignerType", "S3SignerType", "AWS3SignerType", "AWS4SignerType", "AWS4UnsignedPayloadSignerType" and "NoOpSignerType".
fs.s3a.connection.ssl.enabled	Set to "false" if HTTP is used instead of HTTPS
fs.s3a.multiobjectdelete.enable	Set to "false" if bulk delete operations are not supported.
fs.s3a.list.version	Set to "1" if the list directories with the default "2" option fails with an error.

The following aspects need to be considered when using using S3 connector with third-party object stores:

- Third-party object stores are generally consistent. The S3A Committers will still offer better performance, and should be used for MapReduce and Spark.
- Except for versions of HBase specifically designed to work with S3 storage, HBase must not use s3a:// paths for HBase storage.
- S3 can not be used as a replacement for HDFS as the cluster filesystem in Cloudera. S3 can be used as a source and destination of work.
- Encryption may or may not be supported, refer to the documentation of the third-party object store.
- Security permissions are likely to be implemented differently from the IAM role mode —again, refer to the documentation of the third-party object store to see what is available.

Improving Performance for S3A

You can consider various options for improving performance when working with data stored in Amazon S3.

The bandwidth between the Cloudera cluster and Amazon S3 is the upper limit to how fast data can be copied into S3. The further the Cloudera cluster is from the Amazon S3 installation, or the narrower the network connection is, the longer the operation will take. Even a Cloudera cluster deployed within Amazon's own infrastructure may encounter network delays from throttled VM network connections.

Network bandwidth limits notwithstanding, there are some options which can be used to tune the performance of an upload:

- Working with S3 buckets in the same AWS region on page 28
- Configuring and tuning S3A block upload on page 28

Working with S3 buckets in the same AWS region

A foundational step to getting good performance is working with buckets close to the Hadoop cluster, where "close" is measured in network terms.

Maximum performance is achieved from working with S3 buckets in the same AWS region as the cluster. For example, if your cluster is in North Virginia ("US East"), you will achieve best performance if your S3 bucket is in the same region.

In addition to improving performance, working with local buckets ensures that no bills are incurred for reading from the bucket.

Configuring and tuning S3A block upload

Because of the nature of the S3 object store, data written to an S3A OutputStream is not written incrementally — instead, by default, it is buffered to disk until the stream is closed in its close() method. This can make output slow because the execution time for OutputStream.close() is proportional to the amount of data buffered and inversely proportional to the bandwidth between the host to S3; that is O(data/bandwidth).

Other work in the same process, server, or network at the time of upload may increase the upload time.

In summary, the further the process is from the S3 store, or the smaller the EC2 VM is, the longer it will take complete the work. This can create problems in application code:

- Code often assumes that the close() call is fast; the delays can create bottlenecks in operations.
- Very slow uploads sometimes cause applications to time out - generally, threads blocking during the upload stop reporting progress, triggering timeouts.
- Streaming very large amounts of data may consume all disk space before the upload begins.

Tuning S3A Uploads

When data is written to S3, it is buffered locally and then uploaded in multi-Megabyte blocks, with the final upload taking place as the file is closed.

The following major configuration options are available for the S3A block upload options. These are used whenever data is written to S3.

Parameter	Default Value	Description
fs.s3a.multipart.size	100M	Defines the size (in bytes) of the blocks into which the upload or copy operations will be split up. A suffix from the set {K,M,G,T,P} may be used to scale the numeric value.
fs.s3a.fast.upload.active.blocks	8	Defines the maximum number of blocks a single output stream can have active uploading, or queued to the central FileSystem instance's pool of queued operations. This stops a single stream overloading the shared thread pool.
fs.s3a.buffer.dir	Empty value	A comma separated list of temporary directories use for storing blocks of data prior to their being uploaded to S3. When unset (by default), the Hadoop temporary directory hadoop.tmp.dir is used.
fs.s3a.fast.upload.buffer	disk	The fs.s3a.fast.upload.buffer determines the buffering mechanism to use when uploading data. Allowed values are: disk, array, bytebuffer: • (default) "disk" will use the directories listed in fs.s3a.buffer.dir as the location(s) to save data prior to being uploaded. • "array" uses arrays in the JVM heap. • "bytebuffer" uses off-heap memory within the JVM. Both "array" and "bytebuffer" will consume memory in a single stream up to the number of blocks set by: fs.s3a.multipart.size * fs.s3a.fast.upload.active.blocks. If using either of these mechanisms, keep this value low. The total number of threads performing work across all threads is set by fs.s3a.threads.max, with fs.s3a.max.total.tasks values setting the number of queued work items.

Note that:

- If the amount of data written to a stream is below that set in fs.s3a.multipart.size, the upload takes place after the application has written all its data.
- The maximum size of a single file in S3 is one thousand blocks, which, for uploads means 10000 * fs.s3a.multipart.size. Too A small value of fs.s3a.multipart.size can limit the maximum size of files.
- Incremental writes are not visible; the object can only be listed or read when the multipart operation completes in the close() call, which will block until the upload is completed.

Buffering uploads to disk or RAMs

This is the default buffer mechanism. The amount of data which can be buffered is limited by the amount of available disk space.

When fs.s3a.fast.upload.buffer is set to "disk", all data is buffered to local hard disks prior to upload. This minimizes the amount of memory consumed, and so eliminates heap size as the limiting factor in queued uploads.

Buffering uploads in Byte Buffers

When fs.s3a.fast.upload.buffer is set to "bytebuffer", all data is buffered in "direct" ByteBuffers prior to upload. This may be faster than buffering to disk in cases such as when disk space is small there may not be much disk space to buffer with (for example, when using "tiny" EC2 VMs).

The ByteBuffers are created in the memory of the JVM, but not in the Java Heap itself. The amount of data which can be buffered is limited by the Java runtime, the operating system, and, for YARN applications, the amount of memory requested for each container.

The slower the upload bandwidth to S3, the greater the risk of running out of memory — and so the more care is needed in tuning the upload thread settings to reduce the maximum amount of data which can be buffered awaiting upload.

Buffering Uploads with Array Buffers

When fs.s3a.fast.upload.buffer is set to "array", all data is buffered in byte arrays in the JVM's heap prior to upload. This may be faster than buffering to disk.

The amount of data which can be buffered is limited by the available size of the JVM heap heap. The slower the write bandwidth to S3, the greater the risk of heap overflows. This risk can be mitigated by tuning the upload thread settings.

Thread Tuning for S3A Data Upload

Both the array and bytebuffer buffer mechanisms can consume very large amounts of memory, on-heap or off-heap respectively. The disk buffer mechanism does not use much memory up, but it consumes hard disk capacity.

If there are many output streams being written to in a single process, the amount of memory or disk used is the multiple of all streams' active memory and disk use.

You may need to perform careful tuning to reduce the risk of running out memory, especially if the data is buffered in memory. There are a number parameters which can be tuned:

Parameter	Default Value	Description
fs.s3a.fast.upload.active.blocks	4	Maximum number of blocks a single output stream can have active (uploading, or queued to the central FileSystem instance's pool of queued operations). This stops a single stream overloading the shared thread pool.
fs.s3a.threads.max	10	The total number of threads available in the filesystem for data uploads or any other queued filesystem operation.
fs.s3a.max.total.tasks	5	The number of operations which can be queued for execution
fs.s3a.threads.keepalivetime	60	The number of seconds a thread can be idle before being terminated.

When the maximum allowed number of active blocks of a single stream is reached, no more blocks can be uploaded from that stream until one or more of those active block uploads completes. That is, a write() call which would trigger an upload of a now full datablock will instead block until there is capacity in the queue.

Consider the following:

- As the pool of threads set in fs.s3a.threads.max is shared (and intended to be used across all threads), a larger number here can allow for more parallel operations. However, as uploads require network bandwidth, adding more threads does not guarantee speedup.
- The extra queue of tasks for the thread pool (fs.s3a.max.total.tasks) covers all ongoing background S3A operations.
- When using memory buffering, a small value of fs.s3a.fast.upload.active.blocks limits the amount of memory which can be consumed per stream.
- When using disk buffering, a larger value of fs.s3a.fast.upload.active.blocks does not consume much memory. But it may result in a large number of blocks to compete with other filesystem operations.

We recommend a low value of fs.s3a.fast.upload.active.blocks — enough to start background upload without overloading other parts of the system. Then experiment to see if higher values deliver more throughput — especially from VMs running on EC2.

Optimizing S3A read performance for different file types

The S3A filesystem client supports the notion of input policies, similar to that of the POSIX fadvise() API call. This tunes the behavior of the S3A client to optimize HTTP GET requests for reading different filetypes. To optimize HTTP GET requests, you can take advantage of the S3A input policy option fs.s3a.experimental.input.fadvise.

Policy	Description
"normal"	This starts off as "sequential": it asks for the whole file. As soon as the application tries to seek backwards in the file it switches into "random" IO mode. This is not quite as efficient for Random IO as the "random" mode, because that first read may have to be aborted. However, because it is adaptive, it is the best choice if you do not know the data formats which will be read.
"sequential" (default)	Read through the file, possibly with some short forward seeks. The whole document is requested in a single HTTP request; forward seeks within the readahead range are supported by skipping over the intermediate data. This leads to maximum read throughput, but with very expensive backward seeks.
"random"	Optimized for random IO, specifically the Hadoop `PositionedReadable` operations — though `seek(offset); read(byte_buffer)` also benefits. Rather than ask for the whole file, the range of the HTTP request is set to that of the length of data desired in the `read` operation - rounded up to the readahead value set in `setReadahead()` if necessary. By reducing the cost of closing existing HTTP requests, this is highly efficient for file IO accessing a binary file through a series of PositionedReadable.read() and PositionedReadable.readFully() calls. Sequential reading of a file is expensive, as now many HTTP requests must be made to read through the file.

For operations simply reading through a file (copying, DistCp, reading gzip or other compressed formats, parsing .csv files, and so on) the sequential policy is appropriate. This is the default, so you do not need to configure it.

For the specific case of high-performance random access IO (for example, accessing ORC files), you may consider using the random policy in the following circumstances:

- Data is read using the PositionedReadable API.
- There are long distance (many MB) forward seeks.
- Backward seeks are as likely as forward seeks.
- There is little or no use of single character read() calls or small read(buffer) calls.
- Applications are running close to the Amazon S3 data store; that is, the EC2 VMs on which the applications run are in the same region as the Amazon S3 bucket.

You must set the desired fadvise policy in the configuration option fs.s3a.experimental.input.fadvise when the filesystem instance is created. It can only be set on a per-filesystem basis, not on a per-file-read basis. You can set it in core-site.xml:

```
<property>
  <name>fs.s3a.experimental.input.fadvise</name>
  <value>random</value>
</property>
```

Or, you can set it in the spark-defaults.conf configuration of Spark:

```
spark.hadoop.fs.s3a.experimental.input.fadvise random
```

Be aware that this random access performance comes at the expense of sequential IO — which includes reading files compressed with gzip.

Improving S3A read performance using Vectored IO

The Hadoop Vectored IO API allows file formats like ORC and Parquet to fetch a set of data ranges in a single operation instead of issuing individual read calls for each range.

The Hadoop Vectored IO is an asynchronous API that enables the libraries to perform other tasks while waiting for the data. Different implementations of the Hadoop Vectored IO can support additional optimizations such as merging close-by data ranges and fetching data ranges in parallel from remote cloud storage. This results in a faster and more efficient data retrieval from cloud storage. The S3A connector offers a customized implementation that enables parallel and asynchronous reading of different data blocks.

You can enable Hadoop Vectored IO using hive.exec.orc.use.hadoop-vectored.api=true for Hive on ORC queries i, and parquet.hadoop.vectored.io.enabled=true for Spark on Parquet queries.

The S3A filesystem supports implementation of readVectored API using the client to provide a list of file ranges to read, which returns a future read object associated with each range. For more information about the readVectored API, see the *FSDataInputStream* specification and specification and *Hadoop Vectored IO: your data just got faster!* article.

The following configurations can be used to optimize vectored reads based on the client requirements:

Property	Default value	Description
fs.s3a.vectored.read.min.seek.size	4K	Smallest reasonable seek in bytes to group ranges together during vectored read operation.
fs.s3a.vectored.read.max.merged.size	1M	Largest merged read size in bytes to group ranges together during vectored read. Setting this value to 0 disables merging of ranges.
fs.s3a.vectored.active.ranged.reads	4	Maximum number of range reads a single input stream can have active (downloading, or queued) to the central FileSystem instance's pool of queued operations. This stops a single stream overloading the shared thread pool

Related Information

FSDataInputStream

Hadoop Vectored IO: your data just got faster!

S3 Performance Checklist

There are various types of checklists that you can use to ensure optimal performance when working with data in S3.

Checklist for Data

- [ ] Amazon S3 bucket is in same region as the EC2-hosted cluster.
- [ ] The directory layout is "shallow". For directory listing performance, the directory layout prefers "shallow" directory trees with many files over deep directory trees with only a few files per directory.
- [ ] The "pseudo" block size set in fs.s3a.block.size is appropriate for the work to be performed on the data.
- [ ] Copy to HDFS any data that needs to be repeatedly read to HDFS.

Checklist for Cluster Configurations

- [ ] Set yarn.scheduler.capacity.node-locality-delay to 0 to improve container launch times.

Checklist for Code

- [ ] Application does not make rename() calls. Where it does, it does not assume the operation is immediate.
- [ ] Application does not assume that delete() is near-instantaneous.
- [ ] Application uses FileSystem.listFiles(path, recursive=true) to list a directory tree.
- [ ] Application prefers forward seeks through files, rather than full random IO.
- [ ] If making "random" IO through seek() and read() sequences or and Hadoop's PositionedReadable API, fs.s3a.experimental.input.fadvise is set to random.

Troubleshooting S3

It can be a bit troublesome to get the S3A connector to work, with classpath and authentication being the usual trouble spots. Use the tips provided in the following links to troubleshoot errors.

S3: https://aws.amazon.com/premiumsupport/knowledge-center/

Related Information

Encrypting Data on S3

Working with Google Cloud Storage

The Cloudera platform allows you to configure access from a cluster to Google Cloud Storage (GCS) in order to access cloud data.

Configuring Access to Google Cloud Storage

Access to Google Cloud Storage can be configured separately for each cluster by providing the service account email address.

Access from a cluster to a Google Cloud Storage is possible through a service account. Configuring access to Google Cloud Storage involves the following steps.

Table 3: Overview of Configuring Access to Google Cloud Storage

Step	Considerations
Creating a service account on Google Cloud Platform and generating a key associated with it.	• You may need to contact your Google Cloud Platform admin in order to complete these steps. • If you already have a service account, you do not need to perform these steps as long as you are able to provide the service account key. If you have a service account but do not know the service account key, you should be able to generate a new key.

Step	Considerations
Creating a role on Google Cloud Platform with sufficient permissions to access storage buckets.	• You may need to contact your Google Cloud Platform admin in order to complete these steps. • This is a one time operation, and the same role can be used across different service accounts and storage buckets.
Modifying permissions of the Google Cloud Storage bucket so that you can access it by using your service account key.	• You may need to contact your Google Cloud Platform admin in order to complete these steps. • \You should perform these steps for each bucket that you want to access. • You do not need to perform these steps if your service account has project-wide access to all buckets on the account.
Configuring credentials via Cloudera Manager.	• These configuration steps are appropriate for a single-user cluster. • Only one configuration per cluster is recommended; that is, you should use one service account per cluster. If required, it is possible to use multiple service accounts with the same cluster; in that case, each job-specific configuration should be changed to use the desired service account.

Create a GCP Service Account

You must create a Google Cloud Platform service account and generate an access key (in the JSON format). If you are using a corporate GCP account, it is likely that only your GCP admin can perform these steps. Example steps are described below.

These steps assume that you have a Google Cloud Platform account. If you do not have one, you can create it at https://console.cloud.google.com.

Steps

1. In the Google Cloud Platform web console, navigate to IAM & admin Service accounts:
2. Click +Create Service Account.

3. Provide the following information:
   a. Under Service account name, provide some name for your service account.
   b. Under Role, select the project-level roles that the account should have.
   c. Check Furnish a new private key and select JSON.

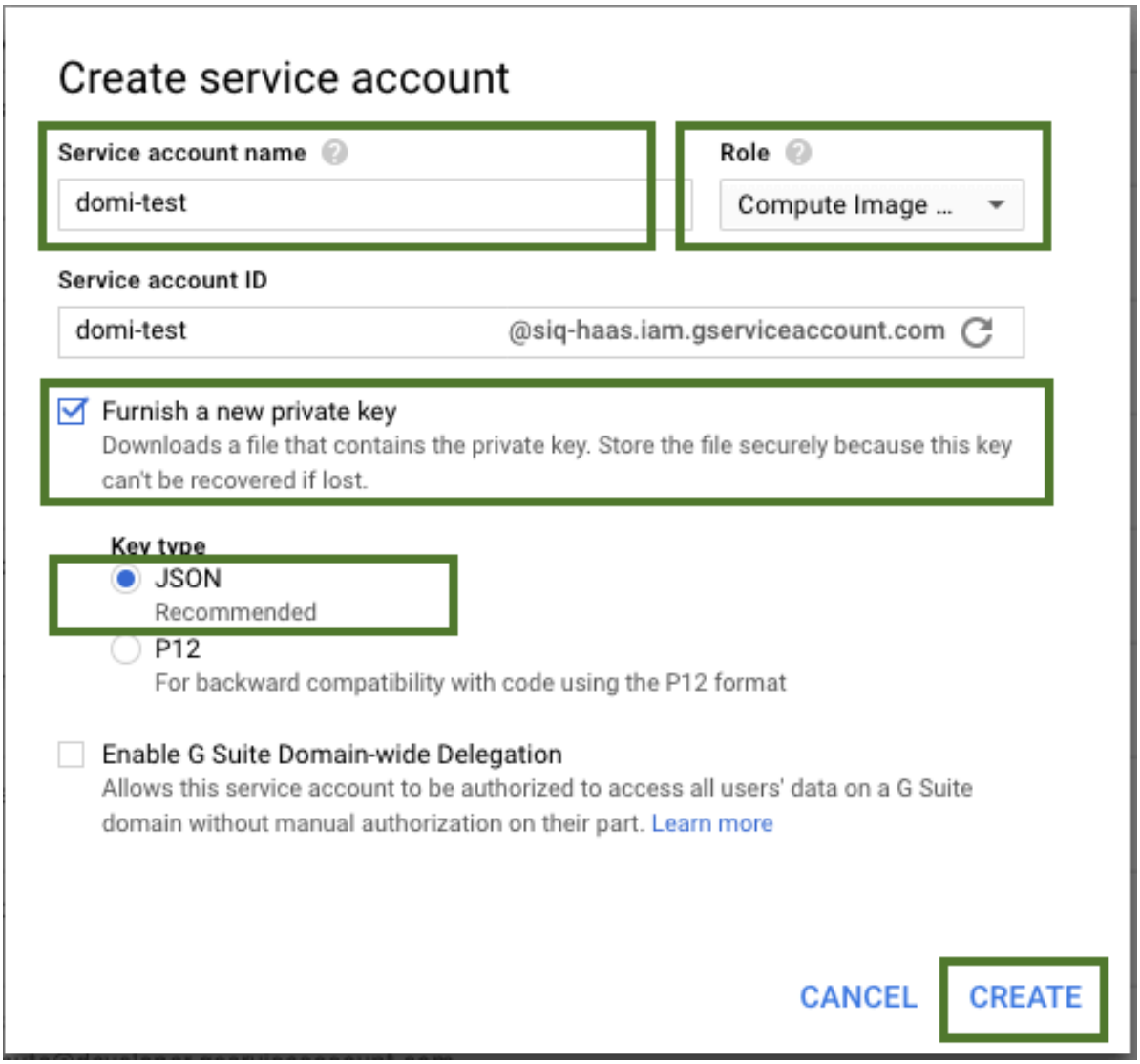

4. Click Create and the file containing the key will be downloaded onto your machine. The name of the key file is usually long and contains spaces, so you may want to rename the file.

Later you will need to place this key on your cluster nodes.

Create a Custom Role

Specific permissions are required for the Google Cloud Storage Connector to access buckets. This set of permissions is the combination of the permissions associated with the existing Google Cloud IAM Role called "Storage Object Admin" and the Google Cloud IAM Permission called "storage.buckets.get".

1. In the Google Cloud Platform web console, navigate to IAM & admin > Roles
2. Click on +Create Role.
3. Provide the following:
   - Enter a title under Title
   - Enter the ID under ID
   - Under Role launch stage select General Availability

4. Click on Add Permissions and add the following permissions:
   - storage.bucket.get
   - storage.objects.create
   - storage.objects.delete
   - storage.objects.get
   - storage.objects.getIamPolicy
   - storage.objects.list
   - storage.objects.setIamPolicy
   - storage.objects.update

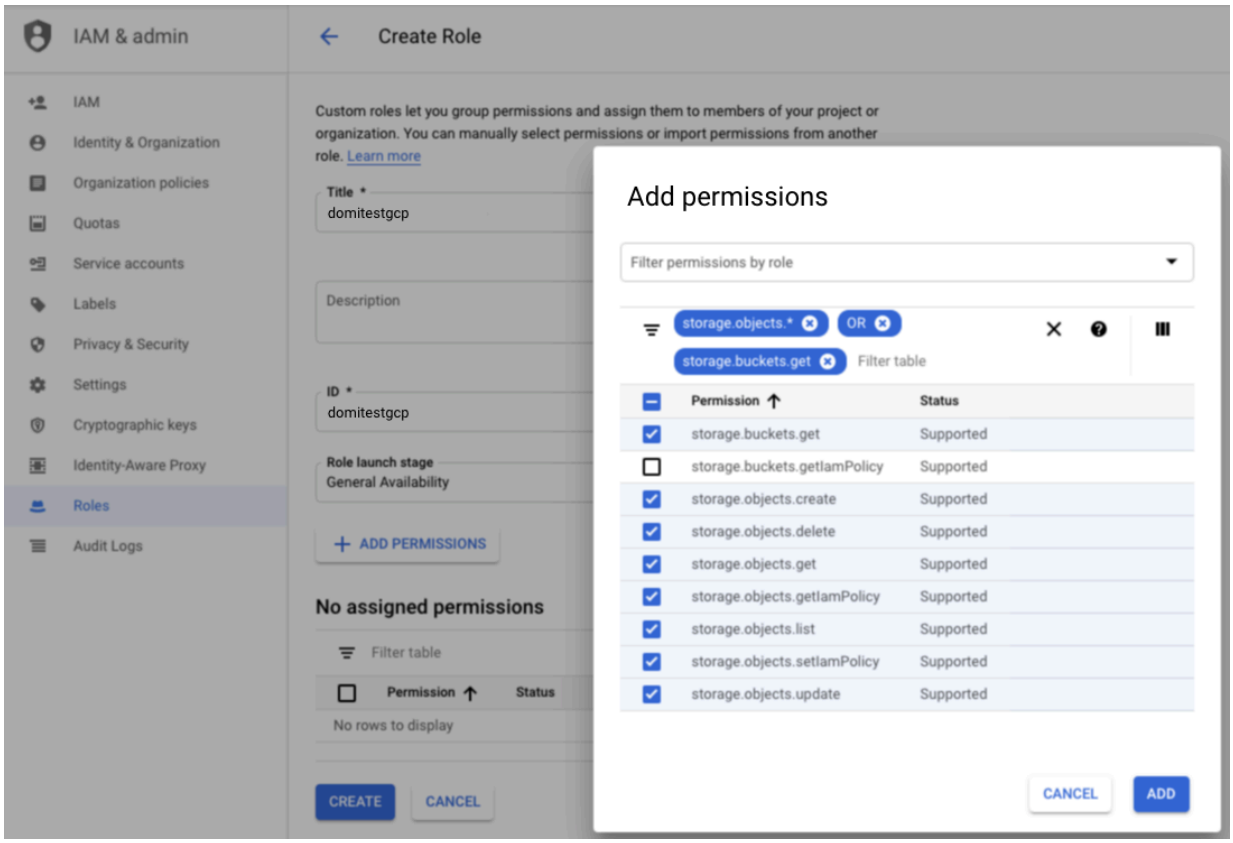

5. Once done adding permissions, click on Create.

After performing these steps, a new role will be created.

Modify GCS Bucket Permissions

You or your GCP admin must set the bucket permissions so that your service account has access to the bucket that you want to access from the cluster. The IAM role created in the previous step, is the minimum role required to access the cluster. Example steps are described below.

1. In the Google Cloud Platform web console, navigate to Storage > Browser.
2. Find the bucket for which you want to edit permissions.
3. Click the ⋮ and select Edit bucket permissions:
4. In the Permissions tab set the bucket-level permissions:
   - Click on Add members and enter the service account created earlier.
   - Under Roles, select the IAM role created in the previous step. The role should be available under Custom.

5. When done, click Add.

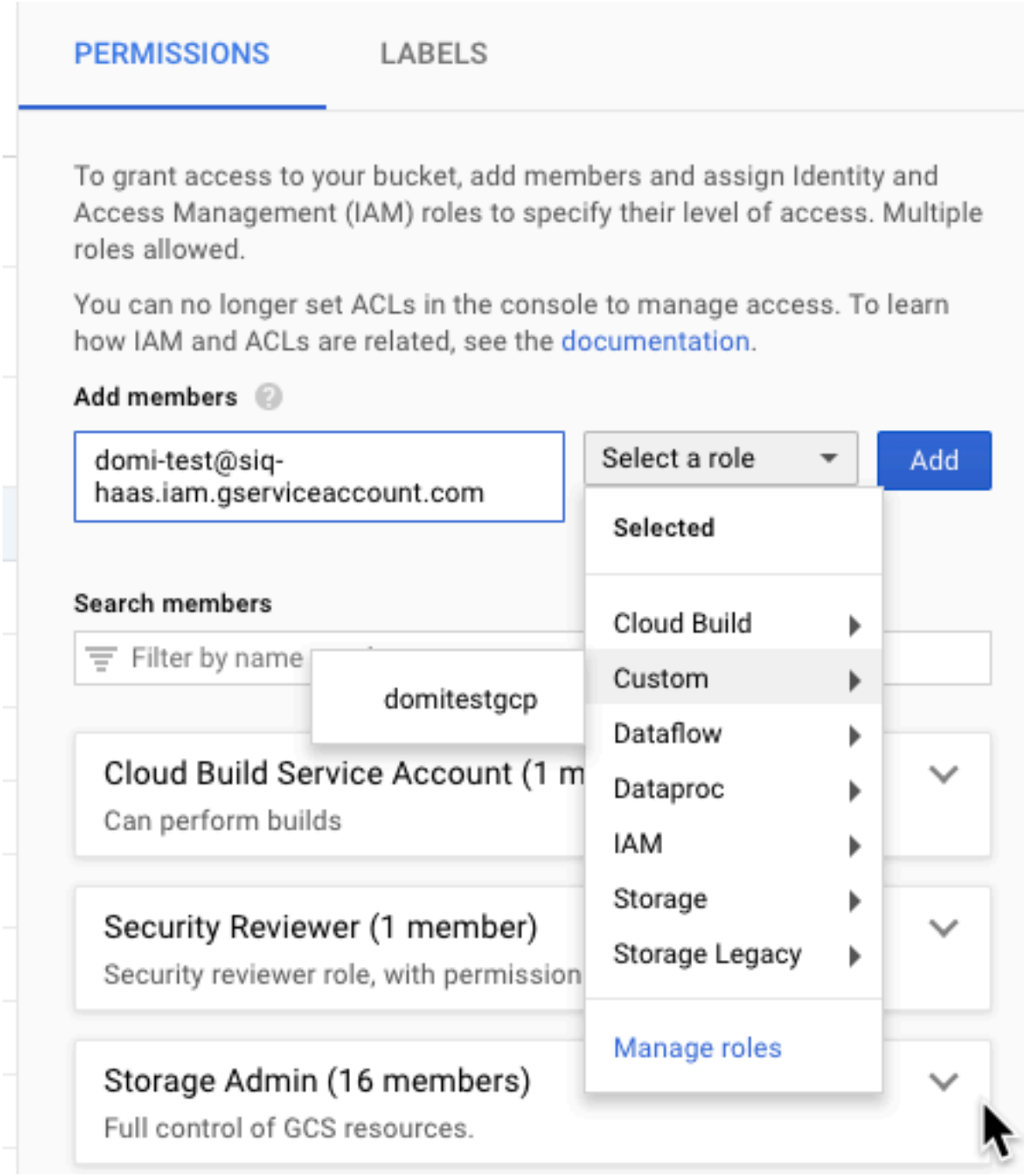

After performing these steps, the bucket-level permissions will be updated.

Configure Access to GCS from Your Cluster

After obtaining the service account key, perform these steps on your cluster. The steps below assume that your service account key is called google-access-key.json. If you choose a different name, make sure to update the commands accordingly.

1. In Cloudera Manager UI, set the following three properties under hdfs core-site.xml. Navigate to Clusters > HDFS > Configuration > Advanced > Add property under Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml.
   - Set the following properties: fs.gs.auth.service.account.email=<VALUE1> fs.gs.auth.service.account.private.key.id=<VALUE2> fs.gs.auth.service.account.private.key=<VALUE3> You can obtain the values for these parameters as follows:

Parameter	Value
fs.gs.auth.service.account.email	The client_email field extracted from the credential's JSON file.
fs.gs.auth.service.account.private.key.id	The private_key_id field extracted from the credential's JSON file.
fs.gs.auth.service.account.private.key	The private_keyfield extracted from the credential's JSON file.

     The JSON key file downloaded in the previous step is in plain text. Open the file in your favorite text editor to extract the relevant values needed in the above configuration.

     Note: For enhanced security, these values can also be configured using Hadoop CredentialProvider.

   - Configure the following properties if they're not already set by Cloudera Manager fs.gs.working.dir=/ fs.gs.path.encoding=uri-path

     Note: Setting fs.gs.working.dir configures the initial working directory of a GHFS instance. This should always be set to "/". Setting fs.gs.path.encoding sets the path encoding to be used, and allows for spaces in the filename. This should always be set to "uri-path".
2. Save the configuration change and restart affected services. Additionally - depending on what services you are using - you must restart other services that access cloud storage such as Spark Thrift Server, HiveServer2, and Hive Metastore; These will not be listed as affected by Cloudera Manager, but require a restart to pick up the configuration changes.
3. Test access to the Google Cloud Storage bucket by running a few commands from any cluster node. For example, you can use the command listed below (replace “mytestbucket” with the name of your bucket): hadoop fs -ls gs://mytestbucket/

After performing these steps, you should be able to start working with the Google Cloud Storage bucket(s).

Manifest committer for ABFS and GCS

The Intermediate Manifest committer is a high performance committer for Spark work that provides performance on ABFS for real world queries, and performance and correctness on GCS. It also works with other filesystems, including HDFS. However, the design is optimized for object stores where listing operatons are slow and expensive.

The v1/v2 file committers

The only committer of work from Spark to Azure ADLS Gen 2 "abfs://" storage that is safe to use is the "v1 file committer."

This is "correct" in that if a task attempt fails, its output is guaranteed not to be included in the final out. The "v2" commit algorithm cannot meet that guarantee, which is why it is no longer the default.

But the v1 file committer is slow, especially on jobs where deep directory trees of output are used. This is due to lack of any instrumentation in the FileOutputCommitter. Stack traces of running jobs generally show rename(), though list operations do surface too.

On GCS, neither the v1 nor v2 algorithm are safe because the Google filesystem doesn't have the atomic directory rename which the v1 algorithm requires.

A further issue is that both Azure and GCS storage may encounter scale issues with deleting directories with many descendants. This can trigger timeouts because the FileOutputCommitter assumes that cleaning up after the job is a fast call to delete("_temporary", true).

The manifest committer

The Manifest Committer is a higher performance committer for ABFS and GCS storage for jobs that create files across deep directory trees through many tasks. It will also work on hdfs:// and file:// URLs, but it is optimized to address listing and renaming performance and throttling issues in cloud storage.

It will not work correctly with S3, because it relies on an atomic rename-no-overwrite operation to commit the manifest file. It will also have the performance problems of copying rather than moving all the generated data.

Although it will work with MapReduce, there is no handling of multiple job attempts with recovery from previous failed attempts.

This committer uses the extension point which came in for the S3A committers. Users can declare a new committer factory for abfs:// and gcs:// URLs. A suitably configured Spark deployment will pick up the new committer.

Using the manifest committer

Define a factory for the ABFS or GCS schema, then bind to the manifest committer in Spark.

Using the committer

Define a factory for the ABFS schema in mapreduce.outputcommitter.factory.scheme.abfs or mapreduce.outputcommitter.factory.scheme.gs for GCS.

Some matching spark configuration changes, especially for parquet binding, will be required. These can be done in core-site.xml, if it is not defined in the mapred-default.xml JAR.

```
<property>
  <name>mapreduce.outputcommitter.factory.scheme.abfs</name>
  <value>org.apache.hadoop.fs.azurebfs.commit.AzureManifestCommitterFactory
</value>
</property>
<property>
  <name>mapreduce.outputcommitter.factory.scheme.gs</name>
  <value>org.apache.hadoop.mapreduce.lib.output.committer.manifest.Manife
stCommitterFactory</value>
</property>
```

Binding to the manifest committer in Spark

In Apache Spark, set the configuration either with command line options (after the '--conf') or by using the spark-defaults.conf file. The following is an example of using spark-defaults.conf, and also including the configuration for Parquet with a subclass of the parquet committer which uses the factory mechansim internally.

```
spark.hadoop.mapreduce.outputcommitter.factory.scheme.abfs org.apache.hadoop
.fs.azurebfs.commit.AzureManifestCommitterFactory
spark.hadoop.mapreduce.outputcommitter.factory.scheme.gs org.apache.hadoop
.mapreduce.lib.output.committer.manifest.ManifestCommitterFactory
spark.sql.parquet.output.committer.class org.apache.spark.internal.io.clou
d.BindingParquetOutputCommitter
spark.sql.sources.commitProtocolClass org.apache.spark.internal.io.cloud.Pat
hOutputCommitProtocol
```

Using the Cloudstore committerinfo command to probe committer bindings

The hadoop committer settings can be validated in a recent build of cloudstore and its committerinfo command. This command instantiates a committer for that path through the same factory mechanism as MR and spark jobs use, then prints its toString value.

```
hadoop jar cloudstore-1.0.jar committerinfo abfs://testing@ukwest.dfs.core.w
indows.net/
2021-09-16 19:42:59,731 [main] INFO  commands.CommitterInfo (StoreDurationI
nfo.java:<init>(53)) - Starting: Create committer
Committer factory for path abfs://testing@ukwest.dfs.core.windows.net/ is
 org.apache.hadoop.mapreduce.lib.output.committer.manifest.ManifestCommitter
Factory@3315d2d7
  (classname org.apache.hadoop.mapreduce.lib.output.committer.manifest.Mani
festCommitterFactory)
2021-09-16 19:43:00,897 [main] INFO  manifest.ManifestCommitter (ManifestC
ommitter.java:<init>(144)) - Created ManifestCommitter with
   JobID job__0000, Task Attempt attempt__0000_r_000000_1 and destination ab
fs://testing@ukwest.dfs.core.windows.net/
Created committer of class org.apache.hadoop.mapreduce.lib.output.committe
r.manifest.ManifestCommitter:
 ManifestCommitter{ManifestCommitterConfig{destinationDir=abfs://testing@u
kwest.dfs.core.windows.net/,
   role='task committer',
   taskAttemptDir=abfs://testing@ukwest.dfs.core.windows.net/_temporary/m
anifest_job__0000/0/_temporary/attempt__0000_r_000000_1,
   createJobMarker=true,
   jobUniqueId='job__0000',
   jobUniqueIdSource='JobID',
   jobAttemptNumber=0,
   jobAttemptId='job__0000_0',
   taskId='task__0000_r_000000',
   taskAttemptId='attempt__0000_r_000000_1'},
   iostatistics=counters=();

gauges=();

minimums=();
maximums=();

means=();
}
```

Verifying that the committer was used

The new committer will write a JSON summary of the operation, including statistics, in the _SUCCESS file.

If this file exists and is zero bytes long: the classic FileOutputCommitter was used.

If this file exists and is greater than zero bytes long, either the manifest committer was used, or in the case of S3A filesystems, one of the S3A committers. They all use the same JSON format.

Configuration options

Option	Meaning	Default value
mapreduce.manifest.committer.delete.target.files	Delete target files?	false
mapreduce.manifest.committer.io.threads	Thread count for parallel operations	64
mapreduce.manifest.committer.summary.report.directory	Directory to save reports	""

Option	Meaning	Default value
mapreduce.manifest.committer.cleanup.parallel.delete	Delete temporary directories in parallel	true
mapreduce.fileoutputcommitter.cleanup.skipped	Skip cleanup of _temporary directory	false
mapreduce.fileoutputcommitter.cleanup-failures.ignored	Ignore errors during cleanup	false
mapreduce.fileoutputcommitter.marksuccessfuljobs	Create a _SUCCESS marker file on successful completion (and delete any existing one in job setup)	true

Scaling jobs mapreduce.manifest.committer.io.threads

The core reason that the manifest committer is faster than the classic FileOutputCommitter is that it tries to parallelize as much file IO as it can during job commit, specifically:

- Task manifest loading
- Deletion of files where directories will be created
- Directory creation
- File-by-file renaming
- Deletion of task attempt directories in job cleanup

These operations are all performed in the same thread pool, whose size is set in the option mapreduce.manifest.committer.io.threads.

Larger values may be used.

XML:

```
<property>
  <name>mapreduce.manifest.committer.io.threads</name>
  <value>200</value>
</property>
```

spark-defaults.conf:

```
spark.hadoop.mapreduce.manifest.committer.io.threads 200
```

A larger value than that of the number of cores allocated to the MapReduce AM or Spark Driver does not directly overload the CPUs, as the threads are normally waiting for (slow) IO against the object store/filesystem to complete.

Caveats:

- In Spark, multiple jobs may be committed in the same process, each of which will create their own thread pool during job commit or cleanup.
- Azure rate throttling may be triggered if too many IO requests are made against the store. The rate throttling option mapreduce.manifest.committer.io.rate can help avoid this.

Optional: deleting target files in Job Commit

The classic FileOutputCommitter deletes files at the destination paths before renaming the job's files into place.

This is optional in the manifest committers, set in the option mapreduce.manifest.committer.delete.target.files with a default value of false. This increases performance and is safe to use when all files created by a job have unique filenames.

Apache Spark does generate unique filenames for ORC and Parquet since SPARK-8406 Adding UUID to output file name to avoid accidental overwriting.

Avoiding checks for/deleting target files saves one delete call per file being committed, so can save a significant amount of store IO.

When appending to existing tables, using formats other than ORC and Parquet, unless confident that unique identifiers are added to each filename, enable deletion of the target files.

```
spark.hadoop.mapreduce.manifest.committer.delete.target.files true
```

Note: The committer will skip deletion operations when it created the directory into which a file is to be renamed. This makes it slightly more efficient, at least if jobs appending data are creating and writing into new partitions.

Note: The committer still requires tasks within a single job to create unique files. This is foundational for any job to generate correct data.

Spark Dynamic Partition overwriting

Spark has a feature called “Dynamic Partition Overwrites” that can be initiated in SQL:

```
INSERT OVERWRITE TABLE ...
```

Or through DataSet writes where the mode is overwrite and the partitioning matches that of the existing table:

```
sparkConf.set("spark.sql.sources.partitionOverwriteMode", "dynamic")
// followed by an overwrite of a Dataset into an existing partitioned table.
eventData2
  .write
  .mode("overwrite")
  .partitionBy("year", "month")
  .format("parquet")
  .save(existingDir)
```

This Spark feature:

- Directs the job to write its new data to a temporary directory
- After job commit completes, scans the output to identify the leaf directories “partitions” into which data was written
- Deletes the content of those directories in the destination table
- Renames the new files into the partitions.

This is all done in Spark, which takes over the tasks of scanning the intermediate output tree, deleting partitions and of renaming the new files.

This feature also adds the ability for a job to write data entirely outside the destination table, which is done by writing new files into the working directory, and then Spark moving them to the final destination in job commit.

The manifest committer is compatible with dynamic partition overwrites on Azure and Google Cloud Storage as together they meet the core requirements of the extension:

- The working directory returned in getWorkPath() is in the same filesystem as the final output.
- rename() is an O(1) operation which is safe and fast to use when committing a job.

None of the S3A committers support this. The first condition is not met by the staging committers, while the second condition is not met by S3 itself.

To use the manifest committer with dynamic partition overwrites, the Spark version must contain SPARK-40034 PathOutputCommitters to work with dynamic partition overwrite.

Be aware that the rename phase of the operation will be slow if many files are renamed--this is done sequentially. Parallel renaming would speed this up, but could trigger the ABFS overload problems the manifest committer is designed to both minimize the risk of and support recovery from.

The Spark side of the commit operation will be listing/treewalking the temporary output directory (some overhead), followed by the file promotion, done with a classic filesystem rename() call. There will be no explicit rate limiting here.

This means that _dynamic partitioning should not be used on Azure Storage for SQL queries/Spark DataSet operations where many thousands of files are created. The fact that these will suffer from performance problems before throttling scale issues surface, should be considered a warning.

Job summaries in _SUCCESS files

You can view and collect job summaries in the _SUCCESS files.

_SUCCESS files

The original Hadoop committer creates a zero byte _SUCCESS file in the root of the output directory unless disabled.

The manifest committer writes a JSON summary which includes:

- The name of the committer.
- Diagnostics information.
- A list of some of the files created (for testing; a full list is excluded as it can get big).
- IO Statistics.

If, after running a query, this _SUCCESS file is zero bytes long, the manifest committer has not been used.

If it is not empty, then it can be examined.

Viewing _SUCCESS file files through the ManifestPrinter tool

The summary files are JSON, and can be viewed in any text editor.

For a more succinct summary, including better display of statistics, use the ManifestPrinter tool.

```
hadoop org.apache.hadoop.mapreduce.lib.output.committer.manifest.files.Manif
estPrinter <path>
```

This works for the files saved at the base of an output directory, and any reports saved to a report directory.

Collecting Job Summaries

The committer can be configured to save the _SUCCESS summary files to a report directory, irrespective of whether the job succeed or failed, by setting a fileystem path in the option mapreduce.manifest.committer.summary.report.dire ctory.

The path does not have to be on the same store/filesystem as the destination of work. For example, a local fileystem could be used.

XML:

```
<property>
  <name>mapreduce.manifest.committer.summary.report.directory</name>
  <value>file:///tmp/reports</value>
</property>
```

spark-defaults.conf:

```
spark.hadoop.mapreduce.manifest.committer.summary.report.directory file:///t
mp/reports
```

This allows for the statistics of jobs to be collected irrespective of their outcome, whether or not saving the _SUC CESS marker is enabled, and without problems caused by a chain of queries overwriting the markers.

Job cleanup

Job cleanup is designed to address a number of issues which may surface in cloud storage:

- Slow performance for deletion of directories.
- Timeout when deleting very deep and wide directory trees.
- General resilience to cleanup issues escalating to job failures.

Option	Meaning	Default Value
mapreduce.fileoutputcommitter.cleanup.sk ipped	Skip cleanup of _temporary directory	false
mapreduce.fileoutputcommitter.cleanup-failur es.ignored	Ignore errors during cleanup	false
mapreduce.manifest.committer.cleanup.paralle l.delete	Delete task attempt directories in parallel	true

The algorithm is:

```
if `mapreduce.fileoutputcommitter.cleanup.skipped`:
  return
if `mapreduce.manifest.committer.cleanup.parallel.delete`:
  attempt parallel delete of task directories; catch any exception
if not `mapreduce.fileoutputcommitter.cleanup.skipped`:
  delete(`_temporary`); catch any exception
if caught-exception and not `mapreduce.fileoutputcommitter.cleanup-failures.
ignored`:
  throw caught-exception
```

The goal is to perform a fast/scalable delete and throw a meaningful exception if that didn't work.

When working with ABFS and GCS, these settings should normally be left alone. If somehow errors surface during cleanup, enabling the option to ignore failures will ensure the job still completes. Disabling cleanup even avoids the overhead of cleanup, but requires a workflow or manual operation to clean up all _temporary directories on a regular basis.

Working with Google Cloud Storage

The manifest committer is compatible with and tested against Google Cloud Storage through the gcs-connector library from Google, which provides a Hadoop filesystem client for the schema gs.

Google Cloud Storage has the semantics needed for the commit protocol to work safely.

The Spark settings to switch to this committer are:

```
spark.hadoop.mapreduce.outputcommitter.factory.scheme.gs org.apache.had
oop.mapreduce.lib.output.committer.manifest.ManifestCommitterFactory
spark.sql.parquet.output.committer.class org.apache.spark.internal.io.c
loud.BindingParquetOutputCommitter
spark.sql.sources.commitProtocolClass org.apache.spark.internal.io.cloud.
PathOutputCommitProtocol
```

```
spark.hadoop.mapreduce.manifest.committer.summary.report.directory  (optio
nal: URI of a directory for job summaries)
```

The store's directory delete operations are O(files) so the value of mapreduce.manifest.committer.cleanup.parallel.delete should be left at the default of true.

For MapReduce, declare the binding in core-site.xmlor mapred-site.xml:

```
<property>
  <name>mapreduce.outputcommitter.factory.scheme.gcs</name>
  <value>org.apache.hadoop.mapreduce.lib.output.committer.manifest.Manife
stCommitterFactory</value>
</property>
```

Advanced topics

Advanced topics and configuration options for the manifest committer.

Advanced configuration

There are some advanced options which are intended for development and testing, rather than production use.

Option	Meaning	Default Value
mapreduce.manifest.committer.store.operations.classname	Classname for Manifest Store Operations	""
mapreduce.manifest.committer.validate.output	Perform output validation?	false

Validating output

The option mapreduce.manifest.committer.validate.output triggers a check of every renamed file to verify it has the expected length.

This adds the overhead of a HEAD request per file, and so is recommended for testing only.

There is no verification of the actual contents.

Controlling storage integration

The manifest committer interacts with filesystems through implementations of the interface ManifestStoreOperations. It is possible to provide custom implementations for store-specific features. There is one of these for ABFS; when the abfs-specific committer factory is used this is automatically set.

It can be explicitly set:

```
<property>
  <name>mapreduce.manifest.committer.store.operations.classname</name>
  <value>org.apache.hadoop.fs.azurebfs.commit.AbfsManifestStoreOperations</
value>
</property>
```

The default implementation may also be configured:

```
<property>
  <name>mapreduce.manifest.committer.store.operations.classname</name>
  <value>org.apache.hadoop.mapreduce.lib.output.committer.manifest.impl.Ma
nifestStoreOperationsThroughFileSystem</value>
</property>
```

There is no need to alter these values, except when writing new implementations for other stores, something which is only needed if the store provides extra integration support for the committer.

Additional Configuration Options for GCS

You may optionally set the following properties related to Google Cloud Storage.

Table 4: Optional Properties Related to Google Cloud Storage

Property	Description
fs.gs.project.id	Allows you to enter Google Cloud's Project ID with access to configured GCS buckets. By default, this option is unset.
fs.gs.block.size	This does not have any effect on storage but allows you to change the way split sizes are computed. The value should be in bytes. We recommend that you set this to the HDFS block size on the cluster, or 134217728 for a block size of 128MB.

Working with the ABFS Connector

The Cloudera platform allows you to configure access from a cluster to Azure in order to access cloud data.

Introduction to Azure Storage and the ABFS Connector

The Hadoop-Azure module provides support for Azure Data Lake Storage Gen2 storage layer through the `abfs` connector.

#Azure Data Lake Storage (ADLS) Gen2 combines the features of `Azure Blob storage` and `Azure Data Lake Storage Gen1.` In addition to the existing features of both the services, an important part of Azure Data Lake Storage Gen2 is the addition of hierarchical namespace to Blob storage. The hierarchical namespace feature processes operations such as moving, renaming, and deletings directories by updating a single entry (the parent directory). This ensures a significant improvement in performance of query engines writing data to the store, including MapReduce, Spark, Hive, and DistCp. You can also set permissions on a directory instead of per file basis.

Note: Hierarchical Namespace feature is available only if the container is created with the namespace support. You cannot enable Hierarchical Namespaces on an existing storage account.

Azure Blob Storage

The Azure Storage data model presents three core concepts:

- Storage Account: All access is done through a storage account.
- Container: A container is a grouping of multiple blobs. A storage account may have multiple containers. In Hadoop, an entire file system hierarchy is stored in a single container.
- Blob: A file of any type and size stored with the existing Windows Azure Storage Blob (wasb) connector

Features of the ABFS Connector

The ABFS connector can be used as a replacement for HDFS on Hadoop clusters deployed in Azure infrastructure. Some of the features of the ABFS connector are as follows:

- Supports reading and writing data stored in an Azure Blob Storage account
- Helps to perform hadoop operations on Azure datastores
- Provides a consistent view of the storage across all clients
- Enables using a simple abfs:// url to access containers and directories

- Presents a hierarchical file system view by implementing the standard Hadoop File System interface
- Supports configuration of multiple Azure Blob Storage accounts
- Acts as a source or destination of data in Hadoop MapReduce, Apache Hive, Apache Spark

Feature Comparisons

ADLS Gen 2 is where all future development of Azure Big Data storage is taking place, in the cloud server as well as the client connector. In contrast, ADLS Gen 1 must be considered a `maintenance only` data store, which is not being rolled out across more Azure regions.

Feature	Azure Storage	ADLS Gen 1	ADLS Gen 2
Broadly supported in applications	Yes	No	Yes in non-hierarchical; growing in hierarchical
Scales to many petabytes	No	Yes	Yes
Available in all Azure region	Yes	No	Yes
Interoperability	ADLS Gen 2 in non-hierarchical	none	Azure Storage
Directories with permissions	No	Yes	Yes

Differences between ADLS Gen 1 and ADLS Gen 2

ADLS Gen 2 should be considered as a reimplementation of the ADLS Gen 1 features, but integrated with the original Azure Storage.

Features of ADLS Gen 2

- Supports the Hadoop FileSystem API, with directories, file and directory permissions, and other key features.
- Reads and writes data stores in the original Azure Storage (which has previously used the wasb:// URL)
- No integration with ADLS Gen1. The adls:// connector must be used there.
- Available in all Azure regions.
- If an ADLS Gen 2 account is created "without hierarchical namespaces" then the wasb:// connector can read/write data stored in ADLS Gen 2.
- Capable of storing many Petabytes of data.

Setting up and configuring the ABFS connector

You must have an Azure storage account with hierarchical namespace enabled and Azure container prior to configuring the ABFS connector.

About this task

You can use the following steps to create a new storage account, create a new container, and configure the ABFS connector:

Procedure

1. Create Azure storage account

   Every storage account must belong to an Azure resource group. A resource group is a logical container for grouping your Azure services. When you create a storage account, you have the option to either create a new resource group, or use an existing resource group.

   You can create an Azure storage account using Azure CLI or Azure portal. When creating a Storage Account, you must ensure to enable namespace support by selecting the `Hierarchical Namespace` option. For information on creating an Azure storage account, see Create a storage account.

2. Create Azure Container

   You must create a container to organize the blobs. One Azure storage account can have multiple containers, each with the container name as the user information field of the URI used to reference it.

   For example, the container `container1` in the storage account will have the URL abfs://container1@<storage_account>.dfs.core.windows.net/ .

   For information on creating a container in Azure, see Create a container.

3. Configure ABFS

   You can configure access credentials to authorise access to Azure containers. For information on configuring ABFS, see Configuring ABFS.

   Note: Delete Azure storage account - Deleting a storage account deletes the entire account, including all data in the account, and cannot be undone. For information about deleting an Azure storage account, see Delete a storage account.

Configuring the ABFS Connector

You can configure access credentials to authorise access to Azure containers in multiple ways including IDBroker, Shared Key, Managed Instance, and Shared Access Signature.

Authenticating with ADLS Gen2

Authentication for ABFS is granted by Azure Active Directory. The authentication mechanism is set using the `fs.azure.account.auth.type` (or the account specific variant) property in the core-site.xml file.

Configuring Access to Azure on Cloudera Public Cloud

IDBroker is a REST API built as part of Apache Knox's authentication services. It allows an authenticated user to exchange a set of credentials or a token for cloud vendor access tokens. IDBroker manages mapping LDAP users to FreeIPA cloud identities for data access. It performs identity mapping for access to object stores.

For information on how IDBroker works in Cloudera, see ADLS Gen2 and managed identities in the *Cloudera Management Console* documentation.

Configuring Access to Azure on Cloudera Private Cloud Base

You can use Shared Key, Managed Instance, Shared Access Signature, and so on to configure access credentials to authorise access to Azure containers. For example, a shared key account can be configured for use regardless of which account is accessed with the property `fs.azure.account.key` or you can configure an identity to be used only for a specific storage account with `fs.azure.account.key.<BUCKET NAME>.blob.core.windows.net`.

Before you begin

After you create a storage account, Azure generates two 512-bit storage account access keys. You can use these keys to authorize access to data in your storage account. For information on how to access this Shared Key, see View account access keys. To configure access to the Azure container, add the Shared Key details in the core-site.xml file using Cloudera Manager.

Procedure

1. In Cloudera Manager, select Clusters > HDFS service.

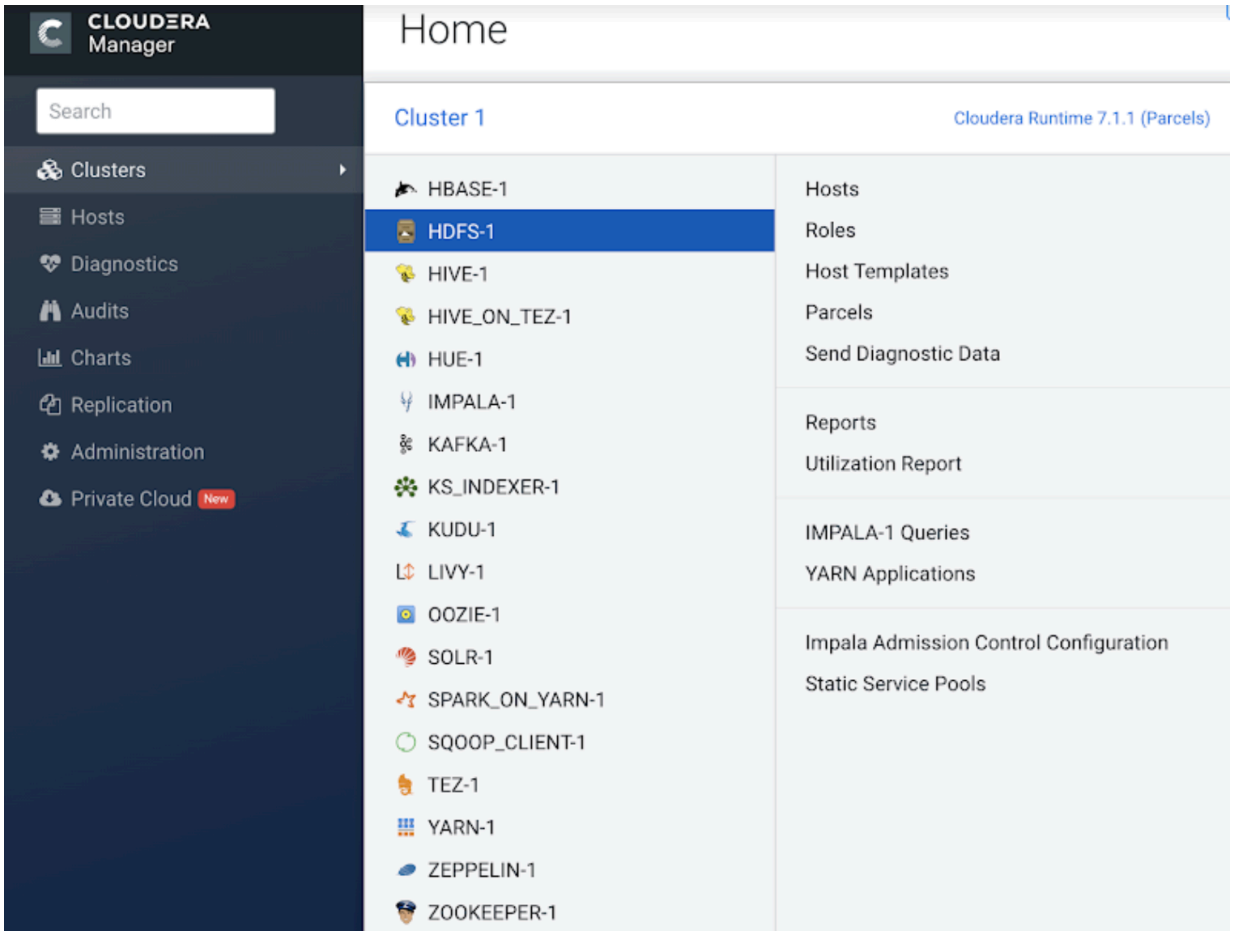

2. Click the Configuration tab.
3. Search for core-site.
4. In the Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml option, click the + sign and add the below properties:

```
Name: fs.azure.account.auth.type.<ACCOUNT_NAME>.dfs.core.windows.net
Value: SHAREDKEY (or any other auth type such as OAUTH or SAS)

Name:fs.azure.account.key.<YOUR_STORAGE_ACCOUNT_NAME>.dfs.core.windows.net
```

```
Value: <ACCESS_KEY>
```

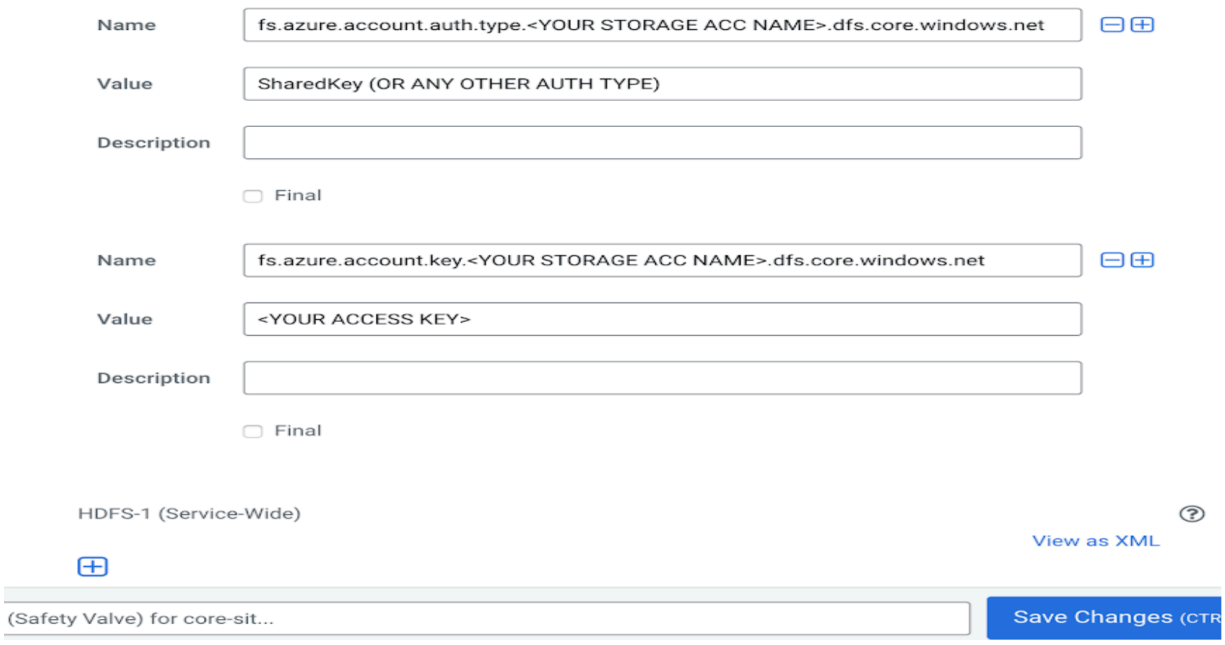

5. Click Save Changes.
6. Next to the HDFS title, click ig> Restart and Redeploy to restart and redeploy the configurations.

ADLS Proxy Setup

The ADLS connector uses the JVM proxy settings to control its proxy setup.

The Oracle Java documentation covers the options to set.

As the ADLS connector uses HTTPS, the https.proxy options are those which must be configured. See DistCp and Proxy Settings for the specifics of how to set these proxy settings in DistCp (or indeed, any MapReduce job).

Manifest committer for ABFS and GCS

The Intermediate Manifest committer is a high performance committer for Spark work that provides performance on ABFS for real world queries, and performance and correctness on GCS. It also works with other filesystems, including HDFS. However, the design is optimized for object stores where listing operatons are slow and expensive.

The v1/v2 file committers

The only committer of work from Spark to Azure ADLS Gen 2 "abfs://" storage that is safe to use is the "v1 file committer."

This is "correct" in that if a task attempt fails, its output is guaranteed not to be included in the final out. The "v2" commit algorithm cannot meet that guarantee, which is why it is no longer the default.

But the v1 file committer is slow, especially on jobs where deep directory trees of output are used. This is due to lack of any instrumentation in the FileOutputCommitter. Stack traces of running jobs generally show rename(), though list operations do surface too.

On GCS, neither the v1 nor v2 algorithm are safe because the Google filesystem doesn't have the atomic directory rename which the v1 algorithm requires.

A further issue is that both Azure and GCS storage may encounter scale issues with deleting directories with many descendants. This can trigger timeouts because the FileOutputCommitter assumes that cleaning up after the job is a fast call to delete("_temporary", true).

The manifest committer

The Manifest Committer is a higher performance committer for ABFS and GCS storage for jobs that create files across deep directory trees through many tasks. It will also work on hdfs:// and file:// URLs, but it is optimized to address listing and renaming performance and throttling issues in cloud storage.

It will not work correctly with S3, because it relies on an atomic rename-no-overwrite operation to commit the manifest file. It will also have the performance problems of copying rather than moving all the generated data.

Although it will work with MapReduce, there is no handling of multiple job attempts with recovery from previous failed attempts.

This committer uses the extension point which came in for the S3A committers. Users can declare a new committer factory for abfs:// and gcs:// URLs. A suitably configured Spark deployment will pick up the new committer.

Using the manifest committer

Define a factory for the ABFS or GCS schema, then bind to the manifest committer in Spark.

Using the committer

Define a factory for the ABFS schema in mapreduce.outputcommitter.factory.scheme.abfs or mapreduce.outputcommitter.factory.scheme.gs for GCS.

Some matching spark configuration changes, especially for parquet binding, will be required. These can be done in core-site.xml, if it is not defined in the mapred-default.xml JAR.

```
<property>
  <name>mapreduce.outputcommitter.factory.scheme.abfs</name>
  <value>org.apache.hadoop.fs.azurebfs.commit.AzureManifestCommitterFactory
</value>
</property>
<property>
  <name>mapreduce.outputcommitter.factory.scheme.gs</name>
  <value>org.apache.hadoop.mapreduce.lib.output.committer.manifest.Manife
stCommitterFactory</value>
</property>
```

Binding to the manifest committer in Spark

In Apache Spark, set the configuration either with command line options (after the '--conf') or by using the spark-defaults.conf file. The following is an example of using spark-defaults.conf, and also including the configuration for Parquet with a subclass of the parquet committer which uses the factory mechansim internally.

```
spark.hadoop.mapreduce.outputcommitter.factory.scheme.abfs org.apache.hadoop
.fs.azurebfs.commit.AzureManifestCommitterFactory
spark.hadoop.mapreduce.outputcommitter.factory.scheme.gs org.apache.hadoop
.mapreduce.lib.output.committer.manifest.ManifestCommitterFactory
spark.sql.parquet.output.committer.class org.apache.spark.internal.io.clou
d.BindingParquetOutputCommitter
spark.sql.sources.commitProtocolClass org.apache.spark.internal.io.cloud.Pat
hOutputCommitProtocol
```

Using the Cloudstore committerinfo command to probe committer bindings

The hadoop committer settings can be validated in a recent build of cloudstore and its committerinfo command. This command instantiates a committer for that path through the same factory mechanism as MR and spark jobs use, then prints its toString value.

```
hadoop jar cloudstore-1.0.jar committerinfo abfs://testing@ukwest.dfs.core.w
indows.net/
2021-09-16 19:42:59,731 [main] INFO  commands.CommitterInfo (StoreDurationI
nfo.java:<init>(53)) - Starting: Create committer
Committer factory for path abfs://testing@ukwest.dfs.core.windows.net/ is
 org.apache.hadoop.mapreduce.lib.output.committer.manifest.ManifestCommitter
Factory@3315d2d7
  (classname org.apache.hadoop.mapreduce.lib.output.committer.manifest.Mani
festCommitterFactory)
2021-09-16 19:43:00,897 [main] INFO  manifest.ManifestCommitter (ManifestC
ommitter.java:<init>(144)) - Created ManifestCommitter with
   JobID job__0000, Task Attempt attempt__0000_r_000000_1 and destination ab
fs://testing@ukwest.dfs.core.windows.net/
Created committer of class org.apache.hadoop.mapreduce.lib.output.committe
r.manifest.ManifestCommitter:
 ManifestCommitter{ManifestCommitterConfig{destinationDir=abfs://testing@u
kwest.dfs.core.windows.net/,
   role='task committer',
   taskAttemptDir=abfs://testing@ukwest.dfs.core.windows.net/_temporary/m
anifest_job__0000/0/_temporary/attempt__0000_r_000000_1,
   createJobMarker=true,
   jobUniqueId='job__0000',
   jobUniqueIdSource='JobID',
   jobAttemptNumber=0,
   jobAttemptId='job__0000_0',
   taskId='task__0000_r_000000',
   taskAttemptId='attempt__0000_r_000000_1'},
   iostatistics=counters=();

gauges=();

minimums=();
maximums=();

means=();
}
```

Verifying that the committer was used

The new committer will write a JSON summary of the operation, including statistics, in the _SUCCESS file.

If this file exists and is zero bytes long: the classic FileOutputCommitter was used.

If this file exists and is greater than zero bytes long, either the manifest committer was used, or in the case of S3A filesystems, one of the S3A committers. They all use the same JSON format.

Configuration options

Option	Meaning	Default value
mapreduce.manifest.committer.delete.target.files	Delete target files?	false
mapreduce.manifest.committer.io.threads	Thread count for parallel operations	64
mapreduce.manifest.committer.summary.report.directory	Directory to save reports	""

Option	Meaning	Default value
mapreduce.manifest.committer.cleanup.parallel.delete	Delete temporary directories in parallel	true
mapreduce.fileoutputcommitter.cleanup.skipped	Skip cleanup of _temporary directory	false
mapreduce.fileoutputcommitter.cleanup-failures.ignored	Ignore errors during cleanup	false
mapreduce.fileoutputcommitter.marksuccessfuljobs	Create a _SUCCESS marker file on successful completion (and delete any existing one in job setup)	true

Scaling jobs mapreduce.manifest.committer.io.threads

The core reason that the manifest committer is faster than the classic FileOutputCommitter is that it tries to parallelize as much file IO as it can during job commit, specifically:

- Task manifest loading
- Deletion of files where directories will be created
- Directory creation
- File-by-file renaming
- Deletion of task attempt directories in job cleanup

These operations are all performed in the same thread pool, whose size is set in the option mapreduce.manifest.committer.io.threads.

Larger values may be used.

XML:

```
<property>
  <name>mapreduce.manifest.committer.io.threads</name>
  <value>200</value>
</property>
```

spark-defaults.conf:

```
spark.hadoop.mapreduce.manifest.committer.io.threads 200
```

A larger value than that of the number of cores allocated to the MapReduce AM or Spark Driver does not directly overload the CPUs, as the threads are normally waiting for (slow) IO against the object store/filesystem to complete.

Caveats:

- In Spark, multiple jobs may be committed in the same process, each of which will create their own thread pool during job commit or cleanup.
- Azure rate throttling may be triggered if too many IO requests are made against the store. The rate throttling option mapreduce.manifest.committer.io.rate can help avoid this.

Optional: deleting target files in Job Commit

The classic FileOutputCommitter deletes files at the destination paths before renaming the job's files into place.

This is optional in the manifest committers, set in the option mapreduce.manifest.committer.delete.target.files with a default value of false. This increases performance and is safe to use when all files created by a job have unique filenames.

Apache Spark does generate unique filenames for ORC and Parquet since SPARK-8406 Adding UUID to output file name to avoid accidental overwriting.

Avoiding checks for/deleting target files saves one delete call per file being committed, so can save a significant amount of store IO.

When appending to existing tables, using formats other than ORC and Parquet, unless confident that unique identifiers are added to each filename, enable deletion of the target files.

```
spark.hadoop.mapreduce.manifest.committer.delete.target.files true
```

Note: The committer will skip deletion operations when it created the directory into which a file is to be renamed. This makes it slightly more efficient, at least if jobs appending data are creating and writing into new partitions.

Note: The committer still requires tasks within a single job to create unique files. This is foundational for any job to generate correct data.

Spark Dynamic Partition overwriting

Spark has a feature called “Dynamic Partition Overwrites” that can be initiated in SQL:

```
INSERT OVERWRITE TABLE ...
```

Or through DataSet writes where the mode is overwrite and the partitioning matches that of the existing table:

```
sparkConf.set("spark.sql.sources.partitionOverwriteMode", "dynamic")
// followed by an overwrite of a Dataset into an existing partitioned table.
eventData2
  .write
  .mode("overwrite")
  .partitionBy("year", "month")
  .format("parquet")
  .save(existingDir)
```

This Spark feature:

- Directs the job to write its new data to a temporary directory
- After job commit completes, scans the output to identify the leaf directories “partitions” into which data was written
- Deletes the content of those directories in the destination table
- Renames the new files into the partitions.

This is all done in Spark, which takes over the tasks of scanning the intermediate output tree, deleting partitions and of renaming the new files.

This feature also adds the ability for a job to write data entirely outside the destination table, which is done by writing new files into the working directory, and then Spark moving them to the final destination in job commit.

The manifest committer is compatible with dynamic partition overwrites on Azure and Google Cloud Storage as together they meet the core requirements of the extension:

- The working directory returned in getWorkPath() is in the same filesystem as the final output.
- rename() is an O(1) operation which is safe and fast to use when committing a job.

None of the S3A committers support this. The first condition is not met by the staging committers, while the second condition is not met by S3 itself.

To use the manifest committer with dynamic partition overwrites, the Spark version must contain SPARK-40034 PathOutputCommitters to work with dynamic partition overwrite.

Be aware that the rename phase of the operation will be slow if many files are renamed--this is done sequentially. Parallel renaming would speed this up, but could trigger the ABFS overload problems the manifest committer is designed to both minimize the risk of and support recovery from.

The Spark side of the commit operation will be listing/treewalking the temporary output directory (some overhead), followed by the file promotion, done with a classic filesystem rename() call. There will be no explicit rate limiting here.

This means that _dynamic partitioning should not be used on Azure Storage for SQL queries/Spark DataSet operations where many thousands of files are created. The fact that these will suffer from performance problems before throttling scale issues surface, should be considered a warning.

Job summaries in _SUCCESS files

You can view and collect job summaries in the _SUCCESS files.

_SUCCESS files

The original Hadoop committer creates a zero byte _SUCCESS file in the root of the output directory unless disabled.

The manifest committer writes a JSON summary which includes:

- The name of the committer.
- Diagnostics information.
- A list of some of the files created (for testing; a full list is excluded as it can get big).
- IO Statistics.

If, after running a query, this _SUCCESS file is zero bytes long, the manifest committer has not been used.

If it is not empty, then it can be examined.

Viewing _SUCCESS file files through the ManifestPrinter tool

The summary files are JSON, and can be viewed in any text editor.

For a more succinct summary, including better display of statistics, use the ManifestPrinter tool.

```
hadoop org.apache.hadoop.mapreduce.lib.output.committer.manifest.files.Manif
estPrinter <path>
```

This works for the files saved at the base of an output directory, and any reports saved to a report directory.

Collecting Job Summaries

The committer can be configured to save the _SUCCESS summary files to a report directory, irrespective of whether the job succeed or failed, by setting a fileystem path in the option mapreduce.manifest.committer.summary.report.dire ctory.

The path does not have to be on the same store/filesystem as the destination of work. For example, a local fileystem could be used.

XML:

```
<property>
  <name>mapreduce.manifest.committer.summary.report.directory</name>
  <value>file:///tmp/reports</value>
</property>
```

spark-defaults.conf:

```
spark.hadoop.mapreduce.manifest.committer.summary.report.directory file:///t
mp/reports
```

This allows for the statistics of jobs to be collected irrespective of their outcome, whether or not saving the _SUC CESS marker is enabled, and without problems caused by a chain of queries overwriting the markers.

Job cleanup

Job cleanup is designed to address a number of issues which may surface in cloud storage:

- Slow performance for deletion of directories.
- Timeout when deleting very deep and wide directory trees.
- General resilience to cleanup issues escalating to job failures.

Option	Meaning	Default Value
mapreduce.fileoutputcommitter.cleanup.sk ipped	Skip cleanup of _temporary directory	false
mapreduce.fileoutputcommitter.cleanup-failur es.ignored	Ignore errors during cleanup	false
mapreduce.manifest.committer.cleanup.paralle l.delete	Delete task attempt directories in parallel	true

The algorithm is:

```
if `mapreduce.fileoutputcommitter.cleanup.skipped`:
  return
if `mapreduce.manifest.committer.cleanup.parallel.delete`:
  attempt parallel delete of task directories; catch any exception
if not `mapreduce.fileoutputcommitter.cleanup.skipped`:
  delete(`_temporary`); catch any exception
if caught-exception and not `mapreduce.fileoutputcommitter.cleanup-failures.
ignored`:
  throw caught-exception
```

The goal is to perform a fast/scalable delete and throw a meaningful exception if that didn't work.

When working with ABFS and GCS, these settings should normally be left alone. If somehow errors surface during cleanup, enabling the option to ignore failures will ensure the job still completes. Disabling cleanup even avoids the overhead of cleanup, but requires a workflow or manual operation to clean up all _temporary directories on a regular basis.

Working with Azure ADLS Gen2 storage

Switch the factory for committers with ABFS destiations to the manifest committer factory.

To switch to the manifest committer, the factory for committers for destinations with abfs:// URLs must be switched to the manifest committer factory, either for the application or the entire cluster.

```
<property>
  <name>mapreduce.outputcommitter.factory.scheme.abfs</name>
  <value>org.apache.hadoop.fs.azurebfs.commit.AzureManifestCommitterFactory
</value>
</property>
```

This allows for ADLS Gen2 -specific performance and consistency logic to be used from within the committer. In particular:

- The Etag header can be collected in listings and used in the job commit phase.
- IO rename operations are rate limited
- Recovery is attempted when throttling triggers rename failures.

Warning: This committer is not compatible with older Azure storage services (WASB or ADLS Gen 1).

The core set of Azure-optimized options becomes:

```
<property>
  <name>mapreduce.outputcommitter.factory.scheme.abfs</name>
  <value>org.apache.hadoop.fs.azurebfs.commit.AzureManifestCommitterFactory
</value>
</property>

<property>
  <name>spark.hadoop.fs.azure.io.rate.limit</name>
  <value>10000</value>
</property>
```

Full set of ABFS options for Spark

```
spark.hadoop.mapreduce.outputcommitter.factory.scheme.abfs org.apache.hadoo
p.fs.azurebfs.commit.AzureManifestCommitterFactory
spark.hadoop.fs.azure.io.rate.limit 10000
spark.sql.parquet.output.committer.class org.apache.spark.internal.io.cloud
.BindingParquetOutputCommitter
spark.sql.sources.commitProtocolClass org.apache.spark.internal.io.cloud.
PathOutputCommitProtocol

spark.hadoop.mapreduce.manifest.committer.summary.report.directory  (optio
nal: URI of a directory for job summaries)
```

Experimental: ABFS Rename Rate Limiting

To avoid triggering store throttling and backoff delays, as well as other throttling-related failure conditions file renames during job commit are throttled through a "rate limiter" which limits the number of rename operations per second a single instance of the ABFS FileSystem client may issue.

Option	Meaning
fs.azure.io.rate.limit	Rate limit in operations/second for IO operations.

Set the option to 0 remove all rate limiting.

The default value of this is set to 10000, which is the default IO capacity for an ADLS storage account.

```
<property>
  <name>fs.azure.io.rate.limit</name>
  <value>10000</value>
  <description>maximum number of renames attempted per second</description>
</property>
```

This capacity is set at the level of the filesystem client, and so not shared across all processes within a single application, let alone other applications sharing the same storage account.

It will be shared with all jobs being committed by the same Spark driver, as these do share that filesystem connector.

If rate limiting is imposed, the statistic store_io_rate_limited will report the time to acquire permits for committing files.

If server-side throttling took place, signs of this can be seen in:

- The store service's logs and their throttling status codes (usually 503 or 500).
- The job statistic commit_file_rename_recovered. This statistic indicates that ADLS throttling manifested as failures in renames, failures which were recovered from in the comitter.

If these are seen or other applications running at the same time experience throttling/throttling-triggered problems, consider reducing the value of fs.azure.io.rate.limit, and/or requesting a higher IO capacity from Microsoft.

Important: if you do get extra capacity from Microsoft and you want to use it to speed up job commits, increase the value of fs.azure.io.rate.limit either across the cluster, or specifically for those jobs which you wish to allocate extra priority to.

This is still a work in progress; it may be expanded to support all IO operations performed by a single filesystem instance.

Advanced topics

Advanced topics and configuration options for the manifest committer.

Advanced configuration

There are some advanced options which are intended for development and testing, rather than production use.

Option	Meaning	Default Value
mapreduce.manifest.committer.store.operations.classname	Classname for Manifest Store Operations	""
mapreduce.manifest.committer.validate.output	Perform output validation?	false

Validating output

The option mapreduce.manifest.committer.validate.output triggers a check of every renamed file to verify it has the expected length.

This adds the overhead of a HEAD request per file, and so is recommended for testing only.

There is no verification of the actual contents.

Controlling storage integration

The manifest committer interacts with filesystems through implementations of the interface ManifestStoreOperations. It is possible to provide custom implementations for store-specific features. There is one of these for ABFS; when the abfs-specific committer factory is used this is automatically set.

It can be explicitly set:

```
<property>
  <name>mapreduce.manifest.committer.store.operations.classname</name>
  <value>org.apache.hadoop.fs.azurebfs.commit.AbfsManifestStoreOperations</
value>
</property>
```

The default implementation may also be configured:

```
<property>
  <name>mapreduce.manifest.committer.store.operations.classname</name>
```

```
  <value>org.apache.hadoop.mapreduce.lib.output.committer.manifest.impl.Ma
nifestStoreOperationsThroughFileSystem</value>
</property>
```

There is no need to alter these values, except when writing new implementations for other stores, something which is only needed if the store provides extra integration support for the committer.

Performance and Scalability

Like all Azure storage services, the Azure Datalake Gen 2 store offers a fully consistent view of the store, with a complete Create, Read, Update, and Delete operation consistency for data and metadata.

Hierarchical namespaces vs. non-namespaces

For containers with hierarchical namespaces, the scalability numbers, in Big-O Notation, are as follows:

Operation	Scalability
File Rename	0(1)
File Delete	0(1)
Directory Rename:	0(1)
Directory Delete	0(1)

For non-namespace stores, the scalability numbers are as follows:

Operation	Scalability
File Rename	`O(1)`
File Delete	`O(1)`
Directory Rename:	`O(files)`
Directory Delete	`O(files)`

Source: Performance_and_Scalability

Flush options

The Azure Blob File System and OutputStream flush options are enabled by default. These flush options have an impact on performance. Those applications which call flush() frequently, for example, with every write of a line, might show poor performance. Hence, you must disable the flush option if your application does not need it.

- Azure Blob File System Flush Options (`fs.azure.enable.flush`) - renders ABFS flush APIs - HFlush() and HSync(), to be no-op. Both the APIs ensure that data persists.
- OutputStream Flush Options (`fs.azure.disable.outputstream.flush`) - renders OutputStream Flush() API to be a no-op in AbfsOutputStream. Hflush() being the only documented API that can provide persistent data transfer, flush() also attempting to persist buffered data will lead to performance issues.

Using ABFS using CLI

After you configure access in the core-site.xml file, you can access your cluster using the CLI. You can run Hadoop file system commands, DistCP commands, create Hive tables, and so on using the CLI.

Hadoop File System commands

You can execute Hadoop file system commands using CLI on your cluster.

- Access your cluster by using the URL of the container.

```
abfs://<NAME OF CONTAINER>@<NAME OF ACCOUNT>.dfs.core.windows.net/
```

- Create a directory using the mkdir command. For example,

```
hadoop fs -mkdir abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/
myDir/
hadoop fs -ls abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/

Found 2 items

drwxr-xr-x - root root 0 2020-05-20 15:14
abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/myDir
drwxr-x--- - root root 0 2020-05-20 12:50
abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/test
```

Similarly, you can use the `put` command to add a file to one of the directories, `ls command` to list the items in your cluster, and `cat` command to read the contents of the file.

Note: If you name a file with the same file name that is already present in the directory, the existing file `is not overwritten`.

Create a table in Hive

You can create, modify, update, and remove tables in Hive using beeline or any other tool to access Hive.

1. Enter the beeline command shell by beeline command in your cluster:

```
~ beelinex
```

2. Enter the database you want to access.

```
~ use <DATABASE_NAME>;
```

Or create and use a new database. In this following example, abfsdb is the name of the database.

```
create database abfsdb;
~ use abfsdb;
```

3. Create a table inside the container in a directory named table_1. If the directory does not exist, it is automatically created:

```
~ create external table myTable(key STRING, value INT) location
'abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/table_1/';
```

4. View the table structure using the `show` command:

```
~ show create table myTable;
INFO  : Compiling command(queryId=hive_20200520153116_301af680-9630-49d1-
af40-3dfa5349e52a): show create table myTable
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:createt
ab_stmt, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20200520153116_301af680
-9630-49d1-af40-3dfa5349e52a); Time taken: 0.052 seconds
INFO  : Executing command(queryId=hive_20200520153116_301af680-9630-49d1-
af40-3dfa5349e52a): show create table myTable
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20200520153116_301af680-
9630-49d1-af40-3dfa5349e52a); Time taken: 0.072 seconds
INFO  : OK

+----------------------------------------------------+
|                    createtab_stmt                  |
```

```
+----------------------------------------------------+
| CREATE EXTERNAL TABLE `myTable`(                   |
|   `key` string,                                    |
|   `value` int)                                     |
| ROW FORMAT SERDE                                   |
|   'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'  |
| STORED AS INPUTFORMAT                              |
|   'org.apache.hadoop.mapred.TextInputFormat'       |
| OUTPUTFORMAT                                       |
|   'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat' |
| LOCATION                                           |
|   'abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/table_1' |
| TBLPROPERTIES (                                    |
|   'bucketing_version'='2',                         |
|   'transient_lastDdlTime'='1589988617')            |
+----------------------------------------------------+
14 rows selected (0.162 seconds)
```

After the table is created inside the Azure storage, it behaves like a regular table and you can use all the `hql` commands. For example, to insert values in the table, run ~ insert into myTable values("myKey", 1);.

Accessing Azure Storage account container from spark-shell

You can use spark-shell to query the files that are stored in the Azure Storage account. You should be able to access spark as an `hdfs` user using the `~ sudo -u hdfs -s` command.

```
~ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use set
LogLevel(newLevel).
20/05/20 16:35:32 WARN cluster.YarnSchedulerBackend$YarnSchedulerEndpoint: A
ttempted to request executors before the AM has registered!
Spark context available as 'sc' (master = yarn, app id = application_1589987
399184_0004).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.0.7.1.1.0-413
      /_/
Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_232)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val sampleRDD = sc.textFile("abfs://abfscontainer@abfstorageacc.dfs
.core.windows.net/myDir/testingFile.txt")
sampleRDD: org.apache.spark.rdd.RDD[String] = abfs://abfscontainer@abfstor
ageacc.dfs.core.windows.net/myDir/testingFile.txt MapPartitionsRDD[1] at tex
tFile at <console>:24

scala> sampleRDD.collect().foreach(println)
testing the hadoop fs commands on azure.
```

Hence, after accessing the azure container, the data in `sampleRDD` would work like any other text file. Now, you can use any spark operation on these flies. For example,

- Find the word count of a particular word:

```
scala> sampleRDD.filter(line => line.contains("azure")).count()
res4: Long = 1
```

- Use wordcount by MapReduce, as popularized by Hadoop. Spark can implement MapReduce flows easily:

```
scala> val wordcount = sampleRDD.flatMap(line => line.split(" ")).map(word
 => (word,1)).reduceByKey(_+_)
wordcount: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[7] at
reduceByKey at <console>:26
scala> wordcount.collect()
res7: Array[(String, Int)] = Array((commands,1), (testing,1), (hadoop,1),
 (on,1), (fs,1), (azure.,1), (the,1))
```

Copying data with Hadoop DistCp

DistCp (distributed copy) is a tool used to copy files in large inter-cluster and intra-cluster environments. It uses MapReduce to affect its distribution, error handling and recovery, and reporting. It expands a list of files and directories into input to map tasks, each of which copy a partition of the files specified in the source list.

The following are some of the examples of distcp commands with object stores:

- Copying between directories in an object store

```
~ hadoop distcp
abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/myDir/testingFi
le.txt \ abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/test/

20/05/21 08:48:09 INFO mapreduce.Job: Job job_1589987399184_0005
completed successfully

~hadoop fs -ls abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/tes
t/

Found 1 items

-rw-r--r--   1 hdfs hdfs         41 2020-05-21 08:48
abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/test/testingFil
e.txt
```

- Copying between two different object stores

```
~ hadoop distcp
abfs://abfscontainer@abfstorageacc.dfs.core.windows.net/myDir/testingFi
le.txt \ abfs://mycontainer@mystoragehastoexist.dfs.core.windows.net/new
test/

20/05/21 08:53:26 INFO mapreduce.Job: Job job_1589987399184_0007
completed successfully
~ hadoop fs -ls
abfs://mycontainer@mystoragehastoexist.dfs.core.windows.net/newtest/

Found 1 items

-rw-r--r--   1 hdfs hdfs         41 2020-05-21 08:53
abfs://mycontainer@mystoragehastoexist.dfs.core.windows.net/newtest/tes
tingFile.txt
```

For more information about the DistCp commands, see DistCP documentation.

DistCp and Proxy Settings

When using DistCp to back up data from an on-site Hadoop cluster, proxy settings may need to be set so as to reach the cloud store. For most of the stores, these proxy settings are hadoop configuration options which must be set in core-site.xml, or as options to the DistCp command.

ADLS uses the JVM proxy settings, which need to be set in DistCp's map and reduce processes. This can be done through the `mapreduce.map.java.opts` and `mapreduce.reduce.java.opts` options respectively.

```
export DISTCP_PROXY_OPTS="-Dhttps.proxyHost=web-proxy.example.com -Dhttps.pr
oxyPort=80"
hadoop distcp \
  -D mapreduce.map.java.opts="$DISTCP_PROXY_OPTS" \
  -D mapreduce.reduce.java.opts="$DISTCP_PROXY_OPTS" \
  -update -skipcrccheck -numListstatusThreads 40 \
  hdfs://namenode:8020/users/alice adl://backups.azuredatalakestore.net/us
ers/alice
```

Without these settings, even though access to ADLS may work from the command line, distcp access can fail with Error fetching access token.

ADLS Trash Folder Behavior

If the fs.trash.interval property is set to a value other than zero on your cluster and you do not specify the -skipTrash flag with your rm command when you remove files, the deleted files are moved to the trash folder in your ADLS account. The trash folder in your ADLS account is located at adl://your_account.azuredatalakestore.net/user/user_name/.Trash/current/.

For more information about HDFS trash, see Configuring HDFS Trash.

Troubleshooting ABFS

You might encounter issues when configuring the ABFS connector. The issues are usually related to classpath, network, and authentication. For tips to troubleshoot these issues, see https://hadoop.apache.org/docs/current/hadoop-azure/abfs.html#Troubleshooting.

Cloudera Runtime 7.2.11

Configuring Apache HBase

Date published: 2020-02-29
Date modified: 2024-09-18

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using DNS with HBase 7

Use the Network Time Protocol (NTP) with HBase 7

Configure the graceful shutdown timeout property 7

Setting user limits for HBase 8
- Configure ulimit for HBase using Cloudera Manager 8
- Configuring ulimit for HBase 8
- Configure ulimit using Pluggable Authentication Modules using the Command Line 9

Using dfs.datanode.max.transfer.threads with HBase 10

Configure encryption in HBase 10

Using hedged reads 11
- Enable hedged reads for HBase 11
- Monitor the performance of hedged reads 11

Understanding HBase garbage collection 11
- Configure HBase garbage collection 12
- Disable the BoundedByteBufferPool 12

Configuring edge node on AWS for Cloudera Operational Database 13
- Prerequisites for configuring an edge node on AWS for Cloudera Operational Database 13
- Configuring network line-of-sight 13
  - Reuse the subnets created for Cloudera 14
- Verify the network line-of-sight 15
- Configure DNS 15
- Verify the DNS configuration 16
- Configure Kerberos 16

Configuring edge node on Azure for Cloudera Operational Database 17
- Prerequisites for configuring an edge node on Azure for Cloudera Operational Database 17
- Configuring network line-of-sight 17
  - Reuse the subnets created for Cloudera 18
- Verify the network line-of-sight 19
- Configure DNS 19
- Verify the DNS configuration 20
- Configure Kerberos 20

Configuring edge node on GCP 21

Configure the HBase canary 21

Configuring auto split policy in an HBase table 21

Using HBase blocksize 22
Configure the blocksize for a column family 23

Configuring HBase BlockCache 24
Contents of the BlockCache 24
Size the BlockCache 24
Decide to use the BucketCache 25
About the Off-heap BucketCache 25
Off-heap BucketCache 25
BucketCache IO engine 28
Configure BucketCache IO engine 29
Configure the off-heap BucketCache using Cloudera Manager 29
Configure the off-heap BucketCache using the command line 30
Cache eviction priorities 31
Bypass the BlockCache 31
Monitor the BlockCache 32

HBase persistent BucketCache 32
Configuring HBase persistent BucketCache 32
Configuration details 33
Using the retain assignment properties 33
Using the cache-aware load balancer 33
Validating the expected behavior 34

Using quota management 34
Configuring quotas 35
General Quota Syntax 35
Throttle quotas 36
Throttle quota examples 36
Space quotas 38
Quota enforcement 39
Quota violation policies 39
Impact of quota violation policy 40
Live write access 40
Bulk Write Access 40
Read access 40
Metrics and Insight 40
Examples of overlapping quota policies 40
Number-of-Tables Quotas 41
Number-of-Regions Quotas 42

Using HBase scanner heartbeat 42
Configure the scanner heartbeat using Cloudera Manager 43

Storing medium objects (MOBs) 43
Prerequisites 44
Configure columns to store MOBs 44
Configure the MOB cache using Cloudera Manager 44
Test MOB storage and retrieval performance 45
MOB cache properties 45

Limiting the speed of compactions 46
Configure the compaction speed using Cloudera Manager 46

Enable HBase indexing 47

Using HBase coprocessors 47
Add a custom coprocessor 48
Disable loading of coprocessors 48

Configuring HBase MultiWAL 49
Configuring MultiWAL support using Cloudera Manager 49

Configuring the storage policy for the Write-Ahead Log (WAL) 49
Configure the storage policy for WALs using Cloudera Manager 50
Configure the storage policy for WALs using the Command Line 50

Using RegionServer grouping 50
Enable RegionServer grouping using Cloudera Manager 51
Configure RegionServer grouping 51
Monitor RegionServer grouping 51
Remove a RegionServer from RegionServer grouping 52
Enabling ACL for RegionServer grouping 52
Best practices when using RegionServer grouping 52
Disable RegionServer grouping 53

HBase load balancer 53
HBase cache-aware load balancer configuration 54
Overview 54
Components of cache-aware load balancer 54
Configuration details 55
HBase stochastic load balancer configuration 55
Introduction to the HBase stochastic load balancer 55
Components of stochastic load balancer 56
Configuration details 58

Optimizing HBase I/O **59**
HBase I/O components 59

Advanced configuration for write-heavy workloads **61**

Enabling HBase META Replicas **62**

Enabling ZooKeeper-less connection registry for HBase client **63**

Using DNS with HBase

You must configure DNS to resolve RegionServer and Master hostnames in your cluster. HBase uses the local hostname to report its IP address. Both forward and reverse DNS resolving works.

Procedure

1. In Cloudera Manager navigate to HBase Configuration .
2. Find the RegionServer Advanced Configuration Snippet (Safety-Valve) for hbase-site.xml property.
3. Click the plus icon to add the property that is applicable for your use case:

   If your server has multiple interfaces, HBase uses the interface that the primary hostname resolves to, and you must ensure that the cluster configuration is consistent and every host has the same network interface configuration. If this is your use case add the following configuration:

   - Name: hbase.regionserver.dns.interface
   - Value: primary interface

   If you want to use a different DNS name server than the system-wise default, add the following configuration:

   - Name: hbase.regionserver.dns.nameserver
   - Value: a different DNS name server
4. Click Save Changes.

Use the Network Time Protocol (NTP) with HBase

Ensure that that the clocks on all the cluster members are sychronized for your cluster to function correctly. You must configure NTP to synchronize the clock.

About this task

The clocks on cluster members must be synchronized for your cluster to function correctly. Some skew is tolerable, but excessive skew could generate odd behaviors.

Using NTP and DNS ensures that you won't run into odd behaviors when one node A thinks that the time is tomorrow and node B thinks it's yesterday. You will also prevent situations where the master node tells node C to serve a region but node C doesn't know its own name and doesn't answer.

Procedure

1. Run NTP or another clock synchronization mechanism on your cluster.
2. Verify that the system time is sychronized across your cluster nodes.

What to do next

For more information about NTP, see the NTP website

Configure the graceful shutdown timeout property

You must configure this property to allow enough time for a graceful shutdown of a RegionServer. A graceful shutdown of an HBase RegionServer allows the regions hosted by that RegionServer to be moved to other RegionServers before stopping the RegionServer.

About this task

This timeout only affects a graceful shutdown of the entire HBase service, not individual RegionServers. Therefore, if you have a large cluster with many RegionServers, you should strongly consider increasing the timeout from its default of 180 seconds.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope > HBASE-1 (Service Wide).
4. Use the Search box to search for the Graceful Shutdown Timeout property and edit the value.
5. Click Save Changes to save this setting.

Related Information

Graceful HBase Shutdown

Setting user limits for HBase

You must set user limits to avoid opening many files at the same time. Overloading many files at the same time leads to failure and causes error messages.

Because HBase is a database, it opens many files at the same time. The default setting of 1024 for the maximum number of open files on most Unix-like systems is insufficient. Any significant amount of loading will result in failures and cause error message such as java.io.IOException...(Too many open files) to be logged in the HBase or HDFS log files. For more information about this issue, see the Apache HBase Book. You may also notice errors such as:

```
2010-04-06 03:04:37,542 INFO org.apache.hadoop.hdfs.DFSClient: Exception inc
reateBlockOutputStream java.io.EOFException
2010-04-06 03:04:37,542 INFO org.apache.hadoop.hdfs.DFSClient: Abandoning bl
ock blk_-6935524980745310745_1391901
```

Another setting you should configure is the number of processes a user is permitted to start. The default number of processes is typically 1024. Consider raising this value if you experience OutOfMemoryException errors.

Configure ulimit for HBase using Cloudera Manager

You can use Cloudera Manager to configure ulimit.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope Master or Scope RegionServer .
4. Locate the Maximum Process File Descriptors property or search for it by typing its name in the Search box.
5. Edit the property value.
6. Restart the role.
7. Restart the service.

Configuring ulimit for HBase

You can configure the ulimit for HBase using the Maximum Process File Descriptors property in Cloudera Manager.

About this task

Cloudera recommends increasing the maximum number of file handles to more than 10,000. Increasing the file handles for the user running the HBase process is an operating system configuration, not an HBase configuration. A common mistake is to increase the number of file handles for a particular user when HBase is running as a different user. HBase prints the ulimit it is using on the first line in the logs. Make sure that it is correct.

Procedure

1. In Cloudera Manager navigate to HBase Configuration .
2. Find the Maximum Process File Descriptors property.
3. Set the property as applicable for your use case.
4. Click Save Changes.

Important: After changing the maximum process file descriptor limit specified in the above steps, the number of file descriptors still limits service roles. Raising the maximum process file descriptors above the Linux kernel file descriptor limit has no effect. Check the Linux kernel file descriptor limit on every host in the cluster and raise that if necessary.

You can find the Linux kernel file descriptor limit by running the following command on the Linux command line:

```
sudo cat /proc/sys/fs/nr_open
```

Configure ulimit using Pluggable Authentication Modules using the Command Line

Follow these command-line instructions to configure ulimit using Pluggable Authentication Modules on systems that do not use Cloudera Manager.

About this task

If you are using ulimit, you must make the following configuration changes:

Procedure

1. In the /etc/security/limits.conf file, add the following lines, adjusting the values as appropriate. This assumes that your HDFS user is called hdfs and your HBase user is called hbase.

```
hdfs  -       nofile  32768
hdfs  -       nproc   2048
hbase -       nofile  32768
hbase -       nproc   2048
```

Note:

- Only the root user can edit this file.
- If this change does not take effect, check other configuration files in the /etc/security/limits.d/ directory for lines containing the hdfs or hbase user and the nofile value. Such entries may be overriding the entries in /etc/security/limits.conf.

2. To apply the changes in /etc/security/limits.conf on Ubuntu and Debian systems, add the following line in the /etc /pam.d/common-session file:

```
session required  pam_limits.so
```

For more information on the ulimit command or per-user operating system limits, refer to the documentation for your operating system.

Using dfs.datanode.max.transfer.threads with HBase

You must configure the dfs.datanode.max.transfer.threads with HBase to specify the maximum number of files that a DataNode can serve at any one time.

About this task

A Hadoop HDFS DataNode has an upper bound on the number of files that it can serve at any one time. The upper bound is controlled by the dfs.datanode.max.transfer.threads property. Before loading, make sure you have configured this property to at least 4096.

Procedure

1. In Cloudera Manager navigate to HDFS Configuration .
2. Find the DataNode Advanced Configuration Snippet (Safety-Valve) for hdfs-site.xml property.
3. Click the plus icon to add a new property:
   - Name: dfs.datanode.max.transfer.threads
   - Value: at least 4096
4. Clik Save Changes.
5. Restart the HDFS service.

If the value is not set to an appropriate value, strange failures can occur and an error message about exceeding the number of transfer threads will be added to the DataNode logs. Other error messages about missing blocks are also logged, such as the following:

```
06/12/14 20:10:31 INFO hdfs.DFSClient: Could not obtain block blk_XXXXXXXXXX
XXXXXXXXXXXX_YYYYYYYY from any node: java.io.IOException: No live nodes cont
ain current block. Will get new block locations from namenode and retry…
```

Configure encryption in HBase

You must encrypt the HBase root directory to ensure that you have an additional layer of protection in case the HDFS filesystem is compromised. You can encrypt the HBase root directory within HDFS, using HDFS Transparent Encryption.

About this task

HBase stores all of its data under its root directory in HDFS configured in the hbase.rootdir.

Procedure

- Enable HDFS encryption using the HDFS encryption wizard.
- Follow the instructions for setting up HDFS Transparent Encryption.
- Validate and verify that HDFS encryption is enabled and working.

  For more information see, HDFS Transparent Encryption.

Using hedged reads

You can enable hedged reads if you want to increase the performance of a read operation from an HDFS block that occasionally takes a long time. This feature helps in situations where a read occasionally takes a long time rather than when there is a systemic problem.

If a read from an HDFS block is slow, the HDFS client starts up another parallel, 'hedged' read against a different block replica. The result of whichever read returns first is used, and the outstanding read is cancelled. Hedged reads can be enabled for HBase when the HFiles are stored in HDFS. This feature is disabled by default.

Enable hedged reads for HBase

You need to enable hedged read if a read operation from a HDFS block is slow.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope > HBASE-1 (Service-Wide).
4. Select Category > Performance.
5. Configure the HDFS Hedged Read Threadpool Size and HDFS Hedged Read Delay Threshold properties. The descriptions for each of these properties on the configuration pages provide more information.
6. Enter a Reason for change, and then click Save Changes to commit the changes.

Monitor the performance of hedged reads

You can monitor the performance of hedged reads using the following metrics provided by Hadoop when hedged reads are enabled.

You can monitor the following properties:

- hedgedReadOps - the number of hedged reads that have occurred
- hedgeReadOpsWin - the number of times the hedged read returned faster than the original read

Understanding HBase garbage collection

You can configure garbage collection to free up the memory that is no longer referenced by Java objects.

Warning: Configuring the JVM garbage collection for HBase is an advanced operation. Incorrect configuration can have major performance implications for your cluster. Test any configuration changes carefully.

Garbage collection (memory cleanup) by the JVM can cause HBase clients to experience excessive latency.

To tune the garbage collection settings, you pass the relevant parameters to the JVM.

Example configuration values are not recommendations and should not be considered as such. This is not the complete list of configuration options related to garbage collection. See the documentation for your JVM for details on these settings.

- -XX:+UseG1GC: Use the 'G1' garbage collection algorithm. You can tune G1 garbage collection to provide a consistent pause time, which benefits long-term running Java processes such as HBase, NameNode, Solr, and ZooKeeper. For more information about tuning G1, see the Oracle documentation on tuning garbage collection.

- -XX:MaxGCPauseMillis=value: The garbage collection pause time. Set this to the maximum amount of latency your cluster can tolerate while allowing as much garbage collection as possible. XX:+ParallelRefProcEnabled Enable or disable parallel reference processing by using a + or - symbol before the parameter name.
- -XX:-ResizePLAB: Enable or disable resizing of Promotion Local Allocation Buffers (PLABs) by using a + or - symbol before the parameter name.
- -XX:ParallelGCThreads=value: The number of parallel garbage collection threads to run concurrently.
- -XX:G1NewSizePercent=value: The percent of the heap to be used for garbage collection. If the value is too low, garbage collection is ineffective. If the value is too high, not enough heap is available for other uses by HBase.

Related Information

Tuning Java Garbage Collection for HBase

Configure HBase garbage collection

You must configure garbage collection using Cloudera Manager.

About this task

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope > RegionServer.
4. Select Category > Advanced.
5. Locate the Java Configuration Options for HBase RegionServer property or search for it by typing its name in the Search box.
6. Add or modify JVM configuration options.
7. Enter a Reason for change, and then click Save Changes to commit the changes.
8. Restart the role.

Disable the BoundedByteBufferPool

HBase uses a BoundedByteBufferPool to avoid fragmenting the heap. You can disable BoundedByteBufferPool using Cloudera Manager.

About this task

The G1 garbage collector reduces the need to avoid fragmenting the heap in some cases. If you use the G1 garbage collector, you can disable the BoundedByteBufferPool in HBase. This can reduce the number of "old generation" items that need to be collected. This configuration is experimental.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope > RegionServer.
4. Select Category > Advanced.
5. Locate the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property, or search for it by typing its name in the Search box.
6. Add the following XML:

```
<property>
```

```
  <name>hbase.ipc.server.reservoir.enabled</name>
  <value>false</value>
</property>
```

7. Enter a Reason for change, and then click Save Changes to commit the changes.
8. Restart the service.

Configuring edge node on AWS for Cloudera Operational Database

You must configure an edge node on Amazon AWS to run applications using Apache HBase Java API or the Apache Phoenix thick JDBC driver. When configuring an edge node, you have to configure a networking line-of-sight, DNS, and Kerberos.

By default, in Cloudera Operational Database, your database is not accessible to the public internet, both due to security groups on the subnets of the VPC which the database is deployed into as well as ingress rules of the VPC itself.

A correctly configured edge node can run applications using the Apache HBase Java API or the Apache Phoenix thick JDBC driver.

Prerequisites for configuring an edge node on AWS for Cloudera Operational Database

Ensure that you meet all the prerequisites; otherwise you will be unable to configure an edge node on AWS.

Ensure that you meet the following requirements before starting to configure an edge node on AWS:

- Permission to log into the AWS account configured with Cloudera
- Permission to launch EC2 nodes in your AWS account
- Basic understanding of the VPC and subnets created for use with Cloudera
- A Cloudera environment and a Cloudera Operational Database in the environment in which you want the edge node to communicate
- Optional: A JSON processing tool such as jq

Configuring network line-of-sight

An edge node must be able to access the HBase services running inside your Cloudera Operational Database. These services are bound to the private network interfaces. Therefore, your edge node must be able to resolve the private IP addresses in your Virtual Private Cloud (VPC).

There are multiple methods how you can enable your edge node to resolve the private IP addresses in your Cloudera VPC. For example:

- Reuse the subnets created for Cloudera
- VPC peering
- Direct connect
- Virtual Private Network

Reuse the subnets created for Cloudera

As a part of the initial Cloudera setup, you have already created subnet(s) or Cloudera has automatically created this for you. You can use the standard EC2 service in the AWS Console or AWS CLI to launch an EC2 node in one of these subnets. For more information, see Reuse the subnets created for Cloudera

VPC peering

Another option is to create a second VPC and peer it with the VPC you created for Cloudera. This setup allows nodes in both VPCs to communicate with one another. For more information about VPC Peering, see What is VPC peering? - Amazon Virtual Private Cloud.

Direct connect

To enable computers in your corporate network to communicate with the Cloudera VPC, you may find that the AWS Direct Connect solution is appropriate. Direct Connect enables hardware which you control to communicate with resources on AWS. For more information about Direct Connect, see AWS Direct Connect.

Virtual Private Network

If you are using a corporate VPN network, you can establish a VPN connection over the Internet to your Cloudera Operational Database. For more information, see VPN connections - Amazon Virtual Private Cloud.

Reuse the subnets created for Cloudera

To configure and edge node on AWS for your Cloudera Operational Database, you have to configure network line-of-sight. One way to configure network line-of-slight is to reuse the subnets that were created for Cloudera.

About this task

As a part of the initial Cloudera setup, you have already created subnet(s) or Cloudera has automatically created this for you. You can use the standard EC2 service in the AWS Console or AWS CLI to launch an EC2 node in one of these subnets. You can choose a Linux distribution, the EBS or instance storage, and the instance type of your choice. Ensure that you allocate enough resources for the edge node based on your use case and requirements.

Important: Launching a new EC2 instance into these subnets reduces the number of IP addresses that Cloudera can use to automatically provision new resources for you.

Procedure

1. Create an EC2 node using the AWS console. For example, Amazon Linux 2 AMI (HVM), SSD Volume Type.
2. Select an instance type. For example: t3.xlarge
3. Click Select Instance Details..
   a) Select your Cloudera VPC.
   b) Select a subnet from the VPC.
      If you have created both public and private subnets, choose the public subnet.
   c) Click Auto-assign Public IP to be true.
4. Click Next: Add Storage.
   Optionally you can increase the EBS volume size depending on your requirements.
5. Click Next, and then Add Tags as required by your IT policies.
6. Click Next: Configure Security Group.
   Add a rule for Inbound SSH on port 22 in one of the following ways:
   a. Choose from My UP to allow SSH from your computer.
   b. Provide an appropriate range for your corporate network.
7. Click Review and Launch, and then click Launch.
8. Specify the SSH key pair.
9. Open a terminal on your computer, and SSH to the instances as the ec2-user using the Public DNS (IPv4) address after the instance launches.

What to do next

Validate that you have a networking line-of-sight correctly configured.

Verify the network line-of-sight

After you create your edge node and SSH into it, validate that you have a networking line-of-sight correctly configured. You can validate this using the ping command from your edge node to an IP address within your Cloudera VPC.

Before you begin

Create and edge node and SSH into it.

Procedure

1. Navigate to Cloudera Control Plane Cloudera Management Console .
2. Click Environments and select your environment from the list.
3. Click Summary.
4. Find the FreeIPA section and copy the content of the Private IP field.

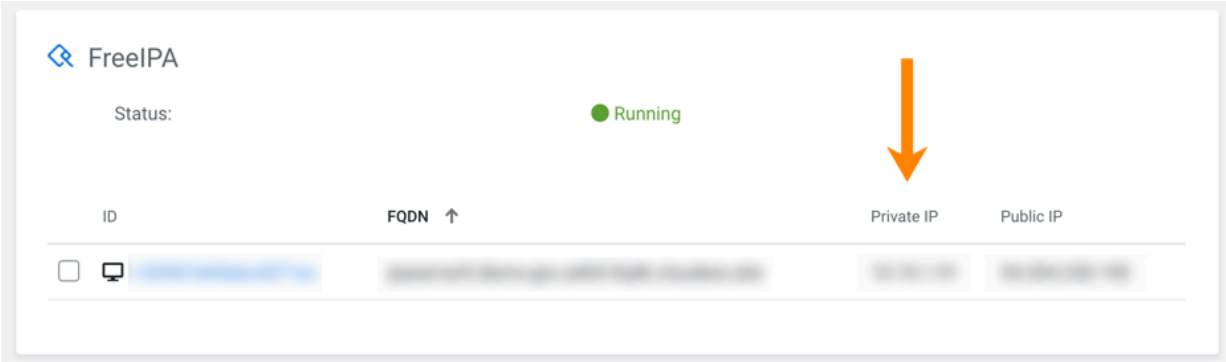

5. Open a terminal.
6. Run the ping [***PRIVATE IP***] command.
   The host will respond to your ping request.
7. If you ping request fail: Run the telnet [***PRIVATE IP***] 88 command.
   A blank screen is displayed if the connection succeeds.

If the ping request failes, it does not necessarily mean that you have failed to configure the edge node, it might indicate that you must take additional steps to validate your line-of-sight. You may have configured your Cloudera VPC to have security groups which do not respond to ping or your KDC listen port might not be open for communication.

Configure DNS

You must configure your edge node to perform forward and reverse DNS lookups with your Cloudera Operational Database. Obtain the private IP address of your Cloudera environment and configure your edge node to resolve hostnames from your Cloudera Operational Database. An edge node must resolve the hostnames from your Cloudera Operational Database.

About this task

Each Cloudera environment acts as its own DNS nameserver. Obtain the private IP address information of your environment and configure resov.conf on your edge node to list your Cloudera environment's private IP address as a nameserver.

Procedure

1. Go to the Cloudera Management Console.
2. Click Environments and select your environment from the list.
3. Click Summary.
4. Find the FreeIPA section and copy the contents of the Private IP field.
5. Add the private IP address as a nameserver to the /etc/resolv.conf file in your edge node. You can do this using the cat resolv.conf nameserver [****NAMESERVER IP ADDRESS****] command.

   Note: When the server running application with HBase client cannot be set up with reverse DNS lookup and can only perform forward DNS, then the hbase.unsafe.client.kerberos.hostname.disable.reversedns property can be set to true at the client configuration. So that HBase client connects to the HBase cluster through SASL Kerberos using the hostname of the principal and skips the reverse DNS lookup.

Verify the DNS configuration

After you configured the nameserver you have to verify that your edge node can now resolve DNS names.

Procedure

1. Navigate to Cloudera Control Plane Cloudera Management Console .
2. Click Environments and select your environment from the list .
3. Click Summary.
4. Find the FreeIPA section and copy the content of the Private IP field.
5. Open a terminal.
6. Run the following command: $ nslookup [***FULLY QUALIFIED DOMAIN NAME***]
   If your DNS is set up correctly, this command retursn an address for the name your provided.

Configure Kerberos

All Cloudera Operational Databases (CODs) are secured with Kerberos-based authentication, meaning that only authorized users can connect to your database. All HBase and Phoenix Thick JDBC clients must have a proper Kerberos configuration on the host where they run a client.

Procedure

1. Run the following command to obtain the necessary Kerberos information and a sufficient krb5.conf file encoded with Base64 :

```
$ cdp opdb describe-client-connectivity --environment-name [***YOUR
 ENVIRONMENT***] \
  --database-name [***YOUR DATABASE NAME***] | jq -r \
  '.kerberosConfiguration.krb5Conf' | base64 --decode
```

2. Copy the output of the command.
3. Add the contents into the /etc/krb5.conf file on your edge node.

What to do next

Validate that Kerberos is correctly set up. Use the kinit command to validate that you are able to obtain a Kerberos ticket:

```
$ kinit [***CLOUDERA DATA PLATFORM WORKLOAD NAME***]
```

```
                    Password: [***CLOUDERA DATA PLATFORM WORKLOAD
PASSWORD***]
```

For more information, see *Cloudera workload user* and *Setting the workload password*.

If you successfully authenticate, you will not receive an error and will be able to validate that you have a ticket using the klist command. For more information, see *Installing Cloudera CLI beta*.

Related Information

Setting the workload password

Cloudera workload user

Installing Cloudera CLI beta

Configuring edge node on Azure for Cloudera Operational Database

You must configure an edge node on Microsoft Azure to run applications using Apache HBase Java API or the Apache Phoenix thick JDBC driver. When configuring an edge node, you have to configure a networking line-of-sight, DNS, and Kerberos.

By default, in Cloudera Operational Database, your database is not accessible to the public internet, both due to security groups on the subnets of the VNet which the database is deployed into as well as ingress rules of the VNet itself.

A correctly configured edge node can run applications using the Apache HBase Java API or the Apache Phoenix thick JDBC driver.

Prerequisites for configuring an edge node on Azure for Cloudera Operational Database

You have the ensure that you fulfill all the prerequisites or you will not be able to configure an edge node on Azure.

Ensure that you meet the following requirements before starting to configure an edge node on Azure:

- Permission to log into the Azure account configured with Cloudera
- Permission to launch Virtual Machines in your Azure account
- Basic understanding of the Virtual Machines and subnets created for use with Cloudera
- A Cloudera environment and a Cloudera Operational Database in the environment in which you want the edge node to communicate
- Optional: A JSON processing tool such as jq

Configuring network line-of-sight

An edge node must be able to access the HBase services running inside of your Cloudera Operational Database. These services are bound to the private network interfaces. Therefore, your edge node must be able to resolve the private IP addresses in your Cloudera VNet.

There are multiple methods how you can allow your edge node to resolve the private IP addresses in your Cloudera VPC. For example:

- Re-use the subnets created for Cloudera
- Direct connect
- Virtual Private Network

Reuse the subnets created for Cloudera

As a part of the initial Cloudera setup, you have already created three subnets or Cloudera would have automatically created this for you. You can use the standard computer service in the Azure Marketplace to launch a Virtual Machine on the same resource group and subnet as your Cloudera Operational Database. For more information, see Reuse the subnets created for Cloudera.

Direct connect

To enable computers in your corporate network to communicate with the Cloudera VNet, you can use Azure ExpressRoute. Azure ExpressRoute enables hardware which you control to communicate with resources on Azure. For more information about Direct Connect, see Azure ExpressRoute Overview: Connect over a private connection.

Virtual Private Network

If you are using a corporate VPN network, you can use establish a VPN connection over the Internet to your Cloudera Operational Database. For more information, see About Azure VPN Gateway.

Reuse the subnets created for Cloudera

To configure and edge node on Azure for your Cloudera Operational Database, you have to configure network line-of-sight. One way to configure network line-of-slight is to reuse the subnets that were created for Cloudera.

About this task

As a part of the initial Cloudera setup, you have already created three subnets or Cloudera would have automatically created this for you. You can use the standard computer service in the Azure Marketplace to launch a Virtual Machine on the same resource group and subnet as your Cloudera Operational Database. You can choose a Linux distribution, the size of the instance, and the authentication based on your requirements and security policies. Ensure that you allocate enough resources for the edge node based on your use case and requirements.

Important: Launching a new Virtual Machine on the same subnet as Cloudera Operational Database reduces the number of IP addresses that Cloudera can use to automatically provision new resources for you.

Procedure

1. Create a Virtual Machine from the Azure Marketplace.

   For example, CentOS, standard instance with and image of your choice.

   Note: You can use a public key or password authentication based on your security requirements. Spot instances are not recommended for the edge note.

2. Set your inbound port rules to allow access to this edge node by both the Cloudera Operational Database as well as your client applications.
3. Click Next Disks .
4. Select a storage type.
   a) Select your Cloudera VNet.
   b) Select a subnet from the VNet.

      If you have created both public and private subnets, choose the public subnet.
   c) Select the Public IP that is accessible from the Internet.
5. Review your settings and then click Review + Create.

What to do next

Validate that you have a networking line-of-sight correctly configured.

Verify the network line-of-sight

After you have created your edge node and can SSH into it, validate that you have a networking line-of-sight correctly configured. You can validate this using the ping command from your edge node to an IP address from within your Cloudera VNet.

Before you begin

Create and edge node and SSH into it.

Procedure

1. Navigate to Cloudera Control Plane Cloudera Management Console .
2. Click Environments and select your environment from the list of environments.
3. Click Summary.
4. Find the FreeIPA section and copy the content of the Private IP field.
5. Open a terminal.
6. Run the ping [****PRIVATE IP****] command.
   The host will respond to your ping request.

If the ping fails, it does not necessarily mean that you have failed to configure the edge node, but it indicates that you have to take additional steps to validate your line-of-sight. You may have configured your Cloudera VNet to have security groups which do not respond to ping.

Configure DNS

You must configure your edge node to perform forward and reverse DNS lookups with your Cloudera Operational Database. Obtain the private IP address of your Cloudera environment and configure your edge node to resolve hostnames from your Cloudera Operational Database. An edge node must resolve the hostnames from your Cloudera Operational Database.

About this task

Each Cloudera environment acts as its own DNS nameserver. Obtain the private IP address information of your environment and configure resov.conf on your edge node to list your Cloudera environment's private IP address as a nameserver.

Procedure

1. Go to the Cloudera Management Console.
2. Click Environments and select your environment from the list.
3. Click Summary.
4. Find the FreeIPA section and copy the contents of the Private IP field.
5. Add the private IP address as a nameserver to the /etc/resolv.conf file in your edge node. You can do this using the cat resolv.conf nameserver [****NAMESERVER IP ADDRESS****] command.

   Note: When the server running application with HBase client cannot be set up with reverse DNS lookup and can only perform forward DNS, then the hbase.unsafe.client.kerberos.hostname.disable.reversedns property can be set to true at the client configuration. So that HBase client connects to the HBase cluster through SASL Kerberos using the hostname of the principal and skips the reverse DNS lookup.

Verify the DNS configuration

After you configured the nameserver you have to verify that your edge node can now resolve DNS names.

Procedure

1. Navigate to Cloudera Control Plane Cloudera Management Console .
2. Click Environments and select your environment from the list .
3. Click Summary.
4. Find the FreeIPA section and copy the content of the Private IP field.
5. Open a terminal.
6. Run the following command: $ nslookup [***FULLY QUALIFIED DOMAIN NAME***]
   If your DNS is set up correctly, this command retursn an address for the name your provided.

Configure Kerberos

All Cloudera Operational Databases (CODs) are secured with Kerberos-based authentication, meaning that only authorized users can connect to your database. All HBase and Phoenix Thick JDBC clients must have a proper Kerberos configuration on the host where they run a client.

Procedure

1. Run the following command to obtain the necessary Kerberos information and a sufficient krb5.conf file encoded with Base64 :

```
$ cdp opdb describe-client-connectivity --environment-name [***YOUR
 ENVIRONMENT***] \
  --database-name [***YOUR DATABASE NAME***] | jq -r \
  '.kerberosConfiguration.krb5Conf' | base64 --decode
```

2. Copy the output of the command.
3. Add the contents into the /etc/krb5.conf file on your edge node.

What to do next

Validate that Kerberos is correctly set up. Use the kinit command to validate that you are able to obtain a Kerberos ticket:

```
$ kinit [***CLOUDERA DATA PLATFORM WORKLOAD NAME***]
                    Password: [***CLOUDERA DATA PLATFORM WORKLOAD
 PASSWORD***]
```

For more information, see *Cloudera workload user* and *Setting the workload password*.

If you successfully authenticate, you will not receive an error and will be able to validate that you have a ticket using the klist command. For more information, see *Installing Cloudera CLI beta*.

Related Information

Setting the workload password

Cloudera workload user

Installing Cloudera CLI beta

Configuring edge node on GCP

You must configure an edge node on GCP to run applications.

1. From the Cloudera Management Console, click Data Hub Clusters
2. Click Create Data Hub.
3. In the Selected Environment with running Data Lake drop-down list, select the same environment used by your Cloudera Operational Database instance.
4. Select the Cluster Definition.
5. In the Cluster Definition drop-down list, select the [***RUNTIME VERSION***] COD Edge Node for [***CLOUD PROVIDER NAME***].
6. In the Cluster Name field, provide a cluster name that you can identify later as an edge node of a specific Cloudera Operational Database instance.
7. Click Provision Cluster.

Configure the HBase canary

The HBase canary is an optional service that you can configure to check periodically if a RegionServer is alive. The HBase canary is disabled by default.

About this task

This canary is different from the Cloudera Service Monitoring canary and is provided by the HBase service. After enabling the canary, you can configure several different thresholds and intervals relating to it, as well as exclude certain tables from the canary checks. The canary works on Kerberos-enabled clusters if you have the HBase client configured to use Kerberos.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope > HBase or HBase Service-Wide.
4. Select Category > Monitoring.
5. Locate the HBase Canary property or search for it by typing its name in the Search box. Several properties have Canary in the property name.
6. Select the checkbox.
7. Review other HBase Canary properties to configure the specific behavior of the canary. To apply this configuration property to other role groups as needed, edit the value for the appropriate role group.See Modifying Configuration Properties Using Cloudera Manager.
8. Enter a Reason for change, and then click Save Changes to commit the changes.
9. Restart the role.
10. Restart the service.

Configuring auto split policy in an HBase table

Know how to configure automatic split policy in an HBase table.

About this task

When a region in an HBase table reaches a certain limit, HBase automatically divides it into two regions. This is the HBase default split policy. However, you can override the default split policy and configure a custom split policy for the regions in an HBase table.

Using the Cloudera Manager, you can configure the default split policy at the system level or the table level using the HBase shell.

In this topic, let us consider an example of configuring the split policy for regions in an HBase table by setting the hbase.regionserver.region.split.policy parameter.

Before you begin

Ensure that you have administrator privileges in the Cloudera environment.

Procedure

1. Log in to the Cloudera Manager as an administrator.
2. Select the HBase service.
3. Go to Configuration Category Advanced HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml .
4. Configure the split policy that you need.

   For example, to set the split policy for the regions in an HBase table set the value of hbase.regionserver.region.split.policy as org.apache.hadoop.hbase.regionserver.ConstantSizeRegionSplitPolicy.

   For more information on custom policies, see *HBase custom split policies*.
5. Click Save Changes.
6. Restart the HBase service.

   Alternatively, you can use the HBase shell to set the split policy for the regions at the table level.

   a. Log in to the HBase shell.
   b. Use the ALTER command to set the value to org.apache.hadoop.hbase.regionserver.ConstantSizeRegionSplit Policy.

      The following is a sample command.

```
$ hbase shell
$ hbase > alter '<TABLE_NAME>', {METADATA => {'SPLIT_POLICY' => 'org.a
pache.hadoop.hbase.regionserver.ConstantSizeRegionSplitPolicy'}}
```

Related Information

Apache HBase Region Splitting and Merging

HBase custom split policies

Using HBase blocksize

You must configure the HBase blocksize to set the smallest unit of data HBase can read from the column family's HFiles.

HBase data is stored in one (after a major compaction) or more (possibly before a major compaction) HFiles per column family per region. The blocksize determines:

- The blocksize for a given column family determines the smallest unit of data HBase can read from the column family's HFiles.
- The basic unit of measure cached by a RegionServer in the BlockCache.

The default blocksize is 64 KB. The appropriate blocksize is dependent upon your data and usage patterns. Use the following guidelines to tune the blocksize size, in combination with testing and benchmarking as appropriate.

Warning: The default blocksize is appropriate for a wide range of data usage patterns, and tuning the blocksize is an advanced operation. The wrong configuration can negatively impact performance.

- Consider the average key/value size for the column family when tuning the blocksize. You can find the average key/value size using the HFile utility:

```
$ hbase org.apache.hadoop.hbase.io.hfile.HFile -f /path/to/HFILE -m -v
...
Block index size as per heapsize: 296
reader=hdfs://srv1.example.com:9000/path/to/HFILE, \
compression=none, inMemory=false, \
firstKey=US6683275_20040127/mimetype:/1251853756871/Put, \
lastKey=US6684814_20040203/mimetype:/1251864683374/Put, \
avgKeyLen=37, avgValueLen=8, \
entries=1554, length=84447
...
```

- Consider the pattern of reads to the table or column family. For instance, if it is common to scan for 500 rows on various parts of the table, performance might be increased if the blocksize is large enough to encompass 500-1000 rows, so that often, only one read operation on the HFile is required. If your typical scan size is only 3 rows, returning 500-1000 rows would be overkill.

  It is difficult to predict the size of a row before it is written, because the data will be compressed when it is written to the HFile. Perform testing to determine the correct blocksize for your data.

Configure the blocksize for a column family

You can configure the blocksize of a column family at table creation or by disabling and altering an existing table.

About this task

These instructions are valid whether or not you use Cloudera Manager to manage your cluster.

To configure the blocksize for a column family:

Procedure

1. In the HBase shell, type:

```
hbase> create ‘test_table#,{NAME => ‘test_cf#, BLOCKSIZE => '262144'}
hbase> disable 'test_table'
hbase> alter 'test_table', {NAME => 'test_cf', BLOCKSIZE => '524288'}
hbase> enable 'test_table'
```

   After changing the blocksize, the HFiles will be rewritten during the next major compaction.

2. To trigger a major compaction, issue the following command in HBase Shell:

```
hbase> major_compact 'test_table'
```

   Depending on the size of the table, the major compaction can take some time and have a performance impact while it is running.

3. To view the blocksize metrics, see the block_cache* entries in the RegionServer metrics section in the HBase web user interface.

Configuring HBase BlockCache

You can configure BlockCache in two different ways in HBase: the default on-heap LruBlockCache and the BucketCache, which is usually off-heap.

If you have less than 20 GB of RAM available for use by HBase, consider tailoring the default on-heap BlockCache implementation (LruBlockCache) for your cluster.

If you have more than 20 GB of RAM available, consider adding off-heap BlockCache (BucketCache).

In the default configuration, HBase uses a single on-heap cache. If you configure the off-heap BucketCache, the on-heap cache is used for Bloom filters and indexes, and the off-heap BucketCache is used to cache data blocks. This is called the Combined Blockcache configuration. The Combined BlockCache allows you to use a larger in-memory cache while reducing the negative impact of garbage collection in the heap, because HBase manages the BucketCache instead of relying on the garbage collector.

Contents of the BlockCache

In HBase, a block is a single unit of I/O. The block cache keeps data blocks resident in the memory after they are read.

To size the BlockCache correctly, you need to understand what HBase places into it.

- Your data: Each time a Get or Scan operation occurs, the result is added to the BlockCache if it was not already cached there. If you use the BucketCache, data blocks are always cached in the BucketCache.
- Row keys: When a value is loaded into the cache, its row key is also cached. This is one reason to make your row keys as small as possible. A larger row key takes up more space in the cache.
- hbase:meta: The hbase:meta catalog table keeps track of which RegionServer is serving which regions. It can consume several megabytes of cache if you have a large number of regions, and has in-memory access priority, which means HBase attempts to keep it in the cache as long as possible.
- Indexes of HFiles: HBase stores its data in HDFS in a format called HFile. These HFiles contain indexes which allow HBase to seek for data within them without needing to open the entire HFile. The size of an index is a factor of the block size, the size of your row keys, and the amount of data you are storing. For big data sets, the size can exceed 1 GB per RegionServer, although the entire index is unlikely to be in the cache at the same time. If you use the BucketCache, indexes are always cached on-heap.
- Bloom filters: If you use Bloom filters, they are stored in the BlockCache. If you use the BucketCache, Bloom filters are always cached on-heap.

The sum of the sizes of these objects is highly dependent on your usage patterns and the characteristics of your data. For this reason, the HBase Web UI and Cloudera Manager each expose several metrics to help you size and tune the BlockCache.

Size the BlockCache

When you use the LruBlockCache, the blocks needed to satisfy each read are cached, old blocks are evicted to make room for new blocks using a Least-Recently-Used algorithm . Set the size of the BlockCache to satisfy your read requirements.

The size cached objects for a given read may be significantly larger than the actual result of the read. For instance, if HBase needs to scan through 20 HFile blocks to return a 100 byte result, and the HFile blocksize is 100 KB, the read will add 20 * 100 KB to the LruBlockCache.

Because the LruBlockCache resides entirely within the Java heap, the amount of which is available to HBase and what percentage of the heap is available to the LruBlockCache strongly impact performance. By default, the amount of HBase heap reserved for LruBlockCache (hfile.block.cache.size) is .40, or 40%. To determine the amount of heap

available for the LruBlockCache, use the following formula. The 0.99 factor allows 1% of heap to be available as a "working area" for evicting items from the cache. If you use the BucketCache, the on-heap LruBlockCache only stores indexes and Bloom filters, and data blocks are cached in the off-heap BucketCache.

```
number of RegionServers * heap size * hfile.block.cache.size * 0.99
```

To tune the size of the LruBlockCache, you can add RegionServers or increase the total Java heap on a given RegionServer to increase it, or you can tune hfile.block.cache.size to reduce it. Reducing it will cause cache evictions to happen more often, but will reduce the time it takes to perform a cycle of garbage collection. Increasing the heap will cause garbage collection to take longer but happen less frequently.

Decide to use the BucketCache

The BucketCache manages areas of memory called *buckets* for holding the cached blocks. You can use BucketCache if any of the conditions listed in here are are true.

- If the result of a Get or Scan typically fits completely in the heap, the default configuration, which uses the on-heap LruBlockCache, is the best choice, as the L2 cache will not provide much benefit. If the eviction rate is low, garbage collection can be 50% less than that of the BucketCache, and throughput can be at least 20% higher.
- Otherwise, if your cache is experiencing a consistently high eviction rate, use the BucketCache, which causes 30-50% of the garbage collection of LruBlockCache when the eviction rate is high.
- BucketCache using file mode on solid-state disks has a better garbage-collection profile but lower throughput than BucketCache using off-heap memory.

About the Off-heap BucketCache

If the BucketCache is enabled, it stores data blocks, leaving the on-heap cache free for storing indexes and Bloom filters.

The physical location of the BucketCache storage can be either in memory (off-heap) or in a file stored in a fast disk.

- Off-heap: This is the default configuration.
- File-based: You can use the file-based storage mode to store the BucketCache on an SSD or FusionIO device,

You can configure a column family to keep its data blocks in the L1 cache instead of the BucketCache, using the HColumnDescriptor.cacheDataInL1(true) method or by using the following syntax in HBase Shell:

```
hbase> alter 'myTable', CONFIGURATION => {CACHE_DATA_IN_L1 => 'true'}}
```

Off-heap BucketCache

If the BucketCache is enabled, it stores data blocks, leaving the on-heap cache free for storing indexes and Bloom filters. The physical location of the BucketCache storage can be either in memory (off-heap) or in a file stored in a fast disk.

- Off-heap: This is the default configuration.
- File-based: You can use the file-based storage mode to store the BucketCache on an SSD or FusionIO device,

This table summaries the important configuration properties for the BucketCache. To configure the BucketCache. The table is followed by three diagrams that show the impacts of different blockcache settings.

Table 1: BucketCache Configuration Properties

Property	Default	Description
hbase.bucketcache.combinedcache.enabled	true	When BucketCache is enabled, use it as a L2 cache for LruBlockCache. If set to true, indexes and Bloom filters are kept in the LruBlockCache and the data blocks are kept in the BucketCache.
hbase.bucketcache.ioengine	none (BucketCache is disabled by default)	Where to store the contents of the BucketCache. Its value can be offheap, file:*PATH*, mmap:*PATH* or pmem:*PATH* where PATH is the path to the file that host the file-based cache.
hfile.block.cache.size	0.4	A float between 0.0 and 1.0. This factor multiplied by the Java heap size is the size of the L1 cache. In other words, the percentage of the Java heap to use for the L1 cache.
hbase.bucketcache.size	not set	When using BucketCache, this is a float that represents one of two different values, depending on whether it is a floating-point decimal less than 1.0 or an integer greater than 1.0. • If less than 1.0, it represents a percentage of total heap memory size to give to the cache. • If greater than 1.0, it represents the capacity of the cache in megabytes
hbase.bucketcache.bucket.sizes	4, 8, 16, 32, 40, 48, 56, 64, 96, 128, 192, 256, 384, 512 KB	A comma-separated list of sizes for buckets for the BucketCache if you prefer to use multiple sizes. The sizes should be multiples of the default blocksize, ordered from smallest to largest. The sizes you use will depend on your data patterns. This parameter is experimental.
-XX:MaxDirectMemorySize	MaxDirectMemorySize = BucketCache + 1	A JVM option to configure the maximum amount of direct memory available for the JVM. It is automatically calculated and configured based on the following formula: MaxDirectMemorySize = BucketCache size + 1 GB for other features using direct memory, such as DFSClient. For example, if the BucketCache size is 8 GB, it will be -XX:MaxDirectMemorySize=9G.

Figure 1: Default LRUCache, L1 only block cache hbase.bucketcache.ioengine=NULL

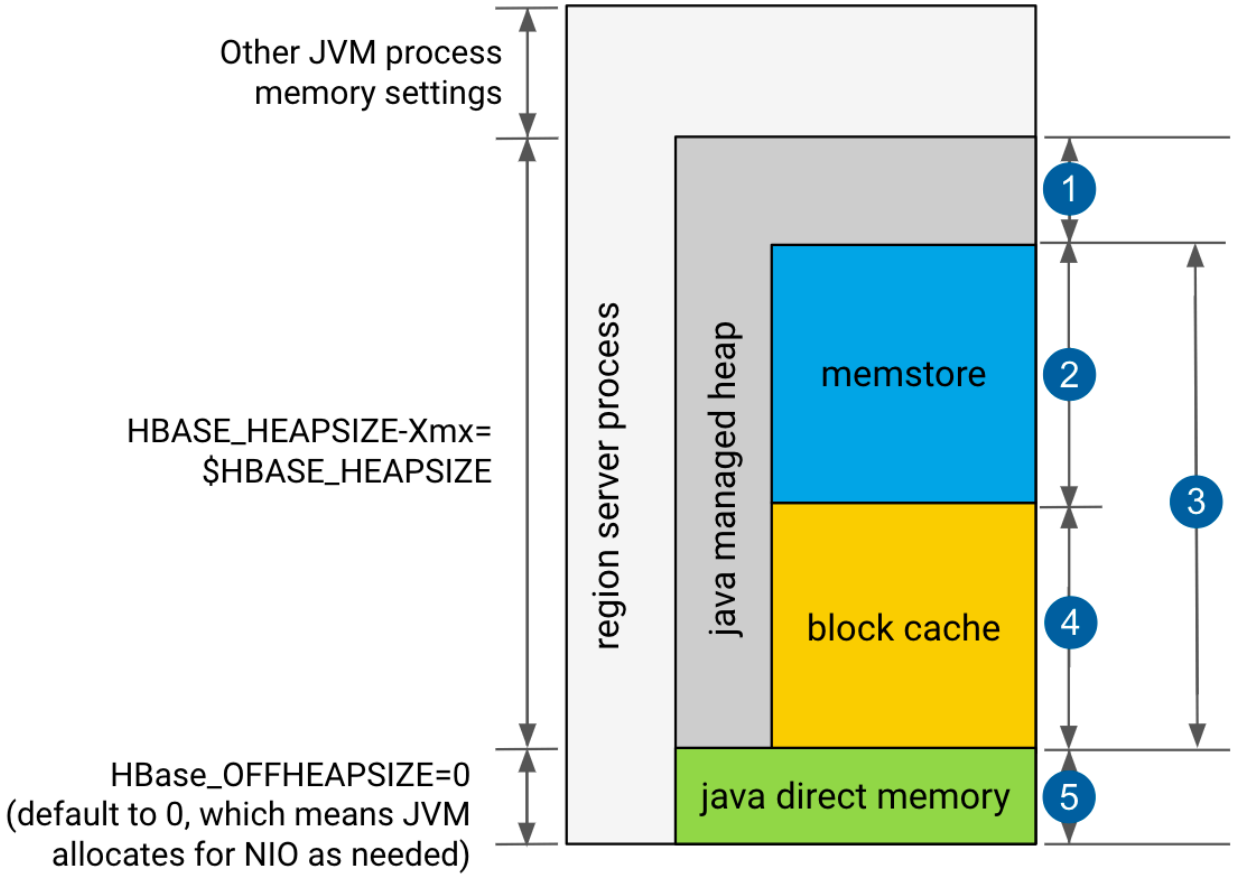

1. 20% minimum reserved for operations and rpc call queues
2. hbase.regionserver.global.memstore.size: default is 0.4, which means 40%
3. hbase.regionserver.global.memstore.size + hfile.block.cache.size #0.80, which means 80%
4. hfile.block.cache.size: default is 0.4, which means 40%
5. slack reserved for HDFS SCR/NIO: number of open HFiles * hbase.dfs.client.read.shortcircuit.buffer.size, where hbase.dfs.client.read.shortcircuit.buffer.size is set to 128k.

Figure 2: Default LRUCache, L1 only block cache hbase.bucketcache.ioengine=offheap

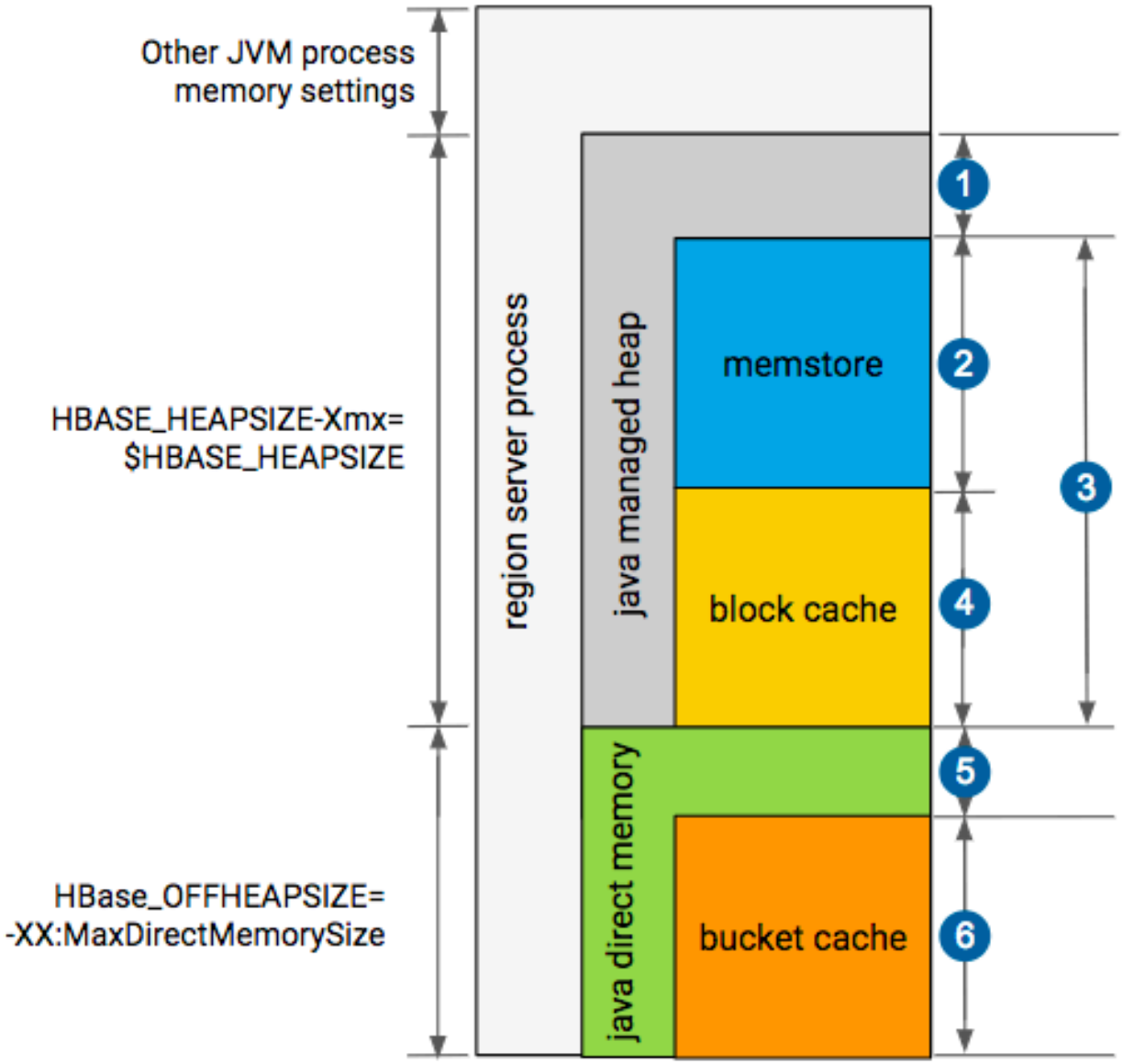

1. 20% minimum reserved for operations and rpc call queues
2. hbase.regionserver.global.memstore.size: default is 0.4, which means 40%
3. hbase.regionserver.global.memstore.size + hfile.block.cache.size #0.80, which means 80%
4. hfile.block.cache.size: default is 0.4 which means 40%
5. slack reserved for HDFS SCR/NIO: number of open HFiles * hbase.dfs.client.read.shortcircuit.buffer.size, where hbase.dfs.client.read.shortcircuit.buffer.size is set to 128k.
6. hbase.bucketcache.size: default is 0.0

   If hbase.bucketcache.size is float <1, it represents the percentage of total heap size.

   If hbase.bucketcache.size is #1, it represents the absolute value in MB. It must be < HBASE_OFFHEAPSIZE

BucketCache IO engine

Use the hbase.bucketcache.ioengine parameter to define where to store the content of the BucketCache. Its value can be offheap, file:*PATH*, mmap:*PATH*, pmem:*PATH* , or it can be empty. By default it is empty which means that BucketCache is disabled.

You can set the following values in the hbase.bucketcache.ioengine parameter to define where to store the BucketCache:

- offheap: When hbase.bucketcache.ioengine is set to offheap the content of the BucketCache is stored off-heap.
- file:*PATH*: When hbase.bucketcache.ioengine is set to file:*PATH*, the BucketCache uses file caching.
- mmap:*PATH*: When hbase.bucketcache.ioengine is set to mmap:*PATH*, the content of the BucketCache is stored and accessed through memory mapping to a file under the specified path.
- pmem:*PATH*: When hbase.bucketcache.ioengine is set to pmem:*PATH*, BucketCache uses direct memory access to and from a file on the specified path. The specified path must be under a volume that is mounted on a persistent memory device that supports direct access to its own address space.

  The advantage of the pmem engine over the mmap engine is that it supports large cache size. That is because pmem allows for reads straight from the device address, which means in this mode no copy is created on DRAM. Therefore, swapping due to DRAM free memory exhaustion is not an issue when large cache size is specified. With devices currently available, the bucket cache size can be set to the order of hundreds of GBs or even a few TBs.

  When bucket cache size is set to larger than 256GB, the OS limit must be increased, which can be configured by the max_map_count property. Make sure you have an extra 10% for other processes on the host that require the use of memory mapping. This additional overhead depends on the load of processes running on the RS hosts. To calculate the OS limit divide the block cache size in GB by 4 MB and then multiply it by 1.1: (block cache size in GB / 4 MB) * 1.1.

Configure BucketCache IO engine

You must configure the BucketCache IO engine using Cloudera Manager.

Set the value offheap and file:*PATH*

1. In Cloudera Manager select the HBase service and go to Configuration.
2. Search for BucketCache IOEngine and set it to the required value.

Set the value mmap:*PATH* and pmem:*PATH*

Important: These values can only be set using safety valves.

1. In Cloudera Manager select the HBase service and go to Configuration.
2. Search for RegionServer Advanced Configuration Snippet (Safety Valve) for hbase-site.xml.
3. Click the plus icon.
4. Set the required value:
   - Name: Add hbase.bucketcache.ioengine.
   - Value: Add either mmap:*PATH*: or pmem:*PATH*.

Configure the off-heap BucketCache using Cloudera Manager

You can configure the off-heap BucketCache engine using Cloudera Manager.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.

3. Select the RegionServer scope and do the following:
   a) Set BucketCache IOEngine to offheap.
   b) Update the value of BucketCache Size according to the required BucketCache size.

      When you set the BucketCache Size, Cloudera Manager automatically sets the -XX:MaxDirectMemorySize JVM option for the RegionServers.

      Cloudera Manager automatically adds the JVM option -XX:MaxDirectMemorySize=<size> replacing <*SIZE*> with a value not smaller than the off-heap BucketCache, expressed as a number of gigabytes + around 1GB used for HDFS short circuit read. For example, if the off-heap BucketCache is 16GB and the heap size is 15GB, the total value of MaxDirectMemorySize could be 32G: -XX:MaxDirectMamorySize=32G.
4. Optionally, when combined BucketCache is in use, you can decrease the heap size ratio allocated to the L1 BlockCache, and increase the Memstore size. The on-heap BlockCache only stores indexes and Bloom filters, the actual data resides in the off-heap BucketCache. A larger Memstore is able to accommodate more write request before flushing them to disks.
   - Decrease HFile Block Cache Size to 0.3 or 0.2.
   - Increase Maximum Size of All Memstores in RegionServer to 0.5 or 0.6 respectively.
5. Enter a Reason for change, and then click Save Changes to commit the changes.
6. Restart or rolling restart your RegionServers for the changes to take effect.

Configure the off-heap BucketCache using the command line

You can configure the off-heap BucketCache engine from the command-line interface.

About this task

Procedure

1. Configure the MaxDirectMemorySize option for the RegionServers JVMS. Add the JVM option $HBASE_REGIONSERVER_OPTS -XX:MaxDirectMemorySize=<size>G, replacing <size> with a value not smaller than the aggregated heap size expressed as a number of gigabytes + the off-heap BucketCache, expressed as a number of gigabytes + around 1GB used for HDFS short circuit read. For example, if the off-heap BucketCache is 16GB and the heap size is 15GB, the total value of MaxDirectMemorySize could be 32: -XX:MaxDirectMamorySize=32G. This can be done adding the following line in hbase-env.sh:

```
HBASE_REGIONSERVER_OPTS="$HBASE_REGIONSERVER_OPTS -XX:MaxDirectMemorySiz
e=<size>G"
```

2. In the hbase-site.xml files on the RegionServers, configure the properties in BucketCache configuration properties as appropriate, using the example below as a model.

   Important: If you are using Cloudera Manager, it can re-generate and therefore overwrite the hbase-site.xml configuration file. If you are using Cloudera Manager, Cloudera recommends to use the RegionServer Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property in Cloudera Manager to edit the hbase-site.xml configuration file on the RegionServer.

```
<property>
  <name>hbase.bucketcache.combinedcache.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.bucketcache.ioengine</name>
  <value>offheap</value>
</property>
<property>
  <name>hbase.bucketcache.size</name>
  <value>8388608</value>
</property>
<property>
```

```
  <name>hfile.block.cache.size</name>
  <value>0.2</value>
</property>
<property>
  <name>hbase.regionserver.global.memstore.size</name>
  <value>0.6</value>
</property>
```

Optionally, when combined BucketCache is in use, you can decrease the heap size ratio allocated to the L1 BlockCache, and increase the Memstore size as it is done in the above example. The on-heap BlockCache only stores indexes and Bloom filters, the actual data resides in the off-heap BucketCache. A larger Memstore is able to accommodate more write request before flushing them to disks.

- Decrease hfile.block.cache.size to 0.3 or 0.2.
- Increase hbase.regionserver.global.memstore.size to 0.5 or 0.6 respectively.

3. Restart each RegionServer for the changes to take effect.

Cache eviction priorities

You must decide on the cache eviction priorities to allow for scan-resistance and in-memory column families.

Both the on-heap cache and the off-heap BucketCache use the same cache priority mechanism to decide which cache objects to evict to make room for new objects. Three levels of block priority allow for scan-resistance and in-memory column families. Objects evicted from the cache are subject to garbage collection.

- Single access priority: The first time a block is loaded from HDFS, that block is given single access priority, which means that it will be part of the first group to be considered during evictions. Scanned blocks are more likely to be evicted than blocks that are used more frequently.
- Multi access priority: If a block in the single access priority group is accessed again, that block is assigned multi access priority, which moves it to the second group considered during evictions, and is therefore less likely to be evicted.
- In-memory access priority: If the block belongs to a column family which is configured with the in-memory configuration option, its priority is changed to in memory access priority, regardless of its access pattern. This group is the last group considered during evictions, but is not guaranteed not to be evicted. Catalog tables are configured with in-memory access priority.

  To configure a column family for in-memory access, use the following syntax in HBase Shell:

  ```
  hbase> alter 'myTable', 'myCF', CONFIGURATION => {IN_MEMORY => 'true'}
  ```

  To use the Java API to configure a column family for in-memory access, use the HColumnDescriptor.setInMemory(true) method.

Bypass the BlockCache

You can bypass the BlockCache if the data needed for a specific but atypical operation does not all fit in memory.

For an atypical operation does not all fit in memory, using the BlockCache can be counter-productive because data that you are still using may be evicted, or even if other data is not evicted, excess garbage collection can adversely effect performance. For this type of operation, you may decide to bypass the BlockCache. To bypass the BlockCache for a given Scan or Get, use the setCacheBlocks(false) method.

In addition, you can prevent a specific column family's contents from being cached, by setting its BLOCKCACHE configuration to false. Use the following syntax in HBase Shell:

```
hbase> alter 'myTable', CONFIGURATION => {NAME => 'myCF', BLOCKCACHE => 'false'}
```

Monitor the BlockCache

Cloudera Manager provides metrics to monitor the performance of the BlockCache, to assist you in tuning your configuration.

View additional details

You can view further detail and graphs using the RegionServer UI. To access the RegionServer UI in Cloudera Manager, go to the Cloudera Manager page for the host, click the RegionServer process, and click HBase RegionServer Web UI.

If you do not use Cloudera Manager, access the BlockCache reports at http://regionServer_host:22102/rs-status#memoryStats, replacing regionServer_host with the hostname or IP address of your RegionServer.

HBase persistent BucketCache

You can configure HBase persistent BucketCache to ensure that the HBase cache does not reset by itself when a RegionServer restarts.

The original BucketCache implementation is always volatile. To concretize this, HBase supports configuring the persistent BucketCache functionality. This functionality ensures that the HBase cache does not reset by itself when RegionServer restarts when you deploy it with the ioengine file.

Configuring HBase persistent BucketCache

Know how to configure the HBase persistent BucketCache in your Cloudera environment using Cloudera Manager.

About this task

This functionality only applies to the BucketCache instances that use the file type ioengine. It also requires either setting both hbase.master.scp.retain.assignment.force and hbase.master.scp.retain.assignment configuration properties in Cloudera Manager, or enabling the cache-aware load balancer, so that regions can have the same assignment distribution after restarts.

For more information on setting the hbase.master.scp.retain.assignment.force and hbase.master.scp.retain.assignment properties, see *Configuration details*.

For more information on HBase cache-aware load balancer, see *HBase cache-aware load balancer configuration*.

The related Apache JIRAs are, *HBASE-27686* and *HBASE-27743*.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Select the HBase service.
3. Go to Configuration Advanced HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml .
4. Set the value of hbase.bucketcache.persistent.path as PATH_IN_LOCAL_DISK to enable the persistent BucketCache. By default, it is disabled.

Related Information

Configuration details

HBase cache-aware load balancer configuration

HBASE-27686

HBASE-27743

Configuration details

Know the additional configuration details related to the HBase BucketCache configuration.

Following are the two different options for configuring the persistent BucketCache:

- Use the retain assignment properties: This option forces the regions assigned on the restarted RegionServer to wait until these RegionServers complete initialization so that they can be reassigned to the same servers, and can leverage the persistent cache. This option has the disadvantage of keeping some regions unavailable for a certain period or until the RegionServers complete initialization.
- Use the cache-aware load balancer: This option considers cache allocations when deciding on a new assignment plan. The disadvantage is that there might be a temporary performance degradation for read requests on a minor portion of the dataset, as some regions might still be moved to different RegionServers.

You must consider your use case requirements when choosing between these options. The retain properties option becomes more suitable if a consistent SLA is preferred over availability. If availability is critical and temporary performance deviations can be tolerated, the cache-aware load balancer option is preferred.

Related Information

Configuring HBase persistent BucketCache

Using the retain assignment properties

Learn how to configure the retain assignment properties.

About this task

This configuration might delay the region assignment until the given RegionServer reports itself as online to the master. The *region in transition* warning may be reported on the master UI or by the HBCK tool.

Procedure

1. Log in to the Cloudera Manager as an Administrator.
2. Select the HBase service.
3. Go to Configuration Advanced HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml .
4. Set the values of hbase.master.scp.retain.assignment and hbase.master.scp.retain.assignment.force as true.

What to do next

The amount of time the master region assignment background process tries to open the region on the given RegionServer is determined by the hbase.master.scp.retain.assignment.force.retries property (default value 600). Between each retry, the master region assignment background process sleeps for an exponential factor of the value defined in the hbase.master.scp.retain.assignment.force.wait-interval property (default value 50) in milliseconds.

Using the cache-aware load balancer

Learn how to configure the HBase cache-aware load balancer.

About this task

This option uses the HBase cache-aware balancer implementation that considers the cache allocation when defining a new region assignment. This avoids the assignment delays observed while using the retain assignment option.

Procedure

1. Log in to the Cloudera Manager as an Administrator.
2. Select the HBase service.

3. Go to Configuration Advanced HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml .
4. Set the value of hbase.master.loadbalancer.class as org.apache.hadoop.hbase.master.balancer.CacheAwareLoadBalancer to enable the HBase cache-aware load balancer.

   For more information, see *HBase cache-aware load balancer configuration*.

Related Information

HBase cache-aware load balancer configuration

Validating the expected behavior

Ensure to validate the BucketCache functionality using Cloudera Manager.

About this task

After you implement the persistent BucketCache functionality, the restart of RegionServers must have a little impact on the cache allocation.

Procedure

1. Log in to the Cloudera Manager as an Administrator.
2. Validate the BucketCache functionality.

   In scenarios, where the dataset is fully cached, the Cloudera Manager chart for the block cache, as shown in the following example, shows a flat line spanning over restart periods (assuming either the additional configuration option *Using the retain assignment properties* or *Using the cache-aware load balancer* is also applied).

Using quota management

Two types of HBase quotas are well established: throttle quota and number-of tables-quota. These two quotas can regulate users and tables.

In a multitenant HBase environment, ensuring that each tenant can use only its allotted portion of the system is key in meeting SLAs.

Table 2: Quota Support Matrix

Quota Type	Resource Type	Purpose	Namespace applicable?	Table applicable?	User applicable?
Throttle	Network	Limit overall network throughput and number of RPC requests	Yes	Yes	Yes
New space	Storage	Limit amount of storage used for table or namespaces	Yes	Yes	No
Number of tables	Metadata	Limit number of tables for each namespace or user	Yes	No	Yes
Number of regions	Metadata	Limit number of regions for each namespace	Yes	No	No

Configuring quotas

HBase quotas are disabled by default. To enable quotas, the relevant hbase-site.xml property must be set to true and the limit of each quota specified on the command line.

Before you begin

hbase superuser privileges

Procedure

1. Set the hbase.quota.enabled property in the hbase-site.xml file to true.
2. Enter the command to set the limit of the quota, type of quota, and to which entity to apply the quota. The command and its syntax are:

```
$hbase_shell> set_quota TYPE =>

    quota_type,
    arguments
```

General Quota Syntax

The general quota syntax are `THROTTLE_TYPE`, `Request sizes and space limit`, `Number of requests`, `Time limits` and `Number of tables or regions`.

THROTTLE_TYPE

Can be expressed as READ-only, WRITE-only, or the default type (both READ and WRITE permissions)

Timeframes

Can be expressed in the following units of time:

- sec (second)

  min (minute)

  hour

  day

Request sizes and space limit

Can be expressed in the following units:

- B: bytes

  K: kilobytes

  M: megabytes

  G: gigabytes

  P: petabytes

When no size units is included, the default value is bytes.

Number of requests

Expressed as integer followed by the string request

Time limits

Expressed as requests per unit-of-time or size per unit-of-time

Examples: 10req/day or 100P/hour

Number of tables or regions

Expressed as integers

Throttle quotas

The throttle quota, also known as RPC limit quota, is commonly used to manage length of RPC queue as well as network bandwidth utilization.

It is best used to prioritize time-sensitive applications to ensure latency SLAs are met.

Throttle quota examples

Following examples details the usage of adding throttle quotas commands, listing throttle quotas commands, and updating and deleting throttle quotas commands.

Examples of Adding Throttle Quotas Commands

Limit user u1 to 10 requests per second globally:

```
hbase> set_quota => TYPE => THROTTLE, USER => 'u1', LIMIT => '10req/sec'
```

Note: When you exceed the throttle quota limit defined under the LIMIT option, the command fails with an error.

Limit user u1 to up to 10MB of traffic per second globally:

```
hbase> set_quota => TYPE => THROTTLE, USER => 'u1', LIMIT => '10M/sec'
```

Limit user u1 to 10 requests/second globally for read operations. User u1 can still issue unlimited writes:

```
hbase> set_quota TYPE => THROTTLE, THROTTLE_TYPE => READ, USER => 'u1', LIMI
T => '10req/sec'
```

Limit user u1 to 10 requests/second globally for read operations. User u1 can still issue unlimited reads:

```
hbase> set_quota TYPE => THROTTLE, THROTTLE_TYPE => WRITE, USER => 'u1', LIM
IT => '10M/sec'
```

Limit user u1 to 5 KB/second for all operations on table t2. User u1 can still issue unlimited requests for other tables, regardless of type of operation:

```
hbase> set_quota TYPE => THROTTLE, USER => 'u1', TABLE => 't2', LIMIT => '5K
/min'
```

Limit request to namespaces:

```
hbase> set_quota TYPE => THROTTLE, NAMESPACE => 'ns1', LIMIT => '10req/sec'
```

Limit request to tables:

```
hbase> set_quota TYPE => THROTTLE, TABLE => 't1', LIMIT => '10M/sec'
```

Limit requests based on type, regardless of users, namespaces, or tables:

```
hbase> set_quota TYPE => THROTTLE, THROTTLE_TYPE => WRITE, TABLE => 't1', LI
MIT => '10M/sec'
```

Examples of Listing Throttle Quotas Commands

Show all quotas:

```
hbase> list_quotas
```

Show all quotas applied to user bob:

```
hbase> list_quotas USER => 'bob.*'
```

Show all quotas applied to user bob and filter by table or namespace:

```
hbase> list_quotas USER => 'bob.*', TABLE => 't1'
hbase> list_quotas USER => 'bob.*', NAMESPACE => 'ns.*'
```

Show all quotas and filter by table or namespace:

```
hbase> list_quotas TABLE => 'myTable'
hbase> list_quotas NAMESPACE => 'ns.*'
```

Examples of Updating and Deleting Throttle Quotas Commands

To update a quota, simply issue a new `set_quota` command. To remove a quota, you can set LIMIT to NONE. The actual quota entry will not be removed, but the policy will be disabled.

```
hbase> set_quota TYPE => THROTTLE, USER => 'u1', LIMIT => NONE
```

```
hbase> set_quota TYPE => THROTTLE, USER => 'u1', NAMESPACE => 'ns2', LIMIT =
> NONE
```

```
hbase> set_quota TYPE => THROTTLE, THROTTLE_TYPE => WRITE, USER => 'u1', LIM
IT => NONE
```

```
hbase> set_quota USER => 'u1', GLOBAL_BYPASS => true
```

Space quotas

Space quotas, also known as filesystem space quotas, limit the amount of stored data. It can be applied at a table or namespace level where table-level quotas take priority over namespace-level quotas.

Space quotas are special in that they can trigger different policies when storage goes above thresholds. The following list describes the policies, and they are listed in order of least strict to most strict:

NO_INSERTS

Prohibits new data from being ingested (for example, data from `Put`, `Increment`, and `Append` operations are not ingested).

NO_WRITES

Performs the same function as NO_INSERTS but `Delete` operations are also prohibited.

NO_WRITES_COMPACTIONS

Performs the same function as NO_INSERTS but compactions are also prohibited.

DISABLE

Diables tables.

Examples of Adding Space Quotas

Add quota with the condition that Insert operations are rejected when table t1 reaches 1 GB of data:

```
hbase> set_quota TYPE => SPACE, TABLE => 't1', LIMIT => '1G', POLICY => NO_I
NSERTS
```

Add quota with the condition that table t2 is disabled when 50 GB of data is exceeded:

```
hbase> set_quota TYPE => SPACE, TABLE => 't2', LIMIT => '50G', POLICY => DIS
ABLE
```

Add quota with the condition that Insert and Delete operations cannot be applied to namespace ns1 when it reaches 50 terabytes of data:

```
hbase> set_quota TYPE => SPACE, NAMESPACE => 'ns1', LIMIT => '50T', POLICY =
> NO_WRITES
```

Listing Space Quotas

See "Examples of Listing Throttle Quotas Commands" above for the supported syntax.

Examples of Updating and Deleting Space Quotas

A quota can be removed by setting LIMIT to NONE.

```
hbase> set_quota TYPE => SPACE, TABLE => 't1', LIMIT => NONE
```

```
hbase> set_quota TYPE => SPACE, NAMESPACE => 'ns1', LIMIT => NONE
```

Quota enforcement

When a quota limit is exceeded, the Master server instructs RegionServers to enable an enforcement policy for the namespace or table that violated the quota.

It is important to note the storage quota is not reported in real-time. There is a window when threshold is reached on RegionServers but the threshold accounted for on the Master server is not updated.

Note:

Set a storage limit lower than the amount of available disk space to provide extra buffer.

Quota violation policies

If quotas are set for the amount of space each HBase tenant can fill on HDFS, then a quota violation policy should be planned and implemented on the system.

When a quota violation policy is enabled, the table owner should not be allowed to remove the policy. The expectation is that the Master automatically removes the policy. However, the HBase superuser should still have permission.

Automatic removal of the quota violation policy after the violation is resolved can be accomplished via the same mechanisms that it was originally enforced. But the system should not immediately disable the violation policy when the violation is resolved.

The following describes quota violation policies that you might consider.

Disabling Tables

This is the “brute-force” policy, disabling any tables that violated the quota. This policy removes the risk that tables over quota affect your system. For most users, this is likely not a good choice as most sites want READ operations to still succeed.

One hypothetical situation when a disabling tables policy might be advisable is when there are multiple active clusters hosting the same data and, because of a quota violation, it is discovered that one copy of the data does not have all of the data it should have. By disabling tables, you can prevent further discrepancies until the administrator can correct the problem.

Rejecting All WRITE Operations, Bulk Imports, and Compactions

This policy rejects all WRITEs and bulk imports to the region which the quota applies. Compactions for this region are also disabled to prevent the system from using more space because of the temporary space demand of a compaction. The only resolution in this case is administrator intervention to increase the quota that is being exceeded.

Rejecting All WRITE Operations and Bulk Imports

This is the same as the previous policy, except that compactions are still allowed. This allows users to set or alter a TTL on table and then perform a compaction to reduce the total used space. Inherently, using this violation policy means that you let used space to slightly rise before it is ultimately reduced.

Allowing DELETE Operations But Rejecting WRITE Operations and Bulk Imports

This is another variation of the two previously listed policies. This policy allows users to run processes to delete data in the system. Like the previous policy, using this violation policy means that you let used space slightly rises before

it is ultimately reduced. In this case, the deletions are propagated to disk and a compaction actually removes data previously stored on disk. TTL configuration and compactions can also be used to remove data.

Impact of quota violation policy

Quota violation policies can impact live write access, bulk write access, and read access. You must understand what the quota violation policies mean for your deployment before you plan and implement it on your system.

Live write access

You must understand how a quota violation policy configuration affects your ability to write data to HBase.

Every violation policy disables the ability to write new data into the system. This means that any operation on the data other than DELETE operation could be rejected by HBase.

Bulk Write Access

Bulk loading HFiles can be an extremely effective way to increase the overall throughput of ingest into HBase. Quota management when bulk loading is important because large HFiles have the potential to quickly violate a quota.

Clients group HFiles by region boundaries and send the file for each column family to the RegionServer presently hosting that region. The RegionServer ultimately inspects each file, ensuring that it should be loaded into this region, and then, sequentially, load each file into the correct column family.

As a part of the precondition-check of the file's boundaries before loading it, the quota state should be inspected to determine if loading the next file will violate the quota. If the RegionServer determines that it will violate the quota, it should not load the file and inform the client that the file was not loaded because it would violate the quota.

Read access

In most cases, quota violation policies can affect the ability to read the data stored in HBase. You must understand how a quota violation policy affects your ability to read the data stored in HBase.

In most cases, quota violation policies can affect the ability to read the data stored in HBase. A goal of applying these HBase quotas is to ensure that the storage remains healthy and sustains a higher level of availability to HBase users. Guaranteeing that there is always free space in your HDFS can yield a higher level of health of the physical machines and the DataNodes. This leaves the HDFS-reserved space percentage as a fail-safe mechanism.

Metrics and Insight

You can view the quotas and metrics about the quotas in the HBase Master user interface. The list of defined quotas are displayed along with those quotas whose violation policy is being enforced.

The list of tables/namespaces with enforced violation policies is also available in the the JMX metrics exposed by the Master.

Examples of overlapping quota policies

With the ability to define a quota policy on namespaces and tables, you have to define how the policies are applied. A table quota should take precedence over a namespace quota.

Scenario 1

For example, consider Scenario 1, which is outlined in the following table. Namespace n has the following collection of tables: n1.t1, n1.t2, and n1.t3. The namespace quota is 100 GB. Because the total storage required for all tables is less than 100 GB, each table can accept new WRITEs.

Table 3: Scenario 1: Overlapping Quota Policies

Object	Quota	Storage Utilization
Namespace n1	100 GB	80 GB

Object	Quota	Storage Utilization
Table n1.t1	10 GB	5 GB
Table n1.t2	(not set)	50 GB
Table n1.t3	(not set)	25 GB

Scenario 2

In Scenario 2, as shown in the following table, WRITEs to table n1.t1 are denied because the table quota is violated, but WRITEs to tablen1.t2 and table n1.t3 are still allowed because they are within the namespace quota. The violation policy for the table quota on table n1.t1 is enacted.

Table 4: Scenario 2: Overlapping Quota Policies

Object	Quota	Storage Utilization
Namespace n1	100 GB	60 GB
Table n1.t1	10 GB	15 GB
Table n1.t2	(not set)	30 GB
Table n1.t3	(not set)	15 GB

Scenario 3

In the Scenario 3 table below, WRITEs to all tables are not allowed because the storage utilization of all tables exceeds the namespace quota limit. The namespace quota violation policy is applied to all tables in the namespace.

Table 5: Scenario 3: Overlapping Quota Policies

Object	Quota	Storage Utilization
Namespace n1	100 GB	108 GB
Table n1.t1	10 GB	8 GB
Table n1.t2	(not set)	50 GB
Table n1.t3	(not set)	50 GB

Scenario 4

In the Scenario 4 table below, table n1.t1 violates the quota set at the table level. The table quota violation policy is enforced. In addition, the disk utilization of table n1t1 plus the sum of disk utilization for table n1t2 and table n1t3 exceeds the 100 GB namespace quota. Therefore, the namespace quota violation policy is also applied.

Table 6: Scenario 4: Overlapping Quota Policies

Object	Quota	Storage Utilization
Namespace n1	100 GB	115 GB
Table n1.t1	10 GB	15 GB
Table n1.t2	(not set)	50 GB
Table n1.t3	(not set)	50 GB

Number-of-Tables Quotas

The number-of-tables quota is set as part of the namespace metadata and does not involve the `set_quota` command.

Examples of Commands Relevant to Setting and Administering Number-of-Tables Quotas

Create namespace ns1 with a maximum of 5 tables

```
hbase> create_namespace 'ns1', {'hbase.namespace.quota.maxtables'=>'5'}
```

Alter an existing namespace ns1 to set a maximum of 8 tables

```
hbase> alter_namespace 'ns1', {METHOD => 'set', 'hbase.namespace.quota.maxta
bles'=>'8'}
```

Show quota information for namespace ns1

```
hbase> describe_namespace 'ns1'
```

Alter existing namespace ns1 to remove a quota

```
hbase> alter_namespace 'ns1', {METHOD => 'unset', NAME=>'hbase.namespace.quo
ta.maxtables'}
```

Number-of-Regions Quotas

The number-of-regions quota is similar to the number-of-tables quota. The number-of-regions quota is set as part of the namespace metadata and does not involve the `set_quota` command.

Examples of Commands Relevant to Setting and Administering Number-of-Regions Quotas

Create namespace ns1 with a maximum of 5 tables

```
hbase> create_namespace 'ns1', {'hbase.namespace.quota.maxregions'=>'5'}
```

Alter an existing namespace ns1 to set a maximum of 8 regions

```
hbase> alter_namespace 'ns1', {METHOD => 'set', 'hbase.namespace.quota.maxre
gions'=>'8'}
```

Show quota information for namespace ns1

```
hbase> describe_namespace 'ns1'
```

Alter existing namespace ns1 to remove a quota

```
hbase> alter_namespace 'ns1', {METHOD => 'unset', NAME=>'hbase.namespace.quo
ta.maxregions'}
```

Using HBase scanner heartbeat

A scanner heartbeat check enforces a time limit on the execution of scan RPC requests. This helps prevent scans from taking too long and causing a timeout at the client.

When the server receives a scan RPC request, a time limit is calculated to be half of the smaller of two values: hbase.client.scanner.timeout.period and hbase.rpc.timeout (which both default to 60000 milliseconds, or one minute). When the time limit is reached, the server returns the results it has accumulated up to that point. This result set may be empty. If your usage pattern includes that scans will take longer than a minute, you can increase these values.

To make sure the timeout period is not too short, you can configure hbase.cells.scanned.per.heartbeat.check to a minimum number of cells that must be scanned before a timeout check occurs. The default value is 10000. A smaller value causes timeout checks to occur more often.

Configure the scanner heartbeat using Cloudera Manager

You can configure the HBase scanner heartbeat using Cloudera Manager.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select HBase or HBase Service-Wide.
4. Locate the RPC Timeout property by typing its name in the Search box, and edit the property.
5. Locate the HBase RegionServer Lease Period property by typing its name in the Search box, and edit the property.
6. Enter a Reason for change, and then click Save Changes to commit the changes.
7. Restart the role.
8. Restart the service.

Storing medium objects (MOBs)

Medium Object Storage (MOB) is a feature in Apache HBase that helps you store medium-size objects in the size of 100 KB to 10 MB. You can use this to feature to store documents, images, and other moderately-sized objects.

Data comes in many sizes. You can save different kinds of data in HBase, including binary objects such as images and documents. HBase can technically handle binary objects with cells that are up to 10 MB. However, HBase normal read and write paths are optimized for values smaller than 100 KB. When HBase handles a large number of objects up to 10 MB, the performance is degraded because of write amplification caused by splits and compactions. MOB operates by storing a reference of the object data within the main table. The reference in the table points to external HFiles that contain the actual data, which can be on any storage.

MOB support must be enabled on individual column families within a table. You can do this either through the HBase shell or using the Java API. MOB settings can be configured at table creation time or can be modified on an existing table's column family.

Cloudera OpDB has a new feature called distributed MOB compaction. This feature overcomes a drawback of the older implementations of MOB compaction by moving maintenance of MOB data files from a centralized process handled by the HBase Master to a parallel process that is distributed across the RegionServers.

If you are currently using MOB compaction in an older version of Cloudera Runtime, CDH, or HDP, you must be aware of the following changes when you use distributed MOB compaction in Cloudera:

- You can no longer set MOB compaction policies
- The storage of MOB values is no longer grouped by the date of the original cell's timestamp according to said compaction policies, daily, or otherwise. Instead, they are grouped by the region that performed the most recent maintenance write of the backing MOB data file.
- The MOB system no longer tracks the deletion of individual cells through the use of special files in the MOB storage area with the suffix_del. After upgrading, you must manually move these files.
- Under the default configuration, the MOB system attempts to maximize the throughput for the compaction of MOB stored values. This means that it will take much less time to perform a given compaction of MOB stored values. However, this change places a much larger load on the underlying filesystem when compared to the HBase Master handled MOB compaction.

- When the MOB system detects that a table has HFiles with references to the MOB data but the reference HFiles do not yet have the needed file-level metadata then it does not archive any MOB HFiles from that table. The reference files will be updated as a part of normal HBase maintenance operations over time.

Prerequisites

You must be aware of the following prerequisites before you configure HBase to store MOBs.

Before you configure HBase to store MOBs, you need:

- HBase superuser privileges
- HFile version 3

Configure columns to store MOBs

You can configure a column to store MOBs using the HBase Shell or the Java API.

About this task

Use the following options in the HBase Shell or Java API to configure a column to store MOBs:

- IS_MOB specifies whether or not the column can store MOBs. This is a Boolean option where you can set it to true or false.
- MOB_THRESHOLD configures the number of bytes at which an object is considered to be a MOB. If you do not specify a value for MOB_THRESHOLD, the default is 100 KB. If you write a value larger than this threshold, it is treated as a MOB.

Procedure

- Using the HBase Shell:

```
hbase> create 't1', {NAME => 'f1', IS_MOB => true, MOB_THRESHOLD => 1024
00}
hbase> alter 't1', {NAME => 'f1', IS_MOB => true, MOB_THRESHOLD => 102400}
```

- Usin the Java API:

```
HColumnDescriptor hcd = new HColumnDescriptor("f");
hcd.setMobEnabled(true);
hcd.setMobThreshold(102400L);
```

Configure the MOB cache using Cloudera Manager

To configure the MOB cache within Cloudera Manager, edit the HBase Service advanced configuration snippet for the cluster. Cloudera recommends testing your configuration with the default settings first.

Procedure

1. Go to the HBase service.
2. Click Configuration.
3. Search for the property HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml.
4. Paste your configuration into the Value field and save your changes. The following example sets the hbase.mob.cache.evict.period property to 5000 seconds. See *MOB Cache Properties* for a list of configurable properties.

```
<property>
```

```
  <name>hbase.mob.cache.evict.period</name>
  <value>5000</value>
</property>
```

5. Restart your cluster for the changes to take effect.

Related Information

MOB cache properties

Test MOB storage and retrieval performance

You can test the MOB storage and retrieval performance by using the Apache HBase Load Test Tool (ltt), and by configuring it to generate MOB values.

About this task

Procedure

- Run this HBase load test tool command in your command-line interface:

```
$ hbase ltt -mob_threshold 102400 -generator \
org.apache.hadoop.hbase.util.LoadTestDataGeneratorWithMOB:medium_column:1
02400:104857 \
    -tn example_table -families small_column,medium_column
    -num_keys 10000 -write 3:1024
```

Important: This command must be run by a user who has the permission to create the passed table, or the table must be created earlier and the user should be given the required permissions.

- -mob_threshold is the size (in bytes) at which an object is considered to be a MOB. You must set this threshold for the tool to configure column families as MOB-enabled.
- -generator is the class that generates load for the tool. You must set this parameter to tell the tool how the values must be created. For example, when you set this as LoadTestDataGeneratorWithMOB:medium_column:102400:104857, the parameter accepts the name of a column family, and a range of data sizes for that column. This option pseudo-randomly chooses to make cells in that column family that are of this size range (102400:104857, rather than the default cell size.
- -tn is the name of the table to create, and use
- -families is a list of column families to create on the table you provided in using the -tn parameter.
- -num_keys is the number of rows to write.
- -write is the upper bound on the number of cells to write per column family and the average size of those cells.

MOB cache properties

Opening a MOB file places the corresponding HFile-formatted data in active memory. Too many open MOB files can cause a RegionServer to exceed the memory capacity and cause performance degradation. To minimize the possibility of this issue arising on a RegionServer, you must tune the MOB file reader cache to a size that HBase can scale.

The MOB file reader cache is a least recently used (LRU) cache that keeps only the most recently used MOB files open. Refer to the MOB Cache Properties table for variables that can be tuned in the cache. MOB file reader cache configuration is specific to each RegionServer, so assess and change, if needed, each RegionServer individually. You must manually add any of the following properties you may require in the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property.

The following properties are available for tuning the HBase MOB cache.

Property	Default	Description

hbase.mob.file.cache.size	1000	The opened file handlers to cache. A larger value benefits read operations by providing more file handlers per MOB file cache and reduces frequent file opening and closing. However, if the value is too high, errors such as "Too many opened file handlers" might be logged.
hbase.mob.cache.evict.period	3600	The amount of time in seconds after a file is opened before the MOB cache evicts cached files.
hbase.mob.cache.evict.remain.ratio	0.5f	The ratio expressed as a float between 0.0 and 1.0, that controls how many files remain cached after an eviction is triggered due to the number of cached files exceeding the value assigned to the hbase.mob.file.cache.size property.

Related Information

Configure the MOB cache using Cloudera Manager

Limiting the speed of compactions

You can limit the speed at which HBase compactions run, by configuring hbase.regionserver.throughput.controller and its related settings.

The default controller is org.apache.hadoop.hbase.regionserver.throttle.PressureAwareCompactionThroughputController, which uses the following algorithm:

- If compaction pressure is greater than 1.0, there is no speed limitation.
- In off-peak hours, use a fixed throughput limitation, configured using hbase.hstore.compaction.throughput.offpeak, hbase.offpeak.start.hour, and hbase.offpeak.end.hour.
- In normal hours, the max throughput is tuned between hbase.hstore.compaction.throughput.higher.bound and hbase.hstore.compaction.throughput.lower.bound (which default to 20 MB/sec and 10 MB/sec respectively), using the following formula, where compactionPressure is between 0.0 and 1.0. The compactionPressure refers to the number of store files that require compaction.

```
lower + (higher - lower) * compactionPressure
```

To disable compaction speed limits, set hbase.regionserver.throughput.controller to org.apache.hadoop.hbase.regionserver.throttle.NoLimitThroughputController.

Configure the compaction speed using Cloudera Manager

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select HBase or HBase Service-Wide.
4. Search for HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml. Paste the relevant properties from the following example into the field and modify the values as needed:

```
<property>
  <name>hbase.hstore.compaction.throughput.higher.bound</name>
```

```
  <value>52428800</value>
  <description>The default is 50 MiB/sec</description>
</property>
<property>
  <name>hbase.hstore.compaction.throughput.lower.bound</name>
  <value>104857600</value>
  <description>The default is 100 MiB/sec</description>
</property>
<property>
  <name>hbase.hstore.compaction.throughput.offpeak</name>
  <value>104857600</value>
  <description>The default is 100 MiB/sec</description>
</property>
<property>
  <name>hbase.offpeak.start.hour</name>
  <value>20</value>
  <description>The start of off-peak hours, expressed as an integer betwe
en 0 and 23, inclusive.
      Set to -1 to disable off-peak.</description>
</property>
<property>
  <name>hbase.offpeak.end.hour</name>
  <value>6</value>
  <description>The end of off-peak hours, expressed as an integer between
 0 and 23, inclusive. Set
      to -1 to disable off-peak.</description>
</property>
<property>
  <name>hbase.hstore.compaction.throughput.tune.period</name>
  <value>60000</value>
</property>
```

5. Enter a Reason for change, and then click Save Changes to commit the changes.
6. Restart the service.

Enable HBase indexing

You can enable HBase indexing using Cloudera Manager.

Before you begin

HBase indexing is dependent on the Key-Value Store Indexer service.

Procedure

1. Go to the HBase service.
2. Select ScopeHBASE-1 (Service Wide)
3. Select CategoryBackup.
4. Select the Enable Replication and Enable Indexing properties.
5. Click Save Changes.

Using HBase coprocessors

You can configure HBase coprocessors to run your own custom code. The HBase coprocessor framework provides a way to extend HBase with custom functionality. Coprocessors provide a way to run server-level code against locally-stored data.

Coprocessors are not designed to be used by end users of HBase, but by HBase developers who need to add specialized functionality to HBase. One example of the use of coprocessors is pluggable compaction and scan policies.

Related Information

Apache HBase blog: Coprocessor Introduction

Add a custom coprocessor

You can add a custom coprocessor to to extend HBase with custom functionality using Cloudera Manager.

About this task

To configure these properties in Cloudera Manager:

Procedure

1. Select the HBase service.
2. Click the Configuration tab.
3. Select Scope All .
4. Select Category All .
5. Type HBase Coprocessor in the Search box.
6. You can configure the values of the following properties:
   - HBase Coprocessor Abort on Error (Service-Wide)
   - HBase Coprocessor Master Classes (Master Default Group)
   - HBase Coprocessor Region Classes (RegionServer Default Group)
7. Enter a Reason for change, and then click Save Changes to commit the changes.

Disable loading of coprocessors

You can disable loading of coprocessors using Cloudera Manager.

About this task

Cloudera recommends against disabling loading of system coprocessors, because HBase security functionality is implemented using system coprocessors. However, disabling loading of user coprocessors may be appropriate.

Procedure

1. Select the HBase service.
2. Click the Configuration tab.
3. Search for HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml.
4. To disable loading of all coprocessors, add a new property with the name hbase.coprocessor.enabled and set its value to false. Cloudera does not recommend this setting.
5. To disable loading of user coprocessors, add a new property with the name hbase.coprocessor.user.enabled and set its value to false.
6. Enter a Reason for change, and then click Save Changes to commit the changes.

Configuring HBase MultiWAL

You can configure multiple write-ahead logs (MultiWAL) for HBase. If you do not configure MultiWAL, each region on a RegionServer writes to the same WAL.

A busy RegionServer might host several regions, and each write to the WAL is serial because HDFS only supports sequentially written files. This causes the WAL to negatively impact performance.

MultiWAL allows a RegionServer to write multiple WAL streams in parallel by using multiple pipelines in the underlying HDFS instance, which increases total throughput during writes.

Note: In the current implementation of MultiWAL, incoming edits are partitioned by Region. Therefore, throughput to a single Region is not increased.

To configure MultiWAL for a RegionServer, set the value of the property hbase.wal.provider to multiwal and restart the RegionServer. To disable MultiWAL for a RegionServer, unset the property and restart the RegionServer.

RegionServers using the original WAL implementation and those using the MultiWAL implementation can each handle recovery of either set of WALs, so a zero-downtime configuration update is possible through a rolling restart.

Configuring MultiWAL support using Cloudera Manager

You can configure MultiWAL using Cloudera Manager.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope RegionServer .
4. Select Category Main .
5. Set WAL Provider to MultiWAL.
6. Set the Per-RegionServer Number of WAL Pipelines to a value greater than 1.
7. Enter a Reason for change, and then click Save Changes to commit the changes.
8. Restart the RegionServer roles.

Configuring the storage policy for the Write-Ahead Log (WAL)

You can configure the preferred HDFS storage policy for HBase's write-ahead log (WAL) replicas. This feature allows you to tune HBase's use of SSDs to your available resources and the demands of your workload.

These instructions assume that you have followed the instructions to configure storage directories for DataNodes, and that your cluster has SSD storage available to HBase. If HDFS is not configured to use SSDs, these configuration changes will have no effect on HBase. The following policies are available:

- NONE: no preference about where the replicas are written.
- ONE_SSD: place one replica on SSD storage and the remaining replicas in default storage. This allows you to derive some benefit from SSD storage even if it is a scarce resource in your cluster.

  Warning: ONE_SSD mode has not been thoroughly tested with HBase and is not recommended.

- ALL_SSD: place all replicas on SSD storage.

Configure the storage policy for WALs using Cloudera Manager

You can configure the preferred HDFS storage policy for HBase's write-ahead log (WAL) replicas using Cloudera Manager.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Search for the property WAL HSM Storage Policy.
4. Select your desired storage policy.
5. Save your changes. Restart all HBase roles.

Results

Changes will take effect after the next major compaction.

Configure the storage policy for WALs using the Command Line

You can configure the preferred HDFS storage policy for HBase's write-ahead log (WAL) replicas using the command line.

About this task

Important: Follow these command-line instructions on systems that do not use Cloudera Manager.

Procedure

- Paste the following XML into hbase-site.xml. Uncomment the <value> line that corresponds to your desired storage policy.

```
<property>
  <name>hbase.wal.storage.policy</name>
  <value>NONE</value>
  <!--<value>ONE_SSD</value>-->
  <!--<value>ALL_SSD</value>-->
</property>
```

  Warning: ONE_SSD mode has not been thoroughly tested with HBase and is not recommended.

- Restart HBase. Changes will take effect for a given region during its next major compaction.

Using RegionServer grouping

You can use RegionServer Grouping (rsgroup) to impose strict isolation between RegionServers by partitioning RegionServers into distinct groups. You can use HBase Shell commands to define and manage RegionServer Grouping.

You must first create an rsgroup before you can add RegionServers to it. Once you have created an rsgroup, you can move your HBase tables into this rsgroup so that only the RegionServers in the same rsgroup can host the regions of the table.

Note: RegionServers and tables can only belong to one rsgroup at a time. By default, all the tables and RegionServers belong to the default rsgroup.

A custom balancer implementation tracks assignments per rsgroup and moves regions to the relevant RegionServers in that rsgroup. The rsgroup information is stored in a regular HBase table, and a ZooKeeper-based read-only cache is used at cluster bootstrap time.

Enable RegionServer grouping using Cloudera Manager

You must use Cloudera Manager to enable RegionServer Grouping before you can define and manage rsgroups.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope Master .
4. Locate the HBase Coprocessor Master Classes property or search for it by typing its name in the Search box.
5. Add the following property value: org.apache.hadoop.hbase.rsgroup.RSGroupAdminEndpoint.
6. Locate the Master Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the Search box.
7. Click View as XML and add the following property:

```
<property>
  <name>hbase.master.loadbalancer.class</name>
  <value>org.apache.hadoop.hbase.rsgroup.RSGroupBasedLoadBalancer</value>
</property>
```

8. Enter a Reason for change, and then click Save Changes to commit the changes.
9. Restart the role.
10. Restart the service.

Configure RegionServer grouping

When you add a new rsgroup, you are creating an rsgroup other than the default group.

About this task

To configure a rsgroup, in the HBase shell:

Procedure

1. Add an rsgroup: $hbase> add_rsgroup '*MYGROUP*'.
2. Add RegionServers and tables to this rsgroup: $hbase> move_servers_tables_rsgroup '*MYGROUP*', ['*SERVER1:PORT*','*SERVER2:PORT*'],['*TABLE1*','*TABLE2*'].
3. Run the balance_rsgroup command if the tables are slow to migrate to the group's dedicated server.

   Note: The term *rsgroup* refers to servers in a cluster with only the hostname and port. It does not make use of the HBase ServerName type identifying RegionServers (hostname + port + start time) to distinguish RegionServer instances.

Monitor RegionServer grouping

You can monitor the status of the commands using the Tables tab on the HBase Master UI home page.

You can monitor the status of the commands using the Tables tab on the HBase Master UI home page. If you click on a table name, you can see the RegionServers that are deployed.

You must manually align the RegionServers referenced in rsgroups with the actual state of nodes in the cluster that is active and running.

Remove a RegionServer from RegionServer grouping

You can remove a RegionServer by moving it to the default rsgroup. Edits made using shell commands to all rsgroups, except the default rsgroup, are persisted to the system hbase:rsgroup table. If an rsgroup references a decommissioned RegionServer, then the rsgroup should be updated to undo the reference.

Procedure

1. Move the RegionServer to the default rsgroup using the command: $hbase> move_servers_rsgroup 'default', ['*SERVER1:PORT*'].
2. Check the list of RegionServers in your rsgourp to ensure that that the RegionServer is successfully removed using the command: $hbase> get_rsgroup '*MYGROUP*'

   The default rsgroup's RegionServer list mirrors the current state of the cluster. If you shut down a RegionServer that was part of the default rsgroup, and then run the get_rsgroup 'default' command to list its content in the shell, the server is no longer listed. If you move the offline server from the non-default rsgroup to default, it will not show in the default list; the server will just be removed from the list.

Enabling ACL for RegionServer grouping

You can enable ACL for RegionServer grouping using Cloudera Manager. If authorization is enabled only Global Admins can manage rsgroups.

Procedure

1. In Cloudera Manager navigate to HBase Configuration .
2. Find the Master Advanced Configuration Snippet (Safety-Valve) for hbase-site.xml property.
3. Click the plus icon to add a new property:
   - Name: hbase.security.authorization
   - Value: true
4. Click Save Changes.
5. Restart your HBase Master server.

Best practices when using RegionServer grouping

You must keep in mind the following best practices when using rsgroups.

- Isolate system tables: You can either have a system rsgroup where all the system tables are present or just leave the system tables in default rsgroup and have all user-space tables in non-default rsgroups.
- Handle dead nodes: You can have a special rsgroup of dead or questionable nodes to help you keep them without running until the nodes are repaired. Be careful when replacing dead nodes in an rsgroup, and ensure there are enough live nodes before you start moving out the dead nodes. You can move the good live nodes first before moving out the dead nodes.

  If you have configured a table to be in a rsgroup, but all the RegionServers in that rsgroup die, the tables become unavailable and you can no longer access those tables.

Disable RegionServer grouping

When you no longer require rsgroups, you can disable it for your cluster.

About this task

Removing RegionServer Grouping for a cluster on which it was enabled involves more steps in addition to removing the relevant properties from hbase-site.xml. You must ensure that you clean the RegionServer grouping-related metadata so that if the feature is re-enabled in the future, the old metadata will not affect the functioning of the cluster.

To disable RegionServer Grouping:

Procedure

1. Move all the tables in non-default rsgroups to default RegionServer group.

```
#Reassigning table t1 from the non-default group - hbase shell
hbase> move_tables_rsgroup 'default',['T1']
```

2. Move all RegionServers in non-default rsgroups to default regionserver group.

```
#Reassigning all the servers in the non-default rsgroup to default - hbase
 shell
hbase> move_servers_rsgroup 'default',
['REGIONSERVER1:PORT','REGIONSERVER2:PORT','REGIONSERVER3:PORT']
```

3. Remove all non-default rsgroups. default rsgroup created implicitly does not have to be removed.

```
#removing non-default rsgroup - hbase shell
hbase> remove_rsgroup 'MYGROUP'
```

4. Remove the changes made in hbase-site.xml and restart the cluster.
5. Drop the table hbase:rsgroup from HBase.

```
#Through hbase shell drop table hbase:rsgroup
hbase> disable 'hbase:rsgroup'
0 row(s) in 2.6270 seconds
hbase> drop 'hbase:rsgroup'
0 row(s) in 1.2730 seconds
```

6. Remove the znode rsgroup from the cluster ZooKeeper using zkCli.sh.

```
#From ZK remove the node /hbase/rsgroup through zkCli.sh
rmr /hbase/rsgroup
```

HBase load balancer

The HBase load balancer mechanism ensures that the region replicas are not co-hosted in the same region servers and also in the same rack. It also decides about the placement and movement of Regions across RegionServers.

HBase load balancer supports two balancing mechanisms, *stochastic load balancing* and *cache-aware load balancing*. The default is *stochastic load balancing*.

HBase cache-aware load balancer configuration

HBase supports configuring a cache-aware load balancer which focuses on enhancing the performance by considering the cache allocation of each region on region servers.

Overview

Know the basics of HBase cache-aware load balancer.

In the stochastic load balancer, the region assignment is done based on multiple factors. However, this region assignment does not consider the amount of data already cached on a region server. During the region assignment, if the region is assigned to another region server, the data for this region that is already cached on the old region server has to be thrown away and it has to be prefetched afresh on the newly assigned region server.

HBase uses an ephemeral cache to cache the blocks by reading them from the slow storage and storing them in the bucket cache. This cache is warmed up every time a region server is started. Depending on the data size and the configured cache size, the cache warm-up activity can take anywhere from a few minutes to a few hours. Doing this every time the region server starts can be an expensive process. To eliminate this, HBase implemented the cache persistence feature where the region servers periodically persist the blocks cached in the bucket cache. This persisted information is then used to resurrect the cache in the event of a region server restart because of a normal restart or crash.

HBase implements the cache-aware load balancer which is aimed at enhancing the capability of HBase to enable the balancer to consider the cache allocation of each region on region servers when calculating a new assignment plan. This balancer uses the region or region server cache allocation information reported by region servers to calculate the percentage of HFiles cached for each region on the hosting server, and then use that as another factor when deciding on an optimal new assignment plan.

The master node captures the prefetch information from all the region servers. It uses this information to decide the region assignments while ensuring a minimal impact on the warm-up cache. A region is assigned to the region server where it has a better cache ratio as compared to the region server where it is currently hosted.

Components of cache-aware load balancer

The cache-aware load balancer uses two cost elements for deciding the region allocation, cache cost and skewness cost.

Cache cost

The cache cost is calculated as the percentage of data for a region cached on the region server where it is either currently hosted or was previously hosted. A region may have multiple HFiles, each of different sizes. An HFile is considered to be fully prefetched when all the data blocks in this file are in the cache. The region server hosting this region calculates the ratio of the number of HFiles cached in the bucket cache to the total number of HFiles in the region. This ratio varies from 0 (region hosted on this server and all the HFiles for this region are cached into the bucket cache) to 1 (region hosted on this server, but none of its HFiles are cached into the bucket cache).

Every region server maintains this information for all the regions currently hosted there. In addition to that, this cache ratio is also maintained for the regions that were previously hosted on this region server giving historical information about the regions.

Skewness cost

The skewness cost is calculated as the number of regions hosted on each region server in the cluster. The skewness cost varies from 0 (regions are equally distributed across the region servers) to 1 (regions are not equally distributed across the region servers).

The balancer considers these two costs and calculates the resulting cost of maintaining the balance in the cluster. The balancer attempts to rebalance the cluster under the following conditions:

- An idle server exists in the cluster. This can happen when an existing server is restarted or a new server is added to the cluster.
- When the cost of maintaining the balance in the cluster is greater than the minimum threshold defined by the configuration parameter, hbase.master.balancer.stochastic.minCostNeedBalance.

Configuration details

You must configure the following properties in the HBase cluster to use the cache-aware load balancer.

hbase.master.loadbalancer.class

This property defines the load balancer class to be used in the cluster. The default load balancer used by the cluster is the stochastic load balancer.

The supported value of the property is org.apache.hadoop.hbase.master.balancer.CacheAwareLoadBalancer.

The following is an example of the configuration property.

```
<property>
  <name>hbase.master.loadbalancer.class</name>
  <value>org.apache.hadoop.hbase.master.balancer.CacheAwareLoadBalancer</v
alue>
</property>
```

hbase.bucketcache.persistent.path

This configuration defines the location of the file where the region servers persist the prefetch information. If this configuration is set, the region servers periodically write the metadata about the HFiles cached into the bucket cache. While restarting the region server, this information is reinstated by the region server. The cache-aware load balancer relies on this information to decide on the region assignment. The cache-aware load balancer does not work if this configuration does not exist.

Note: The path of the configuration file is the same across all the region servers in the same cluster.

The following is an example of the configuration property.

```
<property>
  <name>hbase.bucketcache.persistent.path</name>
  <value>/hadoopfs/ephfs1/persistent_cache</value>
</property>
```

HBase stochastic load balancer configuration

HBase supports configuring a stochastic load balancer which works by calculating the cumulative target cost of maintaining the balance across the cluster by considering the cost of individual cost factors.

HBase is a distributed database and hosts data across the region servers in the cluster. This is done by assigning a region to a region server and that region server is responsible for handling that region. For HBase to perform optimally, these regions must be distributed across the region servers in such a way that the whole cluster is used optimally. To do that, HBase implements the stochastic load balancer which takes into account multiple factors to decide how the regions must be distributed across the cluster. The stochastic load balancer works by calculating the cumulative target cost of maintaining the balance across the cluster by considering the cost of individual factors. Eventually, the balancer uses this cost to determine whether the cluster needs to be balanced to make it optimal to use.

Introduction to the HBase stochastic load balancer

Know the basics of HBase stochastic load balancer.

The stochastic load balancer is a best-effort load balancer and considers multiple costs to calculate the target cost of the balancer. Every cost function returns a number between 0 and 1, both inclusive, where 0 is the lowest cost best solution, and 1 is the highest possible cost solution. The computed costs are scaled by their respective multipliers.

Cost functions

The following are the costs considered by the stochastic load balancer.

- Region skewness cost
- Primary region skewness cost
- Table skewness cost
- Move cost
- Locality cost
- Read request cost
- Write request cost

Each of these cost function is assigned a multiplier which decides the impact it makes on the overall cost calculation. The balancer collects information from the cluster, calculates the costs for each of these cost functions, and also calculates the overall target cost of the cluster.

The outcome of these calculations is a region assignment plan which is then used to do the region movement.

Components of stochastic load balancer

Learn about the important cost functions used by the stochastic load balancer, certain important configuration parameters affecting these cost functions, and the generic configuration parameters that determine the way the balancer calculates the target cost of the cluster.

The following are the key cost functions that affect stochastic load balancer. In addition to this, certain configuration parameters that affect the balancer are described as part of the cost functions.

Region count skewness cost

This cost function calculates the cost of the potential cluster state based on the skewness in the number of regions on a cluster. This function returns a value between 0 and 1. A lower value indicates that the regions are evenly balanced across the cluster while a larger value indicates that the regions are unevenly distributed across the cluster.

The following configuration parameter controls the way this cost function works.

hbase.master.balancer.stochastic.regionCountCost

This configuration parameter defines the value of the multiplier that defines the impact this function makes on the overall target cost of the balancer. The default value of this configuration parameter is 500.

Primary region skewness cost

This cost function calculates the cost of the potential cluster state based on the skewness in the number of primary regions on a cluster. A lower cost returned by this function indicates that the primary regions are evenly distributed across the cluster while a larger value indicates that the regions are unevenly distributed across the cluster. This cost function uses the hbase.master.balancer.stochastic.primaryRegionCountCost configuration to calculate the cost.

hbase.master.balancer.stochastic.primaryRegionCountCost

This configuration defines the value of the multiplier that determines the impact this function makes on the overall target cost of the balancer. The default value of this configuration is 500.

Table skewness cost

This function calculates the cost of a potential cluster configuration based on how evenly distributed tables are across the cluster.

The hbase.master.balancer.stochastic.tableSkewCost configuration parameter defines how the function calculates the target table skewness cost in the cluster and also specifies the value of the multiplier that determines the impact this function makes on the overall target cost of the balancer. The default value of this configuration is 35.

Move cost

Given the starting state of the regions and a potential ending state, this function computes the cost based on the number of regions that have moved.

The following configuration parameters control the way this cost function calculates the cost.

- hbase.master.balancer.stochastic.moveCost

  This is the multiplier for the MoveCostFunction in stochastic load balancer which defines the weightage given to this function while calculating the total cost of the balancer. The default value of this parameter is 7.
- hbase.master.balancer.stochastic.moveCost.offpeak

  This is the value of the multiplier for this cost function during off-peak hours. The default value of this parameter is 3.
- hbase.master.balancer.stochastic.maxMovePercent

  This parameter restricts the number of regions that are moved as part of the balancer run. The default value of this configuration is 1.0.

Locality based cost

The locality-based cost functions compute the cost of a potential cluster configuration based on where the HBase store files are located. The stochastic load balancer uses two types of locality functions.

- **Server Locality Cost**

  This function computes the cost of a potential cluster state based on the location of store files respective to the servers. The more the number of stored files located on the server, the lower the cost and vice-versa.

  The hbase.master.balancer.stochastic.localityCost configuration parameter defines the multiplier value for the server locality cost function with the default value of 25.
- **Rack Locality Cost**

  This function is an extension of the locality cost and computes the cost of the potential cluster configuration based on how the store files are located to the configured disk racks.

  The hbase.master.balancer.stochastic.rackLocalityCost configuration parameter defines the multiplier for the rack locality cost function and has the default value of 15.

Read request cost

This function computes the cost of the total number of read requests. A higher computed cost indicates an unbalanced cluster. This uses a rolling average of region load.

The hbase.master.balancer.stochastic.readRequestCost configuration parameter defines the multiplier for the read request cost function with the default value of 5.

Write request cost

Computes the cost of the total number of write requests. A higher computed cost indicates an unbalanced cluster.

The hbase.master.balancer.stochastic.writeRequestCost configuration parameter defines the multiplier for the write request cost function and has the default value of 5.

Memstore size cost

Computes the cost of total memory store size across the cluster. A higher cost indicates an unbalanced cluster. This uses the rolling average of the statistics received from the region servers.

The hbase.master.balancer.stochastic.memstoreSizeCost configuration parameter defines the multiplier for the memstore size-based cost function. The default value of this parameter is 5.

Storefile cost

This cost function computes the cost of total open-store file sizes. A higher computed cost indicates an unbalanced cluster.

The hbase.master.balancer.stochastic.storefileSizeCost configuration parameter defines the multiplier for the store file size-based cost function. The default value of this parameter is 5.

Configuration details

Know the necessary configuration properties required to use the stochastic load balancer in the HBase cluster.

Configuration steps

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters HBase service Configuration .
3. Search for the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property and add the following configuration properties.

hbase.master.balancer.maxRitPercent

This parameter defines the maximum number of regions that can be in transition at any point in time during balancing. The balancer uses this value to calculate the region plan such that the number of regions in transition is within this limit. For example, a value of 0.01 means that up to 1% of the total number of regions can be in transition during the balancer run.

hbase.master.balancer.stochastic.maxRunningTime

Defines the maximum amount of time for which the balancer runs in a single invocation. The default value for this parameter is 30 seconds.

hbase.master.balancer.stochastic.minCostNeedBalance

The balancer tries to balance the cluster if the overall cost of the balancer is more than the minCostNeedBalance parameter. This parameter ensures that the balancer does not move the regions even if the cluster is slightly off-balanced and hence avoids any unwanted balancer runs. The default value of this balancer is 0.025. The total cost of the balancer is calculated by considering the individual costs returned by the individual cost functions in the balancer.

hbase.master.balancer.stochastic.numRegionLoadsToRemember

The master node periodically receives the statistics from each online region server. This statistic is used by the stochastic load balancer to calculate different costs needed by the balancer to calculate the total cost of the balancer. In some cases, the balancer needs to calculate the rolling averages of the statistics received from the region servers. This parameter defines the number of statistics received from a region server which is then used to calculate the rolling averages. The default value of this parameter is 15 which means the master will maintain up to 15 sets of statistics from a region server.

hbase.master.balancer.stochastic.runMaxSteps

This is a boolean parameter with the default value as false. This parameter defines the number of steps taken by the balancer to calculate the cost of the balancer. If `runMaxSteps` is true, then the balancer considers the maximum number of steps calculated by the balancer. The maximum number of steps to be taken by the parameter is calculated as the maximum value of `maxSteps` defined by the configuration hbase.master.balancer.stochastic.maxSteps.

`maxSteps` = `max`(`valueof`(hbase.master.balancer.stochastic.maxSteps), number of regions * `valueof`(hbase.master.balancer.stochastic.stepsPerRegion) * number of servers)

If this parameter is set to false, then the maximum number of steps is calculated as:

`min`(`valueof`(hbase.master.balancer.stochastic.maxSteps), number of regions * `valueof`(hbase.master.balancer.stochastic.stepsPerRegion) * number of servers)

hbase.master.balancer.stochastic.maxSteps

This is the maximum number of steps to be taken by the balancer to calculate the region plan. This works along with the hbase.master.balancer.stochastic.runMaxSteps configuration parameter.

hbase.master.balancer.stochastic.stepsPerRegion

This parameter defines the maximum number of steps calculated for each region as part of the balancer run and thus controls the size of the region plan generated by the balancer. This value is used along with the hbase.master.balancer.stochastic.runMaxSteps configuration parameter.

Optimizing HBase I/O

You can optimize HBase I/O using several ways. Two HBase key concepts that helps you in the process are BlockCache and MemStore tuning.

The information in this section is oriented toward basic BlockCache and MemStore tuning. As such, it describes only a subset of cache configuration options. Cloudera supports additional BlockCache and MemStore properties, as well as other configurable performance optimizations such as remote procedure calls (RPCs), HFile block size settings, and HFile compaction. For a complete list of configurable properties, see the hbase-default.xml source file in GitHub.

HBase I/O components

The concepts related to HBase file operations and memory (RAM) caching are HFile, Block, BlockCache, MemStore and Write Ahead Log (WAL).

HBase Component	Description
HFile	An HFile contains table data, indexes over that data, and metadata about the data.
Block	An HBase block is the smallest unit of data that can be read from an HFile. Each HFile consists of a series of blocks. (Note: an HBase block is different from an HDFS block or other underlying file system blocks.)
BlockCache	BlockCache is the main HBase mechanism for low-latency random read operations. BlockCache is one of two memory cache structures maintained by HBase. When a block is read from HDFS, it is cached in BlockCache. Frequent access to rows in a block cause the block to be kept in cache, improving read performance.
MemStore	MemStore ("memory store") is in-memory storage for a RegionServer. MemStore is the second of two cache structures maintained by HBase. MemStore improves write performance. It accumulates data until it is full, and then writes ("flushes") the data to a new HFile on disk. MemStore serves two purposes: it increases the total amount of data written to disk in a single operation, and it retains recently written data in memory for subsequent low-latency reads.
Write Ahead Log (WAL)	The WAL is a log file that records all changes to data until the data is successfully written to disk (MemStore is flushed). This protects against data loss in the event of a failure before MemStore contents are written to disk.

HBase Read/Write Operations

BlockCache and MemStore reside in random-access memory (RAM). HFiles and the Write Ahead Log are persisted to HDFS.

The following figure shows these simplified write and read paths:

- During write operations, HBase writes to WAL and MemStore. Data is flushed from MemStore to disk according to size limits and flush interval.
- During read operations, HBase reads the block from BlockCache or MemStore if it is available in those caches. Otherwise, it reads from disk and stores a copy in BlockCache.

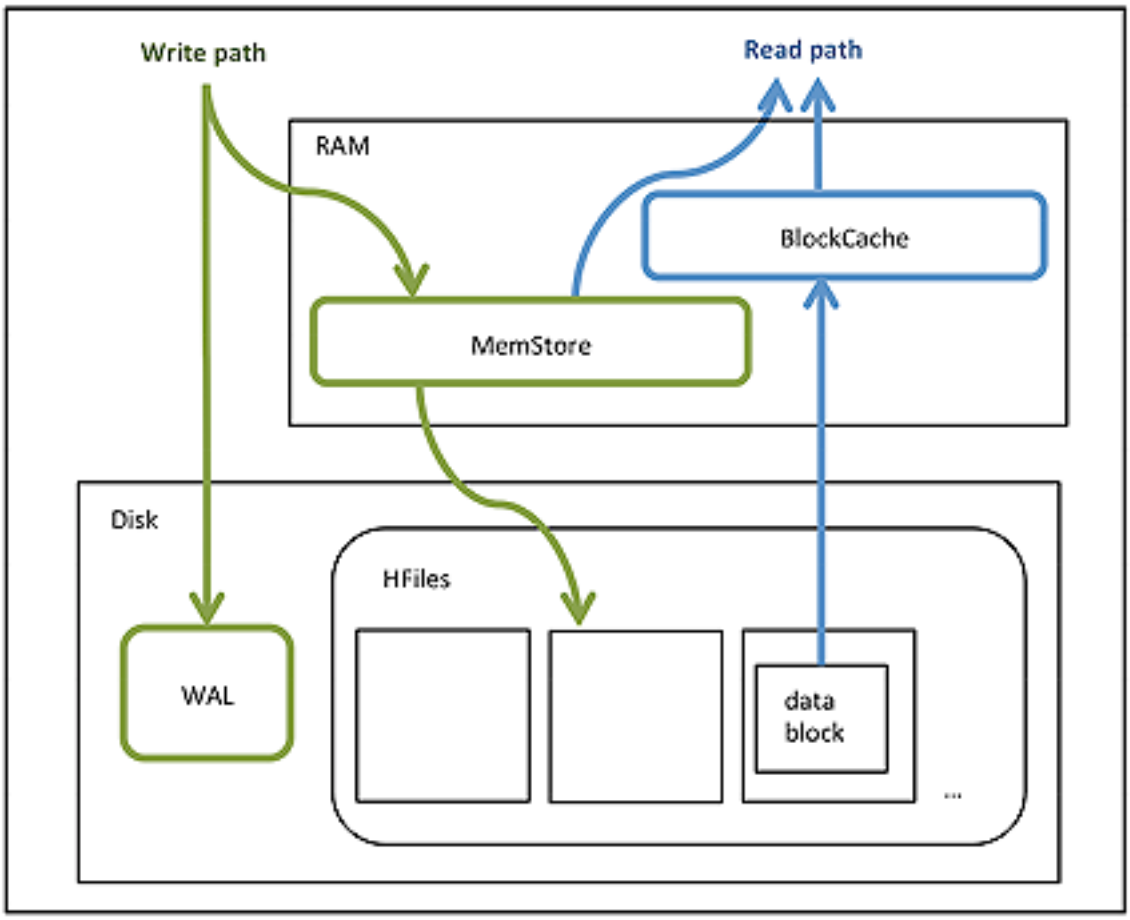

By default, BlockCache resides in an area of RAM that is managed by the Java Virtual Machine (JVM) garbage collector; this area of memory is known as on-heap memory or the JVM heap. The BlockCache implementation that manages the on-heap cache is called LruBlockCache.

If you have stringent read latency requirements and you have more than 20 GB of RAM available on your servers for use by HBase RegionServers, consider configuring BlockCache to use both on-heap and off-heap memory. BucketCache is the off-heap memory equivalent to LruBlockCache in on-heap memory. Read latencies for BucketCache tend to be less erratic than LruBlockCache for large cache loads because BucketCache (not JVM garbage collection) manages block cache allocation. The MemStore always resides in the on-heap memory.

Figure 3: Relationship among Different BlockCache Implementations and MemStore

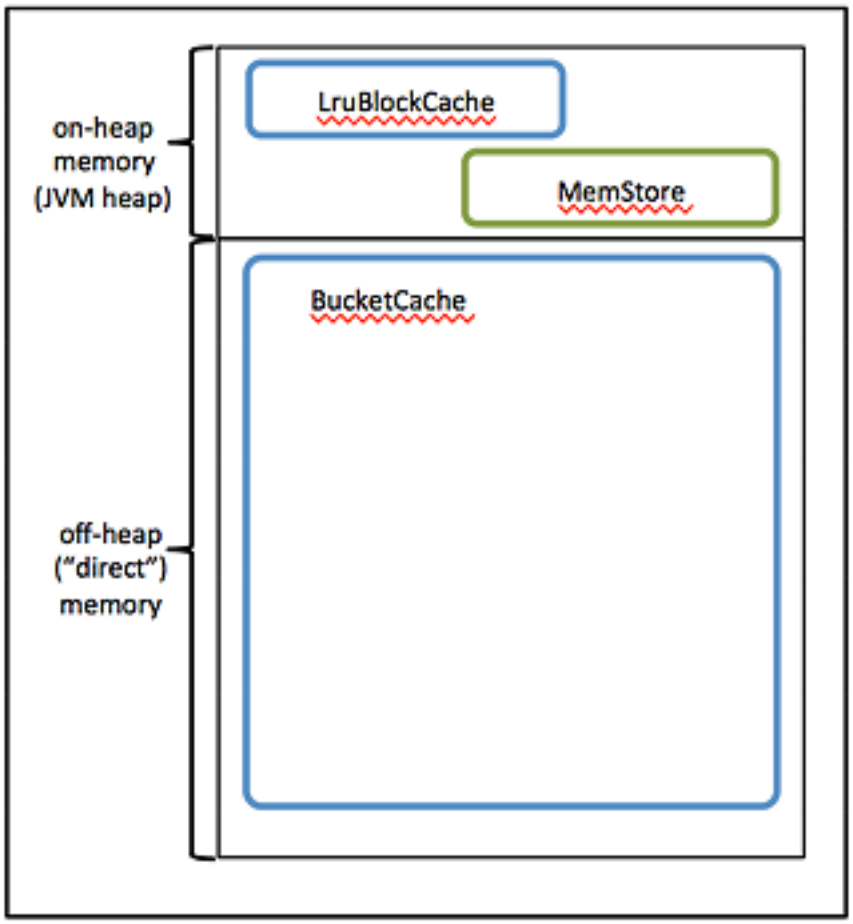

- Additional notes:
- BlockCache is enabled by default for all HBase tables.
- BlockCache is beneficial for both random and sequential read operations although it is of primary consideration for random reads.
- All regions hosted by a RegionServer share the same BlockCache.
- You can turn BlockCache caching on or off per column family.

Advanced configuration for write-heavy workloads

HBase includes several advanced configuration parameters for adjusting the number of threads available to service flushes and compactions in the presence of write-heavy workloads. Tuning these parameters incorrectly can severely degrade performance and is not necessary for most HBase clusters. If you use Cloudera Manager, configure these options using the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml

hbase.hstore.flusher.count

The number of threads available to flush writes from memory to disk. Never increase hbase.hstore .flusher.count to more of 50% of the number of disks available to HBase. For example, if you have 8 solid-state drives (SSDs), hbase.hstore.flusher.count should never exceed 4. This allows scanners and compactions to proceed even in the presence of very high writes.

hbase.regionserver.thread.compaction.large and hbase.regionserver.thread.compaction.small

The number of threads available to handle small and large compactions, respectively. Never increase either of these options to more than 50% of the number of disks available to HBase.

Ideally, hbase.regionserver.thread.compaction.small should be greater than or equal to hbase.region server.thread.compaction.large, since the large compaction threads do more intense work and will be in use longer for a given operation.

In addition to the above, if you use compression on some column families, more CPU will be used when flushing these column families to disk during flushes or compaction. The impact on CPU usage depends on the size of the flush or the amount of data to be decompressed and compressed during compactions.

Enabling HBase META Replicas

When many HBase clients try to access a META table concurrently, it becomes overloaded, which in turn reduces the HBase performance. When you enable HBase META replicas, HBase maintains read-only copies of the META table on additional region servers and ensures that the META table reads are load balanced between the replicas. This improves the HBase performance.

About this task

You can enable HBase META replicas by configuring a set of properties in Cloudera Manager.

Procedure

1. Open Cloudera Manager and go to the HBase service.
2. Click the Configuration tab.
3. Select Scope (Service-Wide) .
4. Locate the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the search box.
5. Add the following property values:
   - Master Replica Count: hbase.meta.replica.count

     Description: The number of region replicas to maintain for the META table.

     Value: 3
   - Region Locator Mode: hbase.locator.meta.replicas.mode

     Description: The Region Locator's mode when META replica is configured.

     Value: LoadBalance
   - Meta Replicas Use: hbase.meta.replicas.use

     Description: Specifies whether to use META table replicas.

     Value: true
   - Region Replica Replication Catalog Enabled: hbase.region.replica.replication.catalog.enabled

     Description: Specifies whether to enable asynchronous WAL replication for META region replicas.

     Value: true
6. Select Scope Gateway .
7. Locate the HBase Client Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the search box.
8. Add the property values mentioned in Step 5.
9. Enter a Reason for change, and then click Save Changes .
10. Restart the role and service when Cloudera Manager prompts you to restart.

Enabling ZooKeeper-less connection registry for HBase client

Operational Database allows you to connect to an HBase server without going through the ZooKeeper thereby removing the dependency and the authentication issues. An HBase client can directly connect to an HBase master and obtain the internally stored META table regions and active master address.

About this task

Previously, when HBase clients connected to HBase servers, they connected through ZooKeeper to obtain the metadata required for client operations. Now, you can enable the ZooKeeper-less HBase client connection using the Client Connection Registry configuration property in Cloudera Manager.

This property supports two values.

- ZooKeeper Quorum: The HBase client connects to the cluster using ZooKeeper. This is the default value.
- Master Registry: The HBase client directly connects to the HBase server.

Procedure

1. Open Cloudera Manager and go to the HBase service.
2. Go to the Configuration tab.
3. Select Scope > (Service-Wide).
4. Locate the Client Connection Registry property or search for it by typing its name in the search box.
5. Set the value to Master Registry.
6. Enter a Reason for change, and then click Save Changes.
7. Restart the role and service when Cloudera Manager prompts you to restart.

Results

The HBase client directly connects to the HBase server without using the ZooKeeper.

Configuring Oozie for Managing Hadoop Jobs

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Overview of Oozie 6

Adding the Oozie service using Cloudera Manager 6

Considerations for Oozie to work with AWS 6

Adding file system credentials to an Oozie workflow 6
- Credentials for token delegation 6
- File System Credentials 7
- Setting file system credentials for Oozie through hadoop properties using Hue Editor 7
- Setting default credentials using Cloudera Manager 12
- Advanced settings: Overriding default configurations 14
- Modifying the workflow file manually 15
- Hue Limitation 15

User authorization configuration for Oozie 16

Redeploying the Oozie ShareLib 17
- Redeploying the Oozie sharelib using Cloudera Manager 18

Oozie configurations with Cloudera services 18
- Using Sqoop actions with Oozie 18
  - Deploying and configuring Oozie Sqoop1 Action JDBC drivers 18
  - Configuring Oozie Sqoop1 Action workflow JDBC drivers 19
- Configuring Oozie to enable MapReduce jobs to read or write from Amazon S3 19
- Configuring Oozie to use HDFS HA 20
- Using Hive Warehouse Connector with Oozie Spark Action 20
  - Appendix - Creating a new ‘hwc’ ShareLib 21
  - Example for using HWC with Oozie Spark action 22
- Oozie and client configurations 28

Spark 3 support in Oozie 29
- Enable Spark actions 29
- Use Spark actions with a custom Python executable 31
- Spark 3 Oozie action schema 33
- Differences between Spark and Spark 3 actions 34
- Use Spark 3 actions with a custom Python executable 35
- Spark 3 compatibility action executor 37
- Spark 3 examples with Python or Java application 38
- Shell action for Spark 3 40
- Migration of Spark 2 applications 40
- Hue support for Oozie 41

Oozie High Availability.....41
Requirements for Oozie High Availability.....41
Configuring Oozie High Availability using Cloudera Manager.....42
Oozie Load Balancer configuration.....42
Enabling Oozie High Availability.....43
Disabling Oozie High Availability.....44

Scheduling in Oozie using cron-like syntax.....44
Oozie scheduling examples.....45

Configuring an external database for Oozie.....47
Configuring PostgreSQL for Oozie.....47
Configuring MariaDB for Oozie.....48
Configuring MySQL 5 for Oozie.....48
Configuring MySQL 8 for Oozie.....49
Configuring Oracle for Oozie.....50

Working with the Oozie server.....50
Starting the Oozie server.....51
Stopping the Oozie server.....51
Accessing the Oozie server with the Oozie Client.....51
Accessing the Oozie server with a browser.....52

Adding schema to Oozie using Cloudera Manager.....53

Enabling the Oozie web console on managed clusters.....54

Enabling Oozie SLA with Cloudera Manager.....55

Disabling Oozie UI using Cloudera Manager.....56

Moving the Oozie service to a different host.....56

Oozie database configurations.....56
Configuring Oozie data purge settings using Cloudera Manager.....57
Loading the Oozie database.....57
Dumping the Oozie database.....58
Setting the Oozie database timezone.....58
Fine-tuning Oozie's database connection.....58
Assembling a secure JDBC URL for Oozie.....60
Oracle TCPS.....60
OpenJPA upgrade.....61

Prerequisites for configuring TLS/SSL for Oozie.....64

Configure TLS/SSL for Oozie.....64

Oozie Java-based actions with Java 17.....65

Oozie security enhancements.....66

Additional considerations when configuring TLS/SSL for Oozie HA.....68

Configure Oozie client when TLS/SSL is enabled..... 68

Configuring custom Kerberos principal for Oozie..... 68

Overview of Oozie

Apache Oozie Workflow Scheduler for Hadoop is a workflow and coordination service for managing Apache Hadoop jobs:

- Oozie Workflow jobs are Directed Acyclic Graphs (DAGs) of actions; actions are Hadoop jobs (such as MapReduce, Streaming, Hive, Sqoop and so on) or non-Hadoop actions such as Java, shell, Git, and SSH.
- Oozie Coordinator jobs trigger recurrent Workflow jobs based on time (frequency) and data availability.
- Oozie Bundle jobs are sets of Coordinator jobs managed as a single job.

Oozie is an extensible, scalable and data-aware service that you can use to orchestrate dependencies among jobs running on Hadoop.

Related Information

Apache Oozie Workflow Scheduler for Hadoop

Adding the Oozie service using Cloudera Manager

The Oozie service can be automatically installed and started during your installation of Cloudera with Cloudera Manager. If required, you can install Oozie manually with the Add Service wizard in Cloudera Manager. The wizard configures and starts Oozie and its dependent services.

Note: If your instance of Cloudera Manager uses an external database, you must also configure Oozie with an external database.

Considerations for Oozie to work with AWS

If you want to access Amazon Web Services (AWS) S3 data through Oozie, then ensure that the required AWS credentials are configured in core-site.xml.

Adding file system credentials to an Oozie workflow

Oozie has to use its Kerberos credentials to obtain delegation tokens on behalf of the user from the services. You must add additional configurations in the Oozie workflow for it to obtain this delegation token.

Credentials for token delegation

A secure cluster requires the Oozie actions to be authenticated; typically using Kerberos. However, due to the way that Oozie runs actions, Kerberos credentials are not available to them. Some actions require communication to an external service like HCatalog, HBase, and Hive or a secure file system like Amazon S3. For these situations, Oozie has to use its Kerberos credentials to obtain delegation tokens which allows Oozie to access the external service or file system.

For information about Action Authentication, see Apache Oozie documentation.

File System Credentials

To allow Oozie to access S3, ABFS, and other filesystems, it has to request a delegation token to obtain credentials. Although HUE supports only the built-in credentials (see Hue limitation) this can be done through properties. There are three options to set file system credentials for an Oozie workflow or action which are resulting the same. You can choose any of them based on your current usage and preference.

1. Set file system credentials for Oozie through hadoop properties using Hue Editor.
2. Set default credentials using Cloudera Manager.
3. Modify the workflow XML file manually.

You can use the advanced settings to override the default configurations.

Setting file system credentials for Oozie through hadoop properties using Hue Editor

This option allows you to configure additional Oozie credentials with configuration properties. The following example shows how to set up an Oozie workflow with a shell action which uses additional file system credentials.

About this task

Note: Hue Limitation - Hue is not capable of adding user defined file system credentials to the Oozie workflow, only built-in credentials can be added. Hence, you must manually define file system credentials for Oozie workflow.

Procedure

1. In Hue web UI, click Scheduler > Workflow to create a new workflow.
2. Click Documents and select Actions in the drop-down list.

3. Select the Shell icon to the workflow to add a shell action. The shell command can be echo and its parameter can be Hello world for this example.

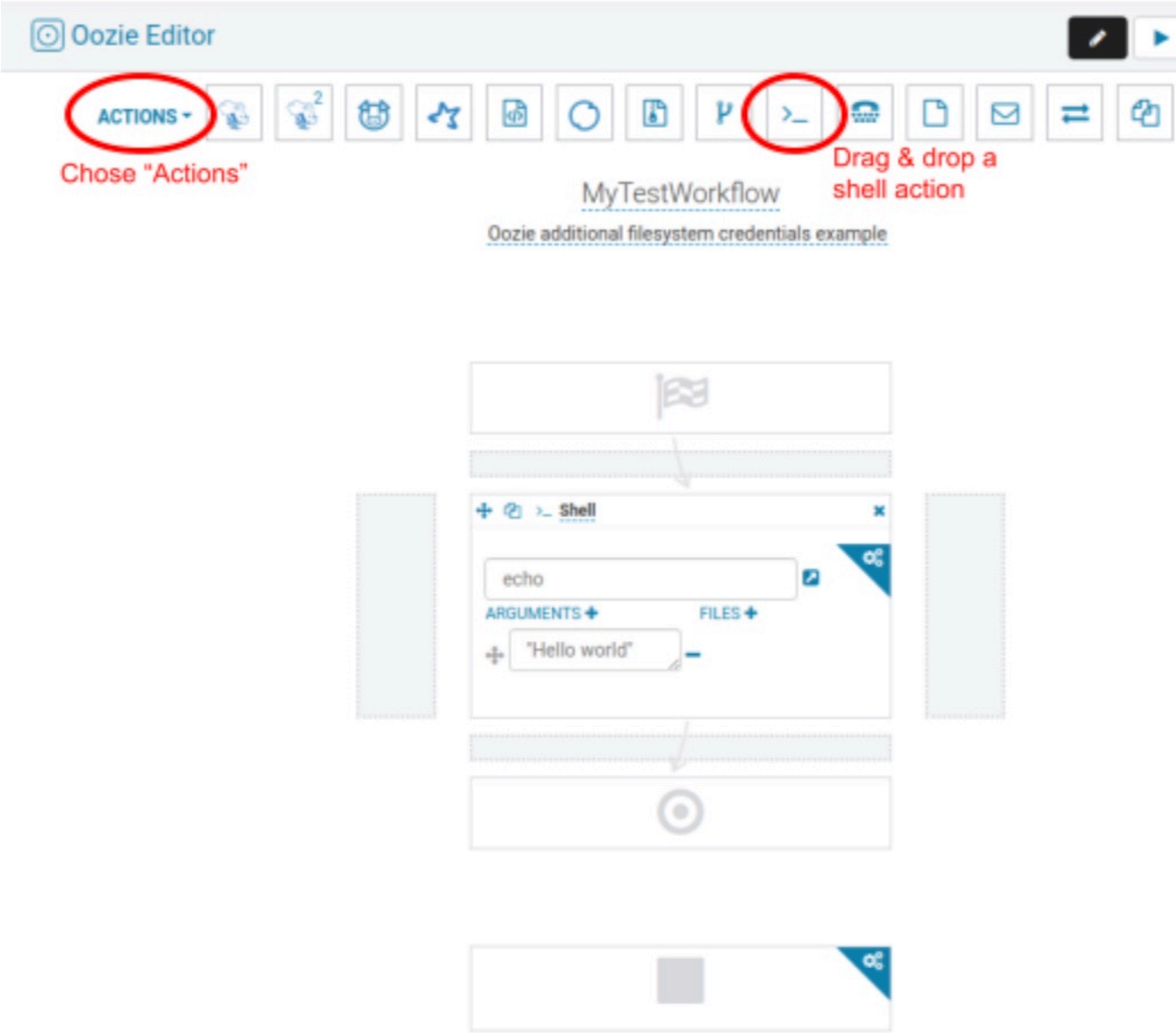

4. Select the settings under the Edit menu to add properties to the workflow. The Workflow Settings dialog box is displayed.

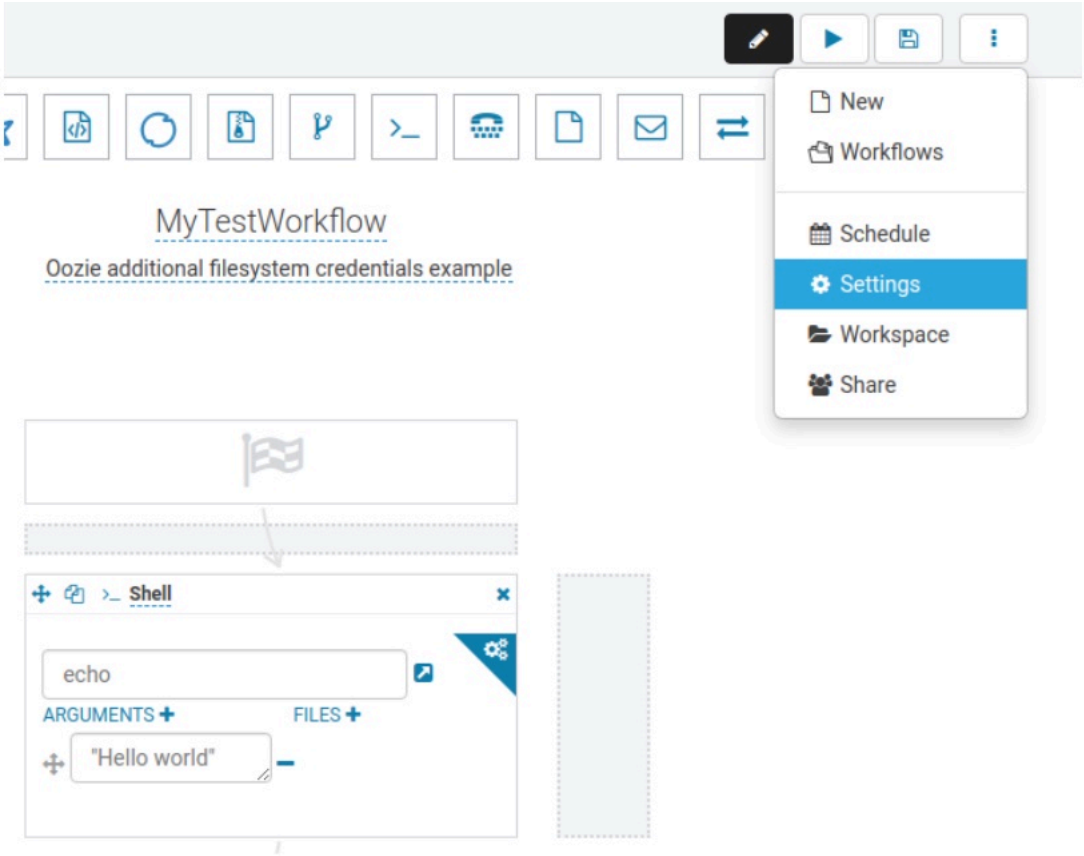

5. In the **Workflow Settings** dialog box, under the **Hadoop Properties** section, click + Add property and add the necessary credentials. The property name must be in the format similar to *OOZIE.ACTION.CREDENTIALS.FILESYSTEM.<MYCUSTOMCREDENTIAL>* where *MYCUSTOMCREDENTIAL* is the description of the credential. The value of this property must be set to a valid

filesystem URI, for example, an S3 bucket URL. Click + Add property to add additional properties if you want to access more than one s3 bucket, and so on, as shown in the below figure.

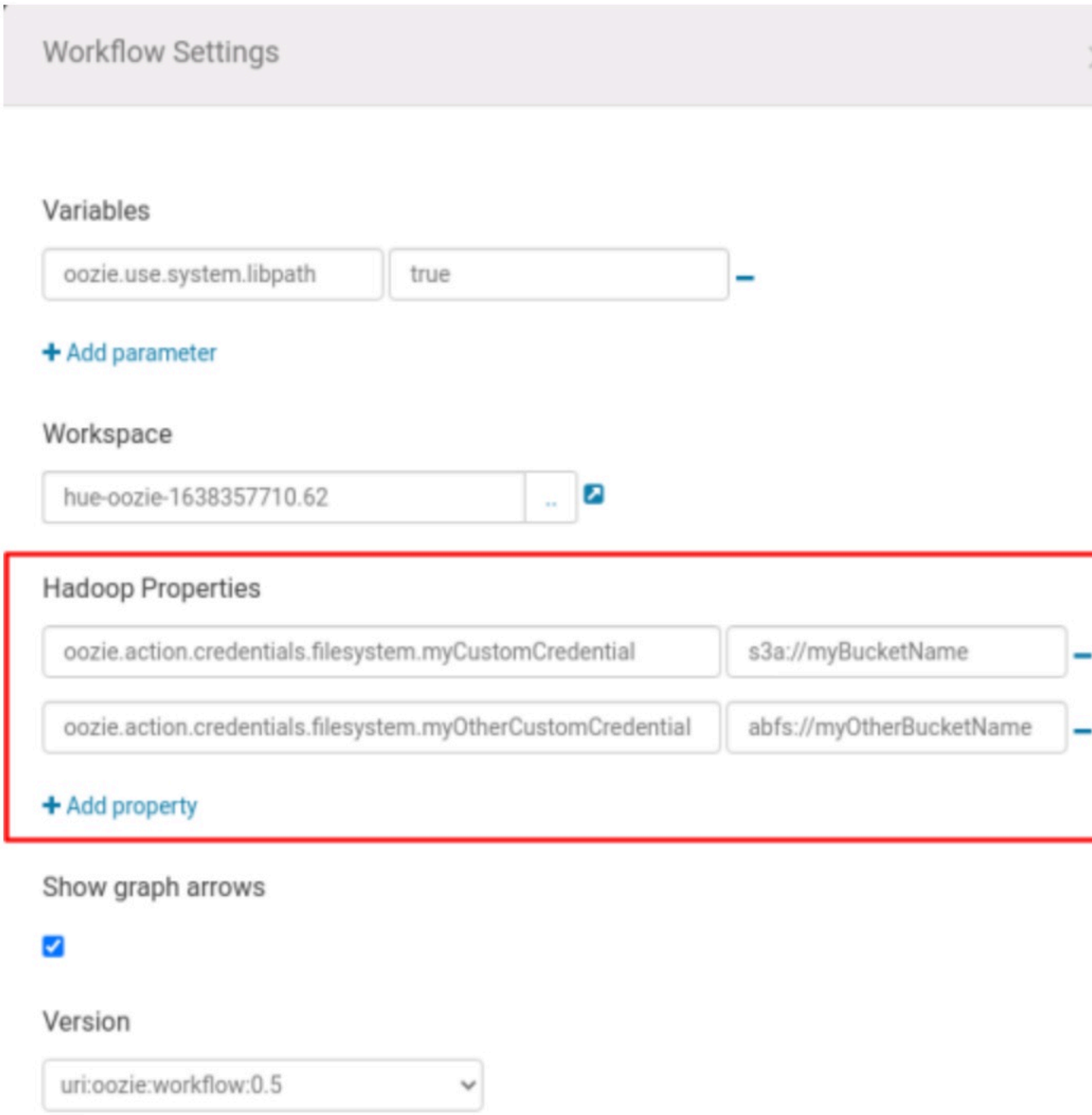

6. Click Action Settings to set the credential defined in the previous step to use it in the workflow action where it is needed.

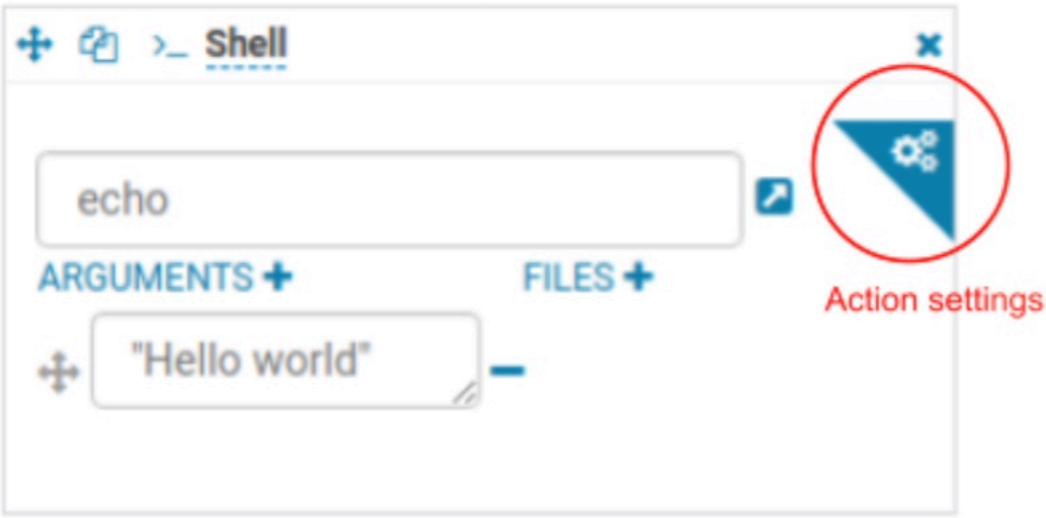

7. Add a new property to the action with the name oozie.action.credentials.filesystem and its value must be a comma-separated list of credential names defined in the hadoop properties. For example “myCustomCredential,myOtherCustomCredential”. Based on this property, Oozie requests delegation tokens for the file-systems defined in the given credentials.

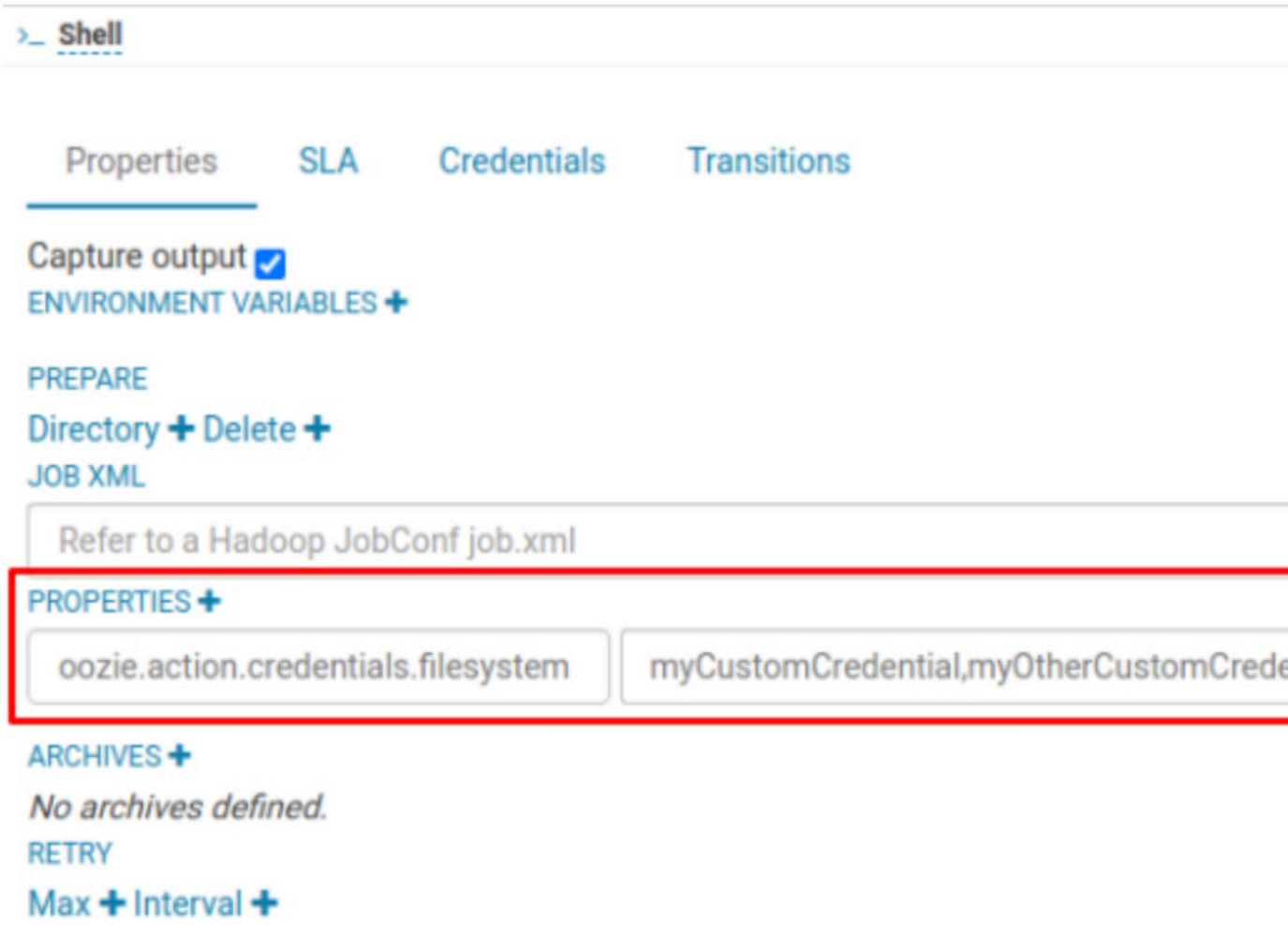

Results

Now the workflow is ready to be submitted and the following is the generated workflow.xml

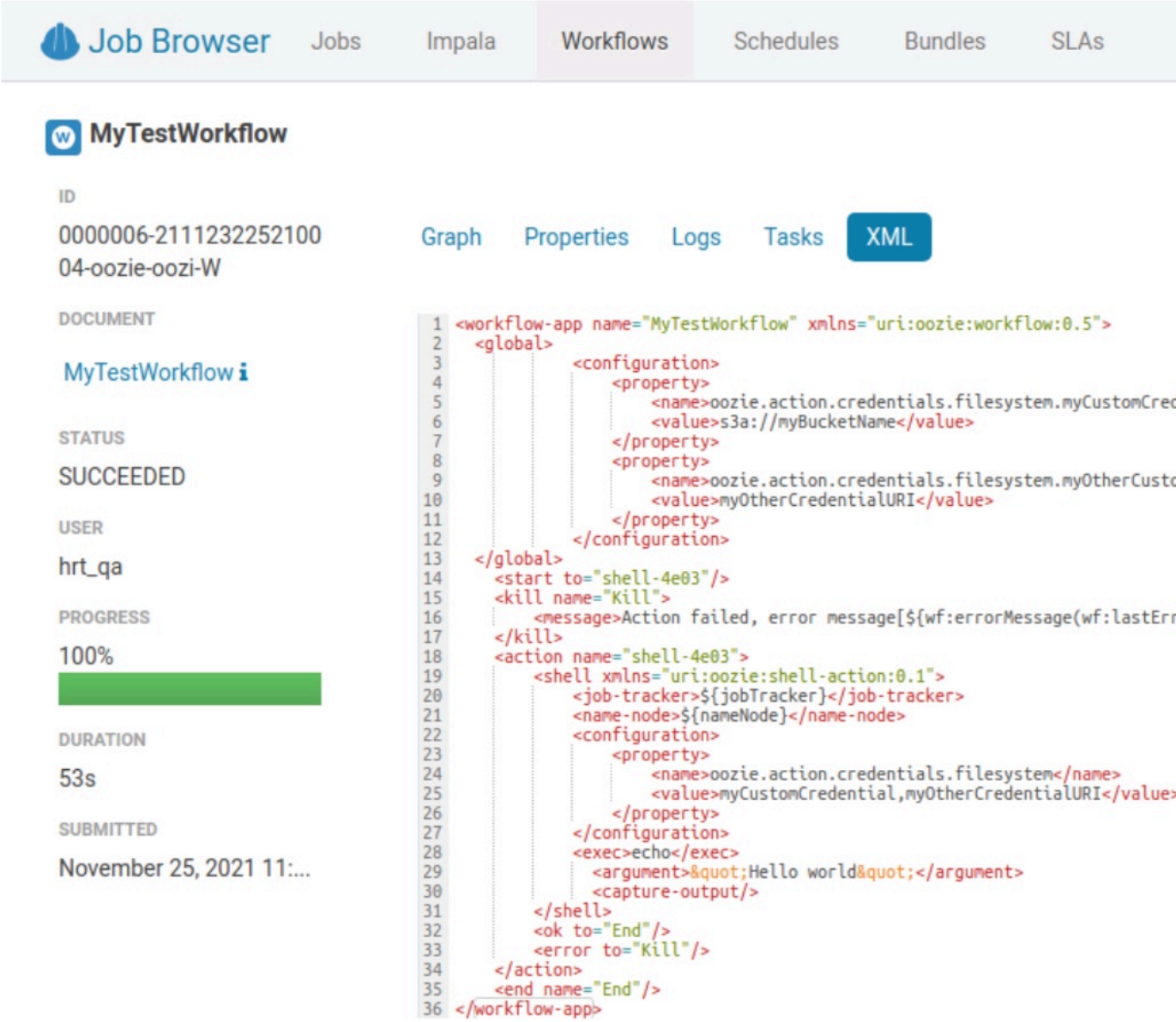

file.

Setting default credentials using Cloudera Manager

In scenarios where you need to set a credential for every action of every workflow, you can set the credential definitions and its usages using Cloudera Manager. You can also set these as the default configurations.

Procedure

1. In Cloudera Manager, click the Oozie service.
2. Click the Configuration tab.
3. Search for action_conf and add the following in the Oozie Server Advanced Configuration Snippet (Safety Valve) for action-conf/default.xml field:

```
Definition (can be seen by all the actions started by Oozie)
```

```
Name: oozie.action.credentials.filesystem.myCustomCredential
Value: s3a://<bucket-name>

Name: oozie.action.credentials.filesystem.myotherCustomCredential
Value: abfs://<bucket-name>
```

```
Usage (applies to all the actions started by Oozie)

Name: oozie.action.credentials.filesystem
Value: myCustomCredential,myotherCustomCredential
```

4. Click Save Changes on the bottom right corner.

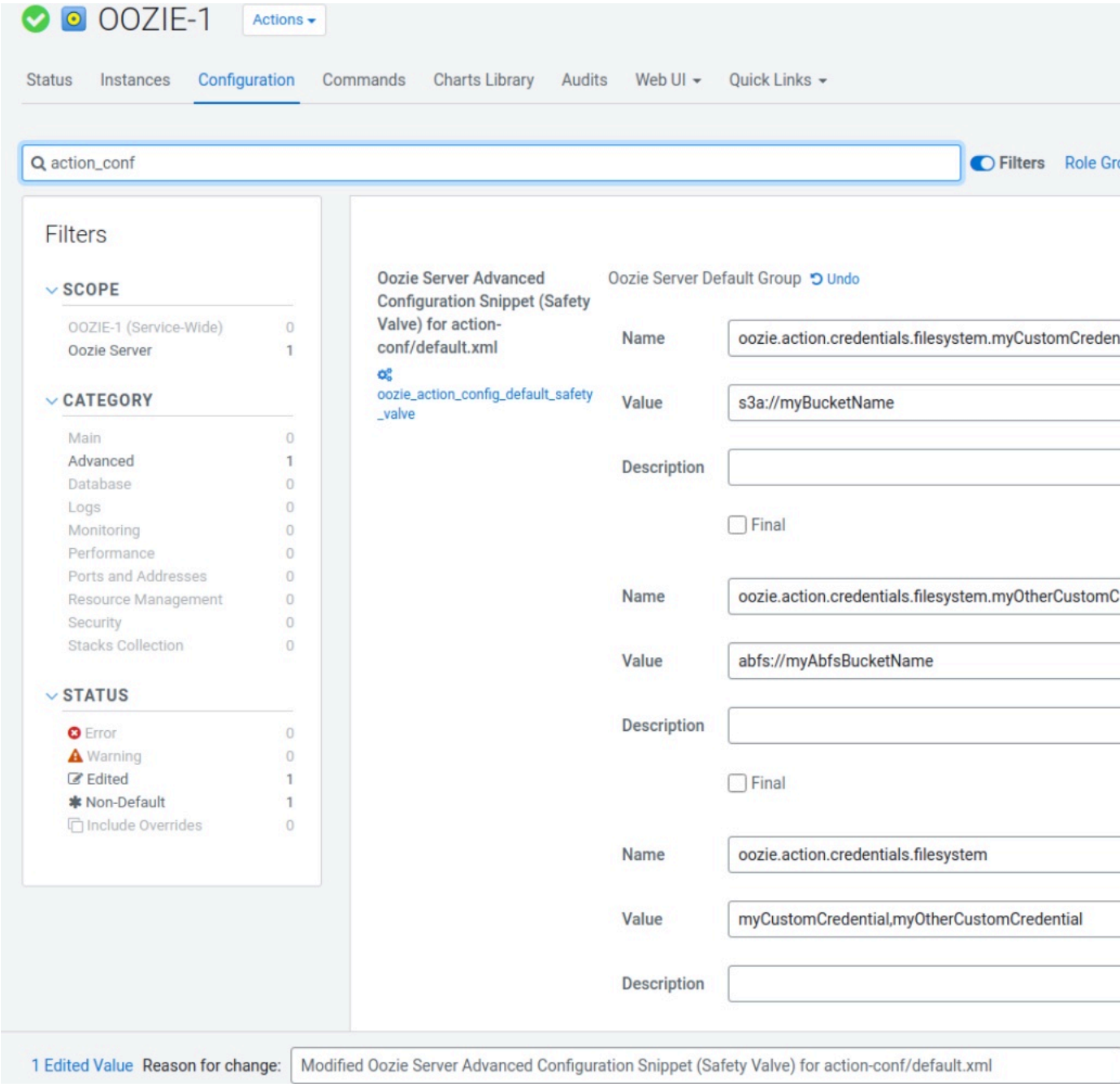

5.

Click Stale Service Restart that is next to the Oozie service name.

6. Review the properties added to the default action configuration. All these properties will be available in every Oozie action.

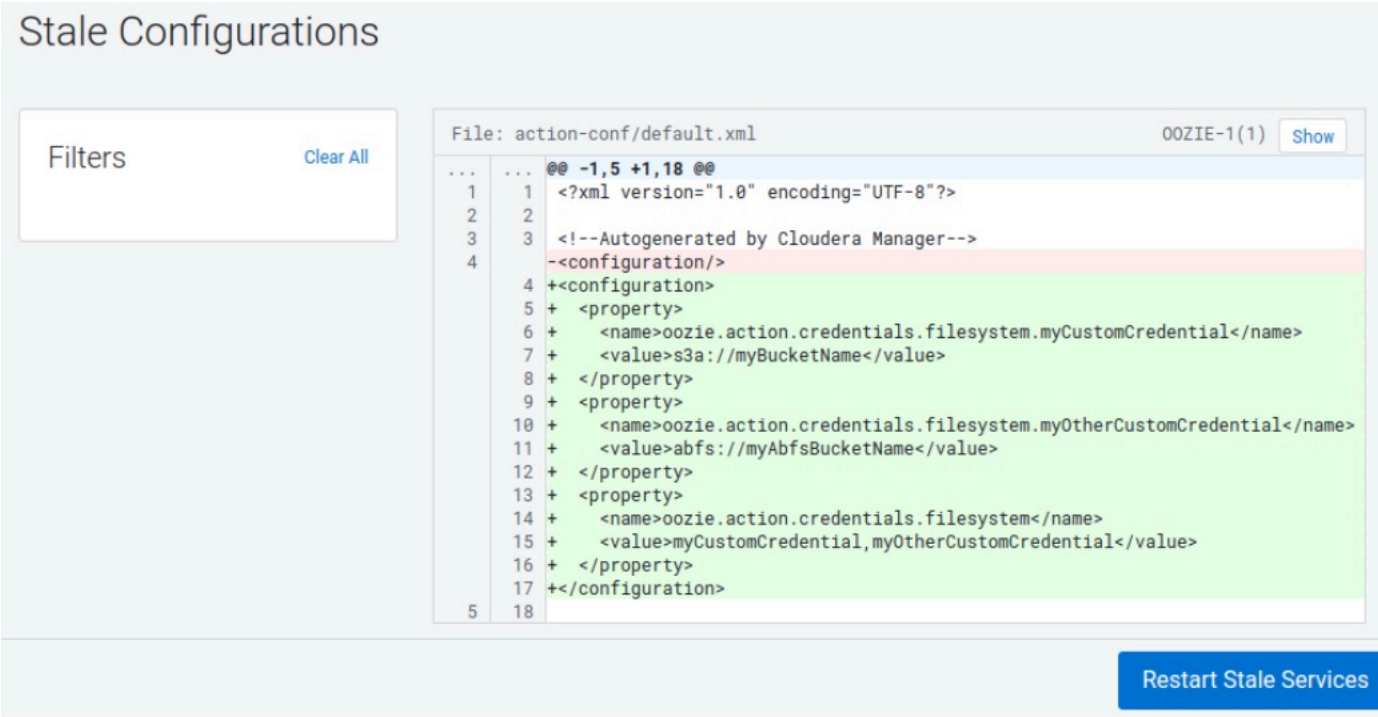

7. Click Restart Stale Services to make this change happen on the Oozie instances.

Note: After this configuration, you need not add anything in HUE, Oozie obtains a delegation token for those file-system paths for every action.

Advanced settings: Overriding default configurations

You can override the default value set using Cloudera Manager, if required. For example, you have five different actions in the workflow and four of them use S3 buckets and one use ABFS. In this case, the S3 can be set as a default credential and can be overridden in only that one action to have a different value.

Procedure

1. Using Cloudera Manager, set the credentials to use s3 bucket in the Oozie Server Advanced Configuration Snippet (Safety Valve) for action-conf/default.xml field:

```
Name: oozie.action.credentials.filesystem.myCustomCredential
Value: s3a://<bucket-name>
```

2. Using the HUE editor, edit the action for which you want to use a different credential and add the following:

```
Name: oozie.action.credentials.filesystem.myCustomCredential
Value: abfs://myAbfsBucketName
```

Add the following property as described in the Step 4 and 5 of the Option 1: Setting file system credentials for Oozie through hadoop properties using Hue Editor .

Now Oozie actions will use the default S3 credential except the one which has the non-default value. It is also possible to combine the global configuration, where you define the possible file-system credentials, and in HUE you use the pre-defined global credentials.

Modifying the workflow file manually

You can modify or add the `"<credentials>...</credentials>"` block at the beginning of the Oozie workflow.xml file. Do not remove any existing credentials from this block, add the new file system credential. Refer the newly defined credential in the action where it needs to be used.

Example

```
<workflow-app name="MyTestWorkflow" xmlns="uri:oozie:workflow:0.5">
    <credentials>
        <credential name="my-s3-creds" type="filesystem">
           <property>
               <name>filesystem.path</name>
               <value>s3a://{yourBucketName}</value>
           </property>
        </credential>
    </credentials>
    <start to="shell-action"/>
    <kill name="Kill">
        <message>Action failed</message>
    </kill>
    <action name="shell-action" cred="my-s3-creds">
        <shell xmlns="uri:oozie:shell-action:0.1">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <exec>echo</exec>
              <argument>&quot;Hello world&quot;</argument>
              <capture-output/>
        </shell>
        <ok to="End"/>
        <error to="Kill"/>
    </action>
    <end name="End"/>
</workflow-app>
```

Hue Limitation

Hue is not capable of adding user defined file system credentials to the Oozie workflow but can add only built-in credentials as seen in the below figure.

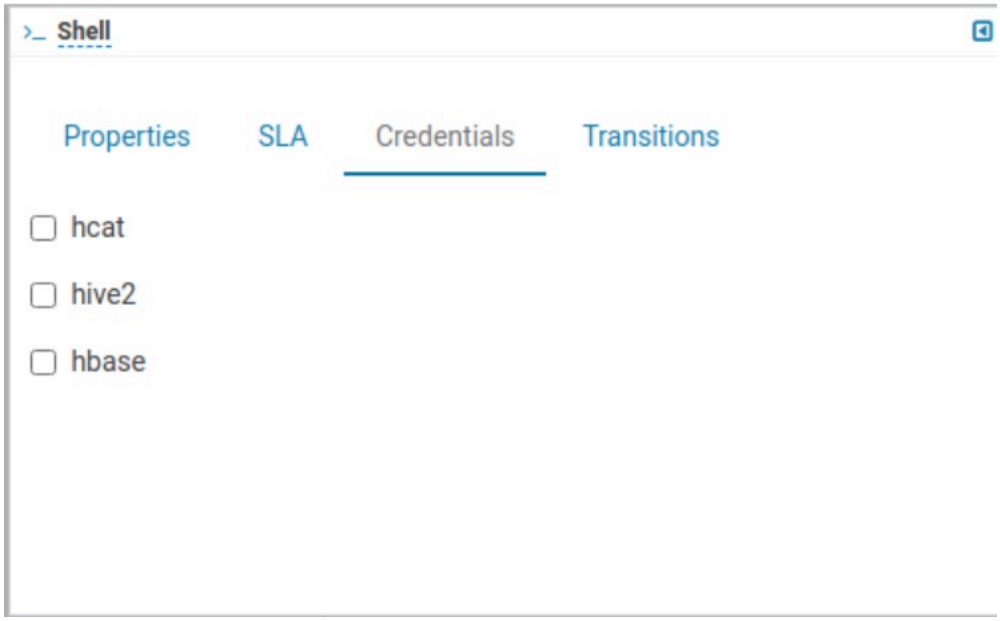

User authorization configuration for Oozie

Learn about user authorization model for Oozie and Access Control List (ACL). Also, learn about how to define admin users for Oozie jobs and ACL.

Oozie has a basic authorization model which is as follows:

- Users have read access to all jobs
- Users have write access to their own jobs
- Users have write access to jobs based on an ACL, which is a list of users and groups
- Users have read access to admin operations
- Admin users have write access to all jobs
- Admin users have write access to admin operations

If security is disabled all users are admin users.

Oozie security is set through the following configuration property, which is false by default:

```
oozie.service.AuthorizationService.security.enabled=false
```

Note: The old ACL model where a group was provided is still supported if the following property is set in the oozie-site.xml file:

```
oozie.service.AuthorizationService.default.group.as.acl=true
```

Defining admin users

Admin users are determined from the list of admin groups, specified in the oozie.service.AuthorizationService.admin .groups property. Use commas to separate multiple groups. Spaces, tabs, and ENTER characters are trimmed.

If the above property for admin groups is not set, then you can define the admin users in the following manner. The list of admin users can be in the conf/adminusers.txt file. The syntax of this file is as follows:

- One user name per line
- Empty lines and lines starting with # are ignored

Admin users can also be defined in the oozie.serviceAuthorizationService.admin.users property. Use commas to separate multiple admin users. Spaces, tabs, and ENTER characters are trimmed.

In case there are admin users defined using both methods, the effective list of admin users will be the union of the admin users found in the adminusers.txt file and those specified with the oozie.serviceAuthorizationService.admin.users property.

Defining access control lists

ACLs are defined in the following ways:

- workflow job submission over CLI

  Configuration property group.name of job.properties.
- workflow job submission over HTTP

  Configuration property group.name of the XML submitted over HTTP.
- workflow job re-run

  Configuration property oozie.job.acl (preferred) or configuration property group.name of job.properties.
- coordinator job submission over CLI

  Configuration property oozie.job.acl (preferred) or configuration property group.name of job.properties.
- bundle job submission over CLI

  Configuration property oozie.job.acl (preferred) or configuration property group.name of job.properties.

For all other workflow, coordinator, or bundle actions, the ACL set in beforehand are used as basis.

Once the ACL for the job is defined, Oozie checks over HDFS whether the user trying to perform a specific action is part of the necessary group(s). For implementation details, check out org.apache.hadoop.security.Groups#getGroups(String user).

Note that it is enough that the submitting user be part of at least one group of the ACL. Note also that the ACL can contain user names as well. If there is an ACL defined and the submitting user is not part of any group or user name present in the ACL, an AuthorizationException is thrown.

Example: A typical ACL setup

Detail of job.properties on workflow job submission:

```
user.name=joe
group.name=marketing,admin,qa,root
```

HDFS group membership of HDFS user joe is qa. That is, the check to org.apache.hadoop.security.Groups#getGroups("joe") returns qa. Hence, ACL check passes inside AuthorizationService, because the user.name provided belongs to at least one of the ACL list elements provided as group.name.

Redeploying the Oozie ShareLib

Some Oozie actions – specifically DistCp, Streaming, Sqoop, and Hive – require external JAR files in order to run. Instead of having to keep these JAR files in each workflow's lib folder, or forcing you to manually manage them using the oozie.libpath property on every workflow using one of these actions, Oozie provides the ShareLib.

The ShareLib behaves very similarly to oozie.libpath, except that it is specific to the aforementioned actions and their required JARs.

Redeploying the Oozie sharelib using Cloudera Manager

When you switch between MapReduce and YARN computation frameworks, you must redeploy the Oozie ShareLib.

About this task

Procedure

1. Go to the Oozie service.
2. Select Actions Install Oozie ShareLib .

Oozie configurations with Cloudera services

You can configure Oozie to work with different Cloudera services.

Some of the different services for which you can configure Oozie are as follows:

- Using Sqoop actions with Oozie
- Enabling MapReduce jobs controlled by Oozie to read from or write to Amazon S3 or Microsoft Azure ADLS
- Configuring Oozie to use HDFS HA

Using Sqoop actions with Oozie

There are certain recommendations that you must consider for using Sqoop actions with Oozie.

Note: Sqoop1 does not ship with third party JDBC drivers. You must download them separately and save them to the /var/lib/sqoop/ directory on the Oozie server.

Recommendations for using Sqoop actions with Oozie

- Cloudera recommends that you not use Sqoop CLI commands with an Oozie Shell Action. Such deployments are not reliable and prone to breaking during upgrades and configuration changes.
- To import data into Hive, use a combination of a Sqoop Action with a Hive2 Action.
  - A Sqoop Action to simply ingest data into HDFS.
  - A Hive2 Action that loads the data from HDFS into Hive.

Deploying and configuring Oozie Sqoop1 Action JDBC drivers

You must deploy and configure the Oozie Sqoop1 action JDBC drivers on HDFS.

Before you begin

Confirm that your Sqoop1 JDBC drivers are present in /var/lib/sqoop.

Procedure

- SSH to the Oozie server host and run the following commands to deploy and configure the drivers on HDFS.

```
cd /var/lib/sqoop
sudo -u hdfs hdfs dfs -mkdir /user/oozie/libext
sudo -u hdfs hdfs dfs -chown oozie:oozie /user/oozie/libext
sudo -u hdfs hdfs dfs -put /opt/cloudera/parcels/SQOOP_NETEZZA_CONNECTOR/s
qoop-nz-connector*.jar /user/oozie/libext/
sudo -u hdfs hdfs dfs -put /opt/cloudera/parcels/SQOOP_TERADATA_CONNECTOR/
lib/*.jar /user/oozie/libext/
```

```
sudo -u hdfs hdfs dfs -put /opt/cloudera/parcels/SQOOP_TERADATA_CONNECTOR/
sqoop-connector-teradata*.jar /user/oozie/libext/
sudo -u hdfs hdfs dfs -put /var/lib/sqoop/*.jar /user/oozie/libext/
sudo -u hdfs hdfs dfs -chown oozie:oozie /user/oozie/libext/*.jar
sudo -u hdfs hdfs dfs -chmod 755 /user/oozie/libext/*.jar
sudo -u hdfs hdfs dfs -ls /user/oozie/libext
# [sample contents of /user/oozie/libext]
-rwxr-xr-x   3 oozie oozie     959987 2016-05-29 09:58 /user/oozie/libext/
mysql-connector-java.jar
-rwxr-xr-x   3 oozie oozie     358437 2016-05-29 09:58 /user/oozie/libext/
nzjdbc3.jar
-rwxr-xr-x   3 oozie oozie    2739670 2016-05-29 09:58 /user/oozie/libext/
ojdbc6.jar
-rwxr-xr-x   3 oozie oozie    3973162 2016-05-29 09:58 /user/oozie/libext/
sqoop-connector-teradata-1.5c5.jar
-rwxr-xr-x   3 oozie oozie      41691 2016-05-29 09:58 /user/oozie/libext/
sqoop-nz-connector-1.3c5.jar
-rwxr-xr-x   3 oozie oozie       2405 2016-05-29 09:58 /user/oozie/libext/
tdgssconfig.jar
-rwxr-xr-x   3 oozie oozie     873860 2016-05-29 09:58 /user/oozie/libext/
terajdbc4.jar
```

Configuring Oozie Sqoop1 Action workflow JDBC drivers

You must confirm that the Sqoop1 JDBC drivers are present in HDFS and then configure the required variables.

Procedure

1. Confirm that the Sqoop1 JDBC drivers are present in HDFS. To do this, SSH to the Oozie Server host and run the following command:

```
sudo -u hdfs hdfs dfs -ls /user/oozie/libext
```

2. Configure the following Oozie Sqoop1 Action workflow variables in Oozie's job.properties file as follows:

```
oozie.use.system.libpath = true
oozie.libpath = /user/oozie/libext
```

Configuring Oozie to enable MapReduce jobs to read or write from Amazon S3

MapReduce jobs controlled by Oozie as part of a workflow can read from and write to Amazon S3.

Before you begin

You will need your AWS credentials (the appropriate Access key ID and Secret access key obtained from Amazon Web Services for your Amazon S3 bucket). After storing these credentials in the keystore (the JCEKS file), specify the path to this keystore in the Oozie workflow configuration.

About this task

This setup is for use in the context of Oozie workflows only, and does not support running shell scripts on Amazon S3 or other types of scenarios.

Note: In the following steps, replace the *PATH/TO/FILE* with the HDFS directory where the .jceks file is located, and replace *ACCESS_KEY_ID* and *SECRET_ACCESS_KEY* with your AWS credentials.

Procedure

1. Create the credential store (.jceks) and add your AWS access key to it as follows:

```
hadoop credential create fs.s3a.access.key -provider \
jceks://hdfs/PATH/TO/FILE.jceks -value ACCESS_KEY_ID
```

For example:

```
hadoop credential create fs.s3a.access.key -provider \
jceks://hdfs/user/root/awskeyfile.jceks -value AKIAIPVYH....
```

2. Add the AWS secret to this same keystore.

```
hadoop credential create fs.s3a.secret.key -provider \
jceks://hdfs/PATH/TO/FILE.jceks -value SECRET_ACCESS_KEY
```

3. Set hadoop.security.credential.provider.path to the path of the .jceks file in Oozie's workflow.xml file in the MapReduce Action's <configuration> section so that the MapReduce framework can load the AWS credentials that give access to Amazon S3.

```
<action name="S3job">
    <map-reduce>
        <job-tracker>${jobtracker}</job-tracker>
        <name-node>${namenode}</name-node>
        <configuration>
            <property>
                <name>hadoop.security.credential.provider.path</name>
                <value>jceks://hdfs/PATH/TO/FILE.jceks</value>
            </property>
            ....
            ....
</action>
```

Configuring Oozie to use HDFS HA

To configure an Oozie workflow to use HDFS HA, use the HDFS nameservice instead of the NameNode URI in the <name-node> element of the workflow.

Example:

```
<action name="mr-node">
  <map-reduce>
    <job-tracker>${jobTracker}</job-tracker>
    <name-node>hdfs://ha-nn</name-node>
```

where *HA-NN* is the value of dfs.nameservices in hdfs-site.xml.

Related Information

Additional considerations when configuring TLS/SSL for Oozie HA

Using Hive Warehouse Connector with Oozie Spark Action

You can use Hive Warehouse Connector (HWC) with Oozie Spark action by updating job.properties file or action-level configurations.

Note: There are known issues related to using Hive Warehouse Connector with Oozie Spark Action. Read the known issues and use the workaround listed in the *Cloudera Runtime Release Notes*.

For Updating job properties file

Steps

1. Create a new ShareLib using a different name, such as hwc.
2. Place the HWC JAR onto the new ShareLib. For information about placing HWC JARs in the new ShareLib, see the **Appendix - Creating a new 'hwc' ShareLib** section below.
3. Execute a ShareLib update.
4. When executing a Spark action using the HWC include the following properties in the job.properties file:

```
oozie.action.sharelib.for.spark=spark,hwc
```

For Updating action-level configuration

You can update the action-level configurations to execute Hive commands using both HWC and non-HWC. If you have a workflow which contains an action where you would like to use HWC and another action where you do not want to use HWC, you can achieve the same by specifying the ShareLib properties at the action level.

Example

```
<spark xmlns="uri:oozie:spark-action:1.0">
            ...
            <configuration>
            <property xmlns="">
            <name>oozie.action.sharelib.for.spark</name>
            <value>spark,hwc</value>
            </property>
            </configuration>
            ...
            </spark>
```

Related Information

Hive Warehouse Connector for accessing Apache Spark data

Appendix - Creating a new 'hwc' ShareLib

The oozie admin commands have to be executed by the oozie user.

1. Kinit as oozie.
2. Check the current available ShareLibs:

```
oozie admin -shareliblist -oozie {url}
```

3. Create the folder for it on HDFS:

```
hdfs dfs -mkdir /user/oozie/share/lib/lib_{latestTimestamp}/hwc
```

4. Add the JAR files to it from the /opt/cloudera/parcels/CDH/jars directory:
   - hive-warehouse-connector-assembly-1.0.0.***VERSION NUMBER***-XXX.jar
   - hive-jdbc-3.1.3000.***VERSION NUMBER***-XXX.jar
   - hive-jdbc-handler-3.1.3000.***VERSION NUMBER***-XXX.jar
   - hive-service-3.1.3000.***VERSION NUMBER***-XXX.jar
   - spark-sql-kafka-0-10_2.11-***VERSION NUMBER***-XXX.jar

Note: Do not add any standalone JARs (*-standalone.jar) in this directory.

5. Update the ShareLib property:

```
oozie admin -sharelibupdate -oozie {url}
```

6. List the ShareLibs again to check if hwc is present:

```
oozie admin -shareliblist -oozie {url}
```

Example for using HWC with Oozie Spark action

Understand how you can use the Hive Warehouse Connector (HWC) with Oozie Spark actions through an example that creates an application to read tables from Hive using HWC and display its contents. You can do it either by using a JAR application or by using a Python application.

Using application JAR

This example provides detailed information about the job.properties file, workflow.xml file, and application logic required for this task, and lists the necessary information required for using HWC in Oozie Spark action when you build an application JAR.

Example application logic

You can package the following Scala application logic into a JAR by using either Maven or SBT command line utility with the compatible Cloudera versions. You can call it the org.example package. The following application logic requires only two dependencies - spark-sql and HWC.

```
import com.hortonworks.hwc.HiveWarehouseSession
import org.apache.spark.sql.SparkSession

object ExampleRun {
  def main(args: Array[String]): Unit = {
    println("Using Hive Warehouse connector")
    //Create a Spark session
    val spark = SparkSession.builder().enableHiveSupport().getOrC
reate()
    //Create a HWC session using the Spark Session
    val hive = HiveWarehouseSession.session(spark).build()
    println(args(0)) // Print the input
    //Query the string provided in the arguments using hive.sql()
    hive.sql("SELECT * FROM " + args(0)).show
    //Close the Spark session
    spark.close()
  }
}
```

- Maven method

  Add the following dependencies when you build an application JAR by using Maven:

```
<dependency>
        <groupId>org.apache.spark</groupId>
        <artifactId>spark-sql_2.11</artifactId>
        <version>${spark.version}</version>
    </dependency>
    <dependency>
        <groupId>com.hortonworks.hive</groupId>
        <artifactId>hive-warehouse-connector_2.11</artifactId>
        <version>${hwc.version}</version>
```

```
    </dependency>
```

Create a JAR using the following command:

```
mvn clean package -Dspark.version=<CDP Spark version> -Dhwc.
version=<CDP HWC Version>
```

For example,

```
mvn clean package -Dspark.version=2.4.7.7.1.7.61-1 -Dhwc.ver
sion=1.0.0.7.1.7.61-1
```

- SBT method

  Add the following library dependencies when you build an application JAR by using SBT:

```
val sparkVersion = sys.props.getOrElse("spark.version", "<CDP
 Spark version>")
val hwcVersion = sys.props.getOrElse("hwc.version", "<CDP HWC
 Version>")
libraryDependencies ++= Seq(
  "com.hortonworks.hive" % "hive-warehouse-connector_2.11" % h
wcVersion % "provided",
  "org.apache.spark" %% "spark-sql" % sparkVersion % "provid
ed" force()
)
```

  Create a JAR file using the following command:

```
sbt clean compile assembly -Dspark.version=<CDP Spark version>
 -Dhwc.version=<CDP HWC Version>
```

  For example,

```
sbt clean compile assembly -Dspark.version=2.4.7.7.1.7.61-1 -
Dhwc.version=1.0.0.7.1.7.61-1
```

Save these JAR files in HDFS in a specific location so that it can be used later in the job.properties file. The class name here is org.example.ExampleRun which you will use later while specifying the job.

Example job.properties file

```
HCAT_METASTORE_URI=thrift://myhost-1.myhost.example.site:9083
ROOT_LOGGER_LEVEL=INFO
HCAT_PRINCIPAL=hive/_HOST@EXAMPLE.COM
oozie.action.sharelib.for.spark=spark,hwc
MASTER=yarn
JDBC_PRINCIPAL=hive/_HOST@EXAMPLE.COM
JDBC_URL=jdbc:hive2://myhost-1.myhost.example.site:10001/default;
transportMode=http;httpPath=cliservice;ssl=true;sslTrustStore=/v
ar/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.j
ks;trustStorePassword=update_this_password
oozie.wf.application.path=hdfs:///tmp/workdir
HIVE_TABLE_NAME=sampleTable
JDBC_MODE=JDBC_CLUSTER
APP_NAME=MyApp
MODE=cluster
JAR=hdfs:///tmp/workdir/hwc-examples-1.0.jar
CLASSNAME=org.example.ExampleRun
OOZIE_LAUNCHER_OPTS=”-verbose:class”
```

```
SPARK_OPTS=--conf spark.driver.extraJavaOptions='-verbose:class'
--conf spark.executor.extraJavaOptions='-verbose:class'
```

All the values are example values and are indicative of what you need to write in the file.

HCAT_METASTORE_URI represents the Hive metastore URI and HCAT_PRINCIPAL is the configuration required for Kerberos authentication for the Hive metastore. oozie.action.sharelib.for.spark=spark,hwc must be set as it is. MASTER specifies running Spark in the YARN mode. JDBC_PRINCIPAL is required for Kerberos authentication for HiveServer2. JDBC_URL is required to create a connection to Hive Server 2.

If you run into any classpath issues while executing the Oozie job, then you can check the details by using OOZIE_LAUNCHER_OPTS and SPARK_OPTS. These configurations show you what classes are loaded from which JAR files in the Spark job by checking the YARN logs of the Spark job.

Note: The sharelib folder contains an additional folder by the name hwc and this folder must not contain a *-standalone.jar (standalone JAR files) in it.

Example workflow.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<workflow-app name="spark-hwc-hive-wf" xmlns="uri:oozie:workflow:
1.0">
    <credentials>
        <credential name="hcatauth" type="hcat">
            <property>
                <name>hcat.metastore.uri</name>
                <value>${HCAT_METASTORE_URI}</value>
            </property>
            <property>
                <name>hcat.metastore.principal</name>
                <value>${HCAT_PRINCIPAL}</value>
            </property>
        </credential>
        <credential name="hs2-creds" type="hive2">
            <property>
                <name>hive2.server.principal</name>
                <value>${JDBC_PRINCIPAL}</value>
            </property>
            <property>
                <name>hive2.jdbc.url</name>
                <value>${JDBC_URL}</value>
            </property>
        </credential>
    </credentials>

    <start to="SPARK_HWC_JDBC_READ"/>
    <action name="SPARK_HWC_JDBC_READ" cred="hs2-creds,hcatauth">
        <spark xmlns="uri:oozie:spark-action:1.0">
            <configuration>
                <property>
                    <name>mapreduce.job.hdfs-servers</name>
                    <value>${firstNotNull(wf:conf('HDFS_SERVER
S'),' ')}</value>
                </property>
                <property>
                    <name>oozie.launcher.mapreduce.map.java.opts<
/name>
      <value>${firstNotNull(wf:conf('OOZIE_LAUNCHER_OPTS'),' ')}<
/value>
                </property>
                <property>
```

```
                    <name>oozie.action.rootlogger.log.level</na
me>
                    <value>${firstNotNull(wf:conf('ROOT_LOGGER_L
EVEL'),'INFO')}</value>
                </property>
            </configuration>
            <master>${MASTER}</master>
            <mode>${MODE}</mode>
     <name>${APP_NAME}</name>
     <class>${CLASSNAME}</class>
     <jar>${JAR}</jar>
            <spark-opts>--conf spark.sql.hive.hiveserver2.jdbc
.url=${JDBC_URL}  --conf spark.sql.extensions="com.hortonworks.s
park.sql.rule.Extensions" --conf spark.datasource.hive.warehouse
.read.mode=${JDBC_MODE} --conf spark.sql.hive.hiveserver2.jdbc.u
rl.principal=${JDBC_PRINCIPAL} ${firstNotNull(wf:conf('SPARK_OPT
S'),' ')}</spark-opts>
            <arg>${HIVE_TABLE_NAME}</arg>
        </spark>
        <ok to="end"/>
        <error to="fail"/>
    </action>

    <kill name="fail">
        <message>Workflow failed, error message[${wf:errorMess
age(wf:lastErrorNode())}</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

Save this workflow.xml file in the directory where you have defined oozie.wf.application.path. The different properties seen in "${}" are properties that are written either in the job.properties file or can be passed in the command line. Notice that the <spark-opts> tag contains the necessary configurations that are required for HWC. The <arg> tag contains the input for the application. The <arg> tag is currently set to a Hive table name which is used by a SELECT statement written in the application code.

Note: Do not include the HWC JAR file anywhere in the workflow.xml file as part of the Spark Job. The file is present in the sharelib folder that is explicitly created for HWC.

Using Python application

This example provides detailed information about the job.properties file, workflow.xml file, and application logic required for this task, and lists the necessary information required for using HWC in Oozie Spark action when a Python application is built.

Example application logic

```
import sys
from pyspark.sql import SparkSession
from pyspark_llap import HiveWarehouseSession

spark = SparkSession.builder.enableHiveSupport().getOrCreate()
hwc = HiveWarehouseSession.session(spark).build()
tableName = sys.argv[1]
print "=======Reading hive table - " + tableName + " via HWC===
====="
# Read via HWC
hwc.sql("select * from " + tableName).show()

hwc.close()
```

```
spark.stop()
```

You are using the pyspark module and HWC specific pyspark_llap module for executing the Python program. The pyspark_llap module is derived from the HWC artifacts given in the Cloudera builds.

Example job.properties file

```
HCAT_METASTORE_URI=thrift://myhos
t-1.myhost.example.site:9083
ROOT_LOGGER_LEVEL=INFO
HCAT_PRINCIPAL=hive/_HOST@EXAMPLE.COM
oozie.action.sharelib.for.spark=spark,hwc
MASTER=yarn
JDBC_PRINCIPAL=hive/_HOST@EXAMPLE.COM
JDBC_URL=jdbc:hive2://myhost-1.myhost.example.site:10001/default
;transportMode=http;httpPath=cliservice;ssl=true;sslTrustStore=/
var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.
jks;trustStorePassword=update_this_password
oozie.wf.application.path=hdfs:///tmp/workdir
HIVE_TABLE_NAME=sampleTable
JDBC_MODE=JDBC_CLUSTER
APP_NAME=MyApp
MODE=cluster
PY_FILE=hdfs:///tmp/workdir/testhwcread.py
PYSPARK_HWC_ZIP=/opt/cloudera/parcels/CDH/lib/hive_warehouse_con
nector/pyspark_hwc-1.0.0.7.1.7.61-1.zip
OOZIE_LAUNCHER_OPTS=”-verbose:class”
SPARK_OPTS=--conf spark.driver.extraJavaOptions='-verbose:class'
 --conf spark.executor.extraJavaOptions='-verbose:class'
```

All the values are example values and are indicative of what you need to write in the file.

HCAT_METASTORE_URI represents the hive metastore URI and HCAT_PRINCIPAL is the configuration required for Kerberos authentication for the Hive metastore. oozie.action.shareli b.for.spark=spark,hwc must be set as it is. MASTER specifies running Spark in the YARN mode. JDBC_PRINCIPAL is required for Kerberos authentication for HiveServer2. JDBC_URL is required to create a connection to Hive Server 2.

If you run into any classpath issues while executing the Oozie job, then you can check the details by using OOZIE_LAUNCHER_OPTS and SPARK_OPTS. These configurations show you what classes are loaded from which JAR files in the Spark job by checking the YARN logs of the Spark job.

Note: The sharelib folder contains an additional folder by the name hwc and this folder must not contain a *-standalone.jar (standalone JAR files) in it.

Example workflow.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<workflow-app name="spark-hwc-hive-wf" xmlns="uri:oozie:workflow:
1.0">
    <credentials>
        <credential name="hcatauth" type="hcat">
            <property>
                <name>hcat.metastore.uri</name>
                <value>${HCAT_METASTORE_URI}</value>
            </property>
            <property>
                <name>hcat.metastore.principal</name>
                <value>${HCAT_PRINCIPAL}</value>
            </property>
        </credential>
        <credential name="hs2-creds" type="hive2">
```

```
            <property>
                <name>hive2.server.principal</name>
                <value>${JDBC_PRINCIPAL}</value>
            </property>
            <property>
                <name>hive2.jdbc.url</name>
                <value>${JDBC_URL}</value>
            </property>
        </credential>
    </credentials>

    <start to="SPARK_HWC_JDBC_READ"/>
    <action name="SPARK_HWC_JDBC_READ" cred="hs2-creds,hcatauth">
        <spark xmlns="uri:oozie:spark-action:1.0">
            <configuration>
                <property>
                    <name>mapreduce.job.hdfs-servers</name>
                    <value>${firstNotNull(wf:conf('HDFS_SERVER
S'),' ')}</value>
               </property>
               <property>
                    <name>oozie.launcher.mapreduce.map.java.opts<
/name>
                    <value>${firstNotNull(wf:conf('OOZIE_LAUNCHER
_OPTS'),' ')}</value>
               </property>
               <property>
                    <name>oozie.action.rootlogger.log.level</name
>
                    <value>${firstNotNull(wf:conf('ROOT_LOGGER
_LEVEL'),'INFO')}</value>
               </property>
            </configuration>
            <master>${MASTER}</master>
            <mode>${MODE}</mode>
     <name>${APP_NAME}</name>
     <jar>${PY_FILE}</jar>
            <spark-opts>--conf spark.sql.hive.hiveserver2.jdbc.ur
l=${JDBC_URL}  --conf spark.sql.extensions="com.hortonworks.spar
k.sql.rule.Extensions" --conf spark.datasource.hive.warehouse.re
ad.mode=${JDBC_MODE} --conf spark.sql.hive.hiveserver2.jdbc.url.
principal=${JDBC_PRINCIPAL} --conf spark.submit.pyFiles=${PYSPAR
K_HWC_ZIP} ${firstNotNull(wf:conf('SPARK_OPTS'),' ')}</spark-opt
s>
            <arg>${HIVE_TABLE_NAME}</arg>
        </spark>
        <ok to="end"/>
        <error to="fail"/>
    </action>

    <kill name="fail">
        <message>Workflow failed, error message[${wf:errorMess
age(wf:lastErrorNode())}</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

Save this workflow.xml file in the directory where you have defined oozie.wf.application.path. The different properties seen in “${}” are properties that are written either in the job.properties file or can be passed in the command line. The <spark-opts> tag contains the necessary configurations that are required for HWC. The <arg> tag contains the input for the application that needs to be run. It is currently set to a Hive table name which is used for executing a SELECT query on the same table.

Note: Do not include the HWC JAR anywhere in the workflow.xml file as part of the Spark Job. It is already present in the sharelib folder that is explicitly created for HWC.

Oozie and client configurations

Learn about how Oozie handles the client configuration files for different actions that rely on Hadoop, Hive, and so on, clients.

In certain cases, it is necessary to have the required client configurations (site.xml files) available for action executions in YARN. Previously, you manually copied some site.xml files to either your workflow lib folder or the corresponding Oozie ShareLib. This section summarizes how Oozie utilizes the client configuration files.

Hive2 action

The hive-site.xml is not automatically propagated to YARN. The Hive2 action executes the beeline command, which uses the hive-site.xml file from /etc/hive/conf.

Oozie automatically propagates the hbase-site.xml to YARN if the conditions described in the *Conditions* section are met.

MapReduce and Shell actions

The hbase-site.xml, hive-site.xml, and sqoop-site.xml are propagated to the YARN container if the corresponding conditions mentioned in the *Conditions* section are met.

Spark and Spark3 actions

Oozie automatically propagates the sqoop-site.xml to YARN if the conditions described in the *Conditions* section are met.

Oozie automatically propagates the hive-site.xml to YARN if the Hive service dependency is enabled for Oozie in Cloudera Manager.

Oozie automatically propagates the hbase-site.xml to YARN if the HBase service dependency is enabled for Oozie in Cloudera Manager.

Sqoop action

Oozie automatically propagates the hive-site.xml to YARN if the conditions described in the *Conditions* section are met.

Oozie automatically propagates the sqoop-site.xml to YARN if the Sqoop client dependency is enabled for Oozie in Cloudera Manager.

Oozie automatically propagates the hbase-site.xml to YARN if the HBase service dependency is enabled for Oozie in Cloudera Manager.

Conditions

All the client configurations are only automatically saved to the YARN container if no existing file is already present. This allows you to manually make a client configuration file available in the Workflow's lib folder or the corresponding Oozie ShareLib.

The following applies to the client configurations mentioned in the sections above, where the *Conditions* section is referenced:

To enable Oozie to automatically copy the client configuration files, the following dependencies must be enabled in Cloudera Manager for Oozie:

- hbase-site.xml

  HBase service dependency.
- hive-site.xml

  Hive service dependency.
- sqoop-site.xml

  Sqoop client dependency.

Automatic propagation of these configurations can be disabled by setting the following settings to true:

- hbase-site.xml: oozie.action.propagate.hbase-site.xml.disabled
- hive-site.xml: oozie.action.propagate.hive-site.xml.disabled
- sqoop-site.xml: oozie.action.propagate.sqoop-site.xml.disabled

You can configure these settings globally using a safety-valve, or in the Workflow's global configuration, or in the job.properties file, or at the action-level through the action's configuration in the workflow definition. The order of precedence is as follows:

1. Action-level configuration
2. Workflow global configuration
3. job.properties configuration
4. Global safety-valve configuration

Spark 3 support in Oozie

This section describes how to re-enable the Spark actions, provides guidance on migrating your workflows from Spark actions to Spark 3 actions, highlights the differences between the two, and outlines the enhancements made to Spark actions to simplify the migration process.

Spark actions based on Spark 2 are no longer supported in Oozie because Spark 2 is deprecated. Both schema definitions and action execution implementations for Spark 2 are completely removed from Oozie. Hence, any Workflow file containing a Spark action fails to parse. Since Spark actions are removed and no longer available, you cannot re-enable them. Oozie supports the new Spark 3 based Spark 3 actions. To continue using Apache Spark with Oozie, you must migrate to Spark 3.

Enable Spark actions

While Spark actions are initially disabled, you have the flexibility to enable them at your discretion if you still want to utilize them.

Enabling Spark actions globally

Perform the following steps to enable Spark action globally through Cloudera Manager:

1. Navigate to Oozie's configuration page in Cloudera Manager.
2. Search for Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml.

3. Add a new property named oozie.action.spark.enabled with the value true.

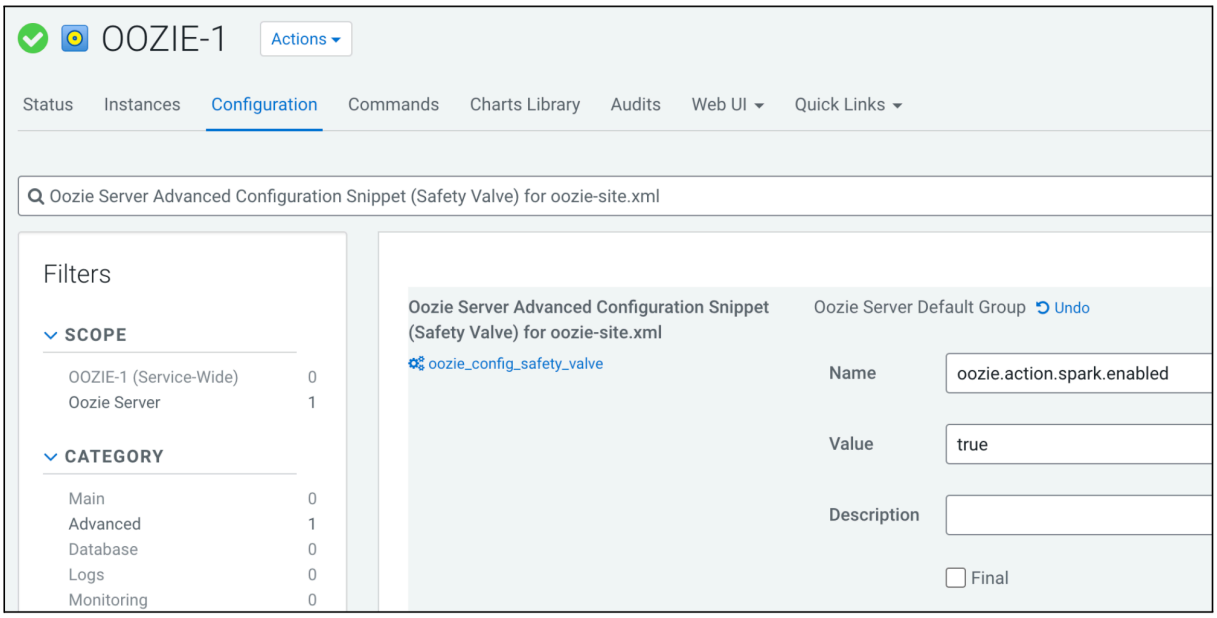

4. Save the modifications.
5. Allow Cloudera Manager some time to recognize the changes.
6. Redeploy Oozie.

Note: You can also enable the above setting through the same safety-valve property before upgrading to Cloudera Private Cloud Base 7.1.9 or higher. By doing so, you can ensure that your scheduled Spark 2 applications with Oozie continue to run successfully after the upgrade, avoiding any potential failures.

Enabling Spark actions per workflows

If you do not want to enable Spark actions globally, you can enable them on a per workflow basis. To do so, set the value of the oozie.action.spark.enabled property to true in the global configuration of the specific workflow. For example:

```
<workflow-app name="spark_workflow" xmlns="uri:oozie:workflow:1.0">
    <global>
        <configuration>
            <property>
                <name>oozie.action.spark.enabled</name>
                <value>true</value>
            </property>
        </configuration>
    </global>
    <start to="spark_action"/>
    <action name="spark_action">
        ...
```

By following this approach, if your workflow contains multiple Spark actions, all of them will be enabled and operational.

You can also enable Spark actions for a given workflow by specifying the oozie.action.spark.enabled property in your job.properties file.

Enabling a given Spark action only

If you want to further restrict the enablement of a Spark action, you have the option to enable it only for a specific Spark action within a workflow. This can be achieved by utilizing the oozie.action.spark.enabled property in the configuration of that particular action. For example:

```
<workflow-app name="spark_workflow" xmlns="uri:oozie:workflow:1.0">
    <start to="spark_action"/>
    <action name="spark_action">
        <spark xmlns="uri:oozie:spark-action:1.0">
            <resource-manager>${resourceManager}</resource-manager>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>oozie.action.spark.enabled</name>
                    <value>true</value>
                </property>
            </configuration>
            ...
```

The order of precedence for the above options is as follows:

1. If you have configured the property at the action level, it takes precedence over all other settings, and the remaining configurations are disregarded.
2. If you have configured the property in the global configuration of the workflow, the value from there is used.
3. If the setting is not available in either of the previous locations, the value configured in your job.properties file is used instead.
4. Lastly, the global safety-valve setting comes into effect.

Use Spark actions with a custom Python executable

Spark 2 supports both PySpark and JavaSpark applications. Learn how to use a custom Python executable in a given Spark action.

In case of PySpark, in Spark 2, you can designate a custom Python executable for your Spark application by utilizing the spark.pyspark.python Spark conf argument. For more details, see Spark 2.4 documentation. Consequently, if you include the spark.pyspark.python Spark conf argument in your Oozie Spark action, the Python executable you specify is used when executing the Spark action through Oozie.

To simplify the usage of a customized Python executable with Oozie's Spark action, you can use the oozie.service.SparkConfigurationService.spark.pyspark.python property. This property functions similarly to Spark's spark.pyspark.python conf argument, allowing you to specify a custom Python executable. Oozie then passes this executable to the underlying Spark application executed through Oozie.

You can specify the oozie.service.SparkConfigurationService.spark.pyspark.python property in different ways.

Setting Spark actions with a custom Python executable globally

You can set it globally in Cloudera Manager through a safety-valve. To do that, perform the following steps:

1. Navigate to Oozie's configuration page in Cloudera Manager.
2. Search for Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml.
3. Add a new property named oozie.service.SparkConfigurationService.spark.pyspark.python.

4. Specify its value to point to your custom Python executable.

   For example, if you installed Python 3.7 to /opt/python37-for-oozie, then specify the value as /opt/python37-for-oozie/bin/python3.

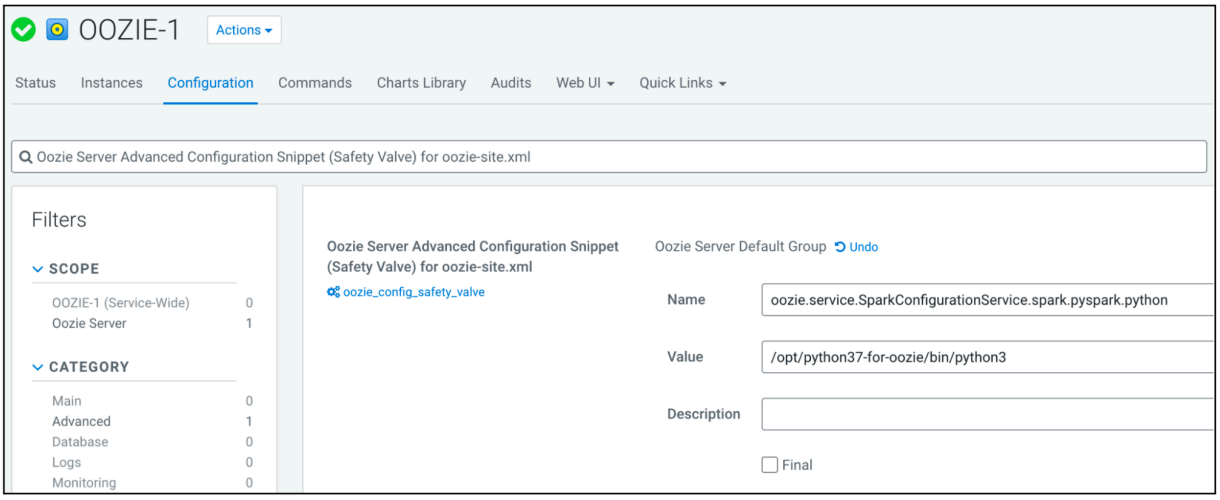

5. Save the modifications.
6. Allow Cloudera Manager some time to recognize the changes.
7. Redeploy Oozie.

Setting Spark actions with a custom Python executable per workflows

You can also specify a custom Python executable for a given workflow using the same property:

```
<workflow-app name="spark_workflow" xmlns="uri:oozie:workflow:1.0">
    <global>
        <configuration>
            <property>
                <name>oozie.service.SparkConfigurationService.spark.pyspark.
python</name>
                <value>/opt/python37-for-oozie/bin/python3</value>
            </property>
        </configuration>
    </global>
    <start to="spark_action"/>
    <action name="spark_action">
        ...
```

The same workflow-level Python executable can be achieved if you set the property in your job.properties file.

Setting Spark actions with a custom Python executable for a given Spark action only

Finally, you can only change the Python executable for a given Spark action. For example:

```
<workflow-app name="spark_workflow" xmlns="uri:oozie:workflow:1.0">
    <start to="spark_action"/>
    <action name="spark_action">
        <spark xmlns="uri:oozie:spark-action:1.0">
            <resource-manager>${resourceManager}</resource-manager>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>oozie.service.SparkConfigurationService.spark.pys
park.python</name>
                    <value>/opt/python37-for-oozie/bin/python3</value>
```

```
                </property>
            </configuration>
            ...
```

The following order of precedence is applied for this configuration:

1. Oozie does not override the configuration of spark.pyspark.python in the <spark-opts> tag of your action definition if you have already set it.
2. If you have configured the property at the action level, it takes precedence over all other settings, and the remaining configurations are disregarded.
3. If you have configured the property in the global configuration of the workflow, the value from there is used.
4. If the setting is not available in either of the previous locations, the value configured in your job.properties file is used.
5. Lastly, the global safety-valve setting comes into effect.

It is also possible to inform Oozie that you do not want to use a custom Python executable in a given Spark action, but you want to use the default one configured for Spark 2, even if you already configured at a lower level of precedence. For instance, if the oozie.service.SparkConfigurationService.spark.pyspark.python is set as a safety-valve to /opt/python37-for-oozie/bin/python3, but in a workflow or in a specific action you want to use the default Python executable configured for Spark 2, you can set the value of the property to default. For example:

```
<workflow-app name="spark_workflow" xmlns="uri:oozie:workflow:1.0">
    <global>
        <configuration>
            <property>
                <name>oozie.service.SparkConfigurationService.spark.pyspark.
python</name>
                <value>default</value>
            </property>
        </configuration>
    </global>
    <start to="spark_action"/>
    <action name="spark_action">
        ...
```

In this scenario, the value set in Cloudera Manager in Oozie's safety-valve configuration, is ignored, and the spark.pyspark.python Spark conf is not set at all.

Important: If you choose to use a custom Python executable in Spark actions, then the given executable must be available on all nodes where YARN Node Manager is also available.

Note: Since Spark 2 supports Python up to Python 3.7, you can use the above configuration option to specify any custom Python installation from 2.7 to 3.7.

Spark 3 Oozie action schema

Refer to the following for the schema of a Spark 3 Oozie action.

```
<xs:schema elementFormDefault="qualified"
           targetNamespace="uri:oozie:spark3-action:1.0"
           xmlns:spark3="uri:oozie:spark3-action:1.0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="oozie-common-1.0.xsd"/>
    <xs:element name="spark3" type="spark3:ACTION"/>
    <xs:complexType name="ACTION">
        <xs:sequence>
            <xs:element maxOccurs="1" minOccurs="1" name="resource-manager"
 type="xs:string"/>
```

```
            <xs:element maxOccurs="1" minOccurs="0" name="name-node" type=
"xs:string"/>
            <xs:element maxOccurs="1" minOccurs="0" name="prepare" type="s
park3:PREPARE"/>
            <xs:element maxOccurs="1" minOccurs="0" name="launcher" type="s
park3:LAUNCHER"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="job-xml"
 type="xs:string"/>
            <xs:element maxOccurs="1" minOccurs="0" name="configuration" typ
e="spark3:CONFIGURATION"/>
            <xs:element maxOccurs="1" minOccurs="1" name="master" type="x
s:string"/>
            <xs:element maxOccurs="1" minOccurs="0" name="mode" type="xs:str
ing"/>
            <xs:element maxOccurs="1" minOccurs="1" name="name" type="xs:
string"/>
            <xs:element maxOccurs="1" minOccurs="0" name="class" type="xs:
string"/>
            <xs:element maxOccurs="1" minOccurs="1" name="jar" type="xs:st
ring"/>
            <xs:element maxOccurs="1" minOccurs="0" name="spark-opts" type="
xs:string"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="arg" type
="xs:string"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="file" t
ype="xs:string"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="archive"
 type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Differences between Spark and Spark 3 actions

Learn about the differences between a Spark action and a Spark 3 action definition, the difference in the logging frameworks used in Spark and Spark 3, and the action credentials.

There are several notable distinctions between a Spark action and a Spark 3 action definition. Firstly, the XML element representing the action differs between the two. For Spark actions, it is denoted as spark, whereas for Spark 3 actions, it is labeled as spark3.

Additionally, the support for certain tags also varies. Specifically, the Spark 3 action definition does not include support for the job-tracker tag, but instead exclusively employs the resource-manager tag.

In contrast to Oozie's Spark action, where omitting the <mode> tag in the workflow definition allows setting the <master> tag to either yarn-cluster or yarn-client, the same flexibility does not apply to Spark 3 actions. With Spark 3 actions, if you previously set the <master> tag to yarn-cluster in Spark actions, you must set the <master> tag to yarn and the <mode> tag to cluster in Spark 3 actions. Similarly, if you previously set the <master> tag to yarn-client in Spark actions, for Spark 3 actions, you must set the <master> tag to yarn and the <mode> tag to client.

Differences in the actions' configuration

To configure Spark actions, there are multiple configuration options available. These options are prefixed with oozie.service.SparkConfigurationService. For example:

- oozie.service.SparkConfigurationService.spark.configurations.blacklist
- oozie.service.SparkConfigurationService.hive2.configurations

In order to differentiate between configurations for Spark actions and Spark 3 actions, the prefix for properties related to Spark 3 actions is modified to oozie.service.Spark3ConfigurationService.

During the migration from Spark actions to Spark 3 actions, it is necessary to adjust the prefix for any Spark action configuration that is configured in Cloudera Manager using a safety-valve.

Log4j vs. Log4j 2

When comparing Spark 2 and Spark 3, there is a difference in the logging frameworks used. While Spark 2 relies on log4j or reload4j, Spark 3 has transitioned to log4j2. As a result, Oozie's Spark 3 action also utilizes log4j2.

Oozie provides a logging property file for both Spark and Spark 3 actions. However, there is a distinction in the naming conventions of these files. In the case of Spark actions, the file is named spark-log4j.properties, whereas for Spark 3 actions, it is named spark3-log4j2.properties.

During the migration process from Spark to Spark 3 actions, if you have a custom spark-log4j.properties file located in the lib folder of your workflow or within Oozie's ShareLib, you need to rename this file. Since Spark 3 uses log4j2, you might also need to modify your custom Spark logging configuration file to ensure compatibility with log4j2.

Action credentials

The Spark 3 action in Oozie provides support for the same credentials as Spark actions.

Use Spark 3 actions with a custom Python executable

Learn how to use a custom Python executable in a given Spark 3 action.

Similar to Spark 2, Spark 3 also provides the capability to define a custom Python executable for use with spark3-submit through the spark.pyspark.python Spark 3 conf argument. For more details, see the Latest Spark3 documentation. Consequently, if you include the spark.pyspark.python Spark 3 conf in your Oozie Spark 3 action, the Python executable you specify is used when executing the Spark 3 action through Oozie.

To simplify the usage of a customized Python executable with Oozie's Spark 3 action, you can use the oozie.service.Spark3ConfigurationService.spark.pyspark.python property. This property functions similar to Spark 3's spark.pyspark.python conf argument, allowing you to specify a custom Python executable. Oozie then passes this executable to the underlying Spark 3 application executed through Oozie.

You can specify this configuration in different ways.

Setting Spark 3 actions with a custom Python executable globally

You can set it globally in Cloudera Manager. To do that, perform the following steps:

1. Navigate to Oozie's configuration page in Cloudera Manager.
2. Search for Python Executable for Spark3 Actions.

3. Specify its value to point to your custom Python executable.

   For example, if you installed Python 3.7 to /opt/python37-for-oozie, then specify the value as /opt/python37-for-oozie/bin/python3.

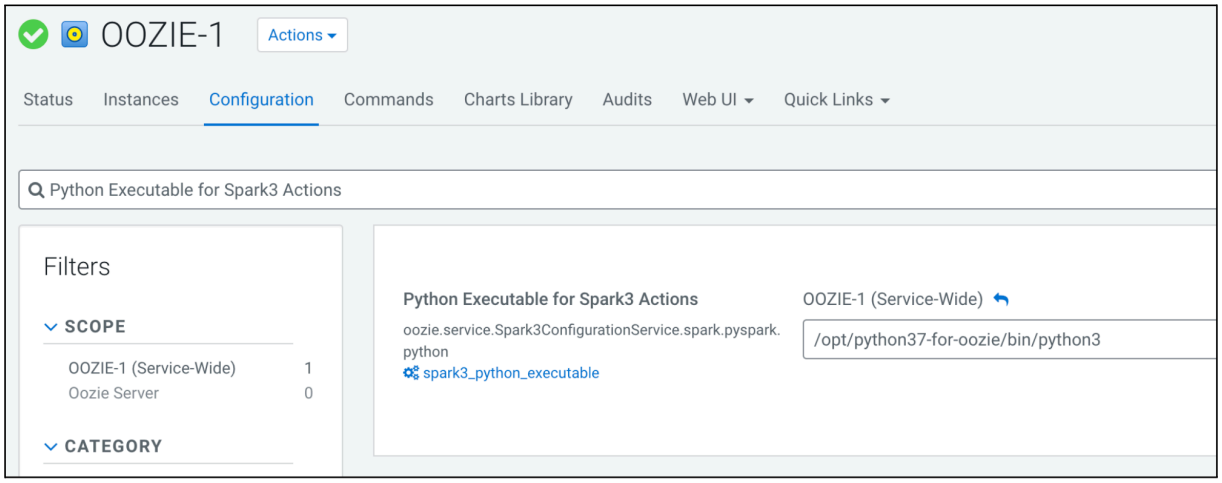

4. Save the modifications.
5. Allow Cloudera Manager some time to recognize the changes.
6. Redeploy Oozie.

Setting Spark 3 actions with a custom Python executable per workflows

You can also specify a custom Python executable for a given workflow using the same property:

```
<workflow-app name="spark_workflow" xmlns="uri:oozie:workflow:1.0">
    <global>
        <configuration>
            <property>
                <name>oozie.service.Spark3ConfigurationService.spark.pyspark
.python</name>
                <value>/opt/python37-for-oozie/bin/python3</value>
            </property>
        </configuration>
    </global>
    <start to="spark_action"/>
    <action name="spark_action">
        ...
```

The same workflow-level Python executable can be achieved if you set the property in your job.properties file.

Setting Spark 3 actions with a custom Python executable for a given Spark action only

Finally, you can only change the Python executable for a given Spark 3 action. For example:

```
<workflow-app name="spark_workflow" xmlns="uri:oozie:workflow:1.0">
    <start to="spark_action"/>
    <action name="spark_action">
        <spark3 xmlns="uri:oozie:spark3-action:1.0">
            <resource-manager>${resourceManager}</resource-manager>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>oozie.service.Spark3ConfigurationService.spark.
pyspark.python</name>
                    <value>/opt/python37-for-oozie/bin/python3</value>
                </property>
```

```
            </configuration>
            ...
```

The following order of precedence is applied for this configuration:

1. Oozie does not override the configuration of spark.pyspark.python in the <spark-opts> tag of your action definition if you have already set it.
2. If you have configured the property at the action level, it takes precedence over all other settings, and the remaining configurations are disregarded.
3. If you have configured the property in the global configuration of the workflow, the value from there is used.
4. If the setting is not available in either of the previous locations, the value configured in your job.properties file is used.
5. Lastly, the global setting in Cloudera Manager comes into effect.

It is also possible to inform Oozie that you do not want to use a custom Python executable in a given Spark 3 action, but you want to use the default one configured for Spark 3, even if you already configured at a lower level of precedence. For instance, if the Python Executable for Spark3 Actions property is set in Cloudera Manager to /opt/python37-for-oozie/bin/python3, but in a workflow or in a specific action you want to use the default Python executable configured for Spark 3, you can set the value of the property to default. For example:

```
<workflow-app name="spark_workflow" xmlns="uri:oozie:workflow:1.0">
    <global>
        <configuration>
            <property>
                <name>oozie.service.Spark3ConfigurationService.spark.pyspark
.python</name>
                <value>default</value>
            </property>
        </configuration>
    </global>
    <start to="spark_action"/>
    <action name="spark_action">
        ...
```

In this scenario, the value set in Cloudera Manager is ignored, and the spark.pyspark.python Spark 3 conf is not set at all.

Important: If you choose to use a custom Python executable in Spark 3 actions, then the given executable must be available on all nodes where YARN Node Manager is also available.

Spark 3 compatibility action executor

To facilitate smooth transitions from Oozie's Spark actions to Spark 3 actions, you can use the Spark 3 compatibility action executor. The purpose of this executor is to allow you to retain your existing Spark action definitions in your workflows while executing them with Spark 3 instead of Spark 2.

When you use this executor, Oozie automatically converts a Spark action definition to a Spark 3 action definition before executing it. To configure the Spark 3 action, Oozie utilizes the Spark action configurations (prefixed with oozie.service.SparkConfigurationService) and converts them to Spark 3 configurations.

This feature enables you to run Spark 3 actions without making any modifications to your workflow definitions.

Important: However, it is important to note that this compatibility executor should be considered as a temporary solution while you transition your Spark action definitions to Spark 3 action definitions. It might not support all the features provided by the actual Spark 3 action executor.

Additionally, remember that your Python or Java Spark actions must be runtime and binary compatible with Spark 3. This is necessary to execute them using the compatibility action executor. For more detailed information on this topic, see the *Migration of Spark 2 applications* in this document.

To enable Spark 3 compatibility mode for Spark 2 action workflow definitions, you use the oozie.action.spark.compatibility property.

You can configure this property in the following ways:

- Global configuration:
  1. Go to the Oozie configuration page in Cloudera Manager and search for Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml.
  2. Add a new key with the name oozie.action.spark.compatibility and set the value to true.
  3. Redeploy Oozie.
- Workflow-level configuration

  Add the property to the global configuration section of your workflow.xml file. Alternatively, you can add it to your job.properties file.
- Action-level Configuration

  If you only want to enable compatibility mode for a specific Spark action in your workflow definition, add an action-level property with the name oozie.action.spark.compatibility and set the value to true.

Limitations

Due to the high customizability of Oozie's Spark and Spark 3 actions, certain restrictions and limitations have been introduced. The current known limitations are outlined as follows:

- The following properties are not permitted in your action configuration, which includes configurations at the global level, workflow level, and action level:
  - oozie.action.sharelib.for.spark
  - oozie.action.sharelib.for.spark3
  - oozie.action.sharelib.for.spark.exclude
  - oozie.action.sharelib.for.spark3.exclude
- As the Spark 3 action utilizes log4j2 instead of log4j (used by the Spark action), it is crucial to avoid mixing the two in compatibility mode. Therefore, having a spark-log4j.properties file in the lib folder of your workflow or in ShareLib is not allowed.

Related Information

Migration of Spark 2 applications

Spark 3 examples with Python or Java application

This section provides you some examples of Spark 3 with Python and Java applications.

Spark3 with a Java application with Hive2 credentials

```
<workflow-app name="JavaSpark" xmlns="uri:oozie:workflow:1.0">
    <credentials>
        <credential name="hive-credential" type="hive2">
            <property>
                <name>hive2.jdbc.url</name>
                <value>jdbc:hive2://...</value>
            </property>
            <property>
                <name>hive2.server.principal</name>
                <value>...</value>
            </property>
        </credential>
    </credentials>
    <start to="spark-node-javaspark"/>
    <action name="spark-node-javaspark" cred="hive-credential">
```

```
        <spark3 xmlns="uri:oozie:spark3-action:1.0">
            <resource-manager>${resourceManager}</resource-manager>
            <name-node>${nameNode}</name-node>
            <master>${master}</master>
            <mode>${mode}</mode>
            <name>JavaSpark-Example</name>
            <class>com.company.spark.JavaSpark</class>
            <jar>${nameNode}/user/${wf:user()}/javaspark_lib/JavaSparkForOo
zie.jar</jar>
            <arg>${inputFile}</arg>
        </spark3>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Workflow failed, error
            message[${wf:errorMessage(wf:lastErrorNode())}]
        </message>
    </kill>
    <end name="end"/>
</workflow-app>
```

Spark3 with a Python application with Hive2 credentials

```
<workflow-app name="PySpark" xmlns="uri:oozie:workflow:1.0">
    <credentials>
        <credential name="hive-credential" type="hive2">
            <property>
                <name>hive2.jdbc.url</name>
                <value>jdbc:hive2://...</value>
            </property>
            <property>
                <name>hive2.server.principal</name>
                <value>...</value>
            </property>
        </credential>
    </credentials>
    <start to="spark-node-pyspark"/>
    <action name="spark-node-pyspark" cred="hive-credential">
        <spark3 xmlns="uri:oozie:spark3-action:1.0">
            <resource-manager>${resourceManager}</resource-manager>
            <name-node>${nameNode}</name-node>
            <master>${master}</master>
            <mode>${mode}</mode>
            <name>PySpark-Example</name>
            <jar>${pythonScript}</jar>
            <arg>${inputFile}</arg>
        </spark3>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Workflow failed, error
            message[${wf:errorMessage(wf:lastErrorNode())}]
        </message>
    </kill>
    <end name="end"/>
</workflow-app>
```

Shell action for Spark 3

Learn about how to execute Spark 3's spark3-submit through Oozie's Shell action.

Similar to the support for executing Spark 2's spark-submit through Oozie's Shell action, Cloudera also provides full support for executing Spark 3's spark3-submit through Oozie's Shell action.

As Oozie utilizes delegation tokens instead of Kerberos tickets in its YARN applications, it is recommended to unset the HADOOP_TOKEN_FILE_LOCATION environment variable in your Shell script before executing spark3-submit, if you intend to use spark3-submit without relying on Oozie's default delegation tokens. This is because spark3-submit might not function properly with both delegation tokens and Kerberos tickets. However, to ensure the successful completion of your Shell action, ensure that you reset the HADOOP_TOKEN_FILE_LOCATION environment variable after the execution of your custom Shell script segment. The following example illustrates how you can accomplish this:

```
#!/usr/bin/env bash
# By executing the commands within brackets,
# we can ensure that the parent environment remains untouched
(
    unset HADOOP_TOKEN_FILE_LOCATION

    kinit -kt /var/keytabs/user.keytab user
    /usr/bin/spark3-submit --master yarn --deploy-mode cluster \
        create_table_with_data_spark3.py tableUsingSpark3FromShellAction
    /usr/bin/spark3-submit --master yarn --deploy-mode cluster \
        read_created_table.py tableUsingSpark3FromShellAction
)
```

Migration of Spark 2 applications

To ensure a smooth migration of your underlying Spark 2 application when using Oozie's Spark 3 action, it is highly recommended to follow the official Spark 2 to Spark 3 migration guide. Specifically, refer to the Cloudera Runtime documentation's comprehensive resource titled *Migrating Spark applications*.

This guide provides you with detailed instructions and best practices specifically tailored to the migration process. By adhering to this guide, you can effectively transition your Spark 2 application and leverage the capabilities of Spark 3 seamlessly within Oozie.

It is highly advised to first test your migrated Spark 3 applications directly using the official Spark 3 runtime. By executing your applications with spark3-submit before running them as Oozie actions, you can identify any potential issues and find the root cause. This approach enables you to determine whether any problems arise from Oozie, Spark 3 itself, or the compatibility of your migrated application. Taking this proactive step assists you in troubleshooting and resolving any potential obstacles during the migration process.

Migrating Java applications

If you are currently running Java applications with Oozie's Spark action, there are several important considerations and steps to follow:

- Recompile your application using Spark 3 dependencies instead of Spark 2 dependencies.
- If you are using Scala, it may be necessary to migrate from Scala 2.11 to Scala 2.12 according to the official Spark migration guide.
- If your application relies on the Scala module of the Jackson library, you might need to replace the Scala 2.11 flavor with the Scala 2.12 flavor.
- Ensure that your 3rd-party runtime dependencies align with the versions used by Spark 3 and Oozie's Spark 3 action.
- Since Spark 3 has transitioned from log4j to log4j2, you might need to adjust the logging library and/or logging configuration used in your application. Additionally, ensure that all necessary logging frameworks are present

in the classpath. Note that Oozie's Spark 3 action executor configures log4j2 instead of log4j, meaning that log4j runtime libraries are no longer included in the classpath by default, only log4j2 libraries are.

By following these guidelines, you can successfully migrate your Java applications to work seamlessly with Oozie's Spark 3 action.

Migrating Python applications

When upgrading PySpark applications from Spark 2 to Spark 3, it is important to consider not only the framework migration but also the compatibility of your Python application with Python 3. If your Python application is not compatible, you need to perform a migration from Python 2 to Python 3 as well.

Given that Spark 2 is now deprecated and Python 2 has reached its end of life, Cloudera strongly advise migrating your PySpark applications from Python 2 and Spark2 simultaneously. This approach ensures that your application runs on up-to-date frameworks and avoids potential security vulnerabilities.

However, depending on the nature of your applications (such as their size or quantity), you might want to take the following steps:

1. Follow the instructions in the *Enable Spark actions* section to re-enable Spark actions.
2. Install Python 2 to a custom location to prevent interference with other services and avoid its usage by other services.
3. Configure Oozie's Spark action globally to use this custom Python 2 installation. Refer to the *Using Spark actions with a custom Python executable* section for guidance.
4. Additionally, install Python 3.7 to a custom location.
5. Begin migrating your PySpark applications to ensure compatibility with Python 3.
6. For workflows where you have already made the underlying PySpark application Python 3 compatible, enable them to run with the custom Python 3 installation instead of Python 2. Refer to the *Using Spark actions with a custom Python executable* section for detailed instructions.
7. After all the Spark actions in a workflow are compatible with Python 3, start migrating the Spark actions to Spark 3 actions.

By following these steps, you can successfully migrate your PySpark applications from Python 2 and Spark 2 to Python 3 and Spark 3.

Related Information

Enable Spark actions

Use Spark actions with a custom Python executable

Migrating Spark applications

Hue support for Oozie

As Hue's Oozie designer feature is not currently being enhanced, there are no plans to provide support for Oozie Spark 3 actions within the Hue platform at this time.

Oozie High Availability

Oozie High Availability is "active-active" so that both Oozie servers are active at the same time, with no failover. High availability for Oozie is supported in both MRv1 and MRv2 (YARN).

Requirements for Oozie High Availability

You must ensure your cluster meets all the requirements for configuring Oozie High Availability (HA).

- Multiple active Oozie servers, preferably identically configured.

- JDBC JAR in the same location across all Oozie hosts (for example, /var/lib/oozie/).
- External database that supports multiple concurrent connections, preferably with HA support.
- ZooKeeper ensemble with distributed locks to control database access, and service discovery for log aggregation.
- Load balancer (preferably with HA support, for example HAProxy), virtual IP, or round-robin DNS to provide a single entry point (of the multiple active servers), and for callbacks from the Application Master or JobTracker.

Configuring Oozie High Availability using Cloudera Manager

You can use Cloudera Manager to enable or disable Oozie High Availability (HA).

Important: Enabling or disabling HA makes the previous monitoring history unavailable.

Oozie Load Balancer configuration

To enable Oozie High Availability, you must manually configure a Load Balancer.

About this task

Cloudera recommends using the HAProxy Load Balancer. These steps explain how to configure the HAProxy load balancer. However, you can choose to configure a different Load Balancer.

Procedure

1. Install HAProxy on the host where you are setting up and configuring the Oozie load balancer. For more information, see the HAProxy documentation.
2. You must configure the Oozie load balancer for both HTTP and HTTPS ports.

```
This is an example:
global
    log         127.0.0.1 local2
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
    stats       socket /tmp/haproxy

defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          10m
    timeout server          10m
    timeout check           10s
    maxconn                 3000

listen admin
        bind *:8000
        stats enable

#---------------------------------------------------------------------
# main frontend which proxys to the backends
```

```
#---------------------------------------------------------------------
frontend                        oozie_front
    bind                        *:5000 ssl crt /var/lib/cloudera-scm-agent
/agent-cert/cdep-host_key_cert_chain_decrypted.pem
    default_backend             oozie

#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend oozie
    balance                     roundrobin
    server  oozie1 my-oozie-host-1:11443/oozie check ssl ca-file /var/lib/
cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
    server  oozie2 my-oozie-host-2:11443/oozie check ssl ca-file /var/lib/
cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
    server  oozie3 my-oozie-host-3:11443/oozie check ssl ca-file /var/lib/
cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem

#---------------------------------------------------------------------
# main frontend which proxys to the http backends
#---------------------------------------------------------------------
frontend                        oozie_front_http
    bind                        *:5002
    default_backend             oozie_http

#---------------------------------------------------------------------
# round robin balancing between the various http backends
#---------------------------------------------------------------------
backend oozie_http
    balance                     roundrobin
    server  oozie_http1 my-oozie-host-1:11000/oozie check
    server  oozie_http2 my-oozie-host-2:11000/oozie check
    server  oozie_http3 my-oozie-host-3:11000/oozie check
```

Using the example, the load balancer is setup for three Oozie instances. The load balancer listens on port 5002 for HTTP connections and forwards it to Oozie's port 11000. The load balancer listens on port 5000 for HTTPS connections and forwards it to Oozie's port 11443.

If you not enabled SSL in Oozie, then you do not need the HTTPS load balancer. For HTTPS load balancing, ensure that you set up the certificate.

3. Continue to configure the load balancer by enabling Oozie High Availability. For information about enabling Oozie High Availability, see Enabling Oozie High Availability.

Enabling Oozie High Availability

You must select the host on which you want to install the additional Oozie server and specify the required property values to install Oozie High Availability (HA).

Before you begin

Ensure that the requirements are satisfied.

Procedure

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Select Actions Enable High Availability to see eligible Oozie server hosts. The host running the current Oozie server is not eligible.
3. Select the host on which to install an additional Oozie server and click Continue.

4. Update the following fields for the Oozie load balancer:
   - Hostname

     For example:

     ```
     NIGHTLY6X-1.VPC.CLOUDERA.COM
     ```
   - HTTP Port

     For example:

     ```
     5002
     ```
   - HTTPS Port

     For example:

     ```
     5000
     ```
5. Click Continue.

   Cloudera Manager stops the Oozie servers, adds another Oozie server, initializes the Oozie server High Availability state in ZooKeeper, configures Hue to reference the Oozie load balancer, and restarts the Oozie servers and dependent services. In addition, Cloudera Manager generates Kerberos credentials for the new Oozie server and regenerates credentials for existing servers.

Disabling Oozie High Availability

Based on your requirements, you can disable Oozie High Availability (HA) using Cloudera Manager.

Procedure

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Select Actions Disable High Availability to see all hosts currently running Oozie servers.
3. Select the one host to run the Oozie server and click Continue.

   Cloudera Manager stops the Oozie service, removes the additional Oozie servers, configures Hue to reference the Oozie service, and restarts the Oozie service and dependent services.

Scheduling in Oozie using cron-like syntax

Most Linux distributions include the cron utility, which is used for scheduling time-based jobs. You can schedule Oozie using Cron-like syntax.

Location

Set the scheduling information in the frequency attribute of the coordinator.xml file. A simple file looks like the following example. The frequency attribute and scheduling information appear in bold.

```
<coordinator-app name="MY_APP" frequency="30 14 * * *"
start="2009-01-01T05:00Z" end="2009-01-01T06:00Z" timezone="UTC" xmlns="ur
i:oozie:coordinator:0.5">
   <action>
      <workflow>
         <app-path>hdfs://localhost:8020/tmp/workflows</app-path>
      </workflow>
   </action>
</coordinator-app>
```

Syntax and structure

The cron-like syntax used by Oozie is a string with five space-separated fields:

- minute
- hour
- day-of-month
- month
- day-of-week

The structure takes the form of * * * * *. For example, 30 14 * * * means that the job runs at at 2:30 p.m. everyday. The minute field is set to 30, the hour field is set to 14, and the remaining fields are set to *.

Allowed values and special characters

The following table describes special characters allowed and indicates in which fields they can be used.

Table 1: Special characters

Character	Fields Allowed	Description
* (asterisk)	All	Match all values.
, (comma)	All	Specify multiple values.
- (dash)	All	Specify a range.
/ (forward slash)	All	Specify an increment.
? (question mark)	Day-of-month, day-of-week	Indicate no specific value (for example, if you want to specify one but not the other).
L	Day-of-month, day-of-week	Indicate the last day of the month or the last day of the week (Saturday). In the day-of-week field, 6L indicates the last Friday of the month.
W	Day-of-month	Indicate the nearest weekday to the given day.
# (pound sign)	Day-of-week	Indicate the nth day of the month

The following table summarizes the valid values for each field.

Field	Allowed Values	Allowed Special Characters
Minute	0-59	, - * /
Hour	0-23	, - * /
Day-of-month	0-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /
Day-of-week	1-7 or SUN-SAT	, - * ? / L #

Important: Some cron implementations accept 0-6 as the range for days of the week. Oozie accepts 1-7 instead.

Oozie scheduling examples

You can use cron scheduling in Oozie to ensure that the jobs run according to the criteria that you specify.

The following examples show cron scheduling in Oozie. Oozie's processing time zone is UTC. If you are in a different time zone, add to or subtract from the appropriate offset in these examples.

Run at the 30th minute of every hour

Set the minute field to 30 and the remaining fields to * so they match every value.

```
frequency="30 * * * *"
```

Run at 2:30 p.m. every day

Set the minute field to 30, the hour field to 14, and the remaining fields to *.

```
frequency="30 14 * * *"
```

Run at 2:30 p.m. every day in February

Set the minute field to 30, the hour field to 14, the day-of-month field to *, the month field to 2 (February), and the day-of-week field to *.

```
frequency="30 14 * 2 *"
```

Run every 20 minutes between 5:00-10:00 a.m. and between 12:00-2:00 p.m. on the fifth day of each month

Set the minute field to 0/20, the hour field to 5-9,12-13, the day-of-month field to 0/5, and the remaining fields to *.

```
frequency="0/20 5-9,12-13 0/5 * *"
```

Run every Monday at 5:00 a.m.

Set the minute field to 0, the hour field to 5, the day-of-month field to ?, the month field to *, and the day-of-week field to MON.

```
frequency="0 5 ? * MON"
```

Note: If the ? was set to *, this expression would run the job every day at 5:00 a.m., not just Mondays.

Run on the last day of every month at 5:00 a.m.

Set the minute field to 0, the hour field to 5, the day-of-month field to L, the month field to *, and the day-of-week field to ?.

```
frequency="0 5 L * ?"
```

Run at 5:00 a.m. on the weekday closest to the 15th day of each month

Set the minute field to 0, the hour field to 5, the day-of-month field to 15W, the month field to *, and the day-of-week field to ?.

```
frequency="0 5 15W * ?"
```

Run every 33 minutes from 9:00-3:00 p.m. on the first Monday of every month

Set the minute field to 0/33, the hour field to 9-14, the day-of-week field to 2#1 (the first Monday), and the remaining fields to *.

```
frequency="0/33 9-14 ? * 2#1"
```

Run every hour from 9:00 a.m.-5:00 p.m. on weekdays

Set the minute field to 0, the hour field to 9-17, the day-of-month field to ?, the month field to *, and the day-of-week field to 2-6.

```
frequency="0 9-17 ? * 2-6"
```

Run on the second-to-last day of every month

Set the minute field to 0, the hour field to 0, the day-of-month field to L-1, the month field to *, and the day-of-week field to ?.

```
frequency="0 0 L-1 * ?"
```

Note: “L-1# means the second-to-last day of the month.

Oozie uses Quartz, a job scheduler library, to parse the cron syntax. For more examples, go to the CronTrigger Tutorial on the Quartz website. Quartz has two fields (second and year) that Oozie does not support.

Configuring an external database for Oozie

Oozie is a stateless web application by design. All information about running and completed workflows, coordinators, and bundle jobs are stored in a relational database. Oozie supports an embedded PostgreSQL database; however, Cloudera strongly recommends that you use an external database for production systems.

Related Information

Oozie database configurations

Configuring PostgreSQL for Oozie

You must install PostgreSQL, create the Oozie user and database, and configure PostgreSQL to accept network connections for the Oozie user.

Procedure

1. Install PostgreSQL

   See the PostgreSQL documentation to install it.
2. Create the Oozie User and Oozie Database.

   For example, using the PostgreSQL psql command-line tool:

```
$ psql -U postgres
Password for user postgres: *****

postgres=# CREATE ROLE oozie LOGIN ENCRYPTED PASSWORD 'oozie'
 NOSUPERUSER INHERIT CREATEDB NOCREATEROLE;
CREATE ROLE

postgres=# CREATE DATABASE "oozie" WITH OWNER = oozie
 ENCODING = 'UTF8'
 TABLESPACE = pg_default
 LC_COLLATE = 'en_US.UTF-8'
 LC_CTYPE = 'en_US.UTF-8'
 CONNECTION LIMIT = -1;
CREATE DATABASE

postgres=# \q
```

3. Configure PostgreSQL to Accept Network Connections for the Oozie User.
   a) Edit the postgresql.conf file and set the listen_addresses property to *, to make sure that the PostgreSQL server starts listening on all your network interfaces. Also make sure that the standard_conforming_strings property is set to off.
   b) Edit the PostgreSQL data/pg_hba.conf file as follows:

```
host    oozie         oozie         0.0.0.0/0             md5
```

4. Reload the PostgreSQL Configuration.

```
sudo -u postgres pg_ctl reload -s -D /opt/PostgreSQL/8.4/data
```

Configuring MariaDB for Oozie

You must install MariaDB, create the Oozie database and MariaDB user, and add the MariaDB JDBC driver jar file to Oozie.

Procedure

1. Install and Start MariaDB.
2. Create the Oozie Database and Oozie MariaDB User.

   For example, using the MariaDB mysql command-line tool:

```
$ mysql -u root -p
Enter password:

MariaDB [(none)]> create database oozie default character set utf8;
Query OK, 1 row affected (0.00 sec)
MariaDB [(none)]>  grant all privileges on oozie.* to 'oozie'@'localhost'
 identified by 'oozie';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]>  grant all privileges on oozie.* to 'oozie'@'%' identi
fied by 'oozie';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> exit
Bye
```

3. Add the MariaDB JDBC Driver JAR to Oozie.

   Cloudera recommends that you use the MySQL JDBC driver for MariaDB. Copy or symbolically link the MySQL JDBC driver JAR to the /var/lib/oozie/ directory.

   Note: You must manually download the MySQL JDBC driver JAR file.

Configuring MySQL 5 for Oozie

You must install MySQL 5, create the Oozie database and MySQL user, and add the MySQL JDBC driver jar file to Oozie.

Procedure

1. Install and Start MySQL 5.

2. Create the Oozie Database and Oozie MySQL User.

   For example, using the MySQL mysql command-line tool:

```
$ mysql -u root -p
Enter password:

mysql> create database oozie default character set utf8;
Query OK, 1 row affected (0.00 sec)

mysql>  grant all privileges on oozie.* to 'oozie'@'localhost' identified
by 'oozie';
Query OK, 0 rows affected (0.00 sec)

mysql>  grant all privileges on oozie.* to 'oozie'@'%' identified by 'oozi
e';
Query OK, 0 rows affected (0.00 sec)

mysql> exit
```

3. Add the MySQL JDBC Driver JAR to Oozie.

   Copy or symbolically link the MySQL JDBC driver JAR into one of the following directories:

   - For installations that use packages: /var/lib/oozie/
   - For installations that use parcels: /opt/cloudera/parcels/CDH/lib/oozie/lib/

   Note: You must manually download the MySQL JDBC driver JAR file.

Configuring MySQL 8 for Oozie

You must install MySQL 8, create the Oozie database and MySQL user, and add the MySQL JDBC driver jar file to Oozie.

Procedure

1. Install and Start MySQL 8.
2. Create the Oozie Database and Oozie MySQL User.

   For example, using the MySQL mysql command-line tool:

```
$ mysql -u root -p
 Enter password:
mysql> create database oozie default character set utf8;
 Query OK, 1 row affected (0.00 sec)
mysql> CREATE USER 'oozie'@'localhost' IDENTIFIED BY 'oozie';
 Query OK, 0 rows affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON oozie.* TO 'oozie'@'localhost';
 Query OK, 0 rows affected (0.00 sec)
mysql> CREATE USER 'oozie'@'%' IDENTIFIED BY 'oozie';
 Query OK, 0 rows affected (0.01 sec)
mysql> GRANT ALL PRIVILEGES ON oozie.* TO 'oozie'@'%';
 Query OK, 0 rows affected (0.00 sec)
mysql> exit
```

3. Add the MySQL JDBC Driver JAR to Oozie.

   Copy or symbolically link the MySQL JDBC driver JAR into one of the following directories:

   - For installations that use packages: /var/lib/oozie/
   - For installations that use parcels: /opt/cloudera/parcels/CDH/lib/oozie/lib/

   Note: You must manually download the MySQL JDBC driver JAR file.

Configuring Oracle for Oozie

You must install Oracle, create the Oozie Oracle user and grant privileges, and add the Oracle JDBC driver jar file to Oozie.

Procedure

1. Install and Start Oracle.

   Use Oracle's instructions.

2. Create the Oozie Oracle User and Grant Privileges.

   The following example uses the Oracle sqlplus command-line tool, and shows the privileges Cloudera recommends. Oozie needs CREATE SESSION to start and manage workflows. The additional roles are needed for creating and upgrading the Oozie database.

```
sqlplus system@localhost/<SERVICE_NAME>

SQL> create user <user> identified by <password> default tablespace <ta
blespace> temporary tablespace temp;
    User created.
SQL> grant create sequence to <user>;
    Grant succeeded.
SQL> grant create session to <user>;
    Grant succeeded.
SQL> grant create table to <user>;
    Grant succeeded.
SQL> alter user <user> quota unlimited on <tablespace>;
    User altered.
SQL> exit
```

   Important:

   For security reasons, do not make the following grant:

```
grant select any table to oozie;
```

3. Add the Oracle JDBC Driver JAR to Oozie.

   Copy or symbolically link the Oracle JDBC driver JAR into the /var/lib/oozie/ directory.

   Note: You must manually download the Oracle JDBC driver JAR file.

Working with the Oozie server

You can use the command-line interface to start or stop the Oozie server. In addition, you can access the Oozie server with the Oozie client or with a browser.

Starting the Oozie server

You can use the service oozie start command to start Oozie.

Before you begin

Ensure that you have performed *all* the required configuration steps.

Procedure

- Use the service oozie start command to start Oozie.

  If you see the message Oozie System ID [oozie-oozie] started in the oozie.log log file, the system has started successfully.

  Note: By default, Oozie server runs on port 11000 and its URL is http://<OOZIE_HOSTNAME>:11000/oozie. If SSL is enabled, then Oozie server runs on port 11443 by default.

Stopping the Oozie server

Use the sudo service oozie stop command to stop a running Oozie server.

Accessing the Oozie server with the Oozie Client

The Oozie client is a command-line utility that interacts with the Oozie server using the Oozie web-services API.

Procedure

- Use the /usr/bin/oozie script to run the Oozie client.

  For example, if you want to invoke the client on the same machine where the Oozie server is running:

  ```
  $ oozie admin -oozie https://<oozie_server>:11443/oozie -status
  System mode: NORMAL
  ```

- To make it convenient to use this utility, set the environment variable OOZIE_URL to point to the URL of the Oozie server. Then you can skip the -oozie option.

  For example, if you want to invoke the client on the same machine where the Oozie server is running, set the OOZIE_URL to https://<oozie_server>:11443/oozie.

  ```
  $ export OOZIE_URL=https://<oozie_server>:11443/oozie
  $ oozie admin -version
  Oozie server build version: 4.0.0-cdh5.0.0
  ```

  Important: If Oozie is configured with Kerberos Security enabled:

  - You must have a Kerberos session running. For example, you can start a session by running the kinit command.
  - Do not use localhost.

  As with every service that uses Kerberos, Oozie has a Kerberos principal in the form <SERVICE>/<HOSTNAME>@<REALM>. In a Kerberos configuration, you must use the <HOSTNAME> value in the Kerberos principal to specify the Oozie server; for example, if the <HOSTNAME> in the principal is myoozieserver.mydomain.com, set OOZIE_URL as follows:

  ```
  export OOZIE_URL=https://myoozieserver.mydomain.com:11443/oozie
  ```

  If you use an alternate hostname or the IP address of the service, Oozie will not work properly.

- If you want to access Oozie client through Knox:

  ```
  export OOZIE_URL=https://<knox_host>:<knox_port>/gateway/cdp-proxy-api/o
  ozie
  ```

  When you access Oozie client through Knox, you need to specify a username and password in the command line as Knox needs it:

  ```
  export OOZIE_URL=https://<knox_host>:<knox_port>/gateway/cdp-proxy-api/o
  ozie
  oozie admin -version -auth BASIC -username <username> -password <password>
  ```

- When the Oozie server has SSL enabled, the Oozie client does not automatically set the necessary trust-store properties to form a connection. You can set these properties using the following methods:
  - Add them as system properties immediately after the oozie command. For instance:

    ```
    oozie \
        "-Djavax.net.ssl.trustStore={trustStorePath}" \
        "-Djavax.net.ssl.trustStorePassword={trustStorePassword}" \
        "-Djavax.net.ssl.trustStoreType={trustStoreType}" \
        {oozieCommand} \
        -oozie "{oozieUrl}" \
        ...
    ```

  - You can also set these properties by defining the OOZIE_CLIENT_OPTS environment variable before running the Oozie command. For instance:

    ```
    export OOZIE_CLIENT_OPTS="-Djavax.net.ssl.trustStore={trustStorePath} -D
    javax.net.ssl.trustStorePassword={trustStorePassword} -Djavax.net.ssl.tr
    ustStoreType={trustStoreType}"
    ```

  - If you prefer, you can also utilize the -insecure argument with the Oozie command line to prevent the client from validating the certificates:

    ```
    oozie \
        {oozieCommand} \
        -oozie "{oozieUrl}" \
        -insecure \
        ...
    ```

Accessing the Oozie server with a browser

If you have enabled the Oozie web console by adding the ExtJS library, you can connect to the console at http://<OOZIE_HOSTNAME>:11000/oozie.

Note: If the Oozie server is configured to use Kerberos HTTP SPNEGO Authentication, you must use a web browser that supports Kerberos HTTP SPNEGO (for example, Firefox or Internet Explorer).

For information on how to enable the Oozie web console on managed clusters by adding the ExtJS library, see *Enabling the Oozie web console on managed clusters*.

Related Information

Enabling the Oozie web console on managed clusters

Adding schema to Oozie using Cloudera Manager

Cloudera Manager automatically configures Oozie with all available official schemas, and corresponding tables. You can manually add a schema (official or custom) with Cloudera Manager.

Procedure

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Click the Configuration tab.
3. Select Scope Oozie Server .
4. Select Category Advanced .
5. Locate the Oozie SchemaService Workflow Extension Schemas property or search for it by typing its name in the Search box.
6. Enter the desired schema from the following schema list appending .xsd to each entry.

   To apply this configuration property to other role groups as needed, edit the value for the appropriate role group.
7. Enter a Reason for change, and then click Save Change to commit the changes.
8. Restart the Oozie service.

Table 2: Oozie schema

Schema	Cloudera
distcp	distcp-action-0.1 distcp-action-0.2 distcp-action-1.0
email	email-action-0.1 email-action-0.2
git	git-action-1.0
hive	hive-action-0.2 hive-action-0.3 hive-action-0.4 hive-action-0.5 hive-action-0.6 hive-action-1.0
HiveServer2	hive2-action-0.1 hive2-action-0.2 hive2-action-1.0
oozie-bundle	oozie-bundle-0.1 oozie-bundle-0.2
oozie-coordinator	oozie-coordinator-0.1 oozie-coordinator-0.2 oozie-coordinator-0.3 oozie-coordinator-0.4 oozie-coordinator-0.5

Schema	Cloudera
oozie-sla	oozie-sla-0.1 oozie-sla-0.2
oozie-common	oozie-common-1.0
oozie-workflow	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 oozie-workflow-0.5 oozie-workflow-1.0
shell	shell-action-0.1 shell-action-0.2 shell-action-0.3 shell-action-1.0
spark	spark-action-0.1 spark-action-0.2 spark-action-1.0
sqoop	sqoop-action-0.2 sqoop-action-0.3 sqoop-action-0.4 sqoop-action-1.0
ssh	ssh-action-0.1 ssh-action-0.2

Enabling the Oozie web console on managed clusters

You must extract the ext-2.2 libraries to your Oozie server host and enable the Oozie web console.

Procedure

1. Download ext-2.2.
2. Extract the contents of the file to /var/lib/oozie on the same host as the Oozie Server.
   After extraction, the content of the directories is as follows:

```
ls -ltr /var/lib/oozie/
```

```
total 984
drwxr-xr-x 9 oozie   oozie   4096 Aug  4  2008 ext-2.2
-rw-r--r-- 1 systest root  999635 Jan 23 23:24 mysql-connector-java.jar
```

```
ls -ltr /var/lib/oozie/ext-2.2/
```

```
total 1752
```

```
-rw-r--r--  1 oozie oozie    893 Feb 24  2008 INCLUDE_ORDER.txt
drwxr-xr-x 33 oozie oozie   4096 Aug  4  2008 examples
drwxr-xr-x  4 oozie oozie     49 Aug  4  2008 resources
drwxr-xr-x 10 oozie oozie    148 Aug  4  2008 source
drwxr-xr-x 10 oozie oozie    120 Aug  4  2008 build
-rw-r--r--  1 oozie oozie  87524 Aug  4  2008 ext-core.js
-rw-r--r--  1 oozie oozie 163794 Aug  4  2008 ext-core-debug.js
-rw-r--r--  1 oozie oozie 974145 Aug  4  2008 ext-all-debug.js
drwxr-xr-x  6 oozie oozie     55 Aug  4  2008 adapter
-rw-r--r--  1 oozie oozie  11548 Aug  4  2008 CHANGES.html
-rw-r--r--  1 oozie oozie 538956 Aug  4  2008 ext-all.js
-rw-r--r--  1 oozie oozie   1513 Aug  4  2008 license.txt
drwxr-xr-x  4 oozie oozie    108 Aug  4  2008 docs
drwxr-xr-x  5 oozie oozie     94 Jan 24 15:49 air
```

For example:

```
unzip ext-2.2.zip -d /var/lib/oozie
chown -R oozie:oozie /var/lib/oozie/ext-2.2
```

3. In Cloudera Manager Admin Console, go to the Oozie service.
4. Restart the Oozie service.

Enabling Oozie SLA with Cloudera Manager

You can use Oozie to define SLA limits for critical applications and actively monitor these jobs.

Procedure

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Click the Configuration tab.
3. Locate the Enable SLA Integration property or search for it by typing its name in the Search box.
4. Select Enable SLA Integration. This sets the required values for oozie.services.ext and oozie.service.EventHandlerService.event.listeners in oozie-site.xml.
5. Enter a Reason for change, and then click Save Changes to commit the changes.
6. Restart the Oozie service.

What to do next

The following properties are set by default when you enable Oozie SLA in Cloudera Manager. You do not have to explicitly define them, unless you want to modify any of these parameters:

```
oozie.service.SchemaService.wf.schemas
oozie.service.SchemaService.coord.schemas
oozie.service.SchemaService.sla.schemas
oozie.service.ELService.groups
oozie.service.ELService.constants.wf-sla-submit
oozie.service.ELService.ext.constants.coord-sla-create
oozie.service.ELService.functions.coord-sla-create
oozie.service.ELService.constants.coord-sla-submit
oozie.service.ELService.functions.coord-sla-submit
oozie.service.EventHandlerService.filter.app.types
oozie.service.EventHandlerService.event.queue
oozie.service.EventHandlerService.queue.size
oozie.service.EventHandlerService.worker.interval
oozie.service.EventHandlerService.batch.size
oozie.service.EventHandlerService.worker.threads
```

```
    oozie.sla.service.SLAService.alert.events
    oozie.sla.service.SLAService.capacity
    oozie.sla.service.SLAService.calculator.impl
    oozie.sla.service.SLAService.job.event.latency
    oozie.sla.service.SLAService.check.interval
```

For oozie.sla.service.SLAService.alert.events, only END_MISS is configured by default. To change the alert events, explicitly set END_MISS, START_MISS, or DURATION_MISS, in Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml.

Disabling Oozie UI using Cloudera Manager

From the Cloudera Runtime 7.1.7 SP1 release onwards, the Oozie UI is enabled by default. You can disable it by setting a new property in the Oozie site configuration.

Procedure

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Click the Configuration tab.
3. Locate the Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml property or search for it by typing its name in the Search box.
4. Add the following property:

   ```
   Name: oozie.ui.enabled
   Value: false
   ```

5. Enter a Reason for change, and then click Save Changes to commit the changes.
6. Restart the Oozie service.

   To enable the Oozie UI again, delete the oozie.ui.enabled property set using the safety valve.

Moving the Oozie service to a different host

To move the Oozie service to a new host, do the following:

Procedure

1. Stop the current Oozie service.
2. Add the Oozie service to the desired new host using Add Service wizard in Cloudera Manager. The wizard configures and starts Oozie and its dependent services.
3. Add the role specific configurations for the previous host to the new host, if any.
4. Restart the Oozie service.

Oozie database configurations

You can use Cloudera Manager to configure data purge settings, loading, and dumping the Oozie database. Depending on the database that you are using with Oozie, you can set the timezone for the database.

Configuring Oozie data purge settings using Cloudera Manager

You can change your Oozie configuration to control when data is purged to improve performance, reduce database disk usage, or keep the history for a longer period of time. Limiting the size of the Oozie database can also improve performance during upgrades.

About this task

All Oozie workflows older than 30 days are purged from the database by default. However, actions associated with long-running coordinators do not purge until the coordinators complete. If, for example, you schedule a coordinator to run for a year, all those actions remain in the database for the year.

Procedure

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Click the Configuration tab.
3. Type purge in the Search box.
4. Set the following properties as required for your environment:
   - Enable Purge for Long-Running Coordinator Jobs

     Select this property to enable purging of long-running coordinator jobs for which the workflow jobs are older than the value you set for the Days to Keep Completed Workflow Jobs property.
   - Days to Keep Completed Workflow Jobs
   - Days to Keep Completed Coordinator Jobs
   - Days to Keep Completed Bundle Jobs
5. Enter a Reason for change, and then click Save Changes to commit the changes.
6. Select Actions Restart to restart the Oozie Service.

Loading the Oozie database

You must configure the database in which to load your Oozie data, create the required database tables, and then load the Oozie database.

Procedure

1. Stop the Oozie server (in HA mode, stop all Oozie servers).
2. Install and configure the empty database in which to load your Oozie data.

   The db.version of the database must match the db.version of the dump file.
3. Select Actions Create Oozie Database Tables .

   Confirm you want to create the database tables by clicking Create Oozie Database Tables.
4. Verify *DATABASE DUMP FILE* is set correctly.
   a) In the Cloudera Manager Admin Console, click the Oozie service.
   b) Go to the Configuration page.
   c) Select Scope Oozie Server .
   d) Select Category Database .
5. Select Actions Load Database .

   Confirm you want to dump the database to the specified location by clicking Load Database.
6. Select Actions Start .

   Confirm you want to start the service by clicking Start.

Dumping the Oozie database

You must stop the Oozie server, specify the location to which you want to dump the Oozie database, and then perform the dumping operation.

Procedure

1. Stop the Oozie server (in HA mode, stop all Oozie servers).
2. In the Cloudera Manager Admin Console, go to the Oozie service status page.
3. Select Actions Stop .

   Confirm you want to stop the service by clicking Stop.
4. Specify *DATABASE DUMP FILE*.
   a) Go to the Configuration page.
   b) Select Scope Oozie Server .
   c) Select Category Database .
   d) Set a file location for the Database Dump File.
5. Select Actions Dump Database .

   Confirm that you want to dump the database to the specified location by clicking Dump Database.

   During the export process, Cloudera Manager fetches and writes the database content a compressed zip specified by the *DATABASE DUMP FILE* property.

Setting the Oozie database timezone

Depending on the type of database you are using with Oozie, you must configure specific properties for setting the database timezone.

Cloudera recommends that you set the timezone in the Oozie database to GMT. Databases do not handle Daylight Saving Time (DST) shifts correctly. There might be problems if you run any Coordinators with actions scheduled to materialize during the one-hour period that gets lost in DST.

Important: Changing the timezone on an existing Oozie database while Coordinators are already running might cause Coordinators to shift by the offset of their timezone from GMT one time after you make this change.

For more information about how to set your database's timezone, see your database's documentation.

Fine-tuning Oozie's database connection

Learn how you can configure Oozie to use its database well.

About this task

When it comes to configuring database connections, simply providing a hostname, port, username, and password may not be sufficient. In order to optimize Oozie's database connection, you might need to manually construct lengthy connection and configuration strings using safety-valve settings. To simplify this process and enable finer control over Oozie's database connection, you can use several enhancements, as described in this section.

The following properties allow you to configure how Oozie uses its database:

- oozie_database_connection_properties

  You can use this property to directly configure the database connection. For example, if you need to pass a trustStore path to the connection, you can add a property named javax.net.ssl.trustStore and with a value to your trustStore file.

- oozie_datasource_properties

  You can use this property to configure the datasource object created by OpenJPA. For example, if you would like to finetune how many idle connections the datasource instance should keep, you can set the maxIdle property.

  Note: These configurations do not configure the underlying database connection, but the commons-dbcp datasource instance. For more information, see BasicDataSource Configuration Parameters.

With the help of the values set for both the properties, Cloudera Manager assembles one single configuration which will be set in oozie-site.xml. The name of this single configuration is, oozie.service.JPAService.connection.properties, and it is assembled in the following way:

- Whatever you specify in oozie_database_connection_properties, is concatenated with a semicolon, and available under the ConnectionProperties property in the final configuration.
- Whatever you specify in oozie_datasource_properties, is directly set in the final configuration.

For example, if you set the following properties in oozie_database_connection_properties, because your database JDBC driver accepts them,

- javax.net.ssl.trustStore=/path/to/trustStore.jks
- ssl.enabled=true

And you set the following properties in oozie_datasource_properties:

- maxIdle=10
- defaultQueryTimeout=120

Then the final value of oozie.service.JPAService.connection.properties will be:

```
ConnectionProperties="javax.net.ssl.trustStore=/path/to/trustStore.jks;ssl.e
nabled=true", maxIdle=10, defaultQueryTimeout=120
```

Important: Kindly be aware that the properties mentioned above (such as javax.net.ssl.trustStore, ssl.enabled, and so on) are provided merely as an example. The specific properties that need configuration should align with the JDBC driver you use to connect to the underlying Oozie database.

Perform the following steps after you configure the correct properties:

Procedure

1. In Cloudera Manager, click the Oozie service.
2. Click the Configuration tab.
3. Search for and set the oozie_database_connection_properties property.

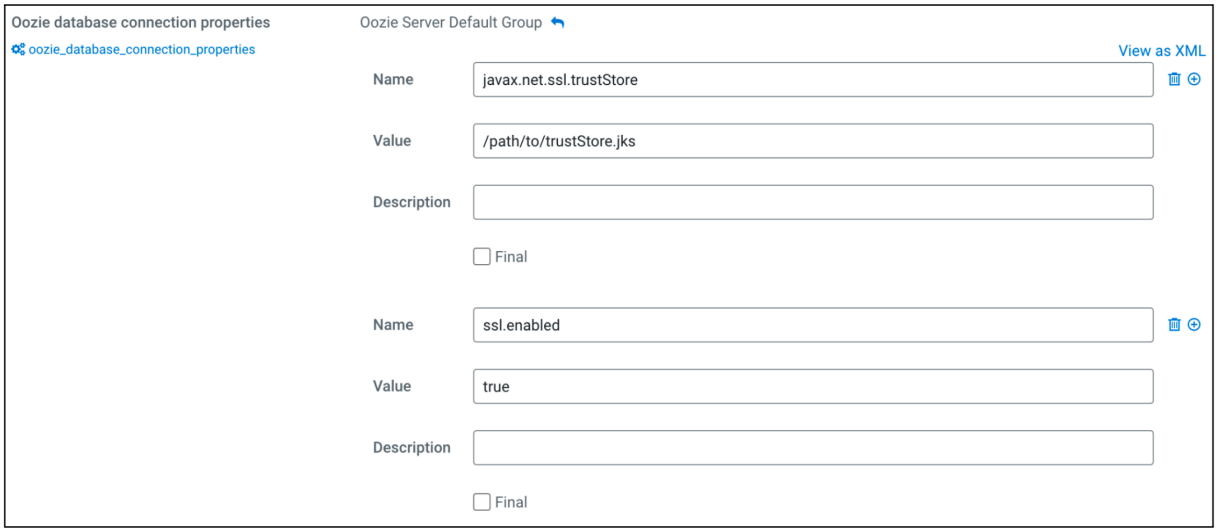

4. Search for and set the oozie_datasource_properties property.

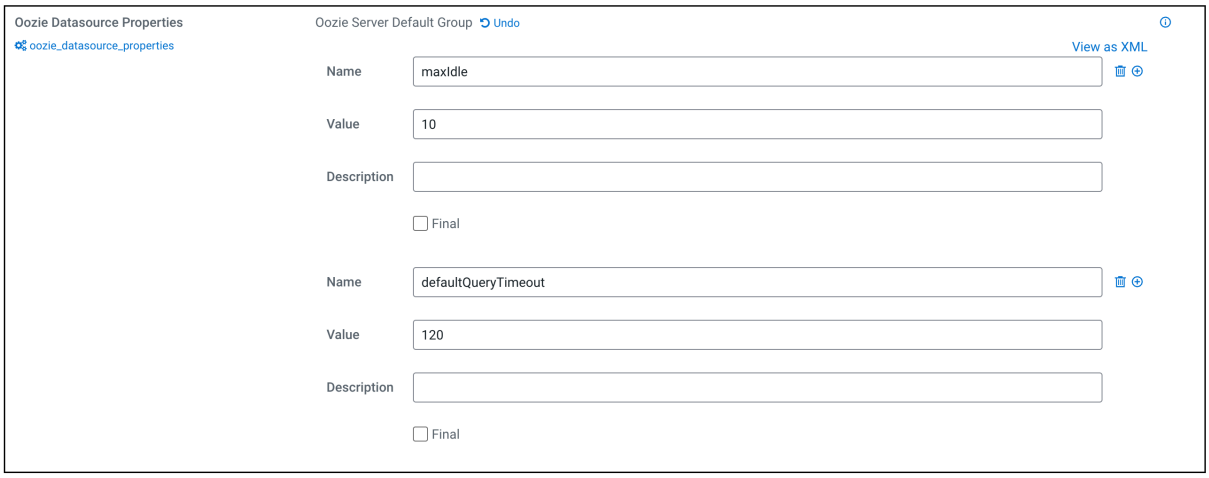

5. Click Save Changes, and allow Cloudera Manager some time to recognize the changes.
6. Follow the instructions provided by Cloudera Manager to redeploy Oozie.

Assembling a secure JDBC URL for Oozie

Learn how to assemble a secure JDBC URL for Oozie.

Procedure

1. In Cloudera Manager, click the Oozie service.
2. Click the Configuration tab.
3. Search and enable the oozie_database_is_secure property.

   If you enable this property, then Cloudera Manager assembles a secure JDBC URL for Oozie based on the selected database types.
4. Click Save Changes, and allow Cloudera Manager some time to recognize the changes.
5. Follow the instructions provided by Cloudera Manager to redeploy Oozie.

Oracle TCPS

Learn how to establish a connection to a secure Oracle database.

If the oozie_database_is_secure property is enabled, then the JDBC URL generated for Oracle looks like:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<HOST>)(PORT=<P
ORT>))(CONNECT_DATA=(SERVICE_NAME=<DB_NAME>)))
```

The values of <HOST>, <PORT>, and <DB_NAME> come from the following Oozie properties set in Cloudera Manager:

- <HOST> and <PORT> – oozie_database_host
- <DB_NAME> – oozie_database_name

While the above specifies the protocol as TCPS, this JDBC URL on its own might not be enough to establish a connection to a secure Oracle database. If you need to specify a trustStore, keyStore, or Oracle wallet properties for the connection, use the oozie_database_connection_properties and/or oozie_datasource_properties properties.

For example,

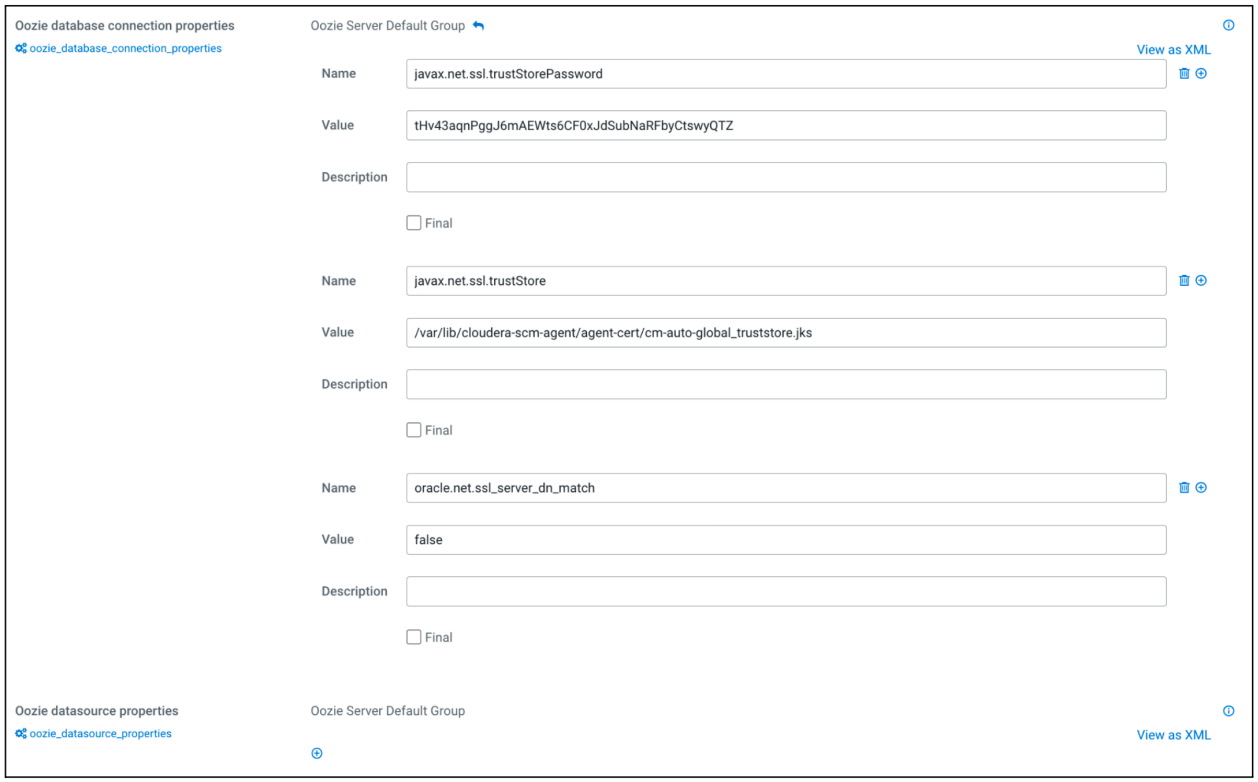

OpenJPA upgrade

Explore the updates introduced in Apache Oozie following the upgrade of OpenJPA from version 2.x to 3.x. This upgrade includes new configuration properties, deprecated configuration properties, and enhanced error handling.

Following third-party libraries were upgraded in the Apache Oozie:

Third-party library name	Upgraded from	Upgraded to
OpenJPA	2.4.2	3.2 and later
commons-dbcp	2.4.2	2.12.0 and later
commons-pool	1.6	2.12.0 and later

For more information about the exact versions that the libraries were upgraded to, see Cloudera Runtime parcel's Oozie directory that is, {CDH_PARCELS_DIR}/lib/oozie/lib.

Upgrading these third-party libraries ensures that Oozie works with the latest versions of these libraries, allowing you to have a more stable and secure experience. Also, new configurations are introduced in Oozie to support this upgrade.

Changes in existing Oozie configuration

The default value of the Oozie configuration oozie.service.JPAService.connection.data.source is updated from org.apache.oozie.util.db.BasicDataSourceWrapper to org.apache.commons.dbcp2.BasicDataSource. This modification was made in response to a bug found in the earlier version of commons-dbcp, which led to the implementation of the BasicDataSourceWrapper in Oozie. With the latest commons-dbcp update resolving the underlying issue, the workaround has now been removed from Oozie.

New Oozie configurations

Following new Apache Oozie configurations are introduced to support the OpenJPA upgrade. The following properties can be overridden in the Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml.

oozie.service.JPAService.pool.total.conn

Default value: 10. The default value of this property is the same as oozie.service.JPAService.pool.max.active.conn.

Description: This property enables you to configure the connection pool's capacity to limit the number of open connections. If the number of connections in the pool is equal to this value, then new queries must wait for an existing connection to be released. If the capacity is set to a negative value (unlimited), then the new connections are created as needed, regardless of the number of currently open connections. This property replaces the oozie.service.JPAService.pool.max.active.conn property.

oozie.service.JPAService.pool.min.idle.conn

Default value: 2

Description: This property enables you to set the minimum number of idle connections to be maintained in the pool. If the number of idle connections is less than this value, then new connections are created. If this property is set to zero, then no idle connections are maintained, and new connections are only created as needed. The combined number of idle and active connections must not be more than the total number of allowed connections.

oozie.service.JPAService.validate.db.query

Default value: SELECT COUNT(*) FROM VALIDATE_CONN

Description: SQL query to validate DB connection. This property is ignored if oozie.service.JPAService.validate.db.connection is set to false.

oozie.service.JPAService.custom.<*OPENJPA.CONFIGURATION***>**

Description: This property enables you to specify any OpenJPA property. For more information about OpenJPA properties, see OpenJPA documentation. Properties set in Oozie's persistence.xml can also be overridden using this configuration. This property can be repeated, and to set multiple openjpa.* properties, you must define an oozie.service.JPAService.custom.<***OPENJPA.CONFIGURATION***> configuration for each one. These properties are not available by default, but you can create them on-demand if you wish to customize additional OpenJPA configurations.

Following example steps provide more information about how you can override any OpenJPA properties:

You can configure multiple openjpa.* properties in Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml as displayed in the following screenshot.

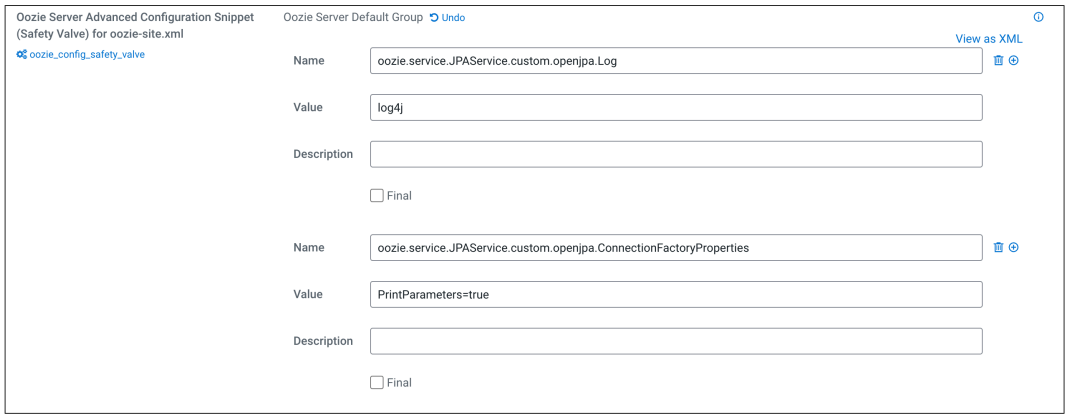

Example on how to override the transaction isolation in Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml:

```
oozie.service.JPAService.custom.openjpa.jdbc.TransactionIsolatio
n=serializable
```

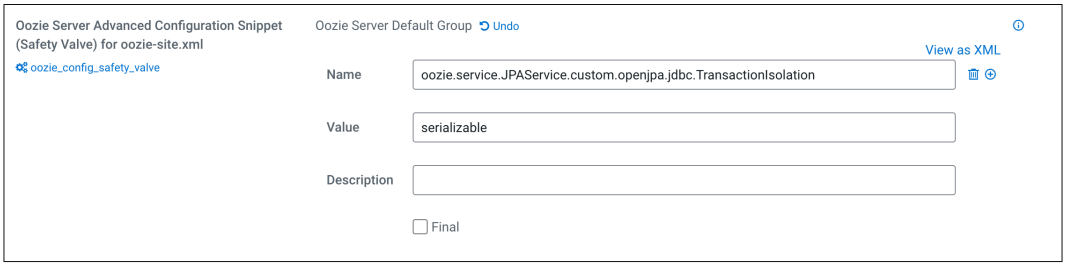

To enable OpenJPA debug logging through log4j, which logs into the Oozie server log files, set the following properties in Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml:

```
oozie.service.JPAService.custom.openjpa.Log=log4j
oozie.service.JPAService.custom.openjpa.ConnectionFactoryPrope
rties=PrintParameters=true
```

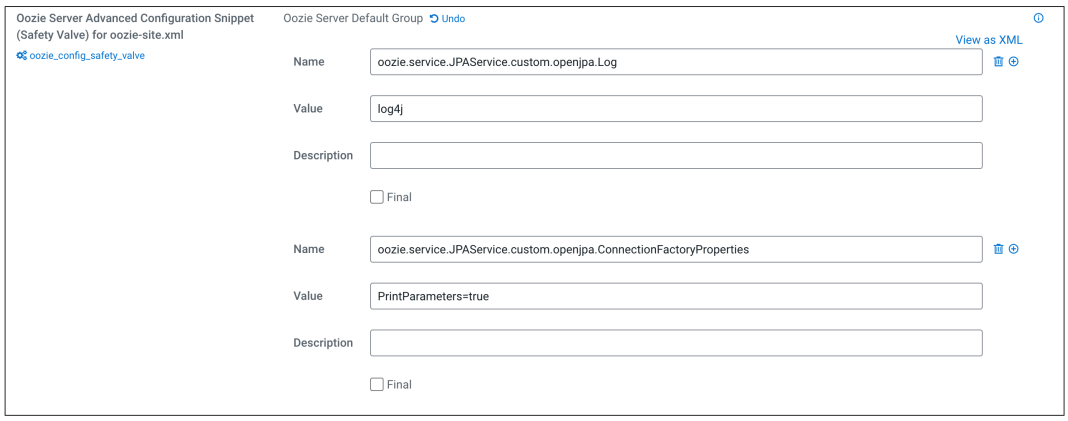

Add the following log4j properties in Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml:

```
log4j.category.openjpa.Tool=TRACE
log4j.category.openjpa.Query=TRACE
log4j.category.openjpa.jdbc.SQL=TRACE
log4j.category.openjpa.Runtime=INFO
log4j.category.openjpa.Remote=WARN
log4j.category.openjpa.DataCache=WARN
log4j.category.openjpa.MetaData=WARN
log4j.category.openjpa.Enhance=WARN
log4j.category.openjpa.jdbc.SQLDiag=WARN
log4j.category.openjpa.jdbc.JDBC=WARN
log4j.category.openjpa.jdbc.Schema=WARN
```

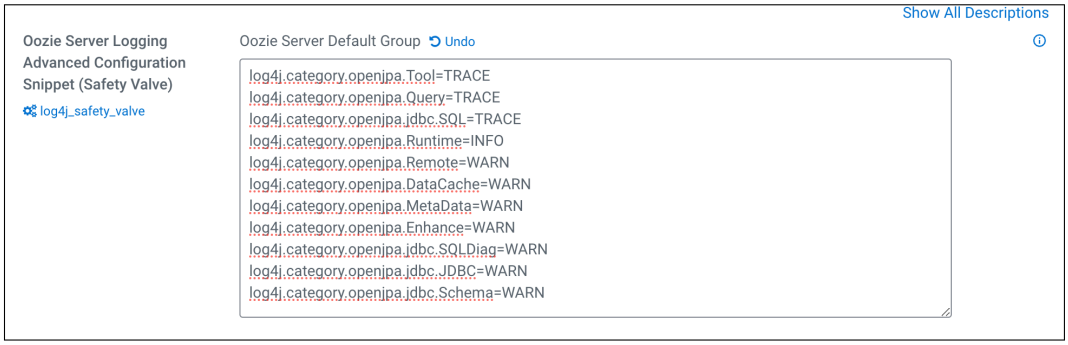

Enhanced database connectivity issue handling

When Oozie is configured to manage the database connectivity issues and if the same operation fails multiple times, then Oozie executes a validation query to check if the database connection is still open. Oozie does not force close the connection if the current retry count is less than half of the maximum retries. For example, if the maximum retries limit is set to 5, Oozie starts validating the connection on the 2nd retry attempt (floor(5/2) = 2). This validation continues until the retries either succeed or reach the maximum limit of 5.

If the connection is found to be dead or closed on the DBMS side, Oozie closes it. Then, OpenJPA automatically requests a new connection from the connection pool. This process ensures that the connection pool remains free of dead connections.

Prerequisites for configuring TLS/SSL for Oozie

There are certain prerequisites that must be fulfilled for configuring TLS/SSL for Oozie.

- Keystores for Oozie must be readable by the oozie user. This can be a copy of the Hadoop services' keystore with permissions set to 0440 and owned by the oozie group.
- Truststores must have permissions set to 0444, which means that all users can read them.
- Specify absolute paths to the keystore and truststore files. These settings apply to all hosts on which daemon roles of the Oozie service run so the paths you choose must be valid on all hosts.
- If there is a DataNode and an Oozie server running on the same host, they can use the same certificate.

Configure TLS/SSL for Oozie

You can edit properties to enable TLS/SSL for Oozie, specify the keystore file location on the local file system, and set the password for the keystore.

Procedure

1. In Cloudera Manager, select the Oozie service.
2. Click the Configuration tab.
3. In the Search field, type TLS/SSL to show the Oozie TLS/SSL properties.
4. Edit the following TLS/SSL properties according to your cluster configuration.

Property	Description
Enable TLS/SSL for Oozie	Check this field to enable TLS/SSL for Oozie.
Oozie TLS/SSL Server JKS Keystore File Location	Path to the keystore file on the local file system.

Property	Description
Oozie TLS/SSL Server JKS Keystore File Password	Password for the keystore file.
Oozie TLS/SSL Client Trust Store File	Path to the client truststore file.
Oozie TLS/SSL Client Trust Store Password	Password for the truststore file.

5. If SSL is enabled for ZooKeeper, edit the following SSL properties:

Property	Description
Oozie ZooKeeper TLS/SSL Server JKS Keystore File Location	Path to the keystore file.
Oozie ZooKeeper TLS/SSL Server JKS Keystore File Password	Password for the keystore file.
Oozie ZooKeeper TLS/SSL Client Trust Store File	Path to the client truststore file.
Oozie ZooKeeper TLS/SSL Client Trust Store Password	Password for the truststore file.

6. Optionally, you can modify the values of the following properties:
   - Enabled TLS Protocols - List of Cipher Suite names that should be excluded.
   - Excluded Cipher Suites - TLS protocols accepted by the Oozie Server.
7. Click Save Changes.
8. Restart the Oozie service.

Oozie Java-based actions with Java 17

In relation to Java 17, certain applications might require reflective access to internal Java classes, packages, or modules. This section describes how to gain reflective access to these with Oozie.

To enable reflective access, use the --add-opens Java parameter. If your applications executed with Oozie need reflective access, you can use the following properties to specify the required --add-opens arguments:

- oozie.launcher.yarn.app.mapreduce.am.command-opts
- oozie.launcher.mapreduce.map.java.opts
- oozie.launcher.mapred.child.java.opts

For example, in your workflow's configuration, you can set the property as follows:

```
<property>
    <name>oozie.launcher.yarn.app.mapreduce.am.command-opts</name>
    <value>--add-opens=java.base/java.lang=ALL-UNNAMED</value>
</property>
```

The Spark and Spark 3 applications inherently require reflective access. Consequently, Oozie automatically adds the following --add-opens properties by default for Spark and Spark 3 actions:

```
--add-opens=java.base/java.io=ALL-UNNAMED
--add-opens=java.base/java.lang.invoke=ALL-UNNAMED
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.base/java.net=ALL-UNNAMED
--add-opens=java.base/java.nio=ALL-UNNAMED
--add-opens=java.base/java.util.concurrent=ALL-UNNAMED
--add-opens=java.base/java.util=ALL-UNNAMED
--add-opens=java.base/sun.nio.ch=ALL-UNNAMED
--add-opens=java.base/sun.nio.cs=ALL-UNNAMED
--add-opens=java.base/sun.security.action=ALL-UNNAMED
```

It is important to note that this behavior can be overridden for Spark actions by using the oozie.action.spark.launcher.jdk17.opens property, and for Spark3 actions by using the oozie.action.spark3.launcher.jdk17.opens property. By specifying Java modules, packages, or classes in a comma-separated list with these properties, Oozie can configure the appropriate access permissions when launching a Spark or Spark 3 application. Each item in the list should be in a format compatible with the Java --add-opens parameter, such as java.base/java.lang=ALL-UNNAMED.

The following comma-separated list shows the default values of these properties. These are derived from the default values in Spark and Spark 3:

```
java.base/java.io=ALL-UNNAMED,java.base/java.lang.invoke=ALL-UNNAMED,java.base/java.lang=ALL-UNNAMED,java.base/java.net=ALL-UNNAMED,java.base/java.nio=ALL-UNNAMED,java.base/java.util.concurrent=ALL-UNNAMED,java.base/java.util=ALL-UNNAMED,java.base/sun.nio.ch=ALL-UNNAMED,java.base/sun.nio.cs=ALL-UNNAMED,java.base/sun.security.action=ALL-UNNAMED
```

However, you have the flexibility to customize the values of these properties by adding a safety-valve setting for the corresponding property on the Oozie configuration page in Cloudera Manager by using the oozie_config_safety_valve setting.

For other Oozie Java-based actions (simple Java action, Distcp, Git, Map-Reduce, Hive2, Shell, Sqoop), Oozie does not add any --add-opens values by default. Nevertheless, you have the option to specify custom --add-opens overrides for each action using the corresponding properties:

- oozie.action.distcp.launcher.jdk17.opens
- oozie.action.git.launcher.jdk17.opens
- oozie.action.hive2.launcher.jdk17.opens
- oozie.action.java.launcher.jdk17.opens
- oozie.action.map-reduce.launcher.jdk17.opens
- oozie.action.shell.launcher.jdk17.opens
- oozie.action.sqoop.launcher.jdk17.opens

These properties can be specified globally through a safety-valve configuration, in the job.properties file, in the workflow's global configuration, or inside the workflow as a specific action's configuration. The order of precedence for these configurations is as follows:

1. If you have configured the property at the action level, it takes precedence over all other settings, and the remaining configurations are disregarded.
2. If you have configured the property in the global configuration of the workflow, the value from there is used.
3. If the setting is not available in either of the previous locations, the value configured in your job.properties file is used instead.
4. Lastly, the global setting in Cloudera Manager comes into effect.

Note: These properties only have an effect on the Launcher application created by Oozie.

Oozie security enhancements

Learn about Oozie security enhancements related to callback, callback endpoint, FIPS compliance, and SMTP (Simple Mail Transfer Protocol). Oozie will be notified about completion of tasks through HTTPS.

Callback

- Prior to this enhancement, even though SSL was enabled for Oozie, the callback mechanism – which notifies the Oozie server after an action finished (success/failed) – was going through HTTP. With the enhanced callback feature if TLS/SSL is enabled for Oozie, the callback invocation goes through HTTPS. This applies to all Oozie actions, including map-reduce actions. For map-reduce actions, as always, the Oozie Application Master (AM)

container does not wait for the map-reduce Job to complete, but YARN makes a callback to Oozie when the map-reduce Job completes. This callback goes through HTTPS as well when TLS/SSL is enabled for Oozie. When TLS/SSL is enabled for Oozie, Oozie listens only on the HTTPS port and not on the HTTP port as the HTTP port was only needed for the callback mechanism. Oozie will `not` explicitly upload the truststore file required for the HTTPS connection to the YARN applications launched by Oozie and neither should you, but Oozie will pass the location of the file used by Oozie itself to the callback mechanism running inside the YARN container. Hence, the truststore file used by Oozie needs to be available on all NodeManager Hosts and accessible by YARN containers.

Note: Until now, the default callback command for SSH actions was `curl`. If you have enabled TLS/SSL for Oozie, Cloudera Manager will change this to `curl -k`. If you have added a custom callback command setup for SSH actions through a safety valve, that setup will not be overridden by Cloudera Manager. You must make sure that your command supports TLS/SSL.

Callback Endpoint

- Along with the callback mechanism, you can also enable authentication for the callback endpoint. If you have Kerberos configured on your cluster, authentication is enabled for all endpoints of Oozie by default except for the callback endpoint. You can enable authentication for the callback endpoint by setting the oozie.servlet.CallbackServlet.authentication.required property to `true` as a safety-valve in Cloudera Manager.

  Note: After the release of Cloudera Manager 7.3.0, you do not need to configure the callback endpoint authentication through a safety-valve because Cloudera is introducing the Oozie Callback Servlet Authentication property. After the release of Cloudera Manager 7.3.0, upgrade Cloudera Manager, and search and select the new Oozie Callback Servlet Authentication option. If you have set the above property using safety-valve, you can remove it and instead enable it through the new checkbox. No new configuration is required in Cloudera Runtime. When callback authentication is enabled, Oozie does not allow an unauthenticated invocation to the endpoint. Before starting the AM container, Oozie generates a new type of delegation token and when the Job finishes and the AM container notifies the Oozie server. This new Oozie delegation token is used to make the callback.

  Note: If you enable authentication on the callback endpoint, when you are executing an SSH action, make sure your SSH command will create a Kerberized environment. Otherwise, the callback will fail.

FIPS Compliance

To make Oozie FIPS compliant, the following changes are introduced:

- When TLS/SSL is enabled for Oozie, apart from setting the trustStore, trustStorePassword, keyStore, and keyStorePassword properties, Cloudera Manager adds two new properties `oozie.https.truststore.type` and `oozie.https.keystore.type` in the oozie-site.xml file. These properties will contain the value of the globally configured keyStore type in Cloudera Manager.
- When TLS/SSL is enabled for ZooKeeper and Oozie runs with High-Availability, Cloudera Manager sets the `oozie.zookeeper.https.truststore.type` and `oozie.zookeeper.https.keystore.type` properties along with the existing `oozie.zookeeper.https.truststore/keystore.file/password` property in the oozie-site.xml file.

SMTP

To configure custom TLS/SSL protocols when executing an email action, add the new `oozie.email.smtp.ssl.protocols` property using a safety valve in Cloudera Manager.

Related Information

Configure TLS/SSL for Oozie

Additional considerations when configuring TLS/SSL for Oozie HA

To enable clients to connect to Oozie servers (the target servers) through the load balancer using TLS/SSL, configure the load balancer for TLS/SSL pass-through.

This means that the load balancer does not perform encryption or decryption but instead passes traffic from clients and servers to the appropriate target host. See the documentation for your load balancer for details.

Configure Oozie client when TLS/SSL is enabled

You must configure the Oozie client if TLS/SSL is enabled in your cluster. You can configure the Oozie command line client using either the JDK certificate store or using the trust-store file.

Procedure

Using JDK Certificate Store

- Import the certificate into the JDK certificate store. For example,

```
keytool -keystore </usr/java/default/lib/security/cacerts> -import -trus
tcacerts -alias autotls -file </opt/cloudera/CMCA/trust-store/cm-auto-gl
obal_cacerts.pem>  --storepass changeit -noprompt
```

  You must specify the JDK/JRE certificate file location with the -keystore parameter and the certificate you want to import with the -file parameter.

Using Trust Store

- Manually specify the trust-store and trust-store password for the Oozie command line client. For example,

```
oozie -Djavax.net.ssl.trustStore={trustStoreFile} -Djavax.net.ssl.trustS
torePassword={trustStorePassword} jobs -oozie https://{oozieHost}:{oozie
Port}/oozie
```

Using insecure SSL connnection

- From the Cloudera Runtime 7.1.7 SP1 release onwards, you can manually set the SSL connection to insecure. For example,

```
oozie jobs -oozie https://{oozieHost}:{ooziePort}/oozie -insecure
```

  This causes Oozie to allow certificate errors while the data remains encrypted. With this, there is no need to import the certificate into the JDK certificate store or specify the trust-store and trust-store password manually for the Oozie command line client.

Configuring custom Kerberos principal for Oozie

The Kerberos principal for Ozone is configured by default to use the same service principal as the default process user. However, you can change the default setting by providing a custom principal in Cloudera Manager.

Procedure

1. In Cloudera Manager, click Clusters > Oozie .
2. Go to the Configuration tab

3. Search for the Kerberos Principal by entering "kerberos" in the search field.
4. For Kerberos Principal, enter your custom principal value.
5. Click Save Changes.
6. Click Actions and select Restart to restart the service.

Cloudera Runtime 7.3.1

How to manage log rotation for Ranger services

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Managing logging properties for Ranger services..... 4
How to manage logging properties for Ranger Admin..... 4
How to manage logging properties for Ranger Usersync.....4
How to manage logging properties for Ranger Tagsync.....5
How to manage logging properties for Ranger KMS.....6
How to manage logging properties for Ranger RMS.....6
How to manage logging properties for Ranger RAZ..... 7

Enabling selective debugging for Ranger Admin.....8

Enabling selective debugging for RAZ.....8

Managing logging properties for Ranger services

How to configure properties that control log file rotation in each Ranger service.

All the Ranger services write out separate log files on the node where they are installed and started. Cluster administrators can monitor and manage the disk space used by Ranger services roles log file and configure log rotation to prevent log files from consuming too much disk space.

How to manage logging properties for Ranger Admin

How to configure properties that control log file rotation in Ranger Admin.

About this task

Ranger Admin has log files stored on the node where it is installed and started from Cloudera Manager.

The log files are:

- ranger-admin-hostname.example.com.log
- ranger_admin_sql_db_patch.log
- ranger_db_patch.log

Use the following properties to configure logging behavior:

Table 1: Ranger Admin Logging Properties

Property	Description	Default Value
Ranger Admin Log Directory	The path to the directory where the log files are saved.	/var/log/ranger/admin
Ranger Admin Logging Threshold	Logging level to limit the number of entries saved in the log file.	INFO
Ranger Admin Max Log Size	Maximum size for a log file before the log file rolls over into a new file.	200 MB
Ranger Admin Maximum Log File Backups	Maximum number of rolled-over log files to retain.	10

To manage log file configurations for Ranger Admin role instance:

Procedure

1. Go to Ranger Configuration .
2. Select the Category Logs filter.
3. Edit the values of any of the logging properties.
4. Click Save Changes.
5. Restart Ranger Admin.

How to manage logging properties for Ranger Usersync

How to configure properties that control log file rotation for Ranger Usersync.

About this task

Ranger Usersync has log files stored on the node where it is installed and started from Cloudera Manager.

The log files are:

- ranger-usersync-hostname.example.com.log

Use the following properties to configure logging behavior:

Table 2: Ranger Usersync Logging Properties

Property	Description	Default Value
Ranger Usersync Log Directory	The path to the directory where the log files are saved.	/var/log/ranger/usersync
Ranger Usersync Logging Threshold	Logging level to limit the number of entries saved in the log file.	INFO
Ranger Usersync Max Log Size	Maximum size for a log file before the log file rolls over into a new file.	200 MB
Ranger Usersync Maximum Log File Backups	Maximum number of rolled-over log files to retain.	10

To manage log file configurations for Ranger Usersync role instance:

Procedure

1. Go to Ranger Configuration .
2. Select the Category Logs filter.
3. Edit the values of any of the logging properties.
4. Click Save Changes.
5. Restart Ranger Usersync.

How to manage logging properties for Ranger Tagsync

How to configure properties that control log file rotation for Ranger Tagsync.

About this task

Ranger Tagsync has log files stored on the node where it is installed and started from Cloudera Manager.

The log files are:

- tagsync.log

Use the following properties to configure logging behavior:

Table 3: Ranger Tagsync Logging Properties

Property	Description	Default Value
Ranger Tagsync Log Directory	The path to the directory where the log files are saved.	/var/log/ranger/tagsync
Ranger Tagsync Logging Threshold	Logging level to limit the number of entries saved in the log file.	INFO
Ranger Tagsync Max Log Size	Maximum size for a log file before the log file rolls over into a new file.	200 MB
Ranger Tagsync Maximum Log File Backups	Maximum number of rolled-over log files to retain.	10

To manage log file configurations for Ranger Tagsync role instance:

Procedure

1. Go to Ranger Configuration .
2. Select the Category Logs filter.
3. Edit the values of any of the logging properties.
4. Click Save Changes.
5. Restart Ranger Tagsync.

How to manage logging properties for Ranger KMS

How to configure properties that control log file rotation in Ranger KMS.

About this task

Ranger KMS has log files stored on the node where it is installed and started from Cloudera Manager.

The log files are:

- ranger-kms-hostname.example.com-user.log
- kms-audit.log

Use the following properties to configure logging behavior:

Table 4: Ranger KMS Logging Properties

Property	Description	Default Value
Ranger KMS Server Log Directory	The path to the directory where the log files are saved.	/var/log/ranger/kms
Ranger KMS Server Logging Threshold	Logging level to limit the number of entries saved in the log file.	INFO
Ranger KMS Server Max Log Size	Maximum size for a log file before the log file rolls over into a new file.	200 MB
Ranger KMS Server Maximum Log File Backups	Maximum number of rolled-over log files to retain.	10

To manage log file configurations for Ranger KMS Server role instance:

Procedure

1. Go to Ranger KMS Configuration .
2. Select the Category Logs filter.
3. Edit the values of any of the logging properties.
4. Click Save Changes.
5. Restart Ranger KMS.

How to manage logging properties for Ranger RMS

How to configure properties that control log file rotation in Ranger RMS.

About this task

Ranger RMS has log files stored on the node where it is installed and started from Cloudera Manager.

The log files are:

- ranger-rms-hostname.example.com-user.log
- ranger-rms-perf-hostname.example.com-user.log

Use the following properties to configure logging behavior:

Table 5: Ranger RMS Logging Properties

Property	Description	Default Value
Ranger RMS Server Log Directory	The path to the directory where the log files are saved.	/var/log/ranger/rms
Ranger RMS Server Logging Threshold	Logging level to limit the number of entries saved in the log file.	INFO
Ranger RMS Server Max Log Size	Maximum size for a log file before the log file rolls over into a new file.	200 MB
Ranger RMS Server Maximum Log File Backups	Maximum number of rolled-over log files to retain.	10

To manage log file configurations for Ranger RMS Server role instance:

Procedure

1. Go to Ranger RMS Configuration .
2. Select the Category Logs filter.
3. Edit the values of any of the logging properties.
4. Click Save Changes.
5. Restart Ranger RMS.

How to manage logging properties for Ranger RAZ

How to configure properties that control log file rotation in Ranger RAZ.

About this task

Ranger RAZ has log files stored on the node where it is installed and started from Cloudera Manager.

The log files are:

- ranger-raz-hostname.example.com-user.log
- ranger-raz-metrics.log
- audit.log

Use the following properties to configure logging behavior:

Table 6: Ranger RAZ Logging Properties

Property	Description	Default Value
Ranger RAZ Server Log Directory	The path to the directory where the log files are saved.	/var/log/ranger/raz
Ranger RAZ Server Logging Threshold	Logging level to limit the number of entries saved in the log file.	INFO
Ranger RAZ Server Max Log Size	Maximum size for a log file before the log file rolls over into a new file.	200 MB
Ranger RAZ Server Maximum Log File Backups	Maximum number of rolled-over log files to retain.	10

To manage log file configurations for Ranger RAZ Server role instance:

Procedure

1. Go to Ranger RAZ Configuration .
2. Select the Category Logs filter.
3. Edit the values of any of the logging properties.
4. Click Save Changes.
5. Restart Ranger RAZ.

Enabling selective debugging for Ranger Admin

You can limit the verbose log output for Ranger Admin by enabling (selective) debugging at the package level.

About this task

When you enable the DEBUG level for Ranger Admin, a detailed verbose log is generated. Alternatively, you can enable debugging at a package level, so that only the package logs show detailed verbose logs.

Procedure

1. From Cloudera Manager, go to Ranger Configuration .
2. In Search, type log4j_
3. In Ranger Admin Logging Advanced Configuration Snippet (Safety Valve), add the following:

```
loggers=rangerPerf, rangerPatch, orgSpring, orgHibernate, jdbcSql,
jdbcSqlTiming, jdbcAudit, jdbcResult, jdbcConnection, comMchange,
rangeradmin
logger.rangeradmin.name=<PACKAGE_OR_FULLY_QUALIFIED_CLASS_NAME>
logger.rangeradmin.level=DEBUG
```

Enabling selective debugging for RAZ

You can limit the verbose log output for RAZ by enabling (selective) debugging at the package level.

About this task

When you enable the DEBUG level for RAZ, a detailed verbose log is generated. Alternatively, you can enable debugging at a package level, so that only the package logs show detailed verbose logs.

Procedure

1. From Cloudera Manager, go to Ranger-RAZ Configuration .
2. In Search, type log4j_
3. In Ranger Admin Logging Advanced Configuration Snippet (Safety Valve), add the following:

```
loggers=AUDIT, METRICS, RANGERRAZ
logger.RANGERRAZ.name=<PACKAGE_OR_FULLY_QUALIFIED_CLASS_NAME>
logger.RANGERRAZ.level=DEBUG
logger.RANGERRAZ.additivity=false
logger.RANGERRAZ.appenderRef.RANGERRAZ.ref=DRFA
```

Cloudera Runtime 7.3.1

Configuring Apache Spark

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring dynamic resource allocation..... 4
Customize dynamic resource allocation settings..... 4
Configure a Spark job for dynamic resource allocation..... 4
Dynamic resource allocation properties..... 5

Spark security..... 6
Enabling Spark authentication..... 6
Enabling Spark Encryption..... 6
Running Spark applications on secure clusters..... 7
Configuring HSTS for Spark..... 7

Accessing compressed files in Spark..... 8

Using Spark History Servers with high availability..... 8
Limitation for Spark History Server with high availability..... 8
Configuring high availability for Spark History Server with an external load balancer..... 9
Configuring high availability for Spark History Server with an internal load balancer..... 10
Configuring high availability for Spark History Server with multiple Knox Gateways..... 11

How to access Spark files on Ozone..... 12

Configuring dynamic resource allocation

This section describes how to configure dynamic resource allocation for Apache Spark.

When the dynamic resource allocation feature is enabled, an application's use of executors is dynamically adjusted based on workload. This means that an application can relinquish resources when the resources are no longer needed, and request them later when there is more demand. This feature is particularly useful if multiple applications share resources in your Spark cluster.

Important: Dynamic resource allocation does not work with Spark Streaming.

You can configure dynamic resource allocation at either the cluster or the job level:

- Cluster level: Dynamic resource allocation is enabled by default. The associated shuffle service starts automatically.
- Job level: You can customize dynamic resource allocation settings on a per-job basis. Job settings override cluster configuration settings.

Cluster configuration is the default, unless overridden by job configuration.

The following subsections describe each configuration approach, followed by a list of dynamic resource allocation properties.

Customize dynamic resource allocation settings

Use the following steps to review and customize dynamic resource allocation settings.

About this task

Dynamic resource allocation requires an external shuffle service that runs on each worker node as an auxiliary service of NodeManager. This service is started automatically; no further steps are needed.

Dynamic resource allocation is enabled by default. To modify dynamic allocation settings, use the following procedure.

Procedure

1. From the Cloudera interface, click through to Cloudera Manager for the cluster running Spark.
2. Go to the Spark service page (ClustersSpark service).
3. Click on the Configuration tab.
4. In the Search box, enter dynamicAllocation.
5. After making the changes you want, enter the reason for the change in the Reason for change... box, and then click Save Changes.
6. Restart the Spark service (Spark serviceActionsRestart).

Configure a Spark job for dynamic resource allocation

Use the following steps to configure dynamic resource allocation for a specific job.

There are two ways to customize dynamic resource allocation properties for a specific job:

- Include property values in the spark-submit command, using the -conf option.

  This approach loads the default spark-defaults.conf file first, and then applies property values specified in your spark-submit command.

  Example:

  spark-submit --conf “property_name=property_value”
- Create a job-specific spark-defaults.conf file and pass it to the spark-submit command.

  This approach uses the specified properties file, without reading the default property file.

  Example:

  spark-submit --properties-file <property_file>

Dynamic resource allocation properties

The following tables provide more information about dynamic resource allocation properties. These properties can be accessed by clicking through to Cloudera Manager from the Cloudera interface for the cluster running Spark.

Table 1: Dynamic Resource Allocation Properties

Property Name	Default Value	Description
spark.dynamicAllocation.enabled	false for Spark jobs	Enable dynamic allocation of executors in Spark applications.
spark.shuffle.service. enabled	true	Enables the external shuffle service. The external shuffle service preserves shuffle files written by executors so that the executors can be deallocated without losing work. Must be enabled if dynamic allocation is enabled.
spark.dynamicAllocation. initialExecutors	Same value as spark.dynamicAllocation. minExecutors	When dynamic allocation is enabled, number of executors to allocate when the application starts. Must be greater than or equal to the minExecutors value, and less than or equal to the maxExecutors value.
spark.dynamicAllocation. maxExecutors	infinity	When dynamic allocation is enabled, maximum number of executors to allocate. By default, Spark relies on YARN to control the maximum number of executors for the application.
spark.dynamicAllocation. minExecutors	0	When dynamic allocation is enabled, minimum number of executors to keep alive while the application is running.
spark.dynamicAllocation. executorIdleTimeout	60 seconds (60s)	When dynamic allocation is enabled, time after which idle executors will be stopped.
spark.dynamicAllocation. cachedExecutorIdleTimeout	infinity	When dynamic allocation is enabled, time after which idle executors with cached RDD blocks will be stopped.
spark.dynamicAllocation. schedulerBacklogTimeout	1 second (1s)	When dynamic allocation is enabled, timeout before requesting new executors when there are backlogged tasks.
spark.dynamicAllocation. sustainedSchedulerBacklogTimeout	Same value as schedulerBacklogTimeout	When dynamic allocation is enabled, timeout before requesting new executors after the initial backlog timeout has already expired. By default this is the same value as the initial backlog timeout.

Related Information

Apache Dynamic Resource Allocation

Spark security

When you create an environment in Cloudera Management Console, it automatically creates a Kerberos- and TLS-enabled data lake cluster. Data hub clusters are linked to environments, and therefore also have Kerberos enabled by default. You do not need to do anything to enable Kerberos or TLS for Apache Spark in Cloudera. Disabling security is not supported.

To submit Spark jobs to the cluster, users must authenticate with their Kerberos credentials. For more information, see the Environments documentation.

Enabling Spark authentication

Spark authentication here refers to an internal authentication mechanism, and not to Kerberos authentication, which is enabled automatically for all Cloudera deployments.

Before you begin

Minimum Required Role: Security Administrator (also provided by Full Administrator)

About this task

Spark has an internal mechanism that authenticates executors with the driver controlling a given application. This mechanism is enabled using the Cloudera Manager Admin Console, as detailed below. Cluster administrators can enable the spark.authenticate mechanism to authenticate the various processes that support a Spark application.

To enable this feature on the cluster:

Procedure

1. In the Cloudera Management Console, go to Data Hub Clusters.
2. Find and select the cluster you want to configure.
3. Click the link for the Cloudera Manager URL.
4. Go to Clusters <Cluster Name>Spark serviceConfiguration .
5. Scroll down to the Spark Authentication setting, or search for spark.authenticate to find it.
6. In the Spark Authentication setting, click the checkbox next to the Spark (Service-Wide) property to activate the setting.
7. Enter the reason for the change at the bottom of the screen, and then click Save Changes.
8. Restart YARN:
   a) Select Clusters YARN .
   b) Select Restart from the Actions drop-down selector.
9. Re-deploy the client configurations:
   a) Select Clusters *CLUSTER_NAME*
   b) Select Deploy Client Configurations from the Actions drop-down selector.
10. Restart stale services.

Enabling Spark Encryption

Before you begin

Before enabling encryption, you must first enable Spark authentication.

Procedure

1. In the Cloudera Management Console, go to Data Hub Clusters.
2. Find and select the cluster you want to configure.
3. Click the link for the Cloudera Manager URL.
4. Go to Clusters <Cluster Name>Spark serviceConfiguration .
5. Search for the Enable Network Encryption property. Use the checkbox to enable encrypted communication between Spark processes belonging to the same application.
6. Search for the Enable I/O Encryption property. Use the checkbox to enabled encryption for temporary shuffle and cache files stored by Spark on local disks.
7. Enter the reason for the change at the bottom of the screen, and then click Save Changes.
8. Re-deploy the client configurations:
   a) Select Clusters *CLUSTER_NAME*
   b) Select Deploy Client Configurations from the Actions drop-down selector.
9. Restart stale services.

Running Spark applications on secure clusters

All Cloudera clusters are secure by default. Disabling security on Cloudera clusters is not supported. To run a Spark application on a secure cluster, you must first authenticate using Kerberos.

Users running Spark applications must first authenticate to Kerberos, using kinit, as follows:

```
kinit username@EXAMPLE.COM
```

After authenticating to Kerberos, users can submit their applications using spark-submit as usual, as shown below. This command submits one of the default Spark sample jobs using an environment variable as part of the path, so modify as needed for your own use:

```
$ spark-submit --class org.apache.spark.examples.SparkPi --master yarn \
--deploy-mode cluster $SPARK_HOME/lib/spark-examples.jar 10
```

For information on creating user accounts in Cloudera, see Onboarding Users.

Configuring HSTS for Spark

You can configure Apache Spark to include HTTP headers to prevent Cross-Site Scripting (XSS), Cross-Frame Scripting (XFS), MIME-Sniffing, and also enforce HTTP Strict Transport Security (HSTS).

Procedure

1. Go to the Spark service.
2. Click the Configuration tab.
3. Select History Server under Scope.
4. Select Advanced under Category.
5. Set the following HSTS credentials in History Server Advanced Configuration Snippet (Safety Valve) for spark3-conf/spark-history-server.conf.

   ```
   spark.ui.strictTransportSecurity=max-age=31536000;includeSubDomains
   ```

6. Restart the Spark service.

Accessing compressed files in Spark

You can read compressed files using one of the following methods:

- textFile(*PATH*)
- hadoopFile(*PATH*,*OUTPUTFORMATCLASS*)

You can save compressed files using one of the following methods:

- saveAsTextFile(*PATH*, compressionCodecClass="*CODEC_CLASS*")
- saveAsHadoopFile(*PATH*,*OUTPUTFORMATCLASS*, compressionCodecClass="*CODEC_CLASS*")

where *CODEC_CLASS* is one of these classes:

- gzip - org.apache.hadoop.io.compress.GzipCodec
- bzip2 - org.apache.hadoop.io.compress.BZip2Codec
- LZO - com.hadoop.compression.lzo.LzopCodec
- Snappy - org.apache.hadoop.io.compress.SnappyCodec
- Deflate - org.apache.hadoop.io.compress.DeflateCodec

For examples of accessing Avro and Parquet files, see Spark with Avro and Parquet.

Using Spark History Servers with high availability

You can configure the load balancer for Spark History Server to ensure high availability, so that users can access and use the Spark History Server UI without any disruption. Learn how you can configure the load balancer for Spark History Server and the limitations associated with it.

You can access the Spark History Server for your Spark cluster from the Cloudera Management Console interface. The Spark History Server has two main functions:

- Reads the Spark event logs from the storage and displays them on the Spark History Server's user interface.
- Cleans the old Spark event log files.

Introducing high availability enables an extra load balancing layer so that users can see the Spark History Server's user interface without failure or disruption when there are two Spark History Servers in a cluster. Additionally, users are able to use the rolling restart feature for Spark History Server.

There are three supported ways to configure the load balancer for Spark History Server:

- Using an external load balancer for example, HAProxy.
- Using an internal load balancer which requires Apache Knox Gateway.
- Using multiple Apache Knox Gateways and external load balancers, for example, HAProxy.

.

Limitation for Spark History Server with high availability

You must be aware of a limitation related to how Spark History Server with high availability operates in Cloudera Manager.

- The second Spark History Server in the cluster does not clean the Spark event logs. Cleaning Spark event logs is automatically disabled from the Custom Service Descriptor. The second server can only read logs. This limitation ensures that two Spark History Servers do not try to delete the same files. If the first Spark History Server is down, the second one does not take over the cleaner task. This is not a critical issue because if the first Spark

History Server starts again, it will delete those old Spark event logs. The default event log cleaner interval (spark.history.fs.cleaner.interval) is 1 day in Cloudera Manager which means that the first Spark History Server only deletes the old logs once per day by default.

Configuring high availability for Spark History Server with an external load balancer

Learn how to configure high availability for Spark History Server using an external load balancer such as HAProxy. The authentication method used is SPNEGO (Kerberos) which requires an external load balancer.

Before you begin

Download and configure an external load balancer. If you do not have a load-balancing proxy, you can experiment your configurations with HAProxy which is a free open-source load balancer. HAProxy is not a CHD component, and Cloudera does not provide support for HAProxy. Ensure that SPNEGO authentication is enabled in Spark History Server.

Procedure

1. In Cloudera Manager, navigate to Spark Configuration and configure the history_server_load_balancer_url property.

   The history_server_load_balancer_url property configures the following automatically:

   - The load balancer SPNEGO principal is automatically generated and added to the keytab file. For example: HTTP/loadbalancer.example.com/EXAMPLE.com
   - The value of SPNEGO_PRINCIPAL is automatically set to * , and all HTTP principals are loaded from the automatic generated keytab
   - The value of spark.yarn.historyServer.address is set to this URL in the appropriate gateway files:
     - Spark3: /etc/spark3/conf/spark-defaults.conf
2. Use the following test commands for this configuration:

   curl --negotiate -u : --location-trusted -c cookies.dat -b cookies.dat -k "https://loadbalancer.example.com:18488"

   curl --negotiate -u : --location-trusted -c cookies.dat -b cookies.dat -k "https://loadbalancer.example.com:18489"

Example

This is the sample haproxy.cfg file that isused in the example below:

```
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 5
    timeout http-request    10s
    timeout queue           1m
    timeout connect         3s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
```

```
#---------------------------------------------------------------------
# Spark2 frontend which proxys to the Spark2 backends
#---------------------------------------------------------------------
frontend                        spark_front
    bind                        *:18488 ssl crt /var/lib/cloudera-scm-agent/
agent-cert/cdep-host_key_cert_chain_decrypted.pem
    default_backend             spark2

#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend spark2
    balance                     source
    server spark2-1 shs1.example.com:18488 check ssl ca-file /var/lib/clo
udera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
    server spark2-2 shs2.example.com:18488 check ssl ca-file /var/lib/clou
dera-scm-agent/agent-cert/cm-auto-global_cacerts.pem

#---------------------------------------------------------------------
# Spark3 frontend which proxys to the Spark3 backends
#---------------------------------------------------------------------
frontend                        spark3_front
    bind                        *:18489 ssl crt /var/lib/cloudera-scm-agent/
agent-cert/cdep-host_key_cert_chain_decrypted.pem
    default_backend             spark3
#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend spark3
    balance                     source
    server spark3-1 shs3.example.com:18489 check ssl ca-file /var/lib/clou
dera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
    server spark3-2 shs4.example.com:18489 check ssl ca-file /var/lib/cloud
era-scm-agent/agent-cert/cm-auto-global_cacerts.pem
```

Related Information

HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer

Configuring Load Balancer for Impala

Configuring high availability for Spark History Server with an internal load balancer

Learn how to configure high availability for Spark History Server using an internal load balancer. The authentication method for the internal load balancer uses a username and password through Apache Knox Gateway. The Cloudera stack includes the Apache Knox Gateway which has a built-in load balancer and failover mechanism.

Before you begin

One Knox service must be installed in Cloudera Manager.

About this task

This internal load balancer is only recommended for testing purposes because only one Knox Gateway is a single point of failure.

The following Knox topology configuration is automatically generated if there are two Spark History Server clusters in a Cloudera Manager cluster:

```
knox.example.com:/var/lib/knox/gateway/conf/topologies/cdp-proxy.xml

    <service>
```

```
        <role>SPARK3HISTORYUI</role>
        <url>https://shs1.example.com:18489</url>
        <url>https://shs2.example.com:18489</url>
    </service>
```

Procedure

To use the Knox load balancing feature, you must use the Knox Gateway URL. If one of the Spark History Servers is down, the connection will be automatically redirected to the other server. See the example Knox Gateway URLs below:

Spark3: https://knox.example.com:8443/gateway/cdp-proxy/spark3history/

Configuring high availability for Spark History Server with multiple Knox Gateways

Learn how to configure high availability for Spark History Server using an external load balancer such as HAProxy with multiple Apache Knox Gateway services. If you have only one Knox Gateway, this will be a single point of failure. For example, if the Knox Gateway goes down, and you are using the Knox Gateway's URL to access the Spark History Server's user interface, it will fail.

Before you begin

You must have installed two or more Knox Gateway services in Cloudera Manager, and one external load balancer, for example, HAProxy.

About this task

The following is an example HAProxy configuration for Knox Gateways. This is the sample haproxy.cfg file that is used in this example:

```
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 5
    timeout http-request    10s
    timeout queue           1m
    timeout connect         3s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

frontend knox-frontend
    bind :8443 ssl crt /etc/haproxy/cm-auto-host_cert_chain_unenckey.pem
    mode http
    stats enable
    option forwardfor
    http-request redirect location https://%[req.hdr(Host)]/gateway/homepag
e/home/ if { path / }
    default_backend knox-backend

backend knox-backend
```

```
    mode http
    option redispatch
    balance leastconn
    option forwardfor
    stick-table type ip size 1m expire 24h
    stick on src
    option httpchk HEAD /gateway/knoxsso/knoxauth/login.html HTTP/1.1\r\nHo
st:\ hwx.site
    http-check expect status 200
    # Knox nodes
    server shs1.example.com shs1.example.com:8443 check ssl ca-file /var/lib
/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
    server shs2.example.com shs2.example.com:8443 check ssl ca-file /var/l
ib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
    server shs3.example.com shs3.example.com:8443 check ssl ca-file /var/lib
/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
    server shs4.example.com shs4.example.com:8443 check ssl ca-file /var/l
ib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
    server shs5.example.com shs5.example.com:8443 check ssl ca-file /var/lib
/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem

frontend stats
    bind *:8404
    stats enable
    stats uri /stats
    stats refresh 1s
```

Procedure

Add the following HAProxy's URL to the Spark gateway configurations:

Spark3: Cloudera Manager > Spark3 > Configuration > history_server_load_balancer_url https://loadbalancer.example.com:8443/gateway/cdp-proxy/spark3history/

Related Information

Apache Knox

HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer

Configuring Load Balancer for Impala

How to access Spark files on Ozone

Learn how you can connect Spark to the Ozone object store with the help of a sample script.

This script, in Scala, counts the number of word occurrences in a text file. The key point in this example is to use the following string to refer to the text file: ofs://omservice1/s3v/hivetest/spark/jedi_wisdom.txt

Word counting example in Scala

```
import sys.process._

// Put the input file into Ozone
//"hdfs dfs -put data/jedi_wisdom.txt ofs://omservice1/s3v/hivetest/spark" !
// Set the following spark setting in the file "spark-defaults.conf" on the
 CML session using terminal
//spark.kerberos.access.hadoopFileSystems=ofs://omservice1/s3v/hivetest
//count lower bound
val threshold = 2
// this file must already exist in hdfs, add a
// local version by dropping into the terminal.
```

```
val tokenized = sc.textFile("ofs://omservice1/s3v/hivetest/spark/jedi_wisd
om.txt").flatMap(_.split(" "))
// count the occurrence of each word
val wordCounts = tokenized.map((_ , 1)).reduceByKey(_ + _)
// filter out words with fewer than threshold occurrences
val filtered = wordCounts.filter(_._2 >= threshold)
System.out.println(filtered.collect().mkString(","))
```

Cloudera Runtime 7.3.1

Data Protection

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Data protection....4
Backing up HDFS metadata....4
Introduction to HDFS metadata files and directories....4
Back up HDFS metadata....8
Configuring HDFS trash....11
Trash behavior with HDFS Transparent Encryption enabled....11
Enabling and disabling trash....12
Setting the trash interval....12
Using HDFS snapshots for data protection....12
Considerations for working with HDFS snapshots....13
Enable snapshot creation on a directory....13
Create snapshots on a directory....13
Recover data from a snapshot....14
Options to determine differences between contents of snapshots....14
CLI commands to perform snapshot operations....15
Managing snapshot policies using 16
Enable and disable snapshot creation using 17
Create snapshots using 17
Delete snapshots using 18
Preventing inadvertent deletion of directories....18

Data protection

You can ensure data protection by preventing accidental deletion of files and backing up HDFS metadata.

Backing up HDFS metadata

HDFS metadata represents the structure and attributes of HDFS directories and files in a tree. You can back up the metadata without affecting NameNode availability.

Introduction to HDFS metadata files and directories

HDFS metadata represents the structure of HDFS directories and files in a tree. It also includes the various attributes of directories and files, such as ownership, permissions, quotas, and replication factor.

Files and directories

Persistence of HDFS metadata is implemented using fsimage file and edits files.

Attention:

Do not attempt to modify metadata directories or files. Unexpected modifications can cause HDFS downtime, or even permanent data loss. This information is provided for educational purposes only.

Persistence of HDFS metadata broadly consist of two categories of files:

fsimage

Contains the complete state of the file system at a point in time. Every file system modification is assigned a unique, monotonically increasing transaction ID. An fsimage file represents the file system state after all modifications up to a specific transaction ID.

edits file

Contains a log that lists each file system change (file creation, deletion or modification) that was made after the most recent fsimage.

Checkpointing is the process of merging the content of the most recent fsimage, with all edits applied after that fsimage is merged, to create a new fsimage. Checkpointing is triggered automatically by configuration policies or manually by HDFS administration commands.

NameNodes

Understand the HDFS metadata directory details taken from a NameNode.

The following example shows an HDFS metadata directory taken from a NameNode. This shows the output of running the tree command on the metadata directory, which is configured by setting *DFS.NAMENODE.NAME.DIR* in hdfs-site.xml.

```
data/dfs/name
### current#
### VERSION#
### edits_0000000000000000001-0000000000000000007
# ### edits_0000000000000000008-0000000000000000015
# ### edits_0000000000000000016-0000000000000000022
# ### edits_0000000000000000023-0000000000000000029
# ### edits_0000000000000000030-0000000000000000030
# ### edits_0000000000000000031-0000000000000000031
# ### edits_inprogress_0000000000000000032
# ### fsimage_0000000000000000030
# ### fsimage_0000000000000000030.md5
# ### fsimage_0000000000000000031
# ### fsimage_0000000000000000031.md5
```

```
# ### seen_txid
### in_use.lock
```

In this example, the same directory has been used for both fsimage and edits. Alternative configuration options are available that allow separating fsimage and edits into different directories. Each file within this directory serves a specific purpose in the overall scheme of metadata persistence:

VERSION

Text file that contains the following elements:

layoutVersion

Version of the HDFS metadata format. When you add new features that require a change to the metadata format, you change this number. An HDFS upgrade is required when the current HDFS software uses a layout version that is newer than the current one.

namespaceID/clusterID/blockpoolID

Unique identifiers of an HDFS cluster. These identifiers are used to prevent DataNodes from registering accidentally with an incorrect NameNode that is part of a different cluster. These identifiers also are particularly important in a federated deployment. Within a federated deployment, there are multiple NameNodes working independently. Each NameNode serves a unique portion of the namespace (namespaceID) and manages a unique set of blocks (blockpoolID). The clusterID ties the whole cluster together as a single logical unit. This structure is the same across all nodes in the cluster.

storageType

Always NAME_NODE for the NameNode, and never JOURNAL_NODE.

cTime

Creation time of file system state. This field is updated during HDFS upgrades.

edits_start transaction ID-end transaction ID

Finalized and unmodifiable edit log segments. Each of these files contains all of the edit log transactions in the range defined by the file name. In an High Availability deployment, the standby can only read up through the finalized log segments. The standby NameNode is not up-to-date with the current edit log in progress. When an HA failover happens, the failover finalizes the current log segment so that it is completely caught up before switching to active.

fsimage_end transaction ID

Contains the complete metadata image up through . Each fsimage file also has a corresponding .md5 file containing a MD5 checksum, which HDFS uses to guard against disk corruption.

seen_txid

Contains the last transaction ID of the last checkpoint (merge of edits into an fsimage) or edit log roll (finalization of current edits_inprogress and creation of a new one). This is not the last transaction ID accepted by the NameNode. The file is not updated on every transaction, only on a checkpoint or an edit log roll. The purpose of this file is to try to identify if edits are missing during startup. It is possible to configure the NameNode to use separate directories for fsimage and edits files. If the edits directory accidentally gets deleted, then all transactions since the last checkpoint would go away, and the NameNode starts up using just fsimage at an old state. To guard against this, NameNode startup also checks seen_txid to verify that it can load transactions at least up through that number. It aborts startup if it cannot verify the load transactions.

in_use.lock

Lock file held by the NameNode process, used to prevent multiple NameNode processes from starting up and concurrently modifying the directory.

JournalNodes

Understand the components of the JournalNode metadata directory.

In an HA deployment, edits are logged to a separate set of daemons called JournalNodes. A JournalNode's metadata directory is configured by setting *DFS.JOURNALNODE.EDITS.DIR*. The JournalNode contains a VERSION file, multiple edits__ files and an edits_inprogress_, just like the NameNode. The JournalNode does not have fsimage files or seen_txid. In addition, it contains several other files relevant to the HA implementation. These files help prevent a split-brain scenario, in which multiple NameNodes could think they are active and all try to write edits.

committed-txid

Tracks last transaction ID committed by a NameNode.

last-promised-epoch

Contains the "epoch," which is a monotonically increasing number. When a new NameNode, starts as active, it increments the epoch and presents it in calls to the JournalNode. This scheme is the NameNode's way of claiming that it is active and requests from another NameNode, presenting a lower epoch, must be ignored.

last-writer-epoch

Contains the epoch number associated with the writer who last actually wrote a transaction.

paxos

Specifies the directory that temporary files used in the implementation of the Paxos distributed consensus protocol. This directory often appears as empty.

DataNodes

Although DataNodes do not contain metadata about the directories and files stored in an HDFS cluster, they do contain a small amount of metadata about the DataNode itself and its relationship to a cluster.

This shows the output of running the tree command on the DataNode's directory, configured by setting *DFS.DATANODE.DATA.DIR IN HDFS-SITE.XML*.

```
data/dfs/data/
### current
# ### BP-1079595417-192.168.2.45-1412613236271
# # ### current
# # # ### VERSION
# # # ### finalized
# # # # ### subdir0# # # # ### subdir1
# # # # ### blk_1073741825
# # # # ### blk_1073741825_1001.meta
# # # ### lazyPersist
# # # ### rbw
# # ### dncp_block_verification.log.curr
# # ### dncp_block_verification.log.prev
# # ### tmp
# ### VERSION
### in_use.lock
```

The purpose of these files are as follows:

BP-random integer-NameNode-IP address-creation time

Top level directory for datanodes. The naming convention for this directory is significant and constitutes a form of cluster metadata. The name is a block pool ID. "BP" stands for "block pool," the abstraction that collects a set of blocks belonging to a single namespace. In the case of a federated deployment, there are multiple "BP" sub-directories, one for each block pool. The remaining components form a unique ID: a random integer, followed by the IP address of the NameNode that created the block pool, followed by creation time.

VERSION

Text file containing multiple properties, such as layoutVersion, clusterId and cTime, which is much like the NameNode and JournalNode. There is a VERSION file tracked for the entire DataNode as well as a separate VERSION file in each block pool sub-directory.

In addition to the properties already discussed earlier, the DataNode's VERSION files also contain:

storageType

storageType field is set to DATA_NODE.

blockpoolID

Repeats the block pool ID information encoded into the sub-directory name.

finalized/rbw

Both finalized and rbw contain a directory structure for block storage. This holds numerous block files, which contain HDFS file data and the corresponding .meta files, which contain checksum information. rbw stands for "replica being written". This area contains blocks that are still being written to by an HDFS client. The finalized sub-directory contains blocks that are not being written to by a client and have been completed.

lazyPersist

HDFS is incorporating a new feature to support writing transient data to memory, followed by lazy persistence to disk in the background. If this feature is in use, then a lazyPersist sub-directory is present and used for lazy persistence of in-memory blocks to disk. We'll cover this exciting new feature in greater detail in a future blog post.

scanner.cursor

File to which the "cursor state" is saved.

The DataNode runs a block scanner which periodically does checksum verification of each block file on disk. This scanner maintains a "cursor," representing the last block to be scanned in each block pool slice on the volume, and called the "cursor state."

in_use.lock

Lock file held by the DataNode process, used to prevent multiple DataNode processes from starting up and concurrently modifying the directory.

HDFS commands for metadata files and directories

You can use HDFS commands to manipulate metadata files and directories.

hdfs namenode

Automatically saves a new checkpoint at NameNode startup. As stated earlier, checkpointing is the process of merging any outstanding edit logs with the latest fsimage, saving the full state to a new fsimage file, and rolling edits. Rolling edits means finalizing the current edits_inprogress and starting a new one.

hdfs dfsadmin -safemode enter
hdfs dfsadmin -saveNamespace

Saves a new checkpoint (similar to restarting NameNode) while the NameNode process remains running. The NameNode must be in safe mode, and all attempted write activity fails while this command runs.

hdfs dfsadmin -rollEdits

Manually rolls edits. Safe mode is not required.

This can be useful if a standby NameNode is lagging behind the active NameNode and you want it to get caught up more quickly. The standby NameNode can only read finalized edit log segments, not the current in progress edits file.

hdfs dfsadmin -fetchImage

Downloads the latest fsimage from the NameNode. This can be helpful for a remote backup type of scenario.

Configuration properties

Use the NameNode and DataNode properties to configure the NameNode and DataNodes.

dfs.namenode.name.dir

Specifies where on the local filesystem the DFS name node stores the name table (fsimage). If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.

dfs.namenode.edits.dir

Specifies where on the local filesystem the DFS name node stores the transaction (edits) file. If this is a comma-delimited list of directories, the transaction file is replicated in all of the directories, for redundancy. The default value is set to the same value as dfs.namenode.name.dir.

dfs.namenode.checkpoint.period

Specifies the number of seconds between two periodic checkpoints.

dfs.namenode.checkpoint.txns

The standby creates a checkpoint of the namespace every dfs.namenode.checkpoint.txns transactions, regardless of whether dfs.namenode.checkpoint.period has expired.

dfs.namenode.checkpoint.check.period

Specifies how frequently to query for the number of un-checkpointed transactions.

dfs.namenode.num.checkpoints.retained

Specifies the number of image checkpoint files to be retained in storage directories. All edit logs necessary to recover an up-to-date namespace from the oldest retained checkpoint are also retained.

dfs.namenode.num.extra.edits.retained

Specifies the number of extra transactions which are retained beyond what is minimally necessary for a NN restart. This can be useful for audit purposes or for an HA setup where a remote Standby Node might have been offline and need to have a longer backlog of retained edits to start again.

dfs.namenode.edit.log.autoroll.multiplier.threshold

Specifies when an active namenode rolls its own edit log. The actual threshold (in number of edits) is determined by multiplying this value by *DFS.NAMENODE.CHECKPOINT.TXNS*. This prevents extremely large edit files from accumulating on the active namenode, which can cause timeouts during namenode start-up and pose an administrative hassle. This behavior is intended as a fail-safe for when the standby fails to roll the edit log by the normal checkpoint threshold.

dfs.namenode.edit.log.autoroll.check.interval.ms

Specifies the time in milliseconds that an active namenode checks if it needs to roll its edit log.

dfs.datanode.data.dir

Determines where on the local filesystem an DFS data node should store its blocks. If this is a comma-delimited list of directories, then data is stored in all named directories, typically on different devices. Directories that do not exist are ignored. Heterogeneous storage allows specifying that each directory resides on a different type of storage: DISK, SSD, ARCHIVE or RAM_DISK.

Back up HDFS metadata

You can back up HDFS metadata without taking down either HDFS or the NameNodes.

Prepare to back up the HDFS metadata

Regardless of the solution, a full, up-to-date continuous backup of the namespace is not possible. Some of the most recent data is always lost. HDFS is not an Online Transaction Processing (OLTP) system. Most data can be easily recreated if you re-run Extract, Transform, Load (ETL) or processing jobs.

- Normal NameNode failures are handled by the Standby NameNode. Doing so creates a safety-net for the very unlikely case where both master NameNodes fail.
- In the case of both NameNode failures, you can start the NameNode service with the most recent image of the namespace.
- Name Nodes maintain the namespace as follows:
  - Standby NameNodes keep a namespace image in memory based on edits available in a storage ensemble in Journal Nodes.
  - Standby NameNodes make a namespace checkpoint and saves an fsimage_* to disk.
  - Standby NameNodes transfer the fsimage to the primary NameNodes using HTTP.

Both NameNodes write fsimages to disk in the following sequence:

- NameNodes write the namespace to a file fsimage.ckpt_* on disk.
- NameNodes creates an fsimage_*.md5 file.
- NameNodes moves the file fsimage.ckpt_* to fsimage_.*.

The process by which both NameNodes write fsimages to disk ensures that:

- The most recent namespace image on disk in an fsimage_* file is on the standby NameNode.
- Any fsimage_* file on disk is finalized and does not receive updates.

Backing up NameNode metadata

You must back up the VERSION file and then back up the NameNode metadata.

Procedure

1. Make a single backup of the VERSION file.

   This does not need to be backed up regularly as it does not change, but it is important since it contains the clusterID, along with other details.
2. Use the following command to back up the NameNode metadata.

   It automatically determines the active NameNode, retrieves the current fsimage, and places it in the defined backup_dir.

   ```
   hdfs dfsadmin -fetchImage backup_dir
   ```

Results

On startup, the NameNode process reads the fsimage file and commits it to memory. If the JournalNodes are up and running, and there are edit files present, any edits newer than the fsimage are also applied. If the JournalNodes are unavailable, it is possible to lose any data transferred in the interim.

Back up HDFS metadata using

HDFS metadata backups can be used to restore a NameNode when both NameNode roles have failed. In addition, recommends backing up HDFS metadata before a major upgrade.

About this task

This backup method requires you to shut down the cluster.

Procedure

1. Note the active NameNode.

2. Stop the cluster.

   It is particularly important that the NameNode role process is not running so that you can make a consistent backup.
3. Go to the HDFS service.
4. Click the Configuration tab.
5. In the Search field, search for "NameNode Data Directories" and note the value.
6. On the active NameNode host, back up the directory listed in the NameNode Data Directories property. If a file with the extension lock exists in the NameNode data directory, the NameNode most likely is still running. Repeat the steps, beginning with shutting down the NameNode role.

   If more than one is listed, make a backup of one directory, because each directory is a complete copy. For example, if the NameNode data directory is /data/dfs/nn, do the following as root:

   ```
   # cd /data/dfs/nn
   # tar -cvf /root/nn_backup_data.tar .
   ```

   You should see output like this:

   ```
   /dfs/nn/current
   ./
   ./VERSION
   ./edits_0000000000000000001-0000000000000008777
   ./edits_0000000000000008778-0000000000000009337
   ./edits_0000000000000009338-0000000000000009897
   ./edits_0000000000000009898-0000000000000010463
   ./edits_0000000000000010464-0000000000000011023
   <snip>
   ./edits_0000000000000063396-0000000000000063958
   ./edits_0000000000000063959-0000000000000064522
   ./edits_0000000000000064523-0000000000000065091
   ./edits_0000000000000065092-0000000000000065648
   ./edits_inprogress_0000000000000065649
   ./fsimage_0000000000000065091
   ./fsimage_0000000000000065091.md5
   ./fsimage_0000000000000065648
   ./fsimage_0000000000000065648.md5
   ./seen_txid
   ```

Restoring NameNode metadata

If both the NameNode and the secondary NameNode were to suddenly go offline, you can restore the NameNode.

Procedure

1. Add a new host to your Hadoop cluster.
2. Add the NameNode role to the host. Make sure it has the same hostname as the original NameNode.
3. Create a directory path for the NameNode name.dir (for example, /dfs/nn/current), ensuring that the permissions are set correctly.
4. Copy the VERSION and latest fsimage file to the /dfs/nn/current directory.
5. Run the following command to create the md5 file for the fsimage.

   ```
   md5sum fsimage > fsimage.md5
   ```

6. Start the NameNode process.

Restore HDFS metadata from a backup using

When both the NameNode hosts have failed, you can use to restore HDFS metadata.

Procedure

1. Remove the NameNode, JournalNode, and Failover Controller roles from the HDFS service.
2. Add the host on which the NameNode role will run.
3. Create the NameNode data directory, ensuring that the permissions, ownership, and group are set correctly.
4. Copy the backed up files to the NameNode data directory.
5. Add the NameNode role to the host.
6. Add the Secondary NameNode role to another host.
7. Enable high availability.

   If not all roles are started after the wizard completes, restart the HDFS service. Upon startup, the NameNode reads the fsimage file and loads it into memory. If the JournalNodes are up and running and there are edit files present, any edits newer than the fsimage are applied.

Perform a backup of the HDFS metadata

You can back up HDFS metadata without affecting the availability of NameNode.

Procedure

1. Make sure the Standby NameNode checkpoints the namespace to fsimage_ once per hour.
2. Deploy monitoring on both NameNodes to confirm that checkpoints are triggering regularly.

   This helps reduce the amount of missing transactions in the event that you need to restore from a backup containing only fsimage files without subsequent edit logs. It is good practice to monitor this because edit logs that are large in size and without checkpoints can cause long delays after a NameNode restart while it replays those transactions.
3. Back up the most recent “fsimage_*” and “fsimage_*.md5” from the standby NameNode periodically.

   Try to keep the latest version of the file on another machine in the cluster.
4. Back up the VERSION file from the standby NameNode.

Configuring HDFS trash

The Hadoop trash feature helps prevent accidental deletion of files and directories.

When you delete a file in HDFS, the file is not immediately expelled from HDFS. Deleted files are first moved to the /user/<username>/.Trash/Current directory, with their original filesystem path being preserved. After a user-configurable period of time (fs.trash.interval), a process known as trash checkpointing renames the Current directory to the current timestamp, that is, /user/<username>/.Trash/<timestamp>. The checkpointing process also checks the rest of the .Trash directory for any existing timestamp directories and removes them from HDFS permanently. You can restore files and directories in the trash simply by moving them to a location outside the .Trash directory.

Important: The trash feature is enabled by default. recommends that you enable it on all production clusters.

Trash behavior with HDFS Transparent Encryption enabled

You can delete files or directories that are part of an HDFS encryption zone. Moving and renaming files or directories is an important part of trash handling in HDFS.

HDFS creates a local .Trash directory every time a new encryption zone is created. For example, when you create an encryption zone, /enc_zone, HDFS will also create the /enc_zone/.Trash/ sub-directory. Files deleted from enc_zone are moved to /enc_zone/.Trash/<username>/Current/. After the checkpoint, the Current directory is renamed to the current timestamp, /enc_zone/.Trash/<username>/<timestamp>.

If you delete the entire encryption zone, it will be moved to the .Trash directory under the user's home directory, /users/<username>/.Trash/Current/enc_zone. Trash checkpointing will occur only after the entire zone has been moved to /users/<username>/.Trash. However, if the user's home directory is already part of an encryption zone, then

attempting to delete an encryption zone will fail because you cannot move or rename directories across encryption zones.

If the trash directory is deleted by mistake, create the .Trash directory using the -provisionTrash option as follows:

```
hdfs crypto -provisionTrash -path /enc_zone
```

If required, you can use the following commands to manually create the .Trash directory within an encryption zone. Make sure you run the commands as an admin user.

```
hdfs dfs -mkdir /enc_zone/.Trash
hdfs dfs -chmod 1777 /enc_zone/.Trash
```

Enabling and disabling trash

You can use to enable and disable HDFS trash.

Procedure

1. Go to the HDFS service.
2. Click the Configurations tab.
3. Select Scope Gateway .
4. Select or clear the Use Trash checkbox.

   To apply this configuration property to other role groups as needed, edit the value for the appropriate role group.
5. Restart the cluster and deploy the cluster client configuration.

Setting the trash interval

You can use to specify the time period after which a trash checkpoint directory is deleted.

Procedure

1. Go to the HDFS service.
2. Click the Configurations tab.
3. Select Scope NameNode .
4. Specify the Filesystem Trash Interval property, which controls the number of minutes after which a trash checkpoint directory is deleted and the number of minutes between trash checkpoints.

   For example, to enable trash so that deleted files are deleted after 24 hours, set the value of the Filesystem Trash Interval property to 1440.

   Note: The trash interval is measured from the point at which the files are moved to trash, not from the last time the files were modified.

   To apply this configuration property to other role groups as needed, edit the value for the appropriate role group.
5. Restart all NameNodes.

Using HDFS snapshots for data protection

HDFS snapshots enable you to capture point-in-time copies of the file system and protect your important data against user or application errors. recommends that you take snapshots of specified subtrees on the file system.

Using snapshots to protect data is efficient because of the following reasons:

- Snapshot creation is instantaneous regardless of the size and depth of the directory subtree.
- Snapshots capture the block list and file size for a specified subtree. Snapshots do not create extra copies of blocks on the file system.

You can either use the command line interface or to manage HDFS snapshots.

Considerations for working with HDFS snapshots

You can create snapshots only for directories that allow the creation of snapshots. If a directory already contains snapshots, you cannot delete or rename the directory unless you remove all the snapshots.

You must consider the following when working with HDFS snapshots:

- You must enable snapshot creation on a particular directory before creating snapshots on that directory. Such a directory is termed as a *snapshottable* directory. However, you cannot create snapshots on a directory if its corresponding child or parent directory is already enabled for snapshot creation.
- You cannot delete or rename a directory that contains snapshots. You must first remove all the snapshots before attempting the delete or rename operation.
- For a snapshottable directory, its path component .snapshot can be used to access the snapshots.

  For example, consider the directory /foo that is enabled for snapshot creation. For the directory /foo with a snapshot snap1, the path /foo/.snapshot/snap1 refers to the snapshot of /foo.
- You can enable or disable snapshot creation on a particular directory only if you have the superuser privilege.
- On a snapshottable directory; you can create, delete, or rename snapshots. These operations require either the superuser privilege or the owner access to the directory. In addition, you can list directories that have snapshot creation enabled or view differences between contents of snapshots.
- You cannot rename a snapshot outside the snapshottable directory.

Enable snapshot creation on a directory

You must enable snapshot creation on a directory before creating snapshots on that directory. If the snapshot creation is enabled, the directory becomes *snapshottable*.

About this task

- You can perform this task only if you have the superuser privilege.
- You cannot enable snapshot creation on any directory if its parent or child directory is already enabled for snapshot creation.

Procedure

Run the hdfs dfsadmin command with the -allowSnapshot option and specify the directory on which you want to enable snapshot creation.
The following example shows how you can enable snapshot creation for the directory /data/dir1:

```
hdfs dfsadmin -allowSnapshot /data/dir1
```

If snapshot creation is successfully enabled on the specified directory, a confirmation message appears.

```
Allowing snapshot on /data/dir1 succeeded
```

Related Information

Enable and disable snapshot creation using

Create snapshots on a directory

You can create snapshots on a specified directory and protect your important data.

Before you begin

You must have enabled snapshot creation.

About this task

Only a user with either of the following privileges can perform this task:

- The owner privilege to the directory on which to create the snapshots
- The superuser privilege

Procedure

Run the hdfs dfs command with the -createSnapshot option and specify the path to the directory on which you want to create snapshots.
The following example shows how you can create a snapshot snap1 on the directory /data/dir1:

```
hdfs dfs -createSnapshot /data/dir1 snap1
```

If snapshot creation is successfully enabled on the specified directory, a confirmation message appears.

```
Created snapshot /data/dir1/.snapshot/snap2
```

Note: You can also run the command without mentioning the snapshot name. In such a situation, the new snapshot has the time stamp of creation as its name. See the following example:

```
Created snapshot /data/dir1/.snapshot/s20180412-065533.159
```

Related Information

Create snapshots using
Delete snapshots using

Recover data from a snapshot

If data is erroneously removed from a directory for which snapshots are available, you can recover the lost data using snapshots. The snapshot ensures that the file blocks corresponding to the deleted files or directories are not removed from the file system. Only the metadata is modified to reflect the deletion.

About this task

You must have read access to the files or directories that you want to restore.

Procedure

Run the hdfs dfs command with the cp option to copy the deleted data from the snapshot to the destination directory. The following example shows how you can recover a file imp_details.xls from a snapshot of the directory (/data/dir1) that contained the file:

```
hdfs dfs -cp /data/dir1/.snapshot/s20180412-065533.159/imp_details.xls /data
/dir1/
```

Related Information

Delete snapshots using

Options to determine differences between contents of snapshots

Run the hdfs snapshotDiff command for a report that lists the difference between the contents of two snapshots. Run the distcp diff command to determine the difference between contents of specified source and target snapshots, and use the command with the -update option to move the difference to a specified target directory.

Generating a report listing the difference between contents of two snapshots

Using the hdfs snapshotDiff between two snapshots on a specified directory path provides the list of changes to the directory. Consider the following example:

```
hdfs snapshotDiff /data/dir1 snap1 snap2
M .
- ./file1.csv
R ./file2.txt -> ./fileold.txt
+ ./filenew.txt
```

This example shows the following changes to the directory /data/dir1 after the creation of snap1 and before the creation of snap2:

Statement	Explanation
M .	The directory /data/dir1 is modified.
- ./file1.csv	The file file1.csv is deleted.
R ./file2.txt -> ./fileold.txt	The file file2.txt is renamed to fileold.txt.
+ ./filenew.txt	The file filenew.txt is added to the directory /data/dir1.

Moving the differences between the contents of two snapshots to a specified directory

Using the distcp diff command with the -update option on snapshots enables you to determine the difference between the contents of two snapshots and move the difference to a specified target directory. Consider the following example:

hadoop distcp -diff snap_old snap_new -update /data/source_dir /data/target_dir

The command in this example determines the changes between the snapshots snap_old and snap_new present in the source_dir directory, and updates the target_dirdirectory with the changes.

The following conditions must be satisfied for the content changes to be moved to /data/target_dir:

- Both /data/source_dir and /data/target_dir are distributed file system paths.
- The snapshots snap_old and snap_new are created for /data/source_dir such that snap_old is older than snap_new.
- The /data/target_dir path also contains snap_old. In addition, no changes are made to /data/target_dir after the creation of snap_old.

CLI commands to perform snapshot operations

As an administrator, you can enable or disable snapshot creation on a directory. These operations require the superuser privilege. As a user; you can create, delete, or rename snapshots on a directory that has snapshot creation enabled. These operations require either the superuser privilege or the owner privilege on the directory.

Administrator operations

The following table lists the snapshot-related administrator operations that you can on specified directories:

Operation	Command
Enable snapshot creation on a directory	hdfs dfsadmin -allowSnapshot <path>
Disable snapshot creation on a directory	hdfs dfsadmin -disallowSnapshot <path>

For more information about these commands, see https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsSnapshots.html#Administrator_Operations.

User operations

The following table lists the user operations that you can perform on snapshots:

Operation	Command
Create snapshots	hdfs dfs -createSnapshot <path> [<snapshotName>]
Delete snapshots	hdfs dfs -deleteSnapshot <path> <snapshotName>
Rename snapshots	hdfs dfs -renameSnapshot <path> <oldName> <newName>
List directories on which snapshot creation is enabled (*snapshottable* directories)	hdfs lsSnapshottableDir
List snapshots on a snapshottable directory with their IDs and timestamp of creation	hdfs lsSnapshot
List differences between contents of snapshots	hdfs snapshotDiff <path> <fromSnapshot> <toSnapshot>

For more information about these commands, see https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsSnapshots.html#User_Operations.

Managing snapshot policies using

enables the creation of snapshot policies that define the directories or tables for which snapshots should be taken, the intervals at which snapshots should be taken, and the number of snapshots that should be kept for each snapshot interval.

For example, you can create a policy that takes both daily and weekly snapshots, and specify that seven daily snapshots and five weekly snapshots should be maintained.

Create a snapshot policy

You can use Cloudera Manager to create snapshot policies for a directory.

Before you begin

The directory for which you want to create a snapshot policy must be *snapshottable*.

Procedure

1. Select Backup Snapshot Policies in the left navigation bar.

   Existing snapshot policies are shown in a table on the Snapshot Policies page.
2. Click Create Snapshot Policy.

   The Create Snapshot Policy window displays.
3. From the drop-down list, select the service (HDFS or HBase) and cluster for which you want to create a policy.
4. Provide a name for the policy and, optionally, a description.
5. Specify the directories, namespaces or tables to include in the snapshot.

   Important: Do not take snapshots of the root directory.

   - For an HDFS service, select the paths of the directories to include in the snapshot. The drop-down list allows you to select only directories that are enabled for snapshot creation. If no directories are enabled for snapshot creation, a warning displays.

     Click ✚ to add a path and ➖ to remove a path.
   - For an HBase service, list the tables to include in your snapshot. You can use a Java regular expression to specify a set of tables. For example, finance.* match all tables with names starting with finance. You can also create a snapshot for all tables in a given namespace, using the {*NAMESPACE*}:.* syntax.

6. Specify the snapshot Schedule.

   You can schedule snapshots hourly, daily, weekly, monthly, or yearly, or any combination of those. Depending on the frequency you select, you can specify the time of day to take the snapshot, the day of the week, day of the month, or month of the year, and the number of snapshots to keep at each interval. Each time unit in the schedule information is shared with the time units of larger granularity. That is, the minute value is shared by all the selected schedules, hour by all the schedules for which hour is applicable, and so on. For example, if you specify that hourly snapshots are taken at the half hour, and daily snapshots taken at the hour 20, the daily snapshot will occur at 20:30.
7. Specify whether Alerts should be generated for various state changes in the snapshot workflow.

   You can alert on failure, on start, on success, or when the snapshot workflow is aborted.
8. Click Save Policy.

   The new Policy displays on the Snapshot Policies page.

Edit or delete a snapshot policy

You can use to edit or delete existing snapshot policies.

Procedure

1. Select Backup Snapshot Policies in the left navigation bar.

   Existing snapshot policies are shown in a table on the Snapshot Policies page.
2. Click next to a policy and select Edit Configuration or Delete.
3. If you want to edit the selected policy, make the required changes and click Save Policy.

Enable and disable snapshot creation using

For snapshots to be created, HDFS directories must be enabled for snapshots. You cannot specify a directory as part of a snapshot policy unless it has been enabled for snapshots.

Procedure

1. From the Clusters tab, select the HDFS service.
2. Go to the File Browser tab.
3. Go to the directory you want to enable for snapshots.
4. In the File Browser, click the drop-down menu next to the full file path and select Enable Snapshots.

   Note: Once you enable snapshots for a directory, you cannot enable snapshots on any of its subdirectories. Snapshots can be taken only on directories that have snapshots enabled.

   Note: To disable snapshots for a directory that has snapshots enabled, use Disable Snapshots from the drop-down menu specified earlier. If snapshots of the directory exist, they must be deleted before snapshots can be disabled.

Related Information

Enable snapshot creation on a directory

Create snapshots using

You can use to create snapshots on a *snapshottable* directory.

Procedure

1. From the Clusters tab, select the HDFS service.
2. Go to the File Browser tab.
3. Go to the directory for which you want to create the snapshot.

4. Click the drop-down menu next to the full path name and select Take Snapshot.

   The Take Snapshot screen displays.
5. Enter a name for the snapshot.
6. Click OK.

   The Take Snapshot button is present, enabling an immediate snapshot of the directory.
7. To take a snapshot, click Take Snapshot, specify the name of the snapshot, and click Take Snapshot.

   The snapshot is added to the snapshot list. Any snapshots that have been taken are listed by the time at which they were taken, along with their names and a menu button.

Related Information

Create snapshots on a directory

Delete snapshots using

You can use select from a snapshot configured for a directory and delete it.

About this task

Procedure

1. From the Clusters tab, select the HDFS service.
2. Go to the File Browser tab.
3. Go to the directory with the snapshot you want to delete.
4. In the list of snapshots, locate the snapshot you want to delete and click .
5. Select Delete.

Related Information

Create snapshots on a directory

Recover data from a snapshot

Preventing inadvertent deletion of directories

You can prevent inadvertent deletion of important data from your HDFS cluster by marking specific directories as *protected*. Marking a directory as protected prevents its recursive deletion. However, this does not protect against graceful deletion of files under the directory. You can delete the files by moving them to trash.

Procedure

1. Go to the HDFS service.
2. Click the Configuration tab.
3. Set the Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml property as specified:
   - Name: fs.protected.directories
   - Value: Specify a comma-separated list of the directories that you want to mark as *protected*; for example, /user, /data, and so on.
4. Enter a Reason for Change, and then click Save Changes to save the property changes.
5. Restart the cluster.

Note: By default, subdirectories present under directories (fs.protected.directories) are not protected. You must set the dfs.protected.subdirectories.enable parameter to true to protect subdirectories. For example, if you set the fs.protected.directories parameter to true for the parent directory testA, the subdirectory testB under testA(/testA/testB) can be deleted or renamed if dfs.protected.subdirectories.enable is set to false.

Cloudera Runtime 7.3.1

Developing Apache Spark Applications

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction 5

Spark application model 5

Spark execution model 5

Developing and running an Apache Spark WordCount application 6

Using the Spark DataFrame API 9

Building Spark Applications 11
- Best practices for building Apache Spark applications 11
- Building reusable modules in Apache Spark applications 11
- Packaging different versions of libraries with an Apache Spark application 13

Using Spark SQL 13
- SQLContext and HiveContext 14
- Querying files into a DataFrame 15
- Spark SQL example 15
- Interacting with Hive views 17
- Performance and storage considerations for Spark SQL DROP TABLE PURGE 17
- TIMESTAMP compatibility for Parquet files 18
- Accessing Spark SQL through the Spark shell 20

Calling Hive user-defined functions (UDFs) 20

Using Spark Streaming 21
- Spark Streaming and Dynamic Allocation 22
- Spark Streaming Example 22
- Enabling fault-tolerant processing in Spark Streaming 24
- Configuring authentication for long-running Spark Streaming jobs 25
- Building and running a Spark Streaming application 25
- Apache Spark 3 integration with Schema Registry 27
  - Configuration 28
  - Fetching Spark schema by name 29
  - Building and deploying your app 30
  - Running in a Kerberos-enabled cluster 30
  - Unsupported features 31
- Sample pom.xml file for Spark Streaming with Kafka 31

Accessing external storage from Spark.....34
Accessing data stored in Amazon S3 through Spark.....34
Examples of accessing Amazon S3 data from Spark.....35

Accessing Hive from Spark.....38

Accessing HDFS Files from Spark.....38

Accessing ORC Data in Hive Tables.....39
Accessing ORC files from Spark.....39
Predicate push-down optimization.....40
Loading ORC data into DataFrames using predicate push-down.....40
Optimizing queries using partition pruning.....41
Enabling vectorized query execution.....41
Reading Hive ORC tables.....42

Accessing Avro data files from Spark SQL applications.....42

Accessing Parquet files from Spark SQL applications.....44

Using Spark MLlib.....44
Running a Spark MLlib example.....44
Enabling Native Acceleration For MLlib.....45

Using custom libraries with Spark.....46

Using Apache Iceberg with Spark.....46

Introduction

Apache Spark enables you to quickly develop applications and process jobs.

Apache Spark is designed for fast application development and processing. Spark Core is the underlying execution engine; other services, such as Spark SQL, MLlib, and Spark Streaming, are built on top of the Spark Core.

Depending on your use case, you can extend your use of Spark into several domains, including the following:

- Spark DataFrames
- Spark SQL
- Calling Hive user-defined functions from Spark SQL
- Spark Streaming
- Accessing HBase tables, HDFS files, and ORC data (Hive)
- Using custom libraries

Related Information

Apache Spark Quick Start

Apache Spark Overview

Apache Spark Programming Guide

Spark application model

Apache Spark is widely considered to be the successor to MapReduce for general purpose data processing on Apache Hadoop clusters. Like MapReduce applications, each Spark application is a self-contained computation that runs user-supplied code to compute a result. As with MapReduce jobs, Spark applications can use the resources of multiple hosts. However, Spark has many advantages over MapReduce.

In MapReduce, the highest-level unit of computation is a *job*. A job loads data, applies a map function, shuffles it, applies a reduce function, and writes data back out to persistent storage. In Spark, the highest-level unit of computation is an *application*. A Spark application can be used for a single batch job, an interactive session with multiple jobs, or a long-lived server continually satisfying requests. A Spark application can consist of more than just a single map and reduce.

MapReduce starts a process for each task. In contrast, a Spark application can have processes running on its behalf even when it's not running a job. Furthermore, multiple tasks can run within the same executor. Both combine to enable extremely fast task startup time as well as in-memory data storage, resulting in orders of magnitude faster performance over MapReduce.

Spark execution model

Spark application execution involves runtime concepts such as *driver*, *executor*, *task*, *job*, and *stage*. Understanding these concepts is vital for writing fast and resource efficient Spark programs.

At runtime, a Spark application maps to a single *driver* process and a set of *executor* processes distributed across the hosts in a cluster.

The driver process manages the job flow and schedules tasks and is available the entire time the application is running. Typically, this driver process is the same as the client process used to initiate the job, although when run on YARN, the driver can run in the cluster. In interactive mode, the shell itself is the driver process.

The executors are responsible for performing work, in the form of tasks, as well as for storing any data that you cache. Executor lifetime depends on whether dynamic allocation is enabled. An executor has a number of slots for running tasks, and will run many concurrently throughout its lifetime.

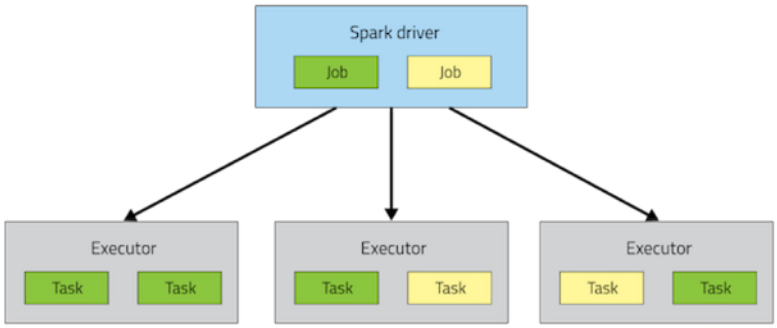

Invoking an action inside a Spark application triggers the launch of a *job* to fulfill it. Spark examines the dataset on which that action depends and formulates an execution plan. The execution plan assembles the dataset transformations into stages. A *stage* is a collection of tasks that run the same code, each on a different subset of the data.

Developing and running an Apache Spark WordCount application

This tutorial describes how to write, compile, and run a simple Spark word count application in two of the languages supported by Spark: Scala and Python. The Scala code was originally developed for a Cloudera tutorial written by Sandy Ryza.

About this task

This example application is an enhanced version of WordCount, the canonical MapReduce example. In this version of WordCount, the goal is to learn the distribution of letters in the most popular words in a corpus. The application:

1. Creates a SparkConf and SparkContext. A Spark application corresponds to an instance of the SparkContext class. When running a shell, the SparkContext is created for you.
2. Gets a word frequency threshold.
3. Reads an input set of text documents.
4. Counts the number of times each word appears.
5. Filters out all words that appear fewer times than the threshold.
6. For the remaining words, counts the number of times each letter occurs.

In MapReduce, this requires two MapReduce applications, as well as persisting the intermediate data to HDFS between them. In Spark, this application requires about 90 percent fewer lines of code than one developed using the MapReduce API.

Procedure

1. Create an empty directory named sparkwordcount in your home directory, and enter it:

```
mkdir $HOME/sparkwordcount
cd $HOME/sparkwordcount
```

2. For the Scala version, create the ./com/cloudera/sparkwordcount subdirectories. For Python, skip this step.

```
mkdir -p com/cloudera/sparkwordcount
```

3. Create the WordCount program in either Scala or Python, using the specified file names and paths:
   - Scala (./com/cloudera/sparkwordcount/SparkWordCount.scala

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf

object SparkWordCount {
  def main(args: Array[String]) {
    // create Spark context with Spark configuration
    val sc = new SparkContext(new SparkConf().setAppName("SparkWordCou
nt"))

    // get threshold
    val threshold = args(1).toInt

    // read in text file and split each document into words
    val tokenized = sc.textFile(args(0)).flatMap(_.split(" "))

    // count the occurrence of each word
    val wordCounts = tokenized.map((_, 1)).reduceByKey(_ + _)

    // filter out words with fewer than threshold occurrences
    val filtered = wordCounts.filter(_._2 >= threshold)

    // count characters
    val charCounts = filtered.flatMap(_._1.toCharArray).map((_, 1)).re
duceByKey(_ + _)

    System.out.println(charCounts.collect().mkString(", "))
  }
}
```

   - Python (./SparkWordCount.py):

```
import sys

from pyspark import SparkContext, SparkConf
if __name__ == "__main__":
  # create Spark context with Spark configuration
  conf = SparkConf().setAppName("SparkWordCount")
  sc = SparkContext(conf=conf)
  # get threshold
  threshold = int(sys.argv[2])

  # read in text file and split each document into words
  tokenized = sc.textFile(sys.argv[1]).flatMap(lambda line: line.split("
 "))

  # count the occurrence of each word
  wordCounts = tokenized.map(lambda word: (word, 1)).reduceByKey(lambda
v1,v2:v1 +v2)

  # filter out words with fewer than threshold occurrences
  filtered = wordCounts.filter(lambda pair:pair[1] >= threshold)

  # count characters
  charCounts = filtered.flatMap(lambda pair:pair[0]).map(lambda c: c).m
ap(lambda c: (c, 1)).reduceByKey(lambda v1,v2:v1 +v2)

  list = charCounts.collect()
  print repr(list)[1:-1]
```

This tutorial uses Maven to compile and package the Scala program. Download the tutorial pom.xml file to the parent $HOME/sparkwordcount directory and modify the sections listed below. For best practices using Maven to build Spark applications, see Building Spark Applications on page 11.

4. In the application pom.xml file, include the Scala tools repo and plugin, and add Scala and Spark as dependencies:

```
<pluginRepositories>
<pluginRepository>
    <id>scala-tools.org</id>
    <name>Scala-tools Maven2 Repository</name>
    <url>http://scala-tools.org/repo-releases</url>
  </pluginRepository>
</pluginRepositories>
```

```
  <build>
    <sourceDirectory>${project.basedir}</sourceDirectory>
    <plugins>
      <plugin>
        <groupId>org.scala-tools</groupId>
        <artifactId>maven-scala-plugin</artifactId>
        <version>2.15.2</version>
        <executions>
          <execution>
            <goals>
              <goal>compile</goal>
              <goal>testCompile</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
```

```
<<dependencies>
  <dependency>
    <groupId>org.scala-lang</groupId>
    <artifactId>scala-library</artifactId>
    <version>2.12.17</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.12</artifactId>
    <version>3.4.1.7.3.1.0-SNAPSHOT</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

5. Make sure that you are in the parent $HOME/sparkwordcount directory, and then generate the application JAR as follows:

```
mvn package
```

This creates sparkwordcount-0.0.1-SNAPSHOT.jar in the target directory.

The input to the application is a large text file in which each line contains all the words in a document, stripped of punctuation. Put an input file in a directory in an S3 bucket that is accessible by your Spark cluster. You can use the tutorial example input file.

6. Run the applications using spark-submit:

- Scala on YARN with threshold 2:

```
spark-submit --class SparkWordCount \
  --master yarn --deploy-mode client --executor-memory 1g \
```

```
  --conf "spark.kerberos.access.hadoopFileSystems=s3a://<BUCKET_NAME>" \
  target/sparkwordcount-0.0.1-SNAPSHOT.jar \
  s3a://<BUCKET_NAME>/<INPUT_FILENAME> 2
```

- Python on YARN with threshold 2:

```
spark-submit --master yarn --deploy-mode client --executor-memory 1g \
  --conf "spark.kerberos.access.hadoopFileSystems=s3a://<BUCKET_NAME>" \
  SparkWordCount.py s3a://<BUCKET_NAME>/<INPUT_FILENAME> 2
```

If you used the example input file, the output is similar to the following:

Scala:

```
19/07/24 09:18:57 INFO scheduler.DAGScheduler: Job 0 finished: collect at
 SparkWordCount.scala:25, took 10.188651 s
(p,2), (t,2), (b,1), (h,1), (n,4), (f,1), (v,1), (r,2), (l,1), (e,6), (a
,4), (i,1), (u,1), (o,2), (c,1)
```

Python:

```
19/07/24 09:23:55 INFO scheduler.DAGScheduler: Job 0 finished: collect a
t /home/user/sparkwordcount/SparkWordCount.py:26, took 11.762150 s
(u'a', 4), (u'c', 1), (u'e', 6), (u'i', 1), (u'o', 2), (u'u', 1), (u'b',
1), (u'f', 1), (u'h', 1), (u'l', 1), (u'n', 4), (u'p', 2), (u'r', 2), (u
't', 2), (u'v', 1)
```

Using the Spark DataFrame API

About this task

A DataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R or in the Python pandas library. You can construct DataFrames from a wide array of sources, including structured data files, Apache Hive tables, and existing Spark resilient distributed datasets (RDD). The Spark DataFrame API is available in Scala, Java, Python, and R.

This section provides examples of DataFrame API use.

To list JSON file contents as a DataFrame:

1. Upload the people.txt and people.json example files to your object store:

```
hdfs dfs -put people.txt people.json s3a://<BUCKET_NAME>/
```

2. Launch the Spark shell:

```
spark-shell --conf "spark.kerberos.access.hadoopFileSystems=s3a:
//<BUCKET_NAME>/"
```

3. At the Spark shell, type the following:

```
scala> val df = spark.read.format("json").load("s3a://<BUCKET_NAME>/peo
ple.json")
```

4. Using df.show, display the contents of the DataFrame:

```
scala> df.show
+----+-------+
| age|   name|
```

```
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

The following examples use Scala to access DataFrame df defined in the previous subsection:

```
// Select all rows, but increment age by 1
scala> df.select(df("name"), df("age") + 1).show()
+-------+---------+
|   name|(age + 1)|
+-------+---------+
|Michael|     null|
|   Andy|       31|
| Justin|       20|
+-------+---------+

// Select people older than 21
scala> df.filter(df("age") > 21).show()
+---+----+
|age|name|
+---+----+
| 30|Andy|
+---+----+

// Count people by age
scala> df.groupBy("age").count().show()
+----+-----+

| age|count|
+----+-----+
|  19|    1|
|null|    1|
|  30|    1|
+----+-----+
```

The following example uses the DataFrame API to specify a schema for people.txt, and then retrieves names from a temporary table associated with the schema:

```
val people = sc.textFile("s3a://<BUCKET_NAME>/people.txt")
val schemaString = "name age"
import org.apache.spark.sql.types.{StructType,StructField,StringType}
import org.apache.spark.sql.Row

val schema = StructType(schemaString.split(" ").map(fieldName => StructField
(fieldName, StringType, true)))
val rowRDD = people.map(_.split(",")).map(p => Row(p(0), p(1).trim))
val peopleDataFrame = spark.createDataFrame(rowRDD, schema)

peopleDataFrame.createOrReplaceTempView("people")
val results = spark.sql("SELECT name FROM people")
results.map(t => "Name: " + t(0)).collect().foreach(println)
```

This produces output similar to the following:

```
Name: Michael
Name: Andy
Name: Justin
```

Building Spark Applications

You can use Apache Maven to build Spark applications developed using Java and Scala.

Best practices for building Apache Spark applications

Follow these best practices when building Apache Spark Scala and Java applications:

- Refrain from using withColumn in chain, loop, and calling it multiple times in a single query. Doing so may cause performance issues. To avoid issues, use select() with multiple columns at once. See the Apache Spark API reference linked below for more information.
- Compile your applications against the same version of Spark that you are running.
- Build a single assembly JAR ("Uber" JAR) that includes all dependencies. In Maven, add the Maven assembly plug-in to build a JAR containing all dependencies:

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <configuration>
    <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
    </descriptorRefs>
  </configuration>
  <executions>
    <execution>
      <id>make-assembly</id>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

  This plug-in manages the merge procedure for all available JAR files during the build. Exclude Spark, Hadoop, and Kafka classes from the assembly JAR, because they are already available on the cluster and contained in the runtime classpath. In Maven, specify Spark, Hadoop, and Kafka dependencies with scope provided. For example:

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.12</artifactId>
  <version>3.4.1.7.3.1.0-SNAPSHOT</version>
  <scope>provided</scope>
</dependency>
```

Related Information

Apache Spark API Reference

Building reusable modules in Apache Spark applications

Using existing Scala and Java classes inside the Spark shell requires an effective deployment procedure and dependency management. For simple and reliable reuse of Scala and Java classes and complete third-party libraries, you can use a *module*, which is a self-contained artifact created by Maven. This module can be shared by multiple users. This topic shows how to use Maven to create a module containing all dependencies.

Create a Maven Project

1. Use Maven to generate the project directory:

```
$ mvn archetype:generate -DgroupId=COM.MYCOMPANY -DartifactId=MYLIBRARY \
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

Download and Deploy Third-Party Libraries

1. Prepare a location for all third-party libraries that are not available through Maven Central but are required for the project:

```
mkdir libs
cd libs
```

2. Download the required artifacts.
3. Use Maven to deploy the library JAR.
4. Add the library to the dependencies section of the POM file.
5. Repeat steps 2-4 for each library. For example, to add the JIDT library:

   a. Download and decompress the zip file:

```
curl http://lizier.me/joseph/software/jidt/download.php?file=infodynamic
s-dist-1.3.zip > infodynamics-dist-1.3.zip
unzip infodynamics-dist-1.3.zip
```

   b. Deploy the library JAR:

```
$ mvn deploy:deploy-file \
-Durl=file:///$HOME/.m2/repository -Dfile=libs/infodynamics.jar \
-DgroupId=org.jlizier.infodynamics -DartifactId=infodynamics -Dpackag
ing=jar -Dversion=1.3
```

   c. Add the library to the dependencies section of the POM file:

```
<dependency>
  <groupId>org.jlizier.infodynamics</groupId>
  <artifactId>infodynamics</artifactId>
  <version>1.3</version>
</dependency>
```

6. Add the Maven assembly plug-in to the plugins section in the pom.xml file.
7. Package the library JARs in a module:

```
mvn clean package
```

Run and Test the Spark Module

1. Run the Spark shell, providing the module JAR in the --jars option:

```
spark-shell --jars target/mylibrary-1.0-SNAPSHOT-jar-with-dependencies.jar
```

2. In the Environment tab of the Spark Web UI application (http://*DRIVER_HOST*:4040/environment/), validate that the spark.jars property contains the library. For example:

Environment

Runtime Information

Name	Value
Java Home	/usr/java/jdk1.7.0_67-cloudera/jre
Java Version	1.7.0_67 (Oracle Corporation)
Scala Version	version 2.10.4

Spark Properties

Name	Value
spark.serializer	org.apache.spark.serializer.KryoSerializer
spark.driver.host	172.26.26.126
spark.eventLog.enabled	true
spark.driver.port	39021
spark.shuffle.service.enabled	true
spark.driver.extraLibraryPath	/opt/cloudera/parcels/CDH-5.5.1-1.cdh5.5.1.p0.11/lib/hadoop/lib/native
spark.repl.class.uri	http://172.26.26.126:52069
spark.jars	file:/var/lib/hadoop-hdfs/mylibrary/target/mylibrary-1.0-SNAPSHOT-jar-with-dependencies.jar

3. In the Spark shell, test that you can import some of the required Java classes from the third-party library. For example, if you use the JIDT library, import MatrixUtils:

```
$ spark-shell
...
scala> import infodynamics.utils.MatrixUtils;
```

Packaging different versions of libraries with an Apache Spark application

To use a version of a library in your application that is different than the version of that library that is shipped with Spark, use the Apache Maven Shade Plugin. This process is technically known as relocation, and often referred to as shading.

See Relocating Classes for an example.

Using Spark SQL

This section provides information about using Spark SQL.

There are two ways to interact with Spark SQL:

- Interactive access using the Spark shell (see "Accessing Spark SQL through the Spark Shell" in this guide).
- From an application, operating through one of the following two APIs and the HiveServer2:
  - JDBC, using your own Java code or the Beeline JDBC client
  - ODBC, through the Simba ODBC driver

The following diagram illustrates the access process, depending on whether you are using the Spark shell or business intelligence (BI) application:

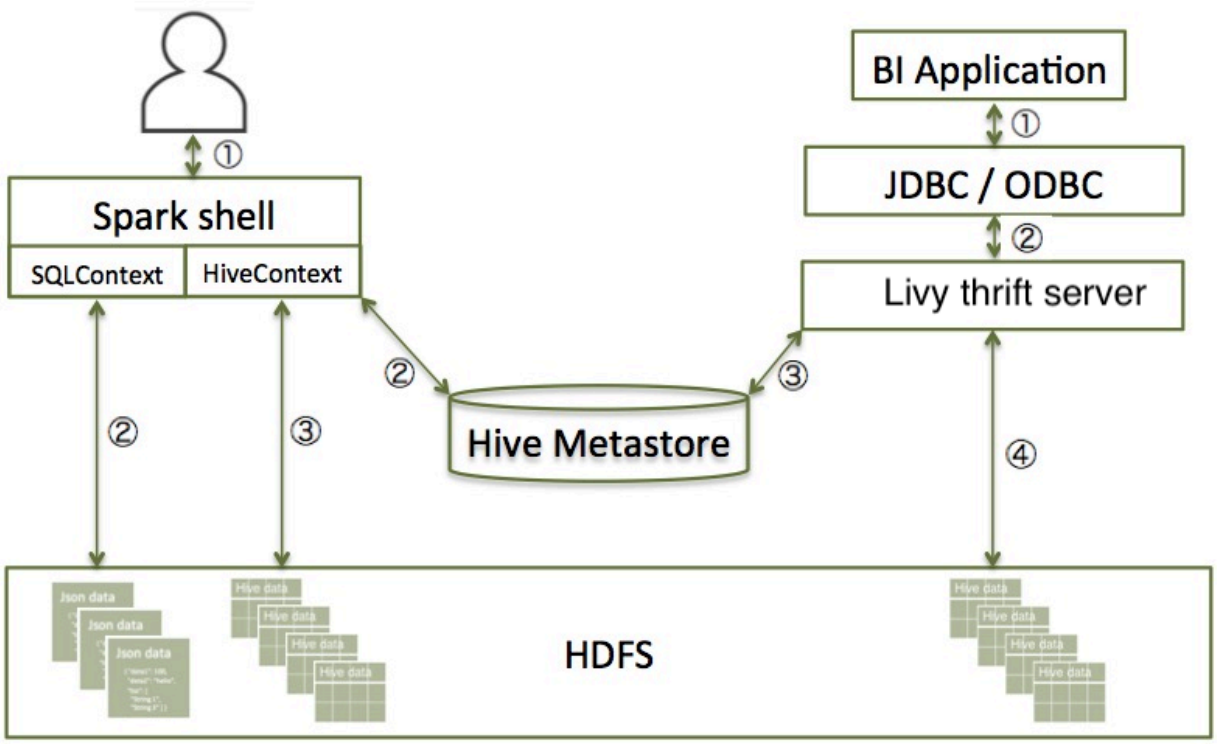

The following subsections describe how to access Spark SQL through the Spark shell, and through JDBC and ODBC.

Related Information

Connecting to the Apache Livy Thrift Server

Beeline Command Line Shell

SQLContext and HiveContext

Beginning in Spark 2.0, all Spark functionality, including Spark SQL, can be accessed through the SparkSessions class, available as spark when you launch spark-shell. You can create a DataFrame from an RDD, a Hive table, or a data source.

Note: Hive and Impala tables and related SQL syntax are interchangeable in most respects. Because Spark uses the underlying Hive infrastructure, with Spark SQL you write DDL statements, DML statements, and queries using the Hive syntax. For interactive query performance, you can access the same tables through Impala using impala-shell or the Impala JDBC and ODBC interfaces.

If you use spark-submit, use code like the following at the start of the program (this example uses Python):

```
from pyspark import SparkContext, HiveContext
sc = SparkContext(appName = "test")
sqlContext = HiveContext(sc)
```

The host from which the Spark application is submitted or on which spark-shell or pyspark runs must have a Hive gateway role defined in and client configurations deployed.

When a Spark job accesses a Hive view, Spark must have privileges to read the data files in the underlying Hive tables. Currently, Spark cannot use fine-grained privileges based on the columns or the WHERE clause in the view definition. If Spark does not have the required privileges on the underlying data files, a SparkSQL query against the view returns an empty result set, rather than an error.

Querying files into a DataFrame

If you have data files that are outside of a Hive or Impala table, you can use SQL to directly read JSON or Parquet files into a DataFrame.

JSON

```
df = sqlContext.sql("SELECT * FROM json.`INPUT DIR`")
```

Parquet

```
df = sqlContext.sql("SELECT * FROM parquet.`INPUT DIR`")
```

Spark SQL example

This example demonstrates how to use spark.sql to create and load two tables and select rows from the tables into two DataFrames. The next steps use the DataFrame API to filter the rows for salaries greater than 150,000 from one of the tables and shows the resulting DataFrame. Then the two DataFrames are joined to create a third DataFrame. Finally the new DataFrame is saved to a Hive table.

1. Copy the Hue sample_07.csv and sample_08.csv files to your object store in a location accessible by the Spark cluster:

   ```
   hdfs dfs -put /opt/cloudera/parcels/CDH/lib/hue/apps/beeswax/data/sample
   _0* s3a://<BUCKET_NAME>/
   ```

2. Launch spark-shell:

   ```
   spark-shell --conf "spark.kerberos.access.hadoopFileSystems=s3a:
   //<BUCKET_NAME>"
   ```

3. Create Hive tables sample_07 and sample_08:

   ```
   scala> spark.sql("CREATE EXTERNAL TABLE sample_07 (code string,description
    string,total_emp int,salary int) ROW FORMAT DELIMITED FIELDS TERMINATED B
   Y '\t' STORED AS TextFile LOCATION 's3a://<BUCKET_NAME>/s07/'")
   scala> spark.sql("CREATE EXTERNAL TABLE sample_08 (code string,descript
   ion string,total_emp int,salary int) ROW FORMAT DELIMITED FIELDS TERMINA
   TED BY '\t' STORED AS TextFile LOCATION 's3a://<BUCKET_NAME>/s08/'")
   ```

4. In another session, launch Beeline:

   ```
   beeline -u "jdbc:hive2://<HIVESERVER2_HOST>:10001/default;principal=hive/
   _HOST@CLOUDERA.SITE;transportMode=http;httpPath=cliservice"
   ```

5. In Beeline, show the Hive tables:

   ```
   0: jdbc:hive2://hs2.cloudera.site:> show tables;
   +------------+--+
   |  tab_name  |
   +------------+--+
   | sample_07  |
   | sample_08  |
   +------------+--+
   ```

6. In the Spark shell, load the data from the CSV files into the tables:

```
scala> spark.sql("LOAD DATA INPATH 's3a://<BUCKET_NAME>/sample_07.csv'
OVERWRITE INTO TABLE sample_07")
scala> spark.sql("LOAD DATA INPATH 's3a://<BUCKET_NAME>/sample_08.csv'
OVERWRITE INTO TABLE sample_08")
```

7. Create DataFrames containing the contents of the sample_07 and sample_08 tables:

```
scala> val df_07 = spark.sql("SELECT * from sample_07")
scala> val df_08 = spark.sql("SELECT * from sample_08")
```

8. Show all rows in df_07 with salary greater than 150,000:

```
scala> df_07.filter(df_07("salary") > 150000).show()
```

The output should be:

```
+-------+--------------------+---------+------+
|   code|         description|total_emp|salary|
+-------+--------------------+---------+------+
|11-1011|    Chief executives|   299160|151370|
|29-1022|Oral and maxillof...|     5040|178440|
|29-1023|       Orthodontists|     5350|185340|
|29-1024|     Prosthodontists|      380|169360|
|29-1061|   Anesthesiologists|    31030|192780|
|29-1062|Family and genera...|   113250|153640|
|29-1063| Internists, general|    46260|167270|
|29-1064|Obstetricians and...|    21340|183600|
|29-1067|            Surgeons|    50260|191410|
|29-1069|Physicians and su...|   237400|155150|
+-------+--------------------+---------+------+
```

9. Create the DataFrame df_09 by joining df_07 and df_08, retaining only the code and description columns.

```
scala> val df_09 = df_07.join(df_08, df_07("code") === df_08("code")).se
lect(df_07.col("code"),df_07.col("description"))
scala> df_09.orderBy($"code".asc).show()
```

The new DataFrame looks like:

```
+-------+--------------------+
|   code|         description|
+-------+--------------------+
|00-0000|     All Occupations|
|11-0000|Management occupa...|
|11-1011|    Chief executives|
|11-1021|General and opera...|
|11-1031|         Legislators|
|11-2011|Advertising and p...|
|11-2021|  Marketing managers|
|11-2022|      Sales managers|
|11-2031|Public relations ...|
|11-3011|Administrative se...|
|11-3021|Computer and info...|
|11-3031|  Financial managers|
|11-3041|Compensation and ...|
|11-3042|Training and deve...|
|11-3049|Human resources m...|
|11-3051|Industrial produc...|
|11-3061| Purchasing managers|
|11-3071|Transportation, s...|
|11-9011|Farm, ranch, and ...|
```

```
|11-9012|Farmers and ranchers|
+-------+--------------------+
```

10. Save DataFrame df_09 as the Hive table sample_09:

```
scala> df_09.write.option("path","s3a://<BUCKET_NAME>/s09/").saveAsTable(
"sample_09")
```

11. In Beeline, show the Hive tables:

```
0: jdbc:hive2://hs2.cloudera.site:> show tables;
+------------+--+
|  tab_name  |
+------------+--+
| sample_07  |
| sample_08  |
| sample_09  |
+------------+--+
```

Here is an equivalent program in Python, that you could submit using spark-submit:

```
from pyspark import SparkContext, SparkConf, HiveContext

if __name__ == "__main__":

  # create Spark context with Spark configuration
  conf = SparkConf().setAppName("Data Frame Join")
  sc = SparkContext(conf=conf)
  sqlContext = HiveContext(sc)
  df_07 = sqlContext.sql("SELECT * from sample_07")
  df_07.filter(df_07.salary > 150000).show()
  df_08 = sqlContext.sql("SELECT * from sample_08")
  tbls = sqlContext.sql("show tables")
  tbls.show()
  df_09 = df_07.join(df_08, df_07.code == df_08.code).select(df_07.code,d
f_07.description)
  df_09.show()
  df_09.write.saveAsTable("sample_09")
  tbls = sqlContext.sql("show tables")
  tbls.show()
```

Note: Instead of displaying the tables using Beeline, the show tables query is run using the Spark SQL API.

Interacting with Hive views

When a Spark job accesses a Hive view, Spark must have privileges to read the data files in the underlying Hive tables. Currently, Spark cannot use fine-grained privileges based on the columns or the WHERE clause in the view definition. If Spark does not have the required privileges on the underlying data files, a SparkSQL query against the view returns an empty result set, rather than an error.

Performance and storage considerations for Spark SQL DROP TABLE PURGE

The PURGE clause in the Hive DROP TABLE statement causes the underlying data files to be removed immediately, without being transferred into a temporary holding area (such as the HDFS trashcan).

Although the PURGE clause is recognized by the Spark SQL DROP TABLE statement, this clause is currently not passed along to the Hive statement that performs the “drop table” operation behind the scenes. Therefore, if you know the PURGE behavior is important in your application for performance, storage, or security reasons, do the DROP TABLE directly in Hive, for example through the `beeline` shell, rather than through Spark SQL.

The immediate deletion aspect of the PURGE clause could be significant in cases such as:

- If the cluster is running low on storage space and it is important to free space immediately, rather than waiting for the HDFS trashcan to be periodically emptied.
- If the underlying data files reside on the Amazon S3 filesystem. Moving files to the HDFS trashcan from S3 involves physically copying the files, meaning that the default DROP TABLE behavior on S3 involves significant performance overhead.
- If the underlying data files contain sensitive information and it is important to remove them entirely, rather than leaving them to be cleaned up by the periodic emptying of the trashcan.

TIMESTAMP compatibility for Parquet files

Impala stores and retrieves the TIMESTAMP values verbatim, with no adjustment for the time zone. When writing Parquet files, Hive and Spark SQL both normalize all TIMESTAMP values to the UTC time zone. During a query, Spark SQL assumes that all TIMESTAMP values have been normalized this way and reflect dates and times in the UTC time zone. Therefore, Spark SQL adjusts the retrieved date/time values to reflect the local time zone of the server. SPARK-12297 introduces a configuration setting, spark.sql.parquet.int96TimestampConversion=true, that you can set to change the interpretation of TIMESTAMP values read from Parquet files that were written by Impala, to match the Impala behavior.

Note: This compatibility workaround only applies to Parquet files created by Impala and has no effect on Parquet files created by Hive, Spark or other Java components.

The following sequence of examples show how, by default, TIMESTAMP values written to a Parquet table by an Apache Impala SQL statement are interpreted differently when queried by Spark SQL, and vice versa.

The initial Parquet table is created by Impala, and some TIMESTAMP values are written to it by Impala, representing midnight of one day, noon of another day, and an early afternoon time from the Pacific Daylight Savings time zone. (The second and third tables are created with the same structure and file format, for use in subsequent examples.)

```
[localhost:21000] > create table parquet_table(t timestamp) stored as parqu
et;
[localhost:21000] > create table parquet_table2 like parquet_table stored as
 parquet;
[localhost:21000] > create table parquet_table3 like parquet_table stored as
 parquet;
[localhost:21000] > select now();
+-------------------------------+
| now()                         |
+-------------------------------+
| 2018-03-23 14:07:01.057912000 |
+-------------------------------+
[localhost:21000] > insert into parquet_table
                  > values ('2018-03-23'), (now()), ('2000-01-01 12:00:00');
[localhost:21000] > select t from parquet_table order by t;
+-------------------------------+
| t                             |
+-------------------------------+
| 2000-01-01 12:00:00           |
| 2018-03-23 00:00:00           |
| 2018-03-23 14:08:54.617197000 |
+-------------------------------+
```

By default, when this table is queried through the Spark SQL using `spark-shell`, the values are interpreted and displayed differently. The time values differ from the Impala result set by either 4 or 5 hours, depending on whether the dates are during the Daylight Savings period or not.

```
scala> spark.sql("select t from jdr.parquet_table order by t").show(truncat
e=false);
+--------------------------+
|t                         |
+--------------------------+
|2000-01-01 04:00:00.0     |
|2018-03-22 17:00:00.0     |
|2018-03-23 07:08:54.617197|
+--------------------------+
```

Running the same Spark SQL query with the configuration setting spark.sql.parquet.int96TimestampConversion=true applied makes the results the same as from Impala:

```
$ spark-shell --conf spark.sql.parquet.int96TimestampConversion=true
...
scala> spark.sql("select t from jdr.parquet_table order by t").show(trunca
te=false);
+--------------------------+
|t                         |
+--------------------------+
|2000-01-01 12:00:00.0     |
|2018-03-23 00:00:00.0     |
|2018-03-23 14:08:54.617197|
+--------------------------+
```

The compatibility considerations also apply in the reverse direction. The following examples show the same Parquet values as before, this time being written to tables through Spark SQL.

```
$ spark-shell
scala> spark.sql("insert into jdr.parquet_table2 select t from jdr.parque
t_table");
scala> spark.sql("select t from jdr.parquet_table2 order by t").show(trun
cate=false);
+--------------------------+
|t                         |
+--------------------------+
|2000-01-01 04:00:00.0     |
|2018-03-22 17:00:00.0     |
|2018-03-23 07:08:54.617197|
+--------------------------+
```

Again, the configuration setting spark.sql.parquet.int96TimestampConversion=true means that the values are both read and written in a way that is interoperable with Impala:

```
$ spark-shell --conf spark.sql.parquet.int96TimestampConversion=true
...
scala> spark.sql("insert into jdr.parquet_table3 select t from jdr.parquet
_table");
scala> spark.sql("select t from jdr.parquet_table3 order by t").show(trunc
ate=false);
+--------------------------+
```

```
|t                          |
+---------------------------+
|2000-01-01 12:00:00.0      |
|2018-03-23 00:00:00.0      |
|2018-03-23 14:08:54.617197|
+---------------------------+
```

Accessing Spark SQL through the Spark shell

Use the following steps to access Spark SQL using the Spark shell.

The following sample command launches the Spark shell on a YARN cluster:

spark-shell --num-executors 1 --executor-memory 512m --master yarn-client

To read data directly from the file system, construct a SQLContext. For an example that uses SQLContext and the Spark DataFrame API to access a JSON file, see Using the Spark DataFrame API on page 9.

To read data by interacting with the Hive Metastore, construct a HiveContext instance (HiveContext extends SQLContext). For an example of the use of HiveContext (instantiated as val sqlContext), see "Accessing ORC Files from Spark" in this guide.

Calling Hive user-defined functions (UDFs)

Use the following steps to call Hive user-defined functions.

About this task

You can call built-in Hive UDFs, UDAFs, and UDTFs and custom UDFs from Spark SQL applications if the functions are available in the standard Hive JAR file. When using Hive UDFs, use HiveContext (not SQLContext).

Using Built-in UDFs

The following interactive example reads and writes to HDFS under Hive directories, using hiveContext and the built-in collect_list(col) UDF. The collect_list(col) UDF returns a list of objects with duplicates. In a production environment, this type of operation runs under an account with appropriate HDFS permissions; the following example uses hdfs user.

1. Launch the Spark Shell on a YARN cluster:

```
spark-shell --num-executors 2 --executor-memory 512m --master yarn-client
```

2. Invoke the Hive collect_list UDF:

```
scala> spark.sql("from TestTable SELECT key, collect_list(value) group by
 key order by key").collect.foreach(println)
```

Using Custom UDFs

You can register custom functions in Python, Java, or Scala, and then use them within SQL statements.

When using a custom UDF, ensure that the .jar file for your UDF is included with your application, or use the --jars command-line option to specify the file.

The following example uses a custom Hive UDF. This example uses the more limited SQLContext, instead of HiveContext.

1. Launch spark-shell with hive-udf.jar as its parameter:

```
spark-shell --jars <path-to-your-hive-udf>.jar
```

2. From spark-shell, define a function:

```
scala> spark.sql("""create temporary function balance as 'org.package.hi
veudf.BalanceFromRechargesAndOrders'"""");
```

3. From spark-shell, invoke your UDF:

```
scala> spark.sql("""
create table recharges_with_balance_array as
select
  reseller_id,
  phone_number,
  phone_credit_id,
  date_recharge,
  phone_credit_value,
  balance(orders,'date_order', 'order_value', reseller_id, date_recharge,
phone_credit_value) as balance
from orders
""");
```

Using Spark Streaming

This section provides information on using Spark streaming.

Before you begin

Before running a Spark Streaming application, Spark and Kafka must be deployed on the cluster.

Unless you are running a job that is part of the Spark examples package installed by , you must add or retrieve the spark-streaming-kafka .jar file and associated .jar files before running your Spark job.

About this task

Spark Streaming is an extension of core Spark that enables scalable, high-throughput, fault-tolerant processing of data streams. Spark Streaming receives input data streams called Discretized Streams (DStreams), which are essentially a continuous series of RDDs. DStreams can be created either from sources such as Kafka, Flume, and Kinesis, or by applying operations on other DStreams.

You can process data using complex algorithms expressed with high-level functions like map, reduce, join, and wind ow, and send results to file systems, databases, and live dashboards.

For detailed information on Spark Streaming, see Spark Streaming Programming Guide in the Apache Spark documentation.

Spark Streaming receives live input data streams and divides the data into batches, which are then processed by the Spark engine to generate the final stream of results in batches:

Apache Spark has built-in support for the Apache Kafka 08 API. If you want to access a Kafka 0.10 cluster using new Kafka 0.10 APIs (such as wire encryption support) from Spark streaming jobs, the spark-kafka-0-10-connector package supports a Kafka 0.10 connector for Spark streaming. See the package readme file for additional documentation.

The remainder of this subsection describes general steps for developers using Spark Streaming with Kafka on a Kerberos-enabled cluster; it includes a sample pom.xml file for Spark Streaming applications with Kafka. For additional examples, see the Apache GitHub example repositories for Scala, Java, and Python.

Important: Dynamic Resource Allocation does not work with Spark Streaming.

Related Information

Apache Streaming Programming Guide
Apache GitHub Scala Streaming Examples
Apache GitHub Java Streaming Examples
Apache GitHub Python Streaming Examples

Spark Streaming and Dynamic Allocation

Dynamic allocation is enabled by default, which means that executors are removed when idle. Dynamic allocation conflicts with Spark Streaming operations.

In Spark Streaming, data comes in batches, and executors run whenever data is available. If the executor idle timeout is less than the batch duration, executors are constantly added and removed. However, if the executor idle timeout is greater than the batch duration, executors are never removed. Therefore, recommends that you disable dynamic allocation by setting spark.dynamicAllocation.enabled to false when running streaming applications.

Spark Streaming Example

This example uses Kafka to deliver a stream of words to a Python word count program.

1. Create a Kafka topic wordcounttopic:

```
kafka-topics --create --zookeeper ZOOKEEPER_SERVER:2181 --topic wordcountt
opic --partitions 1 --replication-factor 1
```

2. Create a Kafka word count Python program adapted from the Spark Streaming example kafka_wordcount.py. This version divides the input stream into batches of 10 seconds and counts the words in each batch:

```
from __future__ import print_function

import sys

from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils
```

```
if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: kafka_wordcount.py <zk> <topic>", file=sys.stderr)
        sys.exit(-1)

    sc = SparkContext(appName="PythonStreamingKafkaWordCount")
    ssc = StreamingContext(sc, 10)

    zkQuorum, topic = sys.argv[1:]
    kvs = KafkaUtils.createStream(ssc, zkQuorum, "spark-streaming-consumer
", {topic: 1})
    lines = kvs.map(lambda x: x[1])
    counts = lines.flatMap(lambda line: line.split(" ")).map(lambda word:
 (word, 1)).reduceByKey(lambda a, b: a+b)
    counts.pprint()

    ssc.start()
    ssc.awaitTermination()
```

3. Submit the application using spark-submit with dynamic allocation disabled and specifying your ZooKeeper server and topic. To run locally, you must specify at least two worker threads: one to receive and one to process data:

```
spark-submit --master yarn --deploy-mode client --conf "spark.dynamicAll
ocation.enabled=false" --jars SPARK_HOME/lib/spark-examples.jar kafka_wo
rdcount.py ZOOKEEPER_SERVER:2181 wordcounttopic
```

In a deployment, *SPARK_HOME* defaults to /opt/cloudera/parcels/CDH/lib/spark. The shells are also available from /usr/bin.

4. In another window, start a Kafka producer that publishes to wordcounttopic:

```
kafka-console-producer --broker-list KAFKA_BROKER:9092 --topic wordcountt
opic
```

5. In the producer window, type the following:

```
hello
hello
hello
hello
hello
hello
gb
gb
gb
gb
gb
gb
```

Depending on how fast you type, in the Spark Streaming application window you will see output like:

```
-------------------------------------------
Time: 2016-01-06 14:18:00
-------------------------------------------
(u'hello', 6)
(u'gb', 2)

-------------------------------------------
Time: 2016-01-06 14:18:10
-------------------------------------------
(u'gb', 4)
```

Enabling fault-tolerant processing in Spark Streaming

Important: Spark Streaming checkpoints do not work across Spark upgrades or application upgrades. If you are upgrading Spark or your streaming application, you must clear the checkpoint directory.

For long-running Spark Streaming jobs, make sure to configure the maximum allowed failures in a given time period. For example, to allow 3 failures per hour, set the following parameters (in spark-defaults.conf or when submitting the job):

```
spark.yarn.maxAppAttempts=3
spark.yarn.am.attemptFailuresValidityInterval=1h
```

If the driver host for a Spark Streaming application fails, it can lose data that has been received but not yet processed. To ensure that no data is lost, you can use Spark Streaming recovery. Recovery uses a combination of a write-ahead log and checkpoints. Spark writes incoming data to HDFS as it is received and uses this data to recover state if a failure occurs.

To enable Spark Streaming recovery:

1. Set the spark.streaming.receiver.writeAheadLog.enable parameter to true in the SparkConf object.
2. Create a StreamingContext instance using this SparkConf, and specify a checkpoint directory.
3. Use the getOrCreate method in StreamingContext to either create a new context or recover from an old context from the checkpoint directory:

```
from __future__ import print_function

import sys

from pyspark import SparkContext, SparkConf
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils

checkpoint = "hdfs://ns1/user/systest/checkpoint"
# Function to create and setup a new StreamingContext
def functionToCreateContext():

  sparkConf = SparkConf()
  sparkConf.set("spark.streaming.receiver.writeAheadLog.enable", "true")
  sc = SparkContext(appName="PythonStreamingKafkaWordCount",conf=sparkConf
)
  ssc = StreamingContext(sc, 10)

  zkQuorum, topic = sys.argv[1:]
  kvs = KafkaUtils.createStream(ssc, zkQuorum, "spark-streaming-consume
r", {topic: 1})
  lines = kvs.map(lambda x: x[1])
  counts = lines.flatMap(lambda line: line.split(" ")).map(lambda word: (w
ord, 1)).reduceByKey(lambda a, b: a+b)
  counts.pprint()

  ssc.checkpoint(checkpoint)   # set checkpoint directory
  return ssc

if __name__ == "__main__":
  if len(sys.argv) != 3:
    print("Usage: kafka_wordcount.py <zk> <topic>", file=sys.stderr)
    exit(-1)
  ssc = StreamingContext.getOrCreate(checkpoint, lambda: functionToCrea
teContext())
  ssc.start()
```

```
ssc.awaitTermination()
```

For more information, see Checkpointing in the Apache Spark documentation.

To prevent data loss if a receiver fails, receivers must be able to replay data from the original data sources if required.

- The Kafka receiver automatically replays if the spark.streaming.receiver.writeAheadLog.enable parameter is set to true.
- The receiverless Direct Kafka DStream does not require the spark.streaming.receiver.writeAheadLog.enable parameter and can function without data loss, even without Streaming recovery.
- Both Flume receivers packaged with Spark replay the data automatically on receiver failure.

Configuring authentication for long-running Spark Streaming jobs

Long-running applications such as Spark Streaming jobs must be able to write data continuously, which means that the user may need to delegate tokens possibly beyond the default lifetime. This workload type requires passing Kerberos principal and keytab to the spark-submit script using the --principal and --keytab parameters.

Note: Only YARN cluster mode is supported with the --principal and --keytab parameters.

The keytab is copied to the host running the ApplicationMaster, and the Kerberos login is renewed periodically by using the principal and keytab to generate the required delegation tokens needed for HDFS.

Note: For secure distribution of the keytab to the ApplicationMaster host, the cluster should be configured for TLS/SSL communication for YARN and HDFS encryption.

Building and running a Spark Streaming application

Use the following steps to build and run a Spark streaming job for .

Depending on your compilation and build processes, one or more of the following tasks might be required before running a Spark Streaming job:

- If you are using maven as a compile tool:
  1. Add the repository to your pom.xml file:

     ```
     <repository>
         <id>cloudera</id>
         <name>Cloudera Repository</name>
         <url>https://repository.cloudera.com/artifactory/cloudera-repos/</ur
     l>
     </repository>
     ```

  2. Specify the version number for Spark streaming Kafka and streaming dependencies to your pom.xml file:

     ```
     <dependency>
         <groupId>org.apache.spark</groupId>
         <artifactId>spark-streaming-kafka-0-10_2.12</artifactId>
         <version>3.4.1.7.3.1.0-1970</version>
     </dependency>

     <dependency>
         <groupId>org.apache.spark</groupId>
         <artifactId>spark-streaming_2.12</artifactId>
         <version>3.4.1.7.3.1.0-19</version>
         <scope>provided</scope>
     ```

```
</dependency>
```

Note that the correct version number includes the Spark version and the version.

3. (Optional) If you prefer to pack an uber .jar rather than use the default ("provided"), add the maven-shade-plug in to your pom.xml file:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-shade-plugin</artifactId>
    <version>2.3</version>
    <executions>
        <execution>
            <phase>package</phase>
            <goals>
                <goal>shade</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <filters>
            <filter>
                <artifact>*:*</artifact>
                <excludes>
                    <exclude>META-INF/*.SF</exclude>
                    <exclude>META-INF/*.DSA</exclude>
                    <exclude>META-INF/*.RSA</exclude>
                </excludes>
            </filter>
        </filters>
        <finalName>uber-${project.artifactId}-${project.version}</finalN
ame>
    </configuration>
</plugin>
```

- Instructions for submitting your job depend on whether you used an uber .jar file or not:
  - If you kept the default .jar scope and you can access an external network, use --packages to download dependencies in the runtime library:

```
spark-submit --master yarn-client \
    --num-executors 1 \
    --packages org.apache.spark:spark-streaming-kafka-0-10_2.12:3.4.1.7.
3.1.0-197 \
    --repositories http://repo.hortonworks.com/content/repositories/rele
ases/ \
    --class <user-main-class> \
    <user-application.jar> \
    <user arg lists>
```

    The artifact and repository locations should be the same as specified in your pom.xml file.
  - If you packed the .jar file into an uber .jar, submit the .jar file in the same way as you would a regular Spark application:

```
spark-submit --master yarn-client \
    --num-executors 1 \
    --class <user-main-class> \
    <user-uber-application.jar> \
    <user arg lists>
```

1. Select or create a user account to be used as principal.

   This should not be the kafka or spark service account.

2. Generate a keytab for the user.
3. Create a Java Authentication and Authorization Service (JAAS) login configuration file: for example, key.conf.
4. Add configuration settings that specify the user keytab.

   The keytab and configuration files are distributed using YARN local resources. Because they reside in the current directory of the Spark YARN container, you should specify the location as ./v.keytab.

   The following example specifies keytab location ./v.keytab for principal vagrant@example.com:

```
KafkaClient {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   keyTab="./v.keytab"
   storeKey=true
   useTicketCache=false
   serviceName="kafka"
   principal="vagrant@EXAMPLE.COM";
};
```

5. In your spark-submit command, pass the JAAS configuration file and keytab as local resource files, using the --files option, and specify the JAAS configuration file options to the JVM options specified for the driver and executor:

```
spark-submit \
    --files key.conf#key.conf,v.keytab#v.keytab \
    --driver-java-options "-Djava.security.auth.login.config=./key.conf" \
    --conf "spark.executor.extraJavaOptions=-Djava.security.auth.login.c
onfig=./key.conf" \
...
```

6. Pass any relevant Kafka security options to your streaming application.

   For example, the KafkaWordCount example accepts PLAINTEXTSASL as the last option in the command line:

```
KafkaWordCount /vagrant/spark-examples.jar c6402:2181 abc ts 1 PLAINTEXT
SASL
```

Apache Spark 3 integration with Schema Registry

Apache Spark 3 integrated with Schema Registry provides a library to leverage Schema Registry for managing Spark schemas and to serialize and/or de-serialize messages in Spark data sources and sinks.

Note: The Apache Spark 3 integration with Schema Registry is available in from version 7.2.18 SP2.

Running the example programs

The examples illustrate the API usage and how to integrate with Schema Registry. The examples can be run from IDE (for example, Intellij) by specifying a master URL or by using spark3-submit.

```
spark3-submit --jars /opt/cloudera/parcels/CDH/lib/spark-schema-registry-for
-spark3/spark-schema-registry-for-spark3_2.12-jar-with-dependencies.jar \
--class com.hortonworks.spark.registry.examples.classname \
/opt/cloudera/parcels/CDH/lib/spark-schema-registry-for-spark3/examples/sp
ark-schema-registry-for-spark3-examples_2.12.jar [***SCHEMA-REGISTRY-URL***]
 \
bootstrap-servers input-topic output-topic checkpoint-location
```

Using the APIs

Typically in a Spark application you define the Spark schema for the data you are going to process:

```
// the schema for truck events
val schema = StructType(Seq(
  StructField("driverId", IntegerType, nullable = false),
  StructField("truckId", IntegerType, nullable = false),
  StructField("miles", LongType, nullable = false),
  StructField("eventType", StringType, nullable = false),
  ...
  ...
)

// read Json string messages from the data source
val messages = spark
      .readStream
      .format(...)
      .option(...)
      .load()

// parse the messages using the above schema and do further operations
val df = messages
      .select(from_json($"value".cast("string"), schema).alias("value"))
      ...
// project (driverId, truckId, miles) for the events where miles > 300
val filtered = df.select($"value.driverId", $"value.truckId", $"value.miles"
)
      .where("value.miles > 300")
```

However, this approach is not practical because the schema information is tightly coupled with the code. The code needs to be changed when the schema changes, and there is no ability to share or reuse the schema between the message producers and the applications that consume the messages.

Using Schema Registry is a better solution because it enables you to manage different versions of the schema and define compatibility policies.

Configuration

The Schema Registry integration comes as a utility method which can be imported into the scope.

```
import com.hortonworks.spark.registry.util._
```

Before invoking the APIs, you need to define an implicit SchemaRegistryConfig which will be passed to the APIs. The main configuration parameter is the schema registry URL.

```
// the schema registry client config
val config = Map[String, Object]("[***SCHEMA.REGISTRY.URL***]" -> schemaR
egistryUrl)
// the schema registry config that will be implicitly passed
implicit val srConfig:SchemaRegistryConfig = SchemaRegistryConfig(config)
```

SSL configuration

SchemaRegistryConfig expects the following SSL configuration properties:

```
"schema.registry.client.ssl.protocol" -> "SSL",
"schema.registry.client.ssl.trustStoreType" -> "JKS",
```

```
"schema.registry.client.ssl.trustStorePath" -> "/var/lib/cloudera-scm-agent/
agent-cert/cm-auto-global_truststore.jks",
"schema.registry.client.ssl.trustStorePassword" ->
 "[***CHANGEMECLIENTPWD***]"
```

Fetching Spark schema by name

The API supports fetching the Schema Registry schema as a Spark schema.

- sparkSchema(schemaName: String)

  Returns the spark schema corresponding to the latest version of schema defined in the Schema Registry.
- sparkSchema(schemaName: String, version: Int)

  Returns the spark schema corresponding to the given version of schema defined in the Schema Registry.

Using the Schema Registry integration, the example previously shown can be simplified, as there is no need to explicitly specify the Spark schema in the code:

```
// retrieve the translated "Spark schema" by specifying the schema registry
schema name
val schema = sparkSchema(name)

// parse the messages using the above schema and do further operations
val df = messages
         .select(from_json($"value".cast("string"), schema).alias("value"))
         ...

// project (driverId, truckId, miles) for the events where miles > 300
val filtered = df.select($"value.driverId", $"value.truckId", $"value.miles
")
                  .where("value.miles > 300")
```

Serializing messages using Schema Registry

The following method can be used to serialize the messages from Spark to Schema Registry binary format using schema registry serializers.

- to_sr(data: Column, schemaName: String, topLevelRecordName: String, namespace: String)

  Converts a Spark column data to binary format of Schema Registry. This looks up a Schema Registry schema for the schemaName that matches the input and automatically registers a new schema, if not found. The topoLevelRecordName and namespace are optional and will be mapped to Avro top level record name and record namespace.

De-serializing messages using Schema Registry

The following methods can be used to de-serialize Schema Registry serialized messages into Spark columns.

- from_sr(data: Column, schemaName: String)

  Converts Schema Registry binary format to Spark column, using the latest version of the schema.
- from_sr(data: Column, schemaName: String, version: Int)

  Converts Schema Registry binary format to Spark column using the given Schema Registry schema name and version.

Serialization - deserialization example

The following is an example that uses the from_sr to de-serialize Schema Registry formatted messages into Spark, transforms and serializes it back to Schema Registry format using to_sr, and writes to a data sink.

This example assumes Spark Structured Streaming use cases, but it should work well for the non-streaming use cases as well (read and write).

```
// Read schema registry formatted messages and deserialize to spark columns.
val df = messages
      .select(from_sr($"value", topic).alias("message"))
// project (driverId, truckId, miles) for the events where miles > 300
val filtered = df.select($"message.driverId", $"message.truckId", $"message.
miles")
      .where("message.miles > 300")
// write the output as schema registry serialized bytes to a sink
// should produce events like {"driverId":14,"truckId":25,"miles":373}
val query = filtered
      .select(to_sr(struct($"*"), outSchemaName).alias("value"))
      .writeStream
      .format(..)
      .start()
```

The output schema outSchemaName is automatically published to the Schema Registry if it does not exist.

Building and deploying your app

Add a Maven dependency in your project to make use of the library and build your application JAR file:

```
<dependency>
    <groupId>com.hortonworks</groupId>
    <artifactId>spark-schema-registry-for-spark3_2.12</artifactId>
    <version>version</version>
 </dependency>
```

Once the application JAR file is built, deploy it by adding the dependency in spark3-submit using --packages:

```
spark3-submit --packages com.hortonworks:spark-schema-registry-for-spark3_2.
12:version \
--conf spark.jars.repositories=[***HTTPS://REPOSITORY.EXAMPLE.COM***] \
--class YourApp \
your-application-jar \
args ...
```

Make sure the package is published in a local or online available repository.

If the package is not published to an available repository, or your Spark application cannot access external networks, you can use an uber JAR file instead:

```
spark3-submit --master [***MASTER URL****] \
--jars /opt/cloudera/parcels/CDH/lib/spark-schema-registry-for-spark3/spark-
schema-registry-for-spark3_2.12-jar-with-dependencies.jar \
--class YourApp \
your-application-jar \
args ...
```

Running in a Kerberos-enabled cluster

The library works in a Kerberos setup, where Spark and Schema Registry has been deployed on a Kerberos-enabled cluster.

To configure, set up the appropriate JAAS config for RegistryClient (and KafkaClient, if the Spark data source or sink is Kafka).

As an example, to run the SchemaRegistryAvroExample in a Kerberos setup, follow these steps:

1. Create a keytab (for example, app.keytab) with the login user and principal you want to run the application.
2. Create an app_jaas.conf file and specify the keytab and principal created in Step 1.

   If deploying to YARN, the keytab and conf files will be distributed as YARN local resources. They will be placed in the current directory of the Spark YARN container, and the location needs to be specified as ./app.keytab.

```
RegistryClient {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    keyTab="./app.keytab"
    storeKey=true
    useTicketCache=false
    principal="[***PRINCIPAL***]";
};

KafkaClient {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   keyTab="./app.keytab"
   storeKey=true
   useTicketCache=false
   serviceName="kafka"
   principal="[***PRINCIPAL***]";
};
```

3. Provide the required ACLs for the kafka topics (in-topic, out-topic) for the principal.
4. Use spark3-submit to pass the JAAS configuration file with extraJavaOptions. (And also as local resource files in YARN cluster mode.)

```
spark3-submit --master yarn --deploy-mode cluster \
    --keytab app.keytab --principal [***PRINCIPAL***] \
    --files app_jaas.conf#app_jaas.conf,app.keytab#app.keytab \
    --jars /opt/cloudera/parcels/CDH/lib/spark-schema-registry-for-spark3/
spark-schema-registry-for-spark3_2.12-jar-with-dependencies.jar \
    --conf "spark.executor.extraJavaOptions=-Djava.security.auth.login.c
onfig=./app_jaas.conf" \
    --conf "spark.driver.extraJavaOptions=-Djava.security.auth.login.co
nfig=./app_jaas.conf" \
    --class com.hortonworks.spark.registry.examples.SchemaRegistryAvroEx
ample \
/opt/cloudera/parcels/CDH/lib/spark-schema-registry-for-spark3/examples/s
park-schema-registry-for-spark3-examples_2.12.jar \
    [***SCHEMA-REGISTRY-URL***] bootstrap-server in-topic out-topic che
ckpoint-dir SASL_PLAINTEXT
```

Unsupported features

Apache Spark 3 integration with Schema Registry is not supported in pyspark.

Sample pom.xml file for Spark Streaming with Kafka

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
```

```
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
             http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>test</groupId>
    <artifactId>spark-kafka</artifactId>
    <version>1.0-SNAPSHOT</version>

    <repositories>
        <repository>
            <id>hortonworks</id>
            <name>hortonworks repo</name>
            <url>http://repo.hortonworks.com/content/repositories/releases/
</url>
        </repository>
    </repositories>

    <dependencies>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-streaming-kafka-0-10_2.12</artifactId>
            <version>3.4.1.7.3.1.0-197</version>
        </dependency>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-streaming_2.12</artifactId>
            <version>3.4.1.7.3.1.0-19</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
        <defaultGoal>package</defaultGoal>
        <resources>
            <resource>
                <directory>src/main/resources</directory>
                <filtering>true</filtering>
            </resource>
            <resource>
                <directory>src/test/resources</directory>
                <filtering>true</filtering>
            </resource>
        </resources>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <configuration>
                    <encoding>UTF-8</encoding>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>copy-resources</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>net.alchim31.maven</groupId>
                <artifactId>scala-maven-plugin</artifactId>
                <version>3.2.0</version>
                <configuration>
                    <recompileMode>incremental</recompileMode>
                    <args>
```

```
                    <arg>-target:jvm-1.8</arg>
                </args>
                <javacArgs>
                    <javacArg>-source</javacArg>
                    <javacArg>1.8</javacArg>
                    <javacArg>-target</javacArg>
                    <javacArg>1.8</javacArg>
                </javacArgs>
            </configuration>
            <executions>
                <execution>
                    <id>scala-compile</id>
                    <phase>process-resources</phase>
                    <goals>
                        <goal>compile</goal>
                    </goals>
                </execution>
                <execution>
                    <id>scala-test-compile</id>
                    <phase>process-test-resources</phase>
                    <goals>
                        <goal>testCompile</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>1.7</source>
                <target>1.7</target>
            </configuration>

            <executions>
                <execution>
                    <phase>compile</phase>
                    <goals>
                        <goal>compile</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>2.3</version>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>shade</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
                <filters>
                    <filter>
                        <artifact>*:*</artifact>
                        <excludes>
                            <exclude>META-INF/*.SF</exclude>
                            <exclude>META-INF/*.DSA</exclude>
                            <exclude>META-INF/*.RSA</exclude>
```

```
                            </excludes>
                        </filter>
                    </filters>
                    <finalName>uber-${project.artifactId}-${project.version
}</finalName>
                </configuration>
            </plugin>

        </plugins>

    </build>
</project>
```

Accessing external storage from Spark

Spark can access all storage sources supported by Hadoop, including a local file system, HDFS, HBase, Amazon S3, and Microsoft ADLS.

Spark supports many file types, including text files, RCFile, SequenceFile, Hadoop InputFormat, Avro, Parquet, and compression of all supported files.

For developer information about working with external storage, see External Datasets in the upstream Apache Spark *RDD Programming Guide*.

Accessing data stored in Amazon S3 through Spark

Important: components writing data to S3 are constrained by the inherent limitation of Amazon S3 known as “eventual consistency”. For more information, see Data Storage Considerations.

To access data stored in Amazon S3 from Spark applications, use Hadoop file APIs (SparkContext.hadoopFile, Java HadoopRDD.saveAsHadoopFile, SparkContext.newAPIHadoopRDD, and JavaHadoopRDD.saveAsNewAPIHadoop File) for reading and writing RDDs, providing URLs of the form s3a://*BUCKET_NAME*/path/to/file. You can read and write Spark SQL DataFrames using the Data Source API.

Make sure that your environment is configured to allow access to the buckets you need. You must also configure the spark.yarn.access.hadoopFileSystems parameter to include the buckets you need to access. You can do this using the Spark client configuration, or at runtime as a command line parameter.

For example:

Client configuration (/etc/spark/conf/spark-defaults.conf)

```
spark.yarn.access.hadoopFileSystems=s3a://bucket1,s3a://bucket2
```

spark-shell

```
spark-shell --conf "spark.yarn.access.hadoopFileSystems=s3a://bu
cket1,s3a://bucket2" ...
```

spark-submit

```
spark-submit --conf "spark.yarn.access.hadoopFileSystems=s3a://b
ucket1,s3a://bucket2" ...
```

Examples of accessing Amazon S3 data from Spark

The following examples demonstrate basic patterns of accessing data in S3 using Spark. The examples show the setup steps, application code, and input and output files located in S3.

Reading and Writing Text Files From and To Amazon S3

Run a word count application on a file stored in Amazon S3 (sonnets.txt in this example):

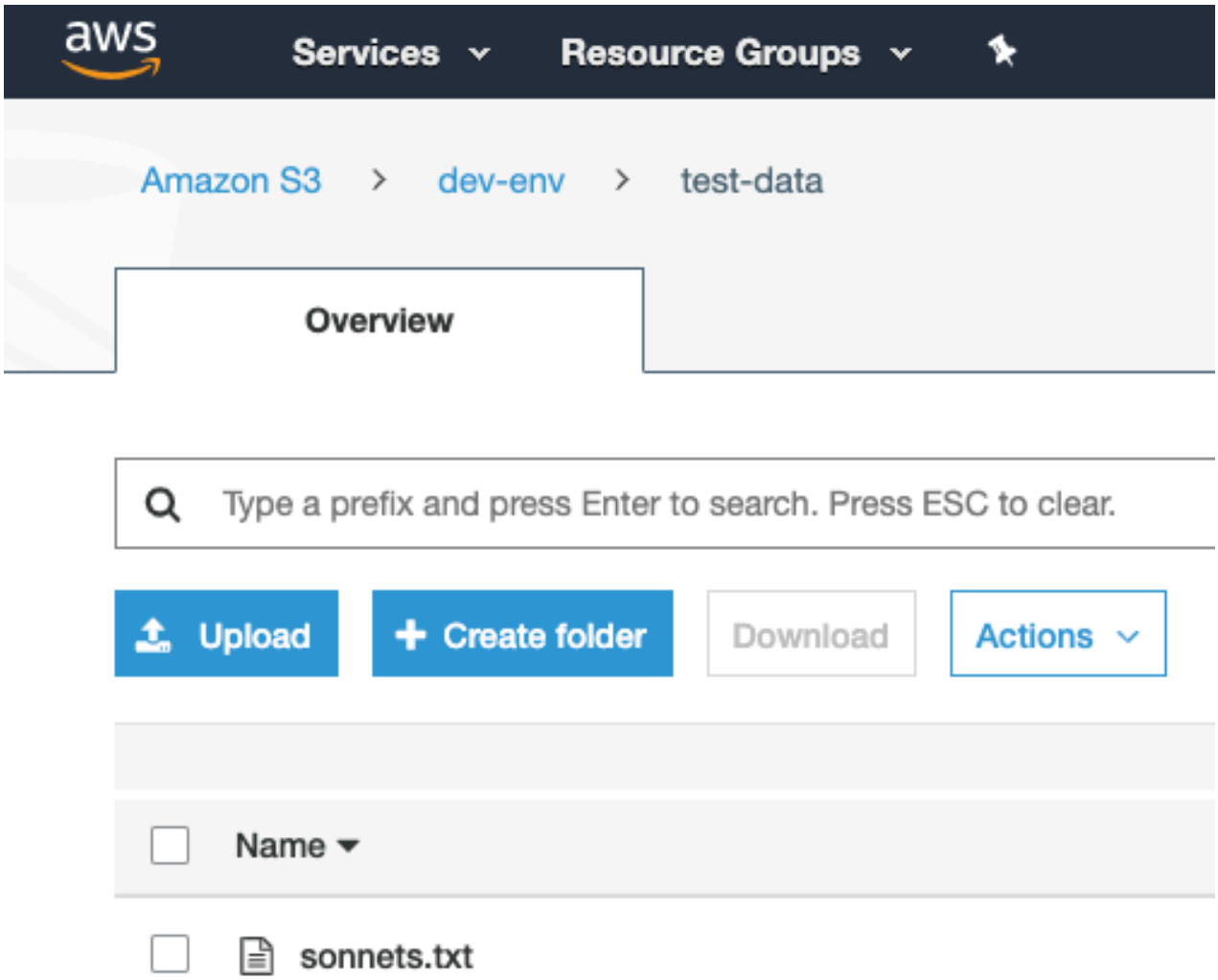

Scala

```
val sonnets = sc.textFile("s3a://dev-env/test-data/sonnets.txt")
val counts = sonnets.flatMap(line => line.split(" ")).map(word =>
 (word, 1)).reduceByKey(_ + _)
counts.saveAsTextFile("s3a://dev-env/test-data/sonnets-wordcoun
t")
```

Python

```
sonnets = sc.textFile("s3a://dev-env/test-data/sonnets.txt")
counts = sonnets.flatMap(lambda line: line.split(" ")).map(lambda
 word: (word, 1)).reduceByKey(lambda v1,v2: v1 + v2)
counts.saveAsTextFile("s3a://dev-env/test-data/sonnets-wordco
unt")
```

Yielding the output:

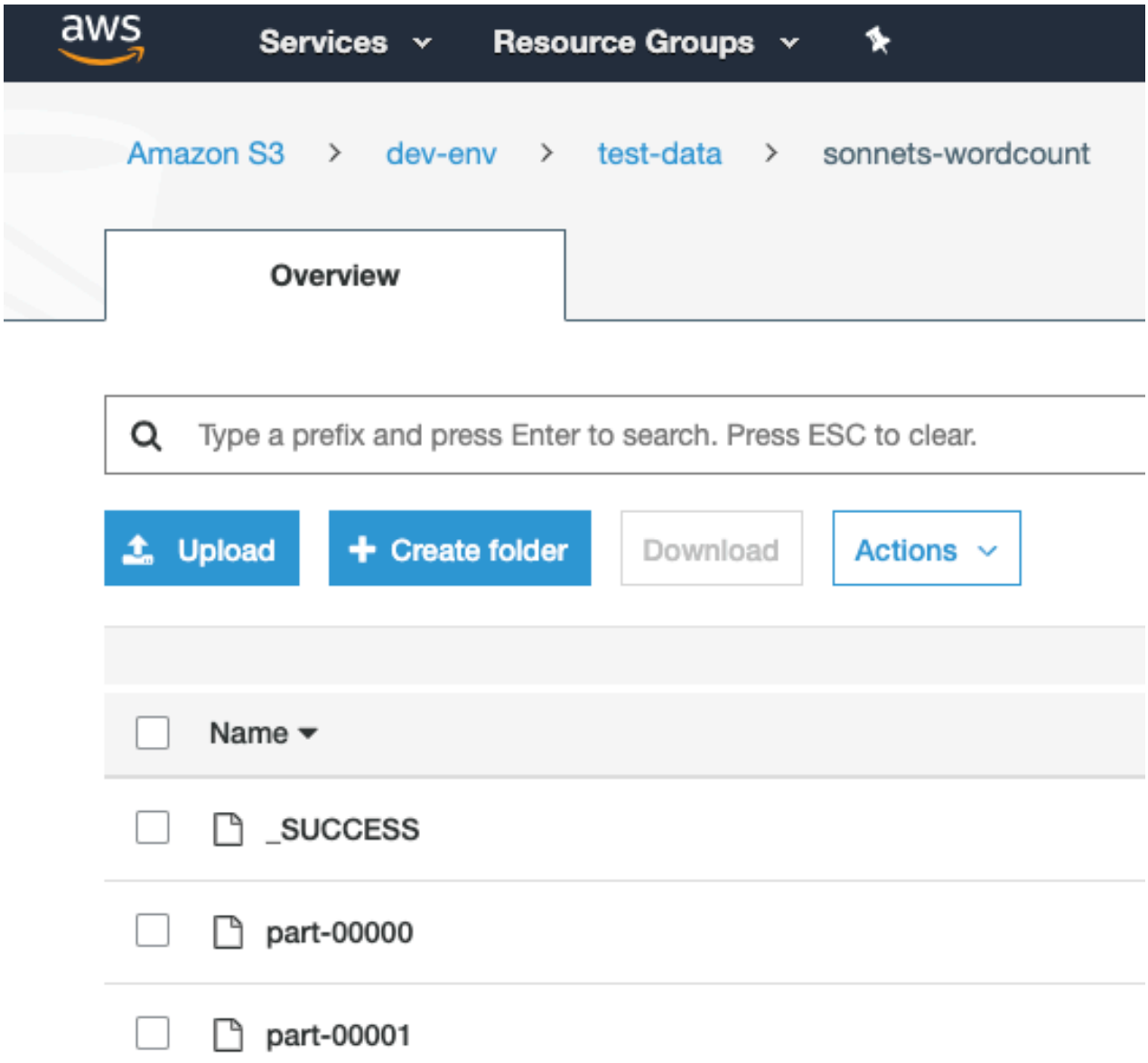

Reading and Writing Data Sources From and To Amazon S3

The following example illustrates how to read a text file from Amazon S3 into an RDD, convert the RDD to a DataFrame, and then use the Data Source API to write the DataFrame into a Parquet file on Amazon S3:

1. Read a text file in Amazon S3:

```
val sample_data = sc.textFile("s3a://dev-env-data/test-data/sample_data.
csv")
```

2. Map lines into columns:

```
import org.apache.spark.sql.Row
val rdd_sample = sample_data.map(_.split('\t')).map(e # Row(e(0), e(1),
e(2).trim.toInt, e(3).trim.toInt))
```

3. Create a schema and apply to the RDD to create a DataFrame:

```
scala> import org.apache.spark.sql.types.{StructType, StructField, Strin
gType, IntegerType};
scala> val schema = StructType(Array(
  StructField("code",StringType,false),
  StructField("description",StringType,false),
  StructField("total_emp",IntegerType,false),
  StructField("salary",IntegerType,false)))
scala> val df_sample = spark.createDataFrame(rdd_sample,schema)
```

4. Write DataFrame to a Parquet file:

```
df_sample.write.parquet("s3a://dev-env-data/test-data/sample_data-parque
t")
```

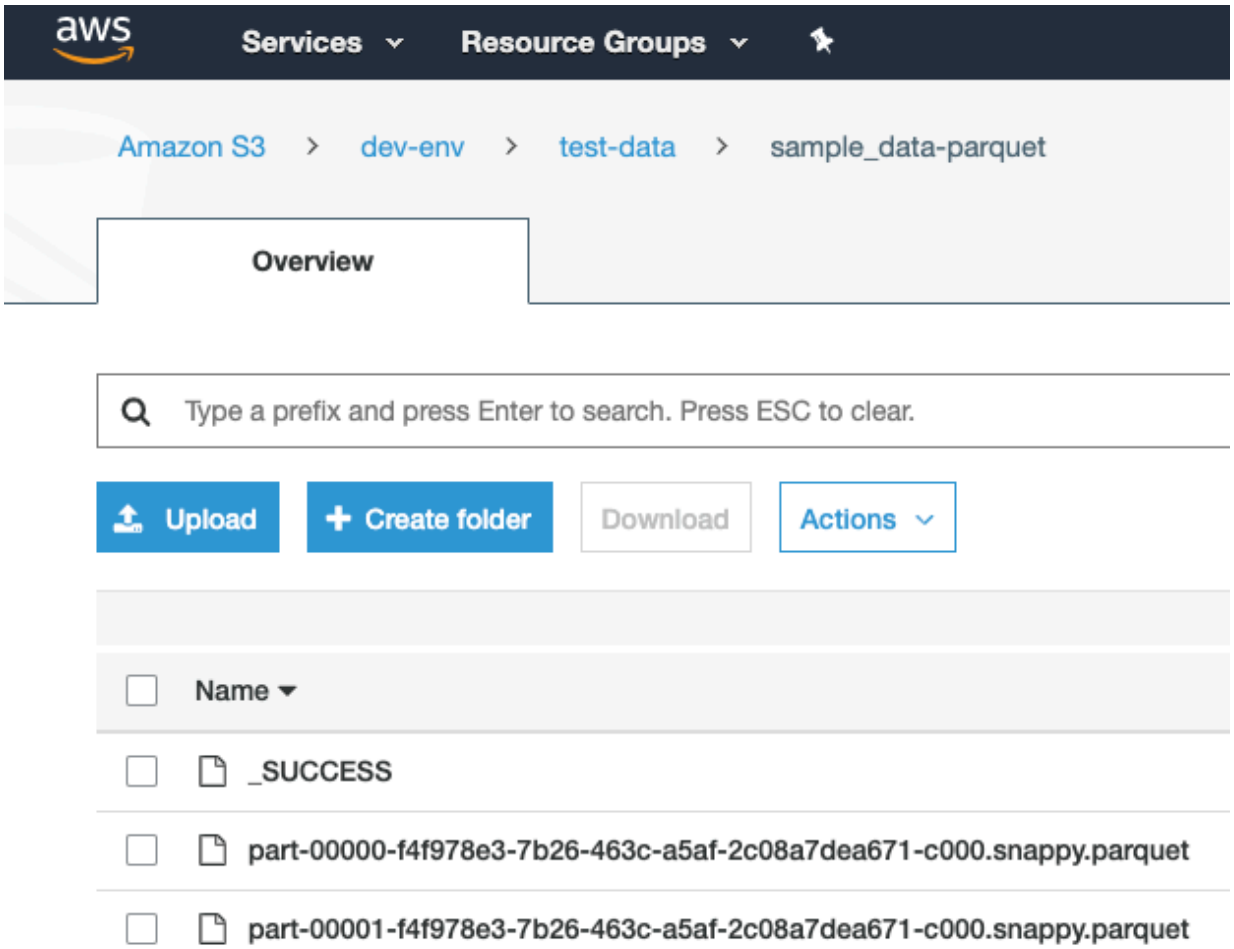

The files are compressed with the default snappy compression.

Important: If your environment does not have a direct internet connection, then you must add an HTTP proxy:

```
--conf spark.executor.extraJavaOptions="-Dhttps.proxyHost=<myproxy.m
ynet.tld>
-Dhttps.proxyPort=<myProxyPort>" --conf spark.driver.extraJavaOptions
="-Dhttps.proxyHost=<myproxy.mynet.tld>
-Dhttps.proxyPort=<myProxyPort>"
```

Accessing Hive from Spark

The host from which the Spark application is submitted or on which spark-shell or pyspark runs must have a Hive gateway role defined in and client configurations deployed.

When a Spark job accesses a Hive view, Spark must have privileges to read the data files in the underlying Hive tables. Currently, Spark cannot use fine-grained privileges based on the columns or the WHERE clause in the view definition. If Spark does not have the required privileges on the underlying data files, a SparkSQL query against the view returns an empty result set, rather than an error.

Accessing HDFS Files from Spark

This section contains information on running Spark jobs over HDFS data.

Specifying Compression

To add a compression library to Spark, you can use the --jars option. For an example, see "Adding Libraries to Spark" in this guide.

To save a Spark RDD to HDFS in compressed format, use code similar to the following (the example uses the GZip algorithm):

```
rdd.saveAsHadoopFile("/tmp/spark_compressed",
                     "org.apache.hadoop.mapred.TextOutputFormat",
                     compressionCodecClass="org.apache.hadoop.io.compress.
GzipCodec")
```

For more information about supported compression algorithms, see "Configuring HDFS Compression" in the HDP Data Storage guide.

Accessing HDFS from PySpark

When accessing an HDFS file from PySpark, you must set HADOOP_CONF_DIR in an environment variable, as in the following example:

```
$ export HADOOP_CONF_DIR=/etc/hadoop/conf
$ pyspark
$ >>>lines = sc.textFile("hdfs://namenode.example.com:8020/tmp/PySparkTest/f
ile-01")
.......
```

If HADOOP_CONF_DIR is not set properly, you might see an error similar to the following:

```
2016-08-22 00:27:06,046|t1.machine|INFO|1580|140672245782272|MainThread|Py4J
JavaError: An error occurred while calling z:org.apache.spark.api.python.Pyt
honRDD.collectAndServe.
2016-08-22 00:27:06,047|t1.machine|INFO|1580|140672245782272|MainThread|: or
g.apache.hadoop.security.AccessControlException: SIMPLE authentication is no
t enabled.  Available:[TOKEN, KERBEROS]
2016-08-22 00:27:06,047|t1.machine|INFO|1580|140672245782272|MainThread|at s
un.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
2016-08-22 00:27:06,047|t1.machine|INFO|1580|140672245782272|MainThread|at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAcces
sorImpl.java:57)
2016-08-22 00:27:06,048|t1.machine|INFO|1580|140672245782272|MainThread|at
```

```
{code}
```

Accessing ORC Data in Hive Tables

Apache Spark in supports the Optimized Row Columnar (ORC) file format, a self-describing, type-aware, column-based file format that is one of the primary file formats supported in Apache Hive.

ORC reduces I/O overhead by accessing only the columns that are required for the current query. It requires significantly fewer seek operations because all columns within a single group of row data (known as a "stripe") are stored together on disk.

Spark ORC data source supports ACID transactions, snapshot isolation, built-in indexes, and complex data types (such as array, map, and struct), and provides read and write access to ORC files. It leverages the Spark SQL Catalyst engine for common optimizations such as column pruning, predicate push-down, and partition pruning.

This subsection has several examples of Spark ORC integration, showing how ORC optimizations are applied to user programs.

Related Information

ORC File Format

Apache Hive ACID Transactions

Accessing ORC files from Spark

Use the following steps to access ORC files from Apache Spark.

About this task

To start using ORC, you can define a SparkSession instance:

```
import org.apache.spark.sql.SparkSession
val spark = SparkSession.builder().getOrCreate()
import spark.implicits._
```

The following example uses data structures to demonstrate working with complex types. The Person struct data type has a name, an age, and a sequence of contacts, which are themselves defined by names and phone numbers.

Procedure

1. Define Contact and Person data structures:

   ```
   case class Contact(name: String, phone: String)
   case class Person(name: String, age: Int, contacts: Seq[Contact])
   ```

2. Create 100 Person records:

   ```
   val records = (1 to 100).map { i =>;
     Person(s"name_$i", i, (0 to 1).map { m => Contact(s"contact_$m", s"ph
   one_$m") })
   }
   ```

   In the physical file, these records are saved in columnar format. When accessing ORC files through the DataFrame API, you see rows.

3. To write person records as ORC files to a directory named “people”, you can use the following command:

   ```
   records.toDF().write.format("orc").save("people")
   ```

4. Read the objects back:

```
val people = sqlContext.read.format("orc").load("people.json")
```

5. For reuse in future operations, register the new "people" directory as temporary table “people”:

```
people.createOrReplaceTempView("people")
```

6. After you register the temporary table “people”, you can query columns from the underlying table:

```
spark.sql("SELECT name FROM people WHERE age < 15").count()
```

Results

In this example the physical table scan loads only columns name and age at runtime, without reading the contacts column from the file system. This improves read performance.

You can also use Spark DataFrameReader and DataFrameWriter methods to access ORC files.

Related Information

Apache Spark DataFrameReader Methods
Apache Spark DataFrameWriter Methods

Predicate push-down optimization

The columnar nature of the ORC format helps avoid reading unnecessary columns, but it is still possible to read unnecessary rows. The example in this subsection reads all rows in which the age value is between 0 and 100, even though the query requested rows in which the age value is less than 15 ("...WHERE age < 15"). Such full table scanning is an expensive operation.

ORC avoids this type of overhead by using predicate push-down, with three levels of built-in indexes within each file: file level, stripe level, and row level:

- File-level and stripe-level statistics are in the file footer, making it easy to determine if the rest of the file must be read.
- Row-level indexes include column statistics for each row group and position, for finding the start of the row group.

ORC uses these indexes to move the filter operation to the data loading phase by reading only data that potentially includes required rows.

This combination of predicate push-down with columnar storage reduces disk I/O significantly, especially for larger datasets in which I/O bandwidth becomes the main bottleneck to performance.

ORC predicate push-down is enabled by default in Spark SQL.

Loading ORC data into DataFrames using predicate push-down

DataFrames are similar to Spark RDDs but have higher-level semantics built into their operators. This allows optimization to be pushed down to the underlying query engine.

Here is the Scala API version of the SELECT query used in the previous section, using the DataFrame API:

```
val spark = SparkSession.builder().getOrCreate()
val people = spark.read.format("orc").load("peoplePartitioned")
people.filter(people("age") < 15).select("name").show()
```

DataFrames are not limited to Scala. There is a Java API and a Python API binding:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
people = spark.read.format("orc").load("peoplePartitioned")
people.filter(people.age < 15).select("name").show()
```

Optimizing queries using partition pruning

When predicate push-down optimization is not applicable—for example, if all stripes contain records that match the predicate condition—a query with a WHERE clause might need to read the entire data set. This becomes a bottleneck over a large table. Partition pruning is another optimization method; it exploits query semantics to avoid reading large amounts of data unnecessarily.

Partition pruning is possible when data within a table is split across multiple logical partitions. Each partition corresponds to a particular value of a partition column and is stored as a subdirectory within the table root directory on HDFS. Where applicable, only the required partitions (subdirectories) of a table are queried, thereby avoiding unnecessary I/O.

Spark supports saving data in a partitioned layout seamlessly, through the partitionBy method available during data source write operations. To partition the "people" table by the “age” column, you can use the following command:

```
people.write.format("orc").partitionBy("age").save("peoplePartitioned")
```

As a result, records are automatically partitioned by the age field and then saved into different directories: for example, peoplePartitioned/age=1/, peoplePartitioned/age=2/, and so on.

After partitioning the data, subsequent queries can omit large amounts of I/O when the partition column is referenced in predicates. For example, the following query automatically locates and loads the file under peoplePartitioned/age=20/and omits all others:

```
val peoplePartitioned = spark.read.format("orc").load("peoplePartitioned")
peoplePartitioned.createOrReplaceTempView("peoplePartitioned")
spark.sql("SELECT * FROM peoplePartitioned WHERE age = 20")
```

Enabling vectorized query execution

About this task

Vectorized query execution is a feature that greatly reduces the CPU usage for typical query operations such as scans, filters, aggregates, and joins. Vectorization is also implemented for the ORC format. Spark also uses Whole Stage Codegen and this vectorization (for Parquet) since Spark 2.0.

Use the following steps to implement the new ORC format and enable vectorization for ORC files with SparkSQL.

Procedure

1. In the , go to Data Hub Clusters.
2. Find and select the cluster you want to configure.
3. Click the link for the URL.
4. Go to Clusters <Cluster Name>Spark serviceConfiguration .
5. Select the ScopeGateway and CategoryAdvanced filters.

6. Add the following properties to Spark Client Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-defaults.conf:
   - spark.sql.orc.enabled=true – Enables the new ORC format to read/write Spark data source tables and files.
   - spark.sql.hive.convertMetastoreOrc=true – Enables the new ORC format to read/write Hive tables.
   - spark.sql.orc.char.enabled=true – Enables the new ORC format to use CHAR types to read Hive tables. By default, STRING types are used for performance reasons. This is an optional configuration for Hive compatibility.
7. Click Save Changes, and then restart Spark and any other components that require a restart.

Reading Hive ORC tables

For existing Hive tables, Spark can read them without createOrReplaceTempView. If the table is stored as ORC format (the default), predicate push-down, partition pruning, and vectorized query execution are also applied according to the configuration.

```
spark.sql("SELECT * FROM hiveTable WHERE age = 20")
```

Accessing Avro data files from Spark SQL applications

Spark SQL supports loading and saving DataFrames from and to a variety of data sources. With the spark-avro library, you can process data encoded in the Avro format using Spark.

The spark-avro library supports most conversions between Spark SQL and Avro records, making Avro a first-class citizen in Spark. The library automatically performs the schema conversion. Spark SQL reads the data and converts it to Spark's internal representation; the Avro conversion is performed only during reading and writing data.

By default, when pointed at a directory, read methods silently skip any files that do not have the .avro extension. To include all files, set the avro.mapred.ignore.inputs.without.extension property to false. See Configuring Spark Applications.

Writing Compressed Data Files

To set the compression type used on write, configure the spark.sql.avro.compression.codec property:

```
sqlContext.setConf("spark.sql.avro.compression.codec","CODEC")
```

The supported *CODEC* values are uncompressed, snappy, and deflate. Specify the level to use with deflate compression in spark.sql.avro.deflate.level.

Accessing Partitioned Data Files

The spark-avro library supports writing and reading partitioned data. You pass the partition columns to the writer.

Specifying Record Name and Namespace

Specify the record name and namespace to use when writing to disk by passing recordName and recordNamespace as optional parameters.

Spark SQL

You can write SQL queries to query a set of Avro files. First, create a temporary table pointing to the directory containing the Avro files. Then query the temporary table:

```
sqlContext.sql("CREATE TEMPORARY TABLE TABLE_NAME
  USING com.databricks.spark.avro OPTIONS (path "INPUT_DIR"))
df = sqlContext.sql("SELECT * FROM TABLE_NAME")
```

Avro to Spark SQL Conversion

The spark-avro library supports conversion for all Avro data types:

- boolean -> BooleanType
- int -> IntegerType
- long -> LongType
- float -> FloatType
- double -> DoubleType
- bytes -> BinaryType
- string -> StringType
- record -> StructType
- enum -> StringType
- array -> ArrayType
- map -> MapType
- fixed -> BinaryType

The spark-avro library supports the following union types:

- union(int, long) -> LongType
- union(float, double) -> DoubleType
- union(any, null) -> any

The library does not support complex union types.

All doc, aliases, and other fields are stripped when they are loaded into Spark.

Spark SQL to Avro Conversion

Every Spark SQL type is supported:

- BooleanType -> boolean
- IntegerType -> int
- LongType -> long
- FloatType -> float
- DoubleType -> double
- BinaryType -> bytes
- StringType -> string
- StructType -> record
- ArrayType -> array
- MapType -> map
- ByteType -> int
- ShortType -> int
- DecimalType -> string
- BinaryType -> bytes
- TimestampType -> long

Limitations

Because Spark is converting data types, keep the following in mind:

- Enumerated types are erased - Avro enumerated types become strings when they are read into Spark, because Spark does not support enumerated types.
- Unions on output - Spark writes everything as unions of the given type along with a null option.
- Avro schema changes - Spark reads everything into an internal representation. Even if you just read and then write the data, the schema for the output is different.
- Spark schema reordering - Spark reorders the elements in its schema when writing them to disk so that the elements being partitioned on are the last elements.

Accessing Parquet files from Spark SQL applications

Spark SQL supports loading and saving DataFrames from and to a variety of data sources and has native support for Parquet.

To read Parquet files in Spark SQL, use the spark.read.parquet("*PATH*") method.

To write Parquet files in Spark SQL, use the DataFrame.write.parquet("*PATH*") method.

To set the compression type, configure the spark.sql.parquet.compression.codec property:

```
spark.conf.set("spark.sql.parquet.compression.codec","CODEC")
```

The supported *CODEC* values are: uncompressed, gzip, lzo, and snappy. The default is gzip.

Currently, Spark looks up column data from Parquet files by using the names stored within the data files. This is different than the default Parquet lookup behavior of Impala and Hive. If data files are produced with a different physical layout due to added or reordered columns, Spark still decodes the column data correctly. If the logical layout of the table is changed in the metastore database, for example through an ALTER TABLE CHANGE statement that renames a column, Spark still looks for the data using the now-nonexistent column name and returns NULLs when it cannot locate the column values. To avoid behavior differences between Spark and Impala or Hive when modifying Parquet tables, avoid renaming columns, or use Impala, Hive, or a CREATE TABLE AS SELECT statement to produce a new table and new set of Parquet files containing embedded column names that match the new layout.

Using Spark MLlib

MLlib is Spark's machine learning library. For information on MLlib, see the Machine Learning Library (MLlib) Guide.

Running a Spark MLlib example

To try Spark MLlib using one of the Spark example applications, do the following:

1. Download MovieLens sample data and copy it to HDFS:

```
$ wget --no-check-certificate \
https://raw.githubusercontent.com/apache/spark/branch-2.4/data/mllib/sam
ple_movielens_data.txt
$ hdfs dfs -copyFromLocal sample_movielens_data.txt /user/hdfs
```

2. Run the Spark MLlib MovieLens example application, which calculates recommendations based on movie reviews:

```
$ spark-submit --master local --class org.apache.spark.examples.mllib.Mo
vieLensALS \
SPARK_HOME/lib/spark-examples.jar \
--rank 5 --numIterations 5 --lambda 1.0 --kryo sample_movielens_data.txt
```

Enabling Native Acceleration For MLlib

MLlib algorithms are compute intensive and benefit from hardware acceleration. To enable native acceleration for MLlib, perform the following tasks.

Install Required Software

- Install the appropriate libgfortran 4.6+ package for your operating system. No compatible version is available for RHEL 6.

OS	Package Name	Package Version
RHEL 7.1	libgfortran	4.8.x
SLES 11 SP3	libgfortran3	4.7.2
Ubuntu 12.04	libgfortran3	4.6.3
Ubuntu 14.04	libgfortran3	4.8.4
Debian 7.1	libgfortran3	4.7.2

- Install the GPL Extras parcel or package.

Verify Native Acceleration

You can verify that native acceleration is working by examining logs after running an application. To verify native acceleration with an MLlib example application:

1. Do the steps in "Running a Spark MLlib Example."
2. Check the logs. If native libraries are not loaded successfully, you see the following four warnings before the final line, where the RMSE is printed:

```
15/07/12 12:33:01 WARN BLAS: Failed to load implementation from: com.git
hub.fommil.netlib.NativeSystemBLAS
15/07/12 12:33:01 WARN BLAS: Failed to load implementation from: com.gith
ub.fommil.netlib.NativeRefBLAS
15/07/12 12:33:01 WARN LAPACK: Failed to load implementation from: com.gi
thub.fommil.netlib.NativeSystemLAPACK
15/07/12 12:33:01 WARN LAPACK: Failed to load implementation from: com.git
hub.fommil.netlib.NativeRefLAPACK
Test RMSE = 1.5378651281107205.
```

You see this on a system with no libgfortran. The same error occurs after installing libgfortran on RHEL 6 because it installs version 4.4, not 4.6+.

After installing libgfortran 4.8 on RHEL 7, you should see something like this:

```
15/07/12 13:32:20 WARN BLAS: Failed to load implementation from: com.git
hub.fommil.netlib.NativeSystemBLAS
15/07/12 13:32:20 WARN LAPACK: Failed to load implementation from: com.gi
thub.fommil.netlib.NativeSystemLAPACK
Test RMSE = 1.5329939324808561.
```

Using custom libraries with Spark

Spark comes equipped with a selection of libraries, including Spark SQL, Spark Streaming, and MLlib.

If you want to use a custom library, such as a compression library or Magellan, you can use one of the following two spark-submit script options:

- The --jars option, which transfers associated .jar files to the cluster. Specify a list of comma-separated .jar files.
- The --packages option, which pulls files directly from Spark packages. This approach requires an internet connection.

For example, you can use the --jars option to add codec files. The following example adds the LZO compression library:

```
spark-submit --driver-memory 1G \
    --executor-memory 1G \
    --master yarn-client \
    --jars /usr/hdp/2.6.0.3-8/hadoop/lib/hadoop-lzo-0.6.0.2.6.0.3-8.jar \
    test_read_write.py
```

For more information about the two options, see Advanced Dependency Management on the Apache Spark "Submitting Applications" web page.

Note:

If you launch a Spark job that references a codec library without specifying where the codec resides, Spark returns an error similar to the following:

```
Caused by: java.lang.IllegalArgumentException: Compression codec com
.hadoop.compression.lzo.LzoCodec not found.
```

To address this issue, specify the codec file with the --jars option in your job submit command.

Related Information

Submitting Applications: Advanced Dependency Management

Using Apache Iceberg with Spark

Spark supports Apache Iceberg which provides a table format for huge analytic datasets. Iceberg enables you to work with large tables, especially on object stores, and supports concurrent reads and writes on all storage media. You can use Spark 3 to interact with Apache Iceberg tables.

For more information about using Iceberg with Spark, see *Using Apache Iceberg in* .

Related Information

Using Apache Iceberg in

Cloudera Runtime 7.3.1

Configuring Fault Tolerance

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

High Availability on HDFS clusters..... 4
Configuring HDFS High Availability..... 4
NameNode architecture..... 4
Preparing the hardware resources for HDFS High Availability..... 5
Using to manage HDFS HA..... 5
Administrative commands..... 12

High Availability on HDFS clusters

Configuring HDFS High Availability (HA) eliminates the NameNode as a potential single point of failure (SPOF) in an HDFS cluster.

Configuring HDFS High Availability

The HDFS NameNode High Availability (HA) feature enables you to run redundant NameNodes in the same cluster in an Active/Passive configuration with a hot standby. This eliminates the NameNode as a potential single point of failure (SPOF) in an HDFS cluster.

In a standard configuration, the NameNode is a single point of failure (SPOF) in an HDFS cluster. Each cluster has a single NameNode, and if that machine or process becomes unavailable, the cluster as a whole is unavailable until the NameNode is either restarted or brought up on a separate machine. This situation impacts the total availability of the HDFS cluster in two major ways:

- In the case of an unplanned event such as a machine crash, the cluster would be unavailable until an operator restarted the NameNode.
- Planned maintenance events such as software or hardware upgrades on the NameNode machine would result in periods of cluster downtime.

HDFS NameNode HA avoids this by facilitating either a fast failover to a standby NameNode during machine crash, or a graceful administrator-initiated failover during planned maintenance.

NameNode architecture

In a typical HA cluster, two different machines are configured as NameNodes. In a working cluster, one of the NameNode machine is in the Active state, and the other is in the Standby state.

The Active NameNode is responsible for all client operations in the cluster, while the Standby NameNode acts as a backup. The Standby machine maintains enough state to provide a fast failover (if required).

In order for the Standby node to keep its state synchronized with the Active node, both nodes communicate with a group of separate daemons called JournalNodes (JNs). When the Active node performs any namespace modification, the Active node durably logs a modification record to a majority of these JNs. The Standby node reads the edits from the JNs and continuously watches the JNs for changes to the edit log. Once the Standby Node observes the edits, it applies these edits to its own namespace. When using QJM, JournalNodes act as the shared editlog storage. In a failover event, the Standby ensures that it has read all of the edits from the JounalNodes before promoting itself to the Active state. (This mechanism ensures that the namespace state is fully synchronized before a failover completes.)

Note:

Secondary NameNode is not required in HA configuration because the Standby node also performs the tasks of the Secondary NameNode.

To provide a fast failover, it is also necessary that the Standby node has up-to-date information on the location of blocks in your cluster. To get accurate information about the block locations, DataNodes are configured with the location of all the NameNodes, and send block location information and heartbeats to all the NameNode machines.

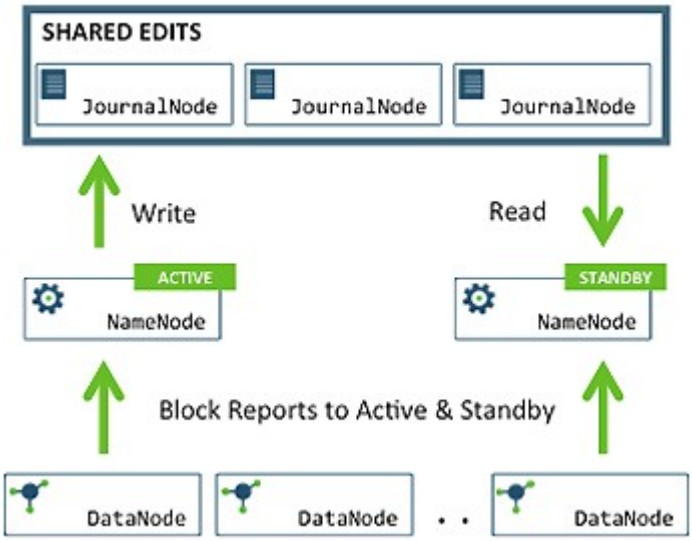

It is vital for the correct operation of an HA cluster that only one of the NameNodes should be Active at a time. Failure to do so would cause the namespace state to quickly diverge between the NameNode machines, thus causing potential data loss. (This situation is referred to as a split-brain scenario.)

To prevent the split-brain scenario, the JournalNodes allow only one NameNode to be a writer at a time. During failover, the NameNode, that is chosen to become active, takes over the role of writing to the JournalNodes. This process prevents the other NameNode from continuing in the Active state and thus lets the new Active node proceed with the failover safely.

Preparing the hardware resources for HDFS High Availability

Make sure that you prepare the required hardware resources for High Availability.

- NameNode machines: The machines where you run Active and Standby NameNodes, should have exactly the same hardware.
- JournalNode machines: The machines where you run the JournalNodes. The JournalNode daemon is relatively lightweight, so these daemons may reasonably be co- located on machines with other Hadoop daemons, for example the NameNodes or the YARN ResourceManager.

  Note:

  There must be at least three JournalNode daemons, because edit log modifications must be written to a majority of JNs. This lets the system tolerate failure of a single machine. You may also run more than three JournalNodes, but in order to increase the number of failures that the system can tolerate, you must run an odd number of JNs (3, 5, 7, and so on).

  Note that when running with N JournalNodes, the system can tolerate at most (N - 1) / 2 failures and continue to function normally.
- ZooKeeper machines: For automated failover functionality, there must be an existing ZooKeeper cluster available. The ZooKeeper service nodes can be co-located with other Hadoop daemons.

In an HA cluster, the Standby NameNode also performs checkpoints of the namespace state. Therefore, do not deploy a Secondary NameNode in an HA cluster.

Using to manage HDFS HA

You can use to configure your cluster for HDFS HA and automatic failover.

Enabling HDFS HA

An HDFS high availability (HA) cluster uses two NameNodes -- an active NameNode and a standby NameNode. Only one NameNode can be active at any point in time.

HDFS HA depends on maintaining a log of all namespace modifications in a location available to both NameNodes, so that in the event of a failure, the standby NameNode has up-to-date information about the edits and location of blocks in the cluster.

Important: Enabling and disabling HA causes a service outage for the HDFS service and all services that depend on HDFS. Before enabling or disabling HA, ensure that there are no jobs running on your cluster.

Prerequisites for enabling HDFS HA using

There are certain prerequisites that your cluster must satisfy before HDFS HA can be enabled using .

You must perform the following tasks *before* enabling HDFS HA:

- Ensure that you perform all the required configuration and setup tasks.
- Enabling and disabling HA causes a service outage for the HDFS service and all services that depend on HDFS. Before enabling or disabling HA, ensure that there are no jobs running on your cluster.
- Ensure that you have a ZooKeeper service configured.

Important:

- Enabling or disabling HA causes the previous monitoring history to become unavailable.
- Some parameters will be automatically set as follows once you have enabled JobTracker HA. If you want to change the value from the default for these parameters, use an advanced configuration snippet.
  - mapred.jobtracker.restart.recover: true
  - mapred.job.tracker.persist.jobstatus.active: true
  - mapred.ha.automatic-failover.enabled: true
  - mapred.ha.fencing.methods: shell(true)

Enabling High Availability and automatic failover

You can use to configure your cluster for HDFS HA and automatic failover. In , HA is implemented using Quorum-based storage. Quorum-based storage relies upon a set of JournalNodes, each of which maintains a local edits directory that logs the modifications to the namespace metadata. Enabling HA enables automatic failover as part of the same command.

About this task

- Cluster Administrator (also provided by Full Administrator)
- The Enable High Availability workflow leads you through adding a second (standby) NameNode and configuring JournalNodes.

Procedure

1. Go to the HDFS service.

2. Select Actions Enable High Availability .

   A screen showing the hosts that are eligible to run a standby NameNode and the JournalNodes displays.

   a) Specify a name for the nameservice and click Continue.

      Attention: Use unique names for the nameservices.

   b) In the NameNode Hosts field, click Select a host.

      The host selection dialog box displays.

   c) Select the checkbox next to the hosts where you want the standby NameNode to be set up and click OK.

      The standby NameNode cannot be on the same host as the active NameNode, and the host that is chosen should have the same hardware configuration (RAM, disk space, number of cores, and so on) as the active NameNode.

   d) In the JournalNode Hosts field, click Select hosts.

      The host selection dialog box displays.

   e) Check the checkboxes next to an odd number of hosts (a minimum of three) to act as JournalNodes and click OK.

      JournalNodes should be hosted on hosts with similar hardware specification as the NameNodes. recommends that you put a JournalNode each on the same hosts as the active and standby NameNodes, and the third JournalNode on similar hardware, such as the JobTracker.

   f) Click Continue.

   g) In the JournalNode Edits Directory property, enter a directory location for the JournalNode edits directory into the fields for each JournalNode host.

      - You may enter only one directory for each JournalNode. The paths do not need to be the same on every JournalNode.
      - The directories you specify should be empty.
      - The directory owner should be hdfs:hadoop and must have read, write, and r permission (drwx------).

   h) Extra Options: Decide whether should clear existing data in ZooKeeper, standby NameNode, and JournalNodes.

      If the directories are not empty (for example, you are re-enabling a previous HA configuration), will not automatically delete the contents—you can select to delete the contents by keeping the default checkbox selection. The recommended default is to clear the directories. If you choose not to do so, the data should be in sync across the edits directories of the JournalNodes and should have the same version data as the NameNodes.

   i) Click Continue.

      executes a set of commands that stop the dependent services, delete, create, and configure roles and directories as required, create a nameservice and failover controller, restart the dependent services, and deploy the new client configuration.

      Important: Some steps, such as formatting the NameNode may report failure if the action was already completed. However, the configuration steps continue to run after reporting non-critical failed steps.

      Note: The HDFS Instances page denotes the active and standby NameNodes in parentheses next to the respective role instances.

3. Restart Ranger KMS, if configured for your cluster.
4. Configure HDFS HA for other services, if required.

   Configuring other components to use HDFS HA on page 8

What to do next

Important: If you change the NameNode Service RPC Port (dfs.namenode.servicerpc-address) while automatic failover is enabled, this will cause a mismatch between the NameNode address saved in the ZooKeeper /hadoop-ha znode and the NameNode address that the Failover Controller is configured with. This will prevent the Failover Controllers from restarting. If you need to change the NameNode Service RPC Port after Auto Failover has been enabled, you must do the following to re-initialize the znode:

1. Stop the HDFS service.
2. Configure the service RPC port:
   a. Go to the HDFS service.
   b. Click the Configuration tab
   c. Select ScopeNameNode.
   d. Select CategoryPorts and Addresses.
   e. Locate the NameNode Service RPC Port property or search for it by typing its name in the Search box.
   f. Change the port value as needed.

      To apply this configuration property to other role groups as needed, edit the value for the appropriate role group.
3. On a ZooKeeper server host, run zookeeper-client.
   a. Run the following to remove the configured nameservice. This example assumes the name of the nameservice is nameservice1. You can identify the nameservice from the Federation and High Availability section on the HDFS Instances tab:

      ```
      rmr /hadoop-ha/nameservice1
      ```
4. Click the Instances tab.
5. Select ActionsInitialize High Availability State in ZooKeeper.
6. Start the HDFS service.

Disabling and redeploying HDFS HA

When you disable HDFS HA, ensures that one NameNode is active, and saves the namespace. Then it stops the standby NameNode, creates a Secondary NameNode, removes the standby NameNode role, and restarts all the HDFS services.

Procedure

1. Go to the HDFS service.
2. Select ActionsDisable High Availability.
3. Select the hosts for the NameNode and the Secondary NameNode and click Continue.
4. Select the HDFS checkpoint directory and click Continue.
5. Confirm that you want to take this action.
6. Configure the Hive Metastore to use HDFS HA.

Configuring other components to use HDFS HA

You can use the HDFS high availability NameNodes with other components of .

Configuring HBase to use HDFS HA

If you configure HBase to use an HA-enabled HDFS instance, automatically handles HA configuration for you.

Configuring the Hive Metastore to use HDFS HA

The Hive metastore can be configured to use HDFS high availability by using or by using the command line for unmanaged clusters.

Procedure

1. In the , go to the Hive service.

2. Select Actions Stop .

   Note: You may want to stop the Hue and Impala services first, if present, as they depend on the Hive service.

   Click Stop again to confirm the command.
3. Back up the Hive metastore database.
4. Select ActionsUpdate Hive Metastore NameNodes and confirm the command.
5. Select ActionsStart and click Start to confirm the command.
6. Restart the Hue and Impala services if you stopped them prior to updating the metastore.

Configuring Impala to work with HDFS HA

You must reconfigure the Hive metastore database and issue from the INVALIDATE METADATA statement from the Impala shell.

Procedure

1. Complete the steps to reconfigure the Hive metastore database.

   Impala shares the same underlying database with Hive, to manage metadata for databases, tables, and so on.
2. Issue the INVALIDATE METADATA statement from an Impala shell.

   This one-time operation makes all Impala daemons across the cluster aware of the latest settings for the Hive metastore database.

   Alternatively, restart the Impala service.

Configuring oozie to use HDFS HA

You can configure an Oozie workflow to use HDFS HA.

Procedure

- Use the HDFS nameservice instead of the NameNode URI in the <name-node> element of the workflow.

```
<action name="mr-node">
  <map-reduce>
    <job-tracker>${jobTracker}</job-tracker>
    <name-node>hdfs://ha-nn
```

  where *HA-NN* is the value of dfs.nameservices in hdfs-site.xml.

Changing a nameservice name for Highly Available HDFS using

Based on your requirements, you can change a nameservice name for a cluster that has HDFS HA configured.

Before you begin

Before you start, make note of the NameNode data directory and JournalNode edits directory. You can find dfs.name node.name.dir and dfs.journalnode.edits.dir from the configuration of HDFS service in the .

recommends you to take a backup for the above directories so that if there is any operator error, you still have a chance to recover.

In this example, the following values are assumed:

- The current nameservice name is nameservice1.
- The NameNode data directory is /data/dfs/nn.
- The JournalNode edits directory is /data/dfs/jn.

Procedure

1. Ensure that both NameNodes enter safemode and you save the current namespace and merge all the recent edits to the same fsimage id for both NameNodes.
   a) Go to HDFS service Instance .
   b) Click NameNodes (Active) Actions Enter Safemode .
   c) Click NameNodes (Active) Actions Save Namespace .
   d) Click NameNodes (Standby) Actions Enter Safemode .
   e) Click NameNodes (Standby) Actions Save Namespace .
   f) ssh to both NameNodes data directories to ensure that the same fsimage is generated.
2. Delete the current nameservice znode from ZooKeeper CLI.
   a) Stop all services except ZooKeeper.
   b) On a ZooKeeper host, run zookeeper-client.
   c) Execute the following to remove the configured nameservice:

      ```
      deleteall /hadoop-ha/nameservice1
      ```

      Login as ZooKeeper super user if Kerberos is enabled, so that you have the authorization to delete the protected znodes.
3. In the , change the NameNode nameservice name.
   a) Go to the HDFS service.
   b) Click Configuration.
   c) Type nameservice in the Search field.
   d) For the NameNode Nameservice property, change the nameservice name in the NameNode (instance_name) field.

      The name must be unique and can contain only alphanumeric characters.
   e) Type quorum in the Search field.
   f) For the Quorum-based Storage Journal name property, change the nameservice name in the NameNode (instance_name) field.
   g) Save changes.
4. Create a new nameservice znode.
   a) Click Instances.
   b) In the Federation and High Availability pane, select Actions Initialize High Availability State in ZooKeeper .
   c) On a ZooKeeper host, run zookeeper-client and verify that the new nameservice znode is created.

      ```
      ls /hadoop-ha
      ```
5. Update Hive Metastore NameNodes.
   a) Go to the Hive service.
   b) Select Actions Update Hive Metastore NameNodes .
6. If you have an Impala service, restart the Impala service or run an INVALIDATE METADATA query.
7. Initialize JournalNode shared edits directory.
   a) Click Instances.
   b) Select the checkboxes next to the JournalNode role instances.
   c) Select Actions for Selected Start .
   d) Click any NameNode, select Actions Initialize Shared Edits Directory .
   e) ssh to JournalNode host to ensure that the JournalNode data directory has the new nameservice folder generated.

8. Start other HDFS roles.
   a) Click the Instance tab, select the checkboxes next to all the HDFS role instances except for datanodes.
   b) Select Actions for Selected Start .

      You can see both NameNodes web UIs show Active and Standby, also nameservice names are updated. However both NameNodes are in safemode waiting for DN to startup and send block reports.
   c) Click Instances, start all the DN roles, wait for both NN to exit safemode automatically.
9. Redeploy client configuration files.
10. Start all services except ZooKeeper.
11. For clusters having Ranger service, change the references to new nameservice name, if any.
    a) Go to Ranger Admin Web UI, click cm_hdfs, check available policies and change the references to new nameservice name on HDFS path, if any.
    b) On Ranger Admin Web UI, click cm_hive, check available policies and change the references to new nameservice name on Impala URI, if any.

Manually failing over to the standby NameNode

If you are running a HDFS service with HA enabled, you can manually cause the active NameNode to failover to the standby NameNode. This is useful for planned downtime—for hardware changes, configuration changes, or software upgrades of your primary host.

About this task

Procedure

1. Go to the HDFS service.
2. Click the Instances tab.
3. Click Federation and High Availability.
4. Locate the row for the Nameservice where you want to fail over the NameNode.

   Multiple rows display only when using HDFS federation.
5. Select Actions Manual Failover .

   Note: This option does not appear if HA is not enabled for the cluster.
6. From the pop-up, select the NameNode that should be made active, then click Manual Failover.

   Note: For advanced use only: You can set the Force Failover checkbox to force the selected NameNode to be active, irrespective of its state or the other NameNode's state. Forcing a failover will first attempt to failover the selected NameNode to active mode and the other NameNode to standby mode. It will do so even if the selected NameNode is in safe mode. If this fails, it will proceed to transition the selected NameNode to active mode. To avoid having two NameNodes be active, use this only if the other NameNode is either definitely stopped, or can be transitioned to standby mode by the first failover step.
7. Click Finish.

Results

transitions the NameNode you selected to be the active NameNode, and the other NameNode to be the standby NameNode. HDFS should never have two active NameNodes.

Additional HDFS haadmin commands to administer the cluster

After your HA NameNodes are configured and started, you will have access to some additional commands to administer your HA HDFS cluster.

The following are high-level uses of some important subcommands. For specific usage information of each subcommand, you should run hdfs haadmin -help <command>.

getservicestate

Determine whether the given NameNode is active or standby

Connect to the provided NameNode to determine its current state, printing either "standby" or "active" to STDOUT as appropriate. This subcommand might be used by `cron` jobs or monitoring scripts which need to behave differently based on whether the NameNode is currently active or standby.

checkhealth

Check the health of the given NameNode

Connect to the provided NameNode to check its health. The NameNode is capable of performing some diagnostics on itself, including checking if internal services are running as expected. This command will return 0 if the NameNode is healthy, non-zero otherwise. One might use this command for monitoring purposes.

Turning safe mode on HA NameNodes

You can use the applicable `dfsadmin` command options to turn safe mode on for a single or both the active and standby NameNodes.

Procedure

- To turn safe mode on for both NameNodes, run hdfs dfsadmin -safemode enter.
- To turn safe mode on for a single NameNode, run hdfs dfsadmin -fs hdfs://<host>:<port> -safemode enter

Note: For a full list of `dfsadmin` command options, run: hdfs dfsadmin -help.

Converting from an NFS-mounted shared edits directory to Quorum-Based Storage

Converting a HA configuration from using an NFS-mounted shared edits directory to Quorum-based storage involves disabling the current HA configuration then enabling HA using Quorum-based storage.

Procedure

1. Disable HA.
2. Empty the name directories of the standby NameNode.
3. Enable HA with Quorum-based Storage.

Administrative commands

The subcommands of `hdfs haadmin` are extensively used for administering an HA cluster.

Running the `hdfs haadmin` command without any additional arguments will display the following usage information:

```
Usage: HAAdmin [-ns <nameserviceId>]
 [-transitionToActive <serviceId>]
 [-transitionToStandby <serviceId>]
 [-failover [--forcefence] [--forceactive] <serviceId> <serviceId>]
 [-getServiceState <serviceId>]
 [-checkHealth <serviceId>]
 [-help <command>
```

This section provides high-level uses of each of these subcommands.

- transitionToActive and transitionToStandby: Transition the state of the given NameNode to Active or Standby.

  These subcommands cause a given NameNode to transition to the Active or Standby state, respectively. These commands do not attempt to perform any fencing, and thus should be used rarely. Instead, recommends using the following subcommand:

  ```
  hdfs haadmin -failover
  ```

- failover: Initiate a failover between two NameNodes.

  This subcommand causes a failover from the first provided NameNode to the second.

  - If the first NameNode is in the Standby state, this command transitions the second to the Active state without error.
  - If the first NameNode is in the Active state, an attempt will be made to gracefully transition it to the Standby state. If this fails, the fencing methods (as configured by dfs.ha.fencing.methods) will be attempted in order until one succeeds. Only after this process will the second NameNode be transitioned to the Active state. If the fencing methods fail, the second NameNode is not transitioned to Active state and an error is returned.
- getServiceState: Determine whether the given NameNode is Active or Standby.

  This subcommand connects to the provided NameNode, determines its current state, and prints either "standby" or "active" to STDOUT appropriately. This subcommand might be used by cron jobs or monitoring scripts.
- checkHealth: Check the health of the given NameNode.

  This subcommand connects to the NameNode to check its health. The NameNode is capable of performing some diagnostics that include checking if internal services are running as expected. This command will return 0 if the NameNode is healthy else it will return a non-zero code.

Cloudera Runtime 7.2.11

Using HBase Backup and Disaster Recovery

Date published: 2020-02-29
Date modified: 2022-11-25

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

HBase backup and disaster recovery strategies.....4

Configuring HBase snapshots.....4
About HBase snapshots.....4
Manage HBase snapshots using Cloudera Operational Database CLI.....5
Create a snapshot.....6
List snapshots.....6
Restore a snapshot.....8
List restored snapshots.....9
Delete snapshots.....11
Manage HBase snapshots using the HBase shell.....12
Shell commands.....13
Take a snapshot using a shell script.....13
Export a snapshot to another cluster.....14
Information and debugging.....15

Using HBase replication.....17
Common replication topologies.....17
Notes about replication.....17
Replication requirements.....18
Deploy HBase replication.....18
Replication across three or more clusters.....19
Enable replication on a specific table.....19
Configure secure replication.....20
Configure bulk load replication.....22
Enable bulk load replication using Cloudera Manager.....22
Create empty table on the destination cluster.....24
Disable replication at the peer level.....24
Stop replication in an emergency.....25
Initiate replication when data already exist.....25
Replicate pre-exist data in an active-active deployment.....26
Using the CldrCopyTable utility to copy data.....26
Effects of WAL rolling on replication.....27
Configuring secure HBase replication.....28
Restore data from a replica.....29
Verify that replication works.....30
Replication caveats.....31

HBase backup and disaster recovery strategies

Backup-and-restore is a standard set of operations for many databases. You must have an effective backup-and-restore strategy to ensure that you can recover data in case of data loss or failures.

HBase snapshot enables you to take a snapshot of a table without much impact on RegionServers, because snapshot, clone, and restore operations do not involve data copying. In addition, exporting a snapshot to another cluster has no impact on RegionServers.

If your data is already in an HBase cluster, replication is useful for getting the data into additional HBase clusters.

Configuring HBase snapshots

HBase snapshots enable you to take a point-in-time copy of a table with little performance impact on HBase. When you create and restore snapshots, it does not copy any data; therefore the impact is very low. In addition, you can export or copy snapshots from one cluster to another without significant impact on HBase because the table files are copied directly rather than reading table data from RegionServers.

About HBase snapshots

HBase snapshots allow you to clone a table without making data copies, and with minimal impact on RegionServers. Exporting the table to another cluster has little impact on the RegionServers.

Without using snapshots, the only way to backup or clone a table was to use the provided CopyTable or ExportTable tools. However, these methods have disadvantages:

- CopyTable and ExportTable can degrade RegionServer performance because they read the table data.
- Disabling the table means no reads or writes; this is usually unacceptable.

Use Cases

- Data migration
  - You can use HBase snapshots to migrate data from CDH 5.x or HDP 2.x to COD on CDP Public Cloud. Refer to the migration guide for more information.
- Recovery from user or application errors
  - Useful because it may be some time before the database administrator notices the error.

    Note:

    An administrator can create automation which takes new snapshots and deletes unneeded old snapshots.
  - The database administrator may want to save a snapshot before a major application upgrade or change.

    Note:

    Snapshots are not primarily used for system upgrade protection because they do not roll back binaries, and would not necessarily prevent bugs or errors in the system or the upgrade.
  - Recovery cases:
    - Roll back to the previous snapshot and merge in reverted data.
    - View previous snapshots and choose which snapshot to apply into production.

- Backup
  - Capture a copy of the database and store it outside HBase for disaster recovery.
  - Capture previous versions of data for compliance, regulation, and archiving.
  - Export from a snapshot on a live system provides a faster and simpler way of moving HBase data than the CopyTable and ExportTable.
- Audit or report view of data at a specific time
  - Capture monthly data for compliance.
  - Use for end-of-day/month/quarter reports.
- Application testing
  - Test schema or application changes on similar production data from a snapshot and then discard.

    For example:

    1. Take a snapshot.
    2. Create a new table from the snapshot content (schema and data)
    3. Manipulate the new table by changing the schema, adding and removing rows, and so on. The original table, the snapshot, and the new table remain independent of each other.
- Offload work
  - Capture, copy, and restore data to another site
  - Export data to another cluster

Zero-Copy Restore and Clone Table

From a snapshot, you can create a new table (clone_snapshot) or restore the original table (restore_snapshot). When cloning snapshots, a new table is created to reflect the state of the original table when the snapshot was taken. Restoring snapshots, however, reverts the original table state to that of when the snapshot was taken (any updates to the table after the snapshot was taken would be lost). These two operations do not involve data copies; instead, a link is created to point to the original hfiles.

Changes to a cloned or restored table do not affect the snapshot.

To clone a table to another cluster, you export the snapshot to the other cluster and then run the clone operation; see Exporting a snapshot to another cluster.

Storage Considerations

Because hfiles are immutable, a snapshot consists of a reference to the files in the table at the moment the snapshot is taken. No copies of the data are made during the snapshot operation, but copies may be made when a compaction or deletion is triggered. In this case, if a snapshot has a reference to the files to be removed, the files are moved to an archive folder, instead of being deleted. This allows the snapshot to be restored in full.

When a snapshot is deleted, all retained HFiles will be automatically deleted when the files are no longer needed.

Manage HBase snapshots using Cloudera Operational Database CLI

Data backup and restore operations are important for your Cloudera Operational Database (COD) database to support data recovery. COD CLI offers multiple commands that help you manage the HBase snapshots on your CDP environment.

Using the COD CLI, you can:

- Create a snapshot.
- View the list of saved snapshots currently maintained. These can include one-off immediate snapshots, as well as scheduled policy-based snapshots.
- Restore from a saved snapshot.
- List all the restore operations in your CDP environment.

- Delete snapshots.

Related Information

Notes about replication

Create a snapshot

You can use COD CLI to create snapshots of HBase tables. The `create-snapshot` command takes a snapshot of a table in HBase, exports the snapshot to the specified storage location and deletes the snapshot after the export is successful.

Command syntax

`cdp opdb create-snapshot --environment-name` *ENVIRONMENT_NAME* `--database-name` *DATABASE_NAME* `--table-name` *TABLE_NAME* `--snapshot-name` *SNAPSHOT_NAME* `--snapshot-location` *SNAPSHOT_LOCATION*

Parameters

--environment-name

The name of the CDP environment.

--database-name

The name of the operational database.

--table-name

The fully qualified name of the HBase table whose snapshot is to be taken.

--snapshot-name

The name of the new snapshot. The snapshot name is unique per database.

--snapshot-location

The name of the snapshot location URL on object store where the snapshot is to be stored.

```
cdp opdb create-snapshot --environment-name odx-wgel6j --database-name test
--table-name table_6uf7xt --snapshot-name s1 --snapshot-location s3a://cod-7
215/cod--yq3p370r6qro/backupdata
```

JSON output:

```
{
    "environmentName": "odx-wgel6j",
    "databaseName": "test",
    "status": "IN_PROGRESS",
    "creationTime": 0,
    "commandID": 1546336350
}
```

Related Information

COD CLI command reference GitHub repository

CDP CLI BETA command reference GitHub repository

List snapshots

You can list the snapshots created in your CDP environment against a COD database using the `list-snapshots` command. You can use different filters to list the snapshots, for example, table name, specific command ID, and time range.

Command syntax

`cdp opdb list-snapshots --environment-name` *ENVIRONMENT_NAME* `--database-name` *DATABASE_NAME* [`--table-name` *TABLE_NAME*] [`--command-id` *COMMAND_ID*] [`--from-creation-time` *FROM_CREATION_TIME*] [`--to-creation-time` *TO_CREATION_TIME*]

Parameters

--environment-name

The name of the CDP environment.

--database-name

The name of the operational database.

--table-name

The fully qualified name of the HBase table whose snapshots are to be listed.

--command-id

Command ID filter.

--from-creation-time

The starting snapshot creation time to search snapshots (inclusive).

--to-creation-time

The ending snapshot creation time to search snapshots (inclusive).

Without filters:

```
cdp opdb list-snapshots  --environment-name odx-wgel6j --database-name test
{
    "environmentName": "odx-wgel6j",
    "databaseName": "test",
    "snapshots": [
        {
            "tableName": "table_tob8vc",
            "snapshotName": "snapshot3_table_tob8vc",
            "creationTime": 1662014265011,
            "status": "SUCCESSFUL",
            "commandID": 1546336058,
            "snapshotLocation": "abfs://qedailynat-filesystem@q.dfs.core.win
dows.net/hbase"
        },
        {
            "tableName": "table_tob8vc",
            "snapshotName": "snapshot2_table_tob8vc",
            "creationTime": 1662014023725,
            "status": "SUCCESSFUL",
            "commandID": 1546336035,
            "snapshotLocation": "abfs://qedailynat-filesystem@q.dfs.core.win
dows.net/hbase"
        },
        {
            "tableName": "table_tob8vc",
            "snapshotName": "snapshot1_table_tob8vc",
            "creationTime": 1662013782370,
            "status": "DELETED",
            "commandID": 1546336077,
            "snapshotLocation": "abfs://qedailynat-filesystem@q.dfs.core.wi
ndows.net/hbase"
        },
        {
```

```
            "tableName": "table_lpirwy",
            "snapshotName": "snapshot_table_lpirwy",
            "creationTime": 1662013568312,
            "status": "DELETED",
            "commandID": 1546336004,
            "snapshotLocation": "abfs://qedailynat-filesystem@q.dfs.core.wi
ndows.net/hbase"
        }
]
}
```

With command ID filter:

```
cdp opdb list-snapshots  --environment-name odx-wgel6j --database-name test
--command-id 1546336385
{
    "environmentName": "odx-wgel6j",
    "databaseName": "test",
    "snapshots": [
        {
            "tableName": "table_6uf7xt",
            "snapshotName": "s2",
            "creationTime": 1662035967244,
            "status": "SUCCESSFUL",
            "commandID": 1546336385,
            "snapshotLocation": "abfs://qedailynat-filesystem@qedailynats
torageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase/"
        }
    ]
}
```

With table name filter:

```
cdp opdb list-snapshots  --environment-name odx-wgel6j --database-name test
--table-name table_6uf7xt
{
    "environmentName": "odx-wgel6j",
    "databaseName": "test",
    "snapshots": [
        {
            "tableName": "table_6uf7xt",
            "snapshotName": "s2",
            "creationTime": 0,
            "status": "SUCCESSFUL",
            "commandID": 1546336385,
            "snapshotLocation": "abfs://qedailynat-filesystem@qedailynatsto
rageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase/"
        }
    ]
}
```

Related Information

COD CLI command reference GitHub repository

CDP CLI BETA command reference GitHub repository

Restore a snapshot

You can use the `restore-snapshot` command to import a snapshot automatically from the snapshot location. This command restores a snapshot and deletes it afterward.

Command syntax

`cdp opdb restore-snapshot --environment-name` *ENVIRONMENT_NAME* `--database-name` *DATABASE_NAME* `--snapshot-name` *SNAPSHOT_NAME* `--target-environment-name` *TARGET_ENVIRONMENT_NAME* `--target-database-name` *TARGET_DATABASE_NAME*

Parameters

--environment-name

The name of the original CDP environment.

--database-name

The name of the original operational database.

--snapshot-name

The name of the snapshot to be restored.

--target-environment-name

The name of the target environment where the snapshot should be restored.

--target-database-name

The name of the target database where the snapshot should be restored.

```
cdp opdb restore-snapshot --environment-name odx-wgel6j --database-name test
 --snapshot-name s2 --target-environment-name odx-wgel6j --target-database-n
ame test
```

JSON output:

```
{
    "environmentName": "odx-wgel6j",
    "databaseName": "test",
    "snapshotName": "s2",
    "targetEnvironmentName": "odx-wgel6j",
    "targetDatabaseName": "test",
    "status": "IN_PROGRESS",
    "restoreTime": 1662036779279,
    "commandID": 1546336475
}
```

Related Information

COD CLI command reference GitHub repository

CDP CLI BETA command reference GitHub repository

List restored snapshots

You can use the `list-restore-snapshots` command to list all the restored snapshots in your CDP environment against a COD database. You can use various filters to list the snapshots, for example, snapshot name, target environment name where the snapshot is restored, target COD database, specific command ID, and time range.

Command syntax

`cdp opdb list-restore-snapshots --environment-name` *ENVIRONMENT_NAME* `--database-name` *DATABASE_NAME* [`--snapshot-name` *SNAPSHOT_NAME*] [`--target-environment-name` *TARGET_ENVIRONMENT_NAME*] [`--target-database-name` *TARGET_DATABASE_NAME*] [`--command-id` *COMMAND_ID*] [`--from-restore-time` *FROM_RESTORE_TIME*] [`--to-restore-time` *TO_RESTORE_TIME*]

Parameters

--environment-name

The name of the CDP environment.

--database-name

The name of the operational database.

--snapshot-name

The snapshot name to be restored.

--target-environment-name

The name of the target environment.

--target-database-name

The name of the target database.

--command-id

Command ID filter.

--from-restore-time

The starting snapshot restore time to search restore snapshots (inclusive).

--to-restore-time

The ending snapshot restore time to search restore snapshots (inclusive).

```
cdp opdb list-restore-snapshots --environment-name odx-wgel6j --database-nam
e test
```

JSON output:

```
{
    "environmentName": "odx-wgel6j",
    "databaseName": "test",
    "restoreSnapshots": [
        {
            "snapshotName": "snapshot2_table_tob8vc",
            "targetEnvironmentName": "odx-wgel6j",
            "targetDatabaseName": "test",
            "status": "IN_PROGRESS",
            "restoreTime": 1662014023725,
            "commandID": 1546336085
        }
    ]
}
```

Using snapshot name filter:

```
cdp opdb list-restore-snapshots    --environment-name odx-wgel6j --database-
name test --snapshot-name snapshot2_table_tob8vc
```

JSON output:

```
{
    "environmentName": "odx-wgel6j",
    "databaseName": "test",
    "restoreSnapshots": [
        {
            "snapshotName": "snapshot2_table_tob8vc",
```

```
            "targetEnvironmentName": "odx-wgel6j",
            "targetDatabaseName": "test",
            "status": "IN_PROGRESS",
            "restoreTime": 1662014023725,
            "commandID": 1546336085
        }
    ]
}
```

Using target environment name and target database name:

```
cdp opdb list-restore-snapshots    --environment-name odx-wgel6j --database-
name test --target-environment-name odx-wgel6j --target-database-name test
```

JSON output:

```
{
    "environmentName": "odx-wgel6j",
    "databaseName": "test",
    "restoreSnapshots": [
        {
            "snapshotName": "s2",
            "targetEnvironmentName": "odx-wgel6j",
            "targetDatabaseName": "test",
            "status": "SUCCESSFUL",
            "restoreTime": 1662035967244,
            "commandID": 1546336475
        },
        {
            "snapshotName": "snapshot2_table_tob8vc",
            "targetEnvironmentName": "odx-wgel6j",
            "targetDatabaseName": "test",
            "status": "IN_PROGRESS",
            "restoreTime": 1662014023725,
            "commandID": 1546336085
        }
    ]
}
```

Related Information

COD CLI command reference GitHub repository

CDP CLI BETA command reference GitHub repository

Delete snapshots

You can use the `delete-snapshot` command to delete a snapshot from the snapshot location where the snapshot is exported.

Command syntax

`cdp opdb delete-snapshot --environment-name` *ENVIRONMENT_NAME* `--database-name` *DATABASE_NAME* `--snapshot-name` *SNAPSHOT_NAME*

Parameters

--environment-name

The name of the CDP environment.

--database-name

The name of the operational database.

--snapshot-name

The snapshot name to be deleted.

```
cdp opdb delete-snapshot --environment-name odx-wgel6j --database-name test
--snapshot-name s3
```

JSON output:

```
{
    "environmentName": "odx-wgel6j",
    "databaseName": "test",
    "status": "DELETING",
    "commandID": 1546336526,
    "snapshotName": "s3"
}
```

Once the snapshot is deleted, the status is displayed as DELETED.

```
cdp opdb list-snapshots --environment-name odx-wgel6j --database-name test -
-command-id 1546336526
```

JSON output:

```
{
    "environmentName": "odx-wgel6j",
    "databaseName": "test",
    "snapshots": [
        {
            "tableName": "table_6uf7xt",
            "snapshotName": "s3",
            "creationTime": 1662036921064,
            "status": "DELETED",
            "commandID": 1546336526,
            "snapshotLocation": "abfs://qedailynat-filesystem@qe.dfs.core.wi
ndows.net/hbase/"
        }
    ]
}
```

Related Information

COD CLI command reference GitHub repository

CDP CLI BETA command reference GitHub repository

Manage HBase snapshots using the HBase shell

You can manage snapshots by using the HBase shell, or you can use a shell script.

Shell commands

The following table shows actions you can take from the shell.

Action	Shell command	Comments
Take a snapshot of tableX called snap shotX	`snapshot 'tableX', 'snapshotX'`	Snapshots can be taken while a table is disabled, or while a table is online and serving traffic. • If a table is disabled (using disable <table>), an offline snapshot is taken. This snapshot is managed by the master and fully consistent with the state when the table was disabled. This is the simplest and safest method, but it involves a service interruption because the table must be disabled to take the snapshot. • In an online snapshot, the table remains available while the snapshot is taken, and incurs minimal performance degradation of normal read/write loads. This snapshot is managed by the master and run on the RegionServers. The current implementation—simple-flush snapshots—provides no causal consistency guarantees. Despite this shortcoming, it offers the same degree of consistency as CopyTable and is a significant improvement.
Restore snapshot snapshotX (replaces the source table content)	`restore_snapshot 'snapshotX'`	Restoring a snapshot replaces the current version of a table with the data in that table when the snapshot was taken. To run this command, you must disable the target table. The restore_snapshot command takes the name of a snapshot e (appending a timestamp code), and then clones data into the original data and removes data not in the snapshot. If the operation succeeds, the target table is enabled.
List all available snapshots	`list_snapshots`	
List all available snapshots starting with 'mysnapshot_' (regular expression)	`list_snapshots 'my_snapshot_.*'`	
Remove a snapshot called snapshotX	`delete_snapshot 'snapshotX'`	
Create a new table tableY from a snapshot snapshotX	`clone_snapshot 'snapshotX', 'tableY'`	Cloning a snapshot creates a new table that serves the data kept at the time of the snapshot. The original table and the cloned table can be modified independently; new data written to one table does not show up on the other.

Take a snapshot using a shell script

You can take a snapshot using an operating system shell script, such as a Bash script, in the HBase Shell noninteractive mode.

This example Bash script shows how to take a snapshot in this way. This script is provided as an illustration only; do not use it in production.

```
#!/bin/bash
# Take a snapshot of the table passed as an argument
# Usage: snapshot_script.sh table_name
# Names the snapshot in the format snapshot-YYYYMMDD

# Parse the arguments
if [ -z $1 ]||[$1 == '-h' ]; then
 echo "Usage: $0 &lt;table&gt;"
 echo "       $0 -h"
```

```
 exit 1
fi

# Modify to suit your environment
export HBASE_PATH=/home/user/hbase
export DATE=`date +"%Y%m%d"`
echo "snapshot '$1', 'snapshot-$DATE'" | $HBASE_PATH/bin/hbase shell -n
status=$?
if [$status -ne 0]; then
 echo "Snapshot may have failed: $status"
fi
exit $status
```

HBase Shell returns an exit code of 0 on successA non-zero exit code indicates the possibility of failure, not a definite failure. Your script should check to see if the snapshot was created before taking the snapshot again, in the event of a reported failure.

Export a snapshot to another cluster

You can export any snapshot from one cluster to another. Exporting the snapshot copies the table's hfiles, logs, and the snapshot metadata, from the source cluster to the destination cluster.

Specify the -copy-from option to copy from a remote cluster to the local cluster or another remote cluster. If you do not specify the -copy-from option, the hbase.rootdir in the HBase configuration is used, which means that you are exporting from the current cluster. You must specify the -copy-to option, to specify the destination cluster.

Note: Snapshots must be enabled on the destination cluster.

The ExportSnapshot tool executes a MapReduce Job similar to distcp to copy files to the other cluster. It works at file-system level, so the HBase cluster can be offline.

Run ExportSnapshot as the hbase user or the user that owns the files. If the user, group, or permissions need to be different on the destination cluster than the source cluster, use the -chuser, -chgroup, or -chmod options as in the second example below, or be sure the destination directory has the correct permissions. In the following examples, replace the HDFS server path and port with the appropriate ones for your cluster.

To copy a snapshot called MySnapshot to an HBase cluster srv2 (hdfs://srv2:8020/hbase) using 16 mappers:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot MySnapshot -
copy-to hdfs://srv2:<hdfs_port>/hbase -mappers 16
```

To export the snapshot and change the ownership of the files during the copy:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot MySnapshot -
copy-to hdfs://srv2:<hdfs_port>/hbase -chuser MyUser -chgroup MyGroup -chmod
 700 -mappers 16
```

You can also use the Java -D option in many tools to specify MapReduce or other configuration properties. For example, the following command copies MY_SNAPSHOT to hdfs://cluster2/hbase using groups of 10 hfiles per mapper:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -Dsnapshot.export.defa
ult.map.group=10 -snapshot MY_SNAPSHOT -copy-to hdfs://cluster2/hbase
```

(The number of mappers is calculated as TotalNumberOfHFiles/10.)

To export from one remote cluster to another remote cluster, specify both -copy-from and -copy-to parameters.

You can then reverse the direction to restore the snapshot back to the first remote cluster.

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot snapshot-tes
t -copy-from hdfs://machine1/hbase -copy-to hdfs://machine2/my-backup
```

To specify a different name for the snapshot on the target cluster, use the -target option.

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot snapshot-tes
t -copy-from hdfs://machine1/hbase -copy-to hdfs://machine2/my-backup -targe
t new-snapshot
```

Information and debugging

You can use the SnapshotInfo tool to get information about a snapshot, including status, files, disk usage, and debugging information.

Examples:

Use the -h option to print usage instructions for the SnapshotInfo utility.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -h
Usage: hbase snapshot info [options]
Options:
    --snapshot <arg>     Snapshot to examine.
    --remote-dir <arg>   Root directory that contains the snapshots.
    --list-snapshots     List all the available snapshots and exit.
    --files              Files and logs list.
    --stats              Files and logs statistics.
    --schema             Describe the snapshotted table.
    --size-in-bytes      Print the size of the files in bytes.
```

For example,

```
hbase snapshot info --snapshot MySnapshot --files
```

Use the --list-snapshots option to list all snapshots and exit.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo –-list-snapshots --sna
pshot foobar
SNAPSHOT             | CREATION TIME        | TABLE NAME
snapshot-test        |  2014-06-24T19:02:54 | test
```

Important: Add a dummy *SNAPSHOT* argument for the SnapshotInfo command to work.

Use the --remote-dir option with the --list-snapshots option to list snapshots located on a remote system.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo --remote-dir s3a://myb
ucket/mysnapshot-dir --list-snapshots --snapshot foobar
SNAPSHOT |   CREATION TIME   | TABLE NAME
snapshot-test        2014-05-01 10:30    myTable
```

Use the --snapshot option to print information about a specific snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo --snapshot test-snapsh
ot
Snapshot Info
----------------------------------------
   Name: test-snapshot
```

```
   Type: DISABLED
  Table: test-table
Version: 0
Created: 2012-12-30T11:21:21
*************************************************************
```

Use the --snapshot with the --stats options to display additional statistics about a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo --stats --snapshot sna
pshot-test
Snapshot Info
----------------------------------------
   Name: snapshot-test
   Type: FLUSH
  Table: test
 Format: 0
Created: 2014-06-24T19:02:54


1 HFiles (0 in archive), total size 1.0k (100.00% 1.0k shared with the sourc
e table)
```

Use the --schema option with the --snapshot option to display the schema of a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo  --schema --snapshot s
napshot-test
Snapshot Info
----------------------------------------
   Name: snapshot-test
   Type: FLUSH
  Table: test
 Format: 0
Created: 2014-06-24T19:02:54
Table Descriptor
----------------------------------------
'test', {NAME => 'cf', DATA_BLOCK_ENCODING => 'FAST_DIFF', BLOOMFILTER =>
 'ROW', REPLICATION_SCOPE => '0',
COMPRESSION => 'GZ', VERSIONS => '1', TTL => 'FOREVER', MIN_VERSIONS => '
0', KEEP_DELETED_CELLS => 'false',
BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
```

Use the --files option with the --snapshot option to list information about files contained in a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo --snapshot test-snapsh
ot --files
Snapshot Info
----------------------------------------
   Name: test-snapshot
   Type: DISABLED
  Table: test-table
Version: 0
Created: 2012-12-30T11:21:21

Snapshot Files
----------------------------------------
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/bdf29c39da2a4f2b8188
9eb4f7b18107 (archive)
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/1e06029d0a2a4a709
051b417aec88291 (archive)
   86.8k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/506f601e14dc4c74a0
58be5843b99577 (archive)
```

```
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/5c7f6916ab724eacbce
a218a713941c4 (archive)
  293.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/aec5e33a6564441d9bd4
23e31fc93abb (archive)
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/97782b2fbf0743eda
acd8fef06ba51e4 (archive)

6 HFiles (6 in archive), total size 589.7k (0.00% 0.0 shared with the sou
rce table)
0 Logs, total size 0.0
```

Using HBase replication

If your data is already in an HBase cluster, replication is useful for getting the data into additional HBase clusters.

Common replication topologies

You must learn about the common replication topologies before you configure and deploy HBase replication.

Some common replication topologies are:

- A central source cluster might propagate changes to multiple destination clusters, for failover or due to geographic distribution.
- A source cluster might push changes to a destination cluster, which might also push its own changes back to the original cluster.
- Many different low-latency clusters might push changes to one centralized cluster for backup or resource-intensive data-analytics jobs. The processed data might then be replicated back to the low-latency clusters.
- Multiple levels of replication can be chained together to suit your needs.

Related Information

Apache documentation: Cluster replication

Replication caveats

Notes about replication

You must consider a few things before using HBase replication.

- The timestamps of the replicated HLog entries are kept intact. In case of a collision (two entries identical as to row key, column family, column qualifier, and timestamp) only the entry arriving later write will be read.
- Increment Column Values (ICVs) are treated as simple puts when they are replicated. In the case where each side of replication is active (new data originates from both sources, which then replicate each other), this may be undesirable, creating identical counters that overwrite one another. Therefore Cloudera recommends that different cluster do not write increments to the same coordinates. For more information about this issue, see https://issues.apache.org/jira/browse/HBase-2804.
- Make sure the source and destination clusters are time-synchronized with each other. Cloudera recommends you use Network Time Protocol (NTP).
- Some changes are not replicated and must be propagated through other means, such as Snapshots or CopyTable.
  - Data that existed in the active cluster before replication was enabled.
  - Operations that bypass the WAL, such as when using BulkLoad or API calls such as writeToWal(false).
  - Table schema modifications.

Related Information

Manage HBase snapshots using Cloudera Operational Database CLI

Use CopyTable

Replication requirements

Before configuring replication, make sure your environment meets all requirements.

- Replication is supported from CDH 5 to Cloudera Private Cloud Base , and from Cloudera Private Cloud Base to CDH 5, if the version of Cloudera Private Cloud Base is 7.1 or higher.
- Each host in both clusters must be able to reach every other host, including those in the ZooKeeper cluster.
- Every table that contains families that are scoped for replication must exist on each cluster and have exactly the same name. If your tables do not yet exist on the destination cluster, see *Create empty table on the destination cluster*.
- Replication is supported from HDP 3.1.5 to Cloudera Private Cloud Base, if the version of Cloudera Private Cloud Base is 7.1.6 or higher.

Related Information

Create empty table on the destination cluster

Deploy HBase replication

Deploy HBase replication

Once you have source and destination clusters, you can enable and configure HBase replication to use it as your data import method.

About this task

Important: You need to run this as the HBase superuser, usually called "hbase". To run replication-related HBase commands, you must have HBase user permissions. If ZooKeeper uses Kerberos, configure HBase Shell to authenticate to ZooKeeper using Kerberos before attempting to run replication-related commands. No replication-related ACLs are available at this time.

Procedure

1. Configure and start the source and destination clusters.
2. Create tables with the same names and column families on both the source and destination clusters, so that the destination cluster knows where to store data it receives. All hosts in the source and destination clusters have to be reachable to each other. see *Create empty table on the destination cluster*.
3. On the source cluster, enable replication in Cloudera Manager, or by setting hbase.replication to true in hbase-site.xml.
4. Obtain Kerberos credentials as the HBase principal. Substitute your fully.qualified.domain.name and realm in the following command:

   ```
   kinit -k -t /etc/hbase/conf/hbase.keytab
   hbase/FULLY.QUALIFIED.DOMAIN.NAME@YOUR-REALM.COM
   ```

5. On the source cluster, in HBase Shell, add the destination cluster as a peer, using the add_peer command.

   ```
   add_peer 'ID', 'CLUSTER_KEY'
   ```

   To compose the CLUSTER_KEY, use the following template:

   ```
   hbase.zookeeper.quorum:hbase.zookeeper.property.clientPort:zookeeper.zno
   de.parent
   ```

   For example:

   ```
   host1.com,host2.com,host3.com:2181:/hbase
   ```

6. On the source cluster, configure each column family to be replicated by setting its REPLICATION_SCOPE to 1, using commands such as the following in HBase Shell:

```
hbase> disable 'example_table'
hbase> alter 'example_table', {NAME => 'example_family', REPLICATION_SCOPE
 => '1'}
hbase> enable 'example_table'
```

7. Verify that replication is occurring by examining the logs on the source cluster for messages such as the following.

```
Considering 1 rs, with ratio 0.1
Getting 1 rs from peer cluster # 0
Choosing peer 192.0.2.49:62020
```

8. To verify the validity of replicated data, use the included VerifyReplication MapReduce job on the source cluster, providing it with the ID of the replication peer and table name to verify. Other options are available, such as a time range or specific families to verify.

   The command has the following form:

```
hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication [-
-starttime=timestamp1] [--stoptime=timestamp] [--families=comma separated
 list of families] <peerId> <tablename>
```

   The VerifyReplication command prints GOODROWS and BADROWS counters to indicate rows that did and did not replicate correctly.

Related Information

Replication requirements

Create empty table on the destination cluster

Replication across three or more clusters

When configuring replication among three or more clusters, Cloudera recommends to enable KEEP_DELETED_CELLS on column families in the destination cluster, where REPLICATION_SCOPE=1 in the source cluster.

The following commands show how to enable this configuration using HBase Shell.

- On the source cluster:

```
create 't1',{NAME=>'f1', REPLICATION_SCOPE=>1}
```

- On the destination cluster:

```
create 't1',{NAME=>'f1', KEEP_DELETED_CELLS=>'true'}
```

Enable replication on a specific table

You can enable HBase replication for a specific table on the source cluster.

To enable replication for a specific table on the source cluster, run the enable_table_replication <*TABLE*> command from the HBase shell on a cluster where a peer has been configured.

Running enable_table_replication <*TABLE*> does the following:

1. Verifies that the table exists on the source cluster.
2. If the table does not exist on the remote cluster, uses the peer configuration to duplicate the table schema, inclusing splits, on the remote cluster.
3. Enables replication on that table.

Configure secure replication

Secure replication configuration is the same whether your clusters are all in the same realm or not, with the exception of the last step.

About this task

The last step involves setting up custom secure replication configurations per peer. This can be convenient when you need to replicate to a cluster that uses different cross-realm authentication rules than the source cluster. For example, a cluster in Realm A may be allowed to replicate to Realm B and Realm C, but Realm B may not be allowed to replicate to Realm C. If you do not need this feature, skip the last step.

To use this feature, service-level principals and keytabs (specific to HBase) must be specified when you create the cluster peers using HBase Shell.

Important: Secure HBase replication is not supported if the source or the destination cluster uses custom ZooKeeper server principal. The default ZooKeeper principal is zookeeper.

Procedure

1. Set up Kerberos on your cluster.

   Note: HBase peer-to-peer replication from a non-Kerberized cluster to a Kerberized cluster is not supported.

2. If necessary, configure Kerberos cross-realm authentication:
   a. At the comman line, use the list_principals command to list the kdc, admin_server, and default_domain for each realm.
   b. Add this information to each cluster using Cloudera Manager:
      1. In Cloudera Manager, select the HDFS service.
      2. Click the Configuration tab.
      3. Find the Truster Kerberos Realms property.
      4. Add the target and source.
      5. Restart HDFS.
3. Configure ZooKeeper.
4. Configure the following HDFS parameters on both cluster:

```
<!-- In hdfs-site.xml or advanced configuration snippet -->
<property>
  <name>dfs.encrypt.data.transfer</name>
  <value>true</value>
</property>
<property>
  <name>dfs.data.transfer.protection</name>
  <value>privacy</value>
</property>

<!-- In core-site.xml or advanced configuration snippet -->
<property>
  <name>hadoop.security.authorization</name>
  <value>true</value>
</property>
<property>
  <name>hadoop.rpc.protection</name>
  <value>privacy</value>
</property>
<property>
```

```
  <name>hadoop.security.crypto.cipher.suite</name>
  <value>AES/CTR/NoPadding</value>
</property>
<property>
  <name>hadoop.ssl.enabled</name>
  <value>true</value>
</property>
```

If you use Cloudera Manager to manage your cluster, do not set these properties directly in configuration files, because Cloudera Manager will overwrite or ignore these settings. You must set these properties in Cloudera Manager.

For brevity, the Cloudera Manager setting names are not listed here, but you can search by property name. For instance, in the HDFS service configuration screen, search for dfs.encrypt.data.transfer. The Enable Data Transfer Encryption setting is shown. Selecting the box is equivalent to setting the value to true.

5. Configure the following HBase parameters on both clusters, using Cloudera Manager or in hbase-site.xml if you do not use Cloudera Managert.

```
<!-- In hbase-site.xml -->
<property>
  <name>hbase.rpc.protection</name>
  <value>privacy</value>
</property>
<property>
  <name>hbase.thrift.security.qop</name>
  <value>auth-conf</value>
</property>
<property>
  <name>hbase.thrift.ssl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.rest.ssl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.ssl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.security.authentication</name>
  <value>kerberos</value>
</property>
<property>
  <name>hbase.security.authorization</name>
  <value>true</value>
</property>
<property>
  <name>hbase.secure.rpc.engine</name>
  <value>true</value>
</property>
```

6. Add the cluster peers using the simplified add_peer syntax:

```
add_peer 'ID', 'CLUSTER_KEY'
```

7. If you need to add any peers which require custom security configuration, modify the add_peer syntax, using the following examples as a model.

```
add_peer 'vegas', CLUSTER_KEY => 'zk1.vegas.example.com:2181:/hbase',
         CONFIG => {'hbase.master.kerberos.principal'           => 'hbase/
_HOST@TO_VEGAS',
```

```
                      'hbase.regionserver.kerberos.principal'     => 'hbase/
_HOST@TO_VEGAS',
                      'hbase.regionserver.keytab.file'            => '/key
tabs/vegas_hbase.keytab',
                      'hbase.master.keytab.file'                  => '/keyta
bs/vegas_hbase.keytab'},
          TABLE_CFS => {"tbl" => [cf1']}

add_peer 'atlanta', CLUSTER_KEY => 'zk1.vegas.example.com:2181:/hbase',
         CONFIG => {'hbase.master.kerberos.principal'           => 'hbase/
_HOST@TO_ATLANTA',
                      'hbase.regionserver.kerberos.principal'     => 'hbase/
_HOST@TO_ATLANTA',
                      'hbase.regionserver.keytab.file'            => '/ke
ytabs/atlanta_hbase.keytab',
                      'hbase.master.keytab.file'                  => '/ke
ytabs/atlanta_hbase.keytab'},
          TABLE_CFS => {"tbl" => [cf2']}
```

Related Information

Initiate replication when data already exist

Configure bulk load replication

Bulk loading is the process of preparing and loading HFiles directly into HBase RegionServers bypassing the write path. If you bulk load data into HBase frequently and want to replicate this data, you must configure and use bulk load replication.

Because bulk loading data bypasses the write path, and this process does not generate WALs, your data will not be replicated to the backup cluster.

Enable bulk load replication using Cloudera Manager

You can enable bulk load replication using Cloudera Manager.

Before you begin

If you have enabled Kerberos cross-realm authentication:

1. At the command line, use the list_principals command to list the kdc, admin_server, and default_domain for each realm.
2. Add this information to each cluster using Cloudera Manager. For each cluster, go to HDFS Configuration Trusted Kerberos Realms . Add the target and source realms. This requires a restart of HDFS.
3. Enter a Reason for change, and then click Save Changes to commit the changes. Restart the role and service when Cloudera Manager prompts you to restart.

Procedure

1. Go to the source cluster from which you want to replicate the buck loaded data.
2. Go to HBase Configuration .
3. Select Scope > (Service-Wide).
4. Locate the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the Search box.

5. Add the following property values:
   - Name: hbase.replication.bulkload.enabled

     Value: true

     Description: Enable bulk load replication
   - Name: hbase.replication.cluster.id

     Value: source1

     Description: Provide a source cluster-ID. For example, source1.
6. Manually copy and paste the source cluster's HBase client configuration files in the target cluster where you want the data to be replicated. Copy core-site.xml, hdfs-site.xml, and hbase-site.xml to the target cluster. Do this for all RegionServers.
7. Go to the target cluster where you want the data to be replicated.
8. Go to HBase Configuration .
9. Select Scope > (Service-Wide).
10. Locate the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the Search box.
11. Add the following property value:
    - Name: hbase.replication.conf.dir

      Value: /opt/cloudera/fs_conf

      Description: Path to the configuration directory where the source cluster's configuration files have been copied. The path for copying the configuration file is [hbase.replication.conf.dir]/[hbase.replication.cluster.id], i.e.:/opt/cloudera/fs_conf/source/
12. Restart the HBase service on both clusters to deploy the new configurations.

    Important:
    - In the target cluster, ensure that you copy the configuration files to the path set in [hbase.replication.conf.dir]. There, you must create a directory with the [hbase.replication.cluster.id] name.
    - Make sure to set the correct file permissions to hbase user using the command:

      ```
      chown -R hbase:hbase /opt/cloudera/fs_conf/source
      ```
13. Add the peer to the source cluster as you would with normal replication.
    - In the HBase shell, add the target cluster as a peer using the following command:

      ```
      add_peer '1', CLUSTER_KEY => '<cluster_name>:<hbase_port>:/hbase'
      ```
    - Enable the replication for the table to which you will be bulk loading data using the command:

      ```
      enable_table_replication 'IntegrationTestBulkLoad'
      ```
    - Alternatively, you can allow replication on a column family using the command:

      ```
      disable 'IntegrationTestBulkLoad'
      alter 'IntegrationTestBulkLoad', {NAME => 'D', REPLICATION_SCOPE => '1'}
      enable 'IntegrationTestBulkLoad'
      ```

    You can verify if BulkLoad Replication is working in your set up by following the example in this blog post:

    https://blog.cloudera.com/blog/2013/09/how-to-use-hbase-bulk-loading-and-why/

Create empty table on the destination cluster

You can create table on the destination cluster by extracting the schema using HBase Shell.

About this task

If the table to be replicated does not yet exist on the destination cluster, you must create it.

Procedure

1. On the source cluster, describe the table using HBase Shell.
   The output is reformatted for readability:

```
hbase> describe acme_users

Table acme_users is ENABLED
acme_users
COLUMN FAMILIES DESCRIPTION
{NAME => 'user', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'NONE',
REPLICATION_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'NONE',
MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE',
BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'false'}
```

2. Copy the output and make the following changes:
   - For the TTL, change FOREVER to org.apache.hadoop.hbase.HConstants::FOREVER.
   - Add the word CREATE before the table name.
   - Remove the line COLUMN FAMILIES DESCRIPTION and everything above the table name.

   A command like the following:

```
create 'cme_users' ,

{NAME => 'user', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'NONE',
REPLICATION_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'NONE',
MIN_VERSIONS => '0', TTL => org.apache.hadoop.hbase.HConstants::FOREVER,
KEEP_DELETED_CELLS => 'FALSE',
BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'false'}
```

3. Paste the command into HBase Shell on the destination cluster.
   The table is created.

Related Information

Replication requirements
Deploy HBase replication

Disable replication at the peer level

You can disable replication for a specific peer.

Use the command disable_peer ("<peerID>") to disable replication for a specific peer. This will stop replication to the peer, but the logs are kept for future reference.

Note: This log accumulation is a powerful side effect of the disable_peer command and can be used to your advantage.

To re-enable the peer, use the command enable_peer(<"peerID">). Replication resumes.

Examples:

- To disable peer 1:

```
disable_peer("1")
```

- To re-enable peer 1:

```
enable_peer("1")
```

Stop replication in an emergency

If replication is causing serious problems, you can stop it while the clusters are running.

Open the shell on the source cluster and use the disable_peer command for each peer, then the disable_table_replication command. For example:

```
hbase> disable_peer("1")
   hbase> disable_table_replication 'table_name'
```

Already queued edits will be replicated after you use the disable_table_replication command, but new entries will not.

To start replication again, use the enable_peer command.

Initiate replication when data already exist

You can initiate replication when data already exist by taking advantage of the accumulation that happens when a replication peer is disabled.

About this task

You may need to start replication from some point in the past. For example, suppose you have a primary HBase cluster in one location and are setting up a disaster-recovery (DR) cluster in another. To initialize the DR cluster, you need to copy over the existing data from the primary to the DR cluster, so that when you need to switch to the DR cluster you have a full copy of the data generated by the primary cluster. Once that is done, replication of new data can proceed as normal.

Procedure

1. Start replication.
2. Add the destination cluster as a peer.
3. Immediately disable it using disable_peer. This will cause the source HBase cluster to temporarily spool replicated updates to tables while the next steps are completed.
4. Run hbase shell and issue a snapshot 'myTable', 'myTableSnapshot-122112' for each table on the source cluster. The snapshot command flushes the table from memory.
5. Export each snapshot from the source cluster and stage it on the destination cluster. On the source cluster, for example, run `hbase org.apache.hadoop.hbase.snapshot. ExportSnapshot -snapshot MySnapshot -copy-to hdfs://yourserver:8020/hbase_root_dir -mappers 16` for each table snapshot.
6. Import and restore the snapshot on the destination cluster.
   - On the destination cluster, run hbase shell and issue a restore_snapshot for each table.

   If you are replicating data from or to a secure cluster, see Configure Replication
7. Run enable_peer to re-enable the destination cluster. Re-enabling the peer will cause the source HBase cluster to send temporarily spooled updates to the destination cluster.

Related Information

Configure secure replication

Replicate pre-exist data in an active-active deployment

You can replicate pre-exist data in an active-active deployment if you run the copyTable job before starting the replication.

About this task

You have to run the copyTable job before starting the replication. If you start the job after enabling replication, the second cluster will re-send the data to the first cluster, because copyTable does not edit the clusterId in the mutation objects.

Procedure

1. Run the copyTable job and note the start timestamp of the job.
2. Start replication.
3. Run the copyTable job again with a start time equal to the start time you noted in step 1.
   Some data being pushed back and forth between the two clusters; but it minimizes the amount of data.

Using the CldrCopyTable utility to copy data

You can use the CldrCopyTable utility to copy data from one cluster to another. You can use it to bring data in sync for replication.

About this task

CldrCopyTable is Cloudera's version of the upstream CopyTable utility.

The --cldr.cross.domain option of CldrCopyTable enables you to copy data cross-realm.

Procedure

1. Ensure that the following properties have the correct values in the hbase-site.xml configuration file of the target cluster:

```
<property>
<name>hbase.security.replication.credential.provider.path</name>
<value>cdprepjceks://hdfs@[***NAMENODE_HOST***]:[***NAMENODE_PORT***]/hba
se-replication/credentials.jceks</value>
</property>

<property>
<name>hbase.security.replication.user.name</name>
<value>srv_[***WORKLOAD USER NAME***]</value>
</property>
```

2. Ensure that the source cluster can communicate with the target cluster:
   a) Get the ZooKeeper quorum address of the target cluster.
   b) Set the address as an environment parameter in the source cluster.
   c) Set a subnet that allows connection from the source cluster.

      For example by enabling the port 2181 for the ZooKeeper client.
3. Issue the CldrCopyTable command from the source cluster to write to the target cluster.

   Based on your target cluster setup you have to use either the --cldr.cross.domain or the --cldr.unsecure.peer option.

   For Secure target cluster

   Use the --cldr.cross.domain option:

```
hbase org.apache.hadoop.hbase.mapreduce.CldrCopyTable --cldr
.cross.domain --peer.adr=[***ZOOKEEPER QUORUM***]:[***ZOOKEEPER
```

```
 PORT***]:[***ZOOKEEPER ROOT FOR HBASE***] --new.name="[***NEW TABLE
 NAME***]" "[***SOURCE TABLE NAME***]"
```

For Unsecure target cluster

Use the --cldr.unsecure.peer option:

```
hbase.org.apache.hadoop.hbase.mapreduce.CldrCopyTable --cldr
.unsecure.peer --peer.adr=[***ZOOKEEPER QUORUM***]:[***ZOOKEEPER
 PORT***]:[***ZOOKEEPER ROOT FOR HBASE***] --new.name="[***NEW TABLE
 NAME***]" "[***SOURCE TABLE NAME***]"
```

4. Once the job is finished, check the target cluster and ensure that the copy was successful.

Effects of WAL rolling on replication

Adding and removing peer clusters with unpredictable WAL rolling occurring have effects on the replication.

When you add a new peer cluster, it only receives new writes from the source cluster since the last time the WAL was rolled.

The following diagram shows the consequences of adding and removing peer clusters with unpredictable WAL rolling occurring. Follow the time line and notice which peer clusters receive which writes. Writes that occurred before the WAL is rolled are not retroactively replicated to new peers that were not participating in the cluster before the WAL was rolled.

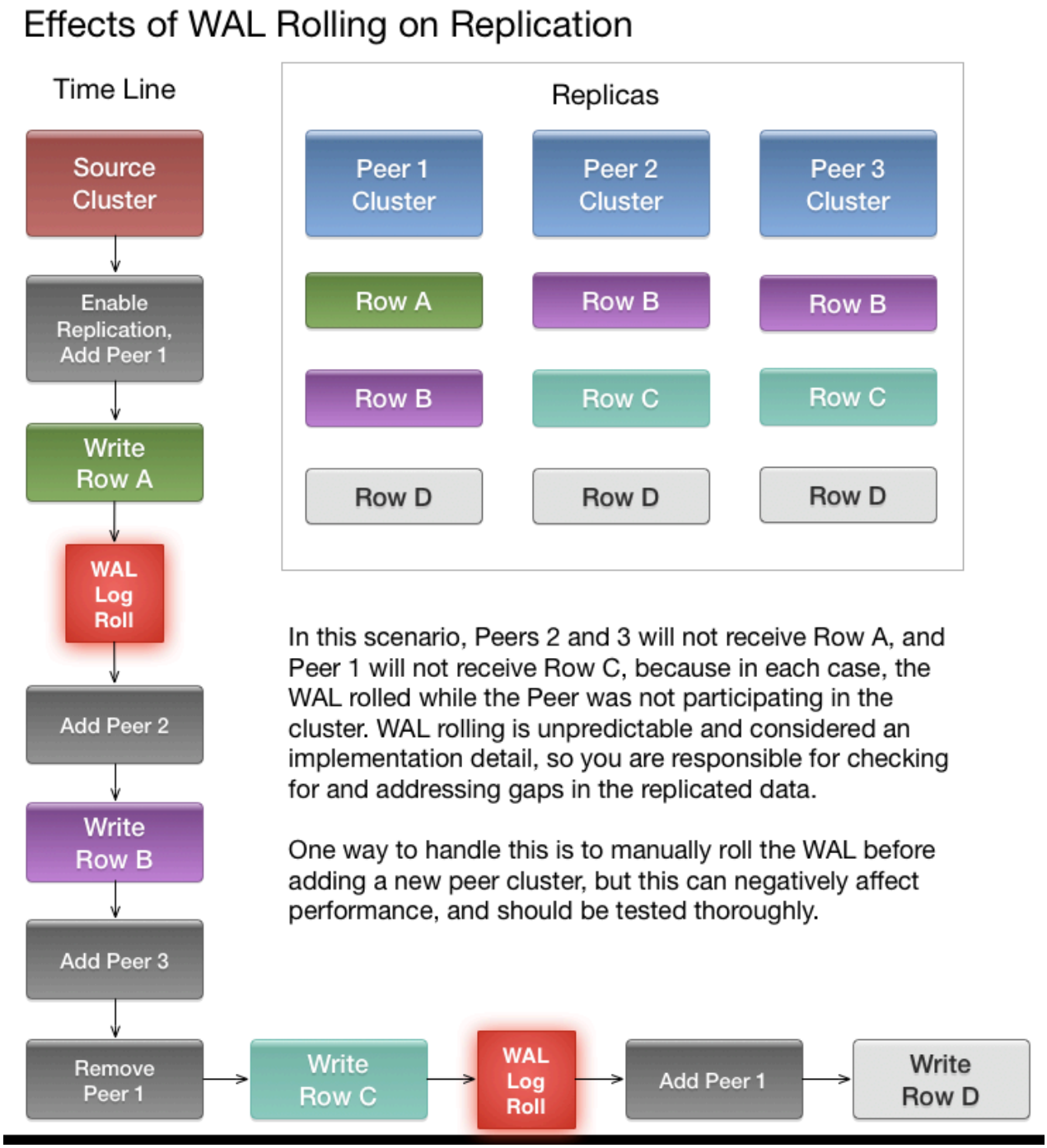

Configuring secure HBase replication

You must configure cross realm support for Kerberos, ZooKeeper, and Hadoop to configure secure HBase replication.

About this task

There must be at least one common encryption mode between the two realms.

Note: HBase peer-to-peer replication from a non-Kerberized cluster to a Kerberized cluster is not supported.

Procedure

1. Create krbtgt principals for the two realms.

   For example, if you have two realms called EXAMPLE.com and COMPANY.TEST, you need to add the following principelas: krbtgt/EXAMPLE.COM@COMPANY.TEST and krbtgt/COMPANY.TEST@EXAMPLE.COM

2. Add the two principals at both realms.

```
kadmin: addprinc -e "<enc_type_list>" krbtgt/EXAMPLE.COM@COMPANY.TEST
kadmin: addprinc -e "<enc_type_list>" krbtgt/COMPANY.TEST@EXAMPLE.COM
```

Add rules creating short names in ZooKeeper:

3. Add a system level property in java.env, defined in the conf directory.

   The following example rule illustrates how to add support for the realm called EXAMPLE.COM and have two members in the principal (such as service/instance@EXAMPLE.COM):

```
-Dzookeeper.security.auth_to_local=RULE:[2:\$1@\$0](.*@\\QEXAMPLE.COM\\E
$)s/@\\QEXAMPLE.COM\\E$//DEFAULT
```

   This example adds support for the EXAMPLE.COM realm in a different realm. So, in the case of replication, you must add a rule for the primary cluster realm in the replica cluster realm. DEFAULT is for defining the default rule

Add rules for creating short names in the Hadoop processes:

4. Add the hadoop.security.auth_to_local property in the core-site.xml file in the replica cluster.

   For example to add support for the EXAMPLE.COM realm:

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
    RULE:[2:$1@$0](.*@\QEXAMPLE.COM\E$)s/@\QEXAMPLE.COM\E$//
  DEFAULT
  </value>
</property>
```

Restore data from a replica

Recover HBase data from a replicated cluster in a disaster recovery scenario.

About this task

One of the main reasons for replications is to be able to restore data, whether during disaster recovery or for other reasons. During restoration, the source and sink roles are reversed. The source is the replica cluster, and the sink is the cluster that needs restoration. This can be confusing, especially if you are in the middle of a disaster recovery scenario. The following image illustrates the role reversal between normal production and disaster recovery.

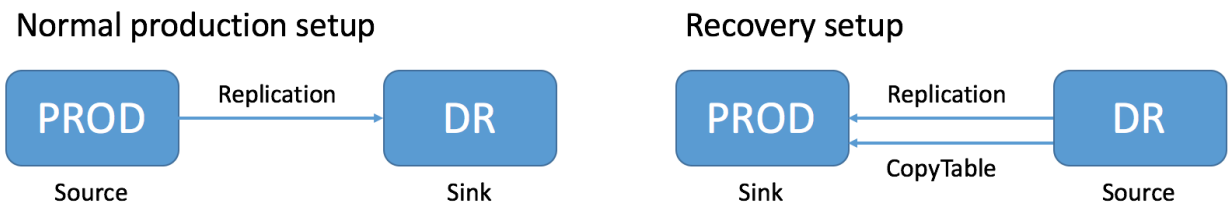

Procedure

1. Change the value of the column family property REPLICATION_SCOPE on the sink to 0 for each column to be restored, so that its data will not be replicated during the restore operation.
2. Change the value of the column family property REPLICATION_SCOPE on the source to 1 for each column to be restored, so that its data will be replicated.
3. Use the copyTable or distcp commands to import the data from the backup to the sink cluster.
4. Add the sink as a replication peer to the source, using the add_peer command.
5. If you used distcp in step 3 on page 30,restart or rolling restart both clusters.
   The RegionServers pick up the new files.

When restoration is complete, do the following:

6. Change the REPLICATION_SCOPE values back to their values before initiating the restoration.

Verify that replication works

Confirm data has been replicated from a source cluster to a remote destination cluster.

Procedure

1. Install and configure YARN on the source cluster.

   If YARN cannot be used in the source cluster, configure YARN on the destination cluster to verify replication.

   If neither the source nor the destination clusters can have YARN installed, you can configure the tool to use local mode; however, performance and consistency could be negatively impacted.
2. Ensure that you have the required permissions:
   - You have sudo permissions to run commands as the hbase user, or a user with admin permissions on both clusters.
   - You are an hbase user configured for submitting jobs with YARN.

   Note: To use the hbase user in a secure cluster, use Cloudera Manager to add the hbase user as a YARN whitelisted user. For a new installation, the hbase user is already added to the whitelisted users. In addition, /user/hbase should exist on HDFS and owned as the hbase user, because YARN will create a job staging directory there.
3. Run the VerifyReplication command:

```
src-node$ sudo -u hbase hbase org.apache.hadoop.hbase.mapreduce.replicat
ion.VerifyReplication peer1 table1
...
        org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication$V
erifier$Counters
                BADROWS=2
                CONTENT_DIFFERENT_ROWS=1
                GOODROWS=1
                ONLY_IN_PEER_TABLE_ROWS=1
        File Input Format Counters
```

```
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
```

The following table describes the VerifyReplication counters:

Table 1: VerifyReplication Counters

Counter	Description
GOODROWS	Number of rows. On both clusters, and all values are the same.
CONTENT_DIFFERENT_ROWS	The key is the same on both source and destination clusters for a row, but the value differs.
ONLY_IN_SOURCE_TABLE_ROWS	Rows that are only present in the source cluster but not in the destination cluster.
ONLY_IN_PEER_TABLE_ROWS	Rows that are only present in the destination cluster but not in the source cluster.
BADROWS	Total number of rows that differ from the source and destination clusters; the sum of CONTENT_DIFFERENT_ROWS + ONLY_IN_SOURCE_TABLE_ROWS + ONLY_IN_PEER_TABLE_ROWS

By default, VerifyReplication compares the entire content of table1 on the source cluster against table1 on the destination cluster that is configured to use the replication peer peer1.

Use the following options to define the period of time, versions, or column families

Table 2: VerifyReplication Counters

Option	Description
--starttime=<timestamp>	Beginning of the time range, in milliseconds. Time range is forever if no end time is defined.
--endtime=<timestamp>	End of the time range, in milliseconds.
--versions=<versions>	Number of cell versions to verify.
--families=<cf1,cf2,..>	Families to copy; separated by commas.

The following example, verifies replication only for rows with a timestamp range of one day:

```
src-node$ sudo -u hbase hbase org.apache.hadoop.hbase.mapreduce.replicat
ion.VerifyReplication --starttime=1472499077000 --endtime=1472585477000 --
families=c1  peer1 table1
```

Replication caveats

Note these caveats when you use HBase replication.

- Two variables govern replication: hbase.replication and a replication znode. Stopping replication (using disable_table_replication as above) sets the znode to false. Two problems can result:
  - If you add a new RegionServer to the active cluster while replication is stopped, its current log will not be added to the replication queue, because the replication znode is still set to false. If you restart replication at this point (using enable_peer), entries in the log will not be replicated.
  - Similarly, if a log rolls on an existing RegionServer on the active cluster while replication is stopped, the new log will not be replicated, because the replication znode was set to false when the new log was created.

- In the case of a long-running, write-intensive workload, the destination cluster may become unresponsive if its meta-handlers are blocked while performing the replication. Cloudera Runtime has three properties to deal with this problem:
  - hbase.regionserver.replication.handler.count - the number of replication handlers in the destination cluster (default is 3). Replication is now handled by separate handlers in the destination cluster to avoid the above-mentioned sluggishness. Increase it to a high value if the ratio of active to passive RegionServers is high.
  - replication.sink.client.retries.number - the number of times the HBase replication client at the sink cluster should retry writing the WAL entries (default is 1).
  - replication.sink.client.ops.timeout - the timeout for the HBase replication client at the sink cluster (default is 20 seconds).
- For namespaces, tables, column families, or cells with associated ACLs, the ACLs themselves are not replicated. The ACLs need to be re-created manually on the target table. This behavior opens up the possibility for the ACLs could be different in the source and destination cluster.

Related Information

Common replication topologies

Cloudera Runtime 7.2.11

Getting Started with Operational Database

Date published: 2020-02-29
Date modified: 2021-09-09

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Operational database cluster **4**

Before you create an operational database cluster **4**

Creating an operational database cluster **5**

Default operational database cluster definition 5

Provision an operational database cluster 6

Operational database cluster

You can use the Cloudera Operational Database Cloudera Data Hub cluster definition in Cloudera to create an operational database workload cluster that has Apache HBase, Apache Phoenix, and additional services that are required such as HDFS and ZooKeeper.

Important: You can create Cloudera Data Hub clusters only in a Cloudera Public Cloud deployment.

Based on the Cloudera Environment that you select, the Cloudera Operational Database Cloudera Data Hub cluster uses Amazon S3 or #Azure Data Lake Storage Gen2 as the storage layer for HBase. This means, that the HFiles are written to the object storage, but the Write Ahead Logs (WALs) are written to HDFS.

Cloudera Data Hub makes use of cluster definitions and reusable cluster templates. The Cloudera Operational Database Cloudera Data Hub cluster includes cloud-provider specific cluster definitions and prescriptive cluster configurations called cluster templates.

The Cloudera Operational Database with SQL cluster definition consists of Apache HBase, Apache Phoenix, and the underlying HDFS and Apache ZooKeeper services that it needs to operate. Apache Knox is used for the proxying of the different user interfaces.

Each cluster definition must reference a specific cluster template. You can also create and upload your custom cluster templates. For example, you can export a cluster template from an existing on-prem HDP cluster and register it in Cloudera to use it for creating Cloudera Data Hub clusters.

Figure 1: Video: Cloudera Operational Database with HBase on Cloudera Data Hub in Cloudera

https://www.youtube.com/embed/OY1sTyniDMI

Related Information

Cluster Templates

Before you create an operational database cluster

You must create an environment, register the environment, and set your user's password before you can create an operational database cluster.

Before you create an Operational Database cluster as a Cloudera Data Hub cluster in Cloudera, you must ensure that you have completed the following steps:

Set up user access and authorization

To provide access to resources such as environments and clusters, you must add users and groups and assign roles and resources to them. Each environment has an IPA instance which acts as a centralized user database. To access non-SSO interfaces, each user must set an IPA password.

For more information, see *Managing user access and authorization*.

Create an environment on AWS or Azure

You require the following to create an environment on AWS or Azure for use with Cloudera:

- Cloud credential to configure access from Cloudera to your cloud provider account
- SSH key-pair in Cloudera. You can generate an access key from the Cloudera Management Console using User Management
- Knowledge of your region and network information

For more information, see *Environment prerequisites on AWS*.

Register your environment in Cloudera

You must register an environment before you can create a cluster on that environment. The environment will retain all of this information for any clusters created within it and will create a FreeIPA cluster for centralized user management.

You can register an environment from the Cloudera Management Console using Environments > Register Environment.

For more information, see *Register an AWS environment* or *Register an Azure environment*.

Note: Your OpDB cluster may fail to initialize if you are reusing a cloud storage location that was used by an older Cloudera Data Hub cluster. If you want to reuse a cloud storage location, you should first stop HBase using Cloudera Manager before deleting an OpDB Cloudera Data Hub cluster. For instructions to stop HBase, see Stop HBase. If these steps are not followed, the new Cloudera Data Hub cluster may start with regions that are not correctly assigned because of the state of the old Cloudera Data Hub cluster, which will cause HBase to fail to fully start, commonly with the error "Master is initializing. By stopping HBase with Cloudera Manager prior to shutting down an older Cloudera Data Hub cluster, you can ensure that a new Cloudera Data Hub cluster will start without error when using the same cloud storage location.

Related Information

AWS account requirements

Azure subscription requirements

Creating an operational database cluster

After you have set up your environment, you can create an operational database as a Cloudera Data Hub cluster. Cloudera Data Hub clusters are those clusters that you can create and manage for your specific use case, and it is powered by Cloudera Runtime.

You can create multiple Cloudera Data Hub clusters using the same environment. For example, you can create an operational database cluster to replicate data from an on-prem cluster using HBase replication, and using the same environment you can create an operational database cluster that can be used for storing new data.

Default operational database cluster definition

When you create an operational database cluster, you can use the Operational Database with SQL template, and the default cluster definition. The cluster can be scaled up or down later, but the underlying cluster definition is unchanged to support the master services and the minimum required configuration.

The cluster definition primarily defines cloud provider settings. The Operational Database with SQL template has the following cluster definition:

- One gateway node that hosts the Knox gateway and Cloudera Manager, and a ZooKeeper Server.
- One node running the HDFS NameNode, and a ZooKeeper Server.
- One node running the HBase Master, and a ZooKeeper Server.
- At least three worker nodes. Each worker node has a RegionServer and a DataNode.
- A Phoenix Query Server runs on each worker node.

Note:

- You cannot access HBase, HDFS, and ZooKeeper from outside your AWS virtual private cloud (VPC) or Azure VNet because there is no network connectivity from outside your cloud instance. You can SSH to a node in the cluster that you started, or to another node which is within the cloud instance that is provided to you. The node provided to you should have the HBase client installation and the required HBase configuration files.
- An HDFS instance stores HBase Write Ahead Logs (WALs) because object storage currently does not provide the durability guarantees that HBase requires for WALs.

Provision an operational database cluster

You can provision an operational database cluster that has the default cluster definition or you can define the custom number of worker nodes that you need.

About this task

To provision a cluster:

Procedure

1. From the Cloudera Management Console, click Environments.
2. Search for your environment using the search box, and click on the environment name. Your environment clusters page appears.
3. Click Create Data Hub.

   The Provision Cloudera Data Hub page appears.
4. From the Selected Environment list, ensure that you have selected the environment in which you want to create the cluster. The templates that you see later will depend on the environment that you choose. For example, only the AWS templates are displayed if you select an AWS environment.
5. Choose the way you want to provision your cluster:
   - Cluster Definition: Select the Cluster Definition option to create your cluster quickly by using one of the prescriptive cluster definitions included by default or one of your previously created custom cluster definitions.
   - Custom: Select the Custom option to install selected services as part of the cluster. The list of services is determined by the selected Cluster Template.
6. If you selected Cluster Definition:
   a) From the Cluster Definition list, select the Operational Database with SQL template.
   b) In the Cluster Name field, provide a uniquely identifiable cluster name.

   or
7. If you selected Custom:
   a) From the Platform Version list, select the Cloudera Runtime version of your choice.
   b) From the Cluster Template list, select a cluster template of your choice.
   c) Under General Settings, in the Cluster Name field, provide a uniquely identifiable cluster name.
8. Under Tags, click Add to add tags to help you find your cluster-related resources in your cloud provider account.
9. Select Advanced Options to configure various option such as Image Catalog, Network And Availability, Hardware And Storage, Cloud Storage, and Cluster Extensions.
10. Set the number of RegionServers by editing the Instance Count in Advanced Options Hardware And Storage .
11. If you want to change the Instance Type of worker nodes to suit your requirements using Advanced Options Hardware And Storage .

    Note: The DataNodes and RegionServers are co-located.

12. Click Provision Cluster.
    After the cluster is successfully provisioned, you can access the Cloudera Manager user interface from the Cluster Information section.

Cloudera Runtime 7.2.11

Configuring Apache HBase High Availability

Date published: 2020-02-29
Date modified: 2021-09-09

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enable HBase high availability using Cloudera Manager....4

HBase read replicas....4

Timeline consistency....5

Keep replicas current....5

Read replica properties....5

Configure read replicas using Cloudera Manager....6

Using rack awareness for read replicas....7

Create a topology map....7

Create a topology script....8

Activate read replicas on a table....8

Request a timeline-consistent read....8

Enable HBase high availability using Cloudera Manager

You can configure one or more backup Master role in Cloudera Manager to ensure that every component is highly available.

About this task

HBase administrators can configure HBase clusters with High Availability, or HA. You must configure HA when HBase applications require low-latency queries and can tolerate minimal (near-zero-second) staleness for read operations. Examples include queries on remote sensor data, distributed messaging, object stores, and user profile management.

Most aspects of HBase are highly available in a standard configuration. A cluster typically consists of one Master and three or more RegionServers. To ensure that every component is highly available, configure one or more backup Masters. The backup Masters run on other hosts than the active Master.

Procedure

1. In Cloudera Manager, select the HBase service.
2. Follow the process for adding a role instance and add a backup Master to a different host than the one on which the active Master is running.

HBase read replicas

You can configure read replicas to enable the HMaster to distribute read-only copies of regions (replicas) to different RegionServers in the cluster.

Without read replicas, only one RegionServer services a read request from a client, regardless of whether RegionServers are colocated with other DataNodes that have local access to the same block. This ensures consistency of the data being read. However, a RegionServer can become a bottleneck due to an underperforming RegionServer, network problems, or other reasons that could cause slow reads. With read replicas enabled, the HMaster distributes read-only copies of regions (replicas) to different RegionServers in the cluster. One RegionServer services the default or primary replica, which is the only replica which can service write requests. If the RegionServer servicing the primary replica is down, writes will fail. Other RegionServers serve the secondary replicas, follow the primary RegionServer and only see committed updates. The secondary replicas are read-only, and are unable to service write requests.

The secondary replicas can be kept up to date by reading the primary replica's HFiles at a set interval or by replication. If they use the first approach, the secondary replicas may not reflect the most recent updates to the data when updates are made and the RegionServer has not yet flushed the memstore to HDFS. If the client receives the read response from a secondary replica, this is indicated by marking the read as "stale". Clients can detect whether or not the read result is stale and react accordingly. Replicas are placed on different RegionServers, and on different racks when possible. This provides a measure of high availability (HA), as far as reads are concerned.

If a RegionServer becomes unavailable, the regions it was serving can still be accessed by clients even before the region is taken over by a different RegionServer, using one of the secondary replicas. The reads may be stale until the entire WAL is processed by the new RegionServer for a given region. For any given read request, a client can request a faster result even if it comes from a secondary replica, or if consistency is more important than speed, it can ensure that its request is serviced by the primary RegionServer. This allows you to decide the relative importance of consistency and availability, in terms of the CAP Theorem, in the context of your application, or individual aspects of your application, using Timeline Consistency semantics.

Timeline consistency

You can change your consistency level to TIMELINE to allow a more flexible standard of consistency than the default HBase model of STRONG consistency.

A client can indicate the level of consistency it requires for a given read (Get or Scan) operation. The default consistency level is STRONG, meaning that the read request is only sent to the RegionServer servicing the region. This is the same behavior as when read replicas are not used. The other possibility, TIMELINE, sends the request to all RegionServers with replicas, including the primary. The client accepts the first response, which includes whether it came from the primary or a secondary RegionServer. If it came from a secondary, the client can choose to verify the read later or not to treat it as definitive.

Keep replicas current

You can choose from two different mechanisms for keeping replicas up to date: using a timer or using replication.

Using a Timer: In this mode, replicas are refreshed at a time interval controlled by the configuration option hbase.regionserver.storefile.refresh.period.

Using Replication: In this mode, replicas are kept current between a source and sink cluster using HBase Async WAL replication. This works similarly to HBase's multi-datacenter replication, but instead the data from a region is replicated to the secondary regions. Each secondary replica always receives and observes the writes in the same order that the primary region committed them. In some sense, this design can be thought of as *in-cluster replication*, where instead of replicating to a different datacenter, the data goes to secondary regions to keep secondary region's in-memory state up-to-date. The data files are shared between the primary region and other replicas, so that no extra storage overhead exists. However, the secondary regions contain recent non-flushed data in their memstores, which increases the memory overhead. The primary region writes flush, compaction, and bulk load events to its WAL as well, which are also replicated through WAL replication to secondaries. When they observe the flush or compaction or bulk load event, the secondary regions replay the event to pick up the new files and drop the old ones.

Committing writes in the same order as in primary ensures that the secondaries do not diverge from the primary regions data, however; the data might still be stale in secondary regions because the log replication is asynchronous.

Async WAL Replication is disabled by default. You can enable this feature by setting hbase.region.replica.replication.enabled to true.

Read replica properties

You must understand the various properties to configure to enable support for read replicas in HBase.

Table 1: HBase Read Replica Properties

Property Name	Default Value	Description
hbase.region.replica.replication.enabled	false	The mechanism for refreshing the secondary replicas. If set to false, secondary replicas are not guaranteed to be consistent at the row level. Secondary replicas are refreshed at intervals controlled by a timer (hbase.regionserver.storefile .refresh.period), and so are guaranteed to be at most that interval of milliseconds behind the primary RegionServer. Secondary replicas read from the HFile in HDFS, and have no access to writes that have not been flushed to the HFile by the primary RegionServer. If true, replicas are kept up-to-date using replication.
hbase.regionserver.storefile.refresh.period	0 (disabled)	The period, in milliseconds, for refreshing the store files for the secondary replicas. The default value of 0 indicates that the feature is disabled. Secondary replicas update their store files from the primary RegionServer at this interval. If refreshes occur too often, this can create a burden for the NameNode. If refreshes occur too infrequently, secondary replicas will be less consistent with the primary RegionServer.
hbase.ipc.client.specificThreadForWriting	true	Whether or not to enable interruption of RPC threads at the client. This is required for region replicas with fallback RPC's to secondary regions. The default value of true enables primary RegionServers to access data from other regions' secondary replicas.
hbase.client.primaryCallTimeout.get	10000 µs	The timeout period, in microseconds, an HBase client waits for a response before the read is submitted to a secondary replica if the read request allows timeline consistency. The default value is 10000 µs. Lower values increase the number of remote procedure calls while lowering latency.
hbase.client.primaryCallTimeout.multiget	10000 µs	The timeout period, in milliseconds, before an HBase client's multi-get request, such as HTab le.get(List<GET>)), is submitted to a secondary replica if the multi-get request allows timeline consistency. Lower values increase the number of remote procedure calls while lowering latency.

Configure read replicas using Cloudera Manager

You can configure read replicas using Cloudera Manager.

Procedure

1. Before you can use replication to keep replicas current, you must set the column attribute REGION_MEMST ORE_REPLICATION to false for the HBase table, using HBase Shell or the client API.
2. In Cloudera Manager, select the HBase service.
3. Click the Configuration tab.
4. Select Scope > HBase or HBase Service-Wide .
5. Select Category > Advanced .
6. Locate the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the Search box.

7. Create a configuration and paste it into the text field. The following example configuration demonstrates the syntax:

```
<property>
  <name>hbase.regionserver.storefile.refresh.period</name>
  <value>0</value>
</property>
<property>
  <name>hbase.ipc.client.allowsInterrupt</name>
  <value>true</value>
  <description>Whether to enable interruption of RPC threads at the client
. The default value of true is
    required to enable Primary RegionServers to access other RegionServers
 in secondary mode. </description>
</property>
<property>
  <name>hbase.client.primaryCallTimeout.get</name>
  <value>10</value>
</property>
<property>
  <name>hbase.client.primaryCallTimeout.multiget</name>
  <value>10</value>
</property>
```

8. Click Save Changes to commit the changes.
9. Restart the HBase service.

Using rack awareness for read replicas

Rack awareness for read replicas is modeled after the mechanism used for rack awareness in Hadoop.

The default implementation, which you can override by setting hbase.util.ip.to.rack.determiner, to custom implementation, is ScriptBasedMapping, which uses a topology map and a topology script to enforce distribution of the replicas across racks.

Create a topology map

The topology map assigns hosts to racks. You can create a topology map using Cloudera Manager.

About this task

The topology map assigns hosts to racks. It is read by the topology script. A rack is a logical grouping, and does not necessarily correspond to physical hardware or location. Racks can be nested. If a host is not in the topology map, it is assumed to be a member of the default rack. The following map uses a nested structure, with two data centers which each have two racks. All services on a host that are rack-aware will be affected by the rack settings for the host.

Procedure

- If you use Cloudera Manager, do not create the map manually. Instead, go to Hosts, select the hosts to assign to a rack, and select Actions for Selected > Assign Rack .

```
<topology>
  <node name="host1.example.com" rack="/dc1/r1"/>
  <node name="host2.example.com" rack="/dc1/r1"/>
  <node name="host3.example.com" rack="/dc1/r2"/>
  <node name="host4.example.com" rack="/dc1/r2"/>
  <node name="host5.example.com" rack="/dc2/r1"/>
  <node name="host6.example.com" rack="/dc2/r1"/>
```

```
  <node name="host7.example.com" rack="/dc2/r2"/>
  <node name="host8.example.com" rack="/dc2/r2"/>
</topology>
```

Create a topology script

The topology script determines rack topology using the topology map.

About this task

The topology script determines rack topology using the topology map. By default, CDH uses /etc/hadoop/conf.cloudera.YARN-1/topology.py

Procedure

- To use a different script, set net.topology.script.file.name to the absolute path of the topology script.

Activate read replicas on a table

After enabling read replica support on your RegionServers, configure the tables for which you want read replicas to be created.

About this task

After enabling read replica support on your RegionServers, configure the tables for which you want read replicas to be created.

Procedure

- At table creation, to create a new table with read replication capabilities enabled, set the REGION_REPLICATION property on the table. Use a command like the following, in HBase Shell:

  ```
  hbase> create 'myTable', 'myCF', {REGION_REPLICATION => '3'}
  ```

- By altering an existing table, you can also alter an existing column family to enable or change the number of read replicas it propagates, using a command similar to the following. The change will take effect at the next major compaction.

  ```
  hbase> disable 'myTable'
  hbase> alter 'myTable', 'myCF', {REGION_REPLICATION => '3'}
  hbase> enable 'myTable'
  ```

Request a timeline-consistent read

You can use request a timeline-consistent read in your application.

About this task

To request a timeline-consistent read in your application, use the get.setConsistency(Consistency.TIMELINE) method before performing the Get or Scan operation.

To check whether the result is stale (comes from a secondary replica), use the isStale() method of the result object. Use the following examples for reference.

Procedure

- Get request

```
Get get = new Get(key);
get.setConsistency(Consistency.TIMELINE);
Result result = table.get(get);
```

- Scan request

```
Scan scan = new Scan();
scan.setConsistency(CONSISTENCY.TIMELINE);
ResultScanner scanner = table.getScanner(scan);
Result result = scanner.next();
```

- Scan request to a specific replica.

  This example overrides the normal behavior of sending the read request to all known replicas, and only sends it to the replica specified by ID.

```
Scan scan = new Scan();
scan.setConsistency(CONSISTENCY.TIMELINE);
scan.setReplicaId(2);
ResultScanner scanner = table.getScanner(scan);
Result result = scanner.next();
```

- Detect a stale result

  You can also request timeline consistency using HBase Shell, allowing the result to come from a secondary replica.

```
hbase> get 'myTable', 'myRow', {CONSISTENCY => "TIMELINE"}
hbase> scan 'myTable', {CONSISTENCY => 'TIMELINE'}
```

Cloudera Runtime 7.2.11

Apache HBase

Date published: 2020-02-29
Date modified: 2021-09-09

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache HBase overview.....4

Apache HBase overview

Apache HBase is a scalable, distributed, column-oriented datastore. Apache HBase provides real-time read/write random access to very large datasets hosted on HDFS.

As a natively non-relational database, Apache HBase can combine data sources that use a wide variety of structures and schemas. HBase is designed for resilient data storage and is intended for hosting very large tables with sparse data. Apache HBase can host big data tables because it scales linearly to handle very large (petabyte scale), column-oriented data sets. The data store is predicated on a key-value model that supports low latency reads, writes, and updates in a distributed environment.

Cloudera Runtime 7.2.11

Managing Apache HBase Security

Date published: 2020-02-29
Date modified: 2022-12-21

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

HBase authentication....4
Configuring HBase servers to authenticate with a secure HDFS cluster.... 4
Configuring secure HBase replication.... 4
Configure the HBase client TGT renewal period....5
Disabling Kerberos authentication for HBase clients....5

HBase authorization.... 8

Configuring TLS/SSL for HBase.... 8
Prerequisites to configure TLS/SSL for HBase....8
Configuring TLS/SSL for HBase Web UIs....8
Configuring TLS/SSL for HBase REST Server.... 9
Configuring TLS/SSL for HBase Thrift Server....9
Configuring HSTS for HBase Web UIs.... 10

HBase authentication

You must establish a mechanism for HBase servers and clients to securely identify themselves with HDFS, ZooKeeper, and each other. This ensures that hosts are who they claim to be.

To enable HBase to work with Kerberos security, you must enable Kerberos Authentication for Cloudera Runtime and use Kerberos authentication for ZooKeeper. This means that HBase Master, RegionServer, and client hosts must each have a Kerberos principal for authenticating to the ZooKeeper ensemble.

Before you start configuring Kerberos authentication for HBase ensure that:

- Kerberos is enabled for the cluster.
- Kerberos principals for Cloudera Manager Server, HBase and ZooKeeper hosts exist and are available for use.

Cloudera Manager automatically configures authentication between HBase to ZooKeeper and sets up the HBase Thrift gateway to support impersonation (doAs). However, you must manually configure the HBase REST service for Kerberos, as it uses Simple authentication by default, instead of Kerberos.

Note: Impersonation (doAs) cannot be used with Thrift framed transport (TFramedTransport) because SASL does not work with Thrift framed transport.

Although an HBase Thrift server can connect to a secured Hadoop cluster, access is not secured from clients to the HBase Thrift server. To encrypt communication between clients and the HBase Thrift Server you must configure TLS/SSL for HBase Thrift Server.

Configuring HBase servers to authenticate with a secure HDFS cluster

You can configure HBase servers to authenticate with a secure HDFS cluster using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the HBase service.
2. Click the Configuration tab.
3. Use the HBase (Service-Wide) and Security filters.
4. Find the Kerberos Principal property.
5. Ensure the Kerberos principal for the HBase service was generated.
6. Find the HBase Secure Authentication property.
7. Select kerberos as authentication type.
8. Click Save Changes.
9. Restart the role.
10. Restart the HBase service.

Configuring secure HBase replication

You must configure cross realm support for Kerberos, ZooKeeper, and Hadoop to configure secure HBase replication.

About this task

There must be at least one common encryption mode between the two realms.

Note: HBase peer-to-peer replication from a non-Kerberized cluster to a Kerberized cluster is not supported.

Procedure

1. Create krbtgt principals for the two realms.

   For example, if you have two realms called EXAMPLE.com and COMPANY.TEST, you need to add the following principelas: krbtgt/EXAMPLE.COM@COMPANY.TEST and krbtgt/COMPANY.TEST@EXAMPLE.COM

2. Add the two principals at both realms.

```
kadmin: addprinc -e "<enc_type_list>" krbtgt/EXAMPLE.COM@COMPANY.TEST
kadmin: addprinc -e "<enc_type_list>" krbtgt/COMPANY.TEST@EXAMPLE.COM
```

Add rules creating short names in ZooKeeper:

3. Add a system level property in java.env, defined in the conf directory.

   The following example rule illustrates how to add support for the realm called EXAMPLE.COM and have two members in the principal (such as service/instance@EXAMPLE.COM):

```
-Dzookeeper.security.auth_to_local=RULE:[2:\$1@\$0](.*@\\QEXAMPLE.COM\\E
$)s/@\\QEXAMPLE.COM\\E$//DEFAULT
```

   This example adds support for the EXAMPLE.COM realm in a different realm. So, in the case of replication, you must add a rule for the primary cluster realm in the replica cluster realm. DEFAULT is for defining the default rule

Add rules for creating short names in the Hadoop processes:

4. Add the hadoop.security.auth_to_local property in the core-site.xml file in the replica cluster.

   For example to add support for the EXAMPLE.COM realm:

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
    RULE:[2:$1@$0](.*@\QEXAMPLE.COM\E$)s/@\QEXAMPLE.COM\E$//
  DEFAULT
  </value>
</property>
```

Configure the HBase client TGT renewal period

You must configure the HBase Client TGT Renewal Period to a value that allows the user enough time to finish HBase client processes.

An HBase client user must have a Kerberos principal which typically has a password that only the user knows. Configure the maxrenewlife setting for the client's principal to a value that allows the user enough time to finish HBase client processes before the ticket granting ticket (TGT) expires.

For example, if the HBase client processes require up to four days to complete, you should create the user's principal and configure the maxrenewlife setting by using this command:

```
kadmin: addprinc -maxrenewlife 4days
```

Disabling Kerberos authentication for HBase clients

Client applications that run on legacy products and do not have Kerberos enabled, fail to connect to Cloudera Operational Database instances that have Kerberos enabled. You can disable Kerberos authentication in your Cloudera Operational Database instances so that HBase or Phoenix clients can connect seamlessly.

About this task

You can also disable the Kerberos authentication using Cloudera Operational Database CLI while creating an operational database.

You can use the --disable-kerberos option while running the `create-database` command to disable the Kerberos authentication.

`cdp opdb create-database --environment-name` *ENVIRONMENT_NAME* `--database-name` *DATABASE_NAME* `--disable-kerberos`

For example,

```
cdp opdb create-database --environment-name cdp_7215 --database-name cod1 --
disable-kerberos
```

Important: Cloudera recommends that you do not disable Kerberos authentication. You might want to disable it for testing purposes until the legacy applications are restructured to use Kerberos authentication. When you disable Kerberos authentication, Cloudera Operational Database becomes vulnerable to malicious clients.

Procedure

1. In Cloudera Manager, go to the HBase service.
2. In the Configuration tab, set hbase.security.authentication=simple under HBase Secure Authentication and hbase.thrift.security.qop=none under HBase Thrift Authentication.

Figure 1: HBase security authentication

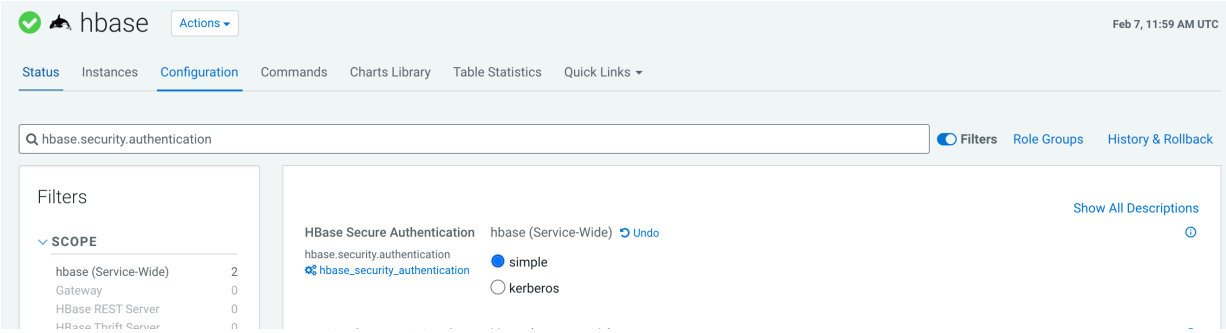

Figure 2: HBase thrift authentication

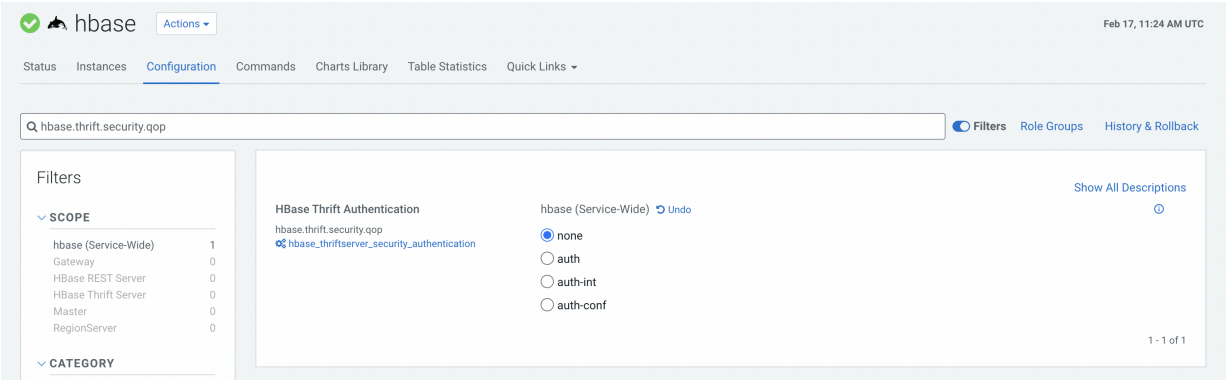

3. In the Ranger web UI, add the phoenix user. This resolves the impersonation issue.
   a) On the Data lake page, select the Ranger service.
   b) On the Ranger Service Manager web UI, find and select the HBase policy for your Cloudera Operational Database instance.

   Figure 3: Ranger service manager

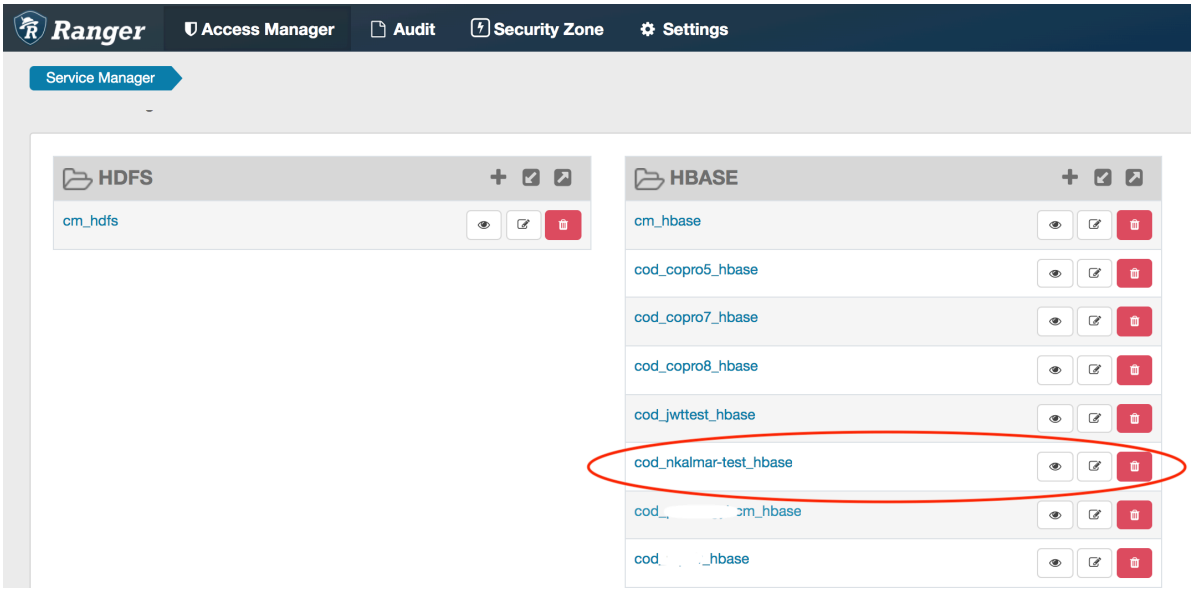

   c) Click on the edit button in the Action column to edit the *ALL - TABLE, COLUMN-FAMILY, COLUMN* policy.

   Figure 4: Ranger service manager policy

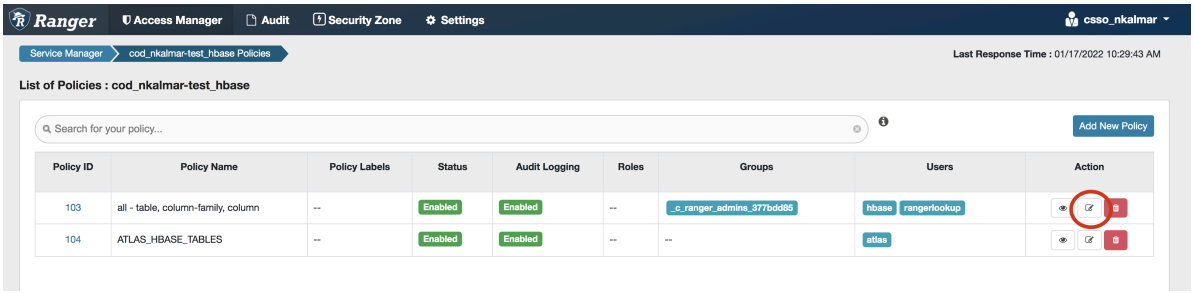

   d) In the Ranger service manager edit policy page, add the phoenix user and save the modified policy.

   Figure 5: Add phoenix user

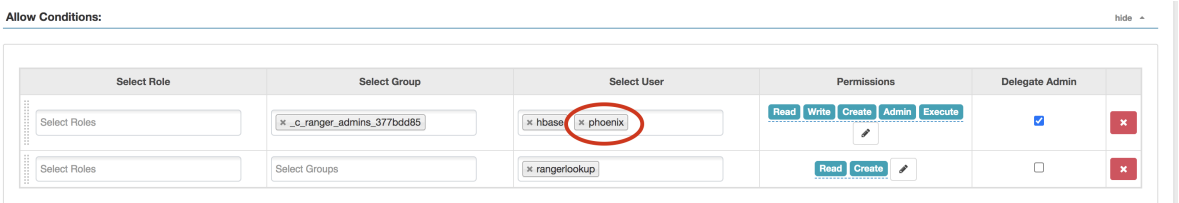

Results

Kerberos authentication is disabled in your Cloudera Operational Database instance and legacy HBase or Phoenix clients without having Kerberos enabled, can connect to your Cloudera Operational Database instance.

Related Information

Cloudera Operational Database CLI command reference GitHub repository

Cloudera CLI BETA command reference GitHub repository

HBase authorization

After configuring HBase authentication, you must define rules on resources that are allowed to access. Apache Ranger manages access control through a user interface that ensures consistent policy administration across Cloudera Data Lake components.

Once a user has been authenticated, their access rights must be determined. Authorization defines user access rights to resources. Authorization is concerned with who or what has access or control over a given resource or service. For example, a user may be allowed to create a policy and view reports, but not allowed to edit users and groups. HBase rules can be defined for individual tables, columns, and cells within a table.

For more information about how to set up HBase authorization using Ranger, see *Configure a Resource-based Service: HBase*, and then *Configure a Resource-based Policy: HBase*.

Related Information

Configure a Resource-based Service: HBase

Configure a Resource-based Policy: HBase

Configuring TLS/SSL for HBase

Once all the prerequisites are fulfilled, you can configure TLS/SSL for HBase Web UIs, HBase REST Server and HBase Thrift Server.

Prerequisites to configure TLS/SSL for HBase

Before configuring TLS/SSL for HBase, ensure that all prerequisites are fulfilled.

- Before enabling TLS/SSL, ensure that keystores containing certificates bound to the appropriate domain names will need to be accessible on all hosts on which at least one HBase daemon role is running.
- Keystores for HBase must be owned by the hbase group, and have permissions 0440 (that is, readable by owner and group).
- You must specify absolute paths to the keystore and truststore files. These settings apply to all hosts on which daemon roles of the HBase service run. Therefore, the paths you choose must be valid on all hosts.
- Cloudera Manager supports the TLS/SSL configuration for HBase at the service level. Ensure you specify absolute paths to the keystore and truststore files. These settings apply to all hosts on which daemon roles of the service in question run. Therefore, the paths you choose must be valid on all hosts.

  An implication of this is that the keystore file names for a given service must be the same on all hosts. If, for example, you have obtained separate certificates for HBase daemons on hosts node1.example.com and node2.example.com, you might have chosen to store these certificates in files called hbase-node1.keystore and hbase-node2.keystore (respectively). When deploying these keystores, you must give them both the same name on the target host — for example, hbase.keystore.

Configuring TLS/SSL for HBase Web UIs

You can configure TLS/SSL for HBase Web UIs using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the HBase service.
2. Click the Configuration tab.
3. Use the Scope / HBase (Service-Wide) filter.

4. Search for tls/ssl.
5. Check Web UI TLS/SSL Encryption Enabled.
6. Edit the HBase TLS/SSL properties according to your configuration.

Table 1: HBase TLS/SSL Properties

Property	Description
HBase TLS/SSL Server JKS Keystore File Location	Path to the keystore file containing the server certificate and private key used for encrypted web UIs.
HBase TLS/SSL Server JKS Keystore File Password	Password for the server keystore file used for encrypted web UIs.
HBase TLS/SSL Server JKS Keystore Key Password	Password that protects the private key contained in the server keystore used for encrypted web UIs.

7. Click Save Changes.
8. Restart the HBase service.

Configuring TLS/SSL for HBase REST Server

You can configure TLS/SSL for HBase REST Server using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the HBase service.
2. Click the Configuration tab.
3. Search for tls/ssl rest.
4. Check Enable TLS/SSL for HBase REST Server.
5. Edit the HBase REST Serer TLS/SSL properties according to your configuration.

Table 2: HBase TLS/SSL Properties

Property	Description
HBase REST Server TLS/SSL Server JKS Keystore File Location	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when HBase REST Server is acting as a TLS/SSL server. The keystore must be in JKS format.file.
HBase REST Server TLS/SSL Server JKS Keystore File Password	The password for the HBase REST Server JKS keystore file.
HBase REST Server TLS/SSL Server JKS Keystore Key Password	The password that protects the private key contained in the JKS keystore used when HBase REST Server is acting as a TLS/SSL server.

6. Click Save Changes.
7. Restart the HBase service.

Configuring TLS/SSL for HBase Thrift Server

You can configure TLS/SSL for HBase Thrift Server using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the HBase service.
2. Click the Configuration tab.

3. Search for tls/ssl thrift.
4. Check Enable TLS/SSL for HBase Thrift Server over HTTP.
5. Edit the HBase REST Server TLS/SSL properties according to your configuration.

Table 3: HBase TLS/SSL Properties

Property	Description
HBase Thrift Server over HTTP TLS/SSL Server JKS Keystore File Location	Path to the TLS/SSL keystore file (in JKS format) with the TLS/SSL server certificate and private key. Used when HBase Thrift Server over HTTP acts as a TLS/SSL server.
HBase Thrift Server over HTTP TLS/SSL Server JKS Keystore File Password	Password for the HBase Thrift Server JKS keystore file.
HBase Thrift Server over HTTP TLS/SSL Server JKS Keystore Key Password	Password that protects the private key contained in the JKS keystore used when HBase Thrift Server over HTTP acts as a TLS/SSL server.

6. Click Save Changes.
7. Restart the HBase service.

Configuring HSTS for HBase Web UIs

You can configure HBase to include HTTP headers to enforce the HTTP Strict Transport Security (HSTS) ensuring that a web browser does not load the service information using HTTP.

About this task

Additionally, all attempts to load the information using HTTP will automatically be converted to HTTPS.

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Advanced under Category.
4. Set the following HSTS credentials in HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml.

```
<property>
<name>hbase.http.filter.hsts.value</name>
<value>max-age=63072000;includeSubDomains;preload</value>
</property>
<property>
<name>hbase.http.filter.csp.value</name>
<value>default-src https: data: 'unsafe-inline' 'unsafe-eval'</value>
</property>
```

5. Restart the HBase service.

Cloudera Runtime 7.3.1

Configuring HDFS ACLs

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

HDFS ACLs....4

Configuring ACLs on HDFS....4

Using CLI commands to create and list ACLs....4

ACL examples....5

ACLS on HDFS features....8

Use cases for ACLs on HDFS....9

Enable authorization for HDFS web UIs....11

Enable authorization for additional HDFS web UIs....12

Configuring HSTS for HDFS Web UIs....12

HDFS ACLs

You can use Access Control Lists (ACLs) on the Hadoop Distributed File System (HDFS). ACLs extend the HDFS permission model to support more granular file access based on arbitrary combinations of users and groups.

Configuring ACLs on HDFS

ACLs are enabled by default. You can use to disable ACLs based on your requirements.

Procedure

1. Go to the and navigate to the HDFS service.
2. Click the Configuration tab.
3. Select Scope > *Service _name* .
4. Select Category > Security .
5. Locate the Enable Access Control Lists property and clear its checkbox to disable the HDFS ACLs.
6. Enter a Reason for change, and then click Save Changes to commit the changes.
7. Restart the NameNode.

Using CLI commands to create and list ACLs

You can use the sub-commands setfacl and getfacl to create and list ACLs on HDFS.These commands are modeled after the same Linux shell commands.

- setfacl

  Sets ACLs for files and directories.

  Example:

```
-setfacl [-bkR] {-m|-x} <acl_spec> <path>
```

```
-setfacl --set <acl_spec> <path>
```

  Options:

Table 1: ACL Options

Option	Description
-b	Remove all entries, but retain the base ACL entries. The entries for User, Group, and Others are retained for compatibility with Permission Bits.
-k	Remove the default ACL.
-R	Apply operations to all files and directories recursively.
-m	Modify the ACL. New entries are added to the ACL, and existing entries are retained.
-x	Remove the specified ACL entries. All other ACL entries are retained.

Option	Description
--set	Fully replace the ACL and discard all existing entries. The acl_spec must include entries for User, Group, and Others for compatibility with Permission Bits.
<acl_spec>	A comma-separated list of ACL entries.
<path>	The path to the file or directory to modify.

Examples:

```
hdfs dfs -setfacl -m user:hadoop:rw- /file
hdfs dfs -setfacl -x user:hadoop /file
hdfs dfs -setfacl -b /file
hdfs dfs -setfacl -k /dir
hdfs dfs -setfacl --set user::rw-,user:hadoop:rw-,group::r--,other::r-- /
file
hdfs dfs -setfacl -R -m user:hadoop:r-x /dir
hdfs dfs -setfacl -m default:user:hadoop:r-x /dir
```

Exit Code:

Returns 0 on success and non-zero on error.

- getfacl

  Displays the ACLs of files and directories. If a directory has a default ACL, getfacl also displays the default ACL.

  Usage:

```
-getfacl [-R] <path>
```

  Options:

  Table 2: getfacl Options

Option	Description
-R	List the ACLs of all files and directories recursively.
<path>	The path to the file or directory to list.

  Examples:

```
hdfs dfs -getfacl /file
hdfs dfs -getfacl -R /dir
```

  Exit Code:

  Returns 0 on success and non-zero on error.

ACL examples

Access Control Lists (ACLs) on HDFS help in addressing access-related issues better than Permission Bits.

Drawbacks of Permission Bits

Before the implementation of ACls, the HDFS permission model was equivalent to traditional UNIX Permission Bits. In this model, permissions for each file or directory are managed by a set of three distinct user classes: Owner, Group, and Others. There are three permissions for each user class: Read, Write, and Execute. Thus, for any file system object, its permissions can be encoded in 3*3=9 bits. When a user attempts to access a file system object, HDFS

enforces permissions according to the most specific user class applicable to that user. If the user is the owner, HDFS checks the Owner class permissions. If the user is not the owner, but is a member of the file system object's group, HDFS checks the Group class permissions. Otherwise, HDFS checks the Others class permissions.

This model can sufficiently address a large number of security requirements. For example, consider a sales department that would like a single user -- Bruce, the department manager -- to control all modifications to sales data. Other members of the sales department need to view the data, but must not be allowed to modify it. Everyone else in the company (outside of the sales department) must not be allowed to view the data. This requirement can be implemented by running chmod 640 on the file, with the following outcome:

```
-rw-r-----1 brucesales22K Nov 18 10:55 sales-data
```

Only Bruce can modify the file, only members of the sales group can read the file, and no one else can access the file in any way.

Suppose new requirements arise as a result of which users Bruce, Diana, and Clark are allowed to make modifications. Permission Bits cannot address this requirement, because there can be only one owner and one group, and the group is already used to implement the read-only requirement for the sales team. A typical workaround is to set the file owner to a synthetic user account, such as "salesmgr," and allow Bruce, Diana, and Clark to use the "salesmgr" account through sudo or similar impersonation mechanisms. The drawback with this workaround is that it forces complexity on end-users, requiring them to use different accounts for different actions.

Consider another example where the sales staff and all the executives require access to the sales data. This is another requirement that Permission Bits cannot address, because there is only one group, and it is already used by sales. A typical workaround is to set the file's group to a new synthetic group, such as "salesandexecs," and add all users of "sales" and all users of "execs" to that group. The drawback with this workaround is that it requires administrators to create and manage additional users and groups.

The preceding examples indicate that Permission Bits cannot address permission requirements that differ from the natural organizational hierarchy of users and groups. ACLs enable you to address these requirements more naturally by allowing multiple users and multiple groups to have different sets of permissions.

Examples of using ACLs for addressing access permission requirements

The following examples explain how you can use ACLs and address different requirements related to access permissions:

Using a Default ACL for Automatic Application to New Children

In addition to an ACL that is enforced during permission checks, a default ACL is also available. A default ACL can only be applied to a directory -- not to a file. Default ACLs have no direct effect on permission checks for existing child files and directories, but instead define the ACL that new child files and directories will receive when they are created.

Suppose we have a "monthly-sales-data" directory that is further subdivided into separate directories for each month. We will set a default ACL to guarantee that members of the "execs" group automatically get access to new subdirectories as they get created each month.

- Set a default ACL on the parent directory:

  ```
  > hdfs dfs -setfacl -m default:group:execs:r-x /monthly-sales-data
  ```

- Make subdirectories:

  ```
  > hdfs dfs -mkdir /monthly-sales-data/JAN
  > hdfs dfs -mkdir /monthly-sales-data/FEB
  ```

- Verify that HDFS has automatically applied the default ACL to the subdirectories:

  ```
  > hdfs dfs -getfacl -R /monthly-sales-data
  # file: /monthly-sales-data
  ```

```
# owner: bruce
# group: sales
user::rwx
group::r-x
other::---
default:user::rwx
default:group::r-x
default:group:execs:r-x
default:mask::r-x
default:other::---

# file: /monthly-sales-data/FEB
# owner: bruce
# group: sales
user::rwx
group::r-x
group:execs:r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:execs:r-x
default:mask::r-x
default:other::---

# file: /monthly-sales-data/JAN
# owner: bruce
# group: sales
user::rwx
group::r-x
group:execs:r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:execs:r-x
default:mask::r-x
default:other::---
```

Blocking Access to a Sub-Tree for a Specific User

Suppose there is a need to immediately block access to an entire sub-tree for a specific user. Applying a named user ACL entry to the root of that sub-tree is the fastest way to accomplish this without accidentally revoking permissions for other users.

- Add an ACL entry to block user Diana's access to "monthly-sales-data":

```
> hdfs dfs -setfacl -m user:diana:--- /monthly-sales-data
```

- Run getfacl to check the results:

```
> hdfs dfs -getfacl /monthly-sales-data
# file: /monthly-sales-data
# owner: bruce
# group: sales
user::rwx
user:diana:---
group::r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:execs:r-x
```

```
default:mask::r-x
default:other::---
```

It is important to keep in mind the order of evaluation for ACL entries when a user attempts to access a file system object:

- If the user is the file owner, the Owner Permission Bits are enforced.
- Else, if the user has a named user ACL entry, those permissions are enforced.
- Else, if the user is a member of the file's group or any named group in an ACL entry, then the union of permissions for all matching entries are enforced. (The user may be a member of multiple groups.)
- If none of the above are applicable, the Other Permission Bits are enforced.

In this example, the named user ACL entry accomplished our goal because the user is not the file owner and the named user entry takes precedence over all other entries.

ACLS on HDFS features

ACLs on HDFS support features such as associating with any files or directories, access through multiple user-facing endpoints, backward compatibility with Permission Bits and so on.

POSIX ACL Implementation

ACLs on HDFS have been implemented with the POSIX ACL model. If you have ever used POSIX ACLs on a Linux file system, the HDFS ACLs work the same way.

Compatibility and Enforcement

HDFS can associate an optional ACL with any file or directory. All HDFS operations that enforce permissions expressed with Permission Bits must also enforce any ACL that is defined for the file or directory. Any existing logic that bypasses Permission Bits enforcement also bypasses ACLs. This includes the HDFS super-user and setting dfs.permissions to "false" in the configuration.

Access Through Multiple User-Facing Endpoints

HDFS supports operations for setting and getting the ACL associated with a file or directory. These operations are accessible through multiple user-facing endpoints. These endpoints include the FsShell CLI, programmatic manipulation through the FileSystem and FileContext classes, WebHDFS, and NFS.

User Feedback: CLI Indicator for ACLs

The plus symbol (+) is appended to the listed permissions of any file or directory with an associated ACL. To view, use the ls -l command.

Backward-Compatibility

The implementation of ACLs is backward-compatible with existing usage of Permission Bits. Changes applied via Permission Bits (chmod) are also visible as changes in the ACL. Likewise, changes applied to ACL entries for the base user classes (Owner, Group, and Others) are also visible as changes in the Permission Bits. Permission Bit and ACL operations manipulate a shared model, and the Permission Bit operations can be considered a subset of the ACL operations.

Low Overhead

The addition of ACLs will not cause a detrimental impact to the consumption of system resources in deployments that choose not to use ACLs. This includes CPU, memory, disk, and network bandwidth.

Using ACLs does impact NameNode performance. It is therefore recommended that you use Permission Bits, if adequate, before using ACLs.

ACL Entry Limits

The number of entries in a single ACL is capped at a maximum of 32. Attempts to add ACL entries over the maximum will fail with a user-facing error. This is done for two reasons: to simplify management, and to limit

resource consumption. ACLs with a very high number of entries tend to become difficult to understand, and may indicate that the requirements are better addressed by defining additional groups or users. ACLs with a very high number of entries also require more memory and storage, and take longer to evaluate on each permission check. The number 32 is consistent with the maximum number of ACL entries enforced by the "ext" family of file systems.

Snapshots

Within a snapshot, all ACLs are frozen at the moment that the snapshot was created. ACL changes in the parent of the snapshot are not applied to the snapshot.

Tooling

Tooling that propagates Permission Bits will propagate ACLs. This includes the cp -p shell command and distcp -p.

Use cases for ACLs on HDFS

Depending on your requirements, you can configure ACLs on HDFS to ensure that the right users and groups have the required access permissions to your data.

The following examples indicate different use cases for configuring ACLs on HDFS:

Multiple Users

In this use case, multiple users require Read access to a file. None of the users are the owner of the file. The users are not members of a common group, so it is impossible to use group Permission Bits.

This use case can be addressed by setting an access ACL containing multiple named user entries:

```
ACLs on HDFS supports the following use cases:
```

Multiple Groups

In this use case, multiple groups require Read and Write access to a file. There is no group containing all of the group members, so it is impossible to use group Permission Bits.

This use case can be addressed by setting an access ACL containing multiple named group entries:

```
group:sales:rw-
group:execs:rw-
```

Hive Partitioned Tables

In this use case, Hive contains a partitioned table of sales data. The partition key is "country". Hive persists partitioned tables using a separate subdirectory for each distinct value of the partition key, so the file system structure in HDFS looks like this:

```
user
`-- hive
 `-- warehouse
 `-- sales
 |-- country=CN
 |-- country=GB
 `-- country=US
```

All of these files belong to the "salesadmin" group. Members of this group have Read and Write access to all files. Separate country groups can run Hive queries that only read data for a specific country, such as "sales_CN", "sales_GB", and "sales_US". These groups do not have Write access.

This use case can be addressed by setting an access ACL on each subdirectory containing an owning group entry and a named group entry:

```
country=CN
```

```
group::rwx
group:sales_CN:r-x

country=GB
group::rwx
group:sales_GB:r-x

country=US
group::rwx
group:sales_US:r-x
```

Note that the functionality of the owning group ACL entry (the group entry with no name) is equivalent to setting Permission Bits.

Note:

Storage-based authorization in Hive does not currently consider the ACL permissions in HDFS. Rather, it verifies access using the traditional POSIX permissions model.

Default ACLs

In this use case, a file system administrator or sub-tree owner would like to define an access policy that will be applied to the entire sub-tree. This access policy must apply not only to the current set of files and directories, but also to any new files and directories that are added later.

This use case can be addressed by setting a default ACL on the directory. The default ACL can contain any arbitrary combination of entries. For example:

```
default:user::rwx
default:user:bruce:rw-
default:user:diana:r--
default:user:clark:rw-
default:group::r--
default:group:sales::rw-
default:group:execs::rw-
default:others::---
```

It is important to note that the default ACL gets copied from the directory to newly created child files and directories at time of creation of the child file or directory. If you change the default ACL on a directory, that will have no effect on the ACL of the files and subdirectories that already exist within the directory. Default ACLs are never considered during permission enforcement. They are only used to define the ACL that new files and subdirectories will receive automatically when they are created.

Minimal ACL/Permissions Only

HDFS ACLs support deployments that may want to use only Permission Bits and not ACLs with named user and group entries. Permission Bits are equivalent to a minimal ACL containing only 3 entries. For example:

```
user::rw-
group::r--
others::---
```

Block Access to a Sub-Tree for a Specific User

In this use case, a deeply nested file system sub-tree was created as world-readable, followed by a subsequent requirement to block access for a specific user to all files in that sub-tree.

This use case can be addressed by setting an ACL on the root of the sub-tree with a named user entry that strips all access from the user.

For this file system structure:

```
dir1
`-- dir2
```

```
  `-- dir3
  |-- file1
  |-- file2
  `-- file3
```

Setting the following ACL on "dir2" blocks access for Bruce to "dir3,""file1,""file2," and "file3":

```
user:bruce:---
```

More specifically, the removal of execute permissions on "dir2" means that Bruce cannot access "dir2", and therefore cannot see any of its children. This also means that access is blocked automatically for any new files added under "dir2". If a "file4" is created under "dir3", Bruce will not be able to access it.

ACLs with Sticky Bit

In this use case, multiple named users or named groups require full access to a shared directory, such as "/tmp". However, Write and Execute permissions on the directory also give users the ability to delete or rename any files in the directory, even files created by other users. Users must be restricted so that they are only allowed to delete or rename files that they created.

This use case can be addressed by combining an ACL with the sticky bit. The sticky bit is existing functionality that currently works with Permission Bits. It will continue to work as expected in combination with ACLs.

Enable authorization for HDFS web UIs

You can enforce authorization for the following HDFS web UIs: the NameNode, DataNode, and JournalNode.

Before you begin

You must have Kerberos authentication for HTTP web consoles and Hadoop Secure Authorization enabled. When both configurations are set, only the hdfs user can access the HDFS web UIs by default. Any other user who attempts to access the web UI will encounter an error because the user is not authorized to access the page.

For users and groups other than hdfs to access the web UIs, you must add them to hdfs-site.xml with an HDFS Service Advanced Configuration Snippet (Safety Valve).

Procedure

1. In the , go to Clusters <*HDFS SERVICE*> .
2. Navigate to the Configurations tab and search for the following property: HDFS Service Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml.
3. Add a value for the dfs.cluster.administrators property.

   For example, a sample property might look like this:

   - Name: dfs.cluster.administrators
   - Description: ACL for the admins, this configuration is used to control who can access the default servlets in the namenode and so on. The value should be a comma separated list of users and groups. The user list comes first and is separated by a space followed by the group list. For example, user1,user2 group1,group2. Both users and groups are optional. So "user1", " group1", "", "user1 group1", "user1,user2 group1,group2" are all valid. You must note the leading space in " group1". '*' grants access to all users and groups, for example, ", ' ' and ' *' are all valid.

   These values would allow the users and groups to the following web UIs: NameNode, DataNode, and JournalNode.

   Note: You can view the Service Monitor Kerberos Principal by navigating to >Configuration and searching for Role-Specific Kerberos Principal.

4. Save the configuration.

5. Restart all stale HDFS services.

Enable authorization for additional HDFS web UIs

You can enforce authorization for the following HDFS web UI servlets, which may contain sensitive data: /jmx, /stack, /conf, and /metrics. When you enforce authorization for the servlets, only the users listed in the dfs.cluster.administrators property can access them.

Before you begin

requires access to the /jmx and /metrics servlets and uses the HTTP user as well as the Service Monitor Kerberos Principal to access them. Make sure to add both users to dfs.cluster.administrators, as mentioned in Enable authorization for HDFS web UIs on page 11.

Procedure

1. In the , go to Clusters <*HDFS SERVICE*> .
2. Navigate to the Configurations tab and search for the following property: HDFS Service Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml.
3. Add the hadoop.security.instrumentation.requires.admin property and set its value to true.
4. Save the configuration.
5. Restart all stale HDFS services.

Configuring HSTS for HDFS Web UIs

Configuring the HTTP Strict Transport Security (HSTS) for HDFS ensures that a web browser does not load the service information using HTTP. Additionally, all attempts to load the information using HTTP will automatically be converted to HTTPS.

Procedure

1. Go to the HDFS service.
2. Click Configuration.
3. Set the HSTS credentials in Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml.

```
<property>
<name>hadoop.http.header.Strict-Transport-Security</name>
<value>max-age=63072000;includeSubDomains;preload</value>
</property>
```

4. Restart the HDFS service.

Cloudera Runtime 7.3.1

HDFS Overview

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction 4
Overview of HDFS 4

NameNodes 4
Moving NameNode roles 5
Moving highly available NameNode, failover controller, and JournalNode roles using the Migrate Roles wizard 5
Moving a NameNode to a different host using Cloudera Manager 6
Sizing NameNode heap memory 7
Environment variables for sizing NameNode heap memory 7
Monitoring heap memory usage 7
Files and directories 7
Disk space versus namespace 8
Replication 8
Examples of estimating NameNode heap memory 9
Remove or add storage directories for NameNode data directories 10

DataNodes 10
How NameNode manages blocks on a failed DataNode 10
Replace a disk on a DataNode host 11
Remove a DataNode 11
Fixing block inconsistencies 12
Add storage directories using Cloudera Manager 12
Remove storage directories using Cloudera Manager 12
Configuring storage balancing for DataNodes 13
Configure storage balancing for DataNodes using Cloudera Manager 13
Perform a disk hot swap for DataNodes using Cloudera Manager 14

JournalNodes 15
Moving the JournalNode edits directory for a role group using Cloudera Manager 15
Moving the JournalNode edits directory for a role instance using Cloudera Manager 16
Synchronizing the contents of JournalNodes 17

Multiple NameNodes overview 17
Multiple Namenodes configurations 18
Known issue and its workaround 18
Adding multiple namenodes using the HDFS service 18

Introduction

Hadoop Distributed File System (HDFS) is a Java-based file system that provides scalable and reliable data storage. An HDFS cluster contains a NameNode to manage the cluster namespace and DataNodes to store data.

Overview of HDFS

Hadoop Distributed File System (HDFS) is a Java-based file system for storing large volumes of data. Designed to span large clusters of commodity servers, HDFS provides scalable and reliable data storage.

HDFS forms the data management layer of Apache Hadoop. YARN provides the resource management while HDFS provides the storage.

HDFS is a scalable, fault-tolerant, distributed storage system that works closely with a wide variety of concurrent data access applications. By distributing storage and computation across many servers, the combined storage resource grows linearly with demand.

Components of an HDFS cluster

An HDFS cluster contains the following main components: a NameNode and DataNodes.

The NameNode manages the cluster metadata that includes file and directory structures, permissions, modifications, and disk space quotas. The file content is split into multiple data blocks, with each block replicated at multiple DataNodes.

The NameNode actively monitors the number of replicas of a block. In addition, the NameNode maintains the namespace tree and the mapping of blocks to DataNodes, holding the entire namespace image in RAM.

High-Availability (HA) clusters contain a standby for the active NameNode to avoid a single point of failure. These clusters use JournalNodes to synchronize the active and standby NameNodes.

Benefits of HDFS

HDFS provides the following benefits as a result of which data is stored efficiently and is highly available in the cluster:

- Rack awareness: A node's physical location is considered when allocating storage and scheduling tasks.
- Minimal data motion: Hadoop moves compute processes to the data on HDFS. Processing tasks can occur on the physical node where the data resides. This significantly reduces network I/O and provides very high aggregate bandwidth.
- Utilities: Dynamically diagnose the health of the file system and rebalance the data on different nodes.
- Version rollback: Allows operators to perform a rollback to the previous version of HDFS after an upgrade, in case of human or systemic errors.
- Standby NameNode: Provides redundancy and supports high availability (HA).
- Operability: HDFS requires minimal operator intervention, allowing a single operator to maintain a cluster of thousands of nodes

Related Information

Apache Hadoop HDFS

NameNodes

NameNodes maintain the namespace tree for HDFS and a mapping of file blocks to DataNodes where the data is stored.

A simple HDFS cluster can have only one primary NameNode, supported by a secondary NameNode that periodically compresses the NameNode edits log file that contains a list of HDFS metadata modifications. This reduces the amount of disk space consumed by the log file on the NameNode, which also reduces the restart time for the primary NameNode. A high-availability cluster contains two NameNodes: active and standby.

Moving NameNode roles

Depending on your cluster configuration, you can follow either of the two approaches to migrate the NameNode role: using the Cloudera Manager wizard in a highly available cluster to automate the migration process, or manually moving the NameNode role to a new host.

Important: Both procedures require cluster downtime.

Moving highly available NameNode, failover controller, and JournalNode roles using the Migrate Roles wizard

The Migrate Roles wizard allows you to move roles of a highly available HDFS service from one host to another. You can use it to move NameNode, JournalNode, and Failover Controller roles.

Before you begin

- This procedure requires cluster downtime, not a shutdown. The services discussed in this list must be running for the migration to complete.
- The configuration of HDFS and services that depend on it must be valid.
- The destination host must be commissioned and healthy.
- The NameNode must be highly available using quorum-based storage.
- HDFS automatic failover must be enabled, and the cluster must have a running ZooKeeper service.
- If a Hue service is present in the cluster, its HDFS Web Interface Role property must refer to an HttpFS role, not to a NameNode role.
- A majority of configured JournalNode roles must be running.
- The Failover Controller role that is not located on the source host must be running.
- On hosts running active and standby NameNodes, back up the data directories.
- On hosts running JournalNodes, back up the JournalNode edits directory.
- If the source host is not functioning properly, or is not reliably reachable, decommission the host.
- If Cloudera and HDFS metadata was recently upgraded, and the metadata upgrade was not finalized, finalize the metadata upgrade.
- Ensure all Kerberos principals for old and new hosts are available before launching the wizard. If the Kerberos principals are missing, you have to start over the wizard from the beginning after fixing the Kerberos issue. Currently, there are no prechecks available for this HDFS role migration wizard tool.

About this task

Note: Nameservice federation (multiple namespaces) is not supported.

Procedure

1. If the host to which you want to move the NameNode is not in the cluster, add the required host.
2. Go to the HDFS service.
3. Click the Instances tab.
4. Click the Migrate Roles button.

5. Click the Source Host text field and specify the host running the roles to migrate.

   In the Search field optionally enter hostnames to filter the list of hosts and click Search.

   Select the checkboxes next to the desired host. The list of available roles to migrate displays. Clear any roles you do not want to migrate. When migrating a NameNode, the co-located Failover Controller must be migrated as well.
6. Click the Destination Host text field and specify the host to which the roles will be migrated.

   On destination hosts, indicate whether to delete data in the NameNode data directories and JournalNode edits directory. If you choose not to delete data and such role data exists, the Migrate Roles command will not complete successfully.
7. Acknowledge that the migration process incurs service unavailability by selecting the Yes, I am ready to restart the cluster now checkbox.
8. Click Continue. The Command Progress screen displays listing each step in the migration process.
9. When the migration completes, click Finish.

What to do next

After moving a NameNode, if you have a Hive or Impala service, perform the following steps:

1. Go to the Hive service.
2. Stop the Hive service.
3. Select Actions Update Hive Metastore NameNodes .
4. If you have an Impala service, restart the Impala service or run an INVALIDATE METADATA query.

Note: The Update Hive Metastore NameNodes step only needs to be run if you have non-HA namenodes, or if the nameservice identifier is changed.

Moving a NameNode to a different host using Cloudera Manager

You can use Cloudera Manager to manually move a NameNode from one host to another.

Before you begin

This procedure requires cluster downtime.

Procedure

1. If the host to which you want to move the NameNode is not in the cluster, add the required host.
2. Stop all the cluster services.
3. Make a backup of the dfs.name.dir directories on the existing NameNode host.

   Make sure you back up the fsimage and edits files. They should be the same across all of the directories specified by the dfs.name.dir property.
4. Copy the files you backed up from dfs.name.dir directories on the old NameNode host to the host where you want to run the NameNode.
5. Go to the HDFS service.
6. Click the Instances tab.
7. Select the checkbox next to the NameNode role instance and then click the Delete button. Click Delete again to confirm.
8. In the Review configuration changes page that appears, click Skip.
9. Click Add Role Instances to add a NameNode role instance.
10. Select the host where you want to run the NameNode and then click Continue.
11. Specify the location of the dfs.name.dir directories where you copied the data on the new host, and then click Accept Changes.

12. Start the cluster services.

    After the HDFS service has started, Cloudera Manager distributes the new configuration files to the DataNodes, which will be configured with the IP address of the new NameNode host.

Sizing NameNode heap memory

Each workload has a unique byte-distribution profile. Some workloads can use the default JVM settings for heap memory and garbage collection, but others require tuning. You can size your NameNode JVM if the dynamic heap settings cause a bottleneck.

All Hadoop processes run on a Java Virtual Machine (JVM). Each daemon runs on its own JVM across a cluster of hosts.

The legacy NameNode configuration is one active (and primary) NameNode for the entire namespace and one Secondary NameNode for checkpoints (but not failover). The recommended high-availability configuration replaces the Secondary NameNode with a standby NameNode that prevents a single point of failure. Each NameNode uses its own JVM.

Note: For optimal performance, a maximum of 300 million files on the NameNode is recommended.

Environment variables for sizing NameNode heap memory

You can configure the values of *HADOOP_HEAPSIZE* and *HADOOP_NAMENODE_OPTS* to size the NameNode heap memory.

HADOOP_HEAPSIZE sets the JVM heap size for all Hadoop project servers such as HDFS, YARN, and MapReduce. *HADOOP_HEAPSIZE* is an integer passed to the JVM as the maximum memory (Xmx) argument. For example:

```
HADOOP_HEAPSIZE=1024
```

HADOOP_NAMENODE_OPTS is specific to the NameNode and sets all JVM flags, which must be specified. *HADOOP_NAMENODE_OPTS* overrides the *HADOOP_HEAPSIZE* Xmx value for the NameNode. For example:

```
HADOOP_NAMENODE_OPTS=-Xms1024m -Xmx1024m -XX:+UseParNewGC -XX:+UseConcMarkSw
eepGC -XX:CMSInitiatingOccupancyFraction=70 -XX:+CMSParallelRemarkEnabled -X
X:+PrintTenuringDistribution -XX:OnOutOfMemoryError={{AGENT_COMMON_DIR}}/kil
lparent.sh
```

Both *HADOOP_NAMENODE_OPTS* and *HADOOP_HEAPSIZE* are stored in /etc/hadoop/conf/hadoop-env.sh.

Monitoring heap memory usage

You can use several ways to monitor the heap memory usage: Cloudera Manager, NameNode web UI, or the command line.

- Cloudera Manager Look at the NameNode chart for heap memory usage. If you need to build the chart from scratch, run:

  ```
  select jvm_max_memory_mb, jvm_heap_used_mb where roleType="NameNode"
  ```

- NameNode Web UI: Scroll down to the Summary and look for "Heap Memory used."
- Command line: Generate a heap dump.

Files and directories

Persistence of HDFS metadata is implemented using fsimage file and edits files.

Attention:

Do not attempt to modify metadata directories or files. Unexpected modifications can cause HDFS downtime, or even permanent data loss. This information is provided for educational purposes only.

Persistence of HDFS metadata broadly consist of two categories of files:

fsimage

Contains the complete state of the file system at a point in time. Every file system modification is assigned a unique, monotonically increasing transaction ID. An fsimage file represents the file system state after all modifications up to a specific transaction ID.

edits file

Contains a log that lists each file system change (file creation, deletion or modification) that was made after the most recent fsimage.

Checkpointing is the process of merging the content of the most recent fsimage, with all edits applied after that fsimage is merged, to create a new fsimage. Checkpointing is triggered automatically by configuration policies or manually by HDFS administration commands.

Disk space versus namespace

The HDFS default block size (*DFS.BLOCKSIZE*) is set to 128 MB. Each namespace object on the NameNode consumes approximately 150 bytes.

On DataNodes, data files are measured by disk space consumed—the actual data length—and not necessarily the full block size. For example, a file that is 192 MB consumes 192 MB of disk space and not some integral multiple of the block size. Using the default block size of 128 MB, a file of 192 MB is split into two block files, one 128 MB file and one 64 MB file. On the NameNode, namespace objects are measured by the number of files and blocks. The same 192 MB file is represented by three namespace objects (1 file inode + 2 blocks) and consumes approximately 450 bytes of memory.

Large files split into fewer blocks generally consume less memory than small files that generate many blocks. One data file of 128 MB is represented by two namespace objects on the NameNode (1 file inode + 1 block) and consumes approximately 300 bytes of memory. By contrast, 128 files of 1 MB each are represented by 256 namespace objects (128 file inodes + 128 blocks) and consume approximately 38,400 bytes. The optimal split size, then, is some integral multiple of the block size, for memory management as well as data locality optimization.

By default, Cloudera Manager allocates a maximum heap space of 1 GB for every million blocks (but never less than 1 GB). How much memory you actually need depends on your workload, especially on the number of files, directories, and blocks generated in each namespace. If all of your files are split at the block size, you could allocate 1 GB for every million files. But given the historical average of 1.5 blocks per file (2 block objects), a more conservative estimate is 1 GB of memory for every million blocks.

Important: Cloudera recommends 1 GB of NameNode heap space per million blocks to account for the namespace objects, necessary bookkeeping data structures, and the remote procedure call (RPC) workload. In practice, your heap requirements will likely be less than this conservative estimate.

Replication

The default block replication factor (*DFS.REPLICATION*) is three. Replication affects disk space but not memory consumption.

Replication changes the amount of storage required for each block but not the number of blocks. If one block file on a DataNode, represented by one block on the NameNode, is replicated three times, the number of block files is tripled but not the number of blocks that represent them.

With replication off, one file of 192 MB consumes 192 MB of disk space and approximately 450 bytes of memory. If you have one million of these files, or 192 TB of data, you need 192 TB of disk space and, without considering the RPC workload, 450 MB of memory: (1 million inodes + 2 million blocks) * 150 bytes. With default replication on, you need 576 TB of disk space: (192 TB * 3) but the memory usage stay the same, 450 MB. When you account for

bookkeeping and RPCs, and follow the recommendation of 1 GB of heap memory for every million blocks, a much safer estimate for this scenario is 2 GB of memory (with or without replication).

Examples of estimating NameNode heap memory

You can estimate the heap memory by consider either the number of file inodes and blocks, or the cluster capacity.

Example 1: Estimating NameNode heap memory used

Alice, Bob, and Carl each have 1 GB (1024 MB) of data on disk, but sliced into differently sized files. Alice and Bob have files that are some integral of the block size and require the least memory. Carl does not and fills the heap with unnecessary namespace objects.

Alice: 1 x 1024 MB file

- 1 file inode
- 8 blocks (1024 MB / 128 MB)

Total = 9 objects * 150 bytes = 1,350 bytes of heap memory

Bob: 8 x 128 MB files

- 8 file inodes
- 8 blocks

Total = 16 objects * 150 bytes = 2,400 bytes of heap memory

Carl: 1,024 x 1 MB files

- 1,024 file inodes
- 1,024 blocks

Total = 2,048 objects * 150 bytes = 307,200 bytes of heap memory

Example 2: Estimating NameNode heap memory needed

In this example, memory is estimated by considering the capacity of a cluster. Values are rounded. Both clusters physically store 4800 TB, or approximately 36 million block files (at the default block size). Replication determines how many namespace blocks represent these block files.

Cluster A: 200 hosts of 24 TB each = 4800 TB.

- Blocksize=128 MB, Replication=1
- Cluster capacity in MB: 200 * 24,000,000 MB = 4,800,000,000 MB (4800 TB)
- Disk space needed per block: 128 MB per block * 1 = 128 MB storage per block
- Cluster capacity in blocks: 4,800,000,000 MB / 128 MB = 36,000,000 blocks

At capacity, with the recommended allocation of 1 GB of memory per million blocks, Cluster A needs 36 GB of maximum heap space.

Cluster B: 200 hosts of 24 TB each = 4800 TB.

- Blocksize=128 MB, Replication=3
- Cluster capacity in MB: 200 * 24,000,000 MB = 4,800,000,000 MB (4800 TB)
- Disk space needed per block: 128 MB per block * 3 = 384 MB storage per block
- Cluster capacity in blocks: 4,800,000,000 MB / 384 MB = 12,000,000 blocks

At capacity, with the recommended allocation of 1 GB of memory per million blocks, Cluster B needs 12 GB of maximum heap space.

Both Cluster A and Cluster B store the same number of block files. In Cluster A, however, each block file is unique and represented by one block on the NameNode; in Cluster B, only one-third are unique and two-thirds are replicas.

Remove or add storage directories for NameNode data directories

This is a cluster administration task where existing Namenode DataDir must be changed. Perform the following steps to change the Namenode directories with downtime.

Procedure

1. Take a backup of the existing DataDir paths on NNs. For instance, for each NN, if there are two DataDir paths, taking backup of one DataDir is sufficient.
2. Create new mounts where the Namenode data will reside and ensure the permissions and ownerships are all same as that of the old Namenode data directories.
3. Shutdown the Standby Namenode.
4. Log in to the host and copy the NN DataDir content from old mount to new mount.
5. Verify the permission and ownership of fsimage* edits* VERSION files.
6. Navigate to Cloudera Manager > HDFS > Instances > click on the SBNN role (stopped one) > Configuration > search for dfs.namenode.name.dir > change the mount path to new one > Save Changes
7. Start the SBNN NameNode.
8. Wait for the SBNN to start and is out of safe mode.
9. Navigate to Cloudera Manager > HDFS > Configuration > Federation and High Availability > Actions > Manual Failover > select SBNN option > click Manual Failover > wait until the failover is complete.
10. You must verify using the NN webUIs if the previous SBNN node is now the ANN and ANN node is now the SBNN.
11. Repeat steps 2-7 on the new SBNN which was previous ANN.

DataNodes

DataNodes store data in a Hadoop cluster and is the name of the daemon that manages the data. File data is replicated on multiple DataNodes for reliability and so that localized computation can be executed near the data.

Within a cluster, DataNodes should be uniform. If they are not uniform, issues can occur. For example, DataNodes with less memory fill up more quickly than DataNodes with more memory, which can result in job failures.

Important: The default replication factor for HDFS is three. That is, three copies of data are maintained at all times. Cloudera recommends that you do not configure a lower replication factor when you have at least three DataNodes. A lower replication factor may lead to data loss.

How NameNode manages blocks on a failed DataNode

A DataNode is considered dead after a set period without any heartbeats (10.5 minutes by default).

When this happens, the NameNode performs the following actions to maintain the configured replication factor (3x replication by default):

1. The NameNode determines which blocks were on the failed DataNode.
2. The NameNode locates other DataNodes with copies of these blocks.
3. The DataNodes with block copies are instructed to copy those blocks to other DataNodes to maintain the configured replication factor.

Keep the following in mind when working with dead DataNodes:

- If a DataNode fails to heartbeat for reasons other than disk failure, it needs to be recommissioned to be added back to the cluster.

- If a DataNode rejoins the cluster, there is a possibility for surplus replicas of blocks that were on that DataNode. The NameNode will randomly remove excess replicas adhering to Rack-Awareness policies.

Replace a disk on a DataNode host

In your Cloudera Private Cloud Base cluster, you can replace faulty disks on the DataNode host. You must stop all the managed services and decommission the DataNode role instance before replacing the faulty disk.

Before you begin

Procedure

1. Stop managed services.
2. Decommission the DataNode role instance.
3. Replace the failed disk.
4. Recommission the DataNode role instance.
5. Run the HDFS fsck utility to validate the health of HDFS.

   The utility normally reports over-replicated blocks immediately after a DataNode is reintroduced to the cluster, which is automatically corrected over time.
6. Start managed services.

Remove a DataNode

Before removing a DataNode, ensure that all the prerequisites for removal are satisfied.

Before you begin

- The number of DataNodes in your cluster must be greater than or equal to the replication factor you have configured for HDFS. (This value is typically 3.)

  In order to satisfy this requirement, add the DataNode role on other hosts as required and start the role instances before removing any DataNodes.
- Ensure the DataNode that is to be removed is running.

Procedure

1. Decommission the DataNode role.

   When asked to select the role instance to decommission, select the DataNode role instance.

   The decommissioning process moves the data blocks to the other available DataNodes.

   Important: There must be at least as many DataNodes running as the replication factor or the decommissioning process will not complete.
2. Stop the DataNode role.

   When asked to select the role instance to stop, select the DataNode role instance.
3. Verify the integrity of the HDFS service.

   a) Run the following command to identify any problems in the HDFS file system:

   ```
   hdfs fsck /
   ```

   b) Fix any errors reported by the fsck command.

   If required, create a Cloudera support case.

4. After all errors are resolved, perform the following steps.
   a) Remove the DataNode role.
   b) Manually remove the DataNode data directories.

      You can determine the location of these directories by examining the DataNode Data Directory property in the HDFS configuration. In Cloudera Manager, go to the HDFS service, select the Configuration tab and search for the property.

Fixing block inconsistencies

You can use the output of `hdfs fsck` or `hdfs dfsadmin -report` commands for information about inconsistencies with the HDFS data blocks such as missing, misreplicated, or underreplicated blocks. You can adopt different methods to address these inconsistencies.

Missing blocks: Ensure that the DataNodes in your cluster and the disks running on them are healthy. This should help in recovering those blocks that have recoverable replicas (indicated as missing blocks). If a file contains corrupt or missing blocks that cannot be recovered, then the file would be missing data, and all this data starting from the missing block becomes inaccessible through the CLI tools and FileSystem API. In most cases, the only solution is to delete the data file (by using the `hdfs fsck <path> -delete` command) and recover the data from another source.

Important: You must exercise caution when deleting files to ensure that you do not lose any important data. Cloudera recommends that you contact support for further assistance or any guidance on deleting files.

Underreplicated blocks: HDFS automatically attempts to fix this issue by replicating the underreplicated blocks to other DataNodes and match the replication factor. If the automatic replication does not work, you can run the HDFS Balancer to address the issue.

Misreplicated blocks: Run the `hdfs fsck -replicate` command to trigger the replication of misreplicated blocks. This ensures that the blocks are correctly replicated across racks in the cluster.

Add storage directories using Cloudera Manager

You can add a new storage directory and specify the storage type using Cloudera Manager.

Procedure

1. Go to the HDFS service.
2. Click the Configuration tab.
3. Select Scope DataNode .
4. Add the new storage directory to the DataNode Data Directory property.

   To specify the storage type for the HDFS heterogeneous storage, add the storage type, surrounded by brackets, at the front of the path. For example: [SSD]/data/example_dir/.
5. Enter a Reason for change, and then click Save Changes to commit the changes.
6. Restart the DataNode.

   Important: You must restart the DataNodes for heterogeneous storage configuration changes to take effect.

Remove storage directories using Cloudera Manager

You can use Cloudera Manager to remove existing storage directories and specify new directories.

Procedure

1. Stop the cluster.
2. Go to the HDFS service.
3. Click the Configuration tab.
4. Select Scope DataNode .
5. Remove the current directories and add new ones to the DataNode Data Directory property.
6. Enter a Reason for change, and then click Save Changes to commit the changes.
7. Copy the contents under the old directory to the new directory.
8. Start the cluster.

Configuring storage balancing for DataNodes

You can configure HDFS to distribute writes on each DataNode in a manner that balances out available storage among that DataNode's disk volumes.

By default a DataNode writes new block replicas to disk volumes solely on a round-robin basis. You can configure a volume-choosing policy that causes the DataNode to take into account how much space is available on each volume when deciding where to place a new replica.

You can configure

- how much DataNode volumes are allowed to differ in terms of bytes of free disk space before they are considered imbalanced, and
- what percentage of new block allocations will be sent to volumes with more available disk space than others.

Configure storage balancing for DataNodes using Cloudera Manager

You must configure certain dfs.datanode properties to enable storage balancing for DataNodes.

Procedure

1. Go to the HDFS service.
2. Click the Configuration tab.
3. Select ScopeDataNode.
4. Select CategoryAdvanced.

5. Configure the following properties (you can use the Search box to locate the properties).

Property	Value	Description
`dfs.datanode.fsdataset.volume.choosing.policy`	`org.apache.hadoop.hdfs.server.datanode.fsdataset.AvailableSpaceVolumeChoosingPolicy`	Enables storage balancing among the DataNode's volumes.
`dfs.datanode.available-space-volume-choosing-policy.balanced-space-threshold`	10737418240 (default)	The amount by which volumes are allowed to differ from each other in terms of bytes of free disk space before they are considered imbalanced. The default is 10737418240 (10 GB). If the free space on each volume is within this range of the other volumes, the volumes will be considered balanced and block assignments will be done on a pure round-robin basis.
`dfs.datanode.available-space-volume-choosing-policy.balanced-space-preference-fraction`	0.75 (default)	What proportion of new block allocations will be sent to volumes with more available disk space than others. The allowable range is 0.0-1.0, but set it in the range 0.5 - 1.0 (that is, 50-100%), since there should be no reason to prefer that volumes with less available disk space receive more block allocations.

6. Enter a Reason for change, and then click Save Changes to commit the changes.
7. Restart the role.

Perform a disk hot swap for DataNodes using Cloudera Manager

You can replace disks on your Cloudera Private Cloud Base cluster without shutting down a DataNode. This is referred to as hot swap.

About this task

Caution:

- Hot swap can only add disks with empty data directories.
- Removing a disk does not move the data off the disk, which could potentially result in data loss.
- Do not perform hot swap on multiple hosts at the same time.

Procedure

1. Configure data directories to remove the disk you are swapping out.
   a) Go to the HDFS service.
   b) Click the Instances tab.
   c) In the Role Type column, click on the affected DataNode.
   d) Click the Configuration tab.
   e) Select Scope DataNode .
   f) Select Category Main .
   g) Change the value of the DataNode Data Directory property to remove the directories that are mount points for the disk you are removing.

      Warning: Change the value of this property only for the specific DataNode instance where you are planning to hot swap the disk. Do not edit the role group value for this property. Doing so will cause data loss.
2. Enter a Reason for change, and then click Save Changes to commit the changes.
3. Refresh the affected DataNode. Select ActionsRefresh DataNode configuration.
4. Remove the old disk and add the replacement disk.
5. Change the value of the DataNode Data Directory property to add back the directories that are mount points for the disk you added.
6. Enter a Reason for change, and then click Save Changes to commit the changes.
7. Refresh the affected DataNode. Select Actions > Refresh DataNode configuration.
8. Run the hdfs fsck / command to validate the health of HDFS.

JournalNodes

High-availability clusters use JournalNodes to synchronize active and standby NameNodes. The active NameNode writes to each JournalNode with changes, or "edits," to HDFS namespace metadata. During failover, the standby NameNode applies all edits from the JournalNodes before promoting itself to the active state.

Moving the JournalNode edits directory for a role group using Cloudera Manager

Depending on your requirements, you can change the location of the edits directory for each JournalNode in the JournalNode Default Group.

Procedure

1. Stop all services on the cluster in Cloudera Manager.
   a) Go to the Cluster.
   b) Select Actions Stop .
2. Find the list of JournalNode hosts.
   a) Go to the HDFS service.
   b) Click JournalNode under Status Summary.

3. Move the location of each JournalNode (jn) directory at the command line.
   a) Connect to each host with a JournalNode.
   b) Per host, copy the JournalNode (jn) directory to its new location with the -a option to preserve permissions.

```
cp -a /<old_path_to_jn_dir>/jn /<new_path_to_jn_dir>/jn
```

   c) Per host, rename the old jn directory to avoid confusion.

```
mv /<old_path_to_jn_dir>/jn /<old_path_to_jn_dir>/jn_to_delete
```

4. Reconfigure the JournalNode Default Group.
   a) Go to the HDFS service.
   b) Click the Configuration tab.
   c) Click JournalNode under Scope.
   d) Set dfs.journalnode.edits.dir to the path of the new jn directory for all JournalNodes in the group.
   e) Click Save Changes.
5. Redeploy the client configuration for the cluster.
   a) Go to the Cluster.
   b) Select Actions Deploy Client Configuration .
6. Start all services on the cluster by selecting Actions Start .
7. Delete the old jn_to_delete directories from the command line.

Moving the JournalNode edits directory for a role instance using Cloudera Manager

Depending on your requirements, you can change the location of the edits directory for one JournalNode instance.

Procedure

1. Reconfigure the JournalNode Edits Directory.
   a) Go to the HDFS service in Cloudera Manager.
   b) Click JournalNode under Status Summary.
   c) Click the JournalNode link for the instance you are changing.
   d) Click the Configuration tab.
   e) Set dfs.journalnode.edits.dir to the path of the new jn directory.
2. Move the location of the JournalNode (jn) directory at the command line.
   a) Connect to host of the JournalNode.
   b) Copy the JournalNode (jn) directory to its new location with the -a option to preserve permissions.

```
cp -a /<old_path_to_jn_dir>/jn /<new_path_to_jn_dir>/jn
```

   c) Rename the old jn directory to avoid confusion.

```
mv /<old_path_to_jn_dir>/jn /<old_path_to_jn_dir>/jn_to_delete
```

3. Redeploy the HDFS client configuration.
   a) Go to the HDFS service.
   b) Select Actions Deploy Client Configuration .
4. Perform a rolling restart for HDFS by selecting Actions Rolling Restart .

   Use the default settings.
5. From the command line, delete the old jn_to_delete directory.

Synchronizing the contents of JournalNodes

You can synchronize the contents of the JournalNodes in your Cloudera Private Cloud Base cluster. Enabling this capability helps in maintaining consistency in the contents of all the JournalNodes across the cluster. For example, a JournalNode whose contents go out of synchronization can automatically copy the contents of the other JournalNodes in the cluster to ensure consistency.

Before you begin

The cluster on which you want to enable JournalNode synchronization must have High Availability enabled.

Procedure

1. Go to the HDFS service.
2. Click the Instances tab.
3. Click the link labeled NameNode (Active).

   The NameNode instance page displays.
4. Go to the Process tab.
5. Click the hdfs-site.xml link under hdfs/hdfs.sh ["namenode"] to open a read-only copy of the file in a separate browser window.
6. Note the URI value set for JournalNodes in the various dfs.namenode.shared.edits.dir parameters.

   The URI value should be in the format:

   ```
   qjournal://host1:port1;host2:port2;host3:port3/journalId
   ```
7. Return to the HDFS service page and click the Configuration tab.

   Note: Ensure that you click the Configuration tab on the HDFS service page, *not* on the NameNode role page.
8. Search for the HDFS Service Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml configuration option.
9. Add the key dfs.namenode.shared.edits.dir to the selected configuration option and provide the value noted in Step 6.
10. Click Save Changes.
11. Restart HDFS.

Multiple NameNodes overview

This feature provides Cloudera Private Cloud Base the ability to use multiple standby namenodes.

Prerequisites

- Cloudera Manager version must be 7.7.1 or higher
- Cloudera parcel version must be 7.1.8 or higher
- HDFS should have High Availability enabled
- The latest upgrade of HDFS must be finalized
- The cluster must be healthy according to Cloudera Manager and without any stale configurations

This feature is enabled only for the 7.1.8 clusters. This requires the High Availability feature enabled. After configuring an additional NameNode to the cluster, you cannot roll back either the Cloudera Manager upgrade or the Cloudera parcel upgrade. You must add an additional namenode after the upgrade to 718 is finalized.

Multiple Namenodes configurations

This feature is not enabled by default in Cloudera Manager. You must first enable this feature in Cloudera Manager configuration. You can then configure HDFS service with multiple standby NameNodes.

Enabling multiple namenode support feature in Cloudera Manager

Procedure

1. Navigate to the /etc/default/cloudera-scm-server file.
2. Add export CMF_FF_HDFS_MULTIPLE_SNN=true at the end of the cloudera-scm-server file.
3. Restart the Cloudera Manager Service.

Known issue and its workaround

Before you proceed further configuring additional namenodes, you must be aware of the following known issue.

Issue 1

In case the JournalNodes were restarted on a cluster recently, the Add new NameNode wizard for HDFS service might fail to bootstrap the new NameNode, if there was not a new fsImage created since the restart of the JournalNodes. However, while restarting the JournalNodes the edit logs were rolled in the system.

If the bootstrap fails during the Add new NameNode wizard, you must perform the following steps:

1. Delete the newly added NameNode and FailoverController
2. Move the active HDFS NameNode to safe mode
3. Perform the Save Namespace operation on the active HDFS NameNode
4. Leave safe mode on the active HDFS NameNode
5. Try to add the new NameNode again

Note: Entering safe mode disables write operation to HDFS and causes a service disruption. If you can not afford to enter safe mode, just delete the newly added NameNode and FailoverController in the HDFS service. You must wait until HDFS automatically creates a new fsImage and try adding the new NameNode again with the wizard.

Issue 2

After adding three NameNodes, performance impact like slowness in the cluster is detected while performing heavy read and write operations.

Tune the following parameters:

- dfs.client.failover.sleep.base.millis=1
- dfs.client.failover.max.attempts=25

Adding multiple namenodes using the HDFS service

You can add multiple namenodes using the HDFS service in Cloudera Manager.
You must ensure that Cloudera Manager is in a clean state without any health issues and no stale configurations on the cluster. If there are any health issues or stale configurations, then you must restart the cluster separately from the command and not within the command. For this, you must clear the Rolling Restart HDFS and all dependent services to activate the changes option in step 10b. Also, if there are any health issues or stale configurations that cannot be cleared from the cluster, you must restart the cluster separately.

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the HDFS service
4. Go to Instances
5. Click Federation and High Availability
6. Click Actions
7. Click Add Additional NameNode(s). Add Additional NameNode wizard opens.
8. In the Assign Roles page, select a host which is not assigned to any namenode.
9. Click Continue.
10. In the Review Changes page, you can specify the namenode data directory.
    a) If you want to delete the data present in the namenode data directory you have provided, then you must select the Clear any existing data present in name directories of Standby NameNode option. Remember to take a backup of the namenode data directory.
    b) If you want to restart HDFS service and its dependencies, then you must select the Rolling Restart HDFS and all dependent services to activate the changes option. Remember that if you have not enabled High Availability feature, the rolling restart may cause a cluster downtime. Note: If you do not select the Rolling Restart option, then you can click Continue and restart the cluster later.
11. Click Continue. This runs the required commands and adds the additional namenodes.
12. Click Finish.

Cloudera Runtime 7.3.1

Apache Hive metastore overview

Date published: 2019-08-21
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Hive storage in Public Cloud..... 4

Apache Hive storage in Public Cloud

Knowing the storage locations of Apache Hive data and metadata helps you troubleshoot problems and replicate data.

On Public Cloud, the Hive warehouse that stores Hive data is located in an object store, such as S3, by default. In the cloud, Hive uses HDFS merely for storing temporary files. The Hive Metastore (HMS) stores the schema for Hive tables. HMS uses a pre-installed MySQL database. You perform little, or no, configuration of HMS in the cloud.

Cloudera Runtime 7.3.1

Apache Hive Overview

Date published: 2019-08-21
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Hive features in 4

Spark integration with Hive..... 5

Hive on Tez introduction..... 5

Hive unsupported interfaces and features in Public Cloud..... 5

Apache Hive 3 in architectural overview..... 6

Apache Hive content roadmap..... 8

Apache Hive features in

Major changes to Apache Hive 2.x improve Apache Hive 3.x transactions and security. Knowing the major differences between these versions is critical for SQL users, including those who use Apache Spark and Apache Impala.

Hive is a data warehouse system for summarizing, querying, and analyzing huge, disparate data sets.

ACID transaction processing

Hive 3 tables are ACID (Atomicity, Consistency, Isolation, and Durability)-compliant. Hive 3 write and read operations improve the performance of transactional tables. Atomic operations include simple writes and inserts, writes to multiple partitions, and multiple inserts in a single SELECT statement. A read operation is not affected by changes that occur during the operation. You can insert or delete data, and it remains consistent throughout software and hardware crashes. Creation and maintenance of Hive tables is simplified because there is no longer any need to bucket tables.

Materialized views

Because multiple queries frequently need the same intermediate roll up or joined table, you can avoid costly, repetitious query portion sharing, by precomputing and caching intermediate tables into views.

Query results cache

Hive filters and caches similar or identical queries. Hive does not recompute the data that has not changed. Caching repetitive queries can reduce the load substantially when hundreds or thousands of users of BI tools and web services query Hive.

Scheduled Queries

Using SQL statements, you can schedule Hive queries to run on a recurring basis, monitor query progress, temporarily ignore a query schedule, and limit the number running in parallel. You can use scheduled queries to start compaction and periodically rebuild materialized views, for example.

Security improvements

Apache Ranger secures Hive data by default. To meet demands for concurrency improvements, ACID support, render security, and other features, Hive tightly controls the location of the warehouse on a file system, or object store, and memory resources.

With Apache Ranger and Apache Hive ACID support, your organization will be ready to support and implement GDPR (General Data Protection Regulation).

Connection Pooling

Hive supports HikariCP JDBC connection pooling.

Unsupported features

does not support the following features that were available in HDP and CDH platforms:

- Multiple insert overwrite queries that read data from a source table.
- CREATE TABLE that specifies a managed table location

  Do not use the LOCATION clause to create a managed table. Hive assigns a default location in the warehouse to managed tables.

- CREATE INDEX

  Hive builds and stores indexes in ORC or Parquet within the main table, instead of a different table, automatically. Set hive.optimize.index.filter to enable use (not recommended--use materialized views instead). Existing indexes are preserved and migrated in Parquet or ORC to during upgrade.

Spark integration with Hive

You need to know a little about Hive Warehouse Connector (HWC) and how to find more information because to access Hive from Spark, you need to use HWC implicitly or explicitly.

You can use the Hive Warehouse Connector (HWC) to access Hive managed tables from Spark. HWC is specifically designed to access managed ACID v2 Hive tables, and supports writing to tables in Parquet, ORC, Avro, or Textfile formats. HWC is a Spark library/plugin that is launched with the Spark app.

You do not need HWC to read from or write to Hive external tables. Spark uses native Spark to access external tables.

Use the Spark Direct Reader and HWC for ETL jobs. For other jobs, consider using Apache Ranger and the HiveWarehouseConnector library to provide row and column, fine-grained access to the data.

HWC supports spark-submit and pyspark. The spark thrift server is not supported.

Related Information

Hive Warehouse Connector for accessing Apache Spark data

Blog: Enabling high-speed Spark direct reader for Apache Hive ACID tables

Hive on Tez introduction

The Cloudera service provides an Apache Hive SQL database that Apache Tez executes.

The Hive on Tez service provides a SQL-based data warehouse system based on Apache Hive 3.x. The enhancements in Hive 3.x over previous versions can improve SQL query performance, security, and auditing capabilities. The Hive metastore (HMS) is a separate service, not part of Hive, not even necessarily on the same cluster. HMS stores the metadata on the backend for Hive, Impala, Spark, and other components.

Apache Tez is the Hive execution engine for the Hive on Tez service, which includes HiveServer (HS2) in Cloudera Manager. MapReduce is not supported. In a Cloudera cluster, if a legacy script or application specifies MapReduce for execution, an exception occurs. Most user-defined functions (UDFs) require no change to run on Tez instead of MapReduce.

With expressions of directed acyclic graphs (DAGs) and data transfer primitives, execution of Hive queries on Tez instead of MapReduce improves query performance. In Cloudera, Tez is usually used only by Hive, and launches and manages Tez AM automatically when Hive on Tez starts. SQL queries you submit to Hive are executed as follows:

- Hive compiles the query.
- Tez executes the query.
- Resources are allocated for applications across the cluster.
- Hive updates the data in the data source and returns query results.

Hive unsupported interfaces and features in Public Cloud

You need to understand the interfaces that are not supported in Cloudera.

The following interfaces are not supported:

- Hcat CLI (however, HCatalog is supported)

- Hive CLI (replaced by Beeline)
- Hive View UI feature in Ambari
- Renaming Hive databases
- Multiple insert overwrite queries that read data from a source table.
- MapReduce execution engine (replaced by LLAP)
- Pig
- Spark execution engine
- Spark thrift server

  Spark and Hive tables interoperate using the Hive Warehouse Connector.
- SQL Standard Authorization
- Tez View UI feature in Ambari
- WebHCat

You can use Hue in lieu of Hive View.

Hive-Kudu integration

Cloudera does not support the integration of HiveServer (HS2) with Kudu tables. You cannot run queries against Kudu tables from HS2.

Unsupported Features

Cloudera does not support the following features that were available in HDP and CDH platforms:

- CREATE TABLE that specifies a managed table location

  Do not use the LOCATION clause to create a managed table. Hive assigns a default location in the warehouse to managed tables. That default location is configured in Hive using the hive.metastore.warehouse.dir configuration property, but can be overridden for the database by setting the CREATE DATABASE MANAGEDLOCATION parameter.
- CREATE INDEX and related index commands were removed in Hive 3, and consequently are not supported in Cloudera.

  In Cloudera, you use the Hive 3 default ORC columnar file formats to achieve the performance benefits of indexing. Materialized Views with automatic query rewriting also improves performance. Indexes migrated to Cloudera are preserved but render any Hive tables with an undroppable index. To drop the index, google the Known Issue for CDPD-23041.
- Hive metastore (HMS) high availablility (HA) load balancing in CDH

  You need to set up HMS HA as described in the documentation.
- Local or Embedded Hive metastore server

  Cloudera does not support the use of a local or embedded Hive metastore setup.

Unsupported Connector Use

Cloudera does not support the Sqoop exports using the Hadoop jar command (the Java API) that Teradata documents. For more information, see Migrating data using Sqoop.

Apache Hive 3 in architectural overview

Understanding Apache Hive 3 major design features, such as default ACID transaction processing, can help you use Hive to address the growing needs of enterprise data warehouse systems.

Data storage and access control

One of the major architectural changes to support Hive 3 design gives Hive much more control over metadata memory resources and the file system, or object store. The following architectural changes from Hive 2 to Hive 3 provide improved security:

- Tightly controlled file system and computer memory resources, replacing flexible boundaries: Definitive boundaries increase predictability. Greater file system control improves security.
- Optimized workloads in shared files and containers

Hive 3 is optimized for object stores in the following ways:

- Hive uses ACID to determine which files to read rather than relying on the storage system.
- In Hive 3, file movement is reduced from that in Hive 2.
- Hive caches metadata and data agressively to reduce file system operations

The major authorization model for Hive is Ranger. Hive enforces access controls specified in Ranger. This model offers stronger security than other security schemes and more flexibility in managing policies.

This model permits only Hive to access the Hive warehouse.

Transaction processing

You can deploy new Hive application types by taking advantage of the following transaction processing characteristics:

- Mature versions of ACID transaction processing:

  ACID tables are the default table type.

  ACID enabled by default causes no performance or operational overload.
- Simplified application development, operations with strong transactional guarantees, and simple semantics for SQL commands

  You do not need to bucket ACID tables.
- Materialized view rewrites
- Automatic query cache
- Advanced optimizations

Hive client changes

You can use the thin client Beeline for querying Hive from a client. You can run Hive administrative commands from the client. Beeline uses a JDBC connection to Hive to run commands. Hive parses, compiles, and runs operations. Beeline supports many of the command-line options that Hive CLI supported. Beeline does not support hive -e set key=value to configure the Hive Metastore.

You enter supported Hive CLI commands by invoking Beeline using the hive keyword, command option, and command. For example, hive -e set. Using Beeline instead of the thick client Hive CLI, which is no longer supported, has several advantages, including low overhead. Beeline does not use the entire Hive code base. A small number of daemons required to run queries simplifies monitoring and debugging.

Hive enforces allowlist and denylist settings that you can change using SET commands. Using the denylist, you can restrict memory configuration changes to prevent instability. Different Hive instances with different allowlists and denylists to establish different levels of stability.

Apache Hive Metastore sharing

Hive, Impala, and other components can share a remote Hive metastore.

Query execution of batch and interactive workloads

You can connect to Hive using a JDBC command-line tool, such as Beeline, or using an JDBC/ODBC driver with a BI tool, such as Tableau. You configure the settings file for each instance to perform either batch or interactive processing.

Apache Hive content roadmap

The content roadmap provides links to the available content resources for Apache Hive.

Table 1: Apache Hive Content roadmap

Task	Resources	Source	Description
Understanding	Presentations and Papers about Hive	Apache wiki	Contains meeting notes, presentations, and whitepapers from the Apache community.
Getting Started	Hive Tutorial	Apache wiki	Provides a basic overview of Apache Hive and contains some examples on working with tables, loading data, and querying and inserting data.
Developing	Materialized Views	Apache wiki	Covers accelerating query processing in Cloudera Data Warehouse by pre-computing summaries using materialized views.
	JdbcStorageHandler	Apache wiki	Describes how to read from a JDBC data source in Hive.
	Hive transactions	Apache wiki	Describes ACID operations in Hive.
	Hive Streaming API	Apache wiki	Explains how to use an API for pumping data continuously into Hive using clients such as NiFi and Flume.
	Hive Operators and Functions	Apache wiki	Describes the Language Manual UDF.
	Beeline: HiveServer2 Client	Apache wiki	Describes how to use the Beeline client.
Reference	SQL Language Manual	Apache wiki	Language reference documentation available in the Apache wiki.
Contributing	Hive Developer FAQ	Apache wiki	Resources available if you want to contribute to the Apache community.
	How to Contribute	Apache wiki	
	Hive Developer Guide	Apache wiki	
	Plug-in Developer Kit	Apache wiki	
	Unit Test Parallel Execution	Apache wiki	
	Hive Architecture Overview	Apache wiki	
	Hive Design Docs	Apache wiki	
	Project Bylaws	Apache wiki	

Cloudera Runtime 7.3.1

Working with Apache Hive metastore

Date published: 2019-08-21
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

HMS table storage **4**

Configuring HMS for high availability **5**

Hive Metastore leader election **6**

HMS table storage

You need to understand how the Hive metastore (HMS) stores Hive tables when you run a CREATE TABLE statement or migrate a table to Cloudera. The success or failure of the statement, the resulting table type, and the table location depends on a number of factors.

HMS table transformations

The HMS includes the following Hive metadata about tables that you create:

- A table definition
- Column names
- Data types
- Comments in a central schema repository

When you use the EXTERNAL keyword in the CREATE TABLE statement, HMS stores the table as an external table. When you omit the EXTERNAL keyword and create a managed table, or ingest a managed table, HMS might translate the table into an external table or the table creation can fail, depending on the table properties. An important table property that affects table transformations is the ACID or Non-ACID table type:

Non-ACID

Table properties do not contain any ACID related properties set to true. For example, the table does not contain such properties transactional=true or insert_only=true.

ACID

Table properties do contain one or more ACID properties set to true.

Full ACID

Table properties contain transactional=true but not insert_only=true

Insert-only ACID

Table properties contain insert_only=true.

The following matrix shows the table type and whether or not the location property is supported.

ACID	Managed	Location Property	Comments	Action
Non-ACID	Yes	Yes	Migrated to Cloudera, for example from an HDP or CDH cluster.	Table stored as external
Non-ACID	Yes	No	Table location is null	Table stored in subdirectory of external warehouse*

* metastore.warehouse.external.dir

HMS detects type of client for interacting with HMS, for example Hive or Spark, and compares the capabilities of the client with the table requirement. HMS performs the following actions, depending on the results of the comparison:

Table requirement	Client meets requirements	Managed Table	ACID table type	Action
Client can write to any type of ACID table	No	Yes	Yes	CREATE TABLE fails
Client can write to full ACID table	No	Yes	insert_only=true	CREATE TABLE fails
Client can write to insert-only ACID table	No	Yes	insert_only=true	CREATE TABLE fails

If, for example, a Spark client does not have the capabilities required, the following type of error message appears:

```
Spark has no access to table `mytable`. Clients can access this table only i
f
they have the following capabilities: CONNECTORREAD,HIVEFULLACIDREAD, HIVE
FULLACIDWRITE,
HIVEMANAGESTATS, HIVECACHEINVALIDATE, . . .
```

Configuring HMS for high availability

To provide failover to a secondary Hive metastore if your primary instance goes down, you need to know how to add a Metastore role in Cloudera Manager and configure a property.

About this task

Multiple HMS instances run in active/active mode. Load balancing is done at the Hive client side (like HiveServer or Spark) as the HMS client picks an HMS instance randomly. By default, the hive.metastore.uri.selection property is set to RANDOM. If that HMS instance is down, then the client randomly picks another instance from the list of HMS instances specified through the hive.metastore.uris property.

Before you begin

Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

Procedure

1. In Cloudera Manager, click Clusters Hive Configuration .
2. Take one of the following actions:
   - If you have a cluster secured by Kerberos, search for Hive Delegation Token Store, which specifies storage for the Kerberos token as described below.
   - If you have an unsecured cluster, skip the next step.
3. Select org.apache.hadoop.hive.thrift.DBTokenStore, and save the change.

   Storage for the Kerberos delegation token is defined by the hive.cluster.delegation.token.store.class property. The available choices are Zookeeper, the Metastore, and memory. Cloudera recommends using the database by setting the org.apache.hadoop.hive.thrift.DBTokenStore property.

4. Click Instances Actions Add Role Instances

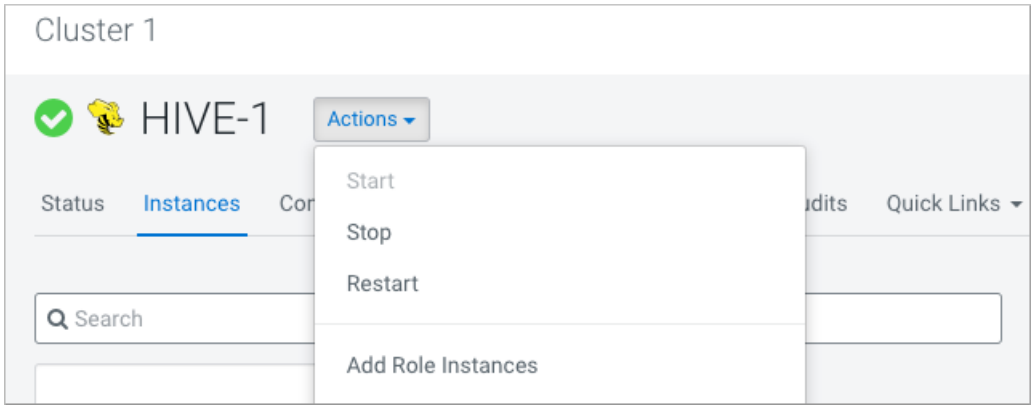

5. In Assign Roles, in Metastore Server, click Select Hosts.
6. In Hosts Selected, scroll and select the host that you want to serve as the backup Metastore, and click OK.
7. Click Continue until you exit the wizard.
8. Start the Metastore role on the host from the Actions menu.

   The hive.metastore.uris property is updated automatically. To verify, go to /etc/hive/config directory in your cluster node and look for the updated property in the hive-site.xml file.
9. To check or to change the hive.metastore.uri.selection property, go to Clusters Hive Configurations in Cloudera Manager, and search for 'Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml'.
10. Add the property and value (SEQUENTIAL or RANDOM).

Related Information

Custom Configuration (about Cloudera Manager Safety Valve)

Hive Metastore leader election

To avoid running the same tasks across all Hive Metastore (HMS) instances, you can configure a HMS leader manually or enable dynamic election.

HMS performs housekeeping tasks, such as execution of compaction tasks, auto-discovering partitions for external tables, generation of compaction tasks, and so on. When there are multiple HMS instances, it is essential to have a single leader HMS elected to avoid running the same tasks across all the instances. The elected leader then performs the housekeeping tasks.

The leader is configured through the metastore.housekeeping.leader.hostname property. The value of this property is an empty string by default, which indicates that the current HMS will run the housekeeping tasks. To configure a leader, specify the host name of the HMS service. However, configuring a leader HMS has the following disadvantages:

- Not useful when the leader HMS goes down in some environments
- Configuring a leader HMS in cloud environments is not easy
- All the tasks are expected to be performed by the leader HMS, which can overburden the HMS instance

Dynamic leader election

You can enable dynamic leader election for HMS to overcome the challenges listed above. This feature uses Hive locks to dynamically elect a leader. When a HMS instance owns a lock, it is elected as the leader and performs the housekeeping tasks. The HMS regularly sends heartbeats to prevent the lock from timing out. If the leader loses the lock, the in-progress housekeeping tasks are stopped. The other HMS instances should monitor the lock periodically and acquire the lock to claim leadership.

The leader election event is audited and the details of the leader are stored in a Hive table. There are listeners that keep track of the leader election and notify if a HMS instance wins or loses a leader election.

To enable dynamic leader election for HMS, configure the following properties in Cloudera Manager using the 'Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml':

Configuration Key	**Value**	**Description**
metastore.housekeeping.leader.election	Default: host	By default, HMS chooses a leader based on the configured metastore.housekeeping.leader.hostname. Set the value to 'lock' if you want HMS to use Hive locks to elect a leader.
metastore.housekeeping.leader.auditTable	Default: "" (empty string)	Optional. Specify a Hive table name if you want to capture HMS leader changes in the specified audit table. The table should be of Text format with JSON serialized. The table is created automatically if it does not already exist. In a production environment, it is recommended that you set this property to view the leader change history by running a Hive query against the table.
metastore.housekeeping.leader.newAuditFile	Default: False	Optional. Specifies whether to create a new audit file in response to the new election event when metastore.housekeeping.leader.auditTable is not empty. Set the value to True to create a new file, else set it to False.
metastore.housekeeping.leader.auditFiles.limit	Default: 10	Optional. Enables you to limit the number of small event files. This property is useful when metastore.housekeeping.leader.newAuditFile is set to True or if the underlying file system does not support the append operation. If the number of the event files exceeds the limit, the oldest file is deleted.
metastore.lock.numretries	Default: 100	Optional. The number of times you want to try to get all the locks in case of failure.
metastore.lock.sleep.between.retries	Default: 60 seconds	Optional. The maximum sleep time between various retries in case of failure.

Additionally, ensure that you enable the following properties to support the HMS dynamic leader election feature:

- hive.compactor.initiator.on=true

- hive.metastore.housekeeping.threads.on=true

  Look for the above properties in Cloudera Manager under Clusters Hive Metastore Configuration .
- hive.compactor.cleaner.on=true

  Set this property using the 'Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml' in Cloudera Manager.

Cloudera Runtime 7.3.1

Apache Hive Performance Tuning

Date published: 2019-08-21
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Query results cache **4**

Managing high partition workloads **4**

Best practices for performance tuning **5**

ORC file format **5**

Advanced ORC properties 7

Performance improvement using partitions **7**

Bucketed tables in Hive **8**

Query results cache

Hive filters and caches similar or identical queries in the query results cache. Caching repetitive queries can reduce the load substantially when hundreds or thousands of users of BI tools and web services query Hive.

Some operations support vast user populations, who connect to Hive using BI systems such as Tableau. In these situations, repetitious queries are inevitable. The query results cache, which is on by default, filters and stores common queries in a cache. When you issue the query again, Hive retrieves the query result from a cache instead of recomputing the result, which takes a load off the backend system.

Every query that runs in Hive 3 stores its result in a cache. Hive evicts invalid data from the cache if the input table changes. For example, if you preform aggregation and the base table changes, queries you run most frequently stay in cache, but stale queries are evicted. The query results cache works with managed tables only because Hive cannot track changes to an external table. If you join external and managed tables, Hive falls back to executing the full query. The query results cache works with ACID tables. If you update an ACID table, Hive reruns the query automatically.

Configuring the query results cache

You can enable and disable the query results cache from the command line. You might want to do so to debug a query. You disable the query results cache in HiveServer by setting the following parameter to false: SET hive.query.results.cache.enabled=false;

By default, Hive allocates 2GB for the query results cache. You can change this setting by configuring the following parameter in bytes: hive.query.results.cache.max.size

Managing high partition workloads

You learn how to identify an error related to high partition workloads that require configuration of your Hive Virtual Warehouse to run successfully. You might need to configure your Virtual Warehouse to prevent these errors when inserting data into workloads with a large number of partitions and columns.

About this task

To prevent an error when inserting data into a high partition workload, such as a table having 5,000 partitions and 100 columns, configure the Hive Virtual Warehouse as described in the steps below.

The error message you might see when inserting data into a high partition workload that is not configured properly looks something like this:

```
ERROR : FAILED: Execution Error, return code 40000 from org.apache.hadoop.hi
ve.ql.exec.MoveTask. MetaException(message:One or more instances could not b
e made persistent)
```

Procedure

1. Try changing the size of your Hive Virtual Warehouse.
   For example, if you experience the high partition error in a small Virtual Warehouse, try using a medium Virtual Warehouse and set min/max nodes to 40.
2. In the Cloudera Data Warehouse UI, in Overview, go to the Virtual Warehouses tab, click ⋮, and select Edit corresponding to your Virtual Warehouse.
3. Click Configurations HiveServer2 .

4. Tune HS2 parameters in the Virtual Warehouse.
   For example, search for and set the following properties, or if the property is not found, click Add Custom Configuration to add a custom configuration, and set it as follows:

```
set hive.optimize.sort.dynamic.partition.threshold=0;
set hive.thrift.client.max.message.size=2147483647;
set hive.metastore.client.skip.columns.for.partitions=true;
set hive.stats.autogather=false;
set hive.stats.column.autogather=false;
set hive.msck.repair.batch.size=[***TABLE SCHEMA SIZE***];
```

5. Go to the Database Catalogs tab, and select your Database Catalog.
6. Click Options ⋮ , and select Edit.
7. Tune Metastore parameters in the Database Catalog.
   For example, click Configurations Metastore , search for and set the following properties, or if the property is not found, click + to add a custom configuration, and set it as follows.

```
hive.metastore.direct.sql.batch.size=[***TABLE SCHEMA SIZE***]
hive.txn.timeout=3600
hive.metastore.try.direct.sql=true
hive.metastore.try.direct.sql.ddl=true
```

8. Click Apply Changes.

Best practices for performance tuning

Review certain performance tuning guidelines related to configuring the cluster, storing data, and writing queries so that you can protect your cluster and dependent services, automatically scale resources to handle queries, and so on.

Best practices

- Use Ranger security service to protect your cluster and dependent services.
- Store data using the ORC File format. Others, such as Parquet are supported, but not as fast for Hive queries.
- Ensure that queries are fully vectorized by examining explain plans.

Related Information

Custom Configuration (about Cloudera Manager Safety Valve)

ORC file format

You can conserve storage in a number of ways, but using the Optimized Row Columnar (ORC) file format for storing Apache Hive data is most effective. ORC is the default storage for Hive data.

The ORC file format for Hive data storage is recommended for the following reasons:

- Efficient compression: Stored as columns and compressed, which leads to smaller disk reads. The columnar format is also ideal for vectorization optimizations.
- Fast reads: ORC has a built-in index, min/max values, and other aggregates that cause entire stripes to be skipped during reads. In addition, predicate pushdown pushes filters into reads so that minimal rows are read. And Bloom filters further reduce the number of rows that are returned.

- Proven in large-scale deployments: Meta (aka Facebook) uses the ORC file format for a 300+ PB deployment.

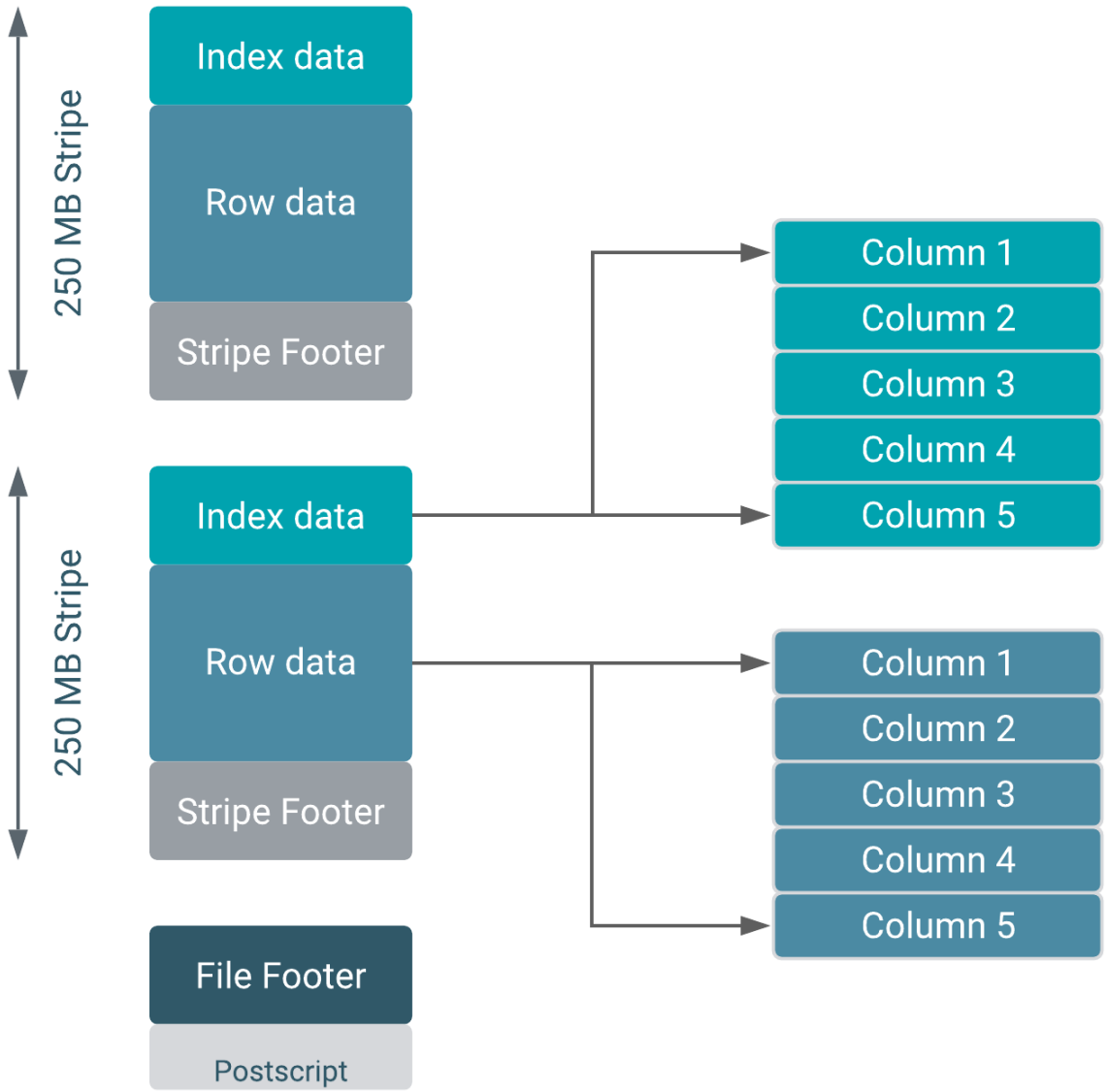

ORC provides the best Hive performance overall. In addition, to specifying the storage format, you can also specify a compression algorithm for the table, as shown in the following example:

```
CREATE TABLE addresses (
 name string,
 street string,
 city string,
 state string,
 zip int
 ) STORED AS orc TBLPROPERTIES ("orc.compress"="Zlib");
```

Setting the compression algorithm is usually not required because your Hive settings include a default algorithm. Using ORC advanced properties, you can create bloom filters for columns frequently used in point lookups.

Hive supports Parquet and other formats for insert-only ACID tables and external tables. You can also write your own SerDes (Serializers, Deserializers) interface to support custom file formats.

Advanced ORC properties

Usually, you do not need to modify Optimized Row Columnar (ORC) properties, but occasionally, Cloudera Support advises making such changes. Review the property keys, default values, and descriptions you can configure ORC to suit your needs.

Property keys and defaults

You use the Safety Valve feature in Cloudera Manager to change ORC properties.

Key	Default Setting	Description
orc.compress	ZLIB	Compression type (NONE, ZLIB, SNAPPY).
orc.compress.size	262,144	Number of bytes in each compression block.
orc.stripe.size	268,435,456	Number of bytes in each stripe.
orc.row.index.stride	10,000	Number of rows between index entries (>= 1,000).
orc.create.index	true	Sets whether to create row indexes.
orc.bloom.filter.columns	--	Comma-separated list of column names for which a Bloom filter must be created.
orc.bloom.filter.fpp	0.05	False positive probability for a Bloom filter. Must be greater than 0.0 and less than 1.0.

Related Information

Custom Configuration (about Cloudera Manager Safety Valve)

Performance improvement using partitions

You must be aware of partition pruning benefits, how to enable dynamic partitioning, and the configuration required for bulk-loading of data to ensure significant improvements in performance. You can use partitions to significantly improve performance.

You can design Hive table and materialized views partitions to map to physical directories on the file system/object store. For example, a table partitioned by date-time can organize data loaded into Hive each day. Large deployments can have tens of thousands of partitions. Partition pruning occurs indirectly when Hive discovers the partition key during query processing. For example, after joining with a dimension table, the partition key might come from the dimension table. A query filters columns by partition, limiting scanning that occurs to one or a few matching partitions. Partition pruning occurs directly when a partition key is present in the WHERE clause. Partitioned columns are virtual, not written into the main table because these columns are the same for the entire partition.

You do not need to specify dynamic partition columns. Hive generates a partition specification if you enable dynamic partitions.

Configuration for loading 1 to 9 partitions:

```
SET hive.exec.dynamic.partition.mode=nonstrict;
SET hive.exec.dynamic.partition=true;
```

For bulk-loading data into partitioned ORC tables, you use the following property, which optimizes the performance of data loading into 10 or more partitions.

Configuration for loading 10 or more partitions:

```
hive.optimize.sort.dynamic.partition=true
```

Bucketed tables in Hive

If you migrated data from earlier Apache Hive versions to Hive 3, you might need to handle bucketed tables that impact performance. Review how Cloudera simplifies handling buckets. You learn about best practices for handling dynamic capabilities.

You can divide tables or partitions into buckets, which are stored in the following ways:

- As files in the directory for the table.
- As directories of partitions if the table is partitioned.

Specifying buckets in Hive 3 tables is not necessary. In Cloudera, Hive 3 buckets data implicitly, and does not require a user key or user-provided bucket number as earlier versions (ACID V1) did. For example:

V1:

```
CREATE TABLE hello_acid (load_date date, key int, value int)
CLUSTERED BY(key) INTO 3 BUCKETS
STORED AS ORC TBLPROPERTIES ('transactional'='true');
```

V2:

```
CREATE TABLE hello_acid_v2 (load_date date, key int, value int);
```

Performance of ACID V2 tables is on a par with non-ACID tables using buckets. ACID V2 tables are compatible with native cloud storage.

A common challenge related to using buckets in tables migrated from earlier versions is maintaining query performance while the workload or data scales up or down. For example, you could have an environment that operates smoothly using 16 buckets to support 1000 users, but a spike in the number of users to 100,000 for a day or two creates problems if you do not promptly tune the buckets and partitions. Tuning the buckets is complicated by the fact that after you have constructed a table with buckets, the entire table containing the bucketed data must be reloaded to reduce, add, or eliminate buckets.

In Cloudera, you only need to deal with the buckets of the biggest table. If workload demands change rapidly, the buckets of the smaller tables dynamically change to complete table JOINs.

Bucket configurations

You can enable buckets as follows:

```
SET hive.tez.bucket.pruning=true
```

When you load data into tables that are both partitioned and bucketed, set the hive.optimize.sort.dynamic.partition property to optimize the process:

```
 SET hive.optimize.sort.dynamic.partition=true
```

If you have 20 buckets on user_id data, the following query returns only the data associated with user_id = 1:

```
SELECT * FROM tab WHERE user_id = 1;
```

To best leverage the dynamic capability of table buckets, adopt the following practices:

- Use a single key for the buckets of the largest table.
- Usually, you need to bucket the main table by the biggest dimension table. For example, the sales table might be bucketed by customer and not by merchandise item or store. However, in this scenario, the sales table is sorted by item and store.

- Normally, do not bucket and sort on the same column.
- A table that has more bucket files than the number of rows is an indication that you should reconsider how the table is bucketed.

Cloudera Runtime 7.3.1

Hue Overview

Date published: 2020-05-21
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Hue overview 4

About Hue Query Processor 6

About the Hue SQL AI Assistant 6

Hue overview

Hue is a web-based interactive query editor that enables you to interact with databases and data warehouses. Data architects, SQL developers, and data engineers use Hue to create data models, clean data to prepare it for analysis, and to build and test SQL scripts for applications.

Hue packs the combined abilities of Data Analytics Studio (DAS) such as query optimization, query debugging framework, and rich query editor experience of Hue, making Hue the next generation SQL assistant on Cloudera. You can search Hive query history, view query details, visual explain plan, and DAG information, compare two queries, and download debug bundles for troubleshooting from the Job Browser page.

Hue offers powerful execution, debugging, and self-service capabilities to the following key Big Data personas:

- Business Analysts
- Data Engineers
- Data Scientists
- Power SQL users
- Database Administrators
- SQL Developers

Business Analysts (BA) are tasked with exploring and cleaning the data to make it more consumable by other stakeholders, such as the data scientists. With Hue, they can import data from various sources and in multiple formats, explore the data using File Browser and Table Browser, query the data using the smart query editor, and create dashboards. They can save the queries, view old queries, schedule long-running queries, and share them with other stakeholders in the organization. They can also use Cloudera Data Visualization to get data insights, generate dashboards, and help make business decisions.

Data Engineers design data sets in the form of tables for wider consumption and for exploring data, as well as scheduling regular workloads. They can use Hue to test various Data Engineering (DE) pipeline steps and help develop DE pipelines.

Data scientists predominantly create models and algorithms to identify trends and patterns. They then analyze and interpret the data to discover solutions and predict opportunities. Hue provides quick access to structured data sets and a seamless interface to compose queries, search databases, tables, and columns, and execute query faster by leveraging Tez and LLAP. They can run ad hoc queries and start the analysis of data as pre-work for designing various machine learning models.

Power SQL users are advanced SQL experts tasked with analyzing and fine-tuning queries to improve query throughput and performance. They often strive to meet the TPC decision support (TPC-DS) benchmark. Hue enables them to run complex queries and provides intelligent recommendations to optimize the query performance. They can further fine-tune the query parameters by comparing two queries, viewing the explain plan, analyzing the Directed Acyclic Graph (DAG) details, and using the query configuration details. They can also create and analyze materialized views.

The Database Administrators (DBA) provide support to the data scientists and the power SQL users by helping them to debug long-running queries. They can download the query logs and diagnostic bundles, export and process log events to a database, or download it in JSON file format for further analysis. They monitor all the queries running in the system and terminate long-running queries to free up resources. Additionally, they may also provision databases, manage users, monitor and maintain the database and schemas for the Hue service, and provide solutions to scale up the capacity or migrate to other databases, such as PostgreSQL, Oracle, MySQL, and so on.

All Hue users can download logs and share them with their DBA or Cloudera Support for debugging and troubleshooting purposes.

SQL Developers can use Hue to create data sets to generate reports and dashboards that are often consumed by other Business Intelligence (BI) tools, such as Cloudera Data Visualization.

Hue can sparsely be used as a Search Dashboard tool, usually to prototype a custom search application for production environments.

For example, the following image shows a graphical representation of Impala SQL query results that you can generate with Hue:

Figure 1: Impala SQL query results generated with Hue

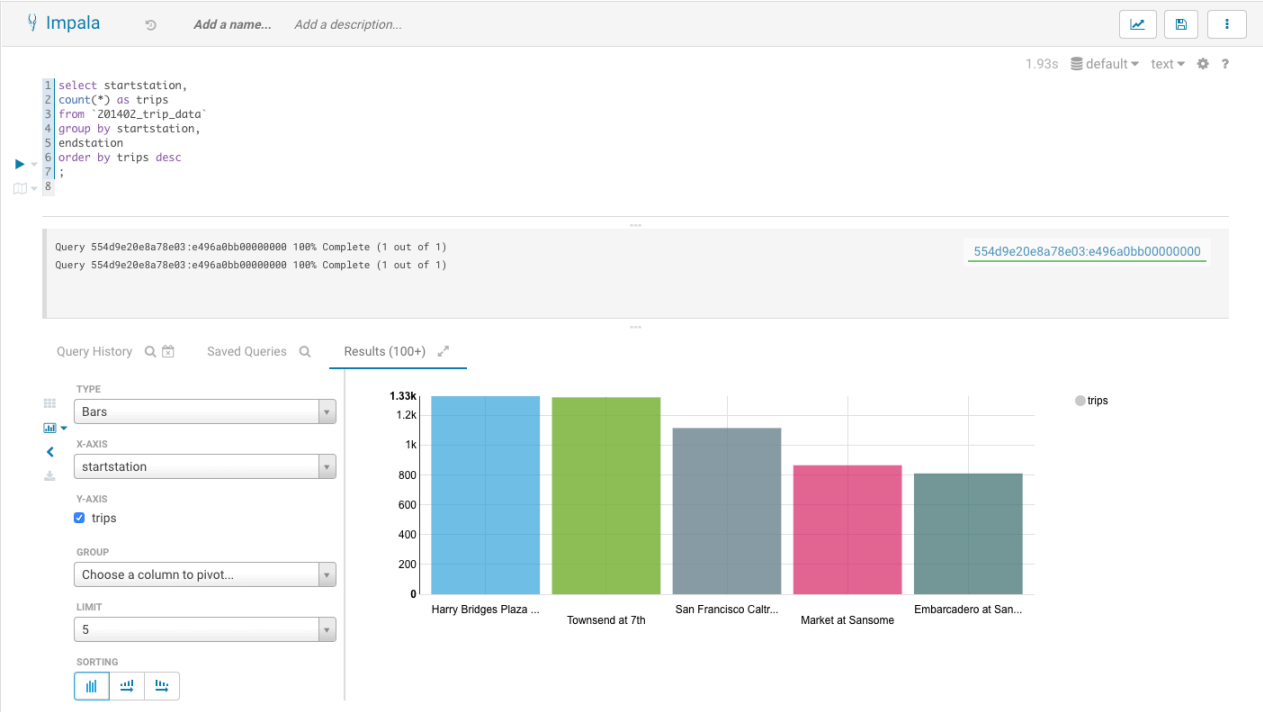

You can use Hue to:

- Explore, browse, and import your data through guided navigation in the left panel of the page.

  From the left panel, you can:

  - Browse your databases
  - Drill down to specific tables
  - View HDFS directories and cloud storage
  - Discover indexes and HBase or Kudu tables
  - Find documents

  Objects can be tagged for quick retrieval, project association, or to assign a more "human-readable" name, if desired.
- Query your data, create a custom dashboard, or schedule repetitive jobs in the central panel of the page.

  The central panel of the page provides a rich toolset, including:

  - Versatile editors that enable you to create a wide variety of scripts.
  - Dashboards that you can create "on-the-fly" by dragging and dropping elements into the central panel of the Hue interface. No programming is required. Then you can use your custom dashboard to explore your data.
  - Schedulers that you can create by dragging and dropping, just like the dashboards feature. This feature enables you to create custom workflows and to schedule them to run automatically on a regular basis. A monitoring interface shows the progress, logs, and makes it possible to stop or pause jobs.
- Get expert advice on how to complete a task using the assistance panel on the right.

  The assistant panel on the right provides expert advice and hints for whatever application is currently being used in the central panel. For example, in the above image, Impala SQL hints are provided to help construct queries in the central panel.

- (Hive-only) View query details such as query information, visual explain, query timeline, query configuration, Directed Acyclic Graph (DAG) information, DAG flow, DAG swimlane, DAG counters, and DAG configurations.
- (Impala-only) View query details such as query type, duration, default database, request pool, CPU time, rows produced, peak memory, HDFS bytes read, coordinator details, execution plan, execution summary, and counters and metrics.
- (Hive-only) Terminate Hive queries.
- (Hive-only) Download debug bundles.
- Compare two queries.
- View query history.

Hue also provides a simple SQL interface to create, access, and query HBase tables using Apache Phoenix in addition to HBase shell and database API.

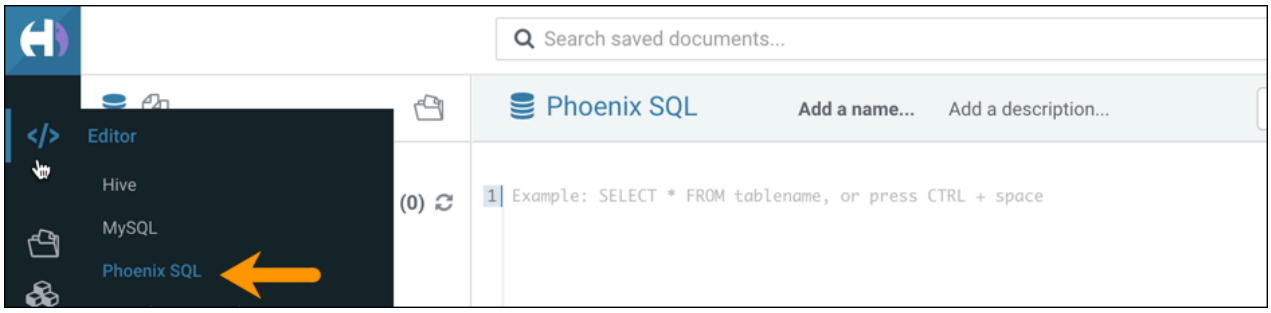

Related Information

Apache Phoenix and SQL

About Hue Query Processor

The Hue Query Processor service indexes Hive and Impala query history and provides APIs to access them. Enabling the processor allows you to view only Hive query history and query details on the Queries tab on the Hue Job Browser page.

Note: Hue does not display Impala query history and query details on the Queries tab on the **Job Browser** page.

Note: Query Processor is set up by default in clusters created using the Data Engineering template. Administrators have the privilege to add or remove services in Cloudera Data Hub clusters.

You can enable or disable Hue to display the **Queries** tab by selecting or deselecting the Query Processor Service option in the Hue Advanced Configuration Snippet in Cloudera Manager.

Related Information

Adding the Query Processor service to a cluster

Removing the Query Processor service from a cluster

Enabling the Queries tab in Hue

About the Hue SQL AI Assistant

Learn about the AI models and services that Hue uses to run the SQL AI Assistant and its limitations. Review what data is shared with the LLM models before you start using the SQL AI Assistant with Hue.

A SQL AI Assistant has been integrated into Hue with the capability to leverage the power of Large Language Models (LLMs) for various SQL tasks. It helps you to create, edit, optimize, fix, and succinctly summarize queries using natural language and makes SQL development faster, easier, and less error-prone. Both Hive and Impala dialects are supported.

AI models and services that Hue uses

The SQL AI Assistant supports various LLMs and hosting services. The models run on cloud infrastructure, and the AI Assistant can be configured to use them remotely. Cloudera has tested with GPT running in Open AI, Microsoft Azure, and Amazon Bedrock. The following service-model combinations are supported:

Service Provider	Model	Model Versions
OpenAI	OpenAI GPT	• gpt-3.5-turbo • gpt-3.5-turbo-16k Current GPT version is 3.5 turbo. You can configure GPT 4 for better results.
Microsoft Azure	OpenAI GPT	• gpt-3.5-turbo • gpt-3.5-turbo-16k
Amazon Bedrock	Anthropic Claude	• anthropic.claude-v1 • anthropic.claude-v2
Amazon Bedrock	Amazon Titan	• amazon.titan-text-express-v1

Note:

You must have access to the Hugging Face to download the required sentence transformer model, and ensure your system can connect to the internet, specifically, huggingface.co. Since these models aren't pre-bundled, they must be downloaded during setup. For more information, see Hugging Face

For better results, Cloudera recommends you to use the SQL AI assistant with the Azure OpenAI service. This ensures that the models run in your Virtual Private Cloud (VPC) network.

The SQL AI Assistant uses a Retrieval Augmented Generation (RAG)-based architecture for augmenting results. It uses the sentence-transformer library for semantic search, and Hue can be configured with any of the pre-trained models for better multi-lingual support. By default, "all-MiniLM-L6-v2" models are used.

Embedding Model	Language Support
all-MiniLM-L6-v2	English
distiluse-base-multilingual-cased-v1	Arabic, Chinese, Dutch, English, French, German, Italian, Korean, Polish, Portuguese, Russian, Spanish, and Turkish.

What data is shared with the LLM models

The following details are shared with the LLMs:

- Everything that a user inputs
- Dialect in use
- Table details such as table name, column names, column data types and related keys, partitions, and constraints that the logged-in user has access to.
- Three sample rows from the tables (as per the best practices specified in *Evaluating the Text-to-SQL Capabilities of Large Language Models*)

Limitations

Non-deterministic nature

LLMs are non-deterministic, which means you cannot guarantee the same output for the same input every time, and it can lead to different responses to similar queries.

Ambiguity

LLMs may struggle to handle ambiguous queries or contexts. SQL queries often rely on specific and unambiguous language, but LLMs can misinterpret or generate ambiguous SQL queries, leading to incorrect results.

Hallucinations

In the context of LLMs, hallucination refers to a phenomenon where these models generate text or responses that are incorrect, nonsensical, or fabricated. Occasionally you might see incorrect identifiers or literals in the response.

..

Using Apache Iceberg

Date published: 2022-03-15
Date modified: 2023-11-20

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Iceberg features **5**
Alter table feature 5
Create table feature 6
Create table as select feature 8
Create partitioned table as select feature 8
Create table … like feature 9
Delete data feature 9
Describe table metadata feature 11
Drop partition feature 12
Drop table feature 12
Expire snapshots feature 13
Insert table data feature 14
Load data inpath feature 15
Load or replace partition data feature 15
Materialized view feature 16
Materialized view rebuild feature 17
Merge feature 19
Migrate Hive table to Iceberg feature 20
Changing the table metadata location 20
Flexible partitioning 21
Partition evolution feature 21
Partition transform feature 22
Query metadata tables feature 23
Rollback table feature 26
Select Iceberg data feature 26
Schema evolution feature 26
Schema inference feature 28
Snapshot management 28
Time travel feature 29
Truncate table feature 29
Best practices for Iceberg in Cloudera **30**
Making row-level changes on V2 tables only 30
Performance tuning **31**
Caching manifest files 31
Configuring manifest caching in Cloudera Manager 32
Unsupported features and limitations **32**
Accessing Iceberg tables **34**
Opening Ranger in Cloudera Data Hub 34
Editing a storage handler policy to access Iceberg files on the file system 35
Creating a SQL policy to query an Iceberg table 38
Creating an Iceberg table **39**
Creating an Iceberg partitioned table **40**
Expiring snapshots **41**
Inserting data into a table **41**
Migrating a Hive table to Iceberg **42**
Selecting an Iceberg table **43**
Running time travel queries **43**
Updating an Iceberg partition **44**

Test driving Iceberg from Impala **45**
Hive demo data **46**
Test driving Iceberg from Hive **49**
Iceberg data types **51**
Iceberg table properties **52**

Apache Iceberg features

You can quickly build on your past experience with SQL to analyze Iceberg tables.

From Hive or Impala, you run SQL queries to create and query Iceberg tables. Impala queries are table-format agnostic. For example, Impala options are supported in queries of Iceberg tables. You can run nested, correlated, or analytic queries on all supported table types. Most Hive queries are also table-format agnostic.

This documentation does not attempt to show every possible query supported from Impala. Many examples of how to run queries on Iceberg tables from Impala are covered.

Supported ACID transaction properties

Iceberg supports atomic and isolated database transaction properties. Writers work in isolation, not affecting the live table, and perform a metadata swap only when the write is complete, making the changes in one atomic commit.

Iceberg uses snapshots to guarantee isolated reads and writes. You see a consistent version of table data without locking the table. Readers always see a consistent version of the data without the need to lock the table. Writers work in isolation, not affecting the live table, and perform a metadata swap only when the write is complete, making the changes in one atomic commit.

Iceberg partitioning

The Iceberg partitioning technique has performance advantages over conventional partitioning, such as Apache Hive partitioning. Iceberg hidden partitioning is easier to use. Iceberg supports in-place partition evolution; to change a partition, you do not rewrite the entire table to add a new partition column, and queries do not need to be rewritten for the updated table. Iceberg continuously gathers data statistics, which supports additional optimizations, such as partition pruning.

Iceberg uses multiple layers of metadata files to find and prune data. Hive and Impala keep track of data at the folder level and not at the file level, performing file list operations when working with data in a table. Performance problems occur during the execution of multiple list operations. Iceberg keeps track of a complete list of files within a table using a persistent tree structure. Changes to an Iceberg table use an atomic object/file level commit to update the path to a new snapshot file. The snapshot points to the individual data files through manifest files.

The manifest files track several data files across many partitions. These files store partition information and column metrics for each data file. A manifest list is an additional index for pruning entire manifests. File pruning increases efficiency.

Iceberg relieves Hive metastore (HMS) pressure by storing partition information in metadata files on the file system/object store instead of within the HMS. This architecture supports rapid scaling without performance hits.

Alter table feature

In Hive or Impala, you can use ALTER TABLE to set table properties. From Impala, you can use ALTER TABLE to rename a table, to change the table owner, or to change the role of the table owner. From Hive, you can alter the metadata location of the table if the new metadata does not belong to another table; otherwise, an exception occurs.

You can convert an Iceberg v1 table to v2 by setting a table property as follows:'format-version' = '2'.

Hive or Impala syntax

```
ALTER TABLE table_name SET TBLPROPERTIES table_properties;
```

- table_properties

  A list of properties and values using the following syntax:

  ```
  ('key' = 'value', 'key' = 'value', ... )
  ```

Impala syntax

```
ALTER TABLE table_name RENAME TO new_table_name;

ALTER TABLE table_name SET OWNER USER user_name;

ALTER TABLE table_name SET OWNER ROLE role_name;
```

Hive example

```
ALTER TABLE test_table SET TBLPROPERTIES('metadata_location'='s3a://bucketNa
me/ice_table/metadata/v1.metadata.json');
ALTER TABLE test_table2 SET TBLPROPERTIES('format-version' = '2');
```

Impala examples

```
ALTER TABLE t1 RENAME TO t2;

ALTER TABLE ice_table1 set OWNER USER john_doe;

ALTER TABLE ice_table2 set OWNER ROLE some_role;
ALTER TABLE ice_8 SET TBLPROPERTIES ('read.split.target-size'='268435456');

ALTER TABLE ice_table3 SET TBLPROPERTIES('format-version' = '2');
```

Create table feature

You use CREATE TABLE from Impala or CREATE EXTERNAL TABLE from Hive to create an external table in Iceberg. You learn the subtle differences in these features for creating Iceberg tables from Hive and Impala. You also learn about partitioning.

Hive and Impala handle external table creation a little differently, and that extends to creating tables in Iceberg. By default, Iceberg tables you create are v1. To create an Iceberg v2 table from Hive or Impala, you need to set a table property as follows:'format-version' = '2'.

Iceberg table creation from Hive

From Hive, CREATE EXTERNAL TABLE is recommended to create an Iceberg table in Cloudera.

When you use the EXTERNAL keyword to create the Iceberg table, by default only the schema is dropped when you drop the table. The actual data is not purged. Conversely, if you do not use EXTERNAL, by default the schema and actual data is purged. You can override the default behavior. For more information, see the Drop table feature.

From Hive, you can create a table that reuses existing metadata by setting the metadata_location table property to the object store path of the metadata. The operation skips generation of new metadata and re-registers the existing metadata. Use the following syntax:

```
CREATE EXTERNAL TABLE ice_fm_hive (i int) STORED BY ICEBERG TBLPROPERTIES ('
metadata_location'='<object store or file system path>')
```

See examples below.

Iceberg table creation from Impala

From Impala, CREATE TABLE is recommended to create an Iceberg table in Cloudera. Impala creates the Iceberg table metadata in the metastore and also initializes the actual Iceberg table data in the object store.

The difference between Hive and Impala with regard to creating an Iceberg table is related to Impala compatibility with Kudu, HBase, and other tables. For more information, see the Apache documentation, "Using Impala with Iceberg Tables".

Metadata storage of Iceberg tables

When you create an Iceberg table using CREATE EXTERNAL TABLE in Hive or using CREATE TABLE in Impala, HiveCatalog creates an HMS table and also stores some metadata about the table on your object store, such as S3. Creating an Iceberg table generates a metadata.json file, but not a snapshot. In the metadata.json, the snapshot-id of a new table is -1. Inserting, deleting, or updating table data generates a snapshot. The Iceberg metadata files and data files are stored in the table directory under the warehouse folder. Any optional partition data is converted into Iceberg partitions instead of creating partitions in the Hive Metastore, thereby removing the bottleneck.

To create an Iceberg table from Hive or from Impala, you associate the Iceberg storage handler with the table using one of the following clauses, respectively:

- Hive: STORED BY ICEBERG
- Impala: STORED AS ICEBERG or STORED BY ICEBERG

Supported file formats

You can write Iceberg tables in the following formats:

- From Hive: Parquet (default), Avro, ORC
- From Impala: Parquet

Impala supports writing Iceberg tables in only Parquet format. Impala does not support defining both file format and storage engine. For example, CREATE TABLE tbl ... STORED AS PARQUET STORED BY ICEBERG works from Hive, but not from Impala.

You can read Iceberg tables in the following formats:

- From Hive: Parquet, Avro, ORC
- From Impala: Parquet, Avro, ORC

Note: Reading Iceberg tables in Avro format from Impala is available as a technical preview. Cloudera recommends that you use this feature in test and development environments. It is not recommended for production deployments.

Hive syntax

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type, ... )]
  [PARTITIONED BY [SPEC]([col_name][, spec(value)][, spec(value)]...)]]
  [STORED AS file_format]
   STORED BY ICEBERG
  [TBLPROPERTIES ('key'='value', 'key'='value', ...)]
```

Impala syntax

```
CREATE TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type, ... )]
  [PARTITIONED BY [SPEC]([col_name][, spec(value)][, spec(value)]...)]]
   STORED {AS | BY} ICEBERG
  [TBLPROPERTIES (property_name=property_value, ...)]
```

Hive examples

```
CREATE EXTERNAL TABLE ice_1 (i INT, t TIMESTAMP, j BIGINT) STORED BY ICEBERG
;
CREATE EXTERNAL TABLE ice_2 (i INT, t TIMESTAMP) PARTITIONED BY (j BIGINT)
 STORED BY ICEBERG;
CREATE EXTERNAL TABLE ice_4 (i int) STORED AS ORC STORED BY ICEBERG;
CREATE EXTERNAL TABLE ice_5 (i int) STORED BY ICEBERG TBLPROPERTIES ('metad
ata_location'='s3a://bucketName/ice_table/metadata/v1.metadata.json')
CREATE EXTERNAL TABLE ice_6 (i int) STORED AS ORC STORED BY ICEBERG TBLPRO
PERTIES ('format-version' = '2');
```

Impala examples

```
CREATE TABLE ice_7 (i INT, t TIMESTAMP, j BIGINT) STORED BY ICEBERG; //creat
es only the schema
CREATE TABLE ice_8 (i INT, t TIMESTAMP) PARTITIONED BY (j BIGINT) STORED BY
 ICEBERG; //creates schema and initializes data
CREATE TABLE ice_v2 (i INT, t TIMESTAMP) PARTITIONED BY (j BIGINT) STORED BY
 ICEBERG TBLPROPERTIES ('format-version' = '2'); //creates a v2 table
```

Related Information

Drop table feature

Partition transform feature

Create table as select feature

You can create an Iceberg table based on an existing Hive or Impala table.

The create table as select (CTAS) query can optionally include a partitioning spec for the table being created.

Hive examples

```
CREATE EXTERNAL TABLE ctas STORED BY ICEBERG AS SELECT i, t, j FROM ice_1;
```

Impala examples

```
CREATE TABLE ctas STORED BY ICEBERG AS SELECT i, b FROM ice_11;
```

Create partitioned table as select feature

You can create a partitioned Iceberg table by selecting another table. You see an example of how to use PARTITIONED BY and TBLPROPERTIES to declare the partition spec and table properties for the new table.

You see an example of using a partition transform with the PARTITIONED BY SPEC clause.

The newly created table does not inherit the partition spec and table properties from the source table in SELECT. The Iceberg table and the corresponding Hive table is created at the beginning of the query execution. The data is inserted / committed when the query finishes. So for a transient period the table exists but contains no data.

Hive syntax

```
CREATE [EXTERNAL] TABLE prod.db.sample
    USING iceberg
```

```
    PARTITIONED BY (part)
    TBLPROPERTIES ('key'='value')
    AS SELECT ...
```

Hive examples

```
CREATE EXTERNAL TABLE ctas STORED BY ICEBERG AS SELECT i, t, j FROM ice_1;

CREATE EXTERNAL TABLE ctas_part PARTITIONED BY(z) STORED BY ICEBERG TBLPROPE
RTIES ('format-version'='2')
AS SELECT x, ts, z FROM t;
CREATE EXTERNAL TABLE ctas_part_spec PARTITIONED BY SPEC (month(d)) STORED
BY ICEBERG TBLPROPERTIES ('format-version'='2')
AS SELECT x, ts, d FROM source_t;
```

Impala examples

```
CREATE TABLE ctas STORED BY ICEBERG AS SELECT i, b FROM ice_11;

CREATE TABLE ctas_part PARTITIONED BY(b) STORED BY ICEBERG AS SELECT i, s, b
 FROM ice_11;
CREATE TABLE ctas_part_spec PARTITIONED BY SPEC (month(d)) STORED BY ICEBERG
 TBLPROPERTIES ('format-version'='2')
AS SELECT x, ts, d FROM source_t;
```

Create table … like feature

You learn by example how to create an empty table based on another table.

From Hive or Impala, you can create an Iceberg table schema based on another table. The table contains no data. The table properties of the original table are carried over to the new table definition. The following examples show how to use this feature:

Hive example

```
CREATE EXTERNAL TABLE target LIKE source STORED BY ICEBERG;
```

Impala example

```
CREATE TABLE target LIKE source STORED BY ICEBERG;
```

Delete data feature

Cloudera supports row-level deletes in Iceberg V2 tables.

From Hive and Impala, you can use the following formatting types defined by the Iceberg Spec:

- position deletes
- equality deletes

Note: The equality deletes feature is in technical preview and not recommended for use in production deployments. Cloudera recommends that you try this feature in test and development environments.

For equality deletes, you must be aware of the following considerations:

- If you are using Apache Flink or Apache NiFi to write equality deletes, then ensure that you provide a PRIMARY KEY for the table. This is required for engines to know which columns to write into the equality delete files.
- If the table is partitioned then the partition columns have to be part of the PRIMARY KEY
- For Apache Flink, the table should be in 'upsert-mode' to write equality deletes
- Partition evolution is not allowed for Iceberg tables that have PRIMARY KEYs

Position delete files contain the following information:

- file_path, which is a full URI
- pos, the file position of the row

Delete files are sorted by file_path and pos. The following table shows an example of delete files in a partitioned table:

Table 1:

Partition	Partition information	Delete file
YEAR(ts)=2022	ts_year=2022/data-abcd.parquet ts_year=2022/data-bcde.parquet	ts_year=2022/delete-wxyz.parquet
YEAR(ts)=2023	ts_year=2023/data-efgh.parquet	ts_year=2023/delete-hxkw.parquet
MONTH(ts)=2023-06	ts_month=2023-06/data-ijkl.parquet	ts_month=2023-06/delete-uzwd.parquet
MONTH(ts)=2023-07	ts_month=2023-07/data-mnop.parquet	ts_month=2023-07/delete-udqx.parquet

Inserting, deleting, or updating table data generates a snapshot.

You use a WHERE clause in your DELETE statement. For example:

```
delete from tbl_ice where a <= 2,1;
```

Hive and Impala evaluate rows from one table against a WHERE clause, and delete all the rows that match WHERE conditions. If you want delete all rows, use the Truncate feature. The WHERE expression is similar to the WHERE expression used in SELECT. The conditions in the WHERE clause can refer to any columns.

Concurrent operations that include DELETE do not introduce inconsistent table states. Iceberg runs validation checks to check for concurrent modifications, such as DELETE+INSERT. Only one will succeed. On the other hand, DELETE+DELETE, and INSERT+INSERT can both succeed, but in the case of a concurrent DELETE+UPDATE, UPDATE+UPDATE, DELETE+INSERT, UPDATE+INSERT from Hive, only the first operation will succeed.

From joined tables, you can delete all matching rows from one of the tables. You can join tables of any kind, but the table from which the rows are deleted must be an Iceberg table. The FROM keyword is required in this case, to separate the name of the table whose rows are being deleted from the table names of the join clauses.

Hive or Impala syntax

```
delete from tablename [where expression]
```

```
delete joined_tablename from [joined_tablename, joined_tablename2, ...] [ wh
ere expression ]
```

Hive or Impala examples

```
create external table tbl_ice(a int, b string, c int) stored by iceberg tblp
roperties ('format-version'='2');

insert into tbl_ice values (1, 'one', 50), (2, 'two', 51), (3, 'three', 5
2), (4, 'four', 53), (5, 'five', 54), (111, 'one', 55), (333, 'two', 56);
```

```
delete from tbl_ice where a <= 2,1;
```

The following example deletes 0, 1, or more rows of the table. If col1 is a primary key, 0 or 1 rows are deleted:

```
delete from ice_table where col1 = 100;
```

Describe table metadata feature

You can use certain Hive and Impala show and describe commands to get information about table metadata. You can also query metadata tables.

The following table lists SHOW and DESCRIBE commands supported by Hive and Impala.

Command Syntax	Description	SQL Engine Support
SHOW CREATE TABLE table_name	Reveals the schema that created the table.	Hive and Impala
SHOW FILES IN table_name	Lists the files related to the table.	Impala
SHOW PARTITIONS table_name	Returns the Iceberg partition spec, just the column information, not actual partitions or files.	Impala
DESCRIBE [EXTENDED] table_name	The optional EXTENDED shows all the metadata for the table in Thrift serialized form, which is useful for debugging.	Hive and Impala
DESCRIBE [FORMATTED] table_name	The optional FORMATTED shows the metadata in tabular format.	Hive
DESCRIBE HISTORY table_name [BET WEEN timestamp1 AND timestamp2]	Optionally limits the output history to a period of time.	Impala

Hive example

```
DESCRIBE t;
```

Hive output includes the following information:

col_name	data_type	comment
x	int	
y	int	
	NULL	NULL
# Partition Transform Information	NULL	NULL
# col_name	transform_type	NULL
y	IDENTITY	NULL

The output of DESCRIBE HISTORY includes the following columns about the snapshot. The first three are self-explanatory. The is_current_ancestor column value is TRUE if the snapshot is the ancestor of the table:

- creation_time
- snapshot_id
- parent_id
- is_current_ancestor

Impala examples

```
DESCRIBE HISTORY ice_t FROM '2022-01-04 10:00:00';
DESCRIBE HISTORY ice_t FROM now() - interval 5 days;
DESCRIBE HISTORY ice_t BETWEEN '2022-01-04 10:00:00' AND '2022-01-05 10:00:0
0';
```

Drop partition feature

You can easily remove a partition from an Iceberg partition using an alter table statement from Impala.

Removing a partition does not affect the table schema. The column is not removed from the schema.

Prerequisites

- The filter expression in the drop partition syntax below is a combination of, or at least one of, the following predicates:
  - Minimum of one binary predicate
  - IN predicate
  - IS NULL predicate
- The argument of the predicate in the filter expression must be a partition transform, such as ‘YEARS(col-name)’ or ‘column’.
- Any non-identity transform must be included in the select statement. For example, if you partition a column by days, the filter must be days.

Impala syntax

```
alter table table-name drop partition (<filter expression>)
```

The following operators are supported in the predicate: =, !=,<, >, <=, >=

Impala example

```
CREATE TABLE ice_table PARTITIONED BY SPEC (day(d)) STORED BY ICEBERG TBLPRO
PERTIES ('format-version'='2') AS SELECT x, ts, d FROM source_t;

INSERT INTO ice_table(d) VALUES ('2024-04-30');

ALTER TABLE ice_table DROP PARTITION (day(d) = '2024-04-30');
```

Drop table feature

The syntax you use to create the table determines the default behavior when you drop the Iceberg table from Hive or Impala.

If you use CREATE TABLE, the external.table.purge flag is set to true. When the table is dropped, the contents of the table directory (actual data) are removed. If you use CREATE EXTERNAL TABLE from Hive, the external.table.purge flag is set to false. Dropping a table purges the schema only. The actual data is not removed. You can explicitly set the external.table.purge property to true to drop the data as well as the schema.

To prevent data loss during migration of a table to Iceberg, do not drop or move the table during migration. Exception: If you set the table property 'external.table.purge'='FALSE', no data loss occurs if you drop the table.

Hive or Impala syntax

```
DROP TABLE [IF EXISTS] table_name
```

Hive or Impala example

```
ALTER TABLE t SET TBLPROPERTIES('external.table.purge'='true');
DROP TABLE t;
```

Related Information

Create table feature

Expire snapshots feature

You can expire snapshots that Iceberg generates when you create or modify a table. During the lifetime of a table the number of snapshots of the table accumulate. You learn how to remove snapshots you no longer need.

You should periodically expire snapshots to delete data files that are no longer needed, and to reduce the size of table metadata. Each write to an Iceberg table from Hive creates a new snapshot, or version, of a table. Snapshots accumulate until expired.

You can expire snapshots based on the following conditions:

- All snapshots older than a timestamp or timestamp expression
- A snapshot having a given ID
- Snapshots having IDs matching a given list of IDs
- Snapshots within the range of two timestamps

You can keep snapshots you are likely to need, for example recent snapshots, and expire old snapshots. For example, you can keep daily snapshots for the last 30 days, then weekly snapshots for the past year, then monthly snapshots for the last 10 years. You can remove specific snapshots to meet GDPR “right to be forgotten” requirements.

Hive or Impala syntax

```
ALTER TABLE <table Name> EXECUTE EXPIRE_SNAPSHOTS(<timestamp expression>)

ALTER TABLE <table Name> EXECUTE EXPIRE_SNAPSHOTS('<Snapshot Id>')

ALTER TABLE <table Name> EXECUTE EXPIRE_SNAPSHOTS('<Snapshot Id1>,<Snapshot
 Id2>... ')

ALTER TABLE <table Name> EXECUTE EXPIRE_SNAPSHOTS BETWEEN (<timestamp exp
ression>) AND (<timestamp expression>)
```

Hive or Impala example

The first example removes snapshots having a timestamp older than August 15, 2022 1:50 pm. The second example removes snapshots from 10 days ago and before.

```
ALTER TABLE ice_11 EXECUTE EXPIRE_SNAPSHOTS('2022-08-15 13:50:00');
ALTER TABLE ice_t EXECUTE EXPIRE_SNAPSHOTS(now() - interval 10 days);
```

Preventing snapshot expiration

You can prevent expiration of recent snapshots by configuring the history.expire.min-snapshots-to-keep table property. You can use the alter table feature to set a property. The history.expire.min-snapshots-to-keep property refers to a number of snapshots, not a time delta. For example, assume you always want to keep all snapshots of your

table for the last 24 hours. You configure history.expire.min-snapshots-to-keep as a safety mechanism to enforce this. If your table receives only one modification (insert / update / merge) per hour, then setting history.expire.min-snapshots-to-keep = 24 is sufficient to meet your requirement. However, if your table was consistently receiving updates every minute, then the last 24 hour period would entail 1440 snapshots, and the history.expire.min-snapshots-to-keep setting would need to be configured appropriately.

Table data and orphan maintenance

The contents of the table directory (actual data) might, or might not, be removed when you drop the table. An orphan data file can remain when you drop an Iceberg table, depending on the external.table.purge flag table property. An orphaned data file is one that has contents in the table directory, but no snapshot.

Expiring a snapshot does not remove old metadata files by default. You must clean up metadata files using writ e.metadata.delete-after-commit.enabled=true and write.metadata.previous-versions-max table properties. For more information, see "Iceberg table properties" below. Setting this property controls automatic metadata file removal after metadata operations, such as expiring snapshots or inserting data.

Related Information

Iceberg table properties

Insert table data feature

From Hive and Impala, you can insert data into Iceberg tables using the standard INSERT INTO a single table. INSERT statements work for V1 and V2 tables.

You can replace data in the table with the result of a query. To replace data, Hive and Impala dynamically overwrite partitions that have rows returned by the SELECT query. Partitions that do not have rows returned by the SELECT query, are not replaced. Using INSERT OVERWRITE on tables that use the BUCKET partition transform is not recommended. Results are unpredictable because dynamic overwrite behavior would be too random in this case.

From Hive, Cloudera also supports inserting into multiple tables as a technical preview; however, this operation is not atomic, so data consistency of Iceberg tables is equivalent to that of Hive external tables. Changes within a single table will remain atomic.

Inserting, deleting, or updating table data generates a snapshot. A new snapshot corresponds to a new manifest list. Manifest lists are named snap-*.avro.

Iceberg specification defines sort orders. At this point, Hive doesn’t support defining sort orders. But if there are sort orders defined by using other engines Hive can utilize them on write operations. For more information about sorting, see sort orders specification.

Hive or Impala syntax

```
INSERT INTO TABLE tablename VALUES values_row [, values_row ...]

INSERT INTO TABLE tablename1 select_statement1 FROM tablename2

INSERT OVERWRITE TABLE tablename1 select_statement1 FROM tablename2
```

Hive or Impala examples

```
CREATE TABLE ice_10 (i INT, s STRING, b BOOLEAN) STORED BY ICEBERG;
INSERT INTO ice_10 VALUES (1, 'asf', true);
CREATE TABLE ice_11 (i INT, s STRING, b BOOLEAN) STORED BY ICEBERG;
INSERT INTO ice_11 VALUES (2, 'apache', false);
INSERT INTO ice_11 SELECT * FROM ice_10;
SELECT * FROM ice_11;
INSERT OVERWRITE ice_11 SELECT * FROM ice_10;
```

Hive example

```
FROM customers
    INSERT INTO target1 SELECT customer_id, first_name;
    INSERT INTO target2 SELECT last_name, customer_id;
```

Load data inpath feature

From Impala, you can load Parquet or ORC data from a file in a directory on your file system or object store into an Iceberg table. You might need to set the mem_limit or pool configuration (max-query-mem-limit, min-query-mem-limit) to accommodate the load.

Impala syntax

```
LOAD DATA INPATH ‘<path to file>’ INTO table t;
```

Impala example

In this example, you create a table using the LIKE clause to point to a table stored as Parquet. This is required for Iceberg to infer the schema. You also load data stored as ORC.

```
CREATE TABLE test_iceberg LIKE my_parquet_table STORED AS ICEBERG;
SET MEM_LIMIT=1MB;
LOAD DATA INPATH 's3a://testbucket/files/my_data.parq' INTO TABLE test_ice
berg;

LOAD DATA INPATH 's3a://testbucket/files/my_data.orc' INTO TABLE test2_ic
eberg;
```

Load or replace partition data feature

There is no difference in the way you insert data into a partitioned or unpartitioned Iceberg table.

Working with partitions is easy because you write the query in the same way for the following operations:

- Insert into, or replace, an unpartitioned table
- Insert into, or replace, an identity partitioned table
- Insert into, or replace, a transform-partitioned table

Do not use INSERT OVERWRITE on tables that went through partition evolution. Truncate such tables first, and then INSERT the tables.

Impala example

```
CREATE TABLE ice_12 (i int, s string, t timestamp, t2 timestamp) STORED BY I
CEBERG;
```

Hive or Impala examples

```
INSERT INTO ice_12 VALUES (42, 'impala', now(), to_date(now()));
INSERT OVERWRITE ice_t VALUES (42, 'impala', now(), to_date(now()));
```

Materialized view feature

Using a materialized view can accelerate query execution. Creating a materialized view on top of Iceberg tables in Cloudera can further accelerate the performance. You can create a materialized view of an Iceberg V1 or V2 table based on an existing Hive or Iceberg table.

The materialized view is stored in Hive ACID or Iceberg format. Materialized view source tables either must be native ACID tables or must support table snapshots. Automatic rewriting of a materialized view occurs under the following conditions:

- The view definition contains the following operators only:
  - TableScan
  - Project
  - Filter
  - Join(inner)
  - Aggregate
- Source tables are native ACID or Iceberg v1 or v2
- The view is not based on time travel queries because those views do not have up-to-date data by definition.

Hive example

The following example creates a materialized view of an Iceberg table from Hive.

```
drop table if exists tbl_ice;
create external table tbl_ice(a int, b string, c int) stored by iceberg st
ored as orc tblproperties ('format-version'='2');

create materialized view mat1 as
select b, c from tbl_ice for system_version as of 5422037307753150798;
```

The following example creates a materialized view of two Iceberg tables. Joined tables must be in the same table format, either Iceberg or Hive ACID.

```
drop table if exists tbl_ice2;
create table tbl_ice2(a int, b string, c int) stored as orc TBLPROPERTIES ('
transactional'='true');
INSERT INTO tbl_ice2 VALUES (2, 'apache', 3);

drop table if exists tbl_ice;
create external table tbl_ice(a int, b string, c int) stored by iceberg sto
red as orc tblproperties ('format-version'='2');
INSERT INTO tbl_acid VALUES (4, 'iceberg', 5);
create materialized view mat1 as
select tbl_ice2.b, tbl_ice2.c from tbl_ice join tbl_ice2 on tbl_ice.a = tb
l_ice2.a;
```

The following example uses explain to examine a materialized view and then creates a materialized view of an Iceberg V1 table from Hive.

```
drop materialized view if exists mat1;
drop table if exists tbl_ice;

create table tbl_ice(a int, b string, c int) stored by iceberg stored as orc
 tblproperties ('format-version'='1');
insert into tbl_ice values (1, 'one', 50), (2, 'two', 51), (3, 'three', 52),
 (4, 'four', 53), (5, 'five', 54);
explain create materialized view mat1 stored by iceberg stored as orc tblpr
operties ('format-version'='1') as
```

```
select tbl_ice.b, tbl_ice.c from tbl_ice where tbl_ice.c > 52;

create materialized view mat1 stored by iceberg stored as orc tblproperties
 ('format-version'='1') as
select tbl_ice.b, tbl_ice.c from tbl_ice where tbl_ice.c > 52;
```

Related Information

Materialized view rebuild feature

Materialized view rebuild feature

Updates to materialized view contents when new data is added to the underlying table are critical; otherwise, queries can return stale data.

An update can occur under the following conditions:

- As a row-level incremental rebuild of the view after inserting data into a table

  Source tables can be Iceberg V2 or Hive full ACID.
- As a full rebuild of the view

A full rebuild can be expensive. An incremental rebuild updates only the affected parts of the materialized view, decreasing rebuild step execution time.

An incremental rebuild occurs automatically when you insert (append) data into a source table; otherwise, after you make some other type of change, for example a delete, you must manually start a full rebuild.

You use the ALTER command to manually start a full rebuild of the materialized view from Hive as follows:

Hive syntax

```
ALTER MATERIALIZED VIEW <name of view> REBUILD;
```

Full rebuild example

In this example, first you set required properties. Next, you create Iceberg tables, a V1 table and a V2 table, from Hive. You insert data into the tables and create a materialized view of the joined tables. You insert some new values into one of the source tables, rendering the materialized view stale. Finally, you rebuild the materialized view using explain cbo to show the rebuild plan The rebuild plan indicates a full rebuild will occur, which means the definition query will be executed.

```
drop table if exists tbl_ice;
drop table if exists tbl_ice_v2;
create external table tbl_ice(a int, b string, c int) stored by iceberg sto
red as orc tblproperties ('format-version'='1');
create external table tbl_ice_v2(d int, e string, f int) stored by iceberg
stored as orc tblproperties ('format-version'='2');
insert into tbl_ice_v2 values (1, 'one v2', 50), (4, 'four v2', 53), (5, 'f
ive v2', 54);

create materialized view mat1 as
select tbl_ice.b, tbl_ice.c, tbl_ice_v2.e from tbl_ice
join tbl_ice_v2 on tbl_ice.a=tbl_ice_v2.d where tbl_ice.c > 52;
group by tbl_ice.b tbl_ice.c;

-- view should be empty
select * from mat1;

-- view is up-to-date, use it
```

```
explain cbo
select tbl_ice.b, tbl_ice.c, tbl_ice_v2.e from tbl_ice join tbl_ice_v2 on
tbl_ice.a=tbl_ice_v2.d where tbl_ice.c > 52;

-- insert some new values to one of the source tables
insert into tbl_ice values (1, 'one', 50), (2, 'two', 51), (3, 'three', 52),
 (4, 'four', 53), (5, 'five', 54);

-- view is outdated, cannot be used
explain cbo
select tbl_ice.b, tbl_ice.c, tbl_ice_v2.e from tbl_ice join tbl_ice_v2 on tb
l_ice.a=tbl_ice_v2.d where tbl_ice.c > 52;

explain cbo
alter materialized view mat1 rebuild;

-- view should contain data
select * from mat1;

-- view is up-to-date again, use it
explain cbo
select tbl_ice.b, tbl_ice.c, tbl_ice_v2.e from tbl_ice join tbl_ice_v2 on tb
l_ice.a=tbl_ice_v2.d where tbl_ice.c > 52;
group by tbl_ice.b tbl_ice.c;
```

Automatic query rewrite example

In this example, you create an Iceberg table, insert some values, and create the materialized view. The view is partitioned using a partition specification and stored in the Iceberg ORC format. The v1 format version is specified in this example (the v2 format is also supported). You then look at a description of the view and see that the query rewrite option is enabled by default. An automatic incremental rebuild is possible when this option is enabled.

```
   drop table if exists tbl_ice;
   create external table tbl_ice(a int, b string, c int) stored by iceberg
 stored as orc tblproperties ('format-version'='1');

   -- insert some new values into one of the source tables

   insert into tbl_ice values (1, 'one', 50), (2, 'two', 51), (3, 'three', 5
2), (4, 'four', 53), (5, 'five', 54);

   explain
   create materialized view mat1 partitioned on spec (bucket(16, b), trunc
ate(3, c)) stored by Iceberg stored as orc tblproperties(‘format-version’=’1
’)as
   select tbl_ice.b, tbl)ice.c from tbl_ice where tbl_ice.c > 52;

   -- the output query plan query indicates a rewrite is enabled

   POSTHOOK: query: explain
   create materialized view mat1 …
   …
   STAGE PLANS
  …
-- In stage one, the materialized view is created by calling the Iceberg
API to create the table object.
rewrite enabled
…
```

You use the DESCRIBE command to see the output query plan, which shows details about the view, including if it can be used in automatic query rewrites.

```
-- check the materialized view details
    describe formatted mat1;
    …
    #col_name     data_type      comment
    b             string
    c             int

    #Partition Transform Information
    #col_name     transform_type
    b             bucket(16)
    c             TRUNCATE[3]


    #detailed table information
    …
    Table_type:    MATERIALIZED_VIEW
    Table Parameters:
    …
    current-snapshot-id   563939E424367334713
    …
    metadata_location     …
    …
    Table_type            Iceberg
    …
    #Materialized View Information
    Original Query:  …
    Expanded Query:  …
    Rewrite Enabled: Yes
```

With the query rewrite option enabled, you insert data into the source table, and incremental rebuild occurs automatically. You do not need to rebuild the view manually before running queries.

Related Information

Materialized view feature

Merge feature

You can perform actions on an Iceberg table based on the results of a join with a v2 Iceberg table.

Hive syntax

```
MERGE INTO <target table> AS T USING <source expression/table> AS S
ON <boolean expression1>
WHEN MATCHED [AND <boolean expression2>] THEN UPDATE SET <set clause list>
WHEN MATCHED [AND <boolean expression3>] THEN DELETE
WHEN NOT MATCHED [AND <boolean expression4>] THEN INSERT VALUES <value list>
```

Hive example

Use the MERGE INTO statement to update an Iceberg table based on a staging table:

```
MERGE INTO customer USING (SELECT * FROM new_customer_stage) sub ON sub.id =
 customer.id
    WHEN MATCHED THEN UPDATE SET name = sub.name, state = sub.new_state
    WHEN NOT MATCHED THEN INSERT VALUES (sub.id, sub.name, sub.state);
```

Create an Iceberg table and merge it with a non-Iceberg table.

```
create external table target_ice(a int, b string, c int) partitioned by spec
 (bucket(16, a), truncate(3, b)) stored by iceberg stored as orc tblproperti
es ('format-version'='2');
create table source(a int, b string, c int);
...

merge into target_ice as t using source src ON t.a = src.a
when matched and t.a > 100 THEN DELETE
when matched then update set b = 'Merged', c = t.c + 10
when not matched then insert values (src.a, src.b, src.c);
```

Migrate Hive table to Iceberg feature

Cloudera supports Hive table migration from Hive to Iceberg tables using ALTER TABLE to set the table properties.

The topic "Test driving Iceberg from Impala" shows how to create Iceberg tables from Impala tables.

Note: Do not drop or move the old table during a migration operation. Doing so will delete the data files of the old and new tables. Exception: If you set the table property 'external.table.purge'='FALSE', no data loss occurs when you drop the table.

In-place table migration process

In-place table migration saves time generating Iceberg tables. There is no need to regenerate data files. Only metadata, which points to source data files, is regenerated.

To convert a Hive table to an Iceberg V1 table, use the following syntax:

```
ALTER TABLE table_name CONVERT TO ICEBERG;
```

To convert a Hive table to an Iceberg V2 table, you must run two queries. Use the following syntax:

```
ALTER TABLE table_name CONVERT TO ICEBERG

ALTER TABLE table_name SET TBLPROPERTIES ('format-version' = '2'
    ...)
```

In-place table migration saves time generating Iceberg tables. There is no need to regenerate data files. Only metadata, which points to source data files, is regenerated.

Changing the table metadata location

From Hive and Impala, you can change the table metadata location, also known as the snapshot location.

Configuring security of Iceberg file metadata

In this release, you can prevent overrides of the Iceberg file metadata location by unauthorized users. If you do not prevent overrides, an unauthorized party who knows the underlying schema and file location outside the table location can rewrite the manifest files within one table location to point to the data files in another table location to read your data. You accept the default (true), or set a property in the Virtual Warehouse to true to prevent overrides:

- Hive property: hive.iceberg.allow.datafiles.in.table.location.only

  ```
  SET hive.iceberg.allow.datafiles.in.table.location.only = true;
  ```

- Impala property: iceberg_restrict_data_file_location

```
SET iceberg_restrict_data_file_location = true;
```

Prerequisites for changing the metadata location

- The new location must contain exactly the same metadata json file as the old location.
- Before changing the metadata location, you must migrate the table to Iceberg.

After migrating a table to Iceberg, you can change the metadata location using ALTER TABLE as shown below:

Hive and Impala example

```
ALTER TABLE test_table SET TBLPROPERTIES('metadata_location'='s3a://bucketN
ame/ice_table/metadata/v1.metadata.json');
```

Flexible partitioning

Iceberg partition evolution, which is a unique Iceberg feature, and the partition transform feature, greatly simplify partitioning tables and changing partitions.

Partitions based on transforms are stored in the Iceberg metadata layer, not in the directory structure. You can change the partitioning completely, or just refine existing partitioning, and write new data based on the new partition layout--no need to rewrite existing data files. For example, change a partition by month to a partition by day.

Use partition transforms, such as IDENTITY, TRUNCATE, BUCKET, YEAR, MONTH, DAY, HOUR. Iceberg solves scalability problems caused by having too many partitions. Partitioning can also include columns with a large number of distinct values. Partitioning is hidden in the Iceberg metadata layer, eliminating the need to explicitly write partition columns (YEAR, MONTH for example) or to add extra predicates to queries for partition pruning.

```
SELECT * FROM tbl WHERE ts = ‘2023-04-21 20:56:08’
AND YEAR = 2023 AND MONTH = 4 AND DAY = 21
```

Year, month, and day can be automatically extracted from ‘2023-04-21 20:56:08’ if the table is partitioned by DAY(ts)

Partition evolution feature

Partition evolution means you can change the partition layout of the table without rewriting existing data files. Old data files can remain partitioned by the old partition layout, while newly added data files are partitioned based on the new layout.

You can use the ALTER TABLE SET PARTITION SPEC statement to change the partition layout of an Iceberg table. A change to the partition spec results in a new metadata.json and a commit, but does not create a new snapshot.

Hive or Impala syntax

```
ALTER TABLE table_name SET PARTITION SPEC ([col_name][, spec(value)][, spec(
value)]...)]
```

- spec

  The specification for a transform listed in the next topic, "Partition transform feature".

Hive or Impala examples

```
ALTER TABLE t
```

```
SET PARTITION SPEC ( TRUNCATE(5, level), HOUR(event_time),
BUCKET(15, message), price);
ALTER TABLE ice_p
SET PARTITION SPEC (VOID(i), VOID(d), TRUNCATE(3, s), HOUR(t), i);
```

Related Information

Partition transform feature

Partition transform feature

From Hive or Impala, you can use one or more partition transforms to partition your data. Each transform is applied to a single column. Identity-transform means no transformation; the column values are used for partitioning. The other transforms apply a function to the column values and the data is partitioned by the transformed values.

Using CREATE TABLE ... PARTITIONED BY you create identity-partitioned Iceberg tables. Identity-partitioned Iceberg tables are similar to the Hive or Impala partitioned tables, which are stored in the same directory structure as the data files. Iceberg stores the partitioning columns of identity-partitioned Iceberg tables in a different directory structure from the data files if the tables are migrated to Iceberg from Hive external tables. Iceberg handles the tables and files regardless of the location.

Hive and Impala support Iceberg advanced partitioning through the PARTITION BY SPEC clause. Using this clause, you can define the Iceberg partition fields and partition transforms.

The following table lists the available transformations of partitions and corresponding transform spec.

Transformation	Spec	Supported by SQL Engine
Partition by year	years(time_stamp) | year(time_stamp)	Hive and Impala
Partition by month	months(time_stamp) | month(time_stamp)	Hive and Impala
Partition by a date value stored as int (dateint)	days(time_stamp) | date(time_stamp)	Hive
Partition by hours	hours(time_stamp)	Hive
Partition by a dateint in hours	date_hour(time_stamp)	Hive
Partition by hashed value mod N buckets	bucket(N, col)	Hive and Impala
Partition by value truncated to L, which is a number of characters	truncate(L, col)	Hive and Impala

Strings are truncated to length L. Integers and longs are truncated to bins. For example, truncate(10, i) yields partitions 0, 10, 20, 30 …

The idea behind transformation partition by hashed value mod N buckets is the same as hash bucketing for Hive tables. A hashing algorithm calculates the bucketed column value (modulus). For example, for 10 buckets, data is stored in column value % 10, ranging from 0-9 (0 to n-1) buckets.

You use the PARTITIONED BY SPEC clause to partition a table by an identity transform.

Hive syntax

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type][, time_stamp TIMESTAMP] )]
  [PARTITIONED BY SPEC([col_name][, spec(value)][, spec(value)]...)]
  [STORED AS file_format]
   STORED BY ICEBERG
  [TBLPROPERTIES (property_name=property_value, ...)]
```

Where spec(value)represents one or more of the following transforms:

- YEARS(col_name)
- MONTHS(col_name)
- DAYS(col_name)

- BUCKET(bucket_num,col_name)
- TRUNCATE(length, col_name)

Impala syntax

```
CREATE TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type, ... )]
  [PARTITIONED BY SPEC([col_name][, spec(value)][, spec(value)]...)]
   STORED (AS | BY) ICEBERG
  [TBLPROPERTIES (property_name=property_value, ...)]
```

Where spec(value) represents one or more of the following transforms:

- YEARS(col_name)
- MONTHS(col_name)
- DAYS(col_name)
- BUCKET(bucket_num,col_name)
- TRUNCATE(length, col_name)

Hive example

The following example creates a top level partition based on column i, a second level partition based on the hour part of the timestamp, and a third level partition based on the first 1000 characters in column j.

```
CREATE EXTERNAL TABLE ice_3 (i INT, t TIMESTAMP, j BIGINT) PARTITIONED BY SP
EC (i, HOUR(t), TRUNCATE(1000, j)) STORED BY ICEBERG;
```

Impala examples

```
CREATE TABLE ice_13 (i INT, t TIMESTAMP, j BIGINT) PARTITIONED BY SPEC (i, H
OUR(t), TRUNCATE(1000, j)) STORED BY ICEBERG;
```

The following examples show how to use the PARTITION BY SPEC clause in a CREATE TABLE query from Impala.The same transforms are available in a CREATE EXTERNAL TABLE query from Hive.

```
CREATE TABLE ice_t(id INT, name STRING, dept STRING)
PARTITIONED BY SPEC (bucket(19, id), dept)
STORED BY ICEBERG
TBLPROPERTIES ('format-version'='2');
```

```
CREATE TABLE ice_ctas
PARTITIONED BY SPEC (truncate(1000, id))
STORED BY ICEBERG
TBLPROPERTIES ('format-version'='2')
AS SELECT id, int_col, string_col FROM source_table;
```

Related Information

Creating an Iceberg partitioned table

Create table feature

Partition evolution feature

Query metadata tables feature

Apache Iceberg stores extensive metadata for its tables. From Hive and Impala, you can query the metadata tables as you would query a regular table. For example, you can use projections, joins, filters, and so on.

Iceberg metadata tables include information that is useful for efficient table maintenance (about snapshots, manifests, data, delete files, etc.) as well as statistics that help query engines plan and execute queries more efficiently (value count, min-max values, number of NULLs, etc.).

Note: Iceberg metadata tables are read-only. You cannot add, remove, or modify records in the tables. Also, you cannot drop or create new metadata tables.

For more information about the Apache Iceberg Iceberg metadata table types, see the *Apache Iceberg MetadataTableType enumeration*.

For more information about querying Iceberg metadata, see the *Apache Iceberg Spark documentation*.

The following sections describe how you can interact with and query Iceberg metadata tables:

List metadata tables

The SHOW METADATA TABLES statement lists all metadata tables belonging to an Iceberg table. You can also filter the tables according to a specific pattern.

Note: The SHOW METADATA TABLES statement is only available in Impala and is not supported in Hive.

Impala Syntax:

```
SHOW METADATA TABLES IN [database_name.]table_name [[LIKE] "pattern"];
```

Impala Example:

```
SHOW METADATA TABLES IN default.ice_table;

Output:
+----------------------+
| name                 |
+----------------------+
| all_data_files       |
| all_delete_files     |
| all_entries          |
| all_files            |
| all_manifests        |
| data_files           |
| delete_files         |
| entries              |
| files                |
| history              |
| manifests            |
| metadata_log_entries |
| partitions           |
| position_deletes     |
| refs                 |
| snapshots            |
+----------------------+
```

Query metadata tables

You can use the regular SELECT statement from Hive or Impala to query Iceberg metadata tables. To reference a metadata table, use the fully qualified path as shown in the syntax.

Hive or Impala Syntax:

```
SELECT … FROM database_name.table_name.metadata_table_name;
```

Hive or Impala Example:

```
SELECT * FROM default.ice_table.files;
```

You can select any subset of the columns or all of them using '*'. In comparison to regular tables, running a SELECT * from Impala on metadata tables always includes complex-typed columns in the result. The Impala query option, EXPAND_COMPLEX_TYPES only applies to regular tables. However, Hive always includes complex columns irresepctive of whether SELECT queries are run on regular tables or metadata tables.

For Impala queries that have a mix of regular tables and metadata tables, a SELECT * expression where the sources are metadata tables always includes complex types, whereas for SELECT * expressions where the sources are regular tables, complex types are included only if the EXPAND_COMPLEX_TYPES query option is set to 'true'.

In the case of Hive, columns with complex types are always included.

You can also filter the result set using a WHERE clause, use aggregate functions such as MAX or SUM, JOIN metadata tables with other metadata tables or regular tables, and so on.

Example:

```
SELECT
    s.operation,
    h.is_current_ancestor,
    s.summary
FROM default.ice_table.history h
JOIN default.ice_table.snapshots s
  ON h.snapshot_id = s.snapshot_id
WHERE s.operation = 'append'
ORDER BY made_current_at;
```

Limitations

- Impala does not support the DATE and BINARY data types. NULL is returned instead of their actual values.
- Impala does not support unnesting collections from metadata tables.

DESCRIBE metadata tables

Like regular tables, Iceberg metadata tables have schemas that can be explored using the DESCRIBE statement. The DESCRIBE statement displays metadata about a table, such as the column names and their data types.

To reference the metadata table, use the fully qualified path as shown in the syntax.

Note: DESCRIBE FORMATTED|EXTENDED is not available for metadata tables. In Impala, using this statement results in an error whereas Hive displays the same output as the regular DESCRIBE statement.

Hive or Impala Syntax:

```
DESCRIBE database_name.table_name.metadata_table_name;
```

Hive or Impala Example:

```
DESCRIBE default.ice_table.history;
Output:
+---------------------+-----------+---------+----------+
| name                | type      | comment | nullable |
+---------------------+-----------+---------+----------+
| made_current_at     | timestamp |         | true     |
| snapshot_id         | bigint    |         | true     |
| parent_id           | bigint    |         | true     |
| is_current_ancestor | boolean   |         | true     |
+---------------------+-----------+---------+----------+
```

Related Information

Apache Iceberg MetadataTableType
Apache Spark documentation

Rollback table feature

In the event of a problem with your table, you can reset a table to a good state as long as the snapshot of the good table is available. You can roll back the table data based on a snapshot id or a timestamp.

When you modify an Iceberg table, a new snapshot of the earlier version of the table is created. When you roll back a table to a snapshot, a new snapshot is created. The creation date of the new snapshot is based on the Timezone of your session. The snapshot id does not change.

Hive and Impala syntax

```
ALTER TABLE test_table EXECUTE rollback(snapshotID);
ALTER TABLE test_table EXECUTE rollback('timestamp');
```

Hive and Impala examples

The following example rolls back to an earlier table, creating a new snapshot having a new creation date timestamp, but keeping the same snapshot id 3088747670581784990.

```
ALTER TABLE ice_t EXECUTE ROLLBACK(3088747670581784990);
```

The following example rolls the table back to the latest snapshot having a creation timestamp earlier than '2022-08-08 00:00:00'.

```
ALTER TABLE ice_7 EXECUTE ROLLBACK('2022-08-08 00:00:00')
```

Select Iceberg data feature

You can read Iceberg tables from Impala as you would any table. Joins, aggregations, and analytical queries, for example, are supported.

Impala supports reading V2 tables with position deletes.

Hive or Impala example

```
SELECT * FROM ice_t;

SELECT count(*) FROM ice_t i LEFT OUTER JOIN other_t b
ON (i.id = other_t.fid)
WHERE i.col = 42;
```

Schema evolution feature

You learn that the Hive or Impala schema changes when the associated Iceberg table changes. You see examples of changing the schema.

Although you can change the schema of your table over time, you can still read old data files because Iceberg uniquely identifies schema elements. A schema change results in a new metadata.json and a commit, but does not create a new snapshot.

The Iceberg table schema is synchronized with the Hive/Impala table schema. A change to the schema of the Iceberg table by an outside entity, such as Spark, changes the corresponding Hive/Impala table. You can change the Iceberg table using ALTER TABLE to make the following changes:

From Hive:

- Add a column
- Replace a column
- Change a column type or its position in the table

From Impala:

- Add a column
- Rename a column
- Drop a column
- Change a column type

An unsafe change to a column type, which would require updating each row of the table for example, is not allowed. The following type changes are safe:

- int to long
- float to double
- decimal(P, S) to decimal(P', S) if precision is increased

You can drop a column by changing the old column to the new column.

Hive syntax

```
ALTER TABLE table_name ADD COLUMNS (col_name type[, …])
ALTER TABLE table_name CHANGE COLUMN col_old_name col_new_name type

ALTER TABLE table_name CHANGE COLUMN col_old_name col_new_name type [FIRST|
AFTER col_name] [existing_col_name
]
ALTER TABLE table_name REPLACE COLUMNS (col_name type)
```

Impala syntax

```
ALTER TABLE table_name ADD COLUMNS(col_name type[, …])

ALTER TABLE table_name CHANGE COLUMN col_old_name col_new_name type
ALTER TABLE table_name DROP COLUMN col_name
```

Hive examples

```
ALTER TABLE t ADD COLUMNS(message STRING, price DECIMAL(8,1));

ALTER TABLE t REPLACE COLUMNS (i int comment '...', a string, ...);
ALTER TABLE t CHANGE COLUMN col_x col_x DECIMAL (22, 3) AFTER col_y;
```

Impala examples

```
ALTER TABLE ice_12 ADD COLUMNS(message STRING, price DECIMAL(8,1));
ALTER TABLE ice_12 DROP COLUMN i;

ALTER TABLE ice_12 CHANGE COLUMN s str STRING;
```

Schema inference feature

From Hive or Impala, you can base a new Iceberg table on a schema in a Parquet file. You see a difference in the Hive and Impala syntax and examples.

From Hive, you must use FILE in the CREATE TABLE LIKE ... statement. From Impala, you must omit FILE in the CREATE TABLE LIKE … statement. The column definitions in the Iceberg table are inferred from the Parquet data file when you create a table like Parquet from Hive or Impala. Set the following table property for creating the table:

```
hive.parquet.infer.binary.as = <value>
```

Where <value> is binary (the default) or string.

This property determines the interpretation of the unannotated Parquet binary type. Some systems expect binary to be interpreted as string.

Hive syntax

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name LIKE FILE PARQU
ET 'object_storage_path_of_parquet_file'
[PARTITIONED BY [SPEC]([col_name][, spec(value)][, spec(value)]...)]]
[STORED AS file_format]
STORED BY ICEBERG
[TBLPROPERTIES (property_name=property_value, ...)]
```

Impala syntax

```
CREATE TABLE [IF NOT EXISTS] [db_name.]table_name LIKE PARQUET 'object_stora
ge_path_of_parquet_file'
[PARTITIONED BY [SPEC]([col_name][, spec(value)][, spec(value)]...)]]
STORED (AS | BY) ICEBERG
[TBLPROPERTIES (property_name=property_value, ...)]
```

Hive example

```
CREATE TABLE ctlf_table LIKE FILE PARQUET 's3a://testbucket/files/schema.par
q'
STORED BY ICEBERG;
```

Impala example

```
CREATE TABLE ctlf_table LIKE PARQUET 's3a://testbucket/files/schema.parq'
STORED BY ICEBERG;
```

Snapshot management

In addition to time travel queries, expiring a snapshot, and using a snapshot to rollback to a version of a table, you can also set any snapshot to be the current snapshot from Hive.

Hive syntax

```
ALTER TABLE TBL_ICEBERG_PART EXECUTE SET_CURRENT_SNAPSHOT (<snapshot ID>)
```

Hive example

```
ALTER TABLE TBL_ICEBERG_PART EXECUTE SET_CURRENT_SNAPSHOT (75212489901265493
11)
```

Time travel feature

From Hive or Impala, you can run point in time queries for auditing and regulatory workflows on Iceberg tables. Time travel queries can be time-based or based on a snapshot ID.

Iceberg generates a snapshot when you create, or modify, a table. A snapshot stores the state of a table. You can specify which snapshot you want to read, and then view the data at that timestamp. In Hive, you can use projections, joins, and filters in time travel queries. You can add expressions to the timestamps, as shown in the examples. You can expire snapshots.

Snapshot storage is incremental and dependent on the frequency and scale of updates. By default, Hive and Impala use the latest snapshot. You can query an earlier snapshot of Iceberg tables to get historical information. Hive and Impala use the latest schema to query an earlier table snapshot even if it has a different schema.

Hive or Impala syntax

```
SELECT * FROM table_name FOR SYSTEM_TIME AS OF 'time_stamp' [expression]

SELECT * FROM table_name FOR SYSTEM_VERSION AS OF snapshot_id [expression]
```

- time_stamp

  The state of the Iceberg table at the time specified by the UTC timestamp.
- snapshot_id

  The ID of the Iceberg table snapshot from the history output.

Hive or Impala examples

```
SELECT * FROM t FOR SYSTEM_TIME AS OF '2021-08-09 10:35:57' LIMIT 100;

SELECT * FROM t FOR SYSTEM_VERSION AS OF 3088747670581784990 limit 100;
SELECT * from ice_11 FOR SYSTEM_TIME AS OF now() - interval 30 minutes;
```

Truncate table feature

Truncating an Iceberg table removes all rows from the table. A new snapshot is created. Truncation works for partitioned and unpartitioned tables.

Although the table data and the table and column stats are cleared, the old snapshots and their data files continue to exist to support time travel in the future.

Hive syntax

```
TRUNCATE table_name
```

Impala syntax

```
TRUNCATE [TABLE] table_name
```

Hive or Impala example

```
TRUNCATE t;
```

Best practices for Iceberg in Cloudera

Based on large scale TPC-DS benchmark testing, performance testing and real-world experiences, Cloudera recommends several best practices when using Iceberg.

Follow these key best practices listed below when using Iceberg:

- Use Iceberg as intended for analytics.

  The table format is designed to manage a large, slow-changing collection of files. For more information, see the Iceberg spec.
- Increase parallelism to handle large manifest list files in Spark.

  By default, the number of processors determines the preset value of the iceberg.worker.num-threads system property. Try increasing parallelism by setting the iceberg.worker.num-threads system property to a higher value to speed up query compilation.
- Reduce read amplification

  Monitor the growth of positional delta files, and perform timely compactions.
- Speed up drop table performance, preventing deletion of data files by using the following table properties:

  ```
  Set external.table.purge=false and gc.enabled=false
  ```
- Tune the following table properties to improve concurrency on writes and reduce commit failures: commit.retry .num-retries (default is 4), commit.retry.min-wait-ms (default is 100)
- Maintain a relatively small number of data files under the iceberg table/partition directory for efficient reads. To alleviate poor performance caused by too many small files, run the following queries:

  ```
  TRUNCATE TABLE target;
                INSERT OVERWRITE TABLE target select * from target FOR SYST
  EM_VERSION AS OF <preTruncateSnapshotId>;
  ```
- To minimize the number of delete files and file handles and improve performance, ensure that the Spark write.distribution.mode table property value is “hash” (the default setting for Spark Iceberg 1.2.0 onwards).

Making row-level changes on V2 tables only

Learn the types of workloads best suited for Iceberg. Under certain conditions, using V2 tables versus V1 tables might improve query response.

Iceberg atomic DELETE and UPDATE operations resemble traditional RDBMS systems, but are not suitable for OLTP workloads. Iceberg is not designed to handle high frequency transactions. To handle very large datasets and frequent updates, use Apache Kudu.

Use Iceberg for managing large, infrequently changing datasets. You can update and delete Iceberg V2 tables at the row-level and not incur the overhead of rewriting the data files of V1 tables. Iceberg stores information about the deleted records in position delete files. These files store the file paths and positions of the deleted records, eliminating the need to rewrite the files. Iceberg performs a DELETE plus an INSERT operation in a single transaction. This technique speeds up queries. Query engines scan the data files and delete files associated with a snapshot and merge them, removing the deleted rows. For example, to remove all data belonging to a single customer:

DELETE FROM ice_tbl WHERE user_id = 1234;

To update a column value in a specific record:

UPDATE ice_tbl SET col_v = col_v + 1 WHERE id = 4321;

You can convert an Iceberg v1 table to v2 by setting a table property as follows:'format-version' = '2'.

Performance tuning

Impala uses its own C++ implementation to deal with Iceberg tables. This implementation provides significant performance advantages over other engines.

To tune performance, try the following actions:

- Increase parallelism to handle large manifest list files in Spark.

  By default, the number of processors determines the preset value of the iceberg.worker.num-threads system property. Try increasing parallelism by setting the iceberg.worker.num-threads system property to a higher value to speed up query compilation.
- Speed up drop table performance, preventing deletion of data files by using the following table properties:

  ```
  Set external.table.purge=false and gc.enabled=false
  ```
- Tune the following table properties to improve concurrency on writes and reduce commit failures: commit.retry .num-retries (default is 4), commit.retry.min-wait-ms (default is 100)
- Read Iceberg V2 tables from Hive using vectorization when heavy table scanning occurs as in SELECT COUNT(*) FROM TBL_ICEBERG_PART.
  - set hive.llap.io.memory.mode=cache;
  - set hive.llap.io.enabled=true;
  - set hive.vectorized.execution.enabled=true;
- Use Iceberg from Impala for querying Iceberg tables when latency is a concern.

  The massively parallel SQL query engine, backend executors written in C++, and frontend (analyzer, planner) written in Java delivers query results fast.
- Cache manifest files as described in the next topic.

Caching manifest files

Apache Iceberg provides a mechanism to cache the contents of Iceberg manifest files in memory. The manifest caching feature helps to reduce repeated reads of small Iceberg manifest files from remote storage by Impala Coordinators and Catalogd.

Impala caches table metadata in CatalogD and the local Coordinator’s catalog, making table metadata analysis fast if the targeted table metadata and files were previously accessed. Impala might analyze the same table multiple times across concurrent query planning and also within single query planning, so caching is very important.

Having a frontend written in Java, Impala can directly analyze many aspects of the Iceberg table metadata through the Java Library provided by Apache Iceberg. Metadata analysis such as listing the table data file, selecting the table snapshot, partition filtering, and predicate filtering is delegated through Iceberg Java API.

To use the Iceberg Java API while still maintaining fast query planning, Iceberg implements caching strategies in the Iceberg Java Library similar to those used by Apache Impala. The Iceberg manifest caching feature constitutes these caching strategies. For more information about manifest caching, see the Iceberg Manifest Caching Blog.

You can configure manifest caching in Cloudera Manager.

Configuring manifest caching in Cloudera Manager

You can enable or disable manifest caching for Impala Coordinators and Catalogd by setting properties in Cloudera Manager.

About this task

By default, only 8 catalogs can have their manifest cache active in memory.

To connect to more than 8 HDFS clusters, in Cloudera Manager, configure iceberg.io.manifest.cache.fileio-max in catalogd_java_opts and the coordinator impalad_embedded_java_opts.

In the following task, you enable Iceberg manifest caching for the Impala Coordinator and Catalog Server.

Procedure

1. Navigate to Cloudera Manager.
2. Search for Iceberg.

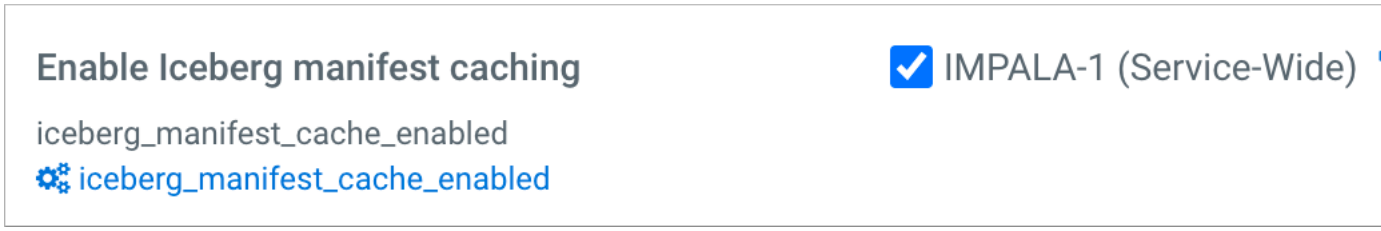

By default, manifest caching is enabled, but if you have turned it off, check Impala-1 (Service-Wide) to re-enable.

Unsupported features and limitations

Cloudera does not support all features in Apache Iceberg. The list of unsupported features for Cloudera differs from release to release. Also, Apache Iceberg in Cloudera has some limitations you need to understand.

Unsupported features

The following table presents feature limitations or unsupported features:

means not yet tested

N/A means will never be tested, not a GA candidate

Iceberg Feature	Hive	Impala	Spark
Branching/Tagging	#	#	#
Read equality deletes for Flink upserts	#	#	#
Read equality deletes for NiFi	#	#	#
Write equality deletes	N/A	N/A	N/A
Read outside files	N/A	N/A	N/A
Bucketing	#	#	#

The table above shows that the following features are not supported in this release of Cloudera:

- Tagging and branching

  A technical preview is supported from Hive (not Impala or Spark) in Cloudera Data Warehouse Public Cloud.
- Reading files outside the table directory

  Reading outside files is not supported due to the security risk. An unauthorized party who knows the underlying schema and file location outside the table location can rewrite the manifest files within one table location to point to the data files in another table location to read your data.
- Buckets defined from Hive do not create like buckets in Iceberg.

  For more information, see "Bucketing workaround" below.
- Using Iceberg tables as Spark Structured Streaming sources or sinks
- PyIceberg
- Migration of Delta Lake tables to Iceberg

Limitations

The following features have limitations or are not supported in this release:

- Multiple insert overwrite queries that read data from a source table.
- When the underlying table is changed, you need to rebuild the materialized view manually, or use the Hive query scheduling to rebuild the materialized view.
- You must be aware of the following considerations when using equality deletes:
  - Equality updates and deletes are not supported.
  - If you are using Apache Flink or Apache NiFi to write equality deletes, then ensure that you provide a PRIMARY KEY for the table. This is required for engines to know which columns to write into the equality delete files.
  - If the table is partitioned then the partition columns have to be part of the PRIMARY KEY
  - For Apache Flink, the table should be in 'upsert-mode' to write equality deletes
  - Partition evolution is not allowed for Iceberg tables that have PRIMARY KEYs
  - An equality delete file in the table is the likely cause of a problem with updates or deletes in the following situations:
    - In Change Data Capture (CDC) applications
    - In upserts from Apache Flink
    - From a third-party engine
- You must be aware of the following:
  - An Iceberg table that points to another Iceberg table in the HiveCatalog is not supported.

    For example:

    ```
    CREATE EXTERNAL TABLE ice_t
    STORED BY ICEBERG
    TBLPROPERTIES ('iceberg.table_identifier'='db.tb');
    ```
  - See also Iceberg data types.

Bucketing workaround

A query from Hive to define buckets/folders in Iceberg do not create the same number of buckets/folders as the same query creates in Hive. In Hive bucketing by multiple columns using the following clause creates 64 buckets maximum inside each partition.

```
| CLUSTERED BY (                                          |
|   id,                                                   |
|   partition_id)                                         |
| INTO 64 BUCKETS
```

Defining bucketing from Hive on multiple columns of an Iceberg table using this query creates 64*64 buckets/folders; consequently, bucketing by group does not occur as expected. The operation will create many small files at scale, a drag on performance.

Add multiple bucket transforms (partitions) to more than one column in the current version of Iceberg as follows:

```
bucket(p, col1, col2) =[ bucket(m, col1) , bucket(n, col2) ] where p = m * n
```

Accessing Iceberg tables

Cloudera uses Apache Ranger to provide centralized security administration and management. The Ranger Admin UI is the central interface for security administration. You can use Ranger to create two policies that allow users to query Iceberg tables.

How you open the Ranger Admin UI differs from one Cloudera service to another. In Cloudera Management Console, you can select your environment, and then click Environment Details Quick Links Ranger .

You log into the Ranger Admin UI, and the Ranger Service Manager appears.

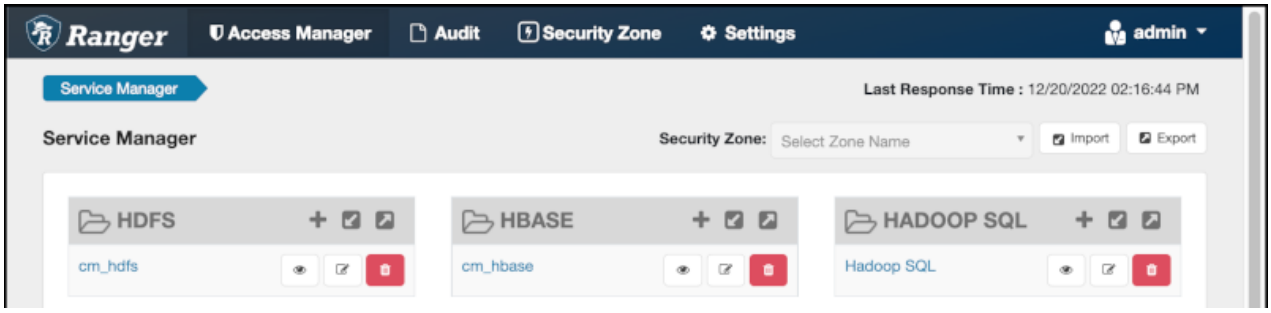

Policies for accessing tables on S3

The default policies that appear differ from service to service. You need to set up two Hadoop SQL policies to query Iceberg tables:

- One to authorize users to access the Iceberg files

  Follow steps in "Editing a policy to access Iceberg files" below.
- One to authorize users to query Iceberg tables

  Follow steps in "Creating a policy to query an Iceberg table" below.

Prerequisites

- Obtain the RangerAdmin role.
- Get the user name and password your Administrator set up for logging into the Ranger Admin.

  The default credentials for logging into the Ranger Admin Web UI are admin/admin123.

Opening Ranger in Cloudera Data Hub

You need to navigate to Ranger Admin UI to create a policy for users to access Iceberg tables. How you navigate to the Ranger Admin UI differs from one Cloudera service to another, and typically there is more than one path. You learn one way to navigate to the Ranger Admin UI from Cloudera Data Hub.

Procedure

1. Log into Cloudera, and in Cloudera Management Console, click Data Hub Clusters.

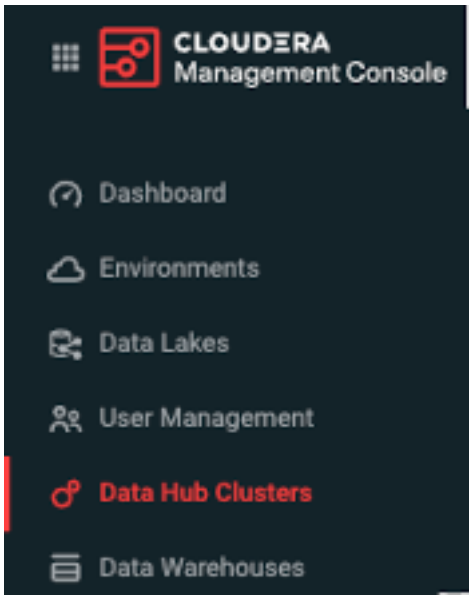

2. In **Data Hubs**, select the name of your Data Hub from the list.
3. In Environment Details, click the link to your Data Lake.
   For example, click dlscale8-bc8bqz.

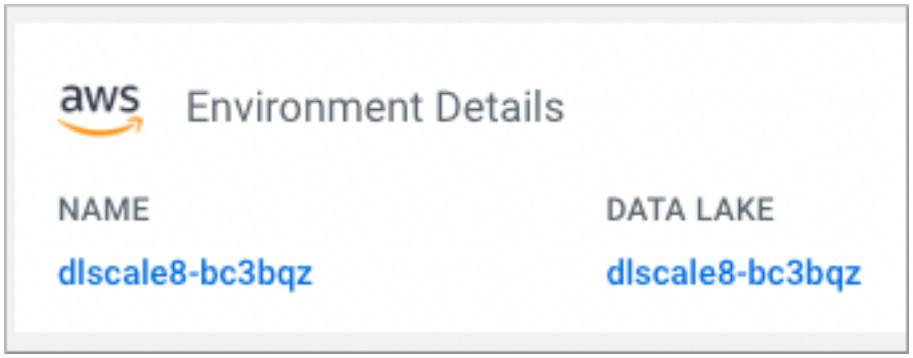

   In Data Lake Details, in Quick Links, Ranger appears:

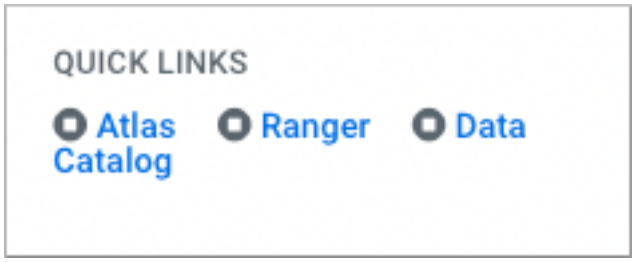

Editing a storage handler policy to access Iceberg files on the file system

You learn how to edit the existing default Hadoop SQL Storage Handler policy to access files. This policy is one of the two Ranger policies required to use Iceberg.

About this task

The Hadoop SQL Storage Handler policy allows references to Iceberg table storage location, which is required for creating or altering a table. You use a storage handler when you create a file stored as Iceberg on the file system or object store.

In this task, you specify Iceberg as the storage-type and allow the broadest access by setting the URL to *.

The Hadoop SQL Storage Handler policy supports only the RW Storage permission. A user having the required RW Storage permission on a resource, such as Iceberg, that you specify in the storage-type properties, is allowed only to reference the table location (for create/alter operations) in Iceberg. The RW Storage permission does not provide access to any table data. You need to create the Hadoop SQL policy described in the next topic in addition to this Hadoop SQL Storage Handler policy to access data in tables.

For more information about these policy settings, see Ranger Storage Handler documentation.

Procedure

1. Log into Ranger Admin Web UI.
   The Ranger Service Manager appears:

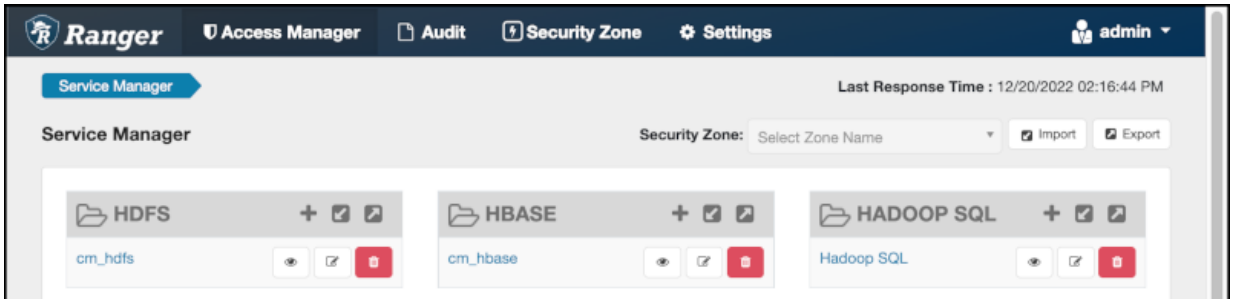

2. In Policy Name, enable the all - storage-type, storage-url policy.

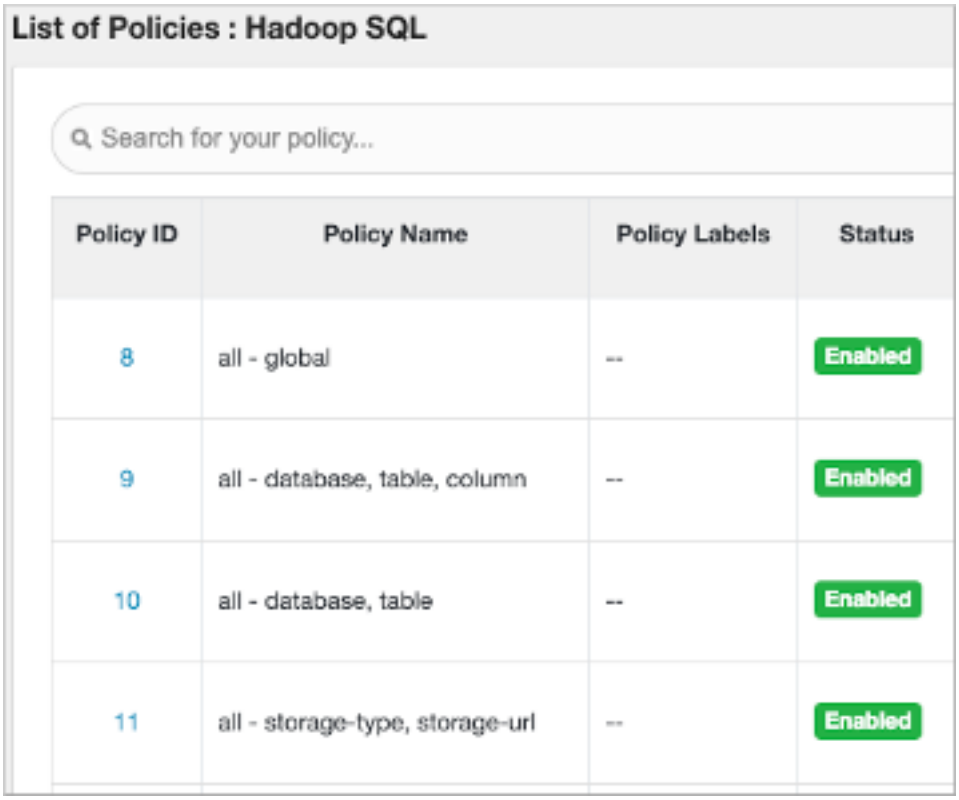

3. In **Service Manager**, in Hadoop SQL, select Edit and edit the all storage-type, storage-url policy.

4. Below Policy Label, select storage-type, and enter iceberg..
5. In Storage URL, enter the value *, enable Include.

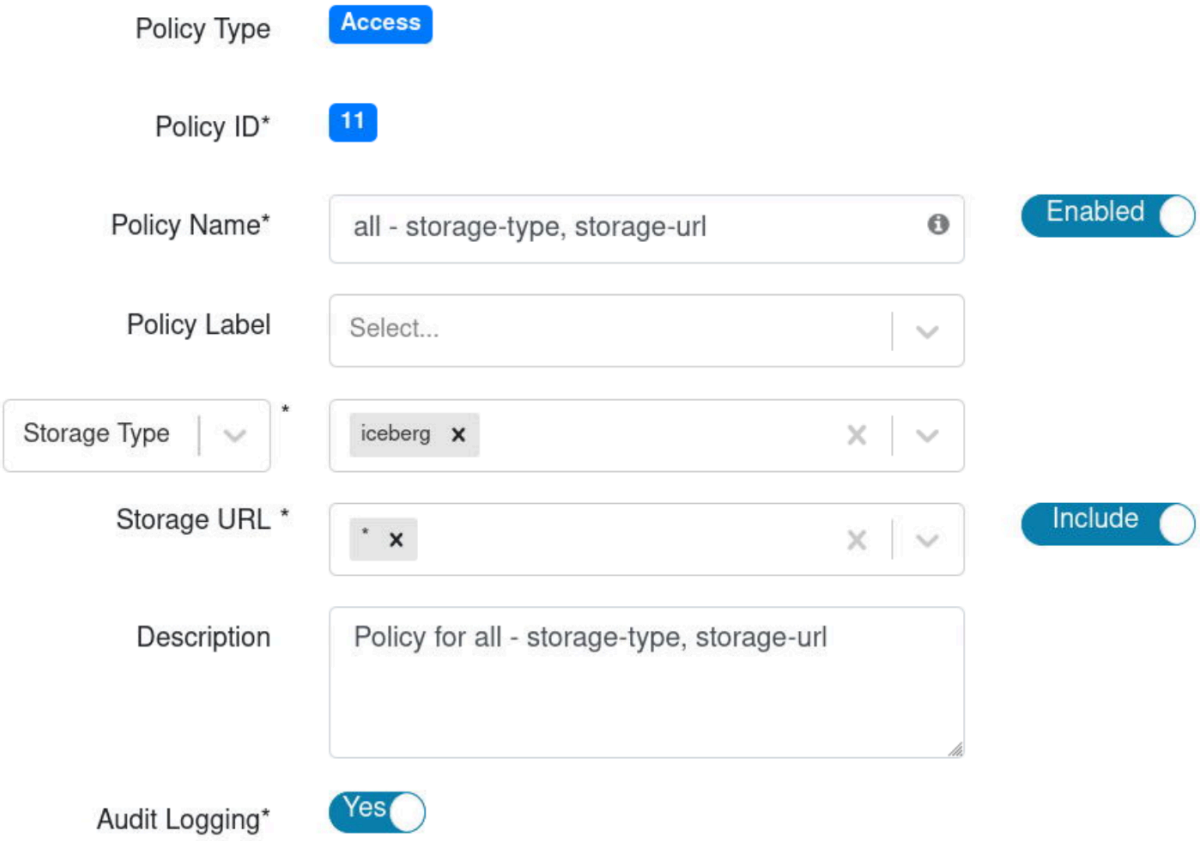

For more information about these policy settings, see Ranger storage handler documentation.

6. In Allow Conditions, specify roles, users, or groups to whom you want to grant RW storage permissions.

   You can specify PUBLIC to grant access to Iceberg tables permissions to all users. Alternatively, you can grant access to one user. For example, add the systest user to the list of users who can access Iceberg:

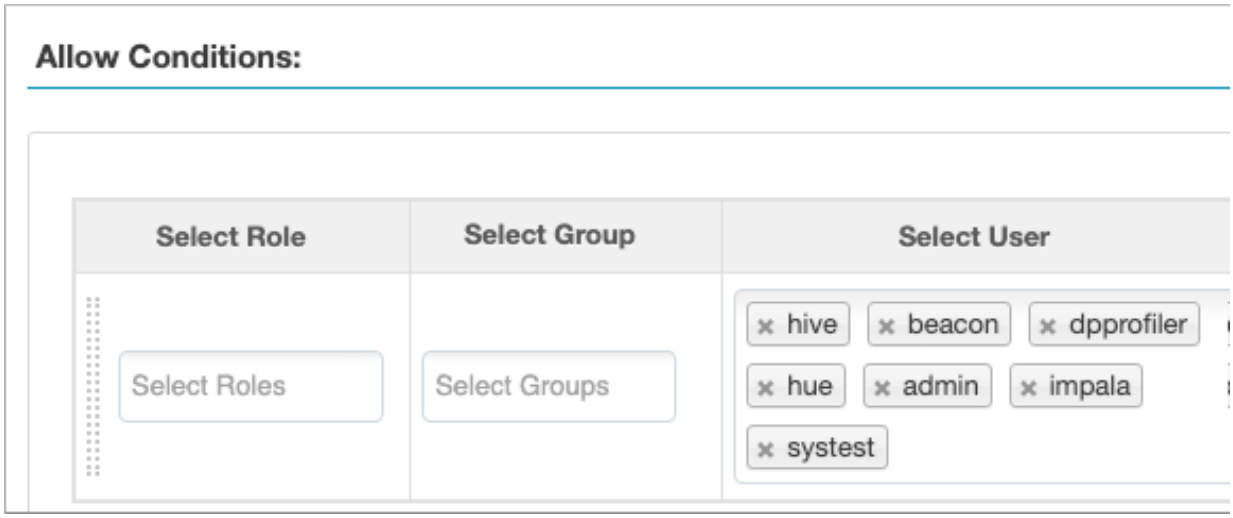

For more information about granting permissions, see Configure a resource-based policy: Hadoop-SQL.

7. Add the RW Storage permission to the policy.
8. Save your changes.

Creating a SQL policy to query an Iceberg table

You learn how to set up the second required policy for using Iceberg. This policy manages SQL query access to Iceberg tables.

About this task

You create a Hadoop SQL policy to allow roles, groups, or users to query an Iceberg table in a database. In this task, you see an example of just one of many ways to configure the policy conditions. You grant (allow) the selected roles, groups, or users the following add or edit permissions on the table: Select, Update, Create, Drop, Alter, and All. You can also deny permissions.

For more information about creating this policy, see Ranger documentation.

Procedure

1. Log into Ranger Admin Web UI.
   The Ranger Service Manager appears.
2. Click Add New Policy.
3. Fill in required fields.
   For example, enter the following required settings:
   - In Policy Name, enter the name of the policy, for example IcebergPolicy1.
   - In database, enter the name of the database controlled by this policy, for example icedb.
   - In table, enter the name of the table controlled by this policy, for example icetable.
   - In columns, enter the name of the column controlled by this policy, for example enter the wildcard asterisk (*) to allow access to all columns of icetable.
   - Accept defaults for other settings.

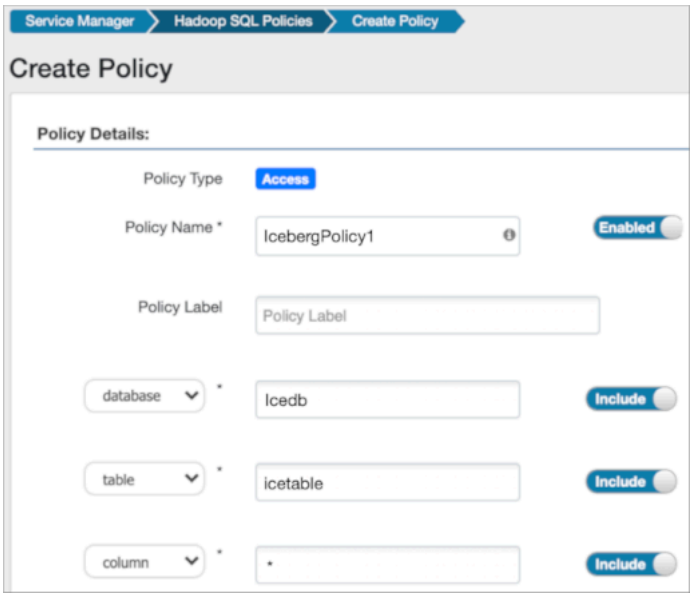

4. Scroll down to Allow Conditions, and select the roles, groups, or users you want to access the table.
   You can use Deny All Other Accesses to deny access to all other roles, groups, or users other than those specified in the allow conditions for the policy.

5. Select permissions to grant.
   For example, select Create, Select, and Alter. Alternatively, to provide the broadest permissions, select All.

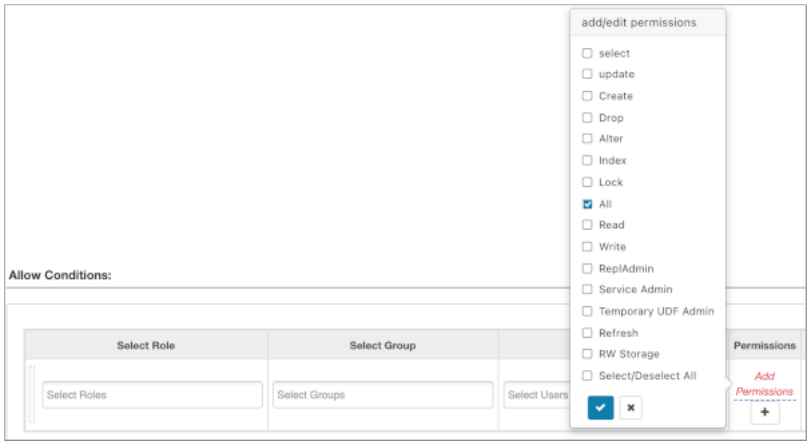

   Ignore RW Storage and other permissions not named after SQL queries. These are for future implementations.
6. Click Add.

Creating an Iceberg table

A step-by-step procedure describes how to create an Apache Iceberg table from a client connection to Hive or Impala, or from Hue in Cloudera Data Hub. You see how to access and use the query editor Hue to create an Iceberg table.

About this task

In this task, from a Data Hub cluster, you open Hue, and use Hive or Impala to create a table.

Before you begin

- You must meet the prerequisites to query Iceberg tables, including obtaining Ranger access permissions.

Procedure

1. Log into Cloudera, and click Data Hub Clusters.

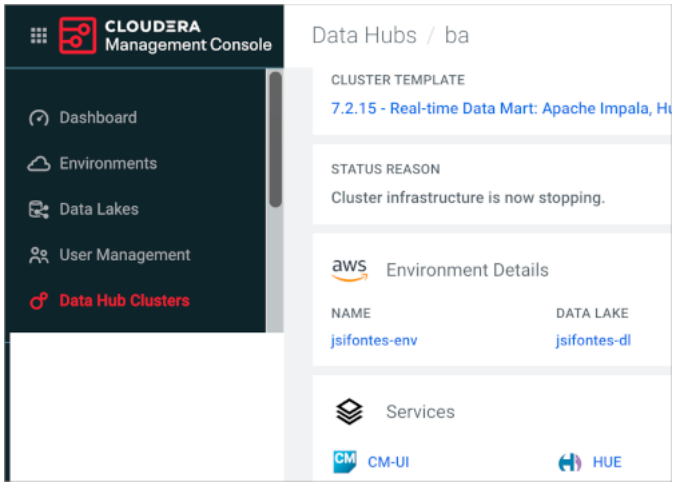

2. Click Hue.
3. Select a database.

4. Enter a query to create a simple Iceberg table in the default Parquet format.
   Hive example:

```
CREATE EXTERNAL TABLE ice_t1 (i int, s string, ts timestamp, d date)
STORED BY ICEBERG;
```

   Impala example:

```
CREATE TABLE ice_t2 (i int, s string, ts timestamp, d date)
STORED BY ICEBERG;
```

   In Cloudera, CREATE EXTERNAL TABLE, and just CREATE TABLE, are valid from Hive. You use the EXTERNAL keyword from Hive to create the Iceberg table to purge the data when you drop the table. In Cloudera, from Impala, you must use CREATE TABLE to initialize the Iceberg table.

5. Click to run the query.

Creating an Iceberg partitioned table

The ease of use of the Iceberg partitioning is clear from an example of how to partition a table using the backward compatible, identity-partition syntax. Alternatively, you can partition an Iceberg table by column values from Hive or Impala.

About this task

You can specify partitioning that is backward compatible with Iceberg V1 using the PARTITION BY clause. This type of table is called an identity-partitioned table. For more information about partitioning, see the Apache Iceberg documentation.

Procedure

1. Select, or use, a database.
2. Create an identity-partitioned table and run the query.
   Hive:

```
CREATE EXTERNAL TABLE ice_ext1 (i int, s string, ts timestamp, d date) P
ARTITIONED BY (state string)
STORED BY ICEBERG
STORED AS ORC;
```

   Impala:

```
CREATE TABLE ice_ext2 (i int, s string, ts timestamp, d date) PARTITIONED
 BY (state string)
STORED BY ICEBERG;
```

3. Create a table and specify an identity transform, such as bucket, truncate, or date, using the Iceberg V2 PARTITION BY SPEC clause.
   Hive:

```
CREATE TABLE ice_t_transforms (i int, s string, ts timestamp, d date)
PARTITIONED BY SPEC (TRUNCATE(10, i), BUCKET(11, s), YEAR(ts))
```

```
STORED BY ICEBERG;
```

Impala:

```
CREATE TABLE ice_t_transforms (i int, s string, ts timestamp, d date)
PARTITIONED BY SPEC (TRUNCATE(10, i), BUCKET(11, s), YEAR(ts))
STORED AS ICEBERG;
```

Related Information

Partition transform feature

Expiring snapshots

You can expire snapshots of an Iceberg table using an ALTER TABLE query. You should periodically expire snapshots to delete data files that are no longer needed, and reduce the size of table metadata.

About this task

Each write to an Iceberg table creates a new snapshot, or version, of a table. You can use snapshots for time-travel queries, or to roll back a table to a valid snapshot. Snapshots accumulate until they are expired by the expire_snapshots operation.

Procedure

1. Enter a query to expire snapshots older than the following timestamp: '2021-12-09 05:39:18.689000000'

```
ALTER TABLE test_table EXECUTE EXPIRE_SNAPSHOTS('2021-12-09 05:39:18.689
000000');
```

2. Enter a query to expire snapshots having between December 10, 2022 and November 8, 2023.

```
ALTER TABLE test_table EXECUTE EXPIRE_SNAPSHOTS BETWEEN ('2022-12-10 00:
00:00.000000000') AND ('2023-11-08 00:00:00.000000000');
```

Inserting data into a table

You can append data to an Iceberg table by inserting values or by selecting the data from another table. You can update data, replacing the old data.

You use the INSERT command in one of the following ways to populate an Iceberg table from Hive:

- INSERT INTO t VALUES (1, ‘asf’, true);
- INSERT INTO t SELECT * FROM s;
- INSERT OVERWRITE t SELECT * FROM s;

Examples

```
INSERT INTO t VALUES (1, ‘asf’, true);
INSERT INTO t SELECT * FROM s;
INSERT OVERWRITE t SELECT * FROM s;
```

Migrating a Hive table to Iceberg

You see how to use a simple ALTER TABLE statement from Hive or Impala to migrate an external Hive table to an Iceberg table. You see how to configure table input and output by setting table properties.

About this task

Note: To prevent loss of new and old table data during migration of a table to Iceberg, do not drop, move, or change the old table during migration.

In this task, from a Cloudera Data Hub cluster, you open Hue, and use Hive or Impala to create a table. In Impala, you can configure the NUM_THREADS_FOR_TABLE_MIGRATION query option to tweak the performance of the table migration. It sets the maximum number of threads to be used for the migration process but could also be limited by the number of CPUs. If set to zero then the number of available CPUs on the coordinator node is used as the maximum number of threads. Parallelism occurs on the basis of data files within a partition, which means one partition is processed at a time with multiple threads processing the files inside the partition. In case there is only one file in each partition, sequential execution occurs.

Before you begin

You must meet the prerequisites to query Iceberg tables, including obtaining Ranger access permissions.

Procedure

1. Log into Cloudera, and click Data Hub Clusters.

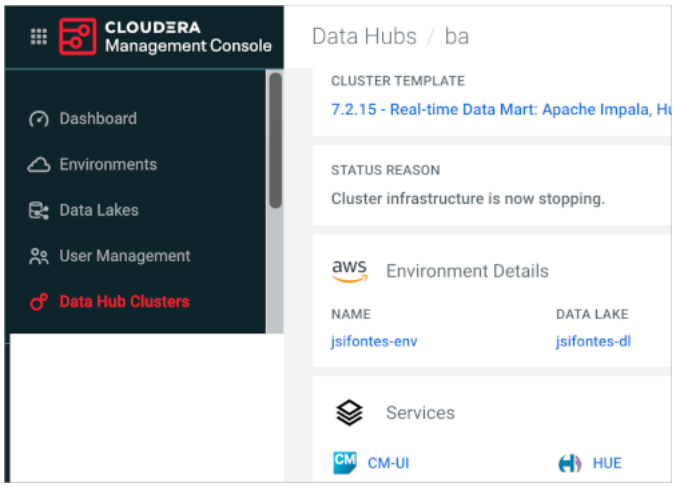

2. Click Hue.
3. Select a database.
4. Enter a query to use a database.
   For example:

```
USE mydb;
```

5. Enter a query to migrate an existing external Hive table to an Iceberg v2 table.
   Hive example:

```
ALTER TABLE tbl
SET TBLPROPERTIES ('storage_handler'='org.apache.iceberg.mr.hive.HiveIc
ebergStorageHandler',
```

```
 'format-version' = '2');
```

Impala example, which requires two queries:

```
ALTER TABLE table_name CONVERT TO ICEBERG;
ALTER TABLE table_name SET TBLPROPERTIES ('format-version'='2');
```

The first ALTER command converts the Hive table to an Iceberg V1 table.

6. Click to run the queries.
   An Iceberg V2 table is created, replacing the Hive table.

Selecting an Iceberg table

You see an example of how to read an Apache Iceberg table, and understand the advantages of Iceberg.

About this task

Working with timestamps in Iceberg, you do not need to know whether the table is actually partitioned by month, day or hour, based on the timestamp value. You can simply supply a predicate for the timestamp value and Iceberg converts the timestamp to month/day/hour transparently. Hive/Impala must maintain actual partition values in a separate column (for example, ts_month or ts_day). Forgetting to reference the derived partition column in your query can lead to inadvertent full table scans.

By default iceberg.table_identifier is not set in Cloudera, so you can use the familiar <db_name.<table_name> in queries.

Procedure

1. Use a database.
   For example:

```
USE mydatabase;
```

2. Query an Iceberg table partitioned by city.
   For example:

```
SELECT * FROM ice_t2 WHERE city="Bangalore";
```

Running time travel queries

You query historical snapshots of data using the FOR SYSTEM_TIME AS OF '<timestamp>' FOR SYSTEM_VERSION AS OF <snapshot_id> clauses in a select statement. You see how to use AS OF to specify a snapshot of your Iceberg data at a certain time.

About this task

You can inspect the history of an Iceberg table to see the snapshots. You can query the metadata of the Iceberg table using a SELECT ... AS OF statement to run time travel queries. You use history information from a query of the database to identify and validate snapshots, and then query a specific snapshot AS OF a certain Timestamp value.

Before you begin

- You must be aware of the table history.

  However, this can include commits that have been rolled back.
- You must have access to valid snapshots.

Procedure

1. View the table history.

```
SELECT * FROM db.table.history;
```

2. Check the valid snapshots of the table.

```
SELECT * FROM db.table.snapshots;
```

3. Query a specific snapshot by providing the timestamp and snapshot_id.

```
SELECT * FROM T
FOR SYSTEM_TIME AS OF <TIMESTAMP>;
SELECT * FROM t
FOR SYSTEM_VERSION AS OF <SNAPSHOT_ID>;
```

Updating an Iceberg partition

You see how to update Iceberg table partitioning in an existing table and then how to change the partitioning to be more granular.

About this task

Partition information is stored logically, and only in table metadata. When you update a partition spec, the old data written with an earlier spec remains unchanged. New data is written using the new spec in a new layout. Metadata for each of the partition versions is separate.

Procedure

1. Create a table partitioned by year.
   Hive

```
CREATE EXTERNAL TABLE ice_t (i int, j int, ts timestamp)
PARTITIONED BY SPEC (truncate(5, j), year(ts))
STORED BY ICEBERG;
```

   Impala:

```
CREATE TABLE ice_t (i int, j int, ts timestmap)
PARTITIONED BY SPEC (truncate(5, j), year(ts))
STORED BY ICEBERG;
```

2. Split the data into manageable files using buckets.

```
ALTER TABLE ice_t SET PARTITION SPEC (bucket(13, i));
```

3. Partition the table by month.

```
ALTER TABLE ice_t SET PARTITION SPEC  (truncate(5, j), month(ts));
```

Test driving Iceberg from Impala

You complete a task that creates Iceberg tables from Impala with mock data that you can test drive using your own queries. You learn how to work with partitioned tables.

Before you begin

- You must obtain Ranger access permissions.

Procedure

1. In Impala, use a database.
2. Create an Impala table to hold mock data for this task.

```
create external table mock_rows stored as parquet as
select x from (
with v as (values (1 as x), (1), (1), (1), (1))
select v.x from v, v v2, v v3, v v4, v v5, v v6
) a;
```

3. Create another Impala table based on mock_rows.

```
create external table customer_demo stored as parquet as
select
FROM_TIMESTAMP(DAYS_SUB(now() , cast ( TRUNC(RAND(7)*365*1) as bigint)), '
yyyy-MM') as year_month,
DAYS_SUB(now() , cast ( TRUNC(RAND(7)*365*1) as bigint)) as ts,
CONCAT(
  cast ( TRUNC(RAND(1) * 250 + 2) as string), '.' ,
  cast ( TRUNC(RAND(2) * 250 + 2) as string), '.',
  cast ( TRUNC(RAND(3) * 250 + 2) as string), '.',
  cast ( TRUNC(RAND(4) * 250 + 2) as string)
) as ip,
CONCAT("USER_", cast ( TRUNC(RAND(4) * 1000) as string),'@somedomain.com')
 as email,
CONCAT("USER_", cast ( TRUNC(RAND(5) * 1000) as string)) as username,
CONCAT("USER_", cast ( TRUNC(RAND(6) * 100) as string)) as country,
cast( RAND(8)*10000 as double) as metric_1,
cast( RAND(9)*10000 as double) as metric_2,
cast( RAND(10)*10000 as double) as metric_3,
cast( RAND(11)*10000 as double) as metric_4,
cast( RAND(12)*10000 as double) as metric_5
from mock_rows
;
```

4. Create another Impala table based on mock_rows.

```
create external table customer_demo2 stored as parquet as
select
FROM_TIMESTAMP(DAYS_SUB(now() , cast ( TRUNC(RAND(7)*365*1) as bigint)),
'yyyy-MM') as year_month,
DAYS_SUB(now() , cast ( TRUNC(RAND(7)*365*1) as bigint)) as ts,
CONCAT(
  cast ( TRUNC(RAND(1) * 250 + 2) as string), '.' ,
  cast ( TRUNC(RAND(2) * 250 + 2) as string), '.',
  cast ( TRUNC(RAND(3) * 250 + 2) as string), '.',
  cast ( TRUNC(RAND(4) * 250 + 2) as string)
) as ip,
```

```
CONCAT("USER_", cast ( TRUNC(RAND(4) * 1000) as string),'@somedomain.com')
 as email,
CONCAT("USER_", cast ( TRUNC(RAND(5) * 1000) as string)) as username,
CONCAT("USER_", cast ( TRUNC(RAND(6) * 100) as string)) as country,
cast( RAND(8)*10000 as double) as metric_1,
cast( RAND(9)*10000 as double) as metric_2,
cast( RAND(10)*10000 as double) as metric_3,
cast( RAND(11)*10000 as double) as metric_4,
cast( RAND(12)*10000 as double) as metric_5
from mock_rows
;
```

5. Create an Iceberg table from the customer_demo table.

```
CREATE TABLE customer_demo_iceberg STORED BY ICEBERG AS SELECT * FROM cu
stomer_demo;
```

6. Insert into the customer_demo_iceberg table the results of selecting all data from the customer_demo2 table.

```
INSERT INTO customer_demo_iceberg select * from customer_demo2;
INSERT INTO customer_demo_iceberg select * from customer_demo2;
INSERT INTO customer_demo_iceberg select * from customer_demo2;
```

7. Create an Iceberg table partitioned by the year_month column and based on the customer_demo_iceberg table.

```
CREATE TABLE customer_demo_iceberg_part PARTITIONED BY(year_month) STORED
 BY ICEBERG
AS SELECT ts, ip , email, username , country, metric_1 , metric_2 , metric
_3 , metric_4 , metric_5, year_month
FROM customer_demo_iceberg;
```

8. Split the partitioned data into manageable files.

```
ALTER TABLE customer_demo_iceberg_part SET PARTITION SPEC (year_month,BU
CKET(15, country));
```

9. Insert the results of reading the customer_demo_iceberg table into the partitioned table.

```
INSERT INTO customer_demo_iceberg_part (year_month, ts, ip, email, usern
ame, country, metric_1, metric_2, metric_3, metric_4, metric_5)
SELECT year_month, ts, ip, email, username, country, metric_1, metric_2,
 metric_3, metric_4, metric_5
FROM customer_demo_iceberg;
```

10. Run time travel queries on the Iceberg tables, using the history output to get the snapshot id, and substitute the id in the second SELECT query.

```
SELECT * FROM customer_demo_iceberg FOR SYSTEM_TIME AS OF '2021-12-09 05
:39:18.689000000' LIMIT 100;
DESCRIBE HISTORY customer_demo_iceberg;
SELECT * FROM customer_demo_iceberg FOR SYSTEM_VERSION AS OF <snapshot
id> LIMIT 100;
```

Hive demo data

To test drive Iceberg from Hive, you need to set up Hive demo data.

Iceberg Database creation and setup

The Airlines demo data for Iceberg is stored in the airline_online_iceberg database. The following queries created and set up this database.

```
create database if not exists airline_ontime_iceberg;
use airline_ontime_iceberg;
set hive.vectorized.execution.enabled=false;
set hive.stats.column.autogather=false;
```

Hive external table creation

The following Hive external tables were created in the airline_online_iceberg database:

- airports
- airlines
- planes
- flights

```
create external table if not exists airports (
    iata string,
    airport string,
    city string,
    state double,
    country string,
    lat double,
    lon double
)
stored as orc;

create external table if not exists airlines (
    code string,
    description string
)
stored as orc;
create external table if not exists planes (
    tailnum string,
    owner_type string,
    manufacturer string,
    issue_date string,
    model string,
    status string,
    aircraft_type string,
    engine_type string,
    year int
)
stored as orc;

create external table if not exists flights (
    month int,
    dayofmonth int,
    dayofweek int,
    deptime  int,
    crsdeptime int,
    arrtime int,
    crsarrtime int,
    uniquecarrier string,
    flightnum int,
    tailnum string,
    actualelapsedtime int,
    crselapsedtime int,
    airtime int,
```

```
    arrdelay int,
    depdelay int,
    origin string,
    dest string,
    distance int,
    taxiin int,
    taxiout int,
    cancelled int,
    cancellationcode string,
    diverted string,
    carrierdelay int,
    weatherdelay int,
    nasdelay int,
    securitydelay int,
    lateaircraftdelay int
)
partitioned by (year int)
stored as orc;
```

Load data into the newly created tables

```
load data inpath '${datapath}/airline_ontime_iceberg.db/airports' into table
 airports;

load data inpath '${datapath}/airline_ontime_iceberg.db/airlines' into table
 airlines;

load data inpath '${datapath}/airline_ontime_iceberg.db/planes' into table p
lanes;

load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=1995' i
nto table flights partition (year=1995);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=1996'
into table flights partition (year=1996);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=1997'
 into table flights partition (year=1997);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=1998'
 into table flights partition (year=1998);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=1999' i
nto table flights partition (year=1999);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2000'
into table flights partition (year=2000);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2001'
 into table flights partition (year=2001);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2002'
 into table flights partition (year=2002);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2003' i
nto table flights partition (year=2003);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2004'
into table flights partition (year=2004);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2005'
 into table flights partition (year=2005);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2006'
 into table flights partition (year=2006);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2007' i
nto table flights partition (year=2007);
load data inpath '${datapath}/airline_ontime_iceberg.db/flights/year=2008'
into table flights partition (year=2008);
```

Convert these existing Hive external tables to Iceberg tables

```
ALTER TABLE planes ADD CONSTRAINT planes_pk PRIMARY KEY (tailnum) DISABLE NO
VALIDATE;
ALTER TABLE flights ADD CONSTRAINT planes_fk FOREIGN KEY (tailnum) REFEREN
CES planes(tailnum) DISABLE NOVALIDATE RELY;

ALTER TABLE airlines ADD CONSTRAINT airlines_pk PRIMARY KEY (code) DISABLE
 NOVALIDATE;
ALTER TABLE flights ADD CONSTRAINT airlines_fk FOREIGN KEY (uniquecarrier)
REFERENCES airlines(code) DISABLE NOVALIDATE RELY;

ALTER TABLE airports ADD CONSTRAINT airports_pk PRIMARY KEY (iata) DISABLE N
OVALIDATE;
ALTER TABLE flights ADD CONSTRAINT airports_orig_fk FOREIGN KEY (origin)
REFERENCES airports(iata) DISABLE NOVALIDATE RELY;
ALTER TABLE flights ADD CONSTRAINT airports_dest_fk FOREIGN KEY (dest) RE
FERENCES airports(iata) DISABLE NOVALIDATE RELY;

ALTER TABLE airports SET TBLPROPERTIES ('storage_handler'='org.apache.iceb
erg.mr.hive.HiveIcebergStorageHandler');
ALTER TABLE airlines SET TBLPROPERTIES ('storage_handler'='org.apache.icebe
rg.mr.hive.HiveIcebergStorageHandler');
ALTER TABLE planes SET TBLPROPERTIES ('storage_handler'='org.apache.iceberg.
mr.hive.HiveIcebergStorageHandler');
ALTER TABLE flights SET TBLPROPERTIES ('storage_handler'='org.apache.iceber
g.mr.hive.HiveIcebergStorageHandler');
```

Test driving Iceberg from Hive

You learn how to access the Hive demo data, which you can use to get hands-on experience running Iceberg queries.

About this task

Query sample airline demo data in Hue.

Before you begin

- You must meet the prerequisites to query Iceberg tables.
- You obtained the Ranger permissions to run Hive queries.

Procedure

1. Connect to Hive running in a Cloudera Data Hub cluster.
2. Run the queries in the previous topic, "Hive demo data" to set up the following databases: airline_ontime_iceberg, airline_ontime_orc, airline_ontime_parquet.
3. Use the airline_ontime_iceberg database.
4. Take a look at the tables in the airline_ontime_iceberg database.

```
USE airline_ontime_iceberg;
```

```
SHOW TABLES;
```

Flights is the fact table. It has 100M rows and three dirmensions, ariline, airports, and planes. This records flights for more than 10 years in the US, and includes the following details:

- origin
- destination
- delay
- air time

5. Query the demo data from Hive.
   For example, find the flights that departed each year, by IATA code, airport, city, state, and country. Find the average departure delay.

```
SELECT f.month, a.iata, a.airport, a.city, a.state, a.country
FROM flights f,
airports a
WHERE f.origin = a.iata
GROUP BY
f.month,
a.iata,
a.airport,
a.city,
a.state,
a.country
HAVING COUNT(*) > 10000
ORDER BY AVG(f.DepDelay) DESC
LIMIT 10;
```

Output appears as follows:

```
+----------+---------+-----------------------------------+---------------
+----------+------------+
| f.month  | a.iata  |             a.airport              |    a.city
 | a.state  | a.country  |
+----------+---------+-----------------------------------+---------------
+----------+------------+
| 12       | ORD     | Chicago O'Hare International       | Chicago
 | NULL     | USA        |
| 6        | EWR     | Newark Intl                        | Newark
 | NULL     | USA        |
| 7        | JFK     | John F Kennedy Intl                | New York
 | NULL     | USA        |
| 6        | IAD     | Washington Dulles International    | Chantilly
 | NULL     | USA        |
| 7        | EWR     | Newark Intl                        | Newark
 | NULL     | USA        |
| 6        | PHL     | Philadelphia Intl                  | Philadelphia
 | NULL     | USA        |
| 1        | ORD     | Chicago O'Hare International       | Chicago
 | NULL     | USA        |
| 6        | ORD     | Chicago O'Hare International       | Chicago
 | NULL     | USA        |
| 7        | ATL     | William B Hartsfield-Atlanta Intl  | Atlanta
 | NULL     | USA        |
| 12       | MDW     | Chicago Midway                     | Chicago
 | NULL     | USA        |
+----------+---------+-----------------------------------+---------------
+----------+------------+
10 rows selected (103.812 seconds)
```

6. Split the partitioned data into manageable files.

```
ALTER TABLE airports SET PARTITION SPEC (iata,BUCKET(15, country));
```

Iceberg data types

References include Iceberg data types and a table of equivalent SQL data types by Hive/Impala SQL engine types.

Iceberg supported data types

Table 2:

Iceberg data type	SQL data type	Hive	Impala
binary		BINARY	BINARY
boolean	BOOLEAN	BOOLEAN	BOOLEAN
date	DATE	DATE	DATE
decimal(P, S)	DECIMAL(P, S)	DECIMAL (P, S)	DECIMAL (P, S)
double	DOUBLE	DOUBLE	DOUBLE
fixed(L)		BINARY	Not supported
float	FLOAT	FLOAT	FLOAT
int	TINYINT, SMALLINT, INT	INTEGER	INTEGER
list	ARRAY	ARRAY	Read only
long	BIGINT	BIGINT	BIGINT
map	MAP	MAP	Read only
string	VARCHAR, CHAR	STRING	STRING
struct	STRUCT	STRUCT	Read only
time		STRING	Not supported
timestamp	TIMESTAMP	TIMESTAMP	TIMESTAMP (see limitation below)
timestamptz	TIMESTAMP WITH LOCAL TIME ZONE	Use TIMESTAMP WITH LOCAL TIMEZONE for handling these in queries	Read timestamptz into TIMESTAMP values Writing not supported
uuid	none	STRING Writing to Parquet is not supported	Not supported

Data type limitations

An implicit conversion to an Iceberg type occurs only if there is an exact match; otherwise, a cast is needed. For example, to insert a VARCHAR(N) column into an Iceberg table you need a cast to the VARCHAR type as Iceberg does not support the VARCHAR(N) type. To insert a SMALLINT or TINYINT into an Iceberg table, you need a cast to the INT type as Iceberg does not support these types.

Iceberg supports two timestamp types:

- timestamp (without timezone)
- timestamptz (with timezone)

With Spark 3.4, Spark SQL supports a timestamp with local timezone (TIMESTAMP_LTZ) type and a timestamp without timezone (TIMESTAMP_NTZ) type, with TIMESTAMP defaulting to the TIMESTAMP_LTZ type. However, this can be configured by setting the spark.sql.timestampType (the default value is TIMESTAMP_LTZ).

When creating an Iceberg table using Spark SQL, if spark.sql.timestampType is set to TIMESTAMP_LTZ, TIMESTAMP is mapped to Iceberg's timestampz type. If spark.sql.timestampType is set to TIMESTAMP_NTZ, then TIMESTAMP is mapped to Iceberg's timestamp type.

Impala is unable to write to Iceberg tables with timestamptz columns. For interoperability, when creating Iceberg tables from Spark, you can use the Spark configuration, spark.sql.timestampType=TIMESTAMP_NTZ.

For consistent results across query engines, all the engines must be running in UTC.

Unsupported data types

Impala does not support the following Iceberg data types:

- TIMESTAMPTZ (only read support)
- TIMESTAMP in tables in AVRO format
- FIXED
- UUID

Iceberg table properties

The Cloudera environment for querying tables from Hive overrides some Iceberg table properties. You learn which table properties are supported for querying tables from Impala.

Iceberg documentation describes all the properties for configuring tables. This documentation focuses on key properties for working with Iceberg tables in Cloudera.

Iceberg supports concurrent writes by default. You can tune Iceberg v2 table properties for concurrent writes. You set the following properties if you plan to have concurrent writers on Iceberg v2 tables:

- commit.retry.min-wait-ms
- commit.retry.num-retries

Cloudera supports adding the Parquet compression type using table properties. For more information, see Iceberg documentation about Compression Types.

You can use the Alter Table feature to set a property. From Hive, the following Iceberg table property overrides are in effect:

- iceberg.mr.split.size overrides read.split.target-size.
- read.split.open-file-cost is overridden.

You can tune Iceberg v2 table properties for concurrent writes. From Impala, the following subset of Iceberg table properties are supported:

- history.expire.min-snapshots-to-keep

  Valid values: integers. Default = 1
- write.format.default

  Valid value: Parquet
- write.metadata.delete-after-commit.enabled

  Valid values: true or false.
- write.metadata.previous-versions-max

  Valid values: integers. Default = 100.

- write.parquet.compression-codec

  Valid values: GZIP, LZ4, NONE, SNAPPY (default value), ZSTD
- write.parquet.compression-level

  Valid values: 1 - 22. Default = 3
- write.parquet.row-group-size-bytes

  Valid values: 8388608 (or 8 MB) - 2146435072 (or 2047MB). Overiden by PARQUET_FILE_SIZE.
- write.parquet.page-size-bytes

  Valid values: 65536 (or 64KB) - 1073741824 (or 1GB).
- write.parquet.dict-size-bytes

  Valid values: 65536 (or 64KB) - 1073741824 (or 1GB)

..

Apache Iceberg Overview

Date published: 2022-07-24
Date modified: 2022-07-24

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Iceberg overview..... 4

Iceberg overview

Apache Iceberg is a table format for huge analytics datasets that defines how metadata is stored and data files are organized. Iceberg is also a library that compute engines can use to read/write a table.

You can efficiently query large Iceberg tables on object stores. Iceberg supports concurrent reads and writes on all storage media. You can use Iceberg when a single table contains tens of petabytes of data, and you can read these tables without compromising performance. By default, Hive and Impala use the Iceberg HiveCatalog for Metastore management of Iceberg tables. The catalog stores Iceberg metadata, including the location of the table. Iceberg catalog use of Hive metastore (HMS) for catalog operations is lightweight. Iceberg relieves HMS pressure by storing partition information in metadata files on the file system/object store instead of within the HMS. This architecture supports rapid scaling without performance hits.

In Cloudera Public Cloud, you can deploy Iceberg based applications across multiple clouds including AWS, Azure, and Google Cloud.

Key features overview

From Hive and Impala, you can read and write V1 or V2 Iceberg tables. The following features are included:

- Reads of Iceberg V2 tables that have had row-level deletes or updates makes Apache Iceberg ACID-compliant with serializable isolation and an optimistic concurrency model
- Materialized views of Iceberg tables
- Enhanced maintenance features, such as expiring and removing old snapshots and compaction of small files
- Performance and scalability enhancements

Iceberg table security and visualization

Apache Iceberg integrates Apache Ranger for security. You can use Ranger integration with Hive and Impala to apply fine-grained access control to sensitive data in Iceberg tables. Iceberg is also integrated with Cloudera Data Visualization for creating dashboards and other graphics of your Iceberg data.

Supported ACID transaction properties

Iceberg supports atomic and isolated database transaction properties. Writers work in isolation, not affecting the live table, and perform an atomic swap for metadata pointers only when the transactions are committed, making the changes in one atomic commit.

Iceberg uses snapshots to guarantee isolated reads and writes. You see a consistent version of table data without locking the table. Readers always see a consistent version of the data without the need to lock the table. Writers work in isolation, not affecting the live table.

Iceberg partitioning

The Iceberg partitioning technique has performance advantages over conventional partitioning, such as Apache Hive partitioning. Iceberg hidden partitioning is easier to use. Iceberg supports in-place partition evolution; to change a partition, you do not rewrite the entire table to add a new partition column, and queries do not need to be rewritten for the updated table. Iceberg continuously gathers data statistics, which supports additional optimizations, such as partition pruning.

Iceberg uses multiple layers of metadata files to find and prune data. Impala, Spark, and Flink keep track of data at the folder level and not at the file level, performing file list operations when working with data in a table. Performance problems occur during the execution of multiple directory listings. Iceberg keeps track of a complete list of files within a table using a persistent tree structure. Changes to an Iceberg table use an atomic object/file level commit to update the path to a new snapshot file. The snapshot points to the individual data files through manifest files.

The manifest files track data files across partitions, storing partition information and column metrics for each data file. A manifest list is an additional index for pruning entire manifests. File pruning increases efficiency.

Iceberg relieves Hive metastore (HMS) pressure by storing partition information in metadata files on the file system/object store instead of within the HMS. This architecture supports rapid scaling without performance hits.

Related Information

Iceberg support from Spark in Cloudera Data Engineering

Apache Software Foundation Iceberg Docs

Using Apache Iceberg

Cloudera Runtime 7.3.1

Configuring Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring Impala.....4
Configuring Load Balancer for Impala.....5

Configuring Impala

You must review and configure the mandatory and recommended settings if you installed Impala without Cloudera Manager or if you want to customize your environment after installing Impala. If you installed Impala using Cloudera Manager, some of these configurations are completed automatically.

In some cases, depending on the level of Impala, Cloudera, and Cloudera Manager, you might need to add particular component configuration details in one of the free-form fields on the Impala configuration pages in Cloudera Manager.

- You must enable short-circuit reads, whether or not Impala was installed through Cloudera Manager. This setting goes in the Impala configuration settings, not the Hadoop-wide settings.
- If you installed Impala in an environment that is not managed by Cloudera Manager, you must enable block location tracking, and you can optionally enable native checksumming for optimal performance.

Short-Circuit Reads

Enabling short-circuit reads allows Impala to read local data directly from the file system. This removes the need to communicate through the DataNodes, improving performance. This setting also minimizes the number of additional copies of data. Short-circuit reads requires libhadoop.so (the Hadoop Native Library) to be accessible to both the server and the client. You must install it from an .rpm, .deb, or parcel to use short-circuit local reads.

Note: If you use Cloudera Manager, you can enable short-circuit reads through a checkbox in the user interface and that setting takes effect for Impala as well.

To Enable Short-Circuit Reads

You can enable short-circuit reads through a checkbox available under Configuration tab for both Impala and HDFS.

1. In Cloudera Manager Clusters select the Impala service, for example, IMPALA.
2. On the Configuration tab, search for dfs.client.read.shortcircuit.
3. Accept the default (enabled), or check to enable the dfs.client.read.shortcircuit property to read the HDFS file blocks directly.
4. Do the above for HDFS service too by clicking Cloudera Manager Clusters and by selecting the HDFS service.
5. Save the changes and Restart the service.

To configure DataNodes for short-circuit reads:

1. On all Impala nodes, configure the following attributes from the HDFS service as shown:
   a. In Cloudera Manager Clusters select the HDFS service, for example, HDFS.
   b. On the Configuration tab, search for dfs.domain.socket.path and set the attribute.
   c. Search for and set the attribute, if necessary, dfs.client.file-block-storage-locations.timeout.millis.
   d. Search for and set the attribute, if necessary, dfs.datanode.hdfs-blocks-metadata.enabled
2. After applying these changes, restart all DataNodes.

Block Location Tracking

Enabling block location metadata allows Impala to know which disk data blocks are located on, allowing better utilization of the underlying disks. Impala will not start unless this setting is enabled.

To enable block location tracking:

1. For each DataNode, enable the following attribute dfs.datanode.hdfs-blocks-metadata.enabled file:
   a. In Cloudera Manager Clusters select the HDFS service, for example, HDFS.
   b. On the Configuration tab, search for dfs.datanode.hdfs-blocks-metadata.enabled and enable the attribute if not already enabled.

2. After applying these changes, restart all DataNodes.

Native Checksumming

Enabling native checksumming causes Impala to use an optimized native library for computing checksums, if that library is available.

To enable native checksumming:

If you installed Cloudera from packages, the native checksumming library is installed and setup correctly, and no additional steps are required.

If you installed by other means, native checksumming may not be available due to missing shared objects. Finding the message "Unable to load native-hadoop library for your platform... using builtin-java classes where applicable" in the Impala logs indicates native checksumming may be unavailable.

To enable native checksumming, you must build and install libhadoop.so (the Hadoop Native Library).

Configuring Load Balancer for Impala

For most clusters that have multiple users and production availability requirements, you might want to set up a load-balancing proxy server to relay requests to and from Impala.

When using a load balancer for Impala, applications connect to a single well-known host and port, rather than keeping track of the hosts where a specific Impala daemon is running. The load balancer also lets the Impala nodes share resources to balance out the work loads.

Set up a software package of your choice to perform these functions.

Most considerations for load balancing and high availability apply to the impalad daemons. The statestored and cata logd daemons do not have special requirements for high availability, because problems with those daemons do not result in data loss.

The following are the general setup steps that apply to any load-balancing proxy software:

1. Select and download a load-balancing proxy software or other load-balancing hardware appliance. It should only need to be installed and configured on a single host, typically on an edge node.
2. Configure the load balancer (typically by editing a configuration file). In particular:
   - To relay Impala requests back and forth, set up a port that the load balancer will listen on.
   - Select a load balancing algorithm.
   - For Kerberized clusters, follow the instructions in Special Proxy Considerations for Clusters Using Kerberos on page 6 below.
3. If you are using Hue or JDBC-based applications, you typically set up load balancing for both ports 21000 and 21050, because these client applications connect through port 21050 while the `impala-shell` command connects through port 21000. See Ports used by Impala for when to use port 21000, 21050, or another value depending on what type of connections you are load balancing.
4. Run the load-balancing proxy server, pointing it at the configuration file that you set up.
5. In Cloudera Manager, navigate to Impala Configuration Impala Daemon Default Group .
6. In the Impala Daemons Load Balancer field, specify the address of the load balancer in the *HOST*:*PORT* format.

   This setting lets Cloudera Manager route all appropriate Impala-related operations through the load-balancing proxy server.
7. For any scripts, jobs, or configuration settings for applications that formerly connected to a specific `impalad` to run Impala SQL statements, change the connection information (such as the -i option in `impala-shell`) to point to the load balancer instead.

Note: The following sections use the HAProxy software as a representative example of a load balancer that you can use with Impala. For information specifically about using Impala with the F5 BIG-IP load balancer, see Impala HA with F5 BIG-IP.

Choosing the Load-Balancing Algorithm

Load-balancing software offers a number of algorithms to distribute requests. Each algorithm has its own characteristics that make it suitable in some situations but not others.

Leastconn

Connects sessions to the coordinator with the fewest connections, to balance the load evenly. Typically used for workloads consisting of many independent, short-running queries. In configurations with only a few client machines, this setting can avoid having all requests go to only a small set of coordinators.

Recommended for Impala with F5.

Source IP Persistence

Sessions from the same IP address always go to the same coordinator. A good choice for Impala workloads containing a mix of queries and DDL statements, such as CREATE TABLE and ALTE R TABLE. Because the metadata changes from a DDL statement take time to propagate across the cluster, prefer to use the Source IP Persistence algorithm in this case. If you are unable to choose Source IP Persistence, run the DDL and subsequent queries that depend on the results of the DDL through the same session, for example by running impala-shell -f *SCRIPT_FILE* to submit several statements through a single session.

Required for setting up high availability with Hue.

Round-robin

Distributes connections to all coordinator nodes. Typically not recommended for Impala.

You might need to perform benchmarks and load testing to determine which setting is optimal for your use case. Always set up using two load-balancing algorithms: Source IP Persistence for Hue and Leastconn for others.

Special Proxy Considerations for Clusters Using Kerberos

In a cluster using Kerberos, applications check host credentials to verify that the host they are connecting to is the same one that is actually processing the request.

In Impala 2.11 and lower versions, once you enable a proxy server in a Kerberized cluster, users will not be able to connect to individual impala daemons directly from impala-shell.

In Impala 2.12 and higher, when you enable a proxy server in a Kerberized cluster, users have an option to connect to Impala daemons directly from `impala-shell` using the -b / --kerberos_host_fqdn `impala-shell` flag. This option can be used for testing or troubleshooting purposes, but not recommended for live production environments as it defeats the purpose of a load balancer/proxy.

Example:

```
impala-shell -i impalad-1.mydomain.com -k -b loadbalancer-1.mydomain.com
```

Alternatively, with the fully qualified configurations:

```
impala-shell --impalad=impalad-1.mydomain.com:21000 --kerberos --kerberos_ho
st_fqdn=loadbalancer-1.mydomain.com
```

To validate the load-balancing proxy server, perform these extra Kerberos setup steps:

1. This section assumes you are starting with a Kerberos-enabled cluster.
2. Choose the host you will use for the proxy server. Based on the Kerberos setup procedure, it should already have an entry impala/*PROXY_HOST@REALM* in its keytab.
3. In Cloudera Manager, navigate to Impala Configuration Impala Daemon Default Group .

4. In the Impala Daemons Load Balancer field, specify the address of the load balancer in the *HOST*:*PORT* format.

   When this field is specified and Kerberos is enabled, Cloudera Manager adds a principal for impala/*PROXY_HOST@REALM* to the keytab for all Impala daemons.
5. Restart the Impala service.

Client Connection to Proxy Server in Kerberized Clusters

When a client connect to Impala, the service principal specified by the client must match the -principal setting, impala/*PROXY_HOST@REALM*, of the Impala proxy server as specified in its keytab. And the client should connect to the proxy server port.

In hue.ini, set the following to configure Hue to automatically connect to the proxy server:

```
[impala]
server_host=PROXY_HOST
impala_principal=impala/PROXY_HOST
```

The following are the JDBC connection string formats when connecting through the load balancer with the load balancer's host name in the principal:

```
jdbc:hive2://PROXY_HOST:LOAD_BALANCER_PORT/;principal=impala/_HOST@REALM
jdbc:hive2://PROXY_HOST:LOAD_BALANCER_PORT/;principal=impala/PROXY_HOST@REALM
```

When starting `impala-shell`, specify the service principal via the -b or --kerberos_host_fqdn flag.

Special Proxy Considerations for TLS/SSL Enabled Clusters

When TLS/SSL is enabled for Impala, the client application, whether impala-shell, Hue, or something else, expects the certificate common name (CN) to match the hostname that it is connected to. With no load balancing proxy server, the hostname and certificate CN are both that of the impalad instance. However, with a proxy server, the certificate presented by the impalad instance does not match the load balancing proxy server hostname. If you try to load-balance a TLS/SSL-enabled Impala installation without additional configuration, you see a certificate mismatch error when a client attempts to connect to the load balancing proxy host.

You can configure a proxy server in several ways to load balance TLS/SSL enabled Impala:

TLS/SSL Bridging

In this configuration, the proxy server presents a TLS/SSL certificate to the client, decrypts the client request, then re-encrypts the request before sending it to the backend impalad. The client and server certificates can be managed separately. The request or resulting payload is encrypted in transit at all times.

TLS/SSL Passthrough

In this configuration, traffic passes through to the backend impalad instance with no interaction from the load balancing proxy server. Traffic is still encrypted end-to-end.

The same server certificate, utilizing either wildcard or Subject Alternate Name (SAN), must be installed on each impalad instance.

TLS/SSL Offload

In this configuration, all traffic is decrypted on the load balancing proxy server, and traffic between the backend impalad instances is unencrypted. This configuration presumes that cluster hosts reside on a trusted network and only external client-facing communication need to be encrypted in-transit.

If you plan to use Auto-TLS, your load balancer must perform TLS/SSL bridging or TLS/SSL offload.

Refer to your load balancer documentation for the steps to set up Impala and the load balancer using one of the options above.

For information specifically about using Impala with the F5 BIG-IP load balancer with TLS/SSL enabled, see Impala HA with F5 BIG-IP.

Example of Configuring HAProxy Load Balancer for Impala

If you are not already using a load-balancing proxy, you can experiment with HAProxy a free, open source load balancer.

Attention: HAProxy is not a CDH component, and Cloudera does not provide the support for HAProxy. Refer to HAProxy for questions and support issues for HAProxy.

This example shows how you might install and configure that load balancer on a Red Hat Enterprise Linux system.

- Install the load balancer:

```
yum install haproxy
```

- Set up the configuration file: /etc/haproxy/haproxy.cfg. See the following section for a sample configuration file.
- Run the load balancer (on a single host, preferably one not running `impalad`):

```
/usr/sbin/haproxy –f /etc/haproxy/haproxy.cfg
```

- In `impala-shell`, JDBC applications, or ODBC applications, connect to the listener port of the proxy host, rather than port 21000 or 21050 on a host actually running `impalad`. The sample configuration file sets haproxy to listen on port 25003, therefore you would send all requests to *HAPROXY_HOST*:25003.

This is the sample haproxy.cfg used in this example:

```
global
    # To have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events.  This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    # 2) configure local2 events to go to the /var/log/haproxy.log
    #   file. A line like the following can be added to
    #   /etc/sysconfig/syslog
    #
    #    local2.*                       /var/log/haproxy.log
    #
    log         127.0.0.1 local0
    log         127.0.0.1 local1 notice
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

    # turn on stats unix socket
    #stats socket /var/lib/haproxy/stats
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#
# You might need to adjust timing values to prevent timeouts.
#
# The timeout values should be dependant on how you use the cluster
# and how long your queries run.
#---------------------------------------------------------------------
defaults
```

```
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    maxconn                 3000
    timeout connect 5000
    timeout client 3600s
    timeout server 3600s

#
# This sets up the admin page for HA Proxy at port 25002.
#
listen stats :25002
    balance
    mode http
    stats enable
    stats auth USERNAME:PASSWORD

# This is the setup for Impala. Impala client connect to load_balancer_hos
t:25003.
# HAProxy will balance connections among the list of servers listed below.
# The list of Impalad is listening at port 21000 for beeswax (impala-shell)
or original ODBC driver.
# For JDBC or ODBC version 2.x driver, use port 21050 instead of 21000.
listen impala :25003
    mode tcp
    option tcplog
    balance leastconn

    server SYMBOLIC_NAME_1 impala-host-1.example.com:21000 check
    server SYMBOLIC_NAME_2 impala-host-2.example.com:21000 check
    server SYMBOLIC_NAME_3 impala-host-3.example.com:21000 check
    server SYMBOLIC_NAME_4 impala-host-4.example.com:21000 check

# Setup for Hue or other JDBC-enabled applications.
# In particular, Hue requires sticky sessions.
# The application connects to load_balancer_host:21051, and HAProxy balances
# connections to the associated hosts, where Impala listens for JDBC
# requests on port 21050.
listen impalajdbc :21051
    mode tcp
    option tcplog
    balance source
    server SYMBOLIC_NAME_5 impala-host-1.example.com:21050 check
    server SYMBOLIC_NAME_6 impala-host-2.example.com:21050 check
    server SYMBOLIC_NAME_7 impala-host-3.example.com:21050 check
    server SYMBOLIC_NAME_8 impala-host-4.example.com:21050 check
```

Important: Hue requires the check option at the end of each line in the above file to ensure HAProxy can detect any unreachable impalad server, and failover can be successful. Without the TCP check, you can encounter an error when the `impalad` daemon to which Hue tries to connect is down.

Note: If your JDBC or ODBC application connects to Impala through a load balancer such as haproxy, be cautious about reusing the connections. If the load balancer has set up connection timeout values, either check the connection frequently so that it never sits idle longer than the load balancer timeout value, or check the connection validity before using it and create a new one if the connection has been closed.

Example of Configuring HAProxy Load Balancer for Impala with Auto-TLS

1. Create unencrypted key file. Auto TLS will not create the unencrypted file so you must create it manually.

```
         openssl rsa -in  /var/lib/cloudera-scm-agent/agent-cert/cm-auto-
host_key.pem -out /etc/haproxy/cm-auto-host_unenckey.pem
         # enter password from /var/lib/cloudera-scm-agent/agent-cert/cm-
auto-host_key.pw
```

Note: The created cm-auto-host_cert_chain_unenckey.pem should be owned by the haproxy user with 600 or 400 privileges.

2. Combine the created unencrypted key with CA certificate:

```
cat /var/lib/cloudera-scm-agent/agent-cert/cm-auto-host_cert_chain.pem /
etc/haproxy/cm-auto-host_unenckey.pem > /etc/haproxy/cm-auto-host_cert_c
hain_unenckey.pem
```

Note: You must reload HAProxy after combining the unencrypted key with CA certificate.

3. Configure haproxy.cfg, most often found in /etc/haproxy/haproxy.cfg. This example shown here uses non-standarad ports however you may want to configure port 21000 and 21050. And you may want to use balance sour ce instead of balance leastconn which is needed for tools like Hue to work properly.

```
frontend impala-auto-tls-thrift
    bind :21013 ssl crt /etc/haproxy/cm-auto-host_cert_chain_unenckey.pem
    mode tcp
    stats enable
    default_backend shell_autotls_backend

backend shell_autotls_backend
    mode tcp
    balance leastconn
    timeout connect 5000ms
    timeout client  3600000ms
    timeout server  3600000ms
    # Impala Nodes
    server ve1311.halxg.cloudera.com ve1311.halxg.cloudera.com:21000 chec
k ssl ca-file /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_cace
rts.pem
    server ve1312.halxg.cloudera.com ve1312.halxg.cloudera.com:21000 check
 ssl ca-file /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts
.pem

frontend impala-autotls-jdbc
    bind :21056 ssl crt /etc/haproxy/cm-auto-host_cert_chain_unenckey.pem
    mode tcp
    stats enable

    default_backend jdbc_autotls_backend

backend jdbc_autotls_backend
    mode tcp
    balance leastconn
    timeout connect 5000ms
    timeout client  3600000ms
    timeout server  3600000ms
    # Impala Nodes
    server ve1311.halxg.cloudera.com ve1311.halxg.cloudera.com:21050 check
 ssl ca-file /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts
.pem
```

```
    server ve1312.halxg.cloudera.com ve1312.halxg.cloudera.com:21050 ch
eck ssl ca-file /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_ca
certs.pem
```

Related Information

Ports used by Impala

Impala Shell Configuration Options

Configuring Kerberos Authentication

Cloudera Runtime 7.3.1

Managing Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ACID Operation **4**
Concepts Used in FULL ACID v2 Tables 4
Key Differences between INSERT-ONLY and FULL ACID Tables 5
Compaction of Data in FULL ACID Transactional Table 6
Managing Resources in Impala **6**
Estimating memory limits 7
Admission Control and Query Queuing 8
Enabling Admission Control 12
Creating Static Pools 12
Configuring Dynamic Resource Pool 13
Dynamic Resource Pool Settings 13
Admission Control Sample Scenario 15
Cancelling a Query 17
Using HLL Datasketch Algorithms in Impala **18**
Using KLL Datasketch Algorithms in Impala **21**
Managing Metadata in Impala **25**
On-demand Metadata 25
Automatic Invalidation of Metadata Cache 26
Automatic Invalidation/Refresh of Metadata 27
Synchronization between Impala Clusters **29**

READ Support for FULL ACID ORC Tables

FULL ACID v2 transactional tables are readable in Impala without modifying any configurations. You must have Cloudera Runtime 7.2.2 or higher and have connection to Hive Metastore server in order to READ from FULL ACID tables.

There are two types of transactional tables available with Hive ACID.

- INSERT-ONLY
- FULL ACID

Until this release, Impala in Cloudera supported INSERT-ONLY transactional tables allowing both READ and WRITE operations. The latest version of Impala in Cloudera now also supports READ of FULL ACID ORC tables.

By default tables created in Impala are INSERT-ONLY managed tables whereas the default tables in Hive are managed tables that are FULL-ACID and INSERT-ONLY.

Limitations

- Impala cannot CREATE or WRITE to FULL ACID transactional tables yet. You can CREATE and WRITE FULL ACID transactional tables with transaction scope at the row level via HIVE and use Impala to READ these tables.
- Impala does not support ACID v1.

Concepts Used in FULL ACID v2 Tables

Before beginning to use FULL ACID v2 tables you must be aware of these new concepts like transactions, WriteIds, rowIDs, delta delete directories, locks, etc. that are added to FULL ACID tables to achieve ACID semantics.

Write IDs

For every transaction, both read and write, Hive will assign a globally unique ID. For transactional writes like INSERT and DELETE, it will also assign a table-wise unique ID, a write ID. The write ID range will be encoded in the delta and delete directory names. Results of a DML transactional query are allocated to a location under partition/table. This location is derived by Write ID allocated to the transaction. This provides Isolation of DML queries and such queries can run in parallel without interfering with each other.

New Sub-directories

New data files resulting from a DML query are written to a unique location derived from WriteId of the transaction. You can find the results of an INSERT query in delta directories under partition/table location. Depending on the operation type there can be two types of delta directories:

- Delta Directory: This type is created for the results of INSERT statements and is named delta_<writeId>_<writeId> under partition/table location.
- Delete Delta Directory: This delta directory is created for results of DELETE statements and is named delete_delta_<writeId>_<writeId> under partition/table location.

UPDATE operations create both delete and delta directories.

Row IDs

rowId is the auto-generated unique ID within the transaction and bucket. This is added to each row to identify each row in a table. RowID is used during a DELETE operation. When a record is deleted from a table, the rowId of the deleted row will be written to the delete_delta directory. So for all subsequent READ operations all rows will be read except these rows.

Schematic differences between INSERT-ONLY and FULL ACID tables

INSERT-ONLY tables do not have a special schema. They store the data just like plain original files from the non-ACID world. However, their files are organized differently. For every INSERT statement the created files are put into a transactional directory which has transactional information in its name.

Full ACID tables do have a special schema. They have row identifiers to support row-level DELETEs. So a row in Full ACID format looks like this:

```
{
   "operation": 0,
   "originalTransaction": 1,
   "bucket": 536870912,
   "rowId": 0,
   "currentTransaction": 1,
   "row": {"i": 1}
   }
```

- The green columns are the hidden/system ACID columns.
- Field “row” holds the user data.
- operation 0 means INSERT, 1 UPDATE, and 2 DELETE. UPDATE will not appear because of the split-update technique (INSERT + DELETE).
- originalTransaction is the write ID of the INSERT operation that created this row.
- bucket is a 32-bit integer defined by BucketCodec class.
- rowId is the auto-generated unique ID within the transaction and bucket.
- currentTransaction is the current write ID. For INSERT, it is the same as currentTransaction. For DELETE, it is the write ID when this record is first created.
- row contains the actual data. For DELETE, row will be null.

Key Differences between INSERT-ONLY and FULL ACID Tables

Before beginning to use FULL ACID v2 tables you must be aware of the key differences between the INSERT-ONLY and FULL-ACID tables.

This table highlights some of the differences between the INSERT-ONLY and FULL ACID tables.

	INSERT-ONLY	FULL ACID
Schema	There is no special data schema. They store the data just like plain original files from the non-ACID world.	Data is in special format, i.e. there are synthetic columns with transactional information in addition to actual data.
Transactional information	Transactional information is encoded in directory names.	Full ACID tables also use the same directory structure as INSERT-only tables. Transactional information is encoded in the directory names. Directory name and filename are the source of transactional information.
Table properties	'transactional'='true', 'transactional_properties'='insert_only'	'transactional'='true'
Supported operations	INSERT-ONLY tables only support insertion of data. UPDATES and DELETES are not supported. These tables also provide CREATE TABLE, DROP TABLE, TRUNCATE, INSERT, SELECT operations.	FULL ACID ORC tables can be READ using IMPALA. These tables also provide UPDATE and DELETE operations at the row level using HIVE. This is achieved using transactions like Insert-Only Tables along with changes in ORC Reader to support deletes.
WRITE operation	WRITE operations are atomic and the results of the insert operation are not visible to other query operations until the operation is committed.	WRITE operations are atomic - The operation either succeeds completely or fails; it does not result in partial data.

	INSERT-ONLY	FULL ACID
INSERT operation	For every INSERT statement the created files are added to a transactional directory which has transactional information in its name.	INSERT operation is done through HIVE and this statement is executed in a single transaction. This operation creates a delta directory containing information about this transaction and its data.
DELETE operation	N/A	DELETE operation is done through HIVE and this event creates a special “delete delta” directory.
UPDATE operation	N/A	UPDATE operation is done through HIVE. This operation is split into an INSERT and DELETE operation. This operation creates a delta dir followed by a delete dir.
READ operation	READ operations always read a consistent snapshot of the data.	READ operations always read a consistent snapshot of the data.
Supported file format	Supports any file formats.	Supports only ORC.
Compactions	Minor and major compactions are supported.	Minor compactions can be created, which means several delta and delete directories can be compacted into one delta and delete directory. Major compactions are also supported.

Note: Currently, ALTER TABLE statement is not supported on both insert-only and full acid transactional tables.

File structure of FULL ACID transational table

Hive 3 achieves atomicity and isolation of operations on transactional tables by using techniques in write, read, insert, create, delete, and update operations that involve delta files, which can provide query status information and help you troubleshoot query problems.

Compaction of Data in FULL ACID Transactional Table

As administrator, you need to manage compaction of delta files that accumulate during data ingestion. Compaction is a process that performs critical cleanup of files.

Hive creates a set of delta files for each transaction that alters a table or partition and stores them in a separate delta directory. When the number of delta and delete directories in the table grow, the read performance will be impacted, since reading is a process of merging the results of valid transactions. To avoid any compromise on the read performance, occasionally Hive performs compaction, namely minor and major. This process merges these directories while preserving the transaction information.

To initiate automatic compaction, you must enable it using Cloudera Manager. For more information on managing the compaction process, see the link provided under Related Information.

Related Information

Data Compaction

Managing Resources in Impala

Impala includes the features that balance and maximize resources to improve query performance and scalability of your Cloudera cluster.

A typical deployment uses the following resource management features:

- Static service pools

  Use the static service pools to allocate dedicated resources for Impala to manage and prioritize workloads on clusters.

- Admission control

  Within the constraints of the static service pool, you can further subdivide Impala's resources using dynamic resource pools and admission control.

Related Information

Creating Static Pools

Dynamic Resource Pool Settings

Estimating memory limits

Use Cloudera Manager Chart Builder to determine memory limits for resource pools.

To identify what is the appropriate minimum, maximum, and default memory limit for each of the pools, Cloudera recommends that you run real queries on your datasets, and then review the memory usage during peak periods of this test.

After you have run a set of your real queries on your datasets, use Cloudera Manager Chart Builder to view charts that reflect memory usage:

1. On the Cloudera Manager Admin Console home page, from the left navigation tree, select Charts Chart Builder :

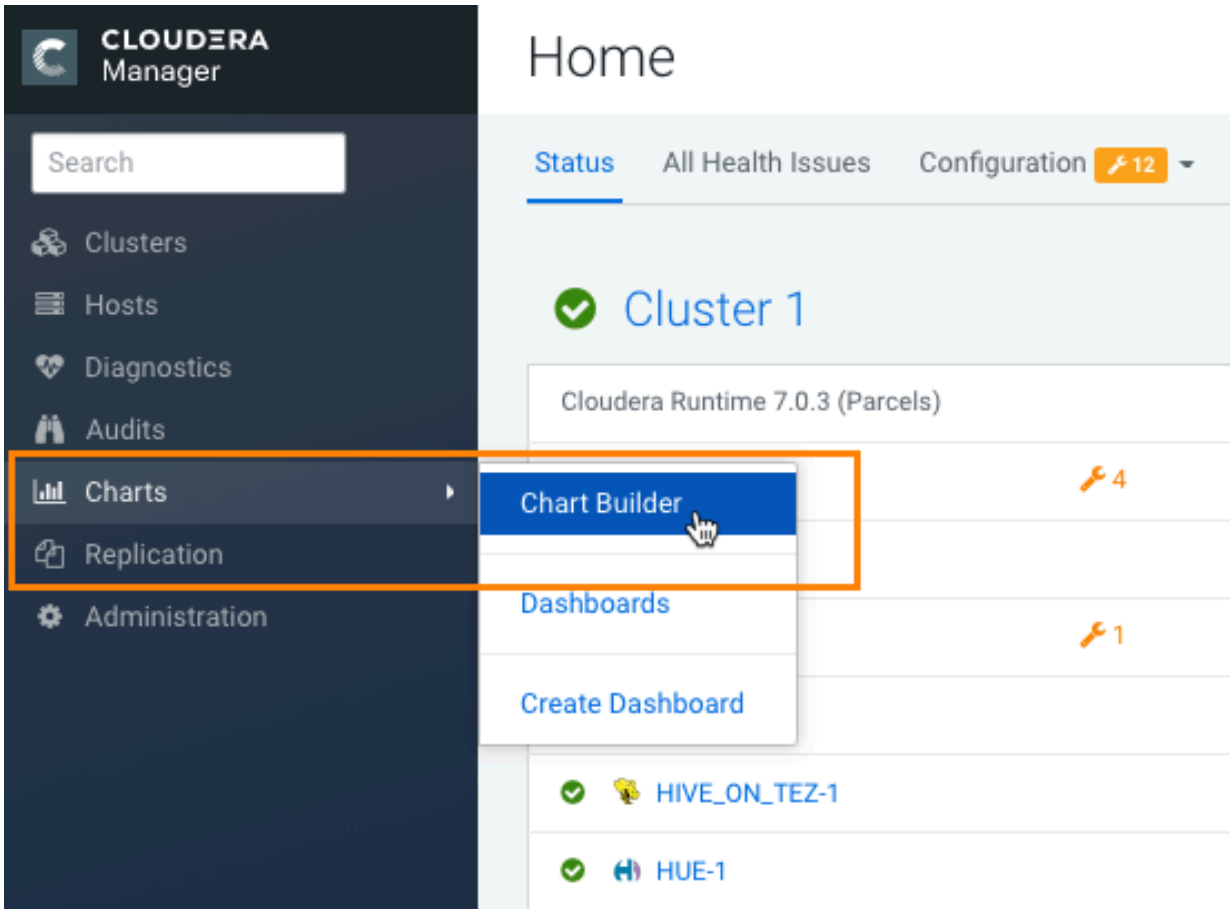

2. On the Chart Builder page, enter the following query, and then click Build Chart:

```
select memory_per_node_peak from IMPALA_QUERIES where
```

```
service_name="impala"
```

Chart Builder then returns a chart that might look similar to the following example:

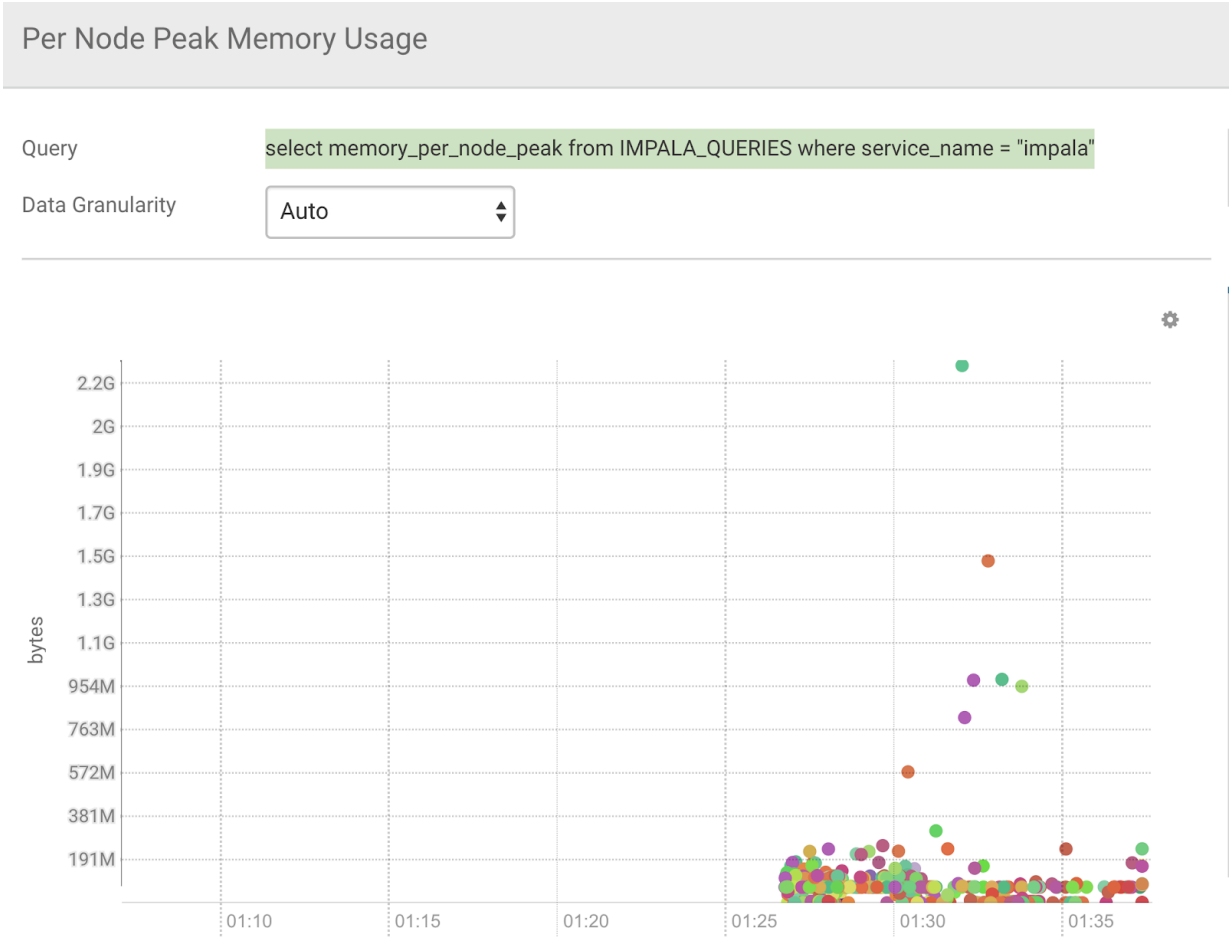

This type of chart can be used to determine the memory limits for a particular pool. Note that most queries use less than 1.2 GB per node. See Admission Control and Query Queuing for more information about Impala admission control.

Admission Control and Query Queuing

Admission control is an Impala feature that imposes limits on concurrent SQL queries to avoid resource usage spikes and out-of-memory conditions on busy Cloudera clusters.

The admission control feature lets you set an upper limit on the number of concurrent Impala queries and on the memory used by those queries. Any additional queries are queued until the earlier ones finish, rather than being cancelled or running slowly and causing contention. As other queries finish, the queued queries are allowed to proceed.

You can specify these limits and thresholds for each pool or globally so that you can balance the resource usage and throughput among steady well-defined workloads, rare resource-intensive queries, and ad-hoc exploratory queries.

In addition to the threshold values for currently executing queries, you can place limits on the maximum number of queries that are queued (waiting) and a limit on the amount of time they might wait before returning with an error. These queue settings let you ensure that queries do not wait indefinitely so that you can detect and correct “starvation” scenarios.

Queries, DML statements, and some DDL statements, including CREATE TABLE AS SELECT and COMPUTE STATS are affected by admission control.

On a busy Cloudera cluster, you might find there is an optimal number of Impala queries that run concurrently. For example, when the I/O capacity is fully utilized by I/O-intensive queries, you might not find any throughput benefit in running more concurrent queries. By allowing some queries to run at full speed while others wait, rather than having all queries contend for resources and run slowly, admission control can result in higher overall throughput.

For another example, consider a memory-bound workload such as many large joins or aggregation queries. Each such query could briefly use many gigabytes of memory to process intermediate results. Because Impala by default cancels queries that exceed the specified memory limit, running multiple large-scale queries at once might require re-running some queries that are cancelled. In this case, admission control improves the reliability and stability of the overall workload by only allowing as many concurrent queries as the overall memory of the cluster can accommodate.

Concurrent Queries and Admission Control

One way to limit resource usage through admission control is to set an upper limit on the number of concurrent queries. This is the initial technique you might use when you do not have extensive information about memory usage for your workload. The settings can be specified separately for each dynamic resource pool.

Max Running Queries

Maximum number of concurrently running queries in this pool. The default value is unlimited. (optional)

The maximum number of queries that can run concurrently in this pool. The default value is unlimited. Any queries for this pool that exceed Max Running Queries are added to the admission control queue until other queries finish. You can use Max Running Queries in the early stages of resource management, when you do not have extensive data about query memory usage, to determine if the cluster performs better overall if throttling is applied to Impala queries.

For a workload with many small queries, you typically specify a high value for this setting, or leave the default setting of “unlimited”. For a workload with expensive queries, where some number of concurrent queries saturate the memory, I/O, CPU, or network capacity of the cluster, set the value low enough that the cluster resources are not overcommitted for Impala.

Once you have enabled memory-based admission control using other pool settings, you can still use Max Running Queries as a safeguard. If queries exceed either the total estimated memory or the maximum number of concurrent queries, they are added to the queue.

If Max Running Queries Multiple is set, the Max Running Queries setting is ignored.

Max Running Queries Multiple

This floating point number is multiplied by the current total number of executors at runtime to give the maximum number of concurrently running queries allowed in the pool. The effect of this setting scales with the number of executors in the resource pool.

This calculation is rounded up to the nearest integer, so the result will always be at least one.

If set to zero or a negative number, the setting is ignored.

Max Queued Queries

Maximum number of queries that can be queued in this pool. The default value is 200. (optional).

If Max Queued Queries Multiple is set, the Max Queued Queries setting is ignored.

Max Queued Queries Multiple

This floating point number is multiplied by the current total number of executors at runtime to give the maximum number of queries that can be queued in the pool. The effect of this setting scales with the number of executors in the resource pool.

This calculation is rounded up to the nearest integer, so the result will always be at least one.

If set to zero or a negative number, the setting is ignored.

Queue Timeout

The amount of time, in milliseconds, that a query waits in the admission control queue for this pool before being canceled. The default value is 60,000 milliseconds.

In the following cases, Queue Timeout is not significant, and you can specify a high value to avoid canceling queries unexpectedly:

- In a low-concurrency workload where few or no queries are queued
- In an environment without a strict SLA, where it does not matter if queries occasionally take longer than usual because they are held in admission control

You might also need to increase the value to use Impala with some business intelligence tools that have their own timeout intervals for queries.

In a high-concurrency workload, especially for queries with a tight SLA, long wait times in admission control can cause a serious problem. For example, if a query needs to run in 10 seconds, and you have tuned it so that it runs in 8 seconds, it violates its SLA if it waits in the admission control queue longer than 2 seconds. In a case like this, set a low timeout value and monitor how many queries are cancelled because of timeouts. This technique helps you to discover capacity, tuning, and scaling problems early, and helps avoid wasting resources by running expensive queries that have already missed their SLA.

If you identify some queries that can have a high timeout value, and others that benefit from a low timeout value, you can create separate pools with different values for this setting.

You can combine these settings with the memory-based approach described below. If either the maximum number of or the expected memory usage of the concurrent queries is exceeded, subsequent queries are queued until the concurrent workload falls below the threshold again.

Memory Limits and Admission Control

Each dynamic resource pool can have an upper limit on the cluster-wide memory used by queries executing in that pool.

Use the following settings to manage memory-based admission control.

Max Memory

The maximum amount of aggregate memory available across the cluster to all queries executing in this pool. This should be a portion of the aggregate configured memory for Impala daemons, which will be shown in the settings dialog next to this option for convenience. Setting this to a non-zero value enables memory based admission control.

Impala determines the expected maximum memory used by all queries in the pool and holds back any further queries that would result in Max Memory being exceeded.

If you specify Max Memory, you should specify the amount of memory to allocate to each query in this pool. You can do this in two ways:

- By setting Maximum Query Memory Limit and Minimum Query Memory Limit. This is preferred as it gives Impala flexibility to set aside more memory to queries that are expected to be memory-hungry.
- By setting Default Query Memory Limit to the exact amount of memory that Impala should set aside for queries in that pool.

Note that in the following cases, Impala will rely entirely on memory estimates to determine how much memory to set aside for each query. This is not recommended because it can result in queries not running or being starved for memory if the estimates are inaccurate. And it can affect other queries running on the same node.

- Max Memory, Maximum Query Memory Limit, and Minimum Query Memory Limit are not set, and the MEM_LIMIT query option is not set for the query.

- Default Query Memory Limit is set to 0, and the MEM_LIMIT query option is not set for the query.

If Max Memory Multiple is set, the Max Memory setting is ignored.

Max Memory Multiple

This number of bytes is multiplied by the current total number of executors at runtime to give the maximum memory available across the cluster for the pool. The effect of this setting scales with the number of executors in the resource pool.

If set to zero or a negative number, the setting is ignored.

Minimum Query Memory Limit and Maximum Query Memory Limit

These two options determine the minimum and maximum per-host memory limit that will be chosen by Impala Admission control for queries in this resource pool. If set, Impala admission control will choose a memory limit between the minimum and maximum values based on the per-host memory estimate for the query. The memory limit chosen determines the amount of memory that Impala admission control will set aside for this query on each host that the query is running on. The aggregate memory across all of the hosts that the query is running on is counted against the pool's Max Memory.

Minimum Query Memory Limit must be less than or equal to Maximum Query Memory Limit and Max Memory.

A user can override Impala's choice of memory limit by setting the MEM_LIMIT query option. If the Clamp MEM_LIMIT Query Option setting is set to TRUE and the user sets MEM_LIMIT to a value that is outside of the range specified by these two options, then the effective memory limit will be either the minimum or maximum, depending on whether MEM_LIMIT is lower than or higher the range.

For example, assume a resource pool with the following parameters set:

- Minimum Query Memory Limit = 2GB
- Maximum Query Memory Limit = 10GB

If a user tries to submit a query with the MEM_LIMIT query option set to 14 GB, the following would happen:

- If Clamp MEM_LIMIT Query Option = true, admission controller would override MEM_LIMIT with 10 GB and attempt admission using that value.
- If Clamp MEM_LIMIT Query Option = false, the admission controller will retain the MEM_LIMIT of 14 GB set by the user and will attempt admission using the value.

Clamp MEM_LIMIT Query Option

If this field is not selected, the MEM_LIMIT query option will not be bounded by the Maximum Query Memory Limit and the Minimum Query Memory Limit values specified for this resource pool. By default, this field is selected. The field is disabled if both Minimum Query Memory Limit and Maximum Query Memory Limit are not set.

You can combine the memory-based settings with the upper limit on concurrent queries. If either the maximum number of or the expected memory usage of the concurrent queries is exceeded, subsequent queries are queued until the concurrent workload falls below the threshold again.

Monitoring Admission Control

To see how admission control works for particular queries, examine the profile output or the summary output for the query.

Profile

The information is available through the PROFILE statement in `impala-shell` immediately after running a query in the shell, on the queries page of the Impala debug web UI, or in the Impala log file (basic information at log level 1, more detailed information at log level 2).

The profile output contains details about the admission decision, such as whether the query was queued or not and which resource pool it was assigned to. It also includes the estimated and actual memory usage for the query, so you can fine-tune the configuration for the memory limits of the resource pools.

Summary

The summary output includes the queuing status consisting of whether the query was queued and what was the latest queuing reason.

The information is available in `impala-shell` when the LIVE_PROGRESS or LIVE_SUMMARY query option is set to TRUE.

You can also start an impala-shell session with the --live_progress or --live_summary flags to monitor all queries in that impala-shell session.

Enabling Admission Control

Enable admission control on all production clusters to alleviate possible capacity issues. The capacity issues could be because of a high volume of concurrent queries, because of heavy-duty join and aggregation queries that require large amounts of memory, or because Impala is being used alongside other Hadoop data management components and the resource usage of Impala must be constrained to work well.

About this task

Procedure

1. Navigate to ClustersImpala.
2. In the Configuration tab, navigate to CategoryAdmission Control.
3. Select or clear both the Enable Impala Admission Control and the Enable Dynamic Resource Pools.
4. Enter a Reason for change, and then click Save Changes to commit the changes.
5. Restart the Impala service.
6. After completing this task, for further configuration settings, customize the configuration settings for the dynamic resource pools.

Creating Static Pools

To manage and prioritize workloads on clusters, use the static service pools to allocate dedicated resources for Impala for predictable resource availability. When static service pools are used, Cloudera Manager creates a cgroup in which Impala runs. This cgroup limits memory, CPU and Disk I/O according to the static partitioning policy.

About this task

Create static resource pools for the services running in your Cloudera cluster.

Procedure

1. In Cloudera Manager, navigate to Clusters Static service pools .

2. In the Configuration tab, allocate a portion of resource to each component.
   - HDFS always needs to have a minimum of 5-10% of the resources.
   - Generally, YARN and Impala split the rest of the resources.
     - For mostly batch workloads, you might allocate YARN 60%, Impala 30%, and HDFS 10%.
     - For mostly ad-hoc query workloads, you might allocate Impala 60%, YARN 30%, and HDFS 10%.
3. Click Continue.
4. Review the changes and click Continue.
5. Click Restart Now.

Configuring Dynamic Resource Pool

Admission control and dynamic resource pools are enabled by default. However, until you configure the settings for the dynamic resource pools, the admission control feature is effectively not enabled.

About this task

There is always a resource pool designated as root.default. By default, all Impala queries run in this pool when the dynamic resource pool feature is enabled for Impala. You create additional pools when your workload includes identifiable groups of queries (such as from a particular application, or a particular group within your organization) that have their own requirements for concurrency, memory use, or service level agreement (SLA). Each pool has its own settings related to memory, number of queries, and timeout interval.

Procedure

1. In Cloudera Manager, navigate to Clusters Impala Admission Control Configuration . If the cluster has an Impala service, the Resource Pools tab displays under the Impala Admission Control tab.
2. In the Impala Admission Control tab, click Create Resource Pool.
3. Specify a name and resource limits for the pool:
   - In the Resource Pool Name field, type a unique name containing only alphanumeric characters.
   - Optionally, in the Submission Access Control tab, specify which users and groups can submit queries. By default, anyone can submit queries. To restrict this permission, select the Allow these users and groups option and provide a comma-delimited list of users and groups in the Users and Groups fields respectively.
4. Click Create.
5. Click Refresh Dynamic Resource Pools.

Dynamic Resource Pool Settings

Use the following settings to configure your dynamic resource pools for Impala.

Max Memory

Maximum amount of aggregate memory available across the cluster to all queries executing in this pool. This should be a portion of the aggregate configured memory for Impala daemons, which will be shown in the settings dialog next to this option for convenience. Setting this to a non-zero value enables memory based admission control.

Impala determines the expected maximum memory used by all queries in the pool and holds back any further queries that would result in Max Memory being exceeded.

If you specify Max Memory, you should specify the amount of memory to allocate to each query in this pool. You can do this in two ways:

- By setting Maximum Query Memory Limit and Minimum Query Memory Limit. Setting them gives Impala flexibility to set aside more memory to queries that are expected to be memory-hungry.

- By setting Default Query Memory Limit to the exact amount of memory that Impala should set aside for queries in that pool.

Note that if you do not set any of the above options, or set Default Query Memory Limit to 0, Impala will rely entirely on memory estimates to determine how much memory to set aside for each query. This is not recommended because it can result in queries not running or being starved for memory if the estimates are inaccurate.

For example, consider the following scenario:

- The cluster is running impalad daemons on five hosts.
- A dynamic resource pool has Max Memory set to 100 GB.
- The Maximum Query Memory Limit for the pool is 10 GB and Minimum Query Memory Limit is 2 GB. Therefore, any query running in this pool could use up to 50 GB of memory (Maximum Query Memory Limit * number of Impala nodes).
- Impala will run varying numbers of queries concurrently because queries may be given memory limits anywhere between 2 GB and 10 GB, depending on the estimated memory requirements. For example, Impala may run up to 10 small queries with 2 GB memory limits or two large queries with 10 GB memory limits because that is what will fit in the 100 GB cluster-wide limit when executing on five hosts.
- The executing queries may use less memory than the per-host memory limit or the Max Memory cluster-wide limit if they do not need that much memory. In general this is not a problem so long as you are able to run enough queries concurrently to meet your needs.

Minimum Query Memory Limit and Maximum Query Memory Limit

These two options determine the minimum and maximum per-host memory limit that will be chosen by Impala Admission control for queries in this resource pool. If set, Impala admission control will choose a memory limit between the minimum and maximum value based on the per-host memory estimate for the query. The memory limit chosen determines the amount of memory that Impala admission control will set aside for this query on each host that the query is running on. The aggregate memory across all of the hosts that the query is running on is counted against the pool's Max Memory.

Minimum Query Memory Limit must be less than or equal to Maximum Query Memory Limit and Max Memory.

You can override Impala's choice of memory limit by setting the MEM_LIMIT query option. If the Clamp MEM_LIMIT Query Option is selected and the user sets MEM_LIMIT to a value that is outside of the range specified by these two options, then the effective memory limit will be either the minimum or maximum, depending on whether MEM_LIMIT is lower than or higher than the range.

Max Running Queries

Maximum number of concurrently running queries in this pool. The default value is unlimited.

The maximum number of queries that can run concurrently in this pool. The default value is unlimited. Any queries for this pool that exceed Max Running Queries are added to the admission control queue until other queries finish. You can use Max Running Queries in the early stages of resource management, when you do not have extensive data about query memory usage, to determine if the cluster performs better overall if throttling is applied to Impala queries.

For a workload with many small queries, you typically specify a high value for this setting, or leave the default setting of "unlimited". For a workload with expensive queries, where some number of concurrent queries saturate the memory, I/O, CPU, or network capacity of the cluster, set the value low enough that the cluster resources are not overcommitted for Impala.

Once you have enabled memory-based admission control using other pool settings, you can still use Max Running Queries as a safeguard. If queries exceed either the total estimated memory or the maximum number of concurrent queries, they are added to the queue.

Max Queued Queries

Maximum number of queries that can be queued in this pool. The default value is 200. (optional)

Queue Timeout

The amount of time, in milliseconds, that a query waits in the admission control queue for this pool before being canceled. The default value is 60,000 milliseconds.

It the following cases, Queue Timeout is not significant, and you can specify a high value to avoid canceling queries unexpectedly:

- In a low-concurrency workload where few or no queries are queued
- In an environment without a strict SLA, where it does not matter if queries occasionally take longer than usual because they are held in admission control

You might also need to increase the value to use Impala with some business intelligence tools that have their own timeout intervals for queries.

In a high-concurrency workload, especially for queries with a tight SLA, long wait times in admission control can cause a serious problem. For example, if a query needs to run in 10 seconds, and you have tuned it so that it runs in 8 seconds, it violates its SLA if it waits in the admission control queue longer than 2 seconds. In a case like this, set a low timeout value and monitor how many queries are cancelled because of timeouts. This technique helps you to discover capacity, tuning, and scaling problems early, and helps avoid wasting resources by running expensive queries that have already missed their SLA.

If you identify some queries that can have a high timeout value, and others that benefit from a low timeout value, you can create separate pools with different values for this setting.

Clamp MEM_LIMIT Query Option

If this field is not selected, the MEM_LIMIT query option will not be bounded by the Maximum Query Memory Limit and the Minimum Query Memory Limit values specified for this resource pool. By default, this field is selected. The field is disabled if both Minimum Query Memory Limit and Maximum Query Memory Limit are not set.

Admission Control Sample Scenario

You can learn about the factors you must consider when allocating Impala's resources and the process you need to follow to set up admission control for the selected workload.

Anne Chang is administrator for an enterprise data hub that runs a number of workloads, including Impala.

Anne has a 20-node cluster that uses Cloudera Manager static partitioning. Because of the heavy Impala workload, Anne needs to make sure Impala gets enough resources. While the best configuration values might not be known in advance, she decides to start by allocating 50% of resources to Impala. Each node has 128 GiB dedicated to each impalad. Impala has 2560 GiB in aggregate that can be shared across the resource pools she creates.

Next, Anne studies the workload in more detail. After some research, she might choose to revisit these initial values for static partitioning.

To figure out how to further allocate Impala's resources, Anne needs to consider the workloads and users, and determine their requirements. There are a few main sources of Impala queries:

- Large reporting queries executed by an external process/tool. These are critical business intelligence queries that are important for business decisions. It is important that they get the resources they need to run. There typically are not many of these queries at a given time.
- Frequent, small queries generated by a web UI. These queries scan a limited amount of data and do not require expensive joins or aggregations. These queries are important, but not as critical, perhaps the client tries resending the query or the end user refreshes the page.
- Occasionally, expert users might run ad-hoc queries. The queries can vary significantly in their resource requirements. While Anne wants a good experience for these users, it is hard to control what they do (for example,

submitting inefficient or incorrect queries by mistake). Anne restricts these queries by default and tells users to reach out to her if they need more resources.

To set up admission control for this workload, Anne first runs the workloads independently, so that she can observe the workload's resource usage in Cloudera Manager. If they could not easily be run manually, but had been run in the past, Anne uses the history information from Cloudera Manager. It can be helpful to use other search criteria (for example, user) to isolate queries by workload. Anne uses the Cloudera Manager chart for Per-Node Peak Memory usage to identify the maximum memory requirements for the queries.

From this data, Anne observes the following about the queries in the groups above:

- Large reporting queries use up to 32 GiB per node. There are typically 1 or 2 queries running at a time. On one occasion, she observed that 3 of these queries were running concurrently. Queries can take 3 minutes to complete.
- Web UI-generated queries use between 100 MiB per node to usually less than 4 GiB per node of memory, but occasionally as much as 10 GiB per node. Queries take, on average, 5 seconds, and there can be as many as 140 incoming queries per minute.
- Anne has little data on ad hoc queries, but some are trivial (approximately 100 MiB per node), others join several tables (requiring a few GiB per node), and one user submitted a huge cross join of all tables that used all system resources (that was likely a mistake).

Based on these observations, Anne creates the admission control configuration with the following pools:

XL_Reporting

Property	Value
Max Memory	1280 GiB
Maximum Query Memory Limit	32 GiB
Minimum Query Memory Limit	32 GiB
Max Running Queries	2
Queue Timeout	5 minutes

This pool is for large reporting queries. To support running 2 queries at a time, the pool memory resources are set to 1280 GiB (aggregate cluster memory). This is for 2 queries, each with 32 GiB per node, across 20 nodes. Anne sets the pool's Maximum Query Memory Limit to 32 GiB so that no query uses more than 32 GiB on any given node. She sets Max Running Queries to 2 (though it is not necessary she do so). She increases the pool's queue timeout to 5 minutes in case a third query comes in and has to wait. She does not expect more than 3 concurrent queries, and she does not want them to wait that long anyway, so she does not increase the queue timeout. If the workload increases in the future, she might choose to adjust the configuration or buy more hardware.

HighThroughput_UI

Property	Value
Max Memory	960 GiB (inferred)
Maximum Query Memory Limit	4 GiB
Minimum Query Memory Limit	2 GiB
Max Running Queries	12
Queue Timeout	5 minutes

This pool is used for the small, high throughput queries generated by the web tool. Anne sets the Maximum Query Memory Limit to 4 GiB per node, and sets Max Running Queries to 12. This implies a maximum amount of memory per node used by the queries in this pool: 48 GiB per node (12 queries * 4 GiB per node memory limit).

Notice that Anne does not set the pool memory resources, but does set the pool's Maximum Query Memory Limit. This is intentional: admission control processes queries faster when a pool uses the Max Running Queries limit instead of the peak memory resources.

This should be enough memory for most queries, since only a few go over 4 GiB per node. For those that do require more memory, they can probably still complete with less memory (spilling if necessary). If, on occasion, a query cannot run with this much memory and it fails, Anne might reconsider this configuration later, or perhaps she does not need to worry about a few rare failures from this web UI.

With regard to throughput, since these queries take around 5 seconds and she is allowing 12 concurrent queries, the pool should be able to handle approximately 144 queries per minute, which is enough for the peak maximum expected of 140 queries per minute. In case there is a large burst of queries, Anne wants them to queue. The default maximum size of the queue is already 200, which should be more than large enough. Anne does not need to change it.

Default

Property	Value
Max Memory	320 GiB
Maximum Query Memory Limit	4 GiB
Minimum Query Memory Limit	2 GiB
Max Running Queries	Unlimited
Queue Timeout	60 Seconds

The default pool (which already exists) is a catch all for ad-hoc queries. Anne wants to use the remaining memory not used by the first two pools, 16 GiB per node (XL_Reporting uses 64 GiB per node, High_Throughput_UI uses 48 GiB per node). For the other pools to get the resources they expect, she must still set the Max Memory resources and the Maximum Query Memory Limit. She sets the Max Memory resources to 320 GiB (16 * 20). She sets the Maximum Query Memory Limit to 4 GiB per node for now. That is somewhat arbitrary, but satisfies some of the ad hoc queries she observed. If someone writes a bad query by mistake, she does not actually want it using all the system resources. If a user has a large query to submit, an expert user can override the Maximum Query Memory Limit (up to 16 GiB per node, since that is bound by the pool Max Memory resources). If that is still insufficient for this user's workload, the user should work with Anne to adjust the settings and perhaps create a dedicated pool for the workload.

Cancelling a Query

Various client applications let you interactively cancel queries submitted or monitored through those applications.

Setting a Time Limit on Query Execution

An Impala administrator can set a default value of the EXEC_TIME_LIMIT_S query option for a resource pool. If a user accidentally runs a large query that executes for longer than the limit, it will be automatically terminated after the time limit expires to free up resources.

You can override the default value per query or per session if you do not want to apply the default EXEC_TIME_LIMIT_S value to a specific query or a session.

Interactively Cancelling a Query

- In the Impala Web UI for the `impalad` host (on port 25000 by default), cancel a query: In the /queriestab, click Cancel for a query in the queries in flight list.
- In `impala-shell`, press ^C
- In Hue, click Cancel from the Watch page.

You can manually cancel a query in the Impala Web UI for the `impalad` host (on port 25000 by default):

Using HLL Datasketch Algorithms in Impala

You can use Datasketch algorithms (HLL) for queries that take too long to calculate exact results due to very large data sets (e.g. number of distinct values).

You may use data sketches (HLL algorithms) to generate approximate results that are much faster to retrieve. HLL is an algorithm that gives approximate answers for computing the number of distinct values in a column. The value returned by this algorithm is similar to the result of COUNT(DISTINCT col) and the NDV function integrated with Impala. However, HLL algorithm is much faster than COUNT(DISTINCT col) and the NDV function and is less memory-intensive for columns with high cardinality.

These HLL algorithms create “sketches”, which are data structures containing an approximate calculation of some facet of the data and which are compatible regardless of which system reads or writes them. For e.g., This particular algorithm works in two phases. First it creates a sketch from the dataset provided for the algorithm and then as the second step in the process you can read the estimate from the sketch. Using this approach it’s also possible to create granular sketches for a specific dataset (e.g. one sketch per partition) and later on use them to provide answers for different count(distinct) queries without reading the actual data as they can also be merged together. Using HLL you can create sketches by Hive and read it with Impala for getting the estimate and vice versa.

Usage Notes

Query results produced by the HLL algorithm are approximate but within well defined error bounds. Because the number of distinct values in a column returned by this HLL algorithm is an estimate, it might not reflect the precise number of different values in the column, especially if the cardinality is very high.

Data Types

The following is a list of currently supported data types of the datasets that these HLL functions are compatible with.

Supported:

1. Numeric Types:

   TINYINT, INT, BIGINT, FLOAT, DOUBLE
2. String types:

   STRING, CHAR, VARCHAR
3. Other types:

   BINARY

Unsupported:

1. Complex types
2. Some of the scalar types: DECIMAL, TIMESTAMP, BOOLEAN, SMALLINT, DATE

File Formats

The currently supported file formats are ORC and Parquet. Since Impala does not yet have read support for materialized views, it is required for Hive to write the sketches into regular tables (i.e. not into a materialized view), using the BINARY type. Then Impala can be used to read these sketches and calculate the estimate as Hive does.

Limitations:

You can write HLL sketches into a text table, however it may not work as the serialized sketches can contain characters that are special for text format.

Internal Configuration

The configuration is hard coded and the level of compression is HLL_4 and the number of buckets in the HLL array is k=12 (2 power of 12).

Using HLL Sketches

This datasketch algorithm works in two phases.

- First it creates a sketch from the dataset provided for the algorithm.
- As the second step in the process the function provides an estimate from the sketch.

The following example shows how to create a sketch and to create an estimate from the sketch.

Creating a sketch

This query creates a sketch using the ds_hll_sketch function from a column date_string_col containing the supported data type.

```
Select ds_hll_sketch(date_string_col) from tableName;
```

This query receives a primitive expression and returns a serialized HLL sketch that is not in a readble format.

Estimating the number of distinct values in a column

The following query returns a cardinality estimate (similarly to ndv()) for that particular column (date_string_col) by using the HLL function ds_hll_estimate.

```
Select ds_hll_estimate(ds_hll_sketch(date_string_col)) from tableName;
```

This query receives a primitive expression (column name) and then creates a sketch from the column, and then passes the sketch to the outer function that gives a cardinality estimate.

```
+-------------------------------------------------+
| ds_hll_estimate(ds_hll_sketch(date_string_col))|
+-------------------------------------------------+
| 729                                             |
+-------------------------------------------------+
Fetched 1 row(s) in 0.25s
```

The above result is similar to the results received using the count(distinct col) or the ndv() approach.

```
Select count(distinct date_string_col) from tableName;
+-------------------------------------------------+
| count (distinct date_string_col)                |
+-------------------------------------------------+
| 730                                             |
+-------------------------------------------------+
```

```
Select ndv(distinct date_string_col) from tableName;
+-------------------------------------------------+
| ndv (date_string_col)                           |
+-------------------------------------------------+
| 736                                             |
+-------------------------------------------------+
```

Inserting the derived sketch into a table

You can create a sketch and save it for later use. If you save the sketches to a table or view with the same granularity (e.g. one sketch for each partition in a table) then later you can simply read the sketch and get the estimate almost for free as the real cost here is just the sketch creation.

The following example shows the creation of a sketch for the column (date_string_col) from the table “tableName” and saving the created sketch into another table called sketch_table and later reading the sketch and its estimate from the saved table called sketch_table.

```
    select year, month, ds_hll_estimate(ds_hll_sketch(date_string_col)) from
 tableName group by year, month;
    +-------------------------------------------------------------+
    | year | month | ds_hll_estimate(ds_hll_sketch(date_string_col))|
    +------+-------+-------------------------------------------------+
    | 2010 | 5     | 31                                              |
    | 2010 | 11    | 30                                              |
    | 2010 | 6     | 30                                              |
    | 2010 | 2     | 28                                              |
    | 2010 | 10    | 31                                              |
    | 2009 | 11    | 30                                              |
    | 2009 | 7     | 31                                              |
    | 2009 | 10    | 31                                              |
    | 2010 | 11    | 31                                              |
    | 2009 | 7     | 31                                              |
    | 2010 | 10    | 31                                              |
    | 2010 | 8     | 30                                              |
    | 2010 | 5     | 31                                              |
    | 2009 | 7     | 30                                              |
    | 2010 | 4     | 31                                              |
    | 2010 | 12    | 30                                              |
    | 2009 | 9     | 31                                              |
    | 2009 | 8     | 30                                              |
    | 2010 | 6     | 28                                              |
    | 2010 | 3     | 31                                              |
    | 2010 | 4     | 31                                              |
    | 2010 | 2     | 30                                              |
    | 2009 | 1     | 31                                              |
    | 2009 | 12    | 30                                              |
    +-------------------------------------------------------------+
```

Insert the created sketch into another table:

```
    insert into sketch_table select year, month, ds_hll_sketch(date_string_
 col) from tableName group by year, month;
```

Verify the number of rows modified using count function in the table:

```
    select count(1) from sketch_table;
```

You can also verify it by getting the estimates from the sketches stored in the table using the following query:

```
    select year, month, ds_hll_estimate(sketch_col) from sketch_table;
```

Combining Sketches

You can combine sketches to allow complex queries to be computed from a few number of simple sketches.

```
select ds_hll_estimate(ds_hll_union(sketch_col)) from sketch_table;
```

Non-deterministic vs Deterministic Results

Based on the characteristics of a query, DataSketches HLL might return results in a non-deterministic manner. During the cache populating phase it returns exact results but in the approximating phase the algorithm can be sensitive to the order of data it receives. As a result you might see the results differ in consecutive runs but all of them will be within the error range of the algorithm.

Due to the characteristics of the algorithm, and to how it was implemented in Impala, the results might be non-deterministic when the query is executed on one node and the input data set is big enough that the algorithm starts approximating instead of giving exact results.

Inter-opreability between Hive and Impala

From the above examples you will notice that the NDV function is much faster than the Datasketches. But NDV is used only by Impala and not by Hive whereas the Datasketches are implemented in Hive as well as in Impala. In order to maintain the inter-operability between Hive and Impala we recommend to use Datasketches to calculate the estimate. A sketch created by Hive using Datasketches can be fed to Impala to produce an estimate.

Using KLL Datasketch Algorithms in Impala

You can use the stochastic streaming algorithm (KLL) that uses the percentile/quantile functions to statistically analyze the approximate distribution of comparable data from a very large stream.

Use the ds_kll_quantiles() or its inverse functions the Probability Mass Function ds_kll_PMF() and the Cumulative Distribution Function ds_kll_CDF() to obtain the analysis. Query results produced by this KLL algorithm are approximate but within well defined error bounds.

Data Types and File Formats

KLL function currently supports only floats. If you sketch an int data the KLL function converts it into float.

You can sketch data from any file formats. However it is recommended to save the sketches into parquet tables. It is not recommended to save the sketches to text format.

Using KLL Sketches

This datasketch algorithm works in two phases.

- First it creates a sketch from the dataset provided for the algorithm.
- As the second step in the process you can read the estimate from the sketch.

Using this approach you can create granular sketches for a specific dataset (e.g. one sketch per partition) and later use them to provide answers for different quantile queries without reading the actual data as they can also be merged together. Using KLL you can create sketches by Hive and read it with Impala for getting the estimate and vice versa.

Creating a sketch

The following example shows how to create a sketch using the ds_kll_sketch function from a column (float_col) containing the supported data type. This created sketch is a data structure containing an approximate calculation of some facet of the data and which are compatible regardless of which system reads or writes them.

```
select ds_kll_sketch(float_col) from tableName;
```

where ds_kll_sketch() is an aggregate function that receives a float parameter (e.g. a float column of a table) and returns a serialized Apache DataSketches KLL sketch of the input data set wrapped into STRING type. Since the serialized data might contain unsupported characters this query may error out. So another approach to this method is to insert the derived sketch into a table or a view and later use for quantile approximations.

Inserting the derived sketch into a table

If you save the sketches to a table or view with the same granularity (e.g. one sketch for each partition in a table) then later you can simply read the sketch for quantile approximation.

```
insert into table_name select ds_kll_sketch(float_col) from tableName;
```

Debugging the created sketch

To troubleshoot the sketch created by the ds_kll_sketch function, use ds_kll_stringify on the sketch. ds_kll_stringify () receives a string that is a serialized Apache DataSketches sketch and returns its stringified format by invoking the related function on the sketch's interface.

```
select ds_kll_stringify(ds_kll_sketch(float_col)) from tableName;
```

```
+-------------------------------------------+
| ds_kll_stringify(ds_kll_sketch(float_col))|
+-------------------------------------------+
| ### KLL sketch summary:                   |
|    K                     : 200            |
|    min K                 : 200            |
|    M                     : 8              |
|    N                     : 100            |
|    Epsilon               : 1.33%          |
|    Epsilon PMF           : 1.65%          |
|    Empty                 : false          |
|    Estimation mode       : false          |
|    Levels                : 1              |
|    Sorted                : false          |
|    Capacity items        : 200            |
|    Retained items        : 100            |
|    Storage bytes         : 432            |
|    Min value             : 0              |
|    Max value             : 9              |
| ### End sketch summary                    |
+-------------------------------------------+
```

Determining the number of datasets sketched

To determine the number of records sketched into this sketch, use ds_kll_n function.

```
select ds_kll_n(ds_kll_sketch(float_col)) from tableName;
```

```
+-------------------------------------+
| ds_kll_n(ds_kll_sketch(float_col)) |
+-------------------------------------+
| 100                                 |
```

```
+-----------------------------------+
```

Calculate Quantile

This function ds_kll_quantile() function receives two parameters, a STRING parameter that contains a serialized KLL sketch and a DOUBLE (0.5) that represents the rank of the quantile in the range of [0,1]. E.g. rank=0.5 means the approximate median of all the sketched data.

```
+-------------------------------------------------+
| ds_kll_quantile(ds_kll_sketch(float_col), 0.5) |
+-------------------------------------------------+
| 4.0                                             |
+-------------------------------------------------+
```

This query returns a data (4.0) that is bigger than the 50% of the data.

Calculating quantiles

To calculate the quantiles for multiple rank parameters, use the function ds_kll_quantiles_as_string () that is very similar to ds_kll_quantile() but this function receives any number of rank parameters and returns a comma separated string that holds the results for all of the given ranks.

```
   select ds_kll_quantiles_as_string(ds_kll_sketch(float_col), 0.5, 0.6, 1)
from tableName;
```

```
+-------------------------------------------------------------------+
| ds_kll_quantiles_as_string(ds_kll_sketch(float_col), 0.5, 0.6, 1) |
+-------------------------------------------------------------------+
| 4, 5, 9                                                           |
+-------------------------------------------------------------------+
```

Calculate Rank

This rank function ds_kll_rank() receives two parameters, a STRING that represents a serialized DataSketches KLL sketch and a float to provide a probing value in the sketch.

```
select ds_kll_rank(ds_kll_sketch(float_col), 4) from tableName;
```

```
+-------------------------------------------------+
| ds_kll_rank(ds_kll_sketch(float_col), 4)        |
+-------------------------------------------------+
| 0.48                                            |
+-------------------------------------------------+
```

This query returns a DOUBLE that is the rank of the given probing value in the range of [0,1]. E.g. a return value of 0.48 means that the probing value given as parameter is greater than 48% of all the values in the sketch.

Note: This is only an approximate calculation.

Calculate Probability Mass Function (PMF)

This Probabilistic Mass Function (PMF) ds_kll_pmf_as_string()receives a serialized KLL sketch and one or more float values to represent ranges in the sketched values. In the following example, the float values [1, 3, 4, 8] represent the following ranges: (-inf, 1), [1, 3), [3, 4), [4, 8) [8, +inf)

Note: The input values for the ranges have to be unique and monotonically increasing.

```
select ds_kll_pmf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) from table
Name
```

```
+-------------------------------------------------------------+
| ds_kll_pmf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) |
+-------------------------------------------------------------+
| 0.12, 0.24, 0.12, 0.36, 0.16                                |
+-------------------------------------------------------------+
```

This query returns a comma separated string where each value in the string is a number in the range of [0,1] and shows what percentage of the data is in the particular ranges.

Calculate Cumulative Distribution Function (CDF)

This Cumulative Distribution Function (CDF) ds_kll_cmf_as_string()receives a serialized KLL sketch and one or more float values to represent ranges in the sketched values. In the following example, the float values [1, 3, 4, 8] represents the following ranges: (-inf, 1), (-inf, 3), (-inf, 4), (-inf, 8), (-inf, +inf)

Note: The input values for the ranges have to be unique and monotonically increasing.

```
select ds_kll_cdf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) from table
Name;
```

```
+-------------------------------------------------------------+
| ds_kll_cdf_as_string(ds_kll_sketch(float_col), 1, 3, 4, 8) |
+-------------------------------------------------------------+
| 0.12, 0.36, 0.48, 0.84, 1                                   |
+-------------------------------------------------------------+
```

This query returns a comma separated string where each value in the string is a number in the range of [0,1] and shows what percentage of the data is in the particular ranges.

Calculate the Union

To take advantage of the UNION function, create granular sketches for a specific dataset (one sketch per partition), write these sketches to a separate table and based on the partition you are interested in, you can UNION the relevant sketches together to get an estimate.

```
insert into sketch_tbl select ds_kll_sketch(float_col) from tableName;
```

```
select ds_kll_quantile(ds_kll_union(sketch_col), 0.5) from sketch_tbl
    where partition_col=1 OR partition_col=5;
```

This function ds_kll_union() receives a set of serialized Apache DataSketches KLL sketches produced by ds_kll_sketch() and merges them into a single sketch.

Managing Metadata in Impala

This section describes various knobs you can use to control how Impala manages its metadata in order to improve performance and scalability.

On-demand Metadata

With the on-demand metadata feature, the Impala coordinators pull metadata as needed from catalogd and cache it locally. The cached metadata gets evicted automatically under memory pressure.

The granularity of on-demand metadata fetches is at the partition level between the coordinator and catalogd. Common use cases like add/drop partitions do not trigger unnecessary serialization/deserialization of large metadata.

The feature can be used in either of the following modes.

Metadata on-demand mode

In this mode, all coordinators use the metadata on-demand.

Set the following on catalogd:

```
--catalog_topic_mode=minimal
```

Set the following on all impalad coordinators:

```
--use_local_catalog=true
```

Mixed mode

In this mode, only some coordinators are enabled to use the metadata on-demand.

We recommend that you use the mixed mode only for testing local catalog's impact on heap usage.

Set the following on catalogd:

```
--catalog_topic_mode=mixed
```

Set the following on impalad coordinators with metdadata on-demand:

```
--use_local_catalog=true
```

Flags related to use_local_catalog

When use_local_catalog is enabled or set to True on the impalad coordinators the following list of flags configure various parameters as described below. It is not recommended to change the default values on these flags.

- The flag local_catalog_cache_mb (defaults to -1) configures the size of the catalog cache within each coordinator. With the default set to -1, the cache is auto-configured to 60% of the configured Java heap size. Note that the Java heap size is distinct from and typically smaller than the overall Impala memory limit.
- The flag local_catalog_cache_expiration_s (defaults to 3600) configures the expiration time of the catalog cache within each impalad. Even if the configured cache capacity has not been reached, items are removed from the cache if they have not been accessed in the defined amount of time.
- The flag local_catalog_max_fetch_retries (defaults to 40) configures the maximum number of retries needed for queries to fetch a metadata object from the impalad coordinator's local catalog cache.

Limitation:

HDFS caching is not supported in On-demand metadata mode coordinators.

Note:

In Impala 3.4.0 and above, global INVALIDATE METADATA statement is supported when On-demand feature is enabled.

INVALIDATE METADATA Usage Notes:

To return accurate query results, Impala needs to keep the metadata current for the databases and tables queried. Through "automatic invalidation" or "HMS event polling" support, Impala automatically picks up most changes in metadata from the underlying systems. However there are some scenarios where you might need to run INVALIDATE METADATA or REFRESH.

- if some other entity modifies information used by Impala in the metastore, the information cached by Impala must be updated via INVALIDATE METADATA or REFRESH,
- if you have "local catalog" enabled without "HMS event polling" and need to pick up metadata changes that were done outside of Impala in Hive and other Hive client, such as SparkSQL,
- and so on.
- **Note:** As this is a very expensive operation compared to the incremental metadata update done by the REFRESH statement, when possible, prefer REFRESH rather than INVALIDATE METADATA.

Related Information

Invalidate Metadata Statement

Automatic Invalidation of Metadata Cache

To keep the size of metadata bounded, the Impala Catalog Server periodically scans all the tables and invalidates those not recently used.

There are two types of configurations in Catalog Server that control the automatic invalidation of metadata in the Catalog Server Command Line Argument Advanced Configuration Snippet (Safety Valve) field in Cloudera Manager.

Time-based cache invalidation

Catalogd invalidates tables that are not recently used in the specified time period (in seconds).

The ##invalidate_tables_timeout_s flag needs to be applied to both impalad and catalogd.

Memory-based cache invalidation

When the memory pressure reaches 60% of JVM heap size after a Java garbage collection in catalogd, Impala invalidates 10% of the least recently used tables.

The ##invalidate_tables_on_memory_pressure flag needs to be applied to both impalad and cata logd.

Automatic invalidation of metadata provides more stability with lower chances of running out of memory, but the feature could potentially cause performance issues and may require tuning.

Automatic Invalidation/Refresh of Metadata

In this release, you can invalidate or refresh metadata automatically after changes to databases, tables or partitions render metadata stale. You control the synching of tables or database metadata by basing the process on events. You learn how to access metrics and state information about the event processor.

When tools such as Hive and Spark are used to process the raw data ingested into Hive tables, new HMS metadata (database, tables, partitions) and filesystem metadata (new files in existing partitions/tables) are generated. In previous versions of Impala, in order to pick up this new information, Impala users needed to manually issue an INVALIDATE or REFRESH commands.

When automatic invalidate/refresh of metadata is enabled,, the Catalog Server polls Hive Metastore (HMS) notification events at a configurable interval and automatically applies the changes to Impala catalog.

Impala Catalog Server polls and processes the following changes.

- Refreshes the tables when it receives the ALTER TABLE event.
- Refreshes the partition when it receives the ALTER, ADD, or DROP partitions.
- Adds the tables or databases when it receives the CREATE TABLE or CREATE DATABASE events.
- Removes the tables from catalogd when it receives the DROP TABLE or DROP DATABASE events.
- Refreshes the table and partitions when it receives the INSERT events.

  If the table is not loaded at the time of processing the INSERT event, the event processor does not need to refresh the table and skips it.
- Changes the database and updates catalogd when it receives the ALTER DATABASE events. The following changes are supported. This event does not invalidate the tables in the database.
  - Change the database properties
  - Change the comment on the database
  - Change the owner of the database
  - Change the default location of the database

    Changing the default location of the database does not move the tables of that database to the new location. Only the new tables which are created subsequently use the default location of the database in case it is not provided in the create table statement.

This feature is controlled by the ##hms_event_polling_interval_s flag. Start the catalogd with the ##hms_event_polling_interval_s flag set to a positive integer to enable the feature and set the polling frequency in seconds. We recommend the value to be less than 5 seconds.

Limitations

The following use cases are not supported:

- When you bypass HMS and add or remove data into table by adding files directly on the filesystem, HMS does not generate the INSERT event, and the event processor will not invalidate the corresponding table or refresh the corresponding partition.

  It is recommended that you use the LOAD DATA command to do the data load in such cases, so that event processor can act on the events generated by the LOAD command.

- The Spark API that saves data to a specified location does not generate events in HMS, thus is not supported. For example:

```
Seq((1, 2)).toDF("i", "j").write.save("/user/hive/warehouse/spark_etl.db/
customers/date=01012019")
```

- Event processing could have delays due to the polling interval and auto-refresh on large tables also takes time. If you want the metadata to be synced up immediately, manual REFRESH/INVALIDATE is a better choice and has a better guarantee.

Disable Event Based Automatic Metadata Sync

When the ##hms_event_polling_interval_s flag is set to a non-zero value for your catalogd, the event-based automatic invalidation is enabled for all databases and tables. If you wish to have the fine-grained control on which tables or databases need to be synced using events, you can use the impala.disableHmsSync property to disable the event processing at the table or database level.

This feature can be turned off by setting the ##hms_event_polling_interval_s flag set to 0.

When you add the DBPROPERTIES or TBLPROPERTIES with the impala.disableHmsSync key, the HMS event based sync is turned on or off. The value of the impala.disableHmsSync property determines if the event processing needs to be disabled for a particular table or database.

- If 'impala.disableHmsSync'='true', the events for that table or database are ignored and not synced with HMS.
- If 'impala.disableHmsSync'='false' or if impala.disableHmsSync is not set, the automatic sync with HMS is enabled if the ##hms_event_polling_interval_s global flag is set to non-zero.
- To disable the event based HMS sync for a new database, set the impala.disableHmsSync database properties in Hive as currently, Impala does not support setting database properties:

```
CREATE DATABASE <name> WITH DBPROPERTIES ('impala.disableHmsSync'='true');
```

- To enable or disable the event based HMS sync for a table:

```
CREATE TABLE <name> ... TBLPROPERTIES ('impala.disableHmsSync'='true' |
'false');
```

- To change the event based HMS sync at the table level:

```
ALTER TABLE <name> SET TBLPROPERTIES ('impala.disableHmsSync'='true' | '
false');
```

When both table and database level properties are set, the table level property takes precedence. If the table level property is not set, then the database level property is used to evaluate if the event needs to be processed or not.

If the property is changed from true (meaning events are skipped) to false (meaning events are not skipped), you need to issue a manual INVALIDATE METADATA command to reset event processor because it doesn't know how many events have been skipped in the past and cannot know if the object in the event is the latest. In such a case, the status of the event processor changes to NEEDS_INVALIDATE.

Metrics for Event Based Automatic Metadata Sync

You can use the web UI of the catalogd to check the state of the automatic invalidate event processor.

By default, the debug web UI of catalogd is at http://*IMPALA-SERVER-HOSTNAME*:25020 (non-secure cluster) or https://*IMPALA-SERVER-HOSTNAME*:25020 (secure cluster).

Under the web UI, there are two pages that presents the metrics for HMS event processor that is responsible for the event based automatic metadata sync.

- /metrics#events

- /events

  This provides a detailed view of the metrics of the event processor, including min, max, mean, median, of the durations and rate metrics for all the counters listed on the /metrics#events page.

The /metrics#events page provides the following metrics about the HMS event processor.

Name	Description
events-processor.avg-events-fetch-duration	Average duration to fetch a batch of events and process it.
events-processor.avg-events-process-duration	Average time taken to process a batch of events received from the Metastore.
events-processor.events-received	Total number of the Metastore events received.
events-processor.events-received-15min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 15 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-received-1min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 1 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-received-5min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 5 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-skipped	Total number of the Metastore events skipped. Events can be skipped based on certain flags are table and database level. You can use this metric to make decisions, such as: • If most of the events are being skipped, see if you might just turn off the event processing. • If most of the events are not skipped, see if you need to add flags on certain databases.
events-processor.status	Metastore event processor status to see if there are events being received or not. Possible states are: • PAUSED The event processor is paused because catalog is being reset concurrently. • ACTIVE The event processor is scheduled at a given frequency. • ERROR • The event processor is in error state and event processing has stopped. • NEEDS_INVALIDATE The event processor could not resolve certain events and needs a manual INVALIDATE command to reset the state. • STOPPED The event processing has been shutdown. No events will be processed. • DISABLED The event processor is not configured to run.

Synchronization between Impala Clusters

Describes the event that notifies other Impala clusters about the changes of LOAD data statement.

When a LOAD statement is run from an Impala in a cluster, an INSERT event is generated. This INSERT event generated notifies other Impala clusters or any other systems that consume HMS events (e.g. Hive Replication) about the changes of LOAD DATA. The other Impala clusters will refresh the table or partitions based on the event. This allows the metadata to be always consistent locally and remotely. However, there can be delays till the event is processed but eventually, it will be consistent. Also, an auto-refresh on large tables takes time. If you want

the metadata to be synced up immediately, manual REFRESH/INVALIDATE is a better choice and has a better guarantee.

When the LOAD DATA statement operates on a partitioned table, it always operates on one partition at a time. So for unpartitioned tables, the granularity of the refresh is the whole table. For partitioned tables, only the partition will be reloaded.

Cloudera Runtime 7.3.1

Monitoring Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Impala Logs **4**
Managing Logs 4
Impala lineage **6**
Web User Interface for Debugging **6**
Debug Web UI for Impala Daemon 7
Debug Web UI for StateStore 10
Debug Web UI for Catalog Server 12
Configuring Impala Web UI 13
Debug Web UI for Query Timeline 15

Impala Logs

You can review Impala log files on each host, when you have traced an issue back to a specific system. The Impala logs contain information about any errors Impala encountered, jobs Impala has completed, and settings Impala is configured with.

The logs store information about Impala startup options. This information appears once for each time Impala is started and may include:

- Machine name.
- Impala version number.
- Flags used to start Impala.
- CPU information.
- The number of available disks.

A new set of log files is produced each time the associated daemon is restarted. These log files have long names including a timestamp. The .INFO, .WARNING, and .ERROR files are physically represented as symbolic links to the latest applicable log files.

Review Impala log files on each host, when you have traced an issue back to a specific system:

- By using the web interface at http://host-name:25000/logs where host-name is your Cloudera cluster host name.

  The web interface limits the amount of logging information displayed. To view every log entry, access the log files directly through the file system. Impala log files can often be several megabytes in size.
- By examining the contents of the log file

  By default, the Impala logs are stored at /var/log/impalad/, /var/log/catalogd/, and /var/log/statestore/. The most comprehensive log, showing informational, warning, and error messages, is in the file name impalad.INFO.

  For each of impalad, statestored, catalogd:

  - Examine the .INFO files to see configuration settings for the processes.
  - Examine the .WARNING files to see all kinds of problem information, including such things as suboptimal settings and also serious runtime errors.
  - Examine the .ERROR and/or .FATAL files to see only the most serious errors, if the processes crash, or queries fail to complete. These messages are also in the .WARNING file.

Cloudera Manager collects front-end and back-end logs together into a single view and let you do a search across log data for all the managed nodes in Diagnostics Logs .

Managing Logs

You must configure the Impala log settings to change the default log locations, to rotate logs, or to log verbose levels.

Procedure

1. To change log file locations:
   a) In Cloudera Manager, navigate to Impala service Configuration .
   b) In the search field, type log_dir.
   c) Specify the new log directories for Impala Daemon, Catalog Server, or StateStore in the respective fields:
      - Impala Daemon Log Directory
      - Catalog Server Log Directory
      - StateStore Log Directory
   d) Click Save Changes and restart Impala.

2. To set up log rotation:
   a) In Cloudera Manager, navigate to Impala service Configuration .
   b) In the search field, type max_log_files.
   c) Specify the values for Impala Daemon, Catalog Server, or StateStore in the respective fields:
      - Impala Daemon Maximum Log Files
      - Catalog Server Maximum Log Files
      - StateStore Maximum Log Files
   d) Click Save Changes and restart Impala.

   The above configuration option specifies how many log files to keep at each severity level (INFO, WARNING, ERROR, and FATAL).

   - A value of 0 preserves all log files, in which case you would set up set up manual log rotation using your Linux tool or technique of choice.
   - A value of 1 preserves only the very latest log file.
   - The default value is 10.

   For some log levels, Impala logs are first temporarily buffered in memory and only written to disk periodically. The --logbufsecs setting controls the maximum time that log messages are buffered for. For example, with the default value of 5 seconds, there may be up to a 5 second delay before a logged message shows up in the log file.

   It is not recommended that you set --logbufsecs to 0 as the setting makes the Impala daemon to spin in the thread that tries to delete old log files.
3. To specify how often the log information is written to disk:
   a) In Cloudera Manager, navigate to Impala service Configuration .
   b) In the search field, type logbuflevel.
   c) Specify the values for Impala Daemon, Catalog Server, or StateStore in the respective fields:
      - Impala Daemon Log Buffer Level
      - Catalog Server Log Buffer Level
      - StateStore Log Buffer Level

      The default is 0, meaning that the log is immediately flushed to disk when Impala outputs an important messages such as a warning or an error, but less important messages such as informational ones are buffered in memory rather than being flushed to disk immediately.
   d) Click Save Changes and restart Impala.

4. To set the logging verbose levels:
   a) In Cloudera Manager, navigate to Impala service Configuration .
   b) In the search field, type GLOG_v.
   c) Specify the values for Impala Daemon, Catalog Server, or StateStore in the respective fields:
      - Impala Daemon Verbose Log Level
      - Catalog Server Verbose Log Level
      - StateStore Verbose Log Level
   d) Click Save Changes and restart Impala.

   As logging levels increase, the categories of information logged are cumulative. For example, GLOG_v=2 records everything GLOG_v=1 records, as well as additional information.

   Increasing logging levels imposes performance overhead and increases log size. Cloudera recommends using GLOG_v=1 for most cases: this level has minimal performance impact but still captures useful troubleshooting information.

   Additional information logged at each level of GLOG_v is as follows:

   - 1: The default level. Logs information about each connection and query that is initiated to an impalad instance, including runtime profiles.
   - 2: Everything from the previous level plus information for each RPC initiated. This level also records query execution progress information, including details on each file that is read.
   - 3: Everything from the previous level plus logging of every row that is read. This level is only applicable for the most serious troubleshooting and tuning scenarios, because it can produce exceptionally large and detailed log files, potentially leading to its own set of performance and capacity problems.

   Note: For performance reasons, Cloudera highly recommends not setting the most verbose logging level to 3.

Impala lineage

You can use the Atlas lineage graph to understand the source and impact of data and changes to data over time and across all your data.

Atlas collects metadata from Impala to represent the lineage among data assets. The Atlas lineage graph shows the input and output processes that the current entity participated in. Entities are included if they were inputs to processes that lead to the current entity or they are output from processes for which the current entity was an input. Impala processes follow this pattern.

Note that lineage is not updated between a table and views that the table is a part of when an Impala ALTER TABLE operation runs on the table.

Related Information

Impala metadata collection

Web User Interface for Debugging

You can use the Impala daemons (`impalad`, `statestored`, and `catalogd`) Web UI to display the diagnostic and status information. Each of these Impala daemons includes a built-in web server.

Impala Daemon Web UI

The Impala Daemon (`impalad`) Web UI includes information about configuration settings, running and completed queries, and associated performance and resource usage for queries. In particular, the Details link for each query displays alternative views of the query including a

graphical representation of the plan, and the output of the EXPLAIN, SUMMARY, and PROFILE statements from `impala-shell`. Each host that runs the `impalad` daemon has its own instance of the Web UI, with details about those queries for which that host served as the coordinator. The `impalad` Web UI is primarily used for diagnosing query problems that can be traced to a particular node.

StateStore Web UI

The StateStore (`statestored`) Web UI includes information about memory usage, configuration settings, and ongoing health checks performed by statestored. Because there is only a single instance of the statestored within any Impala cluster, you access the Web UI only on the particular host that serves as the Impala StateStore.

Catalog Server Web UI

The Catalog Server (`catalogd`0 Web UI includes information about the databases, tables, and other objects managed by Impala, in addition to the resource usage and configuration settings of the catalogd. Because there is only a single instance of the catalogd within any Impala cluster, you access the Web UI only on the particular host that serves as the Impala Catalog Server.

Debug Web UI for Impala Daemon

You can use the Impala Daemon (`impalad`) Web UI to view information about configuration settings, running and completed queries, and associated performance and resource usage for queries.

To debug and troubleshoot an `impalad` using a web-based interface, open the URL http ://*IMPALA#SERVER#HOSTNAME*:25000/ in a browser. (For secure clusters, use the prefix https:// instead of http://.)

Because each Impala node produces its own set of debug information, you should choose a specific node that you want to investigate an issue on.

Main Page

The main impalad Web UI page at / lists the following information about the impalad:

- The version of the `impalad` daemon

  The Version section also contains other information, such as when Impala was built and what build flags were used.
- Process start time
- Hardware information
- OS information
- Process information
- CGroup information

Admission Controller Page

The Admission Controller impalad debug Web UI is at /admission page under the main impalad Web UI.

Use the /admission page to troubleshoot queued queries and the admission control.

The admission page provides the following information about each resource pool to which queries have been submitted at least once:

- Time since the statestored received the last update
- A warning if this impalad is considered disconnected from the statestored and thus the information on this page could be stale.
- Pool configuration
- Queued queries submitted to this coordinator, in the order of submission
- Running queries submitted to this coordinator

- Pool stats
  - Average of time in queue: An exponential moving average which represents the average time in queue over the last 10 to 12 queries. If a query is admitted immediately, the wait time of 0 is used in calculating this average wait time.
- Histogram of the distribution of peak memory used by queries admitted to the pool

  Use the histogram to figure out settings for the minimum and maximum query MEM_LIMIT ranges for this pool.

  The histogram displays data for all queries admitted to the pool, including the queries that finished, got canceled, or encountered an error.

Click on the pool name to only display information relevant to that pool. You can then refresh the debug page to see only the information for that specific pool.

Click Reset informational stats for all pools to reset the stats that keep track of historical data, such as Totals stats, Time in queue (exponential moving average), and the histogram.

The above information is also available as a JSON object from the following HTTP endpoint:

```
http://IMPALA-SERVER-HOSTNAME:PORT/admission?json
```

Known Backends Page

The Known backends page of the impalad debug Web UI is at /backends under the main impalad Web UI.

This page lists the following info for each of the `impalad` nodes in the cluster. Because each `impalad` daemon knows about every other `impalad` daemon through the StateStore, this information should be the same regardless of which node you select.

- Address of the node: Host name and port
- KRPC address: The KRPC address of the impalad. Use this address when you issue the SHUT DOWN command for this impalad.
- Whether acting as a coordinator
- Whether acting as an executor
- Quiescing status: Specify whether the graceful shutdown process has been initiated on this impa lad.
- Memory limit for admission: The amount of memory that can be admitted to this backend by the admission controller.
- Memory reserved: The amount of memory reserved by queries that are active, either currently executing or finished but not yet closed, on this backend.

  The memory reserved for a query that is currently executing is its memory limit, if set. Otherwise, if the query has no limit or if the query finished executing, the current consumption is used.
- Memory of the queries admitted to this coordinator: The memory submitted to this particular host by the queries admitted by this coordinator.

Catalog Page

The Catalog page of the impalad debug Web UI is at /catalog under the main impalad Web UI.

This page displays a list of databases and associated tables recognized by this instance of `impalad`. You can use this page to locate which database a table is in, check the exact spelling of a database or table name, look for identical table names in multiple databases. The primary debugging use case would be to check if an impalad instance has knowledge of a particular table that someone expects to be in a particular database.

Hadoop Configuration

The Hadoop Configuration page of the impalad debug Web UI is at /hadoop-varz under the main impalad Web UI.

This page displays the Hadoop common configurations that Impala is running with.

JMX

The JMX page of the impalad debug Web UI is at /jmx under the main impalad Web UI.

This page displays monitoring information about various JVM subsystems, such as memory pools, thread management, runtime. etc.

Java Log Level

The Change log level page of the impalad debug Web UI is at /log_level under the main impalad Web UI.

This page displays the current Java and backend log levels, and it allows you to change the log levels dynamically without having to restart the impalad.

Logs Page

The INFO logs page of the impalad debug Web UI is at /logs under the main impalad Web UI.

This page shows the last portion of the impalad.INFO log file, including the info, warning, and error logs for the `impalad`. You can see the details of the most recent operations, whether the operations succeeded or encountered errors. This page provides one central place for the log files and saves you from looking around the filesystem for the log files, which could be in different locations on clusters that use cluster management software.

Memz Page

The Memory Usage page of the impalad debug Web UI is at /memz under the main impalad Web UI.

This page displays the summary and detailed information about memory usage by the impalad.

Metrics Page

The Metrics page of the impalad debug Web UI is at /metrics under the main impalad Web UI.

This page displays the current set of metrics, counters and flags representing various aspects of impalad internal operations.

Queries Page

The Queries page of the impalad debug Web UI is at /queries under the main impalad Web UI.

This page lists:

- Currently running queries
- Queries that have completed their execution, but have not been closed yet
- Completed queries whose details still reside in memory

The queries are listed in reverse chronological order, with the most recent at the top. You can control the amount of memory devoted to completed queries by specifying the -#-#query_log_size startup option for impalad.

This page provides:

- How many SQL statements are failing (State value of EXCEPTION)
- How large the result sets are (# rows fetched)
- How long each statement took (Start Time and End Time)

Click the Details link for a query to display the detailed performance characteristics of that query, such as the profile output.

On the query detail page, in the Profile tab, you have options to export the query profile output to the Thrift, text, or Json format.

The Queries page also includes the Query Locations section that lists the number of running queries with fragments on this host.

RPC Services Page

The RPC durations page of the impalad debug Web UI is at /rpcz under the main impalad Web UI.

This page displays information, such as the duration, about the RPC communications of this impa lad with other Impala daemons.

Sessions Page

The Sessions page of the impalad debug Web UI is at /session under the main impalad Web UI.

This page displays information about the sessions currently connected to this `impalad` instance. For example, sessions could include connections from the impala-shell command, JDBC or ODBC applications, or the Impala Query UI in the Hue web interface.

Threadz Page

The Threads page of the impalad debug Web UI is at /threadz under the main impalad Web UI.

This page displays information about the threads used by this instance of impalad, and it shows which categories they are grouped into. Making use of this information requires substantial knowledge about Impala internals.

Varz Page

The Varz page of the impalad debug Web UI is at /varz under the main impalad Web UI.

This page shows the configuration settings in effect when this instance of impalad communicates with other Hadoop components such as HDFS and YARN. These settings are collected from a set of configuration files.

The bottom of this page also lists all the command-line settings in effect for this instance of impa lad.

Prometheus Metrics Page

At /metrics_prometheus, under the main impalad Web UI, the metrics are generated in Prometheus exposition format that Prometheus can consume for event monitoring and alerting.

The /metrics_prometheus is not shown in the Web UI list of pages.

Debug Web UI for StateStore

You can use the StateStore (`statestored`) Web UI to view information about memory usage, configuration settings, and ongoing health checks performed by statestored.

To debug and troubleshoot the statestored daemon using a web-based interface, open the URL http ://*IMPALA#SERVER#HOSTNAME*:25010/ in a browser. (For secure clusters, use the prefix https:// instead of http://.)

Main Page

The main statestored Web UI page at / lists the following information about the statestored:

- The version of the statestored daemon
- Process start time
- Hardware information
- OS information
- Process information
- CGroup information

Logs Page

The INFO logs page of the debug Web UI is at /logs under the main statestored Web UI.

This page shows the last portion of the statestored.INFO log file, including the info, warning, and error logs for the statestored. You can refer here to see the details of the most recent operations, whether the operations succeeded or encountered errors. This page provides one central place for the log files and saves you from looking around the filesystem for the log files, which could be in different locations on clusters that use cluster management software.

Memz Page

The Memory Usage page of the debug Web UI is at /memz under the main statestored Web UI.

This page displays summary and detailed information about memory usage by the statestored. You can see the memory limit in effect for the node, and how much of that memory Impala is currently using.

Metrics Page

The Metrics page of the debug Web UI is at /metrics under the main statestored Web UI.

This page displays the current set of metrics: counters and flags representing various aspects of stat estored internal operation.

RPC Services Page

The RPC durations page of the statestored debug Web UI is at /rpcz under the main statestored Web UI.

This page displays information, such as the durations, about the RPC communications of this stat estored with other Impala daemons.

Subscribers Page

The Subscribers page of the debug Web UI is at /subscribers under the main statestored Web UI.

This page displays information about the other Impala daemons that have registered with the stat estored to receive and send updates.

Threadz Page

The Threads page of the debug Web UI is at /threadz under the main statestored Web UI.

This page displays information about the threads used by this instance of statestored, and shows which categories they are grouped into. Making use of this information requires substantial knowledge about Impala internals.

Topics Page

The Topics page of the debug Web UI is at /topics under the main statestored Web UI.

This page displays information about the topics to which the other Impala daemons have registered to receive updates.

Varz Page

The Varz page of the debug Web UI is at /varz under the main statestored Web UI.

This page shows the configuration settings in effect when this instance of statestored communicates with other Hadoop components such as HDFS and YARN. These settings are collected from a set of configuration files.

The bottom of this page also lists all the command-line settings in effect for this instance of statesto red.

Prometheus Metrics Page

At /metrics_prometheus, under the main statestored Web UI, the metrics are generated in Prometheus exposition format that Prometheus can consume for event monitoring and alerting.

The /metrics_prometheus is not shown in the Web UI list of pages.

Debug Web UI for Catalog Server

You can use the Catalog Server (`catalogd`) Web UI to view information about the databases, tables, and other objects managed by Impala, in addition to the resource usage and configuration settings of the catalogd.

The main page of the debug Web UI is at http://*IMPALA#SERVER#HOSTNAME*:25020/ (non-secure cluster) or https://*IMPALA#SERVER#HOSTNAME*:25020/ (secure cluster).

Main Page

The main catalogd Web UI page at / lists the following information about the catalogd:

- The version of the catalogd daemon
- Process start time
- Hardware information
- OS information
- Process information
- CGroup information

Catalog Page

The Catalog page of the debug Web UI is at /catalog under the main catalogd Web UI.

This page displays a list of databases and associated tables recognized by this instance of catalogd. You can use this page to locate which database a table is in, check the exact spelling of a database or table name, look for identical table names in multiple databases. The catalog information is represented as the underlying Thrift data structures.

JMX

The JMX page of the catalogd debug Web UI is at /jmx under the main catalogd Web UI.

This page displays monitoring information about various JVM subsystems, such as memory pools, thread management, runtime. etc.

Java Log Level

The Change log level page of the catalogd debug Web UI is at /log_level under the main catalogd Web UI.

The page displays the current Java and backend log levels and allows you to change the log levels dynamically without having to restart the catalogd

Logs Page

The INFO logs page of the debug Web UI is at /logs under the main catalogd Web UI.

This page shows the last portion of the catalogd.INFO log file, including the info, warning, and error logs for the catalogd daemon. You can refer here to see the details of the most recent operations, whether the operations succeeded or encountered errors. This page provides one central place for the log files and saves you from looking around the filesystem for the log files, which could be in different locations on clusters that use cluster management software.

Memz Page

The Memory Usage page of the debug Web UI is at /memz under the main catalogd Web UI.

This page displays summary and detailed information about memory usage by the catalogd. You can see the memory limit in effect for the node, and how much of that memory Impala is currently using.

Metrics Page

The Metrics page of the debug Web UI is at /metrics under the main catalogd Web UI.

This page displays the current set of metrics: counters and flags representing various aspects of cata logd internal operation.

RPC Services Page

The RPC durations page of the catalogd debug Web UI is at /rpcz under the main catalogd Web UI.

This page displays information, such as the durations, about the RPC communications of this cata logd with other Impala daemons.

Threadz Page

The Threads page of the debug Web UI is at /threadz under the main catalogd Web UI.

This page displays information about the threads used by this instance of catalogd, and shows which categories they are grouped into. Making use of this information requires substantial knowledge about Impala internals.

Varz Page

The Varz page of the debug Web UI is at /varz under the main catalogd Web UI.

This page shows the configuration settings in effect when this instance of catalogd communicates with other Hadoop components such as HDFS and YARN. These settings are collected from a set of configuration files.

The bottom of this page also lists all the command-line settings in effect for this instance of cata logd.

Prometheus Metrics Page

At /metrics_prometheus, under the main catalogd Web UI, the metrics are generated in Prometheus exposition format that Prometheus can consume for event monitoring and alerting.

The /metrics_prometheus is not shown in the Web UI list of pages.

Configuring Impala Web UI

As an administrator, you can diagnose issues with each daemon on a particular host, or perform other administrative actions such as cancelling a running query from the built-in web server's UI. The built-in web server is inlcuded within each of the Impala-related daemons. By default, these web servers are enabled. You can turn them off in a high-security configuration where it is not appropriate for users to have access to this kind of monitoring information through a web interface.

Enabling and Disabling Access to Impala Web Servers

By default, these web servers are enabled. You might turn them off in a high-security configuration where it is not appropriate for users to have access to this kind of monitoring information through a web interface.

To enable or disable Impala Web Servers for Web UI in Cloudera Manager:

- Impala Daemon
  1. Navigate to ClustersImpala ServiceConfiguration.
  2. Select ScopeImpala Daemon .
  3. Select CategoryPorts and Addresses.
  4. Select or clear Enable Impala Daemon Web Server.
  5. Click Save Changes, and restart the Impala service.

- Impala StateStore
  1. Navigate to ClustersImpala ServiceConfiguration.
  2. Select ScopeImpala StateStore.
  3. Select CategoryMain.
  4. Select or clear Enable StateStore Web Server.
  5. Click Save Changes, and restart the Impala service.
- Impala Catalog Server
  1. Navigate to ClustersImpala ServiceConfiguration.
  2. Select ScopeImpala Catalog Server.
  3. Select CategoryMain.
  4. Check or clear Enable Catalog Server Web Server.
  5. Click Save Changes, and restart the Impala service.

Configuring Secure Access for Impala Web Servers

Cloudera Manager supports two methods of authentication for secure access to the Impala Catalog Server, Impala Daemon, and StateStore web servers: password-based authentication and SPNEGO authentication.

Authentication for the three types of daemons can be configured independently.

Configuring Password Authentication

1. Navigate to ClustersImpala ServiceConfiguration.
2. Search for "password" using the Search box in the Configuration tab. This should display the password-related properties (Username and Password properties) for the Impala Daemon, StateStore, and Catalog Server. If there are multiple role groups configured for Impala Daemon instances, the search should display all of them.
3. Enter a username and password into these fields.
4. Click Save Changes, and restart the Impala service.

Now when you access the Web UI for the Impala Daemon, StateStore, or Catalog Server, you are asked to log in before access is granted.

Enabling Kerberos HTTP SPNEGO Authentication for Web UI

To provide security through Kerberos, Impala Web UIs support SPNEGO. SPNEGO is a protocol for securing HTTP requests with Kerberos by passing negotiation tokens through HTTP headers.

To enable authorization using SPNEGO in Cloudera Manager:

1. Navigate to ClustersImpala ServiceConfiguration.
2. Select ScopeImpala 1 (Service-Wid).
3. Select the Enable Kerberos Authentication for HTTP Web-Consoles field.

   This setting is effective only Kerberos is enabled for the HDFS service.
4. Click Save Changes, and restart the Impala service.

Configuring TLS/SSL for Web UI

1. Create or obtain an TLS/SSL certificate.
2. Place the certificate, in .pem format, on the hosts where the Impala Catalog Server and StateStore are running, and on each host where an Impala Daemon is running. It can be placed in any location (path) you choose. If all the Impala Daemons are members of the same role group, then the .pem file must have the same path on every host.
3. Navigate to ClustersImpala ServiceConfiguration.
4. Search for "certificate" using the Search box in the Configuration tab. This should display the certificate file location properties for the Impala Catalog Server, Impala Daemon, and StateStore. If there are multiple role groups configured for Impala Daemon instances, the search should display all of them.

5. In the property fields, enter the full path name to the certificate file, the private key file path, and the password for the private key file..
6. Click Save Changes, and restart the Impala service.

Important: If Cloudera Manager cannot find the .pem file on the host for a specific role instance, that role will fail to start.

When you access the Web UI for the Impala Catalog Server, Impala Daemon, and StateStore, https will be used.

Opening Impala Web UIs

Procedure

1. Navigate to ClustersImpala ServiceConfiguration.
2. Open the appropriate Web UI:Select Web UIImpala Catalog Web UI.
   - To open StateStore Web UI, select Web UIImpala StateStore Web UI.
   - To open Catalog Server Web UI, select Web UIImpala Catalog Web UI.
   - To open Impala Daemon Web UI:
     a. Click the Instances tab.
     b. Click an Impala Daemon instance.
     c. Click Impala Daemon Web UI.

Debug Web UI for Query Timeline

For a detailed report on how a query was executed and to understand the detailed performance characteristics of a query, you can use the built-in web server's UI and look at the Gantt chart.

The Gantt chart is an alternative to the profile command and is a graphical display in the WebUI that renders timing information and dependencies. This chart allows you to determine where queries are spending time instead of looking for timing information in the profile. Once you identify which fragments or operators are consuming the most time you may need to look at the summary profile or full profile to drill down further.

In subsequent releases, the timeline will be enhanced to display additional details so that looking at the profiles will not be necessary.

Here is the screenshot of the Gantt chart that shows the Query timeline of a TPC_DS query.

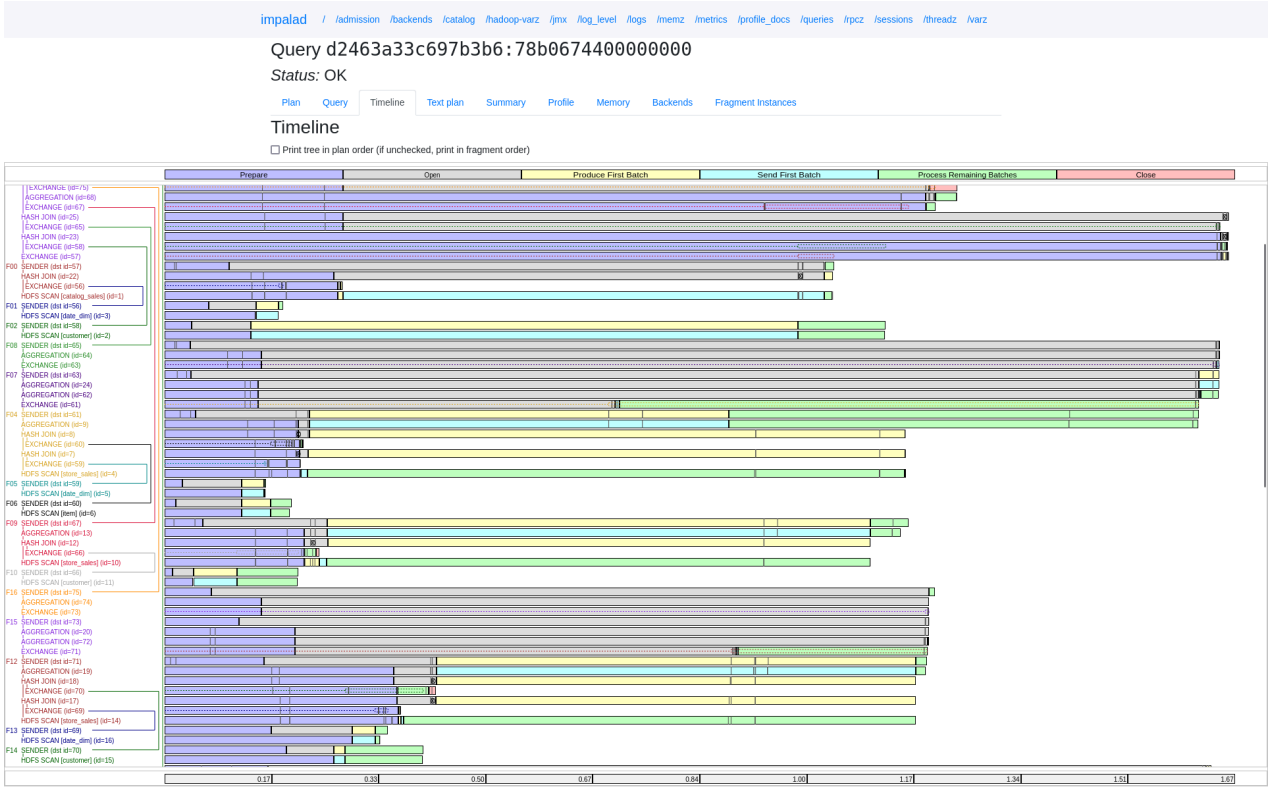
impalad / /admission /backends /catalog /hadoop-varz /jmx /log_level /logs /memz /metrics /profile_docs /queries /rpcz /sessions /threadz /varz
Query d2463a33c697b3b6:78b0674400000000
Status: OK
Plan Query Timeline Text plan Summary Profile Memory Backends Fragment Instances
Timeline
Print tree in plan order (if unchecked, print in fragment order)
Prepare
Open
Produce First Batch
Send First Batch
Process Remaining Batches
Close
0.17
0.33
0.50
0.67
0.84
1.00
1.17
1.34
1.51
1.67

Cloudera Runtime 7.3.1

Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Impala Overview **4**
Components of Impala **4**

Apache Impala Overview

The Apache Impala provides high-performance, low-latency SQL queries on data stored in popular Apache Hadoop file formats.

The Impala solution is composed of the following components.

Impala

The Impala service coordinates and executes queries received from clients. Queries are distributed among Impala nodes, and these nodes then act as workers, executing parallel query fragments.

Hive Metastore

Stores information about the data available to Impala. For example, the metastore lets Impala know what databases are available and what the structure of those databases is. As you create, drop, and alter schema objects, load data into tables, and so on through Impala SQL statements, the relevant metadata changes are automatically broadcast to all Impala nodes by the dedicated catalog service.

Clients

Entities including Hue, ODBC clients, JDBC clients, Business Intelligence applications, and the Impala Shell can all interact with Impala. These interfaces are typically used to issue queries or complete administrative tasks such as connecting to Impala.

Storage for data to be queried

Queries executed using Impala are handled as follows:

1. User applications send SQL queries to Impala through ODBC or JDBC, which provide standardized querying interfaces. The user application may connect to any impalad in the cluster. This impalad becomes the coordinator for the query.
2. Impala parses the query and analyzes it to determine what tasks need to be performed by impalad instances across the cluster. Execution is planned for optimal efficiency.
3. Storage services are accessed by local impalad instances to provide data.
4. Each impalad returns data to the coordinating impalad, which sends these results to the client.

Components of Impala

The Impala service is a distributed, massively parallel processing (MPP) database engine. It consists of different daemon processes that run on specific hosts within your Hadoop cluster.

Impala service consists of the following categories of processes, referred as roles.

Impala Daemon

The core Impala component is the Impala daemon, physically represented by the impalad process. A few of the key functions that an Impala daemon performs are:

- Reads and writes to data files.
- Accepts queries transmitted from the impala-shell command, Hue, JDBC, or ODBC.
- Parallelizes the queries and distributes work across the cluster.
- Transmits intermediate query results back to the central coordinator.

Impala daemons can be deployed in one of the following ways:

- HDFS and Impala are co-located, and each Impala daemon runs on the same host as a DataNode.
- Impala is deployed separately in a compute cluster and reads remotely from HDFS, S3, ADLS, etc.

The Impala daemons are in constant communication with StateStore, to confirm which daemons are healthy and can accept new work.

They also receive broadcast messages from the Catalog Server daemon whenever an Impala daemon in the cluster creates, alters, or drops any type of object, or when an INSERT or LOAD DATA statement is processed through Impala. This background communication minimizes the need for REFRESH or INVALIDATE METADATA statements that were needed to coordinate metadata across Impala daemons.

You can control which hosts act as query coordinators and which act as query executors, to improve scalability for highly concurrent workloads on large clusters.

Note: Impala Daemons should be deployed on nodes using the same Glibc version since different Glibc version supports different Unicode standard version and also ensure that the en_US.UTF-8 locale is installed in the nodes. Not using the same Glibc version might result in inconsistent UTF-8 behavior when UTF8_MODE is set to true.

Impala StateStore

The Impala StateStore checks on the health of all Impala daemons in a cluster, and continuously relays its findings to each of those daemons. It is physically represented by a daemon process named statestored. You only need such a process on one host in a cluster. If an Impala daemon goes offline due to hardware failure, network error, software issue, or other reason, the StateStore informs all the other Impala daemons so that future queries can avoid making requests to the unreachable Impala daemon.

Because the StateStore's purpose is to help when things go wrong and to broadcast metadata to coordinators, it is not always critical to the normal operation of an Impala cluster. If the StateStore is not running or becomes unreachable, the Impala daemons continue running and distributing work among themselves as usual when working with the data known to Impala. The cluster just becomes less robust if other Impala daemons fail, and metadata becomes less consistent as it changes while the StateStore is offline. When the StateStore comes back online, it re-establishes communication with the Impala daemons and resumes its monitoring and broadcasting functions.

If you issue a DDL statement while the StateStore is down, the queries that access the new object the DDL created will fail.

Impala Catalog Server

The Catalog Server relays the metadata changes from Impala SQL statements to all the Impala daemons in a cluster. It is physically represented by a daemon process named catalogd. You only need such a process on one host in a cluster. Because the requests are passed through the StateStore daemon, it makes sense to run the `statestored` and `catalogd` services on the same host.

The catalog service avoids the need to issue REFRESH and INVALIDATE METADATA statements when the metadata changes are performed by statements issued through Impala. When you create a table, load data, and so on through Hive, you do need to issue REFRESH or INVALIDATE METADATA on an Impala node before executing a query there.

Cloudera Runtime 7.3.1

Planning for Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Guidelines for Schema Design **4**
User Account Requirements **5**

Guidelines for Schema Design

Use the guidelines in this topic to construct an optimized and scalable schema that integrates well with your existing data management processes.

Prefer binary file formats over text-based formats

To save space and improve memory usage and query performance, use binary file formats for any large or intensively queried tables. Parquet file format is the most efficient for data warehouse-style analytic queries. Avro is the other binary file format that Impala supports, that you might already have as part of a Hadoop ETL pipeline.

Although Impala can create and query tables with the RCFile and SequenceFile file formats, such tables are relatively bulky due to the text-based nature of those formats, and are not optimized for data warehouse-style queries due to their row-oriented layout. Impala does not support INSERT operations for tables with these file formats.

Guidelines:

- For an efficient and scalable format for large, performance-critical tables, use the Parquet file format.
- To deliver intermediate data during the ETL process, in a format that can also be used by other Hadoop components, Avro is a reasonable choice.
- For convenient import of raw data, use a text table instead of RCFile or SequenceFile, and convert to Parquet in a later stage of the ETL process.

Use Snappy compression where practical

Snappy compression involves low CPU overhead to decompress, while still providing substantial space savings. In cases where you have a choice of compression codecs, such as with the Parquet and Avro file formats, use Snappy compression unless you find a compelling reason to use a different codec.

Prefer numeric types over strings

If you have numeric values that you could treat as either strings or numbers (such as YEAR, MONTH, and DAY for partition key columns), define them as the smallest applicable integer types. For example, YEAR can be SMALLINT, MONTH and DAY can be TINYINT. Although you might not see any difference in the way partitioned tables or text files are laid out on disk, using numeric types will save space in binary formats such as Parquet, and in memory when doing queries, particularly resource-intensive queries such as joins.

Partition, but do not over-partition

Partitioning is an important aspect of performance tuning for Impala. Set up partitioning for your biggest, most intensively queried tables.

If you are moving to Impala from a traditional database system, or just getting started in the Big Data field, you might not have enough data volume to take advantage of Impala parallel queries with your existing partitioning scheme. For example, if you have only a few tens of megabytes of data per day, partitioning by YEAR, MONTH, and DAY columns might be too granular. Most of your cluster might be sitting idle during queries that target a single day, or each node might have very little work to do. Consider reducing the number of partition key columns so that each partition directory contains several gigabytes worth of data.

For example, consider a Parquet table where each data file is 1 HDFS block, with a maximum block size of 1 GB. (In Impala 2.0 and later, the default Parquet block size is reduced to 256 MB. For this exercise, let's assume you have bumped the size back up to 1 GB by setting the query option PARQUET_FILE_SIZE=1g.) if you have a 10-node cluster, you need 10 data files (up to 10 GB) to give each node some work to do for a query. But each core on each machine can process a separate data block in parallel. With 16-core machines on a 10-node cluster, a query could process up to 160 GB fully in parallel. If there are only a few data files per partition, not only are most cluster nodes sitting idle during queries, so are most cores on those machines.

You can reduce the Parquet block size to as low as 128 MB or 64 MB to increase the number of files per partition and improve parallelism. But also consider reducing the level of partitioning so that analytic queries have enough data to work with.

Run COMPUTE STATS after loading data

Impala makes extensive use of statistics about data in the overall table and in each column, to help plan resource-intensive operations such as join queries and inserting into partitioned Parquet tables. Because this information is only available after data is loaded, run the COMPUTE STATS statement on a table after loading or replacing data in a table or partition.

Having accurate statistics can make the difference between a successful operation, or one that fails due to an out-of-memory error or a timeout. When you encounter performance or capacity issues, always use the SHOW STATS statement to check if the statistics are present and up-to-date for all tables in the query.

When doing a join query, Impala consults the statistics for each joined table to determine their relative sizes and to estimate the number of rows produced in each join stage. When doing an INSERT into a Parquet table, Impala consults the statistics for the source table to determine how to distribute the work of constructing the data files for each partition.

Verify sensible execution plans with EXPLAIN and SUMMARY

Before executing a resource-intensive query, use the EXPLAIN statement to get an overview of how Impala intends to parallelize the query and distribute the work. If you see that the query plan is inefficient, you can take tuning steps such as changing file formats, using partitioned tables, running the COMPUTE STATS statement, or adding query hints.

After you run a query, you can see performance-related information about how it actually ran by issuing the SUMM ARY command in `impala-shell`. Prior to Impala 1.4, you would use the PROFILE command, but its highly technical output was only useful for the most experienced users. SUMMARY, new in Impala 1.4, summarizes the most useful information for all stages of execution, for all nodes rather than splitting out figures for each node.

User Account Requirements

Impala creates and uses a user and group named impala.

Do not delete this account or group and do not modify the account's or group's permissions and rights. Ensure no existing systems obstruct the functioning of these accounts and groups. For example, if you have scripts that delete user accounts not in a white-list, add these accounts to the list of permitted accounts.

For correct file deletion during DROP TABLE operations, Impala must be able to move files to the HDFS trashcan. You might need to create an HDFS directory /user/impala, writeable by the impala user, so that the trashcan can be created. Otherwise, data files might remain behind after a DROP TABLE statement.

Impala should not run as root. Best Impala performance is achieved using direct reads, but root is not permitted to use direct reads. Therefore, running Impala as root negatively affects performance.

By default, any user can connect to Impala and access all the associated databases and tables. You can enable authorization and authentication based on the Linux OS user who connects to the Impala server, and the associated groups for that user. These security features do not change the underlying file permission requirements. The impala user still needs to be able to access the data files.

Cloudera Runtime 7.3.1

Apache Impala Reference

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Performance Considerations **4**
Performance Best Practices 4
Query Join Performance 6
Table and Column Statistics 7
Generating Table and Column Statistics 19
Runtime Filtering 22
Min/Max Filtering 25
Bloom Filtering 26
Late Materialization of Columns 27
Partitioning 27
Partition Pruning for Queries 30
HDFS Caching 33
HDFS Block Skew 37
Understanding Performance using EXPLAIN Plan 38
Understanding Performance using SUMMARY Report 39
Understanding Performance using Query Profile 40
Planner changes for CPU usage 41
DDL Bucketed Tables 42
Scalability Considerations **43**
Scaling Limits and Guidelines 48
Dedicated Coordinator 49
Hadoop File Formats Support **53**
Using Text Data Files 54
Using Parquet Data Files 57
Using ORC Data Files 65
Using Avro Data Files 66
Using RCFile Data Files 70
Using SequenceFile Data Files 70
Storage Systems Supports **71**
Impala with HDFS 71
Configure Impala Daemon to spill to HDFS 74
Impala with Kudu 75
Configuring for Kudu Tables 75
Impala DDL for Kudu 76
Impala DML for Kudu Tables 83
Impala with HBase 85
Impala with Azure Data Lake Store (ADLS) 88
Impala with Amazon S3 91
Specifying Impala Credentials to Access S3 95
Impala with Ozone 95
Configure Impala Daemon to spill to Ozone 96
Ports Used by Impala **97**
Migration Guide **98**
Setting up Data Cache for Remote Reads **102**
Managing Metadata in Impala **102**
On-demand Metadata 102
Automatic Invalidation of Metadata Cache 104
Automatic Invalidation/Refresh of Metadata 104
SQL transactions in Impala **107**

Performance Considerations

The following sections explain the factors affecting the performance of Impala features, and procedures for tuning, monitoring, and benchmarking Impala queries and other SQL operations.

Performance Best Practices

Use the performance guidelines and best practices during planning, experimentation, and performance tuning for an Impala-enabled cluster.

Choose the appropriate file format for the data

Typically, for large volumes of data (multiple gigabytes per table or partition), the Parquet file format performs best because of its combination of columnar storage layout, large I/O request size, and compression and encoding.

Note: For smaller volumes of data, a few gigabytes or less for each table or partition, you might not see significant performance differences between file formats. At small data volumes, reduced I/O from an efficient compressed file format can be counterbalanced by reduced opportunity for parallel execution. When planning for a production deployment or conducting benchmarks, always use realistic data volumes to get a true picture of performance and scalability.

Avoid data ingestion processes that produce many small files

When producing data files outside of Impala, prefer either text format or Avro, where you can build up the files row by row. Once the data is in Impala, you can convert it to the more efficient Parquet format and split into multiple data files using a single INSERT ... SELECT statement. Or, if you have the infrastructure to produce multi-megabyte Parquet files as part of your data preparation process, do that and skip the conversion step inside Impala.

Always use INSERT ... SELECT to copy significant volumes of data from table to table within Impala. Avoid INSERT ... VALUES for any substantial volume of data or performance-critical tables, because each such statement produces a separate tiny data file.

For example, if you have thousands of partitions in a Parquet table, each with less than 256 MB of data, consider partitioning in a less granular way, such as by year / month rather than year / month / day. If an inefficient data ingestion process produces thousands of data files in the same table or partition, consider compacting the data by performing an INSERT ... SELECT to copy all the data to a different table; the data will be reorganized into a smaller number of larger files by this process.

Choose partitioning granularity based on actual data volume

Partitioning is a technique that physically divides the data based on values of one or more columns, such as by year, month, day, region, city, section of a web site, and so on. When you issue queries that request a specific value or range of values for the partition key columns, Impala can avoid reading the irrelevant data, potentially yielding a huge savings in disk I/O.

When deciding which column(s) to use for partitioning, choose the right level of granularity. For example, should you partition by year, month, and day, or only by year and month? Choose a partitioning strategy that puts at least 256 MB of data in each partition, to take advantage of HDFS bulk I/O and Impala distributed queries.

Over-partitioning can also cause query planning to take longer than necessary, as Impala prunes the unnecessary partitions. Ideally, keep the number of partitions in the table under 30 thousand.

When preparing data files to go in a partition directory, create several large files rather than many small ones. If you receive data in the form of many small files and have no control over the input format, consider using the INSERT ... SELECT syntax to copy data from one table or partition to another, which compacts the files into a relatively small number (based on the number of nodes in the cluster).

If you need to reduce the overall number of partitions and increase the amount of data in each partition, first look for partition key columns that are rarely referenced or are referenced in non-critical queries (not subject to an SLA). For example, your web site log data might be partitioned by year, month, day, and hour, but if most queries roll up the results by day, perhaps you only need to partition by year, month, and day.

If you need to reduce the granularity even more, consider creating “buckets”, computed values corresponding to different sets of partition key values. For example, you can use the TRUNC() function with a TIMESTAMP column to group date and time values based on intervals such as week or quarter.

Use smallest appropriate integer types for partition key columns

Although it is tempting to use strings for partition key columns, since those values are turned into HDFS directory names anyway, you can minimize memory usage by using numeric values for common partition key fields such as YEAR, MONTH, and DAY. Use the smallest integer type that holds the appropriate range of values, typically TINY INT for MONTH and DAY, and SMALLINT for YEAR. Use the EXTRACT() function to pull out individual date and time fields from a TIMESTAMP value, and CAST() the return value to the appropriate integer type.

Choose an appropriate Parquet block size

By default, the Impala INSERT ... SELECT statement creates Parquet files with a 256 MB block size. (This default was changed in Impala 2.0. Formerly, the limit was 1 GB, but Impala made conservative estimates about compression, resulting in files that were smaller than 1 GB.)

Each Parquet file written by Impala is a single block, allowing the whole file to be processed as a unit by a single host. As you copy Parquet files into HDFS or between HDFS filesystems, use hdfs dfs -pb to preserve the original block size.

If there is only one or a few data block in your Parquet table, or in a partition that is the only one accessed by a query, then you might experience a slowdown for a different reason: not enough data to take advantage of Impala's parallel distributed queries. Each data block is processed by a single core on one of the DataNodes. In a 100-node cluster of 16-core machines, you could potentially process thousands of data files simultaneously. You want to find a sweet spot between “many tiny files” and “single giant file” that balances bulk I/O and parallel processing. You can set the PARQUET_FILE_SIZE query option before doing an INSERT ... SELECT statement to reduce the size of each generated Parquet file. (Specify the file size as an absolute number of bytes, or in Impala 2.0 and later, in units ending with m for megabytes or g for gigabytes.) Run benchmarks with different file sizes to find the right balance point for your particular data volume.

Gather statistics for all tables used in performance-critical or high-volume join queries

Gather the statistics with the COMPUTE STATS statement.

Minimize the overhead of transmitting results back to the client

Use techniques such as:

- Aggregation. If you need to know how many rows match a condition, the total values of matching values from some column, the lowest or highest matching value, and so on, call aggregate functions such as COUNT(), SUM(), and MAX() in the query rather than sending the result set to an application and doing those computations there. Remember that the size of an unaggregated result set could be huge, requiring substantial time to transmit across the network.
- Filtering. Use all applicable tests in the WHERE clause of a query to eliminate rows that are not relevant, rather than producing a big result set and filtering it using application logic.
- LIMIT clause. If you only need to see a few sample values from a result set, or the top or bottom values from a query using ORDER BY, include the LIMIT clause to reduce the size of the result set rather than asking for the full result set and then throwing most of the rows away.
- Avoid overhead from pretty-printing the result set and displaying it on the screen. When you retrieve the results through `impala-shell`, use `impala-shell` options such as -B and --output_delimiter to produce results without special formatting, and redirect output to a file rather than printing to the screen. Consider using INSE RT ... SELECT to write the results directly to new files in HDFS.

Verify that your queries are planned in an efficient logical manner

Examine the EXPLAIN plan for a query before actually running it.

Verify performance characteristics of queries

Verify that the low-level aspects of I/O, memory usage, network bandwidth, CPU utilization, and so on are within expected ranges by examining the query profile for a query after running it.

Hotspot analysis

In the context of Impala, a hotspot is defined as “an Impala daemon that for a single query or a workload is spending a far greater amount of time processing data relative to its neighbours”.

Before discussing the options to tackle this issue, some background is first required to understand how this problem can occur.

By default, the scheduling of scan based plan fragments is deterministic. This means that for multiple queries needing to read the same block of data, the same node will be picked to host the scan. The default scheduling logic does not take into account node workload from prior queries. The complexity of materializing a tuple depends on a few factors, namely: decoding and decompression. If the tuples are densely packed into data pages due to good encoding/compression ratios, there will be more work required when reconstructing the data. Each compression codec offers different performance tradeoffs and should be considered before writing the data. Due to the deterministic nature of the scheduler, single nodes can become bottlenecks for highly concurrent queries that use the same tables.

If, for example, a Parquet based dataset is tiny, e.g. a small dimension table, such that it fits into a single HDFS block (Impala by default will create 256 MB blocks when Parquet is used, each containing a single row group) then there are a number of options that can be considered to resolve the potential scheduling hotspots when querying this data:

- Te scheduler’s deterministic behaviour can be changed using the following query options: REPLICA_PREFERENCE and RANDOM_REPLICA.
- HDFS caching can be used to cache block replicas. This will cause the Impala scheduler to randomly pick a node that is hosting a cached block replica for the scan. Note, although HDFS caching has benefits, it serves only to help with the reading of raw block data and not cached tuple data, but with the right number of cached replicas (by default, HDFS only caches one replica), even load distribution can be achieved for smaller datasets.
- Do not compress the table data. The uncompressed table data spans more nodes and eliminates skew caused by compression.
- Reduce the Parquet file size via the PARQUET_FILE_SIZE query option when writing the table data. Using this approach the data will span more nodes. However it’s not recommended to drop the size below 32 MB.

Query Join Performance

Joins are the main class of queries that you can tune at the SQL level.

Queries involving join operations often require more tuning than queries that refer to only one table. The maximum size of the result set from a join query is the product of the number of rows in all the joined tables. When joining several tables with millions or billions of rows, any missed opportunity to filter the result set, or other inefficiency in the query, could lead to an operation that does not finish in a practical time and has to be cancelled.

The simplest technique for tuning an Impala join query is to collect statistics on each table involved in the join using the COMPUTE STATS statement, and then to let Impala automatically optimize the query based on the size of each table, number of distinct values of each column, and so on. For accurate statistics about each table, issue the COMPUTE STATS statement after loading the data into that table, and again if the amount of data changes substantially due to operations, such as INSERT, LOAD DATA, or adding a partition.

If statistics are not available for all the tables in the join query, or if Impala chooses a join order that is not the most efficient, you can override the automatic join order optimization by specifying the STRAIGHT_JOIN keyword immediately after the SELECT and any DISTINCT or ALL keywords. In this case, Impala uses the order the tables appear in the query to guide how the joins are processed.

When you use the STRAIGHT_JOIN technique, you must order the tables in the join query manually instead of relying on the Impala optimizer. The optimizer uses sophisticated techniques to estimate the size of the result set at each stage of the join. For manual ordering, use this heuristic approach to start with, and then experiment to fine-tune the order:

- Specify the largest table first. This table is read from disk by each Impala node and so its size is not significant in terms of memory usage during the query.
- Next, specify the smallest table. The contents of the second, third, and so on tables are all transmitted across the network. You want to minimize the size of the result set from each subsequent stage of the join query. The most likely approach involves joining a small table first, so that the result set remains small even as subsequent larger tables are processed.
- Join the next smallest table, then the next smallest, and so on.

For example, if you had tables BIG, MEDIUM, SMALL, and TINY, the logical join order to try would be BIG, TINY, SMALL, MEDIUM.

The terms “largest” and “smallest” refers to the size of the intermediate result set based on the number of rows and columns from each table that are part of the result set. For example, if you join one table sales with another table cust omers, a query might find results from 100 different customers who made a total of 5000 purchases. In that case, you would specify SELECT ... FROM sales JOIN customers ..., putting customers on the right side because it is smaller in the context of this query.

The Impala query planner chooses between different techniques for performing join queries, depending on the absolute and relative sizes of the tables. Broadcast joins are the default, where the right-hand table is considered to be smaller than the left-hand table, and its contents are sent to all the other nodes involved in the query. The alternative technique is known as a partitioned join (not related to a partitioned table), which is more suitable for large tables of roughly equal size. With this technique, portions of each table are sent to appropriate other nodes where those subsets of rows can be processed in parallel. The choice of broadcast or partitioned join also depends on statistics being available for all tables in the join, gathered by the COMPUTE STATS statement.

To see which join strategy is used for a particular query, issue an EXPLAIN statement for the query. If you find that a query uses a broadcast join when you know through benchmarking that a partitioned join would be more efficient, or vice versa, add a hint to the query to specify the precise join mechanism to use.

How Joins Are Processed when Statistics Are Unavailable

If table or column statistics are not available for some tables in a join, Impala still reorders the tables using the information that is available. Tables with statistics are placed on the left side of the join order, in descending order of cost based on overall size and cardinality. Tables without statistics are treated as zero-size, that is, they are always placed on the right side of the join order.

Overriding Join Reordering with STRAIGHT_JOIN

If an Impala join query is inefficient because of outdated statistics or unexpected data distribution, you can keep Impala from reordering the joined tables by using the STRAIGHT_JOIN keyword immediately after the SELECT and any DISTINCT or ALL keywords. The STRAIGHT_JOIN keyword turns off the reordering of join clauses that Impala does internally, and produces a plan that relies on the join clauses being ordered optimally in the query text.

Note: The STRAIGHT_JOIN hint affects the join order of table references in the query block containing the hint. It does not affect the join order of nested queries, such as views, inline views, or WHERE-clause subqueries. To use this hint for performance tuning of complex queries, apply the hint to all query blocks that need a fixed join order.

Table and Column Statistics

Impala can do better optimization for complex or multi-table queries when it has access to statistics about the volume of data and how the values are distributed. Impala uses this information to help parallelize and distribute the work for a query. For example, optimizing join queries requires a way of determining if one table is “bigger” than another,

which is a function of the number of rows and the average row size for each table. This topic described the categories of statistics Impala can work with, and how to produce them and keep them up to date.

Overview of table statistics

The Impala query planner can make use of statistics about entire tables and partitions. This information includes physical characteristics such as the number of rows, number of data files, the total size of the data files, and the file format. For partitioned tables, the numbers are calculated per partition, and as totals for the whole table. This metadata is stored in the Metastore database, and can be updated by either Impala or Hive.

If a number is not available, the value -1 is used as a placeholder. Some numbers, such as number and total sizes of data files, are always kept up to date because they can be calculated cheaply, as part of gathering HDFS block metadata.

The following example shows table stats for an unpartitioned Parquet table. The values for the number and sizes of files are always available. Initially, the number of rows is not known, because it requires a potentially expensive scan through the entire table, and so that value is displayed as -1. The COMPUTE STATS statement fills in any unknown table stats values.

```
show table stats parquet_snappy;
+-------+--------+---------+--------------+-------------------+---------+-
------------------+...
| #Rows | #Files | Size    | Bytes Cached | Cache Replication | Format  |
 Incremental stats |...
+-------+--------+---------+--------------+-------------------+---------+---
----------------+...
| -1    | 96     | 23.35GB | NOT CACHED   | NOT CACHED        | PARQUET | f
alse             |...
+-------+--------+---------+--------------+-------------------+---------+-
------------------+...

compute stats parquet_snappy;
+------------------------------------------+
| summary                                  |
+------------------------------------------+
| Updated 1 partition(s) and 6 column(s). |
+------------------------------------------+


show table stats parquet_snappy;
+------------+--------+---------+--------------+-------------------+-------
--+-------------------+...
| #Rows      | #Files | Size    | Bytes Cached | Cache Replication | Format
  | Incremental stats |...
+------------+--------+---------+--------------+-------------------+-------
--+-------------------+...
| 1000000000 | 96     | 23.35GB | NOT CACHED   | NOT CACHED        | PARQ
UET | false            |...
+------------+--------+---------+--------------+-------------------+-------
--+-------------------+...
```

To check that table statistics are available for a table, and see the details of those statistics, use the statement SHOW TABLE STATS *TABLE_NAME*.

Overview of Column Statistics

The Impala query planner can make use of statistics about individual columns when that metadata is available in the metastore database. This technique is most valuable for columns compared across tables in join queries, to help estimate how many rows the query will retrieve from each table. These statistics are also important for correlated subqueries using the EXISTS or IN operators, which are processed internally the same way as join queries.

The following example shows column stats for an unpartitioned Parquet table. The values for the maximum and average sizes of some types are always available, because those figures are constant for numeric and other fixed-size types. Initially, the number of distinct values is not known, because it requires a potentially expensive scan through the entire table, and so that value is displayed as -1. The same applies to maximum and average sizes of variable-sized types, such as STRING. The COMPUTE STATS statement fills in most unknown column stats values. (It does not record the number of NULL values, because currently Impala does not use that figure for query optimization.)

```
show column stats parquet_snappy;
+-------------+----------+------------------+--------+----------+----------+
| Column      | Type     | #Distinct Values | #Nulls | Max Size | Avg Size |
+-------------+----------+------------------+--------+----------+----------+
| id          | BIGINT   | -1               | -1     | 8        | 8        |
| val         | INT      | -1               | -1     | 4        | 4        |
| zerofill    | STRING   | -1               | -1     | -1       | -1       |
| name        | STRING   | -1               | -1     | -1       | -1       |
| assertion   | BOOLEAN  | -1               | -1     | 1        | 1        |
| location_id | SMALLINT | -1               | -1     | 2        | 2        |
+-------------+----------+------------------+--------+----------+----------+

compute stats parquet_snappy;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 6 column(s). |
+-----------------------------------------+
show column stats parquet_snappy;
+-------------+----------+------------------+--------+----------+--------
-----------+
| Column      | Type     | #Distinct Values | #Nulls | Max Size | Avg Size
        |
+-------------+----------+------------------+--------+----------+--------
-----------+
| id          | BIGINT   | 183861280        | -1     | 8        | 8
        |
| val         | INT      | 139017           | -1     | 4        | 4
         |
| zerofill    | STRING   | 101761           | -1     | 6        | 6
        |
| name        | STRING   | 145636240        | -1     | 22       | 13.0002
0027160645 |
| assertion   | BOOLEAN  | 2                | -1     | 1        | 1
        |
| location_id | SMALLINT | 339              | -1     | 2        | 2
         |
+-------------+----------+------------------+--------+----------+----------
---------+
```

Note:

For column statistics to be effective in Impala, you also need to have table statistics for the applicable tables. When you use the Impala COMPUTE STATS statement, both table and column statistics are automatically gathered at the same time, for all columns in the table.

Note: Because Impala does not currently use the NULL count during query planning, Impala speeds up the COMPUTE STATS statement by skipping this NULL counting.

To check whether column statistics are available for a particular set of columns, use the SHOW COLUMN STATS *TABLE_NAME* statement, or check the extended EXPLAIN output for a query against that table that refers to those columns.

How Table and Column Statistics Work for Partitioned Tables

When you use Impala for “big data”, you are highly likely to use partitioning for your biggest tables, the ones representing data that can be logically divided based on dates, geographic regions, or similar criteria. The table and column statistics are especially useful for optimizing queries on such tables. For example, a query involving one year might involve substantially more or less data than a query involving a different year, or a range of several years. Each query might be optimized differently as a result.

The following examples show how table and column stats work with a partitioned table. The table for this example is partitioned by year, month, and day. For simplicity, the sample data consists of 5 partitions, all from the same year and month. Table stats are collected independently for each partition. (In fact, the SHOW PARTITIONS statement displays exactly the same information as SHOW TABLE STATS for a partitioned table.) Column stats apply to the entire table, not to individual partitions. Because the partition key column values are represented as HDFS directories, their characteristics are typically known in advance, even when the values for non-key columns are shown as -1.

```
show partitions year_month_day;
+-------+-------+-----+-------+--------+---------+--------------+----------
---------+---------+...
| year  | month | day | #Rows | #Files | Size    | Bytes Cached | Cache Repl
ication | Format  |...
+-------+-------+-----+-------+--------+---------+--------------+--------
-----------+---------+...
| 2013  | 12    | 1   | -1    | 1      | 2.51MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 2   | -1    | 1      | 2.53MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 3   | -1    | 1      | 2.52MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 4   | -1    | 1      | 2.51MB  | NOT CACHED   | NOT CAC
HED        | PARQUET |...
| 2013  | 12    | 5   | -1    | 1      | 2.52MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| Total |       |     | -1    | 5      | 12.58MB | 0B           |
        |         |...
+-------+-------+-----+-------+--------+---------+--------------+-----------
--------+---------+...

show table stats year_month_day;
+-------+-------+-----+-------+--------+---------+--------------+----------
---------+---------+...
| year  | month | day | #Rows | #Files | Size    | Bytes Cached | Cache Repl
ication | Format  |...
+-------+-------+-----+-------+--------+---------+--------------+--------
-----------+---------+...
| 2013  | 12    | 1   | -1    | 1      | 2.51MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 2   | -1    | 1      | 2.53MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 3   | -1    | 1      | 2.52MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| 2013  | 12    | 4   | -1    | 1      | 2.51MB  | NOT CACHED   | NOT CAC
HED        | PARQUET |...
| 2013  | 12    | 5   | -1    | 1      | 2.52MB  | NOT CACHED   | NOT CACHED
        | PARQUET |...
| Total |       |     | -1    | 5      | 12.58MB | 0B           |
        |         |...
+-------+-------+-----+-------+--------+---------+--------------+-----------
--------+---------+...

show column stats year_month_day;
+-----------+---------+-----------------+--------+----------+----------+
| Column    | Type    | #Distinct Values | #Nulls | Max Size | Avg Size |
```

```
+-----------+---------+------------------+--------+----------+----------+
| id        | INT     | -1               | -1     | 4        | 4        |
| val       | INT     | -1               | -1     | 4        | 4        |
| zfill     | STRING  | -1               | -1     | -1       | -1       |
| name      | STRING  | -1               | -1     | -1       | -1       |
| assertion | BOOLEAN | -1               | -1     | 1        | 1        |
| year      | INT     | 1                | 0      | 4        | 4        |
| month     | INT     | 1                | 0      | 4        | 4        |
| day       | INT     | 5                | 0      | 4        | 4        |
+-----------+---------+------------------+--------+----------+----------+

compute stats year_month_day;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 5 partition(s) and 5 column(s). |
+-----------------------------------------+

show table stats year_month_day;
+-------+-------+-----+--------+--------+---------+--------------+-------
------------+---------+...
| year  | month | day | #Rows  | #Files | Size    | Bytes Cached | Cache
Replication | Format  |...
+-------+-------+-----+--------+--------+---------+--------------+-------
------------+---------+...
| 2013  | 12    | 1   | 93606  | 1      | 2.51MB  | NOT CACHED   | NOT CA
CHED        | PARQUET |...
| 2013  | 12    | 2   | 94158  | 1      | 2.53MB  | NOT CACHED   | NOT CA
CHED        | PARQUET |...
| 2013  | 12    | 3   | 94122  | 1      | 2.52MB  | NOT CACHED   | NOT CA
CHED        | PARQUET |...
| 2013  | 12    | 4   | 93559  | 1      | 2.51MB  | NOT CACHED   | NOT CA
CHED        | PARQUET |...
| 2013  | 12    | 5   | 93845  | 1      | 2.52MB  | NOT CACHED   | NOT CA
CHED        | PARQUET |...
| Total |       |     | 469290 | 5      | 12.58MB | 0B           |
          |         |...
+-------+-------+-----+--------+--------+---------+--------------+-------
------------+---------+...

show column stats year_month_day;
+-----------+---------+------------------+--------+----------+------------
-------+
| Column    | Type    | #Distinct Values | #Nulls | Max Size | Avg Size
     |
+-----------+---------+------------------+--------+----------+--------------
-----+
| id        | INT     | 511129           | -1     | 4        | 4
      |
| val       | INT     | 364853           | -1     | 4        | 4
     |
| zfill     | STRING  | 311430           | -1     | 6        | 6
     |
| name      | STRING  | 471975           | -1     | 22       | 13.0016002655
0293 |
| assertion | BOOLEAN | 2                | -1     | 1        | 1
      |
| year      | INT     | 1                | 0      | 4        | 4
     |
| month     | INT     | 1                | 0      | 4        | 4
     |
| day       | INT     | 5                | 0      | 4        | 4
     |
```

```
+-----------+---------+-----------------+--------+----------+-----------
--------+
```

If you run the Hive statement ANALYZE TABLE COMPUTE STATISTICS FOR COLUMNS, Impala can only use the resulting column statistics if the table is unpartitioned. Impala cannot use Hive-generated column statistics for a partitioned table.

Detecting Missing Statistics

You can check whether a specific table has statistics using the SHOW TABLE STATS statement (for any table) or the SHOW PARTITIONS statement (for a partitioned table). Both statements display the same information. If a table or a partition does not have any statistics, the #Rows field contains -1. Once you compute statistics for the table or partition, the #Rows field changes to an accurate value.

The following example shows a table that initially does not have any statistics. The SHOW TABLE STATS statement displays different values for #Rows before and after the COMPUTE STATS operation.

```
[localhost:21000] > create table no_stats (x int);
[localhost:21000] > show table stats no_stats;
+-------+--------+------+--------------+--------+-------------------+
| #Rows | #Files | Size | Bytes Cached | Format | Incremental stats |
+-------+--------+------+--------------+--------+-------------------+
| -1    | 0      | 0B   | NOT CACHED   | TEXT   | false             |
+-------+--------+------+--------------+--------+-------------------+
[localhost:21000] > compute stats no_stats;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 1 column(s). |
+-----------------------------------------+
[localhost:21000] > show table stats no_stats;
+-------+--------+------+--------------+--------+-------------------+
| #Rows | #Files | Size | Bytes Cached | Format | Incremental stats |
+-------+--------+------+--------------+--------+-------------------+
| 0     | 0      | 0B   | NOT CACHED   | TEXT   | false             |
+-------+--------+------+--------------+--------+-------------------+
```

The following example shows a similar progression with a partitioned table. Initially, #Rows is -1. After a COMP UTE STATS operation, #Rows changes to an accurate value. Any newly added partition starts with no statistics, meaning that you must collect statistics after adding a new partition.

```
[localhost:21000] > create table no_stats_partitioned (x int) partitioned by
 (year smallint);
[localhost:21000] > show table stats no_stats_partitioned;
+-------+-------+--------+------+--------------+--------+-------------------
+
| year  | #Rows | #Files | Size | Bytes Cached | Format | Incremental stats
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
| Total | -1    | 0      | 0B   | 0B           |        |
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
[localhost:21000] > show partitions no_stats_partitioned;
+-------+-------+--------+------+--------------+--------+-------------------
+
| year  | #Rows | #Files | Size | Bytes Cached | Format | Incremental stats
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
```

```
| Total | -1    | 0      | 0B   | 0B           |        |
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
[localhost:21000] > alter table no_stats_partitioned add partition (year=
2013);
[localhost:21000] > compute stats no_stats_partitioned;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 1 column(s). |
+-----------------------------------------+
[localhost:21000] > alter table no_stats_partitioned add partition (year=
2014);
[localhost:21000] > show partitions no_stats_partitioned;
+-------+-------+--------+------+--------------+--------+-------------------
+
| year  | #Rows | #Files | Size | Bytes Cached | Format | Incremental stats
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
| 2013  | 0     | 0      | 0B   | NOT CACHED   | TEXT   | false
 |
| 2014  | -1    | 0      | 0B   | NOT CACHED   | TEXT   | false
 |
| Total | 0     | 0      | 0B   | 0B           |        |
 |
+-------+-------+--------+------+--------------+--------+-------------------
+
```

Note: Because the default COMPUTE STATS statement creates and updates statistics for all partitions in a table, if you expect to frequently add new partitions, use the COMPUTE INCREMENTAL STATS syntax instead, which lets you compute stats for a single specified partition, or only for those partitions that do not already have incremental stats.

If checking each individual table is impractical, due to a large number of tables or views that hide the underlying base tables, you can also check for missing statistics for a particular query. Use the EXPLAIN statement to preview query efficiency before actually running the query. Use the query profile output available through the PROFILE command in `impala-shell` or the web UI to verify query execution and timing after running the query. Both the EXPLAIN plan and the PROFILE output display a warning if any tables or partitions involved in the query do not have statistics.

```
[localhost:21000] > create table no_stats (x int);
[localhost:21000] > explain select count(*) from no_stats;
+-----------------------------------------------------------------------
-----------+
| Explain String
         |
+-----------------------------------------------------------------------
---------+
| Estimated Per-Host Requirements: Memory=10.00MB VCores=1
          |
| WARNING: The following tables are missing relevant table and/or column s
tatistics. |
| incremental_stats.no_stats
         |
|
         |
| 03:AGGREGATE [FINALIZE]
          |
| |  output: count:merge(*)
         |
| |
         |
```

```
| 02:EXCHANGE [UNPARTITIONED]
         |
| |
          |
| 01:AGGREGATE
         |
| |  output: count(*)
         |
| |
         |
| 00:SCAN HDFS [incremental_stats.no_stats]
          |
|    partitions=1/1 files=0 size=0B
         |
+-----------------------------------------------------------------------
----------+
```

Because Impala uses the *partition pruning* technique when possible to only evaluate certain partitions, if you have a partitioned table with statistics for some partitions and not others, whether or not the EXPLAIN statement shows the warning depends on the actual partitions used by the query. For example, you might see warnings or not for different queries against the same table:

```
-- No warning because all the partitions for the year 2012 have stats.
EXPLAIN SELECT ... FROM t1 WHERE year = 2012;

-- Missing stats warning because one or more partitions in this range
-- do not have stats.
EXPLAIN SELECT ... FROM t1 WHERE year BETWEEN 2006 AND 2009;
```

To confirm if any partitions at all in the table are missing statistics, you might explain a query that scans the entire table, such as SELECT COUNT(*) FROM *TABLE_NAME*.

Manually Setting Table Statistics with ALTER TABLE

The most crucial piece of data in all the statistics is the number of rows in the table (for an unpartitioned or partitioned table) and for each partition (for a partitioned table). The COMPUTE STATS statement always gathers statistics about all columns, as well as overall table statistics. If it is not practical to do a full COMPUTE STATS or COMPUTE INCREMENTAL STATS operation after adding a partition or inserting data, or if you can see that Impala would produce a more efficient plan if the number of rows was different, you can manually set the number of rows through an ALTER TABLE statement:

```
-- Set total number of rows. Applies to both unpartitioned and partitioned
tables.
alter table TABLE_NAME set tblproperties('numRows'='NEW_VALUE', 'STATS_GE
NERATED_VIA_STATS_TASK'='true');

-- Set total number of rows for a specific partition. Applies to partitioned
 tables only.
-- You must specify all the partition key columns in the PARTITION clause.
alter table TABLE_NAME partition (KEYCOL1=VAL1,KEYCOL2=VAL2...) set tblprop
erties('numRows'='NEW_VALUE', 'STATS_GENERATED_VIA_STATS_TASK'='true');
```

This statement avoids re-scanning any data files. For example:

```
alter table analysis_data set tblproperties('numRows'='1001000000', 'STATS_
GENERATED_VIA_STATS_TASK'='true');
```

For a partitioned table, update both the per-partition number of rows and the number of rows for the whole table. For example:

```
alter table partitioned_data partition(year=2009, month=4) set tblproperties
 ('numRows'='30000', 'STATS_GENERATED_VIA_STATS_TASK'='true');
alter table partitioned_data set tblproperties ('numRows'='1030000', 'STA
TS_GENERATED_VIA_STATS_TASK'='true');
```

In practice, the COMPUTE STATS statement, or COMPUTE INCREMENTAL STATS for a partitioned table, should be fast and convenient enough that this technique is only useful for the very largest partitioned tables. Because the column statistics might be left in a stale state, do not use this technique as a replacement for COMPUTE STATS. Only use this technique if all other means of collecting statistics are impractical, or as a low-overhead operation that you run in between periodic COMPUTE STATS or COMPUTE INCREMENTAL STATS operations.

Manually Setting Column Statistics with ALTER TABLE

You can use the SET COLUMN STATS clause of ALTER TABLE to manually set or change column statistics. Only use this technique in cases where it is impractical to run COMPUTE STATS or COMPUTE INCREMENTAL ST ATS frequently enough to keep up with data changes for a huge table.

You specify a case-insensitive symbolic name for the kind of statistics: numDVs, numNulls, avgSize, maxSize. The key names and values are both quoted. This operation applies to an entire table, not a specific partition.

For example:

```
alter table t1 set column stats x ('numDVs'='2','numNulls'='0');
```

Examples of Using Table and Column Statistics with Impala

The following examples walk through a sequence of SHOW TABLE STATS, SHOW COLUMN STATS, ALTE R TABLE, and SELECT and INSERT statements to illustrate various aspects of how Impala uses statistics to help optimize queries.

This example shows table and column statistics for the STORE column used in the TPC-DS benchmarks for decision support systems. It is a tiny table holding data for 12 stores. Initially, before any statistics are gathered by a COMP UTE STATS statement, most of the numeric fields show placeholder values of -1, indicating that the figures are unknown. The figures that are filled in are values that are easily countable or deducible at the physical level, such as the number of files, total data size of the files, and the maximum and average sizes for data types that have a constant size such as INT, FLOAT, and TIMESTAMP.

```
[localhost:21000] > show table stats store;
+-------+--------+--------+--------+
| #Rows | #Files | Size   | Format |
+-------+--------+--------+--------+
| -1    | 1      | 3.08KB | TEXT   |
+-------+--------+--------+--------+
Returned 1 row(s) in 0.03s
[localhost:21000] > show column stats store;
+--------------------+-----------+------------------+--------+----------+--
--------+
| Column             | Type      | #Distinct Values | #Nulls | Max Size |
Avg Size |
+--------------------+-----------+------------------+--------+----------+
----------+
| s_store_sk         | INT       | -1               | -1     | 4        | 4
       |
| s_store_id         | STRING    | -1               | -1     | -1       | -1
       |
| s_rec_start_date   | TIMESTAMP | -1               | -1     | 16       | 16
       |
```

```
| s_rec_end_date     | TIMESTAMP | -1                | -1     | 16       |
 16       |
| s_closed_date_sk   | INT       | -1                | -1     | 4        | 4
       |
| s_store_name       | STRING    | -1                | -1     | -1       | -1
       |
| s_number_employees | INT       | -1                | -1     | 4        | 4
       |
| s_floor_space      | INT       | -1                | -1     | 4        | 4
       |
| s_hours            | STRING    | -1                | -1     | -1       | -1
       |
| s_manager          | STRING    | -1                | -1     | -1       | -1
       |
| s_market_id        | INT       | -1                | -1     | 4        | 4
       |
| s_geography_class  | STRING    | -1                | -1     | -1       |
 -1       |
| s_market_desc      | STRING    | -1                | -1     | -1       | -1
       |
| s_market_manager   | STRING    | -1                | -1     | -1       | -1
       |
| s_division_id      | INT       | -1                | -1     | 4        | 4
       |
| s_division_name    | STRING    | -1                | -1     | -1       |
 -1       |
| s_company_id       | INT       | -1                | -1     | 4        | 4
       |
| s_company_name     | STRING    | -1                | -1     | -1       | -1
       |
| s_street_number    | STRING    | -1                | -1     | -1       | -1
       |
| s_street_name      | STRING    | -1                | -1     | -1       |
 -1       |
| s_street_type      | STRING    | -1                | -1     | -1       | -1
       |
| s_suite_number     | STRING    | -1                | -1     | -1       | -1
       |
| s_city             | STRING    | -1                | -1     | -1       | -1
       |
| s_county           | STRING    | -1                | -1     | -1       |
 -1       |
| s_state            | STRING    | -1                | -1     | -1       | -1
       |
| s_zip              | STRING    | -1                | -1     | -1       | -1
       |
| s_country          | STRING    | -1                | -1     | -1       | -1
       |
| s_gmt_offset       | FLOAT     | -1                | -1     | 4        | 4
       |
| s_tax_percentage   | FLOAT     | -1                | -1     | 4        | 4
       |
+--------------------+-----------+------------------+--------+----------+--
--------+
Returned 29 row(s) in 0.04s
```

With the Hive ANALYZE TABLE statement for column statistics, you had to specify each column for which to gather statistics. The Impala COMPUTE STATS statement automatically gathers statistics for all columns, because it reads through the entire table relatively quickly and can efficiently compute the values for all the columns. This example shows how after running the COMPUTE STATS statement, statistics are filled in for both the table and all its columns:

```
[localhost:21000] > compute stats store;
```

```
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 29 column(s). |
+-----------------------------------------+
Returned 1 row(s) in 1.88s
[localhost:21000] > show table stats store;
+-------+--------+--------+--------+
| #Rows | #Files | Size   | Format |
+-------+--------+--------+--------+
| 12    | 1      | 3.08KB | TEXT   |
+-------+--------+--------+--------+
Returned 1 row(s) in 0.02s
[localhost:21000] > show column stats store;
+--------------------+-----------+------------------+--------+----------+-
------------------+
| Column             | Type      | #Distinct Values | #Nulls | Max Size | Av
g Size          |
+--------------------+-----------+------------------+--------+----------+-
------------------+
| s_store_sk         | INT       | 12               | -1     | 4        | 4
               |
| s_store_id         | STRING    | 6                | -1     | 16       | 16
               |
| s_rec_start_date   | TIMESTAMP | 4                | -1     | 16       | 16
               |
| s_rec_end_date     | TIMESTAMP | 3                | -1     | 16       | 16
               |
| s_closed_date_sk   | INT       | 3                | -1     | 4        | 4
               |
| s_store_name       | STRING    | 8                | -1     | 5        |
4.25              |
| s_number_employees | INT       | 9                | -1     | 4        | 4
               |
| s_floor_space      | INT       | 10               | -1     | 4        | 4
               |
| s_hours            | STRING    | 2                | -1     | 8        | 7.
083300113677979 |
| s_manager          | STRING    | 7                | -1     | 15       | 12
               |
| s_market_id        | INT       | 7                | -1     | 4        | 4
               |
| s_geography_class  | STRING    | 1                | -1     | 7        | 7
               |
| s_market_desc      | STRING    | 10               | -1     | 94       | 55
.5             |
| s_market_manager   | STRING    | 7                | -1     | 16       | 14
               |
| s_division_id      | INT       | 1                | -1     | 4        | 4
               |
| s_division_name    | STRING    | 1                | -1     | 7        | 7
               |
| s_company_id       | INT       | 1                | -1     | 4        | 4
               |
| s_company_name     | STRING    | 1                | -1     | 7        | 7
               |
| s_street_number    | STRING    | 9                | -1     | 3        | 2.
833300113677979 |
| s_street_name      | STRING    | 12               | -1     | 11       |
6.583300113677979 |
| s_street_type      | STRING    | 8                | -1     | 9        | 4.
833300113677979 |
| s_suite_number     | STRING    | 11               | -1     | 9        |
8.25              |
```

```
| s_city             | STRING    | 2                 | -1     | 8        | 6.
5                |
| s_county           | STRING    | 1                 | -1     | 17       | 17
                 |
| s_state            | STRING    | 1                 | -1     | 2        | 2
                 |
| s_zip              | STRING    | 2                 | -1     | 5        | 5
                 |
| s_country          | STRING    | 1                 | -1     | 13       | 13
                 |
| s_gmt_offset       | FLOAT     | 1                 | -1     | 4        | 4
                 |
| s_tax_percentage   | FLOAT     | 5                 | -1     | 4        | 4
                 |
+--------------------+-----------+------------------+--------+----------+-
------------------+
Returned 29 row(s) in 0.04s
```

The following example shows how statistics are represented for a partitioned table. In this case, we have set up a table to hold the world's most trivial census data, a single STRING field, partitioned by a YEAR column. The table statistics include a separate entry for each partition, plus final totals for the numeric fields. The column statistics include some easily deducible facts for the partitioning column, such as the number of distinct values (the number of partition subdirectories).

```
localhost:21000] > describe census;
+------+----------+---------+
| name | type     | comment |
+------+----------+---------+
| name | string   |         |
| year | smallint |         |
+------+----------+---------+
Returned 2 row(s) in 0.02s
[localhost:21000] > show table stats census;
+-------+-------+--------+------+---------+
| year  | #Rows | #Files | Size | Format  |
+-------+-------+--------+------+---------+
| 2000  | -1    | 0      | 0B   | TEXT    |
| 2004  | -1    | 0      | 0B   | TEXT    |
| 2008  | -1    | 0      | 0B   | TEXT    |
| 2010  | -1    | 0      | 0B   | TEXT    |
| 2011  | 0     | 1      | 22B  | TEXT    |
| 2012  | -1    | 1      | 22B  | TEXT    |
| 2013  | -1    | 1      | 231B | PARQUET |
| Total | 0     | 3      | 275B |         |
+-------+-------+--------+------+---------+
Returned 8 row(s) in 0.02s
[localhost:21000] > show column stats census;
+--------+----------+------------------+--------+----------+----------+
| Column | Type     | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+----------+------------------+--------+----------+----------+
| name   | STRING   | -1               | -1     | -1       | -1       |
| year   | SMALLINT | 7                | -1     | 2        | 2        |
+--------+----------+------------------+--------+----------+----------+
Returned 2 row(s) in 0.02s
```

The following example shows how the statistics are filled in by a COMPUTE STATS statement in Impala.

```
[localhost:21000] > compute stats census;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 3 partition(s) and 1 column(s). |
```

```
+-----------------------------------------+
Returned 1 row(s) in 2.16s
[localhost:21000] > show table stats census;
+-------+-------+--------+------+---------+
| year  | #Rows | #Files | Size | Format  |
+-------+-------+--------+------+---------+
| 2000  | -1    | 0      | 0B   | TEXT    |
| 2004  | -1    | 0      | 0B   | TEXT    |
| 2008  | -1    | 0      | 0B   | TEXT    |
| 2010  | -1    | 0      | 0B   | TEXT    |
| 2011  | 4     | 1      | 22B  | TEXT    |
| 2012  | 4     | 1      | 22B  | TEXT    |
| 2013  | 1     | 1      | 231B | PARQUET |
| Total | 9     | 3      | 275B |         |
+-------+-------+--------+------+---------+
Returned 8 row(s) in 0.02s
[localhost:21000] > show column stats census;
+--------+----------+------------------+--------+----------+----------+
| Column | Type     | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+----------+------------------+--------+----------+----------+
| name   | STRING   | 4                | -1     | 5        | 4.5      |
| year   | SMALLINT | 7                | -1     | 2        | 2        |
+--------+----------+------------------+--------+----------+----------+
Returned 2 row(s) in 0.02s
```

You can see how Impala executes a query differently in each case by observing the EXPLAIN output before and after collecting statistics. Measure the before and after query times, and examine the throughput numbers in before and after SUMMARY or PROFILE output, to verify how much the improved plan speeds up performance.

Generating Table and Column Statistics

Use the COMPUTE STATS statement to collect table and column statistics. The COMPUTE STATS variants offer different tradeoffs between computation cost, staleness, and maintenance workflows.

Important:

For a particular table, use either COMPUTE STATS or COMPUTE INCREMENTAL STATS, but never combine the two or alternate between them. If you switch from COMPUTE STATS to COMPUTE INCREMENTAL STATS during the lifetime of a table, or vice versa, drop all statistics by running DROP STATS before making the switch.

COMPUTE STATS

The COMPUTE STATS command collects and sets the table-level and partition-level row counts as well as all column statistics for a given table. The collection process is CPU-intensive and can take a long time to complete for very large tables.

To speed up COMPUTE STATS consider the following options which can be combined.

- Limit the number of columns for which statistics are collected to increase the efficiency of COMPUTE STATS. Queries benefit from statistics for those columns involved in filters, join conditions, group by or partition by clauses. Other columns are good candidates to exclude from COMPUTE STATS. This feature is available since Impala 2.12.
- Set the MT_DOP query option to use more threads within each participating impalad to compute the statistics faster - but not more efficiently. Note that computing stats on a large table with a high MT_DOP value can negatively affect other queries running at the same time if the COMPUTE STATS claims most CPU cycles.
- Consider the experimental extrapolation and sampling features (see below) to further increase the efficiency of computing stats.

COMPUTE STATS is intended to be run periodically, e.g. weekly, or on-demand when the contents of a table have changed significantly. Due to the high resource utilization and long response time of COMPUTE STATS, it is most practical to run it in a scheduled maintenance window where the Impala cluster is idle enough to accommodate

the expensive operation. The degree of change that qualifies as “significant” depends on the query workload, but typically, if 30% of the rows have changed then it is recommended to recompute statistics.

If you reload a complete new set of data for a table, but the number of rows and number of distinct values for each column is relatively unchanged from before, you do not need to recompute stats for the table.

Extrapolation and Sampling

Impala supports extrapolation and sampling to alleviate the following common issues for computing and maintaining statistics on very large tables:

- Newly added partitions do not have row count statistics. Table scans that only access those new partitions are treated as not having stats. Similarly, table scans that access both new and old partitions estimate the scan cardinality based on those old partitions that have stats, and the new partitions without stats are treated as having 0 rows.
- The row counts of existing partitions become stale when data is added or dropped.
- Computing stats for tables with a 100,000 or more partitions might fail or be very slow due to the high cost of updating the partition metadata in the Hive Metastore.
- With transient compute resources it is important to minimize the time from starting a new cluster to successfully running queries. Since the cluster might be relatively short-lived, users might prefer to quickly collect stats that are "good enough" as opposed to spending a lot of time and resources on computing full-fidelity stats.

For very large tables, it is often wasteful or impractical to run a full COMPUTE STATS to address the scenarios above on a frequent basis.

The sampling feature makes COMPUTE STATS more efficient by processing a fraction of the table data, and the extrapolation feature aims to reduce the frequency at which COMPUTE STATS needs to be re-run by estimating the row count of new and modified partitions.

The sampling and extrapolation features are disabled by default. They can be enabled globally or for specific tables, as follows.

- Set the impalad start-up configuration --enable_stats_extrapolation to enable the features globally.
- To enable them only for a specific table, set the impala.enable.stats.extrapolation table property to true for the table. The table-level property overrides the global setting, so it is also possible to enable sampling and extrapolation globally, but disable it for specific tables by setting the table property to false. For example:

```
ALTER TABLE mytable test_table SET TBLPROPERTIES("impala.enable.stats.ex
trapolation"="true");
```

Note: Why are these features experimental? Due to their probabilistic nature it is possible that these features perform pathologically poorly on tables with extreme data/file/size distributions. Since it is not feasible for us to test all possible scenarios we only cautiously advertise these new capabilities. That said, the features have been thoroughly tested and are considered functionally stable. If you decide to give these features a try, please tell us about your experience at user@impala.apache.org! We rely on user feedback to guide future improvements in statistics collection.

Stats Extrapolation

The main idea of stats extrapolation is to estimate the row count of new and modified partitions based on the result of the last COMPUTE STATS. Enabling stats extrapolation changes the behavior of COMPUTE STATS, as well as the cardinality estimation of table scans. COMPUTE STATS no longer computes and stores per-partition row counts, and instead, only computes a table-level row count together with the total number of file bytes in the table at that time. No partition metadata is modified. The input cardinality of a table scan is estimated by converting the data volume of relevant partitions to a row count, based on the table-level row count and file bytes statistics. It is assumed that within the same table, different sets of files with the same data volume correspond to the similar number of rows on average. With extrapolation enabled, the scan cardinality estimation ignores per-partition row counts. It only relies on the table-level statistics and the scanned data volume.

The SHOW TABLE STATS and EXPLAIN commands distinguish between row counts stored in the Hive Metastore, and the row counts extrapolated based on the above process.

Sampling

A TABLESAMPLE clause may be added to COMPUTE STATS to limit the percentage of data to be processed. The final statistics are obtained by extrapolating the statistics from the data sample over the entire table. The extrapolated statistics are stored in the Hive Metastore, just as if no sampling was used. The following example runs COMPUTE STATS over a 10 percent data sample.

```
COMPUTE STATS test_table TABLESAMPLE SYSTEM(10) ;
```

We have found that a 10 percent sampling rate typically offers a good tradeoff between statistics accuracy and execution cost. A sampling rate well below 10 percent has shown poor results and is not recommended.

Important: Sampling-based techniques sacrifice result accuracy for execution efficiency, so your mileage may vary for different tables and columns depending on their data distribution. The extrapolation procedure Impala uses for estimating the number of distinct values per column is inherently non-detetministic, so your results may even vary between runs of COMPUTE STATS TABLESAMPLE, even if no data has changed.

COMPUTE INCREMENTAL STATS

In Impala 2.1.0 and higher, you can use the COMPUTE INCREMENTAL STATS and DROP INCREMENTAL STATS commands. The INCREMENTAL clauses work with incremental statistics, a specialized feature for partitioned tables.

When you compute incremental statistics for a partitioned table, by default Impala only processes those partitions that do not yet have incremental statistics. By processing only newly added partitions, you can keep statistics up to date without incurring the overhead of reprocessing the entire table each time.

You can also compute or drop statistics for a specified subset of partitions by including a PARTITION clause in the COMPUTE INCREMENTAL STATS or DROP INCREMENTAL STATS statement.

Important:

When you run COMPUTE INCREMENTAL STATS on a table for the first time, the statistics are computed again from scratch regardless of whether the table already has statistics. Therefore, expect a one-time resource-intensive operation for scanning the entire table when running COMPUTE INCREMENTAL STATS for the first time on a given table.

The metadata for incremental statistics is handled differently from the original style of statistics:

- Issuing a COMPUTE INCREMENTAL STATS without a partition clause causes Impala to compute incremental stats for all partitions that do not already have incremental stats. This might be the entire table when running the command for the first time, but subsequent runs should only update new partitions. You can force updating a partition that already has incremental stats by issuing a DROP INCREMENTAL STATS before running COMPUTE INCREMENTAL STATS.
- The SHOW TABLE STATS and SHOW PARTITIONS statements now include an additional column showing whether incremental statistics are available for each column. A partition could already be covered by the original type of statistics based on a prior COMPUTE STATS statement, as indicated by a value other than -1 under the #Rows column. Impala query planning uses either kind of statistics when available.
- COMPUTE INCREMENTAL STATS takes more time than COMPUTE STATS for the same volume of data. Therefore it is most suitable for tables with large data volume where new partitions are added frequently, making it impractical to run a full COMPUTE STATS operation for each new partition. For unpartitioned tables, or partitioned tables that are loaded once and not updated with new partitions, use the original COMPUTE STATS syntax.
- COMPUTE INCREMENTAL STATS uses some memory in the `catalogd` process, proportional to the number of partitions and number of columns in the applicable table. The memory overhead is approximately 400 bytes for

each column in each partition. This memory is reserved in the `catalogd` daemon, the `statestored` daemon, and in each instance of the `impalad` daemon.

- In cases where new files are added to an existing partition, issue a REFRESH statement for the table, followed by a DROP INCREMENTAL STATS and COMPUTE INCREMENTAL STATS sequence for the changed partition.
- The DROP INCREMENTAL STATS statement operates only on a single partition at a time. To remove statistics (whether incremental or not) from all partitions of a table, issue a DROP STATS statement with no INCREMENTAL or PARTITION clauses.

The following considerations apply to incremental statistics when the structure of an existing table is changed (known as *schema evolution*):

- If you use an ALTER TABLE statement to drop a column, the existing statistics remain valid and COMPUTE INCREMENTAL STATS does not rescan any partitions.
- If you use an ALTER TABLE statement to add a column, Impala rescans all partitions and fills in the appropriate column-level values the next time you run COMPUTE INCREMENTAL STATS.
- If you use an ALTER TABLE statement to change the data type of a column, Impala rescans all partitions and fills in the appropriate column-level values the next time you run COMPUTE INCREMENTAL STATS.
- If you use an ALTER TABLE statement to change the file format of a table, the existing statistics remain valid and a subsequent COMPUTE INCREMENTAL STATS does not rescan any partitions.

Runtime Filtering

Runtime filtering is a wide-ranging optimization feature available in Impala. When only a fraction of the data in a table is needed for a query against a partitioned table or to evaluate a join condition, Impala determines the appropriate conditions while the query is running, and broadcasts that information to all the `impalad` nodes that are reading the table so that they can avoid unnecessary I/O to read partition data, and avoid unnecessary network transmission by sending only the subset of rows that match the join keys across the network.

Runtime filtering is primarily used:

- To optimize queries against large partitioned tables (under the name *dynamic partition pruning*)
- To optimize joins of large tables

The following terms are used in this topic to describe runtime filtering.

plan fragment

Impala decomposes each query into smaller units of work that are distributed across the cluster. Wherever possible, a data block is read, filtered, and aggregated by plan fragments executing on the same host. For some operations, such as joins and combining intermediate results into a final result set, data is transmitted across the network from one Impala daemon to another.

SCAN and HASH JOIN plan nodes

In the Impala query plan, a *scan node* performs the I/O to read from the underlying data files. Although this is an expensive operation from the traditional database perspective, Hadoop clusters and Impala are optimized to do this kind of I/O in a highly parallel fashion. The major potential cost savings come from using the columnar Parquet format (where Impala can avoid reading data for unneeded columns) and partitioned tables (where Impala can avoid reading data for unneeded partitions).

Most Impala joins use the *hash join* mechanism. (It is only fairly recently that Impala started using the nested-loop join technique, for certain kinds of non-equijoin queries.) In a hash join, when evaluating join conditions from two tables, Impala constructs a hash table in memory with all the different column values from the table on one side of the join. Then, for each row from the table on the other side of the join, Impala tests whether the relevant column values are in this hash table or not.

hash join

In a hash join, when evaluating join conditions from two tables, Impala constructs a hash table in memory with all the different column values from the table on one side of the join. Then, for each row from the table on the other side of the join, Impala tests whether the relevant column values are in this hash table or not.

- A *hash join node* constructs such an in-memory hash table, then performs the comparisons to identify which rows match the relevant join conditions and should be included in the result set (or at least sent on to the subsequent intermediate stage of query processing). Because some of the input for a hash join might be transmitted across the network from another host, it is especially important from a performance perspective to prune out ahead of time any data that is known to be irrelevant.

  The more distinct values are in the columns used as join keys, the larger the in-memory hash table and thus the more memory required to process the query.

broadcast join vs shuffle join

In a broadcast join, the table from one side of the join (typically the smaller table) is sent in its entirety to all the hosts involved in the query. Then each host can compare its portion of the data from the other (larger) table against the full set of possible join keys. In a shuffle join, there is no obvious “smaller” table, and so the contents of both tables are divided up, and corresponding portions of the data are transmitted to each host involved in the query.

A shuffle join is sometimes referred to in Impala as a *partitioned join*.

build and probe phases

When Impala processes a join query, the *build phase* is where the rows containing the join key columns, typically for the smaller table, are transmitted across the network and built into an in-memory hash table data structure on one or more destination nodes. The *probe phase* is where data is read locally (typically from the larger table) and the join key columns are compared to the values in the in-memory hash table. The corresponding input sources (tables, subqueries, and so on) for these phases are referred to as the *build side* and the *probe side*.

Runtime Filters

The *filter* that is transmitted between plan fragments is essentially a list of values for join key columns. When this list of values is transmitted in time to a scan node, Impala can filter out non-matching values immediately after reading them, rather than transmitting the raw data to another host to compare against the in-memory hash table on that host.

Impala supports the following types of filters based on the payload:

- *Bloom filter*: For HDFS-based tables, the Bloom filter uses a probability-based algorithm to determine all possible matching values. The probability-based aspects means that the filter might include some non-matching values, but if so, that does not cause any inaccuracy in the final results.
- *Min-max filter*: The filter is a data structure representing a minimum and maximum value. These filters are passed to Kudu to reduce the number of rows returned to Impala when scanning the probe side of the join. This filter currently only applies to Kudu tables.

Based on how filters from all join instances are aggregated, each of the above filters can be categorized as one of the following:

- *Broadcast filter*: A broadcast filter reflects the complete list of relevant values and can be immediately evaluated by a scan node.

  Broadcast filters are also classified as local or global. With a local broadcast filter, the information in the filter is used by a subsequent query fragment that is running on the same host that produced the filter. A non-local broadcast filter must be transmitted across the network to a query fragment that is running on a different host. Impala designates 3 hosts to each produce non-local broadcast filters, to guard against the possibility of a single slow host taking too long. Depending on the setting of the RUNTIME_FILTER_MODE query option (LOCAL or GLOBAL), Impala either uses a conservative optimization strategy where filters are only consumed on the same host that produced them, or a more aggressive strategy where filters are eligible to be transmitted across the network. The default for runtime filtering is the GLOBAL setting.

- *Partitioned filter*: A partitioned filter reflects only the values processed by one host in the cluster; all the partitioned filters must be combined into one (by the coordinator node) before the scan nodes can use the results to accurately filter the data as it is read from storage.

File Format Considerations for Runtime Filtering

Parquet tables get the most benefit from the runtime filtering optimizations. Runtime filtering can speed up join queries against partitioned or unpartitioned Parquet tables, and single-table queries against partitioned Parquet tables.

For other file formats (text, Avro, RCFile, and SequenceFile), runtime filtering speeds up queries against partitioned tables only. Because partitioned tables can use a mixture of formats, Impala produces the filters in all cases, even if they are not ultimately used to optimize the query.

Wait Intervals for Runtime Filters

Because it takes time to produce runtime filters, especially for partitioned filters that must be combined by the coordinator node, there is a time interval above which it is more efficient for the scan nodes to go ahead and construct their intermediate result sets, even if that intermediate data is larger than optimal. If it only takes a few seconds to produce the filters, it is worth the extra time if pruning the unnecessary data can save minutes in the overall query time. You can specify the maximum wait time in milliseconds using the RUNTIME_FILTER_WAIT_TIME_MS query option.

By default, each scan node waits for up to 1 second (1000 milliseconds) for filters to arrive. If all filters have not arrived within the specified interval, the scan node proceeds, using whatever filters did arrive to help avoid reading unnecessary data. If a filter arrives after the scan node begins reading data, the scan node applies that filter to the data that is read after the filter arrives, but not to the data that was already read.

If the cluster is relatively busy and your workload contains many resource-intensive or long-running queries, consider increasing the wait time so that complicated queries do not miss opportunities for optimization. If the cluster is lightly loaded and your workload contains many small queries taking only a few seconds, consider decreasing the wait time to avoid the 1 second delay for each query.

Query Options for Runtime Filtering

The following query options control runtime filtering.

- RUNTIME_FILTER_MODE

  This query option controls how extensively the filters are transmitted between hosts. By default, it is set to the highest level (GLOBAL).
- The other query options are tuning knobs that you typically only adjust after doing performance testing, and that you might want to change only for the duration of a single expensive query.
  - MAX_NUM_RUNTIME_FILTERS
  - DISABLE_ROW_RUNTIME_FILTERING
  - RUNTIME_FILTER_MAX_SIZE
  - RUNTIME_FILTER_MIN_SIZE
  - RUNTIME_BLOOM_FILTER_SIZE

Runtime Filtering and Query Plans

In the same way the query plan displayed by the EXPLAIN statement includes information about predicates used by each plan fragment, it also includes annotations showing whether a plan fragment produces or consumes a runtime filter.

- A plan fragment that produces a filter includes an annotation such as runtime filters: *FILTER_ID* <- *TABLE.COLUMN*
- A plan fragment that consumes a filter includes an annotation such as runtime filters: *FILTER_ID* -> *TABLE.COLUMN*

Setting the query option EXPLAIN_LEVEL=2 adds additional annotations showing the type of the filter:

- *FILTER_ID*[bloom] (for HDFS-based tables)
- *FILTER_ID*[min_max] (for Kudu tables)

The query profile (displayed by the PROFILE command in `impala-shell`) contains both the EXPLAIN plan and more detailed information about the internal workings of the query. The profile output includes the Filter routing table section with information about each filter based on its ID.

Tuning and Troubleshooting Queries that Use Runtime Filtering

These tuning and troubleshooting procedures apply to queries that are resource-intensive enough, long-running enough, and frequent enough that you can devote special attention to optimizing them individually.

Use the EXPLAIN statement and examine the runtime filters: lines to determine whether runtime filters are being applied to the WHERE predicates and join clauses that you expect. For example, runtime filtering does not apply to queries that use the nested loop join mechanism due to non-equijoin operators.

Make sure statistics are up-to-date for all tables involved in the queries. Use the COMPUTE STATS statement after loading data into non-partitioned tables, and COMPUTE INCREMENTAL STATS after adding new partitions to partitioned tables.

If join queries involving large tables use unique columns as the join keys, for example joining a primary key column with a foreign key column, the overhead of producing and transmitting the filter might outweigh the performance benefit because not much data could be pruned during the early stages of the query. For such queries, consider setting the query option RUNTIME_FILTER_MODE=OFF.

Min/Max Filtering

In addition to filtering at the partition level, the Parquet file format supports filtering at three levels: row group, page and row levels based on the minimum and maximum values of row group or page, and the value of the row in the file. These minimum/maximum column values are stored in the footer of the file. If the range between the minimum and maximum value in the file does not overlap with the range of data specified by the query, then the system skips the row group, page or row during scans.

Min/max filters are now enabled by default for equi-joins on lexical (leading column) or Z-Order (all columns) sorted columns in a Parquet table created by Impala. When such a sorted column is also the equi- join column, the min/max values in the filter are computed from the hash table build side and applied to the sorted column. In this way, the user can take advantage of Impala sorting the min/max values in column index in each data file for the table by quickly eliminating non-matching row groups, pages or rows. The query option minmax_filter_sorted_columns which defaults to true can be used to control the feature. When minmax_filter_sorted_columns is true, the patch will generate min/max filters only for the sort columns. The normal control knobs minmax_filter_threshold (for threshold) and minmax_filtering_level (for filtering level) will also continue to work. When the threshold is 0, the patch automatically assigns a reasonable value for the threshold, and selects PAGE to be the filtering level.

In the backend, the skipped pages are quickly found by taking a fast code path to identify the corresponding lower and the upper bounds in the sorted min and max value arrays, given a range in the filter. The skipped pages are expressed as page ranges which are translated into row ranges later on.

A new query option minmax_filter_fast_code_path is added to control the work of the fast code path. This option can be assigned one of these three values ON (default), OFF, or VERIFICATION. The last value helps verify that the results from both the fast and the regular code path are the same.

The following example shows how min/max filtering on leading sort-by columns improves the performance of scan operators greatly.

```
select straight_join a.l_orderkey from
    simpflified_lineitem a join [SHUFFLE] tpch_parquet.lineitem b
    where a.l_orderkey = b.l_orderkey and b.l_receiptdate = "1998-12-31"
```

When this query is run on pages containing no more than 24000 rows, the results with different filtering options are as shown here:

- 84.62ms (page level filtering)
- 115.27ms (row group level filtering)
- 137.14ms (no filtering)

Support reading and writing Parquet bloom filters

Bloom filter is a performance optimization feature now available in Impala. This filter tells you, rapidly and memory-efficiently, whether the data you are looking for is present in a file.

Impala determines the appropriate conditions while the query is running. Impala can now read and write Parquet bloom filters. However, bloom filters can also provide false positives. If a bloom filter evaluates to:

- true: data might be present in the data file or might not be present.
- false: data is not present.

Currently, bloom filters are per column chunk entries which implies that you can skip entire row groups based on the filter. Writing a bloom filter is not useful for dictionary encoded columns, as all distinct values are included in the dictionary and the dictionary can give exact results in filtering with the predicate. If no value passes the predicate the whole row group can be skipped.

For example, if there is a predicate 'WHERE col = some_value' and some_value is not in the bloom filter the row group will be discarded.

Bloom filters support reading and writing columns with the following data types: integers, float, double, and Impala strings. Reading does not need any intervention from Impala, however, writing can be controlled by a new query option parquet_bloom_filter_write and the table property parquet.bloom.filter.columns.

The following table contains the list of the supported data types.

Parquet type	Impala type
INT32	TINYINT, SMALLINT, INT
INT64	BIGINT
FLOAT	FLOAT
DOUBLE	DOUBLE
BYTE_ARRAY	STRING

Query option

The query option for writing Parquet bloom filters (parquet_bloom_filter_write) accepts any of the following values:

- NEVER - never write Parquet bloom filters.
- IF_NO_DICT - write Parquet bloom filters if specified in the table properties AND if the row group is not fully dictionary encoded (the number of distinct values exceeds the maximum dictionary size); the row group may still be partially dictionary encoded, in which case the bloom filter contains all values from the whole row group, including those that are present in the dictionary.
- ALWAYS - always write Parquet bloom filters if specified in the table properties, even if the row group is fully dictionary encoded.

Table setting

The parquet.bloom.filter.columns table property is a comma separated list of 'col_name:bytes' pairs.

Where:

col_name is the name of the column for which a bloom filter should be written;

bytes represents the size (in bytes) of the bitset of the bloom filter, and is optional. If you do not provide the size, it will default to the maximal bloom filter size (ParquetBloomFilter::MAX_BYTES).

Example: “col1:1024,col2,col4:100"

Limitations

The following table contains the data types that are not supported currently. Support for these data types may be added in a future release.

Impala type	Reason for not supporting
VARCHAR(N)	truncation can change hash
CHAR(N)	padding / truncation can change hash
DECIMAL	multiple encodings supported
TIMESTAMP	multiple encodings supported, timezone conversion
DATE	not considered yet

Late Materialization of Columns

This is an optimization feature available in Impala and is implemented to optimize queries that reference a large number of columns and filter out a significant portion of the rows. This currently works only for Parquet tables.

When a query is processed, any fields referenced in the select list are decoded from file storage and materialized into row values in memory to be processed for joins, predicates, and so on. Prior to this optimization, the materialization is applied to all the row values, regardless of whether they are eliminated by predicates. However with late materialization, as the name implies, only fields that are referenced by predicates and rows that are not filtered out by predicates are fully materialized.

When a select * query is run over a 4 billion row table that returns a single row, it may take ~30 seconds to execute, however it may take less than 3 seconds if the query is replaced by select field1, field2 .... This is because the select * query is materializing all the fields for rows, regardless of whether they match or not. And in the case of the select f ield1, field2 query only the columns that are required for filtering the data need to be materialized.

Use the configuration parameter (query option) parquet_late_materialization_threshold to provide the minimum number of consecutive rows that are filtered out to avoid materialization. If set to less than 0, it disables the late materialization. In most cases the default setting of 20 will provide optimal performance.

Partitioning

Use partitioning to speed up queries that retrieve data based on values from one or more columns.

Partitioning is a technique for physically dividing the data during loading based on values from one or more columns. For example, with a school_records table partitioned on a year column, there is a separate data directory for each different year value, and all the data for that year is stored in a data file in that directory. A query that includes a WHERE condition such as YEAR=1966, YEAR IN (1989,1999), or YEAR BETWEEN 1984 AND 1989 can examine only the data files from the appropriate directory or directories, greatly reducing the amount of data to read and test.

Parquet is a popular format for partitioned Impala tables because it is well suited to handle huge data volumes.

You can add, drop, set the expected file format, or set the HDFS location of the data files for individual partitions within an Impala table.

When to Use Partitioned Tables

Partitioning is typically appropriate for:

- Tables that are very large, where reading the entire data set takes an impractical amount of time.
- Tables that are always or almost always queried with conditions on the partitioning columns. In our example of a table partitioned by year, SELECT COUNT(*) FROM school_records WHERE year = 1985 is efficient, only examining a small fraction of the data; but SELECT COUNT(*) FROM school_records has to process a separate data file for each year, resulting in more overall work than in an unpartitioned table. You would probably not partition this way if you frequently queried the table based on last name, student ID, and so on without testing the year.
- Columns that have reasonable cardinality (number of different values). If a column only has a small number of values, for example Male or Female, you do not gain much efficiency by eliminating only about 50% of the data to read for each query. If a column has only a few rows matching each value, the number of directories to process can become a limiting factor, and the data file in each directory could be too small to take advantage of the Hadoop mechanism for transmitting data in multi-megabyte blocks. For example, you might partition census data by year, store sales data by year and month, and web traffic data by year, month, and day. (Some users with high volumes of incoming data might even partition down to the individual hour and minute.)
- Data that already passes through an extract, transform, and load (ETL) pipeline. The values of the partitioning columns are stripped from the original data files and represented by directory names, so loading data into a partitioned table involves some sort of transformation or preprocessing.

SQL Statement for Partitioned Tables

In terms of Impala SQL syntax, partitioning affects these statements:

- CREATE TABLE: You specify a PARTITIONED BY clause when creating the table to identify names and data types of the partitioning columns. These columns are not included in the main list of columns for the table.
- CREATE TABLE AS SELECT: Use the PARTITIONED BY clause to create a partitioned table, copy data into it, and create new partitions based on the values in the inserted data.
- ALTER TABLE: Add or drop partitions, to work with different portions of a huge data set. You can designate the HDFS directory that holds the data files for a specific partition. With data partitioned by date values, you might “age out” data that is no longer relevant.

  If you are creating a partition for the first time and specifying its location, for maximum efficiency, use a single ALTER TABLE statement including both the ADD PARTITION and LOCATIONclauses, rather than separate statements with ADD PARTITION and SET LOCATION clauses.
- INSERT: When you insert data into a partitioned table, you identify the partitioning columns. One or more values from each inserted row are not stored in data files, but instead determine the directory where that row value is stored. You can also specify which partition to load a set of data into, with INSERT OVERWRITE statements; you can replace the contents of a specific partition but you cannot append data to a specific partition.

  By default, if an INSERT statement creates any new subdirectories underneath a partitioned table, those subdirectories are assigned default HDFS permissions for the impala user. To make each subdirectory have the same permissions as its parent directory in HDFS, specify the ##insert_inherit_permissions startup option for the Impala Daemon.
- Although the syntax of the SELECT statement is the same whether or not the table is partitioned, the way queries interact with partitioned tables can have a dramatic impact on performance and scalability. The mechanism that lets queries skip certain partitions during a query is known as partition pruning.
- SHOW PARTITIONS: Displays information about each partition in a table.

Static and Dynamic Partitioning Clauses

Specifying all the partition columns in a SQL statement is called *static partitioning*, because the statement affects a single predictable partition. For example, you use static partitioning with an ALTER TABLE statement that affects only one partition, or with an INSERT statement that inserts all values into the same partition:

```
INSERT INTO t1 PARTITION(x=10, y='a') SELECT c1 FROM some_other_table;
```

When you specify some partition key columns in an INSERT statement, but leave out the values, Impala determines which partition to insert. This technique is called *dynamic partitioning*.

```
INTSERT INTO t1 PARTITION(x, y='b') SELECT c1, c2 FROM some_other_table;
-- Create new partition if necessary based on variable year, month, and day;
 insert a single value.
INSERT INTO weather PARTITION (year, month, day) SELECT 'cloudy',2014,4,21;
-- Create new partition if necessary for specified year and month but variab
le day; insert a single value.
INSERT INTO weather PARTITION (year=2014, month=04, day) SELECT 'sunny',22;
```

The more key columns you specify in the PARTITION clause, the fewer columns you need in the SELECT list. The trailing columns in the SELECT list are substituted in order for the partition key columns with no specified value.

Refreshing a Single Partition

The REFRESH statement is typically used with partitioned tables when new data files are loaded into a partition by some non-Impala mechanism, such as a Hive or Spark job. The REFRESH statement makes Impala aware of the new data files so that they can be used in Impala queries. Because partitioned tables typically contain a high volume of data, the REFRESH operation for a full partitioned table can take significant time.

You can include a PARTITION (*PARTITION_SPEC*) clause in the REFRESH statement so that only a single partition is refreshed. For example, REFRESH big_table PARTITION (year=2017, month=9, day=30). The partition spec must include all the partition key columns.

Partition Key Columns

The columns you choose as the partition keys should be ones that are frequently used to filter query results in important, large-scale queries. Popular examples are some combination of year, month, and day when the data has associated time values, and geographic region when the data is associated with some place.

- For time-based data, split out the separate parts into their own columns, because Impala cannot partition based on a TIMESTAMP column.
- The data type of the partition columns does not have a significant effect on the storage required, because the values from those columns are not stored in the data files, rather they are represented as strings inside HDFS directory names.
- You can enable the OPTIMIZE_PARTITION_KEY_SCANS query option to speed up queries that only refer to partition key columns, such as SELECT MAX(year). This setting is not enabled by default because the query behavior is slightly different if the table contains partition directories without actual data inside.
- All the partition key columns must be scalar types.
- When Impala queries data stored in HDFS, it is most efficient to use multi-megabyte files to take advantage of the HDFS block size. For Parquet tables, the block size (and ideal size of the data files) is 256 MB.. Therefore, avoid specifying too many partition key columns, which could result in individual partitions containing only small amounts of data. For example, if you receive 1 GB of data per day, you might partition by year, month, and day; while if you receive 5 GB of data per minute, you might partition by year, month, day, hour, and minute. If you have data with a geographic component, you might partition based on postal code if you have many megabytes of data for each postal code, but if not, you might partition by some larger region such as city, state, or country. state

If you frequently run aggregate functions such as MIN(), MAX(), and COUNT(DISTINCT) on partition key columns, consider enabling the OPTIMIZE_PARTITION_KEY_SCANS query option, which optimizes such queries.

Setting Different File Formats for Partitions in a Table

Partitioned tables have the flexibility to use different file formats for different partitions. For example, if you originally received data in text format, then received new data in RCFile format, and eventually began receiving data in Parquet format, all that data could reside in the same table for queries. You just need to ensure that the table is structured so that the data files that use different file formats reside in separate partitions.

For example, here is how you might switch from text to Parquet data as you receive data for different years:

```
[localhost:21000] > CREATE TABLE census (name STRING) PARTITIONED BY (year S
MALLINT);
[localhost:21000] > ALTER TABLE census ADD PARTITION (year=2012); -- Text
format;
[localhost:21000] > ALTER TABLE census ADD PARTITION (year=2013); -- Text f
ormat switches to Parquet before data loaded;
[localhost:21000] > ALTER TABLE census PARTITION (year=2013) SET FILEFORMAT
 PARQUET;

[localhost:21000] > INSERT INTO census PARTITION (year=2012) VALUES ('Smith'
),('Jones'),('Lee'),('Singh');
[localhost:21000] > INSERT INTO census PARTITION (year=2013) VALUES ('Flo
res'),('Bogomolov'),('Cooper'),('Appiah');
```

At this point, the HDFS directory for year=2012 contains a text-format data file, while the HDFS directory for year =2013 contains a Parquet data file. As always, when loading non-trivial data, you would use INSERT ... SELECT or LOAD DATA to import data in large batches, rather than INSERT ... VALUES which produces small files that are inefficient for real-world queries.

For other file types that Impala cannot create natively, you can switch into Hive and issue the ALTER TABLE ... SET FILEFORMAT statements and INSERT or LOAD DATA statements there. After switching back to Impala, issue a REFRESH *TABLE_NAME* statement so that Impala recognizes any partitions or new data added through Hive.

Dropping Partitions

What happens to the data files when a partition is dropped depends on whether the partitioned table is designated as internal or external. For an internal (managed) table, the data files are deleted. For example, if data in the partitioned table is a copy of raw data files stored elsewhere, you might save disk space by dropping older partitions that are no longer required for reporting, knowing that the original data is still available if needed later. For an external table, the data files are left alone. For example, dropping a partition without deleting the associated files lets Impala consider a smaller set of partitions, improving query efficiency and reducing overhead for DDL operations on the table; if the data is needed again later, you can add the partition again.

Using Partitioning with Kudu Tables

Kudu tables use a more fine-grained partitioning scheme than tables containing HDFS data files. You specify a PART ITION BY clause with the CREATE TABLE statement to identify how to divide the values from the partition key columns.

Keeping Statistics Up to Date for Partitioned Tables

Because the COMPUTE STATS statement can be resource-intensive to run on a partitioned table as new partitions are added, Impala includes a variation of this statement that allows computing statistics on a per-partition basis such that stats can be incrementally updated when new partitions are added.

The COMPUTE INCREMENTAL STATS variation computes statistics only for partitions that were added or changed since the last COMPUTE INCREMENTAL STATS statement, rather than the entire table. It is typically used for tables where a full COMPUTE STATS operation takes too long to be practical each time a partition is added or dropped.

Partition Pruning for Queries

Partition pruning refers to the mechanism where a query can skip reading the data files corresponding to one or more partitions.

If you can arrange for queries to prune large numbers of unnecessary partitions from the query execution plan, the queries use fewer resources and are thus proportionally faster and more scalable.

For example, if a table is partitioned by columns YEAR, MONTH, and DAY, then WHERE clauses such as WHERE year = 2013, WHERE year < 2010, or WHERE year BETWEEN 1995 AND 1998 allow Impala to skip the data files in all partitions outside the specified range. Likewise, WHERE year = 2013 AND month BETWEEN 1 AND 3 could prune even more partitions, reading the data files for only a portion of one year.

If a view applies to a partitioned table, any partition pruning considers the clauses on both the original query and any additional WHERE predicates in the query that refers to the view.

In queries involving both analytic functions and partitioned tables, partition pruning only occurs for columns named in the PARTITION BY clause of the analytic function call. For example, if an analytic function query has a clause such as WHERE year=2016, the way to make the query prune all other YEAR partitions is to include PARTITION BY year in the analytic function call; for example, OVER (PARTITION BY year,*OTHER_COLUMNS OTHER_ANALYTIC_CLAUSES*).

Checking if Partition Pruning Happens for a Query

To check the effectiveness of partition pruning for a query, check the EXPLAIN output for the query before running it. For example, this example shows a table with 3 partitions, where the query only reads 1 of them. The notation #partitions=1/3 in the EXPLAIN plan confirms that Impala can do the appropriate partition pruning.

```
[localhost:21000] > INSERT INTO census PARTITION (year=2010) VALUES ('Smith'
),('Jones');
[localhost:21000] > INSERT INTO census PARTITION (year=2011) VALUES ('Smith
'),('Jones'),('Doe');
[localhost:21000] > INSERT INTO census PARTITION (year=2012) VALUES ('Smit
h'),('Doe');
[localhost:21000] > EXPLAIN select name from census where year=2010;
+----------------------------------------------------------------+
| Explain String                                                 |
+----------------------------------------------------------------+
| PLAN FRAGMENT 1                                                |
|   PARTITION: RANDOM                                            |
|                                                                |
|   STREAM DATA SINK                                             |
|     EXCHANGE ID: 1                                             |
|     UNPARTITIONED                                              |
|                                                                |
|   0:SCAN HDFS                                                  |
|      table=predicate_propagation.census #partitions=1/3 size=12B |
+----------------------------------------------------------------+
```

For a report of the volume of data that was actually read and processed at each stage of the query, check the output of the SUMMARY command immediately after running the query. For a more detailed analysis, look at the output of the PROFILE command; it includes this same summary report near the start of the profile output.

Predicate Propagation

Impala can do partition pruning in cases where the partition key column is not directly compared to a constant. Using the predicate propagation technique, Impala applies the transitive property to other parts of the WHERE clause.

In this example, the census table includes another column indicating when the data was collected, which happens in 10-year intervals. Even though the query does not compare the partition key column (YEAR) to a constant value, Impala can deduce that only the partition YEAR=2010 is required, and again only reads 1 out of 3 partitions.

```
[localhost:21000] > CREATE TABLE census (name STRING, census_year INT) PARTI
TIONED BY (year INT);
[localhost:21000] > INSERT INTO census PARTITION (year=2010) VALUES  ('Smi
th',2010),('Jones',2010);
[localhost:21000] > INSERT INTO census PARTITION (year=2011) VALUES  ('Smi
th',2020),('Jones',2020),('Doe',2020);
```

```
[localhost:21000] > INSERT INTO census PARTITION (year=2012) VALUES  ('Sm
ith',2020),('Doe',2020);
[localhost:21000] >
[localhost:21000] > EXPLAIN select name from census where year = census_year
 and census_year=2010;
+----------------------------------------------------------------+
| Explain String                                                 |
+----------------------------------------------------------------+
| PLAN FRAGMENT 1                                                |
|   PARTITION: RANDOM                                            |
|                                                                |
|   STREAM DATA SINK                                             |
|     EXCHANGE ID: 1                                             |
|     UNPARTITIONED                                              |
|                                                                |
|   0:SCAN HDFS                                                  |
|      table=predicate_propagation.census #partitions=1/3 size=22B |
|      predicates: census_year = 2010, year = census_year        |
+----------------------------------------------------------------+
```

Dynamic Partition Pruning

Impala supports two types of partition pruning.

- *Static partition pruning*: The conditions in the WHERE clause are analyzed to determine in advance which partitions can be safely skipped.
- *Dynamic partition pruning*: Information about the partitions is collected during the query execution, and Impala prunes unnecessary partitions. The information is not available in advance before runtime.

For example, if partition key columns are compared to literal values in a WHERE clause, Impala can perform static partition pruning during the planning phase to only read the relevant partitions:

```
-- The query only needs to read 3 partitions whose key values are known ahe
ad of time.
-- That's static partition pruning.
SELECT COUNT(*) FROM sales_table WHERE year IN (2005, 2010, 2015);
```

Dynamic partition pruning involves using information only available at run time, such as the result of a subquery. The following example shows a simple dynamic partition pruning.

```
CREATE TABLE yy (s STRING) PARTITIONED BY (year INT);
INSERT INTO yy PARTITION (year) VALUES ('1999', 1999), ('2000', 2000),
  ('2001', 2001), ('2010', 2010), ('2018', 2018);
COMPUTE STATS yy;

CREATE TABLE yy2 (s STRING, year INT);
INSERT INTO yy2 VALUES ('1999', 1999), ('2000', 2000), ('2001', 2001);
COMPUTE STATS yy2;
-- The following query reads an unknown number of partitions, whose key val
ues
-- are only known at run time. The runtime filters line shows the
-- information used in query fragment 02 to decide which partitions to skip.

EXPLAIN SELECT s FROM yy WHERE year IN (SELECT year FROM yy2);
+----------------------------------------------------------------------------+
| PLAN-ROOT SINK                                                             |
| |                                                                          |
| 04:EXCHANGE [UNPARTITIONED]                                                |
| |                                                                          |
| 02:HASH JOIN [LEFT SEMI JOIN, BROADCAST]                                   |
```

```
| |  hash predicates: year = year                                          |
| |  runtime filters: RF000 <- year                                     |
| |                                                                        |
| |--03:EXCHANGE [BROADCAST]                                               |
| |  |                                                                     |
| |  01:SCAN HDFS [default.yy2]                                            |
| |     partitions=1/1 files=1 size=620B                                   |
| |                                                                        |
| 00:SCAN HDFS [default.yy]                                                |
|    partitions=5/5 files=5 size=1.71KB                                  |
|    runtime filters: RF000 -> year                                      |
+------------------------------------------------------------------------+

SELECT s FROM yy WHERE year IN (SELECT year FROM yy2); -- Returns 3 rows fro
m yy
PROFILE;
```

In the above example, Impala evaluates the subquery, sends the subquery results to all Impala nodes participating in the query, and then each `impalad` daemon uses the dynamic partition pruning optimization to read only the partitions with the relevant key values.

The output query plan from the EXPLAIN statement shows that runtime filters are enabled. The plan also shows that it expects to read all 5 partitions of the yy table, indicating that static partition pruning will not happen.

The Filter summary in the PROFILE output shows that the scan node filtered out based on a runtime filter of dynamic partition pruning.

```
Filter 0 (1.00 MB):
 - Files processed: 3
 - Files rejected: 1 (1)
 - Files total: 3 (3)
```

Dynamic partition pruning is especially effective for queries involving joins of several large partitioned tables. Evaluating the ON clauses of the join predicates might normally require reading data from all partitions of certain tables. If the WHERE clauses of the query refer to the partition key columns, Impala can now often skip reading many of the partitions while evaluating the ON clauses. The dynamic partition pruning optimization reduces the amount of I/O and the amount of intermediate data stored and transmitted across the network during the query.

Dynamic partition pruning is part of the runtime filtering feature, which applies to other kinds of queries in addition to queries against partitioned tables.

HDFS Caching

Impala can use the HDFS caching feature to make more effective use of RAM so that repeated queries can take advantage of data “pinned” in memory regardless of how much data is processed overall.

The HDFS caching feature lets you designate a subset of frequently accessed data to be pinned permanently in memory, remaining in the cache across multiple queries and never being evicted. This technique is suitable for tables or partitions that are frequently accessed and are small enough to fit entirely within the HDFS memory cache. For example, you might designate several dimension tables to be pinned in the cache, to speed up many different join queries that reference them. Or in a partitioned table, you might pin a partition holding data from the most recent time period because that data will be queried intensively; then when the next set of data arrives, you could unpin the previous partition and pin the partition holding the new data.

The performance gain comes from two aspects:

- Reading from RAM instead of disk
- Accessing the data straight from the cache area instead of copying from one RAM area to another

  This yields further performance improvement over the standard OS caching mechanism, which still results in memory-to-memory copying of cached data. Because accessing HDFS cached data avoids a memory-to-memory

copy operation, queries involving cached data require less memory on the Impala side than the equivalent queries on uncached data.

Due to a limitation of HDFS, zero-copy reads are not supported with encryption. Cloudera recommends not using HDFS caching for Impala data files in encryption zones. The queries fall back to the normal read path during query execution, which might cause some performance overhead.

Because this Impala performance feature relies on HDFS infrastructure, it only applies to Impala tables that use HDFS data files. HDFS caching for Impala does not apply to HBase tables, S3 tables, Kudu tables, or Isilon tables.

Using HDFS Caching for Impala Tables and Partitions

Begin by choosing which tables or partitions to cache. For example, these might be lookup tables that are accessed by many different join queries, or partitions corresponding to the most recent time period that are analyzed by different reports or ad-hoc queries.

In your SQL statements, you specify logical divisions such as tables and partitions to be cached. Impala translates these requests into HDFS-level directives that apply to particular directories and files. For example, given a partitioned table CENSUS with a partition key column YEAR, you could choose to cache all or part of the data as follows.

```
-- Cache the entire table (all partitions).
ALTER TABLE census SET CACHED IN 'POOL_NAME';

-- Remove the entire table from the cache.
ALTER TABLE census SET UNCACHED;
-- Cache a portion of the table (a single partition).
-- If the table is partitioned by multiple columns (such as year, month,
day),
-- the ALTER TABLE command must specify values for all those columns.
ALTER TABLE census PARTITION (year=1960) SET CACHED IN 'POOL_NAME';

-- Cache the data from one partition on up to 4 hosts, to minimize CPU load
 on any
-- single host when the same data block is processed multiple times.
ALTER TABLE census PARTITION (year=1970)
  SET CACHED IN 'POOL_NAME' WITH REPLICATION = 4;

-- At each stage, check the volume of cached data.
-- For large tables or partitions, the background loading might take some t
ime,
-- so you might have to wait and reissue the statement until all the data
-- has finished being loaded into the cache.
SHOW TABLE STATS census;
+-------+-------+--------+------+--------------+--------+
| year  | #Rows | #Files | Size | Bytes Cached | Format |
+-------+-------+--------+------+--------------+--------+
| 1900  | -1    | 1      | 11B  | NOT CACHED   | TEXT   |
| 1940  | -1    | 1      | 11B  | NOT CACHED   | TEXT   |
| 1960  | -1    | 1      | 11B  | 11B          | TEXT   |
| 1970  | -1    | 1      | 11B  | NOT CACHED   | TEXT   |
| Total | -1    | 4      | 44B  | 11B          |        |
+-------+-------+--------+------+--------------+--------+
```

CREATE TABLE and ALTER TABLE considerations:

The HDFS caching feature affects the Impala CREATE TABLE statement as follows:

- You can put a CACHED IN '*POOL_NAME*' clause and optionally a WITH REPLICATION = *NUMBER_OF_HOSTS* clause at the end of a CREATE TABLE statement to automatically cache the entire contents of the table, including any partitions added later.

  The *POOL_NAME* is a pool that you previously set up with the `hdfs cacheadmin` command in HDFS.

- Once a table is designated for HDFS caching through the CREATE TABLE statement, if new partitions are added later through ALTER TABLE ... ADD PARTITION statements, the data in those new partitions is automatically cached in the same pool.
- If you want to perform repetitive queries on a subset of data from a large table, and it is not practical to designate the entire table or specific partitions for HDFS caching, you can create a new cached table with just a subset of the data by using CREATE TABLE ... CACHED IN '*POOL_NAME*' AS SELECT ... WHERE .... When you are finished with generating reports from this subset of data, drop the table and both the data files and the data cached in RAM are automatically deleted.
- If you have designated a table or partition as cached through the CREATE TABLE or ALTER TABLE statements, subsequent attempts to relocate the table or partition through an ALTER TABLE ... SET LOCATION statement will fail. You must issue an ALTER TABLE ... SET UNCACHED statement for the table or partition first. Otherwise, Impala would lose track of some cached data files and have no way to uncache them later.
- The optional WITH REPLICATION clause for CREATE TABLE and ALTER TABLE lets you specify a *replication factor*, the number of hosts on which to cache the same data blocks. When Impala processes a cached data block, where the cache replication factor is greater than 1, Impala randomly selects a host that has a cached copy of that data block. This optimization avoids excessive CPU usage on a single host when the same cached data block is processed multiple times. Cloudera recommends specifying a value greater than or equal to the HDFS block replication factor.

INSERT and LOAD DATA considerations:

- When HDFS caching is enabled for a table or partition, new data files are cached automatically when they are added to the appropriate directory in HDFS, without the need for a REFRESH statement in Impala. Impala automatically performs a REFRESH once the new data is loaded into the HDFS cache.
- If you perform an INSERT or LOAD DATA through Hive, Impala only recognizes the new data files after a REFRESH *TABLE_NAME* statement in Impala.
- If the cache pool is entirely full, or becomes full before all the requested data can be cached, the Impala DDL statement returns an error. This is to avoid situations where only some of the requested data could be cached.

DROP TABLE and ALTER TABLE DROP PARTITION considerations:

The HDFS caching feature interacts with the Impala DROP TABLE and ALTER TABLE ... DROP PARTITION statements as follows:

- When you issue a DROP TABLE or ALTER TABLE ... DROP PARTITION for a table that is entirely cached, or has some partitions cached, the DROP TABLE succeeds and all the cache directives Impala submitted for that table are removed from the HDFS cache system.
- The underlying data files are removed if the dropped table is an internal table, or the dropped partition is in its default location underneath an internal table. The data files are left alone if the dropped table is an external table, or if the dropped partition is in a non-default location.
- If you drop an HDFS cache pool through the `hdfs cacheadmin` command, all the Impala data files are preserved, just no longer cached. After a subsequent REFRESH, SHOW TABLE STATS reports 0 bytes cached for each associated Impala table or partition.
- If you designated the data files as cached through the `hdfs cacheadmin` command, and the data files are left behind as described in the previous item, the data files remain cached.

  Impala only removes the cache directives submitted by Impala through the CREATE TABLE or ALTER TABLE statements.
- One file can have multiple redundant cache directives. The directives all have unique IDs, and owners so that the system can tell them apart.

SHOW TABLE STATS and SHOW PARTITIONS considerations:

For each table or partition, the SHOW TABLE STATS or SHOW PARTITIONS statement displays the number of bytes currently cached by the HDFS caching feature.

A value of 0, or a smaller number than the overall size of the table or partition, indicates that the cache request has been submitted but the data has not been entirely loaded into memory yet.

If there are no cache directives in place for that table or partition, the result set displays NOT CACHED.

SELECT considerations:

The Impala HDFS caching feature interacts with the SELECT statement and query performance as follows:

- Impala automatically reads from memory any data that has been designated as cached and actually loaded into the HDFS cache. (It could take some time after the initial request to fully populate the cache for a table with large size or many partitions.)
- Impala queries take advantage of HDFS cached data regardless of whether the cache directive was issued by Impala or externally through the `hdfs cacheadmin` command, for example for an external table where the cached data files might be accessed by several different Hadoop components.
- If your query returns a large result set, the time reported for the query could be dominated by the time needed to print the results on the screen. To measure the time for the underlying query processing, query the COUNT() of the big result set, which does all the same processing but only prints a single line to the screen.
- Impala automatically randomizes which host processes a cached HDFS block, to avoid CPU hotspots. For tables where HDFS caching is not applied, Impala designates which host to process a data block using an algorithm that estimates the load on each host. If CPU hotspots still arise during queries, you can enable additional randomization for the scheduling algorithm for non-HDFS cached data by setting the SCHEDULE_RANDOM_REPLICA query option.
- If you drop a cache pool with the `hdfs cacheadmin` command, Impala queries against the associated data files will still work, by falling back to reading the files from disk. After performing a REFRESH on the table, Impala reports the number of bytes cached as 0 for all associated tables and partitions.

Performance Considerations for HDFS Caching with Impala

Impala supports efficient reads from data that is pinned in memory through HDFS caching. Impala takes advantage of the HDFS API and reads the data from memory rather than from disk whether the data files are pinned using Impala DDL statements, or using the command-line mechanism where you specify HDFS paths.

When you examine the output of the `impala-shell SUMMARY` command, or look in the metrics report for the `impalad` daemon, you see how many bytes are read from the HDFS cache. For example, this excerpt from a query profile illustrates that all the data read during a particular phase of the query came from the HDFS cache, because the BytesRead and BytesReadDataNodeCache values are identical.

```
HDFS_SCAN_NODE (id=0):(Total: 11s114ms, non-child: 11s114ms, % non-child: 10
0.00%)
        - AverageHdfsReadThreadConcurrency: 0.00
        - AverageScannerThreadConcurrency: 32.75
        - BytesRead: 10.47 GB (11240756479)
        - BytesReadDataNodeCache: 10.47 GB (11240756479)
        - BytesReadLocal: 10.47 GB (11240756479)
        - BytesReadShortCircuit: 10.47 GB (11240756479)
        - DecompressionTime: 27s572ms
```

For queries involving smaller amounts of data, or in single-user workloads, you might not notice a significant difference in query response time with or without HDFS caching. Even with HDFS caching turned off, the data for the query might still be in the Linux OS buffer cache. The benefits become clearer as data volume increases, and especially as the system processes more concurrent queries. HDFS caching improves the scalability of the overall system. That is, it prevents query performance from declining when the workload outstrips the capacity of the Linux OS cache.

- For small amounts of data, the query speedup might not be noticeable in terms of wall clock time. The performance might be roughly the same with HDFS caching turned on or off, due to recently used data being held in the OS cache. The difference is more pronounced with:
  - Data volumes (for all queries running concurrently) that exceed the size of the OS cache.
  - A busy cluster running many concurrent queries, where the reduction in memory-to-memory copying and overall memory usage during queries results in greater scalability and throughput.

When data is requested to be pinned in memory, that process happens in the background without blocking access to the data while the caching is in progress. Loading the data from disk could take some time. Impala reads each HDFS data block from memory if it has been pinned already, or from disk if it has not been pinned yet.

Memory Considerations

The amount of data that you can pin on each node through the HDFS caching mechanism is subject to a quota that is enforced by the underlying HDFS service. Before requesting to pin an Impala table or partition in memory, check that its size does not exceed this quota.

Note: Because the HDFS cache consists of combined memory from all the DataNodes in the cluster, cached tables or partitions can be bigger than the amount of HDFS cache memory on any single host.

The Impala HDFS caching feature interacts with the Impala memory limits as follows:

- The maximum size of each HDFS cache pool is specified externally to Impala, through the `hdfs cacheadmin` command.
- All the memory used for HDFS caching is separate from the `impalad` daemon address space and does not count towards the limits of the --mem_limit startup option, MEM_LIMIT query option, or further limits imposed through YARN resource management or the Linux cgroups mechanism.

HDFS Block Skew

For best performance of Impala parallel queries, the work is divided equally across hosts in the cluster, and all hosts take approximately equal time to finish their work. If one host takes substantially longer than others, the extra time needed for the slow host can become the dominant factor in query performance. Therefore, one of the first steps in performance tuning for Impala is to detect and correct such conditions.

The main cause of uneven performance that you can correct within Impala is *skew* in the number of HDFS data blocks processed by each host, where some hosts process substantially more data blocks than others. This condition can occur because of uneven distribution of the data values themselves, for example causing certain data files or partitions to be large while others are very small. (Although it is possible to have unevenly distributed data without any problems with the distribution of HDFS blocks.) Block skew could also be due to the underlying block allocation policies within HDFS, the replication factor of the data files, and the way that Impala chooses the host to process each data block.

The most convenient way to detect block skew, or slow-host issues in general, is to examine the “executive summary” information from the query profile after running a query:

- In `impala-shell`, issue the SUMMARY command immediately after the query is complete, to see just the summary information. If you detect issues involving skew, you might switch to issuing the PROFILE command, which displays the summary information followed by a detailed performance analysis.
- In the Cloudera Manager interface or the Impala debug web UI, click on the Profile link associated with the query after it is complete. The executive summary information is displayed early in the profile output.

For each phase of the query, you see an Avg Time and a Max Time value, along with #Hosts indicating how many hosts are involved in that query phase. For all the phases with #Hosts greater than one, look for cases where the maximum time is substantially greater than the average time. Focus on the phases that took the longest, for example, those taking multiple seconds rather than milliseconds or microseconds.

If you detect that some hosts take longer than others, first rule out non-Impala causes. One reason that some hosts could be slower than others is if those hosts have less capacity than the others, or if they are substantially busier due to unevenly distributed non-Impala workloads:

- For clusters running Impala, keep the relative capacities of all hosts roughly equal. Any cost savings from including some underpowered hosts in the cluster will likely be outweighed by poor or uneven performance, and the time spent diagnosing performance issues.
- If non-Impala workloads cause slowdowns on some hosts but not others, use the appropriate load-balancing techniques for the non-Impala components to smooth out the load across the cluster.

If the hosts on your cluster are evenly powered and evenly loaded, examine the detailed profile output to determine which host is taking longer than others for the query phase in question. Examine how many bytes are processed during that phase on that host, how much memory is used, and how many bytes are transmitted across the network.

The most common symptom is a higher number of bytes read on one host than others, due to one host being requested to process a higher number of HDFS data blocks. This condition is more likely to occur when the number of blocks accessed by the query is relatively small. For example, if you have a 10-node cluster and the query processes 10 HDFS blocks, each node might not process exactly one block. If one node sits idle while another node processes two blocks, the query could take twice as long as if the data was perfectly distributed.

Possible solutions in this case include:

- If the query is artificially small, perhaps for benchmarking purposes, scale it up to process a larger data set. For example, if some nodes read 10 HDFS data blocks while others read 11, the overall effect of the uneven distribution is much lower than when some nodes did twice as much work as others. As a guideline, aim for a “sweet spot” where each node reads 2 GB or more from HDFS per query. Queries that process lower volumes than that could experience inconsistent performance that smooths out as queries become more data-intensive.
- If the query processes only a few large blocks, so that many nodes sit idle and cannot help to parallelize the query, consider reducing the overall block size. For example, you might adjust the PARQUET_FILE_SIZE query option before copying or converting data into a Parquet table. Or you might adjust the granularity of data files produced earlier in the ETL pipeline by non-Impala components. In Impala 2.0 and later, the default Parquet block size is 256 MB, reduced from 1 GB, to improve parallelism for common cluster sizes and data volumes.
- Reduce the amount of compression applied to the data. For text data files, the highest degree of compression (gzip) produces unsplittable files that are more difficult for Impala to process in parallel, and require extra memory during processing to hold the compressed and uncompressed data simultaneously. For binary formats such as Parquet and Avro, compression can result in fewer data blocks overall, but remember that when queries process relatively few blocks, there is less opportunity for parallel execution and many nodes in the cluster might sit idle. Note that when Impala writes Parquet data with the query option COMPRESSION_CODEC=NONE enabled, the data is still typically compact due to the encoding schemes used by Parquet, independent of the final compression step.

Understanding Performance using EXPLAIN Plan

To understand the high-level performance considerations for Impala queries, read the output of the EXPLAIN statement for the query. You can get the EXPLAIN plan without actually running the query itself.

The EXPLAIN statement gives you an outline of the logical steps that a query will perform, such as how the work will be distributed among the nodes and how intermediate results will be combined to produce the final result set. You can see these details before actually running the query. You can use this information to check that the query will not operate in some very unexpected or inefficient way.

Read the EXPLAIN plan from bottom to top:

- The last part of the plan shows the low-level details such as the expected amount of data that will be read, where you can judge the effectiveness of your partitioning strategy and estimate how long it will take to scan a table based on total data size and the size of the cluster.
- As you work your way up, next you see the operations that will be parallelized and performed on each Impala node.
- At the higher levels, you see how data flows when intermediate result sets are combined and transmitted from one node to another.
- The EXPLAIN_LEVEL query option lets you customize how much detail to show in the EXPLAIN plan depending on whether you are doing high-level or low-level tuning, dealing with logical or physical aspects of the query.

Read the EXPLAIN plan from bottom to top:

- The last part of the plan shows the low-level details such as the expected amount of data that will be read, where you can judge the effectiveness of your partitioning strategy and estimate how long it will take to scan a table based on total data size and the size of the cluster.

- As you work your way up, next you see the operations that will be parallelized and performed on each Impala node.
- At the higher levels, you see how data flows when intermediate result sets are combined and transmitted from one node to another.
- The EXPLAIN_LEVEL query option lets you customize how much detail to show in the EXPLAIN plan depending on whether you are doing high-level or low-level tuning, dealing with logical or physical aspects of the query.

The example below shows how the standard EXPLAIN output moves from the lowest (physical) level to the higher (logical) levels. The query begins by scanning a certain amount of data; each node performs an aggregation operation (evaluating COUNT(*)) on some subset of data that is local to that node; the intermediate results are transmitted back to the coordinator node (labelled here as the EXCHANGE node); lastly, the intermediate results are summed to display the final result.

```
[impalad-host:21000] > EXPLAIN select count(*) from customer_address;
+----------------------------------------------------------+
| Explain String                                           |
+----------------------------------------------------------+
| Estimated Per-Host Requirements: Memory=42.00MB VCores=1 |
|                                                          |
| 03:AGGREGATE [MERGE FINALIZE]                            |
| |  output: sum(count(*))                                 |
| |                                                        |
| 02:EXCHANGE [PARTITION=UNPARTITIONED]                    |
| |                                                        |
| 01:AGGREGATE                                             |
| |  output: count(*)                                      |
| |                                                        |
| 00:SCAN HDFS [default.customer_address]                  |
|    partitions=1/1 size=5.25MB                            |
+----------------------------------------------------------+
```

The EXPLAIN plan is also printed at the beginning of the query PROFILE report to provide convenience in examining both the logical and physical aspects of the query side-by-side.

The amount of detail displayed in the EXPLAIN output is controlled by the EXPLAIN_LEVEL query option. You customize how much detail to show in the EXPLAIN plan depending on whether you are doing high-level or low-level tuning, dealing with logical or physical aspects of the query. You typically increase this setting from standard to extended when double-checking the presence of table and column statistics during performance tuning, or when estimating query resource usage in conjunction with the resource management features.

Understanding Performance using SUMMARY Report

For an overview of the physical performance characteristics for a query, issue the SUMMARY command in impala-s hell immediately after executing a query. This condensed information shows which phases of execution took the most time, and how the estimates for memory usage and number of rows at each phase compare to the actual values.

Like the EXPLAIN plan, it is easy to see potential performance bottlenecks in the SUMMARY report. Like the PROF ILE output, the SUMMARY report is available after the query is run, and it displays actual timing statistics.

The SUMMARY report is also printed at the beginning of the query profile report described in the PROFILE output, for convenience in examining high-level and low-level aspects of the query side-by-side.

When the MT_DOP query option is set to a value larger than 0, the #Inst column in the output shows the number of fragment instances. Impala decomposes each query into smaller units of work that are distributed across the cluster, and these units are referred as fragments.

When the MT_DOP query option is set to 0, the #Inst column in the output shows the same value as the #Hosts column, since there is exactly one fragment for each host.

For example, here is a query involving an aggregate function on a single-node. The different stages of the query and their timings are shown (rolled up for all nodes), along with estimated and actual values used in planning the query. In this case, the AVG() function is computed for a subset of data on each node (stage 01) and then the aggregated results from all nodes are combined at the end (stage 03). You can see which stages took the most time, and whether any estimates were substantially different than the actual data distribution. (When examining the time values, be sure to consider the suffixes such as us for microseconds and ms for milliseconds, rather than just looking for the largest numbers.)

```
> SELECT AVG(ss_sales_price) FROM store_sales WHERE ss_coupon_amt = 0;
> SUMMARY;
+--------------+--------+--------+----------+----------+-------+------------
+----------+---------------+-----------------+
| Operator     | #Hosts | #Inst  | Avg Time | Max Time | #Rows | Est. #Rows
 | Peak Mem | Est. Peak Mem | Detail          |
+--------------+--------+--------+----------+----------+-------+------------
+----------+---------------+-----------------+
| 03:AGGREGATE | 1      | 1      | 1.03ms   | 1.03ms   | 1     | 1
 | 48.00 KB | -1 B          | MERGE FINALIZE  |
| 02:EXCHANGE  | 1      | 1      | 0ns      | 0ns      | 1     | 1
 | 0 B      | -1 B          | UNPARTITIONED   |
| 01:AGGREGATE | 1      | 1      |30.79ms   | 30.79ms  | 1     | 1
 | 80.00 KB | 10.00 MB      |                 |
| 00:SCAN HDFS | 1      | 1      | 5.45s    | 5.45s    | 2.21M | -1
| 64.05 MB | 432.00 MB     | tpc.store_sales |
+--------------+--------+--------+----------+----------+-------+------------
+----------+---------------+-----------------+
```

Notice how the longest initial phase of the query is measured in seconds (s), while later phases working on smaller intermediate results are measured in milliseconds (ms) or even nanoseconds (ns).

Understanding Performance using Query Profile

To understand the detailed performance characteristics for a query, issue the PROFILE command in `impala-shell` immediately after executing a query. This low-level information includes physical details about memory, CPU, I/O, and network usage, and thus is only available after the query is actually run.

The PROFILE command, available in `impala-shell`, produces a detailed low-level report showing how a query was executed. Unlike the EXPLAIN plan, this information is only available after the query has finished. It shows physical details such as the number of bytes read, maximum memory usage, and so on for each node. You can use this information to determine if the query is I/O-bound or CPU-bound, whether some network condition is imposing a bottleneck, whether a slowdown is affecting some nodes but not others, and to check that recommended configuration settings such as short-circuit local reads are in effect.

By default, time values in the profile output reflect the wall-clock time taken by an operation. For values denoting system time or user time, the measurement unit is reflected in the metric name, such as ScannerThreadsSysTime or ScannerThreadsUserTime. For example, a multi-threaded I/O operation might show a small figure for wall-clock time, while the corresponding system time is larger, representing the sum of the CPU time taken by each thread. Or a wall-clock time figure might be larger because it counts time spent waiting, while the corresponding system and user time figures only measure the time while the operation is actively using CPU cycles.

The EXPLAIN plan is also printed at the beginning of the query profile report, for convenience in examining both the logical and physical aspects of the query side-by-side. The EXPLAIN_LEVEL query option also controls the verbosity of the EXPLAIN output printed by the PROFILE command.

The Per Node Profiles section in the profile output includes the following metrics that can be controlled by the RESOURCE_TRACE_RATIO query option.

- CpuIoWaitPercentage
- CpuSysPercentage
- CpuUserPercentage

- HostDiskReadThroughput: All data read by the host as part of the execution of this query (spilling), by the HDFS data node, and by other processes running on the same system.
- HostDiskWriteThroughput: All data written by the host as part of the execution of this query (spilling), by the HDFS data node, and by other processes running on the same system.
- HostNetworkRx: All data received by the host as part of the execution of this query, other queries, and other processes running on the same system.
- HostNetworkTx: All data transmitted by the host as part of the execution of this query, other queries, and other processes running on the same system.

Until this release, the impala-shell 'profile' command only returns the profile for the most recent profile attempt. There was no option to get the original query profile (the profile of the first query attempt that failed) from the impala-shell. This release added support for returning both the original and retried profiles for a retried query.

impala-shell has been modified and the 'profile' command has a new set of options. The syntax is now:

```
PROFILE [ALL | LATEST | ORIGINAL]
```

If you specify 'ALL', both the latest and original profiles are printed. When you specify 'LATEST', only the latest profile is printed. If 'ORIGINAL' is specified, only the original profile is printed. The default behavior is equivalent to specifying 'LATEST'.

Note: Support for this has only been added to HS2 given that Beeswax is being deprecated soon. The new 'profile' options have no affect when the Beeswax protocol is used.

Planner changes for CPU usage

This release brings some changes to the query planner to improve parallel sizing and resource estimation. This is done for workload-aware autoscaling and will be available as query options. These additional query options are added for tuning purposes. This new functionality will allow additional customers to enable multi-threaded queries globally for improved performance.

Before you begin

- You must obtain the entitlement to use this feature.

Overview

Until this release, estimating the peak memory was the scaling factor in selecting an executor group to run large queries. This estimation was calculated during the query compilation time. When large queries were identified, they were enabled to run on larger executor groups mapped to different resource groups.

This release adds a new CPU costing algorithm as the scaling factor by establishing an infrastructure to allow the weighted total amount of data to process to be used as a new factor in the selection of an executor group. The new costing algorithm returns a number representing an ideal CPU core count required for a query to run efficiently. This number will be compared against the total CPU count of an Impala executor group to determine if it fits to run in that executor group or not. If the CPU requirement is higher than the max-query-cpu-core-per-node-limit, even in the largest exec group, the query may be rejected.

The following query options are added to control this CPU costing algorithm.

- COMPUTE_PROCESSING_COST

  Option to enable this CPU costing algorithm. Set MT_DOP > 0 to enable this query option.

  Note: If entitlement is enabled, COMPUTE_PROCESSING_COST will be set to true by default. If you use the REQUEST_POOL query option to direct a query to a specific pool or queue and you want to maintain the original MT_DOP behavior then you must explicitly set the flag COMPUTE_PROCESSI NG_COST to False.

- PROCESSING_COST_MIN_THREADS

  Option to control the minimum number of fragment instances (threads) that the costing algorithm is allowed to adjust. The costing algorithm is in charge of increasing the fragment's instance count beyond this minimum number through producer-consumer rate comparison. The maximum number of fragments is max between PROCESSING_COST_MIN_THREADS, MT_DOP, and number of cores per executor.

You can also use the following backend flags to tune the algorithm.

- query_cpu_count_divisor

  Use this flag to divide the CPU requirement of a query to fit the total available CPU in the executor group. For example, setting the value to 2 will fit the query with CPU requirement 2X to an executor group with total available CPU X. Note that setting with a fractional value less than 1 effectively multiplies the query CPU requirement. A valid value is > 0.0. The default value is 1.
- processing_cost_use_equal_expr_weight

  The default value of this flag is True. When the default value is retained, all expression evaluations are weighted equally to 1 during the plan node's processing cost calculation.

  Note: This flag is experimental and the default value should not be changed.

- min_processing_per_thread

  This flag defines the minimum processing load (in processing cost unit) that a fragment instance needs to work on before the planner considers increasing the instance count based on the processing cost rather than the MT_DOP setting. The decision is per fragment. If you set this flag to a higher number, it will reduce parallelism of a fragment (more workload per fragment), and when set to a lower number, it will increase parallelism (less workload per fragment). Actual parallelism might still be constrained by the total number of cores in the selected executor groups, MT_DOP, or the PROCESSING_COST_MIN_THREADS query option. The value of this flag must be a positive integer and it defaults to 10M.
- skip_resource_checking_on_last_executor_group_set

  This flag defines if the resource checking will be skipped on the last executor group set. If this backend flag is set to true, memory and cpu resource checking will be skipped when a query is being planned against the last (largest) executor group set. Setting true will ensure that query will always get admitted into the last executor group set if it does not fit in any other group set.

DDL Bucketed Tables

To enhance query performance, create bucketed tables for the purpose of dividing data into smaller, more manageable chunks based on specified columns using the CLUSTER BY clause.

Syntax

Example 1: Bucketed table based on user_id

In the first example, the data is bucketed solely based on the user_id column.

```
CREATE TABLE tbl(
   user_id BIGINT,
   firstname STRING,
   lastname STRING
)
CLUSTERED BY(user_id) INTO 256 BUCKETS;
```

Example 2: Bucketed table based on user_id and sorted by firstname

In the second example, in addition to bucketing by user_id, the data is also sorted within each bucket based on the firs tname column. The bucket partitioning algorithm employs a hash function to evenly distribute rows across the table, aiding in optimizing query performance.

```
CREATE TABLE tbl2 (
  user_id BIGINT,
  firstname STRING,
  lastname STRING
)
CLUSTERED BY (user_id) SORT BY (firstname) INTO 256 BUCKETS;
```

Limitations

- CREATE bucketed table statement is currently not supported on Kudu and Iceberg tables.
- Alter operations such as add/drop/change/replace columns on bucketed tables are also not supported.
- Impala currently does not leverage the bucket info to optimize queries.

Scalability Considerations

The size of your cluster and the volume of data influences query performance. Typically, adding more cluster capacity reduces problems due to memory limits or disk throughput. On the other hand, larger clusters are more likely to have other kinds of scalability issues, such as a single slow node that causes performance problems for queries.

Impact of Many Tables or Partitions on Impala Catalog Performance and Memory Usage

Because Hadoop I/O is optimized for reading and writing large files, Impala is optimized for tables containing relatively few, large data files. Schemas containing thousands of tables, or tables containing thousands of partitions, can encounter performance issues during startup or during DDL operations such as ALTER TABLE statements.

Scalability Considerations for the Impala StateStore

If it takes a very long time for a cluster to start up with the message, This Impala daemon is not ready to accept user requests, the StateStore might be taking too long to send the entire catalog topic to the cluster. In this case, consider setting the Load Catalog in Background field to false in your Catalog Service configuration. This setting stops the StateStore from loading the entire catalog into memory at cluster startup. Instead, metadata for each table is loaded when the table is accessed for the first time.

Effect of Buffer Pool on Memory Usage

Most of the memory needed is reserved at the beginning of the query, avoiding cases where a query might run for a long time before failing with an out-of-memory error. The actual memory estimates and memory buffers are typically smaller than before, so that more queries can run concurrently or process larger volumes of data than previously.

Increase the MAX_ROW_SIZE query option setting when querying tables with columns containing long strings, many columns, or other combinations of factors that produce very large rows. If Impala encounters rows that are too large to process with the default query option settings, the query fails with an error message suggesting to increase the MAX_ROW_SIZE setting.

SQL Operations that Spill to Disk

Certain memory-intensive operations write temporary data to disk (known as *spilling* to disk) when Impala is close to exceeding its memory limit on a particular host.

What kinds of queries might spill to disk:

Several SQL clauses and constructs require memory allocations that could trigger spilling to disk:

- when a query uses a GROUP BY clause for columns with millions or billions of distinct values, Impala keeps a similar number of temporary results in memory, to accumulate the aggregate results for each value in the group.
- When large tables are joined together, Impala keeps the values of the join columns from one table in memory, to compare them to incoming values from the other table.
- When a large result set is sorted by the ORDER BY clause, each node sorts its portion of the result set in memory.
- The DISTINCT and UNION operators build in-memory data structures to represent all values found so far, to eliminate duplicates as the query progresses.

When the spill-to-disk feature is activated for a join node within a query, Impala does not produce any runtime filters for that join operation on that host. Other join nodes within the query are not affected.

The amount data depends on the portion of the data being handled by that host, and thus the operator may end up consuming different amounts of memory on different hosts.

How Impala handles scratch disk space for spilling:

Impala uses intermediate files during large sorts, joins, aggregations, or analytic function operations The files are removed when the operation finishes. By default, intermediate files are stored in the directory /tmp/impala-scratch.

You can specify locations of the intermediate files in one of the following ways:

- By starting the `impalad` daemon with the ##scratch_dirs="*PATH_TO_DIRECTORY*" configuration option.
- By specifying a different location in the Cloudera Manager in the Impala Daemon Scratch Directories field.

With either option above:

- You can specify a single directory or a comma-separated list of directories.
- You can specify an optional a capacity quota per scratch directory using the colon (:) as the delimiter.

  The capacity quota of -1 or 0 is the same as no quota for the directory.
- The scratch directories must be on the local filesystem, not in HDFS.
- You might specify different directory paths for different hosts, depending on the capacity and speed of the available storage devices.

If there is less than 1 GB free on the filesystem where that directory resides, Impala still runs, but writes a warning message to its log.

Impala successfully starts (with a warning written to the log) if it cannot create or read and write files in one of the scratch directories.

The following are examples for specifying scratch directories.

Config option	Description
--scratch_dirs=/dir1,/dir2	Use /dir1 and /dir2 as scratch directories with no capacity quota.
--scratch_dirs=/dir1,/dir2:25G	Use /dir1 and /dir2 as scratch directories with no capacity quota on /dir1 and the 25GB quota on /dir2.
--scratch_dirs=/dir1:5MB,/dir2	Use /dir1 and /dir2 as scratch directories with the capacity quota of 5MB on /dir1 and no quota on /dir2.
--scratch_dirs=/dir1:-1,/dir2:0	Use /dir1 and /dir2 as scratch directories with no capacity quota.

Allocation from a scratch directory will fail if the specified limit for the directory is exceeded.

Tip: Ensure the scratch directory size aligns with your available disk capacity. We recommend setting the scratch directory size to 100GB or larger, with 10GB being the absolute minimum. For example, you can configure it as --scratch_dirs=/dir1:40GB,/dir2:60GB to allocate a total of 100GB. The scratch directory size is particularly important for large queries that frequently spill data to disk. Insufficient capacity can result in a SCRATCH_ALLOCATION_FAILED error, causing query failures. To avoid this, adjust the scratch directory size based on your workload requirements.

If Impala encounters an error reading or writing files in a scratch directory during a query, Impala logs the error, and the query fails.

Memory usage for SQL operators:

The memory required to spill to disk is reserved up front, and you can examine it in the EXPLAIN plan when the EXPLAIN_LEVEL query option is set to 2 or higher.

If an operator accumulates more data than can fit in the reserved memory, it can either reserve more memory to continue processing data in memory or start spilling data to temporary scratch files on disk. Thus, operators with spill-to-disk support can adapt to different memory constraints by using however much memory is available to speed up execution, yet tolerate low memory conditions by spilling data to disk.

The amount of data depends on the portion of the data being handled by that host, and thus the operator may end up consuming different amounts of memory on different hosts.

Avoiding queries that spill to disk:

Because the extra I/O can impose significant performance overhead on these types of queries, try to avoid this situation by using the following steps:

1. Detect how often queries spill to disk, and how much temporary data is written. Refer to the following sources:
   - The output of the PROFILE command in the `impala-shell` interpreter. This data shows the memory usage for each host and in total across the cluster. The WriteIoBytes counter reports how much data was written to disk for each operator during the query.
   - In Impala Queries in Cloudera Manager, you can see the peak memory usage for a query, combined across all nodes in the cluster.
   - In the Queries tab in the Impala debug web user interface, select the query to examine and click the corresponding Profile link. This data breaks down the memory usage for a single host within the cluster, the host whose web interface you are connected to.
2. Use one or more techniques to reduce the possibility of the queries spilling to disk:
   - Increase the Impala memory limit if practical. For example, using the SET MEM_LIMIT SQL statement, increase the available memory by more than the amount of temporary data written to disk on a particular node.
   - Increase the number of nodes in the cluster, to increase the aggregate memory available to Impala and reduce the amount of memory required on each node.
   - On a cluster with resources shared between Impala and other Hadoop components, use resource management features to allocate more memory for Impala.
   - If the memory pressure is due to running many concurrent queries rather than a few memory-intensive ones, consider using the Impala admission control feature to lower the limit on the number of concurrent queries. By spacing out the most resource-intensive queries, you can avoid spikes in memory usage and improve overall response times.
   - Tune the queries with the highest memory requirements, using one or more of the following techniques:
     - Run the COMPUTE STATS statement for all tables involved in large-scale joins and aggregation queries.
     - Minimize your use of STRING columns in join columns. Prefer numeric values instead.
     - Examine the EXPLAIN plan to understand the execution strategy being used for the most resource-intensive queries.
     - If Impala still chooses a suboptimal execution strategy even with statistics available, or if it is impractical to keep the statistics up to date for huge or rapidly changing tables, add hints to the most resource-intensive queries to select the right execution strategy.
   - If your queries experience substantial performance overhead due to spilling, enable the DISABLE_UNSAFE_SPILLS query option. This option prevents queries whose memory usage is likely to be exorbitant from spilling to disk. As you tune problematic queries using the preceding steps, fewer and fewer will be cancelled by this option setting.

When to use DISABLE_UNSAFE_SPILLS:

The DISABLE_UNSAFE_SPILLS query option is suitable for an environment with ad hoc queries whose performance characteristics and memory usage are not known in advance. It prevents “worst-case scenario” queries that use large amounts of memory unnecessarily. Thus, you might turn this option on within a session while developing new SQL code, even though it is turned off for existing applications.

Organizations where table and column statistics are generally up-to-date might leave this option turned on all the time, again to avoid worst-case scenarios for untested queries or if a problem in the ETL pipeline results in a table with no statistics. Turning on DISABLE_UNSAFE_SPILLS lets you “fail fast” in this case and immediately gather statistics or tune the problematic queries.

Some organizations might leave this option turned off. For example, you might have tables large enough that the COMPUTE STATS takes substantial time to run, making it impractical to re-run after loading new data. If you have examined the EXPLAIN plans of your queries and know that they are operating efficiently, you might leave DISABLE_UNSAFE_SPILLS turned off. In that case, you know that any queries that spill will not go overboard with their memory consumption.

Limits on Query Size and Complexity

There are hard-coded limits on the maximum size and complexity of queries. Currently, the maximum number of expressions in a query is 2000. You might exceed the limits with large or deeply nested queries produced by business intelligence tools or other query generators.

If you have the ability to customize such queries or the query generation logic that produces them, replace sequences of repetitive expressions with single operators such as IN or BETWEEN that can represent multiple values or ranges. For example, instead of a large number of OR clauses:

```
WHERE val = 1 OR val = 2 OR val = 6 OR val = 100 ...
```

use a single IN clause:

```
WHERE val IN (1,2,6,100,...)
```

Scalability Considerations for Impala I/O

Impala parallelizes its I/O operations aggressively, therefore the more disks you can attach to each host, the better. Impala retrieves data from disk so quickly using bulk read operations on large blocks, that most queries are CPU-bound rather than I/O-bound.

Because the kind of sequential scanning typically done by Impala queries does not benefit much from the random-access capabilities of SSDs, spinning disks typically provide the most cost-effective kind of storage for Impala data, with little or no performance penalty as compared to SSDs.

Resource management features such as YARN, Llama, and admission control typically constrain the amount of memory, CPU, or overall number of queries in a high-concurrency environment. Currently, there is no throttling mechanism for Impala I/O.

Scalability Considerations for Table Layout

Due to the overhead of retrieving and updating table metadata in the Metastore database, try to limit the number of columns in a table to a maximum of approximately 2000. Although Impala can handle wider tables than this, the Metastore overhead can become significant, leading to query performance that is slower than expected based on the actual data volume.

To minimize overhead related to the Metastore database and Impala query planning, try to limit the number of partitions for any partitioned table to a few tens of thousands.

If the volume of data within a table makes it impractical to run exploratory queries, consider using the TABLESAMPLE clause to limit query processing to only a percentage of data within the table. This technique reduces the overhead for query startup, I/O to read the data, and the amount of network, CPU, and memory needed to process intermediate results during the query.

Kerberos-Related Network Overhead for Large Clusters

When Impala starts up, or after each kinit refresh, Impala sends a number of simultaneous requests to the KDC. For a cluster with 100 hosts, the KDC might be able to process all the requests within roughly 5 seconds. For a cluster

with 1000 hosts, the time to process the requests would be roughly 500 seconds. Impala also makes a number of DNS requests at the same time as these Kerberos-related requests.

While these authentication requests are being processed, any submitted Impala queries will fail. During this period, the KDC and DNS may be slow to respond to requests from components other than Impala, so other secure services might be affected temporarily.

Avoiding CPU Hotspots for HDFS Cached Data

You can use the HDFS caching feature to reduce I/O and memory-to-memory copying for frequently accessed tables or partitions.

To avoid hotspots, include the WITH REPLICATION clause with the CREATE TABLE or ALTER TABLE statements for tables that use HDFS caching. This clause allows more than one host to cache the relevant data blocks, so the CPU load can be shared, reducing the load on any one host.

The work for HDFS cached data is divided better among all the hosts that have cached replicas for a particular data block. When more than one host has a cached replica for a data block, Impala assigns the work of processing that block to whichever host has done the least work (in terms of number of bytes read) for the current query. If hotspots persist even with this load-based scheduling algorithm, you can enable the query option SCHEDULE_RANDOM_REPLICA=TRUE to further distribute the CPU load. This setting causes Impala to randomly pick a host to process a cached data block if the scheduling algorithm encounters a tie when deciding which host has done the least work.

Scalability Considerations for File Handle Caching

One scalability aspect that affects heavily loaded clusters is the load on the metadata layer from looking up the details as each file is opened. On HDFS, that can lead to increased load on the NameNode, and on object stores such as S3 or ABFS, this can lead to an excessive number of metadata requests. For example, a query that does a full table scan on a partitioned table may need to read thousands of partitions, each partition containing multiple data files. Accessing each column of a Parquet file also involves a separate “open” call, further increasing the load on the NameNode. High NameNode overhead can add startup time (that is, increase latency) to Impala queries, and reduce overall throughput for non-Impala workloads that also require accessing HDFS files.

You can reduce the number of calls made to your file system's metadata layer by enabling the file handle caching feature. Data files that are accessed by different queries, or even multiple times within the same query, can be accessed without a new “open” call and without fetching the file details multiple times.

Impala supports file handle caching for the following file systems:

- HDFS

  The cache_remote_file_handles flag controls local and remote file handle caching for an impalad. It is recommended that you use the default value of true as this caching prevents your NameNode from overloading when your cluster has many remote HDFS reads.
- S3

  The cache_s3_file_handles impalad flag controls the S3 file handle caching. The feature is enabled by default with the flag set to true.
- Ozone

  The cache_ozone_file_handles flag controls the Ozone file handle caching. This feature is ebnabled by default with the flag set to true.
- ABFS

  The cache_abfs_file_handles impalad flag controls the ABFS file handle caching. The feature is enabled by default with the flag set to true.

The feature is enabled by default with 20,000 file handles to be cached. To change the value, set the configuration option Maximum Cached File Handles (max_cached_file_handles) to a non-zero value for each Impala daemon (impalad). From the initial default value of 20000, adjust upward if NameNode request load is still significant, or downward if it is more important to reduce the extra memory usage on each host. Each cache entry consumes 6 KB, meaning that caching 20,000 file handles requires up to 120 MB on each Impala executor. The exact memory usage

varies depending on how many file handles have actually been cached; memory is freed as file handles are evicted from the cache.

If a manual operation moves a file to the trashcan while the file handle is cached, Impala still accesses the contents of that file. This is a change from prior behavior. Previously, accessing a file that was in the trashcan would cause an error. This behavior only applies to non-Impala methods of removing files, not the Impala mechanisms such as TRUN CATE TABLE or DROP TABLE.

If files are removed, replaced, or appended by operations outside of Impala, the way to bring the file information up to date is to run the REFRESH statement on the table.

File handle cache entries are evicted as the cache fills up, or based on a timeout period when they have not been accessed for some time.

To evaluate the effectiveness of file handle caching for a particular workload, issue the PROFILE statement in `impala-shell` or examine query profiles in the Impala Web UI. Look for the ratio of CachedFileHandlesHitCount (ideally, should be high) to CachedFileHandlesMissCount (ideally, should be low). Before starting any evaluation, run several representative queries to “warm up” the cache because the first time each data file is accessed is always recorded as a cache miss.

To see metrics about file handle caching for each `impalad` instance, examine the following fields on the /metrics page in the Impala Web UI:

- impala-server.io.mgr.cached-file-handles-miss-count
- impala-server.io.mgr.num-cached-file-handles

Scaling Limits and Guidelines

Consider and respect the scalability limitations provided in this topic in order to achieve optimal scalability and performance in Impala. For example, while you might be able to create a table with 2000 columns, you will experience performance problems while querying the table. This topic does not cover functional limitations in Impala.

Unless noted otherwise, the limits listed in this topic were tested and certified.

The limits noted as "generally safe" are not certified, but recommended as generally safe. A safe range is not a hard limit as unforeseen errors or troubles in your particular environment can affect the range.

Deployment Limits

- Number of Impalad Executors: 500 nodes depending on use case
- Number of Impalad Coordinators: 1 coordinator for at most every 50 executors

Data Storage Limits

There are no hard limits for the following, but you will experience gradual performance degradation as you increase these numbers.

- Number of databases
- Number of tables - total, per database
- Number of partitions - total, per table
- Number of files - total, per table, per table per partition
- Number of views - total, per database
- Number of user-defined functions - total, per database
- Parquet
  - Number of columns per row group
  - Number of row groups per block
  - Number of HDFS blocks per file

Schema Design Limits

- Number of columns
  - 300 for Kudu tables
  - 1000 for other types of tables

Security Limits

- Number of roles: 10,000

Query Limits - Compile Time

- Maximum number of columns in a query, included in a SELECT list, INSERT, and in an expression: no limit
- Number of tables referenced: no limit
- Number of plan nodes: no limit
- Number of plan fragments: no limit
- Depth of expression tree: 1000 hard limit
- Width of expression tree: 10,000 hard limit

- Codegen: Very deeply nested expressions within queries can exceed internal Impala limits, leading to excessive memory usage. Setting the query option disable_codegen=true may reduce the impact, at a cost of longer query runtime.

Dedicated Coordinator

When scalability bottlenecks occur, you can assign one dedicated role to each Impala daemon host, either as a coordinator or as an executor, to address the issues.

Each host that runs the Impala Daemon acts as both a coordinator and as an executor, by default, managing metadata caching, query compilation, and query execution. In this configuration, Impala clients can connect to any Impala daemon and send query requests.

During highly concurrent workloads for large-scale queries, the dual roles can cause scalability issues because:

- The extra work required for a host to act as the coordinator could interfere with its capacity to perform other work for the later phases of the query. For example, coordinators can experience significant network and CPU overhead with queries containing a large number of query fragments. Each coordinator caches metadata for all table partitions and data files, which requires coordinators to be configured with a large JVM heap. Executor-only Impala daemons should be configured with the default JVM heaps, which leaves more memory available to process joins, aggregations, and other operations performed by query executors.
- Having a large number of hosts act as coordinators can cause unnecessary network overhead, or even timeout errors, as each of those hosts communicates with the StateStore daemon for metadata updates.
- The "soft limits" imposed by the admission control feature are more likely to be exceeded when there are a large number of heavily loaded hosts acting as coordinators.

The following factors can further exacerbate the above issues:

- High number of concurrent query fragments due to query concurrency and/or query complexity
- Large metadata topic size related to the number of partitions/files/blocks
- High number of coordinator nodes
- High number of coordinators used in the same resource pool

If such scalability bottlenecks occur, you can assign one dedicated role to each Impala daemon host, either as a coordinator or as an executor, to address the issues.

- All explicit or load-balanced client connections must go to the coordinator hosts. These hosts perform the network communication to keep metadata up-to-date and route query results to the appropriate clients. The dedicated

coordinator hosts do not participate in I/O-intensive operations such as scans, and CPU-intensive operations such as aggregations.
- The executor hosts perform the intensive I/O, CPU, and memory operations that make up the bulk of the work for each query. The executors do communicate with the Statestored daemon for membership status, but the dedicated executor hosts do not process the final result sets for queries.

Using dedicated coordinators offers the following benefits:

- Reduces memory usage by limiting the number of Impala nodes that need to cache metadata.
- Provides better concurrency by avoiding coordinator bottleneck.
- Eliminates query over-admission.
- Reduces resource, especially network, utilization on the StateStore daemon by limiting metadata broadcast to a subset of nodes.
- Improves reliability and performance for highly concurrent workloads by reducing workload stress on coordinators. Dedicated coordinators require 50% or fewer connections and threads.
- Reduces the number of explicit metadata refreshes required.
- Improves diagnosability if a bottleneck or other performance issue arises on a specific host, you can narrow down the cause more easily because each host is dedicated to specific operations within the overall Impala workload.

In this configuration with dedicated coordinators / executors, you cannot connect to the dedicated executor hosts through clients such as impala-shell or business intelligence tools as only the coordinator nodes support client connections.

Determining the Optimal Number of Dedicated Coordinators

You should have the smallest number of coordinators that will still satisfy your workload requirements in a cluster. A rough estimation is 1 coordinator for every 50 executors.

To maintain a healthy state and optimal performance, it is recommended that you keep the peak utilization of all resources used by Impala, including CPU, the number of threads, the number of connections, and RPCs, under 80%.

Consider the following factors to determine the right number of coordinators in your cluster:

- What is the number of concurrent queries?
- What percentage of the workload is DDL?
- What is the average query resource usage at the various stages (merge, runtime filter, result set size, etc.)?
- How many Impala Daemons (impalad) is in the cluster?
- Is there a high availability requirement?
- Compute/storage capacity reduction factor

Start with the below set of steps to determine the initial number of coordinators:

1. If your cluster has less than 10 nodes, we recommend that you configure one dedicated coordinator. Deploy the dedicated coordinator on a DataNode to avoid losing storage capacity. In most of the cases, one dedicated coordinator is enough to support all workloads on a cluster.
2. Add more coordinators if the dedicated coordinator CPU or network peak utilization is 80% or higher. You might need 1 coordinator for every 50 executors.
3. If the Impala service is shared by multiple workgroups with a dynamic resource pool assigned, use one coordinator per pool to avoid admission control over admission.
4. If high availability is required, double the number of coordinators. One set as an active set and the other as a backup set.

Advance Tuning

Use the following guidelines to further tune the throughput and stability.

1. The concurrency of DML statements does not typically depend on the number of coordinators or size of the cluster. Queries that return large result sets (10,000+ rows) consume more CPU and memory resources on the coordinator. Add one or two coordinators if the workload has many such queries.

2. DDL queries, excluding COMPUTE STATS and CREATE TABLE AS SELECT, are executed only on coordinators. If your workload contains many DDL queries running concurrently, you could add one coordinator.
3. The CPU contention on coordinators can slow down query executions when concurrency is high, especially for very short queries (<10s). Add more coordinators to avoid CPU contention.
4. On a large cluster with 50+ nodes, the number of network connections from a coordinator to executors can grow quickly as query complexity increases. The growth is much greater on coordinators than executors. Add a few more coordinators if workloads are complex, i.e. (an average number of fragments * number of Impalad) > 500, but with the low memory/CPU usage to share the load. Watch IMPALA-4603 and IMPALA-7213 to track the progress on fixing this issue.
5. When using multiple coordinators for DML statements, divide queries to different groups (number of groups = number of coordinators). Configure a separate dynamic resource pool for each group and direct each group of query requests to a specific coordinator. This is to avoid query over admission.
6. The front-end connection requirement is not a factor in determining the number of dedicated coordinators. Consider setting up a connection pool at the client side instead of adding coordinators. For a short-term solution, you could increase the value of fe_service_threads on coordinators to allow more client connections.
7. In general, you should have a very small number of coordinators so storage capacity reduction is not a concern. On a very small cluster (less than 10 nodes), deploy a dedicated coordinator on a DataNode to avoid storage capacity reduction.

Estimating Coordinator Resource Usage

Resource	Safe range	Notes / CM tsquery to monitor
Memory	(Max JVM heap setting + query concurrency * query mem_limit) <= 80% of Impala process memory allocation	Memory usage: SELECT mem_rss WHERE entityName = "Coordinator Instance ID" AND category = ROLE JVM heap usage (metadata cache): SELECT impala_jvm_heap_current_usage_bytes WHERE entityName = "Coordinator Instance ID" AND category = ROLE (only in release 5.15 and above)
TCP Connection	Incoming + outgoing < 16K	Incoming connection usage: SELECT thrift_server_backend_connections_in_use WHERE entityName = "Coordinator Instance ID" AND category = ROLE Outgoing connection usage: SELECT backends_client_cache_clients_in_use WHERE entityName = "Coordinator Instance ID" AND category = ROLE
Threads	< 32K	SELECT thread_manager_running_threads WHERE entityName = "Coordinator Instance ID" AND category = ROLE

CPU	Concurrency = non-DDL query concurrency <= number of virtual cores allocated to Impala per node	CPU usage estimation should be based on how many cores are allocated to Impala per node, not a sum of all cores of the cluster. It is recommended that concurrency should not be more than the number of virtual cores allocated to Impala per node. Query concurrency: SELECT total_impala_num_queries_registered_across_impalads WHERE entityName = "IMPALA-1" AND category = SERVICE

If usage of any of the above resources exceeds the safe range, add one more coordinator.

Monitoring Coordinator Resource Usage

Using Cloudera Manager, monitor the coordinator resource usage to understand your workload and adjust the number of coordinators according to the guidelines above. The available options are:

- Impala Queries tab: Monitor such attributes as DDL queries and Rows produced.
- Custom charts: Monitor activities, such as query complexity which is an average fragment count per query (total fragments / total queries).
- tsquery: Build the custom charts to monitor and estimate the amount of resource the coordinator needs.

The following are sample queries for common resource usage monitoring. Replace entityName values with your coordinator instance id.

Per coordinator tsquery

Resource Usage	Tsquery
Memory usage	SELECT impala_memory_total_used, mem_tracker_process_limit WHERE entityName = "Coordinator Instance ID" AND category = ROLE
JVM heap usage (metadata cache)	SELECT impala_jvm_heap_current_usage_bytes WHERE entityName = "Coordinator Instance ID" AND category = ROLE (only in release 5.15 and above)
CPU usage	SELECT cpu_user_rate / getHostFact(numCores, 1) * 100, cpu_system_rate / getHostFact(numCores, 1) * 100 WHERE entityName="Coordinator Instance ID"
Network usage (host level)	SELECT total_bytes_receive_rate_across_network_interfaces, total_bytes_transmit_rate_across_network_interfaces WHERE entityName="Coordinator Instance ID"
Incoming connection usage	SELECT thrift_server_backend_connections_in_use WHERE entityName = "Coordinator Instance ID" AND category = ROLE
Outgoing connection usage	SELECT backends_client_cache_clients_in_use WHERE entityName = "Coordinator Instance ID" AND category = ROLE
Thread usage	SELECT thread_manager_running_threads WHERE entityName = "Coordinator Instance ID" AND category = ROLE

Cluster wide tsquery

Resource usage	Tsquery
Front-end connection usage	SELECT total_thrift_server_beeswax_frontend_connections_in_use_across_impalads, total_thrift_server_hiveserver2_frontend_connections_in_use_across_impalads
Query concurrency	SELECT total_impala_num_queries_registered_across_impalads WHERE entityName = "IMPALA-1" AND category = SERVICE

Related Information

Managing Metadata in Impala

Hadoop File Formats Support

Impala supports a number of file formats used in Apache Hadoop.

Impala can load and query data files produced by other Hadoop components such as Spark, and data files produced by Impala can be used by other components also. The following sections discuss the procedures, limitations, and performance considerations for using each file format with Impala.

The file format used for an Impala table has significant performance consequences. Some file formats include compression support that affects the size of data on the disk and, consequently, the amount of I/O and CPU resources required to deserialize data. The amounts of I/O and CPU resources required can be a limiting factor in query performance since querying often begins with moving and decompressing data. To reduce the potential impact of this part of the process, data is often compressed. By compressing data, a smaller total number of bytes are transferred from disk to memory. This reduces the amount of time taken to transfer the data, but a tradeoff occurs when the CPU decompresses the content.

For the file formats that Impala cannot write to, create the table from within Impala whenever possible and insert data using another component such as Hive or Spark. See the table below for specific file formats.

The following table lists the file formats that Impala supports.

File Type	Format	Compression Codecs	Impala Can CREATE?	Impala Can INSERT?
Parquet	Structured	Snappy, gzip, zstd, LZ4; currently Snappy by default	Yes.	Yes. CREATE TABLE, INSERT, LOAD DATA, and query.
ORC	Structured	gzip, Snappy, LZ4; currently gzip by default	Yes. By default, ORC reads are enabled in Impala 3.4.0 and higher. To disable, set--enable_orc_scanner to false when starting the cluster.	No. Import data by using LOAD DATA on data files already in the right format, or use INSERT in Hive followed by REFRESH *TABLE_NAME* in Impala.
Text	Unstructured	bzip2, deflate, gzip, Snappy,zstd	Yes. CREATE TABLE with no STORED AS clause. The default file format is uncompressed text, with values separated by ASCII 0x01 characters (typically represented as Ctrl-A).	Yes if uncompressed. CREATE TABLE, INSERT, LOAD DATA, and query. No if compressed. If other kinds of compression are used, you must load data through LOAD DATA, Hive, or manually in HDFS.
Avro	Structured	Snappy, gzip, deflate	Yes.	No. Import data by using LOAD DATA on data files already in the right format, or use INSERT in Hive followed by REFRESH *TABLE_NAME* in Impala.
RCFile	Structured	Snappy, gzip, deflate, bzip2	Yes.	No. Import data by using LOAD DATA on data files already in the right format, or use INSERT in Hive followed by REFRESH *TABLE_NAME* in Impala.

File Type	Format	Compression Codecs	Impala Can CREATE?	Impala Can INSERT?
SequenceFile	Structured	Snappy, gzip, deflate, bzip2	Yes.	No. Import data by using LOAD DATA on data files already in the right format, or use INSERT in Hive followed by REFRESH *TABLE_NAME* in Impala.

Impala supports the following compression codecs:

Snappy

Recommended for its effective balance between compression ratio and decompression speed. Snappy compression is very fast, but gzip provides greater space savings. Supported for text, RC, Sequence, and Avro files in Impala 2.0 and higher.

Gzip

Recommended when achieving the highest level of compression (and therefore greatest disk-space savings) is desired. Supported for text, RC, Sequence and Avro files in Impala 2.0 and higher.

Deflate

Supported for AVRO, RC, Sequence, and text files.

Bzip2

Supported for text, RC, and Sequence files in Impala 2.0 and higher.

Zstd

For Parquet and text files only.

Lz4

For Parquet files only.

Choosing the File Format for a Table

Different file formats and compression codecs work better for different data sets. Choosing the proper format for your data can yield performance improvements. Use the following considerations to decide which combination of file format and compression to use for a particular table.

- If you are working with existing files that are already in a supported file format, use the same format for the Impala table if performance is acceptable. If the original format does not yield acceptable query performance or resource usage, consider creating a new Impala table with different file format or compression characteristics, and doing a one-time conversion by rewriting the data to the new table.
- Text files are convenient to produce through many different tools and are human-readable for ease of verification and debugging. Those characteristics are why text is the default format for an Impala CREATE TABLE statement. However, when performance and resource usage are the primary considerations, use one of the structured file formats that include metadata and built-in compression.

  A typical workflow might involve bringing data into an Impala table by copying CSV or TSV files into the appropriate data directory, and then using the INSERT ... SELECT syntax to rewrite the data into a table using a different, more compact file format.

Using Text Data Files

Impala supports using text files as the storage format for input and output. Text files are a convenient format to use for interchange with other applications or scripts that produce or read delimited text files, such as CSV or TSV with commas or tabs for delimiters.

Text files are flexible in their column definitions. For example, a text file could have more fields than the Impala table, and those extra fields are ignored during queries. Or it could have fewer fields than the Impala table, and those missing fields are treated as NULL values in queries.

You could have fields that were treated as numbers or timestamps in a table, then use ALTER TABLE ... REPLACE COLUMNS to switch them to strings, or the reverse.

Creating Text Tables

You can create tables with specific separator characters to import text files in familiar formats such as CSV, TSV, or pipe-separated with the FIELDS TERMINATED BY clause preceded by the ROW FORMAT DELIMITED clause. For example:

```
CREATE TABLE tsv(id INT, s STRING, n INT, t TIMESTAMP, b BOOLEAN)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;
```

You can specify a delimiter character '\0' to use the ASCII 0 (nul) character for text tables.

You can also use these tables to produce output data files, by copying data into them through the INSERT ... SELECT syntax and then extracting the data files from the Impala data directory.

The data files created by any INSERT statements uses the Ctrl-A character (hex 01) as a separator between each column value.

Issue a DESCRIBE FORMATTED *TABLE_NAME* statement to see the details of how each table is represented internally in Impala.

Complex type considerations: Although you can create tables in this file format using the complex types (ARRAY, STRUCT, and MAP), currently, Impala cannot query these types in text tables.

Data Files for Text Tables

When Impala queries a table with data in text format, it consults all the data files in the data directory for that table, with some exceptions:

- Impala ignores any hidden files, that is, files whose names start with a dot or an underscore.
- Impala queries ignore files with extensions commonly used for temporary work files by Hadoop tools. Any files with extensions .tmp or .copying are not considered part of the Impala table. The suffix matching is case-insensitive, so for example Impala ignores both .copying and .COPYING suffixes.
- Impala uses suffixes to recognize when text data files are compressed text. For Impala to recognize the compressed text files, they must have the appropriate file extension corresponding to the compression codec, either .bz2, .gz, .snappy, or .zst, .deflate. The extensions can be in uppercase or lowercase.
- Otherwise, the file names are not significant. When you put files into an HDFS directory through ETL jobs, or point Impala to an existing HDFS directory with the CREATE EXTERNAL TABLE statement, or move data files under external control with the LOAD DATA statement, Impala preserves the original filenames.

An INSERT ... SELECT statement produces one data file from each node that processes the SELECT part of the statement. An INSERT ... VALUES statement produces a separate data file for each statement; because Impala is more efficient querying a small number of huge files than a large number of tiny files, the INSERT ... VALUES syntax is not recommended for loading a substantial volume of data. If you find yourself with a table that is inefficient due to too many small data files, reorganize the data into a few large files by doing INSERT ... SELECT to transfer the data to a new table.

Do not surround string values with quotation marks in text data files that you construct. If you need to include the separator character inside a field value, for example to put a string value with a comma inside a CSV-format data file, specify an escape character on the CREATE TABLE statement with the ESCAPED BY clause, and insert that character immediately before any separator characters that need escaping.

Special values within text data files:

- Impala recognizes the literal strings inf for infinity and nan for “Not a Number”, for FLOAT and DOUBLE columns.
- Impala recognizes the literal string \N to represent NULL. When using Sqoop, specify the options --null-non-s tring and --null-string to ensure all NULL values are represented correctly in the Sqoop output files. \N needs to be escaped as in the below example:

```
--null-string '\\N' --null-non-string '\\N'
```

- By default, Sqoop writes NULL values using the string null, which causes a conversion error when such rows are evaluated by Impala. (A workaround for existing tables and data files is to change the table properties through ALTER TABLE *NAME* SET TBLPROPERTIES("serialization.null.format"="null").)
- Impala can optionally skip an arbitrary number of header lines from text input files on HDFS based on the skip .header.line.count value in the TBLPROPERTIES field of the table metadata.

Loading Data into Text Tables

To load an existing text file into an Impala text table, use the LOAD DATA statement and specify the path of the file in HDFS. That file is moved into the appropriate Impala data directory.

To load multiple existing text files into an Impala text table, use the LOAD DATA statement and specify the HDFS path of the directory containing the files. All non-hidden files are moved into the appropriate Impala data directory.

Use the DESCRIBE FORMATTED statement to see the HDFS directory where the data files are stored, then use Linux commands such as hdfs dfs -ls *HDFS_DIRECTORY* and hdfs dfs -cat *HDFS_FILE* to display the contents of an Impala-created text file.

When you create a text file for use with an Impala text table, specify \N to represent a NULL value.

If a text file has fewer fields than the columns in the corresponding Impala table, all the corresponding columns are set to NULL when the data in that file is read by an Impala query.

If a text file has more fields than the columns in the corresponding Impala table, the extra fields are ignored when the data in that file is read by an Impala query.

You can also use manual HDFS operations such as hdfs dfs -put or hdfs dfs -cp to put data files in the data directory for an Impala table. When you copy or move new data files into the HDFS directory for the Impala table, issue a REFRESH *TABLE_NAME* statement in `impala-shell` before issuing the next query against that table, to make Impala recognize the newly added files.

Query Performance for Text Tables

Data stored in text format is relatively bulky, and not as efficient to query as binary formats such as Parquet. For the tables used in your most performance-critical queries, look into using more efficient alternate file formats.

For frequently queried data, you might load the original text data files into one Impala table, then use an INSERT statement to transfer the data to another table that uses the Parquet file format. The data is converted automatically as it is stored in the destination table.

For more compact data. consider using text data compressed in the gzip, bzip2, or Snappy formats. However, these compressed formats are not “splittable” so there is less opportunity for Impala to parallelize queries on them. You also have the choice to convert the data to Parquet using an INSERT ... SELECT statement to copy the original data into a Parquet table.

Using bzip2, deflate, gzip, Snappy-Compressed, or zstd Text Files

Impala supports using text data files that employ bzip2, deflate, gzip, Snappy, or zstd compression. These compression types are primarily for convenience within an existing ETL pipeline rather than maximum performance. Although it requires less I/O to read compressed text than the equivalent uncompressed text, files compressed by

these codecs are not “splittable” and therefore cannot take full advantage of the Impala parallel query capability. Impala can read compressed text files written by Hive.

As each Snappy-compressed file is processed, the node doing the work reads the entire file into memory and then decompresses it. Therefore, the node must have enough memory to hold both the compressed and uncompressed data from the text file. The memory required to hold the uncompressed data is difficult to estimate in advance, potentially causing problems on systems with low memory limits or with resource management enabled. This memory overhead is reduced for bzip2-, deflate-, gzip-, and zstd-compressed text files. The compressed data is decompressed as it is read, rather than all at once.

To create a table to hold compressed text, create a text table with no special compression options. Specify the delimiter and escape character if required, using the ROW FORMAT clause.

Because Impala can query compressed text files but currently cannot write them, produce the compressed text files outside Impala and use the LOAD DATA statement, manual HDFS commands to move them to the appropriate Impala data directory. (Or, you can use CREATE EXTERNAL TABLE and point the LOCATION attribute at a directory containing existing compressed text files.)

Using Parquet Data Files

Impala allows you to create, manage, and query Parquet tables. Parquet is a column-oriented binary file format intended to be highly efficient for the types of large-scale queries.

Parquet is suitable for queries scanning particular columns within a table, for example, to query “wide” tables with many columns, or to perform aggregation operations such as SUM() and AVG() that need to process most or all of the values from a column.

Each Parquet data file written by Impala contains the values for a set of rows (referred to as the “row group”). Within a data file, the values from each column are organized so that they are all adjacent, enabling good compression for the values from that column. Queries against a Parquet table can retrieve and analyze these values from any column quickly and with minimal I/O.

Creating Parquet Tables

To create a table in the Parquet format, use the STORED AS PARQUET clause in the CREATE TABLE statement. For example:

```
CREATE TABLE PARQUET_TABLE_NAME (x INT, y STRING) STORED AS PARQUET;
```

Or, to clone the column names and data types of an existing table, use the LIKE with the STORED AS PARQUET clause. For example:

```
CREATE TABLE PARQUET_TABLE_NAME LIKE OTHER_TABLE_NAME STORED AS PARQUET;
```

You can derive column definitions from a raw Parquet data file, even without an existing Impala table. For example, you can create an external table pointing to an HDFS directory, and base the column definitions on one of the files in that directory:

```
CREATE EXTERNAL TABLE ingest_existing_files LIKE PARQUET '/user/etl/destinat
ion/datafile1.dat'
  STORED AS PARQUET
  LOCATION '/user/etl/destination';
```

Or, you can refer to an existing data file and create a new empty table with suitable column definitions. Then you can use INSERT to create new data files or LOAD DATA to transfer existing data files into the new table.

```
CREATE TABLE columns_from_data_file LIKE PARQUET '/user/etl/destination/data
file1.dat'
  STORED AS PARQUET;
```

In this example, the new table is partitioned by year, month, and day. These partition key columns are not part of the data file, so you specify them in the CREATE TABLE statement:

```
CREATE TABLE columns_from_data_file LIKE PARQUET '/user/etl/destination/data
file1.dat'
  PARTITION (year INT, month TINYINT, day TINYINT)
  STORED AS PARQUET;
```

If the Parquet table has a different number of columns or different column names than the other table, specify the names of columns from the other table rather than * in the SELECT statement.

Data Type Considerations for Parquet Tables

The Parquet format defines a set of data types whose names differ from the names of the corresponding Impala data types. If you are preparing Parquet files using other Hadoop components such as Pig or MapReduce, you might need to work with the type names defined by Parquet. The following tables list the Parquet-defined types and the equivalent types in Impala.

Primitive types

Parquet type	Impala type
BINARY	STRING
BOOLEAN	BOOLEAN
DOUBLE	DOUBLE
FLOAT	FLOAT
INT32	INT
INT64	BIGINT
INT96	TIMESTAMP

Logical types

Parquet uses type annotations to extend the types that it can store, by specifying how the primitive types should be interpreted.

Parquet primitive type and annotation	Impala type
BINARY annotated with the UTF8 OriginalType	STRING
BINARY annotated with the STRING LogicalType	STRING
BINARY annotated with the ENUM OriginalType	STRING
BINARY annotated with the DECIMAL OriginalType	DECIMAL
INT64 annotated with the TIMESTAMP_MILLIS OriginalType	TIMESTAMP or BIGINT (for backward compatibility)
INT64 annotated with the TIMESTAMP_MICROS OriginalType	TIMESTAMP or BIGINT (for backward compatibility)
INT64 annotated with the TIMESTAMP LogicalType	TIMESTAMP or

Parquet primitive type and annotation	Impala type
	BIGINT (for backward compatibility)

Complex types:

Impala only supports queries against the complex types (ARRAY, MAP, and STRUCT) in Parquet tables.

Loading Data into Parquet Tables

Choose from the following process to load data into Parquet tables based on whether the original data is already in an Impala table, or exists as raw data files outside Impala.

If you already have data in an Impala or Hive table, perhaps in a different file format or partitioning scheme:

- Transfer the data to a Parquet table using the Impala INSERT...SELECT statement.

  For example:

  ```
  INSERT OVERWRITE TABLE parquet_table_name SELECT * FROM other_table_name;
  ```

  You can convert, filter, repartition, and do other things to the data as part of this same INSERT statement.

  When inserting into partitioned tables, especially using the Parquet file format, you can include a hint in the INSERT statement to fine-tune the overall performance of the operation and its resource usage.

Any INSERT statement for a Parquet table requires enough free space in the HDFS filesystem to write one block. Because Parquet data files use a block size of 1 GB by default, an INSERT might fail (even for a very small amount of data) if your HDFS is running low on space.

Avoid the INSERT...VALUES syntax for Parquet tables, because INSERT...VALUES produces a separate tiny data file for each INSERT...VALUES statement, and the strength of Parquet is in its handling of data (compressing, parallelizing, and so on) in large chunks.

If you have one or more Parquet data files produced outside of Impala, you can quickly make the data query-able through Impala by one of the following methods:

- The LOAD DATA statement moves a single data file or a directory full of data files into the data directory for an Impala table. It does no validation or conversion of the data.

  The original data files must be somewhere in HDFS, not the local filesystem.
- The CREATE TABLE statement with the LOCATION clause creates a table where the data continues to reside outside the Impala data directory.

  The original data files must be somewhere in HDFS, not the local filesystem.

  For extra safety, if the data is intended to be long-lived and reused by other applications, you can use the CREATE EXTERNAL TABLE syntax so that the data files are not deleted by an Impala DROP TABLE statement.
- If the Parquet table already exists, you can copy Parquet data files directly into it using the hadoop distcp -pb command, then use the REFRESH statement to make Impala recognize the newly added data.

  You must preserve the block size of the Parquet data files by using the hadoop distcp -pb command rather than a -put or -cp operation on the Parquet files.

Note:

Currently, Impala always decodes the column data in Parquet files based on the ordinal position of the columns, not by looking up the position of each column based on its name. Parquet files produced outside of Impala must write column data in the same order as the columns are declared in the Impala table. Any optional columns that are omitted from the data files must be the rightmost columns in the Impala table definition.

If you created compressed Parquet files through some tool other than Impala, make sure that any compression codecs are supported in Parquet by Impala. For example, Impala does not currently support LZO compression in Parquet files. Also doublecheck that you used any recommended compatibility settings in the other tool, such as spark.sql.parquet.binaryAsString when writing Parquet files through Spark.

Recent versions of Sqoop can produce Parquet output files using the --as-parquetfile option.

If the data exists outside Impala and is in some other format, combine both of the preceding techniques. First, use a LOAD DATA or CREATE EXTERNAL TABLE ... LOCATION statement to bring the data into an Impala table that uses the appropriate file format. Then, use an INSERT...SELECT statement to copy the data to the Parquet table, converting to Parquet format as part of the process.

Loading data into Parquet tables is a memory-intensive operation, because the incoming data is buffered until it reaches one data block in size, then that chunk of data is organized and compressed in memory before being written out. The memory consumption can be larger when inserting data into partitioned Parquet tables, because a separate data file is written for each combination of partition key column values, potentially requiring several large chunks to be manipulated in memory at once.

When inserting into a partitioned Parquet table, Impala redistributes the data among the nodes to reduce memory consumption. You might still need to temporarily increase the memory dedicated to Impala during the insert operation, or break up the load operation into several INSERT statements, or both.

Query Performance for Parquet Tables

Query performance for Parquet tables depends on the number of columns needed to process the SELECT list and WHERE clauses of the query, the way data is divided into large data files with block size equal to file size, the reduction in I/O by reading the data for each column in compressed format, which data files can be skipped (for partitioned tables), and the CPU overhead of decompressing the data for each column.

For example, the following is an efficient query for a Parquet table:

```
SELECT AVG(income) FROM census_data WHERE state = 'CA';
```

The query processes only 2 columns out of a large number of total columns. If the table is partitioned by the STATE column, it is even more efficient because the query only has to read and decode 1 column from each data file, and it can read only the data files in the partition directory for the state 'CA', skipping the data files for all the other states, which will be physically located in other directories.

The following is a relatively inefficient query for a Parquet table:

```
SELECT * FROM census_data;
```

Impala would have to read the entire contents of each large data file, and decompress the contents of each column for each row group, negating the I/O optimizations of the column-oriented format. This query might still be faster for a Parquet table than a table with some other file format, but it does not take advantage of the unique strengths of Parquet data files.

Impala can optimize queries on Parquet tables, especially join queries, better when statistics are available for all the tables. Issue the COMPUTE STATS statement for each table after substantial amounts of data are loaded into or appended to it.

The runtime filtering feature works best with Parquet tables. The per-row filtering aspect only applies to Parquet tables.

Impala queries are optimized for files stored in Amazon S3. For Impala tables that use the Parquet file formats, the PARQUET_OBJECT_STORE_SPLIT_SIZE query option determines how Impala divides the I/O work of reading the data files. By default, this value is 256 MB. Impala parallelizes S3 read operations on the files as if they were made up of 256 MB blocks to match the row group size produced by Impala.

Parquet files written by Impala include embedded metadata specifying the minimum and maximum values for each column, within each row group and each data page within the row group. Impala-written Parquet files typically contain a single row group; a row group can contain many data pages. Impala uses this information (currently, only the metadata for each row group) when reading each Parquet data file during a query, to quickly determine whether each row group within the file potentially includes any rows that match the conditions in the WHERE clause.

For example, if the column X within a particular Parquet file has a minimum value of 1 and a maximum value of 100, then a query including the clause WHERE x > 200 can quickly determine that it is safe to skip that particular file, instead of scanning all the associated column values.

This optimization technique is especially effective for tables that use the SORT BY clause for the columns most frequently checked in WHERE clauses, because any INSERT operation on such tables produces Parquet data files with relatively narrow ranges of column values within each file.

To disable Impala from writing the Parquet page index when creating Parquet files, set the PARQUET_WRITE_PAGE_INDEX query option to FALSE.

Partitioning for Parquet Tables

Partitioning is an important performance technique for Impala generally. This section explains some of the performance considerations for partitioned Parquet tables.

The Parquet file format is ideal for tables containing many columns, where most queries only refer to a small subset of the columns. The physical layout of Parquet data files lets Impala read only a small fraction of the data for many queries. The performance benefits of this approach are amplified when you use Parquet tables in combination with partitioning. Impala can skip the data files for certain partitions entirely, based on the comparisons in the WHERE clause that refer to the partition key columns. For example, queries on partitioned tables often analyze data for time intervals based on columns such as YEAR, MONTH, and/or DAY, or for geographic regions.

As Parquet data files use a large block size, when deciding how finely to partition the data, try to find a granularity where each partition contains 256 MB or more of data, rather than creating a large number of smaller files split among many partitions.

Inserting into a partitioned Parquet table can be a resource-intensive operation, because each Impala node could potentially be writing a separate data file to HDFS for each combination of different values for the partition key columns. The large number of simultaneous open files could exceed the HDFS “transceivers” limit. To avoid exceeding this limit, consider the following techniques:

- Load different subsets of data using separate INSERT statements with specific values for the PARTITION clause, such as PARTITION (year=2010).
- Increase the “transceivers” value for HDFS, sometimes spelled “xcievers” (sic). The property value in the hdfs-site.xml configuration file is dfs.datanode.max.transfer.threads.

  For example, if you were loading 12 years of data partitioned by year, month, and day, even a value of 4096 might not be high enough.
- Use the COMPUTE STATS statement to collect column statistics on the source table from which data is being copied, so that the Impala query can estimate the number of different values in the partition key columns and distribute the work accordingly.

Enabling Compression for Parquet Tables

When Impala writes Parquet data files using the INSERT statement, the underlying compression is controlled by the COMPRESSION_CODEC query option. The allowed values for this query option are snappy (the default), gzip, zstd, lz4, and none, the compression codecs that Impala supports for Parquet.

Snappy

By default, the underlying data files for a Parquet table are compressed with Snappy. The combination of fast compression and decompression makes it a good choice for many data sets.

GZip

If you need more intensive compression (at the expense of more CPU cycles for uncompressing during queries), set the COMPRESSION_CODEC query option to gzip before inserting the data.

Zstd

Zstd is a real-time compression algorithm offering a tradeoff between speed and ratio of compression. Compression levels from 1 up to 22 are supported. The lower the level, the faster the speed at the cost of compression ratio.

Lz4

Lz4 is a lossless compression algorithm providing extremely fast and scalable compression and decompression.

None

If your data compresses very poorly, or you want to avoid the CPU overhead of compression and decompression entirely, set the COMPRESSION_CODEC query option to none before inserting the data.

The actual compression ratios, and relative insert and query speeds, will vary depending on the characteristics of the actual data.

Because Parquet data files are typically large, each directory will have a different number of data files and the row groups will be arranged differently.

At the same time, the less aggressive the compression, the faster the data can be decompressed.

For example, using a table with a billion rows, switching from Snappy to GZip compression shrinks the data by an additional 40% or so, while switching from Snappy compression to no compression expands the data also by about 40%. A query that evaluates all the values for a particular column runs faster with no compression than with Snappy compression, and faster with Snappy compression than with Gzip compression.

The data files using the various compression codecs are all compatible with each other for read operations. The metadata about the compression format is written into each data file, and can be decoded during queries regardless of the COMPRESSION_CODEC setting in effect at the time.

Exchanging Parquet Data Files with Other Cloudera Components

You can read and write Parquet data files from other Cloudera components, such as Hive.

Impala supports the scalar data types that you can encode in a Parquet data file, but not composite or nested types such as maps or arrays. Impala can query Parquet data files that include composite or nested types, as long as the query only refers to columns with scalar types.

If you copy Parquet data files between nodes, or even between different directories on the same node, make sure to preserve the block size by using the command hadoop distcp -pb. To verify that the block size was preserved, issue the command hdfs fsck -blocks *HDFS_PATH_OF_IMPALA_TABLE_DIR* and check that the average block size is at or near 256 MB (or whatever other size is defined by the PARQUET_FILE_SIZE query option).. (The hadoop distcp operation typically leaves some directories behind, with names matching _distcp_logs_*, that you can delete from the destination directory afterward.)

Issue the command `hadoop distcp` for details about `distcp` command syntax.

Impala can query Parquet files that use the PLAIN, PLAIN_DICTIONARY, BIT_PACKED, and RLE encodings. Starting with this release, Impala supports decoding RLE_DICTIONARY encoded pages. This encoding is identical to the already-supported PLAIN_DICTIONARY encoding but the PLAIN enum value is used for the dictionary pages and the RLE_DICTIONARY enum value is used for the data pages. When creating files outside of Impala for use by Impala, make sure to use one of the supported encodings.

In particular, for MapReduce jobs, parquet.writer.version must not be defined (especially as PARQUET_2_0) for writing the configurations of Parquet MR jobs.

Data using the version 2.0 of Parquet writer might not be consumable by Impala, due to use of the RLE_DICTIONARY encoding.

Use the default version of the Parquet writer and refrain from overriding the default writer version by setting the parquet.writer.version property or via WriterVersion.PARQUET_2_0 in the Parquet API.

How Parquet Data Files Are Organized

Although Parquet is a column-oriented file format, Parquet keeps all the data for a row within the same data file, to ensure that the columns for a row are always available on the same node for processing. Parquet sets a large HDFS block size and a matching maximum data file size to ensure that I/O and network transfer requests apply to large batches of data.

Within that data file, the data for a set of rows is rearranged so that all the values from the first column are organized in one contiguous block, then all the values from the second column, and so on. Putting the values from the same column next to each other lets Impala use effective compression techniques on the values in that column.

Note:

Impala INSERT statements write Parquet data files using an HDFS block size that matches the data file size, to ensure that each data file is represented by a single HDFS block, and the entire file can be processed on a single node without requiring any remote reads.

If you create Parquet data files outside of Impala, such as through a MapReduce or Pig job, ensure that the HDFS block size is greater than or equal to the file size, so that the “one file per block” relationship is maintained. Set the dfs.block.size or the dfs.blocksize property large enough that each file fits within a single HDFS block, even if that size is larger than the normal HDFS block size.

If the block size is reset to a lower value during a file copy, you will see lower performance for queries involving those files, and the PROFILE statement will reveal that some I/O is being done suboptimally, through remote reads.

When Impala retrieves or tests the data for a particular column, it opens all the data files, but only reads the portion of each file containing the values for that column. The column values are stored consecutively, minimizing the I/O required to process the values within a single column. If other columns are named in the SELECT list or WHERE clauses, the data for all columns in the same row is available within that same data file.

If an INSERT statement brings in less than one Parquet block's worth of data, the resulting data file is smaller than ideal. Thus, if you do split up an ETL job to use multiple INSERT statements, try to keep the volume of data for each INSERT statement to approximately 256 MB, or a multiple of 256 MB.

RLE and Dictionary Encoding for Parquet Data Files

Parquet uses some automatic compression techniques, such as run-length encoding (RLE) and dictionary encoding, based on analysis of the actual data values. Once the data values are encoded in a compact form, the encoded data can optionally be further compressed using a compression algorithm. Parquet data files created by Impala can use Snappy, GZip, or no compression; the Parquet spec also allows LZO compression, but currently Impala does not support LZO-compressed Parquet files.

RLE and dictionary encoding are compression techniques that Impala applies automatically to groups of Parquet data values, in addition to any Snappy or GZip compression applied to the entire data files. These automatic optimizations can save you time and planning that are normally needed for a traditional data warehouse. For example, dictionary encoding reduces the need to create numeric IDs as abbreviations for longer string values.

Run-length encoding condenses sequences of repeated data values. For example, if many consecutive rows all contain the same value for a country code, those repeating values can be represented by the value followed by a count of how many times it appears consecutively.

Dictionary encoding takes the different values present in a column, and represents each one in compact 2-byte form rather than the original value, which could be several bytes. (Additional compression is applied to the compacted values, for extra space savings.) This type of encoding applies when the number of different values for a column is less than 2**16 (16,384). It does not apply to columns of data type BOOLEAN, which are already very short. TIME

STAMP columns sometimes have a unique value for each row, in which case they can quickly exceed the 2**16 limit on distinct values. The 2**16 limit on different values within a column is reset for each data file, so if several different data files each contained 10,000 different city names, the city name column in each data file could still be condensed using dictionary encoding.

Compacting Data Files for Parquet Tables

If you reuse existing table structures or ETL processes for Parquet tables, you might encounter a “many small files” situation, which is suboptimal for query efficiency.

Here are techniques to help you produce large data files in Parquet INSERT operations, and to compact existing too-small data files:

- When inserting into a partitioned Parquet table, use statically partitioned INSERT statements where the partition key values are specified as constant values. Ideally, use a separate INSERT statement for each partition.
- You might set the NUM_NODES option to 1 briefly, during INSERT or CREATE TABLE AS SELECT statements. Normally, those statements produce one or more data files per data node. If the write operation involves small amounts of data, a Parquet table, and/or a partitioned table, the default behavior could produce many small files when intuitively you might expect only a single output file. SET NUM_NODES=1 turns off the “distributed” aspect of the write operation, making it more likely to produce only one or a few data files.
- Be prepared to reduce the number of partition key columns from what you are used to with traditional analytic database systems.
- Do not expect Impala-written Parquet files to fill up the entire Parquet block size. Impala estimates on the conservative side when figuring out how much data to write to each Parquet file. Typically, the uncompressed data in memory is substantially reduced on disk by the compression and encoding techniques in the Parquet file format. The final data file size varies depending on the compressibility of the data. Therefore, it is not an indication of a problem if 256 MB of text data is turned into 2 Parquet data files, each less than 256 MB.
- If you accidentally end up with a table with many small data files, consider using one or more of the preceding techniques and copying all the data into a new Parquet table, either through CREATE TABLE AS SELECT or INSERT ... SELECT statements.

  To avoid rewriting queries to change table names, you can adopt a convention of always running important queries against a view. Changing the view definition immediately switches any subsequent queries to use the new underlying tables:

Schema Evolution for Parquet Tables

Schema evolution refers to using the statement ALTER TABLE ... REPLACE COLUMNS to change the names, data type, or number of columns in a table. You can perform schema evolution for Parquet tables as follows:

- The Impala ALTER TABLE statement never changes any data files in the tables. From the Impala side, schema evolution involves interpreting the same data files in terms of a new table definition. Some types of schema changes make sense and are represented correctly. Other types of changes cannot be represented in a sensible way, and produce special result values or conversion errors during queries.
- The INSERT statement always creates data using the latest table definition. You might end up with data files with different numbers of columns or internal data representations if you do a sequence of INSERT and ALTER TABLE ... REPLACE COLUMNS statements.
- If you use ALTER TABLE ... REPLACE COLUMNS to define additional columns at the end, when the original data files are used in a query, these final columns are considered to be all NULL values.
- If you use ALTER TABLE ... REPLACE COLUMNS to define fewer columns than before, when the original data files are used in a query, the unused columns still present in the data file are ignored.

- Parquet represents the TINYINT, SMALLINT, and INT types the same internally, all stored in 32-bit integers.
  - That means it is easy to promote a TINYINT column to SMALLINT or INT, or a SMALLINT column to INT. The numbers are represented exactly the same in the data file, and the columns being promoted would not contain any out-of-range values.
  - If you change any of these column types to a smaller type, any values that are out-of-range for the new type are returned incorrectly, typically as negative numbers.
  - You cannot change a TINYINT, SMALLINT, or INT column to BIGINT, or the other way around. Although the ALTER TABLE succeeds, any attempt to query those columns results in conversion errors.
  - Any other type conversion for columns produces a conversion error during queries. For example, INT to STRI NG, FLOAT to DOUBLE, TIMESTAMP to STRING, DECIMAL(9,0) to DECIMAL(5,2), and so on.

You might find that you have Parquet files where the columns do not line up in the same order as in your Impala table. For example, you might have a Parquet file that was part of a table with columns C1,C2,C3,C4, and now you want to reuse the same Parquet file in a table with columns C4,C2. By default, Impala expects the columns in the data file to appear in the same order as the columns defined for the table, making it impractical to do some kinds of file reuse or schema evolution.

The query option PARQUET_FALLBACK_SCHEMA_RESOLUTION=name lets Impala resolve columns by name, and therefore handle out-of-order or extra columns in the data file.

Using ORC Data Files

Impala can read ORC data files.

Disabling ORC reads

By default, ORC reads are enabled in Impala 3.4.0. To disable the support of ORC data files:

1. In Cloudera Manager, navigate to Clusters Impala .
2. In the Configuration tab, set --enable_orc_scanner=false in the Impala Command Line Argument Advanced Configuration Snippet (Safety Valve)field.

   To enable ORC reads, set the property to true.
3. Restart the cluster.

Creating ORC Tables and Loading Data

To create a table in the ORC format, use the STORED AS ORC clause in the CREATE TABLE statement.

Because Impala can query ORC tables but cannot currently write to, after creating ORC tables, use the Hive shell to load the data.

After loading data into a table through Hive or other mechanism outside of Impala, issue a REFRESH *TABLE_NAME* statement the next time you connect to the Impala node, before querying the table, to make Impala recognize the new data.

For example, here is how you might create some ORC tables in Impala (by specifying the columns explicitly, or cloning the structure of another table), load data through Hive, and query them through Impala:

```
$ impala-shell -i localhost
[localhost:21000] default> CREATE TABLE orc_table (x INT) STORED AS ORC;
[localhost:21000] default> CREATE TABLE orc_clone LIKE some_other_table STO
RED AS ORC;
[localhost:21000] default> quit;

$ hive
hive> INSERT INTO TABLE orc_table SELECT x FROM some_other_table;
3 Rows loaded to orc_table
Time taken: 4.169 seconds
```

```
hive> quit;
$ impala-shell -i localhost
[localhost:21000] default> -- Make Impala recognize the data loaded through
 Hive;
[localhost:21000] default> REFRESH orc_table;
[localhost:21000] default> SELECT * FROM orc_table;

Fetched 3 row(s) in 0.11s
```

Data Type Considerations for ORC Tables

The ORC format defines a set of data types whose names differ from the names of the corresponding Impala data types. The following figure lists the ORC-defined types and the equivalent types in Impala.

Primitive types:

ORC type	Impala type
BINARY	STRING
BOOLEAN	BOOLEAN
DOUBLE	DOUBLE
FLOAT	FLOAT
TINYINT	TINYINT
SMALLINT	SMALLINT
INT	INT
BIGINT	BIGINT
TIMESTAMP	TIMESTAMP
DATE	Not supported

Complex types:

Impala supports the complex types ARRAY, STRUCT, and MAP in ORC files. Because Impala has better performance on Parquet than ORC, if you plan to use complex types, become familiar with the performance and storage aspects of Parquet first.

Enabling Compression for ORC Tables

ORC tables are in ZLIB (Deflate in Impala) compression by default. The supported compressions for ORC tables are NONE, ZLIB, and SNAPPY.

Set the compression when you load ORC tables in Hive.

Using Avro Data Files

Impala supports creating and querying Avro tables. You need to use Hive to insert data into Avro tables.

Creating Avro Tables

To create a new table using the Avro file format, use the STORED AS AVRO clause in the CREATE TABLE statement. If you create the table through Impala, you must include column definitions that match the fields specified in the Avro schema. With Hive, you can omit the columns and just specify the Avro schema.

The following examples demonstrate creating an Avro table in Impala, using either an inline column specification or one taken from a JSON file stored in HDFS:

```
[localhost:21000] > CREATE TABLE avro_only_sql_columns
                  > (col1 BOOLEAN, col2 INT)
                  > STORED AS AVRO;

[localhost:21000] > CREATE TABLE impala_avro_table
                  > (col1 BOOLEAN, col2 INT)
                  > STORED AS AVRO
                  > TBLPROPERTIES ('avro.schema.literal'='{
                  >    "name": "my_record",
                  >    "type": "record",
                  >    "fields": [
                  >       {"name":"col1", "type":"boolean"},
                  >       {"name":"col2", "type":"int"}]}');
[localhost:21000] > CREATE TABLE avro_examples_of_all_types
                    (col1 BOOLEAN, col2 INT)
                  >   STORED AS AVRO
                  >   TBLPROPERTIES ('avro.schema.url'='hdfs://localhost:802
0/avro_schemas/alltypes.json');
```

Each field of the record becomes a column of the table. Note that any other information, such as the record name, is ignored.

Note: For nullable Avro columns, make sure to put the "null" entry before the actual type name. In Impala, all columns are nullable; Impala currently does not have a NOT NULL clause. Any non-nullable property is only enforced on the Avro side.

If you create the table through Hive, switch back to `impala-shell` and issue an INVALIDATE METADATA *TABLE_NAME* statement. Then you can run queries for that table through `impala-shell`.

In rare instances, a mismatch could occur between the Avro schema and the column definitions in the Metastore database. Impala checks for such inconsistencies during a CREATE TABLE statement and each time it loads the metadata for a table (for example, after INVALIDATE METADATA). Impala uses the following rules to determine how to treat mismatching columns, a process known as *schema reconciliation*:

- If there is a mismatch in the number of columns, Impala uses the column definitions from the Avro schema.
- If there is a mismatch in column name or type, Impala uses the column definition from the Avro schema. Because a CHAR or VARCHAR column in Impala maps to an Avro STRING, this case is not considered a mismatch and the column is preserved as CHAR or VARCHAR in the reconciled schema.
- An Impala TIMESTAMP column definition maps to an Avro STRING and is presented as a STRING in the reconciled schema, because Avro has no binary TIMESTAMP representation.

Specifying the Avro Schema through JSON:

While you can embed a schema directly in your CREATE TABLE statement, as shown above, column width restrictions in the Hive Metastore limit the length of schema you can specify. If you encounter problems with long schema literals, try storing your schema as a JSON file in HDFS instead. Specify your schema in HDFS using table properties in the following format using the avro.schema.url in TBLPROPERTIES clause.

```
TBLPROPERTIES ('avro.schema.url'='hdfs//your-name-node:port/path/to/schema.j
son');
```

Data Type Considerations for Avro Tables

The Avro format defines a set of data types whose names differ from the names of the corresponding Impala data types. If you are preparing Avro files using other Hadoop components such as Pig or MapReduce, you might need to

work with the type names defined by Avro. The following figure lists the Avro-defined types and the equivalent types in Impala.

Primitive types:

Avro type	Impala type
STRING	STRING
STRING	CHAR
STRING	VARCHAR
INT	INT
BOOLEAN	BOOLEAN
LONG	BIGINT
FLOAT	FLOAT
DOUBLE	DOUBLE

The Avro specification allows string values up to 2**64 bytes in length:

- Impala queries for Avro tables use 32-bit integers to hold string lengths.
- Impala truncates CHAR and VARCHAR values in Avro tables to (2**31)-1 bytes.
- If a query encounters a STRING value longer than (2**31)-1 bytes in an Avro table, the query fails.

Logical types:

Avro type	Impala type
BYTES annotated	DECIMAL
INT32 annotated	DATE

Avro types not supported by Impala

- RECORD
- MAP
- ARRAY
- UNION
- ENUM
- FIXED
- NULL

Impala types not supported by Avro:

- TIMESTAMP

Impala issues warning messages if there are any mismatches between the types specified in the SQL column definitions and the underlying types; for example, any TINYINT or SMALLINT columns are treated as INT in the underlying Avro files, and therefore are displayed as INT in any DESCRIBE or SHOW CREATE TABLE output.

Note:

Currently, Avro tables cannot contain TIMESTAMP columns. If you need to store date and time values in Avro tables, as a workaround you can use a STRING representation of the values, convert the values to BIGINT with the UNIX_TIMESTAMP() function, or create separate numeric columns for individual date and time fields using the EXTRACT() function.

Using a Hive-Created Avro Table in Impala

If you have an Avro table created through Hive, you can use it in Impala as long as it contains only Impala-compatible data types. It cannot contain Avro types not supported by Impala, such as ENUM and FIXED Because

Impala and Hive share the same metastore database, Impala can directly access the table definitions and data for tables that were created in Hive.

If you create an Avro table in Hive, issue an INVALIDATE METADATA in Impala. This is a one-time operation to make Impala aware of the new table. You can issue the statement while connected to any Impala node, and the catalog service broadcasts the change to all other Impala nodes.

If you load new data into an Avro table through Hive, either through a Hive LOAD DATA or INSERT statement, or by manually copying or moving files into the data directory for the table, issue a REFRESH *TABLE_NAME* statement the next time you connect to Impala through `impala-shell`.

If you issue the LOAD DATA statement through Impala, you do not need a REFRESH afterward.

Impala only supports fields of type BOOLEAN, INT, LONG, FLOAT, DOUBLE, and STRING, or unions of these types with null, for example, ["string", "null"]. Unions with null essentially create a nullable type.

Loading Data into Avro Tables

Currently, Impala cannot write Avro data files. Therefore, an Avro table cannot be used as the destination of an Impala INSERT statement or CREATE TABLE AS SELECT.

To copy data from another table, issue any INSERT statements through Hive.

After loading data into a table through Hive or other mechanism outside of Impala, issue a REFRESH *TABLE_NAME* statement the next time you connect to the Impala node, before querying the table, to make Impala recognize the new data.

If you already have data files in Avro format, you can also issue LOAD DATA in either Impala or Hive. Impala can move existing Avro data files into an Avro table, it just cannot create new Avro data files.

Enabling Compression for Avro Tables

To enable compression for Avro tables, specify settings in the Hive shell to enable compression and to specify a codec, then issue a CREATE TABLE statement as in the preceding examples. Impala supports the snappy and deflate codecs for Avro tables.

For example:

```
hive> set hive.exec.compress.output=true;
hive> set avro.output.codec=snappy;
```

Handling Avro Schema Evolution

Impala can handle with Avro data files that employ *schema evolution*, where different data files within the same table use slightly different type definitions. (You would perform the schema evolution operation by issuing an ALTER TABLE statement in the Hive shell.) The old and new types for any changed columns must be compatible, for example a column might start as an INT and later change to a BIGINT or FLOAT.

As with any other tables where the definitions are changed or data is added outside of the current `impalad` node, ensure that Impala loads the latest metadata for the table if the Avro schema is modified through Hive. Issue a REFRESH *TABLE_NAME* or INVALIDATE METADATA *TABLE_NAME* statement. REFRESH reloads the metadata immediately, INVALIDATE METADATA reloads the metadata the next time the table is accessed.

When Avro data files or columns are not consulted during a query, Impala does not check for consistency. Thus, if you issue SELECT c1, c2 FROM t1, Impala does not return any error if the column c3 changed in an incompatible way. If a query retrieves data from some partitions but not others, Impala does not check the data files for the unused partitions.

In the Hive DDL statements, you can specify an avro.schema.literal table property (if the schema definition is short) or an avro.schema.url property (if the schema definition is long, or to allow convenient editing for the definition).

Query Performance for Avro Tables

In general, expect query performance with Avro tables to be faster than with tables using text data, but slower than with Parquet tables.

Using RCFile Data Files

Impala supports creating and querying RCFile tables.

Creating RCFile Tables and Loading Data

To create a table in the RCFile format, use the STORED AS RCFILE clause in the CREATE TABLE statement.

Because Impala can query RCFile tables but cannot currently write to, after creating RCFile tables, use the Hive shell to load the data.

After loading data into a table through Hive or other mechanism outside of Impala, issue a REFRESH *TABLE_NAME* statement the next time you connect to the Impala node, before querying the table, to make Impala recognize the new data.

Query Performance for RCFile Tables

In general, expect query performance with RCFile tables to be faster than with tables using text data, but slower than with Parquet tables.

Enabling Compression for RCFile Tables

You may want to enable compression on existing tables. Enabling compression provides performance gains in most cases and is supported for RCFile tables.

The compression type is specified in the SET mapred.output.compression.codec command.

For example, to enable Snappy compression, you would specify the following additional settings when loading data through the Hive shell.

```
SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCode
c;
```

Using SequenceFile Data Files

Impala supports creating and querying SequenceFile tables.

Creating SequenceFile Tables and Loading Data

To create a table in the SequenceFile format, use the STORED AS SEQUENCEFILE clause in the CREATE T ABLE statement.

Because Impala can query SequenceFile tables but cannot currently write to, after creating SequenceFile tables, use the Hive shell to load the data.

After loading data into a table through Hive or other mechanism outside of Impala, issue a REFRESH *TABLE_NAME* statement the next time you connect to the Impala node, before querying the table, to make Impala recognize the new data.

Enabling Compression for SequenceFile Tables

You may want to enable compression on existing tables. Enabling compression provides performance gains in most cases and is supported for SequenceFile tables.

The compression type is specified in the SET mapred.output.compression.codec command.

For example, to enable Snappy compression, you would specify the following additional settings when loading data through the Hive shell.

```
SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCode
c;
```

Query Performance for SequenceFile Tables

In general, expect query performance with SequenceFile tables to be faster than with tables using text data, but slower than with Parquet tables.

Storage Systems Supports

This section describes the storage systems that Impala supports.

Impala with HDFS

You can use Impala to query data files that reside on HDFS.

Although Impala typically works well with many large files in an HDFS storage system, there are times when you might perform some file cleanup to reclaim space, or advise developers on techniques to minimize space consumption and file duplication.

- Use compact binary file formats where practical. Numeric and time-based data in particular can be stored in more compact form in binary data files. Depending on the file format, various compression and encoding features can reduce file size even further. You can specify the STORED AS clause as part of the CREATE TABLE statement, or ALTER TABLE with the SET FILEFORMAT clause for an existing table or partition within a partitioned table.
- You manage underlying data files differently depending on whether the corresponding Impala table is defined as an internal or external table:
  - Use the DESCRIBE FORMATTED statement to check if a particular table is internal (managed by Impala) or external, and to see the physical location of the data files in HDFS.
  - For Impala-managed (“internal”) tables, use DROP TABLE statements to remove data files.
  - For tables not managed by Impala (“external”) tables, use appropriate HDFS-related commands such as hadoop fs, hdfs dfs, or distcp, to create, move, copy, or delete files within HDFS directories that are accessible by the impala user. Issue a REFRESH *TABLE_NAME* statement after adding or removing any files from the data directory of an external table.
  - Use external tables to reference HDFS data files in their original location. With this technique, you avoid copying the files, and you can map more than one Impala table to the same set of data files. When you drop the Impala table, the data files are left undisturbed.
  - Use the LOAD DATA statement to move HDFS files into the data directory for an Impala table from inside Impala, without the need to specify the HDFS path of the destination directory. This technique works for both internal and external tables.
- Make sure that the HDFS trashcan is configured correctly. When you remove files from HDFS, the space might not be reclaimed for use by other files until sometime later, when the trashcan is emptied.
- Drop all tables in a database before dropping the database itself.

- If an INSERT statement encounters an error, and you see a directory named .impala_insert_staging or _impala_insert_staging left behind in the data directory for the table, it might contain temporary data files taking up space in HDFS. You might be able to salvage these data files.

  For example, delete those files through commands such as hadoop fs or hdfs dfs to reclaim space before re-trying the INSERT. Issue DESCRIBE FORMATTED *TABLE_NAME* to see the HDFS path where you can check for temporary files.

Configuring Scratch Space for Spilling to Disk

Impala uses intermediate files during large sorts, joins, aggregations, or analytic function operations The files are removed when the operation finishes. By default, intermediate files are stored in the directory /tmp/impala-scratch.

You can specify locations of the intermediate files in one of the following ways:

- By starting the `impalad` daemon with the ##scratch_dirs="*PATH_TO_DIRECTORY*" configuration option.
- By specifying a different location in the Cloudera Manager in the Impala Daemon Scratch Directories field.

With either option above:

- You can specify a single directory or a comma-separated list of directories.
- You can specify an optional a capacity quota per scratch directory using the colon (:) as the delimiter.

  The capacity quota of -1 or 0 is the same as no quota for the directory.
- The scratch directories must be on the local filesystem, not in HDFS.
- You might specify different directory paths for different hosts, depending on the capacity and speed of the available storage devices.

If there is less than 1 GB free on the filesystem where that directory resides, Impala still runs, but writes a warning message to its log.

Impala successfully starts (with a warning written to the log) if it cannot create or read and write files in one of the scratch directories.

The following are examples for specifying scratch directories.

Config option	Description
--scratch_dirs=/dir1,/dir2	Use /dir1 and /dir2 as scratch directories with no capacity quota.
--scratch_dirs=/dir1,/dir2:25G	Use /dir1 and /dir2 as scratch directories with no capacity quota on /dir1 and the 25GB quota on /dir2.
--scratch_dirs=/dir1:5MB,/dir2	Use /dir1 and /dir2 as scratch directories with the capacity quota of 5MB on /dir1 and no quota on /dir2.
--scratch_dirs=/dir1:-1,/dir2:0	Use /dir1 and /dir2 as scratch directories with no capacity quota.

Allocation from a scratch directory will fail if the specified limit for the directory is exceeded.

If Impala encounters an error reading or writing files in a scratch directory during a query, Impala logs the error, and the query fails.

Priority Based Scratch Directory Selection

The location of the intermediate files are configured by starting the impalad daemon with the flag ##scratch_dirs="path_to_directory". Currently this startup flag uses the configured scratch directories in a round robin fashion. Automatic selection of scratch directories in a round robin fashion may not always be ideal in every situation since these directories could come from different classes of storage system volumes having different performance characteristics (SSD vs HDD, local storage vs network attached storage, etc.). To optimize your workload, you have an option to configure the priority of the scratch directories based on your storage system configuration.

The scratch directories will be selected for spilling based on how you configure the priorities of the directories and if you provide the same priority for multiple directories then the directories will be selected in a round robin fashion.

The valid formats for specifying the priority directories are as shown here:

```
<dir-path>:<limit>:<priority>
<dir-path>::<priority>
```

Example:

```
/dir1:200GB:0
/dir1::0
```

The following formats use the default priority:

```
/dir1
/dir1:200GB
/dir1:200GB:
```

In the example below, dir1 will be used as a spill target until it is full and then dir2, dir3, and dir4 will be used in a round robin fashion.

```
##scratch_dirs="/dir1:200GB:0, /dir2:1024GB:1, /dir3:1024GB:1, /dir4:1024GB:
1"
```

Increasing Scratch Capacity

You can compress the data spilled to disk to increase the effective scratch capacity. You typically more than double capacity using compression and reduce spilling to disk. Use the --disk_spill_compression_codec and –-disk_spill_punch_holes startup options. The --disk_spill_compression_codec takes any value supported by the COMPRESSION_CODEC query option. The value is not case-sensitive. A value of ZSTD or LZ4 is recommended (default is NONE).

For example:

```
--disk_spill_compression_codec=LZ4
--disk_spill_punch_holes=true
```

If you set --disk_spill_compression_codec to a value other than NONE, you must set --disk_spill_punch_holes to true.

The hole punching feature supported by many file systems is used to reclaim space in scratch files during execution of a query that spills to disk. This results in lower scratch space requirements in many cases, especially when combined with disk spill compression. When this option is not enabled, scratch space is still recycled by a query, but less effectively in many cases.

You can specify a compression level for ZSTD only. For example:

```
--disk_spill_compression_codec=ZSTD:10
--disk_spill_punch_holes=true
```

Compression levels from 1 up to 22 (default 3) are supported for ZSTD. The lower the compression level, the faster the speed at the cost of compression ratio.

Configure Impala Daemon to spill to HDFS

Impala occasionally needs to use persistent storage for writing intermediate files during large sorts, joins, aggregations, or analytic function operations. If your workload results in large volumes of intermediate data being written, it is recommended to configure the heavy spilling queries to use a remote storage location rather than the local one. The advantage of using remote storage for scratch space is that it is elastic and can handle any amount of spilling.

Before you begin

- Identify the HDFS scratch directory where you want your new Impala to write the temporary data.
- Identify the IP address, host name, or service identifier of HDFS.
- Identify the port number of the HDFS NameNode (if not-default).
- Configure Impala to write temporary data to disk during query processing.

Configuring the Start-up Option in Impala daemon

You can use the Impalad start option scratch_dirs to specify the locations of the intermediate files.

Use the following format for this start up option:

```
--scratch_dirs="hdfs://AUTHORITY/PATH(:MAX_BYTES), LOCAL_BUFFER_DIR
 (,LOCAL_DIR…)"
```

- Where hdfs://*AUTHORITY*/*PATH* is the remote directory.
- *AUTHORITY* may include ip_address or hostname and port, or service_id.
- *MAX_BYTES* is optional.

Using the above format:

- You can specify only one remote directory. When you configure a remote directory, you must specify a local buffer directory as the buffer. However you can use multiple local directories with the remote directory. If you specify multiple local directories, the first local directory would be used as the local buffer directory.
- If you configure both remote and local directories, the remote directory is only used when the local directories are fully utilized.
- The size of a remote intermediate file could affect the query performance, and the value can be set by --remote _tmp_file_size=*SIZE* in the start-up option. The default size of a remote intermediate file is 16MB while the maximum is 512MB.

Examples

- A HDFS scratch dir with one local buffer dir, file size 64MB. The space of HDFS scratch dir is limited to 300G.

  ```
  --scratch_dirs=hdfs://10.0.0.49:20500/tmp:300G,/local_buffer_dir --remot
  e_tmp_file_size=64M
  ```

- A HDFS scratch dir with one local buffer dir limited to 512MB, and one local dir limited to 10GB. The space of HDFS scratch dir is limited to 300G. The HDFS NameNode uses its default port (8020).

  ```
  --scratch_dirs=hdfs://hdfsnn/tmp:300G,/local_buffer_dir:512M,/local_dir:
  10G
  ```

- A HDFS scratch dir with one local buffer dir, and multiple prioritized local dirs. The space of HDFS scratch dir is unlimited. The HDFS service identifier is hdfs1.

  ```
  --scratch_dirs=hdfs://hdfs1/tmp,/local_buffer_dir,/local_dir_1:5G:1,/loc
  al_dir_2:5G:2
  ```

Even though max_bytes is optional, it is highly recommended to configure for spilling to HDFS because the HDFS cluster space is limited.

Impala with Kudu

You can use Impala to query tables stored by Apache Kudu. This capability allows convenient access to a storage system that is tuned for different kinds of workloads than the default with Impala.

By default, Impala tables are stored on HDFS. HDFS files are ideal for bulk loads (append operations) and queries using full-table scans, but do not support in-place updates or deletes. Kudu is an alternative storage engine used by Impala which can do both in-place updates (for mixed read/write workloads) and fast scans (for data-warehouse/analytic operations). Using Kudu tables with Impala can simplify the ETL pipeline by avoiding extra steps to segregate and reorganize newly arrived data.

The underlying storage is managed and organized by Kudu, not represented as HDFS data files.

The combination of Kudu and Impala works best for tables where scan performance is important, but data arrives continuously, in small batches, or needs to be updated without being completely replaced. HDFS-backed tables can require substantial overhead to replace or reorganize data files as new data arrives. Impala can perform efficient lookups and scans within Kudu tables, and Impala can also perform update or delete operations efficiently. You can also use the Kudu Java, C++, and Python APIs to do ingestion or transformation operations outside of Impala, and Impala can query the current data at any time.

With HDFS-backed tables, you are typically concerned with the number of DataNodes in the cluster, how many and how large HDFS data files are read during a query, and therefore the amount of work performed by each DataNode and the network communication to combine intermediate results and produce the final result set.

Certain Impala SQL statements and clauses, such as DELETE, UPDATE, UPSERT, and PRIMARY KEY work only with Kudu tables.

Other statements and clauses, such as LOAD DATA, TRUNCATE TABLE, and INSERT OVERWRITE, are not applicable to Kudu tables.

Query performance for Kudu tables

For queries involving Kudu tables, Impala can delegate much of the work of filtering the result set to Kudu, avoiding some of the I/O involved in full table scans of tables containing HDFS data files. This type of optimization is especially effective for partitioned Kudu tables, where the Impala query WHERE clause refers to one or more primary key columns that are also used as partition key columns. For example, if a partitioned Kudu table uses a HASH clause for col1 and a RANGE clause for col2, a query using a clause such as WHERE col1 IN (1,2,3) AND col2 > 100 can determine exactly which tablet servers contain relevant data, and therefore parallelize the query efficiently.

Impala can push down additional information to optimize join queries involving Kudu tables. If the join clause contains predicates of the form *COLUMN = EXPRESSION*, after Impala constructs a hash table of possible matching values for the join columns from the bigger table (either an HDFS table or a Kudu table), Impala can “push down” the minimum and maximum matching column values to Kudu, so that Kudu can more efficiently locate matching rows in the second (smaller) table. These min/max filters are affected by the RUNTIME_FILTER_MODE, RUNTIME_FILTER_WAIT_TIME_MS, and DISABLE_ROW_RUNTIME_FILTERING query options; the min/max filters are not affected by the RUNTIME_BLOOM_FILTER_SIZE, RUNTIME_FILTER_MIN_SIZE, RUNTIME_FILTER_MAX_SIZE, and MAX_NUM_RUNTIME_FILTERS query options.

The TABLESAMPLE clause of the SELECT statement does not apply to a table reference derived from a view, a subquery, or anything other than a real base table. This clause only works for tables backed by HDFS or HDFS-like data files, therefore it does not apply to Kudu tables.

Configuring for Kudu Tables

The Kudu configuration property must be set for Impala to connect to the appropriate Kudu server.

Procedure

1. In Cloudera Manager, navigate to Clusters Impala Service .

2. In the Configuration tab, specify the Kudu service you want to use in the Kudu Service field.

   Typically, the required value for this setting is *KUDU_HOST*:7051. In a high-availability Kudu deployment, specify the names of multiple Kudu hosts separated by commas.
3. If the Kudu Service field is not set, you can still associate the appropriate value for each table by specifying a TBLPROPERTIES('kudu.master_addresses') clause in the CREATE TABLE statement or changing the TBLP ROPERTIES('kudu.master_addresses') value with an ALTER TABLE statement.

Impala DDL for Kudu

You can use the Impala CREATE TABLE and ALTER TABLE statements to create and fine-tune the characteristics of Kudu tables. Impala supports specific features and properties that only apply to Kudu tables.

Non-unique Primary Keys for Kudu Tables

Kudu now allows a user to create a non-unique primary key for a table when creating a table. The data engine handles this by appending a system generated auto-incrementing column to the non-unique primary key columns. This is done to guarantee the uniqueness of the primary key. This auto-incrementing column is named as 'auto_incrementing_id' with bigint type and this column is only system generated and cannot be explicitly created by the user. This auto_incrementing_id column is unique across a partition/tablet i.e. every partition/tablet would have this column starting from one and incrementing monotonically. The assignment to this column during insertion is automatic.

Create a Kudu Table with a non-unique PRIMARY KEY

The following example shows creating a table with a non-unique PRIMARY KEY.

```
CREATE TABLE kudu_tbl1
(
 id INT NON UNIQUE PRIMARY KEY,
 name STRING
)
 PARTITION BY HASH (id) PARTITIONS 3 STORED as KUDU;
```

The effective PRIMARY KEY in the above case will be {id, auto_increment_id}

Note: "auto_incrementing_id" column cannot be added, removed or renamed with ALTER TABLE statements.

Verify the PRIMARY KEY is non-unique

You can now check the PRIMARY KEY created is non-unique by running the following DESCRIBE command. A new property "key_unique" shows if the primary key is unique. System generated column "auto_incrementing_id" is shown in the output for the table as a non-unique primary key.

```
describe kudu_tbl1
+----------------------+--------+---------+-------------+------------+---
-------+---------------+---------------+---------------------+------------+
| name                 | type   | comment | primary_key | key_unique | nulla
ble | default_value | encoding      | compression         | block_size |
+----------------------+--------+---------+-------------+------------+-----
-----+---------------+---------------+---------------------+------------+
| id                   | int    |         | true        | false      | false
    |               | AUTO_ENCODING | DEFAULT_COMPRESSION | 0          |
| auto_incrementing_id | bigint |         | true        | false      | fa
lse    |               | AUTO_ENCODING | DEFAULT_COMPRESSION | 0          |
| name                 | string |         | false       |            | true
    |               | AUTO_ENCODING | DEFAULT_COMPRESSION | 0          |
+----------------------+--------+---------+-------------+------------+-----
-----+---------------+---------------+---------------------+------------+
Fetched 3 row(s) in 4.72s
```

Query Auto Incrementing Column

When you query a table using the SELECT statement, it will not display the system generated auto incrementing column unless the column is explicitly specified in the select list.

Create a Kudu table without a PRIMARY KEY attribute

You can create a Kudu table without specifying a PRIMARY KEY or a PARTITION KEY since they are optional, however you cannot create a Kudu table without specifying both PRIMARY KEY and PARTITION KEY. If you do not specify the primary key attribute, the partition key columns can be promoted as a non-unique primary key. This is possible only if those columns are the beginning columns of the table.

In the following example, 'a' and 'b' will be promoted as a non-unique primary key, 'auto_incrementing_id' column will be added by Kudu engine. 'a', 'b' and 'auto_incrementing_id' form the effective unique composite primary key.

```
CREATE TABLE auto_table
(
a BIGINT,
b STRING,
)
PARTITION BY HASH(a, b) PARTITIONS 2 STORED AS KUDU;
```

The effective primary key in this case would be {a, b, auto_incrementing_id}

Limitations

- UPSERT operation is not supported for Kudu tables with non-unique primary key. If you run an UPSERT statement for a Kudu table with a non-unique primary key it will fail with an error.
- Since the auto generated key for each row will be assigned after the row's data is generated and after the row lands in the tablet, you cannot use this column in the partition key.

PRIMARY KEY Attribute

To create a Kudu table, use the STORED AS KUDU clause in the CREATE TABLE statement.

The column list in a CREATE TABLE statement can include the following attributes, which only apply to Kudu tables:

```
[NON UNIQUE] PRIMARY KEY
| [NOT] NULL
| ENCODING CODEC
| COMPRESSION ALGORITHM
| DEFAULT CONSTANT_EXPRESSION
| BLOCK_SIZE NUMBER
```

The primary key for a Kudu table is a column, or set of columns, that uniquely identifies every row. The primary key value also is used as the natural sort order for the values from the table.

The notion of primary key only applies to Kudu tables. Every Kudu table requires a primary key.

Because all of the primary key columns must have non-null values, specifying a column in the PRIMARY KEY or NON-UNIQUE PRIMARY KEY clause implicitly adds the NOT NULL attribute to that column.

The primary key columns must be the first ones specified in the CREATE TABLE statement.

When the primary key is a single column, you can specify the PRIMARY KEY attribute either inline in a single column definition, or as a separate clause at the end of the column list.

For example:

```
CREATE TABLE pk_inline
(
col1 BIGINT PRIMARY KEY,
col2 STRING,
col3 BOOLEAN
) PARTITION BY HASH(col1) PARTITIONS 2 STORED AS KUDU;

CREATE TABLE pk_at_end
(
col1 BIGINT,
col2 STRING,
col3 BOOLEAN,
PRIMARY KEY (col1)
) PARTITION BY HASH(col1) PARTITIONS 2 STORED AS KUDU;

CREATE TABLE pk_inline
(
col1 BIGINT [NON UNIQUE] PRIMARY KEY,
col2 STRING,
col3 BOOLEAN
) PARTITION BY HASH(col1) PARTITIONS 2 STORED AS KUDU;

CREATE TABLE pk_at_end
(
col1 BIGINT,
col2 STRING,
col3 BOOLEAN,
[NON UNIQUE] PRIMARY KEY (col1)
) PARTITION BY HASH(col1) PARTITIONS 2 STORED AS KUDU;
```

The contents of the primary key columns cannot be changed by an UPDATE or UPSERT statement.

Including too many columns in the primary key (more than 5 or 6) can reduce the performance of write operations. Therefore, pick the most selective and most frequently tested non-null columns for the primary key specification. If a column must always have a value, but that value might change later, leave it out of the primary key and use a NOT NULL clause for that column instead.

NULL | NOT NULL Attribute

For Kudu tables, you can specify which columns can contain nulls or not. This constraint offers an extra level of consistency enforcement for Kudu tables. If an application requires a field to always be specified, include a NOT NULL clause in the corresponding column definition, and Kudu prevents rows from being inserted with a NULL in that column.

For example, a table containing geographic information might require the latitude and longitude coordinates to always be specified. Other attributes might be allowed to be NULL. For example, a location might not have a designated place name, its altitude might be unimportant, and its population might be initially unknown, to be filled in later.

For non-Kudu tables, Impala allows any column to contain NULL values, because it is not practical to enforce a “not null” constraint on HDFS data files that could be prepared using external tools and ETL processes.

```
CREATE TABLE required_columns
(
  id BIGINT PRIMARY KEY,
  latitude DOUBLE NOT NULL,
  longitude DOUBLE NOT NULL,
  place_name STRING,
  altitude DOUBLE,
  population BIGINT
) PARTITION BY HASH(id) PARTITIONS 2 STORED AS KUDU;
```

During performance optimization, Kudu can use the knowledge that nulls are not allowed to skip certain checks on each input row, speeding up queries and join operations. Therefore, specify NOT NULL constraints when appropriate.

The NULL clause is the default condition for all columns that are not part of the primary key. You can omit it, or specify it to clarify that you have made a conscious design decision to allow nulls in a column.

Because primary key columns cannot contain any NULL values, the NOT NULL clause is not required for the primary key columns, but you might still specify it to make your code self-describing.

DEFAULT Attribute

You can specify a default value for columns in Kudu tables. The default value can be any constant expression, for example, a combination of literal values, arithmetic and string operations. It cannot contain references to columns or non-deterministic function calls.

The following example shows different kinds of expressions for the DEFAULT clause. The requirement to use a constant value means that you can fill in a placeholder value such as NULL, empty string, 0, -1, 'N/A' and so on, but you cannot reference functions or column names. Therefore, you cannot use DEFAULT to do things such as automatically making an uppercase copy of a string value, storing Boolean values based on tests of other columns, or add or subtract one from another column representing a sequence number.

```
CREATE TABLE default_vals
(
  id BIGINT PRIMARY KEY,
  name STRING NOT NULL DEFAULT 'unknown',
  address STRING DEFAULT upper('no fixed address'),
  age INT DEFAULT -1,
  earthling BOOLEAN DEFAULT TRUE,
  planet_of_origin STRING DEFAULT 'Earth',
  optional_col STRING DEFAULT NULL
) PARTITION BY HASH(id) PARTITIONS 2 STORED AS KUDU;
```

Note:

When designing an entirely new schema, prefer to use NULL as the placeholder for any unknown or missing values, because that is the universal convention among database systems. Null values can be stored efficiently, and easily checked with the IS NULL or IS NOT NULL operators. The DEFAULT attribute is appropriate when ingesting data that already has an established convention for representing unknown or missing values, or where the vast majority of rows have some common non-null value.

ENCODING Attribute

Each column in a Kudu table can optionally use an encoding, a low-overhead form of compression that reduces the size on disk, then requires additional CPU cycles to reconstruct the original values during queries. Typically, highly compressible data benefits from the reduced I/O to read the data back from disk.

The encoding keywords that Impala recognizes are:

- AUTO_ENCODING: Use the default encoding based on the column type, which are bitshuffle for the numeric type columns and dictionary for the string type columns.
- PLAIN_ENCODING: Leave the value in its original binary format.
- RLE: Compress repeated values (when sorted in primary key order) by including a count.
- DICT_ENCODING: When the number of different string values is low, replace the original string with a numeric ID.
- BIT_SHUFFLE: Rearrange the bits of the values to efficiently compress sequences of values that are identical or vary only slightly based on primary key order. The resulting encoded data is also compressed with LZ4.
- PREFIX_ENCODING: Compress common prefixes in string values; mainly for use internally within Kudu.

COMPRESSION Attribute

You can specify a compression algorithm to use for each column in a Kudu table. This attribute imposes more CPU overhead when retrieving the values than the ENCODING attribute does. Therefore, use it primarily for columns with long strings that do not benefit much from the less-expensive ENCODING attribute.

The choices for COMPRESSION are LZ4, SNAPPY, and ZLIB.

Note:

Columns that use the BITSHUFFLE encoding are already compressed using LZ4, and so typically do not need any additional COMPRESSION attribute.

BLOCK_SIZE Attribute

Although Kudu does not use HDFS files internally, and thus is not affected by the HDFS block size, it does have an underlying unit of I/O called the *block size*. The BLOCK_SIZE attribute lets you set the block size for any column.

The block size attribute is a relatively advanced feature. This is an unsupported feature and is considered experimental.

Kudu Replication Factor

By default, Kudu tables created through Impala use a tablet replication factor of 3. To change the replication factor for a Kudu table, specify the replication factor using the TBLPROPERTIES in the CREATE TABLE statement as below where *N* is the replication factor you want to use:

```
TBLPROPERTIES ('kudu.num_tablet_replicas' = 'n')
```

The number of replicas for a Kudu table must be odd.

Altering the kudu.num_tablet_replicas property after table creation currently has no effect.

How Impala Handles Kudu Metadata

Much of the metadata for Kudu tables is handled by the underlying storage layer. Kudu tables have less reliance on the Metastore database, and require less metadata caching on the Impala side. For example, information about partitions in Kudu tables is managed by Kudu, and Impala does not cache any block locality metadata for Kudu tables.

The REFRESH and INVALIDATE METADATA statements are needed less frequently for Kudu tables than for HDFS-backed tables. Neither statement is needed when data is added to, removed, or updated in a Kudu table, even if the changes are made directly to Kudu through a client program using the Kudu API.

Run REFRESH *TABLE_NAME* or INVALIDATE METADATA *TABLE_NAME* for a Kudu table only after making a change to the Kudu table schema, such as adding or dropping a column.

Because Kudu manages the metadata for its own tables separately from the Metastore database, there is a table name stored in the Metastore database for Impala to use, and a table name on the Kudu side, and these names can be modified independently through ALTER TABLE statements.

To avoid potential name conflicts, the prefix impala:: and the Impala database name are encoded into the underlying Kudu table name.

Partitioning for Kudu Tables

Kudu tables use special mechanisms to distribute data among the underlying tablet servers. Although referred as partitioned tables, they are distinguished from traditional Impala partitioned tables with the different syntax in CREATE TABLE statement.

Kudu tables use PARTITION BY, HASH, RANGE, and range specification clauses rather than the PARTITIONED BY clause for HDFS-backed tables, which specifies only a column name and creates a new partition for each different value.

To see the current partitioning scheme for a Kudu table, you can use the SHOW CREATE TABLE statement or the SHOW PARTITIONS statement. The CREATE TABLE syntax displayed by this statement includes all the hash, range, or both clauses that reflect the original table structure plus any subsequent ALTER TABLE statements that changed the table structure.

To see the underlying buckets and partitions for a Kudu table, use the SHOW TABLE STATS or SHOW PARTITIONS statement.

Hash Partitioning

Hash partitioning is the simplest type of partitioning for Kudu tables. For hash-partitioned Kudu tables, inserted rows are divided up between a fixed number of “buckets” by applying a hash function to the values of the columns specified in the HASH clause. Hashing ensures that rows with similar values are evenly distributed, instead of clumping together all in the same bucket. Spreading new rows across the buckets this way lets insertion operations work in parallel across multiple tablet servers. Separating the hashed values can impose additional overhead on queries, where queries with range-based predicates might have to read multiple tablets to retrieve all the relevant values.

```
-- 1M rows with 50 hash partitions = approximately 20,000 rows per partitio
n.
-- The values in each partition are not sequential, but rather based on a
 hash function.
-- Rows 1, 99999, and 123456 might be in the same partition.
CREATE TABLE million_rows (id string primary key, s string)
  PARTITION BY HASH(id) PARTITIONS 50
  STORED AS KUDU;

-- Because the ID values are unique, we expect the rows to be roughly
-- evenly distributed between the buckets in the destination table.
INSERT INTO million_rows SELECT * FROM billion_rows ORDER BY id LIMIT 1e6;
```

The largest number of buckets that you can create with a PARTITIONS clause varies depending on the number of tablet servers in the cluster, while the smallest is 2. For large tables, prefer to use roughly 10 partitions per server in the cluster.

Range Partitioning

Range partitioning lets you specify partitioning precisely, based on single values or ranges of values within one or more columns. You add one or more RANGE clauses to the CREATE TABLE statement, following the PARTITION BY clause. The RANGE clause includes a combination of constant expressions, VALUE or VALUES keywords, and comparison operators.

```
CREATE TABLE t1 (id STRING PRIMARY KEY, s STRING)
  PARTITION BY RANGE (PARTITION 'a' <= VALUES < '{', PARTITION 'A' <= VAL
UES < '[', PARTITION '00000' <= VALUES < '00001')
  STORED AS KUDU;
```

Note:

When defining ranges, be careful to avoid “fencepost errors” where values at the extreme ends might be included or omitted by accident. For example, in the tables defined in the preceding code listings, the range "a" <= VALUES < "{" ensures that any values starting with z, such as za or zzz or zzz-ZZZ, are all included, by using a less-than operator for the smallest value after all the values starting with z.

For range-partitioned Kudu tables, an appropriate range must exist before a data value can be created in the table. Any INSERT, UPDATE, or UPSERT statements fail if they try to create column values that fall outside the specified ranges. The error checking for ranges is performed on the Kudu side. Impala passes the specified range information to

Kudu, and passes back any error or warning if the ranges are not valid. (A nonsensical range specification causes an error for a DDL statement, but only a warning for a DML statement.)

Partition ranges can be non-contiguous:

```
PARTITION BY RANGE (year) (PARTITION 1885 <= VALUES <= 1889, PARTITION 1893
 <= VALUES <= 1897)
```

The ALTER TABLE statement with the ADD PARTITION or DROP PARTITION clauses can be used to add or remove ranges from an existing Kudu table.

```
ALTER TABLE foo ADD PARTITION 30 <= VALUES < 50;
ALTER TABLE foo DROP RANGE PARTITION '1' <= VALUES < '5';
```

When a range is added, the new range must not overlap with any of the previous ranges; that is, it can only fill in gaps within the previous ranges.

When a range is removed, all the associated rows in the table are deleted regardless whether the table is internal or external.

Kudu tables can also use a combination of hash and range partitioning. For example:

```
PARTITION BY HASH (school) PARTITIONS 10,
  RANGE (letter_grade) (PARTITION VALUE = 'A', PARTITION VALUE = 'B',
    PARTITION VALUE = 'C', PARTITION VALUE = 'D', PARTITION VALUE = 'F')
```

Custom hash partitioning for Kudu tables

You can now use the Impala CREATE TABLE and ALTER TABLE statements to allow custom hash partition specification at the range partition level. To display the hash schema specified for each partition, you can use the SHOW HASH SCHEMA statement.

HASH syntax within a partition is similar to the table-level syntax except that HASH clauses must follow the PARTITION clause and commas are not allowed within a partition. To keep the syntax consistent, commas in the table-level partition spec and between PARTITION clauses are now optional but allowed for backward compatibility.

Example:

```
CREATE TABLE t1 (id int, c2 int, PRIMARY KEY(id, c2))
PARTITION BY HASH(id) PARTITIONS 3 HASH(c2) PARTITIONS 4
RANGE (c2)
(
PARTITION 0 <= VALUES < 10
PARTITION 10 <= VALUES < 20
HASH(id) PARTITIONS 2 HASH(c2) PARTITIONS 3
PARTITION 20 <= VALUES < 30
)
STORED AS KUDU;


ALTER TABLE t1 ADD RANGE PARTITION 30 <= VALUES < 40
HASH(id) PARTITIONS 3 HASH(c2) PARTITIONS 4;
```

Creating external table pointing to an existing Kudu table using storage handler privileges

A user can now create an external Kudu table pointing to an existing Kudu table if the user is granted the RWSTORAGE privilege on the resource specified by a storage handler URI. Before this release, a user was required to have the ALL privilege on SERVER to create an external Kudu table for an existing Kudu table. This has been

simplified by the introduction of a new type of resource called storage handler URI and a new access type called RWSTORAGE that will be supported by Apache Ranger.

For instance, to allow a user <usr> to create an external Kudu table based on an existing Kudu table 'impala::tpch_kudu.nation', it suffices for an administrator to execute the following command to grant the necessary privilege to the user <usr>, where "localhost" is the default address of the Kudu master host assuming there is only one single master host on the cluster.

```
GRANT RWSTORAGE ON
   STORAGEHANDLER_URI 'kudu://localhost/impala::tpch_kudu.nation'
   TO USER <usr>
```

For a user to create a new Kudu table in Impala, the RWSTORAGE privilege on a specific storage handler URI or the ALL privilege on SERVER are not required. A user is allowed to create such a Kudu table once the user is granted the privilege in Ranger's Hadoop SQL policy repository to create a table under the corresponding database.

For information on configuring a policy that allows authorized users to create data tables using storage handlers, see the link under Related Information.

Related Information

Configure a resource-based storage handler policy

Impala DML for Kudu Tables

In DML statements, Impala supports specific features and properties that only apply to Kudu tables.

The UPDATE and DELETE statements let you modify data within Kudu tables without rewriting substantial amounts of table data.

The UPSERT statement acts as a combination of INSERT and UPDATE, inserting rows where the primary key does not already exist, and updating the non-primary key columns where the primary key does already exist in the table.

The INSERT statement for Kudu tables honors the unique and NOT NULL requirements for the primary key columns.

Because Impala and Kudu do not support transactions, the effects of any INSERT, UPDATE, or DELETE statement are immediately visible. For example, you cannot do a sequence of UPDATE statements and only make the changes visible after all the statements are finished. Also, if a DML statement fails partway through, any rows that were already inserted, deleted, or changed remain in the table; there is no rollback mechanism to undo the changes.

In particular, an INSERT ... SELECT statement that refers to the table being inserted into might insert more rows than expected, because the SELECT part of the statement sees some of the new rows being inserted and processes them again.

Query hints:

The INSERT or UPSERT operations into Kudu tables automatically add an exchange and a sort node to the plan that partitions and sorts the rows according to the partitioning/primary key scheme of the target table (unless the number of rows to be inserted is small enough to trigger single node execution). Since Kudu partitions and sorts rows on write, pre-partitioning and sorting takes some of the load off of Kudu and helps large INSERT operations to complete without timing out. However, this default behavior may slow down the end-to-end performance of the INSERT or UPSERT operations.

You can use the /* +NOCLUSTERED */ and /* +NOSHUFFLE */ hints together to disable partitioning and sorting before the rows are sent to Kudu. Additionally, since sorting may consume a large amount of memory, consider setting the MEM_LIMIT query option for those queries.

Loading Data into Kudu Tables

Kudu tables are well-suited to use cases where data arrives continuously, in small or moderate volumes. To bring data into Kudu tables, use the Impala INSERT and UPSERT statements.

The LOAD DATA statement does not support Kudu tables.

Because Kudu manages its own storage layer that is optimized for smaller block sizes than HDFS, and performs its own maintenance to keep data evenly distributed, it is not subject to the “many small files” issue and does not need explicit reorganization and compaction as the data grows over time.

The partitions within a Kudu table can be specified to cover a variety of possible data distributions, instead of hardcoding a new partition for each new day, hour, and so on, which can lead to inefficient, hard-to-scale, and hard-to-manage partition schemes with HDFS tables.

Your strategy for performing ETL or bulk updates on Kudu tables should take into account the limitations on consistency for DML operations.

- The INSERT, UPDATE, and UPSERT operations should produce an identical result even when executed multiple times.
- If a bulk operation is in danger of exceeding capacity limits due to timeouts or high memory usage, split it into a series of smaller operations.
- Avoid running concurrent ETL operations where the end results depend on precise ordering. In particular, do not rely on an INSERT ... SELECT statement that selects from the same table into which it is inserting, unless you include extra conditions in the WHERE clause to avoid reading the newly inserted rows within the same statement.
- Because relationships between tables cannot be enforced by Impala and Kudu, and cannot be committed or rolled back together, do not expect transactional semantics for multi-table operations.

Consistency Considerations for Kudu Tables

Kudu tables have consistency characteristics such as uniqueness, controlled by the primary key columns and non-nullable columns. The emphasis for consistency is on preventing duplicate or incomplete data from being stored in a table.

Currently, Kudu does not enforce strong consistency for order of operations, total success or total failure of a multi-row statement, or data that is read while a write operation is in progress. Changes are applied atomically to each row, but not applied as a single unit to all rows affected by a multi-row DML statement. That is, Kudu does not currently have atomic multi-row statements or isolation between statements.

If some rows are rejected during a DML operation because of a mismatch with duplicate primary key values, NOT NULL constraints, and so on, the statement succeeds with a warning. Impala still inserts, deletes, or updates the other rows that are not affected by the constraint violation.

Consequently, the number of rows affected by a DML operation on a Kudu table might be different than you expect.

Because there is no strong consistency guarantee for information being inserted into, deleted from, or updated across multiple tables simultaneously, consider denormalizing the data where practical. That is, if you run separate INSE RT statements to insert related rows into two different tables, one INSERT might fail while the other succeeds, leaving the data in an inconsistent state. Even if both inserts succeed, a join query might happen during the interval between the completion of the first and second statements, and the query would encounter incomplete inconsistent data. Denormalizing the data into a single wide table can reduce the possibility of inconsistency due to multi-table operations.

Information about the number of rows affected by a DML operation is reported in `impala-shell` output, and in the PROFILE output, but is not currently reported to HiveServer2 clients such as JDBC or ODBC applications.

Handling Date, Time, or Timestamp Data with Kudu

You can include TIMESTAMP columns in Kudu tables. The behavior of TIMESTAMP for Kudu tables has some special considerations:

- Any nanoseconds in the original 96-bit value produced by Impala are not stored, because Kudu represents date/time columns using 64-bit values. The nanosecond portion of the value is rounded, not truncated. Therefore, a TIMESTAMP value that you store in a Kudu table might not be bit-for-bit identical to the value returned by a query.
- The conversion between the Impala 96-bit representation and the Kudu 64-bit representation introduces some performance overhead when reading or writing TIMESTAMP columns. You can minimize the overhead during

writes by performing inserts through the Kudu API. Because the overhead during reads applies to each query, you might continue to use a BIGINT column to represent date/time values in performance-critical applications.

- The Impala TIMESTAMP type has a narrower range for years than the underlying Kudu data type. Impala can represent years 1400-9999. If year values outside this range are written to a Kudu table by a non-Impala client, Impala returns NULL by default when reading those TIMESTAMP values during a query. Or, if the ABORT_ON_ERROR query option is enabled, the query fails when it encounters a value with an out-of-range year.

Impala with HBase

You can use Impala to query data residing in HBase tables, a key-value data store where the value consists of multiple fields. The key is mapped to one column in the Impala table, and the various fields of the value are mapped to the other columns in the Impala table.

HBase tables are the best suited in Impala in the following use cases.

- Storing rapidly incrementing counters, such as how many times a web page has been viewed, or on a social network, how many connections a user has or how many votes a post received.

  The append-only storage mechanism of HBase is efficient for writing each change to disk, and a query always returns the latest value. An application could query specific totals like these from HBase, and combine the results with a broader set of data queried from Impala.
- Storing very wide tables in HBase.

  Wide tables have many columns, possibly thousands, typically recording many attributes for an important subject such as a user of an online service. These tables are also often sparse, that is, most of the columns values are NULL, 0, false, empty string, or other blank or placeholder value. (For example, any particular web site user might have never used some site feature, filled in a certain field in their profile, visited a particular part of the site, and so on.) A typical query against this kind of table is to look up a single row to retrieve all the information about a specific subject, rather than summing, averaging, or filtering millions of rows as in typical Impala-managed tables.

HBase works out of the box with Impala. There is no mandatory configuration needed to use these two components together.

For efficient queries, use WHERE clauses to find a single key value or a range of key values wherever practical, by testing the Impala column corresponding to the HBase row key. Avoid queries that do full-table scans, which are efficient for regular Impala tables but inefficient in HBase.

To work with an HBase table from Impala, ensure that the impala user has read/write privileges for the HBase table, using the GRANT command in the HBase shell.

Creating HBase Tables for Impala

You create the tables on the Impala side using the Hive shell, because the Impala CREATE TABLE statement currently does not support custom SerDes and some other syntax needed for HBase tables.

- You create the new table as an HBase table using the STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' clause on the Hive CREATE TABLE statement.
- You map these specially created tables to corresponding tables that exist in HBase, with the clause TBLPROPERTIES("hbase.table.name" = "*TABLE_NAME_IN_HBASE*") on the Hive CREATE TABLE statement.
- You define the column corresponding to the HBase row key as a string with the #string keyword, or map it to a STRING column.
- After creating a new table through Hive, issue the INVALIDATE METADATA statement in Impala to make Impala aware of the new table.

Because Impala and Hive share the same Metastore database, once you create the table in Hive, you can query or insert into it through Impala.

Supported Data Types for HBase Columns

HBase does not enforce any typing for the key or value fields. All the type enforcement is done on the Impala side.

HBase row key

When creating the table through the Hive shell, use the STRING data type for the column that corresponds to the HBase row key.

For best performance of Impala queries against HBase tables, most queries will perform comparisons in the WHERE clause against the column that corresponds to the HBase row key.

Impala can translate predicates (through operators such as =, <, and BETWEEN) against this column into fast lookups in HBase, but this optimization ("predicate pushdown") only works when that column is defined as STRING.

HBase numeric column

Define HBase numeric columns as the binary data type in Hive CREATE TABLE statement using the #binary (#b) keyword.

Defining numeric columns as binary can reduce the overall data volume in the HBase tables.

Impala also supports reading and writing to columns that are defined in the Hive CREATE TABLE statement using binary data types , represented in the Hive table definition using the #binary keyword, often abbreviated as #b.

Loading Data into HBase Tables

The Impala INSERT statement supports HBase tables. The INSERT ... VALUES syntax is ideally suited to HBase tables, because inserting a single row is an efficient operation for an HBase table.

When you use the INSERT ... SELECT syntax, the result in the HBase table could be fewer rows than you expect. HBase only stores the most recent version of each unique row key, so if an INSERT ... SELECT statement copies over multiple rows containing the same value for the key column, subsequent queries will only return one row with each key column value.

Successive INSERT statements using the same value for the key column achieves the same result as UPDATE.

Examples of Querying HBase Tables from Impala

The following examples create an HBase table with four column families, create a corresponding table through Hive, then insert and query the table through Impala.

In HBase, create a table. Table names are quoted in the CREATE statement in HBase.

```
hbase(main):001:0> CREATE 'hbasealltypessmall', 'boolsCF', 'intsCF', 'floats
CF', 'stringsCF'
```

Issue the following CREATE TABLE statement in the Hive shell.

This example creates an external table mapped to the HBase table, usable by both Impala and Hive. It is defined as an external table so that when dropped by Impala or Hive, the original HBase table is not touched at all.

The WITH SERDEPROPERTIES clause specifies that the first column (ID) represents the row key, and maps the remaining columns of the SQL table to HBase column families. The mapping relies on the ordinal order of the columns in the table, not the column names in the CREATE TABLE statement. The first column is defined to be the lookup key; the STRING data type produces the fastest key-based lookups for HBase tables.

Note: For Impala with HBase tables, the most important aspect to ensure good performance is to use a STRI NG column as the row key, as shown in this example.

```
hive> CREATE EXTERNAL TABLE hbasestringids (
    >   id string,
    >   bool_col boolean,
    >   tinyint_col tinyint,
```

```
    >   smallint_col smallint,
    >   int_col int,
    >   bigint_col bigint,
    >   float_col float,
    >   double_col double,
    >   date_string_col string,
    >   string_col string,
    >   timestamp_col timestamp)
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES (
    >   "hbase.columns.mapping" =
    >   ":key,boolsCF:bool_col,intsCF:tinyint_col,intsCF:smallint_col,intsCF
:int_col,intsCF:\
    >   bigint_col,floatsCF:float_col,floatsCF:double_col,stringsCF:date_s
tring_col,\
    >   stringsCF:string_col,stringsCF:timestamp_col"
    > )
    > TBLPROPERTIES("hbase.table.name" = "hbasealltypessmall");
```

Once you have established the mapping to an HBase table, you can issue DML statements and queries from Impala. The following example shows a series of INSERT statements followed by a query. The ideal kind of query from a performance standpoint retrieves a row from the table based on a row key mapped to a string column.

The INVALIDATE METADATA statement makes the table created through Hive visible to Impala.

```
[impala] > INVALIDATE METADATA hbasestringids;

[impala] > INSERT INTO hbasestringids VALUES ('0001',true,3.141,9.94,1234567
,32768,4000,76,'2014-12-31','Hello world',NOW());

[impala] > IMSERT INTO hbasestringids VALUES ('0002',false,2.004,6.196,15
00,8000,129,127,'2014-01-01','Foo bar',NOW());

[impala] > SELECT * FROM hbasestringids WHERE id = '0001';
+------+----------+------------+-------------------+------------+---------
+--------------+-------------+-----------------+-------------+-------------
------------------+
| id   | bool_col | double_col | float_col         | bigint_col | int_col |
smallint_col | tinyint_col | date_string_col | string_col  | timestamp_col
               |
+------+----------+------------+-------------------+------------+---------+
--------------+-------------+-----------------+-------------+---------------
----------------+
| 0001 | true     | 3.141      | 9.939999580383301 | 1234567    | 32768   |
 4000         | 76          | 2014-12-31      | Hello world | 2015-02-10 16
:36:59.764838000 |
+------+----------+------------+-------------------+------------+---------
+--------------+-------------+-----------------+-------------+-------------
------------------+
```

Performance Considerations for the Impala-HBase Integration

Impala uses the HBase client API via Java Native Interface (JNI) to query data stored in HBase. This querying does not read HFiles directly. The extra communication overhead makes it important to choose what data to store in HBase or in HDFS, and construct efficient queries that can retrieve the HBase data efficiently:

- Use HBase table for queries that return a single row or a small range of rows, not queries that perform a full table scan of an entire table. (If a query has a HBase table and no WHERE clause referencing that table, that is a strong indicator that it is an inefficient query for an HBase table.)
- HBase may offer acceptable performance for storing small dimension tables where the table is small enough that executing a full table scan for every query is efficient enough. However, Kudu is almost always a superior

alternative for storing dimension tables. HDFS tables are also appropriate for dimension tables that do not need to support update queries, delete queries or insert queries with small numbers of rows.

Query predicates are applied to row keys as start and stop keys, thereby limiting the scope of a particular lookup. If row keys are not mapped to string columns, then ordering is typically incorrect and comparison operations do not work. For example, if row keys are not mapped to string columns, evaluating for greater than (>) or less than (<) cannot be completed.

Predicates on non-key columns can be sent to HBase to scan as SingleColumnValueFilters, providing some performance gains. In such a case, HBase returns fewer rows than if those same predicates were applied using Impala. While there is some improvement, it is not as great when start and stop rows are used. This is because the number of rows that HBase must examine is not limited as it is when start and stop rows are used. As long as the row key predicate only applies to a single row, HBase will locate and return that row. Conversely, if a non-key predicate is used, even if it only applies to a single row, HBase must still scan the entire table to find the correct result.

Limitations and Restrictions of the Impala and HBase Integration

The Impala integration with HBase has the following limitations and restrictions, some inherited from the integration between HBase and Hive, and some unique to Impala:

- If you issue a DROP TABLE for an internal (Impala-managed) table that is mapped to an HBase table, the underlying table is not removed in HBase. The Hive DROP TABLE statement removes the HBase table in this case.
- The INSERT OVERWRITE statement is not available for HBase tables. You can insert new data, or modify an existing row by inserting a new row with the same key value, but not replace the entire contents of the table. You can do an INSERT OVERWRITE in Hive if you need this capability.
- If you issue a CREATE TABLE LIKE statement for a table mapped to an HBase table, the new table is also an HBase table, but inherits the same underlying HBase table name as the original. The new table is effectively an alias for the old one, not a new table with identical column structure. Avoid using CREATE TABLE LIKE for HBase tables, to avoid any confusion.
- Copying data into an HBase table using the Impala INSERT ... SELECT syntax might produce fewer new rows than are in the query result set. If the result set contains multiple rows with the same value for the key column, each row supercedes any previous rows with the same key value. Because the order of the inserted rows is unpredictable, you cannot rely on this technique to preserve the “latest” version of a particular key value.
- The LOAD DATA statement cannot be used with HBase tables.
- The TABLESAMPLE clause of the SELECT statement does not suppport HBase tables.

Impala with Azure Data Lake Store (ADLS)

You can use Impala to query data residing on the Azure Data Lake Store (ADLS) filesystem and Azure Blob File System (ABFS). This capability allows convenient access to a storage system that is remotely managed, accessible from anywhere, and integrated with various cloud-based services.

Impala can query files in any supported file format from ADLS. The ADLS storage location can be for an entire table or individual partitions in a partitioned table.

The default Impala tables use data files stored on HDFS, which are ideal for bulk loads and queries using full-table scans. In contrast, queries against ADLS data are less performant, making ADLS suitable for holding “cold” data that is only queried occasionally, while more frequently accessed “hot” data resides in HDFS. In a partitioned table, you can set the LOCATION attribute for individual partitions to put some partitions on HDFS and others on ADLS, typically depending on the age of the data.

Impala requires that the default filesystem for the cluster be HDFS. You cannot use ADLS as the only filesystem in the cluster.

To be able to access ADLS, first set up an Azure account, configure an ADLS store, and configure your cluster with appropriate credentials.

Creating Impala Databases, Tables, and Partitions for Data Stored on ADLS

To create a table that resides on ADLS, specify the ADLS details in the LOCATION clause of the CREATE TABLE or ALTER TABLE statement. The syntax for the LOCATION clause is:

- For ADLS Gen1:

```
LOCATION 'adl://ACCOUNT.azuredatalakestore.net/PATH/FILE'
```

- For ADLS Gen2:

```
LOCATION 'abfs://CONTAINER@ACCOUNT.dfs.core.windows.net/PATH/FILE'
```

or

```
LOCATION 'abfss://CONTAINER@ACCOUNT.dfs.core.windows.net/PATH/FILE'
```

CONTAINER denotes the parent location that holds the files and folders, which is the Containers in the Azure Storage Blobs service.

ACCOUNT is the name given for your storage account.

Any reference to an ADLS location must be fully qualified. (This rule applies when ADLS is not designated as the default filesystem.)

Once a table or partition is designated as residing on ADLS, the SELECT statement transparently accesses the data files from the appropriate storage layer.

ALTER TABLE can also set the LOCATION property for an individual partition so that some data in a table resides on ADLS and other data in the same table resides on HDFS.

Note:

By default, TLS is enabled both with abfs:// and abfss://.

When you set the fs.azure.always.use.https=false property, TLS is disabled with abfs://, and TLS is enabled with abfss://

For a partitioned table, either specify a separate LOCATION clause for each new partition, or specify a base LOCATION for the table and set up a directory structure in ADLS to mirror the way Impala partitioned tables are structured in HDFS.

Although, strictly speaking, ADLS filenames do not have directory paths, Impala treats ADLS filenames with / characters the same as HDFS pathnames that include directories.

To point a nonpartitioned table or an individual partition at ADLS, specify a single directory path in ADLS, which could be any arbitrary directory.

To replicate the structure of an entire Impala partitioned table or database in ADLS requires more care, with directories and subdirectories nested and named to match the equivalent directory tree in HDFS. Consider setting up an empty staging area if necessary in HDFS, and recording the complete directory structure so that you can replicate it in ADLS.

For example, the following session creates a partitioned table where only a single partition resides on ADLS. The partitions for years 2013 and 2014 are located on HDFS. The partition for year 2015 includes a LOCATION attribute with an adl:// URL, and so refers to data residing on ADLS, under a specific path underneath the store impalademo.

```
CREATE TABLE mostly_on_hdfs (x INT) PARTITIONED BY (year INT);
ALTER TABLE mostly_on_hdfs ADD PARTITION (year=2013);
ALTER TABLE mostly_on_hdfs ADD PARTITION (year=2014);
ALTER TABLE mostly_on_hdfs ADD PARTITION (year=2015)
   LOCATION 'adl://impalademo.azuredatalakestore.net/dir1/dir2/dir3/t1';
```

When working with multiple tables with data files stored in ADLS, you can create a database with the LOCATION attribute pointing to an ADLS path. Specify a URL of the form as shown above. Any tables created inside that database automatically create directories underneath the one specified by the database LOCATION attribute.

Use the standard ADLS file upload methods to actually put the data files into the right locations. You can also put the directory paths and data files in place before creating the associated Impala databases or tables, and Impala automatically uses the data from the appropriate location after the associated databases and tables are created.

You can switch whether an existing table or partition points to data in HDFS or ADLS. For example, if you have an Impala table or partition pointing to data files in HDFS or ADLS, and you later transfer those data files to the other filesystem, use an ALTER TABLE statement to adjust the LOCATION attribute of the corresponding table or partition to reflect that change. This location-switching technique is not practical for entire databases that have a custom LOCATION attribute.

You cannot use the ALTER TABLE ... SET CACHED statement for tables or partitions that are located in ADLS.

Internal and External Tables Located on ADLS

Just as with tables located on HDFS storage, you can designate ADLS-based tables as either internal (managed by Impala) or external, by using the syntax CREATE TABLE or CREATE EXTERNAL TABLE respectively.

When you drop an internal table, the files associated with the table are removed, even if they are on ADLS storage. When you drop an external table, the files associated with the table are left alone, and are still available for access by other tools or components.

If the data on ADLS is intended to be long-lived and accessed by other tools in addition to Impala, create any associated ADLS tables with the CREATE EXTERNAL TABLE syntax, so that the files are not deleted from ADLS when the table is dropped.

If the data on ADLS is only needed for querying by Impala and can be safely discarded once the Impala workflow is complete, create the associated ADLS tables using the CREATE TABLE syntax so that dropping the table also deletes the corresponding data files on ADLS.

Loading Data into ADLS for Impala Queries

If your ETL pipeline involves moving data into ADLS and then querying through Impala, you can either use Impala DML statements to create, move, or copy the data, or use the same data loading techniques as you would for non-Impala data.

Using Impala DML Statements for ADLS Data:

The Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in the Azure Data Lake Store (ADLS) or ADLS Gen2.

Manually Loading Data into Impala Tables on ADLS:

You can use the Microsoft-provided methods to bring data files into ADLS for querying through Impala. See the Microsoft ADLS documentation for details.

After you upload data files to a location already mapped to an Impala table or partition, or if you delete files in ADLS from such a location, issue the REFRESH statement to make Impala aware of the new set of data files.

Running and Queries for Data Stored on ADLS

Once the appropriate LOCATION attributes are set up at the table or partition level, you query data stored in ADLS the same as data stored on HDFS or in HBase:

- Queries against ADLS data support all the same file formats as for HDFS data.
- Tables can be unpartitioned or partitioned. For partitioned tables, either manually construct paths in ADLS corresponding to the HDFS directories representing partition key values, or use ALTER TABLE ... ADD PARTITION to set up the appropriate paths in ADLS.
- HDFS, Kudu, and HBase tables can be joined to ADLS tables, or ADLS tables can be joined with each other.

- Authorization to control access to databases, tables, or columns works the same whether the data is in HDFS or in ADLS.
- The `catalogd` daemon caches metadata for both HDFS and ADLS tables. Use REFRESH and INVALIDATE METADATA for ADLS tables in the same situations where you would issue those statements for HDFS tables.
- Queries against ADLS tables are subject to the same kinds of admission control and resource management as HDFS tables.
- Metadata about ADLS tables is stored in the same Metastore database as for HDFS tables.
- You can set up views referring to ADLS tables, the same as for HDFS tables.
- The COMPUTE STATS, SHOW TABLE STATS, and SHOW COLUMN STATS statements support ADLS tables.

Query Performance for ADLS Data

Although Impala queries for data stored in ADLS might be less performant than queries against the equivalent data stored in HDFS, you can still do some tuning. Here are techniques you can use to interpret explain plans and profiles for queries against ADLS data, and tips to achieve the best performance possible for such queries.

All else being equal, performance is expected to be lower for queries running against data on ADLS rather than HDFS. The actual mechanics of the SELECT statement are somewhat different when the data is in ADLS. Although the work is still distributed across the datanodes of the cluster, Impala might parallelize the work for a distributed query differently for data on HDFS and ADLS. ADLS does not have the same block notion as HDFS, so Impala uses heuristics to determine how to split up large ADLS files for processing in parallel. Because all hosts can access any ADLS data file with equal efficiency, the distribution of work might be different than for HDFS data, where the data blocks are physically read using short-circuit local reads by hosts that contain the appropriate block replicas. Although the I/O to read the ADLS data might be spread evenly across the hosts of the cluster, the fact that all data is initially retrieved across the network means that the overall query performance is likely to be lower for ADLS data than for HDFS data.

Because data files written to ADLS do not have a default block size the way HDFS data files do, any Impala INSERT or CREATE TABLE AS SELECT statements use the PARQUET_FILE_SIZE query option setting to define the size of Parquet data files. (Using a large block size is more important for Parquet tables than for tables that use other file formats.)

When optimizing aspects of for complex queries such as the join order, Impala treats tables on HDFS and ADLS the same way.

In query profile reports, the numbers for BytesReadLocal, BytesReadShortCircuit, BytesReadDataNodeCached, and BytesReadRemoteUnexpected are blank because those metrics come from HDFS.

All the I/O for ADLS tables involves remote reads, and they will appear as “remote read” operations in the query profile.

Impala with Amazon S3

You can use Impala to query data residing on the Amazon S3 object store. This capability allows convenient access to a storage system that is remotely managed, accessible from anywhere, and integrated with various cloud-based services.

Impala can query files in any supported file format from S3. The S3 storage location can be for an entire table, or individual partitions in a partitioned table.

Best Practices for Using Impala with S3

The following guidelines summarize the best practices described in the rest of this topic:

- Any reference to an S3 location must be fully qualified when S3 is not designated as the default storage, for example, s3a:://[s3-bucket-name].
- Set the fs.s3a.connection.maximum safety valve setting, to 1500 for `impalad`.

- Set the fs.s3a.block.size safety valve setting to 134217728 (128 MB in bytes) if most Parquet files queried by Impala were written by Hive or ParquetMR jobs.
- Set the PARQUET_OBJECT_STORE_SPLIT_SIZE query option to 268435456 (256 MB in bytes) if most Parquet files queried by Impala were written by Impala.
- DROP TABLE .. PURGE is much faster than the default DROP TABLE. The same applies to ALTER TABLE ... DROP PARTITION PURGE versus the default DROP PARTITION operation. However, due to the eventually consistent nature of S3, the files for that table or partition could remain for some unbounded time when using PURGE. The default DROP TABLE/PARTITION is slow because Impala copies the files to the S3A trashcan, and Impala waits until all the data is moved. DROP TABLE/PARTITION .. PURGE is a fast delete operation, and the Impala statement finishes quickly even though the change might not have propagated fully throughout S3.
- INSERT statements are faster than INSERT OVERWRITE for S3. The S3_SKIP_INSERT_STAGING query option, which is set to true by default, skips the staging step for regular INSERT (but not INSERT OVERWRITE). This makes the operation much faster, but consistency is not guaranteed: if a node fails during execution, the table could end up with inconsistent data. Set this option to false if stronger consistency is required, but this setting will make the INSERT operations slower.
  - For Impala-ACID tables, both INSERT and INSERT OVERWRITE tables for S3 are fast, regardless of the setting of S3_SKIP_INSERT_STAGING. Plus, consistency is guaranteed with ACID tables.
- Enable data cache for remote reads.
- Too many files in a table can make metadata loading and updating slow in S3. If too many requests are made to S3, S3 has a back-off mechanism and responds slower than usual.
  - If you have many small files due to over-granular partitioning, configure partitions with many megabytes of data so that even a query against a single partition can be parallelized effectively.
  - If you have many small files because of many small INSERT queries, use bulk INSERTs so that more data is written to fewer files.

Creating Impala Databases, Tables, and Partitions for Data Stored in S3

To create a table that resides in S3, run the CREATE TABLE or ALTER TABLE statement with the LOCATION clause.

ALTER TABLE can set the LOCATION property for an individual partition, so that some data in a table resides in S3 and other data in the same table resides on HDFS.

The syntax for the LOCATION clause is:

```
LOCATION 's3a://BUCKET_NAME/PATH/TO/FILE'
```

The file system prefix is always s3a://. Impala does not support the s3:// or s3n:// prefixes.

For a partitioned table, either specify a separate LOCATION clause for each new partition, or specify a base LOCATION for the table and set up a directory structure in S3 to mirror the way Impala partitioned tables are structured in S3.

You point a nonpartitioned table or an individual partition at S3 by specifying a single directory path in S3, which could be any arbitrary directory. To replicate the structure of an entire Impala partitioned table or database in S3 requires more care, with directories and subdirectories nested and named to match the equivalent directory tree in HDFS. Consider setting up an empty staging area if necessary in HDFS, and recording the complete directory structure so that you can replicate it in S3.

When working with multiple tables with data files stored in S3, you can create a database with a LOCATION attribute pointing to an S3 path. Specify a URL of the form s3a://*BUCKET*/*ROOT*/*PATH*/*FOR*/*DATABASE* for the LOCATION attribute of the database. Any tables created inside that database automatically create directories underneath the one specified by the database LOCATION attribute.

The following example creates a table with one partition for the year 2017 resides on HDFS and one partition for the year 2018 resides in S3.

The partition for year 2018 includes a LOCATION attribute with an s3a:// URL, and so refers to data residing in S3, under a specific path underneath the bucket impala-demo.

```
CREATE TABLE mostly_on_hdfs (x int) PARTITIONED BY (year INT);
ALTER TABLE mostly_on_hdfs ADD PARTITION (year=2017);
ALTER TABLE mostly_on_hdfs ADD PARTITION (year=2018)
   LOCATION 's3a://impala-demo/dir1/dir2/dir3/t1';
```

The following session creates a database and two partitioned tables residing entirely in S3, one partitioned by a single column and the other partitioned by multiple columns.

- Because a LOCATION attribute with an s3a:// URL is specified for the database, the tables inside that database are automatically created in S3 underneath the database directory.
- To see the names of the associated subdirectories, including the partition key values, use an S3 client tool to examine how the directory structure is organized in S3.

```
CREATE DATABASE db_on_s3 LOCATION 's3a://impala-demo/dir1/dir2/dir3';
CREATE TABLE partitioned_multiple_keys (x INT)
   PARTITIONED BY (year SMALLINT, month TINYINT, day TINYINT);
ALTER TABLE partitioned_multiple_keys
   ADD PARTITION (year=2015,month=1,day=1);
ALTER TABLE partitioned_multiple_keys
   ADD PARTITION (year=2015,month=1,day=31);

!hdfs dfs -ls -R s3a://impala-demo/dir1/dir2/dir3
2015-03-17 13:56:34          0 dir1/dir2/dir3/
2015-03-17 16:47:13          0 dir1/dir2/dir3/partitioned_multiple_keys/
2015-03-17 16:47:44          0 dir1/dir2/dir3/partitioned_multiple_keys/yea
r=2015/month=1/day=1/
2015-03-17 16:47:50          0 dir1/dir2/dir3/partitioned_multiple_keys/ye
ar=2015/month=1/day=31/
```

The CREATE DATABASE and CREATE TABLE statements create the associated directory paths if they do not already exist. You can specify multiple levels of directories, and the CREATE statement creates all appropriate levels, similar to using mkdir -p.

Use the standard S3 file upload methods to put the actual data files into the right locations. You can also put the directory paths and data files in place before creating the associated Impala databases or tables, and Impala automatically uses the data from the appropriate location after the associated databases and tables are created.

Use the ALTER TABLE statement with the LOCATION clause to switch whether an existing table or partition points to data in HDFS or S3. For example, if you have an Impala table or partition pointing to data files in HDFS or S3, and you later transfer those data files to the other filesystem, use the ALTER TABLE statement to adjust the LOCATION attribute of the corresponding table or partition to reflect that change.

Internal and External Tables Located in S3

Just as with tables located on HDFS storage, you can designate S3-based tables as either internal (managed by Impala) or external with the CREATE TABLE or CREATE EXTERNAL TABLE statement respectively.

When you drop an internal table, the files associated with the table are removed, even if they are in S3 storage. When you drop an external table, the files associated with the table are left alone, and are still available for access by other tools or components.

If the data in S3 is intended to be long-lived and accessed by other tools in addition to Impala, create any associated S3 tables with the CREATE EXTERNAL TABLE statement, so that the files are not deleted from S3 when the table is dropped.

If the data in S3 is only needed for querying by Impala and can be safely discarded once the Impala workflow is complete, create the associated S3 tables using the CREATE TABLE statement, so that dropping the table also deletes the corresponding data files in S3.

Loading Data into S3 for Impala Queries

If your ETL pipeline involves moving data into S3 and then querying through Impala, you can either use Impala DML statements to create, move, or copy the data, or use the same data loading techniques as you would for non-Impala data.

Using Impala DML Statements for S3 Data:

Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in S3.

Because of differences between S3 and traditional filesystems, DML operations for S3 tables can take longer than for tables on HDFS. For example, both the LOAD DATA statement and the final stage of the INSERT and CREATE TABLE AS SELECT statements involve moving files from one directory to another. (In the case of INSERT and CREATE TABLE AS SELECT, the files are moved from a temporary staging directory to the final destination directory.) Because S3 does not support a “rename” operation for existing objects, in these cases, Impala copies the data files from one location to another and then removes the original files.

Manually Loading Data into Impala Tables in S3:

You can use the Amazon-provided methods to bring data files into S3 for querying through Impala.

After you upload data files to a location already mapped to an Impala table or partition, or if you delete files in S3 from such a location outside of Impala, issue the REFRESH statement to make Impala aware of the new set of data files.

Running and Tuning Impala Queries for Data Stored in S3

Once a table or partition is designated as residing in S3, the SELECT statement transparently accesses the data files from the appropriate storage layer.

- Queries against S3 data support all the same file formats as HDFS data.
- Tables can be unpartitioned or partitioned. For partitioned tables, either manually construct paths in S3 based on partition key values, or use ALTER TABLE ... ADD PARTITION to set up the appropriate paths in S3.
- HDFS and HBase tables can be joined to S3 tables, or S3 tables can be joined with each other.
- Authorization to control access to databases, tables, or columns works the same whether the data is in HDFS or in S3.
- The Catalog Server (`catalogd`) daemon caches metadata for both HDFS and S3 tables.
- Queries against S3 tables are subject to the same kinds of admission control and resource management as HDFS tables.
- Metadata about S3 tables is stored in the same Metastore database as for HDFS tables.
- You can set up views referring to S3 tables.
- The COMPUTE STATS, SHOW TABLE STATS, and SHOW COLUMN STATS statements work for S3 tables.

Query Performance for S3

All else being equal, performance is expected to be lower for queries running against data in S3 than HDFS. The actual mechanics of the SELECT statement are different when the data is in S3. Although the work is still distributed across the DataNodes of the cluster, Impala might parallelize the work for a distributed query differently for data on HDFS and S3.

S3 does not have the same block notion as HDFS, so Impala uses heuristics to determine how to split up large S3 files for processing in parallel. Because all hosts can access any S3 data file with equal efficiency, the distribution of work might be different than for HDFS data, where the data blocks are physically read using short-circuit local reads by hosts that contain the appropriate block replicas. Although the I/O to read the S3 data might be spread evenly across the hosts of the cluster, the fact that all data is initially retrieved across the network means that the overall query performance is likely to be lower for S3 data than for HDFS data.

Use the PARQUET_OBJECT_STORE_SPLIT_SIZE query option to control the Parquet-specific split size. The default value is 256 MB.

When optimizing aspects of complex queries such as the join order, Impala treats tables on HDFS and S3 the same way. Therefore, follow all the same tuning recommendations for S3 tables as for HDFS ones, such as using the COMPUTE STATS statement to help Impala construct accurate estimates of row counts and cardinality.

In query profile reports, the numbers for BytesReadLocal, BytesReadShortCircuit, BytesReadDataNodeCached, and BytesReadRemoteUnexpected are blank because those metrics come from HDFS. By definition, all the I/O for S3 tables involves remote reads.

Restrictions on Impala Support for S3

The following restrictions apply when using Impala with S3:

- Impala does not support the old s3:// block-based and s3n:// filesystem schemes, and it only supports s3a://.
- Although S3 is often used to store JSON-formatted data, the current Impala support for S3 does not include directly querying JSON data. For Impala queries, use data files in one of the file formats listed in Hadoop File Formats Support on page 53. If you have data in JSON format, you can prepare a flattened version of that data for querying by Impala as part of your ETL cycle.
- You cannot use the ALTER TABLE ... SET CACHED statement for tables or partitions that are located in S3.

Specifying Impala Credentials to Access S3

Specify an Impala credential to access data in Amazon S3.

Procedure

1. In Cloudera Manager, navigate to Administration External Accounts .
2. In the AWS Credentials tab, click Add Access Key Credentials.
3. Enter a Name of your choosing for this account.
4. Enter the AWS Access Key ID.
5. Enter the AWS Secret Key.
6. Click Add.
7. Click Save to finish adding the AWS Credential.
8. Select Cluster Access to S3.

Impala with Ozone

You can use Impala to query data files that reside on Apache Ozone distributed storage, rather than in HDFS.

The typical use case for Impala and Ozone together is to use Ozone for the default filesystem, replacing HDFS entirely. In this configuration, when you create a database, table, or partition, the data always resides on Ozone storage and you do not need to specify any special LOCATION attribute. If you do specify a LOCATION attribute, its value refers to a path within the Ozone filesystem.

For example:

If the default filesystem is Ozone, all Impala data resides there and all Impala databases and tables are also located there.

```
CREATE TABLE t1 (x INT, s STRING);
```

You can specify LOCATION for database, table, or partition, using values from the Ozone filesystem.

```
CREATE DATABASE d1 LOCATION
'/some/path/on/ozone/server/d1.db'; CREATE TABLE d1.t2 (a TINYINT, b BOOLEAN
);
```

Impala can write to, delete, and rename data files and database, table, and partition directories on Ozone storage. Therefore, Impala statements such as CREATE TABLE, DROP TABLE, CREATE DATABASE, DROP DAT ABASE, ALTER TABLE, and INSERT work the same with Ozone storage as with HDFS.

Ozone supports multiple protocols: ofs, o3fs, and s3a. Impala supports reading ofs and o3fs. Impala can also read s3a (see Impala with Amazon S3 for more information). However, ofs is their newer protocol, and the only one Impala supports as a default filesystem. We recommend using it for DDL statements to avoid access limitations, and for DML statements and SELECT statements for performance.

Because Apache Ozone storage buckets use a global value for the block size rather than a configurable value for each file, the PARQUET_FILE_SIZE query option has no effect when Impala inserts data into a table or partition residing on Ozone storage.

Impala's spill-to-disk feature may be configured to use Ozone storage by specifying a full URI (e.g. ofs://host:port/volume/bucket/key) for the spill location.

See Spill to remote storage for details on configuring remote spill-to-disk.

Configure Impala Daemon to spill to Ozone

You can use Ozone as a scratch space for writing intermediate files during large sorts, joins, aggregations, or analytic function operations.

Before you begin

- Identify the Ozone scratch directory where you want your new Impala to write the temporary data.
- Identify the IP address, host name, or service identifier of Ozone.
- Identify the port number of the Ozone Manager (if not-default).

Configuring the Start-up Option in Impala daemon

You can use the Impalad start option scratch_dirs to specify the locations of the intermediate files.

```
--scratch_dirs="ofs://AUTHORITY/PATH(:MAX_BYTES), LOCAL_BUFFER_DIR
 (,LOCAL_DIR…)"
```

- Where ofs://*AUTHORITY*/*PATH* is the remote directory.
- authority may include ip_address or hostname and port, or service_id.
- max_bytes is optional.

Using the above format:

- You can specify only one remote directory. When you configure a remote directory, you must specify a local buffer directory as the buffer. However you can use multiple local directories with the remote directory. If you specify multiple local directories, the first local directory would be used as the local buffer directory.
- If you configure both remote and local directories, the remote directory is only used when the local directories are fully utilized.
- The size of a remote intermediate file could affect the query performance, and the value can be set by --remote _tmp_file_size=*SIZE* in the start-up option. The default size of a remote intermediate file is 16MB while the maximum is 512MB.

Examples

- An Ozone scratch dir with one local buffer dir, file size 64MB. The space of Ozone scratch dir is limited to 300G.

  ```
  --scratch_dirs=ofs://10.0.0.49:29000/tmp:300G,/local_buffer_dir --remote
  _tmp_file_size=64M
  ```

- An Ozone scratch dir with one local buffer dir limited to 512MB, and one local dir limited to 10GB. The space of Ozone scratch dir is limited to 300G. The Ozone Manager uses its default port (9862).

  ```
  --scratch_dirs=ofs://ozonemgr/tmp:300G,/local_buffer_dir:512M,/local_dir
  :10G
  ```

- An Ozone scratch dir with one local buffer dir, and multiple prioritized local dirs. The space of Ozone scratch dir is unlimited. The Ozone service identifier is ozone1.

```
--scratch_dirs=ofs://ozone1/tmp,/local_buffer_dir,/local_dir_1:5G:1,/loc
al_dir_2:5G:2
```

Even though max_bytes is optional, it is highly recommended to configure for spilling to Ozone because the Ozone cluster space is limited.

Ports Used by Impala

Impala uses the TCP ports listed in the following table.

Before deploying Impala, ensure these ports are open on each system. You can use the corresponding settings in Cloudera Manager to overwrite the default port numbers.

Scope	Cloudera Manager Setting / Startup Flag	Default Port	Comment
Impala Daemon	Impala Daemon Frontend Port --beeswax_port	21000	Port on which Beeswax client requests are served by Impala Daemons.
Impala Daemon	Impala Daemon Frontend Port --hs2_port	21050	Port on which HiveServer2 client requests are served by Impala Daemons.
Impala Daemon	Impala Daemon Backend Port --be_port	22000	Internal use only. Impala daemons use this port to communicate with each other.
Impala Daemon	StateStoreSubscriber Service Port --state_store_subscriber_port	23000	Internal use only. Impala daemons listen on this port for updates from the StateStore daemon.
Catalog Daemon	StateStoreSubscriber Service Port --state_store_subscriber_port	23020	Internal use only. The catalog daemon listens on this port for updates from the StateStore daemon.
Impala Daemon	Impala Daemon HTTP Server Port --webserver_port	25000	Impala debug Web UI for administrators to monitor and troubleshoot.
Impala StateStore Daemon	StateStore HTTP Server Port --webserver_port	25010	StateStore debug Web UI for administrators to monitor and troubleshoot.
Impala Catalog Daemon	Catalog HTTP Server Port --webserver_port	25020	Catalog Server debug Web UI for administrators to monitor and troubleshoot.
Impala StateStore Daemon	StateStore Service Port --state_store_port	24000	Internal use only. The StateStore daemon listens on this port for registration/unregistration requests.
Impala Catalog Daemon	Catalog Service Port --catalog_service_port	26000	Internal use only. The catalog service uses this port to communicate with the Impala daemons.
Impala Daemon	KRPC Port --krpc_port	27000	Internal use only. Impala daemons use this port for KRPC based communication with each other.
Impala Daemon	HiveServer2 HTTP Port --hs2_http_port	28000	Used to transmit commands and receive results by client applications over HTTP via the HiveServer2 protocol.

Related Information

Starting and Stopping Impala

Migration Guide

This topic describes the important differences between Impala in CDH and Impala in Cloudera. These changes were made in Cloudera for the optimal interoperability between Hive and Impala for the improved user experience. Review the changes before you migrate your Impala workload from CDH to Cloudera.

Default Value Changes in Configuration Options

Configuration Option	Scope	Default in CDH 6.x	Defaul
DEFAULT_FILE_FORMAT	Query	TEXT	PARQ
hms_event_polling_interval_s	Catalogd	0	2
ENABLE_ORC_SCANNER	Query	TRUE	FALSE
use_local_catalog	Coordinator / Catalogd	false	true
catalog_topic_mode	Coordinator	full	minima

New Configuration Options

Configuration Option	Scope	Default Value
default_transactional_type	Coordinator	insert_only
DEFAULT_TRANSACTIONAL_TYPE	Query	insert_only
disable_hdfs_num_rows_estimate	Impalad	false
disconnected_session_timeout	Coordinator	900
PARQUET_OBJECT_STORE_SPLIT_SIZE	Query	256 MB
SPOOL_QUERY_RESULTS	Query	FALSE
MAX_RESULT_SPOOLING_MEM	Query	100 MB
MAX_SPILLED_RESULT_SPOOLING_MEM	Query	1 GB
FETCH_ROWS_TIMEOUT_MS	Query	10 s
DISABLE_HBASE_NUM_ROWS_ESTIMATE	Query	FALSE

Default File Format

New default behavior:

In Cloudera, the default file format of the tables is Parquet.

When you issue the CREATE TABLE statement without the STORED AS clause, Impala creates a Parquet table instead of a Text table as in CDH.

For example, if you create an external table based on a text file without providing stored as clause and then tries to issue a select query, the query will fail, in Cloudera, since Impala expects the file to be in the Parquet file format.

Steps to switch to the CDH behavior:

- Add the explicitly STORED AS clause to in the CREATE TABLE statements if the file format is not Parquet.
- Start Coordinator with the default_transactional_type flag set to NONE for all tables to be created.
- Set the DEFAULT_FILE_FORMAT query option to TEXT to revert to the default of the Text format for one or more CREATE TABLE statements.

Default Insert-only Transactional Tables

New default behavior:

By default, managed tables are transactional tables with the insert_only property in Cloudera.

- You can no longer perform file system modifications (add/remove files) on a managed table in Cloudera. The directory structure for transactional tables is different than non-transactional tables, and any out-of-band files which are added may or may not be picked up by Hive and Impala.
- The insert_only transactional tables cannot be currently altered in Impala. The ALTER TABLE statement on a transactional table currently throws an error.
- Impala does not currently support compaction on transaction tables. You should use Hive to compact the tables as needed.
- The SELECT, INSERT, INSERT OVERWRITE, TRUNCATE statements are supported on the insert-only transactional tables.

Steps to switch to the CDH behavior:

- If you do not want transactional tables, set the DEFAULT_TRANSACTIONAL_TYPE query option to NONE so that any newly created managed tables are not transactional by default.
- External tables do not drop the data files when the table is dropped. If you wish to purge the data along with table when table is dropped, add external.table.purge = true in the table properties. When external.table.purge is set to true, the data is removed when the DROP TABLE statement is executed.

Managed and External Tablespace Directories

New default behavior:

In Cloudera, there are separate HDFS directories for managed and external tables.

- The data files for managed tables are located in warehouse location specified by the Cloudera Manager configuration setting, hive_warehouse_directory.
- The data files for external tables are located in warehouse location specified by the Cloudera Manager configuration setting, hive_warehouse_external_directory.

If you perform file system level operations for adding/removing files on the table, you need to consider if its an external table or managed table to find the location of the table directory.

Steps to switch to the CDH behavior:

Check the output of the DESCRIBE FORMATTED command to find the table location.

Hive Default File Format Interoperability

New default behavior:

The managed tables created by Hive are of ORC file format, by default, and support full transactional capabilities. If you create a table without specifying the STROED AS clause and load data from Hive, then such tables are not readable or writable by Impala.

Steps to switch to the CDH behavior:

- You must use the STORED AS PARQUET clause when you create tables in Hive if you want interoperability with Impala on those tables.
- If you want to change this default file format at the system level, in the Hive_on_Tez service configuration in Cloudera Manager, set the hive_default_fileformat_managed field to parquet.

Statistics Interoperability Between Hive and Impala

New default behavior:

Statistics for tables are engine specific, namely, Hive or Impala, so that each engine could use its own statistics and not overwrite the statistics generated by the other engine.

When you issue the COMPUTE STATS statement on Impala, you need to issue the corresponding statement on Hive to ensure both Hive and Impala statistics are accurate.

Impala COMPUTE STATS command does not overwrite the Hive stats for the same table.

Steps to switch to the CDH behavior:

There is no workaround.

ORC Table Support Disabled

New default behavior:

ORC table support is disabled by default. The CDH queries against ORC tables will fail in Cloudera.

Steps to switch to the CDH behavior:

Set the query option ENABLE_ORC_SCANNER to TRUE to re-enable ORC table support.

Note that this option does not work on a full transactional ORC table, and the queries will still return an error.

Reconnect to HS2 Session

New default behavior:

Clients can disconnect from Impala while keeping the session running. Clients can reconnect to the same session by presenting the session_token. By default, disconnected sessions are terminated after 15 min.

- Your end users will not notice a difference in behaviour.
- If clients are disconnected without the driver explicitly closing the session (e.g. because of a network fault), disconnected sessions and the queries associated with them may remain open and continue consuming resources until the disconnected session is timed out. Administrators may notice these disconnected sessions and/or the associated resource consumption.

Steps to switch to the CDH behavior:

You can adjust the --disconnected_session_timeout flag to a lower value so that disconnected sessions are cleaned up more quickly.

Automatic Row Count Estimation when Statistics Are Not Present

New default behavior:

If there are no statistics available on a table, Impala will try to estimate the cardinality by estimating the size of table based on the number of rows in the table. This behavior is turned on by default and should result in better plans for most cases when stats are not present.

For some edge cases, it is possible that Impala will generate a bad plan (when compared to the same query in CDH) when the statistics are not present on that table and could negatively affect the query performance.

Steps to switch to the CDH behavior:

Set the DISABLE_HDFS_NUM_ROWS_ESTIMATE query option to TRUE to disable this optimization.

Metadata V2

All catalog metadata improvements are enabled by default in CDP, resulting in many performance and scalability improvements, such as reduced memory footprint on coordinators and automatic cache eviction.

See Impala Metadata Management for the details about catalog improvements.

Amazon EC2 Instances

Impala Data Hub clusters in Cloudera are created using the Data Mart cluster type. The recommended EC2 instance type for Impala Data Mart Cloudera Data Hub clusters is r5d.2xlarge. This instance type provides a good balance of memory vs CPU and also includes fast ephemeral SSD storage which is used by Impala data cache and spill.

When provisioning a Data Mart Cloudera Data Hub cluster with the Cluster Definition option in the UI (or the --cl uster-definition-name option in the CLI), the r5d.2xlarge instance type will be selected by default.

If you choose the Custom option to provision a cluster with the Data Mart blueprint, you should explicitly specify the r5d.2xlarge instance type since this isn't the default type for Custom clusters.

Impala DWX clusters use the r5d.2xlarge instance type for executor instances and it's not possible to override this.

Authorization Support by Ranger

In Cloudera, Ranger is the authorization provider for Impala.

New behavior:

- The CREATE ROLE, GRANT ROLE, SHOW ROLE statements are not supported as Ranger currently does not support roles.
- When a particular resource is renamed, currently, the policy is not automatically transferred to the newly renamed resource.
- SHOW GRANT with an invalid user/group does not return an error.

The following table lists the different access type requirements to run SQL statements in Hive and Impala.

SQL Statement	Impala Access Requirement	Hive Access Requirement
DESCRIBE *VIEW*	VIEW_METADATA on the underlying tables	SELECT on the view
ALTER TABLE RENAME ALTER VIEW RENAME	ALL on the target table / view ALTER on the source table / view	ALTER on the source table / view
SHOW DATABASES SHOW TABLES	VIEW_METADATA	USE

Governance Support by Atlas

Both CDH and Cloudera environments support governance functionality for Impala operations. The two environments collect similar information to describe Impala activities, including:

- Audits for Impala access requests
- Metadata describing Impala queries
- Metadata describing any new data assets created or updated by Impala operations

The services that support these operations are different in the two environment. Functionality is distributed across services as follows:

Feature	CDH	CDP
Auditing		
• Access requests	Audit tab in Navigator console	Audit page in Ranger console
• Service operations that create or update metadata catalog entries	Audit tab in Navigator console	Audit tab for each entity in Atlas dashboard
• Service operations in general	Audit tab in Navigator console	No other audits collected.
Metadata Catalog		

Feature	CDH	CDP
• Impala operations: • CREATE TABLE AS SELECT • CREATE VIEW • ALTER VIEW AS SELECT • INSERT INTO • INSERT • OVERWRITE	Process and Process Execution entities Column- and table-level lineage	Process and Process Execution entities Column- and table-level lineage

Related Information

On-demand Metadata
Impala Authorization
Impala metadata collection
SQL transactions in Impala

Setting up Data Cache for Remote Reads

When Impala compute nodes and its storage are not co-located, the network bandwidth requirement goes up as the network traffic includes the data fetch as well as the shuffling exchange traffic of intermediate results. To mitigate the pressure on the network, you can enable the compute nodes to cache the working set read from remote filesystems, such as, remote HDFS data node, S3, ABFS, ADLS.

About this task

To enable remote data cache as follows.

Procedure

1. In Cloudera Manager, navigate to Clusters Impala Service .
2. In the Configuration tab, select Enable Local Data Cache to enable the local Impala Daemon data cache for caching of remote reads.
3. In Impala Daemon Data Cache Directories, add the directories Impala Daemon will use for caching of remote read data.
4. In Impala Daemon Data Cache Per Directory Capacity, specify the maximum amount of local disk space Impala will use per daemon in each of the configured directrories for caching of remote read data.
5. Click Save Changes and restart the Impala service.

Managing Metadata in Impala

This section describes various knobs you can use to control how Impala manages its metadata in order to improve performance and scalability.

On-demand Metadata

With the on-demand metadata feature, the Impala coordinators pull metadata as needed from catalogd and cache it locally. The cached metadata gets evicted automatically under memory pressure.

The granularity of on-demand metadata fetches is at the partition level between the coordinator and catalogd. Common use cases like add/drop partitions do not trigger unnecessary serialization/deserialization of large metadata.

The feature can be used in either of the following modes.

Metadata on-demand mode

In this mode, all coordinators use the metadata on-demand.

Set the following on catalogd:

```
--catalog_topic_mode=minimal
```

Set the following on all impalad coordinators:

```
--use_local_catalog=true
```

Mixed mode

In this mode, only some coordinators are enabled to use the metadata on-demand.

We recommend that you use the mixed mode only for testing local catalog's impact on heap usage.

Set the following on catalogd:

```
--catalog_topic_mode=mixed
```

Set the following on impalad coordinators with metdadata on-demand:

```
--use_local_catalog=true
```

Flags related to use_local_catalog

When use_local_catalog is enabled or set to True on the impalad coordinators the following list of flags configure various parameters as described below. It is not recommended to change the default values on these flags.

- The flag local_catalog_cache_mb (defaults to -1) configures the size of the catalog cache within each coordinator. With the default set to -1, the cache is auto-configured to 60% of the configured Java heap size. Note that the Java heap size is distinct from and typically smaller than the overall Impala memory limit.
- The flag local_catalog_cache_expiration_s (defaults to 3600) configures the expiration time of the catalog cache within each impalad. Even if the configured cache capacity has not been reached, items are removed from the cache if they have not been accessed in the defined amount of time.
- The flag local_catalog_max_fetch_retries (defaults to 40) configures the maximum number of retries needed for queries to fetch a metadata object from the impalad coordinator's local catalog cache.

Limitation:

HDFS caching is not supported in On-demand metadata mode coordinators.

Note:

In Impala 3.4.0 and above, global INVALIDATE METADATA statement is supported when On-demand feature is enabled.

INVALIDATE METADATA Usage Notes:

To return accurate query results, Impala needs to keep the metadata current for the databases and tables queried. Through "automatic invalidation" or "HMS event polling" support, Impala automatically picks up most changes in metadata from the underlying systems. However there are some scenarios where you might need to run INVALIDATE METADATA or REFRESH.

- if some other entity modifies information used by Impala in the metastore, the information cached by Impala must be updated via INVALIDATE METADATA or REFRESH,
- if you have "local catalog" enabled without "HMS event polling" and need to pick up metadata changes that were done outside of Impala in Hive and other Hive client, such as SparkSQL,

- and so on.
- **Note:** As this is a very expensive operation compared to the incremental metadata update done by the REFRESH statement, when possible, prefer REFRESH rather than INVALIDATE METADATA.

Automatic Invalidation of Metadata Cache

To keep the size of metadata bounded, the Impala Catalog Server periodically scans all the tables and invalidates those not recently used.

There are two types of configurations in Catalog Server that control the automatic invalidation of metadata in the Catalog Server Command Line Argument Advanced Configuration Snippet (Safety Valve) field in Cloudera Manager.

Time-based cache invalidation

Catalogd invalidates tables that are not recently used in the specified time period (in seconds).

The ##invalidate_tables_timeout_s flag needs to be applied to both impalad and catalogd.

Memory-based cache invalidation

When the memory pressure reaches 60% of JVM heap size after a Java garbage collection in cata logd, Impala invalidates 10% of the least recently used tables.

The ##invalidate_tables_on_memory_pressure flag needs to be applied to both impalad and cata logd.

Automatic invalidation of metadata provides more stability with lower chances of running out of memory, but the feature could potentially cause performance issues and may require tuning.

Automatic Invalidation/Refresh of Metadata

In this release, you can invalidate or refresh metadata automatically after changes to databases, tables or partitions render metadata stale. You control the synching of tables or database metadata by basing the process on events. You learn how to access metrics and state information about the event processor.

When tools such as Hive and Spark are used to process the raw data ingested into Hive tables, new HMS metadata (database, tables, partitions) and filesystem metadata (new files in existing partitions/tables) are generated. In previous versions of Impala, in order to pick up this new information, Impala users needed to manually issue an INVALIDATE or REFRESH commands.

When automatic invalidate/refresh of metadata is enabled,, the Catalog Server polls Hive Metastore (HMS) notification events at a configurable interval and automatically applies the changes to Impala catalog.

Impala Catalog Server polls and processes the following changes.

- Refreshes the tables when it receives the ALTER TABLE event.
- Refreshes the partition when it receives the ALTER, ADD, or DROP partitions.
- Adds the tables or databases when it receives the CREATE TABLE or CREATE DATABASE events.
- Removes the tables from catalogd when it receives the DROP TABLE or DROP DATABASE events.
- Refreshes the table and partitions when it receives the INSERT events.

  If the table is not loaded at the time of processing the INSERT event, the event processor does not need to refresh the table and skips it.

- Changes the database and updates catalogd when it receives the ALTER DATABASE events. The following changes are supported. This event does not invalidate the tables in the database.
  - Change the database properties
  - Change the comment on the database
  - Change the owner of the database
  - Change the default location of the database

    Changing the default location of the database does not move the tables of that database to the new location. Only the new tables which are created subsequently use the default location of the database in case it is not provided in the create table statement.

This feature is controlled by the ##hms_event_polling_interval_s flag. Start the catalogd with the ##hms_event_polling_interval_s flag set to a positive integer to enable the feature and set the polling frequency in seconds. We recommend the value to be less than 5 seconds.

Limitations

The following use cases are not supported:

- When you bypass HMS and add or remove data into table by adding files directly on the filesystem, HMS does not generate the INSERT event, and the event processor will not invalidate the corresponding table or refresh the corresponding partition.

  It is recommended that you use the LOAD DATA command to do the data load in such cases, so that event processor can act on the events generated by the LOAD command.
- The Spark API that saves data to a specified location does not generate events in HMS, thus is not supported. For example:

  ```
  Seq((1, 2)).toDF("i", "j").write.save("/user/hive/warehouse/spark_etl.db/
  customers/date=01012019")
  ```
- Event processing could have delays due to the polling interval and auto-refresh on large tables also takes time. If you want the metadata to be synced up immediately, manual REFRESH/INVALIDATE is a better choice and has a better guarantee.

Disable Event Based Automatic Metadata Sync

When the ##hms_event_polling_interval_s flag is set to a non-zero value for your catalogd, the event-based automatic invalidation is enabled for all databases and tables. If you wish to have the fine-grained control on which tables or databases need to be synced using events, you can use the impala.disableHmsSync property to disable the event processing at the table or database level.

This feature can be turned off by setting the ##hms_event_polling_interval_s flag set to 0.

When you add the DBPROPERTIES or TBLPROPERTIES with the impala.disableHmsSync key, the HMS event based sync is turned on or off. The value of the impala.disableHmsSync property determines if the event processing needs to be disabled for a particular table or database.

- If 'impala.disableHmsSync'='true', the events for that table or database are ignored and not synced with HMS.
- If 'impala.disableHmsSync'='false' or if impala.disableHmsSync is not set, the automatic sync with HMS is enabled if the ##hms_event_polling_interval_s global flag is set to non-zero.
- To disable the event based HMS sync for a new database, set the impala.disableHmsSync database properties in Hive as currently, Impala does not support setting database properties:

  ```
  CREATE DATABASE <name> WITH DBPROPERTIES ('impala.disableHmsSync'='true');
  ```

- To enable or disable the event based HMS sync for a table:

```
CREATE TABLE <name> ... TBLPROPERTIES ('impala.disableHmsSync'='true' |
'false');
```

- To change the event based HMS sync at the table level:

```
ALTER TABLE <name> SET TBLPROPERTIES ('impala.disableHmsSync'='true' | '
false');
```

When both table and database level properties are set, the table level property takes precedence. If the table level property is not set, then the database level property is used to evaluate if the event needs to be processed or not.

If the property is changed from true (meaning events are skipped) to false (meaning events are not skipped), you need to issue a manual INVALIDATE METADATA command to reset event processor because it doesn't know how many events have been skipped in the past and cannot know if the object in the event is the latest. In such a case, the status of the event processor changes to NEEDS_INVALIDATE.

Metrics for Event Based Automatic Metadata Sync

You can use the web UI of the catalogd to check the state of the automatic invalidate event processor.

By default, the debug web UI of catalogd is at http://*IMPALA-SERVER-HOSTNAME*:25020 (non-secure cluster) or https://*IMPALA-SERVER-HOSTNAME*:25020 (secure cluster).

Under the web UI, there are two pages that presents the metrics for HMS event processor that is responsible for the event based automatic metadata sync.

- /metrics#events
- /events

  This provides a detailed view of the metrics of the event processor, including min, max, mean, median, of the durations and rate metrics for all the counters listed on the /metrics#events page.

The /metrics#events page provides the following metrics about the HMS event processor.

Name	Description
events-processor.avg-events-fetch-duration	Average duration to fetch a batch of events and process it.
events-processor.avg-events-process-duration	Average time taken to process a batch of events received from the Metastore.
events-processor.events-received	Total number of the Metastore events received.
events-processor.events-received-15min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 15 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-received-1min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 1 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-received-5min-rate	Exponentially weighted moving average (EWMA) of number of events received in last 5 min. This rate of events can be used to determine if there are spikes in event processor activity during certain hours of the day.
events-processor.events-skipped	Total number of the Metastore events skipped. Events can be skipped based on certain flags are table and database level. You can use this metric to make decisions, such as: • If most of the events are being skipped, see if you might just turn off the event processing. • If most of the events are not skipped, see if you need to add flags on certain databases.

Name	Description
events-processor.status	Metastore event processor status to see if there are events being received or not. Possible states are: • PAUSED The event processor is paused because catalog is being reset concurrently. • ACTIVE The event processor is scheduled at a given frequency. • ERROR • The event processor is in error state and event processing has stopped. • NEEDS_INVALIDATE The event processor could not resolve certain events and needs a manual INVALIDATE command to reset the state. • STOPPED The event processing has been shutdown. No events will be processed. • DISABLED The event processor is not configured to run.

SQL transactions in Impala

A transaction is a single logical operation on the data. Impala supports transactions that satisfy a level of consistency that improves the integrity and reliability of the data before and after a transaction.

Specifically, Impala provides atomicity and isolation of insert operations on transactional tables. A single table insert is either committed in full or not committed, and the results of the insert operation are not visible to other query operations until the operation is committed.

For single table, the inserts are ordered, so if Impala doesn't see a committed insert, it won't see any insert committed after it.

For insert-only transactional tables, you can perform the following statements: CREATE TABLE, DROP TABLE, TRUNCATE, INSERT, SELECT

All transactions in Impala automatically commit at the end of the statement. Currently, Impala does not support multi-statement transactions.

Insert-only tables must be the managed and file-format based tables, such as Parquet, Avro, and text.

Note:

- Impala does not support changing transactional properties of tables. For example, you cannot alter a transactional table to a non-transactional table.
- Currently, ALTER TABLE statement is not supported on both insert-only and full acid transactional tables.

Cloudera Runtime 7.3.1

Securing Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Securing Impala....4
Configuring Impala TLS/SSL....5
Impala Authentication....6
Configuring Kerberos Authentication for Impala....6
Configuring LDAP Authentication....7
Enabling LDAP in Hue....9
Enabling LDAP Authentication for impala-shell....9
Configuring JWT Authentication....9
Enabling JWT Authentication for impala-shell....10
Impala Authorization....11
Configuring Authorization....18
Row-level filtering in Impala with Ranger policies....18

Securing Impala

You must be aware of this set of security features Impala provides to protect your critical and sensitive data.

The Impala security features have several objectives.

- At the basic level, security prevents accidents or mistakes that could disrupt application processing, delete or corrupt data, or reveal data to unauthorized users.
- Advanced security features and practices can harden the system against malicious users trying to gain unauthorized access or perform other disallowed operations.
- The auditing feature provides a way to confirm that no unauthorized access occurred, and detect whether any such attempts were made.

Based on the above objectives, Impala security features are divided into the following categories.

Authentication

How does Impala verify the identity of the user to confirm that they really are allowed to exercise the privileges assigned to that user? Impala relies on the Kerberos subsystem for authentication.

Authorization

Which users are allowed to access which resources, and what operations are they allowed to perform? Impala relies on the open source Ranger project for authorization. By default (when authorization is not enabled), Impala does all read and write operations with the privileges of the impala user, which is suitable for a development/test environment but not for a secure production environment. When authorization is enabled, Impala uses the OS user ID of the user who runs `impala-shell` or other client program, and associates various privileges with each user.

TLS/SSL Encryption

This feature encrypts TLS/SSL network encryption, between Impala and client programs, and between the Impala-related daemons running on different nodes in the cluster.

Auditing

What operations were attempted, and did they succeed or not? This feature provides a way to look back and diagnose whether attempts were made to perform unauthorized operations. You use this information to track down suspicious activity, and to see where changes are needed in authorization policies. The audit data produced by this feature is collected by the Cloudera Manager product and then presented in a user-friendly form by the Cloudera Manager product.

The following are the important guidelines to protect a cluster running Impala against accidents and mistakes, or malicious attackers trying to access sensitive data. See the subsequent topics that describe the security features in detail.

- Secure the root account. The root user can tamper with the `impalad` daemon, read and write the data files in HDFS, log into other user accounts, and access other system services that are beyond the control of Impala.
- Restrict membership in the sudoers list (in the /etc/sudoers file). The users who can run the sudo command can do many of the same things as the root user.
- Ensure the Hadoop ownership and permissions for Impala data files are restricted.

  The Impala authorization feature makes use of the HDFS file ownership and permissions mechanism.

  1. Set up users and assign them to groups at the OS level, corresponding to the different categories of users with different access levels for various databases, tables, and HDFS locations (URIs).
  2. Create the associated Linux users using the `useradd` command if necessary.
  3. Add Linux users to the appropriate groups with the `usermod` command.
- Ensure the Hadoop ownership and permissions for Impala log files are restricted.

  If you issue queries containing sensitive values in the WHERE clause, such as financial account numbers, those values are stored in Impala log files in the Linux file system, and you must secure those files.

- Ensure that the Impala web UI (available by default on port 25000 on each Impala node) is password-protected.
- Create a policy file that specifies which Impala privileges are available to users in particular Hadoop groups (which by default map to Linux OS groups). Create the associated Linux groups using the `groupadd` command if necessary.
- Design your databases, tables, and views with database and table structure to allow policy rules to specify simple, consistent rules.

  For example, if all tables related to an application are inside a single database, you can assign privileges for that database and use the * wildcard for the table name. If you are creating views with different privileges than the underlying base tables, you might put the views in a separate database so that you can use the * wildcard for the database containing the base tables, while specifying the specific names of the individual views.
- Enable authorization for all impalad daemons.

  The authorization feature does not apply to the `statestored` daemon, which has no access to schema objects or data files.
- Set up authentication using Kerberos, to make sure users really are who they say they are.

Related Information

Configuring Impala Web UI

Configuring Impala TLS/SSL

To protect sensitive information being transmitted, Impala supports TLS/SSL network encryption, between Impala and client programs, and between the Impala-related daemons running on different nodes in the cluster.

To configure Impala to listen for Beeswax and HiveServer2 requests on TLS/SSL-secured ports:

1. In Cloudera Manager, select the Impala service from the Clusters drop down.
2. In the Configuration tab, select ScopeImpala (Service-Wide).
3. Select CategorySecurity.
4. Edit the following property fields:

Property	Description
Enable TLS/SSL for Impala	Encrypt communication between clients (like ODBC, JDBC, and the Impala shell) and the Impala daemon using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).
Impala TLS/SSL Server Certificate File (PEM Format)	Local path to the X509 certificate that identifies the Impala daemon to clients during TLS/SSL connections. This file must be in PEM format.
Impala TLS/SSL Server Private Key File (PEM Format)	Local path to the private key that matches the certificate specified in the Certificate for Clients. This file must be in PEM format.
Impala TLS/SSL Private Key Password	The password for the private key in the Impala TLS/SSL Server Certificate and Private Key file. If left blank, the private key is not protected by a password.
Impala TLS/SSL CA Certificate	The location on disk of the certificate, in PEM format, used to confirm the authenticity of SSL/TLS servers that the Impala daemons might connect to. Because the Impala daemons connect to each other, this should also include the CA certificate used to sign all the SSL/TLS Certificates. Without this parameter, SSL/TLS between Impala daemons will not be enabled.

Using TLS/SSL with Business Intelligence Tools

You can use Kerberos authentication, TLS/SSL encryption, or both to secure connections from JDBC and ODBC applications to Impala.

Configuring TLS/SSL Communication for the Impala Shell

Typically, a client program has corresponding configuration properties in Cloudera Manager to verify that it is connecting to the right server. For example, with SSL enabled for Impala, you use the following options when starting the `impala-shell` interpreter:

- --ssl: enables TLS/SSL for `impala-shell`.
- --ca_cert: the local pathname pointing to the third-party CA certificate, or to a copy of the server certificate for self-signed server certificates.

If --ca_cert is not set, `impala-shell` enables TLS/SSL, but does not validate the server certificate. This is useful for connecting to a known-good Impala that is only running over TLS/SSL, when a copy of the certificate is not available (such as when debugging customer installations).

For `impala-shell` to successfully connect to an Impala cluster that has the minimum allowed TLS/SSL version set to 1.2 (--ssl_minimum_version=tlsv1.2), the cluster that `impala-shell` runs on must have Python version 2.7.9 or higher (or a vendor-provided Python version with the required support. Some vendors patched Python 2.7.5 versions on Red Hat Enterprise Linux 7 and derivatives).

Specifying TLS/SSL Minimum Allowed Version and Ciphers

Depending on your cluster configuration and the security practices in your organization, you might need to restrict the allowed versions of TLS/SSL used by Impala. Older TLS/SSL versions might have vulnerabilities or lack certain features. You can use startup options for the `impalad`, `catalogd`, and `statestored` daemons to specify a minimum allowed version of TLS/SSL.

Add the following parameter into Cloudera Manager Impala Configuration . Search "Impala Command Line Argument Advanced Configuration Snippet (Safety Valve)"

```
--ssl_minimum_version=tlsv1.2
```

Along with specifying the version, you can also specify the allowed set of TLS ciphers by using the --ssl_cipher_list configuration setting. The argument to this option is a list of keywords, separated by colons, commas, or spaces, and optionally including other notation. For example:

```
--ssl_cipher_list="DEFAULT:!aNULL:!eNULL:!LOW:!EXPORT:!SSLv2:!SSLv3:!TLS1"
```

By default, the cipher list is empty, and Impala uses the default cipher list for the underlying platform. See the output of `man ciphers` for the full set of keywords and notation allowed in the argument string.

Impala Authentication

Authentication is the mechanism to ensure that only specified hosts and users can connect to Impala.

Authentication feature verifies that when clients connect to Impala, they are connected to a legitimate server. The feature prevents spoofing such as *impersonation* (setting up a phony client system with the same account and group names as a legitimate user) and *person-in-the-middle attacks* (intercepting application requests before they reach Impala and eavesdropping on sensitive information in the requests or the results).

Impala automatically handles both Kerberos and LDAP authentication. Each `impalad` daemon can accept both Kerberos and LDAP requests through the same port. No special actions need to be taken if some users authenticate through Kerberos and some through LDAP.

You can also make proxy connections to Impala through Apache Knox.

Regardless of the authentication mechanism used, Impala always creates HDFS directories and data files owned by the same user (typically impala). Once you are finished setting up authentication, set up authorization to control user-level access to databases, tables, columns, partitions when they connect through Impala.

Configuring Kerberos Authentication for Impala

Requirements for Using Impala with Kerberos

Important:

- If you plan to use Impala in your cluster, you must configure your KDC to allow tickets to be renewed, and you must configure krb5.conf to request renewable tickets. Typically, you can do this by adding the max_renewable_life setting to your realm in kdc.conf, and by adding the renew_lifetime parameter to the libdefaults section of krb5.conf.

Impala supports the ODBC driver and the Kerberos interface provided. To use Kerberos through the ODBC driver, the host type must be set depending on the level of the ODBD driver:

- SecImpala for the ODBC 1.0 driver.
- SecBeeswax for the ODBC 1.2 driver.
- Blank for the ODBC 2.0 driver or higher, when connecting to a secure cluster.
- HS2NoSasl for the ODBC 2.0 driver or higher, when connecting to a non-secure cluster.

Enabling Kerberos in Impala-shell

To enable Kerberos in the Impala shell, start the `impala-shell` command using the -k flag.

Enabling Access to Internal Impala APIs for Kerberos Users

For applications that need direct access to Impala APIs, without going through the HiveServer2 or Beeswax interfaces, you can specify a list of Kerberos users who are allowed to call those APIs. By default, the impala and hdfs users are the only ones authorized for this kind of access. Any users not explicitly authorized through the internal _principals_whitelist configuration setting are blocked from accessing the APIs. This setting applies to all the Impala-related daemons, although currently it is primarily used for HDFS to control the behavior of the catalog server.

Mapping Kerberos Principals to Short Names for Impala

Impala can support the additional mapping rules that will be inserted before rules generated from the list of trusted realms and before the default rule. The support is disabled by default in Impala.

To enable mapping Kerberos principals to short names:

1. In Cloudera Manager, select the Impala service.
2. In the Configuration tab, select Impala (Service-Wide) in Scope and Advanced in Category.
3. Select the Use HDFS Rules to Map Kerberos Principals to Short Names field.
4. Click Save Changes, and restart the Impala service.

Configuring LDAP Authentication

Impala uses LDAP for authentication, verifying the credentials of each user who connects through `impala-shell`, Hue, a Business Intelligence tool, JDBC or ODBC applications.

About this task

Authentication is the process of allowing only specified named users to access the server (in this case, the Impala server). This feature is crucial for any production deployment, to prevent misuse, tampering, or excessive load on the server.

Only the connections between clients and Impala can be authenticated by LDAP.

Procedure

To enable and configure LDAP:

1. In Cloudera Manager, select the Impala service.
2. In the Configuration tab, type ldap in the search box. The fields for LDAP configuration will be listed.

3. Set the following fields to enable LDAP.

   Enable LDAP Authentication (enable_ldap_auth)

   Enables LDAP-based authentication between the client and Impala.

   LDAP URL (ldap_uri)

   Sets the URI of the LDAP server to use. Typically, the URI is prefixed with ldap://. You can specify secure SSL-based LDAP transport by using the prefix ldaps://. The URI can optionally specify the port, for example: ldap://ldap_server.example.com:389 or ldaps://ldap_server.example.com:636. (389 and 636 are the default ports for non-SSL and SSL LDAP connections, respectively.)

4. Set the following fields to support custom bind strings.

   When Impala connects to LDAP, it issues a bind call to the LDAP server to authenticate as the connected user. Impala clients, including the Impala-shell, provide the short name of the user to Impala. This is necessary so that Impala can use role-based access, which uses short names.

   However, LDAP servers often require more complex, structured usernames for authentication. Impala supports three ways of transforming the short name (for example, 'henry') to a more complicated string. If necessary, specify one of the following configuration options when starting the `impalad` daemon.

   Active Directory Domain (--ldap_domain)

   Replaces the username with a string *USERNAME@LDAP_DOMAIN*.

   LDAP BaseDN (--ldap_baseDN)

   Replaces the username with a “distinguished name” (DN) of the form: uid=*USERID*,ldap_baseDN. (This is equivalent to a Hive option).

   LDAP Pattern (--ldap_bind_pattern)

   This is the most general option, and replaces the username with the string *LDAP_BIND_PATTERN* where all instances of the string #UID are replaced with *USERID*. For example, an ldap_bind_pattern of "user=#UID,OU=foo,CN=bar" with a username of henry will construct a bind name of "user=henry,OU=foo,CN=bar".

   The above options are mutually exclusive, and Impala does not start if more than one of these options are specified.

5. Set the following fields for secure LDAP connections.

   To avoid sending credentials over the wire in cleartext, you must configure a secure connection between both the client and Impala, and between Impala and the LDAP server. The secure connection could use SSL or TLS.

   Secure LDAP connections through SSL:

   For SSL-enabled LDAP connections, specify a prefix of ldaps:// instead of ldap://. Also, the default port for SSL-enabled LDAP connections is 636 instead of 389.

   Secure LDAP connections through TLS:

   TLS, the successor to the SSL protocol, is supported by most modern LDAP servers. Unlike SSL connections, TLS connections can be made on the same server port as non-TLS connections. To secure all connections using TLS, specify the following flags as startup options to the `impalad` daemon:

   Enable LDAP TLS (--ldap_tls_

   Tells Impala to start a TLS connection to the LDAP server, and to fail authentication if it cannot be done.

   LDAP Server CA Certificate (--ldap_ca_certificate)

   Specifies the location of the certificate in standard .PEM format. Store this certificate on the local filesystem, in a location that only the impala user and other trusted users can read.

6. Click Save Changes and restart the Impala service.

Enabling LDAP in Hue

To connect to Impala using LDAP authentication through Hue, enable the LDAP setting in Hue.

Procedure

1. Go to the Hue service.
2. Click the Configuration tab.
3. Select ScopeHue Server.
4. Select CategoryAdvanced.
5. Add the following properties to the Hue Server Advanced Configuration Snippet (Safety Valve) for hue_safety_valve_server.ini property

```
[impala]
auth_username=<LDAP username of Hue user to be authenticated>
auth_password=<LDAP password of Hue user to be authenticated>
```

6. Click Save Changes.

Enabling LDAP Authentication for impala-shell

To connect to Impala using LDAP authentication, you specify command-line options to the `impala-shell` command interpreter and enter the password when prompted.

-l

Enables LDAP authentication.

-u

Sets the user. Per Active Directory, the user is the short username, not the full LDAP distinguished name. If your LDAP settings include a search base, use the --ldap_bind_pattern on the `impalad` daemon to translate the short user name from `impala-shell` automatically to the fully qualified name.

`impala-shell` automatically prompts for the password.

Configuring JWT Authentication

Impala shell users can now provide a JWT instead of a username/password to authenticate to an Impala instance. When JWT authentication is used, the impala shell enforces the use of the hs2-http protocol since the JWT is sent via the "Authentication" HTTP header.

Procedure

To enable and configure JWT:

1. In Cloudera Manager, select the Impala service.
2. In the Configuration tab, type jwt in the search box. The fields for JWT configuration will be listed.

3. Set the following fields to enable JWT.

JWKS URL (jwks_url)

URL where the JSON Web Key Set (JWKS) can be downloaded for JWT verification. The default value is none, users will need to get the JWKS URL from their authentication provider's documentation.

Username JWT Custom Claim (jwt_custom_claim_username)

JWT claim that contains the username to use when authenticating with Impala. The default value is none, users will need to reference their authentication provider's documentation to determine which JWT payload claim contains the username.

JWT Token Authentication (jwt_token_auth)

Determines if JWT authentication is allowed without TLS being enabled on connections to the Impala daemon. The default value is false. Impala writes a warning message in the log file if it’s set as true since it’s not secure. Only set as true for test environments without sensitive data.

JWT Validate Signature (jwt_validate_signature)

Determines if the signatures on incoming JWTs are validated against the JWKS. If set as true, Impala verifies the signature in JWT with pre-installed public keys. The default value is true. Only set as true for test environments without sensitive data.

JWKS Pull Timeout (jwks_pulling_timeout_s)

The time in seconds to wait for the JWKS to be downloaded from the specified URL before timing out. This flag is used when jwks_url is specified. The default value is 10 seconds.

JWKS Update Frequency (jwks_update_frequency_s)

The time in seconds to wait between re-downloading the JWKS from the specified URL. This flag is used when jwks_url is specified. The default value is 60 seconds.

Verify JWKS Server Certificate (jwks_verify_server_certificate)

Specifies if the TLS certificate of the JWKS server is verified when retrieving the JWKS from the specified JWKS URL. This should only be set to false for development / testing. The default value is true.

Enabling JWT Authentication for impala-shell

To connect to Impala using JWT authentication, you specify the following command-line options to the impala-shell command interpreter and enter the password when prompted.

-j, --jwt

indicates that JWT authentication will be used

--jwt_cmd

shell command to run to retrieve the JWT to be used for authentication

Note: When using the JWT authentication, impala shell uses the hs2-https protocol by default.

Examples

Reading the JWT from a file named my_jwt.txt.

```
impala-shell -i impala_hostname:port -j --ssl --jwt_cmd "cat my_jwt.txt"
```

Note: To read the JWT from a file, you must have created a file named "my_jwt.txt" and stored the JWT without the quotes.

Reading the JWT from an environment variable named MY_JWT_ENV_VAR

```
impala-shell --ssl -i impala_hostname:port -j --jwt_cmd "echo ${MY_JWT_ENV_V
AR}"
```

Reading the JWT from the clipboard (macOS only)

```
impala-shell --ssl -i impala_hostname:port -j --jwt_cmd "pbpaste"
```

Impala Authorization

Authorization determines which users are allowed to access which resources, and what operations they are allowed to perform. You use Apache Ranger to enable and manage authorization in Impala.

Supported Ranger Features in Impala

- Resource-based and tag-based authorization policies
- Resource-based and tag-based column masking
- Row-level filtering is enabled by default

Using Resource-based Authorization Policies for Impala

In the Ranger Service Manager, you can use the Hadoop SQL preloaded resource-based and tag-based services and policies to authorize access to Impala:

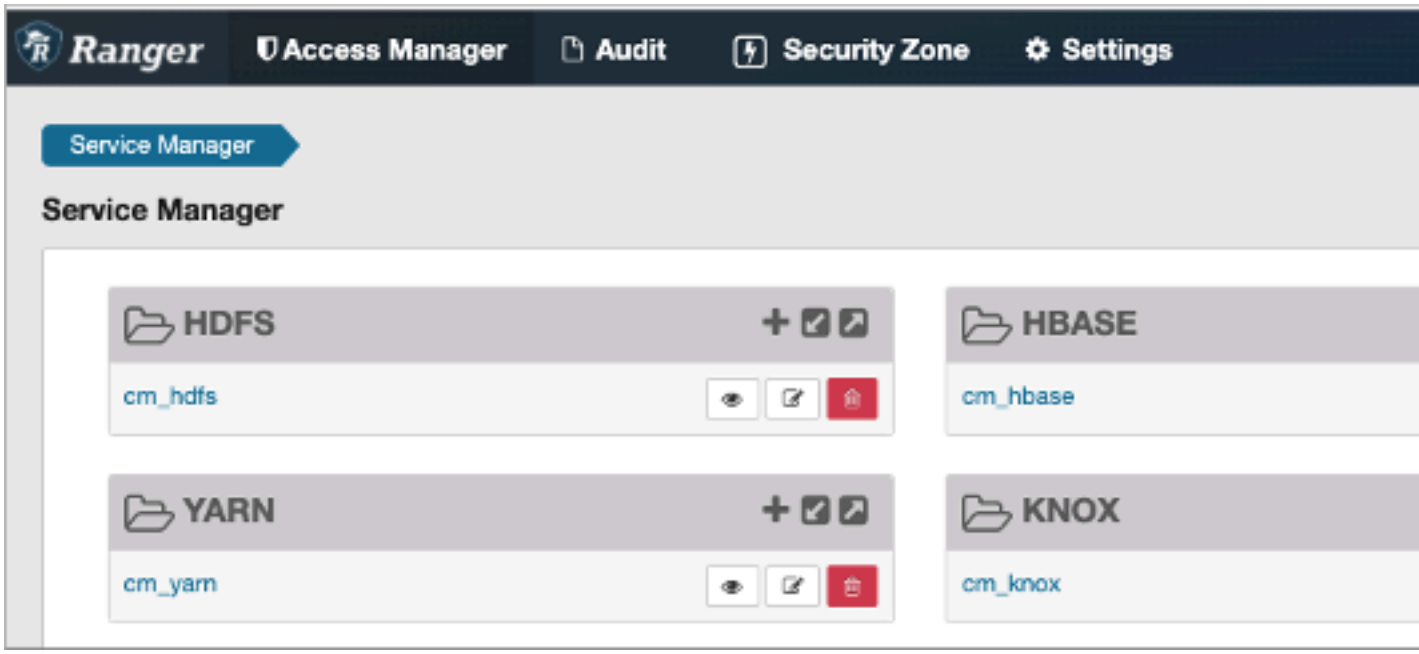

You can configure services for Impala as described in Using Ranger to Provide Authorization in Cloudera.

For example, you can edit the all-global policy for an Impala user:

For information about using tag-based policies, see Tag-based column masking in Hive with Ranger policies.

Privilege Model

You set up privileges through the GRANT and REVOKE statements in either Impala or Hive. Then both components use those same privileges automatically.

By default, when authorization is not enabled, Impala does all read and write operations with the privileges of the impala user, which is suitable for a development/test environment but not for a secure production environment. When authorization is enabled, Impala uses the OS user ID of the user who runs `impala-shell` or other client programs, and associates various privileges with each user.

Privileges can be granted on different resources in the schema and are associated with a level in the resource hierarchy. A privilege on a particular resource automatically inherits the same privilege of its parent.

The resource hierarchy is:

```
Server
    URI
    Database
        Table
            Column
        Function
```

The table-level privileges apply to views as well. Anywhere you specify a table name, you can specify a view name instead.

You can specify privileges for individual columns.

The table below lists the minimum level of privileges and the scope required to run SQL statements in Impala. The following notations are used:

- The SERVER resource type in Ranger implies all databases, all tables, all columns, all UDFs, and all URLs.
  - **Note:** For Impala server level access, you must add the user to the below listed default policies in Ranger:

    all - database, table, column

    all - database, udf

    all - url
- ANY denotes the CREATE, ALTER, DROP, SELECT, INSERT, or REFRESH privilege.
- ALL privilege denotes the SELECT, INSERT, CREATE, ALTER, DROP, and REFRESH privileges.
- VIEW_METADATA privilege denotes the SELECT, INSERT, or REFRESH privileges.
- The parent levels of the specified scope are implicitly supported. For example, if a privilege is listed with the TABLE scope, the same privilege granted on DATABASE and SERVER will allow the user to run that specific SQL statement on TABLE.

Note: For an "impala" user to access any managed tables under the warehouse root directory or external tables with directories outside the warehouse root directory, the "impala" user must have HDFS RW access to the underlying paths that are referenced in a query.

Without this HDFS RW access, you may see Impala queries failing with HDFS errors such as this:

Error(13): Permission denied

Root cause: RemoteException: Permission denied: user=impala, access=WRITE, inode="/path/external/table":hive:supergroup:drwxr-xr-x

For example, to be able to run CREATE VIEW, you need the CREATE privilege on the database and the SELECT privilege on the source table.

SQL Statement	Privileges	Object Type / Resource Type
SELECT	SELECT	TABLE
WITH SELECT	SELECT	TABLE
EXPLAIN SELECT	SELECT	TABLE
INSERT	INSERT	TABLE
EXPLAIN INSERT	INSERT	TABLE
TRUNCATE	INSERT	TABLE
LOAD	INSERT	TABLE
	ALL	URI
CREATE DATABASE	CREATE	SERVER
CREATE DATABASE LOCATION	CREATE	SERVER
	ALL	URI
CREATE TABLE	CREATE	DATABASE
CREATE TABLE LIKE	CREATE	DATABASE
	VIEW_METADATA	TABLE
CREATE TABLE AS SELECT	CREATE	DATABASE
	INSERT	DATABASE
	SELECT	TABLE
EXPLAIN CREATE TABLE AS SELECT	CREATE	DATABASE
	INSERT	DATABASE
	SELECT	TABLE
CREATE TABLE LOCATION	CREATE	TABLE
	ALL	URI
CREATE VIEW	CREATE	DATABASE
	SELECT	TABLE
ALTER DATABASE SET OWNER	ALL WITH GRANT	DATABASE
ALTER TABLE	ALL	TABLE
ALTER TABLE SET LOCATION	ALL	TABLE
	ALL	URI
ALTER TABLE RENAME	CREATE	DATABASE
	ALL	TABLE

ALTER TABLE SET OWNER	ALL WITH GRANT	TABLE
ALTER VIEW	ALL	TABLE
	SELECT	TABLE
ALTER VIEW RENAME	CREATE	DATABASE
	ALL	TABLE
ALTER VIEW SET OWNER	ALL WITH GRANT	VIEW
DROP DATABASE	ALL	DATABASE
DROP TABLE	ALL	TABLE
DROP VIEW	ALL	TABLE
CREATE FUNCTION	CREATE	DATABASE
	ALL	URI
DROP FUNCTION	ALL	DATABASE
COMPUTE STATS	ALL	TABLE
DROP STATS	ALL	TABLE
INVALIDATE METADATA	REFRESH	SERVER
INVALIDATE METADATA <table>	REFRESH	TABLE
REFRESH <table>	REFRESH	TABLE
REFRESH AUTHORIZATION	REFRESH	SERVER
REFRESH FUNCTIONS	REFRESH	DATABASE
COMMENT ON DATABASE	ALL	DATABASE
COMMENT ON TABLE	ALL	TABLE
COMMENT ON VIEW	ALL	TABLE
COMMENT ON COLUMN	ALL	TABLE
DESCRIBE DATABASE	VIEW_METADATA	DATABASE
DESCRIBE <table/view>	VIEW_METADATA	TABLE
If the user has the SELECT privilege at the COLUMN level, only the columns the user has access will show.	SELECT	COLUMN
USE	ANY	TABLE
SHOW DATABASES	ANY	TABLE
SHOW TABLES	ANY	TABLE
SHOW FUNCTIONS	VIEW_METADATA	DATABASE
SHOW PARTITIONS	VIEW_METADATA	TABLE
SHOW TABLE STATS	VIEW_METADATA	TABLE
SHOW COLUMN STATS	VIEW_METADATA	TABLE
SHOW FILES	VIEW_METADATA	TABLE
SHOW CREATE TABLE	VIEW_METADATA	TABLE
SHOW CREATE VIEW	VIEW_METADATA	TABLE
SHOW CREATE FUNCTION	VIEW_METADATA	DATABASE
SHOW RANGE PARTITIONS (Kudu only)	VIEW_METADATA	TABLE
UPDATE (Kudu only)	ALL	TABLE

EXPLAIN UPDATE (Kudu only)	ALL	TABLE
UPSERT (Kudu only)	ALL	TABLE
WITH UPSERT (Kudu only)	ALL	TABLE
EXPLAIN UPSERT (Kudu only)	ALL	TABLE
DELETE (Kudu only)	ALL	TABLE
EXPLAIN DELETE (Kudu only)	ALL	TABLE

The privileges not listed in the table above will be silently ignored by Impala.

Changing Privileges in Impala

Privileges are managed via the GRANT and REVOKE SQL statements that require the Ranger service enabled.

Privileges can be also managed in Ranger UI. Especially, for attribute-based access control, Ranger UI is required to manage authorization.

Impala authorization policies are listed in the Hive service section in Ranger UI.

REFRESH AUTHORIZATION is not required when you make the changes to privileges within Impala. The changes are automatically propagated.

Changing Privileges from Outside of Impala

If you make a change to privileges in Ranger from outside of Impala, e.g. adding a user, removing a user, modifying privileges, there are two options to propagate the change:

- Use the ranger.plugin.hive.policy.pollIntervalMs property to specify how often to do a Ranger refresh. The property is specified in ranger-impala-security.xml in the conf directory under your Impala home directory.
- Run the REFRESH AUTHORIZATION statement to force a refresh.

Warning: As INVALIDATE METADATA is an expensive operation, you should use it judiciously.

Granting Privileges on URI

URIs represent the file paths you specify as part of statements such as CREATE EXTERNAL TABLE and LOAD DATA. Typically, you specify what look like UNIX paths, but these locations can also be prefixed with hdfs:// to make clear that they are really URIs. To set privileges for a URI, specify the name of a directory, and the privilege applies to all the files in that directory.

URIs must start with hdfs://, s3a://, adl://, or file://. If a URI starts with an absolute path, the path will be appended to the default filesystem prefix. For example, if you specify:

```
GRANT ALL ON URI '/tmp';
```

The above statement effectively becomes the following where the default filesystem is HDFS.

```
GRANT ALL ON URI 'hdfs://localhost:20500/tmp';
```

When defining URIs for HDFS, you must also specify the NameNode. For example:

```
GRANT ALL ON URI file:///path/to/dir TO <principal>
          GRANT ALL ON URI hdfs://namenode:port/path/to/dir TO <principal>
```

Warning: Because the NameNode host and port must be specified, it is strongly recommended that you use High Availability (HA). This ensures that the URI will remain constant even if the NameNode changes. For example:

```
GRANT ALL ON URI hdfs://ha-nn-uri/path/to/dir TO <principal>
```

Note:

```
<principal> can be a user, or a group
```

Note:

- If a user uses Impala SQL engine to access the resource of the specified URL/URI provided as part of the SQL statement and as an admin if you must deny access permissions to this location for this particular user, you must add the permission "All" in the "HADOOP SQL" policy.
- However, if the user is using Hive as the execution engine and to deny access permission to a location, then you must add both the “Read” and “Write” permissions in the field of ‘Permissions’ for the corresponding deny conditions.
- If the same Ranger policy is shared by both Hive and Impala, then you must add “All”, “Read”, and “Write” to the field of ‘Permissions’ to enforce the policy.

Object Ownership

Object ownership for tables, views and databases is enabled by default in Impala.

To define owner specific privileges, go to Ranger UI and define appropriate policies on the {OWNER} user.

The CREATE statements implicitly make the user running the statement the owner of the object. For example, if *USER A* creates a database, *FOO*, via the CREATE DATABASE statement, *USER A* now owns the *FOO* database and is authorized to perform any operation on the *FOO* database.

An ownership can be transferred to another user or role via the ALTER DATABASE, ALTER TABLE, or ALTER VIEW with the SET OWNER clause.

Note: Currently, due to a known issue (IMPALA-8937), until the ownership information is fully loaded in the coordinator catalog cache, the owner of a table might not be able to see the table when executing the SHOW TABLES statement. The owner can still query the table.

Ranger Column Masking for Impala

Ranger column masking hides sensitive columnar data in Impala query output. For example, you can define a policy that reveals only the first or last four characters of column data. Column masking is enabled by default. To disable column masking, modify the following configuration property in all coordinators:

```
--enable_column_masking=false
```

This flag might be removed in a future release. The Impala behavior mimics Hive behavior with respect to column masking.

The following table lists all Impala-supported, built-in mask types for defining column masking in a policy using the Ranger REST API or Ranger UI:

Type	Name	Description	Transformer
MASK	Redact	Replace lowercase with 'x', uppercase with 'X', digits with '0'	mask({col})
MASK_SHOW_LAST_4	Partial mask: show last 4	Show last 4 characters; replace rest with 'x'	mask_show_last_n({col}, 4, 'x', 'x', 'x', -1, '1')
MASK_SHOW_FIRST_4	Partial mask: show first 4	Show first 4 characters; replace rest with 'x'	mask_show_first_n({col}, 4, 'x', 'x', 'x', -1, '1')

Type	Name	Description	Transformer
MASK_HASH	Hash	Hash the value	mask_hash({col})
MASK_NULL	Nullify	Replace with NULL	N/A
MASK_NONE	Unmasked (retain original value)	No masking	N/A
MASK_DATE_SHOW_YEAR	Date: show only year	Date: show only year	mask({col}, 'x', 'x', 'x', -1, '1', 1, 0, -1)
CUSTOM	Custom	Custom	N/A

The table includes the mask name as it appears in the Ranger UI.

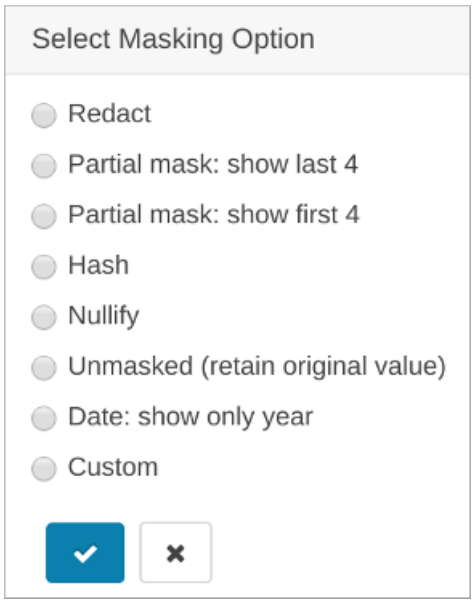

Ranger Column Masking Limitations in Impala

- Column masking introduces unused columns during the query analysis and, consequently, additional SELECT privileges checks on all columns of the masked table.
- Impala might produce more than one audit log entry for a column involved in a column masking policy under all of these conditions: the column appears in multiple places in a query, the query is rewritten, or the query is re-analyzed.
- Column masking policies, shared between Hive and Impala, might be affected by SQL/UDF differences between Hive and Impala, as shown in the following example.

  For instance, UTF-8 strings containing non-ASCII characters are not guaranteed to work properly. Suppose a column masking policy masks the last two characters of a string: s => mask_last_n(s, 2, 'x', 'x', 'x', 'x'). Applying this policy, Hive properly masks SQL## to SQLxx, but the Impala masking produces SQL##xx because each Chinese character is encoded in 3 bytes. Impala and Hive do not handle different lengths in the same way.

Limitations on Mask Functions

The mask functions in Hive are implemented through GenericUDFs. Even though Impala users can call Hive UDFs, Impala does not yet support Hive GenericUDFs, so you cannot use Hive's mask functions in Impala. However, Impala has builtin mask functions that are implemented through overloads. In Impala, when using mask functions, not all parameter combinations are supported. These mask functions are introduced in Impala 3.4

The following list includes all the implemented overloads.

- Overloads used by Ranger default masking policies,
- Overloads with simple arguments,

- Overload with all arguments in int type for full functionality. Char argument needs to be converted to their ASCII value.

To list the available overloads, use the following query:

```
show functions in _impala_builtins like "mask*";
```

Note: An error message that states "No matching function with signature: mask..." implies that Impala does not contain the corresponding overload.

Related Information

GRANT statement
REVOKE statement
Resource-based column masking in Hive with Ranger policies
Tag-based column masking in Hive with Ranger policies
Apache Ranger documentation

Configuring Authorization

Enable Ranger authorization in Impala.

Procedure

1. In Cloudera Manager, navigate to the Impala service.
2. In the Configuration tab, type ranger in the search field.
3. Specify the values in the following fields.
   - Ranger Service: Select the Ranger service you are using. This is the server name to use when granting server level privileges.
   - Ranger Policy Cache Directory: Specify the directory where Ranger security policies are cached locally.
   - Ranger DFS Audit Path: Specify the DFS path on which Ranger audits are written. The special placeholder '${ranger_base_audit_url}' should be used as the prefix, in order to use the centralized location defined in the Ranger service.
   - Ranger Audit DFS Spool Dir: Specify the spool directory for Ranger audits being written to DFS.
   - Ranger Audit Solr Spool Dir: Specify the spool directory for Ranger audits being written to Solr.
4. Restart Impala and Hive.

Row-level filtering in Impala with Ranger policies

Row-level filtering policies are similar to other Ranger access policies. Apache Ranger row-level filtering policy allows to set access policies for rows when reading from a table.

You can use Apache Ranger row-level filtering policies to set access policies for rows when reading from a table. You can set filters for specific users, groups, and conditions. This release adds a new feature flag enable_row_filtering which is set to be true by default. To enable row-filtering feature you must have set the column masking flag enable_column_masking to true since the row-level filtering depends on the column masking implementation. You can use this flag enable_row_filtering to disable this feature as required.

Note: Note that row filtering policies apply prior to any column masking policies, because column masking policies apply only on result data of the target table/view.

The following limitations apply when using row-level filters:

- Row filtering policies on nested tables can't be applied when nested collection columns(e.g. array, map columns) are used directly in the FROM clause. Such queries are currently forbidden.

  This is an example of the currently supported version of a query on the table my_tbl (id int, int_array array<int>) that has a row filter "id = 0".

  ```
  select a.item from my_tbl t, t.int_array a
  ```

  However an equivalent query below is not currently supported since it uses the int_array column non-relatively.

  ```
  select item from my_tbl.int_array
  ```

  Such queries are forbidden and you must rewrite them to the supported format until further notice.
- For information on the steps to set the row-level filtering using Apache Ranger, see the link provided under Related Information.

  Note: Hive features that are not supported by Impala should not be used in the filter since Impala can't evaluate them.

Cloudera Runtime 7.3.1

Apache Impala SQL Reference

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Impala SQL Overview....6
Impala schema objects....6
Impala aliases....6
Impala databases....7
Overview of Impala functions....8
Identifiers....9
Impala tables....10
Impala views....12
Impala SQL data types....13
ARRAY complex type....13
BIGINT data type....17
BINARY data type....17
BOOLEAN data type....19
CHAR data type....21
DATE data type....22
DECIMAL data type....23
DOUBLE data type....28
FLOAT data type....30
INT data type....31
MAP complex type....32
REAL data type....37
SMALLINT data type....37
STRING data type....38
STRUCT complex type....40
TIMESTAMP data type....46
Customizing time zones....51
TINYINT data type....52
VARCHAR data type....53
Complex types....55
Querying arrays....88
Zipping unnest on arrays from views....90
Impala SQL literals....90
Impala SQL operators....94
Impala SQL comments....106
Impala SQL statements....106
ROLE statements....106
DDL statements....107
DML statements....108
ALTER DATABASE statement....109
ALTER TABLE statement....109
ALTER VIEW statement....124
COMMENT statement....125
COMPUTE STATS statement....126
CREATE DATABASE statement....133
CREATE FUNCTION statement....135
CREATE ROLE statement....141
CREATE TABLE statement....142
CREATE VIEW statement....158
DELETE statement....159
DESCRIBE statement....161

DROP DATABASE statement....173
DROP FUNCTION statement....175
DROP ROLE statement....177
DROP STATS statement....177
DROP TABLE statement....182
DROP VIEW statement....183
EXPLAIN statement....184
GRANT statement....187
GRANT ROLE statement....189
INSERT statement....190
INVALIDATE METADATA statement....199
LOAD DATA statement....200
REFRESH statement....204
REFRESH AUTHORIZATION statement....206
REFRESH FUNCTIONS statement....206
REVOKE statement....206
REVOKE ROLE statement....207
SELECT statement....208
Joins in Impala SELECT statements....210
ORDER BY clause....216
GROUP BY clause....221
HAVING clause....225
LIMIT clause....225
OFFSET clause....227
UNION, INTERSECT, and EXCEPT clauses....227
Subqueries in Impala SELECT statements....230
TABLESAMPLE clause....233
WITH clause....239
DISTINCT operator....240
SET statement....240
SHOW statement....241
SHOW ROLES statement....247
SHOW CURRENT ROLES statement....247
SHOW ROLE GRANT GROUP statement....247
SHUTDOWN statement....247
TRUNCATE TABLE statement....249
UPDATE statement....251
UPSERT statement....252
USE statement....253
VALUES statement....254
Optimizer hints in Impala....255
Query options....261
Virtual column....305
Built-in functions....306
Impala mathematical functions....311
Impala bit functions....329
Conversion functions....339
Impala date and time functions....347
Impala conditional functions....366
Impala string functions....371
Impala miscellaneous functions....388
Impala aggregate functions....391
APPX_MEDIAN function....392
AVG function....393
COUNT function....397
GROUPING() and GROUPING_ID() functions....402

GROUP_CONCAT function.....404
MAX function.....405
MIN function.....409
NDV function.....413
STDDEV, STDDEV_SAMP, STDDEV_POP functions.....416
SUM function.....418
VARIANCE, VARIANCE_SAMP, VARIANCE_POP, VAR_SAMP, VAR_POP functions.....422
Impala analytic functions.....424
OVER.....425
WINDOW.....426
AVG.....428
COUNT.....428
CUME_DIST.....428
DENSE_RANK.....430
FIRST_VALUE.....432
LAG.....434
LAST_VALUE.....436
LEAD.....437
MAX.....438
MIN.....438
NTILE.....438
PERCENT_RANK.....439
RANK.....440
ROW_NUMBER.....443
SUM.....444
Creating an Impala user-defined function.....444
UDF concepts.....444
Runtime environment for UDFs.....448
Installing the UDF development package.....448
Writing UDFs.....449
Writing user-defined aggregate functions (UDAFs).....452
Building and deploying UDFs.....453
Performance considerations for UDFs.....455
Examples of creating and using UDFs.....455
Security considerations for UDFs.....462
Limitations and restrictions for Impala UDFs.....462
SQL transactions in Impala.....463
Reserved words.....463
SQL differences between Impala and Hive.....477
Porting SQL from other database systems to Impala.....479
UTF-8 Support.....483

Apache Impala SQL Overview

You can use Apache Impala SQL to manage and access data in Hadoop storage. Review the basic concepts related to Apache Impala SQL such as schema objects you can use to store and manipulate data, supported data types, literals, operators, built-in functions, and so on.

Impala provides a high degree of compatibility with the Hive Query Language. Impala uses the same metadata store as Hive to record information about table structure and properties. Therefore, Impala can access tables defined through the built-in Impala CREATE TABLE command, or tables created using the Hive data definition language (DDL).

Impala schema objects

You must understand what schema objects are, what they include, and where they are stored. The conceptual information related to these objects help you to understand the various ways to specify their names and work with them.

Schema objects are logical structures that you use to store and manipulate data. Schema objects include databases, tables, views, functions, and so on.

Impala aliases

You must be aware of certain guidelines and rules about assigning an alias to the names of tables, columns, or column expressions in SQL statements and using the alias when referencing the table or column in the same statement.

Aliases rules

The following rules apply to aliases:

- You typically specify aliases that are shorter, easier to remember, or both than the original names. The aliases are printed in the query header, making them useful for self-documenting output.

  To set up an alias, add the AS *ALIAS* clause immediately after any table, column, or expression name in the SELE CT list or FROM list of a query. The AS keyword is optional.

  You can specify the alias immediately after the original name.
- You can specify column aliases with or without the AS keyword, and with no quotation marks, single quotation marks, or double quotation marks.
- Aliases are not case sensitive.
- Aliases can be up to the maximum length of a Java string, 2147483647.
- Aliases can include additional characters such as spaces and dashes when they are quoted using backtick characters (``).
- To use an alias name that matches one of the Impala reserved keywords, surround the identifier with either single or double quotation marks, or backtick characters.
- Aliases are allowed at the top level of the GROUP BY, HAVING, and ORDER BY clauses, e.g. GROUP BY *ALIAS*.
- Aliases are not allowed in subexpressions of the GROUP BY, HAVING, and ORDER BY clauses, e,g. GROUP BY *ALIAS/5*.

From Impala 3.0, the alias substitution logic in the GROUP BY, HAVING, and ORDER BY clauses has become more consistent with standard SQL behavior, as follows. Aliases are now only legal at the top level, and not in subexpressions. The following statements are allowed:

```
SELECT int_col / 2 AS x
FROM t
GROUP BY x;

SELECT int_col / 2 AS x
FROM t
ORDER BY x;
SELECT NOT bool_col AS nb
FROM t
GROUP BY nb
HAVING nb;
```

And the following statements are NOT allowed:

```
SELECT int_col / 2 AS x
FROM t
GROUP BY x / 2;

SELECT int_col / 2 AS x
FROM t
ORDER BY -x;

SELECT int_col / 2 AS x
FROM t
GROUP BY x
HAVING x > 3;
```

Note: Impala does not allow column aliases in the WHERE clause.

Related Information

Impala reserved words

Querying arrays

Impala databases

Provides information about how to manage Impala databases that are logical containers for a group of tables.

Each database defines a separate namespace. Creating a database is a lightweight operation. There are minimal database-specific properties to configure, such as LOCATION and COMMENT.

You can change the owner of a database with the ALTER DATABASE statement.

Typically, you create a separate database for each project or application, to avoid naming conflicts between tables and to make clear which tables are related to each other.

The USE statement lets you switch between databases. Unqualified references to tables, views, and functions refer to objects within the current database. You can also refer to objects in other databases by using qualified names of the form *DBNAME.OBJECT_NAME*.

Each database is physically represented by a directory in HDFS. When you do not specify a LOCATION attribute, the directory is located in the Impala data directory with the associated tables managed by Impala. When you do specify a LOCATION attribute, any read and write operations for tables in that database are relative to the specified HDFS directory.

Impala includes two predefined databases:

- default

  By default, you use the default database when you connect to Impala. Tables created in the default database are physically located at one level higher in HDFS than all the user-created databases.
- _impala_builtins

  All Impala built-in functions reside in this database.

Related Information

CREATE DATABASE statement

Overview of Impala functions

Provides information about using Functions to apply arithmetic, string, or other computations and transformations to Impala data.

You can use Functions in SELECT lists and WHERE clauses to filter and format query results so that the result set is exactly what you want, with no further processing needed on the application side.

Scalar functions return a single result for each input row.

```
[localhost:21000] > select name, population from country where continent = '
North America' order by population desc limit 4;
[localhost:21000] > select upper(name), population from country where conti
nent = 'North America' order by population desc limit 4;
+-------------+------------+
| upper(name) | population |
+-------------+------------+
| USA         | 320000000  |
| MEXICO      | 122000000  |
| CANADA      | 25000000   |
| GUATEMALA   | 16000000   |
+-------------+------------+
```

Aggregate functions combine the results from multiple rows: either a single result for the entire table, or a separate result for each group of rows. Aggregate functions are frequently used in combination with GROUP BY and HAVI NG clauses in the SELECT statement.

```
[localhost:21000] > select continent, sum(population) as howmany from cou
ntry group by continent order by howmany desc;
+---------------+------------+
| continent     | howmany    |
+---------------+------------+
| Asia          | 4298723000 |
| Africa        | 1110635000 |
| Europe        | 742452000  |
| North America | 565265000  |
| South America | 406740000  |
| Oceania       | 38304000   |
+---------------+------------+
```

User-defined functions (UDFs) let you code your own logic. They can be either scalar or aggregate functions. UDFs let you implement important business or scientific logic using high-performance code for Impala to automatically parallelize. You can also use UDFs to implement convenience functions to simplify reporting or porting SQL from other database systems.

```
[localhost:21000] > select rot13('Hello world!') as 'Weak obfuscation';
+------------------+
| weak obfuscation |
```

```
+------------------+
| Uryyb jbeyq!     |
+------------------+
[localhost:21000] > select likelihood_of_new_subatomic_particle(sensor1,
 sensor2, sensor3) as probability
                  > from experimental_results group by experiment;
```

Each function is associated with a specific database. For example, if you issue a USE somedb statement followed by CREATE FUNCTION somefunc, the new function is created in the somedb database, and you could refer to it through the fully qualified name somedb.somefunc. You could then issue another USE statement and create a function with the same name in a different database.

Impala built-in functions are associated with a special database named _impala_builtins, which lets you refer to them from any database without qualifying the name.

```
[localhost:21000] > show databases;
+-------------------------+
| name                    |
+-------------------------+
| _impala_builtins        |
| analytic_functions      |
| avro_testing            |
| data_file_size          |
...
[localhost:21000] > show functions in _impala_builtins like '*subs*';
+-------------+-----------------------------------+
| return type | signature                         |
+-------------+-----------------------------------+
| STRING      | substr(STRING, BIGINT)            |
| STRING      | substr(STRING, BIGINT, BIGINT)    |
| STRING      | substring(STRING, BIGINT)         |
| STRING      | substring(STRING, BIGINT, BIGINT) |
+-------------+-----------------------------------+
```

Related Information

Impala built-in functions

Creating an Impala user-defined function

Impala identifiers

Provides information about using Identifiers as the names of databases, tables, or columns when creating the objects.

The following rules apply to identifiers in Impala:

- The minimum length of an identifier is 1 character.
- The maximum length of an identifier is currently 128 characters, enforced by the Metastore database.
- An identifier must start with an alphanumeric or underscore character. The remainder can contain any combination of alphanumeric characters and underscores.

  Quoting the identifier with backticks has no effect on the allowed characters in the name.
- An identifier can contain only ASCII characters.
- To use an identifier name that matches one of the Impala reserved keywords, surround the identifier with `` characters (backticks). Quote the reserved word even if it is part of a fully qualified name.

  The following example shows how a reserved word can be used as a column name if it is quoted with backticks in the CREATE TABLE statement, and how the column name must also be quoted with backticks in a query:

  ```
  [localhost:21000] > create table reserved (`data` string);
  ```

```
[localhost:21000] > select data from reserved;
ERROR: AnalysisException: Syntax error in line 1:
select data from reserved
       ^
Encountered: DATA
Expected: ALL, CASE, CAST, DISTINCT, EXISTS, FALSE, IF, INTERVAL, NOT, NUL
L, STRAIGHT_JOIN, TRUE, IDENTIFIER
CAUSED BY: Exception: Syntax error

[localhost:21000] > select reserved.data from reserved;
ERROR: AnalysisException: Syntax error in line 1:
select reserved.data from reserved
                ^
Encountered: DATA
Expected: IDENTIFIER
CAUSED BY: Exception: Syntax error
[localhost:21000] > select reserved.`data` from reserved;

[localhost:21000] >
```

Important: Because the list of reserved words grows over time as new SQL syntax is added, consider adopting coding conventions (especially for any automated scripts or in packaged applications) to always quote all identifiers with backticks. Quoting all identifiers protects your SQL from compatibility issues if new reserved words are added in later releases.

- Impala identifiers are case-insensitive.

Related Information

Impala reserved words

Impala tables

Provides information about Tables that are the primary containers for data in Impala.

Logically, each table has a structure based on the definition of its columns, partitions, and other properties.

Physically, each table that uses HDFS storage is associated with a directory in HDFS. The table data consists of all the data files underneath that directory:

- Internal tables are managed by Impala, and use directories inside the designated Impala work area.
- External tables use arbitrary HDFS directories, where the data files are typically shared between different Hadoop components.
- Large-scale data is usually handled by partitioned tables, where the data files are divided among different HDFS subdirectories.

Impala queries ignore files with extensions commonly used for temporary work files by Hadoop tools. Any files with extensions .tmp or .copying are not considered part of the Impala table. The suffix matching is case-insensitive, so for example Impala ignores both .copying and .COPYING suffixes.

When you create a table in Impala, you can create an internal table or an external table.

To see whether a table is internal or external, and its associated HDFS location, issue the statement DESCRIBE FORMATTED *TABLE_NAME*. The Table Type field displays MANAGED_TABLE for internal tables and EXTERNAL_TABLE for external tables. The Location field displays the path of the table directory as an HDFS URI.

Internal Tables

The default kind of table produced by the CREATE TABLE statement is known as an internal table.

- Impala creates a directory in HDFS to hold the data files.
- You can create data in internal tables by issuing INSERT or LOAD DATA statements.

- If you add or replace data using HDFS operations, issue the REFRESH command in `impala-shell` so that Impala recognizes the changes in data files, block locations, and so on.
- When you issue a DROP TABLE statement, Impala physically removes all the data files from the directory.
- To see whether a table is internal or external, and its associated HDFS location, issue the statement DESCRIBE FORMATTED *TABLE_NAME*. The Table Type field displays MANAGED_TABLE for internal tables and EXTERNAL_TABLE for external tables. The Location field displays the path of the table directory as an HDFS URI.
- When you issue an ALTER TABLE statement to rename an internal table, all data files are moved into the new HDFS directory for the table. The files are moved even if they were formerly in a directory outside the Impala data directory, for example in an internal table with a LOCATION attribute pointing to an outside HDFS directory.

External Tables

The syntax CREATE EXTERNAL TABLE sets up an Impala table that points at existing data files, potentially in HDFS locations outside the normal Impala data directories.. This operation saves the expense of importing the data into a new table when you already have the data files in a known location in HDFS, in the desired file format.

- You can use Impala to query the data in this table.
- You can create data in external tables by issuing INSERT or LOAD DATA statements.
- If you add or replace data using HDFS operations, issue the REFRESH command in `impala-shell` so that Impala recognizes the changes in data files, block locations, and so on.
- When you issue a DROP TABLE statement in Impala, that removes the connection that Impala has with the associated data files, but does not physically remove the underlying data. You can continue to use the data files with other Hadoop components and HDFS operations.
- When you issue an ALTER TABLE statement to rename an external table, all data files are left in their original locations.
-

You can switch a table between internal to external by using the ALTER TABLE statement with the SET TBLPROPERTIES clause. For example:

```
ALTER TABLE TABLE_NAME SET TBLPROPERTIES('EXTERNAL'='TRUE');
```

If the Kudu service is integrated with the Hive Metastore, the above operations are not supported.

File formats

Each table has an associated file format, which determines how Impala interprets the associated data files.

You set the file format during the CREATE TABLE statement, or change it later using the ALTER TABLE statement.

Partitioned tables can have a different file format for individual partitions, allowing you to change the file format used in your ETL process for new data without going back and reconverting all the existing data in the same table.

Any INSERT statements produce new data files with the current file format of the table.

For existing data files, changing the file format of the table does not automatically do any data conversion. You must use TRUNCATE TABLE or INSERT OVERWRITE to remove any previous data files that use the old file format. Then you use the LOAD DATA statement, INSERT ... SELECT, or other mechanism to put data files of the correct format into the table.

The default file format, Parquet, offers the highest query performance and uses compression to reduce storage requirements; therefore, where practical, use Parquet for Impala tables with substantial amounts of data. Also, the complex types (ARRAY, STRUCT, and MAP) available in Impala 2.3 and higher are currently only supported with the Parquet file type.

Kudu tables

By default, tables stored in Apache Kudu are treated specially, because Kudu manages its data independently of HDFS files.

All metadata that Impala needs is stored in the HMS.

When Kudu is not integrated with the HMS, when you create a Kudu table through Impala, the table is assigned an internal Kudu table name of the form impala::*DB_NAME.TABLE_NAME*. You can see the Kudu-assigned name in the output of DESCRIBE FORMATTED, in the kudu.table_name field of the table properties.

For Impala-Kudu managed tables, ALTER TABLE ... RENAME renames both the Impala and the Kudu table.

For Impala-Kudu external tables, ALTER TABLE ... RENAME renames just the Impala table. To change the Kudu table that an Impala external table points to, use ALTER TABLE *IMPALA_NAME* SET TBLPROPERTIES('kudu.table_name' = '*DIFFERENT_KUDU_TABLE_NAME*'). The underlying Kudu table must already exist.

In practice, external tables are typically used to access underlying Kudu tables that were created outside of Impala, that is, through the Kudu API.

The SHOW TABLE STATS output for a Kudu table shows Kudu-specific details about the layout of the table. Instead of information about the number and sizes of files, the information is divided by the Kudu tablets. For each tablet, the output includes the fields # Rows (although this number is not currently computed), Start Key, Stop Key, Leader Replica, and # Replicas. The output of SHOW COLUMN STATS, illustrating the distribution of values within each column, is the same for Kudu tables as for HDFS-backed tables.

If the Kudu service is not integrated with the Hive Metastore, the distinction between internal and external tables has some special details for Kudu tables. Tables created entirely through Impala are internal tables. The table name as represented within Kudu includes notation such as an impala:: prefix and the Impala database name. External Kudu tables are those created by a non-Impala mechanism, such as a user application calling the Kudu APIs. For these tables, the CREATE EXTERNAL TABLE syntax lets you establish a mapping from Impala to the existing Kudu table:

```
CREATE EXTERNAL TABLE impala_name STORED AS KUDU
  TBLPROPERTIES('kudu.table_name' = 'original_kudu_name');
```

External Kudu tables differ in one important way from other external tables: adding or dropping a column or range partition changes the data in the underlying Kudu table, in contrast to an HDFS-backed external table where existing data files are left untouched.

Related Information

Hadoop file formats supported

Storage systems supported

Impala views

Provides information about using Views as lightweight logical constructs that can act as aliases for queries. Using a view, you can issue complicated reporting queries with compact and simple syntax and reduce maintenance by avoiding the duplication of complicated queries across multiple applications in multiple languages.

You can specify a view name in a query (a SELECT statement or the SELECT portion of an INSERT statement) where you would usually specify a table name.

```
CREATE VIEW v2 AS SELECT t1.c1, t1.c2, t2.c3 FROM t1 JOIN t2 ON (t1.id = t2.
id);
-- This simple query is safer to embed in reporting applications than the l
onger query above.
-- The view definition can remain stable even if the structure of the und
erlying tables changes.
SELECT c1, c2, c3 FROM v2;
```

A view lets you to:

- Build a new, more refined query on top of the original query by adding new clauses, select-list expressions, function calls, and so on: This technique lets you build up several more or less granular variations of the same query, and switch between them when appropriate.
- Set up aliases with intuitive names for tables, columns and result sets from joins.
- Avoid coding lengthy subqueries and repeating the same subquery text in many other queries.

To create, alter, or drop views, use the CREATE VIEW, the ALTER VIEW, and the DROP VIEW statements respectively.

You cannot insert into views.

To see the definition of a view, issue a DESCRIBE FORMATTED statement, which shows the query from the original CREATE VIEW statement

Impala SQL data types

Impala supports a set of data types that you can use for table columns, expression values, and function arguments and return values. You must be aware of the supported data types, the purpose each data type serves, and the casting behaviour for each data type, prior to using them.

Note: Currently, Impala supports only scalar types, not composite or nested types. Accessing a table containing any columns with unsupported types causes an error.

For the notation to write literals of each of these data types, see Literals.

Impala supports a limited set of implicit casts to avoid undesired results from unexpected casting behavior.

- Impala does not implicitly cast between string and numeric or Boolean types. Always use CAST() for these conversions.
- Impala does perform implicit casts among the numeric types, when going from a smaller or less precise type to a larger or more precise one. For example, Impala will implicitly convert a SMALLINT to a BIGINT or FLOAT, but to convert from DOUBLE to FLOAT or INT to TINYINT requires a call to CAST() in the query.
- Impala does perform implicit casts from STRING to TIMESTAMP. Impala has a restricted set of literal formats for the TIMESTAMP data type and the FROM_UNIXTIME() format string; see TIMESTAMP data type on page 46 for details.

See the topics under this section for full details on implicit and explicit casting for each data type, and see Impala type conversion functions on page 339 for details about the CAST() function.

Related Information

Impala SQL literals

SQL differences between Impala and Hive

ARRAY complex type

Review the syntax, usage, and restrictions related to the ARRAY data type which is a complex data type that can represent an arbitrary number of ordered elements. The elements can be scalars or another complex type (ARRAY, STRUCT, or MAP).

Syntax:

```
COLUMN_NAME ARRAY < TYPE >

type ::= PRIMITIVE_TYPE | COMPLEX_TYPE
```

Usage:

Because complex types are often used in combination, for example an ARRAY of STRUCT elements, if you are unfamiliar with the Impala complex types, start with Complex types for background information and usage examples.

The elements of the array have no names. You refer to the value of the array item using the ITEM pseudocolumn, or its position in the array with the POS pseudocolumn.

Each row can have a different number of elements (including none) in the array for that row.

When an array contains items of scalar types, you can use aggregation functions on the array elements without using join notation. For example, you can find the COUNT(), AVG(), SUM(), and so on of numeric array elements, or the MAX() and MIN() of any scalar array elements by referring to *TABLE_NAME.ARRAY_COLUMN* in the FROM clause of the query. When you need to cross-reference values from the array with scalar values from the same row, such as by including a GROUP BY clause to produce a separate aggregated result for each row, then the join clause is required.

A common usage pattern with complex types is to have an array as the top-level type for the column: an array of structs, an array of maps, or an array of arrays. For example, you can model a denormalized table by creating a column that is an ARRAY of STRUCT elements; each item in the array represents a row from a table that would normally be used in a join query. This kind of data structure lets you essentially denormalize tables by associating multiple rows from one table with the matching row in another table.

You typically do not create more than one top-level ARRAY column, because if there is some relationship between the elements of multiple arrays, it is convenient to model the data as an array of another complex type element (either STRUCT or MAP).

You can pass a multi-part qualified name to DESCRIBE to specify an ARRAY, STRUCT, or MAP column and visualize its structure as if it were a table. For example, if table T1 contains an ARRAY column A1, you could issue the statement DESCRIBE t1.a1. If table T1 contained a STRUCT column S1, and a field F1 within the STRUCT was a MAP, you could issue the statement DESCRIBE t1.s1.f1. An ARRAY is shown as a two-column table, with ITEM and POS columns. A STRUCT is shown as a table with each field representing a column in the table. A MAP is shown as a two-column table, with KEY and VALUE columns.

Restrictions:

- Columns with this data type can be used in tables or partitions with the Parquet and ORC file formats.
- Columns with this data type cannot be used as partition key columns in a partitioned table.
- The COMPUTE STATS statement does not produce any statistics for columns of this data type.
- The maximum length of the column definition for any complex type, including declarations for any nested types, is 4000 characters.
- See the *Limitations and restrictions for complex types* topic for a full list of limitations and associated guidelines about complex type columns.

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Examples:

Note: Many of the complex type examples refer to tables such as CUSTOMER and REGION adapted from the tables used in the TPC-H benchmark. See Sample Schema and Data for Experimenting with Impala Complex Types for the table definitions.

The following example shows how to construct a table with various kinds of ARRAY columns, both at the top level and nested within other complex types. Whenever the ARRAY consists of a scalar value, such as in the PETS column or the CHILDREN field, you can see that future expansion is limited. For example, you could not easily evolve the schema to record the kind of pet or the child's birthday alongside the name. Therefore, it is more common to use an ARRAY whose elements are of STRUCT type, to associate multiple fields with each array element.

Note: Practice the CREATE TABLE and query notation for complex type columns using empty tables, until you can visualize a complex data structure and construct corresponding SQL statements reliably.

```
CREATE TABLE array_demo
```

```
(
  id BIGINT,
  name STRING,
-- An ARRAY of scalar type as a top-level column.
  pets ARRAY <STRING>,

-- An ARRAY with elements of complex type (STRUCT).
  places_lived ARRAY < STRUCT <
    place: STRING,
    start_year: INT
  >>,

-- An ARRAY as a field (CHILDREN) within a STRUCT.
-- (The STRUCT is inside another ARRAY, because it is rare
-- for a STRUCT to be a top-level column.)
  marriages ARRAY < STRUCT <
    spouse: STRING,
    children: ARRAY <STRING>
  >>,

-- An ARRAY as the value part of a MAP.
-- The first MAP field (the key) would be a value such as
-- 'Parent' or 'Grandparent', and the corresponding array would
-- represent 2 parents, 4 grandparents, and so on.
  ancestors MAP < STRING, ARRAY <STRING> >
)
STORED AS PARQUET;
```

The following example shows how to examine the structure of a table containing one or more ARRAY columns by using the DESCRIBE statement. You can visualize each ARRAY as its own two-column table, with columns ITEM and POS.

```
DESCRIBE array_demo;
+--------------+---------------------------+
| name         | type                      |
+--------------+---------------------------+
| id           | bigint                    |
| name         | string                    |
| pets         | array<string>             |
| marriages    | array<struct<             |
|              |   spouse:string,          |
|              |   children:array<string>  |
|              | >>                        |
| places_lived | array<struct<             |
|              |   place:string,           |
|              |   start_year:int          |
|              | >>                        |
| ancestors    | map<string,array<string>> |
+--------------+---------------------------+

DESCRIBE array_demo.pets;
+------+--------+
| name | type   |
+------+--------+
| item | string |
| pos  | bigint |
+------+--------+
DESCRIBE array_demo.marriages;
+------+-------------------------+
| name | type                    |
+------+-------------------------+
| item | struct<                 |
```

```
|      |   spouse:string,         |
|      |   children:array<string> |
|      | >                        |
| pos  | bigint                   |
+------+--------------------------+
DESCRIBE array_demo.places_lived;
+------+------------------+
| name | type             |
+------+------------------+
| item | struct<          |
|      |   place:string,  |
|      |   start_year:int |
|      | >                |
| pos  | bigint           |
+------+------------------+

DESCRIBE array_demo.ancestors;
+-------+---------------+
| name  | type          |
+-------+---------------+
| key   | string        |
| value | array<string> |
+-------+---------------+
```

The following example shows queries involving ARRAY columns containing elements of scalar or complex types. You “unpack” each ARRAY column by referring to it in a join query, as if it were a separate table with ITEM and POS columns. If the array element is a scalar type, you refer to its value using the ITEM pseudocolumn. If the array element is a STRUCT, you refer to the STRUCT fields using dot notation and the field names. If the array element is another ARRAY or a MAP, you use another level of join to unpack the nested collection elements.

```
-- Array of scalar values.
-- Each array element represents a single string, plus we know its position
 in the array.
SELECT id, name, pets.pos, pets.item FROM array_demo, array_demo.pets;
-- Array of structs.
-- Now each array element has named fields, possibly of different types.
-- You can consider an ARRAY of STRUCT to represent a table inside another
 table.
SELECT id, name, places_lived.pos, places_lived.item.place, places_lived.it
em.start_year
FROM array_demo, array_demo.places_lived;
-- The .ITEM name is optional for array elements that are structs.
-- The following query is equivalent to the previous one, with .ITEM
-- removed from the column references.
SELECT id, name, places_lived.pos, places_lived.place, places_lived.start_ye
ar
  FROM array_demo, array_demo.places_lived;

-- To filter specific items from the array, do comparisons against the .POS
or .ITEM
-- pseudocolumns, or names of struct fields, in the WHERE clause.
SELECT id, name, pets.item FROM array_demo, array_demo.pets
  WHERE pets.pos in (0, 1, 3);

SELECT id, name, pets.item FROM array_demo, array_demo.pets
  WHERE pets.item LIKE 'Mr. %';

SELECT id, name, places_lived.pos, places_lived.place, places_lived.start_ye
ar
  FROM array_demo, array_demo.places_lived
WHERE places_lived.place like '%California%';
```

Related Information

Complex types

BIGINT data type

An 8-byte integer data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME BIGINT
```

Range: -9223372036854775808 .. 9223372036854775807. There is no UNSIGNED subtype.

Conversions: Impala automatically converts to a floating-point type (FLOAT or DOUBLE) automatically. Use CAST () to convert to TINYINT, SMALLINT, INT, STRING, or TIMESTAMP. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestam p_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Examples:

```
CREATE TABLE t1 (x BIGINT);
SELECT CAST(1000 AS BIGINT);
```

Usage notes:

BIGINT is a convenient type to use for column declarations because you can use any kind of integer values in INSE RT statements and they are promoted to BIGINT where necessary. However, BIGINT also requires the most bytes of any integer type on disk and in memory, meaning your queries are not as efficient and scalable as possible if you overuse this type. Therefore, prefer to use the smallest integer type with sufficient range to hold all input values, and CAST() when necessary to the appropriate type.

For a convenient and automated way to check the bounds of the BIGINT type, call the functions MIN_BIGINT() and MAX_BIGINT().

If an integer value is too large to be represented as a BIGINT, use a DECIMAL instead with sufficient digits of precision.

NULL considerations: Casting any non-numeric value to this type produces a NULL value.

Partitioning: Prefer to use this type for a partition key column. Impala can process the numeric type more efficiently than a STRING representation of the value.

HBase considerations: This data type is fully compatible with HBase tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as an 8-byte value.

Added in: Available in all versions of Impala.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

BINARY data type

This release adds support for BINARY columns for all table formats with the exception of Kudu.

Overview

The BINARY data type is an arbitrary-length byte array data type that can be used in CREATE TABLE and SELECT statements. This data type can contain data like images. The BINARY data type is very similar to the STRING data type except that the BINARY data type is not affected by UTF8_mode.

Syntax

The BINARY data type is defined in the column as shown here:

```
create table binary_table (v varchar, b binary);
```

Length

Take the following considerations for BINARY lengths:

- The hard limit on the size of a BINARY and the total size of a row is 512KB. If a query tries to process or create a BINARY larger than this limit, it will return an error to the user.
- The limit is 1 GB on BINARY when writing to Parquet files.
- Queries operating on BINARYs with 32 KB or less will work reliably and will not cause significant performance or memory problems (unless you have very complex queries, very many columns, etc.)
- Performance and memory consumption may degrade with BINARYs larger than 32 KB.
- The row size, i.e. the total size of all BINARY and other columns, is subject to lower limits at various points in query execution that support spill-to-disk. A few examples for lower row size limits are:
  - Rows coming from the right side of any hash join
  - Rows coming from either side of a hash join that spills to disk
  - Rows being sorted by the SORT operator without a limit
  - Rows in a grouping aggregation

Conversions:The only possible conversion for binary is explicit CAST from/to STRING.

Supported built-in functions

The following list contains some of the supported built-in functions.

- Scalar functions: length, concat, typeof, murmur_hash
- Conditional functions: if, coalesce
- Aggregate functions: min, max, count

Note: If the query option UTF8_MODE is set to FALSE, then the built-ins function the same way for STRING and BINARY.

Restrictions on BINARY columns

- For both table types PARQUET and ORC min/max stat filtering can not be used for BINARY columns, while it can be used for STRING columns.
- Currently, this data type cannot be used with Kudu tables.

Text table considerations:In textfiles Impala writes BINARY columns base64 encoded.

PARQUET considerations:BINARY columns are stored as BYTE_ARRAY in Parquet similar to string. For BINARY columns converted_type/logical type are never set, while they are set for STRINGs if the query option parquet_annotate_strings_utf8 is set to true.

ORC considerations:In ORC file format, there is an exclusive BINARY column type but Impala reads STRING/VARCHAR/CHAR ORC columns also as BINARY.

KUDU considerations:Currently, this data type cannot be used with Kudu tables.

Differences between Impala and Hive

As part of this release, behavior for Impala BINARY type is made to be consistent with Hive in most cases. However, some differences were not resolved to be backward compatible with older releases. You must be aware of these changes to run Hive queries, if needed, without modifying them.

Impala	Hive
Impala doesn't treat STRINGs as UTF8, so BINARY and STRING become nearly identical.	In Hive, the main difference between STRING and BINARY is that STRING is assumed to be UTF8 encoded, while BINARY can be any byte array.
The only possible conversion for binary is explicit CAST from/to STRING.	BINARY can be only implicitly cast from/to STRING
INSERT ... VALUES () string literals need to be explicitly cast to BINARY in Impala.	In Hive, no explici t conversion needed
Impala supports a small subset of built-in STRING functions for the BINARY type. If the query option UTF8_MODE is set to FALSE, then the STRING values behave like BINARY values. So you can store binary values as STRINGs and have all built-in functions available for them.	Only a small subset of built-in STRING functions support BINARY.
In textfiles Impala writes BINARY columns base64 encoded	In several file formats (e.g. text) BINARY is base64 encoded.
No NDV is calculated during COMPUTE STATISTICS.	No NDV is calculated during COMPUTE STATISTICS.
All kinds of UDFs (native, Hive legacy, Hive generic) support BINARY	In Hive UDFs is only supported if the argument and return types are set explicitly

Example

The following example demonstrates calls to string manipulation functions to concatenate binary columns.

```
select concat( cast("a" as binary), cast("b" as binary));
```

The following examples demonstrate how to use the aggregate functions. These functions calculate a return value across all the items in a result set, so they require a FROM clause in the query.

```
select count(a) from product_catalog;
```

```
select max(height), avg(height) from census_data where age > 20;
```

BOOLEAN data type

A data type used in CREATE TABLE and ALTER TABLE statements, representing a single true/false choice.

Syntax: In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME BOOLEAN
```

Range: TRUE or FALSE. Do not use quotation marks around the TRUE and FALSE literal values. You can write the literal values in uppercase, lowercase, or mixed case. The values queried from a table are always returned in lowercase, true or false.

Conversions: Impala does not automatically convert any other type to BOOLEAN. All conversions must use an explicit call to the CAST() function.

You can use CAST() to convert any integer or floating-point type to BOOLEAN: a value of 0 represents false, and any non-zero value is converted to true.

```
SELECT CAST(42 AS BOOLEAN) AS nonzero_int, CAST(99.44 AS BOOLEAN) AS nonzero
_decimal,
  CAST(000 AS BOOLEAN) AS zero_int, CAST(0.0 AS BOOLEAN) AS zero_decimal;
+-------------+-----------------+----------+--------------+
| nonzero_int | nonzero_decimal | zero_int | zero_decimal |
+-------------+-----------------+----------+--------------+
| true        | true            | false    | false        |
+-------------+-----------------+----------+--------------+
```

When you cast the opposite way, from BOOLEAN to a numeric type, the result becomes either 1 or 0:

```
SELECT CAST(true AS INT) AS true_int, CAST(true AS DOUBLE) AS true_double,
  CAST(false AS INT) AS false_int, CAST(false AS DOUBLE) AS false_double;
+----------+-------------+-----------+--------------+
| true_int | true_double | false_int | false_double |
+----------+-------------+-----------+--------------+
| 1        | 1           | 0         | 0            |
+----------+-------------+-----------+--------------+
```

You can cast DECIMAL values to BOOLEAN, with the same treatment of zero and non-zero values as the other numeric types. You cannot cast a BOOLEAN to a DECIMAL.

You cannot cast a STRING value to BOOLEAN, although you can cast a BOOLEAN value to STRING, returning '1' for true values and '0' for false values.

Although you can cast a TIMESTAMP to a BOOLEAN or a BOOLEAN to a TIMESTAMP, the results are unlikely to be useful. Any non-zero TIMESTAMP (that is, any value other than 1970-01-01 00:00:00) becomes TRUE when converted to BOOLEAN, while 1970-01-01 00:00:00 becomes FALSE. A value of FALSE becomes 1970-01-01 00:00:00 when converted to BOOLEAN, and TRUE becomes one second past this epoch date, that is, 1970-01-01 00:00:01.

NULL considerations: An expression of this type produces a NULL value if any argument of the expression is NULL.

Partitioning:

Do not use a BOOLEAN column as a partition key. Although you can create such a table, subsequent operations produce errors:

```
[localhost:21000] > create table truth_table (assertion string) partitioned
by (truth boolean);
[localhost:21000] > insert into truth_table values ('Pigs can fly',false);
ERROR: AnalysisException: INSERT into table with BOOLEAN partition column
 (truth) is not supported: partitioning.truth_table
```

Examples:

```
SELECT 1 < 2;
SELECT 2 = 5;
SELECT 100 < NULL, 100 > NULL;
CREATE TABLE assertions (claim STRING, really BOOLEAN);
INSERT INTO assertions VALUES
  ("1 is less than 2", 1 < 2),
  ("2 is the same as 5", 2 = 5),
  ("Grass is green", true),
  ("The moon is made of green cheese", false);
SELECT claim FROM assertions WHERE really = TRUE;
```

HBase considerations: This data type is fully compatible with HBase tables.

Parquet considerations: This type is fully compatible with Parquet tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Related Information

Impala SQL operators

Impala conditional functions

CHAR data type

A fixed-length character type, padded with trailing spaces if necessary to achieve the specified length. If values are longer than the specified length, Impala truncates any trailing characters.

Performance consideration

Note: For performance consideration, we recommend using VARCHAR or STRING over CHAR as the performance gain of Codegen outweighs the benefits of fixed width CHAR. The CHAR type currently does not have the Impala Codegen support.

Syntax: In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME CHAR(LENGTH)
```

The maximum *LENGTH* you can specify is 255.

Semantics of trailing spaces:

- When you store a CHAR value shorter than the specified length in a table, queries return the value padded with trailing spaces if necessary; the resulting value has the same length as specified in the column definition.
- Leading spaces in CHAR are preserved within the data file.
- If you store a CHAR value containing trailing spaces in a table, those trailing spaces are not stored in the data file. When the value is retrieved by a query, the result could have a different number of trailing spaces. That is, the value includes however many spaces are needed to pad it to the specified length of the column.
- If you compare two CHAR values that differ only in the number of trailing spaces, those values are considered identical.
- When comparing or processing CHAR values:
  - CAST() truncates any longer string to fit within the defined length. For example:

    ```
    SELECT CAST('x' AS CHAR(4)) = CAST('x        ' AS CHAR(4)); -- Returns T
    RUE.
    ```
  - If a CHAR value is shorter than the specified length, it is padded on the right with spaces until it matches the specified length.
  - CHAR_LENGTH() returns the length including any trailing spaces.
  - LENGTH() returns the length excluding trailing spaces.
  - CONCAT() returns the length including trailing spaces.

Partitioning: This type can be used for partition key columns. Because of the efficiency advantage of numeric values over character-based values, if the partition key is a string representation of a number, prefer to use an integer type with sufficient range (INT, BIGINT, and so on) where practical.

HBase considerations: This data type cannot be used with HBase tables.

Parquet considerations:

- This type can be read from and written to Parquet files.
- There is no requirement for a particular level of Parquet.
- Parquet files generated by Impala and containing this type can be freely interchanged with other components such as Hive and MapReduce.
- Any trailing spaces, whether implicitly or explicitly specified, are not written to the Parquet data files.
- Parquet data files might contain values that are longer than allowed by the CHAR(*N*) length limit. Impala ignores any extra trailing characters when it processes those values during a query.

Text table considerations: Text data files might contain values that are longer than allowed for a particular CHAR(*N*) column. Any extra trailing characters are ignored when Impala processes those values during a query. Text data files can also contain values that are shorter than the defined length limit, and Impala pads them with trailing spaces up to the specified length. Any text data files produced by Impala INSERT statements do not include any trailing blanks for CHAR columns.

Avro considerations: The Avro specification allows string values up to 2**64 bytes in length. Impala queries for Avro tables use 32-bit integers to hold string lengths. In Impala 2.5 and higher, Impala truncates CHAR and VARCHAR values in Avro tables to (2**31)-1 bytes. If a query encounters a STRING value longer than (2**31)-1 bytes in an Avro table, the query fails. In earlier releases, encountering such long values in an Avro table could cause a crash.

Compatibility: This type is available using Impala 2.0.0 or higher.

Some other database systems make the length specification optional. For Impala, the length is required.

Internal details: Represented in memory as a byte array with the same size as the length specification. Values that are shorter than the specified length are padded on the right with trailing spaces.

Added in: Impala 2.0.0

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

UDF considerations: This type cannot be used for the argument or return type of a user-defined function (UDF) or user-defined aggregate function (UDA).

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Restrictions: Because the blank-padding behavior requires allocating the maximum length for each value in memory, for scalability reasons, you should avoid declaring CHAR columns that are much longer than typical values in that column.

All data in CHAR and VARCHAR columns must be in a character encoding that is compatible with UTF-8. If you have binary data from another database system (that is, a BLOB type), use a STRING column to hold it.

When an expression compares a CHAR with a STRING or VARCHAR, the CHAR value is implicitly converted to STRING first, with trailing spaces preserved.

This behavior differs from other popular database systems. To get the expected result of TRUE, cast the expressions on both sides to CHAR values of the appropriate length. For example:

```
SELECT CAST("foo  " AS CHAR(5)) = CAST('foo' AS CHAR(3)); -- Returns TRUE.
```

This behavior is subject to change in future releases.

DATE data type

Use the DATE data type to store date values.

Use the DATE data type to store date values. The DATE type is supported for HBase, Text, Avro, and Parquet.

Range:

0001-01-01 to 9999-12-31

Literals and expressions:

The DATE literals are in the form of DATE'YYYY-MM-DD'. For example, DATE '2013-01-01'

Parquet and Avro considerations:

Parquet and Avro use DATE logical type for dates. The DATE logical type annotates an INT32 that stores the number of days from the Unix epoch, January 1, 1970. This representation introduces an interoperability issue between Impala and older versions of Hive:

If Hive versions lower than 3.1 wrote dates earlier than 1582-10-15 to a Parquet or Avro table, those dates will be read back incorrectly by Impala and vice versa. In Hive 3.1 and higher, this is no longer an issue.

Explicit casting between DATE and other data types:

DATE type can only be converted to/from DATE, TIMESTAMP, or STRING types as described below.

Cast from	Cast to	Result
TIMESTAMP	DATE	The date component of the TIMESTAMP is returned, and the time of the day component of the TIMESTAMP is ignored.
STRING	DATE	The DATE value of yyyy-MM-dd is returned. The STRING value must be in the yyyy-MM-dd or yyyy-MM-dd HH:mm:ss.SSSS SSSSS pattern. If the time component is present in STRING, it is silently ignored. If the STRING value does not match the above formats, an error is returned.
DATE	TIMESTAMP	The year, month, and day of the DATE is returned along with the time of day component set to 00:00:00.
DATE	STRING	The STRING value, 'yyyy-MM-dd', is returned.

Implicit casting between DATE and other types:

Implicit casting is supported:

- From STRING to DATE if the source STRING value is in the yyyy-MM-dd or yyyy-MM-dd HH:mm:ss.SSSS SSSSS pattern.
- From DATE to TIMESTAMP.

Added in:

The DATE type is available in Impala 3.3 and higher.

Kudu considerations:

You can read and write DATE values to Kudu tables. For example:

```
create table kudu_date_key (fdatekey date primary key, val string)
stored as kudu
insert into kudu_date_key values (DATE '1970-01-01', 'Unix epoch'), (DATE
 '2019-12-12', 'today')
```

DECIMAL data type

The DECIMAL data type is a numeric data type with fixed scale and precision. The data type is useful for storing and doing operations on precise decimal values.

Syntax:

```
DECIMAL[(PRECISION[, SCALE])]
```

Precision:

PRECISION represents the total number of digits that can be represented regardless of the location of the decimal point.

This value must be between 1 and 38, specified as an integer literal.

The default precision is 9.

Scale:

SCALE represents the number of fractional digits.

This value must be less than or equal to the precision, specified as an integer literal.

The default scale is 0.

When the precision and the scale are omitted, a DECIMAL is treated as DECIMAL(9, 0).

Range:

The range of DECIMAL type is -10^38 +1 through 10^38 –1.

The largest value is represented by DECIMAL(38, 0).

The most precise fractional value (between 0 and 1, or 0 and -1) is represented by DECIMAL(38, 38), with 38 digits to the right of the decimal point. The value closest to 0 would be .0000...1 (37 zeros and the final 1). The value closest to 1 would be .999... (9 repeated 38 times).

Memory and disk storage:

Only the precision determines the storage size for DECIMAL values, and the scale setting has no effect on the storage size. The following table describes the in-memory storage once the values are loaded into memory.

Precision	In-memory storage
1 - 9	4 bytes
10 - 18	8 bytes
19 - 38	10 bytes

The on-disk representation varies depending on the file format of the table.

Text, RCFile, and SequenceFile tables use ASCII-based formats as below:

- Leading zeros are not stored.
- Trailing zeros are stored.
- Each DECIMAL value takes up as many bytes as the precision of the value, plus:
  - One extra byte if the decimal point is present.
  - One extra byte for negative values.

Parquet and Avro tables use binary formats and offer more compact storage for DECIMAL values. In these tables, Impala stores each value in fewer bytes where possible depending on the precision specified for the DECIMAL column. To conserve space in large tables, use the smallest-precision DECIMAL type.

Precision and scale in arithmetic operations:

For all arithmetic operations, the resulting precision is at most 38.

If the resulting precision would be greater than 38, Impala truncates the result from the back, but keeps at least 6 fractional digits in scale and rounds.

For example, DECIMAL(38, 20) * DECIMAL(38, 20) returns DECIMAL(38, 6). According to the table below, the resulting precision and scale would be (77, 40), but they are higher than the maximum precision and scale for DECIMAL. So, Impala sets the precision to the maximum allowed 38, and truncates the scale to 6.

When you use DECIMAL values in arithmetic operations, the precision and scale of the result value are determined as follows. For better readability, the following terms are used in the table below:

- P1, P2: Input precisions
- S1, S2: Input scales
- L1, L2: Leading digits in input DECIMALs, i.e., L1 = P1 - S1 and L2 = P2 - S2

Operation	Resulting precision	Resulting scale
Addition and Subtraction	max (L1, L2) + max (S1, S2) + 1 1 is for carry-over.	max (S1, S2)
Multiplication	P1 + P2 + 1	S1 + S2
Division	L1 + S2 + max (S1 + P2 + 1, 6)	max (S1 + P2 + 1, 6)
Modulo	min (L1, L2) + max (S1, S2)	max (S1, S2)

Precision and scale in functions:

When you use DECIMAL values in built-in functions, the precision and scale of the result value are determined as follows:

- The result of the SUM aggregate function on a DECIMAL value is:
  - Precision: 38
  - Scale: The same scale as the input column
- The result of AVG aggregate function on a DECIMAL value is:
  - Precision: 38
  - Scale: max(Scale of input column, 6)

Implicit conversions in DECIMAL assignments:

Impala enforces strict conversion rules in decimal assignments like in INSERT and UNION statements, or in functions like COALESCE.

If there is not enough precision and scale in the destination, Impala fails with an error.

Impala performs implicit conversions between DECIMAL and other numeric types as below:

- DECIMAL is implicitly converted to DOUBLE or FLOAT when necessary even with a loss of precision. It can be necessary, for example when inserting a DECIMAL value into a DOUBLE column. For example:

  ```
  CREATE TABLE flt(c FLOAT);
  INSERT INTO flt SELECT CAST(1e37 AS DECIMAL(38, 0));
  SELECT CAST(c AS DECIMAL(38, 0)) FROM flt;

  Result: 99999999338158125107115063762579619 84
  ```

  The result has a loss of information due to implicit casting. This is why we discourage using the DOUBLE and FLOAT types in general.
- DOUBLE and FLOAT cannot be implicitly converted to DECIMAL. An error is returned.
- DECIMAL is implicitly converted to DECIMAL if all digits fit in the resulting DECIMAL.

  For example, the following query returns an error because the resulting type that guarantees that all digits fit cannot be determined .

  ```
  SELECT GREATEST (CAST(1 AS DECIMAL(38, 0)), CAST(2 AS DECIMAL(38, 37)));
  ```

- Integer values can be implicitly converted to DECIMAL when there is enough room in the DECIMAL to guarantee that all digits fit. The integer types require the following numbers of digits to the left of the decimal point when converted to DECIMAL:
  - BIGINT: 19 digits
  - INT: 10 digits
  - SMALLINT: 5 digits
  - TINYINT: 3 digits

  For example:

  ```
  CREATE TABLE decimals_10_8 (x DECIMAL(10, 8));
  INSERT INTO decimals_10_8 VALUES (CAST(1 AS TINYINT));
  ```

  The above INSERT statement fails because TINYINT requires room for 3 digits to the left of the decimal point in the DECIMAL.

  ```
  CREATE TABLE decimals_11_8(x DECIMAL(11, 8));
  INSERT INTO decimals_11_8 VALUES (CAST(1 AS TINYINT));
  ```

  The above INSERT statement succeeds because there is enough room for 3 digits to the left of the decimal point that TINYINT requires.

In UNION, the resulting precision and scales are determined as follows.

- Precision: max (L1, L2) + max (S1, S2)

  If the resulting type does not fit in the DECIMAL type, an error is returned. See the first example below.
- Scale: max (S1, S2)

Examples for UNION:

- DECIMAL(20, 0) UNION DECIMAL(20, 20) would require a DECIMAL(40, 20) to fit all the digits. Since this is larger than the max precision for DECIMAL, Impala returns an error. One way to fix the error is to cast both operands to the desired type, for example DECIMAL(38, 18).
- DECIMAL(20, 2) UNION DECIMAL(8, 6) returns DECIMAL(24, 6).
- INT UNION DECIMAL(9, 4) returns DECIMAL(14, 4).

  INT has the precision 10 and the scale 0, so it is treated as DECIMAL(10, 0) UNION DECIMAL(9. 4).

Casting between DECIMAL and other data types:

To avoid potential conversion errors, use CAST to explicitly convert between DECIMAL and other types in decimal assignments like in INSERT and UNION statements, or in functions like COALESCE:

- You can cast the following types to DECIMAL: FLOAT, TINYINT, SMALLINT, INT, BIGINT, STRING
- You can cast DECIMAL to the following types: FLOAT, TINYINT, SMALLINT, INT, BIGINT, STRING, BOOLEAN, TIMESTAMP

Impala performs CAST between DECIMAL and other numeric types as below:

- Precision: If you cast a value with bigger precision than the precision of the destination type, Impala returns an error. For example, CAST(123456 AS DECIMAL(3,0)) returns an error because all digits do not fit into DECI MAL(3, 0)
- Scale: If you cast a value with more fractional digits than the scale of the destination type, the fractional digits are rounded. For example, CAST(1.239 AS DECIMAL(3, 2)) returns 1.24.

Casting STRING to DECIMAL:

You can cast STRING of numeric characters in columns, literals, or expressions to DECIMAL as long as number fits within the specified target DECIMAL type without overflow.

- If scale in STRING > scale in DECIMAL, the fractional digits are rounded to the DECIMAL scale.

  For example, CAST('98.678912' AS DECIMAL(15, 1)) returns 98.7.
- If # leading digits in STRING > # leading digits in DECIMAL, an error is returned.

  For example, CAST('123.45' AS DECIMAL(2, 2)) returns an error.

Exponential notation is supported when casting from STRING.

For example, CAST('1.0e6' AS DECIMAL(32, 0)) returns 1000000.

Casting any non-numeric value, such as 'ABC' to the DECIMAL type returns an error.

Casting DECIMAL to TIMESTAMP:

Casting a DECIMAL value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970).

DECIMAL vs FLOAT consideration:

The FLOAT and DOUBLE types can cause problems or unexpected behavior due to inability to precisely represent certain fractional values, for example dollar and cents values for currency. You might find output values slightly different than you inserted, equality tests that do not match precisely, or unexpected values for GROUP BY columns. The DECIMAL type can help reduce unexpected behavior and rounding errors, but at the expense of some performance overhead for assignments and comparisons.

Literals and expressions:

- Numeric literals without a decimal point
  - The literals are treated as the smallest integer that would fit the literal. For example, 111 is a TINYINT, and 1111 is a SMALLINT.
  - Large literals that do not fit into any integer type are treated as DECIMAL.
  - The literals too large to fit into a DECIMAL(38, 0) are treated as DOUBLE.
- Numeric literals with a decimal point
  - The literal with less than 38 digits are treated as DECIMAL.
  - The literals with 38 or more digits are treated as a DOUBLE.
- Exponential notation is supported in DECIMAL literals.
- To represent a very large or precise DECIMAL value as a literal, for example one that contains more digits than can be represented by a BIGINT literal, use a quoted string or a floating-point value for the number and CAST the string to the desired DECIMAL type.

  For example: CAST('999999999999999999999999999999' AS DECIMAL(38, 5)))

File format considerations:

The DECIMAL data type can be stored in any of the file formats supported by Impala:

- Impala can query Avro, RCFile, or SequenceFile tables that contain DECIMAL columns, created by other Hadoop components.
- Impala can query and insert into Kudu tables that contain DECIMAL columns. Kudu supports the DECIMAL type.
- The DECIMAL data type is fully compatible with HBase tables.
- The DECIMAL data type is fully compatible with Parquet tables.
- Values of the DECIMAL data type are potentially larger in text tables than in tables using Parquet or other binary formats.

UDF consideration:

When writing a C++ UDF, use the DecimalVal data type defined in /usr/include/impala_udf/udf.h.

Changing precision and scale:

You can issue an ALTER TABLE ... REPLACE COLUMNS statement to change the precision and scale of an existing DECIMAL column.

- For text-based formats (text, RCFile, and SequenceFile tables)
  - If the values in the column fit within the new precision and scale, they are returned correctly by a query.
  - If any values that do not fit within the new precision and scale:
    - Impala returns an error if the query option ABORT_ON_ERROR is set to true.
    - Impala returns a NULL and warning that conversion failed if the query option ABORT_ON_ERROR is set to false.
  - Leading zeros do not count against the precision value, but trailing zeros after the decimal point do.
- For binary formats (Parquet and Avro tables)
  - Although an ALTER TABLE ... REPLACE COLUMNS statement that changes the precision or scale of a DECIMAL column succeeds, any subsequent attempt to query the changed column results in a fatal error. This is because the metadata about the columns is stored in the data files themselves, and ALTER TABLE does not actually make any updates to the data files. The other unaltered columns can still be queried successfully.
  - If the metadata in the data files disagrees with the metadata in the metastore database, Impala cancels the query.

Partitioning:

Using a DECIMAL column as a partition key provides you a better match between the partition key values and the HDFS directory names than using a DOUBLE or FLOAT partitioning column.

Column statistics considerations:

Because the DECIMAL type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Compatibility with older version of DECIMAL:

This version of DECIMAL type is the default in Impala 3.0 and higher. In Impala 2.12 and lower, decimal_v2=false. And in Impala 3.0 or higher, decimal_v2=true. The key differences between this version of DECIMAL and the previous DECIMAL V1 in Impala 2.x include the following:

Property	Impala 3.0 or higher	Impala 2.12 or lower
Overall behavior	Returns either the result or an error.	Returns either the result or NULL with a warning.
Overflow behavior	Aborts with an error.	Issues a warning and returns NULL.
Truncation / rounding behavior in arithmetic	Truncates and rounds digits from the back.	Truncates digits from the front.
String cast	Truncates from the back and rounds.	Truncates from the back.

If you need to continue using the first version of the DECIMAL type for the backward compatibility of your queries, set the DECIMAL_V2 query option to FALSE:

```
SET DECIMAL_V2=FALSE;
```

Compatibility with other databases:

Use the DECIMAL data type in Impala for applications where you used the NUMBER data type in Oracle.

The Impala DECIMAL type does not support the Oracle idioms of * for scale.

The Impala DECIMAL type does not support negative values for precision.

DOUBLE data type

A double precision floating-point data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME DOUBLE
```

Range: 4.94065645841246544e-324d .. 1.79769313486231570e+308, positive or negative

Precision: 15 to 17 significant digits, depending on usage. The number of significant digits does not depend on the position of the decimal point.

Representation: The values are stored in 8 bytes, using the IEEE 754 Double Precision Binary Floating Point format.

Conversions: Impala does not automatically convert DOUBLE to any other type. You can use CAST() to convert DOUBLE values to FLOAT, TINYINT, SMALLINT, INT, BIGINT, STRING, TIMESTAMP, or BOOLEAN. You can use exponential notation in DOUBLE literals or when casting from STRING, for example 1.0e6 to represent one million. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Usage notes:

The data type REAL is an alias for DOUBLE.

Impala does not evaluate NaN (not a number) as equal to any other numeric values, including other NaN values. For example, the following statement, which evaluates equality between two NaN values, returns false:

```
SELECT CAST('nan' AS DOUBLE)=CAST('nan' AS DOUBLE);
```

Examples:

```
CREATE TABLE t1 (x DOUBLE);
SELECT CAST(1000.5 AS DOUBLE);
```

Partitioning: Because fractional values of this type are not always represented precisely, when this type is used for a partition key column, the underlying HDFS directories might not be named exactly as you expect. Prefer to partition on a DECIMAL column instead.

HBase considerations: This data type is fully compatible with HBase tables.

Parquet considerations: This type is fully compatible with Parquet tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as an 8-byte value.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Restrictions:

Due to the way arithmetic on FLOAT and DOUBLE columns uses high-performance hardware instructions, and distributed queries can perform these operations in different order for each query, results can vary slightly for aggregate function calls such as SUM() and AVG() for FLOAT and DOUBLE columns, particularly on large data sets where millions or billions of values are summed or averaged. For perfect consistency and repeatability, use the DECIMAL data type for such operations instead of FLOAT or DOUBLE.

The inability to exactly represent certain floating-point values means that DECIMAL is sometimes a better choice than DOUBLE or FLOAT when precision is critical, particularly when transferring data from other database systems that use different representations or file formats.

Kudu considerations:

Currently, the data types BOOLEAN, FLOAT, and DOUBLE cannot be used for primary key columns in Kudu tables.

Related Information

Impala mathematical functions
FLOAT data type
Impala SQL literals

FLOAT data type

A single precision floating-point data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME FLOAT
```

Range: 1.40129846432481707e-45 .. 3.40282346638528860e+38, positive or negative

Precision: 6 to 9 significant digits, depending on usage. The number of significant digits does not depend on the position of the decimal point.

Representation: The values are stored in 4 bytes, using the *IEEE 754 Single Precision Binary Floating Point* format.

Conversions: Impala automatically converts FLOAT to more precise DOUBLE values, but not the other way around. You can use CAST() to convert FLOAT values to TINYINT, SMALLINT, INT, BIGINT, STRING, TIMESTAMP, or BOOLEAN. You can use exponential notation in FLOAT literals or when casting from STRING, for example 1.0e6 to represent one million. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Usage notes:

Impala does not evaluate NaN (not a number) as equal to any other numeric values, including other NaN values. For example, the following statement, which evaluates equality between two NaN values, returns false:

```
SELECT CAST('nan' AS FLOAT)=CAST('nan' AS FLOAT);
```

Examples:

```
CREATE TABLE t1 (x FLOAT);
SELECT CAST(1000.5 AS FLOAT);
```

Partitioning: Because fractional values of this type are not always represented precisely, when this type is used for a partition key column, the underlying HDFS directories might not be named exactly as you expect. Prefer to partition on a DECIMAL column instead.

HBase considerations: This data type is fully compatible with HBase tables.

Parquet considerations: This type is fully compatible with Parquet tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as a 4-byte value.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Restrictions:

Due to the way arithmetic on FLOAT and DOUBLE columns uses high-performance hardware instructions, and distributed queries can perform these operations in different order for each query, results can vary slightly for aggregate function calls such as SUM() and AVG() for FLOAT and DOUBLE columns, particularly on large data sets where millions or billions of values are summed or averaged. For perfect consistency and repeatability, use the DECIMAL data type for such operations instead of FLOAT or DOUBLE.

The inability to exactly represent certain floating-point values means that DECIMAL is sometimes a better choice than DOUBLE or FLOAT when precision is critical, particularly when transferring data from other database systems that use different representations or file formats.

Kudu considerations:

Currently, the data types BOOLEAN, FLOAT, and DOUBLE cannot be used for primary key columns in Kudu tables.

Related Information

IEEE 754 Single Precision Binary Floating Point

Impala mathematical functions

Impala SQL literals

INT data type

A 4-byte integer data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME INT
```

Range: -2147483648 .. 2147483647. There is no UNSIGNED subtype.

Conversions: Impala automatically converts to a larger integer type (BIGINT) or a floating-point type (FLOAT or DOUBLE) automatically. Use CAST() to convert to TINYINT, SMALLINT, STRING, or TIMESTAMP. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Usage notes:

The data type INTEGER is an alias for INT.

For a convenient and automated way to check the bounds of the INT type, call the functions MIN_INT() and MAX_INT().

If an integer value is too large to be represented as a INT, use a BIGINT instead.

NULL considerations: Casting any non-numeric value to this type produces a NULL value.

Examples:

```
CREATE TABLE t1 (x INT);
SELECT CAST(1000 AS INT);
```

Partitioning: Prefer to use this type for a partition key column. Impala can process the numeric type more efficiently than a STRING representation of the value.

HBase considerations: This data type is fully compatible with HBase tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as a 4-byte value.

Added in: Available in all versions of Impala.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

MAP complex type

A complex data type representing an arbitrary set of key-value pairs. The key part is a scalar type, while the value part can be a scalar or another complex type (ARRAY, STRUCT, or MAP).

Syntax:

```
COLUMN_NAME MAP < PRIMITIVE_TYPE, TYPE >

type ::= PRIMITIVE_TYPE | COMPLEX_TYPE
```

Usage notes:

Because complex types are often used in combination, for example an ARRAY of STRUCT elements, if you are unfamiliar with the Impala complex types, start with Complex types for background information and usage examples.

The MAP complex data type represents a set of key-value pairs. Each element of the map is indexed by a primitive type such as BIGINT or STRING, letting you define sequences that are not continuous or categories with arbitrary names. You might find it convenient for modelling data produced in other languages, such as a Python dictionary or Java HashMap, where a single scalar value serves as the lookup key.

In a big data context, the keys in a map column might represent a numeric sequence of events during a manufacturing process, or TIMESTAMP values corresponding to sensor observations. The map itself is inherently unordered, so you choose whether to make the key values significant (such as a recorded TIMESTAMP) or synthetic (such as a random global universal ID).

Note: Behind the scenes, the MAP type is implemented in a similar way as the ARRAY type. Impala does not enforce any uniqueness constraint on the KEY values, and the KEY values are processed by looping through the elements of the MAP rather than by a constant-time lookup. Therefore, this type is primarily for ease of understanding when importing data and algorithms from non-SQL contexts, rather than optimizing the performance of key lookups.

You can pass a multi-part qualified name to DESCRIBE to specify an ARRAY, STRUCT, or MAP column and visualize its structure as if it were a table. For example, if table T1 contains an ARRAY column A1, you could issue the statement DESCRIBE t1.a1. If table T1 contained a STRUCT column S1, and a field F1 within the STRUCT was a MAP, you could issue the statement DESCRIBE t1.s1.f1. An ARRAY is shown as a two-column table, with ITEM and POS columns. A STRUCT is shown as a table with each field representing a column in the table. A MAP is shown as a two-column table, with KEY and VALUE columns.

Added in: Impala 2.3.0

Restrictions:

- Columns with this data type can be used in tables or partitions with the Parquet and ORC file formats.
- Columns with this data type cannot be used as partition key columns in a partitioned table.
- The COMPUTE STATS statement does not produce any statistics for columns of this data type.
- The maximum length of the column definition for any complex type, including declarations for any nested types, is 4000 characters.
- See the *Limitations and restrictions for complex types* topic for a full list of limitations and associated guidelines about complex type columns.

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Examples:

Note: Many of the complex type examples refer to tables such as CUSTOMER and REGION adapted from the tables used in the TPC-H benchmark.

The following example shows a table with various kinds of MAP columns, both at the top level and nested within other complex types. Each row represents information about a specific country, with complex type fields of various levels of nesting to represent different information associated with the country: factual measurements such as area and population, notable people in different categories, geographic features such as cities, points of interest within each city, and mountains with associated facts. Practice the CREATE TABLE and query notation for complex type columns using empty tables, until you can visualize a complex data structure and construct corresponding SQL statements reliably.

```
create TABLE map_demo
(
  country_id BIGINT,

-- Numeric facts about each country, looked up by name.
-- For example, 'Area':1000, 'Population':999999.
-- Using a MAP instead of a STRUCT because there could be
-- a different set of facts for each country.
  metrics MAP <STRING, BIGINT>,
-- MAP whose value part is an ARRAY.
-- For example, the key 'Famous Politicians' could represent an array of 10
 elements,
-- while the key 'Famous Actors' could represent an array of 20 elements.
  notables MAP <STRING, ARRAY <STRING>>,

-- MAP that is a field within a STRUCT.
-- (The STRUCT is inside another ARRAY, because it is rare
-- for a STRUCT to be a top-level column.)
-- For example, city #1 might have points of interest with key 'Zoo',
-- representing an array of 3 different zoos.
-- City #2 might have completely different kinds of points of interest.
-- Because the set of field names is potentially large, and most entries c
ould be blank,
-- a MAP makes more sense than a STRUCT to represent such a sparse data s
tructure.
  cities ARRAY < STRUCT <
    name: STRING,
    points_of_interest: MAP <STRING, ARRAY <STRING>>
  >>,

-- MAP that is an element within an ARRAY. The MAP is inside a STRUCT field
 to associate
-- the mountain name with all the facts about the mountain.
-- The "key" of the map (the first STRING field) represents the name of s
ome fact whose value
-- can be expressed as an integer, such as 'Height', 'Year First Climbed',
and so on.
  mountains ARRAY < STRUCT < name: STRING, facts: MAP <STRING, INT > > >
)
STORED AS PARQUET;
```

```
DESCRIBE map_demo;
+------------+-------------------------------------------------+
| name       | type                                            |
```

```
+------------+-----------------------------------------------+
| country_id | bigint                                        |
| metrics    | map<string,bigint>                            |
| notables   | map<string,array<string>>                     |
| cities     | array<struct<                                 |
|            |   name:string,                                |
|            |   points_of_interest:map<string,array<string>> |
|            | >>                                            |
| mountains  | array<struct<                                 |
|            |   name:string,                                |
|            |   facts:map<string,int>                       |
|            | >>                                            |
+------------+-----------------------------------------------+

DESCRIBE map_demo.metrics;
+-------+--------+
| name  | type   |
+-------+--------+
| key   | string |
| value | bigint |
+-------+--------+

DESCRIBE map_demo.notables;
+-------+---------------+
| name  | type          |
+-------+---------------+
| key   | string        |
| value | array<string> |
+-------+---------------+

DESCRIBE map_demo.notables.value;
+------+--------+
| name | type   |
+------+--------+
| item | string |
| pos  | bigint |
+------+--------+

DESCRIBE map_demo.cities;
+------+-----------------------------------------------+
| name | type                                          |
+------+-----------------------------------------------+
| item | struct<                                       |
|      |   name:string,                                |
|      |   points_of_interest:map<string,array<string>> |
|      | >                                             |
| pos  | bigint                                        |
+------+-----------------------------------------------+

DESCRIBE map_demo.cities.item.points_of_interest;
+-------+---------------+
| name  | type          |
+-------+---------------+
| key   | string        |
| value | array<string> |
+-------+---------------+

DESCRIBE map_demo.cities.item.points_of_interest.value;
+------+--------+
| name | type   |
+------+--------+
| item | string |
| pos  | bigint |
+------+--------+
```

```
DESCRIBE map_demo.mountains;
+------+-------------------------+
| name | type                    |
+------+-------------------------+
| item | struct<                 |
|      |   name:string,          |
|      |   facts:map<string,int> |
|      | >                       |
| pos  | bigint                  |
+------+-------------------------+

DESCRIBE map_demo.mountains.item.facts;
+-------+--------+
| name  | type   |
+-------+--------+
| key   | string |
| value | int    |
+-------+--------+
```

The following example shows a table that uses a variety of data types for the MAP “key” field. Typically, you use BIGINT or STRING to use numeric or character-based keys without worrying about exceeding any size or length constraints.

```
CREATE TABLE map_demo_obscure
(
  id BIGINT,
  m1 MAP <INT, INT>,
  m2 MAP <SMALLINT, INT>,
  m3 MAP <TINYINT, INT>,
  m4 MAP <TIMESTAMP, INT>,
  m5 MAP <BOOLEAN, INT>,
  m6 MAP <CHAR(5), INT>,
  m7 MAP <VARCHAR(25), INT>,
  m8 MAP <FLOAT, INT>,
  m9 MAP <DOUBLE, INT>,
  m10 MAP <DECIMAL(12,2), INT>
)
STORED AS PARQUET;
```

```
CREATE TABLE celebrities (name STRING, birth_year MAP < STRING, SMALLINT >)
STORED AS PARQUET;
-- A typical row might represent values with 2 different birth years, such
 as:
-- ("Joe Movie Star", { "real": 1972, "claimed": 1977 })

CREATE TABLE countries (name STRING, famous_leaders MAP < INT, STRING >) STO
RED AS PARQUET;
-- A typical row might represent values with different leaders, with key val
ues corresponding to their numeric sequence, such as:
-- ("United States", { 1: "George Washington", 3: "Thomas Jefferson", 16:
"Abraham Lincoln" })

CREATE TABLE airlines (name STRING, special_meals MAP < STRING, MAP < STRI
NG, STRING > >) STORED AS PARQUET;
-- A typical row might represent values with multiple kinds of meals, each
 with several components:
-- ("Elegant Airlines",
--   {
```

```
--     "vegetarian": { "breakfast": "pancakes", "snack": "cookies", "dinne
r": "rice pilaf" },
--     "gluten free": { "breakfast": "oatmeal", "snack": "fruit", "dinner":
"chicken" }
--   } )
```

Querying MAPs with SELECT statement

You can use a SELECT statement to run queries on the keys and values of maps. However, you cannot have mixed complex types in the select list such as collections (arrays or maps) in structs or structs in collections. Also sorting is not supported if the select list contains collection columns.

When you query the MAP data type using a SELECT statement, it returns a JSON string. MAP keys can be non-string types but JSON objects can only have string keys. Therefore when a SELECT statement returns a MAP with non-string (for example integer) keys, the resulting JSON is technically invalid.

For example, in Impala the following two maps are not the same:

{1: "a", 2: "b"}

{"1": "a", "2": "b"}

The first map has INT keys, the second has STRING keys. Only the second one is valid json.

Examples of using MAPs in the SELECT list

```
---- QUERY
select id, int_map from complextypestbl
---- RESULTS
1,'{"k1":1,"k2":100}'
2,'{"k1":2,"k2":NULL}'
3,'{}'
4,'{}'
5,'{}'
6,'NULL'
7,'{"k1":NULL,"k3":NULL}'
8,'{"k1":-1}'
---- TYPES
bigint,string
```

```
---- QUERY
select id, int_map from complextypestbl where id=1
---- RESULTS
1,'{"k1":1,"k2":100}'
---- TYPES
bigint,string
```

```
---- QUERY
select id, int_map, int_map_array from complextypestbl
---- RESULTS
1,'{"k1":1,"k2":100}','[{"k1":1}]'
2,'{"k1":2,"k2":NULL}','[{"k3":NULL,"k1":1},NULL,{}]'
3,'{}','[NULL,NULL]'
4,'{}','[]'
5,'{}','NULL'
6,'NULL','NULL'
7,'{"k1":NULL,"k3":NULL}','NULL'
8,'{"k1":-1}','[{},{"k1":1},{},{}]'
---- TYPES
bigint,string,string
```

```
---- QUERY
```

```
select id, int_map from complextypestbl union all select id, int_map from
 complextypestbl
---- RESULTS
1,'{"k1":1,"k2":100}'
2,'{"k1":2,"k2":NULL}'
3,'{}'
4,'{}'
5,'{}'
6,'NULL'
7,'{"k1":NULL,"k3":NULL}'
8,'{"k1":-1}'
1,'{"k1":1,"k2":100}'
2,'{"k1":2,"k2":NULL}'
3,'{}'
4,'{}'
5,'{}'
6,'NULL'
7,'{"k1":NULL,"k3":NULL}'
8,'{"k1":-1}'
---- TYPES
bigint,string
```

REAL data type

An alias for the DOUBLE data type.

Examples:

These examples show how you can use the type names REAL and DOUBLE interchangeably, and behind the scenes Impala treats them always as DOUBLE.

```
[localhost:21000] > create table r1 (x real);
[localhost:21000] > describe r1;
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| x    | double |         |
+------+--------+---------+
[localhost:21000] > insert into r1 values (1.5), (cast (2.2 as double));
[localhost:21000] > select cast (1e6 as real);
+---------------------------+
| cast(1000000.0 as double) |
+---------------------------+
| 1000000                   |
+---------------------------+
```

Related Information

DOUBLE data type

SMALLINT data type

A 2-byte integer data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME SMALLINT
```

Range: -32768 .. 32767. There is no UNSIGNED subtype.

Conversions: Impala automatically converts to a larger integer type (INT or BIGINT) or a floating-point type (FLOAT or DOUBLE) automatically. Use CAST() to convert to TINYINT, STRING, or TIMESTAMP. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Usage notes:

For a convenient and automated way to check the bounds of the SMALLINT type, call the functions MIN_SMALLINT() and MAX_SMALLINT().

If an integer value is too large to be represented as a SMALLINT, use an INT instead.

NULL considerations: Casting any non-numeric value to this type produces a NULL value.

Examples:

```
CREATE TABLE t1 (x SMALLINT);
SELECT CAST(1000 AS SMALLINT);
```

Parquet considerations:

Physically, Parquet files represent TINYINT and SMALLINT values as 32-bit integers. Although Impala rejects attempts to insert out-of-range values into such columns, if you create a new table with the CREATE TABLE ... LIKE PARQUET syntax, any TINYINT or SMALLINT columns in the original table turn into INT columns in the new table.

Partitioning: Prefer to use this type for a partition key column. Impala can process the numeric type more efficiently than a STRING representation of the value.

HBase considerations: This data type is fully compatible with HBase tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as a 2-byte value.

Added in: Available in all versions of Impala.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

STRING data type

A data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE and ALTER TABLE statements:

```
COLUMN_NAME STRING
```

Length:

If you need to manipulate string values with precise or maximum lengths, in Impala 2.0 and higher you can declare columns as VARCHAR(*MAX_LENGTH*) or CHAR(*LENGTH*), but for best performance use STRING where practical.

Take the following considerations for STRING lengths:

- The hard limit on the size of a STRING and the total size of a row is 2 GB.

  If a query tries to process or create a string larger than this limit, it will return an error to the user.

- The limit is 1 GB on STRING when writing to Parquet files.
- Queries operating on strings with 32 KB or less will work reliably and will not cause significant performance or memory problems (unless you have very complex queries, very many columns, etc.)
- Performance and memory consumption may degrade with strings larger than 32 KB.
- The row size, i.e. the total size of all string and other columns, is subject to lower limits at various points in query execution that support spill-to-disk. A few examples for lower row size limits are:
  - Rows coming from the right side of any hash join
  - Rows coming from either side of a hash join that spills to disk
  - Rows being sorted by the SORT operator without a limit
  - Rows in a grouping aggregation

  The max row size is configurable on a per-query basis with the MAX_ROW_SIZE query option. Rows up to MAX_ROW_SIZE (which defaults to 512 KB) can always be processed in the above cases. Rows larger than MAX_ROW_SIZE are processed on a best-effort basis.

Character sets:

For full support in all Impala subsystems, restrict string values to the ASCII character set. Although some UTF-8 character data can be stored in Impala and retrieved through queries, UTF-8 strings containing non-ASCII characters are not guaranteed to work properly in combination with many SQL aspects, including but not limited to:

- String manipulation functions.
- Comparison operators.
- The ORDER BY clause.
- Values in partition key columns.

For any national language aspects such as collation order or interpreting extended ASCII variants such as ISO-8859-1 or ISO-8859-2 encodings, Impala does not include such metadata with the table definition. If you need to sort, manipulate, or display data depending on those national language characteristics of string data, use logic on the application side.

Conversions:

- Impala does not automatically convert STRING to any numeric type. Impala does automatically convert STRING to TIMESTAMP if the value matches one of the accepted TIMESTAMP formats.
- You can use CAST() to convert STRING values to TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, or TIMESTAMP.
- You cannot directly cast a STRING value to BOOLEAN. You can use a CASE expression to evaluate string values such as 'T', 'true', and so on and return Boolean true and false values as appropriate.
- You can cast a BOOLEAN value to STRING, returning '1' for true values and '0' for false values.

Partitioning:

Although it might be convenient to use STRING columns for partition keys, even when those columns contain numbers, for performance and scalability it is much better to use numeric columns as partition keys whenever practical. Although the underlying HDFS directory name might be the same in either case, the in-memory storage for the partition key columns is more compact, and computations are faster, if partition key columns such as YEAR, MONTH, DAY and so on are declared as INT, SMALLINT, and so on.

Zero-length strings: For purposes of clauses such as DISTINCT and GROUP BY, Impala considers zero-length strings (""), NULL, and space to all be different values.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Avro considerations: The Avro specification allows string values up to 2**64 bytes in length. Impala queries for Avro tables use 32-bit integers to hold string lengths. In Impala 2.5 and higher, Impala truncates CHAR and VARCHAR values in Avro tables to (2**31)-1 bytes. If a query encounters a STRING value longer than (2**31)-1 bytes in an Avro table, the query fails. In earlier releases, encountering such long values in an Avro table could cause a crash.

Column statistics considerations: Because the values of this type have variable size, none of the column statistics fields are filled in until you run the COMPUTE STATS statement.

Examples:

The following examples demonstrate double-quoted and single-quoted string literals, and required escaping for quotation marks within string literals:

```
SELECT 'I am a single-quoted string';
SELECT "I am a double-quoted string";
SELECT 'I\'m a single-quoted string with an apostrophe';
SELECT "I\'m a double-quoted string with an apostrophe";
SELECT 'I am a "short" single-quoted string containing quotes';
SELECT "I am a \"short\" double-quoted string containing quotes";
```

The following examples demonstrate calls to string manipulation functions to concatenate strings, convert numbers to strings, or pull out substrings:

```
SELECT CONCAT("Once upon a time, there were ", CAST(3 AS STRING), ' little p
igs.');
SELECT SUBSTR("hello world",7,5);
```

The following examples show how to perform operations on STRING columns within a table:

```
CREATE TABLE t1 (s1 STRING, s2 STRING);
INSERT INTO t1 VALUES ("hello", 'world'), (CAST(7 AS STRING), "wonders");
SELECT s1, s2, length(s1) FROM t1 WHERE s2 LIKE 'w%';
```

Related Information

MAX_ROW_SIZE query option

TIMESTAMP data type

STRUCT complex type

A complex data type, representing multiple fields of a single item. Frequently used as the element type of an ARRAY or the VALUE part of a MAP.

Syntax:

```
COLUMN_NAME STRUCT < NAME : TYPE [COMMENT 'COMMENT_STRING'], ... >

type ::= PRIMITIVE_TYPE | COMPLEX_TYPE
```

The names and number of fields within the STRUCT are fixed. Each field can be a different type. A field within a STRUCT can also be another STRUCT, or an ARRAY or a MAP, allowing you to create nested data structures with a maximum nesting depth of 100.

A STRUCT can be the top-level type for a column, or can itself be an item within an ARRAY or the value part of the key-value pair in a MAP.

When a STRUCT is used as an ARRAY element or a MAP value, you use a join clause to bring the ARRAY or MAP elements into the result set, and then refer to *ARRAY_NAME*.ITEM.*FIELD* or *MAP_NAME*.VALUE.*FIELD*. In the case of a STRUCT directly inside an ARRAY or MAP, you can omit the .ITEM and .VALUE pseudocolumns and refer directly to *ARRAY_NAME.FIELD* or *MAP_NAME.FIELD*.

Usage notes:

Because complex types are often used in combination, for example an ARRAY of STRUCT elements, if you are unfamiliar with the Impala complex types, start with Complex types for background information and usage examples.

A STRUCT is similar conceptually to a table row: it contains a fixed number of named fields, each with a predefined type. To combine two related tables, while using complex types to minimize repetition, the typical way to represent that data is as an ARRAY of STRUCT elements.

Because a STRUCT has a fixed number of named fields, it typically does not make sense to have a STRUCT as the type of a table column. In such a case, you could just make each field of the STRUCT into a separate column of the table. The STRUCT type is most useful as an item of an ARRAY or the value part of the key-value pair in a MAP. A nested type column with a STRUCT at the lowest level lets you associate a variable number of row-like objects with each row of the table.

The STRUCT type is straightforward to reference within a query. You do not need to include the STRUCT column in a join clause or give it a table alias, as is required for the ARRAY and MAP types. You refer to the individual fields using dot notation, such as *STRUCT_COLUMN_NAME.FIELD_NAME*, without any pseudocolumn such as ITEM or VALUE.

You can pass a multi-part qualified name to DESCRIBE to specify an ARRAY, STRUCT, or MAP column and visualize its structure as if it were a table. For example, if table T1 contains an ARRAY column A1, you could issue the statement DESCRIBE t1.a1. If table T1 contained a STRUCT column S1, and a field F1 within the STRUCT was a MAP, you could issue the statement DESCRIBE t1.s1.f1. An ARRAY is shown as a two-column table, with ITEM and POS columns. A STRUCT is shown as a table with each field representing a column in the table. A MAP is shown as a two-column table, with KEY and VALUE columns.

Internal details:

Within the Parquet data file, the values for each STRUCT field are stored adjacent to each other, so that they can be encoded and compressed using all the Parquet techniques for storing sets of similar or repeated values. The adjacency applies even when the STRUCT values are part of an ARRAY or MAP. During a query, Impala avoids unnecessary I/O by reading only the portions of the Parquet data file containing the requested STRUCT fields.

Added in: Impala 2.3.0

Restrictions:

- Columns with this data type can be used in tables or partitions with the Parquet and ORC file formats.
- Columns with this data type cannot be used as partition key columns in a partitioned table.
- The COMPUTE STATS statement does not produce any statistics for columns of this data type.
- The maximum length of the column definition for any complex type, including declarations for any nested types, is 4000 characters.
- See the *Limitations and restrictions for complex types* topic for a full list of limitations and associated guidelines about complex type columns.

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Examples:

Note: Many of the complex type examples refer to tables such as CUSTOMER and REGION adapted from the tables used in the TPC-H benchmark.

The following example shows a table with various kinds of STRUCT columns, both at the top level and nested within other complex types. Practice the CREATE TABLE and query notation for complex type columns using empty tables, until you can visualize a complex data structure and construct corresponding SQL statements reliably.

```
CREATE TABLE struct_demo
(
  id BIGINT,
  name STRING,

-- A STRUCT as a top-level column. Demonstrates how the table ID column
-- and the ID field within the STRUCT can coexist without a name conflict.
  employee_info STRUCT < employer: STRING, id: BIGINT, address: STRING >,
```

```
-- A STRUCT as the element type of an ARRAY.
  places_lived ARRAY < STRUCT <street: STRING, city: STRING, country: STR
ING >>,
-- A STRUCT as the value portion of the key-value pairs in a MAP.
  memorable_moments MAP < STRING, STRUCT < year: INT, place: STRING, deta
ils: STRING >>,
-- A STRUCT where one of the fields is another STRUCT.
  current_address STRUCT < street_address: STRUCT <street_number: INT, stree
t_name: STRING, street_type: STRING>, country: STRING, postal_code: STRING >
)
STORED AS PARQUET;
```

The following example shows how to examine the structure of a table containing one or more STRUCT columns by using the DESCRIBE statement. You can visualize each STRUCT as its own table, with columns named the same as each field of the STRUCT. If the STRUCT is nested inside another complex type, such as ARRAY, you can extend the qualified name passed to DESCRIBE until the output shows just the STRUCT fields.

```
DESCRIBE struct_demo;
+-------------------+--------------------------+
| name              | type                     |
+-------------------+--------------------------+
| id                | bigint                   |
| name              | string                   |
| employee_info     | struct<                  |
|                   |   employer:string,       |
|                   |   id:bigint,             |
|                   |   address:string         |
|                   | >                        |
| places_lived      | array<struct<            |
|                   |   street:string,         |
|                   |   city:string,           |
|                   |   country:string         |
|                   | >>                       |
| memorable_moments | map<string,struct<       |
|                   |   year:int,              |
|                   |   place:string,          |
|                   |   details:string         |
|                   | >>                       |
| current_address   | struct<                  |
|                   |   street_address:struct< |
|                   |     street_number:int,   |
|                   |     street_name:string,  |
|                   |     street_type:string   |
|                   |   >,                     |
|                   |   country:string,        |
|                   |   postal_code:string     |
|                   | >                        |
+-------------------+--------------------------+
```

The top-level column EMPLOYEE_INFO is a STRUCT. Describing *TABLE_NAME.STRUCT_NAME* displays the fields of the STRUCT as if they were columns of a table:

```
DESCRIBE struct_demo.employee_info;
+----------+--------+
| name     | type   |
+----------+--------+
| employer | string |
| id       | bigint |
| address  | string |
+----------+--------+
```

Because PLACES_LIVED is a STRUCT inside an ARRAY, the initial DESCRIBE shows the structure of the ARRAY:

```
DESCRIBE struct_demo.places_lived;
+------+------------------+
| name | type             |
+------+------------------+
| item | struct<          |
|      |   street:string, |
|      |   city:string,   |
|      |   country:string |
|      | >                |
| pos  | bigint           |
+------+------------------+
```

Ask for the details of the ITEM field of the ARRAY to see just the layout of the STRUCT:

```
DESCRIBE struct_demo.places_lived.item;
+---------+--------+
| name    | type   |
+---------+--------+
| street  | string |
| city    | string |
| country | string |
+---------+--------+
```

Likewise, MEMORABLE_MOMENTS has a STRUCT inside a MAP, which requires an extra level of qualified name to see just the STRUCT part:

```
DESCRIBE struct_demo.memorable_moments;
+-------+------------------+
| name  | type             |
+-------+------------------+
| key   | string           |
| value | struct<          |
|       |   year:int,      |
|       |   place:string,  |
|       |   details:string |
|       | >                |
+-------+------------------+
```

For a MAP, ask to see the VALUE field to see the corresponding STRUCT fields in a table-like structure:

```
DESCRIBE struct_demo.memorable_moments.value;
+---------+--------+
| name    | type   |
+---------+--------+
| year    | int    |
| place   | string |
| details | string |
+---------+--------+
```

For a STRUCT inside a STRUCT, we can see the fields of the outer STRUCT:

```
DESCRIBE struct_demo.current_address;
+----------------+-----------------------+
| name           | type                  |
+----------------+-----------------------+
| street_address | struct<               |
|                |   street_number:int,  |
|                |   street_name:string, |
```

```
|                 |   street_type:string  |
|                 | >                     |
| country         | string                |
| postal_code     | string                |
+----------------+-----------------------+
```

Then we can use a further qualified name to see just the fields of the inner STRUCT:

```
DESCRIBE struct_demo.current_address.street_address;
+---------------+--------+
| name          | type   |
+---------------+--------+
| street_number | int    |
| street_name   | string |
| street_type   | string |
+---------------+--------+
```

The following example shows how to examine the structure of a table containing one or more STRUCT columns by using the DESCRIBE statement. You can visualize each STRUCT as its own table, with columns named the same as each field of the STRUCT. If the STRUCT is nested inside another complex type, such as ARRAY, you can extend the qualified name passed to DESCRIBE until the output shows just the STRUCT fields.

```
DESCRIBE struct_demo;
+-------------------+--------------------------+---------+
| name              | type                     | comment |
+-------------------+--------------------------+---------+
| id                | bigint                   |         |
| name              | string                   |         |
| employee_info     | struct<                  |         |
|                   |   employer:string,       |         |
|                   |   id:bigint,             |         |
|                   |   address:string         |         |
|                   | >                        |         |
| places_lived      | array<struct<            |         |
|                   |   street:string,         |         |
|                   |   city:string,           |         |
|                   |   country:string         |         |
|                   | >>                       |         |
| memorable_moments | map<string,struct<       |         |
|                   |   year:int,              |         |
|                   |   place:string,          |         |
|                   |   details:string         |         |
|                   | >>                       |         |
| current_address   | struct<                  |         |
|                   |   street_address:struct< |         |
|                   |     street_number:int,   |         |
|                   |     street_name:string,  |         |
|                   |     street_type:string   |         |
|                   |   >,                     |         |
|                   |   country:string,        |         |
|                   |   postal_code:string     |         |
|                   | >                        |         |
+-------------------+--------------------------+---------+
SELECT id, employee_info.id FROM struct_demo;

SELECT id, employee_info.id AS employee_id FROM struct_demo;

SELECT id, employee_info.id AS employee_id, employee_info.employer
  FROM struct_demo;

SELECT id, name, street, city, country
  FROM struct_demo, struct_demo.places_lived;
```

```
SELECT id, name, places_lived.pos, places_lived.street, places_lived.city, p
laces_lived.country
  FROM struct_demo, struct_demo.places_lived;

SELECT id, name, pl.pos, pl.street, pl.city, pl.country
  FROM struct_demo, struct_demo.places_lived AS pl;

SELECT id, name, places_lived.pos, places_lived.street, places_lived.city, p
laces_lived.country
  FROM struct_demo, struct_demo.places_lived;

SELECT id, name, pos, street, city, country
  FROM struct_demo, struct_demo.places_lived;

SELECT id, name, memorable_moments.key,
  memorable_moments.value.year,
  memorable_moments.value.place,
  memorable_moments.value.details
FROM struct_demo, struct_demo.memorable_moments
WHERE memorable_moments.key IN ('Birthday','Anniversary','Graduation');

SELECT id, name, mm.key, mm.value.year, mm.value.place, mm.value.details
  FROM struct_demo, struct_demo.memorable_moments AS mm
WHERE mm.key IN ('Birthday','Anniversary','Graduation');
SELECT id, name, memorable_moments.key, memorable_moments.value.year,
  memorable_moments.value.place, memorable_moments.value.details
FROM struct_demo, struct_demo.memorable_moments
WHERE key IN ('Birthday','Anniversary','Graduation');

SELECT id, name, key, value.year, value.place, value.details
  FROM struct_demo, struct_demo.memorable_moments
WHERE key IN ('Birthday','Anniversary','Graduation');
SELECT id, name, key, year, place, details
  FROM struct_demo, struct_demo.memorable_moments
WHERE key IN ('Birthday','Anniversary','Graduation');
SELECT id, name,
  current_address.street_address.street_number,
  current_address.street_address.street_name,
  current_address.street_address.street_type,
  current_address.country,
  current_address.postal_code
FROM struct_demo;
```

For example, this table uses a struct that encodes several data values for each phone number associated with a person. Each person can have a variable-length array of associated phone numbers, and queries can refer to the category field to locate specific home, work, mobile, and so on kinds of phone numbers.

```
CREATE TABLE contact_info_many_structs
(
  id BIGINT, name STRING,
  phone_numbers ARRAY < STRUCT <category:STRING, country_code:STRING, are
a_code:SMALLINT, full_number:STRING, mobile:BOOLEAN, carrier:STRING > >
) STORED AS PARQUET;
```

Because structs are naturally suited to composite values where the fields have different data types, you might use them to decompose things such as addresses:

```
CREATE TABLE contact_info_detailed_address
(
  id BIGINT, name STRING,
```

```
  address STRUCT < house_number:INT, street:STRING, street_type:STRING, a
partment:STRING, city:STRING, region:STRING, country:STRING >
);
```

In a big data context, splitting out data fields such as the number part of the address and the street name could let you do analysis on each field independently. For example, which streets have the largest number range of addresses, what are the statistical properties of the street names, which areas have a higher proportion of “Roads”, “Courts” or “Boulevards”, and so on.

TIMESTAMP data type

In Impala, the TIMESTAMP data type holds a value of date and time. It can be decomposed into year, month, day, hour, minute and seconds fields, but with no time zone information available, it does not correspond to any specific point in time.

Internally, the resolution of the time portion of a TIMESTAMP value is in nanoseconds.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME TIMESTAMP
```

```
TIMESTAMP [+ | -] INTERVAL INTERVAL
DATE_ADD (TIMESTAMP, INTERVAL INTERVAL TIME_UNIT)
```

Range: 1400-01-01 to 9999-12-31

Out of range TIMESTAMP values are converted to NULL.

The range of Impala TIMESTAMP is different from the Hive TIMESTAMP type.

INTERVAL expressions:

You can perform date arithmetic by adding or subtracting a specified number of time units, using the INTERVAL keyword and the + operator, the - operator, date_add() or date_sub().

The following units are supported for time_unit in the INTERVAL clause:

- YEAR[S]
- MONTH[S]
- WEEK[S]
- DAY[S]
- HOUR[S]
- MINUTE[S]
- SECOND[S]
- MILLISECOND[S]
- MICROSECOND[S]
- NANOSECOND[S]

You can only specify one time unit in each interval expression, for example INTERVAL 3 DAYS or INTERVAL 25 HOURS, but you can produce any granularity by adding together successive INTERVAL values, such as *TIMESTAMP_VALUE* + INTERVAL 3 WEEKS - INTERVAL 1 DAY + INTERVAL 10 MICROSECONDS.

Internal details: Represented in memory as a 16-byte value.

Time zones:

By default, Impala stores and interprets TIMESTAMP values in UTC time zone when writing to data files, reading from data files, or converting to and from system time values through functions.

When you set the --use_local_tz_for_unix_timestamp_conversions startup flag to TRUE, Impala treats the TIME STAMP values specified in the local time zone. The local time zone is determined in the following order with the TIMESTAMP query option takes the highest precedence:

1. The TIMESTAMP query option
2. $TZ environment variable
3. System time zone where the impalad coordinator runs

The --use_local_tz_for_unix_timestamp_conversions setting can be used to fix discrepancy in INTERVAL operations. For example, a TIMESTAMP + INTERVAL *N-HOURS* can be affected by Daylight Saving Time, which Impala does not consider by default as these operations are applied as if the timestamp was in UTC. You can use the --use_local_tz_for_unix_timestamp_conversions setting to fix the issue.

See Customizing time zones on page 51 for configuring to use custom time zone database and aliases.

See Impala date and time functions for the list of functions affected by the --use_local_tz_for_unix_timestamp_co nversions setting.

Time zone handling between Impala and Hive:

Interoperability between Hive and Impala is different depending on the file format.

- Text

  For text tables, TIMESTAMP values can be written and read interchangeably by Impala and Hive as Hive reads and writes TIMESTAMP values without converting with respect to time zones.
- Parquet

  Note: This section only applies to INT96 TIMESTAMP.

  When Hive writes to Parquet data files, the TIMESTAMP values are normalized to UTC from the local time zone of the host where the data was written. On the other hand, Impala does not make any time zone adjustment when it writes or reads INT96 TIMESTAMP values to Parquet files. This difference in time zone handling can cause potentially inconsistent results when Impala processes TIMESTAMP values in the Parquet files written by Hive.

  To avoid incompatibility problems or having to code workarounds, you can specify one or both of these impalad startup flags:

  - --use_local_tz_for_unix_timestamp_conversions=true
  - --convert_legacy_hive_parquet_utc_timestamps=true

  When the ##convert_legacy_hive_parquet_utc_timestamps setting is enabled, Impala recognizes the Parquet data files written by Hive, and applies the same UTC-to-local-timezone conversion logic during the query as Hive does.

  In Impala 3.0 and lower, the ##convert_legacy_hive_parquet_utc_timestamps setting had a severe impact on multi-threaded performance. The new time zone implementation in Impala 3.1 eliminated most of the performance overhead and made Impala scale well to multiple threads. The ##convert_legacy_hive_parquet_utc_timestamps setting is turned off by default for a performance reason. To avoid unexpected incompatibility problems, you should turn on the option when processing TIMESTAMP columns in Parquet files written by Hive.

  Hive currently cannot write INT64 TIMESTAMP values.

  In Impala 3.2 and higher, INT64 TIMESTAMP values annotated with the TIMESTAMP_MILLIS or TIME STAMP_MICROS OriginalType are assumed to be always UTC normalized, so the UTC to local conversion will be always done. INT64 TIMESTAMP annotated with the TIMESTAMP LogicalType specifies whether UTC to local conversion is necessary depending on the Parquet metadata.

Conversions:

Impala automatically converts STRING literals of the correct format into TIMESTAMP values. Timestamp values are accepted in the format 'yyyy#MM#dd HH:mm:ss.SSSSSS', and can consist of just the date, or just the time, with or

without the fractional second portion. For example, you can specify TIMESTAMP values such as '1966#07#30', '08:30:00', or '1985#09#25 17:45:30.005'.

Leading zeroes are not required in the numbers representing the date component, such as month and date, or the time component, such as hour, minute, and second. For example, Impala accepts both '2018#1#1 01:02:03' and '2018-01-01 1:2:3' as valid.

In STRING to TIMESTAMP conversions, leading and trailing white spaces, such as a space, a tab, a newline, or a carriage return, are ignored. For example, Impala treats the following as equivalent: '1999#12#01 01:02:03 ', ' 1999#12#01 01:02:03', '1999#12#01 01:02:03\r\n\t'.

When you convert or cast a STRING literal to TIMESTAMP, you can use the following separators between the date part and the time part:

- One or more space characters

  Example: CAST('2001-01-09 01:05:01' AS TIMESTAMP)
- The character “T”

  Example: CAST('2001-01-09T01:05:01' AS TIMESTAMP)

Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

In Impala 1.3 and higher, the FROM_UNIXTIME() and UNIX_TIMESTAMP() functions allow a wider range of format strings, with more flexibility in element order, repetition of letter placeholders, and separator characters. In Impala 2.3 and higher, the UNIX_TIMESTAMP() function also allows a numeric timezone offset to be specified as part of the input string. See Impala date and time functions on page 347 for details.

In Impala 2.2.0 and higher, built-in functions that accept or return integers representing TIMESTAMP values use the BIGINT type for parameters and return values, rather than INT. This change lets the date and time functions avoid an overflow error that would otherwise occur on January 19th, 2038 (known as the “Year 2038 problem” or “Y2K38 problem”). This change affects the FROM_UNIXTIME() and UNIX_TIMESTAMP() functions. You might need to change application code that interacts with these functions, change the types of columns that store the return values, or add CAST() calls to SQL statements that call these functions.

Partitioning:

Although you cannot use a TIMESTAMP column as a partition key, you can extract the individual years, months, days, hours, and so on and partition based on those columns. Because the partition key column values are represented in HDFS directory names, rather than as fields in the data files themselves, you can also keep the original TIMESTAMP values if desired, without duplicating data or wasting storage space. See *Partition Key Columns* for more details on partitioning with date and time values.

```
[localhost:21000] > create table timeline (event string) partitioned by (happened timestamp);
ERROR: AnalysisException: Type 'TIMESTAMP' is not supported as partition-column type in column: happened
```

NULL considerations: Casting any unrecognized STRING value to this type produces a NULL value.

HBase considerations: This data type is fully compatible with HBase tables.

Parquet consideration: INT96 and INT64 encoded Parquet timestamps are supported in Impala.

Parquet considerations: This type is fully compatible with Parquet tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

Kudu considerations:

In Impala 2.9 and higher, you can include TIMESTAMP columns in Kudu tables, instead of representing the date and time as a BIGINT value. The behavior of TIMESTAMP for Kudu tables has some special considerations:

- Any nanoseconds in the original 96-bit value produced by Impala are not stored, because Kudu represents date/time columns using 64-bit values. The nanosecond portion of the value is rounded, not truncated. Therefore, a TIMESTAMP value that you store in a Kudu table might not be bit-for-bit identical to the value returned by a query.
- The conversion between the Impala 96-bit representation and the Kudu 64-bit representation introduces some performance overhead when reading or writing TIMESTAMP columns. You can minimize the overhead during writes by performing inserts through the Kudu API. Because the overhead during reads applies to each query, you might continue to use a BIGINT column to represent date/time values in performance-critical applications.
- The Impala TIMESTAMP type has a narrower range for years than the underlying Kudu data type. Impala can represent years 1400-9999. If year values outside this range are written to a Kudu table by a non-Impala client, Impala returns NULL by default when reading those TIMESTAMP values during a query. Or, if the ABORT_ON_ERROR query option is enabled, the query fails when it encounters a value with an out-of-range year.

Restrictions:

If you cast a STRING with an unrecognized format to a TIMESTAMP, the result is NULL rather than an error. Make sure to test your data pipeline to be sure any textual date and time values are in a format that Impala TIMESTAMP can recognize.

Currently, Avro tables cannot contain TIMESTAMP columns. If you need to store date and time values in Avro tables, as a workaround you can use a STRING representation of the values, convert the values to BIGINT with the UNIX_TIMESTAMP() function, or create separate numeric columns for individual date and time fields using the EXTRACT() function.

Examples:

The following examples demonstrate using TIMESTAMP values with built-in functions:

```
select cast('1966-07-30' as timestamp);
select cast('1985-09-25 17:45:30.005' as timestamp);
select cast('08:30:00' as timestamp);
select hour('1970-01-01 15:30:00');         -- Succeeds, returns 15.
select hour('1970-01-01 15:30');            -- Returns NULL because seconds
field required.
select hour('1970-01-01 27:30:00');         -- Returns NULL because hour val
ue out of range.
select dayofweek('2004-06-13');             -- Returns 1, representing Sund
ay.
select dayname('2004-06-13');               -- Returns 'Sunday'.
select date_add('2004-06-13', 365);         -- Returns 2005-06-13 with zeros
 for hh:mm:ss fields.
select day('2004-06-13');                   -- Returns 13.
select datediff('1989-12-31','1984-09-01'); -- How many days between these 2
 dates?
select now();                               -- Returns current date and time
 in local timezone.
```

The following examples demonstrate using TIMESTAMP values with HDFS-backed tables:

```
create table dates_and_times (t timestamp);
insert into dates_and_times values
  ('1966-07-30'), ('1985-09-25 17:45:30.005'), ('08:30:00'), (now());
```

The following examples demonstrate using TIMESTAMP values with Kudu tables:

```
create table timestamp_t (x int primary key, s string, t timestamp, b bigint
)
```

```
  partition by hash (x) partitions 16
  stored as kudu;

-- The default value of now() has microsecond precision, so the final 3 d
igits
-- representing nanoseconds are all zero.
insert into timestamp_t values (1, cast(now() as string), now(), unix_timest
amp(now()));

-- Values with 1-499 nanoseconds are rounded down in the Kudu TIMESTAMP co
lumn.
insert into timestamp_t values (2, cast(now() + interval 100 nanoseconds as
 string), now() + interval 100 nanoseconds, unix_timestamp(now() + interval
 100 nanoseconds));
insert into timestamp_t values (3, cast(now() + interval 499 nanoseconds as
string), now() + interval 499 nanoseconds, unix_timestamp(now() + interval 4
99 nanoseconds));

-- Values with 500-999 nanoseconds are rounded up in the Kudu TIMESTAMP c
olumn.
insert into timestamp_t values (4, cast(now() + interval 500 nanoseconds as
 string), now() + interval 500 nanoseconds, unix_timestamp(now() + interval
 500 nanoseconds));
insert into timestamp_t values (5, cast(now() + interval 501 nanoseconds as
 string), now() + interval 501 nanoseconds, unix_timestamp(now() + interval
501 nanoseconds));

-- The string representation shows how underlying Impala TIMESTAMP can have
nanosecond precision.
-- The TIMESTAMP column shows how timestamps in a Kudu table are rounded to
 microsecond precision.
-- The BIGINT column represents seconds past the epoch and so if not affect
ed much by nanoseconds.
select s, t, b from timestamp_t order by t;
+-------------------------------+-------------------------------+-----------
-+
| s                             | t                             | b
  |
+-------------------------------+-------------------------------+---------
---+
| 2017-05-31 15:30:05.107157000 | 2017-05-31 15:30:05.107157000 | 1496244605
 |
| 2017-05-31 15:30:28.868151100 | 2017-05-31 15:30:28.868151000 | 1496244628
 |
| 2017-05-31 15:34:33.674692499 | 2017-05-31 15:34:33.674692000 | 1496244
873 |
| 2017-05-31 15:35:04.769166500 | 2017-05-31 15:35:04.769167000 | 1496244904
 |
| 2017-05-31 15:35:33.033082501 | 2017-05-31 15:35:33.033083000 | 1496244933
 |
+-------------------------------+-------------------------------+-----------
-+
```

Added in: Available in all versions of Impala.

Related information:

- To convert to or from different date formats, or perform date arithmetic, use the date and time functions described in Impala date and time functions on page 347. In particular, the from_unixtime() function requires a case-sensitive format string such as "yyyy-MM-dd HH:mm:ss.SSSS", matching one of the allowed variations of a TIMESTAMP value (date plus time, only date, only time, optional fractional seconds).
- See SQL differences between Impala and Hive on page 477 for details about differences in TIMESTAMP handling between Impala and Hive.

Related Information

Hive data types
DATE data type
Partitioning

Customizing time zones

Impala 3.1 onward, you can customize the time zone definitions used in Impala.

- By default, Impala uses the OS's time zone database located in /usr/share/zoneinfo. This directory contains the IANA timezone database in a compiled binary format. The contents of the zoneinfo directory is controlled by the OS's package manager.
- Use the following start-up flags managed as impalad safety valves in Cloudera Manager.
  - ##hdfs_zone_info_zip: This flag allows Impala administrators to specify a custom timezone database. The flag should be set to a shared (not necessarily HDFS) path that points to a zip archive of a custom IANA timezone database. The timezone database is expected to be in a compiled binary format. If the startup flag is set, Impala will use the specified timezone database instead of the default /usr/share/zoneinfo database. The timezone db upgrade process is described in detail below.
  - ##hdfs_zone_alias_conf: This flag allows Impala administrators to specify definitions for custom timezone aliases. The flag should be set to a shared (not necessarily HDFS) path that specifies a config file containing custom timezone alias definitions. This config file can be used as a workaround for users who want to keep using their legacy timezone names. Configuring custom aliases is described in detail below.

Upgrading custom IANA time zone database:

1. Download latest IANA time zone database distribution:

   ```
   git clone https://github.com/eggert/tz
   ```

   Alternatively, download a specific tzdb version from:

   ```
    https://www.iana.org/time-zones/repository
   ```

2. Build timezone tools:

   ```
   cd tz
   make TOPDIR=tzdata install
   ```

3. Generate the compiled binary time zone database:

   ```
   ./zic -d ./tzdata/etc/zoneinfo africa antarctica asia australasia backward
    backzone etcetera europe factory northamerica pacificnew southamerica sys
   temv
   ```

4. Create zip archive:

   ```
   pushd ./tzdata/etc
   zip -r zoneinfo.zip zoneinfo
   popd
   ```

5. Copy the time zone database to HDFS:

   ```
   hdfs dfs -mkdir -p /tzdb/latest
   hdfs dfs -copyFromLocal ./tzdata/etc/zoneinfo.zip /tzdb/latest
   ```

6. Set the ##hdfs_zone_info_zip startup flag to /tzdb/latest/zoneinfo.zip as an impalad safety valve.
7. Perform a full restart of Impala service.

Configuring custom time zone aliases:

1. Create a tzalias.conf config file that contains time zone alias definitions formatted as ALIAS = DEFINITION. For example:

```
#
# Define aliases for existing timezone names:
#
Universal Coordinated Time = UTC
Mideast/Riyadh89 = Asia/Riyadh
PDT = America/Los_Angeles
#
# Define aliases as UTC offsets in seconds:
#
GMT-01:00 = 3600
GMT+01:00 = -3600
```

2. Copy the config file to HDFS:

```
hdfs dfs -mkdir -p /tzdb
hdfs dfs -copyFromLocal tzalias.conf /tzdb
```

3. Set the ##hdfs_zone_alias_conf startup flag to /tzdb/tzalias.conf as an impalad safety valve.
4. Perform a full restart of Impala service.

Added in: Impala 3.1.

TINYINT data type

A 1-byte integer data type used in CREATE TABLE and ALTER TABLE statements.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME TINYINT
```

Range: -128 .. 127. There is no UNSIGNED subtype.

Conversions: Impala automatically converts to a larger integer type (SMALLINT, INT, or BIGINT) or a floating-point type (FLOAT or DOUBLE) automatically. Use CAST() to convert to STRING or TIMESTAMP. Casting an integer or floating-point value N to TIMESTAMP produces a value that is N seconds past the start of the epoch date (January 1, 1970). By default, the result value represents a date and time in the UTC time zone. If the setting ##use_local_tz_for_unix_timestamp_conversions=true is in effect, the resulting TIMESTAMP represents a date and time in the local time zone.

Impala does not return column overflows as NULL, so that customers can distinguish between NULL data and overflow conditions similar to how they do so with traditional database systems. Impala returns the largest or smallest value in the range for the type. For example, valid values for a tinyint range from -128 to 127. In Impala, a tinyint with a value of -200 returns -128 rather than NULL. A tinyint with a value of 200 returns 127.

Usage notes:

For a convenient and automated way to check the bounds of the TINYINT type, call the functions MIN_TINYINT() and MAX_TINYINT().

If an integer value is too large to be represented as a TINYINT, use a SMALLINT instead.

NULL considerations: Casting any non-numeric value to this type produces a NULL value.

Examples:

```
CREATE TABLE t1 (x TINYINT);
SELECT CAST(100 AS TINYINT);
```

Parquet considerations:

Physically, Parquet files represent TINYINT and SMALLINT values as 32-bit integers. Although Impala rejects attempts to insert out-of-range values into such columns, if you create a new table with the CREATE TABLE ... LIKE PARQUET syntax, any TINYINT or SMALLINT columns in the original table turn into INT columns in the new table.

HBase considerations: This data type is fully compatible with HBase tables.

Text table considerations: Values of this type are potentially larger in text tables than in tables using Parquet or other binary formats.

Internal details: Represented in memory as a 1-byte value.

Added in: Available in all versions of Impala.

Column statistics considerations: Because this type has a fixed size, the maximum and average size fields are always filled in for column statistics, even before you run the COMPUTE STATS statement.

VARCHAR data type

A variable-length character type, truncated during processing if necessary to fit within the specified length.

Syntax:

In the column definition of a CREATE TABLE statement:

```
COLUMN_NAME VARCHAR(MAX_LENGTH)
```

The maximum length you can specify is 65,535.

Partitioning: This type can be used for partition key columns. Because of the efficiency advantage of numeric values over character-based values, if the partition key is a string representation of a number, prefer to use an integer type with sufficient range (INT, BIGINT, and so on) where practical.

HBase considerations: This data type cannot be used with HBase tables.

Parquet considerations:

- This type can be read from and written to Parquet files.
- There is no requirement for a particular level of Parquet.
- Parquet files generated by Impala and containing this type can be freely interchanged with other components such as Hive and MapReduce.
- Parquet data files can contain values that are longer than allowed by the VARCHAR(*N*) length limit. Impala ignores any extra trailing characters when it processes those values during a query.

Text table considerations:

Text data files can contain values that are longer than allowed by the VARCHAR(*N*) length limit. Any extra trailing characters are ignored when Impala processes those values during a query.

Avro considerations: The Avro specification allows string values up to 2**64 bytes in length. Impala queries for Avro tables use 32-bit integers to hold string lengths. In Impala 2.5 and higher, Impala truncates CHAR and VARCHAR values in Avro tables to (2**31)-1 bytes. If a query encounters a STRING value longer than (2**31)-1 bytes in an Avro table, the query fails. In earlier releases, encountering such long values in an Avro table could cause a crash.

Schema evolution considerations:

You can use ALTER TABLE ... CHANGE to switch column data types to and from VARCHAR. You can convert from STRING to VARCHAR(*N*), or from VARCHAR(*N*) to STRING, or from CHAR(*N*) to VARCHAR(*N*), or from VARCHAR(*N*) to CHAR(*N*). When switching back and forth between VARCHAR and CHAR, you can also change the length value. This schema evolution works the same for tables using any file format. If a table contains values

longer than the maximum length defined for a VARCHAR column, Impala does not return an error. Any extra trailing characters are ignored when Impala processes those values during a query.

Compatibility:

This type is available in Impala version 2.0 or higher.

Internal details: Represented in memory as a byte array with the minimum size needed to represent each value.

Column statistics considerations: Because the values of this type have variable size, none of the column statistics fields are filled in until you run the COMPUTE STATS statement.

Kudu considerations:

Currently, the data types CHAR, ARRAY, MAP, and STRUCT cannot be used with Kudu tables.

Restrictions:

All data in CHAR and VARCHAR columns must be in a character encoding that is compatible with UTF-8. If you have binary data from another database system (that is, a BLOB type), use a STRING column to hold it.

Examples:

The following examples show how long and short VARCHAR values are treated. Values longer than the maximum specified length are truncated by CAST(), or when queried from existing data files. Values shorter than the maximum specified length are represented as the actual length of the value, with no extra padding as seen with CHAR values.

```
create table varchar_1 (s varchar(1));
create table varchar_4 (s varchar(4));
create table varchar_20 (s varchar(20));

insert into varchar_1 values (cast('a' as varchar(1))), (cast('b' as varchar
(1))), (cast('hello' as varchar(1))), (cast('world' as varchar(1)));
insert into varchar_4 values (cast('a' as varchar(4))), (cast('b' as varcha
r(4))), (cast('hello' as varchar(4))), (cast('world' as varchar(4)));
insert into varchar_20 values (cast('a' as varchar(20))), (cast('b' as var
char(20))), (cast('hello' as varchar(20))), (cast('world' as varchar(20)));
select * from varchar_1;
+---+
| s |
+---+
| a |
| b |
| h |
| w |
+---+
select * from varchar_4;
+------+
| s    |
+------+
| a    |
| b    |
| hell |
| worl |
+------+
[localhost:21000] > select * from varchar_20;
+-------+
| s     |
+-------+
| a     |
| b     |
| hello |
| world |
+-------+
select concat('[',s,']') as s from varchar_20;
+---------+
```

```
| s       |
+---------+
| [a]     |
| [b]     |
| [hello] |
| [world] |
+---------+
```

The following example shows how identical VARCHAR values compare as equal, even if the columns are defined with different maximum lengths. Both tables contain 'a' and 'b' values. The longer 'hello' and 'world' values from the VARCHAR_20 table were truncated when inserted into the VARCHAR_1 table.

```
select s from varchar_1 join varchar_20 using (s);
+-------+
| s     |
+-------+
| a     |
| b     |
+-------+
```

The following examples show how VARCHAR values are freely interchangeable with STRING values in contexts such as comparison operators and built-in functions:

```
select length(cast('foo' as varchar(100))) as length;
+--------+
| length |
+--------+
| 3      |
+--------+
select cast('xyz' as varchar(5)) > cast('abc' as varchar(10)) as greater;
+---------+
| greater |
+---------+
| true    |
+---------+
```

UDF considerations: This type cannot be used for the argument or return type of a user-defined function (UDF) or user-defined aggregate function (UDA).

Complex types

Complex types (also referred to as *nested types*) let you represent multiple data values within a single row/column position. They differ from the familiar column types such as BIGINT and STRING, known as *scalar types* or *primitive types*, which represent a single data value within a given row/column position. Impala supports the complex types ARRAY, MAP, and STRUCT in Impala 2.3 and higher. The Hive UNION type is not currently supported.

Once you understand the basics of complex types, refer to the individual type topics when you need to refresh your memory about syntax and examples:

- ARRAY complex type on page 13
- STRUCT complex type on page 40
- MAP complex type on page 32

Benefits of Impala complex types

The reasons for using Impala complex types include the following:

- You already have data produced by Hive or other non-Impala component that uses the complex type column names. You might need to convert the underlying data to Parquet to use it with Impala.

- Your data model originates with a non-SQL programming language or a NoSQL data management system. For example, if you are representing Python data expressed as nested lists, dictionaries, and tuples, those data structures correspond closely to Impala ARRAY, MAP, and STRUCT types.
- Your analytic queries involving multiple tables could benefit from greater locality during join processing. By packing more related data items within each HDFS data block, complex types let join queries avoid the network overhead of the traditional Hadoop shuffle or broadcast join techniques.

The Impala complex type support produces result sets with all scalar values, and the scalar components of complex types can be used with all SQL clauses, such as GROUP BY, ORDER BY, all kinds of joins, subqueries, and inline views. The ability to process complex type data entirely in SQL reduces the need to write application-specific code in Java or other programming languages to deconstruct the underlying data structures.

Overview of Impala complex types

The ARRAY and MAP types are closely related: they represent collections with arbitrary numbers of elements, where each element is the same type. In contrast, STRUCT groups together a fixed number of items into a single element. The parts of a STRUCT element (the *fields*) can be of different types, and each field has a name.

The elements of an ARRAY or MAP, or the fields of a STRUCT, can also be other complex types. You can construct elaborate data structures with up to 100 levels of nesting. For example, you can make an ARRAY whose elements are STRUCTs. Within each STRUCT, you can have some fields that are ARRAY, MAP, or another kind of STRUCT. The Impala documentation uses the terms complex and nested types interchangeably; for simplicity, it primarily uses the term complex types, to encompass all the properties of these types.

When visualizing your data model in familiar SQL terms, you can think of each ARRAY or MAP as a miniature table, and each STRUCT as a row within such a table. By default, the table represented by an ARRAY has two columns, POS to represent ordering of elements, and ITEM representing the value of each element. Likewise, by default, the table represented by a MAP encodes key-value pairs, and therefore has two columns, KEY and VALUE.

The ITEM and VALUE names are only required for the very simplest kinds of ARRAY and MAP columns, ones that hold only scalar values. When the elements within the ARRAY or MAP are of type STRUCT rather than a scalar type, then the result set contains columns with names corresponding to the STRUCT fields rather than ITEM or VALUE.

You write most queries that process complex type columns using familiar join syntax, even though the data for both sides of the join resides in a single table. The join notation brings together the scalar values from a row with the values from the complex type columns for that same row. The final result set contains all scalar values, allowing you to do all the familiar filtering, aggregation, ordering, and so on for the complex data entirely in SQL or using business intelligence tools that issue SQL queries.

Behind the scenes, Impala ensures that the processing for each row is done efficiently on a single host, without the network traffic involved in broadcast or shuffle joins. The most common type of join query for tables with complex type columns is INNER JOIN, which returns results only in those cases where the complex type contains some elements. Therefore, most query examples in this section use either the INNER JOIN clause or the equivalent comma notation.

Note:

Although Impala can query complex types that are present in Parquet files, Impala currently cannot create new Parquet files containing complex types. Therefore, the discussion and examples presume that you are working with existing Parquet data produced through Hive, Spark, or some other source. See Constructing Parquet / ORC files with complex columns using Hive for examples of constructing Parquet data files with complex type columns.

For learning purposes, you can create empty tables with complex type columns and practice query syntax, even if you do not have sample data with the required structure.

Design considerations for complex types

When planning to use Impala complex types, and designing the Impala schema, first learn how this kind of schema differs from traditional table layouts from the relational database and data warehousing fields. Because you might

have already encountered complex types in a Hadoop context while using Hive for ETL, also learn how to write high-performance analytic queries for complex type data using Impala SQL syntax.

How complex types differ from traditional data warehouse schemas

Complex types let you associate arbitrary data structures with a particular row. If you are familiar with schema design for relational database management systems or data warehouses, a schema with complex types has the following differences:

- Logically, related values can now be grouped tightly together in the same table.

  In traditional data warehousing, related values were typically arranged in one of two ways:

  - Split across multiple normalized tables. Foreign key columns specified which rows from each table were associated with each other. This arrangement avoided duplicate data and therefore the data was compact, but join queries could be expensive because the related data had to be retrieved from separate locations. (In the case of distributed Hadoop queries, the joined tables might even be transmitted between different hosts in a cluster.)
  - Flattened into a single denormalized table. Although this layout eliminated some potential performance issues by removing the need for join queries, the table typically became larger because values were repeated. The extra data volume could cause performance issues in other parts of the workflow, such as longer ETL cycles or more expensive full-table scans during queries.

  Complex types represent a middle ground that addresses these performance and volume concerns. By physically locating related data within the same data files, complex types increase locality and reduce the expense of join queries. By associating an arbitrary amount of data with a single row, complex types avoid the need to repeat lengthy values such as strings. Because Impala knows which complex type values are associated with each row, you can save storage by avoiding artificial foreign key values that are only used for joins. The flexibility of the STRUCT, ARRAY, and MAP types lets you model familiar constructs such as fact and dimension tables from a data warehouse, and wide tables representing sparse matrices.

Physical storage for complex types in Parquet

Physically, the scalar and complex columns in each row are located adjacent to each other in the same Parquet data file, ensuring that they are processed on the same host rather than being broadcast across the network when cross-referenced within a query. This co-location simplifies the process of copying, converting, and backing all the columns up at once. Because of the column-oriented layout of Parquet files, you can still query only the scalar columns of a table without imposing the I/O penalty of reading the (possibly large) values of the composite columns.

Within each Parquet data file, the constituent parts of complex type columns are stored in column-oriented format:

- Each field of a STRUCT type is stored like a column, with all the scalar values adjacent to each other and encoded, compressed, and so on using the Parquet space-saving techniques.
- For an ARRAY containing scalar values, all those values (represented by the ITEM pseudocolumn) are stored adjacent to each other.
- For a MAP, the values of the KEY pseudocolumn are stored adjacent to each other. If the VALUE pseudocolumn is a scalar type, its values are also stored adjacent to each other.
- If an ARRAY element, STRUCT field, or MAP VALUE part is another complex type, the column-oriented storage applies to the next level down (or the next level after that, and so on for deeply nested types) where the final elements, fields, or values are of scalar types.

The numbers represented by the POS pseudocolumn of an ARRAY are not physically stored in the data files. They are synthesized at query time based on the order of the ARRAY elements associated with each row.

File format support for Impala complex types

Currently, Impala queries support complex type data in the Parquet and ORC file formats. See *Using Parquet data files* for details about the performance benefits and physical layout of Parquet file format.

Because Impala does not parse the data structures containing nested types for unsupported formats such as text, Avro, SequenceFile, or RCFile, you cannot use data files in these formats with Impala, even if the query does not refer to the nested type columns. Also, if a table using an unsupported format originally contained nested type columns, and then those columns were dropped from the table using ALTER TABLE ... DROP COLUMN, any existing data files in the table still contain the nested type data and Impala queries on that table will generate errors.

The one exception to the preceding rule is COUNT(*) queries on RCFile tables that include complex types. Such queries are allowed in Impala 2.6 and higher.

You can perform DDL operations for tables involving complex types in most file formats other than Parquet or ORC. You cannot create tables in Impala with complex types using text files.

You can have a partitioned table with complex type columns that uses a format other than Parquet or ORC, and use ALTER TABLE to change the file format to Parquet/ORC for individual partitions. When you put Parquet/ORC files into those partitions, Impala can run queries against that data as long as the query does not involve any of the non-Parquet and non-ORC partitions.

If you use the `parquet-tools` command to examine the structure of a Parquet data file that includes complex types, you see that both ARRAY and MAP are represented as a Bag in Parquet terminology, with all fields marked Optional because Impala allows any column to be nullable.

Impala supports either 2-level and 3-level encoding within each Parquet data file. When constructing Parquet data files outside Impala, use either encoding style but do not mix 2-level and 3-level encoding within the same data file.

Choosing between complex types and normalized tables

Choosing between multiple normalized fact and dimension tables, or a single table containing complex types, is an important design decision.

- If you are coming from a traditional database or data warehousing background, you might be familiar with how to split up data between tables. Your business intelligence tools might already be optimized for dealing with this kind of multi-table scenario through join queries.
- If you are pulling data from Impala into an application written in a programming language that has data structures analogous to the complex types, such as Python or Java, complex types in Impala could simplify data interchange and improve understandability and reliability of your program logic.
- You might already be faced with existing infrastructure or receive high volumes of data that assume one layout or the other. For example, complex types are popular with web-oriented applications, for example to keep information about an online user all in one place for convenient lookup and analysis, or to deal with sparse or constantly evolving data fields.
- If some parts of the data change over time while related data remains constant, using multiple normalized tables lets you replace certain parts of the data without reloading the entire data set. Conversely, if you receive related data all bundled together, such as in JSON files, using complex types can save the overhead of splitting the related items across multiple tables.
- From a performance perspective:
  - In Parquet or ORC tables, Impala can skip columns that are not referenced in a query, avoiding the I/O penalty of reading the embedded data. When complex types are nested within a column, the data is physically divided at a very granular level; for example, a query referring to data nested multiple levels deep in a complex type column does not have to read all the data from that column, only the data for the relevant parts of the column type hierarchy.
  - Complex types avoid the possibility of expensive join queries when data from fact and dimension tables is processed in parallel across multiple hosts. All the information for a row

containing complex types is typically to be in the same data block, and therefore does not need to be transmitted across the network when joining fields that are all part of the same row.

- The tradeoff with complex types is that fewer rows fit in each data block. Whether it is better to have more data blocks with fewer rows, or fewer data blocks with many rows, depends on the distribution of your data and the characteristics of your query workload. If the complex columns are rarely referenced, using them might lower efficiency. If you are seeing low parallelism due to a small volume of data (relatively few data blocks) in each table partition, increasing the row size by including complex columns might produce more data blocks and thus spread the work more evenly across the cluster. See *Scalability considerations* for more on this advanced topic.

Differences between Impala and Hive complex types

Impala can query Parquet and ORC tables containing ARRAY, STRUCT, and MAP columns produced by Hive. There are some differences to be aware of between the Impala SQL and Hive SQL syntax for complex types, primarily for queries.

Impala supports a subset of the syntax that Hive supports for specifying ARRAY, STRUCT, and MAP types in the CREATE TABLE statements.

Because Impala STRUCT columns include user-specified field names, you use the NAMED_STRUCT() constructor in Hive rather than the STRUCT() constructor when you populate an Impala STRUCT column using a Hive INSERT statement.

The Hive UNION type is not currently supported in Impala.

While Impala usually aims for a high degree of compatibility with Hive SQL query syntax, Impala syntax differs from Hive for queries involving complex types. The differences are intended to provide extra flexibility for queries involving these kinds of tables.

- Impala uses dot notation for referring to element names or elements within complex types, and join notation for cross-referencing scalar columns with the elements of complex types within the same row, rather than the LATERAL VIEW clause and EXPLODE() function of Hive SQL.
- Using join notation lets you use all the kinds of join queries with complex type columns. For example, you can use a LEFT OUTER JOIN, LEFT ANTI JOIN, or LEFT SEMI JOIN query to evaluate different scenarios where the complex columns do or do not contain any elements.
- You can include references to collection types inside subqueries and inline views. For example, you can construct a FROM clause where one of the “tables” is a subquery against a complex type column, or use a subquery against a complex type column as the argument to an IN or EXISTS clause.
- The Impala pseudocolumn POS lets you retrieve the position of elements in an array along with the elements themselves, equivalent to the POSEXPLODE() function of Hive SQL. You do not use index notation to retrieve a single array element in a query; the join query loops through the array elements and you use WHERE clauses to specify which elements to return.
- Join clauses involving complex type columns do not require an ON or USING clause. Impala implicitly applies the join key so that the correct array entries or map elements are associated with the correct row from the table.
- Impala does not currently support the UNION complex type.

Limitations and restrictions for complex types

Complex type columns can only be used in tables or partitions with the Parquet or ORC file format.

Complex type columns cannot be used as partition key columns in a partitioned table.

When you use complex types with the ORDER BY, GROUP BY, HAVING, or WHERE clauses, you cannot refer to the column name by itself. Instead, you refer to the names of the scalar values within the complex type, such as the ITEM, POS, KEY, or VALUE pseudocolumns, or the field names from a STRUCT.

The maximum depth of nesting for complex types is 100 levels.

The maximum length of the column definition for any complex type, including declarations for any nested types, is 4000 characters.

For ideal performance and scalability, use small or medium-sized collections, where all the complex columns contain at most a few hundred megabytes per row. Remember, all the columns of a row are stored in the same HDFS data block, whose size in Parquet files typically ranges from 256 MB to 1 GB.

Including complex type columns in a table introduces some overhead that might make queries that do not reference those columns somewhat slower than Impala queries against tables without any complex type columns. Expect at most a 2x slowdown compared to tables that do not have any complex type columns.

Currently, the COMPUTE STATS statement does not collect any statistics for columns containing complex types. Impala uses heuristics to construct execution plans involving complex type columns.

Currently, Impala built-in functions and user-defined functions cannot accept complex types as parameters or produce them as function return values. (When the complex type values are materialized in an Impala result set, the result set contains the scalar components of the values, such as the POS or ITEM for an ARRAY, the KEY or VALUE for a MAP, or the fields of a STRUCT; these scalar data items can be used with built-in functions and UDFs as usual.)

Impala currently cannot write new data files containing complex type columns. Therefore, although the SELECT statement works for queries involving complex type columns, you cannot use a statement form that writes data to complex type columns, such as CREATE TABLE AS SELECT or INSERT ... SELECT. To create data files containing complex type data, use the Hive INSERT statement, or another ETL mechanism such as MapReduce jobs, Spark jobs, Pig, and so on.

Currently, Impala can query complex type columns only from Parquet/ORC tables or Parquet/ORC partitions within partitioned tables. Although you can use complex types in tables with Avro, text, and other file formats as part of your ETL pipeline, for example as intermediate tables populated through Hive, doing analytics through Impala requires that the data eventually ends up in a Parquet/ORC table. The requirement for Parquet/ORC data files means that you can use complex types with Impala tables hosted on other kinds of file storage systems such as Isilon and Amazon S3, but you cannot use Impala to query complex types from HBase tables.

Using complex types in SQL

When using complex types through SQL in Impala, you learn the notation for < > delimiters for the complex type columns in CREATE TABLE statements, and how to construct join queries to “unpack” the scalar values nested inside the complex data structures. You might need to condense a traditional RDBMS or data warehouse schema into a smaller number of Parquet tables, and use Hive, Spark, Pig, or other mechanism outside Impala to populate the tables with data.

Complex type syntax for DDL statements

The definition of *DATA_TYPE*, as seen in the CREATE TABLE and ALTER TABLE statements, now includes complex types in addition to primitive types:

```
  primitive_type
| array_type
| map_type
| struct_type
```

Unions are not currently supported.

Array, struct, and map column type declarations are specified in the CREATE TABLE statement. You can also add or change the type of complex columns through the ALTER TABLE statement.

Currently, Impala queries allow complex types only in tables that use the Parquet or ORC format. If an Impala query encounters complex types in a table or partition using any another file format, the query returns a runtime error.

You can use ALTER TABLE ... SET FILEFORMAT PARQUET to change the file format of an existing table containing complex types to Parquet, after which Impala can query it. Make sure to load Parquet files into the table after changing the file format, because the ALTER TABLE ... SET FILEFORMAT statement does not convert existing data to the new file format.

Partitioned tables can contain complex type columns. All the partition key columns must be scalar types.

Because use cases for Impala complex types require that you already have Parquet/ORC data files produced outside of Impala, you can use the Impala CREATE TABLE LIKE PARQUET syntax to produce a table with columns that match the structure of an existing Parquet file, including complex type columns for nested data structures. Remember to include the STORED AS PARQUET clause in this case, because even with CREATE TABLE LIKE PARQUET, the default file format of the resulting table is still text.

You cannot use the CREATE TABLE AS SELECT syntax to create a table with nested type columns because the complex columns are omitted from the result set of an Impala SELECT * or SELECT *COL_NAME* query, and because Impala currently does not support writing Parquet files with complex type columns,

Note:

Once you have a table set up with complex type columns, use the DESCRIBE and SHOW CREATE TABLE statements to see the correct notation with < and > delimiters and comma and colon separators within the complex type definitions. If you do not have existing data with the same layout as the table, you can query the empty table to practice with the notation for the SELECT statement. In the SELECT list, you use dot notation and pseudocolumns such as ITEM, KEY, and VALUE for referring to items within the complex type columns. In the FROM clause, you use join notation to construct table aliases for any referenced ARRAY and MAP columns.

For example, when defining a table that holds contact information, you might represent phone numbers differently depending on the expected layout and relationships of the data, and how well you can predict those properties in advance.

Here are different ways that you might represent phone numbers in a traditional relational schema, with equivalent representations using complex types.

Traditional relational representation of phone numbers: single table

The traditional, simplest way to represent phone numbers in a relational table is to store all contact info in a single table, with all columns having scalar types, and each potential phone number represented as a separate column. In this example, each person can only have these 3 types of phone numbers. If the person does not have a particular kind of phone number, the corresponding column is NULL for that row.

```
CREATE TABLE contacts_fixed_phones
(
    id BIGINT
  , name STRING
  , address STRING
  , home_phone STRING
  , work_phone STRING
  , mobile_phone STRING
) STORED AS PARQUET;
```

An array of phone numbers

Using a complex type column to represent the phone numbers adds some extra flexibility. Now there could be an unlimited number of phone numbers. Because the array elements have an order but not symbolic names, you could decide in advance that phone_number[0] is the home number, [1] is the work number, [2] is the mobile number, and so on. (In subsequent examples, you will see how to create a more flexible naming scheme using other complex type variations, such as a MAP or an ARRAY where each element is a STRUCT.)

```
CREATE TABLE contacts_array_of_phones
(
    id BIGINT
  , name STRING
  , address STRING
  , phone_number ARRAY < STRING >
) STORED AS PARQUET;
```

A map of phone numbers

Another way to represent an arbitrary set of phone numbers is with a MAP column. With a MAP, each element is associated with a key value that you specify, which could be a numeric, string, or other scalar type. This example uses a STRING key to give each phone number a name, such as 'home' or 'mobile'. A query could filter the data based on the key values, or display the key values in reports.

```
CREATE TABLE contacts_unlimited_phones
(
  id BIGINT, name STRING, address STRING, phone_number MAP < S
TRING,STRING >
) STORED AS PARQUET;
```

Traditional relational representation of phone numbers: normalized tables

If you are an experienced database designer, you already know how to work around the limitations of the single-table schema from *Traditional relational representation of phone numbers: single table*. By normalizing the schema, with the phone numbers in their own table, you can associate an arbitrary set of phone numbers with each person, and associate additional details with each phone number, such as whether it is a home, work, or mobile phone.

The flexibility of this approach comes with some drawbacks. Reconstructing all the data for a particular person requires a join query, which might require performance tuning on Hadoop because the data from each table might be transmitted from a different host. Data management tasks such as backups and refreshing the data require dealing with multiple tables instead of a single table.

This example illustrates a traditional database schema to store contact info normalized across 2 tables. The fact table establishes the identity and basic information about person. A dimension table stores information only about phone numbers, using an ID value to associate each phone number with a person ID from the fact table. Each person can have 0, 1, or many phones; the categories are not restricted to a few predefined ones; and the phone table can contain as many columns as desired, to represent all sorts of details about each phone number.

```
CREATE TABLE fact_contacts (id BIGINT, name STRING, address STR
ING) STORED AS PARQUET;
CREATE TABLE dim_phones
(
    contact_id BIGINT
  , category STRING
  , international_code STRING
```

```
  , area_code STRING
  , exchange STRING
  , extension STRING
  , mobile BOOLEAN
  , carrier STRING
  , current BOOLEAN
  , service_start_date TIMESTAMP
  , service_end_date TIMESTAMP
)
STORED AS PARQUET;
```

Phone Numbers Represented as an Array of Structs

To represent a schema equivalent to the one from *Traditional relational representation of phone numbers: normalized tables* using complex types, this example uses an ARRAY where each array element is a STRUCT. As with the earlier complex type examples, each person can have an arbitrary set of associated phone numbers. Making each array element into a STRUCT lets us associate multiple data items with each phone number, and give a separate name and type to each data item. The STRUCT fields of the ARRAY elements reproduce the columns of the dimension table from the previous example.

You can do all the same kinds of queries with the complex type schema as with the normalized schema from the previous example. The advantages of the complex type design are in the areas of convenience and performance. Now your backup and ETL processes only deal with a single table. When a query uses a join to cross-reference the information about a person with their associated phone numbers, all the relevant data for each row resides in the same HDFS data block, meaning each row can be processed on a single host without requiring network transmission.

```
CREATE TABLE contacts_detailed_phones
(
  id BIGINT, name STRING, address STRING
    , phone ARRAY < STRUCT <
        category: STRING
      , international_code: STRING
      , area_code: STRING
      , exchange: STRING
      , extension: STRING
      , mobile: BOOLEAN
      , carrier: STRING
      , current: BOOLEAN
      , service_start_date: TIMESTAMP
      , service_end_date: TIMESTAMP
    >>
) STORED AS PARQUET;
```

SQL statements that support complex types

The Impala SQL statements that support complex types are currently CREATE_TABLE, ALTER_TABLE, DESCRIBE, LOAD_DATA, and SELECT. That is, currently Impala can create or alter tables containing complex type columns, examine the structure of a table containing complex type columns, import existing data files containing complex type columns into a table, and query Parquet/ORC tables containing complex types.

Impala currently cannot write new data files containing complex type columns. Therefore, although the SELECT statement works for queries involving complex type columns, you cannot use a statement form that writes data to complex type columns, such as CREATE TABLE AS SELECT or INSERT ... SELECT. To create data files containing complex type data, use the Hive INSERT statement, or another ETL mechanism such as MapReduce jobs, Spark jobs, Pig, and so on.

DDL statements and complex types

Column specifications for complex or nested types use < and > delimiters:

```
-- What goes inside the < > for an ARRAY is a single type, either
 a scalar or another
-- complex type (ARRAY, STRUCT, or MAP).
CREATE TABLE array_t
(
  id BIGINT,
  a1 ARRAY <STRING>,
  a2 ARRAY <BIGINT>,
  a3 ARRAY <TIMESTAMP>,
  a4 ARRAY <STRUCT <f1: STRING, f2: INT, f3: BOOLEAN>>
)
STORED AS PARQUET;

-- What goes inside the < > for a MAP is two comma-separated t
ypes specifying the types of the key-value pair:
-- a scalar type representing the key, and a scalar or complex
type representing the value.
CREATE TABLE map_t
(
  id BIGINT,
  m1 MAP <STRING, STRING>,
  m2 MAP <STRING, BIGINT>,
  m3 MAP <BIGINT, STRING>,
  m4 MAP <BIGINT, BIGINT>,
  m5 MAP <STRING, ARRAY <STRING>>
)
STORED AS PARQUET;

-- What goes inside the < > for a STRUCT is a comma-separated lis
t of fields, each field defined as
-- name:type. The type can be a scalar or a complex type. The fie
ld names for each STRUCT do not clash
-- with the names of table columns or fields in other STRUCTs. A
 STRUCT is most often used inside
-- an ARRAY or a MAP rather than as a top-level column.
CREATE TABLE struct_t
(
  id BIGINT,
  s1 STRUCT <f1: STRING, f2: BIGINT>,
  s2 ARRAY <STRUCT <f1: INT, f2: TIMESTAMP>>,
  s3 MAP <BIGINT, STRUCT <name: STRING, birthday: TIMESTAMP>>
)
STORED AS PARQUET;
```

Queries and complex types

The following example shows how referring directly to a complex type column returns an error, while SELECT * on the same table succeeds, but only retrieves the scalar columns.

Note: Many of the complex type examples refer to tables such as CUSTOMER and REGION adapted from the tables used in the TPC-H benchmark.

```
SELECT c_orders FROM customer LIMIT 1;
ERROR: AnalysisException: Expr 'c_orders' in select list returns
a complex type 'ARRAY<STRUCT<o_orderkey:BIGINT,o_orderstatus:STR
ING, ... l_receiptdate:STRING,l_shipinstruct:STRING,l_shipmode:S
TRING,l_comment:STRING>>>>'.
Only scalar types are allowed in the select list.

-- Original column has several scalar and one complex column.
```

```
DESCRIBE customer;
+--------------+-----------------------------------+
| name         | type                              |
+--------------+-----------------------------------+
| c_custkey    | bigint                            |
| c_name       | string                            |
...
| c_orders     | array<struct<                     |
|              |   o_orderkey:bigint,              |
|              |   o_orderstatus:string,           |
|              |   o_totalprice:decimal(12,2),     |
...
|              | >>                                |
+--------------+-----------------------------------+

-- When we SELECT * from that table, only the scalar columns come
 back in the result set.
CREATE TABLE select_star_customer STORED AS PARQUET AS SELECT *
 FROM customer;
+------------------------+
| summary                |
+------------------------+
| Inserted 150000 row(s) |
+------------------------+
-- The c_orders column, being of complex type, was not included
 in the SELECT * result set.
DESC select_star_customer;
+--------------+---------------+
| name         | type          |
+--------------+---------------+
| c_custkey    | bigint        |
| c_name       | string        |
| c_address    | string        |
| c_nationkey  | smallint      |
| c_phone      | string        |
| c_acctbal    | decimal(12,2) |
| c_mktsegment | string        |
| c_comment    | string        |
+--------------+---------------+
```

References to fields within STRUCT columns use dot notation. If the field name is unambiguous, you can omit qualifiers such as table name, column name, or even the ITEM or VALUE pseudocolumn names for STRUCT elements inside an ARRAY or a MAP.

```
SELECT id, address.city FROM customers WHERE address.zip = 94305;
```

References to elements within ARRAY columns use the ITEM pseudocolumn:

```
select r_name, r_nations.item.n_name from region, region.r_natio
ns limit 7;
+--------+----------------+
| r_name | item.n_name    |
+--------+----------------+
| EUROPE | UNITED KINGDOM |
| EUROPE | RUSSIA         |
| EUROPE | ROMANIA        |
| EUROPE | GERMANY        |
| EUROPE | FRANCE         |
| ASIA   | VIETNAM        |
| ASIA   | CHINA          |
+--------+----------------+
```

References to fields within MAP columns use the KEY and VALUE pseudocolumns. In this example, once the query establishes the alias MAP_FIELD for a MAP column with a STRING key and an INT value, the query can refer to MAP_FIELD.KEY and MAP_FIELD.VALUE, which have zero, one, or many instances for each row from the containing table.

```
DESCRIBE table_0;
+---------+-----------------------+
| name    | type                  |
+---------+-----------------------+
| field_0 | string                |
| field_1 | map<string,int>       |
...

SELECT field_0, map_field.key, map_field.value
  FROM table_0, table_0.field_1 AS map_field
WHERE length(field_0) = 1
LIMIT 10;
+---------+-----------+-------+
| field_0 | key       | value |
+---------+-----------+-------+
| b       | gshsgkvd  | NULL  |
| b       | twrtcxj6  | 18    |
| b       | 2vp5      | 39    |
| b       | fh0s      | 13    |
| v       | 2         | 41    |
| v       | 8b58mz    | 20    |
| v       | hw        | 16    |
| v       | 65l388pyt | 29    |
| v       | 03k68g91z | 30    |
| v       | r2hlg5b   | NULL  |
+---------+-----------+-------+
```

When complex types are nested inside each other, you use a combination of joins, pseudocolumn names, and dot notation to refer to specific fields at the appropriate level. This is the most frequent form of query syntax for complex columns, because the typical use case involves two levels of complex types, such as an ARRAY of STRUCT elements.

```
SELECT id, phone_numbers.area_code FROM contact_info_many_structs
 INNER JOIN contact_info_many_structs.phone_numbers phone_numbers
 LIMIT 3;
```

You can express relationships between ARRAY and MAP columns at different levels as joins. You include comparison operators between fields at the top level and within the nested type columns so that Impala can do the appropriate join operation.

Note: Many of the complex type examples refer to tables such as CUSTOMER and REGION adapted from the tables used in the TPC-H benchmark.

For example, the following queries work equivalently. They each return customer and order data for customers that have at least one order.

```
SELECT c.c_name, o.o_orderkey FROM customer c, c.c_orders o LIMIT
 5;
+--------------------+------------+
| c_name             | o_orderkey |
+--------------------+------------+
| Customer#000072578 | 558821     |
| Customer#000072578 | 2079810    |
| Customer#000072578 | 5768068    |
| Customer#000072578 | 1805604    |
```

```
| Customer#000072578 | 3436389    |
+--------------------+------------+

SELECT c.c_name, o.o_orderkey FROM customer c INNER JOIN c.c_o
rders o LIMIT 5;
+--------------------+------------+
| c_name             | o_orderkey |
+--------------------+------------+
| Customer#000072578 | 558821     |
| Customer#000072578 | 2079810    |
| Customer#000072578 | 5768068    |
| Customer#000072578 | 1805604    |
| Customer#000072578 | 3436389    |
+--------------------+------------+
```

The following query using an outer join returns customers that have orders, plus customers with no orders (no entries in the C_ORDERS array):

```
SELECT c.c_custkey, o.o_orderkey
  FROM customer c LEFT OUTER JOIN c.c_orders o
LIMIT 5;
+-----------+------------+
| c_custkey | o_orderkey |
+-----------+------------+
| 60210     | NULL       |
| 147873    | NULL       |
| 72578     | 558821     |
| 72578     | 2079810    |
| 72578     | 5768068    |
+-----------+------------+
```

The following query returns only customers that have no orders. (With LEFT ANTI JOIN or LEFT SEMI JOIN, the query can only refer to columns from the left-hand table, because by definition there is no matching information in the right-hand table.)

```
SELECT c.c_custkey, c.c_name
  FROM customer c LEFT ANTI JOIN c.c_orders o
LIMIT 5;
+-----------+--------------------+
| c_custkey | c_name             |
+-----------+--------------------+
| 60210     | Customer#000060210 |
| 147873    | Customer#000147873 |
| 141576    | Customer#000141576 |
| 85365     | Customer#000085365 |
| 70998     | Customer#000070998 |
+-----------+--------------------+
```

You can also perform correlated subqueries to examine the properties of complex type columns for each row in the result set.

Count the number of orders per customer. Note the correlated reference to the table alias C. The COUNT(*) operation applies to all the elements of the C_ORDERS array for the corresponding row, avoiding the need for a GROUP BY clause.

```
select c_name, howmany FROM customer c, (SELECT COUNT(*) howmany
 FROM c.c_orders) v limit 5;
+--------------------+---------+
| c_name             | howmany |
+--------------------+---------+
| Customer#000030065 | 15      |
| Customer#000065455 | 18      |
```

```
| Customer#000113644 | 21      |
| Customer#000111078 | 0       |
| Customer#000024621 | 0       |
+--------------------+---------+
```

Count the number of orders per customer, ignoring any customers that have not placed any orders:

```
SELECT c_name, howmany_orders
FROM
  customer c,
  (SELECT COUNT(*) howmany_orders FROM c.c_orders) subq1
WHERE howmany_orders > 0
LIMIT 5;
+--------------------+----------------+
| c_name             | howmany_orders |
+--------------------+----------------+
| Customer#000072578 | 7              |
| Customer#000046378 | 26             |
| Customer#000069815 | 11             |
| Customer#000079058 | 12             |
| Customer#000092239 | 26             |
+--------------------+----------------+
```

Count the number of line items in each order. The reference to C.C_ORDERS in the FROM clause is needed because the O_ORDERKEY field is a member of the elements in the C_ORDERS array. The subquery labelled SUBQ1 is correlated: it is re-evaluated for the C_ORDERS.O_LINEITEMS array from each row of the CUSTOMERS table.

```
SELECT c_name, o_orderkey, howmany_line_items
FROM
  customer c,
  c.c_orders t2,
  (SELECT COUNT(*) howmany_line_items FROM c.c_orders.o_lineit
ems) subq1
WHERE howmany_line_items > 0
LIMIT 5;
+--------------------+------------+--------------------+
| c_name             | o_orderkey | howmany_line_items |
+--------------------+------------+--------------------+
| Customer#000020890 | 1884930    | 95                 |
| Customer#000020890 | 4570754    | 95                 |
| Customer#000020890 | 3771072    | 95                 |
| Customer#000020890 | 2555489    | 95                 |
| Customer#000020890 | 919171     | 95                 |
+--------------------+------------+--------------------+
```

Get the number of orders, the average order price, and the maximum items in any order per customer. For this example, the subqueries labelled SUBQ1 and SUBQ2 are correlated: they are re-evaluated for each row from the original CUSTOMER table, and only apply to the complex columns associated with that row.

```
SELECT c_name, howmany, average_price, most_items
FROM
  customer c,
  (SELECT COUNT(*) howmany, AVG(o_totalprice) average_price FROM
 c.c_orders) subq1,
  (SELECT MAX(l_quantity) most_items FROM c.c_orders.o_lineitem
s ) subq2
LIMIT 5;
+--------------------+---------+---------------+------------+
| c_name             | howmany | average_price | most_items |
+--------------------+---------+---------------+------------+
```

```
| Customer#000030065 | 15      | 128908.34     | 50.00      |
| Customer#000088191 | 0       | NULL          | NULL       |
| Customer#000101555 | 10      | 164250.31     | 50.00      |
| Customer#000022092 | 0       | NULL          | NULL       |
| Customer#000036277 | 27      | 166040.06     | 50.00      |
+--------------------+---------+---------------+------------+
```

For example, these queries show how to access information about the ARRAY elements within the CUSTOMER table from the “nested TPC-H” schema, starting with the initial ARRAY elements and progressing to examine the STRUCT fields of the ARRAY, and then the elements nested within another ARRAY of STRUCT:

```
-- How many orders does each customer have?
-- The type of the ARRAY column doesn't matter, this is just cou
nting the elements.
SELECT c_custkey, count(*)
  FROM customer, customer.c_orders
GROUP BY c_custkey
LIMIT 5;
+-----------+----------+
| c_custkey | count(*) |
+-----------+----------+
| 61081     | 21       |
| 115987    | 15       |
| 69685     | 19       |
| 109124    | 15       |
| 50491     | 12       |
+-----------+----------+

-- How many line items are part of each customer order?
-- Now we examine a field from a STRUCT nested inside the ARRAY.
SELECT c_custkey, c_orders.o_orderkey, count(*)
  FROM customer, customer.c_orders c_orders, c_orders.o_lineitems
GROUP BY c_custkey, c_orders.o_orderkey
LIMIT 5;
+-----------+------------+----------+
| c_custkey | o_orderkey | count(*) |
+-----------+------------+----------+
| 63367     | 4985959    | 7        |
| 53989     | 1972230    | 2        |
| 143513    | 5750498    | 5        |
| 17849     | 4857989    | 1        |
| 89881     | 1046437    | 1        |
+-----------+------------+----------+

-- What are the line items in each customer order?
-- One of the STRUCT fields inside the ARRAY is another
-- ARRAY containing STRUCT elements. The join finds
-- all the related items from both levels of ARRAY.
SELECT c_custkey, o_orderkey, l_partkey
  FROM customer, customer.c_orders, c_orders.o_lineitems
LIMIT 5;
+-----------+------------+-----------+
| c_custkey | o_orderkey | l_partkey |
+-----------+------------+-----------+
| 113644    | 2738497    | 175846    |
| 113644    | 2738497    | 27309     |
| 113644    | 2738497    | 175873    |
| 113644    | 2738497    | 88559     |
| 113644    | 2738497    | 8032      |
+-----------+------------+-----------+
```

Pseudocolumns for ARRAY and MAP types

Each element in an ARRAY type has a position, indexed starting from zero, and a value. Each element in a MAP type represents a key-value pair. Impala provides pseudocolumns that let you retrieve this metadata as part of a query, or filter query results by including such things in a WHERE clause. You refer to the pseudocolumns as part of qualified column names in queries:

- ITEM: The value of an array element. If the ARRAY contains STRUCT elements, you can refer to either *ARRAY_NAME*.ITEM.*FIELD_NAME* or use the shorthand *ARRAY_NAME*.*FIELD_NAME*.
- POS: The position of an element within an array.
- KEY: The value forming the first part of a key-value pair in a map. It is not necessarily unique.
- VALUE: The data item forming the second part of a key-value pair in a map. If the VALUE part of the MAP element is a STRUCT, you can refer to either *MAP_NAME*.VALUE.*FIELD_NAME* or use the shorthand *MAP_NAME*.*FIELD_NAME*.

ITEM and POS pseudocolumns

When an ARRAY column contains STRUCT elements, you can refer to a field within the STRU CT using a qualified name of the form *ARRAY_COLUMN*.*FIELD_NAME*. If the ARRAY contains scalar values, Impala recognizes the special name *ARRAY_COLUMN*.ITEM to represent the value of each scalar array element. For example, if a column contained an ARRAY where each element was a STRING, you would use *ARRAY_NAME*.ITEM to refer to each scalar value in the SELECT list, or the WHERE or other clauses.

This example shows a table with two ARRAY columns whose elements are of the scalar type STRI NG. When referring to the values of the array elements in the SELECT list, WHERE clause, or ORDER BY clause, you use the ITEM pseudocolumn because within the array, the individual elements have no defined names.

```
create TABLE persons_of_interest
(
person_id BIGINT,
aliases ARRAY <STRING>,
associates ARRAY <STRING>,
real_name STRING
)
STORED AS PARQUET;

-- Get all the aliases of each person.
SELECT real_name, aliases.ITEM
  FROM persons_of_interest, persons_of_interest.aliases
ORDER BY real_name, aliases.item;

-- Search for particular associates of each person.
SELECT real_name, associates.ITEM
  FROM persons_of_interest, persons_of_interest.associates
WHERE associates.item LIKE '% MacGuffin';
```

Because an array is inherently an ordered data structure, Impala recognizes the special name *ARRAY_COLUMN*.POS to represent the numeric position of each element within the array. The POS pseudocolumn lets you filter or reorder the result set based on the sequence of array elements.

The following example uses a table from a flattened version of the TPC-H schema. The REGION table only has a few rows, such as one row for Europe and one for Asia. The row for each region represents all the countries in that region as an ARRAY of STRUCT elements:

```
[localhost:21000] > desc region;
+-------------+-------------------------------------------------
--------------------+
```

```
| name        | type
                  |
+-------------+-------------------------------------------------
------------------+
| r_regionkey | smallint
                  |
| r_name      | string
                  |
| r_comment   | string
                  |
| r_nations   | array<struct<n_nationkey:smallint,n_name:string,n
_comment:string>> |
+-------------+-------------------------------------------------
-------------------+
```

To find the countries within a specific region, you use a join query. To find out the order of elements in the array, you also refer to the POS pseudocolumn in the select list:

```
[localhost:21000] > SELECT r1.r_name, r2.n_name, r2.POS
                  > FROM region r1 INNER JOIN r1.r_nations r2
                  > WHERE r1.r_name = 'ASIA';
+--------+-----------+-----+
| r_name | n_name    | pos |
+--------+-----------+-----+
| ASIA   | VIETNAM   | 0   |
| ASIA   | CHINA     | 1   |
| ASIA   | JAPAN     | 2   |
| ASIA   | INDONESIA | 3   |
| ASIA   | INDIA     | 4   |
+--------+-----------+-----+
```

Once you know the positions of the elements, you can use that information in subsequent queries, for example to change the ordering of results from the complex type column or to filter certain elements from the array:

```
[localhost:21000] > SELECT r1.r_name, r2.n_name, r2.POS
                  > FROM region r1 INNER JOIN r1.r_nations r2
                  > WHERE r1.r_name = 'ASIA'
                  > ORDER BY r2.POS DESC;
+--------+-----------+-----+
| r_name | n_name    | pos |
+--------+-----------+-----+
| ASIA   | INDIA     | 4   |
| ASIA   | INDONESIA | 3   |
| ASIA   | JAPAN     | 2   |
| ASIA   | CHINA     | 1   |
| ASIA   | VIETNAM   | 0   |
+--------+-----------+-----+
[localhost:21000] > SELECT r1.r_name, r2.n_name, r2.POS
                  > FROM region r1 INNER JOIN r1.r_nations r2
                  > WHERE r1.r_name = 'ASIA' AND r2.POS BETWEEN 1
 and 3;
+--------+-----------+-----+
| r_name | n_name    | pos |
+--------+-----------+-----+
| ASIA   | CHINA     | 1   |
| ASIA   | JAPAN     | 2   |
| ASIA   | INDONESIA | 3   |
+--------+-----------+-----+
```

KEY and VALUE pseudocolumns

The MAP data type is suitable for representing sparse or wide data structures, where each row might only have entries for a small subset of named fields. Because the element names (the map keys) vary depending on the row, a query must be able to refer to both the key and the value parts of each key-value pair. The KEY and VALUE pseudocolumns let you refer to the parts of the key-value pair independently within the query, as *MAP_COLUMN*.KEY and *MAP_COLUMN*.VALUE.

The KEY must always be a scalar type, such as STRING, BIGINT, or TIMESTAMP. It can be NULL. Values of the KEY field are not necessarily unique within the same MAP. You apply any required DISTINCT, GROUP BY, and other clauses in the query, and loop through the result set to process all the values matching any specified keys.

The VALUE can be either a scalar type or another complex type. If the VALUE is a STRUCT, you can construct a qualified name *MAP_COLUMN*.VALUE.*STRUCT_FIELD* to refer to the individual fields inside the value part. If the VALUE is an ARRAY or another MAP, you must include another join condition that establishes a table alias for *MAP_COLUMN*.VALUE, and then construct another qualified name using that alias, for example *TABLE_ALIAS*.ITEM or *TABLE_ALIAS*.KEY and *TABLE_ALIAS*.VALUE

The following example shows different ways to access a MAP column using the KEY and VALUE pseudocolumns. The DETAILS column has a STRING first part with short, standardized values such as 'Recurring', 'Lucid', or 'Anxiety'. This is the “key” that is used to look up particular kinds of elements from the MAP. The second part, also a STRING, is a longer free-form explanation. Impala gives you the standard pseudocolumn names KEY and VALUE for the two parts, and you apply your own conventions and interpretations to the underlying values.

Note: If you find that the single-item nature of the VALUE makes it difficult to model your data accurately, the solution is typically to add some nesting to the complex type. For example, to have several sets of key-value pairs, make the column an ARRAY whose elements are MAP. To make a set of key-value pairs that holds more elaborate information, make a MAP column whose VALUE part contains an ARRAY or a STRUCT.

```
CREATE TABLE dream_journal
(
  dream_id BIGINT,
  details MAP <STRING,STRING>
)
STORED AS PARQUET;

-- What are all the types of dreams that are recorded?
SELECT DISTINCT details.KEY FROM dream_journal, dream_journal.de
tails;

-- How many lucid dreams were recorded?
-- Because there is no GROUP BY, we count the 'Lucid' keys across
 all rows.
SELECT COUNT(details.KEY)
  FROM dream_journal, dream_journal.details
WHERE details.KEY = 'Lucid';

-- Print a report of a subset of dreams, filtering based on both
the lookup key
-- and the detailed value.
SELECT dream_id, details.KEY AS "Dream Type", details.VALUE AS
 "Dream Summary"
  FROM dream_journal, dream_journal.details
WHERE
  details.KEY IN ('Happy', 'Pleasant', 'Joyous')
  AND details.VALUE LIKE '%childhood%';
```

The following example shows a more elaborate version of the previous table, where the VALUE part of the MAP entry is a STRUCT rather than a scalar type. Now instead of referring to the VALUE pseudocolumn directly, you use dot notation to refer to the STRUCT fields inside it.

```
CREATE TABLE better_dream_journal
(
  dream_id BIGINT,
  details MAP <STRING,STRUCT <summary: STRING, when_happened: TIM
ESTAMP, duration: DECIMAL(5,2), woke_up: BOOLEAN> >
)
STORED AS PARQUET;


-- Do more elaborate reporting and filtering by examining multipl
e attributes within the same dream.
SELECT dream_id, details.KEY AS "Dream
 Type", details.VALUE.summary AS "Dream
 Summary", details.VALUE.duration AS "Duration"
  FROM better_dream_journal, better_dream_journal.details
WHERE
  details.KEY IN ('Anxiety', 'Nightmare')
  AND details.VALUE.duration > 60
  AND details.VALUE.woke_up = TRUE;

-- Remember that if the ITEM or VALUE contains a STRUCT, you can
reference
-- the STRUCT fields directly without the .ITEM or .VALUE qual
ifier.
SELECT dream_id, details.KEY AS "Dream Type", details.summary AS
 "Dream Summary", details.duration AS "Duration"
  FROM better_dream_journal, better_dream_journal.details
WHERE
  details.KEY IN ('Anxiety', 'Nightmare')
  AND details.duration > 60
  AND details.woke_up = TRUE;
```

Loading data containing complex types

Because the Impala INSERT statement does not currently support creating new data with complex type columns, or copying existing complex type values from one table to another, you primarily use Impala to query Parquet/ORC tables with complex types where the data was inserted through Hive, or create tables with complex types where you already have existing Parquet/ORC data files.

Because the Impala INSERT statement does not currently support creating new data with complex type columns, or copying existing complex type values from one table to another, you primarily use Impala to query Parquet/ORC tables with complex types where the data was inserted through Hive, or create tables with complex types where you already have existing Parquet/ORC data files.

If you have existing Parquet data files containing complex types, located outside of any Impala or Hive table, such as data files created by Spark jobs, you can use an Impala CREATE TABLE ... STORED AS PARQUET statement, followed by an Impala LOAD DATA statement to move the data files into the table. As an alternative, you can use an Impala CREATE EXTERNAL TABLE statement to create a table pointing to the HDFS directory that already contains the Parquet or ORC data files.

The simplest way to get started with complex type data is to take a denormalized table containing duplicated values, and use an INSERT ... SELECT statement to copy the data into a Parquet table and condense the repeated values into complex types. With the Hive INSERT statement, you use the COLLECT_LIST(), NAMED_STRUCT(), and MAP() constructor functions within a GROUP BY query to produce the complex type values. COLLECT_LIST() turns a sequence of values into an ARRAY. NAMED_STRUCT() uses the first, third, and so on arguments as the field names for a STRUCT, to match the field names from the CREATE TABLE statement.

Note: Because Hive currently cannot construct individual rows using complex types through the INSERT ... VALUES syntax, you prepare the data in flat form in a separate table, then copy it to the table with complex columns using INSERT ... S ELECT and the complex type constructors.

Using complex types as nested types

The ARRAY, STRUCT, and MAP types can be the top-level types for “nested type” columns. That is, each of these types can contain other complex or scalar types, with multiple levels of nesting to a maximum depth of 100. For example, you can have an array of structures, a map containing other maps, a structure containing an array of other structures, and so on. At the lowest level, there are always scalar types making up the fields of a STRUCT, elements of an ARRAY, and keys and values of a MAP.

Schemas involving complex types typically use some level of nesting for the complex type columns.

For example, to model a relationship like a dimension table and a fact table, you typically use an ARRAY where each array element is a STRUCT. The STRUCT fields represent what would traditionally be columns in a separate joined table. It makes little sense to use a STRUCT as the top-level type for a column, because you could just make the fields of the STRUCT into regular table columns.

Perhaps the only use case for a top-level STRUCT would be to to allow STRUCT fields with the same name as columns to coexist in the same table. The following example shows how a table could have a column named ID, and two separate STRUCT fields also named ID. Because the STRUCT fields are always referenced using qualified names, the identical ID names do not cause a conflict.

```
CREATE TABLE struct_namespaces
(
  id BIGINT
  , s1 STRUCT < id: BIGINT, field1: STRING >
  , s2 STRUCT < id: BIGINT, when_happened: TIMESTAMP >
)
STORED AS PARQUET;

select id, s1.id, s2.id from struct_namespaces;
```

It is common to make the value portion of each key-value pair in a MAP a STRUCT, ARRAY of STRUCT, or other complex type variation. That way, each key in the MAP can be associated with a flexible and extensible data structure. The key values are not predefined ahead of time (other than by specifying their data type). Therefore, the MAP can accommodate a rapidly evolving schema, or sparse data structures where each row contains only a few data values drawn from a large set of possible choices.

Although you can use an ARRAY of scalar values as the top-level column in a table, such a simple array is typically of limited use for analytic queries. The only property of the array elements, aside from the element value, is the ordering sequence available through the POS pseudocolumn. To record any additional item about each array element, such as a TIMESTAMP or a symbolic name, you use an ARRAY of STRUCT rather than of scalar values.

If you are considering having multiple ARRAY or MAP columns, with related items under the same position in each ARRAY or the same key in each MAP, prefer to use a STRUCT to group all the related items into a single ARRAY or MAP. Doing so avoids the additional storage overhead and potential duplication of key values from having an extra complex type column. Also, because each ARRAY or MAP that you reference in the query SELECT list requires an additional join clause, minimizing the number of complex type columns also makes the query easier to read and maintain, relying more on dot notation to refer to the relevant fields rather than a sequence of join clauses.

For example, here is a table with several complex type columns all at the top level and containing only scalar types. To retrieve every data item for the row requires a separate join for each ARRAY or MAP column. The fields of the STRUCT can be referenced using dot notation, but there is no

real advantage to using the STRUCT at the top level rather than just making separate columns FIEL D1 and FIELD2.

```
CREATE TABLE complex_types_top_level
(
  id BIGINT,
  a1 ARRAY<INT>,
  a2 ARRAY<STRING>,
  s STRUCT<field1: INT, field2: STRING>,
-- Numeric lookup key for a string value.
  m1 MAP<INT,STRING>,
-- String lookup key for a numeric value.
  m2 MAP<STRING,INT>
)
STORED AS PARQUET;

describe complex_types_top_level;
+------+-----------------+
| name | type            |
+------+-----------------+
| id   | bigint          |
| a1   | array<int>      |
| a2   | array<string>   |
| s    | struct<         |
|      |   field1:int,   |
|      |   field2:string |
|      | >               |
| m1   | map<int,string> |
| m2   | map<string,int> |
+------+-----------------+
select
  id,
  a1.item,
  a2.item,
  s.field1,
  s.field2,
  m1.key,
  m1.value,
  m2.key,
  m2.value
from
  complex_types_top_level,
  complex_types_top_level.a1,
  complex_types_top_level.a2,
  complex_types_top_level.m1,
  complex_types_top_level.m2;
```

For example, here is a table with columns containing an ARRAY of STRUCT, a MAP where each key value is a STRUCT, and a MAP where each key value is an ARRAY of STRUCT.

```
CREATE TABLE nesting_demo
(
  user_id BIGINT,
  family_members ARRAY < STRUCT < name: STRING, email: STRING,
 date_joined: TIMESTAMP >>,
  foo map < STRING, STRUCT < f1: INT, f2: INT, f3: TIMESTAMP, f4:
 BOOLEAN >>,
  gameplay MAP < STRING , ARRAY < STRUCT <
    name: STRING, highest: BIGINT, lives_used: INT, total_spent:
 DECIMAL(16,2)
  >>>
)
```

```
STORED AS PARQUET;
```

The DESCRIBE statement rearranges the < and > separators and the field names within each STRUCT for easy readability:

```
DESCRIBE nesting_demo;
+----------------+-----------------------------+
| name           | type                        |
+----------------+-----------------------------+
| user_id        | bigint                      |
| family_members | array<struct<               |
|                |   name:string,              |
|                |   email:string,             |
|                |   date_joined:timestamp     |
|                | >>                          |
| foo            | map<string,struct<          |
|                |   f1:int,                   |
|                |   f2:int,                   |
|                |   f3:timestamp,             |
|                |   f4:boolean                |
|                | >>                          |
| gameplay       | map<string,array<struct<    |
|                |   name:string,              |
|                |   highest:bigint,           |
|                |   lives_used:int,           |
|                |   total_spent:decimal(16,2) |
|                | >>>                         |
+----------------+-----------------------------+
```

To query the complex type columns, you use join notation to refer to the lowest-level scalar values. If the value is an ARRAY element, the fully qualified name includes the ITEM pseudocolumn. If the value is inside a MAP, the fully qualified name includes the KEY or VALUE pseudocolumn. Each reference to a different ARRAY or MAP (even if nested inside another complex type) requires an additional join clause.

```
SELECT
-- The lone scalar field doesn't require any dot notation or join
 clauses.
    user_id
-- Retrieve the fields of a STRUCT inside an ARRAY.
-- The FAMILY_MEMBERS name refers to the FAMILY_MEMBERS table
alias defined later in the FROM clause.
  , family_members.item.name
  , family_members.item.email
  , family_members.item.date_joined
-- Retrieve the KEY and VALUE fields of a MAP, with the value bei
ng a STRUCT consisting of more fields.
-- The FOO name refers to the FOO table alias defined later in th
e FROM clause.
  , foo.key
  , foo.value.f1
  , foo.value.f2
  , foo.value.f3
  , foo.value.f4
-- Retrieve the KEY fields of a MAP, and expand the VALUE part in
to ARRAY items consisting of STRUCT fields.
-- The GAMEPLAY name refers to the GAMEPLAY table alias defined
later in the FROM clause (referring to the MAP item).
-- The GAME_N name refers to the GAME_N table alias defined la
ter in the FROM clause (referring to the ARRAY
-- inside the MAP item's VALUE part.)
  , gameplay.key
```

```
  , game_n.name
  , game_n.highest
  , game_n.lives_used
  , game_n.total_spent
FROM
    nesting_demo
  , nesting_demo.family_members AS family_members
  , nesting_demo.foo AS foo
  , nesting_demo.gameplay AS gameplay
  , nesting_demo.gameplay.value AS game_n;
```

Once you understand the notation to refer to a particular data item in the SELECT list, you can use the same qualified name to refer to that data item in other parts of the query, such as the WHERE clause, ORDER BY or GROUP BY clauses, or calls to built-in functions. For example, you might frequently retrieve the VALUE part of each MAP item in the SELECT list, while choosing the specific MAP items by running comparisons against the KEY part in the WHERE clause.

Allowing embedding complex types into other complex types

Before this release, collections and structs were supported in the select list, but nesting was only supported for the same kinds of complex types, i.e., structs in structs and collections in collections (including the case of maps in arrays and arrays in maps). Mixing different kinds of complex types with structs in collections or vice versa, was not supported. This release adds support for mixed complex types in the select list.

Limitations

- Zipping unnest is not supported for mixed complex types
- Structs containing collections are not (yet) supported in case of sorting

Note: Sorting of collections was not supported before this release. This release adds support for sorting collections that only contain fixed length types, but not for collections in structs. Sorting by the complex types is not supported yet.

Examples of querying a mixed complex type in the SELECT list

```
select
id,
struct_contains_arr,
struct_contains_map,
arr_contains_struct,
arr_contains_nested_struct,
struct_contains_nested_arr,
all_mix
from collection_struct_mix;
```

```
select id, nested_struct from complextypestbl;
```

Accessing complex type data in flattened form using views

The layout of complex and nested types is largely a physical consideration. The complex type columns reside in the same data files rather than in separate normalized tables, for your convenience in managing related data sets and performance in querying related data sets. You can use views to treat tables with complex types as if they were flattened. By putting the join logic and references to the complex type columns in the view definition, you can query the same tables using existing queries intended for tables containing only scalar columns. This technique also lets you use tables with complex types with BI tools that are not aware of the data types and query notation for accessing complex type columns.

For example, the variation of the TPC-H schema containing complex types has a table REGION. This table has 5 rows, corresponding to 5 regions such as NORTH AMERICA and AFRICA. Each

row has an ARRAY column, where each array item is a STRUCT containing details about a country in that region.

```
DESCRIBE region;
+-------------+-------------------------+
| name        | type                    |
+-------------+-------------------------+
| r_regionkey | smallint                |
| r_name      | string                  |
| r_comment   | string                  |
| r_nations   | array<struct<           |
|             |   n_nationkey:smallint, |
|             |   n_name:string,        |
|             |   n_comment:string      |
|             | >>                      |
+-------------+-------------------------+
```

The same data could be represented in traditional denormalized form, as a single table where the information about each region is repeated over and over, alongside the information about each country. The nested complex types let us avoid the repetition, while still keeping the data in a single table rather than normalizing across multiple tables.

To use this table with a JDBC or ODBC application that expected scalar columns, we could create a view that represented the result set as a set of scalar columns (three columns from the original table, plus three more from the STRUCT fields of the array elements). In the following examples, any column with an R_* prefix is taken unchanged from the original table, while any column with an N_* prefix is extracted from the STRUCT inside the ARRAY.

```
CREATE VIEW region_view AS
  SELECT
    r_regionkey,
    r_name,
    r_comment,
    array_field.item.n_nationkey AS n_nationkey,
    array_field.item.n_name AS n_name,
    array_field.n_comment AS n_comment
FROM
  region, region.r_nations AS array_field;
```

Then we point the application queries at the view rather than the original table. From the perspective of the view, there are 25 rows in the result set, one for each nation in each region, and queries can refer freely to fields related to the region or the nation.

```
-- Retrieve info such as the nation name from the original R_NAT
IONS array elements.
select n_name from region_view where r_name in ('EUROPE', 'ASIA
');
+----------------+
| n_name         |
+----------------+
| UNITED KINGDOM |
| RUSSIA         |
| ROMANIA        |
| GERMANY        |
| FRANCE         |
| VIETNAM        |
| CHINA          |
| JAPAN          |
| INDONESIA      |
| INDIA          |
+----------------+
```

```
-- UNITED STATES in AMERICA and UNITED KINGDOM in EUROPE.
SELECT DISTINCT r_name FROM region_view WHERE n_name LIKE 'UNITED
%';
+---------+
| r_name  |
+---------+
| AMERICA |
| EUROPE  |
+---------+

-- For conciseness, we only list some view columns in the SELECT
list.
-- SELECT * would bring back all the data, unlike SELECT *
-- queries on the original table with complex type columns.
SELECT r_regionkey, r_name, n_nationkey, n_name FROM region_view
 LIMIT 7;
+-------------+--------+-------------+----------------+
| r_regionkey | r_name | n_nationkey | n_name         |
+-------------+--------+-------------+----------------+
| 3           | EUROPE | 23          | UNITED KINGDOM |
| 3           | EUROPE | 22          | RUSSIA         |
| 3           | EUROPE | 19          | ROMANIA        |
| 3           | EUROPE | 7           | GERMANY        |
| 3           | EUROPE | 6           | FRANCE         |
| 2           | ASIA   | 21          | VIETNAM        |
| 2           | ASIA   | 18          | CHINA          |
+-------------+--------+-------------+----------------+
```

Tutorials and examples for complex types

The following examples illustrate the query syntax for some common use cases involving complex type columns:

Sample schema and data for experimenting with Impala complex types

The tables used for earlier examples of complex type syntax are trivial ones with no actual data. The more substantial examples of the complex type feature use these tables, adapted from the schema used for TPC-H testing:

```
SHOW TABLES;
+----------+
| name     |
+----------+
| customer |
| part     |
| region   |
| supplier |
+----------+

DESCRIBE customer;
+--------------+-----------------------------------+
| name         | type                              |
+--------------+-----------------------------------+
| c_custkey    | bigint                            |
| c_name       | string                            |
| c_address    | string                            |
| c_nationkey  | smallint                          |
| c_phone      | string                            |
| c_acctbal    | decimal(12,2)                     |
| c_mktsegment | string                            |
| c_comment    | string                            |
| c_orders     | array<struct<                     |
|              |   o_orderkey:bigint,              |
```

```
|               |   o_orderstatus:string,              |
|               |   o_totalprice:decimal(12,2),        |
|               |   o_orderdate:string,                |
|               |   o_orderpriority:string,            |
|               |   o_clerk:string,                    |
|               |   o_shippriority:int,                |
|               |   o_comment:string,                  |
|               |   o_lineitems:array<struct<          |
|               |     l_partkey:bigint,                |
|               |     l_suppkey:bigint,                |
|               |     l_linenumber:int,                |
|               |     l_quantity:decimal(12,2),        |
|               |     l_extendedprice:decimal(12,2),   |
|               |     l_discount:decimal(12,2),        |
|               |     l_tax:decimal(12,2),             |
|               |     l_returnflag:string,             |
|               |     l_linestatus:string,             |
|               |     l_shipdate:string,               |
|               |     l_commitdate:string,             |
|               |     l_receiptdate:string,            |
|               |     l_shipinstruct:string,           |
|               |     l_shipmode:string,               |
|               |     l_comment:string                 |
|               |   >>                                 |
|               | >>                                   |
+---------------+--------------------------------------+

DESCRIBE part;
+---------------+---------------+
| name          | type          |
+---------------+---------------+
| p_partkey     | bigint        |
| p_name        | string        |
| p_mfgr        | string        |
| p_brand       | string        |
| p_type        | string        |
| p_size        | int           |
| p_container   | string        |
| p_retailprice | decimal(12,2) |
| p_comment     | string        |
+---------------+---------------+

DESCRIBE region;
+-------------+-------------------------------------------------------
------------------+
| name        | type
                  |
+-------------+-----------------------------------------------------
--------------------+
| r_regionkey | smallint
                  |
| r_name      | string
                  |
| r_comment   | string
                  |
| r_nations   | array<struct<n_nationkey:smallint,n_name:string
,n_comment:string>> |
+-------------+-----------------------------------------------
---------------------+

DESCRIBE supplier;
+-------------+-----------------------------------------------+
| name        | type                                          |
+-------------+-----------------------------------------------+
```

```
| s_suppkey   | bigint                                          |
| s_name      | string                                          |
| s_address   | string                                          |
| s_nationkey | smallint                                        |
| s_phone     | string                                          |
| s_acctbal   | decimal(12,2)                                   |
| s_comment   | string                                          |
| s_partsupps | array<struct<ps_partkey:bigint,                 |
|             | ps_availqty:int,ps_supplycost:decimal(12,2), |
|             | ps_comment:string>>                             |
+-------------+-------------------------------------------------+
```

The volume of data used in the following examples is:

```
SELECT count(*) FROM customer;
+----------+
| count(*) |
+----------+
| 150000   |
+----------+
SELECT count(*) FROM part;
+----------+
| count(*) |
+----------+
| 200000   |
+----------+

SELECT count(*) FROM region;
+----------+
| count(*) |
+----------+
| 5        |
+----------+
SELECT count(*) FROM supplier;
+----------+
| count(*) |
+----------+
| 10000    |
+----------+
```

Constructing Parquet / ORC files with complex columns using Hive

The following examples demonstrate the Hive syntax to transform flat data (tables with all scalar columns) into Parquet/ORC tables where Impala can query the complex type columns. Each example shows the full sequence of steps, including switching back and forth between Impala and Hive. Although the source table can use any file format, the destination table must use the Parquet/ORC file format. We take Parquet in the following examples. You can replace Parquet with ORC to do the same things in ORC file format.

Create table with ARRAY in Impala, load data in Hive, query in Impala:

This example shows the cycle of creating the tables and querying the complex data in Impala, and using Hive (either the hive shell or beeline) for the data loading step. The data starts in flattened, denormalized form in a text table. Hive writes the corresponding Parquet data, including an ARRAY column. Then Impala can run analytic queries on the Parquet table, using join notation to unpack the ARRAY column.

```
/* Initial DDL and loading of flat, denormalized data happens in
 impala-shell */CREATE TABLE flat_array (country STRING, city ST
RING);INSERT INTO flat_array VALUES
```

```
    ('Canada', 'Toronto') , ('Canada', 'Vancouver') , ('Canada',
 "St. John\'s")
  , ('Canada', 'Saint John') , ('Canada', 'Montreal') , ('Canada
', 'Halifax')
  , ('Canada', 'Winnipeg') , ('Canada', 'Calgary') , ('Canada',
 'Saskatoon')
  , ('Canada', 'Ottawa') , ('Canada', 'Yellowknife') , ('France',
 'Paris')
  , ('France', 'Nice') , ('France', 'Marseilles') , ('France',
 'Cannes')
  , ('Greece', 'Athens') , ('Greece', 'Piraeus') , ('Greece', 'Ha
nia')
  , ('Greece', 'Heraklion') , ('Greece', 'Rethymnon') , ('Gree
ce', 'Fira');

CREATE TABLE complex_array (country STRING, city ARRAY <STRING>)
STORED AS PARQUET;
```

```
/* Conversion to Parquet and complex and/or nested columns happe
ns in Hive */

INSERT INTO complex_array SELECT country, collect_list(city) FROM
 flat_array GROUP BY country;
Query ID = dev_20151108160808_84477ff2-82bd-4ba4-9a77-554fa7b8
c0cb
Total jobs = 1
Launching Job 1 out of 1
...
```

```
/* Back to impala-shell again for analytic queries */

REFRESH complex_array;
SELECT country, city.item FROM complex_array, complex_array.city
+---------+-------------+
| country | item        |
+---------+-------------+
| Canada  | Toronto     |
| Canada  | Vancouver   |
| Canada  | St. John's  |
| Canada  | Saint John  |
| Canada  | Montreal    |
| Canada  | Halifax     |
| Canada  | Winnipeg    |
| Canada  | Calgary     |
| Canada  | Saskatoon   |
| Canada  | Ottawa      |
| Canada  | Yellowknife |
| France  | Paris       |
| France  | Nice        |
| France  | Marseilles  |
| France  | Cannes      |
| Greece  | Athens      |
| Greece  | Piraeus     |
| Greece  | Hania       |
| Greece  | Heraklion   |
| Greece  | Rethymnon   |
| Greece  | Fira        |
+---------+-------------+
```

Create table with STRUCT and ARRAY in Impala, load data in Hive, query in Impala:

This example shows the cycle of creating the tables and querying the complex data in Impala, and using Hive (either the hive shell or beeline) for the data loading step. The data starts in flattened, denormalized form in a text table. Hive writes the corresponding Parquet data, including a STRUCT column with an ARRAY field. Then Impala can run analytic queries on the Parquet table, using join notation to unpack the ARRAY field from the STRUCT column.

```
/* Initial DDL and loading of flat, denormalized data happens in
 impala-shell */

CREATE TABLE flat_struct_array (continent STRING, country STRI
NG, city STRING);
INSERT INTO flat_struct_array VALUES
    ('North America', 'Canada', 'Toronto') , ('North America', 'C
anada', 'Vancouver')
  , ('North America', 'Canada', "St. John\'s") , ('North Americ
a', 'Canada', 'Saint John')
  , ('North America', 'Canada', 'Montreal') , ('North America',
'Canada', 'Halifax')
  , ('North America', 'Canada', 'Winnipeg') , ('North America',
 'Canada', 'Calgary')
  , ('North America', 'Canada', 'Saskatoon') , ('North America',
 'Canada', 'Ottawa')
  , ('North America', 'Canada', 'Yellowknife') , ('Europe', 'Fran
ce', 'Paris')
  , ('Europe', 'France', 'Nice') , ('Europe', 'France', 'Marse
illes')
  , ('Europe', 'France', 'Cannes') , ('Europe', 'Greece', 'Athen
s')
  , ('Europe', 'Greece', 'Piraeus') , ('Europe', 'Greece', 'Hani
a')
  , ('Europe', 'Greece', 'Heraklion') , ('Europe', 'Greece', 'Re
thymnon')
  , ('Europe', 'Greece', 'Fira');
CREATE TABLE complex_struct_array (continent STRING, country ST
RUCT <name: STRING, city: ARRAY <STRING> >) STORED AS PARQUET;
```

```
/* Conversion to Parquet and complex and/or nested columns happe
ns in Hive */

INSERT INTO complex_struct_array SELECT continent, named_struct('
name', country, 'city', collect_list(city)) FROM flat_array_array
 GROUP BY continent, country;
Query ID = dev_20151108163535_11a4fa53-0003-4638-97e6-ef13cdb8e09
e
Total jobs = 1
Launching Job 1 out of 1
...
```

```
/* Back to impala-shell again for analytic queries */

REFRESH complex_struct_array;
SELECT t1.continent, t1.country.name, t2.item
  FROM complex_struct_array t1, t1.country.city t2
+---------------+--------------+-------------+
| continent     | country.name | item        |
+---------------+--------------+-------------+
| Europe        | France       | Paris       |
| Europe        | France       | Nice        |
```

```
| Europe        | France       | Marseilles  |
| Europe        | France       | Cannes      |
| Europe        | Greece       | Athens      |
| Europe        | Greece       | Piraeus     |
| Europe        | Greece       | Hania       |
| Europe        | Greece       | Heraklion   |
| Europe        | Greece       | Rethymnon   |
| Europe        | Greece       | Fira        |
| North America | Canada       | Toronto     |
| North America | Canada       | Vancouver   |
| North America | Canada       | St. John's  |
| North America | Canada       | Saint John  |
| North America | Canada       | Montreal    |
| North America | Canada       | Halifax     |
| North America | Canada       | Winnipeg    |
| North America | Canada       | Calgary     |
| North America | Canada       | Saskatoon   |
| North America | Canada       | Ottawa      |
| North America | Canada       | Yellowknife |
+---------------+--------------+-------------+
```

Flattening normalized tables into a single table with complex types

One common use for complex types is to embed the contents of one table into another. The traditional technique of denormalizing results in a huge number of rows with some column values repeated over and over. With complex types, you can keep the same number of rows as in the original normalized table, and put all the associated data from the other table in a single new column.

In this flattening scenario, you might frequently use a column that is an ARRAY consisting of STRUCT elements, where each field within the STRUCT corresponds to a column name from the table that you are combining.

The following example shows a traditional normalized layout using two tables, and then an equivalent layout using complex types in a single table.

```
/* Traditional relational design */
-- This table just stores numbers, allowing us to look up details
 about the employee
-- and details about their vacation time using a three-table j
oin query.
CREATE table employee_vacations
(
  employee_id BIGINT,
  vacation_id BIGINT
)
STORED AS PARQUET;

-- Each kind of information to track gets its own "fact table".
CREATE table vacation_details
(
  vacation_id BIGINT,
  vacation_start TIMESTAMP,
  duration INT
)
STORED AS PARQUET;
-- Any time we print a human-readable report, we join with this
 table to
-- display info about employee #1234.
CREATE TABLE employee_contact
(
  employee_id BIGINT,
```

```
  name STRING,
  address STRING,
  phone STRING,
  email STRING,
  address_type STRING /* 'home', 'work', 'remote', etc. */
)
STORED AS PARQUET;

/* Equivalent flattened schema using complex types */

-- For analytic queries using complex types, we can bundle the di
mension table
-- and multiple fact tables into a single table.
CREATE TABLE employee_vacations_nested_types
(
-- We might still use the employee_id for other join queries.
-- The table needs at least one scalar column to serve as an ide
ntifier
-- for the complex type columns.
  employee_id BIGINT,

-- Columns of the VACATION_DETAILS table are folded into a STRUC
T.
-- We drop the VACATION_ID column because Impala doesn't need
-- synthetic IDs to join a complex type column.
-- Each row from the VACATION_DETAILS table becomes an array el
ement.
  vacation ARRAY < STRUCT <
    vacation_start: TIMESTAMP,
    duration: INT
  >>,

-- The ADDRESS_TYPE column, with a small number of predefined val
ues that are distinct
-- for each employee, makes the EMPLOYEE_CONTACT table a good
candidate to turn into a MAP,
-- with each row represented as a STRUCT. The string value from
 ADDRESS_TYPE becomes the
-- "key" (the anonymous first field) of the MAP.
  contact MAP < STRING, STRUCT <
    address: STRING,
    phone: STRING,
    email: STRING
  >>
)
STORED AS PARQUET;
```

Interchanging complex type tables and data files with Hive and other components

You can produce Parquet data files through several Hadoop components and APIs.

If you have a Hive-created Parquet table that includes ARRAY, STRUCT, or MAP columns, Impala can query that same table in Impala 2.3 and higher, subject to the usual restriction that all other columns are of data types supported by Impala, and also that the file type of the table must be Parquet.

If you have a Parquet data file produced outside of Impala, Impala can automatically deduce the appropriate table structure using the syntax CREATE TABLE ... LIKE PARQUET

'*HDFS_PATH_OF_PARQUET_FILE*'. In Impala 2.3 and higher, this feature works for Parquet files that include ARRAY, STRUCT, or MAP types.

```
/* In impala-shell, find the HDFS data directory of the original
 table.
DESCRIBE FORMATTED tpch_nested_parquet.customer;
...
| Location: | hdfs://localhost:20500/test-warehouse/tpch_nested
_parquet.db/customer   | NULL |
...
# In the Unix shell, find the path of any Parquet data file in
that HDFS directory.
$ hdfs dfs -ls hdfs://localhost:20500/test-warehouse/tpch_neste
d_parquet.db/customer
Found 4 items
-rwxr-xr-x   3 dev supergroup  171298918 2015-09-22 23:30 hdfs://
localhost:20500/blah/tpch_nested_parquet.db/customer/000000_0
...

/* Back in impala-shell, use the HDFS path in a CREATE TABLE LIKE
 PARQUET statement. */
CREATE TABLE customer_ctlp
  LIKE PARQUET 'hdfs://localhost:20500/blah/tpch_nested_parquet.d
b/customer/000000_0'
  STORED AS PARQUET;

/* Confirm that old and new tables have the same column layout, i
ncluding complex types. */
DESCRIBE tpch_nested_parquet.customer
+--------------+-----------------------------------+---------+
| name         | type                              | comment |
+--------------+-----------------------------------+---------+
| c_custkey    | bigint                            |         |
| c_name       | string                            |         |
| c_address    | string                            |         |
| c_nationkey  | smallint                          |         |
| c_phone      | string                            |         |
| c_acctbal    | decimal(12,2)                     |         |
| c_mktsegment | string                            |         |
| c_comment    | string                            |         |
| c_orders     | array<struct<                     |         |
|              |   o_orderkey:bigint,              |         |
|              |   o_orderstatus:string,           |         |
|              |   o_totalprice:decimal(12,2),     |         |
|              |   o_orderdate:string,             |         |
|              |   o_orderpriority:string,         |         |
|              |   o_clerk:string,                 |         |
|              |   o_shippriority:int,             |         |
|              |   o_comment:string,               |         |
|              |   o_lineitems:array<struct<       |         |
|              |     l_partkey:bigint,             |         |
|              |     l_suppkey:bigint,             |         |
|              |     l_linenumber:int,             |         |
|              |     l_quantity:decimal(12,2),     |         |
|              |     l_extendedprice:decimal(12,2), |         |
|              |     l_discount:decimal(12,2),     |         |
|              |     l_tax:decimal(12,2),          |         |
|              |     l_returnflag:string,          |         |
|              |     l_linestatus:string,          |         |
|              |     l_shipdate:string,            |         |
|              |     l_commitdate:string,          |         |
|              |     l_receiptdate:string,         |         |
|              |     l_shipinstruct:string,        |         |
```

```
|              |     l_shipmode:string,             |         |
|              |     l_comment:string               |         |
|              |   >>                               |         |
|              | >>                                 |         |
+--------------+------------------------------------+---------+

describe customer_ctlp;
+--------------+------------------------------------+---------
--------------------+
| name         | type                               | comment
                |
+--------------+------------------------------------+---------
--------------------+
| c_custkey    | bigint                             | Inferred
 from Parquet file. |
| c_name       | string                             | Inferred
 from Parquet file. |
| c_address    | string                             | Inferred
 from Parquet file. |
| c_nationkey  | int                                | Inferred
 from Parquet file. |
| c_phone      | string                             | Inferred
 from Parquet file. |
| c_acctbal    | decimal(12,2)                      | Inferred
 from Parquet file. |
| c_mktsegment | string                             | Inferred
 from Parquet file. |
| c_comment    | string                             | Inferred
 from Parquet file. |
| c_orders     | array<struct<                      | Inferred
 from Parquet file. |
|              |   o_orderkey:bigint,               |
                |
|              |   o_orderstatus:string,            |
                |
|              |   o_totalprice:decimal(12,2),      |
                |
|              |   o_orderdate:string,              |
                |
|              |   o_orderpriority:string,          |
                |
|              |   o_clerk:string,                  |
                |
|              |   o_shippriority:int,              |
                |
|              |   o_comment:string,                |
                |
|              |   o_lineitems:array<struct<        |
                |
|              |     l_partkey:bigint,              |
                |
|              |     l_suppkey:bigint,              |
                |
|              |     l_linenumber:int,              |
                |
|              |     l_quantity:decimal(12,2),      |
                |
|              |     l_extendedprice:decimal(12,2), |
                |
|              |     l_discount:decimal(12,2),      |
                |
|              |     l_tax:decimal(12,2),           |
                |
```

```
|               |     l_returnflag:string,           |
                |
|               |     l_linestatus:string,           |
                |
|               |     l_shipdate:string,             |
                |
|               |     l_commitdate:string,           |
                |
|               |     l_receiptdate:string,          |
                |
|               |     l_shipinstruct:string,         |
                |
|               |     l_shipmode:string,             |
                |
|               |     l_comment:string               |
                |
|               |   >>                               |
                |
|               | >>                                 |
                |
+---------------+------------------------------------+---------
--------------------+
```

Querying arrays

Describes how to use UNNEST function to query arrays. ARRAY data types represent collections with arbitrary numbers of elements, where each element is the same type.

Querying arrays using JOIN and UNNEST

You can query arrays by making a join between the table and the array inside the table. This approach is improved with the introduction of the UNNEST function in the SELECT list or in the FROM clause in the SELECT statement. When you use UNNEST, you can provide more than one array in the SELECT statement. If you use JOINs for querying arrays it will yield a “joining unnest” however the latter will provide a “zipping unnest”.

Example of querying arrays using JOIN

Use JOIN in cases where you must join unnest of multiple arrays. However if you must zip unnest then use the newly implemented UNNEST function.

Here is an example of a SELECT statement that uses JOINs to query an array.

```
SELECT id, arr1.item, arr2.item FROM tbl_name tbl, tbl.arr1, tbl.arr2;
ID, ARR1.ITEM, ARR2.ITEM
```

```
[1, 1, 10]
[1, 1, 11]
[1, 2, 10]
[1, 2, 11]
[1, 3, 10]
[1, 3, 11]
```

Note:

The test data used in this example is ID: 1, arr1: {1, 2, 3}, arr2: {10, 11}

Examples of querying arrays using UNNEST

You can use one of the two different syntaxes shown here to unnest multiple arrays in one query. This results in the items of the arrays being zipped together instead of joining.

- ISO:SQL 2016 compliant syntax:

```
SELECT a1.item, a2.item
FROM complextypes_arrays t, UNNEST(t.arr1, t.arr2) AS (a1, a2);
```

- Postgres compatible syntax:

```
SELECT UNNEST(arr1), UNNEST(arr2) FROM complextypes_arrays;
```

Unnest operator in SELECT list

```
SELECT id, unnest(arr1), unnest(arr2) FROM tbl_name;
```

Unnest operator in FROM clause

```
SELECT id, arr1.item, arr2.item FROM tbl_name tbl_alias, UNNEST(tbl_alias.ar
r1, tbl_alias.arr2);
```

This new functionality would zip the arrays next to each other as shown here.

```
ID, ARR1.ITEM, ARR2.ITEM
[1, 1, 10]
[1, 2, 11]
[1, 3, NULL]
```

Note, that arr2 is shorter than arr1 so the "missing" items in its column will be filled with NULLs.

Note: The test data used in this example is ID: 1, arr1: {1, 2, 3}, arr2: {10, 11}

Limitations in Using UNNEST

- Only arrays from the same table can be zipping unnested
- The old (joining) and the new (zipping) unnests cannot be used together
- You can add a WHERE filter on an unnested item only if you add a wrapper SELECT and do the filtering

  Example:

```
SELECT id, arr1_unnest FROM (SELECT id, unnest(arr1) as arr1_unnest FROM
 tbl_name) WHERE arr1_unnest < 10;
```

Using ARRAY columns in the SELECT list

Prior to this release to look into the content of an array you had to unnest the array either by the joining syntax or by using the zipping UNNEST operator as shown in the following example:

```
SELECT unnest(IDs), unnest(NAMES) FROM table_name;
```

This release adds support to return ARRAYs as STRINGs (JSON arrays) in the SELECT list, for example:

```
select id, int_array from functional_parquet.complextypestbl where id = 1;
returns: 1, “[1,2,3]”
```

Returning ARRAYs from inline or HMS views is also supported. These arrays can be used both in the select list or as relative table references.

```
select id, int_array from (select id, int_array from complextypestbl) s;
```

Though STRUCTs are already supported, ARRAYs and STRUCTs nested within each other are not supported yet. Using them as non-relative table references is also not supported yet.

Zipping unnest on arrays from views

Before this release, zipping unnest functionality worked for arrays only in "tables" but did not support "views" as a source. This release improves that support further by accepting inputs from views as well.

UNNEST() on array columns

You can use UNNEST() on array columns in two ways. Using one of these two ways results in the items of the arrays being zipped together instead of joining.

- ISO:SQL 2016 compliant syntax

```
SELECT a1.item, a2.item
FROM complextypes_arrays t, UNNEST(t.arr1, t.arr2) AS (a1, a2);
```

- Postgres compatible syntax

```
SELECT UNNEST(arr1), UNNEST(arr2) FROM complextypes_arrays;
```

When unnesting multiple arrays with zipping unnest, the i'th item of one array will be placed next to the i'th item of the other arrays in the results. If the size of the arrays is not equal then the shorter arrays will be filled with NULL values up to the size of the longest array as shown in the following example:

The test data used in this example is arr1: {1, 2, 3}, arr2: {11, 12}

After running any of the queries listed in the examples, the result will be as shown here:

```
arr1 arr2
[1, 11]
[2, 12]
[3, NULL]
```

Example of an UNNEST() in an inline view using SELECT/FILTER of the inline view

In the following example the filter is not in the SELECT query that creates the inline view but a level above that.

```
SELECT id, ac1, ac2 FROM
    (SELECT id, UNNEST(array_col1) AS ac1, UNNEST(array_col2) AS ac2 FROM
some_view)
WHERE id <10;
```

Impala SQL literals

Each of the Impala data types has corresponding notation for literal values of that type. You specify literal values in SQL statements, such as in the SELECT list or WHERE clause of a query, or as an argument to a function call.

Numeric literals

To write literals for the integer types (TINYINT, SMALLINT, INT, and BIGINT), use a sequence of digits with optional leading zeros.

To write literals for the floating-point types (DECIMAL, FLOAT, and DOUBLE), use a sequence of digits with an optional decimal point (. character). To preserve accuracy during arithmetic expressions, Impala interprets floating-point literals as the DECIMAL type with the smallest appropriate precision and scale, until required by the context to convert the result to FLOAT or DOUBLE.

Integer values are promoted to floating-point when necessary, based on the context.

You can also use exponential notation by including an e character. For example, 1e6 is 1 times 10 to the power of 6 (1 million). A number in exponential notation is always interpreted as floating-point.

When Impala encounters a numeric literal, it considers the type to be the “smallest” that can accurately represent the value. The type is promoted to larger or more accurate types if necessary, based on subsequent parts of an expression.

STRING literals

String literals are quoted using either single or double quotation marks. You can use either kind of quotes for string literals, even both kinds for different literals within the same statement.

Quoted literals are considered to be of type STRING. To use quoted literals in contexts requiring a CHAR or VARC HAR value, CAST() the literal to a CHAR or VARCHAR of the appropriate length.

Escaping special characters:

To encode special characters within a string literal, precede them with the backslash (\) escape character:

- \t represents a tab.
- \n represents a newline or linefeed. This might cause extra line breaks in `impala-shell` output.
- \r represents a carriage return. This might cause unusual formatting (making it appear that some content is overwritten) in `impala-shell` output.
- \b represents a backspace. This might cause unusual formatting (making it appear that some content is overwritten) in `impala-shell` output.
- \0 represents an ASCII nul character (not the same as a SQL NULL). This might not be visible in `impala-shell` output.
- \Z represents a DOS end-of-file character. This might not be visible in `impala-shell` output.
- \% and _ can be used to escape wildcard characters within the string passed to the LIKE operator.
- \ followed by 3 octal digits represents the ASCII code of a single character; for example, \101 is ASCII 65, the character A.
- Use two consecutive backslashes (\\) to prevent the backslash from being interpreted as an escape character.
- Use the backslash to escape single or double quotation mark characters within a string literal, if the literal is enclosed by the same type of quotation mark.
- If the character following the \ does not represent the start of a recognized escape sequence, the character is passed through unchanged.

Quotes within quotes:

To include a single quotation character within a string value, enclose the literal with either single or double quotation marks, and optionally escape the single quote as a \' sequence. Earlier releases required escaping a single quote inside double quotes. Continue using escape sequences in this case if you also need to run your SQL code on older versions of Impala.

To include a double quotation character within a string value, enclose the literal with single quotation marks, no escaping is necessary in this case. Or, enclose the literal with double quotation marks and escape the double quote as a \" sequence.

```
[localhost:21000] > select "What\'s happening?" as single_within_double,
                  >        'I\'m not sure.' as single_within_single,
```

```
                   >        "Homer wrote \"The Iliad\"." as double_within_d
ouble,
                   >        'Homer also wrote "The Odyssey".' as double_wi
thin_single;
+----------------------+----------------------+--------------------------+-
--------------------------------+
| single_within_double | single_within_single | double_within_double     |
 double_within_single            |
+----------------------+----------------------+--------------------------
+---------------------------------+
| What's happening?    | I'm not sure.        | Homer wrote "The Iliad". | H
omer also wrote "The Odyssey". |
+----------------------+----------------------+--------------------------+-
--------------------------------+
```

Field terminator character in CREATE TABLE:

Note: The CREATE TABLE clauses FIELDS TERMINATED BY, ESCAPED BY, and LINES TERMINATED BY have special rules for the string literal used for their argument, because they all require a single character. You can use a regular character surrounded by single or double quotation marks, an octal sequence such as '\054' (representing a comma), or an integer in the range '-127'..'128' (with quotation marks but no backslash), which is interpreted as a single-byte ASCII character. Negative values are subtracted from 256; for example, FIELDS TERMINATED BY '-2' sets the field delimiter to ASCII code 254, the “Icelandic Thorn” character used as a delimiter by some data formats.

impala-shell considerations:

When dealing with output that includes non-ASCII or non-printable characters such as linefeeds and backspaces, use the `impala-shell` options to save to a file, turn off pretty printing, or both rather than relying on how the output appears visually.

BOOLEAN literals

For BOOLEAN values, the literals are TRUE and FALSE, with no quotation marks and case-insensitive.

TIMESTAMP literals

Impala automatically converts STRING literals of the correct format into TIMESTAMP values. Timestamp values are accepted in the format 'yyyy#MM#dd HH:mm:ss.SSSSSS', and can consist of just the date, or just the time, with or without the fractional second portion. For example, you can specify TIMESTAMP values such as '1966#07#30', '08:30:00', or '1985#09#25 17:45:30.005'.

Leading zeroes are not required in the numbers representing the date component, such as month and date, or the time component, such as hour, minute, and second. For example, Impala accepts both '2018#1#1 01:02:03' and '2018-01-01 1:2:3' as valid.

In STRING to TIMESTAMP conversions, leading and trailing white spaces, such as a space, a tab, a newline, or a carriage return, are ignored. For example, Impala treats the following as equivalent: '1999#12#01 01:02:03 ', ' 1999#12#01 01:02:03', '1999#12#01 01:02:03\r\n\t'.

When you convert or cast a STRING literal to TIMESTAMP, you can use the following separators between the date part and the time part:

- One or more space characters

  Example: CAST('2001-01-09 01:05:01' AS TIMESTAMP)
- The character “T”

  Example: CAST('2001-01-09T01:05:01' AS TIMESTAMP)

You can also use INTERVAL expressions to add or subtract from timestamp literal values, such as CAST('1966#07#30' AS TIMESTAMP) + INTERVAL 5 YEARS + INTERVAL 3 DAYS.

Depending on your data pipeline, you might receive date and time data as text, in notation that does not exactly match the format for Impala TIMESTAMP literals. See *Impala Date and Time Functions* for functions that can convert between a variety of string literals (including different field order, separators, and timezone notation) and equivalent TIMESTAMP or numeric values.

DATE literals

The DATE literals are in the form of DATE'YYYY-MM-DD'. For example, DATE '2013-01-01'

NULL

The notion of NULL values is familiar from all kinds of database systems, but each SQL dialect can have its own behavior and restrictions on NULL values. For Big Data processing, the precise semantics of NULL values are significant: any misunderstanding could lead to inaccurate results or misformatted data, that could be time-consuming to correct for large data sets.

- NULL is a different value than an empty string. The empty string is represented by a string literal with nothing inside, "" or ''.
- In a delimited text file, the NULL value is represented by the special token \N.
- When Impala inserts data into a partitioned table, and the value of one of the partitioning columns is NULL or the empty string, the data is placed in a special partition that holds only these two kinds of values. When these values are returned in a query, the result is NULL whether the value was originally NULL or an empty string. This behavior is compatible with the way Hive treats NULL values in partitioned tables. Hive does not allow empty strings as partition keys, and it returns a string value such as __HIVE_DEFAULT_PARTITION__ instead of NULL when such values are returned from a query. For example:

```
create table t1 (i int) partitioned by (x int, y string);
-- Select an INT column from another table, with all rows going into a spe
cial HDFS subdirectory
-- named __HIVE_DEFAULT_PARTITION__. Depending on whether one or both of
the partitioning keys
-- are null, this special directory name occurs at different levels of the
 physical data directory
-- for the table.
insert into t1 partition(x=NULL, y=NULL) select c1 from some_other_table;
insert into t1 partition(x, y=NULL) select c1, c2 from some_other_table;
insert into t1 partition(x=NULL, y) select c1, c3  from some_other_table;
```

- There is no NOT NULL clause when defining a column to prevent NULL values in that column.
- There is no DEFAULT clause to specify a non-NULL default value.
- If an INSERT operation mentions some columns but not others, the unmentioned columns contain NULL for all inserted rows.
- In Impala 1.2.1 and higher, all NULL values come at the end of the result set for ORDER BY ... ASC queries, and at the beginning of the result set for ORDER BY ... DESC queries. In effect, NULL is considered greater than all other values for sorting purposes. The original Impala behavior always put NULL values at the end, even for ORDER BY ... DESC queries. The new behavior in Impala 1.2.1 makes Impala more compatible with other popular database systems. In Impala 1.2.1 and higher, you can override or specify the sorting behavior for NULL by adding the clause NULLS FIRST or NULLS LAST at the end of the ORDER BY clause.

  Note: Because the NULLS FIRST and NULLS LAST keywords are not currently available in Hive queries, any views you create using those keywords will not be available through Hive.

- In all other contexts besides sorting with ORDER BY, comparing a NULL to anything else returns NULL, making the comparison meaningless. For example, 10 > NULL produces NULL, 10 < NULL also produces NULL, 5 BETWEEN 1 AND NULL produces NULL, and so on.

Several built-in functions serve as shorthand for evaluating expressions and returning NULL, 0, or some other substitution value depending on the expression result: ifnull(), isnull(), nvl(), nullif(), nullifzero(), and zeroifnull().

Kudu considerations:

Columns in Kudu tables have an attribute that specifies whether or not they can contain NULL values. A column with a NULL attribute can contain nulls. A column with a NOT NULL attribute cannot contain any nulls, and an INSE RT, UPDATE, or UPSERT statement will skip any row that attempts to store a null in a column designated as NOT NULL. Kudu tables default to the NULL setting for each column, except columns that are part of the primary key.

In addition to columns with the NOT NULL attribute, Kudu tables also have restrictions on NULL values in columns that are part of the primary key for a table. No column that is part of the primary key in a Kudu table can contain any NULL values.

Related Information

Impala SQL data types

Impala Shell configuration options

Impala SQL operators

SQL operators are used primarily in the WHERE clause to perform operations, such as comparison operations and arithmetic operations.

Arithmetic operators

The arithmetic operators use expressions with a left-hand argument, the operator, and then (in most cases) a right-hand argument.

Syntax:

```
LEFT_HAND_ARG BINARY_OPERATOR RIGHT_HAND_ARG
UNARY_OPERATOR SINGLE_ARG
```

- + and -: Can be used either as unary or binary operators.
  - With unary notation, such as +5, -2.5, or -*COL_NAME*, they multiply their single numeric argument by +1 or -1. Therefore, unary + returns its argument unchanged, while unary - flips the sign of its argument. Although you can double up these operators in expressions such as ++5 (always positive) or -+2 or +-2 (both always negative), you cannot double the unary minus operator because -- is interpreted as the start of a comment. (You can use a double unary minus operator if you separate the - characters, for example with a space or parentheses.)
  - With binary notation, such as 2+2, 5-2.5, or *COL1* + *COL2*, they add or subtract respectively the right-hand argument to (or from) the left-hand argument. Both arguments must be of numeric types.
- * and /: Multiplication and division respectively. Both arguments must be of numeric types.

  When multiplying, the shorter argument is promoted if necessary (such as SMALLINT to INT or BIGINT, or FLOAT to DOUBLE), and then the result is promoted again to the next larger type. Thus, multiplying a TINY INT and an INT produces a BIGINT result. Multiplying a FLOAT and a FLOAT produces a DOUBLE result. Multiplying a FLOAT and a DOUBLE or a DOUBLE and a DOUBLE produces a DECIMAL(38,17), because DECIMAL values can represent much larger and more precise values than DOUBLE.

  When dividing, Impala always treats the arguments and result as DOUBLE values to avoid losing precision. If you need to insert the results of a division operation into a FLOAT column, use the CAST() function to convert the result to the correct type.
- DIV: Integer division. Arguments are not promoted to a floating-point type, and any fractional result is discarded. For example, 13 DIV 7 returns 1, 14 DIV 7 returns 2, and 15 DIV 7 returns 2. This operator is the same as the QUOTIENT() function.
- %: Modulo operator. Returns the remainder of the left-hand argument divided by the right-hand argument. Both arguments must be of one of the integer types.

- &, |, ~, and ^: Bitwise operators that return the logical AND, logical OR, NOT, or logical XOR (exclusive OR) of their argument values. Both arguments must be of one of the integer types. If the arguments are of different type, the argument with the smaller type is implicitly extended to match the argument with the longer type.

You can chain a sequence of arithmetic expressions, optionally grouping them with parentheses.

The arithmetic operators generally do not have equivalent calling conventions using functional notation. For example, prior to Impala 2.2, there is no MOD() function equivalent to the % modulo operator. Conversely, there are some arithmetic functions that do not have a corresponding operator. For example, for exponentiation you use the POW() function, but there is no ** exponentiation operator. See *Impala Mathematical Functions* for the arithmetic functions you can use.

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
| EUROPE      | 19               |
| EUROPE      | 22               |
```

```
| EUROPE      | 23                |
| MIDDLE EAST | 4                 |
| MIDDLE EAST | 10                |
| MIDDLE EAST | 11                |
| MIDDLE EAST | 13                |
| MIDDLE EAST | 20                |
+-------------+-------------------+

select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

The following example shows how to do an arithmetic operation using a numeric field of a STRUCT type that is an item within an ARRAY column. Once the scalar numeric value R_NATIONKEY is extracted, it can be used in an arithmetic expression, such as multiplying by 10:

```
-- The SMALLINT is a field within an array of structs.
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+
```

```
-- When we refer to the scalar value using dot notation,
-- we can use arithmetic and comparison operators on it
-- like any other number.
select r_name, nation.item.n_name, nation.item.n_nationkey * 10
  from region, region.r_nations as nation
where nation.item.n_nationkey < 5;
+-------------+-------------+-----------------------------+
| r_name      | item.n_name | nation.item.n_nationkey * 10 |
+-------------+-------------+-----------------------------+
| AMERICA     | CANADA      | 30                          |
| AMERICA     | BRAZIL      | 20                          |
| AMERICA     | ARGENTINA   | 10                          |
| MIDDLE EAST | EGYPT       | 40                          |
| AFRICA      | ALGERIA     | 0                           |
+-------------+-------------+-----------------------------+
```

BETWEEN operator

In a WHERE clause, compares an expression to both a lower and upper bound. The comparison is successful is the expression is greater than or equal to the lower bound, and less than or equal to the upper bound. If the bound values are switched, so the lower bound is greater than the upper bound, does not match any values.

Syntax:

```
EXPRESSION BETWEEN LOWER_BOUND AND UPPER_BOUND
```

Data types: Typically used with numeric data types. Works with any data type, although not very practical for BOOLEAN values. (BETWEEN false AND true will match all BOOLEAN values.) Use CAST() if necessary to ensure the lower and upper bound values are compatible types. Call string or date/time functions if necessary to extract or transform the relevant portion to compare, especially if the value can be transformed into a number.

Usage notes:

Be careful when using short string operands. A longer string that starts with the upper bound value will not be included, because it is considered greater than the upper bound. For example, BETWEEN 'A' and 'M' would not match the string value 'Midway'. Use functions such as upper(), lower(), substr(), trim(), and so on if necessary to ensure the comparison works as expected.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

Examples:

The following example shows how to do a BETWEEN comparison using a numeric field of a STRUCT type that is an item within an ARRAY column. Once the scalar numeric value R_NATIONKEY is extracted, it can be used in a comparison operator:

```
-- The SMALLINT is a field within an array of structs.
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
```

```
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

-- When we refer to the scalar value using dot notation,
-- we can use arithmetic and comparison operators on it
-- like any other number.
select r_name, nation.item.n_name, nation.item.n_nationkey
from region, region.r_nations as nation
where nation.item.n_nationkey between 3 and 5
+-------------+-------------+------------------+
| r_name      | item.n_name | item.n_nationkey |
+-------------+-------------+------------------+
| AMERICA     | CANADA      | 3                |
| MIDDLE EAST | EGYPT       | 4                |
| AFRICA      | ETHIOPIA    | 5                |
+-------------+-------------+------------------+
```

Comparison operators

Impala supports the familiar comparison operators for checking equality and sort order for the column data types:

Syntax:

```
LEFT_HAND_EXPRESSION COMPARISON_OPERATOR RIGHT_HAND_EXPRESSION
```

- =, !=, <>: apply to all scalar types.
- <, <=, >, >=: apply to all scalar types; for BOOLEAN, TRUE is considered greater than FALSE.

Alternatives:

The IN and BETWEEN operators provide shorthand notation for expressing combinations of equality, less than, and greater than comparisons with a single operator.

Because comparing any value to NULL produces NULL rather than TRUE or FALSE, use the IS NULL and IS NOT NULL operators to check if a value is NULL or not.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

The following example shows how to do an arithmetic operation using a numeric field of a STRUCT type that is an item within an ARRAY column. Once the scalar numeric value R_NATIONKEY is extracted, it can be used with a comparison operator such as <:

```
-- The SMALLINT is a field within an array of structs.
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
```

```
+-------------+-------------------------+---------+

-- When we refer to the scalar value using dot notation,
-- we can use arithmetic and comparison operators on it
-- like any other number.
select r_name, nation.item.n_name, nation.item.n_nationkey
from region, region.r_nations as nation
where nation.item.n_nationkey < 5
+-------------+-------------+------------------+
| r_name      | item.n_name | item.n_nationkey |
+-------------+-------------+------------------+
| AMERICA     | CANADA      | 3                |
| AMERICA     | BRAZIL      | 2                |
| AMERICA     | ARGENTINA   | 1                |
| MIDDLE EAST | EGYPT       | 4                |
| AFRICA      | ALGERIA     | 0                |
+-------------+-------------+------------------+
```

EXISTS operator

The EXISTS operator tests whether a subquery returns any results. You typically use it to find values from one table that have corresponding values in another table.

The converse, NOT EXISTS, helps to find all the values from one table that do not have any corresponding values in another table.

Syntax:

```
EXISTS (SUBQUERY)
NOT EXISTS (SUBQUERY)
```

Usage notes:

The subquery can refer to a different table than the outer query block, or the same table. For example, you might use EXISTS or NOT EXISTS to check the existence of parent/child relationships between two columns of the same table.

You can also use operators and function calls within the subquery to test for other kinds of relationships other than strict equality. For example, you might use a call to COUNT() in the subquery to check whether the number of matching values is higher or lower than some limit. You might call a UDF in the subquery to check whether values in one table matches a hashed representation of those same values in a different table.

NULL considerations:

If the subquery returns any value at all (even NULL), EXISTS returns TRUE and NOT EXISTS returns false.

Restrictions:

Correlated subqueries used in EXISTS and IN operators cannot include a LIMIT clause.

Prior to Impala 2.6, the NOT EXISTS operator required a correlated subquery. In Impala 2.6 and higher, NOT EXIS TS works with uncorrelated queries also.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

ILIKE operator

A case-insensitive comparison operator for STRING data, with basic wildcard capability using _ to match a single character and % to match any characters. The argument expression must match the entire string value. Typically, it is more efficient to put any % wildcard match at the end of the string.

This operator, available in Impala 2.5 and higher, is the equivalent of the LIKE operator, but with case-insensitive comparisons.

Syntax:

```
STRING_EXPRESSION ILIKE WILDCARD_EXPRESSION
STRING_EXPRESSION NOT ILIKE WILDCARD_EXPRESSION
```

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

IN operator

The IN operator compares an argument value to a set of values, and returns TRUE if the argument matches any value in the set. The NOT IN operator reverses the comparison, and checks if the argument value is not part of a set of values.

Syntax:

```
EXPRESSION IN (EXPRESSION [, EXPRESSION])
EXPRESSION IN (SUBQUERY)

EXPRESSION NOT IN (EXPRESSION [, EXPRESSION])
EXPRESSION NOT IN (SUBQUERY)
```

The left-hand expression and the set of comparison values must be of compatible types.

The left-hand expression must consist only of a single value, not a tuple. Although the left-hand expression is typically a column name, it could also be some other value. For example, the WHERE clauses WHERE id IN (5) and WHERE 5 IN (id) produce the same results.

The set of values to check against can be specified as constants, function calls, column names, or other expressions in the query text. The maximum number of expressions in the IN list is 9999. (The maximum number of elements of a single expression is 10,000 items, and the IN operator itself counts as one.)

In Impala 2.0 and higher, the set of values can also be generated by a subquery. IN can evaluate an unlimited number of results using a subquery.

Usage notes:

Any expression using the IN operator could be rewritten as a series of equality tests connected with OR, but the IN syntax is often clearer, more concise, and easier for Impala to optimize. For example, with partitioned tables, queries frequently use IN clauses to filter data by comparing the partition key columns to specific values.

NULL considerations:

If there really is a matching non-null value, IN returns TRUE:

If the searched value is not found in the comparison values, and the comparison values include NULL, the result is NULL:

If the left-hand argument is NULL, IN always returns NULL. This rule applies even if the comparison values include NULL.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

Restrictions:

Correlated subqueries used in EXISTS and IN operators cannot include a LIMIT clause.

IREGEXP operator

Tests whether a value matches a regular expression, using case-insensitive string comparisons. Uses the POSIX regular expression syntax where ^ and $ match the beginning and end of the string, . represents any single character, * represents a sequence of zero or more items, + represents a sequence of one or more items, ? produces a non-greedy match, and so on.

This operator, available in Impala 2.5 and higher, is the equivalent of the REGEXP operator, but with case-insensitive comparisons.

Syntax:

```
STRING_EXPRESSION IREGEXP REGULAR_EXPRESSION
```

Usage notes:

The | symbol is the alternation operator, typically used within () to match different sequences. The () groups do not allow backreferences. To retrieve the part of a value matched within a () section, use the regexp_extract() built-in function. (Currently, there is not any case-insensitive equivalent for the regexp_extract() function.)

In Impala 1.3.1 and higher, the REGEXP and RLIKE operators now match a regular expression string that occurs anywhere inside the target string, the same as if the regular expression was enclosed on each side by .*. Previously, these operators only succeeded when the regular expression matched the entire target string. This change improves compatibility with the regular expression support for popular database systems. There is no change to the behavior of the regexp_extract() and regexp_replace() built-in functions.

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

IS DISTINCT FROM operator

The IS DISTINCT FROM operator, and its converse the IS NOT DISTINCT FROM operator, test whether or not values are identical. IS NOT DISTINCT FROM is similar to the = operator, and IS DISTINCT FROM is similar to the != operator, except that NULL values are treated as identical. Therefore, IS NOT DISTINCT FROM returns true rather than NULL, and IS DISTINCT FROM returns false rather than NULL, when comparing two NULL values. If one of the values being compared is NULL and the other is not, IS DISTINCT FROM returns true and IS NOT DISTINCT FROM returns false, again instead of returning NULL in both cases.

Syntax:

```
EXPRESSION1 IS DISTINCT FROM EXPRESSION2

EXPRESSION1 IS NOT DISTINCT FROM EXPRESSION2
EXPRESSION1 <=> EXPRESSION2
```

The operator <=> is an alias for IS NOT DISTINCT FROM. It is typically used as a NULL-safe equality operator in join queries. That is, A <=> B is true if A equals B or if both A and B are NULL.

Usage notes:

This operator provides concise notation for comparing two values and always producing a true or false result, without treating NULL as a special case. Otherwise, to unambiguously distinguish between two values requires a compound expression involving IS [NOT] NULL tests of both operands in addition to the = or != operator.

The <=> operator, used like an equality operator in a join query, is more efficient than the equivalent clause: IF (A IS NULL OR B IS NULL, A IS NULL AND B IS NULL, A = B). The <=> operator can use a hash join, while the IF expression cannot.

IS NULL operator

The IS NULL operator, and its converse the IS NOT NULL operator, test whether a specified value is NULL. Because using NULL with any of the other comparison operators such as = or != also returns NULL rather than TRUE or FALSE, you use a special-purpose comparison operator to check for this special condition.

In Impala 2.1.1 and higher, you can use the operators IS UNKNOWN and IS NOT UNKNOWN as synonyms for IS NULL and IS NOT NULL, respectively.

Syntax:

```
EXPRESSION IS NULL
EXPRESSION IS NOT NULL

EXPRESSION IS UNKNOWN
EXPRESSION IS NOT UNKNOWN
```

Usage notes:

In many cases, NULL values indicate some incorrect or incomplete processing during data ingestion or conversion. You might check whether any values in a column are NULL, and if so take some followup action to fill them in.

With sparse data, often represented in “wide” tables, it is common for most values to be NULL with only an occasional non-NULL value. In those cases, you can use the IS NOT NULL operator to identify the rows containing any data at all for a particular column, regardless of the actual value.

With a well-designed database schema, effective use of NULL values and IS NULL and IS NOT NULL operators can save having to design custom logic around special values such as 0, -1, 'N/A', empty string, and so on. NULL lets you distinguish between a value that is known to be 0, false, or empty, and a truly unknown value.

Complex type considerations:

The IS [NOT] UNKNOWN operator, as with the IS [NOT] NULL operator, is not applicable to complex type columns (STRUCT, ARRAY, or MAP). Using a complex type column with this operator causes a query error.

IS TRUE operator

This variation of the IS operator tests for truth or falsity, with right-hand arguments [NOT] TRUE, [NOT] FALSE, and [NOT] UNKNOWN.

Syntax:

```
EXPRESSION IS TRUE
EXPRESSION IS NOT TRUE

EXPRESSION IS FALSE
EXPRESSION IS NOT FALSE
```

Usage notes:

This IS TRUE and IS FALSE forms are similar to doing equality comparisons with the Boolean values TRUE and FALSE, except that IS TRUE and IS FALSE always return either TRUE or FALSE, even if the left-hand side expression returns NULL

These operators let you simplify Boolean comparisons that must also check for NULL, for example X != 10 AND X IS NOT NULL is equivalent to (X != 10) IS TRUE.

In Impala 2.1.1 and higher, you can use the operators IS [NOT] TRUE and IS [NOT] FALSE as equivalents for the built-in functions ISTRUE(), ISNOTTRUE(), ISFALSE(), and ISNOTFALSE().

Complex type considerations:

The IS [NOT] TRUE and IS [NOT] FALSE operators are not applicable to complex type columns (STRUCT, ARRAY, or MAP). Using a complex type column with these operators causes a query error.

LIKE operator

A comparison operator for STRING data, with basic wildcard capability using the underscore (_) to match a single character and the percent sign (%) to match any characters. The argument expression must match the entire string value. Typically, it is more efficient to put any % wildcard match at the end of the string.

Syntax:

```
STRING_EXPRESSION LIKE WILDCARD_EXPRESSION
STRING_EXPRESSION NOT LIKE WILDCARD_EXPRESSION
```

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

Examples:

```
select distinct c_last_name from customer where c_last_name like 'Mc%' or c_
last_name like 'Mac%';
select count(c_last_name) from customer where c_last_name like 'M%';
select c_email_address from customer where c_email_address like '%.edu';

-- We can find 4-letter names beginning with 'M' by calling functions...
select distinct c_last_name from customer where length(c_last_name) = 4 and
 substr(c_last_name,1,1) = 'M';
-- ...or in a more readable way by matching M followed by exactly 3 charac
ters.
select distinct c_last_name from customer where c_last_name like 'M___';
```

For case-insensitive comparisons, see the ILIKE operator. For a more general kind of search operator using regular expressions, see the REGEXP operator or its case-insensitive counterpart the IREGEXP operator.

Logical operators

Logical operators return a BOOLEAN value, based on a binary or unary logical operation between arguments that are also BOOLEAN values. Typically, the argument expressions use comparison operators.

Syntax:

```
BOOLEAN_EXPRESSION BINARY_LOGICAL_OPERATOR BOOLEAN_EXPRESSION
UNARY_LOGICAL_OPERATOR BOOLEAN_EXPRESSION
```

The Impala logical operators are:

- AND: A binary operator that returns true if its left-hand and right-hand arguments both evaluate to true, NULL if either argument is NULL, and false otherwise.
- OR: A binary operator that returns true if either of its left-hand and right-hand arguments evaluate to true, NULL if one argument is NULL and the other is either NULL or false, and false otherwise.
- NOT: A unary operator that flips the state of a Boolean expression from true to false, or false to true. If the argument expression is NULL, the result remains NULL. (When NOT is used this way as a unary logical operator, it works differently than the IS NOT NULL comparison operator, which returns true when applied to a NULL.)

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator. You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

The following example shows how to do an arithmetic operation using a numeric field of a STRUCT type that is an item within an ARRAY column. Once the scalar numeric value R_NATIONKEY is extracted, it can be used in an arithmetic expression, such as multiplying by 10:

```
-- The SMALLINT is a field within an array of structs.
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

-- When we refer to the scalar value using dot notation,
-- we can use arithmetic and comparison operators on it
-- like any other number.
select r_name, nation.item.n_name, nation.item.n_nationkey
  from region, region.r_nations as nation
where
  nation.item.n_nationkey between 3 and 5
  or nation.item.n_nationkey < 15;
+-------------+----------------+------------------+
| r_name      | item.n_name    | item.n_nationkey |
+-------------+----------------+------------------+
| EUROPE      | UNITED KINGDOM | 23               |
| EUROPE      | RUSSIA         | 22               |
| EUROPE      | ROMANIA        | 19               |
| ASIA        | VIETNAM        | 21               |
| ASIA        | CHINA          | 18               |
| AMERICA     | UNITED STATES  | 24               |
| AMERICA     | PERU           | 17               |
| AMERICA     | CANADA         | 3                |
| MIDDLE EAST | SAUDI ARABIA   | 20               |
| MIDDLE EAST | EGYPT          | 4                |
| AFRICA      | MOZAMBIQUE     | 16               |
| AFRICA      | ETHIOPIA       | 5                |
+-------------+----------------+------------------+
```

REGEXP operator

Tests whether a value matches a regular expression. Uses the POSIX regular expression syntax where ^ and $ match the beginning and end of the string, . represents any single character, * represents a sequence of zero or more items, + represents a sequence of one or more items, ? produces a non-greedy match, and so on.

Syntax:

```
STRING_EXPRESSION REGEXP REGULAR_EXPRESSION
```

Usage notes:

The RLIKE operator is a synonym for REGEXP.

The | symbol is the alternation operator, typically used within () to match different sequences. The () groups do not allow backreferences. To retrieve the part of a value matched within a () section, use the regexp_extract() built-in function.

In Impala 1.3.1 and higher, the REGEXP and RLIKE operators now match a regular expression string that occurs anywhere inside the target string, the same as if the regular expression was enclosed on each side by .*. Previously, these operators only succeeded when the regular expression matched the entire target string. This change improves compatibility with the regular expression support for popular database systems. There is no change to the behavior of the regexp_extract() and regexp_replace() built-in functions.

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Complex type considerations:

You cannot refer to a column with a complex data type (ARRAY, STRUCT, or MAP) directly in an operator.
You can apply operators only to scalar values that make up a complex type (the fields of a STRUCT, the items of an ARRAY, or the key or value portion of a MAP) as part of a join query that refers to the scalar value using the appropriate dot notation or ITEM, KEY, or VALUE pseudocolumn names.

Example:

```
- Find all customers whose first name starts with 'J', followed by 0 or more
 of any character.
select c_first_name, c_last_name from customer where c_first_name regexp
'^J.*';
-- Match multiple character sequences, either 'Mac' or 'Mc'.
select c_first_name, c_last_name from customer where c_last_name regexp '^
(Mac|Mc)donald$';
```

RLIKE operator

It is a synonym for the REGEXP operator.

Related Information

Impala mathematical functions

RE2 documentation on GitHub

Impala SQL comments

Impala supports the syntax of standard SQL comments.

- All text from a -- sequence to the end of the line is considered a comment and ignored. This type of comment can occur on a single line by itself, or after all or part of a statement.
- All text from a /* sequence to the next */ sequence is considered a comment and ignored. This type of comment can stretch over multiple lines. This type of comment can occur on one or more lines by itself, in the middle of a statement, or before or after a statement.

Impala SQL statements

The Impala SQL dialect supports a range of standard elements, plus some extensions for Big Data use cases related to data loading and data warehousing.

Note:

In the `impala-shell` interpreter, a semicolon at the end of each statement is required. Since the semicolon is not actually part of the SQL syntax, we do not include it in the syntax definition of each statement, but we do show it in examples intended to be run in `impala-shell`.

The following sections show the major SQL statements that you work with in Impala:

ROLE statements in Impala integrated with Ranger

After upgrading or migrating your workload from CDH to you will have started using Apache Ranger as the authorization provider in Impala. This replaces Apache Sentry. There are some differences in Impala's behavior which you must be aware of when using Ranger as the authorization provider.

Access via Roles in Impala

Impala with Sentry revolved around granting privileges to ROLES, and ROLES to GROUPS. Until this release, Impala's integration with Ranger did not support ROLE related DDL statements. So as a workaround you had to migrate the ROLE-based authorization policies, manage them using Ranger's web UI, to handle them correctly in Impala. You will no longer need to use the Ranger's web UI to manage the ROLEs in 7.1.6 since Impala now supports ROLE management through ROLE related statements.

Note: ROLES can be still managed using the old approach via Ranger's web UI.

Managing ROLES using Ranger's WEB UI

You can perform the following steps to grant to a normal user the privileges on a resource via ROLEs.

1. Assign a user to a group, e.g. using your corporate LDAP provider.
2. In the Ranger web UI, create a ROLE that includes the group consisting of the user we want to grant the privileges.
3. In the Ranger web UI,
   - create a policy for the corresponding resource, e.g., a table,
   - in the 'Select Role' field under the section of 'Allow Conditions', add the ROLE just created above,
   - in the 'Permissions' field under the section of 'Allow Conditions', add the privileges we want to grant to the ROLE associated with the group consisting of that normal user.

ROLE related statements in Impala

To bridge the gap between CDH Impala and Impala in terms of ROLE-related operations, 7.1.6 introduces ROLE related statements in Impala integrated with Ranger.

```
CREATE ROLE <role_name>
DROP ROLE <role_name>
GRANT ROLE <role_name> TO GROUP <group_name>
REVOKE ROLE <role_name> FROM GROUP <group_name>
GRANT <privilege> ON <resource> TO ROLE <role_name>
REVOKE <privilege> ON <resource> FROM ROLE <role_name>
SHOW GRANT ROLE <role_name> ON <resource>
SHOW ROLES
SHOW CURRENT ROLES
SHOW ROLE GRANT GROUP <group_name>
```

Differences in Impala's behavior

The following list describes the major differences in Impala's behavior when using Ranger as the authorization provider in place of Sentry.

- Before dropping a role in Ranger, you must remove all the privileges granted to the role in advance, which was not the case when Sentry was the authorization provider.
- You must specify the resource for the SHOW GRANT ROLE <role_name> ON <resource> statement which is different when using Sentry as the authorization provider. This is due to the fact that there is no API provided by Ranger that allows Impala to directly retrieve the list of all privileges granted to a specified role.

DDL statements

DDL refers to "Data Definition Language", a subset of SQL statements that change the structure of the database schema in some way, typically by creating, deleting, or modifying schema objects such as databases, tables, and views. Most Impala DDL statements start with the keywords CREATE, DROP, or ALTER.

The Impala DDL statements are:

- ALTER TABLE statement on page 109
- ALTER VIEW statement on page 124
- COMPUTE STATS statement on page 126
- CREATE DATABASE statement on page 133
- CREATE FUNCTION statement on page 135
- CREATE TABLE statement on page 142
- CREATE VIEW statement on page 158
- DROP DATABASE statement on page 173
- DROP FUNCTION statement on page 175
- DROP TABLE statement on page 182
- DROP VIEW statement on page 183
- GRANT statement on page 187
- REVOKE statement on page 206

After Impala executes a DDL command, information about available tables, columns, views, partitions, and so on is automatically synchronized between all the Impala nodes in a cluster. (Prior to Impala 1.2, you had to issue a REFR ESH or an INVALIDATE METADATA statement manually on the other nodes to make them aware of the changes.)

If the timing of automatic metadata updates is significant, such as when connecting to different Impala nodes within an impala-shell session for load balancing or using round-robin scheduling, enable the SYNC_DDL query option. This will ensure that the DDL statement waits until all nodes are notified about the metadata changes.

See *Using Impala with the Amazon S3 Filesystem* for details about how Impala DDL statements interact with tables and partitions stored in the Amazon S3 filesystem.

Although INSERT is classified as a DML statement, when the SYNC_DDL option is enabled, INSERT statements also delay their completion until all the underlying data and metadata changes are propagated to all Impala nodes and this option applies to all filesystem-based tables. Internally, Impala inserts have similarities with DDL statements in traditional database systems, because they create metadata needed to track HDFS block locations for new files and they potentially add new partitions to partitioned tables.

Because the SYNC_DDL query option makes each DDL operation take longer than normal, you might only enable it before the last DDL operation in a sequence. For example, if you are running a script that issues multiple of DDL operations to set up an entire new schema, add several new partitions, and so on, you might minimize the performance overhead by enabling the query option only before the last CREATE, DROP, ALTER, or INSERT statement. The script only finishes when all the relevant metadata changes are recognized by all the Impala nodes, so you could connect to any node and issue queries through it.

The classification of DDL, DML, and other statements is not necessarily the same between Impala and Hive. Impala organizes these statements in a way intended to be familiar to people familiar with relational databases or data warehouse products. Statements that modify the metastore database, such as COMPUTE STATS, are classified as DDL. Statements that only query the metastore database, such as SHOW or DESCRIBE, are put into a separate category of utility statements.

Note: The query types shown in the Impala debug web user interface might not match exactly the categories listed here. For example, currently the USE statement is shown as DDL in the debug web UI. The query types shown in the debug web UI are subject to change, for improved consistency.

DML statements

DML refers to “Data Manipulation Language”, a subset of SQL statements that modify the data stored in tables. Because Impala focuses on query performance and leverages the append-only nature of HDFS storage, currently Impala only supports a small set of DML statements.

Impala supports the following DML statements:

- DELETE: Works for Kudu tables only
- INSERT
- LOAD DATA: Does not apply for HBase or Kudu tables
- UPDATE: Works for Kudu tables only
- UPSERT: Works for Kudu tables only

INSERT in Impala is primarily optimized for inserting large volumes of data in a single statement, to make effective use of the multi-megabyte HDFS blocks. This is the way in Impala to create new data files. If you intend to insert one or a few rows at a time, such as using the INSERT ... VALUES syntax, that technique is much more efficient for Impala tables stored in HBase.

LOAD DATA moves existing data files into the directory for an Impala table, making them immediately available for Impala queries. This is one way in Impala to work with data files produced by other Hadoop components. (CREATE EXTERNAL TABLE is the other alternative; with external tables, you can query existing data files, while the files remain in their original location.)

In Impala 2.8 and higher, Impala does support the UPDATE, DELETE, and UPSERT statements for Kudu tables. For HDFS or S3 tables, to simulate the effects of an UPDATE or DELETE statement in other database systems, typically you use INSERT or CREATE TABLE AS SELECT to copy data from one table to another, filtering out or changing the appropriate rows during the copy operation.

You can also achieve a result similar to UPDATE by using Impala tables stored in HBase. When you insert a row into an HBase table, and the table already contains a row with the same value for the key column, the older row is hidden, effectively the same as a single-row UPDATE.

Impala can perform DML operations for tables or partitions stored in the Amazon S3 filesystem.

ALTER DATABASE statement

The ALTER DATABASE statement changes the characteristics of a database.

Use the SET OWNER clause to transfer the ownership of the database from the current owner to another user or a role.

The database owner is originally set to the user who creates the database. When object ownership is enabled in Ranger, an owner of a database can have the ALL with GRANT or ALL without GRANT privilege. The term OWNER is used to differentiate between the ALL privilege that is explicitly granted via the GRANT statement and a privilege that is implicitly granted by the CREATE DATABASE statement.

Syntax:

```
ALTER DATABASE DATABASE_NAME SET OWNER USER user_name;
ALTER DATABASE DATABASE_NAME SET OWNER ROLE role_name;
```

Statement type: DDL

Cancellation: Cannot be cancelled.

Added in: Impala 3.1 release.

ALTER TABLE statement

The ALTER TABLE statement changes the structure or properties of an existing Impala table.

In Impala, this is primarily a logical operation that updates the table metadata in the metastore database that Impala shares with Hive. Most ALTER TABLE operations do not actually rewrite, move, and so on the actual data files. (The RENAME TO clause is the one exception; it can cause HDFS files to be moved to different paths.) When you do an ALTER TABLE operation, you typically need to perform corresponding physical filesystem operations, such as rewriting the data files to include extra fields, or converting them to a different file format.

Syntax:

```
ALTER TABLE [OLD_DB_NAME.]OLD_TABLE_NAME RENAME TO
 [NEW_DB_NAME.]NEW_TABLE_NAME

ALTER TABLE NAME ADD [IF NOT EXISTS] COLUMNS (COL_SPEC[, COL_SPEC ...])
ALTER TABLE NAME REPLACE COLUMNS (COL_SPEC[, COL_SPEC ...])

ALTER TABLE NAME ADD COLUMN [IF NOT EXISTS] COL_SPEC
ALTER TABLE NAME DROP [COLUMN] COLUMN_NAME
ALTER TABLE NAME CHANGE COLUMN_NAME COL_SPEC

ALTER TABLE NAME SET OWNER USER USER_NAME
ALTER TABLE NAME SET OWNER ROLE ROLE_NAME


-- Kudu tables only.
ALTER TABLE NAME ALTER [COLUMN] COLUMN_NAME
  { SET KUDU_STORAGE_ATTR ATTR_VALUE
    | DROP DEFAULT }

kudu_storage_attr ::= { DEFAULT | BLOCK_SIZE | ENCODING | COMPRESSION }

-- Non-Kudu tables only.
ALTER TABLE NAME ALTER [COLUMN] COLUMN_NAME
  SET COMMENT 'COMMENT_TEXT'
```

```
ALTER TABLE NAME ADD [IF NOT EXISTS] PARTITION (PARTITION_SPEC)
  [LOCATION_SPEC]
  [CACHE_SPEC]
ALTER TABLE NAME ADD [IF NOT EXISTS] RANGE PARTITION KUDU_PARTITION_SPEC

ALTER TABLE NAME DROP [IF EXISTS] PARTITION (PARTITION_SPEC)
  [PURGE]
ALTER TABLE NAME DROP [IF EXISTS] RANGE PARTITION KUDU_PARTITION_SPEC

ALTER TABLE NAME RECOVER PARTITIONS

ALTER TABLE NAME [PARTITION (PARTITION_SPEC)]
  SET { FILEFORMAT FILE_FORMAT
  | ROW FORMAT ROW_FORMAT
  | LOCATION 'HDFS_PATH_OF_DIRECTORY'
  | TBLPROPERTIES (TABLE_PROPERTIES)
  | SERDEPROPERTIES (SERDE_PROPERTIES) }

ALTER TABLE NAME COLNAME
  ('STATSKEY'='VAL, ...)

statsKey ::= numDVs | numNulls | avgSize | maxSize

ALTER TABLE NAME [PARTITION (PARTITION_SPEC)] SET { CACHED IN
 'POOL_NAME' [WITH REPLICATION = INTEGER] | UNCACHED }

NEW_NAME ::= [NEW_DATABASE.]NEW_TABLE_NAME

COL_SPEC ::= COL_NAME TYPE_NAME  COMMENT 'COLUMN-COMMENT' [KUDU_ATTRIBUTES]

KUDU_ATTRIBUTES ::= { [NOT] NULL | ENCODING CODEC | COMPRESSION ALGORITHM |
  DEFAULT CONSTANT | BLOCK_SIZE NUMBER }

PARTITION_SPEC ::= SIMPLE_PARTITION_SPEC | COMPLEX_PARTITION_SPEC

SIMPLE_PARTITION_SPEC ::= PARTITION_COL=CONSTANT_VALUE

COMPLEX_PARTITION_SPEC ::= COMPARISON_EXPRESSION_ON_PARTITION_COL

KUDU_PARTITION_SPEC ::= CONSTANT RANGE_OPERATOR
 VALUES RANGE_OPERATOR CONSTANT | VALUE = CONSTANT

cache_spec ::= CACHED IN 'POOL_NAME' [WITH REPLICATION = INTEGER] | UNCACHED

location_spec ::= LOCATION 'HDFS_PATH_OF_DIRECTORY'

TABLE_PROPERTIES ::= 'NAME'='VALUE'[, 'NAME'='VALUE' ...]

SERDE_PROPERTIES ::= 'NAME'='VALUE'[, 'NAME'='VALUE' ...]

FILE_FORMAT ::= { PARQUET | TEXTFILE | RCFILE | SEQUENCEFILE | AVRO }

ROW_FORMAT ::= DELIMITED
  [FIELDS TERMINATED BY 'CHAR' [ESCAPED BY 'CHAR']]
  [LINES TERMINATED BY 'CHAR']
```

Statement type: DDL

Complex type considerations:

In Impala 2.3 and higher, the ALTER TABLE statement can change the metadata for tables containing complex types (ARRAY, STRUCT, and MAP). For example, you can use an ADD COLUMNS, DROP COLUMN, or CHANGE clause to modify the table layout for complex type columns. Although Impala queries only work for complex type

columns in Parquet tables, the complex type support in the ALTER TABLE statement applies to all file formats. For example, you can use Impala to update metadata for a staging table in a non-Parquet file format where the data is populated by Hive. Or you can use ALTER TABLE SET FILEFORMAT to change the format of an existing table to Parquet so that Impala can query it. Remember that changing the file format for a table does not convert the data files within the table; you must prepare any Parquet data files containing complex types outside Impala, and bring them into the table using LOAD DATA or updating the table's LOCATION property.

Usage notes:

Whenever you specify partitions in an ALTER TABLE statement, through the PARTITION (*PARTITION_SPEC*) clause, you must include all the partitioning columns in the specification.

Most of the ALTER TABLE operations work the same for internal tables (managed by Impala) as for external tables (with data files located in arbitrary locations). The exception is renaming a table; for an external table, the underlying data directory is not renamed or moved.

To drop or alter multiple partitions:

In Impala 2.8 and higher, the expression for the partition clause with a DROP or SET operation can include comparison operators such as <, IN, or BETWEEN, and Boolean operators such as AND and OR.

For example, you might drop a group of partitions corresponding to a particular date range after the data “ages out”:

```
alter table historical_data drop partition (year < 1995);
alter table historical_data drop partition (year = 1996 and month between 1
 and 6);
```

For tables with multiple partition keys columns, you can specify multiple conditions separated by commas, and the operation only applies to the partitions that match all the conditions (similar to using an AND clause):

```
alter table historical_data drop partition (year < 1995, last_name like 'A%
');
```

This technique can also be used to change the file format of groups of partitions, as part of an ETL pipeline that periodically consolidates and rewrites the underlying data files in a different file format:

```
alter table fast_growing_data partition (year = 2016, month in (10,11,12))
set fileformat parquet;
```

Note:

The extended syntax involving comparison operators and multiple partitions applies to the SET FILEFORMAT, SET TBLPROPERTIES, SET SERDEPROPERTIES, and SET [UN]CACHED clauses. You can also use this syntax with the PARTITION clause in the COMPUTE INCREMENTAL STATS statement, and with the PARTITION clause of the SHOW FILES statement. Some forms of ALTER TABLE still only apply to one partition at a time: the SET LOCATION and ADD PARTITION clauses. The PARTITION clauses in the LOAD DATA and INSERT statements also only apply to one partition at a time.

A DDL statement that applies to multiple partitions is considered successful (resulting in no changes) even if no partitions match the conditions. The results are the same as if the IF EXISTS clause was specified.

The performance and scalability of this technique is similar to issuing a sequence of single-partition ALTER TABLE statements in quick succession. To minimize bottlenecks due to communication with the metastore database, or causing other DDL operations on the same table to wait, test the effects of performing ALTER TABLE statements that affect large numbers of partitions.

Amazon S3 considerations:

You can specify an s3a:// prefix on the LOCATION attribute of a table or partition to make Impala query data from the Amazon S3 filesystem. In Impala 2.6 and higher, Impala automatically handles creating or removing the associated folders when you issue ALTER TABLE statements with the ADD PARTITION or DROP PARTITION clauses.

In Impala 2.6 and higher, Impala DDL statements such as CREATE DATABASE, CREATE TABLE, DROP DATABASE CASCADE, DROP TABLE, and ALTER TABLE [ADD|DROP] PARTITION can create or remove folders as needed in the Amazon S3 system. Prior to Impala 2.6, you had to create folders yourself and point Impala database, tables, or partitions at them, and manually remove folders when no longer needed. See Related Information for details about reading and writing S3 data with Impala.

HDFS caching (CACHED IN clause):

If you specify the CACHED IN clause, any existing or future data files in the table directory or the partition subdirectories are designated to be loaded into memory with the HDFS caching mechanism.

In Impala 2.2 and higher, the optional WITH REPLICATION clause for CREATE TABLE and ALTER TABLE lets you specify a *replication factor*, the number of hosts on which to cache the same data blocks. When Impala processes a cached data block, where the cache replication factor is greater than 1, Impala randomly selects a host that has a cached copy of that data block. This optimization avoids excessive CPU usage on a single host when the same cached data block is processed multiple times. Cloudera recommends specifying a value greater than or equal to the HDFS block replication factor.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

The following sections show examples of the use cases for various ALTER TABLE clauses.

To rename a table (RENAME TO clause):

The RENAME TO clause lets you change the name of an existing table, and optionally which database it is located in.

For internal tables, this operation physically renames the directory within HDFS that contains the data files; the original directory name no longer exists. By qualifying the table names with database names, you can use this technique to move an internal table (and its associated data directory) from one database to another. For example:

```
create database d1;
create database d2;
create database d3;
use d1;
create table mobile (x int) TBLPROPERTIES ('transactional'='false');
use d2;
-- Move table from another database to the current one.
alter table d1.mobile rename to mobile;
use d1;
-- Move table from one database to another.
alter table d2.mobile rename to d3.mobile;
```

To change the owner of a table:

```
ALTER TABLE NAME SET OWNER USER USER_NAME;
ALTER TABLE NAME SET OWNER ROLE ROLE_NAME;
```

The table owner is originally set to the user who creates the table. When object ownership is enabled in Ranger, an owner of a table can have the ALL with GRANT or ALL without GRANT privilege. The term OWNER is used to differentiate between the ALL privilege that is explicitly granted via the GRANT statement and a privilege that is implicitly granted by the CREATE TABLE statement.

Use the ALTER TABLE SET OWNER to transfer the ownership from the current owner to another user or a role.

To change the physical location where Impala looks for data files associated with a table or partition:

```
ALTER TABLE TABLE_NAME [PARTITION (PARTITION_SPEC)] SET LOCATION
 'HDFS_PATH_OF_DIRECTORY';
```

The path you specify is the full HDFS path where the data files reside, or will be created. Impala does not create any additional subdirectory named after the table. Impala does not move any data files to this new location or change any data files that might already exist in that directory.

To set the location for a single partition, include the PARTITION clause. Specify all the same partitioning columns for the table, with a constant value for each, to precisely identify the single partition affected by the statement:

```
create table p1 (s string) partitioned by (month int, day int);
-- Each ADD PARTITION clause creates a subdirectory in HDFS.
alter table p1 add partition (month=1, day=1);
alter table p1 add partition (month=1, day=2);
alter table p1 add partition (month=2, day=1);
alter table p1 add partition (month=2, day=2);
-- Redirect queries, INSERT, and LOAD DATA for one partition
-- to a specific different directory.
alter table p1 partition (month=1, day=1) set location '/usr/external_data/n
ew_years_day';
```

Note: If you are creating a partition for the first time and specifying its location, for maximum efficiency, use a single ALTER TABLE statement including both the ADD PARTITION and LOCATION clauses, rather than separate statements with ADD PARTITION and SET LOCATION clauses.

To automatically detect new partition directories added through Hive or HDFS operations:

In Impala 2.3 and higher, the RECOVER PARTITIONS clause scans a partitioned table to detect if any new partition directories were added outside of Impala, such as by Hive ALTER TABLE statements or by `hdfs dfs` or `hadoop fs` commands. The RECOVER PARTITIONS clause automatically recognizes any data files present in these new directories, the same as the REFRESH statement does.

For example, here is a sequence of examples showing how you might create a partitioned table in Impala, create new partitions through Hive, copy data files into the new partitions with the `hdfs` command, and have Impala recognize the new partitions and new data:

In Impala, create the table, and a single partition for demonstration purposes:

```
create database recover_partitions;
use recover_partitions;
create table t1 (s string) partitioned by (yy int, mm int);
insert into t1 partition (yy = 2016, mm = 1) values ('Partition exists');
show files in t1;
+-----------------------------------------------------------------------+---
---+--------------+
| Path                                                                  | Size
 | Partition    |
+-----------------------------------------------------------------------+---
---+--------------+
| /user/hive/warehouse/recover_partitions.db/t1/yy=2016/mm=1/data.txt | 17B
 | yy=2016/mm=1 |
+-----------------------------------------------------------------------+---
---+--------------+
quit;
```

In Hive, create some new partitions. In a real use case, you might create the partitions and populate them with data as the final stages of an ETL pipeline.

```
hive> use recover_partitions;
OK
hive> alter table t1 add partition (yy = 2016, mm = 2);
OK
hive> alter table t1 add partition (yy = 2016, mm = 3);
OK
hive> quit;
```

For demonstration purposes, manually copy data (a single row) into these new partitions, using manual HDFS operations:

```
$ hdfs dfs -ls /user/hive/warehouse/recover_partitions.db/t1/yy=2016/
Found 3 items
drwxr-xr-x - impala   hive 0 2016-05-09 16:06 /user/hive/warehouse/recover
_partitions.db/t1/yy=2016/mm=1
drwxr-xr-x - jrussell hive 0 2016-05-09 16:14 /user/hive/warehouse/recove
r_partitions.db/t1/yy=2016/mm=2
drwxr-xr-x - jrussell hive 0 2016-05-09 16:13 /user/hive/warehouse/recover_p
artitions.db/t1/yy=2016/mm=3

$ hdfs dfs -cp /user/hive/warehouse/recover_partitions.db/t1/yy=2016/mm=1/
data.txt \
  /user/hive/warehouse/recover_partitions.db/t1/yy=2016/mm=2/data.txt
$ hdfs dfs -cp /user/hive/warehouse/recover_partitions.db/t1/yy=2016/mm=1/d
ata.txt \
  /user/hive/warehouse/recover_partitions.db/t1/yy=2016/mm=3/data.txt
```

```
hive> select * from t1;
OK
Partition exists  2016  1
Partition exists  2016  2
Partition exists  2016  3
hive> quit;
```

In Impala, initially the partitions and data are not visible. Running ALTER TABLE with the RECOVER PARTITIONS clause scans the table data directory to find any new partition directories, and the data files inside them:

```
select * from t1;
+------------------+------+----+
| s                | yy   | mm |
+------------------+------+----+
| Partition exists | 2016 | 1  |
+------------------+------+----+

alter table t1 recover partitions;
select * from t1;
+------------------+------+----+
```

```
| s                | yy   | mm |
+------------------+------+----+
| Partition exists | 2016 | 1  |
| Partition exists | 2016 | 3  |
| Partition exists | 2016 | 2  |
+------------------+------+----+
```

To change the key-value pairs of the TBLPROPERTIES and SERDEPROPERTIES fields:

```
ALTER TABLE TABLE_NAME SET TBLPROPERTIES ('KEY1'='VALUE1',
 'KEY2'='VALUE2'[, ...]);
ALTER TABLE TABLE_NAME SET SERDEPROPERTIES ('KEY1'='VALUE1',
 'KEY2'='VALUE2'[, ...]);
```

The TBLPROPERTIES clause is primarily a way to associate arbitrary user-specified data items with a particular table.

The SERDEPROPERTIES clause sets up metadata defining how tables are read or written, needed in some cases by Hive but not used extensively by Impala. You would use this clause primarily to change the delimiter in an existing text table or partition, by setting the 'serialization.format' and 'field.delim' property values to the new delimiter character. The SERDEPROPERTIES clause does not change the existing data in the table. The change only affects the future inserts into the table.

Use the DESCRIBE FORMATTED statement to see the current values of these properties for an existing table.

To manually set or update table or column statistics:

Although for most tables the COMPUTE STATS or COMPUTE INCREMENTAL STATS statement is all you need to keep table and column statistics up to date for a table, sometimes for a very large table or one that is updated frequently, the length of time to recompute all the statistics might make it impractical to run those statements as often as needed. As a workaround, you can use the ALTER TABLE statement to set table statistics at the level of the entire table or a single partition, or column statistics at the level of the entire table.

You can set the numrows value for table statistics by changing the TBLPROPERTIES setting for a table or partition. For example:

```
create table analysis_data stored as parquet as select * from raw_data;
Inserted 1000000000 rows in 181.98s
compute stats analysis_data;
insert into analysis_data select * from smaller_table_we_forgot_before;
Inserted 1000000 rows in 15.32s
-- Now there are 1001000000 rows. We can update this single data point in t
he stats.
alter table analysis_data set tblproperties('numRows'='1001000000', 'STATS
_GENERATED_VIA_STATS_TASK'='true');
```

```
-- If the table originally contained 1 million rows, and we add another part
ition with 30 thousand rows,
-- change the numRows property for the partition and the overall table.
alter table partitioned_data partition(year=2009, month=4) set tblproperties
 ('numRows'='30000', 'STATS_GENERATED_VIA_STATS_TASK'='true');
alter table partitioned_data set tblproperties ('numRows'='1030000', 'STATS_
GENERATED_VIA_STATS_TASK'='true');
```

In Impala 2.6 and higher, you can use the SET COLUMN STATS clause to set a specific stats value for a particular column.

You specify a case-insensitive symbolic name for the kind of statistics: numDVs, numNulls, avgSize, maxSize. The key names and values are both quoted. This operation applies to an entire table, not a specific partition. For example:

```
create table t1 (x int, s string);
insert into t1 values (1, 'one'), (2, 'two'), (2, 'deux');
show column stats t1;
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| x      | INT    | -1               | -1     | 4        | 4        |
| s      | STRING | -1               | -1     | -1       | -1       |
+--------+--------+------------------+--------+----------+----------+
alter table t1 set column stats x ('numDVs'='2','numNulls'='0');
alter table t1 set column stats s ('numdvs'='3','maxsize'='4');
show column stats t1;
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| x      | INT    | 2                | 0      | 4        | 4        |
| s      | STRING | 3                | -1     | 4        | -1       |
+--------+--------+------------------+--------+----------+----------+
```

To reorganize columns for a table:

You can add multiple columns at a time using the ALTER TABLE statement. If you specify the IF NOT EXISTS clause, Impala silently ignores the ADD request and does not return an error if a column with the same name exists in the table.

When you replace columns, all the original column definitions are discarded.

You might use these statements if you receive a new set of data files with different data types or columns in a different order. The data files are retained, so if the new columns are incompatible with the old ones, use INSERT OVERWRITE or LOAD DATA OVERWRITE to replace all the data before issuing any further queries.

For example, here is how you might add columns to an existing table. The first ALTER TABLE adds two new columns, and the second ALTER TABLE adds one new column. A single Impala query reads both the old and new data files, containing different numbers of columns. For any columns not present in a particular data file, all the column values are considered to be NULL.

```
create table t1 (x int);
insert into t1 values (1), (2);

alter table t1 add columns (s string, t timestamp);
insert into t1 values (3, 'three', now());

alter table t1 add columns (b boolean);
insert into t1 values (4, 'four', now(), true);

select * from t1 order by x;
+---+-------+-------------------------------+------+
| x | s     | t                             | b    |
+---+-------+-------------------------------+------+
| 1 | NULL  | NULL                          | NULL |
| 2 | NULL  | NULL                          | NULL |
| 3 | three | 2016-05-11 11:19:45.054457000 | NULL |
| 4 | four  | 2016-05-11 11:20:20.260733000 | true |
+---+-------+-------------------------------+------+
```

You might use the CHANGE clause to rename a single column, or to treat an existing column as a different type than before, such as to switch between treating a column as STRING and TIMESTAMP, or between INT and BIGINT. You can only drop a single column at a time; to drop multiple columns, issue multiple ALTER TABLE statements, or define the new set of columns with a single ALTER TABLE ... REPLACE COLUMNS statement.

The following examples show some safe operations to drop or change columns. Dropping the final column in a table lets Impala ignore the data causing any disruption to existing data files. Changing the type of a column works if

existing data values can be safely converted to the new type. The type conversion rules depend on the file format of the underlying table. For example, in a text table, the same value can be interpreted as a STRING or a numeric value, while in a binary format such as Parquet, the rules are stricter and type conversions only work between certain sizes of integers.

```
create table optional_columns (x int, y int, z int, a1 int, a2 int);
insert into optional_columns values (1,2,3,0,0), (2,3,4,100,100);
-- When the last column in the table is dropped, Impala ignores the
-- values that are no longer needed. (Dropping A1 but leaving A2
-- would cause problems, as we will see in a subsequent example.)
alter table optional_columns drop column a2;
alter table optional_columns drop column a1;

select * from optional_columns;
+---+---+---+
| x | y | z |
+---+---+---+
| 1 | 2 | 3 |
| 2 | 3 | 4 |
+---+---+---+
```

```
create table int_to_string (s string, x int);
insert into int_to_string values ('one', 1), ('two', 2);
-- What was an INT column will now be interpreted as STRING.
-- This technique works for text tables but not other file formats.
-- The second X represents the new name of the column, which we keep the s
ame.
alter table int_to_string change x x string;

-- Once the type is changed, we can insert non-integer values into the X
column
-- and treat that column as a string, for example by uppercasing or conca
tenating.
insert into int_to_string values ('three', 'trois');
select s, upper(x) from int_to_string;
+-------+----------+
| s     | upper(x) |
+-------+----------+
| one   | 1        |
| two   | 2        |
| three | TROIS    |
+-------+----------+
```

Remember that Impala does not actually do any conversion for the underlying data files as a result of ALTER TA BLE statements. If you use ALTER TABLE to create a table layout that does not agree with the contents of the underlying files, you must replace the files yourself, such as using LOAD DATA to load a new set of data files, or INSERT OVERWRITE to copy from another table and replace the original data.

The following example shows what happens if you delete the middle column from a Parquet table containing three columns. The underlying data files still contain three columns of data. Because the columns are interpreted based on their positions in the data file instead of the specific column names, a SELECT * query now reads the first and second columns from the data file, potentially leading to unexpected results or conversion errors. For this reason, if you expect to someday drop a column, declare it as the last column in the table, where its data can be ignored by queries after the column is dropped. Or, re-run your ETL process and create new data files if you drop or change the type of a column in a way that causes problems with existing data files.

```
-- Parquet table showing how dropping a column can produce unexpected resul
ts.
```

```
create table p1 (s1 string, s2 string, s3 string);

insert into p1 values ('one', 'un', 'uno'), ('two', 'deux', 'dos'),
  ('three', 'trois', 'tres');
select * from p1;
+-------+-------+------+
| s1    | s2    | s3   |
+-------+-------+------+
| one   | un    | uno  |
| two   | deux  | dos  |
| three | trois | tres |
+-------+-------+------+

alter table p1 drop column s2;
-- The S3 column contains unexpected results.
-- Because S2 and S3 have compatible types, the query reads
-- values from the dropped S2, because the existing data files
-- still contain those values as the second column.
select * from p1;
+-------+-------+
| s1    | s3    |
+-------+-------+
| one   | un    |
| two   | deux  |
| three | trois |
+-------+-------+
```

```
-- Parquet table showing how dropping a column can produce conversion error
s.
create table p2 (s1 string, x int, s3 string);

insert into p2 values ('one', 1, 'uno'), ('two', 2, 'dos'), ('three', 3,
'tres');
select * from p2;
+-------+---+------+
| s1    | x | s3   |
+-------+---+------+
| one   | 1 | uno  |
| two   | 2 | dos  |
| three | 3 | tres |
+-------+---+------+

alter table p2 drop column x;
select * from p2;
WARNINGS:
File 'HDFS_FILENAME' has an incompatible Parquet schema for column 'add_col
umns.p2.s3'.
Column type: STRING, Parquet schema:
optional int32 x [i:1 d:1 r:0]

File 'HDFS_FILENAME' has an incompatible Parquet schema for column 'add_col
umns.p2.s3'.
Column type: STRING, Parquet schema:
optional int32 x [i:1 d:1 r:0]
```

In Impala 2.6 and higher, if an Avro table is created without column definitions in the CREATE TABLE statement, and columns are later added through ALTER TABLE, the resulting table is now queryable. Missing values from the newly added columns now default to NULL.

To change the file format that Impala expects data to be in, for a table or partition:

Use an ALTER TABLE ... SET FILEFORMAT clause. You can include an optional PARTITION (*COL1=VAL1*, *COL2=VAL2*, ... clause so that the file format is changed for a specific partition rather than the entire table.

Because this operation only changes the table metadata, you must do any conversion of existing data using regular Hadoop techniques outside of Impala. Any new data created by the Impala INSERT statement will be in the new format. You cannot specify the delimiter for Text files; the data files must be comma-delimited.

To set the file format for a single partition, include the PARTITION clause. Specify all the same partitioning columns for the table, with a constant value for each, to precisely identify the single partition affected by the statement:

```
create table p1 (s string) partitioned by (month int, day int);
-- Each ADD PARTITION clause creates a subdirectory in HDFS.
alter table p1 add partition (month=1, day=1);
alter table p1 add partition (month=1, day=2);
alter table p1 add partition (month=2, day=1);
alter table p1 add partition (month=2, day=2);
-- Queries and INSERT statements will read and write files
-- in this format for this specific partition.
alter table p1 partition (month=2, day=2) set fileformat parquet;
```

To change the row format with different delimiter characters:

Use the SET ROW FORMAT DELIMITED clause to ingest data files that use a different delimiter character or a different line end character. When specifying delimiter and line end characters with the FIELDS TERMINATED BY, ESCAPED BY, and LINES TERMINATED BY clauses, you can use the following:

- A regular ASCII character surrounded by single or double quotation marks.
- An octal sequence, such as '\054' representing a comma or '\0' for ASCII null (hex 00).
- Special characters, such as:
  - '\t' for tab
  - '\n' for newline or linefeed
  - '\r' for carriage return
- An integer in the range '-127'..'128' (with quotation marks but no backslash)

  Negative values are subtracted from 256. For example, FIELDS TERMINATED BY '-2' sets the field delimiter to ASCII code 254.

For the ESCAPED BY clause, choose an escape character that is not used anywhere else in the file. The character following the escape character is taken literally as part of a field value.

Surrounding field values with quotation marks does not help Impala to parse fields with embedded delimiter characters as the quotation marks are considered to be part of the column value.

If you want to use \ as the escape character, specify the clause in `impala-shell` as ESCAPED BY '\\'.

To add or drop partitions for a table, the table must already be partitioned (that is, created with a PARTITIONED BY clause). The partition is a physical directory in HDFS, with a name that encodes a particular column value (the partition key). The Impala INSERT statement already creates the partition if necessary, so the ALTER TABLE ... ADD PARTITION is primarily useful for importing data by moving or copying existing data files into the HDFS directory corresponding to a partition. (You can use the LOAD DATA statement to move files into the partition directory, or ALTER TABLE ... PARTITION (...) SET LOCATION to point a partition at a directory that already contains data files.

The DROP PARTITION clause is used to remove the HDFS directory and associated data files for a particular set of partition key values; for example, if you always analyze the last 3 months worth of data, at the beginning of each month you might drop the oldest partition that is no longer needed. Removing partitions reduces the amount of metadata associated with the table and the complexity of calculating the optimal query plan, which can simplify and speed up queries on partitioned tables, particularly join queries. Here is an example showing the ADD PARTITION and DROP PARTITION clauses.

To avoid errors while adding or dropping partitions whose existence is not certain, add the optional IF [NOT] EXISTS clause between the ADD or DROP keyword and the PARTITION keyword. That is, the entire clause becomes ADD IF NOT EXISTS PARTITION or DROP IF EXISTS PARTITION. The following example shows how partitions can be created automatically through INSERT statements, or manually through ALTER TABLE statements. The IF [NOT] EXISTS clauses let the ALTER TABLE statements succeed even if a new requested partition already exists, or a partition to be dropped does not exist.

Inserting 2 year values creates 2 partitions:

```
create table partition_t (s string) partitioned by (y int);
insert into partition_t (s,y) values ('two thousand',2000), ('nineteen nine
ty',1990);
show partitions partition_t;
+-------+-------+--------+------+--------------+-------------------+--------
+-------------------+
| y     | #Rows | #Files | Size | Bytes Cached | Cache Replication | Format
 | Incremental stats |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
| 1990  | -1    | 1      | 16B  | NOT CACHED   | NOT CACHED        | TEXT
 | false |
| 2000  | -1    | 1      | 13B  | NOT CACHED   | NOT CACHED        | TEXT
| false |
| Total | -1    | 2      | 29B  | 0B           |                   |
 |       |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
```

Without the IF NOT EXISTS clause, an attempt to add a new partition might fail:

```
alter table partition_t add partition (y=2000);
ERROR: AnalysisException: Partition spec already exists: (y=2000).
```

The IF NOT EXISTS clause makes the statement succeed whether or not there was already a partition with the specified key value:

```
alter table partition_t add if not exists partition (y=2000);
alter table partition_t add if not exists partition (y=2010);
show partitions partition_t;
+-------+-------+--------+------+--------------+-------------------+--------
+-------------------+
| y     | #Rows | #Files | Size | Bytes Cached | Cache Replication | Format
| Incremental stats |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
| 1990  | -1    | 1      | 16B  | NOT CACHED   | NOT CACHED        | TEXT
| false |
| 2000  | -1    | 1      | 13B  | NOT CACHED   | NOT CACHED        | TEXT
 | false |
| 2010  | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT
| false |
| Total | -1    | 2      | 29B  | 0B           |                   |
 |       |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
```

Likewise, the IF EXISTS clause lets DROP PARTITION succeed whether or not the partition is already in the table:

```
alter table partition_t drop if exists partition (y=2000);
alter table partition_t drop if exists partition (y=1950);
show partitions partition_t;
+-------+-------+--------+------+--------------+-------------------+--------
+-------------------+
| y     | #Rows | #Files | Size | Bytes Cached | Cache Replication | Format
 | Incremental stats |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
| 1990  | -1    | 1      | 16B  | NOT CACHED   | NOT CACHED        | TEXT
 | false |
| 2010  | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT
| false |
| Total | -1    | 1      | 16B  | 0B           |                   |
 |       |
+-------+-------+--------+------+--------------+-------------------+--------
+-------+
```

The optional PURGE keyword, available in Impala 2.3 and higher, is used with the DROP PARTITION clause to remove associated HDFS data files immediately rather than going through the HDFS trashcan mechanism. Use this keyword when dropping a partition if it is crucial to remove the data as quickly as possible to free up space, or if there is a problem with the trashcan, such as the trash cannot being configured or being in a different HDFS encryption zone than the data files.

```
-- Create an empty table and define the partitioning scheme.
create table part_t (x int) partitioned by (month int);
-- Create an empty partition into which you could copy data files from some
 other source.
alter table part_t add partition (month=1);
-- After changing the underlying data, issue a REFRESH statement to make
the data visible in Impala.
refresh part_t;
-- Later, do the same for the next month.
alter table part_t add partition (month=2);

-- Now you no longer need the older data.
alter table part_t drop partition (month=1);
-- If the table was partitioned by month and year, you would issue a state
ment like:
-- alter table part_t drop partition (year=2003,month=1);
-- which would require 12 ALTER TABLE statements to remove a year's worth of
 data.
-- If the data files for subsequent months were in a different file format,
-- you could set a different file format for the new partition as you create
 it.
alter table part_t add partition (month=3) set fileformat=parquet;
```

The value specified for a partition key can be an arbitrary constant expression, without any references to columns. For example:

```
alter table time_data add partition (month=concat('Decem','ber'));
alter table sales_data add partition (zipcode = cast(9021 * 10 as string));
```

Note:

An alternative way to reorganize a table and its associated data files is to use CREATE TABLE to create a variation of the original table, then use INSERT to copy the transformed or reordered data to the new table. The advantage of ALTER TABLE is that it avoids making a duplicate copy of the data files, allowing you to reorganize huge volumes of data in a space-efficient way using familiar Hadoop techniques.

To switch a table between internal and external:

You can switch a table from internal to external, or from external to internal, by using the ALTER TABLE statement:

```
-- Switch a table from internal to external.
ALTER TABLE TABLE_NAME SET TBLPROPERTIES('EXTERNAL'='TRUE');

-- Switch a table from external to internal.
ALTER TABLE TABLE_NAME SET TBLPROPERTIES('EXTERNAL'='FALSE');
```

If the Kudu service is integrated with the Hive Metastore, the above operations are not supported.

Cancellation: Cannot be cancelled.

HDFS permissions:

Most ALTER TABLE clauses do not actually read or write any HDFS files, and so do not depend on specific HDFS permissions. For example, the SET FILEFORMAT clause does not actually check the file format existing data files or convert them to the new format, and the SET LOCATION clause does not require any special permissions on the new location. (Any permission-related failures would come later, when you actually query or insert into the table.)

In general, ALTER TABLE clauses that do touch HDFS files and directories require the same HDFS permissions as corresponding CREATE, INSERT, or SELECT statements. The permissions allow the user ID that the `impalad` daemon runs under, typically the impala user, to read or write files or directories, or (in the case of the execute bit) descend into a directory. The RENAME TO clause requires read, write, and execute permission in the source and destination database directories and in the table data directory, and read and write permission for the data files within the table. The ADD PARTITION and DROP PARTITION clauses require write and execute permissions for the associated partition directory.

Kudu considerations:

Because of the extra constraints and features of Kudu tables, such as the NOT NULL and DEFAULT attributes for columns, ALTER TABLE has specific requirements related to Kudu tables:

- In an ADD COLUMNS operation, you can specify the NULL, NOT NULL, and DEFAULT *DEFAULT_VALUE* column attributes.
- In Impala 2.9 and higher, you can also specify the ENCODING, COMPRESSION, and BLOCK_SIZE attributes when adding a column.
- If you add a column with a NOT NULL attribute, it must also have a DEFAULT attribute, so the default value can be assigned to that column for all existing rows.
- The DROP COLUMN clause works the same for a Kudu table as for other kinds of tables.
- Although you can change the name of a column with the CHANGE clause, you cannot change the type of a column in a Kudu table.
- You cannot change the nullability of existing columns in a Kudu table.
- In Impala 2.10, you can change the default value, encoding, compression, or block size of existing columns in a Kudu table by using the SET clause.
- You cannot use the REPLACE COLUMNS clause with a Kudu table.
- The RENAME TO clause for a Kudu table only affects the name stored in the metastore database that Impala uses to refer to the table. To change which underlying Kudu table is associated with an Impala table name, you must change the TBLPROPERTIES property of the table: SET TBLPROPERTIES('kudu.table_name '='*KUDU_TBL_NAME*'). You can only change underlying Kudu tables for the external tables.

The following are some examples of using the ADD COLUMNS clause for a Kudu table:

```
CREATE TABLE t1 ( x INT, PRIMARY KEY (x) )
  PARTITION BY HASH (x) PARTITIONS 16
  STORED AS KUDU

ALTER TABLE t1 ADD COLUMNS (y STRING ENCODING prefix_encoding);
ALTER TABLE t1 ADD COLUMNS (z INT DEFAULT 10);
```

```
ALTER TABLE t1 ADD COLUMNS (a STRING NOT NULL DEFAULT '', t TIMESTAMP COM
PRESSION default_compression);
```

The following are some examples of modifying column defaults and storage attributes for a Kudu table:

```
create table kt (x bigint primary key, s string default 'yes', t timestamp)
  stored as kudu;

-- You can change the default value for a column, which affects any rows
-- inserted after this change is made.
alter table kt alter column s set default 'no';
-- You can remove the default value for a column, which affects any rows
-- inserted after this change is made. If the column is nullable, any
-- future inserts default to NULL for this column. If the column is marked
-- NOT NULL, any future inserts must specify a value for the column.
alter table kt alter column s drop default;
insert into kt values (1, 'foo', now());
-- Because of the DROP DEFAULT above, omitting S from the insert
-- gives it a value of NULL.
insert into kt (x, t) values (2, now());

select * from kt;
+---+------+-------------------------------+
| x | s    | t                             |
+---+------+-------------------------------+
| 2 | NULL | 2017-10-02 00:03:40.652156000 |
| 1 | foo  | 2017-10-02 00:03:04.346185000 |
+---+------+-------------------------------+

-- Other storage-related attributes can also be changed for columns.
-- These changes take effect for any newly inserted rows, or rows
-- rearranged due to compaction after deletes or updates.
alter table kt alter column s set encoding prefix_encoding;
-- The COLUMN keyword is optional in the syntax.
alter table kt alter x set block_size 2048;
alter table kt alter column t set compression zlib;

desc kt;
+------+-----------+---------+-------------+----------+---------------+--
---------------+---------------------+------------+
| name | type      | comment | primary_key | nullable | default_value | enco
ding        | compression         | block_size |
+------+-----------+---------+-------------+----------+---------------+----
-------------+---------------------+------------+
| x    | bigint    |         | true        | false    |               | AU
TO_ENCODING   | DEFAULT_COMPRESSION | 2048       |
| s    | string    |         | false       | true     |               | P
REFIX_ENCODING | DEFAULT_COMPRESSION | 0          |
| t    | timestamp |         | false       | true     |               | AUTO
_ENCODING   | ZLIB                | 0          |
+------+-----------+---------+-------------+----------+---------------+----
-------------+---------------------+------------+
```

Kudu tables all use an underlying partitioning mechanism. The partition syntax is different than for non-Kudu tables. You can use the ALTER TABLE statement to add and drop *range partitions* from a Kudu table. Any new range must not overlap with any existing ranges. Dropping a range removes all the associated rows from the table.

Related Information

Impala with Amazon S3

ALTER VIEW statement

The ALTER VIEW statement changes the characteristics of a view.

Because a view is a logical construct, an alias for a query, with no physical data behind it, ALTER VIEW only involves changes to metadata in the metastore database, not any data files in HDFS.

To see the definition of the updated view, issue a DESCRIBE FORMATTED statement.

Syntax:

```
ALTER VIEW [DATABASE_NAME.]VIEW_NAME
   [(COLUMN_NAME [COMMENT 'COLUMN_COMMENT'][, ...])]
   AS SELECT_STATEMENT;

ALTER VIEW [DATABASE_NAME.]VIEW_NAME
   RENAME TO [DATABASE_NAME.]VIEW_NAME;

ALTER VIEW [DATABASE_NAME.]VIEW_NAME SET OWNER USER user_name;
ALTER VIEW [DATABASE_NAME.]VIEW_NAME SET OWNER ROLE role_name;

ALTER VIEW [DATABASE_NAME.]VIEW_NAME
    SET TBLPROPERTIES ('NAME' = 'VALUE'[, 'NAME' = 'VALUE' ...]);

ALTER VIEW [DATABASE_NAME.]VIEW_NAME
    UNSET TBLPROPERTIES ('NAME'[, ...]);
```

- The AS clause associates the view with a different query.

  An optional list of column names can be specified with or without the column-level comments.

  For example:

  ```
  ALTER VIEW v1 AS SELECT x, UPPER(s) s FROM t2;
  ALTER VIEW v1 (c1, c2) AS SELECT x, UPPER(s) s FROM t2;
  ALTER VIEW v7 (c1 COMMENT 'Comment for c1', c2) AS SELECT t1.c1, t1.c2 F
  ROM t1;
  ```

- The RENAME TO clause changes the name of the view, moves the view to a different database, or both.

  For example:

  ```
  ALTER VIEW db1.v1 RENAME TO db2.v2; -- Move the view to a different data
  base with a new name.
  ALTER VIEW db1.v1 RENAME TO db1.v2;  -- Rename the view in the same databa
  se.
  ALTER VIEW db1.v1 RENAME TO db2.v1; -- Move the view to a difference dat
  abase with the same view name.
  ```

- The SET OWNER clause transfers the ownership of the view from the current owner to another user or a role.

  The view owner is originally set to the user who creates the view. When object ownership is enabled in Ranger, an owner of a view can have the ALL with GRANT or ALL without GRANT privilege. The term OWNER is used to differentiate between the ALL privilege that is explicitly granted via the GRANT statement and a privilege that is implicitly granted by the CREATE VIEW statement.

- The SET TBLPROPERTIES clause is primarily a way to associate arbitrary user-specified data items with a particular view.

  You can associate arbitrary items of metadata with a table by specifying the TBLPROPERTIES clause. This clause takes a comma-separated list of key-value pairs and stores those items in the metastore database. You can also unset the view properties later with an UNSET TBLPROPERTIES clause.

  For example:

  ```
  ALTER VIEW v1 SET TBLPROPERTIES ('tblp1' = '1', 'tblp2' = '2');
  ALTER VIEW v1 UNSET TBLPROPERTIES ('tblp1', 'tblp2');
  ```

Statement type: DDL

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

COMMENT statement

The COMMENT statement adds, changes, or removes a comment about a database, a table, or a column.

You can alternatively use the CREATE and ALTER statements to add comments to the objects.

You can view the comment on a database, a table, or a column using the SHOW or DESCRIBE statement.

Syntax:

```
COMMENT ON DATABASE DB_NAME IS {'COMMENT' | NULL}
```

```
COMMENT ON TABLE [DB_NAME.]TABLE_NAME IS {'COMMENT' | NULL}
```

```
COMMENT ON COLUMN [DB_NAME.]TABLE_NAME.COLUMN_NAME IS {'COMMENT' | NULL}
```

Parameters:

- *DB_NAME*: Specify the database name if not for the current database.
- NULL: If given for the comment, removes the existing comment.
- The *COMMENT* string can be up to 256 characters long.

Privileges required:

To add a comment, the ALTER privilege on the object is required.

To view a comment, the SELECT, INSERT, or REFRESH on the object is required.

Usage notes:

Added in: Impala 3.1 release.

COMPUTE STATS statement

The COMPUTE STATS statement gathers information about volume and distribution of data in a table and all associated columns and partitions. The information is stored in the metastore database, and used by Impala to help optimize queries.

For example, if Impala can determine that a table is large or small, or has many or few distinct values it can organize and parallelize the work appropriately for a join query or insert operation. For details about the kinds of information gathered by this statement, see the *Table and column statistics* topic.

Syntax:

```
COMPUTE STATS [DB_NAME.]TABLE_NAME  [ ( COLUMN_LIST ) ] [TABLESAMPLE
 SYSTEM(PERCENTAGE) [REPEATABLE(SEED)]]

COLUMN_LIST ::= COLUMN_NAME [ , COLUMN_NAME, ... ]
COMPUTE INCREMENTAL STATS [DB_NAME.]TABLE_NAME [PARTITION (PARTITION_SPEC)]

PARTITION_SPEC ::= PARTITION_COL=CONSTANT_VALUE

PARTITION_SPEC ::= SIMPLE_PARTITION_SPEC | COMPLEX_PARTITION_SPEC

SIMPLE_PARTITION_SPEC ::= PARTITION_COL=CONSTANT_VALUE

COMPLEX_PARTITION_SPEC ::= COMPARISON_EXPRESSION_ON_PARTITION_COL
```

The PARTITION clause is only allowed in combination with the INCREMENTAL clause. It is optional for COMP UTE INCREMENTAL STATS, and required for DROP INCREMENTAL STATS. Whenever you specify partitions through the PARTITION (*PARTITION_SPEC*) clause in a COMPUTE INCREMENTAL STATS or DROP INC REMENTAL STATS statement, you must include all the partitioning columns in the specification, and specify constant values for all the partition key columns.

Usage notes:

Originally, Impala relied on users to run the Hive ANALYZE TABLE statement, but that method of gathering statistics proved unreliable and difficult to use. The Impala COMPUTE STATS statement was built to improve the reliability and user-friendliness of this operation. COMPUTE STATS does not require any setup steps or special configuration. You only run a single Impala COMPUTE STATS statement to gather both table and column statistics, rather than separate Hive ANALYZE TABLE statements for each kind of statistics.

For non-incremental COMPUTE STATS statement, the columns for which statistics are computed can be specified with an optional comma-separate list of columns.

If no column list is given, the COMPUTE STATS statement computes column-level statistics for all columns of the table. This adds potentially unneeded work for columns whose stats are not needed by queries. It can be especially costly for very wide tables and unneeded large string fields.

COMPUTE STATS returns an error when a specified column cannot be analyzed, such as when the column does not exist, the column is of an unsupported type for COMPUTE STATS, e.g. colums of complex types, or the column is a partitioning column.

If an empty column list is given, no column is analyzed by COMPUTE STATS.

In Impala 2.12 and higher, an optional TABLESAMPLE clause immediately after a table reference specifies that the COMPUTE STATS operation only processes a specified percentage of the table data. For tables that are so large that a full COMPUTE STATS operation is impractical, you can use COMPUTE STATS with a TABLESAMPLE clause to extrapolate statistics from a sample of the table data. See *Table and column statistics* about the experimental stats extrapolation and sampling features.

The COMPUTE INCREMENTAL STATS variation is a shortcut for partitioned tables that works on a subset of partitions rather than the entire table. The incremental nature makes it suitable for large tables with many partitions, where a full COMPUTE STATS operation takes too long to be practical each time a partition is added or dropped.

Important:

For a particular table, use either COMPUTE STATS or COMPUTE INCREMENTAL STATS. The two kinds of stats do not interoperate with each other at the table level. Without dropping the stats, if you run COMPUTE INCREMENTAL STATS it will overwrite the full compute stats or if you run COMPUTE STATS it will drop all incremental stats for consistency.

When you run COMPUTE INCREMENTAL STATS on a table for the first time, the statistics are computed again from scratch regardless of whether the table already has statistics. Therefore, expect a one-time resource-intensive operation for scanning the entire table when running COMPUTE INCREMENTAL STATS for the first time on a given table.

In Impala 3.0 and lower, approximately 400 bytes of metadata per column per partition are needed for caching. Tables with a big number of partitions and many columns can add up to a significant memory overhead as the metadata must be cached on the `catalogd` host and on every `impalad` host that is eligible to be a coordinator. If this metadata for all tables exceeds 2 GB, you might experience service downtime. In Impala 3.1 and higher, the issue was alleviated with an improved handling of incremental stats.

COMPUTE INCREMENTAL STATS only applies to partitioned tables. If you use the INCREMENTAL clause for an unpartitioned table, Impala automatically uses the original COMPUTE STATS statement. Such tables display false under the Incremental stats column of the SHOW TABLE STATS output.

Note:

Because many of the most performance-critical and resource-intensive operations rely on table and column statistics to construct accurate and efficient plans, COMPUTE STATS is an important step at the end of your ETL process. Run COMPUTE STATS on all tables as your first step during performance tuning for slow queries, or troubleshooting for out-of-memory conditions:

- Accurate statistics help Impala construct an efficient query plan for join queries, improving performance and reducing memory usage.
- Accurate statistics help Impala distribute the work effectively for insert operations into Parquet tables, improving performance and reducing memory usage.
- Accurate statistics help Impala estimate the memory required for each query, which is important when you use resource management features, such as admission control and the YARN resource management framework. The statistics help Impala to achieve high concurrency, full utilization of available memory, and avoid contention with workloads from other Hadoop components.
- In Impala 2.8 and higher, when you run the COMPUTE STATS or COMPUTE INCREMENTAL STATS statement against a Parquet table, Impala automatically applies the query option setting MT_DOP=4 to increase the amount of intra-node parallelism during this CPU-intensive operation. See MT_DOP query option for details about what this query option does and how to use it with CPU-intensive SELECT statements.

Computing stats for groups of partitions:

In Impala 2.8 and higher, you can run COMPUTE INCREMENTAL STATS on multiple partitions, instead of the entire table or one partition at a time. You include comparison operators other than = in the PARTITION clause, and the COMPUTE INCREMENTAL STATS statement applies to all partitions that match the comparison expression.

For example, the INT_PARTITIONS table contains 4 partitions. The following COMPUTE INCREMENTAL STATS statements affect some but not all partitions, as indicated by the Updated *N* partition(s) messages. The partitions that are affected depend on values in the partition key column X that match the comparison expression in the PARTITION clause.

```
show partitions int_partitions;
+-------+-------+--------+------+--------------+-------------------+-------
--+...
| x     | #Rows | #Files | Size | Bytes Cached | Cache Replication | Format
  |...
+-------+-------+--------+------+--------------+-------------------+-------
--+...
```

```
| 99    | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | PARQ
UET |...
| 120   | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT
 |...
| 150   | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT
  |...
| 200   | -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT
 |...
| Total | -1    | 0      | 0B   | 0B           |                   |
  |...
+-------+-------+--------+------+--------------+-------------------+-------
--+...

compute incremental stats int_partitions partition (x < 100);
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 1 column(s). |
+-----------------------------------------+

compute incremental stats int_partitions partition (x in (100, 150, 200));
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 2 partition(s) and 1 column(s). |
+-----------------------------------------+

compute incremental stats int_partitions partition (x between 100 and 175);
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 2 partition(s) and 1 column(s). |
+-----------------------------------------+

compute incremental stats int_partitions partition (x in (100, 150, 200) or
x < 100);
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 3 partition(s) and 1 column(s). |
+-----------------------------------------+

compute incremental stats int_partitions partition (x != 150);
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 3 partition(s) and 1 column(s). |
+-----------------------------------------+
```

Complex type considerations:

Currently, the statistics created by the COMPUTE STATS statement do not include information about complex type columns. The column stats metrics for complex columns are always shown as -1. For queries involving complex type columns, Impala uses heuristics to estimate the data distribution within such columns.

HBase considerations:

COMPUTE STATS works for HBase tables also. The statistics gathered for HBase tables are somewhat different than for HDFS-backed tables, but that metadata is still used for optimization when HBase tables are involved in join queries.

Amazon S3 considerations:

COMPUTE STATS also works for tables where data resides in the Amazon Simple Storage Service (S3).

Performance considerations:

The statistics collected by COMPUTE STATS are used to optimize join queries INSERT operations into Parquet tables, and other resource-intensive kinds of SQL statements.

For large tables, the COMPUTE STATS statement itself might take a long time and you might need to tune its performance. The COMPUTE STATS statement does not work with the EXPLAIN statement, or the SUMMARY command in `impala-shell`. You can use the PROFILE statement in `impala-shell` to examine timing information for the statement as a whole. If a basic COMPUTE STATS statement takes a long time for a partitioned table, consider switching to the COMPUTE INCREMENTAL STATS syntax so that only newly added partitions are analyzed each time.

Examples:

This example shows two tables, T1 and T2, with a small number distinct values linked by a parent-child relationship between T1.ID and T2.PARENT. T1 is tiny, while T2 has approximately 100K rows. Initially, the statistics includes physical measurements such as the number of files, the total size, and size measurements for fixed-length columns such as with the INT type. Unknown values are represented by -1. After running COMPUTE STATS for each table, much more information is available through the SHOW STATS statements. If you were running a join query involving both of these tables, you would need statistics for both tables to get the most effective optimization for the query.

```
[localhost:21000] > show table stats t1;
Query: show table stats t1
+-------+--------+------+--------+
| #Rows | #Files | Size | Format |
+-------+--------+------+--------+
| -1    | 1      | 33B  | TEXT   |
+-------+--------+------+--------+
Returned 1 row(s) in 0.02s
[localhost:21000] > show table stats t2;
Query: show table stats t2
+-------+--------+----------+--------+
| #Rows | #Files | Size     | Format |
+-------+--------+----------+--------+
| -1    | 28     | 960.00KB | TEXT   |
+-------+--------+----------+--------+
Returned 1 row(s) in 0.01s
[localhost:21000] > show column stats t1;
Query: show column stats t1
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| id     | INT    | -1               | -1     | 4        | 4        |
| s      | STRING | -1               | -1     | -1       | -1       |
+--------+--------+------------------+--------+----------+----------+
Returned 2 row(s) in 1.71s
[localhost:21000] > show column stats t2;
Query: show column stats t2
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| parent | INT    | -1               | -1     | 4        | 4        |
| s      | STRING | -1               | -1     | -1       | -1       |
+--------+--------+------------------+--------+----------+----------+
Returned 2 row(s) in 0.01s
[localhost:21000] > compute stats t1;
Query: compute stats t1
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
```

```
| Updated 1 partition(s) and 2 column(s). |
+-----------------------------------------+
Returned 1 row(s) in 5.30s
[localhost:21000] > show table stats t1;
Query: show table stats t1
+-------+--------+------+--------+
| #Rows | #Files | Size | Format |
+-------+--------+------+--------+
| 3     | 1      | 33B  | TEXT   |
+-------+--------+------+--------+
Returned 1 row(s) in 0.01s
[localhost:21000] > show column stats t1;
Query: show column stats t1
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| id     | INT    | 3                | -1     | 4        | 4        |
| s      | STRING | 3                | -1     | -1       | -1       |
+--------+--------+------------------+--------+----------+----------+
Returned 2 row(s) in 0.02s
[localhost:21000] > compute stats t2;
Query: compute stats t2
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 2 column(s). |
+-----------------------------------------+
Returned 1 row(s) in 5.70s
[localhost:21000] > show table stats t2;
Query: show table stats t2
+-------+--------+----------+--------+
| #Rows | #Files | Size     | Format |
+-------+--------+----------+--------+
| 98304 | 1      | 960.00KB | TEXT   |
+-------+--------+----------+--------+
Returned 1 row(s) in 0.03s
[localhost:21000] > show column stats t2;
Query: show column stats t2
+--------+--------+------------------+--------+----------+----------+
| Column | Type   | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+--------+------------------+--------+----------+----------+
| parent | INT    | 3                | -1     | 4        | 4        |
| s      | STRING | 6                | -1     | 14       | 9.3      |
+--------+--------+------------------+--------+----------+----------+
Returned 2 row(s) in 0.01s
```

The following example shows how to use the INCREMENTAL clause, available in Impala 2.1.0 and higher. The COMPUTE INCREMENTAL STATS syntax lets you collect statistics for newly added or changed partitions, without rescanning the entire table.

```
-- Initially the table has no incremental stats, as indicated
-- 'false' under Incremental stats.
show table stats item_partitioned;
+-------------+-------+--------+----------+--------------+---------+-------
-----------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Increme
ntal stats
+-------------+-------+--------+----------+--------------+---------+-----
-------------
| Books       | -1    | 1      | 223.74KB | NOT CACHED   | PARQUET | false
| Children    | -1    | 1      | 230.05KB | NOT CACHED   | PARQUET | false
| Electronics | -1    | 1      | 232.67KB | NOT CACHED   | PARQUET | false
| Home        | -1    | 1      | 232.56KB | NOT CACHED   | PARQUET | false
```

```
| Jewelry     | -1    | 1      | 223.72KB | NOT CACHED   | PARQUET | false
| Men         | -1    | 1      | 231.25KB | NOT CACHED   | PARQUET | false
| Music       | -1    | 1      | 237.90KB | NOT CACHED   | PARQUET | false
| Shoes       | -1    | 1      | 234.90KB | NOT CACHED   | PARQUET | false
| Sports      | -1    | 1      | 227.97KB | NOT CACHED   | PARQUET | false
| Women       | -1    | 1      | 226.27KB | NOT CACHED   | PARQUET | false
| Total       | -1    | 10     | 2.25MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+-------
-----------
-- After the first COMPUTE INCREMENTAL STATS,
-- all partitions have stats. The first
-- COMPUTE INCREMENTAL STATS scans the whole
-- table, discarding any previous stats from
-- a traditional COMPUTE STATS statement.
compute incremental stats item_partitioned;
+-------------------------------------------+
| summary                                   |
+-------------------------------------------+
| Updated 10 partition(s) and 21 column(s). |
+-------------------------------------------+
show table stats item_partitioned;
+-------------+-------+--------+----------+--------------+---------+--------
----------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Incr
emental stats
+-------------+-------+--------+----------+--------------+---------+------
------------
| Books       | 1733  | 1      | 223.74KB | NOT CACHED   | PARQUET | true
| Children    | 1786  | 1      | 230.05KB | NOT CACHED   | PARQUET | true
| Electronics | 1812  | 1      | 232.67KB | NOT CACHED   | PARQUET | true
| Home        | 1807  | 1      | 232.56KB | NOT CACHED   | PARQUET | true
| Jewelry     | 1740  | 1      | 223.72KB | NOT CACHED   | PARQUET | true
| Men         | 1811  | 1      | 231.25KB | NOT CACHED   | PARQUET | true
| Music       | 1860  | 1      | 237.90KB | NOT CACHED   | PARQUET | true
| Shoes       | 1835  | 1      | 234.90KB | NOT CACHED   | PARQUET | true
| Sports      | 1783  | 1      | 227.97KB | NOT CACHED   | PARQUET | true
| Women       | 1790  | 1      | 226.27KB | NOT CACHED   | PARQUET | true
| Total       | 17957 | 10     | 2.25MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+------
------------

-- Add a new partition...
alter table item_partitioned add partition (i_category='Camping');
-- Add or replace files in HDFS outside of Impala,
-- rendering the stats for a partition obsolete.
!import_data_into_sports_partition.sh
refresh item_partitioned;
drop incremental stats item_partitioned partition (i_category='Sports');
-- Now some partitions have incremental stats
-- and some do not.
show table stats item_partitioned;
+-------------+-------+--------+----------+--------------+---------+-------
-----------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Increme
ntal stats
+-------------+-------+--------+----------+--------------+---------+-----
-------------
| Books       | 1733  | 1      | 223.74KB | NOT CACHED   | PARQUET | true
| Camping     | -1    | 1      | 408.02KB | NOT CACHED   | PARQUET | false
| Children    | 1786  | 1      | 230.05KB | NOT CACHED   | PARQUET | true
| Electronics | 1812  | 1      | 232.67KB | NOT CACHED   | PARQUET | true
| Home        | 1807  | 1      | 232.56KB | NOT CACHED   | PARQUET | true
| Jewelry     | 1740  | 1      | 223.72KB | NOT CACHED   | PARQUET | true
| Men         | 1811  | 1      | 231.25KB | NOT CACHED   | PARQUET | true
```

```
| Music       | 1860  | 1      | 237.90KB | NOT CACHED   | PARQUET | true
| Shoes       | 1835  | 1      | 234.90KB | NOT CACHED   | PARQUET | true
| Sports      | -1    | 1      | 227.97KB | NOT CACHED   | PARQUET | false
| Women       | 1790  | 1      | 226.27KB | NOT CACHED   | PARQUET | true
| Total       | 17957 | 11     | 2.65MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+-----
-------------

-- After another COMPUTE INCREMENTAL STATS,
-- all partitions have incremental stats, and only the 2
-- partitions without incremental stats were scanned.
compute incremental stats item_partitioned;
+------------------------------------------+
| summary                                  |
+------------------------------------------+
| Updated 2 partition(s) and 21 column(s). |
+------------------------------------------+
show table stats item_partitioned;
+-------------+-------+--------+----------+--------------+---------+------
------------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Increm
ental stats
+-------------+-------+--------+----------+--------------+---------+--------
----------
| Books       | 1733  | 1      | 223.74KB | NOT CACHED   | PARQUET | true
| Camping     | 5328  | 1      | 408.02KB | NOT CACHED   | PARQUET | true
| Children    | 1786  | 1      | 230.05KB | NOT CACHED   | PARQUET | true
| Electronics | 1812  | 1      | 232.67KB | NOT CACHED   | PARQUET | true
| Home        | 1807  | 1      | 232.56KB | NOT CACHED   | PARQUET | true
| Jewelry     | 1740  | 1      | 223.72KB | NOT CACHED   | PARQUET | true
| Men         | 1811  | 1      | 231.25KB | NOT CACHED   | PARQUET | true
| Music       | 1860  | 1      | 237.90KB | NOT CACHED   | PARQUET | true
| Shoes       | 1835  | 1      | 234.90KB | NOT CACHED   | PARQUET | true
| Sports      | 1783  | 1      | 227.97KB | NOT CACHED   | PARQUET | true
| Women       | 1790  | 1      | 226.27KB | NOT CACHED   | PARQUET | true
| Total       | 17957 | 11     | 2.65MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+------
------------
```

File format considerations:

The COMPUTE STATS statement works with tables created with any of the file formats supported by Impala. The following considerations apply to COMPUTE STATS depending on the file format of the table.

The COMPUTE STATS statement works with text tables with no restrictions. These tables can be created through either Impala or Hive.

The COMPUTE STATS statement works with Parquet tables. These tables can be created through either Impala or Hive.

The COMPUTE STATS statement works with Avro tables without restriction in Impala 2.2 and higher. In earlier releases, COMPUTE STATS worked only for Avro tables created through Hive, and required the CREATE TABLE statement to use SQL-style column names and types rather than an Avro-style schema specification.

The COMPUTE STATS statement works with RCFile tables with no restrictions. These tables can be created through either Impala or Hive.

The COMPUTE STATS statement works with SequenceFile tables with no restrictions. These tables can be created through either Impala or Hive.

The COMPUTE STATS statement works with partitioned tables, whether all the partitions use the same file format, or some partitions are defined through ALTER TABLE to use different file formats.

Statement type: DDL

Cancellation: Certain multi-stage statements (CREATE TABLE AS SELECT and COMPUTE STATS) can be cancelled during some stages, when running INSERT or SELECT operations internally. To cancel this statement, use Ctrl-C from the `impala-shell` interpreter, the Cancel button from the Watch page in Hue, Actions > Cancel from the Queries list in Cloudera Manager, or Cancel from the list of in-flight queries (for a particular node) on the Queries tab in the Impala web UI (port 25000).

Restrictions:

Note: Prior to Impala 1.4.0, COMPUTE STATS counted the number of NULL values in each column and recorded that figure in the metastore database. Because Impala does not currently use the NULL count during query planning, Impala 1.4.0 and higher speeds up the COMPUTE STATS statement by skipping this NULL counting.

Internal details:

Behind the scenes, the COMPUTE STATS statement executes two statements: one to count the rows of each partition in the table (or the entire table if unpartitioned) through the COUNT(*) function, and another to count the approximate number of distinct values in each column through the NDV() function. You might see these queries in your monitoring and diagnostic displays. The same factors that affect the performance, scalability, and execution of other queries (such as parallel execution, memory usage, admission control, and timeouts) also apply to the queries run by the COMPUTE STATS statement.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read permission for all affected files in the source directory: all files in the case of an unpartitioned table or a partitioned table in the case of COMPUTE STATS; or all the files in partitions without incremental stats in the case of COMPUTE INCREMEN TAL STATS. It must also have read and execute permissions for all relevant directories holding the data files. (Essentially, COMPUTE STATS requires the same permissions as the underlying SELECT queries it runs against the table.)

Kudu considerations:

The COMPUTE STATS statement applies to Kudu tables. Impala does not compute the number of rows for each partition for Kudu tables. Therefore, you do not need to re-run the operation when you see -1 in the # Rows column of the output from SHOW TABLE STATS. That column always shows -1 for all Kudu tables.

CREATE DATABASE statement

The CREATE DATABASE statement is used to create a new database.

In Impala, a database is both:

- A logical construct for grouping together related tables, views, and functions within their own namespace. You might use a separate database for each application, set of related tables, or round of experimentation.
- A physical construct represented by a directory tree in HDFS. Tables (internal tables), partitions, and data files are all located under this directory. You can perform HDFS-level operations such as backing it up and measuring space usage, or remove it with a DROP DATABASE statement.

Syntax:

```
CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] DATABASE_NAME[COMMENT
 'DATABASE_COMMENT']
  [LOCATION HDFS_PATH];
```

Statement type: DDL

Usage notes:

A database is physically represented as a directory in HDFS, with a filename extension .db, under the main Impala data directory. If the associated HDFS directory does not exist, it is created for you. All databases and their associated directories are top-level objects, with no physical or logical nesting.

After creating a database, to make it the current database within an `impala-shell` session, use the USE statement. You can refer to tables in the current database without prepending any qualifier to their names.

When you first connect to Impala through `impala-shell`, the database you start in (before issuing any CREATE DATABASE or USE statements) is named default.

Impala includes another predefined database, _impala_builtins, that serves as the location for the built-in functions. To see the built-in functions, use a statement like the following:

```
show functions in _impala_builtins;
show functions in _impala_builtins like '*SUBSTRING*';
```

After creating a database, your `impala-shell` session or another `impala-shell` connected to the same node can immediately access that database. To access the database through the Impala daemon on a different node, issue the INVALIDATE METADATA statement first while connected to that other node.

Setting the LOCATION attribute for a new database is a way to work with sets of files in an HDFS directory structure outside the default Impala data directory, as opposed to setting the LOCATION attribute for each individual table.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Hive considerations:

When you create a database in Impala, the database can also be used by Hive. When you create a database in Hive, issue an INVALIDATE METADATA statement in Impala to make Impala permanently aware of the new database.

The SHOW DATABASES statement lists all databases, or the databases whose name matches a wildcard pattern. In Impala 2.5 and higher, the SHOW DATABASES output includes a second column that displays the associated comment, if any, for each database.

Amazon S3 considerations:

To specify that any tables created within a database reside on the Amazon S3 system, you can include an s3a:// prefix on the LOCATION attribute. In Impala 2.6 and higher, Impala automatically creates any required folders as the databases, tables, and partitions are created, and removes them when they are dropped.

In Impala 2.6 and higher, Impala DDL statements such as CREATE DATABASE, CREATE TABLE, DROP DATABASE CASCADE, DROP TABLE, and ALTER TABLE [ADD|DROP] PARTITION can create or remove folders as needed in the Amazon S3 system. Prior to Impala 2.6, you had to create folders yourself and point Impala database, tables, or partitions at them, and manually remove folders when no longer needed.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have write permission for the parent HDFS directory under which the database is located.

Examples:

```
create database first_db;
use first_db;
create table t1 (x int);

create database second_db;
use second_db;
-- Each database has its own namespace for tables.
-- You can reuse the same table names in each database.
create table t1 (s string);

create database temp;

-- You can either USE a database after creating it,
```

```
-- or qualify all references to the table name with the name of the datab
ase.
-- Here, tables T2 and T3 are both created in the TEMP database.

create table temp.t2 (x int, y int);
use database temp;
create table t3 (s string);

-- You cannot drop a database while it is selected by the USE statement.
drop database temp;
ERROR: AnalysisException: Cannot drop current default database: temp

-- The always-available database 'default' is a convenient one to USE
-- before dropping a database you created.
use default;
-- Before dropping a database, first drop all the tables inside it,
-- or in Impala 2.3 and higher use the CASCADE clause.
drop database temp;
ERROR: ImpalaRuntimeException: Error making 'dropDatabase' RPC to Hive Meta
store:
CAUSED BY: InvalidOperationException: Database temp is not empty
show tables in temp;
+------+
| name |
+------+
| t3   |
+------+

-- Impala 2.3 and higher:
drop database temp cascade;

-- Earlier releases:
drop table temp.t3;
drop database temp;
```

Related Information

Impala with Amazon S3

CREATE FUNCTION statement

Creates a user-defined function (UDF), which you can use to implement custom logic during SELECT or INSERT operations.

Syntax:

The syntax is different depending on whether you create a scalar UDF, which is called once for each row and implemented by a single function, or a user-defined aggregate function (UDA), which is implemented by multiple functions that compute intermediate results across sets of rows.

In Impala 2.5 and higher, the syntax is also different for creating or dropping scalar Java-based UDFs. The statements for Java UDFs use a new syntax, without any argument types or return type specified. Java-based UDFs created using the new syntax persist across restarts of the Impala catalog server, and can be shared transparently between Impala and Hive.

To create a persistent scalar C++ UDF with CREATE FUNCTION:

```
CREATE FUNCTION [IF NOT EXISTS]
 [DB_NAME.]FUNCTION_NAME([ARG_TYPE[, ARG_TYPE...]])
  RETURNS RETURN_TYPE
  LOCATION 'HDFS_PATH_TO_DOT_SO'
  SYMBOL='SYMBOL_NAME'
```

To create a persistent Java UDF with CREATE FUNCTION:

```
CREATE FUNCTION [IF NOT EXISTS] [DB_NAME.]FUNCTION_NAME
  LOCATION 'HDFS_PATH_TO_JAR'
  SYMBOL='CLASS_NAME'
```

To create a persistent UDA, which must be written in C++, issue a CREATE AGGREGATE FUNCTION statement:

```
CREATE [AGGREGATE] FUNCTION [IF NOT EXISTS]
 [DB_NAME.]FUNCTION_NAME([ARG_TYPE[, ARG_TYPE...])
  RETURNS RETURN_TYPE
  [INTERMEDIATE TYPE_SPEC]
  LOCATION 'HDFS_PATH'
  [INIT_FN='FUNCTION]
  UPDATE_FN='FUNCTION
  MERGE_FN='FUNCTION
  [PREPARE_FN='FUNCTION]
  [CLOSEFN='FUNCTION]
  [SERIALIZE_FN='FUNCTION]
  [FINALIZE_FN='FUNCTION]
```

Statement type: DDL

Varargs notation:

Note:

Variable-length argument lists are supported for C++ UDFs, but currently not for Java UDFs.

If the underlying implementation of your function accepts a variable number of arguments:

- The variable arguments must go last in the argument list.
- The variable arguments must all be of the same type.
- You must include at least one instance of the variable arguments in every function call invoked from SQL.
- You designate the variable portion of the argument list in the CREATE FUNCTION statement by including ... immediately after the type name of the first variable argument. For example, to create a function that accepts an INT argument, followed by a BOOLEAN, followed by one or more STRING arguments, your CREATE FUNCTION statement would look like:

  ```
  CREATE FUNCTION FUNC_NAME (INT, BOOLEAN, STRING ...)
    RETURNS TYPE LOCATION 'PATH' SYMBOL='ENTRY_POINT';
  ```

See User-defined functions for how to code a C++ UDF to accept variable-length argument lists.

Scalar and aggregate functions:

The simplest kind of user-defined function returns a single scalar value each time it is called, typically once for each row in the result set. This general kind of function is what is usually meant by UDF. User-defined aggregate functions (UDAs) are a specialized kind of UDF that produce a single value based on the contents of multiple rows. You usually use UDAs in combination with a GROUP BY clause to condense a large result set into a smaller one, or even a single row summarizing column values across an entire table.

You create UDAs by using the CREATE AGGREGATE FUNCTION syntax. The clauses INIT_FN, UPDATE_FN, MERGE_FN, SERIALIZE_FN, FINALIZE_FN, and INTERMEDIATE only apply when you create a UDA rather than a scalar UDF.

The *_FN clauses specify functions to call at different phases of function processing.

- Initialize: The function you specify with the INIT_FN clause does any initial setup, such as initializing member variables in internal data structures. This function is often a stub for simple UDAs. You can omit this clause and a default (no-op) function will be used.
- Update: The function you specify with the UPDATE_FN clause is called once for each row in the original result set, that is, before any GROUP BY clause is applied. A separate instance of the function is called for each

different value returned by the GROUP BY clause. The final argument passed to this function is a pointer, to which you write an updated value based on its original value and the value of the first argument.

- Merge: The function you specify with the MERGE_FN clause is called an arbitrary number of times, to combine intermediate values produced by different nodes or different threads as Impala reads and processes data files in parallel. The final argument passed to this function is a pointer, to which you write an updated value based on its original value and the value of the first argument.
- Serialize: The function you specify with the SERIALIZE_FN clause frees memory allocated to intermediate results. It is required if any memory was allocated by the Allocate function in the Init, Update, or Merge functions, or if the intermediate type contains any pointers. See the *UDA code samples* for details.
- Finalize: The function you specify with the FINALIZE_FN clause does any required teardown for resources acquired by your UDF, such as freeing memory, closing file handles if you explicitly opened any files, and so on. This function is often a stub for simple UDAs. You can omit this clause and a default (no-op) function will be used. It is required in UDAs where the final return type is different than the intermediate type. or if any memory was allocated by the Allocate function in the Init, Update, or Merge functions. See the *UDA code samples* for details.

If you use a consistent naming convention for each of the underlying functions, Impala can automatically determine the names based on the first such clause, so the others are optional.

Complex type considerations:

Currently, Impala UDFs cannot accept arguments or return values of the Impala complex types (STRUCT, ARRAY, or MAP).

Usage notes:

- When authorization is enabled, the CREATE FUNCTION statement requires:
  - The CREATE privilege on the database.
  - The ALL privilege on two URIs where the URIs are:
    - The JAR file on the file system. For example:

      ```
      GRANT ALL ON URI 'file:///PATH_TO_MY.JAR' TO ROLE MY_ROLE;
      ```

    - The JAR on HDFS. For example:

      ```
      GRANT ALL ON URI 'hdfs:///PATH/TO/JAR' TO ROLE MY_ROLE
      ```

- You can write Impala UDFs in either C++ or Java. C++ UDFs are new to Impala, and are the recommended format for high performance utilizing native code. Java-based UDFs are compatible between Impala and Hive, and are most suited to reusing existing Hive UDFs. (Impala can run Java-based Hive UDFs but not Hive UDAs.)
- Impala 2.5 introduces UDF improvements to persistence for both C++ and Java UDFs, and better compatibility between Impala and Hive for Java UDFs.
- The body of the UDF is represented by a .so or .jar file, which you store in HDFS and the CREATE FUNCTION statement distributes to each Impala node.
- Impala calls the underlying code during SQL statement evaluation, as many times as needed to process all the rows from the result set. All UDFs are assumed to be deterministic, that is, to always return the same result when passed the same argument values. Impala might or might not skip some invocations of a UDF if the result value is already known from a previous call. Therefore, do not rely on the UDF being called a specific number of times, and do not return different result values based on some external factor such as the current time, a random number function, or an external data source that could be updated while an Impala query is in progress.
- The names of the function arguments in the UDF are not significant, only their number, positions, and data types.
- You can overload the same function name by creating multiple versions of the function, each with a different argument signature. For security reasons, you cannot make a UDF with the same name as any built-in function.
- In the UDF code, you represent the function return result as a struct. This struct contains 2 fields. The first field is a boolean representing whether the value is NULL or not. (When this field is true, the return value is interpreted

as NULL.) The second field is the same type as the specified function return type, and holds the return value when the function returns something other than NULL.

- In the UDF code, you represent the function arguments as an initial pointer to a UDF context structure, followed by references to zero or more structs, corresponding to each of the arguments. Each struct has the same 2 fields as with the return value, a boolean field representing whether the argument is NULL, and a field of the appropriate type holding any non-NULL argument value.
- For sample code and build instructions for UDFs, see the *UDA code samples*.
- Because the file representing the body of the UDF is stored in HDFS, it is automatically available to all the Impala nodes. You do not need to manually copy any UDF-related files between servers.
- Because Impala currently does not have any ALTER FUNCTION statement, if you need to rename a function, move it to a different database, or change its signature or other properties, issue a DROP FUNCTION statement for the original function followed by a CREATE FUNCTION with the desired properties.
- Because each UDF is associated with a particular database, either issue a USE statement before doing any CREATE FUNCTION statements, or specify the name of the function as *DB_NAME.FUNCTION_NAME*.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Compatibility:

Impala can run UDFs that were created through Hive, as long as they refer to Impala-compatible data types (not composite or nested column types). Hive can run Java-based UDFs that were created through Impala, but not Impala UDFs written in C++.

The Hive current_user() function cannot be called from a Java UDF through Impala.

Persistence:

In Impala 2.5 and higher, Impala UDFs and UDAs written in C++ are persisted in the metastore database. Java UDFs are also persisted, if they were created with the new CREATE FUNCTION syntax for Java UDFs, where the Java function argument and return types are omitted. Java-based UDFs created with the old CREATE FUNCTION syntax do not persist across restarts because they are held in the memory of the `catalogd` daemon. Until you re-create such Java UDFs using the new CREATE FUNCTION syntax, you must reload those Java-based UDFs by running the original CREATE FUNCTION statements again each time you restart the `catalogd` daemon. Prior to Impala 2.5 the requirement to reload functions after a restart applied to both C++ and Java functions.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Examples:

For additional examples of all kinds of user-defined functions, see Creating an Impala user-defined function on page 444.

The following example shows how to take a Java jar file and make all the functions inside one of its classes into UDFs under a single (overloaded) function name in Impala. Each CREATE FUNCTION or DROP FUNCTION statement applies to all the overloaded Java functions with the same name. This example uses the signatureless syntax for CREATE FUNCTION and DROP FUNCTION, which is available in Impala 2.5 and higher.

At the start, the jar file is in the local filesystem. Then it is copied into HDFS, so that it is available for Impala to reference through the CREATE FUNCTION statement and queries that refer to the Impala function name.

```
$ jar -tvf udf-examples-cdh570.jar
     0 Mon Feb 22 04:06:50 PST 2016 META-INF/
   122 Mon Feb 22 04:06:48 PST 2016 META-INF/MANIFEST.MF
     0 Mon Feb 22 04:06:46 PST 2016 com/
     0 Mon Feb 22 04:06:46 PST 2016 com/cloudera/
     0 Mon Feb 22 04:06:46 PST 2016 com/cloudera/impala/
```

```
  2460 Mon Feb 22 04:06:46 PST 2016 com/cloudera/impala/IncompatibleUdfTe
st.class
   541 Mon Feb 22 04:06:46 PST 2016 com/cloudera/impala/TestUdfException.cl
ass
  3438 Mon Feb 22 04:06:46 PST 2016 com/cloudera/impala/JavaUdfTest.class
  5872 Mon Feb 22 04:06:46 PST 2016 com/cloudera/impala/TestUdf.class
...
$ hdfs dfs -put udf-examples-cdh570.jar /user/impala/udfs
$ hdfs dfs -ls /user/impala/udfs
Found 2 items
-rw-r--r--   3 jrussell supergroup        853 2015-10-09 14:05 /user/impala/
udfs/hello_world.jar
-rw-r--r--   3 jrussell supergroup       7366 2016-06-08 14:25 /user/impala/
udfs/udf-examples-cdh570.jar
```

In `impala-shell`, the CREATE FUNCTION refers to the HDFS path of the jar file and the fully qualified class name inside the jar. Each of the functions inside the class becomes an Impala function, each one overloaded under the specified Impala function name.

```
[localhost:21000] > create function testudf location '/user/impala/udfs/udf-
examples-cdh570.jar' symbol='com.cloudera.impala.TestUdf';
[localhost:21000] > show functions;
+-------------+-------------------------------------+-------------+-------
--------+
| return type | signature                           | binary type | is p
ersistent |
+-------------+-------------------------------------+-------------+-------
--------+
| BIGINT      | testudf(BIGINT)                     | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN)                    | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN)           | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN, BOOLEAN)  | JAVA        | true
        |
| DOUBLE      | testudf(DOUBLE)                     | JAVA        | true
        |
| DOUBLE      | testudf(DOUBLE, DOUBLE)             | JAVA        | true
        |
| DOUBLE      | testudf(DOUBLE, DOUBLE, DOUBLE)     | JAVA        | true
        |
| FLOAT       | testudf(FLOAT)                      | JAVA        | true
        |
| FLOAT       | testudf(FLOAT, FLOAT)               | JAVA        | true
        |
| FLOAT       | testudf(FLOAT, FLOAT, FLOAT)        | JAVA        | true
        |
| INT         | testudf(INT)                        | JAVA        | true
        |
| DOUBLE      | testudf(INT, DOUBLE)                | JAVA        | true
        |
| INT         | testudf(INT, INT)                   | JAVA        | true
        |
| INT         | testudf(INT, INT, INT)              | JAVA        | true
        |
| SMALLINT    | testudf(SMALLINT)                   | JAVA        | true
        |
| SMALLINT    | testudf(SMALLINT, SMALLINT)         | JAVA        | true
        |
| SMALLINT    | testudf(SMALLINT, SMALLINT, SMALLINT) | JAVA        | true
        |
```

```
| STRING      | testudf(STRING)                       | JAVA        | true
        |
| STRING      | testudf(STRING, STRING)               | JAVA        | true
        |
| STRING      | testudf(STRING, STRING, STRING)       | JAVA        | true
        |
| TINYINT     | testudf(TINYINT)                      | JAVA        | true
        |
+-------------+---------------------------------------+-------------+-------
--------+
```

These are all simple functions that return their single arguments, or sum, concatenate, and so on their multiple arguments. Impala determines which overloaded function to use based on the number and types of the arguments.

```
insert into bigint_x values (1), (2), (4), (3);
select testudf(x) from bigint_x;
+-----------------+
| udfs.testudf(x) |
+-----------------+
| 1               |
| 2               |
| 4               |
| 3               |
+-----------------+

insert into int_x values (1), (2), (4), (3);
select testudf(x, x+1, x*x) from int_x;
+-------------------------------+
| udfs.testudf(x, x + 1, x * x) |
+-------------------------------+
| 4                             |
| 9                             |
| 25                            |
| 16                            |
+-------------------------------+
select testudf(x) from string_x;
+-----------------+
| udfs.testudf(x) |
+-----------------+
| one             |
| two             |
| four            |
| three           |
+-----------------+
select testudf(x,x) from string_x;
+--------------------+
| udfs.testudf(x, x) |
+--------------------+
| oneone             |
| twotwo             |
| fourfour           |
| threethree         |
+--------------------+
```

The previous example used the same Impala function name as the name of the class. This example shows how the Impala function name is independent of the underlying Java class or function names. A second CREATE FUNCTION statement results in a set of overloaded functions all named my_func, to go along with the overloaded functions all named testudf.

```
create function my_func location '/user/impala/udfs/udf-examples-cdh570.jar'
```

```
  symbol='com.cloudera.impala.TestUdf';

show functions;
+-------------+-------------------------------------+-------------+----
-----------+
| return type | signature                           | binary type | is pe
rsistent |
+-------------+-------------------------------------+-------------+----
-----------+
| BIGINT      | my_func(BIGINT)                     | JAVA        | true
        |
| BOOLEAN     | my_func(BOOLEAN)                    | JAVA        | true
        |
| BOOLEAN     | my_func(BOOLEAN, BOOLEAN)           | JAVA        | true
        |
...
| BIGINT      | testudf(BIGINT)                     | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN)                    | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN)           | JAVA        | true
        |
...
```

The corresponding DROP FUNCTION statement with no signature drops all the overloaded functions with that name.

```
drop function my_func;
show functions;
+-------------+-------------------------------------+-------------+-------
--------+
| return type | signature                           | binary type | is p
ersistent |
+-------------+-------------------------------------+-------------+-------
--------+
| BIGINT      | testudf(BIGINT)                     | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN)                    | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN)           | JAVA        | true
        |
...
```

The signatureless CREATE FUNCTION syntax for Java UDFs ensures that the functions shown in this example remain available after the Impala service (specifically, the Catalog Server) are restarted.

Related Information

UDA sample code

CREATE ROLE statement

The CREATE ROLE statement creates a role to which privileges can be granted. Privileges can be granted to roles, which can then be assigned to a group and the users belonging to the group will be assigned the role. A user that has been assigned a role will only be able to exercise the privileges of that role. Only users with administrative privileges can create/drop roles. By default, the Impala users do not have administrative privileges in Ranger. For Impala users to have administrative privileges in Ranger you must use Ranger Web UI and configure.

Syntax:

```
CREATE ROLE ROLE_NAME
```

Required privileges:

Only administrative users can use this statement.

Compatibility:

Impala makes use of any roles and privileges specified by the GRANT and REVOKE statements in Hive, and Hive makes use of any roles and privileges specified by the GRANT and REVOKE statements in Impala. The Impala GRANT and REVOKE statements for privileges do not require the ROLE keyword to be repeated before each role name, unlike the equivalent Hive statements.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

CREATE TABLE statement

The CREATE TABLE statement creates a new table with the specified properties.

While creating a table, you can optionally specify aspects such as:

- Whether the table is internal or external.
- The columns and associated data types.
- The columns used for physically partitioning the data.
- The file format for data files.
- The HDFS directory where the data files are located.

Syntax:

The general syntax for creating a table and specifying its columns is as follows:

Explicit column definitions:

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
  (COL_NAME DATA_TYPE
    [COMMENT 'COL_COMMENT']
    [, ...]
  )
  [PARTITIONED BY (COL_NAME DATA_TYPE [COMMENT 'COL_COMMENT'], ...)]
  [SORT BY ([COLUMN [, COLUMN ...]])]
  [COMMENT 'TABLE_COMMENT']
  [ROW FORMAT ROW_FORMAT]
  [WITH SERDEPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
  [STORED AS FILE_FORMAT]
  [LOCATION 'HDFS_PATH']
  [CACHED IN 'POOL_NAME' [WITH REPLICATION = INTEGER] | UNCACHED]
  [TBLPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
```

CREATE TABLE AS SELECT:

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] DB_NAME.]TABLE_NAME
    [PARTITIONED BY (COL_NAME[, ...])]
    [SORT BY ([COLUMN [, COLUMN ...]])]
    [COMMENT 'TABLE_COMMENT']
+   [ROW FORMAT ROW_FORMAT]
    [WITH SERDEPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
+   [STORED AS CTAS_FILE_FORMAT]
    [LOCATION 'HDFS_PATH']
+     [CACHED IN 'POOL_NAME' [WITH REPLICATION = INTEGER] | UNCACHED]
    [TBLPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
  AS
```

```
    SELECT_STATEMENT
```

Note: If creating a partitioned table, the partition key columns must be listed last in the SELECT columns list, in the same order as in the PARTITIONED BY clause. Otherwise, you will receive an error about a column name mismatch.

```
primitive_type:
    TINYINT
  | SMALLINT
  | INT
  | BIGINT
  | BOOLEAN
  | FLOAT
  | DOUBLE
  | DECIMAL
  | STRING
  | CHAR
  | VARCHAR
  | TIMESTAMP

complex_type:
    struct_type
  | array_type
  | map_type

struct_type: STRUCT < NAME : PRIMITIVE_OR_COMPLEX_TYPE [COMMENT
 'COMMENT_STRING'], ... >

array_type: ARRAY < PRIMITIVE_OR_COMPLEX_TYPE >

map_type: MAP < PRIMITIVE_TYPE, PRIMITIVE_OR_COMPLEX_TYPE >

row_format:
  DELIMITED [FIELDS TERMINATED BY 'CHAR' [ESCAPED BY 'CHAR']]
  [LINES TERMINATED BY 'CHAR']

file_format:
    PARQUET
  | TEXTFILE
  | AVRO
  | SEQUENCEFILE
  | RCFILE

ctas_file_format:
    PARQUET
  | TEXTFILE
```

Column definitions inferred from data file:

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
  LIKE PARQUET 'HDFS_PATH_OF_PARQUET_FILE'
  [PARTITIONED BY (COL_NAME DATA_TYPE [COMMENT 'COL_COMMENT'], ...)]
  [SORT BY ([COLUMN [, COLUMN ...]])]
  [COMMENT 'TABLE_COMMENT']
  [ROW FORMAT ROW_FORMAT]
  [WITH SERDEPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
  [STORED AS FILE_FORMAT]
  [LOCATION 'HDFS_PATH']
  [CACHED IN 'POOL_NAME' [WITH REPLICATION = INTEGER] | UNCACHED]
  [TBLPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
data_type:
    PRIMITIVE_TYPE
```

```
  | array_type
  | map_type
  | struct_type
```

Kudu tables:

```
CREATE TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
  (COL_NAME DATA_TYPE
    [KUDU_COLUMN_ATTRIBUTE ...]
    [COMMENT 'COL_COMMENT']
    [, ...]
    [PRIMARY KEY (COL_NAME[, ...])]
  )
  [PARTITION BY KUDU_PARTITION_CLAUSE]
  [COMMENT 'TABLE_COMMENT']
  STORED AS KUDU
  [TBLPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
```

Kudu column attributes:

```
  PRIMARY KEY
| [NOT] NULL
| ENCODING CODEC
| COMPRESSION ALGORITHM
| DEFAULT CONSTANT
| BLOCK_SIZE NUMBER
```

kudu_partition_clause:

```
kudu_partition_clause ::= [ HASH_CLAUSE [, ...]] [, RANGE_CLAUSE ]

hash_clause ::=
  HASH [ (PK_COL [, ...]) ]
    PARTITIONS N

range_clause ::=
  RANGE [ (PK_COL [, ...]) ]
  (
    {
      PARTITION CONSTANT_EXPRESSION RANGE_COMPARISON_OPERATOR VAL
UES RANGE_COMPARISON_OPERATOR CONSTANT_EXPRESSION
      | PARTITION VALUE = CONSTANT_EXPRESSION_OR_TUPLE
    }
   [, ...]
  )

range_comparison_operator ::= { < | <= }
```

External Kudu tables:

In Impala 3.4 and earlier, you can create an external Kudu table based on a pre-existing Kudu schema using the table property 'kudu.table_name'='internal_kudu_name'.

```
CREATE EXTERNAL TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
      [COMMENT 'COL_COMMENT']
      STORED AS KUDU
      [TBLPROPERTIES ('KUDU.TABLE.NAME'='INTERNAL_KUDU_NAME',
 'KEY1'='VALUE1',...)]
```

Alternatively, in Impala 3.4 and higher, you can also create an external Kudu table as follows:

```
CREATE EXTERNAL TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
      (COL_NAME DATA_TYPE
      [KUDU_COLUMN_ATTRIBUTE ...]
      [COMMENT 'COL_COMMENT']
      [, ...]
      [PRIMARY KEY (COL_NAME[, ...])]
      )
      [PARTITION BY KUDU_PARTITION_CLAUSE]
      [COMMENT 'TABLE_COMMENT']
      STORED AS KUDU
      [TBLPROPERTIES ('external.table.purge'='true', 'KEY1'='VALUE1',...)]
```

- Use a Hive metastore (HMS) 3 or later.
- Provide column specifications to define the schema when you create the table, similar to creating an internal table.
- Omit the kudu.table_name table property as there is no pre-existing schema.
- Include the required external.table.purge property.

Only the schema metadata is stored in HMS when you create an external table; however, using this create table syntax, drop table on the Kudu external table deletes the data stored outside HMS in Kudu as well as the metadata (schema) inside HMS. Likewise, renaming the table changes the name of the table in HMS and in Kudu. Kudu synchronizes changes to the actual data and metadata; consequently, operations such as dropping a table or altering a table name simulate internal table operations.

CREATE TABLE AS SELECT for Kudu tables:

```
CREATE TABLE [IF NOT EXISTS] DB_NAME.]TABLE_NAME
  [PRIMARY KEY (COL_NAME[, ...])]
  [PARTITION BY KUDU_PARTITION_CLAUSE]
  [COMMENT 'TABLE_COMMENT']
  STORED AS KUDU
  [TBLPROPERTIES ('KEY1'='VALUE1', 'KEY2'='VALUE2', ...)]
AS
  SELECT_STATEMENT
```

Statement type: DDL

Column definitions:

Depending on the form of the CREATE TABLE statement, the column definitions are required or not allowed.

With the CREATE TABLE AS SELECT and CREATE TABLE LIKE syntax, you do not specify the columns at all; the column names and types are derived from the source table, query, or data file.

With the basic CREATE TABLE syntax, you must list one or more columns, its name, type, optionally constraints, and optionally a comment, in addition to any columns used as partitioning keys. There is one exception where the column list is not required: when creating an Avro table with the STORED AS AVRO clause, you can omit the list of columns and specify the same metadata as part of the TBLPROPERTIES clause.

Constraints:

Constraints are advisory and intended for estimating cardinality during query planning in a future release; there is no attempt to enforce constraints. Add primary and foreign key information after column definitions. Do not include a constraint name; the constraint name is generated internally as a UUID. The following constraint states are supported:

- DISABLE
- NOVALIDATE
- RELY

The ENABLE, VALIDATE, and NORELY options are not supported. The foreign key must be defined as the primary key in the referenced table.

Constraint examples:

```
CREATE TABLE pk(col1 INT, col2 STRING, PRIMARY KEY(col1, col2));
```

```
CREATE TABLE fk(id INT, col1 INT, col2 STRING, PRIMARY KEY(id),
        FOREIGN KEY(col1, col2) REFERENCES pk(col1, col2));
```

```
CREATE TABLE pk(id INT, PRIMARY KEY(id) DISABLE, NOVALIDATE, RELY);
```

```
CREATE TABLE fk(id INT, col1 INT, col2 STRING, PRIMARY KEY(id),
        FOREIGN KEY(col1, col2) REFERENCES pk(col1, col2));
```

Complex type considerations:

The Impala complex types (STRUCT, ARRAY, or MAP) are available in Impala 2.3 and higher. Because you can nest these types (for example, to make an array of maps or a struct with an array field), these types are also sometimes referred to as nested types.

Impala can create tables containing complex type columns, with any supported file format. Because currently Impala can only query complex type columns in Parquet tables, creating tables with complex type columns and other file formats such as text is of limited use. For example, you might create a text table including some columns with complex types with Impala, and use Hive as part of your to ingest the nested type data and copy it to an identical Parquet table. Or you might create a partitioned table containing complex type columns using one file format, and use ALTER TABLE to change the file format of individual partitions to Parquet; Impala can then query only the Parquet-format partitions in that table.

Partitioned tables can contain complex type columns. All the partition key columns must be scalar types.

Internal and external tables (EXTERNAL and LOCATION clauses):

By default, Impala creates an “internal” table, where Impala manages the underlying data files for the table, and physically deletes the data files when you drop the table. If you specify the EXTERNAL clause, Impala treats the table as an “external” table, where the data files are typically produced outside Impala and queried from their original locations in HDFS, and Impala leaves the data files in place when you drop the table.

Typically, for an external table you include a LOCATION clause to specify the path to the HDFS directory where Impala reads and writes files for the table. For example, if your data pipeline produces Parquet files in the HDFS directory /user/etl/destination, you might create an external table as follows:

```
CREATE EXTERNAL TABLE external_parquet (c1 INT, c2 STRING, c3 TIMESTAMP)
  STORED AS PARQUET LOCATION '/user/etl/destination';
```

Although the EXTERNAL and LOCATION clauses are often specified together, LOCATION is optional for external tables, and you can also specify LOCATION for internal tables. The difference is all about whether Impala “takes control” of the underlying data files and moves them when you rename the table, or deletes them when you drop the table. For more about internal and external tables and how they interact with the LOCATION attribute, see *Overview of Impala tables*.

Partitioned tables (PARTITIONED BY clause):

The PARTITIONED BY clause divides the data files based on the values from one or more specified columns. Impala queries can use the partition metadata to minimize the amount of data that is read from disk or transmitted across the network, particularly during join queries.

Note:

All Kudu tables require partitioning, which involves different syntax than non-Kudu tables. See the PARTITION BY clause, rather than PARTITIONED BY, for Kudu tables.

In Impala 2.10 and higher, the PARTITION BY clause is optional for Kudu tables. If the clause is omitted, Impala automatically constructs a single partition that is not connected to any column. Because such a table cannot take advantage of Kudu features for parallelized queries and query optimizations, omitting the PARTITION BY clause is only appropriate for small lookup tables.

Prior to Impala 2.5, you could use a partitioned table as the source and copy data from it, but could not specify any partitioning clauses for the new table. In Impala 2.5 and higher, you can now use the PARTITIONED BY clause with a CREATE TABLE AS SELECT statement. See the examples under the following discussion of the CREATE TABLE AS SELECT syntax variation.

Sorted tables (SORT BY clause):

The optional SORT BY clause lets you specify zero or more columns that are sorted in the data files created by each Impala INSERT or CREATE TABLE AS SELECT operation. Creating data files that are sorted is most useful for Parquet tables, where the metadata stored inside each file includes the minimum and maximum values for each column in the file. (The statistics apply to each row group within the file; for simplicity, Impala writes a single row group in each file.) Grouping data values together in relatively narrow ranges within each data file makes it possible for Impala to quickly skip over data files that do not contain value ranges indicated in the WHERE clause of a query, and can improve the effectiveness of Parquet encoding and compression.

This clause is not applicable for Kudu tables or HBase tables. Although it works for other HDFS file formats besides Parquet, the more efficient layout is most evident with Parquet tables, because each Parquet data file includes statistics about the data values in that file.

The SORT BY columns cannot include any partition key columns for a partitioned table, because those column values are not represented in the underlying data files.

Because data files can arrive in Impala tables by mechanisms that do not respect the SORT BY clause, such as LOAD DATA or ETL tools that create HDFS files, Impala does not guarantee or rely on the data being sorted. The sorting aspect is only used to create a more efficient layout for Parquet files generated by Impala, which helps to optimize the processing of those Parquet files during Impala queries. During an INSERT or CREATE TABLE AS SELECT operation, the sorting occurs when the SORT BY clause applies to the destination table for the data, regardless of whether the source table has a SORT BY clause.

For example, when creating a table intended to contain census data, you might define sort columns such as last name and state. If a data file in this table contains a narrow range of last names, for example from Smith to Smythe, Impala can quickly detect that this data file contains no matches for a WHERE clause such as WHERE last_name = 'Jones' and avoid reading the entire file.

```
CREATE TABLE census_data (last_name STRING, first_name STRING, state STRING,
 address STRING)
  SORT BY (last_name, state)
  STORED AS PARQUET;
```

Likewise, if an existing table contains data without any sort order, you can reorganize the data in a more efficient way by using INSERT or CREATE TABLE AS SELECT to copy that data into a new table with a SORT BY clause:

```
CREATE TABLE sorted_census_data
  SORT BY (last_name, state)
  STORED AS PARQUET
  AS SELECT last_name, first_name, state, address
    FROM unsorted_census_data;
```

The metadata for the SORT BY clause is stored in the TBLPROPERTIES fields for the table. Other SQL engines that can interoperate with Impala tables, such as Hive and Spark SQL, do not recognize this property when inserting into a table that has a SORT BY clause.

Transactional tables:

In the version 3.3 and higher, when integrated with Hive 3, Impala can create, read, and insert into transactional tables.

To create a table that supports transactions, use the TBLPROPERTIES clause and set the 'transactional' and 'transac tional_properties' as below. Currently, Impala only supports insert-only transactional tables.

```
TBLPROPERTIES('transactional'='true', 'transactional_properties'='insert_onl
y')
```

When integrated with Hive3 and the DEFAULT_TRANSACTIONAL_TYPE query option is set to INSERT_ONLY, tables are created as insert-only transactional table by default.

Transactional tables are not supported for Kudu and HBase.

Kudu considerations:

Because Kudu tables do not support clauses related to HDFS and S3 data files and partitioning mechanisms, the syntax associated with the STORED AS KUDU clause is shown separately in the above syntax descriptions. Kudu tables have their own syntax for CREATE TABLE, CREATE EXTERNAL TABLE, and CREATE TABLE AS SELECT. Prior to Impala 2.10, all internal Kudu tables require a PARTITION BY clause, different than the PART ITIONED BY clause for HDFS-backed tables.

Here are some examples of creating empty Kudu tables:

```
-- Single partition. Only for  Impala 2.10 and higher.
-- Only suitable for small lookup tables.
CREATE TABLE kudu_no_partition_by_clause
  (
    id bigint PRIMARY KEY, s STRING, b BOOLEAN
  )
  STORED AS KUDU;

-- Single-column primary key.
CREATE TABLE kudu_t1 (id BIGINT PRIMARY key, s STRING, b BOOLEAN)
  PARTITION BY HASH (id) PARTITIONS 20 STORED AS KUDU;

-- Multi-column primary key.
CREATE TABLE kudu_t2 (id BIGINT, s STRING, b BOOLEAN, PRIMARY KEY (id,s))
  PARTITION BY HASH (s) PARTITIONS 30 STORED AS KUDU;

-- Meaningful primary key column is good for range partitioning.
CREATE TABLE kudu_t3 (id BIGINT, year INT, s STRING,
    b BOOLEAN, PRIMARY KEY (id,year))
  PARTITION BY HASH (id) PARTITIONS 20,
  RANGE (year) (PARTITION 1980 <= VALUES < 1990,
    PARTITION 1990 <= VALUES < 2000,
    PARTITION VALUE = 2001,
    PARTITION 2001 < VALUES)
  STORED AS KUDU;
```

Here is an example of creating an external Kudu table:

```
-- Inherits column definitions from original table.
-- For tables created through Impala, the kudu.table_name property
-- comes from DESCRIBE FORMATTED output from the original table.
CREATE EXTERNAL TABLE external_t1 STORED AS KUDU
  TBLPROPERTIES ('kudu.table_name'='kudu_tbl_created_via_api');
```

In Impala 3.4 and higher, by default HMS implicitly translates internal Kudu tables to external Kudu tables with the 'external.table.purge' property set to true. You can explicitly create such external Kudu tables similar to the way you create internal Kudu tables. You must set the table property 'external.table.purge' to true. Here is an example of creating an external Kudu table:

```
CREATE EXTERNAL TABLE myextkudutbl (
      id int PRIMARY KEY,
      name string)
      PARTITION BY HASH PARTITIONS 8
      STORED AS KUDU
      TBLPROPERTIES ('external.table.purge'='true');
```

Operations on the resulting external table in Impala, HMS, and Kudu table metadata is synchronized. HMS-Kudu integration does not need to be enabled for external table synchronization. Such synchronized tables behave similar to internal tables. For example, dropping a table removes the underlying Kudu table data as well as the table metadata in HMS.

If you want to drop only the HMS metadata and not drop the Kudu table, you can set external.table.purge to false, as shown in the following example:

```
      ALTER TABLE myextkudutbl set tblproperties('external.table.purge'='fa
lse');
```

Here is an example of CREATE TABLE AS SELECT syntax for a Kudu table:

```
-- The CTAS statement defines the primary key and partitioning scheme.
-- The rest of the column definitions are derived from the select list.
CREATE TABLE ctas_t1
  PRIMARY KEY (id) PARTITION BY HASH (id) PARTITIONS 10
  STORED AS KUDU
  AS SELECT id, s FROM kudu_t1;
```

The following CREATE TABLE clauses are not supported for Kudu tables:

- PARTITIONED BY (Kudu tables use the clause PARTITION BY instead)
- LOCATION
- ROWFORMAT
- CACHED IN | UNCACHED
- WITH SERDEPROPERTIES

Partitioning for Kudu tables (PARTITION BY clause)

For Kudu tables, you specify logical partitioning across one or more columns using the PARTITION BY clause. In contrast to partitioning for HDFS-based tables, multiple values for a partition key column can be located in the same partition. The optional HASH clause lets you divide one or a set of partition key columns into a specified number of buckets. You can use more than one HASH clause, specifying a distinct set of partition key columns for each. The optional RANGE clause further subdivides the partitions, based on a set of comparison operations for the partition key columns.

Here are some examples of the PARTITION BY HASH syntax:

```
-- Apply hash function to 1 primary key column.
create table hash_t1 (x bigint, y bigint, s string, primary key (x,y))
  partition by hash (x) partitions 10
  stored as kudu;
```

```
-- Apply hash function to a different primary key column.
create table hash_t2 (x bigint, y bigint, s string, primary key (x,y))
  partition by hash (y) partitions 10
  stored as kudu;

-- Apply hash function to both primary key columns.
-- In this case, the total number of partitions is 10.
create table hash_t3 (x bigint, y bigint, s string, primary key (x,y))
  partition by hash (x,y) partitions 10
  stored as kudu;

-- When the column list is omitted, apply hash function to all primary key c
olumns.
create table hash_t4 (x bigint, y bigint, s string, primary key (x,y))
  partition by hash partitions 10
  stored as kudu;

-- Hash the X values independently from the Y values.
-- In this case, the total number of partitions is 10 x 20.
create table hash_t5 (x bigint, y bigint, s string, primary key (x,y))
  partition by hash (x) partitions 10, hash (y) partitions 20
  stored as kudu;
```

Here are some examples of the PARTITION BY RANGE syntax:

```
-- Create partitions that cover every possible value of X.
-- Ranges that span multiple values use the keyword VALUES between
-- a pair of < and <= comparisons.
create table range_t1 (x bigint, s string, s2 string, primary key (x, s))
  partition by range (x)
    (
      partition 0 <= values <= 49, partition 50 <= values <= 100,
      partition values < 0, partition 100 < values
    )
  stored as kudu;
-- Create partitions that cover some possible values of X.
-- Values outside the covered range(s) are rejected.
-- New range partitions can be added through ALTER TABLE.
create table range_t2 (x bigint, s string, s2 string, primary key (x, s))
  partition by range (x)
    (
      partition 0 <= values <= 49, partition 50 <= values <= 100
    )
  stored as kudu;

-- A range can also specify a single specific value, using the keyword VALUE
-- with an = comparison.
create table range_t3 (x bigint, s string, s2 string, primary key (x, s))
  partition by range (s)
    (
      partition value = 'Yes', partition value = 'No', partition value = 'M
aybe'
    )
  stored as kudu;

-- Using multiple columns in the RANGE clause and tuples inside the parti
tion spec
-- only works for partitions specified with the VALUE= syntax.
create table range_t4 (x bigint, s string, s2 string, primary key (x, s))
  partition by range (x,s)
    (
```

```
      partition value = (0,'zero'), partition value = (1,'one'), partition v
alue = (2,'two')
    )
  stored as kudu;
```

Here are some examples combining both HASH and RANGE syntax for the PARTITION BY clause:

```
-- Values from each range partition are hashed into 10 associated buckets.
-- Total number of partitions in this case is 10 x 2.
create table combined_t1 (x bigint, s string, s2 string, primary key (x, s))
  partition by hash (x) partitions 10, range (x)
    (
      partition 0 <= values <= 49, partition 50 <= values <= 100
    )
  stored as kudu;

-- The hash partitioning and range partitioning can apply to different colum
ns.
-- But all the columns used in either partitioning scheme must be from the p
rimary key.
create table combined_t2 (x bigint, s string, s2 string, primary key (x, s))
  partition by hash (s) partitions 10, range (x)
    (
      partition 0 <= values <= 49, partition 50 <= values <= 100
    )
  stored as kudu;
```

Specifying file format (STORED AS and ROW FORMAT clauses):

The STORED AS clause identifies the format of the underlying data files. Currently, Impala can query more types of file formats than it can create or insert into. Use Hive to perform any create or data load operations that are not currently available in Impala. For example, Impala can create an Avro, SequenceFile, or RCFile table but cannot insert data into it. There are also Impala-specific procedures for using compression with each kind of file format. For details about working with data files of various formats, see *How Impala works with Hadoop fle formats*.

Note: In Impala 1.4.0 and higher, Impala can create Avro tables, which formerly required doing the CREATE TABLE statement in Hive. See *Using the Avro file format with Impala Tables* for details and examples.

By default (when no STORED AS clause is specified), data files in Impala tables are created as text files with Ctrl-A (hex 01) characters as the delimiter. Specify the ROW FORMAT DELIMITED clause to produce or ingest data files that use a different delimiter character such as tab or |, or a different line end character such as carriage return or newline. When specifying delimiter and line end characters with the FIELDS TERMINATED BY and LINES TERMINATED BY clauses, use '\t' for tab, '\n' for newline or linefeed, '\r' for carriage return, and \0 for ASCII nul (hex 00).

The ESCAPED BY clause applies both to text files that you create through an INSERT statement to an Impala TEXT FILE table, and to existing data files that you put into an Impala table directory. (You can ingest existing data files either by creating the table with CREATE EXTERNAL TABLE ... LOCATION, the LOAD DATA statement, or through an HDFS operation such as hdfs dfs -put *FILE HDFS_PATH*.) Choose an escape character that is not used anywhere else in the file, and put it in front of each instance of the delimiter character that occurs within a field value. Surrounding field values with quotation marks does not help Impala to parse fields with embedded delimiter characters; the quotation marks are considered to be part of the column value. If you want to use \ as the escape character, specify the clause in `impala-shell` as ESCAPED BY '\\'.

Note: The CREATE TABLE clauses FIELDS TERMINATED BY, ESCAPED BY, and LINES TERMINATED BY have special rules for the string literal used for their argument, because they all require a single character. You can use a regular character surrounded by single or double quotation marks, an octal sequence such as '\054' (representing a comma), or an integer in the range '-127'..'128' (with quotation marks but no backslash), which is interpreted as a single-byte ASCII character. Negative values are subtracted from 256; for example, FIELDS TERMINATED BY '-2' sets the field delimiter to ASCII code 254, the “Icelandic Thorn” character used as a delimiter by some data formats.

Cloning tables (LIKE clause):

To create an empty table with the same columns, comments, and other attributes as another table, use the following variation. The CREATE TABLE ... LIKE form allows a restricted set of clauses, currently only the LOCATION, COMMENT, and STORED AS clauses.

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [DB_NAME.]TABLE_NAME
  LIKE { [DB_NAME.]TABLE_NAME | PARQUET 'HDFS_PATH_OF_PARQUET_FILE' }
  [COMMENT 'TABLE_COMMENT']
  [STORED AS FILE_FORMAT]
  [LOCATION 'HDFS_PATH']
```

Note:

To clone the structure of a table and transfer data into it in a single operation, use the CREATE TABLE AS SELECT syntax described in the next subsection.

When you clone the structure of an existing table using the CREATE TABLE ... LIKE syntax, the new table keeps the same file format as the original one, so you only need to specify the STORED AS clause if you want to use a different file format, or when specifying a view as the original table. (Creating a table “like” a view produces a text table by default.)

Although normally Impala cannot create an HBase table directly, Impala can clone the structure of an existing HBase table with the CREATE TABLE ... LIKE syntax, preserving the file format and metadata from the original table.

There are some exceptions to the ability to use CREATE TABLE ... LIKE with an Avro table. For example, you cannot use this technique for an Avro table that is specified with an Avro schema but no columns. When in doubt, check if a CREATE TABLE ... LIKE operation works in Hive; if not, it typically will not work in Impala either.

If the original table is partitioned, the new table inherits the same partition key columns. Because the new table is initially empty, it does not inherit the actual partitions that exist in the original one. To create partitions in the new table, insert data or issue ALTER TABLE ... ADD PARTITION statements.

Prior to Impala 1.4.0, it was not possible to use the CREATE TABLE LIKE *VIEW_NAME* syntax. In Impala 1.4.0 and higher, you can create a table with the same column definitions as a view using the CREATE TABLE LIKE technique. Although CREATE TABLE LIKE normally inherits the file format of the original table, a view has no underlying file format, so CREATE TABLE LIKE *VIEW_NAME* produces a text table by default. To specify a different file format, include a STORED AS *FILE_FORMAT* clause at the end of the CREATE TABLE LIKE statement.

Because CREATE TABLE ... LIKE only manipulates table metadata, not the physical data of the table, issue INSERT INTO TABLE statements afterward to copy any data from the original table into the new one, optionally converting the data to a new file format. For some file formats, Impala can do a CREATE TABLE ... LIKE to create the table, but Impala cannot insert data in that file format; in these cases, you must load the data in Hive.

CREATE TABLE AS SELECT:

The CREATE TABLE AS SELECT syntax is a shorthand notation to create a table based on column definitions from another table, and copy data from the source table to the destination table without issuing any separate INSERT statement. This idiom is so popular that it has its own acronym, “CTAS”.

The following examples show how to copy data from a source table T1 to a variety of destinations tables, applying various transformations to the table properties, table layout, or the data itself as part of the operation:

```
-- Sample table to be the source of CTAS operations.
CREATE TABLE t1 (x INT, y STRING);
INSERT INTO t1 VALUES (1, 'one'), (2, 'two'), (3, 'three');

-- Clone all the columns and data from one table to another.
CREATE TABLE clone_of_t1 AS SELECT * FROM t1;
+-------------------+
| summary           |
+-------------------+
| Inserted 3 row(s) |
+-------------------+

-- Clone the columns and data, and convert the data to a different file form
at.
CREATE TABLE parquet_version_of_t1 STORED AS PARQUET AS SELECT * FROM t1;
+-------------------+
| summary           |
+-------------------+
| Inserted 3 row(s) |
+-------------------+
-- Copy only some rows to the new table.
CREATE TABLE subset_of_t1 AS SELECT * FROM t1 WHERE x >= 2;
+-------------------+
| summary           |
+-------------------+
| Inserted 2 row(s) |
+-------------------+
-- Same idea as CREATE TABLE LIKE: clone table layout but do not copy any d
ata.
CREATE TABLE empty_clone_of_t1 AS SELECT * FROM t1 WHERE 1=0;
+-------------------+
| summary           |
+-------------------+
| Inserted 0 row(s) |
+-------------------+

-- Reorder and rename columns and transform the data.
CREATE TABLE t5 AS SELECT upper(y) AS s, x+1 AS a, 'Entirely new column' AS
n FROM t1;
+-------------------+
| summary           |
+-------------------+
| Inserted 3 row(s) |
+-------------------+
SELECT * FROM t5;
+-------+---+---------------------+
| s     | a | n                   |
+-------+---+---------------------+
| ONE   | 2 | Entirely new column |
| TWO   | 3 | Entirely new column |
| THREE | 4 | Entirely new column |
+-------+---+---------------------+
```

See the SELECT statement on page 208 topic for details about query syntax for the SELECT portion of a CREA
TE TABLE AS SELECT statement.

The newly created table inherits the column names that you select from the original table, which you can override by specifying column aliases in the query. Any column or table comments from the original table are not carried over to the new table.

Note: When using the STORED AS clause with a CREATE TABLE AS SELECT statement, the destination table must be a file format that Impala can write to: currently, text or Parquet. You cannot specify an Avro, SequenceFile, or RCFile table as the destination table for a CTAS operation.

Prior to Impala 2.5 you could use a partitioned table as the source and copy data from it, but could not specify any partitioning clauses for the new table. In Impala 2.5 and higher, you can now use the PARTITIONED BY clause with a CREATE TABLE AS SELECT statement. The following example demonstrates how you can copy data from an unpartitioned table in a CREATE TABLE AS SELECT operation, creating a new partitioned table in the process. The main syntax consideration is the column order in the PARTITIONED BY clause and the select list: the partition key columns must be listed last in the select list, in the same order as in the PARTITIONED BY clause. Therefore, in this case, the column order in the destination table is different from the source table. You also only specify the column names in the PARTITIONED BY clause, not the data types or column comments.

```
create table partitions_no (year smallint, month tinyint, s string);
insert into partitions_no values (2016, 1, 'January 2016'),
  (2016, 2, 'February 2016'), (2016, 3, 'March 2016');

-- Prove that the source table is not partitioned.
show partitions partitions_no;
ERROR: AnalysisException: Table is not partitioned: ctas_partition_by.partit
ions_no
-- Create new table with partitions based on column values from source tabl
e.
create table partitions_yes partitioned by (year, month)
  as select s, year, month from partitions_no;
+--------------------+
| summary            |
+--------------------+
| Inserted 3 row(s) |
+--------------------+

-- Prove that the destination table is partitioned.
show partitions partitions_yes;
+-------+-------+-------+--------+------+...
| year  | month | #Rows | #Files | Size |...
+-------+-------+-------+--------+------+...
| 2016  | 1     | -1    | 1      | 13B  |...
| 2016  | 2     | -1    | 1      | 14B  |...
| 2016  | 3     | -1    | 1      | 11B  |...
| Total |       | -1    | 3      | 38B  |...
+-------+-------+-------+--------+------+...
```

The most convenient layout for partitioned tables is with all the partition key columns at the end. The CTAS PARTITIONED BY syntax requires that column order in the select list, resulting in that same column order in the destination table.

```
describe partitions_no;
+-------+----------+---------+
| name  | type     | comment |
+-------+----------+---------+
| year  | smallint |         |
| month | tinyint  |         |
| s     | string   |         |
+-------+----------+---------+

-- The CTAS operation forced us to put the partition key columns last.
-- Having those columns last works better with idioms such as SELECT *
-- for partitioned tables.
describe partitions_yes;
+-------+----------+---------+
| name  | type     | comment |
+-------+----------+---------+
| s     | string   |         |
| year  | smallint |         |
```

```
| month | tinyint  |         |
+-------+----------+---------+
```

Attempting to use a select list with the partition key columns not at the end results in an error due to a column name mismatch:

```
-- We expect this CTAS to fail because non-key column S
-- comes after key columns YEAR and MONTH in the select list.
create table partitions_maybe partitioned by (year, month)
  as select year, month, s from partitions_no;
ERROR: AnalysisException: Partition column name mismatch: year != month
```

As part of a CTAS operation, you can convert the data to any file format that Impala can write (currently, TEXTFILE and PARQUET). You cannot specify the lower-level properties of a text table, such as the delimiter.

Sorting considerations: Although you can specify an ORDER BY clause in an INSERT ... SELECT statement, any ORDER BY clause is ignored and the results are not necessarily sorted. An INSERT ... SELECT operation potentially creates many different data files, prepared by different executor Impala daemons, and therefore the notion of the data being stored in sorted order is impractical.

CREATE TABLE LIKE PARQUET:

The variation CREATE TABLE ... LIKE PARQUET '*HDFS_PATH_OF_PARQUET_FILE*' lets you skip the column definitions of the CREATE TABLE statement. The column names and data types are automatically configured based on the organization of the specified Parquet data file, which must already reside in HDFS. You can use a data file located outside the Impala database directories, or a file from an existing Impala Parquet table; either way, Impala only uses the column definitions from the file and does not use the HDFS location for the LOCATION attribute of the new table. (Although you can also specify the enclosing directory with the LOCATION attribute, to both use the same schema as the data file and point the Impala table at the associated directory for querying.)

The following considerations apply when you use the CREATE TABLE LIKE PARQUET technique:

- Any column comments from the original table are not preserved in the new table. Each column in the new table has a comment stating the low-level Parquet field type used to deduce the appropriate SQL column type.
- If you use a data file from a partitioned Impala table, any partition key columns from the original table are left out of the new table, because they are represented in HDFS directory names rather than stored in the data file. To preserve the partition information, repeat the same PARTITION clause as in the original CREATE TABLE statement.
- The file format of the new table defaults to text, as with other kinds of CREATE TABLE statements. To make the new table also use Parquet format, include the clause STORED AS PARQUET in the CREATE TABLE LIKE PARQUET statement.
- If the Parquet data file comes from an existing Impala table, currently, any TINYINT or SMALLINT columns are turned into INT columns in the new table. Internally, Parquet stores such values as 32-bit integers.
- When the destination table uses the Parquet file format, the CREATE TABLE AS SELECT and INSERT ... SELECT statements always create at least one data file, even if the SELECT part of the statement does not match any rows. You can use such an empty Parquet data file as a template for subsequent CREATE TABLE LIKE PARQUET statements.

For more details about creating Parquet tables, and examples of the CREATE TABLE LIKE PARQUET syntax, see *Using the Parquet file format with Impala tables*.

Visibility and Metadata (TBLPROPERTIES and WITH SERDEPROPERTIES clauses):

You can associate arbitrary items of metadata with a table by specifying the TBLPROPERTIES clause. This clause takes a comma-separated list of key-value pairs and stores those items in the metastore database. You can also change the table properties later with an ALTER TABLE statement. You can observe the table properties for different delimiter and escape characters using the DESCRIBE FORMATTED command, and change those settings for an existing table with ALTER TABLE ... SET TBLPROPERTIES.

You can also associate SerDes properties with the table by specifying key-value pairs through the WITH SERDEPROPERTIES clause. This metadata is not used by Impala, which has its own built-in serializer and deserializer for the file formats it supports. Particular property values might be needed for Hive compatibility with certain variations of file formats, particularly Avro.

Some DDL operations that interact with other Hadoop components require specifying particular values in the SERDEPROPERTIES or TBLPROPERTIES fields, such as creating an Avro table or an HBase table. (You typically create HBase tables in Hive, because they require additional clauses not currently available in Impala.)

To see the column definitions and column comments for an existing table, for example before issuing a CREATE TABLE ... LIKE or a CREATE TABLE ... AS SELECT statement, issue the statement DESCRIBE *TABLE_NAME*. To see even more detail, such as the location of data files and the values for clauses such as ROW FORMAT and STORED AS, issue the statement DESCRIBE FORMATTED *TABLE_NAME*. DESCRIBE FORMATTED is also needed to see any overall table comment (as opposed to individual column comments).

After creating a table, your `impala-shell` session or another `impala-shell` connected to the same node can immediately query that table. There might be a brief interval (one statestore heartbeat) before the table can be queried through a different Impala node. To make the CREATE TABLE statement return only when the table is recognized by all Impala nodes in the cluster, enable the SYNC_DDL query option.

HDFS caching (CACHED IN clause):

If you specify the CACHED IN clause, any existing or future data files in the table directory or the partition subdirectories are designated to be loaded into memory with the HDFS caching mechanism.

In Impala 2.2 and higher, the optional WITH REPLICATION clause for CREATE TABLE and ALTER TABLE lets you specify a *replication factor*, the number of hosts on which to cache the same data blocks. When Impala processes a cached data block, where the cache replication factor is greater than 1, Impala randomly selects a host that has a cached copy of that data block. This optimization avoids excessive CPU usage on a single host when the same cached data block is processed multiple times. Cloudera recommends specifying a value greater than or equal to the HDFS block replication factor.

Column order:

If you intend to use the table to hold data files produced by some external source, specify the columns in the same order as they appear in the data files.

If you intend to insert or copy data into the table through Impala, or if you have control over the way externally produced data files are arranged, use your judgment to specify columns in the most convenient order:

- If certain columns are often NULL, specify those columns last. You might produce data files that omit these trailing columns entirely. Impala automatically fills in the NULL values if so.
- If an unpartitioned table will be used as the source for an INSERT ... SELECT operation into a partitioned table, specify last in the unpartitioned table any columns that correspond to partition key columns in the partitioned table, and in the same order as the partition key columns are declared in the partitioned table. This technique lets you use INSERT ... SELECT * when copying data to the partitioned table, rather than specifying each column name individually.
- If you specify columns in an order that you later discover is suboptimal, you can sometimes work around the problem without recreating the table. You can create a view that selects columns from the original table in a permuted order, then do a SELECT * from the view. When inserting data into a table, you can specify a permuted order for the inserted columns to match the order in the destination table.

Hive considerations:

Impala queries can make use of metadata about the table and columns, such as the number of rows in a table or the number of different values in a column. Prior to Impala 1.2.2, to create this metadata, you issued the ANALYZE TABLE statement in Hive to gather this information, after creating the table and loading representative data into it. In Impala 1.2.2 and higher, the COMPUTE STATS statement produces these statistics within Impala, without needing to use Hive at all.

HBase considerations:

Note:

The Impala CREATE TABLE statement cannot create an HBase table, because it currently does not support the STORED BY clause needed for HBase tables. Create such tables in Hive, then query them through Impala.

Amazon S3 considerations:

To create a table where the data resides in the Amazon Simple Storage Service (S3), specify a s3a:// prefix LOCATION attribute pointing to the data files in S3.

In Impala 2.6 and higher, you can use this special LOCATION syntax as part of a CREATE TABLE AS SELECT statement.

In Impala 2.6 and higher, Impala DDL statements such as CREATE DATABASE, CREATE TABLE, DROP DATABASE CASCADE, DROP TABLE, and ALTER TABLE [ADD|DROP] PARTITION can create or remove folders as needed in the Amazon S3 system. Prior to Impala 2.6, you had to create folders yourself and point Impala database, tables, or partitions at them, and manually remove folders when no longer needed.

Sorting considerations: Although you can specify an ORDER BY clause in an INSERT ... SELECT statement, any ORDER BY clause is ignored and the results are not necessarily sorted. An INSERT ... SELECT operation potentially creates many different data files, prepared by different executor Impala daemons, and therefore the notion of the data being stored in sorted order is impractical.

HDFS considerations:

The CREATE TABLE statement for an internal table creates a directory in HDFS. The CREATE EXTERNAL TABLE statement associates the table with an existing HDFS directory, and does not create any new directory in HDFS. To locate the HDFS data directory for a table, issue a DESCRIBE FORMATTED *TABLE* statement. To examine the contents of that HDFS directory, use an OS command such as hdfs dfs -ls hdfs://*PATH*, either from the OS command line or through the shell or `!` commands in `impala-shell`.

The CREATE TABLE AS SELECT syntax creates data files under the table data directory to hold any data copied by the INSERT portion of the statement. (Even if no data is copied, Impala might create one or more empty data files.)

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have both execute and write permission for the database directory where the table is being created.

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Cancellation: Certain multi-stage statements (CREATE TABLE AS SELECT and COMPUTE STATS) can be cancelled during some stages, when running INSERT or SELECT operations internally. To cancel this statement, use Ctrl-C from the `impala-shell` interpreter, the Cancel button from the Watch page in Hue, Actions > Cancel from the Queries list in Cloudera Manager, or Cancel from the list of in-flight queries (for a particular node) on the Queries tab in the Impala web UI (port 25000).

Related Information

Impala tables

Partitioning

Hadoop file formats supported

SQL transactions in Impala

CREATE VIEW statement

The CREATE VIEW statement lets you create a shorthand abbreviation for a more complicated query. The base query can involve joins, expressions, reordered columns, column aliases, and other SQL features that can make a query hard to understand or maintain.

Because a view is purely a logical construct (an alias for a query) with no physical data behind it, ALTER VIEW only involves changes to metadata in the metastore database, not any data files in HDFS.

Syntax:

```
CREATE VIEW [IF NOT EXISTS] VIEW_NAME
    [(COLUMN_NAME [COMMENT 'COLUMN_COMMENT'][, ...])]
    [COMMENT 'VIEW_COMMENT']
    [TBLPROPERTIES ('NAME' = 'VALUE'[, ...])]
  AS SELECT_STATEMENT
```

Statement type: DDL

Usage notes:

The CREATE VIEW statement can be useful in scenarios such as the following:

- To turn even the most lengthy and complicated SQL query into a one-liner. You can issue simple queries against the view from applications, scripts, or interactive queries in `impala-shell`. For example:

  ```
  select * from VIEW_NAME;
  select * from VIEW_NAME order by c1 desc limit 10;
  ```

  The more complicated and hard-to-read the original query, the more benefit there is to simplifying the query using a view.
- To hide the underlying table and column names, to minimize maintenance problems if those names change. In that case, you re-create the view using the new names, and all queries that use the view rather than the underlying tables keep running with no changes.
- To experiment with optimization techniques and make the optimized queries available to all applications. For example, if you find a combination of WHERE conditions, join order, join hints, and so on that works the best for a class of queries, you can establish a view that incorporates the best-performing techniques. Applications can then make relatively simple queries against the view, without repeating the complicated and optimized logic over and over. If you later find a better way to optimize the original query, when you re-create the view, all the applications immediately take advantage of the optimized base query.
- To simplify a whole class of related queries, especially complicated queries involving joins between multiple tables, complicated expressions in the column list, and other SQL syntax that makes the query difficult to understand and debug. For example, you might create a view that joins several tables, filters using several WHERE conditions, and selects several columns from the result set. Applications might issue queries against this view that only vary in their LIMIT, ORDER BY, and similar simple clauses.

For queries that require repeating complicated clauses over and over again, for example in the select list, ORDER BY, and GROUP BY clauses, you can use the WITH clause as an alternative to creating a view.

You can optionally specify the table-level and the column-level comments as in the CREATE TABLE statement.

Complex type considerations:

For tables containing complex type columns (ARRAY, STRUCT, or MAP), you typically use join queries to refer to the complex values. You can use views to hide the join notation, making such tables seem like traditional denormalized tables, and making those tables queryable by business intelligence tools that do not have built-in support for those complex types.

Because you cannot directly issue SELECT *COL_NAME* against a column of complex type, you cannot use a view or a WITH clause to “rename” a column by selecting it with a column alias.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Examples:

```
-- Create a view that is exactly the same as the underlying table.
CREATE VIEW v1 AS SELECT * FROM t1;

-- Create a view that includes only certain columns from the underlying tabl
e.
CREATE VIEW v2 AS SELECT c1, c3, c7 FROM t1;
-- Create a view that filters the values from the underlying table.
CREATE VIEW v3 AS SELECT DISTINCT c1, c3, c7 FROM t1 WHERE c1 IS NOT NULL A
ND c5 > 0;

-- Create a view that that reorders and renames columns from the underlying
table.
CREATE VIEW v4 AS SELECT c4 AS last_name, c6 AS address, c2 AS birth_date
 FROM t1;

-- Create a view that runs functions to convert or transform certain colu
mns.
CREATE VIEW v5 AS SELECT c1, CAST(c3 AS STRING) c3, CONCAT(c4,c5) c5, TRIM(
c6) c6, "Constant" c8 FROM t1;

-- Create a view that hides the complexity of a view query.
CREATE VIEW v6 AS SELECT t1.c1, t2.c2 FROM t1 JOIN t2 ON t1.id = t2.id;
-- Create a view with a column comment and a table comment.
CREATE VIEW v7 (c1 COMMENT 'Comment for c1', c2) COMMENT 'Comment for v7' AS
 SELECT t1.c1, t1.c2 FROM t1;

-- Create a view with tblproperties.
CREATE VIEW v7 (c1 , c2) TBLPROPERTIES ('tblp1' = '1', 'tblp2' = '2') AS SE
LECT t1.c1, t1.c2 FROM t1;
```

DELETE statement

The DELETE statement deletes an arbitrary number of rows from a Kudu table. This statement only works for Impala tables that use the Kudu storage engine.

Syntax:

```
DELETE [FROM] [DATABASE_NAME.]TABLE_NAME [ WHERE WHERE_CONDITIONS ]

DELETE TABLE_REF FROM [JOINED_TABLE_REFS] [ WHERE WHERE_CONDITIONS ]
```

The first form evaluates rows from one table against an optional WHERE clause, and deletes all the rows that match the WHERE conditions, or all rows if WHERE is omitted.

The second form evaluates one or more join clauses, and deletes all matching rows from one of the tables. The join clauses can include non-Kudu tables, but the table from which the rows are deleted must be a Kudu table. The FROM keyword is required in this case, to separate the name of the table whose rows are being deleted from the table names of the join clauses.

Usage notes:

The conditions in the WHERE clause are the same ones allowed for the SELECT statement.

The conditions in the WHERE clause can refer to any combination of primary key columns or other columns. Referring to primary key columns in the WHERE clause is more efficient than referring to non-primary key columns.

If the WHERE clause is omitted, all rows are removed from the table.

Because Kudu currently does not enforce strong consistency during concurrent DML operations, be aware that the results after this statement finishes might be different than you intuitively expect:

- If some rows cannot be deleted because their some primary key columns are not found, due to their being deleted by a concurrent DELETE operation, the statement succeeds but returns a warning.
- A DELETE statement might also overlap with INSERT, UPDATE, or UPSERT statements running concurrently on the same table. After the statement finishes, there might be more or fewer rows than expected in the table because it is undefined whether the DELETE applies to rows that are inserted or updated while the DELETE is in progress.

The number of affected rows is reported in an `impala-shell` message and in the query profile.

Statement type: DML

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMPUTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

The following examples show how to delete rows from a specified table, either all rows or rows that match a WHERE clause:

```
-- Deletes all rows. The FROM keyword is optional.
DELETE FROM kudu_table;
DELETE kudu_table;

-- Deletes 0, 1, or more rows.
-- (If c1 is a single-column primary key, the statement could only
-- delete 0 or 1 rows.)
DELETE FROM kudu_table WHERE c1 = 100;

-- Deletes all rows that match all the WHERE conditions.
DELETE FROM kudu_table WHERE
  (c1 > c2 OR c3 IN ('hello','world')) AND c4 IS NOT NULL;
DELETE FROM t1 WHERE
  (c1 IN (1,2,3) AND c2 > c3) OR c4 IS NOT NULL;
DELETE FROM time_series WHERE
  year = 2016 AND month IN (11,12) AND day > 15;
-- WHERE condition with a subquery.
DELETE FROM t1 WHERE
  c5 IN (SELECT DISTINCT other_col FROM other_table);
-- Does not delete any rows, because the WHERE condition is always false.
DELETE FROM kudu_table WHERE 1 = 0;
```

The following examples show how to delete rows that are part of the result set from a join:

```
-- Remove _all_ rows from t1 that have a matching X value in t2.
DELETE t1 FROM t1 JOIN t2 ON t1.x = t2.x;
-- Remove _some_ rows from t1 that have a matching X value in t2.
DELETE t1 FROM t1 JOIN t2 ON t1.x = t2.x
  WHERE t1.y = FALSE and t2.z > 100;

-- Delete from a Kudu table based on a join with a non-Kudu table.
DELETE t1 FROM kudu_table t1 JOIN non_kudu_table t2 ON t1.x = t2.x;

-- The tables can be joined in any order as long as the Kudu table
-- is specified as the deletion target.
DELETE t2 FROM non_kudu_table t1 JOIN kudu_table t2 ON t1.x = t2.x;
```

Related Information

SELECT statement

DESCRIBE statement

The DESCRIBE statement displays metadata about a table, such as the column names and their data types. In Impala 2.3 and higher, you can specify the name of a complex type column, which takes the form of a dotted path. The path might include multiple components in the case of a nested type definition. In Impala 2.5 and higher, the DESCRIBE DATABASE form can display information about a database.

Syntax:

```
DESCRIBE [DATABASE] [FORMATTED|EXTENDED] OBJECT_NAME

object_name ::=
    [DB_NAME.]TABLE_NAME[.COMPLEX_COL_NAME ...]
  | DB_NAME
```

You can use the abbreviation DESC for the DESCRIBE statement.

The DESCRIBE FORMATTED variation displays additional information, in a format familiar to users of Apache Hive. The extra information includes low-level details such as whether the table is internal or external, when it was created, the file format, the location of the data in HDFS, whether the object is a table or a view, and (for views) the text of the query from the view definition.

Note: The Compressed field is not a reliable indicator of whether the table contains compressed data. It typically always shows No, because the compression settings only apply during the session that loads data and are not stored persistently with the table metadata.

Describing databases:

By default, the DESCRIBE output for a database includes the location and the comment, which can be set by the LOCATION and COMMENT clauses on the CREATE DATABASE statement.

The additional information displayed by the FORMATTED or EXTENDED keyword includes the HDFS user ID that is considered the owner of the database, and any optional database properties. The properties could be specified by the WITH DBPROPERTIES clause if the database is created using a Hive CREATE DATABASE statement. Impala currently does not set or do any special processing based on those properties.

The following examples show the variations in syntax and output for describing databases. This feature is available in Impala 2.5 and higher.

```
describe database default;
+---------+----------------------+-----------------------+
```

```
| name    | location             | comment               |
+---------+----------------------+-----------------------+
| default | /user/hive/warehouse | Default Hive database |
+---------+----------------------+-----------------------+

describe database formatted default;
+---------+----------------------+-----------------------+
| name    | location             | comment               |
+---------+----------------------+-----------------------+
| default | /user/hive/warehouse | Default Hive database |
| Owner:  |                      |                       |
|         | public               | ROLE                  |
+---------+----------------------+-----------------------+

describe database extended default;
+---------+----------------------+-----------------------+
| name    | location             | comment               |
+---------+----------------------+-----------------------+
| default | /user/hive/warehouse | Default Hive database |
| Owner:  |                      |                       |
|         | public               | ROLE                  |
+---------+----------------------+-----------------------+
```

Describing tables:

If the DATABASE keyword is omitted, the default for the DESCRIBE statement is to refer to a table.

If you have the SELECT privilege on a subset of the table columns and no other relevant table/database/server-level privileges, DESCRIBE returns the data from the columns you have access to.

If you have the SELECT privilege on a subset of the table columns and no other relevant table/database/server-level privileges, DESCRIBE FORMATTED/EXTENDED does not return the LOCATION field. The LOCATION data is shown if you have any privilege on the table, the containing database or the server.

```
-- By default, the table is assumed to be in the current database.
describe my_table;
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| x    | int    |         |
| s    | string |         |
+------+--------+---------+

-- Use a fully qualified table name to specify a table in any database.
describe my_database.my_table;
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| x    | int    |         |
| s    | string |         |
+------+--------+---------+

-- The formatted or extended output includes additional useful information.
-- The LOCATION field is especially useful to know for DDL statements and HD
FS commands
-- during ETL jobs. (The LOCATION includes a full hdfs:// URL, omitted here
for readability.)
describe formatted my_table;
+-------------------------------+----------------------------------------
-----+----------------------+
| name                          | type
  | comment              |
```

```
+-----------------------------+-----------------------------------------
---+----------------------+
| # col_name                  | data_type
  | comment              |
|                             | NULL
   | NULL                 |
| x                           | int
  | NULL                 |
| s                           | string
  | NULL                 |
|                             | NULL
  | NULL                 |
| # Detailed Table Information | NULL
   | NULL                 |
| Database:                   | my_database
  | NULL                 |
| Owner:                      | jrussell
  | NULL                 |
| CreateTime:                 | Fri Mar 18 15:58:00 PDT 2016
  | NULL                 |
| LastAccessTime:             | UNKNOWN
   | NULL                 |
| Protect Mode:               | None
  | NULL                 |
| Retention:                  | 0
  | NULL                 |
| Location:                   | /user/hive/warehouse/my_database.db/my_tabl
e | NULL                 |
| Table Type:                 | MANAGED_TABLE
   | NULL                 |
| Table Parameters:           | NULL
  | NULL                 |
|                             | transient_lastDdlTime
  | 1458341880           |
|                             | NULL
  | NULL                 |
| # Storage Information       | NULL
   | NULL                 |
| SerDe Library:              | org. ... .LazySimpleSerDe
  | NULL                 |
| InputFormat:                | org.apache.hadoop.mapred.TextInputFormat
  | NULL                 |
| OutputFormat:               | org. ... .HiveIgnoreKeyTextOutputFormat
  | NULL                 |
| Compressed:                 | No
   | NULL                 |
| Num Buckets:                | 0
  | NULL                 |
| Bucket Columns:             | []
  | NULL                 |
| Sort Columns:               | []
  | NULL                 |
+-----------------------------+----------------------------------------
-----+----------------------+
```

Complex type considerations:

Because the column definitions for complex types can become long, particularly when such types are nested, the DESCRIBE statement uses special formatting for complex type columns to make the output readable.

For the ARRAY, STRUCT, and MAP types available in Impala 2.3 and higher, the DESCRIBE output is formatted to avoid excessively long lines for multiple fields within a STRUCT, or a nested sequence of complex types.

You can pass a multi-part qualified name to DESCRIBE to specify an ARRAY, STRUCT, or MAP column and visualize its structure as if it were a table. For example, if table T1 contains an ARRAY column A1, you could issue the statement DESCRIBE t1.a1. If table T1 contained a STRUCT column S1, and a field F1 within the STRUCT was a MAP, you could issue the statement DESCRIBE t1.s1.f1. An ARRAY is shown as a two-column table, with ITEM and POS columns. A STRUCT is shown as a table with each field representing a column in the table. A MAP is shown as a two-column table, with KEY and VALUE columns.

For example, here is the DESCRIBE output for a table containing a single top-level column of each complex type:

```
create table t1 (x int, a array<int>, s struct<f1: string, f2: bigint>, m ma
p<string,int>) stored as parquet;

describe t1;
+------+-----------------+---------+
| name | type            | comment |
+------+-----------------+---------+
| x    | int             |         |
| a    | array<int>      |         |
| s    | struct<         |         |
|      |   f1:string,    |         |
|      |   f2:bigint     |         |
|      | >               |         |
| m    | map<string,int> |         |
+------+-----------------+---------+
```

Here are examples showing how to “drill down” into the layouts of complex types, including using multi-part names to examine the definitions of nested types. The < > delimiters identify the columns with complex types; these are the columns where you can descend another level to see the parts that make up the complex type. This technique helps you to understand the multi-part names you use as table references in queries involving complex types, and the corresponding column names you refer to in the SELECT list. These tables are from the “nested TPC-H” schema, shown in detail in *Sample schema and data for experimenting with Impala complex types*.

The REGION table contains an ARRAY of STRUCT elements:

- The first DESCRIBE specifies the table name, to display the definition of each top-level column.
- The second DESCRIBE specifies the name of a complex column, REGION.R_NATIONS, showing that when you include the name of an ARRAY column in a FROM clause, that table reference acts like a two-column table with columns ITEM and POS.
- The final DESCRIBE specifies the fully qualified name of the ITEM field, to display the layout of its underlying STRUCT type in table format, with the fields mapped to column names.

```
-- #1: The overall layout of the entire table.
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

-- #2: The ARRAY column within the table.
describe region.r_nations;
+------+------------------------+---------+
| name | type                   | comment |
```

```
+------+-------------------------+---------+
| item | struct<                 |         |
|      |   n_nationkey:smallint, |         |
|      |   n_name:string,        |         |
|      |   n_comment:string      |         |
|      | >                       |         |
| pos  | bigint                  |         |
+------+-------------------------+---------+

-- #3: The STRUCT that makes up each ARRAY element.
--     The fields of the STRUCT act like columns of a table.
describe region.r_nations.item;
+-------------+----------+---------+
| name        | type     | comment |
+-------------+----------+---------+
| n_nationkey | smallint |         |
| n_name      | string   |         |
| n_comment   | string   |         |
+-------------+----------+---------+
```

The CUSTOMER table contains an ARRAY of STRUCT elements, where one field in the STRUCT is another ARRAY of STRUCT elements:

- Again, the initial DESCRIBE specifies only the table name.
- The second DESCRIBE specifies the qualified name of the complex column, CUSTOMER.C_ORDERS, showing how an ARRAY is represented as a two-column table with columns ITEM and POS.
- The third DESCRIBE specifies the qualified name of the ITEM of the ARRAY column, to see the structure of the nested ARRAY. Again, it has has two parts, ITEM and POS. Because the ARRAY contains a STRUCT, the layout of the STRUCT is shown.
- The fourth and fifth DESCRIBE statements drill down into a STRUCT field that is itself a complex type, an ARRAY of STRUCT. The ITEM portion of the qualified name is only required when the ARRAY elements are anonymous. The fields of the STRUCT give names to any other complex types nested inside the STRUCT. Therefore, the DESCRIBE parameters CUSTOMER.C_ORDERS.ITEM.O_LINEITEMS and CUSTOMER.C_ORDERS.O_LINEITEMS are equivalent. (For brevity, leave out the ITEM portion of a qualified name when it is not required.)
- The final DESCRIBE shows the layout of the deeply nested STRUCT type. Because there are no more complex types nested inside this STRUCT, this is as far as you can drill down into the layout for this table.

```
-- #1: The overall layout of the entire table.
describe customer;
+--------------+-----------------------------------+
| name         | type                              |
+--------------+-----------------------------------+
| c_custkey    | bigint                            |
... more scalar columns ...
| c_orders     | array<struct<                     |
|              |   o_orderkey:bigint,              |
|              |   o_orderstatus:string,           |
|              |   o_totalprice:decimal(12,2),     |
|              |   o_orderdate:string,             |
|              |   o_orderpriority:string,         |
|              |   o_clerk:string,                 |
|              |   o_shippriority:int,             |
|              |   o_comment:string,               |
|              |   o_lineitems:array<struct<       |
|              |     l_partkey:bigint,             |
|              |     l_suppkey:bigint,             |
|              |     l_linenumber:int,             |
|              |     l_quantity:decimal(12,2),     |
|              |     l_extendedprice:decimal(12,2), |
```

```
|              |     l_discount:decimal(12,2),      |
|              |     l_tax:decimal(12,2),           |
|              |     l_returnflag:string,           |
|              |     l_linestatus:string,           |
|              |     l_shipdate:string,             |
|              |     l_commitdate:string,           |
|              |     l_receiptdate:string,          |
|              |     l_shipinstruct:string,         |
|              |     l_shipmode:string,             |
|              |     l_comment:string               |
|              |   >>                               |
|              | >>                                 |
+--------------+------------------------------------+

-- #2: The ARRAY column within the table.
describe customer.c_orders;
+------+-----------------------------------+
| name | type                              |
+------+-----------------------------------+
| item | struct<                           |
|      |   o_orderkey:bigint,              |
|      |   o_orderstatus:string,           |
... more struct fields ...
|      |   o_lineitems:array<struct<       |
|      |     l_partkey:bigint,             |
|      |     l_suppkey:bigint,             |
... more nested struct fields ...
|      |     l_comment:string              |
|      |   >>                              |
|      | >                                 |
| pos  | bigint                            |
+------+-----------------------------------+

-- #3: The STRUCT that makes up each ARRAY element.
--     The fields of the STRUCT act like columns of a table.
describe customer.c_orders.item;
+-----------------+---------------------------------+
| name            | type                            |
+-----------------+---------------------------------+
| o_orderkey      | bigint                          |
| o_orderstatus   | string                          |
| o_totalprice    | decimal(12,2)                   |
| o_orderdate     | string                          |
| o_orderpriority | string                          |
| o_clerk         | string                          |
| o_shippriority  | int                             |
| o_comment       | string                          |
| o_lineitems     | array<struct<                   |
|                 |   l_partkey:bigint,             |
|                 |   l_suppkey:bigint,             |
... more struct fields ...
|                 |   l_comment:string              |
|                 | >>                              |
+-----------------+---------------------------------+

-- #4: The ARRAY nested inside the STRUCT elements of the first ARRAY.
describe customer.c_orders.item.o_lineitems;
+------+---------------------------------+
| name | type                            |
+------+---------------------------------+
| item | struct<                         |
|      |   l_partkey:bigint,             |
|      |   l_suppkey:bigint,             |
... more struct fields ...
```

```
|      |   l_comment:string                |
|      | >                                 |
| pos  | bigint                            |
+------+-----------------------------------+

-- #5: Shorter form of the previous DESCRIBE. Omits the .ITEM portion of the
 name
--     because O_LINEITEMS and other field names provide a way to refer to
 things
--     inside the ARRAY element.
describe customer.c_orders.o_lineitems;
+------+-----------------------------------+
| name | type                              |
+------+-----------------------------------+
| item | struct<                           |
|      |   l_partkey:bigint,               |
|      |   l_suppkey:bigint,               |
... more struct fields ...
|      |   l_comment:string                |
|      | >                                 |
| pos  | bigint                            |
+------+-----------------------------------+
-- #6: The STRUCT representing ARRAY elements nested inside
--     another ARRAY of STRUCTs. The lack of any complex types
--     in this output means this is as far as DESCRIBE can
--     descend into the table layout.
describe customer.c_orders.o_lineitems.item;
+-----------------+---------------+
| name            | type          |
+-----------------+---------------+
| l_partkey       | bigint        |
| l_suppkey       | bigint        |
... more scalar columns ...
| l_comment       | string        |
+-----------------+---------------+
```

Usage notes:

After the `impalad` daemons are restarted, the first query against a table can take longer than subsequent queries, because the metadata for the table is loaded before the query is processed. This one-time delay for each table can cause misleading results in benchmark tests or cause unnecessary concern. To “warm up” the Impala metadata cache, you can issue a DESCRIBE statement in advance for each table you intend to access later.

When you are dealing with data files stored in HDFS, sometimes it is important to know details such as the path of the data files for an Impala table, and the hostname for the namenode. You can get this information from the DESC RIBE FORMATTED output. You specify HDFS URIs or path specifications with statements such as LOAD DATA and the LOCATION clause of CREATE TABLE or ALTER TABLE. You might also use HDFS URIs or paths with Linux commands such as `hadoop` and `hdfs` to copy, rename, and so on, data files in HDFS.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Each table can also have associated table statistics and column statistics. To see these categories of information, use the SHOW TABLE STATS *TABLE_NAME* and SHOW COLUMN STATS *TABLE_NAME* statements. See the *SHOW statement* topic for details.

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMP UTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

The following example shows the results of both a standard DESCRIBE and DESCRIBE FORMATTED for different kinds of schema objects:

- DESCRIBE for a table or a view returns the name, type, and comment for each of the columns. For a view, if the column value is computed by an expression, the column name is automatically generated as _c0, _c1, and so on depending on the ordinal number of the column.
- A table created with no special format or storage clauses is designated as a MANAGED_TABLE (an “internal table” in Impala terminology). Its data files are stored in an HDFS directory under the default Hive data directory. By default, it uses Text data format.
- A view is designated as VIRTUAL_VIEW in DESCRIBE FORMATTED output. Some of its properties are NULL or blank because they are inherited from the base table. The text of the query that defines the view is part of the DESCRIBE FORMATTED output.
- A table with additional clauses in the CREATE TABLE statement has differences in DESCRIBE FORMATTED output. The output for T2 includes the EXTERNAL_TABLE keyword because of the CREATE EXTERNAL TABLE syntax, and different InputFormat and OutputFormat fields to reflect the Parquet file format.

```
[localhost:21000] > create table t1 (x int, y int, s string);
Query: create table t1 (x int, y int, s string)
[localhost:21000] > describe t1;
Query: describe t1
Query finished, fetching results ...
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| x    | int    |         |
| y    | int    |         |
| s    | string |         |
+------+--------+---------+
Returned 3 row(s) in 0.13s
[localhost:21000] > describe formatted t1;
Query: describe formatted t1
Query finished, fetching results ...
+------------------------------+-------------------------------------------
+------------+
| name                         | type
 | comment    |
+------------------------------+-------------------------------------------
+------------+
| # col_name                   | data_type
| comment    |
|                              | NULL
 | NULL       |
| x                            | int
 | None       |
| y                            | int
 | None       |
| s                            | string
| None       |
|                              | NULL
 | NULL       |
| # Detailed Table Information | NULL
 | NULL       |
| Database:                    | describe_formatted
 | NULL       |
| Owner:                       | doc_demo
| NULL       |
| CreateTime:                  | Mon Jul 22 17:03:16 EDT 2013
 | NULL       |
| LastAccessTime:              | UNKNOWN
 | NULL       |
```

```
| Protect Mode:                | None
 | NULL       |
| Retention:                   | 0
| NULL       |
| Location:                    | hdfs://127.0.0.1:8020/user/hive/warehouse/
 |            |
|                              |   describe_formatted.db/t1
 | NULL       |
| Table Type:                  | MANAGED_TABLE
 | NULL       |
| Table Parameters:            | NULL
| NULL       |
|                              | transient_lastDdlTime
 | 1374526996 |
|                              | NULL
 | NULL       |
| # Storage Information        | NULL
 | NULL       |
| SerDe Library:               | org.apache.hadoop.hive.serde2.lazy.
|            |
|                              |   LazySimpleSerDe
 | NULL       |
| InputFormat:                 | org.apache.hadoop.mapred.TextInputFormat
 | NULL       |
| OutputFormat:                | org.apache.hadoop.hive.ql.io.
 |            |
|                              |   HiveIgnoreKeyTextOutputFormat
| NULL       |
| Compressed:                  | No
 | NULL       |
| Num Buckets:                 | 0
 | NULL       |
| Bucket Columns:              | []
 | NULL       |
| Sort Columns:                | []
| NULL       |
+------------------------------+-------------------------------------------
+------------+
Returned 26 row(s) in 0.03s
[localhost:21000] > create view v1 as select x, upper(s) from t1;
Query: create view v1 as select x, upper(s) from t1
[localhost:21000] > describe v1;
Query: describe v1
Query finished, fetching results ...
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| x    | int    |         |
| _c1  | string |         |
+------+--------+---------+
Returned 2 row(s) in 0.10s
[localhost:21000] > describe formatted v1;
Query: describe formatted v1
Query finished, fetching results ...
+------------------------------+------------------------------+------------
----------+
| name                         | type                         | comment
        |
+------------------------------+------------------------------+----------
------------+
| # col_name                   | data_type                    | comment
        |
|                              | NULL                         | NULL
        |
```

```
| x                             | int                           | None
        |
| _c1                           | string                        | None
         |
|                               | NULL                          | NULL
        |
| # Detailed Table Information | NULL                          | NULL
        |
| Database:                     | describe_formatted            | NULL
        |
| Owner:                        | doc_demo                      | NULL
         |
| CreateTime:                   | Mon Jul 22 16:56:38 EDT 2013 | NULL
        |
| LastAccessTime:               | UNKNOWN                       | NULL
        |
| Protect Mode:                 | None                          | NULL
        |
| Retention:                    | 0                             | NULL
         |
| Table Type:                   | VIRTUAL_VIEW                  | NULL
        |
| Table Parameters:             | NULL                          | NULL
        |
|                               | transient_lastDdlTime         | 1374526598
        |
|                               | NULL                          | NULL
         |
| # Storage Information         | NULL                          | NULL
        |
| SerDe Library:                | null                          | NULL
        |
| InputFormat:                  | null                          | NULL
        |
| OutputFormat:                 | null                          | NULL
         |
| Compressed:                   | No                            | NULL
        |
| Num Buckets:                  | 0                             | NULL
        |
| Bucket Columns:               | []                            | NULL
        |
| Sort Columns:                 | []                            | NULL
         |
|                               | NULL                          | NULL
        |
| # View Information            | NULL                          | NULL
        |
| View Original Text:           | SELECT x, upper(s) FROM t1    | NULL
        |
| View Expanded Text:           | SELECT x, upper(s) FROM t1    | NULL
         |
+-------------------------------+-------------------------------+-----------
-----------+
Returned 28 row(s) in 0.03s
[localhost:21000] > create external table t2 (x int, y int, s string) stored
 as parquet location '/user/doc_demo/sample_data';
[localhost:21000] > describe formatted t2;
Query: describe formatted t2
Query finished, fetching results ...
+-------------------------------+-----------------------------------------
---------+------------+
| name                          | type
        | comment    |
```

```
+------------------------------+----------------------------------------
-----------+------------+
| # col_name                   | data_type
        | comment    |
|                              | NULL
        | NULL       |
| x                            | int
        | None       |
| y                            | int
         | None       |
| s                            | string
        | None       |
|                              | NULL
        | NULL       |
| # Detailed Table Information | NULL
        | NULL       |
| Database:                    | describe_formatted
         | NULL       |
| Owner:                       | doc_demo
        | NULL       |
| CreateTime:                  | Mon Jul 22 17:01:47 EDT 2013
        | NULL       |
| LastAccessTime:              | UNKNOWN
        | NULL       |
| Protect Mode:                | None
         | NULL       |
| Retention:                   | 0
        | NULL       |
| Location:                    | hdfs://127.0.0.1:8020/user/doc_demo/sample
_data    | NULL       |
| Table Type:                  | EXTERNAL_TABLE
        | NULL       |
| Table Parameters:            | NULL
         | NULL       |
|                              | EXTERNAL
        | TRUE       |
|                              | transient_lastDdlTime
        | 1374526907 |
|                              | NULL
        | NULL       |
| # Storage Information        | NULL
         | NULL       |
| SerDe Library:               | org.apache.hadoop.hive.serde2.lazy.LazySi
mpleSerDe | NULL       |
| InputFormat:                 | com.cloudera.impala.hive.serde.ParquetInpu
tFormat  | NULL       |
| OutputFormat:                | com.cloudera.impala.hive.serde.ParquetOutpu
tFormat | NULL       |
| Compressed:                  | No
         | NULL       |
| Num Buckets:                 | 0
        | NULL       |
| Bucket Columns:              | []
        | NULL       |
| Sort Columns:                | []
        | NULL       |
+------------------------------+----------------------------------------
-----------+------------+
Returned 27 row(s) in 0.17s
```

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all directories that are part of the table. (A table could span multiple different HDFS directories if it is partitioned. The directories could be widely scattered because a partition can reside in an arbitrary HDFS directory based on its LOCATION attribute.)

Cloudera Manager considerations:

A change in the behavior of metadata loading in Impala 2.9 could lead to certain long-running statements being left out of the Cloudera Manager list of Impala queries until the statements are completed.

Prior to Impala 2.9, statements such as DESCRIBE could cause the Impala web UI, and Cloudera Manager monitoring pages that rely on information from the web UI, to become unresponsive while they ran. The first time Impala references a large table, for example one with thousands of partitions, the statement might take longer than normal while the metadata for the table is loaded. In Impala 2.9 and higher, the Impala web UI and associated Cloudera Manager monitoring pages are more responsive while a metadata loading operation is in progress.

Although the statement that loads the metadata shows up on the Impala web UI /queries page immediately, it does not show up in the Cloudera Manager list of queries until the metadata is finished loading. For example, the first DESC RIBE of a large partitioned table might take 30 minutes due to metadata loading, and the statement does not show up in Cloudera Manager during those 30 minutes.

Kudu considerations:

The information displayed for Kudu tables includes the additional attributes that are only applicable for Kudu tables:

- Whether or not the column is part of the primary key. Every Kudu table has a true value here for at least one column. There could be multiple true values, for tables with composite primary keys.
- Whether or not the column is nullable. Specified by the NULL or NOT NULL attributes on the CREATE TABLE statement. Columns that are part of the primary key are automatically non-nullable.
- The default value, if any, for the column. Specified by the DEFAULT attribute on the CREATE TABLE statement. If the default value is NULL, that is not indicated in this column. It is implied by nullable being true and no other default value specified.
- The encoding used for values in the column. Specified by the ENCODING attribute on the CREATE TABLE statement.
- The compression used for values in the column. Specified by the COMPRESSION attribute on the CREATE T ABLE statement.
- The block size (in bytes) used for the underlying Kudu storage layer for the column. Specified by the BLOCK_SI ZE attribute on the CREATE TABLE statement.

The following example shows DESCRIBE output for a simple Kudu table, with a single-column primary key and all column attributes left with their default values:

```
describe million_rows;
+------+--------+---------+-------------+----------+---------------+--------
-------+---------------------+------------+
| name | type   | comment | primary_key | nullable | default_value | encodin
g      | compression         | block_size |
+------+--------+---------+-------------+----------+---------------+--------
-------+---------------------+------------+
| id   | string |         | true        | false    |               | AUTO_EN
CODING | DEFAULT_COMPRESSION | 0          |
| s    | string |         | false       | false    |               | AUTO_EN
CODING | DEFAULT_COMPRESSION | 0          |
+------+--------+---------+-------------+----------+---------------+--------
-------+---------------------+------------+
```

The following example shows DESCRIBE output for a Kudu table with a two-column primary key, and Kudu-specific attributes applied to some columns:

```
create table kudu_describe_example
(
  c1 int, c2 int,
  c3 string, c4 string not null, c5 string default 'n/a', c6 string default
'',
  c7 bigint not null, c8 bigint null default null, c9 bigint default -1 enco
ding bit_shuffle,
  primary key(c1,c2)
)
partition by hash (c1, c2) partitions 10 stored as kudu;

describe kudu_describe_example;
+------+--------+---------+-------------+----------+---------------+-----
----------+---------------------+------------+
| name | type   | comment | primary_key | nullable | default_value | enco
ding      | compression         | block_size |
+------+--------+---------+-------------+----------+---------------+-----
----------+---------------------+------------+
| c1   | int    |         | true        | false    |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c2   | int    |         | true        | false    |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c3   | string |         | false       | true     |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c4   | string |         | false       | false    |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c5   | string |         | false       | true     | n/a           | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c6   | string |         | false       | true     |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c7   | bigint |         | false       | false    |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c8   | bigint |         | false       | true     |               | AUTO
_ENCODING | DEFAULT_COMPRESSION | 0          |
| c9   | bigint |         | false       | true     | -1            | BIT_
SHUFFLE   | DEFAULT_COMPRESSION | 0          |
+------+--------+---------+-------------+----------+---------------+-----
----------+---------------------+------------+
```

Related Information

SHOW statement

DROP DATABASE statement

The DROP DATABASE statement removes a database from the system. The physical operations involve removing the metadata for the database from the metastore, and deleting the corresponding *.db directory from HDFS.

Syntax:

```
DROP (DATABASE|SCHEMA) [IF EXISTS] DATABASE_NAME [RESTRICT | CASCADE];
```

Statement type: DDL

Usage notes:

By default, the database must be empty before it can be dropped, to avoid losing any data.

In Impala 2.3 and higher, you can include the CASCADE clause to make Impala drop all tables and other objects in the database before dropping the database itself. The RESTRICT clause enforces the original requirement that the database be empty before being dropped. Because the RESTRICT behavior is still the default, this clause is optional.

The automatic dropping resulting from the CASCADE clause follows the same rules as the corresponding DROP TABLE, DROP VIEW, and DROP FUNCTION statements. In particular, the HDFS directories and data files for any external tables are left behind when the tables are removed.

When you do not use the CASCADE clause, drop or move all the objects inside the database manually before dropping the database itself:

- Use the SHOW TABLES statement to locate all tables and views in the database, and issue DROP TABLE and DROP VIEW statements to remove them all.
- Use the SHOW FUNCTIONS and SHOW AGGREGATE FUNCTIONS statements to locate all user-defined functions in the database, and issue DROP FUNCTION and DROP AGGREGATE FUNCTION statements to remove them all.
- To keep tables or views contained by a database while removing the database itself, use ALTER TABLE and ALTER VIEW to move the relevant objects to a different database before dropping the original database.

You cannot drop the current database, that is, the database your session connected to either through the USE statement or the -d option of `impala-shell`. Issue a USE statement to switch to a different database first. Because the default database is always available, issuing USE default is a convenient way to leave the current database before dropping it.

Hive considerations:

When you drop a database in Impala, the database can no longer be used by Hive.

Examples:

See *CREATE DATABASE* statement for examples covering CREATE DATABASE, USE, and DROP DATABASE.

Amazon S3 considerations:

In Impala 2.6 and higher, Impala DDL statements such as CREATE DATABASE, CREATE TABLE, DROP DATABASE CASCADE, DROP TABLE, and ALTER TABLE [ADD|DROP] PARTITION can create or remove folders as needed in the Amazon S3 system. Prior to Impala 2.6, you had to create folders yourself and point Impala database, tables, or partitions at them, and manually remove folders when no longer needed.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have write permission for the directory associated with the database.

Examples:

```
create database first_db;
use first_db;
create table t1 (x int);

create database second_db;
use second_db;
-- Each database has its own namespace for tables.
-- You can reuse the same table names in each database.
create table t1 (s string);

create database temp;

-- You can either USE a database after creating it,
-- or qualify all references to the table name with the name of the datab
ase.
-- Here, tables T2 and T3 are both created in the TEMP database.

create table temp.t2 (x int, y int);
use database temp;
create table t3 (s string);
```

```
-- You cannot drop a database while it is selected by the USE statement.
drop database temp;
ERROR: AnalysisException: Cannot drop current default database: temp

-- The always-available database 'default' is a convenient one to USE
-- before dropping a database you created.
use default;
-- Before dropping a database, first drop all the tables inside it,
-- or in Impala 2.3 and higher use the CASCADE clause.
drop database temp;
ERROR: ImpalaRuntimeException: Error making 'dropDatabase' RPC to Hive Meta
store:
CAUSED BY: InvalidOperationException: Database temp is not empty
show tables in temp;
+------+
| name |
+------+
| t3   |
+------+

-- Impala 2.3 and higher:
drop database temp cascade;

-- Earlier releases:
drop table temp.t3;
drop database temp;
```

Related Information

CREATE DATABASE statement

Impala with Amazon S3

DROP FUNCTION statement

The DROP FUNCTION statement removes a user-defined function (UDF), so that it is not available for execution during Impala SELECT or INSERT operations.

Syntax:

To drop C++ UDFs and UDAs:

```
DROP [AGGREGATE] FUNCTION [IF EXISTS]
 [DB_NAME.]FUNCTION_NAME(TYPE[, TYPE...])
```

Note:

The preceding syntax, which includes the function signature, also applies to Java UDFs that were created using the corresponding CREATE FUNCTION syntax that includes the argument and return types. After upgrading to Impala 2.5 or higher, consider re-creating all Java UDFs with the CREATE FUNCTION syntax that does not include the function signature. Java UDFs created this way are now persisted in the metastore database and do not need to be re-created after an Impala restart.

To drop Java UDFs (created using the CREATE FUNCTION syntax with no function signature):

```
DROP FUNCTION [IF EXISTS] [DB_NAME.]FUNCTION_NAME
```

Statement type: DDL

Usage notes:

Because the same function name could be overloaded with different argument signatures, you specify the argument types to identify the exact function to drop.

Restrictions:

In Impala 2.5 and higher, Impala UDFs and UDAs written in C++ are persisted in the metastore database. Java UDFs are also persisted, if they were created with the new CREATE FUNCTION syntax for Java UDFs, where the Java function argument and return types are omitted. Java-based UDFs created with the old CREATE FUNCTION syntax do not persist across restarts because they are held in the memory of the `catalogd` daemon. Until you re-create such Java UDFs using the new CREATE FUNCTION syntax, you must reload those Java-based UDFs by running the original CREATE FUNCTION statements again each time you restart the `catalogd` daemon. Prior to Impala 2.5 the requirement to reload functions after a restart applied to both C++ and Java functions.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, does not need any particular HDFS permissions to perform this statement. All read and write operations are on the metastore database, not HDFS files and directories.

Examples:

The following example shows how to drop Java functions created with the signatureless CREATE FUNCTION syntax in Impala 2.5 and higher. Issuing DROP FUNCTION *FUNCTION_NAME* removes all the overloaded functions under that name. See the *CREATE FUNCTION* statement for a longer example showing how to set up such functions in the first place.

```
create function my_func location '/user/impala/udfs/udf-examples-cdh570.jar'
  symbol='com.cloudera.impala.TestUdf';

show functions;
+-------------+-------------------------------------+-------------+----
-----------+
| return type | signature                           | binary type | is pe
rsistent |
+-------------+-------------------------------------+-------------+----
-----------+
| BIGINT      | my_func(BIGINT)                     | JAVA        | true
        |
| BOOLEAN     | my_func(BOOLEAN)                    | JAVA        | true
         |
| BOOLEAN     | my_func(BOOLEAN, BOOLEAN)           | JAVA        | true
        |
...
| BIGINT      | testudf(BIGINT)                     | JAVA        | true
         |
| BOOLEAN     | testudf(BOOLEAN)                    | JAVA        | true
        |
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN)           | JAVA        | true
         |
...

drop function my_func;
show functions;
+-------------+-------------------------------------+-------------+------
---------+
| return type | signature                           | binary type | is
persistent |
+-------------+-------------------------------------+-------------+------
---------+
| BIGINT      | testudf(BIGINT)                     | JAVA        | true
         |
| BOOLEAN     | testudf(BOOLEAN)                    | JAVA        | true
        |
```

```
| BOOLEAN     | testudf(BOOLEAN, BOOLEAN)             | JAVA        | true
         |
...
```

Related Information

CREATE FUNCTION statement

DROP ROLE statement

The DROP ROLE statement removes a role from the metastore database. Once dropped, the role is revoked for all groups it was previously assigned. Queries that are already executing are not affected. Impala verifies the role information approximately every 60 seconds, so the effects of DROP ROLE might not take effect for new Impala queries for a brief period.

Syntax:

```
DROP ROLE ROLE_NAME
```

Note: Before dropping a role in Ranger, you must remove all the privileges granted to the role in advance.

Required privileges:

Only administrative users can use this statement.

Compatibility:

Impala makes use of any roles and privileges specified by the GRANT and REVOKE statements in Hive, and Hive makes use of any roles and privileges specified by the GRANT and REVOKE statements in Impala. The Impala GRANT and REVOKE statements for privileges do not require the ROLE keyword to be repeated before each role name, unlike the equivalent Hive statements.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

DROP STATS statement

The DROP STATS statement removes the specified statistics from a table or partition. The statistics were originally created by the COMPUTE STATS or COMPUTE INCREMENTAL STATS statement.

Syntax:

```
DROP STATS [DATABASE_NAME.]TABLE_NAME
DROP INCREMENTAL STATS [DATABASE_NAME.]TABLE_NAME PARTITION (PARTITION_SPEC)

PARTITION_SPEC ::= PARTITION_COL=CONSTANT_VALUE
```

The PARTITION clause is only allowed in combination with the INCREMENTAL clause. It is optional for COMPUTE INCREMENTAL STATS, and required for DROP INCREMENTAL STATS. Whenever you specify partitions through the PARTITION (*PARTITION_SPEC*) clause in a COMPUTE INCREMENTAL STATS or DROP INCREMENTAL STATS statement, you must include all the partitioning columns in the specification, and specify constant values for all the partition key columns.

DROP STATS removes all statistics from the table, whether created by COMPUTE STATS or COMPUTE INCREMENTAL STATS.

DROP INCREMENTAL STATS only affects incremental statistics for a single partition, specified through the PART ITION clause. The incremental stats are marked as outdated, so that they are recomputed by the next COMPUTE INCREMENTAL STATS statement.

Usage notes:

You typically use this statement when the statistics for a table or a partition have become stale due to data files being added to or removed from the associated HDFS data directories, whether by manual HDFS operations or INSERT, INSERT OVERWRITE, or LOAD DATA statements, or adding or dropping partitions.

When a table or partition has no associated statistics, Impala treats it as essentially zero-sized when constructing the execution plan for a query. In particular, the statistics influence the order in which tables are joined in a join query. To ensure proper query planning and good query performance and scalability, make sure to run COMPUTE STATS or COMPUTE INCREMENTAL STATS on the table or partition after removing any stale statistics.

Dropping the statistics is not required for an unpartitioned table or a partitioned table covered by the original type of statistics. A subsequent COMPUTE STATS statement replaces any existing statistics with new ones, for all partitions, regardless of whether the old ones were outdated. Therefore, this statement was rarely used before the introduction of incremental statistics.

Dropping the statistics is required for a partitioned table containing incremental statistics, to make a subsequent COMPUTE INCREMENTAL STATS statement rescan an existing partition. See the *Tables and columns topic* for information about incremental statistics, a new feature available in Impala 2.1.0 and higher.

Statement type: DDL

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, does not need any particular HDFS permissions to perform this statement. All read and write operations are on the metastore database, not HDFS files and directories.

Examples:

The following example shows a partitioned table that has associated statistics produced by the COMPUTE INCR EMENTAL STATS statement, and how the situation evolves as statistics are dropped from specific partitions, then the entire table.

Initially, all table and column statistics are filled in.

```
show table stats item_partitioned;
+-------------+-------+--------+----------+--------------+---------+-----
------------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Increm
ental stats
+-------------+-------+--------+----------+--------------+---------+--------
---------
| Books       | 1733  | 1      | 223.74KB | NOT CACHED   | PARQUET | true
| Children    | 1786  | 1      | 230.05KB | NOT CACHED   | PARQUET | true
| Electronics | 1812  | 1      | 232.67KB | NOT CACHED   | PARQUET | true
| Home        | 1807  | 1      | 232.56KB | NOT CACHED   | PARQUET | true
| Jewelry     | 1740  | 1      | 223.72KB | NOT CACHED   | PARQUET | true
| Men         | 1811  | 1      | 231.25KB | NOT CACHED   | PARQUET | true
| Music       | 1860  | 1      | 237.90KB | NOT CACHED   | PARQUET | true
| Shoes       | 1835  | 1      | 234.90KB | NOT CACHED   | PARQUET | true
| Sports      | 1783  | 1      | 227.97KB | NOT CACHED   | PARQUET | true
| Women       | 1790  | 1      | 226.27KB | NOT CACHED   | PARQUET | true
| Total       | 17957 | 10     | 2.25MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+-----
------------
show column stats item_partitioned;
+------------------+-----------+------------------+--------+----------+----
----------
```

```
| Column            | Type      | #Distinct Values | #Nulls | Max Size | Avg
 Size
+------------------+-----------+------------------+--------+----------+-----
---------
| i_item_sk         | INT       | 19443            | -1     | 4        | 4
| i_item_id         | STRING    | 9025             | -1     | 16       | 16
| i_rec_start_date | TIMESTAMP | 4                | -1     | 16       | 16
| i_rec_end_date   | TIMESTAMP | 3                | -1     | 16       | 16
| i_item_desc       | STRING    | 13330            | -1     | 200      | 100
.302803039
| i_current_price  | FLOAT     | 2807             | -1     | 4        | 4
| i_wholesale_cost | FLOAT     | 2105             | -1     | 4        | 4
| i_brand_id        | INT       | 965              | -1     | 4        | 4
| i_brand           | STRING    | 725              | -1     | 22       | 16.
1776008605
| i_class_id        | INT       | 16               | -1     | 4        | 4
| i_class           | STRING    | 101              | -1     | 15       | 7.7
6749992370
| i_category_id     | INT       | 10               | -1     | 4        | 4
| i_manufact_id     | INT       | 1857             | -1     | 4        | 4
| i_manufact        | STRING    | 1028             | -1     | 15       | 1
1.3295001983
| i_size            | STRING    | 8                | -1     | 11       | 4.3
3459997177
| i_formulation     | STRING    | 12884            | -1     | 20       | 1
9.9799995422
| i_color           | STRING    | 92               | -1     | 10       | 5.3
8089990615
| i_units           | STRING    | 22               | -1     | 7        | 4
.18690013885
| i_container       | STRING    | 2                | -1     | 7        | 6.9
9259996414
| i_manager_id      | INT       | 105              | -1     | 4        | 4
| i_product_name    | STRING    | 19094            | -1     | 25       | 18.
0233001708
| i_category        | STRING    | 10               | 0      | -1       | -1
+------------------+-----------+------------------+--------+----------+---
-----------
```

To remove statistics for particular partitions, use the DROP INCREMENTAL STATS statement. After removing statistics for two partitions, the table-level statistics reflect that change in the #Rows and Incremental stats fields. The counts, maximums, and averages of the column-level statistics are unaffected.

Note: (It is possible that the row count might be preserved in future after a DROP INCREMENTAL STATS statement. Check the resolution of the issue IMPALA-1615.)

```
drop incremental stats item_partitioned partition (i_category='Sports');
drop incremental stats item_partitioned partition (i_category='Electronics'
);

show table stats item_partitioned
+-------------+-------+--------+----------+--------------+---------+------
------------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Increm
ental stats
+-------------+-------+--------+----------+--------------+---------+--------
---------
| Books       | 1733  | 1      | 223.74KB | NOT CACHED   | PARQUET | true
| Children    | 1786  | 1      | 230.05KB | NOT CACHED   | PARQUET | true
| Electronics | -1    | 1      | 232.67KB | NOT CACHED   | PARQUET | false
| Home        | 1807  | 1      | 232.56KB | NOT CACHED   | PARQUET | true
| Jewelry     | 1740  | 1      | 223.72KB | NOT CACHED   | PARQUET | true
```

```
| Men          | 1811  | 1      | 231.25KB | NOT CACHED   | PARQUET | true
| Music        | 1860  | 1      | 237.90KB | NOT CACHED   | PARQUET | true
| Shoes        | 1835  | 1      | 234.90KB | NOT CACHED   | PARQUET | true
| Sports       | -1    | 1      | 227.97KB | NOT CACHED   | PARQUET | false
| Women        | 1790  | 1      | 226.27KB | NOT CACHED   | PARQUET | true
| Total        | 17957 | 10     | 2.25MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+-------
----------
show column stats item_partitioned
+------------------+-----------+------------------+--------+----------+---
-----------
| Column           | Type      | #Distinct Values | #Nulls | Max Size | Avg
Size
+------------------+-----------+------------------+--------+----------+----
----------
| i_item_sk        | INT       | 19443            | -1     | 4        | 4
| i_item_id        | STRING    | 9025             | -1     | 16       | 16
| i_rec_start_date | TIMESTAMP | 4                | -1     | 16       | 16
| i_rec_end_date   | TIMESTAMP | 3                | -1     | 16       | 16
| i_item_desc      | STRING    | 13330            | -1     | 200      | 10
0.302803039
| i_current_price  | FLOAT     | 2807             | -1     | 4        | 4
| i_wholesale_cost | FLOAT     | 2105             | -1     | 4        | 4
| i_brand_id       | INT       | 965              | -1     | 4        | 4
| i_brand          | STRING    | 725              | -1     | 22       | 16
.1776008605
| i_class_id       | INT       | 16               | -1     | 4        | 4
| i_class          | STRING    | 101              | -1     | 15       | 7.
76749992370
| i_category_id    | INT       | 10               | -1     | 4        | 4
| i_manufact_id    | INT       | 1857             | -1     | 4        | 4
| i_manufact       | STRING    | 1028             | -1     | 15       | 11.3
295001983
| i_size           | STRING    | 8                | -1     | 11       | 4.
33459997177
| i_formulation    | STRING    | 12884            | -1     | 20       | 19.9
799995422
| i_color          | STRING    | 92               | -1     | 10       | 5.
38089990615
| i_units          | STRING    | 22               | -1     | 7        | 4.18
690013885
| i_container      | STRING    | 2                | -1     | 7        | 6.
99259996414
| i_manager_id     | INT       | 105              | -1     | 4        | 4
| i_product_name   | STRING    | 19094            | -1     | 25       | 18
.0233001708
| i_category       | STRING    | 10               | 0      | -1       | -1
+------------------+-----------+------------------+--------+----------+--
------------
```

To remove all statistics from the table, whether produced by COMPUTE STATS or COMPUTE INCREMENTAL STATS, use the DROP STATS statement without the INCREMENTAL clause). Now, both table-level and column-level statistics are reset.

```
drop stats item_partitioned;

show table stats item_partitioned
+-------------+-------+--------+----------+--------------+---------+--------
----------
| i_category  | #Rows | #Files | Size     | Bytes Cached | Format  | Incr
emental stats
+-------------+-------+--------+----------+--------------+---------+------
------------
```

```
| Books        | -1    | 1      | 223.74KB | NOT CACHED   | PARQUET | false
| Children     | -1    | 1      | 230.05KB | NOT CACHED   | PARQUET | false
| Electronics  | -1    | 1      | 232.67KB | NOT CACHED   | PARQUET | false
| Home         | -1    | 1      | 232.56KB | NOT CACHED   | PARQUET | false
| Jewelry      | -1    | 1      | 223.72KB | NOT CACHED   | PARQUET | false
| Men          | -1    | 1      | 231.25KB | NOT CACHED   | PARQUET | false
| Music        | -1    | 1      | 237.90KB | NOT CACHED   | PARQUET | false
| Shoes        | -1    | 1      | 234.90KB | NOT CACHED   | PARQUET | false
| Sports       | -1    | 1      | 227.97KB | NOT CACHED   | PARQUET | false
| Women        | -1    | 1      | 226.27KB | NOT CACHED   | PARQUET | false
| Total        | -1    | 10     | 2.25MB   | 0B           |         |
+-------------+-------+--------+----------+--------------+---------+--------
----------
show column stats item_partitioned
+------------------+-----------+------------------+--------+----------+---
-------+
| Column           | Type      | #Distinct Values | #Nulls | Max Size | Avg
 Size |
+------------------+-----------+------------------+--------+----------+-----
-----+
| i_item_sk        | INT       | -1               | -1     | 4        | 4
     |
| i_item_id        | STRING    | -1               | -1     | -1       | -1
     |
| i_rec_start_date | TIMESTAMP | -1               | -1     | 16       | 16
     |
| i_rec_end_date   | TIMESTAMP | -1               | -1     | 16       | 16
     |
| i_item_desc      | STRING    | -1               | -1     | -1       | -1
     |
| i_current_price  | FLOAT     | -1               | -1     | 4        | 4
     |
| i_wholesale_cost | FLOAT     | -1               | -1     | 4        | 4
     |
| i_brand_id       | INT       | -1               | -1     | 4        | 4
     |
| i_brand          | STRING    | -1               | -1     | -1       | -1
     |
| i_class_id       | INT       | -1               | -1     | 4        | 4
     |
| i_class          | STRING    | -1               | -1     | -1       | -1
     |
| i_category_id    | INT       | -1               | -1     | 4        | 4
     |
| i_manufact_id    | INT       | -1               | -1     | 4        | 4
     |
| i_manufact       | STRING    | -1               | -1     | -1       | -1
     |
| i_size           | STRING    | -1               | -1     | -1       | -1
     |
| i_formulation    | STRING    | -1               | -1     | -1       | -1
     |
| i_color          | STRING    | -1               | -1     | -1       | -1
     |
| i_units          | STRING    | -1               | -1     | -1       | -1
     |
| i_container      | STRING    | -1               | -1     | -1       | -1
     |
| i_manager_id     | INT       | -1               | -1     | 4        | 4
     |
| i_product_name   | STRING    | -1               | -1     | -1       | -1
     |
| i_category       | STRING    | 10               | 0      | -1       | -1
     |
```

```
+------------------+-----------+------------------+--------+----------+----
------+
```

Related Information

Table and column statistics

DROP TABLE statement

The DROP TABLE statement removes an Impala table. Also removes the underlying HDFS data files for internal tables, although not for external tables.

Syntax:

```
DROP TABLE [IF EXISTS] [DB_NAME.]TABLE_NAME [PURGE]
```

IF EXISTS clause:

The optional IF EXISTS clause makes the statement succeed whether or not the table exists. If the table does exist, it is dropped; if it does not exist, the statement has no effect. This capability is useful in standardized setup scripts that remove existing schema objects and create new ones. By using some combination of IF EXISTS for the DROP statements and IF NOT EXISTS clauses for the CREATE statements, the script can run successfully the first time you run it (when the objects do not exist yet) and subsequent times (when some or all of the objects do already exist).

PURGE clause:

The optional PURGE keyword, available in Impala 2.3 and higher, causes Impala to remove the associated HDFS data files immediately, rather than going through the HDFS trashcan mechanism. Use this keyword when dropping a table if it is crucial to remove the data as quickly as possible to free up space, or if there is a problem with the trashcan, such as the trash cannot being configured or being in a different HDFS encryption zone than the data files.

Statement type: DDL

Usage notes:

By default, Impala removes the associated HDFS directory and data files for the table. If you issue a DROP TABLE and the data files are not deleted, it might be for the following reasons:

- If the table was created with the EXTERNAL clause, Impala leaves all files and directories untouched. Use external tables when the data is under the control of other Hadoop components, and Impala is only used to query the data files from their original locations.
- Impala might leave the data files behind unintentionally, if there is no HDFS location available to hold the HDFS trashcan for the impala user.

Make sure that you are in the correct database before dropping a table, either by issuing a USE statement first or by using a fully qualified name *DB_NAME.TABLE_NAME*.

If you intend to issue a DROP DATABASE statement, first issue DROP TABLE statements to remove all the tables in that database.

Examples:

```
create database temporary;
use temporary;
create table unimportant (x int);
create table trivial (s string);
-- Drop a table in the current database.
drop table unimportant;
-- Switch to a different database.
use default;
-- To drop a table in a different database...
drop table trivial;
ERROR: AnalysisException: Table does not exist: default.trivial
```

```
-- ...use a fully qualified name.
drop table temporary.trivial;
```

Amazon S3 considerations:

The DROP TABLE statement can remove data files from S3 if the associated S3 table is an internal table. In Impala 2.6 and higher, as part of improved support for writing to S3, Impala also removes the associated folder when dropping an internal table that resides on S3.

For compatibility with the S3 write support in Impala, follow these steps for querying table through Impala:

1. Use native Hadoop techniques, such as hadoop fs -cp or INSERT in Impala or Hive to create data files in S3.
2. Use the PURGE clause with DROP TABLE when dropping internal (managed) tables.

By default, when you do not include the PURGE clause in the statement, the data files are moved to the S3A trashcan. This operation is expensive. When you do include the PURGE clause, the data files are deleted immediately, skipping the expensive S3A trashcan operation.

In Impala 2.6 and higher, Impala DDL statements such as CREATE DATABASE, CREATE TABLE, DROP DATABASE CASCADE, DROP TABLE, and ALTER TABLE [ADD|DROP] PARTITION can create or remove folders as needed in the Amazon S3 system. Prior to Impala 2.6, you had to create folders yourself and point Impala database, tables, or partitions at them, and manually remove folders when no longer needed.

Cancellation: Cannot be cancelled.

HDFS permissions:

For an internal table, the user ID that the `impalad` daemon runs under, typically the impala user, must have write permission for all the files and directories that make up the table.

For an external table, dropping the table only involves changes to metadata in the metastore database. Because Impala does not remove any HDFS files or directories when external tables are dropped, no particular permissions are needed for the associated HDFS files or directories.

Kudu considerations:

Kudu tables can be managed or external, the same as with HDFS-based tables. For a managed table, the underlying Kudu table and its data are removed by DROP TABLE. For an external table, the underlying Kudu table and its data remain after a DROP TABLE.

Related Information

CREATE DATABASE statement

Managing disk space for Impala data

DROP VIEW statement

The DROP VIEW statement removes the specified view, which was originally created by the CREATE VIEW statement. Because a view is purely a logical construct (an alias for a query) with no physical data behind it, DROP VIEW only involves changes to metadata in the metastore database, not any data files in HDFS.

Syntax:

```
DROP VIEW [IF EXISTS] [DB_NAME.]VIEW_NAME
```

Statement type: DDL

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Examples:

The following example creates a series of views and then drops them. These examples illustrate how views are associated with a particular database, and both the view definitions and the view names for CREATE VIEW and DROP VIEW can refer to a view in the current database or a fully qualified view name.

```
-- Create and drop a view in the current database.
CREATE VIEW few_rows_from_t1 AS SELECT * FROM t1 LIMIT 10;
DROP VIEW few_rows_from_t1;

-- Create and drop a view referencing a table in a different database.
CREATE VIEW table_from_other_db AS SELECT x FROM db1.foo WHERE x IS NOT N
ULL;
DROP VIEW table_from_other_db;
USE db1;
-- Create a view in a different database.
CREATE VIEW db2.v1 AS SELECT * FROM db2.foo;
-- Switch into the other database and drop the view.
USE db2;
DROP VIEW v1;

USE db1;
-- Create a view in a different database.
CREATE VIEW db2.v1 AS SELECT * FROM db2.foo;
-- Drop a view in the other database.
DROP VIEW db2.v1;
```

EXPLAIN statement

The EXPLAIN statement returns the execution plan for a statement, showing the low-level mechanisms that Impala will use to read the data, divide the work among nodes in the cluster, and transmit intermediate and final results across the network.

Use explain followed by a complete SELECT query. For example:

Syntax:

```
EXPLAIN { SELECT_QUERY | CTAS_STMT | INSERT_STMT }
```

The *SELECT_QUERY* is a SELECT statement, optionally prefixed by a WITH clause.

The *INSERT_STMT* is an INSERT statement that inserts into or overwrites an existing table. It can use either the INSERT ... SELECT or INSERT ... VALUES syntax.

The *CTAS_STMT* is a CREATE TABLE statement using the AS SELECT clause, typically abbreviated as a “CTAS” operation.

Usage notes:

You can interpret the output to judge whether the query is performing efficiently, and adjust the query and/or the schema if not. For example, you might change the tests in the WHERE clause, add hints to make join operations more efficient, introduce subqueries, change the order of tables in a join, add or change partitioning for a table, collect column statistics and/or table statistics in Hive, or any other performance tuning steps.

The EXPLAIN output reminds you if table or column statistics are missing from any table involved in the query. These statistics are important for optimizing queries involving large tables or multi-table joins. See COMPUTE STATS statement on page 126 for how to gather statistics, and Table and column statistics for how to use this information for query tuning.

Read the EXPLAIN plan from bottom to top:

- The last part of the plan shows the low-level details such as the expected amount of data that will be read, where you can judge the effectiveness of your partitioning strategy and estimate how long it will take to scan a table based on total data size and the size of the cluster.
- As you work your way up, next you see the operations that will be parallelized and performed on each Impala node.
- At the higher levels, you see how data flows when intermediate result sets are combined and transmitted from one node to another.
- The EXPLAIN_LEVEL query option lets you customize how much detail to show in the EXPLAIN plan depending on whether you are doing high-level or low-level tuning, dealing with logical or physical aspects of the query.

If you come from a traditional database background and are not familiar with data warehousing, keep in mind that Impala is optimized for full table scans across very large tables. The structure and distribution of this data is typically not suitable for the kind of indexing and single-row lookups that are common in OLTP environments. Seeing a query scan entirely through a large table is common, not necessarily an indication of an inefficient query. Of course, if you can reduce the volume of scanned data by orders of magnitude, for example by using a query that affects only certain partitions within a partitioned table, then you might be able to optimize a query so that it executes in seconds rather than minutes.

Extended EXPLAIN output:

For performance tuning of complex queries, and capacity planning (such as using the admission control and resource management features), you can enable more detailed and informative output for the EXPLAIN statement. In the `impala-shell` interpreter, issue the command SET EXPLAIN_LEVEL=*LEVEL*, where *LEVEL* is an integer from 0 to 3 or corresponding mnemonic values minimal, standard, extended, or verbose.

When extended EXPLAIN output is enabled, EXPLAIN statements print information about estimated memory requirements, minimum number of virtual cores, and so on.

Starting in Impala 3.2, if the EXPLAIN_LEVEL option is set to EXTENDED level or VERBOSE, the output contains the following additional information.

- The analyzed query, in the output header.

  The analyzed query may have been rewritten to include various optimizations and implicit casts. See the example below.
- The predicates in the plan output includes the same implicit casts and literals printed with a cast to show the type.

Examples:

This example shows how the standard EXPLAIN output moves from the lowest (physical) level to the higher (logical) levels. The query begins by scanning a certain amount of data; each node performs an aggregation operation (evaluating COUNT(*)) on some subset of data that is local to that node; the intermediate results are transmitted back to the coordinator node (labelled here as the EXCHANGE node); lastly, the intermediate results are summed to display the final result.

```
[impalad-host:21000] > explain select count(*) from customer_address;
+----------------------------------------------------------+
| Explain String                                           |
+----------------------------------------------------------+
| Estimated Per-Host Requirements: Memory=42.00MB VCores=1 |
|                                                          |
| 03:AGGREGATE [MERGE FINALIZE]                            |
| |  output: sum(count(*))                                 |
| |                                                        |
| 02:EXCHANGE [PARTITION=UNPARTITIONED]                    |
| |                                                        |
| 01:AGGREGATE                                             |
| |  output: count(*)                                      |
| |                                                        |
| 00:SCAN HDFS [default.customer_address]                  |
|    partitions=1/1 size=5.25MB                            |
```

```
+-----------------------------------------------------------+
```

The following example shows an extended EXPLAIN output. Note that the analyzed query was rewritten to include:

- The 'constant folding' optimization, which simplified the expression in the original query, '1000 / 100' to '10'.
- The implicit casts in the WHERE clause.

```
EXPLAIN SELECT * FROM functional_kudu.alltypestiny WHERE bigint_col < 1000 /
 100;
+-----------------------------------------------------------+
| Explain String                                            |
+-----------------------------------------------------------+
| ...
| Analyzed query: SELECT * FROM mytable WHERE CAST(bigint_col AS DOUBLE) <
 CAST(10 AS DOUBLE)
| ...
| 00:SCAN KUDU [functional_kudu.alltypestiny]
| predicates: CAST(bigint_col AS DOUBLE) < CAST(10 AS DOUBLE)
...
```

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all applicable directories in all source tables for the query that is being explained. (A SELECT operation could read files from multiple different HDFS directories if the source table is partitioned.)

Kudu considerations:

The EXPLAIN statement displays equivalent plan information for queries against Kudu tables as for queries against HDFS-based tables.

To see which predicates Impala can “push down” to Kudu for efficient evaluation, without transmitting unnecessary rows back to Impala, look for the kudu predicates item in the scan phase of the query. The label kudu predicates indicates a condition that can be evaluated efficiently on the Kudu side. The label predicates in a SCAN KUDU node indicates a condition that is evaluated by Impala. For example, in a table with primary key column X and non-primary key column Y, you can see that some operators in the WHERE clause are evaluated immediately by Kudu and others are evaluated later by Impala:

```
EXPLAIN SELECT x,y from kudu_table WHERE
  x = 1 AND y NOT IN (2,3) AND z = 1
  AND a IS NOT NULL AND b > 0 AND length(s) > 5;
+----------------
| Explain String
+----------------
...
| 00:SCAN KUDU [kudu_table]
|    predicates: y NOT IN (2, 3), length(s) > 5
|    kudu predicates: a IS NOT NULL, b > 0, x = 1, z = 1
```

Only binary predicates, IS NULL and IS NOT NULL (in Impala 2.9 and higher), and IN predicates containing literal values that exactly match the types in the Kudu table, and do not require any casting, can be pushed to Kudu.

Related Information

EXPLAIN_LEVEL query option

Table and column statistics
Understanding performance using Explain plan

GRANT statement

The GRANT statement grants a privilege on a specified object to a user or to a group.

Statement

Syntax:

```
     GRANT PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
     TO USER USER_NAME
     GRANT PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
     TO GROUP GROUP_NAME

     privilege ::= ALL | ALTER | CREATE | DROP | INSERT | REFRESH | SELECT
| SELECT(COLUMN_NAME)
     object_type ::= SERVER | URI | DATABASE | TABLE
```

Typically, the *OBJECT_NAME* is an identifier. For URIs, it is a string literal.

The ability to grant or revoke SELECT privilege on specific columns is available in Impala 2.3 and higher.

Usage notes:

You can only grant the ALL privilege to the URI object. Finer-grained privileges mentioned below on a URI are not supported.

The table below lists the minimum level of privileges and the scope required to execute SQL statements. The following notations are used:

- ANY denotes the SELECT, INSERT, CREATE, or REFRESH privilege.
- ALL privilege denotes the SELECT, INSERT, CREATE, and REFRESH privileges.
- The owner of an object effectively has the ALL privilege on the object.
- The parent levels of the specified scope are implicitly supported. For example, if a privilege is listed with the TABLE scope, the same privilege granted on DATABASE and SERVER will allow the user to execute that specific SQL statement on TABLE.

SQL Statement	Privileges	Scope
SELECT	SELECT	TABLE
WITH SELECT	SELECT	TABLE
EXPLAIN SELECT	SELECT	TABLE
INSERT	INSERT	TABLE
EXPLAIN INSERT	INSERT	TABLE
TRUNCATE	INSERT	TABLE
LOAD	INSERT	TABLE
	ALL	URI
CREATE DATABASE	CREATE	SERVER
CREATE DATABASE LOCATION	CREATE	SERVER
	ALL	URI
CREATE TABLE	CREATE	DATABASE

CREATE TABLE LIKE	CREATE	DATABASE
	SELECT, INSERT, or REFRESH	TABLE
CREATE TABLE AS SELECT	CREATE	DATABASE
	INSERT	DATABASE
	SELECT	TABLE
EXPLAIN CREATE TABLE AS SELECT	CREATE	DATABASE
	INSERT	DATABASE
	SELECT	TABLE
CREATE TABLE LOCATION	CREATE	TABLE
	ALL	URI
CREATE VIEW	CREATE	DATABASE
	SELECT	TABLE
ALTER DATABASE SET OWNER	ALL WITH GRANT	DATABASE
ALTER TABLE	ALL	TABLE
ALTER TABLE SET LOCATION	ALL	TABLE
	ALL	URI
ALTER TABLE RENAME	CREATE	DATABASE
	ALL	TABLE
ALTER TABLE SET OWNER	ALL WITH GRANT	TABLE
ALTER VIEW	ALL	TABLE
	SELECT	TABLE
ALTER VIEW RENAME	CREATE	DATABASE
	ALL	TABLE
ALTER VIEW SET OWNER	ALL WITH GRANT	VIEW
DROP DATABASE	ALL	DATABASE
DROP TABLE	ALL	TABLE
DROP VIEW	ALL	TABLE
CREATE FUNCTION	CREATE	DATABASE
	ALL	URI
DROP FUNCTION	ALL	DATABASE
COMPUTE STATS	ALL	TABLE
DROP STATS	ALL	TABLE
INVALIDATE METADATA	REFRESH	SERVER
INVALIDATE METADATA <table>	REFRESH	TABLE
REFRESH <table>	REFRESH	TABLE
REFRESH AUTHORIZATION	REFRESH	SERVER
REFRESH FUNCTIONS	REFRESH	DATABASE
COMMENT ON DATABASE	ALL	DATABASE
COMMENT ON TABLE	ALL	TABLE
COMMENT ON VIEW	ALL	TABLE

COMMENT ON COLUMN	ALL	TABLE
DESCRIBE DATABASE	SELECT, INSERT, or REFRESH	DATABASE
DESCRIBE <table/view>	SELECT, INSERT, or REFRESH	TABLE
If the user has the SELECT privilege at the COLUMN level, only the columns the user has access will show.	SELECT	COLUMN
USE	ANY	TABLE
SHOW DATABASES	ANY	TABLE
SHOW TABLES	ANY	TABLE
SHOW FUNCTIONS	SELECT, INSERT, or REFRESH	DATABASE
SHOW PARTITIONS	SELECT, INSERT, or REFRESH	TABLE
SHOW TABLE STATS	SELECT, INSERT, or REFRESH	TABLE
SHOW COLUMN STATS	SELECT, INSERT, or REFRESH	TABLE
SHOW FILES	SELECT, INSERT, or REFRESH	TABLE
SHOW CREATE TABLE	SELECT, INSERT, or REFRESH	TABLE
SHOW CREATE VIEW	SELECT, INSERT, or REFRESH	TABLE
SHOW CREATE FUNCTION	SELECT, INSERT, or REFRESH	DATABASE
SHOW RANGE PARTITIONS (Kudu only)	SELECT, INSERT, or REFRESH	TABLE
UPDATE (Kudu only)	ALL	TABLE
EXPLAIN UPDATE (Kudu only)	ALL	TABLE
UPSERT (Kudu only)	ALL	TABLE
WITH UPSERT (Kudu only)	ALL	TABLE
EXPLAIN UPSERT (Kudu only)	ALL	TABLE
DELETE (Kudu only)	ALL	TABLE
EXPLAIN DELETE (Kudu only)	ALL	TABLE

Compatibility:

- The Impala GRANT and REVOKE statements are available in Impala 2.0 and later.
- In Impala 1.4 and later, Impala can make use of any privileges specified by the GRANT and REVOKE statements in Hive.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

GRANT ROLE statement

The GRANT statement assigns a role to a group.

Syntax

```
GRANT ROLE ROLE_NAME TO GROUP GROUP_NAME
```

INSERT statement

Impala supports inserting into tables and partitions that you create with the Impala CREATE TABLE statement, or pre-defined tables and partitions created through Hive.

Syntax:

```
[WITH_CLAUSE]
  INSERT [HINT_CLAUSE] { INTO | OVERWRITE } [TABLE] TABLE_NAME
  [(COLUMN_LIST)]
  [ PARTITION (PARTITION_CLAUSE)]
{
    [HINT_CLAUSE] SELECT_STATEMENT
  | VALUES (VALUE [, VALUE ...]) [, (VALUE [, VALUE ...]) ...]
}

partition_clause ::= COL_NAME [= CONSTANT] [, COL_NAME [= CONSTANT] ...]

hint_clause ::=
  HINT_WITH_DASHES |
  HINT_WITH_CSTYLE_DELIMITERS |
  HINT_WITH_BRACKETS

hint_with_dashes ::= -- +SHUFFLE | -- +NOSHUFFLE -- +CLUSTERED

hint_with_cstyle_comments ::= /* +SHUFFLE */ | /* +NOSHUFFLE */ | /*
 +CLUSTERED */

hint_with_brackets ::= [SHUFFLE] | [NOSHUFFLE]
  (With this hint format, the square brackets are part of the syntax.)
```

Note: The square bracket style of hint is now deprecated and might be removed in a future release. For that reason, any newly added hints are not available with the square bracket syntax.

Appending or replacing (INTO and OVERWRITE clauses):

The INSERT INTO syntax appends data to a table. The existing data files are left as-is, and the inserted data is put into one or more new data files.

The INSERT OVERWRITE syntax replaces the data in a table. Currently, the overwritten data files are deleted immediately; they do not go through the HDFS trash mechanism.

Complex type considerations:

The INSERT statement currently does not support writing data files containing complex types (ARRAY, STRUCT, and MAP). To prepare Parquet data for such tables, you generate the data files outside Impala and then use LOAD DATA or CREATE EXTERNAL TABLE to associate those data files with the table. Currently, such tables must use the Parquet file format.

Kudu considerations:

Currently, the INSERT OVERWRITE syntax cannot be used with Kudu tables.

Kudu tables require a unique primary key for each row. If an INSERT statement attempts to insert a row with the same values for the primary key columns as an existing row, that row is discarded and the insert operation continues. When rows are discarded due to duplicate primary keys, the statement finishes with a warning, not an error. (This is a change from early releases of Kudu where the default was to return in error in such cases, and the syntax INSERT I GNORE was required to make the statement succeed. The IGNORE clause is no longer part of the INSERT syntax.)

For situations where you prefer to replace rows with duplicate primary key values, rather than discarding the new data, you can use the UPSERT statement instead of INSERT. UPSERT inserts rows that are entirely new, and for

rows that match an existing primary key in the table, the non-primary-key columns are updated to reflect the values in the “upserted” data.

If you really want to store new rows, not replace existing ones, but cannot do so because of the primary key uniqueness constraint, consider recreating the table with additional columns included in the primary key.

Usage notes:

Impala currently supports:

- Copy data from another table using SELECT query. In Impala 1.2.1 and higher, you can combine CREATE TABLE and INSERT operations into a single step with the CREATE TABLE AS SELECT syntax, which bypasses the actual INSERT keyword.
- An optional WITH clause before the INSERT keyword, to define a subquery referenced in the SELECT portion.
- Create one or more new rows using constant expressions through VALUES clause. (The VALUES clause was added in Impala 1.0.1.)
- By default, the first column of each newly inserted row goes into the first column of the table, the second column into the second column, and so on.

  You can also specify the columns to be inserted, an arbitrarily ordered subset of the columns in the destination table, by specifying a column list immediately after the name of the destination table. This feature lets you adjust the inserted columns to match the layout of a SELECT statement, rather than the other way around. (This feature was added in Impala 1.1.)

  The number of columns mentioned in the column list (known as the “column permutation”) must match the number of columns in the SELECT list or the VALUES tuples. The order of columns in the column permutation can be different than in the underlying table, and the columns of each input row are reordered to match. If the number of columns in the column permutation is less than in the destination table, all unmentioned columns are set to NULL.
- An optional hint clause immediately either before the SELECT keyword or after the INSERT keyword, to fine-tune the behavior when doing an INSERT ... SELECT operation into partitioned Parquet tables. The hint clause cannot be specified in multiple places. The hint keywords are [SHUFFLE] and [NOSHUFFLE], including the square brackets. Inserting into partitioned Parquet tables can be a resource-intensive operation because it potentially involves many files being written to HDFS simultaneously, and separate large memory buffers being allocated to buffer the data for each partition.

Note:

- Insert commands that partition or add files result in changes to Hive metadata. Because Impala uses Hive metadata, such changes may necessitate a metadata refresh. For more information, see the REFRESH function.
- Currently, Impala can only insert data into tables that use the text and Parquet formats. For other file formats, insert the data using Hive and use Impala to query it.
- As an alternative to the INSERT statement, if you have existing data files elsewhere in HDFS, the LOAD DATA statement can move those files into a table. This statement works with tables of any file format.

Statement type: DML (but still affected by the SYNC_DDL query option)

Usage notes:

When you insert the results of an expression, particularly of a built-in function call, into a small numeric column such as INT, SMALLINT, TINYINT, or FLOAT, you might need to use a CAST() expression to coerce values into the appropriate type. Impala does not automatically convert from a larger type to a smaller one. For example, to insert cosine values into a FLOAT column, write CAST(COS(angle) AS FLOAT) in the INSERT statement to make the conversion explicit.

File format considerations:

Because Impala can read certain file formats that it cannot write, the INSERT statement does not work for all kinds of Impala tables. See *How Impala works with Hadoop file formats* for details about what file formats are supported by the INSERT statement.

Any INSERT statement for a Parquet table requires enough free space in the HDFS filesystem to write one block. Because Parquet data files use a block size of 1 GB by default, an INSERT might fail (even for a very small amount of data) if your HDFS is running low on space.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMPUTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

The following example sets up new tables with the same definition as the TAB1 table using different file formats, and demonstrates inserting data into the tables created with the STORED AS TEXTFILE and STORED AS PARQUET clauses:

```
CREATE DATABASE IF NOT EXISTS file_formats;
USE file_formats;

DROP TABLE IF EXISTS text_table;
CREATE TABLE text_table
( id INT, col_1 BOOLEAN, col_2 DOUBLE, col_3 TIMESTAMP )
STORED AS TEXTFILE;

DROP TABLE IF EXISTS parquet_table;
CREATE TABLE parquet_table
( id INT, col_1 BOOLEAN, col_2 DOUBLE, col_3 TIMESTAMP )
STORED AS PARQUET;
```

With the INSERT INTO TABLE syntax, each new set of inserted rows is appended to any existing data in the table. This is how you would record small amounts of data that arrive continuously, or ingest new batches of data alongside the existing data. For example, after running 2 INSERT INTO TABLE statements with 5 rows each, the table contains 10 rows total:

```
[localhost:21000] > insert into table text_table select * from default.tab1;
Inserted 5 rows in 0.41s

[localhost:21000] > insert into table text_table select * from default.tab1;
Inserted 5 rows in 0.46s

[localhost:21000] > select count(*) from text_table;
+----------+
| count(*) |
+----------+
| 10       |
+----------+
Returned 1 row(s) in 0.26s
```

With the INSERT OVERWRITE TABLE syntax, each new set of inserted rows replaces any existing data in the table. This is how you load data to query in a data warehousing scenario where you analyze just the data for a particular day, quarter, and so on, discarding the previous data each time. You might keep the entire set of data in one raw table, and transfer and transform certain rows into a more compact and efficient form to perform intensive analysis on that subset.

For example, here we insert 5 rows into a table using the INSERT INTO clause, then replace the data by inserting 3 rows with the INSERT OVERWRITE clause. Afterward, the table only contains the 3 rows from the final INSERT statement.

```
[localhost:21000] > insert into table parquet_table select * from default.ta
b1;
Inserted 5 rows in 0.35s

[localhost:21000] > insert overwrite table parquet_table select * from def
ault.tab1 limit 3;
Inserted 3 rows in 0.43s
[localhost:21000] > select count(*) from parquet_table;
+----------+
| count(*) |
+----------+
| 3        |
+----------+
Returned 1 row(s) in 0.43s
```

The VALUES clause lets you insert one or more rows by specifying constant values for all the columns. The number, types, and order of the expressions must match the table definition.

Note: The INSERT ... VALUES technique is not suitable for loading large quantities of data into HDFS-based tables, because the insert operations cannot be parallelized, and each one produces a separate data file. Use it for setting up small dimension tables or tiny amounts of data for experimenting with SQL syntax, or with HBase tables. Do not use it for large ETL jobs or benchmark tests for load operations. Do not run scripts with thousands of INSERT ... VALUES statements that insert a single row each time. If you do run INSE RT ... VALUES operations to load data into a staging table as one stage in an ETL pipeline, include multiple row values if possible within each VALUES clause, and use a separate database to make cleanup easier if the operation does produce many tiny files.

The following example shows how to insert one row or multiple rows, with expressions of different types, using literal values, expressions, and function return values:

```
create table val_test_1 (c1 int, c2 float, c3 string, c4 boolean, c5 timesta
mp);
insert into val_test_1 values (100, 99.9/10, 'abc', true, now());
create table val_test_2 (id int, token string);
insert overwrite val_test_2 values (1, 'a'), (2, 'b'), (-1,'xyzzy');
```

These examples show the type of “not implemented” error that you see when attempting to insert data into a table with a file format that Impala currently does not write to:

```
DROP TABLE IF EXISTS sequence_table;
CREATE TABLE sequence_table
( id INT, col_1 BOOLEAN, col_2 DOUBLE, col_3 TIMESTAMP )
STORED AS SEQUENCEFILE;

DROP TABLE IF EXISTS rc_table;
CREATE TABLE rc_table
( id INT, col_1 BOOLEAN, col_2 DOUBLE, col_3 TIMESTAMP )
STORED AS RCFILE;

[localhost:21000] > insert into table rc_table select * from default.tab1;
Remote error
Backend 0:RC_FILE not implemented.

[localhost:21000] > insert into table sequence_table select * from default.t
ab1;
Remote error
Backend 0:SEQUENCE_FILE not implemented.
```

The following examples show how you can copy the data in all the columns from one table to another, copy the data from only some columns, or specify the columns in the select list in a different order than they actually appear in the table:

```
-- Start with 2 identical tables.
create table t1 (c1 int, c2 int);
create table t2 like t1;

-- If there is no () part after the destination table name,
-- all columns must be specified, either as * or by name.
insert into t2 select * from t1;
insert into t2 select c1, c2 from t1;

-- With the () notation following the destination table name,
-- you can omit columns (all values for that column are NULL
-- in the destination table), and/or reorder the values
-- selected from the source table. This is the "column permutation" feature.
insert into t2 (c1) select c1 from t1;
insert into t2 (c2, c1) select c1, c2 from t1;

-- The column names can be entirely different in the source and destination
 tables.
-- You can copy any columns, not just the corresponding ones, from the so
urce table.
-- But the number and type of selected columns must match the columns mentio
ned in the () part.
alter table t2 replace columns (x int, y int);
insert into t2 (y) select c1 from t1;
```

Sorting considerations: Although you can specify an ORDER BY clause in an INSERT ... SELECT statement, any ORDER BY clause is ignored and the results are not necessarily sorted. An INSERT ... SELECT operation potentially creates many different data files, prepared by different executor Impala daemons, and therefore the notion of the data being stored in sorted order is impractical.

Concurrency considerations: Each INSERT operation creates new data files with unique names, so you can run multiple INSERT INTO statements simultaneously without filename conflicts. While data is being inserted into an Impala table, the data is staged temporarily in a subdirectory inside the data directory; during this period, you cannot issue queries against that table in Hive. If an INSERT operation fails, the temporary data file and the subdirectory could be left behind in the data directory. If so, remove the relevant subdirectory and any data files it contains manually, by issuing an hdfs dfs -rm -r command, specifying the full path of the work subdirectory, whose name ends in _dir.

VALUES clause

The VALUES clause is a general-purpose way to specify the columns of one or more rows, typically within an INSERT statement.

Note: The INSERT ... VALUES technique is not suitable for loading large quantities of data into HDFS-based tables, because the insert operations cannot be parallelized, and each one produces a separate data file. Use it for setting up small dimension tables or tiny amounts of data for experimenting with SQL syntax, or with HBase tables. Do not use it for large ETL jobs or benchmark tests for load operations. Do not run scripts with thousands of INSERT ... VALUES statements that insert a single row each time. If you do run INSERT ... VALUES operations to load data into a staging table as one stage in an ETL pipeline, include multiple row values if possible within each VALUES clause, and use a separate database to make cleanup easier if the operation does produce many tiny files.

The following examples illustrate:

- How to insert a single row using a VALUES clause.
- How to insert multiple rows using a VALUES clause.

- How the row or rows from a VALUES clause can be appended to a table through INSERT INTO, or replace the contents of the table through INSERT OVERWRITE.
- How the entries in a VALUES clause can be literals, function results, or any other kind of expression. See Impala SQL literals on page 90 for the notation to use for literal values, especially for quoting and escaping conventions for strings. See Impala SQL operators on page 94 and Impala built-in functions on page 306 for other things you can include in expressions with the VALUES clause.

```
[localhost:21000] > describe val_example;
Query: describe val_example
Query finished, fetching results ...
+-------+---------+---------+
| name  | type    | comment |
+-------+---------+---------+
| id    | int     |         |
| col_1 | boolean |         |
| col_2 | double  |         |
+-------+---------+---------+

[localhost:21000] > insert into val_example values (1,true,100.0);
Inserted 1 rows in 0.30s
[localhost:21000] > select * from val_example;
+----+-------+-------+
| id | col_1 | col_2 |
+----+-------+-------+
| 1  | true  | 100   |
+----+-------+-------+
[localhost:21000] > insert overwrite val_example values (10,false,pow(2,5)),
 (50,true,10/3);
Inserted 2 rows in 0.16s
[localhost:21000] > select * from val_example;
+----+-------+-------------------+
| id | col_1 | col_2             |
+----+-------+-------------------+
| 10 | false | 32                |
| 50 | true  | 3.333333333333333 |
+----+-------+-------------------+
```

When used in an INSERT statement, the Impala VALUES clause can specify some or all of the columns in the destination table, and the columns can be specified in a different order than they actually appear in the table. To specify a different set or order of columns than in the table, use the syntax:

```
INSERT INTO DESTINATION
  (COL_X, COL_Y, COL_Z)
  VALUES
  (VAL_X, VAL_Y, VAL_Z);
```

Any columns in the table that are not listed in the INSERT statement are set to NULL.

HDFS considerations:

Impala physically writes all inserted files under the ownership of its default user, typically impala. Therefore, this user must have HDFS write permission in the corresponding table directory.

The permission requirement is independent of the authorization performed by the Ranger framework. (If the connected user is not authorized to insert into a table, Ranger blocks that operation immediately, regardless of the privileges available to the impala user.) Files created by Impala are not owned by and do not inherit permissions from the connected user.

The number of data files produced by an INSERT statement depends on the size of the cluster, the number of data blocks that are processed, the partition key columns in a partitioned table, and the mechanism Impala uses for dividing the work in parallel. Do not assume that an INSERT statement will produce some particular number of output files. In case of performance issues with data written by Impala, check that the output files do not suffer from

issues such as many tiny files or many tiny partitions. (In the Hadoop context, even files or partitions of a few tens of megabytes are considered “tiny”.)

The INSERT statement has always left behind a hidden work directory inside the data directory of the table. Formerly, this hidden work directory was named .impala_insert_staging . In Impala 2.0.1 and later, this directory name is changed to _impala_insert_staging . (While HDFS tools are expected to treat names beginning either with underscore and dot as hidden, in practice names beginning with an underscore are more widely supported.) If you have any scripts, cleanup jobs, and so on that rely on the name of this work directory, adjust them to use the new name.

HBase considerations:

You can use the INSERT statement with HBase tables as follows:

- You can insert a single row or a small set of rows into an HBase table with the INSERT ... VALUES syntax. This is a good use case for HBase tables with Impala, because HBase tables are not subject to the same kind of fragmentation from many small insert operations as HDFS tables are.
- You can insert any number of rows at once into an HBase table using the INSERT ... SELECT syntax.
- If more than one inserted row has the same value for the HBase key column, only the last inserted row with that value is visible to Impala queries. You can take advantage of this fact with INSERT ... VALUES statements to effectively update rows one at a time, by inserting new rows with the same key values as existing rows. Be aware that after an INSERT ... SELECT operation copying from an HDFS table, the HBase table might contain fewer rows than were inserted, if the key column in the source table contained duplicate values.
- You cannot INSERT OVERWRITE into an HBase table. New rows are always appended.
- When you create an Impala or Hive table that maps to an HBase table, the column order you specify with the INSERT statement might be different than the order you declare with the CREATE TABLE statement. Behind the scenes, HBase arranges the columns based on how they are divided into column families. This might cause a mismatch during insert operations, especially if you use the syntax INSERT INTO *HBASE_TABLE* SELECT * FROM *HDFS_TABLE*. Before inserting data, verify the column order by issuing a DESCRIBE statement for the table, and adjust the order of the select list in the INSERT statement.

Amazon S3 considerations:

In Impala 2.6 and higher, the Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in the Amazon Simple Storage Service (S3). The syntax of the DML statements is the same as for any other tables, because the S3 location for tables and partitions is specified by an s3a:// prefix in the LOCATION attribute of CREATE TABLE or ALTER TABLE statements. If you bring data into S3 using the normal S3 transfer mechanisms instead of Impala DML statements, issue a REFRESH statement for the table before using Impala to query the S3 data.

Because of differences between S3 and traditional filesystems, DML operations for S3 tables can take longer than for tables on HDFS. For example, both the LOAD DATA statement and the final stage of the INSERT and CREATE TABLE AS SELECT statements involve moving files from one directory to another. (In the case of INSERT and CREATE TABLE AS SELECT, the files are moved from a temporary staging directory to the final destination directory.) Because S3 does not support a “rename” operation for existing objects, in these cases Impala actually copies the data files from one location to another and then removes the original files. In Impala 2.6, the S3_SKIP_INSERT_STAGING query option provides a way to speed up INSERT statements for S3 tables and partitions, with the tradeoff that a problem during statement execution could leave data in an inconsistent state. It does not apply to INSERT OVERWRITE or LOAD DATA statements.

ADLS considerations:

In Impala 2.9 and higher, the Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in the Azure Data Lake Store (ADLS). ADLS Gen2 is supported in Impala 3.1 and higher.

In theCREATE TABLE or ALTER TABLE statements, specify the ADLS location for tables and partitions with the adl:// prefix for ADLS Gen1 and abfs:// or abfss:// for ADLS Gen2 in the LOCATION attribute.

If you bring data into ADLS using the normal ADLS transfer mechanisms instead of Impala DML statements, issue a REFRESH statement for the table before using Impala to query the ADLS data.

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Cancellation: Can be cancelled. To cancel this statement, use Ctrl-C from the `impala-shell` interpreter, the Cancel button from the Watch page in Hue, Actions > Cancel from the Queries list in Cloudera Manager, or Cancel from the list of in-flight queries (for a particular node) on the Queries tab in the Impala web UI (port 25000).

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read permission for the files in the source directory of an INSERT ... SELECT operation, and write permission for all affected directories in the destination table. (An INSERT operation could write files to multiple different HDFS directories if the destination table is partitioned.) This user must also have write permission to create a temporary work directory in the top-level HDFS directory of the destination table. An INSERT OVERWRITE operation does not require write permission on the original data files in the table, only on the table directories themselves.

Restrictions:

For INSERT operations into CHAR or VARCHAR columns, you must cast all STRING literals or expressions returning STRING to to a CHAR or VARCHAR type with the appropriate length.

Related startup options:

By default, if an INSERT statement creates any new subdirectories underneath a partitioned table, those subdirectories are assigned default HDFS permissions for the impala user. To make each subdirectory have the same permissions as its parent directory in HDFS, specify the ##insert_inherit_permissions startup option for the `impalad` daemon.

Inserting into partitioned tables with PARTITION clause

For a partitioned table, the optional PARTITION clause identifies which partition or partitions the values are inserted into.

All examples in this section will use the table declared as below:

```
CREATE TABLE t1 (w INT) PARTITIONED BY (x INT, y STRING);
```

Static partition inserts

In a static partition insert where a partition key column is given a constant value, such as PART ITION (year=2012, month=2), the rows are inserted with the same values specified for those partition key columns.

The number of columns in the SELECT list must equal the number of columns in the column permutation.

The PARTITION clause must be used for static partitioning inserts.

Example:

The following statement will insert the some_other_table.c1 values for the w column, and all the rows inserted will have the same x value of 10, and the same y value of ‘a’.

```
INSERT INTO t1 PARTITION (x=10, y='a')
            SELECT c1 FROM some_other_table;
```

Dynamic partition inserts

In a dynamic partition insert where a partition key column is in the INSERT statement but not assigned a value, such as in PARTITION (year, region)(both columns unassigned) or PARTITIO N(year, region='CA') (year column unassigned), the unassigned columns are filled in with the final

columns of the SELECT or VALUES clause. In this case, the number of columns in the SELECT list must equal the number of columns in the column permutation plus the number of partition key columns not assigned a constant value.

The following rules apply to dynamic partition inserts.

- The columns are bound in the order they appear in the INSERT statement.

  The table below shows the values inserted with the INSERT statements of different column orders.

	Column w Value	Column x Value	Column y Value
INSERT INTO t1 (w, x, y) VALUES (1, 2, 'c');	1	2	‘c’
INSERT INTO t1 (x,w) PARTITION (y) VALUES (1, 2, 'c');	2	1	‘c’

- When a partition clause is specified but the non-partition columns are not specified in the INSERT statement, as in the first example below, the non-partition columns are treated as though they had been specified before the PARTITION clause in the SQL.

  Example: These three statements are equivalent, inserting 1 to w, 2 to x, and ‘c’ to y columns.

```
INSERT INTO t1 PARTITION (x,y) VALUES (1, 2, ‘c’);
INSERT INTO t1 (w) PARTITION (x, y) VALUES (1, 2, ‘c’);
INSERT INTO t1 PARTITION (x, y='c') VALUES (1, 2);
```

- The PARTITION clause is not required for dynamic partition, but all the partition columns must be explicitly present in the INSERT statement in the column list or in the PARTITION clause. The partition columns cannot be defaulted to NULL.

  Example:

  The following statements are valid because the partition columns, x and y, are present in the INSERT statements, either in the PARTITION clause or in the column list.

```
INSERT INTO t1 PARTITION (x,y) VALUES (1, 2, ‘c’);
INSERT INTO t1 (w, x) PARTITION (y) VALUES (1, 2, ‘c’);
```

  The following statement is not valid for the partitioned table as defined above because the partition columns, x and y, are not present in the INSERT statement.

```
INSERT INTO t1 VALUES (1, 2, 'c');
```

- If partition columns do not exist in the source table, you can specify a specific value for that column in the PARTITION clause.

  Example: The source table only contains the column w and y. The value, 20, specified in the PARTITION clause, is inserted into the x column.

```
INSERT INTO t1 PARTITION (x=20, y) SELECT * FROM source;
```

TRUNCATE on Insert-Only Transactional Tables

The TRUNCATE table statement on Insert-Only Transactional tables creates new empty ACID base directories and does not remove the files. To maintain transactional isolation TRUNCATE statement will NOT delete the previous base and delta directories.

Related Information

Reading and writing ADLS data with Impala

Examples and performance characteristics of static and dynamic partitioned inserts

INVALIDATE METADATA statement

The INVALIDATE METADATA statement marks the metadata for one or all tables as stale. The next time the Impala service performs a query against a table whose metadata is invalidated, Impala reloads the associated metadata before the query proceeds. As this is a very expensive operation compared to the incremental metadata update done by the REFRESH statement, when possible, prefer REFRESH rather than INVALIDATE METADATA.

INVALIDATE METADATA is required when the following changes are made outside of Impala, in Hive and other Hive client, such as SparkSQL:

- Metadata of existing tables changes.
- New tables are added, and Impala will use the tables.
- The SERVER or DATABASE level privileges are changed from outside of Impala.
- Block metadata changes, but the files remain the same (HDFS rebalance).
- UDF jars change.
- Some tables are no longer queried, and you want to remove their metadata from the catalog and coordinator caches to reduce memory requirements.

No INVALIDATE METADATA is needed when the changes are made by impalad.

Once issued, the INVALIDATE METADATA statement cannot be cancelled.

Syntax:

```
INVALIDATE METADATA [[DB_NAME.]TABLE_NAME]
```

If there is no table specified, the cached metadata for all tables is flushed and synced with Hive Metastore (HMS). If tables were dropped from the HMS, they will be removed from the catalog, and if new tables were added, they will show up in the catalog.

If you specify a table name, only the metadata for that one table is flushed and synced with the HMS.

Usage notes:

To return accurate query results, Impala need to keep the metadata current for the databases and tables queried. Therefore, if some other entity modifies information used by Impala in the metastore, the information cached by Impala must be updated via INVALIDATE METADATA or REFRESH.

INVALIDATE METADATA and REFRESH are counterparts:

- INVALIDATE METADATA is an asynchronous operations that simply discards the loaded metadata from the catalog and coordinator caches. After that operation, the catalog and all the Impala coordinators only know about the existence of databases and tables and nothing more. Metadata loading for tables is triggered by any subsequent queries.
- REFRESH reloads the metadata synchronously. REFRESH is more lightweight than doing a full metadata load after a table has been invalidated. REFRESH cannot detect changes in block locations triggered by operations like HDFS balancer, hence causing remote reads during query execution with negative performance implications.

Use REFRESH after invalidating a specific table to separate the metadata load from the first query that's run against that table.

Examples:

This example illustrates creating a new database and new table in Hive, then doing an INVALIDATE METADATA statement in Impala using the fully qualified table name, after which both the new table and the new database are visible to Impala.

Before the INVALIDATE METADATA statement was issued, Impala would give a “not found” error if you tried to refer to those database or table names.

```
$ hive
hive> CREATE DATABASE new_db_from_hive;
hive> CREATE TABLE new_db_from_hive.new_table_from_hive (x INT);
hive> quit;
$ impala-shell
> REFRESH new_db_from_hive.new_table_from_hive;
ERROR: AnalysisException: Database does not exist: new_db_from_hive
> INVALIDATE METADATA new_db_from_hive.new_table_from_hive;

> SHOW DATABASES LIKE 'new*';
+--------------------+
| new_db_from_hive   |
+--------------------+

> SHOW TABLES IN new_db_from_hive;
+---------------------+
| new_table_from_hive |
+---------------------+
```

Use the REFRESH statement for incremental metadata update.

```
> REFRESH new_table_from_hive;
```

HDFS considerations:

By default, the INVALIDATE METADATA command checks HDFS permissions of the underlying data files and directories, caching this information so that a statement can be cancelled immediately if for example the impala user does not have permission to write to the data directory for the table. (This checking does not apply when the `catalogd` configuration option --load_catalog_in_background is set to false, which it is by default.) Impala reports any lack of write permissions as an INFO message in the log file.

If you change HDFS permissions to make data readable or writeable by the Impala user, issue another INVALIDATE METADATA to make Impala aware of the change.

Kudu considerations:

By default, much of the metadata for Kudu tables is handled by the underlying storage layer. Kudu tables have less reliance on the Metastore database, and require less metadata caching on the Impala side. For example, information about partitions in Kudu tables is managed by Kudu, and Impala does not cache any block locality metadata for Kudu tables. If the Kudu service is not integrated with the Hive Metastore, Impala will manage Kudu table metadata in the Hive Metastore.

The REFRESH and INVALIDATE METADATA statements are needed less frequently for Kudu tables than for HDFS-backed tables. Neither statement is needed when data is added to, removed, or updated in a Kudu table, even if the changes are made directly to Kudu through a client program using the Kudu API. Run REFRESH *TABLE_NAME* or INVALIDATE METADATA *TABLE_NAME* for a Kudu table only after making a change to the Kudu table schema, such as adding or dropping a column.

Related Information

On-demand Metadata

LOAD DATA statement

The LOAD DATA statement streamlines the ETL process for an internal Impala table by moving a data file or all the data files in a directory from an HDFS location into the Impala data directory for that table.

Syntax:

```
LOAD DATA INPATH 'HDFS_FILE_OR_DIRECTORY_PATH' [OVERWRITE] INTO T
ABLE TABLENAME
  [PARTITION (PARTCOL1=VAL1, PARTCOL2=VAL2 ...)]
```

When the LOAD DATA statement operates on a partitioned table, it always operates on one partition at a time. Specify the PARTITION clauses and list all the partition key columns, with a constant value specified for each.

Statement type: DML (but still affected by the SYNC_DDL query option)

Note:

- When the LOAD DATA statement operates on a partitioned table, ensure that the partition exists.
- You must be aware of the authorization behavior for the LOAD DATA command in Impala. When the LOAD DATA command is executed in Impala, the command checks if the user has ALL access to the source HDFS file/dir through a URL policy. This implies that the requesting user must have the ALL privilege on the source path in Ranger's policy repository of HADOOP SQL even if the requesting user is granted some privilege on the source path using Rangers' policy repository of HDFS or HDFS ACLs.

Usage notes:

- The loaded data files are moved, not copied, into the Impala data directory.
- You can specify the HDFS path of a single file to be moved, or the HDFS path of a directory to move all the files inside that directory. You cannot specify any sort of wildcard to take only some of the files from a directory. When loading a directory full of data files, keep all the data files at the top level, with no nested directories underneath.
- Currently, the Impala LOAD DATA statement only imports files from HDFS, not from the local filesystem. It does not support the LOCAL keyword of the Hive LOAD DATA statement. You must specify a path, not an hdfs :// URI.
- In the interest of speed, only limited error checking is done. If the loaded files have the wrong file format, different columns than the destination table, or other kind of mismatch, Impala does not raise any error for the LOAD DATA statement. Querying the table afterward could produce a runtime error or unexpected results. Currently, the only checking the LOAD DATA statement does is to avoid mixing together uncompressed and LZO-compressed text files in the same table.
- When you specify an HDFS directory name as the LOAD DATA argument, any hidden files in that directory (files whose names start with a .) are not moved to the Impala data directory.
- The operation fails if the source directory contains any non-hidden directories. Prior to Impala 2.5 if the source directory contained any subdirectory, even a hidden one such as _impala_insert_staging, the LOAD DATA statement would fail. In Impala 2.5 and higher, LOAD DATA ignores hidden subdirectories in the source directory, and only fails if any of the subdirectories are non-hidden.
- The loaded data files retain their original names in the new location, unless a name conflicts with an existing data file, in which case the name of the new file is modified slightly to be unique. (The name-mangling is a slight difference from the Hive LOAD DATA statement, which replaces identically named files.)
- By providing an easy way to transport files from known locations in HDFS into the Impala data directory structure, the LOAD DATA statement lets you avoid memorizing the locations and layout of HDFS directory tree containing the Impala databases and tables. (For a quick way to check the location of the data files for an Impala table, issue the statement DESCRIBE FORMATTED *TABLE_NAME*.)
- The PARTITION clause is especially convenient for ingesting new data for a partitioned table. As you receive new data for a time period, geographic region, or other division that corresponds to one or more partitioning columns, you can load that data straight into the appropriate Impala data directory, which might be nested several levels down if the table is partitioned by multiple columns. When the table is partitioned, you must specify constant values for all the partitioning columns.

Complex type considerations:

Because Impala currently cannot create Parquet data files containing complex types (ARRAY, STRUCT, and MAP), the LOAD DATA statement is especially important when working with tables containing complex type columns.

You create the Parquet data files outside Impala, then use either LOAD DATA, an external table, or HDFS-level file operations followed by REFRESH to associate the data files with the corresponding table.

If you connect to different Impala nodes within an `impala-shell` session for load-balancing purposes, you can enable the SYNC_DDL query option to make each DDL statement wait before returning, until the new or changed metadata has been received by all the Impala nodes.

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMPUTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

First, we use a trivial Python script to write different numbers of strings (one per line) into files stored in the doc_demo HDFS user account. (Substitute the path for your own HDFS user account when doing `hdfs dfs` operations like these.)

```
random_strings.py 1000 | hdfs dfs -put - /user/doc_demo/thousand_strings.txt
random_strings.py 100 | hdfs dfs -put - /user/doc_demo/hundred_strings.txt
random_strings.py 10 | hdfs dfs -put - /user/doc_demo/ten_strings.txt
```

Next, we create a table and load an initial set of data into it. Remember, unless you specify a STORED AS clause, Impala tables default to TEXTFILE format with Ctrl-A (hex 01) as the field delimiter. This example uses a single-column table, so the delimiter is not significant. For large-scale ETL jobs, you would typically use binary format data files such as Parquet or Avro, and load them into Impala tables that use the corresponding file format.

```
[localhost:21000] > create table t1 (s string);
[localhost:21000] > load data inpath '/user/doc_demo/thousand_strings.txt' i
nto table t1;
Query finished, fetching results ...
+----------------------------------------------------------+
| summary                                                  |
+----------------------------------------------------------+
| Loaded 1 file(s). Total files in destination location: 1 |
+----------------------------------------------------------+
Returned 1 row(s) in 0.61s
[kilo2-202-961.cs1cloud.internal:21000] > select count(*) from t1;
Query finished, fetching results ...
+------+
| _c0  |
+------+
| 1000 |
+------+
Returned 1 row(s) in 0.67s
[localhost:21000] > load data inpath '/user/doc_demo/thousand_strings.txt'
 into table t1;
ERROR: AnalysisException: INPATH location '/user/doc_demo/thousand_strings.
txt' does not exist.
```

As indicated by the message at the end of the previous example, the data file was moved from its original location. The following example illustrates how the data file was moved into the Impala data directory for the destination table, keeping its original filename:

```
$ hdfs dfs -ls /user/hive/warehouse/load_data_testing.db/t1
Found 1 items
-rw-r--r--   1 doc_demo doc_demo      13926 2013-06-26 15:40 /user/hive/wa
rehouse/load_data_testing.db/t1/thousand_strings.txt
```

The following example demonstrates the difference between the INTO TABLE and OVERWRITE TABLE clauses. The table already contains 1000 rows. After issuing the LOAD DATA statement with the INTO TABLE clause, the table contains 100 more rows, for a total of 1100. After issuing the LOAD DATA statement with the OVERWRITE INTO TABLE clause, the former contents are gone, and now the table only contains the 10 rows from the just-loaded data file.

```
[localhost:21000] > load data inpath '/user/doc_demo/hundred_strings.txt' in
to table t1;
Query finished, fetching results ...
+-----------------------------------------------------------+
| summary                                                   |
+-----------------------------------------------------------+
| Loaded 1 file(s). Total files in destination location: 2 |
+-----------------------------------------------------------+
Returned 1 row(s) in 0.24s
[localhost:21000] > select count(*) from t1;
Query finished, fetching results ...
+------+
| _c0  |
+------+
| 1100 |
+------+
Returned 1 row(s) in 0.55s
[localhost:21000] > load data inpath '/user/doc_demo/ten_strings.txt' overwr
ite into table t1;
Query finished, fetching results ...
+-----------------------------------------------------------+
| summary                                                   |
+-----------------------------------------------------------+
| Loaded 1 file(s). Total files in destination location: 1 |
+-----------------------------------------------------------+
Returned 1 row(s) in 0.26s
[localhost:21000] > select count(*) from t1;
Query finished, fetching results ...
+-----+
| _c0 |
+-----+
| 10  |
+-----+
Returned 1 row(s) in 0.62s
```

Amazon S3 considerations:

In Impala 2.6 and higher, the Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in the Amazon Simple Storage Service (S3). The syntax of the DML statements is the same as for any other tables, because the S3 location for tables and partitions is specified by an s3a:// prefix in the LOCATION attribute of CREATE TABLE or ALTER TABLE statements. If you bring data into S3 using the normal S3 transfer mechanisms instead of Impala DML statements, issue a REFRESH statement for the table before using Impala to query the S3 data.

Because of differences between S3 and traditional filesystems, DML operations for S3 tables can take longer than for tables on HDFS. For example, both the LOAD DATA statement and the final stage of the INSERT and CREATE TABLE AS SELECT statements involve moving files from one directory to another. (In the case of INSERT and CREATE TABLE AS SELECT, the files are moved from a temporary staging directory to the final destination directory.) Because S3 does not support a “rename” operation for existing objects, in these cases Impala actually copies the data files from one location to another and then removes the original files. In Impala 2.6, the S3_SKIP_INSERT_STAGING query option provides a way to speed up INSERT statements for S3 tables and partitions, with the tradeoff that a problem during statement execution could leave data in an inconsistent state. It does not apply to INSERT OVERWRITE or LOAD DATA statements.

ADLS considerations:

In Impala 2.9 and higher, the Impala DML statements (INSERT, LOAD DATA, and CREATE TABLE AS SELECT) can write data into a table or partition that resides in the Azure Data Lake Store (ADLS). ADLS Gen2 is supported in Impala 3.1 and higher.

In theCREATE TABLE or ALTER TABLE statements, specify the ADLS location for tables and partitions with the adl:// prefix for ADLS Gen1 and abfs:// or abfss:// for ADLS Gen2 in the LOCATION attribute.

If you bring data into ADLS using the normal ADLS transfer mechanisms instead of Impala DML statements, issue a REFRESH statement for the table before using Impala to query the ADLS data.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and write permissions for the files in the source directory, and write permission for the destination directory.

Kudu considerations:

The LOAD DATA statement cannot be used with Kudu tables.

HBase considerations:

The LOAD DATA statement cannot be used with HBase tables.

Related information:

The LOAD DATA statement is an alternative to the INSERT statement. Use LOAD DATA when you have the data files in HDFS but outside of any Impala table.

The LOAD DATA statement is also an alternative to the CREATE EXTERNAL TABLE statement. Use LOAD DATA when it is appropriate to move the data files under Impala control rather than querying them from their original location.

Related Information

Reading and writing ADLS data with Impala

REFRESH statement

The REFRESH statement reloads the metadata for the table from the metastore database and does an incremental reload of the file and block metadata from the HDFS NameNode. REFRESH is used to avoid inconsistencies between Impala and external metadata sources, namely Hive Metastore (HMS) and NameNodes.

The REFRESH statement is only required if you load data from outside of Impala. Updated metadata, as a result of running REFRESH, is broadcast to all Impala coordinators.

Syntax:

```
REFRESH [DB_NAME.]TABLE_NAME [PARTITION (KEY_COL1=VAL1 [, KEY_COL2=VAL2...]
)]
```

Usage notes:

The table name is a required parameter, and the table must already exist and be known to Impala.

Only the metadata for the specified table is reloaded.

Use the REFRESH statement to load the latest metastore metadata for a particular table after one of the following scenarios happens outside of Impala:

- Deleting, adding, or modifying files.

  For example, after loading new data files into the HDFS data directory for the table, appending to an existing HDFS file, inserting data from Hive via INSERT or LOAD DATA.

- Deleting, adding, or modifying partitions.

  For example, after issuing ALTER TABLE or other table-modifying SQL statement in Hive

  Note:

  In Impala 2.3 and higher, the ALTER TABLE *TABLE_NAME* RECOVER PARTITIONS statement maybe a faster alternative to REFRESH in some scenarios such as when you are only adding new partition directories through Hive or manual HDFS operations however the REFRESH query will NOT detect the new partitions unless the new partitions are in HMS already. You must be aware of the following limitations before you choose between REFRESH and ALTER TABLE.

  - REFRESH does not detect new partition directories that are added by HDFS.
  - RECOVER PARTITIONS does not detect changes in already existing partitions.
  - RECOVER PARTITIONS can cause performance issues if SYNC_DDL=1.

INVALIDATE METADATA and REFRESH are counterparts:

- INVALIDATE METADATA is an asynchronous operations that simply discards the loaded metadata from the catalog and coordinator caches. After that operation, the catalog and all the Impala coordinators only know about the existence of databases and tables and nothing more. Metadata loading for tables is triggered by any subsequent queries.
- REFRESH reloads the metadata synchronously. REFRESH is more lightweight than doing a full metadata load after a table has been invalidated. REFRESH cannot detect changes in block locations triggered by operations like HDFS balancer, hence causing remote reads during query execution with negative performance implications.

Refreshing a single partition:

In Impala 2.7 and higher, the REFRESH statement can apply to a single partition at a time, rather than the whole table. Include the optional PARTITION (*PARTITION_SPEC*) clause and specify values for each of the partition key columns.

The following rules apply:

- The PARTITION clause of the REFRESH statement must include all the partition key columns.
- The order of the partition key columns does not have to match the column order in the table.
- Specifying a nonexistent partition does not cause an error.
- The partition can be one that Impala created and is already aware of, or a new partition created through Hive.

The following examples demonstrates the above rules.

```
-- Partition doesn't exist.
REFRESH p2 PARTITION (y=0, z=3);
REFRESH p2 PARTITION (y=0, z=-1)

-- Key columns specified in a different order than the table definition.
REFRESH p2 PARTITION (z=1, y=0)

-- Incomplete partition spec causes an error.
REFRESH p2 PARTITION (y=0)
ERROR: AnalysisException: Items in partition spec must exactly match the
partition columns in the table definition: default.p2 (1 vs 2)
```

Related impala-shell options:

Due to the expense of reloading the metadata for all tables, the `impala-shell` -r option is not recommended.

HDFS considerations:

All HDFS and Ranger permissions and privilege requirements are the same whether you refresh the entire table or a single partition.

The REFRESH statement checks HDFS permissions of the underlying data files and directories, caching this information so that a statement can be cancelled immediately if for example the impala user does not have permission to write to the data directory for the table. Impala reports any lack of write permissions as an INFO message in the log file.

If you change HDFS permissions to make data readable or writeable by the Impala user, issue another REFRESH to make Impala aware of the change.

Kudu considerations:

By default, much of the metadata for Kudu tables is handled by the underlying storage layer. Kudu tables have less reliance on the Metastore database, and require less metadata caching on the Impala side. For example, information about partitions in Kudu tables is managed by Kudu, and Impala does not cache any block locality metadata for Kudu tables. If the Kudu service is not integrated with the Hive Metastore, Impala will manage Kudu table metadata in the Hive Metastore.

The REFRESH and INVALIDATE METADATA statements are needed less frequently for Kudu tables than for HDFS-backed tables. Neither statement is needed when data is added to, removed, or updated in a Kudu table, even if the changes are made directly to Kudu through a client program using the Kudu API. Run REFRESH *TABLE_NAME* or INVALIDATE METADATA *TABLE_NAME* for a Kudu table only after making a change to the Kudu table schema, such as adding or dropping a column.

REFRESH AUTHORIZATION statement

The REFRESH AUTHORIZATION statement explicitly refreshes authorization data, including privileges and principals. When there is an external update to authorization metadata, use this statement to force Impala to refresh its authorization data without having to wait for the Ranger polling or run INVALIDATE METADATA.

Once issued, the REFRESH AUTHORIZATION statement cannot be cancelled.

Syntax:

```
REFRESH AUTHORIZATION
```

Usage notes: If authorization is not enabled, Impala returns an error.

Added in: Impala 3.2

Related Information

Enabling and using authorization in Impala

REFRESH FUNCTIONS statement

In Impala 2.9 and higher, you can run the REFRESH FUNCTIONS statement to refresh the user-defined functions (UDFs) created outside of Impala. For example, you can add Java-based UDFs to the metastore database through the Hive CREATE FUNCTION statements and make those UDFs visible to Impala at the database level by subsequently running REFRESH FUNCTIONS.

The REFRESH FUNCTIONS statement is only required if you create or modify UDFs from outside of Impala. Updated metadata, as a result of running REFRESH FUNCTIONS, is broadcast to all Impala coordinators.

Once issued, the REFRESH FUNCTIONS statement cannot be cancelled.

Syntax:

REFRESH FUNCTIONS *DB_NAME*

REVOKE statement

The REVOKE statement revokes privileges on a specified object from groups or users.

Syntax:

```
      REVOKE PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
      FROM USER USER_NAME
      REVOKE PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
      FROM GROUP GROUP_NAME

        privilege ::= ALL | ALTER | CREATE | DROP | INSERT | REFRESH |
 SELECT | SELECT(COLUMN_NAME)

      object_type ::= SERVER | URI | DATABASE | TABLE
```

See the GRANT statement for the required privileges and the scope for SQL operations.

The ALL privilege is a distinct privilege and not a union of all other privileges.

The following syntax is supported when Impala is using Ranger to manage authorization.

```
REVOKE PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
  FROM USER USER_NAME

REVOKE PRIVILEGE ON OBJECT_TYPE OBJECT_NAME
  FROM GROUP GROUP_NAME


  privilege ::= ALL | ALTER | CREATE | DROP | INSERT | REFRESH | SELECT |
 SELECT(COLUMN_NAME)

object_type ::= SERVER | URI | DATABASE | TABLE
```

You cannot revoke a privilege granted with the WITH GRANT OPTION. If a privilege is granted with the WITH GRANT OPTION, first revoke the grant option, and then revoke the privilege.

Typically, the object name is an identifier. For URIs, it is a string literal.

The ability to grant or revoke SELECT privilege on specific columns is available in Impala 2.3 and higher.

Required privileges:

Only administrative users those with ALL privileges on the server can use this statement.

Compatibility:

- The Impala REVOKE statement is available in Impala 2.0 and higher.
- In Impala 1.4 and higher, Impala makes use of any privileges specified by the GRANT and REVOKE statements in Hive, when your system is configured to use the Ranger service.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Related Information

GRANT statement

REVOKE ROLE statement

The REVOKE statement REMOVES the role assigned to the group.

Syntax

```
REVOKE ROLE ROLE_NAME FROM GROUP GROUP_NAME
```

SELECT statement

The SELECT statement performs queries, retrieving data from one or more tables and producing result sets consisting of rows and columns.

The Impala INSERT statement also typically ends with a SELECT statement, to define data to copy from one table to another.

Syntax:

```
[WITH name AS (select_expression) [, ...] ]
SELECT
  [ALL | DISTINCT]
  [STRAIGHT_JOIN]
  expression [, expression ...]
FROM table_reference [, table_reference ...]
[[FULL | [LEFT | RIGHT] INNER | [LEFT | RIGHT] OUTER | [LEFT | RIGHT] SEMI |
 [LEFT | RIGHT] ANTI | CROSS]
  JOIN table_reference
  [ON join_equality_clauses | USING (COL1[, COL2 ...]] ...
WHERE conditions
GROUP BY { column | expression [, ...] }
HAVING conditions
ORDER BY { column | expression [ASC | DESC] [NULLS FIRST | NULLS LAST] [,
...] }
LIMIT expression [OFFSET expression]
[UNION [ALL] select_statement] ...]

table_reference := { TABLE_NAME | (SUBQUERY) }
  [ TABLESAMPLE SYSTEM(PERCENTAGE) [REPEATABLE(SEED)] ]
```

Impala SELECT queries support:

- SQL scalar data types: BOOLEAN, TINYINT, SMALLINT, INT, BIGINT, DECIMAL FLOAT, DOUBLE, TIMESTAMP, STRING, VARCHAR, CHAR.
- The complex data types ARRAY, STRUCT, and MAP, are available in Impala 2.3 and higher. Queries involving these types typically involve special qualified names using dot notation for referring to the complex column fields, and join clauses for bringing the complex columns into the result set.
- An optional WITH clause before the SELECT keyword, to define a subquery whose name or column names can be referenced from later in the main query. This clause lets you abstract repeated clauses, such as aggregation functions, that are referenced multiple times in the same query.
- Subqueries in a FROM clause. In Impala 2.0 and higher, subqueries can also go in the WHERE clause, for example with the IN(), EXISTS, and NOT EXISTS operators.
- WHERE, GROUP BY, HAVING clauses.
- ORDER BY. Prior to Impala 1.4.0, Impala required that queries using an ORDER BY clause also include a LIMIT clause. In Impala 1.4.0 and higher, this restriction is lifted; sort operations that would exceed the Impala memory limit automatically use a temporary disk work area to perform the sort.
- You can refer to SELECT-list items by their ordinal position. Impala supports ordinals in the GROUP BY, HAVI NG, and ORDER BY clauses. From Impala 3.0, ordinals can only be used at the top level. For example, the following statements are allowed:

```
  SELECT int_col / 2, sum(x)
  FROM t
  GROUP BY 1;

  SELECT int_col / 2
  FROM t
  ORDER BY 1;

  SELECT NOT bool_col
  FROM t
  GROUP BY 1
  HAVING 1;
```

Numbers in subexpressions are not interpreted as ordinals:

```
  SELECT int_col / 2, sum(x)
  FROM t
  GROUP BY 1 * 2;
The above parses OK, however GROUP BY 1 * 2 has no effect.
  SELECT int_col / 2
  FROM t
  ORDER BY 1 + 2;
The above parses OK, however ORDER BY 1 + 2 has no effect.
  SELECT NOT bool_col
  FROM t
  GROUP BY 1
  HAVING not 1;
The above raises an error at parse-time.
```

- Impala supports a wide variety of JOIN clauses. Left, right, semi, full, and outer joins are supported in all Impala versions. The CROSS JOIN operator is available in Impala 1.2.2 and higher. During performance tuning, you can override the reordering of join clauses that Impala does internally by including the keyword STRAIGHT_JOIN immediately after the SELECT and any DISTINCT or ALL keywords.
- UNION ALL.
- LIMIT.
- External tables.
- Relational operators such as greater than, less than, or equal to.
- Arithmetic operators such as addition or subtraction.
- Logical/Boolean operators AND, OR, and NOT. Impala does not support the corresponding symbols &&, ||, and !.
- Common SQL built-in functions such as COUNT, SUM, CAST, LIKE, IN, BETWEEN, and COALESCE.
- On optional TABLESAMPLE clause immediately after a table reference, to specify that the query only processes a specified percentage of the table data.

Impala queries ignore files with extensions commonly used for temporary work files by Hadoop tools. Any files with extensions .tmp or .copying are not considered part of the Impala table. The suffix matching is case-insensitive, so for example Impala ignores both .copying and .COPYING suffixes.

Security considerations:

If these statements in your environment contain sensitive literal values such as credit card numbers or tax identifiers, Impala can redact this sensitive information when displaying the statements in log files and other administrative contexts.

Amazon S3 considerations:

In Impala 2.6 and higher, Impala queries are optimized for files stored in Amazon S3. For Impala tables that use the file formats Parquet, ORC, RCFile, SequenceFile, Avro, and uncompressed text, the setting fs.s3a.block .size in the core-site.xml configuration file determines how Impala divides the I/O work of reading the data files. This configuration setting is specified in bytes. By default, this value is 33554432 (32 MB), meaning that Impala parallelizes S3 read operations on the files as if they were made up of 32 MB blocks. For example, if your S3 queries

primarily access Parquet files written by MapReduce or Hive, increase fs.s3a.block.size to 134217728 (128 MB) to match the row group size of those files. If most S3 queries involve Parquet files written by Impala, increase fs.s3a.block.size to 268435456 (256 MB) to match the row group size produced by Impala.

Cancellation: Can be cancelled. To cancel this statement, use Ctrl-C from the `impala-shell` interpreter, the Cancel button from the Watch page in Hue, Actions > Cancel from the Queries list in Cloudera Manager, or Cancel from the list of in-flight queries (for a particular node) on the Queries tab in the Impala web UI (port 25000).

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read permissions for the files in all applicable directories in all source tables, and read and execute permissions for the relevant data directories. (A SELECT operation could read files from multiple different HDFS directories if the source table is partitioned.) If a query attempts to read a data file and is unable to because of an HDFS permission error, the query halts and does not return any further results.

Related information:

The SELECT syntax is so extensive that it forms its own category of statements: queries. The other major classifications of SQL statements are data definition language and data manipulation language.

Because the focus of Impala is on fast queries with interactive response times over huge data sets, query performance and scalability are important considerations.

Joins in Impala SELECT statements

A join query is a SELECT statement that combines data from two or more tables, and returns a result set containing items from some or all of those tables. It is a way to cross-reference and correlate related data that is organized into multiple tables, typically using identifiers that are repeated in each of the joined tables.

Syntax:

Impala supports a wide variety of JOIN clauses. Left, right, semi, full, and outer joins are supported in all Impala versions. The CROSS JOIN operator is available in Impala 1.2.2 and higher. During performance tuning, you can override the reordering of join clauses that Impala does internally by including the keyword STRAIGHT_JOIN immediately after the SELECT and any DISTINCT or ALL keywords.

```
SELECT SELECT_LIST FROM
  TABLE_OR_SUBQUERY1 [INNER] JOIN TABLE_OR_SUBQUERY2 |
  TABLE_OR_SUBQUERY1 {LEFT [OUTER] | RIGHT [OUTER] | FULL [OUTER]} J
OIN TABLE_OR_SUBQUERY2 |
  TABLE_OR_SUBQUERY1 {LEFT | RIGHT} SEMI JOIN TABLE_OR_SUBQUERY2 |
  TABLE_OR_SUBQUERY1 {LEFT | RIGHT} ANTI JOIN TABLE_OR_SUBQUERY2 |
    [ ON COL1 = COL2 [AND COL3 = COL4 ...] |
      USING (COL1 [, COL2 ...]) ]
  [OTHER_JOIN_CLAUSE ...]
[ WHERE WHERE_CLAUSES ]

SELECT SELECT_LIST FROM
  TABLE_OR_SUBQUERY1, TABLE_OR_SUBQUERY2 [, TABLE_OR_SUBQUERY3 ...]
  [OTHER_JOIN_CLAUSE ...]
WHERE
    COL1 = COL2 [AND COL3 = COL4 ...]

SELECT SELECT_LIST FROM
  TABLE_OR_SUBQUERY1 CROSS JOIN TABLE_OR_SUBQUERY2
  [OTHER_JOIN_CLAUSE ...]
[ WHERE WHERE_CLAUSES ]
```

SQL-92 and SQL-89 Joins:

Queries with the explicit JOIN keywords are known as SQL-92 style joins, referring to the level of the SQL standard where they were introduced. The corresponding ON or USING clauses clearly show which columns are used as the join keys in each case:

```
SELECT t1.c1, t2.c2 FROM t1 JOIN t2
  ON t1.id = t2.id and t1.type_flag = t2.type_flag
  WHERE t1.c1 > 100;

SELECT t1.c1, t2.c2 FROM t1 JOIN t2
  USING (id, type_flag)
  WHERE t1.c1 > 100;
```

The ON clause is a general way to compare columns across the two tables, even if the column names are different. The USING clause is a shorthand notation for specifying the join columns, when the column names are the same in both tables. You can code equivalent WHERE clauses that compare the columns, instead of ON or USING clauses, but that practice is not recommended because mixing the join comparisons with other filtering clauses is typically less readable and harder to maintain.

Queries with a comma-separated list of tables and subqueries are known as SQL-89 style joins. In these queries, the equality comparisons between columns of the joined tables go in the WHERE clause alongside other kinds of comparisons. This syntax is easy to learn, but it is also easy to accidentally remove a WHERE clause needed for the join to work correctly.

```
SELECT t1.c1, t2.c2 FROM t1, t2
  WHERE
  t1.id = t2.id AND t1.type_flag = t2.type_flag
  AND t1.c1 > 100;
```

Self-joins:

Impala can do self-joins, for example to join on two different columns in the same table to represent parent-child relationships or other tree-structured data. There is no explicit syntax for this; just use the same table name for both the left-hand and right-hand table, and assign different table aliases to use when referring to the fully qualified column names:

```
-- Combine fields from both parent and child rows.
SELECT lhs.id, rhs.parent, lhs.c1, rhs.c2 FROM tree_data lhs, tree_data rhs
 WHERE lhs.id = rhs.parent;
```

Inner and outer joins:

An inner join is the most common and familiar type: rows in the result set contain the requested columns from the appropriate tables, for all combinations of rows where the join columns of the tables have identical values. If a column with the same name occurs in both tables, use a fully qualified name or a column alias to refer to the column in the select list or other clauses. Impala performs inner joins by default for both SQL-89 and SQL-92 join syntax:

```
-- The following 3 forms are all equivalent.
SELECT t1.id, c1, c2 FROM t1, t2 WHERE t1.id = t2.id;
SELECT t1.id, c1, c2 FROM t1 JOIN t2 ON t1.id = t2.id;
SELECT t1.id, c1, c2 FROM t1 INNER JOIN t2 ON t1.id = t2.id;
```

An outer join retrieves all rows from the left-hand table, or the right-hand table, or both; wherever there is no matching data in the table on the other side of the join, the corresponding columns in the result set are set to NULL. To perform an outer join, include the OUTER keyword in the join operator, along with either LEFT, RIGHT, or FULL:

```
SELECT * FROM t1 LEFT OUTER JOIN t2 ON t1.id = t2.id;
SELECT * FROM t1 RIGHT OUTER JOIN t2 ON t1.id = t2.id;
SELECT * FROM t1 FULL OUTER JOIN t2 ON t1.id = t2.id;
```

For outer joins, Impala requires SQL-92 syntax; that is, the JOIN keyword instead of comma-separated table names. Impala does not support vendor extensions such as (+) or *= notation for doing outer joins with SQL-89 query syntax.

Equijoins and Non-Equijoins:

By default, Impala requires an equality comparison between the left-hand and right-hand tables, either through ON, USING, or WHERE clauses. These types of queries are classified broadly as equijoins. Inner, outer, full, and semi joins can all be equijoins based on the presence of equality tests between columns in the left-hand and right-hand tables.

In Impala 1.2.2 and higher, non-equijoin queries are also possible, with comparisons such as != or < between the join columns. These kinds of queries require care to avoid producing huge result sets that could exceed resource limits. Once you have planned a non-equijoin query that produces a result set of acceptable size, you can code the query using the CROSS JOIN operator, and add the extra comparisons in the WHERE clause:

```
SELECT * FROM t1 CROSS JOIN t2 WHERE t1.total > t2.maximum_price;
```

In Impala 2.3 and higher, additional non-equijoin queries are possible due to the addition of nested loop joins. These queries typically involve SEMI JOIN, ANTI JOIN, or FULL OUTER JOIN clauses. Impala sometimes also uses nested loop joins internally when evaluating OUTER JOIN queries involving complex type columns. Query phases involving nested loop joins do not use the spill-to-disk mechanism if they exceed the memory limit. Impala decides internally when to use each join mechanism; you cannot specify any query hint to choose between the nested loop join or the original hash join algorithm.

```
SELECT * FROM t1 LEFT OUTER JOIN t2 ON t1.int_col < t2.int_col;
```

Semi-joins:

Semi-joins are a relatively rarely used variation. With the left semi-join, only data from the left-hand table is returned, for rows where there is matching data in the right-hand table, based on comparisons between join columns in ON or WHERE clauses. Only one instance of each row from the left-hand table is returned, regardless of how many matching rows exist in the right-hand table. A right semi-join (available in Impala 2.0 and higher) reverses the comparison and returns data from the right-hand table.

```
SELECT t1.c1, t1.c2, t1.c2 FROM t1 LEFT SEMI JOIN t2 ON t1.id = t2.id;
```

Natural joins (not supported):

Impala does not support the NATURAL JOIN operator, again to avoid inconsistent or huge result sets. Natural joins do away with the ON and USING clauses, and instead automatically join on all columns with the same names in the left-hand and right-hand tables. This kind of query is not recommended for rapidly evolving data structures such as are typically used in Hadoop. Thus, Impala does not support the NATURAL JOIN syntax, which can produce different query results as columns are added to or removed from tables.

If you do have any queries that use NATURAL JOIN, make sure to rewrite them with explicit USING clauses, because Impala could interpret the NATURAL keyword as a table alias:

```
-- 'NATURAL' is interpreted as an alias for 't1' and Impala attempts an inne
r join,
-- resulting in an error because inner joins require explicit comparisons be
tween columns.
SELECT t1.c1, t2.c2 FROM t1 NATURAL JOIN t2;
ERROR: NotImplementedException: Join with 't2' requires at least one conjunc
tive equality predicate.
  To perform a Cartesian product between two tables, use a CROSS JOIN.
-- If you expect the tables to have identically named columns with matching
 values,
-- list the corresponding column names in a USING clause.
SELECT t1.c1, t2.c2 FROM t1 JOIN t2 USING (id, type_flag, name, address);
```

Anti-joins (Impala 2.0 and higher only):

Impala supports the LEFT ANTI JOIN and RIGHT ANTI JOIN clauses in Impala 2.0 and higher. The LEFT or RIGHT keyword is required for this kind of join. For LEFT ANTI JOIN, this clause returns those values from the left-hand table that have no matching value in the right-hand table. RIGHT ANTI JOIN reverses the comparison and returns values from the right-hand table. You can express this negative relationship either through the ANTI JOIN clause or through a NOT EXISTS operator with a subquery.

Complex type considerations:

When referring to a column with a complex type (STRUCT, ARRAY, or MAP) in a query, you use join notation to “unpack” the scalar fields of the struct, the elements of the array, or the key-value pairs of the map. (The join notation is not required for aggregation operations, such as COUNT() or SUM() for array elements.) Because Impala recognizes which complex type elements are associated with which row of the result set, you use the same syntax as for a cross or cartesian join, without an explicit join condition.

Usage notes:

You typically use join queries in situations like these:

- When related data arrives from different sources, with each data set physically residing in a separate table. For example, you might have address data from business records that you cross-check against phone listings or census data.

  Note: Impala can join tables of different file formats, including Impala-managed tables and HBase tables. For example, you might keep small dimension tables in HBase, for convenience of single-row lookups and updates, and for the larger fact tables use Parquet or other binary file format optimized for scan operations. Then, you can issue a join query to cross-reference the fact tables with the dimension tables.
- When data is normalized, a technique for reducing data duplication by dividing it across multiple tables. This kind of organization is often found in data that comes from traditional relational database systems. For example, instead of repeating some long string such as a customer name in multiple tables, each table might contain a numeric customer ID. Queries that need to display the customer name could “join” the table that specifies which customer ID corresponds to which name.
- When certain columns are rarely needed for queries, so they are moved into separate tables to reduce overhead for common queries. For example, a biography field might be rarely needed in queries on employee data. Putting that field in a separate table reduces the amount of I/O for common queries on employee addresses or phone numbers. Queries that do need the biography column can retrieve it by performing a join with that separate table.
- In Impala 2.3 or higher, when referring to complex type columns in queries.

When comparing columns with the same names in ON or WHERE clauses, use the fully qualified names such as *DB_NAME.TABLE_NAME*, or assign table aliases, column aliases, or both to make the code more compact and understandable:

```
select t1.c1 as first_id, t2.c2 as second_id from
  t1 join t2 on first_id = second_id;
select fact.custno, dimension.custno from
  customer_data as fact join customer_address as dimension
  using (custno)
```

Note:

Performance for join queries is a crucial aspect for Impala, because complex join queries are resource-intensive operations. An efficient join query produces much less network traffic and CPU overhead than an inefficient one. For best results:

- Make sure that both table and column statistics are available for all the tables involved in a join query, and especially for the columns referenced in any join conditions. Impala uses the statistics to automatically deduce an efficient join order. Use SHOW TABLE STATS *TABLE_NAME* and SHOW COLUMN STATS *TABLE_NAME* to check if statistics are already present. Issue the COMPUTE STATS *TABLE_NAME* for a nonpartitioned table, or (in Impala 2.1.0 and higher) COMPUTE INCREMENTAL STATS *TABLE_NAME* for a partitioned table, to collect the initial statistics at both the table and column levels, and to keep the statistics up to date after any substantial INSERT or LOAD DATA operations.
- If table or column statistics are not available, join the largest table first. You can check the existence of statistics with the SHOW TABLE STATS *TABLE_NAME* and SHOW COLUMN STATS *TABLE_NAME* statements.
- If table or column statistics are not available, join subsequent tables according to which table has the most selective filter, based on overall size and WHERE clauses. Joining the table with the most selective filter results in the fewest number of rows being returned.

To control the result set from a join query, include the names of corresponding column names in both tables in an ON or USING clause, or by coding equality comparisons for those columns in the WHERE clause.

```
[localhost:21000] > select c_last_name, ca_city from customer join customer_
address where c_customer_sk = ca_address_sk;
+-------------+-----------------+
| c_last_name | ca_city         |
+-------------+-----------------+
| Lewis       | Fairfield       |
| Moses       | Fairview        |
| Hamilton    | Pleasant Valley |
| White       | Oak Ridge       |
| Moran       | Glendale        |
...
| Richards    | Lakewood         |
| Day         | Lebanon          |
| Painter     | Oak Hill         |
| Bentley     | Greenfield       |
| Jones       | Stringtown       |
+-------------+------------------+
Returned 50000 row(s) in 9.82s
```

One potential downside of joins is the possibility of excess resource usage in poorly constructed queries. Impala imposes restrictions on join queries to guard against such issues. To minimize the chance of runaway queries on large data sets, Impala requires every join query to contain at least one equality predicate between the columns of the various tables. For example, if T1 contains 1000 rows and T2 contains 1,000,000 rows, a query SELECT *COLUMNS* FROM t1 JOIN t2 could return up to 1 billion rows (1000 * 1,000,000); Impala requires that the query include a clause such as ON t1.c1 = t2.c2 or WHERE t1.c1 = t2.c2.

Because even with equality clauses, the result set can still be large, as we saw in the previous example, you might use a LIMIT clause to return a subset of the results:

```
[localhost:21000] > select c_last_name, ca_city from customer, customer_addr
ess where c_customer_sk = ca_address_sk limit 10;
+-------------+-----------------+
| c_last_name | ca_city         |
+-------------+-----------------+
| Lewis       | Fairfield       |
| Moses       | Fairview        |
| Hamilton    | Pleasant Valley |
```

```
| White        | Oak Ridge       |
| Moran        | Glendale        |
| Sharp        | Lakeview        |
| Wiles        | Farmington      |
| Shipman      | Union           |
| Gilbert      | New Hope        |
| Brunson      | Martinsville    |
+-------------+-----------------+
Returned 10 row(s) in 0.63s
```

Or you might use additional comparison operators or aggregation functions to condense a large result set into a smaller set of values:

```
[localhost:21000] > -- Find the names of customers who live in one particula
r town.
[localhost:21000] > select distinct c_last_name from customer, customer_addr
ess where
  c_customer_sk = ca_address_sk
  and ca_city = "Green Acres";
+---------------+
| c_last_name   |
+---------------+
| Hensley       |
| Pearson       |
| Mayer         |
| Montgomery    |
| Ricks         |
...
| Barrett       |
| Price         |
| Hill          |
| Hansen        |
| Meeks         |
+---------------+
Returned 332 row(s) in 0.97s

[localhost:21000] > -- See how many different customers in this town have
 names starting with "A".
[localhost:21000] > select count(distinct c_last_name) from customer, cust
omer_address where
  c_customer_sk = ca_address_sk
  and ca_city = "Green Acres"
  and substr(c_last_name,1,1) = "A";
+-----------------------------+
| count(distinct c_last_name) |
+-----------------------------+
| 12                          |
+-----------------------------+
Returned 1 row(s) in 1.00s
```

Because a join query can involve reading large amounts of data from disk, sending large amounts of data across the network, and loading large amounts of data into memory to do the comparisons and filtering, you might do benchmarking, performance analysis, and query tuning to find the most efficient join queries for your data set, hardware capacity, network configuration, and cluster workload.

The two categories of joins in Impala are known as partitioned joins and broadcast joins. If inaccurate table or column statistics, or some quirk of the data distribution, causes Impala to choose the wrong mechanism for a particular join, consider using query hints as a temporary workaround.

Handling NULLs in join columns:

By default, join key columns do not match if either one contains a NULL value. To treat such columns as equal if both contain NULL, you can use an expression such as A = B OR (A IS NULL AND B IS NULL). In Impala 2.5

and higher, the <=> operator (shorthand for IS NOT DISTINCT FROM) performs the same comparison in a concise and efficient form. The <=> operator is more efficient in for comparing join keys in a NULL-safe manner, because the operator can use a hash join while the OR expression cannot.

Examples:

The following examples refer to these simple tables containing small sets of integers:

```
[localhost:21000] > create table t1 (x int);
[localhost:21000] > insert into t1 values (1), (2), (3), (4), (5), (6);

[localhost:21000] > create table t2 (y int);
[localhost:21000] > insert into t2 values (2), (4), (6);
[localhost:21000] > create table t3 (z int);
[localhost:21000] > insert into t3 values (1), (3), (5);
```

The following example demonstrates an anti-join, returning the values from T1 that do not exist in T2 (in this case, the odd numbers 1, 3, and 5):

```
[localhost:21000] > select x from t1 left anti join t2 on (t1.x = t2.y);
+---+
| x |
+---+
| 1 |
| 3 |
| 5 |
+---+
```

Related Information

Optimizer hints in Impala

Performance considerations for join queries

SHOW statement

ORDER BY clause

The ORDER BY clause of a SELECT statement sorts the result set based on the values from one or more columns.

First, data is sorted locally by each impalad daemon, then streamed to the coordinator daemon, which merges the sorted result sets. For distributed queries, this is a relatively expensive operation and can require more memory capacity than a query without ORDER BY. Even if the query takes approximately the same time to finish with or without the ORDER BY clause, subjectively it can appear slower because no results are available until all processing is finished, rather than results coming back gradually as rows matching the WHERE clause are found. Therefore, if you only need the first N results from the sorted result set, also include the LIMIT clause, which reduces network overhead and the memory requirement on the coordinator node.

Syntax:

The full syntax for the ORDER BY clause is:

```
ORDER BY COL_REF [, COL_REF ...] [ASC | DESC] [NULLS FIRST | NULLS LAST]

col_ref ::= COLUMN_NAME | INTEGER_LITERAL
```

Although the most common usage is ORDER BY *COLUMN_NAME*, you can also specify ORDER BY 1 to sort by the first column of the result set, ORDER BY 2 to sort by the second column, and so on. The number must be a numeric literal, not some other kind of constant expression. (If the argument is some other expression, even a STRING value, the query succeeds but the order of results is undefined.)

ORDER BY *COLUMN_NUMBER* can only be used when the query explicitly lists the columns in the SELECT list, not with SELECT * queries.

Ascending and descending sorts:

The default sort order (the same as using the ASC keyword) puts the smallest values at the start of the result set, and the largest values at the end. Specifying the DESC keyword reverses that order.

Sort order for NULL values:

See the NULL on page 93 section for details about how NULL values are positioned in the sorted result set, and how to use the NULLS FIRST and NULLS LAST clauses. (The sort position for NULL values in ORDER BY ... DESC queries is changed in Impala 1.2.1 and higher to be more standards-compliant, and the NULLS FIRST and NULLS LAST keywords are new in Impala 1.2.1.)

Prior to Impala 1.4.0, Impala required any query including an ORDER_BY clause to also use a LIMIT clause. In Impala 1.4.0 and higher, the LIMIT clause is optional for ORDER BY queries. In cases where sorting a huge result set requires enough memory to exceed the Impala memory limit for a particular executor Impala daemon, Impala automatically uses a temporary disk work area to perform the sort operation.

Complex type considerations:

In Impala 2.3 and higher, the complex data types STRUCT, ARRAY, and MAP are available. These columns cannot be referenced directly in the ORDER BY clause. When you query a complex type column, you use join notation to “unpack” the elements of the complex type, and within the join query you can include an ORDER BY clause to control the order in the result set of the scalar elements from the complex type.

The following query shows how a complex type column cannot be directly used in an ORDER BY clause:

```
CREATE TABLE games (id BIGINT, score ARRAY <BIGINT>) STORED AS PARQUET;
...use LOAD DATA to load externally created Parquet files into the table...
SELECT id FROM games ORDER BY score DESC;
ERROR: AnalysisException: ORDER BY expression 'score' with complex type 'A
RRAY<BIGINT>' is not supported.
```

Examples:

The following query retrieves the user ID and score, only for scores greater than one million, with the highest scores for each user listed first. Because the individual array elements are now represented as separate rows in the result set, they can be used in the ORDER BY clause, referenced using the ITEM pseudo-column that represents each array element.

```
SELECT id, item FROM games, games.score
  WHERE item > 1000000
ORDER BY id, item desc;
```

The following queries use similar ORDER BY techniques with variations of the GAMES table, where the complex type is an ARRAY containing STRUCT or MAP elements to represent additional details about each game that was played. For an array of structures, the fields of the structure are referenced as ITEM.*FIELD_NAME*. For an array of maps, the keys and values within each array element are referenced as ITEM.KEY and ITEM.VALUE.

```
CREATE TABLE games2 (id BIGINT, play array < struct <game_name: string, scor
e: BIGINT, high_score: boolean> >) STORED AS PARQUET
...use LOAD DATA to load externally created Parquet files into the table...
SELECT id, item.game_name, item.score FROM games2, games2.play
  WHERE item.score > 1000000
ORDER BY id, item.score DESC;

CREATE TABLE games3 (id BIGINT, play ARRAY < MAP <STRING, BIGINT> >) STORED
AS PARQUET;
...use LOAD DATA to load externally created Parquet files into the table...
SELECT id, info.key AS k, info.value AS v from games3, games3.play AS plays,
 games3.play.item AS info
  WHERE info.KEY = 'score' AND info.VALUE > 1000000
ORDER BY id, info.value desc;
```

Usage notes:

Although the LIMIT clause is now optional on ORDER BY queries, if your query only needs some number of rows that you can predict in advance, use the LIMIT clause to reduce unnecessary processing. For example, if the query has a clause LIMIT 10, each executor Impala daemon sorts its portion of the relevant result set and only returns 10 rows to the coordinator node. The coordinator node picks the 10 highest or lowest row values out of this small intermediate result set.

If an ORDER BY clause is applied to an early phase of query processing, such as a subquery or a view definition, Impala ignores the ORDER BY clause. To get ordered results from a subquery or view, apply an ORDER BY clause to the outermost or final SELECT level.

ORDER BY is often used in combination with LIMIT to perform “top-N” queries:

```
SELECT user_id AS "Top 10 Visitors", SUM(page_views) FROM web_stats
  GROUP BY page_views, user_id
  ORDER BY SUM(page_views) DESC LIMIT 10;
```

ORDER BY is sometimes used in combination with OFFSET and LIMIT to paginate query results, although it is relatively inefficient to issue multiple queries like this against the large tables typically used with Impala:

```
SELECT page_title AS "Page 1 of search results", page_url FROM search_conten
t
  WHERE LOWER(page_title) LIKE '%game%')
  ORDER BY page_title LIMIT 10 OFFSET 0;
SELECT page_title AS "Page 2 of search results", page_url FROM search_conten
t
  WHERE LOWER(page_title) LIKE '%game%')
  ORDER BY page_title LIMIT 10 OFFSET 10;
SELECT page_title AS "Page 3 of search results", page_url FROM search_conte
nt
  WHERE LOWER(page_title) LIKE '%game%')
  ORDER BY page_title LIMIT 10 OFFSET 20;
```

Internal details:

Impala sorts the intermediate results of an ORDER BY clause in memory whenever practical. In a cluster of N executor Impala daemons, each daemon sorts roughly 1/Nth of the result set, the exact proportion varying depending on how the data matching the query is distributed in HDFS.

If the size of the sorted intermediate result set on any executor Impala daemon would cause the query to exceed the Impala memory limit, Impala sorts as much as practical in memory, then writes partially sorted data to disk. (This technique is known in industry terminology as “external sorting” and “spilling to disk”.) As each 8 MB batch of data is written to disk, Impala frees the corresponding memory to sort a new 8 MB batch of data. When all the data has been processed, a final merge sort operation is performed to correctly order the in-memory and on-disk results as the result set is transmitted back to the coordinator node. When external sorting becomes necessary, Impala requires approximately 60 MB of RAM at a minimum for the buffers needed to read, write, and sort the intermediate results. If more RAM is available on the Impala daemon, Impala will use the additional RAM to minimize the amount of disk I/O for sorting.

This external sort technique is used as appropriate on each Impala daemon (possibly including the coordinator node) to sort the portion of the result set that is processed on that node. When the sorted intermediate results are sent back to the coordinator node to produce the final result set, the coordinator node uses a merge sort technique to produce a final sorted result set without using any extra resources on the coordinator node.

Sorting considerations:

Although you can specify an ORDER BY clause in an INSERT ... SELECT statement, any ORDER BY clause is ignored, and the results are not necessarily sorted.

An ORDER BY clause without an additional LIMIT clause is ignored in any view definition. If you need to sort the entire result set from a view, use an ORDER BY clause in the SELECT statement that queries the view. You can still make a simple “top 10” report by combining the ORDER BY and LIMIT clauses in the same view definition:

```
[localhost:21000] > create table unsorted (x bigint);
[localhost:21000] > insert into unsorted values (1), (9), (3), (7), (5), (
8), (4), (6), (2);
[localhost:21000] > create view sorted_view as select x from unsorted order
 by x;
[localhost:21000] > select x from sorted_view; -- ORDER BY clause in view h
as no effect.
+---+
| x |
+---+
| 1 |
| 9 |
| 3 |
| 7 |
| 5 |
| 8 |
| 4 |
| 6 |
| 2 |
+---+
[localhost:21000] > select x from sorted_view order by x; -- View query requ
ires ORDER BY at outermost level.
+---+
| x |
+---+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
+---+
[localhost:21000] > create view top_3_view as select x from unsorted order b
y x limit 3;
[localhost:21000] > select x from top_3_view; -- ORDER BY and LIMIT together
 in view definition are preserved.
+---+
| x |
+---+
| 1 |
| 2 |
| 3 |
+---+
```

With the lifting of the requirement to include a LIMIT clause in every ORDER BY query (in Impala 1.4 and higher):

- Now the use of scratch disk space raises the possibility of an “out of disk space” error on a particular Impala daemon, as opposed to the previous possibility of an “out of memory” error. Make sure to keep at least 1 GB free on the filesystem used for temporary sorting work.

In Impala 1.2.1 and higher, all NULL values come at the end of the result set for ORDER BY ... ASC queries, and at the beginning of the result set for ORDER BY ... DESC queries. In effect, NULL is considered greater than all other values for sorting purposes. The original Impala behavior always put NULL values at the end, even for ORDER BY ... DESC queries. The new behavior in Impala 1.2.1 makes Impala more compatible with other popular database

systems. In Impala 1.2.1 and higher, you can override or specify the sorting behavior for NULL by adding the clause NULLS FIRST or NULLS LAST at the end of the ORDER BY clause.

```
[localhost:21000] > create table numbers (x int);
[localhost:21000] > insert into numbers values (1), (null), (2), (null), (
3);
[localhost:21000] > select x from numbers order by x nulls first;
+------+
| x    |
+------+
| NULL |
| NULL |
| 1    |
| 2    |
| 3    |
+------+
[localhost:21000] > select x from numbers order by x desc nulls first;
+------+
| x    |
+------+
| NULL |
| NULL |
| 3    |
| 2    |
| 1    |
+------+
[localhost:21000] > select x from numbers order by x nulls last;
+------+
| x    |
+------+
| 1    |
| 2    |
| 3    |
| NULL |
| NULL |
+------+
[localhost:21000] > select x from numbers order by x desc nulls last;
+------+
| x    |
+------+
| 3    |
| 2    |
| 1    |
| NULL |
| NULL |
+------+
```

Using ORDER BY for collections of fixed length types in SELECT list

You can now use collections of fixed length types in the sorting tuple. However, you cannot sort by these collection columns but they can be in the SELECT list along with other column(s) by which you sort.

Example of a supported query:

```
select id, int_array from complextypestbl order by id;
```

Example of an unsupported query:

```
select int_array from complextypestbl order by int_array;
```

Also note that structs that contain any type of collection (even fixed length) are not allowed in the sorting tuple.

Examples:

Sort a collection.

```
select id, int_array from complextypestbl order by id;
```

Sort collection from HMS view.

```
select id, int_array from complextypes_arrays_only_view order by id;
```

Sort collection from WITH-clause inline view.

```
 with v as (select id, int_array from complextypestbl)
 select id, int_array from v order by id;
```

Sort collection from nested query inline view.

```
select id, int_array
from (select id, int_array from complextypestbl) v
order by id;
```

Related information:

See the SELECT statement for further examples of queries with the ORDER BY clause.

Analytic functions use the ORDER BY clause in a different context to define the sequence in which rows are analyzed. See Impala analytic functions on page 424 for details.

GROUP BY clause

Specify the GROUP BY clause in queries that use aggregation functions, such as COUNT(), SUM(), AVG(), MIN(), and MAX(). Specify in the GROUP BY clause the names of all the columns that do not participate in the aggregation operation.

Complex type considerations:

In Impala 2.3 and higher, the complex data types STRUCT, ARRAY, and MAP are available. These columns cannot be referenced directly in the ORDER BY clause. When you query a complex type column, you use join notation to “unpack” the elements of the complex type, and within the join query you can include an ORDER BY clause to control the order in the result set of the scalar elements from the complex type.

Zero-length strings: For purposes of clauses such as DISTINCT and GROUP BY, Impala considers zero-length strings (""), NULL, and space to all be different values.

Examples:

For example, the following query finds the 5 items that sold the highest total quantity (using the SUM() function, and also counts the number of sales transactions for those items (using the COUNT() function). Because the column representing the item IDs is not used in any aggregation functions, we specify that column in the GROUP BY clause.

```
select
  ss_item_sk as Item,
  count(ss_item_sk) as Times_Purchased,
  sum(ss_quantity) as Total_Quantity_Purchased
from store_sales
  group by ss_item_sk
  order by sum(ss_quantity) desc
  limit 5;
+-------+-----------------+--------------------------+
| item  | times_purchased | total_quantity_purchased |
+-------+-----------------+--------------------------+
| 9325  | 372             | 19072                    |
| 4279  | 357             | 18501                    |
```

```
| 7507  | 371             | 18475                    |
| 5953  | 369             | 18451                    |
| 16753 | 375             | 18446                    |
+-------+-----------------+--------------------------+
```

The HAVING clause lets you filter the results of aggregate functions, because you cannot refer to those expressions in the WHERE clause. For example, to find the 5 lowest-selling items that were included in at least 100 sales transactions, we could use this query:

```
select
  ss_item_sk as Item,
  count(ss_item_sk) as Times_Purchased,
  sum(ss_quantity) as Total_Quantity_Purchased
from store_sales
  group by ss_item_sk
      having count(ss_item_sk) >= 100
  order by sum(ss_quantity)
  limit 5;
+-------+-----------------+--------------------------+
| item  | times_purchased | total_quantity_purchased |
+-------+-----------------+--------------------------+
| 13943 | 105             | 4087                     |
| 2992  | 101             | 4176                     |
| 4773  | 107             | 4204                     |
| 14350 | 103             | 4260                     |
| 11956 | 102             | 4275                     |
+-------+-----------------+--------------------------+
```

When performing calculations involving scientific or financial data, remember that columns with type FLOAT or DOUBLE are stored as true floating-point numbers, which cannot precisely represent every possible fractional value. Thus, if you include a FLOAT or DOUBLE column in a GROUP BY clause, the results might not precisely match literal values in your query or from an original Text data file. Use rounding operations, the BETWEEN operator, or another arithmetic technique to match floating-point values that are “near” literal values you expect. For example, this query on the ss_wholesale_cost column returns cost values that are close but not identical to the original figures that were entered as decimal fractions.

```
select ss_wholesale_cost, avg(ss_quantity * ss_sales_price) as avg_revenue_p
er_sale
  from sales
  group by ss_wholesale_cost
  order by avg(ss_quantity * ss_sales_price) desc
  limit 5;
+-------------------+----------------------+
| ss_wholesale_cost | avg_revenue_per_sale |
+-------------------+----------------------+
| 96.94000244140625 | 4454.351539300434    |
| 95.93000030517578 | 4423.119941283189    |
| 98.37999725341797 | 4332.516490316291    |
| 97.97000122070312 | 4330.480601655014    |
| 98.52999877929688 | 4291.316953108634    |
+-------------------+----------------------+
```

Notice how wholesale cost values originally entered as decimal fractions such as 96.94 and 98.38 are slightly larger or smaller in the result set, due to precision limitations in the hardware floating-point types. The imprecise representation of FLOAT and DOUBLE values is why financial data processing systems often store currency using data types that are less space-efficient but avoid these types of rounding errors.

ROLLUP

The GROUP BY ROLLUP clause creates a group for each combination of column expressions. When you run a query with the “rollup” clause on a group of columns, it is run from right to left decreasing the number of column expressions over which it creates groups and the aggregation(s).

For example, GROUP BY ROLLUP (col1, col2, col3) creates groups for each combination of column expressions in the following lists.

- col1, col2, col3
- col1, col2, col1NULL
- col1, NULL, NULL
- NULL, NULL, NULL This is the grand total

Example:

The following query analyzes sales data, to study the amount of sales that is occurring for different products, in different states and regions using the ROLLUP feature.

```
select
 region, state, product, sum(sales) as total_sales,
from sales_history
 group by rollup (region, state, product);
```

Semantically the above query is equivalent to:

```
select region, state, product, sum(sales) as total_sales
      from sales_history
      group by region, state, product;

      union all

      select region, state, null, sum(sales) as  total_sales
      from sales_history
      group by region, state;

      union all

      select region, null, null, sum(sales) as  total_sales
      from sales_history
      group by region;

      union all

      select null, null, null, sum(sales) as total_sales
      from sales_history;
```

The query result has the same aggregations as the simple GROUP BY without the ROLLUP. In addition, it creates subtotals for each value of Region. Finally, it gives a grand total for all rows. The result looks like this:

```
      +--------+-------+---------+-------------+
      | region | state | product | total_sales |
      +--------+-------+---------+-------------+
      | NULL   | NULL  | NULL    | 6200        |
      | EAST   | MA    | BOATS   | 100         |
      | EAST   | MA    | CARS    | 1500        |
      | EAST   | MA    | NULL    | 1600        |
      | EAST   | NY    | BOATS   | 150         |
      | EAST   | NY    | CARS    | 1000        |
      | EAST   | NY    | NULL    | 1150        |
      | EAST   | NULL  | NULL    | 2750        |
```

```
| WEST   | CA    | CARS    | 500         |
| WEST   | CA    | BOATS   | 750         |
| WEST   | CA    | NULL    | 1250        |
| WEST   | AZ    | BOATS   | 2000        |
| WEST   | AZ    | CARS    | 200         |
| WEST   | AZ    | NULL    | 2200        |
| WEST   | NULL  | NULL    | 3450        |
+--------+-------+---------+-------------+
```

CUBE

This creates groups for all possible combinations of columns. For e.g., GROUP BY CUBE for (col1, col2) creates groups for:

- col1, col2
- col1, NULL
- NULL, col2
- NULL, NULL

Example:

```
select region, state, sum(sales) as total_sales
       from sales_history
       group by cube (region, state);
```

This will produce results for the following combinations (region, state), (NULL, state), (Region, NULL), and (NULL, NULL).

GROUPING SETS

A grouping set is a set of columns by which you group using the GROUP BY clause.

Example:

```
select region, state, sum(sales) as total_sales
from sales_history
group by grouping sets ((region, state), (region), (state), ());
```

This lets you define four grouping sets (region, state), (region), (state), and (). Semantically the above query is equivalent to:

```
select region, state, sum(sales) as total_sales
from sales_history
group by region, state;

union all

select region, null, sum(sales) as total_sales
from sales_history
group by region;

union all

select null, state, sum(sales) as total_sales
from sales_history
group by state;
```

```
        union all

        select null, null, sum(sales) as total_sales
        from sales_history;
```

HAVING clause

Performs a filter operation on a SELECT query, by examining the results of aggregation functions rather than testing each individual table row. Therefore, it is always used in conjunction with a function such as COUNT(), SUM(), AVG(), MIN(), or MAX(), and typically with the GROUP BY clause also.

Restrictions:

The filter expression in the HAVING clause cannot include a scalar subquery.

LIMIT clause

The LIMIT clause in a SELECT query sets a maximum number of rows for the result set. Pre-selecting the maximum size of the result set helps Impala to optimize memory usage while processing a distributed query.

Syntax:

```
LIMIT CONSTANT_INTEGER_EXPRESSION
```

The argument to the LIMIT clause must evaluate to a constant value. It can be a numeric literal, or another kind of numeric expression involving operators, casts, and function return values. You cannot refer to a column or use a subquery.

Usage notes:

This clause is useful in contexts such as:

- To return exactly N items from a top-N query, such as the 10 highest-rated items in a shopping category or the 50 hostnames that refer the most traffic to a web site.
- To demonstrate some sample values from a table or a particular query. (To display some arbitrary items, use a query with no ORDER BY clause. An ORDER BY clause causes additional memory and/or disk usage during the query.)
- To keep queries from returning huge result sets by accident if a table is larger than expected, or a WHERE clause matches more rows than expected.

Originally, the value for the LIMIT clause had to be a numeric literal. In Impala 1.2.1 and higher, it can be a numeric expression.

Prior to Impala 1.4.0, Impala required any query including an ORDER_BY clause to also use a LIMIT clause. In Impala 1.4.0 and higher, the LIMIT clause is optional for ORDER BY queries. In cases where sorting a huge result set requires enough memory to exceed the Impala memory limit for a particular executor Impala daemon, Impala automatically uses a temporary disk work area to perform the sort operation.

In Impala 1.2.1 and higher, you can combine a LIMIT clause with an OFFSET clause to produce a small result set that is different from a top-N query, for example, to return items 11 through 20. This technique can be used to simulate “paged” results. Because Impala queries typically involve substantial amounts of I/O, use this technique only for compatibility in cases where you cannot rewrite the application logic. For best performance and scalability, wherever practical, query as many items as you expect to need, cache them on the application side, and display small groups of results to users using application logic.

Restrictions:

Correlated subqueries used in EXISTS and IN operators cannot include a LIMIT clause.

Examples:

The following example shows how the LIMIT clause caps the size of the result set, with the limit being applied after any other clauses such as WHERE.

```
[localhost:21000] > create database limits;
[localhost:21000] > use limits;
[localhost:21000] > create table numbers (x int);
[localhost:21000] > insert into numbers values (1), (3), (4), (5), (2);
Inserted 5 rows in 1.34s
[localhost:21000] > select x from numbers limit 100;
+---+
| x |
+---+
| 1 |
| 3 |
| 4 |
| 5 |
| 2 |
+---+
Returned 5 row(s) in 0.26s
[localhost:21000] > select x from numbers limit 3;
+---+
| x |
+---+
| 1 |
| 3 |
| 4 |
+---+
Returned 3 row(s) in 0.27s
[localhost:21000] > select x from numbers where x > 2 limit 2;
+---+
| x |
+---+
| 3 |
| 4 |
+---+
Returned 2 row(s) in 0.27s
```

For top-N and bottom-N queries, you use the ORDER BY and LIMIT clauses together:

```
[localhost:21000] > select x as "Top 3" from numbers order by x desc limit 3
;
+-------+
| top 3 |
+-------+
| 5     |
| 4     |
| 3     |
+-------+
[localhost:21000] > select x as "Bottom 3" from numbers order by x limit 3;
+----------+
| bottom 3 |
+----------+
| 1        |
| 2        |
| 3        |
+----------+
```

You can use constant values besides integer literals as the LIMIT argument:

```
-- Other expressions that yield constant integer values work too.
SELECT x FROM t1 LIMIT 1e6;                        -- Limit is one million.
SELECT x FROM t1 LIMIT length('hello world');      -- Limit is 11.
```

```
SELECT x FROM t1 LIMIT 2+2;                        -- Limit is 4.
SELECT x FROM t1 LIMIT cast(truncate(9.9) AS INT); -- Limit is 9.
```

OFFSET clause

The OFFSET clause in a SELECT query causes the result set to start some number of rows after the logical first item. The result set is numbered starting from zero, so OFFSET 0 produces the same result as leaving out the OFFSET clause. Always use this clause in combination with ORDER BY (so that it is clear which item should be first, second, and so on) and LIMIT (so that the result set covers a bounded range, such as items 0-9, 100-199, and so on).

In Impala 1.2.1 and higher, you can combine a LIMIT clause with an OFFSET clause to produce a small result set that is different from a top-N query, for example, to return items 11 through 20. This technique can be used to simulate “paged” results. Because Impala queries typically involve substantial amounts of I/O, use this technique only for compatibility in cases where you cannot rewrite the application logic. For best performance and scalability, wherever practical, query as many items as you expect to need, cache them on the application side, and display small groups of results to users using application logic.

Examples:

The following example shows how you could run a “paging” query originally written for a traditional database application. Because typical Impala queries process megabytes or gigabytes of data and read large data files from disk each time, it is inefficient to run a separate query to retrieve each small group of items. Use this technique only for compatibility while porting older applications, then rewrite the application code to use a single query with a large result set, and display pages of results from the cached result set.

```
[localhost:21000] > create table numbers (x int);
[localhost:21000] > insert into numbers select x from very_long_sequence;
Inserted 1000000 rows in 1.34s
[localhost:21000] > select x from numbers order by x limit 5 offset 0;
+----+
| x  |
+----+
| 1  |
| 2  |
| 3  |
| 4  |
| 5  |
+----+
[localhost:21000] > select x from numbers order by x limit 5 offset 5;
+----+
| x  |
+----+
| 6  |
| 7  |
| 8  |
| 9  |
| 10 |
+----+
```

UNION, INTERSECT, and EXCEPT clauses

The UNION, INTERSECT, and EXCEPT clauses allow you to combine rows from two or more tables. You may use them when you must combine the result sets of multiple queries into one single result.

UNION Clause

The UNION clause lets you combine the result sets of multiple queries. By default, the result sets are combined as if the DISTINCT operator was applied.

Syntax:

```
QUERY_1 UNION [DISTINCT | ALL] QUERY_2
```

Usage notes:

The UNION keyword by itself is the same as UNION DISTINCT. Because eliminating duplicates can be a memory-intensive process for a large result set, prefer UNION ALL where practical. (That is, when you know the different queries in the union will not produce any duplicates, or where the duplicate values are acceptable.)

When an ORDER BY clause applies to a UNION ALL or UNION query, in Impala 1.4 and higher, the LIMIT clause is no longer required. To make the ORDER BY and LIMIT clauses apply to the entire result set, turn the UNION query into a subquery, SELECT from the subquery, and put the ORDER BY clause at the end, outside the subquery.

Examples:

First, set up some sample data, including duplicate 1 values:

```
[localhost:21000] > create table few_ints (x int);
[localhost:21000] > insert into few_ints values (1), (1), (2), (3);
[localhost:21000] > set default_order_by_limit=1000;
```

This example shows how UNION ALL returns all rows from both queries, without any additional filtering to eliminate duplicates. For the large result sets common with Impala queries, this is the most memory-efficient technique.

```
[localhost:21000] > select x from few_ints order by x;
+---+
| x |
+---+
| 1 |
| 1 |
| 2 |
| 3 |
+---+
Returned 4 row(s) in 0.41s
[localhost:21000] > select x from few_ints union all select x from few_ints;
+---+
| x |
+---+
| 1 |
| 1 |
| 2 |
| 3 |
| 1 |
| 1 |
| 2 |
| 3 |
+---+
Returned 8 row(s) in 0.42s
[localhost:21000] > select * from (select x from few_ints union all select x
 from few_ints) as t1 order by x;
+---+
| x |
+---+
| 1 |
| 1 |
| 1 |
| 1 |
| 2 |
| 2 |
| 3 |
| 3 |
```

```
+---+
Returned 8 row(s) in 0.53s
[localhost:21000] > select x from few_ints union all select 10;
+----+
| x  |
+----+
| 10 |
| 1  |
| 1  |
| 2  |
| 3  |
+----+
Returned 5 row(s) in 0.38s
```

This example shows how the UNION clause without the ALL keyword condenses the result set to eliminate all duplicate values, making the query take more time and potentially more memory. The extra processing typically makes this technique not recommended for queries that return result sets with millions or billions of values.

```
[localhost:21000] > select x from few_ints union select x+1 from few_ints;
+---+
| x |
+---+
| 3 |
| 4 |
| 1 |
| 2 |
+---+
Returned 4 row(s) in 0.51s
[localhost:21000] > select x from few_ints union select 10;
+----+
| x  |
+----+
| 2  |
| 10 |
| 1  |
| 3  |
+----+
Returned 4 row(s) in 0.49s
[localhost:21000] > select * from (select x from few_ints union select x fr
om few_ints) as t1 order by x;
+---+
| x |
+---+
| 1 |
| 2 |
| 3 |
+---+
Returned 3 row(s) in 0.53s
```

EXCEPT and INTERSECT Clauses

The EXCEPT and INTERSECT clause let you combine the result sets of multiple queries and return distinct rows by comparing the results of two queries. EXCEPT returns distinct rows from the left input query that are not part of the result set of the right input query. INTERSECT returns distinct rows that are output by both the left and right input queries operator.

Syntax:

```
QUERY_1 EXCEPT [DISTINCT] QUERY_2
```

```
QUERY_1 INTERSECT [DISTINCT] QUERY_2
```

Usage notes:

When you use EXCEPT or INTERSECT to combine the result sets of two queries the number and the order of the columns must be the same in all queries and the data types must be compatible.

Note: INTERSECT ALL and EXCEPT ALL are not currently supported.

Examples:

This example shows the number of Boeing planes leaving from DFW in 2015. And the subquery shown here uses an INTERSECT on the tail number (tailnum), the unique identification of a plane.

```
SELECT COUNT(*) FROM (
                 SELECT tailnum FROM flights
                 WHERE
                     flights.origin='DFW'
                     AND flights.year = 2005
                 INTERSECT
                 SELECT tailnum FROM planes
                 WHERE
                      planes.manufacturer = 'BOEING') subq;
```

The following example shows the number of non-Boeing planes leaving from DFW. This is done by executing the above query using the INTERSECT operator and then removing the planes that are not from the BOEING manufacturer using the EXCEPT clause.

```
SELECT COUNT(*) FROM (
                 SELECT tailnum FROM flights
                 WHERE
                     flights.origin='DFW'
                     AND flights.year = 2005
                 INTERSECT
                 SELECT tailnum FROM planes
                 EXCEPT
                 SELECT tailnum FROM planes
                 WHERE
                     planes.manufacturer = 'BOEING') subq;
```

Subqueries in Impala SELECT statements

A *subquery* is a query that is nested within another query. Subqueries let queries on one table dynamically adapt based on the contents of another table. This technique provides great flexibility and expressive power for SQL queries.

A subquery can return a result set for use in the FROM or WITH clauses, or with operators such as IN or EXISTS.

A *scalar subquery* produces a result set with a single row containing a single column, typically produced by an aggregation function such as MAX() or SUM(). This single result value can be substituted in scalar contexts such as arguments to comparison operators. If the result set is empty, the value of the scalar subquery is NULL. For example, the following query finds the maximum value of T2.Y and then substitutes that value into the WHERE clause of the outer block that queries T1:

```
SELECT x FROM t1 WHERE x > (SELECT MAX(y) FROM t2);
```

Uncorrelated subqueries do not refer to any tables from the outer block of the query. The same value or set of values produced by the subquery is used when evaluating each row from the outer query block. In this example, the subquery

returns an arbitrary number of values from T2.Y, and each value of T1.X is tested for membership in that same set of values:

```
SELECT x FROM t1 WHERE x IN (SELECT y FROM t2);
```

Uncorrelated subqueries are now supported in the SELECT list statement.

Example:

This query returns a row for every sale, along with the corresponding employee information.

```
      SELECT EmpID, EmpName, sum(SaleAmount) as EmpSales,
      sum(SaleAmount) / (SELECT sum(SaleAmount) FROM Sales) as PortionOfT
otalSales
      FROM Employee
      JOIN Sales on Employee.EmpID = Sales.EmpID GROUP BY EmpID, EmpName;
```

In this example, the subquery “(SELECT sum(SaleAmount) FROM Sales)” is an uncorrelated subquery that's evaluated separately and returns the sum across all sales, not grouped by any Employee details.

Then the first and second sum(saleAmount) will be the total of all sales for that employment.

Impala also supports *Uncorrelated subqueries* in the HAVING clause.

Example:

This query finds all the departments with the average salary greater than the average salary across all departments.

```
      SELECT dept_id, avg(salary) FROM employees GROUP BY dept_id HAVING
      avg(salary) > (SELECT avg(salary) FROM employees);
```

Correlated subqueries compare one or more values from the outer query block to values referenced in the WHERE clause of the subquery. Each row evaluated by the outer WHERE clause can be evaluated using a different set of values. These kinds of subqueries are restricted in the kinds of comparisons they can do between columns of the inner and outer tables. (See the following Restrictions item.)

For example, the following query finds all the employees with salaries that are higher than average for their department. The subquery potentially computes a different AVG() value for each employee.

```
SELECT employee_name, employee_id FROM employees one WHERE
  salary > (SELECT avg(salary) FROM employees two WHERE one.dept_id = two
.dept_id);
```

Syntax:

Subquery in the FROM clause:

```
SELECT SELECT_LIST FROM TABLE_REF [, TABLE_REF ...]

TABLE_REF ::= TABLE_NAME | (SELECT_STATEMENT)
```

Subqueries in WHERE clause:

```
WHERE VALUE COMPARISON_OPERATOR (SCALAR_SELECT_STATEMENT)
WHERE VALUE [NOT] IN (SELECT_STATEMENT)
WHERE [NOT] EXISTS (CORRELATED_SELECT_STATEMENT)
WHERE NOT EXISTS (CORRELATED_SELECT_STATEMENT)
```

comparison_operator is a numeric comparison such as =, <=, !=, and so on, or a string comparison operator such as LIKE or REGEXP.

Although you can use non-equality comparison operators such as < or >=, the subquery must include at least one equality comparison between the columns of the inner and outer query blocks.

All syntax is available for both correlated and uncorrelated queries, except that the NOT EXISTS clause cannot be used with an uncorrelated subquery.

Impala subqueries can be nested arbitrarily deep.

Standards compliance: Introduced in SQL:1999.

Examples:

This example illustrates how subqueries can be used in the FROM clause to organize the table names, column names, and column values by producing intermediate result sets, especially for join queries.

```
SELECT avg(t1.x), max(t2.y) FROM
  (SELECT id, cast(a AS DECIMAL(10,5)) AS x FROM raw_data WHERE a BETWEEN 0
 AND 100) AS t1
  JOIN
  (SELECT id, length(s) AS y FROM raw_data WHERE s LIKE 'A%') AS t2;
  USING (id);
```

These examples show how a query can test for the existence of values in a separate table using the EXISTS() operator with a subquery.

The following examples show how a value can be compared against a set of values returned by a subquery.

```
SELECT count(x) FROM t1 WHERE EXISTS(SELECT 1 FROM t2 WHERE t1.x = t2.y * 10
);

SELECT x FROM t1 WHERE x IN (SELECT y FROM t2 WHERE state = 'CA');
```

The following examples demonstrate scalar subqueries. When a subquery is known to return a single value, you can substitute it where you would normally put a constant value.

```
SELECT x FROM t1 WHERE y = (SELECT max(z) FROM t2);
SELECT x FROM t1 WHERE y > (SELECT count(z) FROM t2);
```

Usage notes:

If the same table is referenced in both the outer and inner query blocks, construct a table alias in the outer query block and use a fully qualified name to distinguish the inner and outer table references:

```
SELECT * FROM t1 one WHERE id IN (SELECT parent FROM t1 two WHERE t1.parent
= t2.id);
```

The STRAIGHT_JOIN hint affects the join order of table references in the query block containing the hint. It does not affect the join order of nested queries, such as views, inline views, or WHERE-clause subqueries. To use this hint for performance tuning of complex queries, apply the hint to all query blocks that need a fixed join order.

Internal details:

Internally, subqueries involving IN, NOT IN, EXISTS, or NOT EXISTS clauses are rewritten into join queries. Depending on the syntax, the subquery might be rewritten to an outer join, semi join, cross join, or anti join.

A query is processed differently depending on whether the subquery calls any aggregation functions. There are correlated and uncorrelated forms, with and without calls to aggregation functions. Each of these four categories is rewritten differently.

Column statistics considerations:

Because queries that include correlated and uncorrelated subqueries in the WHERE clause are written into join queries, to achieve best performance, follow the same guidelines for running the COMPUTE STATS statement as you do for tables involved in regular join queries. Run the COMPUTE STATS statement for each associated tables after loading or substantially changing the data in that table.

Restrictions:

The initial Impala support for nested subqueries addresses the most common use cases. Some restrictions remain:

- Although you can use subqueries in a query involving UNION or UNION ALL in Impala 2.1.0 and higher, currently you cannot construct a union of two subqueries (for example, in the argument of an IN or EXISTS operator).
- Subqueries returning scalar values cannot be used with the operators ANY or ALL. (Impala does not currently have a SOME operator, but if it did, the same restriction would apply.)
- For the EXISTS and NOT EXISTS clauses, any subquery comparing values from the outer query block to another table must use at least one equality comparison, not exclusively other kinds of comparisons such as less than, greater than, BETWEEN, or !=.
-
- Expressions inside a subquery, for example in the WHERE clause, can use OR conjunctions but the OR conjunction cannot be combined with more than one subquery.
- Scalar subqueries are only supported in numeric contexts.
- You cannot use subqueries with the CASE function to generate the comparison value, the values to be compared against, or the return value.
- Only uncorrelated subqueries are supported in the filter condition for the HAVING clause. Similarly only a SELE CT uncorrelated scalar subquery is supported.
- You must use a fully qualified name (*TABLE_NAME.COLUMN_NAME* or *DATABASE_NAME.TABLE_NAME.COLUMN_NAME*) when referring to any column from the outer query block within a subquery.
- The TABLESAMPLE clause of the SELECT statement does not apply to a table reference derived from a view, a subquery, or anything other than a real base table. This clause only works for tables backed by HDFS or HDFS-like data files, therefore it does not apply to Kudu or HBase tables.

Complex type considerations:

For the complex types (ARRAY, STRUCT, and MAP) available in Impala 2.3 and higher, the join queries that “unpack” complex type columns often use correlated subqueries in the FROM clause. For example, if the first table in the join clause is CUSTOMER, the second join clause might have a subquery that selects from the column CUST OMER.C_ORDERS, which is an ARRAY. The subquery re-evaluates the ARRAY elements corresponding to each row from the CUSTOMER table.

Related Information

Table and column statistics

TABLESAMPLE clause

Specify the TABLESAMPLE clause when you need to explore the data distribution within the table, the table is very large, and it is impractical or unnecessary to process all the data from the table or selected partitions.

The clause makes the query process a randomized set of data files from the table, so that the total volume of data is greater than or equal to the specified percentage of data bytes within that table. (Or the data bytes within the set of partitions that remain after partition pruning is performed.)

Syntax:

```
TABLESAMPLE SYSTEM(PERCENTAGE) [REPEATABLE(SEED)]
```

The TABLESAMPLE clause comes immediately after a table name or table alias.

The SYSTEM keyword represents the sampling method. Currently, Impala only supports a single sampling method named SYSTEM.

The *PERCENTAGE* argument is an integer literal from 0 to 100. A percentage of 0 produces an empty result set for a particular table reference, while a percentage of 100 uses the entire contents. Because the sampling works by selecting a random set of data files, the proportion of sampled data from the table may be greater than the specified percentage, based on the number and sizes of the underlying data files. See the usage notes for details.

The optional REPEATABLE keyword lets you specify an arbitrary positive integer seed value that ensures that when the query is run again, the sampling selects the same set of data files each time. REPEATABLE does not have a default value. If you omit the REPEATABLE keyword, the random seed is derived from the current time.

See the COMPUTE STATS statement for the TABLESAMPLE clause used in the COMPUTE STATS statement.

Usage notes:

You might use this clause with aggregation queries, such as finding the approximate average, minimum, or maximum where exact precision is not required. You can use these findings to plan the most effective strategy for constructing queries against the full table or designing a partitioning strategy for the data.

Some other database systems have a TABLESAMPLE clause. The Impala syntax for this clause is modeled on the syntax for popular relational databases, not the Hive TABLESAMPLE clause. For example, there is no BUCKETS keyword as in Hive SQL.

The precision of the *PERCENTAGE* threshold depends on the number and sizes of the underlying data files. Impala brings in additional data files, one at a time, until the number of bytes exceeds the specified percentage based on the total number of bytes for the entire set of table data. The precision of the percentage threshold is higher when the table contains many data files with consistent sizes. See the code listings later in this section for examples.

When you estimate characteristics of the data distribution based on sampling a percentage of the table data, be aware that the data might be unevenly distributed between different files. Do not assume that the percentage figure reflects the percentage of rows in the table. For example, one file might contain all blank values for a STRING column, while another file contains long strings in that column; therefore, one file could contain many more rows than another. Likewise, a table created with the SORT BY clause might contain narrow ranges of values for the sort columns, making it impractical to extrapolate the number of distinct values for those columns based on sampling only some of the data files.

Because a sample of the table data might not contain all values for a particular column, if the TABLESAMPLE is used in a join query, the key relationships between the tables might produce incomplete result sets compared to joins using all the table data. For example, if you join 50% of table A with 50% of table B, some values in the join columns might not match between the two tables, even though overall there is a 1:1 relationship between the tables.

The REPEATABLE keyword makes identical queries use a consistent set of data files when the query is repeated. You specify an arbitrary integer key that acts as a seed value when Impala randomly selects the set of data files to use in the query. This technique lets you verify correctness, examine performance, and so on for queries using the TABLESAMPLE clause without the sampled data being different each time. The repeatable aspect is reset (that is, the set of selected data files may change) any time the contents of the table change. The statements or operations that can make sampling results non-repeatable are:

- INSERT.
- TRUNCATE TABLE.
- LOAD DATA.
- REFRESH or INVALIDATE METADATA after files are added or removed by a non-Impala mechanism.

This clause is similar in some ways to the LIMIT clause, because both serve to limit the size of the intermediate data and final result set. LIMIT 0 is more efficient than TABLESAMPLE SYSTEM(0) for verifying that a query can run without producing any results. TABLESAMPLE SYSTEM(*N*) often makes query processing more efficient than using a LIMIT clause by itself, because all phases of query execution use less data overall. If the intent is to retrieve some representative values from the table in an efficient way, you might combine TABLESAMPLE, ORDER BY, and LIMIT clauses within a single query.

Partitioning:

When you query a partitioned table, any partition pruning happens before Impala selects the data files to sample. For example, in a table partitioned by year, a query with WHERE year = 2017 and a TABLESAMPLE SYSTEM(10) clause would sample data files representing at least 10% of the bytes present in the 2017 partition.

Amazon S3 considerations:

This clause applies to S3 tables the same way as tables with data files stored on HDFS.

ADLS considerations:

This clause applies to ADLS tables the same way as tables with data files stored on HDFS.

Kudu considerations:

This clause does not apply to Kudu tables.

HBase considerations:

This clause does not apply to HBase tables.

Performance considerations:

From a performance perspective, the TABLESAMPLE clause is especially valuable for exploratory queries on text, Avro, or other file formats other than Parquet. Text-based or row-oriented file formats must process substantial amounts of redundant data for queries that derive aggregate results such as MAX(), MIN(), or AVG() for a single column. Therefore, you might use TABLESAMPLE early in the ETL pipeline, when data is still in raw text format and has not been converted to Parquet or moved into a partitioned table.

Restrictions:

This clause applies only to tables that use a storage layer with underlying raw data files, such as HDFS, Amazon S3, or Microsoft ADLS.

This clause does not apply to table references that represent views. A query that applies the TABLESAMPLE clause to a view or a subquery fails with a semantic error.

Because the sampling works at the level of entire data files, it is by nature coarse-grained. It is possible to specify a small sample percentage but still process a substantial portion of the table data if the table contains relatively few data files, if each data file is very large, or if the data files vary substantially in size. Be sure that you understand the data distribution and physical file layout so that you can verify if the results are suitable for extrapolation. For example, if the table contains only a single data file, the “sample” will consist of all the table data regardless of the percentage you specify. If the table contains data files of 1 GiB, 1 GiB, and 1 KiB, when you specify a sampling percentage of 50 you would either process slightly more than 50% of the table (1 GiB + 1 KiB) or almost the entire table (1 GiB + 1 GiB), depending on which data files were selected for sampling.

If data files are added by a non-Impala mechanism, and the table metadata is not updated by a REFRESH or INVALIDATE METADATA statement, the TABLESAMPLE clause does not consider those new files when computing the number of bytes in the table or selecting which files to sample.

If data files are removed by a non-Impala mechanism, and the table metadata is not updated by a REFRESH or INVALIDATE METADATA statement, the query fails if the TABLESAMPLE clause attempts to reference any of the missing files.

Examples:

The following examples demonstrate the TABLESAMPLE clause. These examples intentionally use very small data sets to illustrate how the number of files, size of each file, and overall size of data in the table interact with the percentage specified in the clause.

These examples use an unpartitioned table, containing several files of roughly the same size:

```
create table sample_demo (x int, s string);

insert into sample_demo values (1, 'one');
insert into sample_demo values (2, 'two');
```

```
insert into sample_demo values (3, 'three');
insert into sample_demo values (4, 'four');
insert into sample_demo values (5, 'five');

show files in sample_demo;
+---------------------+------+-----------+
| Path                | Size | Partition |
+---------------------+------+-----------+
| 991213608_data.0.   | 7B   |           |
| 982196806_data.0.   | 6B   |           |
| _2122096884_data.0. | 8B   |           |
| _586325431_data.0.  | 6B   |           |
| 1894746258_data.0.  | 7B   |           |
+---------------------+------+-----------+
show table stats sample_demo;
+-------+--------+------+--------+-------------------------+
| #Rows | #Files | Size | Format | Location                |
+-------+--------+------+--------+-------------------------+
| -1    | 5      | 34B  | TEXT   | /tsample.db/sample_demo |
+-------+--------+------+--------+-------------------------+
```

A query that samples 50% of the table must process at least 17 bytes of data. Based on the sizes of the data files, we can predict that each such query uses 3 arbitrary files. Any 1 or 2 files are not enough to reach 50% of the total data in the table (34 bytes), so the query adds more files until it passes the 50% threshold:

```
select distinct x from sample_demo tablesample system(50);
+---+
| x |
+---+
| 4 |
| 1 |
| 5 |
+---+

select distinct x from sample_demo tablesample system(50);
+---+
| x |
+---+
| 5 |
| 4 |
| 2 |
+---+

select distinct x from sample_demo tablesample system(50);
+---+
| x |
+---+
| 5 |
| 3 |
| 2 |
+---+
```

To help run reproducible experiments, the REPEATABLE clause causes Impala to choose the same set of files for each query. Although the data set being considered is deterministic, the order of results varies (in the absence of an ORDER BY clause) because of the way distributed queries are processed:

```
select distinct x from sample_demo
  tablesample system(50) repeatable (12345);
+---+
| x |
+---+
| 3 |
```

```
| 2 |
| 1 |
+---+

select distinct x from sample_demo
  tablesample system(50) repeatable (12345);
+---+
| x |
+---+
| 2 |
| 1 |
| 3 |
+---+
```

The following examples show how uneven data distribution affects which data is sampled. Adding another data file containing a long string value changes the threshold for 50% of the total data in the table:

```
insert into sample_demo values
(1000, 'Boyhood is the longest time in life for a boy. The last term of the
 school-year is made of decades, not of weeks, and living through them is
like waiting for the millennium. Booth Tarkington');

show files in sample_demo;
+---------------------+------+-----------+
| Path                | Size | Partition |
+---------------------+------+-----------+
| 991213608_data.0.   | 7B   |           |
| 982196806_data.0.   | 6B   |           |
| _253317650_data.0.  | 196B |           |
| _2122096884_data.0. | 8B   |           |
| _586325431_data.0.  | 6B   |           |
| 1894746258_data.0.  | 7B   |           |
+---------------------+------+-----------+

show table stats sample_demo;
+-------+--------+------+--------+-------------------------+
| #Rows | #Files | Size | Format | Location                |
+-------+--------+------+--------+-------------------------+
| -1    | 6      | 230B | TEXT   | /tsample.db/sample_demo |
+-------+--------+------+--------+-------------------------+
```

Even though the queries do not refer to the S column containing the long value, all the sampling queries include the data file containing the column value X=1000, because the query cannot reach the 50% threshold (115 bytes) without including that file. The large file might be considered first, in which case it is the only file processed by the query. Or an arbitrary set of other files might be considered first.

```
select distinct x from sample_demo tablesample system(50);
+------+
| x    |
+------+
| 1000 |
| 3    |
| 1    |
+------+

select distinct x from sample_demo tablesample system(50);
+------+
| x    |
+------+
| 1000 |
+------+
```

```
select distinct x from sample_demo tablesample system(50);
+------+
| x    |
+------+
| 1000 |
| 4    |
| 2    |
| 1    |
+------+
```

The following examples demonstrate how the TABLESAMPLE clause interacts with other table aspects, such as partitioning and file format:

```
create table sample_demo_partitions (x int, s string) partitioned by (n int)
 stored as parquet;
insert into sample_demo_partitions partition (n = 1) select * from sample_d
emo;
insert into sample_demo_partitions partition (n = 2) select * from sample_d
emo;
insert into sample_demo_partitions partition (n = 3) select * from sample_d
emo;

show files in sample_demo_partitions;
+--------------------------------+--------+-----------+
| Path                           | Size   | Partition |
+--------------------------------+--------+-----------+
| 000000_364262785_data.0.parq   | 1.24KB | n=1       |
| 000001_973526736_data.0.parq   | 566B   | n=1       |
| 0000000_1300598134_data.0.parq | 1.24KB | n=2       |
| 0000001_689099063_data.0.parq  | 568B   | n=2       |
| 0000000_1861371709_data.0.parq | 1.24KB | n=3       |
| 0000001_1065507912_data.0.parq | 566B   | n=3       |
+--------------------------------+--------+-----------+
show table stats tablesample_demo_partitioned;
+-------+-------+--------+--------+---------+------------------------------
---------------+
| n     | #Rows | #Files | Size   | Format  | Location
               |
+-------+-------+--------+--------+---------+----------------------------
-----------------+
| 1     | -1    | 2      | 1.79KB | PARQUET | /tsample.db/tablesample_dem
o_partitioned/n=1 |
| 2     | -1    | 2      | 1.80KB | PARQUET | /tsample.db/tablesample_demo_p
artitioned/n=2 |
| 3     | -1    | 2      | 1.79KB | PARQUET | /tsample.db/tablesample_demo_
partitioned/n=3 |
| Total | -1    | 6      | 5.39KB |         |
               |
+-------+-------+--------+--------+---------+---------------------------
------------------+
```

If the query does not involve any partition pruning, the sampling applies to the data volume of the entire table:

```
-- 18 rows total.
select count(*) from sample_demo_partitions;
+----------+
| count(*) |
+----------+
| 18       |
+----------+
-- The number of rows per data file is not
-- perfectly balanced, therefore the count
-- is different depending on which set of files
```

```
-- is considered.
select count(*) from sample_demo_partitions
  tablesample system(75);
+----------+
| count(*) |
+----------+
| 14       |
+----------+
select count(*) from sample_demo_partitions
  tablesample system(75);
+----------+
| count(*) |
+----------+
| 16       |
+----------+
```

If the query only processes certain partitions, the query computes the sampling threshold based on the data size and set of files only from the relevant partitions:

```
select count(*) from sample_demo_partitions
  tablesample system(50) where n = 1;
+----------+
| count(*) |
+----------+
| 6        |
+----------+

select count(*) from sample_demo_partitions
  tablesample system(50) where n = 1;
+----------+
| count(*) |
+----------+
| 2        |
+----------+
```

Related Information

COMPUTE STATS statement

WITH clause

A clause that can be added before a SELECT statement, to define aliases for complicated expressions that are referenced multiple times within the body of the SELECT. Similar to CREATE VIEW, except that the table and column names defined in the WITH clause do not persist after the query finishes, and do not conflict with names used in actual tables or views. Also known as “subquery factoring”.

You can rewrite a query using subqueries to work the same as with the WITH clause. The purposes of the WITH clause are:

- Convenience and ease of maintenance from less repetition with the body of the query. Typically used with queries involving UNION, joins, or aggregation functions where the similar complicated expressions are referenced multiple times.
- SQL code that is easier to read and understand by abstracting the most complex part of the query into a separate block.
- Improved compatibility with SQL from other database systems that support the same clause (primarily Oracle Database).

Note:

The Impala WITH clause does not support recursive queries in the WITH, which is supported in some other database systems.

Standards compliance: Introduced in SQL:1999.

Examples:

```
-- Define 2 subqueries that can be referenced from the body of a longer quer
y.
with t1 as (select 1), t2 as (select 2) insert into tab select * from t1
union all select * from t2;
-- Define one subquery at the outer level, and another at the inner level as
 part of the
-- initial stage of the UNION ALL query.
with t1 as (select 1) (with t2 as (select 2) select * from t2) union all
select * from t1;
```

DISTINCT operator

The DISTINCT operator in a SELECT statement filters the result set to remove duplicates.

```
-- Returns the unique values from one column.
-- NULL is included in the set of values if any rows have a NULL in this c
olumn.
SELECT DISTINCT c_birth_country FROM customer;

-- Returns the unique combinations of values from multiple columns.
SELECT DISTINCT c_salutation, c_last_name FROM customer;
```

You can use DISTINCT in combination with an aggregation function, typically COUNT(), to find how many different values a column contains.

```
-- Counts the unique values from one column.
-- NULL is not included as a distinct value in the count.
SELECT COUNT(DISTINCT c_birth_country) FROM customer;

-- Counts the unique combinations of values from multiple columns.
SELECT COUNT(DISTINCT c_salutation, c_last_name) FROM customer;
```

Zero-length strings: For purposes of clauses such as DISTINCT and GROUP BY, Impala considers zero-length strings (""), NULL, and space to all be different values.

Note:

In contrast with some database systems that always return DISTINCT values in sorted order, Impala does not do any ordering of DISTINCT values. Always include an ORDER BY clause if you need the values in alphabetical or numeric sorted order.

SET statement

The SET statement specifies values for query options that control the runtime behavior of other statements within the same session.

When issued in impala-shell, the SET command is interpreted as an impala-shell command that has differences from the SQL SET statement.

Syntax:

```
SET
SET ALL
SET QUERY_OPTION=OPTION_VALUE
SET QUERY_OPTION=""
```

SET and SET ALL with no arguments return a result set consisting of all the applicable query options and their current values.

The *QUERY_OPTION* and *OPTION_VALUE* are case-insensitive.

Unlike the impala-shell command version of SET, when used as a SQL statement, the string values for *OPTION_VALUE* need to be quoted, e.g. SET option="new_value".

The SET *QUERY_OPTION* = "" statement unsets the value of the *QUERY_OPTION* in the current session, reverting it to the default state. In impala-shell, use the UNSET command to set a query option to its default.

Each query option has a specific allowed notation for its arguments. See *Query options for the SET statement* for the details of each query option.

Usage notes:

The options are divided into groups: Regular Query Options, Advanced Query Options, Development Query Op tions, and Deprecated Query Options.

- The advanced options are intended for use in specific kinds of performance tuning and debugging scenarios.
- The development options are related to internal development of Impala or features that are not yet finalized. These options might be changed or removed without notice.
- The deprecated options are related to features that are removed or changed so that the options no longer have any purpose. These options might be removed in future versions.

- By default, only the first two groups, regular and advanced, are displayed by the SET command. Use SET ALL to see all groups of options.
- `impala-shell` options and user-specified variables are always displayed at the end of the list of query options, after all appropriate option groups.

SET has always been available as an `impala-shell` command. Promoting it to a SQL statement lets you use this feature in client applications through the JDBC and ODBC APIs.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Cloudera Manager: You can set any of the query options globally in Cloudera Manager to affect all the queries in the cluster.

1. Navigate to *IMPALA CLUSTER* > Configuration > Impala Daemon Query Options Advanced Configuration Snippet (Safety Valve)
2. In the field, type a key-value pair of a query option and the value, e.g. EXPLAIN_LEVEL=2.
3. Click + to enter an additional option.
4. Clive Save Changes.

Related Information

Query options

SET command in impala-shell

SHOW statement

Use the SHOW statement to get information about different types of Impala objects.

Syntax:

```
SHOW DATABASES [[LIKE] 'PATTERN']
SHOW SCHEMAS [[LIKE] 'PATTERN'] - an alias for SHOW DATABASES
SHOW TABLES [IN DATABASE_NAME] [[LIKE] 'PATTERN']
SHOW [AGGREGATE | ANALYTIC] FUNCTIONS [IN DATABASE_NAME] [[LIKE] 'PATTERN']
SHOW CREATE TABLE [DATABASE_NAME].TABLE_NAME
SHOW CREATE VIEW [DATABASE_NAME].VIEW_NAME
SHOW TABLE STATS [DATABASE_NAME.]TABLE_NAME
SHOW COLUMN STATS [DATABASE_NAME.]TABLE_NAME
```

```
SHOW [RANGE] PARTITIONS [DATABASE_NAME.]TABLE_NAME
SHOW FILES IN [DATABASE_NAME.]TABLE_NAME [PARTITION (KEY_COL_EXPRESSION
 [, KEY_COL_EXPRESSION]]
SHOW VIEWS IN functional LIKE 'alltypes*'
SHOW VIEWS [IN database_name] [[LIKE] 'pattern']

SHOW GRANT USER USER_NAME ON SERVER
SHOW GRANT USER USER_NAME ON DATABASE DATABASE_NAME
SHOW GRANT USER USER_NAME ON TABLE TABLE_NAME
SHOW GRANT USER USER_NAME ON TABLE [DATABASE_NAME].TABLE_NAME.COLUMN_NAME
SHOW GRANT USER USER_NAME ON URI URI


SHOW GRANT GROUP GROUP_NAME ON SERVER
SHOW GRANT GROUP GROUP_NAME ON DATABASE DATABASE_NAME
SHOW GRANT GROUP GROUP_NAME ON TABLE TABLE_NAME
SHOW GRANT GROUP GROUP_NAME ON TABLE [DATABASE_NAME].TABLE_NAME.COLUMN_NAME
SHOW GRANT GROUP GROUP_NAME ON URI URI
```

Note: SHOW GRANT USER/GROUP is not supported without specifying a resource.

Issue a SHOW *OBJECT_TYPE* statement to see the appropriate objects in the current database, or SHOW *OBJECT_TYPE* IN *DATABASE_NAME* to see objects in a specific database.

The optional *PATTERN* argument is a quoted string literal, using Unix-style * wildcards and allowing | for alternation. The preceding LIKE keyword is also optional. All object names are stored in lowercase, so use all lowercase letters in the pattern string. For example:

```
show databases 'a*';
show databases like 'a*';
show tables in some_db like '*fact*';
use some_db;
show tables '*dim*|*fact*';
```

Cancellation: Cannot be cancelled.

SHOW FILES statement

The SHOW FILES statement displays the files that constitute a specified table, or a partition within a partitioned table. This syntax is available in Impala 2.2 and higher only. The output includes the names of the files, the size of each file, and the applicable partition for a partitioned table. The size includes a suffix of B for bytes, MB for megabytes, and GB for gigabytes.

In Impala 2.8 and higher, you can use general expressions with operators such as <, IN, LIKE, and BETWEEN in the PARTITION clause, instead of only equality operators. For example:

```
show files in sample_table partition (j < 5);
show files in sample_table partition (k = 3, l between 1 and 10);
show files in sample_table partition (month like 'J%');
```

Note: This statement applies to tables and partitions stored on HDFS, or in the Amazon Simple Storage System (S3). It does not apply to views. It does not apply to tables mapped onto HBase or Kudu, because those data management systems do not use the same file-based storage layout.

Usage notes:

You can use this statement to verify the results of your ETL process: that is, that the expected files are present, with the expected sizes. You can examine the file information to detect conditions such as empty files, missing files, or inefficient layouts due to a large number of small files. When you use INSERT statements to copy from one table to another, you can see how the file layout changes due to file format conversions, compaction of small input files into large data blocks, and multiple output files from parallel queries and partitioned inserts.

The output from this statement does not include files that Impala considers to be hidden or invisible, such as those whose names start with a dot or an underscore, or that end with the suffixes .copying or .tmp.

The information for partitioned tables complements the output of the SHOW PARTITIONS statement, which summarizes information about each partition. SHOW PARTITIONS produces some output for each partition, while SHOW FILES does not produce any output for empty partitions because they do not include any data files.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read permission for all the table files, read and execute permission for all the directories that make up the table, and execute permission for the database directory and all its parent directories.

SHOW GRANT USER statement

The SHOW GRANT USER statement shows the list of privileges for a given user. This statement is only allowed for administrative users. However, the current user can run SHOW GRANT USER for themselves.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

SHOW DATABASES

The SHOW DATABASES statement is often the first one you issue when connecting to an instance for the first time. You typically issue SHOW DATABASES to see the names you can specify in a USE *DB_NAME* statement, then after switching to a database you issue SHOW TABLES to see the names you can specify in SELECT and INSERT statements.

In Impala 2.5 and higher, the output includes a second column showing any associated comment for each database.

The output of SHOW DATABASES includes the special _impala_builtins database, which lets you view definitions of built-in functions, as described under SHOW FUNCTIONS.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

SHOW TABLES statement

Displays the names of tables. By default, lists tables in the current database, or with the IN clause, in a specified database. By default, lists all tables, or with the LIKE clause, only those whose name match a pattern with * wildcards.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all directories that are part of the table. (A table could span multiple different HDFS directories if it is partitioned. The directories could be widely scattered because a partition can reside in an arbitrary HDFS directory based on its LOCATION attribute.)

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

SHOW CREATE TABLE statement

As a schema changes over time, you might run a CREATE TABLE statement followed by several ALTER TABLE statements. To capture the cumulative effect of all those statements, SHOW CREATE TABLE displays a CREATE TABLE statement that would reproduce the current structure of a table. You can use this output in scripts that set up or clone a group of tables, rather than trying to reproduce the original sequence of CREATE TABLE and ALTER TA BLE statements. When creating variations on the original table, or cloning the original table on a different system, you might need to edit the SHOW CREATE TABLE output to change things such as the database name, LOCATION field, and so on that might be different on the destination system.

If you specify a view name in the SHOW CREATE TABLE, it returns a CREATE VIEW statement with column names and the original SQL statement to reproduce the view. You need the VIEW_METADATA privilege on the view and SELECT privilege on all underlying views and tables to successfully run the SHOW CREATE VIEW statement for a view. The SHOW CREATE VIEW is available as an alias for SHOW CREATE TABLE.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Kudu considerations:

For Kudu tables:

- The column specifications include attributes such as NULL, NOT NULL, ENCODING, and COMPRESSION. If you do not specify those attributes in the original CREATE TABLE statement, the SHOW CREATE TABLE output displays the defaults that were used.
- The specifications of any RANGE clauses are not displayed in full. To see the definition of the range clauses for a Kudu table, use the SHOW RANGE PARTITIONS statement.
- The TBLPROPERTIES output reflects the Kudu master address and the internal Kudu name associated with the Impala table.

SHOW CREATE VIEW statement

The SHOW CREATE VIEW, it returns a CREATE VIEW statement with column names and the original SQL statement to reproduce the view. You need the VIEW_METADATA privilege on the view and SELECT privilege on all underlying views and tables to successfully run the SHOW CREATE VIEW statement for a view.

The SHOW CREATE VIEW is an alias for SHOW CREATE TABLE.

SHOW TABLE STATS statement

The SHOW TABLE STATS and SHOW COLUMN STATS variants are important for tuning performance and diagnosing performance issues, especially with the largest tables and the most complex join queries.

Any values that are not available (because the COMPUTE STATS statement has not been run yet) are displayed as -1.

SHOW TABLE STATS provides some general information about the table, such as the number of files, overall size of the data, whether some or all of the data is in the HDFS cache, and the file format, that is useful whether or not you have run the COMPUTE STATS statement. A -1 in the #Rows output column indicates that the COMPUTE ST ATS statement has never been run for this table. If the table is partitioned, SHOW TABLE STATS provides this information for each partition. (It produces the same output as the SHOW PARTITIONS statement in this case.)

The output of SHOW COLUMN STATS is primarily only useful after the COMPUTE STATS statement has been run on the table. A -1 in the #Distinct Values output column indicates that the COMPUTE STATS statement has never been run for this table. Currently, Impala always leaves the #Nulls column as -1, even after COMPUTE STATS has been run.

These SHOW statements work on actual tables only, not on views.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

Kudu considerations:

Because Kudu tables do not have characteristics derived from HDFS, such as number of files, file format, and HDFS cache status, the output of SHOW TABLE STATS reflects different characteristics that apply to Kudu tables. If the Kudu table is created with the clause PARTITIONS 20, then the result set of SHOW TABLE STATS consists of 20 rows, each representing one of the numbered partitions.

Impala does not compute the number of rows for each partition for Kudu tables. Therefore, you do not need to re-run COMPUTE STATS when you see -1 in the # Rows column of the output from SHOW TABLE STATS. That column always shows -1 for all Kudu tables.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all directories that are part of the table. (A table could span multiple different HDFS directories if it is partitioned. The directories could be widely scattered because a partition can reside in an arbitrary HDFS directory based on its LOCATION attribute.) The Impala user must also have execute permission for the database directory, and any parent directories of the database directory in HDFS.

SHOW COLUMN STATS statement

The SHOW TABLE STATS and SHOW COLUMN STATS variants are important for tuning performance and diagnosing performance issues, especially with the largest tables and the most complex join queries.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

Kudu considerations:

The output for SHOW COLUMN STATS includes the relevant information for Kudu tables. The information for column statistics that originates in the underlying Kudu storage layer is also represented in the metastore database that Impala uses.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all directories that are part of the table. (A table could span multiple different HDFS directories if it is partitioned. The directories could be widely scattered because a partition can reside in an arbitrary HDFS directory based on its

LOCATION attribute.) The Impala user must also have execute permission for the database directory, and any parent directories of the database directory in HDFS.

SHOW PARTITIONS statement

SHOW PARTITIONS displays information about each partition for a partitioned table. (The output is the same as the SHOW TABLE STATS statement, but SHOW PARTITIONS only works on a partitioned table.) Because it displays table statistics for all partitions, the output is more informative if you have run the COMPUTE STATS statement after creating all the partitions. For example, on a CENSUS table partitioned on the YEAR column:

Because Kudu tables are all considered to be partitioned, the SHOW PARTITIONS statement works for any Kudu table. The default output is the same as for SHOW TABLE STATS, with the same Kudu-specific columns in the result set.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

Kudu considerations:

The optional RANGE clause only applies to Kudu tables. It displays only the partitions defined by the RANGE clause of CREATE TABLE or ALTER TABLE.

Although you can specify < or <= comparison operators when defining range partitions for Kudu tables, Kudu rewrites them if necessary to represent each range as *LOW_BOUND* <= VALUES < *HIGH_BOUND*. This rewriting might involve incrementing one of the boundary values or appending a \0 for string values, so that the partition covers the same range as originally specified.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have read and execute permissions for all directories that are part of the table. (A table could span multiple different HDFS directories if it is partitioned. The directories could be widely scattered because a partition can reside in an arbitrary HDFS directory based on its LOCATION attribute.) The Impala user must also have execute permission for the database directory, and any parent directories of the database directory in HDFS.

SHOW FUNCTIONS statement

By default, SHOW FUNCTIONS displays user-defined functions (UDFs) and SHOW AGGREGATE FUNCTIONS displays user-defined aggregate functions (UDAFs) associated with a particular database. The output from SHOW FUNCTIONS includes the argument signature of each function. You specify this argument signature as part of the DROP FUNCTION statement. You might have several UDFs with the same name, each accepting different argument data types.

Usage notes:

In Impala 2.5 and higher, the SHOW FUNCTIONS output includes a new column, labelled is persistent. This property is true for Impala built-in functions, C++ UDFs, and Java UDFs created using the new CREATE FUNCTION syntax with no signature. It is false for Java UDFs created using the old CREATE FUNCTION syntax that includes the types for the arguments and return value. Any functions with false shown for this property must be created again by the CREATE FUNCTION statement each time the Impala catalog server is restarted. See CREATE FUNCTION for information on switching to the new syntax, so that Java UDFs are preserved across restarts. Java UDFs that are persisted this way are also easier to share across Impala and Hive.

Security considerations:

When authorization is enabled, the output of the SHOW statement only shows those objects for which you have the privilege to view. If you believe an object exists but you cannot see it in the SHOW output, check with the system administrator if you need to be granted a new privilege for that object.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

SHOW ROLES statement

The SHOW ROLES statement displays all the roles created in the Ranger server.

Syntax

```
SHOW ROLES
```

Note: Only a Ranger administrator is allowed to run this statement.

SHOW CURRENT ROLES statement

The SHOW CURRENT ROLES statement displays the roles assigned to the groups of the requesting user.

Syntax

```
SHOW ROLES
```

SHOW ROLE GRANT GROUP statement

The SHOW ROLE GRANT GROUP statement displays the roles assigned to the group.

Syntax

```
SHOW ROLE GRANT GROUP GROUP_NAME
```

Note: To run this statement, the requesting user must belong to the group <group>, or the requesting user has to be a Ranger administrator.

SHUTDOWN statement

The SHUTDOWN statement performs a graceful shutdown of Impala Daemon. The Impala daemon will notify other Impala daemons that it is shutting down, wait for a grace period, then shut itself down once no more queries or fragments are executing on that daemon. The --shutdown_grace_period_s flag determines the duration of the grace period in seconds.

Syntax:

```
:SHUTDOWN()
:SHUTDOWN([HOST_NAME[:PORT_NUMBER] )
:SHUTDOWN(DEADLINE)
:SHUTDOWN([HOST_NAME[:PORT_NUMBER], DEADLINE)
```

Usage notes:

All arguments are optional for SHUTDOWN.

Argument	Type	Default	Description
HOST_NAME	STRING	The current impalad host to whom the SHUTDOWN statement is submitted.	Address of the impalad to be shut down.
PORT_NUMBER	INT	• In Impala 3.1, the current impalad's port used for the thrift based communication with other impalads (by default, 22000). • In Impala 3.2 and higher, the current impalad's port used for the KRPC based communication with other impalads (by default, 27000).	Specifies the port by which the impalad can be contacted. • In Impala 3.1, use the same impalad port used for the thrift based inter-Impala communication. • In Impala 3.2 and higher, use the same impalad port used for the KRPC based inter-Impala communication.
DEADLINE	INT	The value of the ##shutdown_deadline_s flag, which defaults to 1 hour.	*DEADLINE* must be a non-negative number, specified in seconds. The value, 0, for *DEADLINE* specifies an immediate shutdown.

Take the following points into consideration when running the SHUTDOWN statement:

- A client can shut down the coordinator impalad that it is connected to via :SHUTDOWN().
- A client can remotely shut down any impalad via :SHUTDOWN('*HOSTNAME*').
- The shutdown time limit can be overridden to force a quicker or slower shutdown by specifying a deadline. The default deadline is determined by the --shutdown_deadline_s flag, which defaults to 1 hour.
- The executors can be shut down without disrupting running queries. Short-running queries will finish, and long-running queries will continue until the *DEADLINE* is reached.
- If queries are submitted to a coordinator after shutdown of that coordinator has started, they will fail.
- Long running queries or other issues, such as stuck fragments, will slow down but not prevent eventual shutdown.

Security considerations:

The ALL privilege is required on the server.

Cancellation: Cannot be cancelled.

Examples:

```
:SHUTDOWN(); -- Shut down the current impalad  with the default deadline.
:SHUTDOWN('hostname'); --  Shut down impalad running on hostname  with the
 default deadline.
:SHUTDOWN(\"hostname:1234\"); -- Shut down impalad running on host at port
 1234  with the default deadline.
:SHUTDOWN(10); - Shut down the current impalad after 10 seconds.
:SHUTDOWN('hostname', 10); - Shut down impalad running on hostname when all
queries running on hostname finish, or after 10 seconds.
:SHUTDOWN('hostname:11', 10 * 60); -- Shut down impalad running on hostname
 at port 11 when all queries running on hostname finish, or after 600 second
s.
:SHUTDOWN(0); -- Perform an immdediate shutdown of the current impalad.
```

Added in: Impala 3.1

TRUNCATE TABLE statement

The TRUNCATE TABLE statement removes the data from an Impala table while leaving the table itself.

Syntax:

```
TRUNCATE [TABLE] [IF EXISTS] [DB_NAME.]TABLE_NAME
```

Statement type: DDL

Usage notes:

Often used to empty tables that are used during ETL cycles, after the data has been copied to another table for the next stage of processing. This statement is a low-overhead alternative to dropping and recreating the table, or using INSERT OVERWRITE to replace the data during the next ETL cycle.

This statement removes all the data and associated data files in the table. It can remove data files from internal tables, external tables, partitioned tables, and tables mapped to HBase or the Amazon Simple Storage Service (S3). The data removal applies to the entire table, including all partitions of a partitioned table.

Any statistics produced by the COMPUTE STATS statement are reset when the data is removed.

Make sure that you are in the correct database before truncating a table, either by issuing a USE statement first or by using a fully qualified name *DB_NAME.TABLE_NAME*.

The optional TABLE keyword does not affect the behavior of the statement.

The optional IF EXISTS clause makes the statement succeed whether or not the table exists. If the table does exist, it is truncated; if it does not exist, the statement has no effect. This capability is useful in standardized setup scripts that are might be run both before and after some of the tables exist. This clause is available in Impala 2.5 and higher.

Amazon S3 considerations:

Although Impala cannot write new data to a table stored in the Amazon S3 filesystem, the TRUNCATE TABLE statement can remove data files from S3.

Cancellation: Cannot be cancelled.

HDFS permissions:

The user ID that the `impalad` daemon runs under, typically the impala user, must have write permission for all the files and directories that make up the table.

Kudu considerations:

Currently, the TRUNCATE TABLE statement cannot be used with Kudu tables.

Examples:

The following example shows a table containing some data and with table and column statistics. After the TRUN CATE TABLE statement, the data is removed and the statistics are reset.

```
CREATE TABLE truncate_demo (x INT);
INSERT INTO truncate_demo VALUES (1), (2), (4), (8);
SELECT COUNT(*) FROM truncate_demo;
+----------+
| count(*) |
+----------+
| 4        |
+----------+
COMPUTE STATS truncate_demo;
+------------------------------------------+
| summary                                  |
+------------------------------------------+
```

```
| Updated 1 partition(s) and 1 column(s). |
+-----------------------------------------+
SHOW TABLE STATS truncate_demo;
+-------+--------+------+--------------+-------------------+--------+----
---------------+
| #Rows | #Files | Size | Bytes Cached | Cache Replication | Format | Incre
mental stats |
+-------+--------+------+--------------+-------------------+--------+----
---------------+
| 4     | 1      | 8B   | NOT CACHED   | NOT CACHED        | TEXT   | false
           |
+-------+--------+------+--------------+-------------------+--------+----
---------------+
SHOW COLUMN STATS truncate_demo;
+--------+------+------------------+--------+----------+----------+
| Column | Type | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+------+------------------+--------+----------+----------+
| x      | INT  | 4                | -1     | 4        | 4        |
+--------+------+------------------+--------+----------+----------+

-- After this statement, the data and the table/column stats will be gone.
TRUNCATE TABLE truncate_demo;
SELECT COUNT(*) FROM truncate_demo;
+----------+
| count(*) |
+----------+
| 0        |
+----------+
SHOW TABLE STATS truncate_demo;
+-------+--------+------+--------------+-------------------+--------+-----
--------------+
| #Rows | #Files | Size | Bytes Cached | Cache Replication | Format | Increm
ental stats |
+-------+--------+------+--------------+-------------------+--------+-----
--------------+
| -1    | 0      | 0B   | NOT CACHED   | NOT CACHED        | TEXT   | false
           |
+-------+--------+------+--------------+-------------------+--------+-----
--------------+
SHOW COLUMN STATS truncate_demo;
+--------+------+------------------+--------+----------+----------+
| Column | Type | #Distinct Values | #Nulls | Max Size | Avg Size |
+--------+------+------------------+--------+----------+----------+
| x      | INT  | -1               | -1     | 4        | 4        |
+--------+------+------------------+--------+----------+----------+
```

The following example shows how the IF EXISTS clause allows the TRUNCATE TABLE statement to be run without error whether or not the table exists:

```
CREATE TABLE staging_table1 (x INT, s STRING);
Fetched 0 row(s) in 0.33s

SHOW TABLES LIKE 'staging*';
+----------------+
| name           |
+----------------+
| staging_table1 |
+----------------+
Fetched 1 row(s) in 0.25s
-- Our ETL process involves removing all data from several staging tables
-- even though some might be already dropped, or not created yet.

TRUNCATE TABLE IF EXISTS staging_table1;
```

```
Fetched 0 row(s) in 5.04s

TRUNCATE TABLE IF EXISTS staging_table2;
Fetched 0 row(s) in 0.25s

TRUNCATE TABLE IF EXISTS staging_table3;
Fetched 0 row(s) in 0.25s
```

TRUNCATE on Insert-Only Transactional Tables

The TRUNCATE table statement on Insert-Only Transactional tables creates new empty ACID base directories and does not remove the files. To maintain transactional isolation TRUNCATE statement will NOT delete the previous base and delta directories.

Related Information

Impala with Amazon S3

UPDATE statement

The UPDATE statement updates an arbitrary number of rows in a Kudu table. This statement only works for Impala tables that use the Kudu storage engine.

Syntax:

```
UPDATE [DATABASE_NAME.]TABLE_NAME SET COL = VAL [, COL = VAL ... ]
  [ FROM JOINED_TABLE_REFS ]
  [ WHERE WHERE_CONDITIONS ]
```

Usage notes:

None of the columns that make up the primary key can be updated by the SET clause.

The conditions in the WHERE clause are the same ones allowed for the SELECT statement.

If the WHERE clause is omitted, all rows in the table are updated.

The conditions in the WHERE clause can refer to any combination of primary key columns or other columns. Referring to primary key columns in the WHERE clause is more efficient than referring to non-primary key columns.

Because Kudu currently does not enforce strong consistency during concurrent DML operations, be aware that the results after this statement finishes might be different than you intuitively expect:

- If some rows cannot be updated because their some primary key columns are not found, due to their being deleted by a concurrent DELETE operation, the statement succeeds but returns a warning.
- An UPDATE statement might also overlap with INSERT, UPDATE, or UPSERT statements running concurrently on the same table. After the statement finishes, there might be more or fewer matching rows than expected in the table because it is undefined whether the UPDATE applies to rows that are inserted or updated while the UPDA TE is in progress.

The number of affected rows is reported in an `impala-shell` message and in the query profile.

The optional FROM clause lets you restrict the updates to only the rows in the specified table that are part of the result set for a join query. The join clauses can include non-Kudu tables, but the table from which the rows are deleted must be a Kudu table.

Statement type: DML

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMPUTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

The following examples show how to perform a simple update on a table, with or without a WHERE clause:

```
-- Set all rows to the same value for column c3.
-- In this case, c1 and c2 are primary key columns
-- and so cannot be updated.
UPDATE kudu_table SET c3 = 'not applicable';

-- Update only the rows that match the condition.
UPDATE kudu_table SET c3 = NULL WHERE c1 > 100 AND c3 IS NULL;

-- Does not update any rows, because the WHERE condition is always false.
UPDATE kudu_table SET c3 = 'impossible' WHERE 1 = 0;

-- Change the values of multiple columns in a single UPDATE statement.
UPDATE kudu_table SET c3 = upper(c3), c4 = FALSE, c5 = 0 WHERE c6 = TRUE;
```

The following examples show how to perform an update using the FROM keyword with a join clause:

```
-- Uppercase a column value, only for rows that have
-- an ID that matches the value from another table.
UPDATE kudu_table SET c3 = upper(c3)
  FROM kudu_table JOIN non_kudu_table
  ON kudu_table.id = non_kudu_table.id;

-- Same effect as previous statement.
-- Assign table aliases in FROM clause, then refer to
-- short names elsewhere in the statement.
UPDATE t1 SET c3 = upper(c3)
  FROM kudu_table t1 JOIN non_kudu_table t2
  ON t1.id = t2.id;

-- Same effect as previous statements, but more efficient.
-- Use WHERE clause to skip updating values that are
-- already uppercase.
UPDATE t1 SET c3 = upper(c3)
  FROM kudu_table t1 JOIN non_kudu_table t2
  ON t1.id = t2.id
  WHERE c3 != upper(c3);
```

UPSERT statement

The UPSERT statement acts as a combination of the INSERT and UPDATE statements.

For each row processed by the UPSERT statement:

- If another row already exists with the same set of primary key values, the other columns are updated to match the values from the row being “UPSERTed”.
- If there is not any row with the same set of primary key values, the row is created, the same as if the INSERT statement was used.

This statement only works for Impala tables that use the Kudu storage engine.

Syntax:

```
UPSERT [HINT_CLAUSE] INTO [TABLE] [DB_NAME.]TABLE_NAME
  [(COLUMN_LIST)]
{
    [HINT_CLAUSE] SELECT_STATEMENT
  | VALUES (VALUE [, VALUE ...]) [, (VALUE [, VALUE ...]) ...]
}

hint_clause ::= [SHUFFLE] | [NOSHUFFLE]
  (Note: the square brackets are part of the syntax.)
```

The *SELECT_STATEMENT* clause can use the full syntax, such as WHERE and JOIN clauses.

Statement type: DML

Usage notes:

If you specify a column list, any omitted columns in the inserted or updated rows are set to their default value (if the column has one) or NULL (if the column does not have a default value). Therefore, if a column is not nullable and has no default value, it must be included in the column list for any UPSERT statement. Because all primary key columns meet these conditions, all the primary key columns must be specified in every UPSERT statement.

Because Kudu tables can efficiently handle small incremental changes, the VALUES clause is more practical to use with Kudu tables than with HDFS-based tables.

Important: After adding or replacing data in a table used in performance-critical queries, issue a COMPUTE STATS statement to make sure all statistics are up-to-date. Consider updating statistics for a table after any INSERT, LOAD DATA, or CREATE TABLE AS SELECT statement in Impala, or after loading data through Hive and doing a REFRESH *TABLE_NAME* in Impala. This technique is especially important for tables that are very large, used in join queries, or both.

Examples:

```
UPSERT INTO kudu_table (pk, c1, c2, c3) VALUES (0, 'hello', 50, true), (1,
'world', -1, false);
UPSERT INTO production_table SELECT * FROM staging_table;
UPSERT INTO production_table SELECT * FROM staging_table WHERE c1 IS NOT
NULL AND c2 > 0;
```

USE statement

The USE statement switches the current session to a specified database. The *current database* is where any CREATE TABLE, INSERT, SELECT, or other statements act when you specify a table or other object name, without prefixing it with a database name. The new current database applies for the duration of the session or unti another USE statement is executed.

Syntax:

```
USE DB_NAME
```

By default, when you connect to an Impala instance, you begin in a database named default.

Usage notes:

Switching the default database is convenient in the following situations:

- To avoid qualifying each reference to a table with the database name. For example, SELECT * FROM t1 JOIN t2 rather than SELECT * FROM db.t1 JOIN db.t2.

- To do a sequence of operations all within the same database, such as creating a table, inserting data, and querying the table.

To start the `impala-shell` interpreter and automatically issue a USE statement for a particular database, specify the option -d *DB_NAME* for the `impala-shell` command. The -d option is useful to run SQL scripts, such as setup or test scripts, against multiple databases without hardcoding a USE statement into the SQL source.

Examples:

See the *CREATE DATABASE* statement for examples covering CREATE DATABASE, USE, and DROP DATABASE.

Cancellation: Cannot be cancelled.

HDFS permissions: This statement does not touch any HDFS files or directories, therefore no HDFS permissions are required.

Related Information

CREATE DATABASE statement

VALUES statement

The VALUES clause can be used as stand-alone statement, in the INSERT statement, and in the SELECT statement to construct a data set.

Syntax:

```
VALUES (ROW)[, (ROW), ...];

SELECT SELECT_LIST FROM (VALUES (ROW)[, (ROW), ...]) AS ALIAS;

ROW ::= COLUMN [[AS ALIAS], COLUMN [AS ALIAS], ...]
```

- The VALUES keyword is followed by a comma separated list of one or more *ROW*s.
- *ROW* is a comma-separated list of one or more *COLUMN*s.
- Each *ROW* must have the same number of *COLUMN*s.
- *COLUMN* can be a constant, a variable, or an expression.
- The corresponding *COLUMN*s must have compatible data types in all *ROW*s. See the third query in the Examples section below.
- By default, the first row is used to name columns. But using the AS keyword, you can optionally give the column an *ALIAS*.
- If used in the SELECT statement, the AS keyword with an *ALIAS* is required.
- *SELECT_LIST* is the columns to be selected for the result set.

Examples:

```
> SELECT * FROM (VALUES(4,5,6),(7,8,9)) AS t;
+---+---+---+
| 4 | 5 | 6 |
+---+---+---+
| 4 | 5 | 6 |
| 7 | 8 | 9 |
+---+---+---+

> SELECT * FROM (VALUES(1 AS c1, true AS c2, 'abc' AS c3),(100,false,'xyz
')) AS t;
+-----+-------+-----+
| c1  | c2    | c3  |
+-----+-------+-----+
```

```
| 1   | true  | abc |
| 100 | false | xyz |
+-----+-------+-----+

> VALUES (CAST('2019-01-01' AS TIMESTAMP)), ('2019-02-02');
+---------------------------------+
| cast('2019-01-01' as timestamp) |
+---------------------------------+
| 2019-01-01 00:00:00             |
| 2019-02-02 00:00:00             |
+---------------------------------+
```

Related Information

SELECT statement

Optimizer hints in Impala

The Impala SQL dialect supports query hints, for fine-tuning the inner workings of queries. Specify hints as a temporary workaround for expensive queries, where missing statistics or other factors cause inefficient performance.

Hints are most often used for the most resource-intensive kinds of Impala queries:

- Join queries involving large tables, where intermediate result sets are transmitted across the network to evaluate the join conditions.
- Inserting into partitioned Parquet tables, where many memory buffers could be allocated on each host to hold intermediate results for each partition.

Syntax:

In Impala 2.0 and higher, you can specify the hints inside comments that use either the /* */ or -- notation. Specify a + symbol immediately before the hint name. Recently added hints are only available using the /* */ and -- notation. For clarity, the /* */ and -- styles are used in the syntax and examples throughout this section. With the /* */ or -- notation for hints, specify a + symbol immediately before the first hint name. Multiple hints can be specified separated by commas, for example /* +clustered,shuffle */

```
SELECT STRAIGHT_JOIN SELECT_LIST FROM
JOIN_LEFT_HAND_TABLE
  JOIN /* +BROADCAST|SHUFFLE */
JOIN_RIGHT_HAND_TABLE
REMAINDER_OF_QUERY;

SELECT SELECT_LIST FROM
JOIN_LEFT_HAND_TABLE
  JOIN -- +BROADCAST|SHUFFLE
JOIN_RIGHT_HAND_TABLE
REMAINDER_OF_QUERY;

INSERT INSERT_CLAUSES
  /* +SHUFFLE|NOSHUFFLE */
  SELECT REMAINDER_OF_QUERY;

INSERT INSERT_CLAUSES
  -- +SHUFFLE|NOSHUFFLE
  SELECT REMAINDER_OF_QUERY;

INSERT /* +SHUFFLE|NOSHUFFLE */
  INSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

INSERT -- +SHUFFLE|NOSHUFFLE
```

```
  INSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

UPSERT /* +SHUFFLE|NOSHUFFLE */
  UPSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

UPSERT -- +SHUFFLE|NOSHUFFLE
  UPSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

SELECT SELECT_LIST
 FROM
TABLE_REF
  /* +{SCHEDULE_CACHE_LOCAL | SCHEDULE_DISK_LOCAL | SCHEDULE_REMOTE}
    [,RANDOM_REPLICA] */
REMAINDER_OF_QUERY;

INSERT INSERT_CLAUSES
  -- +CLUSTERED
  SELECT REMAINDER_OF_QUERY;

INSERT INSERT_CLAUSES
  /* +CLUSTERED */
  SELECT REMAINDER_OF_QUERY;

INSERT -- +CLUSTERED
  INSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

INSERT /* +CLUSTERED */
  INSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

UPSERT -- +CLUSTERED
  UPSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

UPSERT /* +CLUSTERED */
  UPSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;

CREATE /* +SHUFFLE|NOSHUFFLE */
  TABLE_CLAUSES
  AS SELECT REMAINDER_OF_QUERY;

CREATE -- +SHUFFLE|NOSHUFFLE
  TABLE_CLAUSES
  AS SELECT REMAINDER_OF_QUERY;

CREATE /* +CLUSTERED|NOCLUSTERED */
  TABLE_CLAUSES
  AS SELECT REMAINDER_OF_QUERY;

CREATE -- +CLUSTERED|NOCLUSTERED
  TABLE_CLAUSES
  AS SELECT REMAINDER_OF_QUERY;
```

The square bracket style hints are supported for backward compatibility, but the syntax is deprecated and will be removed in a future release. For that reason, any newly added hints are not available with the square bracket syntax.

```
SELECT STRAIGHT_JOIN SELECT_LIST FROM
JOIN_LEFT_HAND_TABLE
```

```
  JOIN [{ /* +BROADCAST */ | /* +SHUFFLE */ }]
JOIN_RIGHT_HAND_TABLE
REMAINDER_OF_QUERY;

INSERT INSERT_CLAUSES
  [{ /* +SHUFFLE */ | /* +NOSHUFFLE */ }]
  [/* +CLUSTERED */]
  SELECT REMAINDER_OF_QUERY;

UPSERT [{ /* +SHUFFLE */ | /* +NOSHUFFLE */ }]
  [/* +CLUSTERED */]
  UPSERT_CLAUSES
  SELECT REMAINDER_OF_QUERY;
```

Usage notes:

With both forms of hint syntax, include the STRAIGHT_JOIN keyword immediately after the SELECT and any DISTINCT or ALL keywords to prevent Impala from reordering the tables in a way that makes the join-related hints ineffective.

The STRAIGHT_JOIN hint affects the join order of table references in the query block containing the hint. It does not affect the join order of nested queries, such as views, inline views, or WHERE-clause subqueries. To use this hint for performance tuning of complex queries, apply the hint to all query blocks that need a fixed join order.

To reduce the need to use hints, run the COMPUTE STATS statement against all tables involved in joins, or used as the source tables for INSERT ... SELECT operations where the destination is a partitioned Parquet table. Do this operation after loading data or making substantial changes to the data within each table. Having up-to-date statistics helps Impala choose more efficient query plans without the need for hinting.

To see which join strategy is used for a particular query, examine the EXPLAIN output for that query.

Hints for join queries:

The /* +BROADCAST */ and /* +SHUFFLE */ hints control the execution strategy for join queries. Specify one of the following constructs immediately after the JOIN keyword in a query:

- /* +SHUFFLE */ makes that join operation use the “partitioned” technique, which divides up corresponding rows from both tables using a hashing algorithm, sending subsets of the rows to other nodes for processing. (The keyword SHUFFLE is used to indicate a “partitioned join”, because that type of join is not related to “partitioned tables”.) Since the alternative “broadcast” join mechanism is the default when table and index statistics are unavailable, you might use this hint for queries where broadcast joins are unsuitable; typically, partitioned joins are more efficient for joins between large tables of similar size.
- /* +BROADCAST */ makes that join operation use the “broadcast” technique that sends the entire contents of the right-hand table to all nodes involved in processing the join. This is the default mode of operation when table and index statistics are unavailable, so you would typically only need it if stale metadata caused Impala to mistakenly choose a partitioned join operation. Typically, broadcast joins are more efficient in cases where one table is much smaller than the other. (Put the smaller table on the right side of the JOIN operator.)

Hints for INSERT ... SELECT and CREATE TABLE AS SELECT (CTAS):

When inserting into partitioned tables, such as using the Parquet file format, you can include a hint in the INSERT or CREATE TABLE AS SELECT(CTAS) statements to fine-tune the overall performance of the operation and its resource usage.

You would only use hints if an INSERT or CTAS into a partitioned table was failing due to capacity limits, or if such an operation was succeeding but with less-than-optimal performance.

- /* +SHUFFLE */ and /* +NOSHUFFLE */ Hints
  - /* +SHUFFLE */ adds an exchange node, before writing the data, which re-partitions the result of the SELE CT based on the partitioning columns of the target table. With this hint, only one node writes to a partition at a time, minimizing the global number of simultaneous writes and the number of memory buffers holding data for individual partitions. This also reduces fragmentation, resulting in fewer files. Thus it reduces overall

resource usage of the INSERT or CTAS operation and allows some operations to succeed that otherwise would fail. It does involve some data transfer between the nodes so that the data files for a particular partition are all written on the same node.

Use /* +SHUFFLE */ in cases where an INSERT or CTAS statement fails or runs inefficiently due to all nodes attempting to write data for all partitions.

If the table is unpartitioned or every partitioning expression is constant, then /* +SHUFFLE */ will cause every write to happen on the coordinator node.

- /* +NOSHUFFLE */ does not add exchange node before inserting to partitioned tables and disables re-partitioning. So the selected execution plan might be faster overall, but might also produce a larger number of small data files or exceed capacity limits, causing the INSERT or CTAS operation to fail.

  Impala automatically uses the /* +SHUFFLE */ method if any partition key column in the source table, mentioned in the SELECT clause, does not have column statistics. In this case, use the /* +NOSHUFFLE */ hint if you want to override this default behavior.
- If column statistics are available for all partition key columns in the source table mentioned in the INSERT ... SELECT or CTAS query, Impala chooses whether to use the /* +SHUFFLE */ or /* +NOSHUFFLE */ technique based on the estimated number of distinct values in those columns and the number of nodes involved in the operation. In this case, you might need the /* +SHUFFLE */ or the /* +NOSHUFFLE */ hint to override the execution plan selected by Impala.

- /* +CLUSTERED */ and /* +NOCLUSTERED */ Hints
  - /* +CLUSTERED */ sorts data by the partition columns before inserting to ensure that only one partition is written at a time per node. Use this hint to reduce the number of files kept open and the number of buffers kept in memory simultaneously. This technique is primarily useful for inserts into Parquet tables, where the large block size requires substantial memory to buffer data for multiple output files at once. This hint is available in Impala 2.8 or higher.

    Starting in Impala 3.0, /* +CLUSTERED */ is the default behavior for HDFS tables.
  - /* +NOCLUSTERED */ does not sort by primary key before insert. This hint is available in Impala 2.8 or higher.

    Use this hint when inserting to Kudu tables.

    In the versions lower than Impala 3.0, /* +NOCLUSTERED */ is the default in HDFS tables.

Hint for Table cardinalities

Currently, as part of query planning, you use the 'COMPUTE STATS' command to compute table stats. Without these stats, a query plan may not be optimal. However, these stats may not be available, up to date, or valid. To workaround this problem, this release adds a new query hint 'TABLE_NUM_ROWS' to be used in cases where stats were missing or invalid. You can use this new hint for a HDFS or a Kudu table in a query as shown in the following example.

```
select col from t /* +TABLE_NUM_ROWS(1000) */;
```

Impala will use this value as table scanned rows when a table has no stats or has corrput stats. This hint value will not be valid if table stats is normal.

Query hints for predicate selectivities

Currently, Impala only uses simple estimation to compute selectivity. For some predicates, the estimation might deviate significantly from the actual value, which leads to a worse query plan.You can now use a new query hint 'SELECTIVITY' to help specify a selectivity value for a predicate.

In the below examples, the parser will interpret expressions wrapped in () followed by a C-style comment /* <predicate hint> */ as a predicate hint. You can use the format +SELECTIVITY(f) for the predicate hint where 'f' is a positive floating point number, in the range of (0, 1], to use as the selectivity for the preceding expression.

Single predicate example:

```
select col from t where (a=1) /* +SELECTIVITY(0.5) */;
```

Compound predicate example:

```
select col from t where (a=1 or b=2) /* +SELECTIVITY(0.5) */;
```

As a limitation of this feature path, the selectivity hints for 'AND' compound predicates, either in the original SQL query or internally generated, are ignored. This may be supported in a future release.

Examples of hints that will be ignored:

- This hint will be ignored because it is used in a predicate with the ‘AND’ compound.

  ```
  select col from t where (a=1 and b=2) /* +SELECTIVITY(0.5) */;
  ```

- The hint used in a 'BETWEEN' predicate will also be ignored since it's converted to 'AND' compound predicate internally.

  ```
  select col from t where (a between 1 and 10) /* +SELECTIVITY(0.5) */;
  ```

Kudu consideration:

Starting from Impala 2.9, the INSERT or UPSERT operations into Kudu tables automatically add an exchange and a sort node to the plan that partitions and sorts the rows according to the partitioning/primary key scheme of the target table (unless the number of rows to be inserted is small enough to trigger single node execution). Since Kudu partitions and sorts rows on write, pre-partitioning and sorting takes some of the load off of Kudu and helps large INSERT operations to complete without timing out. However, this default behavior may slow down the end-to-end performance of the INSERT or UPSERT operations. Starting from Impala 2.10, you can use the /* +NOCLUSTERED */ and /* +NOSHUFFLE */ hints together to disable partitioning and sorting before the rows are sent to Kudu. Additionally, since sorting may consume a large amount of memory, consider setting the MEM_LIMIT query option for those queries.

Hints for scheduling of scan ranges (HDFS data blocks or Kudu tablets):

The hints /* +SCHEDULE_CACHE_LOCAL */, /* +SCHEDULE_DISK_LOCAL */, and /* +SCHEDULE_REMOTE */ have the same effect as specifying the REPLICA_PREFERENCE query option with the respective option settings of CACHE_LOCAL, DISK_LOCAL, or REMOTE.

Specifying the replica preference as a query hint always overrides the query option setting.

The hint /* +RANDOM_REPLICA */ is the same as enabling the SCHEDULE_RANDOM_REPLICA query option.

You can use these hints in combination by separating them with commas, for example, /* +SCHEDULE_CACHE_LOCAL,RANDOM_REPLICA */.

Specifying either the SCHEDULE_RANDOM_REPLICA query option or the corresponding RANDOM_REPLICA query hint enables the random tie-breaking behavior when processing data blocks during the query.

Suggestions versus directives:

In early Impala releases, hints were always obeyed and so acted more like directives. Once Impala gained join order optimizations, sometimes join queries were automatically reordered in a way that made a hint irrelevant. Therefore, the hints act more like suggestions in Impala 1.2.2 and higher.

To force Impala to follow the hinted execution mechanism for a join query, include the STRAIGHT_JOIN keyword in the SELECT statement. When you use this technique, Impala does not reorder the joined tables at all, so you must be careful to arrange the join order to put the largest table (or subquery result set) first, then the smallest, second smallest, third smallest, and so on. This ordering lets Impala do the most I/O-intensive parts of the query using local reads on the DataNodes, and then reduce the size of the intermediate result set as much as possible as each subsequent table or subquery result set is joined.

Restrictions:

Queries that include subqueries in the WHERE clause can be rewritten internally as join queries. Currently, you cannot apply hints to the joins produced by these types of queries.

Because hints can prevent queries from taking advantage of new metadata or improvements in query planning, use them only when required to work around performance issues, and be prepared to remove them when they are no longer required, such as after a new Impala release or bug fix.

In particular, the /* +BROADCAST */ and /* +SHUFFLE */ hints are expected to be needed much less frequently in Impala 1.2.2 and higher, because the join order optimization feature in combination with the COMPUTE STATS statement now automatically choose join order and join mechanism without the need to rewrite the query and add hints.

Compatibility:

The hints embedded within -- comments are compatible with Hive queries. The hints embedded within /* */ comments or [ ] square brackets are not recognized by or not compatible with Hive. For example, Hive raises an error for Impala hints within /* */ comments because it does not recognize the Impala hint names.

Considerations for views:

If you use a hint in the query that defines a view, the hint is preserved when you query the view. Impala internally rewrites all hints in views to use the -- comment notation, so that Hive can query such views without errors due to unrecognized hint names.

Examples:

For example, this query joins a large customer table with a small lookup table of less than 100 rows. The right-hand table can be broadcast efficiently to all nodes involved in the join. Thus, you would use the /* +broadcast */ hint to force a broadcast join strategy:

```
select straight_join customer.address, state_lookup.state_name
  from customer join /* +broadcast */ state_lookup
  on customer.state_id = state_lookup.state_id;
```

This query joins two large tables of unpredictable size. You might benchmark the query with both kinds of hints and find that it is more efficient to transmit portions of each table to other nodes for processing. Thus, you would use the /* +shuffle */ hint to force a partitioned join strategy:

```
select straight_join weather.wind_velocity, geospatial.altitude
  from weather join /* +shuffle */ geospatial
  on weather.lat = geospatial.lat and weather.long = geospatial.long;
```

For joins involving three or more tables, the hint applies to the tables on either side of that specific JOIN keyword. The STRAIGHT_JOIN keyword ensures that joins are processed in a predictable order from left to right. For example, this query joins t1 and t2 using a partitioned join, then joins that result set to t3 using a broadcast join:

```
select straight_join t1.name, t2.id, t3.price
  from t1 join /* +shuffle */ t2 join /* +broadcast */ t3
  on t1.id = t2.id and t2.id = t3.id;
```

Table Level Hint

If you attach this table level hint convert_limit_to_sample(n) to a table either in the main query block or within a view/subquery and if the query satisfies the simple limit optimization conditions, the limit is converted to a table sample. The parameter 'n' in parenthesis specifies the sample percentage and must be an integer between 1 and 100.

```
        set optimize_simple_limit = true;


        CREATE VIEW v1 as SELECT * FROM T [convert_limit_to_sample(5)]
        WHERE [always_true] <predicate>;
          SELECT * FROM v1 LIMIT 10;
```

In this case, the limit 10 is applied on top of a 5 percent sample of T which is applied after partition pruning.

Related Information

Background information about join queries
Performance considerations for join queries
Query options

Query options

Specify query options in the SET statement to apply the settings to the subsequently issued queries.

Some query options are useful in day-to-day operations for improving usability, performance, or flexibility.

Other query options control special-purpose aspects of Impala operation and are intended primarily for advanced debugging or troubleshooting.

Options with Boolean parameters can be set to 1 or true to enable, or 0 or false to turn off.

You can set query options directly through the JDBC and ODBC interfaces by using the SET statement. Formerly, SET was only available as a command within the `impala-shell` interpreter.

You can set query options for an `impala-shell` session by specifying one or more command-line arguments of the form --query_option=*OPTION=VALUE*.

Impala supports the following query options:

- ABORT_ON_ERROR
- AGG_MEM_CORRELATION_FACTOR
- ALLOW_ERASURE_CODED_FILES
- APPX_COUNT_DISTINCT
- BATCH_SIZE
- BROADCAST_BYTES_LIMIT
- BUFFER_POOL_LIMIT
- COMPRESSION_CODEC
- COMPUTE_STATS_MIN_SAMPLE_SIZE
- CONVERT_LEGACY_HIVE_PARQUET_UTC_TIMESTAMPS
- DEBUG_ACTION
- DECIMAL_V2
- DEFAULT_FILE_FORMAT
- DEFAULT_HINTS_INSERT_STATEMENT
- DEFAULT_JOIN_DISTRIBUTION_MODE
- DEFAULT_NDV_SCALE
- DEFAULT_SPILLABLE_BUFFER_SIZE
- DEFAULT_TRANSACTIONAL_TYPE
- DELETE_STATS_IN_TRUNCATE
- DISABLE_CODEGEN
- DISABLE_CODEGEN_ROWS_THRESHOLD
- DISABLE_HBASE_NUM_ROWS_ESTIMATE
- DISABLE_ROW_RUNTIME_FILTERING
- DISABLE_STREAMING_PREAGGREGATIONS
- DISABLE_UNSAFE_SPILLS

- ENABLE_DISTINCT_SEMI_JOIN_OPTIMIZATION
- ENABLE_EXPR_REWRITES
- EXEC_SINGLE_NODE_ROWS_THRESHOLD
- EXEC_TIME_LIMIT_S
- EXPAND_COMPLEX_TYPES
- EXPLAIN_LEVEL
- FETCH_ROWS_TIMEOUT_MS
- HBASE_CACHE_BLOCKS
- HBASE_CACHING
- IDLE_SESSION_TIMEOUT
- JOIN_ROWS_PRODUCED_LIMIT
- JOIN_SELECTIVITY_CORRELATION_FACTOR
- KUDU_READ_MODE
- KUDU_REPLICA_SELECTION
- LARGE_AGG_MEM_THRESHOLD
- LIVE_PROGRESS
- LIVE_SUMMARY
- MAX_CNF_EXPRS
- MAX_ERRORS
- MAX_FRAGMENT_INSTANCES_PER_NODE
- MAX_MEM_ESTIMATE_FOR_ADMISSION
- MAX_NUM_RUNTIME_FILTERS
- MAX_RESULT_SPOOLING_MEM
- MAX_ROW_SIZE
- MAX_SCAN_RANGE_LENGTH
- MAX_SPILLED_RESULT_SPOOLING_MEM
- MEM_LIMIT
- MEM_LIMIT_COORDINATORS
- MIN_SPILLABLE_BUFFER_SIZE
- MT_DOP
- NUM_NODES
- NUM_ROWS_PRODUCED_LIMIT
- NUM_SCANNER_THREADS
- OPTIMIZE_SIMPLE_LIMIT
- OPTIMIZE_PARTITION_KEY_SCANS
- ORC_SCHEMA_RESOLUTION
- PARQUET_COMPRESSION_CODEC
- PARQUET_ANNOTATE_STRINGS_UTF8
- PARQUET_ARRAY_RESOLUTION
- PARQUET_DICTIONARY_FILTERING
- PARQUET_FALLBACK_SCHEMA_RESOLUTION
- PARQUET_FILE_SIZE
- PARQUET_OBJECT_STORE_SPLIT_SIZE
- PARQUET_PAGE_ROW_COUNT_LIMIT
- PARQUET_READ_PAGE_INDEX
- PARQUET_READ_STATISTICS
- PARQUET_WRITE_PAGE_INDEX
- PREFETCH_MODE
- QUERY_TIMEOUT_S
- REFRESH_UPDATED_HMS_PARTITIONS

- REPLICA_PREFERENCE
- REQUEST_POOL
- RESOURCE_TRACE_RATIO
- RETRY_FAILED_QUERIES
- RUNTIME_BLOOM_FILTER_SIZE
- RUNTIME_FILTER_MAX_SIZE
- RUNTIME_FILTER_MIN_SIZE
- RUNTIME_FILTER_MODE
- RUNTIME_FILTER_WAIT_TIME_MS
- S3_SKIP_INSERT_STAGING
- SCAN_BYTES_LIMIT
- SCHEDULE_RANDOM_REPLICA
- SCRATCH_LIMIT
- SHUFFLE_DISTINCT_EXPRS
- SPOOL_QUERY_RESULTS
- SPOOL_ALL_RESULTS_FOR_RETRIES
- SUPPORT_START_OVER
- SYNC_DDL
- THREAD_RESERVATION_AGGREGATE_LIMIT
- THREAD_RESERVATION_LIMIT
- TIMEZONE
- TOPN_BYTES_LIMIT
- UNSET_ALL
- USE_DOP_FOR_COSTING
- USE_LOCAL_TZ_FOR_UNIX_TIMESTAMP_CONVERSIONS

ABORT_ON_ERROR query option

When this option is enabled, Impala cancels a query immediately when any of the nodes encounters an error, rather than continuing and possibly returning incomplete results. This option is disabled by default, to help gather maximum diagnostic information when an error occurs, for example, whether the same problem occurred on all nodes or only a single node. Currently, the errors that Impala can skip over involve data corruption, such as a column that contains a string value when expected to contain an integer value.

To control how much logging Impala does for non-fatal errors when ABORT_ON_ERROR is turned off, use the MAX_ERRORS option.

Type: BOOLEAN

Default: FALSE (0)

AGG_MEM_CORRELATION_FACTOR and LARGE_AGG_MEM_THRESHOLD query options

These query options are an improvement in memory estimation for aggregation node and do not replace the existing query option PREAGG_BYTES_LIMIT.

Prior to this release, you used the query option PREAGG_BYTES_LIMIT to control memory usage of preagg node. But it has to be manually tweaked to be more conservative with memory and to directly limit the memory consumption of the preaggregations. When PREAGG_BYTES_LIMIT is not set, the memory estimation of each streaming preaggregation node in query plan can be expanded to consume nearly all of the memory reservation on a host, starving the merge aggregation or possibly other streaming aggregation. This is because the Planner often overestimates aggregation node memory estimates since it uses simple multiplication of NDVs of contributing grouping columns.

To reduce this large memory estimation issue, this patch includes two new query options AGG_MEM_CORRELATION_FACTOR and LARGE_AGG_MEM_THRESHOLD to the existing query option to control memory estimation. If the estimated memory from the default NDV multiplication method exceed LARGE_AGG_MEM_THRESHOLD, recompute the estimated memory again by comparing against the max(NDV) & AGG_MEM_CORRELATION_FACTOR method.

Default values:

LARGE_AGG_MEM_THRESHOLD = 512 MB

AGG_MEM_CORRELATION_FACTOR = 0.5

Note: You can revert to the old behavior by setting the query options to the following values LARGE_AGG_MEM_THRESHOLD=0 or -1, or AGG_MEM_CORRELATION_FACTOR=0.0

ALLOW_ERASURE_CODED_FILES query option

Use the ALLOW_ERASURE_CODED_FILES query option to enable or disable the support of erasure coded files in Impala. Until Impala is fully tested and certified with erasure coded files, this query option is set to FALSE by default.

When the ALLOW_ERASURE_CODED_FILES query option is set to FALSE, Impala returns an error when a query requires scanning an erasure coded file.

Type: BOOLEAN

Default: FALSE (0)

Added in: Impala 3.1

APPX_COUNT_DISTINCT query option

When the APPX_COUNT_DISTINCT query option is set to TRUE, Impala implicitly converts COUNT(DISTINCT) operations to the NDV() function calls. The resulting count is approximate rather than precise. Enable the query option when a tolerable amount of error is acceptable in order to obtain faster query results than with a COUNT (DISTINCT) queries.

Type: BOOLEAN

Default: FALSE (0)

BATCH_SIZE query option

Number of rows evaluated at a time by SQL operators. Unspecified or a size of 0 uses a predefined default size. Using a large number improves responsiveness, especially for scan operations, at the cost of a higher memory footprint.

This option is primarily for testing during Impala development, or for use under the direction of Cloudera support.

Type: Number

Default: 0 (meaning the predefined default of 1024)

BROADCAST_BYTES_LIMIT query option

Sets the limit for the size of the broadcast input based on estimated size. The Impala planner may in rare cases make a bad choice to broadcast a large table or intermediate result and encounter performance problems due to high memory pressure. Setting this limit will make the planner pick a partition based hash join instead of broadcast and avoid such performance problems.

Type: INT

Default: The default value is 34359738368 (32 GB). A value of 0 causes the option to be ignored.

Usage notes:

```
Set this option in bytes. For example:
-- Change the limit to 16GB.
set broadcast_bytes_limit=17179869184;

-- Disable the BROADCAST_BYTES_LIMIT.
set broadcast_bytes_limit=0;
```

BUFFER_POOL_LIMIT query option

Defines a limit on the amount of memory that a query can allocate from the internal buffer pool. The value for this limit applies to the memory on each host, not the aggregate memory across the cluster.

Typically not changed by users, except during diagnosis of out-of-memory errors during queries.

You can set it to an absolute value, e.g. 8GB, or a relative value, e.g. 80%, based on the MEM_LIMIT setting.

Type: INT

Default: The default setting for this option is the lower of 80% of the MEM_LIMIT setting, or the MEM_LIMIT setting minus 100 MB.

Usage notes:

If queries encounter out-of-memory errors, consider decreasing the BUFFER_POOL_LIMIT setting to less than 80% of the MEM_LIMIT setting.

COMPRESSION_CODEC query option

When Impala writes Parquet data files and other supported data files using the INSERT statement, the underlying compression is controlled by the COMPRESSION_CODEC query option.

For the ZSTD compression, the optional compression level can be specified as shown in the syntax below.

Note: Prior to Impala 2.0, this option was named PARQUET_COMPRESSION_CODEC. In Impala 2.0 and later, the PARQUET_COMPRESSION_CODEC name is not recognized. Use the more general name COMPRESSION_CODEC for new code.

Syntax:

```
SET COMPRESSION_CODEC=CODEC_NAME; // Supported for all codecs.
SET COMPRESSION_CODEC=CODEC_NAME:COMPRESSION_LEVEL; // Only su
pported for ZSTD.
```

The allowed values for this query option are SNAPPY (the default), GZIP, ZSTD, LZ4, and NONE.

ZSTD also supports setting a compression level. The lower the level, the faster the speed at the cost of compression ratio. Compression levels from 1 up to 22 are supported for ZSTD. The default compression level 3 is used, if one is not passed using the compression_codec query option.

Note: A Parquet file created with COMPRESSION_CODEC=NONE is still typically smaller than the original data, due to encoding schemes such as run-length encoding and dictionary encoding that are applied separately from compression.

The option value is not case-sensitive.

If the option is set to an unrecognized value, all kinds of queries will fail due to the invalid option setting, not just queries involving Parquet tables. (The value BZIP2 is also recognized, but is not compatible with Parquet tables.)

Type: STRING

Default: SNAPPY

COMPUTE_STATS_MIN_SAMPLE_SIZE query option

The COMPUTE_STATS_MIN_SAMPLE_SIZE query option specifies the minimum number of bytes that will be scanned in COMPUTE STATS TABLESAMPLE, regardless of the user-supplied sampling percent. This query option prevents sampling for very small tables where accurate stats can be obtained cheaply without sampling because the minimum sample size is required to get meaningful stats.

Type: INT

Default: 1GB

CONVERT_LEGACY_HIVE_PARQUET_UTC_TIMESTAMPS query option

When Hive writes to Parquet data files, the TIMESTAMP values are normalized to UTC from the local time zone of the host where the data was written. On the other hand, Impala does not make any time zone adjustment when it writes or reads INT96 TIMESTAMP values to Parquet files. This difference in time zone handling can cause potentially inconsistent results when Impala processes TIMESTAMP values in the Parquet files written by Hive.

To avoid incompatibility problems or having to code workarounds, you can specify this CONVERT_LEGACY_HIVE_PARQUET_UTC_TIMESTAMPS query option.

Type: BOOLEAN

Note: This query option can also be set as a flag on the Coordinator or by using the --default_query_options configuration. When set as a query option, it will take precedence over the old flag. The preferred method is to set this as a query option because it offers more flexibility.

DEBUG_ACTION query option

Introduces artificial problem conditions within queries. For internal Cloudera debugging and troubleshooting.

Type: STRING

Default: An empty string

DECIMAL_V2 query option

A query option that changes behavior related to the DECIMAL data type. Set this option to FALSE for backward compatibility to Impala 2.x.

Type: Boolean

Default: TRUE

DEFAULT_FILE_FORMAT query option

Use the DEFAULT_FILE_FORMAT query option to set the default table file format. The following values are supported:

- TEXT (0)
- RC_FILE (1)
- SEQUENCE_FILE (2)
- AVRO (3)
- PARQUET (4)
- KUDU (5)
- ORC (6)

In impala-shell or Hue, the SET DEFAULT_FILE_FORMAT statement will not return an error when the option was set to an unsupported value. Impala validates the value and returns an error when you submitted the next query.

Type: Enum as specified above

Default: TEXT (0)

Added in: Impala 3.3

DEFAULT_HINTS_INSERT_STATEMENT query option

The DEFAULT_HINTS_INSERT_STATEMENT query option sets the default hints for the INSERT statements with no optimizer hint specified.

When there are hints specified in an INSERT statement, these default hints are ignored.

You can specify multiple hints separated by ':'.

For example:

```
SET DEFAULT_HINTS_INSERT_STATEMENT=CLUSTERED;
SET DEFAULT_HINTS_INSERT_STATEMENT=SHUFFLE;
SET DEFAULT_HINTS_INSERT_STATEMENT=NOCLUSTERED:NOSHUFFLE;
```

The default hints apply to the HDFS and Kudu table formats and are ignored for the HBase table format.

Type: STRING

Default: No default hint

Added in: Impala 3.3

DEFAULT_JOIN_DISTRIBUTION_MODE query option

Impala optimizes join queries based on the presence of table statistics, which are produced by the Impala COMPUTE STATS statement. By default, when a table involved in the join query does not have statistics, Impala uses the “broadcast” technique that transmits the entire contents of the table to all executor nodes participating in the query. If one table involved in a join has statistics and the other does not, the table without statistics is broadcast. If both tables are missing statistics, the table on the right-hand side of the join is broadcast. This behavior is appropriate when the table involved is relatively small, but can lead to excessive network, memory, and CPU overhead if the table being broadcast is large.

Because Impala queries frequently involve very large tables, and suboptimal joins for such tables could result in spilling or out-of-memory errors, the setting DEFAULT_JOIN_DISTRIBUTION_MODE=SHUFFLE lets you override the default behavior. The shuffle join mechanism divides the corresponding rows of each table involved in a join query using a hashing algorithm, and transmits subsets of the rows to other nodes for processing. Typically, this kind of join is more efficient for joins between large tables of similar size.

The setting DEFAULT_JOIN_DISTRIBUTION_MODE=SHUFFLE is recommended when setting up and deploying new clusters, because it is less likely to result in serious consequences such as spilling or out-of-memory errors if the query plan is based on incomplete information. This setting is not the default, to avoid changing the performance characteristics of join queries for clusters that are already tuned for their existing workloads.

The allowed values are BROADCAST (0) or SHUFFLE (1).

Type: INT

Examples:

The following examples demonstrate appropriate scenarios for each setting of this query option.

```
-- Create a billion-row table.
create table big_table stored as parquet
  as select * from huge_table limit 1e9;

-- For a big table with no statistics, the
-- shuffle join mechanism is appropriate.
set default_join_distribution_mode=shuffle;
...join queries involving the big table...
```

```
-- Create a hundred-row table.
create table tiny_table stored as parquet
  as select * from huge_table limit 100;

-- For a tiny table with no statistics, the
-- broadcast join mechanism is appropriate.
set default_join_distribution_mode=broadcast;

...join queries involving the tiny table...
```

```
compute stats tiny_table;
compute stats big_table;
-- Once the stats are computed, the query option has
-- no effect on join queries involving these tables.
-- Impala can determine the absolute and relative sizes
-- of each side of the join query by examining the
-- row size, cardinality, and so on of each table.

...join queries involving both of these tables...
```

DEFAULT_NDV_SCALE query option

Improving the precision of NDV()

As a cluster admin, you can improve the precision of NDV() using this new query option DEFA ULT_NDV_SCALE. You may adjust the value in this query option to change the default precision setting for NDV() so that the SQL writers do not have to rewrite the SQLs to adjust NDV's precision. This new option allows the cluster admin to adjust precision for each Admission Control queue according to cluster's resource usage.

Controlling the NDV scale

Until this release, the scale used in NDV(<expr>) functions was fixed at 2. Now the scale can be provided using the newly added query option. This option will not influence the NDV scale for SQL function NDV(<expr>, <scale>) in which the NDV scale is provided by the 2nd argument <scale>

Type: Number

Allowed value: Range [1 .. 10]

Default: 2

Example:

The following is an example of a typical workload with different NDV scales and with a cardinality of 40,090,650.

```
      SET DEFAULT_NDV_SCALE = 10;
```

```
SET DEFAULT_NDV_SCALE = 5;
SET DEFAULT_NDV_SCALE = 2;
```

Metric	Count Distinct	NDV2	NDV5	NDV10
Memory(GB)	3.83	1.84	1.85	1.89
Duration(s)	182.89	30.22	29.72	29.24
ErrorRate	0%	1.8%	1.17%	0.06%

DEFAULT_SPILLABLE_BUFFER_SIZE query option

Specifies the default size for a memory buffer used when the spill-to-disk mechanism is activated, for example for queries against a large table with no statistics, or large join operations.

Type: INT

Default: 2097152 (2 MB)

Unit: A numeric argument represents a size in bytes; you can also use a suffix of m or mb for megabytes, or g or gb for gigabytes. If you specify a value with unrecognized formats, subsequent queries fail with an error.

Usage notes:

This query option sets an upper bound on the size of the internal buffer size that can be used during spill-to-disk operations. The actual size of the buffer is chosen by the query planner.

If overall query performance is limited by the time needed for spilling, consider increasing the DEFAULT_SPILLABLE_BUFFER_SIZE setting. Larger buffer sizes result in Impala issuing larger I/O requests to storage devices, which might result in higher throughput, particularly on rotational disks.

The tradeoff with a large value for this setting is increased memory usage during spill-to-disk operations. Reducing this value may reduce memory consumption.

To determine if the value for this setting is having an effect by capping the spillable buffer size, you can see the buffer size chosen by the query planner for a particular query. EXPLAIN the query while the setting EXPLAIN_LEVEL=2 is in effect.

DEFAULT_TRANSACTIONAL_TYPE query option

Use the DEFAULT_TRANSACTIONAL_TYPE query option to create insert-only transactional tables by default.

Note: The DEFAULT_TRANSACTIONAL_TYPE query option applies only when integrated with Hive 3.

The supported values are:

- NONE: The table will not be created as transactional.
- INSERT_ONLY: The table will be created as transactional.

If either table properties, transactional or transactional_properties, are set, this query option is ignored.

Note: The DEFAULT_TRANSACTIONAL_TYPE query option does not affect external, Kudu, or HBase tables as these cannot be transactional.

Type: Enum as specified above

Default: NONE

Added in: Impala 3.3

DELETE_STATS_IN_TRUNCATE query option

This query option DELETE_STATS_IN_TRUNCATE can be used to delete or retain table statistics. The default value of this option is 1 or true which means table statistics will be deleted as part of truncate operation. However the default may not be an ideal solution in case of non-transactional tables when the table and column statistics for the table are also deleted by default. This can be an expensive operation especially when many truncate table commands are running concurrently. In cases where truncate operation is used to remove the existing data and then reload new data, it is likely that you will compute stats again as soon as the new data is reloaded. This would overwrite the existing statistics and hence the additional time spent by the truncate operation to delete column and table statistics becomes obsolete. To avoid this expensive operation and to improve performance, you can set this query option to false or 0. When set to false, a truncate operation will not delete the table and column statistics for the table.

Type:BOOLEAN; recognized values are 1 and 0, or true and false;

Default:TRUE (shown as 1 in output of SET statement).

DISABLE_CODEGEN query option

The DISABLE_CODEGEN is a debug option, and it's used to work around any issues with Impala's runtime code generation. If a query fails with an “illegal instruction” or other hardware-specific message, try setting DISABLE_CODEGEN=true and running the query again. If the query succeeds only when the DISABLE_CODEGEN option is turned on, submit the problem to Cloudera support and include that detail in the problem report.

Most queries will run significantly slower with DISABLE_CODEGEN=true.

In Impala 2.10 and higher, the DISABLE_CODEGEN_ROWS_THRESHOLD optimisation automatically disables codegen for small queries because short-running queries may run faster without the overhead of codegen.

The following values are supported:

- TRUE or 1: Disables codegen.
- FALSE or 0: Enables codegen.

Type: BOOLEAN

Default: FALSE (0)

DISABLE_CODEGEN_ROWS_THRESHOLD query option

Specify the number of rows processed per Impala daemon which is the cutoff point below which Impala disables native code generation for the whole query. Native code generation is beneficial for queries that process many rows because it reduces the time taken to process of each row. However, generating the native code adds latency to query startup. Therefore, automatically disabling codegen for queries that process relatively small amounts of data can improve query response time.

This setting applies to queries where the number of rows processed can be accurately determined, either through table and column statistics, or by the presence of a LIMIT clause. If Impala cannot accurately estimate the number of rows, then this setting does not apply.

If a query uses the complex data types STRUCT, ARRAY, or MAP, then codegen is never automatically disabled regardless of the DISABLE_CODEGEN_ROWS_THRESHOLD setting.

Type: Number

Default: 50000

Usage notes: Typically, you increase the default value to make this optimization apply to more queries. If incorrect or corrupted table and column statistics cause Impala to apply this optimization incorrectly to queries that actually involve substantial work, you might see the queries being slower as a result of codegen being disabled. In that case, recompute statistics with the COMPUTE STATS or COMPUTE INCREMENTAL STATS statement. If there is a problem collecting accurate statistics, you can turn this feature off by setting the value to 0.

DISABLE_HBASE_NUM_ROWS_ESTIMATE

Use the DISABLE_HBASE_NUM_ROWS_ESTIMATE query option to disable key sampling of HBase tables in row count and row size estimation.

While generating a plan for an HBase query, the planner samples the underlying HBase tables to estimate their row count and row size, and the sampling can negatively impact the planning time. When the HBase table stats do not change much in a short time, disable the sampling by setting the DISABLE_HBASE_NUM_ROWS_ESTIMATE query option to TRUE. And Impala planner will fall back to using Hive Metastore (HMS) table stats instead.

When DISABLE_HBASE_NUM_ROWS_ESTIMATE query option is set to TRUE, you need to update the HMS table stats by running COMPUTE STATS. Alternatively, you can manually set table statistics by running ALTER TABLE.

The following values are supported:

- TRUE or 1: Disables the normal key sampling of HBase tables and uses HMS table stats for estimation.
- FALSE or 0: Enables the normal sampling of HBase tables.

Type: BOOLEAN

Default: FALSE

DISABLE_ROW_RUNTIME_FILTERING query

The DISABLE_ROW_RUNTIME_FILTERING query option reduces the scope of the runtime filtering feature. Queries still dynamically prune partitions, but do not apply the filtering logic to individual rows within partitions.

Only applies to queries against HDFS-based tables using the Parquet file format. For other file formats, Impala only prunes at the level of partitions, not individual rows.

This query option only applies to queries against

Type: BOOLEAN

Default: FALSE (0)

Usage notes:

Impala automatically evaluates whether the per-row filters are being effective at reducing the amount of intermediate data. Therefore, this option is typically only needed for the rare case where Impala cannot accurately determine how effective the per-row filtering is for a query.

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables.

Kudu consideration:

When applied to a query involving a Kudu table, this option turns off all runtime filtering for the Kudu table.

DISABLE_STREAMING_PREAGGREGATIONS query option

Turns off the “streaming preaggregation” optimization that is available in Impala 2.5 and higher. This optimization reduces unnecessary work performed by queries that perform aggregation operations on columns with few or no duplicate values, for example DISTINCT *ID_COLUMN* or GROUP BY *UNIQUE_COLUMN*. If the optimization causes regressions in existing queries that use aggregation functions, you can turn it off as needed by setting this query option.

Type: BOOLEAN

Default: FALSE (0)

Usage notes:

Typically, queries that would require enabling this option involve very large numbers of aggregated values, such as a billion or more distinct keys being processed on each worker node.

DISABLE_UNSAFE_SPILLS query option

Enable this option if you prefer to have queries fail when they exceed the Impala memory limit, rather than write temporary data to disk.

Queries that “spill” to disk typically complete successfully, when in earlier Impala releases they would have failed. However, queries with exorbitant memory requirements due to missing statistics or inefficient join clauses could become so slow as a result that you would rather have them cancelled automatically and reduce the memory usage through standard Impala tuning techniques.

This option prevents only “unsafe” spill operations, meaning that one or more tables are missing statistics or the query does not include a hint to set the most efficient mechanism for a join or INSE RT ... SELECT into a partitioned table. These are the tables most likely to result in suboptimal execution plans that could cause unnecessary spilling. Therefore, leaving this option enabled is a good way to find tables on which to run the COMPUTE STATS statement.

Type: BOOLEAN

Default: FALSE (0)

ENABLE_ASYNC_DDL_EXECUTION query option

When an Impala client connects to an Impala server through a Network Load Balancer, it tracks the state of that connection. The connection gets closed if the time taken in a blocking call to the server is longer than the idle timeout. If an impala client sends a TCP request after the idle timeout period elapses, it receives a TCP RST packet to indicate that the connection is no longer valid. However, if the Impala client is waiting for a response from the server, it will get stuck because it never notices the closed connection.

This limitation applies to the following clients:

- impala-shell
- Beeswax
- Impyla
- HUE

An asynchronous model has been applied to the execution of certain DDL statements that require a long execution time. So any request from an impala client to the Impala server will execute asynchronously in different threads without blocking the RPC so clients can get a query handle and poll for it for state and results to avoid impala clients hanging indefinitely.

A new query option 'enable_async_ddl_execution' has been added for this asynchronous mode and is set to true by default. You can set it to false to turn off this capability.

ENABLE_DISTINCT_SEMI_JOIN_OPTIMIZATION query option

When this query option is enabled the query planner considers inserting a distinct aggregation on the inner side of the join based on whether that aggregation would reduce the number of rows by more than 75%. This optimization reduces the number of potential plan regressions.

ENABLE_EXPR_REWRITES query option

The ENABLE_EXPR_REWRITES query option controls whether to enable or disable the query compile time optimizations. These optimizations rewrite the expression trees to a more compact and optimized form that helps avoid redundant expression evaluation at run time. Performance optimizations controlled by this query option include:

- Constant folding
- Extracting common conjuncts from disjunctions

- Simplify conditionals with constant conditions

Set the option to FALSE or 0 to disable the performance optimizations.

Type: BOOLEAN

Default: TRUE (1)

EXEC_SINGLE_NODE_ROWS_THRESHOLD query option

Specify the number of rows scanned as the cutoff point below which Impala treats a query as a "small" query, turning off optimizations such as parallel execution and native code generation. The overhead for these optimizations is applicable for queries involving substantial amounts of data, but it makes sense to skip them for queries involving tiny amounts of data. Reducing the overhead for small queries allows Impala to complete them more quickly, keeping admission control slots, CPU, memory, and so on available for resource-intensive queries.

This setting applies to queries where the number of rows processed can be accurately determined, either through table and column statistics, or by the presence of a LIMIT clause. If Impala cannot accurately estimate the number of rows, then this setting does not apply.

In Impala 2.3 and higher, where Impala supports the complex data types STRUCT, ARRAY, and MAP, if a query refers to any column of those types, the small-query optimization is turned off for that query regardless of the EXEC_SINGLE_NODE_ROWS_THRESHOLD setting.

For a query that is determined to be "small", all work is performed on the coordinator node. This might result in some I/O being performed by remote reads. The savings from not distributing the query work and not generating native code are expected to outweigh any overhead from the remote reads.

Type: Number

Default: 100

Usage notes: Typically, you increase the default value to make this optimization apply to more queries. If incorrect or corrupted table and column statistics cause Impala to apply this optimization incorrectly to queries that actually involve substantial work, you might see the queries being slower as a result of remote reads. In that case, recompute statistics with the COMPUTE STATS or COMPUTE INCREMENTAL STATS statement. If there is a problem collecting accurate statistics, you can turn this feature off by setting the value to -1.

EXEC_TIME_LIMIT_S query option

The EXEC_TIME_LIMIT_S query option sets a time limit on query execution in seconds. If a query is still executing when time limit expires, it is automatically canceled. The option is intended to prevent runaway queries that run for much longer than intended.

For example, an Impala administrator could set a default value of EXEC_TIME_LIMIT_S=3600 for a resource pool to automatically stop queries that run for longer than one hour. Then, if a user accidentally runs a large query that executes for more than one hour, it will be automatically killed after the time limit expires to free up resources. Users can override the default value per query or per session if they do not want the default EXEC_TIME_LIMIT_S value to apply to a specific query or a session.

Note:

The time limit only starts once the query is executing. Time spent planning the query, scheduling the query, or in admission control is not counted towards the execution time limit. SELECT statements are eligible for automatic cancellation until the client has fetched all result rows. DML queries are eligible for automatic cancellation until the DML statement has finished.

Type: Number

Default: 0 (no time limit)

EXPAND_COMPLEX_TYPES Query Option

Older versions of Impala did not support complex types in the select list, and therefore a SELECT * statement only expanded to primitive types even when the table contained complex-typed columns. Support for complex types in the select list has since been added, but for backward compatibility the behaviour that "*" expressions skip complex types has been kept as the default. If the EXPAND_COMPLEX_TYPES query option is set to true, "*" expressions will include complex types as well.

Note: Even if EXPAND_COMPLEX_TYPES is false, complex-typed columns that are listed explicitly in the select list (i.e. not as a "*" expression) will be included in the result.

Type:BOOLEAN

Default:FALSE

Added in:Impala 4.2

EXPLAIN_LEVEL query option

Controls the amount of detail provided in the output of the EXPLAIN statement. The basic output can help you identify high-level performance issues such as scanning a higher volume of data or more partitions than you expect. The higher levels of detail show how intermediate results flow between nodes and how different SQL operations such as ORDER BY, GROUP BY, joins, and WHERE clauses are implemented within a distributed query.

Type: STRING or INT

Default: 1

Arguments:

The allowed range of numeric values for this option is 0 to 3:

- 0 or MINIMAL: A barebones list, one line per operation. Primarily useful for checking the join order in very long queries where the regular EXPLAIN output is too long to read easily.
- 1 or STANDARD: The default level of detail, showing the logical way that work is split up for the distributed query.
- 2 or EXTENDED: Includes additional detail about how the query planner uses statistics in its decision-making process, to understand how a query could be tuned by gathering statistics, using query hints, adding or removing predicates, and so on. In Impala 3.2 and higher, the output also includes the analyzed query with the cast information in the output header, and the implicit cast info in the Predicate section.
- 3 or VERBOSE: The maximum level of detail, showing how work is split up within each node into "query fragments" that are connected in a pipeline. This extra detail is primarily useful for low-level performance testing and tuning within Impala itself, rather than for rewriting the SQL code at the user level.

Changing the value of this option controls the amount of detail in the output of the EXPLAIN statement. The extended information from level 2 or 3 is especially useful during performance tuning, when you need to confirm whether the work for the query is distributed the way you expect, particularly for the most resource-intensive operations such as join queries against large tables, queries against tables with large numbers of partitions, and insert operations for Parquet tables. The extended information also helps to check estimated resource usage when you use the admission control or resource management features.

Usage notes:

Read the EXPLAIN output from bottom to top. The lowest lines represent the initial work of the query (scanning data files), the lines in the middle represent calculations done on each node and how intermediate results are transmitted from one node to another, and the topmost lines represent the final results being sent back to the coordinator node.

The numbers in the left column are generated internally during the initial planning phase and do not represent the actual order of operations, so it is not significant if they appear out of order in the EXPLAIN output.

At all EXPLAIN levels, the plan contains a warning if any tables in the query are missing statistics. Use the COMPUTE STATS statement to gather statistics for each table and suppress this warning. See *Tables and column statistics* for details about how the statistics help query performance.

The PROFILE command in `impala-shell` always starts with an explain plan showing full detail, the same as with EXPLAIN_LEVEL=3. After the explain plan comes the executive summary, the same output as produced by the SUMMARY command in `impala-shell`.

FETCH_ROWS_TIMEOUT_MS query option

Use the FETCH_ROWS_TIMEOUT_MS query option to control how long Impala waits for query results when clients fetch rows.

When this query option is set to 0, fetch requests wait indefinitely.

The timeout applies both when query result spooling is enabled and disabled:

- When result spooling is disabled (SPOOL_QUERY_RESULTS = FALSE), the timeout controls how long a client waits for a single row batch to be produced by the coordinator.
- When result spooling is enabled ((SPOOL_QUERY_RESULTS = TRUE), a client can fetch multiple row batches at a time, so this timeout controls the total time a client waits for row batches to be produced.

The timeout also applies to fetch requests issued against queries in the 'RUNNING' state. A 'RUNNING' query has no rows available, so any fetch request will wait until the query transitions to the 'FINISHED' state and for it to fetch all requested rows. A query in the 'FINISHED' state means that the rows are available to be fetched.

Type: INT

Default: 10000 (10 seconds)

Added in: Impala 3.4

HBASE_CACHE_BLOCKS query option

Setting this option is equivalent to calling the setCacheBlocks method of the class org.apache.hadoop.hbase.client.Scan in an HBase Java application. Helps to control the memory pressure on the HBase RegionServer, in conjunction with the HBASE_CACHING query option.

Type: BOOLEAN

Default: FALSE (0)

HBASE_CACHING query option

Setting this option is equivalent to calling the setCaching method of the class org.apache.hadoop.hbase.client.Scan in an HBase Java application. Helps to control the memory pressure on the HBase RegionServer, in conjunction with the HBASE_CACHE_BLOCKS query option.

Type: BOOLEAN

Default: FALSE (0)

IDLE_SESSION_TIMEOUT query option

Specify the time in seconds after which an idle session is cancelled. A session is idle when no activity is occurring for any of the queries in that session, and the session has not started any new queries. Once a session is expired, you cannot issue any new query requests to it. The session remains open, but the only operation you can perform is to close it.

The IDLE_SESSION_TIMEOUT query option overrides the ##idle_session_timeout startup option. See *Setting timeout periods for daemons, queries, and sessions* for the ##idle_session_timeout startup option.

The IDLE_SESSION_TIMEOUT query option allows JDBC/ODBC connections to set the session timeout as a query option with the SET statement.

Type: Number

Default: 0

When this option is set to default:

- If ##idle_session_timeout is not set, the session never expires.
- If ##idle_session_timeout is set, use that timeout value.

JOIN_ROWS_PRODUCED_LIMIT query option

The JOIN_ROWS_PRODUCED_LIMIT is an advanced query option that limits the number of join rows produced by a join node and is used to prevent runaway join queries. A query is canceled when any one of the joins in the query produces more rows (totaled across all instances of that join) than the specified limit set by the JOIN_ROWS_PRODUCED_LIMIT option.

This limit is checked periodically, so it is possible that the actual number of rows produced may be slightly higher than the limit.

The default value of 0 specifies that there is no limit on the number of join rows produced.

Type: BIGINT

Allowed values: 0 or positive numbers

Default: 0

JOIN_SELECTIVITY_CORRELATION_FACTOR

Any conjuncts in an inner or outer join, similar to the following example, will be used only when no other equijoin conjunct is present.

```
SELECT * FROM t1 inner join
  (SELECT a2, MAX(b2) as max_b2 FROM t2 GROUP BY a2) s2
  ON t1.a1 = s2.a2 AND t1.b1 = s2.max_b2
```

In this example, the stats for the second conjunct involving the MAX expression gets added to the 'otherEqJoinStats' list. These stats were being used only when no other equijoin conjunct (involving base columns) was present leading to over-estimation.

To mitigate this issue, an improvement is made to the cardinality estimate for such cases by considering the conjuncts in the 'otherEqJoinStats list' in combination with the equijoin conjuncts.

A new query option join_selectivity_correlation_factor which is a floating point number between 0 and 1 inclusive is introduced in this release. It defaults to 0.0 which preserves the existing behavior of using the Minimum selectivity of the conjuncts. Given multiple join conjuncts C1, C2 ....Cn having comparable selectivity, you may notice over-estimation by several orders of magnitude.

Setting the above query option to a value higher than 0 first computes the product of the selectivities: sel(C1) * sel(C2) .. sel(Cn) and then scales it by dividing by the join_selectivity_correlation_factor.

Note that this setting applies to all the joins in the query. Another approach is to use per-join hints, but there are limitations in the hints approach too.

Default: "0.0"

KUDU_READ_MODE query option

The KUDU_READ_MODE query option allows you to set a desired consistency level for scans of Kudu tables.

The following values are supported for the query option:

- "DEFAULT": The value of the startup flag, --kudu_read_mode, is used.
- "READ_LATEST": Commonly known as the Read Committed isolation mode, in this mode, Kudu provides no consistency guarantees for this mode, except that all returned rows were committed at some point.
- "READ_AT_SNAPSHOT": Kudu will take a snapshot of the current state of the data and perform the scan over the snapshot, possibly after briefly waiting for ongoing writes to complete. This provides "Read Your Writes" consistency within a single Impala session, except in the case of a Kudu leader change. See the Kudu documentation for more details.

Type: String

Default: "DEFAULT"

Added in: Impala 3.1

KUDU_REPLICA_SELECTION query option

With the location-aware placement policy for tablet replicas in Kudu, the replicas are spread across locations so that the failure of tablet servers in one location does not make tablets unavailable. With this advanced placement policy the replicas of a tablet on Kudu tablet servers are assigned with the role as a leader or a follower. By default when Impala attempts to read from the tablet server, it is targeted to the closest tablet server that is to choose "nearby" replicas on the tablet server for scanning. However using this new query option KUDU_REPLICA_SELECTION, the queries can be targeted to the leader-only replica and skip non-leader replicas while scanning. When the new query option KUDU_REPLICA_SELECTION is set as LEADER_ONLY, Impala planner will generate a query plan that enables Impala to scan Kudu tables at the leader-only replicas.

LIVE_PROGRESS query option

When the LIVE_PROGRESS query option is set to TRUE, Impala displays an interactive progress bar showing roughly what percentage of processing has been completed for queries submitted through the `impala-shell` command. When the query finishes, the progress bar is erased from the `impala-shell` console output.

You can enable this query option within `impala-shell` by starting the shell with the --live_progress command-line option. You can still turn this setting off and on again within the shell through the SET command.

Starting in Impala 3.1, the summary output also includes the queuing status consisting of whether the query was queued and what was the latest queuing reason.

Type: BOOLEAN

Default: FALSE (0)

Usage notes:

The output from this query option is printed to standard error. The output is only displayed in interactive mode, that is, not when the -q or -f options are used.

Restrictions:

Because the percentage complete figure is calculated using the number of issued and completed “scan ranges”, which occur while reading the table data, the progress bar might reach 100% before the query is entirely finished. For example, the query might do work to perform aggregations after all the table data has been read. If many of your queries fall into this category, consider using the LIVE_SUMMARY option instead for more granular progress reporting.

The LIVE_PROGRESS and LIVE_SUMMARY query options currently do not produce any output during COMPUTE STATS operations.

Because the LIVE_PROGRESS query option is available only within the `impala-shell` interpreter, you cannot change the query option through the SQL SET statement using the JDBC or ODBC interfaces. The SET command in `impala-shell` recognizes the name as a shell-only option.

LIVE_SUMMARY query option

When the LIVE_SUMMARY query option is set to TRUE, Impala displays the same output as the SUMMARY command for queries submitted through the `impala-shell` command, with the measurements updated in real time as the query progresses. When the query finishes, the final SUMMARY output remains visible in the `impala-shell` console output.

You can enable this query option within `impala-shell` by starting the shell with the --live_summary command-line option. You can still turn this setting off and on again within the shell through the SET command.

Starting in Impala 3.1, the summary output also includes the queuing status consisting of whether the query was queued and what was the latest queuing reason.

Type: Boolean

Default: FALSE (0)

Usage notes:

The output from this query option is printed to standard error. The output is only displayed in interactive mode, that is, not when the -q or -f options are used.

Only a single report is displayed at any one time, with each update overwriting the previous numbers.

The live summary output can be useful for evaluating long-running queries, to evaluate which phase of execution takes up the most time, or if some hosts take much longer than others for certain operations, dragging overall performance down. By making the information available in real time, this feature lets you decide what action to take even before you cancel a query that is taking much longer than normal.

For example, you might see the HDFS scan phase taking a long time, and therefore revisit performance-related aspects of your schema design such as constructing a partitioned table, switching to the Parquet file format, running the COMPUTE STATS statement for the table, and so on. Or you might see a wide variation between the average and maximum times for all hosts to perform some phase of the query, and therefore investigate if one particular host needed more memory or was experiencing a network problem.

Restrictions:

The LIVE_PROGRESS and LIVE_SUMMARY query options currently do not produce any output during COMPUTE STATS operations.

Because the LIVE_SUMMARY query option is available only within the `impala-shell` interpreter, you cannot change the query option through the SQL SET statement using the JDBC or ODBC interfaces. The SET command in `impala-shell` recognizes the name as a shell-only option.

MAX_CNF_EXPRS query option

To optimize the performance of queries, the predicates are reorganized so that they can be applied earlier in the data processing. For example, some predicates can be pushed down to the scan node. This helps improve performance for some complex queries. However the CNF rewrite can lead to significant frontend memory usage and eventually OutOfMemory for a complex query that contains many predicates. To avoid this memory problem while maintaining performance improvement in basic circumstances and avoiding performance regressions on complex queries, this release lowered the default value of MAX_CNF_EXPRS from unlimited to 200.

MAX_ERRORS query option

Maximum number of non-fatal errors for any particular query that are recorded in the Impala log file. For example, if a billion-row table had a non-fatal data error in every row, you could diagnose the problem without all billion errors being logged. Unspecified or 0 indicates the built-in default value of 100.

This option only controls how many errors are reported. To specify whether Impala continues or halts when it encounters such errors, use the ABORT_ON_ERROR option.

Type: Number

Default: 0 (meaning 100 errors)

MAX_FRAGMENT_INSTANCES_PER_NODE query option

When you executed a query with COMPUTE_PROCESSING_COST=1, Impala relied on the MT_DOP option to decide the degree of parallelism of the scan fragment. Now scan node's processing cost is used as another factor to consider raising scan parallelism beyond MT_DOP.

Scan node cost now includes the number of effective scan ranges. Each scan range is given a weight of (0.5% * min_processing_per_thread), which roughly means that one scan node instance can handle at most 200 scan ranges. The new query option MAX_FRAGMENT_INSTANCES_PER_NODE is introduced to cap the maximum number of fragment instances per node. This newly introduced query option works in conjunction with PROCESSING_COST_MIN_THREADS.

Query option MAX_FRAGMENT_INSTANCES_PER_NODE is added as an upper bound on scan parallelism if COMPUTE_PROCESSING_COST=true. If the number of scan ranges is fewer than the maximum parallelism allowed by the scan node's processing cost, then the processing cost will be clamped down to (min_processing_per_thread / number of scan ranges). Lowering MAX_FRAGMENT_INSTANCES_PER_NODE can also clamp down the scan processing cost. For interior fragments, a combination of MAX_FRAGMENT_INSTANCES_PER_NODE, PROCESSING_COST_MIN_THREADS, and the number of available cores per node is accounted for to determine maximum fragment parallelism per node. For the scan fragments, only the first two are considered to encourage frontend to choose a larger executor group as needed.

Type: INT

Default: 128

MAX_MEM_ESTIMATE_FOR_ADMISSION query option

Use the MAX_MEM_ESTIMATE_FOR_ADMISSION query option to set an upper limit on the memory estimates of a query as a workaround for over-estimates precluding a query from being admitted.

The query option takes effect when all of the below conditions are met:

- Memory-based admission control is enabled for the pool.
- The MEM_LIMIT query option is not set at the query, session, resource pool, or global level.

When the above conditions are met, MIN(MAX_MEM_ESTIMATE_FOR_ADMISSION, mem_estimate) is used for admission control.

Setting the MEM_LIMIT query option is usually a better option. Use the MAX_MEM_ESTIMATE_FOR_ADMISSION query option when it is not feasible to set MEM_LIMIT for each individual query.

Type: INT

Added in: Impala 3.1

MAX_NUM_RUNTIME_FILTERS query option

The MAX_NUM_RUNTIME_FILTERS query option sets an upper limit on the number of runtime filters that can be produced for each query.

Type: INT

Default: 10

Usage notes:

Each runtime filter imposes some memory overhead on the query. Depending on the setting of the RUNTIME_BLOOM_FILTER_SIZE query option, each filter might consume between 1 and 16 megabytes per plan fragment. There are typically 5 or fewer filters per plan fragment.

Impala evaluates the effectiveness of each filter, and keeps the ones that eliminate the largest number of partitions or rows. Therefore, this setting can protect against potential problems due to excessive memory overhead for filter production, while still allowing a high level of optimization for suitable queries.

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables.

Kudu consideration:

This query option affects only Bloom filters, not the min/max filters that are applied to Kudu tables. Therefore, it does not affect the performance of queries against Kudu tables.

MAX_RESULT_SPOOLING_MEM query option

Use the MAX_RESULT_SPOOLING_MEM query option to set the maximum amount of memory used when spooling query results.

If the amount of memory exceeds this value when spooling query results, all memory will most likely be spilled to disk.

The MAX_RESULT_SPOOLING_MEM query option is applicable only when query result spooling is enabled with the SPOOL_QUERY_RESULTS query option set to TRUE.

Setting the option to 0 or -1 means the memory is unbounded.

You cannot set this query option to values below -1.

Type: INT

Default: 100 * 1024 * 1024 (100 MB)

Added in: Impala 3.4

MAX_ROW_SIZE query option

Ensures that Impala can process rows of at least the specified size. (Larger rows might be successfully processed, but that is not guaranteed.) Applies when constructing intermediate or final rows in the result set. This setting prevents out-of-control memory use when accessing columns containing huge strings.

A numeric argument represents a size in bytes; you can also use a suffix of m or mb for megabytes, or g or gb for gigabytes. If you specify a value with unrecognized formats, subsequent queries fail with an error.

Type: INT

Default: 524288 (512 KB)

Usage notes:

If a query fails because it involves rows with long strings and/or many columns, causing the total row size to exceed MAX_ROW_SIZE bytes, increase the MAX_ROW_SIZE setting to accommodate the total bytes stored in the largest row. Examine the error messages for any failed queries to see the size of the row that caused the problem.

Impala attempts to handle rows that exceed the MAX_ROW_SIZE value where practical, so in many cases, queries succeed despite having rows that are larger than this setting.

Specifying a value that is substantially higher than actually needed can cause Impala to reserve more memory than is necessary to run the query.

In a Hadoop cluster with highly concurrent workloads and queries that process high volumes of data, traditional SQL tuning advice about minimizing wasted memory is worth remembering. For example, if a table has STRING columns where a single value might be multiple megabytes, make sure that the SELECT lists in queries only refer to columns that are actually needed in the result set, instead of using the SELECT * shorthand.

MAX_SCAN_RANGE_LENGTH query option

Maximum length of the scan range. Interacts with the number of HDFS blocks in the table to determine how many CPU cores across the cluster are involved with the processing for a query. (Each core processes one scan range.)

Lowering the value can sometimes increase parallelism if you have unused CPU capacity, but a too-small value can limit query performance because each scan range involves extra overhead.

Only applicable to HDFS tables. Has no effect on Parquet tables. Unspecified or 0 indicates backend default, which is the same as the HDFS block size for each table.

Although the scan range can be arbitrarily long, Impala internally uses an 8 MB read buffer so that it can query tables with huge block sizes without allocating equivalent blocks of memory.

The value can include unit specifiers, such as 100m or 100mb.

Type: Number

Default: 0

MAX_SPILLED_RESULT_SPOOLING_MEM query option

Use the MAX_SPILLED_RESULT_SPOOLING_MEM query option to set the maximum amount of memory that can be spilled when spooling query results.

If the amount of memory exceeds this value when spooling query results, the coordinator fragment will block until the client has consumed enough rows to free up more memory.

The MAX_SPILLED_RESULT_SPOOLING_MEM query option is applicable only when query result spooling is enabled with the SPOOL_QUERY_RESULTS query option set to TRUE.

The value must be greater than or equal to the value of MAX_RESULT_SPOOLING_MEM.

Setting the option to 0 or -1 means the memory is unbounded.

Values below -1 are not allowed for this query option.

Type: INT

Default: 1024 * 1024 * 1024 (1 GB)

Added in: Impala 3.4

MEM_LIMIT query option

The MEM_LIMIT query option defines the maximum amount of memory a query can allocate on each node. The total memory that can be used by a query is the MEM_LIMIT times the number of nodes.

There are two levels of memory limit for Impala. The ##mem_limit startup option sets an overall limit for the `impalad` process (which handles multiple queries concurrently). That process memory limit can be expressed either as a percentage of RAM available to the process such as -mem_limit=70% or as a fixed amount of memory, such as ##mem_limit=100gb. The memory available to the process is based on the host's physical memory and memory limits from Linux Control Groups. For example, if an `impalad` process is running in a Docker container on a host with 100 GB of memory, the memory available is 100 GB or the Docker container's memory limit, whichever is less.

The MEM_LIMIT query option, which you set through `impala-shell` or the SET statement in a JDBC or ODBC application, applies to each individual query. The MEM_LIMIT query option is usually expressed as a fixed size such as 10gb, and must always be less than the `impalad` memory limit.

If query processing approaches the specified memory limit on any node, either the per-query limit or the `impalad` limit, then the SQL operations will start to reduce their memory consumption, for example by writing the temporary data to disk (known as spilling to disk). The result is a query that completes successfully, rather than failing with an out-of-memory error. The tradeoff is decreased performance due to the extra disk I/O to write the temporary data and read it back in. The slowdown could potentially be significant. Thus, while this feature improves reliability, you should optimize your queries, system parameters, and hardware configuration to make this spilling a rare occurrence.

Type: Number

Units: A numeric argument represents memory size in bytes; you can also use a suffix of m or mb for megabytes, or more commonly g or gb for gigabytes. If you specify a value with unrecognized formats, subsequent queries fail with an error.

Default: 0 (unlimited)

Usage notes:

The MEM_LIMIT setting is primarily useful for production workloads. Impala's Admission Controller can be configured to automatically assign memory limits to queries and limit memory consumption of resource pools.

Use the output of the SUMMARY command in `impala-shell` to get a report of memory used for each phase of your most heavyweight queries on each node, and then set a MEM_LIMIT somewhat higher than that. See Understanding Performance using SUMMARY Report for usage information about the SUMMARY command.

The following examples show how to set the MEM_LIMIT query option using a fixed number of bytes, or suffixes representing gigabytes or megabytes.

```
[localhost:21000] > set mem_limit=3000000000;
MEM_LIMIT set to 3000000000
[localhost:21000] > select 5;
Query: select 5
+---+
| 5 |
+---+
| 5 |
+---+

[localhost:21000] > set mem_limit=3g;
MEM_LIMIT set to 3g
[localhost:21000] > select 5;
Query: select 5
+---+
| 5 |
+---+
| 5 |
+---+

[localhost:21000] > set mem_limit=3gb;
MEM_LIMIT set to 3gb
[localhost:21000] > select 5;
+---+
| 5 |
+---+
| 5 |
+---+
```

```
[localhost:21000] > set mem_limit=3m;
MEM_LIMIT set to 3m
[localhost:21000] > select 5;
+---+
| 5 |
+---+
| 5 |
+---+
[localhost:21000] > set mem_limit=3mb;
MEM_LIMIT set to 3mb
[localhost:21000] > select 5;
+---+
| 5 |
+---+
```

The following examples show how unrecognized MEM_LIMIT values lead to errors for subsequent queries.

```
[localhost:21000] > set mem_limit=3pb;
MEM_LIMIT set to 3pb
[localhost:21000] > select 5;
ERROR: Failed to parse query memory limit from '3pb'.

[localhost:21000] > set mem_limit=xyz;
MEM_LIMIT set to xyz
localhost:21000] > select 5;
Query: select 5
ERROR: Failed to parse query memory limit from 'xyz'.
```

The following examples shows the automatic query cancellation when the MEM_LIMIT value is exceeded on any host involved in the Impala query. First it runs a successful query and checks the largest amount of memory used on any node for any stage of the query. Then it sets an artificially low MEM_LIMIT setting so that the same query cannot run.

```
[localhost:21000] > select count(*) from customer;
Query: select count(*) from customer
+----------+
| count(*) |
+----------+
| 150000   |
+----------+

[localhost:21000] > select count(distinct c_name) from customer;
Query: select count(distinct c_name) from customer
+------------------------+
| count(distinct c_name) |
+------------------------+
| 150000                 |
+------------------------+

[localhost:21000] > summary;
+--------------+--------+--------+----------+----------+---------
+------------+----------+---------------+---------------+
| Operator     | #Hosts | #Inst  | Avg Time | Max Time | #Rows
 | Est. #Rows | Peak Mem | Est. Peak Mem | Detail        |
+--------------+--------+--------+----------+----------+---------
+------------+----------+---------------+---------------+
| 06:AGGREGATE | 1      | 1      | 230.00ms | 230.00ms | 1
 | 1          | 16.00 KB | -1 B          | FINALIZE      |
| 05:EXCHANGE  | 1      | 1      | 43.44us  | 43.44us  | 1
| 1          | 0 B      | -1 B          | UNPARTITIONED |
```

```
| 02:AGGREGATE | 1      | 1      | 227.14ms | 227.14ms | 1
 | 1          | 12.00 KB | 10.00 MB      |               |
| 04:AGGREGATE | 1      | 1      | 126.27ms | 126.27ms | 150.00K
 | 150.00K    | 15.17 MB | 10.00 MB      |               |
| 03:EXCHANGE  | 1      | 1      | 44.07ms  | 44.07ms  | 150.00K
 | 150.00K    | 0 B      | 0 B           | HASH(c_name)  |
| 01:AGGREGATE | 1      | 1      | 361.94ms | 361.94ms | 150.00K
 | 150.00K    | 23.04 MB | 10.00 MB      |               |
| 00:SCAN HDFS | 1      | 1      | 43.64ms  | 43.64ms  | 150.00K
 | 150.00K    | 24.19 MB | 64.00 MB      | tpch.customer |
+--------------+--------+--------+----------+----------+---------
+------------+----------+---------------+---------------+

[localhost:21000] > set mem_limit=15mb;
MEM_LIMIT set to 15mb
[localhost:21000] > select count(distinct c_name) from customer;
Query: select count(distinct c_name) from customer
ERROR:
Rejected query from pool default-pool: minimum memory reservation
 is greater than memory available to the query
for buffer reservations. Memory reservation needed given the curr
ent plan: 38.00 MB. Adjust either the mem_limit
or the pool config (max-query-mem-limit, min-query-mem-limit) for
 the query to allow the query memory limit to be
at least 70.00 MB. Note that changing the mem_limit may also c
hange the plan. See the query profile for more
information about the per-node memory requirements.
```

MEM_LIMIT_COORDINATORS query option

The new MEM_LIMIT_COORDINATORS query option functions similarly to the MEM_LIMIT option but sets the query memory limit only on coordinators. This new option addresses the issue related to MEM_LIMIT and is recommended in scenarios where the query needs higher or lower memory on coordinators compared to the planner estimates.

Note: the MEM_LIMIT_COORDINATORS option does not work with MEM_LIMIT. If you set both, only MEM_LIMIT applies.

MIN_SPILLABLE_BUFFER_SIZE query option

Specifies the minimum size for a memory buffer used when the spill-to-disk mechanism is activated, for example for queries against a large table with no statistics, or large join operations.

Type: INT

Default: 65536 (64 KB)

Units: A numeric argument represents memory size in bytes; you can also use a suffix of m or mb for megabytes, or more commonly g or gb for gigabytes. If you specify a value with unrecognized formats, subsequent queries fail with an error.

Usage notes:

This query option sets a lower bound on the size of the internal buffer size that can be used during spill-to-disk operations. The actual size of the buffer is chosen by the query planner.

If overall query performance is limited by the time needed for spilling, consider increasing the MIN_SPILLABLE_BUFFER_SIZE setting. Larger buffer sizes result in Impala issuing larger I/O requests to storage devices, which might result in higher throughput, particularly on rotational disks.

The tradeoff with a large value for this setting is increased memory usage during spill-to-disk operations. Reducing this value may reduce memory consumption.

To determine if the value for this setting is having an effect by capping the spillable buffer size, you can see the buffer size chosen by the query planner for a particular query. EXPLAIN the query while the setting EXPLAIN_LEVEL=2 is in effect.

MT_DOP query option

Sets the degree of parallelism used for certain operations that can benefit from multithreaded execution. You can specify values higher than zero to find the ideal balance of response time, memory usage, and CPU usage during statement processing.

Note:

The Impala execution engine is being revamped incrementally to add additional parallelism within a single host for certain statements and kinds of operations. The setting MT_DOP=0 uses the “old” code path with limited intra-node parallelism.

Higher value of MT_DOP is directly proportional to resource requirements.

Currently, MT_DOP support varies by statement type:

- COMPUTE [INCREMENTAL] STATS. Impala automatically sets MT_DOP=4 for COMPUTE STATS and COMPUTE INCREMENTAL STATS statements on Parquet tables.
- SELECT statements. MT_DOP is 0 by default for SELECT statements but can be set to a value greater than 0 to control intra-node parallelism. This may be useful to tune query performance and in particular to reduce execution time of long-running, CPU-intensive queries.
- DML statements. MT_DOP values greater than zero are not currently supported for DML statements. DML statements will produce an error if MT_DOP is set to a non-zero value.
- In Impala 3.4 (Cloudera 1.2) and earlier, not all SELECT statements support setting MT_DOP. Specifically, only scan and aggregation operators, and local joins that do not need data exchanges (such as for nested types) are supported. Other SELECT statements produce an error if MT_DOP is set to a non-zero value.

Type: INT

Default: 0

Because COMPUTE STATS and COMPUTE INCREMENTAL STATS statements for Parquet tables benefit substantially from extra intra-node parallelism, Impala automatically sets MT_DOP=4 when computing stats for Parquet tables.

Range: 0 to 64

Examples:

Note:

Any timing figures in the following examples are on a small, lightly loaded development cluster. Your mileage may vary. Speedups depend on many factors, including the number of rows, columns, and partitions within each table.

The following example shows how to run a COMPUTE STATS statement against a Parquet table with or without an explicit MT_DOP setting:

```
-- Explicitly setting MT_DOP to 0 selects the old code path.
set mt_dop = 0;
MT_DOP set to 0

-- The analysis for the billion rows is distributed among hosts,
-- but uses only a single core on each host.
compute stats billion_rows_parquet;
```

```
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 2 column(s). |
+-----------------------------------------+

drop stats billion_rows_parquet;

-- Using 4 logical processors per host is faster.
set mt_dop = 4;
MT_DOP set to 4

compute stats billion_rows_parquet;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 2 column(s). |
+-----------------------------------------+

drop stats billion_rows_parquet;

-- Unsetting the option reverts back to its default.
-- Which for COMPUTE STATS and a Parquet table is 4,
-- so again it uses the fast path.
unset MT_DOP;
Unsetting option MT_DOP

compute stats billion_rows_parquet;
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 2 column(s). |
+-----------------------------------------+
```

The following example shows the effects of setting MT_DOP for a query on a Parquet table:

```
set mt_dop = 0;
MT_DOP set to 0

-- COUNT(DISTINCT) for a unique column is CPU-intensive.
select count(distinct id) from billion_rows_parquet;
+--------------------+
| count(distinct id) |
+--------------------+
| 1000000000         |
+--------------------+
Fetched 1 row(s) in 67.20s

set mt_dop = 16;
MT_DOP set to 16

-- Introducing more intra-node parallelism for the aggregation
-- speeds things up, and potentially reduces memory overhead by
-- reducing the number of scanner threads.
select count(distinct id) from billion_rows_parquet;
+--------------------+
| count(distinct id) |
+--------------------+
| 1000000000         |
+--------------------+
```

```
Fetched 1 row(s) in 17.19s
```

The following example shows how queries that are not compatible with non-zero MT_DOP settings produce an error when MT_DOP is set:

```
set mt_dop=1;
MT_DOP set to 1

insert into a1
select * from a2;
ERROR: NotImplementedException: MT_DOP not supported for DML st
atements.
```

Related Information:

Compute Stats Statement

Impala Aggregate Functions

NUM_NODES query option

Limit the number of nodes that process a query, typically during debugging.

Type: Number

Allowed values: Only accepts the values 0 (meaning all nodes) or 1 (meaning all work is done on the coordinator node).

Default: 0

Usage notes:

If you are diagnosing a problem that you suspect is due to a timing issue due to distributed query processing, you can set NUM_NODES=1 to verify if the problem still occurs when all the work is done on a single node.

You might set the NUM_NODES option to 1 briefly, during INSERT or CREATE TABLE AS SELECT statements. Normally, those statements produce one or more data files per data node. If the write operation involves small amounts of data, a Parquet table, and/or a partitioned table, the default behavior could produce many small files when intuitively you might expect only a single output file. SET NUM_NODES=1 turns off the “distributed” aspect of the write operation, making it more likely to produce only one or a few data files.

Warning:

Because this option results in increased resource utilization on a single host, it could cause problems due to contention with other Impala statements or high resource usage. Symptoms could include queries running slowly, exceeding the memory limit, or appearing to stop responding. Use it only in a single-user development/test environment; do not use it in a production environment or in a cluster with a high-concurrency or high-volume or performance-critical workload.

NUM_ROWS_PRODUCED_LIMIT query option

The NUM_ROWS_PRODUCED_LIMIT query option limits the number of rows produced by a query. A query is canceled when its execution produces more rows than the specified limit set by the NUM_ROWS_PRODUCED_LIMIT option.

This limit only applies when the results are returned to a client, for example, to a SELECT query, but not to an INSERT query.

The default value of 0 specifies that there is no limit on the number of rows produced.

Type: INT

Allowed values: 0 or positive numbers

Default: 0

NUM_SCANNER_THREADS query option

Maximum number of scanner threads (on each node) used for each query. By default, Impala uses as many cores as are available (one thread per core). You might lower this value if queries are using excessive resources on a busy cluster. Impala imposes a maximum value automatically, so a high value has no practical effect.

Type: Number

Default: 0

OPTIMIZE_PARTITION_KEY_SCANS query option

Enables a fast code path for queries that apply simple aggregate functions to partition key columns: MIN(*KEY_COLUMN*), MAX(*KEY_COLUMN*), or COUNT(DISTINCT *KEY_COLUMN*).

Type: BOOLEAN

Default: FALSE (0)

Usage notes:

This optimization speeds up common “introspection” operations over partition key columns, for example determining the distinct values of partition keys.

This optimization does not apply to SELECT statements that reference columns that are not partition keys. It also only applies when all the partition key columns in the SELECT statement are referenced in one of the following contexts:

- Within a MAX() or MAX() aggregate function or as the argument of any aggregate function with the DISTINCT keyword applied.
- Within a WHERE, GROUP BY or HAVING clause.

This optimization is enabled by a query option because it skips some consistency checks and therefore can return slightly different partition values if partitions are in the process of being added, dropped, or loaded outside of Impala. Queries might exhibit different behavior depending on the setting of this option in the following cases:

- If files are removed from a partition using HDFS or other non-Impala operations, there is a period until the next REFRESH of the table where regular queries fail at run time because they detect the missing files. With this optimization enabled, queries that evaluate only the partition key column values (not the contents of the partition itself) succeed, and treat the partition as if it still exists.
- If a partition contains any data files, but the data files do not contain any rows, a regular query considers that the partition does not exist. With this optimization enabled, the partition is treated as if it exists.

  If the partition includes no files at all, this optimization does not change the query behavior: the partition is considered to not exist whether or not this optimization is enabled.

OPTIMIZE_SIMPLE_LIMIT query option

This new planner query option OPTIMIZE_SIMPLE_LIMIT optimizes the planning time for simple limit queries by only considering a minimal set of partitions whose file descriptors add up to N (the specified limit). Each file is conservatively estimated to contain 1 row.

This reduces the number of partitions processed by the planner which, according to query profiling, has been the main contributor to the planning time especially for large numbers of partitions.

Further, within each partition, only the number of non-empty files that brings the total to N will be considered. For the optimization to be considered, this option should be always set to "true".

This is an opt-in optimization and also applies to subqueries and view.

ORC_SCHEMA_RESOLUTION query option

The ORC_SCHEMA_RESOLUTION query option allows Impala to look up columns within ORC files by column names, rather than by column indices, when necessary.

The allowed values are:

- POSITION (0)
- NAME (1)

Type: Enum of the above-allowed values

Usage notes:

By default, Impala looks up columns within an ORC file based on the order of columns in the table.

Note: Hive only writes column names to ORC files when deployed in Cloudera. So this feature is only required when deployed in Cloudera.

Impact of column renaming

Position-based resolution is resilient to column renaming however, a name-based resolution will fail when you rename a column. The following example demonstrates the impact of column renaming on position-based and name-based resolution.

1. View the details of the table test_orc

```
select * from test_orc;
+---------+------------+------------+
| int_col | string_col | double_col |
+---------+------------+------------+
| 0       | A          | 0.5        |
| 1       | B          | 1.5        |
| 2       | C          | 2.5        |
+---------+------------+------------+
```

2. Rename the column “double_col” to “new_double_col”

```
alter table test_orc change double_col new_double_col double;
```

3. Set resolving ORC file columns by Index

```
set orc_schema_resolution=POSITION;
```

4. View the table

```
select int_col, string_col, new_double_col from test_orc;
+---------+------------+----------------+
| int_col | string_col | new_double_col |
+---------+------------+----------------+
| 0       | A          | 0.5            |
| 1       | B          | 1.5            |
| 2       | C          | 2.5            |
+---------+------------+----------------+
```

The above query results show that renaming a column did not impact when you resolved the ORC file columns by POSITION. However, if you resolve ORC file columns by NAME, set orc_sche ma_resolution=NAME, viewing the table details will fail to fetch the renamed columns.

```
select int_col, string_col, new_double_col from test_orc;
```

```
+---------+------------+----------------+
| int_col | string_col | new_double_col |
+---------+------------+----------------+
| 0       | A          | NULL           |
| 1       | B          | NULL           |
| 2       | C          | NULL           |
+---------+------------+----------------+
```

PARQUET_COMPRESSION_CODEC query option

Deprecated. Use COMPRESSION_CODEC in Impala 2.0 and later.

PARQUET_ANNOTATE_STRINGS_UTF8 query option

Causes Impala INSERT and CREATE TABLE AS SELECT statements to write Parquet files that use the UTF-8 annotation for STRING columns.

Type: BOOLEAN

Default: FALSE (0)

Usage notes:

By default, Impala represents a STRING column in Parquet as an unannotated binary field.

Impala always uses the UTF-8 annotation when writing CHAR and VARCHAR columns to Parquet files. An alternative to using the query option is to cast STRING values to VARCHAR.

This option is to help make Impala-written data more interoperable with other data processing engines. Impala itself currently does not support all operations on UTF-8 data. Although data processed by Impala is typically represented in ASCII, it is valid to designate the data as UTF-8 when storing on disk, because ASCII is a subset of UTF-8.

PARQUET_ARRAY_RESOLUTION query option

The PARQUET_ARRAY_RESOLUTION query option controls the behavior of the indexed-based resolution for nested arrays in Parquet.

In Parquet, you can represent an array using a 2-level or 3-level representation. The modern, standard representation is 3-level. The legacy 2-level scheme is supported for compatibility with older Parquet files. However, there is no reliable metadata within Parquet files to indicate which encoding was used. It is even possible to have mixed encodings within the same file if there are multiple arrays. The PARQUET_ARRAY_RESOLUTION option controls the process of resolution that is to match every column/field reference from a query to a column in the Parquet file.

The supported values for the query option are:

- THREE_LEVEL: Assumes arrays are encoded with the 3-level representation, and does not attempt the 2-level resolution.
- TWO_LEVEL: Assumes arrays are encoded with the 2-level representation, and does not attempt the 3-level resolution.
- TWO_LEVEL_THEN_THREE_LEVEL: First tries to resolve assuming a 2-level representation, and if unsuccessful, tries a 3-level representation.

All of the above options resolve arrays encoded with a single level.

A failure to resolve a column/field reference in a query with a given array resolution policy does not necessarily result in a warning or error returned by the query. A mismatch might be treated like a missing column (returns NULL values), and it is not possible to reliably distinguish the 'bad resolution' and 'legitimately missing column' cases.

The name-based policy generally does not have the problem of ambiguous array representations. You specify to use the name-based policy by setting the PARQUET_FALLBACK_SCHEMA_RESOLUTION query option to NAME.

Type: Enum of TWO_LEVEL, TWO_LEVEL_THEN_THREE_LEVEL, and THREE_LEVEL

Default: THREE_LEVEL

Examples:

EXAMPLE A: The following Parquet schema of a file can be interpreted as a 2-level or 3-level:

```
ParquetSchemaExampleA {
  optional group single_element_groups (LIST) {
    repeated group single_element_group {
      required int64 count;
    }
  }
}
```

The following table schema corresponds to a 2-level interpretation:

```
CREATE TABLE t (col1 array<struct<f1: bigint>>) STORED AS PARQU
ET;
```

Successful query with a 2-level interpretation:

```
SET PARQUET_ARRAY_RESOLUTION=TWO_LEVEL;
SELECT ITEM.f1 FROM t.col1;
```

The following table schema corresponds to a 3-level interpretation:

```
CREATE TABLE t (col1 array<bigint>) STORED AS PARQUET;
```

Successful query with a 3-level interpretation:

```
SET PARQUET_ARRAY_RESOLUTION=THREE_LEVEL;
SELECT ITEM FROM t.col1
```

EXAMPLE B: The following Parquet schema of a file can be only be successfully interpreted as a 2-level:

```
ParquetSchemaExampleB {
  required group list_of_ints (LIST) {
    repeated int32 list_of_ints_tuple;
  }
}
```

The following table schema corresponds to a 2-level interpretation:

```
CREATE TABLE t (col1 array<int>) STORED AS PARQUET;
```

Successful query with a 2-level interpretation:

```
SET PARQUET_ARRAY_RESOLUTION=TWO_LEVEL;
SELECT ITEM FROM t.col1
```

Unsuccessful query with a 3-level interpretation. The query returns NULLs as if the column was missing in the file:

```
SET PARQUET_ARRAY_RESOLUTION=THREE_LEVEL;
SELECT ITEM FROM t.col1
```

PARQUET_DICTIONARY_FILTERING query option

The PARQUET_DICTIONARY_FILTERING query option controls whether Impala uses dictionary filtering for Parquet files.

To efficiently process a highly selective scan query, when this option is enabled, Impala checks the values in the Parquet dictionary page and determines if the whole row group can be thrown out.

A column chunk is purely dictionary encoded and can be used by dictionary filtering if any of the following conditions are met:

1. If the encoding_stats is in the Parquet file, dictionary filtering uses it to determine if there are only dictionary encoded pages (i.e. there are no data pages with an encoding other than PLAIN_DICTIONARY).
2. If the encoding stats are not present, dictionary filtering looks at the encodings. The column is purely dictionary encoded if both of the conditions satisfy:
   - PLAIN_DICTIONARY is present.
   - Only PLAIN_DICTIONARY, RLE, or BIT_PACKED encodings are listed.
3. Dictionary filtering works for the Parquet dictionaries with less than 40000 values if the file was written by Impala 2.9.

In the query runtime profile output for each Impalad instance, the NumDictFilteredRowGroups field in the SCAN node section shows the number of row groups that were skipped based on dictionary filtering.

Note that row groups can be filtered out by Parquet statistics, and in such cases, dictionary filtering will not be considered.

The supported values for the query option are:

- TRUE (1): Use dictionary filtering.
- FALSE (0): Do not use dictionary filtering
- Any other values are treated as FALSE.

Type: BOOLEAN

Default: TRUE (1)

PARQUET_FALLBACK_SCHEMA_RESOLUTION query option

The PARQUET_FALLBACK_SCHEMA_RESOLUTION query option allows Impala to look up columns within Parquet files by column name, rather than column order, when necessary. The allowed values are:

- POSITION (0)
- NAME (1)

Type: Enum of the above allowed values

Usage notes:

By default, Impala looks up columns within a Parquet file based on the order of columns in the table. The name setting for this option enables behavior for Impala queries similar to the Hive setting parquet.column.index access=false. It also allows Impala to query Parquet files created by Hive with the parquet.column.index.access=false setting in effect.

PARQUET_FILE_SIZE query option

Specifies the maximum size of each Parquet data file produced by Impala INSERT statements in bytes, or with a trailing m or g character to indicate megabytes or gigabytes.

With tables that are small or finely partitioned, the default Parquet block size (formerly 1 GB, now 256 MB in Impala 2.0 and later) could be much larger than needed for each data file. For INSERT operations into such tables, you can increase parallelism by specifying a smaller PARQUET_FILE _SIZE value, resulting in more HDFS blocks that can be processed by different nodes.

Type: Number, with optional unit specifier

Important:

Currently, the maximum value for this setting is 1 gigabyte (1g). Setting a value higher than 1 gigabyte could result in errors during an INSERT operation.

Default: 0 (produces files with a target size of 256 MB; files might be larger for very wide tables)

PARQUET_OBJECT_STORE_SPLIT_SIZE

Use the PARQUET_OBJECT_STORE_SPLIT_SIZE query option to control the Parquet split sizes on non-block based stores, e.g. S3, ADLS, etc.

Type: INT

Allowed values: The value must be greater than or equal to 1 MB.

Default: 256 MB

Added in: Impala 3.4 / Cloudera 1.2

PARQUET_PAGE_ROW_COUNT_LIMIT query option

Use the PARQUET_PAGE_ROW_COUNT_LIMIT query option to set the maximum number of rows that can be written on a single Parquet data page. By default there is no row count limit.

Type: INT

Allowed values: Positive integers

Added in: Impala 3.3

PARQUET_READ_PAGE_INDEX query option

Use the PARQUET_READ_PAGE_INDEX query option to disable or enable using the Parquet page index during scans. The page index contains min/max statistics at the page-level granularity. It can be used to skip pages and rows that do not match the conditions in the WHERE clause.

This option enables the same optimization as the PARQUET_READ_STATISTICS at the finer grained page level.

Impala supports filtering based on Parquet statistics:

- Of the types: BOOLEAN, INT, DECIMAL, STRING, TIMESTAMP, DATE
- For simple predicates of the forms: <slot> <op> <constant> or <constant> <op> <slot>, where <op> is LT, LE, GE, GT, and EQ

The supported values for the query option are:

- TRUE (1): Read the page-level statistics from the Parquet page index during query processing and filter out pages based on the statistics.
- FALSE (0): Do not use the Parquet page index.
- Any other values are treated as false.

Type: BOOLEAN

Default: TRUE (1)

PARQUET_READ_STATISTICS query option

The PARQUET_READ_STATISTICS query option controls whether to read statistics from Parquet files and use them during query processing.

Parquet stores min/max stats which can be used to skip reading row groups if they don't qualify a certain predicate. When this query option is set to true, Impala reads the Parquet statistics and skips reading row groups that do not match the conditions in the WHERE clause.

Impala supports filtering based on Parquet statistics:

- Of the numerical types for the old version of the statistics: BOOLEAN, INTEGER, FLOAT
- Of the types for the new version of the statistics (starting in impala 2.8): BOOLEAN, INTEGER, FLOAT, DECIMAL, STRING, TIMESTAMP, DATE
- For simple predicates of the forms: <slot> <op> <constant> or <constant> <op> <slot>, where <op> is LT, LE, GE, GT, and EQ

The PARQUET_READ_STATISTICS option provides a workaround when dealing with files that have corrupt Parquet statistics and unknown errors.

In the query runtime profile output for each Impalad instance, the NumStatsFilteredRowGroups field in the SCAN node section shows the number of row groups that were skipped based on Parquet statistics.

The supported values for the query option are:

- TRUE (1): Read statistics from Parquet files and use them in query processing.
- FALSE (0): Do not use Parquet read statistics.
- Any other values are treated as FALSE.

Type: BOOLEAN

Default: TRUE

PARQUET_WRITE_PAGE_INDEX query option

The PARQUET_WRITE_PAGE_INDEX query option disables or enables the Parquet page index writing.

Impala writes page-level statistics into the Parquet page index of the types: INT, DECIMAL, STRING, TIMESTAMP, DATE

The supported values for the query option are:

- TRUE (1): Write the Parquet page index when creating Parquet files.
- FALSE (0): Do not write the Parquet page index when creating Parquet files.
- Any other values are treated as false.

Type: BOOLEAN

Default: TRUE (1)

Added in: Impala 3.3

PREFETCH_MODE query option

Determines whether the prefetching optimization is applied during join query processing.

Allowed values are:

- NONE (0)
- HT_BUCKET (1): Hash table buckets are prefetched during join query processing.

Type: Enum of the above allowed values

Default: 1 (HT_BUCKET)

QUERY_TIMEOUT_S query option

Sets the idle query timeout value for the session, in seconds. Queries that sit idle for longer than the timeout value are automatically cancelled. If the system administrator specified the --idle_query _timeout startup option, QUERY_TIMEOUT_S must be smaller than or equal to the --idle_query _timeout value.

Note:

The timeout clock for queries and sessions only starts ticking when the query or session is idle.

For queries, this means the query has results ready but is waiting for a client to fetch the data. A query can run for an arbitrary time without triggering a timeout, because the query is computing results rather than sitting idle waiting for the results to be fetched. The timeout period is intended to prevent unclosed queries from consuming resources and taking up slots in the admission count of running queries, potentially preventing other queries from starting.

For sessions, this means that no query has been submitted for some period of time.

Type: Number

Default: 0 (no timeout if --idle_query_timeout not in effect; otherwise, use --idle_query_timeout value)

REFRESH_UPDATED_HMS_PARTITIONS query option

As the name implies the query option REFRESH_UPDATED_HMS_PARTITIONS is used to refresh any updated HMS partitions.

This option is disabled by default so that the performance is not compromised when refreshing a table. However, for certain corner case scenarios refresh table command does not detect changed partitions. In case of the default refresh, catalogd detects, adds any new partitions and removes any partitions which are not present in HMS anymore. However, it does not update any partitions that changed (eg. change of location). When this query option is enabled, the refresh table command will detect certain changes to the partitions and update them accordingly. Currently, catalogd will update the partitions if any of the following StorageDescriptor properties have been modified. 1. Partition Location 2. Partition Fileformat. 3 SerdeInfo. 4. Partition schema changes. 5. Partition bucketInfo.

Type:BOOLEAN; recognized values are 1 and 0, or true and false; any other value will be interpreted as false.

Default:FALSE (shown as 0 in output of SET statement).

REPLICA_PREFERENCE query option

The REPLICA_PREFERENCE query option lets you distribute the work more evenly if hotspots and bottlenecks persist. It causes the access cost of all replicas of a data block to be considered equal to or worse than the configured value. This allows Impala to schedule reads to suboptimal replicas (e.g. local in the presence of cached ones) in order to distribute the work across more executor nodes.

Allowed values are:

- CACHE_LOCAL (0)
- DISK_LOCAL (2)
- REMOTE (4)

Type: Enum

Default: CACHE_LOCAL (0)

Usage Notes:

By default Impala selects the best replica it can find in terms of access cost. The preferred order is cached, local, and remote. With REPLICA_PREFERENCE, the preference of all replicas are

capped at the selected value. For example, when REPLICA_PREFERENCE is set to DISK_LOCAL, cached and local replicas are treated with the equal preference. When set to REMOTE, all three types of replicas, cached, local, remote, are treated with equal preference.

REQUEST_POOL query option

The pool or queue name that queries should be submitted to. Only applies when you enable the Impala admission control feature. Specifies the name of the pool used by requests from Impala to the resource manager.

Type: STRING

Default: An empty string (use the user-to-pool mapping defined by an `impalad` startup option in the Impala configuration file)

RESOURCE_TRACE_RATIO query option

The RESOURCE_TRACE_RATIO query option specifies the ratio of queries where the CPU usage info will be included in the profiles. Collecting CPU usage and sending it around adds a slight overhead during query execution. This query option lets you control whether to collect additional information to diagnose the resource usage.

For example, setting RESOURCE_TRACE_RATIO=1 adds a trace of the CPU usage to the profile of each query.

Setting RESOURCE_TRACE_RATIO=0.5 means that randomly selected half of all queries will have that information collected by the coordinator and included in the profiles.

Setting RESOURCE_TRACE_RATIO=0 means that CPU usage will not be tracked and included in the profiles.

Values from 0 to 1 are allowed.

Type: Number

Default: 0

Added in: Impala 3.2

RETRY_FAILED_QUERIES query option

Use the RETRY_FAILED_QUERIES query option to control whether or not queries are transparently retried on cluster membership changes.

Cluster membership changes typically occur when an impalad crashes, or if the node is blacklisted by the Impala Coordinator. If a SELECT query fails due to a cluster membership change, the Coordinator will cancel and unregister the running query and then launch a retry of the query. For example, if one of the executor nodes fails during query execution, the query fails but is transparently re-executed, either with the executor node immediately replaced, or with a temporarily reduced number of executor nodes. This feature supports retrying the entire query and NOT the individual query fragments. INSERT and DDL queries will NOT be retried.

Note: Query retry will be skipped if the query returned any results to the client. To avoid this, enable Result Spooling and spool_all_results_for_retries.

Type: BOOLEAN

Default: TRUE

RUNTIME_BLOOM_FILTER_SIZE query option

Size (in bytes) of Bloom filter data structure used by the runtime filtering feature.

Important:

In Impala 2.6 and higher, this query option only applies as a fallback, when statistics are not available. By default, Impala estimates the optimal size of the Bloom filter structure regardless of the setting for this option. (This is a change from the original behavior in Impala 2.5.)

In Impala 2.6 and higher, when the value of this query option is used for query planning, it is constrained by the minimum and maximum sizes specified by the RUNTIME_FILTER_MIN_SIZE and RUNTIME_FILTER_MAX_SIZE query options. The filter size is adjusted upward or downward if necessary to fit within the minimum/maximum range.

Type: INTEGER

Default: 1048576 (1 MB)

Maximum: 16 MB

Usage notes:

This setting affects optimizations for large and complex queries, such as dynamic partition pruning for partitioned tables, and join optimization for queries that join large tables. Larger filters are more effective at handling higher cardinality input sets, but consume more memory per filter.

If your query filters on high-cardinality columns (for example, millions of different values) and you do not get the expected speedup from the runtime filtering mechanism, consider doing some benchmarks with a higher value for RUNTIME_BLOOM_FILTER_SIZE. The extra memory devoted to the Bloom filter data structures can help make the filtering more accurate.

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables.

Because the effectiveness of this setting depends so much on query characteristics and data distribution, you typically only use it for specific queries that need some extra tuning, and the ideal value depends on the query. Consider setting this query option immediately before the expensive query and unsetting it immediately afterward.

Kudu consideration:

This query option affects only Bloom filters, not the min/max filters that are applied to Kudu tables. Therefore, it does not affect the performance of queries against Kudu tables.

RUNTIME_FILTER_MAX_SIZE query option

The RUNTIME_FILTER_MAX_SIZE query option adjusts the settings for the runtime filtering feature. This option defines the maximum size for a filter, no matter what the estimates produced by the planner are. This value also overrides any lower number specified for the RUNTIME_BLOOM_FILTER_SIZE query option. Filter sizes are rounded up to the nearest power of two.

Type: INTEGER

Default: 1048576 (1 MB)

Usage notes:

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables.

Kudu consideration:

This query option affects only Bloom filters, not the min/max filters that are applied to Kudu tables. Therefore, it does not affect the performance of queries against Kudu tables.

RUNTIME_FILTER_MIN_SIZE query option

The RUNTIME_FILTER_MIN_SIZE query option adjusts the settings for the runtime filtering feature. This option defines the minimum size for a filter, no matter what the estimates produced by the planner are. This value also overrides any lower number specified for the RUNTIME_BLOOM_FILTER_SIZE query option. Filter sizes are rounded up to the nearest power of two.

Type: INTEGER

Default: 0 (meaning use the value from the corresponding `impalad` startup option)

Usage notes:

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables.

Kudu consideration:

This query option affects only Bloom filters, not the min/max filters that are applied to Kudu tables. Therefore, it does not affect the performance of queries against Kudu tables.

RUNTIME_FILTER_MODE query option

The RUNTIME_FILTER_MODE query option adjusts the settings for the runtime filtering feature. It turns this feature on and off, and controls how extensively the filters are transmitted between hosts.

Allowed values are:

- OFF (0)
- LOCAL (1)
- GLOBAL (2)

Type: Enum of the above allowed values

Default: GLOBAL (2)

Usage notes:

The default is GLOBAL setting is recommended for a wide variety of workloads, to provide best performance with “out of the box” settings.

The lowest setting of LOCAL does a similar level of optimization (such as partition pruning) as in earlier Impala releases. This setting was the default in Impala 2.5, to allow for a period of post-upgrade testing for existing workloads. This setting is suitable for workloads with non-performance-critical queries, or if the coordinator node is under heavy CPU or memory pressure.

You might change the setting to OFF if your workload contains many queries involving partitioned tables or joins that do not experience a performance increase from the runtime filters feature. If the overhead of producing the runtime filters outweighs the performance benefit for queries, you can turn the feature off entirely.

RUNTIME_FILTER_WAIT_TIME_MS query option

The RUNTIME_FILTER_WAIT_TIME_MS query option adjusts the settings for the runtime filtering feature. It specifies a time in milliseconds that each scan node waits for runtime filters to be produced by other plan fragments.

Type: INTEGER

Default: 0 (meaning use the value from the corresponding `impalad` startup option)

Usage notes:

Because the runtime filtering feature applies mainly to resource-intensive and long-running queries, only adjust this query option when tuning long-running queries involving some combination of large partitioned tables and joins involving large tables

S3_SKIP_INSERT_STAGING query option

Speeds up INSERT operations on tables or partitions residing on the Amazon S3 filesystem. The tradeoff is the possibility of inconsistent data left behind if an error occurs partway through the operation.

By default, Impala write operations to S3 tables and partitions involve a two-stage process. Impala writes intermediate files to S3, then (because S3 does not provide a “rename” operation) those intermediate files are copied to their final location, making the process more expensive as on a filesystem that supports renaming or moving files. This query option makes Impala skip the intermediate files, and instead write the new data directly to the final destination.

Type: BOOLEAN

Default: TRUE (1)

Usage notes:

Important:

If a host that is participating in the INSERT operation fails partway through the query, you might be left with a table or partition that contains some but not all of the expected data files. Therefore, this option is most appropriate for a development or test environment where you have the ability to reconstruct the table if a problem during INSERT leaves the data in an inconsistent state.

The timing of file deletion during an INSERT OVERWRITE operation makes it impractical to write new files to S3 and delete the old files in a single operation. Therefore, this query option only affects regular INSERT statements that add to the existing data in a table, not INSERT OVERWRITE statements. Use TRUNCATE TABLE if you need to remove all contents from an S3 table before performing a fast INSERT with this option enabled.

Performance improvements with this option enabled can be substantial. The speed increase might be more noticeable for non-partitioned tables than for partitioned tables.

SCAN_BYTES_LIMIT query option

The SCAN_BYTES_LIMIT query option sets a limit on the bytes scanned by HDFS and HBase SCAN operations. If a query is still executing when the query’s coordinator detects that it has exceeded the limit, the query is terminated with an error. The option is intended to prevent runaway queries that scan more data than is intended.

For example, an Impala administrator could set a default value of SCAN_BYTES_LIMIT=100GB for a resource pool to automatically stop queries that scan more than 100 GB of data (see Impala Admission Control and Query Queuing for information about default query options). If a user accidentally omits a partition filter in a WHERE clause and runs a large query that scans a lot of data, the query will be automatically terminated after it scans more data than the SCAN_BYTES_LIMIT.

You can override the default value per-query or per-session, in the same way as other query options, if you do not want the default SCAN_BYTES_LIMIT value to apply to a specific query or session.

Note:

- Only data actually read from the underlying storage layer is counted towards the limit. E.g. Impala’s Parquet scanner employs several techniques to skip over data in a file that is not relevant to a specific query, so often only a fraction of the file size is counted towards SCAN_BYTES_LIMIT.
- As of Impala 3.1, bytes scanned by Kudu tablet servers are not counted towards the limit.

Because the checks are done periodically, the query may scan over the limit at times.

Type: Number

Units:

- A numeric argument represents memory size in bytes.
- Specify a suffix of m or mb for megabytes.
- Specify a suffix of g or gb for gigabytes.
- If you specify a suffix with unrecognized formats, subsequent queries fail with an error.

Default: 0 (no limit)

Added in: Impala 3.1

SCHEDULE_RANDOM_REPLICA query option

The SCHEDULE_RANDOM_REPLICA query option fine-tunes the scheduling algorithm for deciding which host processes each HDFS data block or Kudu tablet to reduce the chance of CPU hotspots.

By default, Impala estimates how much work each host has done for the query, and selects the host that has the lowest workload. This algorithm is intended to reduce CPU hotspots arising when the same host is selected to process multiple data blocks / tablets. Use the SCHEDULE_RANDOM_REPLICA query option if hotspots still arise for some combinations of queries and data layout.

The SCHEDULE_RANDOM_REPLICA query option only applies to tables and partitions that are not enabled for the HDFS caching.

Type: BOOLEAN

Default: FALSE

SCRATCH_LIMIT query option

Specifies the maximum amount of disk storage, in bytes, that any Impala query can consume on any host using the “spill to disk” mechanism that handles queries that exceed the memory limit.

Specify the size in bytes, or with a trailing m or g character to indicate megabytes or gigabytes.

A value of zero turns off the spill to disk feature for queries in the current session, causing them to fail immediately if they exceed the memory limit.

The amount of memory used per host for a query is limited by the MEM_LIMIT query option.

The more DataNodes in the cluster, the less memory is used on each host, and therefore also less scratch space is required for queries that exceed the memory limit.

Type: Number, with optional unit specifier

Default: -1 (amount of spill space is unlimited)

SHUFFLE_DISTINCT_EXPRS query option

The SHUFFLE_DISTINCT_EXPRS query option controls the shuffling behavior when a query has both grouping and distinct expressions. Impala can optionally include the distinct expressions in the hash exchange to spread the data among more nodes. However, this plan requires one more hash exchange phase.

It is recommended that you turn off this option if the NDVs of the grouping expressions are high.

Type: BOOLEAN

Default: FALSE

SPOOL_QUERY_RESULTS query Option

Use the SPOOL_QUERY_RESULTS query option to enable query result spooling, which is disabled by default.

Query result spooling controls how rows are returned to the client.

- When query result spooling is disabled (SPOOL_QUERY_RESULTS = FALSE), Impala relies on clients to fetch results to trigger the generation of more result row batches until all the result rows have been produced. If a client issues a query without fetching all the results, the query fragments will continue to consume the resources until the query is cancelled and unregistered, potentially tying up resources and cause other queries to wait for extended period of time in admission control.
- When query result spooling is enabled (SPOOL_QUERY_RESULTS = TRUE), the result sets of queries are eagerly fetched and spooled, either in memory or on disk.

  Once all result rows have been fetched and stored in the spooling location, the resources are freed up. Incoming client fetches can get the data from the spooled results.

Type: INT

Default: FALSE

Added in: Impala 3.4

SPOOL_ALL_RESULTS_FOR _RETRIES

Leave this query option SPOOL_ALL_RESULTS_FOR _RETRIES at its default value “true” for queries that can be retried. This new query option only applies if the query options RETRY_FA ILED_QUERIES and SPOOL_QUERY_RESULTS are enabled.

Currently, to avoid incorrect results, query retry will be skipped if any results are returned to the client in the original query. This new query option is introduced, for queries that can be retried, to control spooling all query results before returning any to the client. If the query results cannot be contained in the allocated result-spooling space, results will be returned and query retry will be disabled on the query.

If you set SPOOL_ALL_RESULTS_FOR _RETRIES to false, it will fall back to the original behavior - the client can fetch results when they are ready. For queries that cannot be retried, this option does not impact.

Type: BOOLEAN

Default: TRUE

SUPPORT_START_OVER query option

Leave this setting at its default value. It is a read-only setting, tested by some client applications such as Hue.

If you accidentally change it through `impala-shell`, subsequent queries encounter errors until you undo the change by issuing UNSET support_start_over.

Type: BOOLEAN

Default: FALSE

SYNC_DDL query option

When enabled, causes any DDL operation such as CREATE TABLE or ALTER TABLE to return only when the changes have been propagated to all other Impala nodes in the cluster by the Impala catalog service. That way, if you issue a subsequent CONNECT statement in `impala-shell` to connect to a different node in the cluster, you can be sure that other node will already recognize any added or changed tables. (The catalog service automatically broadcasts the DDL changes to all nodes automatically, but without this option there could be a period of inconsistency if you quickly switched to another node, such as by issuing a subsequent query through a load-balancing proxy.)

Although INSERT is classified as a DML statement, when the SYNC_DDL option is enabled, INSE RT statements also delay their completion until all the underlying data and metadata changes are propagated to all Impala nodes. Internally, Impala inserts have similarities with DDL statements in traditional database systems, because they create metadata needed to track HDFS block locations for new files and they potentially add new partitions to partitioned tables.

Note: Because this option can introduce a delay after each write operation, if you are running a sequence of CREATE DATABASE, CREATE TABLE, ALTER TABLE, INSERT, and similar statements within a setup script, to minimize the overall delay you can enable the SYNC_DDL query option only near the end, before the final DDL statement.

Type: BOOLEAN

Default: FALSE

THREAD_RESERVATION_AGGREGATE_LIMIT query option

The THREAD_RESERVATION_AGGREGATE_LIMIT query option limits the number of reserved threads for a query across all nodes on which it is executing. The option is intended to prevent execution of complex queries that can consume excessive CPU or operating system resources on a cluster. Queries that have more threads than this threshold are rejected by Impala's admission controller before they start executing.

For example, an Impala administrator could set a default value of THREAD_RESERVATION_AGGREGATE_LIMIT=2000 for a resource pool on a 100 node where they expect only relatively simple queries with less than 20 threads per node to run. This will reject queries that require more than 2000 reserved threads across all nodes, for example a query with 21 fragments running on all 100 nodes of the cluster.

You can override the default value per-query or per-session, in the same way as other query options, if you do not want the default THREAD_RESERVATION_AGGREGATE_LIMIT value to apply to a specific query or session.

Type: Number

Default: 0 (no limit)

Added in: Impala 3.1

THREAD_RESERVATION_LIMIT query option

The THREAD_RESERVATION_LIMIT query option limits the number of reserved threads for a query on each node. The option is intended to prevent execution of complex queries that can consume excessive CPU or operating system resources on a single node. Queries that have more threads per node than this threshold are rejected by Impala's admission controller before they start executing. You can see the number of reserved threads for a query in its explain plan in the "Per-Host Resource Reservation" line.

For example, an Impala administrator could set a default value of THREAD_RESERVATION_LIMIT=100 for a resource pool where they expect only relatively simple queries to run. This will reject queries that require more than 100 reserved threads on a node, for example, queries with more than 100 fragments.

You can override the default value per-query or per-session, in the same way as other query options, if you do not want the default THREAD_RESERVATION_LIMIT value to apply to a specific query or session.

Note: The number of reserved threads on a node may be lower than the maximum value in the explain plan if not all fragments of that query are scheduled on every node.

Type: Number

Default: 3000

Added in: Impala 3.1

TIMEZONE query option

The TIMEZONE query option defines the timezone used for conversions between UTC and the local time. If not set, Impala uses the system time zone where the Coordinator Impalad runs. As

query options are not sent to the Coordinator immediately, the timezones are validated only when the query runs.

Impala takes the timezone into a consideration in the following cases:

- When calling the NOW() function
- When converting between Unix time and timestamp if the use_local_tz_for_unix_timestamp_conversions flag is TRUE
- When reading Parquet timestamps written by Hive if the convert_legacy_hive_parquet_utc_timestamps flag is TRUE

Syntax:

```
SET TIMEZONE=TIME ZONE
```

TIME ZONE can be a canonical code or a time zone name defined in IANA Time Zone Database. The value is case-sensitive.

Leading/trailing quotes (') and double quotes (") are stripped.

If *TIME ZONE* is an empty string, the time zone for the query is set to the default time zone of the Impalad Coordinator.

If *TIME ZONE* is NULL or a space character, Impala returns an error when the query is executed.

Type: STRING

Default: The system time zone where the Coordinator impalad runs

Examples:

```
SET TIMEZONE=UTC;
SET TIMEZONE="Europe/Budapest";
```

Added in: Impala 3.1

TOPN_BYTES_LIMIT query option

The TOPN_BYTES_LIMIT query option places a limit on the amount of estimated memory that Impala can process for top-N queries.

Top-N queries are the queries that include both ORDER BY and LIMIT clauses. Top-N queries don't spill to disk so have to keep all rows they process in memory, and those queries can cause out-of-memory issues when running with a large limit and an offset. If the Impala planner estimates that a top-N operator will process more bytes than the TOPN_BYTES_LIMIT value, it will replace the top-N operator with the sort operator. Switching to the sort operator allows Impala to spill to disk, thus requiring less memory than top-N, but potentially with performance penalties.

The option has no effect when set to 0 or -1.

Syntax:

```
SET TOPN_BYTES_LIMIT=LIMIT
```

Type: Number

Default: 536870912 (512 MB)

Added in: Impala 3.1

UNSET_ALL query option

When you use a JDBC connection pool, it sets the query options for mem limit and broadcast bytes. And if you reuse the connection pool, the query options set during the previous connection do not

reset. This release introduces a new query option UNSET ALL to reset all the query options that were set for that specific connection.

USE_DOP_FOR_COSTING query option

The Impala query planner chooses between different techniques for performing join queries, depending on the absolute and relative sizes of the tables. Broadcast distribution is the default where the right-hand table is considered to be smaller than the left-hand table, and its contents are sent to all the other nodes involved in the query. The alternative technique is known as a partitioned distribution, which is more suitable for large tables of roughly equal size. With this technique, portions of each table are sent to appropriate other nodes where those subsets of rows can be processed in parallel. In some cases the current planner may pick broadcast distribution even when the partition distribution will be more optimal.

For the planner to choose the correct distribution method for the best possible performance, 2 new query options have been added in this release.

- use_dop_for_costing (Type:BOOLEAN, Default:true)
- broadcast_to_partition_factor (Type:DOUBLE, Default:1.0)

When the query option use_dop_for_costing is enabled, the planner incorporates the join operator's degree of parallelism (dop) and broadcast-to-partition factor in the costing of the build side of a join when comparing broadcast vs partition distribution thereby increasing the cost of the broadcast join's build side. The broadcast cost is now dependent on m and C, where:

m = degree of parallelism of the join node and,

C = the broadcast_to_partition_factor

This allows the planner to consider partition distribution where appropriate.

USE_LOCAL_TZ_FOR_UNIX_TIMESTAMP_CONVERSIONS query option

By default, Impala stores and interprets TIMESTAMP values in UTC time zone when writing to data files, reading from data files, or converting to and from system time values through functions.

When you set the use_local_tz_for_unix_timestamp_conversions query option to TRUE, Impala treats the TIMESTAMP values specified in the local time zone. The local time zone is determined in the following order with the TIMESTAMP query option takes the highest precedence:

1. The TIMESTAMP query option
2. $TZ environment variable
3. System time zone where the impalad coordinator runs

The use_local_tz_for_unix_timestamp_conversions query option can be used to fix discrepancy in INTERVAL operations. For example, a TIMESTAMP + INTERVAL *N-HOURS* can be affected by Daylight Saving Time, which Impala does not consider by default as these operations are applied as if the timestamp was in UTC. You can use the use_local_tz_for_unix_timestamp_conversions option to fix the issue.

Type: BOOLEAN

Note: This query option can also be set as a flag on the Coordinator or by using the default_query_options configuration. When set as a query option, it will take precedence over the old flag. The preferred method is to set this as a query option because it offers more flexibility.

UTF8_MODE query option

UTF-8 support allows string functions to recognize the UTF-8 characters, thus processing strings in a compatible way as other engines.

You can use the new query option, UTF8_MODE, to turn on/off the UTF-8 aware behavior. The query option can be set globally, or at per session level. Only queries with UTF8_MODE=true will

have UTF-8 aware behaviors. If the query option UTF8_MODE is turned on globally, existing queries that depend on the original binary behavior need to be explicitly set to UTF8_MODE=false.

Type:BOOLEAN

Default:FALSE

Added in:Impala 4.1

Related Information

Query Join Performance
Compute Stats Statement
Impala Aggregate Functions

Virtual column

You can include a virtual column in a standard SELECT statement, select INPUT__FILE__NAME from <tablename>, to retrieve the data file name that stores the actual row in a table.

List of use cases

You can use this virtual column INPUT__FILE__NAME in multiple ways.

- To view the file names corresponding to each record.
- To extract the metadata using virtual columns if external source files ingested to HDFS contain metadata like a timestamp or source system identifier, and if you prefer to use it in subsequent operations.
- To extract the timestamp of the data derived from the fully qualified file name.
- To forward the technical timeline in a data warehouse scenario (for example, the "LoadDateTS" in a hybrid Data Vault architecture).

Supported operations

You can perform ceratin operations on this virtual column. The following list contains some of the supported operations:

- Invoke “Functions”
- Filter rows
- "GROUP BY" operation

Example

This example provides the file name corresponding to each record.

```
select INPUT__FILE__NAME, id, name from users where dt='20140210' group by
          INPUT__FILE__NAME;
```

Result

The above query returns results similar to the following

```
hdfs://localhost.localdomain:8020/user/hive/warehouse/users/users1.txt 2 use
r2
```

```
hdfs://localhost.localdomain:8020/user/hive/warehouse/users/users2.txt 42 jo
hn.doe
```

You can use the supported string functions to extract information from the URL.

Impala built-in functions

Impala supports several categories of built-in functions. These functions let you perform mathematical calculations, string manipulation, date calculations, and other kinds of data transformations directly in SQL statements.

Supported functions

The categories of built-in functions supported by Impala are:

- Impala mathematical functions on page 311
- Impala type conversion functions on page 339
- Impala date and time functions on page 347
- Impala conditional functions on page 366
- Impala string functions on page 371
- Impala aggregate functions on page 391.
- Impala analytic functions on page 424
- Impala bit functions on page 329
- Impala miscellaneous functions on page 388

The following is a complete list of built-in functions supported in Impala:

ABS
ACOS
ADD_MONTHS
ADDDATE
APPX_MEDIAN
ASCII
ASIN
ATAN
ATAN2
AVG
AVG - Analytic Function
BASE64DECODE
BASE64ENCODE
BITAND
BIN
BITNOT
BITOR
BITXOR
BTRIM
CASE
CASE WHEN
CAST
CEIL, CEILING, DCEIL
CHAR_LENGTH

CHR
COALESCE
CONCAT
CONCAT_WS
CONV
COS
COSH
COT
COUNT
COUNT - Analytic Function
COUNTSET
CUME_DIST
CURRENT_DATABASE
CURRENT_TIMESTAMP
DATE_ADD
DATE_PART
DATE_SUB
DATE_TRUNC
DATEDIFF
DAY
DAYNAME
DAYOFWEEK
DAYOFYEAR
DAYS_ADD
DAYS_SUB
DECODE
DEGREES
DENSE_RANK
E
EFFECTIVE_USER
EXP
EXTRACT
FACTORIAL
FIND_IN_SET
FIRST_VALUE
FLOOR, DFLOOR
FMOD
FNV_HASH
GET_JSON_OBJECT
FROM_UNIXTIME

FROM_TIMESTAMP
FROM_UTC_TIMESTAMP
GETBIT
GREATEST
GROUP_CONCAT
GROUP_CONCAT - Analytic Function
HEX
HOUR
HOURS_ADD
HOURS_SUB
IF
IFNULL
INITCAP
INSTR
INT_MONTHS_BETWEEN
IS_INF
IS_NAN
ISFALSE
ISNOTFALSE
ISNOTTRUE
ISNULL
ISTRUE
JARO_DISTANCE, JARO_DIST
JARO_SIMILARITY, JARO_SIM
JARO_WINKER_DISTANCE, JW_DST
JARO_WINKER_SIMILARITY, JW_SIM
LAG
LAST_VALUE
LEAD
LEAST
LEFT
LENGTH
LN
LOCATE
LOG
LOG10
LOG2
LOWER, LCASE
LPAD
LTRIM

MAX
MAX - Analytic Function
MAX_INT, MAX_TINYINT, MAX_SMALLINT, MAX_BIGINT
MICROSECONDS_ADD
MICROSECONDS_SUB
MILLISECOND
MILLISECONDS_ADD
MILLISECONDS_SUB
MIN
MIN - Analytic Function
MIN_INT, MIN_TINYINT, MIN_SMALLINT, MIN_BIGINT
MINUTE
MINUTES_ADD
MINUTES_SUB
MOD
MONTH
MONTHNAME
MONTHS_ADD
MONTHS_BETWEEN
MONTHS_SUB
MURMUR_HASH
NANOSECONDS_ADD
NANOSECONDS_SUB
NDV
NEGATIVE
NEXT_DAY
NONNULLVALUE
NOW
NTILE
NULLIF
NULLIFZERO
NULLVALUE
NVL
NVL2
OVER Clause
PARSE_URL
PERCENT_RANK
PI
PID
PMOD

POSITIVE
POW, POWER, DPOW, FPOW
PRECISION
QUARTER
QUOTIENT
RADIANS
RAND, RANDOM
RANK
REGEXP_ESCAPE
REGEXP_EXTRACT
REGEXP_LIKE
REGEXP_REPLACE
REPEAT
REPLACE
REVERSE
RIGHT
ROTATELEFT
ROTATERIGHT
ROUND, DROUND
ROW_NUMBER
RPAD
RTRIM
SCALE
SECOND
SECONDS_ADD
SECONDS_SUB
SETBIT
SHIFTLEFT
SHIFTRIGHT
SIGN
SIN
SINH
SLEEP
SPACE
SPLIT_PART
SQRT
STDDEV, STDDEV_SAMP, STDDEV_POP
STRLEFT
STRRIGHT
SUBDATE

SUBSTR, SUBSTRING
SUM
SUM - Analytic Function
TAN
TANH
TIMEOFDAY
TIMESTAMP_CMP
TO_DATE
TO_TIMESTAMP
TO_UTC_TIMESTAMP
TRANSLATE
TRIM
TRUNC
TRUNCATE, DTRUNC, TRUNC
TYPEOF
UNHEX
UNIX_TIMESTAMP
UPPER, UCASE
USER
UTC_TIMESTAMP
UUID
VARIANCE, VARIANCE_SAMP, VARIANCE_POP, VAR_SAMP, VAR_POP
VERSION
WEEKOFYEAR
WEEKS_ADD
WEEKS_SUB
WIDTH_BUCKET
YEAR
YEARS_ADD
YEARS_SUB
ZEROIFNULL

Impala mathematical functions

Mathematical functions, or arithmetic functions, perform numeric calculations that are typically more complex than basic addition, subtraction, multiplication, and division. For example, these functions include trigonometric, logarithmic, and base conversion operations.

Note: In Impala, exponentiation uses the POW() function rather than an exponentiation operator such as **.

Related information:

The mathematical functions operate mainly on these data types:

- INT
- BIGINT
- SMALLINT
- TINYINT
- DOUBLE
- FLOAT
- DECIMAL

For the operators that perform the standard operations such as addition, subtraction, multiplication, and division, see Arithmetic operators on page 94.

Functions that perform bitwise operations are explained in Impala bit functions on page 329.

Function reference:

Impala supports the following mathematical functions:

- ABS
- ACOS
- ASIN
- ATAN
- ATAN2
- BIN
- CEIL, CEILING, DCEIL
- CONV
- COS
- COSH
- COT
- DEGREES
- E
- EXP
- FACTORIAL
- FLOOR, DFLOOR
- FMOD
- FNV_HASH
- GREATEST
- HEX
- IS_INF
- IS_NAN
- LEAST
- LN
- LOG
- LOG10
- LOG2
- MAX_INT, MAX_TINYINT, MAX_SMALLINT, MAX_BIGINT
- MIN_INT, MIN_TINYINT, MIN_SMALLINT, MIN_BIGINT
- MOD
- MURMUR_HASH
- NEGATIVE
- PI
- PMOD
- POSITIVE
- POW, POWER, DPOW, FPOW

- PRECISION
- QUOTIENT
- RADIANS
- RAND, RANDOM
- ROUND, DROUND
- SCALE
- SIGN
- SIN
- SINH
- SQRT
- TAN
- TANH
- TRUNCATE, DTRUNC, TRUNC
- UNHEX
- WIDTH_BUCKET

ABS(numeric_type a)

Purpose: Returns the absolute value of the argument.

Return type: Same as the input value

Usage notes: Use this function to ensure all return values are positive. This is different than the positive() function, which returns its argument unchanged (even if the argument was negative).

ACOS(DOUBLE a)

Purpose: Returns the arccosine of the argument.

Return type: DOUBLE

ASIN(DOUBLE a)

Purpose: Returns the arcsine of the argument.

Return type: DOUBLE

ATAN(DOUBLE a)

Purpose: Returns the arctangent of the argument.

Return type: DOUBLE

ATAN(DOUBLE a, DOUBLE b)

Purpose: Returns the arctangent of the two arguments, with the signs of the arguments used to determine the quadrant of the result.

Return type: DOUBLE

BIN(BIGINT a)

Purpose: Returns the binary representation of an integer value, that is, a string of 0 and 1 digits.

Return type: STRING

CEIL(DOUBLE a), CEIL(DECIMAL(p,s) a), CEILING(DOUBLE a), CEILING(DECIMAL(p,s) a), DCEIL(DOUBLE a), DCEIL(DECIMAL(p,s) a)

Purpose: Returns the smallest integer that is greater than or equal to the argument.

Return type: Same as the input type

CONV(BIGINT n, INT from_base, INT to_base), CONV(STRING s, INT from_base, INT to_base)

Purpose: Returns a string representation of the first argument converted from from_base to to_base. The first argument can be specified as a number or a string. For example, conv(100, 2, 10) and conv('100', 2, 10) both return '4'.

Return type: STRING

Usage notes:

If to_base is negative, the first argument is treated as signed, and otherwise, it is treated as unsigned. For example:

- conv(-17, 10, -2) returns '-10001', -17 in base 2.
- conv(-17, 10, 10) returns '18446744073709551599'. -17 is interpreted as an unsigned, 2^64-17, and then the value is returned in base 10.

The function returns NULL when the following invalid arguments are specified:

- Any argument is NULL.
- from_base or to_base is below -36 or above 36.
- from_base or to_base is -1, 0, or 1.
- The first argument represents a positive number and from_base is a negative number.

If the first argument represents a negative number and from_base is a negative number, the function returns 0.

If the first argument represents a number larger than the maximum bigint, the function returns:

- The string representation of -1 in to_base if to_base is negative.
- The string representation of 18446744073709551615' (2^64 - 1) in to_base if to_base is positive.

If the first argument does not represent a valid number in from_base, e.g. 3 in base 2 or '1a23' in base 10, the digits in the first argument are evaluated from left-to-right and used if a valid digit in from_base. The invalid digit and the digits to the right are ignored.

For example:

- conv(445, 5, 10) is converted to conv(44, 5, 10) and returns '24'.
- conv('1a23', 10, 16) is converted to conv('1', 10 , 16) and returns '1'.

COS(DOUBLE a)

Purpose: Returns the cosine of the argument.

Return type: DOUBLE

COSH(DOUBLE a)

Purpose: Returns the hyperbolic cosine of the argument.

Return type: DOUBLE

COT(DOUBLE a)

Purpose: Returns the cotangent of the argument.

Return type: DOUBLE

Added in: Impala 2.3.0

DEGREES(DOUBLE a)

Purpose: Converts argument value from radians to degrees.

Return type: DOUBLE

E()

Purpose: Returns the mathematical constant e.

Return type: DOUBLE

EXP(DOUBLE a), DEXP(DOUBLE a)

Purpose: Returns the mathematical constant e raised to the power of the argument.

Return type: DOUBLE

FACTORIAL(integer_type a)

Purpose: Computes the factorial of an integer value. It works with any integer type.

Added in: Impala 2.3.0

Usage notes: You can use either the factorial() function or the ! operator. The factorial of 0 is 1. Likewise, the factorial() function returns 1 for any negative value. The maximum positive value for the input argument is 20; a value of 21 or greater overflows the range for a BIGINT and causes an error.

Return type: BIGINT

Added in: Impala 2.3.0

```
select factorial(5);
+--------------+
| factorial(5) |
+--------------+
| 120          |
+--------------+

select 5!;
+-----+
| 5!  |
+-----+
| 120 |
+-----+
```

FLOOR(DOUBLE a), FLOOR(DECIMAL(p,s) a), DFLOOR(DOUBLE a), DFLOOR(DECIMAL(p,s) a)

Purpose: Returns the largest integer that is less than or equal to the argument.

Return type: Same as the input type

FMOD(DOUBLE a, DOUBLE b), FMOD(FLOAT a, FLOAT b)

Purpose: Returns the modulus of a floating-point number.

Return type: FLOAT or DOUBLE, depending on type of arguments

Added in: Impala 1.1.1

Usage notes:

Because this function operates on DOUBLE or FLOAT values, it is subject to potential rounding errors for values that cannot be represented precisely. Prefer to use whole numbers, or values that you know can be represented precisely by the DOUBLE or FLOAT types.

Examples:

The following examples show equivalent operations with the fmod() function and the % arithmetic operator, for values not subject to any rounding error.

```
select fmod(10,3);
+-------------+
| fmod(10, 3) |
+-------------+
| 1           |
+-------------+

select fmod(5.5,2);
+--------------+
| fmod(5.5, 2) |
+--------------+
| 1.5          |
+--------------+
```

```
select 10 % 3;
+--------+
| 10 % 3 |
+--------+
| 1      |
+--------+
select 5.5 % 2;
+---------+
| 5.5 % 2 |
+---------+
| 1.5     |
+---------+
```

The following examples show operations with the fmod() function for values that cannot be represented precisely by the DOUBLE or FLOAT types, and thus are subject to rounding error. fmod(9.9,3.0) returns a value slightly different than the expected 0.9 because of rounding. fmod(9.9 ,3.3) returns a value quite different from the expected value of 0 because of rounding error during intermediate calculations.

```
select fmod(9.9,3.0);
+--------------------+
| fmod(9.9, 3.0)     |
+--------------------+
| 0.8999996185302734 |
+--------------------+

select fmod(9.9,3.3);
+-------------------+
| fmod(9.9, 3.3)    |
+-------------------+
| 3.299999713897705 |
+-------------------+
```

FNV_HASH(type v)

Purpose: Returns a consistent 64-bit value derived from the input argument, for convenience of implementing hashing logic in an application.

Return type: BIGINT

Usage notes:

You might use the return value in an application where you perform load balancing, bucketing, or some other technique to divide processing or storage.

Because the result can be any 64-bit value, to restrict the value to a particular range, you can use an expression that includes the ABS() function and the % (modulo) operator. For example, to produce a hash value in the range 0-9, you could use the expression ABS(FNV_HASH(x)) % 10.

This function implements the same algorithm that Impala uses internally for hashing, on systems where the CRC32 instructions are not available.

This function implements the Fowler–Noll–Vo hash function, in particular the FNV-1a variation. This is not a perfect hash function: some combinations of values could produce the same result value. It is not suitable for cryptographic use.

Similar input values of different types could produce different hash values, for example the same numeric value represented as SMALLINT or BIGINT, FLOAT or DOUBLE, or DECIMAL(5,2) or DECIMAL(20,5).

Examples:

```
[localhost:21000] > create table h (x int, s string);
```

```
[localhost:21000] > insert into h values (0, 'hello'), (1,'wor
ld'), (1234567890,'antidisestablishmentarianism');
[localhost:21000] > select x, fnv_hash(x) from h;
+------------+---------------------+
| x          | fnv_hash(x)         |
+------------+---------------------+
| 0          | -2611523532599129963 |
| 1          | 4307505193096137732  |
| 1234567890 | 3614724209955230832  |
+------------+---------------------+
[localhost:21000] > select s, fnv_hash(s) from h;
+-------------------------------+---------------------+
| s                             | fnv_hash(s)         |
+-------------------------------+---------------------+
| hello                         | 6414202926103426347 |
| world                         | 6535280128821139475 |
| antidisestablishmentarianism | -209330013948433970 |
+-------------------------------+---------------------+
[localhost:21000] > select s, abs(fnv_hash(s)) % 10 from h;
+-------------------------------+-------------------------+
| s                             | abs(fnv_hash(s)) % 10.0 |
+-------------------------------+-------------------------+
| hello                         | 8                       |
| world                         | 6                       |
| antidisestablishmentarianism | 4                       |
+-------------------------------+-------------------------+
```

For short argument values, the high-order bits of the result have relatively low entropy:

```
[localhost:21000] > create table b (x boolean);
[localhost:21000] > insert into b values (true), (true), (false),
 (false);
[localhost:21000] > select x, fnv_hash(x) from b;
+-------+---------------------+
| x     | fnv_hash(x)         |
+-------+---------------------+
| true  | 2062020650953872396 |
| true  | 2062020650953872396 |
| false | 2062021750465500607 |
| false | 2062021750465500607 |
+-------+---------------------+
```

Added in: Impala 1.2.2

GREATEST(BIGINT a[, BIGINT b ...]), GREATEST(DOUBLE a[, DOUBLE b ...]), GREATEST(DECIMAL(p,s) a[, DECIMAL(p,s) b ...]), GREATEST(STRING a[, STRING b ...]), GREATEST(TIMESTAMP a[, TIMESTAMP b ...])

Purpose: Returns the largest value from a list of expressions.

Return type: same as the initial argument value, except that integer values are promoted to BIGINT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

HEX(BIGINT a), HEX(STRING a)

Purpose: Returns the hexadecimal representation of an integer value, or of the characters in a string.

Return type: STRING

IS_INF(DOUBLE a)

Purpose: Tests whether a value is equal to the special value “inf”, signifying infinity.

Return type: BOOLEAN

Usage notes:

Infinity and NaN can be specified in text data files as inf and nan respectively, and Impala interprets them as these special values. They can also be produced by certain arithmetic expressions; for example, 1/0 returns Infinity and pow(-1, 0.5) returns NaN. Or you can cast the literal values, such as CAST('nan' AS DOUBLE) or CAST('inf' AS DOUBLE).

IS_NAN(DOUBLE a)

Purpose: Tests whether a value is equal to the special value “NaN”, signifying “not a number”.

Return type: BOOLEAN

Usage notes:

Infinity and NaN can be specified in text data files as inf and nan respectively, and Impala interprets them as these special values. They can also be produced by certain arithmetic expressions; for example, 1/0 returns Infinity and pow(-1, 0.5) returns NaN. Or you can cast the literal values, such as CAST('nan' AS DOUBLE) or CAST('inf' AS DOUBLE).

LEAST(BIGINT a[, BIGINT b ...]), LEAST(DOUBLE a[, DOUBLE b ...]), LEAST(DECIMAL(p,s) a[, DECIMAL(p,s) b ...]), LEAST(STRING a[, STRING b ...]), LEAST(TIMESTAMP a[, TIMESTAMP b ...])

Purpose: Returns the smallest value from a list of expressions.

Return type: same as the initial argument value, except that integer values are promoted to BIGINT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

LN(DOUBLE a), DLOG1(DOUBLE a)

Purpose: Returns the natural logarithm of the argument.

Return type: DOUBLE

LOG(DOUBLE base, DOUBLE a)

Purpose: Returns the logarithm of the second argument to the specified base.

Return type: DOUBLE

LOG10(DOUBLE a), DLOG10(DOUBLE a)

Purpose: Returns the logarithm of the argument to the base 10.

Return type: DOUBLE

LOG2(DOUBLE a)

Purpose: Returns the logarithm of the argument to the base 2.

Return type: DOUBLE

MAX_INT(), MAX_TINYINT(), MAX_SMALLINT(), MAX_BIGINT()

Purpose: Returns the largest value of the associated integral type.

Return type: The same as the integral type being checked.

Usage notes: Use the corresponding min_ and max_ functions to check if all values in a column are within the allowed range, before copying data or altering column definitions. If not, switch to the next higher integral type or to a DECIMAL with sufficient precision.

MIN_INT(), MIN_TINYINT(), MIN_SMALLINT(), MIN_BIGINT()

Purpose: Returns the smallest value of the associated integral type (a negative number).

Return type: The same as the integral type being checked.

Usage notes: Use the corresponding min_ and max_ functions to check if all values in a column are within the allowed range, before copying data or altering column definitions. If not, switch to the next higher integral type or to a DECIMAL with sufficient precision.

MOD(numeric_type a, same_type b)

Purpose: Returns the modulus of a number. Equivalent to the % arithmetic operator. Works with any size integer type, any size floating-point type, and DECIMAL with any precision and scale.

Return type: Same as the input value

Added in: Impala 2.20

Usage notes:

Because this function works with DECIMAL values, prefer it over fmod() when working with fractional values. It is not subject to the rounding errors that make fmod() problematic with floating-point numbers.

Query plans shows the MOD() function as the % operator.

Examples:

The following examples show how the mod() function works for whole numbers and fractional values, and how the % operator works the same way. In the case of mod(9.9,3), the type conversion for the second argument results in the first argument being interpreted as DOUBLE, so to produce an accurate DECIMAL result requires casting the second argument or writing it as a DECIMAL literal, 3.0.

```
select mod(10,3);
+-------------+
| mod(10, 3) |
+-------------+
| 1           |
+-------------+

select mod(5.5,2);
+--------------+
| mod(5.5, 2) |
+--------------+
| 1.5          |
+--------------+

select 10 % 3;
+--------+
| 10 % 3 |
+--------+
| 1      |
+--------+
select 5.5 % 2;
+---------+
| 5.5 % 2 |
+---------+
| 1.5     |
+---------+

select mod(9.9,3.3);
+---------------+
| mod(9.9, 3.3) |
+---------------+
| 0.0           |
+---------------+

select mod(9.9,3);
+--------------------+
| mod(9.9, 3)        |
+--------------------+
| 0.8999996185302734 |
+--------------------+
```

```
select mod(9.9, cast(3 as decimal(2,1)));
+-----------------------------------+
| mod(9.9, cast(3 as decimal(2,1))) |
+-----------------------------------+
| 0.9                               |
+-----------------------------------+

select mod(9.9,3.0);
+---------------+
| mod(9.9, 3.0) |
+---------------+
| 0.9           |
+---------------+
```

MURMUR_HASH(type v)

Purpose: Returns a consistent 64-bit value derived from the input argument, for convenience of implementing MurmurHash2 non-cryptographic hash function.

Return type: BIGINT

Usage notes:

You might use the return value in an application where you perform load balancing, bucketing, or some other technique to divide processing or storage. This function provides a good performance for all kinds of keys such as number, ascii string and UTF-8. It can be recommended as general-purpose hashing function.

Regarding comparison of murmur_hash with fnv_hash, murmur_hash is based on Murmur2 hash algorithm and fnv_hash function is based on FNV-1a hash algorithm. Murmur2 and FNV-1a can show very good randomness and performance compared with well known other hash algorithms, but Murmur2 slightly show better randomness and performance than FNV-1a.

Similar input values of different types could produce different hash values, for example the same numeric value represented as SMALLINT or BIGINT, FLOAT or DOUBLE, or DECIMAL(5,2) or DECIMAL(20,5).

Examples:

```
[localhost:21000] > create table h (x int, s string);
[localhost:21000] > insert into h values (0, 'hello'), (1,'wor
ld'), (1234567890,'antidisestablishmentarianism');
[localhost:21000] > select x, murmur_hash(x) from h;
+------------+----------------------+
| x          | murmur_hash(x)       |
+------------+----------------------+
| 0          | 6960269033020761575  |
| 1          | -7806115816811537 83 |
| 1234567890 | -5754914572385924334 |
+------------+----------------------+
[localhost:21000] > select s, murmur_hash(s) from h;
+-------------------------------+----------------------+
| s                             | murmur_hash(s)       |
+-------------------------------+----------------------+
| hello                         | 2191231550387646743  |
| world                         | 5568329560871645431  |
| antidisestablishmentarianism | -2261804666958489663 |
+-------------------------------+----------------------+
```

For short argument values, the high-order bits of the result have relatively higher entropy than fnv_hash:

```
[localhost:21000] > create table b (x boolean);
```

```
[localhost:21000] > insert into b values (true), (true), (false),
 (false);
[localhost:21000] > select x, murmur_hash(x) from b;
+-------+---------------------+
| x     | murmur_hash(x)      |
+-------+---------------------++
| true  | -5720937396023583481 |
| true  | -5720937396023583481 |
| false | 6351753276682545529  |
| false | 6351753276682545529  |
+-------+--------------------+-+
```

Added in: Impala 2.12.0

NEGATIVE(numeric_type a)

Purpose: Returns the argument with the sign reversed; returns a positive value if the argument was already negative.

Return type: Same as the input value

Usage notes: Use -abs(a) instead if you need to ensure all return values are negative.

PI()

Purpose: Returns the constant pi.

Return type: double

PMOD(BIGINT a, BIGINT b), PMOD(DOUBLE a, DOUBLE b)

Purpose: Returns the positive modulus of a number. Primarily for Hive SQL compatibility.

Return type: INT or DOUBLE, depending on type of arguments

Examples:

The following examples show how the fmod() function sometimes returns a negative value depending on the sign of its arguments, and the pmod() function returns the same value as fmod(), but sometimes with the sign flipped.

```
select fmod(-5,2);
+-------------+
| fmod(-5, 2) |
+-------------+
| -1          |
+-------------+

select pmod(-5,2);
+-------------+
| pmod(-5, 2) |
+-------------+
| 1           |
+-------------+

select fmod(-5,-2);
+--------------+
| fmod(-5, -2) |
+--------------+
| -1           |
+--------------+

select pmod(-5,-2);
+--------------+
| pmod(-5, -2) |
+--------------+
| -1           |
```

```
+--------------+

select fmod(5,-2);
+-------------+
| fmod(5, -2) |
+-------------+
| 1           |
+-------------+

select pmod(5,-2);
+-------------+
| pmod(5, -2) |
+-------------+
| -1          |
+-------------+
```

POSITIVE(numeric_type a)

Purpose: Returns the original argument unchanged (even if the argument is negative).

Return type: Same as the input value

Usage notes: Use abs() instead if you need to ensure all return values are positive.

POW(DOUBLE a, double p), POWER(DOUBLE a, DOUBLE p), DPOW(DOUBLE a, DOUBLE p), FPOW(DOUBLE a, DOUBLE p)

Purpose: Returns the first argument raised to the power of the second argument.

Return type: DOUBLE

PRECISION(numeric_expression)

Purpose: Computes the precision (number of decimal digits) needed to represent the type of the argument expression as a DECIMAL value.

Usage notes:

Typically used in combination with the scale() function, to determine the appropriate DECI MAL(*PRECISION*,*SCALE*) type to declare in a CREATE TABLE statement or CAST() function.

Return type: INT

Examples:

The following examples demonstrate how to check the precision and scale of numeric literals or other numeric expressions. Impala represents numeric literals in the smallest appropriate type. 5 is a TINYINT value, which ranges from -128 to 127, therefore 3 decimal digits are needed to represent the entire range, and because it is an integer value there are no fractional digits. 1.333 is interpreted as a DECIMAL value, with 4 digits total and 3 digits after the decimal point.

```
[localhost:21000] > select precision(5), scale(5);
+--------------+----------+
| precision(5) | scale(5) |
+--------------+----------+
| 3            | 0        |
+--------------+----------+
[localhost:21000] > select precision(1.333), scale(1.333);
+------------------+--------------+
| precision(1.333) | scale(1.333) |
+------------------+--------------+
| 4                | 3            |
+------------------+--------------+
[localhost:21000] > with t1 as
  ( select cast(12.34 as decimal(20,2)) x union select cast(1 as
 decimal(8,6)) x )
  select precision(x), scale(x) from t1 limit 1;
```

```
+--------------+----------+
| precision(x) | scale(x) |
+--------------+----------+
| 24           | 6        |
+--------------+----------+
```

QUOTIENT(BIGINT numerator, BIGINT denominator), QUOTIENT(DOUBLE numerator, DOUBLE denominator)

Purpose: Returns the first argument divided by the second argument, discarding any fractional part. Avoids promoting integer arguments to DOUBLE as happens with the / SQL operator. Also includes an overload that accepts DOUBLE arguments, discards the fractional part of each argument value before dividing, and again returns BIGINT. With integer arguments, this function works the same as the DIV operator.

Return type: BIGINT

RADIANS(DOUBLE a)

Purpose: Converts argument value from degrees to radians.

Return type: DOUBLE

RAND(), RAND(BIGINT seed), RANDOM(), RANDOM(BIGINT seed)

Purpose: Returns a random value between 0 and 1. After rand() is called with a seed argument, it produces a consistent random sequence based on the seed value.

Return type: DOUBLE

Usage notes: Currently, the random sequence is reset after each query, and multiple calls to rand () within the same query return the same value each time. For different number sequences that are different for each query, pass a unique seed value to each call to rand(). For example, select rand(unix_timestamp()) from ...

Examples:

The following examples show how rand() can produce sequences of varying predictability, so that you can reproduce query results involving random values or generate unique sequences of random values for each query. When rand() is called with no argument, it generates the same sequence of values each time, regardless of the ordering of the result set. When rand() is called with a constant integer, it generates a different sequence of values, but still always the same sequence for the same seed value. If you pass in a seed value that changes, such as the return value of the expression unix _timestamp(now()), each query will use a different sequence of random values, potentially more useful in probability calculations although more difficult to reproduce at a later time. Therefore, the final two examples with an unpredictable seed value also include the seed in the result set, to make it possible to reproduce the same random sequence later.

```
select x, rand() from three_rows;
+---+-----------------------+
| x | rand()                |
+---+-----------------------+
| 1 | 0.0004714746030380365 |
| 2 | 0.5895895192351144    |
| 3 | 0.4431900859080209    |
+---+-----------------------+
select x, rand() from three_rows order by x desc;
+---+-----------------------+
| x | rand()                |
+---+-----------------------+
| 3 | 0.0004714746030380365 |
| 2 | 0.5895895192351144    |
| 1 | 0.4431900859080209    |
+---+-----------------------+
```

```
select x, rand(1234) from three_rows order by x;
+---+----------------------+
| x | rand(1234)           |
+---+----------------------+
| 1 | 0.7377511392057646   |
| 2 | 0.009428468537250751 |
| 3 | 0.208117277924026    |
+---+----------------------+

select x, rand(1234) from three_rows order by x desc;
+---+----------------------+
| x | rand(1234)           |
+---+----------------------+
| 3 | 0.7377511392057646   |
| 2 | 0.009428468537250751 |
| 1 | 0.208117277924026    |
+---+----------------------+
select x, unix_timestamp(now()), rand(unix_timestamp(now()))
  from three_rows order by x;
+---+-----------------------+-----------------------------+
| x | unix_timestamp(now()) | rand(unix_timestamp(now())) |
+---+-----------------------+-----------------------------+
| 1 | 1440777752            | 0.002051228658320023        |
| 2 | 1440777752            | 0.5098743483004506          |
| 3 | 1440777752            | 0.9517714925817081          |
+---+-----------------------+-----------------------------+
select x, unix_timestamp(now()), rand(unix_timestamp(now()))
  from three_rows order by x desc;
+---+-----------------------+-----------------------------+
| x | unix_timestamp(now()) | rand(unix_timestamp(now())) |
+---+-----------------------+-----------------------------+
| 3 | 1440777761            | 0.9985985015512437          |
| 2 | 1440777761            | 0.3251255333074953          |
| 1 | 1440777761            | 0.02422675025846192         |
+---+-----------------------+-----------------------------+
```

ROUND(DOUBLE a), ROUND(DOUBLE a, INT d), ROUND(DECIMAL a, int_type d), DROUND(DOUBLE a), DROUND(DOUBLE a, INT d), DROUND(DECIMAL(p,s) a, int_type d)

Purpose: Rounds a floating-point value. By default (with a single argument), rounds to the nearest integer. Values ending in .5 are rounded up for positive numbers, down for negative numbers (that is, away from zero). The optional second argument specifies how many digits to leave after the decimal point; values greater than zero produce a floating-point return value rounded to the requested number of digits to the right of the decimal point.

Return type: Same as the input type

SCALE(numeric_expression)

Purpose: Computes the scale (number of decimal digits to the right of the decimal point) needed to represent the type of the argument expression as a DECIMAL value.

Usage notes:

Typically used in combination with the precision() function, to determine the appropriate DECIMAL(*PRECISION*,*SCALE*) type to declare in a CREATE TABLE statement or CAST() function.

Return type: int

Examples:

The following examples demonstrate how to check the precision and scale of numeric literals or other numeric expressions. Impala represents numeric literals in the smallest appropriate type. 5 is a TINYINT value, which ranges from -128 to 127, therefore 3 decimal digits are needed to represent

the entire range, and because it is an integer value there are no fractional digits. 1.333 is interpreted as a DECIMAL value, with 4 digits total and 3 digits after the decimal point.

```
[localhost:21000] > select precision(5), scale(5);
+--------------+----------+
| precision(5) | scale(5) |
+--------------+----------+
| 3            | 0        |
+--------------+----------+
[localhost:21000] > select precision(1.333), scale(1.333);
+------------------+--------------+
| precision(1.333) | scale(1.333) |
+------------------+--------------+
| 4                | 3            |
+------------------+--------------+
[localhost:21000] > with t1 as
  ( select cast(12.34 as decimal(20,2)) x union select cast(1 as
 decimal(8,6)) x )
  select precision(x), scale(x) from t1 limit 1;
+--------------+----------+
| precision(x) | scale(x) |
+--------------+----------+
| 24           | 6        |
+--------------+----------+
```

SIGN(DOUBLE a)

Purpose: Returns -1, 0, or 1 to indicate the signedness of the argument value.

Return type: INT

SIN(DOUBLE a)

Purpose: Returns the sine of the argument.

Return type: DOUBLE

SINH(DOUBLE a)

Purpose: Returns the hyperbolic sine of the argument.

Return type: DOUBLE

SQRT(DOUBLE a), DSQRT(DOUBLE a)

Purpose: Returns the square root of the argument.

Return type: DOUBLE

TAN(DOUBLE a)

Purpose: Returns the tangent of the argument.

Return type: DOUBLE

TANH(DOUBLE a)

Purpose: Returns the hyperbolic tangent of the argument.

Return type: DOUBLE

TRUNCATE(DOUBLE_or_DECIMAL a[, digits_to_leave]), DTRUNC(DOUBLE_or_DECIMAL a[, digits_to_leave]), TRUNC(DOUBLE_or_DECIMAL a[, digits_to_leave])

Purpose: Removes some or all fractional digits from a numeric value.

Arguments: With a single floating-point argument, removes all fractional digits, leaving an integer value. The optional second argument specifies the number of fractional digits to include in the return value, and only applies when the argument type is DECIMAL. A second argument of 0

truncates to a whole integer value. A second argument of negative N sets N digits to 0 on the left side of the decimal

Scale argument: The scale argument applies only when truncating DECIMAL values. It is an integer specifying how many significant digits to leave to the right of the decimal point. A scale argument of 0 truncates to a whole integer value. A scale argument of negative N sets N digits to 0 on the left side of the decimal point.

TRUNCATE(), DTRUNC(), and TRUNC() are aliases for the same function.

Return type: Same as the input type

Added in: The TRUNC() alias was added in Impala 2.10.0.

Usage notes:

You can also pass a DOUBLE argument, or DECIMAL argument with optional scale, to the DTRUNC() or TRUNCATE functions. Using the TRUNC() function for numeric values is common with other industry-standard database systems, so you might find such TRUNC() calls in code that you are porting to Impala.

The TRUNC() function also has a signature that applies to TIMESTAMP values.

Examples:

The following examples demonstrate the TRUNCATE() and DTRUNC() signatures for this function:

```
select truncate(3.45);
+----------------+
| truncate(3.45) |
+----------------+
| 3              |
+----------------+

select truncate(-3.45);
+-----------------+
| truncate(-3.45) |
+-----------------+
| -3              |
+-----------------+

select truncate(3.456,1);
+--------------------+
| truncate(3.456, 1) |
+--------------------+
| 3.4                |
+--------------------+

select dtrunc(3.456,1);
+------------------+
| dtrunc(3.456, 1) |
+------------------+
| 3.4              |
+------------------+

select truncate(3.456,2);
+--------------------+
| truncate(3.456, 2) |
+--------------------+
| 3.45               |
+--------------------+

select truncate(3.456,7);
+--------------------+
```

```
| truncate(3.456, 7) |
+--------------------+
| 3.4560000          |
+--------------------+
```

The following examples demonstrate using TRUNC() with DECIMAL or DOUBLE values, and with an optional scale argument for DECIMAL values. (The behavior is the same for the TRUNCATE() and DTRUNC() aliases also.)

```
create table t1 (d decimal(20,7));
-- By default, no digits to the right of the decimal point.
insert into t1 values (1.1), (2.22), (3.333), (4.4444), (5.5555
5);
select trunc(d) from t1 order by d;
+----------+
| trunc(d) |
+----------+
| 1        |
| 2        |
| 3        |
| 4        |
| 5        |
+----------+

-- 1 digit to the right of the decimal point.
select trunc(d,1) from t1 order by d;
+-------------+
| trunc(d, 1) |
+-------------+
| 1.1         |
| 2.2         |
| 3.3         |
| 4.4         |
| 5.5         |
+-------------+

-- 2 digits to the right of the decimal point,
-- including trailing zeroes if needed.
select trunc(d,2) from t1 order by d;
+-------------+
| trunc(d, 2) |
+-------------+
| 1.10        |
| 2.22        |
| 3.33        |
| 4.44        |
| 5.55        |
+-------------+
insert into t1 values (9999.9999), (8888.8888);

-- Negative scale truncates digits to the left
-- of the decimal point.
select trunc(d,-2) from t1 where d > 100 order by d;
+--------------+
| trunc(d, -2) |
+--------------+
| 8800         |
| 9900         |
+--------------+

-- The scale of the result is adjusted to match the
-- scale argument.
```

```
select trunc(d,2),
  precision(trunc(d,2)) as p,
  scale(trunc(d,2)) as s
from t1 order by d;
+-------------+----+---+
| trunc(d, 2) | p  | s |
+-------------+----+---+
| 1.10        | 15 | 2 |
| 2.22        | 15 | 2 |
| 3.33        | 15 | 2 |
| 4.44        | 15 | 2 |
| 5.55        | 15 | 2 |
| 8888.88     | 15 | 2 |
| 9999.99     | 15 | 2 |
+-------------+----+---+
```

```
create table dbl (d double);

insert into dbl values
  (1.1), (2.22), (3.333), (4.4444), (5.55555),
  (8888.8888), (9999.9999);

-- With double values, there is no optional scale argument.
select trunc(d) from dbl order by d;
+----------+
| trunc(d) |
+----------+
| 1        |
| 2        |
| 3        |
| 4        |
| 5        |
| 8888     |
| 9999     |
+----------+
```

UNHEX(STRING a)

Purpose: Returns a string of characters with ASCII values corresponding to pairs of hexadecimal digits in the argument.

Return type: STRING

WIDTH_BUCKET(DECIMAL expr, DECIMAL min_value, DECIMAL max_value, INT num_buckets)

Purpose: Returns the bucket number in which the expr value would fall in the histogram where its range between min_value and max_value is divided into num_buckets buckets of identical sizes.

The function returns:

- NULL if any argument is NULL.
- 0 if expr < min_value.
- num_buckets + 1 if expr >= max_val.
- If none of the above, the bucket number where expr falls.

Arguments:The following rules apply to the arguments.

- min_val is the minimum value of the histogram range.
- max_val is the maximum value of the histogram range.
- num_buckets must be greater than 0.
- min_value must be less than max_value.

Usage notes:

Each bucket contains values equal to or greater than the base value of that bucket and less than the base value of the next bucket. For example, with width_bucket(8, 1, 10, 3), the bucket ranges are actually the 0th "underflow bucket" with the range (-infinity to 0.999...), (1 to 3.999...), (4, to 6.999...), (7 to 9.999...), and the "overflow bucket" with the range (10 to infinity).

Return type: BIGINT

Added in: Impala 3.1.

Examples:

The below function creates 3 buckets between the range of 1 and 20 with the bucket width of 6.333, and returns 2 for the bucket #2 where the value 8 falls in:

```
WIDTH_BUCKET(8, 1, 20, 3)
```

The below statement returns a list of accounts with the energy spending and the spending bracket each account falls in, between 0 and 11. Bucket 0 (underflow bucket) will be assigned to the accounts whose energy spendings are less than $50. Bucket 11 (overflow bucket) will be assigned to the accounts whose energy spendings are more than or equal to $1000.

```
SELECT account, invoice_amount, WIDTH_BUCKET(invoice_amount,50,1
000,10)
FROM invoices_june2018
ORDER BY 3;
```

Related Information

Mathematical constant e

Factorial

Fowler–Noll–Vo hash function

Natural logarithm

MurmurHash2

More Hash Function Tests

Hash functions: An empirical comparison

Impala bit functions

Bit manipulation functions perform bitwise operations involved in scientific processing or computer science algorithms. For example, these functions include setting, clearing, or testing bits within an integer value, or changing the positions of bits with or without wraparound.

If a function takes two integer arguments that are required to be of the same type, the smaller argument is promoted to the type of the larger one if required. For example, BITAND(1,4096) treats both arguments as SMALLINT, because 1 can be represented as a TINYINT but 4096 requires a SMALLINT.

Remember that all Impala integer values are signed. Therefore, when dealing with binary values where the most significant bit is 1, the specified or returned values might be negative when represented in base 10.

Whenever any argument is NULL, either the input value, bit position, or number of shift or rotate positions, the return value from any of these functions is also NULL

Related information:

The bit functions operate on all the integral data types:

- INT
- BIGINT
- SMALLINT
- TINYINT

Function reference:

Impala supports the following bit functions:

- BITAND
- BITNOT
- BITOR
- BITXOR
- COUNTSET
- GETBIT
- ROTATELEFT
- ROTATERIGHT
- SETBIT
- SHIFTLEFT
- SHIFTRIGHT

BITAND(integer_type a, same_type b)

Purpose: Returns an integer value representing the bits that are set to 1 in both of the arguments. If the arguments are of different sizes, the smaller is promoted to the type of the larger.

Usage notes: The BITAND() function is equivalent to the & binary operator.

Return type: Same as the input value

Examples:

The following examples show the results of ANDing integer values. 255 contains all 1 bits in its lowermost 7 bits. 32767 contains all 1 bits in its lowermost 15 bits. You can use the bin() function to check the binary representation of any integer value, although the result is always represented as a 64-bit value. If necessary, the smaller argument is promoted to the type of the larger one.

```
select bitand(255, 32767); /* 0000000011111111 & 0111111111111111
 */
+--------------------+
| bitand(255, 32767) |
+--------------------+
| 255                |
+--------------------+
select bitand(32767, 1); /* 0111111111111111 & 0000000000000001
 */
+-------------------+
| bitand(32767, 1) |
+-------------------+
| 1                 |
+-------------------+

select bitand(32, 16); /* 00010000 & 00001000 */
+----------------+
| bitand(32, 16) |
+----------------+
| 0              |
+----------------+

select bitand(12,5); /* 00001100 & 00000101 */
+---------------+
| bitand(12, 5) |
+---------------+
| 4             |
+---------------+
select bitand(-1,15); /* 11111111 & 00001111 */
+----------------+
| bitand(-1, 15) |
```

```
+----------------+
| 15             |
+----------------+
```

BITNOT(integer_type a)

Purpose: Inverts all the bits of the input argument.

Usage notes: The BITNOT() function is equivalent to the ~ unary operator.

Examples:

These examples illustrate what happens when you flip all the bits of an integer value. The sign always changes. The decimal representation is one different between the positive and negative values.

```
select bitnot(127); /* 01111111 -> 10000000 */
+-------------+
| bitnot(127) |
+-------------+
| -128        |
+-------------+

select bitnot(16); /* 00010000 -> 11101111 */
+------------+
| bitnot(16) |
+------------+
| -17        |
+------------+

select bitnot(0); /* 00000000 -> 11111111 */
+-----------+
| bitnot(0) |
+-----------+
| -1        |
+-----------+

select bitnot(-128); /* 10000000 -> 01111111 */
+--------------+
| bitnot(-128) |
+--------------+
| 127          |
+--------------+
```

BITOR(integer_type a, same_type b)

Purpose: Returns an integer value representing the bits that are set to 1 in either of the arguments. If the arguments are of different sizes, the smaller is promoted to the type of the larger.

Usage notes: The BITOR() function is equivalent to the | binary operator.

Return type: Same as the input value

Examples:

The following examples show the results of ORing integer values.

```
select bitor(1,4); /* 00000001 | 00000100 */
+-------------+
| bitor(1, 4) |
+-------------+
| 5           |
+-------------+

select bitor(16,48); /* 00001000 | 00011000 */
+---------------+
```

```
| bitor(16, 48) |
+---------------+
| 48            |
+---------------+

select bitor(0,7); /* 00000000 | 00000111 */
+-------------+
| bitor(0, 7) |
+-------------+
| 7           |
+-------------+
```

BITXOR(integer_type a, same_type b)

Purpose: Returns an integer value representing the bits that are set to 1 in one but not both of the arguments. If the arguments are of different sizes, the smaller is promoted to the type of the larger.

Usage notes: The BITXOR() function is equivalent to the ^ binary operator.

Return type: Same as the input value

Examples:

The following examples show the results of XORing integer values. XORing a non-zero value with zero returns the non-zero value. XORing two identical values returns zero, because all the 1 bits from the first argument are also 1 bits in the second argument. XORing different non-zero values turns off some bits and leaves others turned on, based on whether the same bit is set in both arguments.

```
select bitxor(0,15); /* 00000000 ^ 00001111 */
+---------------+
| bitxor(0, 15) |
+---------------+
| 15            |
+---------------+

select bitxor(7,7); /* 00000111 ^ 00000111 */
+--------------+
| bitxor(7, 7) |
+--------------+
| 0            |
+--------------+

select bitxor(8,4); /* 00001000 ^ 00000100 */
+--------------+
| bitxor(8, 4) |
+--------------+
| 12           |
+--------------+

select bitxor(3,7); /* 00000011 ^ 00000111 */
+--------------+
| bitxor(3, 7) |
+--------------+
| 4            |
+--------------+
```

COUNTSET(integer_type a [, INT zero_or_one])

Purpose: By default, returns the number of 1 bits in the specified integer value. If the optional second argument is set to zero, it returns the number of 0 bits instead.

Usage notes:

In discussions of information theory, this operation is referred to as the “population count” or “popcount”.

Return type: Same as the input value

Examples:

The following examples show how to count the number of 1 bits in an integer value.

```
select countset(1); /* 00000001 */
+-------------+
| countset(1) |
+-------------+
| 1           |
+-------------+

select countset(3); /* 00000011 */
+-------------+
| countset(3) |
+-------------+
| 2           |
+-------------+

select countset(16); /* 00010000 */
+--------------+
| countset(16) |
+--------------+
| 1            |
+--------------+

select countset(17); /* 00010001 */
+--------------+
| countset(17) |
+--------------+
| 2            |
+--------------+

select countset(7,1); /* 00000111 = 3 1 bits; the function counts
 1 bits by default */
+----------------+
| countset(7, 1) |
+----------------+
| 3              |
+----------------+

select countset(7,0); /* 00000111 = 5 0 bits; third argument can
only be 0 or 1 */
+----------------+
| countset(7, 0) |
+----------------+
| 5              |
+----------------+
```

GETBIT(integer_type a, INT position)

Purpose: Returns a 0 or 1 representing the bit at a specified position. The positions are numbered right to left, starting at zero. The position argument cannot be negative.

Usage notes:

When you use a literal input value, it is treated as an 8-bit, 16-bit, and so on value, the smallest type that is appropriate. The type of the input value limits the range of the positions. Cast the input value to the appropriate type if you need to ensure it is treated as a 64-bit, 32-bit, and so on value.

Return type: Same as the input value

Examples:

The following examples show how to test a specific bit within an integer value.

```
select getbit(1,0); /* 00000001 */
+--------------+
| getbit(1, 0) |
+--------------+
| 1            |
+--------------+
select getbit(16,1) /* 00010000 */
+---------------+
| getbit(16, 1) |
+---------------+
| 0             |
+---------------+

select getbit(16,4) /* 00010000 */
+---------------+
| getbit(16, 4) |
+---------------+
| 1             |
+---------------+
select getbit(16,5) /* 00010000 */
+---------------+
| getbit(16, 5) |
+---------------+
| 0             |
+---------------+

select getbit(-1,3); /* 11111111 */
+---------------+
| getbit(-1, 3) |
+---------------+
| 1             |
+---------------+

select getbit(-1,25); /* 11111111 */
ERROR: Invalid bit position: 25

select getbit(cast(-1 as int),25); /* 11111111111111111111111111
111111 */
+-----------------------------+
| getbit(cast(-1 as int), 25) |
+-----------------------------+
| 1                           |
+-----------------------------+
```

ROTATELEFT(integer_type a, INT positions)

Purpose: Rotates an integer value left by a specified number of bits. As the most significant bit is taken out of the original value, if it is a 1 bit, it is “rotated” back to the least significant bit. Therefore, the final value has the same number of 1 bits as the original value, just in different positions. In computer science terms, this operation is a “circular shift”.

Usage notes:

Specifying a second argument of zero leaves the original value unchanged. Rotating a -1 value by any number of positions still returns -1, because the original value has all 1 bits and all the 1 bits are preserved during rotation. Similarly, rotating a 0 value by any number of positions still returns 0. Rotating a value by the same number of bits as in the value returns the same value. Because this is a circular operation, the number of positions is not limited to the number of bits in the input value.

For example, rotating an 8-bit value by 1, 9, 17, and so on positions returns an identical result in each case.

Return type: Same as the input value

Examples:

```
select rotateleft(1,4); /* 00000001 -> 00010000 */
+------------------+
| rotateleft(1, 4) |
+------------------+
| 16               |
+------------------+
select rotateleft(-1,155); /* 11111111 -> 11111111 */
+----------------------+
| rotateleft(-1, 155) |
+----------------------+
| -1                   |
+----------------------+

select rotateleft(-128,1); /* 10000000 -> 00000001 */
+----------------------+
| rotateleft(-128, 1) |
+----------------------+
| 1                    |
+----------------------+

select rotateleft(-127,3); /* 10000001 -> 00001100 */
+----------------------+
| rotateleft(-127, 3) |
+----------------------+
| 12                   |
+----------------------+
```

ROTATERIGHT(integer_type a, INT positions)

Purpose: Rotates an integer value right by a specified number of bits. As the least significant bit is taken out of the original value, if it is a 1 bit, it is “rotated” back to the most significant bit. Therefore, the final value has the same number of 1 bits as the original value, just in different positions. In computer science terms, this operation is a “circular shift”.

Usage notes:

Specifying a second argument of zero leaves the original value unchanged. Rotating a -1 value by any number of positions still returns -1, because the original value has all 1 bits and all the 1 bits are preserved during rotation. Similarly, rotating a 0 value by any number of positions still returns 0. Rotating a value by the same number of bits as in the value returns the same value. Because this is a circular operation, the number of positions is not limited to the number of bits in the input value. For example, rotating an 8-bit value by 1, 9, 17, and so on positions returns an identical result in each case.

Return type: Same as the input value

Examples:

```
select rotateright(16,4); /* 00010000 -> 00000001 */
+---------------------+
| rotateright(16, 4) |
+---------------------+
| 1                   |
+---------------------+
select rotateright(-1,155); /* 11111111 -> 11111111 */
+-----------------------+
```

```
| rotateright(-1, 155) |
+----------------------+
| -1                   |
+----------------------+

select rotateright(-128,1); /* 10000000 -> 01000000 */
+----------------------+
| rotateright(-128, 1) |
+----------------------+
| 64                   |
+----------------------+

select rotateright(-127,3); /* 10000001 -> 00110000 */
+----------------------+
| rotateright(-127, 3) |
+----------------------+
| 48                   |
+----------------------+
```

SETBIT(integer_type a, INT position [, INT zero_or_one])

Purpose: By default, changes a bit at a specified position to a 1, if it is not already. If the optional third argument is set to zero, the specified bit is set to 0 instead.

Usage notes:

If the bit at the specified position was already 1 (by default) or 0 (with a third argument of zero), the return value is the same as the first argument. The positions are numbered right to left, starting at zero. (Therefore, the return value could be different from the first argument even if the position argument is zero.) The position argument cannot be negative.

When you use a literal input value, it is treated as an 8-bit, 16-bit, and so on value, the smallest type that is appropriate. The type of the input value limits the range of the positions. Cast the input value to the appropriate type if you need to ensure it is treated as a 64-bit, 32-bit, and so on value.

Return type: Same as the input value

Examples:

```
select setbit(0,0); /* 00000000 -> 00000001 */
+--------------+
| setbit(0, 0) |
+--------------+
| 1            |
+--------------+
select setbit(0,3); /* 00000000 -> 00001000 */
+--------------+
| setbit(0, 3) |
+--------------+
| 8            |
+--------------+

select setbit(7,3); /* 00000111 -> 00001111 */
+--------------+
| setbit(7, 3) |
+--------------+
| 15           |
+--------------+

select setbit(15,3); /* 00001111 -> 00001111 */
+---------------+
| setbit(15, 3) |
+---------------+
| 15            |
```

```
+---------------+

select setbit(0,32); /* By default, 0 is a TINYINT with only 8
 bits. */
ERROR: Invalid bit position: 32

select setbit(cast(0 as bigint),32); /* For BIGINT, the position
can be 0..63. */
+-------------------------------+
| setbit(cast(0 as bigint), 32) |
+-------------------------------+
| 4294967296                    |
+-------------------------------+

select setbit(7,3,1); /* 00000111 -> 00001111; setting to 1 is t
he default */
+-----------------+
| setbit(7, 3, 1) |
+-----------------+
| 15              |
+-----------------+

select setbit(7,2,0); /* 00000111 -> 00000011; third argument of
0 clears instead of sets */
+-----------------+
| setbit(7, 2, 0) |
+-----------------+
| 3               |
+-----------------+
```

SHIFTLEFT(integer_type a, INT positions)

Purpose: Shifts an integer value left by a specified number of bits. As the most significant bit is taken out of the original value, it is discarded and the least significant bit becomes 0. In computer science terms, this operation is a “logical shift”.

Usage notes:

The final value has either the same number of 1 bits as the original value, or fewer. Shifting an 8-bit value by 8 positions, a 16-bit value by 16 positions, and so on produces a result of zero.

Specifying a second argument of zero leaves the original value unchanged. Shifting any value by 0 returns the original value. Shifting any value by 1 is the same as multiplying it by 2, as long as the value is small enough; larger values eventually become negative when shifted, as the sign bit is set. Starting with the value 1 and shifting it left by N positions gives the same result as 2 to the Nth power, or pow(2,*N*).

Return type: Same as the input value

Examples:

```
select shiftleft(1,0); /* 00000001 -> 00000001 */
+-----------------+
| shiftleft(1, 0) |
+-----------------+
| 1               |
+-----------------+

select shiftleft(1,3); /* 00000001 -> 00001000 */
+-----------------+
| shiftleft(1, 3) |
+-----------------+
| 8               |
+-----------------+
```

```
select shiftleft(8,2); /* 00001000 -> 00100000 */
+-----------------+
| shiftleft(8, 2) |
+-----------------+
| 32              |
+-----------------+
select shiftleft(127,1); /* 01111111 -> 11111110 */
+-------------------+
| shiftleft(127, 1) |
+-------------------+
| -2                |
+-------------------+

select shiftleft(127,5); /* 01111111 -> 11100000 */
+-------------------+
| shiftleft(127, 5) |
+-------------------+
| -32               |
+-------------------+

select shiftleft(-1,4); /* 11111111 -> 11110000 */
+------------------+
| shiftleft(-1, 4) |
+------------------+
| -16              |
+------------------+
```

SHIFTRIGHT(integer_type a, INT positions)

Purpose: Shifts an integer value right by a specified number of bits. As the least significant bit is taken out of the original value, it is discarded and the most significant bit becomes 0. In computer science terms, this operation is a “logical shift”.

Usage notes:

Therefore, the final value has either the same number of 1 bits as the original value, or fewer. Shifting an 8-bit value by 8 positions, a 16-bit value by 16 positions, and so on produces a result of zero.

Specifying a second argument of zero leaves the original value unchanged. Shifting any value by 0 returns the original value. Shifting any positive value right by 1 is the same as dividing it by 2. Negative values become positive when shifted right.

Return type: Same as the input value

Examples:

```
select shiftright(16,0); /* 00010000 -> 00010000 */
+-------------------+
| shiftright(16, 0) |
+-------------------+
| 16                |
+-------------------+

select shiftright(16,4); /* 00010000 -> 00000001 */
+-------------------+
| shiftright(16, 4) |
+-------------------+
| 1                 |
+-------------------+

select shiftright(16,5); /* 00010000 -> 00000000 */
+-------------------+
```

```
| shiftright(16, 5) |
+-------------------+
| 0                 |
+-------------------+
select shiftright(-1,1); /* 11111111 -> 01111111 */
+-------------------+
| shiftright(-1, 1) |
+-------------------+
| 127               |
+-------------------+

select shiftright(-1,5); /* 11111111 -> 00000111 */
+-------------------+
| shiftright(-1, 5) |
+-------------------+
| 7                 |
+-------------------+
```

Related Information

Hamming weight (population count)

Circular shift

Logical shift

Impala type conversion functions

Conversion functions are usually used in combination with other functions, to explicitly pass the expected data types. Impala has strict rules regarding data types for function parameters. For example, Impala does not automatically convert a DOUBLE value to FLOAT, a BIGINT value to INT, or other conversion where precision could be lost or overflow could occur. Also, for reporting or dealing with loosely defined schemas in big data contexts, you might frequently need to convert values to or from the STRING type.

Note: Although in Impala 2.3, the SHOW FUNCTIONS output for database _IMPALA_BUILTINS contains some function signatures matching the pattern castto*, these functions are not intended for public use and are expected to be deprecated in future.

Function reference:

Impala supports the following type conversion functions:

- CAST
- TYPEOF

CAST(expression AS type)

Purpose: Returns *EXPRESSION* converted to the *TYPE* data type.

If the *EXPRESSION* value is of a type that cannot be converted to the target *TYPE*:

- Of DECIMAL, DATE, and BOOLEAN, the function returns an error.
- Of all other types, the function returns NULL.

Usage notes:

Use CAST when passing a column value or literal to a function that expects a parameter with a different type. Frequently used in SQL operations such as CREATE TABLE AS SELECT and INSERT ... VALUES to ensure that values from various sources are of the appropriate type for the destination columns.

Where practical, do a one-time CAST() operation during the ingestion process to make each column into the appropriate type, rather than using many CAST() operations in each query; doing type

conversions for each row during each query can be expensive for tables with millions or billions of rows.

The way this function deals with time zones when converting to or from TIMESTAMP values is affected by the ##use_local_tz_for_unix_timestamp_conversions startup flag for the `impalad` daemon. See TIMESTAMP Data type for details about how Impala handles time zone considerations for the TIMESTAMP data type.

CAST(expression AS type FORMAT pattern)

Purpose: Returns *EXPRESSION* converted to the *TYPE* data type based on the *PATTERN* format string. This signature of CAST() with the FORMAT clause is supported only for casts between STRING / CHAR / VARCHAR types and TIMESTAMP / DATE types.

Note: The patterns allowed in the FORMAT clause support ISO SQL:2016 standard patterns. Those patterns are not the same format patterns used with the other Impala conversion functions, e.g. TO_TIMESTAMP() and FROM_TIMESTAMP().

The following rules apply to *PATTERN*. Any exceptions to these rules are noted in the Details column of the table below.

- *PATTERN* is a case-insensitive STRING unless noted otherwise in the table below.
- If *PATTERN* is NULL, an empty string, or a number, an error returns.
- A fewer digits in *EXPRESSION* than specified in the *PATTERN* is accepted if a separator is correctly specified in the *PATTERN*. For example, CAST('5-01-2017' AS DATE FORMAT 'MM-dd-yyyy') returns DATE'2017-05-01'.
- If fewer number of digits are in *EXPRESSION* than specified in the *PATTERN*, the current date is used to complete the year pattern. For example, CAST('19/05' AS DATE FORMAT 'YYY Y/MM') will return DATE'2019-05-01' when executed on August 8, 2019.

The following format patterns are supported in the FORMAT clause.

Pattern	Description	Details
YYYY	4-digit year.	
YYY	Last 3 digits of a year.	
YY	Last 2 digits of a year.	
Y	Last digit of a year	
RRRR	4-digit round year	If 1, 3 or 4-digit year values are provided in *EXPRESSION*, treated as YYYY. If 2-digit years are provided in *EXPRESSION*, treated as RR. For date/time to string conversions, treated as YYYY. If YYYY, YYY, YY, Y, or RR is given in the same *PATTERN* for a string to date/time conversion, an error returns.

Pattern	Description	Details

RR	2-digit round year.	For date/time to string conversion, same as YY.
		For string to date/time conversions, the first 2 digits of the year in the return value depends on the specified two-digit year and the last two digits of the current year as follows:
		• If the specified 2-digit year is 00 to 49:
		  • If the last 2 digits of the current year are 00 to 49, the returned year has the same first 2 digits as the current year.
		  • If the last 2 digits of the current year are 50 to 99, the first 2 digits of the returned year are 1 greater than the first 2 digits of the current year.
		• If the specified 2-digit year is 50 to 99:
		  • If the last 2 digits of the current year are 00 to 49, the first 2 digits of the returned year are 1 less than the first 2 digits of the current year.
		  • If the last 2 digits of the current year are 50 to 99, the returned year has the same first 2 digits as the current year.
		If YYYY, YYY, YY, Y, or RR is given in the same *PATTERN* for a string to date/time conversion, an error returns.
		If 1-digit year values are provided in *EXPRESSION*, it is treated as YYYY.

Q	Quarter of year (1-4)	Not supported in string to date/time conversions.
MM	Month	In date/time to string conversions, 1-digit months are prefixed with a zero.
MONTH Month month	Full month name	In string to date/time conversions: • Converts a textual month name to a 2-digit month number. • The input strings are expected without trailing spaces, e.g. "June", "August". • If the FX modifier is active for the token, a 9-character-long full month name with trailing spaces is expected for the input string. In date/time to string conversions: • Converts to a 9-character-long full month name with trailing spaces. • If the FM modifier is active for this token, converts to a month name without trailing spaces. This pattern token is case sensitive. MONTH, Month, month would require different casing of the input in string to date/time conversions and would produce cased outputs in date/time to string conversions, For example, MONTH, Month, month would convert respectively JULY, July, and july.
MON Mon mon	Abbreviated 3-character month name	This pattern token is case sensitive. MON, Mon, mon would require different casing of the input in string to date/time conversions and would produce cased outputs in date/time to string conversions, For example, MON, Mon, mon would convert respectively JUL, Jul, and jul.
WW	Week of year (1-53)	Not supported in string to date/time conversions. The 1st week begins on January 1st and ends on January 7th.
W	Week of month (1-5)	Week of month (1-5)
D	Day of week (1-7)	Day of week (1-7)
DD	Day of month (1-31)	In date/time to string conversions, a 1-digit day is prefixed with a zero.
DDD	Day of year (1-366)	In string to date/time conversions, providing MM and DD along with DDD results an error, e.g. YYYY-MM-DDD.

Pattern	Description	Details
DAY Day day	Day name	Not supported in string to date/time conversions. In date/time to string conversions: • Converts to a 9-character-long full day name with trailing spaces. • If the FM modifier is active for this token, converts to a day name without trailing spaces. This pattern token is case sensitive. DAY, Day, day would produce cased outputs in date/time to string conversions, For example, DAY, Day, day would convert respectively MONDAY, Monday, and monday.
	DY Dy dy	Not supported in string to date/time conversions. This pattern token is case sensitive. DY, Dy, dy would produce cased outputs in date/time to string conversions, For example, DY, Dy, dy would convert respectively MON, Mon, and mon.
HH HH12	Hour of day (1-12)	In date/time to string conversions, 1-digit hours are prefixed with a zero. If provided hour in *EXPRESSION* is not between 1 and 12, returns an error. If no AM/PM is provided in *EXPRESSION*, the default is AM.
HH HH12	Hour of day (1-12)	In date/time to string conversions, 1-digit hours are prefixed with a zero. If provided hour in *EXPRESSION* is not between 1 and 12, returns an error. If no AM/PM is provided in *EXPRESSION*, the default is AM.
HH24	Hour of day (0-23)	In string to date/time conversions, if HH12, AM, PM are given in the same *PATTERN*, an error returns.
MI	Minute of hour (0-59)	In date/time to string conversions, 1-digit minutes are prefixed with a zero.
SS	Second of minute (0-59)	In date/time to string conversions, 1-digit seconds are prefixed with a zero.
SSSSS	Second of Day (0-86399)	In string to timestamp conversions, if SS, HH, HH12, HH24, MI, AM/PM are given in the same *PATTERN*, an error returns.
FF FF1, ..., FF9	Fractional second	A number, 1 to 9, can be used to indicate the number of digits. FF specifies a 9 digits fractional second.
AM PM A.M. P.M.	Meridiem indicator	For date/time to string conversions, AM and PM are treated as synonyms. For example, casting '2019-01-01 11:00 am' to TIMESTAMP with the 'YYYY-MM-DD HH12:MI PM' pattern returns 01-JAN-19 11.00.00.000000 AM. For string to date/time conversion, HH24 in the same *PATTERN* returns an error.
TZH	Timezone offset hour	An optional sign, + or -, and 2 digits for the value of signed numbers are allowed for the source *EXPRESSION*, e.g. +10, -05, 04. The allowed values are from -15 to +5.
TZM	Timezone offset minute	Unsigned numbers are allowed for the source *EXPRESSION*. The allowed values are from 0 to 59.

Pattern	Description	Details
- . / , ' ; : <space>	Separator	For string to date/time conversions, any separator character in the *PATTERN* string would match any separator character in the input *EXPRESSION*. For example, CAST(“20191010” AS DATE FORMAT “YYYY-MM-DD”) returns an error, but CAST("2019-.;10 10" AS DATE FORMAT "YYYY-MM-DD") succeeds.
T	Separates the date from the time.	This pattern is used for accepting ISO 8601 date/time formats. Example: YYYY-MM-DDTHH24:MI:SS.FF9Z
Z	Indicates the zero hour offset from UTC.	This pattern is used for accepting ISO 8601 date/time formats.
FX	Forces an exact match between the format pattern, *PATTERN*, and the input argument, *EXPRESSION*.	Must be specified at the beginning of the format *PATTERN* and is valid for the whole *PATTERN*. In string to date/time conversions: • Forces strict separator matching, including the separator character. • Expects all the tokens to have the maximum possible length. In date/time to string conversions, the outputs are padded as follows: • Numeric outputs are left padded with zeros. • Text outputs are right padded with spaces.
FM	Overrides FX.	In a date/time to string conversions, FM suppresses blank padding for the element immediately following the FM in the *PATTERN* string, e.g. "2010-1-9". In string to date/time conversions, FM is used to override the effect of FX for certain tokens as follows: • The length of the token in the input argument can be shorter than the max length of that token type if followed by a separator, e.g. 1-digit month, less than 4-digit year. • FM modifier skips the separators and affects the next non-separator token. For example, CAST('1999-10-2' AS TIMESTAMP FORMAT 'FXYYYY-MMFM-DD') returns 1999-10-02 00:00:00 as FM is applied to DD and not to the separator(-).

Pattern	Description	Details
Free text	Nested string	You can specify a free text with surrounding double quotes (") in *PATTERN* where the same text is expected in the input argument without surrounding double quotes. In date/time to string conversions, the string is included in the output with the characters' case preserved. In string to date/time conversions, the nested string is not included in the resulting date/time object. However, the nested string has to match the related part of the input string, including the case. The following rules apply: • If the *PATTERN* is surrounded by double quotes, double quotes surrounding the free text token must be escaped with a single backslash (\"). For example: CAST('"Date:"2019-11-10' AS DATE FORMAT "\"Date:\"YYYY-MM-DD") • If the *PATTERN* is surrounded by double quotes, and there is an escaped double quotes inside a text token that is itself surrounded by escaped double quotes, escape the double quotes in the free text token with a triple backslash: (\\\") For example: CAST("1985 some \"text 11-28" AS DATE"' FORMAT "YYYY\" some \\\"text \"MM-DD") • If the *PATTERN* is surrounded by single quotes, the free text token can be surrounded by (non-escaped) double quotes. To include double quotes within the free text token, those double quotes must be escaped by a single backslash. For example: CAST('"Date:"2019-11-10' AS DATE FORMAT '"\"Date:\""YYYY-MM-DD') • Literal single quotes/apostrophes (') in the nested string must be escaped with a single backslash if the whole pattern string is delimited by single quotes. • If a free text token contains any separator characters in the beginning, and the text token is right after a separator token sequence, an error returns as it's not trivial to find where the separator sequence ends and where the free text starts. In this case, use FX for strict matching. For example: CAST("1985-11- ' 10" AS DATE FORMAT "YYYY-MM-\" ' \"DD") returns an error, but removing the dash before the text token makes this succeed.

Examples:

Input	Output
`CAST("02-11-2014" AS DATE FORMAT 'dd-mm-yyyy')`	2014-11-02
`CAST("365 2014" AS DATE FORMAT 'DDD-YYYY')`	2014-12-31
`CAST("5-01-26" AS DATE FORMAT 'YY-MM-DD')`	Executed at 2019-01-01 11:11:11: 2015-01-26
`CAST('2018-11-10T15:11:04Z' AS TIMESTAMP FORMAT 'YYYY-MM-DDTHH24:MI:SSZ')`	2018-11-10 15:11:04
`CAST("95-01-28" AS DATE FORMAT 'YYY-MM-DD')`	Executed at 2019-01-01 11:11:11: 2095-01-28
`CAST("49-01-15" AS DATE FORMAT 'RR-MM-DD')`	Round year when last 2 digits of current year is greater than 49. Executed at 2099-01-01 11:11:11: 2149-01-15
`CAST('2019.10.10 13:30:40.123456 +01:30' AS TIMESTAMP FORMAT 'YYYY-MM-DD HH24:MI:SS.FF9 TZH:TZM')`	2019-10-10 13:30:40.123456000
`CAST('2001-03-01 03:10:15.123456 -01:30' AS TIMESTAMP FORMAT 'FXYYYY-MM-DD HH12:MI:SS.FF6 TZH:TZM')`	2001-03-01 03:10:15.123456000
`CAST('2001-03-02 03:10:15' AS TIMESTAMP FORMAT 'FXYYYY MM-DD HH12:MI:SS')`	NULL because the separator between the year and month do not match.
`CAST('2001-3-05' AS TIMESTAMP FORMAT 'FXYYYY-MM-DD')`	NULL because the length of the month token does not match the *PATTERN*.
`CAST('2001-3-11 3:15:00.12345' AS TIMESTAMP FORMAT 'FXYYYY-FMMM-DD FMHH12:MI:SS.FMFF')`	2001-03-11 03:15:00.123450000 The multiple FM modifiers for the month and hour override the FX and suppress padding.
`CAST('2019-01-01 01:01:01' AS TIMESTAMP`	1:01:1

Allow implicit casts between numeric and string types when inserting into table

The current implementation requires explicit casts for numeric and string-based literals, however this release relaxes the implicit casting rules for these cases. This is controlled through a query option allow_unsafe_casts and is turned off by default. This query option allows implicit casting between some numeric types and string types.

Possibilities:

From	To
String	Float
String	Double, Real
String	Tinyint
String	Smallint
String	Int
String	Bigint
Float	String
Double, Real	String
Tinyint	String
Smallint	String
Int	String
Bigint	String

Example

In the following example, numeric and string literals are inserted into a FLOAT column. The assumption is that all available numeric types and string types should be interchangeable when they are listed as literals in insert statements.

- String to Float column:
  - INSERT INTO example(float_col) VALUES ("041241");
- Float to String column:
  - INSERT INTO example(string_col) VALUES (CAST(123.0 AS FLOAT);
- Integer to String column:
  - INSERT INTO example(string_col) VALUES (10000000);
- String and Integer literals to String column:
  - INSERT INTO example(STRING_COL) VALUES (123), ("example");

Casting rules

A hand-written query containing different types qualifies for implicit casting, but a referenced column does not. All available numeric types and string types are interchangeable when they are listed as literals in insert statements.

The insert statements and set operations accept unsafe implicitly casted expressions when the source expressions are constant.

The following implicit type casts are enabled in unsafe mode:

- String -> Float, Double
- String -> Tinyint, Smallint, Int, Bigint
- Float, Double -> String
- Tinyint, Smallint, Int, Bigint -> String

TYPEOF(expression)

Purpose: Returns the name of the data type corresponding to *EXPRESSION*. For types with extra attributes, such as length for CHAR and VARCHAR, or precision and scale for DECIMAL, includes the full specification of the type.

Return type: STRING

Usage notes: Typically used in interactive exploration of a schema, or in application code that programmatically generates schema definitions such as CREATE TABLE statements, for example, to get the type of an expression such as col1 / col2 or CONCAT(col1, col2, col3). This function is especially useful for arithmetic expressions involving DECIMAL types because the precision and scale of the result is can be different than that of the operands.

Examples:

TYPEOF(2) returns TINYINT.

TYPEOF(NOW()) returns TIMESTAMP.

Related Information

TIMESTAMP data type

Impala date and time functions

The underlying Impala data type for date and time data is TIMESTAMP and DATE.

Some of the datetime functions are affected by the setting of the ##use_local_tz_for_unix_timestamp_conversions startup flag for the `impalad` daemon.

- This setting is off by default, meaning that functions such as FROM_UNIXTIME() and UNIX_TIMESTAMP() consider the input values to always represent the UTC time zone.
- This setting also applies when you CAST() a BIGINT value to TIMESTAMP, or a TIMESTAMP value to BIGINT. When this setting is enabled, these functions and operations convert to and from values representing the local time zone. See TIMESTAMP data type on page 46 for details about how Impala handles time zone considerations for the TIMESTAMP data type.

Function reference:

Impala supports the following data and time functions:

- ADD_MONTHS
- ADDDATE
- CURRENT_TIMESTAMP
- DATE_ADD
- DATE_PART
- DATE_SUB
- DATE_TRUNC
- DATEDIFF
- DAY
- DAYNAME
- DAYOFWEEK
- DAYOFYEAR
- DAYS_ADD
- DAYS_SUB
- EXTRACT
- FROM_TIMESTAMP
- FROM_UNIXTIME
- FROM_UTC_TIMESTAMP

- HOUR
- HOURS_ADD
- HOURS_SUB
- INT_MONTHS_BETWEEN
- MICROSECONDS_ADD
- MICROSECONDS_SUB
- MILLISECOND
- MILLISECONDS_ADD
- MILLISECONDS_SUB
- MINUTE
- MINUTES_ADD
- MINUTES_SUB
- MONTH
- MONTHNAME
- MONTHS_ADD
- MONTHS_BETWEEN
- MONTHS_SUB
- NANOSECONDS_ADD
- NANOSECONDS_SUB
- NEXT_DAY
- NOW
- QUARTER
- SECOND
- SECONDS_ADD
- SECONDS_SUB
- SUBDATE
- TIMEOFDAY
- TIMESTAMP_CMP
- TO_DATE
- TO_TIMESTAMP
- TO_UTC_TIMESTAMP
- TRUNC
- UNIX_TIMESTAMP
- UTC_TIMESTAMP
- WEEKOFYEAR
- WEEKS_ADD
- WEEKS_SUB
- YEAR
- YEARS_ADD
- YEARS_SUB

ADD_MONTHS(TIMESTAMP / DATE date, INT months), ADD_MONTHS(TIMESTAMP / DATE date, BIGINT months)

Purpose: Adds *MONTHS* to *DATE* and returns the new date value.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

Usage notes:

Same as MONTHS_ADD(). Available in Impala 1.4 and higher. For compatibility when porting code with vendor extensions.

ADDDATE(TIMESTAMP / DATE date, INT / BIGINT days)

Purpose: Adds *DAYS* to *DATE* and returns the new date value.

The *DAYS* value can be negative, which gives the same result as the SUBDATE() function.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

CURRENT_DATE()

Purpose: Returns the current date.

Any references to the CURRENT_DATE() function are evaluated at the start of a query. All calls to CURRENT_DATE() within the same query return the same value, and the value does not depend on how long the query takes.

Return type: DATE

CURRENT_TIMESTAMP()

Purpose: Alias for the NOW() function.

Return type: TIMESTAMP

DATE_ADD(TIMESTAMP / DATE date, INT / BIGINT days), DATE_ADD(TIMESTAMP / DATE date, interval_expression)

Purpose: Adds a specified number of days to the *DATE* argument.

With an INTERVAL expression as the second argument, you can calculate a delta value using other units such as weeks, years, hours, seconds, and so on.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

The following examples show the shorthand notation of an INTERVAL expression, instead of specifying the precise number of days. The INTERVAL notation also lets you work with units smaller than a single day.

```
select now() as right_now, date_add(now(), interval 3 weeks) as
 in_3_weeks;
+-------------------------------+-------------------------------+
| right_now                     | in_3_weeks                    |
+-------------------------------+-------------------------------+
| 2016-05-20 11:05:39.173331000 | 2016-06-10 11:05:39.173331000 |
+-------------------------------+-------------------------------+

select now() as right_now, date_add(now(), interval 6 hours) as
in_6_hours;
+-------------------------------+-------------------------------+
| right_now                     | in_6_hours                    |
+-------------------------------+-------------------------------+
| 2016-05-20 11:13:51.492536000 | 2016-05-20 17:13:51.492536000 |
+-------------------------------+-------------------------------+
```

Like all date/time functions that deal with months, date_add() handles nonexistent dates past the end of a month by setting the date to the last day of the month. The following example shows how the nonexistent date April 31st is normalized to April 30th:

```
select date_add(cast('2016-01-31' as timestamp), interval 3 mon
ths) as 'april_31st';
+---------------------+
| april_31st          |
+---------------------+
| 2016-04-30 00:00:00 |
+---------------------+
```

DATE_CMP(DATE date1, DATE date2)

Purpose: Compares *DATE1* and *DATE2* and returns:

- 0 if the dates are identical.
- 1 if *DATE1* > *DATE2*.
- -1 if *DATE1* < *DATE2*.
- NULL if *DATE1* or *DATE2* is NULL.

Return type: INT

DATE_PART(STRING part, TIMESTAMP / DATE date)

Purpose: Similar to EXTRACT(), with the argument order reversed. Supports the same date and time units as EXTRACT(). For compatibility with SQL code containing vendor extensions.

Return type: BIGINT

DATE_SUB(TIMESTAMP startdate, INT days), DATE_SUB(TIMESTAMP startdate, interval_expression)

Purpose: Subtracts a specified number of days from a TIMESTAMP value. With an INTERVAL expression as the second argument, you can calculate a delta value using other units such as weeks, years, hours, seconds, and so on.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

Examples:

The following example shows the simplest usage, of subtracting a specified number of days from a TIMESTAMP value:

```
select now() as right_now, date_sub(now(), 7) as last_week;
+-------------------------------+-------------------------------+
| right_now                     | last_week                     |
+-------------------------------+-------------------------------+
| 2016-05-20 11:21:30.491011000 | 2016-05-13 11:21:30.491011000 |
+-------------------------------+-------------------------------+
```

The following examples show the shorthand notation of an INTERVAL expression, instead of specifying the precise number of days. The INTERVAL notation also lets you work with units smaller than a single day.

```
select now() as right_now, date_sub(now(), interval 3 weeks) as
 3_weeks_ago;
+-------------------------------+-------------------------------+
| right_now                     | 3_weeks_ago                   |
```

```
+-------------------------------+-------------------------------+
| 2016-05-20 11:23:05.176953000 | 2016-04-29 11:23:05.176953000 |
+-------------------------------+-------------------------------+
select now() as right_now, date_sub(now(), interval 6 hours) as
 6_hours_ago;
+-------------------------------+-------------------------------+
| right_now                     | 6_hours_ago                   |
+-------------------------------+-------------------------------+
| 2016-05-20 11:23:35.439631000 | 2016-05-20 05:23:35.439631000 |
+-------------------------------+-------------------------------+
```

Like all date/time functions that deal with months, date_add() handles nonexistent dates past the end of a month by setting the date to the last day of the month. The following example shows how the nonexistent date April 31st is normalized to April 30th:

```
select date_sub(cast('2016-05-31' as timestamp), interval 1 mon
ths) as 'april_31st';
+---------------------+
| april_31st          |
+---------------------+
| 2016-04-30 00:00:00 |
+---------------------+
```

DATE_TRUNC(STRING unit, TIMESTAMP / DATE ts)

Purpose: Returns the *TS* value truncated to the specified *UNIT*.

Argument: The unit argument is not case-sensitive. This argument string can be one of:

Unit	Supported for TIMESTAMP	Supported for DATE
'MICROSECONDS'	Yes	No
'MILLISECONDS'	Yes	No
'SECOND'	Yes	No
'MINUTE'	Yes	No
'HOUR'	Yes	No
'DAY'	Yes	Yes
'WEEK'	Yes	Yes
'MONTH'	Yes	Yes
'YEAR'	Yes	Yes
'DECADE'	Yes	Yes
'CENTURY'	Yes	Yes
'MILLENNIUM'	Yes	Yes

Usage notes:

Although this function is similar to calling TRUNC() with a TIMESTAMP or DATE argument, the order of arguments and the recognized units are different between TRUNC() and DATE_TRUNC(). Therefore, these functions are not interchangeable.

This function is typically used in GROUP BY queries to aggregate results from the same hour, day, week, month, quarter, and so on. You can also use this function in an INSERT ... SELECT into a partitioned table to divide TIMESTAMP or DATE values into the correct partition.

Return type:

- TIMESTAMP if the second argument, *TS*, is TIMESTAMP.

- DATE if the second argument, *TS*, is DATE.

Examples:

DATE_TRUNC('HOUR', NOW()) returns 2017-12-05 13:00:00.

DATE_TRUNC('MILLENNIUM', DATE'2019-08-02') returns 2001-01-01.

DATEDIFF(TIMESTAMP / DATE enddate, TIMESTAMP / DATE startdate)

Purpose: Returns the number of days from *STARTDATE* to *ENDDATE*.

If *ENDDATE* > *STARTDATE*, the return value is positive.

If *ENDDATE* < *STARTDATE*, the return value is negative.

If *ENDDATE* = *STARTDATE*, the return value is zero.

Return type: INT

Usage notes:

The time portions of the *ENDDATE* and *STARTDATE* values are ignored. For example, 11:59 PM on one day and 12:01 AM on the next day represent a DATEDIFF() of -1 because the date/time values represent different days, even though the TIMESTAMP values differ by only 2 minutes.

DAY(TIMESTAMP / DATE date), DAYOFMONTH(TIMESTAMP / DATE date)

Purpose: Returns the day value from the *DATE* argument. The value represents the day of the month, therefore is in the range 1-31, or less for months without 31 days.

Returns NULL for nonexistent dates, e.g. Feb 30, or misformatted date strings, e.g. '1999-02-013'.

Return type: INT

DAYNAME(TIMESTAMP / DATE date)

Purpose: Returns the day name of the *DATE* argument. The range of return values is 'Sunday' to 'Sat urday'. Used in report-generating queries, as an alternative to calling DAYOFWEEK() and turning that numeric return value into a string using a CASE expression.

Return type: STRING

DAYOFWEEK(TIMESTAMP / DATE date)

Purpose: Returns the day field of the *DATE* arguement, corresponding to the day of the week. The range of return values is 1 (Sunday) to 7 (Saturday).

Return type: INT

DAYOFYEAR(TIMESTAMP / DATE date)

Purpose: Returns the day field from the *DATE* argument, corresponding to the day of the year. The range of return values is 1 (January 1) to 366 (December 31 of a leap year).

Return type: INT

DAYS_ADD(TIMESTAMP / DATE date, INT / BIGINT days)

Purpose: Returns the value with the number of *DAYS* added to *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

DAYS_SUB(TIMESTAMP / DATE date, INT / BIGINT days)

Purpose: Returns the value with the number of *DAYS* subtracted from *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

EXTRACT(TIMESTAMP / DATE ts, STRING unit), EXTRACT(unit FROM TIMESTAMP / DATE ts)

Purpose: Returns one of the numeric date or time fields, specified by *UNIT*, from *TS*.

Argument: The unit argument value is not case-sensitive. The unit string can be one of:

Unit	Supported for TIMESTAMP ts	Supported for DATE ts
'EPOCH'	Yes	No
'MILLISECOND'	Yes	No
'SECOND'	Yes	No
'MINUTE'	Yes	No
'HOUR'	Yes	No
'DAY'	Yes	Yes
'MONTH'	Yes	Yes
'QUARTER'	Yes	Yes
'YEAR'	Yes	Yes

Usage notes:

Typically used in GROUP BY queries to arrange results by hour, day, month, and so on. You can also use this function in an INSERT ... SELECT statement to insert into a partitioned table to split up TIMESTAMP values into individual parts, if the partitioned table has separate partition key columns representing year, month, day, and so on. If you need to divide by more complex units of time, such as by week or by quarter, use the TRUNC() function instead.

Return type: BIGINT

Examples:

EXTRACT(DAY FROM DATE'2019-08-17') returns 17.

If you specify 'MILLISECOND' for the *UNIT* argument, the function returns the seconds component and the milliseconds component.

EXTRACT(CAST('2006-05-12 18:27:28.123456789' AS TIMESTAMP), 'MILLISECOND') returns 28123.

FROM_TIMESTAMP(TIMESTAMP datetime, STRING pattern), FROM_TIMESTAMP(STRING datetime, STRING pattern)

Purpose: Converts a TIMESTAMP value into a string representing the same value.

Return type: STRING

Usage notes:

The FROM_TIMESTAMP() function provides a flexible way to convert TIMESTAMP values into arbitrary string formats for reporting purposes.

Because Impala implicitly converts string values into TIMESTAMP, you can pass date/time values represented as strings (in the standard yyyy-MM-dd HH:mm:ss.SSS format) to this function. The result is a string using different separator characters, order of fields, spelled-out month names, or other variation of the date/time string representation.

The allowed tokens for the pattern string are the same as for the FROM_UNIXTIME() function.

FROM_UNIXTIME(BIGINT unixtime[, STRING pattern])

Purpose: Converts the number of seconds from the Unix epoch to the specified time into a string in the local time zone.

Return type: STRING

The *PATTERN* string supports the following subset of Java SimpleDateFormat.

Pattern	Description
y	Year
M	Month
d	Day
H	Hour
m	Minute
s	Second
S	Fractional second
+/-hh:mm	Time zone offset
+/-hhmm	Time zone offset
+/-hh	Time zone offset

The following rules apply to the *PATTERN* string:

- The *PATTERN* string is case-sensitive.
- All fields are variable length, and thus must use separators to specify the boundaries of the fields, with the exception of the time zone values.
- Time zone offset formats must be at the end of the *PATTERN* string.
- Formatting character groups can appear in any order along with any separators except for the time zone offset. For example:
  - yyyy/MM/dd
  - dd-MMM-yy
  - (dd)(MM)(yyyy) HH:mm:ss
  - yyyy-MM-dd HH:mm:ss+hh:mm

In Impala 1.3 and later, you can switch the order of elements, use alternative separator characters, and use a different number of placeholders for each unit. Adding more instances of y, d, H, and so on produces output strings zero-padded to the requested number of characters. The exception is M for months, where M produces a non-padded value such as 3, MM produces a zero-padded value such as 03, MMM produces an abbreviated month name such as Mar, and sequences of 4 or more M are not allowed.

A date string including all fields could be 'yyyy-MM-dd HH:mm:ss.SSSSSS', 'dd/MM/yyyy HH:mm:ss.SSSSSS', 'MMM dd, yyyy HH.mm.ss (SSSSSS)' or other combinations of placeholders and separator characters.

Usage notes:

The way this function deals with time zones when converting to or from TIMESTAMP values is affected by the ##use_local_tz_for_unix_timestamp_conversions startup flag for the `impalad` daemon.

FROM_UTC_TIMESTAMP(TIMESTAMP ts, STRING timezone)

Purpose: Converts a specified UTC timestamp value into the appropriate value for a specified time zone.

Return type: TIMESTAMP

Usage notes: Often used to translate UTC time zone data stored in a table back to the local date and time for reporting. The opposite of the TO_UTC_TIMESTAMP() function.

To determine the time zone of the server you are connected to, you can call the TIMEOFDAY() function, which includes the time zone specifier in its return value. Remember that with cloud computing, the server you interact with might be in a different time zone than you are, or different

sessions might connect to servers in different time zones, or a cluster might include servers in more than one time zone.

See discussion of time zones in TIMESTAMP data type on page 46 for information about using this function for conversions between the local time zone and UTC.

HOUR(TIMESTAMP ts)

Purpose: Returns the hour field from a TIMESTAMP field.

Return type: INT

HOURS_ADD(TIMESTAMP date, INT hours), HOURS_ADD(TIMESTAMP date, BIGINT hours)

Purpose: Returns the specified date and time plus some number of hours.

Return type: TIMESTAMP

HOURS_SUB(TIMESTAMP date, INT hours), HOURS_SUB(TIMESTAMP date, BIGINT hours)

Purpose: Returns the specified date and time minus some number of hours.

Return type: TIMESTAMP

INT_MONTHS_BETWEEN(TIMESTAMP / DATE enddate, TIMESTAMP / DATE startdate)

Purpose: Returns the number of months from *STARTDATE* to *ENDDATE*, representing only the full months that passed.

Return type: INT

Usage notes:

Typically used in business contexts, for example to determine whether a specified number of months have passed or whether some end-of-month deadline was reached.

The method of determining the number of elapsed months includes some special handling of months with different numbers of days that creates edge cases for dates between the 28th and 31st days of certain months. See MONTHS_BETWEEN() for details. The INT_MONTHS_BETWEEN() result is essentially the FLOOR() of the MONTHS_BETWEEN() result.

If either value is NULL, which could happen for example when converting a nonexistent date string such as '2015-02-29' to a TIMESTAMP, the result is also NULL.

If the first argument represents an earlier time than the second argument, the result is negative.

LAST_DAY(TIMESTAMP / DATE ts)

Purpose: Returns the beginning of the last calendar day in the same month of *TS*.

Return type:

- Returns TIMESTAMP if *TS* is of the TIMESTAMP type.
- Returns DATE if *TS* is of the DATE type.

Usage notes:

If the input argument does not represent a valid Impala TIMESTAMP including both date and time portions, the function returns NULL. For example, if the input argument is a string that cannot be implicitly cast to TIMESTAMP, does not include a date portion, or is out of the allowed range for Impala TIMESTAMP values, the function returns NULL.

MICROSECONDS_ADD(TIMESTAMP date, INT microseconds), MICROSECONDS_ADD(TIMESTAMP date, BIGINT microseconds)

Purpose: Returns the specified date and time plus some number of microseconds.

Return type: TIMESTAMP

MICROSECONDS_SUB(TIMESTAMP date, INT microseconds), MICROSECONDS_SUB(TIMESTAMP date, BIGINT microseconds)

Purpose: Returns the specified date and time minus some number of microseconds.

Return type: TIMESTAMP

MILLISECOND(TIMESTAMP ts)

Purpose: Returns the millisecond portion of a TIMESTAMP value.

Return type: INT

Usage notes:

The millisecond value is truncated, not rounded, if the TIMESTAMP value contains more than 3 significant digits to the right of the decimal point.

MILLISECONDS_ADD(TIMESTAMP date, INT milliseconds), MILLISECONDS_ADD(TIMESTAMP date, BIGINT milliseconds)

Purpose: Returns the specified date and time plus some number of milliseconds.

Return type: TIMESTAMP

MILLISECONDS_SUB(TIMESTAMP date, INT milliseconds), MILLISECONDS_SUB(TIMESTAMP date, BIGINT milliseconds)

Purpose: Returns the specified date and time minus some number of milliseconds.

Return type: TIMESTAMP

MINUTE(TIMESTAMP date)

Purpose: Returns the minute field from a TIMESTAMP value.

Return type: INT

MINUTES_ADD(TIMESTAMP date, INT minutes), MINUTES_ADD(TIMESTAMP date, BIGINT minutes)

Purpose: Returns the specified date and time plus some number of minutes.

Return type: TIMESTAMP

MINUTES_SUB(TIMESTAMP date, INT minutes), MINUTES_SUB(TIMESTAMP date, BIGINT minutes)

Purpose: Returns the specified date and time minus some number of minutes.

Return type: TIMESTAMP

MONTH(TIMESTAMP / DATE date)

Purpose: Returns the month field, represented as an integer, from the *DATE* argument.

Return type: INT

MONTHNAME(TIMESTAMP / DATE date)

Purpose: Returns the month name of the *DATE* argument.

Return type: STRING

MONTHS_ADD(TIMESTAMP / DATE date, INT / BIGINT months)

Purpose: Returns the value with the number of *MONTHS* added to *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

Usage notes:

If *DATE* is the last day of a month, the return date will fall on the last day of the target month, e.g. MONTHS_ADD(DATE'2019-01-31', 1) returns DATE'2019-02-28'.

MONTHS_BETWEEN(TIMESTAMP / DATE enddate, TIMESTAMP / DATE startdate)

Purpose: Returns the number of months from *STARTDATE* to *ENDDATE*.

This result can include a fractional part representing extra days in addition to the full months between the dates. The fractional component is computed by dividing the difference in days by 31 (regardless of the month).

Return type: DOUBLE

Usage notes:

Typically used in business contexts, for example to determine whether a specified number of months have passed or whether some end-of-month deadline was reached.

If the only consideration is the number of full months and any fractional value is not significant, use INT_MONTHS_BETWEEN() instead.

The method of determining the number of elapsed months includes some special handling of months with different numbers of days that creates edge cases for dates between the 28th and 31st days of certain months.

If either value is NULL, which could happen for example when converting a nonexistent date string such as '2015-02-29' to a TIMESTAMP, the result is also NULL.

If the first argument represents an earlier time than the second argument, the result is negative.

The time portion of the input arguements are ignored.

MONTHS_SUB(TIMESTAMP / DATE date, INT / BIGINT months)

Purpose: Returns the value with the number of *MONTHS* subtracted from *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

Usage notes:

If *DATE* is the last day of a month, the return date will fall on the last day of the target month, e.g. MONTHS_SUB(DATE'2019-02-28', 1) returns DATE'2019-01-31'.

NANOSECONDS_ADD(TIMESTAMP date, INT nanoseconds), NANOSECONDS_ADD(TIMESTAMP date, BIGINT nanoseconds)

Purpose: Returns the specified date and time plus some number of nanoseconds.

Return type: TIMESTAMP

Kudu considerations:

The nanosecond portion of an Impala TIMESTAMP value is rounded to the nearest microsecond when that value is stored in a Kudu table.

NANOSECONDS_SUB(TIMESTAMP date, INT nanoseconds), NANOSECONDS_SUB(TIMESTAMP date, BIGINT nanoseconds)

Purpose: Returns the specified date and time minus some number of nanoseconds.

Return type: TIMESTAMP

Kudu considerations:

The nanosecond portion of an Impala TIMESTAMP value is rounded to the nearest microsecond when that value is stored in a Kudu table.

NEXT_DAY(TIMESTAMP / DATE date, STRING weekday)

Purpose: Returns the date of the *WEEKDAY* that follows the specified *DATE*.

Argument: The *WEEKDAY* is not case-sensitive.

The following values are accepted for *WEEKDAY*: "Sunday"/"Sun", "Monday"/"Mon", "Tue sday"/"Tue", "Wednesday"/"Wed", "Thursday"/"Thu", "Friday"/"Fri", "Saturday"/"Sat"

Return type:

- Returns TIMESTAMP if *DATE* is of the TIMESTAMP type.
- Returns DATE if *DATE* is of the DATE type.

Examples:

NEXT_DAY('2013-12-25','Saturday') returns '2013-12-28 00:00:00' which is the first Saturday after December 25, 2013.

NOW()

Purpose: Returns the current date and time (in the local time zone) as a TIMESTAMP value.

Return type: TIMESTAMP

Usage notes:

To find a date/time value in the future or the past relative to the current date and time, add or subtract an INTERVAL expression to the return value of NOW().

To produce a TIMESTAMP representing the current date and time that can be shared or stored without interoperability problems due to time zone differences, use the TO_UTC_TIMESTAMP() function and specify the time zone of the server. When TIMESTAMP data is stored in UTC form, any application that queries those values can convert them to the appropriate local time zone by calling the inverse function, FROM_UTC_TIMESTAMP().

To determine the time zone of the server you are connected to, you can call the timeofday() function, which includes the time zone specifier in its return value. Remember that with cloud computing, the server you interact with might be in a different time zone than you are, or different sessions might connect to servers in different time zones, or a cluster might include servers in more than one time zone.

Any references to the NOW() function are evaluated at the start of a query. All calls to NOW() within the same query return the same value, and the value does not depend on how long the query takes.

QUARTER(TIMESTAMP / DATE date)

Purpose: Returns the quarter in the input *DATE* argument as an integer value, 1, 2, 3, or 4, where 1 represents January 1 through March 31.

Return type: INT

SECOND(TIMESTAMP date)

Purpose: Returns the second field from a TIMESTAMP value.

Return type: INT

SECONDS_ADD(TIMESTAMP date, INT seconds), SECONDS_ADD(TIMESTAMP date, BIGINT seconds)

Purpose: Returns the specified date and time plus some number of seconds.

Return type: TIMESTAMP

SECONDS_SUB(TIMESTAMP date, INT seconds), SECONDS_SUB(TIMESTAMP date, BIGINT seconds)

Purpose: Returns the specified date and time minus some number of seconds.

Return type: TIMESTAMP

SUBDATE(TIMESTAMP / DATE date, INT / BIGINT days)

Purpose: Subtracts *DAYS* from *DATE* and returns the new date value.

The *DAYS* value can be negative, which gives the same result as the ADDDATE() function.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

TIMEOFDAY()

Purpose: Returns a string representation of the current date and time, according to the time of the local system, including any time zone designation.

Return type: STRING

Usage notes:

The result value represents similar information as the now() function, only as a STRING type and with somewhat different formatting. For example, the day of the week and the time zone identifier are included. This function is intended primarily for compatibility with SQL code from other systems that also have a timeofday() function. Prefer to use now() if practical for any new Impala code.

TIMESTAMP_CMP(TIMESTAMP t1, TIMESTAMP t2)

Purpose: Tests if one TIMESTAMP value is newer than, older than, or identical to another TIME STAMP

- If the first argument represents a later point in time than the second argument, the result is 1.
- If the first argument represents an earlier point in time than the second argument, the result is -1.
- If the first and second arguments represent identical points in time, the result is 0.
- If either argument is NULL, the result is NULL.

Return type: INT (either -1, 0, 1, or NULL)

Usage notes:

A comparison function for TIMESTAMP values that only tests whether the date and time increases, decreases, or stays the same. Similar to the SIGN() function for numeric values.

TO_DATE(TIMESTAMP ts)

Purpose: Returns a string representation of the date field from the *TS* argument.

Return type: STRING

TO_TIMESTAMP(BIGINT unixtime), TO_TIMESTAMP(STRING date, STRING pattern)

Purpose: Converts an integer or string representing a date/time value into the corresponding TIME STAMP value.

Return type: TIMESTAMP

Usage notes:

An integer argument represents the number of seconds past the epoch (midnight on January 1, 1970). It is the converse of the UNIX_TIMESTAMP() function, which produces a BIGINT representing the number of seconds past the epoch.

A string argument, plus another string argument representing the pattern, turns an arbitrary string representation of a date and time into a true TIMESTAMP value. The ability to parse many kinds of date and time formats allows you to deal with temporal data from diverse sources, and if desired to convert to efficient TIMESTAMP values during your ETL process. Using TIMESTAMP directly in queries and expressions lets you perform date and time calculations without the overhead of extra function calls and conversions each time you reference the applicable columns.

Examples:

The following examples demonstrate how to convert an arbitrary string representation to TIME STAMP based on a pattern string:

```
select to_timestamp('Sep 25, 1984', 'MMM dd, yyyy');
+-----------------------------------------------+
| to_timestamp('sep 25, 1984', 'mmm dd, yyyy') |
+-----------------------------------------------+
| 1984-09-25 00:00:00                           |
+-----------------------------------------------+

select to_timestamp('1984/09/25', 'yyyy/MM/dd');
+-----------------------------------------+
| to_timestamp('1984/09/25', 'yyyy/mm/dd') |
+-----------------------------------------+
| 1984-09-25 00:00:00                     |
+-----------------------------------------+
```

The following examples show how to convert a BIGINT representing seconds past epoch into a TIMESTAMP value:

```
-- One day past the epoch.
select to_timestamp(24 * 60 * 60);
+-----------------------------+
| to_timestamp(24 * 60 * 60) |
+-----------------------------+
| 1970-01-02 00:00:00        |
+-----------------------------+

-- 60 seconds in the past.
select now() as 'current date/time',
  unix_timestamp(now()) 'now in seconds',
  to_timestamp(unix_timestamp(now()) - 60) as '60 seconds ago';
+-------------------------------+----------------+-------------
--------+
| current date/time             | now in seconds | 60 seconds
ago      |
+-------------------------------+----------------+---------------
------+
| 2017-10-01 22:03:46.885624000 | 1506895426     | 2017-10-01 22
:02:46 |
+-------------------------------+----------------+-------------
--------+
```

TO_UTC_TIMESTAMP(TIMESTAMP ts, STRING timezone)

Purpose: Converts a specified timestamp value in a specified time zone into the corresponding value for the UTC time zone.

Return type: TIMESTAMP

Usage notes:

Often used in combination with the now() function, to translate local date and time values to the UTC time zone for consistent representation on disk. The opposite of the FROM_UTC_TIMESTA MP() function.

See discussion of time zones in TIMESTAMP data type on page 46 for information about using this function for conversions between the local time zone and UTC.

Examples:

The simplest use of this function is to turn a local date/time value to one with the standardized UTC time zone. Because the time zone specifier is not saved as part of the Impala TIMESTAMP value, all applications that refer to such data must agree in advance which time zone the values represent. If different parts of the ETL cycle, or different instances of the application, occur in different time zones, the ideal reference point is to convert all TIMESTAMP values to UTC for storage.

```
select now() as 'Current time in California USA',
  to_utc_timestamp(now(), 'PDT') as 'Current time in Greenwich UK
';
+--------------------------------+-------------------------------
+
| current time in california usa | current time in greenwich uk
 |
+--------------------------------+-------------------------------
+
| 2016-06-01 15:52:08.980072000  | 2016-06-01 22:52:08.980072000
 |
+--------------------------------+-------------------------------
+
```

Once a value is converted to the UTC time zone by TO_UTC_TIMESTAMP(), it can be converted back to the local time zone with FROM_UTC_TIMESTAMP(). You can combine these functions using different time zone identifiers to convert a TIMESTAMP between any two time zones. This example starts with a TIMESTAMP value representing Pacific Daylight Time, converts it to UTC, and converts it to the equivalent value in Eastern Daylight Time.

```
select now() as 'Current time in California USA',
  from_utc_timestamp
  (
    to_utc_timestamp(now(), 'PDT'),
    'EDT'
  ) as 'Current time in New York, USA';
+--------------------------------+-------------------------------
+
| current time in california usa | current time in new york, usa
 |
+--------------------------------+-------------------------------
+
| 2016-06-01 18:14:12.743658000  | 2016-06-01 21:14:12.743658000
|
+--------------------------------+-------------------------------
+
```

TRUNC(TIMESTAMP / DATE ts, STRING unit)

Purpose: Returns the *TS* truncated to the *UNIT* specified.

Argument: The *UNIT* argument is not case-sensitive. This argument string can be one of:

Unit	Supported for TIMESTAMP ts	Supported for DATE ts	Description
'SYYYY' 'YYYY' 'YEAR' 'SYEAR' 'YYY' 'YY' 'Y'	Yes	Yes	Year
'Q'	Yes	Yes	Quarter
'MONTH' 'MON' 'MM' 'RM'	Yes	Yes	Month
'WW'	Yes	Yes	The most recent date that is the same day of the week as the first day of the year
'W'	Yes	Yes	The most recent date that is the same day of the week as the first day of the month
'DDD' 'DD' 'J'	Yes	Yes	Day
'DAY' 'DY' 'D'	Yes	Yes	Starting day of the week (Monday)
'HH' 'HH12' 'HH24'	Yes	No	Hour. A TIMESTAMP value truncated to the hour is always represented in 24-hour notation, even for the HH12 argument string.
'MI'	Yes	No	Minute

Usage notes:

The TIMESTAMP form is typically used in GROUP BY queries to aggregate results from the same hour, day, week, month, quarter, and so on. You can also use this function in an INSERT ... S ELECT into a partitioned table to divide TIMESTAMP values into the correct partition.

Return type:

- TIMESTAMP if the first argument, *TS*, is TIMESTAMP.
- DATE if the first argument, *TS*, is DATE.

Example:

TRUNC(DATE'2019-05-08','YEAR') returns 2019-01-01.

TRUNC(DATE'2019-05-08', 'QUARTER') returns 2019-04-01.

UNIX_TIMESTAMP(), UNIX_TIMESTAMP(STRING datetime), UNIX_TIMESTAMP(STRING datetime, STRING pattern), UNIX_TIMESTAMP(TIMESTAMP datetime)

Purpose: Returns a Unix time, which is a number of seconds elapsed since '1970-01-01 00:00:00' UTC. If called with no argument, the current date and time is converted to its Unix time. If called with arguments, the first argument represented as the TIMESTAMP or STRING is converted to its Unix time.

Return type: BIGINT

Usage notes:

See FROM_UNIXTIME() for details about the patterns you can use in the *PATTERN* string to represent the position of year, month, day, and so on in the date string. In Impala 1.3 and higher, you have more flexibility to switch the positions of elements and use different separator characters.

You can include a trailing uppercase Z qualifier to indicate “Zulu” time, a synonym for UTC.

You can include a timezone offset specified as minutes and hours, provided you also specify the details in the *PATTERN* string argument. The offset is specified in the *PATTERN* string as a plus or minus sign followed by hh:mm, hhmm, or hh. The hh must be lowercase, to distinguish it from the HH represent hours in the actual time value. Currently, only numeric timezone offsets are allowed, not symbolic names.

Built-in functions that accept or return integers representing TIMESTAMP values use the BIGINT type for parameters and return values, rather than INT. This change lets the date and time functions avoid an overflow error that would otherwise occur on January 19th, 2038 (known as the “Year 2038 problem” or “Y2K38 problem”). This change affects the FROM_UNIXTIME() and UNIX_TIMESTAMP() functions. You might need to change application code that interacts with these functions, change the types of columns that store the return values, or add CAST() calls to SQL statements that call these functions.

UNIX_TIMESTAMP() and FROM_UNIXTIME() are often used in combination to convert a TIMESTAMP value into a particular string format. For example, FROM_UNIXTIME(UNIX_TIMESTAMP(NOW() + interval 3 days), 'yyyy/MM/dd HH:mm').

The way this function deals with time zones when converting to or from TIMESTAMP values is affected by the ##use_local_tz_for_unix_timestamp_conversions startup flag for the `impalad` daemon. See TIMESTAMP data type on page 46 for details about how Impala handles time zone considerations for the TIMESTAMP data type.

Examples:

The following examples show different ways of turning the same date and time into an integer value. A *PATTERN* string that Impala recognizes by default is interpreted as a UTC date and time. The trailing Z is a confirmation that the timezone is UTC. If the date and time string is formatted differently, a second argument specifies the position and units for each of the date and time values.

The final two examples show how to specify a timezone offset of Pacific Daylight Saving Time, which is 7 hours earlier than UTC. You can use the numeric offset -07:00 and the equivalent suffix of -hh:mm in the *PATTERN* string, or specify the mnemonic name for the time zone in a call to TO_UTC_TIMESTAMP(). This particular date and time expressed in PDT translates to a different number than the same date and time expressed in UTC.

```
-- 3 ways of expressing the same date/time in UTC and converting
 to an integer.

select unix_timestamp('2015-05-15 12:00:00');
+----------------------------------------+
| unix_timestamp('2015-05-15 12:00:00') |
+----------------------------------------+
| 1431691200                             |
+----------------------------------------+

select unix_timestamp('2015-05-15 12:00:00Z');
```

```
+----------------------------------------+
| unix_timestamp('2015-05-15 12:00:00z') |
+----------------------------------------+
| 1431691200                             |
+----------------------------------------+

select unix_timestamp
(
  'May 15, 2015 12:00:00',
  'MMM dd, yyyy HH:mm:ss'
) as may_15_month_day_year;
+------------------------+
| may_15_month_day_year |
+------------------------+
| 1431691200            |
+------------------------+

-- 2 ways of expressing the same date and time but in a differ
ent timezone.
-- The resulting integer is different from the previous examples.
select unix_timestamp
(
  '2015-05-15 12:00:00-07:00',
  'yyyy-MM-dd HH:mm:ss-hh:mm'
) as may_15_year_month_day;
+------------------------+
| may_15_year_month_day |
+------------------------+
| 1431716400            |
+------------------------+
select unix_timestamp
  (to_utc_timestamp(
    '2015-05-15 12:00:00',
    'PDT')
  ) as may_15_pdt;
+------------+
| may_15_pdt |
+------------+
| 1431716400 |
+------------+
```

UTC_TIMESTAMP()

Purpose: Returns a TIMESTAMP corresponding to the current date and time in the UTC time zone.

Return type: TIMESTAMP

Examples:

Similar to the NOW() or CURRENT_TIMESTAMP() functions, but does not use the local time zone as those functions do. Use UTC_TIMESTAMP() to record TIMESTAMP values that are interoperable with servers around the world, in arbitrary time zones, without the need for additional conversion functions to standardize the time zone of each value representing a date/time.

For working with date/time values represented as integer values, you can convert back and forth between TIMESTAMP and BIGINT with the UNIX_MICROS_TO_UTC_TIMESTAMP() and UTC_TO_UNIX_MICROS() functions. The integer values represent the number of microseconds since the Unix epoch (midnight on January 1, 1970).

Examples:

The following example shows how NOW() and CURRENT_TIMESTAMP() represent the current date/time in the local time zone (in this case, UTC-7), while utc_timestamp() represents the same date/time in the standardized UTC time zone:

```
select now(), utc_timestamp();
+-------------------------------+-------------------------------+
| now()                         | utc_timestamp()               |
+-------------------------------+-------------------------------+
| 2017-10-01 23:33:58.919688000 | 2017-10-02 06:33:58.919688000 |
+-------------------------------+-------------------------------+

select current_timestamp(), utc_timestamp();
+-------------------------------+-------------------------------+
| current_timestamp()           | utc_timestamp()               |
+-------------------------------+-------------------------------+
| 2017-10-01 23:34:07.400642000 | 2017-10-02 06:34:07.400642000 |
+-------------------------------+-------------------------------+
```

WEEK(TIMESTAMP / DATE date), WEEKOFYEAR(TIMESTAMP / DATE date)

Purpose: Returns the corresponding week (1-53) from the *DATE* argument.

Return type: INT

WEEKS_ADD(TIMESTAMP / DATE date, INT / BIGINT weeks)

Purpose: Returns the value with the number of *WEEKS* added to *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

WEEKS_SUB(TIMESTAMP / DATE date, INT / BIGINT weeks)

Purpose: Returns the value with the number of *WEEKS* subtracted from *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

YEAR(TIMESTAMP / DATE date)

Purpose: Returns the year field from the *DATE* argument.

Return type: INT

YEARS_ADD(TIMESTAMP / DATE date, INT / BIGINT years)

Purpose: Returns the value with the number of *YEARS* added to *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.
- If *DATE* is DATE, returns DATE.

Usage notes:

If the equivalent date does not exist in the year of the result due to a leap year, the date is changed to the last day of the appropriate month.

YEARS_SUB(TIMESTAMP / DATE date, INT / BIGINT years)

Purpose: Returns the value with the number of *YEARS* subtracted from *DATE*.

Return type:

- If *DATE* is TIMESTAMP, returns TIMESTAMP.

- If *DATE* is DATE, returns DATE.

Usage notes:

If the equivalent date does not exist in the year of the result due to a leap year, the date is changed to the last day of the appropriate month.

Impala conditional functions

Impala supports the following conditional functions for testing equality, comparison operators, and nullity.

- CASE
- CASE2
- COALESCE
- DECODE
- IF
- IFNULL
- ISFALSE
- ISNOTFALSE
- ISNOTTRUE
- ISNULL
- ISTRUE
- NONNULLVALUE
- NULLIF
- NULLIFZERO
- NULLVALUE
- NVL
- NVL2
- ZEROIFNULL

CASE a WHEN b THEN c [WHEN d THEN e]... [ELSE f] END

Purpose: Compares an expression to one or more possible values, and returns a corresponding result when a match is found.

Return type: same as the initial argument value, except that integer values are promoted to BIGI NT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

Usage notes:

In this form of the CASE expression, the initial value A being evaluated for each row it typically a column reference, or an expression involving a column. This form can only compare against a set of specified values, not ranges, multi-value comparisons such as BETWEEN or IN, regular expressions, or NULL.

Examples:

Although this example is split across multiple lines, you can put any or all parts of a CASE expression on a single line, with no punctuation or other separators between the WHEN, ELSE, and END clauses.

```
select case x
    when 1 then 'one'
    when 2 then 'two'
    when 0 then 'zero'
    else 'out of range'
  end
```

```
    from t1;
```

CASE WHEN a THEN b [WHEN c THEN d]... [ELSE e] END

Purpose: Tests whether any of a sequence of expressions is TRUE, and returns a corresponding result for the first TRUE expression.

Return type: same as the initial argument value, except that integer values are promoted to BIGINT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

Usage notes:

CASE expressions without an initial test value have more flexibility. For example, they can test different columns in different WHEN clauses, or use comparison operators such as BETWEEN, IN and IS NULL rather than comparing against discrete values.

CASE expressions are often the foundation of long queries that summarize and format results for easy-to-read reports. For example, you might use a CASE function call to turn values from a numeric column into category strings corresponding to integer values, or labels such as “Small”, “Medium” and “Large” based on ranges. Then subsequent parts of the query might aggregate based on the transformed values, such as how many values are classified as small, medium, or large. You can also use CASE to signal problems with out-of-bounds values, NULL values, and so on.

By using operators such as OR, IN, REGEXP, and so on in CASE expressions, you can build extensive tests and transformations into a single query. Therefore, applications that construct SQL statements often rely heavily on CASE calls in the generated SQL code.

Because this flexible form of the CASE expressions allows you to perform many comparisons and call multiple functions when evaluating each row, be careful applying elaborate CASE expressions to queries that process large amounts of data. For example, when practical, evaluate and transform values through CASE after applying operations such as aggregations that reduce the size of the result set; transform numbers to strings after performing joins with the original numeric values.

Examples:

Although this example is split across multiple lines, you can put any or all parts of a CASE expression on a single line, with no punctuation or other separators between the WHEN, ELSE, and END clauses.

```
select case
    when dayname(now()) in ('Saturday','Sunday') then 'result un
defined on weekends'
    when x > y then 'x greater than y'
    when x = y then 'x and y are equal'
    when x is null or y is null then 'one of the columns is null'
    else null
  end
    from t1;
```

COALESCE(type v1, type v2, ...)

Purpose: Returns the first specified argument that is not NULL, or NULL if all arguments are NULL.

Return type: same as the initial argument value, except that integer values are promoted to BIGINT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

DECODE(type expression, type search1, type result1 [, type search2, type result2 ...] [, type default])

Purpose: Compares the first argument, expression, to the search expressions using the IS NOT DISTINCT operator, and returns:

- The corresponding result when a match is found.

- The first corresponding result if there are more than one matching search expressions.
- The default expression if none of the search expressions matches the first argument expression.
- NULL if the final default expression is omitted and none of the search expressions matches the first argument.

Return type: Same as the first argument with the following exceptions:

- Integer values are promoted to BIGINT.
- Floating-point values are promoted to DOUBLE.
- Use CAST() when inserting into a smaller numeric column.

Usage notes:

- Can be used as shorthand for a CASE expression.
- The first argument, expression, and the search expressions must be of the same type or convertible types.
- The result expression can be a different type, but all result expressions must be of the same type.
- Returns a successful match if the first argument is NULL and a search expression is also NULL.
- NULL can be used as a search expression.

Examples:

The following example translates numeric day values into weekday names, such as 1 to Monday, 2 to Tuesday, etc.

```
SELECT event, DECODE(day_of_week, 1, "Monday", 2, "Tuesday", 3,
"Wednesday",
  4, "Thursday", 5, "Friday", 6, "Saturday", 7, "Sunday", "Unkn
own day")
  FROM calendar;
```

IF(BOOLEAN condition, type ifTrue, type ifFalseOrNull)

Purpose: Tests an expression and returns a corresponding result depending on whether the result is TRUE, FALSE, or NULL.

Return type: Same as the ifTrue argument value

IFNULL(type a, type ifNull)

Purpose: Alias for the ISNULL() function, with the same behavior. To simplify porting SQL with vendor extensions to Impala.

Added in: Impala 1.3.0

ISFALSE(BOOLEAN expression)

Purpose: Returns TRUE if the expression is FALSE. Returns FALSE if the expression is TRUE or NULL.

Same as the IS FALSE operator.

Similar to ISNOTTRUE(), except it returns the opposite value for a NULL argument.

Return type: BOOLEAN

Usage notes:

In Impala 2.1.1 and higher, you can use the operators IS [NOT] TRUE and IS [NOT] FALSE as equivalents for the built-in functions ISTRUE(), ISNOTTRUE(), ISFALSE(), and ISNOTFALSE().

ISNOTFALSE(BOOLEAN expression)

Purpose: Tests if a Boolean expression is not FALSE (that is, either TRUE or NULL). Returns TRUE if so. If the argument is NULL, returns TRUE.

Same as the IS NOT FALSE operator.

Similar to ISTRUE(), except it returns the opposite value for a NULL argument.

Return type: BOOLEAN

Usage notes: Primarily for compatibility with code containing industry extensions to SQL.

Usage notes:

In Impala 2.1.1 and higher, you can use the operators IS [NOT] TRUE and IS [NOT] FALSE as equivalents for the built-in functions ISTRUE(), ISNOTTRUE(), ISFALSE(), and ISNOTFALSE().

ISNOTTRUE(BOOLEAN expression)

Purpose: Tests if a Boolean expression is not TRUE (that is, either FALSE or NULL). Returns TRUE if so. If the argument is NULL, returns TRUE.

Same as the IS NOT TRUE operator.

Similar to ISFALSE(), except it returns the opposite value for a NULL argument.

Return type: BOOLEAN

Usage notes:

In Impala 2.1.1 and higher, you can use the operators IS [NOT] TRUE and IS [NOT] FALSE as equivalents for the built-in functions ISTRUE(), ISNOTTRUE(), ISFALSE(), and ISNOTFALSE().

ISNULL(type a, type ifNull)

Purpose: Tests if an expression is NULL, and returns the expression result value if not. If the first argument is NULL, returns the second argument.

Compatibility notes: Equivalent to the NVL() function from Oracle Database or IFNULL() from MySQL. The NVL() and IFNULL() functions are also available in Impala.

Return type: Same as the first argument value

ISTRUE(BOOLEAN expression)

Purpose: Returns TRUE if the expression is TRUE. Returns FALSE if the expression is FALSE or NULL.

Same as the IS TRUE operator.

Similar to ISNOTFALSE(), except it returns the opposite value for a NULL argument.

Return type: BOOLEAN

Usage notes: Primarily for compatibility with code containing industry extensions to SQL.

Usage notes:

In Impala 2.1.1 and higher, you can use the operators IS [NOT] TRUE and IS [NOT] FALSE as equivalents for the built-in functions ISTRUE(), ISNOTTRUE(), ISFALSE(), and ISNOTFALSE().

NONNULLVALUE(type expression)

Purpose: Tests if an expression (of any type) is NULL or not. Returns FALSE if so. The converse of nullvalue().

Return type: BOOLEAN

Usage notes: Primarily for compatibility with code containing industry extensions to SQL.

NULLIF(expr1, expr2)

Purpose: Returns NULL if the two specified arguments are equal. If the specified arguments are not equal, returns the value of *EXPR1*. The data types of the expressions must be compatible, according to the conversion rules from Impala SQL data types on page 13. You cannot use an expression that evaluates to NULL for *EXPR1*; that way, you can distinguish a return value of NULL from an argument value of NULL, which would never match *EXPR2*.

Usage notes: This function is effectively shorthand for a CASE expression of the form:

```
CASE
  WHEN EXPR1 = EXPR2 THEN NULL
  ELSE EXPR1
END
```

It is commonly used in division expressions, to produce a NULL result instead of a divide-by-zero error when the divisor is equal to zero:

```
select 1.0 / nullif(c1,0) as reciprocal from t1;
```

You might also use it for compatibility with other database systems that support the same NULLIF() function.

Return type: same as the initial argument value, except that integer values are promoted to BIGINT and floating-point values are promoted to DOUBLE; use CAST() when inserting into a smaller numeric column

NULLIFZERO(numeric_expr)

Purpose: Returns NULL if the numeric expression evaluates to 0, otherwise returns the result of the expression.

Usage notes: Used to avoid error conditions such as divide-by-zero in numeric calculations. Serves as shorthand for a more elaborate CASE expression, to simplify porting SQL with vendor extensions to Impala.

Return type: Same type as the input argument

NULLVALUE(expression)

Purpose: Tests if an expression (of any type) is NULL or not. Returns TRUE if so. The converse of nonnullvalue().

Return type: BOOLEAN

Usage notes: Primarily for compatibility with code containing industry extensions to SQL.

NVL(type a, type ifNull)

Purpose: Alias for the ISNULL() function. Returns the first argument if the first argument is not NULL. Returns the second argument if the first argument is NULL.

Equivalent to the NVL() function in Oracle Database or IFNULL() in MySQL.

Return type: Same as the first argument value

NVL2(type a, type ifNotNull, type ifNull)

Purpose: Returns the second argument, ifNotNull, if the first argument is not NULL. Returns the third argument, ifNull, if the first argument is NULL.

Equivalent to the NVL2() function in Oracle Database.

Return type: Same as the first argument value

Examples:

```
SELECT NVL2(NULL, 999, 0); -- Returns 0
SELECT NVL2('ABC', 'Is Not Null', 'Is Null'); -- Returns 'Is Not
 Null'
```

ZEROIFNULL(numeric_expr)

Purpose: Returns 0 if the numeric expression evaluates to NULL, otherwise returns the result of the expression.

Usage notes: Used to avoid unexpected results due to unexpected propagation of NULL values in numeric calculations. Serves as shorthand for a more elaborate CASE expression, to simplify porting SQL with vendor extensions to Impala.

Return type: Same type as the input argument

Impala string functions

String functions are classified as those primarily accepting or returning STRING, VARCHAR, or CHAR data types, for example to measure the length of a string or concatenate two strings together.

List of supported string functions

- All the functions that accept STRING arguments also accept the VARCHAR and CHAR types introduced in Impala 2.0.
- Whenever VARCHAR or CHAR values are passed to a function that returns a string value, the return type is normalized to STRING. For example, a call to concat() with a mix of STRING, VARCHAR, and CHAR arguments produces a STRING result.

Function reference:

Impala supports the following string functions:

- ASCII
- BASE64DECODE
- BASE64ENCODE
- BTRIM
- BYTES
- CHAR_LENGTH
- CHR
- CONCAT
- CONCAT_WS
- FIND_IN_SET
- GROUP_CONCAT
- INITCAP
- INSTR
- JARO_DISTANCE, JARO_DIST
- JARO_SIMILARITY, JARO_SIM
- JARO_WINKER_DISTANCE, JW_DST
- JARO_WINKER_SIMILARITY, JW_SIM
- LEFT
- LENGTH
- LEVENSHTEIN, LE_DST
- LOCATE
- LOWER, LCASE
- LPAD
- LTRI
- PARSE_URL
- REGEXP_ESCAPE
- REGEXP_EXTRACT
- REGEXP_LIKE
- REGEXP_REPLACE
- REPEAT
- REPLACE

- REVERSE
- RIGHT
- RPAD
- RTRIM
- SPACE
- SPLIT_PART
- STRLEFT
- STRRIGHT
- SUBSTR, SUBSTRING
- TRANSLATE
- TRIM
- UPPER, UCASE

ASCII(STRING str)

Purpose: Returns the numeric ASCII code of the first character of the argument.

Return type: INT

BASE64DECODE(STRING str)

Purpose:

Return type: STRING

Usage notes:

The functions BASE64ENCODE() and BASE64DECODE() are typically used in combination, to store in an Impala table string data that is problematic to store or transmit. For example, you could use these functions to store string data that uses an encoding other than UTF-8, or to transform the values in contexts that require ASCII values, such as for partition key columns. Keep in mind that base64-encoded values produce different results for string functions such as LENGTH(), MAX(), and MIN() than when those functions are called with the unencoded string values.

The set of characters that can be generated as output from BASE64ENCODE(), or specified in the argument string to BASE64DECODE(), are the ASCII uppercase and lowercase letters (A-Z, a-z), digits (0-9), and the punctuation characters +, /, and =.

All return values produced by BASE64ENCODE() are a multiple of 4 bytes in length. All argument values supplied to BASE64DECODE() must also be a multiple of 4 bytes in length. If a base64-encoded value would otherwise have a different length, it can be padded with trailing = characters to reach a length that is a multiple of 4 bytes.

If the argument string to BASE64DECODE() does not represent a valid base64-encoded value, subject to the constraints of the Impala implementation such as the allowed character set, the function returns NULL.

Examples:

The following examples show how to use BASE64ENCODE() and BASE64DECODE() together to store and retrieve string values:

```
-- An arbitrary string can be encoded in base 64.
-- The length of the output is a multiple of 4 bytes,
-- padded with trailing = characters if necessary.
select base64encode('hello world') as encoded,
  length(base64encode('hello world')) as length;
+------------------+--------+
| encoded          | length |
+------------------+--------+
| aGVsbG8gd29ybGQ= | 16     |
+------------------+--------+
```

```
-- Passing an encoded value to base64decode() produces
-- the original value.
select base64decode('aGVsbG8gd29ybGQ=') as decoded;
+-------------+
| decoded     |
+-------------+
| hello world |
+-------------+
```

These examples demonstrate incorrect encoded values that produce NULL return values when decoded:

```
-- The input value to base64decode() must be a multiple of 4 by
tes.
-- In this case, leaving off the trailing = padding character
-- produces a NULL return value.
select base64decode('aGVsbG8gd29ybGQ') as decoded;
+---------+
| decoded |
+---------+
| NULL    |
+---------+
WARNINGS: UDF WARNING: Invalid base64 string; input length is 15,
  which is not a multiple of 4.

-- The input to base64decode() can only contain certain charac
ters.
-- The $ character in this case causes a NULL return value.
select base64decode('abc$');
+-----------------------+
| base64decode('abc$') |
+-----------------------+
| NULL                  |
+-----------------------+
WARNINGS: UDF WARNING: Could not base64 decode input in space 4;
 actual output length 0
```

These examples demonstrate “round-tripping” of an original string to an encoded string, and back again. This technique is applicable if the original source is in an unknown encoding, or if some intermediate processing stage might cause national characters to be misrepresented:

```
select 'circumflex accents: â, ê, î, ô, û' as original,
  base64encode('circumflex accents: â, ê, î, ô, û') as encoded;
+-----------------------------------+-------------------------
----------------------------+
| original                          | encoded
                            |
+-----------------------------------+----------------------------
--------------------------+
| circumflex accents: â, ê, î, ô, û | Y2lyY3VtZmxleCBhY2NlbnRzOi
DDoiwgw6osIMOuLCDDtCwgw7s= |
+-----------------------------------+-------------------------
----------------------------+

select base64encode('circumflex accents: â, ê, î, ô, û') as encod
ed,
  base64decode(base64encode('circumflex accents: â, ê, î, ô, û'))
 as decoded;
```

```
+-------------------------------------------------------+------
-----------------------------+
| encoded                                               | decoded
                          |
+-------------------------------------------------------+--------
--------------------------+
| Y2lyY3VtZmxleCBhY2NlbnRzOiDDoiwgw6osIMOuLCDDtCwgw7s= | circum
flex accents: â, ê, î, ô, û |
+-------------------------------------------------------+------
----------------------------+
```

BASE64ENCODE(STRING str)

Purpose:

Return type: STRING

Usage notes:

The functions BASE64ENCODE() and BASE64DECODE() are typically used in combination, to store in an Impala table string data that is problematic to store or transmit. For example, you could use these functions to store string data that uses an encoding other than UTF-8, or to transform the values in contexts that require ASCII values, such as for partition key columns. Keep in mind that base64-encoded values produce different results for string functions such as LENGTH(), MAX(), and MIN() than when those functions are called with the unencoded string values.

The set of characters that can be generated as output from BASE64ENCODE(), or specified in the argument string to BASE64DECODE(), are the ASCII uppercase and lowercase letters (A-Z, a-z), digits (0-9), and the punctuation characters +, /, and =.

All return values produced by BASE64ENCODE() are a multiple of 4 bytes in length. All argument values supplied to BASE64DECODE() must also be a multiple of 4 bytes in length. If a base64-encoded value would otherwise have a different length, it can be padded with trailing = characters to reach a length that is a multiple of 4 bytes.

Examples:

The following examples show how to use BASE64ENCODE() and BASE64DECODE() together to store and retrieve string values:

```
-- An arbitrary string can be encoded in base 64.
-- The length of the output is a multiple of 4 bytes,
-- padded with trailing = characters if necessary.
select base64encode('hello world') as encoded,
  length(base64encode('hello world')) as length;
+------------------+--------+
| encoded          | length |
+------------------+--------+
| aGVsbG8gd29ybGQ= | 16     |
+------------------+--------+

-- Passing an encoded value to base64decode() produces
-- the original value.
select base64decode('aGVsbG8gd29ybGQ=') as decoded;
+-------------+
| decoded     |
+-------------+
| hello world |
+-------------+
```

These examples demonstrate incorrect encoded values that produce NULL return values when decoded:

```
-- The input value to base64decode() must be a multiple of 4 by
tes.
-- In this case, leaving off the trailing = padding character
-- produces a NULL return value.
select base64decode('aGVsbG8gd29ybGQ') as decoded;
+---------+
| decoded |
+---------+
| NULL    |
+---------+
WARNINGS: UDF WARNING: Invalid base64 string; input length is 15,
  which is not a multiple of 4.

-- The input to base64decode() can only contain certain charac
ters.
-- The $ character in this case causes a NULL return value.
select base64decode('abc$');
+----------------------+
| base64decode('abc$') |
+----------------------+
| NULL                 |
+----------------------+
WARNINGS: UDF WARNING: Could not base64 decode input in space 4;
 actual output length 0
```

These examples demonstrate “round-tripping” of an original string to an encoded string, and back again. This technique is applicable if the original source is in an unknown encoding, or if some intermediate processing stage might cause national characters to be misrepresented:

```
select 'circumflex accents: â, ê, î, ô, û' as original,
  base64encode('circumflex accents: â, ê, î, ô, û') as encoded;
+----------------------------------+-------------------------
----------------------------+
| original                         | encoded
                            |
+----------------------------------+---------------------------
--------------------------+
| circumflex accents: â, ê, î, ô, û | Y2lyY3VtZmxleCBhY2NlbnRzOi
DDoiwgw6osIMOuLCDDtCwgw7s= |
+----------------------------------+-------------------------
----------------------------+

select base64encode('circumflex accents: â, ê, î, ô, û') as encod
ed,
  base64decode(base64encode('circumflex accents: â, ê, î, ô, û'))
 as decoded;
+-------------------------------------------------------+------
----------------------------+
| encoded                                               | decoded
                          |
+-------------------------------------------------------+--------
--------------------------+
| Y2lyY3VtZmxleCBhY2NlbnRzOiDDoiwgw6osIMOuLCDDtCwgw7s= | circum
flex accents: â, ê, î, ô, û |
+-------------------------------------------------------+------
----------------------------+
```

BTRIM(STRING a), BTRIM(STRING a, STRING chars_to_trim)

Purpose: Removes all instances of one or more characters from the start and end of a STRING value. By default, removes only spaces. If a non-NULL optional second argument is specified, the function removes all occurrences of characters in that second argument from the beginning and end of the string.

Return type: STRING

Examples:

The following examples show the default btrim() behavior, and what changes when you specify the optional second argument. All the examples bracket the output value with [ ] so that you can see any leading or trailing spaces in the btrim() result. By default, the function removes and number of both leading and trailing spaces. When the second argument is specified, any number of occurrences of any character in the second argument are removed from the start and end of the input string; in this case, spaces are not removed (unless they are part of the second argument) and any instances of the characters are not removed if they do not come right at the beginning or end of the string.

```
-- Remove multiple spaces before and one space after.
select concat('[',btrim('    hello '),']');
+---------------------------------------+
| concat('[', btrim('    hello '), ']') |
+---------------------------------------+
| [hello]                               |
+---------------------------------------+
-- Remove any instances of x or y or z at beginning or end. Leave
 spaces alone.
select concat('[',btrim('xy    hello zyzzxx','xyz'),']');
+-------------------------------------------------------+
| concat('[', btrim('xy    hello zyzzxx', 'xyz'), ']') |
+-------------------------------------------------------+
| [    hello ]                                          |
+-------------------------------------------------------+
-- Remove any instances of x or y or z at beginning or end.
-- Leave x, y, z alone in the middle of the string.
select concat('[',btrim('xyhelxyzlozyzzxx','xyz'),']');
+-----------------------------------------------------+
| concat('[', btrim('xyhelxyzlozyzzxx', 'xyz'), ']') |
+-----------------------------------------------------+
| [helxyzlo]                                          |
+-----------------------------------------------------+
```

BYTES

Purpose: Returns the number of bytes contained in the specified byte string.

Syntax: BYTES (byte_expression)

Where:

byte_expression is the byte string to check the number of bytes.

Usage notes:

The BYTES function is similar to the LENGTH() function except that it always returns the number of bytes regardless of the status of UTF-8 mode, ON or OFF.

The following is the list of supported string data types allowed in byte_expression:

- STRING
- VARCHAR
- CHAR

Note: The BYTES (byte_expression) function counts the trailing zeros because trailing zero bytes are considered bytes. In the case of a fixed length column, the length of the value is always equal to the length defined for the column whereas the length of the value in a variable length column is always equal to the number of bytes, including any trailing zero bytes, contained in that value.

Example:

The following example obtains the number of bytes from “cloudera” by applying the BYTES function to the column “cloudera”, which is type VARCHAR.

```
SELECT bytes(cast('cloudera' as varchar(20))) FROM xyz;
            +----------------------------------------+
            | bytes(cast('cloudera' as varchar(20))) |
            +----------------------------------------+
            | 8                                      |
            +----------------------------------------+
```

Note: Use the TRIM function on the byte_expression to exclude the trailing blanks included in the byte count for a data value.

CHAR_LENGTH(STRING a), CHARACTER_LENGTH(STRING a)

Purpose: Returns the length in characters of the argument string. Aliases for the length() function.

Return type: INT

CHR(INT character_code)

Purpose: Returns a character specified by a decimal code point value. The interpretation and display of the resulting character depends on your system locale. Because consistent processing of Impala string values is only guaranteed for values within the ASCII range, only use this function for values corresponding to ASCII characters. In particular, parameter values greater than 255 return an empty string.

Return type: STRING

Usage notes: Can be used as the inverse of the ascii() function, which converts a character to its numeric ASCII code.

Examples:

```
SELECT chr(65);
+---------+
| chr(65) |
+---------+
| A       |
+---------+
SELECT chr(97);
+---------+
| chr(97) |
+---------+
| a       |
+---------+
```

CONCAT(STRING a, STRING b...)

Purpose: Returns a single string representing all the argument values joined together.

If any argument is NULL, the function returns NULL.

Return type: STRING

Usage notes: concat() and concat_ws() are appropriate for concatenating the values of multiple columns within the same row, while group_concat() joins together values from different rows.

CONCAT_WS(STRING sep, STRING a, STRING b...)

Purpose: Returns a single string representing the second and following argument values joined together, delimited by a specified separator.

Return type: STRING

Usage notes: concat() and concat_ws() are appropriate for concatenating the values of multiple columns within the same row, while group_concat() joins together values from different rows.

FIND_IN_SET(STRING str, STRING strList)

Purpose: Returns the position (starting from 1) of the first occurrence of a specified string within a comma-separated string. Returns NULL if either argument is NULL, 0 if the search string is not found, or 0 if the search string contains a comma.

Return type: INT

GROUP_CONCAT(STRING s [, STRING sep])

Purpose: Returns a single string representing the argument value concatenated together for each row of the result set. If the optional separator string is specified, the separator is added between each pair of concatenated values.

Return type: STRING

Usage notes: concat() and concat_ws() are appropriate for concatenating the values of multiple columns within the same row, while group_concat() joins together values from different rows.

By default, returns a single string covering the whole result set. To include other columns or values in the result set, or to produce multiple concatenated strings for subsets of rows, include a GROUP BY clause in the query.

Strictly speaking, group_concat() is an aggregate function, not a scalar function like the others in this list. For additional details and examples, see the GROUP_CONCAT function.

INITCAP(STRING str)

Purpose: Returns the input string with the first letter of each word capitalized and all other letters in lowercase.

Return type: STRING

Example:

INITCAP("i gOt mY ChiCkeNs in tHe yard.") returns "I Got My Chickens In The Yard.".

INSTR(STRING str, STRING substr [, BIGINT position [, BIGINT occurrence]])

Purpose: Returns the position (starting from 1) of the first occurrence of a substring within a longer string.

Return type: INT

Usage notes:

If the *SUBSTR* is not present in *STR*, the function returns 0.

The optional third and fourth arguments let you find instances of the *SUBSTR* other than the first instance starting from the left.

- The third argument, *POSITION*, lets you specify a starting point within the *STR* other than 1.

```
-- Restricting the search to positions 7..end,
-- the first occurrence of 'b' is at position 9.
select instr('foo bar bletch', 'b', 7);
+---------------------------------+
| instr('foo bar bletch', 'b', 7) |
+---------------------------------+
```

```
| 9                                |
+----------------------------------+
```

- If there are no more occurrences after the specified position, the result is 0.
- If *POSITION* is negative, the search works right-to-left starting that many characters from the right. The return value still represents the position starting from the left side of *STR*.

```
-- Scanning right to left, the first occurrence of 'o'
-- is at position 8. (8th character from the left.)
select instr('hello world','o',-1);
+-------------------------------+
| instr('hello world', 'o', -1) |
+-------------------------------+
| 8                             |
+-------------------------------+
```

- The fourth argument, *OCCURRENCE*, lets you specify an occurrence other than the first.

```
-- 2nd occurrence of 'b' is at position 9.
select instr('foo bar bletch', 'b', 1, 2);
+------------------------------------+
| instr('foo bar bletch', 'b', 1, 2) |
+------------------------------------+
| 9                                  |
+------------------------------------+
```

- If *OCCURRENCE* is greater than the number of matching occurrences, the function returns 0.
- *OCCURRENCE* cannot be negative or zero. A non-positive value for this argument causes an error.
- If either of the optional arguments, *POSITION* or *OCCURRENCE*, is NULL, the function also returns NULL.

JARO_DISTANCE(STRING str1, STRING str2)
JARO_DST(STRING str1, STRING str2)

Purpose: Returns the Jaro distance between two input strings. The Jaro distance is a measure of similarity between two strings and is the complementary of JARO_SIMILARITY(), i.e. (1 - JARO _SIMILARITY()).

Return type: DOUBLE

Usage notes:

If the two input strings are identical, the function returns 0.0.

If there is no matching character between the input strings, the function returns 1.0.

If either input strings is NULL, the function returns NULL.

If the length of either input string is bigger than 255 characters, the function returns an error.

JARO_SIMILARITY(STRING str1, STRING str2)
JARO_SIM(STRING str1, STRING str2)

Purpose: Returns the Jaro similarity of two strings. The higher the Jaro similarity for two strings is, the more similar the strings are.

Return type: DOUBLE

Usage notes:

If the two input strings are identical, the function returns 1.0.

If there is no matching character between the input strings, the function returns 0.0.

If either input strings is NULL, the function returns NULL.

If the length of either input string is bigger than 255 characters, the function returns an error.

JARO_WINKLER_DISTANCE(STRING str1, STRING str2[, DOUBLE scaling_factor, DOUBLE boost_threshold])
JW_DST(STRING str1, STRING str2[, DOUBLE scaling_factor, DOUBLE boost_threshold])

Purpose: Returns the Jaro-Winkler distance of two input strings. It is the complementary of JARO_WINKLER_SIMILARITY(), i.e. 1 - JARO_WINKLER_SIMILARITY().

Return type: DOUBLE

Usage notes:

If the two input strings are identical, the function returns 0.0.

If there is no matching character between the input strings, the function returns 1.0.

The function returns an error in the following cases:

- The length of either input string is bigger than 255 characters.
- *SCALING_FACTOR* < 0.0 or *SCALING_FACTOR* > 0.25
- *BOOST_THRESHOLD* < 0.0 or *BOOST_THRESHOLD* > 1.0

If either input strings is NULL, the function returns NULL.

The default *SCALING FACTOR* is 0.1.

The prefix weight will only be applied if the Jaro-distance exceeds the optional *BOOST_THRESHOLD*. By default, the *BOOST_THRESHOLD* value is 0.7.

JARO_WINKLER_SIMILARITY(STRING str1, STRING str2[, DOUBLE scaling_factor, DOUBLE boost_threshold])
JARO_SIM(STRING str1, STRING str2[, DOUBLE scaling_factor, DOUBLE boost_threshold])

Purpose: Returns the Jaro-Winkler Similarity between two input strings. The Jaro-Winkler similarity uses a prefix weight, specified by *SCALING FACTOR*, which gives more favorable ratings to strings that match from the beginning for a set prefix length, up to a maximum of four characters.

Use Jaro or Jaro-Winkler functions to perform fuzzy matches on relatively short strings, e.g. to scrub user inputs of names against the records in the database.

Return type: DOUBLE

Usage notes:

If the two input strings are identical, the function returns 1.0.

If there is no matching character between the input strings, the function returns 0.0.

The function returns an error in the following cases:

- The length of either input string is bigger than 255 characters.
- *SCALING_FACTOR* < 0.0 or *SCALING_FACTOR* > 0.25
- *BOOST_THRESHOLD* < 0.0 or *BOOST_THRESHOLD* > 1.0

If either input strings is NULL, the function returns NULL.

The default *SCALING FACTOR* is 0.1.

The prefix weight will only be applied if the Jaro-similarity exceeds the optional *BOOST_THRESHOLD*. By default, the *BOOST_THRESHOLD* value is 0.7.

LEFT(STRING a, INT num_chars)

See the STRLEFT() function.

LENGTH(STRING a)

Purpose: Returns the length in characters of the argument string.

Return type: INT

LEVENSHTEIN(STRING str1, STRING str2), LE_DST(STRING str1, STRING str2)

Purpose: Returns the Levenshtein distance between two input strings. The Levenshtein distance between two strings is the minimum number of single-character edits required to transform one string to other. The function indicates how different the input strings are.

Return type: INT

Usage notes:

If input strings are equal, the function returns 0.

If either input exceeds 255 characters, the function returns an error.

If either input string is NULL, the function returns NULL.

If the length of one input string is zero, the function returns the length of the other string.

Example:

LEVENSHTEIN ('welcome', 'We come') returns 2, first change to replace 'w' to 'W', and then to replace 'l' to a space character.

LOCATE(STRING substr, STRING str[, INT pos])

Purpose: Returns the position (starting from 1) of the first occurrence of a substring within a longer string, optionally after a particular position.

Return type: INT

LOWER(STRING a), LCASE(STRING a)

Purpose: Returns the argument string converted to all-lowercase.

Return type: STRING

Usage notes:

In Impala 2.5 and higher, you can simplify queries that use many UPPER() and LOWER() calls to do case-insensitive comparisons, by using the ILIKE or IREGEXP operators instead.

LPAD(STRING str, INT len, STRING pad)

Purpose: Returns a string of a specified length, based on the first argument string. If the specified string is too short, it is padded on the left with a repeating sequence of the characters from the pad string. If the specified string is too long, it is truncated on the right.

Return type: STRING

LTRIM(STRING a [, STRING chars_to_trim])

Purpose: Returns the argument string with all occurrences of characters specified by the second argument removed from the left side. Removes spaces if the second argument is not specified.

Return type: STRING

PARSE_URL(STRING urlString, STRING partToExtract [, STRING keyToExtract])

Purpose: Returns the portion of a URL corresponding to a specified part. The part argument can be 'PROTOCOL', 'HOST', 'PATH', 'REF', 'AUTHORITY', 'FILE', 'USERINFO', or 'QUERY'. Uppercase is required for these literal values. When requesting the QUERY portion of the URL, you can optionally specify a key to retrieve just the associated value from the key-value pairs in the query string.

Return type: STRING

Usage notes: This function is important for the traditional Hadoop use case of interpreting web logs. For example, if the web traffic data features raw URLs not divided into separate table columns, you

can count visitors to a particular page by extracting the 'PATH' or 'FILE' field, or analyze search terms by extracting the corresponding key from the 'QUERY' field.

REGEXP_ESCAPE(STRING source)

Purpose: The REGEXP_ESCAPE function returns a string escaped for the special character in RE2 library so that the special characters are interpreted literally rather than as special characters. The following special characters are escaped by the function:

```
.\+*?[^]$(){}=!<>|:-
```

Return type: string

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Because the `impala-shell` interpreter uses the \ character for escaping, use \\ to represent the regular expression escape character in any regular expressions that you submit through `impala-shell`. You might prefer to use the equivalent character class names, such as [[:digit:]] instead of \d which you would have to escape as \\d.

Examples:

This example shows escaping one of special characters in RE2.

```
+-----------------------------------------------------+
| regexp_escape('Hello.world')                        |
+-----------------------------------------------------+
| Hello\.world                                        |
+-----------------------------------------------------+
```

This example shows escaping all the special characters in RE2.

```
+-------------------------------------------------------------+
| regexp_escape('a.b\\c+d*e?f[g]h$i(j)k{l}m=n!o<p>q|r:s-t')  |
+-------------------------------------------------------------+
| a\.b\\c\+d\*e\?f\[g\]h\$i\(j\)k\{l\}m\=n\!o\<p\>q\|r\:s\-t |
+-------------------------------------------------------------+
```

REGEXP_EXTRACT(STRING subject, STRING pattern, INT index)

Purpose: Returns the specified () group from a string based on a regular expression pattern. Group 0 refers to the entire extracted string, while group 1, 2, and so on refers to the first, second, and so on (...) portion.

Return type: STRING

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Because the `impala-shell` interpreter uses the \ character for escaping, use \\ to represent the regular expression escape character in any regular expressions that you submit through `impala-shell`. You might prefer to use the equivalent character class names, such as [[:digit:]] instead of \d which you would have to escape as \\d.

Examples:

This example shows how group 0 matches the full pattern string, including the portion outside any () group:

```
[localhost:21000] > select regexp_extract('abcdef123ghi456jkl','
.*?(\\d+)',0);
+-------------------------------------------------------+
| regexp_extract('abcdef123ghi456jkl', '.*?(\\d+)', 0) |
+-------------------------------------------------------+
| abcdef123ghi456                                       |
+-------------------------------------------------------+
Returned 1 row(s) in 0.11s
```

This example shows how group 1 matches just the contents inside the first () group in the pattern string:

```
[localhost:21000] > select regexp_extract('abcdef123ghi456jkl','
.*?(\\d+)',1);
+-------------------------------------------------------+
| regexp_extract('abcdef123ghi456jkl', '.*?(\\d+)', 1) |
+-------------------------------------------------------+
| 456                                                   |
+-------------------------------------------------------+
Returned 1 row(s) in 0.11s
```

Unlike in earlier Impala releases, the regular expression library used in Impala 2.0 and later supports the .*? idiom for non-greedy matches. This example shows how a pattern string starting with .*? matches the shortest possible portion of the source string, returning the rightmost set of lowercase letters. A pattern string both starting and ending with .*? finds two potential matches of equal length, and returns the first one found (the leftmost set of lowercase letters).

```
[localhost:21000] > select regexp_extract('AbcdBCdefGHI','.*?([[
:lower:]]+)',1);
+-----------------------------------------------------------+
| regexp_extract('abcdbcdefghi', '.*?([[:lower:]]+)', 1) |
+-----------------------------------------------------------+
| def                                                       |
+-----------------------------------------------------------+
[localhost:21000] > select regexp_extract('AbcdBCdefGHI','.*?([[
:lower:]]+).*?',1);
+-------------------------------------------------------------+
| regexp_extract('abcdbcdefghi', '.*?([[:lower:]]+).*?', 1) |
+-------------------------------------------------------------+
| bcd                                                         |
+-------------------------------------------------------------+
```

REGEXP_LIKE(STRING source, STRING pattern[, STRING options])

Purpose: Returns true or false to indicate whether the source string contains anywhere inside it the regular expression given by the pattern. The optional third argument consists of letter flags that change how the match is performed, such as i for case-insensitive matching.

Syntax:

The flags that you can include in the optional third argument are:

- c: Case-sensitive matching (the default).

- i: Case-insensitive matching. If multiple instances of c and i are included in the third argument, the last such option takes precedence.
- m: Multi-line matching. The ^ and $ operators match the start or end of any line within the source string, not the start and end of the entire string.
- n: Newline matching. The . operator can match the newline character. A repetition operator such as .* can match a portion of the source string that spans multiple lines.

Return type: BOOLEAN

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Because the `impala-shell` interpreter uses the \ character for escaping, use \\ to represent the regular expression escape character in any regular expressions that you submit through `impala-shell`. You might prefer to use the equivalent character class names, such as [[:digit:]] instead of \d which you would have to escape as \\d.

Examples:

This example shows how REGEXP_LIKE() can test for the existence of various kinds of regular expression patterns within a source string:

```
-- Matches because the 'f' appears somewhere in 'foo'.
select regexp_like('foo','f');
+-------------------------+
| regexp_like('foo', 'f') |
+-------------------------+
| true                    |
+-------------------------+

-- Does not match because the comparison is case-sensitive by de
fault.
select regexp_like('foo','F');
+-------------------------+
| regexp_like('foo', 'f') |
+-------------------------+
| false                   |
+-------------------------+

-- The 3rd argument can change the matching logic, such as 'i' me
aning case-insensitive.
select regexp_like('foo','F','i');
+------------------------------+
| regexp_like('foo', 'f', 'i') |
+------------------------------+
| true                         |
+------------------------------+
-- The familiar regular expression notations work, such as ^ and
 $ anchors...
select regexp_like('foo','f$');
+--------------------------+
| regexp_like('foo', 'f$') |
+--------------------------+
| false                    |
+--------------------------+
select regexp_like('foo','o$');
```

```
+--------------------------+
| regexp_like('foo', 'o$') |
+--------------------------+
| true                     |
+--------------------------+

-- ...and repetition operators such as * and +
select regexp_like('foooooobar','fo+b');
+-----------------------------------+
| regexp_like('foooooobar', 'fo+b') |
+-----------------------------------+
| true                              |
+-----------------------------------+

select regexp_like('foooooobar','fx*y*o*b');
+---------------------------------------+
| regexp_like('foooooobar', 'fx*y*o*b') |
+---------------------------------------+
| true                                  |
+---------------------------------------+
```

REGEXP_REPLACE(STRING initial, STRING pattern, STRING replacement)

Purpose: Returns the initial argument with the regular expression pattern replaced by the final argument string.

Return type: STRING

In Impala 2.0 and later, the Impala regular expression syntax conforms to the POSIX Extended Regular Expression syntax used by the Google RE2 library. For details, see the *RE2 documentation*. It has most idioms familiar from regular expressions in Perl, Python, and so on, including .*? for non-greedy matches.

In Impala 2.0 and later, a change in the underlying regular expression library could cause changes in the way regular expressions are interpreted by this function. Test any queries that use regular expressions and adjust the expression patterns if necessary.

Because the `impala-shell` interpreter uses the \ character for escaping, use \\ to represent the regular expression escape character in any regular expressions that you submit through `impala-shell`. You might prefer to use the equivalent character class names, such as [[:digit:]] instead of \d which you would have to escape as \\d.

Examples:

These examples show how you can replace parts of a string matching a pattern with replacement text, which can include backreferences to any () groups in the pattern string. The backreference numbers start at 1, and any \ characters must be escaped as \\.

Replace a character pattern with new text:

```
[localhost:21000] > select regexp_replace('aaabbbaaa','b+','xyz'
);
+-----------------------------------------+
| regexp_replace('aaabbbaaa', 'b+', 'xyz') |
+-----------------------------------------+
| aaaxyzaaa                               |
+-----------------------------------------+
Returned 1 row(s) in 0.11s
```

Replace a character pattern with substitution text that includes the original matching text:

```
[localhost:21000] > select regexp_replace('aaabbbaaa','(b+)','<\
\1>');
```

```
+---------------------------------------------+
| regexp_replace('aaabbbaaa', '(b+)', '<\\1>') |
+---------------------------------------------+
| aaa<bbb>aaa                                 |
+---------------------------------------------+
Returned 1 row(s) in 0.11s
```

Remove all characters that are not digits:

```
[localhost:21000] > select regexp_replace('123-456-789','[^[:dig
it:]]','');
+---------------------------------------------------+
| regexp_replace('123-456-789', '[^[:digit:]]', '') |
+---------------------------------------------------+
| 123456789                                         |
+---------------------------------------------------+
Returned 1 row(s) in 0.12s
```

REPEAT(STRING str, INT n)

Purpose: Returns the argument string repeated a specified number of times.

Return type: STRING

REPLACE(STRING initial, STRING target, STRING replacement)

Purpose: Returns the initial argument with all occurrences of the target string replaced by the replacement string.

Return type: STRING

Usage notes:

Because this function does not use any regular expression patterns, it is typically faster than REGE XP_REPLACE() for simple string substitutions.

If any argument is NULL, the return value is NULL.

Matching is case-sensitive.

If the replacement string contains another instance of the target string, the expansion is only performed once, instead of applying again to the newly constructed string.

Examples:

```
-- Replace one string with another.
select replace('hello world','world','earth');
+-----------------------------------------+
| replace('hello world', 'world', 'earth') |
+-----------------------------------------+
| hello earth                             |
+-----------------------------------------+
-- All occurrences of the target string are replaced.
select replace('hello world','o','0');
+----------------------------------+
| replace('hello world', 'o', '0') |
+----------------------------------+
| hell0 w0rld                      |
+----------------------------------+

-- If no match is found, the original string is returned unchang
ed.
select replace('hello world','xyz','abc');
+--------------------------------------+
| replace('hello world', 'xyz', 'abc') |
+--------------------------------------+
```

```
| hello world                          |
+--------------------------------------+
```

REVERSE(STRING a)

Purpose: Returns the argument string with characters in reversed order.

Return type: STRING

RIGHT(STRING a, INT num_chars)

See the STRRIGHT function.

RPAD(STRING str, INT len, STRING pad)

Purpose: Returns a string of a specified length, based on the first argument string. If the specified string is too short, it is padded on the right with a repeating sequence of the characters from the pad string. If the specified string is too long, it is truncated on the right.

Return type: STRING

RTRIM(STRING a [, STRING chars_to_trim])

Purpose: Returns the argument string with all occurrences of characters specified by the second argument removed from the right side. Removes spaces if the second argument is not specified.

Return type: STRING

SPACE(INT n)

Purpose: Returns a concatenated string of the specified number of spaces. Shorthand for repeat(' ',*N*).

Return type: STRING

SPLIT_PART(STRING source, STRING delimiter, BIGINT index)

Purpose: Returns the requested indexth part of the input *SOURCE* string split by the *DELIMITER*.

- If *INDEX* is a positive number, returns the *INDEX*th part from the left within the *SOURCE* string.
- If *INDEX* is a negative number, returns the *INDEX*th part from the right within the *SOURCE* string.
- If *INDEX* is 0, returns an error.

The *DELIMITER* can consist of multiple characters, not just a single character.

All matching of the delimiter is done exactly, not using any regular expression patterns.

Return type: STRING

Examples:

SPLIT_PART('x,y,z',',',2) returns 'y'.

SPLIT_PART('one***two***three','***',2) returns 'two'.

SPLIT_PART('abc@@def@@ghi', '@@', 3) returns 'ghi'.

SPLIT_PART('abc@@def@@ghi', '@@', -3) returns 'abc'.

STRLEFT(STRING a, INT num_chars)

Purpose: Returns the leftmost characters of the string. Shorthand for a call to substr() with 2 arguments.

Return type: STRING

STRRIGHT(STRING a, INT num_chars)

Purpose: Returns the rightmost characters of the string. Shorthand for a call to substr() with 2 arguments.

Return type: STRING

SUBSTR(STRING a, INT start [, INT len]), SUBSTRING(STRING a, INT start [, INT len])

Purpose: Returns the portion of the string starting at a specified point, optionally with a specified maximum length. The characters in the string are indexed starting at 1.

Return type: STRING

TRANSLATE(STRING input, STRING from, STRING to)

Purpose: Returns the input string with each character in the from argument replaced with the corresponding character in the to argument. The characters are matched in the order they appear in from and to.

For example: translate ('hello world','world','earth') returns 'hetta earth'.

Return type: STRING

Usage notes:

If from contains more characters than to, the from characters that are beyond the length of to are removed in the result.

For example:

translate('abcdedg', 'bcd', '1') returns 'a1eg'.

translate('Unit Number#2', '# ', '_') returns 'UnitNumber_2'.

If from is NULL, the function returns NULL.

If to contains more characters than from, the extra characters in to are ignored.

If from contains duplicate characters, the duplicate character is replaced with the first matching character in to.

For example: translate ('hello','ll','67') returns 'he66o'.

TRIM(STRING a)

Purpose: Returns the input string with both leading and trailing spaces removed. The same as passing the string through both LTRIM() and RTRIM().

Usage notes: Often used during data cleansing operations during the ETL cycle, if input values might still have surrounding spaces. For a more general-purpose function that can remove other leading and trailing characters besides spaces, see BTRIM().

Return type: STRING

UPPER(STRING a), UCASE(STRING a)

Purpose: Returns the argument string converted to all-uppercase.

Return type: STRING

Usage notes:

In Impala 2.5 and higher, you can simplify queries that use many UPPER() and LOWER() calls to do case-insensitive comparisons, by using the ILIKE or IREGEXP operators instead.

Related Information

General information on Base64 encoding

GROUP_CONCAT function

Impala miscellaneous functions

Impala supports some utility functions that do not operate on a particular column or data type.

For example:

- CURRENT_DATABASE
- EFFECTIVE_USER
- GET_JSON_OBJECT
- LOGGED_IN_USER
- PID
- SLEEP
- USER
- UUID
- VERSION

CURREN_DATABASE()

Purpose: Returns the database that the session is currently using, either default if no database has been selected, or whatever database the session switched to through a USE statement or the `impalad` -d option.

Return type: STRING

EFFECTIVE_USER()

Purpose: Typically returns the same value as USER(), except if delegation is enabled, in which case it returns the ID of the delegated user.

Return type: STRING

GET_JSON_OBJECT(STRING json_str, STRING selector)

Purpose: Extracts JSON object from the *JSON_STR* based on the *SELECTOR* JSON path and returns the string of the extracted JSON object.

The function returns NULL if the input *JSON_STR* is invalid or if nothing is selected based on the *SELECTOR* JSON path.

The following characters are supported in the *SELECTOR* JSON path:

- $: Denotes the root object
- . : Denotes the child operator
- [] : Denotes the subscript operator for array
- * : Denotes the wildcard for [] or .

Return type: STRING

Examples:

```
---- QUERY
SELECT GET_JSON_OBJECT ('{"a":true, "b":false, "c":true}', '$.*')
;
---- RESULTS
[true,false,true]
```

```
---- QUERY
SELECT GET_JSON_OBJECT(t.json, '$.a.b.c') FROM (VALUES (
 ('{"a": {"b": {"c": 1}}}' AS json),
 ('{"a": {"b": {"c": 2}}}'),
 ('{"a": {"b": {"c": 3}}}')
)) t
---- RESULTS
'1'
'2'
```

```
'3'
```

```
---- QUERY
SELECT GET_JSON_OBJECT(t.json, '$.a'),
 GET_JSON_OBJECT(t.json, '$.b'),
 GET_JSON_OBJECT(t.json, '$.c')
FROM (VALUES (
 ('{"a":1, "b":2, "c":3}' AS json),
 ('{"b":2, "c":3}'),
 ('{"c":3}')
)) t
---- RESULTS
'1','2','3'
'NULL','2','3'
'NULL','NULL','3'
```

```
---- QUERY
SELECT GET_JSON_OBJECT(t.json, '$[1]'),
 GET_JSON_OBJECT(t.json, '$[*]')
FROM (VALUES (
 ('["a", "b", "c"]' AS json),
 ('["a", "b"]'),
 ('["a"]')
)) t
---- RESULTS
'b','["a","b","c"]'
'b','["a","b"]'
'NULL','a'
```

Added in: Impala 3.1

LOGGED_IN_USER()

Purpose: Typically returns the same value as USER(). If delegation is enabled, it returns the ID of the delegated user.

LOGGED_IN_USER() is an alias of EFFECTIVE_USER().

Return type: STRING

Added in: Impala 3.1

PID()

Purpose: Returns the process ID of the `impalad` daemon that the session is connected to. You can use it during low-level debugging, to issue Linux commands that trace, show the arguments, and so on the `impalad` process.

Return type: INT

SLEEP(INT ms)

Purpose: Pauses the query for a specified number of milliseconds. For slowing down queries with small result sets enough to monitor runtime execution, memory usage, or other factors that otherwise would be difficult to capture during the brief interval of query execution. When used in the SELECT list, it is called once for each row in the result set; adjust the number of milliseconds accordingly. For example, a query SELECT *, sleep(5) FROM table_with_1000_rows would take at least 5 seconds to complete (5 milliseconds * 1000 rows in result set). To avoid an excessive number of concurrent queries, use this function for troubleshooting on test and development systems, not for production queries.

Return type: N/A

USER()

Purpose: Returns the username of the Linux user who is connected to the `impalad` daemon. Typically called a single time, in a query without any FROM clause, to understand how authorization settings apply in a security context; once you know the logged-in username, you can check which groups that user belongs to, and from the list of groups you can check which roles are available to those groups through the authorization policy file.

Impala 2.0 and later, USER() returns the full Kerberos principal string, such as user@example.com, in a Kerberized environment.

When delegation is enabled, consider calling the effective_user() function instead.

Return type: STRING

UUID()

Purpose: Returns a universal unique identifier, a 128-bit value encoded as a string with groups of hexadecimal digits separated by dashes.

Each call to UUID() produces a new arbitrary value.

If you get a UUID for each row of a result set, you can use it as a unique identifier within a table, or even a unique ID across tables.

Return type: STRING

Usage notes:

Ascending numeric sequences of type BIGINT are often used as identifiers within a table, and as join keys across multiple tables. The uuid() value is a convenient alternative that does not require storing or querying the highest sequence number. For example, you can use it to quickly construct new unique identifiers during a data import job, or to combine data from different tables without the likelihood of ID collisions.

VERSION()

Purpose: Returns information such as the precise version number and build date for the impalad daemon that you are currently connected to. Typically used to confirm that you are connected to the expected level of Impala to use a particular feature, or to connect to several nodes and confirm they are all running the same level of `impalad`.

Return type: STRING (with one or more embedded newlines)

COORDINATOR()

Purpose: Returns the name of the host which is running the impalad daemon that is acting as the coordinator for the current query.

Return type: STRING

Added in: Impala 3.1

Impala aggregate functions

Aggregate functions are a special category with different rules. These functions calculate a return value across all the items in a result set, so they require a FROM clause in the query.

```
select count(product_id) from product_catalog;
select max(height), avg(height) from census_data where age > 20;
```

Aggregate functions also ignore NULL values rather than returning a NULL result. For example, if some rows have NULL for a particular column, those rows are ignored when computing the AVG() for that column. Likewise, specifying COUNT(*COL_NAME*) in a query counts only those rows where *COL_NAME* contains a non-NULL value.

APPX_MEDIAN function

An aggregate function that returns a value that is approximately the median (midpoint) of values in the set of input values.

Syntax:

```
APPX_MEDIAN([DISTINCT | ALL] EXPRESSION)
```

This function works with any input type, because the only requirement is that the type supports less-than and greater-than comparison operators.

Usage notes:

Because the return value represents the estimated midpoint, it might not reflect the precise midpoint value, especially if the cardinality of the input values is very high. If the cardinality is low (up to approximately 20,000), the result is more accurate because the sampling considers all or almost all of the different values.

Return type: Same as the input value, except for CHAR and VARCHAR arguments which produce a STRING result

The return value is always the same as one of the input values, not an “in-between” value produced by averaging.

Restrictions:

This function cannot be used in an analytic context. That is, the OVER() clause is not allowed at all with this function.

The APPX_MEDIAN function returns only the first 10 characters for string values (string, varchar, char). Additional characters are truncated.

Examples:

The following example uses a table of a million random floating-point numbers ranging up to approximately 50,000. The average is approximately 25,000. Because of the random distribution, we would expect the median to be close to this same number. Computing the precise median is a more intensive operation than computing the average, because it requires keeping track of every distinct value and how many times each occurs. The APPX_MEDIAN() function uses a sampling algorithm to return an approximate result, which in this case is close to the expected value. To make sure that the value is not substantially out of range due to a skewed distribution, subsequent queries confirm that there are approximately 500,000 values higher than the APPX_MEDIAN() value, and approximately 500,000 values lower than the APPX_MEDIAN() value.

```
[localhost:21000] > select min(x), max(x), avg(x) from million_numbers;
+-------------------+-------------------+-------------------+
| min(x)            | max(x)            | avg(x)            |
+-------------------+-------------------+-------------------+
| 4.725693727250069 | 49994.56852674231 | 24945.38563793553 |
+-------------------+-------------------+-------------------+
[localhost:21000] > select appx_median(x) from million_numbers;
+----------------+
| appx_median(x) |
+----------------+
| 24721.6        |
+----------------+
[localhost:21000] > select count(x) as higher from million_numbers where x >
 (select appx_median(x) from million_numbers);
+--------+
| higher |
+--------+
| 502013 |
+--------+
[localhost:21000] > select count(x) as lower from million_numbers where x <
 (select appx_median(x) from million_numbers);
+--------+
| lower  |
+--------+
```

```
| 497987 |
+--------+
```

The following example computes the approximate median using a subset of the values from the table, and then confirms that the result is a reasonable estimate for the midpoint.

```
[localhost:21000] > select appx_median(x) from million_numbers where x betwe
en 1000 and 5000;
+-------------------+
| appx_median(x)    |
+-------------------+
| 3013.107787358159 |
+-------------------+
[localhost:21000] > select count(x) as higher from million_numbers where x b
etween 1000 and 5000 and x > 3013.107787358159;
+--------+
| higher |
+--------+
| 37692  |
+--------+
[localhost:21000] > select count(x) as lower from million_numbers where x
 between 1000 and 5000 and x < 3013.107787358159;
+-------+
| lower |
+-------+
| 37089 |
+-------+
```

AVG function

An aggregate function that returns the average value from a set of numbers or TIMESTAMP values. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to AVG are NULL, AVG returns NULL.

Syntax:

```
AVG([DISTINCT | ALL] EXPRESSION) [OVER (ANALYTIC_CLAUSE)]
```

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Return type: DOUBLE for numeric values; TIMESTAMP for TIMESTAMP values

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
```

```
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
| EUROPE      | 19               |
| EUROPE      | 22               |
| EUROPE      | 23               |
| MIDDLE EAST | 4                |
| MIDDLE EAST | 10               |
| MIDDLE EAST | 11               |
| MIDDLE EAST | 13               |
| MIDDLE EAST | 20               |
+-------------+------------------+

select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
```

```
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Examples:

```
-- Average all the non-NULL values in a column.
insert overwrite avg_t values (2),(4),(6),(null),(null);
-- The average of the above values is 4: (2+4+6) / 3. The 2 NULL values are
 ignored.
select avg(x) from avg_t;
-- Average only certain values from the column.
select avg(x) from t1 where month = 'January' and year = '2013';
-- Apply a calculation to the value of the column before averaging.
select avg(x/3) from t1;
-- Apply a function to the value of the column before averaging.
-- Here we are substituting a value of 0 for all NULLs in the column,
-- so that those rows do factor into the return value.
select avg(isnull(x,0)) from t1;
-- Apply some number-returning function to a string column and average the
results.
-- If column s contains any NULLs, length(s) also returns NULL and those ro
ws are ignored.
select avg(length(s)) from t1;
-- Can also be used in combination with DISTINCT and/or GROUP BY.
-- Return more than one result.
select month, year, avg(page_visits) from web_stats group by month, year;
-- Filter the input to eliminate duplicates before performing the calcula
tion.
select avg(distinct x) from t1;
-- Filter the output after performing the calculation.
select avg(x) from t1 group by y having avg(x) between 1 and 20;
```

The following examples show how to use AVG() in an analytic context. They use a table containing integers from 1 to 10. Notice how the AVG() is reported for each input value, as opposed to the GROUP BY clause which condenses the result set.

```
select x, property, avg(x) over (partition by property) as avg from int_t wh
ere property in ('odd','even');
+----+----------+-----+
| x  | property | avg |
+----+----------+-----+
| 2  | even     | 6   |
| 4  | even     | 6   |
| 6  | even     | 6   |
| 8  | even     | 6   |
| 10 | even     | 6   |
| 1  | odd      | 5   |
| 3  | odd      | 5   |
| 5  | odd      | 5   |
| 7  | odd      | 5   |
| 9  | odd      | 5   |
+----+----------+-----+
```

Adding an ORDER BY clause lets you experiment with results that are cumulative or apply to a moving set of rows (the “window”). The following examples use AVG() in an analytic context (that is, with an OVER() clause) to produce a running average of all the even values, then a running average of all the odd values. The basic ORDER BY x clause implicitly activates a window clause of RANGE BETWEEN UNBOUNDED PRECEDING AND CURREN

T ROW, which is effectively the same as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, therefore all of these examples produce the same results:

```
select x, property,
  avg(x) over (partition by property order by x) as 'cumulative average'
  from int_t where property in ('odd','even');
+----+----------+--------------------+
| x  | property | cumulative average |
+----+----------+--------------------+
| 2  | even     | 2                  |
| 4  | even     | 3                  |
| 6  | even     | 4                  |
| 8  | even     | 5                  |
| 10 | even     | 6                  |
| 1  | odd      | 1                  |
| 3  | odd      | 2                  |
| 5  | odd      | 3                  |
| 7  | odd      | 4                  |
| 9  | odd      | 5                  |
+----+----------+--------------------+

select x, property,
  avg(x) over
  (
    partition by property
    order by x
    range between unbounded preceding and current row
  ) as 'cumulative average'
from int_t where property in ('odd','even');
+----+----------+--------------------+
| x  | property | cumulative average |
+----+----------+--------------------+
| 2  | even     | 2                  |
| 4  | even     | 3                  |
| 6  | even     | 4                  |
| 8  | even     | 5                  |
| 10 | even     | 6                  |
| 1  | odd      | 1                  |
| 3  | odd      | 2                  |
| 5  | odd      | 3                  |
| 7  | odd      | 4                  |
| 9  | odd      | 5                  |
+----+----------+--------------------+
select x, property,
  avg(x) over
  (
    partition by property
    order by x
    rows between unbounded preceding and current row
  ) as 'cumulative average'
  from int_t where property in ('odd','even');
+----+----------+--------------------+
| x  | property | cumulative average |
+----+----------+--------------------+
| 2  | even     | 2                  |
| 4  | even     | 3                  |
| 6  | even     | 4                  |
| 8  | even     | 5                  |
| 10 | even     | 6                  |
| 1  | odd      | 1                  |
| 3  | odd      | 2                  |
| 5  | odd      | 3                  |
| 7  | odd      | 4                  |
```

```
| 9  | odd      | 5                  |
+----+----------+--------------------+
```

The following examples show how to construct a moving window, with a running average taking into account 1 row before and 1 row after the current row, within the same partition (all the even values or all the odd values). Because of a restriction in the Impala RANGE syntax, this type of moving window is possible with the ROWS BETWEEN clause but not the RANGE BETWEEN clause:

```
select x, property,
  avg(x) over
  (
    partition by property
    order by x
    rows between 1 preceding and 1 following
  ) as 'moving average'
  from int_t where property in ('odd','even');
+----+----------+----------------+
| x  | property | moving average |
+----+----------+----------------+
| 2  | even     | 3              |
| 4  | even     | 4              |
| 6  | even     | 6              |
| 8  | even     | 8              |
| 10 | even     | 9              |
| 1  | odd      | 2              |
| 3  | odd      | 3              |
| 5  | odd      | 5              |
| 7  | odd      | 7              |
| 9  | odd      | 8              |
+----+----------+----------------+

-- Doesn't work because of syntax restriction on RANGE clause.
select x, property,
  avg(x) over
  (
    partition by property
    order by x
    range between 1 preceding and 1 following
  ) as 'moving average'
from int_t where property in ('odd','even');
ERROR: AnalysisException: RANGE is only supported with both the lower and
upper bounds UNBOUNDED or one UNBOUNDED and the other CURRENT ROW.
```

Restrictions:

Due to the way arithmetic on FLOAT and DOUBLE columns uses high-performance hardware instructions, and distributed queries can perform these operations in different order for each query, results can vary slightly for aggregate function calls such as SUM() and AVG() for FLOAT and DOUBLE columns, particularly on large data sets where millions or billions of values are summed or averaged. For perfect consistency and repeatability, use the DECIMAL data type for such operations instead of FLOAT or DOUBLE.

COUNT function

An aggregate function that returns the number of rows, or the number of non-NULL rows.

Syntax:

```
COUNT([DISTINCT | ALL] EXPRESSION) [OVER (ANALYTIC_CLAUSE)]
```

Depending on the argument, COUNT() considers rows that meet certain conditions:

- The notation COUNT(*) includes NULL values in the total.

- The notation COUNT(*COLUMN_NAME*) only considers rows where the column contains a non-NULL value.
- You can also combine COUNT with the DISTINCT operator to eliminate duplicates before counting, and to count the combinations of values across multiple columns.

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Return type: BIGINT

Usage notes:

If you frequently run aggregate functions such as MIN(), MAX(), and COUNT(DISTINCT) on partition key columns, consider enabling the OPTIMIZE_PARTITION_KEY_SCANS query option, which optimizes such queries. See the *OPTIMIZE_PARTITION_KEY_SCANS query option* topic for the kinds of queries that this option applies to, and slight differences in how partitions are evaluated when this query option is enabled.

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
```

```
| EUROPE      | 19               |
| EUROPE      | 22               |
| EUROPE      | 23               |
| MIDDLE EAST | 4                |
| MIDDLE EAST | 10               |
| MIDDLE EAST | 11               |
| MIDDLE EAST | 13               |
| MIDDLE EAST | 20               |
+-------------+------------------+

select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Examples:

```
-- How many rows total are in the table, regardless of NULL values?
select count(*) from t1;
-- How many rows are in the table with non-NULL values for a column?
select count(c1) from t1;
-- Count the rows that meet certain conditions.
-- Again, * includes NULLs, so COUNT(*) might be greater than COUNT(col).
select count(*) from t1 where x > 10;
select count(c1) from t1 where x > 10;
-- Can also be used in combination with DISTINCT and/or GROUP BY.
-- Combine COUNT and DISTINCT to find the number of unique values.
-- Must use column names rather than * with COUNT(DISTINCT ...) syntax.
-- Rows with NULL values are not counted.
select count(distinct c1) from t1;
-- Rows with a NULL value in _either_ column are not counted.
select count(distinct c1, c2) from t1;
-- Return more than one result.
select month, year, count(distinct visitor_id) from web_stats group by mon
th, year;
```

The following examples show how to use COUNT() in an analytic context. They use a table containing integers from 1 to 10. Notice how the COUNT() is reported for each input value, as opposed to the GROUP BY clause which condenses the result set.

```
select x, property, count(x) over (partition by property) as count from int_
t where property in ('odd','even');
+----+----------+-------+
| x  | property | count |
+----+----------+-------+
| 2  | even     | 5     |
| 4  | even     | 5     |
| 6  | even     | 5     |
| 8  | even     | 5     |
| 10 | even     | 5     |
| 1  | odd      | 5     |
| 3  | odd      | 5     |
| 5  | odd      | 5     |
| 7  | odd      | 5     |
| 9  | odd      | 5     |
+----+----------+-------+
```

Adding an ORDER BY clause lets you experiment with results that are cumulative or apply to a moving set of rows (the “window”). The following examples use COUNT() in an analytic context (that is, with an OVER() clause) to produce a running count of all the even values, then a running count of all the odd values. The basic ORDER BY x clause implicitly activates a window clause of RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, which is effectively the same as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, therefore all of these examples produce the same results:

```
select x, property,
  count(x) over (partition by property order by x) as 'cumulative count'
  from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative count |
+----+----------+------------------+
| 2  | even     | 1                |
| 4  | even     | 2                |
| 6  | even     | 3                |
| 8  | even     | 4                |
| 10 | even     | 5                |
| 1  | odd      | 1                |
| 3  | odd      | 2                |
| 5  | odd      | 3                |
| 7  | odd      | 4                |
| 9  | odd      | 5                |
+----+----------+------------------+

select x, property,
  count(x) over
  (
    partition by property
    order by x
    range between unbounded preceding and current row
  ) as 'cumulative total'
from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative count |
+----+----------+------------------+
| 2  | even     | 1                |
| 4  | even     | 2                |
| 6  | even     | 3                |
| 8  | even     | 4                |
| 10 | even     | 5                |
```

```
| 1  | odd      | 1                |
| 3  | odd      | 2                |
| 5  | odd      | 3                |
| 7  | odd      | 4                |
| 9  | odd      | 5                |
+----+----------+------------------+

select x, property,
  count(x) over
  (
    partition by property
    order by x
    rows between unbounded preceding and current row
  ) as 'cumulative total'
  from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative count |
+----+----------+------------------+
| 2  | even     | 1                |
| 4  | even     | 2                |
| 6  | even     | 3                |
| 8  | even     | 4                |
| 10 | even     | 5                |
| 1  | odd      | 1                |
| 3  | odd      | 2                |
| 5  | odd      | 3                |
| 7  | odd      | 4                |
| 9  | odd      | 5                |
+----+----------+------------------+
```

The following examples show how to construct a moving window, with a running count taking into account 1 row before and 1 row after the current row, within the same partition (all the even values or all the odd values). Therefore, the count is consistently 3 for rows in the middle of the window, and 2 for rows near the ends of the window, where there is no preceding or no following row in the partition. Because of a restriction in the Impala RANGE syntax, this type of moving window is possible with the ROWS BETWEEN clause but not the RANGE BETWEEN clause:

```
select x, property,
  count(x) over
  (
    partition by property
    order by x
    rows between 1 preceding and 1 following
  ) as 'moving total'
  from int_t where property in ('odd','even');
+----+----------+--------------+
| x  | property | moving total |
+----+----------+--------------+
| 2  | even     | 2            |
| 4  | even     | 3            |
| 6  | even     | 3            |
| 8  | even     | 3            |
| 10 | even     | 2            |
| 1  | odd      | 2            |
| 3  | odd      | 3            |
| 5  | odd      | 3            |
| 7  | odd      | 3            |
| 9  | odd      | 2            |
+----+----------+--------------+
-- Doesn't work because of syntax restriction on RANGE clause.
select x, property,
  count(x) over
  (
    partition by property
```

```
    order by x
    range between 1 preceding and 1 following
  ) as 'moving total'
from int_t where property in ('odd','even');
ERROR: AnalysisException: RANGE is only supported with both the lower and up
per bounds UNBOUNDED or one UNBOUNDED and the other CURRENT ROW.
```

GROUPING() and GROUPING_ID() functions

These aggregate functions are used in conjunction with the GROUP BY ROLLUP, CUBE, and GROUPING SETS clauses.

GROUPING()

GROUPING (column) function indicates whether the column in a GROUP BY list is aggregated or not. This function can be used only on a single column. Grouping returns 1 if the data is aggregated across the specified column or 0 for not aggregated in the result set. So these functions can be used to distinguish between a NULL value in the raw data vs the TOTAL across all values that are returned by ROLLUP, CUBE or GROUPING SETS for a specific column.

Example:

The following example shows GROUPING function applied to the columns "country" and "city".

```
Select country, city, sum(sales) as total_sales
      GROUPING(country) as GP_country,
      GROUPING(city) as GP_city
      from sales_history
      GROUP BY ROLLUP (country, city);
```

```
+--------+----------+-------------+-------------+------------+
| country| city     | total_sales | GP_country  | GP_city    |
+--------+----------+-------------+-------------+------------+
| US     | San Jose | 1000        | 0           | 0          |
| US     | Fremont  | 2000        | 0           | 0          |
| US     | NULL     | 3000        | 0           | 1          |
| Japan  | Hiroshima| 5000        | 0           | 0          |
| Japan  | Tokyo    | 3000        | 0           | 0          |
| Japan  | NULL     | 4000        | 0           | 0          |
| Japan  | NULL     | 12000       | 0           | 1          |
| NULL   | NULL     | 15000       | 1           | 1          |
+--------+----------+-------------+-------------+------------+
```

In the above example the value “NULL” in the third row for the column "city" represents the total sales for both cities (San jose and Fremont) in the US. This is the summary row added by the ROLLUP operation on the column "city". Since this is an aggregated value the result set for the GROUPING (city) is indicated by 1 in the GROUPING alias GP_city column for the corresponding row. However the NULL value in the 6th row for the column city shows the sales for an UNKNOWN city in Japan. SInce this is an actual NULL value in the raw data and does not indicate an aggregate the corresponding GROUPING alias GP_city column is indicated by 0.

GROUPING_ID()

GROUPING_ID function computes the level of grouping and returns an integer value which is unique for each combination of grouping values. The integer value can be interpreted as a bit vector where bits in the binary representation of the integer correspond to each GROUPING() value.

```
GROUPING_ID(col1, col2, col3) = GROUPING(col1) * pow(2,2) + GROUPING(col2) *
 pow(2,1) + GROUPING(col3) * pow(2,0)
```

This function can be used with multiple columns however they must match with the GROUP BY column list.

Return type:

GROUPING_ID function performs the binary to decimal conversion and returns a BIGINT.

Note: GROUPING_ID() with no arguments is equivalent to GROUPING_ID() with all of the grouping columns specified in the group by clause. This function can be used only in the SELECT list, HAVING, or ORDER BY clauses when GROUP BY is specified.

Example:

The following example shows GROUPING_ID applied to the column “country, city” and also matches the "column_expression" in the GROUP BY list.

```
Select country, city, sum(sales) as total_sales
      GROUPING_ID(country, city) as GPID,
     from sales_history
      GROUP BY ROLLUP (country, city);
```

In this example the GROUPING_ID(country, city) function is equivalent to GROUPING(country) * pow(2,1) + GROUPING(city). However, GROUPING_ID(country, city) returns an integer value whereas GROUPING(country) + GROUPING(city) returns a binary string.

```
      +--------+----------+-------------+----------------+
      | country| city     | total_sales | GPID (integer) |
      +--------+----------+-------------+----------------+
      | US     | San Jose | 1000        | 0              |
      | US     | Fremont  | 2000        | 0              |
      | US     | NULL     | 3000        | 1              |
      | Japan  | Hiroshima| 5000        | 0              |
      | Japan  | Tokyo    | 3000        | 0              |
      | Japan  | NULL     | 4000        | 0              |
      | Japan  | NULL     | 12000       | 1              |
      | NULL   | NULL     | 15000       | 3              |
      +--------+----------+-------------+----------------+
```

The following table shows the binary equivalent of the GROUPING_ID () function GPID calcualted in the above example.

```
    +-------------------------------------------+-------------------------
----------------------------------------------------------------------+
    | GROUPING_ID(country, city) GPID in integer | Binary equivalent of GRO
UPING_ID(country, city) = LPAD(BIN(GROUPING_ID(country, city)), 2, ‘0’)|
    +-------------------------------------------+-------------------------
----------------------------------------------------------------------+
    | 0                                          | 00
                                                                       |
    | 0                                          | 00
                                                                       |
    | 1                                          | 01
                                                                       |
    | 0                                          | 00
                                                                       |
    | 0                                          | 00
                                                                       |
    | 0                                          | 00
                                                                       |
    | 1                                          | 01
                                                                       |
```

```
    | 3                                          | 11
                                                                          |
    +-------------------------------------------+-------------------------
-----------------------------------------------------------------------+
```

GROUP_CONCAT function

An aggregate function that returns a single string representing the argument value concatenated together for each row of the result set. If the optional separator string is specified, the separator is added between each pair of concatenated values. The default separator is a comma followed by a space.

Syntax:

```
GROUP_CONCAT([ALL | DISTINCT] EXPRESSION [, SEPARATOR])
```

Usage notes: concat() and concat_ws() are appropriate for concatenating the values of multiple columns within the same row, while group_concat() joins together values from different rows.

By default, returns a single string covering the whole result set. To include other columns or values in the result set, or to produce multiple concatenated strings for subsets of rows, include a GROUP BY clause in the query.

Return type: STRING

This function cannot be used in an analytic context. That is, the OVER() clause is not allowed at all with this function.

Currently, Impala returns an error if the result value grows larger than 1 GiB.

Examples:

The following examples illustrate various aspects of the GROUP_CONCAT() function.

You can call the function directly on a STRING column. To use it with a numeric column, cast the value to STRING.

```
[localhost:21000] > create table t1 (x int, s string);
[localhost:21000] > insert into t1 values (1, "one"), (3, "three"), (2, "
two"), (1, "one");
[localhost:21000] > select group_concat(s) from t1;
+----------------------+
| group_concat(s)      |
+----------------------+
| one, three, two, one |
+----------------------+
[localhost:21000] > select group_concat(cast(x as string)) from t1;
+---------------------------------+
| group_concat(cast(x as string)) |
+---------------------------------+
| 1, 3, 2, 1                      |
+---------------------------------+
```

Specify the DISTINCT keyword to eliminate duplicate values from the concatenated result:

```
[localhost:21000] > select group_concat(distinct s) from t1;
+--------------------------+
| group_concat(distinct s) |
+--------------------------+
| three, two, one          |
+--------------------------+
```

The optional separator lets you format the result in flexible ways. The separator can be an arbitrary string expression, not just a single character.

```
[localhost:21000] > select group_concat(s,"|") from t1;
+----------------------+
| group_concat(s, '|') |
+----------------------+
| one|three|two|one    |
+----------------------+
[localhost:21000] > select group_concat(s,'---') from t1;
+-------------------------+
| group_concat(s, '---')  |
+-------------------------+
| one---three---two---one |
+-------------------------+
```

The default separator is a comma followed by a space. To get a comma-delimited result without extra spaces, specify a delimiter character that is only a comma.

```
[localhost:21000] > select group_concat(s,',') from t1;
+----------------------+
| group_concat(s, ',') |
+----------------------+
| one,three,two,one    |
+----------------------+
```

Including a GROUP BY clause lets you produce a different concatenated result for each group in the result set. In this example, the only X value that occurs more than once is 1, so that is the only row in the result set where GROUP_CONCAT() returns a delimited value. For groups containing a single value, GROUP_CONCAT() returns the original value of its STRING argument.

```
[localhost:21000] > select x, group_concat(s) from t1 group by x;
+---+-----------------+
| x | group_concat(s) |
+---+-----------------+
| 2 | two             |
| 3 | three           |
| 1 | one, one        |
+---+-----------------+
```

MAX function

An aggregate function that returns the maximum value from a set of numbers. Opposite of the MIN function. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to MAX are NULL, MAX returns NULL.

Syntax:

```
MAX([DISTINCT | ALL] EXPRESSION) [OVER (ANALYTIC_CLAUSE)]
```

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Restrictions: In Impala 2.0 and higher, this function can be used as an analytic function, but with restrictions on any window clause. For MAX() and MIN(), the window clause is only allowed if the start bound is UNBOUNDED PRECEDING.

Return type: Same as the input value, except for CHAR and VARCHAR arguments which produce a STRING result

Usage notes:

If you frequently run aggregate functions such as MIN(), MAX(), and COUNT(DISTINCT) on partition key columns, consider enabling the OPTIMIZE_PARTITION_KEY_SCANS query option, which optimizes such queries. See the *OPTIMIZE_PARTITION_KEY_SCANS query option* topic for the kinds of queries that this option applies to, and slight differences in how partitions are evaluated when this query option is enabled.

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
| EUROPE      | 19               |
| EUROPE      | 22               |
| EUROPE      | 23               |
| MIDDLE EAST | 4                |
| MIDDLE EAST | 10               |
| MIDDLE EAST | 11               |
| MIDDLE EAST | 13               |
| MIDDLE EAST | 20               |
+-------------+------------------+
```

```
select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Examples:

```
-- Find the largest value for this column in the table.
select max(c1) from t1;
-- Find the largest value for this column from a subset of the table.
select max(c1) from t1 where month = 'January' and year = '2013';
-- Find the largest value from a set of numeric function results.
select max(length(s)) from t1;
-- Can also be used in combination with DISTINCT and/or GROUP BY.
-- Return more than one result.
select month, year, max(purchase_price) from store_stats group by month,
year;
-- Filter the input to eliminate duplicates before performing the calculat
ion.
select max(distinct x) from t1;
```

The following examples show how to use MAX() in an analytic context. They use a table containing integers from 1 to 10. Notice how the MAX() is reported for each input value, as opposed to the GROUP BY clause which condenses the result set.

```
select x, property, max(x) over (partition by property) as max from int_t wh
ere property in ('odd','even');
+----+----------+-----+
| x  | property | max |
+----+----------+-----+
| 2  | even     | 10  |
| 4  | even     | 10  |
| 6  | even     | 10  |
| 8  | even     | 10  |
| 10 | even     | 10  |
```

```
| 1  | odd      | 9   |
| 3  | odd      | 9   |
| 5  | odd      | 9   |
| 7  | odd      | 9   |
| 9  | odd      | 9   |
+----+----------+-----+
```

Adding an ORDER BY clause lets you experiment with results that are cumulative or apply to a moving set of rows (the “window”). The following examples use MAX() in an analytic context (that is, with an OVER() clause) to display the smallest value of X encountered up to each row in the result set. The examples use two columns in the ORDER BY clause to produce a sequence of values that rises and falls, to illustrate how the MAX() result only increases or stays the same throughout each partition within the result set. The basic ORDER BY x clause implicitly activates a window clause of RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, which is effectively the same as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, therefore all of these examples produce the same results:

```
select x, property,
  max(x) over (order by property, x desc) as 'maximum to this point'
from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | maximum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 7                     |
| 3 | prime    | 7                     |
| 2 | prime    | 7                     |
| 9 | square   | 9                     |
| 4 | square   | 9                     |
| 1 | square   | 9                     |
+---+----------+-----------------------+
select x, property,
  max(x) over
  (
    order by property, x desc
    rows between unbounded preceding and current row
  ) as 'maximum to this point'
from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | maximum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 7                     |
| 3 | prime    | 7                     |
| 2 | prime    | 7                     |
| 9 | square   | 9                     |
| 4 | square   | 9                     |
| 1 | square   | 9                     |
+---+----------+-----------------------+
select x, property,
  max(x) over
  (
    order by property, x desc
    range between unbounded preceding and current row
  ) as 'maximum to this point'
from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | maximum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 7                     |
| 3 | prime    | 7                     |
| 2 | prime    | 7                     |
```

```
| 9 | square   | 9                    |
| 4 | square   | 9                    |
| 1 | square   | 9                    |
+---+----------+----------------------+
```

The following examples show how to construct a moving window, with a running maximum taking into account all rows before and 1 row after the current row. Because of a restriction in the Impala RANGE syntax, this type of moving window is possible with the ROWS BETWEEN clause but not the RANGE BETWEEN clause. Because of an extra Impala restriction on the MAX() and MIN() functions in an analytic context, the lower bound must be UNBO UNDED PRECEDING.

```
select x, property,
  max(x) over
  (
    order by property, x
    rows between unbounded preceding and 1 following
  ) as 'local maximum'
from int_t where property in ('prime','square');
+---+----------+---------------+
| x | property | local maximum |
+---+----------+---------------+
| 2 | prime    | 3             |
| 3 | prime    | 5             |
| 5 | prime    | 7             |
| 7 | prime    | 7             |
| 1 | square   | 7             |
| 4 | square   | 9             |
| 9 | square   | 9             |
+---+----------+---------------+
-- Doesn't work because of syntax restriction on RANGE clause.
select x, property,
  max(x) over
  (
    order by property, x
    range between unbounded preceding and 1 following
  ) as 'local maximum'
from int_t where property in ('prime','square');
ERROR: AnalysisException: RANGE is only supported with both the lower and u
pper bounds UNBOUNDED or one UNBOUNDED and the other CURRENT ROW.
```

MIN function

An aggregate function that returns the minimum value from a set of numbers. Opposite of the MAX function. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to MIN are NULL, MIN returns NULL.

Syntax:

```
MIN([DISTINCT | ALL] EXPRESSION) [OVER (ANALYTIC_CLAUSE)]
```

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Restrictions: In Impala 2.0 and higher, this function can be used as an analytic function, but with restrictions on any window clause. For MAX() and MIN(), the window clause is only allowed if the start bound is UNBOUNDED PR ECEDING.

Return type: Same as the input value, except for CHAR and VARCHAR arguments which produce a STRING result

Usage notes:

If you frequently run aggregate functions such as MIN(), MAX(), and COUNT(DISTINCT) on partition key columns, consider enabling the OPTIMIZE_PARTITION_KEY_SCANS query option, which optimizes such queries. See the

OPTIMIZE_PARTITION_KEY_SCANS query option topic for the kinds of queries that this option applies to, and slight differences in how partitions are evaluated when this query option is enabled.

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
| EUROPE      | 19               |
| EUROPE      | 22               |
| EUROPE      | 23               |
| MIDDLE EAST | 4                |
| MIDDLE EAST | 10               |
| MIDDLE EAST | 11               |
| MIDDLE EAST | 13               |
| MIDDLE EAST | 20               |
+-------------+------------------+

select
```

```
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Examples:

```
-- Find the smallest value for this column in the table.
select min(c1) from t1;
-- Find the smallest value for this column from a subset of the table.
select min(c1) from t1 where month = 'January' and year = '2013';
-- Find the smallest value from a set of numeric function results.
select min(length(s)) from t1;
-- Can also be used in combination with DISTINCT and/or GROUP BY.
-- Return more than one result.
select month, year, min(purchase_price) from store_stats group by month, y
ear;
-- Filter the input to eliminate duplicates before performing the calculati
on.
select min(distinct x) from t1;
```

The following examples show how to use MIN() in an analytic context. They use a table containing integers from 1 to 10. Notice how the MIN() is reported for each input value, as opposed to the GROUP BY clause which condenses the result set.

```
select x, property, min(x) over (partition by property) as min from int_t wh
ere property in ('odd','even');
+----+----------+-----+
| x  | property | min |
+----+----------+-----+
| 2  | even     | 2   |
| 4  | even     | 2   |
| 6  | even     | 2   |
| 8  | even     | 2   |
| 10 | even     | 2   |
| 1  | odd      | 1   |
| 3  | odd      | 1   |
```

```
| 5  | odd      | 1   |
| 7  | odd      | 1   |
| 9  | odd      | 1   |
+----+----------+-----+
```

Adding an ORDER BY clause lets you experiment with results that are cumulative or apply to a moving set of rows (the “window”). The following examples use MIN() in an analytic context (that is, with an OVER() clause) to display the smallest value of X encountered up to each row in the result set. The examples use two columns in the ORDER BY clause to produce a sequence of values that rises and falls, to illustrate how the MIN() result only decreases or stays the same throughout each partition within the result set. The basic ORDER BY x clause implicitly activates a window clause of RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, which is effectively the same as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, therefore all of these examples produce the same results:

```
select x, property, min(x) over (order by property, x desc) as 'minimum to
this point'
  from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | minimum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 5                     |
| 3 | prime    | 3                     |
| 2 | prime    | 2                     |
| 9 | square   | 2                     |
| 4 | square   | 2                     |
| 1 | square   | 1                     |
+---+----------+-----------------------+

select x, property,
  min(x) over
  (
    order by property, x desc
    range between unbounded preceding and current row
  ) as 'minimum to this point'
from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | minimum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 5                     |
| 3 | prime    | 3                     |
| 2 | prime    | 2                     |
| 9 | square   | 2                     |
| 4 | square   | 2                     |
| 1 | square   | 1                     |
+---+----------+-----------------------+
select x, property,
  min(x) over
  (
    order by property, x desc
    rows between unbounded preceding and current row
  ) as 'minimum to this point'
from int_t where property in ('prime','square');
+---+----------+-----------------------+
| x | property | minimum to this point |
+---+----------+-----------------------+
| 7 | prime    | 7                     |
| 5 | prime    | 5                     |
| 3 | prime    | 3                     |
| 2 | prime    | 2                     |
| 9 | square   | 2                     |
```

```
| 4 | square   | 2                     |
| 1 | square   | 1                     |
+---+----------+-----------------------+
```

The following examples show how to construct a moving window, with a running minimum taking into account all rows before and 1 row after the current row. Because of a restriction in the Impala RANGE syntax, this type of moving window is possible with the ROWS BETWEEN clause but not the RANGE BETWEEN clause. Because of an extra Impala restriction on the MAX() and MIN() functions in an analytic context, the lower bound must be UNBO UNDED PRECEDING.

```
select x, property,
  min(x) over
  (
    order by property, x desc
    rows between unbounded preceding and 1 following
  ) as 'local minimum'
from int_t where property in ('prime','square');
+---+----------+---------------+
| x | property | local minimum |
+---+----------+---------------+
| 7 | prime    | 5             |
| 5 | prime    | 3             |
| 3 | prime    | 2             |
| 2 | prime    | 2             |
| 9 | square   | 2             |
| 4 | square   | 1             |
| 1 | square   | 1             |
+---+----------+---------------+
-- Doesn't work because of syntax restriction on RANGE clause.
select x, property,
  min(x) over
  (
    order by property, x desc
    range between unbounded preceding and 1 following
  ) as 'local minimum'
from int_t where property in ('prime','square');
ERROR: AnalysisException: RANGE is only supported with both the lower and u
pper bounds UNBOUNDED or one UNBOUNDED and the other CURRENT ROW.
```

NDV function

An aggregate function that returns an approximate value similar to the result of COUNT(DISTINCT *COL*), the “number of distinct values”. It is much faster than the combination of COUNT and DISTINCT, and uses a constant amount of memory and thus is less memory-intensive for columns with high cardinality.

Syntax:

```
NDV([DISTINCT | ALL] EXPRESSION [,scale])
```

Note: The optional argument scale must be an integer and can be in the range from 1 to 10 and maps to a precision used by the HyperLogLog (HLL) algorithm with the following mapping formula:

```
precision = scale + 8
```

Therefore a scale of 1 is mapped to a precision of 9 and a scale of 10 is mapped to a precision of 18.

Without the optional argument, the precision which determines the total number of different estimators in the HLL algorithm will be still 10.

A large precision value generally produces a better estimation with less error than a small precision value. This is due to the extra number of estimators involved. The expense is at the need of extra memory. For a given precision p, the amount of memory used by the HLL algorithm is in the order of 2^p bytes.

When provided a scale of 10 against a total of 22 distinct data sets loaded into external Impala tables, the error will be computed as abs(<true_unique_value> - <estimated_unique_value>) / <true_unique_value>

The scale of 10, mapped to the precision of 18, yielded the worst estimation error at 0.42% (for one set of 10 million integers), and average error no more than 0.17%. This was at the cost of 256Kb of memory for the internal data structure per evaluation of the HLL algorithm.

Usage notes:

This is the mechanism used internally by the COMPUTE STATS statement for computing the number of distinct values in a column.

Because this number is an estimate, it might not reflect the precise number of different values in the column, especially if the cardinality is very low or very high. If the estimated number is higher than the number of rows in the table, Impala adjusts the value internally during query planning.

Return type: DOUBLE in Impala 2.0 and higher; STRING in earlier releases

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
| AMERICA     | 24               |
```

```
| ASIA        | 8                |
| ASIA        | 9                |
| ASIA        | 12               |
| ASIA        | 18               |
| ASIA        | 21               |
| EUROPE      | 6                |
| EUROPE      | 7                |
| EUROPE      | 19               |
| EUROPE      | 22               |
| EUROPE      | 23               |
| MIDDLE EAST | 4                |
| MIDDLE EAST | 10               |
| MIDDLE EAST | 11               |
| MIDDLE EAST | 13               |
| MIDDLE EAST | 20               |
+-------------+------------------+

select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Restrictions:

This function cannot be used in an analytic context. That is, the OVER() clause is not allowed at all with this function.

Examples:

The following example queries a billion-row table to illustrate the relative performance of COUNT(DISTINCT) and NDV(). It shows how COUNT(DISTINCT) gives a precise answer, but is inefficient for large-scale data where an approximate result is sufficient. The NDV() function gives an approximate result but is much faster.

```
select count(distinct col1) from sample_data;
+---------------------+
| count(distinct col1)|
+---------------------+
| 100000              |
```

```
+---------------------+
Fetched 1 row(s) in 20.13s

select cast(ndv(col1) as bigint) as col1 from sample_data;
+----------+
| col1     |
+----------+
| 139017   |
+----------+
Fetched 1 row(s) in 8.91s
```

The following example shows how you can code multiple NDV() calls in a single query, to easily learn which columns have substantially more or fewer distinct values. This technique is faster than running a sequence of queries with COUNT(DISTINCT) calls.

```
select cast(ndv(col1) as bigint) as col1, cast(ndv(col2) as bigint) as col2,
    cast(ndv(col3) as bigint) as col3, cast(ndv(col4) as bigint) as col4
  from sample_data;
+----------+-----------+------------+-----------+
| col1     | col2      | col3       | col4      |
+----------+-----------+------------+-----------+
| 139017   | 282       | 46         | 145636240 |
+----------+-----------+------------+-----------+
Fetched 1 row(s) in 34.97s

select count(distinct col1) from sample_data;
+---------------------+
| count(distinct col1)|
+---------------------+
| 100000              |
+---------------------+
Fetched 1 row(s) in 20.13s

select count(distinct col2) from sample_data;
+----------------------+
| count(distinct col2) |
+----------------------+
| 278                  |
+----------------------+
Fetched 1 row(s) in 20.09s
select count(distinct col3) from sample_data;
+-----------------------+
| count(distinct col3)  |
+-----------------------+
| 46                    |
+-----------------------+
Fetched 1 row(s) in 19.12s
select count(distinct col4) from sample_data;
+----------------------+
| count(distinct col4) |
+----------------------+
| 147135880            |
+----------------------+
Fetched 1 row(s) in 266.95s
```

STDDEV, STDDEV_SAMP, STDDEV_POP functions

An aggregate function that returns the standard deviation of a set of numbers.

Syntax:

```
{ STDDEV | STDDEV_SAMP | STDDEV_POP } ([DISTINCT | ALL] EXPRESSION)
```

This function works with any numeric data type.

Return type: DOUBLE in Impala 2.0 and higher; STRING in earlier releases

This function is typically used in mathematical formulas related to probability distributions.

The STDDEV_POP() and STDDEV_SAMP() functions compute the population standard deviation and sample standard deviation, respectively, of the input values. (STDDEV() is an alias for STDDEV_SAMP().) Both functions evaluate all input rows matched by the query. The difference is that STDDEV_SAMP() is scaled by 1/(N-1) while STDDEV_POP() is scaled by 1/N.

If no input rows match the query, the result of any of these functions is NULL. If a single input row matches the query, the result of any of these functions is "0.0".

Examples:

This example demonstrates how STDDEV() and STDDEV_SAMP() return the same result, while STDDEV_POP() uses a slightly different calculation to reflect that the input data is considered part of a larger “population”.

```
[localhost:21000] > select stddev(score) from test_scores;
+---------------+
| stddev(score) |
+---------------+
| 28.5          |
+---------------+
[localhost:21000] > select stddev_samp(score) from test_scores;
+--------------------+
| stddev_samp(score) |
+--------------------+
| 28.5               |
+--------------------+
[localhost:21000] > select stddev_pop(score) from test_scores;
+-------------------+
| stddev_pop(score) |
+-------------------+
| 28.4858           |
+-------------------+
```

This example demonstrates that, because the return value of these aggregate functions is a STRING, you must currently convert the result with CAST.

```
[localhost:21000] > create table score_stats as select cast(stddev(score) as
 decimal(7,4)) `standard_deviation`, cast(variance(score) as decimal(7,4)) `
variance` from test_scores;
+-------------------+
| summary           |
+-------------------+
| Inserted 1 row(s) |
+-------------------+
[localhost:21000] > desc score_stats;
+--------------------+--------------+---------+
| name               | type         | comment |
+--------------------+--------------+---------+
| standard_deviation | decimal(7,4) |         |
| variance           | decimal(7,4) |         |
+--------------------+--------------+---------+
```

Restrictions:

This function cannot be used in an analytic context. That is, the OVER() clause is not allowed at all with this function.

Related information:

The STDDEV(), STDDEV_POP(), and STDDEV_SAMP() functions compute the standard deviation (square root of the variance) based on the results of VARIANCE(), VARIANCE_POP(), and VARIANCE_SAMP() respectively. See VARIANCE, VARIANCE_SAMP, VARIANCE_POP, VAR_SAMP, VAR_POP functions on page 422 for details about the variance property.

Related Information

Standard deviation

SUM function

An aggregate function that returns the sum of a set of numbers. Its single argument can be numeric column, or the numeric result of a function or expression applied to the column value. Rows with a NULL value for the specified column are ignored. If the table is empty, or all the values supplied to MIN are NULL, SUM returns NULL.

Syntax:

```
SUM([DISTINCT | ALL] EXPRESSION) [OVER (ANALYTIC_CLAUSE)]
```

When the query contains a GROUP BY clause, returns one value for each combination of grouping values.

Return type: BIGINT for integer arguments, DOUBLE for floating-point arguments

Complex type considerations:

To access a column with a complex type (ARRAY, STRUCT, or MAP) in an aggregation function, you unpack the individual elements using join notation in the query, and then apply the function to the final scalar item, field, key, or value at the bottom of any nested type hierarchy in the column.

The following example demonstrates calls to several aggregation functions using values from a column containing nested complex types (an ARRAY of STRUCT items). The array is unpacked inside the query using join notation. The array elements are referenced using the ITEM pseudocolumn, and the structure fields inside the array elements are referenced using dot notation. Numeric values such as SUM() and AVG() are computed using the numeric R_NATIONKEY field, and the general-purpose MAX() and MIN() values are computed from the string N_NAME field.

```
describe region;
+-------------+-------------------------+---------+
| name        | type                    | comment |
+-------------+-------------------------+---------+
| r_regionkey | smallint                |         |
| r_name      | string                  |         |
| r_comment   | string                  |         |
| r_nations   | array<struct<           |         |
|             |   n_nationkey:smallint, |         |
|             |   n_name:string,        |         |
|             |   n_comment:string      |         |
|             | >>                      |         |
+-------------+-------------------------+---------+

select r_name, r_nations.item.n_nationkey
  from region, region.r_nations as r_nations
order by r_name, r_nations.item.n_nationkey;
+-------------+------------------+
| r_name      | item.n_nationkey |
+-------------+------------------+
| AFRICA      | 0                |
| AFRICA      | 5                |
| AFRICA      | 14               |
| AFRICA      | 15               |
| AFRICA      | 16               |
| AMERICA     | 1                |
| AMERICA     | 2                |
| AMERICA     | 3                |
| AMERICA     | 17               |
```

```
| AMERICA     | 24                |
| ASIA        | 8                 |
| ASIA        | 9                 |
| ASIA        | 12                |
| ASIA        | 18                |
| ASIA        | 21                |
| EUROPE      | 6                 |
| EUROPE      | 7                 |
| EUROPE      | 19                |
| EUROPE      | 22                |
| EUROPE      | 23                |
| MIDDLE EAST | 4                 |
| MIDDLE EAST | 10                |
| MIDDLE EAST | 11                |
| MIDDLE EAST | 13                |
| MIDDLE EAST | 20                |
+-------------+-------------------+

select
  r_name,
  count(r_nations.item.n_nationkey) as count,
  sum(r_nations.item.n_nationkey) as sum,
  avg(r_nations.item.n_nationkey) as avg,
  min(r_nations.item.n_name) as minimum,
  max(r_nations.item.n_name) as maximum,
  ndv(r_nations.item.n_nationkey) as distinct_vals
from
  region, region.r_nations as r_nations
group by r_name
order by r_name;
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| r_name      | count | sum | avg  | minimum   | maximum        | distinct
_vals |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
| AFRICA      | 5     | 50  | 10   | ALGERIA   | MOZAMBIQUE     | 5
    |
| AMERICA     | 5     | 47  | 9.4  | ARGENTINA | UNITED STATES  | 5
    |
| ASIA        | 5     | 68  | 13.6 | CHINA     | VIETNAM        | 5
    |
| EUROPE      | 5     | 77  | 15.4 | FRANCE    | UNITED KINGDOM | 5
    |
| MIDDLE EAST | 5     | 58  | 11.6 | EGYPT     | SAUDI ARABIA   | 5
    |
+-------------+-------+-----+------+-----------+----------------+-----------
----+
```

Examples:

The following example shows how to use SUM() to compute the total for all the values in the table, a subset of values, or the sum for each combination of values in the GROUP BY clause:

```
-- Total all the values for this column in the table.
select sum(c1) from t1;
-- Find the total for this column from a subset of the table.
select sum(c1) from t1 where month = 'January' and year = '2013';
-- Find the total from a set of numeric function results.
select sum(length(s)) from t1;
-- Often used with functions that return predefined values to compute a s
core.
```

```
select sum(case when grade = 'A' then 1.0 when grade = 'B' then 0.75 else 0)
 as class_honors from test_scores;
-- Can also be used in combination with DISTINCT and/or GROUP BY.
-- Return more than one result.
select month, year, sum(purchase_price) from store_stats group by month,
year;
-- Filter the input to eliminate duplicates before performing the calculat
ion.
select sum(distinct x) from t1;
```

The following examples show how to use SUM() in an analytic context. They use a table containing integers from 1 to 10. Notice how the SUM() is reported for each input value, as opposed to the GROUP BY clause which condenses the result set.

```
select x, property, sum(x) over (partition by property) as sum from int_t w
here property in ('odd','even');
+----+----------+-----+
| x  | property | sum |
+----+----------+-----+
| 2  | even     | 30  |
| 4  | even     | 30  |
| 6  | even     | 30  |
| 8  | even     | 30  |
| 10 | even     | 30  |
| 1  | odd      | 25  |
| 3  | odd      | 25  |
| 5  | odd      | 25  |
| 7  | odd      | 25  |
| 9  | odd      | 25  |
+----+----------+-----+
```

Adding an ORDER BY clause lets you experiment with results that are cumulative or apply to a moving set of rows (the “window”). The following examples use SUM() in an analytic context (that is, with an OVER() clause) to produce a running total of all the even values, then a running total of all the odd values. The basic ORDER BY x clause implicitly activates a window clause of RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, which is effectively the same as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, therefore all of these examples produce the same results:

```
select x, property,
  sum(x) over (partition by property order by x) as 'cumulative total'
  from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative total |
+----+----------+------------------+
| 2  | even     | 2                |
| 4  | even     | 6                |
| 6  | even     | 12               |
| 8  | even     | 20               |
| 10 | even     | 30               |
| 1  | odd      | 1                |
| 3  | odd      | 4                |
| 5  | odd      | 9                |
| 7  | odd      | 16               |
| 9  | odd      | 25               |
+----+----------+------------------+

select x, property,
  sum(x) over
  (
    partition by property
    order by x
    range between unbounded preceding and current row
```

```
  ) as 'cumulative total'
from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative total |
+----+----------+------------------+
| 2  | even     | 2                |
| 4  | even     | 6                |
| 6  | even     | 12               |
| 8  | even     | 20               |
| 10 | even     | 30               |
| 1  | odd      | 1                |
| 3  | odd      | 4                |
| 5  | odd      | 9                |
| 7  | odd      | 16               |
| 9  | odd      | 25               |
+----+----------+------------------+

select x, property,
  sum(x) over
  (
    partition by property
    order by x
    rows between unbounded preceding and current row
  ) as 'cumulative total'
  from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative total |
+----+----------+------------------+
| 2  | even     | 2                |
| 4  | even     | 6                |
| 6  | even     | 12               |
| 8  | even     | 20               |
| 10 | even     | 30               |
| 1  | odd      | 1                |
| 3  | odd      | 4                |
| 5  | odd      | 9                |
| 7  | odd      | 16               |
| 9  | odd      | 25               |
+----+----------+------------------+
```

Changing the direction of the ORDER BY clause causes the intermediate results of the cumulative total to be calculated in a different order:

```
select sum(x) over (partition by property order by x desc) as 'cumulative
 total'
  from int_t where property in ('odd','even');
+----+----------+------------------+
| x  | property | cumulative total |
+----+----------+------------------+
| 10 | even     | 10               |
| 8  | even     | 18               |
| 6  | even     | 24               |
| 4  | even     | 28               |
| 2  | even     | 30               |
| 9  | odd      | 9                |
| 7  | odd      | 16               |
| 5  | odd      | 21               |
| 3  | odd      | 24               |
| 1  | odd      | 25               |
+----+----------+------------------+
```

The following examples show how to construct a moving window, with a running total taking into account 1 row before and 1 row after the current row, within the same partition (all the even values or all the odd values). Because

of a restriction in the Impala RANGE syntax, this type of moving window is possible with the ROWS BETWEEN clause but not the RANGE BETWEEN clause:

```
select x, property,
  sum(x) over
  (
    partition by property
    order by x
    rows between 1 preceding and 1 following
  ) as 'moving total'
  from int_t where property in ('odd','even');
+----+----------+--------------+
| x  | property | moving total |
+----+----------+--------------+
| 2  | even     | 6            |
| 4  | even     | 12           |
| 6  | even     | 18           |
| 8  | even     | 24           |
| 10 | even     | 18           |
| 1  | odd      | 4            |
| 3  | odd      | 9            |
| 5  | odd      | 15           |
| 7  | odd      | 21           |
| 9  | odd      | 16           |
+----+----------+--------------+
-- Doesn't work because of syntax restriction on RANGE clause.
select x, property,
  sum(x) over
  (
    partition by property
    order by x
    range between 1 preceding and 1 following
  ) as 'moving total'
from int_t where property in ('odd','even');
ERROR: AnalysisException: RANGE is only supported with both the lower and up
per bounds UNBOUNDED or one UNBOUNDED and the other CURRENT ROW.
```

Restrictions:

Due to the way arithmetic on FLOAT and DOUBLE columns uses high-performance hardware instructions, and distributed queries can perform these operations in different order for each query, results can vary slightly for aggregate function calls such as SUM() and AVG() for FLOAT and DOUBLE columns, particularly on large data sets where millions or billions of values are summed or averaged. For perfect consistency and repeatability, use the DECIMAL data type for such operations instead of FLOAT or DOUBLE.

VARIANCE, VARIANCE_SAMP, VARIANCE_POP, VAR_SAMP, VAR_POP functions

An aggregate function that returns the variance of a set of numbers. This is a mathematical property that signifies how far the values spread apart from the mean. The return value can be zero (if the input is a single value, or a set of identical values), or a positive number otherwise.

Syntax:

```
{ VARIANCE | VAR[IANCE]_SAMP | VAR[IANCE]_POP } ([DISTINCT | ALL
] EXPRESSION)
```

This function works with any numeric data type.

Return type: DOUBLE in Impala 2.0 and higher; STRING in earlier releases

This function is typically used in mathematical formulas related to probability distributions.

The VARIANCE_SAMP() and VARIANCE_POP() functions compute the sample variance and population variance, respectively, of the input values. (VARIANCE() is an alias for VARIANCE_SAMP().) Both functions evaluate all input rows matched by the query. The difference is that STDDEV_SAMP() is scaled by 1/(N-1) while STDDEV_POP() is scaled by 1/N.

The functions VAR_SAMP() and VAR_POP() are the same as VARIANCE_SAMP() and VARIANCE_POP(), respectively. These aliases are available in Impala 2.0 and later.

If no input rows match the query, the result of any of these functions is NULL. If a single input row matches the query, the result of any of these functions is "0.0".

Examples:

This example demonstrates how VARIANCE() and VARIANCE_SAMP() return the same result, while VARIANCE_POP() uses a slightly different calculation to reflect that the input data is considered part of a larger “population”.

```
[localhost:21000] > select variance(score) from test_scores;
+-----------------+
| variance(score) |
+-----------------+
| 812.25          |
+-----------------+
[localhost:21000] > select variance_samp(score) from test_scores;
+----------------------+
| variance_samp(score) |
+----------------------+
| 812.25               |
+----------------------+
[localhost:21000] > select variance_pop(score) from test_scores;
+---------------------+
| variance_pop(score) |
+---------------------+
| 811.438             |
+---------------------+
```

This example demonstrates that, because the return value of these aggregate functions is a STRING, you convert the result with CAST if you need to do further calculations as a numeric value.

```
[localhost:21000] > create table score_stats as select cast(stddev(score) as
 decimal(7,4)) `standard_deviation`, cast(variance(score) as decimal(7,4)) `
variance` from test_scores;
+-------------------+
| summary           |
+-------------------+
| Inserted 1 row(s) |
+-------------------+
[localhost:21000] > desc score_stats;
+--------------------+--------------+---------+
| name               | type         | comment |
+--------------------+--------------+---------+
| standard_deviation | decimal(7,4) |         |
| variance           | decimal(7,4) |         |
+--------------------+--------------+---------+
```

Restrictions:

This function cannot be used in an analytic context. That is, the OVER() clause is not allowed at all with this function.

Related information:

The STDDEV(), STDDEV_POP(), and STDDEV_SAMP() functions compute the standard deviation (square root of the variance) based on the results of VARIANCE(), VARIANCE_POP(), and VARIANCE_SAMP() respectively.

See STDDEV, STDDEV_SAMP, STDDEV_POP functions on page 416 for details about the standard deviation property.

Related Information

Mathematical variance

Impala analytic functions

Analytic functions (also known as window functions) are a special category of built-in functions. Like aggregate functions, they examine the contents of multiple input rows to compute each output value. However, rather than being limited to one result value per GROUP BY group, they operate on *windows* where the input rows are ordered and grouped using flexible conditions expressed through an OVER() clause.

Some functions, such as LAG() and RANK(), can only be used in this analytic context. Some aggregate functions do double duty: when you call the aggregation functions such as MAX(), SUM(), AVG(), and so on with an OVER() clause, they produce an output value for each row, based on computations across other rows in the window.

Although analytic functions often compute the same value you would see from an aggregate function in a GROUP BY query, the analytic functions produce a value for each row in the result set rather than a single value for each group. This flexibility lets you include additional columns in the SELECT list, offering more opportunities for organizing and filtering the result set.

Analytic function calls are only allowed in the SELECT list and in the outermost ORDER BY clause of the query. During query processing, analytic functions are evaluated after other query stages such as joins, WHERE, and GROUP BY,

The rows that are part of each partition are analyzed by computations across an ordered or unordered set of rows. For example, COUNT() and SUM() might be applied to all the rows in the partition, in which case the order of analysis does not matter. The ORDER BY clause might be used inside the OVER() clause to defines the ordering that applies to functions such as LAG() and FIRST_VALUE().

Analytic functions are frequently used in fields such as finance and science to provide trend, outlier, and bucketed analysis for large data sets. You might also see the term “window functions” in database literature, referring to the sequence of rows (the “window”) that the function call applies to, particularly when the OVER clause includes a ROWS or RANGE keyword.

Related Information

OVER
WINDOW
AVG
COUNT
CUME_DIST
DENSE_RANK
FIRST_VALUE
LAG
LAST_VALUE
LEAD
MAX
MIN
NTILE
PERCENT_RANK
RANK
ROW_NUMBER
SUM

OVER

The OVER clause is required for calls to pure analytic functions such as LEAD(), RANK(), and FIRST_VALUE(). When you include an OVER clause with calls to aggregate functions such as MAX(), COUNT(), or SUM(), they operate as analytic functions.

Syntax:

```
function(ARGS) OVER([PARTITION_BY_CLAUSE] [ORDER_BY_CLAUSE [WINDOW_CLAUSE]])
partition_by_clause ::= PARTITION BY EXPR [, EXPR ...]
order_by_clause ::= ORDER BY EXPR  [ASC | DESC] [NULLS FIRST | NULLS LAST]
              [, EXPR [ASC | DESC] [NULLS FIRST | NULLS LAST] ...]
window_clause: See Window clause
```

PARTITION BY clause:

The PARTITION BY clause acts much like the GROUP BY clause in the outermost block of a query. It divides the rows into groups containing identical values in one or more columns. These logical groups are known as *partitions*. Throughout the discussion of analytic functions, “partitions” refers to the groups produced by the PARTITION BY clause, not to partitioned tables. However, note the following limitation that applies specifically to analytic function calls involving partitioned tables.

In queries involving both analytic functions and partitioned tables, partition pruning only occurs for columns named in the PARTITION BY clause of the analytic function call. For example, if an analytic function query has a clause such as WHERE year=2016, the way to make the query prune all other YEAR partitions is to include PART ITION BY year in the analytic function call; for example, OVER (PARTITION BY year,*OTHER_COLUMNS OTHER_ANALYTIC_CLAUSES*).

The sequence of results from an analytic function “resets” for each new partition in the result set. That is, the set of preceding or following rows considered by the analytic function always come from a single partition. Any MAX(), SUM(), ROW_NUMBER(), and so on apply to each partition independently. Omit the PARTITION BY clause to apply the analytic operation to all the rows in the table.

ORDER BY clause:

The ORDER BY clause works much like the ORDER BY clause in the outermost block of a query. It defines the order in which rows are evaluated for the entire input set, or for each group produced by a PARTITION BY clause. You can order by one or multiple expressions, and for each expression optionally choose ascending or descending order and whether nulls come first or last in the sort order. Because this ORDER BY clause only defines the order in which rows are evaluated, if you want the results to be output in a specific order, also include an ORDER BY clause in the outer block of the query.

When the ORDER BY clause is omitted, the analytic function applies to all items in the group produced by the PART ITION BY clause. When the ORDER BY clause is included, the analysis can apply to all or a subset of the items in the group, depending on the optional window clause.

The order in which the rows are analyzed is only defined for those columns specified in ORDER BY clauses.

One difference between the analytic and outer uses of the ORDER BY clause: inside the OVER clause, ORDER BY 1 or other integer value is interpreted as a constant sort value (effectively a no-op) rather than referring to column 1.

Window clause:

The window clause is only allowed in combination with an ORDER BY clause. If the ORDER BY clause is specified but the window clause is not, the default window is RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

HBase considerations:

Because HBase tables are optimized for single-row lookups rather than full scans, analytic functions using the OVER () clause are not recommended for HBase tables. Although such queries work, their performance is lower than on comparable tables using HDFS data files.

Parquet considerations:

Analytic functions are very efficient for Parquet tables. The data that is examined during evaluation of the OVER() clause comes from a specified set of columns, and the values for each column are arranged sequentially within each data file.

Text table considerations:

Analytic functions are convenient to use with text tables for exploratory business intelligence. When the volume of data is substantial, prefer to use Parquet tables for performance-critical analytic queries.

Examples:

The following example shows how to synthesize a numeric sequence corresponding to all the rows in a table. The new table has the same columns as the old one, plus an additional column ID containing the integers 1, 2, 3, and so on, corresponding to the order of a TIMESTAMP column in the original table.

```
CREATE TABLE events_with_id AS
  SELECT
    row_number() OVER (ORDER BY date_and_time) AS id,
    c1, c2, c3, c4
  FROM events;
```

The following example shows how to determine the number of rows containing each value for a column. Unlike a corresponding GROUP BY query, this one can analyze a single column and still return all values (not just the distinct ones) from the other columns.

```
SELECT x, y, z,
  count() OVER (PARTITION BY x) AS how_many_x
FROM t1;
```

Restrictions:

You cannot directly combine the DISTINCT operator with analytic function calls. You can put the analytic function call in a WITH clause or an inline view, and apply the DISTINCT operator to its result set.

```
WITH t1 AS (SELECT x, sum(x) OVER (PARTITION BY x) AS total FROM t1)
  SELECT DISTINCT x, total FROM t1;
```

WINDOW

Certain analytic functions accept an optional *window clause*, which makes the function analyze only certain rows “around” the current row rather than all rows in the partition. For example, you can get a moving average by specifying some number of preceding and following rows, or a running count or running total by specifying all rows up to the current position. This clause can result in different analytic results for rows within the same partition.

The window clause is supported with the AVG(), COUNT(), FIRST_VALUE(), LAST_VALUE(), and SUM() functions. For MAX() and MIN(), the window clause only allowed if the start bound is UNBOUNDED PRECED ING

Syntax:

```
ROWS BETWEEN [ { M | UNBOUNDED } PRECEDING | CURRENT ROW] [ AND [CURRENT ROW
 | { UNBOUNDED | N } FOLLOWING] ]
RANGE BETWEEN [ {M | UNBOUNDED } PRECEDING | CURRENT ROW] [ AND [CURRENT ROW
 | { UNBOUNDED | N } FOLLOWING] ]
```

ROWS BETWEEN defines the size of the window in terms of the indexes of the rows in the result set. The size of the window is predictable based on the clauses the position within the result set.

RANGE BETWEEN does not currently support numeric arguments to define a variable-size sliding window.

Currently, Impala supports only some combinations of arguments to the RANGE clause:

- RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW (the default when ORDER BY is specified and the window clause is omitted)
- RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
- RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

When RANGE is used, CURRENT ROW includes not just the current row but all rows that are tied with the current row based on the ORDER BY expressions.

Examples:

The following examples show financial data for a fictional stock symbol JDR. The closing price moves up and down each day.

```
create table stock_ticker (stock_symbol string, closing_price decimal(8,2),
closing_date timestamp);
...load some data...
select * from stock_ticker order by stock_symbol, closing_date
+--------------+---------------+---------------------+
| stock_symbol | closing_price | closing_date        |
+--------------+---------------+---------------------+
| JDR          | 12.86         | 2014-10-02 00:00:00 |
| JDR          | 12.89         | 2014-10-03 00:00:00 |
| JDR          | 12.94         | 2014-10-04 00:00:00 |
| JDR          | 12.55         | 2014-10-05 00:00:00 |
| JDR          | 14.03         | 2014-10-06 00:00:00 |
| JDR          | 14.75         | 2014-10-07 00:00:00 |
| JDR          | 13.98         | 2014-10-08 00:00:00 |
+--------------+---------------+---------------------+
```

The queries use analytic functions with window clauses to compute moving averages of the closing price. For example, ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING produces an average of the value from a 3-day span, producing a different value for each row. The first row, which has no preceding row, only gets averaged with the row following it. If the table contained more than one stock symbol, the PARTITION BY clause would limit the window for the moving average to only consider the prices for a single stock.

```
select stock_symbol, closing_date, closing_price,
  avg(closing_price) over (partition by stock_symbol order by closing_date
    rows between 1 preceding and 1 following) as moving_average
  from stock_ticker;
+--------------+---------------------+---------------+----------------+
| stock_symbol | closing_date        | closing_price | moving_average |
+--------------+---------------------+---------------+----------------+
| JDR          | 2014-10-02 00:00:00 | 12.86         | 12.87          |
| JDR          | 2014-10-03 00:00:00 | 12.89         | 12.89          |
| JDR          | 2014-10-04 00:00:00 | 12.94         | 12.79          |
| JDR          | 2014-10-05 00:00:00 | 12.55         | 13.17          |
| JDR          | 2014-10-06 00:00:00 | 14.03         | 13.77          |
| JDR          | 2014-10-07 00:00:00 | 14.75         | 14.25          |
| JDR          | 2014-10-08 00:00:00 | 13.98         | 14.36          |
+--------------+---------------------+---------------+----------------+
```

The clause ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW produces a cumulative moving average, from the earliest data up to the value for each day.

```
select stock_symbol, closing_date, closing_price,
  avg(closing_price) over (partition by stock_symbol order by closing_date
    rows between unbounded preceding and current row) as moving_average
  from stock_ticker;
+--------------+---------------------+---------------+----------------+
| stock_symbol | closing_date        | closing_price | moving_average |
+--------------+---------------------+---------------+----------------+
| JDR          | 2014-10-02 00:00:00 | 12.86         | 12.86          |
```

```
| JDR           | 2014-10-03 00:00:00 | 12.89          | 12.87          |
| JDR           | 2014-10-04 00:00:00 | 12.94          | 12.89          |
| JDR           | 2014-10-05 00:00:00 | 12.55          | 12.81          |
| JDR           | 2014-10-06 00:00:00 | 14.03          | 13.05          |
| JDR           | 2014-10-07 00:00:00 | 14.75          | 13.33          |
| JDR           | 2014-10-08 00:00:00 | 13.98          | 13.42          |
+---------------+---------------------+----------------+----------------+
```

AVG

You can include an OVER clause with a call to this function to use it as an analytic function.

Related Information

AVG function

COUNT

You can include an OVER clause with a call to this function to use it as an analytic function.

Related Information

COUNT function

CUME_DIST

Returns the cumulative distribution of a value. The value for each row in the result set is greater than 0 and less than or equal to 1.

Syntax:

```
CUME_DIST (EXPR)
  OVER ([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

Within each partition of the result set, the CUME_DIST() value represents an ascending sequence that ends at 1. Each value represents the proportion of rows in the partition whose values are less than or equal to the value in the current row.

If the sequence of input values contains ties, the CUME_DIST() results are identical for the tied values.

Impala only supports the CUME_DIST() function in an analytic context, not as a regular aggregate function.

Examples:

This example uses a table with 9 rows. The CUME_DIST() function evaluates the entire table because there is no PARTITION BY clause, with the rows ordered by the weight of the animal. the sequence of values shows that 1/9 of the values are less than or equal to the lightest animal (mouse), 2/9 of the values are less than or equal to the second-lightest animal, and so on up to the heaviest animal (elephant), where 9/9 of the rows are less than or equal to its weight.

```
create table animals (name string, kind string, kilos decimal(9,3));
insert into animals values
  ('Elephant', 'Mammal', 4000), ('Giraffe', 'Mammal', 1200), ('Mouse', 'Mamm
al', 0.020),
  ('Condor', 'Bird', 15), ('Horse', 'Mammal', 500), ('Owl', 'Bird', 2.5),
  ('Ostrich', 'Bird', 145), ('Polar bear', 'Mammal', 700), ('Housecat', '
Mammal', 5);

select name, cume_dist() over (order by kilos) from animals;
+------------+-----------------------+
| name       | cume_dist() OVER(...) |
+------------+-----------------------+
| Elephant   | 1                     |
```

```
| Giraffe    | 0.8888888888888888    |
| Polar bear | 0.7777777777777778    |
| Horse      | 0.6666666666666666    |
| Ostrich    | 0.5555555555555556    |
| Condor     | 0.4444444444444444    |
| Housecat   | 0.3333333333333333    |
| Owl        | 0.2222222222222222    |
| Mouse      | 0.1111111111111111    |
+------------+-----------------------+
```

Using a PARTITION BY clause produces a separate sequence for each partition group, in this case one for mammals and one for birds. Because there are 3 birds and 6 mammals, the sequence illustrates how 1/3 of the “Bird” rows have a kilos value that is less than or equal to the lightest bird, 1/6 of the “Mammal” rows have a kilos value that is less than or equal to the lightest mammal, and so on until both the heaviest bird and heaviest mammal have a CUME_DIS T() value of 1.

```
select name, kind, cume_dist() over (partition by kind order by kilos) from
animals
+------------+--------+-----------------------+
| name       | kind   | cume_dist() OVER(...) |
+------------+--------+-----------------------+
| Ostrich    | Bird   | 1                     |
| Condor     | Bird   | 0.6666666666666666    |
| Owl        | Bird   | 0.3333333333333333    |
| Elephant   | Mammal | 1                     |
| Giraffe    | Mammal | 0.8333333333333334    |
| Polar bear | Mammal | 0.6666666666666666    |
| Horse      | Mammal | 0.5                   |
| Housecat   | Mammal | 0.3333333333333333    |
| Mouse      | Mammal | 0.1666666666666667    |
+------------+--------+-----------------------+
```

We can reverse the ordering within each partition group by using an ORDER BY ... DESC clause within the OVER() clause. Now the lightest (smallest value of kilos) animal of each kind has a CUME_DIST() value of 1.

```
select name, kind, cume_dist() over (partition by kind order by kilos desc)
from animals
+------------+--------+-----------------------+
| name       | kind   | cume_dist() OVER(...) |
+------------+--------+-----------------------+
| Owl        | Bird   | 1                     |
| Condor     | Bird   | 0.6666666666666666    |
| Ostrich    | Bird   | 0.3333333333333333    |
| Mouse      | Mammal | 1                     |
| Housecat   | Mammal | 0.8333333333333334    |
| Horse      | Mammal | 0.6666666666666666    |
| Polar bear | Mammal | 0.5                   |
| Giraffe    | Mammal | 0.3333333333333333    |
| Elephant   | Mammal | 0.1666666666666667    |
+------------+--------+-----------------------+
```

The following example manufactures some rows with identical values in the kilos column, to demonstrate how the results look in case of tie values. For simplicity, it only shows the CUME_DIST() sequence for the “Bird” rows. Now with 3 rows all with a value of 15, all of those rows have the same CUME_DIST() value. 4/5 of the rows have a value for kilos that is less than or equal to 15.

```
insert into animals values ('California Condor', 'Bird', 15), ('Andean Condo
r', 'Bird', 15)
select name, kind, cume_dist() over (order by kilos) from animals where kind
 = 'Bird';
+-------------------+------+----------------------+
```

```
| name               | kind | cume_dist() OVER(...) |
+--------------------+------+-----------------------+
| Ostrich            | Bird | 1                     |
| Condor             | Bird | 0.8                   |
| California Condor  | Bird | 0.8                   |
| Andean Condor      | Bird | 0.8                   |
| Owl                | Bird | 0.2                   |
+--------------------+------+-----------------------+
```

The following example shows how to use an ORDER BY clause in the outer block to order the result set in case of ties. Here, all the “Bird” rows are together, then in descending order by the result of the CUME_DIST() function, and all tied CUME_DIST() values are ordered by the animal name.

```
select name, kind, cume_dist() over (partition by kind order by kilos) as or
dering
  from animals
where
  kind = 'Bird'
order by kind, ordering desc, name;
+--------------------+------+----------+
| name               | kind | ordering |
+--------------------+------+----------+
| Ostrich            | Bird | 1        |
| Andean Condor      | Bird | 0.8      |
| California Condor  | Bird | 0.8      |
| Condor             | Bird | 0.8      |
| Owl                | Bird | 0.2      |
+--------------------+------+----------+
```

DENSE_RANK

Returns an ascending sequence of integers, starting with 1. The output sequence produces duplicate integers for duplicate values of the ORDER BY expressions. After generating duplicate output values for the “tied” input values, the function continues the sequence with the next higher integer. Therefore, the sequence contains duplicates but no gaps when the input contains duplicates. Starts the sequence over for each group produced by the PARTITIONED BY clause.

Syntax:

```
DENSE_RANK() OVER([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The PARTITION BY clause is optional. The ORDER BY clause is required. The window clause is not allowed.

Usage notes:

Often used for top-N and bottom-N queries. For example, it could produce a “top 10” report including all the items with the 10 highest values, even if several items tied for 1st place.

Similar to ROW_NUMBER and RANK. These functions differ in how they treat duplicate combinations of values.

Examples:

The following example demonstrates how the DENSE_RANK() function identifies where each value “places” in the result set, producing the same result for duplicate values, but with a strict sequence from 1 to the number of groups. For example, when results are ordered by the X column, both 1 values are tied for first; both 2 values are tied for second; and so on.

```
select x, dense_rank() over(order by x) as rank, property from int_t;
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 1  | 1    | square   |
| 1  | 1    | odd      |
```

```
| 2  | 2    | even     |
| 2  | 2    | prime    |
| 3  | 3    | prime    |
| 3  | 3    | odd      |
| 4  | 4    | even     |
| 4  | 4    | square   |
| 5  | 5    | odd      |
| 5  | 5    | prime    |
| 6  | 6    | even     |
| 6  | 6    | perfect  |
| 7  | 7    | lucky    |
| 7  | 7    | lucky    |
| 7  | 7    | lucky    |
| 7  | 7    | odd      |
| 7  | 7    | prime    |
| 8  | 8    | even     |
| 9  | 9    | square   |
| 9  | 9    | odd      |
| 10 | 10   | round    |
| 10 | 10   | even     |
+----+------+----------+
```

The following examples show how the DENSE_RANK() function is affected by the PARTITION property within the ORDER BY clause.

Partitioning by the PROPERTY column groups all the even, odd, and so on values together, and DENSE_RANK() returns the place of each value within the group, producing several ascending sequences.

```
select x, dense_rank() over(partition by property order by x) as rank, prope
rty from int_t;
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 2  | 1    | even     |
| 4  | 2    | even     |
| 6  | 3    | even     |
| 8  | 4    | even     |
| 10 | 5    | even     |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 1  | 1    | odd      |
| 3  | 2    | odd      |
| 5  | 3    | odd      |
| 7  | 4    | odd      |
| 9  | 5    | odd      |
| 6  | 1    | perfect  |
| 2  | 1    | prime    |
| 3  | 2    | prime    |
| 5  | 3    | prime    |
| 7  | 4    | prime    |
| 10 | 1    | round    |
| 1  | 1    | square   |
| 4  | 2    | square   |
| 9  | 3    | square   |
+----+------+----------+
```

Partitioning by the X column groups all the duplicate numbers together and returns the place each value within the group; because each value occurs only 1 or 2 times, DENSE_RANK() designates each X value as either first or second within its group.

```
select x, dense_rank() over(partition by x order by property) as rank, prope
rty from int_t;
```

```
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 1  | 1    | odd      |
| 1  | 2    | square   |
| 2  | 1    | even     |
| 2  | 2    | prime    |
| 3  | 1    | odd      |
| 3  | 2    | prime    |
| 4  | 1    | even     |
| 4  | 2    | square   |
| 5  | 1    | odd      |
| 5  | 2    | prime    |
| 6  | 1    | even     |
| 6  | 2    | perfect  |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 2    | odd      |
| 7  | 3    | prime    |
| 8  | 1    | even     |
| 9  | 1    | odd      |
| 9  | 2    | square   |
| 10 | 1    | even     |
| 10 | 2    | round    |
+----+------+----------+
```

The following example shows how DENSE_RANK() produces a continuous sequence while still allowing for ties. In this case, Croesus and Midas both have the second largest fortune, while Crassus has the third largest. (In the *RANK function* section, you see a similar query with the RANK() function that shows that while Crassus has the third largest fortune, he is the fourth richest person.)

```
select dense_rank() over (order by net_worth desc) as placement, name, net_w
orth from wealth order by placement, name;
+-----------+---------+---------------+
| placement | name    | net_worth     |
+-----------+---------+---------------+
| 1         | Solomon | 2000000000.00 |
| 2         | Croesus | 1000000000.00 |
| 2         | Midas   | 1000000000.00 |
| 3         | Crassus | 500000000.00  |
| 4         | Scrooge | 80000000.00   |
+-----------+---------+---------------+
```

FIRST_VALUE

Returns the expression value from the first row in the window. If your table has null values, you can use the IGNORE NULLS clause to return the first non-null value from the window. This same value is repeated for all result rows for the group. The return value is NULL if the input expression is NULL.

Syntax:

```
FIRST_VALUE(EXPR [IGNORE NULLS]) OVER([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE
 [WINDOW_CLAUSE])
```

The PARTITION BY clause is optional. The ORDER BY clause is required. The window clause is optional.

Usage notes:

If any duplicate values occur in the tuples evaluated by the ORDER BY clause, the result of this function is not deterministic. Consider adding additional ORDER BY columns to ensure consistent ordering.

Examples:

The following example shows a table with a wide variety of country-appropriate greetings. For consistency, we want to standardize on a single greeting for each country. The FIRST_VALUE() function helps to produce a mail merge report where every person from the same country is addressed with the same greeting.

```
select name, country, greeting from mail_merge;
+---------+---------+--------------+
| name    | country | greeting     |
+---------+---------+--------------+
| Pete    | USA     | Hello        |
| John    | USA     | Hi           |
| Boris   | Germany | Guten tag    |
| Michael | Germany | Guten morgen |
| Bjorn   | Sweden  | Hej          |
| Mats    | Sweden  | Tja          |
+---------+---------+--------------+

select country, name,
  first_value(greeting)
    over (partition by country order by name, greeting) as greeting
  from mail_merge;
+---------+---------+-----------+
| country | name    | greeting  |
+---------+---------+-----------+
| Germany | Boris   | Guten tag |
| Germany | Michael | Guten tag |
| Sweden  | Bjorn   | Hej       |
| Sweden  | Mats    | Hej       |
| USA     | John    | Hi        |
| USA     | Pete    | Hi        |
+---------+---------+-----------+
```

Changing the order in which the names are evaluated changes which greeting is applied to each group.

```
select country, name,
  first_value(greeting)
    over (partition by country order by name desc, greeting) as greeting
  from mail_merge;
+---------+---------+--------------+
| country | name    | greeting     |
+---------+---------+--------------+
| Germany | Michael | Guten morgen |
| Germany | Boris   | Guten morgen |
| Sweden  | Mats    | Tja          |
| Sweden  | Bjorn   | Tja          |
| USA     | Pete    | Hello        |
| USA     | John    | Hello        |
+---------+---------+--------------+
```

If you introduce null values in the mail_merge table, the FIRST_VALUE() function will produce a different result with the IGNORE NULLS clause.

```
select * from mail_merge;
+---------+---------+--------------+
| name    | country | greeting     |
+---------+---------+--------------+
| Boris   | Germany | Guten tag    |
| Peng    | China   | Nihao        |
| Mats    | Sweden  | Tja          |
| Bjorn   | Sweden  | Hej          |
| Kei     | Japan   | NULL         |
| Li      | China   | NULL         |
| John    | USA     | Hi           |
```

```
| Pete    | USA     | Hello        |
| Michael | Germany | Guten morgen |
+---------+---------+--------------+

select country, name,
  first_value(greeting ignore nulls)
    over (partition by country order by name,greeting) as greeting
  from mail_merge;
+---------+---------+-----------+
| country | name    | greeting  |
+---------+---------+-----------+
| Japan   | Kei     | NULL      |
| Germany | Boris   | Guten tag |
| Germany | Michael | Guten tag |
| China   | Li      | NULL      |
| China   | Peng    | Nihao     |
| Sweden  | Bjorn   | Hej       |
| Sweden  | Mats    | Hej       |
| USA     | John    | Hi        |
| USA     | Pete    | Hi        |
+---------+---------+-----------+
```

Changing the order in which the names are evaluated changes the result because null values are now encountered in a different order.

```
select country, name,
  first_value(greeting ignore nulls)
    over (partition by country order by name desc, greeting) as greeting
  from mail_merge
+---------+---------+--------------+
| country | name    | greeting     |
+---------+---------+--------------+
| Japan   | Kei     | NULL         |
| China   | Peng    | Nihao        |
| China   | Li      | Nihao        |
| Sweden  | Mats    | Tja          |
| Sweden  | Bjorn   | Tja          |
| USA     | Pete    | Hello        |
| USA     | John    | Hello        |
| Germany | Michael | Guten morgen |
| Germany | Boris   | Guten morgen |
+---------+---------+--------------+
```

LAG

This function returns the value of an expression using column values from a preceding row. You specify an integer offset, which designates a row position some number of rows previous to the current row. Any column references in the expression argument refer to column values from that prior row. Typically, the table contains a time sequence or numeric sequence column that clearly distinguishes the ordering of the rows.

Syntax:

```
LAG (EXPR [, OFFSET] [, DEFAULT])
  OVER ([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

Sometimes used an an alternative to doing a self-join.

Examples:

The following example uses the same stock data created in the *Window clause* section. For each day, the query prints the closing price alongside the previous day's closing price. The first row for each stock symbol has no previous row, so that LAG() value is NULL.

```
select stock_symbol, closing_date, closing_price,
    lag(closing_price,1) over (partition by stock_symbol order by closing_
date) as "yesterday closing"
  from stock_ticker
    order by closing_date;
+--------------+---------------------+---------------+-------------------+
| stock_symbol | closing_date        | closing_price | yesterday closing |
+--------------+---------------------+---------------+-------------------+
| JDR          | 2014-09-13 00:00:00 | 12.86         | NULL              |
| JDR          | 2014-09-14 00:00:00 | 12.89         | 12.86             |
| JDR          | 2014-09-15 00:00:00 | 12.94         | 12.89             |
| JDR          | 2014-09-16 00:00:00 | 12.55         | 12.94             |
| JDR          | 2014-09-17 00:00:00 | 14.03         | 12.55             |
| JDR          | 2014-09-18 00:00:00 | 14.75         | 14.03             |
| JDR          | 2014-09-19 00:00:00 | 13.98         | 14.75             |
+--------------+---------------------+---------------+-------------------+
```

The following example does an arithmetic operation between the current row and a value from the previous row, to produce a delta value for each day. This example also demonstrates how ORDER BY works independently in the different parts of the query. The ORDER BY closing_date in the OVER clause makes the query analyze the rows in chronological order. Then the outer query block uses ORDER BY closing_date DESC to present the results with the most recent date first.

```
select stock_symbol, closing_date, closing_price,
    cast(
      closing_price - lag(closing_price,1) over
        (partition by stock_symbol order by closing_date)
      as decimal(8,2)
    )
    as "change from yesterday"
  from stock_ticker
    order by closing_date desc;
+--------------+---------------------+---------------+---------------------
--+
| stock_symbol | closing_date        | closing_price | change from yesterday
 |
+--------------+---------------------+---------------+-------------------
----+
| JDR          | 2014-09-19 00:00:00 | 13.98         | -0.76
 |
| JDR          | 2014-09-18 00:00:00 | 14.75         | 0.72
 |
| JDR          | 2014-09-17 00:00:00 | 14.03         | 1.47
 |
| JDR          | 2014-09-16 00:00:00 | 12.55         | -0.38
  |
| JDR          | 2014-09-15 00:00:00 | 12.94         | 0.04
 |
| JDR          | 2014-09-14 00:00:00 | 12.89         | 0.03
 |
| JDR          | 2014-09-13 00:00:00 | 12.86         | NULL
 |
+--------------+---------------------+---------------+-------------------
----+
```

Related information:

This function is the converse of the LEAD function.

LAST_VALUE

Returns the expression value from the last row in the window. If your table has null values, you can use the IGNORE NULLS clause to return the last non-null value from the window. This same value is repeated for all result rows for the group. The return value is NULL if the input expression is NULL.

Syntax:

```
LAST_VALUE(EXPR [IGNORE NULLS]) OVER([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE
 [WINDOW_CLAUSE])
```

The PARTITION BY clause is optional. The ORDER BY clause is required. The window clause is optional.

Usage notes:

If any duplicate values occur in the tuples evaluated by the ORDER BY clause, the result of this function is not deterministic. Consider adding additional ORDER BY columns to ensure consistent ordering.

Examples:

The following example uses the same MAIL_MERGE table as in the example for the FIRST_VALUE function. Because the default window when ORDER BY is used is BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, the query requires the UNBOUNDED FOLLOWING to look ahead to subsequent rows and find the last value for each country.

```
select country, name,
  last_value(greeting) over (
    partition by country order by name, greeting
    rows between unbounded preceding and unbounded following
  ) as greeting
  from mail_merge
+---------+---------+--------------+
| country | name    | greeting     |
+---------+---------+--------------+
| Germany | Boris   | Guten morgen |
| Germany | Michael | Guten morgen |
| Sweden  | Bjorn   | Tja          |
| Sweden  | Mats    | Tja          |
| USA     | John    | Hello        |
| USA     | Pete    | Hello        |
+---------+---------+--------------+
```

Introducing null values into the MAIL_MERGE table as in the example for the FIRST VALUE function, the result set changes when you use the IGNORE NULLS clause.

```
select * from mail_merge;
+---------+---------+--------------+
| name    | country | greeting     |
+---------+---------+--------------+
| Kei     | Japan   | NULL         |
| Boris   | Germany | Guten tag    |
| Li      | China   | NULL         |
| Michael | Germany | Guten morgen |
| Bjorn   | Sweden  | Hej          |
| Peng    | China   | Nihao        |
| Pete    | USA     | Hello        |
| Mats    | Sweden  | Tja          |
| John    | USA     | Hi           |
+---------+---------+--------------+

select country, name,
  last_value(greeting ignore nulls) over (
    partition by country order by name, greeting
    rows between unbounded preceding and unbounded following
```

```
  ) as greeting
  from mail_merge;
+---------+---------+--------------+
| country | name    | greeting     |
+---------+---------+--------------+
| Japan   | Kei     | NULL         |
| Germany | Boris   | Guten morgen |
| Germany | Michael | Guten morgen |
| China   | Li      | Nihao        |
| China   | Peng    | Nihao        |
| Sweden  | Bjorn   | Tja          |
| Sweden  | Mats    | Tja          |
| USA     | John    | Hello        |
| USA     | Pete    | Hello        |
+---------+---------+--------------+
```

LEAD

This function returns the value of an expression using column values from a following row. You specify an integer offset, which designates a row position some number of rows after to the current row. Any column references in the expression argument refer to column values from that later row. Typically, the table contains a time sequence or numeric sequence column that clearly distinguishes the ordering of the rows.

Syntax:

```
LEAD (EXPR [, OFFSET] [, DEFAULT])
  OVER ([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

Sometimes used an an alternative to doing a self-join.

Examples:

The following example uses the same stock data created in the Window clause. The query analyzes the closing price for a stock symbol, and for each day evaluates if the closing price for the following day is higher or lower.

```
select stock_symbol, closing_date, closing_price,
  case
    (lead(closing_price,1)
      over (partition by stock_symbol order by closing_date)
        - closing_price) > 0
    when true then "higher"
    when false then "flat or lower"
  end as "trending"
from stock_ticker
  order by closing_date;
+--------------+---------------------+---------------+---------------+
| stock_symbol | closing_date        | closing_price | trending      |
+--------------+---------------------+---------------+---------------+
| JDR          | 2014-09-13 00:00:00 | 12.86         | higher        |
| JDR          | 2014-09-14 00:00:00 | 12.89         | higher        |
| JDR          | 2014-09-15 00:00:00 | 12.94         | flat or lower |
| JDR          | 2014-09-16 00:00:00 | 12.55         | higher        |
| JDR          | 2014-09-17 00:00:00 | 14.03         | higher        |
| JDR          | 2014-09-18 00:00:00 | 14.75         | flat or lower |
| JDR          | 2014-09-19 00:00:00 | 13.98         | NULL          |
+--------------+---------------------+---------------+---------------+
```

Related information:

This function is the converse of the LAG function.

MAX

You can include an OVER clause with a call to this function to use it as an analytic function.

Related Information

MAX function

MIN

You can include an OVER clause with a call to this function to use it as an analytic function.

Related Information

MIN function

NTILE

Returns the “bucket number” associated with each row, between 1 and the value of an expression. For example, creating 100 buckets puts the lowest 1% of values in the first bucket, while creating 10 buckets puts the lowest 10% of values in the first bucket. Each partition can have a different number of buckets.

Syntax:

```
NTILE (EXPR [, OFFSET ...]
  OVER ([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

The “ntile” name is derived from the practice of dividing result sets into fourths (quartile), tenths (decile), and so on. The NTILE() function divides the result set based on an arbitrary percentile value.

The number of buckets must be a positive integer.

The number of items in each bucket is identical or almost so, varying by at most 1. If the number of items does not divide evenly between the buckets, the remaining N items are divided evenly among the first N buckets.

If the number of buckets N is greater than the number of input rows in the partition, then the first N buckets each contain one item, and the remaining buckets are empty.

Examples:

The following example shows divides groups of animals into 4 buckets based on their weight. The ORDER BY ... DESC clause in the OVER() clause means that the heaviest 25% are in the first group, and the lightest 25% are in the fourth group. (The ORDER BY in the outermost part of the query shows how you can order the final result set independently from the order in which the rows are evaluated by the OVER() clause.) Because there are 9 rows in the group, divided into 4 buckets, the first bucket receives the extra item.

```
create table animals (name string, kind string, kilos decimal(9,3));

insert into animals values
  ('Elephant', 'Mammal', 4000), ('Giraffe', 'Mammal', 1200), ('Mouse', 'Mam
mal', 0.020),
  ('Condor', 'Bird', 15), ('Horse', 'Mammal', 500), ('Owl', 'Bird', 2.5),
  ('Ostrich', 'Bird', 145), ('Polar bear', 'Mammal', 700), ('Housecat', 'Mam
mal', 5);

select name, ntile(4) over (order by kilos desc) as quarter
  from animals
order by quarter desc;
+------------+---------+
| name       | quarter |
+------------+---------+
| Owl        | 4       |
| Mouse      | 4       |
| Condor     | 3       |
```

```
| Housecat   | 3       |
| Horse      | 2       |
| Ostrich    | 2       |
| Elephant   | 1       |
| Giraffe    | 1       |
| Polar bear | 1       |
+------------+---------+
```

The following examples show how the PARTITION clause works for the NTILE() function. Here, we divide each kind of animal (mammal or bird) into 2 buckets, the heavier half and the lighter half.

```
select name, kind, ntile(2) over (partition by kind order by kilos desc) as
half
  from animals
order by kind;
+------------+--------+------+
| name       | kind   | half |
+------------+--------+------+
| Ostrich    | Bird   | 1    |
| Condor     | Bird   | 1    |
| Owl        | Bird   | 2    |
| Elephant   | Mammal | 1    |
| Giraffe    | Mammal | 1    |
| Polar bear | Mammal | 1    |
| Horse      | Mammal | 2    |
| Housecat   | Mammal | 2    |
| Mouse      | Mammal | 2    |
+------------+--------+------+
```

Again, the result set can be ordered independently from the analytic evaluation. This next example lists all the animals heaviest to lightest, showing that elephant and giraffe are in the “top half” of mammals by weight, while housecat and mouse are in the “bottom half”.

```
select name, kind, ntile(2) over (partition by kind order by kilos desc) as
half
  from animals
order by kilos desc;
+------------+--------+------+
| name       | kind   | half |
+------------+--------+------+
| Elephant   | Mammal | 1    |
| Giraffe    | Mammal | 1    |
| Polar bear | Mammal | 1    |
| Horse      | Mammal | 2    |
| Ostrich    | Bird   | 1    |
| Condor     | Bird   | 1    |
| Housecat   | Mammal | 2    |
| Owl        | Bird   | 2    |
| Mouse      | Mammal | 2    |
+------------+--------+------+
```

PERCENT_RANK

The PERCENT_RANK() is a window function that calculates the percentile ranking of rows in a result set.

Syntax:

```
PERCENT_RANK (EXPR)
  OVER ([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

Calculates the rank, expressed as a percentage, of each row within a group of rows. If rank is the value for that same row from the RANK() function (from 1 to the total number of rows in the partition group), then the PERCENT_

RANK() value is calculated as (*RANK* - 1) / (*ROWS_IN_GROUP* - 1) . If there is only a single item in the partition group, its PERCENT_RANK() value is 0.

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

This function is similar to the RANK and CUME_DIST() functions: it returns an ascending sequence representing the position of each row within the rows of the same partition group. The actual numeric sequence is calculated differently, and the handling of duplicate (tied) values is different.

The return values range from 0 to 1 inclusive. The first row in each partition group always has the value 0. A NULL value is considered the lowest possible value. In the case of duplicate input values, all the corresponding rows in the result set have an identical value: the lowest PERCENT_RANK() value of those tied rows. (In contrast to CUME_DIST(), where all tied rows have the highest CUME_DIST() value.)

Examples:

The following example uses the same ANIMALS table as the examples for CUME_DIST() and NTILE(), with a few additional rows to illustrate the results where some values are NULL or there is only a single row in a partition group.

```
insert into animals values ('Komodo dragon', 'Reptile', 70);
insert into animals values ('Unicorn', 'Mythical', NULL);
insert into animals values ('Fire-breathing dragon', 'Mythical', NULL);
```

As with CUME_DIST(), there is an ascending sequence for each kind of animal. For example, the “Birds” and “Mammals” rows each have a PERCENT_RANK() sequence that ranges from 0 to 1. The “Reptile” row has a PERCENT_RANK() of 0 because that partition group contains only a single item. Both “Mythical” animals have a PERCENT_RANK() of 0 because a NULL is considered the lowest value within its partition group.

```
select name, kind, percent_rank() over (partition by kind order by kilos) fr
om animals;
+-----------------------+----------+--------------------------+
| name                  | kind     | percent_rank() OVER(...) |
+-----------------------+----------+--------------------------+
| Mouse                 | Mammal   | 0                        |
| Housecat              | Mammal   | 0.2                      |
| Horse                 | Mammal   | 0.4                      |
| Polar bear            | Mammal   | 0.6                      |
| Giraffe               | Mammal   | 0.8                      |
| Elephant              | Mammal   | 1                        |
| Komodo dragon         | Reptile  | 0                        |
| Owl                   | Bird     | 0                        |
| California Condor     | Bird     | 0.25                     |
| Andean Condor         | Bird     | 0.25                     |
| Condor                | Bird     | 0.25                     |
| Ostrich               | Bird     | 1                        |
| Fire-breathing dragon | Mythical | 0                        |
| Unicorn               | Mythical | 0                        |
+-----------------------+----------+--------------------------+
```

RANK

Returns an ascending sequence of integers, starting with 1. The output sequence produces duplicate integers for duplicate values of the ORDER BY expressions. After generating duplicate output values for the “tied” input values, the function increments the sequence by the number of tied values. Therefore, the sequence contains both duplicates and gaps when the input contains duplicates. Starts the sequence over for each group produced by the PARTITIONED BY clause.

Syntax:

```
RANK() OVER([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The PARTITION BY clause is optional. The ORDER BY clause is required. The window clause is not allowed.

Usage notes:

Often used for top-N and bottom-N queries. For example, it could produce a “top 10” report including several items that were tied for 10th place.

Similar to ROW_NUMBER and DENSE_RANK. These functions differ in how they treat duplicate combinations of values.

Examples:

The following example demonstrates how the RANK() function identifies where each value “places” in the result set, producing the same result for duplicate values, and skipping values in the sequence to account for the number of duplicates. For example, when results are ordered by the X column, both 1 values are tied for first; both 2 values are tied for third; and so on.

```
select x, rank() over(order by x) as rank, property from int_t;
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 1  | 1    | square   |
| 1  | 1    | odd      |
| 2  | 3    | even     |
| 2  | 3    | prime    |
| 3  | 5    | prime    |
| 3  | 5    | odd      |
| 4  | 7    | even     |
| 4  | 7    | square   |
| 5  | 9    | odd      |
| 5  | 9    | prime    |
| 6  | 11   | even     |
| 6  | 11   | perfect  |
| 7  | 13   | lucky    |
| 7  | 13   | lucky    |
| 7  | 13   | lucky    |
| 7  | 13   | odd      |
| 7  | 13   | prime    |
| 8  | 18   | even     |
| 9  | 19   | square   |
| 9  | 19   | odd      |
| 10 | 21   | round    |
| 10 | 21   | even     |
+----+------+----------+
```

The following examples show how the RANK() function is affected by the PARTITION property within the ORDER BY clause.

Partitioning by the PROPERTY column groups all the even, odd, and so on values together, and RANK() returns the place of each value within the group, producing several ascending sequences.

```
select x, rank() over(partition by property order by x) as rank, property fr
om int_t;
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 2  | 1    | even     |
| 4  | 2    | even     |
| 6  | 3    | even     |
| 8  | 4    | even     |
| 10 | 5    | even     |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
```

```
| 1  | 1    | odd      |
| 3  | 2    | odd      |
| 5  | 3    | odd      |
| 7  | 4    | odd      |
| 9  | 5    | odd      |
| 6  | 1    | perfect  |
| 2  | 1    | prime    |
| 3  | 2    | prime    |
| 5  | 3    | prime    |
| 7  | 4    | prime    |
| 10 | 1    | round    |
| 1  | 1    | square   |
| 4  | 2    | square   |
| 9  | 3    | square   |
+----+------+----------+
```

Partitioning by the X column groups all the duplicate numbers together and returns the place each value within the group; because each value occurs only 1 or 2 times, RANK() designates each X value as either first or second within its group.

```
select x, rank() over(partition by x order by property) as rank, property fr
om int_t;
+----+------+----------+
| x  | rank | property |
+----+------+----------+
| 1  | 1    | odd      |
| 1  | 2    | square   |
| 2  | 1    | even     |
| 2  | 2    | prime    |
| 3  | 1    | odd      |
| 3  | 2    | prime    |
| 4  | 1    | even     |
| 4  | 2    | square   |
| 5  | 1    | odd      |
| 5  | 2    | prime    |
| 6  | 1    | even     |
| 6  | 2    | perfect  |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 1    | lucky    |
| 7  | 4    | odd      |
| 7  | 5    | prime    |
| 8  | 1    | even     |
| 9  | 1    | odd      |
| 9  | 2    | square   |
| 10 | 1    | even     |
| 10 | 2    | round    |
+----+------+----------+
```

The following example shows how a magazine might prepare a list of history's wealthiest people. Croesus and Midas are tied for second, then Crassus is fourth.

```
select rank() over (order by net_worth desc) as rank, name, net_worth from w
ealth order by rank, name;
+------+---------+---------------+
| rank | name    | net_worth     |
+------+---------+---------------+
| 1    | Solomon | 2000000000.00 |
| 2    | Croesus | 1000000000.00 |
| 2    | Midas   | 1000000000.00 |
| 4    | Crassus | 500000000.00  |
| 5    | Scrooge | 80000000.00   |
```

```
+------+---------+---------------+
```

ROW_NUMBER

Returns an ascending sequence of integers, starting with 1. Starts the sequence over for each group produced by the PARTITIONED BY clause. The output sequence includes different values for duplicate input values. Therefore, the sequence never contains any duplicates or gaps, regardless of duplicate input values.

Syntax:

```
ROW_NUMBER() OVER([PARTITION_BY_CLAUSE] ORDER_BY_CLAUSE)
```

The ORDER BY clause is required. The PARTITION BY clause is optional. The window clause is not allowed.

Usage notes:

Often used for top-N and bottom-N queries where the input values are known to be unique, or precisely N rows are needed regardless of duplicate values.

Because its result value is different for each row in the result set (when used without a PARTITION BY clause), ROW_NUMBER() can be used to synthesize unique numeric ID values, for example for result sets involving unique values or tuples.

Similar to RANK and DENSE_RANK. These functions differ in how they treat duplicate combinations of values.

Examples:

The following example demonstrates how ROW_NUMBER() produces a continuous numeric sequence, even though some values of X are repeated.

```
select x, row_number() over(order by x, property) as row_number, property fr
om int_t;
+----+------------+----------+
| x  | row_number | property |
+----+------------+----------+
| 1  | 1          | odd      |
| 1  | 2          | square   |
| 2  | 3          | even     |
| 2  | 4          | prime    |
| 3  | 5          | odd      |
| 3  | 6          | prime    |
| 4  | 7          | even     |
| 4  | 8          | square   |
| 5  | 9          | odd      |
| 5  | 10         | prime    |
| 6  | 11         | even     |
| 6  | 12         | perfect  |
| 7  | 13         | lucky    |
| 7  | 14         | lucky    |
| 7  | 15         | lucky    |
| 7  | 16         | odd      |
| 7  | 17         | prime    |
| 8  | 18         | even     |
| 9  | 19         | odd      |
| 9  | 20         | square   |
| 10 | 21         | even     |
| 10 | 22         | round    |
+----+------------+----------+
```

The following example shows how a financial institution might assign customer IDs to some of history's wealthiest figures. Although two of the people have identical net worth figures, unique IDs are required for this purpose. ROW_NUMBER() produces a sequence of five different values for the five input rows.

```
select row_number() over (order by net_worth desc) as account_id, name, net_
worth
  from wealth order by account_id, name;
+------------+---------+---------------+
| account_id | name    | net_worth     |
+------------+---------+---------------+
| 1          | Solomon | 2000000000.00 |
| 2          | Croesus | 1000000000.00 |
| 3          | Midas   | 1000000000.00 |
| 4          | Crassus | 500000000.00  |
| 5          | Scrooge | 80000000.00   |
+------------+---------+---------------+
```

SUM

You can include an OVER clause with a call to this function to use it as an analytic function.

Related Information

SUM function

Creating an Impala user-defined function

User-defined functions (frequently abbreviated as UDFs) let you code your own application logic for processing column values during an Impala query. For example, a UDF could perform calculations using an external math library, combine several column values into one, do geospatial calculations, or other kinds of tests and transformations that are outside the scope of the built-in SQL operators and functions.

You can use UDFs to simplify query logic when producing reports, or to transform data in flexible ways when copying from one table to another with the INSERT ... SELECT syntax.

You might be familiar with this feature from other database products, under names such as stored functions or stored routines.

Impala support for UDFs is available in Impala 1.2 and higher:

- In Impala 1.1, using UDFs in a query required using the Hive shell. (Because Impala and Hive share the same metastore database, you could switch to Hive to run just those queries requiring UDFs, then switch back to Impala.)
- Starting in Impala 1.2, Impala can run both high-performance native code UDFs written in C++, and Java-based Hive UDFs that you might already have written.
- Impala can run scalar UDFs that return a single value for each row of the result set, and user-defined aggregate functions (UDAFs) that return a value based on a set of rows. Currently, Impala does not support user-defined table functions (UDTFs) or window functions.

Related Information

Sample UDF and UDAs for Impala

Memoization

UDF concepts

Depending on your use case, you might write all-new functions, reuse Java UDFs that you have already written for Hive, or port Hive Java UDF code to higher-performance native Impala UDFs in C++. You can code either scalar

functions for producing results one row at a time, or more complex aggregate functions for doing analysis across. The following sections discuss these different aspects of working with UDFs.

UDFs and UDAFs

Depending on your use case, the user-defined functions (UDFs) you write might accept or produce different numbers of input and output values:

- The most general kind of user-defined function (the one typically referred to by the abbreviation UDF) takes a single input value and produces a single output value. When used in a query, it is called once for each row in the result set. For example:

```
select customer_name, is_frequent_customer(customer_id) from
 customers;
select obfuscate(sensitive_column) from sensitive_data;
```

- A user-defined aggregate function (UDAF) accepts a group of values and returns a single value. You use UDAFs to summarize and condense sets of rows, in the same style as the built-in COUNT, MAX(), SUM(), and AVG() functions. When called in a query that uses the GROU P BY clause, the function is called once for each combination of GROUP BY values. For example:

```
-- Evaluates multiple rows but returns a single value.
select closest_restaurant(latitude, longitude) from places;

-- Evaluates batches of rows and returns a separate value for
 each batch.
select most_profitable_location(store_id, sales, expenses, tax
_rate, depreciation) from franchise_data group by year;
```

- Currently, Impala does not support other categories of user-defined functions, such as user-defined table functions (UDTFs) or window functions.

Native Impala UDFs

Impala supports UDFs written in C++, in addition to supporting existing Hive UDFs written in Java. Cloudera recommends using C++ UDFs because the compiled native code can yield higher performance, with UDF execution time often 10x faster for a C++ UDF than the equivalent Java UDF.

Using Hive UDFs with Impala

Impala can run Java-based user-defined functions (UDFs), originally written for Hive, with no changes, subject to the following conditions:

- The parameters and return value must all use scalar data types supported by Impala. For example, complex or nested types are not supported.
- Hive/Java UDFs must extend org.apache.hadoop.hive.ql.exec.UDF class.
- Currently, Hive UDFs that accept or return the TIMESTAMP type are not supported.
- Prior to Impala 2.5 the return type must be a “Writable” type such as Text or IntWritable, rather than a Java primitive type such as String or int. Otherwise, the UDF returns NULL. In Impala 2.5 and higher, this restriction is lifted, and both UDF arguments and return values can be Java primitive types.
- Hive UDAFs and UDTFs are not supported.
- Typically, a Java UDF will run several times slower in Impala than the equivalent native UDF written in C++.
- In Impala 2.5 and higher, you can transparently call Hive Java UDFs through Impala, or call Impala Java UDFs through Hive. This feature does not apply to built-in Hive functions. Any Impala Java UDFs created with older versions must be re-created using new CREATE FUNCT ION syntax, without any signature for arguments or the return value.

To take full advantage of the Impala architecture and performance features, you can also write Impala-specific UDFs in C++.

For background about Java-based Hive UDFs, see the Hive documentation for UDF. For examples or tutorials for writing such UDFs, search the web for related blog posts.

The ideal way to understand how to reuse Java-based UDFs (originally written for Hive) with Impala is to take some of the Hive built-in functions (implemented as Java UDFs) and take the applicable JAR files through the UDF deployment process for Impala, creating new UDFs with different names:

1. Take a copy of the Hive JAR file containing the Hive built-in functions.
2. Use jar tf *JAR_FILE* to see a list of the classes inside the JAR. You will see names like org/apache/hadoop/hive/ql/udf/UDFLower.class and org/apache/hadoop/hive/ql/udf/UDFOPNegative.class. Make a note of the names of the functions you want to experiment with. When you specify the entry points for the Impala CREATE FUNCTION statement, change the slash characters to dots and strip off the .class suffix, for example org.apache.hadoop.hive.ql.udf.UDFLower and org.apache.hadoop.hive.ql.udf.UDFOPNegative.
3. Copy that file to an HDFS location that Impala can read. (In the examples here, we renamed the file to hive-builtins.jar in HDFS for simplicity.)
4. For each Java-based UDF that you want to call through Impala, issue a CREATE FUNCTION statement, with a LOCATION clause containing the full HDFS path of the JAR file, and a SYMBOL clause with the fully qualified name of the class, using dots as separators and without the .class extension. Remember that user-defined functions are associated with a particular database, so issue a USE statement for the appropriate database first, or specify the SQL function name as *DB_NAME.FUNCTION_NAME*. Use completely new names for the SQL functions, because Impala UDFs cannot have the same name as Impala built-in functions.
5. Call the function from your queries, passing arguments of the correct type to match the function signature. These arguments could be references to columns, arithmetic or other kinds of expressions, the results of CAST functions to ensure correct data types, and so on.

Note:

In Impala 2.9 and higher, you can refresh the user-defined functions (UDFs) that Impala recognizes, at the database level, by running the REFRESH FUNCTIONS statement with the database name as an argument. Java-based UDFs can be added to the metastore database through Hive CREATE FUNCTION statements, and made visible to Impala by subsequently running REFRESH FUNCTIONS. For example:

```
CREATE DATABASE shared_udfs;
USE shared_udfs;
...use CREATE FUNCTION statements in Hive to create so
me Java-based UDFs
   that Impala is not initially aware of...
REFRESH FUNCTIONS shared_udfs;
SELECT udf_created_by_hive(c1) FROM ...
```

Java UDF example: Reusing lower() function

For example, the following `impala-shell` session creates an Impala UDF my_lower() that reuses the Java code for the Hive lower(): built-in function. We cannot call it lower() because Impala does not allow UDFs to have the same name as built-in functions. From SQL, we call the function in a basic way (in a query with no WHERE clause), directly on a column, and on the results of a string expression:

```
[localhost:21000] > create database udfs;
[localhost:21000] > use udfs;
localhost:21000] > create function lower(string) returns string
location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.hi
ve.ql.udf.UDFLower';
```

```
ERROR: AnalysisException: Function cannot have the same name as a
 builtin: lower
[localhost:21000] > create function my_lower(string) returns s
tring location '/user/hive/udfs/hive.jar' symbol='org.apache.had
oop.hive.ql.udf.UDFLower';
[localhost:21000] > select my_lower('Some String NOT ALREADY LOWE
RCASE');
+----------------------------------------------------+
| udfs.my_lower('some string not already lowercase') |
+----------------------------------------------------+
| some string not already lowercase                  |
+----------------------------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > create table t2 (s string);
[localhost:21000] > insert into t2 values ('lower'),('UPPER'),('I
nit cap'),('CamelCase');
Inserted 4 rows in 2.28s
[localhost:21000] > select * from t2;
+-----------+
| s         |
+-----------+
| lower     |
| UPPER     |
| Init cap  |
| CamelCase |
+-----------+
Returned 4 row(s) in 0.47s
[localhost:21000] > select my_lower(s) from t2;
+------------------+
| udfs.my_lower(s) |
+------------------+
| lower            |
| upper            |
| init cap         |
| camelcase        |
+------------------+
Returned 4 row(s) in 0.54s
[localhost:21000] > select my_lower(concat('ABC ',s,' XYZ')) f
rom t2;
+-----------------------------------------+
| udfs.my_lower(concat('abc ', s, ' xyz')) |
+-----------------------------------------+
| abc lower xyz                           |
| abc upper xyz                           |
| abc init cap xyz                        |
| abc camelcase xyz                       |
+-----------------------------------------+
Returned 4 row(s) in 0.22s
```

Java UDF example: Reusing negative() function

Here is an example that reuses the Hive Java code for the negative() built-in function. This example demonstrates how the data types of the arguments must match precisely with the function signature. At first, we create an Impala SQL function that can only accept an integer argument. Impala cannot find a matching function when the query passes a floating-point argument, although we can call the integer version of the function by casting the argument. Then we overload the same function name to also accept a floating-point argument.

```
[localhost:21000] > create table t (x int);
[localhost:21000] > insert into t values (1), (2), (4), (100);
Inserted 4 rows in 1.43s
```

```
[localhost:21000] > create function my_neg(bigint) returns bigin
t location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.
hive.ql.udf.UDFOPNegative';
[localhost:21000] > select my_neg(4);
+----------------+
| udfs.my_neg(4) |
+----------------+
| -4             |
+----------------+
[localhost:21000] > select my_neg(x) from t;
+----------------+
| udfs.my_neg(x) |
+----------------+
| -2             |
| -4             |
| -100           |
+----------------+
Returned 3 row(s) in 0.60s
[localhost:21000] > select my_neg(4.0);
ERROR: AnalysisException: No matching function with signature:
 udfs.my_neg(FLOAT).
[localhost:21000] > select my_neg(cast(4.0 as int));
+--------------------------------+
| udfs.my_neg(cast(4.0 as int)) |
+--------------------------------+
| -4                             |
+--------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > create function my_neg(double) returns double
 location '/user/hive/udfs/hive.jar' symbol='org.apache.hadoop.h
ive.ql.udf.UDFOPNegative';
[localhost:21000] > select my_neg(4.0);
+------------------+
| udfs.my_neg(4.0) |
+------------------+
| -4               |
+------------------+
Returned 1 row(s) in 0.11s
```

You can find the sample files mentioned here in the Impala github repo.

Runtime environment for UDFs

By default, Impala copies UDFs into /tmp, and you can configure this location through the --local_library_dir startup flag for the `impalad` daemon.

Installing the UDF development package

To develop UDFs for Impala, download and install the impala-udf-devel package (RHEL-based distributions). This package contains header files, sample source, and build configuration files.

- In Cloudera Archive (at https://archive.cloudera.com/cdh7/), locate the appropriate package installation file for your operating system version.

  Access to Cloudera Runtime packages for production purposes requires authentication.

Note: The UDF development code does not rely on Impala being installed on the same machine. You can write and compile UDFs on a minimal development system, then deploy them on a different one for use with Impala. If you develop UDFs on a server managed by Cloudera Manager through the parcel mechanism, you still install the UDF development kit through the package mechanism; this small standalone package does not interfere with the parcels containing the main Impala code.

When you are ready to start writing your own UDFs, download the sample code and build scripts from the Cloudera sample UDF github. Then see Writing UDFs on page 449 for how to code UDFs, and Examples of creating and using UDFs on page 455 for how to build and run UDFs.

Writing UDFs

Before starting UDF development, make sure to install the development package and download the UDF code samples.

When writing UDFs:

- Keep in mind the data type differences as you transfer values from the high-level SQL to your lower-level UDF code. For example, in the UDF code you might be much more aware of how many bytes different kinds of integers require.
- Use best practices for function-oriented programming: choose arguments carefully, avoid side effects, make each function do a single thing, and so on.

Getting started with UDF coding

To understand the layout and member variables and functions of the predefined UDF data types, examine the header file /usr/include/impala_udf/udf.h:

```
// This is the only Impala header required to develop UDFs and U
DAs. This header
// contains the types that need to be used and the FunctionCont
ext object. The context
// object serves as the interface object between the UDF/UDA and
 the impala process.
```

For the basic declarations needed to write a scalar UDF, see the header file udf-sample.h within the sample build environment, which defines a simple function named AddUdf():

```
#ifndef IMPALA_UDF_SAMPLE_UDF_H
#define IMPALA_UDF_SAMPLE_UDF_H
#include <impala_udf/udf.h>

using namespace impala_udf;

IntVal AddUdf(FunctionContext* context, const IntVal& arg1, const
 IntVal& arg2);
#endif
```

For sample C++ code for a simple function named AddUdf(), see the source file udf-sample.cc within the sample build environment:

```
#include "udf-sample.h"
// In this sample we are declaring a UDF that adds two ints and
 returns an int.
IntVal AddUdf(FunctionContext* context, const IntVal& arg1, const
 IntVal& arg2) {
  if (arg1.is_null || arg2.is_null) return IntVal::null();
  return IntVal(arg1.val + arg2.val);
}
```

```
// Multiple UDFs can be defined in the same file
```

Data types for function arguments and return values

Each value that a user-defined function can accept as an argument or return as a result value must map to a SQL data type that you could specify for a table column.

Currently, Impala UDFs cannot accept arguments or return values of the Impala complex types (STRUCT, ARRAY, or MAP).

Each data type has a corresponding structure defined in the C++ and Java header files, with two member fields and some predefined comparison operators and constructors:

- is_null indicates whether the value is NULL or not. val holds the actual argument or return value when it is non-NULL.
- Each struct also defines a null() member function that constructs an instance of the struct with the is_null flag set.
- The built-in SQL comparison operators and clauses such as <, >=, BETWEEN, and ORDER BY all work automatically based on the SQL return type of each UDF. For example, Impala knows how to evaluate BETWEEN 1 AND udf_returning_int(col1) or ORDER BY udf_returning_string(col2) without you declaring any comparison operators within the UDF itself.

  For convenience within your UDF code, each struct defines == and != operators for comparisons with other structs of the same type. These are for typical C++ comparisons within your own code, not necessarily reproducing SQL semantics. For example, if the is_null flag is set in both structs, they compare as equal. That behavior of null comparisons is different from SQL (where NULL == NULL is NULL rather than true), but more in line with typical C++ behavior.
- Each kind of struct has one or more constructors that define a filled-in instance of the struct, optionally with default values.
- Impala cannot process UDFs that accept the composite or nested types as arguments or return them as result values. This limitation applies both to Impala UDFs written in C++ and Java-based Hive UDFs.
- You can overload functions by creating multiple functions with the same SQL name but different argument types. For overloaded functions, you must use different C++ or Java entry point names in the underlying functions.

The data types defined on the C++ side (in /usr/include/impala_udf/udf.h) are:

- IntVal represents an INT column.
- BigIntVal represents a BIGINT column. Even if you do not need the full range of a BIGINT value, it can be useful to code your function arguments as BigIntVal to make it convenient to call the function with different kinds of integer columns and expressions as arguments. Impala automatically casts smaller integer types to larger ones when appropriate, but does not implicitly cast large integer types to smaller ones.
- SmallIntVal represents a SMALLINT column.
- TinyIntVal represents a TINYINT column.
- StringVal represents a STRING column. It has a len field representing the length of the string, and a ptr field pointing to the string data. It has constructors that create a new StringVal struct based on a null-terminated C-style string, or a pointer plus a length; these new structs still refer to the original string data rather than allocating a new buffer for the data. It also has a constructor that takes a pointer to a FunctionContext struct and a length, that does allocate space for a new copy of the string data, for use in UDFs that return string values.
- BooleanVal represents a BOOLEAN column.
- FloatVal represents a FLOAT column.
- DoubleVal represents a DOUBLE column.

- TimestampVal represents a TIMESTAMP column. It has a date field, a 32-bit integer representing the Gregorian date, that is, the days past the epoch date. It also has a time_of_day field, a 64-bit integer representing the current time of day in nanoseconds.

Variable-length argument lists

UDFs typically take a fixed number of arguments, with each one named explicitly in the signature of your C++ function. Your function can also accept additional optional arguments, all of the same type. For example, you can concatenate two strings, three strings, four strings, and so on. Or you can compare two numbers, three numbers, four numbers, and so on.

To accept a variable-length argument list, code the signature of your function like this:

```
StringVal Concat(FunctionContext* context, const StringVal& sepa
rator,
  int num_var_args, const StringVal* args);
```

In the CREATE FUNCTION statement, after the type of the first optional argument, include ... to indicate it could be followed by more arguments of the same type. For example, the following function accepts a STRING argument, followed by one or more additional STRING arguments:

```
[localhost:21000] > create function my_concat(string, string ...
) returns string location '/user/test_user/udfs/sample.so' symbo
l='Concat';
```

The call from the SQL query must pass at least one argument to the variable-length portion of the argument list.

When Impala calls the function, it fills in the initial set of required arguments, then passes the number of extra arguments and a pointer to the first of those optional arguments.

Handling NULL values

For correctness, performance, and reliability, it is important for each UDF to handle all situations where any NULL values are passed to your function. For example, when passed a NULL, UDFs typically also return NULL. In an aggregate function, which could be passed a combination of real and NULL values, you might make the final value into a NULL (as in CONCAT()), ignore the NULL value (as in AVG()), or treat it the same as a numeric zero or empty string.

Each parameter type, such as IntVal or StringVal, has an is_null Boolean member. Test this flag immediately for each argument to your function, and if it is set, do not refer to the val field of the argument structure. The val field is undefined when the argument is NULL, so your function could go into an infinite loop or produce incorrect results if you skip the special handling for NULL.

If your function returns NULL when passed a NULL value, or in other cases such as when a search string is not found, you can construct a null instance of the return type by using its null() member function.

Memory allocation for UDFs

By default, memory allocated within a UDF is deallocated when the function exits, which could be before the query is finished. The input arguments remain allocated for the lifetime of the function, so you can refer to them in the expressions for your return values. If you use temporary variables to construct all-new string values, use the StringVal() constructor that takes an initial FunctionCont ext* argument followed by a length, and copy the data into the newly allocated memory buffer.

Thread-safe work area for UDFs

One way to improve performance of UDFs is to specify the optional PREPARE_FN and CLOS E_FN clauses on the CREATE FUNCTION statement. The “prepare” function sets up a thread-safe data structure in memory that you can use as a work area. The “close” function deallocates that memory. Each subsequent call to the UDF within the same thread can access that same

memory area. There might be several such memory areas allocated on the same host, as UDFs are parallelized using multiple threads.

Within this work area, you can set up predefined lookup tables, or record the results of complex operations on data types such as STRING or TIMESTAMP. Saving the results of previous computations rather than repeating the computation each time is an optimization known as Memoization. For example, if your UDF performs a regular expression match or date manipulation on a column that repeats the same value over and over, you could store the last-computed value or a hash table of already-computed values, and do a fast lookup to find the result for subsequent iterations of the UDF.

Each such function must have the signature:

```
void FUNCTION_NAME(impala_udf::FunctionContext*, impala_udf::F
unctionContext::FunctionScope)
```

Currently, only THREAD_SCOPE is implemented, not FRAGMENT_SCOPE. See udf.h for details about the scope values.

Error handling for UDFs

To handle errors in UDFs, you call functions that are members of the initial FunctionContext* argument passed to your function.

A UDF can record one or more warnings, for conditions that indicate minor, recoverable problems that do not cause the query to stop. The signature for this function is:

```
bool AddWarning(const char* warning_msg);
```

For a serious problem that requires cancelling the query, a UDF can set an error flag that prevents the query from returning any results. The signature for this function is:

```
void SetError(const char* error_msg);
```

Related Information

Building and deploying UDFs

Building and deploying UDFs

Writing user-defined aggregate functions (UDAFs)

User-defined aggregate functions (UDAFs or UDAs) are a powerful and flexible category of user-defined functions. If a query processes N rows, calling a UDAF during the query condenses the result set, anywhere from a single value (such as with the SUM or MAX functions), or some number less than or equal to N (as in queries using the GROUP BY or HAVING clause).

The underlying functions for a UDA

A UDAF must maintain a state value across subsequent calls, so that it can accumulate a result across a set of calls, rather than derive it purely from one set of arguments. For that reason, a UDAF is represented by multiple underlying functions:

- An initialization function that sets any counters to zero, creates empty buffers, and does any other one-time setup for a query.
- An update function that processes the arguments for each row in the query result set and accumulates an intermediate result for each node. For example, this function might increment a counter, append to a string buffer, or set flags.
- A merge function that combines the intermediate results from two different nodes.
- A serialize function that flattens any intermediate values containing pointers, and frees any memory allocated during the init, update, and merge phases.

- A finalize function that either passes through the combined result unchanged, or does one final transformation.

In the SQL syntax, you create a UDAF by using the statement CREATE AGGREGATE FUNCTION. You specify the entry points of the underlying C++ functions using the clauses INIT_FN, UPDATE_FN, MERGE_FN, SERIALIZE_FN, and FINALIZE_FN.

For convenience, you can use a naming convention for the underlying functions and Impala automatically recognizes those entry points. Specify the UPDATE_FN clause, using an entry point name containing the string update or Update. When you omit the other _FN clauses from the SQL statement, Impala looks for entry points with names formed by substituting the update or Update portion of the specified name.

uda-sample.h:

See this file online at: uda-sample.h

uda-sample.cc:

See this file online at: uda-sample.cc

Intermediate results for UDAs

A user-defined aggregate function might produce and combine intermediate results during some phases of processing, using a different data type than the final return value. For example, if you implement a function similar to the built-in AVG() function, it must keep track of two values, the number of values counted and the sum of those values. Or, you might accumulate a string value over the course of a UDA, then in the end return a numeric or Boolean result.

In such a case, specify the data type of the intermediate results using the optional INTERMEDIATE *TYPE_NAME* clause of the CREATE AGGREGATE FUNCTION statement. If the intermediate data is a typeless byte array (for example, to represent a C++ struct or array), specify the type name as CHAR(*N*), with *N* representing the number of bytes in the intermediate result buffer.

For an example of this technique, see the trunc_sum() aggregate function, which accumulates intermediate results of type DOUBLE and returns BIGINT at the end. View the appropriate CREATE FUNCTION statement and the implementation of the underlying TruncSum*() functions on Github.

- test_udfs.py
- test-udas.cc

Building and deploying UDFs

This section explains the steps to compile Impala UDFs from C++ source code, and deploy the resulting libraries for use in Impala queries.

Impala UDF development package ships with a sample build environment for UDFs, that you can study, experiment with, and adapt for your own use.

To build the sample environment:

1. Install the Impala UDF development package as described in Installing the UDF development package on page 448.
2. Run the following commands:

```
cmake .
make
```

The `cmake` configuration command reads the file CMakeLists.txt and generates a Makefile customized for your particular directory paths. Then the `make` command runs the actual build steps based on the rules in the Makefile.

Impala loads the shared library from an HDFS location. After building a shared library containing one or more UDFs, use hdfs dfs or hadoop fs commands to copy the binary file to an HDFS location readable by Impala.

The final step in deployment is to issue a CREATE FUNCTION statement in the `impala-shell` interpreter to make Impala aware of the new function. Because each function is associated with a particular database, always issue a USE statement to the appropriate database before creating a function, or specify a fully qualified name, that is, CREATE FUNCTION *DB_NAME.FUNCTION_NAME*.

As you update the UDF code and redeploy updated versions of a shared library, use DROP FUNCTION and CREATE FUNCTION to let Impala pick up the latest version of the code.

Note:

In Impala 2.5 and higher, Impala UDFs and UDAs written in C++ are persisted in the metastore database. Java UDFs are also persisted, if they were created with the new CREATE FUNCTION syntax for Java UDFs, where the Java function argument and return types are omitted. Java-based UDFs created with the old CREATE FUNCTION syntax do not persist across restarts because they are held in the memory of the `catalogd` daemon. Until you re-create such Java UDFs using the new CREATE FUNCTION syntax, you must reload those Java-based UDFs by running the original CREATE FUNCTION statements again each time you restart the `catalogd` daemon. Prior to Impala 2.5 the requirement to reload functions after a restart applied to both C++ and Java functions.

See CREATE FUNCTION statement on page 135 and DROP FUNCTION statement on page 175 for the new syntax for the persistent Java UDFs.

Prerequisites for the build environment are:

1. Install the packages using the appropriate package installation command for your Linux distribution.

```
sudo yum install gcc-c++ cmake boost-devel
sudo yum install impala-udf-devel
# The package name on Ubuntu and Debian is impala-udf-dev.
```

2. Download the UDF sample code:

```
git clone https://github.com/cloudera/impala-udf-samples
cd impala-udf-samples &amp;amp;&amp;amp; cmake . &amp;amp;&amp;amp; make
```

3. Unpack the sample code in udf_samples.tar.gz and use that as a template to set up your build environment.

To build the original samples:

```
# Process CMakeLists.txt and set up appropriate Makefiles.
cmake .
# Generate shared libraries from UDF and UDAF sample code,
# udf_samples/libudfsample.so and udf_samples/libudasample.so
make
```

The sample code to examine, experiment with, and adapt is in these files:

- udf-sample.h: Header file that declares the signature for a scalar UDF (AddUDF).
- udf-sample.cc: Sample source for a simple UDF that adds two integers. Because Impala can reference multiple function entry points from the same shared library, you could add other UDF functions in this file and add their signatures to the corresponding header file.
- udf-sample-test.cc: Basic unit tests for the sample UDF.
- uda-sample.h: Header file that declares the signature for sample aggregate functions. The SQL functions will be called COUNT, AVG, and STRINGCONCAT. Because aggregate functions require more elaborate coding to handle the processing for multiple phases, there are several underlying C++ functions such as CountInit, AvgUpdate, and StringConcatFinalize.

- uda-sample.cc: Sample source for simple UDAFs that demonstrate how to manage the state transitions as the underlying functions are called during the different phases of query processing.
  - The UDAF that imitates the COUNT function keeps track of a single incrementing number; the merge functions combine the intermediate count values from each Impala node, and the combined number is returned verbatim by the finalize function.
  - The UDAF that imitates the AVG function keeps track of two numbers, a count of rows processed and the sum of values for a column. These numbers are updated and merged as with COUNT, then the finalize function divides them to produce and return the final average value.
  - The UDAF that concatenates string values into a comma-separated list demonstrates how to manage storage for a string that increases in length as the function is called for multiple rows.
- uda-sample-test.cc: basic unit tests for the sample UDAFs.

Performance considerations for UDFs

Because a UDF typically processes each row of a table, potentially being called billions of times, the performance of each UDF is a critical factor in the speed of the overall ETL or ELT pipeline. Tiny optimizations you can make within the function body can pay off in a big way when the function is called over and over when processing a huge result set.

Examples of creating and using UDFs

This section demonstrates how to create and use all kinds of user-defined functions (UDFs).

For downloadable examples that you can experiment with, adapt, and use as templates for your own functions, see the Cloudera sample UDF github. You must have already installed the appropriate header files, as explained in *Building and deploying UDFs*.

Sample C++ UDFs: HasVowels, CountVowels, StripVowels

This example shows 3 separate UDFs that operate on strings and return different data types. In the C++ code, the functions are HasVowels() (checks if a string contains any vowels), CountVowels() (returns the number of vowels in a string), and StripVowels() (returns a new string with vowels removed).

First, we add the signatures for these functions to udf-sample.h in the demo build environment:

```
BooleanVal HasVowels(FunctionContext* context, const StringVal&
input);
IntVal CountVowels(FunctionContext* context, const StringVal& ar
g1);
StringVal StripVowels(FunctionContext* context, const StringVal&
 arg1);
```

Then, we add the bodies of these functions to udf-sample.cc:

```
BooleanVal HasVowels(FunctionContext* context, const StringVal&
input)
{
        if (input.is_null) return BooleanVal::null();

        int index;
        uint8_t *ptr;

        for (ptr = input.ptr, index = 0; index <= input.len; i
ndex++, ptr++)
        {
```

```
                uint8_t c = tolower(*ptr);
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o'
 || c == 'u')
                {
                        return BooleanVal(true);
                }
        }
        return BooleanVal(false);
}

IntVal CountVowels(FunctionContext* context, const StringVal&
arg1)
{
        if (arg1.is_null) return IntVal::null();

        int count;
        int index;
        uint8_t *ptr;

        for (ptr = arg1.ptr, count = 0, index = 0; index <= arg1.
len; index++, ptr++)
        {
                uint8_t c = tolower(*ptr);
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o'
 || c == 'u')
                {
                        count++;
                }
        }
        return IntVal(count);
}

StringVal StripVowels(FunctionContext* context, const StringVal&
 arg1)
{
        if (arg1.is_null) return StringVal::null();

        int index;
        std::string original((const char *)arg1.ptr,arg1.len);
        std::string shorter("");

        for (index = 0; index < original.length(); index++)
        {
                uint8_t c = original[index];
                uint8_t l = tolower(c);

                if (l == 'a' || l == 'e' || l == 'i' || l == 'o'
 || l == 'u')
                {
                        ;
                }
                else
                {
                    shorter.append(1, (char)c);
                }
        }
// The modified string is stored in 'shorter', which is destroyed
 when this function ends. We need to make a string val
// and copy the contents.
        StringVal result(context, shorter.size()); // Only the ve
rsion of the ctor that takes a context object allocates new memo
ry
        memcpy(result.ptr, shorter.c_str(), shorter.size());
        return result;
```

```
}
```

We build a shared library, libudfsample.so, and put the library file into HDFS where Impala can read it:

```
$ make
[  0%] Generating udf_samples/uda-sample.ll
[ 16%] Built target uda-sample-ir
[ 33%] Built target udasample
[ 50%] Built target uda-sample-test
[ 50%] Generating udf_samples/udf-sample.ll
[ 66%] Built target udf-sample-ir
Scanning dependencies of target udfsample
[ 83%] Building CXX object CMakeFiles/udfsample.dir/udf-sample.o
Linking CXX shared library udf_samples/libudfsample.so
[ 83%] Built target udfsample
Linking CXX executable udf_samples/udf-sample-test
[100%] Built target udf-sample-test
$ hdfs dfs -put ./udf_samples/libudfsample.so /user/hive/udfs/li
budfsample.so
```

Finally, we go into the `impala-shell` interpreter where we set up some sample data, issue CREATE FUNCTION statements to set up the SQL function names, and call the functions in some queries:

```
[localhost:21000] > create database udf_testing;
[localhost:21000] > use udf_testing;

[localhost:21000] > create function has_vowels (string) returns b
oolean location '/user/hive/udfs/libudfsample.so' symbol='HasVow
els';
[localhost:21000] > select has_vowels('abc');
+------------------------+
| udfs.has_vowels('abc') |
+------------------------+
| true                   |
+------------------------+
Returned 1 row(s) in 0.13s
[localhost:21000] > select has_vowels('zxcvbnm');
+----------------------------+
| udfs.has_vowels('zxcvbnm') |
+----------------------------+
| false                      |
+----------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select has_vowels(null);
+-----------------------+
| udfs.has_vowels(null) |
+-----------------------+
| NULL                  |
+-----------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > select s, has_vowels(s) from t2;
+-----------+--------------------+
| s         | udfs.has_vowels(s) |
+-----------+--------------------+
| lower     | true               |
| UPPER     | true               |
| Init cap  | true               |
| CamelCase | true               |
+-----------+--------------------+
Returned 4 row(s) in 0.24s
```

```
[localhost:21000] > create function count_vowels (string) retur
ns int location '/user/hive/udfs/libudfsample.so' symbol='CountV
owels';
[localhost:21000] > select count_vowels('cat in the hat');
+-------------------------------------+
| udfs.count_vowels('cat in the hat') |
+-------------------------------------+
| 4                                   |
+-------------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select s, count_vowels(s) from t2;
+-----------+----------------------+
| s         | udfs.count_vowels(s) |
+-----------+----------------------+
| lower     | 2                    |
| UPPER     | 2                    |
| Init cap  | 3                    |
| CamelCase | 4                    |
+-----------+----------------------+
Returned 4 row(s) in 0.23s
[localhost:21000] > select count_vowels(null);
+-------------------------+
| udfs.count_vowels(null) |
+-------------------------+
| NULL                    |
+-------------------------+
Returned 1 row(s) in 0.12s

[localhost:21000] > create function strip_vowels (string) returns
 string location '/user/hive/udfs/libudfsample.so' symbol='Strip
Vowels';
[localhost:21000] > select strip_vowels('abcdefg');
+------------------------------+
| udfs.strip_vowels('abcdefg') |
+------------------------------+
| bcdfg                        |
+------------------------------+
Returned 1 row(s) in 0.11s
[localhost:21000] > select strip_vowels('ABCDEFG');
+------------------------------+
| udfs.strip_vowels('abcdefg') |
+------------------------------+
| BCDFG                        |
+------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select strip_vowels(null);
+-------------------------+
| udfs.strip_vowels(null) |
+-------------------------+
| NULL                    |
+-------------------------+
Returned 1 row(s) in 0.16s
[localhost:21000] > select s, strip_vowels(s) from t2;
+-----------+----------------------+
| s         | udfs.strip_vowels(s) |
+-----------+----------------------+
| lower     | lwr                  |
| UPPER     | PPR                  |
| Init cap  | nt cp                |
| CamelCase | CmlCs                |
+-----------+----------------------+
Returned 4 row(s) in 0.24s
```

Sample C++ UDA: SumOfSquares

```
[localhost:21000] > insert overwrite sos values (1, 1), (2, 0),
(3, 1), (4, 0);
Inserted 4 rows in 1.24s

[localhost:21000] > -- Compute 1 squared + 3 squared, and 2 sq
uared + 4 squared;
[localhost:21000] > select y, sum_of_squares(x) from sos group by
 y;
+---+------------------------+
| y | udfs.sum_of_squares(x) |
+---+------------------------+
| 1 | 10                     |
| 0 | 20                     |
+---+------------------------+
Returned 2 row(s) in 0.43s
```

This example demonstrates a user-defined aggregate function (UDA) that produces the sum of the squares of its input values.

The coding for a UDA is a little more involved than a scalar UDF, because the processing is split into several phases, each implemented by a different function. Each phase is relatively straightforward: the “update” and “merge” phases, where most of the work is done, read an input value and combine it with some accumulated intermediate value.

As in our sample UDF from the previous example, we add function signatures to a header file (in this case, uda-sample.h). Because this is a math-oriented UDA, we make two versions of each function, one accepting an integer value and the other accepting a floating-point value.

```
void SumOfSquaresInit(FunctionContext* context, BigIntVal* val);
void SumOfSquaresInit(FunctionContext* context, DoubleVal* val);

void SumOfSquaresUpdate(FunctionContext* context, const BigIntVal
& input, BigIntVal* val);
void SumOfSquaresUpdate(FunctionContext* context, const Double
Val& input, DoubleVal* val);

void SumOfSquaresMerge(FunctionContext* context, const BigIntV
al& src, BigIntVal* dst);
void SumOfSquaresMerge(FunctionContext* context, const DoubleV
al& src, DoubleVal* dst);

BigIntVal SumOfSquaresFinalize(FunctionContext* context, const Bi
gIntVal& val);
DoubleVal SumOfSquaresFinalize(FunctionContext* context, const Do
ubleVal& val);
```

We add the function bodies to a C++ source file (in this case, uda-sample.cc):

```
void SumOfSquaresInit(FunctionContext* context, BigIntVal* val) {
  val->is_null = false;
  val->val = 0;
}
void SumOfSquaresInit(FunctionContext* context, DoubleVal* val) {
  val->is_null = false;
  val->val = 0.0;
}

void SumOfSquaresUpdate(FunctionContext* context, const BigIntVal
& input, BigIntVal* val) {
  if (input.is_null) return;
```

```
  val->val += input.val * input.val;
}
void SumOfSquaresUpdate(FunctionContext* context, const DoubleVal
& input, DoubleVal* val) {
  if (input.is_null) return;
  val->val += input.val * input.val;
}

void SumOfSquaresMerge(FunctionContext* context, const BigIntVal&
 src, BigIntVal* dst) {
  dst->val += src.val;
}
void SumOfSquaresMerge(FunctionContext* context, const DoubleV
al& src, DoubleVal* dst) {
  dst->val += src.val;
}
BigIntVal SumOfSquaresFinalize(FunctionContext* context, const
BigIntVal& val) {
  return val;
}
DoubleVal SumOfSquaresFinalize(FunctionContext* context, const
 DoubleVal& val) {
  return val;
}
```

As with the sample UDF, we build a shared library and put it into HDFS:

```
$ make
[  0%] Generating udf_samples/uda-sample.ll
[ 16%] Built target uda-sample-ir
Scanning dependencies of target udasample
[ 33%] Building CXX object CMakeFiles/udasample.dir/uda-sample.o
Linking CXX shared library udf_samples/libudasample.so
[ 33%] Built target udasample
Scanning dependencies of target uda-sample-test
[ 50%] Building CXX object CMakeFiles/uda-sample-test.dir/uda-s
ample-test.o
Linking CXX executable udf_samples/uda-sample-test
[ 50%] Built target uda-sample-test
[ 50%] Generating udf_samples/udf-sample.ll
[ 66%] Built target udf-sample-ir
[ 83%] Built target udfsample
[100%] Built target udf-sample-test
$ hdfs dfs -put ./udf_samples/libudasample.so /user/hive/udfs/li
budasample.so
```

To create the SQL function, we issue a CREATE AGGREGATE FUNCTION statement and specify the underlying C++ function names for the different phases:

```
[localhost:21000] > use udf_testing;

[localhost:21000] > create table sos (x bigint, y double);
[localhost:21000] > insert into sos values (1, 1.1), (2, 2.2),
(3, 3.3), (4, 4.4);
Inserted 4 rows in 1.10s

[localhost:21000] > create aggregate function sum_of_squares(b
igint) returns bigint
  > location '/user/hive/udfs/libudasample.so'
  > init_fn='SumOfSquaresInit'
  > update_fn='SumOfSquaresUpdate'
  > merge_fn='SumOfSquaresMerge'
  > finalize_fn='SumOfSquaresFinalize';
```

```
[localhost:21000] > -- Compute the same value using literals or
 the UDA;
[localhost:21000] > select 1*1 + 2*2 + 3*3 + 4*4;
+-------------------------------+
| 1 * 1 + 2 * 2 + 3 * 3 + 4 * 4 |
+-------------------------------+
| 30                            |
+-------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select sum_of_squares(x) from sos;
+------------------------+
| udfs.sum_of_squares(x) |
+------------------------+
| 30                     |
+------------------------+
Returned 1 row(s) in 0.35s
```

Until we create the overloaded version of the UDA, it can only handle a single data type. To allow it to handle DOUBLE as well as BIGINT, we issue another CREATE AGGREGATE FUNCTION statement:

```
[localhost:21000] > select sum_of_squares(y) from sos;
ERROR: AnalysisException: No matching function with signature: ud
fs.sum_of_squares(DOUBLE).

[localhost:21000] > create aggregate function sum_of_squares(dou
ble) returns double
  > location '/user/hive/udfs/libudasample.so'
  > init_fn='SumOfSquaresInit'
  > update_fn='SumOfSquaresUpdate'
  > merge_fn='SumOfSquaresMerge'
  > finalize_fn='SumOfSquaresFinalize';

[localhost:21000] > -- Compute the same value using literals or t
he UDA;
[localhost:21000] > select 1.1*1.1 + 2.2*2.2 + 3.3*3.3 + 4.4*4.4;
+-----------------------------------------------+
| 1.1 * 1.1 + 2.2 * 2.2 + 3.3 * 3.3 + 4.4 * 4.4 |
+-----------------------------------------------+
| 36.3                                          |
+-----------------------------------------------+
Returned 1 row(s) in 0.12s
[localhost:21000] > select sum_of_squares(y) from sos;
+------------------------+
| udfs.sum_of_squares(y) |
+------------------------+
| 36.3                   |
+------------------------+
Returned 1 row(s) in 0.35s
```

Typically, you use a UDA in queries with GROUP BY clauses, to produce a result set with a separate aggregate value for each combination of values from the GROUP BY clause. Let's change our sample table to use 0 to indicate rows containing even values, and 1 to flag rows containing odd values. Then the GROUP BY query can return two values, the sum of the squares for the even values, and the sum of the squares for the odd values:

Related Information

Building and deploying UDFs

Building and deploying UDFs

Security considerations for UDFs

When the Impala authorization feature is enabled:

- To call a UDF in a query, you must have the required read privilege for any databases and tables used in the query.
- The CREATE FUNCTION statement requires:
  - The CREATE privilege on the database.
  - The ALL privilege on two URIs where the URIs are:
    - The JAR file on the file system. For example:

```
GRANT ALL ON URI 'file:///PATH_TO_MY.JAR' TO ROLE MY_ROLE;
```

    - The JAR on HDFS. For example:

```
GRANT ALL ON URI 'hdfs:///PATH/TO/JAR' TO ROLE MY_ROLE
```

Limitations and restrictions for Impala UDFs

The following limitations and restrictions apply to Impala UDFs in the current release.

Limited support for Hive Generic UDFs

Hive has 2 types of UDFs. This release contains limited support for the second generation UDFs called GenericUDFs. The main limitations are as follows:

- Decimal types are not supported
- Complex types are not supported
- Functions are not extracted from the jar file

GenericUDFs cannot be made permanent. They will need to be recreated every time the server is restarted.

Other limitations

- Impala does not support Hive UDFs that accept or return composite or nested types, or other types not available in Impala tables.
- The Hive current_user() function cannot be called from a Java UDF through Impala.
- All Impala UDFs must be deterministic, that is, produce the same output each time when passed the same argument values. For example, an Impala UDF must not call functions such as rand() to produce different values for each invocation. It must not retrieve data from external sources, such as from disk or over the network.
- An Impala UDF must not spawn other threads or processes.
- Prior to Impala 2.5 when the `catalogd` process is restarted, all UDFs become undefined and must be reloaded. In Impala 2.5 and higher, this limitation only applies to older Java UDFs. Re-create those UDFs using the new CREATE FUNCTION syntax for Java UDFs, which excludes the function signature, to remove the limitation entirely.
- Impala currently does not support user-defined table functions (UDTFs).
- The CHAR and VARCHAR types cannot be used as input arguments or return values for UDFs.

SQL transactions in Impala

A transaction is a single logical operation on the data. Impala supports transactions that satisfy a level of consistency that improves the integrity and reliability of the data before and after a transaction.

Specifically, Impala provides atomicity and isolation of insert operations on transactional tables. A single table insert is either committed in full or not committed, and the results of the insert operation are not visible to other query operations until the operation is committed.

For single table, the inserts are ordered, so if Impala doesn't see a committed insert, it won't see any insert committed after it.

For insert-only transactional tables, you can perform the following statements: CREATE TABLE, DROP TABLE, TRUNCATE, INSERT, SELECT

All transactions in Impala automatically commit at the end of the statement. Currently, Impala does not support multi-statement transactions.

Insert-only tables must be the managed and file-format based tables, such as Parquet, Avro, and text.

Note:

- Impala does not support changing transactional properties of tables. For example, you cannot alter a transactional table to a non-transactional table.
- Currently, ALTER TABLE statement is not supported on both insert-only and full acid transactional tables.

Related Information

CREATE TABLE statement

Impala reserved words

Impala reserves a set of keywords that cannot be used directly as identifiers in Impala SQL. You must be aware of the list of reserved words to avoid name conflicts with any of your table, columns, or other object names.

If you need to use it as an identifier, you must quote it with backticks. For example:

- CREATE TABLE select (x INT): fails
- CREATE TABLE `select` (x INT): succeeds

Because different database systems have different sets of reserved words, and the reserved words change from release to release, carefully consider database, table, and column names to ensure maximum compatibility between products and versions.

Also consider whether your object names are the same as any Hive keywords, and rename or quote any that conflict as you might switch between Impala and Hive when doing analytics and ETL.

To future-proof your code, you should avoid additional words in case they become reserved words if Impala adds features in later releases. This kind of planning can also help to avoid name conflicts in case you port SQL from other systems that have different sets of reserved words. The Future Keyword column in the table below indicates those additional words that you should avoid for table, column, or other object names, even though they are not currently reserved by Impala.

The following is a summary of the process for deciding whether a particular SQL 2016 word is to be reserved in Impala.

- By default, Impala targets to have the same list of reserved words as SQL 2016.
- At the same time, to be compatible with earlier versions of Impala and to avoid breaking existing tables/workloads, Impala built-in function names are removed from the reserved words list, e.g. COUNT, AVG, as Impala generally does not need to reserve the names of built-in functions for parsing to work.
- For those remaining SQL 2016 reserved words, if a word is likely to be in-use by users of older Impala versions and if there is a low chance of Impala needing to reserve that word in the future, then the word is not reserved.
- Otherwise, the word is reserved in Impala.

The following table lists the reserved words in Impala.

Keyword	Reserved in SQL:2016	Reserved in Impala 3.0 and higher	Future Keyword
abs	X		
acos	X		
add		X	
aggregate		X	
all	X	X	
allocate	X	X	
alter	X	X	
analytic		X	
and	X	X	
anti		X	
any	X	X	
api_version		X	
are	X	X	
array	X	X	
array_agg	X	X	
array_max_cardinality	X	X	
as	X	X	
asc		X	
asensitive	X	X	
asin	X		
asymmetric	X	X	
at	X	X	
atan	X		
atomic	X	X	
authorization	X	X	
avg	X		
avro		X	
backup			X
begin	X		X
begin_frame	X	X	
begin_partition	X	X	

between	X	X	
bigint	X	X	
binary	X	X	
blob	X	X	
block_size		X	
boolean	X	X	
both	X	X	
break			X
browse			X
bulk			X
by	X	X	
cache			
cached		X	
call	X		
called	X	X	
cardinality	X	X	
cascade		X	
cascaded	X	X	
case	X	X	
cast	X	X	
ceil	X		
ceiling	X		
change		X	
char	X	X	
char_length	X		
character	X	X	
character_length	X		
check	X		X
checkpoint			X
class		X	
classifier	X		
clob	X	X	
close	X		X
close_fn		X	
clustered			X
coalesce	X		X
collate	X	X	
collect	X	X	
column	X	X	
columns		X	

comment		X	
commit	X	X	
compression		X	
compute		X	
condition	X	X	
conf			
connect	X	X	
constraint	X	X	
contains	X	X	
continue			X
convert	X	X	
copy	X	X	
corr	X	X	
corresponding	X	X	
cos	X		
cosh	X		
count	X		
covar_pop	X	X	
covar_samp	X	X	
create	X	X	
cross	X	X	
cube	X	X	
cume_dist	X		
current	X	X	
current_catalog	X		
current_date	X	X	
current_default_transform_group	X	X	
current_path	X	X	
current_role	X	X	
current_row	X	X	
current_schema	X	X	
current_time	X	X	
current_timestamp	X		X
current_transform_group_for_type	X	X	
current_user	X		X
cursor	X	X	
cycle	X	X	
data		X	
database		X	
databases		X	

date	X	X	
datetime		X	
day	X		
dayofweek			
dbcc			X
deallocate	X	X	
dec	X	X	
decfloat	X	X	
decimal	X	X	
declare	X	X	
default	X	X	
define	X	X	
delete	X	X	
delimited		X	
dense_rank	X		
deny			X
deref	X	X	
desc		X	
describe	X	X	
deterministic	X	X	
disconnect	X	X	
disk			X
distinct	X	X	
distributed			X
div		X	
double	X	X	
drop	X	X	
dump			X
dynamic	X	X	
each	X	X	
element	X	X	
else	X	X	
empty	X	X	
encoding		X	
end	X	X	
end-exec	X	X	
end_frame	X	X	
end_partition	X	X	
equals	X	X	
errlvl			X

escape	X	X	
escaped		X	
every	X	X	
except	X	X	
exchange			
exec	X	X	
execute	X	X	
exists	X	X	
exit			X
exp	X		
explain		X	
extended		X	
external	X	X	
extract	X		
false	X	X	
fetch	X	X	
fields		X	
file			X
filefactor			X
fileformat		X	
files		X	
filter	X	X	
finalize_fn		X	
first		X	
first_value	X		
float	X	X	
floor	X		
following		X	
for	X	X	
foreign	X	X	
format		X	
formatted		X	
frame_row	X	X	
free	X	X	
freetext			X
from	X	X	
full	X	X	
function	X	X	
functions		X	
fusion	X	X	

get	X	X	
global	X	X	
goto			X
grant	X	X	
group	X	X	
grouping	X	X	
groups	X	X	
hash		X	
having	X	X	
hold	X	X	
holdlock			X
hour	X		
iceberg		X[1]	
identity	X		X
if		X	
ignore		X	
ilike		X	
import			
in	X	X	
incremental		X	
index			X
indicator	X	X	
init_fn		X	
initial	X	X	
inner	X	X	
inout	X	X	
inpath		X	
insensitive	X	X	
insert	X	X	
int	X	X	
integer	X	X	
intermediate		X	
intersect	X	X	
intersection	X	X	
interval	X	X	
into	X	X	
invalidate		X	
iregexp		X	
is	X	X	

[1] Impala 4.0 and higher

join	X	X	
json_array	X	X	
json_arrayagg	X	X	
json_exists	X	X	
json_object	X	X	
json_objectagg	X	X	
json_query	X	X	
json_table	X	X	
json_table_primitive	X	X	
json_value	X	X	
key			X
kill			X
kudu		X	
lag	X		
language	X		
large	X	X	
last		X	
last_value	X		
lateral	X	X	
lead	X		
leading	X	X	
left	X	X	
less			
like	X	X	
like_regex	X	X	
limit		X	
lineno			X
lines		X	
listagg	X	X	
ln	X		
load		X	
local	X	X	
localtime	X		
localtimestamp	X	X	
location		X	
log	X		
log10	X	X	
lower	X		
macro			
map		X	

match	X	X	
match_number	X	X	
match_recognize	X	X	
matches	X	X	
max	X		
member	X		
merge	X	X	
merge_fn		X	
metadata		X	
method	X	X	
min	X		
minute	X		
mod	X		
modifies	X	X	
module	X		
month	X		
more			
multiset	X	X	
national	X	X	
natural	X	X	
nchar	X	X	
nclob	X	X	
new	X		
no	X	X	
nocheck			X
nonclustered			X
none	X	X	
normalize	X	X	
not	X	X	
nth_value	X	X	
ntile	X		
null	X	X	
nullif	X		X
nulls		X	
numeric	X	X	
occurrences_regex	X	X	
octet_length	X	X	
of	X	X	
off			X
offset	X	X	

offsets			X
old	X		
omit	X	X	
on	X	X	
one	X	X	
only	X	X	
open	X		X
option			X
or	X	X	
order	X	X	
out	X	X	
outer	X	X	
over	X	X	
overlaps	X	X	
overlay	X	X	
overwrite		X	
parameter	X		
parquet		X	
parquetfile		X	
partialscan			
partition	X	X	
partitioned		X	
partitions		X	
pattern	X	X	
per	X	X	
percent	X	X	
percent_rank	X		
percentile_cont	X	X	
percentile_disc	X	X	
period	X		
pivot			X
plan			X
portion	X	X	
position	X	X	
position_regex	X	X	
power	X		
precedes	X	X	
preceding		X	
precision	X		X
prepare	X	X	

prepare_fn		X	
preserve			
primary	X	X	
print			X
proc			X
procedure	X	X	
produced		X	
ptf	X	X	
public			X
purge		X	
raiseerror			X
range	X	X	
rank	X		
rcfile		X	
read			X
reads	X	X	
readtext			X
real	X	X	
reconfigure			X
recover		X	
recursive	X	X	
reduce			
ref	X	X	
references	X	X	
referencing	X	X	
refresh		X	
regexp		X	
regr_avgx	X	X	
regr_avgy	X	X	
regr_count	X	X	
regr_intercept	X	X	
regr_r2	X	X	
regr_slope	X	X	
regr_sxx	X	X	
regr_sxy	X	X	
regr_syy	X	X	
release	X	X	
rename		X	
repeatable		X	
replace		X	

replication		X	
restore			X
restrict		X	
result	X		
return	X		X
returns	X	X	
revert			X
revoke	X	X	
right	X	X	
rlike		X	
role		X	
roles		X	
rollback	X	X	
rollup	X	X	
row	X	X	
row_number	X		
rowcount			X
rows	X	X	
rule			X
running	X	X	
save			X
savepoint	X	X	
schema		X	
schemas		X	
scope	X	X	
scroll	X	X	
search	X	X	
second	X		
securityaudit			X
seek	X	X	
select	X	X	
semi		X	
sensitive	X	X	
sequencefile		X	
serdeproperties		X	
serialize_fn		X	
session_user	X		X
set	X	X	
setuser			X
show	X	X	

shutdown			X
similar	X	X	
sin	X		
sinh	X		
skip	X	X	
smallint	X	X	
some	X	X	
sort		X	
spec		X	
specific	X	X	
specifictype	X	X	
sql	X		
sqlexception	X	X	
sqlstate	X	X	
sqlwarning	X	X	
sqrt	X		
start	X		
static	X	X	
statistics			X
stats		X	
stddev_pop	X		
stddev_samp	X		
stored		X	
straight_join		X	
string		X	
struct		X	
submultiset	X	X	
subset	X	X	
substring	X		
substring_regex	X	X	
succeeds	X	X	
sum	X		
symbol		X	
symmetric	X	X	
system	X		
system_time	X	X	
system_user	X	X	
table	X	X	
tables		X	
tablesample	X	X	

tan	X		
tanh	X		
tblproperties		X	
terminated		X	
textfile		X	
textsize			X
then	X	X	
time	X		
timestamp	X	X	
timezone_hour	X	X	
timezone_minute	X	X	
tinyint		X	
to	X	X	
top			X
trailing	X	X	
tran			X
transform			
translate	X		
translate_regex	X	X	
translation	X	X	
treat	X	X	
trigger	X	X	
trim	X		
trim_array	X	X	
true	X	X	
truncate	X	X	
try_convert			X
uescape	X	X	
unbounded		X	
uncached		X	
union	X	X	
unique	X	X	
uniquejoin			
unknown	X	X	
unnest	X	X	
unpivot			X
update	X	X	
update_fn		X	
updatetext			X
upper	X		

upsert		X	
use		X	
user	X		X
using	X	X	
utc_tmestamp			
validate		X[2]	
value	X		
value_of	X	X	
values	X	X	
var_pop	X		
var_samp	X		
varbinary	X	X	
varchar	X	X	
varying	X	X	
versioning	X	X	
view		X	
views			
waitfor			X
when	X	X	
whenever	X	X	
where	X	X	
while			X
width_bucket	X	X	
window	X	X	
with	X	X	
within	X	X	
without	X	X	
writetext			X
year	X		

SQL differences between Impala and Hive

Impala's SQL syntax follows the SQL-92 standard, and includes extensions, such as built-in functions. Because Impala and Hive share the same Metastore database and their tables are often used interchangeably, this topic covers differences between Impala and Hive in detail.

The current release of Impala does not support the following SQL features that available in Hive SQL:

- Extensibility mechanisms such as TRANSFORM, custom file formats, or custom SerDes.
- XML functions.
- BINARY data type.

[2] Impala 3.4 and higher

- Certain aggregate functions from Hive SQL: covar_pop, covar_samp, corr, percentile, percentile_approx, histogram_numeric, collect_set. Impala supports the set of aggregate functions and analytic functions.
- Lateral views. In Impala 2.3 and higher, Impala supports queries on complex types (STRUCT, ARRAY, or MAP), using join notation rather than the EXPLODE() keyword.

User-defined functions (UDFs) are supported from Impala 1.2 onward.

- Impala supports high-performance UDFs written in C++, as well as reusing some Java-based Hive UDFs.
- Impala supports scalar UDFs and user-defined aggregate functions (UDAFs). Impala does not currently support user-defined table generating functions (UDTFs).
- Only Impala-supported column types are supported in Java-based UDFs.
- The Hive current_user() function cannot be called from a Java UDF through Impala.

Impala does not currently support these Hive SQL statements:

- ANALYZE TABLE (the Impala equivalent is COMPUTE STATS)
- DESCRIBE COLUMN
- DESCRIBE DATABASE
- EXPORT TABLE
- IMPORT TABLE
- SHOW TABLE EXTENDED
- SHOW TBLPROPERTIES
- SHOW INDEXES
- SHOW COLUMNS
- INSERT OVERWRITE DIRECTORY; use INSERT OVERWRITE *TABLE_NAME* or CREATE TABLE AS SELECT to materialize query results into the HDFS directory associated with an Impala table.

Impala respects the serialization.null.format table property only for TEXT tables and ignores the property for Parquet and other formats. Hive respects the serialization.null.format property for Parquet and other formats and converts matching values to NULL during the scan. See the *Data files for text tables* section for using the table property in Impala.

Semantic differences between Impala and Hive SQL features

This section covers instances where Impala and Hive have similar functionality, sometimes including the same syntax, but there are differences in the runtime semantics of those features.

SQL statements and clauses:

- Impala uses different syntax and names for query hints, [SHUFFLE] and [NOSHUFFLE] rather than MapJoin or StreamJoin.
- Impala does not expose MapReduce specific features of SORT BY, DISTRIBUTE BY, or CLUSTER BY.
- Impala does not require queries to include a FROM clause.

Data types:

- Impala supports a limited set of implicit casts. This can help avoid undesired results from unexpected casting behavior.
  - Impala does not implicitly cast between string and numeric or Boolean types. Always use CAST() for these conversions.
  - Impala does perform implicit casts among the numeric types, when going from a smaller or less precise type to a larger or more precise one. For example, Impala will implicitly convert a SMALLINT to a BIGINT or FLOAT, but to convert from DOUBLE to FLOAT or INT to TINYINT requires a call to CAST() in the query.
  - Impala does perform implicit casts from STRING to TIMESTAMP. Impala has a restricted set of literal formats for the TIMESTAMP data type and the FROM_UNIXTIME() format string.

  See the topics under Impala SQL data types on page 13 for full details on implicit and explicit casting for all types, and Impala type conversion functions on page 339 for details about the CAST() function.

- Impala does not store or interpret TIMESTAMP values using the local timezone, to avoid undesired results from unexpected time zone issues. Timestamps are stored and interpreted relative to UTC. This difference can produce different results for some calls to similarly named date/time functions between Impala and Hive. See TIMESTAMP data type on page 46 for a discussion of how Impala handles time zones, and configuration options you can use to make Impala match the Hive behavior more closely when dealing with Parquet-encoded TIMESTAMP data or when converting between the local time zone and UTC.
- The Impala TIMESTAMP type can represent dates ranging from 1400-01-01 to 9999-12-31. This is different from the Hive date range, which is 0000-01-01 to 9999-12-31.
- Impala does not return column overflows as NULL, so that customers can distinguish between NULL data and overflow conditions similar to how they do so with traditional database systems. Impala returns the largest or smallest value in the range for the type. For example, valid values for a tinyint range from -128 to 127. In Impala, a tinyint with a value of -200 returns -128 rather than NULL. A tinyint with a value of 200 returns 127.

Miscellaneous features:

- Impala does not provide virtual columns.
- Impala does not expose locking.
- Impala does not expose some configuration properties.

Related Information

Adapting SQL code from a variety of database systems to Impala

Using text files in Impala

Porting SQL from other database systems to Impala

Although Impala uses standard SQL for queries, you might need to modify SQL source when bringing applications to Impala, due to variations in data types, built-in functions, vendor language extensions, and Hadoop-specific syntax. Even when SQL is working correctly, you might make further minor modifications for best performance.

Porting DDL and DML statements

When adapting SQL code from a legacy database system to Impala, expect to find a number of differences in the DDL statements that you use to set up the schema. Clauses related to physical layout of files, tablespaces, and indexes have no equivalent in Impala. You might restructure your schema considerably to account for the Impala partitioning scheme and Hadoop file formats.

Expect SQL queries to have a much higher degree of compatibility. With modest rewriting to address vendor extensions and features not yet supported in Impala, you might be able to run identical or almost-identical query text on both systems.

Therefore, consider separating out the DDL into a separate Impala-specific setup script. Focus your reuse and ongoing tuning efforts on the code for SQL queries.

Porting data types from other database systems

- Change any VARCHAR, VARCHAR2, and CHAR columns to STRING. Remove any length constraints from the column declarations; for example, change VARCHAR(32) or CHAR(1) to STRING. Impala is very flexible about the length of string values; it does not impose any length constraints or do any special processing (such as blank-padding) for STRING columns. (In Impala 2.0 and higher, there are data types VARCHAR and CHAR, with length constraints for both types and blank-padding for CHAR. However, for performance reasons, it is still preferable to use STRING columns where practical.)
- For national language character types such as NCHAR, NVARCHAR, or NCLOB, be aware that while Impala can store and query UTF-8 character data, currently some string manipulation operations only work correctly with ASCII data.
- Change any DATETIME or TIME columns to DATE or TIMESTAMP. Remove any precision constraints. Remove any timezone clauses, and make sure your application logic or ETL process accounts for the fact that

Impala expects all TIMESTAMP values to be in the Coordinated Universal Time (UTC). See TIMESTAMP data type on page 46 for information about the TIMESTAMP data type, and Impala date and time functions on page 347 for conversion functions for different date and time formats.

You might also need to adapt date- and time-related literal values and format strings to use the supported Impala date and time formats. If you have date and time literals with different separators or different numbers of YY, MM, and so on placeholders than Impala expects, consider using calls to regexp_replace() to transform those values to the Impala-compatible format. See TIMESTAMP data type on page 46 for information about the allowed formats for date and time literals, and Impala string functions on page 371 for string conversion functions such as regexp_replace().

Instead of SYSDATE, call the function NOW().

Instead of adding or subtracting directly from a date value to produce a value *N* days in the past or future, use an INTERVAL expression, for example NOW() + INTERVAL 30 DAYS.

- Although Impala supports INTERVAL expressions for datetime arithmetic, as shown in TIMESTAMP data type on page 46, INTERVAL is not available as a column data type in Impala. For any INTERVAL values stored in tables, convert them to numeric values that you can add or subtract using the functions in Impala date and time functions on page 347. For example, if you had a table DEADLINES with an INT column TIME_PERIOD, you could construct dates N days in the future like so:

```
SELECT NOW() + INTERVAL time_period DAYS from deadlines;
```

- For YEAR columns, change to the smallest Impala integer type that has sufficient range. See Impala SQL data types on page 13 for details about ranges, casting, and so on for the various numeric data types.
- Change any DECIMAL and NUMBER types. If fixed-point precision is not required, you can use FLOAT or DOUBLE on the Impala side depending on the range of values. For applications that require precise decimal values, such as financial data, you might need to make more extensive changes to table structure and application logic, such as using separate integer columns for dollars and cents, or encoding numbers as string values and writing UDFs to manipulate them.
- FLOAT, DOUBLE, and REAL types are supported in Impala. Remove any precision and scale specifications. (In Impala, REAL is just an alias for DOUBLE; columns declared as REAL are turned into DOUBLE behind the scenes.)
- Most integer types from other systems have equivalents in Impala, perhaps under different names such as BIGINT instead of INT8. For any that are unavailable, for example MEDIUMINT, switch to the smallest Impala integer type that has sufficient range. Remove any precision specifications.
- Remove any UNSIGNED constraints. All Impala numeric types are signed.
- For any types holding bitwise values, use an integer type with enough range to hold all the relevant bits within a positive integer.

  For example, TINYINT has a maximum positive value of 127, not 256, so to manipulate 8-bit bitfields as positive numbers switch to the next largest type SMALLINT. For example, CAST(127*2 AS SMALLINT).

  Impala does not support notation such as b'0101' for bit literals.
- For BLOB values, use STRING to represent CLOB or TEXT types (character based large objects) up to 32 KB in size. Binary large objects such as BLOB, RAW BINARY, and VARBINARY do not currently have an equivalent in Impala.
- For Boolean-like types such as BOOL, use the Impala BOOLEAN type.
- Because Impala currently does not support composite or nested types, any spatial data types in other database systems do not have direct equivalents in Impala. You could represent spatial values in string format and write UDFs to process them. Where practical, separate spatial types into separate tables so that Impala can still work with the non-spatial data.
- Take out any DEFAULT clauses. Impala can use data files produced from many different sources, such as Pig, Hive, or MapReduce jobs. The fast import mechanisms of LOAD DATA and external tables mean that Impala is flexible about the format of data files, and Impala does not necessarily validate or cleanse data before querying it. When copying data through Impala INSERT statements, you can use conditional functions such as CASE or NVL to substitute some other value for NULL fields.

- Take out any constraints from your CREATE TABLE and ALTER TABLE statements, for example PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, UNSIGNED, or CHECK constraints. Impala can use data files produced from many different sources, such as Pig, Hive, or MapReduce jobs. Therefore, Impala expects initial data validation to happen earlier during the ETL or ELT cycle. After data is loaded into Impala tables, you can perform queries to test for NULL values. When copying data through Impala INSERT statements, you can use conditional functions such as CASE or NVL to substitute some other value for NULL fields.

  Do as much verification as practical before loading data into Impala. After data is loaded into Impala, you can do further verification using SQL queries to check if values have expected ranges, if values are NULL or not, and so on. If there is a problem with the data, you will need to re-run earlier stages of the ETL process, or do an INSERT ... SELECT statement in Impala to copy the faulty data to a new table and transform or filter out the bad values.
- Take out any CREATE INDEX, DROP INDEX, and ALTER INDEX statements, and equivalent ALTER TABLE statements. Remove any INDEX, KEY, or PRIMARY KEY clauses from CREATE TABLE and ALTER TABLE statements. Impala is optimized for bulk read operations for data warehouse-style queries, and therefore does not support indexes for its tables.
- Calls to built-in functions with out-of-range or otherwise incorrect arguments, return NULL in Impala as opposed to raising exceptions. (This rule applies even when the ABORT_ON_ERROR=true query option is in effect.) Run small-scale queries using representative data to doublecheck that calls to built-in functions are returning expected values rather than NULL. For example, unsupported CAST operations do not raise an error in Impala:
- For any other type not supported in Impala, you could represent their values in string format and write UDFs to process them.
- To detect the presence of unsupported or unconvertable data types in data files, do initial testing with the ABORT_ON_ERROR=true query option in effect. This option causes queries to fail immediately if they encounter disallowed type conversions.

SQL statements to remove or adapt

The following SQL statements or clauses are not currently supported or supported with limitations in Impala:

- Impala supports the DELETE statement only for Kudu tables.

  Impala is intended for data warehouse-style operations where you do bulk moves and transforms of large quantities of data. When not using Kudu tables, instead of DELETE, use INSERT OVERWRITE to entirely replace the contents of a table or partition, or use INSERT ... SELECT to copy a subset of data (everything but the rows you intended to delete) from one table to another.
- Impala supports the UPDATE statement only for Kudu tables.

  When not using Kudu tables, instead of UPDATE, do all necessary transformations early in the ETL process, such as in the job that generates the original data, or when copying from one table to another to convert to a particular file format or partitioning scheme.
- Impala has no transactional statements, such as COMMIT or ROLLBACK.

  Impala effectively works like the AUTOCOMMIT mode in some database systems, where changes take effect as soon as they are made.
- If your database, table, column, or other names conflict with Impala reserved words, use different names or quote the names with backticks.

  Conversely, if you use a keyword that Impala does not recognize, it might be interpreted as a table or column alias.

  For example, in SELECT * FROM t1 NATURAL JOIN t2, Impala does not recognize the NATURAL keyword and interprets it as an alias for the table t1. If you experience any unexpected behavior with queries, check the list of reserved words to make sure all keywords in join and WHERE clauses are supported keywords in Impala.
- Impala has some restrictions on subquery support.

- Impala supports UNION and UNION ALL set operators, but not INTERSECT.

  Prefer UNION ALL over UNION when you know the data sets are disjoint or duplicate values are not a problem; UNION ALL is more efficient because it avoids materializing and sorting the entire result set to eliminate duplicate values.
- Impala requires query aliases for the subqueries used as inline views in the FROM clause.

  For example, without the alias contents_of_t1 at the end, the following query gives a syntax error:

  ```
  SELECT COUNT(*) FROM (SELECT * FROM t1) contents_of_t1;
  ```

  Aliases are not required for the subqueries used in other parts of queries. For example:

  ```
  SELECT * FROM functional.alltypes WHERE id = (SELECT MIN(id) FROM functi
  onal.alltypes);
  ```
- When an alias is declared for an expression in a query, that alias cannot be referenced again within the same SELE CT list.

  For example, the average alias cannot be referenced twice in the SELECT list as below. You will receive an error:

  ```
  SELECT AVG(x) AS average, average+1 FROM t1 GROUP BY x;
  ```

  An alias can be referenced again in the same query if not in the SELECT list. For example, the average alias can be referenced twice as shown below:

  ```
  SELECT AVG(x) AS average FROM t1 GROUP BY x HAVING average > 3;
  ```
- Impala does not support NATURAL JOIN, and it does not support the USING clause in joins.
- Impala supports a limited choice of partitioning types.

  Partitions are defined based on each distinct combination of values for one or more partition key columns. Impala does not redistribute or check data to create evenly distributed partitions. You must choose partition key columns based on your knowledge of the data volume and distribution. Adapt any tables that use range, list, hash, or key partitioning to use the Impala partition syntax for CREATE TABLE and ALTER TABLE statements.

  Impala partitioning is similar to range partitioning where every range has exactly one value, or key partitioning where the hash function produces a separate bucket for every combination of key values.

  Note: Because the number of separate partitions is potentially higher than in other database systems, keep a close eye on the number of partitions and the volume of data in each one; scale back the number of partition key columns if you end up with too many partitions with a small volume of data in each one.

  To distribute work for a query across a cluster, you need at least one HDFS block per node. HDFS blocks are typically multiple megabytes, especially for Parquet files. Therefore, if each partition holds only a few megabytes of data, you are unlikely to see much parallelism in the query because such a small amount of data is typically processed by a single node.
- For the “top-N” queries, Impala uses the LIMIT clause rather than comparing against a pseudo column named ROWNUM or ROW_NUM.

SQL constructs to double-check

Some SQL constructs that are supported have behavior or defaults more oriented towards convenience than optimal performance. Also, sometimes machine-generated SQL, perhaps issued through JDBC or ODBC applications, might have inefficiencies or exceed internal Impala limits. As you port SQL code, examine and possibly update the following where appropriate:

- A CREATE TABLE statement with no STORED AS clause creates data files in plain text format, which is convenient for data interchange but not a good choice for high-volume data with high-performance queries.

- Adapting tables that were already partitioned in a different database system could produce an Impala table with a high number of partitions and not enough data in each one, leading to underutilization of Impala's parallel query features.
- The INSERT ... VALUES syntax is suitable for setting up toy tables with a few rows for functional testing when used with HDFS. Each such statement creates a separate tiny file in HDFS, and it is not a scalable technique for loading megabytes or gigabytes (let alone petabytes) of data.

  Consider revising your data load process to produce raw data files outside of Impala, then setting up Impala external tables or using the LOAD DATA statement to use those data files instantly in Impala tables, with no conversion or indexing stage.

  INSERT works fine for Kudu tables even though not particularly fast.
- If your ETL process is not optimized for Hadoop, you might end up with highly fragmented small data files, or a single giant data file that cannot take advantage of distributed parallel queries or partitioning. In this case, use an INSERT ... SELECT statement to copy the data into a new table and reorganize into a more efficient layout in the same operation.

  You can do INSERT ... SELECT into a table with a more efficient file format or from an unpartitioned table into a partitioned one.
- Complex queries may have high codegen time. As a workaround, set the query option DISABLE_CODEGEN= true if queries fail for this reason.
- If practical, rewrite UNION queries to use the UNION ALL operator instead. Prefer UNION ALL over UNION when you know the data sets are disjoint or duplicate values are not a problem; UNION ALL is more efficient because it avoids materializing and sorting the entire result set to eliminate duplicate values.

Next porting steps after verifying syntax and semantics

Some of the decisions you make during the porting process can have an impact on performance. After your SQL code is ported and working correctly, examine the performance-related aspects of your schema design, physical layout, and queries to make sure that the ported application is taking full advantage of Impala's parallelism, performance-related SQL features, and integration with Hadoop components. The following are a few of the areas you should examine:

- For the optimal performance, we recommend that you run COMPUTE STATS on all tables.
- Use the most efficient file format for your data volumes, table structure, and query characteristics.
- Partition on columns that are often used for filtering in WHERE clauses.
- Your ETL process should produce a relatively small number of multi-megabyte data files rather than a huge number of small files.

Related Information

Impala conditional functions

ABORT_ON_ERROR query option

How to use specific file formats for compact data and high-performance queries

Most heavily optimized file format for large-scale data warehouse queries

UTF-8 Support

Impala has traditionally offered a single-byte binary character set for STRING data type with the character data encoded in an ASCII character set. This release provides a UTF-8 aware behavior for Impala STRING type to get consistent behavior with Hive on UTF-8 strings using a query option.

Turning ON the UTF-8 behavior

Before this release, Impala was incompatible with Hive in some functions applying on non-ASCII strings. For example, the length() function in Impala returns the length of bytes of the string, while the length() function in Hive

returns the length of UTF-8 characters of the string. UTF-8 characters (code points) assemble in variant-length bytes (1~4 bytes), so the results differ for non-ASCII characters in the string.

UTF-8 support enables you to read and write UTF-8 from standard formats like Parquet and ORC, thus improving interoperability with other engines supporting those standard formats.

You can use the new query option, UTF8_MODE, to enable and disable the UTF-8 aware behavior. You can set it globally or at a session level. Only queries with UTF8_MODE=true will have UTF-8-related behaviors.

Note:

- If you enable the UTF8_MODE query option globally, you must explicitly set UTF8_MODE=false for the existing queries that depend on the past binary behavior.
- Impala Daemons should be deployed on nodes using the same Glibc version since different Glibc version supports different Unicode standard version and also ensure that the en_US.UTF-8 locale is installed in the nodes. Not using the same Glibc version might result in inconsistent UTF-8 behavior when UTF8_MODE is set to true.

List of STRING functions

The new query option enables the UTF-8 aware behavior of the following string functions:

- LENGTH(STRING a)
  - returns the number of UTF-8 characters instead of bytes
- SUBSTR(STRING a, INT start [, INT len])
- SUBSTRING(STRING a, INT start [, INT len])()
  - the substring start position and length are counted as UTF-8 characters instead of bytes
- REVERSE(STRING a)
  - the unit of the operation is a UTF-8 character. That is, it does not reverse bytes inside a UTF-8 character.

    Note: The results of reverse("###SQL##") used to be "####LQS#####" and now "##LQS###".
- INSTR(STRING str, STRING substr[, BIGINT position[, BIGINT occurrence]])
- LOCATE(STRING substr, STRING str[, INT pos])
  - These functions have an optional position argument. The return values are also positions in the string. In UTF-8 mode, these positions are counted as UTF-8 characters instead of bytes.
- mask functions
  - The unit of the operation is a UTF-8 character. That is, they do not mask the string byte-to-byte.
- upper/lower/initcap
  - These functions recognize non-ascii characters and transform them based on the current locale used by the Impala process.

Limitations

- Use the UTF8_MODE option only when needed because the performance of UTF-8 is not optimized yet. It is an experimental feature.
- UTF-8 support for CHAR and VARCHAR types is not implemented yet. So VARCHAR(N) returns N bytes instead of N UTF-8 characters.

Cloudera Runtime 7.3.1

Starting and Stopping Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Modifying Impala Startup Options....4
Configuring Client Access to Impala....4
Impala Startup Options for Client Connections....5
Impala Shell Tool....6
Impala Shell Configuration Options....6
Impala Shell Configuration File....9
Connecting to Impala Daemon in Impala Shell....11
Running Commands and SQL Statements in Impala Shell....12
Impala Shell Command Reference....13
Configuring ODBC for Impala....14
Configuring JDBC for Impala....14
Configuring Impyla for Impala....16
Connecting to Cloudera Data Hub....18
Connecting to Cloudera Data Hub Data Mart....19
Configuring Delegation for Clients....19
Spooling Impala Query Results....20
Shut Down Impala....21
Setting Timeouts in Impala....22
Setting Timeout and Retries for Thrift Connections to Backend Client....22
Increasing StateStore Timeout....23
Setting the Idle Query and Idle Session Timeouts....23
Adjusting Heartbeat TCP Timeout Interval....24

Modifying Impala Startup Options

You can view and edit the configuration options for the Impala daemons to customize your Impala environment, such as to specify which hosts and ports to use, to assign directories for logging,and to control resource usage and security.

Configuring Impala Startup Options

Navigate to the following page to configure the settings for all the Impala-related daemons:

ClustersImpalaConfiguration.

If the Cloudera Manager interface does not yet have a form field for a newly added option, or if you need to use special options for debugging and troubleshooting, the Advanced category page for each daemon includes one or more Safety Valve fields where you can enter option names directly.

Checking the Values of Impala Startup Options

You can check the current runtime value of all these settings through the Impala Web UI, available by default at:

- http://*IMPALA_HOSTNAME*:25000/varz for the `impalad` daemon
- http://*IMPALA_HOSTNAME*:25010/varz for the `statestored` daemon
- http://*IMPALA_HOSTNAME*:25020/varz for the `catalogd` daemon

Related Information

Web User Interface for Debugging

Configuring Client Access to Impala

Application developers have a number of options to interface with Impala.

The core development language with Impala is SQL. You can also use Java or other languages to interact with Impala through the standard JDBC and ODBC interfaces used by many business intelligence tools. For specialized kinds of analysis, you can supplement the Impala built-in functions by writing user-defined functions in C++ or Java.

You can connect and submit requests to the Impala through:

- The impala-shell interactive command interpreter
- The Hue web-based user interface
- JDBC
- ODBC
- Impyla

Impala clients can connect to any Coordinator Impala Daemon (impalad) via HiveServer2 over HTTP or over the TCP binary or via Beeswax. All interfaces support Kerberos and LDAP for authentication to Impala. See below for the default ports and the Impala configuration field names to change the ports in Cloudera Manager.

Protocol	Default Port	Cloudera Manager Field to Specify an Alternate Port
HiveServer2 HTTP	28000	Impala Daemon HiveServer2 HTTP Port
HiveServer2 binary TCP	21050	Impala Daemon HiveServer2 Port
Beeswax	21000	Impala Daemon Beeswax Port

Impala Startup Options for Client Connections

Use the following flags to control client connections to Impala when starting Impala Daemon coordinator. If a configuration field exists in Cloudera Manager, the field name is shown in parenthesis next to the flag name.

If the Cloudera Manager interface does not yet have a form field for an option, the Advanced category page for each daemon includes one or more Safety Valve fields where you can enter option names directly.

--accepted_client_cnxn_timeout

Controls how Impala treats new connection requests if it has run out of the number of threads configured by --fe_service_threads.

If --accepted_client_cnxn_timeout > 0, new connection requests are rejected if Impala can't get a server thread within the specified (in seconds) timeout.

If --accepted_client_cnxn_timeout=0, i.e. no timeout, clients wait indefinitely to open the new session until more threads are available.

The default timeout is 5 minutes.

The timeout applies only to client facing thrift servers, i.e., HS2 and Beeswax servers.

--disconnected_session_timeout

When a HiveServer2 session has had no open connections for longer than this value, the session will be closed, and any associated queries will be unregistered.

Specify the value in hours.

The default value is 1 hour.

This flag does not apply to Beeswax clients. When a Beeswax client connection is closed, Impala closes the session associated with that connection.

--fe_service_threads (Impala Daemon Max Client)

Specifies the maximum number of concurrent client connections allowed. The default value is 64 with which 64 queries can run simultaneously.

If you have more clients trying to connect to Impala than the value of this setting, the later arriving clients have to wait for the duration specified by --accepted_client_cnxn_timeout. You can increase this value to allow more client connections. However, a large value means more threads to be maintained even if most of the connections are idle, and it could negatively impact query latency. Client applications should use the connection pool to avoid need for large number of sessions.

--idle_client_poll_time_s

The value of this setting specifies how frequently Impala polls to check if a client connection is idle and closes it if the connection is idle. A client connection is idle if all sessions associated with the client connection are idle.

By default, --idle_client_poll_time_s is set to 30 seconds.

If --idle_client_poll_time_s is set to 0, idle client connections stay open until explicitly closed by the clients.

The connection will only be closed if all the associated sessions are idle or closed. Sessions cannot be idle unless either the flag --idle_session_timeout or the IDLE_SESSION_TIMEOUT query option is set to greater than 0. If idle session timeout is not configured, a session cannot become idle by definition, and therefore its connection stays open until the client explicitly closes it.

--max_cookie_lifetime_s

Impala uses cookies for authentication when clients connect via HiveServer2 over HTTP. Use the --max_cookie_lifetime_s startup flag to control how long generated cookies are valid for.

Specify the value in seconds.

The default value is 1 day.

Setting the flag to 0 disables cookie support.

When an unexpired cookie is successfully verified, the user name contained in the cookie is set on the connection.

Each impalad uses its own key to generate the signature, so clients that reconnect to a different impa lad have to re-authenticate.

On a single impalad, cookies are valid across sessions and connections.

--beeswax_port (Impala Daemon Beeswax Port)

Specifies the port for clients to connect to Impala daemon via the Beeswax protocol.

You can disable the Beeswax end point for clients by setting the flag to 0.

--hs2_http_port (Impala Daemon HiveServer2 HTTP Port)

Specifies the port for clients to connect to Impala daemon over HTTP.

You can disable the HTTP end point for clients by setting the flag to 0.

To enable TLS/SSL for HiveServer2 HTTP endpoint, use --ssl_server_certificate and --ssl_privat e_key.

--hs2_port (Impala Daemon HiveServer2 Port)

Specifies the port for clients to connect to Impala daemon via the HiveServer2 protocol.

You can disable the binary HiveServer2 end point for clients by setting the flag to 0.

--ping_expose_webserver_url

Controls whether PingImpalaService, PingImpalaHS2Service RPC calls should expose the debug web url to the client or not.

By default this flag is set to true so that the users will not see an empty string when the debug web UI is not available. If the flag is set to false, the RPC calls will return an empty string instead of the real url signalling that the debug web UI is not available.

Note: If the webserver is disabled (--enable_webserver flag is set to false) the RPC calls will return an empty string for the url.

Impala Shell Tool

You can use the Impala shell tool (impala-shell) to set up databases and tables, insert data, and issue queries.

For ad-hoc queries and exploration, you can submit SQL statements in an interactive session. To automate your work, you can specify command-line options to process a single statement or a script file. The `impala-shell` accepts all the same SQL statements, plus some shell-only commands that you can use for tuning performance and diagnosing problems.

Cloudera Manager installs `impala-shell` automatically. You might install `impala-shell` manually on other systems not managed by Cloudera Manager, so that you can issue queries from client systems that are not also running the Impala daemon or other Apache Hadoop components.

Impala Shell Configuration Options

You can specify the following options when starting impala-shell to control how shell commands are executed. You can specify options on the command line or in the impala-shell configuration file.

Command-Line Option	Configuration File Setting	Explanation
-B or --delimited	write_delimited=true	Causes all query results to be printed in plain format as a delimited text file. Useful for producing data files to be used with other Hadoop components. Also useful for avoiding the performance overhead of pretty-printing all output, especially when running benchmark tests using queries returning large result sets. Specify the delimiter character with the --output_delimiter option. Store all query results in a file rather than printing to the screen with the -B option.
-E or --vertical	vertical=true	Causes all query results to print in vertical format. In this mode, impala-shell will print each row in the following style. • The first line will contain the line number followed by the row's columns in separate lines. Each column line will have a prefix with its name and a colon. To enable this mode, use the shell option '-E' or '--vertical', or 'set VERTICAL= true' in the interactive mode. To disable it in interactive mode, 'set VERTICAL=false'. NOTE: This vertical option will be disabled if the '-B' option or 'set WRITE_DELIMITED=true' is specified.
--connect_max_tries	--connect_max_tries=4	Sets the maximum number of attempts to connect to a coordinator. Currently, the maximum number of attempts to connect to a coordinator is hard coded to 4 in hs2-HTTP mode. From this release, you can configure the maximum number of attempts impala-shell can make using this option --connect_max_tries. The default value of this option is 4. After the number of attempts specified through this option, impala-shell returns with error "Not connected to Impala"
--hs2_fp_format	hs2_fp_format=<a Python-based format specification expression which will get parsed and applied to floating-pointcolumn values>	Sets the printing format specification for floating point values when using HS2 protocol. The default behaviour makes the values handled by Python's str() built-in method. Use '16G' to match Beeswax protocol's floating-point output format.
--http_cookie_names	http_cookie_names=KNOX_BACKEND=IMPALA	This parameter has to be set to include the cookie that Knox uses for stateful connections. This is what the Knox enableStickySession=true uses. When using impala-shell and http mode, you must update the client connection string to include the cookie being used for session persistence using the `-http_cookie_names`. This config is needed for Active-Active HA in Impala.
--live_progress	live_progress=true	Prints a progress bar showing roughly the percentage complete for each query. Information is updated interactively as the query progresses.
--disable_live_progress	live_progress=false	Disables live_progress in the interactive mode.
-b or --kerberos_host_fqdn	kerberos_host_fqdn= *LOAD-BALANCER-HOSTNAME*	If set, the setting overrides the expected hostname of the Impala daemon's Kerberos service principal. `impala-shell` will check that the server's principal matches this hostname. This may be used when impalad is configured to be accessed via a load-balancer, but it is desired for impala-shell to talk to a specific impalad directly.
--print_header	print_header=true	
-o *FILENAME* or --output_file *FILENAME*	output_file=*FILENAME*	Stores all query results in the specified file. Typically used to store the results of a single query issued from the command line with the -q option. Also works for interactive sessions; you see the messages such as number of rows fetched, but not the actual result set. To suppress these incidental messages when combining the -q and -o options, redirect stderr to /dev/null.
--output_delimiter= *CHARACTER*	output_delimiter=*CHARACTER*	Specifies the character to use as a delimiter between fields when query results are printed in plain format by the -B option. Defaults to tab ('\t'). If an output value contains the delimiter character, that field is quoted, escaped by doubling quotation marks, or both.
-p or --show_profiles	show_profiles=true	Displays the query execution plan (same output as the EXPLAIN statement) and a more detailed low-level breakdown of execution steps, for every query executed by the shell.

Command-Line Option	Configuration File Setting	Explanation
--profile_format= text | json | prettyjson	N/A	json and prettyjson output the JSON representation of each profile in a dense single-line form and in a human-readable multi-line form respectively.
-h or --help	N/A	Displays help information.
N/A	history_max=1000	Sets the maximum number of queries to store in the history file.
-i *HOSTNAME* or --impalad=*HOSTNAME*[:*PORTNUM*]	impalad=*HOSTNAME*[:*PORTNUM*]	Connects to the impalad daemon on the specified host. The default port of 21050 is assumed unless you provide another value. You can connect to any host in your cluster that is running impalad. If you connect to an instance of impalad that was started with an alternate port specified by the --fe_port flag, provide that alternative port.
-q *QUERY* or --query=*QUERY*	query=*QUERY*	Passes a query or other `impala-shell` command from the command line. The `impala-shell` interpreter immediately exits after processing the statement. It is limited to a single statement, which could be a SELECT, CREATE TABLE, SHOW TABLES, or any other statement recognized in impala-shell. Because you cannot pass a USE statement and another query, fully qualify the names for any tables outside the default database. (Or use the -f option to pass a file with a USE statement followed by other queries.)
-f *QUERY_FILE* or --query_file=*QUERY_FILE*	query_file=*PATH_TO_QUERY_FILE*	Passes a SQL query from a file. Multiple statements must be semicolon (;) delimited.
-k or --kerberos	use_kerberos=true	Kerberos authentication is used when the shell connects to impalad. If Kerberos is not enabled on the instance of impalad to which you are connecting, errors are displayed.
--query_option= "*OPTION=VALUE* -Q "*OPTION=VALUE*"	Header line [impala.query_options], followed on subsequent lines by *OPTION=VALUE*, one option per line.	Sets default query options for an invocation of the `impala-shell` command. To set multiple query options at once, use more than one instance of this command-line option. The query option names are not case-sensitive.
-s *KERBEROS_SERVICE_NAME* or --kerberos_service_name= *NAME*	kerberos_service_name=*NAME*	Instructs impala-shell to authenticate to a particular impalad service principal. If a *KERBEROS_SERVICE_NAME* is not specified, impala is used by default. If this option is used in conjunction with a connection in which Kerberos is not supported, errors are returned.
-V or --verbose	verbose=true	Enables verbose output.
--quiet	verbose=false	Disables verbose output.
-v or --version	version=true	Displays version information.
-c	ignore_query_failure=true	Continues on query failure.
-d *DEFAULT_DB* or --database=*DEFAULT_DB*	default_db=*DEFAULT_DB*	Specifies the database to be used on startup. Same as running the USE statement after connecting. If not specified, a database named DEFAULT is used.
--ssl	ssl=true	Enables TLS/SSL for `impala-shell`.
--http_socket_timeout_s	http_socket_timeout_s=*HTTP_SOCKET_TIMEOUT_S*	Sets the timeout in seconds after which the socket will time out if the associated operation cannot be completed. Set to None to disable any timeout. This configurable option is only supported for hs2-http mode and the DEFAULT is NONE.

Command-Line Option	Configuration File Setting	Explanation
PATH_TO_CERTIFICATE	ca_cert=*PATH_TO_CERTIFICATE*	The local pathname pointing to the third-party CA certificate, or to a copy of the server certificate for self-signed server certificates. If --ca_cert is not set, `impala-shell` enables TLS/SSL, but does not validate the server certificate. This is useful for connecting to a known-good Impala that is only running over TLS/SSL, when a copy of the certificate is not available (such as when debugging customer installations).
-l	use_ldap=true	Enables LDAP authentication.
-u	user=*USER_NAME*	Supplies the username, when LDAP authentication is enabled by the -l option. (Specify the short username, not the full LDAP distinguished name.) The shell then prompts interactively for the password.
--ldap_password_cmd= *COMMAND*	N/A	Specifies a command to run to retrieve the LDAP password, when LDAP authentication is enabled by the -l option. If the command includes space-separated arguments, enclose the command and its arguments in quotation marks.
-j, --jwt	N/A	Indicates that JWT authentication will be used when this command line option is specified in the impala-shell command.
--jwt_cmd	N/A	Shell command to run to retrieve the JWT to be used for authentication.
--config_file= *PATH_TO_CONFIG_FILE*	N/A	Specifies the path of the file containing `impala-shell` configuration settings. The default is /etc/impalarc. This setting can only be specified on the command line.
--live_progress	N/A	Prints a progress bar showing roughly the percentage complete for each query. The information is updated interactively as the query progresses.
--live_summary	N/A	Prints a detailed report, similar to the SUMMARY command, showing progress details for each phase of query execution. The information is updated interactively as the query progresses.
--var= *VARIABLE_NAME*= *VALUE*	N/A	Defines a substitution variable that can be used within the `impala-shell` session. The variable can be substituted into statements processed by the -q or -f options, or in an interactive shell session. Within a SQL statement, you substitute the value by using the notation ${var:*VARIABLE_NAME*}.
--auth_creds_ok_in_clear	N/A	Allows LDAP authentication to be used with an insecure connection to the shell. WARNING: This will allow authentication credentials to be sent unencrypted, and hence may be vulnerable to an attack.
--protocol= *PROTOCOL*	N/A	Protocol to use for the connection to Impala. Valid *PROTOCOL* values are: • 'hs2': Impala-shell uses the binary TCP based transport to speak to the Impala Daemon via the HiveServer2 protocol. This is the current default setting. • 'hs2-http': Impala-shell uses HTTP transport to speak to the Impala Daemon via the HiveServer2 protocol. • 'beeswax': Impala-shell uses the binary TCP based transport to speak to the Impala Daemon via Beeswax. You cannot connect to the 3.2 or earlier versions of Impala using the 'hs2' or 'hs2-http' option. Beeswax support is deprecated and will be removed in the future.

Impala Shell Configuration File

You can store a set of default settings for `impala-shell` in the `impala-shell` configuration file.

The global `impala-shell` configuration file is located in /etc/impalarc.

The user-level `impala-shell` configuration file is located in ~/.impalarc.

Note that the global-level file name is different from the user-level file name. The global-level file name does not include a dot (.) in the file name.

The default path of the global configuration file can be changed by setting the $IMPALA_SHELL_GLOBAL_CONFIG_FILE environment variable.

To specify a different file name or path for the user-level configuration file, start `impala-shell` with the --config_file `impala-shell` option set to the path of the configuration file.

Typically, an administrator creates the global configuration file for the `impala-shell`, and if the user-level configuration file exists, the options set in the user configuration file take precedence over those in the global configuration file.

In turn, any options you specify on the `impala-shell` command line override any corresponding options within the configuration file.

The `impala-shell` configuration file (global or user) must contain a header label [impala], followed by the options specific to `impala-shell`.

The `impala-shell` configuration file consists of key-value pairs, one option per line. Everything after the # character on a line is treated as a comment and ignored.

The names of the options in the configuration file are similar (although not necessarily identical) to the long-form command-line arguments to the `impala-shell` command. For the supported options in the configuration file, see Impala Shell Configuration Options on page 6.

You can specify key-value pair options using keyval, similar to the --var command-line option. For example, keyval=*VARIABLE1=VALUE1*.

The query options specified in the [impala] section override the options specified in the [impala.query_options] section.

The following example shows a configuration file that you might use during benchmarking tests. It sets verbose mode, so that the output from each SQL query is followed by timing information. `impala-shell` starts inside the database containing the tables with the benchmark data, avoiding the need to issue a USE statement or use fully qualified table names.

In this example, the query output is formatted as delimited text rather than enclosed in ASCII art boxes, and is stored in a file rather than printed to the screen. Those options are appropriate for benchmark situations, so that the overhead of `impala-shell` formatting and printing the result set does not factor into the timing measurements. It also enables the show_profiles option. That option prints detailed performance information after each query, which might be valuable in understanding the performance of benchmark queries.

```
[impala]
verbose=true
default_db=tpc_benchmarking
write_delimited=true
output_delimiter=,
output_file=/home/tester1/benchmark_results.csv
show_profiles=true
keyval=msg1=hello,keyval=msg2=world
```

The following example shows a configuration file that connects to a specific remote Impala node, runs a single query within a particular database, then exits. Any query options predefined under the [impala.query_options] section in the configuration file take effect during the session.

You would typically use this kind of single-purpose configuration setting with the `impala-shell` command-line option --config_file=*PATH_TO_CONFIG_FILE*, to easily select between many predefined queries that could be run against different databases, hosts, or even different clusters. To run a sequence of statements instead of a single query, specify the configuration option query_file=*PATH_TO_QUERY_FILE* instead.

```
[impala]
impalad=impala-test-node1.example.com
```

```
default_db=site_stats
# Issue a predefined query and immediately exit.
query=select count(*) from web_traffic where event_date = trunc(now(),'dd')

[impala.query_options]
mem_limit=32g
```

Connecting to Impala Daemon in Impala Shell

In an `impala-shell` session, you need to connect to an `impalad` daemon to issue queries. When you connect to an impalad, and that daemon coordinates the execution of all queries sent to it.

About this task

Specify the connection information using the following options:

- Through command-line options when you run the `impala-shell` command.
- Through a configuration file that is read when you run the `impala-shell` command.
- During an `impala-shell` session, by issuing a CONNECT command.

Note: You cannot connect to the 3.2 or earlier versions of Impala using the 'hs2' or 'hs2-http' protocol (--pr otocol option).

To connect the Impala shell during shell startup:

1. Locate the hostname that is running an instance of the `impalad` daemon. If that `impalad` uses a non-default port (something other than port 21050) for `impala-shell` connections, find out the port number also.
2. Use the -i option to the `impala-shell` interpreter to specify the connection information for that instance of `impalad`:

```
# When you are connecting to an impalad running on the same machine.
# The prompt will reflect the current hostname.
$ impala-shell
# When you are connecting to an impalad running on a remote machine, and
 impalad is listening
# on a non-default port over the HTTP HiveServer2 protocol.
$ impala-shell -i SOME.OTHER.HOSTNAME:PORT_NUMBER --protocol='hs2-http'
# When you are connecting to an impalad running on a remote machine, and
impalad is listening
# on a non-default port.
$ impala-shell -i SOME.OTHER.HOSTNAME:PORT_NUMBER
```

To connect to an Impala in the`impala-shell` session:

1. Start the Impala shell with no connection:

```
impala-shell
```

2. Locate the hostname that is running the `impalad` daemon. If that `impalad` uses a non-default port (something other than port 21050) for `impala-shell` connections, find out the port number also.
3. Use the connect command to connect to an Impala instance. Enter a command and replace *IMPALAD-HOST* with the hostname you have configured to run Impala in your environment.

```
[Not connected] > connect IMPALAD-HOST
[IMPALAD-HOST:21050] >
```

To start `impala-shell` in a specific database:

You can use all the same connection options as in previous examples. For simplicity, these examples assume that you are logged into one of the Impala daemons.

1. Find the name of the database containing the relevant tables, views, and so on that you want to operate on.

2. Use the -d option to the `impala-shell` interpreter to connect and immediately switch to the specified database, without the need for a USE statement or fully qualified names:

```
# Subsequent queries with unqualified names operate on
# tables, views, and so on inside the database named 'staging'.
$ impala-shell -i localhost -d staging

# It is common during development, ETL, benchmarking, and so on
# to have different databases containing the same table names
# but with different contents or layouts.
$ impala-shell -i localhost -d parquet_snappy_compression
$ impala-shell -i localhost -d parquet_gzip_compression
```

To run one or several statements in non-interactive mode:

You can use all the same connection options as in previous examples. For simplicity, these examples assume that you are logged into one of the Impala daemons.

1. Construct a statement, or a file containing a sequence of statements, that you want to run in an automated way, without typing or copying and pasting each time.
2. Invoke `impala-shell` with the -q option to run a single statement, or the -f option to run a sequence of statements from a file. The `impala-shell` command returns immediately, without going into the interactive interpreter.

```
# A utility command that you might run while developing shell scripts
# to manipulate HDFS files.
$ impala-shell -i localhost -d database_of_interest -q 'show tables'

# A sequence of CREATE TABLE, CREATE VIEW, and similar DDL statements
# can go into a file to make the setup process repeatable.
$ impala-shell -i localhost -d database_of_interest -f recreate_tables.sql
```

Running Commands and SQL Statements in Impala Shell

This topic provides the commonly used syntax and shortcut keys in impala-shell.

The following are a few of the key syntax and usage rules for running commands and SQL statements in impala-shell.

- To see the full set of available commands, press TAB twice.
- To cycle through and edit previous commands, click the up-arrow and down-arrow keys.
- Use the standard set of keyboard shortcuts in GNU Readline library for editing and cursor movement, such as Ctrl-A for the beginning of line and Ctrl-E for the end of line.
- Commands and SQL statements must be terminated by a semi-colon.
- Commands and SQL statements can span multiple lines.
- Use -- to denote a single-line comment and /* */ to denote a multi-line comment.

  A comment is considered part of the statement it precedes, so when you enter a -- or /* */ comment, you get a continuation prompt until you finish entering a statement ending with a semicolon. For example:

```
[impala] > -- This is a test comment
                  > SHOW TABLES LIKE 't*';
```

- If a comment contains the ${*VARIABLE_NAME*} and it is not for a variable substitution, the $ character must be escaped, e.g. -- \${hello}.

Variable Substitution in impala-shell

You can define substitution variables to be used within SQL statements processed by `impala-shell`.

1. You specify the variable and its value as below.
   - On the command line, you specify the option --var=*VARIABLE_NAME=VALUE*
   - Within an interactive session or a script file processed by the -f option, use the SET VAR:*VARIABLE_NAME=VALUE* command.
2. Use the above variable in SQL statements in the impala-shell session using the notation: ${VAR:*VARIABLE_NAME*}.

For example, here are some `impala-shell` commands that define substitution variables and then use them in SQL statements executed through the -q and -f options. Notice how the -q argument strings are single-quoted to prevent shell expansion of the ${var:value} notation, and any string literals within the queries are enclosed by double quotation marks.

```
$ impala-shell --var=tname=table1 --var=colname=x --var=coltype=string -q '
CREATE TABLE ${var:tname} (${var:colname} ${var:coltype}) STORED AS PARQUET'
Query: CREATE TABLE table1 (x STRING) STORED AS PARQUET
```

The below example shows a substitution variable passed in by the --var option, and then referenced by statements issued interactively. Then the variable is reset with the SET command.

```
$ impala-shell --quiet --var=tname=table1

[impala] > SELECT COUNT(*) FROM ${var:tname};

[impala] > SET VAR:tname=table2;
[impala] > SELECT COUNT(*) FROM ${var:tname};
```

When you run a query, the live progress bar appears in the output of a query. The bar shows roughly the percentage of completed processing. When the query finishes, the live progress bar disappears from the console output.

Impala Shell Command Reference

Use the following commands within impala-shell to pass requests to the impalad daemon that the shell is connected to. You can enter a command interactively at the prompt or pass it as the argument to the -q option of impala-shell.

Command	Explanation
Impala SQL statements	You can issue valid SQL statements to be executed.
connect	Connects to the specified instance of impalad. The default port of 21050 is assumed unless you provide another value. You can connect to any host in your cluster that is running impalad. If you connect to an instance of impalad that was started with an alternate port specified by the --fe_port flag, you must provide that alternate port.
help	Help provides a list of all available commands and options.
history	Maintains an enumerated cross-session command history. This history is stored in the ~/.impalahistory file.
profile	Displays low-level information about the most recent query. Used for performance diagnosis and tuning. The report starts with the same information as produced by the EXPLAIN statement and the SUMMARY command.
quit	Exits the shell. Remember to include the final semicolon so that the shell recognizes the end of the command.
rerun or @	Executes a previous `impala-shell` command again, from the list of commands displayed by the history command. These could be SQL statements, or commands specific to `impala-shell` such as quit or profile. Specify an integer argument. A positive integer N represents the command labelled N in the output of the HISTORY command. A negative integer -N represents the Nth command from the end of the list, such as -1 for the most recent command. Commands that are executed again do not produce new entries in the HISTORY output list.

Command	Explanation
set	Manages query options for an `impala-shell` session. These options are used for query tuning and troubleshooting. Issue SET with no arguments to see the current query options, either based on the `impalad` defaults, as specified by you at `impalad` startup, or based on earlier SET statements in the same session. To modify option values, issue commands with the syntax set *OPTION=VALUE*. To restore an option to its default, use the unset command.
shell	Executes the specified command in the operating system shell without exiting impala-shell. You can use the ! character as shorthand for the shell command. **Note:** Quote any instances of the -- or /* tokens to avoid them being interpreted as the start of a comment. To embed comments within source or ! commands, use the shell comment character # before the comment portion of the line.
source or src	Executes one or more statements residing in a specified file from the local filesystem. Allows you to perform the same kinds of batch operations as with the -f option, but interactively within the interpreter. The file can contain SQL statements and other `impala-shell` commands, including additional SOURCE commands to perform a flexible sequence of actions. Each command or statement, except the last one in the file, must end with a semicolon.
summary	Summarizes the work performed in various stages of a query. It provides a higher-level view of the information displayed by the EXPLAIN command. Added in Impala 1.4.0. The time, memory usage, and so on reported by SUMMARY only include the portions of the statement that read data, not when data is written. Therefore, the PROFILE command is better for checking the performance and scalability of INSERT statements. You can see a continuously updated report of the summary information while a query is in progress.
unset	Removes any user-specified value for a query option and returns the option to its default value. You can also use it to remove user-specified substitution variables using the notation UNSET VA R:*VARIABLE_NAME*.
use	Indicates the database against which to run subsequent commands. Lets you avoid using fully qualified names when referring to tables in databases other than default. Not effective with the -q option, because that option only allows a single statement in the argument.
version	Returns Impala version information.

Configuring ODBC for Impala

Download and configure the ODBC driver to integrate your applications with Impala.

About this task

Impala has been tested with the Impala ODBC driver version 2.5.42, and Cloudera recommends that you use this version with the current version of Impala.

Procedure

1. Download and install an ODBC driver.
2. Configure the ODBC port.

   Versions 2.5 and 2.0 of the Cloudera ODBC Connector use the HiveServer2 protocol, corresponding to Impala port 21050.

   Version 1.x of the Cloudera ODBC Connector uses the original HiveServer1 protocol, corresponding to Impala port 21000.

Configuring JDBC for Impala

Download and configure the JDBC driver to access Impala from a Java program that you write, or a Business Intelligence or similar tool that uses JDBC to communicate with database products.

About this task

The following are the default ports that Impala server accepts JDBC connections through:

Protocol	Default Port	Flag to Specify an Alternate Port
HTTP	28000	##hs2_http_port
Binary TCP	21050	##hs2_port

Make sure the port for the protocol you are using is available for communication with clients, for example, that it is not blocked by firewall software.

If your JDBC client software connects to a different port, specify that alternative port number with the flag in the above table when starting the impalad.

Procedure

1. Configure the JDBC port.

   Impala server accepts JDBC connections through port 21050 by default. Make sure this port is available for communication with other hosts on your network, for example, that it is not blocked by firewall software. If your JDBC client software connects to a different port, specify that alternative port number in the Impala Daemon HiveServer2 Portfield in Cloudera Manager, in the Configuration tab.
2. Install the JDBC driver.

   Impala has been tested using the Impala JDBC driver version 2.5.45 and 2.6.2. Cloudera recommends that you use one of these two versions with Impala.
3. Enable Impala JDBC support on client systems.

4. Establish JDBC connections.

   The JDBC driver class depends on which driver you select.

   Using the Cloudera JDBC Connector (recommended):

   Depending on the level of the JDBC API your application is targeting, you can use the following fully-qualified class names (FQCNs):

   - com.cloudera.impala.jdbc41.Driver
   - com.cloudera.impala.jdbc41.DataSource
   - com.cloudera.impala.jdbc4.Driver
   - com.cloudera.impala.jdbc4.DataSource
   - com.cloudera.impala.jdbc3.Driver
   - com.cloudera.impala.jdbc3.DataSource

   The connection string has the following format:

   ```
   jdbc:impala://HOST:PORT[/SCHEMA];PROPERTY1=VALUE;PROPERTY2=VALUE;...
   ```

   The port value is typically 21050 for Impala.

   To connect to an instance of Impala that requires Kerberos authentication, use a connection string of the form jdbc:impala://*HOST*:*PORT*/;principal=*PRINCIPAL_NAME*. The principal must be the same user principal you used when starting Impala.

   To connect to an instance of Impala that requires LDAP authentication, use a connection string of the form jdbc:impala://*HOST*:*PORT*/*DB_NAME*;user=*LDAP_USERID*;password=*LDAP_PASSWORD*.

   To connect to an instance of Impala over HTTP, specify the HTTP port, 28000 by default, and transportMode=http in the connection string.

   Note: To establish a connection with an Impala instance from your client, using any authentication mode, you must use the connection string jdbc:impala. Using the connection string jdbc:hive2 is not recommended and is not supported.

   For updated information on the version of the JDBC driver you are using and for connection string examples for different supported authentications, refer to the link provided under Related Information.

Related Information

Cloudera Enterprise Connector Documentation

Configuring Impyla for Impala

Explains how to install Impyla to connect to and submit SQL queries to Impala. Impyla is a Python client wrapper around the HiveServer2 Thrift Service. It connects to Impala and implements Python DB API 2.0.

About this task

Impyla releases are available at pypi.org. To get the available releases, check Release history.

Note: Cloudera will not support versions of Impyla that are built manually from source code.

Key Features of Impyla

- HiveServer2 compliant.
- Works with Impala including nested data.
- DB API 2.0 (PEP 249)-compliant Python client (similar to sqlite or MySQL clients) supporting Python 2.6+ and Python 3.3+.

- Works with Kerberos, LDAP, SSL.
- SQLAlchemy connector.
- Converts to pandas DataFrame, allowing easy integration into the Python data stack (including scikit-learn and matplotlib); see the Ibis project for a richer experience.
- For more information, see here.

Before you begin

Different systems require different packages to be installed to enable SASL support in Impyla. The following list shows some examples of how to install the packages on different distributions.

You must have the following installed in your environment before installing impyla. Python 2.6+ or 3.3+ and the pip packages six, bitarray, thrift and thriftpy2 will be automatically installed as dependencies when installing impyla. However if you clone the impyla repo and run their local copy, you must install these pip packages manually.

- Install the latest pip and setuptools:

  ```
  python -m pip install --upgrade pip setuptools
  ```

  Optionally, to install Impyla with Hive and/or GSSAPI (kerberos) support, you will also need to install additional software packages:
- RHEL/CentOS:

  ```
  sudo yum install gcc-c++ cyrus-sasl-md5 cyrus-sasl-plain cyrus-sasl-gssapi
   cyrus-sasl-devel
  ```
- Ubuntu:

  ```
  sudo apt install g++ libsasl2-dev libsasl2-2 libsasl2-modules-gssapi-mit
  ```

Procedure

1. Using pip you can install the latest release: pip install impyla
2. Optionally, to install Impyla with GSSAPI (kerberos) support:pip install impyla[kerberos]
3. You also need to pip-install pandas for conversion to DataFrame objects or sqlalchemy for the SQLAlchemy engine.

Example

Sample codes

Impyla implements the Python DB API v2.0 (PEP 249) database interface (refer to it for API details):

```
from impala.dbapi import connect

conn = connect(host = "my.host.com", port = 21050)
cursor = conn.cursor()
cursor.execute("SELECT * FROM mytable LIMIT 100")
print(cursor.description)  # prints the result set's schema
results = cursor.fetchall()
cursor.close()
conn.close()
```

The Cursorobject also exposes the iterator interface, which is buffered (controlled by cursor.arraysize):

```
cursor.execute("SELECT * FROM mytable LIMIT 100")
for row in cursor:
    print(row)
```

Furthermore the Cursor object returns you information about the columns returned in the query. This is useful to export your data as a csv file.

```
import csv

cursor.execute("SELECT * FROM mytable LIMIT 100")
columns = [datum[0] for datum in cursor.description]
targetfile = "/tmp/foo.csv"

with open(targetfile, "w", newline = "") as outcsv:
    writer = csv.writer(
        outcsv,
        delimiter = ",",
        quotechar = '"',
        quoting = csv.QUOTE_ALL,
        lineterminator = "\n")
writer.writerow(columns)
for row in cursor:
    writer.writerow(row)
```

You can also get back a pandas DataFrame object

```
from impala.util import as_pandas

# carry df through scikit-learn, for example
df = as_pandas(cur)
```

Connecting over HTTP/HTTPS

GSSAPI (kerberos) authentication over HTTP has been supported since 0.17a1. Use this example to establish connection over HTTP/HTTPS.

```
from impala.dbapi import connect

conn = connect(
    "impala-coordinator.example.com",
    28000,
    auth_mechanism = "GSSAPI",
    kerberos_service_name = "impala",
    use_http_transport = True,
    http_path = "cliservice",
    auth_cookie_name = "impala.auth")
cursor = conn.cursor()
cursor.execute("SHOW DATABASES")
res = cursor.fetchall()
cursor.close()
conn.close()
```

Connecting to Cloudera Data Hub

Lists an example code to connect to Impala with LDAP over http using LDAP as the authentication mechanism.

Example

Sample code

```
from impala.dbapi import connect

conn = connect(
    host = "aaaaaaa-aaaa-master0.se-sandb.a465-9q4k.cloudera.site",
    port = 443,
    auth_mechanism = "LDAP",
    use_ssl = True,
```

```
    use_http_transport = True,
    http_path = "aaaaaaa-aaaa/cdp-proxy-api/impala",
    user = "aaaaaaa",
    password = "xxxxx")
cursor = conn.cursor()
cursor.execute("SELECT * FROM default.emax_temp")

for row in cursor:
    print(row)
cursor.close()
conn.close()
```

Connecting to Cloudera Data Hub Data Mart

Lists an example code to connect to Cloudera Data Hub Data Mart using LDAP as the authentication mechanism.

Before you begin

Must have the latest Impyla release.

Example

Sample code

```
from impala.dbapi import connect

conn = connect(
    "xxxxxxx-data-mart-master0.xxxxxxx.xcu2-8y8x.dev.cldr.work",
    443,
    auth_mechanism = "LDAP",
    user = "XXXXX",
    password = "XXXXX",
    use_ssl = True,
    use_http_transport = True,
    http_path = "xxxxxxx-data-mart/cdp-proxy-api/impala")
cursor = conn.cursor()
cursor.execute("SHOW DATABASES")
res = cursor.fetchall()
print(res)
cursor.close()
conn.close()
```

Configuring Delegation for Clients

Impala supports user and group delegation for client connections.

About this task

When users submit Impala queries through a separate application, such as Hue or a business intelligence tool, typically all requests are treated as coming from the same user. Impala supports “delegation” where users whose names you specify can delegate the execution of a query to another user. The query runs with the privileges of the delegated user, not the original authenticated user.

You also have an option to delegate using groups. Instead of listing a large number of delegated users, you can create a group of those users and specify the delegated group name in the Impalad startup option. The client sends the delegated user name, and Impala performs an authorization to see if the delegated user belongs to a delegated group.

The name of the delegated user is passed using the HiveServer2 protocol configuration property impala.doas.user when the client connects to Impala.

When the client connects over HTTP, the doAs parameter can be specified in the HTTP path. For example:

```
/?doAs=DELEGATED_USER
```

Currently, the delegation feature is available only for Impala queries submitted through application interfaces such as Hue and BI tools. For example, Impala cannot issue queries using the privileges of the HDFS user.

Attention:

- When the delegation is enabled in Impala, the Impala clients should take an extra caution to prevent unauthorized access for the delegate-able users.
- Impala requires Apache Ranger on the cluster to enable delegation. Without Ranger installed, the delegation feature will fail with the following error: User user1 is not authorized to delegate to user2. User/group delegation is disabled.

Procedure

To enable delegation:

1. In Cloudera Manager, navigate to Clusters Impala .
2. In the Configuration tab, click Impala-1 (Service-Wide) in the Scope and click Security in the Category.
3. In the Proxy User Configuration field, type the a semicolon-separated list of key=value pairs of authorized proxy users to the user(s) they can impersonate.

   The list of delegated users are delimited with a comma, e.g. hue=user1, user2.
4. In the Proxy Group Configuration field, type the a semicolon-separated list of key=value pairs of authorized proxy users to the group(s) they can impersonate.

   The list of delegated groups are delimited with a comma, e.g. hue=group1, group2.
5. Click Save Changes and restart the Impala service.

Spooling Impala Query Results

In Impala, you can control how query results are materialized and returned to clients, e.g. impala-shell, Hue, JDBC apps.

Result spooling is turned off by default, but can be enabled via the SPOOL_QUERY_RESULTS query option.

- When query result spooling is disabled, Impala relies on clients to fetch results to trigger the generation of more result row batches until all the result rows have been produced. If a client issues a query without fetching all the results then that query continues to be in the running state indefinitely and the query fragments continue to consume the resources until the query is cancelled and unregistered, potentially tying up resources and causing other queries to wait for an extended period of time in admission control.

  Impala would materialize rows on-demand where rows are created only when the client requests them.

  For example, if a Hue user runs a complex query that returns 1000 rows, but does not scroll through all the returned rows, and then stays idle for a while, the query will remain running and will hold onto all of its resources until it is explicitly closed or the session times out.
- When query result spooling is enabled, result sets of queries are eagerly fetched and spooled in the spooling location, either in memory or on disk.

  Once all result rows have been fetched and stored in the spooling location, the resources are freed up. Incoming client fetches can get the data from the spooled results.

Admission Control and Result Spooling

Query results spooling collects and stores query results in memory that is controlled by admission control. Use the following query options to calibrate how much memory to use and when to spill to disk.
MAX_RESULT_SPOOLING_MEM

The maximum amount of memory used when spooling query results. If this value is exceeded when spooling results, all memory will most likely be spilled to disk. Set to 100 MB by default.

MAX_SPILLED_RESULT_SPOOLING_MEM

The maximum amount of memory that can be spilled to disk when spooling query results. Must be greater than or equal to MAX_RESULT_SPOOLING_MEM. If this value is exceeded, the coordinator fragment will block until the client has consumed enough rows to free up more memory. Set to 1 GB by default.

Fetch Timeout

Resources for a query are released when the query completes its execution. To prevent clients from indefinitely waiting for query results, use the FETCH_ROWS_TIMEOUT_MS query option to set the timeout when clients fetch rows. Timeout applies both when query result spooling is enabled and disabled:

- When result spooling is disabled (SPOOL_QUERY_RESULTS = FALSE), the timeout controls how long a client waits for a single row batch to be produced by the coordinator.
- When result spooling is enabled ((SPOOL_QUERY_RESULTS = TRUE), a client can fetch multiple row batches at a time, so this timeout controls the total time a client waits for row batches to be produced.

Explain Plans

Below is the part of the EXPLAIN plan output for result spooling.

```
F01:PLAN FRAGMENT [UNPARTITIONED] hosts=1 instances=1
|  Per-Host Resources: mem-estimate=4.02MB mem-reservation=4.00MB thread-r
eservation=1
PLAN-ROOT SINK
|  mem-estimate=4.00MB mem-reservation=4.00MB spill-buffer=2.00MB thread-res
ervation=0
```

- The mem-estimate for the PLAN-ROOT SINK is an estimate of the amount of memory needed to spool all the rows returned by the query.
- The mem-reservation is the number and size of the buffers necessary to spool the query results. By default, the read and write buffers are 2 MB in size each, which is why the default is 4 MB.

PlanRootSink

In Impala, the PlanRootSink class controls the passing of batches of rows to the clients and acts as a queue of rows to be sent to clients.

- When result spooling is disabled, a single batch or rows is sent to the PlanRootSink, and then the client must consume that batch before another one can be sent.
- When result spooling is enabled, multiple batches of rows can be sent to the PlanRootSink, and multiple batches can be consumed by the client.

Related Information

Impala query options

Shut Down Impala

Explains how to gracefully shut down Impala Daemons by first allowing running queries a specified amount of time to complete the process.

The flow to gracefully shut down an Impala Daemon is as follows:

1. The shutdown is initiated.
2. The grace period starts. The Impala Daemon informs other coordinators not to schedule any new queries on it. This allows queries already scheduled to run on this daemon by other coordinators to start executing.
3. The grace period expires.
4. The Impala Daemon continuously checks if there are no queries or fragments running.
5. If there are no queries or fragments running, it shuts down.
6. Otherwise, when it reaches the IMPALA_GRACEFUL_SHUTDOWN_DEADLINE duration, Impala Daemon shuts down.

Once Cloudera Manager initiates the stop/shutdown command, the Impala Daemon starts up the graceful shutdown process, and the process cannot be reverted. However, if you need to change the hard deadline in Cloudera Manager, you can cancel the shutdown command, change the Impala Daemon Graceful Shutdown, and start the shutdown command again.

Procedure

1. Optionally, set the grace period.
   a) In Cloudera Manager, navigate to Impala Service Configuration Scope Impala Daemon .
   b) In the Impala Daemon Command Line Argument Advanced Configuration Snippet (Safety Valve) field, specify the grace period:

      --shutdown_grace_period_s=<new grace period in seconds>

      The default grace period is 2 minutes.

      It is strongly recommended that you use the default value and not change the setting.
2. Optionally, set the hard deadline after which Impala is shut down regardless of whether queries are still running on it.
   a) In Cloudera Manager, navigate to Impala Service Configuration Scope Impala Daemon .
   b) In the Impala Graceful Shutdown Deadline field, specify the time to wait for running queries.

      The default is 60 minutes.

      If you specify 0, Impala will shutdown immediately without waiting for running queries
3. In Cloudera Manager, navigate to Impala Service Instances .
4. Click an Impala Daemon role.
5. Click ActionsImpala Daemon Graceful Shutdown.

Related Information

SHUTDOWN statement

Setting Timeouts in Impala

Depending on how busy your cluster is, you might increase or decrease various timeout values. Increase timeouts if Impala is cancelling operations prematurely, when the system is responding slower than usual but the operations are still successful if given extra time. Decrease timeouts if operations are idle or hanging for long periods, and the idle or hung operations are consuming resources and reducing concurrency.

Setting Timeout and Retries for Thrift Connections to Backend Client

Impala connections to the backend client are subject to failure in cases when the network is momentarily overloaded.

About this task

To avoid failed queries due to transient network problems, you can configure the number of Thrift connection retries using the following option:

Procedure

1. In Cloudera Manager, navigate to Impala service Configuration .
2. In the Impala Daemon Command Line Argument Advanced Configuration Snippet (Safety Valve) field, specify the following.

   To avoid failed queries due to transient network problems, you can configure the number of Thrift connection retries using the following option:

   - The --backend_client_connection_num_retries option specifies the number of times Impala will try connecting to the backend client after the first connection attempt fails. By default, `impalad` will attempt three re-connections before it returns a failure.

   You can configure timeouts for sending and receiving data from the backend client. Therefore, if for some reason a query does not respond, instead of waiting indefinitely for a response, Impala will terminate the connection after a configurable timeout.

   - The --backend_client_rpc_timeout_ms option can be used to specify the number of milliseconds Impala should wait for a response from the backend client before it terminates the connection and signals a failure. The default value for this property is 300000 milliseconds, or 5 minutes.
3. Click Save Changes and restart Impala.

Increasing StateStore Timeout

If you have an extensive Impala schema, for example, with hundreds of databases, tens of thousands of tables, you might encounter timeout errors during startup as the Impala catalog service broadcasts metadata to all the Impala nodes using the StateStore service. To avoid such timeout errors on startup, increase the StateStore timeout value from its default of 10 seconds.

About this task

Increase the timeout value of the StateStore service if you see messages in the impalad log such as:

```
Connection with state-store lost
Trying to re-register with state-store
```

Procedure

1. In Cloudera Manager, navigate to Impala service Configuration .
2. In the search field, type -statestore_subscriber_timeout_seconds.
3. In the StateStoreSubscriber Timeout field, specify a new timeout value larger than the current value.
4. Click Save Changes and restart Impala.

Setting the Idle Query and Idle Session Timeouts

To keep long-running queries or idle sessions from tying up cluster resources, you can set timeout intervals for both individual queries, and entire sessions.

About this task

Procedure

1. In Cloudera Manager, navigate to Impala service Configuration .
2. In the search field, type idle.
3. In the Idle Query Timeout field, specify the time in seconds after which an idle query is cancelled.

   This could be a query whose results were all fetched but was never closed, or one whose results were partially fetched and then the client program stopped requesting further results. This condition is most likely to occur in a client program using the JDBC or ODBC interfaces, rather than in the interactive `impala-shell` interpreter. Once a query is cancelled, the client program cannot retrieve any further results from the query.

   You can reduce the idle query timeout by using the QUERY_TIMEOUT_S query option at the query level. Any non-zero value specified in this field serves as an upper limit for the QUERY_TIMEOUT_S query option.

   The value of 0 disables query timeouts.
4. In the Idle Session Timeout field, specify the time in seconds after which an idle session expires.

   A session is idle when no activity is occurring for any of the queries in that session, and the session has not started any new queries. Once a session is expired, you cannot issue any new query requests to it. The session remains open, but the only operation you can perform is to close it.

   The default value of 0 specifies sessions never expire.

   You can override this setting with the IDLE_SESSION_TIMEOUT query option at the session or query level.
5. Click Save Changes and restart Impala.

Results

Impala checks periodically for idle sessions and queries to cancel. The actual idle time before cancellation might be up to 50% greater than the specified configuration setting. For example, if the timeout setting was 60, the session or query might be cancelled after being idle between 60 and 90 seconds.

Adjusting Heartbeat TCP Timeout Interval

Using the TCP flag, you can prevent the Statestore from waiting indefinitely for a response from the subscribers that fail to respond to the heartbeat RPC within the set period.

About this task

This flag statestore_heartbeat_tcp_timeout_seconds defines the time that may elapse before a heartbeat RPC connection request from a Statestore server to an Impalad or a Catalog server (subscribers) should be considered dead.

You can increase the flag value if you see intermittent heartbeat RPC timeouts listed in the statestore's log. You may find the max value of "statestore.priority-topic-update-durations" metric on the statestore to get an idea of a reasonable value to be used in this flag.

Note: The priority topic updates are only small amounts of data that take little time to process, similar to the heartbeat complexity.

Procedure

1. In Cloudera Manager, navigate to Impala service Configuration .
2. In the Impala Daemon Command Line Argument Advanced Configuration Snippet (Safety Valve) field, add the flag statestore_heartbeat_tcp_timeout_seconds with an appropriate value.
3. You can also control the maximum number of consecutive heartbeat messages an impalad can miss before being declared failed by the statestore by adding this flag statestore_max_missed_heartbeats. Typically, you will not have to change this value.
4. Click Save Changes and restart Impala.

Cloudera Runtime 7.3.1

Troubleshooting Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Troubleshooting common issues in Impala.....4
Using Breakpad Minidumps for Crash Reporting.....4
Performance Issues Related to Data Encryption.....5
Troubleshooting Crashes Caused by Memory Resource Limit.....6

Troubleshooting common issues in Impala

This topic describes the general troubleshooting procedures to diagnose some of the commonly encountered issues in Impala.

Symptom	Explanation	Recommendation
Impala takes a long time to start.	Impala instances with large numbers of tables, partitions, or data files take longer to start because the metadata for these objects is broadcast to all `impalad` nodes and cached.	Adjust timeout and synchronicity settings.
Query rejected with the default pool-defined memory limit settings.	Some complex queries fail because the minimum memory reservation per host is greater than the memory available to the query for buffer reservations.	Increase VW t-shirt size so that there are more hosts in the executor group and less memory is needed per host.
Joins fail to complete.	There may be insufficient memory. During a join, data from the second, third, and so on sets to be joined is loaded into memory. If Impala chooses an inefficient join order or join mechanism, the query could exceed the total memory available.	Start by gathering statistics with the COMPUTE STATS statement for each table involved in the join. Consider specifying the [SHUFFLE] hint so that data from the joined tables is split up between nodes rather than broadcast to each node. If tuning at the SQL level is not sufficient, add more memory to your system or join smaller data sets.
Queries return incorrect results.	Impala metadata may be outdated after changes are performed in Hive.	After inserting data, adding a partition, or other operation in Hive, refresh the metadata for the table with the REFRESH statement.
Attempts to complete Impala tasks such as executing INSE RT SELECT statements fail. The Impala logs include notes that files could not be opened due to permission denied.	This can be the result of permissions issues. For example, you could use the Hive shell as the hive user to create a table. After creating this table, you could attempt to complete some action, such as an INSERT SELECT on the table. Because the table was created using one user and the INSERT SELECT is attempted by another, this action may fail due to permissions issues.	Ensure the Impala user has sufficient permissions to the table that the Hive user created.
Impala fails to start up, with the `impalad` logs referring to errors connecting to the statestored service and attempts to re-register.	A large number of databases, tables, partitions, and so on can require metadata synchronization, particularly on startup, that takes longer than the default timeout for the stat estore service.	Configure the statestore timeout value and possibly other settings related to the frequency of statestore updates and metadata loading.

Using Breakpad Minidumps for Crash Reporting

The breakpad project is an open-source framework for crash reporting. Impala can use breakpad to record stack information and register values when any of the Impala-related daemons crash due to an error such as SIGSEGV or unhandled exceptions. The dump files are much smaller than traditional core dump files. The dump mechanism itself uses very little memory, which improves reliability if the crash occurs while the system is low on memory.

Using the Minidump Files for Problem Resolution

You can see in the Impala log files or in the Cloudera Manager charts for Impala when crash events occur that generate minidump files. Because each restart begins a new log file, the “crashed” message is always at or near the bottom of the log file. (There might be another later message if core dumps are also enabled.)

Important: If an Impala-related daemon experiences a crash due to an out-of-memory condition, it does not generate a minidump for that error.

Typically, you provide minidump files to Cloudera Support as part of problem resolution, in the same way that you might provide a core dump. The Send Diagnostic Data under the Support menu in Cloudera Manager guides you through the process of selecting a time period and volume of diagnostic data, then collects the data from all hosts and transmits the relevant information for you.

Procedure

1. In Cloudera Manager, navigate to Impala service Configuration .
2. In the search field, type minidump.
3. Set the following fields to configure breakpad minidumps.
   - minidump_path: Turn on or off generation of the minidump files.

     By default, a minidump file is generated when an Impala-related daemon crashes.
   - minidump_path: Specify the Location for minidump files.

     By default, all minidump files are written to the following location on the host where a crash occurs: /var/log/impala-minidumps/*DAEMON_NAME*

     The minidump files for `impalad`, `catalogd`, and `statestored` are each written to a separate directory.

     If you specify a relative path for this setting, the value is interpreted relative to the default minidump_path directory.
   - max_minidumps: Specify the number of minidump files.

     Like any files used for logging or troubleshooting, consider limiting the number of minidump files, or removing unneeded ones, depending on the amount of free storage space on the hosts in the cluster.

     Because the minidump files are only used for problem resolution, you can remove any such files that are not needed to debug current issues.

     The default for this setting is 9. A zero or negative value is interpreted as “unlimited”.
4. Click Save Changes and restart Impala.
5. To provide minidump files to Cloudera Support as part of problem resolution, in Cloudera Manager, navigate to Support Send Diagnostic Data and follow the steps.

Performance Issues Related to Data Encryption

To troubleshoot any performance issues related to data encryption, run the PROFILE statement directly after running a query to view the encrypted bytes read.

Viewing the encrypted bytes read using query profile

When Impala executes any query on an Ozone encrypted file, the query PROFILE captures the runtime details of the execution including the total number of encrypted bytes read from Ozone by the query.

Note: The PROFILE statement should be run immediately after running the query. You cannot revisit later and view an earlier session.

You can view query profiles from the command line by logging in to the Impala shell and running the profile; command. You can also collect from Cloudera Manager Web UI, by navigating to CM Impala Queries , locate the query you just ran and click on Query Details . Look for the excerpts “BytesReadEncrypted” in the profile output.

Troubleshooting Crashes Caused by Memory Resource Limit

Default settings for max_map_count can be insufficient for Impala installations that run with many concurrent queries. We recommend increasing max_map_count to avoid potential failures due to exhausting memory mapping limits under heavy load.

Increasing the virtual memory parameter

max_map_count is the OS virtual memory parameter and defines the maximum number of memory map areas that a process can use.

To increase the max_map_countparameter in your OS and to make the above setting durable, refer to your OS documentation.

- For example, if you are using RHEL 9, add the following line to /etc/sysctl.conf:

```
/etc/sysctl.conf:vm.max_map_count=8000000
```

- Reload the config as root: sysctl -p.
- Check the new value: cat /proc/sys/vm/max_map_count.
- Restart.

Cloudera Runtime 7.3.1

Tuning Apache Impala

Date published: 2020-11-30
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Setting Up HDFS Caching **4**
Setting up Data Cache for Remote Reads **4**
Configuring Dedicated Coordinators and Executors **5**

Setting Up HDFS Caching

Set up HDFS caching with Impala for improved performance.

Before you begin

Decide how much memory to devote to the HDFS cache on each host. The total memory available for cached data is the sum of the cache sizes on all the hosts. By default, any data block is only cached on one host although you can cache a block across multiple hosts by increasing the replication factor.

Procedure

1. Enable or disable HDFS caching through Cloudera Manager using the configuration setting Maximum Memory Used for Caching for the HDFS service.

   This control sets the HDFS configuration parameter dfs_datanode_max_locked_memory, which specifies the upper limit of HDFS cache size on each node. Set up the HDFS caching for your Hadoop cluster.

   - All the other manipulation of the HDFS caching settings, such as what files are cached, is done through the command line, either Impala DDL statements or the Linux `hdfs cacheadmin` command.
2. Using the `hdfs cacheadmin` command, set up one or more pools owned by the same user as the `impalad` daemon (typically impala).

   For example:

   ```
   hdfs cacheadmin -addPool four_gig_pool -owner impala -limit 4000000000
   ```
3. Once HDFS caching is enabled and one or more pools are available, on the Impala side, you specify the cache pool name defined by the hdfs cacheadmin command in the Impala DDL statements that enable HDFS caching for a table or partition, such as CREATE TABLE ... CACHED IN *POOL* or ALTER TABLE ... SET CACHED IN *POOL*.
4. You can use `hdfs cacheadmin -listDirectives` to get a list of existing cache pools.
5. You can use `hdfs cacheadmin -listDirectives -stats` to get detailed information about the pools.

Setting up Data Cache for Remote Reads

When Impala compute nodes and its storage are not co-located, the network bandwidth requirement goes up as the network traffic includes the data fetch as well as the shuffling exchange traffic of intermediate results. To mitigate the pressure on the network, you can enable the compute nodes to cache the working set read from remote filesystems, such as, remote HDFS data node, S3, ABFS, ADLS.

About this task

To enable remote data cache as follows.

Procedure

1. In Cloudera Manager, navigate to Clusters Impala Service .
2. In the Configuration tab, select Enable Local Data Cache to enable the local Impala Daemon data cache for caching of remote reads.
3. In Impala Daemon Data Cache Directories, add the directories Impala Daemon will use for caching of remote read data.
4. In Impala Daemon Data Cache Per Directory Capacity, specify the maximum amount of local disk space Impala will use per daemon in each of the configured directrories for caching of remote read data.

5. Click Save Changes and restart the Impala service.

Configuring Dedicated Coordinators and Executors

Configure a dedicated coordinator and a dedicated executor roles to improve scalability of Impala.

Guidelines for Dedicated Coordinators and Executors

- Dedicated coordinator:
  - Should be on an edge node with no other services running on it.
  - Does not need large local disks but still needs some that can be used for Spilling.
  - Require at least the same or even larger memory allocation than executors.
- (Dedicated)Executors:
  - Should be co-located with DataNodes.
  - The number of hosts with dedicated executors typically increases as the cluster grows larger and handles more table partitions, data files, and concurrent queries.

Procedure

1. In Cloudera Manager, navigate to Clusters Impala Configuration Role Groups .
2. Click Create to create two role groups with the following values.
   a) Group for Coordinators
      1. Group Name: Coordinators
      2. Role Type: Impala Daemon
      3. Copy from:
         - Select Impala Daemon Default Group if you want the existing configuration gets carried over to the Coordinators.
         - Select None if you want to start with a blank configuration.
   b) Group for Executors
      1. Group Name: Executors
      2. Role Type: Impala Daemon
      3. Copy from:
         - Select Impala Daemon Default Group if you want the existing configuration gets carried over to the Executors.
         - Select None if you want to start with a blank configuration.
3. In the Role Groups page, click Impala Daemon Default Group.
   a) Select the set of nodes intended to be coordinators.
      1. Click Action for Selected and select Move To Different Role Group….
      2. Select the Coordinators.
   b) Select the set of nodes intended to be Executors.
      1. Click Action for Selected and select Move To Different Role Group….
      2. Select the Executors.
4. Click Configuration. In the search field, type Impala Daemon Specialization.
5. Click Edit Individual Values.
6. For Coordinators role group, select COORDINATOR_ONLY.
7. For Executors role group, select EXECUTOR_ONLY.
8. Click Save Changes and then restart the Impala service.

Related Information

Dedicated Coordinator

Cloudera Runtime 7.3.1

Integrating Apache Hive with BI tools

Date published: 2019-08-21
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Connecting Hive to BI tools using a JDBC/ODBC driver in Cloudera Data Hub 4
Getting the JDBC or ODBC driver in Cloudera Data Hub 4
Configuring the BI tool 4

Connecting Hive to BI tools using a JDBC/ODBC driver in Cloudera Data Hub

To query, analyze, and visualize data stored in Cloudera, you use drivers provided by Cloudera to connect Apache Hive to Business Intelligence (BI) tools.

About this task

How you connect to Hive depends on a number of factors: the location of Hive inside or outside the cluster, the HiveServer deployment, the type of transport, transport-layer security, and authentication. HiveServer is the server interface that enables remote clients to run queries against Hive and retrieve the results using a JDBC or ODBC connection.

Before you begin

- Choose a Hive authorization model.
- Configure authenticated users for querying Hive through JDBC or ODBC driver. For example, set up a Ranger policy.

Getting the JDBC or ODBC driver in Cloudera Data Hub

You need to download the Cloudera Hive JDBC or ODBC driver to your Cloudera Data Hub cluster and provide it to your client. In the BI tool on the client's cluster, you need to provide the HiveServer (HS2) host name portion of the JDBC URL. The BI tool requires this information to connect to Hive in Cloudera.

Procedure

1. Download the JDBC or ODBC driver from the Cloudera Downloads page.
2. Follow instructions in the documentation to install the driver on the same downloads page.

Configuring the BI tool

After you download an ODBC or JDBC driver and give it to your client, you need to provide the information that the BI tool requires to connect to Hive in Cloudera.

Before you begin

You have downloaded the JDBC or ODBC driver.

Procedure

1. Depending on the type of driver you obtain, proceed as follows:
   - ODBC driver: follow instructions on the ODBC driver download site, and skip the rest of the steps in this procedure.
   - JDBC driver: add the driver to the classpath of your JDBC client. For example, check the client documentation about where to put the driver.
2. In Cloudera Manager, click Clusters Hive on Tez to go to the Hive on Tez service page.

3. From the Hive on Tez service page, click Actions and select Download Client Configuration.

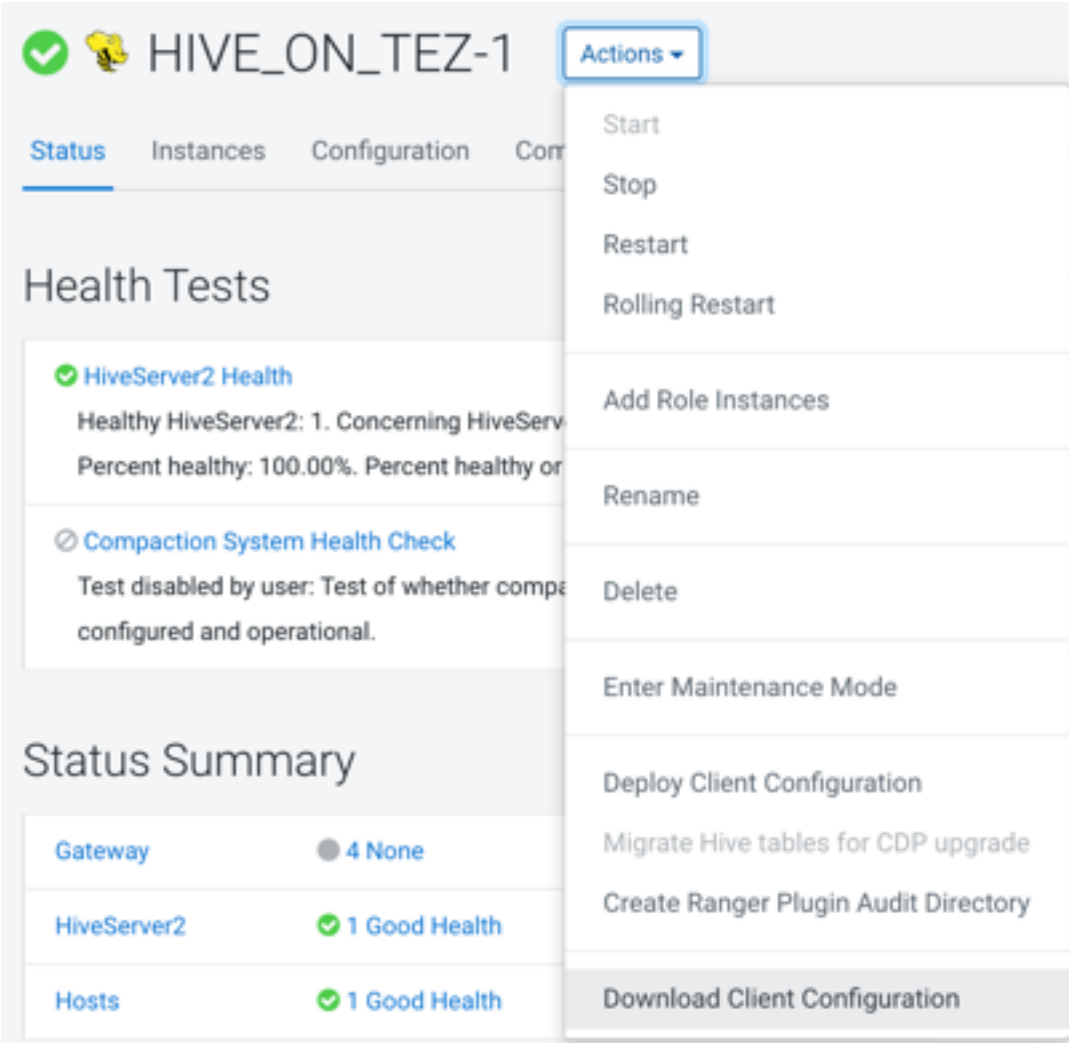

4. Unpack hive_on_tez-clientconfig.zip, open the beeline-site.xml file, and copy the value of beeline.hs2.jdbc.url.hive_on_tez. This value is the JDBC URL.
   For example

```
jdbc:hive2://my_hiveserver.com:2181/;serviceDiscoveryMode=zooKeeper; \
zooKeeperNamespace=hiveserver2
```

5. In the BI tool, configure the JDBC connection using the JDBC URL and driver class name, com.cloudera.hive.jdbc.HS2Driver.

Cloudera Runtime 7.3.1

Integrating Apache Hive with Spark and Kafka

Date published: 2019-08-21
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to HWC 5

Set up 5
HWC limitations 11
Reading data through HWC 12
Example of configuring and reading a Hive managed table 12
Direct Reader mode introduction 13
Using Direct Reader mode 15
Direct Reader configuration properties 16
Direct Reader limitations 17
Secure access mode introduction 17
Setting up secure access mode in Cloudera Data Hub 18
Using secure access mode 20
Configuring caching for secure access mode 22
JDBC read mode introduction 23
Using JDBC read mode 24
JDBC mode configuration properties 25
JDBC mode limitations 26
Kerberos configurations for HWC 26
Writing data through HWC 27
Apache Spark executor task statistics 28
Introduction to HWC and DataFrame APIs 30
HWC and DataFrame API limitations 32
HWC supported types mapping 33
Catalog operations 34
Read and write operations 34
Committing a transaction for Direct Reader 36
Closing HiveWarehouseSession operations 36
Using HWC for streaming 37
Hive Warehouse Connector streaming for transactional tables 37
Managing streaming with Hive Warehouse Connector 38
HWC API Examples 39
Hive Warehouse Connector Interfaces 40
Submitting a Scala or Java application 42
Examples of writing data in various file formats 42
HWC integration pyspark, sparklyr, and Zeppelin 44
Submitting a Python app 44
Reading and writing Hive tables in R 45
Livy interpreter configuration 48
Reading and writing Hive tables in Zeppelin 48

Apache Hive-Kafka integration 50

Creating a table for a Kafka stream 50
Querying Kafka data 51
Querying live data from Kafka 52
Perform ETL by ingesting data from Kafka into Hive 53
Writing data to Kafka 54
Writing transformed Hive data to Kafka 55
Setting consumer and producer table properties 55

Kafka storage handler and table properties.....55

Introduction to HWC

HWC securely accesses Hive managed tables from Spark. You need to use Hive Warehouse Connector (HWC) software to query Apache Hive managed tables from Apache Spark.

To read Hive external tables from Spark, you do not need HWC. Spark uses native Spark to read external tables. If you configure HWC to work with managed tables, you can use the same configuration to work with external tables. However, you must know that accessing external tables through HWC is slower as compared to accessing external tables through native Spark libraries.

Supported applications and operations

The Hive Warehouse Connector supports the following applications:

- Spark 3
- Spark shell
- PySpark
- The spark-submit script
- sparklyr
- Zeppelin with the Livy interpreter

The following list describes a few of the operations supported by the Hive Warehouse Connector:

- Describing a table
- Creating a table in ORC using .createTable() or in any format using .executeUpdate()
- Writing to a pre-existing or new table in Parquet, ORC, AVRO, or Textfile formats
- Selecting Hive data and retrieving a DataFrame
- Writing a DataFrame to a Hive-managed ORC table in batch
- Executing a Hive update statement
- Reading table data, transforming it in Spark, and writing it to a new Hive table
- Writing a DataFrame or Spark stream to Hive using HiveStreaming
- Partitioning data when writing a DataFrame

Related Information

HMS storage

Orc vs Parquet

Set up

You need to know how to use the Hive Warehouse Connector (HWC) with different programming languages and build systems. You find out where HWC binaries are located in Cloudera parcels and how a Spark application consumes the binaries.

Cloudera artifactory and HWC dependency

To pull the HWC dependency corresponding to a release, use the following artifactory:

```
https://repository.cloudera.com/artifactory/cloudera-repos
```

Use with Maven

To use HWC with maven, define the cloudera artifactory as a repository.

```
<repository>
  <id>cloudera</id>
  <name>cloudera</name>
  <url>https://repository.cloudera.com/artifactory/cloudera-repos</url>
 </repository>
```

In the pom.xml of the project, add the dependency as shown in the following example:

```
<dependency>
   <groupId>com.hortonworks.hive</groupId>
   <artifactId>hive-warehouse-connector_2.11</artifactId>
   <version>[***HWC VERSION***]</version>
   <scope>provided</scope>
</dependency>
```

Use with Sbt

Add the Cloudera and Shibboleth repositories, and the HWC dependency to the build sbt as follows:

```
resolvers += "Cloudera repo" at "https://repository.cloudera.com/artifactory
/cloudera-repos",

resolvers += "opensaml Repository" at "https://build.shibboleth.net/nexus/
content/repositories/releases",

libraryDependencies += "com.hortonworks.hive" % "hive-warehouse-connector_
2.11" % "[***HWC VERSION***]" % "provided",
```

Dependency scope

Generally, you add HWC dependencies in provided scope unless there is a specific requirement to do otherwise. While running spark application, you can specify the HWC jar present in your distribution using the --jars option to spark-submit or spark-shell.

HWC binaries in Spark 3 parcel

HWC supports Spark 3 through Cloudera Runtime.

The HWC binaries are located in /opt/cloudera/parcels/CDH/lib/hwc_for_spark3/. This directory contains HWC jar and a python zip. Use these binaries to launch Spark applications in Scala, Java, or Python.

The following files are in /opt/cloudera/parcels/CDH/lib/hwc_for_spark3/.

- hive-warehouse-connector-spark3-assembly-<version>.jar
- pyspark_hwc-spark3-<version>.zip

HWC configurations

Learn about the configurations that are required by Spark when using HWC. Setting HWC configurations has been simplified. As a cluster administrator, you can specify the required configurations in Cloudera Manager and then enable HWC by setting spark.cloudera.useHWC=true. You can either add this property to spark-defaults.conf to enable HWC for all Spark jobs or enable HWC for a specific Spark job by using the --conf option in spark-shell or spark-submit.

Add the following configurations in the spark-defaults.conf file by going to Clusters Spark 3 Configuration , search for spark-defaults.conf in Cloudera Manager. This is a one-time activity that is performed by a cluster administrator.

- spark.sql.hive.hiveserver2.jdbc.url="jdbc:hive2://<jdbc-url>"

- spark.datasource.hive.warehouse.read.mode=<mode>
- spark.security.credentials.hiveserver2.enabled=true
- spark.sql.hive.hiveserver2.jdbc.url.principal=<principal>/_HOST@ROOT.HWX.SITE
- spark.datasource.hive.warehouse.load.staging.dir=<hdfs://>/tmp/staging/hwc

Setting spark.cloudera.useHWC=true also adds all the jar files present in /opt/cloudera/parcels/CDH/lib/hwc_for_spark3/ to the Spark classpath along with adding the following configurations to Spark configuration:

- spark.kryo.registrator=com.qubole.spark.hiveacid.util.HiveAcidKyroRegistrator
- spark.sql.extensions=com.hortonworks.spark.sql.rule.Extensions

Note: You can choose to override the configuration values during runtime by using the --conf option while submitting a Spark job.

The following example shows how you can enable HWC when launching the Spark shell:

```
[root@sim-hwc-1 hive_warehouse_connector]# spark-shell  --conf spark.clouder
a.useHWC=true  --master yarn

Spark context available as 'sc' (master = yarn, app id = application_17325
48085468_0001).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.4.1.7.3.1.0-178
      /_/

Using Scala version 2.12.17 (OpenJDK 64-Bit Server VM, Java 1.8.0_232)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val hwc = com.hortonworks.hwc.HiveWarehouseSession.session(spark).bui
ld()
hive: com.hortonworks.spark.sql.hive.llap.HiveWarehouseSessionImpl = com.ho
rtonworks.spark.sql.hive.llap.HiveWarehouseSessionImpl@5ac9ae05
scala> hwc.sql("select * from acidtable").show
+---+-----+----+-----+----+-----+
| id| name|col3| col4|col5| col6|
+---+-----+----+-----+----+-----+
|  1|name1|   2|name2|   3|name3|
|  4|name4|   5|name5|   6|name6|
+---+-----+----+-----+----+-----+
```

Optional HWC configurations

Optionally, you can set the following properties:

- spark.datasource.hive.warehouse.write.path.strictColumnNamesMapping — Validates the mapping of columns against those in Hive to alert the user to input errors. Default = true.
- spark.sql.hive.conf.list — Propagates one or more configuration properties from the HWC to Hive. Set properties on the command line using the --conf option. For example:

  ```
  --conf spark.sql.hive.conf.list="hive.vectorized.execution.filesink.arro
  w.native.enabled=true;hive.vectorized.execution.enabled=true"
  ```

Do not attempt to set spark.sql.hive.conf.list programmatically.

Working with different languages

You use HWC APIs to perform basic read and write operations. You need to understand how to use HWC APIs with different languages. The following examples show basic capabilities that are covered in detail later in this documentation.

Use with Scala

```
import com.hortonworks.hwc.HiveWarehouseSession
import org.apache.spark.sql.{SaveMode, SparkSession}


object HWCApp {

  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder.appName("HWCApp").enableHiveSupport.g
etOrCreate
    val hwc = HiveWarehouseSession.session(spark).build
    // create sample data
    val tvSeries = createSampleDataDf(spark)
    val tableName = "tv_series"

    hwc.dropTable(tableName, true, true)
    println(s"=======Writing to hive table - $tableName via HWC=======")
    // write to hive table via HWC
    tvSeries.write.format(HiveWarehouseSession.HIVE_WAREHOUSE_CONNECTOR)
      .option("table", tableName)
      .mode(SaveMode.Append).save

    println(s"=======Reading hive table $tableName via HWC=======")
    // Read via HWC
    hwc.sql(s"select * from $tableName").show(truncate = false)

    hwc.close()
    spark.stop
  }

  private def createSampleDataDf(spark: SparkSession) = {
    spark.sql("drop table if exists tv_series_dataset")
    spark.sql("create table tv_series_dataset(id int, name string, genres
string, rating double) using orc")
    spark.sql("insert into tv_series_dataset values " +
      "(1, 'Chernobyl', 'Drama|History|Tragedy|Science', 9.4), " +
      "(2, 'Westworld', 'Sci-fi', 8.6), (3, 'Sense8', 'Sci-fi', 8.3), " +
      "(4, 'Person of Interest', 'Drama|Sci-fi', 8.4), " +
      "(5, 'Its okay to not be okay', 'Drama', 8.7), " +
      "(6, 'Daredevil', 'Action|Sci-fi', 8.6), " +
      "(7, 'Money Heist', 'Drama|Thriller', 8.3), " +
      "(8, 'Breaking Bad', 'Crime|Drama', 9.5)")
    spark.sql("select * from tv_series_dataset")
  }
```

Important: Spark 3 does not support automatic table creation using the SaveMode.Append and SaveMode .Overwrite modes. The table that you are writing to must exist in order to append or overwrite data in it. Instead, you can use the SaveMode.ErrorIfExists or SaveMode.Ignore modes.

Use with Java

The following Java code is equivalent to the scala code above.

```
import com.hortonworks.hwc.HiveWarehouseSession
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SaveMode;
```

```
import org.apache.spark.sql.SparkSession;

public class HWCApp {
  public static void main(String[] args) {
    SparkSession spark = SparkSession.builder().appName("HWCApp").enableHive
Support().getOrCreate();
    // HiveWarehouseSession creation
    HiveWarehouseSession hwc = HiveWarehouseBuilder.session(spark).build();
    // create sample data
    Dataset<Row> tvSeries = createSampleDataDf(spark);
    String tableName = "tv_series";
    hwc.dropTable(tableName, true, true);
    System.out.println("=======Writing to hive table - " + tableName + " via
 HWC=======");
    // write data to hive table via HWC
    tvSeries.write().format(HiveWarehouseSession.HIVE_WAREHOUSE_CONNECTOR)
        .option("table", tableName)
        .mode(SaveMode.Append).save();

    System.out.println("=======Reading hive table - " + tableName + " via
 HWC=======");
    // read hive table as dataframe using HWC
    hwc.sql("select * from " + tableName).show(false);
    hwc.close();
    spark.stop();
  }
  private static Dataset<Row> createSampleDataDf(SparkSession spark) {
    spark.sql("drop table if exists tv_series_dataset");
    spark.sql("create table tv_series_dataset(id int, name string, genres st
ring, rating double) using orc");
    spark.sql("insert into tv_series_dataset values " +
        "(1, 'Chernobyl', 'Drama|History|Tragedy|Science', 9.4), " +
        "(2, 'Westworld', 'Sci-fi', 8.6), (3, 'Sense8', 'Sci-fi', 8.3), " +
        "(4, 'Person of Interest', 'Drama|Sci-fi', 8.4), " +
        "(5, 'Its okay to not be okay', 'Drama', 8.7), " +
        "(6, 'Daredevil', 'Action|Sci-fi', 8.6), " +
        "(7, 'Money Heist', 'Drama|Thriller', 8.3), " +
        "(8, 'Breaking Bad', 'Crime|Drama', 9.5)");
    return spark.sql("select * from tv_series_dataset");
  }
}
```

Launching a Java or Scala app

After packaging the app in a jar, launch the app using standard Spark syntax for launching applications. Provide HWC jar from the distribution. The Spark application can be launched as follows:

```
spark-submit --jars /opt/cloudera/parcels/CDH/lib/hwc_for_spark3/hive-wareho
use-connector-spark3-assembly-<version>.jar \
--class com.cloudera.HWCApp \
...More spark/HWC confs...
...More spark/HWC confs...
/path-to-jar/hwc-app.jar
```

Use with Python

```
from pyspark.sql import SparkSession
from pyspark_llap import HiveWarehouseSession

spark = SparkSession.builder.enableHiveSupport().appName("hwc-app").getOrCre
ate()
hwc = HiveWarehouseSession.session(spark).build()
tableName = "tv_series"
```

```
hwc.dropTable(tableName, True, True)

tvSeries = spark.createDataFrame([
    (1, "Chernobyl", "Drama|History|Tragedy|Science", 9.4),
    (2, "Westworld", "Sci-fi", 8.6),
    (3, "Sense8", "Sci-fi", 8.3),
    (4, "Person of Interest", "Drama|Sci-fi", 8.4),
    (5, "It's okay to not be okay", "Drama", 8.7),
    (6, "Daredevil", "Action|Sci-fi", 8.6),
    (7, "Money Heist", "Drama|Thriller", 8.3),
    (8, "Breaking Bad", "Crime|Drama", 9.5)
], ["id", "name", "genres", "rating"])

print("=======Writing to hive table - " + tableName + " via HWC=======")
# write to hive table via HWC
tvSeries.write.format(HiveWarehouseSession.HIVE_WAREHOUSE_CONNECTOR).optio
n("table", tableName).mode("append").save()
print("=======Reading hive table - " + tableName + " via HWC=======")
# Read via HWC
hwc.sql("select * from " + tableName).show()

hwc.close()
spark.stop()
```

Launching a Python app

After getting the python code ready, launch it using spark-submit. Provide the HWC jar and HWC python zip as follows:

```
    spark-submit --jars /opt/cloudera/parcels/CDH/lib/hwc_for_spark3/hive-wa
rehouse-connector-spark3-assembly-<version>.jar \
    --py-files /opt/cloudera/parcels/CDH/lib/hwc_for_spark3/pyspark_hwc-<v
ersion>.zip \
    ...More spark/HWC confs...
    ...More spark/HWC confs...
    /path-to-python-app/hwc-app.py
```

Use with Sparklyr

You can access Hive tables through R by loading the sparklyr library along with the SparklyrHWC package available in /opt/cloudera/parcels/CDH/lib/hwc_for_spark3/, which can be used to trigger HWC APIs from R.

```
library(sparklyr)
library(SparklyrHWC, lib.loc = c(file.path("<path to SparklyrHWC>")))
#Set env variables
Sys.setenv(SPARK_HOME = "/opt/cloudera/parcels/CDH/lib/spark/")
Sys.setenv(HADOOP_HOME = "/opt/cloudera/parcels/CDH/lib/hadoop")

#Configurations needed to use spark-acid and related configurations.
config <- spark_config()
config$spark.sql.hive.hiveserver2.jdbc.url="jdbc:hive2://<url>:10000/defau
lt"
config$spark.datasource.hive.warehouse.user.name="hive"
config$spark.hadoop.hive.metastore.uris="thrift://<url>:9083"
config$spark.sql.extensions="com.hortonworks.spark.sql.rule.Extensions"
config$spark.kryo.registrator="com.qubole.spark.hiveacid.util.HiveAcidKyro
Registrator"
config$spark.datasource.hive.warehouse.read.mode="DIRECT_READER_V2"

#Build HWC session
hs <- build(HiveWarehouseBuilder.session(sc))

#Use database
```

```
sparklyr::sdf_sql(sc,"use test")
#Reading a managed table using spark acid direct-reader
intDf <- sparklyr::spark_read_table(sc, 'emp_hwc')
#Converts SparkDataframe to R dataframe
 sparklyr::sdf_collect(intDf1)
#Writing into a managed table
#Read first table
intDf <- sparklyr::spark_read_table(sc, 'emp_hwc')
#read second table
intDf1 <-  sparklyr::spark_read_table(sc, 'emp_overwrite')
#Commit transaction if read using spark-acid
commitTxn(hs)
#Append the second table, to the first.
SparklyrHWC::spark_write_table('emp_hwc',intDf1,'append')
#Overwrite the first table with the second table.
SparklyrHWC::spark_write_table('emp_hwc',intDf1,'overwrite')

#Using HWC Api’s
#create a table from existing table
SparklyrHWC::executeUpdate(hs,"create table hwc1 as select * from 'emp_hwc'
")
#Execute query
hwcDf <- SparklyrHWC::executeQuery(hs, "select * from hwc1")
#convert into R dataframe.
hwcSdf <- sparklyr::sdf_copy_to(sc, hwcDf)
```

HWC limitations

You need to be aware of HWC limitations, including Kerberos properties that are not allowed, and unsupported operations and connections. These limitations are in addition to Direct Reader mode, JDBC mode, Secure access mode, and HWC and DataFrames API limitations.

General HWC limitations are:

- The spark and livy thrift servers are not supported.
- Spark UDFs are not supported.
- The Hive Union types are not supported.

Workaround for using the Hive Warehouse Connector with Oozie Spark action

Hive and Spark use different Thrift versions and are incompatible with each other. Upgrading Thrift in Hive is complicated and may not be resolved in the near future. Therefore, Thrift packages are shaded inside the HWC JAR to make Hive Warehouse Connector work with Spark and Oozie Spark action. See the workaround in Cloudera Oozie documentation.

Secured cluster configurations

Set the following configurations in a secured cluster:

- ```
  --conf "spark.security.credentials.hiveserver2.enabled=true"
  ```
- ```
  --conf "spark.sql.hive.hiveserver2.jdbc.url.principal=hive/_HOST@ROOT.HW
  X.SITE"
  ```

  The jdbc url must not contain the jdbc url principal and must be passed as shown here.

Related Information

Using Hive Warehouse Connector with Oozie Spark Action

Union Types

Reading data through HWC

You can configure one of the several HWC modes to read Apache Hive managed tables from Apache Spark. You need to know about the modes you can configure for querying Hive from Spark. Examples of how to configure the modes are presented.

About this task

In this release, HWC configuration has been simplified.

You set the following configurations when starting the spark shell:

- spark.sql.extensions="com.hortonworks.spark.sql.rule.Extensions"
- spark.datasource.hive.warehouse.read.mode=<mode>

  where <mode> is one of the following:

- DIRECT_READER_V1 or DIRECT_READER_V2
- JDBC_CLUSTER
- SECURE_ACCESS

You can transparently read with HWC in different modes using just spark.sql("<query>"). You can specify the mode in the spark-shell when you run Spark SQL commands to query Apache Hive tables from Apache Spark. You can also specify the mode in configuration/spark-defaults.conf, or using the --conf option in spark-submit.

For backward compatibility, configuring spark.datasource.hive.warehouse.read.mode is the same as the following configurations.

- --conf spark.datasource.hive.warehouse.read.jdbc.mode //deprecated
- --conf spark.sql.hive.hwc.execution.mode //deprecated
- --conf spark.datasource.hive.warehouse.read.via.llap //deprecated

The old configurations are still supported for backward compatibility, but in a later release, support will end for these configurations and spark.datasource.hive.warehouse.read.mode will replace these configurations. HWC gives precedence to new configurations when old and new ones are encountered.

Related Information

Using Direct Reader mode

Using JDBC read mode

Apache Spark executor task statistics

Using secure access mode

Example of configuring and reading a Hive managed table

Before you begin

Set Kerberos for HWC.

Procedure

1. Choose a read mode.
2. Start the Spark session using the following configurations.
   For example, start the Spark session using Direct Reader and configure for kyro serialization:

```
spark-shell --jars ./hive-warehouse-connector-assembly-<version>.jar \
--master yarn \
--conf spark.sql.extensions="com.hortonworks.spark.sql.rule.Extensions" \
--conf spark.kryo.registrator=com.qubole.spark.hiveacid.util.HiveAcidKy
roRegistrator \
```

```
--conf spark.sql.hive.hiveserver2.jdbc.url="jdbc:hive2://hwc-2.hwc.root.
hwx.site:2181/default;retries=5;serviceDiscoveryMode=zooKeeper;zooKeeper
Namespace=hiveserver2" \
--conf spark.sql.hive.hiveserver2.jdbc.url.principal=hive/_HOST@ROOT.HW
X.SITE \
--conf spark.datasource.hive.warehouse.read.mode=DIRECT_READER_V2
```

For example, start the Spark session using the JDBC_CLUSTER option:

```
spark-shell --jars ./hive-warehouse-connector-assembly-<version>.jar
--master yarn
--conf spark.sql.extensions="com.hortonworks.spark.sql.rule.Extensions"
--conf spark.kryo.registrator=com.qubole.spark.hiveacid.util.HiveAcidKy
roRegistrator
--conf spark.sql.hive.hiveserver2.jdbc.url="jdbc:hive2://hwc-2.hwc.root.hw
x.site:2181/default;retries=5;serviceDiscoveryMode=zooKeeper;zooKeeperNa
mespace=hiveserver2"
--conf spark.sql.hive.hiveserver2.jdbc.url.prinicpal=hive/_HOST@ROOT.HW
X.SITE
--conf spark.datasource.hive.warehouse.read.mode=JDBC_CLUSTER
```

You must start the Spark session after setting the Direct Read option, so include the configurations in the launch string.

3. Read Apache Hive managed tables.
   For example:

```
scala> val hive = com.hortonworks.hwc.HiveWarehouseSession.session(spark
).build()

scala> hive.sql("select * from managedTable").show
```

Note: HWC supports the hive.sql() API for executing queries. The .execute() and .executeQuery() methods are deprecated and it is recommended that you use the hive.sql() method.

Direct Reader mode introduction

Direct Reader mode is a transparent connection that Hive Warehouse Connector (HWC) makes to Apache Hive metastore (HMS) to get transaction information. You use this mode if you do not need production-level Ranger authorization. Direct Reader mode does not support Ranger authorization. Direct Reader does support Spark-consistent timestamps in this release and later.

In Direct Reader mode, Spark reads the data directly from the managed table location using the transaction snapshot.

Requirements and recommendations

Spark Direct Reader mode requires a connection to Hive metastore. A HiveServer (HS2) connections is not needed.

The user running the Spark session must have access (read access on data and execute access on folder structure) to the underlying data of the table on the filesystem and to HMS.

Spark Direct Reader for reading Hive ACID, transactional tables from Spark is supported for production use. Use Spark Direct Reader mode if your ETL jobs do not require authorization and run as super user.

Component interaction

The following diagram shows component interaction for Direct Reader reads.

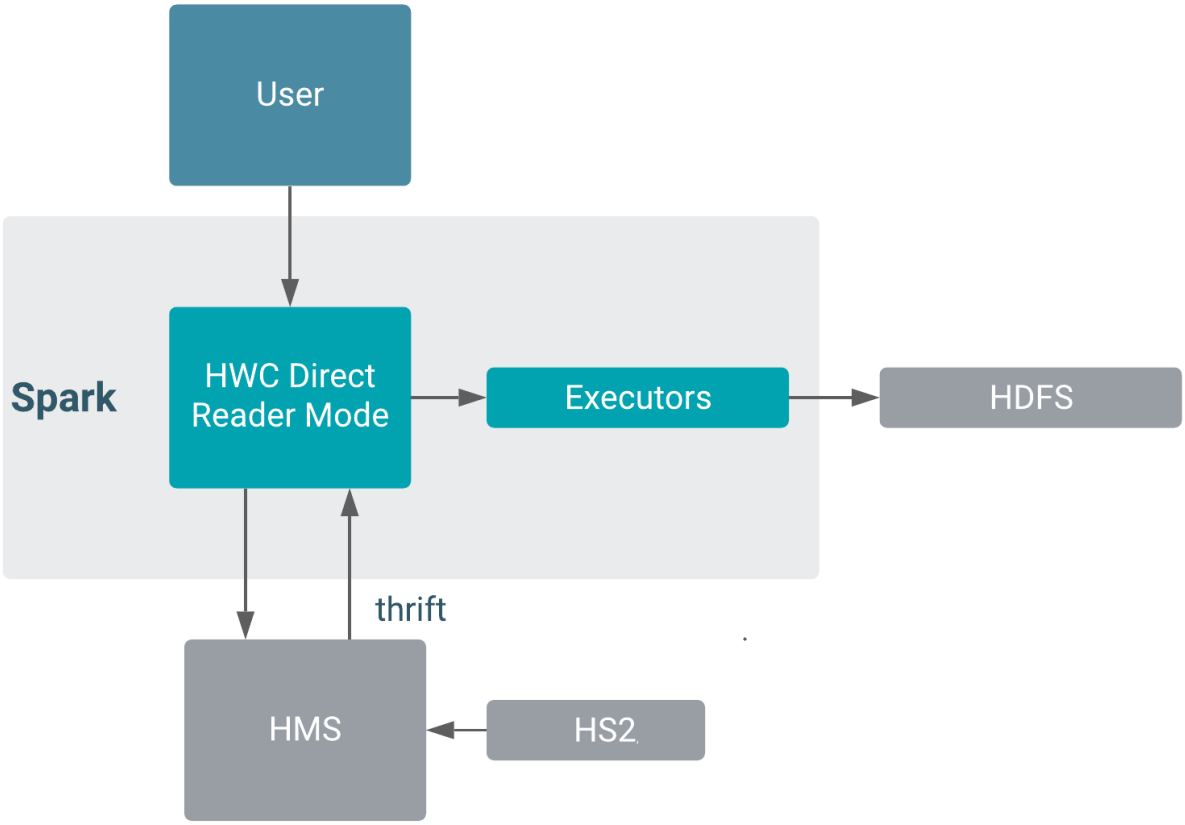

Using the timestamp type

When reading tables with Direct Reader, the behavior of the timestamp type is consistent with Spark. If you use the same storage format to store a timestamp value in an external table and a managed, transactional table, you see exactly the same value using the native Spark reader as you see using Direct Reader.

When querying a Hive table using Beeline, the behavior of the timestamp type is also consistent with Spark, assuming the Spark JVA and session time zones are consistent with the following exception: tables stored in Parquet format. Spark timestamp values differ depending on the value of spark.sql.hive.convertMetastoreParquet even for the external tables. Direct Reader reads of tables stored in Parquet format yields results as if spark.sql.hive.convertMetastoreParquet is false.

This design ensures that the timestamp values remain consistent across all the formats when read by Direct Reader. In the following section, examples show how Spark and Direct Reader behave with timestamps. The examples query external and transactional tables in different formats. All the tables were created using Hive beeline. A timestamp value “1989-01-05 00:00:00” was inserted. The Spark native reader reads external tables. Direct Reader reads transactional tables. A union of all the dataframes make the result more presentable and comparable.

With spark.sql.hive.convertMetastoreParquet = true (default in spark)

```
+-------------------------+-------------------+
|type                     |timestamp          |
+-------------------------+-------------------+
|External parquet         |1989-01-04 16:00:00|
|Transactional parquet    |1989-01-05 00:00:00|
|External orc             |1989-01-05 00:00:00|
|Transactional orc        |1989-01-05 00:00:00|
|External avro            |1989-01-05 00:00:00|
|Transactional avro       |1989-01-05 00:00:00|
```

```
|External text            |1989-01-05 00:00:00|
|Transactional text       |1989-01-05 00:00:00|
+-------------------------+-------------------+
```

With spark.sql.hive.convertMetastoreParquet = false

```
+-------------------------+-------------------+
|type                     |timestamp          |
+-------------------------+-------------------+
|External parquet         |1989-01-05 00:00:00|
|Transactional parquet    |1989-01-05 00:00:00|
|External orc             |1989-01-05 00:00:00|
|Transactional orc        |1989-01-05 00:00:00|
|External avro            |1989-01-05 00:00:00|
|Transactional avro       |1989-01-05 00:00:00|
|External text            |1989-01-05 00:00:00|
|Transactional text       |1989-01-05 00:00:00|
+-------------------------+-------------------+
```

Direct Reader is aligned to Spark for all the storage formats, including convertMetastoreParquet.

Related Information

Using Direct Reader mode

Using JDBC read mode

Using Direct Reader mode

In a few steps, you configure Apache Spark to connect to the Apache Hive metastore. An example shows how to configure Direct Reader reads while launching the Spark shell.

About this task

This procedure assumes you require serialization and sets spark.kryo.registrator=com.qubole.spark.hiveacid.util.HiveAcidKyroRegistrator.

Before you begin

For secure clusters, additional configurations will be needed by spark.

.

Procedure

1. In Cloudera Manager, in Hosts Roles , if Hive Metastore appears in the list of roles, copy the host name or IP address.

   You use the host name or IP address in the next step to set the host value.

2. Launch the Spark shell and include the Direct Reader configurations.
   For example:

```
spark-shell --jars ./hive-warehouse-connector-assembly-<version>.jar \
--master yarn \
--conf spark.sql.extensions="com.hortonworks.spark.sql.rule.Extensions" \
--conf spark.kryo.registrator=com.qubole.spark.hiveacid.util.HiveAcidKy
roRegistrator \
--conf spark.sql.hive.hiveserver2.jdbc.url="jdbc:hive2://<domain name>:<
port>/default;principal=hive/_HOST@ROOT.HWX.SITE;retries=5;serviceDiscov
eryMode=zooKeeper;zooKeeperNamespace=hiveserver2"

--conf spark.datasource.hive.warehouse.read.mode=DIRECT_READER_V2
```

3. Read data in table customer.
   View data in table customer.

```
scala>
hive.sql("select c_customer_sk, c_customer_id, c_last_name, c_birth_count
ry from customer where c_birth_year=1983 limit 2 ").show()

21/02/08 11:03:31 INFO rule.HWCSwitchRule: using DIRECT_READER_V2 extens
ion for reading
+-------------+----------------+-----------+---------------+

|c_customer_sk|   c_customer_id|c_last_name|c_birth_country|
+-------------+----------------+-----------+---------------+
|        55634|AAAAAAAACFJNAAAA|   Campbell|       THAILAND|
|        74213|AAAAAAAAFOBCBAAA|    Hudgins|     KYRGYZSTAN|
+-------------+----------------+-----------+---------------+
```

Direct Reader configuration properties

You need to know the property names and valid values for configuring Direct Reader mode. The advantage of using Direct Reader V2 over Direct Reader V1 is its ability to process ORC data using vectorization, which improves performance.

Options

In configuration/spark-defaults.conf, or using the --conf option in spark-submit/spark-shell set the following properties:

Name: spark.sql.extensions

Value: com.hortonworks.spark.sql.rule.Extensions

Required for using Spark SQL in auto-translate direct reader mode. Set before creating the spark session.

spark.datasource.hive.warehouse.read.mode

DIRECT_READER_V1 or DIRECT_READER_V2

Required for using Direct Reader with the Spark Data Source API V1 or V2 (recommended), respectively.

Name: spark.kryo.registrator

Value: com.qubole.spark.hiveacid.util.HiveAcidKyroRegistrator

Set before the spark session. Required if serialization = kryo.

Name: spark.hadoop.hive.metastore.uris

Value: thrift://<host>:<port>

Hive metastore URI.

Name: --jars

Value: HWC jar

Pass the HWC jar to spark-shell or spark-submit using the --jars option while launching the application. For example, launch spark-shell as follows.

Example: Launch a spark-shell

```
spark-shell --jars \
    /opt/cloudera/parcels/CDH/lib/hive_warehouse_connector/hive-warehouse-c
onnector-assembly-<version>.jar \
    --conf "spark.sql.extensions=com.hortonworks.spark.sql.rule.Extensions"
 \
```

```
    --conf "spark.datasource.hive.warehouse.read.mode=DIRECT_READER_V2
    --conf "spark.kryo.registrator=com.qubole.spark.hiveacid.util.HiveAci
dKyroRegistrator" \
    --conf "spark.hadoop.hive.metastore.uris=<metastore_uri>"
```

Direct Reader limitations

You must understand the limitations of Direct Reader mode and what functionality is not supported.

Limitations

- You cannot write data using HWC Direct Reader.
- Transaction semantics of Spark RDDs are not ensured when using Spark Direct Reader to read ACID tables.
- Supports only single-table transaction consistency. The direct reader does not guarantee that multiple tables referenced in a query read the same snapshot of data.
- This mode does not require a Hive Server (HS2) connection, therefore, the audit event is generated by HMS, which captures just the type of access (for example, SELECT) and does not capture all the details (about columns). Also, the audit event does not log the actual query (SQL).
- Does not auto-commit transactions submitted by dataframe or rdd APIs. Explicitly close transactions to release locks.

  Some of the operations, such as df.cache, df.persist, and df.rdd open transactions but do not close them. This is expected and the transactions are closed automatically after the end of the spark application. If such transactions have to be closed immediately, then it is recommended to execute the following in Spark:

  ```
  com.qubole.spark.hiveacid.transaction.HiveAcidTxnManagerObject.commitTxn
  (spark)
  ```

- Does not support Ranger authorization.

  You must configure read access to the HDFS, or other, location for managed tables. You must have Read and Execute permissions on hive warehouse location (hive.metastore.warehouse.dir).
- Blocks compaction on open read transactions.
- Does not support stoarge-handlers.

The way Spark handles null and empty strings can cause a discrepancy between metadata and actual data when writing the data read by Spark Direct Reader to a CSV file.

Unsupported functionality

Spark Direct Reader does not support the following functionality:

- Writes
- Streaming inserts
- CTAS statements

Secure access mode introduction

Use Hive Warehouse Connector (HWC) secure access mode if you want fine-grained access control (FGAC) column masking and row filtering to secure managed (ACID); or external, Hive table data that you read from Spark. If you have large workloads, low-latency requirements, and require fine-grained access control, secure access mode is recommended over the Direct Reader mode.

As an administrator, you set up Ranger FGAC, consisting of column masking and row filtering, to secure the data. You select a staging location in your cloud storage service, such as S3 or ADLS. After you configure secure access mode, HWC creates external tables on your designated staging location when users launch Spark. HWC uses CREATE TABLE AS SELECT (CTAS) to create the tables. No code refactoring is necessary. Ranger policies are applied and FGAC is enforced during the CTAS operation. Users read external tables in the secure staging location.

Requirements

- As an administrator, you set up Ranger policies.
- As a user, you launch Spark and query Hive tables in secure access mode.

Considerations

- Intermediate data is generated every time the Spark job runs even if it is running on the same underlying data.
- Intermediate data is automatically cleared when the Spark session ends.
- Additional storage requirements depend on concurrent Spark jobs running.

Related Information

Setting up secure access mode

Using secure access mode

Setting up secure access mode in Cloudera Data Hub

Learn how to set up Ranger policies on a staging location. This location is used to temporarily store Hive files that users need to read from Spark using the HWC secure access mode.

About this task

Before using secure access mode to read Hive data, you must set up two Ranger S3 access policies on the staging location for the Hive and Spark session users, and then set up a Hive URL authorization policy on the staging location.

As an example, let us consider S3 as the cloud storage service and assume the S3 staging location to be s3a://s3-hwc/stagingHWC/. In the following procedure, as an administrator, you set up Ranger policies on files and directories in the staging location. These policies secure managed ACID as well as external tables.

Procedure

1. In Cloudera Manager, click Hive on Tez Configuration , search for hive.server2.enable.doAs, and, if necessary, clear the selection to disable.

   This step is required by Hive for managing ACID tables.

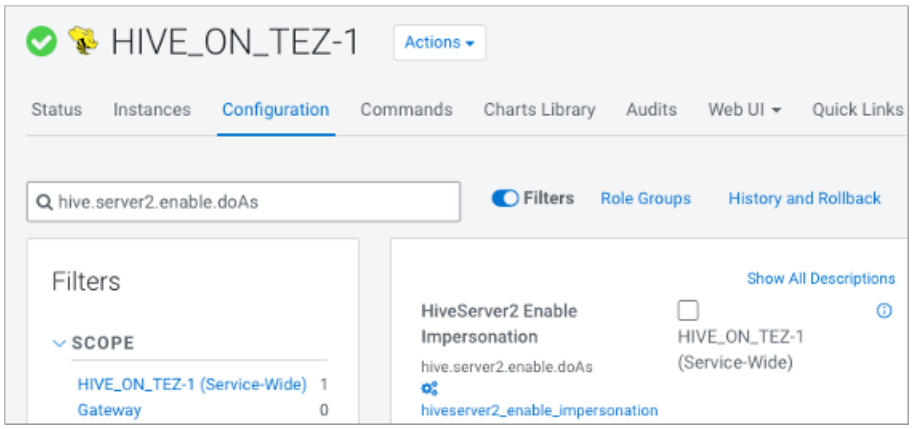

2. Log into the Ranger Admin UI and in the **Service Manager** page, click the S3 preloaded resource-based service, for example, cm_s3.
3. Click Add New Policy to create an access policy for the end user who launches the spark-shell and initiates a Spark session.

4. In the **Create Policy** page, specify a policy name, S3 bucket name, and path of the staging location.

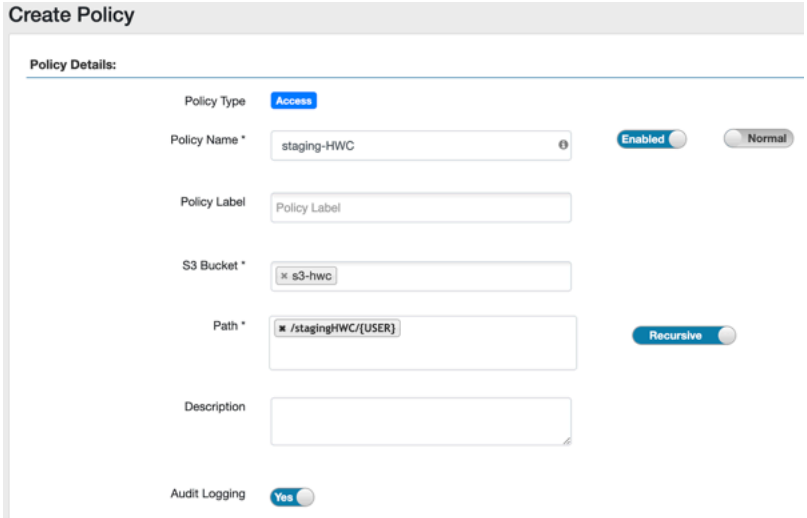

5. In the Allow Conditions section, select {USER} and add the read and write permissions, and then click Add to create the policy.

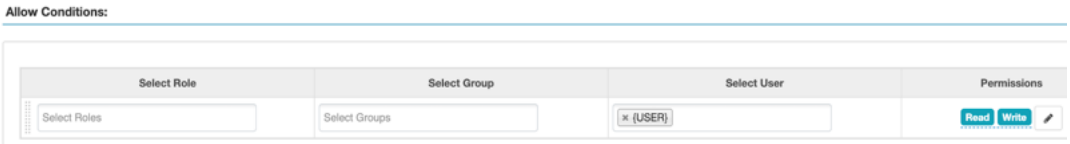

The Spark session users must have access to the staging location. This action sets a single Ranger policy for all users.

The policy is created and displayed in the list of available S3 policies.

6. Click Add New Policy again to add another S3 policy that grants the Hive user additional privileges on the staging path.
7. In the **Create Policy** page, specify the policy name, S3 bucket name, and path of the staging location.

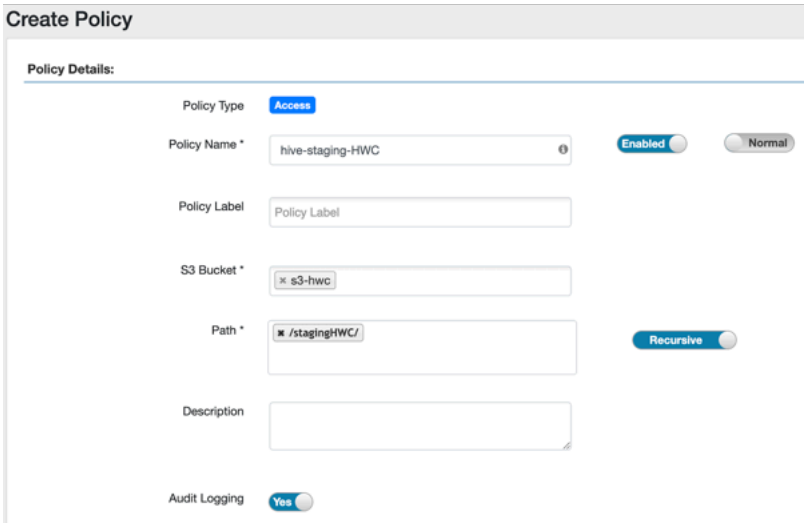

8. In the Allow Conditions section, select hive as the user and add the read and write permissions, and then click Add to create the policy.

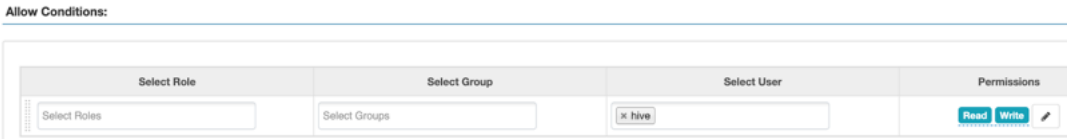

The policy is created and displayed in the list of available S3 policies.

9. Click the Service Manager link in the breadcrumb trail and then click the Hadoop SQL preloaded resource-based service.
10. Click Add New Policy to create the Hive URL authorization policy.
11. In the **Create Policy** page, select url from the drop-down list, and specify a policy name and the S3 staging location.

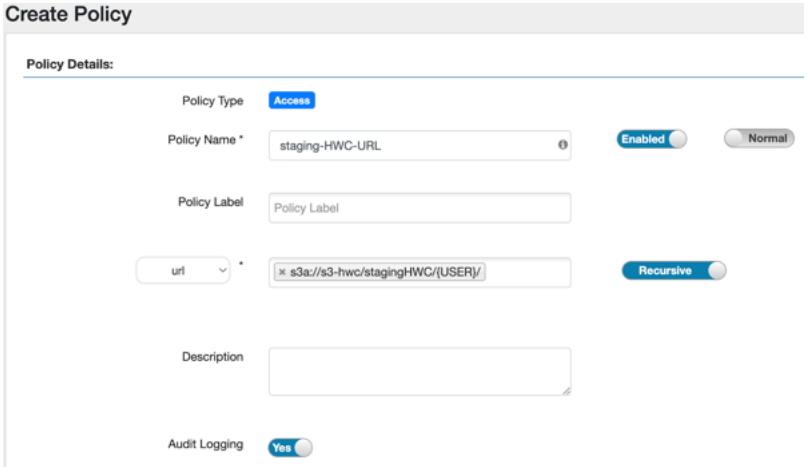

12. In the Allow Conditions section, select {USER} and add all permissions, and then click Add to create the policy.

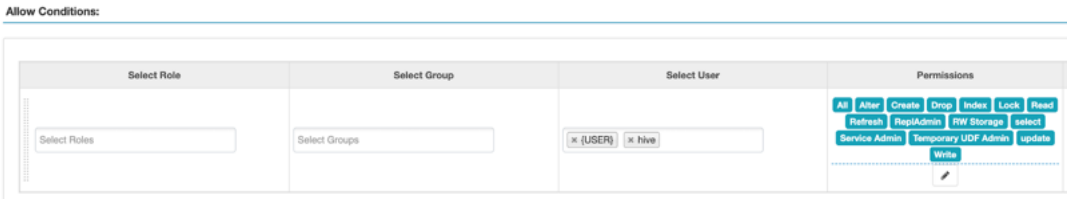

The policy is created and displayed in the list of available Hadoop SQL policies.

What to do next

You can use the HWC secure access mode to securely read Hive data from Spark.

Related Information

Configure a resource-based policy: S3

Configure a resource-based policy: HadoopSQL

Using secure access mode

Learn how to use HWC secure access mode that offers fine-grained access control (FGAC) column masking and row filtering to secure managed (ACID), or external Hive table data that you read from Spark.

About this task

In this task, you enter the following configuration options when you launch the Spark shell.

- spark.datasource.hive.warehouse.load.staging.dir=<path of the staging location in your cloud storage service>, for example, s3a://s3-hwc/stagingHWC/. The path represents the temporary subdirectories per user, per session that are created under this directory. This is the temporary location where the output is staged for the duration of

the session. Make sure to provide the absolute path, or fully qualified name (FQDN), not the relative path of the staging directory.

- spark.datasource.hive.warehouse.read.mode=secure_access starts using the staging output mode with fine-grained access control (FGAC). No code refactoring is required. You can use hive.sql() in your queries.

Note: Using Spark UDFs on HWC is a limitation. The workaround is to use Hive UDFs instead, as they are compatible and provide similar functionality within the Hive Warehouse Connector framework.

Before you begin

- Your administrator has set up ranger permissions for using secure access mode on Hive tables.
- Your administrator has granted you permission to the staging location in your cloud storage service, such as S3 or ADLS.

Procedure

1. Launch the Spark shell, configuring secure access mode.
   For example, use the Spark session user name livy.

```
[cloudbreak@hwc-dh-master0 hwc]$ sudo -u livy spark-shell
--jars  /opt/cloudera/parcels/CDH/jars/hive-warehouse-connector-assembly-
<version>-SNAPSHOT.jar
--master yarn spark.sql.extensions="com.hortonworks.spark.sql.rule.Extens
ions"
--conf spark.kryo.registrator=com.qubole.spark.hiveacid.util.HiveAcidKyro
Registrator
--conf spark.sql.hive.hiveserver2.jdbc.url="jdbc:hive2://<domain name>:<
port>
/default;httpPath=cliservice;retries=5;serviceDiscoveryMode=zooKeeper;ssl=
true;transportMode=http;zooKeeperNamespace=hiveserver2"
--conf spark.datasource.hive.warehouse.read.mode=secure_access
--conf spark.datasource.hive.warehouse.load.staging.dir="s3a://s3-hwc/sta
gingHWC/"
--conf spark.security.credentials.hiveserver2.enabled=true
--conf spark.sql.hive.hiveserver2.jdbc.url.principal=hive/_HOST@ROOT.HW
X.SITE
```

   Output is:

```
Setting default log level to "WARN".

Spark context Web UI available at http://<domain name>:<port>
Spark context available as 'sc' (master = yarn, app id = application_163
7657444149_0024).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\
      /_/

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_292)
Type in expressions to have them evaluated.
Type :help for more information.
```

2. Read a Hive table.

```
scala> val hive = com.hortonworks.hwc.HiveWarehouseSession.session(spark
).build()
```

```
hive: com.hortonworks.spark.sql.hive.llap.HiveWarehouseSessionImpl = com.h
ortonworks.spark.sql.hive.llap.HiveWarehouseSessionImpl@159b4611

scala> hive.sql("select id, name from test.acidtable limit 3").show

+---+--------------------+
| id|                name|
+---+--------------------+
|  1|   namex x xxxx xxxx|
|  4|namex xxxx xxxxxx...|
|  1|   namex x xxxx xxxx|
+---+--------------------+
```

Example

To illustrate how fine-grained access control column masking and row filtering works, consider the following schema of a managed Hive table, test.acidtable:

```
+--------+---------+-------+
|col_name|data_type|comment|
+--------+---------+-------+
|      id|      int|       |
|    name|   string|       |
|    col3|      int|       |
|    col4|   string|       |
|    col5|      int|       |
|    col6|   string|       |
+--------+---------+-------+
```

The Spark session user, 'livy' has access to only the id and name columns and there is a masking policy on the name column.

```
scala> val hive = com.hortonworks.hwc.HiveWarehouseSession.session(spark).bu
ild()
hive: com.hortonworks.spark.sql.hive.llap.HiveWarehouseSessionImpl = com.h
ortonworks.spark.sql.hive.llap.HiveWarehouseSessionImpl@6e12d8be

scala> hive.sql("select col4, col5 from test.acidtable").show
```

Output is:

```
22/01/01 08:31:48 WARN conf.HiveConf: HiveConf of name hive.masking.algo doe
s not exist
java.lang.RuntimeException: Error while compiling statement: FAILED: HiveAcc
essControlException Permission denied: user [livy] does not have [SELECT] pr
ivilege on [test/acidtable/col4,col5]
```

The exception indicates that the user, 'livy' does not have access to columns — col4 and col5.

Related Information

Introduction to secure access mode

Setting up secure access mode

Configuring caching for secure access mode

You can enable or disable caching for the Hive Warehouse Connector (HWC) secure access mode to have finer control over read queries and ensure that the content updated outside of a Spark session is considered during reads. Caching is enabled by default because queries that run with caching enabled tend to run faster.

Before you begin

You must ensure that you are using the HWC secure access mode for reads: spark.datasource.hive.warehouse.read .mode=secure_access

Procedure

Caching is enabled by default, however, you can choose to disable caching by setting the spark.hadoop.secure.acce ss.cache.disable property to true either at a global-level, session-level, or at a runtime-level when running a query. Queries that run with caching disabled tend to run slower.

- Global-level — Specify the property in the spark-defaults.conf file.
- Session-level — Specify the property using the --conf option in spark-submit:

```
.bin/spark-submit \
--conf "spark.hadoop.secure.access.cache.disable=true"
```

- Runtime-level — Specify the property just before running your queries:

```
scala> spark.conf.set("spark.hadoop.secure.access.cache.disable","true")
scala> hive.sql("select * from anytable").show
```

Note: The value specified for the configuration at a runtime-level is valid for all subsequent queries until it is modified.

The order of preference for configuration is as follows:

a. Property passed at a run-time level
b. Property passed at a Spark session-level
c. Property set in the spark-defaults.conf file

JDBC read mode introduction

JDBC read mode is a connection that Hive Warehouse Connector (HWC) makes to HiveServer (HS2) to get transaction information. JDBC read mode is secured through Ranger authorization and supports fine-grained access control, such as column masking. You need to understand how you read Apache Hive tables from Apache Spark through HWC using the JDBC mode. The location where your queries are executed affects configuration. Understanding execution locations and recommendations help you configure JDBC reads for your use case.

Component Interaction

Only one JDBC connection to HiveServer (HS2) is a potential bottleneck in data transfer to Spark. The following diagram shows interaction when you configure HWC in JDBC mode with Hive metastore (HMS), TEZ, and HDFS.

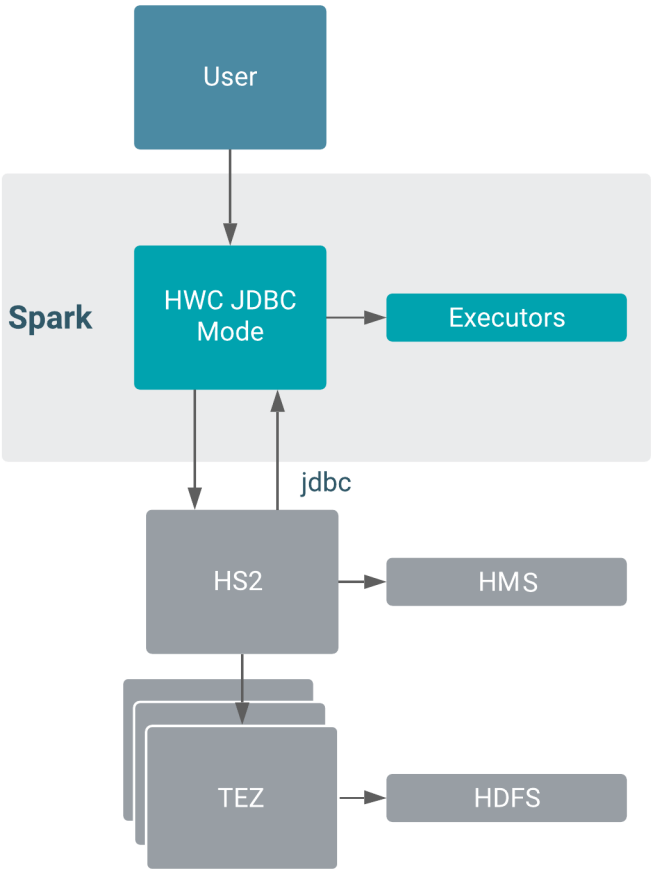

HWC does not use JDBC to write. HWC writes to an intermediate location from Spark, and then executes a LOAD DATA query to write the data. Using HWC to write data is recommended for production.

Configuration

A JDBC read takes place in the cluster — From Spark executors, connects to Hive through JDBC and executes the query. Authorization occurs on the server and any failures to connect to HS2 will be retried automatically.

JDBC reads are recommended for production for workloads having a data size of 1GB or less. Larger workloads are not recommended for JDBC reads in production due to slow performance.

Where your queries are executed affects the Kerberos configurations for HWC.

Optimize reads using HWC session APIs

Using the HWC session API, you can use hive.sql to execute a fast read. This command processes queries through HWC to perform JDBC or Direct Reader reads.

Using JDBC read mode

In a few steps, you configure Apache Spark to connect to HiveServer (HS2). Examples show how to configure JDBC Cluster mode while launching the Spark shell.

Before you begin

- Accept the default spark.datasource.hive.warehouse.load.staging.dir for the temporary staging location required by HWC.
- In spark-defaults.conf, check that spark.hadoop.hive.zookeeper.quorum is configured.

- In spark-defaults.conf, set Kerberos configurations for HWC, or for an unsecured cluster, set spark.security.credentials.hiveserver2.enabled=false.

Procedure

1. Find the HiveServer (HS2) JDBC URL in /etc/hive/conf.cloudera.HIVE_ON_TEZ-1/beeline-site.xml
   The value of beeline.hs2.jdbc.url.HIVE_ON_TEZ-1 is the HS2 JDBC URL in this sample file.

```
...
<configuration>
 <property>
 <name>beeline.hs2.jdbc.url.default</name>
 <value>HIVE_ON_TEZ-1</value>
 </property>
 <property>
 <name>beeline.hs2.jdbc.url.HIVE_ON_TEZ-1</name>
 <value>jdbc:hive2://<domain name>:2181/;serviceDiscoveryMode=zooKeeper; \
    zooKeeperNamespace=hiveserver2;retries=5</value>
 </property>
</configuration>
```

2. Launch the Spark shell, including the configuration of the JDBC cluster option, and setting the Spark property to the value of the HS2 JDBC URL.
   For example:

```
spark-shell --jars ./hive-warehouse-connector-assembly-<version>.jar \
--master yarn \
--conf spark.sql.extensions="com.hortonworks.spark.sql.rule.Extensions" \
--conf spark.kryo.registrator=com.qubole.spark.hiveacid.util.HiveAcidKy
roRegistrator \
--conf spark.sql.hive.hiveserver2.jdbc.url="jdbc:hive2://hwc-2.hwc.root.
hwx.site:2181/default;principal=hive/_HOST@ROOT.HWX.SITE;retries=5;servi
ceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2"

--conf spark.datasource.hive.warehouse.read.mode=JDBC_CLUSTER
```

3. Read a hive table.

```
scala> hive.sql("select * from managedTable").show(1, false)
OR
scala> spark.sql("select * from managedTable").show(1, false)
```

JDBC mode configuration properties

You need to know the property names and valid values for configuring JDBC mode.

In configuration/spark-defaults.conf, or using the --conf option in spark-submit/spark-shell set the following properties:

Name: spark.sql.extensions

Value: com.hortonworks.spark.sql.rule.Extensions

Name: spark.datasource.hive.warehouse.read.mode

Value: JDBC_CLUSTER

Configures the driver location.

Name: spark.sql.hive.hiveserver2.jdbc.url

Value: The JDBC endpoint for HiveServer. For more information, see the Apache Hive Wiki (link below). For Knox, provide the HiveServer, not Knox, endpoint.

Name: spark.datasource.hive.warehouse.load.staging.dir

Value: Temporary staging location required by HWC. Set the value to a file system location where the HWC user has write permission.

Name: spark.hadoop.hive.zookeeper.quorum

JDBC mode limitations

You must understand the limitations of JDBC mode and what functionality is not supported.

Keep the following limitations of JDBC mode in mind:

- JDBC_CLUSTER is used for reads only, and is recommended for production workloads of 1 GB or less. With larger workload, bottlenecks develop in data transfer to Spark. Use only for running DML to extract data into an external table for direct spark.sql access.

  Writes through HWC of any size are recommended for production. Writes do not use JDBC mode.
- Data transfer is through a single JDBC connection. Does not scale and is slow for large volumes of data.
- Executors require access to HiveServer (HS2).
- In JDBC_CLUSTER mode, HWC fails to correctly resolve queries that use the ORDER BY clause when run as hive.sql(" <query> "). The query returns unordered rows of data even though the query contains an ORDER BY clause.
- In JDBC read mode, a query of a table having a column of a complex type, such as ARRAY, STRUCT and MAP, incorrectly represents the type as String in the returned DataFrame.
- Spark fetches only the first x rows from the connection. This is to prevent HS2 timeouts which might otherwise cause Spark to hit Session/Operation Handle exceptions.

Kerberos configurations for HWC

You learn how to set up HWC for Kerberos, or not. You set properties and values depending on the JDBC_CLUSTER or JDBC_CLIENT option you configure.

In Cloudera Private Cloud Base, you need to configure HWC options in configuration/spark-defaults.conf, depending on the read option you select. Alternatively, you can set the properties using the spark-submit/spark-shell --conf option.

Secured cluster configuration

For Spark applications on a kerberized Yarn cluster, set the following property: spark.sql.hive.hiveserver2.jdbc.url.principal. This property must be equal to hive.server2.authentication.kerberos.principal.

On a kerberized YARN cluster, set the following property:

- JDBC_CLUSTER option in a secured cluster
  - Property: spark.security.credentials.hiveserver2.enabled
  - Value: true
  - Comment: true by default
- JDBC_CLIENT option in a secured cluster
  - Property: spark.security.credentials.hiveserver2.enabled
  - Value: false

Unsecured cluster configuration

In an unsecured cluster, set the following property:

- Property: spark.security.credentials.hiveserver2.enabled
- Value: false

Writing data through HWC

A step-by-step procedure walks you through connecting to HiveServer (HS2) to perform batch writes from Spark, which is recommended for production. You configure HWC for the managed table write, launch the Spark session, and write ACID, managed tables to Apache Hive.

Before you begin

- Accept the default spark.datasource.hive.warehouse.load.staging.dir for the temporary staging location required by HWC.
- Check that spark.hadoop.hive.zookeeper.quorum is configured.
- Set Kerberos configurations for HWC, or for an unsecured cluster, set spark.security.credentials.hiveserver2.enabled=false.

About this task

The way data is written from HWC is not impacted by the read modes configured for HWC. For write operations, HWC writes to an intermediate location (as defined by the value of config spark.datasource.hive.warehouse.load.staging.dir) from Spark, followed by executing a "LOAD DATA" query in hive via JDBC. Exception: writing to dynamic partitions creates and intermediate temporary external table.

Using HWC to write data is recommended for production in Cloudera.

Procedure

1. Open a terminal window, start the Apache Spark session, and include the URL for HiveServer.

```
spark-shell --jars /opt/cloudera/parcels/CDH/jars/hive-warehouse-connector-assembly-<version>.jar \
-- conf spark.sql.hive.hiveserver2.jdbc.url=<JDBC endpoint for HiveServer>
...
```

2. Include in the launch string a configuration of the intermediate location to use as a staging directory.
   Example syntax:

```
...
--conf spark.datasource.hive.warehouse.load.staging.dir=<path to directory>
```

3. Write a Hive managed table.
   For example, in Scala:

```
import com.hortonworks.hwc.HiveWarehouseSession
import com.hortonworks.hwc.HiveWarehouseSession._

val hive = HiveWarehouseSession.session(spark).build();
hive.setDatabase("tpcds_bin_partitioned_orc_1000");
val df = hive.sql("select * from web_sales");
df.createOrReplaceTempView("web_sales");
hive.setDatabase("testDatabase");
hive.createTable("newTable").ifNotExists()
 .column("ws_sold_time_sk", "bigint")
 .column("ws_ship_date_sk", "bigint")
 .create();
sql("SELECT ws_sold_time_sk, ws_ship_date_sk FROM web_sales WHERE ws_sold_time_sk > 80000)
 .write.format(HIVE_WAREHOUSE_CONNECTOR)
 .mode("append")
 .option("table", "newTable")
```

```
 .save();
```

HWC internally fires the following query to Hive through JDBC:

```
LOAD DATA INPATH '<spark.datasource.hive.warehouse.load.staging.dir>' INTO
 TABLE tpcds_bin_partitioned_orc_1000.newTable
```

4. Write to a statically partitioned, Hive managed table named t1 having two partitioned columns c1 and c2.

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("partiti
on", "c1='val1',c2='val2'").option("table", "t1").save();
```

HWC internally fires the following query to Hive through JDBC after writing data to a temporary location.

```
LOAD DATA INPATH '<spark.datasource.hive.warehouse.load.staging.dir>' [O
VERWRITE] INTO TABLE db.t1 PARTITION (c1='val1',c2='val2');
```

5. Write to a dynamically partitioned table named t1 having two partitioned cols c1 and c2.

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("partiti
on", "c1='val1',c2").option("table", "t1").save();
```

HWC internally fires the following query to Hive through JDBC after writing data to a temporary location.

```
CREATE TEMPORARY EXTERNAL TABLE db.job_id_table(cols....) STORED AS ORC
LOCATION '<spark.datasource.hive.warehouse.load.staging.dir>';

INSERT INTO TABLE t1 PARTITION (c1='val1',c2)  SELECT <cols> FROM db.job
_id_table;
```

where <cols> should have comma separated list of columns in the table with dynamic partition columns being the last in the list and in the same order as the partition definition.

Related Information

Using Direct Reader mode

Using JDBC read mode

Apache Spark executor task statistics

Apache Spark executor task statistics

You can view Spark executor task statistics, such as the read or write metrics when you use Hive Warehouse Connector (HWC) to query Hive managed tables from Spark. These metrics enable you to view information about running Spark executors and help in troubleshooting performance issues.

For example, you can view the write metrics — .bytesWritten and .recordsWritten to understand the amount of data and number of records written to Spark. You can choose to view these metrics either at a granular task level or at an aggregated level.

For the complete list of Spark Executor Task metrics, see Apache Spark documentation.

You can view these metrics through one of the following ways:

- Using the Spark UI
- Using Spark REST APIs
- Using Spark Listeners

Using the Spark UI

You can use the Web UI that is available for each Spark context to monitor the status and resource consumption of your Spark cluster. You can navigate to the Stages tab of the Spark UI to view the current state of all stages of all jobs in the Spark application, and view the list of executor task metrics.

The Details for Stage pane displays the metrics at an overall Stage level and the Aggregated Metrics by Executor table displays metrics at an individual task level.

Using Spark REST APIs

Apart from viewing metrics through the Web UI, you can use the appropriate Spark REST APIs that returns a result set containing metrics in JSON format. You can then use the JSON to create new visualizations and monitoring tools for your Spark application.

The metrics in this JSON can be viewed either at a task level or at higher levels, such as stage or jobs. For example, the following API displays a list of all tasks for a specific stage of a YARN application:

https://<spark-ui>:port/proxy/[app-id]/api/v1/applications/[app-id]/stages/[stage-id]

The API returns a JSON that contains a summary of the statistics aggregated at a stage level and individual statistics for each task. A sample extract of the JSON is provided below:

```
"executorSummary" : {
    "driver" : {
      "taskTime" : 309,
      "failedTasks" : 0,
      "succeededTasks" : 4,
      "killedTasks" : 0,
      "inputBytes" : 0,
      "inputRecords" : 0,
      "outputBytes" : 1412,
      "outputRecords" : 16,
      "shuffleRead" : 0,
      "shuffleReadRecords" : 0,
      "shuffleWrite" : 0,
      "shuffleWriteRecords" : 0,
      "memoryBytesSpilled" : 0,
      "diskBytesSpilled" : 0,
      "isBlacklistedForStage" : false
    }
  },
```

The outputBytes and outputRecords metrics in the JSON extract correspond to the amount of data and number of records written by all tasks in a specified stage.

Using Spark Listeners

You can add Spark Listeners to a required event (completion of job or stage) to view metrics and to monitor your Spark application while the application is still running. Listeners intercept events from the Spark Scheduler to give you useful information at the end of each event.

The following code snippet shows how you can add a Spark Listener to view the bytesWritten and recordsWritten metrics at the end of a task:

```
import org.apache.spark.scheduler.{SparkListener, SparkListenerTaskEnd}
var recordsWrittenCount = 0L
var bytesWrittenCount = 0L
sc.addSparkListener(new SparkListener() {
override def onTaskEnd(taskEnd: SparkListenerTaskEnd) {
 synchronized {
recordsWrittenCount += taskEnd.taskMetrics.outputMetrics.recordsWritten
 bytesWrittenCount += taskEnd.taskMetrics.outputMetrics.bytesWritten
```

```
  }
}})
```

Related Information

Apache Spark documentation

Introduction to HWC and DataFrame APIs

As an Apache Spark developer, you learn the code constructs for executing Apache Hive queries using the HiveWarehouseSession API. In Spark source code, you see how to create an instance of HiveWarehouseSession. You also learn how to access a Hive ACID table using DataFrames.

Supported APIs

- Spark SQL

  Supports built-in Spark SQL query read (only) patterns. Output conforms to built-in spark.sql conventions.

  Example

  ```
  $ spark-shell <parameters to specify HWC jar and config settings>
  scala> sql("select * from managedTable").show
  scala> spark.read.table("managedTable").show
  ```

- HWC

  Supports HiveWarehouse Session API operations using the HWC sql API. The .execute() and .executeQuery() methods are deprecated.

  Example

  ```
  scala> val hive = com.hortonworks.hwc.HiveWarehouseSession.session(spark
  ).build()
  scala> hive.sql("select * from emp_acid").show
  scala> hive.sql("select e.emp_id, e.first_name, d.name department from
  emp_acid e join dept_ext d on e.dept_id = d.id").show
  ```

  Note: hive.sql() is used to execute read operations and does not support write operations, such as INSERT, UPDATE, and DELETE.

- DataFrames

  Supports accessing a Hive ACID table from Scala, or pySpark, directly using DataFrames. Direct reads and writes from the file are not supported.

  Hive ACID tables are tables in Hive metastore and must be formatted using DataFrames as follows:

  Syntax

  ```
  val df = hive.sql("<SELECT query>")
  ```

  Example

  ```
  scala> val df = hive.sql("select * from managedTable where a=100")
  scala> df.collect()
  ```

Import statements and variables

The following string constants are defined by the API:

- HIVE_WAREHOUSE_CONNECTOR

- DATAFRAME_TO_STREAM
- STREAM_TO_STREAM

Assuming spark is running in an existing SparkSession, use this code for imports:

- Scala

```
import com.hortonworks.hwc.HiveWarehouseSession
import com.hortonworks.hwc.HiveWarehouseSession._
val hive = HiveWarehouseSession.session(spark).build()
```

- Java

```
import com.hortonworks.hwc.HiveWarehouseSession;
import static com.hortonworks.hwc.HiveWarehouseSession.*;
HiveWarehouseSession hive = HiveWarehouseSession.session(spark).build();
```

- Python

```
from pyspark_llap import HiveWarehouseSession
hive = HiveWarehouseSession.session(spark).build()
```

Executing queries

HWC supports the hive.sql() API for executing queries. You can also use the Spark SQL to query Hive managed tables, however, it is recommened that you use the HWC sql method.

- .sql()
  - Executes queries in all the read modes — Direct Reader, JDBC, and Secure access modes.
  - Consistent with the Spark sql interface.
  - Masks the internal implementation based on the cluster type you configured.
  - Used to execute read operations and does not support write operations, such as INSERT, UPDATE, and DELETE.

Results are returned as a DataFrame to Spark.

Note: The .execute() and .executeQuery() methods are deprecated and it is recommended that you use the hive.sql() method.

Support of HWC read modes on Hive tables or views

The following table represents the different Hive tables or views that are supported by the various HWC read modes:

Mode vs Table	Full ACID table	Insert-only ACID table	View (created on a managed table)	Materialized view (created on a managed table)
DIRECT_READER_V1	Yes	Yes	Yes	Yes
DIRECT_READER_V2	Yes	Yes	Yes	Yes
JDBC_CLUSTER	Yes	Yes	Yes	Yes
SECURE_ACCESS	Yes	Yes	Yes	Yes

It is recommended that you do not use Managed non-transactional tables. Such tables should ideally be converted to external tables.

Support of HWC read modes on table formats

The following table formats are supported by HWC while reading a table:

Mode	ORC	Parquet	Avro	Textfile
DIRECT_READER_V1	Yes	Yes	Yes	Yes
DIRECT_READER_V2	Yes	Yes	Yes	Yes
JDBC_CLUSTER	Yes	Yes	Yes	Yes
SECURE_ACCESS	Yes	Yes	Yes	Yes

hive.sql vs. spark.sql

There are a number of important differences between the hive.sql and spark.sql functions:

- hive.sql() is explicitly defined in HWC and can be used across all read modes to query Apache Hive managed tables (full ACID and insert-only ACID tables).
- spark.sql() can also be used across all the read modes to query an Apache Hive managed table. However, it is recommended that you use hive.sql() over spark.sql().
- The Direct Reader imposes the constraint that the Hive table must be transactional.

Related Information

HMS storage

Orc vs Parquet

HWC and DataFrame API limitations

These limitations are in addition to Direct Reader mode, JDBC mode, Secure access mode, and HWC and DataFrames API limitations.

- Table stats (basic stats and column stats) are not generated when you write a DataFrame to Hive.
- When the HWC API save mode is overwrite, writes are limited.

  You cannot read from and overwrite the same table. If your query accesses only one table and you try to overwrite that table using an HWC API write method, a deadlock state might occur. Do not attempt this operation.

  Example: Operation Not Supported

  ```
  scala> val df = hive.executeQuery("select * from t1")
  scala> df.write.format("com.hortonworks.spark.sql.hive.llap.HiveWarehouseC
  onnector"). \
  mode("overwrite").option("table", "t1").save
  ```

- Automatic table creation during HWC write operation is not supported in Spark 3.

  If you are using Spark 3 and performing a DataFrame write using HWC with the SaveMode.Append or SaveMode .Overwrite modes, the write operation fails if the table you are writing to does not exist. These modes can save a DataFrame only if the table you are writing to, already exists.

  For example, the following statement tries to append to a table, "ex2" that does not exist:

  ```
  df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("table",
   "ex2").save()
  ```

  The statement results in the following exception:

  ```
  22/05/26 12:01:12 ERROR llap.HiveWarehouseConnector: Failed to connect to
   HS2
  org.apache.hive.service.cli.HiveSQLException: Error while compiling statem
  ent: FAILED: SemanticException [Error 10001]: Line 1:29 Table not found
  'ex2'
   at org.apache.hive.jdbc.Utils.verifySuccess(Utils.java:373)
   at org.apache.hive.jdbc.Utils.verifySuccessWithInfo(Utils.java:359)
   at org.apache.hive.jdbc.HiveStatement.runAsyncOnServer(HiveStatement.ja
  va:334)
  ```

```
at org.apache.hive.jdbc.HiveStatement.execute(HiveStatement.java:275)
      ...
      ...
```

Alternatively, you can use the SaveMode.ErrorIfExists or SaveMode.Ignore modes to save DataFrames to a table.

HWC supported types mapping

To create HWC API apps, you must know how Hive Warehouse Connector maps Apache Hive types to Apache Spark types, and vice versa. Awareness of a few unsupported types helps you avoid problems.

Spark-Hive supported types mapping

The following types are supported by the HiveWareHouseConnector library:

Spark Type	Hive Type
ByteType	TinyInt
ShortType	SmallInt
IntegerType	Integer
LongType	BigInt
FloatType	Float
DoubleType	Double
DecimalType	Decimal
StringType*	String, Varchar*
BinaryType	Binary
BooleanType	Boolean
TimestampType**	Timestamp**
DateType	Date
ArrayType	Array
StructType	Struct

Notes:

* StringType (Spark) and String, Varchar (Hive)

A Hive String or Varchar column is converted to a Spark StringType column. When a Spark StringType column has maxLength metadata, it is converted to a Hive Varchar column; otherwise, it is converted to a Hive String column.

** Timestamp (Hive)

The Hive Timestamp column loses submicrosecond precision when converted to a Spark TimestampType column because a Spark TimestampType column has microsecond precision, while a Hive Timestamp column has nanosecond precision.

Hive timestamps are interpreted as UTC. When reading data from Hive, timestamps are adjusted according to the local timezone of the Spark session. For example, if Spark is running in the America/New_York timezone, a Hive timestamp 2018-06-21 09:00:00 is imported into Spark as 2018-06-21 05:00:00 due to the 4-hour time difference between America/New_York and UTC.

Spark-Hive unsupported types

Spark Type	Hive Type
CalendarIntervalType	Interval
N/A	Char

Spark Type	Hive Type
MapType	Map
N/A	Union
NullType	N/A
TimestampType	Timestamp With Timezone

Related Information

HMS storage

Catalog operations

Short descriptions and the syntax of catalog operations, which include creating, dropping, and describing an Apache Hive database and table from Apache Spark, helps you write HWC API apps.

Methods

.sql is the recommended method for executing catalog operations.

- Set the current database for unqualified Hive table references

  hive.setDatabase(<database>)
- Run a catalog operation and return a DataFrame

  hive.sql("describe extended web_sales").show()
- Show databases

  hive.showDatabases().show(100)
- Show tables for the current database

  hive.showTables().show(100)
- Describe a table

  hive.describeTable(<table_name>).show(100)
- Create a database

  hive.createDatabase(<database_name>,<ifNotExists>)
- Create an ORC table

  ```
  hive.createTable("web_sales").ifNotExists().column("sold_time_sk", "bigi
  nt").column("ws_ship_date_sk", "bigint").create()
  ```

  See the CreateTableBuilder interface section below for additional table creation options. You can also create Hive tables using hive.executeUpdate.
- Drop a database

  hive.dropDatabase(<databaseName>, <ifExists>, <useCascade>)
- Drop a table

  hive.dropTable(<tableName>, <ifExists>, <usePurge>)

Related Information

HMS storage

Read and write operations

Brief descriptions of HWC API operations and examples cover how to read and write Apache Hive tables from Apache Spark. You learn how to update statements and write DataFrames to partitioned Hive tables, perform batch writes, and use HiveStreaming.

Read operations

Run a Hive SELECT query and return a DataFrame.

hive.sql("select * from web_sales")

HWC supports push-downs of DataFrame filters and projections applied to .sql().

Run a Hive update statement

Run CREATE, UPDATE, DELETE, INSERT, and MERGE statements in this way:

hive.executeUpdate("ALTER TABLE old_name RENAME TO new_name")

Write a DataFrame to Hive in batch

This operation uses LOAD DATA INTO TABLE.

Java/Scala:

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).option("table", <tableName>).s
ave()
```

Python:

```
df.write.format(HiveWarehouseSession().HIVE_WAREHOUSE_CONNECTOR).option("tab
le", &tableName>).save()
```

Write a DataFrame to Hive, specifying partitions

HWC follows Hive semantics for overwriting data with and without partitions and is not affected by the setting of spark.sql.sources.partitionOverwriteMode to static or dynamic. This behavior mimics the latest Spark Community trend reflected in Spark-20236 (link below).

Java/Scala:

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).option("table", <tableName>).optio
n("partition", <partition_spec>).save()
```

Python:

```
df.write.format(HiveWarehouseSession().HIVE_WAREHOUSE_CONNECTOR).option("tab
le", &tableName>).option("partition", <partition_spec>).save()
```

Where <partition_spec> is in one of the following forms:

- option("partition", "c1='val1',c2=val2") // static
- option("partition", "c1='val1',c2") // static followed by dynamic
- option("partition", "c1,c2") // dynamic

Depending on the partition spec, HWC generates queries in one of the following forms for writing data to Hive.

- No partitions specified = LOAD DATA
- Only static partitions specified = LOAD DATA...PARTITION
- Some dynamic partition present = CREATE TEMP TABLE + INSERT INTO/OVERWRITE query.

Note: Writing static partitions is faster than writing dynamic partitions.

Write a DataFrame to Hive using HiveStreaming

When using HiveStreaming to write a DataFrame to Hive or a Spark Stream to Hive, you need to escape any commas in the stream.

Java/Scala:

```
//Using dynamic partitioning
df.write.format(DATAFRAME_TO_STREAM).option("table", <tableName>).save()

//Or, writing to a static partition
df.write.format(DATAFRAME_TO_STREAM).option("table", <tableName>).option("p
artition", <partition>).save()
```

Python:

```
//Using dynamic partitioning
df.write.format(HiveWarehouseSession().DATAFRAME_TO_STREAM).option("table",
 <tableName>).save()

//Or, writing to a static partition
df.write.format(HiveWarehouseSession().DATAFRAME_TO_STREAM).option("table",
<tableName>).option("partition", <partition>).save()
```

Write a Spark Stream to Hive using HiveStreaming

Java/Scala:

```
stream.writeStream.format(STREAM_TO_STREAM).option("table", "web_sales").sta
rt()
```

Python:

```
stream.writeStream.format(HiveWarehouseSession().STREAM_TO_STREAM).option("t
able", "web_sales").start()
```

Related Information

HMS storage

SPARK-20236

Committing a transaction for Direct Reader

For Direct Reader operations, you need to know how to commit or abort transactions.

About this task

A sql listener normally handles this task automatically when a dataframe operation or spark sql query finishes. In some cases when .explain() , .rdd() , or .cache() are invoked on a dataframe, the transaction is not automatically closed. In Spark Direct Reader mode, commit or abort a transaction as follows:

```
scala> com.qubole.spark.hiveacid.transaction.HiveAcidTxnManagerObject.commit
Txn(spark)
scala> hive.commitTxn
```

Or, if you are using Hive Warehouse Connector with Direct Reader Mode enabled, you can invoke following API to commit transaction:

```
 scala> hive.commitTxn
```

Closing HiveWarehouseSession operations

You need to know how to release locks that Apache Spark operations puts on Apache Hive resources. An example shows how and when to release these locks.

About this task

Spark can invoke operations, such as cache(), persist(), and rdd(), on a DataFrame you obtain from running a HiveWarehouseSession .table() or .sql(). The Spark operations can lock Hive resources. You can release any locks and resources by calling the HiveWarehouseSession close(). Calling close() invalidates the HiveWarehouseSession instance and you cannot perform any further operations on the instance.

Procedure

Call close() when you finish running all other operations on the instance of HiveWarehouseSession.

```
import com.hortonworks.hwc.HiveWarehouseSession
import com.hortonworks.hwc.HiveWarehouseSession._
val hive = HiveWarehouseSession.session(spark).build()
hive.setDatabase("tpcds_bin_partitioned_orc_1000")
val df = hive.sql("select * from web_sales")
. . .  //Any other operations
.close()
```

You can also call close() at the end of an iteration if the application is designed to run in a microbatch, or iterative, manner that does not need to share previous states.

No more operations can occur on the DataFrame obtained by table() or sql().

Using HWC for streaming

When using HiveStreaming to write a DataFrame to Apache Hive or an Apache Spark Stream to Hive, you need to know how to escape any commas in the stream because the Hive Warehouse Connector uses the commas as the field delimiter.

Procedure

Change the value of the default delimiter property escape.delim to a backslash that the Hive Warehouse Connector uses to write streams to mytable.
ALTER TABLE mytable SET TBLPROPERTIES ('escape.delim' = '\\');

Related Information

HMS storage

Hive Warehouse Connector streaming for transactional tables

Learn how to use Hive Warehouse Connector to stream data from Apache Spark into transactional Hive tables.

Running HWC Streaming

Hive Warehouse Connector supports streaming data from Apache Spark into Hive tables, enabling real-time data ingestion. However, the target Hive table must be transactional to use this feature. Use the steps below to set up and run streaming with Hive Warehouse Connector.

1. Create the Hive Table

   Before writing streaming data, pre-create the target Hive table. This table must be transactional and accessible to the Spark session. Example:

   ```
   CREATE TABLE spark_rate_source(`timestamp` STRING, value BIGINT);
   ```

2. Configure and Write Streaming Data from Spark

   The following example demonstrates streaming data to the Hive table using Spark's inbuilt rate source, ideal for testing.

   - Spark streaming requires a checkpoint location to manage state.

     ```
     spark.conf.set("spark.sql.streaming.checkpointLocation", "/tmp/spark_che
     ckpoint");
     ```

   - Use Spark's rate source to generate rows at a configurable rate.

     ```
     val rateDF = spark.readStream.format("rate").option("rowsPerSecond", 1).
     load
     ```

   - Use Hive Warehouse Connector's streaming mode (STREAM_TO_STREAM) to write data directly to the Hive table.

     ```
     import org.apache.spark.sql.streaming.Trigger
     import com.hortonworks.hwc.HiveWarehouseSession
     import com.hortonworks.hwc.HiveWarehouseSession._
     val query = rateDF.writeStream .format(STREAM_TO_STREAM).outputMode("app
     end").option("metastoreUri", "thrift://<metastore-host>:9083").option("d
     atabase", "default").option("table", "spark_rate_source").trigger(Trigge
     r.ProcessingTime("1 seconds")).start
     ```

Key Requirements for Streaming

- Transactional Tables: The Hive table must be transactional and pre-created.
- HMS Connection: Hive Warehouse Connector connects to the Hive Metastore Service to initiate transactions, obtain write IDs, and fetch file locations.

File System Access: The Spark session must have write access to the table's file system location.

Note: The managed tables are typically restricted to the Hive user to maintain ACID v2 consistency.

Managing streaming with Hive Warehouse Connector

Understand how Hive Warehouse Connector uses HMS for transaction management and directly writes ORC files to Hive table locations without relying on HiveServer2.

Hive Warehouse Connector does not rely on HiveServer2 for streaming. Instead, it interacts with HMS for transaction management and writes ORC bucket files directly to the table's location.

An example of using the DATAFRAME_TO_STREAM method for non-streaming writes:

```
myDF.write.format(DATAFRAME_TO_STREAM)
  .option("metastoreUri", "thrift://jkovacs-1.jkovacs.root.hwx.site:9083")
  .option("metastoreKrbPrincipal", "hive/_HOST@AD.HALXG.CLOUDERA.COM")
  .option("database", "default")
  .option("table", "hwctest")
  .save()
```

Important Notes:

- Always pre-create the Hive table before writing to it.
- Ensure that the Spark session user has appropriate permissions for the table's file system location.
- Verify that the Hive Metastore URI is correctly configured in the options.

By following these steps, you can leverage Hive Warehouse Connector to efficiently stream data into Hive tables using Spark.

HWC API Examples

Examples of using the HWC API include how to create the DataFrame from any data source and include an option to write the DataFrame to an Apache Hive table.

Write a DataFrame from Spark to Hive example

You specify one of the following Spark SaveMode modes to write a DataFrame to Hive:

- Append
- ErrorIfExists
- Ignore
- Overwrite

In Overwrite mode, HWC does not explicitly drop and recreate the table. HWC queries Hive to overwrite an existing table using LOAD DATA...OVERWRITE or INSERT OVERWRITE...

When you write the DataFrame, the Hive Warehouse Connector creates the Hive table if it does not exist.

Important: Spark 3 does not support automatic table creation using the Append and Overwrite modes. The table that you are writing to must exist in order to append or overwrite data in it. Instead, you can use the ErrorIfExists or Ignore modes.

The following example uses Append mode.

```
df = //Create DataFrame from any source


val hive = com.hortonworks.hwc.HiveWarehouseSession.session(spark).build()

df.write.format(HIVE_WAREHOUSE_CONNECTOR)
.mode("append")
.option("table", "my_Table")
.save()
```

ETL example (Scala)

Read table data from Hive, transform it in Spark, and write to a new Hive table.

```
import com.hortonworks.hwc.HiveWarehouseSession
import com.hortonworks.hwc.HiveWarehouseSession._
val hive = HiveWarehouseSession.session(spark).build()
hive.setDatabase("tpcds_bin_partitioned_orc_1000")
val df = hive.sql("select * from web_sales")
df.createOrReplaceTempView("web_sales")
hive.setDatabase("testDatabase")
hive.createTable("newTable")
.ifNotExists()
.column("ws_sold_time_sk", "bigint")
.column("ws_ship_date_sk", "bigint")
.create()
sql("SELECT ws_sold_time_sk, ws_ship_date_sk FROM web_sales WHERE ws_sold_
time_sk > 80000)
.write.format(HIVE_WAREHOUSE_CONNECTOR)
.mode("append")
.option("table", "newTable")
.save()
```

Related Information

HMS storage

Hive Warehouse Connector Interfaces

The HiveWarehouseSession, CreateTableBuilder, and MergeBuilder interfaces present available HWC operations.

HiveWarehouseSession interface

```
package com.hortonworks.hwc;

public interface HiveWarehouseSession {

//Execute Hive SELECT query and return DataFrame (recommended)
Dataset<Row> sql(String sql);
//Execute Hive SELECT query and return DataFrame in JDBC client mode (depre
cated)
//Execute Hive catalog-browsing operation and return DataFrame (deprecated)
Dataset<Row> execute(String sql);

//Execute Hive SELECT query and return DataFrame in LLAP mode (not available
 in this release)(deprecated)
Dataset<Row> executeQuery(String sql);

//Execute Hive update statement
boolean executeUpdate(String sql);

//Reference a Hive table as a DataFrame
Dataset<Row> table(String sql);

//Return the SparkSession attached to this HiveWarehouseSession
SparkSession session();

//Set the current database for unqualified Hive table references
void setDatabase(String name);

/**
* Helpers: wrapper functions over execute or executeUpdate
*/

//Helper for show databases
Dataset<Row> showDatabases();

//Helper for show tables
Dataset<Row> showTables();

//Helper for describeTable
Dataset<Row> describeTable(String table);

//Helper for create database
void createDatabase(String database, boolean ifNotExists);

//Helper for create table stored as ORC
CreateTableBuilder createTable(String tableName);

//Helper for drop database
void dropDatabase(String database, boolean ifExists, boolean cascade);

//Helper for drop table
void dropTable(String table, boolean ifExists, boolean purge);

//Helper for merge query
MergeBuilder mergeBuilder();

//Closes the HWC session. Session cannot be reused after being closed.
void close();
```

```
// Closes the transaction started by the direct reader. The transaction is n
ot committed if user
// uses rdd APIs.
void commitTxn();
}
```

CreateTableBuilder interface

```
package com.hortonworks.hwc;

public interface CreateTableBuilder {

//Silently skip table creation if table name exists
CreateTableBuilder ifNotExists();

//Add a column with the specific name and Hive type
//Use more than once to add multiple columns
CreateTableBuilder column(String name, String type);

//Specific a column as table partition
//Use more than once to specify multiple partitions
CreateTableBuilder partition(String name, String type);

//Add a table property
//Use more than once to add multiple properties
CreateTableBuilder prop(String key, String value);

//Make table bucketed, with given number of buckets and bucket columns
CreateTableBuilder clusterBy(long numBuckets, String ... columns);

//Creates ORC table in Hive from builder instance
void create();
}
```

MergeBuilder interface

```
package com.hortonworks.hwc;

public interface MergeBuilder {

//Specify the target table to merge
MergeBuilder mergeInto(tring targetTable, String alias);

//Specify the source table or expression, such as (select * from some_table)
// Enclose expression in braces if specified.
MergeBuilder using(String sourceTableOrExpr, String alias);

//Specify the condition expression for merging
MergeBuilder on(String expr);

//Specify fields to update for rows affected by merge condition and match
Expr
MergeBuilder whenMatchedThenUpdate(String matchExpr, String... nameValuePa
irs);

//Delete rows affected by the merge condition and matchExpr
MergeBuilder whenMatchedThenDelete(String matchExpr);

//Insert rows into target table affected by merge condition and matchExpr
MergeBuilder whenNotMatchedInsert(String matchExpr, String... values);
```

```
//Execute the merge operation
void merge();
}
```

Related Information

HMS storage

Submitting a Scala or Java application

A step-by-step procedure shows you how to submit an app based on the HiveWarehouseConnector library to run on Apache Spark Shell.

Procedure

1. Choose a mode, for example JDBC, for your application.
2. Check that you meet the prerequisites for using JDBC read mode and make the HiveServer (HS2) connection as described for using the JDBC read mode.
3. Locate the hive-warehouse-connector-assembly jar in the /hive_warehouse_connector/ directory.
   For example, find hive-warehouse-connector-assembly-<version>.jar in the following location:

   ```
   /opt/cloudera/parcels/CDH/jars
   ```

4. Add the connector jar and configurations to the app submission using the --jars option.
   Example syntax:

   ```
   spark-shell --jars <path to jars>/hive_warehouse_connector/hive-warehouse-
   connector-assembly-<version>.jar \
   --conf <configuration properties>
   ```

5. Add the path to app you wrote based on the HiveWarehouseConnector API.
   Example syntax:

   ```
    <path to app>
   ```

   For example:

   ```
   spark-shell --jars /opt/cloudera/parcels/CDH/jars/hive-warehouse-connect
   or-assembly-<version>.jar \
   --conf spark.sql.extensions=com.hortonworks.spark.sql.rule.Extensions \
   --conf spark.datasource.hive.warehouse.read.mode=JDBC_CLUSTER \
   --conf "spark.hadoop.hive.metastore.uris=thrift://172.27.74.137:9083" \
   --conf spark.datasource.hive.warehouse.load.staging.dir=<path to directo
   ry> \
   /home/myapps/myapp.jar
   ```

Related Information

Using Direct Reader mode

Using JDBC read mode

HMS storage

Examples of writing data in various file formats

Hive Warehouse Connector (HWC) enables you to write to tables in various formats, such as Parquet, ORC, AVRO, and Textfile. You see by example how to write a Dataframe in these formats, to a pre-existing or new table.

Initialize

import com.hortonworks.hwc.HiveWarehouseSession

import com.hortonworks.hwc.HiveWarehouseSession._

val hive = HiveWarehouseSession.session(spark).build()

Create a Dataframe

val df = Seq((1, "bat"), (2, "mouse"), (3, "horse")).toDF("id", "name")

df.show

Table does not exist

Examples of a Dataframe write to new tables:

- Dataframe write in Parquet

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("table",
    "parquet_table").option("fileformat","parquet").save()
```

- Dataframe write in ORC

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("table",
    "orc_table").option("fileformat","orc").save()
```

- Dataframe write in Avro

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("table",
    "avro_table").option("fileformat","avro").save()
```

- Dataframe write in Textfile
  - With Default Field Delimiter (,)

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("table",
    "text_table1").option("fileformat","textfile").save()
```

  - With Custom Field Delimiter (*)

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("table",
    "text_table2").option("fileformat","textfile").option("sep","*").sa
ve()
```

Important: Spark 3 does not support automatic table creation using the Append and Overwrite modes. The table that you are writing to must exist in order to append or overwrite data in it. Instead, you can use the ErrorIfExists or Ignore modes.

Table already exists

If you already have a table, you do not need to specify a file format, but you can as shown in the following examples:

Without file format specification

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("table",
"parquet_table").save()
```

With file format specification

Results differ depending on whether the file format specification matches that of the table or not. If there is a mismatch in the file format, an exception is displayed.

A match succeeds

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("table",
"parquet_table").option("fileformat","parquet").save()
```

Throws exception if there is a mismatch in the file format

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("table",
"parquet_table").option("fileformat","orc").save()
```

Default Table Format is ORC

If you do not specify a file format for the table, the table is created in ORC format.

```
df.write.format(HIVE_WAREHOUSE_CONNECTOR).mode("append").option("table", "sa
mple").save()
```

To change the default format, use the set method:

```
sql("set spark.datasource.hive.warehouse.default.write.format=parquet")
```

You can specify properties as options as follows:

.option("compression", "SNAPPY")

.option("transactional", "false")

HWC integration pyspark, sparklyr, and Zeppelin

Submitting a Python app

A step-by-step procedure shows you how to submit a Python app based on the HiveWarehouseConnector library by submitting an application, and then adding a Python package.

Procedure

1. Choose a read option, for example LLAP, for your application and check that you meet the configuration requirements, described earlier.
2. Configure a Spark-HiveServer connection, described earlier or, in your app submission include the appropriate --conf in step 4.
3. Locate the hive-warehouse-connector-assembly jar in the /hive_warehouse_connector/ directory.
   For example, find hive-warehouse-connector-assembly-<version>.jar in the following location:

   ```
   /opt/cloudera/parcels/CDH/jars
   ```

4. Add the connector jar and configurations to the app submission using the --jars option.
   Example syntax:

   ```
   pyspark --jars <path to jars>/hive_warehouse_connector/hive-warehouse-co
   nnector-assembly-<version>.jar \
   --conf <configuration properties>
   ```

5. Locate the pyspark_hwc zip package in the /hive_warehouse_connector/ directory.
6. Add the Python package for the connector to the app submission.
   Example syntax:

   ```
   --py-files <path>/hive_warehouse_connector/pyspark_hwc-<version>.zip
   ```

   Example submission in JDBC execution mode:

   ```
   pyspark --jars /opt/cloudera/parcels/CDH/jars/hive-warehouse-connector-a
   ssembly-<version>.jar
   ```

```
--conf spark.sql.extensions=com.hortonworks.spark.sql.rule.Extensions \
--conf spark.datasource.hive.warehouse.read.mode=JDBC_CLUSTER \
--conf spark.datasource.hive.warehouse.load.staging.dir=<path to directory
> \
--py-files /opt/cloudera/parcels/CDH/lib/hive_warehouse_connector/pyspar
k_hwc-<version>.zip
```

Related Information

Using Direct Reader mode

Using JDBC read mode

HMS storage

Reading and writing Hive tables in R

The Hive Warehouse Connector (HWC) supports reads and writes to Apache Hive managed ACID tables in R. Cloudera provides an R package SparklyrHWC that includes all HWC methods, such as hive.sql, and a spark_write_table method to write to managed tables. The native sparklyr spark_write_table method supports writes to external tables only.

Support

HWC should work with Sparklyr 1.0.4. Versions later than 1.0.4 should also work if interfaces are not changed by sparklyr. However, sparklyr isn't supported by Cloudera. We will support any issues around using HWC from sparklyr.

Downloading SparklyrHWC

You can download the SparklyrHWC R package that includes HWC methods from your Cloudera Cluster. Go to /opt/cloudera/parcels/CDH/lib/hive_warehouse_connector/. Copy the SparklyrHWC to your download location.

About this task

To read and write Hive tables in R, you need to configure an HWC execution mode. You can configure either one of the following HWC execution modes in R:

- JDBC mode
  - Suitable for writing production workloads.
  - Suitable for reading production workloads having a data size of 1 GB or less.
  - Use this mode for reading if latency is not an issue.
- Spark-ACID mode
  - Suitable for reading production workloads.
  - Does not support writes

Reading and writing managed tables

You can read Hive managed tables using either JDBC or Spark-ACID mode. The mode you configure affects the background process. You use the same R code regardless of the mode with one exception: You do not need to call the commitTxn(hs) when using JDBC mode.

To write to Hive managed tables, you must connect to HWC in JDBC mode.

Reading and writing external tables

You can read and write Hive external tables in R using the sparklyr package. HWC is not required.

In the following procedure, you configure Spark-Acid execution mode to read tables on a production cluster. You use the native sparklyr spark_read_table and spark_load_table to read Hive managed tables in R.

Reading a Hive managed table example

Procedure

1. Read tables using spark-acid - direct_reader_v2 mode.
   For example:

```
library(sparklyr)

// Configurations needed  to use spark-acid
config <- spark_config()
config$sparklyr.jars.default <- "<path to hwc jar>"
config$spark.sql.hive.hiveserver2.jdbc.url="jdbc:hive2://<url>:10000/defau
lt"
config$spark.datasource.hive.warehouse.user.name="hive"
config$spark.hadoop.hive.metastore.uris="thrift://<url>:9083"
config$spark.sql.extensions="com.hortonworks.spark.sql.rule.Extensions"
config$spark.kryo.registrator="com.qubole.spark.hiveacid.util.HiveAcidKyro
Registrator"
config$spark.datasource.hive.warehouse.read.mode="DIRECT_READER_V2"

//Set env variables

Sys.setenv(SPARK_HOME = "/opt/cloudera/parcels/CDH/lib/spark/")
Sys.setenv(HADOOP_HOME = "/opt/cloudera/parcels/CDH/lib/hadoop")
// Spark context can be set to local or yarn mode
sc <- spark_connect(master = "local",config = config)
```

2. Read Hive using sparklyr methods spark_read_table() or spark_load_table().
   For example:

```
// reading a managed table using spark acid direct-reader

> intDf <- sparklyr::spark_read_table(sc, 'emp_hwc')
>  sparklyr::sdf_collect(intDf1) // Converts SparkDataframe to R dataframe
```

3. Explicitly commit your code. For example, use SparklyrHWC::commitTxn(hs), where hs is the HiveWarehouseSessionImpl object.

   You need to commit your code using commitTxn() due to sparklyr calling specific cache methods internally.

Writing a Hive managed table in R

About this task

You configure JDBC mode and use and SparklyrHWC::spark_write_table() to write to a managed table.

Procedure

1. In R, configure JDBC mode by setting the following configs and values:

```
> config$spark.sql.extensions="com.hortonworks.spark.sql.rule.Extensions"
> config$spark.datasource.hive.warehouse.read.via.llap="false"
> config$spark.datasource.hive.warehouse.read.jdbc.mode="cluster"
```

2. Include the sparklyr library.

```
> library(sparklyr)  // Load sparklyr library
```

3. Write to a managed table using SparklyrHWC::spark_write_table.

```
> library(sparklyr)  // Load sparklyr library
```

```
> Sys.setenv(HADOOP_HOME = "/opt/cloudera/parcels/CDH/lib/hadoop") // set
environment variables
> Sys.setenv(SPARK_HOME = "/opt/cloudera/parcels/CDH/lib/spark/")
> sc <- spark_connect(master = "yarn", config = config) // Create a spark
 Connection.

> intDf <- sparklyr::spark_read_table(sc, 'emp_hwc') // read first table
> intDf1 <-  sparklyr::spark_read_table(sc, 'emp_overwrite') // read sec
ond table
> commitTxn(hs) // Commit transaction if read using spark-acid
> SparklyrHWC::spark_write_table('emp_hwc',intDf1,'append') // Append the
 second table, to the first.
> SparklyrHWC::spark_write_table('emp_hwc',intDf1,'overwrite') // Overwri
te the first table with the second table.
```

You do not need to call commitTxn(hs) when using JDBC mode.

Supported HWC APIs in R

About this task

The SparklyrHWC package supports the following HWC APIs in Scala.

- hive.sql() (recommended), execute(), and executeQuery() APIs to run read SQL statements

  Note: It is recommended that you use the hive.sql() method because execute() and executeQuery() methods will soon be deprecated.

- executeUpdate() API to run write SQL statements
- API call for Hive CTASK operations (create table as select .....)
- Other HWC APIs, such as dropTable, dropDatabase, showTable

Any HWC API in Scala can be used in R. The behavior of these APIs in R and HWC is similar.

Procedure

1. Create a table using another existing table.

```
SparklyrHWC::executeUpdate(hs,"create table hwc1 as select * from hwc")
```

2. Read the new table.

```
hwcDf <- SparklyrHWC::sql(hs, "select * from hwc1")
```

3. Convert the table to a SparkDataframe.

```
hwcSdf <- sparklyr::sdf_copy_to(sc, hwcDf)
```

4. Write the table to a new table.

```
SparklyrHWC::spark_write_table('hwcNew',hwcSdf,'OVERWRITE')
```

5. Append the table data to the new table.

```
SparklyrHWC::spark_write_table('hwcNew',hwcSdf,'APPEND')
```

6. Insert data into the table using HWC API.

```
SparklyrHWC::executeUpdate(hs,"insert into hwcNew values(2,'B')")
```

7. Read the newly created table using HWC API.

```
SparklyrHWC::sql(hs, "select * from hwcNew")
```

Livy interpreter configuration

You can use the Hive Warehouse Connector in Zeppelin notebooks with the Livy interpreter by modifying or adding properties to your livy interpreter settings. The Livy interpreter accesses processing engines and data sources from the Zeppelin UI.

Requirements

- Configurations require a livy prefilx.
- A reference to the HWC jar on the local file system is required.

Interpreter properties

- livy.spark.hadoop.hive.metastore.uris — thrift://<domain name>:<port>

  Example: thrift://hwc-secure-1.hwc-secure.root.hwx.site:9083
- livy.spark.jars — local:/opt/cloudera/parcels/<version>/jars/hive-warehouse-connector-assembly-<version>.jar

  Example: local:/opt/cloudera/parcels/CDH-7.2.1-1.cdh7.2.1.p0.4847773/jars/hive-warehouse-connector-assembly-<version>.jar

  Use the local file ("local:").
- livy.spark.kryo.registrator — com.qubole.spark.hiveacid.util.HiveAcidKyroRegistrator
- livy.spark.security.credentials.hiveserver2.enabled — true
- livy.spark.sql.extensions — com.hortonworks.spark.sql.rule.Extensions
- livy.spark.sql.hive.hiveserver2.jdbc.url — jdbc:hive2://<domain name>:<port>/default;retries=5;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2

  Example: jdbc:hive2://hwc-secure-1.hwc-secure.root.hwx.site:2181/default;retries=5;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2
- livy.spark.sql.hive.hiveserver2.jdbc.url.principal — hive/_HOST@ROOT.HWX.SITE

Reading and writing Hive tables in Zeppelin

You can read and write Hive ACID tables from a Spark application using Zeppelin, a browser-based GUI for interactive data exploration, modeling, and visualization.

Before you begin

You must be running spark application and have all the appropriate permissions to read the data from the hive warehouse directory for managed (ACID) tables.

Procedure

1. Open the livy configuration file on the livy node and configure HWC Spark Direct Reader mode.

```
%livy2.conf
...
livy.spark.hadoop.hive.metastore.uris thrift://hwc-secure-1.hwc-secure.r
oot.hwx.site:9083
livy.spark.jars local:/opt/cloudera/parcels/CDH-7.2.1-1.cdh7.2.1.p0.484777
3/jars/hive-warehouse-connector-assembly-<version>.jar
livy.spark.kryo.registrator com.qubole.spark.hiveacid.util.HiveAcidKyroRe
gistrator
livy.spark.security.credentials.hiveserver2.enabled true
livy.spark.sql.extensions com.hortonworks.spark.sql.rule.Extensions
```

```
livy.spark.sql.hive.hiveserver2.jdbc.url jdbc:hive2://hwc-secure-1.hwc-se
cure.root.hwx.site:2181/default;retries=5;serviceDiscoveryMode=zooKeeper
;zooKeeperNamespace=hiveserver2
livy.spark.sql.hive.hiveserver2.jdbc.url.principal hive/_HOST@ROOT.HWX.S
ITE
...
```

2. In a Zeppelin notebook, read a Hive ACID table.

```
sql("show tables").show
sql("select * from hwc2").show
```

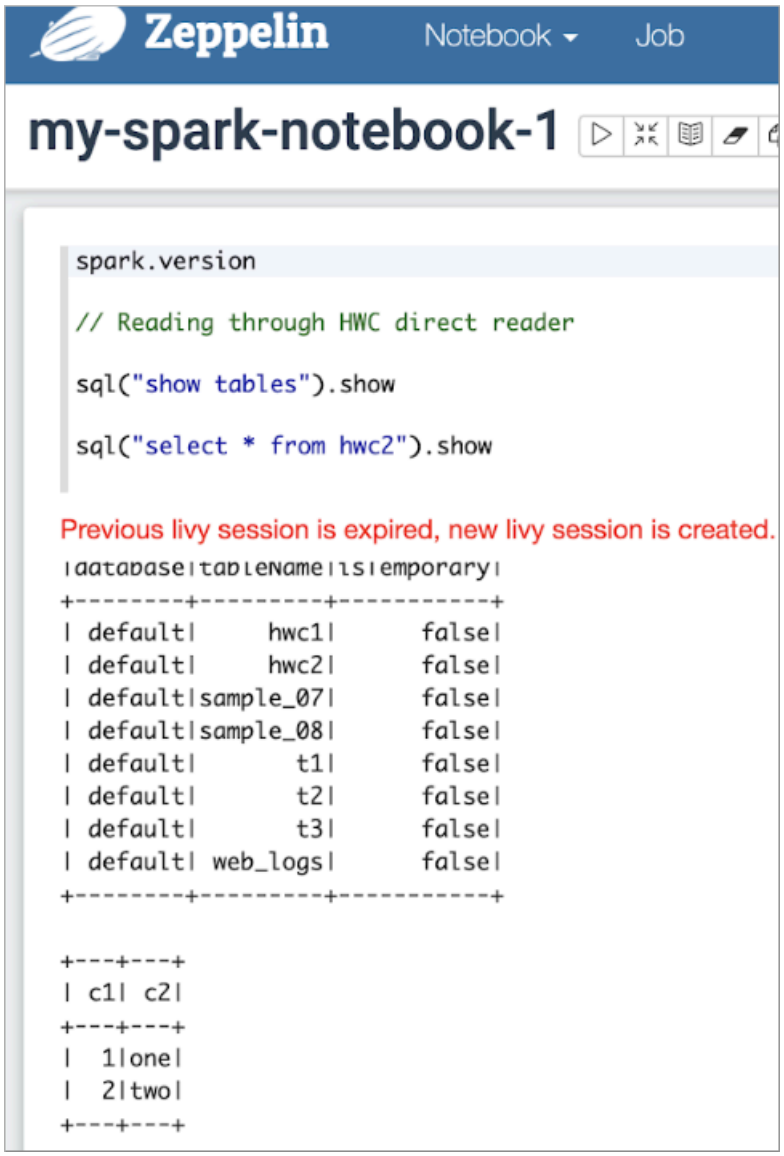

3. Perform a batch write from Spark.

```
val hive = com.hortonworks.hwc.HiveWarehouseSession.session(spark).build()
val df = Seq(
        (1, "a", 1.1),
        (2, "b", 2.2),
        (3, "c", 3.3)
    ).toDF("col1", "col2", "col3")
val tableName = "t3"
hive.dropTable(tableName, true, true)
```

```
df.write.format("com.hortonworks.spark.sql.hive.llap.HiveWarehouseConnect
or").mode("append").option("table", tableName).save()
```

4. Read data from table t3.

```
sql("select * from t3").show
```

Output is:

```
tableName: String = t3
+----+----+----+
|col1|col2|col3|
+----+----+----+
|   1|   a| 1.1|
|   2|   b| 2.2|
|   3|   c| 3.3|
+----+----+----+
```

Apache Hive-Kafka integration

As an Apache Hive user, you can connect to, analyze, and transform data in Apache Kafka from Hive. You can offload data from Kafka to the Hive warehouse. Using Hive-Kafka integration, you can perform actions on real-time data and incorporate streamed data into your application.

You connect to Kafka data from Hive by creating an external table that maps to a Kafka topic. The table definition includes a reference to a Kafka storage handler that connects to Kafka. On the external table, Hive-Kafka integration supports ad hoc queries, such as questions about data changes in the stream over a period of time. You can transform Kafka data in the following ways:

- Perform data masking
- Join dimension tables or any stream
- Aggregate data
- Change the SerDe encoding of the original stream
- Create a persistent stream in a Kafka topic

You can achieve data offloading by controlling its position in the stream. The Hive-Kafka connector supports the following serialization and deserialization formats:

- JsonSerDe (default)
- OpenCSVSerde
- AvroSerDe

Related Information

Apache Kafka Documentation

Creating a table for a Kafka stream

You can create an external table in Apache Hive that represents an Apache Kafka stream to query real-time data in Kafka. You use a storage handler and table properties that map the Hive database to a Kafka topic and broker. If the Kafka data is not in JSON format, you alter the table to specify a serializer-deserializer for another format.

Procedure

1. Get the name of the Kafka topic you want to query to use as a table property.
   For example: "kafka.topic" = "wiki-hive-topic"

2. Construct the Kafka broker connection string.
   For example: "kafka.bootstrap.servers"="kafka.hostname.com:9092"
3. Create an external table named kafka_table by using 'org.apache.hadoop.hive.kafka.KafkaStorageHandler', as shown in the following example:

```
CREATE EXTERNAL TABLE kafka_table
  (`timestamp` timestamp , `page` string,  `newPage` boolean,
  added int, deleted bigint, delta double)
  STORED BY 'org.apache.hadoop.hive.kafka.KafkaStorageHandler'
  TBLPROPERTIES
  ("kafka.topic" = "test-topic", "kafka.bootstrap.servers"="node2:9092");
```

4. If the default JSON serializer-deserializer is incompatible with your data, choose another format in one of the following ways:
   - Alter the table to use another supported serializer-deserializer. For example, if your data is in Avro format, use the Kafka serializer-deserializer for Avro:

     ```
     ALTER TABLE kafka_table SET TBLPROPERTIES ("kafka.serde.class"="org.apac
     he.hadoop.hive.serde2.avro.AvroSerDe");
     ```

   - Create an external table that specifies the table in another format. For example, create a table named that specifies the Avro format in the table definition:

     ```
     CREATE EXTERNAL TABLE kafka_t_avro
        (`timestamp` timestamp , `page` string,  `newPage` boolean,
        added int, deleted bigint, delta double)
        STORED BY 'org.apache.hadoop.hive.kafka.KafkaStorageHandler'
        TBLPROPERTIES
        ("kafka.topic" = "test-topic",
        "kafka.bootstrap.servers"="node2:9092"
        -- STORE AS AVRO IN KAFKA
        "kafka.serde.class"="org.apache.hadoop.hive.serde2.avro.AvroSerDe");
     ```

Related Information

Apache Kafka Documentation

Querying Kafka data

You can get useful information, including Kafka record metadata from a table of Kafka data by using typical Hive queries.

Each Kafka record consists of a user payload key (byte []) and value (byte[]), plus the following metadata fields:

- Partition int32
- Offset int64
- Timestamp int64

The Hive row represents the dual composition of Kafka data:

- The user payload serialized in the value byte array
- The metadata: key byte array, partition, offset, and timestamp fields

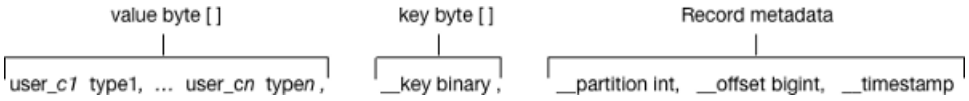

In the Hive representation of the Kafka record, the key byte array is called __key and is of type binary. You can cast __key at query time. Hive appends __key to the last column derived from value byte array, and appends the partition, offset, and timestamp to __key columns that are named accordingly.

Related Information

Apache Kafka Documentation

Querying live data from Kafka

You can get useful information from a table of Kafka data by running typical queries, such as counting the number of records streamed within an interval of time or defining a view of streamed data over a period of time.

Before you begin

This task requires Kafka 0.11 or later to support time-based lookups and prevent full stream scans.

About this task

This task assumes you created a table named kafka_table for a Kafka stream.

Procedure

1. List the table properties and all the partition or offset information for the topic.
   DESCRIBE EXTENDED kafka_table;
2. Count the number of Kafka records that have timestamps within the past 10 minutes.

```
SELECT COUNT(*) FROM kafka_table
  WHERE `__timestamp` >  1000 * to_unix_timestamp(CURRENT_TIMESTAMP - inte
rval '10' MINUTES);
```

   Such a time-based seek requires Kafka 0.11 or later, which has a Kafka broker that supports time-based lookups; otherwise, this query leads to a full stream scan.
3. Define a view of data consumed within the past 15 minutes and mask specific columns.

```
CREATE VIEW last_15_minutes_of_kafka_table AS SELECT  `timestamp`, `user`,
 delta,
  ADDED FROM kafka_table
  WHERE `__timestamp` >  1000 * to_unix_timestamp(CURRENT_TIMESTAMP - i
nterval '15' MINUTES) ;
```

4. Create a dimension table.

```
CREATE TABLE user_table (`user` string, `first_name` string , age int, g
ender string, comments string) STORED as ORC ;
```

5. Join the view of the stream over the past 15 minutes to user_table, group by gender, and compute aggregates over metrics from fact table and dimension tables.

```
SELECT SUM(added) AS added, SUM(deleted) AS deleted, AVG(delta) AS delta,
 AVG(age) AS avg_age , gender
  FROM last_15_minutes_of_kafka_table
  JOIN user_table ON `last_15_minutes_of_kafka_table`.`user` = `user_tab
le`.`user`
  GROUP BY gender LIMIT 10;
```

6. Perform a classical user retention analysis over the Kafka stream consisting of a stream-to-stream join that runs adhoc queries on a view defined over the past 15 minutes.

```
-- Stream join over the view itself
-- Assuming l15min_wiki is a view of the last 15 minutes
SELECT  COUNT( DISTINCT activity.`user`) AS active_users,
COUNT(DISTINCT future_activity.`user`) AS retained_users
FROM l15min_wiki AS activity
LEFT JOIN l15min_wiki AS future_activity ON activity.`user` = future_activ
ity.`user`
```

```
AND activity.`timestamp` = future_activity.`timestamp` - interval '5' min
utes ;

--  Stream-to-stream join
-- Assuming wiki_kafka_hive is the entire stream.
SELECT floor_hour(activity.`timestamp`), COUNT( DISTINCT activity.`user`)
 AS active_users,
COUNT(DISTINCT future_activity.`user`) as retained_users
FROM wiki_kafka_hive AS activity
LEFT JOIN wiki_kafka_hive AS future_activity ON activity.`user` = future_
activity.`user`
AND activity.`timestamp` = future_activity.`timestamp` - interval '1' ho
ur
GROUP BY floor_hour(activity.`timestamp`);
```

Related Information

Apache Kafka Documentation

Perform ETL by ingesting data from Kafka into Hive

You can extract, transform, and load a Kafka record into Hive in a single transaction.

Procedure

1. Create a table to represent source Kafka record offsets.

```
CREATE TABLE kafka_table_offsets(partition_id int, max_offset bigint, in
sert_time timestamp);
```

2. Initialize the table.

```
INSERT OVERWRITE TABLE kafka_table_offsets
SELECT `__partition`, min(`__offset`) - 1, CURRENT_TIMESTAMP
FROM wiki_kafka_hive
GROUP BY `__partition`, CURRENT_TIMESTAMP;
```

3. Create the destination table.

```
CREATE TABLE orc_kafka_table (partition_id int, koffset bigint, ktimestamp
 bigint,
  `timestamp` timestamp , `page` string, `user` string, `diffurl` string,
  `isrobot` boolean, added int, deleted int, delta bigint
) STORED AS ORC;
```

4. Insert Kafka data into the ORC table.

```
FROM wiki_kafka_hive ktable JOIN kafka_table_offsets offset_table
ON (ktable.`__partition` = offset_table.partition_id
AND ktable.`__offset` > offset_table.max_offset )
INSERT INTO TABLE orc_kafka_table
SELECT `__partition`, `__offset`, `__timestamp`,
  `timestamp`, `page`, `user`, `diffurl`, `isrobot`, added , deleted , del
ta
INSERT OVERWRITE TABLE kafka_table_offsets
SELECT `__partition`, max(`__offset`), CURRENT_TIMESTAMP
GROUP BY `__partition`, CURRENT_TIMESTAMP;
```

5. Check the insertion.

```
SELECT MAX(`koffset`) FROM orc_kafka_table LIMIT 10;
```

```
SELECT COUNT(*) AS c FROM orc_kafka_table
GROUP BY partition_id, koffset HAVING c > 1;
```

6. Repeat step 4 periodically until all the data is loaded into Hive.

Writing data to Kafka

You can extract, transform, and load a Hive table to a Kafka topic for real-time streaming of a large volume of Hive data. You need some understanding of write semantics and the metadata columns required for writing data to Kafka.

Write semantics

The Hive-Kafka connector supports the following write semantics:

- At least once (default)
- Exactly once

At least once (default)

The default semantic. At least once is the most common write semantic used by streaming engines. The internal Kafka producer retries on errors. If a message is not delivered, the exception is raised to the task level, which causes a restart, and more retries. The At least once semantic leads to one of the following conclusions:

- If the job succeeds, each record is guaranteed to be delivered at least once.
- If the job fails, some of the records might be lost and some might not be sent.

  In this case, you can retry the query, which eventually leads to the delivery of each record at least once.

Exactly once

Following the exactly once semantic, the Hive job ensures that either every record is delivered exactly once, or nothing is delivered. You can use only Kafka brokers supporting the Transaction API (0.11.0.x or later). To use this semantic, you must set the table property "kafka.write.semantic"="EXACTLY_ONCE".

Metadata columns

In addition to the user row payload, the insert statement must include values for the following extra columns:

__key

Although you can set the value of this metadata column to null, using a meaningful key value to avoid unbalanced partitions is recommended. Any binary value is valid.

__partition

Use null unless you want to route the record to a particular partition. Using a nonexistent partition value results in an error.

__offset

You cannot set this value, which is fixed at -1.

__timestamp

You can set this value to a meaningful timestamp, represented as the number of milliseconds since epoch. Optionally, you can set this value to null or -1, which means that the Kafka broker strategy sets the timestamp column.

Related Information

Apache Kafka Documentation

Writing transformed Hive data to Kafka

You can change streaming data and include the changes in a stream. You extract a Kafka input topic, transform the record in Hive, and load a Hive table back into a Kafka record.

About this task

This task assumes that you already queried live data from Kafka. When you transform the record in the Hive execution engine, you compute a moving average over a window of one minute. The resulting record that you write back to another Kafka topic is named moving_avg_wiki_kafka_hive.

.

Procedure

1. Create an external table to represent the Hive data that you want to load into Kafka.

```
CREATE EXTERNAL TABLE moving_avg_wiki_kafka_hive
(`channel` string, `namespace` string,`page` string, `timestamp` timestamp
 , avg_delta double )
STORED BY 'org.apache.hadoop.hive.kafka.KafkaStorageHandler'
TBLPROPERTIES
  ("kafka.topic" = "moving_avg_wiki_kafka_hive",
  "kafka.bootstrap.servers"="kafka.hostname.com:9092",
  -- STORE AS AVRO IN KAFKA
  "kafka.serde.class"="org.apache.hadoop.hive.serde2.avro.AvroSerDe");
```

2. Insert data that you select from the Kafka topic back into the Kafka record.

```
INSERT INTO TABLE moving_avg_wiki_kafka_hive
SELECT `channel`, `namespace`, `page`, `timestamp`,
  AVG(delta) OVER (ORDER BY `timestamp` ASC ROWS BETWEEN  60 PRECEDING AND
 CURRENT ROW) AS avg_delta,
  null AS `__key`, null AS `__partition`, -1 AS `__offset`, to_epoch_milli
(CURRENT_TIMESTAMP) AS `__timestamp`
FROM l15min_wiki;
```

The timestamps of the selected data are converted to milliseconds since epoch for clarity.

Related Information

Querying live data from Kafka

Setting consumer and producer table properties

You can use Kafka consumer and producer properties in the TBLPROPERTIES clause of a Hive query. By prefixing the key with kafka.consumer or kafka.producer, you can set the table properties.

Procedure

For example, if you want to inject 5000 poll records into the Kafka consumer, use the following syntax.

```
ALTER TABLE kafka_table SET TBLPROPERTIES ("kafka.consumer.max.poll.records"
="5000");
```

Kafka storage handler and table properties

You use the Kafka storage handler and table properties to specify the query connection and configuration.

Kafka storage handler

You specify 'org.apache.hadoop.hive.kafka.KafkaStorageHandler' in queries to connect to, and transform a Kafka topic into, a Hive table. In the definition of an external table, the storage handler creates a view over a single Kafka topic. For example, to use the storage handler to connect to a topic, the following table definition specifies the storage handler and required table properties: the topic name and broker connection string.

```
CREATE EXTERNAL TABLE kafka_table
   (`timestamp` timestamp , `page` string,  `newPage` boolean,
   added int, deleted bigint, delta double)
   STORED BY 'org.apache.hadoop.hive.kafka.KafkaStorageHandler'
   TBLPROPERTIES
   ("kafka.topic" = "test-topic", "kafka.bootstrap.servers"="localhost:90
92");
```

You set the following table properties forwith the Kafka storage handler:

kafka.topic

The Kafka topic to connect to

kafka.bootstrap.servers

The broker connection string

Storage handler-based optimizations

The storage handler can optimize reads using a filter push-down when you run a query such as the following time-based lookup supported on Kafka 0.11 or later:

```
SELECT COUNT(*) FROM kafka_table
   WHERE `__timestamp` >  1000 * to_unix_timestamp(CURRENT_TIMESTAMP - int
erval '10' MINUTES) ;
```

The Kafka consumer supports seeking on the stream based on an offset, which the storage handler leverages to push down filters over metadata columns. The storage handler in the example above performs seeks based on the Kafka record __timestamp to read only recently arrived data.

The following logical operators and predicate operators are supported in the WHERE clause:

Logical operators: OR, AND

Predicate operators: <, <=, >=, >, =

The storage handler reader optimizes seeks by performing partition pruning to go directly to a particular partition offset used in the WHERE clause:

```
SELECT COUNT(*)  FROM kafka_table
  WHERE (`__offset` < 10 AND `__offset` > 3 AND `__partition` = 0)
  OR (`__partition` = 0 AND `__offset` < 105 AND `__offset` > 99)
  OR (`__offset` = 109);
```

The storage handler scans partition 0 only, and then read only records between offset 4 and 109.

Kafka metadata

In addition to the user-defined payload schema, the Kafka storage handler appends to the table some additional columns, which you can use to query the Kafka metadata fields:

__key

Kafka record key (byte array)

__partition

Kafka record partition identifier (int 32)

__offset

Kafka record offset (int 64)

__timestamp

Kafka record timestamp (int 64)

The partition identifier, record offset, and record timestamp plus a key-value pair constitute a Kafka record. Because the key-value is a 2-byte array, you must use SerDe classes to transform the array into a set of columns.

Table Properties

You use certain properties in the TBLPROPERTIES clause of a Hive query that specifies the Kafka storage handler.

Property	Description	Required	Default
kafka.topic	Kafka topic name to map the table to	Yes	null
kafka.bootstrap.servers	Table property indicating the Kafka broker connection string	Yes	null
kafka.serde.class	Serializer and Deserializer class implementation	No	org.apache.hadoop.hive.serde2.JsonSerDe
hive.kafka.poll.timeout.ms	Parameter indicating Kafka Consumer poll timeout period in milliseconds. (This is independent of internal Kafka consumer timeouts.)	No	5000 (5 Seconds)
hive.kafka.max.retries	Number of retries for Kafka metadata fetch operations	No	6
hive.kafka.metadata.poll.timeout.ms	Number of milliseconds before consumer timeout on fetching Kafka metadata	No	30000 (30 Seconds)
kafka.write.semantic	Writer semantic with allowed values of NONE, AT_LEAST_ONCE, EXACTLY_ONCE	No	AT_LEAST_ONCE

Cloudera Runtime 7.3.1

Integrating with Schema Registry

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Integrating Schema Registry with NiFi....4
NiFi record-based Processors and Controller Services....4
Configuring Schema Registry instance in NiFi....4
Setting schema access strategy in NiFi....5
Adding and configuring record-enabled Processors....5

Integrating Schema Registry with Kafka....6

Integrating Schema Registry with Flink and Cloudera SQL Stream Builder....7

Integrating Schema Registry with Atlas....8

Improving performance in Schema Registry....11

Integrating Schema Registry with NiFi

When using NiFi to convert events from one file format to another, you need to integrate Schema Registry with NiFi because NiFi fetches the schema for the event from Schema Registry to be able to perform the conversion.

NiFi record-based Processors and Controller Services

Learn about the NiFi Processors and Controller Services that facilitate converting events from one file format to another.

The RecordReader and RecordWriter Processors and Controller Services allow you to convert events from one file format (JSON, XML, CSV, Avro) to another (JSON, XML, CSV, Avro). These Controller Services use Schema Registry to fetch the schema for the event to perform this conversion.

NiFi includes the following RecordReader and RecordWriter Processors:

- ConsumeKafkaRecord_0_10 1.2.0
- ConvertRecord
- PublishKafkaRecord_0_10
- PutDatabaseRecord
- QueryRecord
- SplitRecord

NiFi also includes the following record-based Controller Services:

- HortonworksSchemaRegistry
- AvroRecordSetWriter
- CSVRecordSetWriter
- FreeFormTextRecordSetWriter
- JsonRecordSetWriter
- ScriptedRecordSetWriter

Configuring Schema Registry instance in NiFi

Configuring Schema Registry in NiFi is essential for seamless communication between the two components. To enable Schema Registry to communicate with NiFi dataflows, you need to configure the HortonworksSchemaRegistry Controller Service with the Schema Registry instance URL.

Before you begin

You have already installed Schema Registry.

Procedure

1. Navigate to Controller Settings in the Global Menu and select the Controller Services tab.
2. Click the + icon to access the Add Controller Service dialog.
3. Search for HortonworksSchemaRegistry in the filter box and click Add.
4. Open the Configure Controller Service dialog by clicking the Edit icon.
5. Provide the Schema Registry URL with which you want NiFi to communicate and click Apply.
6. Enable HortonworksSchemaRegistry by clicking the Enable icon, selecting the Scope, and then clicking Enable again.

Setting schema access strategy in NiFi

Learn how to add Controller Services in NiFi globally or per Process Group and then how to configure them so that they can fetch schemas from Schema Registry.

You can configure Controller Services either globally, before you have created a Process Group, or at any time, on a per Process Group basis.

Adding Controller Services globally

1. To access the Controller Services configuration dialog for gobal configuration, click the Global Menu at the top right of your canvas, and select Controller Settings.
2. Click the + icon to display the NiFi Settings dialog.
3. Use the Filter box to search for the Controller Service you want to add, select that service, and click Add.

Adding Controller Services per Process Group

1. Click on your Process Group, and then right-click anywhere on your canvas.
2. Click Configure to display the Process Group Configuration dialog.
3. Click the Controller Services tab, and then click + to display the Add Controller Service dialog.
4. Use the Filter box to search for the Controller Service you want to add, select that service, and click Add.

Configuring record reader and writer Controller Services for integration with Schema Registry

1. From the Process Group Configuration view, click the Edit icon from the right-hand column. The Configure Controller Service dialog appears.
2. Click the Properties tab.
3. The *SCHEMA ACCESS STRATEGY* specifies how to obtain the schema to be used for interpreting FlowFile data. To ensure integration with Schema Registry, configure *SCHEMA ACCESS STRATEGY* with one of the following two values:
   - HWX Schema Reference Attributes

     The NiFi FlowFile is given a set of 3 attributes to describe the schema:

     - schema.identifier
     - schema.version
     - schema.protocol.version
   - HWX Content-Encoded Schema Reference

     Each NiFi FlowFile contains a reference to a schema stored in Schema Registry. The reference is encoded as a single byte indicating the protocol version, 8 bytes indicating the schema identifier and 4 bytes indicating the schema version.
4. The *SCHEMA WRITE STRATEGY* specifies how the schema for a record should be added to FlowFile data. To ensure integration with Schema Registry, configure *SCHEMA WRITE STRATEGY* with either *HWX SCHEMA REFERENCE ATTRIBUTES* or *HWX CONTENT-ENCODED SCHEMA REFERENCE*.

Adding and configuring record-enabled Processors

To streamline your dataflows and improve overall performance, you can use record-enabled RecordReader and RecordWriter Processors.

About this task

Record-enabled Processors allow you to convert data between formats by specifying Controller Services for record reading and record writing. To add and configure record-enabled Processors, perform the following tasks:

Procedure

1. From the NiFi UI, drag the Processor icon onto your canvas to display the Add Processor dialog.
2. Use the Filter box to find the processor you want to add. Available record-enabled processors are:
   - ConsumeKafkaRecord_0_10
   - ConvertRecord
   - PublishKafkaRecord_0_10
   - PutDatabaseRecord
   - QueryRecord
   - SplitRecord
3. Select the processor you want, and click Add.
4. Right-click the processor on the canvas, and select Configure to display the Configure Processor dialog.
5. Click the Properties tab and select a Controller Service value for the Record Reader and Record Writer values.
6. Click OK, and then Apply.

Integrating Schema Registry with Kafka

Learn the manual configuration steps required to integrate Kafka and Schema Registry if you are running Cloudera Public Cloud without NiFi.

If you are running Cloudera Public Cloud without NiFi, integrate your Kafka producer and consumer manually. To do this, add a dependency on the Schema Registry SerDes and update the Kafka producer and Kafka consumer configuration files.

Adding a Schema Registry SerDes dependency

Add the following text to schema-registry-serdes:

```
<dependency>
      <groupId>com.hortonworks.registries</groupId>
      <artifactId>schema-registry-serdes</artifactId>
</dependency>
```

Integrating the Kafka producer

1. Add the following text to the Kafka producer configuration:

   ```
   config.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServers);
   config.putAll(Collections.singletonMap(SchemaRegistryClient.Configuration
   .SCHEMA_REGISTRY_URL.name(), props.get(SCHEMA_REGISTRY_URL)));
   config.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.c
   lass.getName());
   config.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, KafkaAvroSeria
   lizer.class.getName());
   ```

2. Edit the above text with values for the following properties:
   - schema.registry.url
   - key.serializer
   - value.serializer

Integrating the Kafka consumer

1. Add the following text to the Kafka consumer configuration:

```
config.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServers);
config.putAll(Collections.singletonMap(SchemaRegistryClient.Configuration
.SCHEMA_REGISTRY_URL.name(), props.get(SCHEMA_REGISTRY_URL)));
config.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializ
er.class.getName());
config.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, KafkaAvroDes
erializer.class.getName());
```

2. Edit the above text with values for the following properties:
   - schema.registry.url
   - key.deserializer
   - value.deserializer

Customizing client cache duration

By default, Schema Registry clients are only able to cache the schema for 5 minutes. To customize the cache duration, configure the cache properties.

Customize the values for the following properties:

- Schema Metadata Cache

  ```
  schema.registry.client.schema.metadata.cache.expiry.interval.secs
  ```

  Expiry interval (in seconds) of an entry in schema metadata cache. Default value is 300.
- Schema Version Cache

  ```
  schema.registry.client.schema.version.cache.expiry.interval.secs
  ```

  Expiry interval (in seconds) of an entry in schema version cache. Default value is 300.
- Schema Text Cache

  ```
  schema.registry.client.schema.text.cache.expiry.interval.secs
  ```

  Expiry interval (in seconds) of an entry in schema text cache. Default value is 300.
- Class Loader Cache

  ```
  schema.registry.client.class.loader.cache.expiry.interval.secs
  ```

  Expiry interval (in seconds) of an entry in the serializer/deserializer class loader cache. Default value is 300.

Integrating Schema Registry with Flink and Cloudera SQL Stream Builder

When using Kafka with Schema Registry, you can integrate them with Flink for large-scale data streams processing, and with Cloudera SQL Stream Builder to import schema as virtual tables and use the tables to create queries on streams of data through SQL.

When Kafka is chosen as source and sink for your application, you can use Schema Registry to register and retrieve schema information of the different Kafka topics. You must add Schema Registry dependency to your Flink project and add the appropriate schema object to your Kafka topics. Additionally, you can add Schema Registry as a catalog to Cloudera SQL Stream Builder to import existing schemas for use with SQL queries.

You can integrate Flink with your project along with Kafka and Schema Registry to process data streams at a large scale and to deliver real-time analytical insights about your processed data. For more information about how to integrate Flink with Schema Registry, see *Schema Registry with Flink*.

You can integrate Schema Registry as a catalog with Cloudera SQL Stream Builder to import schema as virtual tables which you can use to create queries on the streams of data through SQL. For more information about how to integrate Cloudera SQL Stream Builder with Schema Registry, see *Adding Catalogs*.

Related Information

Schema Registry with Flink

Adding Catalogs

Integrating Schema Registry with Atlas

If you want to view schemas in the Atlas UI, you need to integrate Schema Registry with Apache Atlas.

When you integrate Schema Registry with Atlas, the relationship between the schema, topic, and all versions of a schema can be explored with ease in Atlas.

The following image shows a sample relationship:

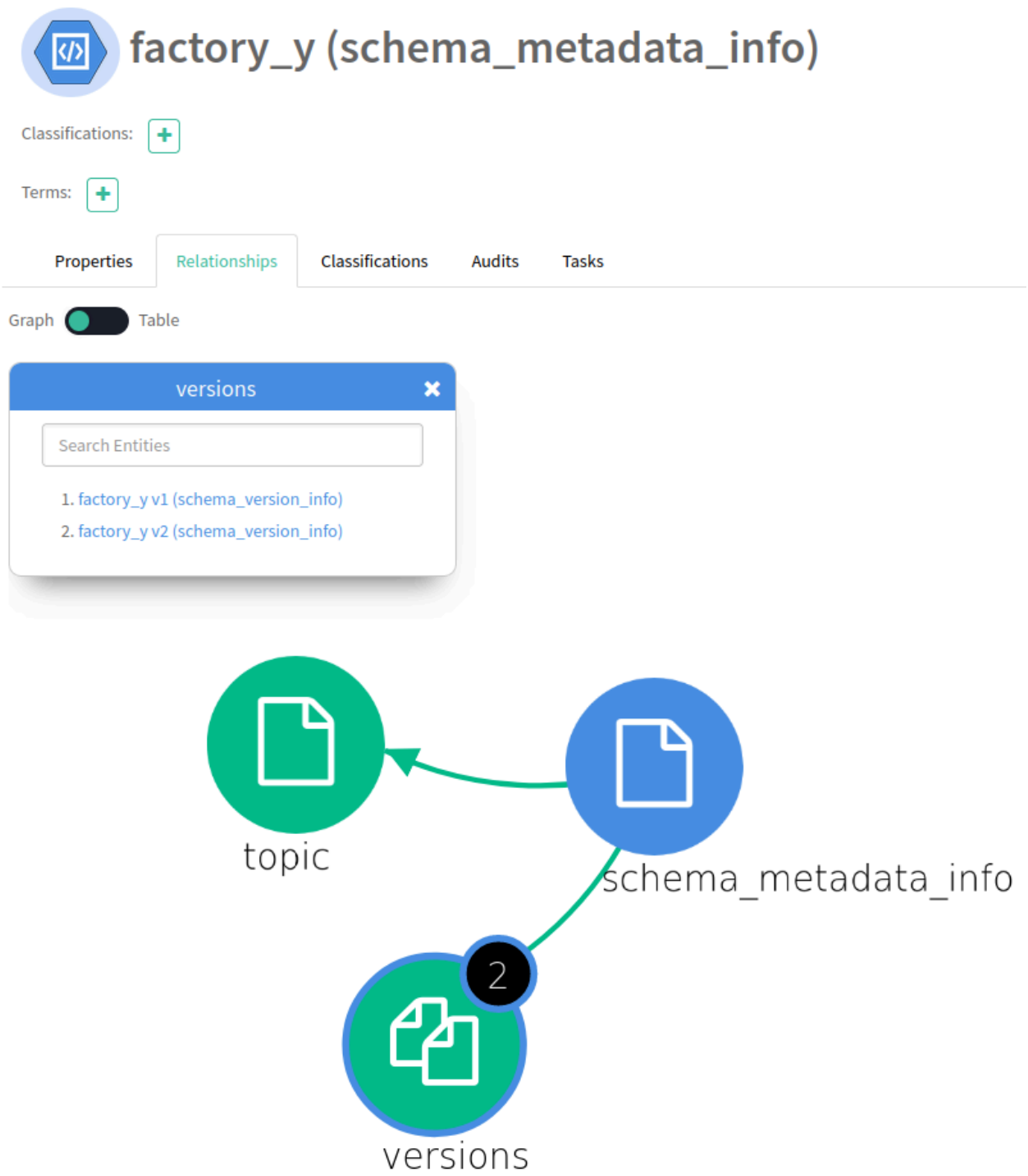

You can select a schema version to view the schema text in JSON.

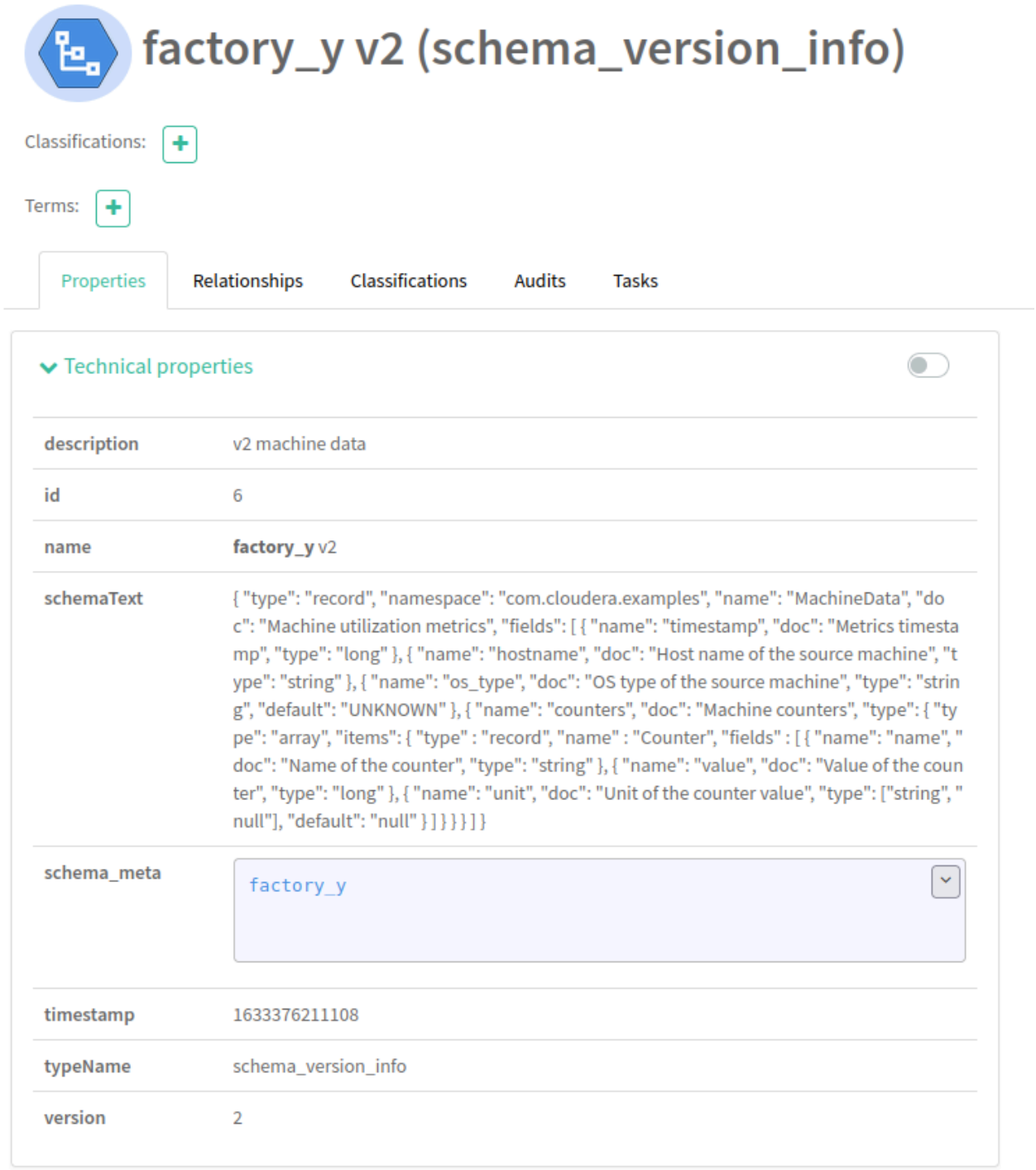

For more information, see *Schema Registry metadata collection*.

Related Information

Schema Registry metadata collection

Improving performance in Schema Registry

Adding Knox as a load balancer can increase the performance of Schema Registry.

You can increase the performance of Schema Registry by adding Apache Knox, the Cloudera load balancer, to your Schema Registry configuration. This increase in performance offsets the minor decrease in performance caused by the lack of server-side caching.

By default, Schema Registry does not have server-side caching. Each time a client sends a query, the client reads the data directly from the database.

The lack of server-side caching creates a minor performance impact. However, it enables Schema Registry to be stateless and be distributed over multiple hosts. All hosts use the same database. This enables a client to insert a schema on host A and query the same schema on host B.

You can store JAR files similarly in HDFS and retrieve a file even if one of the hosts is down.

To make Schema Registry highly available in this setup, include a load balancer on top of the services. Clients communicate through the load balancer. The load balancer then determines which host to send the requests to. If one host is down, the request is sent to another host. In CDP, the load balancer is Apache Knox.

The following diagram shows the flow of a client query and response:

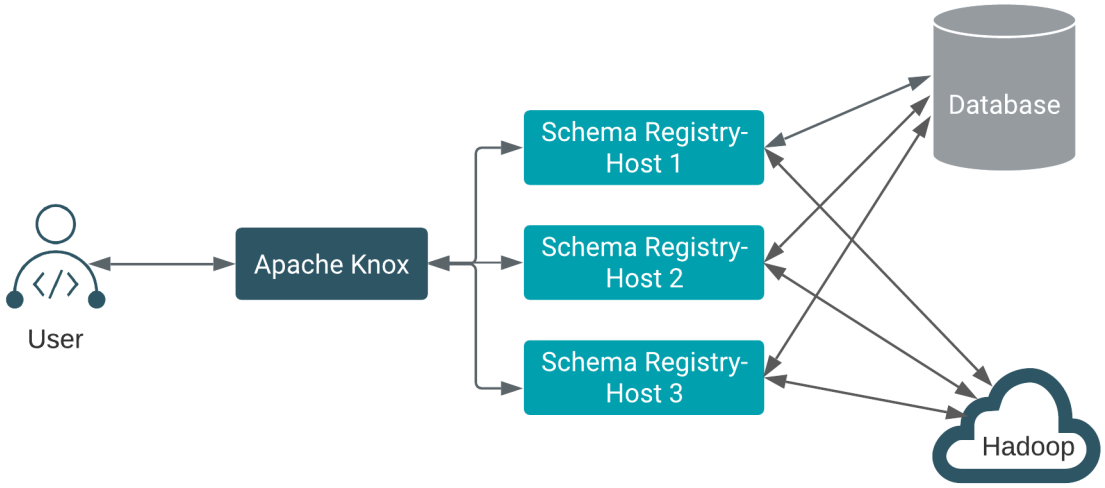

Cloudera Runtime 7.3.1

Configuring Apache Kafka

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Operating system requirements 4

Performance considerations 4

Quotas 5
- Configuring quotas 6

JBOD 7
- JBOD setup 8
- JBOD Disk migration 9

Setting user limits for Kafka 11

Connecting Kafka clients to Cloudera Data Hub provisioned clusters 12

Rolling restart checks 15
- Configuring rolling restart checks 17
- Configuring the client configuration used for rolling restart checks 18

Cluster discovery with multiple Apache Kafka clusters 19
- Cluster discovery using DNS records 19
  - A records and round robin DNS 20
  - client.dns.lookup property options for client 20
  - CNAME records configuration 21
  - Connection to the cluster with configured DNS aliases 22
- Cluster discovery using load balancers 22
  - Setup for SASL with Kerberos 24
  - Setup for TLS/SSL encryption 26
  - Connecting to the Kafka cluster using load balancer 26

Configuring Kafka ZooKeeper chroot 27

Configure Kafka rack awareness 27
- Configuring rack awareness for Kafka brokers 28
- Configuring multi-level rack awareness for brokers 29
- Configuring rack awareness for Kafka consumers 30
- Configuring multi-level rack awareness for consumers 31
- Configuring rack awareness for Kafka producers 32

Operating system requirements

A collection of operating system requirements for Kafka.

SUSE Linux Enterprise Server (SLES)

Unlike CentOS, SLES limits virtual memory by default. Changing this default requires adding the following entries to the /etc/security/limits.conf file:

```
* hard as unlimited
* soft as unlimited
```

Kernel Limits

There are three settings you must configure properly for the kernel.

File Descriptors

You can set file descriptors in Cloudera Manager by going to KafkaConfigurationMaximum Process File Descriptors and setting the required value. Cloudera recommends a configuration of 100000 or higher.

Important:

After changing the maximum process file descriptor limit, you might find that service roles are still limited by the number of file descriptors. Raising the maximum process file descriptors above the Linux kernel file descriptor limit will have no effect. Check the Linux kernel file descriptor limit on every host in the cluster and raise that if necessary.

You can find the Linux kernel file descriptor limit by running the following command on the Linux command line:

```
sudo cat /proc/sys/fs/nr_open
```

Max Memory Map

You must configure the maximum number of memory maps in your specific kernel settings. Cloudera recommends a configuration of 32000 or higher.

Max Socket Buffer Size

Set the buffer size larger than any Kafka send buffers that you define.

Performance considerations

A collection of basic recommendations for Kafka clusters.

The simplest recommendation for running Kafka with maximum performance is to have dedicated hosts for the Kafka brokers and a dedicated ZooKeeper cluster for the Kafka cluster. If that is not an option, consider these additional guidelines for resource sharing with the Kafka cluster:

Running in VMs

It is common practice in modern data centers to run processes in virtual machines. This generally allows for better sharing of resources. Kafka is sufficiently sensitive to I/O throughput that VMs interfere with the regular operation of brokers. For this reason, it is generally not recommended to run Kafka in VMs. However, if you are running Kafka in a virtual environment you will need to rely on your VM vendor for help with optimizing Kafka performance.

Do not run other processes with Brokers or ZooKeeper

Due to I/O contention with other processes, it is generally recommended to avoid running other such processes on the same hosts as Kafka brokers.

Keep the Kafka-ZooKeeper Connection Stable

Kafka relies heavily on having a stable ZooKeeper connection. Putting an unreliable network between Kafka and ZooKeeper will appear as if ZooKeeper is offline to Kafka. Examples of unreliable networks include:

- Do not put Kafka/ZooKeeper nodes on separated networks
- Do not put Kafka/ZooKeeper nodes on the same network with other high network loads

Quotas

Learn about Apache Kafka quotas, quota types, client groups, quota violation enforcement and detection, quota storage, as well as how you can configure quotas.

Kafka can enforce quotas on produce and fetch requests. Producers and consumers can use very high volumes of data. This can monopolize broker resources, cause network saturation, and generally deny service to other clients and the brokers themselves. Quotas protect against these issues and are important for large, multitenant clusters where a small set of clients using high volumes of data can degrade the user experience.

Quota types

There are two types of client quotas that can be enforced by Kafka brokers for each group of clients sharing a quota. These are as follows.

Network bandwidth quotas (byte-rate thresholds)

Network bandwidth quotas are defined as the byte rate threshold for each group of clients sharing a quota. Each group of clients can publish or fetch a maximum of X bytes/second per broker before clients are throttled.

Request rate quotas (CPU utilization thresholds)

Request rate quotas are defined as the percentage of time a client can utilize on request handler I/O threads and network threads of each broker within a quota window. A quota of n% represents n% of one thread, so the quota is out of a total capacity of ((num.io.threads + num.network.threads) * 100)%.

Each group of clients can use a total percentage of up to n% across all I/O and network threads in a quota window before being throttled. Since the number of threads allocated for I/O and network threads are typically based on the number of cores available on the broker host, request rate quotas represent the total percentage of CPU that each group of clients sharing the quota can use.

Client Groups

The identity of Kafka clients is the user principal, which represents an authenticated user in a secure cluster. In a cluster that supports unauthenticated clients, the user principal is a grouping of unauthenticated users chosen by the broker using a configurable `PrincipalBuilder`.

A client-id logically identifies an application making a request. A single client-id can span multiple producer and consumer instances. The tuple (user + client-id) defines a secure logical group of clients that share both user principal and client-id.

You can apply quotas to tuple (user + client-id), user, or client-id groups. For a given connection, the most specific quota matching the connection is applied. All connections of a quota group share the quota configured for the group. For example, if the tuple (user="test-user", client-id="test-client") has a produce quota of 10 MB/sec, then that quota is shared across all producer instances using the test-user user with the test-client client-id.

Quota violation detection

To detect quota violations quickly, byte-rate and thread utilization are measured over multiple small windows. For example, 30 windows that are each 1 second long. Having large measurement windows, for example, 10 windows that are 30 seconds each, leads to large bursts of traffic followed by long delays.

Quota violation enforcement

When a client exceeds its specified quota, the broker attempts to throttle the client instead of returning an error message. Throttling happens using the following logic.

1. The broker calculates the amount of delay needed to bring a client under its quota.
2. The broker sends a response to the client that includes the delay that the broker calculated. If the broker is responding to a fetch request, the response only contains the delay. The data that was requested is not included in the response.
3. The broker mutes the channel to the client. Any additional requests that the broker receives from the client are only processed after the delay is over.
4. If a client receives a response that includes a delay, the client refrains from sending further requests to the broker. This means that requests form a throttled client are blocked by both broker and client.

Quota storage and precedence

Quota configurations are stored in ZooKeeper. Specifically, user and tuple (user + client-id) quota configurations are written to ZooKeeper under /config/users, and client-id quota configurations are written under /config/clients.

The order of precedence for quota configuration is as follows.

1. /config/users/*[***USER***]*/clients/*[***CLIENT ID***]*
2. /config/users/*[***USER***]*/clients/<default>
3. /config/users/*[***USER***]*
4. /config/users/<default>/clients/*[***CLIENT ID***]*
5. /config/users/<default>/clients/<default>
6. /config/users/<default>
7. /config/clients/*[***CLIENT ID***]*
8. /config/clients/<default>

Configuring quotas

Learn how to configure Apache Kafka quotas.

By default, each client receives an unlimited quota. However, customizing quotas is possible. You can define quota configuration on the level of tuple (user + client-id), user, and client-id groups. In addition, there are two categories of quota configuration, default and custom.

For example, you can define a default quota configuration that applies to all clients, but also specify custom configurations that only apply to a specific subset of clients. Both default and custom configurations can be specified for any of the levels. Additionally, changes to quota configuration are dynamic and are effective immediately. This means that a broker restart is not required for the configuration to take effect.

Configuration is done with the kafka-configs tool using the --alter, --add-config, --entity-type, --entity-name, and --entity-default options.

The following collects various example commands that configure quotas as well as an example demonstrating how you can describe quotas.

Custom quota configuration examples

- Configure a custom quota for a tuple (user + client-id)

```
kafka-configs --bootstrap-server [***HOST***]:[***PORT***] --alter --add-
config 'producer_byte_rate=1024,consumer_byte_rate=2048,request_percentag
e=200' --entity-type users --entity-name [***USER NAME***] --entity-type c
lients --entity-name [***CLIENT-ID***]
```

- Configure a custom quota for a user

```
kafka-configs --bootstrap-server [***HOST***]:[***PORT***] --alter --add-
config 'producer_byte_rate=1024,consumer_byte_rate=2048,request_percentag
e=200' --entity-type users --entity-name [***USER NAME***]
```

- Configure a custom quota for a client-id

```
kafka-configs --bootstrap-server [***HOST***]:[***PORT***] --alter --add-
config 'producer_byte_rate=1024,consumer_byte_rate=2048,request_percentag
e=200' --entity-type clients --entity-name [***CLIENT-ID***]
```

Default quota configuration examples

Notice that default quota configurations are set by specifying the --entity-default option instead of --entity-name.

- Configure a default client-id quota for a user

```
kafka-configs --bootstrap-server [***HOST***]:[***PORT***] --alter --add-
config 'producer_byte_rate=1024,consumer_byte_rate=2048,request_percentag
e=200' --entity-type users --entity-name [***USER NAME***] --entity-type c
lients --entity-default
```

- Configure a default quota for a user

```
kafka-configs --bootstrap-server [***HOST***]:[***PORT***] --alter --add-
config 'producer_byte_rate=1024,consumer_byte_rate=2048,request_percentag
e=200' --entity-type users --entity-default
```

- Configure a default quota for a client-id

```
kafka-configs --bootstrap-server [***HOST***]:[***PORT***] --alter --add-
config 'producer_byte_rate=1024,consumer_byte_rate=2048,request_percentag
e=200' --entity-type clients --entity-default
```

Describing quotas

In addition to configuring quotas using kafka-configs, you can also describe the quota configuration for any specific tuple (user + client-id), user, or client-id group. This is done by using the --describe option. For example, the quota configuration of a specific tuple can be retrieved with the following command.

```
kafka-configs --bootstrap-server [***HOST***]:[***PORT***] --describe  --e
ntity-type users --entity-name [***USER NAME***] --entity-type clients --ent
ity-name [***CLIENT-ID***]
```

JBOD

Overview on Kafka with JBOD.

JBOD refers to a system configuration where disks are used independently rather than organizing them into redundant arrays (RAID). Using RAID usually results in more reliable hard disk configurations even if the individual disks are not reliable. RAID setups like these are common in large scale big data environments built on top of commodity

hardware. RAID enabled configurations are more expensive and more complicated to set up. In a large number of environments, JBOD configurations are preferred for the following reasons:

- Reduced storage cost: RAID-10 is recommended to protect against disk failures. However, scaling RAID-10 configurations can become excessively expensive. Storing the data redundantly on each node means that storage space requirements have to be multiplied because the data is also replicated across nodes.
- Improved performance: Just like HDFS, the slowest disk in RAID-10 configuration limits overall throughput. Writes need to go through a RAID controller. On the other hand, when using JBOD, IO performance is increased as a result of isolated writes across disks without a controller.

JBOD setup

Learn how to set up JBOD in your Kafka environment.

Before you begin

Consider the following before using JBOD support in Kafka:

- Manual operation and administration: Monitoring offline directories and JBOD related metrics is done through Cloudera Manager. However, identifying failed disks and rebalancing partitions between disks is done manually.
- Manual load balancing between disks: Unlike with RAID-10, JBOD does not automatically distribute data across disks. The process is fully manual.

To provide robust JBOD support in Kafka, changes in the Kafka protocol have been made. When performing an upgrade to a new version of Kafka, make sure that you follow the recommended rolling upgrade process.

For more information regarding the JBOD related Kafka protocol changes, see KIP-112 and KIP-113.

Procedure

1. Mount the required number of disks on your system.
2. In Cloudera Manager, set up log directories for all Kafka brokers:
   a) Go to the Kafka service, select Instances and select the broker.
   b) Go to Configuration and find the Data Directories property.
   c) Modify the path of the log directories so that they correspond with the newly mounted disks.

      Note: Depending on your, setup you may need to add or remove multiple data directories.

   d) Enter a Reason for change, and then click Save Changes to commit the changes.
3. Go to the Kafka service and select Configuration.
4. Find and configure the following properties depending on your system and use case.
   - Number of I/O Threads
   - Number of Network Threads
   - Number of Replica Fetchers
   - Minimum Number of Replicas in ISR
5. Set replication factor to at least 3.

   Important: If you set replication factor to less than 3, your data will be at risk. In addition, in case of a disk failure, disk maintenance cannot be carried out without system downtime.

6. Restart the service:
   a) Return to the home page by clicking hte Cloudera Manager logo.
   b) Go to the Kafka service and select Actions Rolling Restart
   c) Check the Restart roles with stale configurations only checkbox and click Rolling restart.
   d) Click Close when the restart has finished.

Results

JBOD disks are set up in your Kafka environment.

Related Information

KIP-112

KIP-113

JBOD Disk migration

Learn how to migrate existing Kafka partitions to JBOD configured disks.

About this task

Migrating data from one disk to another is achieved with the kafka-reassign-partitions tool. The following instructions focus on migrating existing Kafka partitions to JBOD configured disks.

Note: Cloudera recommends that you minimize the volume of replica changes per command instance. Instead of moving 10 replicas with a single command, move two at a time in order to save cluster resources.

Before you begin

- Set up JBOD in your Kafka environment. For more information, see JBOD Setup.
- Collect the log directory paths on the JBOD disks where you want to migrate existing data.
- Collect the broker IDs of the brokers you want to migrate data to.
- Collect the name of the topics you want to migrate partitions from.

Note: Output examples in these instructions are cleaned and formatted to make them easily readable.

Procedure

1. Create a topics-to-move JSON file that specifies the topics you want to reassign. Use the following format:

   Use the following format:

   ```
   {"topics":  [{"topic": "MYTOPIC1"},
                {"topic": "MYTOPIC2"}],
    "version":1
   }
   ```

2. Generate the content for the reassignment configuration JSON with the following command:

   ```
   kafka-reassign-partitions --zookeeper HOSTNAME:PORT --topics-to-move-json-
   file TOPICS TO MOVE.json --broker-list BROKER 1, BROKER 2 --generate
   ```

   Running the command lists the distribution of partition replicas on your current brokers followed by a proposed partition reassignment configuration.

   Example output:

   ```
   Current partition replica assignment
   {"version":1,
    "partitions":
      [{"topic":"mytopic2","partition":1,"replicas":[2,3],"log_dirs":["any","
   any"]},
       {"topic":"mytopic1","partition":0,"replicas":[1,2],"log_dirs":["any"
   ,"any"]},
       {"topic":"mytopic2","partition":0,"replicas":[1,2],"log_dirs":["any","
   any"]},
   ```

```
    {"topic":"mytopic1","partition":2,"replicas":[3,1],"log_dirs":["any"
,"any"]},
    {"topic":"mytopic1","partition":1,"replicas":[2,3],"log_dirs":["any","
any"]}]
}

Proposed partition reassignment configuration

{"version":1,
 "partitions":
   [{"topic":"mytopic1","partition":0,"replicas":[4,5],"log_dirs":["any",
"any"]},
    {"topic":"mytopic1","partition":2,"replicas":[4,5],"log_dirs":["any
","any"]},
    {"topic":"mytopic2","partition":1,"replicas":[4,5],"log_dirs":["any",
"any"]},
    {"topic":"mytopic1","partition":1,"replicas":[5,4],"log_dirs":["any
","any"]},
    {"topic":"mytopic2","partition":0,"replicas":[5,4],"log_dirs":["any",
"any"]}]
}
```

In this example, the tool proposed a configuration which reassigns existing partitions on broker 1, 2, and 3 to brokers 4 and 5.

3. Copy and paste the proposed partition reassignment configuration into an empty JSON file.
4. Modify the suggested reassignment configuration.

   When migrating data you have two choices. You can move partitions to a different log directory on the same broker, or move it to a different log directory on another broker.

   a. 1. To reassign partitions between log directories on the same broker, change the appropriate any entry to an absolute path. For example:

      ```
      {"topic":"mytopic1","partition":0,"replicas":[4,5],"log_dirs":["/
      JBOD-disk/directory1","any"]}
      ```

      2. To reassign partitions between log directories across different brokers, change the broker ID specified in replicas and the appropriate any entry to an absolute path. For example:

      ```
      {"topic":"mytopic1","partition":0,"replicas":[6,5],"log_dirs":["/
      JBOD-disk/directory1","any"]}
      ```

5. Save the file.
6. Start the redistribution process with the following command:

   ```
   kafka-reassign-partitions --zookeeper HOSTNAME:PORT  --reassignment-json-
   file REASSIGNMENT CONFIGURATION.json --bootstrap-server HOSTNAME:PORT --e
   xecute
   ```

   Important: The bootstrap server has to be specified with the --bootstrap-server option if an absolute log directory path is specified for a replica in the reassignment configuration JSON file.

   The tool prints a list containing the original replica assignment and a message that reassignment has started. Example output:

   ```
   Current partition replica assignment

   {"version":1,
    "partitions":
      [{"topic":"mytopic2","partition":1,"replicas":[2,3],"log_dirs":["any",
   "any"]},
   ```

```
    {"topic":"mytopic1","partition":0,"replicas":[1,2],"log_dirs":["any
","any"]},
    {"topic":"mytopic2","partition":0,"replicas":[1,2],"log_dirs":["any",
"any"]},
    {"topic":"mytopic1","partition":2,"replicas":[3,1],"log_dirs":["any
","any"]},
    {"topic":"mytopic1","partition":1,"replicas":[2,3],"log_dirs":["any",
"any"]}]
}

Save this to use as the --reassignment-json-file option during rollback
Successfully started reassignment of partitions.
```

7. Verify the status of the reassignment with the following command:

```
kafka-reassign-partitions --zookeeper HOSTNAME:PORT --reassignment-json-
file REASSIGNMENT CONFIGURATION.json  --bootstrap-server HOSTNAME:PORT --v
erify
```

The tool prints the reassignment status of all partitions. Example output:

```
Status of partition reassignment:
Reassignment of partition mytopic2-1 completed successfully
Reassignment of partition mytopic1-0 completed successfully
Reassignment of partition mytopic2-0 completed successfully
Reassignment of partition mytopic1-2 completed successfully
Reassignment of partition mytopic1-1 completed successfully
```

Results

Existing Kafka partitions are migrated to JBOD configured disks.

Related Information

JBOD setup

kafka-reassign-partitions

Setting user limits for Kafka

Learn more about Kafka User limits and how to monitor them.

Kafka opens many files at the same time. The default setting of 1024 for the maximum number of open files on most Unix-like systems is insufficient. Any significant load can result in failures and cause error messages such as java.io.IOException...(Too many open files) to be logged in the Kafka or HDFS log files. You might also notice errors such as this:

```
ERROR Error in acceptor (kafka.network.Acceptor)
java.io.IOException: Too many open files
```

Cloudera recommends setting the value to a relatively high starting point, such as 32,768.

You can monitor the number of file descriptors in use on the Kafka Broker dashboard. In Cloudera Manager:

1. Go to the Kafka service.
2. Select a Kafka Broker.
3. Open Charts Library Process Resources and scroll down to the File Descriptors chart.

Connecting Kafka clients to Cloudera Data Hub provisioned clusters

Learn how to connect Kafka clients to clusters provisioned with Cloudera Data Hub.

About this task

Use the following steps to connect Kafka clients to clusters provisioned with Cloudera Data Hub. Configuration examples provided in this list of steps assume that the cluster you are connecting to was provisioned with a Streams Messaging cluster definition.

Note:

These instructions are for connecting Kafka clients to Cloudera Data Hub provisioned Kafka clusters. For information on connecting NiFi to Kafka within the same Cloudera Public Cloud environment, see Ingesting data into Apache Kafka.

Before you begin

- If you are connecting your clients from outside of your virtual network (VPC or Vnet) verify that both inbound and outbound traffic is enabled on the port used by Kafka brokers for secure communication. The default port is 9093. For more information, see the following resources:
  - AWS: Security Groups for Your VPC
  - Azure: How to open ports to a virtual machine with the Azure portal
- If you are connecting your clients over the internet, verify that your virtual network (VPC or Vnet) is assigned a public IP address. For more information, see the following resources:
  - AWS: IP Addressing in Your VPC
  - Azure: Associate a public IP address to a virtual machine
- Clients connecting to Cloudera Data Hub provisioned clusters require a Cloudera user account that provides access to the required Cloudera resources. Verify that a Cloudera user account with the required roles and permissions is available for use. If not, create one. Any type of Cloudera user account can be used. If you are creating a new account to be used by Kafka clients, Cloudera recommends that you create a machine user account. For more information, see User Management in the Cloudera Management Console documentation.
- In addition to the Cloudera user account having access to the required Cloudera resources, the user account also needs to have the correct policies assigned to it in Ranger. Otherwise, the client cannot perform tasks on Kafka resources. These policies are specified within the Ranger instance that provides authorization to the Kafka service you want to connect to. For more information, see the Cloudera Runtime documentation on Apache Ranger and the Kafka specific Ranger documentation.

Procedure

1. Obtain the FreeIPA certificate of your environment:
   a) From the Cloudera Home Page navigate to Management Console Environments .
   b) Locate and select your environment from the list of available environments.
   c) Go to the FreeIPA tab.
   d) Click Get FreeIPA Certificate.
   The FreeIPA certificate file, *[***ENVIRONMENT NAME***]*.crt, is downloaded to your computer.

2. Add the FreeIPA certificate to the truststore of the client.

   The certificate needs to be added for all clients that you want to connect to the Cloudera Data Hub provisioned cluster. The exact steps of adding the certificate to the truststore depends on the platform and key management software used. For example, you can use the Java Keytool command line tool:

```
keytool -import -keystore [***CLIENT TRUSTSTORE.JKS***] -al
ias [***ALIAS***] -file [***FREEIPA CERTIFICATE***]
```

   Tip: This command creates a new truststore file if the file specified with the -keystore option does not exist.

3. Obtain Cloudera workload credentials:

   A valid workload username and password has to be provided to the client, otherwise it cannot connect to the cluster. Credentials can be obtained from Management Console.

   a) From the Cloudera Home Page navigate to Management Console User Management .
   b) Locate and select the user account you want to use from the list of available accounts.
      The user details page displays information about the user.
   c) Find the username found in the Workload Username entry and note it down.
   d) Find the Workload Password entry and click Set Workload Password.
   e) In the dialog box that appears, enter a new workload password, confirm the password and note it down.
   f) Fill out the Environment text box.
   g) Click Set Workload Password and wait for the process to finish.
   h) Click close.

4. Configure clients.

   In order for clients to be able to connect to Kafka brokers, all required security related properties have to be added to the client's properties file. The following example configuration lists the default properties that are needed when connecting clients to a cluster provisioned by Cloudera Data Hub with a Streams Messaging cluster definition. If you made changes to the security configuration of the brokers, or provisioned a custom cluster with non-default Kafka security settings, make sure to change the appropriate parameters in the client configuration as well.

```
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
ssl.truststore.location=[***CLIENT TRUSTSTORE.JKS***]
ssl.truststore.password=[***TRUSTSTORE PASSWORD***]
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
 required \
    username="[***USERNAME***]" \
    password="[***PASSWORD***]";
```

   Replace *[***CLIENT TRUSTSTORE.JKS***]* with the path to the client's truststore file.This is the same file that you added the FreeIPA certificate to in Step 2 on page 13.

   Replace *[***TRUSTSTORE PASSWORD***]* with the password of the truststore file.

   Replace *[***USERNAME***]* and *[***PASSWORD***]* with the workload username and password obtained in Step 3 on page 13.

5. Obtain Kafka broker hostnames:

   You can obtain the Kafka broker hostnames from the Cloudera Manager UI.

   a) From the Cloudera Home Page navigate to Management Console Environments .
   b) Locate and select your environment from the list of available environments.
   c) Select the Cloudera Data Hub cluster you want to connect to from the list of available clusters.
   d) Click the link found in the Cloudera Manger Info section.
      You are redirected to the Cloudera Manager web UI.
   e) Click Clusters and select the cluster that the Kafka service is running on.

      The default name for clusters created with a Streams Messaging Cluster definition is streams-messaging.
   f) Select the Kafka service.
   g) Go to Instances.

      The Kafka broker hostnames are listed in the Hostname column.

6. Connect clients to brokers.

   Connect the clients by supplying them with the broker hostnames obtained in step 5 on page 14. The actions you need to take differ depending on the type of client you are using.

   Custom developed Kafka Applications

   When producing or consuming messages with your own Kafka client application, you have to provide the Kafka broker hostnames within the client code.

   Kafka console producer and consumer

   When producing or consuming messages with the kafka-console-consumer or kafka-console-producer command line tools, run the producer or consumer with the appropriate hostnames. Additionally, you must also pass the client properties file containing the security related properties with --producer.config or --consumer.config. For example:

   ```
   kafka-console-producer --broker-
   list [***HOSTNAME***]:[***PORT***] --topic [***TOPIC***] --produ
   cer.config [***CLIENT PROPERTIES FILE***]
   ```

   ```
   kafka-console-consumer --bootstrap-serve
   r [***HOSTNAME***]:[***PORT***] --topic [***TOPIC***] --from-begi
   nning --consumer.config [***CLIENT PROPERTIES FILE***]
   ```

   Replace *[***CLIENT PROPERTIES FILE***]* with the path to the client's properties file. This is the same file that you updated in Step 4 on page 13.

7. To connect to Schema Registry, you must also set the following properties:

   ```
   schema.registry.url=<SCHEMA_REGISTRY_ENDPOINT>
   schema.registry.auth.username=<USERNAME>
   schema.registry.auth.password=<PASSWORD>
   ```

   - If schema.registry.url is not set, the client does not try to connect to Schema Registry.
   - If this property is set, the username and password must also be configured.
   - You can find the Schema Registry endpoint in Management Console Data Hub Clusters [***YOUR DATA HUB CLUSTER***] Endpoints .

   Some applications may need to connect directly to Schema Registry to interact with it. For example, to retrieve or store a schema.

   A Schema Registry API client is also provided by Cloudera for these cases. For example, to retrieve the latest version of a schema you can use the following:

   ```
   Map<String, Object> config = new HashMap<>();
   config.put("schema.registry.url", "<SCHEMA_REGISTRY_ENDPOINT>");
   config.put("schema.registry.auth.username", "<USERNAME>");
   ```

```
config.put("schema.registry.auth.password", "<PASSWORD>");

String topicName = "my-topic";
SchemaRegistryClient client = new SchemaRegistryClient(config);
try {
    SchemaVersionInfo latest = client.getLatestSchemaVersionInfo(topicNam
e);
    System.out.println(latest.getSchemaText());
} catch (SchemaNotFoundException e) {
    LOG.info("Schema [{}] not found", topicName);
    throw e;
}
```

Results

Kafka clients and Schema Registry are configured and are able to connect to Cloudera Data Hub provisioned clusters.

Rolling restart checks

You can configure Cloudera Manager to perform a check on Kafka brokers during a rolling restart. Using this check can ensure that Kafka brokers stay healthy after the rolling restart. There are multiple types of checks available, each providing a different level of guarantee on Kafka broker and cluster health.

By default, during a rolling restart, Cloudera Manager only checks whether restarting a Kafka broker has failed or succeeded. As a result of this behaviour, Kafka brokers might go into a state where some of the topics and partitions become unreachable by the clients. For example, by default Cloudera Manager might restart a broker while the previous broker is not fully ready for operation. This can cause outages and corrupted log indexes. To avoid such issues, you can configure Cloudera Manager to perform a more thorough check on the Kafka brokers during a rolling restart.

Check types and configuration

There are multiple checks available, each providing a different (higher) level of guarantee on Kafka cluster and broker health. The type of check performed is configured with the Cluster Health Guarantee During Rolling Restart property. The property has four different settings, each setting corresponds to a different type of check. The available settings and the check types that the settings correspond to are as follows:

none

This setting disables rolling restart checks. If this option is selected, no checks are performed and no health guarantees are provided.

ready for request

This setting ensures that when a broker is restarted, the restarted broker is accepting and responding to requests made on its service port. The next broker is only restarted after the previous broker is ready for requests.

healthy partitions stay healthy (default)

This setting ensures that no partitions go into an under-min-isr state when a broker is stopped. This is achieved by waiting before each broker is stopped so that all other brokers can catch up with all replicas that are in an at-min-isr state. Additionally, this setting ensures that the restarted broker is accepting and is responding to requests made on its service port before restarting the next broker. This setting ignores partitions which are already in an under-min-isr state.

all partitions stay healthy (recommended)

This setting ensures that no partitions are in an under-min-isr or at-min-isr state when a broker is stopped. This is achieved by waiting before each broker is stopped so that all other brokers can catch up with all replicas that are in an at-min-isr or under-min-isr state. Additionally, this setting

ensures that the restarted broker is accepting requests on its service port before the next broker is restarted.

all partitions fully replicated

This setting ensures that all partitions are in a fully synchronized state when a broker is stopped. This is achieved by waiting before each broker is stopped so that all other brokers can catch up with all replicas that are out of sync. Additionally, this setting ensures that the restarted broker is accepting requests on its service port before the next broker is restarted.

In addition to configuring and enabling these checks using Cluster Health Guarantee During Rolling Restart, a number of other configuration properties are also available that enable you to fine-tune the behaviour of the checks. For detailed steps on how to enable and configure rolling restart checks, see *Configuring rolling restart checks*.

How checks are performed

When Cloudera Manager executes a rolling restart check, it uses the kafka-topics tool to gather information about the brokers, topics, and partitions. The kafka-topics tool requires a valid client configuration file to run. In the case of rolling restart checks, two configuration files are required. One for the kafka-topics commands that are initiated before a broker is stopped, and a separate one for the commands initiated after a broker is restarted. Cloudera Manager automatically generates these client configuration files based on the configuration of the Kafka service. These files can also be manually updated using advanced security snippets.

Using these files, Cloudera Manager executes kafka-topics commands on the brokers. Based on the response from the tool, Cloudera Manager either waits for a specified amount of time or continues with the rolling restart.

Depending on what type of check is configured, Cloudera Manager polls information with kafka-topics at different points in time. As a result, the checks can be categorised in two groups. Pre-checks and post-checks. If either healthy partitions stay healthy or all partitions stay healthy is selected, information is polled both before a broker is stopped (pre-check) and after a broker is restarted (post-check). If the ready for request setting is selected, information is only polled after a broker is restarted.

If a pre-check fails to find a proper state when a broker can be stopped, the check will stop the entire rolling restart process. This can happen if the broker that is about to be stopped still has at-min-isr or under-min-isr partitions after the configured timeout interval is reached. Post-checks behave in a similar way. If the post-check fails to receive validation (a correct exit code) within the specified timeout interval from the kafka-topics command that the broker is ready for requests, the check will stop the entire rolling restart process. In both of these cases the brokers are not stopped or restarted. The rolling restart fails and the brokers continue to run.

Note: Configuring and using any type of check increases the time required for a rolling restart. This is the result of Cloudera Manager waiting between restarting the brokers. The timeout intervals however can be configured to a lower value if the rolling restart check takes too much time to finish. Alternatively, if you are experiencing timeout related rolling restart failures, you can also configure the timeout intervals to a higher value.

Advanced configuration

There are two scenarios when additional configuration is required. These scenarios are as follows:

Kafka brokers are configured to use a custom listeners

If you configured your Kafka brokers with advanced configuration snippets to use custom listeners (for example a custom host:port pair), you must manually update both client configuration files that Cloudera Manager generates. This is required because Cloudera Manager might not be able to automatically extract the information required to establish a connection with the Kafka brokers when custom listeners are configured. For more information, see *Configuring the client configuration used for rolling restart checks*.

A broker connectivity change is made after rolling restart checks are enabled

A broker connectivity change is any type of change made to listeners, bootstrap servers, ports, or security. If a change like this is made after rolling restart checks are enabled, Cloudera Manager uses the newly set configuration to generate the client configuration files. However, until a restart is

executed, the Kafka brokers still operate with the old configuration. As a result, Cloudera Manager will run the kafka-topics tool with an invalid configuration causing the check and the rolling restart to fail. In a case like this, you must disable rolling restart checks until the Kafka brokers are restarted at least once. This can be done by setting Cluster Health Guarantee During Rolling Restart to none. Following the initial restart, the brokers will operate with the new configuration and you can re-enable rolling restart checks.

Configuring rolling restart checks

You can configure Cloudera Manager to perform different types of checks on Kafka brokers during a rolling restart. The type of check performed by Cloudera Manager is configured with the Cluster Health Guarantee During Rolling Restart property. The property has multiple settings, each setting corresponds to a different type of check.

About this task

The following steps walk you through the basic configuration method of how you can enable and configure rolling restart checks.

Before you begin

If your Kafka service is configured to use custom listeners, complete Configuring the client configuration used for rolling restart checks before continuing with this task.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find and configure the Cluster Health Guarantee During Rolling Restart property.

   Select one of the available options. Click the ? icon next to the property name to reveal a full description of each option and the check to which they correspond.

   Cloudera recommends that you set this property to all partitions stay healthy to avoid service outages.
4. Fine-tune rolling restart check behaviour by configuring the following properties:
   - Maximum Allowed Runtime For Kafka Broker Rolling Restart Check
   - Retry Interval For Kafka Broker Rolling Restart Check
   - Default API Timeout For Kafka Topics Client Used In Kafka Broker Rolling Restart Check

   These properties allow you to configure different interval and timeout values related to the rolling restart check. Configure these properties based on your cluster and requirements.
5. Click Save Changes.
6. Restart the Kafka service.

Results

Rolling restart checks are configured and enabled. During any subsequent rolling restarts, Cloudera Manager executes the type of check you configured.

What to do next

If you make any configuration changes related to broker connectivity (security, listeners, port, bootstrap) after rolling restart checks are enabled, you must disable rolling restart checks for the first restart after the change was made. Otherwise, the check and the rolling restart might fail. Following the initial restart, you can re-enable rolling restart checks.

Note: If Kafka brokers are configured with JVM properties that specify a unique port, such as when configuring a JMX agent, ensure that you configure the Kafka Broker JVM properties in the Additional Broker Java Options configuration item instead of adding the properties using environment variables. This prevents rolling restart check failures caused by port collisions.

Configuring the client configuration used for rolling restart checks

Cloudera Manager requires Kafka client configuration files to perform rolling restart checks. These files are generated automatically. However, if your Kafka service has custom listeners configured, you must manually update these client configuration files. Otherwise, the rolling restart check might fail.

About this task

When Cloudera Manager executes a rolling restart check, it uses the kafka-topics tool to gather information about the brokers, topics, and partitions. The kafka-topics tool requires a valid client configuration file to run. Cloudera Manager automatically generates two configuration files for this purpose. One is used for the kafka-topics commands initiated before the brokers are stopped, the other, after brokers are restarted.

If your Kafka service is configured to use custom listeners, you must manually update the configuration files generated by Cloudera Manager. This is required because Cloudera Manager might not be able to automatically extract the information required to establish a connection with the Kafka service when custom listeners are configured. The client configuration files can be updated using advanced security snippets.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Manually update the client configuration files used during rolling restart checks.

   This can be done by adding a valid client configuration to the following advanced configuration snippets:

   - Kafka Broker Advanced Configuration Snippet (Safety Valve) for rolling_restart_check_before_stop_admin_client_configs.properties
   - Kafka Broker Advanced Configuration Snippet (Safety Valve) for rolling_restart_check_after_start_admin_client_configs.properties

   Ensure that you add the same client configuration to both snippets. The client configuration you add must contain all properties that are required to establish a connection with the brokers. The client configuration you add here is similar to any other client configuration you create for Kafka command line tools. However, this specific configuration accepts the bootstrap.servers property. Use this property to specify your custom host:port pairs that you use as your custom listeners.

   The following client configuration example is for a Kafka service that has both TLS/SSL and Kerberos enabled. You can use this example as a template and make changes as needed. For more client configuration examples, see the Securing Apache Kafka publication in the *Streams Messaging* documentation.

```
bootstrap.servers=[***HOST***]:[***PORT***]
security.protocol=SASL_SSL
ssl.client.auth=none
sasl.mechanism=GSSAPI
sasl.kerberos.service.name=kafka
sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule requir
ed useKeyTab=true storeKey=true keyTab="[***PATH TO KEYTAB***]" princi
pal="[***KERBEROS PRINCIPAL***]";
ssl.keystore.location=[***PATH TO KEYSTORE.JKS***]
ssl.key.password=[***PASSWORD***]
ssl.keystore.password=[***PASSWORD***]
ssl.keystore.type=jks
ssl.truststore.location=[***PATH TO TRUSTSTORE.JKS***]
```

```
ssl.truststore.type=jks
ssl.truststore.password=[***PASSWORD***]
```

4. Click Save Changes.

Results

The client configuration files used by Cloudera Manager during rolling restart checks are configured.

What to do next

Enable and configure rolling restart checks. Complete *Configuring rolling restart checks*.

Related Information

Streams Messaging

Configuring rolling restart checks

Cluster discovery with multiple Apache Kafka clusters

When you have multiple Kafka clusters for load balancing or failover purposes, a client can find a suitable cluster either using a DNS server or using a load balancer.

A common architectural pattern for Kafka is to have redundant Kafka clusters across multiple data centers so that applications can either load balance or fail over to the remote data center in case the local one becomes unavailable. Finding a suitable cluster by the client is called cluster discovery. If you connect to a cluster, you do not need all the brokers, one broker is enough. (However, to avoid a single point of failure, it is better to have multiple brokers.) This means that your discovery service should provide at least one active broker. The client gets metadata of all the cluster members after connecting to the broker it got from the service/mechanism. Getting brokers from the discovery service is only useful for the bootstrap phase of the connection. The client must connect to all the brokers directly to operate properly.

The two options for cluster discovery in case of multiple Apache Kafka clusters are using DNS records or load balancers. Cluster discovery using DNS records is the simpler solution.

Cluster discovery using DNS records

Learn about using a DNS server, which is the simplest cluster discovery method.

Kafka has built-in support for cluster discovery using a DNS server as a discovery service. In this case, the client uses a DNS server to resolve hostname aliases and then according to the response, it connects directly to a broker:

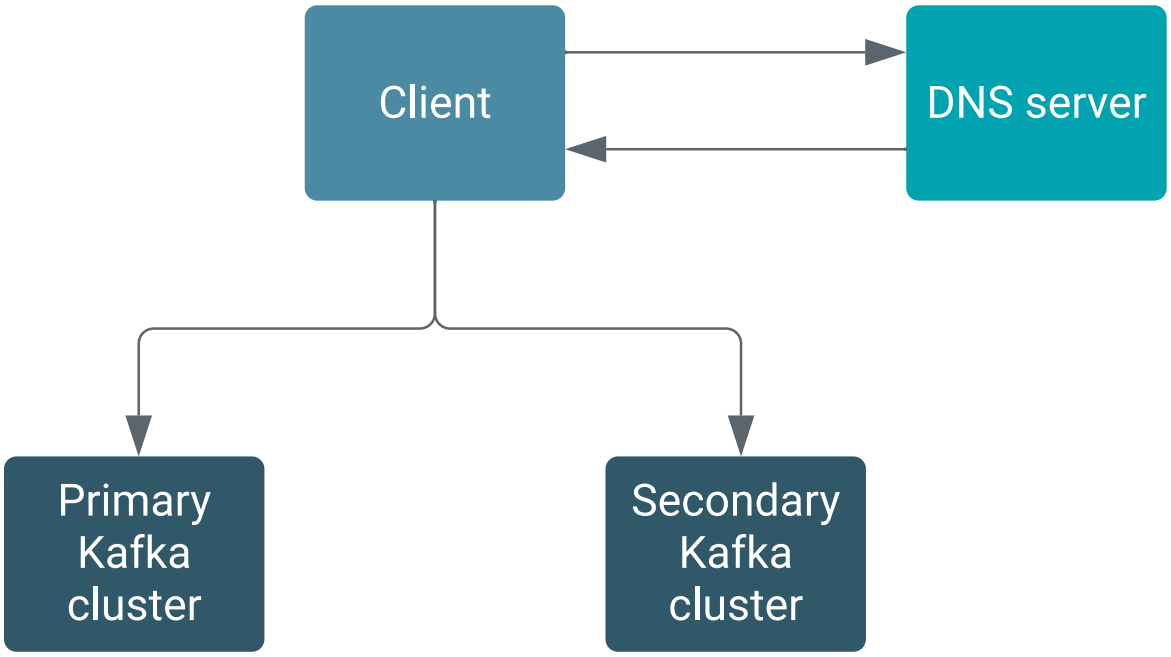

A possible solution is shown in the following sections using some DNS records with some examples. The examples are given using a possible syntax for a BIND DNS server, but any other DNS server can be used for the solution. BIND is used here for simplicity.

A records and round robin DNS

Learn about A records and round robin DNS.

A DNS A record is an entry that holds a hostname and the corresponding IP address. You can have multiple A records for different IP addresses using the same hostname as follows (the brokers also have their own FQDNs):

```
; PRIMARY CLUSTER
first.primary.kafka.fqdn.          IN  A       1.2.3.4
second.primary.kafka.fqdn.         IN  A       4.5.6.7
third.primary.kafka.fqdn.          IN  A       6.7.8.9

; DNS alias
primary.kafka.cluster.hostname.    IN  A       1.2.3.4
primary.kafka.cluster.hostname.    IN  A       4.5.6.7
primary.kafka.cluster.hostname.    IN  A       6.7.8.9
```

If you try to resolve primary.kafka.cluster.hostname multiple times with any tools (nslookup, dig), you will get different results each time. This is called round robin load balancing, which is used by the DNS server automatically.

The /etc/hosts file holds the IP address and hostname mappings (similar to A records in DNS servers), but it is not capable of balancing. You always get the same IP if you try to resolve the same hostname.

In production environments, a low TTL might be used for DNS records for the clients to detect changes as early as possible.

client.dns.lookup property options for client

If you are using DNS aliases, you need to configure the right value for the client.dns.lookup property for your setup.

The client.dns.lookup property is needed when DNS aliases are used. There can be problems with any security protocol if the client.dns.lookup property is not set properly.

The client.dns.lookup property can take the following values:

- client.dns.lookup=default

  In this case, the client connects to the first broker it gets from the DNS response, even if that broker is down. If you only provided a single DNS alias as bootstrap server for the client, the client might be blocked because the mentioned hostname is resolved for multiple brokers where the first broker (to which the client connected) is actually stopped.
- client.dns.lookup=use_all_dns_ips

  This setup takes care of retrying all the possible IP addresses behind a hostname if the first does not succeed. This setup is also good when brokers have multiple IP addresses and it is possible that those addresses change (especially in Kubernetes environments). This configuration does not handle the case when the hostname used by the client is not a real FQDN for a host. To avoid man-in-the-middle attacks, the client gets SSL handshake exceptions when using SSL and SASL authentication exceptions when using SASL. Since Kafka version 2.6, this is the default.
- client.dns.lookup=resolve_canonical_bootstrap_servers_only

  This setup ensures a behavior similar to use_all_dns_ips, but it also handles the SSL and SASL problem. With this configuration, the client resolves the DNS alias into a list of brokers it maps to and after the bootstrap phase, this behaves the same as use_all_dns_ips.

  Tip: Cloudera recommends that you set the resolve_canonical_bootstrap_servers_only value because this option provides the most fault tolerance.

CNAME records configuration

When you use Kafka's built-in support for cluster discovery you can use CNAME records as shorter alternatives to the longer hostname that is an alias.

In addition to A records, it is also possible to have CNAME records in the DNS servers. It is good to have a simpler/shorter hostname as an alternative for the one that maps to all the brokers.

```
;CNAME record
active.kafka.     IN CNAME        primary.kafka.cluster.hostname.
```

It this case, clients can use the simpler and shorter alternative for the hostname as follows:

```
kafka-topics --list --bootstrap-server active.kafka:9092
```

Using a CNAME also provides the possibility to easily switch between standby and active clusters. Assume there is also a standby cluster defined:

```
; PRIMARY CLUSTER
first.primary.kafka.fqdn.          IN  A       1.2.3.4
second.primary.kafka.fqdn.         IN  A       4.5.6.7
third.primary.kafka.fqdn.          IN  A       6.7.8.9

; DNS alias
primary.kafka.cluster.hostname.    IN  A       1.2.3.4
primary.kafka.cluster.hostname.    IN  A       4.5.6.7
primary.kafka.cluster.hostname.    IN  A       6.7.8.9

; STANDBY CLUSTER
first.standby.kafka.fqdn.          IN  A       4.3.2.1
second.standby.kafka.fqdn.         IN  A       7.6.5.4
third.standby.kafka.fqdn.          IN  A       9.8.7.6

; DNS alias
standby.kafka.cluster.hostname.    IN  A       4.3.2.1
standby.kafka.cluster.hostname.    IN  A       7.6.5.4
```

```
standby.kafka.cluster.hostname.    IN  A       9.8.7.6

;CNAME record
active.kafka.     IN CNAME        primary.kafka.cluster.hostname.
```

If the primary cluster completely stops, it is only needed to change the active.kafka CNAME to point to standby.kafka.cluster.hostname instead of the primary one:

```
active.kafka.     IN CNAME        standby.kafka.cluster.hostname.
```

Important: Even though using of CNAME records is convenient, they can double the number of DNS queries.

Connection to the cluster with configured DNS aliases

If you only want to use a single hostname for the whole cluster, configure a DNS alias.

This A record setup is a convenient solution for having a single hostname for the whole cluster. You only need to provide a hostname that serves as an alias for the brokers:

```
kafka-topics --list --bootstrap-server primary.kafka.cluster.hostname:9092
```

When connecting your clients to the brokers, the DNS alias is specified and then translated to the actual hostname.

However, a port number must also be specified, and the port you specify is used for all brokers identified by the DNS alias. As a result of this, you must ensure that all brokers identified by a specific DNS alias use the same port number. Otherwise, clients fail to connect even if DNS resolution is successful.

In case of a broker failure, you can change the records in the DNS server to point to other brokers. The next time a client tries to connect, it resolves to a different cluster's broker. This way you do not hardcode broker FQDNs to the bootstrap server list.

If the client has discovered a cluster and is actively using it, and the cluster suddenly stops, the client tries to connect to the bootstrap servers it got from the DNS server before the bootstrap phase. So it does not automatically ask the DNS server for new bootstrap servers and does not fail over to the other cluster, even if the DNS records have already been changed.

Cluster discovery using load balancers

In case of more complex Kafka cluster setups, you might need a cluster discovery solution more sophisticated than a DNS server. In such cases, you should consider using a load balancer.

Cluster discovery using load balancers is less lightweight than using DNS servers, but a viable solution for more complex cases. With a load balancer, it is possible to poll the nodes, check their health status and exclude stopped nodes from targets, automatically redirect requests to living nodes. When using a load balancer the requests are forwarded to a broker as shown in the figure:

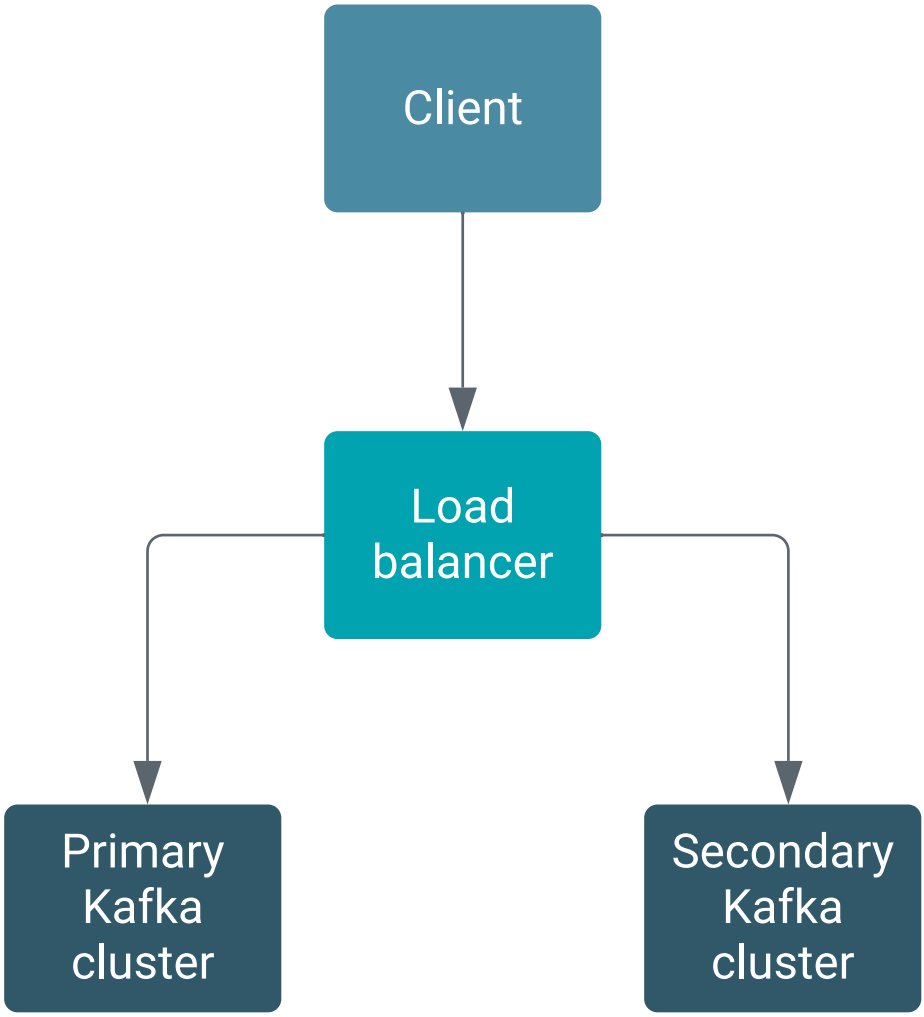

Because the client connects to the load balancer and is then forwarded to a broker, SSL handshake and SASL authentication errors can occur (this is a defending mechanism to avoid man-in-the-middle attacks), therefore, additional setup is needed.

The configuration depends on your security protocol:

- No security in the cluster (security protocol is PLAINTEXT)

  Setup steps are not required before connecting to the Kafka cluster, the load balancers should work out of the box with Kafka.
- SASL with Kerberos enabled

  Perform the setup described in section *Setup for SASL with Kerberos*.
- TLS/SSL encryption is enabled

  Perform the setup described in section *Setup for TLS/SSL encryption*.
- SASL with Kerberos and TLS/SSL are both enabled

  Perform the setup described in *Setup for SASL with Kerberos* and *Setup for TLS/SSL encryption*.

Related Information

Setup for SASL with Kerberos

Setup for TLS/SSL encryption

Setup for SASL with Kerberos

If you are using SASL with Kerberos for authentication, you must configure the load balancer and select the relevant architecture in Cloudera Manager.

Setting the Kafka Broker Load Balancer Host property

Learn how to configure the Kafka Broker Load Balancer Host property to avoid a ticket mismatch when using SASL with Kerberos.

About this task

If you are using SASL with Kerberos in the system and your clients connect to a load balancer that forwards requests to the actual brokers, the client gets a Kerberos service ticket including the load balancer host. This happens because the client believes that the load balancer does not forward requests but it is a broker. Meanwhile, the Kafka brokers in the configured listeners are logged in to Kerberos using their service principal that contains their corresponding FQDN. This results in a ticket mismatch after the client is routed to an actual broker with the load balancer related service ticket.

If you are using SASL_SSL or SASL_PLAINTEXT security protocol in the cluster with Kerberos, you have to set the load balancer's host in Cloudera Manager for the Kafka broker roles by setting the Kafka Broker Load Balancer Host property.

Before you begin

Ensure that you have reviewed Kerberos-related architecture options.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to the Configuration tab.
3. Find and configure the Kafka Broker Load Balancer Host property.

   Add the hostname of the load balancer.
4. Find and configure the Kafka Broker Load Balancer Listener Port property.

   Important: Set the load balancer to forward requests to this port.
5. Click Save Changes.
6. Restart the Kafka service.

Results

After restarting the brokers, the following is automatically set:

- A new load balancer principal is added for each broker's kafka.keytab:

  ```
  kafka_principal/load_balancer_host@REALM
  ```

  By default, the Kafka principal in the cluster is kafka.
- A new listener named LB is added to the kafka.properties:

  ```
  listeners=LB://HOST:9094
  ```

  The LB listener port can be configured by setting the Kafka Broker Load Balancer Listener Port property.

- Similarly to listeners, a new advertised listener is added to the kafka.properties:

  ```
  advertised.listeners=LB://HOST:9093
  ```

  The LB advertised listener advertises the normal Kafka listener's port that is also automatically set (9092 or 9093 depending on SSL settings).
- The security protocol for the new LB listener is added to its corresponding map configuration with the same security protocol as set for the normal listener the clients connect to without using the load balancer:

  ```
  listener.security.protocol.map=LB:SASL_SSL
  ```

- The JAAS configuration for the LB listener is set to log in with the load balancer principal using the actual Kafka process folder for the keytab path:

  ```
  listener.name.lb.gssapi.sasl.jaas.config=com.sun.security.auth.module.Kr
  b5LoginModule
          required doNotPrompt=true useKeyTab=true storeKey=true
          keyTab="/process_folder/kafka.keytab"
          principal="kafka_principal/load_balancer_host@REALM";"
  ```

Note: If you have already set any of these properties using advanced configuration snippets (safety valves), the feature does not overwrite your changes. It might simply not work, so you have to extend your advanced configuration snippets manually with the mentioned settings.

- LB can be used with uppercase letters if the load balancer listener should be referenced in property values.
- lb can be used with lowercase letters if the load balancer listener should be referenced in property keys.

Turning off the load balancer listener

If you are no longer using the load balancer, you need to turn off the load balancer listener and restart the Kafka service.

About this task

If the load balancer is not used anymore in front of Kafka brokers, it is enough to clear the Kafka Broker Load Balancer Host property to make it empty and restart the Kafka service. The Kafka broker will not use the load balancer listener port anymore with its listener.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to the Configuration tab.
3. Find and clear the Kafka Broker Load Balancer Host property.
4. Click Save Changes.
5. Restart the Kafka service.

Kerberos-related architecture options for Kafka broker behind load balancer setup

Learn about possible Kerberos-related architectures before setting up Kafka brokers behind a load balancer to use SASL with Kerberos.

There are possible Kerberos-related architectures with their own tradeoffs. It is important to consider them to design the system according to needs before setting up Kafka brokers behind a load balancer. There can be differences between the number of Cloudera Manager and Key Distribution Center (KDC) instances connected with the Kafka clusters.

- Single Cloudera Manager , single KDC instance

  This is the simplest and easiest setup. All the Kafka clusters are managed by a single Cloudera Manager instance that uses a single KDC instance. In this case, the load balancer feature can be enabled in all the Kafka clusters and the rest is enabled automatically.

- Multiple Cloudera Manager, multiple KDC instances

  In this case, all the Kafka clusters have their own separate Cloudera Manager and KDC instances. The load balancer feature can be enabled in all the Kafka clusters, but cross-realm trust should be set by the system administrator for the client principals to be able to authenticate to any of the Kafka services. The load balancer listeners log in into their own KDC using the load balancer principal, and the Kafka client needs a credential that can log in into any of the KDCs.
- Multiple Cloudera Manager, single KDC instances

  If you use multiple Kafka clusters with multiple Cloudera Manager instances and yet they use the same single KDC, the feature does not work, because the CM servers invalidate each other's load balancer principal credentials in the keytab when getting the load balancer principal. Only one Cloudera Manager instance's Kafka cluster load balancer credentials will be fine, others will be stale. In this case, after enabling the feature as mentioned above, it is possible to override keytab or principal for the load balancer listener setting the following property as safety valve:

  ```
  listener.name.lb.gssapi.sasl.jaas.config=com.sun.security.auth.module.Kr
  b5LoginModule required doNotPrompt=true useKeyTab=true storeKey=true key
  Tab="/tmp/loadbalancer.keytab" principal="load_balancer_principal";"
  ```

  The keytab can be created manually or the ones that are actually valid in one of the Kafka clusters can be copied to other cluster brokers. It is important to make sure that the process has permissions to read the keytab files because the permissions can change while copying keytabs from one host to another. The mentioned safety valve is also a way to override the load balancer principal if the generated one is not appropriate (or if the credentials are created manually and the administrator chose a different principal name than the default Kafka). Whichever option you use, it is important for the host part to be the same as load balancer host; otherwise, the forwarded requests will fail.

Setup for TLS/SSL encryption

If you are using TLS/SSL encryption, you need to select a method to resolve SSL hostname verification failure.

If TLS encryption is used and a client connects to the load balancer host, the SSL hostname verification fails on the Kafka client side, because the client compares the hostnames in the broker certificates with the actual hostnames that are used in bootstrap.servers for the connection.

You can use one of the following methods to prevent an SSL hostname verification failure.

- Using Subject Alternative Name (SAN) in the certificates

  The optimal solution for the SSL hostname verification is to add the load balancer hostname as a SAN to the certificates of each broker.

  Important: This is currently not done automatically with AutoTLS.

- Using wildcard certificates

  If the load balancer and the brokers are in the same domain, you can also use wildcard certificates where it is not needed to enumerate the brokers and the load balancer one by one. Ensure you include the domain in the certificate.
- Disabling hostname verification on the client side

  If modifying the certificates is a big effort, it is also possible to disable the hostname verification on the Kafka client side. The clients should include an empty string for the SSL algorithm:

  ```
  ssl.endpoint.identification.algorithm=
  ```

Connecting to the Kafka cluster using load balancer

Learn how to connect a client to a Kafka cluster that is located behind a load balancer.

Before you begin

If you have set the Kafka Broker Load Balancer Listener Port property during broker configuration, ensure that you have also configured the load balancer to forward requests to the port number specified in Kafka Broker Load Balancer Listener Port.

Procedure

To connect to the Kafka cluster using the load balancer host and port, use the following command:

```
kafka-topics --list --bootstrap-server load_balancer_host:load_balancer_port
```

The load balancer is only used for connection bootstrap. The clients still need to be able to connect to the brokers directly. This also means that if the client has discovered a cluster and is actively using it and the cluster suddenly stops, the client tries to connect to the bootstrap servers it discovered in the bootstrap phase as metadata, so it does not automatically ask the load balancer for new bootstrap servers and it does not fail over to the other cluster even if the load balancer already routes new requests to the other cluster.

Configuring Kafka ZooKeeper chroot

By default, the /kafka path is used in ZooKeeper to store Kafka related metadata. This path can be changed by configuring the ZooKeeper Root Kafka property.

About this task

Complete the following steps to change the Kafka ZooKeeper chroot on an already existing service. You can also configure the ZooKeeper Root property when adding a new Kafka service to a cluster. The property can be configured on the Review Changes page when using the Add a Service wizard.

Important: Configuring the Kafka ZooKeeper chroot must be done during broker setup, before the broker is started for the first time. If the property is changed on an already running broker, metadata stored in the previously configured paths will not be available to Kafka once Kafka is restarted. This can lead to potential data loss.

Procedure

1. Select the Kafka service.
2. Go to Configuration and find the ZooKeeper Root property.
3. Add the path to use as a chroot environment for the Kafka cluster.

   Cloudera recommends that you use /kafka.
4. Enter a Reason for change and click Save Changes.
5. Restart the Kafka service.

Results

The Kafka ZooKeeper chroot is configured. Kafka uses the configured path to store its metadata in ZooKeeper.

Configure Kafka rack awareness

Learn how to configure rack awareness and multi-level rack awareness for Kafka brokers and clients.

Racks provide information about the physical location of a broker or a client. A Kafka deployment can be made rack aware by configuring rack awareness for the Kafka brokers and clients respectively. Enabling rack awareness can help in hardening your deployment, it provides durability guarantees for your Kafka service, and significantly decreases the chances of data loss.

The following tasks give step by step instructions on how you can configure and enable rack awareness and multi-level rack awareness for Kafka brokers and clients. The following covers configuration only. If you want to learn more about the rack awareness feature and how rack-aware Kafka deployments behave, see Kafka rack awareness.

Note: Although the feature is called rack awareness, the term rack does not necessarily mean an actual physical server rack. Instead, a rack from Kafka's perspective represents a physical location or independent physical infrastructure. For example, in many production deployments, the feature is used to specify the individual Data Centers that the brokers and clients are running in.

Configuring rack awareness for Kafka brokers

Learn how to configure rack awareness for Kafka brokers.

About this task

Rack awareness can be enabled and configured by specifying rack IDs. This can be done in two ways.

You can configure Kafka to use the rack information that is already set in Cloudera Manager on the level of the hosts. This is done by selecting the Enable Rack Awareness Kafka service property. In Cloudera Public Cloud, rack information in Cloudera Manager is set up automatically.

Alternatively, you can manually specify the rack ID for each broker role using the Broker Rack ID property. In this case, selecting Enable Rack Awareness is not required.

If IDs are specified in Broker Rack ID and Enable Rack Awareness is selected, the ID's specified in Broker Rack ID take precedence.

Important: If after configuring and enabling rack awareness you make changes to rack information (for example, change a rack name), ensure that you restart the Kafka and Cruise Control services. If the rack information is changed, the services become stale, Cloudera Manager, however, does not display the services as stale.

Before you begin

- In order for rack awareness to properly function, the brokers in your deployment must be spread across available racks. If all brokers are deployed on the same rack, enabling and configuring rack awareness will not provide you with any benefits.
- If you previously configured and enabled rack awareness by manually configuring the broker.rack property with Kafka Broker Advanced Configuration Snippet (Safety Valve), ensure that you remove all broker.rack entries from the advanced configuration snippet. The advanced configuration snippet takes precedence and overwrites the configuration set by both Enable Rack Awareness and Broker Rack ID.
- If you are using Broker Rack ID to set rack IDs, ensure that you set the property for each broker instance separately. Do not specify this property on a Kafka service level.

Procedure

1. In Cloudera Manager, select the Kafka service.

2. Configure rack IDs.

   For Use the IDs set on the host level

   a. Go to Configuration.
   b. Find and select the Enable Rack Awareness property.
   c. Click Save Changes.

   For Set IDs on a broker level

   Repeat the following steps for each Kafka broker role instance.

   a. Go to Instances.
   b. Click any of the broker role instances and go to Configuration.
   c. Find and configure the Broker Rack ID property.

      Add the ID of the rack (or other physical infrastructure) that the broker is deployed in. For example, rack1 or DC1
   d. Click Save Changes.

3. Restart the Kafka service.

Results

Rack awareness is enabled and configured for the Kafka brokers.

What to do next

Configure rack awareness for Kafka clients.

Configuring multi-level rack awareness for brokers

Learn how to enable and configure multi-level rack awareness for brokers.

About this task

Multi-level rack awareness can be enabled and configured by specifying multi-level rack IDs and selecting the Enable multi-level rack awareness Kafka service property.

A multi-level rack ID has a different format than a standard rack ID. It is an absolute path, it starts with a slash (/), but does not end in a slash. For example, /uswest/R0 or /DC1/Rack1/HostA. The ID can be specified in two ways.

You can configure Kafka to use the rack information that is already set in Cloudera Manager on the level of the hosts. This is done by selecting the Enable Rack Awareness Kafka service property. In Cloudera Public Cloud, rack information in Cloudera Manager is set up automatically.

Alternatively, you can manually specify the rack ID for each broker role using the Broker Rack ID property. In this case, selecting Enable Rack Awareness is not required.

If IDs are specified in Broker Rack ID and Enable Rack Awareness is selected, the ID's specified in Broker Rack ID take precedence.

AlI IDs you specify for your brokers must have an identical number of levels in the ID.

Important: If after configuring and enabling rack awareness you make changes to rack information (for example, change a rack name), ensure that you restart the Kafka and Cruise Control services. If the rack information is changed, the services become stale, Cloudera Manager, however, does not display the services as stale.

Before you begin

- In order for multi-level rack awareness to properly function, the topology of your deployment must be compatible. That is, the brokers in your deployment must be spread across available physical infrastructure (for example DCs and racks). If your deployment topology is not compatible, enabling and configuring multi-level rack awareness will not provide you with any benefits.
- If you previously configured and enabled rack awareness by manually configuring the broker.rack property with Kafka Broker Advanced Configuration Snippet (Safety Valve), ensure that you remove all broker.rack entries from the advanced configuration snippet. The advanced configuration snippet takes precedence and overwrites the configuration set by both Enable Rack Awareness and Broker Rack ID.
- If you are using Broker Rack ID to set rack IDs, ensure that you set the property for each broker instance separately. Do not specify this property on a Kafka service level.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Configure rack IDs.

   For Use the IDs set on the host level

   a. Go to Configuration.
   b. Find and select the Enable Rack Awareness property.
   c. Click Save Changes.

   For Set IDs on a broker level

   Repeat the following steps for each Kafka broker role instance.

   a. Go to Instances.
   b. Click any of the broker role instances and go to Configuration.
   c. Find and configure the Broker Rack ID property.

      Add the ID of the rack (or other physical infrastructure) that the broker is deployed in. Ensure that you set multi-level IDs. For example, /uswest/R0 or /DC1/Rack1/HostA.
   d. Click Save Changes.
3. Find and select the Enable multi-level rack awareness property.
4. Restart the Kafka service.

Results

Multi-level rack awareness is enabled for the broker. The brokers create replica assignments with multi-level rack awareness guarantees.

What to do next

Configure rack awareness for Kafka clients.

Configuring rack awareness for Kafka consumers

Learn how to make Kafka consumers rack aware by enabling and configuring follower fetching.

About this task

Kafka Consumers can be made rack aware enabling follower fetching for your Kafka deployment. Follower fetching can be enabled by configuring the replica.selector.class property for the broker and configuring the client.rack property in the consumer's configuration. The replica.selector.class property is not directly available for configuration in Cloudera Manager and you must use an advanced security snippet to configure it.

Before you begin

Ensure that brokers have rack awareness enabled. For more information, see Configuring rack awareness for Kafka brokers.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find the Kafka Broker Advanced Configuration Snippet (Safety Valve) for kafka.properties property.
4. Add the following configuration entry to the advanced configuration snippet.

```
replica.selector.class=org.apache.kafka.common.replica.RackAwareReplicaS
elector
```

5. Click Save Changes.
6. Restart the Kafka service.
7. Add the following to your consumer configuration.

```
client.rack=[***RACK ID***]
```

   Replace *[***RACK ID***]* with the ID of the rack that the consumer is running in. The rack ID should match one of the rack ID's you configured for the brokers. Ensure that you configure each consumer and add its corresponding rack ID. If the consumer is deployed in a rack with no brokers, specify the rack ID of a broker that is closest to the rack that the consumer is running in.

Results

Follower fetching is enabled for the Kafka deployment. Kafka consumers are now rack aware and attempt to consume from the replica that is in the closet rack instead of consuming from the replica leader.

Configuring multi-level rack awareness for consumers

Learn how to configure consumer follower fetching in Kafka deployment that has multi-level rack awareness enabled.

About this task

If multi-level rack awareness is enabled for the brokers, you can specify a multi-level rack ID for the consumers in their configuration. If a multi-level rack ID is specified for the consumer, the consumer will fetch messages from the closest replica in the multi-level hierarchy based on the ID it is provided. The rack ID can be set with the client.rack property in the consumer's configuration.

Note: Unlike standard rack awareness and follower fetching, manually configuring a replica selector class in the broker's configuration is not required. If multi-level rack awareness is enabled for the broker, the required ReplicaSelector class is automatically installed.

Before you begin

Ensure that brokers have multi-level rack awareness enabled. For more information, see Configuring multi-level rack awareness for brokers.

Procedure

1. Choose an appropriate multi-level rack ID for the consumer.

   A multi-level rack ID is an absolute path, it starts with a slash (/) but does not end in a slash. The exact ID that you specify depends on the levels of hierarchy in your deployment (cluster topology) and the rack IDs specified for the broker.

   With standard rack awareness and follower fetching, the IDs you specify for the consumer must exactly match the ID specified for the brokers. With multi-level rack awareness, this is not the case. The levels of hierarchy represented in the ID of the consumer can differ compared to the broker rack IDs. Only a partial match is required to find a closest replica.

   For example, assume you have a two level hierarchy, DCs on the top level, racks on the second level. Additionally, assume there are two replicas. They are located in /DC1/R1 and /DC2/R3. Depending on the configuration of client.rack, the consumer will select different replicas. The behavior is as follows:

   - If client.rack is /DC1, the /DC1/R1 replica is selected.
   - If client.rack is /DC2/R5, the /DC2/R3 replica is selected.
   - If client.rack is /DC2/R3, the /DC2/R3 replica is selected.
   - If client.rack is not set, the leader is selected.
   - If client.rack is random_id, the leader is selected.
   - If client.rack is DC3, the leader is selected.
2. Set the client.rack property.

   For example:

   ```
   client.rack=/DC1/R1
   ```

Results

Consumer IDs are configured. The consumers will fetch data from the closest replica in the multi-level deployment.

Configuring rack awareness for Kafka producers

Learn how to enable and configure rack awareness for Kafka producers.

About this task

Enabling rack awareness for Kafka producers involves configuring your Kafka deployment in a way that ensures that producers commit messages to at least two separate brokers that are deployed on different racks. This can be done by configuring your producers to provide the highest available guarantee on message delivery and configuring min.insy nc.replicas for your topics.

Before you begin

Ensure that brokers have rack awareness enabled. For more information, see Configuring rack awareness for Kafka brokers.

Procedure

1. Add the following to your producer configuration.

   ```
   acks=all
   ```

   This is the default configuration for producer version 3.0.0 or later. As a result, configuring this property might not be required.
2. Configure min.insync.replicas for the produced topics to a value that ensures the desired number of racks (minimum of 2) get the message before the produce request is considered successful.

Results

Rack awareness for Kafka producers is configured. Producers will now ensure that messages are produced to at least 2 (or more) of the available racks.

Cloudera Runtime 7.3.1

Kafka Connect

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Kafka Connect Overview.....5

Setting up Kafka Connect in Cloudera Public Cloud.....5

Using Kafka Connect..... 6
Configuring the Kafka Connect Role..... 7
Managing, Deploying and Monitoring Connectors..... 7
Writing Kafka data to Ozone with Kafka Connect..... 8
Writing data to Ozone in an unsecured cluster with Kafka Connect.....8
Writing data to Ozone in a Kerberos and TLS/SSL enabled cluster with Kafka Connect.....10
Using the AvroConverter..... 12
Enabling and configuring the AvroConverter.....14
Configuring AvroConverter for Debezium connectors with Schema Registry integration.....16
Reading Debezium Avro records from Kafka..... 17
Configuring EOS for source connectors..... 18

Securing Kafka Connect..... 19
Kafka Connect to Kafka broker security..... 20
Configuring TLS/SSL encryption for the Kafka Connect role.....20
Configuring Kerberos authentication for the Kafka Connect role.....20
Kafka Connect REST API security.....21
Kafka Connect REST API authentication.....21
Kafka Connect Rest API authorization.....22
Kafka Connect connector configuration security.....24
Kafka Connect Secrets Storage.....24
Configuring connector JAAS configuration and Kerberos principal overrides..... 32
Configuring a Nexus repository allow list.....33

Single Message Transforms.....33
Configuring an SMT chain.....34
ConvertFromBytes..... 36
ConvertToBytes.....38

Connectors..... 40
Installing Kafka Connect connectors..... 43
Installing custom developed Kafka Connect connectors..... 44
Installing FileStream connectors..... 44
Setting up the Debezium Db2 Source connector.....47
Setting up the Debezium MySQL Source connector.....47
Setting up the Debezium Oracle Source connector..... 48
Setting up the Debezium PostgreSQL Source connector..... 49
Setting up the Debezium SQL Server Source connector.....50
HTTP Source connector..... 51
JDBC Source connector..... 53

JMS Source connector....56
MQTT Source connector....58
SFTP Source connector....59
Stateless NiFi Source and Sink....62
Dataflow development best practices for Stateless NiFi....63
Kafka Connect worker assignment and Stateless NiFi....70
Kafka Connect log files and Stateless NiFi....73
Kafka Connect tasks and Stateless NiFi....73
Developing a dataflow for Stateless NiFi....76
Deploying a dataflow using Stateless NiFi....80
Downloading and viewing the predefined Stateless NiFi dataflows shipped in Cloudera....84
Configuring flow.snapshot for Stateless NiFi connectors....85
Tutorial: developing and deploying a JDBC Source dataflow in Kafka Connect using Stateless NiFi....87
Syslog TCP Source connector....97
Syslog UDP Source connector....100
ADLS Sink connector....102
Amazon S3 Sink....105
Configuration example....106
HDFS Sink....107
Configuration example for writing data to HDFS....107
Configuration example for writing data to Ozone FS....109
HDFS Stateless Sink connector....110
HTTP Sink connector....116
InfluxDB Sink connector....118
JDBC Sink connector....120
Kudu Sink connector....122
S3 Sink connector....124

Kafka Connect Overview

Get started with Kafka Connect.

Kafka Connect is a tool for streaming data between Apache Kafka and other systems in a reliable and scalable fashion. Kafka Connect makes it simple to quickly define connectors that move large collections of data into and out of Kafka. Source connectors can ingest entire databases or collect metrics from all your application servers into Kafka topics, making the data available for stream processing with low latency. Sink connectors can deliver data from Kafka topics into secondary storage and query systems or into batch systems for offline analysis.

Kafka Connect in Cloudera is shipped with many differentCloudera-developed as well as publicly available sink and source connectors. Each of which cover a specific use case for streaming data. In addition to the connectors available by default, installing custom developed or third-party connectors is also possible. All connectors can be deployed, managed, and monitored using the Streams Messaging Manager UI (recommended), Streams Messaging Manager REST API, or Kafka Connect REST API.

Related Information

Setting up Kafka Connect in Cloudera Public Cloud
Using Kafka Connect
Securing Kafka Connect
Single Message Transforms
Connectors
Streams Messaging Reference

Setting up Kafka Connect in Cloudera Public Cloud

Learn how you can set up Kafka Connect in Cloudera Public Cloud with the Streams Messaging cluster templates.

Kafka Connect in Cloudera is provided in the form of a Kafka service role. The role is called Kafka Connect. In Cloudera Public Cloud, Kafka Connect can be deployed with the Streams Messaging Light Duty, Heavy Duty, and High Availability templates available in the Cloudera Data Hub service. However, there are some differences between how Kafka Connect is deployed with each of the templates.

Light Duty

In the Streams Messaging Light Duty template, Kafka Connect roles share a host group and nodes with other service roles. Specifically, Kafka Connect is deployed in the Broker and Core_broker host groups and shares nodes with Kafka broker, ZooKeeper, and Streams Replication Manager Driver roles. The Core_broker is a mandatory host group in Light Duty clusters and has an instance (node) count of three. This means that by default, all Light Duty cluster you deploy will include a minimum of three Kafka Connect roles. If required, Kafka Connect can also be scaled in Light Duty clusters by scaling the Broker host group.

Heavy Duty and High Availability

In the Heavy Duty and High Availability templates, Kafka Connect roles are deployed in a dedicated host group named Connect. The instance count of this host group is set to zero by default. This means that, unless specifically provisioned, Kafka Connect roles are **not** deployed in Heavy Duty or High Availability clusters by default. The instance (node) count of the Connect host group can be configured in Cloudera Data Hub during cluster provisioning. Additionally, you can add Kafka Connect to already running Heavy Duty or High Availability cluster by scaling the Connect host group.

For more information regarding the Streams Messaging cluster templates, scaling, and cluster deployment with Cloudera Data Hub, see the *Related Information* section below.

Streams Messaging Manager integration

Kafka Connect deployed in Cloudera Public Cloud is by default configured to integrate with the Streams Messaging Manager instance running in the cluster. This means that, if you provisioned your cluster with Kafka Connect, you will be able to manage, monitor, and deploy Kafka Connect connectors with Streams Messaging Manager as soon as the cluster is deployed with no additional configuration required. Kafka Connect can be managed in Streams Messaging Manager on the Connect page. For more information, see *Monitoring Kafka Connect using Streams Messaging Manager*.

Related Information

Setting up your Streams Messaging cluster

Streams Messaging cluster layout

Scaling Kafka Connect

Monitoring Kafka Connect using Streams Messaging Manager

Using Kafka Connect

Learn how you can manage, monitor and configure Kafka Connect.

There are multiple ways that you can manage, monitor and configure Kafka Connect in Cloudera. It is important to make a distinction between the Kafka Connect roles and Kafka Connect connectors as they are managed through different interfaces.

The Kafka Connect Role

Kafka Connect roles are roles that you can deploy within a Kafka service. They represent Kafka Connect workers. Because these roles are deployed with and managed by Cloudera Manager, their management and configuration is done through Cloudera Manager. In other words the Kafka Connect workers you deploy are managed and configured in Cloudera Manager. For more information see *Configuring Kafka Connect* and the Cloudera Manager documentation.

Kafka Connect Connectors

Connectors are not managed by Cloudera Manager, instead multiple other interfaces can be used to interact with them. These are the following:

- Streams Messaging Manager UI

  The Streams Messaging Manager UI is a recommended interface in Cloudera to manage connectors. Comprehensive documentation is provided on the usage of the UI in *Monitoring Kafka Connect*.
- The Streams Messaging Manager REST API

  The Streams Messaging Manager REST API is a recommended interface in Cloudera to manage connectors. For more information, see the Streams Messaging Manager REST API Reference.

  Note: Both the Streams Messaging Manager UI and Streams Messaging Manager REST API support the same management actions.
- The Kafka Connect REST API

  The Kafka connect REST API can be used to manage connectors. However, its usage is not recommended by Cloudera, nor is separate documentation provided. For more information, see the upstream Apache Kafka documentation.

Related Information

Managing and monitoring Kafka Connect using Streams Messaging Manager

Streams Messaging Manager REST API Reference

Apache Kafka Documentation

Configuring the Kafka Connect Role

Learn more about how you can configure the Kafka Connect role as well as its notable properties.

The Kafka Connect role is deployed with and managed by Cloudera Manager. You can view a list of configuration properties as well as configure them by going to Kafka service Configuration and selecting the Kafka Connect filter in the Filters pane. Most of the properties available are the standard worker properties defined in upstream Apache Kafka. However, there are a number of notable properties that you should be familiar with when using Kafka Connect in Cloudera. The notable properties are the following:

Broker List for Kafka Connect

These are the brokers Kafka Connect should connect to. You specify the brokers by adding IP:port or hostname:port pairs. The brokers you specify here should be the brokers that are running in the same cluster that Kafka Connect is deployed on. While you can specify a single broker, Cloudera recommends that you specify multiple for high availability.

group.id

The group ID is a unique string that Kafka Connect roles use to form a cluster of connect workers. This property should always be set to the same string for all Kafka Connect roles that are deployed in the same cluster. Kafka Connect will not function properly If there is a mismatch between the group IDs. Therefore, Cloudera recommends that you use the default values for all roles that you deploy.

plugin.path

The directory where the Kafka Connect connector plugins are stored. This is the directory where JAR files for custom connectors can be placed. Cloudera recommends that you use the default path which is /var/lib/kafka.

kafka.connect.rest.port and kafka.connect.secure.rest.port

These are the ports that Kafka Connect API will accept requests on. If the Kafka Connect role is TLS/SSL enabled, it will use the port specified in kafka.connect.secure.rest.port, if TLS/SSL is not enabled, it will use the port specified in kafka.connect.rest.port. Take note of the ports as they are needed when you set up Streams Messaging Manager to manage and monitor Kafka Connect.

A comprehensive list of all properties available for the Kafka Connect role is available in Kafka Properties in Cloudera Runtime.

Related Information

Kafka Properties in Cloudera Runtime

Managing, Deploying and Monitoring Connectors

Learn more about managing connectors.

You can manage, monitor, deploy and interact with Kafka Connect and Kafka Connect connectors either though the Streams Messaging Manager UI or Streams Messaging Manager REST API. For more information see Monitoring Kafka Connect, as well as the Streams Messaging Manager REST API Reference.

Related Information

Managing and monitoring Kafka Connect using Streams Messaging Manager

Streams Messaging Manager REST API Reference

Writing Kafka data to Ozone with Kafka Connect

You can use the HDFS Sink Connector developed by Cloudera and shipped with Cloudera Runtime to write Kafka data to the Ozone filesystem. This can be done on both secure and unsecure clusters by deploying and configuring a new connector with the Streams Messaging Manager UI.

Writing data to Ozone in an unsecured cluster with Kafka Connect

You can use the HDFS Sink Connector developed by Cloudera, i to write Kafka topic data to Ozone in an unsecure cluster. Connector deployment and configuration is done using the Streams Messaging Manager UI.

About this task

The following steps walk you through how the Cloudera-developed HDFS Sink Connector can be set up and deployed using the Streams Messaging Manager UI.

In addition to the connector setup, these steps also describe how you can create a test topic and populate it with data using Kafka command line tools. If you already have a topic that is ready for use and do not want to create a test topic, you can skip steps 1 through 3. These steps deal with topic creation, message consumption, and message production. These steps are optional.

Before you begin

An unsecure Cloudera Private Cloud Base cluster with Kafka, Streams Messaging Manager, and Ozone is set up and configured.

Procedure

1. Create a Kafka topic:
   a) SSH into one of the hosts in your cluster.

   ```
   ssh [***USER***]@[***MY-CLUSTER-HOST.COM***]
   ```

   b) Create a topic with the kafka-topics tool.

   ```
   kafka-topics --create --bootstrap-server [***MY-CLUSTER-
   HOST.COM:9092***] --replication-factor 1 --partitions 1 --top
   ic [***TOPIC***]
   ```

   If an out of memory exception is displayed while running this command, increase the JVM heap with the following command and try again:

   ```
   export KAFKA_OPTS="-Xmx1g -Xms1g"
   ```

   c) Verify that the topic is created.

   ```
   kafka-topics --list --bootstrap-server [***MY-CLUSTER-HOST.COM:9092***]
   ```

2. Produce messages to your topic with the kafka-console-producer.

   ```
   kafka-console-producer --broker-list [***MY-CLUSTER-HOST.COM:9092***] --t
   opic [***TOPIC***]
   ```

   Enter test messages once the tool is running.

   ```
   >MY FIRST MESSAGE
   >MY SECOND MESSAGE
   ```

3. Consume messages:
   a) Open a new terminal session and log in to one of the hosts in your cluster.
   b) Consume messages with the kafka-console-consumer.

```
kafka-console-consumer --from-beginning --bootstrap-server [***MY-CLUSTER-HOST.COM:9092***] --topic [***TOPIC***]
```

   The messages you produced with kafka-console-producer appear. In addition, you can switch to the terminal session that is running kafka-console-producer and produce additional messages. These new messages appear in real time in the session running kafka-console-consumer.

4. Deploy and configure an HDFS Sink Connector:
   a) In Cloudera Manager, select the Streams Messaging Manager service.
   b) Click Streams Messaging Manager Web UI.
   c) Click the Connect option in the left-side menu.
   d) Click + New Connector to add a new connector.
   e) Go to the Sink Connectors tab and select the HDFS Sink Connector.

      On the UI the HDFS Sink Connector is represented by its class name, which is com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConnector.
   f) Enter a name for the connector.
   g) Configure the connector.

      Use the following example as a template:

```
{
 "connector.class": "com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConnector",
 "hdfs.uri": "ofs://ozone1/vol1/bucket1/",
 "hdfs.output": "/topics_output/",
 "tasks.max": "1",
 "topics": "testTopic",
 "hadoop.conf.path": "file:///etc/hadoop/conf",
 "output.writer": "com.cloudera.dim.kafka.connect.partition.writers.txt.TxtPartitionWriter",
 "value.converter": "org.apache.kafka.connect.storage.StringConverter",
 "output.storage": "com.cloudera.dim.kafka.connect.hdfs.HdfsPartitionStorage",
 "hdfs.kerberos.authentication": "false"
}
```

      Ensure that you replace the values of hdfs.uri and hdfs.output with valid Ozone paths. The template gives an example of how these paths should look like. Replace any other values depending on your cluster configuration and requirements.
   h) Click Validate.

      The validator displays any JSON errors in your configuration. Fix any errors that are displayed. If your JSON is valid, the JSON is valid message is displayed in the validator.
   i) Click Next.
   j) Review your connector configuration.
   k) Click Deploy .

5. Verify that connector deployment is successful:
   a) In the Streams Messaging Manager UI, click the Connect option in the left-side menu.
   b) Click on either the topic or the connector you created.

      If connector deployment is successful, a flow is displayed between the topic you specified and the connector you created.

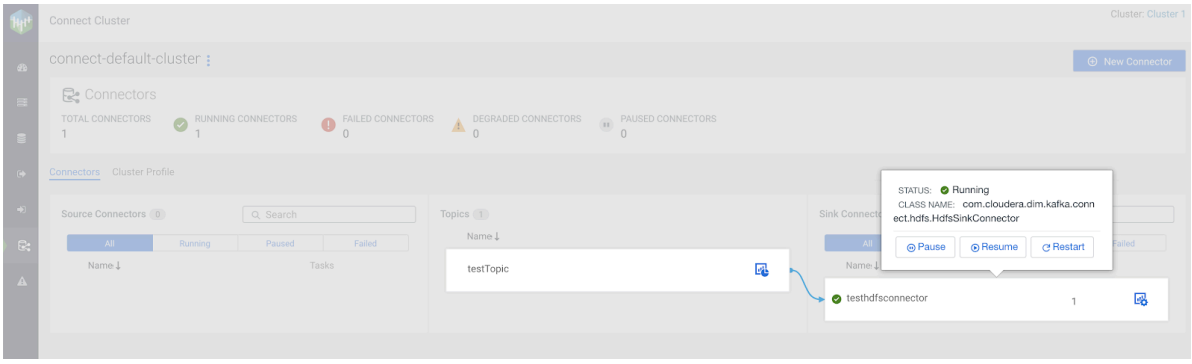

6. Verify that topic data is written to Ozone.

   You can do this by listing the files under the ofs:// location you specified in the connector configuration.

Writing data to Ozone in a Kerberos and TLS/SSL enabled cluster with Kafka Connect

You can use the HDFS Sink Connector developed by Cloudera to write Kafka topic data to Ozone in a secure cluster. Connector deployment and configuration is done using the Streams Messaging Manager UI.

About this task

The following steps walk you through how the Cloudera-developed HDFS Sink Connector can be set up and deployed using the Streams Messaging Manager UI.

In addition to the connector setup, these steps also describe how you can create a test topic and populate it with data using Kafka command line tools. If you already have a topic that is ready for use and do not want to create a test topic, you can skip steps 1 through 3. These steps deal with topic creation, message consumption, and message production. They are not necessary to carry out.

Before you begin

- Ensue that a Kerberos and TLS/SSL enabled Cloudera Private Cloud Base cluster with Kafka, Streams Messaging Manager, and Ozone is set up and configured.
- To create a test topic with the Kafka console tools, you must ensure that a .properties client configuration file is available for use.

  You can create one using the following example as a template:

```
sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required u
seKeyTab=true keyTab="[***PATH TO KEYTAB FILE***]" principal="[***KERBEROS
 PRINCIPAL***]";
security.protocol=SASL_SSL
sasl.mechanism=GSSAPI
sasl.kerberos.service.name=kafka
ssl.truststore.location=[***TRUSTSTORE LOCATION***]
```

Procedure

1. Create a Kafka topic:
   a) SSH into one of the hosts in your cluster.

   ```
   ssh [***USER***]@[***MY-CLUSTER-HOST.COM***]
   ```

   b) Create a topic with the kafka-topics tool.

   ```
   kafka-topics --create --bootstrap-server [***MY-CLUSTER-
   HOST.COM:9093***] --replication-factor 1 --partitions 1 --topic [***T
   OPIC***] --command-config [***CLIENT CONFIG FILE***]
   ```

   If an out of memory exception is thrown, increase the JVM heap with the following command and try again:

   ```
   export KAFKA_OPTS="-Xmx1g -Xms1g"
   ```

   c) Verify that the topic is created.

   ```
   kafka-topics --list --bootstrap-server [***MY-CLUSTER-HOST.COM:9093***]
    --command-config [***CLIENT CONFIG FILE***]
   ```

2. Produce messages to your topic with the kafka-console-producer.

   ```
   kafka-console-producer --broker-list [***MY-CLUSTER-HOST.COM:9093***] --t
   opic [***TOPIC***] --producer.config [***CLIENT CONFIG FILE***]
   ```

   Enter test messages once the tool is running.

   ```
   >MY FIRST MESSAGE
   >MY SECOND MESSAGE
   ```

3. Consume messages:
   a) Open a new terminal session and log in to one of the hosts in your cluster.
   b) Consume messages with the kafka-console-consumer.

   ```
   kafka-console-consumer --from-beginning --bootstrap-server [***MY-
   CLUSTER-HOST.COM:9093***] --topic [***TOPIC***] --consumer.conf
   ig [***CLIENT CONFIG FILE***]
   ```

   The messages you produced with kafka-console-producer appear. You can switch to the terminal session that is running kafka-console-producer and produce additional messages. These new messages appear in real time in the session running kafka-console-consumer.

4. Deploy and configure an HDFS Sink Connector:
   a) In Cloudera Manager, select the Streams Messaging Manager service.
   b) Click Streams Messaging Manager Web UI to log in to the UI.

   If prompted, enter a valid username as well as a password and click SIGN IN.
   c) Click the Connect option in the left-side menu.
   d) Click + New Connector to add a new connector.
   e) Go to the Sink Connectors tab and select the HDFS Sink Connector.

   On the UI the HDFS Sink Connector is represented by its class name, which is com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConnector.
   f) Enter a name for the connector.
   g) Configure the connector.

   Use the following example as a template:

   ```
   {
   ```

```
 "connector.class": "com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConne
ctor",
 "hdfs.uri": "ofs://ozone1/vol1/bucket1/",
 "tasks.max": "1",
 "topics": "testTopic",
 "hdfs.kerberos.authentication": "true",
 "hdfs.kerberos.user.principal": "${cm-agent:ENV:kafka_connect_service
_principal}",
 "hdfs.kerberos.keytab.path": "${cm-agent:keytab}",
 "hdfs.kerberos.namenode.principal": "hdfs/_HOST@REALM",
 "hdfs.output": "/topics_output/",
 "hadoop.conf.path": "file:///etc/hadoop/conf",
 "output.writer": "com.cloudera.dim.kafka.connect.partition.writers.tx
t.TxtPartitionWriter",
 "value.converter": "org.apache.kafka.connect.storage.StringConverter",
 "output.storage": "com.cloudera.dim.kafka.connect.hdfs.HdfsPartition
Storage"
}
```

Ensure that you replace the values of hdfs.uri and hdfs.output with valid Ozone paths. The template gives an example of how these paths should look like. Replace other values depending on your cluster and requirements.

h) Click Validate.

The validator displays any JSON errors in your configuration. Fix any errors that are displayed. If your JSON is valid, the JSON is valid message is displayed.

i) Click Next.

j) Review your connector configuration.

k) Click Deploy to deploy the connector.

5. Verify that connector deployment is successful:

a) In the Streams Messaging Manager UI, click the Connect option in the left-side menu.

b) Click on either the topic or the connector you created.

If connector deployment is successful, a flow is displayed between the topic you specified and the connector you created.

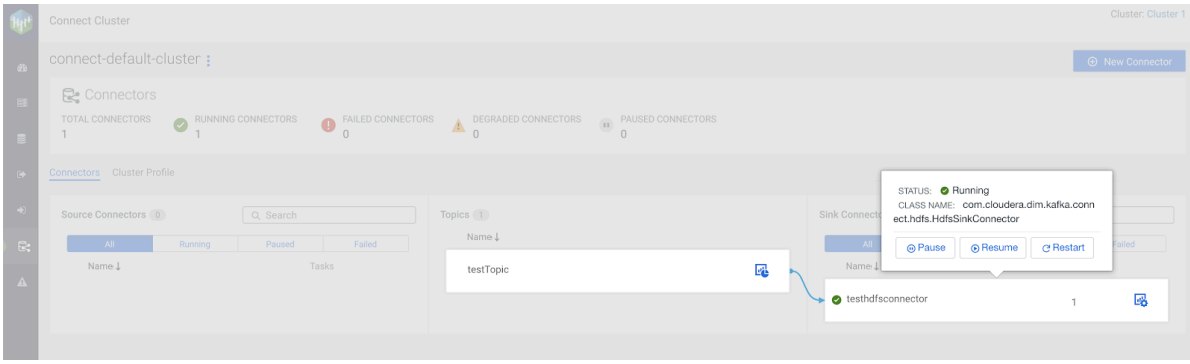

6. Verify that topic data is written to Ozone.

You can do this by listing the files under the ofs:// location you specified in the connector configuration.

Using the AvroConverter

The AvroConverter is a Kafka Connect converter shipped with Cloudera Runtime that enables Kafka Connect connectors to serialize or deserialize Kafka messages, consisting of key and value pairs, from or to Avro. Using the Avro format can make your messages more compact compared to using for example JSON format. AvroConverter is able to convert from Avro format to Kafka Connect internal data format, and from Kafka Connect internal data to Avro format, and is able to handle Avro schemas integrated with Schema Registry automatically.

Using AvroConverter with a source connector

When using the AvroConverter with a Kafka Connect source connector, the source connector fetches the data from an external system. The fetched data is converted into Kafka Connect internal data format. If Single Message Transforms (SMT) are configured, the messages are converted based on the SMT conditions as well. If the AvroConverter is enabled, the messages are converted from the internal data format to Avro. At this step, the AvroConverter automatically also creates a new schema version in Schema Registry for the Avro data, if the schema has not been already created. In case the schema already exists in Schema Registry, schema creation is skipped and the existing ID will be assigned to the payload during the serialization. The converted data in Avro is written to the Kafka cluster with its corresponding schema ID from Schema Registry.

The following illustration shows the steps how AvroConverter works with a Kafka Connect source connector:

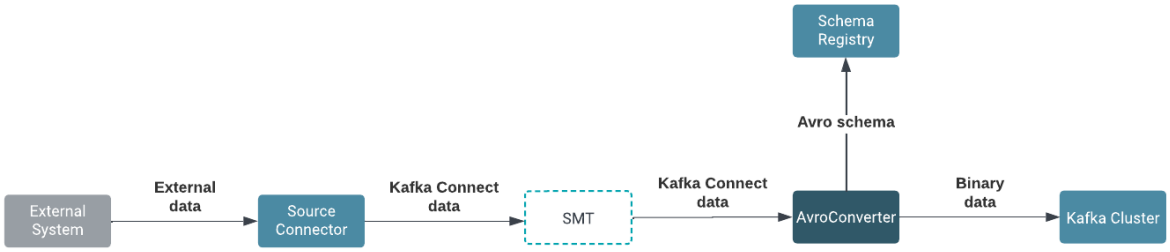

Using AvroConverter with a sink connector

When using the AvroConverter with a Kafka Connect sink connector, the data is fetched from the Kafka cluster, which includes its schema ID from Schema Registry. The AvroConverver extracts the schema ID and gets the corresponding schema from Schema Registry. The data is converted from Avro format to the Kafka Connect internal data format. If SMT are configured, the messages are converted based on the SMT conditions as well. The converted data is written by the Kafka Connect sink connector to the external system.

The following illustration shows the steps how AvroConverter works with a Kafka Connect sink connector:

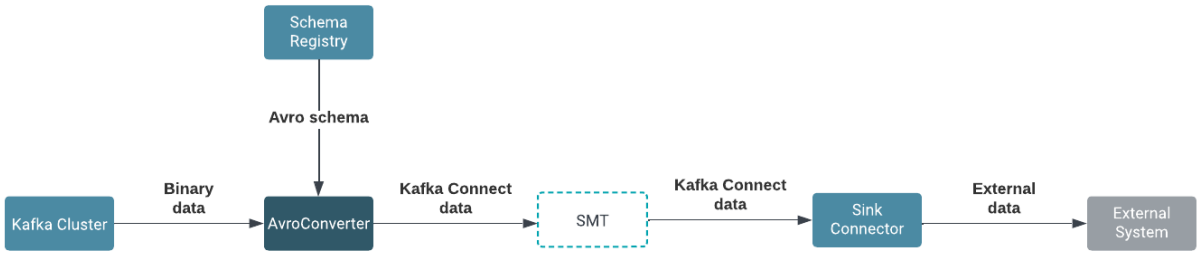

Note: If the SMTs contain steps to change the schema, the schema will not be updated in Schema Registry. The schema is handled automatically by the AvroConverter before any SMTs are applied.

Mapping logical data types

AvroConverter is capable of mapping the data from the Kafka Connect internal data format to Avro. The mapping happens automatically during the conversion, which means that no property needs be configured. The AvroConverter also handles Kafka Connect specific logical types and maps them to Avro ones and vice versa.

The corresponding logical types for Kafka Connect and Avro with their direction is shown in the following table:

Table 1: Logical Type Mapping

Kafka Connect Logical Type	Direction	Avro Logical Type
Decimal	##	Decimal
Date	##	Date
Time	##	TimeMillis
Time	#	TimeMicros
Timestamp	##	TimestampMillis
Timestamp	#	TimestampMicros

Handling passthrough

When passthrough is enabled, messages are not converted between the Avro format and the Kafka Connect internal data format. In this case, the data is passed through unchanged to the Kafka Connect sink connector. If the passthrough is disabled, messages are converted from Avro to the Kafka Connect internal data format, which can make your input and output formats to be different.

The following illustration shows what happens when passthrough is enabled in the AvroConverter for a sink connector:

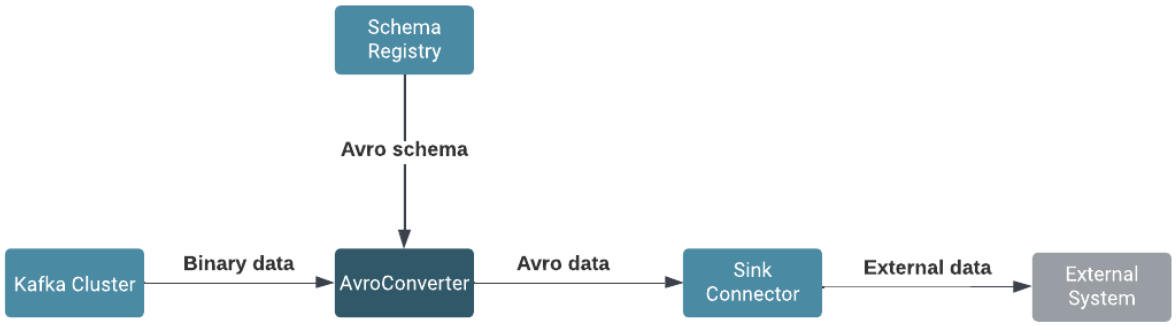

Enabling passthrough is recommended when both the input and output formats of your messages are Avro, and it can save any performance overhead generated by the conversion process. You need to ensure that the passthrough is only enabled if the connector can handle Avro objects. Cloudera provides the following connectors that are capable of handling passthrough:

- HDFS Sink connector
- S3 Sink connector

Any other third party connectors can make use of the passthrough if the connectors are implemented to handle the passthrough of the message.

Warning: Passthrough is enabled by default in AvroConverter. You need to make sure that the connector is able to handle passthrough before using AvroConverter or you must disable the passthrough. In case the connector cannot handle the passthrough, the connector will fail.

You can enable and disable passthrough by setting the passthrough.enabled property to true or false when configuring the connector. For more information, see the *KafkaAvroSerializer properties reference* documentation under the *Related information*

Enabling and configuring the AvroConverter

When the AvroConverter is enabled as a key or value converter, you also need to provide further configuration properties that affect the behavior of the AvroConverter. All configuration properties must be prefixed with either the

value.converter. or key.converter. prefixes. These prefixes control whether a specific property is applied to the key or value converter.

Enabling AvroConverter

You can enable AvroConverter for any connector using the key.converter and value.converter properties to the connector configuration. The value of these properties must be set to the class name of the AvroConverter.

For example, if you want to use the AvroConverter for both key and value conversion, you must add the following properties to the connector configuration:

```
"key.converter": "com.cloudera.dim.kafka.connect.converts.AvroConverter",
"value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConverter",
```

Conversion is enabled separately for key or value conversion, which enables you to use different converters for the key and value. For example, if you want to use the AvroConverter to convert values, but use a different converter, such as the StringConverter, to convert keys, you must add the following properties to the connector configuration:

```
"key.converter": "org.apache.kafka.connect.storage.StringConverter",
"value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConverter",
```

Configuring passthrough

The passthrough configuration in AvroConverter controls whether or not the data is converted into the Kafka Connect internal data format before it is written into an output file or if data is passed through the connector without conversion. Passthrough is enabled by the passthrough.enabled property, which is set to true by default.

The passthrough can be set with different prefixes for key and value converters as shown in the following example configuration:

```
"key.converter.passthrough.enabled":  "false",
"value.converter.passthrough.enabled":  "false",
```

Warning:

Passthrough is enabled by default in AvroConverter. You need to make sure that the connector is able to handle passthrough before using AvroConverter or you must disable the passthrough. In case the connector cannot handle the passthrough, the connector will fail. For more information, see the *Handling passthrough* section.

Enabling SerDes

To transform messages, the AvroConverter uses an embedded KafkaAvroSerializer and KafkaAvroDeserializer, which need to be configured with the AvroConverter to enable further control over the behavior.

You must provide the URL of the Schema Registry server to be able to use the KafkaAvroSerializer and KafkaAvroDeserializer, because both the serializer and the deserializer contain an embedded Schema Registry client. You can specify the Schema Registry server URL with the schema.registry.url property as shown in the following example:

```
"key.converter.schema.registry.url":  "http://registryserver.example.com:9090/api/v1"
"value.converter.schema.registry.url":  "http://registryserver.example.com:9090/api/v1"
```

You can configure additional properties related to serialization, deserialization and the Schema Registry client using the property reference documentations under the *Related information*.

Configuration example

The following example configuration shows a connector configuration with AvroConverter set as key and value converter with connection to a remote Schema Registry server.

```
{
"value.converter":  "com.cloudera.dim.kafka.connect.converts.AvroConverter
", // use AvroConverter for the conversion of the Kafka message value
"value.converter.passthrough.enabled":  "false", // disable passthrough
"value.converter.schema.registry.url": "http://registryserver.example.com:
9090/api/v1", // URL of the schema registry server that the Schema Registry
client inside the value converter will connect to
"value.converter.schema.registry.auth.username = [***USERNAME***],  // us
ername to be used for authentication by the value converter's embedded Schem
a Registry client
"value.converter.schema.registry.auth.password = [***PASSWORD***], //  pa
ssword to be used for authentication by the value converter's embedded Schem
a Registry client
"key.converter":  "com.cloudera.dim.kafka.connect.converts.AvroConverter",
"key.converter.schema.registry.auth.username = [***USERNAME***],
"key.converter.schema.registry.auth.password = [***PASSWORD***],
"key.converter.passthrough.enabled":  "false",
"key.converter.schema.registry.url":  "HTTP://
REGISTRYSERVER.EXAMPLE.COM:9090/API/V1"
}
```

Configuring AvroConverter for Debezium connectors with Schema Registry integration

Learn about how to configure the AvroConverter for Debezium connector with Schema Registry integration.

When you want to use the AvroConverter for the Debezium connectors, you need to configure the passthrough, schema compatibility, schema name value and key prefix properties.

passthrough.enabled

You must disable the passthrough to have automatically created schemas for CDC records in Kafka as shown in the following example:

```
"value.converter.passthrough.enabled": "false"
"key.converter.passthrough.enabled": "false"
```

This will ensure that the CDC event fields are loaded into the internal schema of Kafka Connect, from which the field will be converted into Avro format by the AvroConverter before writing it to Kafka. When doing this conversion, the schemas are automatically created in Schema Registry as well.

schema.compatibility

Whenever a schema is created in Schema Registry, compatibility is also set to it by default. Based on the compatibility, if the schema evolves, the new version will be validated against old ones. For more information about the compatibility in Schema Registry, see the *Compatibility policies* documentation under the *Related information*.

The compatibility must be set before the first CDC event is processed by the connector, because the first schema will be created based on the compatibility policies. If backward incompatible changes could happen in the database, consider setting the parameters to NONE, because in this case the Avro schemas will also be incompatible.

The default compatibility can be set based on the following example:

```
“value.converter.schema.compatibility”: “NONE”
“key.converter.schema.compatibility”: “NONE”
```

Note:

The connector task will stop and fail if compatibility issues happen during schema evolution. In this case manual intervention is needed to continue the task.

schema.name.key.suffix and schema.name.value.suffix

When a new schema is created, the name of the schema consists of the name of the topic and an appended value, if set. However, it is possible to set additional suffixes for schema names. The default suffix is empty for the values, but :k is added as a prefix for keys. Key and value suffixes can be set on converters according the following example:

```
"value.converter.schema.name.value.suffix", "value_suffix"
"key.converter.schema.name.key.suffix", "key_suffix"
```

Additional configuration properties for Debezium connectors and Schema Registry integration with AvroConverter

The bigint.unsigned.handling.mode in a Debezium connector handles how bigint data types are mapped, which have the following values that can affect the behavior of the logical data type conversion:

precise

The bigint unsigned values will be stored as byte arrays in Avro with decimal logical type. You can get a byte array or a BigDecimal automatically when you would like to read this data. The object type you get depends on if logical type conversion is enabled or not.

long

Whatever is stored in the database, it will be stored in Avro as long, and you get a Java long object in Java Virtual Machine (JVM). In this case, precision of the mapping is smaller, but no need for logical type handling.

Check schema.name.adjustment.mode and field.name.adjustment.mode to ensure that there are no conflicts between your database related names and the Avro specific naming convention.

Check decimal.handling.mode to decide how precise numbers you need in different cases, and if you need to use logical type for higher precision

Check time.precision.mode to decide how precise dates and times you need in different cases, and if you need to use logical type for higher precision.

Reading Debezium Avro records from Kafka

Learn how to read Debezium CDC Avro records from Kafka using the AvroConverter functions.

Avro records can be consumed from Kafka with the Schema Registry integration as described in the *Integrating Kafka with Schema Registry* documentation.

If any of your connectors produce Avro records with logical types, you do not need to write the conversion from raw Avro data to proper Java objects as you can enable automatic conversion. To enable automatic logical type conversion, you need to set the logical.type.conversion.enabled property in the consumer configuration as shown with the following in the following example:

```
“logical.type.conversion.enabled”: “true”
```

This conversion is handled by the internal SerDes classes in KafkaAvroSerializer and KafkaAvroDeserializer.

Logical type conversion automatically happens in Debezium connectors when they produce the Avro events to Kafka. Ensure that the logical.type.conversion.enabled property is not set as true for the key/value SerDes classes of the Avro Converter. The logical.type.conversion.enabled property only needs to be enabled on the consumer/producer side when the consumer/producer is not part of a Kafka Connect connector.

Related Information

Single Message Transforms

KafkaAvroDeserializer properties reference
Schema Registry Client properties reference
Compatibility policies
Integrating Kafka with Schema Registry

Configuring EOS for source connectors

Learn how to enable and configure exactly-once semantics (EOS) for Kafka Connect source connectors.

About this task

EOS is a framework that enables Kafka and Kafka applications to guarantee that each message is delivered precisely once without it being duplicated or lost. EOS is supported by some Kafka Connect source connectors. To enable EOS, you must enable EOS for the Kafka Connect service roles (Kafka Connect workers) and configure your connectors to use EOS.

EOS is enabled for the Kafka Connect service roles in Cloudera Manager with the Exactly-once support for source connectors property. Connectors can be configured using the Streams Messaging Manager UI (recommended), Streams Messaging Manager API, or Kafka Connect API.

Before you begin

- Enabling EOS for Kafka Connect service roles only enables the use of EOS on the level of the service. For individual connectors to make use of EOS, the connector must be specifically configured to use EOS.
- Not all connectors shipped in Cloudera support EOS. The following list collects the source connectors shipped in Cloudera that support EOS.
  - JDBC Source
  - SFTP Source
  - NiFi Stateless Source
  - FileStreamSource

  Important: The NiFi Stateless Source connector is used to deploy your own, custom developed dataflows as Kafka Connect connectors. In order for your connector to support EOS, the dataflow that you deploy must include a source processor that supports state management on the scope of the cluster. For more information see the official NiFi documentation.
- If you are using a third-party connector, review the documentation of the connector to learn whether EOS is supported.

Procedure

1. Configure Kafka Connect service roles:
   a) In Cloudera Manager, select the Kafka service.
   b) Go to Configuration.
   c) Set Exactly-once support for source connectors in the cluster to preparing.
   d) Restart all Kafka Connect service roles.
   e) Set the Exactly-once support for source connectors in the cluster to enabled.
   f) Restart all Kafka Connect service roles.
2. Configure source connectors.

   The following table collects the common source connector properties that you can use to enable and configure EOS. Configure the following properties based on your requirements. Cloudera recommends that you use the Streams Messaging Manager UI to configure and deploy connectors. For more information on connector

configuration and deployment see *Deploying and managing Kafka Connect connectors in Streams Messaging Manager*

Table 2: EOS specific Kafka Connect source connector properties

Name	Default value	Description
exactly.once.support	requested	Permitted values are requested and required. If set to required, forces a preflight check for the connector to ensure that it can provide exactly-once delivery with the given configuration. Some connectors may be capable of providing exactly-once delivery but not signal to Kafka Connect that they support this. In this case, review the documentation for the connector before connector deployment and set this property to requested. Additionally, if the value is set to required but the worker that performs preflight validation does not have exactly-once support enabled for source connectors, requests to create or validate the connector will fail.
transaction.boundary	poll	Permitted values are poll, connector, and interval. If set to poll, a new producer transaction is started and committed for every batch of records that each task from this connector provides to Kafka Connect. If set to connector, relies on connector-defined transaction boundaries; note that not all connectors are capable of defining their own transaction boundaries, and in that case, attempts to create them with this property set to connector will fail. If set to interval, commits transactions only after a user-defined time interval has passed.
offsets.storage.topic	null	The name of a separate offsets topic to use for this connector. If left empty or not specified, the worker’s global offsets topic name is used. If specified, the offsets topic is created if it does not already exist on the Kafka cluster targeted by this connector (which may be different from the one used for the worker's global offsets topic if the bootstrap.servers property of the connector's producer has been overridden from the worker's).
transaction.boundary.interval.ms	null	If transaction.boundary is set to interval, determines the interval for producer transaction commits by connector tasks. If unset, defaults to the value of the worker-level offset.flush.interval.ms property.

Related Information

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Securing Kafka Connect

Learn about the different options and features available for securing your Kafka Connect deployment.

Kafka Connect to Kafka broker security

Learn about the options you have for securing the connection between Kafka Connect and Kafka brokers.

Configuring TLS/SSL encryption for the Kafka Connect role

Kafka Connect roles inherit the TLS/SSL configuration of the parent Kafka service. If you are deploying Kafka Connect roles under a Kafka service that already has TLS/SSL enabled, Cloudera Manager will automatically enable TLS/SSL for Connect as well. If required however, you can manually enable or disable TLS/SSL.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find and configure the following properties based on your cluster and requirements.

Table 3:

Cloudera Manager Property	Description
Enable TLS/SSL for Kafka Connect ssl_enabled	Encrypt communication between clients and Kafka Connect using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).
Kafka Connect TLS/SSL Server JKS Keystore File Location ssl_server_keystore_location	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when Kafka Connect is acting as a TLS/SSL server.
Kafka Connect TLS/SSL Server JKS Keystore File Password ssl_server_keystore_password	The password for the Kafka Connect keystore file.
Kafka Connect TLS/SSL Server JKS Keystore Key Password ssl_server_keystore_keypassword	The password that protects the private key contained in the JKS keystore used when Kafka Connect is acting as a TLS/SSL server.
Kafka Connect TLS/SSL Trust Store File ssl_client_truststore_location	The location on disk of the trust store, used to confirm the authenticity of TLS/SSL servers that Kafka Connect might connect to. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
Kafka Connect TLS/SSL Trust Store Password ssl_client_truststore_password	The password for the Kafka Connect TLS/SSL Trust Store File. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.
ssl.client.auth	Client authentication mode for SSL connections. This configuration has three valid values, required, requested, and none. If set to required, client authentication is required. If set to requested, client authentication is requested and clients without certificates can still connect. If set to none, which is the default value, no client authentication is required

4. Click Save Changes.
5. Restart the service.

Results

TLS/SSL encryption is configured for the Kafka Connect role.

Configuring Kerberos authentication for the Kafka Connect role

Learn how Kerberos authentication is configured for the Kafka Connect role.

Kafka Connect roles inherit the Kerberos configuration of the parent Kafka service. If you are deploying Kafka Connect roles under a Kafka service that already has Kerberos enabled, Cloudera Manager will automatically enable Kerberos for Kafka Connect as well. Other than making sure that the Kafka service's Kerberos configuration is correctly set, no additional user action is required.

Kafka Connect REST API security

Learn about the options you have for securing the Kafka Connect REST API.

Kafka Connect REST API authentication

Learn about the authentication options that are available for the Kafka Connect REST API.

Configuring TLS/SSL client authentication for the Kafka Connect REST API

Learn how to configure TLS/SSL client authentication for the Kafka Connect REST API.

About this task

You can secure the Kafka Connect API by configuring the Kafka Connect roles to require SSL Client authentication. This can be done by setting the SSL Client Authentication property to required. When set to required, only clients that pass SSL client authentication will be able to access the Kafka Connect API. As a result, any client that you would like to give access to should have its certificate added to the Kafka Connect truststore. This includes Streams Messaging Manager as well. Cloudera recommends that in secure environments only Streams Messaging Manager is given access to the Kafka Connect API.

In addition to setting client authentication to required, you may also want to consider setting up a firewall using third party tools to further secure access to the Kafka Connect API. Note however, that even with a firewall in place and SSL authentication set to required, if Streams Messaging Manager is given access to the Kafka Connect API, then any user that has access to Streams Messaging Manager will be able to interact with the Kafka Connect API. This is due to Streams Messaging Manager not enforcing authorization checks when users are accessing Kafka Connect functionality within Streams Messaging Manager. This is true for both the Streams Messaging Manager UI and Streams Messaging Manager REST API. As a result, caution is advised even if the Kafka Connect API itself is secured.

Complete the following steps to set SSL Client Authentication to required.

Procedure

1. Select the Kafka Service.
2. Go to Configuration.
3. Find the SSL Client Authentication property.
4. Set the property to required.

   Important:

   The SSL Client Authentication property is available for both Broker and Connect roles. Make sure that you are configuring the property for the Kafka Connect role or for the role group that includes your Kafka Connect roles.
5. Click Save Changes.
6. Restart the service.

Results

Only authenticated clients are allowed to connect to the Kafka Connect API.

What to do next

If you are using Streams Messaging Manager to manage and monitor Kafka Connect, and you are not using auto TLS, add the certificate of Streams Messaging Manager to the Kafka Connect truststore.

Configuring SPNEGO Authentication and trusted proxies for the Kafka Connect REST API

Learn how you enable SPNEGO authentication and configure trusted proxies for the Kafka Connect REST API.

About this task

Note: SPNEGO authentication is automatically enabled if Kerberos is enabled for the Kafka service. As a result, SPNEGO authentication might already be enabled. However, If you upgraded your cluster from a version that did not have SPNEGO authentication available to a version that includes SPNEGO authentication, SPNEGO is not automatically enabled.

You can secure the Kafka Connect REST API by enabling SPNEGO authentication. This can be done with the Enable SPNEGO Authentication For Kafka Connect property in Cloudera Manager. If SPNEGO authentication is enabled, only users authenticated with Kerberos are able to access and use the REST API. Additionally, if Ranger authorization is enabled for the Kafka service, authenticated users are only able perform the operations that they are authorized for. If Ranger is not enabled, by default all authenticated users are able to perform all operations. Because users authenticate using Kerberos, securing the REST API using SPNEGO requires that Kerberos is enabled for the Kafka service.

In addition, the authentication framework supports trusted proxies. If the authenticated principal is recognized as a trusted proxy principal, then the authenticator accepts the doAs query parameter in the REST URL to specify the acting user. For example:

```
https://[***KAFKA CONNECT HOST***]:28085/connector-permissions?doAs=systest
```

In this example, systest is specified as the acting user. The doAs parameter is only accepted if the authenticated principal is recognized as a trusted proxy. By default, the Knox and Streams Messaging Manager service principals (specifically, the short names of their Kerberos service principals) are recognized as trusted proxies. Trusted proxies can be configured with the List Of Trusted Proxy Services Cloudera Manager property.

Before you begin

Ensure that Kerberos is enabled for the Kafka service. For more information, see Enable Kerberos authentication.

Procedure

1. In Cloudera Manager select the Kafka service.
2. Go to Configuration.
3. Find and enable the Enable SPNEGO Authentication For Kafka Connect property.
4. Find and configure the List Of Trusted Proxy Services property.

   Add the Kerberos principal short names of the services that you want to act as trusted proxies.
5. Click Save Changes.
6. Restart the Kafka service.

Kafka Connect Rest API authorization

Learn about the authorization model shipped with Kafka Connect as well as how the model is integrated in Ranger.

Kafka Connect authorization model

Learn about the authorization model shipped with Kafka Connect.

Kafka Connect comes with an authorization model. Implementations are pluggable and it is up to the implementation how the capabilities of the model are utilized. The authorization model is implemented by default in Ranger. As a result, Cloudera recommends that you use Ranger.

Resource types

Cluster

Cluster is a singleton resource representing the cluster. This resource has the following operations:

- View: view connector plugins and the root path (basic cluster information).
- Validate: validate connector plugin configuration.
- Manage: manage cluster level resources such as loggers.

Connector

Connector is the resource representing a connector and its tasks. This resource has the following operations:

- View: view connector and task configuration as well as status.
- Manage: pause, resume, and restart connectors.
- Create: create connectors.
- Edit: edit connector configuration.
- Delete: delete connectors.

Error codes

When enabled, the Kafka Connect authentication and authorization framework can return the following HTTP errors in case of an authentication or authorization failure:

401 Unauthorized

The user attempts to connect the Kafka Connect REST server without authentication.

403 Forbidden

The user attempts an operation on a resource that is not permitted, but the user is entitled to view the resource. For example, the user tries to restart a connector that they only have view access to.

404 Not Found

The user attempts an operation on a resource that is not permitted and the resource is not visible by the user.

API endpoint to report user rights on a connector

The Kafka Connect version shipped with Cloudera Runtime includes the /connector-permissions endpoint. This endpoint is unique to Kafka Connect in Cloudera Runtime. It returns all connectors the user has permission to view along with access rights for each connector. The purpose of the endpoint is to allow the Streams Messaging Manager UI to enable and disable UI controls based on what the user can do with each connector. This endpoint is not meant to check access rights in the client, these are always checked on the backend. The endpoint responds in the following format:

```
{"createAllowed":true,
  "permissions": {
      "connectorName-1":{"editable":true, "deletable":true, "manageable":t
rue},
      "connectorName-2":{"editable":true, "deletable":true, "manageable":t
rue}
  }
}
```

Semantics for listing type endpoints

Listing endpoints (/connectors and /connector-permissions) only return the connectors that the user is permitted to view. In case the authenticated user has no rights to view any connectors it returns an empty list. Similarly to other endpoints, it responds with 401 Unauthorized if there is no authenticated user.

Kafka Connect Ranger integration

Learn about how the Kafka Connect authorization model is integrated with Ranger.

If Ranger authorization is enabled for the Kafka service and you have Kerberos and SPNEGO authentication enabled for Kafka Connect, you have the ability to set up fine grained access to the Kafka Connect REST API using the Ranger console.

This means that you can create policies and limit what operations an authenticated user is permitted to carry out. Policies that are set up are valid for any user that accesses the Kafka Connect REST API. This includes direct REST API access as well as access through Streams Messaging Manager.

If a user tries to carry out an operation on a resource that they do not have permission for, the REST API returns an error. For more information on the specific error messages, see *Kafka Connect authorization model*. In Streams Messaging Manager, the policies control what UI operations are available for the user. For example, if a user does not have access rights to view select connectors, those connectors are not shown when logged in to the Streams Messaging Manager UI. Similarly, if a user for example does not have the rights to create connectors, the New Connector option is not be available for the user.

Related Information

Kafka Connect authorization model

Configuring SPNEGO Authentication and trusted proxies for the Kafka Connect REST API

Kafka Connect connector configuration security

Learn about the options you have for hardening the security of connector configurations.

Kafka Connect Secrets Storage

Learn about the Kafka Connect Secrets Storage feature which can be used to hide and securely store sensitive data found in connector configurations.

In most production environments, the Kafka Connect connectors that you deploy connect to and move data either from or into secured services. Because of this, connectors that you deploy might require various credentials to access services. Credentials like passwords, access keys, or any other sensitive information must be provided to the connectors in their configuration. By default, the connector configurations can be easily read, their contents are unencrypted. Additionally, if you are using Streams Messaging Manager to manage and deploy connectors, all sensitive information might be visible to anyone who has access to Streams Messaging Manager with appropriate authorization rights.

Kafka Connect Secrets Storage is a feature within Kafka Connect that makes it possible for you to hide any sensitive values found in connector configurations and to store these values in a secure manner. For example, assume that you want to deploy a connector that connects to a secured service that requires a username and password for access. By default the password could be easily read by anyone with access to the connector configuration as the configuration is stored in plaintext. This poses a security risk. To circumvent this, you can mark the password as a secret. Once you mark the password as a secret, the Kafka Connect Secret Storage feature ensures that the password is stored and handled in a secure manner by doing the following:

- The property is replaced by a reference in the configuration. The reference resolves to the actual password at runtime. That is, the value is no longer readable if the configuration is accessed.
- The actual value of the password is encrypted and stored in an internal Kafka topic used by the feature.
- The value gets hidden (locked) on the Streams Messaging Manager UI.

Kafka Connect Secrets Storage terms and concepts

Review the list of terms and concepts related to the Kafka Connect Secrets Storage feature.

Secure Connector

A secure connector is any connector intended to be used in Cloudera Kafka Connect whose configuration has a sensitive property.

Secret

Part of a secure connector configuration that must be protected. A secret can be any value used in the connector configuration that you mark as confidential data. For example, a secret can be a password, a JAAS credential, an SSL certificate, and so on. When you mark a property as a secret it becomes protected. Specifically, the value of the property is no longer stored in plaintext within the configuration. Rather, it is replaced by a secret reference. The actual value is encrypted and stored in an internal Kafka topic (secrets topic).

Understand however, that this protection covers the configuration only. If you mark a property a secret that is otherwise public information, while the value of that property will be hidden in the configuration, the value could be looked up by other means. For example, sink connectors have a topics property. If auto topic creation is enabled, topics will be created by the connector automatically. This means that even if the topics property is marked as a secret, the value of the property could be retrieved by other means. For example, by listing the topics of the Kafka service.

Secrets topic

The secrets topic is an internal Kafka topic. The Kafka Connect Secrets Storage feature stores its internal state, configuration, secrets, and so on in this internal topic.

Secret reference

Part of a secure connector configuration that is already protected. Instead of the configuration's original value, a placeholder (reference) is used, which refers to the original value using a connector name and a bundle ID. The Kafka Connect Secrets Storage feature resolves these references and substitutes them with the original values stored in the secrets topic when connectors are used.

Encryption key

The encryption key is a randomly generated key. Its established value is unique at a certain point in time. However, it can be changed. It is used to encrypt and decrypt hashed secrets protecting them against tampering and retrieval.

Global key

Global key is a unique cryptographic key that is derived from a user supplied password. The global key protects the encryption key when the encryption key is persisted to the disk or exchanged on the network. The encryption key is hashed and encrypted with the global key. As a result, it is impossible to tamper with it, or retrieve the encryption key from wiretapping the network traffic in a man-in-the-middle style attack.

Bundle and Bundle ID

The set of secrets of a secure connector constitute a bundle. The bundle is identified by its bundle ID, which is maintained by the Kafka Connect Secrets Storage feature. If a secret is added, edited, or deleted, a new bundle is created, with a new bundle ID.

Managing secrets using the REST API

Learn how you can use the Kafka Connect Secrets Storage feature to manage secrets in connector configurations using the Kafka Connect REST API.

You can use the Kafka Connect Secrets Storage feature with the Kafka Connect REST API. The following sections walk you through how you can mark properties as secrets and how you can manage already existing secrets.

Important: The Kafka Connect Secrets Storage feature can also be used when managing connectors using the Streams Messaging Manager UI. For more information, see Deploying and managing Kafka Connect connectors in Streams Messaging Manager.

Mark a configuration value as a secret in a connector configuration

To mark a property value in a connector configuration as a secret, you must:

- Add the property to the configuration.
- Add the secret.properties property. The value of this property is a comma separated list of the property keys that you want to mark as secrets.

For example, assume that you have two sensitive properties in your configuration that you want to mark as secrets, password.for.connector and some.confidential.config.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config”,
“topics” = “...”,
“password.for.connector” = “mypassword”,
“some.confidential.config” = “sensitive information”
}
```

If the secret.properties is present in the configuration, the Kafka Connect Secrets Storage feature gets enabled. Once the connector configuration is submitted, the feature processes the configuration and makes a number of changes.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config”,
“secret.bundle.id” = “18a1b3a6-1e2f”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/18a1b3a6-1e2f:password.for.conne
ctor}”,
“some.confidential.config” = “${secret:Custom/18a1b3a6-1e2f:some.confidentia
l.config}”
}
```

Notice the following changes:

- A secret.bundle.id property is added. The bundle is the set of secrets of a secure connector.

  The bundle is identified by its bundle ID, which is maintained by the Kafka Connect Secrets Storage feature. If a secret is added, deleted, or edited, a new bundle is created with a new bundle ID.
- The password.for.connector and some.confidential.config properties had their values replaced with secret references.

  These references are placeholder values consisting of the connector name, bundle ID, and property key. If a reference like this is present in the configuration, it means that the value for that property is securely stored in an internal Kafka topic (secrets topic). The Kafka Connect Secrets Storage feature resolves these references and substitutes them with the original values stored in the secrets topic when the connectors are used.

Update a connector configuration with a new secret

If you want to update an already existing configuration with a new secret property, you must add the new property to the configuration as well as the secret.properties property. For example, assume that you have a configuration where password.for.connector and some.confidential.config are already marked as secrets. Now, you want to add the new. secret.config property to the configuration and mark it as a secret as well.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config,ne
w.secret.config”,
“secret.bundle.id” = “18a1b3a6-1e2f”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/18a1b3a6-1e2f:password.for.con
nector}”,
“some.confidential.config” = “${secret:Custom/18a1b3a6-1e2f:some.confident
ial.config}”,
“new.secret.config” = “[***A NEW SECRET VALUE***]”
}
```

Once submitted and processed, the configuration will look like the following example:

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config,ne
w.secret.config”,
“secret.bundle.id” = “c8a753d6-8242”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/c8a753d6-8242:password.for.con
nector}”,
“some.confidential.config” = “${secret:Custom/c8a753d6-8242:some.confident
ial.config}”,
“new.secret.config” = “${secret:Custom/c8a753d6-8242:new.secret.config}”
}
```

Notice the following changes:

- The secret.bundle.id value has changed.

  This is done because adding a new secret creates a new set of secrets resulting in a new bundle being created.
- The values (secret references) of password.for.connector and some.confidential.config are updated with the new bundle ID.
- The value of new.secret.config is replaced by a secret reference.

  This means that the value is now securely stored in the secrets topic.

Update a connector configuration by changing a secret

Updating already existing secret values can be done by updating the appropriate property values. For example, assume that you have three properties marked as secrets, password.for.connector, some.confidential.config, and new.secret.config. You want to update the value of new.secret.config.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config,ne
w.secret.config”,
“secret.bundle.id” = “c8a753d6-8242”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/c8a753d6-8242:password.for.con
nector}”,
“some.confidential.config” = “${secret:Custom/c8a753d6-8242:some.confident
ial.config}”,
“new.secret.config” = “[***CHANGED VALUE***]”
}
```

Once submitted and processed, the configuration will look like the following example:

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config,ne
w.secret.config”,
“secret.bundle.id” = “14b2f85f-1dcb”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/14b2f85f-1dcb:password.for.con
nector}”,
“some.confidential.config” = “${secret:Custom/14b2f85f-1dcb:some.confident
ial.config}”,
“new.secret.config” = “${secret:Custom/14b2f85f-1dcb:new.secret.config}”
}
```

Notice the following changes:

- The secret.bundle.id value has changed.

  This is done because deleting a secret creates a new set of secrets resulting in a new bundle being created. In this case the deleted secret is removed from the bundle.
- The values (secret references) of password.for.connector and some.confidential.config are updated with the new bundle ID.

  This is done because updating a secret creates a new set of secrets resulting in a new bundle being created.
- The value of new.secret.config is replaced by a secret reference.

  This means that the new value is now securely stored in the secrets topic.

Remove an existing secret from an existing configuration

Secrets can be removed from a configuration by updating the configuration and deleting the unnecessary properties from both the configuration and secret.properties. For example, assume you have three properties marked as secrets, password.for.connector, some.confidential.config, and new.secret.config.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config,ne
w.secret.config”,
“secret.bundle.id” = “14b2f85f-1dcb”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/14b2f85f-1dcb:password.for.con
nector}”,
“some.confidential.config” = “${secret:Custom/14b2f85f-1dcb:some.confident
ial.config}”,
“new.secret.config” = “${secret:Custom/14b2f85f-1dcb:new.secret.config}”
}
```

Assume you want to delete some.confidential.config. To do this, you need to delete the property from the configuration and remove the property key from secret.properties.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,new.secret.config”,
“secret.bundle.id” = “14b2f85f-1dcb”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/14b2f85f-1dcb:password.for.conn
ector}”,
“new.secret.config” = “${secret:Custom/14b2f85f-1dcb:new.secret.config}”
}
```

Once submitted and processed, the configuration will look like the following example:

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,new.secret.config”,
“secret.bundle.id” = “15641fca-483b”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/15641fca-483b:password.for.conn
ector}”,
“new.secret.config” = “${secret:Custom/15641fca-483b:new.secret.config}”
}
```

Notice the following changes:

- The secret.bundle.id value has changed.

  This is done because deleting a secret creates a new set of secrets resulting in a new bundle being created. In this case the deleted secret is removed from the bundle.
- The values (secret references) of password.for.connector and some.confidential.config are updated with the new bundle ID.

Remove all secrets

To remove all secrets from a configuration, you must remove all properties that were marked as secrets and also remove the secret.properties property. However, you must leave secret.bundle.id intact. For example, assume you had a single property marked as a secret, password.for.connector.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,new.secret.config”,
“secret.bundle.id” = “c8a753d6-8242”,
“topics” = “...”,
“password.for.connector”= “${secret:Custom/c8a753d6-8242:password.for.conne
ctor}
}
```

In a case like this, you need to remove password.for.connector and secret.properties from the configuration. However, you must preserve secret.bundle.id. The reason why you need to leave this property intact is because the encrypted values for the secret properties must be deleted from the internal secrets topic. The Kafka Connect Secrets Storage feature can only do this if it has a bundle ID for reference.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.bundle.id” = “c8a753d6-8242”,
“topics” = “...”
}
```

Once the configuration is submitted and processed, secret.bundle.id is removed. The connector configuration at this point is considered unsecured.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“topics” = “...”
}
```

Re-encrypting secrets

Using the Kafka Connect Secrets Storage feature places encrypted confidential information in an internal Kafka topic (secrets topic) used by the feature. If you want to change the encryption key used to encrypt the information placed in the topic, all secrets must be re-encrypted and must be migrated to a new topic. This is done using the connect-secret-storage-migration tool that is shipped with Cloudera Runtime.

About this task

The connect-secret-storage-migration tool migrates existing secrets from the currently used secrets topic (source) to a newly created one (target). It achieves this by reading through the source topic, decrypting the secrets stored within it using the existing global key, and re-encrypts them with a new, randomly generated encryption key and a new global key that you configure. After encryption is finished, newly encrypted secrets are produced into the target topic. The new encryption key will have a different security context (different global password and salt) than the currently used one.

The tool requires three configuration files to function. These configuration files provide the tool with information regarding the Kafka service as well as the source and target topics. The configuration files must be created manually.

Before you begin

- In the configuration files you prepare, you must specify the current global password and PBE salt used by the feature. These values are specified in Cloudera Manager with the Kafka Connect Secrets Storage PBE Salt and Kafka Connect Secrets Storage Global Password properties. The default values for these properties are randomly generated and are hidden, but can be changed at any time. As a result, if you do not know the global password or PBE salt value, you must change both values, note down the values you configured, and restart all Kafka Connect roles before you complete the following steps.

Procedure

1. Stop all Kafka Connect roles.
2. Prepare the configuration files required for the tool.

   You need to prepare three different configuration files. A Kafka client configuration file that the tool can use to access the Kafka service. Additionally, two configuration files that contain information regarding the source and target topics. All three files must be deployed on the cluster host where you run the tool.

   a) Create a Kafka client configuration file.

   This configuration file must contain all client properties that the tool requires to access the Kafka service. The properties you need to specify in this file depend on the security configuration of the Kafka service. The following tabs contain examples for some of the most commonly used security configurations.

   For SASL_SSL Kerberos

   ```
   bootstrap.servers=[***HOST***]:[***PORT***]
   security.protocol=SASL_SSL
   sasl.mechanism=GSSAPI
   sasl.kerberos.service.name=kafka
   sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule require
   d useKeyTab=true storeKey=true keyTab="[***PATH TO KEYTAB***]" princi
   pal="[***KERBEROS PRINCIPAL***]";
   ssl.truststore.location=[***PATH TO TRUSTSTORE.JKS***]
   ssl.truststore.password=[***TRUSTSTORE PASSWORD***]
   ```

   For SASL_SSL PLAIN

   ```
   bootstrap.servers=[***HOST***]:[***PORT***]
   security.protocol=SASL_SSL
   sasl.mechanism=PLAIN
   sasl.jaas.config=org.apache.kafka.common.security.plain.Pla
   inLoginModule required username="[***USERNAME***]" passwo
   rd="[***PASSWORD***]";
   ssl.truststore.location=[***PATH TO TRUSTSTORE.JKS***]
   ssl.truststore.password=[***TRUSTSTORE PASSWORD***]
   ```

   b) Create the configuration file identifying the source topic and its security context. For example:

   ```
   kafka.connect.secret.storage.topic : [***SOURCE TOPIC NAME***]
   kafka.connect.secret.global.password : [***CURRENT GLOBAL PASSWORD***]
   ```

```
kafka.connect.secret.pbe.salt : [***CURRENT PBE SALT***]
```

Replace *[***SOURCE TOPIC NAME***]* with the name of the currently used secrets topic. The name of the current topic is specified in Cloudera Manager in the Kafka Connect Secrets Storage Topic Name property.

Replace *[***GLOBAL PASSWORD***]* with the currently used global password.

Replace *[***CURRENT PBE SALT***]* with the currently used PBE salt value.

You can choose to not specify kafka.connect.secret.global.password in the configuration. If the property is not added to the configuration, you are prompted to enter the password when you run the tool.

c) Create the configuration file identifying the target topic and its security context. For example:

```
kafka.connect.secret.storage.topic : [***TARGET TOPIC NAME***]
kafka.connect.secret.global.password : [***NEW GLOBAL PASSWORD***]
kafka.connect.secret.pbe.salt : [***NEW PBE SALT***]
kafka.connect.secret.storage.topic.replication.factor : [***REPLICATION
 FACTOR***]
kafka.connect.secret.storage.topic.configs.min.insync.replicas : [***IN-
SYNC REPLICA NUMBER***]
```

You can choose to not specify kafka.connect.secret.global.password in the configuration. If the property is not added to the configuration, you are prompted to enter the password when you run the tool.

3. Run the tool.

You must specify the three configuration files you created as parameters. The order in which you specify the files is fixed. The first file must be the Kafka client configuration, the second must be the file for the source topic, and the third file must be the file for the target topic. For example:

```
connect-secret-storage-migration [***KAFKA CLIENT
 CONFIGURATION***] [***SOURCE TOPIC CONFIGURATION***] [***TARGET TOPIC
 CONFIGURAITON***]
```

4. If prompted, enter the current and new global password.

The tool only prompts you to enter the passwords if they were not specified in the configuration files. The tool does not echo the passwords that you type in the console.

5. The tool prompts you to ensure that the Kafka Connect cluster (all Connect roles) are stopped. Press Enter to continue.

6. Wait until the tool is finished with re-encryption.

The tool reports its progress. Upon successful completion, the tool exits. Upon failure, an error message is printed. In such a case, fix any issues that the tool reports and run the tool again.

7. In Cloudera Manager, select the Kafka service.

8. Go to Configuration.

9. Find and configure the following properties:

- Kafka Connect Secrets Storage Topic Name
- Kafka Connect Secrets Storage Topic Replication Factor
- Kafka Connect Secrets Storage Topic Minimum In-Sync Replicas
- Kafka Connect Secrets Storage Global Password
- Kafka Connect Secrets Storage PBE Salt

Add the values you specified in the configuration file identifying the target topic.

10. Restart all Kafka Connect roles.

11. Verify that Kafka Connect is working as expected.

This can be done by verifying that all connectors that have properties marked as secrets are running and are in a healthy state. That is, the connectors are resolving all referenced secrets correctly. If you are experiencing issues, you can revert the configuration changes in Cloudera Manager and restart the re-encryption process.

12. Delete the configuration files you created.
13. Delete the source secrets topic.

Important: Ensure that you verified that Kafka Connect and your connectors are working as expected before deleting the topic. When you delete the topic, the secrets stored within it are lost. As a result, if there are issues with Kafka Connect, you might not be able to revert the cluster to a working state.

Configuring connector JAAS configuration and Kerberos principal overrides

Learn how to configure Kafka Connect connectors to override the default JAAS configuration and Kerberos principal that they use to connect to the Kafka service. Configuring these overrides enables you to set up fine grained access rules for the connectors that you deploy.

About this task

By default Kafka Connect connectors, specifically the clients managed by Connect, use the JAAS configuration and Kerberos principal of the Connect worker (Kafka Connect role) to establish a connection with the Kafka service. As a result, all connectors that you deploy authenticate with the same principal. Additionally, if authorization is enabled, authorization also happens using that same principal. Because the connectors use the same principal as the worker and because the worker has all permission on the Kafka service, all connectors have full access to Kafka by default.

This behavior might not be desirable and you might want to configure fine grained access rules for individual connectors that you deploy. For example, you might want to have your connectors to be able to manage specific Kafka resources only, but not have access to other resources. To achieve this, you must configure connectors to be required to specify unique connection credentials for the Kafka service connection. Configuration is done in Cloudera Manager using Connector Kafka Client Configuration Override Policy and Require Connectors To Override Kafka Client JAAS Configuration Kafka service properties.

Procedure

1. In Cloudera Manager select the Kafka service.
2. Go to Configuration.
3. Find and configure the following properties.

Table 4:

Property Name	Description
Connector Kafka Client Configuration Override Policy	Specifies what client configuration can be overridden by the connector. Its value is the class name or alias of the ConnectorClientConfigOverridePolicy implementation. The framework includes three implementations (policies). These are None, All, and Principal. • If set to None, configurations cannot be overridden. This is the default setting. • If set to Principal, SASL configurations can be overridden • If set to All, all configurations can be overridden
Require Connectors To Override Kafka Client JAAS Configuration	Specifies whether Connector configurations should override the sasl .jaas.config property of the Kafka clients used by the connector. If set to true, connector configurations must override the client's sasl .jaas.config property. The override must also not match the Kerberos credentials of the Connect worker. Additionally, if set to true, a Unio nComposite'policy is installed under the connector.client.config.over ride.policy configuration. If Connector Kafka Client Configuration Override Policy is set to None, it is upgraded to Principal. Otherwise, the existing policy is also applied by the composite policy.

4. Click Save Changes.
5. Restart the Kafka service.

What to do next

- Update the configuration of all existing connectors to include the sasl.jaas.config property with a valid JAAS configuration.
- Ensure that the principal specified for the connector has appropriate permissions assigned to it in your authorization service.

Configuring a Nexus repository allow list

To harden the security of your Kafka Connect deployment, you can configure a Nexus repository allow list. If an allow list is configured, Stateless NiFi Source and Sink connectors are only able to fetch NiFi extensions from the specified list of Nexus repositories.

About this task

NiFi dataflows within Kafka Connect are deployed using the Stateless NiFi Source and Sink connectors. When you deploy your custom developed dataflows with the Stateless NiFi connectors, you can configure the nexus.url property. This property specifies the URL of a Nexus repository that hosts the NiFi extensions that are not bundled with the connectors but are required by the dataflow that is run within the connector.

By default connectors you deploy are allowed to connect to any Nexus repository that you specify. However, this may pose a security risk. Because of this, you can configure Kafka Connect to only allow its connectors to connect to specific Nexus repositories. That is, you can configure a Nexus server allow list. If an allow list is configured, connectors are only allowed to connect and fetch NiFi extensions from the servers that are included in the list.

The allow list is configured in Cloudera Manager using the List Of Allowed Nexus Repository Urls property.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find and configure the List Of Allowed Nexus Repository Urls property.

   Add the URLs of the Nexus repositories that you want to allow your connectors to fetch extensions from.
4. Click Save Changes.
5. Restart the Kafka service.

Results

The Nexus URLs specified in the nexus.url property of the Stateless NiFi Source and Sink connectors are validated against the list of configured URLs. Connections are only allowed to the repositories configured in the list.

Single Message Transforms

Single Message Transforms (SMT) is a message transformation framework that you can deploy on top of Kafka Connect connectors to apply message transformations and filtering. Learn about the SMT framework as well as the transformation plugins available in Cloudera.

Kafka Connect connectors provide ready-to-use tools to integrate between Kafka and external data stores. Still, in many use cases, the data moved by the connectors require some sanitization and transformation. To provide extra flexibility built on top of connectors, Kafka Connect also supports an SMT framework.

The SMT framework installs a transformation chain on top of connectors that modifies and filters the data on a single message basis. An SMT chain consists of transform and predicate plugins. Transform plugins are used to modify the data. For example, you can insert, replace, mask as well as perform various other modifications on the messages moved by connectors. Predicate plugins are used to add additional logic to your chain so that the transformation chain is only applied to messages which satisfy specified conditions.

The SMT framework requires that data is converted to the Kafka Connect internal data format. This data format is specific to Kafka Connect and consists of a structure and schema descriptor (SchemaAndValue) specific to Connect.

Supported SMT plugins

Cloudera Kafka ships with and supports the following SMT plugins:

- All Apache Kafka plugins. For more information, see Transformations in the Apache Kafka documentation.
- All Debezium plugins. For more information, see Transformations in the Debezium documentation.
- The following Cloudera specific plugins:
  - ConvertFromBytes on page 36
  - ConvertToBytes on page 38

Note: The ConverFromBytes and ConvertToBytes transformation plugins transform binary data to and from the Kafka Connect internal data format. These plugins are specifically developed to enable the use of the SMT framework with connectors that only support binary data.

Configuring an SMT chain

Learn how to configure a Single Message Transformation (SMT) chain for Kafka Connect connectors.

SMT chains can be configured within the configuration of a Kafka Connect connector using SMT specific configuration properties. To set up a chain, you first define your transformation chain with the transforms property and optionally define your predicates using the predicates property. Afterward, you use transforms.* and predicates.* to configure the plugins in the chain. For example, the following configuration snippet sets up a transformation chain that filters messages based on their header and removes a specified field from messages.

```
transforms=FilterAudit,MaskField

transforms.MaskField.type=org.apache.kafka.connect.transforms.MaskField$Va
lue
transforms.MaskField.fields=CreditCardNumber

transforms.FilterAudit.type=org.apache.kafka.connect.transforms.Filter
transforms.FilterAudit.predicate=IsAudit
transforms.FilterAudit.negate=false

predicates=IsAudit
predicates.IsAudit.type=org.apache.kafka.connect.transforms.predicates.Has
HeaderKey
predicates.IsAudit.name=Audit
```

The following sections go through the properties in this example and give an overview on how to set up a transformation chain.

Configuring transforms

The transforms property contains a comma-separated list of transformation aliases. Each alias represents one step in the transformation chain. The aliases you add to the property are arbitrary names, they are used in other properties to configure that particular transformation step. For example, the following defines a two step transformation chain.

```
transforms=FilterAudit,MaskField
```

The transforms.*[***ALIAS***]*.type property specifies which transformation plugin should be used in a transformation step. *[***ALIAS***]* is one of the aliases that you specified in transforms. The value of the property is the fully qualified name of the transformation plugin that should be used in the step. For example, the following line specifies org.apache.kafka.connect.transforms.MaskField$Value as the plugin for the MaskField step.

```
transforms.MaskField.type=org.apache.kafka.connect.transforms.MaskField$
Value
```

Many transformation plugins support changing both the key and the value of a record. For these plugins, typically, a nested value or key class can be referenced as the type.

The transforms.*[***ALIAS***]*.*[***KEY***]* property is used to configure the transformation plugins in your chain. This property is passed to the transformation plugin itself with transforms.*[***ALIAS***]* stripped from the property key. *[***ALIAS***]* is the alias of a plugin you specified in transforms. *[***KEY***]* is a property key that the plugin accepts. For example, the MaskField plugin has a fields property that specifies which fields should be removed from the structure.

```
transforms.MaskField.fields=CreditCardNumber
```

Configuring predicates

Predicates are a separate set of plugins. You use them to conditionally enable certain steps in the transformation chain. Predicates are configured in a similar way to transforms. You must specify the predicate aliases, associate the aliases with a plugin, and set plugin specific properties.

```
predicates=IsAudit
predicates.IsAudit.type=org.apache.kafka.connect.transforms.predicates.Ha
sHeaderKey
predicates.IsAudit.name=Audit
```

In this example the IsAudit predicate is an instance of the HasHeaderKey predicate plugin. This predicate returns true for records where a specific header key is present. predicates.IsAudit.name=Audit configures the predicate to look for the Audit header in the records.

After a predicate is set up, you can associate the predicate with any number of transformation steps using the pred icate property. If a predicate is associated with a transformation, that transformation step is only applied to the messages that match the condition specified in the predicate.

A good example for using a predicate is the Filter transformation plugin. This is because Filter filters (drops) all messages by default. Therefore, it must be used together with predicates to specify filtering logic. For example, the following configuration instructs the SMT framework that the FilterAudit step should only be invoked for messages where the IsAudit predicate returns true. That is, all messages with the Audit header will be filtered and will not be transformed by any subsequent steps in the transformation chain.

```
transforms.FilterAudit.predicate=IsAudit
transforms.FilterAudit.negate=false
```

The condition of a predicate can be inverted using negate. If negate is set to true, the SMT framework applies the transformation to any record that does **not** match the condition. For example, the following configuration instructs the

SMT framework that the FilterAudit step should only be invoked for messages where the IsAudit predicate returns false.

```
transforms.FilterAudit.predicate=IsAudit
transforms.FilterAudit.negate=true
```

ConvertFromBytes

ConvertFromBytes is a Cloudera specific transformation plugin that converts binary data to the Kafka Connect internal data format. You can use this plugin to make connectors that only support binary data compatible with the Single Message Transforms (SMT) framework.

Fully qualified names

- com.cloudera.dim.kafka.connect.transformations.convert.ConvertFromBytes$Key
- com.cloudera.dim.kafka.connect.transformations.convert.ConvertFromBytes$Value

Description

Important: Ensure that you have an in-depth understanding about the following aspects of the connector that you plan on using with ConvertFromBytes.

- The type of the connector
- The converter type used by the connector
- The header converter type used by the connector

These aspects of a connector heavily influence how the plugin and how the transformation chain that includes this plugin must be configured.

The ConvertFromBytes transformation plugin accepts binary data and converts it into the Kafka Connect internal data format with a nested converter that transforms binary data. To support header based converter logic, for example when using the AvroConverter with the schema ID encoded in the record header, the plugin requires a header converter to correctly transform record headers when interacting with the converter. This plugin supports both key and value conversion.

Using this plugin with connectors that only support binary data makes the connector fully compatible with the SMT framework. On their own, connectors that only support binary data have limited compatibility with transformations even if the binary data is structured. This is because transformations are only fully supported on data that is in the Kafka Connect internal data format. Binary only connectors like the Mirror Source (MirrorSourceConnector) or the Stateless NiFi Source connectors emit data that has the BYTES schema and do not provide conversion to the Kafka Connect internal data format by default. When you use a binary only connector with the ConvertFromBytes plugin, the binary data is parsed into a compatible structure, which can then be further processed with the transformation chain.

Figure 1: Source connector example flow with ConvertFromBytes

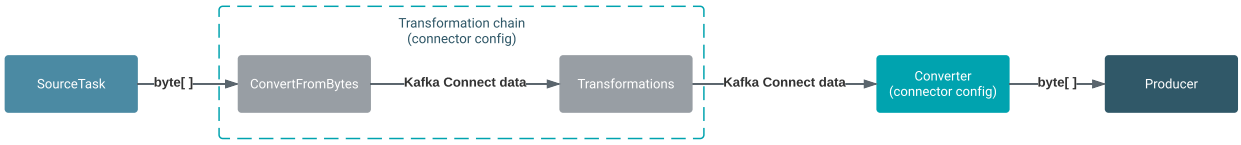

Example

The following configuration example adds a ConvertFromBytes transformation as a first step of the transformation chain. The conversion uses a schemaless JSON transformation to parse the binary data. The transformation steps, the

connector, or the converter, whichever comes directly after FromBytes, receives a properly structured record instead of binary data.

```
{
     "transforms" : "FromBytes,...",
     "transforms.FromBytes.type" : "com.cloudera.dim.kafka.connect.transformations.convert.ConvertFromBytes$Value",
     "transforms.FromBytes.converter" : "org.apache.kafka.connect.json.JsonConverter",
     "transforms.FromBytes.converter.schemas.enable" : "false"
}
```

Configuration properties

Table 5: ConvertFromBytes properties reference

Property	Default Value	Required	Description
converter		True	The fully qualified name of the converter implementation to use. For example: com.cloudera.dim.kafka.connect.converts.AvroConverter
header.converter	org.apache.kafka.connect.storage.SimpleHeaderConverter	True	The fully qualified name of the header converter implementation to use. This converter must match the header converter of the connector.
converter.		False	A configuration prefix. Use this prefix to configure the properties of the converter specified in converter. Property keys and values specified with the prefix are passed directly to the converter with the prefix stripped. For example: `transforms.[***TRANSFORM ALIAS***].converter.[***CONVERTER PROPERTY KEY***]:[***CONVERTER PROPERTY VALUE***]`

Property	Default Value	Required	Description
header.converter.		False	A configuration prefix. Use this prefix to configure the properties of the header converter specified in header.converter. Property keys and values specified with the prefix are passed directly to the header converter with the prefix stripped. For example: `transforms.[***TRANSFORM ALIAS***].converter.[***HEADER CONVERTER PROPERTY KEY***]:[***HEADER CONVERTER PROPERTY VALUE***]`

ConvertToBytes

ConvertToBytes is a Cloudera specific transformation plugin that converts Kafka Connect internal data to binary data. You can use this plugin to make connectors that only support binary data compatible with the Single Message Transforms (SMT) framework.

Fully qualified names

- com.cloudera.dim.kafka.connect.transformations.convert.ConvertToBytes$Key
- com.cloudera.dim.kafka.connect.transformations.convert.ConvertToBytes$Value

Description

Important: Ensure that you have an in-depth understanding about the following aspects of the connector that you plan on using with ConvertToBytes.

- The type of the connector
- The converter type used by the connector
- The header converter type used by the connector

These aspects of a connector heavily influence how the plugin and how the transformation chain that includes this plugin must be configured.

The ConvertToBytes transformation plugin accepts data in the Kafka Connect internal data format and converts it to binary data with a nested converter. To support header based converter logic, for example when using the AvroConverter with the schema ID encoded in the record header, the plugin requires a header converter to correctly transform record headers when interacting with the converter. This plugin supports both key and value conversion.

Using this plugin with connectors that only support binary data makes the connector fully compatible with the SMT framework. On their own, connectors that only support binary data, for example the Stateless NiFi Sink connector, have limited compatibility with transformations even if the binary data is structured. This is because the format of the data after transformations are carried out is normally the Kafka Connect internal data format. Binary only connectors, however, expect data that has the BYTES schema and do not provide conversion from the Kafka Connect internal data format by default. When you use the ConvertToBytes plugin with a binary only connector, the structured data is converted to binary format, which can then be picked up by the connector.

Figure 2: Sink connector example flow with ConvertToBytes

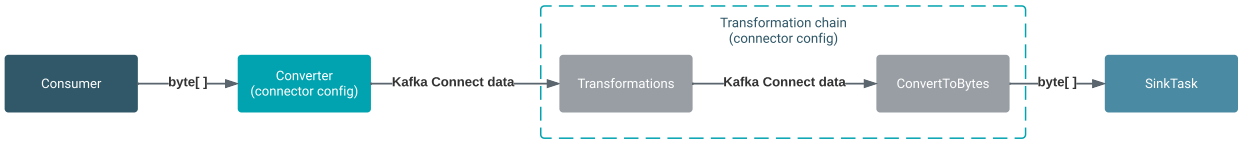

Example

The following configuration example adds a `ConvertToBytes` transformation as the last step of the transformation chain. The conversion uses a schemaless JSON transformation to serialize the structured data. The transformation steps, the connector, or the converter, whichever comes directly after ToBytes, receives a properly structured record instead of binary data.

```
{
     "transforms" : "...,ToBytes",
     "transforms.ToBytes.type" : "com.cloudera.dim.kafka.connect.transfor
mations.convert.ConvertToBytes$Value",
     "transforms.ToBytes.converter" : "org.apache.kafka.connect.json.Json
Converter",
     "transforms.ToBytes.converter.schemas.enable" : "false"
}
```

Configuration properties

Table 6: ConvertToBytes properties reference

Property	Default Value	Required	Description
converter		True	The fully qualified name of the converter implementation to use. For example: com.cloudera.dim .kafka.connect.converts.Avro Converter
header.converter	org.apache.kafka.connect.storage .SimpleHeaderConverter	True	The fully qualified name of the header converter implementation to use. This converter must match the header converter of the connector.
converter.		False	A configuration prefix. Use this prefix to configure the properties of the converter specified in conv erter. Property keys and values specified with the prefix are passed directly to the converter with the prefix stripped. For example: `transfor ms.[***TRANSFORM ALIAS***].con vert er.[***CONVERTER PROPERTY KEY***]:[***CONVERTER PROPERTY VALUE***]`

Property	Default Value	Required	Description
header.converter.		False	A configuration prefix. Use this prefix to configure the properties of the header converter specified in header.converter. Property keys and values specified with the prefix are passed directly to the header converter with the prefix stripped. For example: `transforms.[***TRANSFORM ALIAS***].converter.[***HEADER CONVERTER PROPERTY KEY***]:[***HEADER CONVERTER PROPERTY VALUE***]`

Connectors

Learn what Kafka Connect connectors are shipped with Cloudera Runtime.

Cloudera Runtime comes prepackaged with a number of Cloudera developed Kafka Connect connectors. In addition, connectors that come packaged with the version of Apache Kafka that is included in Cloudera Runtime are also available for use. Manually installing and using your own custom connectors is also possible. The following collects the connectors shipped in Cloudera Runtime.

Debezium connectors

Debezium connectors capture changes from a wide variety of databases and produce the captured changes into Kafka. Using Debezium connectors makes it possible for your applications to consume and respond to change events regardless of where the changes originated from.

Table 7: Debezium connectors

Connector	Description
Debezium Db2 Source	The Debezium Db2 Source connector captures row-level changes in Db2 database tables and transfers the changes to Kafka topics.
Debezium MySQL Source	The Debezium MySQL Source connector reads the binary log (binlog) of a MySQL server, produces change events for row-level INSERT, UPDATE, and DELETE operations, and transfers the changes to Kafka topics.
Debezium Oracle Source	The Debezium Oracle Source connector captures and records row-level changes that occur in databases on an Oracle server, including tables that are added while the connector is running, and transfers the changes to Kafka topics.
Debezium PostgreSQL Source	The Debezium PostgreSQL Source connector captures row-level INSERT, UPDATE, and DELETE operations, produces change evens for each change, and transfers the changes to Kafka topics.

Connector	Description
Debezium SQL Server Source	The Debezium SQL Server Source connector captures row-level INSERT, UPDATE, and DELETE operations that occur in the schemas of a SQL Server database, produces change events for each change, and transfer the changes to Kafka topics.

Note: Cloudera does not provide documentation on how to use or configure Debezium connectors. Documentation only covers the setup steps that are required before you can deploy the connectors in a Cloudera cluster. To learn more about these connectors, see the appropriate version of the Debezium Documentation.

Stateless NiFi connectors

The Stateless NiFi Source and Sink connectors allow you to run NiFi dataflows within Kafka Connect. Using these connectors can grant you access to a number of NiFi features without having the need to deploy or maintain NiFi on your cluster.

Stateless NiFi connectors fall in to two categories. You have the base Stateless NiFi Source (`StatelessNiFiSource`) and Stateless NiFi Sink (`StatelessNiFiSink`) connectors. In addition, there are a number of ready-to-use connectors based on Stateless NiFi Source and Sink. These Stateless NiFi-based connectors run predefined dataflows developed by Cloudera and cover common data movement use cases.

Table 8: Base connectors

Connector	Description
Stateless NiFi Source	The Stateless NiFi Source and Sink connectors allow you to run NiFi dataflows within Kafka Connect. Using these connectors can grant you access to a number of NiFi features without having the need to deploy or maintain NiFi on your cluster.
Stateless NiFi Sink	

Table 9: Predefined flows/Stateless NiFi-based connectors

Connector	Description
HTTP Source	The HTTP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The HTTP Source connector listens on a port for HTTP POST requests and transfers the request body to a Kafka topic.
JDBC Source	The JDBC Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The JDBC Source connector reads records from a database table and transfers each record to Kafka in Avro or JSON format.
JMS Source	The JMS Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The JMS Source Connector consumes messages from a JMS broker and transfers the message body to Kafka.
MQTT Source	The MQTT Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The MQTT Source connector consumes messages from an MQTT broker and transfers them to Kafka.
SFTP Source	The SFTP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The SFTP Source connector obtains files from an SFTP server and transfers them to Kafka.
Syslog TCP Source	The Syslog TCP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The Syslog TCP Source connector listens on a port for syslog messages over TCP and transfers them to Kafka.

Connector	Description
Syslog UDP Source	The Syslog UDP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The Syslog UDP Source connector listens on a port for syslog messages over UDP and transfers them to Kafka.
ADLS Sink	The ADLS Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka connect framework. The ADLS Sink connector fetches messages from Kafka and uploads them to ADLS.
HDFS Stateless Sink	The HDFS Stateless Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect Framework. The HDFS Stateless Sink Connector fetches messages from Kafka and stores them in HDFS.
HTTP Sink	The HTTP Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The HTTP Sink connector obtains messages from a Kafka topic and transfers their content in a HTTP POST requests to a specified endpoint.
InfluxDB Sink	The InfluxDB Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect Framework. The InfluxDB Sink Connector fetches messages from Kafka and loads them to InfluxDB.
JDBC Sink	The JDBC Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The JDBC Sink connector fetches messages from Kafka and loads them into a database table.
Kudu Sink	The Kudu Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The Kudu Sink connector fetches messages from Kafka and loads them into a table in Kudu.
S3 Sink	The S3 Sink Connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The S3 Sink connector fetches messages from Kafka and uploads them to AWS S3.

Standard connectors

The following are either Cloudera developed connectors or connectors that come packaged with Apache Kafka.

Table 10: Standard connectors

Connector	Description
Amazon S3 Sink	The Amazon S3 Sink connector is Cloudera developed connector that consumes data from Kafka topics and streams the data to an S3 bucket.
HDFS Sink	The HDFS Sink connector is a Cloudera developed connector that transfer data from Kafka topics to files on HDFS clusters.
`MirrorSourceConnector`	The `MirrorSourceConnector` is a connector used internally by Streams Replication Manager. Within Streams Replication Manager, this connector is responsible for replicating topics between the source and target cluster. Standalone use of this connector is not recommended by Cloudera.
`MirrorHeartbeatConnector`	The `MirrorHeartbeatConnector` is a connector used internally by Streams Replication Manager. Within Streams Replication Manager, this connector is responsible for creating the heartbeats topic in the target cluster. It also periodically produces heartbeats into the heartbeats topic. Standalone use of this connector is not recommended by Cloudera.

Connector	Description
`MirrorCheckpointConnector`	The `MirrorCheckpointConnector` is a connector used internally by Streams Replication Manager. Within Streams Replication Manager, this connector is responsible for replicating the committed group offsets between the source and target cluster. In addition, the connector is also capable of periodically applying the offsets to the consumer groups in the target cluster. Standalone use of this connector is not recommended by Cloudera.

Note: Standalone use of the `MirrorCheckpointConnector`, `MirrorHeartbeatConnector`, and `MirrorSourceConnector` is not recommended by Cloudera. Additionally Cloudera does not provide documentation on how to use or configure these connectors. However, you can review Streams Replication Manager driver architecture to learn more about how Streams Replication Manager uses these connectors internally.

Example connectors (non-production)

The FileStream Source and Sink connectors are example connectors packaged with Apache Kafka. Both FileStream connectors are meant to be used to demonstrate the capabilities of Kafka Connect and are not production ready. As a result, even though these connectors are shipped with Cloudera Runtime, they must be installed before they can be deployed. Prior to connector installation, the connectors are also not selectable for deployment on the Streams Messaging Manager UI. See Installing FileStream connectors on page 44 for installation steps.

Table 11: Non-production example connectors

Connector	Description
FileStream Sink	The FileStream Sink connector reads data from Kafka and transfers that data to a local file.
FileStream Source	The FileStream Source connector reads data from a file and transfers the data to Kafka.

Note: Cloudera does not provide documentation regarding how to use or configure FileStream connectors. For more information on connector use, see the Apache Kafka Documentation.

Note:

Information presented on this page contains content that is modified and adapted from the Debezium Documentation by the Debezium Community, which is licensed under CC BY 3.0.

Installing Kafka Connect connectors

Learn how to install custom developed (third party) connectors as well as the FileStream connectors in Cloudera.

Kafka Connect connectors are distributed as:

- A directory of JAR files:

  The directory includes the JAR for the connector itself, as well as all its dependencies.
- An uber JAR/FAT JAR/JAR with dependencies file:

  This is a single JAR file that contains the connector, as well as its dependencies.

In Cloudera, all connectors that do not come prepackaged with Cloudera Runtime, as well as the FileStream connectors (FileStreamSourceConnector and FileStreamsSinkConnector), must be installed manually. This is done by making the connector JAR files available on all cluster hosts in a specific location. After installation is complete, you will be able to deploy the connector using the Streams Messaging Manager UI (recommended), Streams Messaging Manager API, or Kafka Connect API.

The location of the JAR files is determined by the Kafka Connect role's plugin.path property. Kafka Connect discovers connectors by looking at this directory path on the host machines. By default, the plugin.path property is set to /var/lib/kafka. This means that, by default, any connector placed in this directory will be discovered by Kafka Connect. Cloudera recommends that you use the default path.

The installation steps differ for custom developed connectors and the FileStream connectors. This is because the JAR file for the FileStream connectors is by default available on Cloudera cluster hosts. Additionally, FileStream connectors can be installed with an alternate installation method that involves the usage of an advanced configuration snippet.

Installing custom developed Kafka Connect connectors

Learn how to install custom developed (third party) connectors in Cloudera.

Before you begin

The following steps assume that plugin.path is set to /var/lib/kafka, which is the default path.

Important: Steps 1 and 2 must be carried out on all hosts in the cluster that have Kafka Connect roles deployed on them.

Procedure

1. Log in to a host that is running a Kafka Connect role.
2. Make the connector files available in or readable from /var/lib/kafka.

   How you choose to complete this step will largely depend on your cluster environment. For example:

   - You can download or copy the files directly to /var/lib/kafka.
   - You can choose to place connector files in a location different from /var/lib/kafka and create symlinks that point to the location where the connector files are available.

   Regardless of what method you choose, this step is considered complete once the JAR files are readable from /var/lib/kafka.
3. Restart all Kafka Connect roles:
   a) In Cloudera Manager, select the Kafka service.
   b) Go to Instances.
   c) Select all Kafka Connect instances by checking the checkbox next to each instance.
   d) Click Actions for selectedRestart.
   e) Click Restart to confirm.

      The roles are restarted once a **Finished** status is displayed.

Results

The connector is installed and is available for deployment. You are now able to deploy and manage the new connector from the Streams Messaging Manager UI.

What to do next

Deploy an instance of the connector using Streams Messaging Manager. For more information, see *Deploying and managing Kafka Connect connectors in Streams Messaging Manager*.

Related Information

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Installing FileStream connectors

Learn how to install the FileStream example connectors (FileStreamSourceConnector and FileStreamSinkConnector) that are shipped with Cloudera Runtime but are not installed by default. You can choose between two installation methods.

About this task

The JAR file for the FileStream connectors is shipped with Cloudera Runtime and is readily available on the cluster hosts. However, the file is not added to the Kafka Connect plugin.path directory by default. This is because the connectors are meant to be used for demonstrating the capabilities of Kafka Connect and are not production ready. As a result, even though these connectors do come packaged with Cloudera Runtime, they must be installed before they can be deployed. The JAR file is located at /opt/cloudera/parcels/CDH/jars/connect-file-*[***KAFKA COMPONENT VERSION***]*.jar.

You have two options when installing the FileStream connectors. You can install the connectors by copying or symlinking the JAR files to the plugin.path directory. Alternatively, you can add the location of the JAR file to the Kafka Connect role's CLASSPATH environment variable using an advanced configuration snippet.

The main difference between the two installation methods is that copying or symlinking the file requires that you log in to each Kafka Connect host in your cluster. Using an advanced configuration snippet, on the other hand, enables you to install the connector on all hosts by changing a single property in Cloudera Manager.

Although using an advanced configuration snippet is more convenient than copying or symlinking, be aware that setting an advanced configuration snippet is considered an advanced configuration practice. Therefore, Cloudera advises caution if you choose to install the FileStream connectors using an advanced configuration snippet.

Before you begin

- Go to Cloudera Runtime component versions and note down the component version of Apache Kafka. You need to specify the version during installation.

  The version is made up of three parts. It contains the upstream Apache Kafka version (first three digits), the Cloudera Runtime version (digits four to six), and the Cloudera Runtime build number (last three digits denominated with a dash). For example: 3.1.1.7.1.8.0-801.
- The following steps assume that plugin.path is set to /var/lib/kafka, which is the default path.

Procedure

1. Install the FileStream connectors:

   For Copy or symlink

   Complete the following actions on all Kafka Connect hosts.

   a. Log in to a host that is running a Kafka Connect role.
   b. Copy or symlink the JAR file of the connectors to /var/lib/kafka.
      - To copy the file directly to /var/lib/kafka use the following command:

        ```
        cp /opt/cloudera/parcels/CDH/jars/connect-file-[***KAFKA COMPONENT VERSION***].jar /var/lib/kafka/
        ```

      - To create a symlink in /var/lib/kafka that points to /opt/cloudera/parcels/CDH/jars/connect-file-*[***KAFKA COMPONENT VERSION***]*.jar use the following command:

        ```
        ln -sf /opt/cloudera/parcels/CDH/jars/connect-file-[***KAFKA COMPONENT VERSION***].jar /var/lib/kafka/connect-file-[***KAFKA COMPONENT VERSION***].jar
        ```

   Replace *[***KAFKA COMPONENT VERSION***]* with the component version of Apache Kafka. Regardless of what method you choose, this step is considered complete once the JAR file is readable from /var/lib/kafka.

   For Using an advanced configuration snippet

   a. In Cloudera Manager, select the Kafka service.
   b. Go to Configuration.
   c. Find the Kafka Connect Environment Advanced Configuration Snippet (Safety Valve) property.
   d. Click ✚ to add a property and enter the following key and value pair:
      - Key: CLASSPATH
      - Value: /opt/cloudera/parcels/CDH/jars/connect-file-*[***KAFKA COMPONENT VERSION***]*.jar

      Replace *[***KAFKA COMPONENT VERSION***]* with the component version of Apache Kafka.
   e. Click Save Changes.

2. Restart all Kafka Connect roles:
   a) In Cloudera Manager, select the Kafka service.
   b) Go to Instances.
   c) Select all Kafka Connect instances by checking the checkbox next to each instance.
   d) Click Actions for selectedRestart.
   e) Click Restart to confirm.

   The roles are restarted once a **Finished** status is displayed.

Results

The FileStream connectors are installed and available for deployment. You are now able to deploy and manage the FileStream connectors from the Streams Messaging Manager UI.

What to do next

Deploy an instance of the FileStream Source or FileStream Sink connector using Streams Messaging Manager. For more information, see *Deploying and managing Kafka Connect connectors in Streams Messaging Manager*.

Related Information

Setting an Advanced Configuration Snippet for a Cloudera Runtime Service

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Setting up the Debezium Db2 Source connector

Learn about the Cloudera specific setup steps required before you can deploy the Debezium Db2 Source connector.

About this task

In Cloudera, before deploying an instance of the Debezium Db2 Source connector, you must download and deploy the Db2 JDBC driver on all Kafka Connect hosts. Otherwise, you will not be able to deploy the connector. The following list of steps walks you through this process.

For more information regarding how the Debezium DB2 connector works as well as its configuration and properties, see *Debezium connector for Db2* in the Debezium documentation.

Important: In addition to the following configuration steps, appropriately configuring your Db2 tables is also required. Ensure that you review *Setting up Db2* in the Debezium documentation and complete all setup tasks either before or after completing the following steps.

Procedure

1. Download the Db2 JDBC driver.

   The JDBC driver can be downloaded from *Maven Central*. Cloudera recommends that you use version 11.5.0.0.
2. Copy and rename the driver to the location specified in the JDBC Driver Jars Path For Debezium Kafka property.

   The default location is /var/lib/kafka_connect_jdbc. The JDBC driver must be renamed to db2-connector-java.jar. For example:

   ```
   cp ./jcc-[***VERSION***].jar var/lib/kafka_connect_jdbc/db2-connector-ja
   va.jar
   ```
3. Set correct file permissions. For example:

   ```
   chmod 644 /var/lib/kafka_connect_jdbc/db2-connector-java.jar
   ```
4. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The Db2 JDBC driver is deployed on all Kafka Connect hosts. Deploying an instance of the Debezium Db2 Source connector in your cluster is now possible.

What to do next

Deploy an instance of the Debezium Db2 Source connector. Additionally, review *Debezium connector for Db2* in the Debezium documentation to learn more about the connector and its features.

Related Information

Debezium connector for Db2

Setting up Db2

Maven central

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Setting up the Debezium MySQL Source connector

Learn about the Cloudera specific setup steps required before you can deploy the Debezium MySQL Source connector.

About this task

In Cloudera, before deploying an instance of the Debezium MySQL Source connector, you must download and deploy the MySQL JDBC driver on all Kafka Connect hosts. Otherwise, you will not be able to deploy the connector. The following list of steps walks you through this process.

For more information regarding how the Debezium MySQL connector works as well as its configuration and properties, see the *Debezium connector for MySQL* in the Debezium documentation.

Important: In addition to the following configuration steps, appropriately configuring your MySQL server is also required. Ensure that you review *Setting up MySQL* in the Debezium documentation and complete all setup tasks either before or after completing the following steps.

Procedure

1. Download the MySQL JDBC driver.

   The JDBC driver can be downloaded from the *MySQL Product Archives* page. Download the platform independent version. Cloudera recommends that you use version 8.0.28.
2. Extract the files from the downloaded archive.

   You can use the tar command or any other archive manager to extract the archive. For example:

   ```
   tar -xf mysql-connector-java-[***VERSION***].tar.gz
   ```
3. Navigate to the directory where the driver was extracted.

   ```
   cd mysql-connector-java-[***VERSION***]
   ```
4. Copy and rename the driver to the location specified in the JDBC Driver Jars Path For Debezium Kafka property.

   The default location is /var/lib/kafka_connect_jdbc. The JDBC driver must be renamed to mysql-connector-java .jar. For example:

   ```
   cp ./mysql-connector-java-[***VERSION***].jar /var/lib/kafka_connect_jdbc/
   mysql-connector-java.jar
   ```
5. Set correct file permissions. For example:

   ```
   chmod 644 /var/lib/kafka_connect_jdbc/mysql-connector-java.jar
   ```
6. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The MySQL JDBC driver is deployed on all Kafka Connect hosts. Deploying an instance of the Debezium MySQL Source connector in your cluster is now possible.

What to do next

Deploy an instance of the Debezium MySQL Source connector. Additionally, review *Debezium connector for MySQL* in the Debezium documentation to learn more about the connector and its features.

Related Information

Debezium connector for MySQL

Setting up MySQL

MySQL Product Archives

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Setting up the Debezium Oracle Source connector

Learn about the Cloudera specific setup steps required before you can deploy the Debezium Oracle Source connector.

About this task

In Cloudera, before deploying an instance of the Debezium Oracle Source connector, you must download and deploy the Oracle JDBC driver on all Kafka Connect hosts. Otherwise, you will not be able to deploy the connector. The following list of steps walks you through this process.

For more information regarding how the Debezium Oracle Source connector works as well as its configuration and properties, see the *Debezium Connector for Oracle* in the Debezium documentation.

Note: In addition to the following configuration steps, appropriately configuring your Oracle server is also required. Ensure that you review the *Setting up Oracle* in the Debezium documentation and complete all setup tasks either before or after completing the following steps.

Procedure

1. Download the Oracle JDBC driver.

   The JDBC driver can be downloaded from *Maven Central*. Cloudera recommends that you use version 12.2.0.1, 19.8.0.0, or 21.1.0.0.
2. Copy and rename the driver to the location specified in the JDBC Driver Jars Path For Debezium Kafka property.
3. The default location is /var/lib/kafka_connect_jdbc. The JDBC driver must be renamed to oracle-connector-jav a.jar. For example:

   ```
   cp ./ojdbc8-[***VERSION***].jar /var/lib/kafka_connect_jdbc/oracle-connect
   or-java.jar
   ```
4. Set correct file permissions. For example:

   ```
   chmod 644 /var/lib/kafka_connect_jdbc/oracle-connector-java.jar
   ```
5. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The Oracle JDBC driver is deployed on all Kafka Connect hosts. Deploying an instance of the Debezium Oracle Source connector in your cluster is now possible.

What to do next

Deploy an instance of the Debezium Oracle Source connector. Additionally, review *Debezium Connector for Oracle* in the Debezium documentation to learn more about the connector and its features.

Related Information

Debezium Connector for Oracle

Setting up Oracle

Maven Central

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Setting up the Debezium PostgreSQL Source connector

Learn about the Cloudera specific setup steps required before you can deploy the Debezium PostgreSQL Source connector.

About this task

In Cloudera, before deploying an instance of the Debezium PostgreSQL Source connector, you must download and deploy the PostgreSQL JDBC driver on all Kafka Connect hosts. Otherwise, you will not be able to deploy the connector. The following list of steps walks you through this process.

For more information regarding how the Debezium PostgreSQL connector works as well as its configuration and properties, see the *Debezium connector for PostgreSQL* in the Debezium documentation.

Important: In addition to the following configuration steps, appropriately configuring your PostgreSQL server is also required. Ensure that you review the *Set up* section in the Debezium documentation for the PostgreSQL connector and complete all setup tasks either before or after completing the following steps.

Procedure

1. Download the PostgreSQL JDBC driver.

   The JDBC driver can be downloaded from *Maven Central*. Cloudera recommends that you use version 42.3.5.
2. Copy and rename the driver to the location specified in the JDBC Driver Jars Path For Debezium Kafka property.

   The default location is /var/lib/kafka_connect_jdbc. The JDBC driver must be renamed to postgresql-connector-java.jar. For example:

   ```
   cp ./postgresql-[***VERSION***].jar /var/lib/kafka_connect_jdbc/postgre
   sql-connector-java.jar
   ```
3. Set correct file permissions. For example:

   ```
   chmod 644 /var/lib/kafka_connect_jdbc/postgresql-connector-java.jar
   ```
4. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The PostgreSQL JDBC driver is deployed on all Kafka Connect hosts. Deploying instance of the Debezium PostgreSQL Source connector in your cluster is now possible

What to do next

Deploy an instance of the Debezium PostgreSQL Source connector. Additionally, review *Debezium connector for PostgreSQL* in the Debezium documentation to learn more about the connector and its features.

Related Information

Debezium connector for PostgreSQL

Set up (PostgreSQL connector)

Maven Central

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Setting up the Debezium SQL Server Source connector

Learn about the Cloudera specific setup steps required before you can deploy the Debezium SQL Server Source connector.

About this task

In Cloudera, before deploying an instance of the Debezium SQL Server Source connector, you must download and deploy the SQL Server JDBC driver on all Kafka Connect hosts. Otherwise, you will not be able to deploy the connector. The following list of steps walks you through this process.

For more information regarding how the Debezium PostgreSQL connector works as well as its configuration and properties, see the *Debezium connector for SQL Server* in the Debezium documentation.

Important: In addition to the following configuration steps, appropriately configuring your SQL Server database is also required. Ensure that you review the *Setting up SQL Server* in the Debezium documentation and complete all setup tasks either before or after completing the following steps.

Procedure

1. Download the SQL Server JDBC driver.

   The JDBC driver can be downloaded from the *mssql-jdbc Github repository*. Cloudera recommends that you use version 9.4.1.jre8.

2. Copy and rename the driver to the location specified in the JDBC Driver Jars Path For Debezium Kafka property.

   The default location is /var/lib/kafka_connect_jdbc. The JDBC driver must be renamed to mssql-connector-java .jar. For example:

   ```
   cp ./mssql-jdbc-[***VERSION***].jre8.jar /var/lib/kafka_connect_jdbc/ms
   sql-connector-java.jar
   ```

3. Set correct file permissions. For example:

   ```
   chmod 644 /var/lib/kafka_connect_jdbc/mssql-connector-java.jar
   ```

4. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The SQL Server JDBC JDBC driver is deployed on all Kafka Connect hosts. Deploying instance of the Debezium SQL Server Source connector in your cluster is now possible.

What to do next

Deploy an instance of the Debezium SQL Server Source connector. Additionally, review *Debezium connector for SQL Server* in the Debezium documentation to learn more about the connector and its features.

Related Information

Debezium connector for SQL Server

Setting up SQL Server

mssql-jdbc GitHub repository

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

HTTP Source connector

The HTTP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The HTTP Source connector listens on a port for HTTP POST requests and transfers the request body to a Kafka topic. The Kafka topic this connector transfers messages to is determined by the value of the topics property. The connector does not perform record processing, the messages are transferred to Kafka as they are received in the HTTP request. The connector uses HTTPS for secure communication. Optionally, client certificate authentication (mutual TLS) can also be configured.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Source property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *HTTP Source properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- The HTTP Source connector must use at least one-way SSL, it cannot be used without SSL.
- The ListenHTTP processor in the NiFi flow keeps running when the connector is paused. Due to this, the connector accepts requests, but ultimately times out and sends an error response to the client.
- The tasks.max property must be 1 to avoid port conflict due to executing multiple tasks on the same Kafka Connect node. Cloudera recommends that you use a load balancer to keep track of the actual node which runs the connector.

Configuration example

In this example, the connector accepts client requests sent to the connector’s URL. The connector's URL is based on the configuration of multiple properties. For example assume the following:

- The connector is running on kc-host-1.
- Listening Port is set to port 8080.
- Base Path is set to data-ingest.

In a case like this, the connector's URL will be the following:

```
https://kc-host-1:8080/data-ingest
```

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case and therefore can be removed from the configuration JSON.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "HTTP Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.HTTP Source Parameters:Listening Port": "[***PORT***]",
 "parameter.HTTP Source Parameters:Base Path": "[***BASE PATH***]",
 "parameter.HTTP Source Parameters:Keystore Filename": "[***PATH TO
 KEYSTORE***]",
```

```
 "parameter.HTTP Source Parameters:Keystore Key Password": "[***KEYSTORE KEY
 PASSWORD***]",
 "parameter.HTTP Source Parameters:Keystore Password": "[***KEYSTORE
 PASSWORD***]",
 "parameter.HTTP Source Parameters:Keystore Type": "[***KEYSTORE TYPE***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

Listening Port

The port that the client needs to use in the URL.

Base Path

The path that the client needs to use in the URL.

Keystore *

The properties for accessing the keystore used for HTTPS.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

HTTP Source properties reference

JDBC Source connector

The JDBC Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The JDBC Source connector reads records from a database table and transfers each record to Kafka in Avro or JSON format. The Kafka topic this connector transfers messages to is determined by the value of the topics parameter in the configuration. The schema of the records can be inherited from the schema of the database table, or it can be a predefined schema retrieved from Schema Registry.

The connector supports incremental loading based on a table column containing increasing values (typically an ID sequence or timestamp). When the connector is started for the first time with incremental loading turned on, it can load all the existing data from the source table, or it can start from the current maximum value of the incrementing column.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Source property reference*.

Connector/dataflow-specific properties

These are the properties that are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *JDBC Source properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- The tasks.max property must be set to 1. Setting the property to any other value will not have an effect. This is because this connector performs incremental loading and must run as a single task.
- If Initial Load Strategy is set to Start at Beginning and “Maximum-value Columns is not set, the connector tries to load the entire content of the source database table to the target Kafka topic with every execution of the connector. Set Maximum-value Columns to avoid this.

Configuration example

In this example, the connector connects to a PostgreSQL database using username/password authentication, queries the records from a table, converts each record to JSON format, and sends them to a Kafka topic. The schema of the records is determined from the schema of the database table. The data loading is incremental, only new records are loaded based on a column containing incrementing values in the table.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "JDBC Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "key.attribute": "kafka.message.key",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.JDBC Source Parameters:Database Connection URL": "[***JDBC
 URL***]",
 "parameter.JDBC Source Parameters:Database Driver Location": "[***PATH TO
 JDBC DRIVER***]",
 "parameter.JDBC Source Parameters:Database Driver Class Name": "org.post
gresql.Driver",
 "parameter.JDBC Source Parameters:Database Type": "PostgreSQL",
 "parameter.JDBC Source Parameters:Database User Name": "[***USERNAME***]",
```

```
  "parameter.JDBC Source Parameters:Database User Password":
  "[***PASSWORD***]",
  "parameter.JDBC Source Parameters:Database Table Name": "[***TABLE
  NAME***]",
  "parameter.JDBC Source Parameters:Kafka Message Key Column": "[***COLUMN
  NAME***]",
  "parameter.JDBC Source Parameters:Maximum-value Columns": "[***COLUMN
  NAME***]",
  "parameter.JDBC Source Parameters:Kafka Message Data Format": "JSON",
  "parameter.JDBC Source Parameters:Schema Access Strategy": "Inherit Schema"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

Database Connection URL

The JDBC URL of the PostgreSQL database. For example, jdbc:postgresql://myhost:5432/mydb.

Database Driver Location

A comma-separated list of files or folders containing the JDBC client libraries.

Database Driver Class Name

The Java class name of the PostgreSQL Driver implementation.

Database Type

The type of the database. Because this is a PostgreSQL example, the property is set to PostgreSQL.

Database User Name

The username used for authenticating to the database.

Database User Password

The password of the user.

Database Table Name

The name of the database table to query data from.

Kafka Message Key Column

Specifies a database table column. The value of the column specified is used as the key of the Kafka message.

Maximum-value Columns

The name of the column used for incremental loading. The column value must be increasing, for example, coming from a sequence or system timestamp.

Kafka Message Data Format

The format of the messages the connector sends to Kafka.

Schema Access Strategy

Specifies the strategy used for determining the schema of the database record. In this example, this property is set to Inherit Schema, meaning that the schema is determined (inherited) from the database table.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

JDBC Source properties reference

JMS Source connector

The JMS Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The JMS Source Connector consumes messages from a JMS broker and transfers the message body to Kafka. The Kafka topic this connector transfers messages to is determined by the value of the topics parameter in the configuration. The JMS Source connector does not perform record processing, the messages are transferred to Kafka as they are consumed from the JMS broker.

TLS can be used to establish a secure connection between the connector and the JMS broker. The keystore and truststore files necessary for securing the connection must be present on the cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Source property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *JMS Source properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.

Configuration example

In this example, the connector connects to an ActiveMQ Message Broker using a secured connection and username/password authentication.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
```

```
 "meta.smm.predefined.flow.name": "JMS Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter"
,,
 "tasks.max": "1"
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",,
 "topics": "[***TOPIC NAME***]"
 "parameter.JMS Source Parameters:JMS Broker URI": "[***BROKER URI***]",
 "parameter.JMS Source Parameters:JMS Client Libraries": "[***PATH TO CLIENT
 LIBRARIES***]",
 "parameter.JMS Source Parameters:JMS Connection Factory Class Name": "or
g.apache.activemq.ActiveMQSslConnectionFactory",
 "parameter.JMS Source Parameters:JMS Destination Name": "[***QUEUE
 NAME***]",
 "parameter.JMS Source Parameters:JMS Destination Type": "QUEUE",
 "parameter.JMS Source Parameters:JMS User Name": "[***USERNAME***]",
 "parameter.JMS Source Parameters:JMS User Password": "[***PASSWORD***]",
 "parameter.JMS Source Parameters:Truststore Filename": "[***PATH TO
 TRUSTSTORE***]",
 "parameter.JMS Source Parameters:Truststore Password": "[***TRUSTSTORE
 PASSWORD***]",
 "parameter.JMS Source Parameters:Truststore Type": "[***TRUSTSTORE
 TYPE***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

JMS Broker URI

The URI of the ActiveMQ Message Broker. For example, ssl://activemq-hostname:61617.

JMS Client Libraries

The path to the directory containing ActiveMQ client libraries.

JMS Connection Factory Class Name

The Java class name of the ActiveMQ ConnectionFactory implementation.

JMS Destination Name

The name of the destination (in this case a queue) on the ActiveMQ Message Broker that the messages are received from.

JMS Destination Type

The type of the destination. In this example, the type is QUEUE.

JMS User Name

The username used for authenticating to the ActiveMQ Message Broker.

JMS User Password

The password of the user.

Truststore *

These are the properties for accessing the truststore containing the ActiveMQ Message Broker's certificate used for secure communication.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

JMS Source properties reference

MQTT Source connector

The MQTT Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The MQTT Source connector consumes messages from an MQTT broker and transfers them to Kafka. The Kafka topic this connector transfers messages to is determined by the value of the topics property in the configuration. This connector does not perform record processing. The messages are transferred to Kafka as they are consumed from the MQTT broker. TLS can be used to establish a secure connection between the connector and the MQTT broker. The keystore and truststore files necessary for securing the connection must be present on the cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Source property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *MQTT Source properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- If the keystore-related properties are removed and an empty truststore is provided, the connector does not use TLS for connecting to the MQTT broker. TLS is used if a truststore is provided that has a key in it.
- When the connector is deployed in a cluster environment, the system might move it from one cluster node to another. During this process, the connector is stopped, deployed to the target node, and started there. The time between the connector stopping and starting again might be several minutes. To avoid losing the messages that arrived to the MQTT broker during this time, set the following properties:
  - Set MQTT Clean Session to false.
  - Specify the MQTT Client ID.
  - Set MQTT Quality of Service to 1 or 2.

Configuration example

In this example, the connector fetches messages from an MQTT broker and transfers them to a Kafka topic.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "MQTT Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.MQTT Source Parameters:MQTT Broker URI": "tcp
://[***HOST***]:[***PORT***]",
 "parameter.MQTT Source Parameters:MQTT Quality of Service": "0",
 "parameter.MQTT Source Parameters:MQTT Topics": "[***MQTT TOPIC NAME***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case:

topics

The name of the Kafka topic that the connector sends messages to.

MQTT Broker URI

The URI of the MQTT broker. In this example, the connection is not secure. As a result, the URI starts with tcp://. The port number you typically use in a case like this is 1883. If TLS is used, the URI should start with ssl://, and typically you use port number 88883.

MQTT Topics

The comma-separated list of MQTT topics that the connector fetches messages from.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

MQTT Source properties reference

SFTP Source connector

The SFTP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The SFTP Source connector obtains files from an SFTP server and transfers them to Kafka. The SFTP Source connector supports record processing. In case record processing is enabled, input file is split into records and the records are transferred to Kafka. If record processing is not enabled, the connector forwards files to Kafka as they are fetched from the SFTP server.

If record processing is enabled, the input file can contain records in either JSON, CSV or Grok format. If the input records are in JSON or CSV format, the connector can either infer the schema of the records based on the data or read the schema from Schema Registry. If the input records are in Grok format, the connector can either derive the schema using the field names from the value of the Grok Expression property or read the schema from Schema Registry.

In addition, if record processing is enabled, the connector writes the records to Kafka in Avro format. How the record schema is attached to these Avro messages depends on the value of the Avro Schema Write Strategy property.

If Schema Registry is used, and it is on a Kerberized cluster, the krb5.file property must point to the krb5.conf file that provides access to the cluster on which Schema Registry is present. This means that the krb5.conf file must be on the same cluster node that the connector runs on. The Kerberos keytab used to access Schema Registry must also be on the same cluster node that the connector runs on. The connection to Schema Registry can be secured by TLS. The truststore file necessary for securing the connection must also be on the same cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Source property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *SFTP Source properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- The Schema Registry URL property is mandatory even if Schema Registry is not used. If Schema Registry is not used, use the default value, or completely remove the property from the configuration JSON.
- Schema Branch and Schema Version can not be specified at the same time.
- The value of the Schema Access Strategy property is not independent of the value of the Input Data Format property. As a result, you must exercise caution when configuring Schema Access Strategy.
  - If the value of Input Data Format is JSON, the possible values for Schema Access Strategy are Schema Registry or Infer Schema.
  - If the value of Input Data Format is CSV, the possible values for Schema Access Strategy are Schema Registry or Infer Schema.
  - If the value of Input Data Format is GROK, the possible values for Schema Access Strategy are Schema Registry or Field Names From Grok Expression.
- If the Enable Record Processing property is set to false (record processing disabled), the entire input file is transferred to Kafka as one message.
- If the Enable Record Processing property is set to true (record processing enabled), the output is always in Avro format. How the record schema is attached to these Avro messages depends on the value of the Avro Schema Write Strategy property.

Configuration example

In this example, the connector fetches files from an SFTP server that only requires Basic Authentication (username and password) and transfers these files to a Kafka topic.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "SFTP Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[*** KAFKA TOPIC NAME***]",
 "parameter.SFTP Source Parameters:Hostname": "[***SFTP SERVER
 HOSTNAME***]",
 "parameter.SFTP Source Parameters:Port": "22",
 "parameter.SFTP Source Parameters:Username": "[***SFTP USERNAME***]",
 "parameter.SFTP Source Parameters:Password": "[***SFTP PASSWORD***]",
 "parameter.SFTP Source Parameters:Remote Path": "[***FOLDER PATH***]",
 "parameter.SFTP Source Parameters:File Filter Regex": ".*",
 "parameter.SFTP Source Parameters:Enable Record Processing": "false"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

Hostname

The hostname of the remote system where the SFTP server runs. For example, my.sftp-server.com.

Port

The port that the remote system is listening on for file transfers.

Username

The username for connecting to the SFTP server.

Password

The password used to authenticate the user towards the SFTP server. In this example, the SFTP server requires Basic Authentication.

Remote Path

The path on the remote system that points to the directory from which to pull files. For example, /uploads.

File Filter Regex

The Java regular expression to use for filtering filenames. Only files whose names match the regular expression are fetched.

Enable Record Processing

Determines whether the contents of a file get parsed as records before sending them to Kafka. In this example, the property is set to false, meaning that the entire file gets forwarded to Kafka as one message.

Related Information

Apache Kafka documentation
Stateless NiFi Source properties reference
SFTP Source properties reference

Stateless NiFi Source and Sink

The Stateless NiFi Source and Sink connectors allow you to run NiFi dataflows within Kafka Connect. Using these connectors can grant you access to a number of NiFi features without having the need to deploy or maintain NiFi on your cluster. Learn more about what Stateless NiFi is and how it shipped in Cloudera.

Apache NiFi is a powerful tool for authoring and running dataflows. It provides many capabilities that are necessary for large-scale enterprise deployments, such as data persistence and resilience, data lineage and traceability, as well as multi-tenancy. This, however, requires an administrator who ensures that this process is running and operational. Additionally, in most cases, adding more capabilities to your deployments results in more complexity.

There are times, however, when users do not need all the power of NiFi, and running it in a much simpler form factor is sufficient. A common use case is to use NiFi to pull data from many different sources, perform manipulations (for example, convert JSON to Avro), filter some records, and then publish the data to Apache Kafka. Another common use case is to pull data from Apache Kafka, perform manipulations and filtering, and then publish the data elsewhere.

For deployments where NiFi acts only as a bridge into and out of Kafka, it can be simpler to operationalize such a deployment by running the dataflow within Kafka Connect. The Stateless NiFi Source and Sink connectors allow users to do just that.

Stateless NiFi

Dataflows within Kafka Connect run using the Stateless NiFi dataflow engine. Stateless NiFi differs from the traditional NiFi engine in the following ways:

- Stateless NiFi is an engine that is designed to be embedded. This makes it convenient to run it within the Kafka Connect framework.
- Stateless NiFi does not provide a user interface (UI) or a REST API. Additionally, it does not support modifying the dataflow while it is running.
- Stateless NiFi does not persist flowfile content to disk. Instead, it holds the content in memory.
- Stateless NiFi does not use data prioritizers. Instead, it operates on data in a first-in-first-out order. Dataflows built for Stateless NiFi must have a single source and a single destination. The only exception to this is when the data is routed to exactly one of multiple destinations, such as a failure destination or a success destination.
- Stateless NiFi does not currently provide access to data lineage/provenance.
- Stateless NiFi does not support cyclic graphs. While it is common and desirable in traditional NiFi to have a failure relationship from a processor route back to the same processor, this can result in a `StackOverflowException` in Stateless NiFi. The preferred approach in Stateless NiFi is to create an output port for failures and route the data to that output port.

Stateless NiFi in Cloudera

Stateless NiFi in Cloudera comes in two flavors. You have the base Stateless NiFi Source (`StatelessNiFiSource`) and Stateless NiFi Sink (`StatelessNiFiSink`) connectors. In addition, there are a number of ready-to-use connectors based on Stateless NiFi Source and Sink.

Stateless NiFi Source and Sink connectors

The Stateless NiFi Source and Sink connectors are generic connectors that act as the base for Stateless NiFi in Kafka Connect. The idea behind these connectors is that you develop your own dataflow in NiFi, and deploy that dataflow as a Kafka Connect connector using either Stateless Nifi Source or Sink connectors. This way, you can create virtually any connector without having to write a single line of code. All of the heavy lifting is done on the NiFi UI.

Predefined dataflows and Stateless NiFi-based connectors

In addition to the base Stateless NiFi Source and Sink connectors, there are a number of readily-available connectors that are based on Stateless NiFi. Specifically, Cloudera ships a set of predefined dataflows. These dataflows are developed and maintained by Cloudera. Each dataflow covers a common data movement use case. These predefined flows are presented as standalone connectors on the Streams Messaging Manager UI. Using these dataflows/connectors, you can immediately start using Stateless NiFi without having to develop your own dataflow.

You can find a full list of the Stateless NiFi-based connectors in Connectors on page 40. In addition, you can also use the Streams Messaging Manager UI to check which of the connectors are Stateless NiFi-based. When adding a new connector using Streams Messaging Manager, you are presented with a number of cards that represents the connectors available for deployment. Each card includes the following information about the connector:

- The connector's fully qualified class name.
- The connector's display name. If no display name is available, the card includes the unqualified classname.
- The version of the connector.

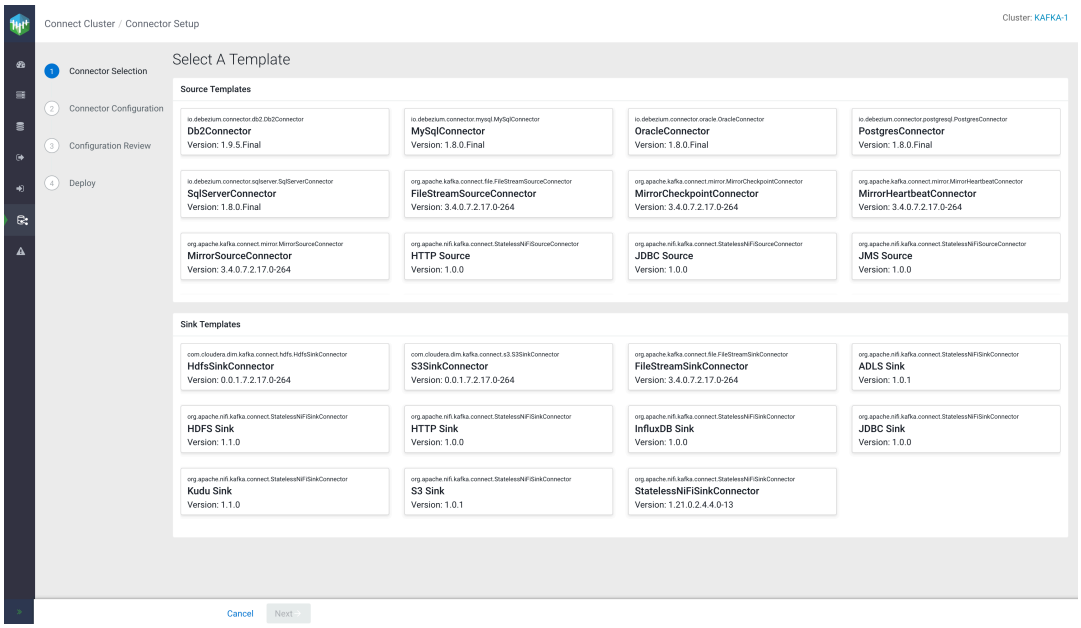

Notice how the fully qualified class name for many of the default available connectors is either `org.apache.nifi.kafka.connect.StatelessNiFiSourceConnector` or `org.apache.nifi.kafka.connect.StatelessNiFiSinkConnector`. Even though these are standalone connectors and have a unique display name, they are all Stateless NiFi connectors. Each of them runs a predefined flow designed for a specific use case. For example, JDBC Source runs a dataflow that reads records from a database table and transfers each record to Kafka.

Note: Unlike the base Stateless NiFi connectors, connectors running predefined flows **do not** accept a custom dataflow with flow.snapshot. The dataflow they use is specified by the meta.smm.predefined.flow.name and meta.smm.predefined.flow .version properties. While configuring these properties is possible, Cloudera does not recommend that you do so.

Dataflow development best practices for Stateless NiFi

By leveraging Stateless NiFi, you can build your own dataflows and use those as Kafka Connect connectors. However, there are some principles to keep in mind when building such a dataflow in NiFi.

General recommendations and criteria

Learn about the general recommendations and criteria that you need to follow when building a dataflow that you will run as a Kafka Connect connector.

- If your dataflow depends on external resources (like a JDBC driver), the resource must be made available on all Kafka Connect role instance nodes (Connect workers). Additionally, the resource must be in the same location on all nodes with proper file permissions. Cloudera also recommends parameterizing the file path so that you can specify it when deploying the connector.
- The dataflow should be designed in its dedicated process group and should have a parameter context assigned to its process group. This way, you have the ability to parameterize your connector, which enables you to provide custom property values when deploying the connector using Streams Messaging Manager.
- A source connector flow should always have a single source. This means that the flow should have only one processor with no incoming connections.
- A sink connector flow should always have a single destination. If you want to transfer the data consumed from the Kafka topic to two different destinations at the same time, it is preferred to have two distinct sink connectors with different consumer groups. If using a single sink connector, you must take into consideration the possibility of creating duplicates. If one destination is working as expected but not the other, the Kafka message is not acknowledged and consumed again at a later time while the data was still successfully sent to one destination.

Dataflow execution and scheduling

Learn how Stateless NiFi Source and Sink connectors handle dataflow execution and scheduling.

Stateless NiFi does not consider the scheduling settings of processors. When a Stateless NiFi connector is deployed, Kafka Connect starts an instance of the Stateless NiFi engine that executes the dataflow. The following sections discuss in detail how the dataflow is executed by both source and sink connectors

Source connectors

A Kafka Connect source connector obtains data from an external (source) data store and delivers the data to a Kafka service. The architecture of a source connector is as follows.

Figure 3: Source Connector Architecture

When a Stateless NiFi Source connector instance is deployed, a number of tasks are started for the connector instance. The number of tasks is determined by the tasks.max property, which can be set during connector deployment. Each task consists of the following:

- `SourceTask`

  In Stateless NiFi connectors, the `SourceTask` is responsible for executing one instance of the connector's dataflow.
- SMT Chain

  The Single Message Transforms (SMT) chain can modify and filter the data on a single message basis.
- Converter

  The converter is responsible for converting Kafka message keys and values with a configured conversion mechanism. However, Stateless NiFi connectors currently only use keys of string type and values of byte array type. As a result, the converter does not perform any meaningful conversion.
- Kafka Producer

  The Kafka producer is responsible for delivering messages to the configured Kafka topics.

With the Stateless NiFi Source connector, the `SourceTask` is the component executing an instance of the dataflow specified in the connector's flow.snapshot property. Any source connector flow you build should not attempt to deliver directly to Kafka itself using a processor such as PublishKafka. Instead, the data should be routed to a NiFi output port in the dataflow. Any flowfile delivered to that output port is obtained by the connector and delivered to Kafka.

The dataflow is triggered continuously in one thread within the `SourceTask`. This means that in one iteration, the entire flow is executed (one processor after another), and the data it fetches from the source data store in that execution ends up on the flow’s output port in the form of flowfiles. These flowfiles are forwarded by the Kafka

Connect framework to the task's producer as messages (one flowfile equals one message). At this point the flow's execution is finished, and the flow is triggered again.

The producer is a regular Kafka producer that is responsible for delivering the messages to the Kafka topics. The producer is running on a separate thread. This means that while the dataflow is running, the producer can work on delivering the messages it received at the end of the flow's previous execution.

Each flowfile is delivered as a single Kafka message. Therefore, if a flowfile contains thousands of JSON records, totaling 5 MB for example, it is important to split those flowfiles into individual records using a SplitRecord processor before transferring the data to the output port. Otherwise, this results in a single Kafka message that is 5 MB while the default message size for Kafka is 1 MB.

Scheduling settings of the processors in the dataflow are ignored. The dataflow is executed in one thread from the first processor through the output port. Afterward, the flow is retriggered. One execution of the dataflow takes as long as necessary to fetch and process the data that accumulated in the source data store since the previous execution. However, the execution has a default timeout of one minute. This timeout can be configured on a connector level with the dataflow.timeout property.

Transactional sources

Unlike with traditional NiFi, Stateless NiFi keeps the contents of flowfiles in memory. As long as the source of the data is replayable and transactional, there is no concern over data loss. This is handled by treating the entire dataflow as a single transaction. Once data is obtained from a source component, the data is then transferred to the next processor in the flow. At this point, in traditional NiFi, the processor would acknowledge the data and NiFi would take ownership of that data, having persisted it to disk.

With Stateless NiFi, however, the data is not yet acknowledged. Instead, data is transferred from one processor to the next with each processor performing and completing its task. This is repeated until all data is queued up at the output port. At this point, the NiFi connector provides the data to Kafka. Only after Kafka has acknowledged the records does the connector acknowledge this to NiFi. Only at that point is the session committed, allowing the source processor to acknowledge receipt of the data.

As a result, if NiFi is restarted while processing some piece of data, the source processor will not have acknowledged the data and is able to replay the data, resulting in no data loss.

Initial data load

Stateless NiFi connectors are intended to run continuously. This means that the dataflow is retriggered immediately after a single execution finishes. This also means that the dataflow is only supposed to process as many messages at a time that accumulated in the source data store since the previous execution.

Depending on the connector, if there is already a significant amount of data in the source data store and the connector is not configured properly, you might run into problems with the dataflow's initial execution. In such a case, the dataflow tries to fetch all the data from the source data store during the initial execution. If there is too much data, one of the following happens:

- No data appears in the target Kafka topics, and the log file includes an `InterruptedException` entry.

  This means that loading and processing the data took more time than the timeout configured in dataflow.timeout.
- Not data appears in the target Kafka topics, and the connector throws an `OutOfMemory` error.

  This can happen because the Stateless NiFi engine, which runs the dataflow, stores all data in memory. Trying to read too much data can exhaust the available memory capacity.

If you need to handle a large initial data load, Cloudera recommends that you process and transfer the initial load using another tool, for example regular NiFi. Afterward, you can configure and start the Kafka connector to handle subsequent changes in the source system.

Sink connectors

A Kafka Connect sink connector obtains data from a Kafka service and delivers that data to an external (sink) data store. The architecture of a sink connector is as follows.

Figure 4: Sink Connector Architecture

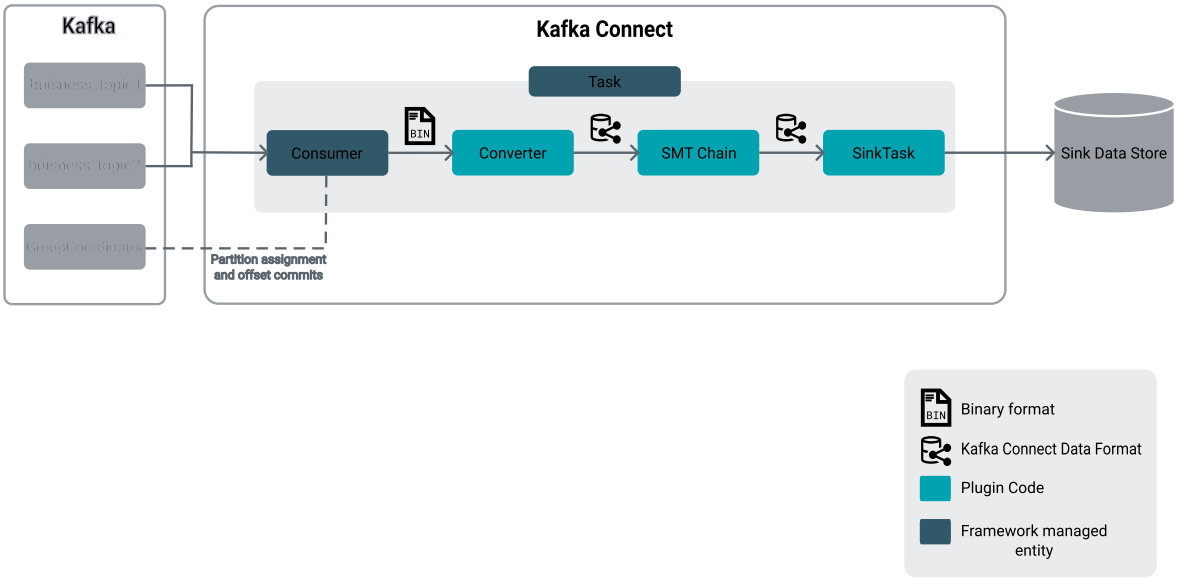

When a Stateless NiFi Sink connector instance is deployed, a number of tasks are started for the connector instance. The number of tasks is determined by the tasks.max property, which can be set during connector deployment. Each task consists of the following:

- Kafka Consumer

  The consumer is responsible for fetching the messages from the Kafka topics and delivering them to the converter.
- SMT Chain

  The Single Message Transforms (SMT) chain can modify and filter the data on a single message basis.
- Converter

  The converter is responsible for converting Kafka message keys and values with a configured conversion mechanism. However, Stateless NiFi connectors currently only use keys of string type and values of byte array type. As a result, the converter does not perform any meaningful conversion.
- `SinkTask`

  In Stateless NiFi connectors, the `SinkTask` is responsible for executing one instance of the connector's dataflow.

The consumer is a regular Kafka consumer reading messages from the configured topics and delivering them to the `SinkTask`. Messages are delivered through the converter and the SMT chain.

In a Stateless NiFi Sink connector, the `SinkTask` is the one executing an instance of the dataflow configured in the connector’s flow.snapshot property. Any sink connector flow you build should not attempt to source data directly from Kafka using a processor such as ConsumeKafka. Instead, the data should be received from an input port within the dataflow. Each Kafka record is queued into the outbound connection of the input port so that the next processor in the flow is able to process it. The entire task (consumer + `SinkTask`) is executed in one thread. There are two important timeout parameters relevant for sink connectors.

- Offset Flush Interval (offset.flush.interval.ms)

  This is the interval, in milliseconds, at which Kafka Connect attempts to commit task offsets.
- dataflow.timeout

  Specifies the maximum amount of time to wait for the dataflow to complete.

Important: The configuration scope of the two properties is different. Offset Flush Interval (offset.flush.interval.ms) is a Kafka Connect role property that you set in Cloudera Manager. The value set for this property is **global** and affects **all** connectors in your deployment. On the other hand, dataflow.timeout is a connector level property. You set it during connector deployment and is configured on a per-connector basis.

The execution of a `SinkTask` is the following:

1. The consumer consumes messages from the Kafka topics for the period of time determined by Offset Flush Interval (default is one minute), and puts the messages on the input of the `SinkTask`, that is, the input port of the connector's dataflow.
2. After Offset Flush Interval is reached, the consumer stops consuming messages and the flow is triggered. Once triggered, the flow processes all messages that accumulated on its input port. Processing time is not infinite. The flow will time out once the value set in dataflow.timeout is reached.
3. After flow is finished running, the process restarts with step 1.

Notice that the flow is not triggered for each Kafka message individually. Instead, it is triggered for a batch of messages that the consumer put on its input port in the time specified by Offset Flush Interval. One execution (trigger) of the flow is one transaction. This means that the flow either manages to process all messages within the batch successfully (delivers them to the target system and commits the transaction), or rolls back the session. If the session is rolled back, the consumer is going to put the same batch of messages on the input of the flow in the next execution of step 1. Therefore, in the next execution of step 2, the flow is going to try and process the same batch of messages.

Each Kafka message is delivered to the target system as a single flowfile. Depending on the destination, Cloudera recommends that you consider merging your Kafka records together into bigger chunks before delivering them. For example, assume you are delivering data to HDFS. Because HDFS is not built for accessing and handling many small files, it is not recommended to send each individual Kafka message to HDFS as a separate file.

Merging considerations for sink connectors

Learn about the limitations and considerations when using MergeContent and MergeRecord processors with Stateless NiFi.

Tip: Before you continue, ensure that you read Dataflow execution and scheduling first. You mus have an understanding of how dataflow execution works to understand the examples and settings discussed here.

NiFi supports many processors that can be used as sinks for Kafka data. If the destination service operates well with many small messages, then the data from Kafka can often be delivered directly to a sink processor. For example, a PublishJMS processor can easily handle many small messages. However, other services, like S3 or HDFS, perform much better if the data is first batched or merged together. For this reason, the MergeContent and MergeRecord processors are extremely popular in NiFi. They make it possible to merge many small flowfiles into one larger flowfile.

With MergeContent and MergeRecord in traditional NiFi, you can set a minimum and maximum size for the merged data along with a timeout. However, with Stateless NiFi and Kafka Connect, this might not work. This is because only a limited number of flowfiles are made available to the processor. You can still use these processors to merge the data together, but some limitations apply.

If MergeContent or MergeRecord is triggered but not enough flowfiles are present to create a batch, the processor does nothing. Therefore, Cloudera recommends that you set Minimum Number of Entries to 1 and Minimum Group Size to 0 B (these are default values). With these settings, the MergeContent and MergeRecord processors create a merged flowfile from the input flowfiles available in the given execution of the connector and do not wait for more data to arrive. Configuring Max Bin Age is not required in this case.

The Maximum Number of Entries and Maximum Group Size processor properties can also be used to limit the maximum size of the merged flowfile. For example, in a NiFi Stateless Sink connector the consumer running inside the connector puts a group of messages on the input flow. Afterward, the flow is executed. This execution of the flow can only work with the amount of messages that it has on its input when it gets triggered. If it had 115 messages on its input, and MergeContent processor's Maximum Number of Entries is 100, the flow is going to produce two outgoing messages, one with 100 entries and another one with 15.

To control how many messages are placed on the input of the flow for one execution, you can configure the Offset Flush Interval (offset.flush.interval.ms) Kafka Connect role property. The lower the interval, the less time the consumer has to put messages on the input of the flow.

NiFi Stateless Sink merging example

Let's look at a merging example with a NiFi Stateless Sink connector. Assume that the dataflow contains the following elements in the specified order:

```
input port -> MergeContent -> PutHDFS
```

Additionally, assume the following configurations:

- Offset Flush Interval is set to one minute (default). In this timeframe the internal consumer puts 1100 messages on the input port of the dataflow.
- dataflow.timeout is 90 seconds.
- The MergeContent processor has the following properties:
  - Maximum Number of Entries = 1
  - Minimum Group Size = 0B
  - Maximum Number of Entries = 100

Important: Offset Flush Interval is a Kafka Connect role property. It is configured in Cloudera Manager. dataflow.timeout is a connector property. It is configured during connector deployment, for example in Streams Messaging Manager. MergeContent's properties are processor properties configured in NiFi when you build the dataflow.

In this case, MergeContent creates 11 flowfiles, and PutHDFS has to upload these 11 files to HDFS within what is left of dataflow.timeout (90 seconds). Assume that PutHDFS manages to upload six files when dataflow.timeout is up. Since the dataflow did not finish execution within the dedicated timeframe, it gets interrupted, and the transaction is rolled back. The consumer starts putting messages on the input of the dataflow for a minute again.

However, since the previous transaction was rolled back, the messages that the dataflow tried to process were not committed. Therefore, the consumer puts the same 1100 messages on the input port of the dataflow. This behavior can result in an endless loop. Additionally, HDFS was not notified that the six files that were uploaded successfully should be removed. This results in these files being reuploaded in the next execution. You can resolve an issue like this by doing the following:

1. Decrease Offset Flush Interval.

   For example, decreasing this interval to 30 seconds (from one minute) halves the number of messages that are put on the input port of the dataflow. Fewer messages on the input port means that the dataflow needs to process fewer messages in one execution. 550 instead of 1100.
2. Set Maximum Number of Entries to a value that is larger than the number of messages the you expect the dataflow will receive.

   Increasing this value to, for example, 1,000,000,000 ensures that all messages are put in one HDFS file. This way all data is either successfully uploaded to HDFS, or the upload is interrupted and rolled back, but without leftover bits in the target system.

Considerations for listen-type source connectors

There are a number of limitations that apply to listen-type connectors that you should be aware of.

A listen-type source connector consists of a NiFi dataflow in which the first processor is a ListenX processor. These processors listen on specific interfaces and specific ports on the host where they are running. For example, a flow starting with a ListenHTTP processor listens on a specific port for HTTP requests made by external clients.

The Kafka Connect framework does not provide the concept of task affinity mapping, meaning that if a connector is deployed with multiple tasks, it is not possible to ensure that tasks are assigned to different Kafka Connect workers (Kafka Connect role instances). Two tasks started on the same worker in a listen-type connector results in a failure.

This happens because both connectors try to bind to the same port. For this reason, you must ensure that listen-type source connectors are deployed with a single task. That is, you must set tasks.max=1 when configuring the connector.

A common deployment model is to have a load balancer in front of the Kafka Connect workers and a mapping rule linking to the port on each Kafka Connect worker. This way, the load balancer FQDN can be provided to the client without assuming which worker the task is going to be deployed on.

If multiple tasks are required for performance, you can choose to have multiple standalone Kafka Connect workers and deploy the connector independently on each worker. However, in such a case, Cloudera recommends using Cloudera Flow Management and NiFi as they provide a significantly more powerful environment for this type of use cases.

Mapping of NiFi features

There are some features that exist in NiFi that translate seamlessly in the context of Kafka Connect. The following discusses some of these NiFi features and how they can be utilized in Kafka Connect.

State management

In NiFi, a processor is capable of storing state about the work that it has accomplished. This is particularly important for source components such as the ListS3 processor. This processor keeps state about the data that it has already seen so that it does not list the same files repeatedly. State in NiFi can be stored local to the node (local state), or across all nodes in a cluster (cluster-wide state).

When using the Stateless NiFi Source connector, the state that is stored by these processors is provided to Kafka Connect and is stored within Kafka itself as source offsets and source partition information. This allows a tasks to restart and resume where they left off. If a processor stores local state in NiFi, it will be stored in Kafka using a source partition that corresponds directly to that task. As a result, each task is analogous to a node in a NiFi cluster. If the processor stores cluster-wide state in NiFi, the state will be stored in Kafka using a source partition that corresponds to a cluster-wide state.

Note: Kafka Connect does not allow for state to be stored for a sink task.

Primary node

NiFi provides several processors that are expected to run only on a single node in the cluster. This type of execution is configured by setting Execution to Primary Node Only in the SCHEDULING tab on the NiFi UI when configuring a processor.

When using the Stateless NiFi Source connector, if any source processor in the dataflow is set to run on a primary node only, only a single task will ever run, even if the tasks.max connector property is set to a value higher than one. If you attempt to use multiple tasks for a dataflow that has Primary Node Only set, a warning is logged.

Because processors should only be scheduled with Primary Node Only if they are sources of data, this setting is ignored for all sink tasks and for any processor in a source task that has incoming connections.

Processor yielding

When a processor determines that it is not capable of performing any work (for example, because the system that the processor is pulling from has no more data available), it might choose to yield. This means that the processor stops running for some amount of time. You can configure how long the processors stops running in a dataflow by configuring Processor Configure SETTINGS Yield Duration in NiFi. When using a source connector, if a processor chooses to yield, the source connector pauses for the configured amount of time before triggering the dataflow to run again.

Kafka Connect worker assignment and Stateless NiFi

Learn how NiFi Stateless connectors and tasks get assigned to Connect workers, in what cases you need to be aware of worker assignments, and how you can look up worker assignments.

When running Kafka Connect in a cluster environment, each cluster node in the Kafka Connect cluster has a Kafka Connect worker running on it. In Cloudera, when the cluster is managed by Cloudera Manager, Kafka Connect workers are represented by Kafka Connect role instances, which are deployed under your Kafka service. This means that the number of workers available in the cluster is identical to the number of Kafka Connect role instances.

When you deploy a connector, the connector is assigned to a worker. Additionally the task or tasks of the connector are also assigned to workers. The workers that the connector and tasks are assigned to might be the same, but might also be different. If the connector has multiple tasks (tasks.max>1), each task might be assigned to the same worker, or they might be assigned to different workers. In addition, if a Kafka Connect worker is stopped, the connectors and tasks that were assigned to it are moved to different workers in the Connect cluster. In general, you have no agency over worker assignment, it is decided by the Connect Framework, and it is not configurable. Additionally, worker assignments might change dynamically, however, this is a normal and expected behavior of Kafka Connect.

Note: In a test environment you can force deploy connectors and tasks to a specific worker by stopping all other workers In the Connect cluster.

With Stateless NiFi connectors, knowing exact worker assignments is required when you are running a listen-type connector (for example, HTTP Source, Syslog TCP Source, or Syslog UDP Source). With these types of connectors you must ensure that you send messages to the host of the task's worker and not the connector's worker.

You can look up worker assignments in Streams Messaging Manager on the Connector Profile page of the connector. The worker assigned to the connector is displayed in the **Connector Profile** section under **ASSIGNED WORKER** (1). The worker assigned to the task is displayed in the **Tasks** section under **Worker ID** (2).

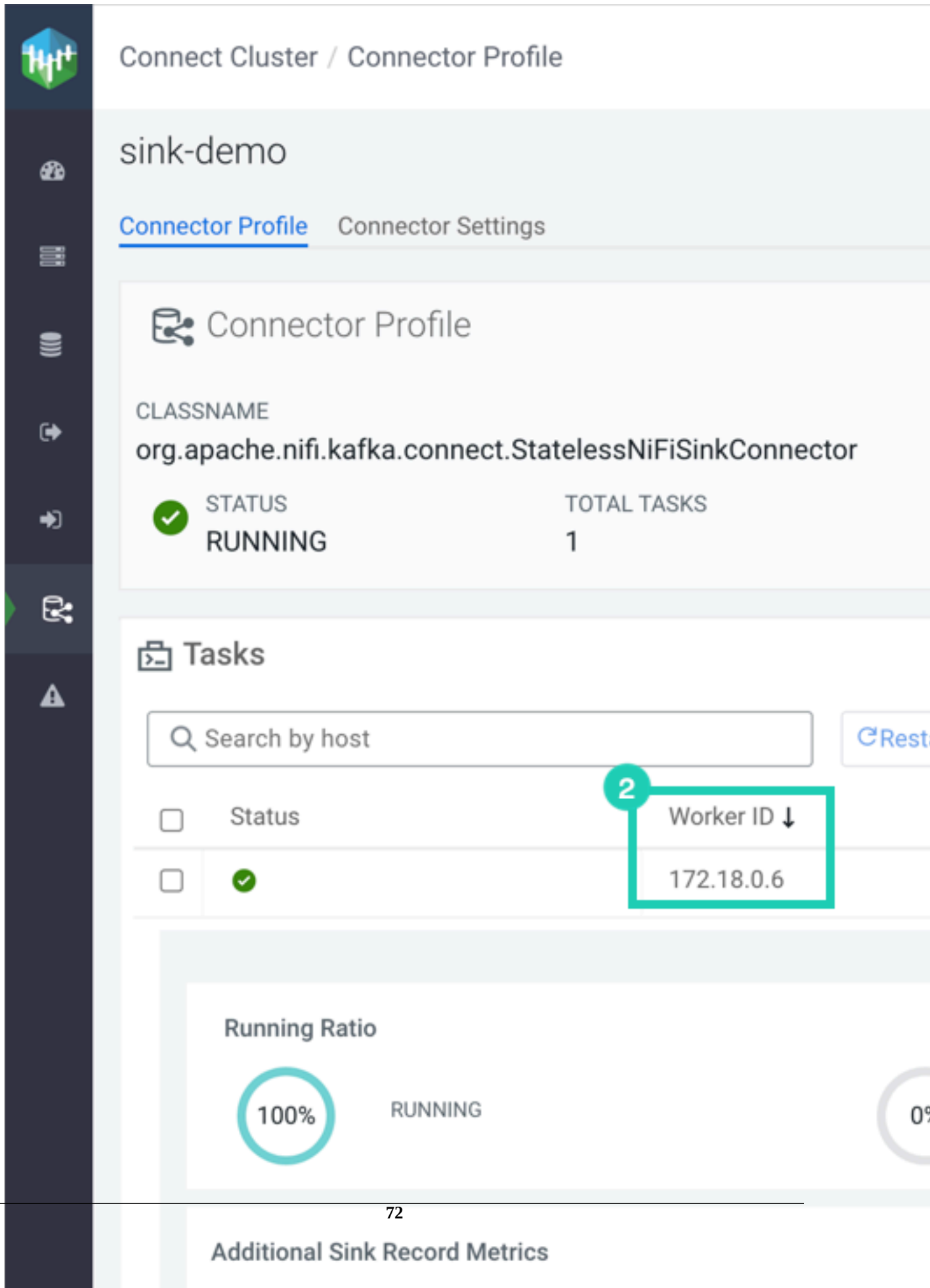
Connect Cluster / Connector Profile
sink-demo
Connector Profile
Connector Settings
Connector Profile
CLASSNAME
org.apache.nifi.kafka.connect.StatelessNiFiSinkConnector
STATUS
RUNNING
TOTAL TASKS
1
Tasks
Search by host
2
Status
Worker ID
172.18.0.6
Running Ratio
100%
RUNNING
Additional Sink Record Metrics

Related Information

Kafka Connect log files and Stateless NiFi

Kafka Connect log files and Stateless NiFi

Learn how and where Stateless NiFi connectors produce their logs.

Each cluster host that runs a Kafka Connect worker (Kafka Connect role) has a log file specific to Kafka Connect. The logs are stored by default in /var/log/kafka. The Connect related logs are named kafka-connect-*[***HOSTNAME***]*.log. The log directory path is specified by the Kafka Connect Log Directory Kafka Connect role property. This property is configured in Cloudera Manager.

Every log entry produced by a NiFi dataflow that is running in a Kafka Connect connector is directed to the Kafka Connect log file. Specifically, the dataflow log entries are written to the log file located on the cluster host that is running the Kafka Connect worker that is executing the connector's task.

For example, if your connector has a single task, which is executed by worker-1 running on host-1, then the log for that connector will be located on host-1. If a connector has multiple tasks, each running on a different worker, then each task's dataflow writes logs to the respective worker's log file.

If a Kafka Connect worker is stopped, its tasks are moved to other workers. As a result, it is possible that a specific connector's log entries first appear in the log file located on one host, but later on appear in another log file located on a different host.

Related Information

Kafka connect worker assignment and Stateless NiFi

Kafka Connect tasks and Stateless NiFi

Learn which Stateless NiFi connectors you can or cannot run with multiple Kafka Connect tasks (task.max>=2).

By default all connectors that you deploy start with a single Kafka Connect task. The number of tasks deployed for a connector can be configured using the tasks.max connector property, but not all types of connectors are meant to run using more than a single task.

Specifically with Stateless NiFi, there are a handful of source connectors that either cannot run with multiple tasks or function incorrectly when multiple tasks are configured. The following provides information on which connectors can or cannot be run with multiple tasks and what effects configuring multiple tasks has on both sink and source connectors.

Source connectors

Notice that the connector's task is made up of multiple components, these are as follows:

- `SourceTask`

  In Stateless NiFi connectors, the `SourceTask` is responsible for executing one instance of the connector's dataflow.
- SMT Chain

  The Single Message Transforms (SMT) chain can modify and filter the data on a single message basis.
- Converter

  The converter is responsible for converting Kafka message keys and values with a configured conversion mechanism. However, Stateless NiFi connectors currently only use keys of string type and values of byte array type. As a result, the converter does not perform any meaningful conversion.
- Kafka Producer

  The Kafka producer is responsible for delivering messages to the configured Kafka topics.

If you deploy a source connector with more than one task, then each task is going to have its own instance of the task components. Because the dataflow in the `SourceTask` is responsible for fetching the data, rather than the source data store providing the data to the `SourceTask`, it is not always possible to start a connector with more than one task.

For example, starting a JDBC Source connector, which is a Stateless NiFi-based connector, with two tasks would mean that two instances of the dataflow are running in parallel. Both are fetching data from the same table of the same database. These two tasks would be delivering the messages to the same Kafka topic. However, the two dataflow instances running in parallel would have no way of knowing whether a certain record was already been fetched by the other instance. This results in the same piece of data appearing in multiple Kafka messages.

Because of this, there are a number of limitations that you must keep in mind when using the Stateless NiFi Source connectors. These limitations apply when you are running a custom dataflow with a Stateless NiFi Source connector, as well when running any of the Stateless NiFi-based source connectors that Cloudera develops.

Listen-type connectors

A listen-type connector consists of a NiFi dataflow in which the first processor is a ListenX processor. Listen-type processors start a server with one or more background threads, listening on a defined network port. Stateless NiFi-based connectors of this category are:

- HTTP Source
- Syslog TCP Source
- Syslog UDP Source

For example, the HTTP Source connector's flow uses a ListenHTTP NiFi processor to start an HTTP server listening on a configured port. Deploying this connector with two tasks would mean that two ListenHTTP processors are running simultaneously, both listening on the same network port. Since there is no user-defined mapping between tasks and Kafka Connect workers, it is possible that these two instances get deployed to the same worker. This means that two HTTP servers would be listening on the same port on the same cluster node. This results in port collision. To avoid issues like this, you must ensure that listen-type connectors are always deployed with exactly one task.

Primary node only connectors

Connectors in this category are connectors that use a dataflow with a NiFi processor that is configured to run on the primary node only if the flow was running on a regular NiFi cluster. For example, the QueryDatabaseTableRecord processor is a processor that is intended to run on primary nodes only. This processor is used in the JDBC Source connector. Stateless NiFi-based connectors of this category are:

- JDBC Source
- SFTP Source

For these connectors, the configured value of the tasks.max property is ignored. The connector is always started with exactly one task.

All other connectors

Connectors in this category are connectors that are neither listen-type or primary node only. Stateless NiFi-based connectors of this category are:

- JMS Source
- MQTT Source

There is no limitation to the number of tasks for these connectors. In the case of these connectors, it is the responsibility of the source data store system to make sure that one piece of data is only given to one of the connector's tasks

Sink Connectors

Figure 5: Sink Connector Architecture

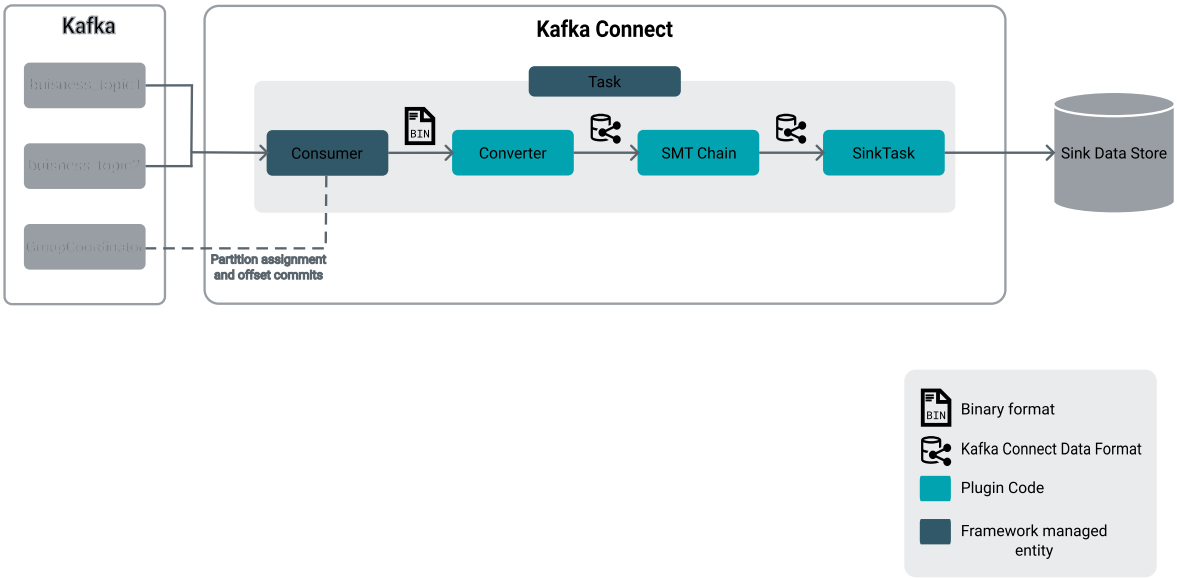

Notice that the connector's task is made up of multiple components, these are as follows:

- Kafka Consumer

  The consumer is responsible for fetching the messages from the Kafka topics and delivering them to the converter.
- SMT Chain

  The Single Message Transforms (SMT) chain can modify and filter the data on a single message basis.
- Converter

  The converter is responsible for converting Kafka message keys and values with a configured conversion mechanism. However, Stateless NiFi connectors currently only use keys of string type and values of byte array type. As a result, the converter does not perform any meaningful conversion.
- `SinkTask`

  In Stateless NiFi connectors, the `SinkTask` is responsible for executing one instance of the connector's dataflow.

If you deploy a sink connector with more than one task, each task is going to have its own instance of the four task components.

With sink connectors, there is no limitation to using more than one task for a connector. When a connector is started with multiple tasks, each task has its own Kafka consumer, but these consumers belong to the same Kafka consumer group. Kafka makes sure that a message fetched by one consumer is not given to another consumer within the same consumer group.

There is no user-defined mapping between tasks and Kafka Connect workers, meaning that once deployed, the tasks of a connector may or may not be executed on the same worker.

Developing a dataflow for Stateless NiFi

Learn about the recommended process of building a dataflow that you can deploy with the Stateless NiFi Sink or Source connectors. This process involves building and designing a parameterized dataflow within a process group and then downloading the dataflow as a flow definition.

About this task

The general steps for building a dataflow are identical for both source and sink connector flows. For ease of understanding a dataflow example for a simple MQTT Source and MQTT Sink connector is provided without going into details (what processors to use, what parameters and properties to set, what relationships to define and so on).

Before you begin

- Ensure that you reviewed Dataflow development best practices for Stateless NiFi.
- You have access to a running instance of NiFi. The NiFi instance does not need to run on a cluster. A standalone instance running on **any machine**, like your own computer, is sufficient.
- Ensure that the version of your NiFi instance matches the versions of the Stateless NiFi plugin used by Kafka Connect on your cluster.

  You can look up the Stateless NiFi plugin version either by using the Streams Messaging Manager UI, or by logging into a Kafka Connect host and checking the Connect plugin directory. In addition, ensure that you note down the plugin version. You will need to manually edit the flow definition JSON and replace the NiFi version with the plugin version.

For Use Streams Messaging Manager UI

1. Access the Streams Messaging Manager UI, and click **Connect** in the navigation sidebar.
2. Click the New Connector option.
3. Locate the StatelessNiFiSourceConnector or StatelessNiFiSinkConnector cards. The version is located on the card.

   The version is made up of multiple digits. The first three represent the NiFi version. For example, if the version on the card is 1.18.0.2.4.3.0-63, then you should use NiFi 1.18.0 to build your flow.

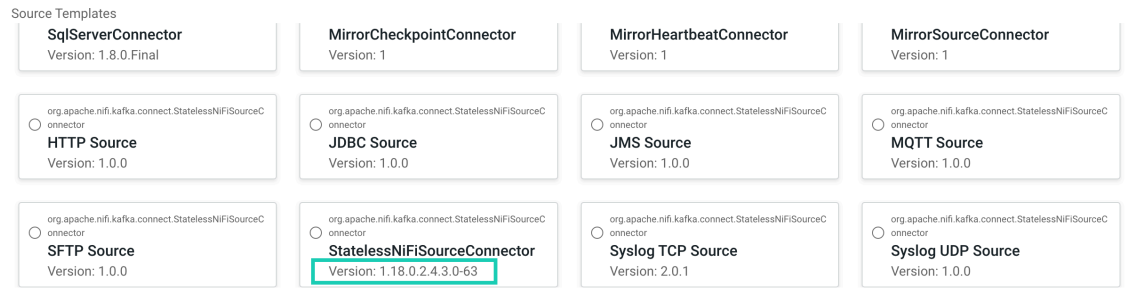

For Check Connect plugin directory

1. Using SSH, log in to one of your Kafka Connect hosts.
2. Navigate to the directory where your Kafka Connect plugins are located.

   The default directory is /var/lib/kafka. If you are using a custom plugin directory, go to Cloudera Manager Kafka service Configuration and search for the Plugin Path property. This property specifies the directory where Kafka Connect plugins are stored.
3. List the contents of the directory and look for a nifi-kafka-connector-*[***VERSION***]* entry.

   The version is made up of multiple digits. The first three represent the NiFi version. For example, if the version in the name of the entry is 1.18.0.2.4.3.0-63, then you should use NiFi 1.18.0 to build your flow.

Procedure

1. Access the NiFi UI.
2. Create a new process group.
3. Right-click the process group and select Configure.
4. Create a new parameter context and assign it to the process group:
   a) Select Process Group Parameter Context Create new parameter context...
   b) Enter a name for the parameter context.
   c) Click Apply.

5. Click Apply and then click OK.
6. Close the configuration dialog.
7. Double-click the process group you created.

8. Design and parameterize your dataflow.

   For example, you can build a dataflow that you deploy as a source connector. The following is an example dataflow for a simple MQTT source connector. It consists of a ConsumeMQTT processor and an output port (representing a Kafka topic).

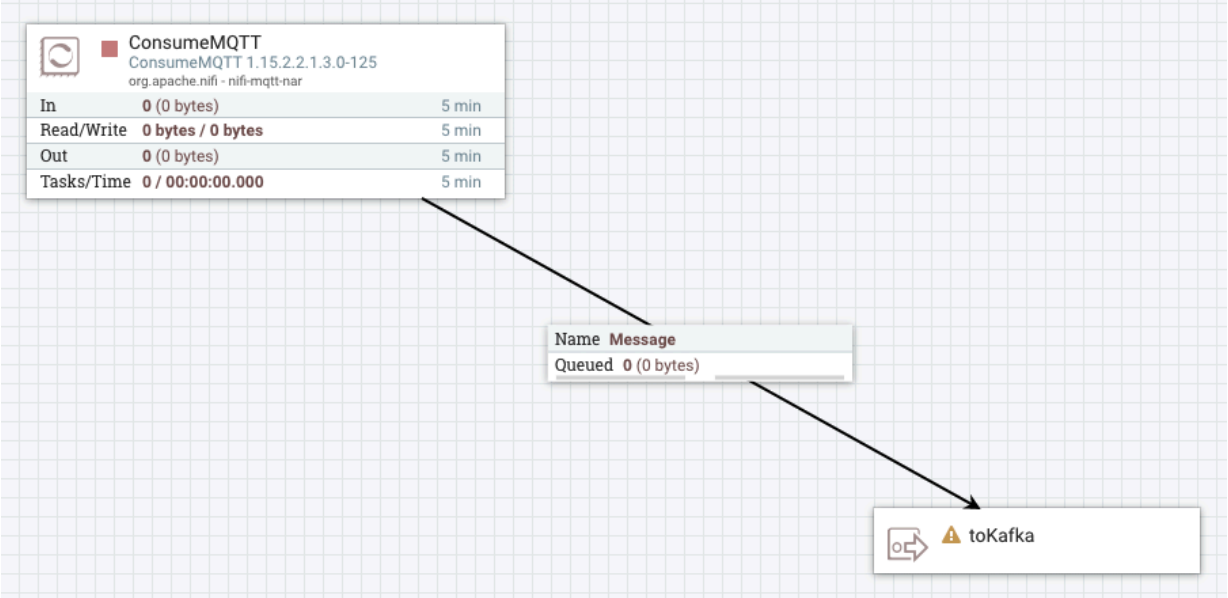

   Alternatively, you can also build a dataflow to deploy as a sink connector. The following is an example dataflow for a simple MQTT sink connector. It consists of an input port (representing a Kafka topic), a PublishMQTT processor, and two output ports.

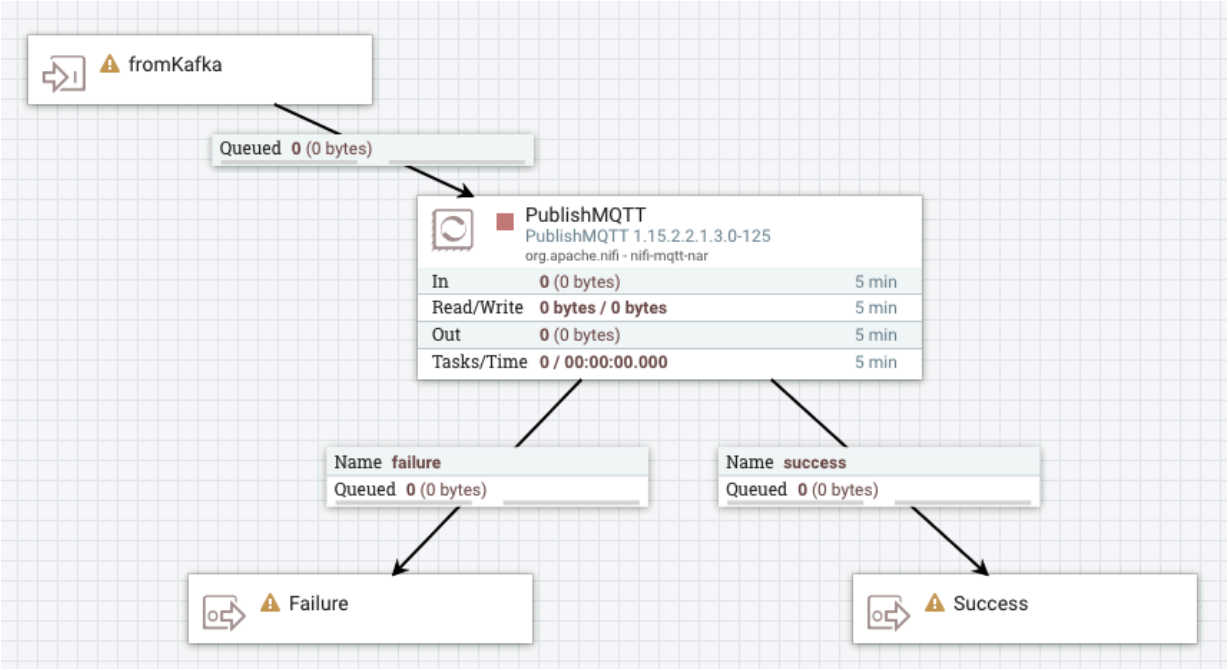

   In the case of this example, a failure output port is required. This is done so that in case the destination is not available, the session can be rolled back and the Kafka message can be declined. The failure port is specified in the configuration of the connector when deploying it through Streams Messaging Manager.

   When designing and building your dataflow, ensure that you update your parameter context and reference the parameters in your components. For example:

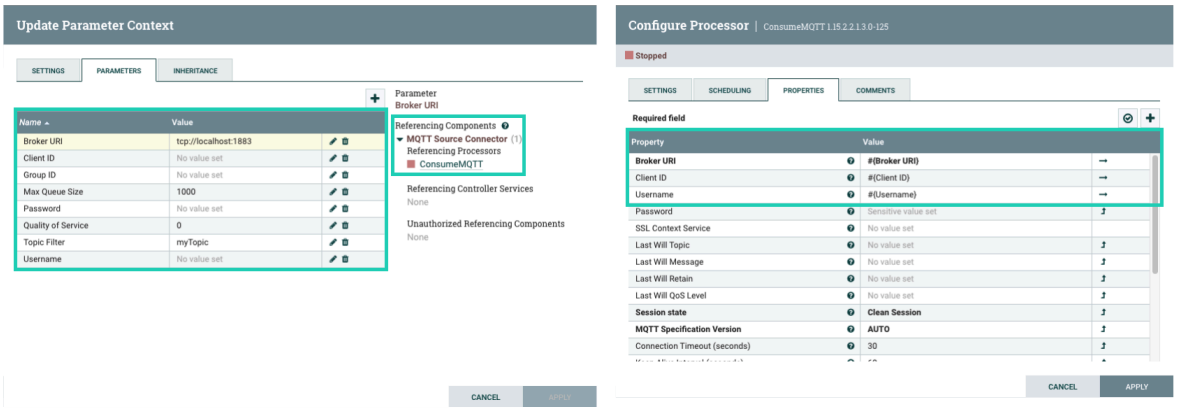

Keep in mind that sensitive properties in processors or controller services must always be parametrized. That is, they must get their values from a parameter in the process group's parameter context. This is because the values of sensitive properties are not exported with the process group, therefore, they must be configurable when deploying the connector.

Additionally, sensitive processor or controller service properties can only reference sensitive parameter context parameters. That is, the parameter must be marked as sensitive during creation.

For more information on building dataflows, parameters, and referencing parameters, see the *Flow Management* library.

9. Exit the process group.
10. Right-click the process group and select Download flow definition Without external service .

    If you followed these steps, your process group is not referencing controller services that are outside of the process group. As a result, you can choose to download the flow without external services included.
11. Open the JSON file you downloaded in a text editor, and replace all occurrences of the NiFi version with the Stateless NiFi plugin version used in your cluster.

    For example, if the NiFi version is 1.18.0 and your plugin version is 1.18.0.2.4.3.0-63, then every occurrence of 1.18.0 in the JSON must be replaced with 1.18.0.2.4.3.0-63.

Results

The dataflow is downloaded as a flow definition (JSON file).

What to do next

Deploy the dataflow as a Kafka Connect connector using the Stateless NiFi Source or Sink connectors. Continue with *Deploying a dataflow using stateless NiFi*.

Related Information

Deploying a dataflow using Stateless NiFi

Flow Management

Deploying a dataflow using Stateless NiFi

A custom developed NiFi dataflow can be deployed with the Stateless NiFi Source or Sink connectors using Streams Messaging Manager.

About this task

After building and downloading your dataflow, you can deploy it in Kafka Connect as a source or sink connector. This can be done using Streams Messaging Manager. The following list of steps walk you through the process of deploying a Stateless NiFi Sink or Source connector and provide examples on how dataflows are configured using Streams Messaging Manager.

These steps are for the Streams Messaging Manager UI, however, the actions described here can also be completed using the Streams Messaging Manager REST API. For more information, see *Streams Messaging Manager REST API Reference*.

Before you begin

- A Cloudera cluster containing Streams Messaging Manager and Kafka is available.
- You have access to the Streams Messaging Manager UI.
- The flow definition of the dataflow that you want to deploy is available to you. For more information on how to design, build, and export a dataflow using the NiFi UI, see Developing a dataflow for Stateless NiFi .
- The properties of Kafka consumers and producers used internally by the connector (Connect-managed clients) can be configured on a per-connector basis. The following steps give an example of how this can be done. However, in order for consumer and producer property overrides to take effect, you must ensure that the Connector Kafka Client Configuration Override Policy Kafka service property is set to All in Cloudera Manager. By default this property is set to None, which means that no overrides are allowed.

Procedure

1. Access the Streams Messaging Manager UI.
2. Click (Connect) on the navigation sidebar.
3. Click + New Connector to add a new connector.
4. Select a connector.

   Which connector you select depends on the type of dataflow you want to run. Select StatelessNiFiSourceConnector found on the **Source Templates** tab if your dataflow collects (sources) data from systems and publishes it to Kafka. Select the StatelessNiFiSinkConnector from the Sink **Templates** tab if your dataflow delivers (sinks) Kafka data into other systems.
5. Enter a name for the connector in the Enter Name field.

   Ensure that you add a unique and easily identifiable name. The name of the connector cannot be changed once the connector is deployed.
6. Configure the connector

   Connector configuration can be broken into three steps. First, you configure the properties available in the default configuration template. Second, you configure other connector properties. Finally, you specify the flow parameters specific to your dataflow. These are the parameters that you set up in the parameter context when you built the flow in NiFi. Even though the following breaks the process up into multiple substeps, all properties are configured on the same page in the Streams Messaging Manager UI.

   a) Configure the properties available in the default template.

      The following properties from the default template are the ones that you should configure. Other properties in the template that are not highlighted here have working default values configured. Configuring the properties that are present in the template but not highlighted in the following list is not recommended by Cloudera.

      flow.snapshot

      Specifies the dataflow to run. You have multiple options on how to specify the value of this property. You can copy and paste the contents of the flow definition, upload the flow definition, or reference a file path. For more information on flow.snapshot configuration, see *Configuring flow.snapshot for Stateless NiFi connectors*.

      input.port

      The name of the input port in the NiFi dataflow that Kafka records are sent to by the connector's Kafka consumer. This property is specific to the Stateless NiFi Sink connector and is a port in NiFi terminology, not a network port. If the dataflow contains exactly one input port, this property is

optional and can be omitted. However, if the dataflow contains multiple Input Ports, this property must be specified.

output.port

The name of the output port in the NiFi dataflow that is used to deliver the messages from the dataflow to the connector's Kafka producer. This property is specific to the Stateless NiFi Source connector and is a port in NiFi terminology, and not a network port. If the dataflow contains exactly one port, this can be omitted. However, if the dataflow contains multiple ports (for example, a success and a failure port), this property must be specified. If any flowfile is sent to any port other than the specified port, it is considered as a failure. The session is rolled back and no data is collected.

topics

For source connectors, this is the name of the topic to deliver data to. For sink connectors, this can be a comma-separated list of topics to fetch data from.

krb5.file

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. This property is optional and is only required if a connection is established with a Kerberized service. Use the default value if an appropriate krb5.conf is located in /etc.

b) Add other connector properties.

There are many other properties that you can set. These include the properties specific to the Stateless NiFi connectors as well as the Kafka Connect properties that come from the base framework. Which of these you require depends on your dataflow and use case.

In Stateless NiFi connectors, the Kafka message's key is always a string, its value is always a byte array. As a result, the key and value converters of the connector should not be modified. Use the default values, which are as follows:

```
"key.converter": "org.apache.kafka.connect.storage.StringConverter"
"value.converter": "org.apache.kafka.connect.converters.ByteArrayConvert
er"
```

A comprehensive list of all other properties that the connectors accept can be found in *Stateless NiFi Source connector reference*, *Stateless NiFi Sink Connector reference*, and the *Apache Kafka documentation*.

c) Configure dataflow-specific parameters.

If you followed the recommendations provided in *Dataflow development best practices for Stateless NiFi* and *Developing a dataflow for Stateless NiFi*, your dataflow is parameterized. The parameters that you defined within the dataflow can be assigned values using the configuration pane in Streams Messaging Manager.

This is done by adding a configuration entry for each of the dataflow parameters that you want to configure. Dataflow parameters can be added to the configuration as follows:

Tip: Dataflow-specific parameters can also be automatically imported by clicking the Save and Enhance option when specifying the flow.snapshot property. For more information, see *Configuring flow.snapshot for Stateless NiFi connectors*.

```
"parameter.[***PARAMETER NAME***]": "[***VALUE***]"
```

For example, assume that you have a the following configuration entry:

```
"parameter.Directory": "/mydir"
```

In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory.

You can also apply a parameter to a specific parameter context. This is done by prefixing the parameter name with the name of the parameter context followed by a colon.

```
"parameter.[***PARAMETER CONTEXT NAME***]:[***PARAMETER NAME***]":
 "[***VALUE***]"
```

For example, assume you have the following configuration entry:

```
"parameter.My Context:Directory": "/mydir"
```

In a case like this, only the parameter context called My Context gets the specified value for the Directory parameter.

Once all properties are configured, your configuration should look similar to the following example:

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSource
Connector",
 "flow.snapshot": "[***FLOW DEFINITION JSON***]",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConve
rter",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.MQTT Source Parameters:MQTT Broker URI": "tcp
://[***HOST***]:[***PORT***]",
 "parameter.MQTT Source Parameters:MQTT Quality of Service": "0",
 "parameter.MQTT Source Parameters:MQTT Topics": "[***MQTT TOPIC
 NAME***]"
}
```

This example is for a custom built MQTT source connector. The example uses additional Connect properties such as key.converter and value.converter. The example also specifies a number of flow parameters such as MQTT Broker URI and MQTT Topics.

d) Configure Kafka producer and consumer properties.

Source connectors use internal Kafka producers to write message to topic, sink connectors use internal consumers to fetch messages from topics. The properties of these producers and consumers can be modified on

a per-connector basis. This is done by adding the required properties with the producer.override and consumer .override prefixes to the connector configuration.

For example, if you want to set the consumer of your sink connector to use auto.offset.reset=earliert, you need to add the following configuration entry to the configuration of the connector.

```
consumer.override.auto.offset.reset=earliest
```

7. Click Validate.

   The validator displays any JSON errors in your configuration. Fix any errors that are displayed. If your JSON is valid, the **JSON is valid** message is displayed in the validator.
8. Click Next.
9. Review your connector configuration.
10. Click Deploy.

Related Information

Streams Messaging Manager REST API Reference

Stateless NiFi Source properties reference

Stateless NiFi Sink properties reference

Apache Kafka documentation

Dataflow development best practices for Stateless NiFi

Developing a dataflow for Stateless NiFi

Configuring flow.snapshot for Stateless NiFi connectors

Downloading and viewing the predefined Stateless NiFi dataflows shipped in Cloudera

Learn how to download and view the predefined dataflows used by the Stateless NiFi-based connectors shipped in Cloudera. Predefined dataflows are downloaded from Cloudera cluster hosts. The downloaded files can be uploaded to NiFi to examine the dataflow's structure.

About this task

A number connectors shipped and developed by Cloudera are based on Stateless NiFi and run predefined dataflows. For example, the JDBC Source connector is a Stateless NiFi-based connector that runs a predefined dataflow.

To better understand how Stateless NiFi-based connectors like JDBC Source work, and how they move and manipulate Kafka data, it can be useful to view the dataflow that the connectors use.

These dataflows are installed on every cluster host that is running a Kafka Connect service role. You can fetch the dataflows from your cluster hosts, load them into NiFi, and look at their flow structure.

The flow definition JSON files of these dataflows are present on the cluster hosts at /etc/kafka_connect_ext/flow-definitions. However, you cannot simply load these flow definition files as is into NiFi. They must be edited first.

Before you begin

- You have access to a running Cloudera cluster that has a Kafka service and Kafka Connect service roles deployed.
- You have access to a running instance of NiFi. The NiFi instance does not need to run on a cluster. A standalone instance running on **any machine**, like your own computer, is sufficient.

Procedure

1. Fetch the flow definition of the chosen connector from the cluster.

   The flow definition files are located on the cluster hosts at /etc/kafka_connect_ext/flow-definitions. How you complete this step depends on your cluster environment and what utilities are available to you. For example, if using scp, you can run the following command.

   ```
   scp [***USER***]@[***CLUSTER HOST***]:/etc/kafka_connect_ext/flow-definit
   ions/[***CONNECTOR NAME***]-[***VERSION***].json .
   ```

2. Open .json file in a text editor.
3. Copy contents of the "flow" JSON element to a separate .json file.
4. In the .json you created, replace the value of each "version" element with the version of NiFi you will use to view the dataflow.
5. In the NiFi UI, drag a new process group to the canvas.
6. In the Add Process Group modal, click and browse for your .json file.
7. Click Add.

Results

The dataflow is imported to the process group with all of its parameters.

What to do next

After the dataflow is imported, you can view the structure of the flow to gain a better understanding of how the dataflow operates. If you want to, you can make modifications and redeploy the modified dataflow as a custom connector.

Related Information

Developing a dataflow for Stateless NiFi

Deploying a dataflow for Stateless NiFi

Configuring flow.snapshot for Stateless NiFi connectors

Learn about the options you have in Streams Messaging Manager when configuring the flow.snapshot Stateless NiFi connector property as well as the Save and Enhance feature that can be used to automatically import NiFi parameters into the connector configuration.

Important: Configuring the flow.snapshot property is only required when you are deploying your custom developed dataflows with the Stateless NiFi Source or Sink connectors. Cloudera developed connectors powered by Stateless NiFi do not require flow.snapshot to be set. If the connector you are deploying does not by default have a flow.snapshot entry in its configuration template in Streams Messaging Manager, you do not need to configure it.

NiFi dataflows within Kafka Connect are deployed using the Stateless NiFi Source and Sink connectors. When you deploy your custom developed dataflows with the Stateless NiFi connectors, you must configure the flow.snapshot property. This property specifies the dataflow that is run within the connector.

Because the value of this property is the dataflow itself, the value you add to the property is not a simple string or integer. Instead, the value is the full contents of a flow definition JSON file, which depending on the dataflow, can be considerable in size.

As a result of this, the property is handled in a unique way by Streams Messaging Manager. You can use multiple configuration methods to specify value. Additionally, you can choose to automatically enhance (amend) the configuration with the NiFi parameters that are available within the flow definition.

Configuration methods

You can use the following configuration methods when specifying a value for flow.snapshot. The following instructions assume that a flow definition JSON file is available to you and that you are logged in to the **Connect** tab of the Streams Messaging Manager UI.

Note:

If the flow definition is larger than 64 KB, the following message appears:

```
Too large config, to view flow.snapshot click the edit button...
```

In a case like this, the dataflow is added, but you are not able to edit it directly in the **Properties** section.

Instead, you must click and edit or review the flow definition in the modal window that appears.

Copy and paste the contents of the flow definition

This method is similar to configuring any other connector property. The only difference is that the value you specify is considerably longer than other property value.

1. Open the flow definition in an editor.
2. Copy the full contents of the file.
3. Paste the contents of the file in the value field of the flow.snapshot property.

Upload the flow definition

This method involves uploading the flow definition from your own machine. This method is convenient if the flow definition JSON is available on the machine you are using to access the Streams Messaging Manager UI.

1. Click next to flow.snapshot.
2. Click Browse... in the modal window that appears.
3. Select the flow definition file that you want to use.
4. Click Save or Save and Enhance.

Reference a file path

The flow.snapshot property accepts file paths. This means that instead of copy-pasting or uploading the full contents of the flow definition JSON, you can also specify a location on disk where the file is available. Cloudera recommends that you use this method if the flow definition is larger than 1 MB.

1. Deploy the flow definition file on all Kafka Connect hosts.

   Important: Ensure that the file is deployed to all hosts at the same location and that the user that Kafka Connect is running as, which defaults to kafka, has read access to the file.
2. Set the file path as the value of the flow.snapshot property.

Using Save and Enhance

NiFi dataflows that you run within Kafka Connect are in the majority of cases parameterized. You can assign values to the parameters available within the dataflow on the Streams Messaging Manager UI during connector configuration. This is done by adding each parameter, with specific prefixes, in the **Properties** section. However, this can take a considerable amount of time if done manually. This is especially true if the dataflow has many parameters and multiple parameter contexts.

In cases like this, you can choose to enhance the configuration by clicking the Save and Enhance option after a value for the flow.snapshot property is added. If this option is used, Streams Messaging Manager parses the dataflow specified in flow.snapshot, extracts all parameters that are available in the dataflow, and adds them to properties of the connector. The Save and Enhance option is available on the modal window that appears after you click next to the property.

Tutorial: developing and deploying a JDBC Source dataflow in Kafka Connect using Stateless NiFi

A step-by step tutorial that walks you through how you can create a JDBC Source dataflow and how to deploy the dataflow as a Kafka Connect connector using the Stateless NiFi Source connector. The connector/dataflow presented in this tutorial reads records from an Oracle database table and forwards them to Kafka in JSON format.

Before you begin

- Look up the Stateless NiFi plugin version either by using the Streams Messaging Manager UI, or by logging into a Kafka Connect host and checking the Connect plugin directory.

For Use Streams Messaging Manager UI

1. Access the Streams Messaging Manager UI, and click **Connect** in the navigation sidebar.
2. Click the New Connector option.
3. Locate the StatelessNiFiSourceConnector or StatelessNiFiSinkConnector cards. The version is located on the card.

   The version is made up of multiple digits. The first three represent the NiFi version. For example, if the version on the card is 1.18.0.2.4.3.0-63, then you should use NiFi 1.18.0 to build your flow.

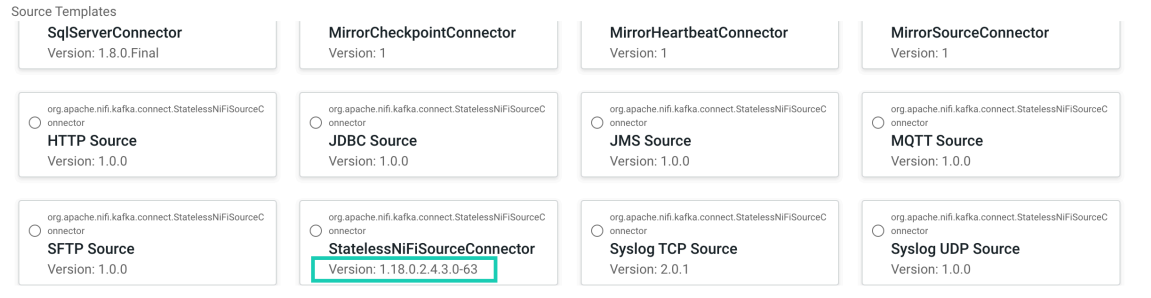

For Check Connect plugin directory

1. Using SSH, log in to one of your Kafka Connect hosts.
2. Navigate to the directory where your Kafka Connect plugins are located.

   The default directory is /var/lib/kafka. If you are using a custom plugin directory, go to Cloudera Manager Kafka service Configuration and search for the Plugin Path property. This property specifies the directory where Kafka Connect plugins are stored.
3. List the contents of the directory and look for a nifi-kafka-connector-*[***VERSION***]* entry.

   The version is made up of multiple digits. The first three represent the NiFi version. For example, if the version in the name of the entry is 1.18.0.2.4.3.0-63, then you should use NiFi 1.18.0 to build your flow.

- Download and start NiFi. You can download NiFi from https://archive.apache.org/dist/nifi/. This example uses NiFi 1.18.0 (nifi-1.18.0-bin.zip).
- The connector/dataflow developed in this tutorial requires the Oracle JDBC driver to function. Ensure that the driver JAR is deployed and available on every Kafka Connect host under the same path with correct file

permissions. Note down the location where you deploy the driver, you will need to set the location as a property value during connector deployment. For example:

```
cp ./ojdbc8-[***VERSION***].jar /var/lib/kafka_connect_jdbc/oracle-
connector-java.jar
```

```
chmod 644 /var/lib/kafka_connect_jdbc/oracle-connector-java.jar
```

Procedure

1. In NiFi, create a process group and give it a name.

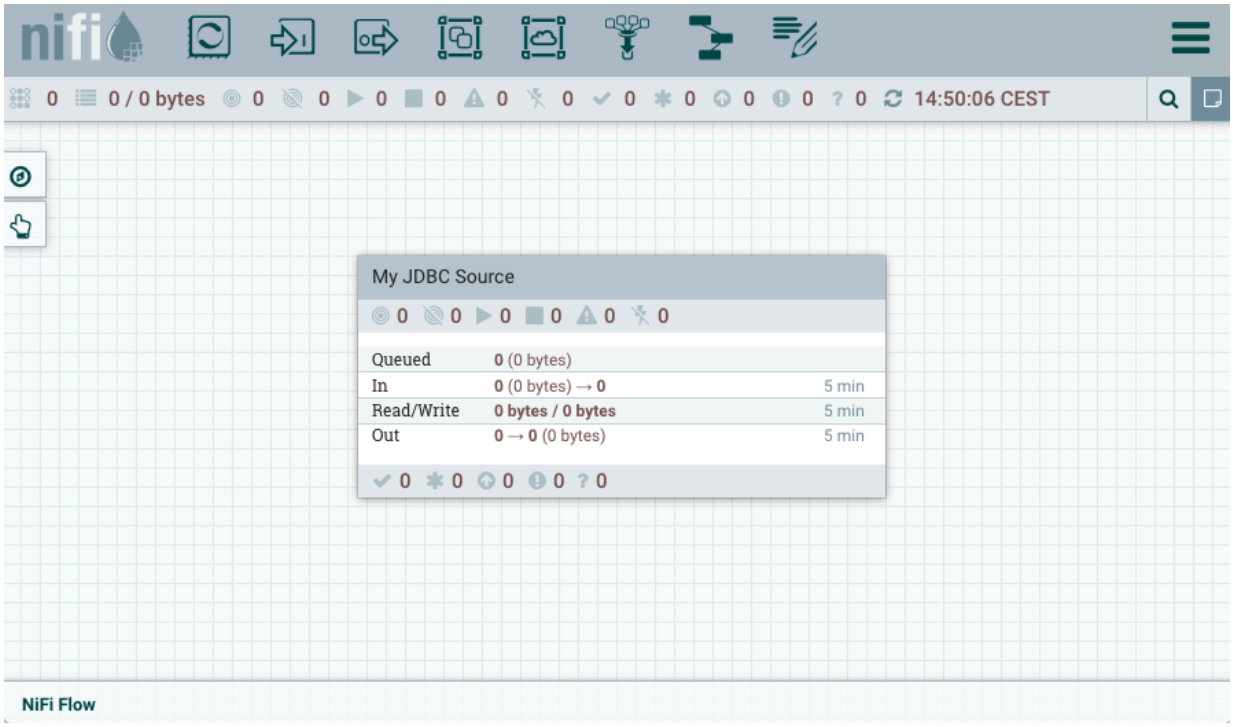

2. Add a QueryDatabaseTableRecord processor and an output port to the process group.

   Note down the name of the output port, you will need to set the name as the value of a connector property during connector deployment.

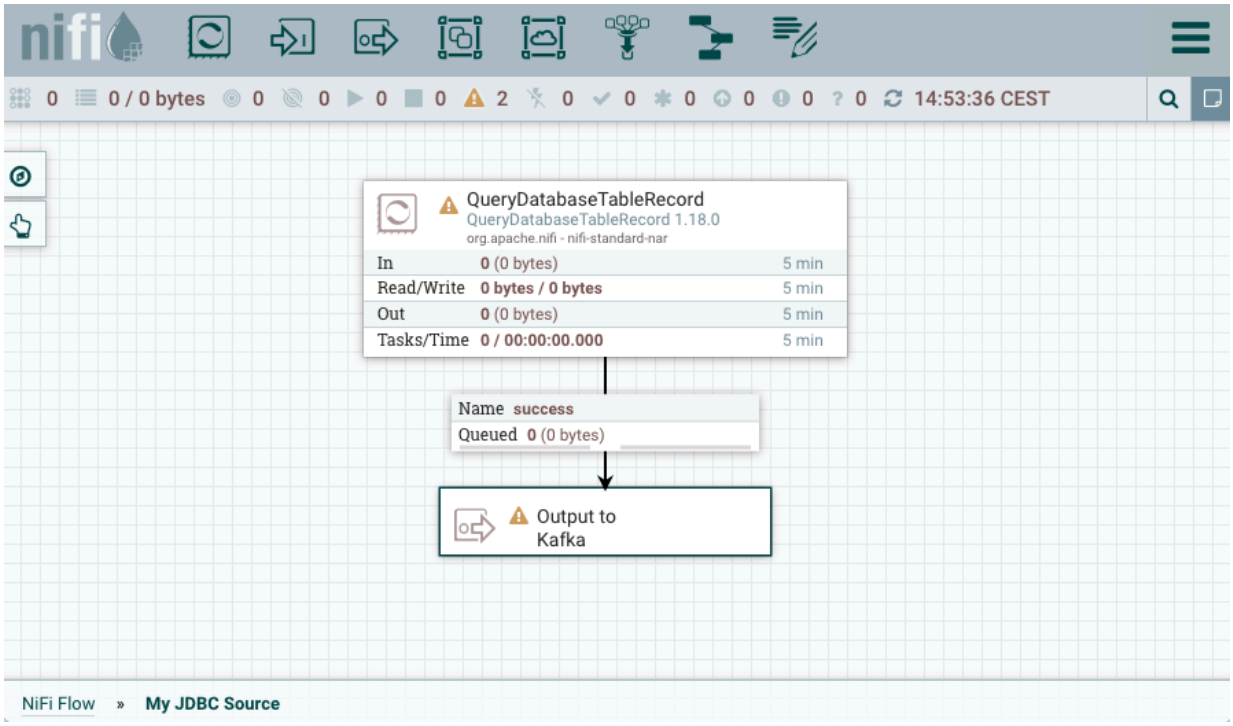

3. Add a DBCPConnectionPool and a JsonRecordSetWriter controller service to the process group.

   The controller services are required for the QueryDatabaseTableRecord processor to access the database and to convert records.

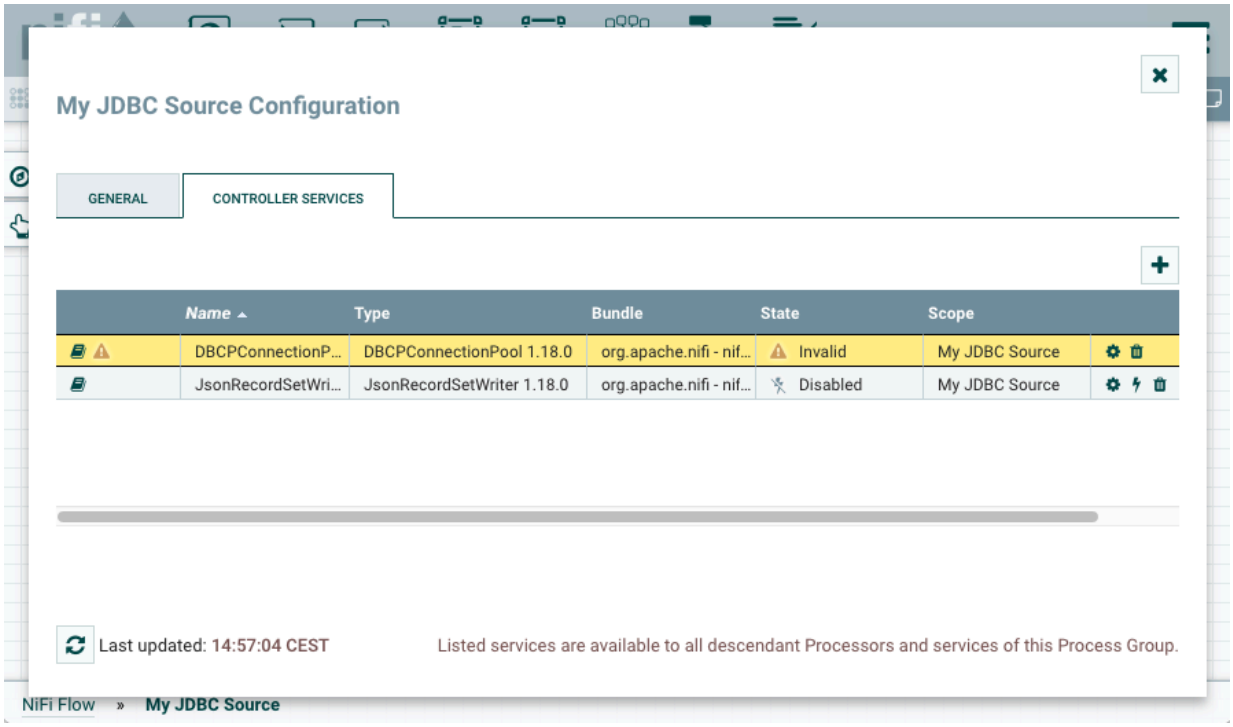

4. Examine the properties of the QueryDatabaseTable processor and the two controller services. Collect the properties that you want to parameterize.

   Parameterized properties can be set when deploying the connector. In this example, the following properties are parametrized:

   - Database Type
   - Table Name
   - Maximum-value Columns
   - Initial Load Strategy
   - Database Connection URL
   - Database Driver Class Name
   - Database Driver Location
   - Database User
   - Database Password
5. Create a parameter context, give it a name, and add the above mentioned parameters to the parameter context.

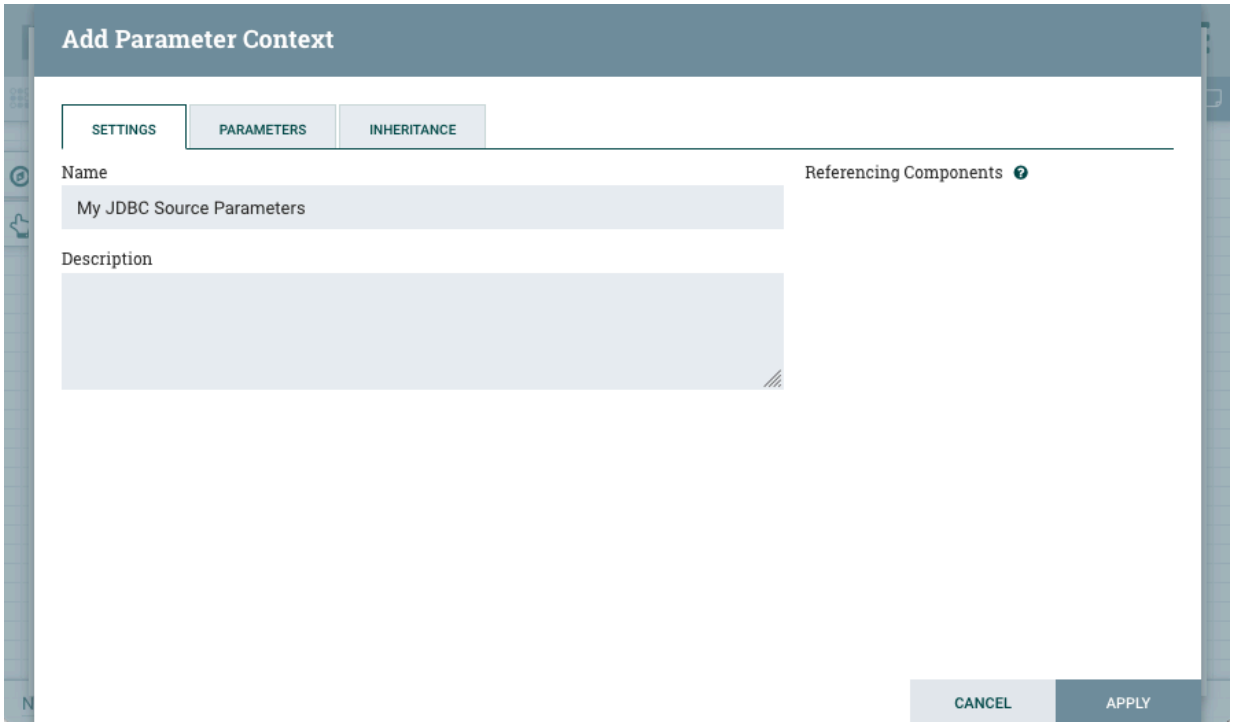

   You can assign values to the parameters in this step. The values you assign here serve as default values for the parameters when the flow is deployed as a connector. Assigning values is not mandatory as all parameters will

be configurable during connector deployment. However, assigning sensible default values is a good practice and makes the dataflow easier to deploy in Kafka Connect.

In this example, the connector will connect to an Oracle database. Therefore, values are specified for Database Driver Class Name and Database Type parameters. Additionally, Initial Load Strategy is set to Start at Beginning.

Ensure that you mark the Database Password parameter as a Sensitive Value during creation. This is because only sensitive parameters can be referenced from sensitive processor or controller service properties.

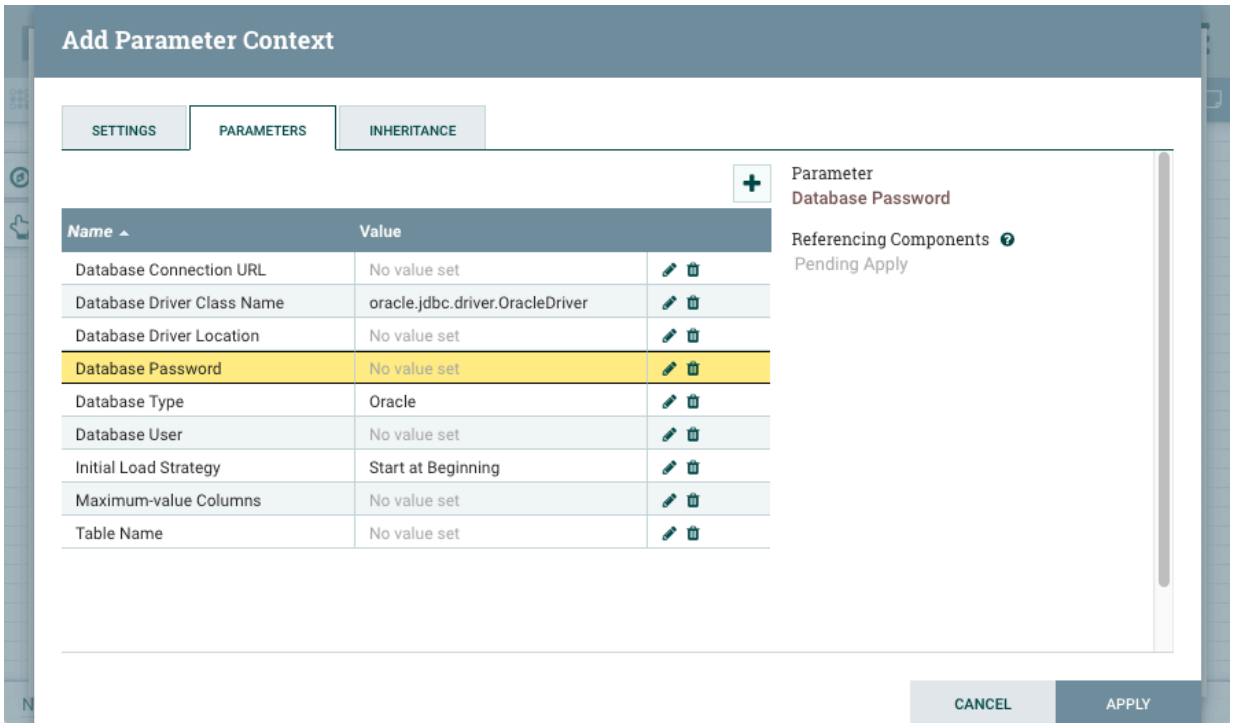

6. Assign the parameter context to the process group.

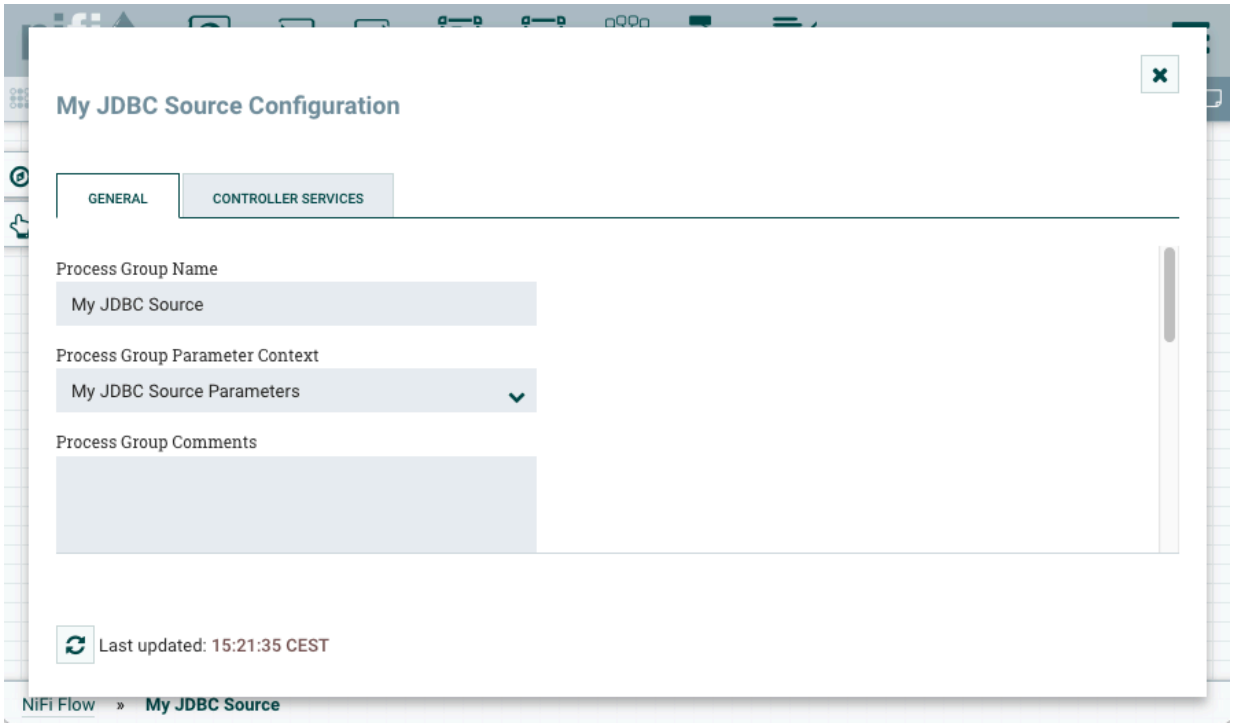

7. Reference the parameters from the processor and the controller services.

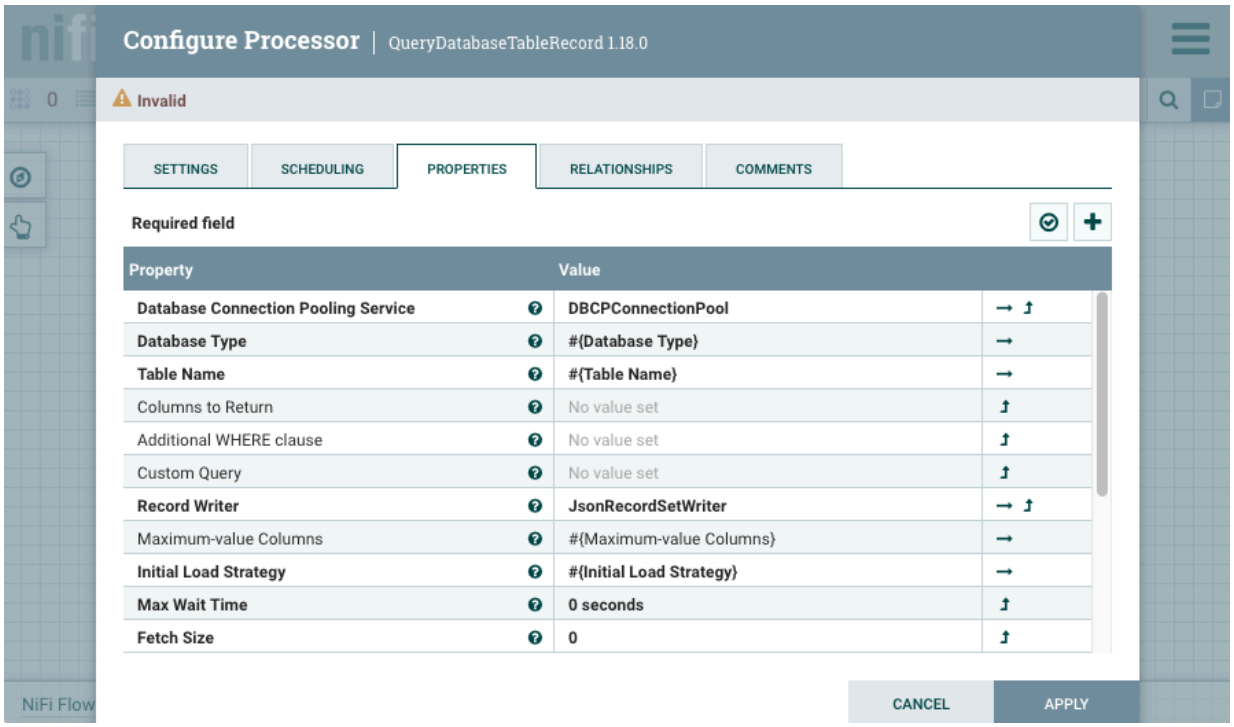

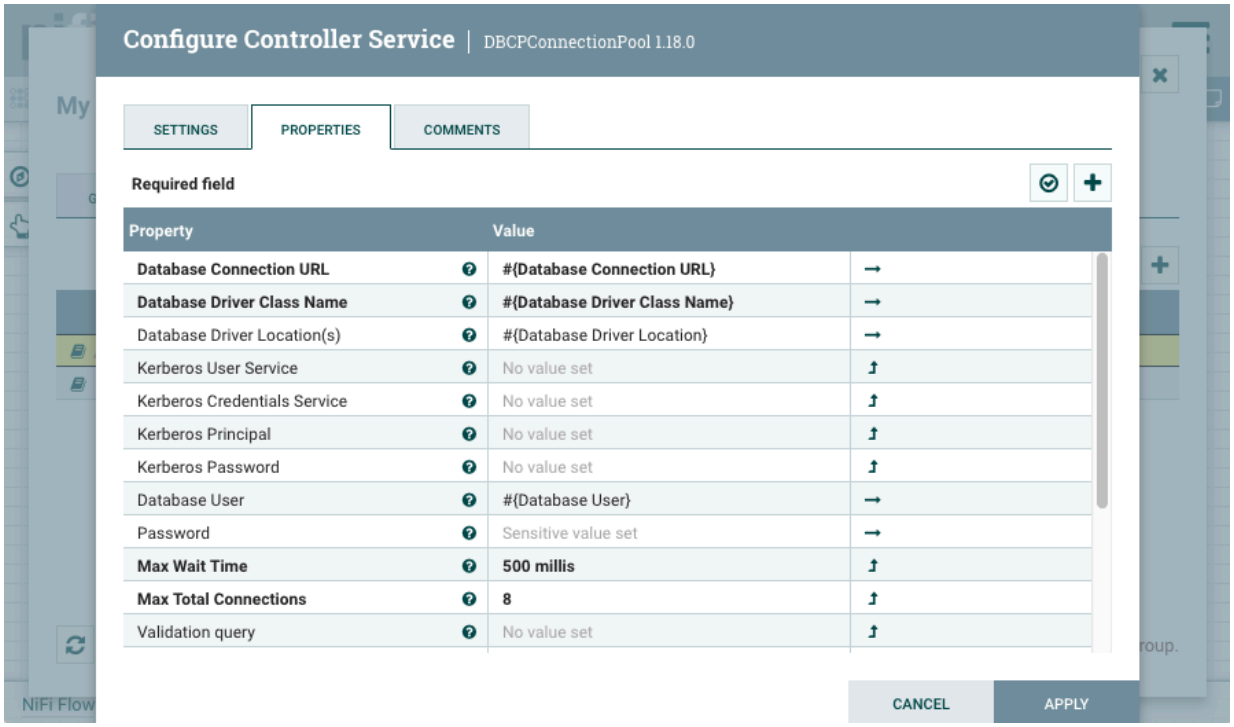

At this point, one controller service is invalid, and the other is disabled, but that is expected. Neither processors, nor controller services need to be valid, started, or enabled at this point.

8. Open the configuration of the QueryDatabaseTableRecord processor, and go to the SCHEDULING tab.

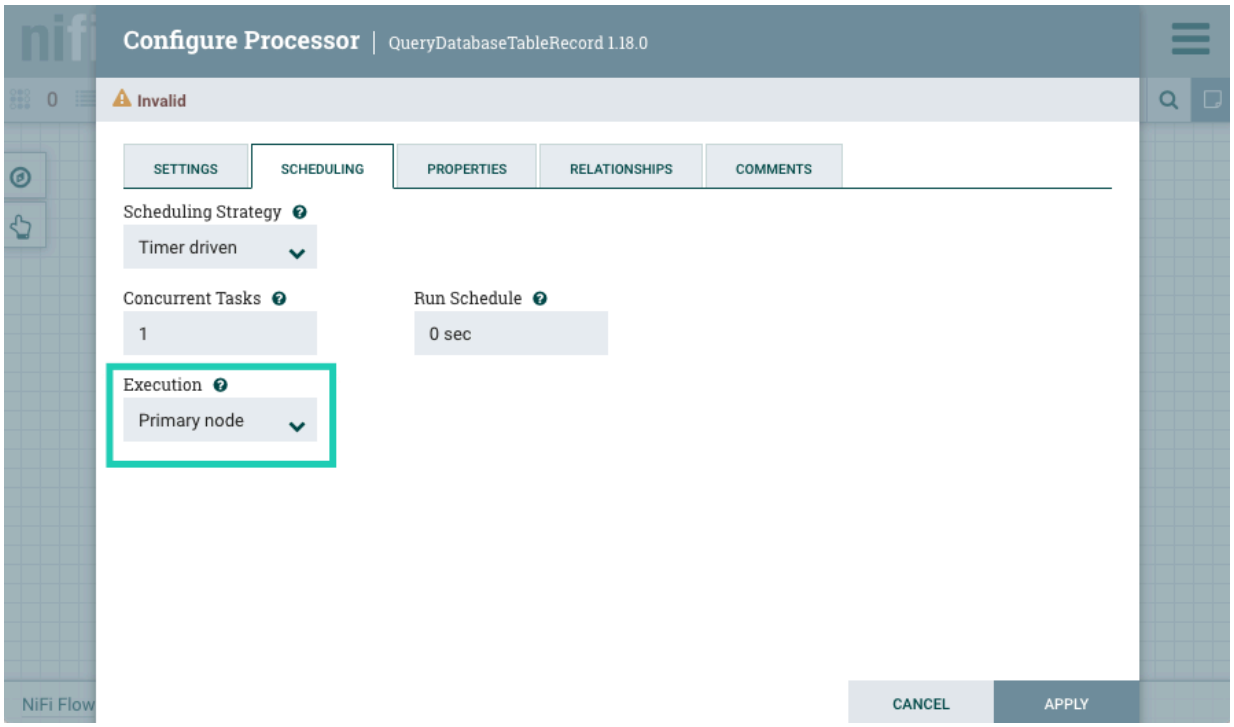

Notice that Execution is set to Primary node (which is the only option for this particular processor). For Stateless NiFi Kafka connectors, this setting means that the connector is going to be deployed with one task, regardless of the value of the tasks.max connector property.

9. At this point, your dataflow is in the following state.

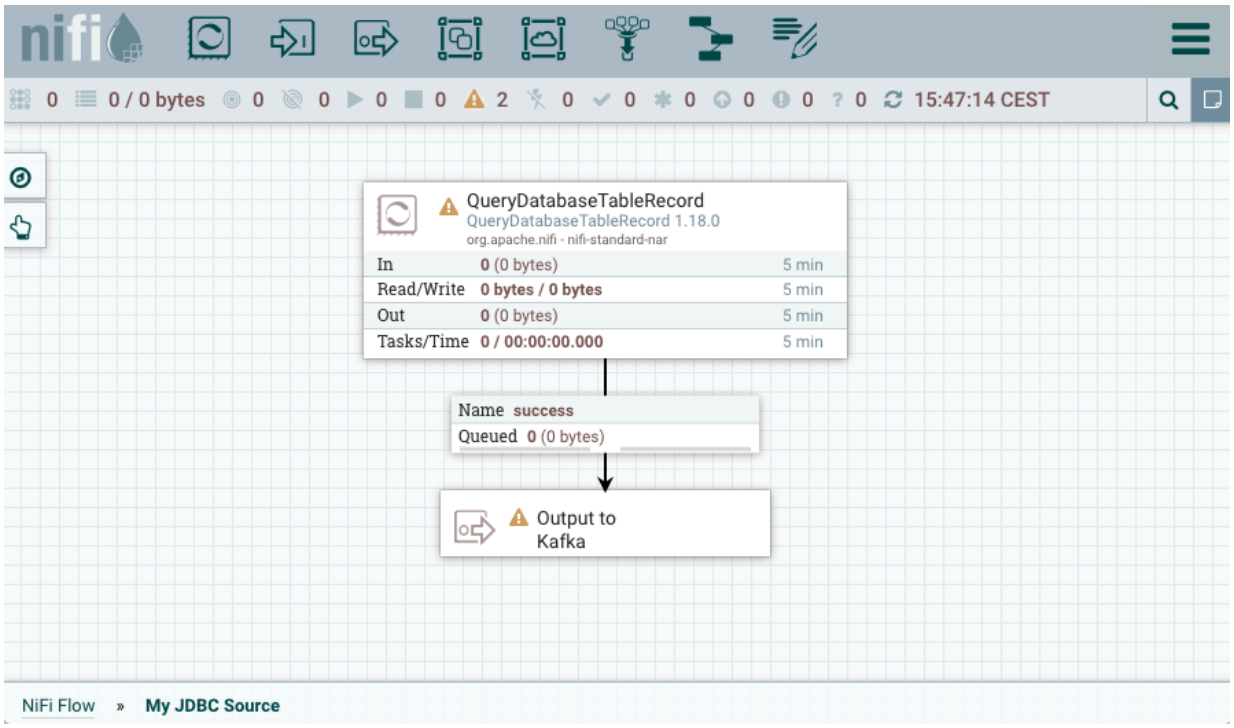

Notice how the dataflow is not running, and there are warnings present. This is expected. The dataflow does not need to run in order for you to be able to download the flow definition. The properties you must configure to make the dataflow function will be set later on during connector deployment.

10. Right-click the process group and select Download flow definition Without external service .

    If you followed these steps, your process group is not referencing controller services that are outside of the process group. As a result, you can choose to download the flow without external services included.

11. Open the downloaded JSON file, and replace **every occurrence** of the version number (1.18.0) with the version number of your Stateless NiFi plugin.


```
{
    "flowContents": {
        "identifier": "7f4e524d-e155-3d50-af2a-d31a49e87a50",
        "instanceIdentifier": "a3db7014-0187-1000-0e07-ae819736fda3",
        "name": "My JDBC Source",
        "comments": "",
        "position": {
            "x": 160.0,
            "y": 48.0
        },
        "processGroups": [],
        "remoteProcessGroups": [],
        "processors": [
            {
                "identifier": "d19b2cf9-f9ef-39c4-8591-51147e432fcd",
                "instanceIdentifier": "a3df4caa-0187-1000-9066-5fa475f8206d",
                "name": "QueryDatabaseTableRecord",
                "comments": "",
                "position": {
                    "x": 64.0,
                    "y": -56.0
                },
                "type": "org.apache.nifi.processors.standard.QueryDatabaseTableRecord",
                "bundle": {
                    "group": "org.apache.nifi",
                    "artifact": "nifi-standard-nar",
                    "version": "1.18.0"
                },
                "properties": {
                    "Table Name": "#{Table Name}",
```


Replace with Stateless NiFi plugin version

12. Access the Streams Messaging Manager UI.
13. Click **Connect** in the navigation sidebar.
14. Click the New Connector option.
15. Select StatelessNiFiSourceConnector from the available connectors.

    The Connector Configuration page appears.

16. Configure the following connector properties.

    - name=My Custom JDBC Connector
    - output.port=Output to Kafka

      This property must be set to the name of the output port in the dataflow. The output port name was specified in Step 2 on page 89.
    - topics=jdbc_example

      This is the name of the target topic that the connector writes data to.

17. Configure the flow.snapshot property.

The flow.snapshot property specifies the dataflow that the connector runs. This is the dataflow that you developed and exported as flow definition JSON in the previous steps. While you have multiple options when configuring this property, this tutorial uses the Import and Enhance option in Streams Messaging Manager.

a) Click ⋮ found next to flow.snapshot, and then click Edit.

b) Paste the contents of the flow definition JSON, or click Browse... to upload the file from your machine.

c) Click Save and Enhance.

Save and Enhance is a unique Streams Messaging Manager feature only available for this property. If this option is used, Streams Messaging Manager parses the dataflow specified in flow.snapshot, extracts all parameters that are available in the dataflow, and adds them to connector configuration.

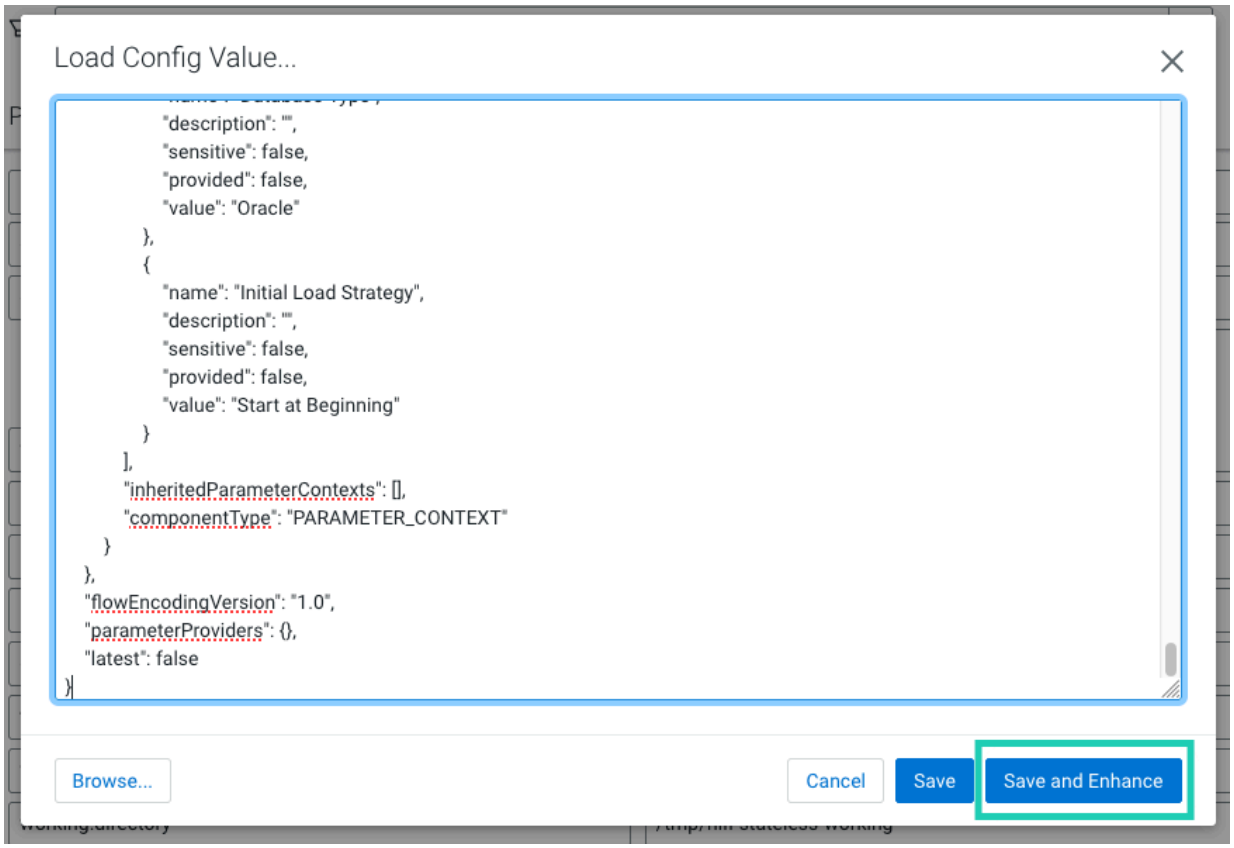

18. Configure the dataflow parameters.

These are the properties that have the parameter. prefix. In this specific example, values for Database Class N ame, Database Type, and Initial Load Strategy are loaded automatically because values were specified within the dataflow in Step 5 on page 90. If needed, you can change the values.

Set Database Driver Location to the path where you deployed the JDBC driver.

19. Review your configuration.

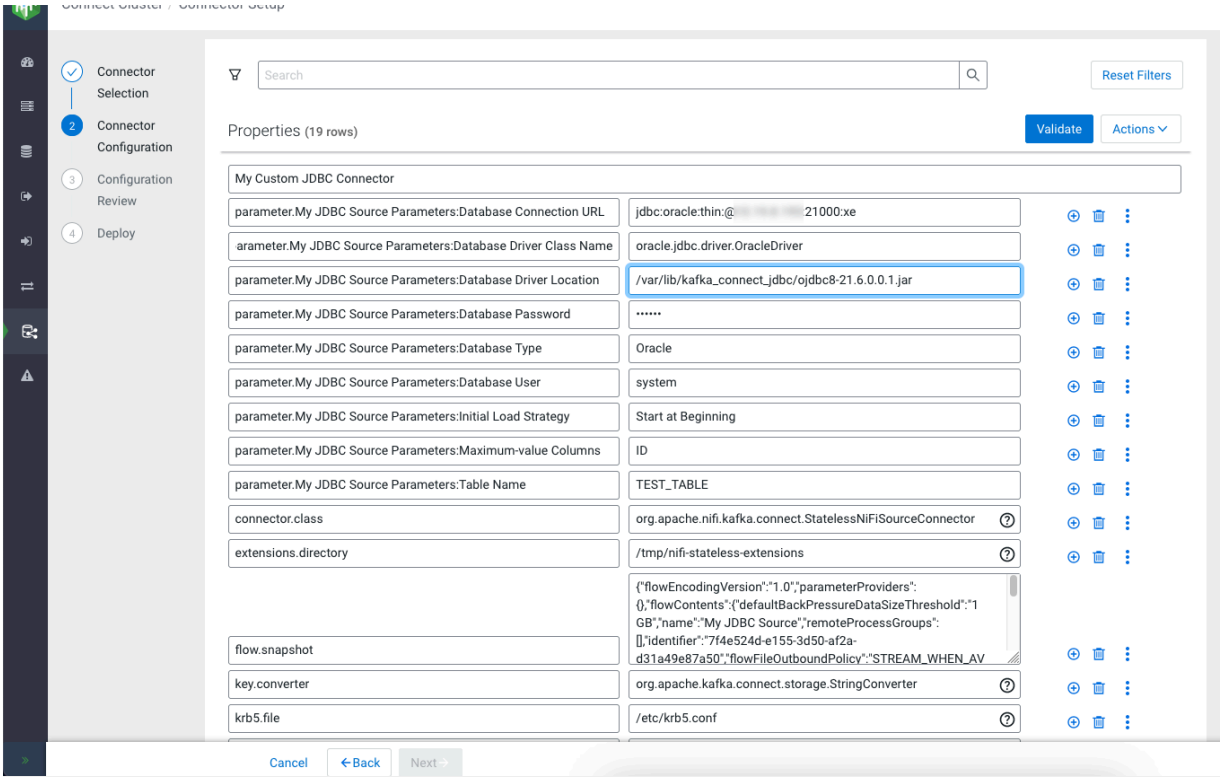

20. Click Validate.

21. Click Next.

22. Click Deploy to deploy the connector.

Connector deployment is not instantaneous. You might need to wait a few seconds for deployment to finish.

Results

Your custom developed flow is deployed in Kafka Connect and is running as a Kafka Connect connector.

What to do next

Once deployment finishes, navigate to the Connector Profile page of the connector. On this page you can view various details regarding the connector. Most importantly, you can view what the connector's status is, and which workers the connector and its task were assigned to.

Note: It might take some time for tasks to appear on this page.

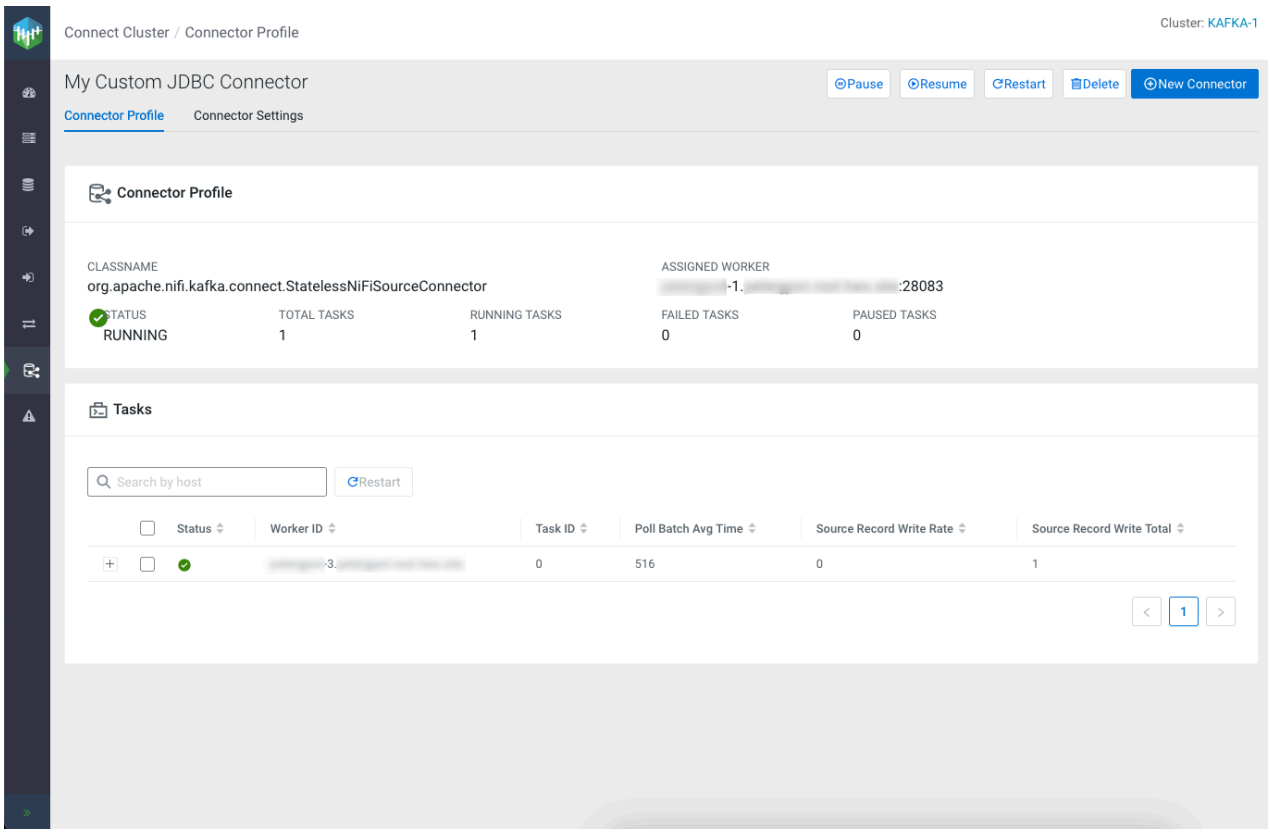

In this particular case, the connector was assigned to the worker running on cluster host 1, whereas the task was assigned to worker running on cluster host 3. This means that dataflow log entries will be present in the Kafka Connect log file on host 3.

In addition to monitoring the connector, you can also check the contents of the topic that the connector is writing to.

This can be done in Streams Messaging Manager by navigating to the Topics page, searching for the topic, and clicking . This opens the Streams Messaging Manager Data Explorer in a modal window. You can use the Data Explorer to sample the data that is flowing through the topic. The database table's records appear in the topic in JSON format.

Related Information

Kafka Connect tasks and Stateless NiFi

Syslog TCP Source connector

The Syslog TCP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The Syslog TCP Source connector listens on a port for syslog messages over TCP and transfers them to Kafka. The connector accepts messages in one of the following formats: Syslog 3164, Syslog 5424, or Grok. If the input messages are in Grok format, the connector can either derive the schema using the field names from the value of the Grok Expression property or read the schema from Schema Registry.

The connector can write messages into Kafka in one of the following formats: Avro, JSON, or text. If the output format is text, the raw message is transferred to Kafka. If the output format is JSON, the input message is processed and converted into JSON format before it is transferred to Kafka. If the output format is Avro, the input message is processed and converted into Avro format before it is transferred to Kafka. How the record schema is attached to these Avro messages depends on the value of the Avro Schema Write Strategy property.

If Schema Registry is used, and it is on a Kerberized cluster, the krb5.file property must point to the krb5.conf file that provides access to the cluster on which Schema Registry is present. This means that the krb5.conf file must be on the same cluster node that the connector runs on. The Kerberos keytab that is used to access Schema Registry must also be on the same cluster node that the connector runs on.

The connection to Schema Registry can be secured by TLS. The truststore file necessary for securing the connection must be on the same cluster node that the connector runs on. Mutual TLS for securing the communication between the message sender and the connector itself is also supported. The keystore and truststore files necessary for securing the connection must be on the same cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Source property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *Syslog TCP Source properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- The Syslog TCP Source connector must use at least one-way SSL. It cannot be used without SSL.
- The Schema Registry URL property is mandatory even if Schema Registry is not used. If Schema Registry is not used, use the default value, or completely remove the property from the configuration JSON.
- Schemas are only read from Schema Registry if Input Data Format is set to GROK and Schema Access Strategy is set to Schema Registry.
- If Output Format is AVRO, the schema of the records can be embedded in the output data. Whether the schema is embedded is determined by the Avro Schema Write Strategy property.
- Schema Branch and Schema Version can not be specified at the same time.

Configuration example

In this example, the connector uses mutual TLS to receive data in Syslog 3164 format which is then transferred to Kafka in JSON format.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "Syslog TCP Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConvert
er",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.Syslog TCP Source Parameters:Port": "[***PORT***]",
 "parameter.Syslog TCP Source Parameters:Input Data Format": "Syslog 3164",
 "parameter.Syslog TCP Source Parameters:Output Format": "JSON",
 "parameter.Syslog TCP Source Parameters:Client Authentication": "REQUIRED",
 "parameter.Syslog TCP Source Parameters:SSL Keystore Filename": "[***THE
 FULLY-QUALIFIED FILENAME OF THE KEYSTORE***]",
 "parameter.Syslog TCP Source Parameters:SSL Keystore Key Password":
 "[***KEYSTORE KEY PASSWORD***]",
 "parameter.Syslog TCP Source Parameters:SSL Keystore Password":
 "[***KEYSTORE PASSWORD***]",
 "parameter.Syslog TCP Source Parameters:SSL Keystore Type": "[***KEYSTORE
 TYPE***]",
 "parameter.Syslog TCP Source Parameters:SSL Truststore Filename": "[***THE
 FULLY-QUALIFIED FILENAME OF THE TRUSTSTORE***]",
 "parameter.Syslog TCP Source Parameters:SSL Truststore Password":
 "[***TRUSTSTORE PASSWORD***]",
 "parameter.Syslog TCP Source Parameters:SSL Truststore Type":
 "[***TRUSTSTORE TYPE***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

Port

The port that the connector listens on for incoming messages.

Input Data Format

Determines what format incoming messages are expected in.

Output Format

Determines the format in which messages are transferred to Kafka.

Client Authentication

Determines if one-way or two-way SSL is used. In this example, this property is set to REQUIRED, meaning that two-way SSL is used.

Keystore *

These are the properties for accessing the keystore containing the keypair used for secure communication.

Truststore *

These are the parameters for accessing the truststore containing the message sender’s certificate used for secure communication.

Related Information

Apache Kafka documentation
Stateless NiFi Source properties reference
Syslog TCP Source properties reference

Syslog UDP Source connector

The Syslog UDP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The Syslog UDP Source connector listens on a port for syslog messages over UDP and transfers them to Kafka. The connector accepts messages in one of the following formats: Syslog 3164, Syslog 5424, or Grok. If the input messages are in Grok format, the connector can either derive the schema using the field names from the value of the Grok Expression property or read the schema from Schema Registry.

The connector can write messages into Kafka in one of the following formats: Avro, JSON, or text. If the output format is text, the raw message is transferred to Kafka. If the output format is JSON, the input message is processed and converted into JSON format before it is transferred to Kafka. If the output format is Avro, the input message is processed and converted into Avro format before it is transferred to Kafka. How the record schema is attached to these Avro messages depends on the value of the Avro Schema Write Strategy property.

If Schema Registry is used, and it is on a Kerberized cluster, the krb5.file property must point to the krb5.conf file that provides access to the cluster on which Schema Registry is present. This means that the krb5.conf file must be on the same cluster node that the connector runs on.

The Kerberos keytab that is used to access Schema Registry must also be on the same cluster node that the connector runs on. The connection to Schema Registry can be secured by TLS. The truststore file necessary for securing the connection must also be on the same cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Source property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *Syslog UDP Source properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- The Schema Registry URL property is mandatory even if Schema Registry is not used. If Schema Registry is not used, use the default value, or completely remove the property from the configuration JSON.
- Schemas are only read from Schema Registry if Input Data Format is set to GROK and Schema Access Strategy is set to Schema Registry.
- If Output Format is AVRO, the schema of the records can be embedded in the output data. Whether the schema is embedded is determined by the Avro Schema Write Strategy property.
- Schema Branch and Schema Version can not be specified at the same time.

Configuration example

In this example, the connector receives data in Syslog 3164 format and transfers it to Kafka in JSON format.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "Syslog UDP Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConvert
er",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.Syslog UDP Source Parameters:Port": "[***PORT***]",
 "parameter.Syslog UDP Source Parameters:Input Data Format": "Syslog 3164",
 "parameter.Syslog UDP Source Parameters:Output Format": "JSON",
 "parameter.Syslog UDP Source Parameters:Output Grouping for JSON": "output-
oneline"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

Port

The port that the connector listens on for incoming messages.

Input Data Format

Determines what format incoming messages are expected in.

Output Format

Determines the format in which messages are transferred to Kafka.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

Syslog UDP Source properties reference

ADLS Sink connector

The ADLS Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka connect framework. Learn about the connector, its properties, and configuration.

The ADLS Sink connector fetches messages from Kafka and uploads them to ADLS. The topic this connector receives messages from is determined by the value of the topics property in the configuration. The messages can contain unstructured (character or binary) data or they can be in Avro or JSON format.

If the input is unstructured data, record processing is disabled. In a case like this, multiple messages can be concatenated into a single output file on ADLS using a demarcator (for example, newline for text messages). Merging is optional, large binary data can be forwarded to Azure at a 1:1 ratio, one Kafka message equals a single ADLS file.

If the input is either Avro or JSON, record processing is enabled. In a case like this, the schema of the records can be a predefined schema retrieved from Schema Registry (for both Avro and JSON data), it can be embedded in the Avro data, or inferred from the JSON data. The output can be Avro, JSON or CSV. Multiple records are typically merged into a single output file before uploading to ADLS.

The connector can authenticate to Azure using an Account Key, SAS Token, Service Principal, or a Managed Identity.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Sink property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *ADLS Sink properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- Schema Branch and Schema Version can not be specified at the same time.

Configuration example for fetching unstructured data

In this example, the connector fetches unstructured messages containing single line character data from Kafka. The connector concatenates the messages using newline characters as the demarcator. The files that the connector creates and uploads to ADLS are maximum 10 MB in size. The files are named according to the following pattern: mess ages_*[***UUID***]*. The connector authenticates to Azure using an Account Key.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "ADLS Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "failure.ports": "PutAzureDataLakeStorage Failure",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.ADLS Sink Parameters:ADLS Account Name": "[***ACCOUNT NAME***]",
 "parameter.ADLS Sink Parameters:ADLS Account Key": "[***ACCOUNT KEY***]",
 "parameter.ADLS Sink Parameters:ADLS Filesystem Name": "[***FILESYSTEM
 NAME***]",
 "parameter.ADLS Sink Parameters:ADLS Directory Name": "[***DIRECTORY
 NAME***]",
 "parameter.ADLS Sink Parameters:Maximum File Size": "10 MB",
 "parameter.ADLS Sink Parameters:Kafka Message Data Format": "Raw",
 "parameter.ADLS Sink Parameters:Output File Demarcator": "\n",
 "parameter.ADLS Sink Parameters:Output Filename Pattern": "messages_${file
name.uuid}"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

ADLS Account Name

The Storage Account Name to use for authentication to Azure. This is also the target account where the output file is uploaded.

ADLS Account Key

The Storage Account Key to use for authentication to Azure.

ADLS Filesystem Name

The filesystem (or container) in the Storage Account to upload data to.

ADLS Directory Name

The target directory in the filesystem.

Maximum File Size

The maximum size of the output data file. In this example, the maximum size is 10 MB.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to Raw. This means that the connector expects unstructured text or binary data.

Output File Demarcator

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages. In this example, the property is set to \n (newline). This means that the newline character is used to separate the single line character data of the Kafka messages.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to messages_${filename.uuid}.

messages_ is a string literal. ${filename.uuid} is an expression that inserts a generated UUID in the filename. As a result, the files are named according to the following pattern: messages_*[***UUID***]*.

Configuration example for fetching JSON data

In this example, the connector fetches messages in JSON format from Kafka. The connector parses the JSON records and converts them to Avro format using the schema inferred from the JSON data. The schema is also embedded in the output file. The connector merges multiple records (all messages that are available for a single execution of the connector) into an Avro file, generates a name for the file using the following pattern data_*[***TIMESTAMP***]*_*[***SEQUENCE***]*.avro, and uploads the file to ADLS. The connector authenticates to Azure using a Service Principal.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "ADLS Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "failure.ports": "PutAzureDataLakeStorage Failure",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.ADLS Sink Parameters:ADLS Account Name": "[***ACCOUNT NAME***]",
 "parameter.ADLS Sink Parameters:Azure Service Principal Tenant ID":
 "[***SERVICE PRINCIPAL TENANT ID***]",
 "parameter.ADLS Sink Parameters:Azure Service Principal Client ID":
 "[***SERVICE PRINCIPAL CLIENT ID***]",
 "parameter.ADLS Sink Parameters:Azure Service Principal Client Secret":
 "[***SERVICE PRINCIPAL CLIENT SECRET***]",
 "parameter.ADLS Sink Parameters:ADLS Filesystem Name": "[***FILESYSTEM
 NAME***]",
 "parameter.ADLS Sink Parameters:ADLS Directory Name": "[***DIRECTORY
 NAME***]",
 "parameter.ADLS Sink Parameters:Kafka Message Data Format": "JSON",
 "parameter.ADLS Sink Parameters:Schema Access Strategy": "Infer Schema",
 "parameter.ADLS Sink Parameters:Avro Schema Write Strategy": "Embed Avro
Schema",
 "parameter.ADLS Sink Parameters:Output File Data Format": "Avro",
 "parameter.ADLS Sink Parameters:Output Filename Pattern": "data_${filename.
timestamp}_${filename.sequence}.avro",
 "parameter.ADLS Sink Parameters:Output Filename Timestamp Format": "yyyy
MMdd_HHmmss_SSS"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

ADLS Account Name

The Storage Account Name to use for authentication to Azure. This is also the target account where the output file is uploaded.

Service Principal *

The properties of the Service Principal to use for authentication to Azure.

ADLS Filesystem Name

The filesystem (or container) in the Storage Account to upload data to.

ADLS Directory Name

The target directory in the filesystem.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to JSON. This means that the connector expects JSON data.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, this property is set to Infer Schema, meaning that the schema is determined (inferred) from the JSON data.

Avro Schema Write Strategy

Specifies whether the record schema is written to the output data file. In this example, this property is set to Embed Avro Schema, meaning that the schema is embedded in the output Avro file.

Output File Data Format

Specifies the format of the records written to the output file. In this example, this property is set to Avro, meaning that the output file format is Avro.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to data_${filename.timestamp}_${filename.sequence}.avro.

data_, the underscore (_), and .avro are string literals. ${filename.timestamp} and ${filename.sequence} are expressions. ${filename.timestamp} inserts the current timestamp in the filename. ${filename.sequence} inserts an incrementing sequence value in the filename. As a result, the files are named according to the following pattern: data*_[***TIMESTAMP***]_[***SEQUENCE***]*.avro.

Output Filename Timestamp Format

Specifies the timestamp format used in the ${filename.timestamp} expression. The expression is used when generating the output filename.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

ADLS Sink properties reference

Amazon S3 Sink

Learn more about the Amazon S3 Sink connector

The Amazon S3 Sink connector allows users to stream Kafka data into S3 buckets.

Configuration example

A simple configuration example for the Amazon S3 Sink connector.

The following is a simple configuration example for the Amazon S3 Sink connector. Short descriptions of the properties set in this example are also provided. For a full properties reference, see the *Amazon S3 Sink properties reference*.

```
{
    "aws.s3.bucket": "bring-me-the-bucket",
    "aws.s3.service_endpoint": "http://myendpoint:9090/",
    "aws.access_key_id": "EXAMPLEID",
    "aws.secret_access_key": “EXAMPLEKEY",
    "connector.class": "com.cloudera.dim.kafka.connect.s3.S3SinkConnector",
    "tasks.max": 1,
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConver
ter",
    "value.converter.passthrough.enabled": true,
    "value.converter.schema.registry.url": "http://schema-registry:9090/api/
v1",
    "topics": "avro_topic",
    "output.storage": "com.cloudera.dim.kafka.connect.s3.S3PartitionStorage"
,
    "output.writer": "com.cloudera.dim.kafka.connect.partition.writers.avr
o.AvroPartitionWriter",
    "output.avro.passthrough.enabled": true
  }
```

aws.s3.bucket

Target S3 bucket name.

aws.s3.service_endpoint

Target S3 host and port.

aws.access_key_id

The AWS secret key ID used for authentication.

aws.secret_access_key

The AWS secret access key used for authentication.

connector.class

Class name of the Amazon S3 Sink connector.

tasks.max

Maximum number of tasks.

key.converter

The converter capable of understanding the data format of the key of each record on this topic.

value.converter

The converter capable of understanding the data format of the value of each record on this topic.

Note: When the AvroConverter is used, you can specify Schema Registry properties to be used by the AvroConverter’s Schema Registry client. This is done by adding the required Schema Registry property as a suffix to the value.converter property. For example, value.converter.schema.registry.url. Properties defined this way are passed on to the Schema Registry client used by the AvroConverter.

value.converter.passthrough.enabled

This property controls whether or not data is converted into the Kafka Connect intermediate data format before writing into an output file. Because in this example the input and output format is the same, the property is set to true, that is, data is not converted.

value.converter.schema.registry.url

The URL to Schema Registry. This is a mandatory property if the topic has records encoded in Avro format.

topics

List of topics to consume data from.

output.storage

The S3 storage implementation class.

output.writer

Determines the output file format. Because in this example the output format is Avro, AvroPartitionWriter is used.

output.avro.passthrough.enabled

This property has to match the configuration of the value.converter.passthrough.enabled property because both the input and output formats are Avro.

Related Information

Amazon S3 Sink Connector Properties Reference

HDFS Sink

Learn more about the HDFS Sink connector.

The HDFS Sink connector can be used to transfer data from Kafka topics to files on HDFS clusters. Each partition of every topic results in a collection of files named in the following pattern:

```
{topic name}_{partition number}_{end_offset}.{file extension}
```

For example, running the HDFS Sink connector on partition 0 of a topic named sourceTopic can yield the following series of files:

```
sourceTopic_0_50.avro - for record 0 ~ 50
sourceTopic_0_79.avro - holding record 51 ~ 79
...
```

The HDFS Sink connector periodically commits records to final result files. Each commit results in a separate "chunk" file.

Configuration example for writing data to HDFS

A simple configuration example for the HDFS Sink connector.

The following is a simple configuration example for the HDFS Sink connector. Short descriptions of the properties set in this example are also provided. For a full properties reference, see the *HDFS Sink properties reference*.

```
{
    "connector.class": "com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConnec
tor",
    "tasks.max": 1,
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConve
rter",
    "value.converter.passthrough.enabled": true,
    "value.converter.schema.registry.url": "http://localhost:9090/api/v1",
    "topics": "avro_topic",
```

```
    "hdfs.uri": "hdfs://my-host.my-realm.com:8020",
    "hdfs.output": "/topics_output/",
    "output.writer": "com.cloudera.dim.kafka.connect.partition.writers.avro
.AvroPartitionWriter",
    "output.avro.passthrough.enabled": true,
    "hdfs.kerberos.authentication": true,
    "hdfs.kerberos.user.principal": "user_account@MY-REALM.COM",
    "hdfs.kerberos.keytab.path": "/path/to/user_account.keytab",
    "hdfs.kerberos.namenode.principal": "hdfs/_HOST@MY-REALM.COM",
    "hadoop.conf.path": "/etc/hadoop/"
  }
```

connector.class

Class name of the HDFS Sink connector.

key.converter

The converter capable of understanding the data format of the key of each record on this topic.

value.converter

The converter capable of understanding the data format of the value of each record on this topic.

Note: When the AvroConverter is used, you can specify Schema Registry properties to be used by the AvroConverter's Schema Registry client. This is done by adding the required Schema Registry property as a suffix to the value.converter property. For example, value.converter.schema.registry.url. Properties defined this way are passed on to the Schema Registry client used by the AvroConverter.

value.converter.passthrough.enabled

This property controls whether or not data is converted into the Kafka Connect intermediate data format before writing into an output file. Because in this example the input and output format is the same, the property is set to true, that is, data is not converted.

value.converter.schema.registry.url

The URL to Schema Registry. This is a mandatory property if the topic has records encoded in Avro format.

topics

List of topics to consume data from.

hdfs.uri

The URI to the namenode of the HDFS cluster.

hdfs.output

The destination folder on the HDFS cluster where output files will reside.

output.writer

Determines the output file format. Because in this example the output format is Avro, AvroPartitio nWriter is used.

output.avro.passthrough.enabled

This property has to match the configuration of the value.converter.passthrough.enabled property because both the input and output formats are Avro.

hdfs.kerberos.authentication

Enables or disables kerberos authentication.

hdfs.kerberos.user.principal

The user principal that the Kafka Connect role will use.

hdfs.kerberos.keytab.path

The path to the kerberos keytab file.

hdfs.kerberos.namenode.principal

The Kerberos principal used by the namenode. This is necessary when the HDFS cluster has data encryption turned on.

hadoop.conf.path

The path to the hadoop configuration files. This is necessary when the HDFS cluster has data encryption turned on.

Related Information

HDFS Sink Connector Properties Reference

Configuration example for writing data to Ozone FS

A simple configuration example for the HDFS Sink connector that writes data to the Ozone FS.

The following is a simple configuration example for the HDFS Sink connector. In this example data is written to the Ozone FS. Short descriptions of the properties set in this example are also provided. For a full properties reference, see the *HDFS Sink properties reference*.

```
{
    "connector.class": "com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConnec
tor",
    "tasks.max": 1,
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConve
rter",
    "value.converter.passthrough.enabled": true,
    "value.converter.schema.registry.url": "http://localhost:9090/api/v1",
    "topics": "avro_topic",
    "hdfs.uri": "ofs://ozone1/volume1/bucket1/",
    "hdfs.output": "/topics_output/",
    "output.writer": "com.cloudera.dim.kafka.connect.hdfs.avro.AvroPartiti
onWriter",
    "output.avro.passthrough.enabled": true,
    "hdfs.kerberos.authentication": true,
    "hdfs.kerberos.user.principal": "user_account@MY-REALM.COM",
    "hdfs.kerberos.keytab.path": "/path/to/user_account.keytab",
    "hadoop.conf.path": "/etc/hadoop/"
  }
```

connector.class

Class name of the HDFS Sink connector.

key.converter

The converter capable of understanding the data format of the key of each record on this topic.

value.converter

The converter capable of understanding the data format of the value of each record on this topic.

Note: When the AvroConverter is used, you can specify Schema Registry properties to be used by the AvroConverter's Schema Registry client. This is done by adding the required Schema Registry property as a suffix to the value.converter property. For example, value.converter.schema.registry.url. Properties defined this way are passed on to the Schema Registry client used by the AvroConverter.

value.converter.passthrough.enabled

This property controls whether or not data is converted into the Kafka Connect intermediate data format before writing into an output file. Because in this example the input and output format is the same, the property is set to true, that is, data is not converted.

value.converter.schema.registry.url

The URL to Schema Registry. This is a mandatory property if the topic has records encoded in Avro format.

topics

List of topics to consume data from.

hdfs.uri

The Ozone FS (ofs) URI.

hdfs.output

The destination folder on the HDFS cluster where output files will reside.

output.writer

Determines the output file format. Because in this example the output format is Avro, AvroPartitionWriter is used.

output.avro.passthrough.enabled

This property has to match the configuration of the value.converter.passthrough.enabled property because both the input and output formats are Avro.

hdfs.kerberos.authentication

Enables or disables kerberos authentication.

hdfs.kerberos.user.principal

The user principal that the Kafka Connect role will use.

hdfs.kerberos.keytab.path

The path to the kerberos keytab file.

hadoop.conf.path

The path to the hadoop configuration files. This is necessary when the HDFS cluster has data encryption turned on.

HDFS Stateless Sink connector

The HDFS Stateless Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect Framework. Learn about the connector, its properties, and configuration.

The HDFS Stateless Sink Connector fetches messages from Kafka and stores them in HDFS. The topic this connector receives messages from is determined by the value of the topics parameter in the configuration. The messages can contain unstructured data (character or binary) or they can be in Avro or JSON format.

If the input is unstructured data, record processing is disabled. Multiple messages can be concatenated into a single output file on HDFS using a demarcator (for example newline for text messages). Merging is optional, large binary data can be forwarded to HDFS at a 1:1 ratio, one Kafka message equals a single HDFS file. However, in case of large files, this usage is encouraged only to avoid small file problems in HDFS.

If the input is either Avro or JSON, record processing is enabled. In a case like this, the schema of the records can be a predefined schema retrieved from Schema Registry (for both Avro and JSON data), it can be embedded in the Avro data, or inferred from the JSON data. The output can be Avro, JSON, CSV, or Parquet. Multiple records are typically merged into a single output file to store in HDFS.

The connector supports connecting to Kerberos enabled HDFS and Schema Registry services.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Sink property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *HDFS Stateless Sink properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- If HDFS is running on a different cluster than the connector, the HDFS configuration resource files must be copied to the Kafka Connect node that the connector runs on.

Configuration example for unstructured data

In this example, the connector fetches unstructured messages from Kafka. The connector concatenates the messages using newline characters as the demarcator. The files that the connector creates and stores in HDFS are maximum 10 MB in size. The files are named according to the following pattern: messages_*[***UUID***]*.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
"connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnect
or",
"meta.smm.predefined.flow.name": "HDFS Sink",
"meta.smm.predefined.flow.version": "1.0.0",
"key.converter": "org.apache.kafka.connect.storage.StringConverter",
"value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter",
"tasks.max": "1",
"nexus.url": "https://repository.cloudera.com/artifactory/repo/",
"working.directory": "/tmp/nifi-stateless-working",
"failure.ports": "",
"topics": "[***TOPIC NAME***]",
"parameter.HDFS Sink Parameters:Hadoop Configuration Resources": "/etc/ha
doop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml",
"parameter.HDFS Sink Parameters:Kafka Message Data Format": "Raw",
"parameter.HDFS Sink Parameters:Output Directory Pattern": "[***OUTPUT
 DIRECTORY PATH***]",
"parameter.HDFS Sink Parameters:Output Filename Pattern": "messages_${fil
ename.uuid}",
```

```
"parameter.HDFS Sink Parameters:Maximum File Size": "10 MB",
"parameter.HDFS Sink Parameters:Output File Demarcator": " \n"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

Hadoop Configuration Resources

A comma delimited list of Hadoop configuration files (core-site.xml, hdfs-site.xml, and so on). If HDFS is running on a different cluster than the connector, ensure that you copy the configuration files to the Kafka Connect node that the connector runs on.

Note: In the majority of cases, these files can be found in/etc/hadoop/conf.

Output Directory Pattern

Specifies the path to the output directory in HDFS.

Maximum File Size

The maximum size of the output data file. In this example, the maximum size is 10 MB.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to Raw. This means that the connector expects unstructured text or binary data and that record processing is disabled.

Output File Demarcator

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages. In this example, the property is set to \n (newline). This means that the newline character is used to separate the single line character data of the Kafka messages.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to messages_${filename.uuid}. The word messages and the underscore (_) are string literals. ${filename.uuid} is an expression, which inserts a Universal Unique Identifier (UUID). As a result, the files are named according to the following pattern: messages_*[***UUID***]*.

Configuration example for Avro input and JSON output

In this example, the connector fetches messages in Avro format from Kafka. The connector parses the Avro records and converts them to JSON format using the schema definition stored in Schema Registry. The Avro file does not contain any schema reference. This is the reason why the schema must be specified in the connector parameters. The connector merges multiple records (all messages that are available for a single execution of the connector) into a JSON file, generates a name for the file using the following pattern: messages_*[***UUID***]*.json, and stores the file in HDFS.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
"connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnect
or",
"meta.smm.predefined.flow.name": "HDFS Sink",
"meta.smm.predefined.flow.version": "1.0.0",
"key.converter": "org.apache.kafka.connect.storage.StringConverter",
"value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter",
```

```
"tasks.max": "1",
"nexus.url": "https://repository.cloudera.com/artifactory/repo/",
"working.directory": "/tmp/nifi-stateless-working",
"failure.ports": "",
"topics": "[***TOPIC NAME***]",
"parameter.HDFS Sink Parameters:Hadoop Configuration Resources": "/etc/ha
doop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml",
"parameter.HDFS Sink Parameters:Kafka Message Data Format": "Avro",
"parameter.HDFS Sink Parameters:Output Directory Pattern": "[***OUTPUT
 DIRECTORY NAME***]",
"parameter.HDFS Sink Parameters:Output File Data Format": "JSON",
"parameter.HDFS Sink Parameters:Output Filename Pattern": "messages_${filen
ame.uuid}.json",
"parameter.HDFS Sink Parameters:Schema Access Strategy": "Schema Registry",
"parameter.HDFS Sink Parameters:Schema Name": "[***SCHEMA NAME***]"
"parameter.HDFS Sink Parameters:Schema Registry URL": "[***SCHEMA REGISTRY
 SERVER URL***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

Hadoop Configuration Resources

A comma delimited list of Hadoop configuration files (core-site.xml, hdfs-site.xml, and so on). If HDFS is running on a different cluster than the connector, ensure that you copy the configuration files to the Kafka Connect node that the connector runs on.

Note: In the majority of cases, these files can be found in/etc/hadoop/conf.

Output Directory Pattern

Specifies the path to the output directory in HDFS.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to Avro. This means that the connector expects Avro data and that record processing is enabled.

Output File Data Format

Specifies the format of the records written to the output file. In this example, this property is set to JSON, meaning that the output file format is JSON.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to messages_${filename.uuid }.json. The word messages, the underscore (_), and .json are string literals. ${filename.uuid} is an expression, which inserts a Universal Unique Identifier (UUID). As a result, the files are named according to the following pattern: messages_*[***UUID***]*.json.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, this property is set to Schema Registry, meaning that the schema definition used to parse input data is taken from the Schema Registry.

Schema Name

Specifies the schema name to look up in Schema Registry.

Schema Registry URL

The URL of the Schema Registry server.

Note: Although not shown in this example, if Schema Registry is deployed on a secure cluster, the following Kerberos and/or TLS/SSL properties must also be set:

- Truststore Filename for Schema Registry
- Truststore Password for Schema Registry
- Truststore Type for Schema Registry
- Kerberos Keytab for Schema Registry
- Kerberos Principal for Schema Registry

Configuration example for JSON input and Parquet output

In this example, the connector fetches messages in JSON format from Kafka. The connector parses the JSON records and converts them to Parquet format using the schema inferred from the JSON data. The connector merges multiple records (all messages that are available for a single execution of the connector) into a Parquet file, generates a name for the file using the following pattern: messages*_[***TIMESTAMP***]_[***SEQUENCE***]*.parquet.gz, and stores the files in HDFS. The output directory path contains a fixed name and a timestamp appended to it: /outputDir/*[***TIMESTAMP***]*. The connector authenticates to HDFS using Kerberos.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
"connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnect
or",
"meta.smm.predefined.flow.name": "HDFS Sink",
"meta.smm.predefined.flow.version": "1.0.0",
"key.converter": "org.apache.kafka.connect.storage.StringConverter",
"value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter",
"tasks.max": "1",
"nexus.url": "https://repository.cloudera.com/artifactory/repo/",
"working.directory": "/tmp/nifi-stateless-working",
"failure.ports": "",
"topics": "[***TOPIC NAME***]",
"parameter.HDFS Sink Parameters:Hadoop Configuration Resources": "/etc/ha
doop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml",
"parameter.HDFS Sink Parameters:Kafka Message Data Format": "JSON",
"parameter.HDFS Sink Parameters:Output Directory Pattern": "/outputDir/${di
rectory.timestamp}",
"parameter.HDFS Sink Parameters:Output Directory Timestamp Format":"yyMMdd",
"parameter.HDFS Sink Parameters:Output File Data Format": "Parquet",
"parameter.HDFS Sink Parameters:Output Filename Pattern": "messages_${fil
ename.timestamp}_${filename.sequence}.parquet.gz",
"parameter.HDFS Sink Parameters:Output Filename Timestamp Format”: “HHmmss”,
"parameter.HDFS Sink Parameters:Schema Access Strategy": "Infer Schema",
"parameter.HDFS Sink Parameters:Kerberos Keytab for HDFS”:”[***KEYTAB
 LOCATION***]”,
"parameter.HDFS Sink Parameters:Kerberos Principal for HDFS”:”[***PRINCIPAL
 NAME***]”,
"parameter.HDFS Sink Parameters:Compression Codec for Parquet”:”GZIP”
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

Hadoop Configuration Resources

A comma delimited list of Hadoop configuration files (core-site.xml, hdfs-site.xml, and so on). If HDFS is running on a different cluster than the connector, ensure that you copy the configuration files to the Kafka Connect node that the connector runs on.

Note: In the majority of cases, these files can be found in/etc/hadoop/conf.

Output Directory Pattern

Specifies the path to the output directory in HDFS. In this example, the path is set to /outputDir/${directory.timestamp}. As a result, the directory path in this example resolves to /outputDir/*[***TIMESTAMP***]*. The timestamp format is configured in the Output Directory Timestamp Format property.

Output Directory Timestamp Format

Specifies the timestamp format used in the ${directory.timestamp} expression, which can be used in the Output Directory Pattern property's value. Directory Time Based Partitioning can be implemented using this expression. For example, if Output Directory Pattern is set to /outputDir/${directory.timestamp} and the timestamp format specified is yyMMdd, the output files will be generated in the /outputDir/220101/ directory if the date is January 1, 2022.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to JSON. This means that the connector expects JSON data and that record processing is enabled.

Output File Data Format

Specifies the format of the records written to the output file. In this example, this property is set to Parquet, meaning that the output file format is Parquet.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to messages_${filename.timestamp}_${filename.sequence}.parquet.gz.The word messages, the underscores (_), and parquet.gz are string literals. ${filename.timestamp} and ${filename.sequence} are expressions. ${filename.timestamp} inserts the current timestamp in the filename. ${filename.sequence} inserts an incrementing sequence value in the filename. As a result, the files are named according to the following pattern: messages_*[***TIMESTAMP***]*_*[***SEQUENCE***]*.parquet.gz.

Output Filename Timestamp Format

Specifies the timestamp format used in the ${filename.timestamp} expression. The expression is used when generating the output filename. In this example, the format is HHmmss.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, this property is set to Infer Schema, meaning that the schema is determined (inferred) from the JSON data.

Kerberos Keytab for HDFS

Specifies the keytab used to authenticate to HDFS with Kerberos.

Kerberos Principal for HDFS

Specifies the principal used to authenticate to HDFS with Kerberos.

Compression Codec

Specifies the codec used to compress files and store them in compressed format in HDFS, in the example the GZIP codec is used.

Related Information

Apache Kafka documentation

HDFS Stateless Sink properties reference

Stateless NiFi Sink properties reference

HTTP Sink connector

The HTTP Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The HTTP Sink connector obtains messages from a Kafka topic and transfers their content in a HTTP POST requests to a specified endpoint. The topic the connector receives messages from is determined by the value of the topics parameter in the configuration. The connector can forward the data it reads from Kafka as is (raw data) or can be configured to execute record processing. When record processing is enabled, the connector expects the incoming data in Avro or JSON format. In case of Avro, the connector can either read the record schema from the Avro file it receives (provided that the schema is embedded) or it can fetch the schema from Schema Registry. In case of JSON, it can either infer the schema or fetch it from Schema Registry. What strategy is used to retrieve the schema is determined by the Schema Access Strategy property.

If Schema Registry is used, and it is on a Kerberized cluster, the krb5.file property must point to the krb5.conf file that provides access to the cluster on which Schema Registry is present. This means that the krb5.conf file must be on the same cluster node that the connector runs on. The connections to Schema Registry and the HTTP server can be secured by TLS. The keystore and truststore files necessary for securing these connections must also be on the same cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Sink property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *HTTP Sink properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- Schema Branch and Schema Version can not be specified at the same time.

- The value of the Schema Access Strategy property is not independent of the value of the Kafka Message Data Format property. As a result, you must exercise caution when configuring Schema Access Strategy.
  - If the value of Kafka Message Data Format is AVRO, the possible values for Schema Access Strategy are Schema Registry, Embedded Schema or HWX Content-Encoded Schema Reference.
  - If the value of Kafka Message Data Format is JSON, the possible values for Schema Access Strategy are Schema Registry or Infer Schema.

Configuration example

In this example, the connector receives data in any format and forwards the raw data as the content of an HTTP POST request.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnector",
 "meta.smm.predefined.flow.name": "HTTP Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.converters.ByteArrayConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "input.port": "Input from Kafka",
 "failure.ports": "Failure",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.HTTP Sink Parameters:Forward Raw Data": "true",
 "parameter.HTTP Sink Parameters:Remote URL": "http://[***SERVER HOSTNAME***]:[***PORT***]/[***PATH***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

Forward Raw Data

Specifies whether messages from Kafka should be forwarded as is or converted to JSON. In this example, the property is set to true, meaning that the connector does not process any records. It forwards incoming data as is.

Remote URL

Identifies the HTTP endpoint that receives the messages sent by this connector. In this example, SSL is not used. As a result, the URL starts with http not https. For example, http://my.http-server.com:22000/contentListener.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

HTTP Sink properties reference

InfluxDB Sink connector

The InfluxDB Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect Framework. Learn about the connector, its properties, and configuration.

Important: The InfluxDB Sink connector only supports InfluxDB v1.

The InfluxDB Sink Connector fetches messages from Kafka and loads them to InfluxDB. The topic this connector receives messages from is determined by the value of the topics parameter in the configuration. The messages the connector receives from Kafka can be in either Avro or JSON format.

If the connector input is in Avro format, then it can either read the schema from the Avro file it receives (provided that the schema is embedded) or it can fetch the schema from Schema Registry. If the connector's input is in JSON format, then it can either infer the schema or fetch it from Schema Registry. The strategy that is used to retrieve the schema is determined by the Schema Access Strategy property.

The connector can authenticate to InfluxDB using username and password.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Sink property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *InfluxDB Sink properties reference*.

Notes and limitations

- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- The value of the Schema Access Strategy property is not independent of the value of the Kafka Message Data Format property. As a result, you must exercise caution when configuring Schema Access Strategy.
  - If the value of Kafka Message Data Format is AVRO, the possible values for Schema Access Strategy are Schema Registry, Embedded Schema or HWX Content-Encoded Schema Reference.
  - If the value of Kafka Message Data Format is JSON, the possible values for Schema Access Strategy are Schema Registry or Infer Schema.
- The Line Protocol Query parameter contains a query expression which creates a string from input record fields. The query depends on the record schema. The result of the query must comply with the line protocol format.

Configuration example

In this example, the connector receives data in JSON format (which is the default setting), transforms the data to line protocol format, and inserts it to InfluxDB. Schema Registry is not used in the example. As a result, this example does not include Schema Registry key and truststore configurations.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "InfluxDB Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "extensions.directory": "/tmp/stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo/",
 "failure.ports": "Retry from PutInfluxDB",
 "topics": "[***TOPIC NAME***]",
 "parameter.InfluxDB Connection URL": "http://[**SERVER
 HOSTNAME**]:[**PORT**]",
 "parameter.InfluxDB Database Name": "[***DATABASE NAME***]",
 "parameter.InfluxDB User Name": "[***USER NAME***]",
 "parameter.InfluxDB User Password": "[***USER PASSWORD***]",
 "parameter.Line Protocol Query": "select 'temp,device_id=' || device_id ||
 ',device_state=' || device_state || ',part_id=' || part_id || ',code=' ||
 code || ',result=' || result_code || ' temperature=' || temperature || ' '
|| ts || '000000' as payload from FLOWFILE",
 "parameter.Schema Access Strategy":"Infer Schema"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

InfluxDB Connection URL

The URL used to connect to InfluxDB.

InfluxDB Database Name

The name of the InfluxDB database.

InfluxDB User Name

The username used to connect to InfluxDB.

InfluxDB User Password

The password used to connect to InfluxDB.

Line Protocol Query

Specifies a query based on the Avro schema of the records and the measurement used.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, the property is set to Infer Schema. This means that the schema is determined (inferred) from the JSON data.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference
InfluxDB Sink properties reference

JDBC Sink connector

The JDBC Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The JDBC Sink connector fetches messages from Kafka and loads them into a database table. The topic this connector receives messages from is determined by the value of the topics property in the configuration. The messages the connector receives from Kafka can be in either Avro or JSON format and must contain records that can be inserted into the database table. The schema of the records can be a predefined schema retrieved from Schema Registry (for both Avro and JSON data), it can be embedded in the Avro data, or inferred from the JSON data. The strategy that is used to retrieve the schema is determined by the Schema Access Strategy property.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Sink property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *JDBC Sink properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- Schema Branch and Schema Version can not be specified at the same time.

Configuration example

In this example, the connector fetches messages in JSON format from Kafka, translates it to a SQL INSERT statement in order to load the data into a PostgreSQL database. The schema of the records must match the schema of the database table. The record schema in this example is inferred from the input JSON.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "JDBC Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "failure.ports": "Retry from PutDatabaseRecord,Failure from PutDatabaseReco
rd",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.JDBC Sink Parameters:Database Connection URL": "[***JDBC
 URL***]",
 "parameter.JDBC Sink Parameters:Database Driver Location": "[***PATH TO
 JDBC DRIVER***]",
 "parameter.JDBC Sink Parameters:Database Driver Class Name": "org.postgr
esql.Driver",
 "parameter.JDBC Sink Parameters:Database Type": "PostgreSQL",
 "parameter.JDBC Sink Parameters:Database User Name": "[***USERNAME***]",
 "parameter.JDBC Sink Parameters:Database User Password":
 "[***PASSWORD***]",
 "parameter.JDBC Sink Parameters:Database Table Name": "[***TABLE NAME***]",
 "parameter.JDBC Sink Parameters:Database Schema Name": "[***SCHEMA
 NAME***]",
 "parameter.JDBC Sink Parameters:Kafka Message Data Format": "JSON",
 "parameter.JDBC Sink Parameters:Schema Access Strategy": "Infer Schema",
 "parameter.JDBC Sink Parameters:Quote Table Name": "true",
 "parameter.JDBC Sink Parameters:Quote Column Names": "true"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

Database Connection URL

The JDBC URL of the PostgreSQL database. For example, jdbc:postgresql://myhost:5432/mydb.

Database Driver Location

The path to the PostgreSQL JDBC driver JAR file (or the directory containing the JAR).

Database Driver Class Name

The Java class name of the PostgreSQL Driver implementation.

Database Type

The type of the database. Because this is a PostgreSQL example, this property is set to PostgreSQL.

Database User Name

The username used for authenticating to the database.

Database User Password

The name of the database table to load data to.

Kafka Message Data Format

The format of the messages the connector receives from Kafka.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, the property is set to Infer Schema. This means that the schema is determined (inferred) from the JSON data.

Quote Table Name and Quote Column Name

These properties specify whether to place double quotes around the table or column names in the SQL statement used by the connector. Some databases, like Oracle and PostgreSQL are case-sensitive for table and column names. By default, Oracle converts all table column names to uppercase if they are not double-quoted upon creation. Similarly, PostgreSQL converts names to lowercase. If tables or columns are double-quoted upon creation and have camel cased names, set these properties to true so that double quotes are also used around the table or column names in the SQL statement used by the connector. If this is not done, the connector will not find the tables on the database.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

JDBC Sink properties reference

Kudu Sink connector

The Kudu Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The Kudu Sink connector fetches messages from Kafka and loads them into a table in Kudu. The topic this connector receives messages from is determined by the value of the topics property in the configuration. The messages the connector receives from Kafka can be in either Avro or JSON format and must contain records that can be inserted into a Kudu table. If the connector input is in Avro format, then it can either read the schema from the Avro file it receives (provided that the schema is embedded) or it can fetch the schema from Schema Registry. If the connector's input is in JSON format, then it can either infer the schema, or fetch it from Schema Registry. The strategy that is used to retrieve the schema is determined by the Schema Access Strategy property.

Kudu is expected to be on a Kerberized cluster. It is not mandatory to have Kudu and Schema Registry on the same cluster. In the connector's configuration the krb5.file property must point to the krb5.conf file that provides access to the cluster that Kudu and Schema Registry are on. The keytab and truststore files that are necessary to access Kudu and Schema Registry must be present on the cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Sink property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *Kudu Sink properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- Schema Registry URL is mandatory even if no registry is used. In a case like this, use the default value, or remove the property completely from the configuration JSON.
- Truststore parameters are mandatory even if connecting to a non-secure Schema Registry or no registry is used. In that case the default (empty) truststore can be used.
- Kerberos parameters for Schema Registry are mandatory even if connecting to a non-secure Schema Registry or no registry is used. In that case the default (empty) keytab can be used.
- Kudu is expected to be on a Kerberized cluster. As a result, Kerberos parameters for Kudu are mandatory.

Configuration example

In this example, the connector receives data in JSON format from a Kafka topic and forwards it to a Kudu table. The record schema is inferred based on the data.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "Kudu Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "input.port": "Input from Kafka",
 "failure.ports": "Failure",
 "krb5.file": "[***PATH TO KRB5 FILE***]",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.Kudu Sink Parameters:Kafka Message Data Format": "JSON",
 "parameter.Kudu Sink Parameters:Kerberos Keytab for Kudu": "[***PATH TO
 KUDU KEYTAB***]",
 "parameter.Kudu Sink Parameters:Kerberos Principal for Kudu": "[***KUDU
 PRINCIPAL***]",
 "parameter.Kudu Sink Parameters:Kudu Masters": "[***KUDU MASTER 1 WITH
 PORT***],[***KUDU MASTER 2 WITH PORT***],[***KUDU MASTER 3 WITH PORT***]",
 "parameter.Kudu Sink Parameters:Schema Access Strategy": "Infer Schema",
 "parameter.Kudu Sink Parameters:Table Name With Schema": "[***KUDU
 SCHEMA***].[***KUDU TABLE***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

krb5.file

The path to a krb5.conf file that contains the information needed to access the Kudu service on a Kerberized cluster

topics

The name of the Kafka topic that the connector fetches messages from.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to JSON. This means that the connector expects data in JSON format.

Kerberos Keytab for Kudu

The path to the keytab file that enables access to the Kudu service on a Kerberized cluster.

Kerberos Principal for Kudu

The principal used to access Kudu

Kudu Masters

A comma-separated list of server URLs that identify the Kudu service's master nodes. For example: localhost:7051,localhost:7151,localhost:7251

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, the property is set to Infer Schema. This means that the record schema is inferred based on the data the connector receives.

Table Name With Schema

The schema and name of the Kudu table in which the records get inserted.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

Kudu Sink properties reference

S3 Sink connector

The S3 Sink Connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The S3 Sink connector fetches messages from Kafka and uploads them to AWS S3. The topic this connector receives messages from is determined by the value of the topics property in the configuration. The messages can contain unstructured (character or binary) data or they can be in Avro or JSON format.

If the input is unstructured data, record processing is disabled. In a case like this, multiple messages can be concatenated into a single output file on S3 using a demarcator (for example, newline for text messages). Merging is optional, large binary data can be forwarded to S3 at a 1:1 ratio, one Kafka message equals a single S3 object.

If the input is either Avro or JSON, record processing is enabled. In a case like this, the schema of the records can be a predefined schema retrieved from Schema Registry (for both Avro and JSON data), it can be embedded in the Avro data, or inferred from the JSON data. The output can be Avro, JSON, CSV or Parquet. Multiple records are typically merged into a single output file before uploading to S3.

The connector can authenticate to AWS using an Access Key.

Properties and configuration

Configuration is passed to the connector in a JSON file during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the *Apache Kafka documentation*.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the *Stateless NiFi Sink property reference*.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the *ADLS Sink properties reference*.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration JSON.
- If a property that has a default value is completely removed from the configuration JSON, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration JSON if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- It is not recommended to configure the following combination of settings:
  - Kafka Message Data Format = JSON
  - Schema Access Strategy = Infer Schema
  - Maximum Number of Entries > 1
  - Output File Data Format = Avro

  In case the input records do not strictly match to one schema (for example, one record has a missing field while the other records do not), schema inference determines a schema for each record, while batching the schemas together assigns only one schema to the entire batch. Because of this, there might be records in the batch that do not match with the schema that is assigned to the batch. This can be avoided by disabling batching with the Maximum Number of Entries set to 1.

Configuration example for fetching unstructured data

In this example, the connector fetches unstructured messages containing single line character data from Kafka. The connector concatenates the messages using newline characters as the demarcator. The files that the connector creates and uploads to ADLS are maximum 10 MB in size. The files are named according to the following pattern: messages_*[***UUID***]*. The connector authenticates to AWS using an Access Key.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "S3 Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
```

```
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConvert
er",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "failure.ports": "PutS3Object Failure",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.S3 Sink Parameters:AWS Access Key ID": "[***ACCESS KEY ID***]",
 "parameter.S3 Sink Parameters:AWS Secret Access Key": "[***SECRET ACCESS
 KEY***]",
 "parameter.S3 Sink Parameters:S3 Region": "[***S3 REGION***]",
 "parameter.S3 Sink Parameters:S3 Bucket": "[***S3 BUCKET***]",
 "parameter.S3 Sink Parameters:Maximum File Size": "10 MB",
 "parameter.S3 Sink Parameters:Kafka Message Data Format": "Raw",
 "parameter.S3 Sink Parameters:Output File Demarcator": "\n",
 "parameter.ADLS Sink Parameters:Output Filename Pattern": "messages_${fi
lename.uuid}"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

AWS Access Key ID

The Access Key ID to use for authentication to AWS.

AWS Secret Access Key

The Secret Access Key to use for authentication to AWS.

S3 Region

The AWS Region of the S3 Bucket to upload data to.

S3 Bucket

The S3 Bucket to upload data to.

Maximum File Size

The maximum size of the output data file. In this example, the maximum size is 10 MB.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to Raw. This means that the connector expects unstructured text or binary data.

Output File Demarcator

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages. In this example, the property is set to \n (newline). This means that the newline character is used to separate the single line character data of the Kafka messages.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to messages_${filename.uuid}.

messages_ is a string literal, ${filename.uuid} is an expression that inserts a generated UUID in the filename. As a result, the files are named according to the following pattern: mess ages_*[***UUID***]*.

Configuration example for fetching JSON data

In this example, the connector fetches messages in JSON format from Kafka. The connector parses the JSON records and converts them to Avro format using the schema inferred from the JSON data. The schema is also

embedded in the output file. The connector merges multiple records (all messages that are available for a single execution of the connector) into an Avro file, generates a name for the file using the following pattern data_*[***TIMESTAMP***]_[***SEQUENCE***]*.avro, and uploads the file to S3. The connector authenticates to AWS using an Access Key.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "S3 Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConvert
er",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "failure.ports": "PutS3Object Failure",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.S3 Sink Parameters:AWS Access Key ID": "[***ACCESS KEY ID***]",
 "parameter.S3 Sink Parameters:AWS Secret Access Key": "[***SECRET ACCESS
 KEY***]",
 "parameter.S3 Sink Parameters:S3 Region": "[***S3 REGION***]",
 "parameter.S3 Sink Parameters:S3 Bucket": "[***S3 BUCKET***]"
 "parameter.ADLS Sink Parameters:Kafka Message Data Format": "JSON",
 "parameter.ADLS Sink Parameters:Schema Access Strategy": "Infer Schema",
 "parameter.ADLS Sink Parameters:Schema Write Strategy": "Embed Avro Schema
",
 "parameter.ADLS Sink Parameters:Output File Data Format": "Avro",
 "parameter.ADLS Sink Parameters:Output Filename Pattern": "data_${filenam
e.timestamp}_${filename.sequence}.avro",
 "parameter.ADLS Sink Parameters:Output Filename Timestamp Format": "yyyyMM
dd_HHmmss_SSS"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

AWS Access Key ID

The Access Key ID to use for authentication to AWS.

AWS Secret Access Key

The Secret Access Key to use for authentication to AWS.

S3 Region

The AWS Region of the S3 Bucket to upload data to.

S3 Bucket

The S3 Bucket to upload data to.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to JSON. This means that the connector expects JSON data.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, this property is set to Infer Schema, meaning that the schema is determined (inferred) from the JSON data.

Schema Write Strategy

Specifies whether the record schema is written to the output data file. In this example, this property is set to Embed Avro Schema, meaning that the schema is embedded in the output Avro file.

Output File Data Format

Specifies the format of the records written to the output file. In this example, this property is set to Avro, meaning that the output file format is Avro.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to data_${filename.timestamp}_${filename.sequence}.avro.

data_, the underscore (_), and .avro are string literals. ${filename.timestamp} and ${filename.sequence} are expressions. ${filename.timestamp} inserts the current timestamp in the filename. ${filename.sequence} inserts an incrementing sequence value in the filename. As a result, the files are named according to the following pattern: data*_[***TIMESTAMP***]_[***SEQUENCE***]*.avro.

Output Filename Timestamp Format

Specifies the timestamp format used in the ${filename.timestamp} expression. The expression is used when generating the output filename.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

S3 Sink properties reference

Cloudera Runtime 7.3.1

Kafka Connect Connector Reference

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

HTTP Source properties reference....5
JDBC Source properties reference....8
JMS Source properties reference....14
MQTT Source properties reference....17
SFTP Source properties reference....21
Stateless NiFi Source properties reference....32
Syslog TCP Source properties reference....36
Syslog UDP Source properties reference....44
ADLS Sink properties reference....49
Amazon S3 Sink properties reference....57
HDFS Sink properties reference....58
HDFS Stateless Sink properties reference....60
HTTP Sink properties reference....68
InfluxDB Sink properties reference....75
JDBC Sink properties reference....80
Kudu Sink properties reference....86

S3 Sink properties reference..... 92

Stateless NiFi Sink properties reference..... 99

HTTP Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the HTTP Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source properties reference.

Authorized Issuer DN Pattern

Description

A regular expression to apply against the Issuer's Distinguished Name of incoming connections. You can use this property when the Client Authentication property is set to REQUIRED (mutual TLS).

Default Value

.*

Accepted Values

Required

false

Authorized Subject DN Patter

Description

A regular expression to apply against the Subject's Distinguished Name of incoming connections. You can use this property when the Client Authentication property is set to REQUIRED (mutual TLS).

Default Value

.*

Accepted Values

Required

false

Base Path

Description

The base path or context path of the URL.

Default Value

contentListener

Accepted Values

Required

true

Client Authentication

Description

The client authentication policy used for HTTPS.

Default Value

NONE

Accepted Values

NONE, REQUIRED

Required

true

Keystore Filename

Description

The fully-qualified filename of a keystore.

This keystore is used to establish a secure connection between this connector and its clients using HTTPS.

Default Value

Accepted Values

Required

true

Keystore Key Password

Description

The password used to access the key stored in the keystore file configured in the Keystore Filename property.

Default Value

Accepted Values

Required

true

Keystore Password

Description

The password used to access the contents of the keystore configured in the Keystore Filename property.

Default Value

Accepted Values

Required

true

Keystore Type

Description

The type of the keystore configured in the Keystore Filename property.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Listening Port

Description

The port to listen on for communication.

Default Value

Accepted Values

Required

true

Truststore Filename

Description

The fully-qualified filename of a truststore.

When using one-way SSL (Client Authentication is set to NONE), this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

When using one-way SSL (Client Authentication is set to NONE), this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

When using one-way SSL (Client Authentication is set to NONE), this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

false

JDBC Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the JDBC Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source properties reference.

Additional WHERE clause

Description

Specifies a custom condition to add in the WHERE clause of the SQL query.

Default Value

Accepted Values

Required

false

Columns to Return

Description

A comma-separated list of column names to be returned by the query. All columns are returned if it is not specified.

Default Value

Accepted Values

Required

false

Custom Query

Description

Specifies a custom SQL query to use instead of generating the query from the table and column names.

Default Value

Accepted Values

Required

false

Database Connection URL

Description

The database specific connection URL used for connecting to the database. For example, jdbc:postgresql://localhost:5432/postgres.

Default Value

Accepted Values

Required

true

Database Driver Class Name

Description

The database driver class name For example, org.postgresql.Driver.

Default Value

Accepted Values

Required

true

Database Driver Location

Description

A comma-separated list of files or folders containing the JDBC client libraries.

Default Value

Accepted Values

Required

true

Database Table Name

Description

The name of the database table to query.

Default Value

Accepted Values

Required

true

Database Type

Description

The database type used for generating database specific SQL queries.

Default Value

Generic

Accepted Values

Generic, Oracle, Oracle 12+, MS SQL 2008, MS SQL 2012+, MySQL, PostgreSQL

Required

true

Database User Name

Description

The database user name. If username/password authentication is not required by the database server, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Database User Password

Description

The database user password. If username/password authentication is not required by the database server, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Date Format

Description

Specifies the format to use when writing Date fields to JSON.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Initial Load Strategy

Description

Specifies how existing rows in the database table are handled when the connector is started for the first time.

Default Value

Start at Beginning

Accepted Values

Start at Beginning, Start at Current Maximum Values

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector sends to Kafka. The database row is converted to this format. If the output format is Avro and Schema Access Strategy is set to Inherit Schema, the schema is embedded in the output message. If the output format is Avro and Schema Access Strategy is set to Schema Registry, the schema is not embedded in the output message.

Default Value

Avro

Accepted Values

Avro, JSON

Required

true

Kafka Message Key Column

Description

Specifies a database table column. The value of the column specified is used as the key of the Kafka message.

Default Value

Accepted Values

Required

false

Kerberos Keytab

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Maximum-value Columns

Description

A comma-separated list of column names used for incremental loading.

Default Value

Accepted Values

Required

false

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the database record.

- If set to Inherit Schema, the schema is determined from the database schema.

- If set to Schema Registry, the schema is read from Schema Registry.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

Default Value

Inherit Schema

Accepted Values

Inherit Schema, Schema Registry, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry. If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when writing Time fields to JSON.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when writing Timestamp fields to JSON.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using HTTPS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

tue

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

JMS Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the JMS Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source properties reference.

JMS Broker URI

Description

The URI to use to connect to the JMS Message Broker. For example, ssl://myhost:61617

Default Value

Accepted Values

Required

true

JMS Client Libraries

Description

The path to the directory containing the JMS client libraries.

Default Value

Accepted Values

Required

true

JMS Connection Factory Class Name

Description

The fully qualified name of the JMS ConnectionFactory implementation class. For example, org.apache.activemq.ActiveMQSslConnectionFactory.

Default Value

Accepted Values

Required

true

JMS Destination Name

Description

The name of the JMS Destination.

Default Value

Accepted Values

Required

true

JMS Destination Type

Description

The type of the JMS Destination.

Default Value

QUEUE

Accepted Values

QUEUE, TOPIC

Required

true

JMS User Name

Description

The username used for authentication.

If username/password authentication is not required by the JMS Message Broker, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

JMS User Password

Description

The password used for authentication.

If username/password authentication is not required by the JMS Message Broker, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Keystore Filename

Description

The fully-qualified filename of a keystore. This keystore is used for mutual TLS towards the JMS Message Broker.

If the JMS Message Broker does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Keystore Key Password

Description

The password used to access the key stored in the keystore file configured in the Keystore Filename property.

If the JMS Message Broker broker does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Keystore Password

Description

The password used to access the contents keystore configured in the Keystore Filename property.

If the JMS Message Broker does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Keystore Type

Description

The type of the keystore configured in the Keystore Filename property.

If the JMS Message Broker does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

false

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with the JMS server using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

MQTT Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the MQTT Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the

accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source properties reference.

Keystore Filename

Description

The fully-qualified filename of a keystore. This keystore is used for mutual TLS towards the MQTT server.

If the MQTT broker does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Keystore Key Password

Description

The password used to access the key stored in the keystore file configured in the Keystore Filename property.

If the MQTT broker does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Keystore Password

Description

The password used to access the contents keystore configured in the Keystore Filename property.

If the MQTT broker does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Keystore Type

Description

The type of the keystore configured in the Keystore Filename property.

If the MQTT broker does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

false

MQTT Broker URI

Description

The URI to use to connect to the MQTT broker.

Example URI if SSL is not used (keystore-related parameters are removed and and the default truststore is used): tcp://localhost:1883

Example URI if SSL is used: ssl://localhost:8883

Default Value

tcp://localhost:1883

Accepted Values

Required

true

MQTT Password

Description

Password to use when connecting to the MQTT broker.

If username-password authentication is not required by the MQTT broker, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

MQTT Quality of Service

Description

The Quality of Service (QoS) to receive the message with.

0 - at most once

1 - at least once

2 - exactly once

Default Value

0

Accepted Values

0,1 or 2

Required

true

MQTT Topics

Description

Specifies the MQTT topic to subscribe to. Use an MQTT wildcard to subscribe to multiple topics simultaneously.

Default Value

Accepted Values

Required

true

MQTT Username

Description

Username to use when connecting to the MQTT broker.

If username-password authentication is not required by the MQTT broker, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with the MQTT server using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

MQTT Client ID

Description

The MQTT client ID to use.

If it is not set, a UUID is generated. In such a case, this property must be removed from the configuration.

Default Value

Accepted Values

Required

false

MQTT Clean Session

Description

Specifies whether to start a new session.

If it is set to true, the client and server discard any previous sessions and start a new one. The new session lasts as long as the network connection. State data associated with a session is not reused in any subsequent sessions.

If it is set to false, the server resumes communications with the client based on the state from the current session (identified by the Client ID). The client and server store the session after being disconnected. When a session that is not a clean session is disconnected, the server stores further QoS 1 and QoS 2 messages that match any subscriptions that the client has at the time of disconnection. These messages are stored as part of the session state.

Default Value

true

Accepted Values

true, false

Required

false

SFTP Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the SFTP Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source properties reference.

Avro Schema Write Strategy

Description

Specifies whether the record schema is written to the output Avro file.

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

CSV Character Set

Description

The character set used to read the input CSV files.

This property is ignored if the input is not a CSV file or if record processing is not enabled.

Default Value

UTF-8

Accepted Values

Required

true

CSV Escape Character

Description

The escape character used in the input CSV files to escape other special characters.

This property is ignored if the input is not a CSV file or if record processing is not enabled.

Default Value

\

Accepted Values

Required

true

CSV Quote Character

Description

The quote character used in the input CSV files.

This property is ignored if the input is not a CSV file or if record processing is not enabled.

Default Value

"

Accepted Values

Required

true

CSV Record Separator

Description

The record separator used in the input CSV files.

This property is ignored if the input is not a CSV file or if record processing is not enabled.

Default Value

\n

Accepted Values

Required

true

CSV Treat First Line as Header

Description

Specifies whether the first line in the input file is handled as a header.

Ignored if the input is not a CSV file or record processing is not enabled.

Default Value

false

Accepted Values

true, false

Required

true

CSV Trim Fields

Description

Specifies whether whitespace characters are removed from the beginning and the end of fields.

Ignored if the input is not a CSV file or record processing is not enabled.

Default Value

true

Accepted Values

true, false

Required

true

CSV Value Separator

Description

The value separator used in the input CSV files.

Ignored if the input is not a CSV file or record processing is not enabled.

Default Value

,

Accepted Values

Required

true

Completion Strategy

Description

Specifies what to do with the original file on the server once it has been fetched.

If the Completion Strategy fails, a warning is logged but the data is still transferred.

Default Value

None

Accepted Values

None, Move File, Delete File

Required

true

Date Format

Description

Specifies the format used for parsing date fields in the input data.

This property is only used if Input Data Format is set to CSV or JSON.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Enable Record Processing

Description

Enables or disables record processing.

If set to true, the Input Data Format is considered and the file gets parsed into records. In this case the Records Per Kafka Message property defines how many records are written into one Kafka message.

If set to false, the entire file gets forwarded to Kafka as one message.

Default Value

true

Accepted Values

true, false

Required

true

File Filter Regex

Description

The Java regular expression to use for filtering filenames. Only files whose names match the regular expression are fetched.

Default Value

.*

Accepted Values

Required

true

Follow Symlink

Description

If set to true, both symbolic files and nested symbolic subdirectories are pulled. Otherwise, symbolic files are not read and symbolic link subdirectories are not traversed.

Default Value

false

Accepted Values

true, false

Required

true

Grok Expression

Description

Specifies the format of a line in Grok format. This allows the connector to understand how to parse each line in the input file. If a line in the file does not match this pattern, the line is handled according to what is set in the Grok No Match Behavior property.

A valid Grok expression must be specified using this property even if Grok format is not used.

Default Value

%{GREEDYDATA:message}

Accepted Values

Required

true

Grok No Match Behavior

Description

Specifies how to handle lines that do not match the pattern set in the Grok Expression property.

If set to append-to-previous-message, non-matching lines are appended to the last field of the previous message.

If set to skip-line, non-matching lines are skipped.

If set to raw-line, non-matching lines are only added to the _raw field.

Default Value

append-to-previous-message

Accepted Values

append-to-previous-message, skip-line, raw-line

Required

true

Host Key File

Description

The fully-qualified filename of the host key file.

If supplied, this file is used as the host key.

If a host key is not supplied, but Strict Host Key Checking is set to true, the known_hosts and known_hosts2 files from the ~/.ssh directory are used.

If a host key is not supplied and Strict Host Key Checking is set to false, no host key file is used.

This parameter must either contain the fully-qualified name of a file, or be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Hostname

Description

The hostname or IP address of the remote system.

Default Value

localhost

Accepted Values

Required

true

Ignored Dotted Files

Description

Specifies whether to ignore files whose names begin with a dot (".").

Default Value

true

Accepted Values

true, false

Required

true

Input Data Format

Description

The format in which the input file contains record-oriented data.

If Enable Record Processing is set to false, this setting is ignored.

Default Value

JSON

Accepted Values

JSON, CSV, GROK

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Move Destination Directory

Description

The fully-qualified name of the directory on the remote server to move the original file to once it is ingested. This property is ignored unless the Completion Strategy property is set to Move File. The specified directory must already exist on the remote system.

This parameter must either contain the fully-qualified name of a directory, or be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Password

Description

The password to use when connecting to the SFTP server.

If the server does not require a password, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Path Filter Regex

Description

The Java Regular Expression to use for filtering paths.

If Search Recursively is set to true, only subdirectories whose path matches the given regular expression are scanned.

If Search Recursively is set to false, this property is ignored.

Default Value

.*

Accepted Values

Required

true

Port

Description

The port that the remote system is listening on for file transfers.

Default Value

22

Accepted Values

Required

true

Private Key File

Description

The fully-qualified filename of a private key file.

If no private key is used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Private Key Password

Description

The password used to access the private key.

If no private key is used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Record Per Kafka Message

Description

Specifies how many records to write into each Kafka message.

If Enable Record Processing is set to false, this setting is ignored.

Default Value

1

Accepted Values

Required

true

Remote Path

Description

The path on the remote system from which to pull files.

Default Value

.

Accepted Values

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the input records if the Enable Record Pr ocessing property is set to true .

The value you set here depends on the input data format.

If set to Schema Registry, the schema is read from Schema Registry.

This setting works with all input data formats.

If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if your input data format is either JSON or CSV.

If set to Field Names From Grok Expression, the schema is determined using the field names in the Grok Expression property. This setting can only be used if your input data format is GROK.

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Field Names From Grok Expression

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry.

If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name.

If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server.

If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

true

Search Recursively

Description

Specifies whether to pull files from arbitrarily nested subdirectories. Subdirectories are not traversed if set to false.

Default Value

false

Accepted Values

true, false

Required

true

Strict Host Key Checking

Description

Specifies whether strict enforcement of host keys is applied.

Default Value

false

Accepted Values

true, false

Required

true

Time Format

Description

Specifies the format used for parsing time fields in the input data. This property is only used if Input Data Format is set to CSV or JSON.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format used for parsing timestamp fields in the input data. This property is only used if Input Data Format is set to CSV or JSON.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Username

Description

Username for connecting to the SFTP server.

Default Value

Accepted Values

Required

true

Stateless NiFi Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Source connector.

In addition to the properties listed here, this connector also accepts the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

header.attribute.regex

Description

A Java regular expression that is evaluated against all flowfile attribute names. Any attribute name matching the regular expression is converted into a Kafka message header. The name of the attribute is used as the header key, the value of the attribute is used as the header value. If not specified, headers are not added to the Kafka record.

Default Value

Accepted Values

Required

false

header.name.regex

Description

A Java regular expression that will be evaluated against all flowfile attribute names. For any attribute whose name matches the regular expression, the Kafka record will have a header whose name matches the attribute name and whose value matches the attribute value. If not specified, the Kafka record will have no headers added to it.

Default Value

Accepted Values

Required

false

key.attribute

Description

Specifies the name of a flowfile attribute that should be used to specify the key of the Kafka record. If not specified, the Kafka record will not have a key associated with it. If specified, but the attribute does not exist on a particular flowfile, it will also have no key associated with it.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

True

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

output.port

Description

The name of the output port in the NiFi dataflow to pull data from. If the dataflow contains exactly one port, this property is optional and can be omitted. However, if the dataflow contains multiple ports (for example, a success and a failure port), this property must be specified. If any flowfile is sent to any port other than the specified port, it is considered as a failure. The session is rolled back and no data is collected.

Default Value

Accepted Values

Required

false

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named

Directory. Parameters can also be applied to specific Parameter Contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the parameter context named My Context.

Default Value

Accepted Values

Required

false

topic.name.attribute

Description

Specifies the name of a flowfile attribute to use for determining which Kafka topic a flowfile is sent to. Either the topics or topic.name.attribute property must be specified. If both are specified, topi c.name.attribute takes precedence. However, if a flowfile does not have the specified attribute name, then the connector falls back to using the topics property.

Default Value

Accepted Values

Required

false

topics

Description

The name of the topic to deliver data to. All flowfiles are delivered to the topic specified here. However, it is also possible to determine the topic individually for each flowfile. To do this, ensure that the dataflow specifies the topic name in an attribute, and then use topic.name.attribute to specify the name of the attribute instead of topic name. For example, if you wanted a separate Kafka topic for each data source, you can omit the topics property and instead specify the attribute (for example, datasource.hostname) corresponding to the topic using the topic.name.attribute property.

Default Value

Accepted Values

Required

true

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to / tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

Syslog TCP Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Syslog TCP Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source properties reference.

Authorized Issuer DN Pattern

Description

A regular expression that can be applied against the Issuer's Distinguished Name of incoming TLS connections.

Default Value

.*

Accepted Values

Required

false

Authorized Subject DN Pattern

Description

A regular expression that can be applied against the Subject's Distinguished Name of incoming TLS connections.

Default Value

.*

Accepted Values

Required

false

Avro Schema Write Strategy

Description

Specifies how the schema is attached to the outgoing Avro messages. This property only takes effect if the Output Format is AVRO.

- If set to Embed Avro Schema then the schema is embedded in every output Avro message.
- If set to Do Not Write Schema then no schema information is attached to the output Avro messages.
- If set to HWX Content-Encoded Schema Reference then a reference to the schema (identified by Schema Name) within Schema Registry is encoded in the content of the outgoing Avro messages.

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

Character Set

Description

The character set used in the input as well as the output data.

Default Value

UTF-8

Accepted Values

Required

true

Client Authentication

Description

The client authentication policy used for SSL.

Default Value

REQUIRED

Accepted Values

NONE, WANT, REQUIRED

Required

true

Date Format

Description

Specifies the format used for writing date fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Grok Expression

Description

Specifies the format of a line in Grok format. This allows the connector to understand how to parse each line in the input file. If a line in the file does not match this pattern, the line is handled according to the Grok No Match Behavior. A valid Grok expression must be specified using this property even if Grok format is not used.

Default Value

%{GREEDYDATA:message}

Accepted Values

Required

true

Grok No Match Behaviour

Description

Specifies how to handle lines that do not match the pattern set in the Grok Expression property.

- If set to append-to-previous-message, non-matching lines are appended to the last field of the previous message.
- If set to skip-line, non-matching lines are skipped. If set to raw-line, non-matching lines are only added to the _raw field.

Default Value

append-to-previous-message

Accepted Values

append-to-previous-message, skip-line, raw-line

Required

true

Input Data Format

Description

The format of incoming messages.

Default Value

Syslog 3164

Accepted Values

Syslog 3164, Syslog 5424, Grok

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Max Batch Size

Description

The maximum number of messages to add to a single batch of messages. If multiple messages are available, they are concatenated with a new line character up to this configured maximum number of messages.

Default Value

1

Accepted Values

Required

true

Max Number of Worker Threads

Description

he maximum number of worker threads available for handling TCP connections.

Default Value

2

Accepted Values

Required

true

Output Format

Description

The format of the messages written to Kafka.

Default Value

AVRO

Accepted Values

TEXT, AVRO, JSON

Required

true

Output Grouping for JSON

Description

Specifies how JSON objects are grouped in the connector output.

- If set to output-array, the output will consist of an array of JSON objects.
- If set to output-oneline, each line of the output data will be one JSON object. That is, each JSON object occupies one line in the output.

Default Value

output-oneline

Accepted Values

output-array, output-oneline

Required

true

Port

Description

The port to listen on for communication.

Type

int

Default Value

Accepted Values

Required

true

SSL Keystore Filename

Description

The fully-qualified filename of a keystore. This keystore is used to establish a secure connection between this connector and its clients using SSL.

Default Value

Accepted Values

Required

true

SSL Keystore Key Password

Description

The password used to access the key stored in the keystore file configured in SSL Keystore Filename.

Default Value

Accepted Values

Required

true

SSL Keystore Password

Description

The password used to access the contents of the keystore configured in the SSL Keystore Filename property.

Default Value

Accepted Values

Required

true

SSL Keystore Type

Description

The type of the keystore configured in the SSL Keystore Filename property.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

true

SSL Truststore Filename

Description

The fully-qualified filename of a truststore. It can be used for establishing connections using mutual TLS. When using one-way SSL (Client Authentication parameter set to NONE), this parameter must be removed completely from the configuration JSON.

Default Value

Accepted Values

Required

false

SSL Truststore Password

Description

The password used to access the contents of the truststore configured in the SSL Truststore Filen ame property. When using one-way SSL (Client Authentication parameter set to NONE), this parameter must be removed completely from the configuration JSON.

Default Value

Accepted Values

Required

false

SSL Truststore Type

Description

The type of the truststore configured in the SSL Truststore Filename property. When using one-way SSL (Client Authentication parameter set to NONE), this parameter must be removed completely from the configuration JSON.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

false

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the input message. This property only takes effect if the input data format is GROK.

- If set to Schema Registry then the schema is read from Schema Registry.
- If set to Field Names from Grok Expression then the schema is determined using the field names in the Grok Expression property.

Default Value

Schema Registry

Accepted Values

Schema Registry, Field Names From Grok Expression

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry.

- If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name.
- If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format used for writing time fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format used for writing timestamp fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Syslog UDP Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Syslog UDP Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source properties reference.

Avro Schema Write Strategy

Description

Specifies how the schema is attached to the outgoing Avro messages. This property only takes effect if the Output Format is AVRO.

- If set to Embed Avro Schema then the schema is embedded in every output Avro message.
- If set to Do Not Write Schema then no schema information is attached to the output Avro messages.
- If set to HWX Content-Encoded Schema Reference then a reference to the schema (identified by Schema Name) within Schema Registry is encoded in the content of the outgoing Avro messages.

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

Character Set

Description

The character set used in the input as well as the output data.

Default Value

UTF-8

Accepted Values

Required

true

Date Format

Description

Specifies the format used for writing date fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Grok Expression

Description

Specifies the format of a line in Grok format. This allows the connector to understand how to parse each line in the input file. If a line in the file does not match this pattern, the line is handled according to what is set in the Grok No Match Behavior property. A valid Grok expression must be specified using this property even if Grok format is not used.

Default Value

%{GREEDYDATA:message}

Accepted Values

Required

true

Grok No Match Behaviour

Description

Specifies how to handle lines that do not match the pattern set in the Grok Expression property.

- If set to append-to-previous-message, non-matching lines are appended to the last field of the previous message.
- If set to skip-line, non-matching lines are skipped. If set to raw-line, non-matching lines are only added to the _raw field.

Default Value

append-to-previous-message

Accepted Values

append-to-previous-message, skip-line, raw-line

Required

true

Input Data Format

Description

The format of incoming messages.

Default Value

Syslog 3164

Accepted Values

Syslog 3164, Syslog 5424, Grok

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Output Format

Description

The format of the messages written to Kafka.

Default Value

AVRO

Accepted Values

TEXT, AVRO, JSON

Required

true

Output Grouping for JSON

Description

Specifies how JSON objects are grouped in the connector output.

- If set to output-array, the output will consist of an array of JSON objects.
- If set to output-oneline, each line of the output data becomes one JSON object. That is, each JSON object occupies one line in the output.

Default Value

output-oneline

Accepted Values

output-array, output-oneline

Required

true

Port

Description

The port to listen on for communication.

Default Value

Accepted Values

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the input message. This property only takes effect if the input data format is GROK.

- If set to Schema Registry then the schema is read from Schema Registry.
- If set to Field Names from Grok Expression then the schema is determined using the field names in the Grok Expression property.

Default Value

Schema Registry

Accepted Values

Schema Registry, Field Names From Grok Expression

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry.

- If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name.
- If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format used for writing time fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format used for writing timestamp fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

ADLS Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the ADLS Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

ADLS Account Key

Description

The Storage Account Key to use for authentication to Azure. Required when Account Key authentication is used.

Default Value

Accepted Values

Required

false

ADLS Account Name

Description

The name of the Azure Storage Account to use for authentication to Azure and the target account where the output file is uploaded.

Default Value

Accepted Values

Required

true

ADLS Directory Name

Description

The name of the directory where the output file is uploaded. Use an empty string to specify the root directory.

Default Value

Accepted Values

Required

true

ADLS Endpoint Suffix

Description

Specifies the endpoint suffix used for accessing the ADLS service.

Default Value

dfs.core.windows.net

Accepted Values

Required

true

ADLS Filesystem Name

Description

The name of the filesystem (or container) in the Azure Storage Account where the output file is uploaded.

Default Value

Accepted Values

Required

true

ADLS SAS Token

Description

The SAS Token to use for authentication to Azure. Required when SAS Token authentication is used.

Default Value

Accepted Values

Required

false

Avro Schema Write Strategy

Description

Specifies how the record schema is attached to the output data file. Applicable only for Avro output (Output File Data Format is set to Avro).

Do Not Write Schema

Neither the schema nor reference to the schema is attached to the output Avro messages.

Embed Avro Schema

The schema is embedded in every output Avro message.

HWX Content-Encoded Schema Reference

A reference to the schema (identified by Schema Name) within Schema Registry is encoded in the content of the outgoing Avro messages.

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

Azure Service Principal Client ID

Description

The Client ID of the Service Principal to use for authentication to Azure. Required when Service Principal authentication is used.

Default Value

Accepted Values

Required

false

Azure Service Principal Client Secret

Description

The Client Secret of the Service Principal to use for authentication to Azure. Required when Service Principal authentication is used.

Default Value

Accepted Values

Required

false

Azure Service Principal Tenant ID

Description

The Tenant ID of the Service Principal to use for authentication to Azure. Required when Service Principal authentication is used.

Default Value

Accepted Values

Required

false

Date Format

Description

Specifies the format to use when writing date fields to JSON or CSV.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka. If set to Avro or JSON, record processing is enabled. Raw can be used for unstructured text or binary data.

Default Value

Avro

Accepted Values

Avro, JSON, Raw

Required

true

Kerberos Keytab

Description

The fully-qualified filename of the Kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Maximum File Size

Description

The maximum size of the output data file. No size limit is applied if this property is not specified. Example values: 100 MB, 1 GB.

Default Value

Accepted Values

Required

false

Maximum Number of Entries

Description

The maximum number of entries in the output data file. In the context of this property, entry can mean one of two things. If record processing is enabled (Output File Data Format is set to Avro or JSON), an entry is a record. Otherwise, entry means a Kafka message. Set this property to 1 if you want to create a separate output file for each Kafka message.

Default Value

1000000

Accepted Values

Required

true

Output File Data Format

Description

Specifies the format of the records written to the output file. Required when record processing is enabled (Kafka Message Data Format is set to Avro or JSON).

Default Value

Avro

Accepted Values

Avro, JSON, CSV

Required

false

Output File Demarcator

Description

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages (no record processing). This property can only be used if Kafka Message Data Format is set to Raw. If you want to use newline as the delimiter, set this property to \n.

Default Value

Accepted Values

Required

false

Output Filename Pattern

Description

Specifies the structure of the name of the output file. The pattern can contain string literal (fixed text) parts and one or more of the following expressions:

- ${filename.uuid}: Inserts a generated UUID in the filename.

- ${filename.timestamp}: Inserts the current timestamp in the filename.
- ${filename.sequence}: Inserts an incrementing sequence value in the filename.

In order to generate unique filenames, either ${filename.uuid} or ${filename.sequence} must be used in the pattern.

Examples:

- data_${filename.uuid}.json
- records_${filename.timestamp}_${filename.sequence}.avro

Default Value

${filename.uuid}

Accepted Values

Required

false

Output Filename Sequence Initial Value

Description

This property is used to configure the initial value of the ${filename.sequence} expression. The value you set in this property is not the initial value of the sequence. The sequence starts at the value of this property +1. For example, if you set this property to 0, the sequence starts at 1.

Default Value

0

Accepted Values

Required

false

Output Filename Sequence Padding Length

Description

Specifies the length of the ${filename.sequence} expression in characters. If the sequence has fewer characters than the value set in this property, it is padded with zeros (0). Padding is added to the left of the sequence.

Default Value

6

Accepted Values

Required

false

Output Filename Timestamp Format

Description

The timestamp format to use for the ${filename.timestamp} expression. For example, yyyyMMdd _HHmmss_SSS.

Default Value

Accepted Values

Required

false

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

This property is not used if record processing is disabled (Kafka Message Data Format is set to Raw).

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry. If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when writing Time fields to JSON or CSV.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when writing Timestamp fields to JSON or CSV.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using HTTPS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Use Azure Managed Identity

Description

Turns on authentication using Managed Identity.

Default Value

false

Accepted Values

true, false

Required

true

Amazon S3 Sink properties reference

Amazon S3 Sink connector properties reference.

The following table collects connector properties that are specific for the Amazon S3 Sink Connector. For properties common to all sink connectors, see the upstream Apache Kafka documentation.

Property Name	Description	Type	Default Value	Accepted Values	Recommended Value
aws.s3.bucket	The target S3 bucket name.	String	none	Any valid S3 bucket name.	
aws.s3.service_endpoint	The target S3 host and port.	String	none	Any valid S3 endpoint.	

Property Name	Description	Type	Default Value	Accepted Values	Recommended Value
aws.access_key_id	The AWS secret key ID to authenticate.	String	none	Any valid secret key issued by AWS.	
aws.secret_access_key	The AWS secret access key to authenticate.	String	none	Any valid access key issued by AWS.	
value.converter	Value conversion class.	String	none		com.cloudera.dim.kafka.connect.converts
value.converter.passthrough.enabled	Configures whether the AvroConverter translates an Avro record into Kafka Connect Data or transparently passes the Avro encoded bytes as payload.	Boolean	true	true, false	True if input and output are both Avro.
value.converter.schema.registry.url	The URL to the Schema Registry server.	String	none		
output.storage	The S3 storage implementation class.	String	none		com.cloudera.dim.kafka.connect.s3.S3Pa
output.writer	The output file writer which determines the type of file to be written. The value of this property should be the FQCN of a class that implements the PartitionWriter interface.	String	none	• com.cloudera.dim.kafka.connect.partition.writers.avro.AvroP • com.cloudera.dim.kafka.connect.partition.writers.json.JsonPa • com.cloudera.dim.kafka.connect.hdfs.parquet.ParquetPartitio • com.cloudera.dim.kafka.connect.partition.writers.txt.TxtParti	com.cloudera.dim.kafka.connect.partition
output.avro.passthrough.enabled	Configures Whether the output writer expects an Avro encoded Kafka Connect data record. Must match the configuration ofvalue.converter.passthrough.enabled.	Boolean	none	true, false	True if input and output are both Avro.

HDFS Sink properties reference

HDFS Sink connector properties reference.

The following table collects connector properties that are specific for the HDFS Sink Connector. For properties common to all sink connectors, see the upstream Apache Kafka documentation.

Property Name	Description	Type	Default Value	Accepted Values	Recommended Values
hdfs.uri	The file system URI to connect to on the destination cluster. This property supports any valid Hadoop-compatible filesystem (HCFS, For example, HDFS or ofs) URI.	String	None		
hdfs.output	The root directory on the HDFS cluster where all the output files will reside. The sub path has the following pattern: {topic}/{topic}_{partition}_{endoffset}.{file extension}	String	/tmp		Any path on the HDFS file system where the role has read write permission.
hdfs.kerberos.authentication	Enables or disables secure access to the HDFS cluster by authenticating with Kerberos.	Boolean	false	true or false	
hdfs.kerberos.user.principal	The kerberos user principal.	String	null	The host-dependent Kerberos principal assigned to the Kafka Connect role.	
hdfs.kerberos.keytab.path	The path to the Kerberos keytab file.	String	null		In a Cloudera Manager provisioned environment, it's recommended to use the Cloudera Manager Config Provider to automatically provision the path.
hdfs.kerberos.namenode.principal	The kerberos name node principal. Required when the HDFS cluster has data encryption on.	String	null		
hadoop.conf.path	The path to the site specific Hadoop configuration XML files. Required when the HDFS cluster has data encryption on.	String	null		
output.writer	The output file writer which determines the type of file to be written to the HDFS cluster. The value of this property should be the FQCN of a class that implements the PartitionWriter interface.	String	null	• com.cloudera.dim.kafka.connect.partition.writers.avro.AvroP • com.cloudera.dim.kafka.connect.partition.writers.json.JsonPa • com.cloudera.dim.kafka.connect.hdfs.parquet.ParquetPartitio • com.cloudera.dim.kafka.connect.partition.writers.txt.TxtParti	com.cloudera.dim.kafka.connect.partition

Property Name	Description	Type	Default Value	Accepted Values	Recommended Values
output.avro.passthrough.enabled	Configures whether the output writer expects an Avro encoded Kafka Connect data record. Must match the configuration of value.converter.passthrough.enabled.	Boolean	true	true or false	True if input and output are both Avro.
value.converter	The converter to be used to translate the value field of the source Kafka record into Kafka Connect Data format.	String	Inherited from Kafka Connect worker properties.	• org.apache.kafka.connect.json.JsonConverter • org.apache.kafka.connect.storage.StringConverter • com.cloudera.dim.kafka.connect.converts.AvroConverter	com.cloudera.dim.kafka.connect.converts
value.converter.schema.registry.url	The URL to the Schema Registry server.	String	null	true or false	
value.converter.passthrough.enabled	Configures whether the AvroConverter translates an Avro record into Kafka Connect Data or transparently passes the Avro encoded bytes as payload.	Boolean	true	true or false	True if input and output are both Avro.

HDFS Stateless Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the HDFS Stateless Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Avro Schema Write Strategy

Description

Specifies how the record schema is attached to the output data file. Applicable only for Avro output (Output File Data Format is set to Avro).

Do Not Write Schema

Neither the schema nor reference to the schema is attached to the output Avro messages.

Embed Avro Schema

The schema is embedded in every output Avro message.

HWX Content-Encoded Schema Reference

A reference to the schema (identified by Schema Name) within Schema Registry is encoded in the content of the outgoing Avro messages.

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

Compression Codec

Description

The codec used for file compression in HDFS. Use this property to set the codec if the output file format is JSON, Avro, or CSV.

Default Value

NONE

Accepted Values

NONE, DEFAULT, BZIP, GZIP, LZ4, LZO, SNAPPY, AUTOMATIC

Required

true

Compression Codec for Parquet

Description

Codec used for file compression in HDFS. Use this property to set the codec if the output file format is Parquet.

Default Value

UNCOMPRESSED

Accepted Values

UNCOMPRESSED,SNAPPY, GZIP, LZO

Required

true

Date Format

Description

Specifies the format to use when writing date fields to JSON or CSV.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Hadoop Configuration Resources

Description

A comma separated list of files which contains the Hadoop file system configuration.

Default Value

/etc/hadoop/conf/core-site.xml, /etc/hadoop/conf/hdfs-site.xml

Accepted Values

Required

false

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka. If set to Avro or JSON, record processing is enabled. Raw can be used for unstructured text or binary data.

Default Value

Avro

Accepted Values

Avro, JSON, Raw

Required

true

Kerberos Keytab for HDFS

Description

The fully-qualified filename of the Kerberos keytab associated with the principal for accessing HDFS.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the Kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for HDFS

Description

The Kerberos principal used for authenticating to HDFS.

Default Value

default

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Maximum File Size

Description

The maximum size of the output data file. No size limit is applied if this property is not specified. Example values: 100 MB, 1 GB.

Default Value

Accepted Values

Required

false

Maximum Number of Entries

Description

The maximum number of entries in the output data file. In the context of this property, entry can mean one of two things. If record processing is enabled (Output File Data Format is set to Avro or JSON), an entry is a record. Otherwise, entry means a Kafka message. Set this property to 1 if you want to create a separate output file for each Kafka message.

Default Value

1000000

Accepted Values

Required

true

Output Directory Pattern

Description

Specifies the full path of output HDFS directory. The pattern can contain string literals (fixed text) as well as the ${directory.timestamp} expression, which inserts the current timestamp in the directory name.

Default Value

Accepted Values

Required

true

Output Directory Timestamp Format

Description

The timestamp format to use for the ${directory.timestamp} expression. For example: yyyyMMdd.

Default Value

Accepted Values

Required

false

Output File Data Format

Description

Specifies the format of the records written to the output file. Required when record processing is enabled (Kafka Message Data Format is set to Avro or JSON).

Default Value

Avro

Accepted Values

Avro, JSON, CSV, Parquet

Required

false

Output File Demarcator

Description

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages (no record processing). This property can only be used if Kafka Message Data Format is set to Raw. If you want to use newline as the delimiter, set this property to \n.

Default Value

Accepted Values

Required

false

Output Filename Pattern

Description

Specifies the structure of the name of the output file. The pattern can contain string literal (fixed text) parts and one or more of the following expressions:

- ${filename.uuid}: Inserts a generated UUID in the filename.
- ${filename.timestamp}: Inserts the current timestamp in the filename.
- ${filename.sequence}: Inserts an incrementing sequence value in the filename.

In order to generate unique filenames, either ${filename.uuid} or ${filename.sequence} must be used in the pattern.

Examples:

- data_${filename.uuid}.json
- records_${filename.timestamp}_${filename.sequence}.avro

Default Value

${filename.uuid}

Accepted Values

Required

false

Output Filename Sequence Initial Value

Description

This property is used to configure the initial value of the ${filename.sequence} expression. The value you set in this property is not the initial value of the sequence. The sequence starts at the value of this property +1. For example, if you set this property to 0, the sequence starts at 1.

Default Value

0

Accepted Values

Required

false

Output Filename Sequence Padding Length

Description

Specifies the length of the ${filename.sequence} expression in characters. If the sequence has fewer characters than the value set in this property, it is padded with zeros (0). Padding is added to the left of the sequence.

Default Value

6

Accepted Values

Required

false

Output Filename Timestamp Format

Description

The timestamp format to use for the ${filename.timestamp} expression. For example, yyyyMMdd_HHmmss_SSS.

Default Value

Accepted Values

Required

false

Output Files by Kafka Partitions

Description

Controls how fetched messages are merged together in the output file. If set to true, only messages fetched from the same Kafka partition get merged together in one output file (provided that Maximum Number of Entries is greater than 1). If set to false, messages are merged together in the order they are fetched from the Kafka topic. This property has no effect if Maximum Number of Entries is 1.

Default Value

false

Accepted Values

true, false

Required

false

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

This property is not used if record processing is disabled (Kafka Message Data Format is set to Raw).

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry. If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when reading or writing Time fields to JSON or CSV.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when reading or writing Timestamp fields to JSON or CSV.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using HTTPS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

HTTP Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the HTTP Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Basic Authentication Password

Description

The password to be used for authentication when connecting to the URL specified in the Remote U RL property. If an authentication method other than Basic Authentication is used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Basic Authentication Username

Description

The username used for authentication when connecting to the URL specified in the Remote U RL property. Cannot include control characters (0-31), ':', or DEL (127). If an authentication method other than Basic Authentication is used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Content-Type

Description

The Content-Type to specify in the HTTP header of the POST request sent by this connector.

Default Value

application/octet-stream

Accepted Values

Required

true

Date Format

Description

Specifies the format to use when reading date fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the date fields in the output JSON message.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Forward Raw Data

Description

Specifies whether messages from Kafka should be forwarded as is or converted to JSON. If set to false the Kafka Message Data Format parameter must be specified.

Default Value

true

Accepted Values

true, false

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka. If the Forward Raw Data property is set to true then this property is ignored. However, even in a case like this, this property must be assigned a valid value.

Default Value

AVRO

Accepted Values

AVRO, JSON

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the Kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Keystore Filename for Secure HTTP

Description

The fully-qualified filename of a keystore. This keystore is used to establish a secure connection with the HTTP server using mutual TLS.

If the HTTP server does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Keystore Key Password for Secure HTTP

Description

The password used to access the key stored in the keystore file configured in the Keystore Filename for Secure HTTP property.

If the HTTP server does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Keystore Password for Secure HTTP

Description

The password used to access the contents of the keystore configured in the Keystore Filename for Secure HTTP property.

If the HTTP server does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Keystore Type for Secure HTTP

Description

The type of the keystore configured in the Keystore Filename for Secure HTTP property.

If the HTTP server does not require client certificate authentication, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

false

Output Grouping for JSON

Description

Specifies how JSON objects are grouped in the connector output.

If set to output-array, the output will consist of an array of JSON objects.

f set to output-oneline, each line of the output data becomes one JSON object. That is, each JSON object occupies one line in the output.

This property is only used if forwarding raw data is disabled (Forward Raw Data is set to false). This is because when forwarding raw data is disabled, the data coming from Kafka gets converted to JSON. However, this property must still be set even if forwarding raw data is enabled.

Default Value

output-oneline

Accepted Values

output-array, output-oneline

Required

true

Remote URL

Description

The remote URL to connect to. Must include scheme, host, port, and path.

Default Value

https://localhost:22000/contentListener

Accepted Values

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

This property is not used if record processing is disabled (Forward Raw Data is set to true).

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry.

If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name.

If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when reading time fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the time fields in the output JSON message.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when reading timestamp fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the timestamp fields in the output JSON message.

Default Value

HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connection.

Accepted Values

Required

true

Truststore Filename for Secure HTTP

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with the HTTP server using TLS

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Password for Secure HTTP

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Secure HTTP property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Truststore Type for Secure HTTP

Description

The type of the truststore configured in the Truststore Filename for Secure HTTP property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

InfluxDB Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the InfluxDB Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Batch Size

Description

Maximum size of records merged together and put to InfluxDB in one batch. Example values: 100 MB, 1 GB

Default Value

1 MB

Accepted Values

Required

true

Consistency Level

Description

Specifies the InfluxDB consistency level

Default Value

ONE

Accepted Values

ONRE ANY, ALL, QUORUM

Required

true

Date Format

Description

Specifies the format to use when reading date fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the date fields in the output JSON message.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

InfluxDB Connection URL

Description

Specifies the InfluxDB URL to connect to.

Default Value

http://localhost:8086

Accepted Values

Required

true

InfluxDB Database Name

Description

The name of the InfluxDB database to connect to.

Default Value

Accepted Values

Required

true

InfluxDB User Name

Description

The username for accessing InfluxDB.

Default Value

Accepted Values

Required

false

InfluxDB User Password

Description

The password for InfluxDB user.

Default Value

Accepted Values

Required

false

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka. If the Forward Raw Data property is set to true then this property is ignored. However, even in a case like this, this property must be assigned a valid value.

Default Value

AVRO

Accepted Values

AVRO, JSON

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the Kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Line Protocol Query

Description

A record schema based query which returns the attributes in line protocol format.

Default Value

Accepted Values

Required

true

Retention Policy

Description

Specifies the retention policy for saving records in InfluxDB.

Default Value

autogen

Accepted Values

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry.

If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name.

If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when reading time fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the time fields in the output JSON message.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when reading timestamp fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the timestamp fields in the output JSON message.

Default Value

HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connection.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

JDBC Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the JDBC Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Database Connection URL

Description

The database specific connection URL used for connecting to the database. For example, jdbc:postgresql://localhost:5432/postgres.

Default Value

Accepted Values

Required

true

Database Driver Class Name

Description

The database driver class name. For example, org.postgresql.Driver.

Default Value

Accepted Values

Required

true

Database Driver Location

Description

A comma-separated list of files or folders containing the JDBC client libraries.

Default Value

Accepted Values

Required

true

Database Schema Name

Description

The name of the database schema containing the table to load data.

Default Value

Accepted Values

Required

false

Database Table Name

Description

The name of the database table to load data.

Default Value

Accepted Values

Required

true

Database Type

Description

The database type used for generating database specific SQL queries.

Default Value

Generic

Accepted Values

Generic, Oracle, Oracle 12+, MS SQL 2008, MS SQL 2012+, MySQL, PostgreSQL

Required

true

Database User Name

Description

The database user name. If username/password authentication is not required by the database server, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Database User Password

Description

The database user password. If username/password authentication is not required by the database server, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Date Format

Description

Specifies the format to use when reading date fields from JSON.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka.

Default Value

Avro

Accepted Values

Avro, JSON

Required

true

Kerberos Keytab

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Quote Column Names

Description

Specifies whether to use quotes around the column names in the generated SQL statement.

Default Value

false

Accepted Values

true, false

Required

true

Quote Table Name

Description

Specifies whether to use quotes around the table name in the generated SQL statement.

Default Value

false

Accepted Values

true, false

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry. If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Statement Type

Description

Specifies the type of SQL statement that is generated to put data into the database.

Default Value

INSERT

Accepted Values

INSERT, INSERT_IGNORE, UPDATE, UPSERT

Required

true

Time Format

Description

Specifies the format to use when reading Time fields from JSON.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when reading Timestamp fields from JSON.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using HTTPS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Kudu Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Kudu Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the

accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Date Format

Description

Specifies the format to use when reading Date fields from JSON.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Handle Schema Drift

Description

Specifies whether to handle Schema Drift. If set to true, when fields with names that are not in the target Kudu table are encountered, the Kudu table is altered to include new columns for those fields. If set to false, fields that only exist in the input data are skipped.

Default Value

false

Accepted Values

true, false

Required

true

Ignore Null

Description

Ignore NULL on Kudu Put Operation. If set to true, only non-null columns get updated.

Default Value

false

Accepted Values

true, false

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka.

Default Value

AVRO

Accepted Values

AVRO, JSON

Required

true

Kerberos Keytab for Kudu

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Kudu.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections

Accepted Values

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Kudu

Description

The Kerberos principal used for authenticating to Kudu.

Default Value

default

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Kudu Kerberos Service Name

Description

The SASL protocol name to use for authenticating with Kerberos. Must match the Kudu service principal name.

Default Value

kudu

Accepted Values

Required

false

Kudu Masters

Description

The comma separated addresses of the Kudu masters to connect to. For example: localhost: 7051 ,localhost:7151,localhost:7251.

Default Value

Accepted Values

Required

true

Kudu Operation Type

Description

Specifies what Kudu operation gets executed.

Default Value

INSERT

Accepted Values

INSERT, INSERT_IGNORE, UPSERT, UPDATE, DELETE, UPDATE_IGNORE, DELETE_IGNORE

Required

true

Lowercase Field Names

Description

Specifies whether to convert column names to lowercase when finding indexes of Kudu table columns.

Default Value

false

Accepted Values

true, false

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry. If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Table Name With Schema

Description

The schema and name of the Kudu Table to put data into. For example: default.mytable. When using Impala: impala::default.mytable.

Default Value

Accepted Values

Required

true

Time Format

Description

Specifies the format to use when reading Time fields from JSON.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when reading Timestamp fields from JSON. The value must match the Java Simple Date Format. (For example, MM/dd/yyyy HH:mm:ss for a two-digit month, followed by a two-digit day, followed by a four-digit year, all separated by forward slashes. This is followed by a two-digit hour in 24-hour format, followed by a two-digit minute, followed by a two-digit second, all separated by colons. The resulting timestamp looks like this: 01/01/2017 18:04:15.)

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

true

S3 Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the S3 Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Avro Schema Write Strategy

Description

Specifies whether the record schema is written to the output data file. Applicable only for Avro output (Output File Data Format is set to Avro).

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

AWS Access Key ID

Description

The Access Key ID to use for authentication to AWS.

Default Value

Accepted Values

Required

true

AWS Secret Access Key

Description

The Secret Access Key to use for authentication to AWS.

Default Value

Accepted Values

Required

true

Date Format

Description

Specifies the format to use when writing date fields to JSON or CSV.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka. If set to Avro or JSON, record processing is enabled. Raw can be used for unstructured text or binary data

Default Value

Avro

Accepted Values

Avro, JSON, Raw

Required

true

Kerberos Keytab

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Maximum File Size

Description

The maximum size of the output data file. No size limit is applied if this property is not specified. Example values: 100 MB, 1 GB.

Default Value

Accepted Values

Required

false

Maximum Number of Entries

Description

The maximum number of entries in the output data file. In the context of this property, entry can mean one of two things. If record processing is enabled (Kafka Message Data Format is set to Avro or JSON), an entry is a record. Otherwise, entry means a Kafka message. Set this property to 1 if you want to create a separate output file for each Kafka message.

Default Value

1000000

Accepted Values

Required

true

Output File Data Format

Description

Specifies the format of the records written to the output file. Required when record processing is enabled (Kafka Message Data Format is set to Avro or JSON).

Default Value

Avro

Accepted Values

Avro, JSON, CSV, Parquet

Required

Output File Demarcator

Description

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages (no record processing). This property can only be used if Kafka Message Data Format is set to Raw. If you want to use newline as the delimiter, set this property to \n.

Default Value

Accepted Values

Required

false

Output Filename Pattern

Description

Specifies the structure of the name of the output file (S3 object name including the prefix). The pattern can contain string literal (fixed text) parts and one or more of the following expressions:

- ${filename.uuid}: Inserts a generated UUID in the filename.
- ${filename.timestamp}: Inserts the current timestamp in the filename.
- ${filename.sequence}: Inserts an incrementing sequence value in the filename.

In order to generate unique filenames, either ${filename.uuid} or ${filename.sequence} must be used in the pattern.

Examples:

- data_${filename.uuid}.json
- records_${filename.timestamp}_${filename.sequence}.avro

Default Value

${filename.uuid}

Accepted Values

Required

false

Output Filename Sequence Initial Value

Description

This property is used to configure the initial value of the ${filename.sequence} expression. The value you set in this property is not the initial value of the sequence. The sequence starts at the value of this property +1. For example, if you set this property to 0, the sequence starts at 1.

Default Value

0

Accepted Values

Required

false

Output Filename Sequence Padding Length

Description

Specifies the length of the ${filename.sequence} expression in characters. If the sequence has fewer characters than the value set in this property, it is padded with zeros (0). Padding is added to the left of the sequence.

Default Value

6

Accepted Values

Required

false

Output Filename Timestamp Format

Description

The timestamp format to use for the ${filename.timestamp} expression. For example, yyyyMMdd_HHmmss_SSS.

Default Value

Accepted Values

Required

false

Parquet Compression Type

Description

The type of compression used for writing parquet files. Required when Output File Data Format is Parquet.

Default Value

UNCOMPRESSED

Accepted Values

UNCOMPRESSED, SNAPPY, GZIP, LZO, BROTLI, LZ4, ZSTD

Required

false

S3 Bucket

Description

The name of the S3 Bucket where the output file is uploaded.

Default Value

Accepted Values

Required

false

S3 Region

Description

The Region of the S3 Bucket.

Default Value

Accepted Values

Required

true

S3 Storage Class

Description

The Storage Class to use for the output file on S3.

Default Value

Standard

Accepted Values

Standard, IntelligentTiering, StandardInfrequentAccess, OneZoneInfrequentAccess, Glacier, GlacierInstantRetrieval, DeepArchive, ReducedRedundancy

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

This property is not used if record processing is disabled (Kafka Message Data Format is set to Raw).

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. If Schema Registry is not used, this property must be completely removed from the configuration JSON

Default Value

Accepted Values

Required

false

Schema Name

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when writing Time fields to JSON or CSV.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when writing Timestamp fields to JSON or CSV.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using HTTPS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Stateless NiFi Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Sink connector.

In addition to the properties listed here, this connector also accepts the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

attribute.prefix

Description

The prefix to add to the key of each header that matches the regular expression specified in headers.as.attributes.regex. For example, if the header key is MyHeader, its value is MyValue, headers.as.attributes.regex is set to My.*, and this property is set to kafka, the flowfile that is created for the Kafka message will have an attribute named kafka.MyHeader with a value of MyValue.

Default Value

Accepted Values

Required

false

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

failure.ports

Description

A comma separated list of output ports that are considered as failure conditions. If any flowfile is routed to an output port specified in this property, the dataflow is considered a failure and the session is rolled back. After a set amount of time, the dataflow reattempts to process the Kafka record. Any data transferred to an output port that is not in the list of failure ports is discarded.

Because of how Stateless NiFi Sink connectors behave, even if a single flowfile ends up in an output port that is marked as failure, the entire sessions is rolled back with all messages in the batch. Furthermore, if a flowfile ends up in a failure port in each subsequent iteration, the result is an endless loop. With some sink connectors (for example. MQTT Sink) this is the desired behavior. For more information regarding this behavior, see Dataflow execution and scheduling.

Default Value

Accepted Values

Required

false

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

headers.as.attributes.regex

Description

A Java regular expression that is evaluated against all Kafka record headers. Headers are added to the flowfile as an attribute if the header key matches the regular expression. The header key is used as the attribute name. The header value is used as the attribute value. Additionally, the name of the attribute can also contain an optional prefix which is defined by the attribute.prefix property.

Default Value

Accepted Values

Required

false

input.port

Description

The name of the input port in the NiFi dataflow that Kafka records are sent to. If the dataflow contains exactly one input port, this property is optional and can be omitted. However, if the dataflow contains multiple input ports, this property must be specified.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

true

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific parameter contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the Parameter Context named My Context.

Default Value

Accepted Values

Required

false

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

Cloudera Runtime 7.3.1

Developing Apache Kafka Applications

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Kafka producers 4

Kafka consumers 5
- Subscribing to a topic 7
- Groups and fetching 9
- Protocol between consumer and broker 10
- Rebalancing partitions 13
- Retries 13
- Kafka clients and ZooKeeper 14

Java client 15
- Java Client examples 16
  - Simple Java consumer 16
  - Simple Java producer 17
- Java client security examples 18

.NET client 19
- .Net client examples 20
  - Simple .NET consumer example 20
  - Simple .NET producer example 21
  - Performant .NET producer example 21
  - Simple .Net consumer using Schema Registry 22
  - Simple .Net producer using Schema Registry 23
- .NET client security examples 25

Kafka Streams 27

Kafka public APIs 27

Recommendations for using the producer and consumer APIs 28

Kafka producers

Learn more about Kafka producers and their most important configuration properties.

Kafka producers are the publishers responsible for writing records to topics. Typically, this means writing a program using the producer API available in your chosen client library. To instantiate a producer:

For Java

```
KafkaProducer<String, String> producer = new
KafkaProducer<>(producerConfig);
```

For C#

```
var producer = new ProducerBuilder<string, string>(config).Build();
```

Most of the important producer settings, and mentioned below, are in the configuration passed by this constructor.

Serialization of Keys and Values

For Java

For each producer, there are two serialization properties that must be set, key.serializer (for the key) and value.serializer (for the value). You can write custom code for serialization or use one of the ones already provided by Kafka. Some of the more commonly used ones are:

- ByteArraySerializer: Binary data
- StringSerializer: String representations

For C#

Serialization is passed to the producer builder object for key and value serialization:

```
var producer = new ProducerBuilder<string, string>(producerConfi
g)
                    .SetKeySerializer(Serializers.Utf8)
                    .SetValueSerializer(Serializers.ByteArray)
                    .Build())
```

Managing Record Throughput

There are several settings to control how many records a producer accumulates before actually sending the data to the cluster. This tuning is highly dependent on the data source. Some possibilities include:

For Java

- batch.size: Combine this fixed number of records before sending data to the cluster.
- linger.ms: Always wait at least this amount of time before sending data to the cluster; then send however many records has accumulated in that time.
- max.request.size: Put an absolute limit on data size sent. This technique prevents network congestion caused by a single transfer request containing a large amount of data relative to the network speed.
- compression.type: Enable compression of data being sent.
- retries: Enable the client for retries based on transient network errors. Used for reliability.

For C#

- BatchNumMessages: Combine this fixed number of records before sending data to the cluster.
- LingerMS: Always wait at least this amount of time before sending data to the cluster; then send however many records has accumulated in that time.
- CompressionType: Enable compression of data being sent.
- MessageSendMaxRetries: Maximum number of retries for sending a failed message.

Acknowledgments

The full write path for records from a producer is to the leader partition and then to all of the follower replicas. The producer can control which point in the path triggers an acknowledgment. Depending on the acks setting, the producer may wait for the write to propagate all the way through the system or only wait for the earliest success point.

Valid acks values are:

- 0: Do not wait for any acknowledgment from the partition (fastest throughput).
- 1: Wait only for the leader partition response.
- all: Wait for follower partitions responses to meet minimum (slowest throughput).

Partitioning

In Kafka, the partitioner determines how records map to partitions. Use the mapping to ensure the order of records within a partition and manage the balance of messages across partitions. The default partitioner uses the entire key to determine which partition a message corresponds to. Records with the same key are always mapped to the same partition (assuming the number of partitions does not change for a topic). Consider writing a custom partitioner if you have information about how your records are distributed that can produce more efficient load balancing across partitions. A custom partitioner lets you take advantage of the other data in the record to control partitioning.

For Java

If a partitioner is not provided to the KafkaProducer, Kafka uses a default partitioner. The ProducerRecord class is the actual object processed by the KafkaProducer. It takes the following parameters:

- Kafka Record: The key and value to be stored.
- Intended Destination: The destination topic and the specific partition (optional).

For C#

You can set what partitioner to use with the Partitioner property of ProducerConfig. By default the consistent_random partitioner is used. In C# you define the key and value types in the ProducerBuilder. When a new message is sent to a topic, a Message<Key,Value> object is processed with the key and value types specified in ProducerBuilder.

Kafka consumers

Learn more about Kafka consumers.

Kafka consumers are the subscribers responsible for reading records from one or more topics and one or more partitions of a topic. Consumers subscribing to a topic can happen manually or automatically; typically, this means writing a program using the consumer API available in your chosen client library.

To instantiate a consumer:

For Java

```
KafkaConsumer<String, String> kafkaConsumer = new
KafkaConsumer<>(consumerConfig);
```

For C#

```
var consumer = new ConsumerBuilder<string, string>(config).Build();
```

The consumer class has two generic type parameters. Just as producers can send data (the values) with keys, the consumer can read data by keys. In this example both the keys and values are strings. If you define different types, you need to define a deserializer to accommodate the alternate types. For deserializers you need to implement an interface:

- For Java implement the org.apache.kafka.common.serialization.Deserializer interface
- For C# implement the Confluent.Kafka.IDeserializer interface

The most important configuration parameters that we need to specify are:

For Java

- bootstrap.servers: A list of brokers to initially connect to. List 2 to 3 brokers; you don't needed to list the full cluster.
- group.id: Every consumer belongs to a group. That way they'll share the partitions of a topic.
- key.deserializer/value.deserializer: Specify how to convert the Java representation to a sequence of bytes to send data through the Kafka protocol.

For C#

- BootstrapServers: A list of brokers to initially connect to. List 2 to 3 brokers; you don't need to list the full cluster.
- GroupId: Every consumer belongs to a group. That way they'll share the partitions of a topic.
- key.deserializer/value.deserializer: Specify how to convert the Java representation to a sequence of bytes to send data through the Kafka protocol.
- Key and value deserializer: In C# the key and value deserializers are not configuration parameters, instead they are passed to the builder object. For example:

```
var consumer = new ConsumerBuilder<string, GenericRecord>(config)
                        .SetKeyDeserializer(new AvroDeserializer<strin
g>(schemaRegistry).AsSyncOverAsync())
                        .SetValueDeserializer(new AvroDeserializer<Gen
ericRecord>(schemaRegistry).AsSyncOverAsync())
                        .Build()
```

In addition to the configuration properties presented above, there are a number of other important configurations that any user of Kafka must know about. These are:

For Java

- heartbeat.interval.ms: The interval of the heartbeats. For example, if the heartbeat interval is set to 3 seconds, the consumer sends a short heartbeat message to the broker every 3 seconds to indicate that it is alive.
- session.timeout.ms: The consumer tells this timeout to the coordinator. This is used to control the heartbeats and remove the dead consumers. If it's set to 10 seconds, the consumer can miss sending 2 heartbeats, assuming the previous heartbeat setting. If we increase the timeout, the consumer has more room for delays but the broker notices lagging consumers later.
- max.poll.interval.ms: It is a very important detail: the consumers must maintain polling and should never do long-running processing. If a consumer is taking too much time between two polls, it will be detached from the consumer group. We can tune this configuration according to our needs. Note that if a consumer is stuck in processing, it will be noticed later if the value is increased.
- request.timeout.ms: Generally every request has a timeout. This is an upper bound that the client waits for the server's response. If this timeout elapses, then retries might happen if the number of retries are not exhausted.

For C#

- HeartbeatIntervalMs: The interval of the heartbeats. For example, if the heartbeat interval is set to 3 seconds, the consumer sends a short heartbeat message to the broker every 3 seconds to indicate that it is alive.
- SessionTimeoutMs: The consumer tells this timeout to the coordinator. This is used to control the heartbeats and remove the dead consumers. If it's set to 10 seconds, the consumer can miss sending 2 heartbeats, assuming the previous heartbeat setting. If we increase the timeout, the consumer has more room for delays but the broker notices lagging consumers later.
- MaxPollIntervalMs: It is a very important detail: the consumers must maintain polling and should never do long-running processing. If a consumer is taking too much time between two polls, it will be detached from the consumer group. We can tune this configuration according to our needs. Note that if a consumer is stuck in processing, it will be noticed later if the value is increased.

Subscribing to a topic

Learn more about subscribing to a topic.

Subscribing to a topic

In order for the consumer to be able to consume messages, it first needs to subscribe to a topic. This can be done using a subscribe method:

For Java

```
kafkaConsumer.subscribe(Collections.singletonList(topic), rebalanceListe
ner);
```

For C#

```
consumer.Subscribe(new List<string>() { topic });
```

Here you specify a list of topics that you want to consume from. In Java a 'rebalance listener' is also specified. Rebalancing is an important part of the consumer's life. Whenever the cluster or the consumers' state changes, a rebalance will be issued. This will ensure that all the partitions are assigned to a consumer.

Polling

After subscribing to a topic, the consumer has to poll to see if there are new records.

For Java

The poll () method returns multiple records that can be processed by the client. After processing the records the client commits offsets synchronously, thus waiting until processing completes before continuing to poll.

```
while (true) {
  data = kafkaConsumer.poll();
  // do something with 'data'
}
```

For C#

The Consume() method returns a single result which can either be a single Kafka message, or an end of partition event if the IsPartitionEOF property is true.

```
while (true)
```

```
{
    var result = consumer.Consume();
    // Handle result
}
```

Saving progress

The last important point is to save the progress.

In Java this can be done with by the commitSync() and commitAsync() methods respectively. Auto commit is not recommended; manual commit is appropriate in the majority of use cases.

In C# this can be done by calling the Close() method at the end of message processing which automatically commits offsets. Alternatively, this can also be done manually by calling the Commit() method:

For Java

commitSync()

```
kafkaConsumer.subscribe(Collections.singletonList(topic), rebala
nceListener);

        while (true) {
            ConsumerRecords<String, String> records = kafkaCon
sumer.poll(1000);
            for (ConsumerRecord<String, String> record : recor
ds) {
                System.out.printf("Received Message with topic =
%s, partition =%s, offset = %d, key = %s, value = %s\n",
                        record.topic(), record.partition(), recor
d.offset(), record.key(), record.value());
            }
            // commit and wait until the offset is committed
            kafkaConsumer.commitSync();
        }
```

commitAsync()

```
kafkaConsumer.subscribe(Collections.singletonList(topic), rebala
nceListener);

        while (true) {
            ConsumerRecords<String, String> records = kafkaCon
sumer.poll(1000);
            for (ConsumerRecord<String, String> record : recor
ds) {
                System.out.printf("Received Message with topic =
%s, partition =%s, offset = %d, key = %s, value = %s\n",
                        record.topic(), record.partition(), recor
d.offset(), record.key(), record.value());
            }

            // Commit the offset and proceed with execution. The
 callback will be invoked when the offset
            // commit's result comes back from the broker.
            kafkaConsumer.commitAsync((offsets, exception) -> {
                if (exception != null) {
                    // handle the error that happened during off
set commit
                } else {
                    // do something on successful offset commit
if needed
```

```
                }
            });
        }
```

For C#

Close()

```
consumer.Close();  // commit offset and unsubscribe
```

Commit()

```
consumer.Commit(); // commit offset
```

Groups and fetching

Learn more about how consumer groups fetch messages.

Kafka consumers are usually assigned to a group. This happens statically by setting the group.id configuration property in the consumer configuration. Consuming with groups will result in the consumers balancing the load in the group. That means each consumer will have their fair share of partitions. Also it can never be more consumers than partitions as that way there would be idling consumers.

As shown in the figure below, both consumer groups share the partitions and each partition multicasts messages to both consumer groups. The consumers pull messages from the broker instead of the broker periodically pushing what is available. This helps the consumer as it won't be overloaded and it can query the broker at its own speed. Furthermore, to avoid tight looping, it uses a so called "long-poll". The consumer sends a fetch request to poll for data and receives a reply only when enough data accumulates on the broker.

Figure 1: Consumer Groups and Fetching from Partitions

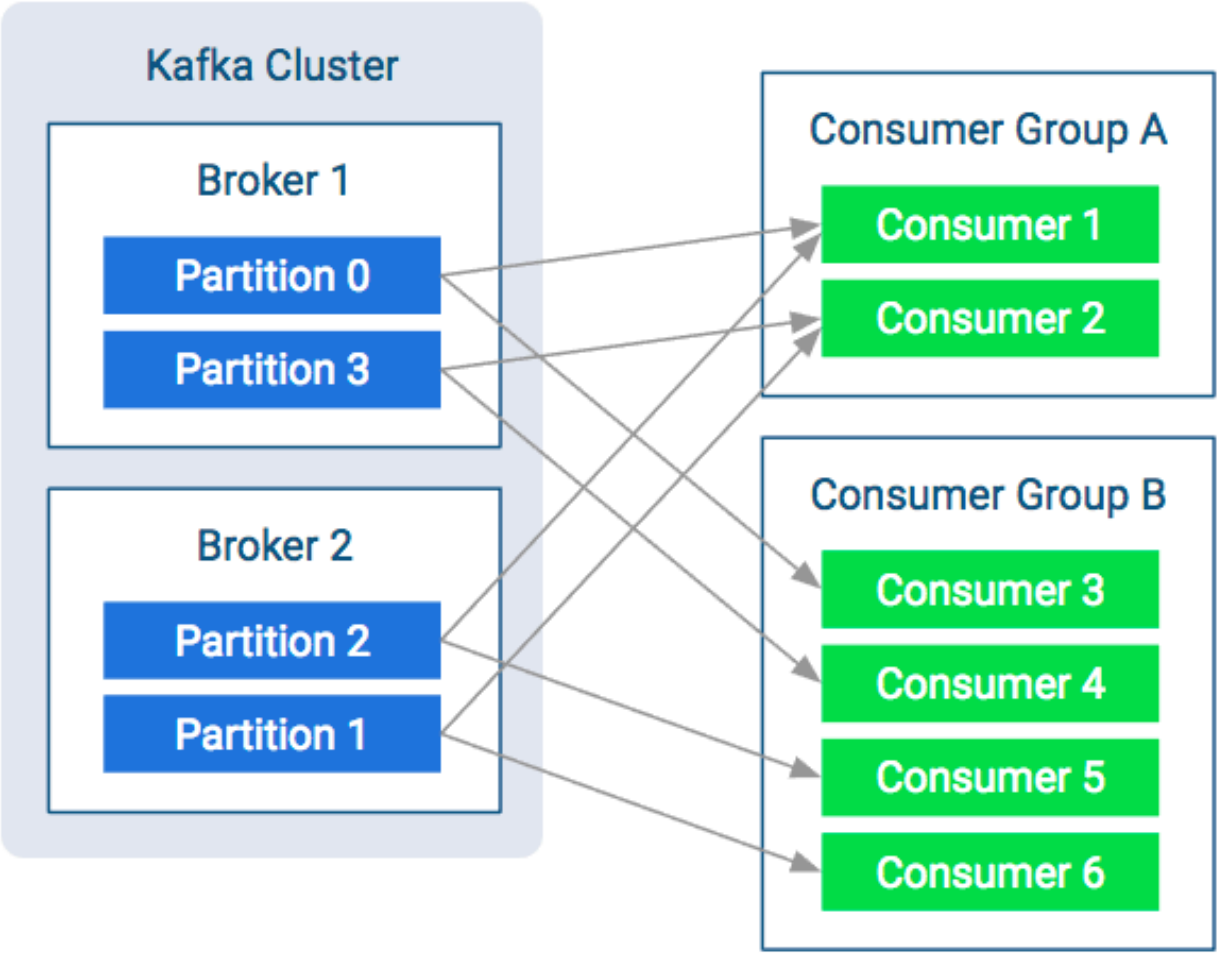

Protocol between consumer and broker

Get to know how the protocol works, what messages are going on the wire, and how that contributes to the overall behavior of the consumer.

When discussing the internals of the consumers, there are a couple of basic terms to know:

Heartbeat

When the consumer is alive and is part of the consumer group, it sends heartbeats. These are short periodic messages that tell the brokers that the consumer is alive and everything is fine.

Session

Often one missing heartbeat is not a big deal, but how do you know if a consumer is not sending heartbeats for long enough to indicate a problem? A session is such a time interval. If the consumer didn't send any heartbeats for longer than the session, the broker can consider the consumer dead and remove it from the group.

Coordinator

The special broker which manages the group on the broker side is called the coordinator. The coordinator handles heartbeats and assigns the leader. Every group has a coordinator that organizes the startup of a consumer group and assist whenever a consumer leaves the group.

Leader

The leader consumer is elected by the coordinator. Its job is to assign partitions to every consumer in the group at startup or whenever a consumer leaves or joins the group. The leader holds the

assignment strategy, it is decoupled from the broker. That means consumers can reconfigure the partition assignment strategy without restarting the brokers.

Startup Protocol

As mentioned before, the consumers are working usually in groups. So a major part of the startup process is spent with figuring out the consumer group.

At startup, the first step is to match protocol versions. It is possible that the broker and the consumer are of different versions (the broker is older and the consumer is newer, or vice versa). This matching is done by the API_VERS IONS request.

Figure 2: Startup Protocol

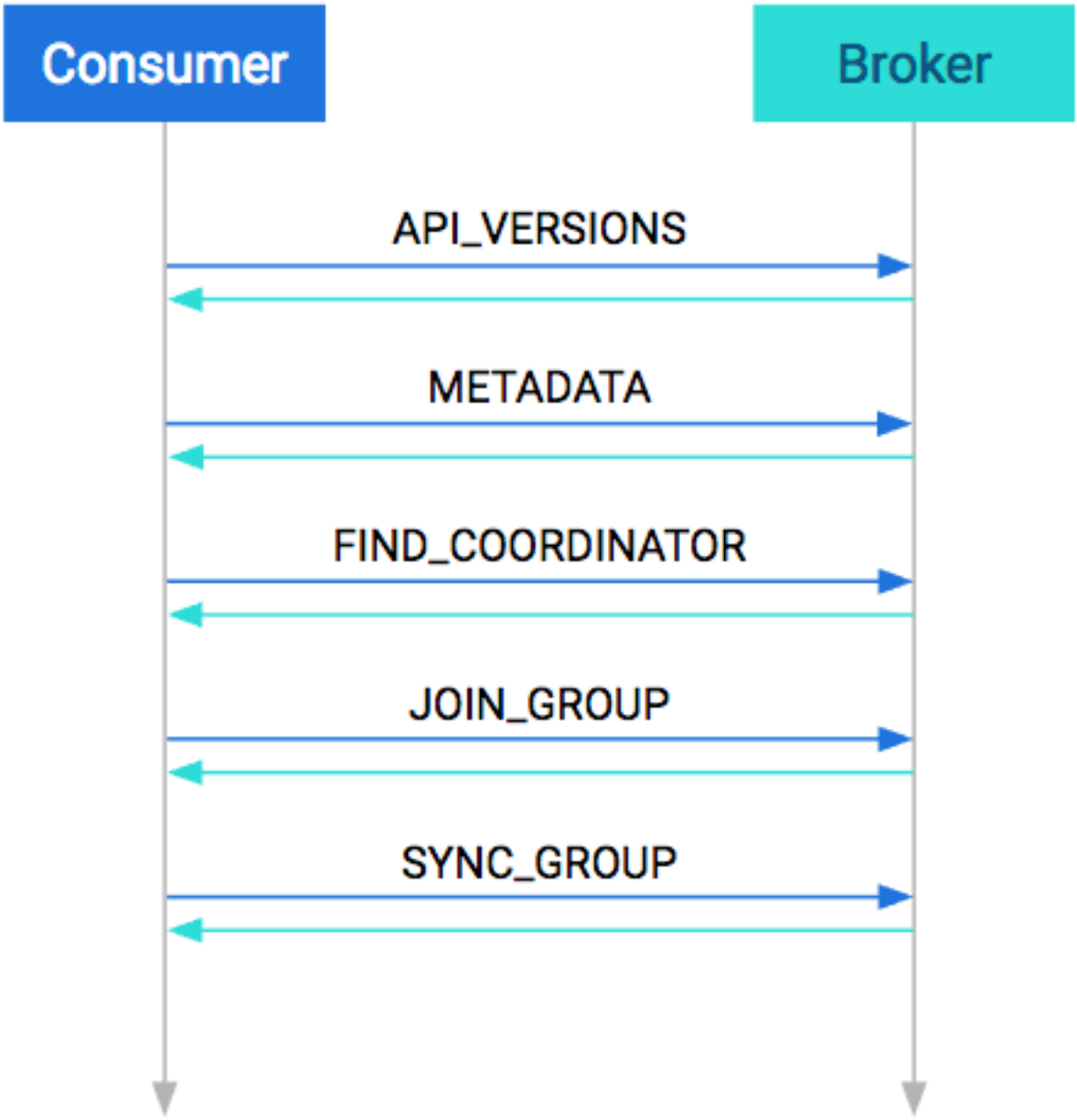

The next step is to collect cluster information, such as the addresses of all the brokers (prior to this point we used the bootstrap server as a reference), partition counts, and partition leaders. This is done in the METADATA request.

After acquiring the metadata, the consumer has the information needed to join the group. By this time on the broker side, a coordinator has been selected per consumer group. The consumers must find their coordinator with the FIND _COORDINATOR request.

After finding the coordinator, the consumer(s) are ready to join the group. Every consumer in the group sends their own member-specific metadata to the coordinator in the JOIN_GROUP request. The coordinator waits until all the consumers have sent their request, then assigns a leader for the group. At the response plus the collected metadata are sent to the leader, so it knows about its group.

The remaining step is to assign partitions to consumers and propagate this state. Similar to the previous request, all consumers send a SYNC_GROUP request to the coordinator; the leader provides the assignments in this request. After it receives the sync request from each group member, the coordinator propagates this member state in the response. By the end of this step, the consumers are ready and can start consuming.

Consumption Protocol

When consuming, the first step is to query where should the consumer start. This is done in the OFFSET_FETCH request. This is not mandatory: the consumer can also provide the offset manually. After this, the consumer is free to pull data from the broker. Data consumption happens in the FETCH requests. These are the long-pull requests. They are answered only when the broker has enough data; the request can be outstanding for a longer period of time.

Figure 3: Consumption Protocol

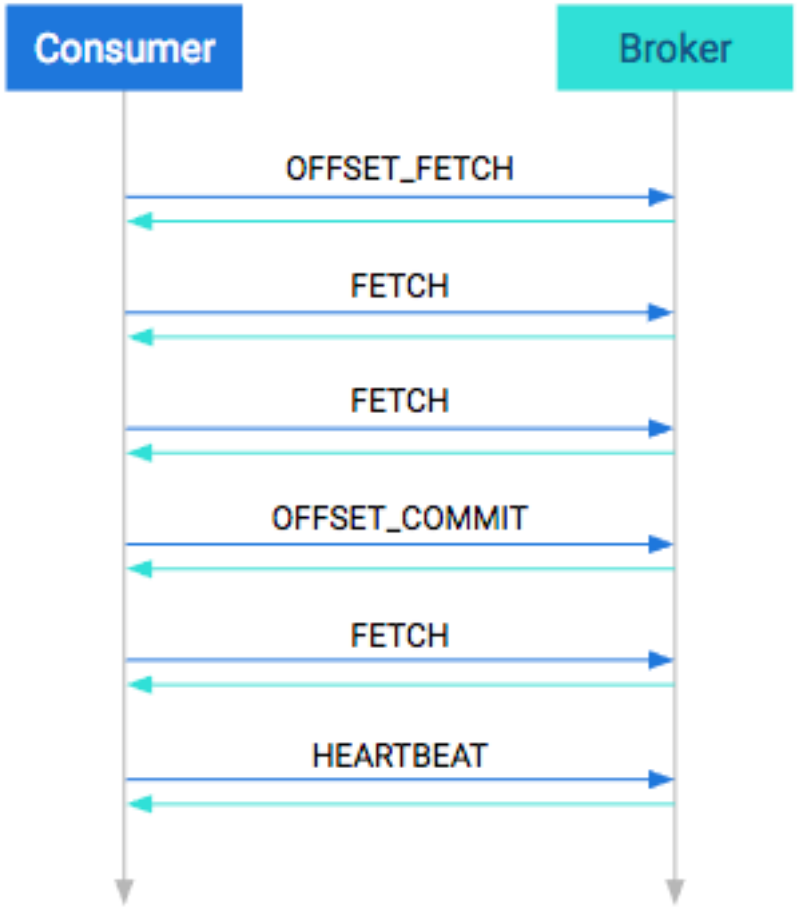

From time to time, the application has to either manually or automatically save the offsets in an OFFSET_COMMIT request and send heartbeats too in the HEARTBEAT requests. The first ensures that the position is saved while the latter ensures that the coordinator knows that the consumer is alive.

Shutdown Protocol

The last step when the consumption is done is to shut down the consumer gracefully. This is done in one single step, called the LEAVE_GROUP protocol.

Figure 4: Shutdown Protocol

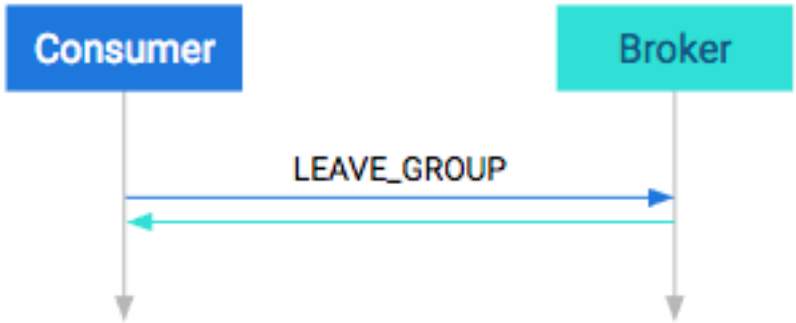

Rebalancing partitions

Learn what rebalancing is, when it can occur, and how its propagated to the client.

There are multiple points in the protocol between consumers and brokers where failures can occur. There are points in the normal operation of the system where you need to change the consumer group assignments. For example, to consume a new partition or to respond to a consumer going offline. The process or responding to cluster information changing is called rebalance. It can occur in the following cases:

- A consumer leaves. It can be a software failure where the session times out or a connection stalls for too long, but it can also be a graceful shutdown.
- A consumer joins. It can be a new consumer but an old one that just recovered from a software failure (automatically or manually).
- Partition is adjusted. A partition can simply go offline because of a broker failure or a partition coming back online. Alternatively an administrator can add or remove partitions to/from the broker. In these cases the consumers must reassign who is consuming.
- The cluster is adjusted. When a broker goes offline, the partitions that are lead by this broker will be reassigned. In turn the consumers must rebalance so that they consume from the new leader. When a broker comes back, then eventually a preferred leader election happens which restores the original leadership. The consumers must follow this change as well.

On the consumer side, this rebalance is propagated to the client via the ConsumerRebalanceListener interface. It has two methods. The first, onPartitionsRevoked, will be invoked when any partition goes offline. This call happens before the changes would reflect in any of the consumers, so this is the chance to save offsets if manual offset commit is used. On the other hand onPartitionsAssigned is invoked after partition reassignment. This would allow for the programmer to detect which partitions are currently assigned to the current consumer. Complete examples can be found in the development section.

Retries

Learn more about retries and how they are constrained.

In Kafka retries typically happen only for certain kinds of errors. When a retriable error is returned, the clients are constrained by two facts: the timeout period and the backoff period.

The timeout period tells how long the consumer can retry the operation. The backoff period how often the consumer should retry. There is no generic approach for "number of retries." Number of retries are usually controlled by timeout periods.

Kafka clients and ZooKeeper

Learn more about the differences between the old and new model for storing consumer offsets.

The default consumer model provides the metadata for offsets in the Kafka cluster. There is a topic named __consumer_offsets that the Kafka consumers write their offsets to.

Figure 5: Kafka Consumer Dependencies

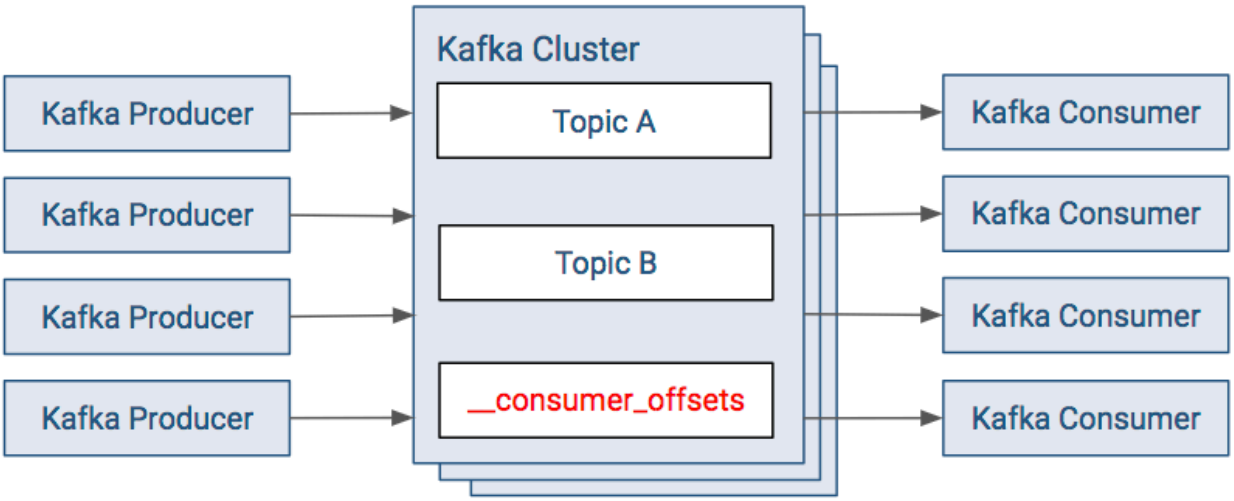

In releases before version 2.0 of CDK Powered by Apache Kafka, the same metadata was located in ZooKeeper. The new model removes the dependency and load from Zookeeper. In the old approach:

- The consumers save their offsets in a "consumer metadata" section of ZooKeeper.
- With most Kafka setups, there are often a large number of Kafka consumers. The resulting client load on ZooKeeper can be significant, therefore this solution is discouraged.

Figure 6: Kafka Consumer Dependencies (Old Approach)

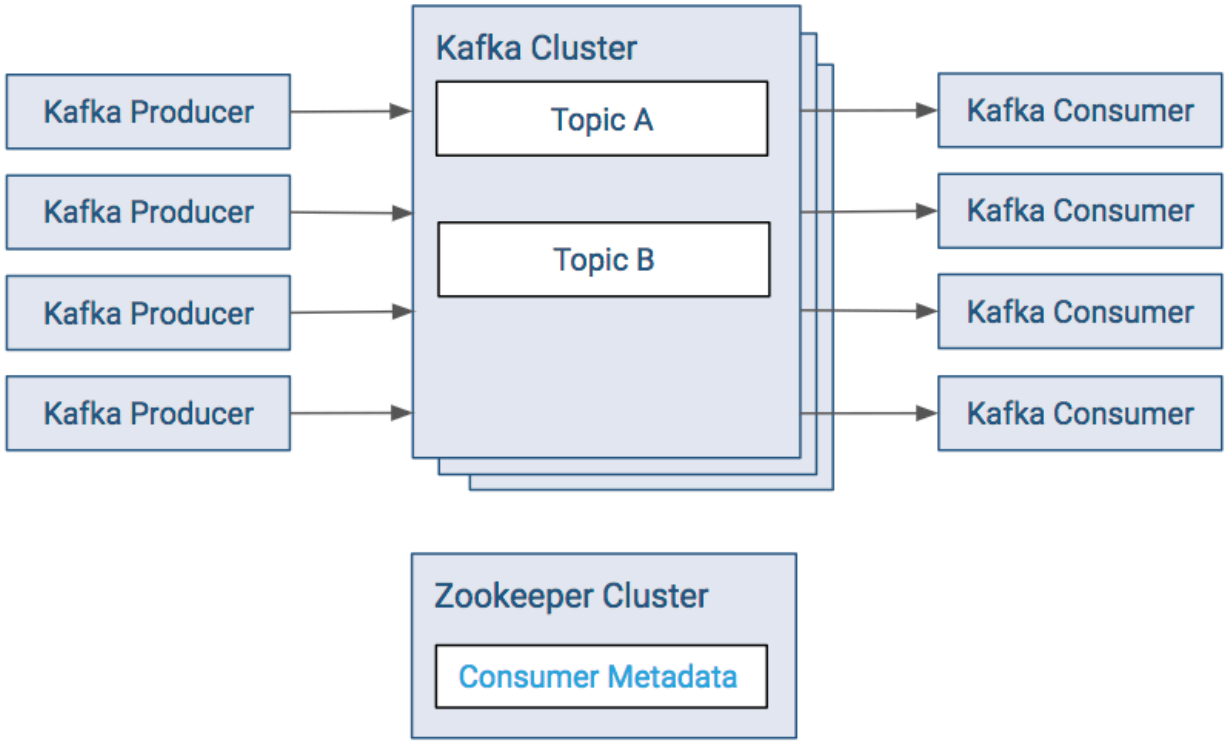

Java client

Learn how the Java client libraries are distributed and how you can add the necessary artifacts to your projects.

Cloudera Runtime is shipped with the Java client libraries that are maintained as part of Apache Kafka. These are available in the Cloudera Maven Repository. Review the following POM example to understand how you can add the necessary artifacts to your project.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache
.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.cloudera.kafkaexamples</groupId>
  <artifactId>kafka-examples</artifactId>
  <packaging>jar</packaging>
  <version>1.0</version>
  <name>kafkadev</name>
  <url>http://maven.apache.org</url>
  <repositories>
    <repository>
      <id>cloudera</id>
      <url>https://repository.cloudera.com/artifactory/cloudera-repos/</url>
    </repository>
  </repositories>
  <dependencies>
    <dependency>
      <groupId>org.apache.kafka</groupId>
      <artifactId>kafka-clients</artifactId>
      <version>[***KAFKA VERSION***]</version>
      <scope>compile</scope>
    </dependency>
```

```
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.7.0</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Note:

Replace [***KAFKA VERSION***] with the version of the kafka-clients artifact that you want to use. For available versions, see https://repository.cloudera.com/artifactory/cloudera-repos/org/apache/kafka/kafka-clients/.

Related Information

Cloudera Maven Repository

Java Client examples

You can produce messages to and consume messages from a Kafka cluster using the command line. For most cases however, running Kafka producers and consumers using shell scripts and Kafka's command line scripts cannot be used in practice. In these cases, native Kafka client development is the generally accepted option. Review these code example to better understand how you can develop your own clients using the Java client library.

Simple Java consumer

A simple working example of a producer program.

Note that this consumer is designed as an infinite loop. In normal operation of Kafka, all the producers could be idle while consumers are likely to be still running.

The example includes Java properties for setting up the client identified in the comments; the functional parts of the code are in bold. This code is compatible with versions as old as the 0.9.0-kafka-2.0.0 version of Kafka.

```
package com.cloudera.kafkaexamples;

import java.util.Arrays;
import java.util.Properties;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.kafka.common.serialization.StringDeserializer;

public class SimpleConsumer {
    public static void main(String[] args) {

        // Set up client Java properties
        Properties props = new Properties();
        props.setProperty(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
```

```
                "host1:9092,host2:9092,host3:9092");
        // Just a user-defined string to identify the consumer group
        props.put(ConsumerConfig.GROUP_ID_CONFIG, "test");
        // Enable auto offset commit
        props.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "true");
        props.put(ConsumerConfig.AUTO_COMMIT_INTERVAL_MS_CONFIG, "1000");
        props.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
                StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
                StringDeserializer.class.getName());

        try (KafkaConsumer<String, String> consumer = new
 KafkaConsumer<>(props)) {
            // List of topics to subscribe to
            consumer.subscribe(Arrays.asList("ufo_sightings"));
            while (true) {
                try {
                    ConsumerRecords<String, String> records =
 consumer.poll(100);
                    for (ConsumerRecord<String, String> record : records) {
                        System.out.printf("Offset = %d\n", record.offset());
                        System.out.printf("Key    = %s\n", record.key());
                        System.out.printf("Value  = %s\n", record.value());
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Simple Java producer

A simple working example of a producer program.

The example includes Java properties for setting up the client identified in the comments; the functional parts of the code are in bold. This code is compatible with versions as old as the 0.9.0-kafka-2.0.0 version of Kafka.

```
package com.cloudera.kafkaexamples;

import java.util.Date;
import java.util.Properties;

import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.common.serialization.StringSerializer;
public class SimpleProducer {
    public static void main(String[] args) {
        // Generate total consecutive events starting with ufoId
        long total = Long.parseLong("10");
        long ufoId = Math.round(Math.random() * Integer.MAX_VALUE);
        // Set up client Java properties
        Properties props = new Properties();
        props.setProperty(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
                "host1:9092,host2:9092,host3:9092");
        props.setProperty(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
                StringSerializer.class.getName());
        props.setProperty(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
                StringSerializer.class.getName());
        props.setProperty(ProducerConfig.ACKS_CONFIG, "1");
```

```
        try (KafkaProducer<String, String> producer = new
 KafkaProducer<>(props)) {
            for (long i = 0; i < total; i++) {
                String key = Long.toString(ufoId++);
                long runtime = new Date().getTime();
                double latitude = (Math.random() * (2 * 85.05112878)) - 85.
05112878;
                double longitude = (Math.random() * 360.0) - 180.0;
                String msg = runtime + "," + latitude + "," + longitude;
                try {
                    ProducerRecord<String, String> data = new
                            ProducerRecord<String, String>("ufo_sightings",
 key, msg);
                    producer.send(data);
                    long wait = Math.round(Math.random() * 25);
                    Thread.sleep(wait);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Java client security examples

Review the Java client security examples to learn what configuration properties you have to set when connecting to secured or unsecured clusters.

The following code snippets give you a few simple examples on what configuration properties you need to set for your Kafka clients when connecting them to either secured or unsecured Kafka clusters. Use the following examples as a starting point and make changes as necessary. Depending on your environment, you may need to set other optional properties. See the upstream Apache Kafka documentation for a comprehensive list of available properties.

Unsecured

```
Properties producerConfig = new Properties();
producerConfig.put("bootstrap.servers", "***BROKER HOST***:***PORT***");
```

SSL/TLS

```
Properties producerConfig = new Properties();
producerConfig.put("bootstrap.servers", "***BROKER HOST***:***PORT***");
producerConfig.put("security.protocol", "SSL");
producerConfig.put("ssl.truststore.location", "***PATH TO TRUSTSTORE***");
// .jks format
producerConfig.put("ssl.truststore.password", "***TRUSTSTORE PASSWORD***");
producerConfig.put("ssl.keystore.location", "***PATH TO KEYSTORE***"); // .
jks format
producerConfig.put("ssl.keystore.password", "***KEYSTORE PASSWORD***");
```

PLAIN (LDAP, PAM, and others)

Important: Credentials are sent across the wire, Cloudera recommends that you always use channel encryption.

```
Properties producerConfig = new Properties();
```

```
producerConfig.put("bootstrap.servers", "***BROKER HOST***:***PORT***");
producerConfig.put("security.protocol", "SASL_SSL");
producerConfig.put("sasl.mechanism", "PLAIN");
producerConfig.put("ssl.truststore.location", "***PATH TO TRUSTSTORE***");
// .jks format
producerConfig.put("ssl.truststore.password", "***TRUSTSTORE PASSWORD***");
producerConfig.put("sasl.jaas.config", "org.apache.kafka.common.secu
rity.plain.PlainLoginModule required username="***USERNAME***" passwo
rd="***PASSWORD***"); // credentials sent across the wire, also enable SSL/
TLS encryption
```

Kerberos

```
Properties producerConfig = new Properties();
producerConfig.put("bootstrap.servers", "***BROKER HOST***:***PORT***");
producerConfig.put("security.protocol", "SASL_PLAINTEXT");
producerConfig.put("sasl.kerberos.service.name", "***SERVICE NAME***"); // k
afka
producerConfig.put("sasl.jaas.config", "com.sun.security.auth.module.Krb5Lo
ginModule required useTicketCache=true;"); // use kerberos ticket cache
```

OAuth2/OAUTHBEARER

```
Properties producerConfig = new Properties();
producerConfig.put("bootstrap.servers", "***BROKER HOST***:***PORT***");
producerConfig.put("security.protocol", "SASL_PLAINTEXT");
producerConfig.put("sasl.mechanism", "OAUTHBEARER");
producerConfig.put("sasl.login.callback.handler.class", org.apache.kafka.com
mon.security.oauthbearer.secured.OAuthBearerLoginCallbackHandler.class.getNa
me());
producerConfig.put("sasl.jaas.config", "org.apache.kafka.common.security.oau
thbearer.OAuthBearerLoginModule required " +
                     "clientId=\"***CLIENT_ID***\" " +
                     "clientSecret=\"***CLIENT_SECRET***\" " +
                     "scope=\"***SCOPE***\";"
     );
producerConfig.put("sasl.oauthbearer.token.endpoint.url", "http://***OAUTH
 SERVER***/***TOKEN ENDPOINT***");
```

.NET client

Learn more about how you can download and install the Cloudera .NET client library for Apache Kafka.

Cloudera's .NET client library for Apache Kafka, called Cloudera.Kafka, is available through NuGet. This client library is not distributed or packaged with Cloudera Runtime.

Installation

The client library can be downloaded and installed through Visual Studio or a command line interface. Alternatively, you can also download it directly from nuget.org.

Visual Studio

In Visual Studio you can use the NuGet Package Manager UI to find and install the Cloudera.Kafka package for the appropriate project.

Command line interface

The Cloudera.Kafka package can be installed through a command line interface. You can use the Package Manager Console or the dotnet CLI.

Package Manager Console:

```
Install-Package Cloudera.Kafka -Version [***VERSION***]
```

dotnet CLI:

```
dotnet add package Cloudera.Kafka --version [***VERSION***]
```

Direct download

The Cloudera.Kafka package is available on the NuGet website at https://www.nuget.org/packages/Cloudera.Kafka/.

.Net client examples

You can produce messages to and consume messages from a Kafka cluster using the command line. For most cases however, running Kafka producers and consumers using shell scripts and Kafka's command line scripts cannot be used in practice. In these cases, built-in Kafka client development is the generally accepted option. Review these code example to better understand how you can develop your own clients using the Cloudera .Net client library.

Simple .NET consumer example

Review this example code to learn how you can create a simple .NET consumer. This consumer is capable of consuming around 50000 messages every second in development environment.

```
var config = new ConsumerConfig
{
    BootstrapServers = "broker:9092",
    GroupId = "simple-dotnet-consumer",
    AutoOffsetReset = AutoOffsetReset.Earliest,
    EnablePartitionEof = true
};
using var consumer = new ConsumerBuilder<string, string>(config)
    .SetErrorHandler((_, e) => Console.WriteLine($"Error: {e.Reason}"))
    .Build();
consumer.Subscribe(new List<string>() {"topic"});
var start = DateTime.Now;
long messageCounter = 0;
try
{
    while (!(Console.KeyAvailable && Console.ReadKey().Key == ConsoleKey.Q))
    {
        var result = consumer.Consume(TimeSpan.FromMilliseconds(100));
        if (result == null) { continue; }
        if (result.IsPartitionEOF) { break; }

        ++messageCounter;
        if (messageCounter % 1024 == 0) { Console.WriteLine($"Received mess
age key: \"{result.Message.Key}\" value: {result.Message.Value}"); }
    }
} catch (OperationCanceledException) {}

consumer.Close();  // commit offset and unsubscribe
var elapsed = DateTime.Now - start;
Console.WriteLine("average throughput: {0:N3} msg/sec, {1} messages over {2
:N3} sec", messageCounter / elapsed.TotalSeconds, messageCounter, elapsed.To
talSeconds);
```

Simple .NET producer example

Review this example code to learn how you can create a simple .NET producer. This producer is capable of producing around 36000 messages every second in development environment.

```
internal class Program
{
    private static int _inFlight;
    private static long _delivered;

    private static void HandleDeliveryResult(int msgid, DeliveryResult<strin
g, string> deliveryResult)
    {
        if (msgid % 1024 == 0)  // writing to console on every message would
 be a bottleneck
        {
            Console.WriteLine($"Delivered '{deliveryResult.Value}' to '{deli
veryResult.TopicPartitionOffset}', in flight on delivery confirmation: {_inF
light}");
        }
        --_inFlight;
        ++_delivered;
    }

    private static void Main(string[] args)
    {
        using var producer = new ProducerBuilder<string, string>(new Produce
rConfig { BootstrapServers = "broker:9092" }).Build();
        int msgCounter = 0;
        var start = DateTime.Now;
        while (!(Console.KeyAvailable && Console.ReadKey().Key == ConsoleKe
y.Q || (DateTime.Now - start).TotalSeconds >= 60))
        {
            int msgid = ++msgCounter;
            try
            {
                producer.Produce("topic", new Message<string, string> { Key
 = msgid.ToString(), Value = $"{msgid}, in flight on send: {++_inFlight}" },
                    result => HandleDeliveryResult(msgid, result));
            }
            catch (ProduceException<string, string> e)
            {
                Console.WriteLine($"Produce failed: {e.Error.Reason}");
            }
        }
        producer.Flush();
        Console.WriteLine("average throughput: {0:N3} msg/sec", _delivered /
 (DateTime.Now - start).TotalSeconds);
    }
}
```

Performant .NET producer example

Review this example code to learn how you can create a more complex, high-performing .NET producer with async processing and TPL. This producer is capable of producing around 50000 messages every second in development environment.

```
internal class Program
{
    private static int _inFlight;
    private static long _delivered;
```

```
    private static void HandleDeliveryResult(int msgid, DeliveryResult<strin
g, string> deliveryResult)
    {
        if (msgid % 1024 == 0)  // writing to console on every message would
 be a bottleneck
        {
            Console.WriteLine($"Delivered '{deliveryResult.Value}' to '{deli
veryResult.TopicPartitionOffset}', in flight on delivery confirmation: {_inF
light}");
        }
        Interlocked.Decrement(ref _inFlight);
        Interlocked.Increment(ref _delivered);
    }

    private static async Task Main(string[] args)
    {
        const int inFlightRequests = 16384;
        const int bufferedMessages = inFlightRequests * 4;
        var config = new ProducerConfig { BootstrapServers = "broker:9092",
 LingerMs = 50, QueueBufferingMaxMessages = bufferedMessages };
        using var p = new ProducerBuilder<string, string>(config).Build();
        var actionBlock = new ActionBlock<int>(
            msgid => p.ProduceAsync("topic", new Message<string, string> {
Key=msgid.ToString(), Value = $"{msgid}, in flight on send: {Interlocked.Inc
rement(ref _inFlight)}" }
                .ContinueWith(async task => HandleDeliveryResult(msgid, aw
ait task)),
            new ExecutionDataflowBlockOptions { MaxDegreeOfParallelism = inF
lightRequests, BoundedCapacity = inFlightRequests });

        int msgCounter = 0;
        bool accepted = true;
        var start = DateTime.Now;
        while (accepted && !(Console.KeyAvailable && Console.ReadKey().Key
 == ConsoleKey.Q || (DateTime.Now - start).TotalSeconds >= 60))
        {
            try
            {
                accepted = await actionBlock.SendAsync(msgCounter++);
            }
            catch (ProduceException<string, string> e)
            {
                Console.WriteLine($"Produce failed: {e.Error.Reason}");
            }
        }
        actionBlock.Complete();
        await actionBlock.Completion;
        Console.WriteLine("average throughput: {0:N3} msg/sec", _delivered /
 (DateTime.Now - start).TotalSeconds);
    }
}
```

Simple .Net consumer using Schema Registry

Review this example code to learn how you can create a simple .NET consumer which is using Schema Registry.

```
internal class Program
{
    private static void Main(string[] args)
    {
        var schemaRegistryConfig = new SchemaRegistryConfig {
            Url = "http://schemareg:7788/api/v1"
        };
```

```
        var consumerConfig = new ConsumerConfig {
            BootstrapServers = "broker:9092",
            GroupId = "simple-dotnet-consumer",
            AutoOffsetReset = AutoOffsetReset.Earliest,
            EnablePartitionEof = true
        };

        using (var schemaRegistry = new CachedSchemaRegistryClient(schemaR
egistryConfig))
        using (var consumer = new ConsumerBuilder<string, GenericRecord>(con
sumerConfig)
            .SetValueDeserializer(new AvroDeserializer<GenericRecord>(schem
aRegistry).AsSyncOverAsync())
            .Build())
        {
            consumer.Subscribe("topic");
            var start = DateTime.Now;
            long messageCounter = 0;
            try
            {
                while (!(Console.KeyAvailable && Console.ReadKey().Key == Co
nsoleKey.Q))
                {
                    var result = consumer.Consume(TimeSpan.FromMillisecon
ds(100));
                    if (result == null) { continue; }
                    if (result.IsPartitionEOF) { break; }

                    ++messageCounter;
                    if (messageCounter % 1024 == 0) {
                        Console.WriteLine(
                            $"Received message key: \"{result.Message.Key
}\" " +
                            $"value: < {result.Message.Value["message_id"]},
 in flight on send: {result.Message.Value["in_flight_on_send"]} >");
                    }
                }
            } catch (OperationCanceledException) {}

            consumer.Close();  // commit offset and unsubscribe

            var elapsed = DateTime.Now - start;
            Console.WriteLine("average throughput: {0:N3} msg/sec, {1} mess
ages over {2:N3} sec", messageCounter / elapsed.TotalSeconds, messageCounter
, elapsed.TotalSeconds);
        }
    }
}
```

Simple .Net producer using Schema Registry

Review this example code to learn how you can create a simple .NET producer which is using Schema Registry.

```
internal class Program
{
    private static int _inFlight;
    private static long _delivered;

    private static void HandleDeliveryResult(int msgid, DeliveryResult<stri
ng, GenericRecord> deliveryResult)
    {
```

```
        if (msgid % 1024 == 0)  // writing to console on every message would
 be a bottleneck
        {
            Console.WriteLine($"Delivered '{deliveryResult.Value}' to '{deli
veryResult.TopicPartitionOffset}', in flight on delivery confirmation: {_inF
light}");
        }
        --_inFlight;
        ++_delivered;
    }

    private static void Main(string[] args)
    {
        var schema = (Avro.RecordSchema) Avro.Schema.Parse(@"{
                ""type"": ""record"",
                ""name"": ""ExampleRecord"",
                ""namespace"": ""ExampleSchema"",
                ""fields"": [
                    {
                        ""name"": ""message_id"",
                        ""type"": ""int""
                    },
                    {
                        ""name"": ""in_flight_on_send"",
                        ""type"": ""int""
                    }
                ]
            }");

        var schemaRegistryConfig = new SchemaRegistryConfig { Url = "http://
schemareg:7788/api/v1" };
        var producerConfig = new ProducerConfig { BootstrapServers = "brok
er:9092" };

        using (var schemaRegistry = new CachedSchemaRegistryClient(schemaR
egistryConfig))
        using (var producer = new ProducerBuilder<string, GenericRecord>(pro
ducerConfig)
            .SetValueSerializer(new AvroSerializer<GenericRecord>(schemaReg
istry, new AvroSerializerConfig
            {
                ProtocolVersion = AvroSerializerConfig.ProtocolVersions.Sc
hemaMetadataIdAsVersionProtocol
            }))
            .Build())
        {
            int msgCounter = 0;
            var start = DateTime.Now;
            while (!(Console.KeyAvailable && Console.ReadKey().Key == Cons
oleKey.Q || (DateTime.Now - start).TotalSeconds >= 60))
            {
                int msgid = ++msgCounter;

                var record = new GenericRecord(schema);
                record.Add("message_id", msgid);
                record.Add("in_flight_on_send", ++_inFlight);

                try
                {
                    producer.ProduceAsync("topic", new Message<string, Gene
ricRecord>
                        {
                            Key = msgid.ToString(),
                            Value = record
```

```
                        }
                    ).ContinueWith(task => HandleDeliveryResult(msgid, tas
k.Result));
                }
                catch (ProduceException<string, string> e)
                {
                    Console.WriteLine($"Produce failed: {e.Error.Reason}");
                }
            }
            producer.Flush();
            Console.WriteLine("average throughput: {0:N3} msg/sec", _delive
red / (DateTime.Now - start).TotalSeconds);
        }
    }
}
```

Supported values for ProtocolVersion

The Schema Registry client consists of two major components: The REST client and the serializers (SerDes). You are supposed to define the record schemas, and to set the respective SerDes to Kafka client. When the first message with the given schema definition is produced, the REST client stores it on the Schema Registry server. The server might have multiple ways to reference these schemas from the client. Each message produced to Kafka is transformed to contain an ID to the schema stored on the Schema Registry server, so when the other side consumes these messages, it is able to download these schemas by the IDs. The format of these IDs are denoted by the first byte of the message, which is called the Magic Byte or Protocol ID.

Note: Only Confluent protocol was supported until 1.8.2. Since 1.9.0, consumers can also handle non-confluent protocols.

ProtocolVersion	Integer value	Specification
AvroSerializerConfig.ProtocolVersions.ConfluentProtocol	0	https://registry-project.readthedocs.io/en/latest/serdes.html#confluent-protocol
AvroSerializerConfig.ProtocolVersions.SchemaMetadataIdAsVersionProtocol	1	https://registry-project.readthedocs.io/en/latest/serdes.html#schema-metadata-id-and-version-protocol
AvroSerializerConfig.ProtocolVersions.SchemaVersionIdAsLongProtocol	2	https://registry-project.readthedocs.io/en/latest/serdes.html#schema-version-id-as-long-protocol
AvroSerializerConfig.ProtocolVersions.SchemaVersionIdAsIntProtocol	3	https://registry-project.readthedocs.io/en/latest/serdes.html#schema-version-id-as-int-protocol

.NET client security examples

Review the .NET client security examples to learn what configuration properties you have to set when connecting to secured or unsecured clusters.

The following code snippets give you a few simple examples on what configuration properties you need to set for your Kafka clients when connecting them to either secured or unsecured Kafka clusters. Use the following examples as a starting point and make changes as necessary.

Unsecure

```
var producerConfig = new ProducerConfig
{
   BootstrapServers = "***BROKER HOST***:***PORT***"
};
```

SSL

Note: The Kafka .NET client does not use a truststore (.jks) like the Java clients. Instead, the Kafka .NET client requires configuring the location of the broker certificate authority (CA) certificate in .pem format by setting the SslCaLocation property.

```
var producerConfig = new ProducerConfig
{
   BootstrapServers = "***BROKER HOST***:***PORT***",
   SecurityProtocol = SecurityProtocol.Ssl,
   SslCaLocation = "***PATH TO BROKER CA CERTIFICATE***",
   SslKeystoreLocation = "***KEYSTORE LOCATION***", // client’s keystore,
 pkcs12 format
   SslKeystorePassword = "***KEYSTORE PASSWORD***” // client’s keystore p
assword
 };
```

PLAIN (LDAP, PAM, and others)

Note: Credentials are sent across the wire, Cloudera recommends that you always use channel encryption.

```
var producerConfig = new ProducerConfig
{
   BootstrapServers = "***BROKER HOST***:***PORT***",
   SecurityProtocol = SecurityProtocol.SaslSsl,
   SaslMechanism = SaslMechanism.Plain,
   SslCaLocation = "***PATH TO BROKER CA CERTIFICATE***",
   SaslUsername = "***USERNAME***",
   SaslPassword = "***PASSWORD***"
};
```

Kerberos

Important:

Windows does not support Kerberos principals or keytabs for authentication, it only supports Kerberos authentication through SSPI, The native windows authentication interface, which uses the currently logged on user as its credentials.

```
var producerConfig = new ProducerConfig
{
   BootstrapServers = "***BROKER HOST***:***PORT***",
   SecurityProtocol = SecurityProtocol.Sasl,
   SaslMechanism = SaslMechanism.Gssapi,
   SaslKerberosServiceName = "kafka"
};
```

Schema Registry

Review the .NET client security examples to learn what configuration properties you have to set when connecting to secured or unsecured clusters.

The following code snippets give you a few simple examples on what configuration properties you need to set for your Schema Registry clients when connecting them to either secured or unsecured Schema Registry. Use the following examples as a starting point and make changes as necessary.

Unsecure

```
var schemaRegistryConfig = new SchemaRegistryConfig {
    Url = "http://***SCHEMA REGISTRY HOST***:***PORT***/api/v1"
```

```
};
```

SSL/Mutual TLS

```
var schemaRegistryConfig = new SchemaRegistryConfig {
    Url = "https://***SCHEMA REGISTRY HOST***:***PORT***/api/v1",
    SslCaLocation = "***PATH TO SCHEMA REGISTRY CA CERTIFICATE**
*",
    SslKeystoreLocation = "***KEYSTORE LOCATION***",
    SslKeystorePassword = "***KEYSTORE PASSWORD***"
};
```

PLAIN/Knox authentication

```
var schemaRegistryConfig = new SchemaRegistryConfig {
    Url = "https://***SCHEMA REGISTRY HOST***:***PORT***/api/v1",
    SslCaLocation = "***PATH TO SCHEMA REGISTRY CA CERTIFICATE**
*",
    BasicAuthCredentialsSource = AuthCredentialsSource.UserInfo,
    BasicAuthUserInfo = "***USERNAME***:***PASSWORD***"
};
```

Note: Kerberos authentication is currently not supported by the .NET Schema Registry client.

Kafka Streams

Learn more about Kafka Streams.

You can access the Apache Kafka website for information on how to use Kafka Streams.

- Read the Kafka Streams Introduction for an overview of the feature and an introductory video.
- Get familiar with Kafka Streams Core Concepts.
- Understand Kafka Streams Architecture.
- Access the Quick Start documentation to run a demonstration Kafka Streams Application.
- Use the Tutorial to write your first Kafka Streams Application.

Kafka public APIs

Learn what is and what is not considered a Kafka public API by Cloudera.

What is a Public API

The following parts of Apache Kafka in CDH are considered as public APIs:

- Kafka wire protocol format: the format itself might change, but brokers will be able to use the old format as long as documentation and upgrade instructions are followed properly.
- Binary log format: the format itself might change, but brokers will be able to use the old format as long as documentation and upgrade instructions are followed properly.
- Interfaces and classes in the following packages:
  - org/apache/kafka/common/serialization
  - org/apache/kafka/common/errors
  - org/apache/kafka/clients/producer
  - org/apache/kafka/clients/consumer

- Command-line admin tools: arguments, except ZooKeeper related options, that are subject to change and/or removal.
- HttpMetricsReporter: existing fields will stay backward compatible, but new fields may be introduced. The only public API of HttpMetricsReporter is the /api/metrics REST endpoint. For a list of supported metrics, see Kafka Metrics.
- Properties, excluding their default values
- Config file content and format, and the effect of configuration attributes
- Endpoints

What is NOT a public API

There are structures that third parties might regard as an interface but Cloudera Kafka distributions do not consider them public APIs. In general, any API that is not listed as public in the What is a Public API on page 27 section should be considered private, and client code should not rely on behavior/data content or format. Some examples are:

- Data structures in ZooKeeper: the content and format what Kafka stores in ZooKeeper are internal implementation details.
- Authorizer interface: the only supported authorizer in Cloudera is the Ranger one.
- AdminClient: it is a new and rapidly evolving part of Kafka, so Cloudera can't provide the same guarantees as for other interfaces.
- Interfaces marked with the @Evolving or @Unstable annotations in the Kafka source code
- Index files generated by Kafka
- Application log file content and format (for example what Log4J/SLF4J/… produces)
- Any classes used for testing
- Relying on transitive dependencies: any dependency pulled in by Kafka
- Any other interfaces not listed above
- Anything that Cloudera does not support, even if it fits the definition of a public API

Related Information

Kafka Metrics

Recommendations for using the producer and consumer APIs

Review the following collection of code snippets and recommendations regarding the use of the producer and consumer APIs to learn how you can develop better Kafka clients.

After reviewing the basic examples of a producer and consumer, prototyping your own designs shouldn't be too difficult. However, your code will likely undergo several iterations that improve on scalability, debuggability, robustness, and maintainability.

This topic presents recommendations in the form of code snippets that illustrate some of the important ways to use the producer and consumer APIs.

In addition to the recommendations presented here, it is highly recommended that you also review the Javadoc for producers and consumers or any other API documentation available for the client library you are using. These will have additional details about Kafka client programming .

API documentation is known to be dense with information. They assume you have sufficient background in reliable computing, networking, multithreading, and distributed systems to use the APIs correctly. While the following recommendations point out many caveats in using the client APIs, the API documentation (and ultimately the source code) provides a more detailed explanation.

Reuse your Producer/Consumer object

In these examples, the consumer constructor should be called once and the poll method called within a loop. If this object is not reused, then a new connection to the broker is opened with each new consumer object created.

For Java
Recommended

```
KafkaConsumer<String, String> consumer = new KafkaConsumer<>(pro
ps);

while (true) {
  ConsumerRecords<String, String> records = consumer.poll(100);
}
```

Not Recommended

```
while (true) {
  KafkaConsumer<String, String> consumer = new KafkaConsumer<>(pr
ops);
  ConsumerRecords<String, String> records = consumer.poll(100);
}
```

For C#
Recommended

```
var consumer = new ConsumerBuilder<string, string>(config).Build
();
while (true)
{
    var result = consumer.Consume();
}
```

Not recommended

```
while (true)
{
    var consumer = new ConsumerBuilder<string, string>(config).Bu
ild();
    var result = consumer.Consume();
}
```

Similarly, it is recommended that you use one consumer and/or producer object per thread. Creating more objects opens multiple ports per broker connection. Overusing ephemeral ports can cause performance issues.

In addition, Cloudera recommends to set and use a fixed client ID for producers and consumers when they are connecting to the brokers. If this is not done, Kafka will assign a new client id every time a new connection is established, which can severely increase resource utilization (memory) on the broker side.

Each KafkaConsumer object requires frequent polling

Any consumer connected to a partition will time out if no polling is done before the maximum poll interval is reached.

In the example below, the call to myDataProcess.doStuff(records) can cause infrequent polling. This could be due to a combination of reasons. This could be due to a combination of reasons:

- Being a blocking method call.
- Doing work on a remote machine.
- Having highly variable processing time.

- Saving to storage that has highly variable I/O throughput.

In such cases, consider having another thread or process doing the actual work and making the handoff as lightweight as possible.

For Java
Example: poll() gets KafkaException due to session timeout

```
while (true) {
  KafkaConsumer<String, String> consumer = new KafkaConsumer<>(pr
ops);
  ConsumerRecords<String, String> records = consumer.poll(100);
  // the call below should return quickly in all cases
  myDataProcess.doStuff(records);
}
```

For C#
Example: Consume() gets KafkaException due to session timeout

```
while (true) {
  var consumer = new ConsumerBuilder<string, string>(config).Buil
d();
  var result = consumer.Consume(100);
  // the call below should return quickly in all cases
  myDataProcess.DoStuff(records);
}
```

Catch all exceptions when polling

For Java

From the poll() Javadoc page, you can see that the poll() method throws several exceptions. If the catch statements are not complete, then any uncaught exception will end up in the finally statement calling KafkaConsumer#close(). This will not be the desired behavior in many cases.

```
while (true) {
  try {
    ConsumerRecords<String, String> records = consumer.poll(100);
  } catch (Exception e) {
    e.printStackTrace();
  } finally {
    consumer.close();
  }
```

For C#

In the .NET client some overloads can only throw one, but another overload can throw multiple exceptions. Make sure that all exceptions are caught.

```
while (true)
{
  try
  {
    var result = consumer.Consume();
  }
  catch (Exception e)
  {
    Console.WriteLine($"Exception caught:\n {e.StackTrace}");
  }
  finally
```

```
  {
    consumer.Close();
  }
}
```

Callback#onCompletion() should always exit without errors

Note: This recommendation is for the Java client only.

The interface org.apache.kafka.clients.producer.Callback (Javadoc) is used to define a class that can be used upon completion of a KafkaProducer#send() call. It allows for tracking, clean up, or other administrative code to be called. An example of unintended usage is to call KafkaProducer#send() within the Callback#onCompletion() method, essentially mimicking a retry. Because the onCompletion() method is always expected to return cleanly and the send () method makes no such guarantees, calling send() within the callback could end up hanging the code in case of network or broker issues.

Check your API usage against the latest API

The documentation for the latest upstream release of Apache Kafka indicates if there have been any changes to how the APIs are used (setup, read, write). Reviewing the latest information could help avoid upgrade-related changes to your producer or consumer.

Some examples from past versions include:

Old Class or Package	New Class or Package
kafka.producer.ProducerConfig	java.util.Properties
kafka.javaapi.*	kafka.api.*
kafka.producer.KeyedMessage	kafka.clients.producer.ProducerRecord

Hidden Dependency on Network Availability

Network dependency is one of the more subtle issues. Given the consumer dependencies on Ranger and Zookeeper, having a combination of frequent or prolonged DNS or network outages can also cause various session timeouts to occur. Such timeouts will force partition rebalancing on the brokers, which will worsen general Kafka reliability.

Should these issues be common in your network, you may need to have a less straightforward design that can handle such reliability issues outside of the Kafka client.

Related Information

KafkaConsumer Javadoc

KafkaProducer Javadoc

Cloudera Runtime 7.3.1

Apache Kafka KRaft

Date published: 2023-06-27
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

KRaft setup.....4

Extracting KRaft metadata.....4

Securing KRaft.....5

Configuring TLS/SSL for KRaft Controllers.....5

KRaft Ranger authorization.....6

KRaft setup

Learn how you can set up Kafka KRaft in Cloudera Public Cloud

Note: Kafka KRaft is available in this version of Cloudera but is not ready for production deployment. Cloudera encourages you to explore this technical preview feature in non-production environments and provide feedback on your experiences through the Cloudera Community Forums. For more information regarding KRaft limitations and unsupported features, see Known Issues in Apache Kafka.

Kafka KRaft in Cloudera is implemented in the form of a Kafka service role. The role is called KRaft Controller. In Cloudera Public Cloud, KRaft Controller roles can be deployed with the Streams Messaging Light Duty, Heavy Duty, and High Availability cluster definitions available in the Cloudera Data Hub service.

Each of the definitions include an optional KRaft Nodes host group. The KRaft Controller roles are deployed on the nodes of this host group. If you want to deploy a Streams Messaging cluster that uses KRaft for metadata management, you must provision your cluster with at least a single KRaft node (three is recommended). KRaft nodes are also scalable after the cluster is provisioned.

For more information regarding the Streams Messaging cluster definitions, scaling, cluster deployment with Cloudera Data Hub, as well as KRaft, see the *Related Information*.

Related Information

Kafka KRaft Overview

Setting up your Streams Messaging cluster

Scaling KRaft

Streams Messaging cluster layout

Extracting KRaft metadata

Learn how to extract Kafka metadata from the __cluster_metadata topic. Metadata extracted from this topic can be used for debugging and troubleshooting issues with a Kafka deployment running in KRaft mode.

About this task

Note: Kafka KRaft is available in this version of Cloudera but is not ready for production deployment. Cloudera encourages you to explore this technical preview feature in non-production environments and provide feedback on your experiences through the Cloudera Community Forums. For more information regarding KRaft limitations and unsupported features, see Known Issues in Apache Kafka.

When Kafka is running in KRaft mode, metadata describing the state of the Kafka cluster is stored in the __cluster_metadata topic. This topic can be found in the /var/local/kraft/data directory on each KRaft Controller service role host.

In case you encounter any issues when running your deployment in KRaft mode, generally the first step is to print the contents of the __cluster_metadata topic. Reviewing the contents of the topic can help in identifying the issues with the cluster.

The contents of the __cluster_metadata topic can be printed using the kafka-dump-log command with the --cluster-metadata-decoder option.

Procedure

1. Log in to one of your cluster hosts that has a KRaft service role (KRaft Controller) deployed on it.

2. Run the kafka-dump-log command with the --cluster-metadata-decoder option. For example:

```
kafka-dump-log --cluster-metadata-decoder --files /var/local/kraft/data/
__cluster_metadata-0/00000000000000000000.log
```

Securing KRaft

Learn about KRaft security and security configuration in Cloudera.

Note: Kafka KRaft is available in this version of Cloudera but is not ready for production deployment. Cloudera encourages you to explore this technical preview feature in non-production environments and provide feedback on your experiences through the Cloudera Community Forums. For more information regarding KRaft limitations and unsupported features, see Known Issues in Apache Kafka.

When you deploy Kafka in KRaft mode a set of specialized broker roles, KRaft Controller roles, are deployed on your cluster. KRaft Controllers communicate with brokers to serve their requests and to manage Kafka's metadata. The connection between controllers and brokers can be secured using TLS/SSL encryption, TLS/SSL authentication, and/or Kerberos authentication.

By default KRaft Controllers inherit the security configuration of the parent Kafka service. For example, if TLS/SSL is enabled for Kafka, then Cloudera Manager automatically enables TLS/SSL for the KRaft Controllers in the cluster. As a result, if you configure security for the Kafka service, no additional configuration is required to secure KRaft Controllers.

However, if required, some security properties related to encryption and authentication can be configured separately for KRaft Controllers.

- TLS/SSL encryption and authentication

  TLS/SSL configuration can be configured separately as the KRaft Controller role has its own set of TLS/SSL properties. You can enable or disable TLS/SSL as well as configure the key and truststore that the KRaft Controller roles use. For more information see, Configuring TLS/SSL for KRaft Controllers on page 5.
- Kerberos authentication

  Kerberos cannot be enabled or disabled separately for KRaft Controllers. The default Kerberos principal for KRaft controllers, the kraft user, can be changed using the Role-Specific Kerberos Principal Kafka service property.

  Important: Cloudera Manager configures Cloudera services to use the default Kerberos principal names. Cloudera recommends that you do not change the default Kerberos principal names. If it is unavoidable to do so, contact Cloudera Professional Services because it requires extensive additional custom configuration.

Ranger authorization

In addition to encryption and authentication, the default principal that KRaft Controllers run as is integrated with Ranger. For more information on the default policies set up for the user, see KRaft Ranger authorization on page 6.

Configuring TLS/SSL for KRaft Controllers

Learn how to configure TLS/SSL for KRaft Controllers.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.

3. Find and configure the following properties based on your cluster and requirements.

Table 1: KRaft TLS/SSL configuration properties

Cloudera Manager Property	Description
Enable TLS/SSL for Kraft Controller ssl_enabled	Encrypt communication between clients and KRaft Controller using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).
KRaft Controller TLS/SSL Server JKS Keystore File Location ssl_server_keystore_location	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when KRaft Controller is acting as a TLS/SSL server. The keystore must be in the format specified in Administration Settings Java Keystore Type .
KRaft Controller TLS/SSL Server JKS Keystore File Password ssl_server_keystore_password	The password for the KRaft Controller keystore file.
KRaft Controller TLS/SSL Server JKS Keystore Key Password ssl_server_keystore_keypassword	The password that protects the private key contained in the keystore used when KRaft Controller is acting as a TLS/SSL server.
KRaft Controller TLS/SSL Trust Store File ssl_client_truststore_location	The location on disk of the trust store, in .jks format, used to confirm the authenticity of TLS/SSL servers that KRaft Controller might connect to. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
KRaft Controller TLS/SSL Trust Store Password ssl_client_truststore_password	The password for the KRaft Controller TLS/SSL Trust Store File. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.
SSL Client Authentication ssl.client.auth	Client authentication mode for SSL connections. This configuration has three valid values, required, requested, and none. If set to required, client authentication is required. If set to requested, client authentication is requested and clients without certificates can still connect. If set to none, which is the default value, no client authentication is required.

4. Click Save Changes.
5. Restart the Kafka service.

What to do next

TLS/SSL encryption is configured for the KRaft Controller role.

KRaft Ranger authorization

Learn how KRaft integrates with Ranger as well as the default policies and permissions set up for KRaft.

Note: If Ranger authorization is enabled, Kafka still connects to ZooKeeper for auditing. As a result, Kafka's JAAS configuration includes a client entry for ZooKeeper. Additionally, the -Dzookeeper.sasl.client.user name=*[**ZOOKEEPER PRINCIPAL SHORTNAME***]* system property is set for the process. This is the result of Ranger's dependency on ZooKeeper. Even though Ranger makes this connection, Kafka does not require or use ZooKeeper for metadata management if it is running in KRaft mode.

KRaft in Cloudera uses the `KafkaRangerAuthorizer` to authorize requests coming from other entities. In KRaft mode, Kafka brokers forward requests to the controllers and the controllers authorize these requests.

Kraft Controllers run as the kraft user. By default, the Kafka resource-based service in Ranger includes a kraft in ternal - topic policy. This policy grants all permission on the __cluster_metadata topic for the kraft user as well as Describe, Describe Configs, and Consume permissions for the kafka user (default user for brokers). By default, other users do not have access to the __cluster_metadata topic.

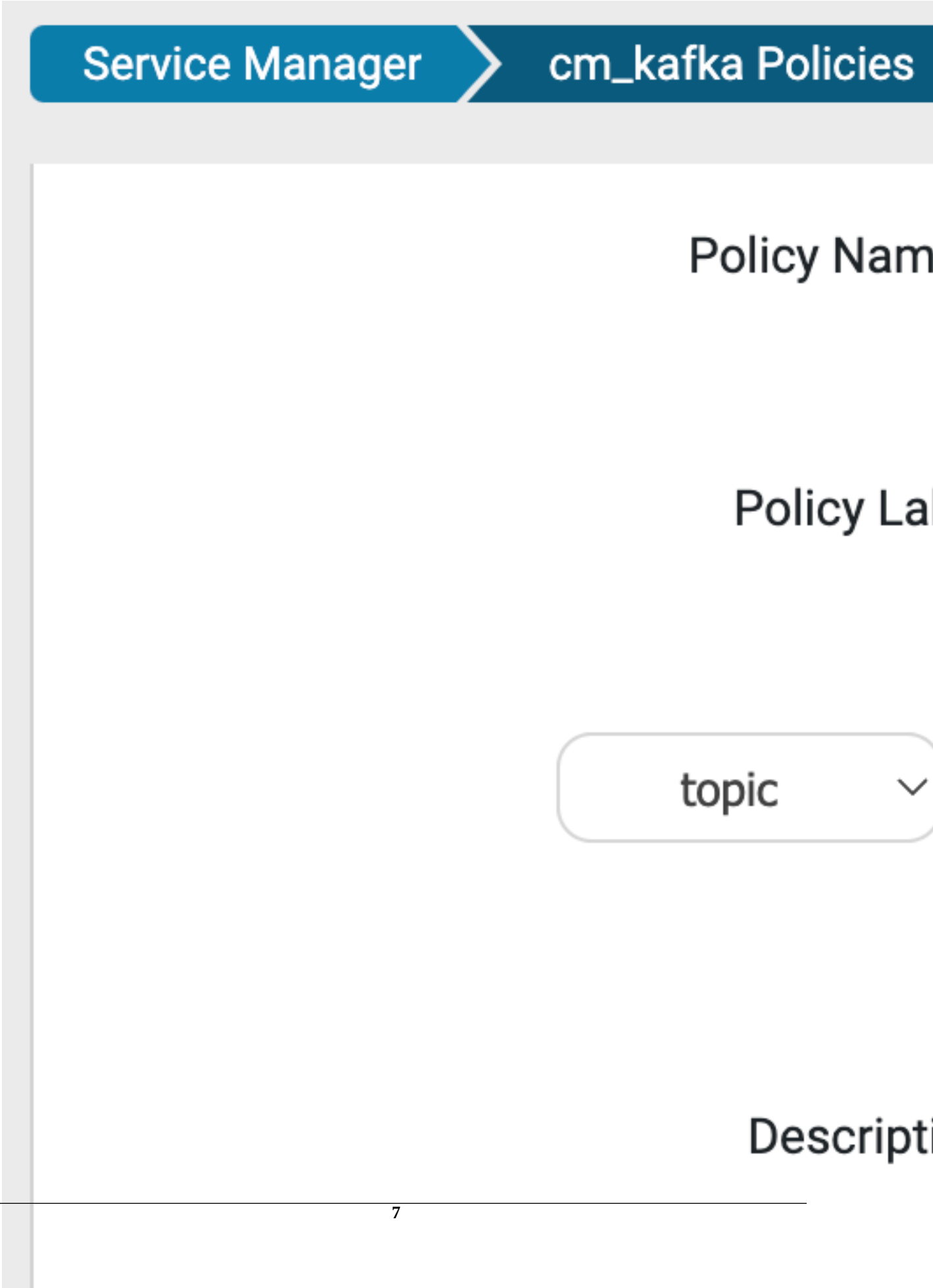
Service Manager
cm_kafka Policies
Policy Nam
Policy La
topic
Descripti

In addition, the kraft user is added to all default Kafka policies that grant all permissions on Kafka resources.

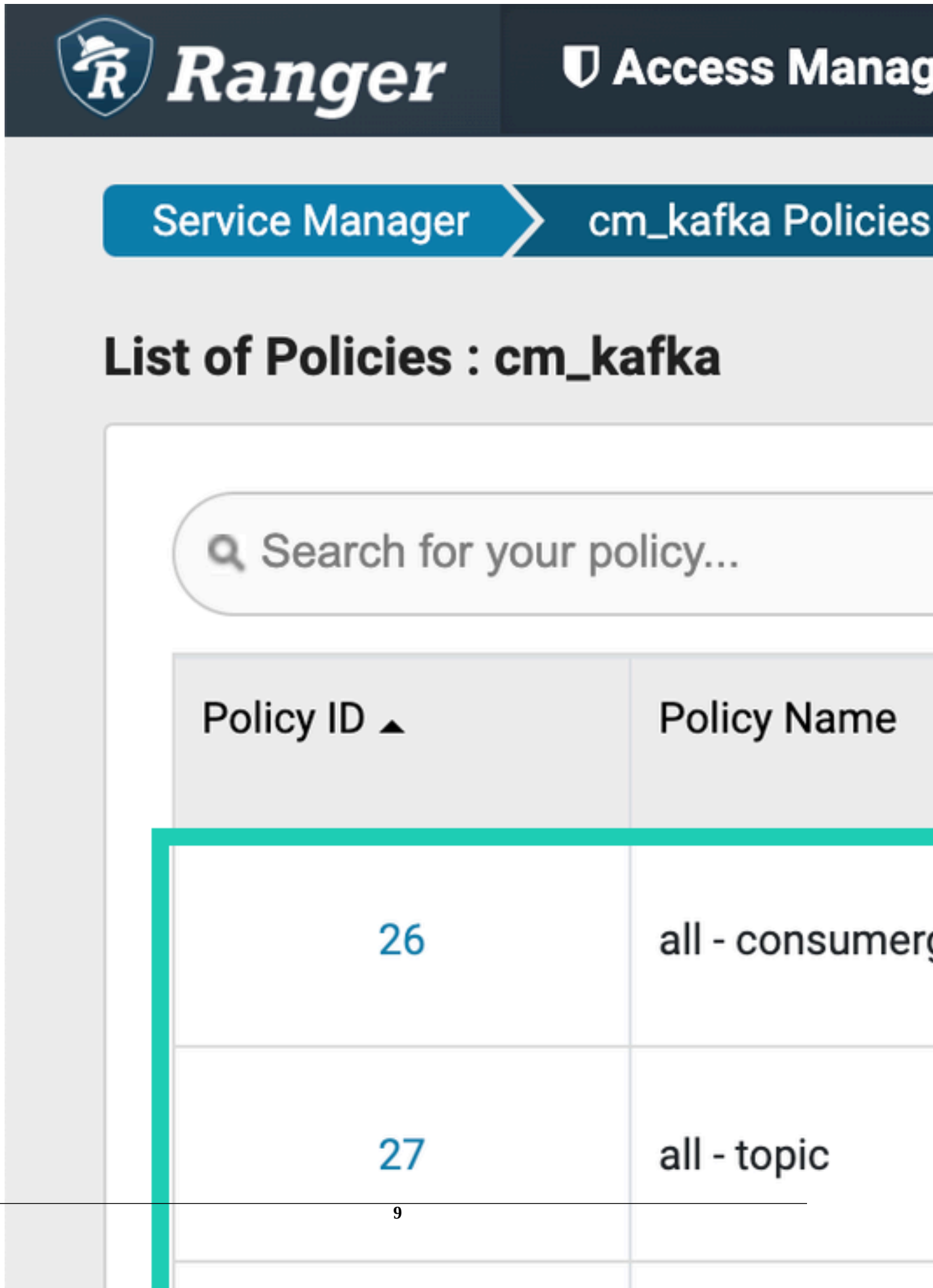
Ranger
Access Manag
Service Manager
cm_kafka Policies
List of Policies : cm_kafka
Search for your policy...
Policy ID
Policy Name
26
all - consumerg
27
all - topic

Cloudera Runtime 7.3.1

Managing Apache Kafka

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Management basics.....5
Broker log management.....5
Record management.....5
Broker garbage log collection and log rotation.....6

Client and broker compatibility across Kafka versions.....6

Managing topics across multiple Kafka clusters.....7
Set up MirrorMaker in Cloudera Manager.....8
Settings to avoid data loss.....9

Broker migration.....9
Migrate brokers by modifying broker IDs in meta.properties.....10
Use rsync to copy files from one broker to another.....10

Disk management.....11
Monitoring.....11
Handling disk failures.....11
Disk Replacement.....12
Disk Removal.....12
Reassigning replicas between log directories.....13
Retrieving log directory replica assignment information.....13

Metrics.....13
Building Cloudera Manager charts with Kafka metrics.....14
Essential metrics to monitor.....15

Command Line Tools.....17
Unsupported command line tools.....17
kafka-topics.....18
kafka-cluster.....19
kafka-configs.....19
kafka-console-producer.....20
kafka-console-consumer.....21
kafka-consumer-groups.....22
kafka-features.....24
kafka-reassign-partitions.....24
Tool usage.....25
Reassignment examples.....27
kafka-log-dirs.....28
zookeeper-security-migration.....29
kafka-delegation-tokens.....30
kafka-*-perf-test.....31

Configuring log levels for command line tools.....31
Understanding the kafka-run-class Bash Script.....32

Management basics

Broker log management

Learn more about the configuration properties related to broker log management, as well as the ways you can monitor log directory space.

Kafka brokers save their data as log segments in a directory. The logs are rotated depending on the size and time settings.

The most common log retention settings to adjust for your cluster are shown below. These are accessible in Cloudera Manager via the Kafka Configuration tab.

- log.dirs: The location for the Kafka data (that is, topic directories and log segments).
- log.retention.{ms|minutes|hours}: The retention period for the entire log. Any older log segments are removed.
- log.retention.bytes: The retention size for the entire log.

In addition, there are many more configuration properties available for fine-tuning broker log management. For more information look for the following properties in theBroker Configs section in the upstream Apache Kafka documentation.

- log.dirs
- log.flush.*
- log.retention.*
- log.roll.*
- log.segment.*

You can monitor the log directory space of the Kafka Brokers on the Log Directory Free Capacity chart on the Kafka service Status page.

This chart relies on the Broker Log Directory Free Capacity Check trigger. The check is triggered if the capacity of any log directory falls below 10%. The trigger is automatically created for all newly deployed Kafka services. For existing services, you need to create the trigger using the Create Kafka Log Directory Free Capacity Check action on the Kafka service Actions page following an upgrade.

Related Information

Broker Configs

Record management

Learn how you can manage records.

There are two pieces to record management, log segments and log cleaner.

As part of the general data storage, Kafka rolls logs periodically based on size or time limits. Once either limit is reached, a new log segment is created with the all new data being placed there, while older log segments should generally no longer change. This helps limit the risk of data loss or corruption to a single segment instead of the entire log.

- log.roll.{ms|hours}: The time period for each log segment. Once the current segment is older than this value, it goes through log segment rotation.
- log.segment.bytes: The maximum size for a single log segment.

There is an alternative to simply removing log segments for a partition. There is another feature based on the log cleaner. When the log cleaner is enabled, individual records in older log segments can be managed differently:

- log.cleaner.enable: This is a global setting in Kafka to enable the log cleaner.

- cleanup.policy: This is a per-topic property that is usually set at topic creation time. There are two valid values for this property, delete and compact.
- log.cleaner.min.compaction.lag.ms: This is the retention period for the “head” of the log. Only records outside of this retention period will be compacted by the log cleaner.

The compact policy, also called log compaction, assumes that the "most recent Kafka record is important." Some examples include tracking a current email address or tracking a current mailing address. With log compaction, older records with the same key are removed from a log segment and the latest one is kept. This effectively removes some offsets from the partition.

Broker garbage log collection and log rotation

Learn more about broker garbage collection and how garbage log rotation can be configured.

About this task

Both broker JVM garbage collection and JVM garbage log rotation is enabled by default in Cloudera Kafka delivered with Cloudera Runtime. Garbage collection logs are written in the agent process directory by default.

Example path:

```
/run/cloudera-scm-agent/process/99-kafka-KAFKA_BROKER/kafkaServer-gc.log
```

Changing the default directory of garbage collection logs is currently not supported. However, you can configure properties related garbage log rotation.

Garbage log rotation properties can be configured with the Kafka Broker Environment Advanced Configuration Snippet (Safety Valve) property.

Procedure

1. In Cloudera Manager, go to the Kafka service and click Configuration.
2. Find the Kafka Broker Environment Advanced Configuration Snippet (Safety Valve) property.
3. Add the following line to the property:

   Modify the values of as required.

   ```
   KAFKA_GC_LOG_OPTS="-XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX
   :GCLogFileSize=100M"
   ```

   The flags used are as follows:

   - +UseGCLogFileRotation: Enables garbage log rotation.
   - -XX:NumberOfGCLogFiles: Specifies the number of files to use when rotating logs.
   - -XX:GCLogFileSize: Specifies the size when the log will be rotated.
4. Click on Save Changes.
5. Restart the Kafka service to apply the changes.

Results

Kafka garbage log rotation is configured.

Client and broker compatibility across Kafka versions

An overview on client and broker version compatibility.

Maintaining compatibility across different Kafka clients and brokers is a common issue. Mismatches among client and broker versions can occur as part of any of the following scenarios:

- Upgrading your Kafka cluster without upgrading your Kafka clients.
- Using a third-party application that produces to or consumes from your Kafka cluster.
- Having a client program communicating with two or more Kafka clusters (such as consuming from one cluster and producing to a different cluster).
- Using Spark as a Kafka consumer.

In these cases, it is important to understand client/broker compatibility across Kafka versions. Here are general rules that apply:

- Changes in either the major part or the minor part of the Kafka version *MAJOR.MINOR* determines whether the client and broker are fully compatible. If there are differences, some features might not be available.
- Differences among the maintenance versions are not considered when determining compatibility.
- All clients and brokers later than version 1.0.0 are considered compatible. Additionally, all client and broker versions shipped with Cloudera Runtime are also considered compatible. If you are using an older client version with a newer broker version, some limitations on the available features may apply, but the client and broker will be able to communicate with each other.

Managing topics across multiple Kafka clusters

You may have more than one Kafka cluster to support:

- Geographic distribution
- Disaster recovery
- Organizational requirements

You can distribute messages across multiple clusters. It can be handy to have a copy of one or more topics from other Kafka clusters available to a client on one cluster. Mirror Maker is a tool that comes bundled with Kafka to help automate the process of mirroring or publishing messages from one cluster to another. "Mirroring" occurs between clusters where "replication" distributes message within a cluster.

Figure 1: Mirror Maker Makes Topics Available on Multiple Clusters

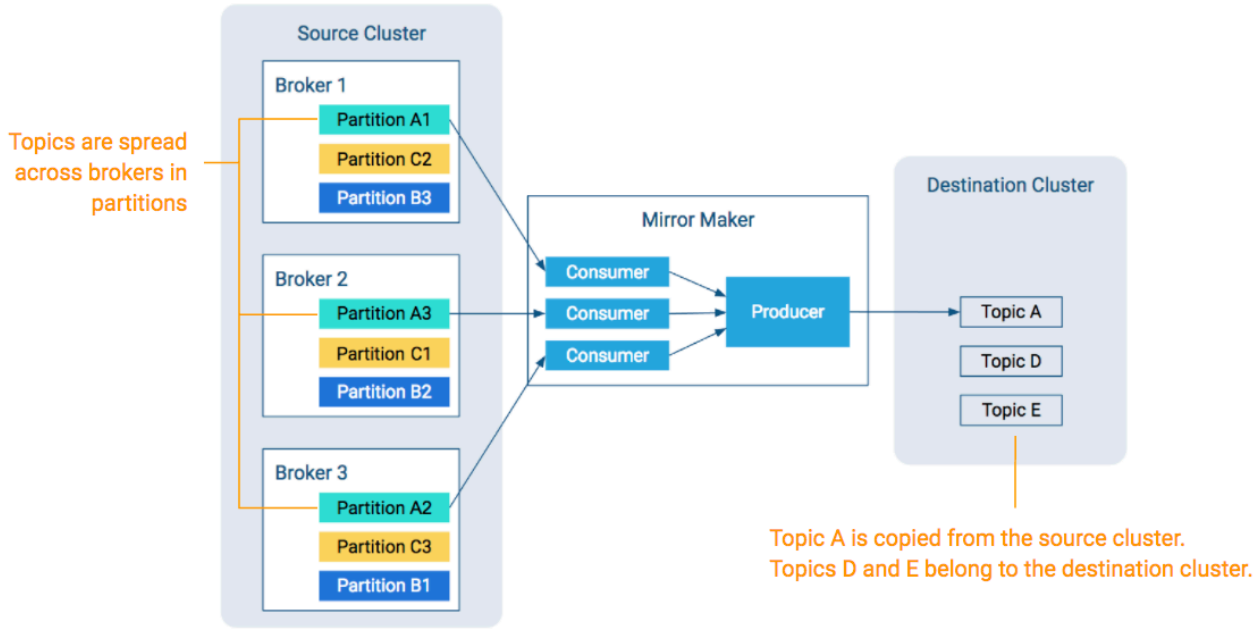

While the diagram shows copying to one topic, Mirror Maker's main mode of operation is running continuously, copying one or more topics from the source cluster to the destination cluster.

Keep in mind the following design notes when configuring Mirror Maker:

- Mirror Maker runs as a single process.
- Mirror Maker can run with multiple consumers that read from multiple partitions in the source cluster.
- Mirror Maker uses a single producer to copy messages to the matching topic in the destination cluster.

Set up MirrorMaker in Cloudera Manager

Learn how to set up MirrorMaker in Cloudera Manager

Before you begin

Consider the following before setting up MirrorMaker for Kafka:

Consumer/Producer Compatibility

The Mirror Maker consumer needs to be client compatible with the source cluster. The Mirror Maker producer needs to be client compatible with the destination cluster. See Client and Broker Compatibility Across Kafka Versions for further details on compatibility.

Topic Differences between Clusters

Because messages are copied from the source cluster to the destination cluster—potentially through many consumers funneling into a single producer—there is no guarantee of having identical offsets or timestamps between the two clusters. In addition, as these copies occur over the network, there can be some mismatching due to retries or dropped messages.

Optimize Mirror Maker Producer Location

Because Mirror Maker uses a single producer and since producers typically have more difficulty with high latency and/or unreliable connections, it is better to have the producer run "closer" to the destination cluster, meaning in the same data center or on the same rack.

Destination Cluster Configuration

Before starting Mirror Maker, make sure that the destination cluster is configured correctly:

- Make sure there is sufficient disk space to copy the topic from the source cluster to the destination cluster.
- Make sure the topic exists in the destination cluster or enable topic auto creation by setting the auto.create.topics.enable property to true. The latter can be done in Cloudera Manager by going to KafkaConfiguration.

Kerberos and Mirror Maker

As mentioned earlier, Mirror Maker runs as a single process. The resulting consumers and producers rely on a single configuration setup. Mirror Maker requires that the source cluster and the destination cluster belong to the same Kerberos realm.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Choose Action Add Role Instances.
3. Under Kafka Mirror Maker, click Select hosts.
4. Select the host where Mirror Maker will run and click Continue.
5. Fill in the Destination Broker List and Source Broker List with your source and destination Kafka clusters.

   Use host name, IP address, or fully qualified domain name.
6. Fill out the Topic Whitelist.

   The allowlist (whitelist) is required.

7. Fill out the TLS/SSL sections if security needs to be enabled.
8. Start the Mirror Maker instance.

Results

MirrorMaker is configured and started.

Related Information

Client and broker compatibility across Kafka versions

Settings to avoid data loss

An overview of default configuration properties to avoid data loss.

If for some reason the producer cannot deliver messages that have been consumed and committed by the consumer, it is possible for a MirrorMaker process to lose data. The following collection of properties are set by default and help with preventing data loss when replicating data. In addition, note that MirrorMaker starts correctly if you enter the numeric values in the configuration snippet (rather than using "max integer" for retries and "max long" for max.bloc k.ms).

Producer settings

- acks=all
- retries=2147483647
- max.block.ms=9223372036854775807

Consumer setting

- auto.commit.enable=false

MirrorMaker setting

- abort.on.send.failure=true

Broker migration

Learn more about the options you have when migrating brokers and copying data between brokers.

In case of catastrophic hardware failure brokers can be moved to a new host in a Kafka cluster. There multiple methods with which you can move brokers between clusters:

Using kafka-reassign-partitions tool

This method involves more manual work to modify JSON, but does not require manual edits to configuration files.

Modify the broker IDs in meta.properties

This method involves less manual work, but requires modifying an internal configuration file.

Using rsync

You can use rsync to copy over all data from an old broker to a new broker, preserving modification times and permissions.

Continue reading for instructions on how to move brokers by modifying broker IDs or using rsync. For instructions on how to migrate brokers with kafka-reassign-partitions, see the kafka-reassign-partitions tool description. For specific instructions on how to migrate existing Kafka partitions to JBOD configured disks, see JBOD disk migration.

Related Information

kafka-reassign-partitions

JBOD disk migration

Migrate brokers by modifying broker IDs in meta.properties

Learn how to migrate brokers between hosts in a Kafka cluster by modifying broker IDs in meta.properties.

About this task

In case of catastrophic hardware failure brokers can be moved to a new host in a Kafka cluster. The following steps walk you through broker migration by modifying broker IDs in meta.properties. Compared to migration with the help of the kafka-reassign-partitions tool, this method involves less manual work, but requires modifying an internal configuration file.

Important: Data intensive administration operations such as rebalancing partitions, adding a broker, removing a broker, or bootstrapping a new machine can cause significant additional load on the cluster. Therefore, Cloudera highly recommends that you migrate one broker at a time and migrate brokers when there is minimal load on the cluster

Before you begin

- Verify that the cluster is healthy.
- Verify that all replicas are in sync.

Procedure

1. Start up the new broker as a member of the old cluster.

   This creates files in the data directory.
2. Stop both the new broker and the old broker that it is replacing.
3. Change broker.id of the new broker to the broker.id of the old one both in Cloudera Manager and in *DATA DIRECTORY*/meta.properties.
4. Optional: Run rsync to copy files from one broker to another.
5. Start up the new broker.

   When started the new broker re-replicates data from the other nodes.

Results

The Kafka broker is migrated to a new host.

Use rsync to copy files from one broker to another

Learn how to use rsync to copy over all data from an old broker to a new broker.

About this task

You can run rsync command to copy over all data from an old broker to a new broker, preserving modification times and permissions. Using rsync allows you to avoid having to re-replicate the data from the leader.

Before you begin

- Ensure that the disk structures match between the two brokers, or verify the meta.properties file between the source and destination brokers (because there is one meta.properties file for each data directory).

Procedure

Run the following command on destination broker:

```
rsync -avz
        SRC_BROKER:SRC_DATA_DIR
```

```
        DEST_DATA_DIR
```

If you plan to change the broker ID, edit *DEST_DATA_DIR*/meta.properties.

Results

Data from the source broker is copied over to the destination broker.

Disk management

Monitoring

Recommendations on what to monitor on clusters.

Cloudera recommends that administrators continuously monitor the following on a cluster:

Replication Status

Monitor replication status using Cloudera Manager Health Tests. Cloudera Manager automatically and continuously monitors both the OfflineLogDirectoryCount and OfflineReplicaCount metrics. Alters are raised when failures are detected.

Disk Capacity

Monitor free space on mounted disks and open file descriptors. Reassign partitions or move log files around if necessary. For more information, see the kafka-reassign-partitions tool description.

Related Information

Cloudera Manager Health Tests

kafka-reassign-partitions

Handling disk failures

An overview on how to handle disk failures.

Cloudera Manager has built in monitoring functionalities that automatically trigger alerts when disk failures are detected. When a log directory fails, Kafka also detects the failure and takes the partitions stored in that directory offline.

Important: If there are no healthy log directories present in the system, the broker stops working.

The cause of disk failures can be analyzed with the help of the kafka-log-dirs tool, or by reviewing the error messages of KafkaStorageException entries in the Kafka broker log file. To access the log file go to InstancesLog FilesRole Log File.

In case of a disk failure, a Kafka administrator can carry out either of the following actions. The action taken depends on the failure type and system environment:

- Replace the faulty disk with a new one.
- Remove the disk and redistribute data across remaining disks to restore the desired replication factor.

Note: Disk replacement and disk removal both require stopping the broker. Therefore, Cloudera recommends that you perform these actions during a maintenance window.

Related Information

kafka-log-dirs

Disk Replacement

Learn how to replace a disk.

About this task

In case of a disk failure, a Kafka administrator can replace the faulty disk with a new one.

Procedure

1. Stop the broker that has a faulty disk:
   a) In Cloudera Manager, go to the Kafka service, select Instances and select the broker.
   b) Go to ActionsGracefully stop this Kafka Broker.
2. Replace the disk.
3. Mount the disk.
4. Set up the directory structure on the new disk the same way as it was set up on the previous disk.

   Note: You can find the directory paths for the old disk in the Data Directories property of the broker.

5. Start the broker:
   a) In Cloudera Manager go to the Kafka service, selectInstances and select the broker.
   b) Go to ActionsStart this Kafka Broker.

Results

The disk is replaced. The Kafka broker re-creates topic partitions in the same directory by replicating data from other brokers.

Disk Removal

Learn how to remove a disk from the configuration.

About this task

In case of a disk failure, a Kafka administrator can remove the disk and redistribute data across remaining disks to restore the desired replication factor.

Procedure

1. Stop the broker that has a faulty disk:
   a) In Cloudera Manager, go to the Kafka service, select Instances and select the broker.
   b) Go to ActionsGracefully stop this Kafka Broker.
2. Remove the log directories on the faulty disk from the broker:
   a) Go to Configuration and find the Data Directories property.
   b) Remove the affected log directories with the Remove button.
   c) Enter a Reason for change, and then click Save Changes to commit the changes.
3. Start the broker:
   a) In Cloudera Manager, go to the Kafka service, select Instances and select the broker.
   b) Go to ActionsStart this Kafka Broker.

Results

The disk is removed from the configuration. The Kafka broker redistributes data across the cluster.

Reassigning replicas between log directories

Learn about replica reassignment between log directories.

Reassigning replicas between log directories can prove useful when you have multiple disks available, but one or more of them is nearing capacity. Moving a replica from one disk to another ensures that the service will not go down due to disks reaching capacity. To balance storage loads, the Kafka administrator has to continuously monitor the system and reassign replicas between log directories on the same broker or across different brokers. These actions can be carried out with the kafka-reassign-partitions tool.

For more information on tool usage, see the kafka-reassign-paritions tool description.

Related Information

kafka-reassign-partitions

Retrieving log directory replica assignment information

Learn how log directory replica assignment information can be retrieved.

To optimize replica assignment across log directories, the list of partitions per log directory and the size of each partition is required. This information can be exposed with the kafka-log-dirs tool.

For more information on tool usage, see the kafka-log-dirs tool description.

Related Information

kafka-log-dirs

Metrics

Learn about Kafka metrics and how to view them.

Kafka uses Yammer metrics to record internal performance measurements. The metrics are exposed via Java Management Extensions (JMX) and can be read with a JMX console.

Metrics Categories

There are metrics available in the various components of Kafka. In addition, there are some metrics specific to how Cloudera Manager and Kafka interact. This table has pointers to both the Apache Kafka metrics names and the Cloudera Manager metric names.

Table 1: Metrics by Category

Category	Cloudera Manager Metrics Doc	Apache Kafka Metrics Doc
Cloudera Manager Kafka Service	Base Metrics	
Broker	Broker Metrics, Broker Topic Metrics, Replica Metrics, Broker Topic Partition Metrics	Broker
Common Producer/Consumer		Client Client-to-Broker
Producer	Producer Metrics	Producer Producer Sender

Category	Cloudera Manager Metrics Doc	Apache Kafka Metrics Doc
Consumer	Consumer Group Metrics	Consumer Group Consumer Fetch
Mirror Maker	Mirror Maker Metrics	Same as Producer or Consumer tables

Note the following about Cloudera Manager metrics:

Broker Topic Partitions Metrics:

Required for Streams Messaging Manager. These metrics have to be specifically enabled within the host's configuration.

Producer Metrics:

Required for Streams Messaging Manager. These metrics have to be specifically enabled within Kafka's configuration and are different from the metrics available on client instances.

Consumer Groups Metrics

Generated by and required for Streams Messaging Manager. These metrics are different from the metrics available on client instances.

Viewing Metrics

Cloudera Manager records most of these metrics and makes them available via Chart Builder.

Because Cloudera Manager cannot track metrics on any clients (that is, producer or consumer), you may wish to use an alternative JMX console program to check metrics. There are several JMX console options, for example:

- JConsole, which comes bundled with the JDK.
- VisualVM, with the MBeans plugin.

Related Information

JConsole Documentation

VisualVM Homepage

Building Cloudera Manager charts with Kafka metrics

A collection query examples to build Kafka metrics charts within Cloudera Manager.

Cloudera Manager enables you to build charts based on Kafka specific metrics. To access the chart builder go to ChartsChart Builderin Cloudera Manager. The following are a few specific examples of queries for cloudera metrics:

Controllers across all brokers

This chart shows the active controller across all brokers. It is useful for checking active controller status (should be one at any given time, transitions should be fast).

```
SELECT
    kafka_active_controller
WHERE
    roleType=KAFKA_BROKER
```

Network idle rate

>Chart showing the network processor idle rate across all brokers. If idle time is always zero, then probably the num.network.threads property may need to be increased.

```
SELECT
    kafka_network_processor_avg_idle_rate
WHERE
```

```
    roleType=KAFKA_BROKER
```

Partitions per broker

Chart showing the number of partitions per broker. It is useful for detecting partition imbalances early.

```
SELECT
    kafka_partitions
WHERE
    roleType=KAFKA_BROKER
```

Partition activity

Chart tracking partition activity on a single broker.

```
SELECT
    kafka_partitions, kafka_under_replicated_partitions
WHERE
    hostname=host1.domain.com
```

Mirror Maker activity

Chart for tracking Mirror Maker behavior. Since Mirror Maker has one or more consumers and a single producer, most consumer or metrics should be usable with this query.

```
SELECT
    PRODUCER OR CONSUMER METRIC
WHERE
    roleType=KAFKA_MIRROR_MAKER
```

Related Information

Charting Time-Series Data

Essential metrics to monitor

Cloudera Manager collects a high number of performance metrics for the Kafka services running on your clusters. Certain metrics should be monitored in any Kafka deployment as they can help you to improve the stability and performance of your Kafka deployment.

The following tables collect the Kafka broker metrics that Cloudera recommends you to monitor in any Kafka deployment. For more information on metrics, including a full list of Kafka metrics, see Cloudera Manager Metrics.

Table 2: ZooKeeper connectivity metrics

Metric Name	Description	Unit	Importance	Parents	Version Availability
kafka_zookeeper_expires_rate	Measures the session expires per second.	Expires per second	High	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_zookeeper_request_latency_avg	Request latency between the broker and Zookeeper.	ms	High	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7

Table 3: Active controller metrics

Metric Name	Description	Unit	Importance	Parents	Version Availability
kafka_active_controller	Shows the number of active controllers at a given time. Ideally it should be 1.	Number of controllers	High	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7

Table 4: Network metrics

Metric Name	Description	Unit	Importance	Parents	Version Availability
kafka_network_processor_avg_idle	The average free capacity of the network processors. Should be > 0.3.	Percentage of free capacity	High	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_request_queue_size	Size of the request queue in Kafka.	Message count	Medium	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_response_queue_size	Size of the response queue in Kafka.	Message count	Medium	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_messages_received_rate	Number of messages written to topic on this broker.	Messages per second	Medium	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7

Table 5: Disk utilization metrics

Metric Name	Description	Unit	Importance	Parents	Version Availability
kafka_produce_local_time_rate	Local Time spent in responding to Produce requests.	Requests per second	Low	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_log_flush_rate	Rate of flushing Kafka logs to disk.	Flushes per second	Low	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_request_handler_avg_idle_rate	The average free capacity of the request handler. Should be > 0.3.	Percentage of free capacity	High	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7

Note: While the disk utilization metrics do not measure disk performance directly, they can point to problems with the disks.

Table 6: Kafka metrics

Metric Name	Description	Unit	Importance	Parents	Version Availability
kafka_broker_state	The state the broker is in. 0 = NotRunning, 1 = Starting, 2 = RecoveringFromUncleanShutdown, 3 = RunningAsBroker, 4 = RunningAsController, 6 = PendingControlledShutdown, 7 = BrokerShuttingDown	Discrete states	Medium	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_jvm_gc_runs_rate	Number of garbage collector runs performed on this broker.	Events per second	Medium	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7

Metric Name	Description	Unit	Importance	Parents	Version Availability
kafka_isr_expands_rate	ISR expands per second.	Events per second	Medium	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_isr_shrinks_rate	ISR shrinks per second.	Events per second	Medium	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_max_replication_lag	Maximum replication lag on the broker, across all fetchers, topics, and partitions.	Messages	Medium	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_offline_partitions	Number of offline partitions.	Partition count	High	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_under_min_isr_partition_count	Count of partitions with less than the configured minimum in-sync replicas available.	Partition count	High	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7
kafka_under_replicated_partitions	Count of partitions with unavailable replicas.	Partition count	Low	cluster, kafka, rack	CDH 5, CDH 6, Cloudera 7

Related Information

Cloudera Manager Metrics

Command Line Tools

Kafka command line tools overview.

In some situations, it is convenient to use the command line tools available in Kafka to administer your cluster. However, it is important to note that not all tools available for Kafka are supported by Cloudera. Moreover, certain administration tasks can be carried out more easily and conveniently using Cloudera Manager.

When administering Kafka with command line tools, be aware of the following:

- Use Cloudera Manager to start and stop Kafka and Zookeeper services. Do not use the kafka-server-start, kafka-server-stop, zookeeper-server-start, or zookeeper-server-stop commands.
- For a parcel installation, all Kafka command line tools are located in /opt/cloudera/parcels/KAFKA/lib/kafka/bin/. For a package installation, all such tools can be found in /usr/bin/.
- Ensure that the JAVA_HOME environment variable is set to your JDK installation directory before using the command-line tools. For example:

```
export JAVA_HOME=/usr/java/jdk1.8.0_144-cloudera
```

- Using any Zookeeper command manually can be very difficult to get right when it comes to interaction with Kafka. Cloudera recommends that you avoid doing any write operations or ACL modifications in Zookeeper.

Note: Output examples in the command line tool descriptions are cleaned and formatted for easier readability.

Unsupported command line tools

Kafka command line tools not supported by Cloudera.

The following tools can be found as part of the Kafka distribution, but their use is generally discouraged for various reasons as documented here.

Tool	Notes
connect-mirror-maker	Use Streams Replication Manager instead.
connect-distributed connect-standalone	Use Cloudera Manager to deploy and manage Kafka Connect workers. In Cloudera, Kafka Connect workers are represented by Kafka Connect roles, which can be deployed under a Kafka service.
kafka-acls	Cloudera recommends using Ranger for authorization instead.
kafka-broker-api-versions	Primarily useful for Client-to-Broker protocol related development.
kafka-configs	Use Cloudera Manager to adjust any broker or security properties instead of the kafka#configs tool. This tool should only be used to modify topic properties.
kafka-delete-records	Cloudera does not recommend using this tool in.
kafka-features	Used to describe and modify currently active features in the cluster. Limited support is provided. Only use this tool to describe currently active features.
kafka-mirror-maker	Use Cloudera Manager to deploy Mirror Maker instances.
kafka-replay-log-producer	Can be used to “rename” a topic.
kafka-replica-verification	Validates that all replicas for a set of topics have the same data. This tool is a “heavy duty” version of the ISR column of kafka-topics tool.
kafka-server-start kafka-server-stop	Use Cloudera Manager to manage any Kafka host.
kafka-storage	Used to generate a universally unique identifier (UUID) for the cluster and to format log directories when Kafka is running in KRaft mode. Both UUID generation and log directory formatting are done automatically by Cloudera Manager when you deploy the Kafka service for the first time as well as when you add new Kafka role instances to the cluster. The generated UUID is stored internally by Cloudera Manager and cannot be replaced. Additionally, adding and manually formatting new log directories in KRaft not supported.
kafka-verifiable-consumer kafka-verifiable-producer	These scripts are intended for system testing.
zookeeper-server-start zookeeper-server-stop	Use Cloudera Manager to manage any Zookeeper host.
zookeeper-shell	Limit usage of this script to reading information from Zookeeper.

kafka-topics

Learn more about the kafka-topics tool.

Use the kafka-topics tool to generate a snapshot of topics in the Kafka cluster.

```
kafka-topics --bootstrap-server [BROKER_HOST]:[PORT] --describe
```

```
Topic: topic-a1      PartitionCount:3      ReplicationFactor:3      Configs:
         Topic: topic-a1      Partition: 0      Leader: 64      Replicas: 6
4,62,63      Isr: 64,62,63
         Topic: topic-a1      Partition: 1      Leader: 62      Replicas:
 62,63,64      Isr: 62,63,64
         Topic: topic-a1      Partition: 2      Leader: 63      Replicas: 6
3,64,62      Isr: 63,64,62
```

```
Topic: topic-a2      PartitionCount:1      ReplicationFactor:3      Configs:
         Topic: topic-a2      Partition: 0      Leader: 64      Replicas: 64
,62,63      Isr: 64,62,63
```

The output lists each topic and basic partition information. Note the following about the output:

- Partition count: The more partitions, the higher the possible parallelism among consumers and producers.
- Replication factor: Shows 1 for no redundancy and higher for more redundancy.
- Replicas and in-sync replicas (ISR): Shows which broker ID's have the partitions and which replicas are current.

There are situations where this tool shows an invalid value for the leader broker ID or the number of ISRs is fewer than the number of replicas. In those cases, there may be something wrong with those specific topics.

It is possible to change topic configuration properties using this tool. Increasing the partition count, the replication factor or both is not recommended.

kafka-cluster

Learn about the kafka-cluster CLI tool, which you can use to print the ID of a Kafka cluster.

Use the kafka-cluster tool to retrieve the ID of the Kafka cluster. The ID of a Kafka cluster is automatically generated by Cloudera Manager when you first start your Kafka service. The ID you retrieve is used when carrying out management actions for your Kafka service using other CLI tools.

```
kafka-cluster cluster-id --bootstrap-server [***HOST***]:[***PORT***] --c
onfig [***PROPERTY FILE***]
```

- Replace *[***HOST***]:[***PORT***]* with the host and port of a broker.
- Replace *[***PROPERTY FILE***]* with the location of a configuration file that contains all necessary security properties that are required to establish a connection with the Kafka service. This option is only required if your Kafka service is secured.

Output example:

```
Cluster ID: 8TMmf99zQR-KIy5bF1lmZQ
```

kafka-configs

In an environment managed by Cloudera Manager the kafka-configs tool can be used to set, describe, or delete topic properties.

Important: Cloudera does not recommend that you use the kafka-configs tool to configure broker properties. This is because the tool bypasses Cloudera Manager safety checks. Use Cloudera Manager instead if you want to configure your brokers. Only use this tool to configure topic properties.

Setting topic properties

You can set a topic property using the --alter option together with the --add-config option. For example:

```
kafka-configs --bootstrap-server [HOST:PORT] --entity-type
 topics --entity-name [TOPIC] --alter --add-config [PROPERTY
 NAME]=[VALUE]
```

Describing topic properties

You can list the configuration properties of a topic with the --describe option. For example:

```
kafka-configs --bootstrap-server [HOST:PORT] --entity-type topics
 --entity-name [TOPIC] --describe
```

Deleting topic properties

You can delete a topic property using the --alter option together with the --delete-config option. For example:

```
kafka-configs --bootstrap-server [HOST:PORT] --entity-type topics
 --entity-name [TOPIC] --alter --delete-config [PROPERTY_NAME]
```

Related Information

Topic-Level Configs

kafka-console-producer

Learn how you can use the kafka-console-producer tool to produce messages to a topic.

This tool is used to write messages to a topic in a text based format.

The following examples demonstrate the basic usage of the tool. In addition to reviewing these examples, you can also use the --help option to see a list of all available options.

Produce messages to a topic

To start producing message to a topic, you need to run the tool and specify a server as well as a topic.

```
kafka-console-producer --bootstrap-server [HOST1:PORT1] --topic [TOPIC]
```

Start typing messages once the tool is running.

```
>MY FIRST MESSAGE
>MY SECOND MESSAGE
```

Alternatively, you can also produce the contents of a file to a topic.

```
cat [FILE] | kafka-console-producer --bootstrap-server [HOST1:PORT1] --t
opic [TOPIC]
```

Produce messages to a topic in a secure cluster

In order to use the tool on a secure cluster, additional client configuration is required. You do this by creating a .properties file that contains the necessary configurations to run the tool on a secure cluster. Which properties are configured in this file depends on the security configuration of your cluster. See the client configuration topics in Securing Apache Kafka to learn more about which exact properties you need to use for your security configuration. Once the file is created, you can use --producer.config to pass the file to the tool.

Example .properties file:

```
security.protocol = SASL_SSL
sasl.mechanism=GSSAPI
sasl.kerberos.service.name = kafka
ssl.truststore.location = /var/private/ssl/kafka.client.truststore.jks
ssl.truststore.password = test1234
sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required useTi
cketCache=true;
```

This example shows what properties you have to set when both Kerberos and TLS/SSL are configured.

Do the following to run the tool in a secure cluster:

1. Create a .properties file.

   Use the example above. Make changes as necessary.
2. Run the tool with the --producer.config option.

```
kafka-console-producer --bootstrap-server [HOST1:PORT1] --topic [TOPIC] --
producer.config client.properties
```

Define a key-value delimiter

It is possible to define a key-value delimiter for the given producer instance. The delimiter can vary each time you run the tool. For example, if you run the tool with the delimiter set to - and then a second time using :, Kafka will know how to store the data. The first term is always stored as the key, the second as the value. As a result, when the data written this way is consumed, the keys and values are consumed irrespective of what delimiter was used when the records were written to the topic.

To produce messages with key-value delimiters, you need to set two properties, parse.key and key.seperator. Both properties can be set with the --properties option.

```
kafka-console-producer --bootstrap-server [HOST1:PORT1] --topic [TOPIC] --p
roperty parse.key=true --property key.separator=":"
```

Configure retry backoff

Before each retry, the producer refreshes the metadata of the relevant topics. You can configure the amount of time the producer waits before refreshing the metadata with the --retry-backoff-ms option.

```
kafka-console-producer --bootstrap-server [HOST1:PORT1] --topic [TOPIC] --r
etry-backoff-ms 1000
```

Related Information

Configure Kafka clients for TLS/SSL encryption

Enable Kerberos authentication

Client authentication using delegation tokens

Configure Kafka clients for LDAP authentication

Configure Kafka clients for PAM authentication

kafka-console-consumer

Learn how to use the kafka-console-consumer tool.

The kafka-console-consumer tool can be useful in a couple of ways:

- Acting as an independent consumer of particular topics. This can be useful to compare results against a consumer program that you've written.
- To test general topic consumption without the need to write any consumer code.

Examples of usage:

```
kafka-console-consumer --bootstrap-server [BROKER1],[BROKER2]... --to
pic [TOPIC] --from-beginning
[RECORD-EARLIEST-OFFSET]
[RECORD-EARLIEST-OFFSET+1]
```

Note the following about the tool:

- This tool prints all records and keeps outputting as more records are written to the topic.
- If the kafka-console-consumer tool is given no flags, it displays the full help message.
- In older versions of Kafka, it may have been necessary to use the --new-consumer flag. As of Apache Kafka version 0.10.2, this is no longer necessary.

kafka-consumer-groups

Learn how to use the kafka-consumer-groups tool.

The kafka-consumer-groups tool can be used to list all consumer groups, describe a consumer group, delete consumer group info, or reset consumer group offsets. The following topic gives an overview on how to describe or reset consumer group offsets.

Describe Offsets

This tool is primarily used for describing consumer groups and debugging any consumer offset issues, like consumer lag. The output from the tool shows the log and consumer offsets for each partition connected to the consumer group that is being described. You can see at a glance which consumers are current with their partition and which ones are lagging. From there, you can determine which partitions (and likely the corresponding brokers) are slow.

Using the tool on secure and unsecure clusters differs slightly. On secure clusters, you have to use the command-conf ig option together with an appropriate property file.

Describing offsets on an unsecure cluster

Use the following command to describe offsets committed to Kafka:

```
kafka-consumer-groups --bootstrap-server [HOST]:9092 --describe
 --group [CONSUMER GROUP]
```

Output Example:

```
GROUP   TOPIC  PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG
 OWNER
group1   topic1     0          1               3               2
      test-consumer-group_postamac.local-1456198719410-29ccd54f-0
```

Describing offsets on a secure cluster

In order to describe offsets on a secure Kafka cluster, the consumer-groups tool has to be run with the command-config option. The command-config option specifies the property file that contains the necessary configurations to run the tool on a secure cluster. Which properties are configured in this file depends on the security configuration of your cluster.

Example client.properties file:

```
exclude.internal.topics=false
security.protocol = SASL_SSL
sasl.kerberos.service.name = kafka
ssl.truststore.location = /var/private/ssl/kafka.client.trusts
tore.jks
ssl.truststore.password = test1234
```

This example shows what properties you have to set when both Kerberos and TLS/SSL are configured.

To describe offsets do the following:

1. Pass the jaas.conf file location as a JVM parameter.

```
export KAFKA_OPTS='-Djava.security.auth.login.config=[PATH TO
 JAAS.CONF]
```

2. Create a client.properties file.

   Use the example above. Make changes as necessary.
3. Run the tool with the command-config option.

```
kafka-consumer-groups --bootstrap-server [HOST]:9093 --desc
ribe --command-config CLIENT.PROPERTIES --group [CONSUMER
 GROUP]
```

Resetting Offsets

You can use the --reset-offset option to reset the offsets of a consumer group to a particular value. The tool can be used to reset all offsets on all topics. However, this is something you probably won't ever want to do. Therefore, it is highly recommended that you exercise caution when resetting offsets.

To reset offsets you need to define a scope, an execution option, and a scenario.

Scope

There are two supported scopes:

- --topic: Restricts the change to a specific topic or a specific set of partitions within a topic.
- --all-topics: Executes the change for all topics.

Note: Choosing a scope is mandatory for all scenarios except for the --from-file scenario where specifying a scope is not required.

Execution option

There are three execution options:

- --dry-run --reset-offsets: Default option. Displays which offsets will be reset if the process is executed.
- --execute --reset-offsets : Executes the process.
- --export --reset-offsets : Exports results in CSV format.

Scenarios

There are a number of supported scenarios. Scenarios control what value the offsets are reset to. Specifying a scenario is mandatory.

- --to-datetime
- --by-period
- --to-earliest
- --to-latest
- --shift-by
- --from-file
- --to-current

Example:

```
kafka-consumer-groups --dry-run --reset-offsets --bootstrap-serv
er [HOST]:9092 --group [CONSUMER GROUP] --topic [TOPIC] --to-current
```

kafka-features

Learn about the kafka-features CLI tool, which you can use to describe currently active features.

Important: Support for this tool is limited in this version of Cloudera Runtime. Do not use the upgrade, downgrade, or disable, options to manage Kafka features in your cluster. Only use this tool to describe currently active features.

Describing currently active features

The kafka-features describe command returns the current features active in the cluster. The command returns a list of feature flags with their supported minimum and maximum versions as well as the finalized version, which is the version currently configured.

```
kafka-features --bootstrap-server [***HOST***]:[***PORT**] --command-config
 describe
```

- Replace *[***HOST***]:[***PORT**]* with the host and port of a broker.
- Replace *[***PROPERTY FILE***]* with the location of a configuration file that contains all necessary security properties that are required to establish a connection with the Kafka service. This option is only required if your Kafka service is secured.

kafka-reassign-partitions

Learn about the kafka-reassign-partitions tool, which is used to reassign partitions between brokers in a cluster.

This tool provides substantial control over partitions in a Kafka cluster. It is mainly used to balance storage loads across brokers through the following reassignment actions:

- Change the ordering of the partition assignment list. Used to control leader imbalances between brokers.
- Reassign partitions from one broker to another. Used to expand existing clusters.
- Increase the replication factor of partitions. Used to expand existing clusters.
- Reassign partitions between log directories on the same broker. Used to resolve storage load imbalance among available disks in the broker.
- Reassign partitions between log directories across multiple brokers. Used to resolve storage load imbalance across multiple brokers.

The tool uses two JSON files for input. Both of these are created by the user. The two files are the following:

- Topics-to-Move JSON
- Reassignment Configuration JSON

Topics-to-Move JSON

This JSON file specifies the topics that you want to reassign. This a simple file that tells the kafk a-reassign-partitions tool which partitions it should look at when generating a proposal for the reassignment configuration. The user has to create the topics-to-move JSON file from scratch.

The format of the file is the following:

```
{"topics":  [{"topic": "mytopic1"},
             {"topic": "mytopic2"}],
 "version":1
}
```

Reassignment Configuration JSON

This JSON file is a configuration file that contains the parameters used in the reassignment process. This file is created by the user, however, a proposal for its contents is generated by the tool. When the kafka-reasssign-partitions tool is executed with the --generate option, it generates a proposed

configuration which can be fine-tuned and saved as a JSON file. The file created this way is the reassignment configuration JSON. To generate a proposal, the tool requires a topics-to-move file as input.

The format of the file is the following:

```
{"version":1,
 "partitions":
   [{"topic":"mytopic1","partition":3,"replicas":[4,5],"log_dir
s":["any","any"]},
    {"topic":"mytopic1","partition":1,"replicas":[5,4],"log_dirs"
:["any","any"]},
    {"topic":"mytopic2","partition":2,"replicas":[6,5],"log_dir
s":["any","any"]}]
}
```

The reassignment configuration contains multiple properties that each control and specify an aspect of the configuration. The Reassignment Configuration Properties table lists each property and its description.

Table 7: Reassignment Configuration Properties

Property	Description
topic	Specifies the topic.
partition	Specifies the partition.
replicas	Specifies the brokers that the selected partition is assigned to. The brokers are listed in order, which means that the first broker in the list is always the leader for that partition. Change the order of brokers to resolve any leader balancing issues among brokers. Change the broker IDs to reassign partitions to different brokers.
log_dirs	Specifies the log directory of the brokers. The log directories are listed in the same order as the brokers. By default any is specified as the log directory, which means that the broker is free to choose where it places the replica. By default, the current broker implementation selects the log directory using a round-robin algorithm. An absolute path beginning with a / can be used to explicitly set where to store the partition replica.

Notes and Recommendations:

- Cloudera recommends that you minimize the volume of replica changes per command instance. Instead of moving 10 replicas with a single command, move two at a time in order to save cluster resources.
- This tool cannot be used to make an out-of-sync replica into the leader partition.
- Use this tool only when all brokers and topics are healthy.
- Anticipate system growth. Redistribute the load when the system is at 70% capacity. Waiting until redistribution becomes necessary due to reaching resource limits can make the redistribution process extremely time consuming.

Tool usage

Learn how to reassign partitions with the kafka-reassign-partitions-tool.

Procedure

1. Create a topics-to-move JSON file that specifies the topics you want to reassign.

   Use the following format:

```
{"topics":  [{"topic": "MYTOPIC1"},
             {"topic": "MYTOPIC2"}],
 "version":1
```

```
}
```

2. Generate the content for the reassignment configuration JSON with the following command:

```
kafka-reassign-partitions --bootstrap-server [***HOSTNAME***]:[***PORT***]
 --topics-to-move-json-file [***TOPICS TO MOVE.JSON***] --broker-li
st [***BROKER 1***], [***BROKER 2***] --generate
```

Running the command lists the distribution of partition replicas on your current brokers followed by a proposed partition reassignment configuration.

```
Current partition replica assignment
{"version":1,
 "partitions":
   [{"topic":"mytopic2","partition":1,"replicas":[2,3],"log_dirs":["any","
any"]},
    {"topic":"mytopic1","partition":0,"replicas":[1,2],"log_dirs":["any"
,"any"]},
    {"topic":"mytopic2","partition":0,"replicas":[1,2],"log_dirs":["any","
any"]},
    {"topic":"mytopic1","partition":2,"replicas":[3,1],"log_dirs":["any"
,"any"]},
    {"topic":"mytopic1","partition":1,"replicas":[2,3],"log_dirs":["any","
any"]}]
}

Proposed partition reassignment configuration

{"version":1,
 "partitions":
   [{"topic":"mytopic1","partition":0,"replicas":[4,5],"log_dirs":["any",
"any"]},
    {"topic":"mytopic1","partition":2,"replicas":[4,5],"log_dirs":["any
","any"]},
    {"topic":"mytopic2","partition":1,"replicas":[4,5],"log_dirs":["any",
"any"]},
    {"topic":"mytopic1","partition":1,"replicas":[5,4],"log_dirs":["any
","any"]},
    {"topic":"mytopic2","partition":0,"replicas":[5,4],"log_dirs":["any",
"any"]}]
}
```

In this example, the tool proposed a configuration which reassigns existing partitions on broker 1, 2, and 3 to brokers 4 and 5.

3. Copy and paste the proposed partition reassignment configuration into an empty JSON file.
4. Review, and if required, modify the suggested reassignment configuration.
5. Save the file.
6. Start the redistribution process with the following command:

```
kafka-reassign-partitions --bootstrap-server [***HOSTNAME***]:[***PORT***]
 --reassignment-json-file [***REASSIGNMENT CONFIGURATION.JSON***] --execu
te
```

The tool prints a list containing the original replica assignment and a message that reassignment has started. Example output:

```
Current partition replica assignment

{"version":1,
 "partitions":
```

```
   [{"topic":"mytopic2","partition":1,"replicas":[2,3],"log_dirs":["any",
"any"]},
    {"topic":"mytopic1","partition":0,"replicas":[1,2],"log_dirs":["any
","any"]},
    {"topic":"mytopic2","partition":0,"replicas":[1,2],"log_dirs":["any",
"any"]},
    {"topic":"mytopic1","partition":2,"replicas":[3,1],"log_dirs":["any
","any"]},
    {"topic":"mytopic1","partition":1,"replicas":[2,3],"log_dirs":["any",
"any"]}]
}

Save this to use as the --reassignment-json-file option during rollback
Successfully started reassignment of partitions.
```

7. Verify the status of the reassignment with the following command:

```
kafka-reassign-partitions --bootstrap-server [***HOSTNAME***]:[***PORT***]
 --reassignment-json-file [***REASSIGNMENT CONFIGURATION.JSON***] --verify
```

The tool prints the reassignment status of all partitions.

```
Status of partition reassignment:
Reassignment of partition mytopic2-1 completed successfully
Reassignment of partition mytopic1-0 completed successfully
Reassignment of partition mytopic2-0 completed successfully
Reassignment of partition mytopic1-2 completed successfully
Reassignment of partition mytopic1-1 completed successfully
```

Results

Partitions are reassigned.

Reassignment examples

A collection of examples that demonstrate how users can modify the proposed configuration file generated by the kafka-reassign-partitions.

There are multiple ways to modify the configuration file. The following list of examples shows how a user can modify a proposed configuration and what these changes do.

Suppose that the kafka-reassign-partitions tool generated the following proposed reassignment configuration:

```
{"version":1,
 "partitions":
   [{"topic":"mytopic1","partition":0,"replicas":[1,2],"log_dirs":["any","a
ny"]}]}
```

Now let's look at how this reassignment configuration has to be changed in different reassignment scenarios.

Reassign partitions between brokers

To reassign partitions from one broker to another, change the broker ID specified in replicas. For example:

```
{"topic":"mytopic1","partition":0,"replicas":[5,2],"log_dirs":[
"any","any"]}
```

This reassignment configuration moves partition mytopic1-0 from broker 1 to broker 5.

Reassign partitions to another log directory on the same broker

To reassign partitions between log directories on the same broker, change the appropriate any entry to an absolute path. For example:

```
{"topic":"mytopic1","partition":0,"replicas":[1,2],"log_dirs":["/
log/directory1","any"]}
```

This reassignment configuration moves partition mytopic1-0 to the /log/directory1 log directory.

Reassign partitions between log directories across multiple brokers

To reassign partitions between log directories across multiple brokers, change the broker ID specified in replicas and the appropriate any entry to an absolute path. For example:

```
{"topic":"mytopic1","partition":0,"replicas":[5,2],"log_dirs":["/
log/directory1","any"]}
```

This reassignment configuration moves partition mytopic1-0 to /log/directory1 on broker 5.

Change partition assignment order (elect a new leader)

To change the ordering of the partition assignment list, change the order of the brokers in replicas. For example:

```
{"topic":"mytopic1","partition":0,"replicas":[2,1],"log_dirs":["a
ny","any"]}
```

This reassignment configuration elects broker 2 as the new leader.

Increase the replication factor of a topic

To increase the replication factor of a topic, add one or more new brokers to replicas. For example:

```
{"topic":"mytopic1","partition":0,"replicas":[1,2,3],"log_dirs":
["any","any","any"]}
```

kafka-log-dirs

Learn how to use the kafka-log-dirs tool.

The kafka-log-dirs tool allows user to query a list of replicas per log directory on a broker. The tool provides information that is required for optimizing replica assignment across brokers.

To retrieve replica assignment information, run the following command:

```
kafka-log-dirs --describe --bootstrap-server HOSTNAME:PORT --broker-li
st BROKER 1, BROKER 2 --topic-list TOPIC 1, TOPIC 2
```

Important: On secure clusters the admin client config property file has to be specified with the --command-config option. Otherwise, the tool fails to execute.

If no topic is specified with the --topic-list option, then all topics are queried. If no broker is specified with the --br oker-list option, then all brokers are queried. If a log directory is offline, the log directory will be marked offline in the script output. Error example:

```
"error":"org.apache.kafka.common.errors.KafkaStorageException"
```

On successful execution, the tool prints a list of partitions per log directory for the specified topics and brokers. The list contains information on topic partition, size, offset lag, and reassignment state. Example output:

```
{
```

```
    "brokers": [
        {
            "broker": 86,
            "logDirs": [
                {
                    "error": null,
                    "logDir": "/var/local/kafka/data",
                    "partitions": [
                        {
                            "isFuture": false,
                            "offsetLag": 0,
                            "partition": "mytopic1-2",
                            "size": 0
                        }
                    ]
                }
            ]
        },
        ...
    ],
    "version": 1
}
```

The following table gives an overview of the information provided by the output of the kafka-log-dirs tool.

Table 8: Contents of the kafka-log-dirs Output

Property	Description
broker	Displays the ID of the broker.
error	Indicates if there is a problem with the disk that hosts the topic partition. If an error is detected, org.apache.kafka.common.errors.KafkaStorageException is displayed. If no error is detected, the value is null.
logDir	Specifies the location of the log directory. Returns an absolute path.
isfuture	The reassignment state of the partition. This property shows whether there is currently replica movement underway between the log directories.
offsetLag	Displays the offset lag of the partition.
partition	Displays the name of the partition.
size	Displays the size of the partition in bytes.

zookeeper-security-migration

Learn how to use the zookeeper-security-migration tool.

The zookeeper-security-migration tool is used in the process of restricting or unrestricting access to metadata stored in Zookeeper. When executed, the tool updates the ACLs of znodes based on the configuration specified by the user.

Important: Running the zookeeper-security-migration tool is only one of the steps required when restricting or unrestricting access. For full instructions, see Restricting Access to Kafka Metadata in Zookeeper.

Set the ACLs on all existing Zookeeper znodes to secure with the following command:

```
zookeeper-security-migration --zookeeper.connect HOSTNAME:PORT --zookeeper
.acl secure
```

Set the ACLs on all existing Zookeeper znodes to unsecure with the following command:

```
zookeeper-security-migration --zookeeper.connect HOSTNAME:PORT --zookeeper
.acl unsecure
```

Related Information

Restricting Access to Kafka Metadata in Zookeeper

kafka-delegation-tokens

Learn how to use the kafka-delegation-tokens tool.

The kafka-delegation-tokens provides the user with the functionality required for using and managing delegation tokens.

The tool can be used to issue, renew, expire, or describe delegation tokens.

Issue, and store for verification

The owner of the token is the currently authenticated principal. A renewer can be specified when requesting the token.

```
kafka-delegation-tokens --bootstrap-server HOSTNAME:PORT  --crea
te   --max-life-time-period -1 --command-config CLIENT.PROPERTIES
 --renewer-principal USER:USER1
```

Renew

Only the owner and the principals that are renewers of the delegation token can extend its validity by renewing it before it expires. A successful renewal extends the Delegation Token's expiration time for another renew-interval, until it reaches its max lifetime. Expired delegation tokens cannot be renewed. The brokers remove expired delegation tokens from the broker's cache and from Zookeeper.

Important: A delegation token cannot be used to renew a delegation token. The delegation token owner must authenticate using a different method to renew a token.

```
kafka-delegation-tokens --bootstrap-server HOSTNAME:PORT --renew
    --renew-time-period -1 --command-config CLIENT.PROPERTIES --h
mac LAYYSFMLS4BTJF+LTZ1LCHR/ZZFNA==
```

Remove

Delegation tokens are removed when they are canceled by the client or when they expire.

```
kafka-delegation-tokens --bootstrap-server HOSTNAME:PORT --expir
e   --expiry-time-period -1   --command-config CLIENT.PROPERTIES
 --hmac LAYYSFMLS4BTJF+LTZ1LCHR/ZZFNA==
```

Describe

Tokens can be described by owners, renewers or the Kafka super user.

```
kafka-delegation-tokens --bootstrap-server HOSTNAME:PORT --
describe --command-config CLIENT.PROPERTIES  --owner-princi
pal USER:USER1
```

Note: In Apache Kafka, principals that have the describe permission on the token resource can also describe the token.

kafka-*-perf-test

The kafka-*-perf-test tools can be used in several ways. In general, it is expected that these tools should be used on a test or development cluster.

The kafka-*-perf-test tools allow you to:

- Measure, read, and write throughput.
- Stress test the cluster based on specific parameters (such as message size).
- Load test for the purpose of evaluating specific metrics or determining the impact of cluster configuration changes.

The kafka-producer-perf-test script can either create a randomly generated byte record:

```
kafka-producer-perf-test --topic TOPIC --record-size SIZE_IN_BYTES
```

or randomly read from a set of provided records:

```
kafka-producer-perf-test --topic TOPIC --payload-delimiter DELIMITER --paylo
ad-file INPUT_FILE
```

where the *INPUT_FILE* is a concatenated set of pre-generated messages separated by *DELIMITER*. This script keeps producing messages or limited based on the --num-records flag.

The kafka-consumer-perf-test is:

```
kafka-consumer-perf-test --broker-list HOST1:PORT1,HOST2:PORT2,... --zooke
eper ZK1:PORT1,ZK2:PORT2,... --topic TOPIC
```

The flags of most interest for this command are:

- --group *GID*: If you run more than one instance of this test, you will want to set different ids for each instance.
- --num-fetch-threads: Defaults to 1. Increase if higher throughput testing is needed.
- --from-latest: To start consuming from the latest offset. May be needed for certain types of testing.

Configuring log levels for command line tools

Learn how to change the logging level of command line tools.

About this task

In some cases it can prove useful to change the default logging level of the command line tools. This can be done by setting the required logging level in the log4j2 properties file that the tools use.

Procedure

1. Create a copy of the log4j2.properties file that the tools use.

   You are free to change the location where the copy gets created.

   ```
   cp /etc/kafka/conf/tools-log4j2.properties /VAR/TMP
   ```

2. Open the copied file and change the value of the root.logger property as required.

   For example:

   ```
   root.logger=TRACE,console
   ```

3. Pass the location of the newly created properties file as a JVM parameter.

   You can do this by adding -Dlog4j2.configurationFile=file:*[PATH_TO_FILE]* to the KAFKA_OPTS variable. For example:

   ```
   export KAFKA_OPTS="-Dlog4j2.configurationFile=file:/VAR/TMP/TOOLS-LOG4J2.PROPERTIES"
   ```

Results

The specified logging level is configured.

Understanding the kafka-run-class Bash Script

Almost all the provided Kafka tools eventually call the kafka-run-class script. This script is generally not called directly. However, if you are proficient with bash and want to understand certain features available in all Kafka scripts as well as some potential debugging scenarios, familiarity with the kafka-run-class script can prove highly beneficial.

For example, there are some useful environment variables that affect all the command line scripts:

- KAFKA_DEBUG allows a Java debugger to attach to the JVM launched by the particular script. Setting KAFKA_DEBUG also allows some further debugging customization:
  - JAVA_DEBUG_PORT sets the JVM debugging port.
  - JAVA_DEBUG_OPTS can be used to override the default debugging arguments being passed to the JVM.
- KAFKA_HEAP_OPTS can be used to pass memory setting arguments to the JVM.
- KAFKA_JVM_PERFORMANCE_OPTS can be used to pass garbage collection flags to the JVM.

Cloudera Runtime 7.3.1

Apache Kafka Overview

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Kafka Introduction 4

Kafka Architecture 4
- Brokers 5
- Topics 6
- Records 7
- Partitions 7
- Record order and assignment 8
- Logs and log segments 9
- Kafka brokers and Zookeeper 10
- Leader positions and in-sync replicas 11

Kafka stretch clusters 14
- Kafka stretch cluster requirements 14
- Kafka stretch cluster architectures 15
- Cluster configuration for Kafka stretch clusters 17

Kafka disaster recovery 20

Kafka rack awareness 21
- Rack awareness for Kafka brokers 21
- Rack awareness for Kafka consumers 25
- Rack awareness for Kafka producers 26

Kafka KRaft [Technical Preview] 28
- The Raft algorithm 31
- KRaft metadata management 31
- KRaft metadata replication and the HWM 32
- KRaft leader changes 34
- KRaft log reconciliation 38
- KRaft broker state machines 43

Kafka FAQ 43
- Basics 43
- Use cases 45

Kafka Introduction

Apache Kafka is a high performance, highly available, and redundant streaming message platform.

Kafka functions much like a publish/subscribe messaging system, but with better throughput, built-in partitioning, replication, and fault tolerance. Kafka is a good solution for large scale message processing applications. It is often used in tandem with Apache Hadoop, and Spark Streaming.

You might think of a log as a time-sorted file or data table. Newer entries are appended to the log over time, from left to right. The log entry number is a convenient replacement for a timestamp.

Kafka integrates this unique abstraction with traditional publish/subscribe messaging concepts (such as producers, consumers, and brokers), parallelism, and enterprise features for improved performance and fault tolerance.

The original use case for Kafka was to track user behavior on websites. Site activity (page views, searches, or other actions users might take) is published to central topics, with one topic per activity type.

Kafka can be used to monitor operational data, aggregating statistics from distributed applications to produce centralized data feeds. It also works well for log aggregation, with low latency and convenient support for multiple data sources.

Kafka provides the following:

- Persistent messaging with O(1) disk structures, meaning that the execution time of Kafka's algorithms is independent of the size of the input. Execution time is constant, even with terabytes of stored messages.
- High throughput, supporting hundreds of thousands of messages per second, even with modest hardware.
- Explicit support for partitioning messages over Kafka servers. It distributes consumption over a cluster of consumer machines while maintaining the order of the message stream.
- Support for parallel data load into Hadoop.

Kafka Architecture

Learn about Kafka's architecture and how it compares to an ideal publish-subscribe system.

The ideal publish-subscribe system is straightforward: Publisher A’s messages must make their way to Subscriber A, Publisher B’s messages must make their way to Subscriber B, and so on.

Figure 1: Ideal Publish-Subscribe System

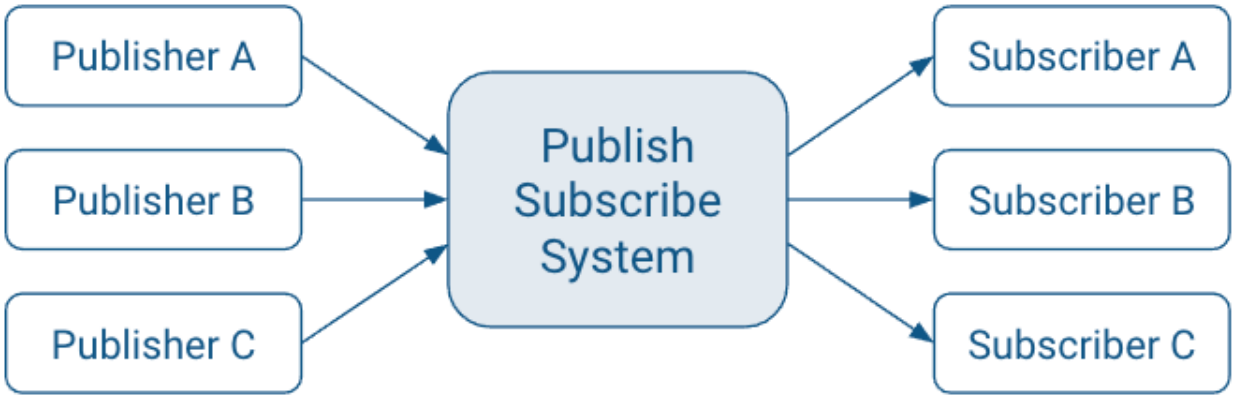

An ideal system has the benefit of:

- Unlimited Lookback. A new Subscriber A1 can read Publisher A’s stream at any point in time.

- Message Retention. No messages are lost.
- Unlimited Storage. The publish-subscribe system has unlimited storage of messages.
- No Downtime. The publish-subscribe system is never down.
- Unlimited Scaling. The publish-subscribe system can handle any number of publishers and/or subscribers with constant message delivery latency.

Kafka's architecture however deviates from this ideal system. Some of the key differences are:

- Messaging is implemented on top of a replicated, distributed commit log.
- The client has more functionality and, therefore, more responsibility.
- Messaging is optimized for batches instead of individual messages.
- Messages are retained even after they are consumed; they can be consumed again.

The results of these design decisions are:

- Extreme horizontal scalability
- Very high throughput
- High availability
- Different semantics and message delivery guarantees

Kafka Terminology

Kafka uses its own terminology when it comes to its basic building blocks and key concepts. The usage of these terms might vary from other technologies. The following provides a list and definition of the most important concepts of Kafka:

Broker

A broker is a server that stores messages sent to the topics and serves consumer requests.

Topic

A topic is a queue of messages written by one or more producers and read by one or more consumers.

Producer

A producer is an external process that sends records to a Kafka topic.

Consumer

A consumer is an external process that receives topic streams from a Kafka cluster.

Client

Client is a term used to refer to either producers and consumers.

Record

A record is a publish-subscribe message. A record consists of a key/value pair and metadata including a timestamp.

Partition

Kafka divides records into partitions. Partitions can be thought of as a subset of all the records for a topic.

Continue reading to learn more about each key concept.

Brokers

Learn more about Brokers.

Kafka is a distributed system that implements the basic features of an ideal publish-subscribe system. Each host in the Kafka cluster runs a server called a broker that stores messages sent to the topics and serves consumer requests.

Figure 2: Brokers in a Publish-Subscribe System

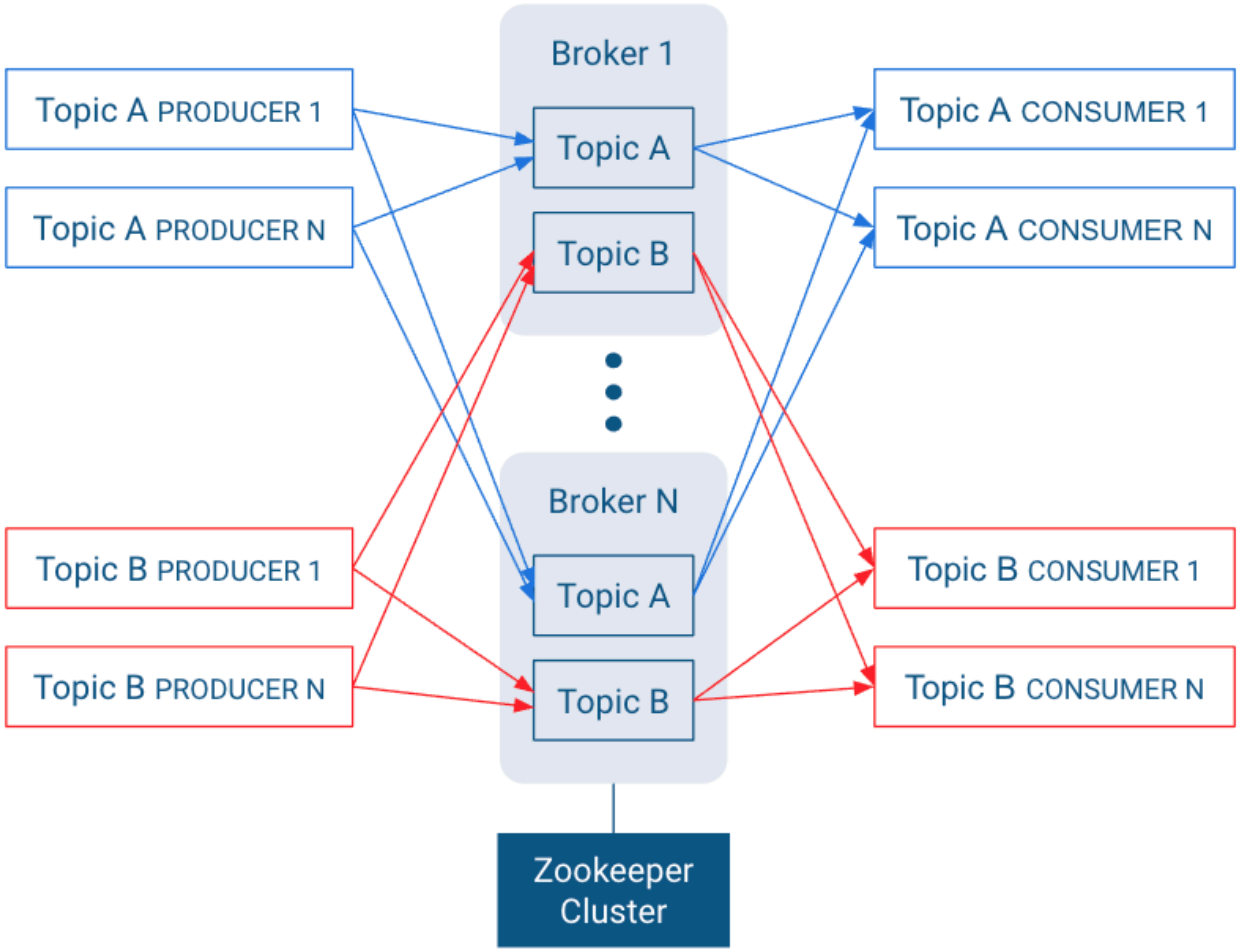

Kafka is designed to run on multiple hosts, with one broker per host. If a host goes offline, Kafka does its best to ensure that the other hosts continue running. This solves part of the “No Downtime” and “Unlimited Scaling” goals of the ideal publish-subscribe system.

Kafka brokers all talk to Zookeeper for distributed coordination, which also plays a key role in achieving the "Unlimited Scaling" goal from the ideal system.

Topics are replicated across brokers. Replication is an important part of “No Downtime”, “Unlimited Scaling,” and “Message Retention” goals.

There is one broker that is responsible for coordinating the cluster. That broker is called the controller.

Topics

Learn more about Kafka topics.

In any publish-subscribe system, messages from one publisher, called producers in Kafka, have to find their way to the subscribers, called consumers in Kafka. To achieve this, Kafka introduces the concept of topics, which allow for easy matching between producers and consumers.

A topic is a queue of messages that share similar characteristics. For example, a topic might consist of instant messages from social media or navigation information for users on a web site. Topics are written by one or more

producers and read by one or more consumers. A topic is identified by its name. This name is part of a global namespace of that Kafka cluster.

As each producer or consumer connects to the publish-subscribe system, it can read from or write to a specific topic.

Figure 3: Topics in a Publish-Subscribe System

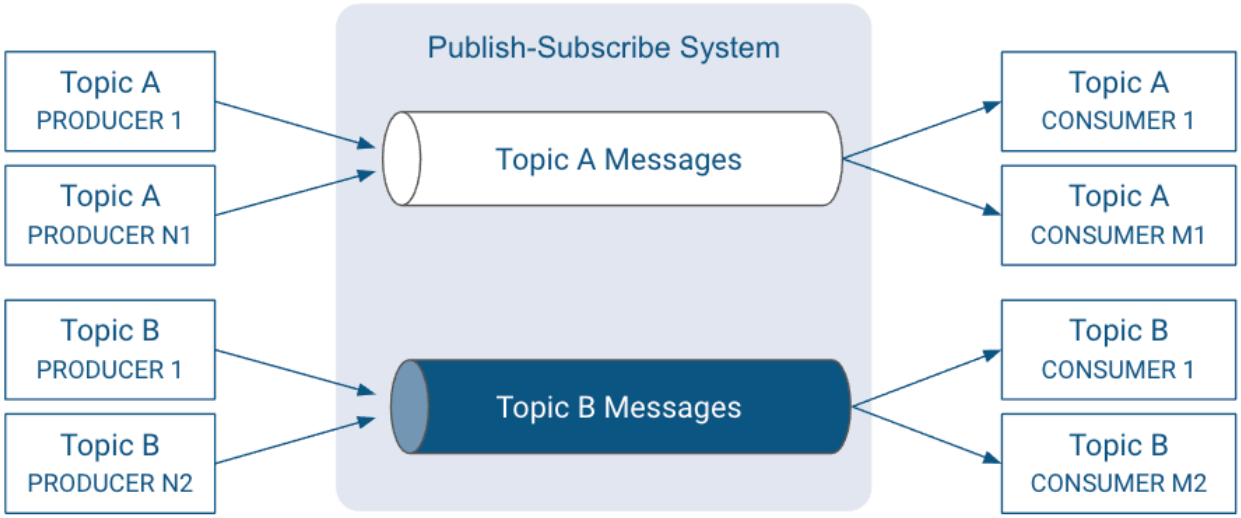

Records

Learn more about Kafka records.

In Kafka, a publish-subscribe message is called a record. A record consists of a key/value pair and metadata including a timestamp. The key is not required, but can be used to identify messages from the same data source. Kafka stores keys and values as arrays of bytes. It does not otherwise care about the format.

The metadata of each record can include headers. Headers may store application-specific metadata as key-value pairs. In the context of the header, keys are strings and values are byte arrays.

For specific details of the record format, see Apache Kafka documentation.

Related Information

Record Format

Partitions

Learn more about Kafka partitions.

Instead of all records handled by the system being stored in a single log, Kafka divides records into partitions. Partitions can be thought of as a subset of all the records for a topic. Partitions help with the ideal of “Unlimited Scaling”.

Records in the same partition are stored in order of arrival.

When a topic is created, it is configured with two properties:

partition count

The number of partitions that records for this topic will be spread among.

replication factor

The number of copies of a partition that are maintained to ensure consumers always have access to the queue of records for a given topic.

Each topic has one leader partition. If the replication factor is greater than one, there will be additional follower partitions. (For the replication factor = M, there will be M-1 follower partitions.)

Any Kafka client (a producer or consumer) communicates only with the leader partition for data. All other partitions exist for redundancy and failover. Follower partitions are responsible for copying new records from their leader partitions. Ideally, the follower partitions have an exact copy of the contents of the leader. Such partitions are called in-sync replicas (ISR).

With N brokers and topic replication factor M, then

- If M < N, each broker will have a subset of all the partitions
- If M = N, each broker will have a complete copy of the partitions

In the following illustration, there are N = 2 brokers and M = 2 replication factor. Each producer may generate records that are assigned across multiple partitions.

Figure 4: Records in a Topic are Stored in Partitions, Partitions are Replicated across Brokers

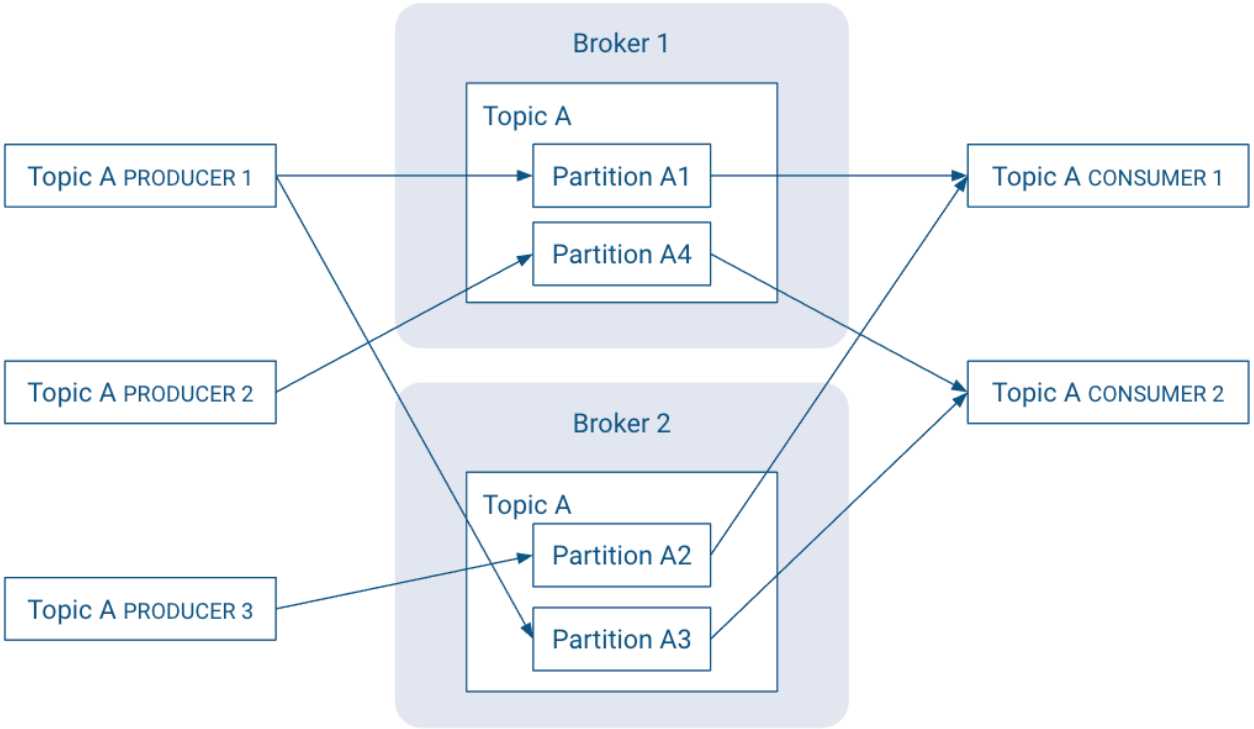

Partitions are the key to keeping good record throughput. Choosing the correct number of partitions and partition replications for a topic:

- Spreads leader partitions evenly on brokers throughout the cluster
- Makes partitions within the same topic are roughly the same size
- Balances the load on brokers.

Record order and assignment

Learn about how Kafka assigns records to partitions.

By default, Kafka assigns records to a partitions round-robin. There is no guarantee that records sent to multiple partitions will retain the order in which they were produced. Within a single consumer, your program will only have record ordering within the records belonging to the same partition. This tends to be sufficient for many use cases, but does add some complexity to the stream processing logic.

Tip: Kafka guarantees that records in the same partition will be in the same order in all replicas of that partition.

If the order of records is important, the producer can ensure that records are sent to the same partition. The producer can include metadata in the record to override the default assignment in one of two ways:

- The record can indicate a specific partition.
- The record can includes an assignment key.

  The hash of the key and the number of partitions in the topic determines which partition the record is assigned to. Including the same key in multiple records ensures all the records are appended to the same partition.

Logs and log segments

Learn more about logs and log segments.

Within each topic, each partition in Kafka stores records in a log structured format. Conceptually, each record is stored sequentially in this type of “log”.

Figure 5: Partitions in Log Structured Format

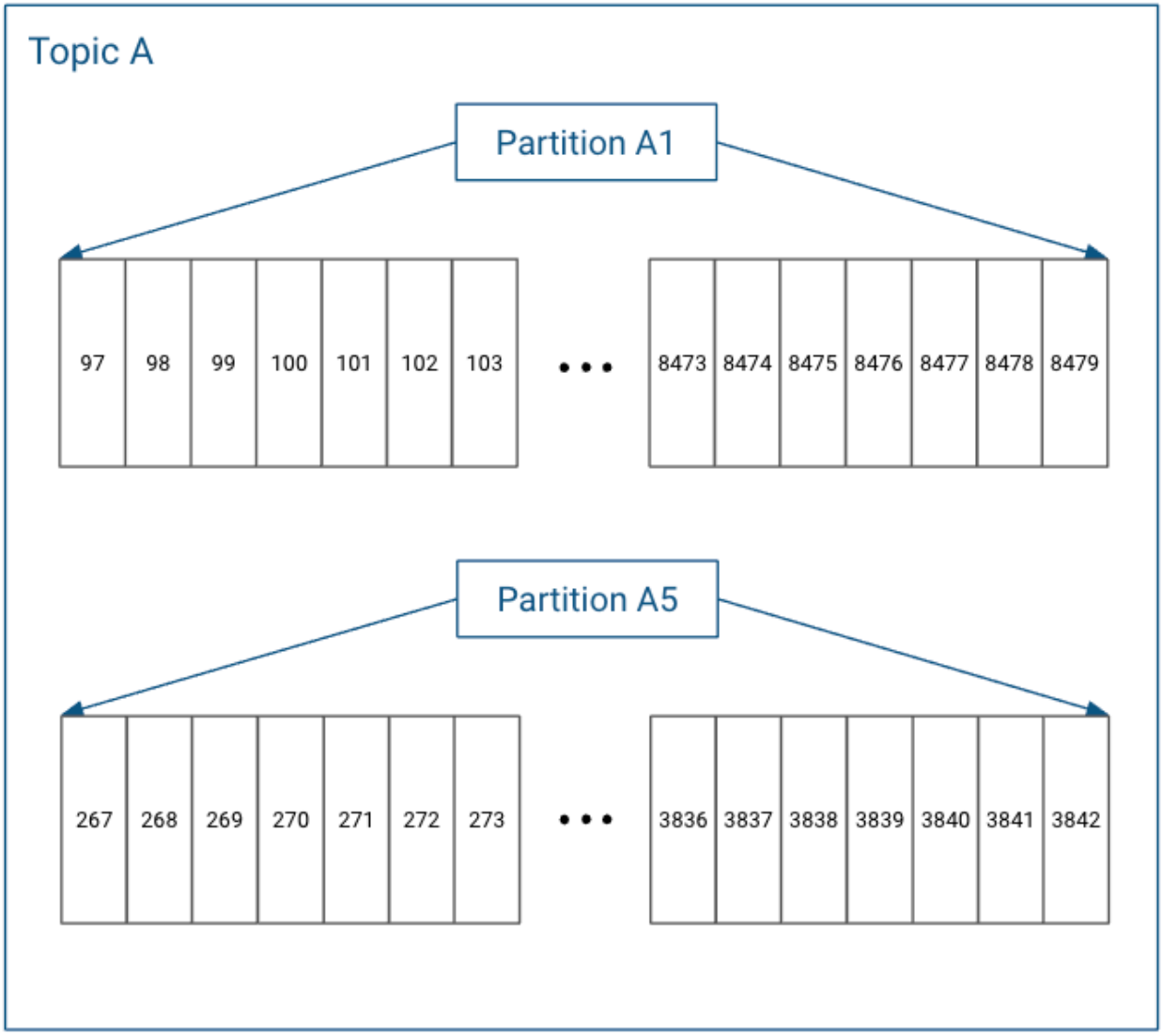

Note: These references to “log” should not be confused with where the Kafka broker stores their operational logs.

In actuality, each partition does not keep all the records sequentially in a single file. Instead, it breaks each log into log segments. Log segments can be defined using a size limit (for example, 1 GB), as a time limit (for example, 1 day), or both. Administration around Kafka records often occurs at the log segment level.

Each of the partitions is broken into segments, with Segment N containing the most recent records and Segment 1 containing the oldest retained records. This is configurable on a per-topic basis.

Figure 6: Partition Log Segments

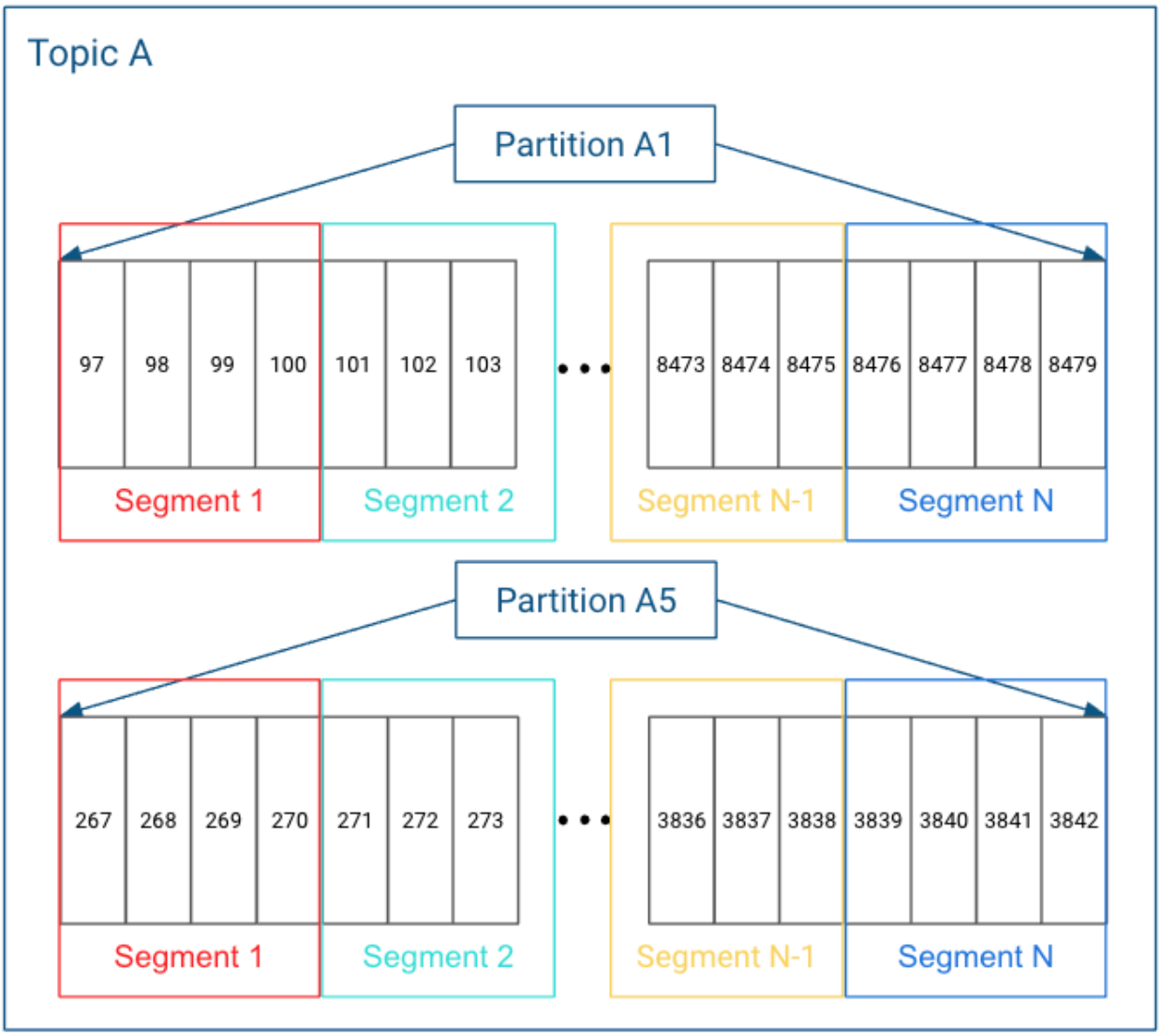

Related Information

Log-structured file system

Kafka brokers and Zookeeper

Learn about the types of data maintained in Zookeeper by the brokers.

The broker, topic, and partition information are maintained in Zookeeper. In particular, the partition information, including partition and replica locations, updates fairly frequently. Because of frequent metadata refreshes, the

connection between the brokers and the Zookeeper cluster needs to be reliable. Similarly, if the Zookeeper cluster has other intensive processes running on it, that can add sufficient latency to the broker/Zookeeper interactions to cause issues.

- Kafka Controller maintains leadership through Zookeeper (shown in orange)
- Kafka Brokers also store other relevant metadata in Zookeeper (also in orange)
- Kafka Partitions maintain replica information in Zookeeper (shown in blue)

Figure 7: Broker/ZooKeeper Dependencies

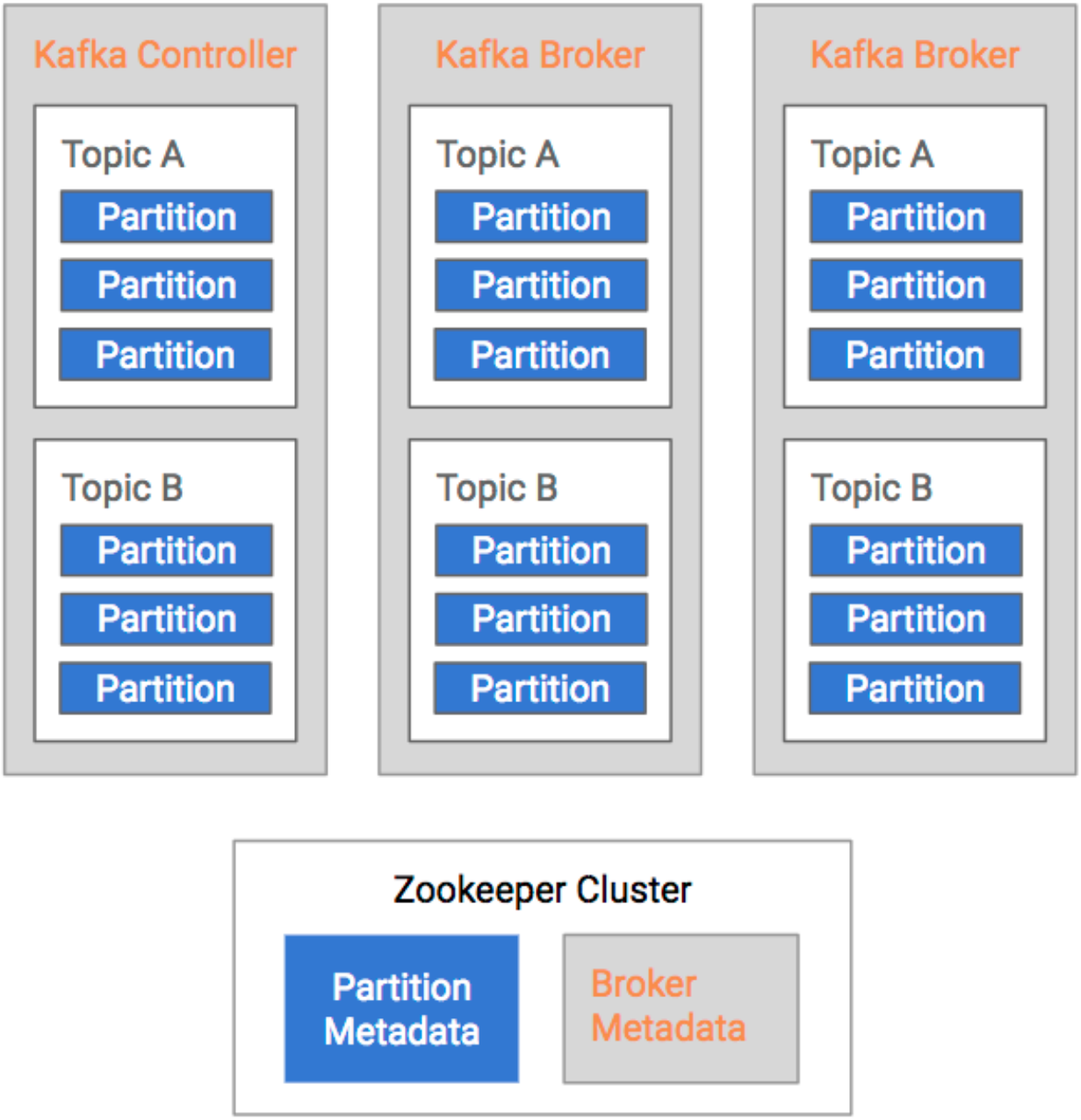

Leader positions and in-sync replicas

An overview of how leader positions and in-sync replicas can affect Kafka performance.

Consider the following example which shows a simplified version of a Kafka cluster in steady state. There are N brokers, two topics with nine partitions each. Replicated partitions are not shown for simplicity.

Figure 8: Kafka Cluster in Steady State

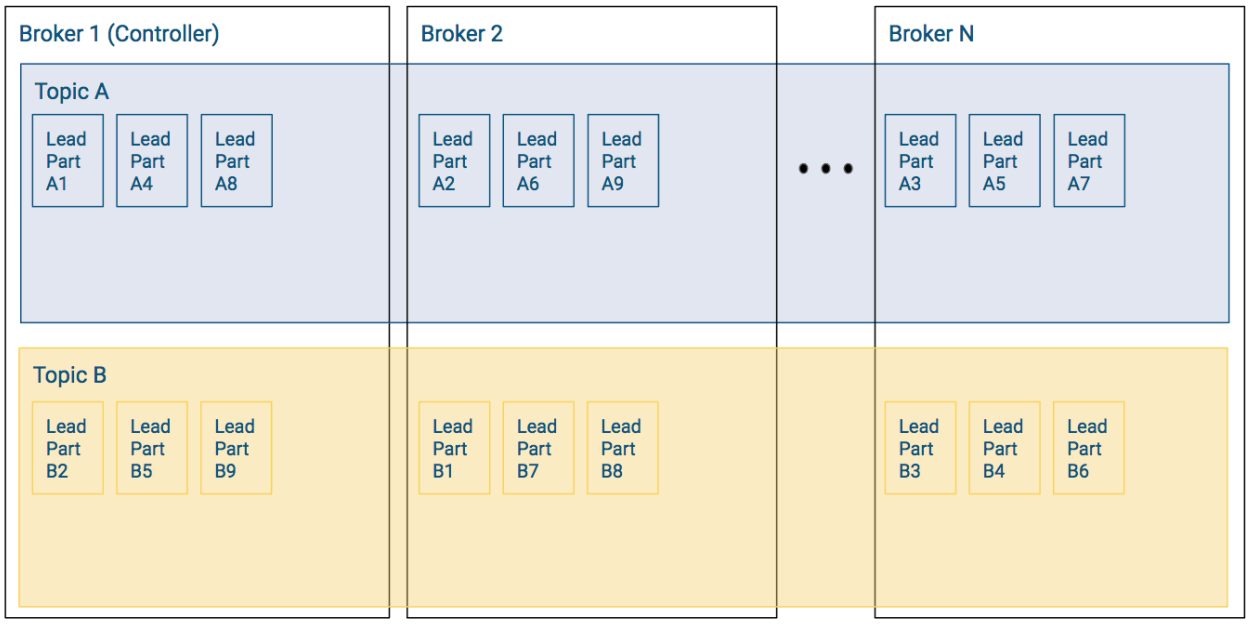

In this example, each broker shown has three partitions per topic and the Kafka cluster has well balanced leader partitions. Recall the following:

- Producer writes and consumer reads occur at the partition level.
- Leader partitions are responsible for ensuring that the follower partitions keep their records in sync.

Since the leader partitions are evenly distributed, most of the time the load to the overall Kafka cluster is relatively balanced.

Leader Positions

Now lets look at an example where a large chunk of the leaders for Topic A and Topic B are on Broker 1.

Figure 9: Kafka Cluster with Leader Partition Imbalance

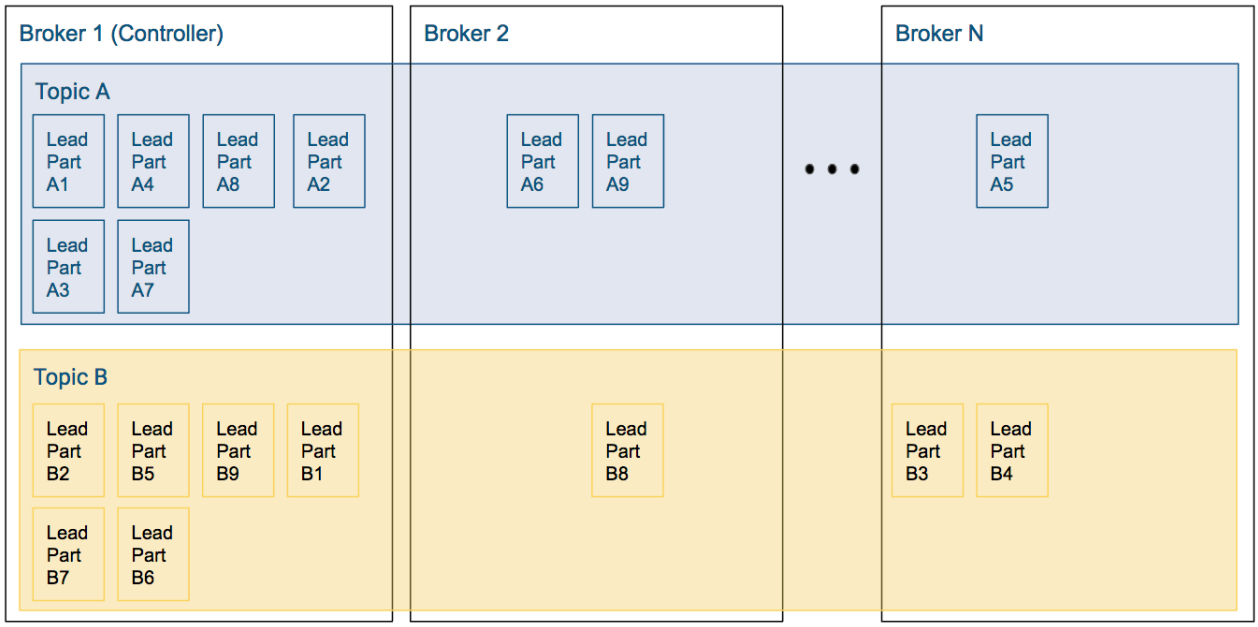

In a scenario like this a lot more of the overall Kafka workload occurs on Broker 1. Consequently this also causes a backlog of work, which slows down the cluster throughput, which will worsen the backlog. Even if a cluster starts with perfectly balanced topics, failures of brokers can cause these imbalances: if the leader of a partition goes down one of the replicas will become the leader. When the original (preferred) leader comes back, it will get back leadership only if automatic leader rebalancing is enabled; otherwise the node will become a replica and the cluster gets imbalanced.

In-Sync Replicas

Let’s take a closer look at Topic A from the previous example that had imbalanced leader partitions. However, this time let's visualize follower partitions as well:

- Broker 1 has six leader partitions, broker 2 has two leader partitions, and broker 3 has one leader partition.
- Assuming a replication factor of 3.

Figure 10: Kafka Topic with Leader and Follower Partitions

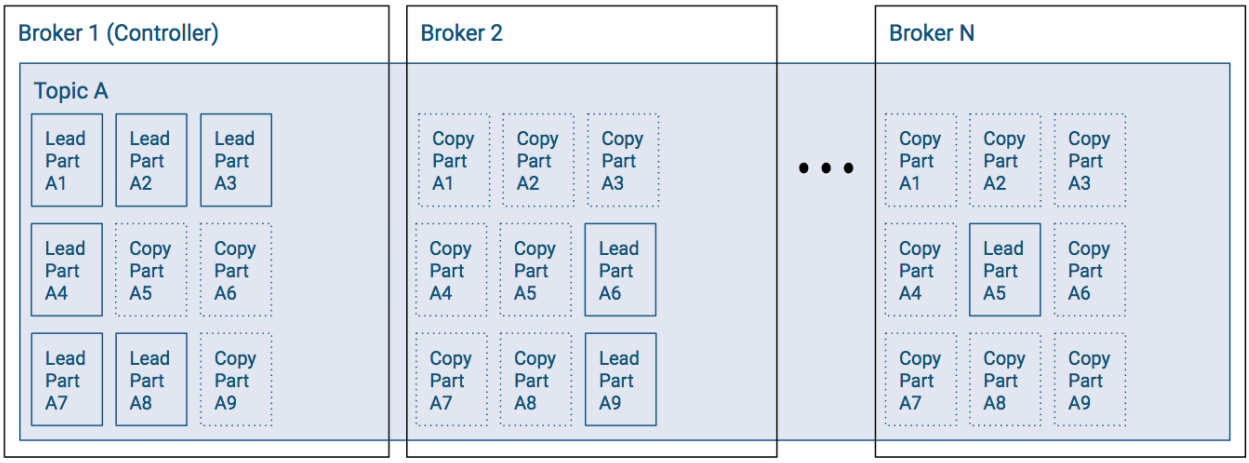

Assuming all replicas are in-sync, then any leader partition can be moved from Broker 1 to another broker without issue. However, in the case where some of the follower partitions have not caught up, then the ability to change leaders or have a leader election will be hampered.

Kafka stretch clusters

Stretch clusters are highly resilient and cost-effective Kafka deployments. Learn the requirements, possible deployment architectures, and configuration needed to deploy Kafka stretch clusters.

A stretch cluster is a single logical Kafka cluster deployed across multiple Data Centers (DC) or other independent physical infrastructures such as cloud availability zones. By utilizing a single Kafka cluster over multiple DCs, you can achieve the following:

- Strong data durability guarantees as a result of synchronous replication between DCs.
- Automatic failover of clients when a DC goes down.

Stretch clusters have specific requirements for latency between DCs and require specific configurations for achieving strong guarantees on data durability. Additionally, in some use cases, complimentary features can be used to minimize cross-DC traffic. It is important to note that the extra latency introduced by this setup can have a significant impact on the throughput of Kafka.

Kafka stretch cluster requirements

Learn about the requirements of a stretch cluster deployment.

Number of Data Centers and the ZooKeeper quorum

A stretch cluster requires at least three DCs to function properly. While Kafka can support a setup where brokers are hosted in two DCs, Kafka is dependent on ZooKeeper for metadata storage. Theoretically, it is possible to use more than three DCs in the stretch cluster architecture. However, in the majority of cases, the cost of the additional cross-DC data traffic will outweigh the benefits of having the data copied to more than three DCs.

Since ZooKeeper is a quorum-based system, a majority is required to keep the metadata storage and Kafka running. To support service continuity in case of a DC failure, sufficient ZooKeeper nodes must remain so that a majority vote can happen. For example, in a two DC setup, one of the two DCs must host more ZooKeeper nodes than the other. If the DC that hosts the higher number of ZooKeeper nodes goes down, ZooKeeper majority is lost. This results in the Kafka cluster also going down.

Note: *KIP-500* introduced a Raft based quorum system (KRaft) that acts as an alternative to ZooKeeper. While this document discusses the use of Zookeeper, the requirements and architecture of a stretch cluster are the similar for deployments that use Kraft. This is because Kraft is also a quorum based system.

Latency between Data Centers

In order to achieve hight throughput, Kafka is designed from the ground up with the assumption that latency is low between the cluster nodes. Because of this, a stretch cluster deployment requires low latency between the DCs. There are two major issues with high latency in a stretch cluster: Kafka replication throughput and metadata operation latency.

While the replication stays functionally correct even with high latency, the throughput of Kafka suffers greatly from increased latency.

ZooKeeper is also sensitive to latency. ZooKeeper changes are synchronized in the cluster, making metadata changes run slow. These changes involve controller election, topic metadata related operations (create/delete/update config), and ISR changes. In an ideal deployment with low latency, these are infrequent operations and do not affect the throughput of data production and consumption. However, the higher the latency, the more frequent these operation become. With increased latency, replicas lag behind the ISR more easily, which generates more ISR change operations. There is also a phenomenon called ISR thrashing. This occurs when a replica frequently joins but then lags behind the ISR, which is also tied to high latency scenarios.

Because of these issues, Cloudera recommends using infrastructure where the maximum latency between the DCs is 50 ms. In general, the latency should be minimized. Any increase in latency greatly affects the throughput of a Kafka stretch cluster. While it is possible for a Kafka stretch cluster to function correctly in some use cases with higher latency (for example, light duty clusters), Cloudera does not recommend using the stretch cluster architecture if you have high latency.

Related Information

KIP-500: Replace ZooKeeper with a Self-Managed Metadata Quorum

Kafka stretch cluster architectures

A Kafka stretch cluster can be operated in a 3 or 2.5 DC configuration.

A stretch cluster requires at least three DCs to function correctly. Based on the use case, it is possible to span the Kafka cluster over two DCs, and only use the third cluster as a tie-breaker ZooKeeper node. This architecture is called the 2.5 DC setup. Typical use cases would either use the 2.5 DC setup, or a 3 DC setup. It is possible to use multiple DCs (given that the latency requirements are met), but in general, cross-DC traffic should be minimized.

2.5 DC stretch cluster

Figure 11: 2.5 DC Kafka stretch cluster

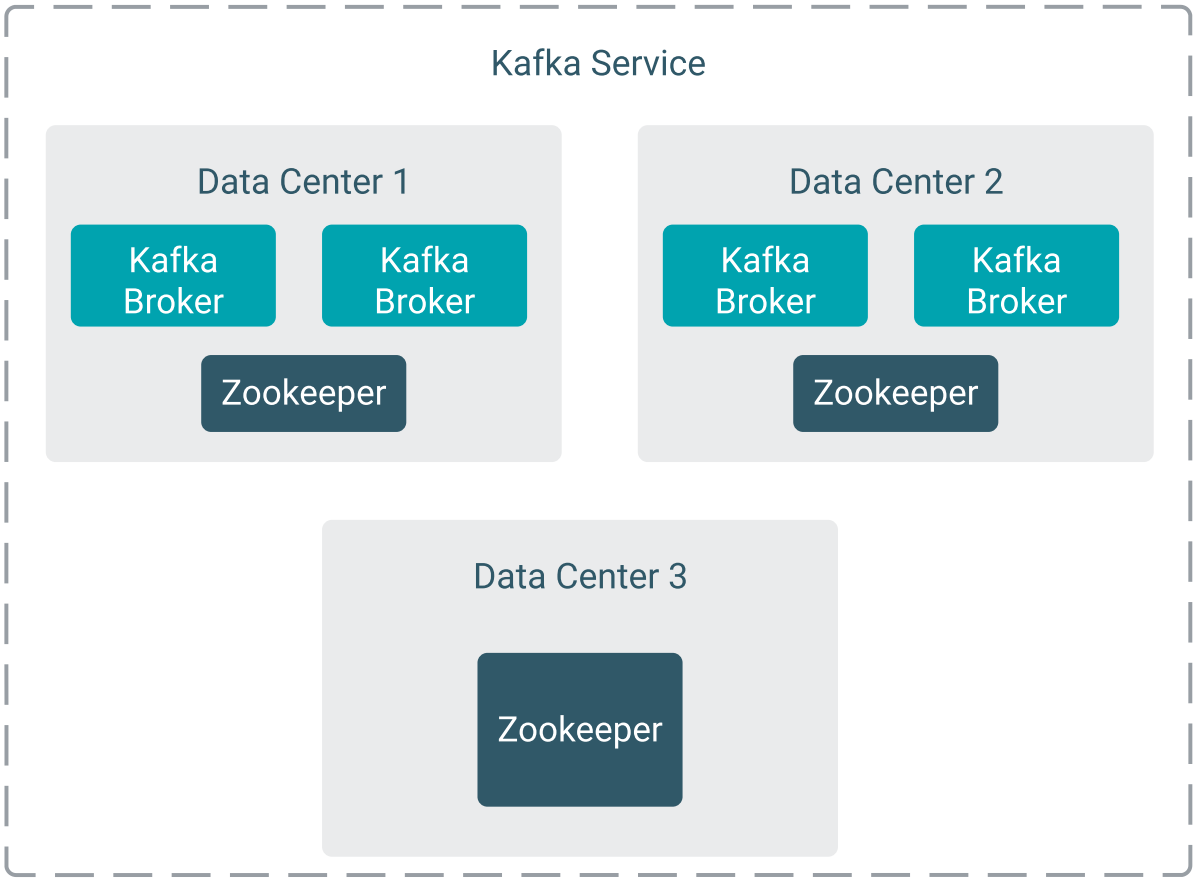

Pros:

- Cost efficient (both in terms of nodes used and cross-DC data traffic).
- Can ensure synchronized writes into two DCs (data durability, RPO=0).

- Can tolerate single DC failure, reads supported.
- Less sensitive to the tie-breaker DC failure.

Con:

- Writes for durable topics are not supported after a DC that hosts brokers goes down.

3 DC stretch cluster

Figure 12: 3 DC stretch cluster

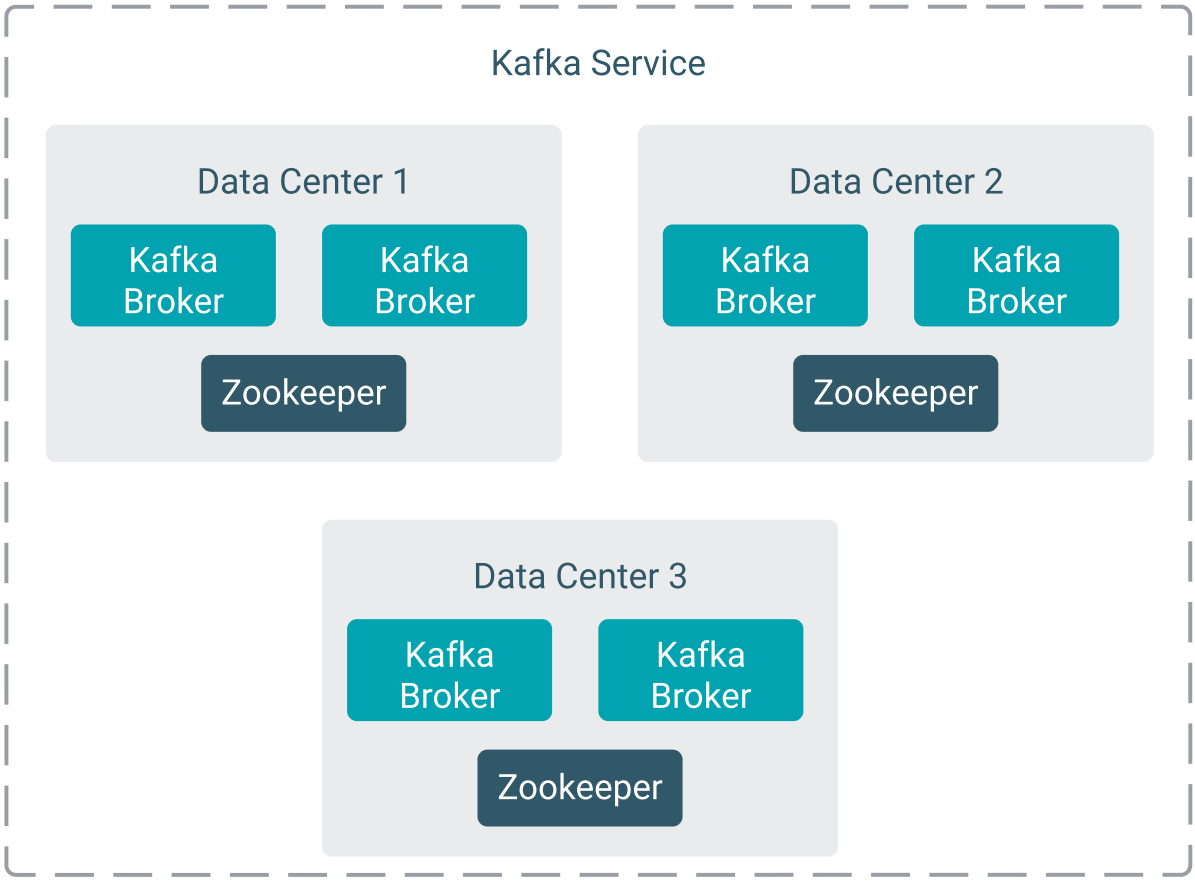

Pros:

- Can ensure synchronized writes into at least two DCs (data durability, RPO=0).
  - Depending on the configuration, can guarantee writes over three DCs, but write availability is reduced.
- Can tolerate a single DC going down. This is true for both reads and writes.
  - Write availability depends on the configuration (whether two or three DC guarantee was configured).

Con:

- As a result of more nodes in all DCs and more cross-DC data traffic, this architecture is more expensive than a 2.5 DC setup.

Cluster configuration for Kafka stretch clusters

Kafka stretch clusters require the brokers, topics, and clients of the deployment to be configured. Configuration is needed to achieve the desired guarantees on data durability.

To achieve the high data durability in a stretch cluster, configuring the brokers, topics and clients in the deployment is required. Specifically you have to:

- Configure brokers and topics so that topic partition replicas are distributed evenly among the DCs.
- Configure producers to write messages with the highest durability guarantee (acks=all).
- Configure topics to have the required minimum in-sync replicas when accepting writes.

Even distribution of topic replicas among DCs

To ensure that replicas are distributed evenly among the DCs, stretch clusters use Kafka's rack awareness feature (*KIP-36*). Kafka brokers have a broker.rack configuration. This property specifies the physical location of the broker in the form of a rack ID. The rack ID is an arbitrary, user defined string that can represent any type of physical infrastructure. In stretch clusters, the property and the ID can be used to let the broker know which DC it is running in. For example, the ID can be set to DC1 for the brokers inside DC1, DC2 for DC2, and so on.

If a topic is created with a replication factor of two or more in a Kafka instance that has its rack IDs specified, Kafka assigns the replicas in a way that not only ensures that the replicas of the same partition are located in different brokers, but will also try to distribute them evenly among the DCs (using round-robin).

For example, assume you have two DCs ("racks") with six brokers and two topics. The replication factor of the topics is not uniform. Topic_A has a replication factor of 4, Topic_B has a replication factor of 5. In this scenario, Kafka will create the following replica assignments:

```
Topic_A, RF=4 partition 0: [1, 4, 2, 5]
Topic_B, RF=5 partition 0: [1, 4, 2, 5, 3]
```

For Topic_A there are two replicas in DC1 and two replicas in DC2. Topic_A is evenly distributed. This happens if *[***RF***] %[***DC COUNT***]* == 0.

For Topic_B there are three replicas in DC1 and two replicas in DC2. Since *[***RF***] % [***DC COUNT***]* != 0, distribution is not perfectly even, but round-robin ensures that the maximum difference between replica counts per rack is one.

Figure 13: Kafka stretch cluster replica assignment example

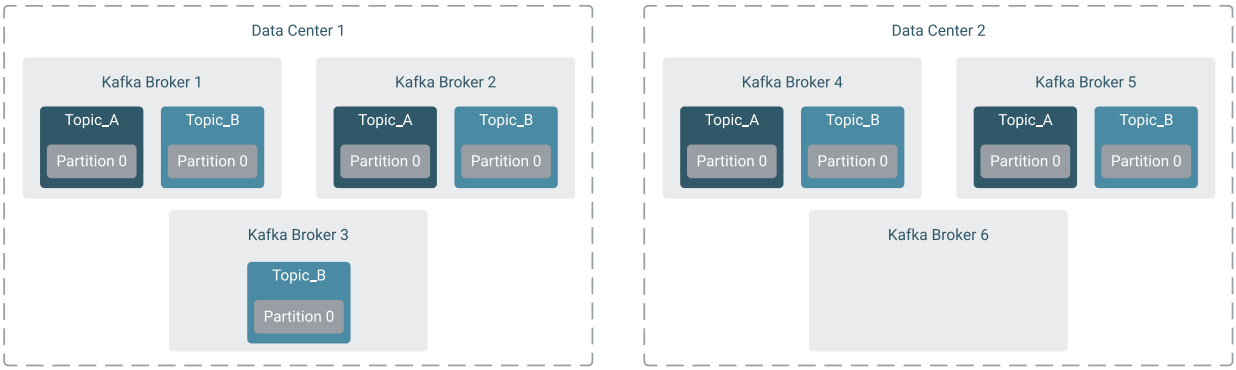

To utilize this behavior to your advantage, durable topics should be configured to have a replication factor which is a multiple of the number of DCs. For example, if you have three DCs, set the replication factor to 3, 6, 9, and so on.

In the majority of cases, having a single replica inside a DC is sufficient. That is, setting the replication factor higher than the number of DCs is usually not necessary. Doing so increases cross-DC traffic and storage capacity requirements. However, having more replicas does have benefits, which are as follows:

1. Increased availability. More broker failures can be tolerated without the topic being unavailable.
2. Follower fetching is supported even in case of single broker failures.
3. Increased data durability in case of disk failures.

Note: Increased data durability can also be achieved with fault tolerant disk technologies or services.

Producer acknowledgments (acks)

Producers have a configuration property called acks. The acks property affects when a leader should send an acknowledgement to the producer after a write operation. The property can be set to three different values, each corresponding to a level of acknowledgement. Each level represents a tradeoff between throughput and write durability. The configuration values are as follows:

0

No acknowledgement. The producer handles the write as finished right after it is sent to the leader. This setting provides the highest throughput, but also has the highest risk of losing writes. Writes might not even make it to the leader log (and disk) before the leader goes down.

1

Leader acknowledgement. The leader sends an acknowledgement right after the message was appended to its log. This setting provides lower throughput compared to acks=0, but ensures that at least a single log has the message appended.

all

ISR acknowledgement. The leader appends the message to its log, waits for the in-sync replicas (ISR) to replicate the message, and then sends the acknowledgment. This setting provides the lowest throughput. This is because the leader might need to wait for multiple followers to fetch and write the message to their log before the acknowledgment can be sent out. However, it has the highest durability as it ensures that multiple brokers have appended the message to their log. With this setting, the decision on when a write should be considered done is handed over from the producer to the partition leader.

For high durability, producers must be configured to use acks=all. To understand and fully utilize the ISR concept in the stretch cluster, you need to also correctly configure the topics.

ISR and minimum in-sync replicas of a topic

For durable writes, besides the replication factor, it is also important to correctly configure the minimum in-sync replica count (min.insync.replicas) of a topic.

The ISR is tracked by Kafka based on broker configuration. The leader always keeps track which of its followers are in sync (caught up), and which are lagging behind. The replicas that are caught up are part of the ISR.

When a producer sends a request with acks=all, the leader will wait for **all members of the ISR** to replicate the message before sending an acknowledgment. If there are nine replicas (one leader and eight followers), and all are part of the ISR, then the acknowledgment will be sent after all eight followers have replicated the message. By default, if only the leader is part of the ISR, then the acknowledgment is sent right after the leader appends the message to its log.

To change this behavior, topics have a min.insync.replicas property. This property represents a minimum size requirement for the ISR when a producer writes into a topic with acks=all. This means that if min.insync.replicas=2, then the leader and at least a single follower must be part of the ISR and that the ISR must replicate the message before the acknowledgement is sent.

In a stretch cluster setup, if you want to ensure that writes are replicated to multiple DCs before acknowledgment happens, the min.insync.replicas property must be correctly configured. The following example demonstrates how you can calculate the correct min.insync.replicas value for your deployment.

Assume that you have three DCs. You want to replicate writes into at least two DCs before a write is acknowledged. Replication factor is six. That is:

```
[***DC COUNT***]= 3
[***MINIMUM DC REPLICAS***]= 2
[***RF***]= 6
```

First, you need to calculate how many replicas reside in each DC, *[***REPLICA PER DC***]*. This can be done by dividing the replication factor by the number of DCs:

```
[***RF***]/[***DC COUNT***]=[***REPLICA PER DC***]
```

```
6 / 3 = 2
```

Afterwards, you can calculate min.insync.replicas using the following formula:

```
[***REPLICA PER DC***] * ([***MINIMUM DC REPLICAS***] - 1) + 1 = min.insy
nc.replicas
```

```
2 * (2 - 1) + 1 = 3
```

This formula ensures that whichever replicas are in sync for the topic, there will always be at least a *[***MINIMUM DC REPLICAS***]* number of DCs hosting the active replicas. However, whenever you have fewer replicas in the ISR, writes will start to fail because the min.insync.replica requirement is not met.

With min.insync.replicas=3 you can ensure that even in the worst case scenario (most of the replicas in the ISR are located in the same DC), at least one replica will be located in a different DC.

To look at another example, assume you have the same setup, but change *[***MINIMUM DC REPLICAS***]* to three, min.insync.replicas would change to five.

```
2 * (3 - 1) + 1 = 5
```

With min.insync.replicas=5 you can ensure that even in a worst case scenario of ISR members, all three DCs are replicating the write before it is acknowledged. However, at the same time, this means that any DC going down reduces the ISR size to four, which will cause the cluster to fail durable produce requests.

Partition leadership

Everything described so far about the ISR and durable writes depends on the fact that partition leadership changes depend on the ISR. When the leader is not available, Kafka transfers the leadership to one of the ISR members. This ensures that all writes acknowledged by the cluster will be present on the next leader. Because of this, all durable topics must have unclean leader election disabled. Otherwise, accepted writes might get lost in an unclean leader election.

Conclusion: Durable writes (RPO=0)

With the three essential configurations done (broker.rack, acks=all, min.insync.replicas), you ensure that:

- Replicas are evenly distributed among DCs.
- Producers write with the highest durability.
- The level of durability configured with min.insync.replicas ensures that writes are synchronized to the required number of DCs.
- Topics only allow clean leader elections (based on the ISR).

Related Information

KIP-36 Rack aware replica assignment

Kafka disaster recovery

Learn about the cluster architectures you can use when designing highly available and resilient Kafka clusters.

Kafka has built-in replication for its topics and allows users to carefully tweak the data durability configurations to achieve desired redundancy guarantees. When designing a deployment where data is replicated over multiple Data Centers (DC) and has disaster recovery (DR) options, you need to carefully analyse your data durability and business continuity requirements. The following sections introduce the architectural options that you can choose from when building a resilient multi-DC Kafka deployment. Additionally, guidance is provided that help you in choosing the right architecture for your use case.

There are two major architectural groups for achieving multi-DC DR with Kafka. These are as follows:

Stretch clusters

A stretch cluster is a single logical Kafka cluster deployed across multiple DCs or other independent physical infrastructures such as cloud availability zones. For more information, see *Stretch clusters* or *Kafka Stretch cluster reference architecture*.

Replication using Streams Replication Manager

Streams Replication Manager is an enterprise-grade replication solution that enables fault tolerant, scalable, and robust cross-cluster Kafka topic replication. A deployment that utilizes Streams Replication Manager for disaster recovery uses multiple Kafka clusters. Streams Replication Manager acts as a bridge between the clusters and replicates the data. For more information, see *Streams Replication Manager Overview*.

The major difference between stretch clusters and an Streams Replication Manager based architecture is how data is replicated between DCs. With stretch clusters, synchronous replication can be achieved. This enables strict guarantees on data durability. With Streams Replication Manager based replication, only asynchronous replication is available. This provides weaker guarantees but higher performance.

Note: In addition to using either a stretch cluster or Streams Replication Manager, you can also choose to have multiple stretch cluster deployments that are replicated by Streams Replication Manager. This can, under the right circumstances, provide the benefits of both approaches.

The DCs located close to each other (low latency) can run as a stretch cluster. Streams Replication Manager can be used to replicate the data to remote DCs (high latency). In this setup, using the 2.5 DC architecture for the stretch clusters can also lower the cost, while the usage of Streams Replication Manager compensates for the lower availability guarantees provided by the 2.5 DC architecture. Additionally, when requirements allow it, if a DC goes down, clients can fail over to a backup cluster and keep the service functional with a temporarily lowered durability guarantee.

This type of architecture is not covered in detail in this document.

Use a stretch cluster if

You have a zero Recovery Point Objective (RPO=0) requirement

RPO=0 means that data should be replicated to multiple DCs before a write is deemed successful (acknowledged by the broker). This can only be achieved by using synchronous replication between DCs, which can be achieved by using a stretch cluster.

However, consider the following. If you can replay data from upstream sources, such as databases, then implementing that recovery function once may be easier than operating and maintaining a stretch cluster.

You need strict message ordering guarantees across DCs

Strict message ordering per partition can only be achieved with a single topic spanning multiple DCs. The Streams Replication Manager architecture involves multiple topics (one original and many replicas), which cannot guarantee the strict ordering during failover or failback operations.

However, consider the following. If your data has an attribute that can be reliably used for ordering, then implementing a reordering step in your downstream processing during application development might be an easier and more cost-effective solution compared to operating and maintaining a stretch cluster.

You need automatic failover for your clients when a DC goes down

The Kafka protocol has built-in cluster discovery and leadership migration on failures. Therefore, fully automatic failover operations can be achieved using a stretch cluster. The Streams Replication Manager based architecture requires a manual step in the failover process. This makes Streams Replication Manager unsuitable for this use case.

You need exactly once transactional processing

Exactly once processing in Kafka is currently only supported within a single cluster.

Use Streams Replication Manager with multiple clusters if

You need high availability during cluster maintenance

When a Kafka cluster needs to be stopped for maintenance, clients can fail over to a backup cluster. With the stretch cluster solution, this is not supported. This is due to the fact there is a single Kafka cluster in the architecture.

You need replication between clusters that have high latency (replication across multiple regions)

The stretch cluster architecture is sensitive to high latency, making it unsuitable for multi-region deployments. The asynchronous replication provided by Streams Replication Manager works well even in high latency environments.

You need high throughput replication between DCs

The throughput of the stretch cluster architecture degrades rapidly with increasing latency. Streams Replication Manager, on the other hand, can provide better replication throughput even in high latency environments.

Related Information

Kafka stretch clusters

Apache Kafka stretch cluster reference architecture

Streams Replication Manager Overview

Kafka rack awareness

Learn about Kafka rack awareness and multi-level rack awareness.

Racks provide information about the physical location of a broker or a client. A Kafka deployment can be made rack aware by configuring rack awareness for the Kafka brokers and clients respectively. Enabling rack awareness can help in hardening your deployment, it provides durability guarantees for your Kafka service, and significantly decreases the chances of data loss

Note: Although the feature is called rack awareness, the term rack does not necessarily mean an actual physical server rack. Instead, a rack from Kafka's perspective represents a physical location or independent physical infrastructure. For example, in many production deployments, the feature is used to specify the individual Data Centers that the brokers and clients are running in.

Rack awareness for Kafka brokers

Learn about Kafka broker rack awareness and how rack-aware Kafka brokers behave.

To avoid a single point of failure, instead of putting all brokers into the same rack, it is considered a best practice to spread your Kafka brokers among racks. In cloud environments Kafka brokers located in different availability zones

or data centers are usually deployed in different racks. Kafka brokers have built in support for this type of cluster topology and can be configured to be aware of the racks they are in.

If you create, modify, or redistribute a topic in a rack-aware Kafka deployment, rack awareness ensures that replicas of the same partition are spread across as many racks as possible. This limits the risk of data loss if a complete rack fails. Replica assignment will try to assign an equal number of leaders for each broker, therefore, it is advised to configure an equal number of brokers for each rack to avoid uneven load of racks.

For example, assume you have a topic partition with 3 replicas and have the brokers configured in 3 different racks. If rack awareness is enabled, Kafka will try to distribute the replicas among the racks evenly in a round-robin fashion. In the case of this example, this means that Kafka will ensure to spread all replicas among the 3 different racks, significantly decreasing the chances of data loss in case of a rack failure.

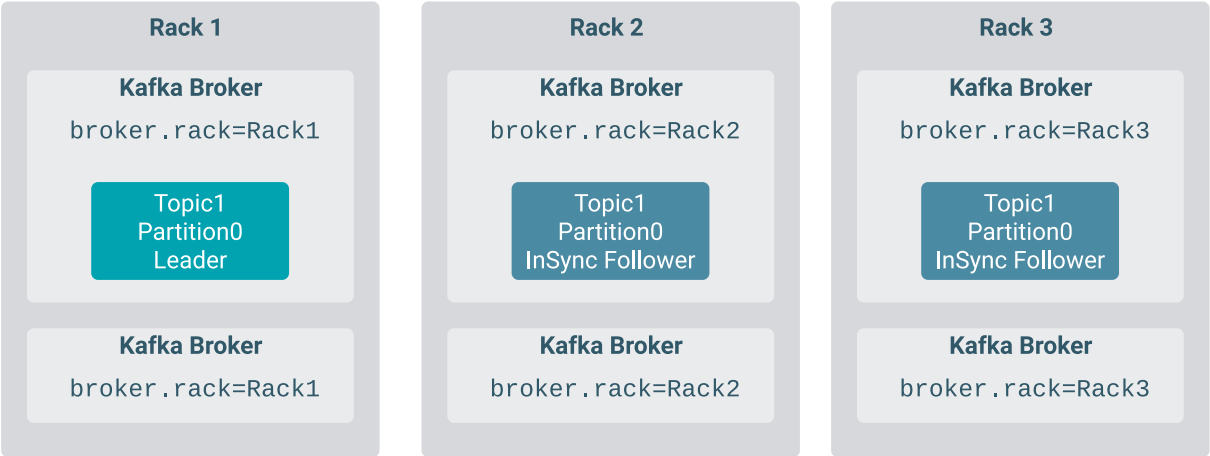

Multi-level rack awareness

Standard rack awareness handles all racks as unique physical locations (for example, Data Centers) that have identical importance. In some use cases, physical locations follow a hierarchical system. For example, besides having multiple DCs, there can be (physical) racks located inside those DCs. In a use case like this, the aim is not only to distribute the replicas among the topmost racks (DCs), but among the second level racks as well (physical racks).

With standard rack awareness, this goal cannot be met. If you use the topmost level as the broker rack IDs (for example, /DC1, /DC2, /DC3), you lose the subsequent levels of the infrastructure. This means that no guarantees are provided for even replica distribution on the second level. Notice how in the following example Rack 2 in each DC is unpopulated.

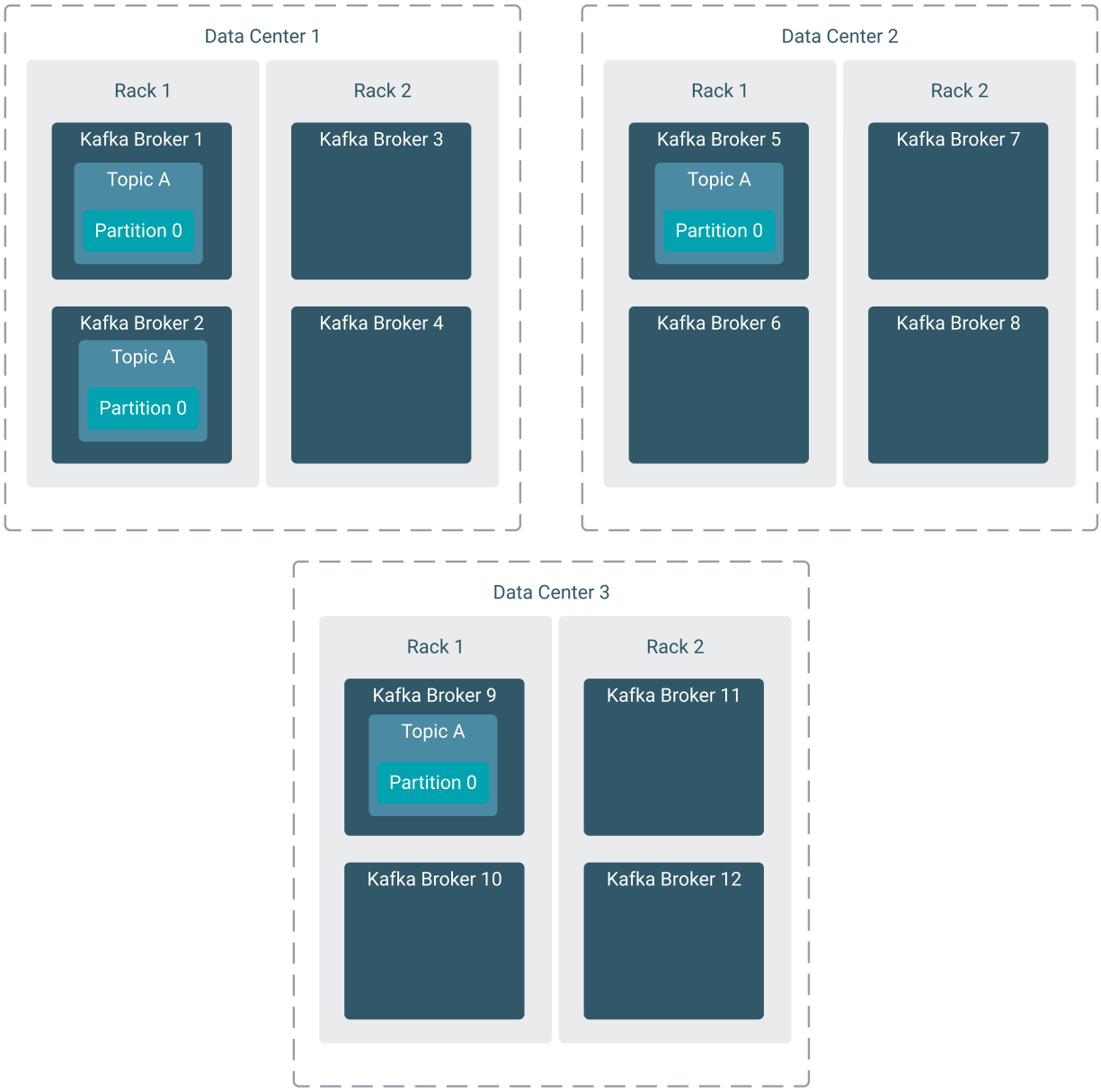

You can use the full physical location as the broker rack IDs (for example, /DC1/R2, /DC2/R5), but then the standard rack awareness feature handles all of the IDs as unique locations that are on the same level. As a result, no guarantee is provided that Kafka evenly distributes replicas on the top (DC) level. Notice how in the following example Data Center 3 is unpopulated.

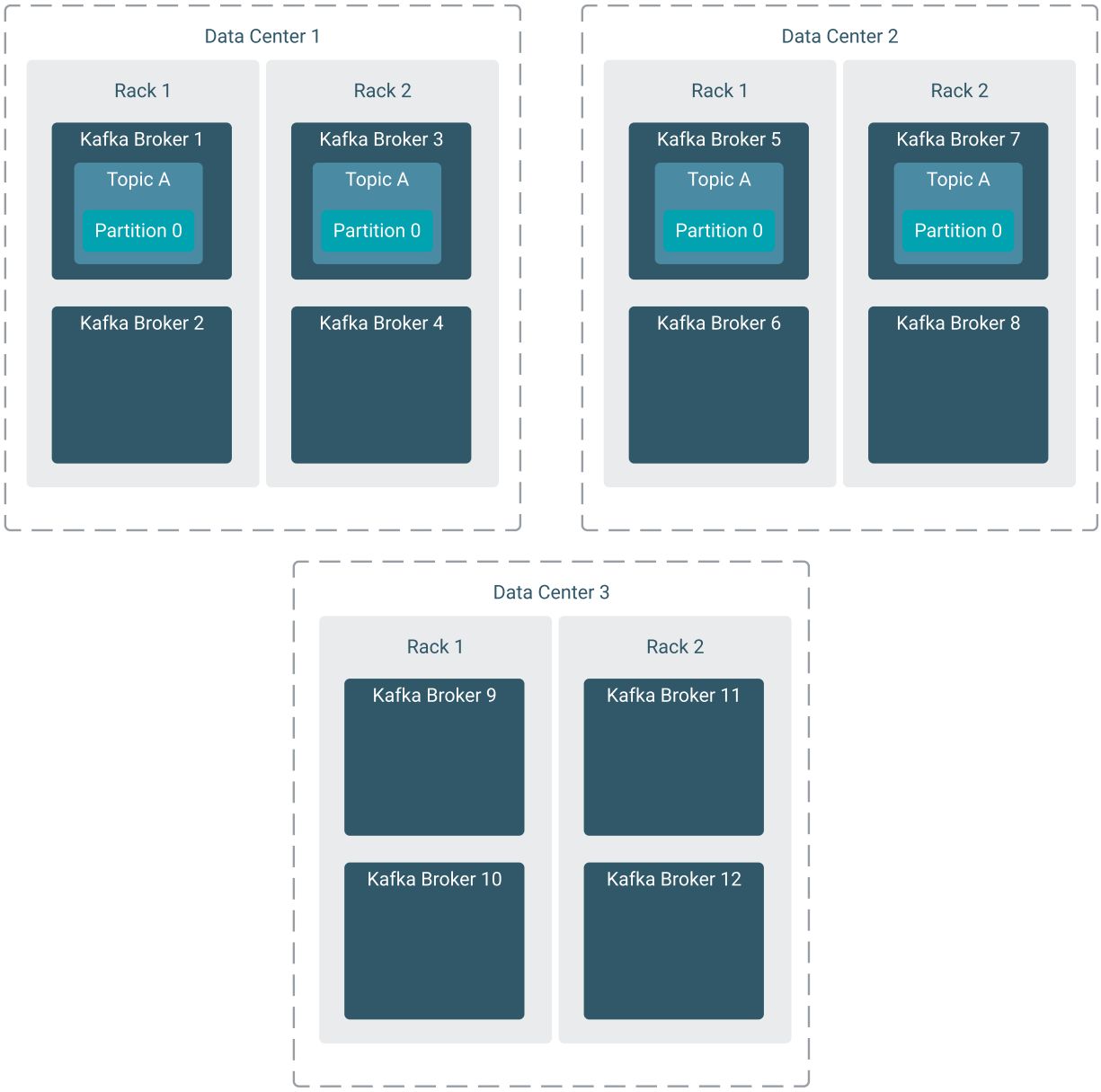

To ensure that multi-level rack guarantees can be met, in addition to standard rack awareness, Cloudera Kafka supports a multi-level rack aware mode. This mode of rack awareness can be configured by specifying multi-level rack IDs and selecting a feature toggle in Cloudera Manager. When enabled, Kafka takes into consideration all levels of the hierarchy and ensures that replicas are spread evenly in the deployment.

Cruise Control optimizations with multi-level rack awareness

If Cruise Control is present in the cluster, and the Kafka brokers run with multi-level rack awareness enabled, Cruise Control will replace all the standard rack aware goals in its configuration with a multi-level rack-aware goal. This ensures that Cruise Control optimizations do not violate the multi-level rack awareness guarantees. For more information on how the goal works, see *Setting capacity estimations and goals* in the Cruise Control documentation.

Related Information

Setting capacity estimations and goals

Configuring rack awareness for Kafka brokers

Configuring multi-level rack awareness for brokers

Rack awareness for Kafka consumers

Learn about follower fetching, which can be used to make Kafka consumers rack aware

When a Kafka consumer tries to consume a topic partition, it fetches from the partition leader by default. If the partition leader and the consumer are not in the same rack, fetching generates significant cross-rack traffic, which has a number of disadvantages. For example, it can generate high costs and lead to lower consumer bandwidth and throughput.

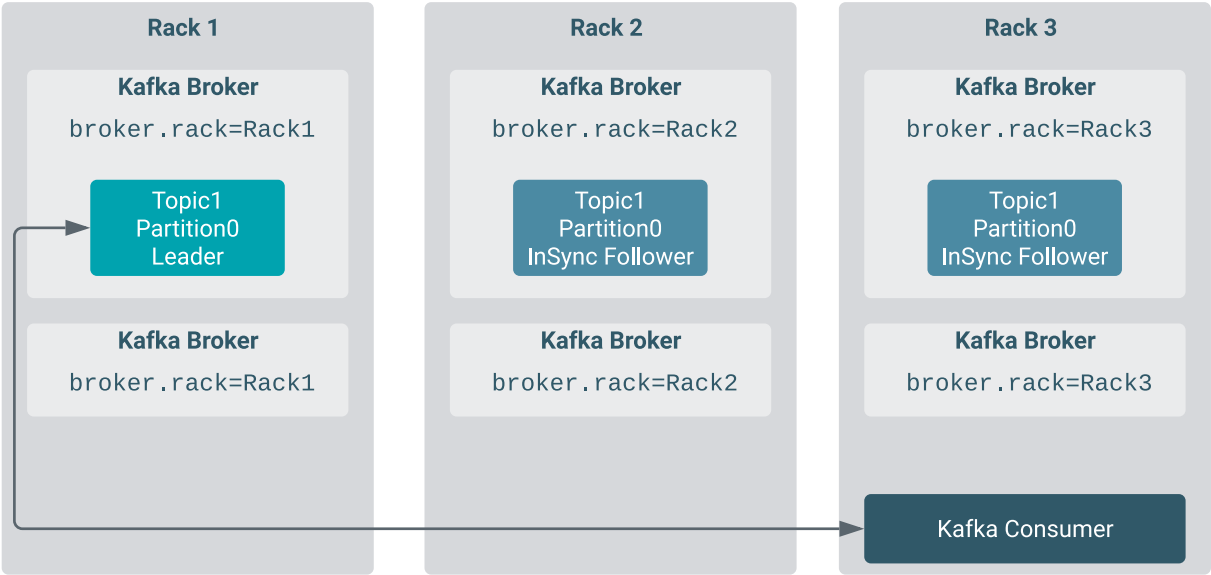

For this reason, it is possible to provide the client with rack information so that the client fetches from the closest replica instead of the leader. If the configured closest replica does not exist (there is no replica for the needed partition in the configured closest rack), it uses the partition leader. This feature is called follower fetching and it can be used to mitigate the costs generated by cross-rack traffic or increase consumer throughput.

Note: Due to the nature of the Kafka protocol and high watermark propagation, consumers might experience increased message latency when fetching from a replica compared to when they are fetching from the leader.

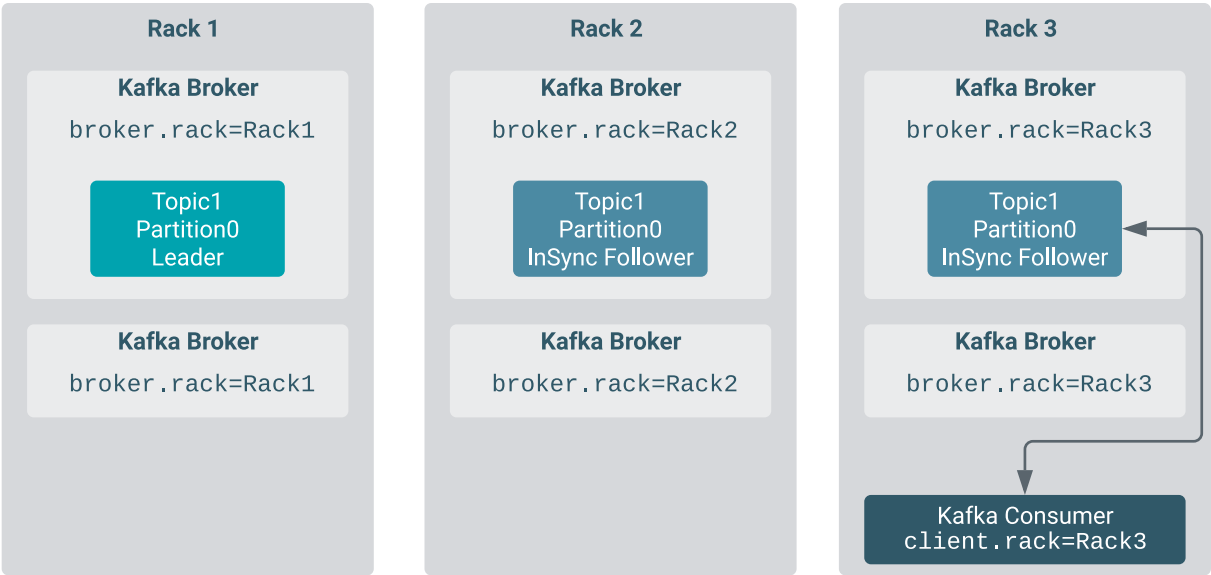

Follower fetching in multi-level deployments

Follower fetching is also supported if multi-level rack awareness is enabled for the brokers. When Kafka brokers are running in multi-level rack-aware mode, a multi-level rack-aware ReplicaSelector is automatically installed. This selector ensures that if a consumer that has a multi-level rack ID, the closest replica is selected from the multi-level hierarchy.

Related Information

Configuring rack awareness for Kafka consumers

Configuring multi-level rack awareness for consumers

Rack awareness for Kafka producers

Learn about rack awareness for Kafka producers.

Compared to brokers or consumers, there are no producer specific rack-awareness features or toggles that you can enable. However, in a deployment where rack awareness is an important factor, you can make configuration changes so that producers make use of rack awareness and have messages replicated to multiple racks.

Specifically, Cloudera recommends a configuration that ensures that the produced messages are replicated to at least two different racks before the messages are considered to be successful. This involves configuring acks to all in the producer configuration and setting up min.insync.replicas for the topics in a way that ensures a minimum of two racks get the message before the produce request is considered successful.

The configuration of the acks property is fixed. If you want to make your producers rack aware, the property must be set to all no matter the cluster topology or deployment.

The exact value you set for min.insync.replicas on the other hand depends on your cluster deployment. Specifically, the min.insync.replicas value you must set will depend on the number of racks, brokers, and the replication factor of your topics. Cloudera recommends that you exercise caution and review the following examples to better understand configuration.

For example, consider a Cloudera recommended deployment that has three racks with topic replication set to 3. In a case like this, a min.insync.replicas setting of 2 ensures that you always have data written to at least two different racks even if one replica is lagging.

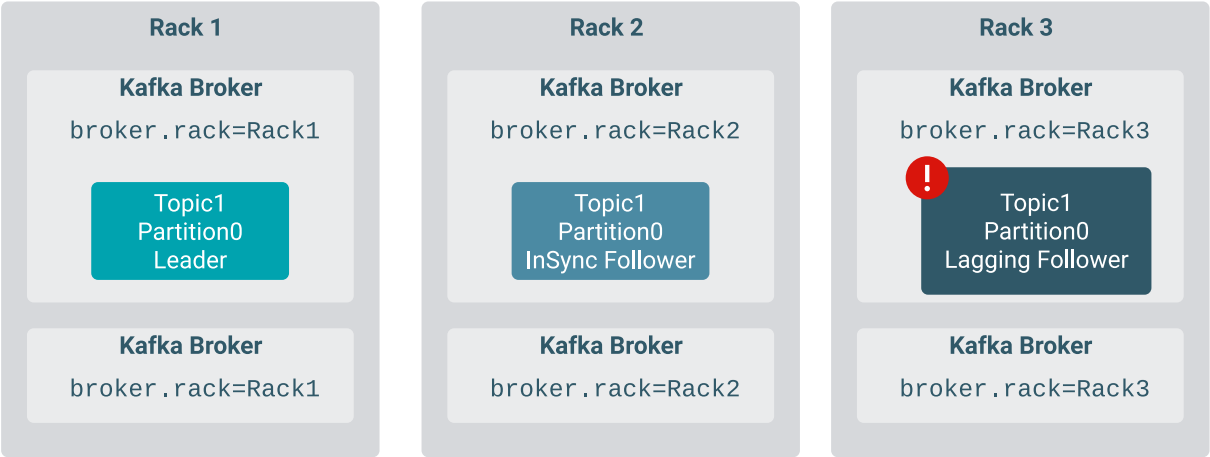

Understand however, that setting min.insync.replicas to 2 does not universally work for all deployments and may not guarantee that you always have your produced message in at least two racks. Configuration depends on the number of replicas, as well as the number of racks and brokers.

If you have more replicas and brokers than racks, you will have at least two replicas in the same rack. In a case like this, setting min.insync.replicas to 2 is not sufficient, a partition might become unavailable under certain circumstances.

For example, assume you have three racks with topic replication factor set to 4, meaning that there are a total of four replicas. Additionally, assume that only two of the replicas are in the in-sync replica set (ISR), the leader and one of the followers, and both are located in the same rack. The other two replicas are lagging. Unclean leader election is disabled to avoid data loss.

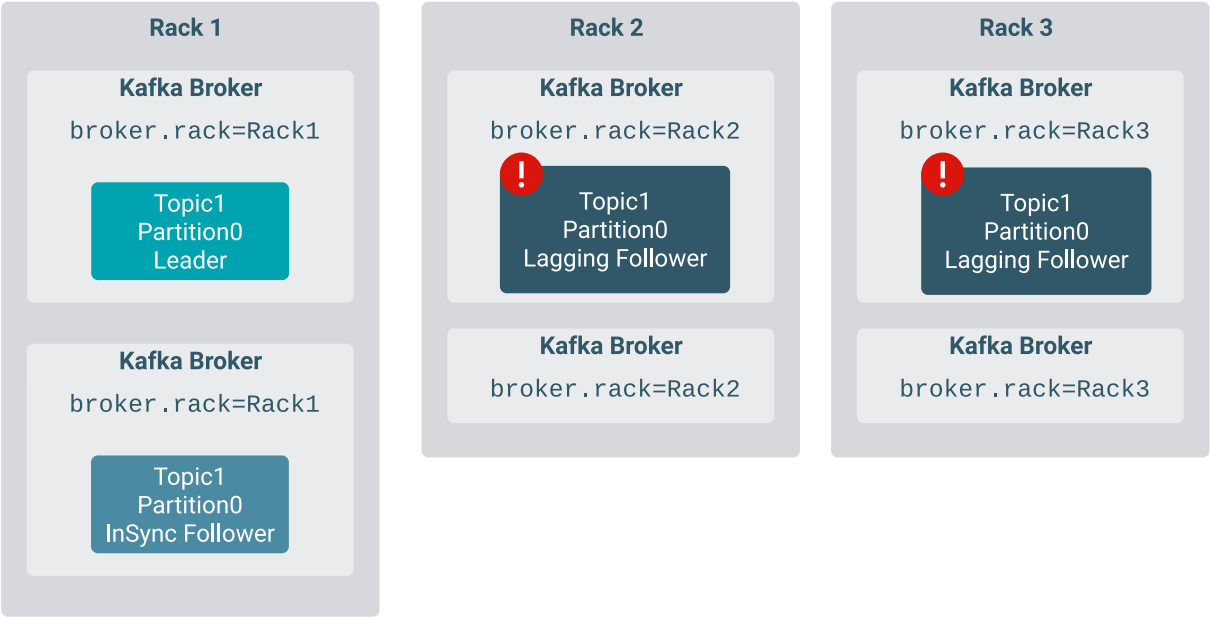

When the leader and the in-sync follower (located in the same rack) successfully append a produced message to the log, message production is considered successful. The leader does not wait for acknowledgement from the lagging replicas. This is because acks=all only guarantees that the leader waits for the replicas that are in the ISR (including itself). This means that while the latest messages are available on two brokers, both are located on the same rack. If the rack goes down at the same time or shortly after production is successful, the partition will become unavailable as only the two lagging replicas remain, which cannot become leaders.

In cases like this, a correct value for min.insync.replicas would be 3 instead of 2 as three ISRs would guarantee that messages are produced to at least two different racks.

Related Information

Configuring rack awareness for Kafka producers

Kafka KRaft [Technical Preview]

Apache Kafka Raft (KRaft) is a consensus protocol used for metadata management that was developed as a replacement for Apache ZooKeeper. Using KRaft for managing Kafka metadata instead of ZooKeeper offers various benefits including a simplified architecture and a reduced operational footprint. Learn more about what KRaft is, how it works, and how it compares to ZooKeeper deployments.

Note: Kafka KRaft is available in this version of Cloudera but is not ready for production deployment. Cloudera encourages you to explore this technical preview feature in non-production environments and provide feedback on your experiences through the Cloudera Community Forums. For more information regarding KRaft limitations and unsupported features, see Known Issues in Apache Kafka.

In traditional Kafka clusters, Kafka's metadata (including broker metadata, configurations, and client quotas) is stored in ZooKeeper. The brokers access this data in ZooKeeper through the broker that is designated as the active controller. This is the traditional way of how Kafka operates.

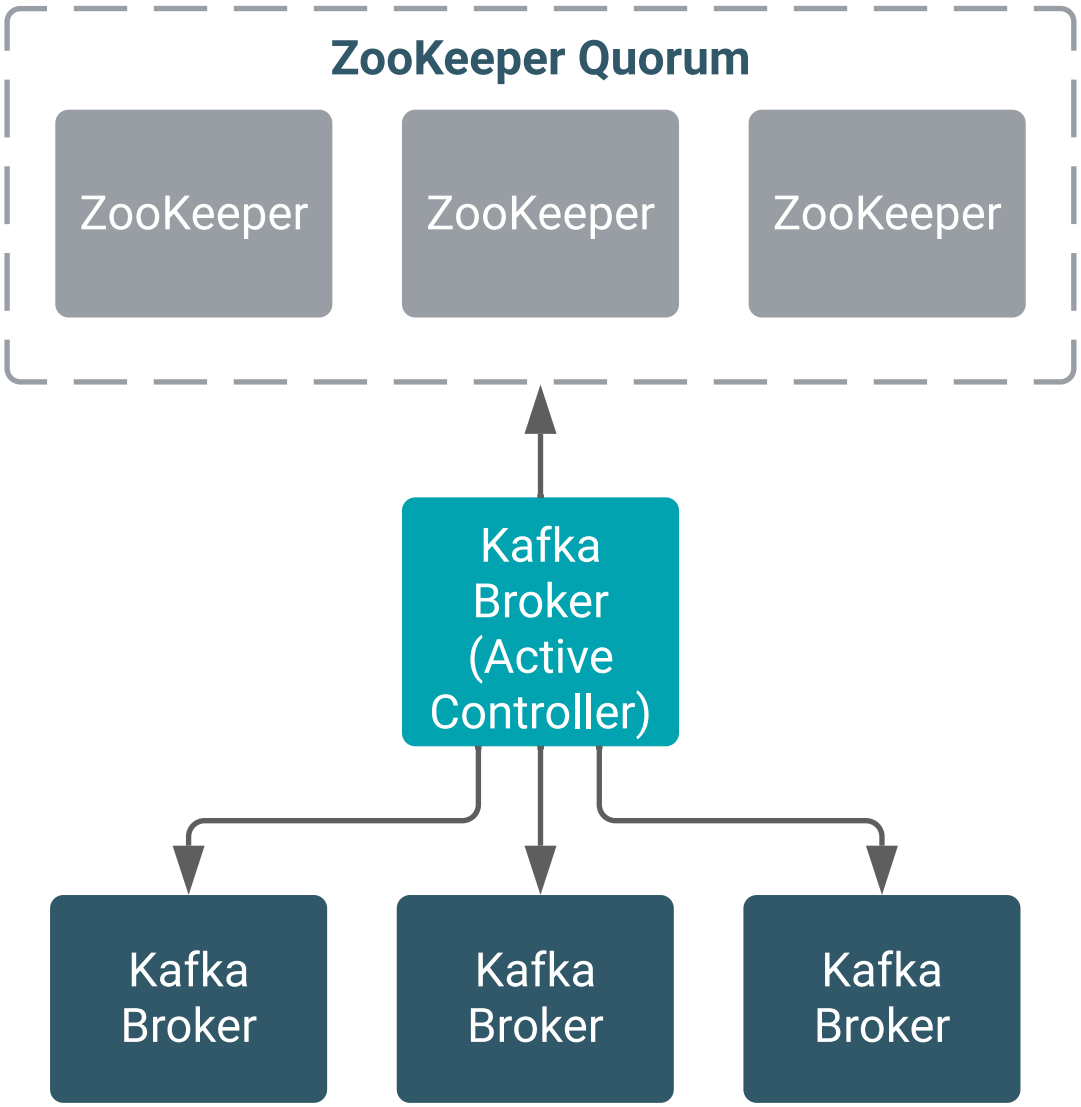

In the controller quorum mode, ZooKeeper is replaced with Kafka's own consensus implementation called Kafka Raft, or KRaft for short. KRaft is based on the Raft algorithm with some changes that are otherwise native in Kafka.

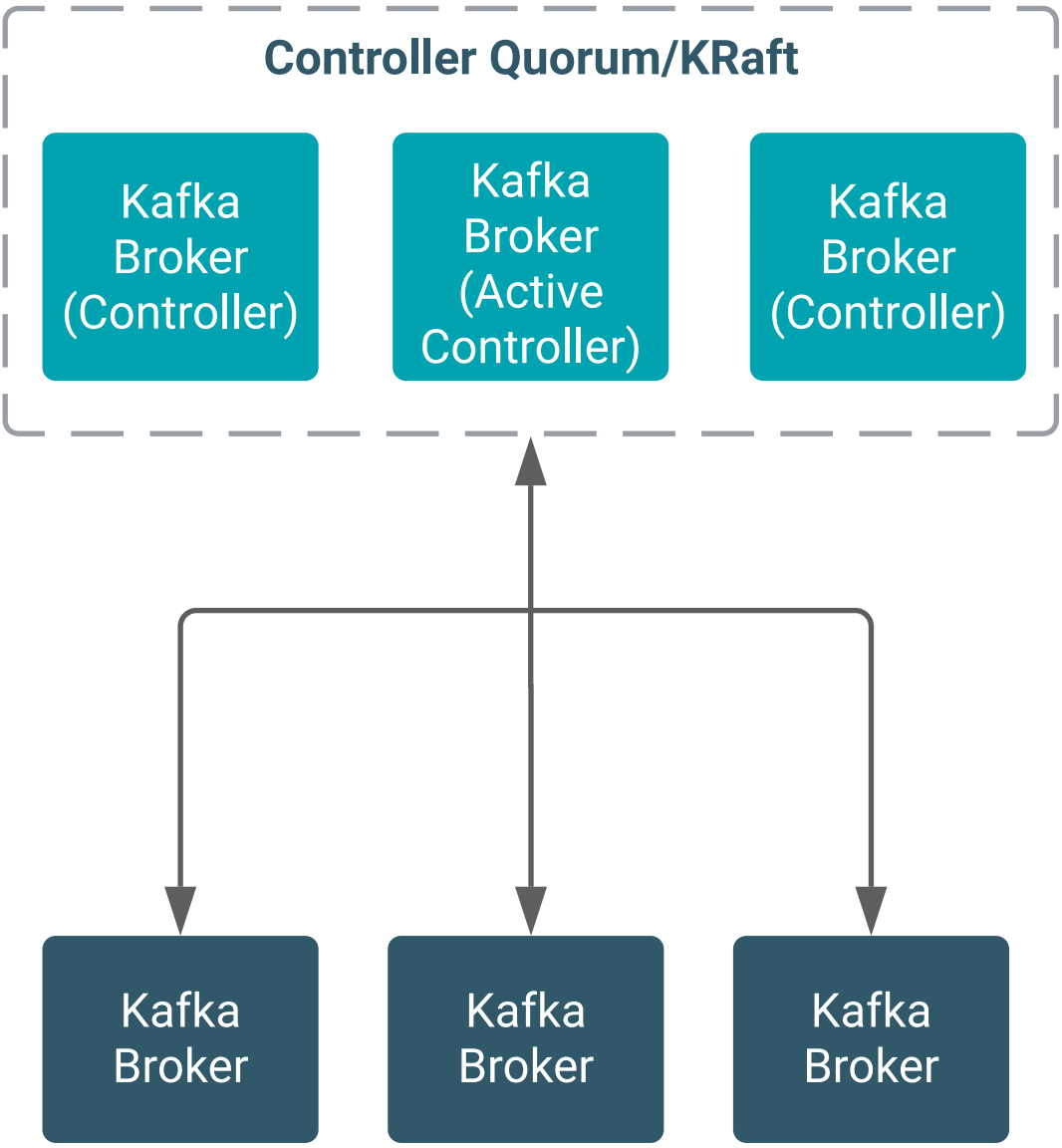

In this mode, a set of specialized brokers, called controllers, are deployed. These controllers form a cluster and are responsible for storing and maintaining Kafka metadata. The quorum of controllers ensures that the metadata is available at all times and that it is consistent. Additionally, one of the controllers is designated as the active controller (also known as the leader), which coordinates metadata changes with the brokers. The controllers store the metadata in a Kafka topic that can be found on each node.

Compared to the ZooKeeper, KRaft has a number of advantages including:

- **Fast and efficient metadata storage and propagation**

  With ZooKeeper, cluster updates are committed to ZooKeeper through the active controller. In KRaft, updates are sent directly to the controller quorum, eliminating one hop in the chain. In addition, because metadata is stored in a Kafka topic, both brokers and controllers can easily replicate and store the data locally. Both of these aspects of KRaft can help in substantially improving Kafka's performance.
- **Simplified, deployment, setup, and management**

  KRaft eliminates the need for a ZooKeeper service. This means that you only need to manage and maintain a single service, which is Kafka.

The Raft algorithm

Learn the basics of the Raft consensus algorithm, which KRaft is based on.

In many distributed systems there is a need for consensus. For example, consensus is needed to decide what is a distributed configuration's value, which servers are active, or which ones are passive. This type of information is critical for a distributed actor. Therefore, the information must be stored in a way that it is accessible reliably at all times.

There are many iterations to a safe consensus algorithm that ensure that all of the requirements above are met. A more modern version of such an algorithm is the Raft algorithm.

Note: The most prominent component to implement a consensus system is ZooKeeper, which has been used in many projects. Even though both provide highly reliable distributed coordination, ZooKeeper and Raft are somewhat different.

Raft uses timers and periodic heartbeats to keep track of alive nodes. As long as the majority of nodes are functioning, clients can publish and access the data. Raft replicates data on all of the Raft quorum nodes. Additionally a leader is elected randomly on first startup. When data is produced, it is pushed to peer nodes with the periodic heartbeats. Follower nodes acknowledge this data. A request is committed only when the majority of nodes confirm it.

Raft can ensure that partitioned controller quorums operate correctly. That is, if a partition does not have the majority, then data sent to those nodes is not committed. When partitions merge, the majority wins.

In case of a leader failure, remaining nodes hold a vote. A new leader is chosen from the remaining pool of nodes. After the new leader is elected, it becomes the origin of the data. Other nodes switch over and replicate from the new leader.

Voting is timeout based. If a node doesn't get a heartbeat within the session timeout, it sends out vote requests. Other nodes do the same, but at the same time accept incoming votes. Eventually, the first node who sent out requests wins and is elected the new leader.

For more information on Raft, see https://raft.github.io/.

KRaft metadata management

Learn how Kafka metadata is stored and managed in KRaft.

In KRaft mode, metadata is stored in a dedicated internal Kafka topic called __cluster_metadata. This topic has a single partition that contains all information related to the current state of the Kafka service. The leader of the single partition is the active controller. The active controller is responsible for writing metadata changes to the __cluster_metadata topic.

Both brokers and controllers fetch and replicate this topic to store it locally. Controllers are replica followers, meaning that they are part of the in-sync replica set (ISR) for the topic. Brokers are observers only. This means that while they replicate the topic, they are not part of the ISR. Both brokers and controllers periodically commit metadata from __cluster_metadata to their metadata cache.

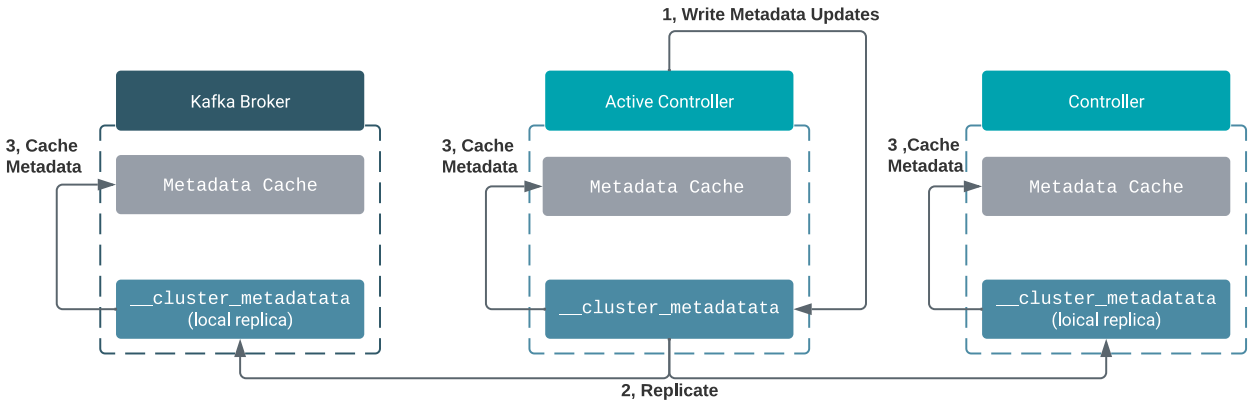

When a broker starts up, it reads the most recent changes available in its metadata cache. Afterward, it requests the changes from the controller that happened since the last update.

Next, the active controller responds with a series of deltas or the full state of the cluster. The full state of the cluster is only sent if the broker has no cached metadata (it's an empty broker starting for the first time) or if it is severely lagging behind. The broker then commits the updates it receives to its cache and the topic replica.

Going forward, the broker periodically asks for updates from the active controller. These periodic fetch requests that the broker sends also act as heartbeats.

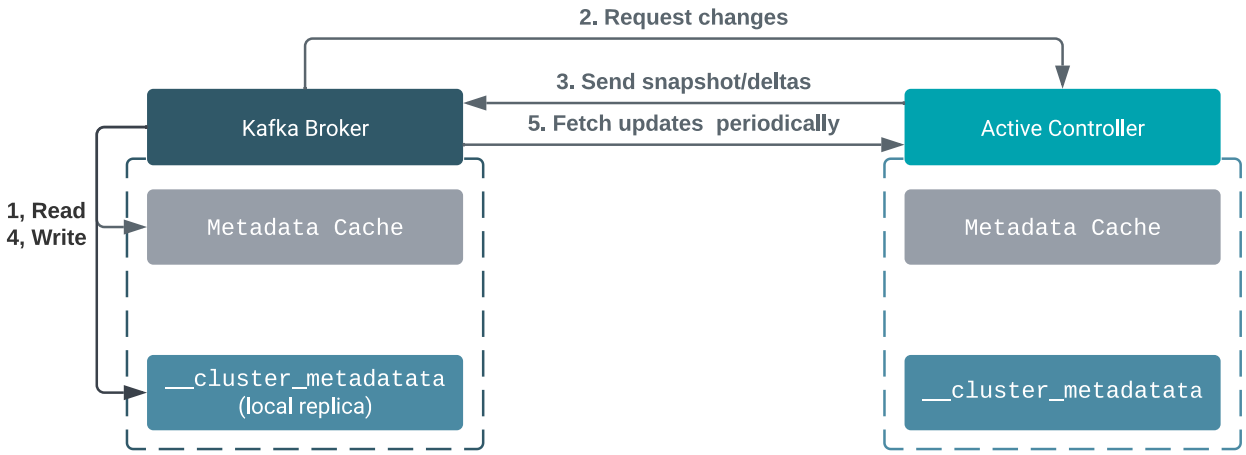

KRaft metadata replication and the HWM

Learn about the high watermark and how Kafka metadata is replicated in KRaft.

In KRaft mode, Kafka uses its own implementation of the Raft protocol. This implementation is based on Raft, but is heavily influenced by Kafka's log replication protocol.

One of the cornerstones of Kafka replication is the high watermark (HWM). The HWM defines the highest offset that has been replicated across all ISR replicas. It is maintained by the leader, but followers cache it and update it with each offset fetch so a consistent state is maintained across the cluster.

On leader change, all partitions truncate their log to the HWM (messages above are considered unsuccessful and the client retries). In general, the HWM must be larger than the log start offset and smaller than the log end offset.

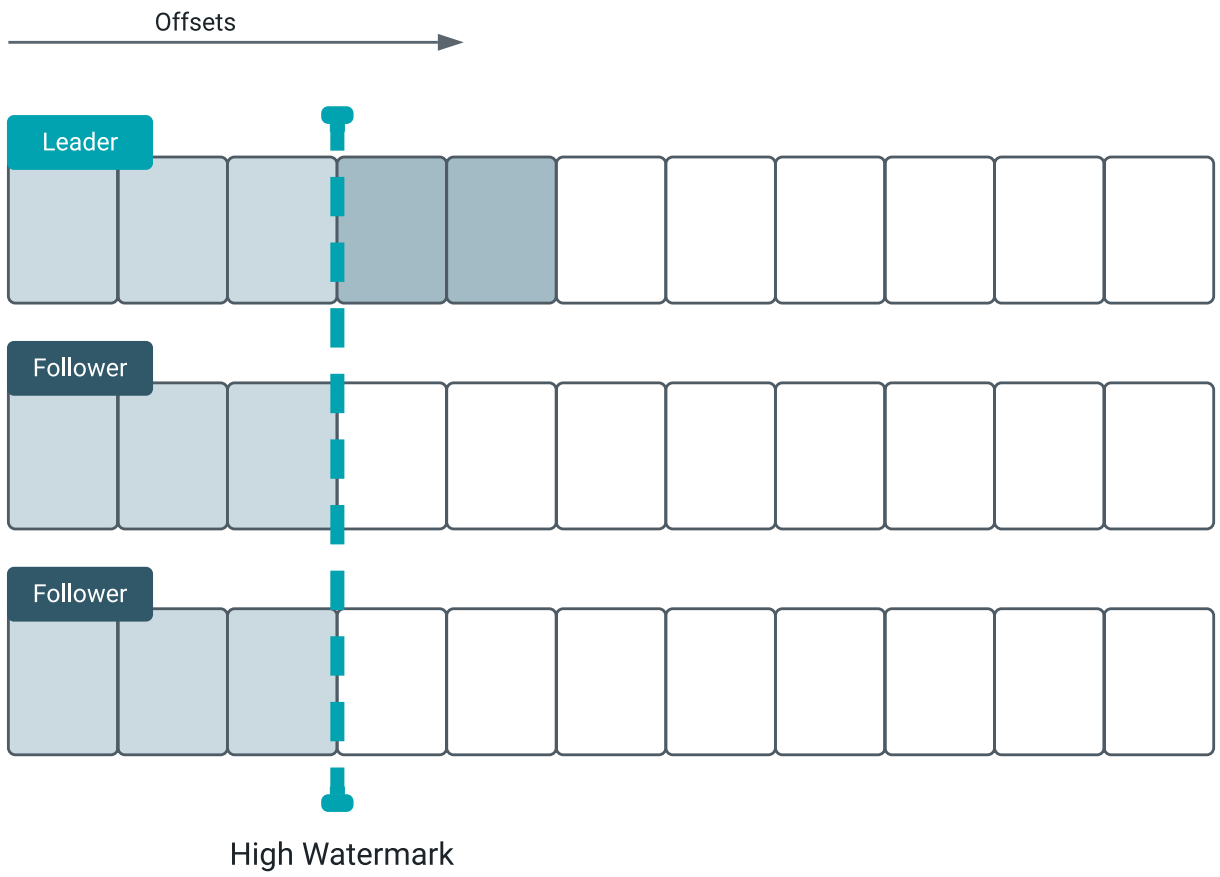

In KRaft, the concept of the HWM is slightly different. Instead of being the **highest offset** that is replicated across **all ISR** replicas, it is the highest offset that is replicated by the **majority** of replicas. This way Kafka's replication closely mimics the Raft algorithm's behavior.

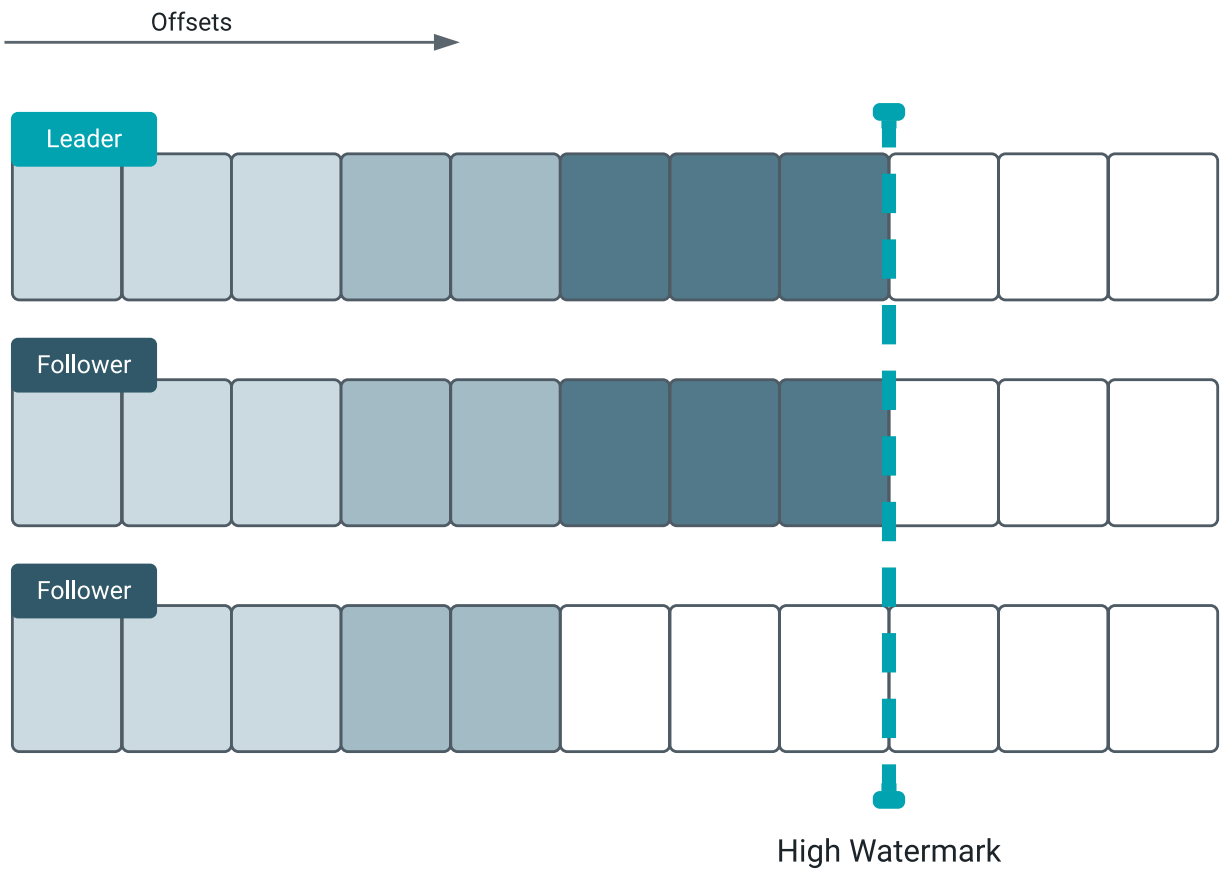

KRaft leader changes

Learn how leader changes are handled in KRaft.

When the active controller (the leader for the controllers and the __cluster_metadata topic) fails, a new leader must be elected. Otherwise, Kafka can not continue to function. Leader changes in KRaft work similarly to the original protocol.

On leader change, as according to the protocol, each node gets to vote. In KRaft each node can cast a single vote per epoch. Nodes always vote for the node that has the “longest” log. Longest in this case means highest epoch, or, if the epoch is identical, larger offset. The new leader is decided by a simple majority. That is, the node with the most votes is elected leader.

For example, assume that there are three controllers. Each of them has a replica of __cluster_metadata. Replica A was the leader, but due to a node failure it goes offline and falls out of the in-sync replica set (ISR).

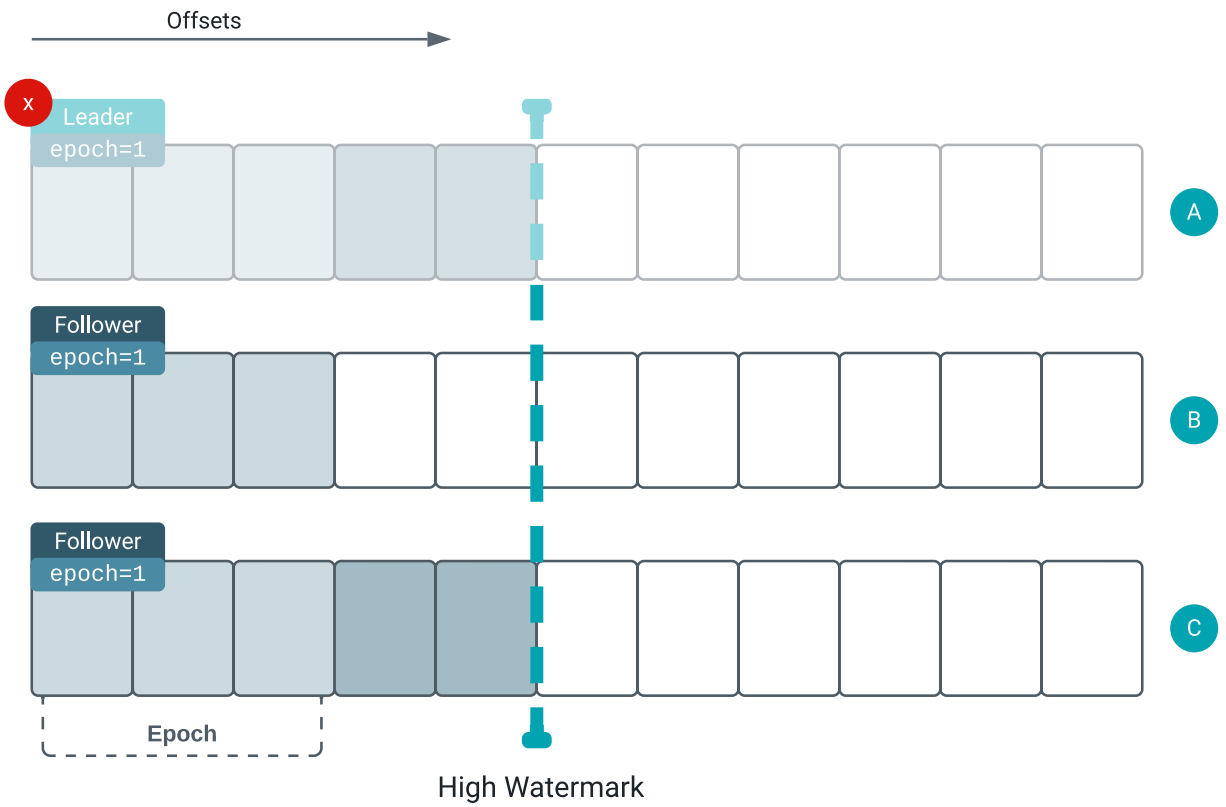

A new leader is required. Each candidate sends a vote request which contains the epoch and the offset. If the sender has a longer log than the recipient of the request, the recipient votes for the sender.

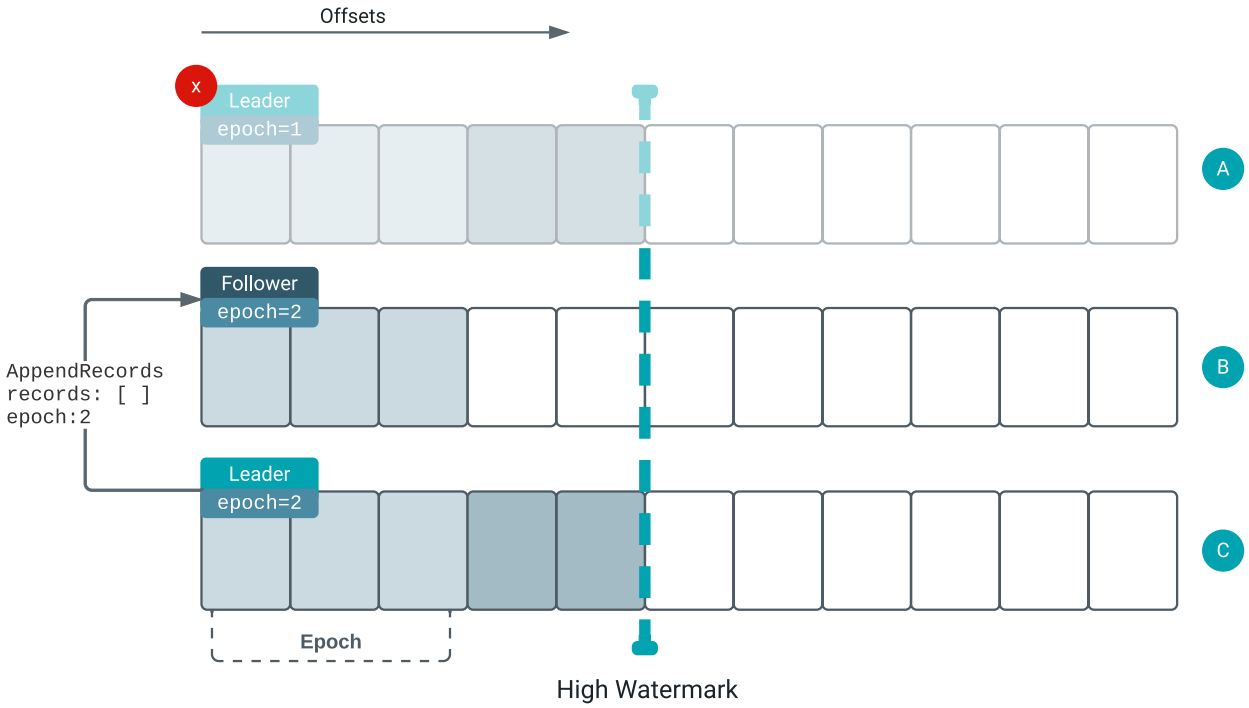

In this example the candidate (replica C) has an epoch count of 2, while the follower (replica B) only has an epoch count of 1. The candidate has the “longest” log. The follower, therefore, votes for the candidate and the candidate becomes the leader. If leadership is decided, the followers truncate back to the HWM and fetch the new leader’s data.

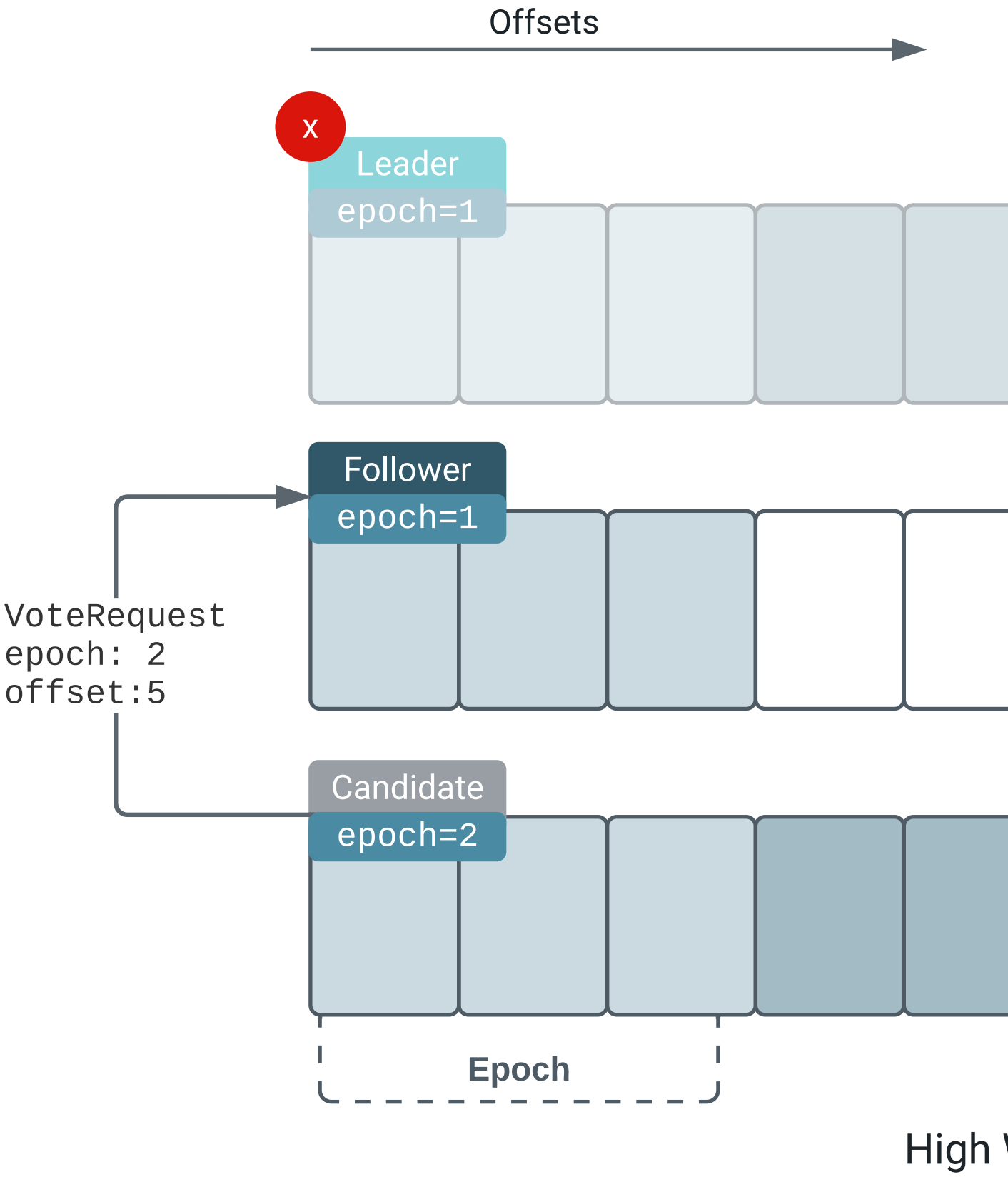
Offsets
x
Leader
epoch=1
Follower
epoch=1
VoteRequest
epoch: 2
offset:5
Candidate
epoch=2
Epoch
High

KRaft log reconciliation

Learn how logs are reconciled and how the high watermark (HWM) is incremented in KRaft.

When a leader change happens, the log must be reconciled. After a new leader is elected, followers send fetch requests to the leader. The fetch request contains the last fetched epoch and an offset. The epoch is initially 1, but increments with each leader change. Data is in fact stored with the epoch numbers. Therefore, the high watermark (HWM) is incremented only if the majority of nodes replicated the data with the highest epoch.

For example, assume that you have three controllers. Each of them has a replica of __cluster_metadata. Replica A, which is the leader, is in epoch 1, offset 3. Replica B, a follower, is caught up with the leader. The majority of the nodes replicated the data with the highest epoch. As a result, the HWM is moved to offset 3.

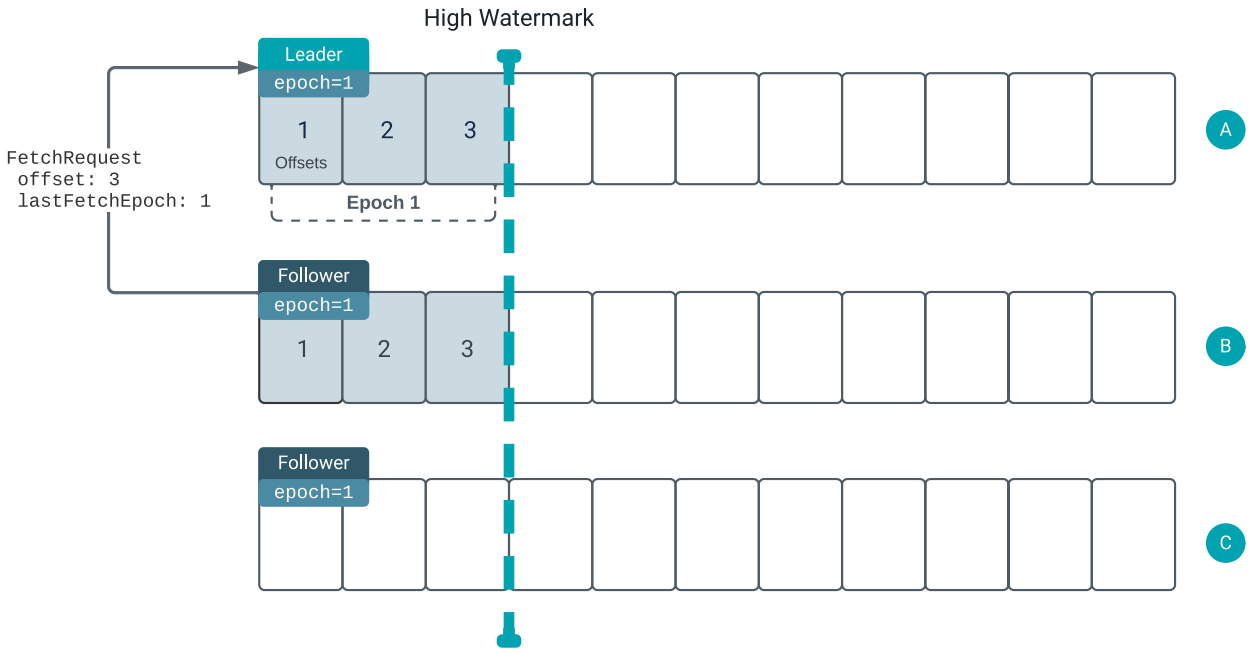

Let's look at a more complex example. Assume that you have the same setup, but more messages were committed and a number of leader changes happened. The cluster is in the following state.

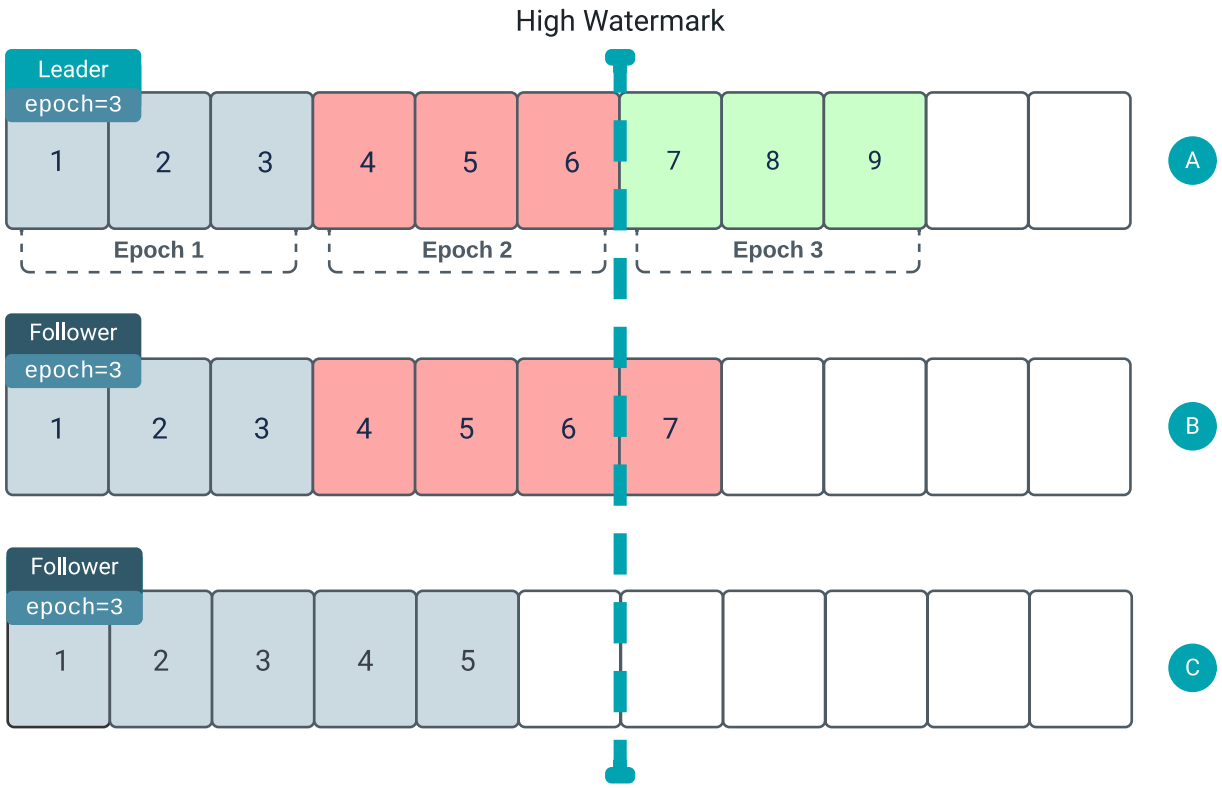

Notice how both followers (B and C) already accepted the leader's epoch, and that replica A and B have additional messages above the HWM. However, not all messages were replicated to the majority of nodes and the HWM is not incremented. In this scenario, the log must be reconciled before the HWM can be incremented. The reconciliation process is as follows.

First, followers send a fetch request to the leader. The fetch request contains the offset as well as the last fetched epoch. For replica B, the offset is 7, the last fetched epoch is 2.

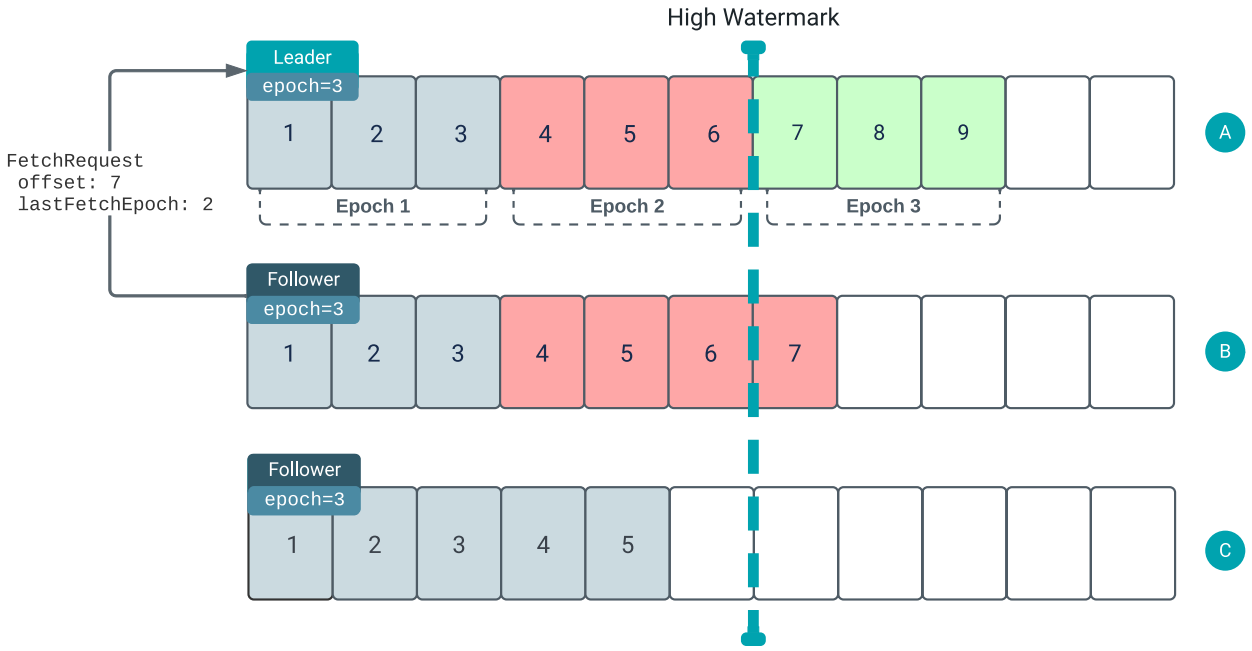

Based on the fetch request, the leader knows that the logs have diverged and sends a response that includes the diverging epoch and the end offset of that epoch. In this case, the diverging epoch is 2, the end offset is 6.

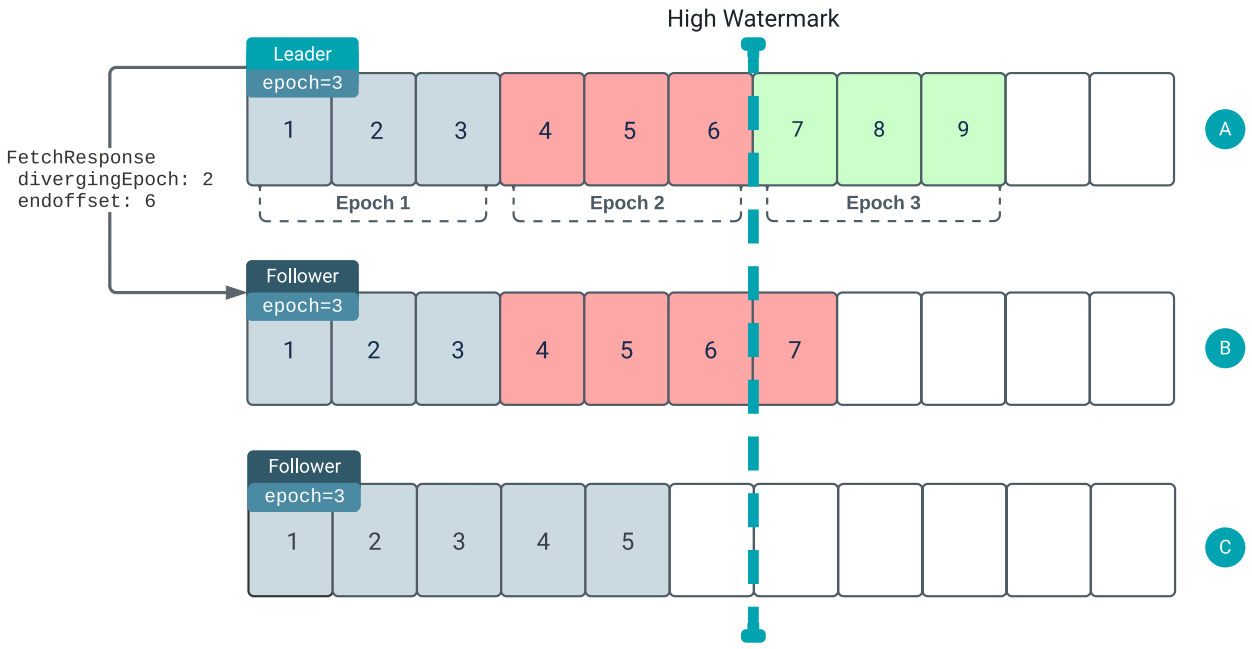

The information included in the fetch responses enables the follower to truncate its log to the correct position.

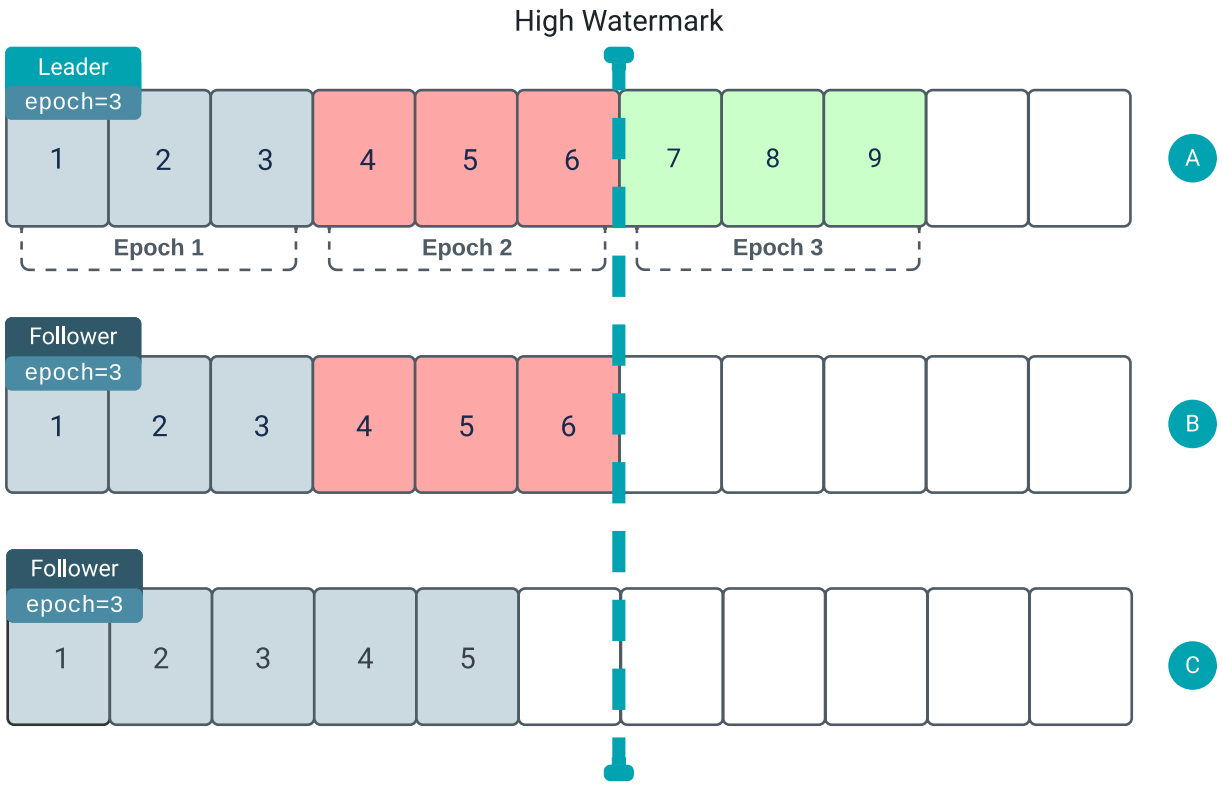

After truncating the log, the follower can continue to replicate any new messages from the leader. Once all new messages are replicated to the majority of nodes the HWM is incremented.

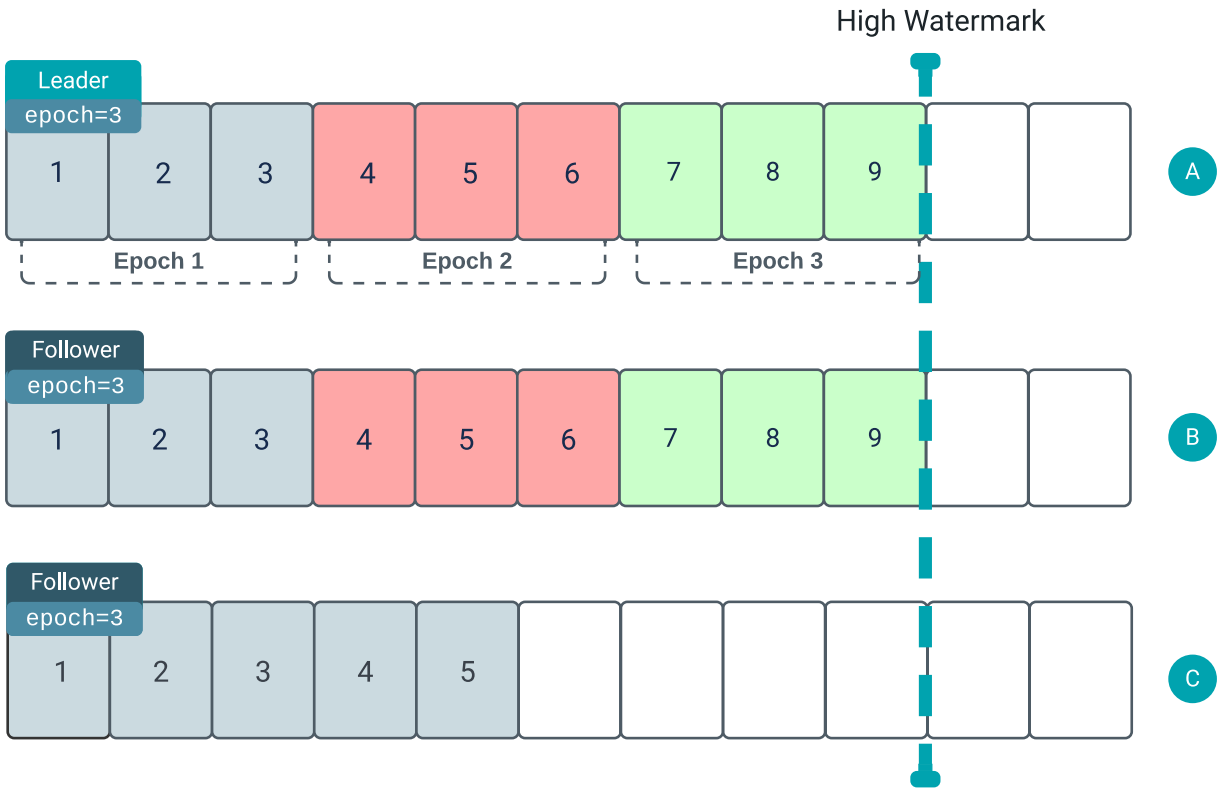

The same process also happens for replica C. Ultimately, it catches up to the leader as well.

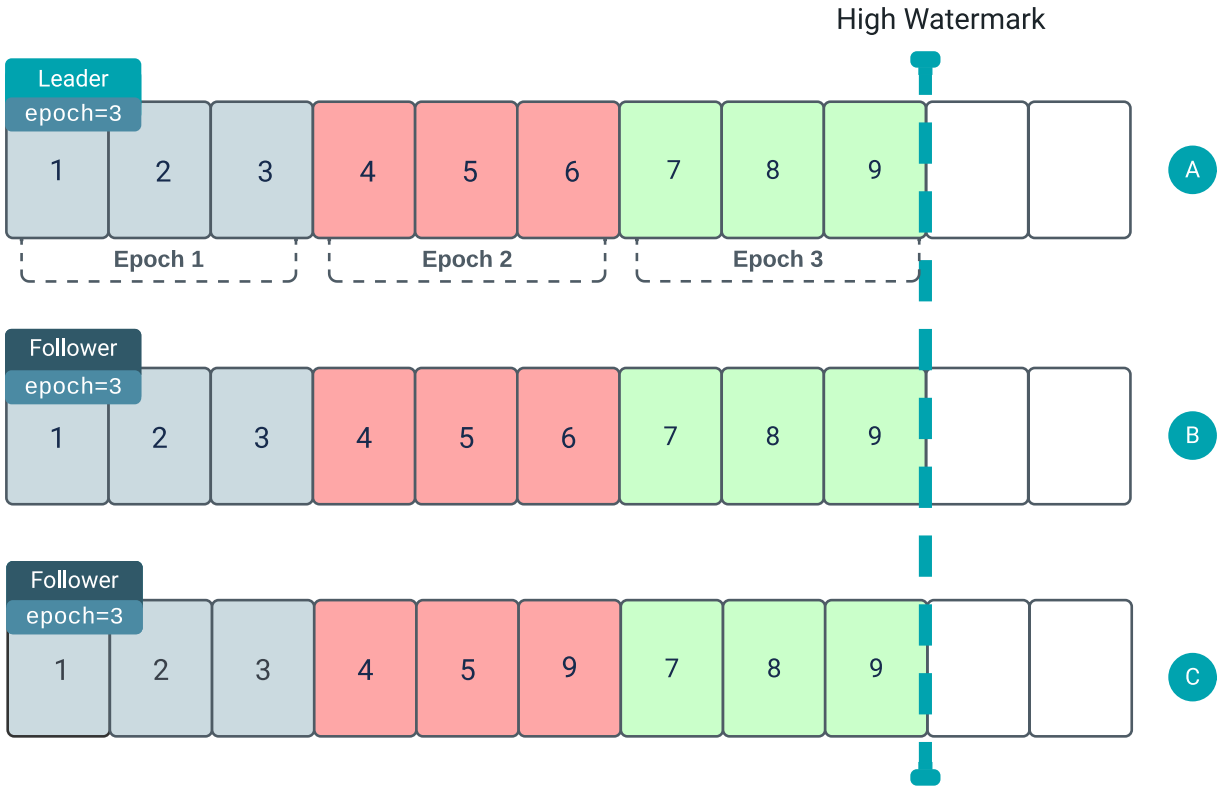

KRaft broker state machines

Brokers in KRaft mode have four states. Learn what these states are and when brokers enter each state.

The states are as follows:

- **Offline**: The broker hasn't started. The broker remains in this state until the local logs from the file system are loaded.
- **Fenced**: The broker has started, but can't receive requests from clients. Replicas aren't in sync with the cluster. The broker stays in this state until it catches up with the current metadata of the cluster.
- **Running**: The broker has started and is ready to receive requests from clients.
- **Stopping**: The broker is still running, but it is migrating the partition leaders to other brokers.

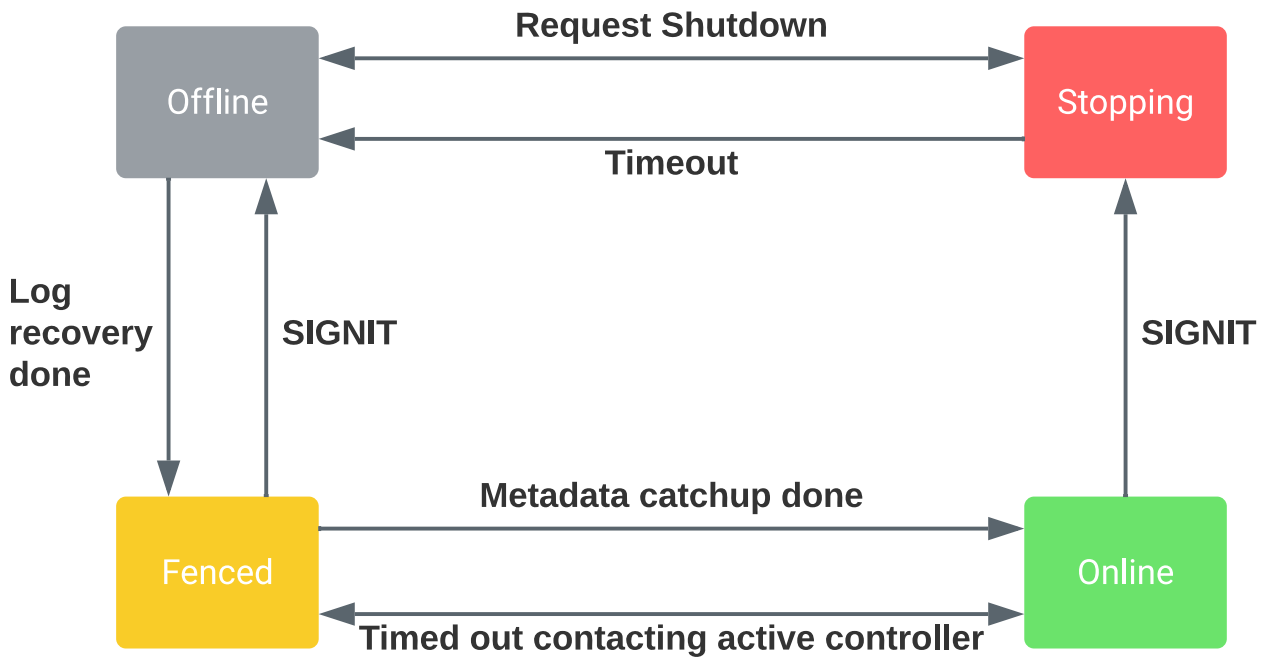

Kafka FAQ

A collection of frequently asked questions on the topic of Kafka.

Basics

A collection of frequently asked questions on the topic of Kafka aimed for beginners.

What is Kafka?

Kafka is a streaming message platform. Breaking it down a bit further:

"Streaming": Lots of messages (think tens or hundreds of thousands) being sent frequently by publishers ("producers"). Message polling occurring frequently by lots of subscribers ("consumers").

"Message": From a technical standpoint, a key value pair. From a non-technical standpoint, a relatively small number of bytes (think hundreds to a few thousand bytes).

If this isn't your planned use case, Kafka may not be the solution you are looking for. Contact your favorite Cloudera representative to discuss and find out. It is better to understand what you can and cannot do upfront than to go ahead based on some enthusiastic arbitrary vendor message with a solution that will not meet your expectations in the end.

What is Kafka designed for?

Kafka was designed at LinkedIn to be a horizontally scaling publish-subscribe system. It offers a great deal of configurability at the system- and message-level to achieve these performance goals. There are well documented cases that showcase how well Kafka can scale when everything is done right. One such example is LinkedIn.

What is Kafka not well fitted for (or what are the tradeoffs)?

It's very easy to get caught up in all the things that Kafka can be used for without considering the tradeoffs. Kafka configuration is also not automatic. You need to understand each of your use cases to determine which configuration properties can be used to tune (and retune!) Kafka for each use case.

Some more specific examples where you need to be deeply knowledgeable and careful when configuring are:

- Using Kafka as your microservices communication hub

  Kafka can replace both the message queue and the services discovery part of your software infrastructure. However, this is generally at the cost of some added latency as well as the need to monitor a new complex system (i.e. your Kafka cluster).
- Using Kafka as long-term storage

  While Kafka does have a way to configure message retention, it's primarily designed for low latency message delivery. Kafka does not have any support for the features that are usually associated with filesystems (such as metadata or backups). As such, using some form of long-term ingestion, such as HDFS, is recommended instead.
- Using Kafka as an end-to-end solution

  Kafka is only part of a solution. There are a lot of best practices to follow and support tools to build before you can get the most out of it (see this wise LinkedIn post).
- Deploying Kafka without the right support

  Uber has given some numbers for their engineering organization. These numbers could help give you an idea what it takes to reach that kind of scale: 1300 microservers, 2000 engineers.

Where can I get a general Kafka overview?

Read Kafka Introduction and Kafka Architecture, which cover the basics and design of Kafka. This should serve as a good starting point. If you have any remaining questions, come to this FAQ or talk to your favorite Cloudera representative about training or a best practices deep dive.

Where does Kafka fit well into an Analytic Database solution?

Analytic Database deployments benefit from Kafka by utilizing it for data ingest. Data can then populate tables for various analytics workloads. For ad hoc BI the real-time aspect is less critical, but the ability to utilize the same data used in real time applications, in BI and analytics as well, is a benefit that Cloudera's platform provides, as you will have Kafka for both purposes, already integrated, secured, governed and centrally managed.

Where does Kafka fit well into an Operational Database solution?

Kafka is commonly used in the real-time, mission-critical world of Operational Database deployments. It is used to ingest data and allow immediate serving to other applications and services through Kudu or HBase. The benefit of utilizing Kafka in the Cloudera platform for Operational Database is the integration, security, governance and central management. You avoid the risks and costs of siloed architecture and "yet another solution" to support.

What is a Kafka consumer?

If Kafka is the system that stores messages, then a consumer is the part of your system that reads those messages from Kafka.

While Kafka does come with a command line tool that can act as a consumer, practically speaking, you will most likely write Java code using the KafkaConsumer API for your production system.

What is a Kafka producer?

While consumers read from a Kafka cluster, producers write to a Kafka cluster.

Similar to the consumer (see previous question), your producer is also custom Java code for your particular use case.

Your producer may need some tuning for write performance and SLA guarantees, but will generally be simpler (fewer error cases) to tune than your consumer.

What functionality can I call in my Kafka Java code?

The best way to get more information on what functionality you can call in your Kafka Java code is to look at the Java documents. And read very carefully!

What's a good size of a Kafka record if I care about performance and stability?

There is an older blog post from 2014 from LinkedIn titled: Benchmarking Apache Kafka: 2 Million Writes Per Second (On Three Cheap Machines). In the "Effect of Message Size" section, you can see two charts which indicate that Kafka throughput starts being affected at a record size of 100 bytes through 1000 bytes and bottoming out around 10000 bytes. In general, keeping topics specific and keeping message sizes deliberately small helps you get the most out of Kafka.

Excerpting from Deploying Apache Kafka: A Practical FAQ:

> How to send large messages or payloads through Kafka?
>
> Cloudera benchmarks indicate that Kafka reaches maximum throughput with message sizes of around 10 KB. Larger messages show decreased throughput. However, in certain cases, users need to send messages much larger than 10 KB.
>
> If the message payload sizes are in the order of 100s of MB, consider exploring the following alternatives:
>
> - If shared storage is available (HDFS, S3, NAS), place the large payload on shared storage and use Kafka just to send a message with the payload location.
> - Handle large messages by chopping them into smaller parts before writing into Kafka, using a message key to make sure all the parts are written to the same partition so that they are consumed by the same Consumer, and re-assembling the large message from its parts when consuming.

Where can I get Kafka training?

Cloudera Educational Services offers various courses on Kafka. For a basic training that gives an overview of Kafka, its architecture, messages, producers and consumers (clients), as well as command line tools, see Apache Kafka Basics. Additionally, if you are looking for more in depth training on Kafka, Kafka security, Kafka Connect, Streams Messaging Manager, and so on, search for *Kafka* in the course catalog on http://education.cloudera.com.

Use cases

A collection of frequently asked questions on the topic of Kafka aimed for advanced users.

Like most Open Source projects, Kafka provides a lot of configuration options to maximize performance. In some cases, it is not obvious how best to map your specific use case to those configuration options. We attempt to address some of those situations.

What can I do to ensure that I never lose a Kafka event?

This is a simple question which has lots of far-reaching implications for your entire Kafka setup. A complete answer includes the next few related FAQs and their answers.

What is the recommended node hardware for best reliability?

Operationally, you need to make sure your Kafka cluster meets the following hardware setup:

- Have a 3 or 5 node cluster only running Zookeeper (higher only necessary at largest scales).
- Have at least a 3 node cluster only running Kafka.
- Have the disks on the Kafka cluster running in RAID 10. (Required for resiliency against disk failure.)
- Have sufficient memory for both the Kafka and Zookeeper roles in the cluster. (Recommended: 4GB for the broker, the rest of memory automatically used by the kernel as file cache.)
- Have sufficient disk space on the Kafka cluster.
- Have a sufficient number of disks to handle the bandwidth requirements for Kafka and Zookeeper.
- You need a number of nodes greater than or equal to the highest replication factor you expect to use.

What are the network requirements for best reliability?

Kafka expects a reliable, low-latency connection between the brokers and the Zookeeper nodes:

- The number of network hops between the Kafka cluster and the Zookeeper cluster is relatively low.
- Have highly reliable network services (such as DNS).

What are the system software requirements for best reliability?

Assuming you’re following the recommendations of the previous two questions, the actual system outside of Kafka must be configured properly.

1. The kernel must be configured for maximum I/O usage that Kafka requires.
   a. Large page cache
   b. Maximum file descriptions
   c. Maximum file memory map limits
2. Kafka JVM configuration settings:
   a. Brokers generally don’t need more than 4GB-8GB of heap space.
   b. Run with the +G1GC garbage collection using Java 8 or later.

How can I configure Kafka to ensure that events are stored reliably?

The following recommendations for Kafka configuration settings make it extremely difficult for data loss to occur.

- Producer
  - block.on.buffer.full=true
  - retries=Long.MAX_VALUE
  - acks=all
  - max.in.flight.requests.per.connections=1
  - Remember to close the producer when it is finished or when there is a long pause.
- Broker
  - Topic replication.factor >= 3
  - Min.insync.replicas = 2
  - Disable unclean leader election
- Consumer
  - Disable enable.auto.commit
  - Commit offsets after messages are processed by your consumer client(s).

If you have more than 3 hosts, you can increase the broker settings appropriately on topics that need more protection against data loss.

Once I've followed all the previous recommendations, my cluster should never lose data, right?

Kafka does not ensure that data loss never occurs. There are the following tradeoffs:

- Throughput vs. reliability. For example, the higher the replication factor, the more resilient your setup will be against data loss. However, to make those extra copies takes time and can affect throughput.
- Reliability vs. free disk space. Extra copies due to replication use up disk space that would otherwise be used for storing events.

Beyond the above design tradeoffs, there are also the following issues:

- To ensure events are consumed you need to monitor your Kafka brokers and topics to verify sufficient consumption rates are sustained to meet your ingestion requirements.
- Ensure that replication is enabled on any topic that requires consumption guarantees. This protects against Kafka broker failure and host failure.
- Kafka is designed to store events for a defined duration after which the events are deleted. You can increase the duration that events are retained up to the amount of supporting storage space.
- You will always run out of disk space unless you add more nodes to the cluster.

My Kafka events must be processed in order. How can I accomplish this?

After your topic is configured with partitions, Kafka sends each record (based on key/value pair) to a particular partition based on key. So, any given key, the corresponding records are "in order" within a partition.

For global ordering, you have two options:

- Your topic must consist of one partition (but a higher replication factor could be useful for redundancy and failover). However, this will result in very limited message throughput.
- You configure your topic with a small number of partitions and perform the ordering after the consumer has pulled data. This does not result in guaranteed ordering, but, given a large enough time window, will likely be equivalent.

Conversely, it is best to take Kafka's partitioning design into consideration when designing your Kafka setup rather than rely on global ordering of events.

How do I size my topic? Alternatively: What is the "right" number of partitions for a topic?

Choosing the proper number of partitions for a topic is the key to achieve a high degree of parallelism with respect to writes and reads and to distribute load. Evenly distributed load over partitions is a key factor to have good throughput (avoid hot spots). Making a good decision requires estimation based on the desired throughput of producers and consumers per partition.

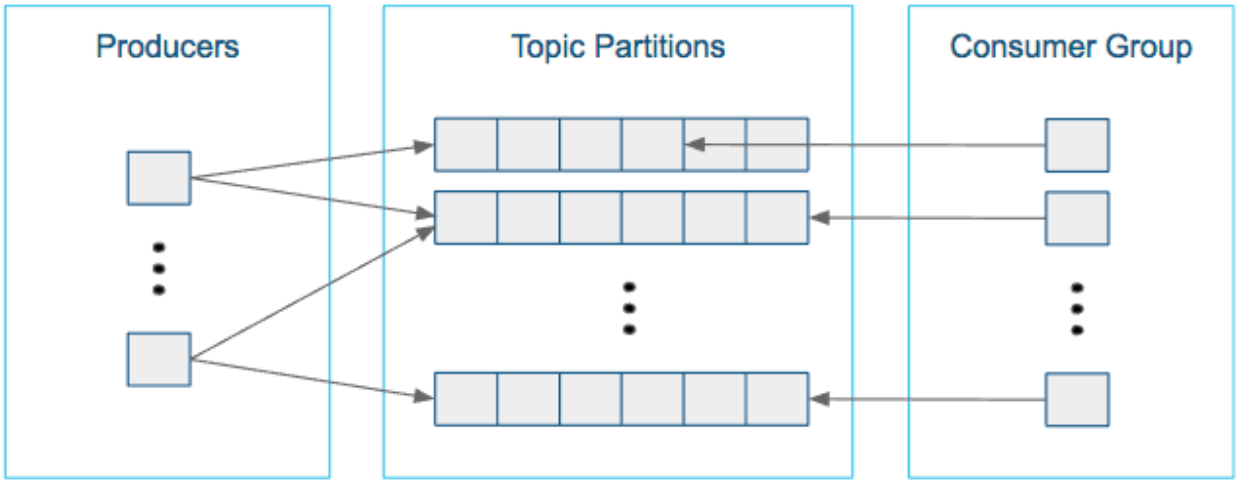

For example, if you want to be able to read 1 GB/sec, but your consumer is only able to process 50 MB/sec, then you need at least 20 partitions and 20 consumers in the consumer group. Similarly, if you want to achieve the same for producers, and 1 producer can only write at 100 MB/sec, you need 10 partitions. In this case, if you have 20 partitions, you can maintain 1 GB/sec for producing and consuming messages. You should adjust the exact number of partitions to number of consumers or producers, so that each consumer and producer achieve their target throughput.

So a simple formula could be:

```
#Partitions = max(NP, NC)
```

where:

- NP is the number of required producers determined by calculating: TT/TP
- NC is the number of required consumers determined by calculating: TT/TC
- TT is the total expected throughput for our system
- TP is the max throughput of a single producer to a single partition
- TC is the max throughput of a single consumer from a single partition

This calculation gives you a rough indication of the number of partitions. It's a good place to start. Keep in mind the following considerations for improving the number of partitions after you have your system in place:

- The number of partitions can be specified at topic creation time or later.
- Increasing the number of partitions also affects the number of open file descriptors. So make sure you set file descriptor limit properly.
- Reassigning partitions can be very expensive, and therefore it's better to over- than under-provision.
- Changing the number of partitions that are based on keys is challenging and involves manual copying.
- Reducing the number of partitions is not currently supported. Instead, create a new a topic with a lower number of partitions and copy over existing data.
- Metadata about partitions are stored in ZooKeeper in the form of znodes. Having a large number of partitions has effects on ZooKeeper and on client resources:
  - Unneeded partitions put extra pressure on ZooKeeper (more network requests), and might introduce delay in controller and/or partition leader election if a broker goes down.
  - Producer and consumer clients need more memory, because they need to keep track of more partitions and also buffer data for all partitions.
- As guideline for optimal performance, you should not have more than 4000 partitions per broker and not more than 200,000 partitions in a cluster.

Make sure consumers don't lag behind producers by monitoring consumer lag. To check consumers' position in a consumer group (that is, how far behind the end of the log they are), use the following command:

```
$ kafka-consumer-groups --bootstrap-server BROKER_ADDRESS --describe --gr
oup CONSUMER_GROUP --new-consumer
```

How can I scale a topic that's already deployed in production?

Recall the following facts about Kafka:

- When you create a topic, you set the number of partitions. The higher the partition count, the better the parallelism and the better the events are spread somewhat evenly through the cluster.
- In most cases, as events go to the Kafka cluster, events with the same key go to the same partition. This is a consequence of using a hash function to determine which key goes to which partition.

Now, you might assume that scaling means increasing the number of partitions in a topic. However, due to the way hashing works, simply increasing the number of partitions means that you will lose the "events with the same key go to the same partition" fact.

Given that, there are two options:

1. Your cluster may not be scaling well because the partition loads are not balanced properly (for example, one broker has four very active partitions, while another has none). In those cases, you can use the kafka-reassign-partitions script to manually balance partitions.
2. Create a new topic with more partitions, pause the producers, copy data over from the old topic, and then move the producers and consumers over to the new topic. This can be a bit tricky operationally.

How do I rebalance my Kafka cluster?

This one comes up when new nodes or disks are added to existing nodes. Partitions are not automatically balanced. If a topic already has a number of nodes equal to the replication factor (typically 3), then adding disks does not help with rebalancing.

Using the kafka-reassign-partitions command after adding new hosts is the recommended method.

Caveats

There are several caveats to using this command:

- It is highly recommended that you minimize the volume of replica changes to make sure the cluster remains healthy. Say, instead of moving ten replicas with a single command, move two at a time.
- It is not possible to use this command to make an out-of-sync replica into the leader partition.
- If too many replicas are moved, then there could be serious performance impact on the cluster. When using the kafka-reassign-partitions command, look at the partition counts and sizes. From there, you can test various partition sizes along with the --throttle flag to determine what volume of data can be copied without affecting broker performance significantly.
- Given the earlier restrictions, it is best to use this command only when all brokers and topics are healthy.

How do I monitor my Kafka cluster?

Cloudera Kafka can be monitored and managed using Streams Messaging Manager. Streams Messaging Manager is an operations monitoring and management tool that provides end-to-end visibility in an enterprise Apache Kafka environment. For more information, see Introduction to Streams Messaging Manager or the various Streams Messaging Manager publications available in How to>Streams Messaging.

What are the best practices concerning consumer group.id?

The group.id is just a string that helps Kafka track which consumers are related (by having the same group id).

- In general, timestamps as part of group.id are not useful. Because each group.id corresponds to multiple consumers, you cannot have a unique timestamp for each consumer.

- Add any helpful identifiers. This could be related to a group (for example, transactions, marketing), purpose (fraud, alerts), or technology (Spark).

How do I monitor consumer group lag?

This is typically done using the kafka-consumer-groups command line tool. Copying directly from the upstream documentation, we have this example output (reformatted for readability):

```
$ bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe
--group my-group
TOPIC  PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID       HOST
  CLIENT-ID
my-topic   0           2             4         2  consumer-1-69d6 /127.0.0.1
  consumer-1
my-topic   1           2             3         1  consumer-1-69d6 /127.0.0.1
  consumer-1
my-topic   2           2             3         1  consumer-2-9bb2 /127.0.
0.1  consumer-2
```

In general, if everything is going well with a particular topic, each consumer's CURRENT-OFFSET should be up-to-date or nearly up-to-date with the LOG-END-OFFSET. From this command, you can determine whether a particular host or a particular partition is having issues keeping up with the data rate.

How do I reset the consumer offset to an arbitrary value?

This is also done using the kafka-consumer-groups command line tool. This is generally an administration feature used to get around corrupted records, data loss, or recovering from failure of the broker or host. Aside from those special cases, using the command line tool for this purpose is not recommended.

By using the --execute --reset-offsets flags, you can change the consumer offsets for a consumer group (or even all groups) to a specific setting based on each partitions log's beginning/end or a fixed timestamp. Typing the kafka-co nsumer-groups command with no arguments will give you the complete help output.

How do I configure MirrorMaker for bidirectional replication across DCs?

Mirror Maker is a one way copy of one or more topics from a Source Kafka Cluster to a Destination Kafka Cluster. Given this restriction on Mirror Maker, you need to run two instances, one to copy from A to B and another to copy from B to A.

In addition, consider the following:

- Cloudera recommends using the "pull" model for Mirror Maker, meaning that the Mirror Maker instance that is writing to the destination is running on a host "near" the destination cluster.
- The topics must be unique across the two clusters being copied.
- On secure clusters, the source cluster and destination cluster must be in the same Kerberos realm.

How does the consumer max retries vs timeout work?

With the newer versions of Kafka, consumers have two ways they communicate with brokers.

- Retries: This is generally related to reading data. When a consumer reads from a brokers, it's possible for that attempt to fail due to problems such as intermittent network outages or I/O issues on the broker. To improve reliability, the consumer retries (up to the configured max.retries value) before actually failing to read a log offset.
- Timeout. This term is a bit vague because there are two timeouts related to consumers:
  - Poll Timeout: This is the timeout between calls to KafkaConsumer.poll(). This timeout is set based on whatever read latency requirements your particular use case needs.
  - Heartbeat Timeout: The newer consumer has a "heartbeat thread" which give a heartbeat to the broker (actually the Group Coordinator within a broker) to let the broker know that the consumer is still alive. This

happens on a regular basis and if the broker doesn't receive at least one heartbeat within the timeout period, it assumes the consumer is dead and disconnects it.

How do I size my Kafka cluster?

There are several considerations for sizing your Kafka cluster.

- Disk space

  Disk space will primarily consist of your Kafka data and broker logs. When in debug mode, the broker logs can get quite large (10s to 100s of GB), so reserving a significant amount of space could save you some future headaches.

  For Kafka data, you need to perform estimates on message size, number of topics, and redundancy. Also remember that you will be using RAID10 for Kafka's data, so half your hard drives will go towards redundancy. From there, you can calculate how many drives will be needed.

  In general, you will want to have more hosts than the minimum suggested by the number of drives. This leaves room for growth and some scalability headroom.
- Zookeeper nodes

  One node is fine for a test cluster. Three is standard for most Kafka clusters. At large scale, five nodes is fairly common for reliability.
- Looking at leader partition count/bandwidth usage

  This is likely the metric with the highest variability. Any Kafka broker will be overloaded if it has too many leader partitions. In the worst cases, each leader partition requires high bandwidth, high message rates, or both. For other topics, leader partitions will be a tiny fraction of what a broker can handle (limited by software and hardware). To estimate an average that works on a per-host basis, try grouping topics by partition data throughput requirements, such as 2 high bandwidth data partitions, 4 medium bandwidth data partitions, 20 small bandwidth data partitions. From there, you can determine how many hosts are needed.

How can I build a Spark streaming application that consumes data from Kafka?

You will need to set up your development environment to use both Spark libraries and Kafka libraries:

- Building Spark Applications
- The kafka-examples directory on Cloudera's public GitHub has an example pom.xml.

From there, you should be able to read data using the KafkaConsumer class and using Spark libraries for real-time data processing. The blog post Reading data securely from Apache Kafka to Apache Spark has a pointer to a GitHub repository that contains a word count example.

For further background, read the blog post Architectural Patterns for Near Real-Time Data Processing with Apache Hadoop.

Cloudera Runtime 7.3.1

Tuning Apache Kafka Performance

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Handling large messages **4**

Cluster sizing **4**
Sizing estimation based on network and disk message throughput 5
Choosing the number of partitions for a topic 5

Broker Tuning **7**
JVM and garbage collection 7
Network and I/O threads 7
ISR management 8
Log cleaner 8

System Level Broker Tuning **9**
File descriptor limits 9
Filesystems 9
Virtual memory handling 10
Networking parameters 10
Configure JMX ephemeral ports 10

Kafka-ZooKeeper performance tuning **11**

Handling large messages

Learn more about the options you have when handling large messages with Kafka.

Kafka can be tuned to handle large messages. This can be done by configuring broker and consumer properties relating to maximum message and file sizes. However, before doing so, Cloudera recommends that you try and reduce the size of messages first. Review the following options that can help you reduce message size.

- The Kafka producer can compress messages. For example, if the original message is a text-based format (such as XML), in most cases the compressed message will be sufficiently small.
- Use the compression.type producer configuration parameters to enable compression. gzip, lz4, Snappy, and Zstandard are supported.
- If shared storage (such as NAS, HDFS, or S3) is available, consider placing large files on the shared storage and using Kafka to send a message with the file location. In many cases, this can be much faster than using Kafka to send the large file itself.
- Split large messages into 1 KB segments with the producing client, using partition keys to ensure that all segments are sent to the same Kafka partition in the correct order. The consuming client can then reconstruct the original large message.

If none the options presented are viable, and you need to send large messages with Kafka, you can do so by configuring the following parameters based on your requirements.

Table 1: Broker Configuration Properties

Property	Default Value	Description
message.max.bytes	1000000 (1 MB)	Maximum message size the broker accepts. When tweaking this property, of Producer Batch Size and max.request.size on the producer side.
log.segment.bytes	1073741824 (1 GiB)	Size of a Kafka data file. Must be larger than any single message.
replica.fetch.max.bytes	1048576 (1 MiB)	Maximum message size a broker can replicate. Must be larger than messa accept messages it cannot replicate, potentially resulting in data loss.

Table 2: Consumer Configuration Properties

Property	Default Value	Description
max.partition.fetch.bytes	1048576 (10 MiB)	The maximum amount of data per-partition the server will return.
fetch.max.bytes	52428800 (50 MiB)	The maximum amount of data the server should return for a fetch request

Note: The consumer is able to consume a message batch that is larger than the default value of the max.partition.fetch.bytes or fetch.max.bytes property. However, the batch will be sent alone, which can cause performance degradation.

Cluster sizing

Learn how to size your Kafka cluster.

Sizing estimation based on network and disk message throughput

Learn how to estimate the size of your cluster based on network and disk throughput requirements.

There are many variables that go into determining the correct hardware footprint for a Kafka cluster. The most accurate way to model your use case is to simulate the load you expect on your own hardware. You can do this using the load generation tools that ship with Kafka, kafka-producer-perf-test and kafka-consumer-perf-test.

However, if you want to size a cluster without simulation, a very simple rule could be to size the cluster based on the amount of disk-space required (which can be computed from the estimated rate at which you get data times the required data retention period).

A slightly more sophisticated estimation can be done based on network and disk throughput requirements. To make this estimation, let's plan for a use case with the following characteristics:

- W - MB/sec of data that will be written
- R - Replication factor
- C - Number of consumer groups, that is the number of readers for each write

Kafka is mostly limited by the disk and network throughput.

The volume of writing expected is W * R (that is, each replica writes each message). Data is read by replicas as part of the internal cluster replication and also by consumers. Because every replicas but the master read each write, the read volume of replication is (R-1) * W. In addition each of the C consumers reads each write, so there will be a read volume of C * W. This gives the following:

- Writes: W * R
- Reads: (R+C- 1) * W

However, note that reads may actually be cached, in which case no actual disk I/O happens. We can model the effect of caching fairly easily. If the cluster has M MB of memory, then a write rate of W MB/second allows M/(W * R) seconds of writes to be cached. So a server with 32 GB of memory taking writes at 50 MB/second serves roughly the last 10 minutes of data from cache. Readers may fall out of cache for a variety of reasons—a slow consumer or a failed server that recovers and needs to catch up. An easy way to model this is to assume a number of lagging readers you to budget for. To model this, let’s call the number of lagging readers L. A very pessimistic assumption would be that L = R + C -1, that is that all consumers are lagging all the time. A more realistic assumption might be to assume no more than two consumers are lagging at any given time.

Based on this, we can calculate our cluster-wide I/O requirements:

- Disk Throughput (Read + Write): W * R + L * W
- Network Read Throughput: (R + C -1) * W
- Network Write Throughput: W * R

A single server provides a given disk throughput as well as network throughput. For example, if you have a 1 Gigabit Ethernet card with full duplex, then that would give 125 MB/sec read and 125 MB/sec write; likewise 6 7200 SATA drives might give roughly 300 MB/sec read + write throughput. Once we know the total requirements, as well as what is provided by one machine, you can divide to get the total number of machines needed. This gives a machine count running at maximum capacity, assuming no overhead for network protocols, as well as perfect balance of data and load. Since there is protocol overhead as well as imbalance, you want to have at least 2x this ideal capacity to ensure sufficient capacity.

Related Information

kafka-*-perf-test

Choosing the number of partitions for a topic

Learn how to determine the number of partitions each of your Kafka topics requires.

Choosing the proper number of partitions for a topic is the key to achieving a high degree of parallelism with respect to writes to and reads and to distribute load. Evenly distributed load over partitions is a key factor to have good throughput (avoid hot spots). Making a good decision requires estimation based on the desired throughput of producers and consumers per partition.

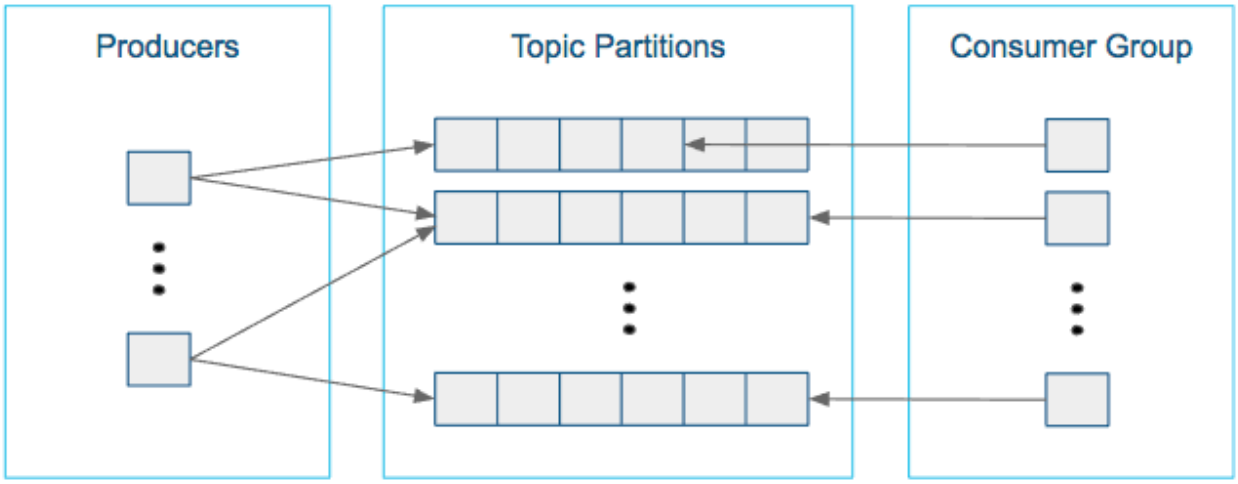

For example, if you want to be able to read 1 GB/sec, but your consumer is only able process 50 MB/sec, then you need at least 20 partitions and 20 consumers in the consumer group. Similarly, if you want to achieve the same for producers, and 1 producer can only write at 100 MB/sec, you need 10 partitions. In this case, if you have 20 partitions, you can maintain 1 GB/sec for producing and consuming messages. You should adjust the exact number of partitions to number of consumers or producers, so that each consumer and producer achieve their target throughput.

So a simple formula could be:

```
#Partitions = max(NP, NC)
```

where:

- NP is the number of required producers determined by calculating: TT/TP
- NC is the number of required consumers determined by calculating: TT/TC
- TT is the total expected throughput for our system
- TP is the max throughput of a single producer to a single partition
- TC is the max throughput of a single consumer from a single partition

This calculation gives you a rough indication of the number of partitions. It's a good place to start. Keep in mind the following considerations for improving the number of partitions after you have your system in place:

- The number of partitions can be specified at topic creation time or later.
- Increasing the number of partitions also affects the number of open file descriptors. So make sure you set file descriptor limit properly.
- Reassigning partitions can be very expensive, and therefore it's better to over- than under-provision.
- Changing the number of partitions that are based on keys is challenging and involves manual copying.
- Reducing the number of partitions is not currently supported. Instead, create a new a topic with a lower number of partitions and copy over existing data.
- Metadata about partitions are stored in ZooKeeper in the form of znodes. Having a large number of partitions has effects on ZooKeeper and on client resources:
    - Unneeded partitions put extra pressure on ZooKeeper (more network requests), and might introduce delay in controller and/or partition leader election if a broker goes down.
    - Producer and consumer clients need more memory, because they need to keep track of more partitions and also buffer data for all partitions.

- As guideline for optimal performance, you should not have more than 4000 partitions per broker and not more than 200,000 partitions in a cluster.

Make sure consumers don't lag behind producers by monitoring consumer lag. To check consumers' position in a consumer group (that is, how far behind the end of the log they are), use the following command:

```
kafka-consumer-groups --bootstrap-server BROKER_ADDRESS --describe --gr
oup CONSUMER_GROUP --new-consumer
```

Related Information

Managing Apache kafka

Broker Tuning

JVM and garbage collection

Learn more about garbage collection for Kafka.

Garbage collection has a huge impact on performance of JVM based applications. It is recommended to use the Garbage-First (G1) garbage collector for Kafka broker. In Cloudera Manager specify the following under Additional Broker Java Options in the Kafka service configuration:

```
-server -XX:+UseG1GC -XX:MaxGCPauseMillis=20
-XX:InitiatingHeapOccupancyPercent=35 -XX:+DisableExplicitGC
-Djava.awt.headless=true -Djava.net.preferIPv4Stack=true
```

Cloudera recommends to set 4-8 GB of JVM heap size memory for the brokers depending on your use case. As Kafka's performance depends heavily on the operating systems page cache, it is not recommended to collocate with other memory-hungry applications.

- Large messages can cause longer garbage collection (GC) pauses as brokers allocate large chunks. Monitor the GC log and the server log.

  Add this to Broker Java Options:

  ```
   -XX:+PrintGC -XX:+PrintGCDetails
   -XX:+PrintGCTimeStamps
  -Xloggc:</path/to/file.txt>
  ```

- If long GC pauses cause Kafka to abandon the ZooKeeper session, you may need to configure longer timeout values, see Kafka-ZooKeeper performance tuning for details.

Related Information

Kafka-ZooKeeper performance tuning

Network and I/O threads

Learn more about the configuration properties related to network and I/O threads.

Kafka brokers use network threads to handle client requests. Incoming requests (such as produce and fetch requests) are placed into a requests queue from where I/O threads are taking them up and process them. After a request is

processed, the response is placed into an internal response queue from where a network thread picks it up and sends response back to the client.

- num.network.threads is an important cluster-wide setting that determines the number of threads used for handling network requests (that is, receiving requests and sending responses). Set this value mainly based on number of producers, consumers and replica fetchers.
- queued.max.requests controls how many requests are allowed in the request queue before blocking network threads.
- num.io.threads specifies the number of threads that a broker uses for processing requests from the request queue (might include disk I/O).

ISR management

Learn more about in-sync replica management.

An in-sync replica (ISR) set for a topic partition contains all follower replicas that are caught-up with the leader partition, and are situated on a broker that is alive.

- If a replica lags “too far” behind from the partition leader, it is removed from the ISR set. The definition of what is too far is controlled by the configuration setting replica.lag.time.max.ms. If a follower hasn't sent any fetch requests or hasn't consumed up to the leaders log end offset for at least this time, the leader removes the follower from the ISR set.
- num.replica.fetchers is a cluster-wide configuration setting that controls how many fetcher threads are in a broker. These threads are responsible for replicating messages from a source broker (that is, where partition leader resides). Increasing this value results in higher I/O parallelism and fetcher throughput. Of course, there is a trade-off: brokers use more CPU and network.
- replica.fetch.min.bytes controls the minimum number of bytes to fetch from a follower replica. If there is not enough bytes, wait up to replica.fetch.wait.max.ms.
- replica.fetch.wait.max.ms controls how long to sleep before checking for new messages from a fetcher replica. This value should be less than replica.lag.time.max.ms, otherwise the replica is kicked out of the ISR set.
- To check the ISR set for topic partitions, run the following command:

```
kafka-topics --zookeeper ${ZOOKEEPER_HOSTNAME}:2181/kafka --describe --t
opic ${TOPIC}
```

- If a partition leader dies, a new leader is selected from the ISR set. There will be no data loss. If there is no ISR, unclean leader election can be used with the risk of data-loss.
- Unclean leader election occurs if unclean.leader.election.enable is set to true. By default, this is set to false.

Log cleaner

An overview on configuration properties related to log compaction.

The following cluster-wide configuration settings can be used to fine tune log compaction:

- log.cleaner.threads controls how many background threads are responsible for log compaction. Increasing this value improves performance of log compaction at the cost of increased I/O activity.
- log.cleaner.io.max.bytes.per.second throttles log cleaner’s I/O activity so that the sum of its read and write is less than this value on average.
- log.cleaner.dedupe.buffer.size specifies memory used for log compaction across all cleaner threads.
- log.cleaner.io.buffer.size controls total memory used for log cleaner I/O buffers across all cleaner threads.
- log.cleaner.min.compaction.lag.ms controls how long messages are left uncompacted.
- log.cleaner.io.buffer.load.factor controls log cleaner’s load factor for the dedupe buffer. Increasing this value allows the system to clean more logs at once but increases hash collisions.
- log.cleaner.backoff.ms controls how long to wait until the next check if there is no log to compact.

Related Information

Record Management

System Level Broker Tuning

Operating system related kernel parameters affect overall performance of Kafka. These parameters can be configured via sysctl at runtime. To make kernel configuration changes persistent (that is, use adjusted parameters after a reboot), edit /etc/sysctl.conf. The following sections describe some important kernel settings.

File descriptor limits

Learn more about file descriptor requirements.

As Kafka works with many log segment files and network connections, the Maximum Process File Descriptors setting may need to be increased in some cases in production deployments, if a broker hosts many partitions. For example, a Kafka broker needs at least the following number of file descriptors to just track log segment files:

```
(number of partitions)*(partition size / segment size)
```

The broker needs additional file descriptors to communicate via network sockets with external parties (such as clients, other brokers, Zookeeper, and Kerberos).

The Maximum Process File Descriptors setting can be monitored in Cloudera Manager and increased if usage requires a larger value than the default ulimit (often 64K). It should be reviewed for use case suitability.

- To review FD limit currently set for a running Kafka broker, run cat /proc/*KAFKA_BROKER_PID*/limits, and look for Max open files.
- To see open file descriptors, run:

  ```
  lsof -p KAFKA_BROKER_PID
  ```

Important:

After changing the maximum process file descriptor limit, you might find that service roles are still limited by the number of file descriptors. Raising the maximum process file descriptors above the Linux kernel file descriptor limit will have no effect. Check the Linux kernel file descriptor limit on every host in the cluster and raise that if necessary.

You can find the Linux kernel file descriptor limit by running the following command on the Linux command line:

```
sudo cat /proc/sys/fs/nr_open
```

Filesystems

Learn how mount options such as noatime and relatime affect Kafka performance.

Linux records when a file was created (ctime), modified (mtime) and accessed (atime). The value noatime is a special mount option for filesystems (such as EXT4) in Linux that tells the kernel not to update inode information every time a file is accessed (that is, when it was last read). Using this option can result in write performance gain. Kafka is not relying on atime. The value relatime is another mounting option that optimizes how atime is persisted. Access time is only updated if the previous atime was earlier than the current modified time.

To view mounting options, run mount -l or cat /etc/fstab command.

Virtual memory handling

Learn about how virtual memory handling affects Kafka performance.

Kafka uses system page cache extensively for producing and consuming the messages. The Linux kernel parameter, vm.swappiness, is a value from 0-100 that controls the swapping of application data (as anonymous pages) from physical memory to virtual memory on disk. The higher the value, the more aggressively inactive processes are swapped out from physical memory. The lower the value, the less they are swapped, forcing filesystem buffers to be emptied. It is an important kernel parameter for Kafka because the more memory allocated to the swap space, the less memory can be allocated to the page cache. Cloudera recommends to set vm.swappiness value to 1.

- To check memory swapped to disk, run vmstat and look for the swap columns.

Kafka heavily relies on disk I/O performance. vm.dirty_ratio and vm.dirty_background_ratio are kernel parameters that control how often dirty pages are flushed to disk. Higher vm.dirty_ratio results in less frequent flushes to disk.

- To display the actual number of dirty pages in the system, run egrep "dirty|writeback" /proc/vmstat

Networking parameters

An overview on Linux kernel parameters related to networking to fine tune for better Kafka performance.

Kafka is designed to handle a huge amount of network traffic. By default, the Linux kernel is not tuned for this scenario. The following kernel settings may need to be tuned based on use case or specific Kafka workload:

- net.core.wmem_default: Default send socket buffer size.
- net.core.rmem_default: Default receive socket buffer size.
- net.core.wmem_max: Maximum send socket buffer size.
- net.core.rmem_max: Maximum receive socket buffer size.
- net.ipv4.tcp_wmem: Memory reserved for TCP send buffers.
- net.ipv4.tcp_rmem: Memory reserved for TCP receive buffers.
- net.ipv4.tcp_window_scaling: TCP Window Scaling option.
- net.ipv4.tcp_max_syn_backlog: Maximum number of outstanding TCP SYN requests (connection requests).
- net.core.netdev_max_backlog: Maximum number of queued packets on the kernel input side (useful to deal with spike of network requests).

Use the Cloudera Enterprise Reference Architecture for Bare Metal Deploymentsas a guideline when specifying values for the parameters.

Related Information

Cloudera Enterprise Reference Architecture for Bare Metal Deployments

Configure JMX ephemeral ports

Learn more aboutKafka JMX ephermal ports and how to configure them.

Kafka uses two high-numbered ephemeral ports for JMX. These ports are listed when you view netstat -anp information for the Kafka broker process.

- You can change the number for the first port by adding a command similar to the following to the field Additional Broker Java Options (broker_java_opts) in Cloudera Manager.

  ```
  -Dcom.sun.management.jmxremote.rmi.port=PORT
  ```

- The JMX_PORT configuration maps to com.sun.management.jmxremote.port by default.

  To access JMX via JConsole, run jconsole ${BROKER_HOST}:9393

- The second ephemeral port used for JMX communication is implemented for the JRMP protocol and cannot be changed.

Kafka-ZooKeeper performance tuning

Recommendations and parameters in connection with Kafka-Zookeeper performance tuning.

Kafka uses Zookeeper to store metadata information about topics, partitions, brokers and system coordination (such as membership statuses). Unavailability or slowness of Zookeeper makes the Kafka cluster unstable, and Kafka brokers do not automatically recover from it. Cloudera recommends to use a 3-5 machines Zookeeper ensemble solely dedicated to Kafka (co-location of applications can cause unwanted service disturbances).

- zookeeper.session.timeout.ms is a setting for Kafka that specifies how long Zookeeper shall wait for heartbeat messages before it considers the client (the Kafka broker) unavailable. If this happens, metadata information about partition leadership owned by the broker will be reassigned. If this setting is too high, then it might take a long time for the system to detect a broker failure. On the other hand, if it is set to too small, it might result in frequent leadership reassignments.
- jute.maxbuffer is a crucial Java system property for both Kafka and Zookeeper. It controls the maximum size of the data a znode can contain. The default value, one megabyte, might be increased for certain production use cases.
- There are cases where Zookeeper can require more connections. In those cases, it is recommended to increase the maxClientCnxns parameter in Zookeeper.
- Note that old Kafka consumers store consumer offset commits in Zookeeper (deprecated). It is recommended to use new consumers that store offsets in internal Kafka topics as this reduces the load on Zookeeper.

Cloudera Runtime 7.3.1

Planning for Apache Kafka

Date published: 2019-08-22
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Kafka stretch cluster reference architecture..... 4
Planning your deployment.....5
Configuring Kafka brokers.....8
Configuring topics and clients..... 9

Apache Kafka stretch cluster reference architecture

A Kafka reference architecture for a three DC stretch cluster deployment. Use this reference architecture to better understand the steps required to set up a Kafka stretch cluster.

Important: The following reference architecture is primarily for Cloudera Private Cloud deployments. Although the general concepts and configuration documented here can be applied to Cloudera Public Cloud deployments, Cloudera recommends that you use the Streams Messaging High Availability cluster template in conjunction with the multi-AZ capabilities of the Cloudera Data Hub service if you want to deploy a highly available stretch cluster in a public cloud environment. For more information, see the following resources:

- Streams Messaging cluster layout
- Deploying Cloudera in multiple AWS availability zones
- Creating a multi-AZ Cloudera Data Hub cluster on AWS

A stretch cluster is a single logical Kafka cluster deployed across multiple Data Centers (DC) or other independent physical infrastructures such as cloud availability zones. Stretch clusters can be operated with three or more DCs. The most cost-effective and frequently used architecture is the three DC cluster.

Assuming that there is a zero Recovery Point Objective (RPO=0) requirement and the minimum required copies of the data residing in different DCs is two (topics have replication factor of three, the number of minimum in sync replicas is two), a three DC stretch cluster can tolerate the failure of a single DC while remaining fully functional. In case of a second DC failure, the cluster will fail to function as the Zookeeper quorum is lost, and Kafka cannot operate without a functioning Zookeeper cluster. However, even after the failure of the second DC, the latest (acknowledged) state of the Kafka topics will still be accessible on the last cluster.

The following sections describe the planning and configuration steps needed to set up a three DC stretch cluster.

Figure 1: Kafka stretch cluster example with three Data Centers

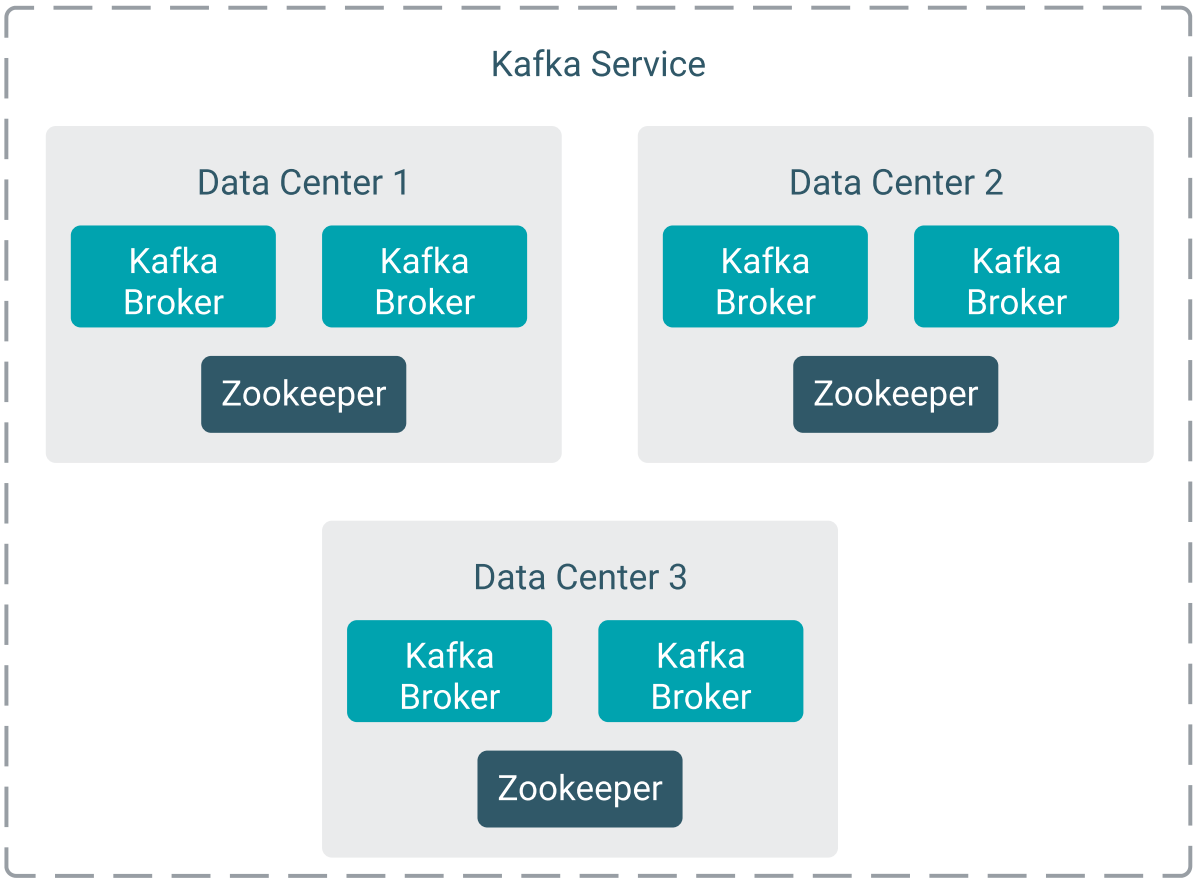

Planning your deployment

Learn about the planning steps zou need to take when building and designing a Kafka stretch cluster.

Naming the Data Centers

In a Kafka stretch cluster deployment, each DC must have a name (ID). The names are used as the rack IDs when configuring the brokers of the stretch cluster. How you name the DCs is determined by your organization policies and rules on naming. Cloudera recommends that you choose unique and easily identifiable names.This guide refers to the three DCs as DC1, DC2, and DC3.

Choosing the number of replicas in a Data Center

The number of replicas inside the same DC affects the required storage capacity and the availability of the cluster in case of node failures. The number of replicas also defines the minimum number of brokers required per DC.

In the majority of cases, having a single replica inside a DC is sufficient. However, having more replicas has the following benefits:

1. Increased availability. More broker failures can be tolerated without the topic being unavailable.
2. Follower fetching is supported even in case of single broker failures.
3. Increased data durability in case of disk failures.

Note: Increased data durability can also be achieved with fault tolerant disk technologies or services.

Choose the number of replicas per DC based on your availability requirements. Once you have decided on the number, you need to calculate the replication factor (replication.factor) and the number of minimum in sync replicas (min.insync.replicas). The values you calculate are used when you configure the brokers, topics, and clients.

The replication factor of the topics can be calculated based on the number of DCs and the number of replicas per DC

```
[***DC COUNT***] * [***REPLICA PER DC***] = replication.factor
```

The minimum in sync replicas of the topics can be calculated based on the minimum DC copies and the replicas per DC.

```
[***REPLICA PER DC***] * ([***MINIMUM DC REPLICAS***] - 1) + 1 = min.insy
nc.replicas
```

For example, assume that you have 3 Data Centers with 2 Kafka brokers each. Additionally, you have 2 replicas in each DC and you want to have a minimum of 2 DCs with replicas. In a case like this, the replication factor of the topics is 6, 3 * 2 = 6. The minimum in sync replicas is 3, 2 * (2-1) + 1 = 3. The cluster would look similar to the following example when all DCs are online:

Figure 2: Kafka stretch cluster with three DCs and six replicas

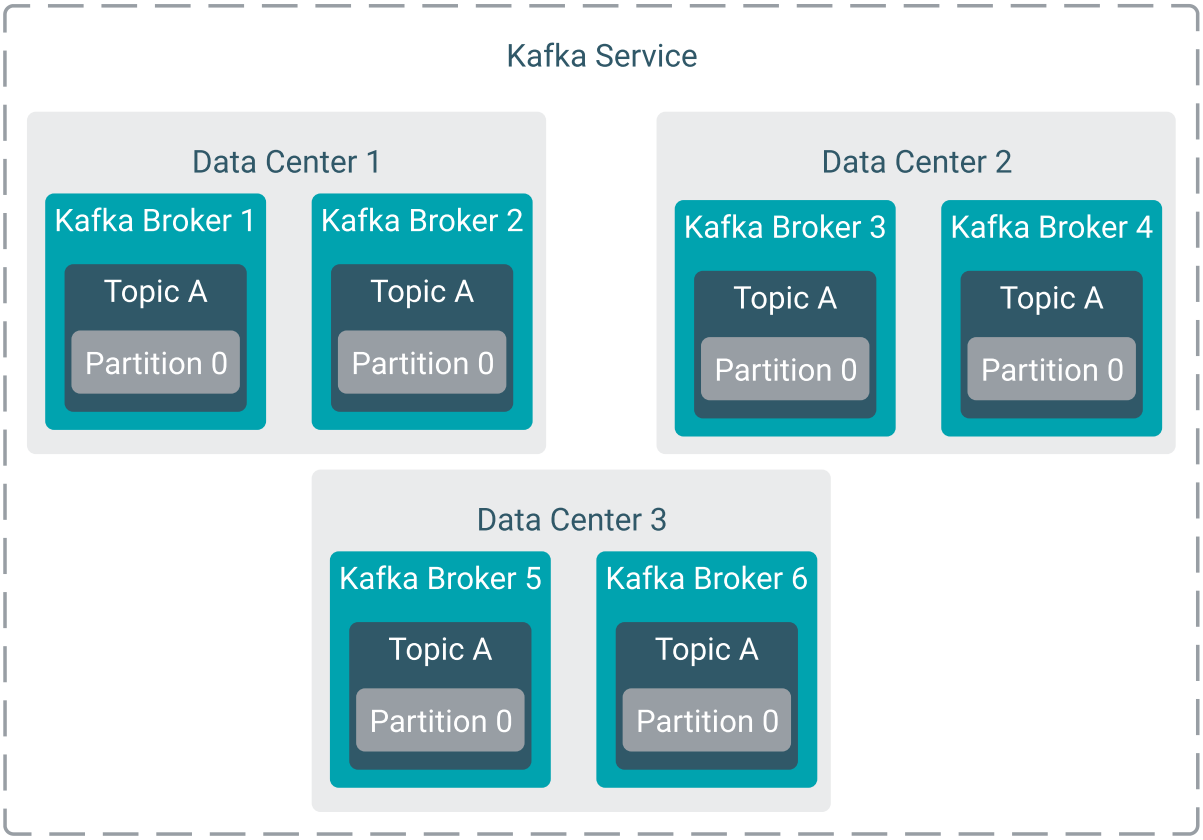

The formula for calculating the minimum in-sync replicas ensures that even if you are experiencing a severe outage, there will be at least one replica available in two different DCs.

Figure 3: Kafka stretch cluster outage example

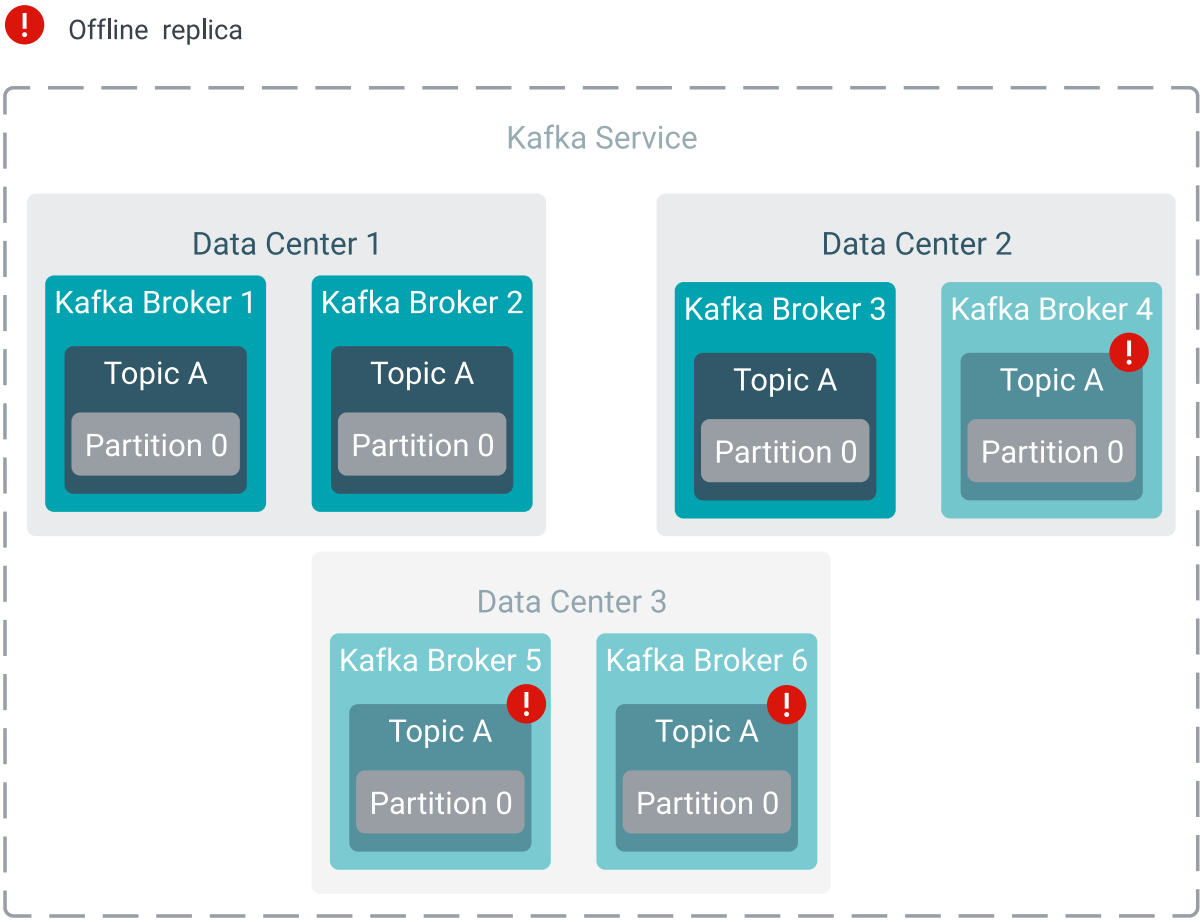

Collect network information and calculate buffer size

Because a stretch cluster is spread across multiple DCs, and those DCs can be located far away from each other, a stretch cluster will have higher latency compared to a traditional Kafka cluster. The latency must be accounted for and you will need to configure broker buffer sizes to adapt to the increased latency.

To calculate the buffer size, first, you must collect network information. Do the following:

- Measure average round-trip time (RTT) between the DCs. Take note of the maximum of the averages. Cloudera recommends you take the measurements in seconds.
- Measure the bandwidth of the links between the DCs. Cloudera recommends that you take the measurements in bytes/second.

Important: The RTT between the selected DCs should be lower than 50 ms. The throughput of Kafka degrades rapidly with increasing latency. Because of this, Cloudera does not recommend using a stretch cluster with DCs that have a latency higher than 50 ms.

Next, using these measurements, calculate the buffer size. Use the bandwidth-delay product formula:

```
[***RTT***] * [***BANDWIDTH***] = [***BUFFER SIZE***]
```

Ensure that the buffer size is calculated in bytes. For example if the RTT is 30 ms (0.03 s) and the bandwidth is 1 GB/s then:

```
0.03 s * 1 GB/s ~ 31 MB = 32505856 B
```

Configuring Kafka brokers

Learn how to configure Kafka brokers for a stretch cluster deployment.

About this task

After you finish planning your deployment, you will need to configure your brokers. The following steps assume that Cloudera Runtime, Cloudera Manager, and the Kafka service are already installed on your cluster. Additionally, the following steps only cover the minimum configurations needed for a stretch cluster deployment. Depending on your requirements, additional configuration not detailed here, such as security configuration, might be required.

Procedure

1. Configure broker rack IDs (enable Kafka rack awareness).

   Broker rack IDs must be specified in order to ensure that replicas are evenly distributed among DCs. Broker rack IDs are specified using the Broker Rack ID property of the broker instance. The values you specify for the Broker Rack ID property should match the name of the DCs. This ensures that when a topic is created, its replicas are assigned in a round-robin fashion among the racks (or DCs in the case of this scenario).

   Repeat the following steps for each Kafka broker role instance.

   a) In Cloudera Manager, select the Kafka service.
   b) Go to Instances.
   c) Click any of the broker role instances and go to Configuration.
   d) Find and configure the Broker Rack ID property.

      Add the name of the DCs that the broker role instance is deployed in. For example, for the brokers deployed in DC1, you need to specify DC1 as the ID, for DC2 specify DC2, and so on.
   e) Click Save Changes.

   Note: Broker rack IDs can also be configured on the level of the host. This can be done by specifying rack information for your hosts using the HostsAll HostsActions for SelectedAssign Rack action and selecting the Enable Rack Awareness Kafka service property. For more information, see *Configuring Kafka rack awareness*.

   Tip: You can create a role group for each of your DCs and add the broker instances that reside in the DC to the appropriate role group. Afterwards, you can set Broker Rack ID on the level of the role group. This allows you to configure the rack IDs on a per DC basis rather than having to do it manually for each broker instance.

2. Configure the internal topics of Kafka for the designed durability.

   This step involves configuring the replication factor and minimum in-sync replica count of the offsets and transactions internal topics. Use the values you calculated during deployment planning.

   a) In Cloudera Manager, select the Kafka service.
   b) Go to Configuration.
   c) Find and configure the following properties:

      - Offset Commit Topic Replication Factor
      - Minimum Number of Replicas in ISR
      - Transaction State Log Replication Factor
      - Minimum Number of Replicas in ISR for Transaction State Log

      Note: The offsets topic does not have a dedicated configuration for minimum in-sync replicas. As a result, you must use Minimum Number of Replicas in ISR, which sets a cluster-wide default.

3. Change defaults of automatically created topics.

   If Topic Auto Creation is set selected, topics can be automatically created by clients. To ensure that these topics are created with the required durability settings, the broker-level default replication factor must be changed to the value you calculated during planning. Additionally, unclean leader election must be disabled.

   a) Find and configure the Default Replication Factor property.
   b) Find and clear the Enable Unclean Leader Election property.

4. Configure the broker buffer sizes to adapt to the increased latency of the stretch cluster.

   The buffer size you configure must be equivalent to or greater than the buffer size you calculated during planning.

   a) Find and configure the Socket send buffer size. property.
   b) Find and configure the Socket receive buffer size. (socket.receive.buffer.bytes) property.

      Note: There are two properties in Cloudera Manager called Socket send buffer size.. Ensure that you configure socket.receive.buffer.bytes and not socket.request.max.bytes.

   c) Find the Kafka Broker Advanced Configuration Snippet (Safety Valve) for kafka.properties property.
   d) Add the following configuration entry to the advanced configuration snippet:

```
replica.socket.receive.buffer.bytes=[***BUFFER SIZE***]
```

5. Enable follower fetching.

   To allow consumers to fetch from the closest replica instead of the leader, brokers must be configured to support follower fetching. Follower fetching allows consumers that have their physical location specified (in this case a DC) to fetch data from the replica located in the same location as the consumer. This can reduce cross-DC traffic.

   a) Find the Kafka Broker Advanced Configuration Snippet (Safety Valve) for kafka.properties property.
   b) Add the following configuration entry to the advanced configuration snippet:

```
replica.selector.class=org.apache.kafka.common.replica.RackAwareReplicaS
elector
```

6. Click Save Changes.
7. Restart the Kafka service.

Related Information

Configuring Kafka rack awareness

Configuring topics and clients

Learn about the configuration required for topics and clients in a Kafka stretch cluster.

Configure topics

Topics must always use the calculated replication factor and minimum in-sync replica count to achieve the desired durability. Additionally, unclean leader election must be disabled. Based on how topics are created in the system, ensure that they are created with the correct configurations.

- If topics are automatically created and you have followed Configuring Kafka brokers on page 8, no additional configuration is needed.
- If topics are manually created, make sure that the administrators use the required configurations when creating the topics.
- In case topics are created by the client applications, make sure to configure the clients to use the correct replication factor and minimum in-sync replica count.

Configure consumers

Consumer configuration in a stretch cluster deployment is only necessary if follower fetching is enabled for the broker. For follower fetching to function, configuration on the consumer's side is also required. Configuration involves specifying the physical location of the consumer using the client.rack property.

The value of client.rack must match the DC names used in the Broker Rack ID property of each broker instance. For the consumers that are running in the DCs, set client.rack to the ID of the DC that the consumer is running in. If the consumer is deployed in a DC with no brokers, specify the ID of a DC that is closest to the consumer. Closest meaning either lowest latency or lowest cost of data transfer.

For example, if a consumer client is located in or closest to DC2, then your configuration should be the following:

```
client.rack=DC2
```

Consumers that do not have client.rack specified fall back to fetching from the leader.

Configure producers

Producers applications must respect the acknowledgment of records. Producers can only handle records as successfully written when the acknowledgement arrives. Additionally, producers must use acks=all to only receive the acknowledgement when the sufficient number of followers have replicated the record.

Note: The acks property is set to all by default for producer versions 3.0.0 or later.

Configure Kafka Streams applications

Kafka Streams applications create internal topics based on their topology. To make sure that these topics meet the durability requirements, their replication factor must be correctly configured by setting the replication factor to the value you calculated during deployment planning. For example:

```
replication.factor=6
```

Cloudera Runtime 7.3.1

Securing Apache Kafka

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Channel encryption.....5
- Configure TLS/SSL encryption for Kafka brokers.....5
- Configure TLS/SSL encryption for Kafka clients.....6
- Configuring TLS/SSL encryption for Kafka MirrorMaker.....7
- Configure Zookeeper TLS/SSL support for Kafka.....8

Authentication.....9
- TLS/SSL client authentication.....9
  - Configure TLS/SSL client authentication for Kafka brokers.....9
  - Configure TLS/SSL authentication for Kafka clients.....10
  - Principal name mapping.....12
- Enable Kerberos authentication for Kafka.....12
- Delegation token based authentication.....14
  - Enable or disable authentication with delegation tokens.....15
  - Manage individual delegation tokens.....15
  - Rotate the master key/secret.....16
  - Client authentication using delegation tokens.....17
- LDAP authentication.....18
  - Configure LDAP authentication for Kafka brokers.....18
  - Configure Kafka LDAP authentication for Kafka clients.....19
- PAM authentication.....21
  - Configure PAM authentication for Kafka brokers.....21
  - Configure PAM authentication for Kafka clients.....21
- OAuth2 authentication.....22
  - Configuring OAuth2 authentication for Kafka brokers.....24
  - Configuring OAuth2 authentication for Kafka clients.....26

Authorization.....28
- Ranger.....28
  - Enable authorization in Kafka with Ranger.....28
  - Configure the resource-based Ranger service used for authorization.....29
  - Kafka ACL APIs support in Ranger.....30

Governance.....32
- Importing Kafka entities into Atlas.....32
- Configuring the Atlas hook in Kafka.....33

Inter-broker security.....34

Configuring multiple listeners.....35

Kafka security hardening with Zookeeper ACLs.....36
- Restricting access to Kafka metadata in Zookeeper.....36

Unlocking access to Kafka metadata in Zookeeper.....37

Channel encryption

Kafka supports client to broker and inter-broker TLS/SSL encrypted communications. Configuring TLS/SSL encryption for a Kafka deployment involves configuring both clients and brokers. In addition to this, Kafka also supports TLS/SSL communication with Zookeeper.

Configure TLS/SSL encryption for Kafka brokers

Kafka supports TLS/SSL encrypted communication with both brokers and clients. To enable and configure TLS/SSL, you need to enable TLS/SSL for the brokers and enter key and truststore related information.

About this task

The following list of steps walks you through the configuration required to set up TLS/SSL encryption for Kafka brokers. It lists all mandatory configuration properties as well as a number of optional properties that you can configure.

Kafka brokers support multiple key and truststore types. The following instructions, however, do not provide details regarding how the key or truststore type used by the brokers is configured. This is because the store type is not configured at a broker level. Instead, it is configured on Cloudera Manager's central security page by going to AdministrationSettingsJava Keystore Type.

Before you begin

- Generate or acquire a key and truststore for your brokers which contain all necessary keys and certificates.
- Note down the locations and passwords for the key and truststores. You will need to provide these during configuration.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find and configure the following properties based on your cluster and requirements.

Cloudera Manager Property	Description
Enable TLS/SSL for Kafka Broker	Enables or disables TLS/SSL communication between clients and the Kafka broker.
Kafka Broker TLS/SSL Server JKS Keystore File Location	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when the Kafka broker is acting as a TLS/SSL server.
Kafka Broker TLS/SSL Server JKS Keystore File Password	The password for the Kafka broker keystore file.
Kafka Broker TLS/SSL Server JKS Keystore Key Password	The password that protects the private key contained in the keystore used when the Kafka broker is acting as a TLS/SSL server.
Kafka Broker TLS/SSL Client Trust Store File	The location on disk of the truststore used to confirm the authenticity of TLS/SSL servers that the Kafka broker might connect to. This is used when the Kafka broker is the client in a TLS/SSL connection. This truststore must contain the certificate(s) that signed the certificate(s) of the connected service(s). If this parameter is not provided, the default list of known certificate authorities is used instead.

Cloudera Manager Property	Description
Kafka Broker TLS/SSL Client Trust Store Password	The password for the Kafka broker truststore file. This password is not mandatory to access the truststore; this field can be left blank. This password provides optional integrity checking of the file. The contents of truststores are certificates, and certificates are public information.

4. (Optional) Configure additional properties.

   If required, additional TLS/SSL related properties can be configured with the Kafka Broker Advanced Configuration Snippet (Safety Valve) for kafka.properties property. The following are some of the most commonly used optional properties:

Kafka Broker Property	Description
ssl.provider	The name of the security provider used for TLS/SSL connections. Default is the default security provider of the JVM.
ssl.cipher.suites	A cipher suite is a named combination of authentication, encryption, MAC, and a key exchange algorithm used to negotiate the security settings for a network connection using the TLS/SSL network protocol.
ssl.enabled.protocols	List of enabled protocols, for example, TLSv1.2,TLSv1.1,TLSv1. Should contain at least one protocol.

5. Click Save Changes.
6. Restart the Kafka service.

Results

Kafka brokers are configured for TLS/SSL encryption.

What to do next

Configure your clients for TLS/SSL encryption. Alternatively, you can also continue with configuring TLS/SSL authentication for the brokers.

Related Information

Configure TLS/SSL client authentication for Kafka brokers

Configure TLS/SSL encryption for Kafka clients

Kafka supports TLS/SSL encrypted communication with both brokers and clients. Client configuration is done by setting the relevant security-related properties for the client.

About this task

The following steps demonstrate configuration for the console consumer or producer. If you are configuring a custom developed client, see *Java client security examples* or *.Net client security examples* for code examples.

Before you begin

- Generate or acquire a truststore containing the certificate of the Certificate Authority that issued the broker certificates.
- Note down the location and password for the truststore. You will need to provide these during configuration.

Procedure

1. Create a .properties file.

   In this example the file is named client.properties.

2. Add the mandatory properties to the file.

   The following configuration example contains all mandatory properties:

```
security.protocol=SSL
ssl.truststore.location= [***PATH TO CLIENT TRUSTSTORE***]
ssl.truststore.password=[***PASSWORD***]
```

3. (Optional) Add additional security properties to the file.

   Depending on your requirements and broker configuration, additional configuration properties might also be needed. The following are some of the most commonly used optional properties:

   - ssl.provider
   - ssl.cipher.suites
   - ssl.enabled.protocols
   - ssl.truststore.type
   - ssl.keystore.type

4. Run the client.

   Console Producer

```
kafka-console-producer --bootstrap-server [***HOST1:PORT1***] --t
opic [***TOPIC***] --producer.config client.properties
```

   Console Consumer

```
kafka-console-consumer --bootstrap-server [***HOST1:PORT1***] --t
opic [***TOPIC***] --consumer.config client.properties
```

Results

Kafka clients are configured for TLS/SSL encryption.

Related Information

Java client security examples

.Net client security examples

Configuring TLS/SSL encryption for Kafka MirrorMaker

The Kafka MirrorMaker role supports TLS/SSL encrypted communication with Kafka brokers. To enable and configure TLS/SSL, you need to enable TLS/SSL for the MirrorMaker role, enter key and truststore related information, and specify the client authentication used by the source and destination Kafka clusters.

Before you begin

- Generate or acquire a key and truststore for the MirrorMaker role which contain all necessary keys and certificates.
- Note down the locations and passwords for the key and truststores. You will need to provide these during configuration.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.

3. Find and configure the following properties based on your cluster and requirements.

Table 1:

Cloudera Manager Property	Description
Enable TLS/SSL for Kafka MirrorMaker ssl_enabled	Encrypt communication between clients and Kafka MirrorMaker using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).
Kafka MirrorMaker TLS/SSL Server JKS Keystore File Location ssl_server_keystore_location	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when Kafka MirrorMaker is acting as a TLS/SSL server.
Kafka MirrorMaker TLS/SSL Server JKS Keystore File Password ssl_server_keystore_password	The password for the Kafka MirrorMaker keystore file.
Kafka MirrorMaker TLS/SSL Server JKS Keystore Key Password ssl_server_keystore_keypassword	The password that protects the private key contained in the JKS keystore used when Kafka MirrorMaker is acting as a TLS/SSL server.
Kafka MirrorMaker TLS/SSL Trust Store File ssl_client_truststore_location	The location on disk of the trust store, used to confirm the authenticity of TLS/SSL servers that Kafka MirrorMaker might connect to. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
Kafka MirrorMaker TLS/SSL Trust Store Password ssl_client_truststore_password	The password for the Kafka MirrorMaker TLS/SSL Trust Store File. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.
Source Kafka Cluster's Client Auth source.ssl.client.auth	Only required if the source Kafka cluster requires client authentication.
Destination Kafka Cluster's Client Auth destination.ssl.client.auth	Only required if destination Kafka cluster requires client authentication.

4. Click Save Changes.
5. Restart the Kafka service.

Results

The Kafka MirrorMaker role is configured for TLS/SSL encryption.

Configure Zookeeper TLS/SSL support for Kafka

Learn how to configure TLS/SSL communication between Kafka and Zookeeper.

About this task

You can configure Kafka to connect to and communicate with Zookeeper through a secure TLS/SSL channel. The feature can be enabled or disabled with the Enable Secure Connection to ZooKeeper property. This property is set to true by default, however, it only takes effect if the Enable TLS/SSL for ZooKeeper property is also enabled for the dependent ZooKeeper service.

Before you begin

If you want to enable secure connections to Zookeeper, make sure that the Enable TLS/SSL for ZooKeeper property is enabled for the dependent ZooKeeper service. For more information, see Configure ZooKeeper TLS/SSL using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Select Configuration and find the Enable Secure Connection to ZooKeeper property.
3. Enable or disable Zookeeper TLS/SSL support for Kafka for all required services by selecting or clearing the checkbox next to the name of the service.
4. Enter a Reason for change, and click Save Changes to commit the changes.
5. Restart the Kafka service.

Results

Zookeeper TLS/SSL support for Kafka is enabled or disabled for the selected Kafka services. If the feature is enabled, the selected Kafka services communicate with their dependent Zookeeper service through a secure TLS/SSL channel.

Authentication

TLS/SSL client authentication

Kafka supports TLS/SSL authentication for its clients. Configuring TLS/SSL authentication for a Kafka deployment involves enabling TLS/SSL encryption for the brokers and then configuring both clients and brokers for TLS/SSL authentication.

Configure TLS/SSL client authentication for Kafka brokers

Kafka supports TLS/SSL authentication (two-way authentication). To enable and configure TLS/SSL client authentication, you need to enable TLS/SSL encryption and set client authentication to be required by the brokers.

About this task

TLS/SSL authentication for Kafka brokers can be configured with the SSL Client Authentication property. The property has three valid values, required, requested, and none. If set to required, all clients connecting to the broker will be required to authenticate with TLS/SSL. If set to requested, authentication will be requested by the broker, but clients without certificates will still be able to connect. If set to none, no SSL authentication is required.

The authenticated user's identity is determined by how the SSL Client Authentication property is configured and whether a certificate is presented. The following table collects the possible outcomes:

Table 2: SSL Client Authentication options and outcomes

SSL Client Authentication	Client certificate is presented	Client certificate is not presented
required	• authentication: successful • authenticated user: *[***SUBJECT FROM THE CERTIFICATE***]*	• authentication: fails
requested	• authentication: successful • authenticated user: *[***SUBJECT FROM THE CERTIFICATE***]*	• authentication: successful • authenticated user: ANONYMOUS
none	• authentication: successful • authenticated user: ANONYMOUS	• authentication: successful • authenticated user: ANONYMOUS

If Ranger is used for authorization, the authenticated user's identity is used to determine what operations the client is authorized to carry out. As a result, you must ensure that policies in Ranger are set up accordingly.

Cloudera does not recommend that you set this property to requested. It is only useful in a limited number of scenarios and provides a false sense of security. Clients that present no certificates or present an invalid certificate will still be able to establish a connection, but will authenticate as ANONYMOUS. Depending on how your cluster is configured, the ANONYMOUS user might not have access to the required Kafka resources. This can lead to client failure.

Before you begin

Configure TLS/SSL encryption for the Kafka brokers. For more information, see Configure TLS/SSL encryption for Kafka brokers.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find and configure the SSL Client Authentication property based on your cluster and requirements.

Cloudera Manager Property	Description
SSL Client Authentication ssl.client.auth	Client authentication mode for SSL connections. This configuration has three valid values, required, requested, and none. If set to required, client authentication is required. If set to requested, client authentication is requested and clients without certificates can still connect. If set to none, which is the default value, no client authentication is required

Note: If you are using the SASL_SSL protocol, for example you authenticate with Kerberos and encrypt with TLS/SSL, and set SSL Client Authentication to required, Kafka will ignore the SSL Client Authentication property. In a case like this, SASL authentication takes precedence. Authenticating the client twice with both SASL and SSL would be redundant.

4. Configure principal mapping rules:
   a) Find the Kafka Broker Advanced Configuration Snippet (Safety Valve) for kafka.properties property.
   b) Add principal mapping rules.

   For example:

```
ssl.principal.mapping.rules=RULE:^.*[Cc][Nn]=([a-zA-Z0-9.]*).*$/$1/L,DEF
AULT
```

Note: For more information on principal mapping rules, see Principal name mapping.

5. Click Save Changes.
6. Restart the Kafka service.

Results

Kafka brokers are configured for TLS/SSL authentication.

What to do next

Configure your clients for TLS/SSL authentication.

Related Information

Configure TLS/SSL encryption for Kafka brokers

Principal name mapping

Configure TLS/SSL authentication for Kafka clients

Kafka supports TLS/SSL authentication (two-way authentication). Client configuration is done by setting the relevant security-related properties for the client.

About this task

The following steps demonstrate configuration for the console consumer or producer. If you are configuring a custom developed client, see Java client security examples or .Net client security examples for code examples.

Before you begin

- Generate or acquire a key and truststore for your clients which contain all necessary keys and certificates.
- Note down the locations and passwords for the key and truststores. You will need to provide these during configuration.
- If the Certificate Authority of the client certificates is different from the Certificate Authority of the broker certificates, ensure the client Certificate Authority certificate is added to the truststore of the Kafka brokers.

  Otherwise, the broker will not trust the certificates used by its clients and a trusted connection will not be established.

Procedure

1. Create a .properties file.

   In this example the file is named client.properties.
2. Add the mandatory properties to the file.

   The following configuration example contains all mandatory properties:

```
security.protocol=SSL
ssl.truststore.location=[***PATH TO CLIENT TRUSTSTORE***]
ssl.truststore.password=[***PASSWORD***]
ssl.keystore.location=[***PATH TO CLIENT KEYSTORE***]
ssl.keystore.password=[***PASSWORD***]
ssl.key.password=[***PASSWORD***]
```

3. (Optional) Add additional properties.

   Depending on your requirements and broker configuration, other configuration properties might also be needed. The following are some of the most commonly used optional properties:

   - ssl.provider
   - ss.cipher.suites
   - ssl.enabled.protocols
   - ssl.truststore.type
   - ssl.keystore.type
4. Run the client.

   Console Consumer

```
kafka-console-producer --bootstrap-server [***HOST1:PORT1***] --t
opic [***TOPIC***] --producer.config client.properties
```

   Console Producer

```
kafka-console-consumer --bootstrap-server [***HOST1:PORT1***] --t
opic [***TOPIC***] --consumer.config client.properties
```

Results

Kafka clients are configured for TLS/SSL authentication.

Related Information

Java client security examples

.Net client security examples

Principal name mapping

Kafka can be configured to translate certificate subject names into short names. This is done by adding mapping rules to Kafka's configuration. These short names can be used as the unique identifier of the user. Compared to subject names, short names are much easier to manage.

When a client authenticates using a TLS/SSL keystore, by default Kafka assumes that the username for that client is the certificate's subject name, which is usually a Distinguished Name such as the following:

```
cn=alice,cn=groups,cn=accounts,dc=hadoopsecurity,dc=local
```

Working with these long names is difficult. Security policies and group mappings are usually defined in terms of the user's short name (alice) rather than the full Distinguished Name. Kafka can be configured to translate the certificate's subject into a short name that can be used as the unique identifier of the user.

This can be done by adding the necessary mapping rules to the ssl.principal.mapping.rules Kafka property. However, this property is not directly configurable in Cloudera Manager. As a result, you need to use the Kafka Broker Advanced Configuration Snippet (Safety Valve) for kafka.properties property to add it to your configuration.

Note: The ssl.principal.mapping.rules property is only supported in Kafka 2.4.0 or higher. In older versions of Kafka, a custom principal builder needs to be created and provided.

The rule takes the form of a regular expression to match the subject name of the certificate and the transformation to apply to the match. The property accepts multiple rules. Each rule has to be separated by a comma. The last rule is usually the DEFAULT rule, which uses the full subject name.

For example, consider the following setting:

```
ssl.principal.mapping.rules=RULE:^.*[Cc][Nn]=([a-zA-Z0-9.]*).*$/$1/L,DEFAULT
```

This configuration has two rules which are processed in the following order:

1. RULE:^[Cc][Nn]=([a-zA-Z0-9.]*).*$/$1/L
2. DEFAULT

The first rule to match the certificate's subject name is used, later ones are ignored. The DEFAULT rule is a "catch all" rule. It always matches and does not do any replacement if none of the previous ones were matched.

The regular expression of the first rule,^[Cc][Nn]=([a-zA-Z0-9.]*).*$, matches any subject that starts with CN=,cn=, Cn=, or cN=, followed by the user's short name, that contains characters ranging between a-z,A-Z, and 0-9, followed by any string. It then replaces the matched string with the user's short name. The short name is the content matched inside the parenthesis and is referenced in the second part of the rule as $1. The L at the end of the rule converts the resulting string to lowercase.

For more information and examples on principal mapping rules, see the Apache Kafka documentation.

Related Information

Apache Kafka documentation

Enable Kerberos authentication for Kafka

Learn how to enable Kerberos Authentication for Kafka.

About this task

Apache Kafka supports Kerberos authentication, but it is supported only for the new Kafka Producer and Consumer APIs.

Before you begin

If you already have a Kerberos server, you can add Kafka to your current configuration. If you do not have a Kerberos server, install it before proceeding.

If you already have configured the mapping from Kerberos principals to short names using the hadoop.security.auth_to_local HDFS configuration property, configure the same rules for Kafka by adding the sasl.kerberos.principal.to.local.rules property to the Advanced Configuration Snippet for Kafka Broker Advanced Configuration Snippet using Cloudera Manager. Specify the rules as a comma separated list.

Procedure

1. In Cloudera Manager, navigate to Kafka Configuration.
2. Set SSL Client Authentication to none.
3. Set Inter Broker Protocol to SASL_PLAINTEXT.
4. Click Save Changes.
5. Restart the Kafka service, select Action Restart.
6. Make sure that listeners = SASL_PLAINTEXT is present in the Kafka broker logs, by default in /var/log/kafka/server.log.
7. Create a jaas.conf file with either cached credentials or keytabs.
   - To use cached Kerberos credentials, where you use kinit first, use this configuration:

     ```
     KafkaClient {
     com.sun.security.auth.module.Krb5LoginModule required
     useTicketCache=true;
     };
     ```

   - If you use a keytab, use this configuration:

     ```
     KafkaClient {
     com.sun.security.auth.module.Krb5LoginModule required
     useKeyTab=true
     keyTab="/etc/security/keytabs/mykafkaclient.keytab"
     principal="mykafkaclient/clients.hostname.com@EXAMPLE.COM";
     };
     ```

8. Create the client.properties file containing the following properties.

   ```
   security.protocol=SASL_PLAINTEXT
   sasl.kerberos.service.name=kafka
   ```

9. Test with the Kafka console producer and consumer.

   To obtain a Kerberos ticket-granting ticket (TGT):

   ```
   kinit USER
   ```

10. Verify that the jaas.conf file is used by setting the environment.

    ```
    export KAFKA_OPTS="-Djava.security.auth.login.config=/home/user/jaas.conf"
    ```

11. Verify that your topic exists.

    ```
    kafka-topics --list --bootstrap-server ANYBROKER:9092 --command-config client.properties
    ```

12. Run a Kafka console producer.

```
kafka-console-producer --broker-list ANYBROKER:9092 --topic test1 --produc
er.config client.properties
```

13. Run a Kafka console consumer.

```
kafka-console-consumer --topic test1 --from-beginning --bootstrap-se
rver ANYBROKER:9092 --consumer.config client.properties
```

Delegation token based authentication

An overview on Kafka delegation tokens.

Delegation tokens are lightweight authentication method to complement existing SASL authentication. Kafka is designed to support a high number of clients. However, using Kerberos authentication might be difficult in some environments due to the following reasons:

- With Kerberos authentication, all clients need access to a keytab or a TGT. Securely distributing the keytabs requires a lot of effort and careful administration. When the TGT is compromised, it has a high blast radius, especially when the same keytabs are used to access multiple services.
- With Kerberos, client authentication is centralized, and the high number of clients can put a high load on the KDC (Key Distribution Center), resulting in a bottleneck.

Many Hadoop components use delegation tokens to mitigate these problems:

- Delegation tokens allow these components to secure distributed workloads with low administrative overhead.
- It is not required to distribute a Kerberos TGT or keytab, which, if compromised, may grant access to all services.
- A delegation token is strictly tied to its associated service causing less damage if exposed.
- Delegation tokens make credential renewal more lightweight. This is because the renewal is designed in such a way that only the renewer and the service are involved in the renewal process. The token itself remains the same, so parties already using the token do not have to be updated.

Kafka Delegation Token Basics

Kafka delegation tokens are modeled after Hadoop delegation tokens, and many of their mechanism are the same or very similar. However, this does not mean that they are interchangeable.

Delegation tokens are generated and verified following the HMAC mechanism. There are two basic parts of information in a delegation token, the tokenID (public part) and the HMAC value (private part).

The following list of steps give a generic overview on how delegation tokens are used:

1. The user initially authenticates with the Kafka cluster via SASL, and obtains a delegation token using either the AdminClient APIs or the kafka-delegation-token tool. The principal that created the delegation token is its owner.
2. The delegation token details are securely passed to Kafka clients. This can be achieved by sending the token data over an SSL/TLS encrypted connection or writing them to a secure shared storage.
3. Instead of using Kerberos, the Kafka client uses the delegation tokens for subsequent authentication with the brokers.
4. The token is valid for a certain time period, but it can be:

   Renewed

   A delegation token can be renewed multiple times up until its maximum life before it expires. The token can be renewed by the owner or any other principals the owner sets as “renewer” at time of creation.

   Revoked

   A delegation token can be revoked ahead of its expiry time.

Broker Configuration

Certain delegation token properties can be configured on a service level in Cloudera Manager. You can enable or disable delegation tokens, as well as configure expiry time and maximum lifetime. For more information, see Cloudera Manager Configuration Properties Reference.

Related Information

HMAC Wikipedia Page

Client authentication using delegation tokens

Cloudera Manager Configuration Properties Reference

Enable or disable authentication with delegation tokens

Learn how to enable or disable Kafka delegation tokens.

About this task

Delegation token based authentication requires that both the Enable Kerberos Authentication and Enable Delegation Tokens properties are set to true. The default value of the Enable Delegation Tokens property is true, but will have no effect until Kerberos is also enabled.

Although enabling delegation tokens enables authentication between clients and servers using the SASL/SCRAM mechanism, it is only as a vehicle for delegation tokens. Using SCRAM credentials is not supported otherwise.

Sensitive delegation token metadata is stored in Zookeeper. It is recommended to restrict access on Zookeeper nodes to prevent access to sensitive delegation token related data through Zookeeper. The connection between Kafka and Zookeeper is not encrypted, therefore, it is also recommended to use delegation tokens only if no unauthorized person can read and manipulate the traffic between these services.

Delegation tokens can be enabled separately for each Kafka service.

Before you begin

A secure Kafka cluster with Kerberos authentication enabled is required for delegation tokens to function.

Procedure

1. In Cloudera Manager select the Kafka service.
2. Select Configuration and find the Enable Delegation Tokens property.
3. Enable or disable delegation tokens for all required services by checking or unchecking the checkbox next to the name of the service.
4. Click Save Changes.
5. Perform a Rolling Restart:
   a) Return to the Home page by clicking the Cloudera Manager logo.
   b) Go to the Kafka service and select ActionsRolling Restart.
   c) Check the Restart roles with stale configurations only checkbox and click Rolling restart.
   d) Click Close when the restart has finished.

Results

Delegation tokens are enabled or disabled for the selected Kafka services. If delegation tokens were enabled, then the necessary secrets and settings are generated.

Related Information

Enable Kerberos authentication for Kafka

Kafka security hardening with Zookeeper ACLs

Manage individual delegation tokens

A list of example actions you can issue to manage individual delegation tokens.

The functionality that's needed to manage and use delegation tokens is accessible using the AdminClient APIs or the kafka-delegation-tokens tool. All of their operations are allowed only via SASL authenticated channels.

Both the API and the script provide the following actions:

Note: The examples presented show how these actions can be executed with the kafka-delegation-tokens tool.

Issue, and store for verification

The owner of the token is the currently authenticated principal. A renewer can be specified when requesting the token.

```
kafka-delegation-tokens --bootstrap-server HOSTNAME:PORT  --crea
te   --max-life-time-period -1 --command-config CLIENT.PROPERTIES
 --renewer-principal USER:USER1
```

Renew

Only the owner and the principals that are renewers of the delegation token can extend its validity by renewing it before it expires. A successful renewal extends the Delegation Token's expiration time for another renew-interval, until it reaches its max lifetime. Expired delegation tokens cannot be used to authenticate, the brokers will remove expired delegation tokens from the broker's cache and from Zookeeper.

```
kafka-delegation-tokens --bootstrap-server HOSTNAME:PORT --renew
    --renew-time-period -1 --command-config CLIENT.PROPERTIES --h
mac LAYYSFMLS4BTJF+LTZ1LCHR/ZZFNA==
```

Remove

Delegation tokens are removed when they are canceled by the client or when they expire.

```
kafka-delegation-tokens --bootstrap-server HOSTNAME:PORT --expir
e   --expiry-time-period -1   --command-config CLIENT.PROPERTIES
 --hmac LAYYSFMLS4BTJF+LTZ1LCHR/ZZFNA==
```

Describe

Tokens can be described by owners, renewers or the Kafka super user.

```
kafka-delegation-tokens --bootstrap-server HOSTNAME:PORT --
describe --command-config CLIENT.PROPERTIES  --owner-princi
pal USER:USER1
```

Note: In Apache Kafka, principals that have the describe permission on the token resource can also describe the token.

Rotate the master key/secret

Learn how to rotate the delegation token Master Key/Secret.

About this task

The brokers generate and verify delegation tokens using a secret called delegation.token.master.key. This secret is generated by Cloudera Manager and securely passed to Kafka brokers when authentication with delegation tokens is enabled. You can change the secret with the Cloudera Manager API. This should be done if the secret becomes compromised, or simply as a precautionary measure.

Important: Clients that were already connected to brokers before starting the process, will continue to work even after the master key/secret is rotated. However, any new connections (authentication requests), as well as renew and expire requests with old tokens can fail.

Procedure

1. Expire existing tokens.

```
kafka-delegation-tokens --bootstrap-server HOSTNAME:PORT --expire --expi
ry-time-period -1 --command-config CLIENT.PROPERTIES --hmac LAYYSFMLS4BTJF
+LTZ1LCHR/ZZFNA==
```

2. Generate a new master key

```
curl -X PUT -u "USER" -H "content-type:application/json" -i "htt
ps://CLOUDERA MANAGER HOST:7183/api/v31/clusters/CLUSTER NAME/service
s/KAFKA SERVICE NAME/config" -d '{"items" : [ {"name" : "delegation.toke
n.master.key","value" : "'$(openssl rand -base64 24)'","sensitive" : tru
e}]}'
```

3. Perform a Rolling Restart:
   a) In Cloudera Manager go to the Kafka service and select ActionsRolling Restart.
   b) Check the Restart roles with stale configurations only checkbox and click Rolling restart.
   c) Click Close when the restart has finished.
4. Reauthenticate with all clients.

   This generates the new tokens.

Results

The Master Key/Secret is rotated.

Client authentication using delegation tokens

An overview of the methods available for configuring clients to use delegation tokens.

Brokers authenticate clients by verifying the delegation tokens provided by the client against the stored delegation tokens. Delegation token authentication makes use of SASL/SCRAM authentication mechanism under the hood. You can configure Kafka clients in two ways, to use individually assigned delegation tokens or to use a common delegation token.

Configure clients on a producer or consumer level

Learn how to configure client authentication using delegation tokens on a producer or consumer level.

About this task

You can set up client authentication by configuring the JAAS configuration property as well as other mandatory properties for each client. All properties can be set in the producer.properties or consumer.properties file of the client. With this configuration method, you have the ability to specify different token details for each Kafka client within a JVM. As a result you can configure Kafka clients in a way that each of them use a unique token for authentication.

Procedure

Configure each client.

Example Configuration:

```
sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule  re
quired \
    username="TOKENID" \
    password="LAYYSFMLS4BTJF+LTZ1LCHR/ZZFNA==" \
    tokenauth="TRUE";
security.protocol=SASL_SSL
sasl.mechanism=SCRAM-SHA-256
```

Within the JAAS configuration, there are three options that need to be specified. These are the username, password and tokenauth options.

The username and password options specify the token ID and token HMAC. The tokenauth option expresses the intent to use token authentication to the server.

Configure clients on an application level

Learn how to configure client authentication using delegation tokens on an application level.

About this task

With this configuration method, you can set up all clients within a JVM to use the same delegation token for authentication.

Procedure

1. Add a KafkaClient entry with a login module item to your JAAS configuration file.

   The module has to specify the username, password and tokenauth options.

   Example Configuration:

   ```
   KafkaClient {
   org.apache.kafka.common.security.scram.ScramLoginModule  required
       username="TOKENID"
       password="LAYYSFMLS4BTJF+LTZ1LCHR/ZZFNA=="
       tokenauth="TRUE";
   }
   ```

   The username and password options specify the token ID and token HMAC. The tokenauth option expresses the intent to use token authentication to the server.

2. Configure the following properties for your clients.

   These properties are added to producer.properties or consumer.properties file that the client uses.

   ```
   security.protocol=SASL_SSL
   sasl.mechanism=SCRAM-SHA-256
   ```

3. Pass the location of your JAAS configuration file as a JVM parameter through a command line interface.

   This sets the JAAS configuration on the Java process level.

   ```
   export KAFKA_OPTS="-Djava.security.auth.login.config=[PATH_TO_JAAS.CONF]"
   ```

LDAP authentication

You can configure Kafka to make use of LDAP credentials for client to broker authentication. In order to enable LDAP authentication you must configure both brokers and clients accordingly.

Configure LDAP authentication for Kafka brokers

Learn how to configure LDAP authentication for Kafka brokers.

About this task

You can enable Kafka to use LDAP credentials for client to broker authentication. Broker configuration is done by configuring the required properties in Cloudera Manager.

Before you begin

Important: Kafka Brokers provisioned after July 28, 2022 use LDAP authentication by default. Previously PAM was the default. If your cluster was provisioned following this date, LDAP is already configured and you do not need to complete the following steps. If you are transitioning from PAM to LDAP, you might need to reconfigure your clients and ensure that your clients use short names. Specifying usernames in the client configuration that contain the realm/domain are not supported.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Select Configuration.
3. Enable LDAP authentication:
   a) Find the SASL/PLAIN Authentication property.
   b) Click the radio button next to LDAP. Do this for all required Kafka services.
4. Click Save Changes.
5. Restart the Kafka service.

Results

LDAP authentication is configured for the brokers.

What to do next

Configure clients to use LDAP authentication.

Related Information

Configure Kafka LDAP authentication for Kafka clients

Cloudera Blog – How to configure clients to connect to Apache Kafka Clusters securely – Part 2: LDAP

Configure Kafka LDAP authentication for Kafka clients

Learn how to configure Kafka clients for LDAP authentication.

About this task

You can enable Kafka to use LDAP credentials for client to broker authentication. Client configuration is done by adding the required properties to the client's client.properties file.

Before you begin

- Clients connecting to Cloudera Data Hub provisioned clusters require a Cloudera user account that provides access to the required Cloudera resources. Verify that a Cloudera user account with the required roles and permissions is available for use. If not, create one. Any type of Cloudera user account can be used. If you are creating a new account to be used by Kafka clients, Cloudera recommends that you create a machine user account. For more information, see User Management in the Cloudera Management Console documentation.
- In addition to the Cloudera user account having access to the required Cloudera resources, the user account also needs to have the correct policies assigned to it in Ranger. Otherwise, the client cannot perform tasks on Kafka resources. These policies are specified within the Ranger instance that provides authorization to the Kafka service you want to connect to. For more information, see the Cloudera Runtime documentation on Apache Ranger and the Kafka specific Ranger documentation.
- If you are transitioning from PAM to LDAP, ensure that the usernames you specify in the client configuration are short names. Usernames that include the realm/domain are not supported.

Procedure

1. Set the SASL mechanism to PLAIN.

   Add the following property to the client.properties file.

   ```
   sasl.mechanism=PLAIN
   ```

2. Configure the security protocol.

   You can either use SASL_SSL or SASL_PLAINTEXT. Which security protocol you use will depend on whether or not SSL encryption is enabled on the broker. Add one of the following properties to the client.properties file.

   - If encryption is enabled, use SASL_SSL:

     ```
     security.protocol=SASL_SSL
     ```

   - If encryption is not enabled, use SASL_PLAINTEXT:

     ```
     security.protocol=SASL_PLAINTEXT
     ```

   Note: In a public network Cloudera recommends that you use SASL_SSL as LDAP credentials can become exposed.

3. Configure the JAAS.

   You have two options when configuring the JAAS:

   a) Embed the required properties in the client.properties file with the sasl.jaas.config property.

   ```
   sasl.jaas.config= \
   org.apache.kafka.common.security.plain.PlainLoginModule required \
       username="[***USERNAME***]" \
       password="[***PASSWORD***]";
   ```

   Replace *[***USERNAME***]* and *[***PASSWORD***]* with the credentials of the Cloudera user you set up for the client in Management Console. For the username, ensure that you use short names. Specifying usernames that contain the realm/domain are not supported.

   b) Use a separate JAAS config file:

   1. Add a KafkaClient entry with a login module item to your JAAS configuration file.

      Example configuration:

      ```
      KafkaClient {
           org.apache.kafka.common.security.plain.PlainLoginModule required
           username="[***USERNAME***]"
           password="[***PASSWORD***]";
          };
      ```

      Replace *[***USERNAME***]* and *[***PASSWORD***]* with the credentials of the Cloudera user you set up for the client in Management Console. For the username, ensure that you use short names. Specifying usernames that contain the realm/domain are not supported.

   2. Pass the location of your JAAS configuration file as a JVM parameter through a command line interface

      ```
      export KAFKA_OPTS="-Djava.security.auth.login.config=[***PATH TO
       JAAS.CONF***]"
      ```

      Replace *[***PATH TO JAAS.CONF***]* with the location of the JAAS configuration file you created.

Results

LDAP authentication is configured for the client.

PAM authentication

You can configure Kafka to use the PAM authentication mechanism for client to broker authentication. In order to enable PAM authentication you must configure both brokers and clients accordingly.

Configure PAM authentication for Kafka brokers

Learn how to configure PAM authentication for Kafka brokers.

About this task

You can enable Kafka to use PAM for client to broker authentication. Broker configuration is done by configuring the required properties in Cloudera Manager.

Procedure

1. In Cloudera Manager select the Kafka service.
2. Select Configuration.
3. Enable PAM authentication:
   a) Find the SASL/PLAIN Authentication property.
   b) Click the radio button next to PAM. Do this for all required Kafka services.
4. Configure the PAM service name:
   a) Find the PAM Service property.
   b) Enter a valid PAM service name. The property defaults to login.
5. Click Save Changes.
6. Restart the Kafka service

Results

PAM authentication is configured for the brokers.

What to do next

Configure clients to use PAM authentication.

Configure PAM authentication for Kafka clients

Learn how to configure Kafka clients for PAM authentication.

About this task

You can enable Kafka to use PAM for client to broker authentication. Client configuration is done by adding the required properties to the client's client.properties file.

Procedure

1. Set the SASL mechanism to PLAIN.

   Add the following property to the client.properties file.

```
sasl.mechanism=PLAIN
```

2. Configure the security protocol.

   You can either use SASL_SSL or SASL_PLAINTEXT. Which security protocol you use will depend on whether or not SSL encryption is enabled on the broker. Add one of the following properties to the client.properties file.

   - If encryption is enabled, use SASL_SSL:

     ```
     security.protocol=SASL_SSL
     ```

   - If encryption is not enabled, use SASL_PLAINTEXT:

     ```
     security.protocol=SASL_PLAINTEXT
     ```

     Note: In a public network Cloudera recommends that you use SASL_SSL as credentials can become exposed.

3. Configure the JAAS.

   You have two options when configuring the JAAS:

   a) Embed the required properties in the client.properties file with the sasl.jaas.config property.

      ```
      sasl.jaas.config= \
      org.apache.kafka.common.security.plain.PlainLoginModule required \
          username="[USERNAME]" \
          password="[PASSWORD]";
      ```

      Replace *[USERNAME]* and *[PASSWORD]* with a valid username and password.

   b) Use a separate JAAS config file:

      1. Add a KafkaClient entry with a login module item to your JAAS configuration file.

         Example configuration:

         ```
         KafkaClient {
              org.apache.kafka.common.security.plain.PlainLoginModule required
              username="[USERNAME]"
              password="[PASSWORD]";
             };
         ```

         Replace *[USERNAME]* and *[PASSWORD]* with a valid username and password.

      2. Pass the location of your JAAS configuration file as a JVM parameter through a command line interface

         ```
         export KAFKA_OPTS="-Djava.security.auth.login.config=[PATH_TO_JAAS.C
         ONF]"
         ```

         Replace *[PATH_TO_JAAS.CONF]* with the location of the JAAS configuration file you created.

Results

PAM authentication is configured for the client.

OAuth2 authentication

Learn about OAuth2 authentication and how OAuth2 is implemented in Apache Kafka.

OAuth2 is a set of open standards used for access delegation. It is used for enabling applications and services to access other services or resources. Using OAuth2 in Kafka is made possible by an OAuth2-compatible token-based mechanism called SASL OAUTHBEARER. The OAUTHBEARER implementation in Cloudera Kafka uses and accepts signed JSON Web Tokens (JWTs). The mechanism can be used for client to broker authentication.

Important: Inter-broker authentication using OAuth2 is not supported.

To ensure that the information transferred through JWTs is trusted, that is, to make JWTs tamper-proof, they can be signed. JWTs can be signed with a secret (using the HMAC algorithm) or with a key pair consisting of a public and private key using RSA or ECDSA. While both signature methods are viable options, Cloudera recommends that you use RSA or ECDSA. This is because using RSA or ECDSA also certifies that the token is issued by the authorization server.

Note: Cloudera Kafka only supports the client_credentials grant type for access tokens.

The authentication flow involves the client, an OAuth2 authorization server, and the Kafka broker.

Figure 1: OAuth2 authentication flow

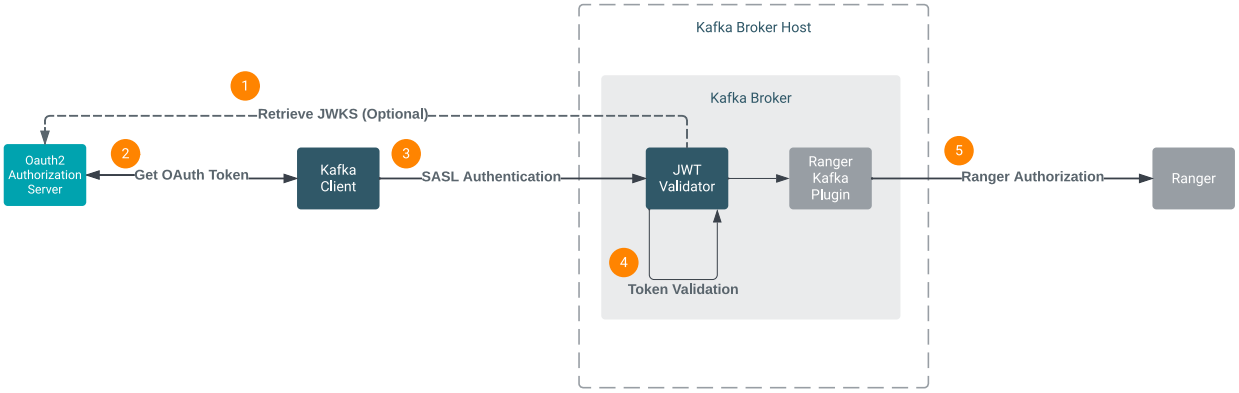

1. The Kafka Broker retrieves the JSON Web Key Set (JWKS) from the authorization server.

   JWKS retrieval happens at broker startup and is optional. It is only done if the broker is configured for online JWKS access. The broker can also be configured to access a locally available JWKS (not shown on the diagram).
2. The client obtains a signed JWT token from the authorization server.
3. The client initiates a connection to the broker using the JWT token.
4. The Kafka broker verifies the JWT token.
5. If Ranger authorization is enabled for the Kafka service, the claim within the token containing the user ID (principal) is checked against the policies configured for that user in Ranger. The client is only allowed to carry out permitted operations.

Token verification

The broker verifies the tokens by checking the signature of the token as well as the claims found within the token. The broker can verify the following claims:

- “iat” (issued at/timestamp) and “exp” (expiry): Used to verify that the token is valid and has not expired.
- "iss” (issuer): Used to verify that the token was issued by a trusted issuer. The broker only verifies this claim if it is configured to do so. This is done by providing the broker with an “iss” claim value that it should accept. If the broker is not provided with an “iss” claim value, it does not verify the claim.
- “aud” (audience/intended receiver): Used to verify that the token was created specifically for accessing the broker. If the "aud" claim is present in the JWTs presented to the broker, the broker must be provided with the "aud" claim value found within the tokens. Otherwise, the token is considered invalid.

JWKS access

In order for the broker to be able to verify signed JWTs, it must have access to a JWKS. The JWKS is a JSON object that contains the keys which the authorization server uses to sign JWTs. Kafka brokers can be provided access to a JWKS in two different ways. These are as follows:

Note: The two JWKS access options are mutually exclusive. If you configure the broker for both online and local JWKS access, the broker uses local JWKS access.

Online JWKS access

Kafka brokers can retrieve the JWKS directly from the authorization server. If an endpoint is made available to the broker, using appropriate configuration property, the broker connects to that endpoint at startup and fetches the JWKS.

For online JWKS access to work, the broker must be able to access the endpoint through the network. As a result, you might need to set up firewall rules. Additionally, because the broker attempts to retrieve the JWKS at startup, if for example the authorization server is inaccessible or returns an error, the broker will fail to start.

Local JWKS access

If you do not want to rely on the availability of the authorization server to retrieve the JWKS or if establishing a connection between the broker and the authorization server is not possible, you can choose to have the Kafka broker use a JWKS that is available locally on the cluster. For this method to work, you must provide the JWKS JSON directly to the Kafka broker using an advanced configuration snippet.

Configuring OAuth2 authentication for Kafka brokers

Learn how to enable and configure OAuth2 authentication for Kafka brokers.

About this task

You can enable Kafka to use OAuth2 for client to broker authentication. Broker configuration is done by configuring OAuth2 related properties. OAuth2 related properties can be configured using Cloudera Manager. However, there are two distinct paths of configuration. Which configuration path you take depends on whether the Kafka service is Kerberos enabled.

If Kerberos is enabled, you can use the OAuth2 related properties directly available in Cloudera Manager to enable and configure OAuth2. If Kerberos is not enabled, you must use an advanced configuration snippet to enable and configure OAuth2.

In addition, when configuring OAuth2 you can choose how the broker gets access to the JSON Web Key Set (JWKS) used to verify tokens. Kafka brokers can be provided access to a JWKS in two different ways. They can be configured to retrieve the JWKS from the authorization server's JWKS endpoint (HTTPS) at startup. This is called online JWKS access. Alternatively, it is also possible to provide the broker directly with the JWKS used by the authorization server using an advanced configuration snippet. This is called local JWKS access.

Note: The two JWKS access options are mutually exclusive. If you configure the broker for both online and local JWKS access, the broker uses local JWKS access.

Before you begin

- If you want to configure online JWKS access, ensure that the broker can open a network connection to the authorization server. Otherwise the broker will fail to start.
- If you want to configure local JWKS access, acquire the JWKS JSON from the authorization server.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.

3. Enable and configure Oauth2 authentication.

   OAuth2 related properties are configured in either of two ways. If Kerberos is enabled for the Kafka service, you can use the properties directly available in Cloudera Manager. In a case like this follow the instructions in the Using dedicated properties tab. If Kerberos is not enabled for the Kafka service, you must use an advanced configuration snippet. In a case like this, follow the instruction in the Using an advanced configuration snippet tab.

   For Using dedicated properties

   a. Find and configure the following properties:

   Note: The following properties do not take effect if Kerberos is not enabled for the Kafka service. Follow the instructions in the Using an advanced configuration snippet tab if you do not have Kerberos enabled

   Table 3: OAuth properties

Cloudera Manager Property	Description
Enable OAuth Authentication oauth.enabled	Enables authentication using the SASL OAUTHBEARER mechanism for this Kafka service. Clients must present a signed JSON Web Token (JWT) when authenticating with OAuth2. Kerberos must be enabled for this property to take effect.
JWKS URL For OAuth2 oauth.jwks.url	The endpoint URL that returns the authorization server's public keys in JWKS format. Configure a valid JWKS endpoint URL if you want to configure the broker for online JWKS access. If you want to configure the broker for local JWKS access, leave this property empty and instead configure the Kafka Broker Advanced Configuration Snippet (Safety Valve) for oauth_jwks.json property.
Kafka Broker Advanced Configuration Snippet (Safety Valve) for oauth_jwks.json oauth_jwks.json_role_safety_valve	Specifies the JWKS used by this Kafka service to verify JWTs presented by clients. Add the JWKS JSON you obtained from the authorization server's JWKS endpoint. Set this property if you want to configure the broker for local JWKS access. If you want to configure the broker for online JWKS access, leave this property empty and instead configure the JWKS URL For OAuth2 propert.
JWT Expected Audience For OAuth2 oauth.expected.audience	The JWT token can optionally contain an"aud" (audience/intended receiver) claim. When this claim is present, the same audience value must be specified for the broker, otherwise the token will be considered invalid. This property specifies what the value of the "aud" claim is. Configuring this property is only required if the "aud" claim is present in the tokens presented to the broker.
JWT Expected Issuer For OAuth2 oauth.expected.issuer	The JWT token can optionally contain an "iss" (issuer) claim. This property specifies what the value of the "iss" claim is. Specifying an "iss" claim configures the broker to only accept tokens issued by a specific issuer. Leave this property empty if you do not want the broker to validate the "iss" claim.
JWT Principal Claim Name For OAuth2 oauth.principal.claim.name	A JWT token must contain a user ID (principal) that can be used for Ranger authorization. This property specifies the name (or key) of the claim in the JWT token that contains the user ID. The "sub" (subject) claim is used if this property is left empty. Configuring this property is mandatory if the JWT token does not include a "sub" claim.

b. Optional: Configure additional properties.

If required, additional OAuth2 related properties can be configured with the Kafka Broker Advanced Configuration Snippet (Safety Valve) for kafka.properties property. See *Broker Configs* in the Apache Kafka documentation for a full list of these properties.

For Using an advanced configuration snippet

a. Find and configure the Kafka Broker Advanced Configuration Snippet (Safety Valve) for kafka.properties property.

Use the following example as a template and make changes as necessary. The following example contains all minimum required OAuth2 properties. If required, additional OAuth2 related properties can be configured as well. See *Broker Configs* in the Apache Kafka documentation for a full list of these properties.

Example configuration:

```
listeners=PLAINTEXT://:9092,SASL_PLAINTEXT://:34343
listener.name.sasl_plaintext.sasl.enabled.mechanisms=OAUTHBEARER
listener.name.sasl_plaintext.oauthbearer.sasl.server.callback.handl
er.class=org.apache.kafka.common.security.oauthbearer.secured.OAuthB
earerValidatorCallbackHandler
listener.name.sasl_plaintext.oauthbearer.sasl.jaas.config=org.apac
he.kafka.common.security.oauthbearer.OAuthBearerLoginModule required;
sasl.oauthbearer.jwks.endpoint.url=https://my-authorization-server/
jwks
```

4. Click Save Changes.

5. Restart the Kafka service.

Results

The broker is ready to accept connections authenticated with OAuth2. Depending on how you configured the broker, it either retrieves the JWKS from the authorization server at startup, or loads the JWKS keys from local storage.

What to do next

Configure your Kafka clients to use OAuth2 authentication. Continue with *Configuring OAuth authentication for Kafka clients*.

Related Information

Broker Configs

Configuring OAuth2 authentication for Kafka clients

OAuth2 configuration examples

Configuring OAuth2 authentication for Kafka clients

Learn how to enable and configure OAuth authentication for Kafka clients.

About this task

In order for the clients to successfully establish a connection with a broker that has OAuth2 authentication enabled, you must configure OAuth2 related properties for the client. Configuring these properties instructs the client to acquire a signed JSON Web Token (JWT) and present that token to the broker when requesting access.

Note: Cloudera Kafka only supports the client_credentials grant type for access tokens.

The following steps demonstrate configuration for console clients. This is done by creating a client.properties configuration file that includes the required properties. If you are configuring a custom developed client, see *Java client security examples* or *.Net client security examples* for code examples.

Before you begin

- Ensure that the authorization server is reachable from the client host.
- Obtain the authorization server's token endpoint URL.
- Obtain the necessary credentials (client ID, client secret) and other parameters (for example, scope) from the authorization server.
- Cloudera recommends that you configure your authorization servers as well as your Kafka brokers to use TLS/SSL. This is recommended so that JWT tokens and client credentials do not get exposed.

Procedure

1. Create a client.properties file containing the following properties:

```
security.protocol=[***SECURITY PROTOCOL***]
sasl.mechanism=OAUTHBEARER
sasl.login.callback.handler.class=org.apache.kafka.common.security.oauth
bearer.secured.OAuthBearerLoginCallbackHandler
sasl.oauthbearer.token.endpoint.url=http://[***OAUTH SERVER***]/[***TOKEN
 ENDPOINT***]
```

   Replace *[***SECURITY PROTOCOL***]* with either SASL_SSL or SASL_PLAINTEXT. The security protocol you specify depends on whether TLS/SSL encryption is enabled on the broker.

2. If TLS/SSL is enabled on the broker, add all required TLS/SSL properties to the client.properties file. For example:

```
ssl.truststore.location= [***PATH TO CLIENT TRUSTSTORE***]
ssl.truststore.password=[***PASSWORD***]
```

   This example contains the minimum required TLS/SSL properties. Depending on your requirements and how TLS/SSL is configured on the broker, other properties might be required. For more information regarding TLS/SSL configuration, see *Channel Encryption*.

3. Configure the JAAS.

   You have two options when configuring the JAAS. You can either embed the full JAAS configuration in the client.properties file or use a separate JAAS configuration file.

   a) Embed the required properties in the client.properties file with the sasl.jaas.config property.

```
sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthB
earerLoginModule required clientId="[***CLIENT ID***]" clientSecr
et="[***CLIENT SECRET***]" scope="[***SCOPE***]";
```

   b) Use a separate JAAS config file:

      1. Add a KafkaClient entry with a login module item to your JAAS configuration file.

         You can also create a new JAAS configuration file if you do not have an existing one available.

```
KafkaClient {
  org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginMod
ule required
  clientId="[***CLIENT ID***]"
  clientSecret="[***CLIENT SECRET***]"
  scope="[***SCOPE***]";
};
```

      2. Pass the location of your JAAS configuration file as a JVM parameter through a command line interface

```
export KAFKA_OPTS="-Djava.security.auth.login.config=[***PATH TO
 JAAS.CONF***]"
```

Results

Clients that you run with the configuration file you created authenticate themselves to the Kafka broker with OAuth2.

What to do next

Run a client:

Console consumer

```
kafka-console-consumer --bootstrap-server [***HOST1:PORT1***] --t
opic [***TOPIC***] --consumer.config client.properties
```

Console producer

```
kafka-console-producer --bootstrap-server [***HOST1:PORT1***] --t
opic [***TOPIC***] --producer.config client.properties
```

Related Information

Java client security examples

.Net client security examples

Channel encryption

OAuth2 configuration examples

Authorization

Ranger

Learn more about authorization with Ranger.

You can use Ranger to provide authorization for Kafka. For more information, see *Using Ranger to Provide Authorization in Cloudera*.

Related Information

Using Ranger to Provide authorization in Cloudera

Enable authorization in Kafka with Ranger

Learn how to enable Ranger authorization for Kafka.

About this task

The following instructions can be used to enable and configure Ranger authorization for Kafka where the Ranger service is either deployed on the same cluster as the Kafka service or if the Ranger service is deployed in a Data Context cluster.

Before you begin

- Ranger authorization requires that at least one of the following authentication mechanisms is enabled in Kafka:
  - Kerberos
  - Two-way TLS/SSL
  - LDAP
  - PAM

- It is also possible to have a Kafka service depend on a Ranger service which is deployed on a remote, non Data Context cluster. This is achieved by configuring the following advanced configuration snippets instead of the configuration steps described below:
  - Kafka Broker Advanced Configuration Snippet (Safety Valve) for ranger-kafka-security.xml

    ```
    Name: ranger.plugin.kafka.policy.rest.url
    Value: http://[****FQDN OF RANGER ADMIN HOST***]:6080/
    ```

  - Kafka Broker Advanced Configuration Snippet (Safety Valve) for ranger-kafka-audit.xml

    ```
    Name: xasecure.audit.destination.solr.zookeepers
    Value: [***FQDN OF ZOOKEEPER HOST***]:2181/solr-infra
    ```

Procedure

1. In Cloudera Manager select the Kafka service.
2. Select Configuration and find the RANGER Service property.
3. Check the checkbox next to the name of the Ranger service that you want this Kafka service to depend on.
4. Click Save Changes.
5. Restart the Kafka service.

Results

Ranger authorization for Kafka is enabled. The Kafka service depends on the selected Ranger service for authorization.

What to do next

Configure resource-based services and policies for Kafka. Additionally, configure which resource-based service should be used for authorization.

Related Information

Configure a resource-based service: Kafka

Configure a resource-based policy: Kafka

Configure the resource-based Ranger service used for authorization

Enable Kerberos authentication for Kafka

Channel encryption

Configure TLS/SSL client authentication for Kafka brokers

LDAP authentication

PAM authentication

Configure the resource-based Ranger service used for authorization

Learn how to configure the resource-based Ranger service used by Kafka for authorization.

About this task

You can configure which resource-based service in Ranger is used by Kafka for authorization. This is controlled by the Ranger service name for this Kafka cluster property which can be set in Cloudera Manager. The property can be configured in two ways:

Set a user defined string

The property accepts a user defined string. Alphanumeric characters as well as underscores are allowed, for example, kafka_1. In this case, Kafka checks if a resource-based service in Ranger matching the defined string exists. If it does, Kafka will use that resource-based service. If it does not, Kafka automatically creates and uses a new resource-based service. The name of the newly created resource-based service will be the name that was defined in the property.

Set {{GENERATED_RANGER_SERVICE_NAME}}

The property accepts {{GENERATED_RANGER_SERVICE_NAME}} as a valid value. When this specific placeholder value is set, Cloudera Manager generates a name for a new resource-based service. The generated name is based on the Kafka service name and the cluster name. Kafka then checks whether a resource-based service with the generated name exists in Ranger. If it does, Kafka will use that resource-based service. If it does not, Kafka automatically creates and uses a new resource-based service. The name of the newly created resource-based service will be the generated name.

In both scenarios, the creation of the role-based service happens once the Kafka service is restarted.

The property by default is set to cm_kafka. However, for clusters provisioned with the Streams Messaging cluster definitions in Cloudera Data Hub the default is {{GENERATED_RANGER_SERVICE_NAME}}.

Configuring this property can prove useful when you have multiple Kafka clusters that use the same Ranger service for authorization, but you want to define unique Ranger policies for each Kafka cluster.

Procedure

1. In Cloudera Manager select the Kafka service.
2. Select Configuration and find the Ranger service name for this Kafka cluster property.
3. Configure the property.

   You have two choices:

   - Set the property to a user defined string.
   - Set the property to {{GENERATED_RANGER_SERVICE_NAME}}.
4. Click Save Changes.
5. Restart the Kafka service.

Results

The selected resource-based service is configured and is used by Kafka for authorization. If the resource-based service does not exist, it will be automatically created in Ranger.

Related Information

Apache Ranger Authorization

Kafka ACL APIs support in Ranger

Kafka ACL operations in Ranger

Background

Kafka supports various ACL operations, such as APIs like `createAcls`, `deleteAcls`, `acls`, and `authorizeByResourceType`. However, when Ranger is enabled, Kafka does not support:

- grant and revoke of policies on Kafka resources
- list the resources ACLs based on Ranger policies
- check the ACLs on resources

RangerKafkaAuthorizer includes ACL APIs that refer to Ranger Policies when these commands are executed. Ranger relies on the grant, revoke and policy engine APIs to cater the needed functionality.

Note: The following functionality is NOT supported in 7.1.9:

--allow-host and --deny-host

Host name in the API call. This is not supported as this requires grant/revoke ranger API to support the creation of policy conditions for the policy that is getting created.

--deny-principal

Grant except and Revoke except are not supported.

–-operations

Multiple operations on Single command is not supported.

Permission ALL is not supported in the command line.

Reference Commands

List acl

```
kafka-acls --bootstrap-server rm-718ssl-1.rm-718ssl.root.hwx.sit
e:9093 --command-config /kafkatest/kafka-client.conf --list --to
pic connect-configs
```

```
kafka-acls --bootstrap-server rm-718ssl-1.rm-718ssl.root.hwx.sit
e:9093 --command-config /kafkatest/kafka-client.conf --list --cl
uster test_cluster1
```

```
afka-acls --bootstrap-server rm-718ssl-1.rm-718ssl.root.hwx.site
:9093 --command-config /kafkatest/kafka-client.conf --list --clu
ster
```

Create acl

```
kafka-acls --bootstrap-server rm-718ssl-1.rm-718ssl.root.hwx.sit
e:9093 --command-config /kafkatest/kafka-client.conf --add --all
ow-principal User:testuser1 --operation read --topic finance-top
ic
```

```
kafka-acls --bootstrap-server rm-718ssl-1.rm-718ssl.root.hwx.sit
e:9093 --command-config /kafkatest/kafka-client.conf --add --all
ow-principal Group:mysql --operation read --topic finance-topic
```

```
kafka-acls --bootstrap-server rm-718ssl-1.rm-718ssl.root.hwx.sit
e:9093 --command-config /kafkatest/kafka-client.conf --add --all
ow-principal Group:mysql --allow-principal User:testuser1 --oper
ation read --topic finance-topic
```

```
kafka-acls --bootstrap-server rm-718ssl-1.rm-718ssl.root.hwx.sit
e:9093 --command-config /kafkatest/kafka-client.conf --add --all
ow-principal User:testuser1 -operation read --topic finance-topic
 --resource-pattern-type  prefixed
```

Revoke acl

```
kafka-acls --bootstrap-server rm-718ssl-1.rm-718ssl.root.hwx.sit
e:9093 --command-config  /kafkatest/kafka-client.conf --remove --
allow-principal Group:mysql -operation read --topic finance-topic
```

```
kafka-acls --bootstrap-server rm-718ssl-1.rm-718ssl.root.hwx.sit
e:9093 --command-config  /kafkatest/kafka-client.conf --remove -
-allow-principal User:testuser1 -operation read --topic finance-
topic
```

Related Information

Authorization using ACLs: Confluent Developer documentation

Apache Kafka documentation

Governance

You can monitor and govern Kafka metadata in Atlas. This can be done by importing Kafka topics and clients as entities (metadata) into Atlas.

Kafka entities can be imported into Atlas by running a one-time import using the Import Kafka Topics Into Atlas action available in Cloudera Manager or by configuring the Atlas hook in Kafka. The Import Kafka Topics Into Atlas action imports existing entities only. Setting up the Atlas hook enables continuous import of newly created Kafka entities, but does not import existing ones.

Importing Kafka entities into Atlas

Existing Kafka topics and clients can be imported into Atlas as entities using the Import Kafka Topics Into Atlas action available in Cloudera Manager. Importing Kafka entities allows you to manage, govern, and monitor your topics and clients in Atlas.

About this task

The following list of steps walk you through how you can import Kafka topics and clients as entities (metadata) into Atlas using the Import Kafka Topics Into Atlas action in Cloudera Manager. This action only imports existing topics and clients. Topics and clients created after the import is completed are not automatically imported into Atlas.

Continuous import can be enabled by configuring the Atlas hook in Kafka. For more information, see *Configuring the Atlas hook in Kafka*.

Before you begin

- An Atlas service is deployed on the Kafka cluster or the data context cluster. The Atlas service must be started and must be in a healthy state.
- You are logged in to Cloudera Manager as a user that can perform service and role level configuration changes (Cluster Administrator, Full Administrator, Configurator, Limited Cluster Administrator).
- If authorization is enabled, ensure that the kafka user is authorized to create, update, delete, and read Kafka entities in Atlas.

Procedure

1. In Cloudera Manager select the Kafka service.
2. Click ActionsImport Kafka Topics Into Atlas.
3. Click Import Kafka Topics Into Atlas and wait until the process is finished.

Results

Existing Kafka topics and clients imported into the Atlas service.

What to do next

Access the Atlas dashboard to view, monitor, and manage imported Kafka entities.

Related Information

Configuring the Atlas hook in Kafka

Configuring the Atlas hook in Kafka

An Atlas hook collects metadata from Kafka and transfers it to Atlas so that you can manage, govern, and monitor Kafka metadata and metadata lineage in the Atlas UI. You can enable and configure an Atlas hook in Kafka.

About this task

The following instructions walk you through how you enable the Atlas hook, and how you configure the topic and client metadata namespaces used by the hook. The namespaces you configure for the hook in Kafka, are used in Atlas, to group and represent Kafka entities. Once the hook is set up and configured, Kafka metadata will be available in Atlas.

Enabling and configuring the hook only imports and exposes newly created Kafka topics. Existing topics are not imported automatically. To access existing topics, you must use the kafka-import.sh tool and import them manually into Atlas.

Before you begin

- Ensure that an Atlas service is deployed on the Kafka cluster or the data context cluster.
- Import existing Kafka topics into Atlas using the Import Kafka Topics Into Atlas action in Cloudera Manager. For more information, see Importing Kafka entities into Atlas.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find and enable the Enable Auditing to Atlas property.

   This property enables or disables the Atlas hook in Kafka.
4. Specify the topic and client metadata namespaces.

   The topic and client metadata namespaces are configured with the Atlas metadata namespace for Kafka Topics and Atlas metadata namespace for Kafka Clients properties. How you configure these properties depends on your environment and use case. Cloudera recommends that you to follow these guidelines when configuring namespaces:

   - Use the same topic namespace you used with the kafka-topics.sh tool if you manually imported topics.
   - Use identical client and topics namespaces if only a single Kafka cluster is audited by Atlas.

     In this case, you can also consider using the default namespace, which is cm.
   - Use unique topic namespaces in environments where there are multiple Kafka clusters audited by Atlas.

     This is a recommended practice to avoid the conflict of topic entities.
   - Configure client namespaces based on your use case.

     Unlike topic namespaces, client namespaces do not have to be unique even if there are multiple Kafka clusters in your environment. For example, if there is an application communicating with multiple Kafka clusters, and it is using the same client.id, the client metadata namespace can be set to the same value for all Kafka clusters. This way, the application is represented as a single producer or consumer entity in Atlas.
5. Click Save Changes.

6. Restart the Kafka service.

Results

The Atlas hook in Kafka is enabled and configured. Kafka topic, consumer, producer, and consumer group metadata can be monitored using the Atlas UI.

What to do next

Learn more about the Atlas hook as well as Kafka metadata collection. For more information see *Kafka metadata collection*.

Related Information

Setting up Atlas Kafka import tool

Kafka metadata collection

Inter-broker security

Kafka can expose multiple communication endpoints, each supporting a different protocol. Supporting multiple communication endpoints enables you to use different communication protocols for client-to-broker communications and inter-broker communications.

The security protocol used for inter-broker communication is controlled by the Inter Broker Protocol Cloudera Manager property. By default this property is set to INFERRED, which sets the security protocol based on how other security properties of the broker are configured.

The INFERRED setting configures the protocol according to the following logic:

Kerberos or LDAP enabled	TLS/SSL enabled	Protocol
YES	YES	SASL_SSL
YES	NO	SASL_PLAINTEXT
NO	YES	SSL
NO	NO	PLAINTEXT

Cloudera recommends that you use the protocol set by INFERRED. However, you can change this setting and use a specific protocol. This can be done by setting the Inter Broker Protocol property to the protocol that you want to use for inter-broker communication.

Changing the inter-broker protocol from the default is primarily useful for the following reasons:

Improving performance

Enabling TLS/SSL is known to have performance overhead. If your Kafka brokers are behind a firewall and are not susceptible to network snooping, you can consider enabling TLS/SSL for client-to-broker communication, but keep inter-broker communication set as PLAINTEXT. A configuration like this can result in a better performing Kafka cluster.

Securing a non-secure Kafka deployment without downtime

It is possible to migrate from a non-secure Kafka configuration to a secure Kafka configuration without downtime. This can be achieved with a rolling restart and by setting the inter-broker protocol to a protocol that is supported by all brokers until all brokers are updated to support the new protocol. For example, if you have a Kafka cluster that needs to be configured to enable Kerberos without downtime, you would take the following steps:

1. Set inter-broker protocol to PLAINTEXT.
2. Update the Kafka service configuration to enable Kerberos.
3. Perform a rolling restart.
4. Set inter-broker protocol to SASL_PLAINTEXT.

Configuring multiple listeners

Kafka brokers can simultaneously listen to connection requests on multiple ports with various protocols. By default Cloudera Manager automatically configures the ports and the protocols used by the brokers to listen to requests. However, manual configuration is possible and may be required in an advanced deployment.

About this task

Kafka Brokers support listening for connections on multiple ports with different protocols. For example, a broker is capable of listening on port 9092 for PLAINTEXT requests and 9093 using TLS/SSL simultaneously. Which ports a broker listens to is controlled by the listeners Kafka broker property.

In Cloudera Manager, the listeners property is not available directly for configuration. Instead, it is set by Cloudera Manager automatically based on how other security properties are configured.

For example, if TLS/SSL is enabled, Enable TLS/SSL for Kafka Broker is set to true, Cloudera Manager automatically configures the Kafka broker to listen to TLS/SSL requests on port 9093. That is, the listeners property is set to SSL://*[***KAFKA BROKER FQDN***]*:9093.

It is usually sufficient to rely on Cloudera Manager to automatically configure listeners for you. However, in a deployment where the clients use various protocols and ports to establish a connection with the broker, you need to configure the broker so that it listens on all ports and with all protocols used by the clients. This configuration has to be done manually with the help of an advanced configuration snippet.

Complete the following steps to manually configure listeners for Kafka brokers.

Before you begin

During configuration, ensure that listeners are set individually for each broker, using each broker's FQDN.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Instances.
3. Configure listeners using an advanced configuration snippet:

   Repeat the following steps for each Kafka broker role.

   a) Click on a Kafka broker role.
   b) Go to Configuration.
   c) Find the Kafka Broker Advanced Configuration Snippet (Safety Valve) for kafka.properties property and configure listeners.

      For example:

      ```
      listeners=SASL_SSL://[***KAFKA BROKER FQDN***]:9093,SSL://[***KAFKA
       BROKER FQDN***]:9094
      ```

      This example shows a configuration where the broker is accepting both SASL_SSL and SSL requests.

      Important: Ensure that you configure a listener for all protocols and ports that you want the broker to listen to.

   d) Click Save Changes.
4. Restart the Kafka service.

Results

The Kafka broker now listens for connection requests on the configured ports with the configured protocols.

Kafka security hardening with Zookeeper ACLs

Access to Kafka metadata in Zookeeper is restricted by default. However, when completing configuration and management tasks for your Kafka service, you might be required to first unlock and then restrict access to metadata in Zookeeper. Continue reading to learn how you can restrict or unlock access to Kafka metadata in Zookeeper.

Restricting access to Kafka metadata in Zookeeper

Learn how to restrict access to Kafka metadata in Zookeeper.

About this task

Locking down Kafka znodes (metadata) in Zookeeper can be used to protect Kafka metadata against unauthorized access. Direct manipulation of metadata in Zookeeper is not only dangerous for the health of the cluster, but can also serve as an entry point for malicious users to gain elevated. Access to Kafka metadata in Zookeeper is restricted by default. Follow these steps if you have previously unlocked access, but want to re-enable access restrictions.

Before you begin

A secure Kafka cluster with Kerberos authentication enabled is required.

Procedure

1. Enable the use of secure ACLs:

   This can be achieved by setting the zookeeper.set.acl configuration parameter to true.

   a) In Cloudera Manager select the Kafka service.
   b) Select Configuration.
   c) Find the Enable Zookeeper ACL property.
   d) Set the property to true by selecting the checkbox next to the name of the role group.
2. Perform a Rolling Restart:

   a) Return to the Home page by clicking the Cloudera Manager logo.
   b) Check the Restart roles with stale configurations only checkbox and click Rolling restart.
   c) Click Close when the restart has finished.
3. Pass the JAAS configuration file location as a JVM parameter through a command line interface.

   You can do this by setting the value of the KAFKA_OPTS environment variable to -Djava.security.auth.login.c onfig=*[PATH_TO_JAAS.CONF]*.

   ```
   export KAFKA_OPTS="-Djava.security.auth.login.config=[PATH_TO_JAAS.CONF]"
   ```

   Replace *[PATH_TO_JAAS.CONF]* with the path to your JAAS configuration file.
4. Run the zookeeper-security-migration tool with the zookeeper.acl option set to secure.

   ```
   zookeeper-security-migration --zookeeper.connect [HOSTNAME]:[PORT]/[ZOOK
   EEPER_KAFKA_CHROOT] --zookeeper.acl secure
   ```

   Replace *[ZOOKEEPER_KAFKA_CHROOT]* with the value of the zookeeper.chroot property. To view the value of the property, in Cloudera Manager go to Kafka Configuration and search for zookeeper.chroot.

   The tool traverses the corresponding sub-trees changing the ACLs of the znodes.

5. Reset the ACLs on the root node to allow full access:

   Resetting the ACLS on the root node is required because the zookeeper-security-migration tool also changes the ACLs on the root znode. This leads to the failure of the Zookeeper canary tests, which subsequently makes the service display as unhealthy in Cloudera Manager.

   Important: This step is only necessary if the zookeeper.chroot parameter of the broker is set to /.

   Note: Because the Kafka metadata at this point is already restricted, only authorized users or Zookeeper super users can complete this step.

   a) Change the JVMFLAGS environment variable to -Djava.security.auth.login.confi g=*[PATH_TO_JAAS.CONF]*.

   ```
   export JVMFLAGS="-Djava.security.auth.login.config=[PATH_TO_JAAS.CONF]"
   ```

   b) Start the zookeeper client.

   ```
   zookeeper-client -server $(hostname -f):2181
   ```

   c) Enter the following to reset the ACLs of the root node.

   ```
   setAcl / world:anyone:crdwa
   ```

6. Verify znode permissions.

   You can do this with the getAcl ZooKeeper CLI command. For example.

   ```
   getAcl [ZK NODE PATH]
   ```

   If everything is set up correctly, running this command on the /kafka znode (default chroot for Kafka) should return the following permissions.

   ```
   'sasl,'kafka
   : cdrwa
   'world,'anyone
   : r
   ```

Results

Kafka metadata in Zookeeper is restricted via ACLS. Administrative operations, for example topic creation, deletion, any configuration changes and so on, can only be performed by authorized users.

Related Information

Enable Kerberos authentication for Kafka

Unlocking access to Kafka metadata in Zookeeper

Learn how to unlock access to Kafka metadata in Zookeeper

About this task

Cloudera does not recommend that you leave access to Kafka znodes (metadata) in Zookeeper unlocked. Only unlock access if you are planning to make configuration changes in Kafka that require znode access to be unlocked. Once configuration changes are complete, Cloudera recommends that you re-restrict access.

Before you begin

A secure Kafka cluster with Kerberos authentication enabled is required.

Procedure

1. Disable the use of secure ACLs:

   This can be achieved by setting the zookeeper.set.acl configuration parameter to false.

   a) In Cloudera Manager select the Kafka service.
   b) Select Configuration.
   c) Find the Enable Zookeeper ACL property.
   d) Set the property to false by unchecking the checkbox next to the name of the role group.
2. Perform a Rolling Restart:

   a) Return to the Home page by clicking the Cloudera Manager logo.
   b) Go to the Kafka service and select Actions Rolling Restart.
   c) Check the Restart roles with stale configurations only checkbox and click Rolling restart.
   d) Click Close when the restart has finished.
3. Run the zookeeper-security-migration tool with the zookeeper.acl option set to unsecure

   ```
   zookeeper-security-migration --zookeeper.connect [HOSTNAME]:[PORT]/[ZOOK
   EEPER_KAFKA_CHROOT] --zookeeper.acl unsecure
   ```

   Replace *[ZOOKEEPER_KAFKA_CHROOT]* with the value of the zookeeper.chroot property. To view the value of the property, in Cloudera Manager go to Kafka Configuration and search for zookeeper.chroot.

Results

The tool traverses the corresponding sub-trees changing the ACLs of the znodes. Access to Kafka metadata stored in Zookeeper becomes unrestricted.

Related Information

Enable Kerberos authentication for Kafka

Cloudera Runtime 7.3.1

Apache Knox Authentication

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Knox overview....4
Securing access to Hadoop cluster: Apache Knox....4
Apache Knox Gateway overview....4
Knox Supported Services Matrix....5
Considerations for Knox....6

Proxy Cloudera Manager through Apache Knox....7

Installing Apache Knox....7
Apache Knox Install Role Parameters....9

Management of Knox shared providers in Cloudera Manager....11
Management of existing Apache Knox shared providers....11
Add a new provider in an existing provider configuration....12
Modify a provider in an existing provider configuration....13
Disable a provider in an existing provider configuration....14
Remove a provider parameter in an existing provider configuration....16
Saving aliases....17
Configuring Kerberos authentication in Apache Knox shared providers....19
Configuring group mapping in Knox....21

Management of services for Apache Knox through Cloudera Manager....23
Enable proxy for a known service in Apache Knox....23
Disable proxy for a known service in Apache Knox....25
Add a custom descriptor to Apache Knox....26

Load balancing for Apache Knox....28
Generate and configure a signing keystore for Knox in HA....29

Knox Gateway token integration....29
Overview....29
Token configurations....31
Generate tokens....36
Manage Knox Gateway tokens....39
Knox Token API....41
Manage Knox metadata....45

Knox SSO Cookie Invalidation....46

Concurrent session verification (Tech Preview)....48

Apache Knox overview

Securing access to Hadoop cluster: Apache Knox

The Apache Knox Gateway ("Knox") is a system to extend the reach of Apache™ Hadoop® services to users outside of a Hadoop cluster without reducing Hadoop Security. Knox also simplifies Hadoop security for users who access the cluster data and execute jobs. The Knox Gateway is designed as a reverse proxy.

Establishing user identity with strong authentication is the basis for secure access in Hadoop. Users need to reliably identify themselves and then have that identity propagated throughout the Hadoop cluster to access cluster resources.

Layers of defense for a Cloudera cluster

- Authentication: Kerberos

  Cloudera uses Kerberos for authentication. Kerberos is an industry standard used to authenticate users and resources within a Hadoop cluster. Cloudera also includes Cloudera Manager, which simplifies Kerberos setup, configuration, and maintenance.
- Perimeter Level Security: Apache Knox

  Apache Knox Gateway is used to help ensure perimeter security for Cloudera customers. With Knox, enterprises can confidently extend the Hadoop REST API to new users without Kerberos complexities, while also maintaining compliance with enterprise security policies. Knox provides a central gateway for Hadoop REST APIs that have varying degrees of authorization, authentication, SSL, and SSO capabilities to enable a single access point for Hadoop.

  Cloudera recommends that you leverage the default PAM Authentication Provider for the benefits in performance and ease of administration rather than direct LDAP. For more details, see *Considerations for Knox*.
- Authorization: Ranger

  OS Security: Data Encryption and HDFS

Related Information

Considerations for Knox

Apache Knox Gateway overview

A conceptual overview of the Apache Knox Gateway, a reverse proxy.

Overview

Knox integrates with Identity Management and SSO systems used in enterprises and allows identity from these systems be used for access to Hadoop clusters.

Knox Gateway provides security for multiple Hadoop clusters, with these advantages:

- Simplifies access: Extends Hadoop's REST/HTTP services by encapsulating Kerberos to within the Cluster.
- Enhances security: Exposes Hadoop's REST/HTTP services without revealing network details, providing SSL out of the box.
- Centralized control: Enforces REST API security centrally, routing requests to multiple Hadoop clusters.
- Enterprise integration: Supports LDAP, Active Directory, SSO, SAML and other authentication systems.

Typical security flow: Firewall, routed through Knox Gateway

Knox can be used with both unsecured Hadoop clusters, and Kerberos secured clusters. In an enterprise solution that employs Kerberos secured clusters, the Apache Knox Gateway provides an enterprise security solution that:

- Integrates well with enterprise identity management solutions
- Protects the details of the Hadoop cluster deployment (hosts and ports are hidden from end users)
- Simplifies the number of services with which a client needs to interact

Knox Gateway deployment architecture

Users who access Hadoop externally do so either through Knox, via the Apache REST API, or through the Hadoop CLI tools.

Knox Supported Services Matrix

A support matrix showing which services Apache Knox supports for Proxy and SSO, for both Kerberized and Non-Kerberized clusters.

Table 1: Knox Supported Components

Component	UI Proxy (with SSO)	API Proxy
Atlas API	#	#
Atlas UI	#	#
Beacon		
Cloudera Manager API	#	#
Cloudera Manager UI	#	
Data Analytics Studio (DAS)	#	
Druid		
Falcon		
Flink		
HBase REST API(aka WebHBase & Stargate)		#
HBase UI	#	
HDFS UI	#	
HiveServer2 HTTP JDBC API (HS2 via HTTP)		#
HiveServer2 LLAP JDBC API		
HiveServer2 LLAP UI		
HiveServer2 UI		
Hue	#	
Impala HTTP JDBC API		#
Impala UI	#	
JobHistory UI	#	
JobTracker		#
Kudu UI	#	
Livy API + UI	#	#
LogSearch		

Component	UI Proxy (with SSO)	API Proxy
NameNode	#	#
NiFi	#	#
NiFi Registry	#	#
Oozie API	#	#
Oozie UI	#	
Ozone	#	
Phoenix (aka Avatica)		#
Profiler	#	
Ranger API	#	#
Ranger UI	#	
Yarn ResourceManager API	#	#
Schema Registry API + UI	#	#
Streams Messaging Manager API	#	#
Streams Messaging Manager UI	#	
Solr	#	#
Spark3History UI	#	
SparkHistory UI	#	
Storm		
Storm LogViewer		
Superset		
WebHCat		
WebHDFS		#
YARN UI	#	
YARN UI V2	#	
Zeppelin UI	#	
Zeppelin WS	#	

Note:

APIs, UIs, and SSO in the Apache Knox project that are not listed above are considered Community Features.

Community Features are developed and tested by the Apache Knox community but are not officially supported by Cloudera. These features are excluded for a variety of reasons, including insufficient reliability or incomplete test case coverage, declaration of non-production readiness by the community at large, and feature deviation from Cloudera best practices. Do not use these features in your production environments.

Considerations for Knox

Learn about the considerations before you get started with Knox.

Default PAM settings for Knox

Secure clusters require local OS accounts. Local OS accounts are most often achieved by using something like SSSD or Centrify which localizes user accounts from user stores or directories including LDAP/AD and so on.

This means that you should be able to use PAM and the local OS accounts straight away as long as they are on the same host as Knox. There are various ways to make local OS accounts available. SSSD or Centrify are technical solutions that make it seem like there are local OS accounts even though they are only in LDAP/AD. You can use real local OS accounts as well.

Proxy Cloudera Manager through Apache Knox

In order to have Cloudera Manager proxied through Knox, there are some steps you must complete.

Procedure

1. Set the value for frontend_url: Cloudera Manager Administration Settings Cloudera Manager Frontend URL :
   - Non-HA value: https://$Knox_host:$knox_port
   - HA value: https://$Knox_loadbalancer_host:$Knox_loadbalancer_port
2. Set allowed groups, hosts, and users for Knox Proxy: Cloudera Manager Administration Settings External Authentication :
   - Allowed Groups for Knox Proxy: *
   - Allowed Hosts for Knox Proxy: *
   - Allowed Users for Knox Proxy: *
3. Enable Kerberos/SPNEGO authentication for the Admin Console and API: Cloudera Manager Administration Settings External Authentication Enable SPNEGO/Kerberos Authentication for the Admin Console and API: : true
4. From Cloudera Manager Administration Settings External Authentication , set Knox Proxy Principal: knox.

What to do next

External authentication must be set up correctly. Cloudera Manager must be configured to use LDAP, following the standard procedure for setting up LDAP. This LDAP server should be the same LDAP that populates local users on Knox hosts (if using PAM authentication with Knox), or the same LDAP that Knox is configured to use (if using LDAP authentication with Knox).

However in cases where no LDAP server is available, corresponding Cloudera Manager local users can be created and assigned roles manually in Cloudera Manager Administrator Users & Roles Add Local User .

Installing Apache Knox

This document provides instructions on how to install Apache Knox using the Cloudera Private Cloud Base installation process.

About this task

Apache Knox is an application gateway for interacting with the REST APIs and UIs. The Knox Gateway provides a single access point for all REST and HTTP interactions in your Cloudera cluster.

Before you begin

When installing Knox, you must have Kerberos enabled on your cluster.

Procedure

1. From your Cloudera Manager homepage, go to Status tab $Cluster Name ... Add Service

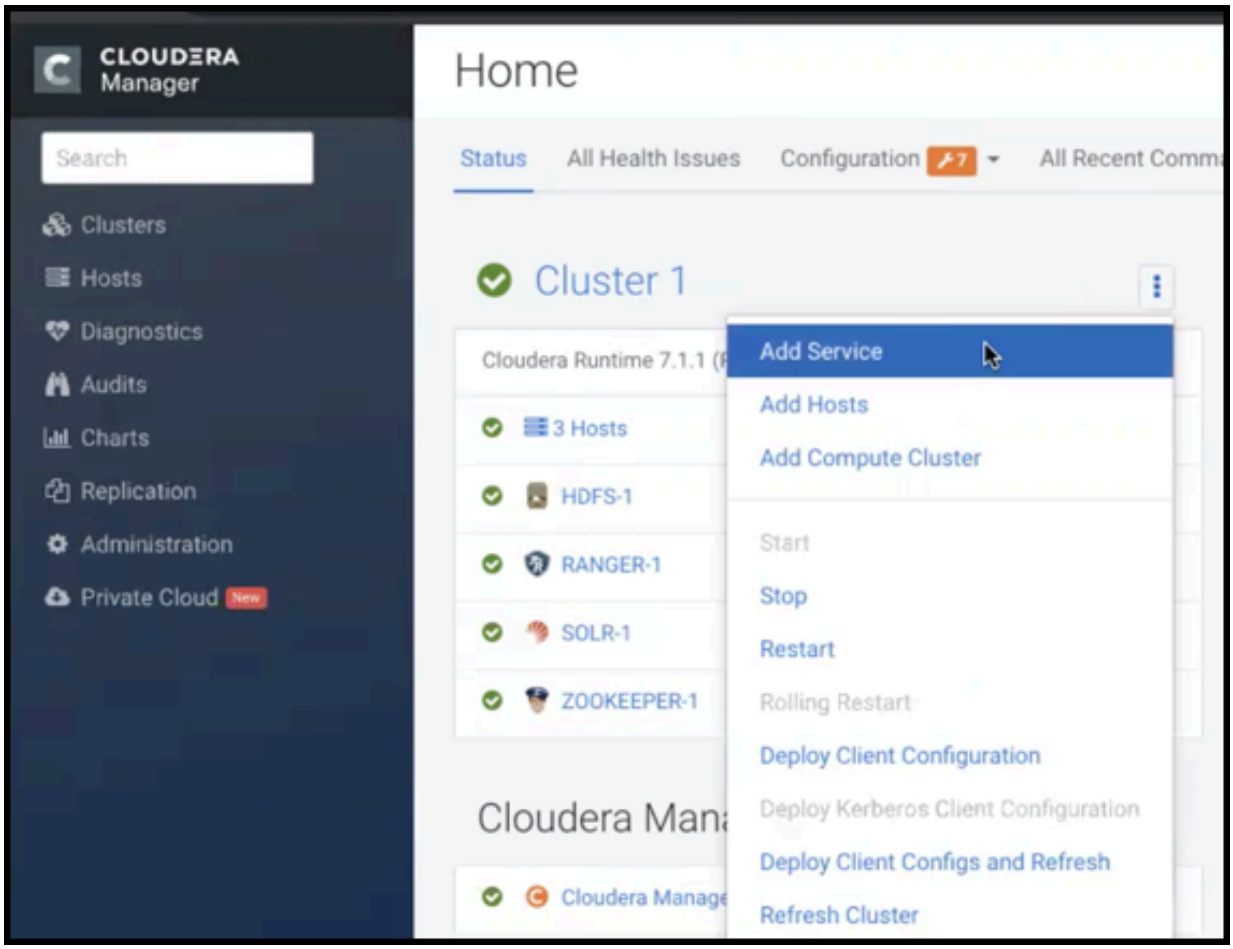

2. From the list of services, select Knox and click Continue.
3. On the **Select Dependencies** page, choose the dependencies you want Knox to set up:

HDFS, Ranger, Solr, Zookeeper	For users that require Apache Ranger for authorization. HDFS with Ranger. HDFS depends on Zookeeper, and Ranger depends on Solr.
HDFS, Zookeeper	HDFS depends on Zookeeper.
No optional dependencies	For users that do not wish to have Knox integrate with HDFS or Ranger.

4. On the **Assign Roles** page, select role assignments for your dependencies and click Continue:

Knox service roles	Description	Required?
Knox Gateway	If Knox is installed, at least one instance of this role should be installed. This role represents the Knox Gateway which provides a single access point for all REST and HTTP interactions with Apache Hadoop clusters.	Required

Knox service roles	Description	Required?
KnoxIDBroker*	It is strongly recommended that this role is installed on its own dedicated host. As its name suggests this role will allow you to take advantage of Knox's Identity Broker capabilities, an identity federation solution that exchanges cluster authentication for temporary cloud credentials.*	Optional*
Gateway	This role comes with the CSD framework. The gateway structure is used to describe the client configuration of the service on each host where the gateway role is installed.	Optional

* Note: KnoxIDBroker appears in the Assign Roles page, but it is not currently supported in Cloudera Private Cloud.

5. On the **Review Changes** page, most of the default values are acceptable, but you must Enable Kerberos Authentication and supply the Knox Master Secret. There are additional parameters you can specify or change, listed in "Knox Install Role Parameters".
   a) Click Enable Kerberos Authentication

      Kerberos is required where Knox is enabled.
   b) Supply the Knox Master Secret, e.g. knoxsecret.
   c) Click Continue.
6. The **Command Details** page shows the status of your operation. After completion, your system admin can view logs for your installation under stdout.

Apache Knox Install Role Parameters

Reference information on all the parameters available for Knox service roles.

Service-level parameters

Table 2: Required service-level parameters

Name	In Wizard	Type	Default Value
kerberos.auth.enabled*	Yes	Boolean	false
ranger_knox_plugin_hdfs_audit_directory	No	Text	${ranger_base_audit_url}/knox
autorestart_on_stop	No	Boolean	false
knox_pam_realm_service	No	Text	login
save_alias_command_input_password	No	Text	-

Knox Gateway role parameters

Table 3: Required parameters for Knox Gateway role

Name	In Wizard	Type	Default Value
gateway_master_secret	Yes	Password	-
gateway_conf_dir	Yes	Path	/var/lib/knox/gateway/conf
gateway_data_dir	Yes	Path	/var/lib/knox/gateway/data
gateway_port	No	Port	8443
gateway_path	No	Text	gateway

Name	In Wizard	Type	Default Value
gateway_heap_size	No	Memory	1 GB (min = 256 MB; soft min = 512 MB)
gateway_ranger_knox_plugin_conf_path	No	Path	/var/lib/knox/ranger-knox-plugin
gateway_ranger_knox_plugin_policy_cache_directory	No	Path	/var/lib/ranger/knox/gateway/policy-cache
gateway_ranger_knox_plugin_hdfs_audit_spool_directory	No	Path	/var/log/knox/gateway/audit/hdfs/spool
gateway_ranger_knox_plugin_solr_audit_spool_directory	No	Path	/var/log/knox/gateway/audit/solr/spool

Table 4: Optional parameters for Knox Gateway role

Name	Type	Default Value
gateway_default_topology_name	Text	cdp-proxy
gateway_auto_discovery_enabled	Boolean	true
gateway_cluster_configuration_monitor_interval	Time	60 seconds (minimum = 30 seconds)
gateway_auto_discovery_advanced_configuration_monitor_interval	Time	10 seconds (minimum = 5 seconds)
gateway_cloudera_manager_descriptors_monitor_interval	Time	10 seconds (minimum = 5 seconds)
gateway_auto_discovery_cdp_proxy_enabled_*	Boolean	true
gateway_auto_discovery_cdp_proxy_api_enabled_*	Boolean	true
gateway_descriptor_cdp_proxy	Text Array	Contains the required properties of cdp-proxy topology
gateway_descriptor_cdp_proxy_api	Text Array	Contains the required properties of cdp-proxy-api topology
gateway_sso_authentication_provider	Text Array	Contains the required properties of the authentication provider used by the UIs using the Knox SSO capabilities (such as Home Page UI). Defaults to PAM authentication.
gateway_api_authentication_provider	Text Array	Contains the required properties of the authentication provider used by pre-defined topologies such as admin, metadata or cdp-proxy-api. Defaults to PAM authentication.

Knox IDBroker role parameters

Note: Knox IDBroker is not currently supported in Cloudera Private Cloud.

Table 5: Required parameters for Knox IDBroker role

Name	In Wizard	Type	Default Value
idbroker_master_secret	Yes	Password	-
idbroker_conf_dir	Yes	Path	/var/lib/knox/idbroker/conf
idbroker_data_dir	Yes	Path	/var/lib/knox/idbroker/data
idbroker_gateway_port	No	Port	8444
idbroker_gateway_path	No	Text	gateway

Name	In Wizard	Type	Default Value
idbroker_heap_size	No	Memory	1 GB (min = 256 MB; soft min = 512 MB)

Table 6: Optional parameters for Knox IDBroker role

Name	Type	Default Value
idbroker_aws_user_mapping	Text	-
idbroker_aws_group_mapping	Text	-
idbroker_aws_user_default_group_mapping	Text	-
idbroker_aws_credentials_key	Password	-
idbroker_aws_credentials_secret	Password	-
idbroker_gcp_user_mapping	Text	-
idbroker_gcp_group_mapping	Text	-
idbroker_gcp_user_default_group_mapping	Text	-
idbroker_gcp_credential_key	Password	-
idbroker_gcp_credential_secret	Password	-
idbroker_azure_user_mapping	Text	-
idbroker_azure_group_mapping	Text	-
idbroker_azure_user_default_group_mapping	Text	-
idbroker_azure_adls2_tenant_name	Text	-
idbroker_azure_vm_assumer_identity	Text	-
idbroker_relaodable_refresh_interval_ms	Time	10 seconds (minimum = 1 second)
idbroker_kerberos_dt_proxyuser_block	Text Array	A comma-separated list of proxy user configuration used in Knox's dt topology in case Kerberos is enabled
idbroker_knox_token_ttl_ms	Time	1 hour (minimum = 1 second)

Management of Knox shared providers in Cloudera Manager

Information on Cloudera Private Cloud topology management for Knox from within Cloudera Manager.

- Modifying the SSO authentication provider used by the UIs using the Knox SSO capabilities, such as the Home Page UI.
- Modifying the API authentication provider used by predefined topologies, such as admin, metadata or cdp-proxy-api.
- Adding/modifying new/existing shared provider configurations.
- Saving aliases using a new Knox Gateway command.

Management of existing Apache Knox shared providers

You can add, modify, or disable an existing shared provider configuration in Apache Knox via Cloudera Manager.

Add a new provider in an existing provider configuration

An example of how to add a new provider to the authorization provider in the manager shared provider configuration.

About this task

In this example you will see how to add a new HA provider (this time only the ATLAS service will be configured for high availability) in the manager shared provider configuration . This particular authorization provider is set as follows (in its JSON descriptor):

```
{
        "role": "authorization",
        "name": "AclsAuthz",
        "enabled": "true",
        "params": {
           "knox.acl.mode": "OR",
           "knox.acl": "KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;*"
        }
     }
```

Procedure

1. From Cloudera Manager Knox Configuration , add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml:
   - name = providerConfigs:manager
   - value = role=authorization#authorization.name=AclsAuthz#authorization.enabled=false#authorization.param.knox.acl=myTestUser;KNOX_ADMIN_GROUPS;*#authorization.param.knox.acl.mode=OR#role=ha#ha.name=HaProvider#ha.param.ATLAS=enabled=true;maxFailoverAttempts=3;failoverSleep=1000;maxRetr

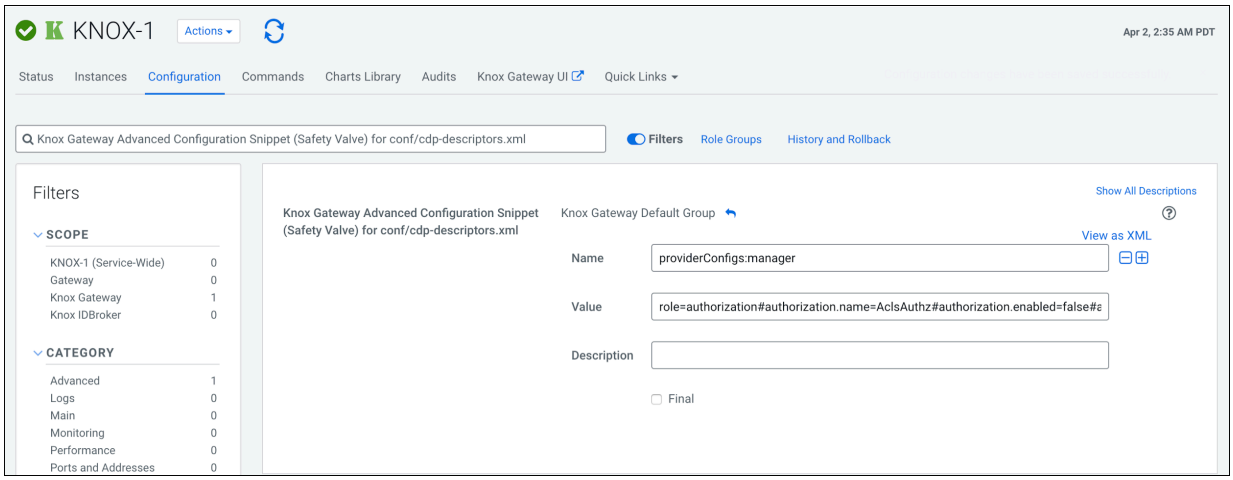

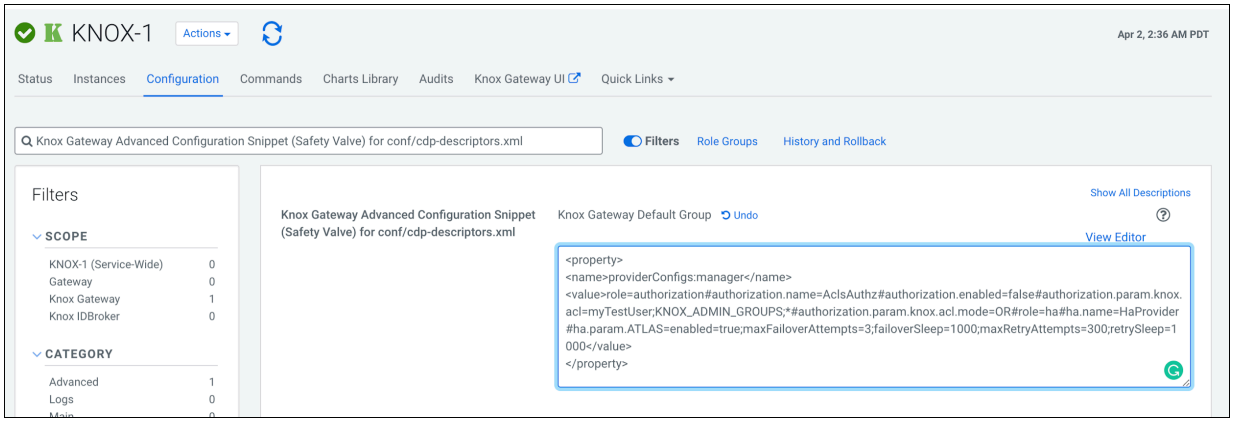

2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/
gateway/admin/api/v1/providerconfig/manager'
{
  "providers" : [
 ...
  }, {
    "role" : "authorization",
    "name" : "AclsAuthz",
    "enabled" : false,
    "params" : {
      "knox.acl" : "myTestUser;KNOX_ADMIN_GROUPS;*",
      "knox.acl.mode" : "OR"
    }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
    "enabled" : true,
    "params" : {
      "ATLAS" : "enabled=true;maxFailoverAttempts=3;failoverSleep=1000;m
axRetryAttempts=300;retrySleep=1000"
    }
  } ]
}
```

Modify a provider in an existing provider configuration

An example of how to modify the authorization provider in the manager shared provider configuration.

About this task

In this example you will see how to modify the authorization provider in the manager shared provider configuration. This particular authorization provider is set as follows (in its JSON descriptor):

```
{
         "role": "authorization",
         "name": "AclsAuthz",
         "enabled": "true",
         "params": {
            "knox.acl.mode": "OR",
            "knox.acl": "KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;*"
         }
      }
```

Procedure

1. From Cloudera Manager Knox Configuration , add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml:
   - name = providerConfigs:manager
   - value = role=authorization#authorization.name=AclsAuthz#authorization.enabled=false#authorization.param.knox.acl=myTestUser;KNOX_ADMIN_GROUPS;*#authorization.param.knox.acl.mode=OR

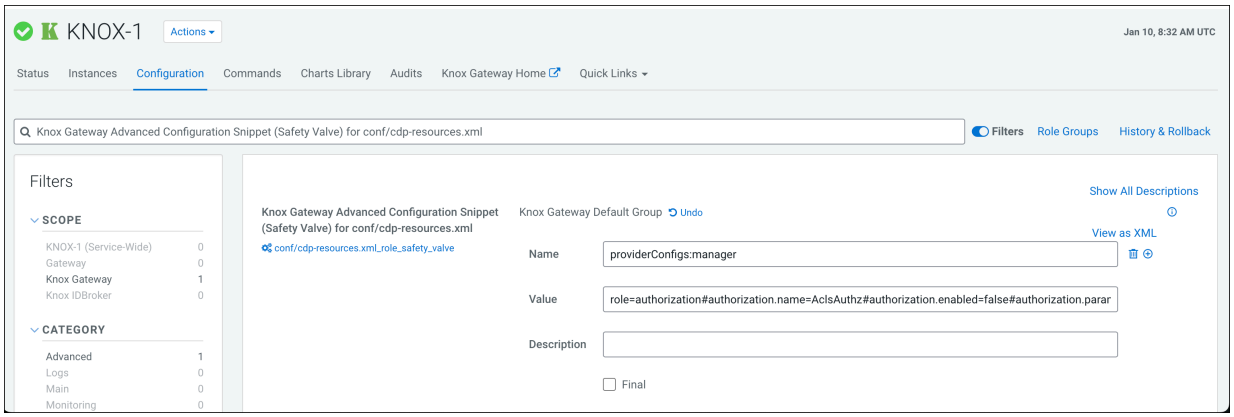

   With this change you are authorizing a user called myTestUser to login and execute administrative actions on the Knox Admin API.
2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/
gateway/admin/api/v1/providerconfig/manager'
{
  "providers" : [
 ...
  }, {
    "role" : "authorization",
    "name" : "AclsAuthz",
    "enabled" : false,
    "params" : {
      "knox.acl" : "myTestUser;KNOX_ADMIN_GROUPS;*",
      "knox.acl.mode" : "OR"
    }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
    "enabled" : true,
    "params" : {
      "ATLAS" : "enabled=true;maxFailoverAttempts=3;failoverSleep=1000;m
axRetryAttempts=300;retrySleep=1000"
    }
  } ]
}
```

Disable a provider in an existing provider configuration

An example of how to disable the authorization provider in the manager shared provider configuration.

About this task

In this example you will see how to disable the authorization provider in the manager shared provider configuration. This particular authorization provider is set as follows (in its JSON descriptor):

```
{
        "role": "authorization",
        "name": "AclsAuthz",
        "enabled": "true",
        "params": {
           "knox.acl.mode": "OR",
           "knox.acl": "KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;*"
        }
     }
```

Procedure

1. From Cloudera Manager Knox Configuration , add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml:
   - name = providerConfigs:manager
   - value = role=authorization#authorization.name=AclsAuthz#authorization.enabled=false#authorization.param.knox.acl=KNOX_ADMIN_USERS;KNOX_ADMIN_GROUPS;*#authorization.param.knox.acl.mode=OR

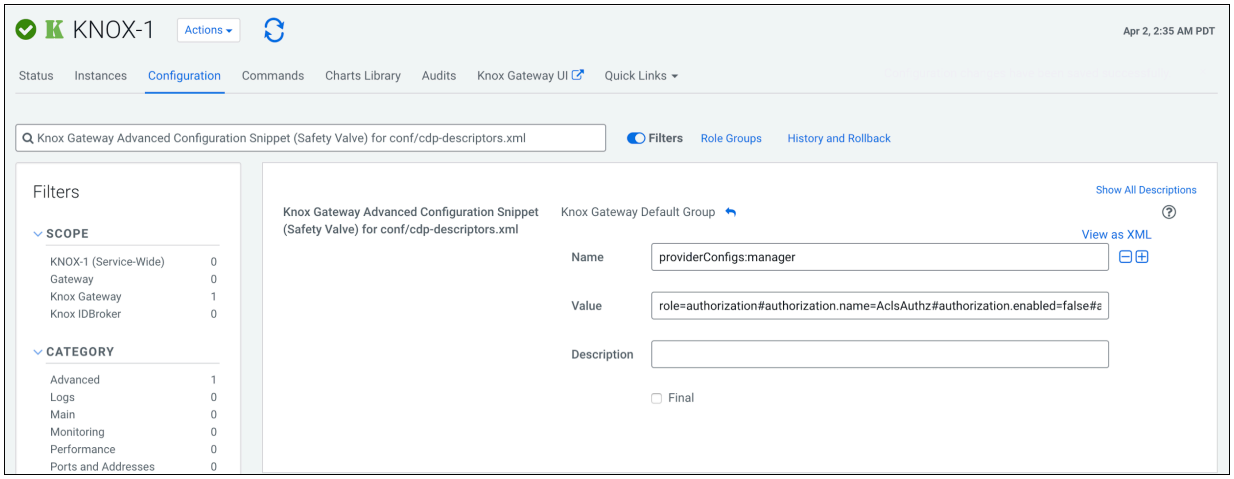

2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/
gateway/admin/api/v1/providerconfig/manager'
{
  "providers" : [
 ...
  }, {
    "role" : "authorization",
    "name" : "AclsAuthz",
    "enabled" : false,
    "params" : {
      "knox.acl" : "myTestUser;KNOX_ADMIN_GROUPS;*",
      "knox.acl.mode" : "OR"
    }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
```

```
    "enabled" : true,
    "params" : {
      "ATLAS" : "enabled=true;maxFailoverAttempts=3;failoverSleep=1000;m
axRetryAttempts=300;retrySleep=1000"
    }
  } ]
}
```

What to do next

The only change is that the enabled flag was changed to false.

Remove a provider parameter in an existing provider configuration

An example of how to remove the authentication parameter from a shared provider configuration.

About this task

In this example you will see how to remove an authentication provider parameter in the pam shared provider configuration. This particular provider is set as follows:

```
{
  "providers" : [ {
    "role" : "authentication",
    "name" : "ShiroProvider",
    "enabled" : true,
    "params" : {
      "main.pamRealm" : "org.apache.knox.gateway.shirorealm.KnoxPamRealm",
      "main.pamRealm.service" : "login",
      "sessionTimeout" : "30"
    }
  } ],
  "readOnly" : true
}
```

Procedure

1. From Cloudera Manager Knox Configuration , add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml:
   - name = providerConfigs:pam
   - value = role=authentication#authentication.name=ShiroProvider#authentication.param.remove=sessionTimeout#authentication.param.main.pamRealm=org.apache.knox.gateway.shirorealm.KnoxPamRealm#authentication.param.main.pamRealm.service=login
2. Save your changes.
3. Refresh the cluster.
4. Validate:

```
$ curl -ku <username>:<password> 'https://johndoe-1.abc.cloudera.com:8443/
gateway/admin/api/v1/providerconfig/pam'{
  "providers" : [ {
    "role" : "authentication",
    "name" : "ShiroProvider",
    "enabled" : true,
    "params" : {
      "main.pamRealm" : "org.apache.knox.gateway.shirorealm.KnoxPamRealm",
      "main.pamRealm.service" : "login"
    }
  } ],
  "readOnly" : true
```

```
}
```

Saving aliases

There is a new command available for the Knox Gateway role which allows end-users to save an alias=password pair to an arbitrary number of topologies on each host where an instance of the Knox Gateway is installed without the need of running the Knox CLI tool manually.

A new password-type input field is added, called save_alias_command_input_password. The format of an entry in this input field should be: topology_name_1[:topology_name_2:...:topology_name_N].alias_name=password

Example: cdp-proxy-api:admin:metadata.knoxLdapSystemPassword=guest-password.

After the end-users entered a meaningful and valid value and saved the configuration changes they can run the command from Knox's action list: Actions/Save Alias.

Tip: If you need to add a Gateway level alias, please use __gateway as topology name. For instance: __gateway.knoxLdapSystemPassword=admin-password.

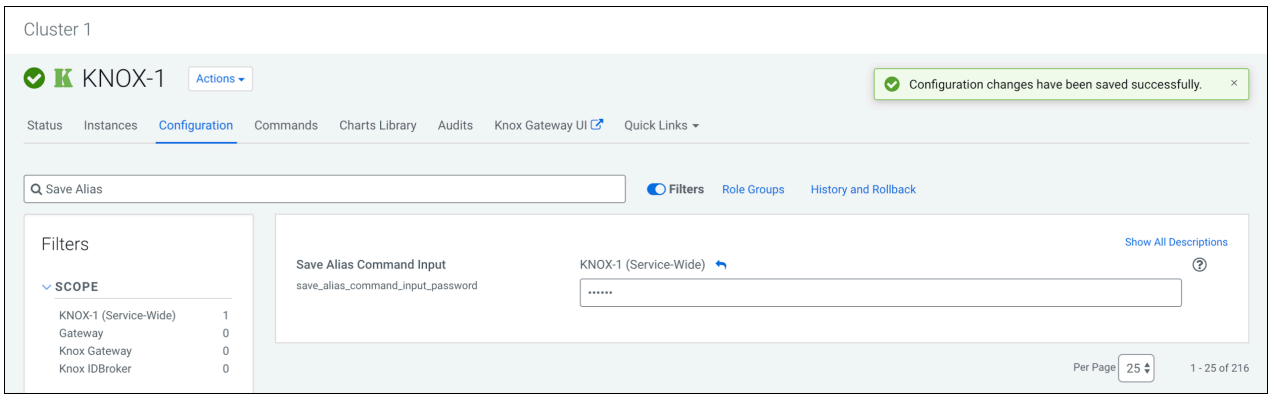

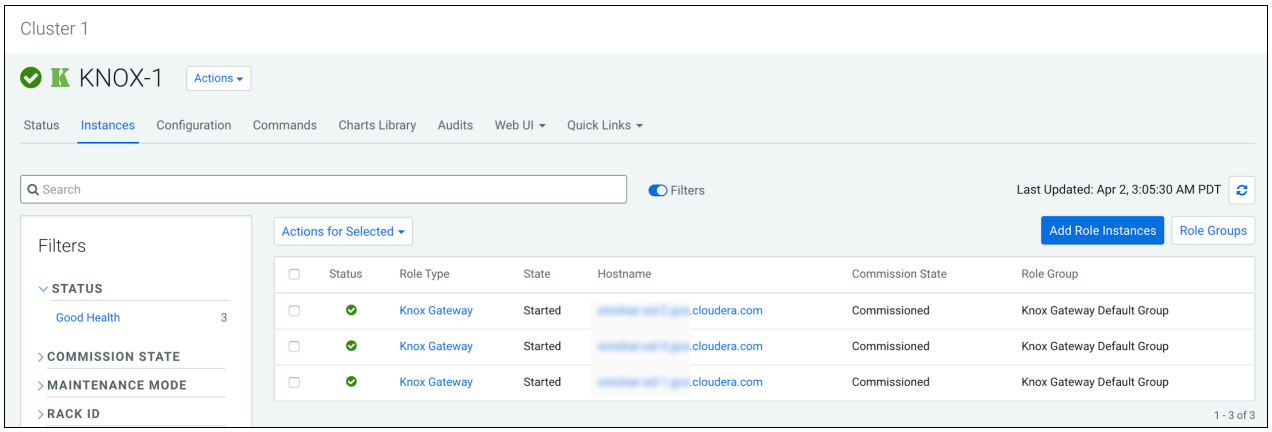

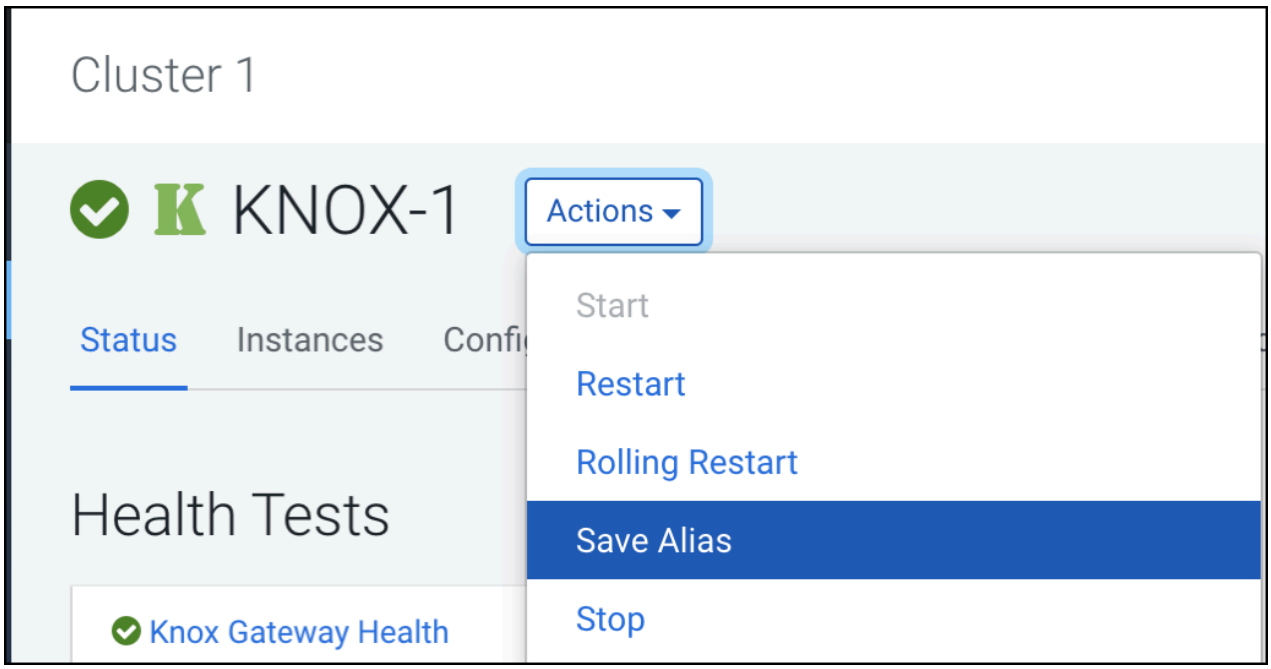
Cluster 1
KNOX-1
Actions
Status
Instances
Start
Restart
Rolling Restart
Save Alias
Stop
Health Tests
Knox Gateway Health

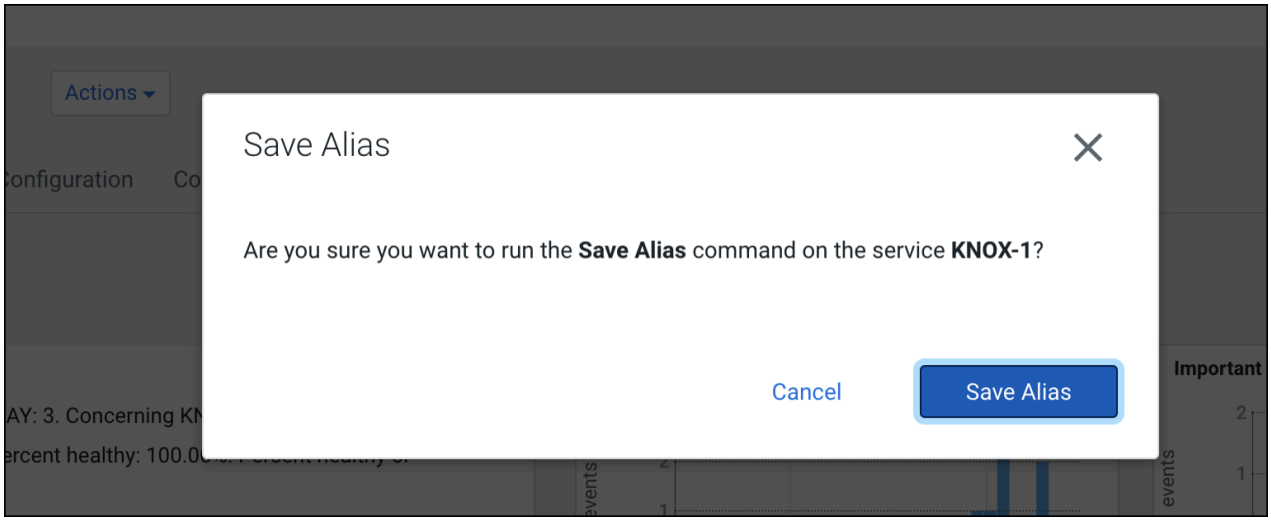
Actions
Save Alias
Are you sure you want to run the **Save Alias** command on the service **KNOX-1**?
Cancel
Save Alias
Important

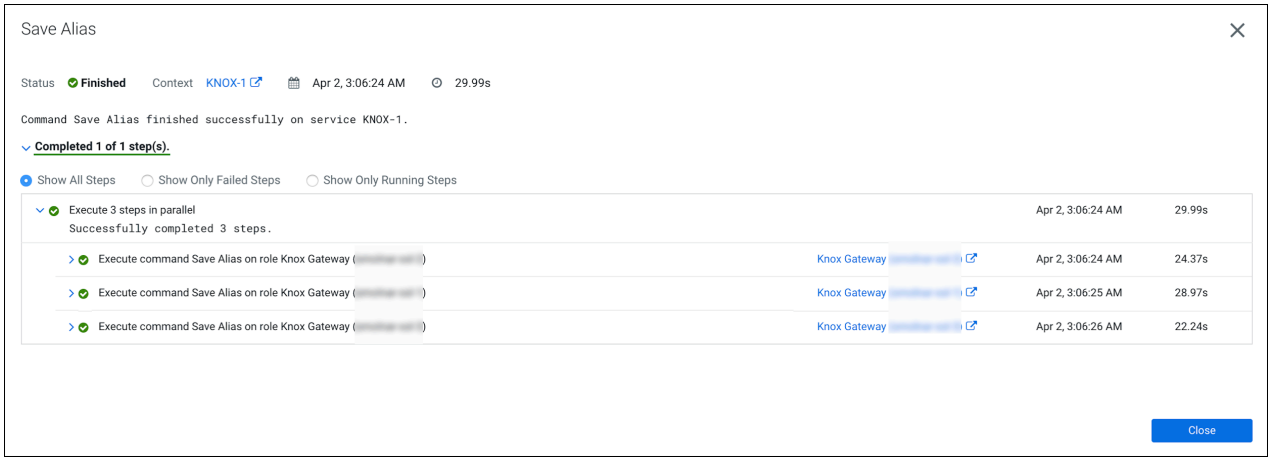
Save Alias
Status Finished Context KNOX-1 Apr 2, 3:06:24 AM 29.99s
Command Save Alias finished successfully on service KNOX-1.
Completed 1 of 1 step(s).
Show All Steps Show Only Failed Steps Show Only Running Steps
Execute 3 steps in parallel Apr 2, 3:06:24 AM 29.99s
Successfully completed 3 steps.
Execute command Save Alias on role Knox Gateway Knox Gateway Apr 2, 3:06:24 AM 24.37s
Execute command Save Alias on role Knox Gateway Knox Gateway Apr 2, 3:06:25 AM 28.97s
Execute command Save Alias on role Knox Gateway Knox Gateway Apr 2, 3:06:26 AM 22.24s
Close

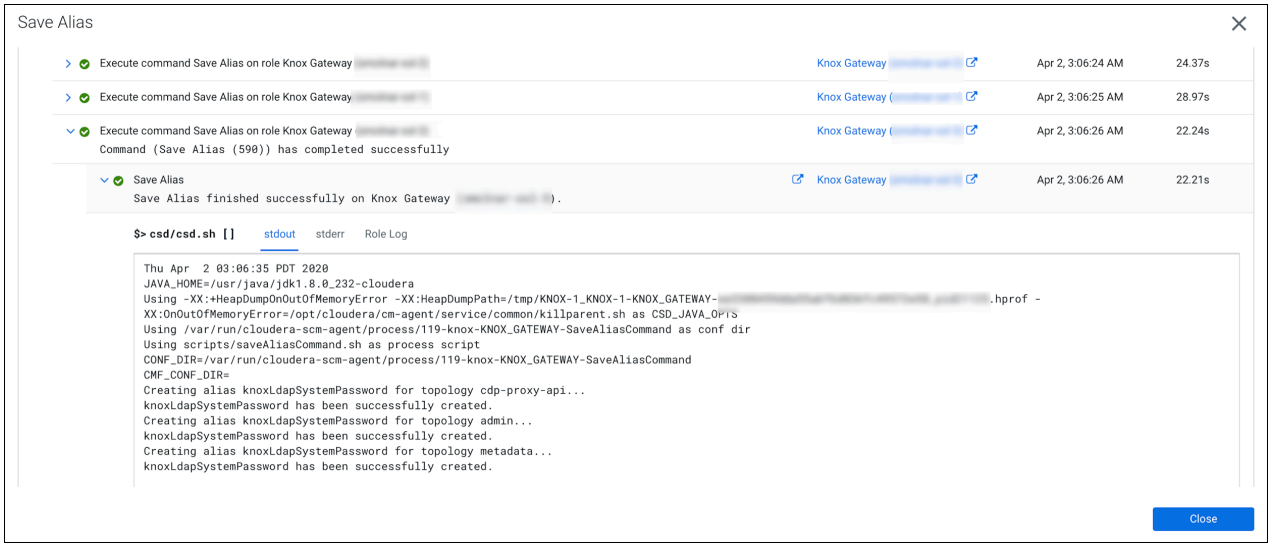

Configuring Kerberos authentication in Apache Knox shared providers

An example of how to add the kerberos-auth configuration provider from Cloudera Manager.

Procedure

1. From Cloudera Manager Knox Configuration , add the following entry in the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml:

   - Name = providerConfigs:kerberos-providers
   - Value =

   ```
   role=authentication#
   authentication.name=HadoopAuth#
   authentication.param.sessionTimeout=30#
   authentication.param.config.prefix=hadoop.auth.config#
   authentication.param.hadoop.auth.config.type=kerberos#
   authentication.param.hadoop.auth.config.signature.secret=${ALIAS=AUTH
   _CONFIG_SIGNATURE_SECRET}
   authentication.param.hadoop.auth.config.token.validity=1800#
   authentication.param.hadoop.auth.config.cookie.path=/#
   authentication.param.hadoop.auth.config.simple.anonymous.allowed=false#
   authentication.param.hadoop.auth.config.kerberos.principal=AUTH_CONFIG
   _KERBEROS_PRINCIPAL#
   authentication.param.hadoop.auth.config.kerberos.keytab=AUTH_CONFIG_KER
   BEROS_KEYTAB#
   authentication.param.hadoop.auth.config.kerberos.name.rules=DEFAULT
   ```

   Important: Paste the value as a single line, without line-breaks.

2. Add a safety valve name/value pair in Cloudera Manager Knox Configuration ,in Knox Gateway Environment Advanced Configuration Snippet (Safety Valve):

   ```
   Name = IDBROKER_KERBEROS_DT_PROXYUSER_BLOCK
   ```

```
Value = "proxyuser_block": "none"
```

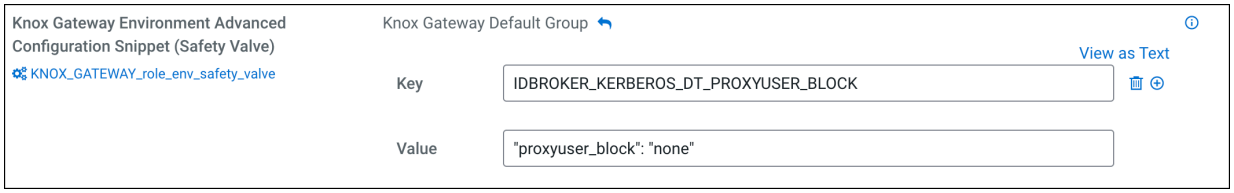

3. Save your changes.
4. Refresh the cluster.
5. Validate with a curl command: curl -k https://*HOST-10-00-100-100*:8443/gateway/admin/api/v1/providerconfig/kerberos-providers.

```
# curl -k https://host-10-00-100-100:8443/gateway/admin/api/v1/providerc
onfig/kerberos-providers
{
  "providers" : [ {
    "role" : "authentication",
    "name" : "HadoopAuth",
    "enabled" : true,
    "params" : {
      "config.prefix" : "hadoop.auth.config",
      "hadoop.auth.config.kerberos.keytab" : "/var/run/cloudera-scm-agent/
process/81-knox-KNOX_GATEWAY/knox.keytab",
      "hadoop.auth.config.kerberos.name.rules" : "DEFAULT",
      "hadoop.auth.config.kerberos.principal" : "HTTP/host-10-00-100-100.
coe.cloudera.com@CLOUDERA.COM",
      "hadoop.auth.config.signature.secret" : "${ALIAS=AUTH_CONFIG_SIGNA
TURE_SECRET}",
      "hadoop.auth.config.simple.anonymous.allowed" : "false",
      "hadoop.auth.config.token.validity" : "1800",
      "hadoop.auth.config.type" : "kerberos",
      "proxyuser_block" : "none"
    }
  }, {
    "role" : "identity-assertion",
    "name" : "HadoopGroupProvider",
    "enabled" : true,
    "params" : {
      "CENTRAL_GROUP_CONFIG_PREFIX" : "gateway.group.config."
    }
  }, {
    "role" : "authorization",
    "name" : "XASecurePDPKnox",
    "enabled" : true,
    "params" : { }
  }, {
    "role" : "ha",
    "name" : "HaProvider",
    "enabled" : true,
    "params" : {
      "HBASE" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true",
      "HIVE" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true;zoo
keeperEnsemble=maxFailoverAttempts=3;failoverSleep=1000;enabled=true;zoo
keeperEnsemble=gbl20175161.systems.uk.company:2181,gbl20175162.systems.u
k.company:2181,gbl20175163.systems.uk.company:2181;zookeeperNamespace=hi
veserver2",
      "OOZIE" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true",
      "WEBHCAT" : "maxFailoverAttempts=3;failoverSleep=1000;enabled=true",
      "WEBHDFS" : "maxFailoverAttempts=3;failoverSleep=1000;maxRetryAtte
mpts=300;retrySleep=1000;enabled=true"
```

```
      }
    } ],
    "readOnly" : true
  }
```

Related Information

Saving aliases

Configuring group mapping in Knox

Learn how to use HadoopGroupProvider to configure group mapping.

About this task

The role for this provider is identity-assertion and name is HadoopGroupProvider:

```
<provider>
   <role>identity-assertion</role>
   <name>HadoopGroupProvider</name>
   <enabled>true</enabled>
   <<param> ... </param>
</provider>
```

All the configurations for HadoopGroupProvider reside in the provider section of a gateway topology file. The hadoop.security.group.mapping property determines the implementation. Some of the valid implementations are as follows:

- org.apache.hadoop.security.JniBasedUnixGroupsMappingWithFallback

  This is the default implementation and is picked up if hadoop.security.group.mapping is not specified. This implementation determines if the Java Native Interface (JNI) is available. If JNI is available, the implementation uses the API to resolve a list of groups for a user. If JNI is not available then the shell implementation, org.apache.hadoop.security.ShellBasedUnixGroupsMapping is used. It shells out with the `bash -c groups` command (for a Linux/Unix environment) or the `net group` command (for a Windows environment) to resolve a list of groups for a user.
- org.apache.hadoop.security.LdapGroupsMapping

  This implementation connects directly to an LDAP server to resolve the list of groups. However, this should only be used if the required groups reside exclusively in LDAP, and are not materialized on the Unix servers.

To enable group lookup using identity assertion as HadoopGroupProvider, perform the following steps:

Procedure

1. Go to Cloudera Manager Knox Configuration .
2. Search for the Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml property, and add the following entry:

```
# name = providerConfigs:sso
# value = role=federation#federation.name=SSOCookieProvider#federation.
param.sso.authentication.provider.url=
https://<knox_hostname>:8443/gateway/knoxsso/api/v1/websso#role=identity-
assertion#identity-assertio...
HadoopGroupProvider#identity-assertion.enabled=true#identity-assertion.pa
ram.CENTRAL_GROUP_CONFIG_PREFIX=gateway.group.
```

```
config#role=authorization#authorization.name=XASecurePDPKnox#authorizatio
n.enabled=true
```

Note: Replace your Knox hostname in <knox_hostname>.

The following is a sample demo of LDAP configurations for the LDAP setting:

```
<param>
<name>gateway.group.config.hadoop.security.group.mapping</name>
 <value>org.apache.hadoop.security.LdapGroupsMapping</value>

</param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.bind.user</
name>
 <value>uid=tom,ou=people,dc=hadoop,dc=apache,dc=org</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.bind.pass
word</name>
 <value>tom-password</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.url</name>
 <value>ldap://localhost:33389</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.base</name>
 <value></value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.search.fil
ter.user</name>
 <value>(&amp;(|(objectclass=person)(objectclass=applicationProcess))(cn
={0}))</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.search.filt
er.group</name>
 <value>(objectclass=groupOfNames)</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.search.attr
.member</name>
 <value>member</value>
 </param>
 <param>
 <name>gateway.group.config.hadoop.security.group.mapping.ldap.search.a
ttr.group.name</name>
 <value>cn</value>
 </param>
 </provider>
```

3. Save your changes.
4. Refresh the cluster.

Management of services for Apache Knox through Cloudera Manager

You can enable or disable known or custom services in Knox proxy through Cloudera Manager.

There are two kinds of services in cdp-proxy:

- Known: Officially-supported Knox services. Cloudera Manager provides and manages all the required service definition files.
- Custom: Unofficial, tech preview, or community feature Knox services. You must ensure that the service definition files (service.xml and rewrite.xml) exist in the $CDP_PARCEL_DIR/lib/knox/data/services folder. These are not recommended for production environments, and not supported by Cloudera.

Note: If you upgrade to a newer version of Cloudera, the previously added custom service definitions might disappear from the new location. Therefore, you must ensure that those files are there after upgrading to a newer Cloudera version.

Important:

These topologies will be deployed by Cloudera Manager only if Knox's service auto-discovery feature is turned on using the Enable/Disable Service Auto-Discovery checkbox on Cloudera Manager UI:

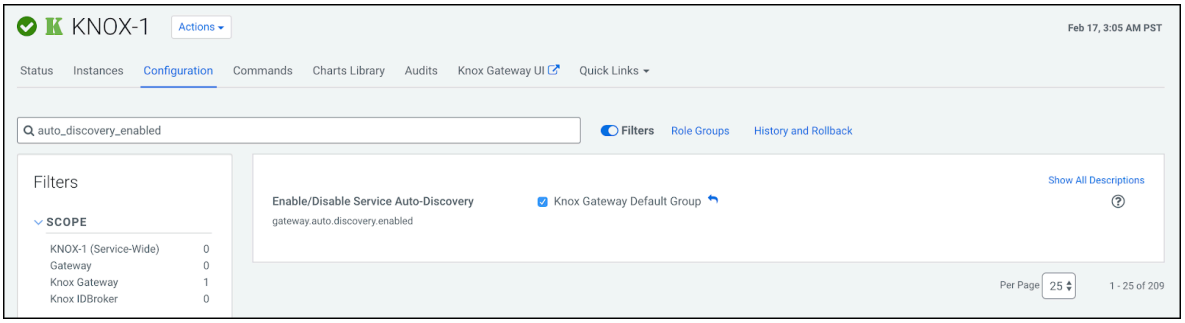

For a comprehensive list of known services that can be enabled, see "Knox Supported Services Matrix".

Related Information

Knox Supported Services Matrix

Enable proxy for a known service in Apache Knox

How to enable auto-discovery for a known service in Knox proxy via Cloudera Manager.

About this task

"Known" services are officially-supported Knox services (like Apache Atlas, Ranger, Solr, etc.) Cloudera Manager provides and manages all the required service definition files.

For the purposes of this example, we add ATLAS and ATLAS UI to cdp-proxy. You can add more services; for a comprehensive list of knoxn services that can be enabled, see "Knox Supported Services Matrix".

Procedure

1. From Cloudera Manager Knox Configuration , check the Gateway Auto Discovery (cdp-proxy) - $Component boxes.
   In this example, we enable:
   - gateway_auto_discovery_cdp_proxy_enabled_atlas
   - gateway_auto_discovery_cdp_proxy_enabled_atlas_ui

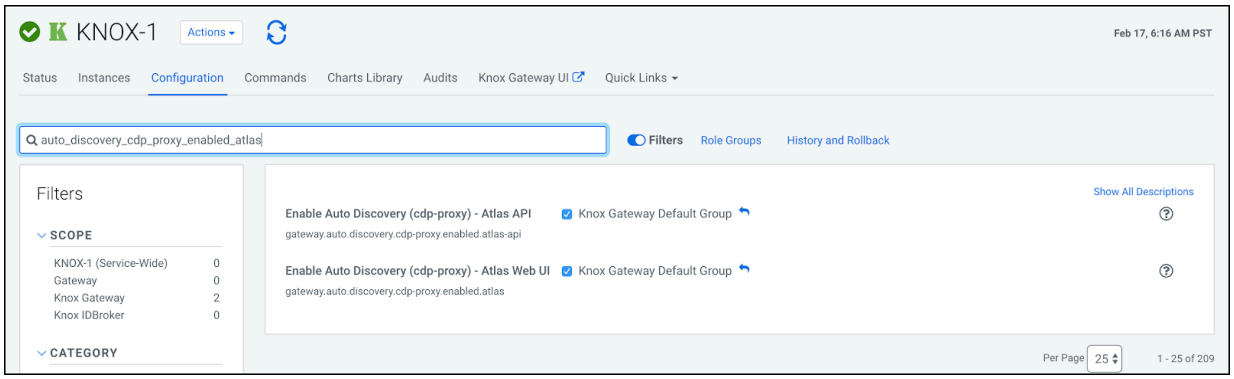

2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process finishes.

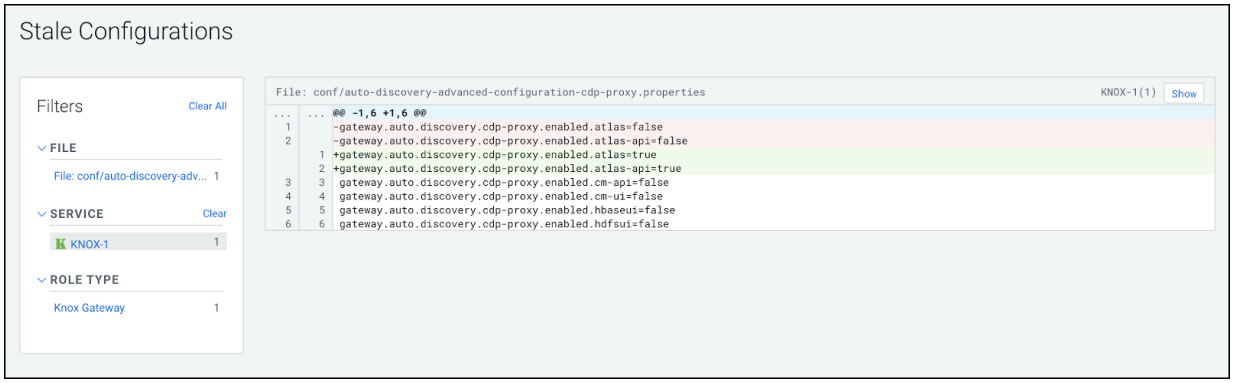

4. Validate that ATLAS in cdp-proxy was added by going to the following URL: http s://*$KNOX_GATEWAY_HOST*:*$PORT*/*$GATEWAY_PATH*/admin/api/v1/topologies/cdp-proxy.

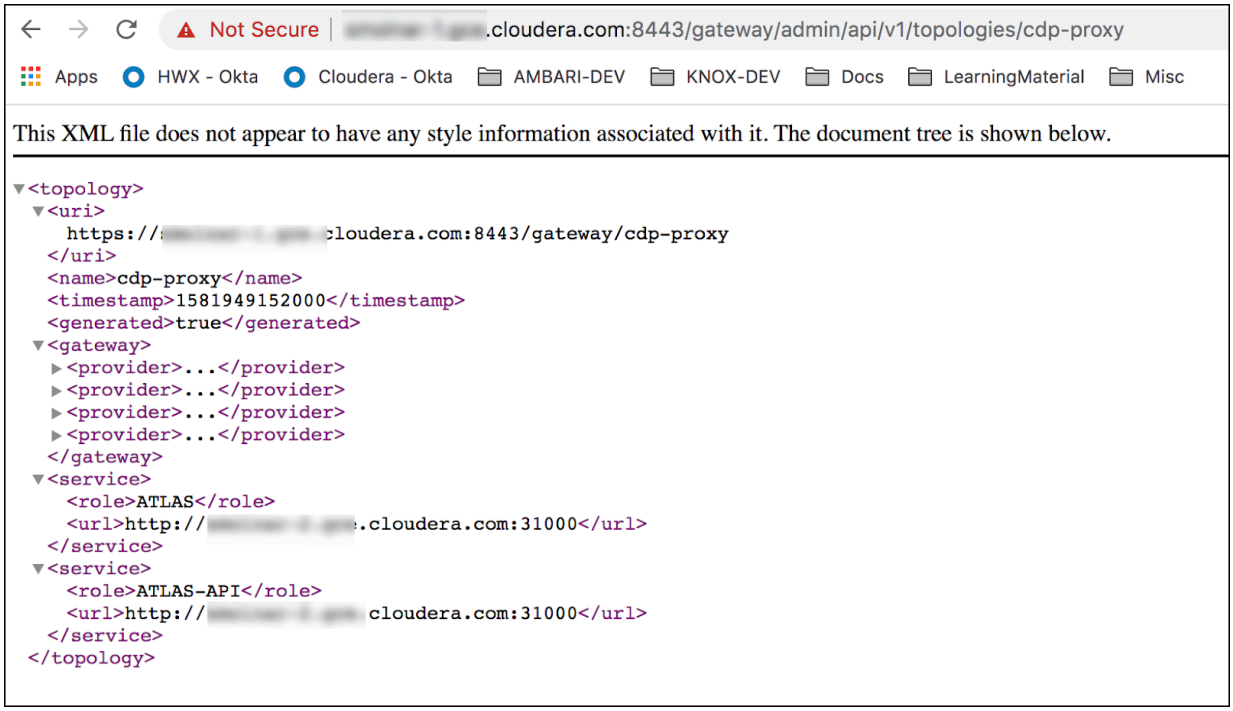

Related Information

Knox Supported Services Matrix

Disable proxy for a known service in Apache Knox

How to remove auto-discovery for a known service in Knox proxy via Cloudera Manager.

About this task

“Known” services are officially-supported Knox services (like Apache Atlas, Ranger, Solr, etc.) Cloudera Manager provides and manages all the required service definition files.

In this example, we are going to remove the previously added ATLAS and ATLAS-UI services from cdp-proxy. We disable the gateway_auto_discovery_cdp_proxy_enabled_atlas and gateway_auto_discovery_cdp_proxy_enabled_atl as_ui checkboxes on Knox’s Configuration page in CM, save the changes and refresh the cluster.

Procedure

1. From Cloudera Manager Knox Configuration , uncheck the Gateway Auto Discovery (cdp-proxy) - $Component boxes.
   In this example, we disable:
   - gateway_auto_discovery_cdp_proxy_enabled_atlas
   - gateway_auto_discovery_cdp_proxy_enabled_atlas_ui

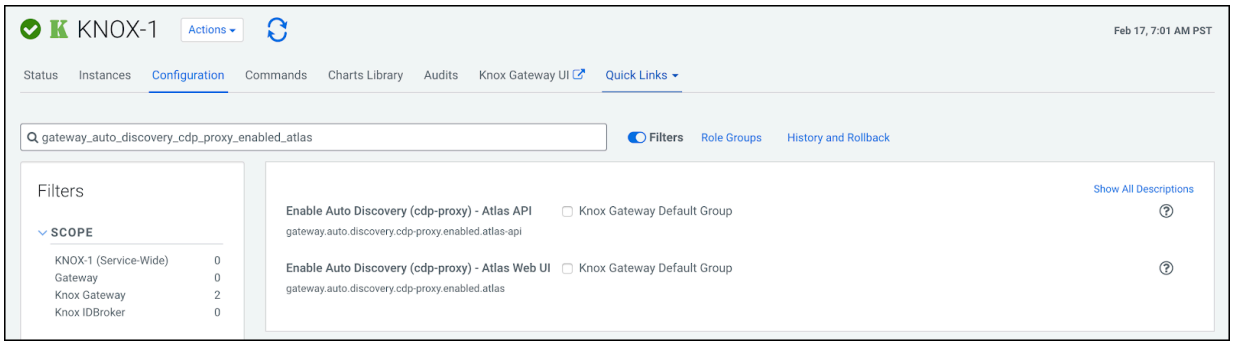

2. Save your changes.
3. The ‘Refresh needed’ stale configuration indicator appears; click it and wait until the refresh process finishes.

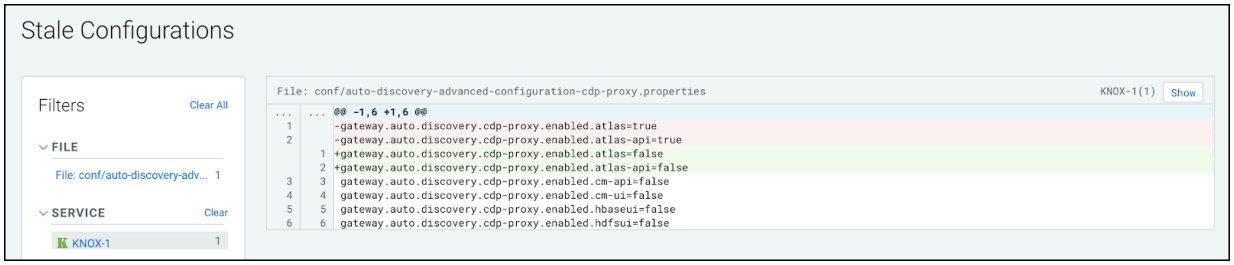

4. Validate that custom service got removed by going to the following URL: http s://*$KNOX_GATEWAY_HOST*:*$PORT*/*$GATEWAY_PATH*/admin/api/v1/topologies/cdp-proxy.

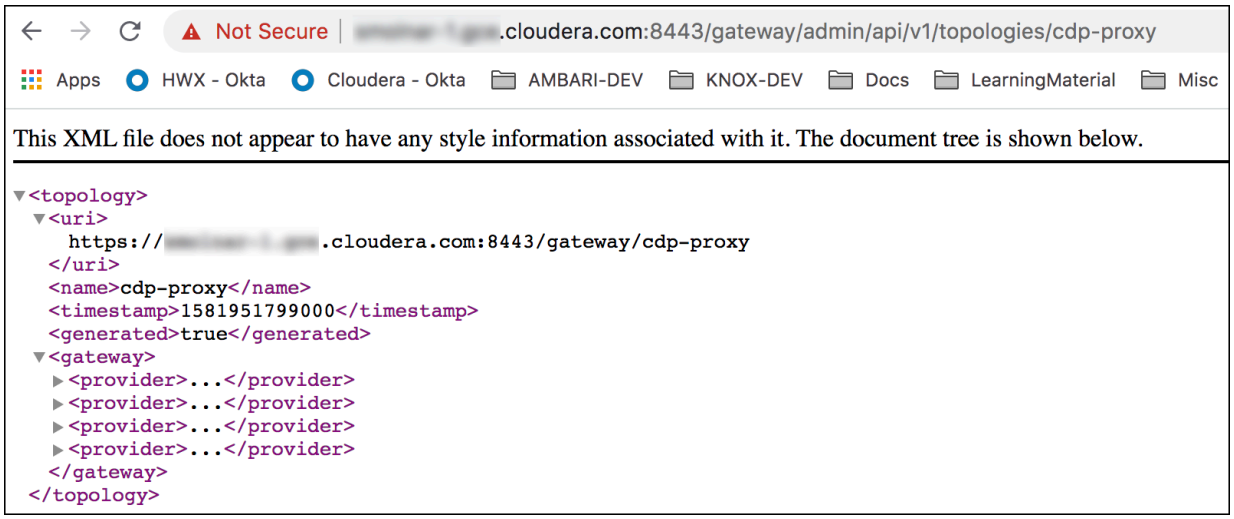

Add a custom descriptor to Apache Knox

How to add a custom descriptor to Apache Knox using Cloudera Manager.

About this task

Custom descriptors can be deployed to Apache Knox using Cloudera Manager. These descriptors, combined with referenced provider configurations, are transformed into Knox topologies. Using Cloudera Manager means that these descriptors only ever need to be changed in one place to affect all Knox Gateway instances in the cluster.

Fundamentally, descriptors contain the declaration of services to proxy and a reference to provider configuration defining how authentication and authorization for those proxied services should be handled. A descriptor also may similarly declare Knox applications as topologies do.

Service declarations consist of at least the name of the service being proxied. They optionally include one or more endpoint URLs and one or more service-specific parameters.

Descriptors optionally include discovery information, allowing Knox to dynamically discover the endpoint URLs for the declared services.

Procedure

1. Define the descriptor contents:
   a) From Cloudera Manager Knox Configuration , add a new entry in Knox Gateway Advanced Configuration Snippet (Safety Valve) for conf/cdp-resources.xml_role_safety_valve.
   b) Name the topology, specify the providerConfigRef, and enumerate the services and associated service URLs.

      Optional service details include version (E.G., HIVE:version=0.13.0) and service parameters (E.G., HIVE:httpclient.connectionTimeout=5m)

      Note: The following are predefined read-only topologies:
      - admin
      - cdp-proxy
      - cdp-proxy-api
      - cdp-proxy-token
      - knoxsso
      - manager

      These names cannot be used when defining a custom descriptor, and these topologies cannot be changed from the CM UI.

   Static URL Example (HIVE and WEBHDFS with PAM authentication)

   - Name=my-custom-topology
   - Value=

   ```
   providerConfigRef=pam#
   HIVE:url=https://hive-host-1:10001/cliservice#
   WEBHDFS:url=https://hdfs-host-1:20470/webhdfs#
   WEBHDFS:url=https://hdfs-host-2:20470/webhdfs
   ```

   Discovery Example (HIVE and WEBHDFS with PAM authentication)

   Note: If the Cloudera cluster is not enabled with Auto-TLS, then you must add the Cloudera Manager certificate to the Knox truststore and restart the Knox service.

   - Name=my-discoverable-topology
   - Value=

   ```
   discoveryType=ClouderaManager#
   discoveryAddress=https://cm-host:7183#
   cluster=Cluster 1#
   providerConfigRef=pam#
   HIVE#
   ```

```
WEBHDFS
```

Note: The gateway can monitor cluster configurations, and respond to changes by dynamically regenerating and redeploying the affected topologies. This feature is turned off by default. To turn it on, add the following safety valve:

Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml => gateway.cluster.config.monitor.cm.enabled = true.

2. Save the changes.
3. Refresh the Knox instances' configuration: the Refresh needed stale configuration indicator appears; click it and wait until the refresh process completes.
4. Validate using the Knox homepage.

   Verify that your topology is generated with the services and URLs you specified.

Load balancing for Apache Knox

Knox offers load balancing using a simple round robin algorithm which prevents load on one specific node.

- For services that are stateless, Knox loadbalances them using a simple round robin algorithm which prevents load on one specific node.
- For services that are stateful (i.e., require sessions, such as Ranger and Hive,) sessions are loadbalanced using a round robin algorithm, where each new session will use a different host and all the requests in the same session will be routed to the same host. This will continue until a session terminates or there is a failover.
- In case of failover, services that are stateful will return error response 502.

This behavior is configurable and can be changed by tuning various flags in Knox HA provider for the respective services.

Load balancing vs high availability (HA)

Currently, Knox offers load balancing using a simple round robin algorithm which prevents load on one specific node.

Because we do not support session persistence, this is not true HA, as there could be a case where stateful service will not failover to other node.

Supported services

The following services support Knox load balancing:

- Hive
- Impala
- Oozie
- Phoenix
- Ranger
- Solr

Default enabled values

The following default values are enabled in the Knox topology. API is located in cdp-proxy-api.xml; UI is located in cdp-proxy.xml.

- Hive
  - API: enableStickySession=true;noFallback=true;enableLoadBalancing=true

- Phoenix
  - API: enableStickySession=true;noFallback=true;enableLoadBalancing=true
- Ranger
  - API: enableStickySession=false;noFallback=false;enableLoadBalancing=true
  - UI: enableStickySession=true;noFallback=true;enableLoadBalancing=true
- Solr
  - API: enableStickySession=false;noFallback=false;enableLoadBalancing=true
  - UI: enableStickySession=true;noFallback=true;enableLoadBalancing=true

Generate and configure a signing keystore for Knox in HA

When Knox is installed on more than one instance (i.e., when Knox is running in HA), then signing keystore configurations must be set in Cloudera Manager.

Procedure

1. Generate your own certificate and keystore file, and then copy to /var/lib/knox/gateway/data/security/keystores/.
2. Set the following values:
   - gateway_signing_keystore_name: the filename of keystore file that contains the signing keypair.
   - gateway_signing_keystore_type: the type of the keystore file where the signing keypair is stored. In non-FIPS environments, this should be PKCS12.
   - gateway_signing_key_alias: the alias for the signing keypair within the keystore.
3. If you do not want the master secret to be used, you can set an alias for the password to the keystore file that holds the signing keypair.
   a) Go to Saving Aliases and follow the instructions.
   b) From Cloudera Manager Knox Configuration Knox Service (or Gateway) Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml_service_safety_valve: , configure gateway.signing.keystore.passwor d.alias to the alias previously defined.

Knox Gateway token integration

You can use the Apache Knox homepage to generate and manage Knox Gateway tokens for Cloudera.

Related Information

Knox token management (in v1.6.0 and above)

Overview

Instead of using a basic username/password pair, you can improve security by generating Knox Gateway tokens. Tokens are more secure than plaintext username/password because they are signed, anonymized from the source data, and have a specified lifetime (by default, one hour).

About Knox gateway tokens

Before Cloudera 7.2.14, Knox on Cloudera Public Cloud had two default topologies: cdp-proxy and cdp-proxy-api. To enable passcode tokens, a third Knox topology was added: cdp-proxy-token. While very similar to cdp-proxy-api, the authentication provider for cdp-proxy-token is configured with the JWTFederation provider, so that newly generated tokens can be used.

View Knox token integration

Knox token integration can be accessed via Cloudera Manager or the Knox homepage:

- (Recommended) Cloudera Manager: Cloudera Manager Clusters Knox Configuration and search for “Knox Token Integration”.

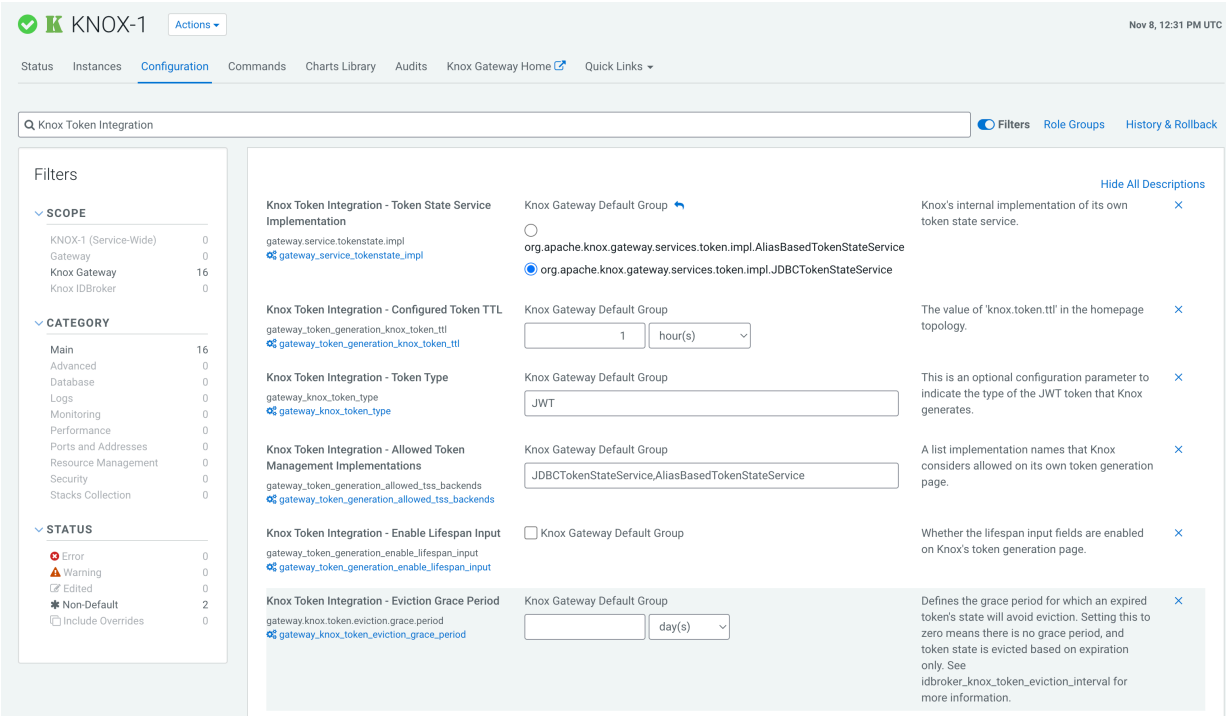

- Navigate to the Cloudera Management Console service > Data Lakes > (Your cluster) > Token Integration (under the Services tab). This will bring you to the Knox homepage. There are two new links on your Knox homepage homepage: Token Management and Token Generation.

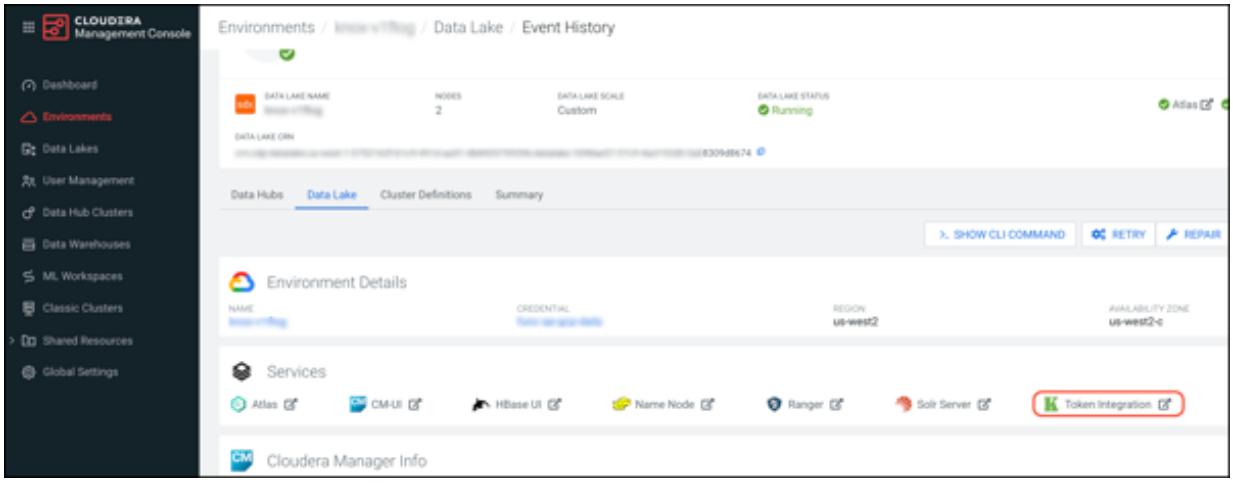

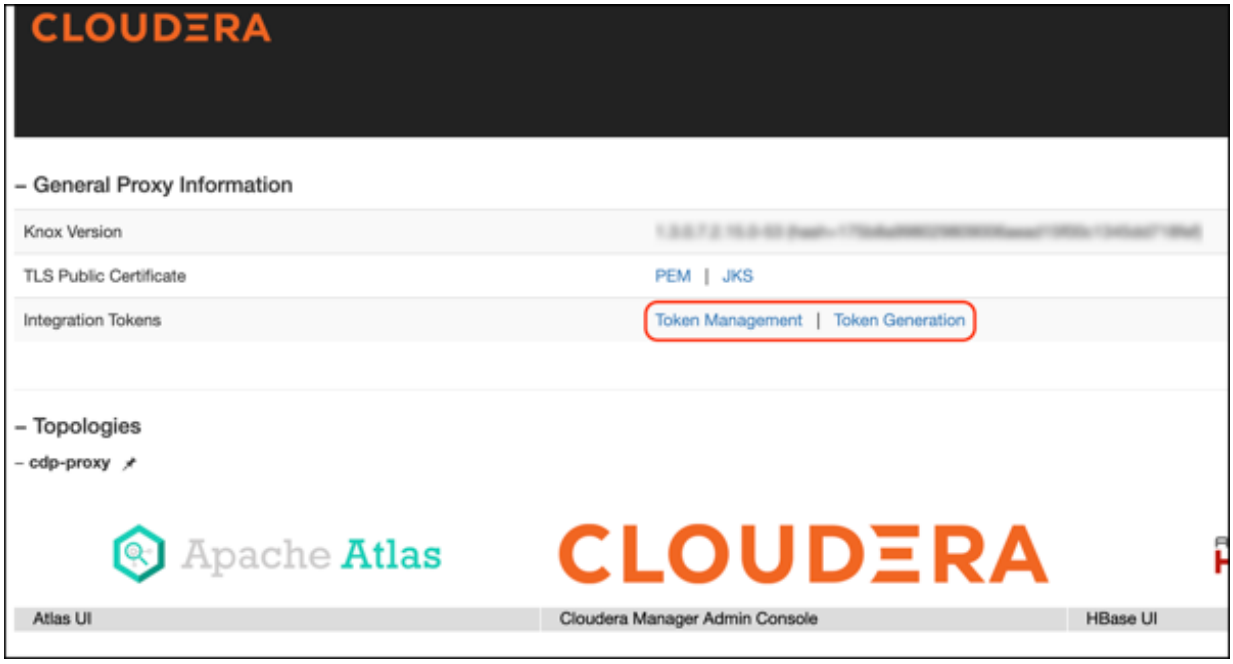

Knox token integration in Cloudera works out of the box using the Knox Token Generation page. However, the token integration API can be re-used in your own custom topology.

Attention: The only restriction of the above approach is that your custom topology must not use the HadoopAuth authentication provider because it won't work with the KNOXTOKEN service due to a known issue (which will be fixed in future releases).

Token configurations

The default configurations for Knox token integration are as follows.

Default configurations

Table 7: Default token configurations

Property	Sample values	Default
Token State Service Implementation	Knox's internal implementation of its own token state service.	org.apache.knox.gateway.services.token.impl.JDBCTokenStateService
Allowed Token Management Implementations	A list of implementation names that Knox considers allowed on its own token generation page.	JDBCTokenStateService,AliasBasedTokenStateService
Configured Token TTL	The value of “knox.token.ttl” in the homepage topology.	1 hour
Token Type	This is an optional configuration parameter to indicate the type of the JWT token that Knox generates.	JWT
Enable Lifespan Input	Whether the lifespan input fields are enabled on Knox's token generation page.	false
User Limit	The number of tokens a user is allowed to manage. Setting this to -1 indicates unlimited token management.	10
User Limit Exceeded Action	The action Knox will take if user limit is exceeded while trying to create a new Knox Token. If REMOVE_OLDEST is selected then the oldest token of the user, who the token is being generated for, will be removed. Otherwise, if RETURN_ERROR is selected, Knox will return an error response with 403 error code.	RETURN_ERROR
Renewer Whitelist	This is an optional configuration parameter to authorize the comma-separated list of users to invoke the associated token renewal and/or revocation APIs.	empty string
JWKS URL	This optional configuration parameter enables end-users to declare their JWKS URL. The JSON Web Key Set (JWKS) is a set of keys containing the public keys used to verify any JSON Web Token (JWT) issued by the authorization server and signed using the RS256 signing algorithm.	empty string
Allowed JWS Types	This is an optional configuration parameter to allow a comma-separated list of token types Knox will allow while validating JSON Web Signature (JWS) using JWKS URLs. Defaults to “JWT”. The typical customized value is “at +jwt, JWT”.	JWT
Expected Principal Claim	If that configuration parameter is defined, Knox will use this to get the value of this claim from the submitted JWT upon verification instead of using the default principal.	empty string
Expected JWT Signature Algorithm	Indicates the expected signature algorithm Knox should use to verify the submitted JWT's signature. If not defined, Knox will use 'RS256'.	empty string
Expected JWT Issuer	Indicates the expected issuer of a received token must match. If not defined, Knox will use 'KNOXSSO'.	empty string
Enable Impersonation	Indicates if Knox Token impersonation is enabled.	false

Property	Sample values	Default
Proxyuser Block	If Knox token impersonation is enabled, it allows the specified user(s) to impersonate members of certain groups from specific hosts. For example, the following values allow a user named "changeme" to impersonate members of any group from any host: • "knox.token.proxyuser.changeme.hosts": "*" • "knox.token.proxyuser.changeme.groups": "*" The values for this property must be in JSON key/value format.	empty string

Default configurations seen from Cloudera Manager:

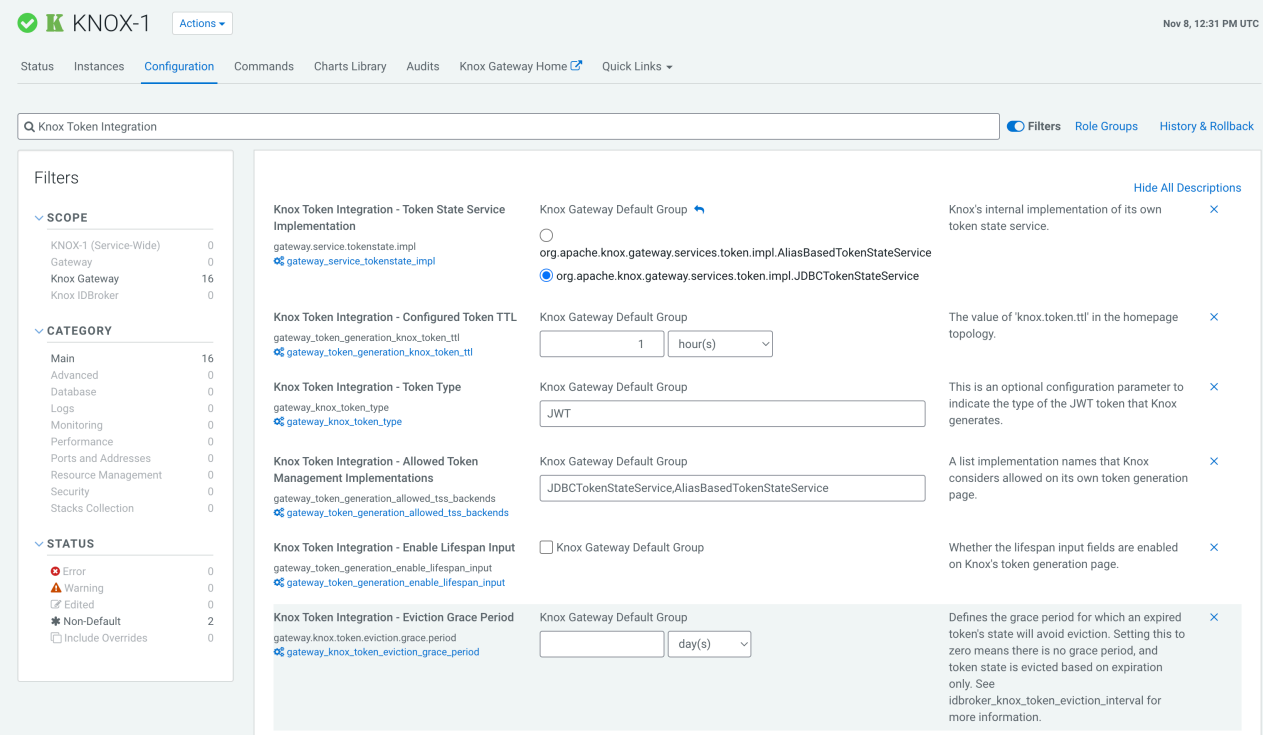

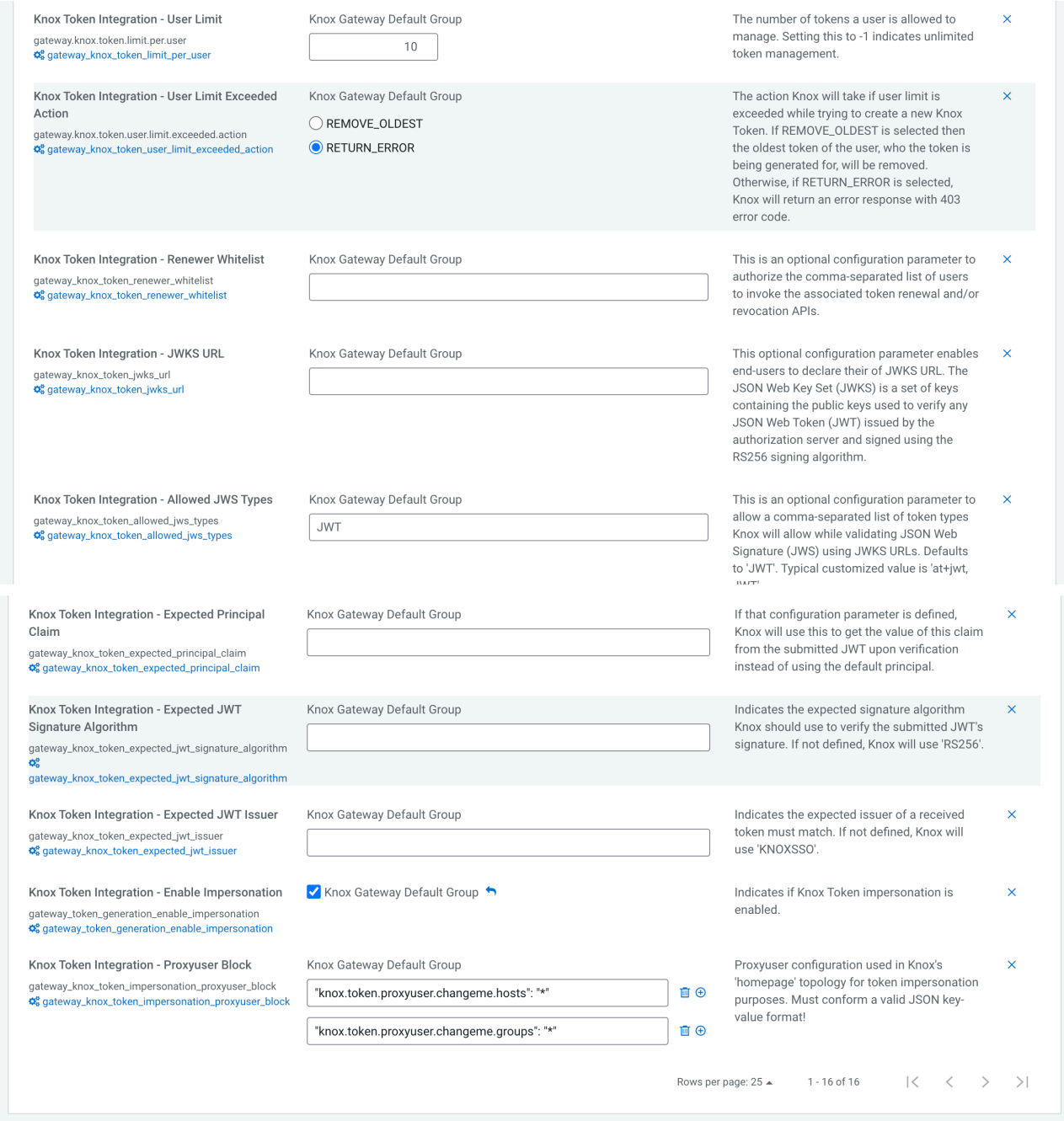

Default configurations seen from the Knox homepage UI:

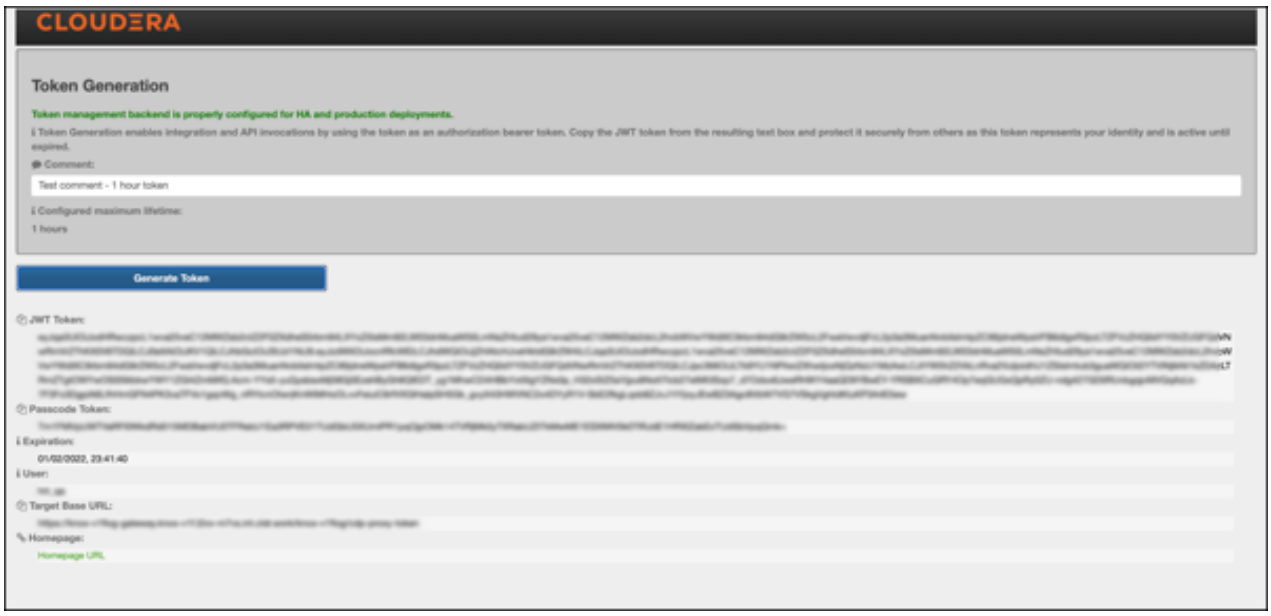

Database connection properties

Optional database connection properties that you can declare individually:

- gateway.database.type: Set to postgresql or mysql.
- gateway.database.host: Host where your DB server is running.
- gateway.database.port: Port that your DB server is listening on.
- gateway.database.name: Name of the database you are connecting to.

Important: From Cloudera Private Cloud Base 7.1.9 SP1 CHF1 release onwards, if you are using a MySQL database, then the JDBC driver must be installed manually to /usr/share/java. For more information, see Installing the MySQL JDBC Driver.

Token TTL details

Out of the box, Knox will display the custom lifetime spinners on the Token Generation page. However, they can be hidden by disabling the Knox Token Integration - Enable Lifespan Input checkbox on the CM UI. Given that input property, and the configured maximum lifetime property, the generated token can have the following TTL value:

- If there is no configured token TTL and lifespan inputs are disabled, the default TTL is used (30 seconds).
- If there is configured TTL and lifespan inputs are disabled, the configured TTL is used.
- If there is configured TTL and lifespan inputs are enabled and lifespan inputs result in a value that is less than or equal to the configured TTL, the lifespan query param is used.
- If there is configured TTL and lifespan inputs are enabled and lifespan inputs result in a value that is greater than the configured TTL, the configured TTL is used.

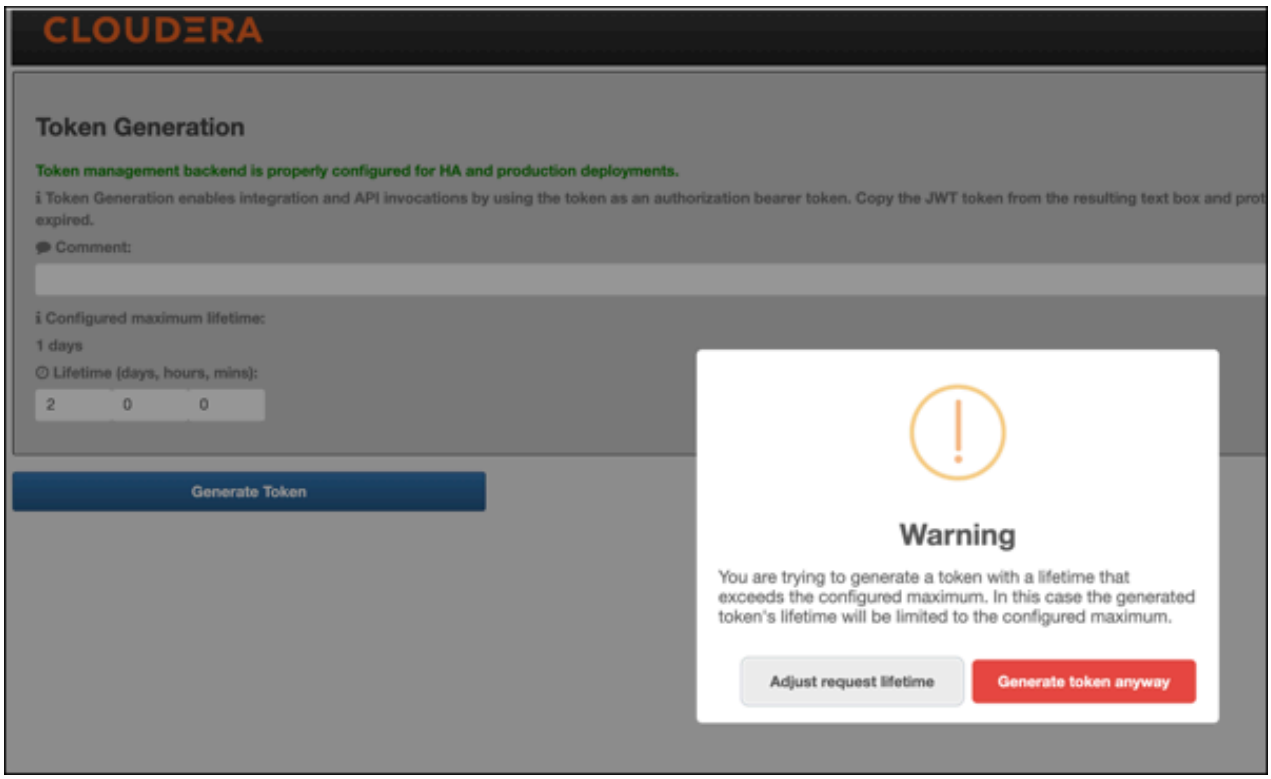

Generate-jwk options

CM automatically creates a token hash key for you. But if you want to do this manually, such as when scripting, configure the knox.token.hash.key alias with:

```
generate-jwk --saveAlias knox.token.hash.key
```

This generates a JSON Web Key using the supplied algorithm name.

Table 8: Options

Option	Description	Sample values
jwkAlg	(Optional) The desired JSON Web Signature algorithm name. Determines if the gateway-level alias is configured with a 256, 384, or 512-bit length JWK.	HS256 (Default) HS384 HS512
saveAlias	(Optional, Recommended) Given alias name used to save the generated JWK, instead of printing this sensitive information on the screen.	knox.token.hash.key
topology	(Optional) Name of the topology (i.e., the cluster) to be used when saving the JWK as an alias. If none specified, the alias is going to be saved for the Gateway.	cdp-proxy (Default) cdp-proxy api

Generate tokens

How to generate Knox gateway tokens from the Knox homepage.

Procedure

1. To access Knox generation management, go to https://*KNOX_GATEWAY_HOST*:*PORT*/*GATEWAY_PATH*/homepage/home, e.g. https://localhost:8443/gateway/homepage/home. Click on Token Generation.

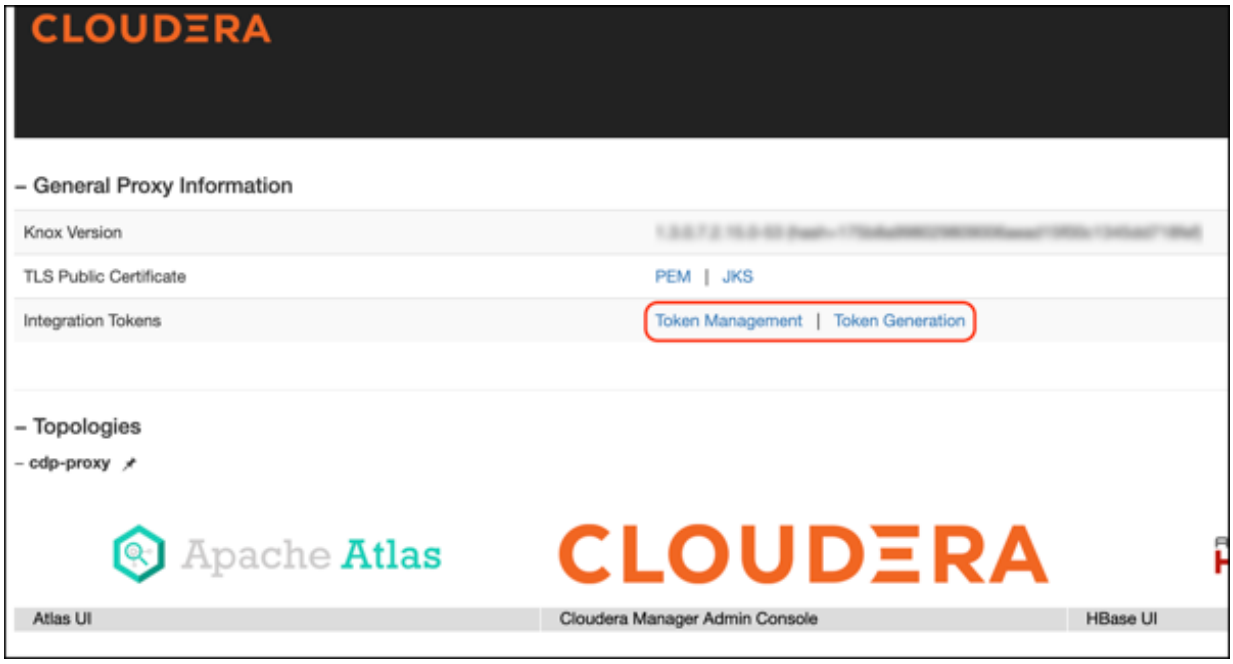

2. The following sections are displayed on the page:
   - Status bar: Message about the configured token state backend. There are 3 different statuses:
     - ERROR: Displayed in red. Indicates a problem with the service backend which makes the feature not work. Usually, this is visible when end-users configure JDBC token state service, but they make a mistake in their DB settings.
     - WARN: Displayed in yellow. Indicates that the feature is enabled and working, but there are some limitations.
     - INFO: Displayed in green. Indicates when the token management backend is properly configured for HA and production deployments.
   - Information label: Explains the purpose of the **Token Generation** page.
   - Comment: Optional input field that allows end-users to add meaningful comments (mnemonics) to their generated tokens. The maximum length is 255 characters.
   - Configured maximum lifetime: Informs the clients about the knox.token.ttl property set in the homepage topology (defaults to 1 day(s)). If that property is not set (e.g. someone removes it from he homepage topology), Knox uses a hard-coded value of 30 seconds (aka. default Knox token TTL).
   - Custom maximum (token) lifetime: Can be set by adjusting the days/hours/minutes fields. The default configuration will yield one hour.

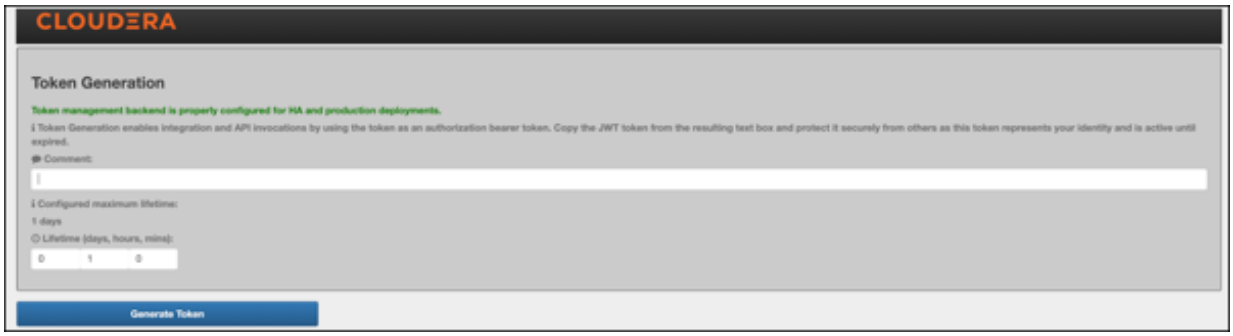

If Knox Token Integration - Enable Impersonation is set to true, another input field is shown on the UI called Generating token for (impersonation).

Using that input field our customers should be able to generate tokens on behalf of other users. For this to work, the Knox Token Integration - Proxyuser Block property has to be configured properly.

Important: If Knox is behind a Load Balancer and Token Impersonation support is used while generating tokens (that input field is populated with a username), the Load Balancer host must be added to the Proxy User configuration too. If the user wants to decline requests from a specific host, then that can be configured on the Load Balancer side.

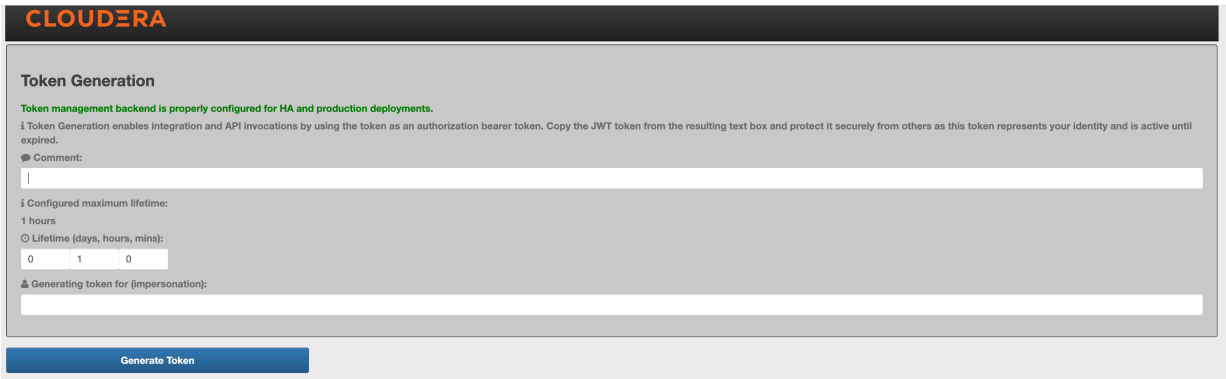

For more information, see Knox Apache User-guide: Token impersonation

3. Click Generate Token.

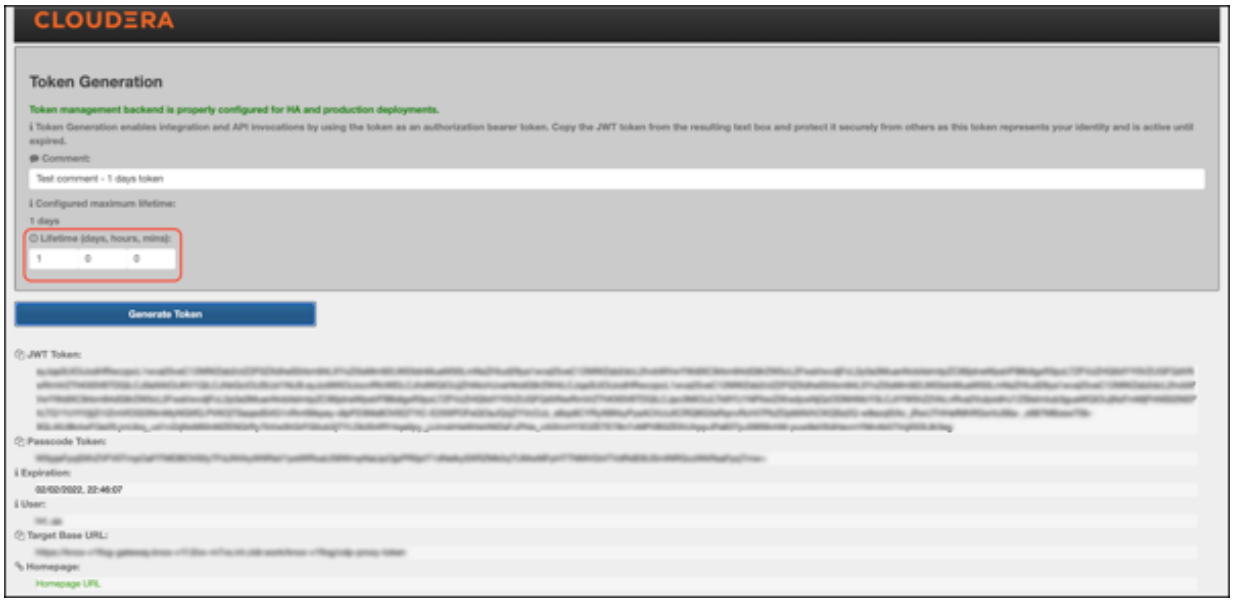

4. Use the token to authenticate your request. Click the icon beside your choice on the page to copy the value to the clipboard:

   - JWT token: serialized JWT, fully compatible with the old-style bearer authorization method. You can use it as the ‘Token’ user:

```
$ curl -ku Token:eyJqa3U[...]uT5AxQGyMMP3VLGw https:/localhost:8443/gate
way/cdp-proxy-token/webhdfs/v1?op=LISTSTATUS

{"FileStatuses":{"FileStatus":[{"accessTime":0,"blockSize":0,"childre
nNum":1,"fileId":16386,"group":"supergroup",
"length":0,"modificationTime":1621238405734,"owner":"hdfs","pathSuffix"
:"tmp","permission":"1777","replication":0,
"storagePolicy":0,"type":"DIRECTORY"},{"accessTime":0,"blockSize":0,"chi
ldrenNum":1,"fileId":16387,"group":"supergroup",
"length":0,"modificationTime":1621238326078,"owner":"hdfs","pathSuffix"
:"user","permission":"755","replication":0,
"storagePolicy":0,"type":"DIRECTORY"}]}}
```

   - Passcode token: Serialized passcode token, which can be used as the ‘Passcode’ user:

```
$ curl -ku Passcode:WkRFMk1XTmh[...]RVNFpXRTA= https://localhost:8443/ga
teway/cdp-proxy-token/webhdfs/v1?op=LISTSTATUS

{"FileStatuses":{"FileStatus":[{"accessTime":0,"blockSize":0,"childrenN
um":1,"fileId":16386,"group":"supergroup",
"length":0,"modificationTime":1621238405734,"owner":"hdfs","pathSuffi
x":"tmp","permission":"1777","replication":0,
"storagePolicy":0,"type":"DIRECTORY"},{"accessTime":0,"blockSize":0,"c
hildrenNum":1,"fileId":16387,"group":"supergroup",
"length":0,"modificationTime":1621238326078,"owner":"hdfs","pathSuffi
x":"user","permission":"755","replication":0,
"storagePolicy":0,"type":"DIRECTORY"}]}}
```

Manage Knox Gateway tokens

You can enable, disable, or revoke tokens via the Knox homepage.

Procedure

1. To access Knox token management, go to https://*KNOX_GATEWAY_HOST*:*PORT*/*GATEWAY_PATH*/homepage/home, e.g. https://localhost:8443/gateway/homepage/home. Click on Token Management.

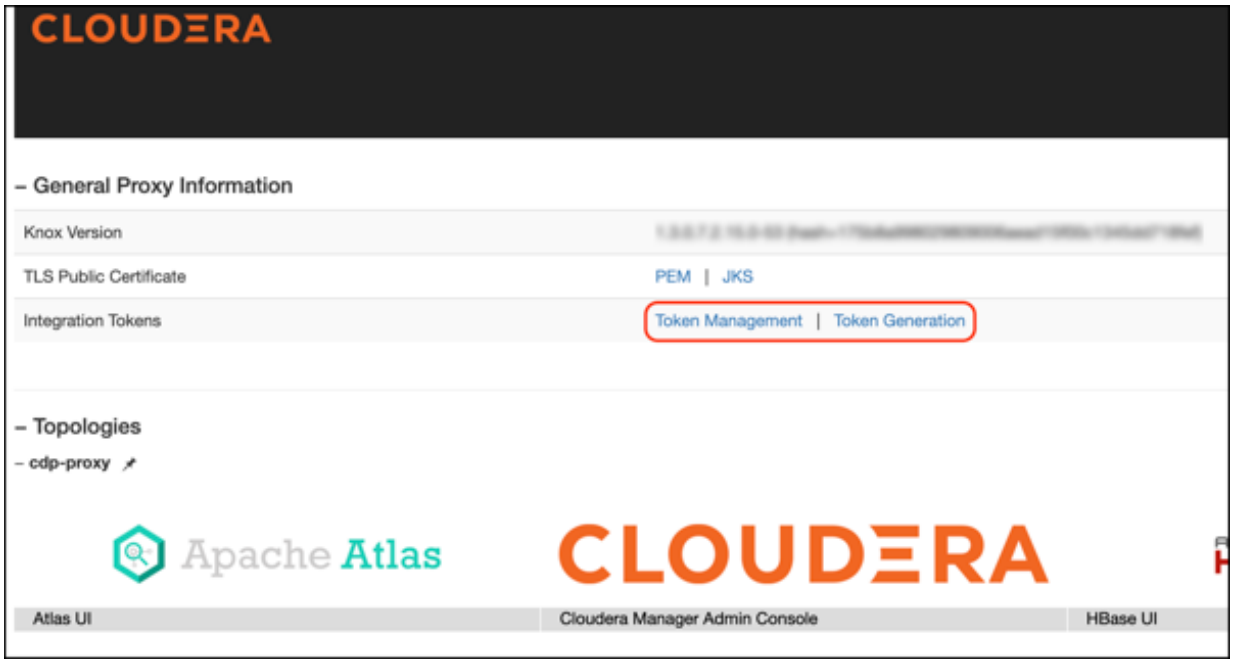

A compact view of all tokens generated within the system is shown in a single table with the following information.

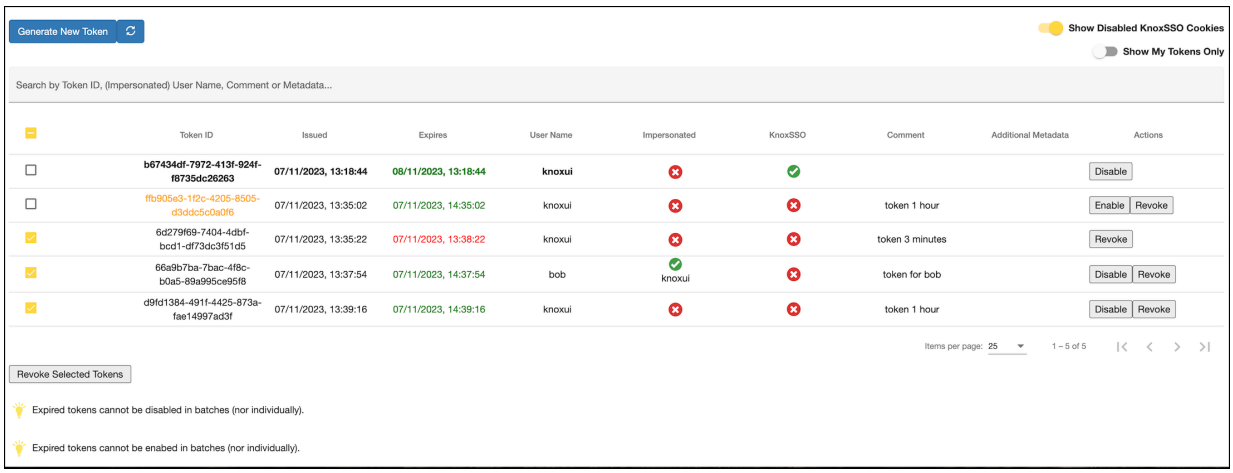

   a. Each row starts with a selection checkbox for batch operations (except for disabled KnoxSSO cookies, as there is no point in doing anything with them).
   b. A unique token identifier. Disabled token's Token ID value is shown in orange.
   c. Information on when the token was created and when it will expire.
      1. If the token is already expired, the expiration time is shown in red.
      2. If the token is still valid, the expiration time is shown in green.
   d. Username indicates the user for whom the token is created for.
   e. Impersonated is a boolean flag indicating if this is an impersonated token:
      - Green check: Yes, this is impersonated. You'll see the user who created the token under the icon.
      - Red cross: No, this is not an impersonated token.

f. KnoxSSO is another boolean flag that indicates if this token is created by the KNOXSSO service if the feature was enabled. The line is shown in bold if the token in this line represents the SSO cookie currently used as the authentication token to login to the Token Management page.
   - green check: yes, this is KnoxSSO cookie (token)
   - red cross: no, this is not a KnoxSSO cookie (it was created by a regular token API call or on the Token Generation page or the feature is disabled)

g. Comment: users may add a short comment to the tokens they create to make it easier for them to distinguish certain tokens later (e.g. "1-hour token for user XY")

h. Additional Metadata : In some cases, it's beneficial to add different metadata to the generated token as a key-value pair (e.g. shouldBeRemovedBy=09_Nov_2023). One token can have more than one associated metadata. In this column, we display that information.

i. In the Actions column, you will see
   - The enable/disable/revoke actions are visible for impersonated tokens too
   - KnoxSSO cookies cannot be revoked nor re-enabled.

In order to refresh the table, you can use the Refresh icon above the table (if you generated tokens on another tab for instance).

2. You can perform batch operations on the tokens. When at least one token is selected, the following buttons are shown under the table:
   - Disable - when executed, all the selected tokens become disabled (if they were disabled originally, they will remain disabled). Please note this option is shown only, if there is no expired token selected (i.e. batch disablement only works with live tokens).
   - Enable - when executed, all the selected tokens become enabled (if they were enabled originally, they will remain enabled). Please note this option is shown only, if there is no expired token selected (i.e. batch enablement only works with live tokens).
   - Revoke - when executed, all the selected tokens will be revoked. Please note this option is shown only, if there is no KnoxSSO cookie (token) selected (i.e. batch revocation only works with regular tokens).

     Note: If the selected tokens contain any that cannot be disabled/enabled (expired tokens) or revoked (Knox SSO Cookies), an informational message is displayed below the batch operation buttons indicating the root cause.
3. You can use the Search by field to narrow down tokens by :
   - Token ID
   - User Name (either own user name or impersonated)
   - Comment
   - Additional Metadata
4. You can view the disabled Knox cookies or only your tokens by using the following toggle buttons.
   - Show Disabled KnoxSSO Cookies - This is true by default. Since disabled KnoxSSO cookies remain in the underlying token state service until they expire, it may bother users to see them in the tokens table. Flipping this toggle button helps to hide them.
   - Show My Tokens Only - this toggle button is only visible to users, who can see all tokens. By default, this is false. Enabling it will filter the tokens table in a way such that it will contain tokens only that were generated for the logged-in user (impersonated or not).
5. Click the Refresh icon above the table.

Knox Token API

The Knox Token Service enables the ability for clients to acquire the same JSON Web Token (JWT) that is used for KnoxSSO with WebSSO flows for UIs to be used for accessing REST APIs.

Introduction

By acquiring the token and setting it as a Bearer token on a request, a client is able to access REST APIs that are protected with the JWTProvider federation provider.

In Cloudera, the only Knox topology that is properly configured for Knox Token management is the homepage topology, so that the following samples will assume KnoxSSO authentication. To use the Knox Token API, you should copy the hadoop-jwt cookie from your browser and export it as an ENV variable :

```
export HJWT="eyJraWQiOiJtSWdPbjRpSWZ1UmZ1RlVLWjVSb3dxbFh3SGUycGJUcm9lWjlEX1B
LUUJNIiwiYWxnIjoiUlMyNTYifQ.eyJzdWIiOiJrbm94dWkiLCJraWQiOiJtSWdPbjRpSWZ1UmZ1
RlVLWjVSb3dxbFh3SGUycGJUcm9lWjlEX1BLUUJNIiwiaXNzIjoiS05PWFNTTyIsImV4cCI6MTcw
MjU1MjE1OCwibWFuYWdlZC50b2tlbiI6ImZhbHNlIiwia25veC5pZCI6IjY5M2EwMmE3LTVhYTgt
NDA2MS1iYjMyLTA4MDk4YzdlMTkxYiJ9.tLzmxSd64bAQGpKdETsNXaaDBUKyMZzp0j0YNi-l4Jm
cm1oG5PerUt0OEmLWQnsDBuqtzExkR-g8metxwyIwjV61rqZRXLFycrN8x-nMTCExdxcjtMegIS3
XyETut8MRx8nk6WPVcBlwGHnOCG52CvxsvBe7pUFD4jYYbGzF_WlkPDzPjSRCdQ3xRFDq2IFt7Rx
OIye_50ZdMLbZBm9rNi0RErgdrLKJse68fly-58BcfquubFgWUA0Z0QND7Gg3lPBzyBOhe_5YA23
jQsicgvtc-HhNkY6W2RP-qpXmgjInGcy7dnpvbHXQNfA8cXffdQA6e3bFrTHpjNHgpEsGeg"
```

Alternatively, you can create a custom topology with another authentication provider (PAM, for instance) and use that topology instead of homepage in the following examples.

Important: KnoxToken API 'v1' is no longer supported in 7.2.18 and subsequent Cloudera releases. All the samples use the new 'v2' endpoints with the corresponding HTTP methods (GET, PUT, DELETE).

Acquire a token

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -X GET
 'https://[***HOST_NAME***][***DATALAKE_NAME***]/homepage/knoxtoken/api/v1/
token'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 11:15:16 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 3073{"access_token":"eyJqa3UiOiJo...4BnSw","token_id":"8f7d
e7f7-a094-4ba4-b64d-f21be86f10eb","managed":"true","target_url":"cdp-proxy-t
oken","homepage_url":"homepage/home?profile=token&topologies=cdp-proxy-token
","endpoint_public_cert":"MIIE1jCCAz...cXff0","token_type":"Bearer","expires
_in":1702469717045,"passcode":"T0dZM1pH...U1ESXo="}
```

In the result JSON, end-users can find the following information:

- accessToken: this is the serialized JWT and is fully compatible with the old-style Bearer authorization method. End-users might want to use it as the 'Token' user in the cdp-proxy-token topology for authentication purposes.
- passcode: this is another sensitive data, the serialized passcode token, which end-users can use as the 'Passcode' user for authentication purposes
- token_id: the unique identifier of the token within Knox
- managed: this is a boolean flag indicating if the token is managed. Managed tokens can be renewed, revoked, disabled, and enabled. By default, in Cloudera, the homepage topology is configured to manage tokens.
- expires: indicates the expiration time of this token. This may be updated with token renewal.

Renew a token

Currently, renewing a token is feasible only if you pass the value of the access_token field as the request payload. For instance:

```
$ export KNOX_TOKEN="eyJqa3UiOiJo...4BnSw"
```

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:valid" -d $KNOX
_TOKEN -X PUT 'https://[***HOST_NAME***][***DATALAKE_NAME***]/homepage/
knoxtoken/api/v2/token/renew'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:22:01 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: text/plain
Content-Length: 54

{
    "renewed": "true",
    "expires": "1702567321838"
}
```

The result JSON will tell you if the renewal was a success (this information is stored in the renewed field) and the (new) expiration time. In case of an invalid/unknown token, you should expect a response like this :

```
{
    "renewed": "false",
    "error": "Unknown token: 9caf743e-1e0d-4708-a9ac-a684a576067c"
}
```

Note: Token Renewal is allowed only for a certain set of users, which end-users can define using the Knox Token Integration - Renewer Whitelist Knox configuration on the CM UI.

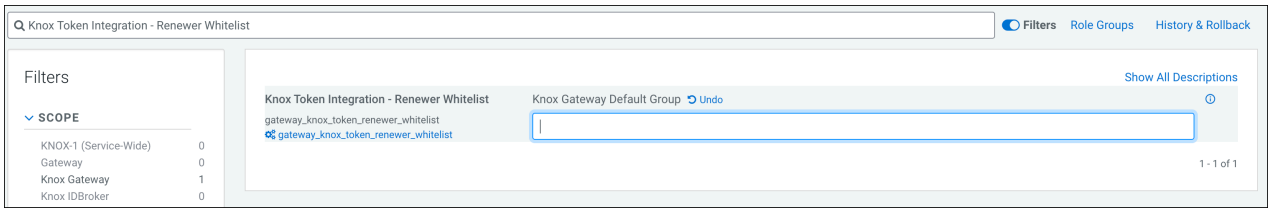

If the requesting user is an unauthorized caller, you should expect a response like this:

```
{
    "renewed": "false",
    "error": "Caller (myTestUser) not authorized to renew tokens."
}
```

Token revocation

End-users can revoke a token using either the token_id or the access_token fields from the above acquired token response. For instance, the following sample uses the token_id :

```
curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:v
alid" -d 'cb538d38-3076-4a7a-90a5-0f26bff2939a' -X DELETE
 'https://[***HOST_NAME***][***DATALAKE_NAME***]/homepage/knoxtoken/api/v2/
token/revoke'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:31:31 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 24

{
```

```
    "revoked": "true"
}
```

In case of an invalid or unknown token, you should expect a similar error like this:

```
{
    "revoked": "false",
    "error": "Unknown token: cb538d38...0f26bff2939a",
    "code": 50
}
```

Token revocation also requires authorization. The same Cloudera Manager configuration should be used that is listed above for token renewals. There is an exception though with revocation: end-users can revoke tokens that belong to them. That is, you can revoke your very own token even if your user name is not defined in the Knox Token Integration - Renewer Whitelist configuration.

Enable/Disable a Token

End-users might need to temporarily disable a token for security purposes and then re-enable the same token. You can do that with the following API calls which use the token_id field from the acquired token response:

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER
:valid" -d '1cad8a13-08e2-4b8a-8076-082678bb641b' -X PUT
 'https://[***HOST_NAME***][***DATALAKE_NAME***]/homepage/knoxtoken/api/v2/
token/disable'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:42:57 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 55


{
    "setEnabledFlag": "true",
    "isEnabled": "false"
}


$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER:
valid" -d '1cad8a13-08e2-4b8a-8076-082678bb641b' -X PUT
 'https://[***HOST_NAME***][***DATALAKE_NAME***]/homepage/knoxtoken/api/v2/
token/disable'
HTTP/1.1 400 Bad Request
Date: Wed, 13 Dec 2023 15:43:21 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 86


{
    "setEnabledFlag": "false",
    "error": "Token is already disabled",
    "code": 60
}

$ curl -ik --cookie "hadoop-jwt=$HJWT" -H "X-XSRF-HEADER
:valid" -d '1cad8a13-08e2-4b8a-8076-082678bb641b' -X PUT
```

```
 'https://[***HOST_NAME***][***DATALAKE_NAME***]/homepage/knoxtoken/api/v2/
token/enable'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:43:45 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 54


{
    "setEnabledFlag": "true",
    "isEnabled": "true"
}
```

Fetching user tokens

The KnoxToken API provides a powerful way to fetch/filter previously created tokens. See the following samples :

```
$ curl -ik --cookie "hadoop-jwt=$HJWT" -X GET
 'https://[***HOST_NAME***][***DATALAKE_NAME***]/homepage/knoxtoken/api/v2/
token/getUserTokens?userName=knoxui'
HTTP/1.1 200 OK
Date: Wed, 13 Dec 2023 15:46:50 GMT
X-Frame-Options: DENY
X-XSS-Protection: 1;mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 413


{"tokens":[{"tokenId":"1cad8a13-08e2-4b8a-8076-082678bb641b","issueTime":
"2023-12-13T15:42:17.965+0000","expiration":"2023-12-13T16:42:17.957+0000","
maxLifetime":"2023-12-20T15:42:17.965+0000","metadata":{"knoxSsoCookie":fals
e,"customMetadataMap":{},"createdBy":null,"comment":null,"enabled":true,"use
rName":"knoxui"},"maxLifetimeLong":1703086937965,"issueTimeLong":17024821379
65,"expirationLong":1702485737957}]}
```

Manage Knox metadata

This document describes how to manage Token Metadata.

As indicated in the previous sections, the KNOXTOKEN service maintains some hard-coded token metadata out-of-the-box:

- userName
- comment
- enabled
- passcode
- createdBy (in case of impersonated tokens)

In Cloudera Runtime version 7.2.16, Cloudera has introduced support for a new feature that allows end-users to add accept query parameters starting with the md_ prefix and treat them as Knox Token Metadata.

Example

```
curl -iku admin:admin-password -X GET 'https://$KNOX_GATEWAY_HOST:8443/gatew
ay/sandbox/knoxtoken/api/v1/token?md_notebookName=accountantKnoxToken&md_sou
ldBeRemovedBy=31March2022&md_otherMeaningfuMetadata=KnoxIsCool'
```

When such a token is created by Knox, the following metadata should be saved:

- notebookName=accountantKnoxToken
- shouldBeRemovedBy=31March2022
- otherMeaningfulMetadata=KnoxIsCool

It will not only enable Knox to save these metadata, but will also enable Knox's existing getUserTokens API endpoint to fetch basic token information using the supplied metadata name besides the username information.

Note: The getUserTokens API returns tokens if any of the supplied metadata exists for the given token. Metadata values may or may not be matched: you can either use the * wildcard to match all metadata values with a given name or you can further filter the stored metadata information by specifying the desired value.

Example:

```
curl -iku admin:admin-password -X GET 'https://$KNOX_GATEWAY_HOST:8443/gatew
ay/sandbox/knoxtoken/api/v1/token/getUserTokens?userName=admin&md_notebookNa
me=accountantKnoxToken&md_name=*'
```

It will return all Knox tokens where metadata with notebookName exists and equals accountantKnoxToken OR metadata with name exists.

Another Sample:

1. Create token1 with md_Name=reina&md_Score=50
2. Create token2 with md_Name=mary&md_Score=100
3. Create token3 with md_Name=mary&md_Score=20&md_Grade=A

The following table shows the returned token(s) in case metadata filtering is added in the getUserTokens API:

Metadata	Token returned
md_Name=reina	token1
md_Name=mary	token2 and token3
md_Score=100	token2
md_Name=mary&md_Score=20	token2 and token3
md_Name=mary&md_Name=reina	token1, token2 and token3
md_Name=*	token1, token2 and token3
md_Uknown=*	Empty list

For more information on sample curl commands, see Managing custom Knox Token metadata.

Knox SSO Cookie Invalidation

This feature allows a list of pre-configured superusers to invalidate previously issued Knox SSO tokens for (a) particular user(s) in case there is a malicious attack where one (or more) of those users' SSO tokens get compromised.

Enabling the feature

By default, the feature is disabled. There are 2 separate steps to enable it:

1. Go to Cloudera Manager Knox Configuration and enable Knox SSO - Cookie Management Enabled.

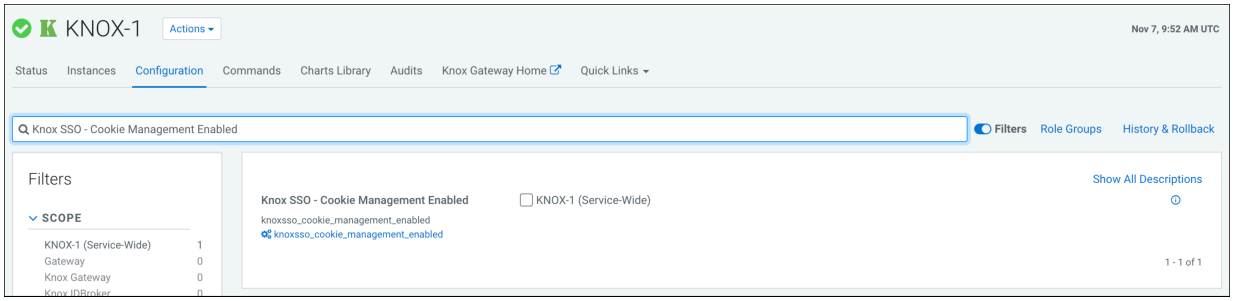

2. In Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml, press +.
   a. In Name, type gateway.knox.token.exp.server-managed.
   b. In Value, type true.
   c. Click Save Changes(CTRL+S)

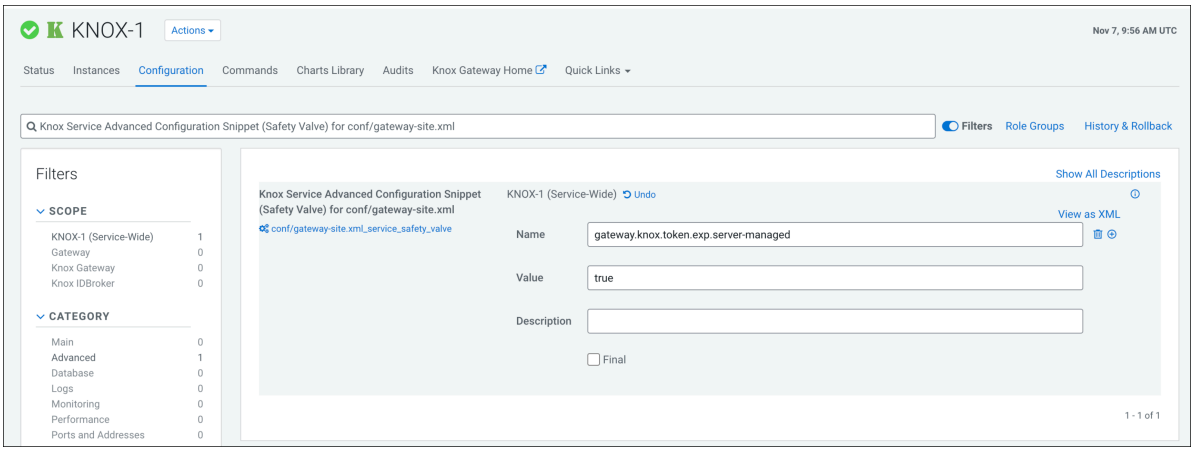

Additional configuration

In addition to enabling the feature, you should review and update the following configuration, if needed:

- Knox Home Page - Global Logout Page URL - when the knoxsso topology is configured to use the Pac4J federation filter (which is the default case in Cloudera Public Cloud), this configuration is an essential parameter (thus it must not be empty). This usually points to the logout endpoint of the pre-configured SAML/OIDC callback.

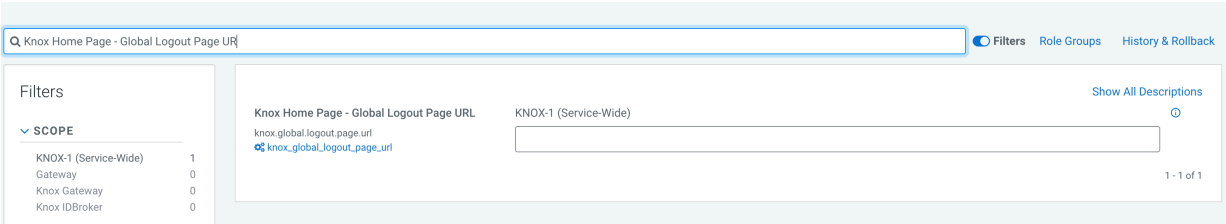

- Knox Token Integration - Users Who Can See All Tokens - A comma-separated list of user names who can see all tokens on the Token Management page. By default, this is an empty list. Each organization should configure this

property to a narrowed set of users, security officers for instance, who will have the capability of disabling SSO cookies in case of a security breach.

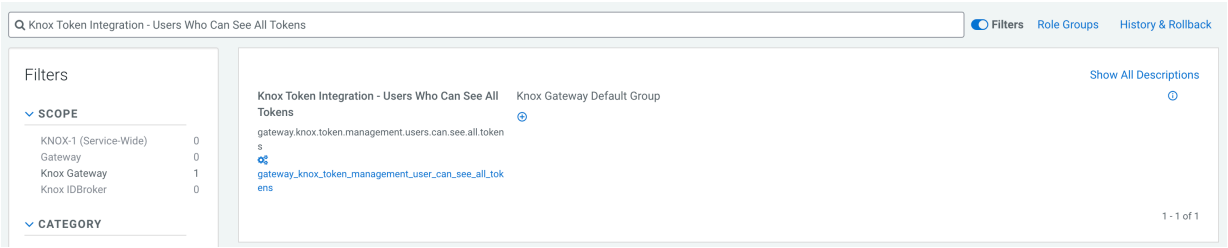

How it works

After enabling the feature, every SSO cookie, the result of a login event through the Knox SSO service, will be recorded in the same database that Knox uses for token management purposes. These SSO cookies are included on the Token Management page. If the logged-in user is a configured "superuser" (added in the above-referenced Users Who Can See All Tokens list), that user is capable of narrowing down user tokens, for whom they suspect are the subject of malicious activities, and disabling the active tokens on the UI (either individually or in batches).

Once a Knox SSO cookie is disabled, it cannot be re-enabled or revoked. Knox has its own cleanup strategy to remove expired tokens from the underlying token state repository (a database in Cloudera Public Cloud) periodically, on a pre-configured schedule.

It is also important to emphasize, that the default Time To Live (TTL) value of Knox SSO cookies is set to 1 day by default. It's highly recommended that organizations overview their own UI jobs and reduce this value to as short as possible to reduce the security risk involved here.

Concurrent session verification (Tech Preview)

This feature is a security measure that enables end-users limiting the number of concurrent UI sessions the users can have. To achieve this goal the users can be sorted out into three groups: non-privileged, privileged, unlimited.

The non-privileged and privileged groups each have a configurable limit, which the members of the group can not exceed. The members of the unlimited group are able to create an unlimited number of concurrent sessions.

All of the users, who are not configured in either the privileged or in the unlimited group, shall become the member of the non-privileged group by default.

Note: Conccurent session verification feature is under Technical Preview. The technical preview feature and considered under development. Do not use this in your production systems. To share your feedback, contact Support by logging a case on our Cloudera Support Portal. Technical preview features are not guaranteed troubleshooting guidance and fixes.

Configuration

The following table shows the relevant gateway-level parameters that are essential for this feature to work:

Parameter	Description	Default
gateway.service.concurrentsessionverifier.impl	To enable the session verification feature, end-users should set this parameter to org.apache.knox.gateway.session.control.InMemoryConcurrentSessionVerifier	org.apache.knox.gateway.session.contronVerifier
gateway.session.verification.privileged.users	Indicates a list of users that are qualified “privileged”.	Empty list
gateway.session.verification.unlimited.users	Indicates a list of (super) users that can have as many UI sessions as they want.	Empty list
gateway.session.verification.privileged.user.limit	The number of UI sessions a “privileged” user can have	3
gateway.session.verification.non.privileged.user.limit	The number of UI sessions a “non-privileged” user can have	2

How this works

If the verifier is disabled it will not do anything even if the other parameters are configured.

When the verifier is enabled all of the users are considered as a non-privileged user by default and they will not be able to create more concurrent sessions than the non-privileged limit. The same is true after you added someone in the privileged user group: that user will not be able to create more UI sessions than the configured privileged user limit. Whereas the members of the unlimited users group are able to create an unlimited number of concurrent sessions even if they are configured in the privileged group as well.

In Cloudera, currently, there are no first-class Cloudera Manager parameters for this feature, so all of those properties have to be set through Knox Service Advanced Configuration Snippet (Safety Valve) for conf/gateway-site.xml configuration in Cloudera Manager.

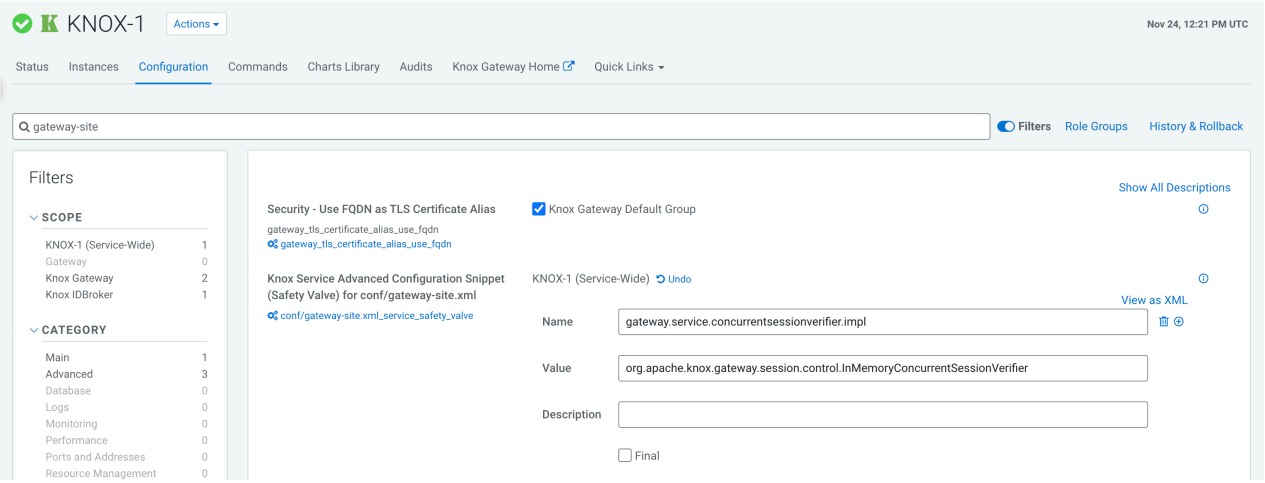

Apache Kudu Background Operations

Date published:
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Maintenance manager 4

Flushing data to disk 4

Compacting on-disk data 4

Write-ahead log garbage collection 4

Tablet history garbage collection and the ancient history mark 5

Maintenance manager

The maintenance manager schedules and runs background tasks. At any given point in time, the maintenance manager is prioritizing the next task based on improvements needed at that moment, such as relieving memory pressure, improving read performance, or freeing up disk space.

The number of worker threads dedicated to running background tasks can be controlled by setting --maintenance_manager_num_threads.

The maintenance work is now scheduled immediately whenever any configured thread is available. Make sure that the --maintenance_manager_num_threads property is set to at most a 1:3 ratio for Maintenance Manager threads to the number of data directories (for spinning disks). This will improve the throughput of write-heavy workloads.

Flushing data to disk

Flushing data from memory to disk relieves memory pressure and can improve read performance by switching from a write-optimized, row-oriented in-memory format in the MemRowSet, to a read-optimized, column-oriented format on disk.

Background tasks that flush data include FlushMRSOp and FlushDeltaMemStoresOp. The metrics associated with these operations have the prefix flush_mrs and flush_dms, respectively.

With Kudu 1.4, the maintenance manager aggressively schedules flushes of in-memory data when memory consumption crosses 60 percent of the configured process-wide memory limit. The backpressure mechanism which begins to throttle client writes was also adjusted to not begin throttling until memory consumption reaches 80 percent of the configured limit. These two changes together result in improved write throughput, more consistent latency, and fewer timeouts due to memory exhaustion.

Compacting on-disk data

Kudu constantly performs several compaction tasks in order to maintain consistent read and write performance over time.

- A merging compaction, which combines multiple DiskRowSets together into a single DiskRowSet, is run by CompactRowSetsOp.
- Kudu also runs two types of delta store compaction operations: MinorDeltaCompactionOp and MajorDeltaCompactionOp.

  For more information on what these compaction operations do, see the Kudu Tablet design document.

The metrics associated with these tasks have the prefix compact_rs, delta_minor_compact_rs, and delta_major_compact_rs, respectively.

Write-ahead log garbage collection

Kudu maintains a write-ahead log per tablet that is split into discrete fixed-size segments. A tablet periodically rolls the WAL to a new log segment when the active segment reaches a size threshold (configured by the --log_segment_size_mb property).

In order to save disk space and decrease startup time, a background task called LogGCOp attempts to garbage-collect old write-ahead log segments by deleting them from disk once it is determined that they are no longer needed by the local node for durability.

The metrics associated with this background task have the prefix log_gc.

Tablet history garbage collection and the ancient history mark

Kudu uses a multiversion concurrency control (MVCC) mechanism to ensure that snapshot scans can proceed isolated from new changes to a table. Therefore, periodically, old historical data should be garbage-collected (removed) to free up disk space. While Kudu never removes rows or data that are visible in the latest version of the data, Kudu does remove records of old changes that are no longer visible.

The specific threshold in time (in the past) beyond which historical MVCC data becomes inaccessible and is free to be deleted is called the ancient history mark (AHM). The AHM can be configured by setting the --tablet_history_max_age_sec property.

There are two background tasks that remove historical MVCC data older than the AHM:

- The one that runs the merging compaction, called CompactRowSetsOp (see above).
- A separate background task deletes old undo delta blocks, called UndoDeltaBlockGCOp. Running UndoDeltaBlockGCOp reduces disk space usage in all workloads, but particularly in those with a higher volume of updates or upserts. The metrics associated with this background task have the prefix undo_delta_block.

Configuring Apache Kudu

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configure Kudu processes **4**
Experimental flags 4

Configuring the Kudu master **5**

Configuring tablet servers **5**

Rack awareness (Location awareness) **6**

Directory configurations **7**
Changing directory configuration 7

Configure Kudu processes

You can pass command-line flags when you start a Kudu process to configure its behavior or read those options from configuration files by passing them using one or more --flagfile=<file> options.

You can include the --flagfile option within your configuration file to include other files. Learn more about gflags by reading its documentation.

You can place options for masters and tablet servers in the same configuration file, and each will ignore options that do not apply.

Flags can be prefixed with either one or two - characters. This documentation standardizes on two: --example_flag.

Only the most common configuration options are documented in this topic. For a more exhaustive list of configuration options, see the Kudu Configuration Reference. To see all configuration flags for a given executable, run it with the --help option.

Experimental flags

Some configuration flags are marked 'unsafe' and 'experimental'. Such flags are disabled by default.
You can access these flags by enabling the additional flags, `--unlock_unsafe_flags` and `--unlock_experimental_flags`.

Note that these flags might be removed or modified without a deprecation period or any prior notice in future Kudu releases. Cloudera does not support using unsafe and experimental flags. As a rule of thumb, Cloudera will not support any configuration flags not explicitly documented in the Kudu Configuration Reference Guide.

Some particularly impactful unsafe Kudu configuration options are:

- --use-hybrid-clock

  Setting this flag to false is highly discouraged in any production-grade deployment, as it can introduce extensive latency and behavior such as not operating in COMMIT_WAIT consistency mode, failing to handle scan operations in READ_AT_SNAPSHOT and READ_YOUR_WRITES modes, and so on.
- --allow_world_readable_credentials

  Do not set this flag to true, but rather set the ownership and file access mask properly using chown and chmod commands.
- --allow_unsafe_replication_factor

  Do not set this flag to true since even replication factors of (2 * N) is not any better than (2 * N - 1) because Kudu uses Raft consensus protocol.
- --max_cell_size_bytes

  If you need to increase it, you must consult with Cloudera to evaluate possible performance degradation and other issues depending on specific workload.
- --max_num_columns

  If you need to adjust it, you must consult with Cloudera to evaluate possible performance degradation and other issues depending on specific workload.
- --rpc_num_service_threads

  You should never set it higher than the number of CPU cores at a node.

Configuring the Kudu master

To see all available configuration options for the kudu-master executable, run it with the --help option:

```
kudu-master --help
```

Table 1: Supported configuration flags for Kudu masters

Flag	Valid options	Default	Description
--master_addresses	string	localhost	Comma-separated list of all the RPC addresses for Master consensus-configuration. If not specified, assumes a standalone Master.
--fs_data_dirs	string		List of directories where the Master will place its data blocks.
--fs_wal_dir	string		The directory where the Master will place its write-ahead logs.
--log_dir	string	/tmp	The directory to store Master log files.

For the complete list of flags for masters, see the Kudu Master Configuration Reference.

Configuring tablet servers

To see all available configuration options for the kudu-tserver executable, run it with the --help option:

```
kudu-tserver --help
```

Table 2: Supported configuration flags for Kudu tablet servers

Flag	Valid options	Default	Description
--fs_data_dirs	string		List of directories where the Tablet Server will place its data blocks.
--fs_wal_dir	string		The directory where the Tablet Server will place its write-ahead logs.
--log_dir	string	/tmp	The directory to store Tablet Server log files
--tserver_master_addrs	string	127.0.0.1:7051	Comma separated addresses of the masters that the tablet server should connect to. The masters do not read this flag.
--block_cache_capacity_mb	integer	512	Maximum amount of memory allocated to the Kudu Tablet Server's block cache.

Flag	Valid options	Default	Description
--memory_limit_hard_bytes	integer	4294967296	Maximum amount of memory a Tablet Server can consume before it starts rejecting all incoming writes.

For the complete list of flags for tablet servers, see the Kudu Tablet Server Configuration Reference.

Rack awareness (Location awareness)

Kudu's rack awareness feature provides protection from correlated failures. The rack awareness feature has two elements: location assignment and placement policy.

Kudu supports a rack awareness feature. Kudu’s ordinary re-replication methods ensure the availability of the cluster in the event of a single node failure. However, clusters can be vulnerable to correlated failures of multiple nodes. For example, all of the physical hosts on the same rack in a datacenter may become unavailable simultaneously if the top-of-rack switch fails. Kudu’s rack awareness feature provides protection from certain kinds of correlated failures, such as the failure of a single rack in a datacenter.

Location assignment

The first element of Kudu’s rack awareness feature is location assignment. When a tablet server registers with a master, the master assigns it a location. A location is a /-separated string that begins with a / and where each /-separated component consists of characters from the set [a-zA-Z0-9_-.].

For example, /dc-0/rack-09 is a valid location, while rack-04 and /rack=1 are not valid locations. Thus location strings resemble absolute UNIX file paths where characters in directory and file names are restricted to the set [a-zA-Z0-9_-.].

Kudu does not use the hierarchical structure of locations. Location assignment is done by a user-provided command, whose path should be specified using the --location_mapping_cmd master flag. The command should take a single argument, the IP address or hostname of a tablet server, and return the location for the tablet server. Make sure that all Kudu masters are using the same location mapping command.

Placement policy

The second element of Kudu’s rack awareness feature is the placement policy: Do not place a majority of replicas of a tablet on tablet servers in the same location.

The leader master, when placing newly created replicas on tablet servers and when re-replicating existing tablets, will attempt to place the replicas in a way that complies with the placement policy.

For example, in a cluster with five tablet servers A, B, C, D, and E, with respective locations /L0, /L0, /L1, /L1, /L2, to comply with the placement policy a new 3x replicated tablet could have its replicas placed on A, C, and E, but not on A, B, and C, because then the tablet would have 2/3 replicas in location /L0.

As another example, if a tablet has replicas on tablet servers A, C, and E, and then C fails, the replacement replica must be placed on D in order to comply with the placement policy.

In the case where it is impossible to place replicas in a way that complies with the placement policy, Kudu will violate the policy and place a replica anyway. For example, using the setup described in the previous paragraph, if a tablet has replicas on tablet servers A, C, and E, and then E fails, Kudu will re-replicate the tablet onto one of B or D, violating the placement policy, rather than leaving the tablet under-replicated indefinitely.

The kudu cluster rebalance tool can reestablish the placement policy if it is possible to do so. The kudu cluster rebalance tool can also be used to reimpose the placement policy on a cluster if the cluster has just been configured to use the rack awareness feature and existing replicas need to be moved to comply with the placement policy. See *Run a tablet rebalancing tool on a rack-aware cluster* for more information.

Related Information

Running a tablet rebalancing tool on a rack-aware cluster

Directory configurations

Every Kudu node requires the specification of directory flags.

The --fs_wal_dir configuration indicates where Kudu will place its write-ahead logs.

The --fs_metadata_dir configuration indicates where Kudu will place metadata for each tablet. It is recommended, although not necessary, that these directories be placed on a high-performance drives with high bandwidth and low latency, e.g. solid-state drives. If --fs_metadata_dir is not specified, metadata will be placed in the directory specified by --fs_wal_dir.

Since a Kudu node cannot tolerate the loss of its write-ahead log or metadata directories, you might want to mirror the drives containing these directories in order to make recovering from a drive failure easier. However, mirroring may increase the latency of Kudu writes.

The --fs_data_dirs configuration indicates where Kudu will write its data blocks. This is a comma-separated list of directories; if multiple values are specified, data will be striped across the directories. If not specified, data blocks will be placed in the directory specified by --fs_wal_dir.

Note: While a single data directory backed by a RAID-0 array will outperform a single data directory backed by a single storage device, it is better to let Kudu manage its own striping over multiple devices rather than delegating the striping to a RAID-0 array.

Additionally, --fs_wal_dir and --fs_metadata_dir may be the same as one of the directories listed in --fs_data_dirs, but must not be sub-directories of any of them.

Warning: Each directory specified by a configuration flag on a given machine should be used by at most one Kudu process. If multiple Kudu processes on the same machine are configured to use the same directory, Kudu may refuse to start up.

Warning: Once --fs_data_dirs is set, extra tooling is required to change it.

Note: The --fs_wal_dir and --fs_metadata_dir configurations can be changed, provided the contents of the directories are also moved to match the flags.

Changing directory configuration

For higher read parallelism and larger volumes of storage per server, you can configure servers to store data in multiple directories on different devices using the --fs_data_dirs Gflag configuration.

About this task

You can add or remove data directories to an existing master or tablet server by updating the --fs_data_dirs Gflag configuration and restarting the server. Data is striped across data directories, and when a new data directory is added, new data will be striped across the union of the old and new directories.

Note:

Removing a data directory from --fs_data_dirs may result in failed tablet replicas in cases where there were data blocks in the directory that was removed. Use ksck to ensure the cluster can fully recover from the directory removal before moving onto another server.

In versions of Kudu below 1.12, Kudu requires that the kudu fs update_dirs tool be run before restarting with a different set of data directories. Such versions will fail to start if not run.

If on a Kudu version below 1.12, once a server is started, users must go through the below steps to change the directory configuration:

Note: Unless the --force flag is specified, Kudu will not allow for the removal of a directory across which tablets are configured to spread data. If --force is specified, all tablets configured to use that directory will fail upon starting up and be replicated elsewhere.

Note: If the metadata directory overlaps with a data directory, as was the default prior to Kudu 1.7, or if a non-default metadata directory is configured, the --fs_metadata_dir configuration must be specified when running the kudu fs update_dirs tool.

Note: Only new tablet replicas, i.e. brand new tablets' replicas and replicas that are copied to the server for high availability, will use the new directory. Existing tablet replicas on the server will not be rebalanced across the new directory.

Attention: All of the command line steps below should be executed as the Kudu UNIX user, typically kudu.

Procedure

1. Use ksck to ensure the cluster is healthy, and establish a maintenance window to bring the tablet server offline.
2. The tool can only run while the server is offline, so establish a maintenance window to update the server. The tool itself runs quickly, so this offline window should be brief, and as such, only the server to update needs to be offline.

   However, if the server is offline for too long (see the follower_unavailable_considered_failed_sec flag), the tablet replicas on it may be evicted from their Raft groups. To avoid this, it may be desirable to bring the entire cluster offline while performing the update.
3. Run the tool with the desired directory configuration flags. For example, if a cluster was set up with --fs_wal_dir =/wals, ##fs_metadata_dir=/meta, and ##fs_data_dirs=/data/1,/data/2,/data/3, and /data/3 is to be removed (e.g. due to a disk error), run the command:

   ```
   $ sudo -u kudu kudu fs update_dirs --force --fs_wal_dir=/wals --fs_metad
   ata_dir=/meta --fs_data_dirs=/data/1,/data/2
   ```
4. Modify the value of the --fs_data_dirs flag for the updated sever. If using Cloudera Manager, make sure to only update the configurations of the updated server, rather than of the entire Kudu service.
5. Once complete, the server process can be started. When Kudu is installed using system packages, service is typically used:

   ```
   $ sudo service kudu-tserver start
   ```
6. Use ksck to ensure Kudu returns to a healthy state before resuming normal operation.

Developing Applications with Apache Kudu

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

View the API documentation 4

Kudu example applications 4

Maven artifacts 5

Kudu Python client 5

Kudu integration with Spark 7

Spark integration known issues and limitations 9
Spark integration best practices 9
Upsert option in Kudu Spark 9
Use Spark with a secure Kudu cluster 10
Spark tuning 10

View the API documentation

This topic provides you information on how to find the API documentation for C++ and Java.

Warning: Use of server-side or private interfaces is not supported, and interfaces which are not part of public APIs have no stability guarantees.

C++ API Documentation

The documentation for the C++ client APIs is included in the header files in /usr/include/kudu/ if you installed Kudu using packages or subdirectories of src/kudu/client/ if you built Kudu from source. If you installed Kudu using parcels, no headers are included in your installation. and you will need to build Kudu from source in order to have access to the headers and shared libraries.

The following command is a naive approach to finding relevant header files. Use of any APIs other than the client APIs is unsupported.

```
find /usr/include/kudu -type f -name *.h
```

Java API Documentation

View the *Java API documentation* online. Alternatively, after building the Java client, Java API documentation is available in java/kudu-client/target/apidocs/index.html.

Related Information

Java API documentation

Kudu example applications

Several example applications are provided in the examples directory of the Apache Kudu git repository. Each example includes a README that shows how to compile and run it. The following list includes some of the examples that are available today. Check the repository itself in case this list goes out of date.

cpp/example.cc:

A simple C++ application which connects to a Kudu instance, creates a table, writes data to it, then drops the table.

java/java-example:

A simple Java application which connects to a Kudu instance, creates a table, writes data to it, then drops the table.

java/insert-loadgen:

A small Java application which listens on a TCP socket for time series data corresponding to the Collectl wire protocol. The commonly-available collectl tool can be used to send example data to the server.

python/dstat-kudu:

A Java application that generates random insert load.

python/graphite-kudu:

An example program that shows how to use the Kudu Python API to load data into a new / existing Kudu table generated by an external program, dstat in this case.

python/graphite-kudu:

An example plugin for using graphite-web with Kudu as a backend.

These examples should serve as helpful starting points for your own Kudu applications and integrations.

Related Information

Kudu examples

Maven artifacts

The following Maven <dependency> element is valid for the Apache Kudu GA release:

```
<dependency>
  <groupId>org.apache.kudu</groupId>
  <artifactId>kudu-client</artifactId>
  <version><kudu-cdp-version></version>
</dependency>
```

For <kudu-cdp-version>, check *Cloudera Runtime component versions* in *Release Notes*.

Convenience binary artifacts for the Java client and various Java integrations (e.g. Spark, Flume) are also now available via the *ASF Maven repository* and the *Central Maven repository*.

Related Information

ASF Maven repository

Central Maven repository

Kudu Python client

The Kudu Python client provides a Python friendly interface to the C++ client API. To install and use the Kudu Python client, you need to install the Kudu C++ client libraries and headers.

Before you begin

See Install Using Packages topic for installing the Kudu C++ client libraries.

Procedure

1. Update all the packages on your system by using the following command:

   yum -y update
2. Install the extra packages for the Enterprise Linux distribution:

   sudo yum -y install epel-release
3. Install the Python package manager:

   sudo yum -y install python-pip
4. Verify the version of the PIP manager that you just installed:

   pip --version
5. Install Cython:

   sudo pip install cython

6. Download the following files using wget:
   - Kudu artifact: http://username:password@archive.cloudera.com/p/cdp-public/*RUNTIME_VERSION*/redhat7/yum/kudu/*KUDU_ARTIFACT*
   - Kudu-client artifact: http://username:password@archive.cloudera.com/p/cdp-public/*RUNTIME_VERSION*/redhat7/yum/kudu/*KUDU-CLIENT_ARTIFACT*
   - Kudu-client-devel artifact: http://username:password@archive.cloudera.com/p/cdp-public/*RUNTIME_VERSION*/redhat7/yum/kudu/*KUDU-CLIENT-DEVEL_ARTIFACT*
7. Install the kudu package from the local directory:

   sudo yum -y localinstall ./kudu-*
8. Install the package used for developing Python extensions:

   sudo yum -y install python-devel
9. Upgrade the setup tools:

   sudo pip install --upgrade pip setuptools
10. Install the C++ compiler:

    sudo yum -y install gcc-c++
11. Setup the Python package version:

    export PACKAGE_VERSION=<kudu-python-cloudera-version>

    For the 7.1.9 SP1 release, use the export PACKAGE_VERSION=7.1.9.1000 command.
12. Install the Kudu-Python client:

    sudo pip install kudu-python-cloudera==$PACKAGE_VERSION
13. The following sample demonstrates the use of part of the Python client:

```
import kudu
from kudu.client import Partitioning
from datetime import datetime

# Connect to Kudu master server
client = kudu.connect(host='kudu.master', port=7051)

# Define a schema for a new table
builder = kudu.schema_builder()
builder.add_column('key').type(kudu.int64).nullable(False).primary_key()
builder.add_column('ts_val', type_=kudu.unixtime_micros, nullable=False,
 compression='lz4')
schema = builder.build()

# Define partitioning schema
partitioning = Partitioning().add_hash_partitions(column_names=['key'],
num_buckets=3)

# Create new table
client.create_table('python-example', schema, partitioning)

# Open a table
table = client.table('python-example')

# Create a new session so that we can apply write operations
session = client.new_session()

# Insert a row
op = table.new_insert({'key': 1, 'ts_val': datetime.utcnow()})
session.apply(op)
# Upsert a row
op = table.new_upsert({'key': 2, 'ts_val': "2016-01-01T00:00:00.000000"})
session.apply(op)
# Updating a row
```

```
op = table.new_update({'key': 1, 'ts_val': ("2017-01-01", "%Y-%m-%d")})
session.apply(op)
# Delete a row
op = table.new_delete({'key': 2})
session.apply(op)
# Flush write operations, if failures occur, capture print them.
try:
    session.flush()
except kudu.KuduBadStatus as e:
    print(session.get_pending_errors())

# Create a scanner and add a predicate
scanner = table.scanner()
scanner.add_predicate(table['ts_val'] == datetime(2017, 1, 1))
# Open Scanner and read all tuples
# Note: This doesn't scale for large scans
result = scanner.open().read_all_tuples()
```

Kudu integration with Spark

Kudu integrates with Spark through the Data Source API as of version 1.0.0. Include the kudu-spark dependency using the --packages or --jars option.

Note that Spark 1 is no longer supported in Kudu starting from version 1.6.0. So in order to use Spark 1 integrated with Kudu, version 1.5.0 is the latest to go to.

Use kudu-spark3_2.12 artifact if using Spark 3 with Scala 2.12.
For --packages option:

```
spark3-shell --packages org.apache.kudu:kudu-spark3_2.12:<kudu-c
dp-version> --repositories https://repository.cloudera.com/artif
actory/cloudera-repos/
```

For <kudu-cdp-version>, check *Cloudera Runtime component versions* in *Release Notes*.

For --jars option:

```
spark-shell --jars /opt/cloudera/parcels/CDH/lib/kudu/kudu-spark
3_2.12.jar
```

Below is a minimal Spark SQL "select" example for a Kudu table created with Impala in the "default" database. You first import the kudu spark package, then create a DataFrame, and then create a view from the DataFrame. After those steps, the table is accessible from Spark SQL. You can also refer to the Spark quickstart guide or this Kudu-Spark example.

Note: You can use the Kudu CLI tool to create table and generate data by kudu perf loadgen kudu.master:7051 -keep_auto_table for the following two examples:

```
import org.apache.kudu.spark.kudu._
// Create a DataFrame that points to the Kudu table we want to query.
val df = spark.read.options(Map("kudu.master" -> "kudu.master:7051",
                                "kudu.table" -> "default.my_table")).format(
"kudu").load
// Create a view from the DataFrame to make it accessible from Spark SQL.
df.createOrReplaceTempView("my_table")
// Now we can run Spark SQL queries against our view of the Kudu table.
spark.sql("select * from my_table").show()
```

Below is a more sophisticated example that includes both reads and writes:

```
import org.apache.kudu.client._
import org.apache.kudu.spark.kudu.KuduContext
import collection.JavaConverters._

// Read a table from Kudu
val df = spark.read
  .options(Map("kudu.master" -> "kudu.master:7051", "kudu.table" -> "kudu_ta
ble"))
  .format("kudu").load

// Query using the Spark API...
df.select("key").filter("key >= 5").show()

// ...or register a temporary table and use SQL
df.createOrReplaceTempView("kudu_table")
val filteredDF = spark.sql("select key from kudu_table where key >= 5").show
()

// Use KuduContext to create, delete, or write to Kudu tables
val kuduContext = new KuduContext("kudu.master:7051", spark.sparkContext)
// Create a new Kudu table from a DataFrame schema
// NB: No rows from the DataFrame are inserted into the table
kuduContext.createTable(
    "test_table", df.schema, Seq("key"),
    new CreateTableOptions()
        .setNumReplicas(1)
        .addHashPartitions(List("key").asJava, 3))

// Check for the existence of a Kudu table
kuduContext.tableExists("test_table")

// Insert data
kuduContext.insertRows(df, "test_table")

// Delete data
kuduContext.deleteRows(df, "test_table")

// Upsert data
kuduContext.upsertRows(df, "test_table")
// Update data
val updateDF = df.select($"key", ($"int_val" + 1).as("int_val"))
kuduContext.updateRows(updateDF, "test_table")
// Data can also be inserted into the Kudu table using the data source, tho
ugh the methods on
// KuduContext are preferred
// NB: The default is to upsert rows; to perform standard inserts instead, s
et operation = insert
// in the options map
// NB: Only mode Append is supported
df.write
  .options(Map("kudu.master"-> "kudu.master:7051", "kudu.table"-> "test_ta
ble"))
  .mode("append")
  .format("kudu").save
// Delete a Kudu table
kuduContext.deleteTable("test_table")
```

Spark integration known issues and limitations

Here are the limitations that you should consider while integrating Kudu and Spark.

- Spark 2.2 (and higher) requires Java 8 at runtime even though Kudu Spark 2.x integration is Java 7 compatible. Spark 2.2 is the default dependency version as of Kudu 1.5.0.
- Kudu tables with a name containing upper case or non-ASCII characters must be assigned an alternate name when registered as a temporary table.
- Kudu tables with a column name containing upper case or non-ASCII characters must not be used with SparkSQL. Columns can be renamed in Kudu to work around this issue.
- <> and ORpredicates are not pushed to Kudu, and instead will be evaluated by the Spark task. Only LIKE predicates with a suffix wildcard are pushed to Kudu. This means LIKE "FOO%" will be pushed, but LIKE "FOO%BAR" won't.
- Kudu does not support all the types supported by Spark SQL. For example, Date and complex types are not supported.
- Kudu tables can only be registered as temporary tables in SparkSQL.
- Kudu tables cannot be queried using HiveContext.

Spark integration best practices

It is best to avoid multiple Kudu clients per cluster.

A common Kudu-Spark coding error is instantiating extra KuduClient objects. In kudu-spark, a KuduClient is owned by the KuduContext. Spark application code should not create another KuduClient connecting to the same cluster. Instead, application code should use the KuduContext to access a KuduClient using KuduContext#syncClient.

To diagnose multiple KuduClient instances in a Spark job, look for signs in the logs of the master being overloaded by many GetTableLocations or GetTabletLocations requests coming from different clients, usually around the same time. This symptom is especially likely in Spark Streaming code, where creating a KuduClient per task will result in periodic waves of master requests from new clients.

Upsert option in Kudu Spark

The upsert operation in kudu-spark supports an extra write option of ignoreNull. If set to true, it will avoid setting existing column values in Kudu table to Null if the corresponding DataFrame column values are Null. If unspecified, ignoreNull is false by default.

```
val dataFrame = spark.read
  .options(Map("kudu.master" -> "kudu.master:7051", "kudu.table" -> simpl
eTableName))
  .format("kudu").load
dataFrame.createOrReplaceTempView(simpleTableName)
dataFrame.show()

// Below is the original data in the table 'simpleTableName'
+---+---+
|key|val|
+---+---+
|  0|foo|
+---+---+

// Upsert a row with existing key 0 and val Null with ignoreNull set to true
val nullDF = spark.createDataFrame(Seq((0, null.asInstanceOf[String]))).t
oDF("key", "val")
val wo = new KuduWriteOptions
wo.ignoreNull = true
kuduContext.upsertRows(nullDF, simpleTableName, wo)
```

```
dataFrame.show()
// The val field stays unchanged
+---+---+
|key|val|
+---+---+
|  0|foo|
+---+---+

// Upsert a row with existing key 0 and val Null with ignoreNull default/set
 to false
kuduContext.upsertRows(nullDF, simpleTableName)
// Equivalent to:
// val wo = new KuduWriteOptions
// wo.ignoreNull = false
// kuduContext.upsertRows(nullDF, simpleTableName, wo)
df.show()
// The val field is set to Null this time
+---+----+
|key| val|
+---+----+
|  0|null|
+---+----+
```

Use Spark with a secure Kudu cluster

The Kudu-Spark integration is able to operate on secure Kudu clusters which have authentication and encryption enabled, but the submitter of the Spark job must provide the proper credentials. For Spark jobs using the default 'client' deploy mode, the submitting user must have an active Kerberos ticket granted through kinit. For Spark jobs using the 'cluster' deploy mode, a Kerberos principal name and keytab location must be provided through the --principal and --keytab arguments to spark2-submit.

Spark tuning

In general the Spark jobs were designed to run with minimal tuning and configuration. You can adjust the number of executors and resources to increase parallelism and performance using Spark's configuration options.

If your tables are super wide and your default memory allocation is fairly low, you may see jobs fail. To resolve this, increase the Spark executor memory. A conservative rule of thumb is 1 GiB per 50 columns.

If your Spark resources drastically outscale the Kudu cluster, then you may want to limit the number of concurrent tasks allowed to run on restore.

Using the Hive Metastore with Apache Kudu

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Integrating the Hive Metastore with Apache Kudu.....4

Databases and Table Names.....4

Administrative tools for Hive Metastore integration.....5

Upgrading existing Kudu tables for Hive Metastore integration.....6

Enabling the Hive Metastore integration.....7

Integrating the Hive Metastore with Apache Kudu

You can integrate Kudu's own catalog with the Hive Metastore (HMS) allowing external HMS-aware tool to discover and use Kudu tables.

The HMS is the de-facto standard catalog and metadata provider in the Hadoop ecosystem. When the HMS integration is enabled, Kudu tables can be discovered and used by external HMS-aware tools, even if they are not otherwise aware of, or integrated with Kudu. There is no need to create Impala external tables to map any Kudu tables created directly using API or from Spark. Additionally, these components can use the HMS to discover necessary information to connect to the Kudu cluster which owns the table, such as the Kudu master addresses.

Databases and Table Names

When Hive Matastore (HMS) integration is enabled in Kudu, both the namespace and table names properties change to match the HMS model.

With the Hive Metastore integration disabled, Kudu presents tables as a single flat namespace, with no hierarchy or concept of a database. Additionally, Kudu's only restriction on table names is that they be a valid UTF-8 encoded string. When the HMS integration is enabled in Kudu, both of these properties change in order to match the HMS model: the table name must indicate the table's membership of a Hive database, and table name identifiers (i.e. the table name and database name) are subject to the Hive table name identifier constraints.

Databases

Hive has the concept of a database, which is a collection of individual tables. Each database forms its own independent namespace of table names.

In order to fit into this model, Kudu tables must be assigned a database when the HMS integration is enabled. No new APIs have been added to create or delete databases, nor are there APIs to assign an existing Kudu table to a database. Instead, a new convention has been introduced that Kudu table names must be in the format <hive-database-name>.<hive-table-name>. Thus, databases are an implicit part of the Kudu table name. By including databases as an implicit part of the Kudu table name, existing applications that use Kudu tables can operate on non-HMS-integrated and HMS-integrated table names with minimal or no changes.

Kudu provides no additional tooling to create or drop Hive databases. Administrators or users should use existing Hive tools such as the Beeline Shell or Impala to do so.

Naming Constraints

When the Hive Metastore integration is enabled, the database and table names of Kudu tables must follow the Hive Metastore naming constraints. Namely, the database and table name must contain only alphanumeric ASCII characters and underscores.

When the hive.support.special.characters.tablename Hive configuration is true, the forward-slash (/) character in table name identifiers (i.e. the table name and database name) is also supported.

Additionally, the Hive Metastore does not enforce case sensitivity for table name identifiers. As such, when enabled, Kudu will follow suit and disallow tables from being created when one already exists whose table name identifier differs only by case. Operations that open, alter, or drop tables will also be case-insensitive for the table name identifiers.

Warning: Given the case insensitivity upon enabling the integration, if multiple Kudu tables exist whose names only differ by case, the Kudu master(s) will fail to start up. Be sure to rename such conflicting tables before enabling the Hive Metastore integration.

Metadata Synchronization

When the Hive Metastore integration is enabled, Kudu will automatically synchronize metadata changes to Kudu tables between Kudu and the HMS. As such, it is important to always ensure that the Kudu and HMS have a consistent view of existing tables, using the administrative tools. Failure to do so may result in issues like Kudu tables not being discoverable or usable by external, HMS-aware components (e.g. Apache Sentry, Apache Impala).

The Hive Metastore automatically creates directories for Kudu tables. These directories are benign and can safely be ignored.

Impala has notions of internal and external Kudu tables. When dropping an internal table from Impala, the table's data is dropped in Kudu; in contrast when dropping an external table, the table's data is not dropped in Kudu. External tables may refer to tables by names that are different from the names of the underlying Kudu tables, while internal tables must use the same names as those stored in Kudu. Additionally, multiple external tables may refer to the same underlying Kudu table. Thus, since external tables may not map one-to-one with Kudu tables, the Hive Metastore integration and tooling will only automatically synchronize metadata for internal tables.

Related Information

Administrative tools for Hive Metastore integration

Administrative tools for Hive Metastore integration

Kudu provides the command line tools kudu hms list, kudu hms precheck, kudu hms check, and kudu hms fix to allow administrators to find and fix metadata inconsistencies between the internal Kudu catalog and the Hive Metastore catalog.

The kudu hms tools should be run from the command line as the Kudu admin user. They require the full list of master addresses to be specified:

```
$ sudo -u kudu kudu hms check master-name-1:7051,master-name-2:7051,master-n
ame-3:7051
```

To see a full list of the options available with the kudu hms tool, use the --help flag.

When fine-grained authorization is enabled, the Kudu admin user, commonly "kudu", needs to have access to all the Kudu tables to be able to run the kudu hms tools. This can be done by configuring the user as a trusted user via the --trusted_user_acl master configuration.

If the Hive Metastore is configured with fine-grained authorization using Apache Sentry, the Kudu admin user needs to have read and write privileges on HMS table entries. Configured this in the Hive Metastore using the sentry.metastore.service.users property.

kudu hms list

The kudu hms list tool scans the Hive Metastore catalog, and lists the HMS entries (including table name and type) for Kudu tables, as indicated by their HMS storage handler.

kudu hms check

The kudu hms check tool scans the Kudu and Hive Metastore catalogs, and validates that the two catalogs agree on what Kudu tables exist. The tool will make suggestions on how to fix any inconsistencies that are found. Typically, the suggestion will be to run the kudu hms fix tool, however some certain inconsistencies require using Impala Shell for fixing.

kudu hms precheck

The kudu hms precheck tool scans the Kudu catalog and validates that if there are multiple Kudu tables whose names only differ by case and logs the conflicted table names.

kudu hms fix

The kudu hms fix tool analyzes the Kudu and HMS catalogs and attempts to fix any automatically-fixable issues, for instance, by creating a table entry in the HMS for each Kudu table that doesn't

already have one. The --dryrun option shows the proposed fix instead of actually executing it. When no automatic fix is available, it will make suggestions on how a manual fix can help.

kudu hms downgrade

The kudu hms downgrade downgrades the metadata to legacy format for Kudu and the Hive Metastores. It is discouraged to use unless necessary, since the legacy format can be deprecated in future releases.

Upgrading existing Kudu tables for Hive Metastore integration

Before enabling the Kudu-HMS integration upgrade existing Kudu tables to ensure that Kudu and Hive Metstore (HMS) start with a consistent view of existing tables

Before you begin

- Establish a maintenance window. During this time the Kudu cluster will still be available, but tables in Kudu and the Hive Metastore may be altered or renamed as a part of the upgrade process.
- Make note of all external tables using the following command and drop them. This reduces the chance of having naming conflicts with Kudu tables which can lead to errors during upgrading process. It also helps in cases where a catalog upgrade breaks external tables, due to the underlying Kudu tables being renamed. The external tables can be recreated after upgrade is complete.

```
$ sudo -u kudu kudu hms list master-name-1:7051,master-name-2:7051,master-
name-3:7051
```

Procedure

1. Run the kudu hms precheck tool to ensure no Kudu tables only differ by case. If the tool does not report any warnings, you can skip the next step.

```
$ sudo -u kudu kudu table rename_table master-name-1:7051,master-name-2:
7051,master-name-3:7051 <conflicting_table_name> <new_table_name>
```

2. Run the kudu hms check tool using the following command. If the tool does not report any catalog inconsistencies, you can skip the following steps and can continue with *Enabling the Hive Metastore integration.*

```
$ sudo -u kudu kudu hms check master-name-1:7051,master-name-2:7051,mast
er-name-3:7051 --hive_metastore_uris=<hive_metastore_uris> [--ignore_oth
er_clusters=<ignores_other_clusters>]
```

By default, the kudu hms tools will ignore metadata in the HMS that refer to a different Kudu cluster than that being operated on, as indicated by having different masters specified. The tools compare the value of the kudu .master_addresses table property (either supplied at table creation or as --kudu_master_hosts on impalad daemons) in each HMS metadata entry against the RPC endpoints (including the ports) of the Kudu masters. To have the tooling account for and fix metadata entries with different master RPC endpoints specified (e.g. if ports are not specified in the HMS), supply --ignore_other_clusters=false as an argument to the kudu hms check and fix tools.

For example:

```
$ sudo -u kudu kudu hms check master-name-1:7051,master-name-2:7051,mast
er-name-3:7051 --hive_metastore_uris=thrift://hive-metastore:9083 --igno
re_other_clusters=false
```

3. If the kudu hms check tool reports an inconsistent catalog, perform a dry-run of the kudu hms fix tool to understand how the tool will attempt to address the automatically-fixable issues.

```
$ sudo -u kudu kudu hms fix master-name-1:7051,master-name-2:7051,master-
name-3:7051 --hive_metastore_uris=<hive_metastore_uris> --dryrun=true [--
ignore_other_clusters=<ignore_other_clusters>]
```

For example:

```
$ sudo -u kudu kudu hms check master-name-1:7051,master-name-2:7051,mast
er-name-3:7051 --hive_metastore_uris=thrift://hive-metastore:9083 --dryr
un=true --ignore_other_clusters=false
```

4. Manually fix any issues that are reported by the check tool that cannot be automatically fixed. For example, rename any tables with names that are not Hive-conformant.
5. Run the kudu hms fix tool to automatically fix all the remaining issues.

```
$ sudo -u kudu kudu hms fix master-name-1:7051,master-name-2:7051,master
-name-3:7051 --hive_metastore_uris=<hive_metastore_uris> [--drop_orphan_
hms_tables=<drops_orphan_hms_tables>] [--ignore_other_clusters=<ignore_o
ther_clusters>]
```

For example:

```
$ sudo -u kudu kudu hms fix master-name-1:7051,master-name-2:7051,master
-name-3:7051 --hive_metastore_uris=thrift://hive-metastore:9083 --ignore
_other_clusters=false
```

Note: The --drop_orphan_hms_tables argument indicates whether to drop orphan Hive Metastore tables that refer to non-existent Kudu tables. Due to KUDU-2883, this option may fail to drop HMS entries that have no table ID. A workaround to this is to drop the table via the Impala Shell.

6. Recreate any external tables that were dropped when preparing for the upgrade by using Impala Shell.

What to do next

Enabling the Hive Metastore integration.

Related Information

Enabling the Hive Metastore integration

Enabling the Hive Metastore integration

You can enable the Hive Metastore (HMS) integration with Kudu using Cloudera Manager.

Before you begin

Upgrade any tables that may exist in Kudu's or in the HMS's catalog. For more information, see *Upgrading Existing Tables*.

Procedure

1. In Cloudera Manager navigate to Kudu Configuration .
2. Find the HMS Service property and select the Hive Metastore with which Kudu will synchronize its system catalog.

Related Information

Upgrading existing Kudu tables for Hive Metastore integration

Using Apache Impala with Apache Kudu

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Understanding Impala integration with Kudu.....4
Impala database containment model.....4
Internal and external Impala tables.....4
Verifying the Impala dependency on Kudu.....5
Impala integration limitations.....5

Using Impala to query Kudu tables.....5
Query an existing Kudu table from Impala.....6
Create a new Kudu table from Impala.....6
Use CREATE TABLE AS SELECT.....7
Partitioning tables.....8
Basic partitioning.....8
Advanced partitioning.....10
Non-covering range partitions.....11
Partitioning guidelines.....11
Optimize performance for evaluating SQL predicates.....12
Insert data into Kudu using Impala.....12
INSERT and primary key uniqueness violations.....13
Update data in Kudu using Impala.....13
Upsert a row.....13
Alter a table.....15
Delete data from Kudu using Impala.....15
Failures during INSERT, UPDATE, UPSERT, and DELETE operations.....16
Drop a Kudu table using Impala.....16

Understanding Impala integration with Kudu

Apache Kudu has tight integration with Apache Impala. Before you start using Impala to query Kudu tables, learn about the Impala database containment model and Impala tables, and read about the limitations of the integration.

The tight integration between Impala and Kudu allows you to use Impala to insert, query, update, and delete data from Kudu tablets using Impala's SQL syntax, as an alternative to using the Kudu APIs to build a custom Kudu application. In addition, you can use JDBC or ODBC to connect existing or new applications written in any language, framework, or business intelligence tool to your Kudu data, using Impala as the broker.

Impala database containment model

Every Impala table is contained within a namespace called a database. The default database is called default, and you can create and drop additional databases as desired.

Important: The Impala database containment model is not supported if HMS integration is enabled.

To create the database, use a CREATE DATABASE statement. To use the database for further Impala operations such as CREATE TABLE, use the USE statement.

For example, to create a table in a database called impala_kudu, use the following statements:

```
CREATE DATABASE impala_kudu;
USE impala_kudu;
CREATE TABLE my_first_table (
...
```

The my_first_table table is created within the impala_kudu database.

The prefix impala:: and the Impala database name are appended to the underlying Kudu table name: impala::<database>.<table>

For example, to specify the my_first_table table in database impala_kudu, as opposed to any other table with the same name in another database, refer to the table as impala::impala_kudu.my_first_table. This also applies to INSERT, UPDATE, DELETE, and DROP statements.

Internal and external Impala tables

When creating a new Kudu table using Impala, you can create the table as an internal table or an external table.

Internal

An internal table (created by CREATE TABLE) is managed by Impala, and can be dropped by Impala. When you create a new table using Impala, it is generally a internal table. When such a table is created in Impala, the corresponding Kudu table will be named impala::database_name.table_name. The prefix is always impala::, and the database name and table name follow, separated by a dot.

External

An external table (created by CREATE EXTERNAL TABLE) is not managed by Impala, and dropping such a table does not drop the table from its source location (here, Kudu). Instead, it only removes the mapping between Impala and Kudu. This is the mode used in the syntax provided by Kudu for mapping an existing table to Impala.

Verifying the Impala dependency on Kudu

In a Cloudera Manager deployment, once the Kudu service is installed, Impala will automatically identify the Kudu Master. However, if your Impala queries don't work as expected, use the following steps to make sure that the Impala service is set to be dependent on Kudu.

Procedure

1. Go to the Impala service.
2. Click the Configuration tab.
3. Search for kudu.
4. Make sure the Kudu Service property is set to the right Kudu service.
5. Click Save Changes.

Impala integration limitations

Here are the limitations when integrating Kudu with Impala, and list of Impala keywords that are not supported for creating Kudu tables.

- When creating a Kudu table, the CREATE TABLE statement must include the primary key columns before other columns, in primary key order.
- Impala cannot update values in primary key columns.
- Kudu tables with a name containing upper case or non-ASCII characters must be assigned an alternate name when used as an external table in Impala.
- Kudu tables with a column name containing upper case or non-ASCII characters cannot be used as an external table in Impala. Columns can be renamed in Kudu to work around this issue.
- != and LIKE predicates are not pushed to Kudu, and instead will be evaluated by the Impala scan node. This may decrease performance relative to other types of predicates.
- Updates, inserts, and deletes using Impala are non-transactional. If a query fails part of the way through, its partial effects will not be rolled back.
- The maximum parallelism of a single query is limited to the number of tablets in a table. For good analytic performance, aim for 10 or more tablets per host or use large tables.

Impala keywords not supported for creating Kudu tables

- PARTITIONED
- LOCATION
- ROWFORMAT

Using Impala to query Kudu tables

If you want to use Impala to query Kudu tables, you have to create a mapping between the Impala and Kudu tables.

Neither Kudu nor Impala need special configuration in order for you to use the Impala Shell or the Impala API to insert, update, delete, or query Kudu data using Impala. However, you do need to create a mapping between the Impala and Kudu tables. Kudu provides the Impala query to map to an existing Kudu table in the web UI.

- Make sure you are using the impala-shell binary provided by the default Impala binary. The following example shows how you can verify this using the alternatives command on a RHEL 6 host. Do not copy and paste the alte rnatives --set command directly, because the file names are likely to differ.

```
$ sudo alternatives --display impala-shell
```

```
impala-shell - status is auto.
 link currently points to /opt/cloudera/parcels/<current_CDP_parcel>/bin/
impala-shell
/opt/cloudera/parcels/<current_CDP_parcel>/bin/impala-shell - priority 10
Current `best' version is opt/cloudera/parcels/<current_CDP_parcel>/bin/i
mpala-shell
```

- Although not necessary, it is recommended that you configure Impala with the locations of the Kudu Masters using the --kudu_master_hosts=<master1>[:port] flag. If this flag is not set, you will need to manually provide this configuration each time you create a table by specifying the kudu.master_addresses property inside a TBLP ROPERTIES clause. If you are using Cloudera Manager, no such configuration is needed. The Impala service will automatically recognize the Kudu Master hosts. However, if your Impala queries don't work as expected, use the following steps to make sure that the Impala service is set to be dependent on Kudu:

  1. Go to the Impala service.
  2. Click the Configuration tab and search for kudu.
  3. Make sure that the Kudu Service property is set to the right Kudu service.
  4. Click Save Changes.

  Before you carry out any of the operations listed within this section, make sure that this configuration has been set.
- Start Impala Shell using the impala-shell command. By default, impala-shell attempts to connect to the Impala daemon on localhost on port 21000. To connect to a different host, use the -i <host:port> option.

  To automatically connect to a specific Impala database, use the -d <database> option. For instance, if all your Kudu tables are in Impala in the database impala_kudu, use -d impala_kudu to use this database.
- To quit the Impala Shell, use the following command: quit;

Query an existing Kudu table from Impala

Tables created through the Kudu API or other integrations such as Apache Spark are not automatically visible in Impala.

To query them, you must first create an external table within Impala to map the Kudu table into an Impala database:

```
CREATE EXTERNAL TABLE my_mapping_table
STORED AS KUDU
TBLPROPERTIES (
  'kudu.table_name' = 'my_kudu_table'
);
```

Create a new Kudu table from Impala

Creating a new table in Kudu from Impala is similar to mapping an existing Kudu table to an Impala table, except that you need to specify the schema and partitioning information yourself. Use the examples in this section as a guideline. Impala first creates the table, then creates the mapping.

In the CREATE TABLE statement, the columns that comprise the primary key must be listed first. Additionally, primary key columns are implicitly considered NOT NULL.

When creating a new table in Kudu, you must define a partition schema to pre-split your table. The best partition schema to use depends upon the structure of your data and your data access patterns. The goal is to maximize parallelism and use all your tablet servers evenly.

Note: In Impala included in CDH 5.13 and higher, the PARTITION BY clause is optional for Kudu tables. If the clause is omitted, Impala automatically constructs a single partition that is not connected to any column. Because such a table cannot take advantage of Kudu features for parallelized queries and query optimizations, omitting the PARTITION BY clause is only appropriate for small lookup tables.

The following CREATE TABLE example distributes the table into 16 partitions by hashing the id column, for simplicity.

```
CREATE TABLE my_first_table
(
  id BIGINT,
  name STRING,
  PRIMARY KEY(id)
)
PARTITION BY HASH PARTITIONS 16
STORED AS KUDU;
```

By default, Kudu tables created through Impala use a tablet replication factor of 3. To specify the replication factor for a Kudu table, add a TBLPROPERTIES clause to the CREATE TABLE statement as shown below where n is the replication factor you want to use:

```
TBLPROPERTIES ('kudu.num_tablet_replicas' = 'n')
```

A replication factor must be an odd number.

Changing the kudu.num_tablet_replicas table property using the ALTER TABLE currently has no effect.

The Impala SQL Reference *CREATE TABLE* topic has more details and examples.

The mapping between Kudu and Impala data types is summarized in the the following table:

Kudu type	Impala type
BOOL (boolean)	BOOLEAN
INT8 (8-bit signed integer)	TINYINT
INT16 (16-bit signed integer)	SMALLINT
INT32 (32-bit signed integer)	INT
INT64 (64-bit signed integer)	BIGINT
DATE (32-bit days since the Unix epoch)	NA (Not supported by Impala)
UNIXTIME_MICROS (64-bit microseconds Unix epoch)	TIMESTAMP (For details, see *Notes*.)
FLOAT (single-precision 32-bit IEEE-754 floating-point)	FLOAT
DOUBLE (double-precision 64-bit IEEE-754 floating-point)	DOUBLE
DECIMAL	DECIMAL
VARCHAR	VARCHAR
STRING	STRING
BINARY	BINARY (Not yet supported. For details, see IMPALA-5323.)

Note: For more details about Timestamp data type, see TIMESTAMP data type. For more details on handling Kudu types, column attributes, partitioning, and using Impala to create and query Kudu tables, see Column Design and Handling Date, Time, or Timestamp Data with Kudu.

Related Information

Partitioning tables

Use CREATE TABLE AS SELECT

You can create a table by querying any other table or tables in Impala, using a CREATE TABLE ... AS SELECT statement.

The following example imports all rows from an existing table, old_table, into a new Kudu table, new_table. The columns in new_table will have the same names and types as the columns in old_table, but you will need to additionally specify the primary key and partitioning schema.

```
CREATE TABLE new_table
PRIMARY KEY (ts, name)
PARTITION BY HASH(name) PARTITIONS 8
STORED AS KUDU
AS SELECT ts, name, value FROM old_table;
```

You can refine the SELECT statement to only match the rows and columns you want to be inserted into the new table. You can also rename the columns by using syntax like SELECT name as new_col_name.

Partitioning tables

Tables are partitioned into tablets according to a partition schema on the primary key columns. Each tablet is served by at least one tablet server. Ideally, a table should be split into tablets that are distributed across a number of tablet servers to maximize parallel operations. The details of the partitioning schema you use will depend entirely on the type of data you store and how you access it.

Kudu currently has no mechanism for splitting or merging tablets after the table has been created. Until this feature has been implemented, you must provide a partition schema for your table when you create it. When designing your tables, consider using primary keys that will allow you to partition your table into tablets which grow at similar rates.

You can partition your table using Impala's PARTITION BY clause, which supports distribution by RANGE or HASH. The partition scheme can contain zero or more HASH definitions, followed by an optional RANGE definition. The RANGE definition can refer to one or more primary key columns. Examples of basic and advanced partitioning are shown below.

Monotonically Increasing Values - If you partition by range on a column whose values are monotonically increasing, the last tablet will grow much larger than the others. Additionally, all data being inserted will be written to a single tablet at a time, limiting the scalability of data ingest. In that case, consider distributing by HASH instead of, or in addition to, RANGE.

Note: Impala keywords, such as group, are enclosed by back-tick characters when they are used as identifiers, rather than as keywords.

In general, be mindful the number of tablets limits the parallelism of reads, in the current implementation. Increasing the number of tablets significantly beyond the number of cores is likely to have diminishing returns.

Basic partitioning

In basic partitioning, you can either partition by range, or partition by hash.

PARTITION BY RANGE

You can specify range partitions for one or more primary key columns. Range partitioning in Kudu allows splitting a table based on specific values or ranges of values of the chosen partition keys. This allows you to balance parallelism in writes with scan efficiency.

For instance, if you have a table that has the columns state, name, and purchase_count, and you partition the table by state, it will create 50 tablets, one for each US state.

```
CREATE TABLE customers (
  state STRING,
  name STRING,
  purchase_count int,
  PRIMARY KEY (state, name)
)
PARTITION BY RANGE (state)
```

```
(
  PARTITION VALUE = 'al',
  PARTITION VALUE = 'ak',
  PARTITION VALUE = 'ar',
  ...
  ...
  PARTITION VALUE = 'wv',
  PARTITION VALUE = 'wy'
)
STORED AS KUDU;
```

PARTITION BY HASH

Instead of distributing by an explicit range, or in combination with range distribution, you can distribute into a specific number of partitions by hash. You specify the primary key columns you want to partition by, and the number of partitions you want to use. Rows are distributed by hashing the specified key columns. Assuming that the values being hashed do not themselves exhibit significant skew, this will serve to distribute the data evenly across all partitions.

You can specify multiple definitions, and you can specify definitions which use compound primary keys. However, one column cannot be mentioned in multiple hash definitions. Consider two columns, a and b:

- HASH(a), HASH(b) -- will succeed
- HASH(a,b) -- will succeed
- HASH(a), HASH(a,b) -- will fail

Note: PARTITION BY HASH with no column specified is a shortcut to create the desired number of partitions by hashing all primary key columns.

Hash partitioning is a reasonable approach if primary key values are evenly distributed in their domain and no data skew is apparent, such as timestamps or serial IDs.

The following example creates 16 tablets by hashing the id column. A maximum of 16 tablets can be written to in parallel. In this example, a query for a range of sku values is likely to need to read from all 16 tablets, so this may not be the optimum schema for this table. See the *Advanced partinioning* section for an extended example.

```
CREATE TABLE cust_behavior (
  id BIGINT,
  sku STRING,
  salary STRING,
  edu_level INT,
  usergender STRING,
  `group` STRING,
  city STRING,
  postcode STRING,
  last_purchase_price FLOAT,
  last_purchase_date BIGINT,
  category STRING,
  rating INT,
  fulfilled_date BIGINT,
  PRIMARY KEY (id, sku)
)
PARTITION BY HASH PARTITIONS 16
STORED AS KUDU;
```

Related Information

Advanced partitioning

Advanced partitioning

You can combine HASH and RANGE partitioning to create more complex partition schemas. You can also specify zero or more HASH definitions, followed by zero or one RANGE definitions. Each schema definition can encompass one or more columns. While enumerating every possible distribution schema is out of the scope of this topic, the following examples illustrate some of the possibilities.

PARTITION BY HASH and RANGE

Consider the basic PARTITION BY HASH example above. If you often query for a range of sku values, you can optimize the example by combining hash partitioning with range partitioning.

The following example still creates 16 tablets, by first hashing the id column into 4 partitions, and then applying range partitioning to split each partition into four tablets, based upon the value of the sku string. At least four tablets (and possibly up to 16) can be written to in parallel, and when you query for a contiguous range of sku values, there's a good chance you only need to read a quarter of the tablets to fulfill the query.

By default, the entire primary key (id, sku) will be hashed when you use PARTITION BY HASH. To hash on only part of the primary key, and use a range partition on the rest, use the syntax demonstrated below.

```
CREATE TABLE cust_behavior (
  id BIGINT,
  sku STRING,
  salary STRING,
  edu_level INT,
  usergender STRING,
  `group` STRING,
  city STRING,
  postcode STRING,
  last_purchase_price FLOAT,
  last_purchase_date BIGINT,
  category STRING,
  rating INT,
  fulfilled_date BIGINT,
  PRIMARY KEY (id, sku)
)
PARTITION BY HASH (id) PARTITIONS 4,
RANGE (sku)
(
  PARTITION VALUES < 'g',
  PARTITION 'g' <= VALUES < 'o',
  PARTITION 'o' <= VALUES < 'u',
  PARTITION 'u' <= VALUES
)
STORED AS KUDU;
```

Multiple PARTITION BY HASH Definitions

Once again expanding on the example above, let's assume that the pattern of incoming queries will be unpredictable, but you still want to ensure that writes are spread across a large number of tablets. You can achieve maximum distribution across the entire primary key by hashing on both primary key columns.

```
CREATE TABLE cust_behavior (
  id BIGINT,
  sku STRING,
  salary STRING,
  edu_level INT,
  usergender STRING,
  `group` STRING,
  city STRING,
  postcode STRING,
  last_purchase_price FLOAT,
```

```
  last_purchase_date BIGINT,
  category STRING,
  rating INT,
  fulfilled_date BIGINT,
  PRIMARY KEY (id, sku)
)
PARTITION BY HASH (id) PARTITIONS 4,
             HASH (sku) PARTITIONS 4
STORED AS KUDU;
```

The example creates 16 partitions. You could also use HASH (id, sku) PARTITIONS 16. However, a scan for sku values would almost always impact all 16 partitions, rather than possibly being limited to 4.

Non-covering range partitions

Kudu supports the use of non-covering range partitions, which can be used to address the following scenarios:

- In the case of time-series data or other schemas which need to account for constantly-increasing primary keys, tablets serving old data will be relatively fixed in size, while tablets receiving new data will grow without bounds.
- In cases where you want to partition data based on its category, such as sales region or product type, without non-covering range partitions you must know all of the partitions ahead of time or manually recreate your table if partitions need to be added or removed, such as the introduction or elimination of a product type.

Note: See *Range partitioning* for the caveats of non-covering range partitions.

The following example creates a tablet per year (5 tablets total), for storing log data. The table only accepts data from 2012 to 2016. Keys outside of these ranges will be rejected.

```
CREATE TABLE sales_by_year (
  year INT, sale_id INT, amount INT,
  PRIMARY KEY (year, sale_id)
)
PARTITION BY RANGE (year) (
  PARTITION VALUE = 2012,
  PARTITION VALUE = 2013,
  PARTITION VALUE = 2014,
  PARTITION VALUE = 2015,
  PARTITION VALUE = 2016
)
STORED AS KUDU;
```

When records start coming in for 2017, they will be rejected. At that point, the 2017 range should be added as follows:

```
ALTER TABLE sales_by_year ADD RANGE PARTITION VALUE = 2017;
```

In use cases where a rolling window of data retention is required, range partitions may also be dropped. For example, if data from 2012 should no longer be retained, it may be deleted in bulk:

```
ALTER TABLE sales_by_year DROP RANGE PARTITION VALUE = 2012;
```

Note that just like dropping a table, this irrecoverably deletes all data stored in the dropped partition.

Partitioning guidelines

Here are some partitioning guidelines for small and large tables.

- For large tables, such as fact tables, aim for as many tablets as you have cores in the cluster.
- For small tables, such as dimension tables, ensure that each tablet is at least 1 GB in size.

Optimize performance for evaluating SQL predicates

If the WHERE clause of your query includes comparisons with the operators =, <=, <, >, >=, BETWEEN, or IN, Kudu evaluates the condition directly and only returns the relevant results. This provides optimum performance, because Kudu only returns the relevant results to Impala.

For predicates such as !=, LIKE, or any other predicate type supported by Impala, Kudu does not evaluate the predicates directly. Instead, it returns all results to Impala and relies on Impala to evaluate the remaining predicates and filter the results accordingly. This may cause differences in performance, depending on the delta of the result set before and after evaluating the WHERE clause. In some cases, creating and periodically updating materialized views may be the right solution to work around these inefficiencies.

Insert data into Kudu using Impala

Using Impala you can index one or more rows, or you can insert in bulk. Before inserting, learn about the syntax, and the advantages and disadvantages of the three inserting in bulk methods.

Inserting a row

The following syntax shows how to insert one or more rows using Impala:

```
INSERT INTO my_first_table VALUES (99, "sarah");
INSERT INTO my_first_table VALUES (1, "john"), (2, "jane"), (3, "jim");
```

The primary key must not be null.

Multiple single INSERT statements

Using multiple single INSERT statements is an insert in bulk method.

This approach has the advantage of being easy to understand and implement. This approach is likely to be inefficient because Impala has a high query start-up cost compared to Kudu's insertion performance. This will lead to relatively high latency and poor throughput.

Single INSERT statement with multiple VALUES subclauses

Using a single INSERT statements with multiple VALUES subclauses is an insert in bulk method.

If you include more than 1024 VALUES statements, Impala batches them into groups of 1024 (or the value of batch_size) before sending the requests to Kudu. This approach may perform slightly better than multiple sequential INSERT statements by amortizing the query start-up penalties on the Impala side. To set the batch size for the current Impala Shell session, use the following syntax:

```
set batch_size=10000;
```

Note: Increasing the Impala batch size causes Impala to use more memory. You should verify the impact on your cluster and tune accordingly.

Batch insert

Batch insert is an insert in bulk method.

The approach that usually performs best, from the standpoint of both Impala and Kudu, is usually to import the data using a SELECT FROM subclause in Impala.

1. If your data is not already in Impala, one strategy is to import it from a text file, such as a TSV or CSV file.
2. Create the Kudu table, being mindful that the columns designated as primary keys cannot have null values.

3. Insert values into the Kudu table by querying the table containing the original data, as in the following example:

```
INSERT INTO my_kudu_table
SELECT * FROM legacy_data_import_table;
```

Ingest using the C++ or Java API

In many cases, the appropriate ingest path is to use the C++ or Java API to insert directly into Kudu tables. Unlike other Impala tables, data inserted into Kudu tables using the API becomes available for query in Impala without the need for any INVALIDATE METADATA statements or other statements needed for other Impala storage types.

Related Information

Create a new Kudu table from Impala

INSERT and primary key uniqueness violations

In many relational databases, if you try to insert a row that has already been inserted, the operation can fail because the primary key gets duplicated. Impala, however, does not fail the query. Instead, it generates a warning and continues to run the remainder of the insert statement.

If you want to replace existing rows from the table, use the UPSERT statement instead.

```
INSERT INTO my_first_table VALUES (99, "sarah");
UPSERT INTO my_first_table VALUES (99, "zoe");
```

The current value of the row is now zoe.

Update data in Kudu using Impala

Using Impala, you can update one or more rows, or update in bulk.

Updating a row

The following syntax shows how to update one or more rows using Impala:

```
UPDATE my_first_table SET name="bob" where id = 3;
```

You cannot change or null the primary key value.

Important: The UPDATE statement only works in Impala when the underlying data source is Kudu.

Updating in bulk

You can update in bulk using the Inserting in bulk approaches outlined in the *Inserting* topic.

Related Information

Insert data into Kudu using Impala

Upsert a row

The UPSERT command acts as a combination of the INSERT and UPDATE statements.

For each row processed by the UPSERT statement:

- If another row already exists with the same set of primary key values, the other columns are updated to match the values from the row being 'UPSERTed'.

- If there is no row with the same set of primary key values, the row is created, the same as if the INSERT statement was used.

UPSERT Example

The following example demonstrates how the UPSERT statement works. We start by creating two tables, foo1 and foo2.

```
CREATE TABLE foo1 (
 id INT PRIMARY KEY,
 col1 STRING,
 col2 STRING
)
PARTITION BY HASH(id) PARTITIONS 3
STORED AS KUDU;
```

```
CREATE TABLE foo2 (
 id INT PRIMARY KEY,
 col1 STRING,
 col2 STRING
)
PARTITION BY HASH(id) PARTITIONS 3
STORED AS KUDU;
```

Populate foo1 and foo2 using the following INSERT statements. For foo2, we leave column col2 with NULL values to be upserted later:

```
INSERT INTO foo1 VALUES (1, "hi", "alice");
```

```
INSERT INTO foo2 select id, col1, NULL from foo1;
```

The contents of foo2 will be:

```
SELECT * FROM foo2;
...
+----+------+------+
| id | col1 | col2 |
+----+------+------+
| 1  | hi   | NULL |
+----+------+------+
Fetched 1 row(s) in 0.15s
```

Now use the UPSERT command to now replace the NULL values in foo2 with the actual values from foo1.

```
UPSERT INTO foo2 (id, col2) select id, col2 from foo1;
```

```
SELECT * FROM foo2;
...
+----+------+-------+
| id | col1 | col2  |
+----+------+-------+
| 1  | hi   | alice |
+----+------+-------+
Fetched 1 row(s) in 0.15s
```

Alter a table

You can use the ALTER TABLE statement to change the default value, encoding, compression, or block size of existing columns in a Kudu table, or to change Impala's metadata relating to a given Kudu table by altering the table's properties.

The Impala SQL Reference *ALTER TABLE* includes a Kudu Considerations section with examples and a list of constraints relevant to altering a Kudu table in Impala.

Altering the table properties

Important: Alter table statements do not make any change to the Kudu data for EXTERNAL tables.

Rename an external Impala table

```
ALTER TABLE my_table RENAME TO my_new_table;
```

If you rename an external table, it does not change the name of the Kudu table.

If you rename a managed table, the name of the Kudu table is changed accordingly.

Remapping an external table to a different Kudu table

If another application has renamed a Kudu table under Impala, it is possible to re-map an external table to point to a different Kudu table name.

```
ALTER TABLE my_external_table_
SET TBLPROPERTIES('kudu.table_name' = 'some_other_kud
u_table')
```

Change the Kudu master addresses

```
ALTER TABLE my_table SET TBLPROPERTIES('kudu.master_
addresses' = 'kudu-original-master.example.com:7051,
kudu-new-master.example.com:7051');
```

Change an internally-managed table to external

```
ALTER TABLE my_table SET TBLPROPERTIES('EXTERNAL' =
'TRUE');
```

Related Information

ALTER TABLE statement

Delete data from Kudu using Impala

You can delete Kudu rows in near real time using Impala.

```
DELETE FROM my_first_table WHERE id < 3;
```

You can even use more complex joins when deleting rows. For example, Impala uses a comma in the FROM sub-clause to specify a join query.

```
DELETE c FROM my_second_table c, stock_symbols s WHERE c.name = s.symbol;
```

Important: The DELETE statement only works in Impala when the underlying data source is Kudu.

You can delete in bulk using the Inserting in bulk approaches outlined in the *Inserting* topic.

Related Information

Insert data into Kudu using Impala

Failures during INSERT, UPDATE, UPSERT, and DELETE operations

INSERT, UPDATE, and DELETE statements cannot be considered transactional as a whole. If one of these operations fails part of the way through, the keys may have already been created (in the case of INSERT) or the records may have already been modified or removed by another process (in the case of UPDATE or DELETE). You should design your application with this in mind.

Drop a Kudu table using Impala

You can drop internal tables in Impala using the DROP TABLE syntax. If you want to drop an external table, you first have to change it to internal.

If the table was created as an internal table in Impala, using CREATE TABLE, the standard DROP TABLE syntax drops the underlying Kudu table and all its data.

```
DROP TABLE my_first_table;
```

If the table was created as an external table, using CREATE EXTERNAL TABLE, the mapping between Impala and Kudu is dropped, but the Kudu table is left intact, with all its data. To change an external table to internal, and the other way around, see *Altering table properties*.

Related Information

Alter a table

Managing Apache Kudu

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Limitations 5
Server management limitations 5
Cluster management limitations 5

Start and stop Kudu processes 5

Orchestrate a rolling restart with no downtime 6
Minimize cluster disruption during temporary planned downtime of a single tablet server 7

Kudu web interfaces 7
Kudu master web interface 7
Kudu tablet server web interface 7
Common web interface pages 7

Best practices when adding new tablet servers 8

Decommission or remove a tablet server 8

Use cluster names in the kudu command line tool 9

Migrate Kudu data from one directory to another on the same host 9

Migrate to a multiple Kudu master configuration 10

Change master hostnames 12
Prepare for master hostname changes 12
Perform master hostname changes 12

Removing Kudu masters through Cloudera Manager 14
Recommissioning Kudu masters through Cloudera Manager 14

Remove Kudu masters through CLI 15
Prepare for removal 15
Perform the removal 15

How Range-aware replica placement in Kudu works 16

Run the tablet rebalancing tool..... 17
Run a tablet rebalancing tool on a rack-aware cluster..... 18
Example of rebalancing tool with rack awareness functionality..... 19
Run a tablet rebalancing tool in Cloudera Manager..... 20
Run a tablet rebalancing tool in command line..... 21

Managing Kudu tables with range-specific hash schemas..... 21
Range-specific hash schemas example: Using impala-shell..... 21
Range-specific hash schemas example: Using Kudu C++ client API..... 23
Range-specific hash schemas example: Using Kudu Java client API..... 24

Limitations

Review the server management and cluster management guidelines that you should consider before implementing Kudu.

Server management limitations

Here are some of the server management guidelines that you should consider before implementing Kudu.

- Production deployments should configure a least 4 GiB of memory for tablet servers, and ideally more than 16 GiB when approaching the data and tablet scale limits.
- Write ahead logs (WALs) can only be stored on one disk.
- Data directories cannot be removed. You must reformat the data directories to remove them.
- Tablet servers cannot change their address or port.
- Kudu has a hard requirement on having an up-to-date NTP. Kudu masters and tablet servers will crash when out of sync.

Cluster management limitations

When managing Kudu clusters, review the following limitations and recommended maximum point-to-point latency and bandwidth values.

- Recommended maximum point-to-point latency within a Kudu cluster is 20 milliseconds.
- Recommended minimum point-to-point bandwidth within a Kudu cluster is 10 Gbps.
- If you intend to use the location awareness feature to place tablet servers in different locations, it is recommended that you measure the bandwidth and latency between servers to ensure they fit within the above guidelines.
- All masters must be started at the same time when the cluster is started for the very first time.

Start and stop Kudu processes

You can start, stop, and configure Kudu services to start automatilcally by using the CLI commands.

Start Kudu services using the following commands:

```
sudo service kudu-master start
sudo service kudu-tserver start
```

To stop Kudu services, use the following commands:

```
sudo service kudu-master stop
sudo service kudu-tserver stop
```

Configure the Kudu services to start automatically when the server starts, by adding them to the default runlevel.

```
sudo chkconfig kudu-master on            # RHEL / CentOS
sudo chkconfig kudu-tserver on           # RHEL / CentOS
sudo update-rc.d kudu-master defaults    # Ubuntu
sudo update-rc.d kudu-tserver defaults   # Ubuntu
```

Orchestrate a rolling restart with no downtime

Kudu 1.12 provides tooling to restart a cluster with no downtime. This topic provides the steps to perform rolling restart.

About this task

Note: If any tables in the cluster have a replication factor of 1, some quiescing tablet servers will never become fully quiesced, as single-replica tablets do not naturally relinquish leadership. If such tables exist, use the kudu cluster rebalance tool to move replicas of these tables away from the quiescing tablet server by specifying the --ignored_tservers, --move_replicas_from_ignored_tservers, and --tables options.

Note: If running with rack awareness, the following steps can be performed by restarting multiple tablet servers within a single rack at the same time. Use ksck to ensure that the location assignment policy is enforced while going through these steps, and that no more than a single location is restarted at the same time. At least three locations should be defined in the cluster to safely restart multiple tablet service within one location.

Cloudera Manager can automate this process, by using the “Rolling Restart” command on the Kudu service.

Note: Cloudera Manager does not support automatic moving of the single-replica tablets.

Cloudera Manager will prompt you to specify how many tablet servers to restart concurrently. If running with rack awareness with and at least three racks specified across all hosts that contain Kudu roles, it is safe to specify the restart batch with up to one rack at a time, provided the rack assignment policy is being enforced.

The following service configurations can be set to tune the parameters the rolling restart will run with:

- Rolling Restart Health Check Interval: the interval in seconds that Cloudera Manager will run ksck after restarting a batch of tablet servers, waiting for the cluster to become healthy.
- Maximum Allowed Runtime to Rolling Restart a Batch of Servers: the total amount of time in secondsCloudera Manager will wait for the cluster to become healthy after restarting a batch of tablet servers, before exiting with an error.

Procedure

1. Restart the master(s) one-by-one. If there is only a single master, this may cause brief interference with on-going workloads.
2. Starting with a single tablet server, put the tablet server into maintenance mode by using the kudu tserver state e nter_maintenance tool.
3. Start quiescing the tablet server using the kudu tserver quiesce start tool. This signals Kudu to stop hosting leaders on the specified tablet server and to redirect new scan requests to other tablet servers.
4. Periodically run kudu tserver quiesce start with the --error_if_not_fully_quiesced option, until it returns success, indicating that all leaders have been moved away from the tablet server and that all on-going scans have completed.
5. Restart the tablet server.
6. Periodically run ksck until the cluster ireports a healthy status.
7. Exit maintenance mode on the tablet server by running kudu tserver state exit_maintenance. This allows new tablet replicas to be placed on the tablet server.
8. Repeat these steps for all tablet servers in the cluster.

Related Information

Changing directory configuration

Minimize cluster disruption during temporary planned downtime of a single tablet server

If a single tablet server is brought down temporarily in a healthy cluster, all tablets will remain available and clients will function as normal, after potential short delays due to leader elections. However, if the downtime lasts for more than --follower_unavailable_considered_failed_sec (default 300) seconds, the tablet replicas on the down tablet server will be replaced by new replicas on available tablet servers. This will cause stress on the cluster as tablets re-replicate and, if the downtime lasts long enough, significant reduction in the number of replicas on the down tablet server, which may require the rebalancer to fix.

To work around this, in Kudu versions 1.11 onward, the kudu CLI contains a tool to put tablet servers into maintenance mode. While in this state, the tablet server's replicas are not re-replicated due to its downtime alone, though re-replication may still occur in the event that the server in maintenance suffers from a disk failure or if a follower replica on the tablet server falls too far behind its leader replica. Upon exiting maintenance, re-replication is triggered for any remaining under-replicated tablets.

The kudu tserver state enter_maintenance and kudu tserver state exit_maintenance tools are added to orchestrate tablet server maintenance. The following can be run from a tablet server to put it into maintenance:

```
$ TS_UUID=$(sudo -u kudu kudu fs dump uuid --fs_wal_dir=<wal_dir> --fs_data_
dirs=<data_dirs>)
$ sudo -u kudu kudu tserver state enter_maintenance <master_addresses> "$T
S_UUID"
```

The tablet server maintenance mode is shown in the "Tablet Servers" page of the Kudu leader master's web UI, and in the output of kudu cluster ksck. To exit maintenance mode, run the following command:

```
sudo -u kudu kudu tserver state exit_maintenance <master_addresses> "$TS_UUI
D"
```

Kudu web interfaces

Kudu tablet servers and masters expose useful operational information on a built-in web interface.

Kudu master web interface

Kudu master processes serve their web interface on port 8051. The interface exposes several pages with information about the state of the cluster.

- A list of tablet servers, their host names, and the time of their last heartbeat.
- A list of tables, including schema and tablet location information for each.
- SQL code which you can paste into Impala Shell to add an existing table to Impala's list of known data sources.

Kudu tablet server web interface

Each tablet server serves a web interface on port 8050. The interface exposes information about each tablet hosted on the server, its current state, and debugging information about maintenance background operations.

Common web interface pages

Both Kudu masters and tablet servers expose the following information via their web interfaces:

- HTTP access to server logs.

- An /rpcz endpoint which lists currently running RPCs via JSON.
- Details about the memory usage of different components of the process.
- The current set of configuration flags.
- Currently running threads and their resource consumption.
- A JSON endpoint exposing metrics about the server.
- The version number of the daemon deployed on the cluster.

These interfaces are linked from the landing page of each daemon's web UI.

Best practices when adding new tablet servers

A common workflow when administering a Kudu cluster is adding additional tablet server instances, in an effort to increase storage capacity, decrease load or utilization on individual hosts, increase compute power, and more.

By default, any newly added tablet servers will not be utilized immediately after their addition to the cluster. Instead, newly added tablet servers will only be utilized when new tablets are created or when existing tablets need to be replicated, which can lead to imbalanced nodes. It's recommended to run the rebalancer CLI tool just after adding a new tablet server into the cluster.

Avoid placing multiple tablet servers on a single node. Doing so nullifies the point of increasing the overall storage capacity of a Kudu cluster and increases the likelihood of tablet unavailability when a single node fails (the latter drawback is not applicable if the cluster is properly configured to use the rack awareness (location awareness) feature.

To add additional tablet servers to an existing cluster, the following steps can be taken to ensure tablet replicas are uniformly distributed across the cluster:

1. Ensure that Kudu is installed on the new machines being added to the cluster, and that the new instances have been correctly configured to point to the pre-existing cluster. Then, start the new tablet server instances.
2. Verify that the new instances check in with the Kudu Master(s) successfully. A quick method for verifying whether they have successfully checked in with the existing Master instances is to view the Kudu Master WebUI, specifically the /tablet-servers section, and validate that the newly added instances are registered, and have a heartbeat.
3. Once the tablet server(s) are successfully online and healthy, follow the steps to run the rebalancing tool which spreads the existing tablet replicas to the newly added tablet servers.
4. After the rebalancer tool has completed, or even during its execution, you can check the health of the cluster using the ksck command-line utility.

Decommission or remove a tablet server

You can decommission or permanently remove a tablet server from a cluster.

About this task

Starting with Kudu 1.12, the Kudu rebalancer tool can be used to decommission a tablet server by supplying the --ignored_tservers and --move_replicas_from_ignored_tservers arguments.

Note: Do not decommission multiple tablet servers at once. To remove multiple tablet servers from the cluster, follow the below instructions for each tablet server, ensuring that the previous tablet server is removed from the cluster and ksck is healthy before shutting down the next.

Procedure

1. Ensure the cluster is in good health using ksck.
2. Put the tablet server into a maintenance mode by using the kudu tserver state enter_maintenance tool.

3. Run the kudu cluster rebalance tool, supplying the --ignored_tservers argument with the UUIDs of the tablet servers to be decommissioned, and the --move_replicas_from_ignored_tservers flag.
4. Wait for the moves to complete and for ksck to show the cluster in a healthy state.

   Note: To verify that all the moves are completed from the server placed in maintenance mode, ensure that there are no replicas on the server by running kudu cluster ksck or running kudu cluster rebalancer with --report_only and --output_replica_distribution_details flags or checking the Web UI of the tablet server.
5. The decommissioned tablet server can be brought offline by stopping the tablet server in Cloudera Manager.
6. To completely remove it from the cluster so ksck shows the cluster as completely healthy, restart the masters.

   If you have only one master in your deployment, this may cause cluster downtime. In a multi-master deployment, restart the masters in sequence to avoid cluster downtime.

Related Information

Minimize cluster distruption during temporary planned downtime of a single tablet server

Use cluster names in the kudu command line tool

When using the kudu command line tool, it can be difficult to remember the precise list of Kudu master RPC addresses needed to communicate with a cluster, especially when managing multiple clusters. As an alternative, you can use the command line tool to identify clusters by name.

Procedure

1. Create a new directory to store the Kudu configuration file.
2. Export the path to this newly created directory in the KUDU_CONFIG environment variable.
3. Create a file called kudurc in the new directory.
4. Populate kudurc as follows, substituting your own cluster names and RPC addresses:

```
clusters_info:
  cluster_name1:
    master_addresses: ip1:port1,ip2:port2,ip3:port3
  cluster_name2:
    master_addresses: ip4:port4
```

5. When using the kudu command line tool, replace the list of Kudu master RPC addresses with the cluster name, prepended with the character @. For example:

```
$ sudo -u kudu kudu cluster @cluster_name1
```

   Note: Cluster names may be used as input in any invocation of the kudu command line tool that expects a list of Kudu master RPC addresses.

Migrate Kudu data from one directory to another on the same host

Take the following steps to move the entire Kudu data from one directory to another.

About this task

Note: The steps were verified on an environment where the master and the server instances were configured to write the WAL/Data to the same directory.

Procedure

1. Stop the Kudu service.
2. Modify the directory configurations for the Master/Server instances.
3. Move the existing data from the old directory, to the new one.
4. Make sure the file/directory ownership is set to the kudu user.
5. Restart the Kudu service.
6. Run ksck and verify for the healthy status.

Related Information

Changing directory configuration

Migrate to a multiple Kudu master configuration

Before migrating to a multiple Kudu masters set up, you need to perform many migration planning steps, such as deciding the number of masters, and choosing the nodes where to add the new Kudu masters.

About this task

The migration procedure does not require stopping all the Kudu processes. The restarting of the Kudu processes can be done without incurring downtime.

Important: The procedure does not support adding multiple masters at the same time by selecting multiple hosts. It is only supported to add one by one.

Before you begin

- You must decide how many masters to use.

  The number of masters should be odd because an even number of masters does not provide any benefit over having one fewer masters. Three or five node master configurations are recommended as they can tolerate the failure of one or two masters respectively.
- You must establish a maintenance window.

  One hour should be sufficient maintenance window time. During this time the Kudu cluster might become unavailable if there is some problem during the procedure.

Procedure

1. Configure a DNS alias for the new master.

   The alias can be:

   - a CNAME record: if the machine already has an A record in DNS
   - and A record: if the machine is only known by its IP address

   Important: Without DNS aliases, it is not possible to recover from permanent master failures without restarting the masters and tablet servers in the cluster to pick up the replacement master node with a different hostname. It is highly recommended that you use DNS aliases.
2. Perform the following preparatory steps for the new master that you are planning to add:

   a) Choose a node in the cluster where there is no running Kudu master yet and which has enough spare CPU and memory capacity to host a new Kudu master.

      The master generates very little load so it can be collocated with other data services or load-generating processes, though not with another Kudu master from the same configuration.

   b) Choose and record the directories where the new master's data and WAL will be located.

   c) Choose and record the port the master should use for RPCs.

3. In Cloudera Manager, add a new Kudu Master role to the selected new master node:
   a) Select the Kudu service.
   b) Select Instances.
   c) Click Add Role Instances.
   d) Click Select hosts under Master × 1.
   e) Select the host node and click OK.
   f) Click Continue.
   g) Review the changes and if everything is correct, click Finish.

   Now, the newly added master role instance is commissioned but not part of the cluster. To make it a part of the cluster, you need to start it.
4. Start the newly added master role instance.
   a) Select the master role instance.
   b) Go to Actions for Selected Start .

   Upon starting, the new Kudu master registers with the existing master(s).
5. Review the log files to make sure no errors have been reported, and click Close.

   Now, the newly added master role instance is a part of the cluster.
6. To propagate the new master membership configuration, restart all Kudu masters (the old ones and the newly added one).

   It can be done on a one-by-one (i.e. Rolling Restart) or all-at-once manner.
7. After all the Kudu masters are up and running, open the Web UI of the newly added master and click the Masters tab.

   A page with **Live Masters** table appears.
8. Verify the following:
   - The table shows all the Kudu masters; the old ones and the newly added one.
   - There is one master marked as LEADER and the rest are marked as FOLLOWER in the Role column.
9. In the Live Masters table, click the UUID link of the leader Kudu master in the UUID column of the corresponding row.

   The Web UI of the current leader master appears.
10. In the Web UI of the current leader master, click the Masters tab.

    A page with Live Masters table appears.
11. Verify the following:
    - The table shows all the Kudu masters in the cluster; the old ones and the newly added one.
    - The information in the Registration column makes sense for the newly added Kudu master: the entries in the rpc_addresses field RPC match the hostnames and/or IP addresses of the nodes hosting KUDU_MASTER roles.
    - The information in the Start time column reflects the time of the recent restart performed at step 6.

    Note: Now, the newly added Kudu master role instance is a part of the cluster. To add one more Kudu master into the cluster, repeat steps 1-11.

What to do next

1. If you have Kudu tables that are accessed from Impala and you did not set up DNS aliases, update the HMS database manually in the underlying database that provides the storage for HMS:
   a. Connect to the HMS database.
   b. Run an SQL statement similar to the following example:

```
UPDATE TABLE_PARAMS
SET PARAM_VALUE =
```

```
  'master-1.example.com,master-2.example.com,master-3.example.com'
WHERE PARAM_KEY = 'kudu.master_addresses' AND PARAM_VALUE = 'master-1.e
xample.com';
```

   c. In impala-shell run the following command: INVALIDATE METADATA;

2. After adding all the desired masters into the cluster, modify the value of the tserver_master_addrs configuration parameter for each tablet server. The new value must be a comma-separated list of masters where each entry is a string of the form <hostname>:<port>, where
   - hostname is master's hostname
   - port is master's RPC port number
3. Restart all the tablet servers to pick up the new masters' configuration.
4. To verify that all masters are working properly, consider performing the following checks:
   - Using a browser, visit each master's web UI and navigate to the /masters page.

     All the masters should be listed there with one master in the LEADER role and the others in the FOLLOWER role. The contents of /masters on each master should be the same.
   - Run a Kudu system check (ksck) on the cluster using the kudu command line tool.
   - If applicable, run a few quick SQL queries against a couple of migrated Kudu tables using impala-shell or Hue.

Change master hostnames

When replacing dead masters, use DNS aliases to prevent long maintenance windows. If the cluster was set up without aliases, change the host names as described in this section.

Prepare for master hostname changes

In this step, you need to identify a down-time window, and note the UUID and the RPC address of each master.

Procedure

1. Establish a maintenance window during which the Kudu cluster will be unavailable. One hour should be sufficient.
2. On the **Masters** page in Kudu Web UI, note the UUID and RPC address of each master.
3. Stop all the Kudu processes in the cluster.
4. Set up the new hostnames to point to the masters and verify all servers and clients properly resolve them.

Perform master hostname changes

You need to bring the Kudu clusters down to update the master hostnames. Therefore, identify at least a one-hour maintenance window for this task.

Procedure

1. Rewrite each master's Raft configuration with the following command, executed on each master host:

```
$ sudo -u kudu kudu local_replica cmeta rewrite_raft_config --fs_wal_dir
=<master_wal_dir> [--fs_data_dirs=<master_data_dir>] 0000000000000000000
0000000000000 <all_masters>
```

   For example:

```
$ sudo -u kudu kudu local_replica cmeta rewrite_raft_config --fs_wal_dir=/
data/kudu/master/wal --fs_data_dirs=/data/kudu/master/data 000000000000000
00000000000000000 4aab798a69e94fab8d77069edff28ce0:new-master-name-1:705
1 f5624e05f40649b79a757629a69d061e:new-master-name-2:7051 988d8ac6530f42
6cbe180be5ba52033d:new-master-name-3:7051
```

2. Update the master address:
   - In an environment not managed by Cloudera Manager, change the gflag file of the masters so the master_addre sses parameter reflects the new hostnames.
   - In an environment managed by Cloudera Manager, specify the new hostname in the Master Address (server.address) field on each Kudu role.
3. Change the gflag file of the tablet servers to update the tserver_master_addrs parameter with the new hostnames. In an environment managed by Cloudera Manager, this step is not needeed.
4. Start the masters.
5. To verify that all masters are working properly, perform the following checks:
   a) In each master's Web UI, click Masters on the Status Pages. All of the masters should be listed there with one master in the LEADER role field and the others in the FOLLOWER role field. The contents of Masters on all master should be the same.
   b) Run the below command to verify all masters are up and listening. The UUIDs are the same and belong to the same master as before the hostname change:

```
$ sudo -u kudu kudu master list new-master-name-1:7051,new-master-name-2
:7051,new-master-name-3:7051
```

6. Start all of the tablet servers.
7. Run a Kudu system check (ksck) on the cluster using the kudu command line tool. After startup, some tablets may be unavailable as it takes some time to initialize all of them.
8. If you have Kudu tables that are accessed from Impala, update the HMS database manually in the underlying database that provides the storage for HMS.
   a) The following is an example SQL statement you run in the HMS database:

```
UPDATE TABLE_PARAMSSET PARAM_VALUE =
'new-master-name-1:7051,new-master-name-2:7051,new-master-name-3:7051'
WHERE PARAM_KEY = 'kudu.master_addresses'
AND PARAM_VALUE = 'master-1:7051,master-2:7051,master-3:7051';
```

   b) In impala-shell, run:

```
INVALIDATE METADATA;
```

   c) Verify updating the metadata worked by running a simple SELECT query on a Kudu-backed Impala table.

Related Information

Monitoring cluster health with ksck

Removing Kudu masters through Cloudera Manager

In the event that a multi-master deployment has overallocated nodes, you can remove or decommission the unwanted masters through Cloudera Manager.

About this task

You can not decommission a leader master role from the cluster. You can only decommission a follower master role. If you want to decommission multiple master roles, do it one by one.

Note:

- In planning the new multi-master configuration, keep in mind that the number of masters should be odd and that three or five node master configurations are recommended.
- Dropping the number of masters below the number of masters currently needed for a Raft majority can incur data loss. To mitigate this, ensure that the leader master is not removed during this process.

Before you begin

You must have multiple master roles added in your cluster.

Procedure

1. In Cloudera Manager, go to Kudu Instances .
2. Select the master role instance you want to decommission, and click Actions for Selected Decommission .
3. Review the changes in log files, and click Close.

   Now, the master role instance is decommissioned from the cluster.

Recommissioning Kudu masters through Cloudera Manager

You can recommission any decommissioned master role instance in a multi-master deployment and include the instance in the cluster.

Before you begin

You must have decommissioned a master role instance.

Procedure

1. In Cloudera Manager, go to Kudu Instances .
2. Select the master role instance you want to recommission, and click Actions for Selected Recommission .
3. Review the changes in log files, and click Close.

   Now, the recommissioned master role instance is commissioned but is not part of the cluster. To make it a part of the cluster, you need to start it.
4. Start the newly added master role instance or instances.
   a) Select the master role instance.
   b) Click Actions for Selected Start .
5. Review the changes in log files, and click Close.

   Now, the recommissioned master role instance is a part of the cluster.

Remove Kudu masters through CLI

In the event that a multi-master deployment has been overallocated nodes, the following steps should be taken to remove the unwanted masters through CLI.

Important:

- In planning the new multi-master configuration, keep in mind that the number of masters should be odd and that three or five node master configurations are recommended.
- Dropping the number of masters below the number of masters currently needed for a Raft majority can incur data loss. To mitigate this, ensure that the leader master is not removed during this process.

Prepare for removal

In order to remove the unwanted masters from a multi-master deployment, you need to identify them and note their UUID and RPC addresses.

Procedure

1. Establish a maintenance window (one hour should be sufficient). During this time the Kudu cluster will be unavailable.
2. Identify the UUID and RPC address current leader of the multi-master deployment by visiting the /masters page of any master's web UI. This master must not be removed during this process; its removal may result in severe data loss.
3. Stop the unwanted Kudu master processes.

Perform the removal

When you remove any Kudu masters from a multi-master deployment, you need to rewrite the Raft configuration on the remaining masters, remove data and WAL directories from the unwanted masters, and finaly modify the value of the tserver_master_addrs configuration parameter for the tablet servers to remove the unwanted masters. You need to bring the Kudu clusters down. Therefore, identify at least a one-hour maintenance window for this task.

Procedure

1. Perform the Raft configuration change by running the kudu master remove tool.

   Only a single master can be removed at a time. If multiple masters need to be removed, run the tool multiple times. In the following example master-2 is being removed from a Kudu cluster with two masters master-1, and master-2:

   ```
   $ sudo -u kudu kudu master remove master-1,master-2 master-2
   ```

2. Remove the data directories and WAL directory on the unwanted masters. This is a precaution to ensure that they cannot start up again and interfere with the new multi-master deployment.
3. Modify the value of the master_addresses configuration parameter for the masters of the new multi-master deployment.
4. Restart all of the masters that were not removed.
5. If you are not using Cloudera Manager: Modify the value of the tserver_master_addrs configuration parameter for the tablet servers to remove any unwanted masters.
6. Retart all of the tablet servers.

What to do next

To verify that all masters are working properly, consider performing the following checks:

- Using a browser, visit each master's web UI and navigate to the /masters page. All the masters should now be listed there with one master in the LEADER role and the others in the FOLLOWER role. The contents of /masters on each master should be the same.
- Run a Kudu system check (ksck) on the cluster using the kudu command line tool.

Related Information

Configure Kudu processes

Monitoring cluster health with ksck

How Range-aware replica placement in Kudu works

Learn how the range-aware replica placement algorithm works during initial tablet replica placement in Kudu tablet servers.

Kudu places new tablet replicas using an algorithm which is both range and table aware. This algorithm helps to avoid hotspotting that occurs if many replicas from the same range are placed on the same few tablet servers. Hotspotting causes tablet servers to be overwhelmed with write or read requests and can result in increased latency for these requests. To avoid hotspotting, this algorithm avoids targeting the same set of tablet servers for a set of replicas created in parallel. Rather, it spreads the replicas across multiple tablet servers.

This algorithm is independent of the location. Replica placement can be broken down into two steps; location selection and tablet server placement.

The range-aware algorithm works in the following way:

1. It ranks the tablet servers (TS) according to number of replicas per corresponding range (Range X in the following image).

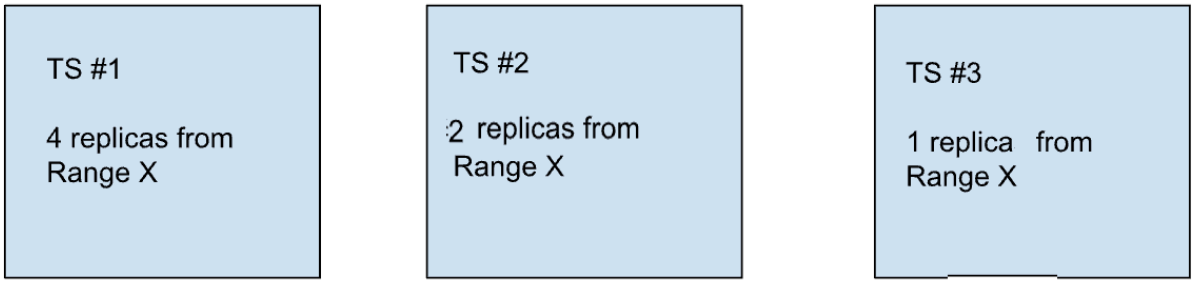

2. It places the replica on the tablet server with the least amount of replicas from the corresponding range.

If two tablet servers have the same number of replicas from a range, the number of replicas from the table the range belongs to, is used as a tiebreaker. If the tablet servers contain the same number of replicas from the table as well, the number of total replicas hosted by the tablet servers is used as a final tiebreaker.

When determining the number of replicas of a range or table per tablet server, the algorithm uses two separate metrics. The first metric is the number of existing live replicas from the range or table. The second metric is the number of pending replicas from the range or table being placed. This metric is incremented when a pending replica is placed, but eventually the value decays to zero.

Replicas by range are stored with the table the range belongs to, in case multiple tables have ranges with the same range start keys. If the table is not stored along with the range, the algorithm can not differentiate between two ranges with the same start keys but on different tables.

Important: Both the range start key and the table ID for a certain range must be defined for this feature to work on that range.

The power of two choices algorithm, used earlier, was quick and efficient but did not prevent hotspotting.

Run the tablet rebalancing tool

The kudu CLI contains a rebalancing tool that can be used to rebalance tablet replicas among tablet servers. For each table, the tool attempts to balance the number of replicas per tablet server. It also, without unbalancing any table, attempts to even out the number of replicas per tablet server across the cluster as a whole.

The rebalancing tool should be run as the Kudu admin user, specifying all master addresses:

```
sudo -u kudu kudu cluster rebalance master-01.example.com,master-02.example.
com,master-03.example.com
```

When run, the rebalancer reports on the initial tablet replica distribution in the cluster, logs the replicas it moves, and prints a final summary of the distribution when it terminates:

```
Per-server replica distribution summary:
       Statistic       |   Value
-----------------------+-----------
 Minimum Replica Count | 0
 Maximum Replica Count | 24
 Average Replica Count | 14.400000
Per-table replica distribution summary:
 Replica Skew |  Value
--------------+----------
 Minimum      | 8
 Maximum      | 8
 Average      | 8.000000

I0613 14:18:49.905897 3002065792 rebalancer.cc:779] tablet e7ee9ade95b342a7a
94649b7862b345d: 206a51de1486402bbb214b5ce97a633c -> 3b4d9266ac8c45ff9a5d4d7
c3e1cb326 move scheduled
I0613 14:18:49.917578 3002065792 rebalancer.cc:779] tablet 5f03944529f44626
a0d6ec8b1edc566e: 6e64c4165b864cbab0e67ccd82091d60 -> ba8c22ab030346b4baa289
d6d11d0809 move scheduled
I0613 14:18:49.928683 3002065792 rebalancer.cc:779] tablet 9373fee3bfe74ce
c9054737371a3b15d: fab382adf72c480984c6cc868fdd5f0e -> 3b4d9266ac8c45ff9a5d4
d7c3e1cb326 move scheduled

... (full output elided)

I0613 14:19:01.162802 3002065792 rebalancer.cc:842] tablet f4c046f18b174cc
2974c65ac0bf52767: 206a51de1486402bbb214b5ce97a633c -> 3b4d9266ac8c45ff9a5d4
d7c3e1cb326 move completed: OK

rebalancing is complete: cluster is balanced (moved 28 replicas)
Per-server replica distribution summary:
       Statistic       |   Value
-----------------------+-----------
 Minimum Replica Count | 14
 Maximum Replica Count | 15
 Average Replica Count | 14.400000

Per-table replica distribution summary:
 Replica Skew |  Value
--------------+----------
 Minimum      | 1
 Maximum      | 1
 Average      | 1.000000
```

If more details are needed in addition to the replica distribution summary, use the --output_replica_distribution_details flag. If added, the flag makes the tool print per-table and per-tablet server replica distribution statistics as well.

Use the --report_only flag to get a report on table-wide and cluster-wide replica distribution statistics without starting any rebalancing activity.

The rebalancer can also be restricted to run on a subset of the tables by supplying the --tables flag. Note that, when running on a subset of tables, the tool does not attempt to balance the cluster as a whole.

The length of time rebalancing is run for can be controlled with the flag --max_run_time_sec. By default, the rebalancer runs until the cluster is balanced. To control the amount of resources devoted to rebalancing, modify the flag --max_moves_per_server. See kudu cluster rebalance --help for more.

It is safe to stop the rebalancer tool at any time. When restarted, the rebalancer continues rebalancing the cluster.

The rebalancer tool requires all registered tablet servers to be up and running to proceed with the rebalancing process in order to avoid possible conflicts and races with the automatic re-replication and to keep replica placement optimal for current configuration of the cluster. If a tablet server becomes unavailable during the rebalancing session, the rebalancer exits. As noted above, it is safe to restart the rebalancer after resolving the issue with unavailable tablet servers. However, if it is necessary to rebalance the cluster when a few tablet servers in a Kudu cluster are not available, it is possible to specify their UUIDs as a comma-separated list with the --\ignored_tservers flag and rebalance the rest of the cluster. With the --ignored_tservers flag, the specified tablet servers are effectively ignored by the rebalancer tool, which means they are not considered as a part of the cluster along with tablet replicas they host.

The rebalancing tool can rebalance Kudu clusters running older versions as well, with some restrictions. Consult the following table for more information. In the table, "RF" stands for "replication factor".

Version Range	Rebalances RF = 1 Tables?	Rebalances RF > 1 Tables?
v < 1.4.0	No	No
1.4.0 <= v < 1.7.1	No	Yes
v >= 1.7.1	Yes	Yes

If the rebalancer is running against a cluster where rebalancing replication factor one tables are not supported, it rebalances all the other tables and the cluster as if those singly-replicated tables did not exist.

Range-aware rebalancing

You can use the Kudu cluster rebalance CLI tool to run range-aware rebalancing. Range-aware rebalancing runs on a per-table basis, which means that the rebalancing can be run on one table at a time. To enable range-aware rebalancing for a particular table, you need to add the following flags when using the CLI tool:

- --enable_range_rebalancing
- --tables=<table_name_for_range_aware_rebalancing>

To perform cluster-wide rebalancing, it is recommended to run the kudu cluster rebalance tool. In addition to the cluster-wide rebalancing, you can run the range-aware rebalancing for larger tables in the cluster that are multilevel-partitioned, and can suffer from the hot-spotting issue.

The default replica placement algorithm is range-aware.

Run a tablet rebalancing tool on a rack-aware cluster

It is possible to use the kudu cluster rebalance tool to establish the placement policy on a cluster. This might be necessary when the rack awareness feature is first configured or when re-replication violated the placement policy.

About this task

The rebalancing tool breaks its work into three phases:

Procedure

1. The rack-aware rebalancer tries to establish the placement policy. Use the ##disable_policy_fixer flag to skip this phase.
2. The rebalancer tries to balance load by location, moving tablet replicas between locations in an attempt to spread tablet replicas among locations evenly. The load of a location is measured as the total number of replicas in the location divided by the number of tablet servers in the location. Use the ##disable_cross_location_rebalancing flag to skip this phase.
3. The rebalancer tries to balance the tablet replica distribution within each location, as if the location were a cluster on its own. Use the ##disable_intra_location_rebalancing flag to skip this phase.

Note: Each of the above rebalancing phases can be independently skipped using the corresponding flags.

What to do next

By using the ##report_only flag, it's also possible to check if all tablets in the cluster conform to the placement policy without attempting any replica movement.

Example of rebalancing tool with rack awareness functionality

The behavior of each of these flags is explained through the following examples.

Consider three locations and three tablets (a total of 9 replicas) while running the tool with the flags below (placement policy fixer):

```
--disable_cross_location_rebalancing --disable_intra_location_rebalancing
```

Before running the tool:

Location A	Location B	Location C
Replica X	Replica Y	Replica Z
Replica X	Replica Y	Replica Z
Replica X	Replica Y	Replica Z

After running the tool with the flags:

Location A	Location B	Location C
Replica X	Replica X	Replica X
Replica Y	Replica Y	Replica Y
Replica Z	Replica Z	Replica Z

The replicas of every tablet are now distributed across all three locations.

Consider the following tablet distribution and run the tool with the flags below (cross-location rebalancing):

```
--disable_policy_fixer --disable_intra_location_rebalancing
```

Before running the tool:

Location	Number of replicas across all tables in the location
A (5 tablet servers)	15

Location	Number of replicas across all tables in the location
B (5 tablet servers)	18
C (5 tablet servers)	21

After running the tool with the flags:

Location	Number of replicas across all tables in the location
A (5 tablet servers)	21
B (5 tablet servers)	21
C (5 tablet servers)	21

The number of replicas in each of the locations is now equal.

Let's analyze Location A before and after using the tool with the flags (intra-location rebalancing):

```
--disable_policy_fixer --disable_cross_location_rebalancing
```

Before running the tool:

Tablet server (TS)	Number of replicas across all tables in the server
TS_1	3
TS_2	5
TS_3	8
TS_4	4
TS_5	1

After running the tool with the flags:

Tablet server (TS)	Number of replicas across all tables in the server
TS_1	4
TS_2	5
TS_3	4
TS_4	4
TS_5	4

The number of replicas in each tablet server is now balanced.

Run a tablet rebalancing tool in Cloudera Manager

You access and run the tablet rebalancing tool from Cloudera Manager.

Procedure

1. Browse to Clusters Kudu .
2. Click Actions and select Run Kudu Rebalancer Tool.

Results

In Cloudera Manager, the rebalancer runs with the default flags.

Run a tablet rebalancing tool in command line

Learn how to use Kudu service keytab while running the Kudu tablet rebalancing tool.

About this task

The following instructions are applicable when Kerberos is enabled in the cluster. If Kerberos is not enabled, you need to only run the following command:

```
sudo -u kudu kudu cluster rebalance master-01.example.com,master-02.example.
com,master-03.example.com
```

Procedure

1. Run kinit as Kudu user through keytab.

```
$ kinit -kt /var/run/cloudera-scm-agent/process/$(ls -rt /var/run/cloude
ra-scm-agent/process|grep KUDU|tail -n1)/kudu.keytab kudu/`hostname -f`
```

2. Check if kinit ran successfully.

```
$ klist
```

3. Run the rebalancing tool as the Kudu admin user, specifying all master addresses.

```
kudu cluster rebalance master-01.example.com,master-02.example.com,maste
r-03.example.com
```

Managing Kudu tables with range-specific hash schemas

The custom hash partition feature enables you to vary the number of hash buckets per range partition, both at table creation and alteration time.

It is possible to specify a different number of hash buckets per range partition when creating or altering a table. This allows for accommodating new ranges of already existing Kudu tables to changes in workload patterns. For example, for a table range-partitioned by date or timestamp it makes sense to increase the number of hash buckets with an increase in ingestion rate to maximize write throughput.

When no hash schema per range is specified for a table's range either during creation or alteration time, it defaults to the table-wide hash schema. If no table-wide hash schema is specified when the table is created, no hash bucketing for the data is done at all.

Any hash schema specified for a range partition overrides the table-wide hash schema. However, the number of hash dimensions in any range-specific hash schema must be the same as in the table-wide hash schema.

The newly introduced range-specific hash schemas in Kudu are supported by impala-shell.

Range-specific hash schemas example: Using impala-shell

Review examples of using impala-shell to create and alter Kudu tables with range-specific hash schemas.

The following is a few examples of creating a new table with custom hash schemas for some of its range partitions.

The table-wide hash schema for table t1 has two hash dimensions (hash bucketing by the ‘id’ and the ‘name’ columns independently):

```
CREATE TABLE t1 (id INT, name STRING, PRIMARY KEY(id, name))
PARTITION BY HASH(id) PARTITIONS 3 HASH(name) PARTITIONS 4
RANGE (id)
(
    // hash partition by “id” and “name” as in the table-wide hash schema,
    // but customize the number of hash buckets in each dimension
    PARTITION 15 <= VALUES < 20 HASH(id) PARTITIONS 5 HASH(name) PARTITIONS
 3,
    // this range has the table-wide hash schema
    PARTITION 20 <= VALUES < 25
)
```

The table-wide hash schema for table t2 has one hash dimension (hash bucketing by the ‘name’ column):

```
CREATE TABLE t2 (id INT, name STRING, PRIMARY KEY(id, name))
PARTITION BY HASH(name) PARTITIONS 3
RANGE (id)
(
    // hash partition by “name” but use custom number of hash buckets
    PARTITION  5 <= VALUES < 10 HASH(name) PARTITIONS 5,

    // hash partition by “id”
    PARTITION 10 <= VALUES < 15 HASH(id) PARTITIONS 2,
    // hash partition by “id” and “name”
    PARTITION 15 <= VALUES < 20 HASH(id, name) PARTITIONS 8,

    // this range has the same schema as the range [15, 20) above:
    // hash partition by the primary key (i.e. by (id, name) pair) into 8
 buckets
    PARTITION 20 <= VALUES < 25 PARTITIONS 8,

    // using the table-wide hash schema if no override is specified
    PARTITION 25 <= VALUES < 30
)
```

The following are examples of adding new range partitions with custom hash schemas to a table that already exists:

```
// t2: add new range partition by “id”, hashed by “id” and “name” together
ALTER TABLE t2 ADD RANGE PARTITION -5 <= VALUES < 0 HASH PARTITIONS 10;

// t1: add new range partition by “id”, additionally hashed by “id” and “nam
e” separately
// (defaults to the table-wide hash schema since no range-specific hash sc
hema specified)
ALTER TABLE t1 ADD RANGE PARTITION 25 <= VALUES < 30

// t2: add new range partition by “id”, additionally hashed by “name”
ALTER TABLE t2 ADD RANGE PARTITION 30 <= VALUES < 40 HASH(name) PARTITIONS 5
;

// t1: add new range partition by “id”, additionally hashed by “id” and “
name” separately
ALTER TABLE t1 ADD RANGE PARTITION 40 <= VALUES < 50 HASH(id) PARTITIONS 10,
 HASH(name) PARTITIONS 5;
```

Range-specific hash schemas example: Using Kudu C++ client API

Review an example of using Kudu C++ client API to create a table with a range partition having range-specific hash schema and then alter the table, adding one more range with custom hash schema.

Note: After adding or dropping ranges with custom hash schemas you have to refresh the KuduTable handle.

```
 // Define the column schema for the table.
  KuduSchema schema;
  {
    KuduSchemaBuilder b;
    b.AddColumn("key")->Type(KuduColumnSchema::INT32)->NotNull()->PrimaryK
ey();
    b.AddColumn("int_col")->Type(KuduColumnSchema::INT32)->NotNull();
    b.AddColumn("string_col")->Type(KuduColumnSchema::STRING);
    RETURN_NOT_OK(b.Build(&schema));
  }
  unique_ptr<KuduTableCreator> tc(client->NewTableCreator());
  // The table is range-partitioned by the ‘key’ column and has table-wide
 hash schema:
  // one-dimensional hash bucketing into two buckets by the ‘key’ column.
  table_creator->table_name("test")
      .schema(&schema)
      .add_hash_partitions({ "key" }, 2)
      .set_range_partition_columns({ "key" });
  // Add [0, 100) range partition with custom hash schema:
  // 5 buckets with hash based on the "key" column with hash seed 8.
  {
    unique_ptr<KuduPartialRow> lower(schema.NewRow());
    RETURN_NOT_OK(lower->SetInt32("key", 0));

    unique_ptr<KuduPartialRow> upper(schema.NewRow());
    RETURN_NOT_OK(upper->SetInt32("key", 100));

    unique_ptr<KuduRangePartition> p(
        new KuduRangePartition(lower.release(), upper.release()));
    RETURN_NOT_OK(p->add_hash_partitions({ "key" }, 5, 8));
    tc->add_custom_range_partition(p.release());
  }

  // Create the ‘test’ table with the specified column schema
  // and one range partition.
  RETURN_NOT_OK(tc->Create());

  // Open the table and work with it.
  shared_ptr<KuduTable> table;
  RETURN_NOT_OK(client->OpenTable("test", &table));
  . . .

  // Alter the table, adding a new range [100, 200) with range-specific hash
  // schema: 3 buckets with hash based on the "key" column with hash seed 1.
  {
    unique_ptr<KuduPartialRow> lower(schema.NewRow());
    RETURN_NOT_OK(lower->SetInt32("key", 222));

    unique_ptr<KuduPartialRow> upper(schema.NewRow());
    RETURN_NOT_OK(upper->SetInt32("key", 333));

    unique_ptr<KuduRangePartition> p(
```

```
        new KuduRangePartition(lower.release(), upper.release()));
    RETURN_NOT_OK(p->add_hash_partitions({ "key" }, 3, 1));

    unique_ptr<KuduTableAlterer> ta(client->NewTableAlterer(kTableName));
    ta->AddRangePartition(p.release());
    RETURN_NOT_OK(ta->Alter());
  }

  // NOTE: after adding or dropping a range with custom hash schemas
  //       it's necessary to refresh the KuduTable handle
  RETURN_NOT_OK(client->OpenTable("test", &table));

  // Continue working with the table.
  . . .
```

Range-specific hash schemas example: Using Kudu Java client API

Review an example of using Kudu Java client API to create a table with a range partition having range-specific hash schema and then alter the table, adding one more range with custom hash schema.

Note: After adding or dropping ranges with custom hash schemas you have to refresh the KuduTable handle.

```
  // Define the column schema.
  ArrayList<ColumnSchema> columns = new ArrayList<>(2);
  columns.add(new ColumnSchema.ColumnSchemaBuilder(
      "key", Type.INT32).key(true).build());
  columns.add(new ColumnSchema.ColumnSchemaBuilder(
      "col", Type.INT64).build());
  final Schema schema = new Schema(columns);
  CreateTableOptions builder = new CreateTableOptions();

  // Do range partitioning on the "key" column.
  builder.setRangePartitionColumns(ImmutableList.of("key"));
  // Define the table-wide schema: two buckets on the "key" column.
  builder.addHashPartitions(ImmutableList.of("key"), 2, 0);

  // Add [0, 100) range partition with custom hash schema.
  {
    PartialRow lower = schema.newPartialRow();
    lower.addInt("key", 0);
    PartialRow upper = schema.newPartialRow();
    upper.addInt("key", 100);

    // Custom hash schema for the range: five buckets on the "key" column.
    RangePartitionWithCustomHashSchema rangePartition =
        new RangePartitionWithCustomHashSchema(
            lower,
            upper,
            RangePartitionBound.INCLUSIVE_BOUND,
            RangePartitionBound.EXCLUSIVE_BOUND);
    rangePartition.addHashPartitions(ImmutableList.of("key"), 5, 0);
    builder.addRangePartition(rangePartition);
  }

  // Create the table.
  KuduTable table = client.createTable("mytable", schema, builder);

  // Work with the table.
```

```
  . . .

  // Alter the table: add [100, 200) range partition with custom hash sche
ma.
  {
    PartialRow lower = schema.newPartialRow();
    lower.addInt("key", 100);
    PartialRow upper = schema.newPartialRow();
    upper.addInt("key", 200);
    RangePartitionWithCustomHashSchema range =
        new RangePartitionWithCustomHashSchema(
            lower,
            upper,
            RangePartitionBound.INCLUSIVE_BOUND,
            RangePartitionBound.EXCLUSIVE_BOUND);
    range.addHashPartitions(ImmutableList.of("key"), 3, 0);
    client.alterTable("mytable",
        new AlterTableOptions().addRangePartition(range));
  }

  // NOTE: after adding or dropping a range with custom hash schemas
  //       it’s necessary to refresh the KuduTable handle
  table = client.openTable("mytable");

  // Continue working with the table.
  . . .
```

Monitoring Apache Kudu

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Kudu metrics **4**
Listing available metrics 4
Collecting metrics through HTTP 4
Diagnostics logging 5

Monitor cluster health with ksck **6**

Report Kudu crashes using breakpad **7**

Enable core dump for the Kudu service **8**

Use the Charts Library with the Kudu service **8**

Kudu metrics

Kudu daemons expose a large number of metrics. Some metrics are associated with an entire server process, whereas others are associated with a particular tablet replica.

Listing available metrics

The full set of available metrics for a Kudu server can be dumped using a special command line flag:

```
$ kudu-tserver --dump_metrics_json
$ kudu-master --dump_metrics_json
```

This will output a large JSON document. Each metric indicates its name, label, description, units, and type. Because the output is JSON-formatted, this information can easily be parsed and fed into other tooling which collects metrics from Kudu servers.

For the complete list of metrics collected by Cloudera Manager for a Kudu service, look for the Kudu metrics listed under *Cloudera Manager Metrics*.

Related Information

Cloudera Manager Metrics

Collecting metrics through HTTP

Metrics can be collected from a server process via its HTTP interface by visiting /metrics. The output of this page is JSON for easy parsing by monitoring services. This endpoint accepts several GET parameters in its query string:

- /metrics?metrics=<substring1>,<substring2>,… - Limits the returned metrics to those which contain at least one of the provided substrings. The substrings also match entity names, so this may be used to collect metrics for a specific tablet.
- /metrics?include_schema=1 - Includes metrics schema information such as unit, description, and label in the JSON output. This information is typically omitted to save space.
- /metrics?compact=1 - Eliminates unnecessary whitespace from the resulting JSON, which can decrease bandwidth when fetching this page from a remote host.
- /metrics?include_raw_histograms=1 - Include the raw buckets and values for histogram metrics, enabling accurate aggregation of percentile metrics over time and across hosts.
- /metrics?level=info - Limits the returned metrics based on their severity level. The levels are ordered and lower levels include the levels above them. If no level is specified, debug is used to include all metrics. The valid values are:
  - debug - Metrics that are diagnostically helpful but generally not monitored during normal operation.
  - info - Generally useful metrics that operators always want to have available but may not be monitored under normal circumstances.
  - warn - Metrics which can often indicate operational oddities, which may need more investigation.
- /metrics?types=type1,type2 - Limits the returned metrics based on their types.
- /metrics?ids=id1,id2 - Limits the returned metrics based on their ids.
- /metrics?attributes=table_name,table1,table_name,table2 - Limits the returned metrics based on their attributes.

For example:

```
$ curl -s 'http://example-ts:8050/metrics?include_schema=1&metrics=connectio
ns_accepted'
```

```
[
    {
        "type": "server",
        "id": "kudu.tabletserver",
        "attributes": {},
        "metrics": [
            {
                "name": "rpc_connections_accepted",
                "label": "RPC Connections Accepted",
                "type": "counter",
                "unit": "connections",
                "description": "Number of incoming TCP connections made to
the RPC server",
                "value": 92
            }
        ]
    }
]
```

```
$ curl -s 'http://example-ts:8050/metrics?metrics=log_append_latency'
```

```
[
    {
        "type": "tablet",
        "id": "c0ebf9fef1b847e2a83c7bd35c2056b1",
        "attributes": {
            "table_name": "lineitem",
            "partition": "hash buckets: (55), range: [(<start>), (<end>))",
            "table_id": ""
        },
        "metrics": [
            {
                "name": "log_append_latency",
                "total_count": 7498,
                "min": 4,
                "mean": 69.3649,
                "percentile_75": 29,
                "percentile_95": 38,
                "percentile_99": 45,
                "percentile_99_9": 95,
                "percentile_99_99": 167,
                "max": 367244,
                "total_sum": 520098
            }
        ]
    }
]
```

Diagnostics logging

Kudu may be configured to periodically dump all of its metrics to a local log file using the --metrics_log_interv al_msflag. Set this flag to the interval at which metrics should be written to a diagnostics log file.

The diagnostics log will be written to the same directory as the other Kudu log files, with a similar naming format, substituting diagnostics instead of a log level like INFO. After any diagnostics log file reaches 64MB uncompressed, the log will be rolled and the previous file will be gzip-compressed.

The log file generated has three space-separated fields. The first field is the word metrics. The second field is the current timestamp in microseconds since the Unix epoch. The third is the current value of all metrics on the server, using a compact JSON encoding. The encoding is the same as the metrics fetched via HTTP described above.

Monitor cluster health with ksck

The kudu CLI includes a tool called ksck that can be used for gathering information about the state of a Kudu cluster, including checking its health. ksck will identify issues such as under-replicated tablets, unreachable tablet servers, or tablets without a leader.

ksck should be run from the command line as the Kudu admin user, and requires the full list of master addresses to be specified:

```
$ sudo -u kudu kudu cluster ksck master-01.example.com,master-02.example.com
,master-03.example.com
```

To see a full list of the options available with ksck, use the --help flag. If the cluster is healthy, ksck will print information about the cluster, a success message, and return a zero (success) exit status.

```
Master Summary
               UUID                |       Address         | Status
-----------------------------------+-----------------------+---------
 a811c07b99394df799e6650e7310f282 | master-01.example.com | HEALTHY
 b579355eeeea446e998606bcb7e87844 | master-02.example.com | HEALTHY
 cfdcc8592711485fad32ec4eea4fbfcd | master-02.example.com | HEALTHY

Tablet Server Summary
               UUID                |        Address         | Status
-----------------------------------+------------------------+---------
 a598f75345834133a39c6e51163245db | tserver-01.example.com | HEALTHY
 e05ca6b6573b4e1f9a518157c0c0c637 | tserver-02.example.com | HEALTHY
 e7e53a91fe704296b3a59ad304e7444a | tserver-03.example.com | HEALTHY

Version Summary
 Version |      Servers
---------+-------------------------
  1.7.1  | all 6 server(s) checked

Summary by table
   Name    | RF | Status  | Total Tablets | Healthy | Recovering | Under-rep
licated | Unavailable
----------+----+---------+---------------+---------+------------+---------
---------+-------------
 my_table | 3  | HEALTHY | 8             | 8       | 0          | 0
       | 0

                | Total Count
----------------+-------------
 Masters        | 3
 Tablet Servers | 3
 Tables         | 1
 Tablets        | 8
 Replicas       | 24
OK
```

If the cluster is unhealthy, for instance if a tablet server process has stopped, ksck will report the issue(s) and return a non-zero exit status, as shown in the abbreviated snippet of ksck output below:

```
Tablet Server Summary
               UUID                |        Address         |   Status
-----------------------------------+------------------------+-------------
 a598f75345834133a39c6e51163245db | tserver-01.example.com | HEALTHY
 e05ca6b6573b4e1f9a518157c0c0c637 | tserver-02.example.com | HEALTHY
 e7e53a91fe704296b3a59ad304e7444a | tserver-03.example.com | UNAVAILABLE
Error from 127.0.0.1:7150: Network error: could not get status from server:
 Client connection negotiation failed: client connection to 127.0.0.1:7150:
connect: Connection refused (error 61) (UNAVAILABLE)

... (full output elided)

------------------
Errors:
------------------
Network error: error fetching info from tablet servers: failed to gather i
nfo for all tablet servers: 1 of 3 had errors
Corruption: table consistency check error: 1 out of 1 table(s) are not hea
lthy

FAILED
Runtime error: ksck discovered errors
```

To verify data integrity, the optional --checksum_scan flag can be set, which will ensure the cluster has consistent data by scanning each tablet replica and comparing results. The --tables or --tablets flags can be used to limit the scope of the checksum scan to specific tables or tablets, respectively.

For example, checking data integrity on the my_table table can be done with the following command:

```
$ sudo -u kudu kudu cluster ksck --checksum_scan --tables my_table master-01
.example.com,master-02.example.com,master-03.example.com
```

By default, ksck will attempt to use a snapshot scan of the table, so the checksum scan can be done while writes continue.

Finally, ksck also supports output in JSON format using the --ksck_format flag. JSON output contains the same information as the plain text output, but in a format that can be used by other tools. See kudu cluster ksck --help for more information.

Report Kudu crashes using breakpad

Kudu uses the Google breakpad library to generate a minidump whenever Kudu experiences a crash. A minidump file contains important debugging information about the process that crashed, including shared libraries loaded and their versions, a list of threads running at the time of the crash, the state of the processor registers and a copy of the stack memory for each thread, and CPU and operating system version information. These minidumps are typically only a few MB in size and are generated even if core dump generation is disabled. Currently, generating minidumps is only possible on Linux deployments.

By default, Kudu stores its minidumps in a subdirectory of the configured glog directory called minidumps. This location can be customized by setting the --minidump_path flag. Kudu will retain only a certain number of minidumps before deleting the older ones, in an effort to avoid filling up the disk with minidump files. The maximum number of minidumps that will be retained can be controlled by setting the --max_minidumps gflag.

Minidumps contain information specific to the binary that created them and are therefore not useful without access to the exact binary that crashed, or a very similar binary.

Kudu developers can access the minidump tools in their development environment because they are installed as part of the Kudu thirdparty build. They can be found in the Kudu development environment under uninstrumented/bin. For example, thirdparty/installed/uninstrumented/bin/minidump-2-core.

If minidumps are enabled, it is possible to force Kudu to create a minidump without killing the process. To do that, send a USR1 signal to the kudu-tserver or kudu-master process. For example:

```
sudo pkill -USR1 kudu-tserver
```

Viewing the minidump stack trace with the GNU debugger

Although a minidump contains no heap information, it does contain thread and stack information. You can convert a minidump to a core file to view it with GDB.

To convert the minidump (.dmp file) to a core file:

```
minidump-2-core -o 02cb4a97-ee37-6454-73a9d9cb-590c7dde.core \
02cb4a97-ee37-6454-73a9d9cb-590c7dde.dmp
```

To view the core file with GDB (on a parcel deployment):

```
gdb /opt/cloudera/parcels/KUDU/lib/kudu/sbin-release/kudu-master \
-s /opt/cloudera/parcels/KUDU/lib/debug/usr/lib/kudu/sbin-release/kudu-ma
ster.debug \
02cb4a97-ee37-6454-73a9d9cb-590c7dde.core
```

Enable core dump for the Kudu service

If Kudu crashes, you can use Cloudera Manager to generate a core dump to get more information about the crash.

Procedure

1. Go to the Kudu service.
2. Click the Configuration tab.
3. Search for core dump.
4. Check the checkbox for the Enable Core Dump property.
5. (Optional) Unless otherwise configured, the dump file is generated in the default core dump directory, /var/log/kudu, for both the Kudu master and the tablet servers.
   - To configure a different dump directory for the Kudu master, modify the value of the Kudu Master Core Dump Directory property.
   - To configure a different dump directory for the Kudu tablet servers, modify the value of the Kudu Tablet Server Core Dump Directory property.
6. Click Save Changes.

Use the Charts Library with the Kudu service

By default, the Status tab for the Kudu service displays a dashboard containing a limited set of charts.

About this task

For details on the terminology used in these charts, and instructions on how to query for time-series data, display chart details, and edit charts, see *Charting Time-Series Data*.

The Kudu service's Charts Library tab also displays a dashboard containing a much larger set of charts, organized by categories such as process charts, host charts, CPU charts, and so on, depending on the entity (service, role, or host) that you are viewing. You can use these charts to keep track of disk space usage, the rate at which data is being inserted/modified in Kudu across all tables, or any critical cluster events. You can also use them to keep track of individual tables. For example, to find out how much space a Kudu table is using on disk:

Procedure

1. Go to the Kudu service and navigate to the Charts Library tab.
2. On the left-hand side menu, click Tables to display the list of tables currently stored in Kudu.
3. Click on a table name to view the default dashboard for that table. The Total Tablet Size On Disk Across Kudu Replicas chart displays the total size of the table on disk using a time-series chart.

   Hovering with your mouse over the line on the chart opens a small pop-up window that displays information about that data point. Click the data stream within the chart to display a larger pop-up window that includes additional information for the table at the point in time where the mouse was clicked.

Related Information

Charting Time-Series Data

Apache Kudu Overview

Date published:
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Kudu introduction 4

Kudu architecture in a Cloudera Public Cloud deployment 4

Kudu network architecture 5

Kudu-Impala integration 7

Example use cases 7

Kudu concepts 8

Apache Kudu usage limitations 10

- Schema design limitations 10
- Partitioning limitations 11
- Scaling recommendations and limitations 11
- Server management limitations 12
- Cluster management limitations 12
- Impala integration limitations 12
- Spark integration limitations 12
- Kudu security limitations 13
- Other known issues 13

More resources for Apache Kudu 13

Kudu introduction

Apache Kudu is a columnar storage manager developed for the Hadoop platform. Kudu shares the common technical properties of Hadoop ecosystem applications: Kudu runs on commodity hardware, is horizontally scalable, and supports highly-available operation.

Apache Kudu is a top-level project in the Apache Software Foundation.

This is a test line to verify if the form factor is functioning properly.

Kudu's benefits include:

- Fast processing of OLAP workloads
- Integration with MapReduce, Spark, Flume, and other Hadoop ecosystem components
- Tight integration with Apache Impala, making it a good, mutable alternative to using HDFS with Apache Parquet
- Strong but flexible consistency model, allowing you to choose consistency requirements on a per-request basis, including the option for strict serialized consistency
- Strong performance for running sequential and random workloads simultaneously
- Easy administration and management through Cloudera Manager
- High availability

  Tablet Servers and Master use the Raft consensus algorithm, which ensures availability as long as more replicas are available than unavailable. Reads can be serviced by read-only follower tablets, even in the event of a leader tablet failure.
- Structured data model

Important: Hive-Kudu integration: Cloudera does not support the integration of HiveServer (HS2) with Kudu tables. You cannot run queries against Kudu tables from HS2.

By combining all of these properties, Kudu targets support applications that are difficult or impossible to implement on currently available Hadoop storage technologies. Applications for which Kudu is a viable solution include:

- Reporting applications where new data must be immediately available for end users
- Time-series applications that must support queries across large amounts of historic data while simultaneously returning granular queries about an individual entity
- Applications that use predictive models to make real-time decisions, with periodic refreshes of the predictive model based on historical data

Kudu architecture in a Cloudera Public Cloud deployment

In a Cloudera Public Cloud deployment, Kudu is available as one of the many Cloudera Runtime services within the Real-time Data Mart template. To use Kudu, you can create a Cloudera Data Hub cluster by selecting Real-time Data Mart template template in the Cloudera Management Console.

Each Cloudera Data Hub cluster that is deployed using the Real-time Data Mart template has an instance of Apache Kudu, Impala, Spark, and Knox. It also contains YARN—which is used to run Spark, Hue—which can be used to issue queries using Impala, and HDFS—because the cluster is managed using Cloudera Manager. These components use the shared resources present within the Data Lake.

Impala is primarily used to run analytical queries against the data stored in Kudu. It also manages hierarchical storage between Kudu and the object storage. You can use SQL statements to move older data in the range partitions from Kudu into the Parquet partitions on object storage, for example, on Amazon S3 as cold storage. You can then use an Impala UNION ALL or VIEW statement to query the Kudu and Parquet data together.

Besides ingesting data, Spark is used for backup and disaster recovery (BDR) operations. The Kudu BDR feature is built as a Spark application, and it can take either full or incremental table backups. It then restores a stack of backups into a table.

The following diagram shows Apache Kudu and its dependencies as a part of the Cloudera Data Hub cluster, and the shared resources they use which are present in the Data Lake:

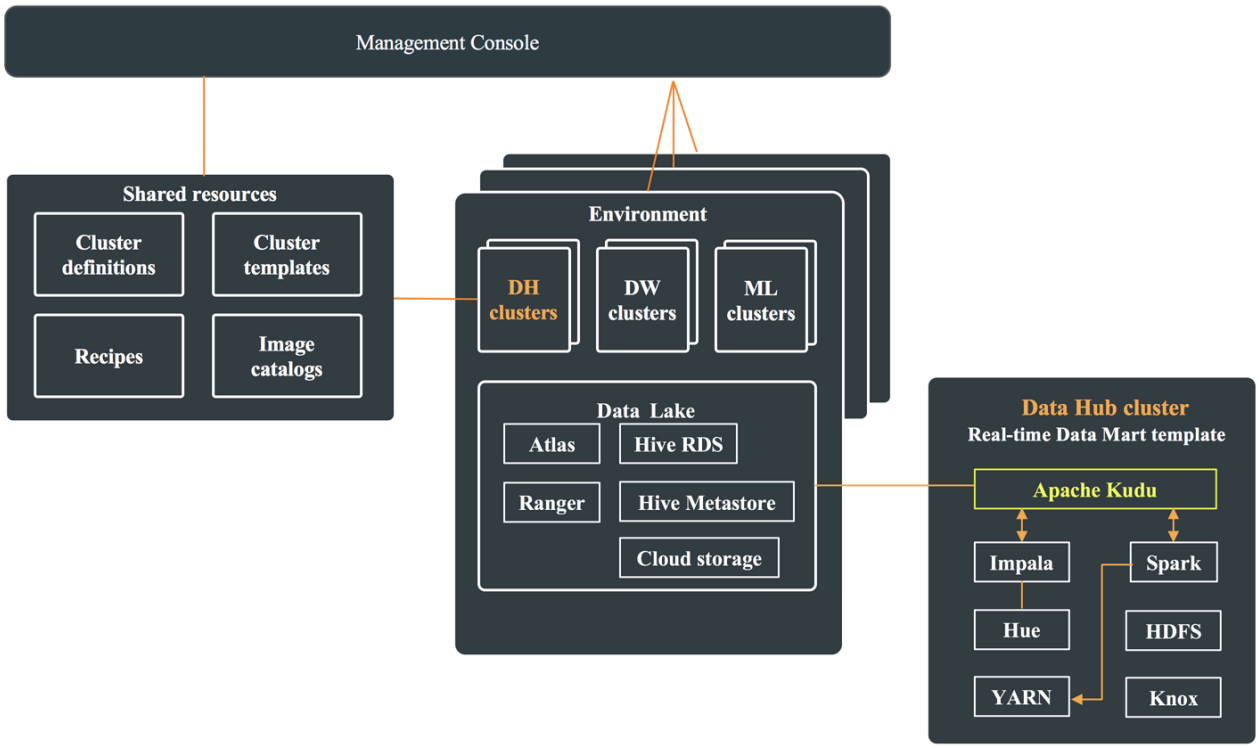

Kudu network architecture

The highly scalable feature of Kudu is a result of Kudu's distributed nature. This topic describes how the logical components within a Kudu cluster interact with each other to achieve high availability and fault tolerance.

Kudu runs on a cluster of machines. These machines collectively store and manage Kudu's data. The cluster architecture is designed for fault tolerance and high availability. You can add more nodes to the Kudu cluster as your storage and throughput needs increase.

Kudu's architecture is based on tablets. A tablet contains a subset of the data in a table. Instead of storing the data in a table in a single location, Kudu distributes the table's tablets to multiple machines in the cluster. Those machines store the tablet's data on their local disks and respond to client requests to read and write table data. A Kudu cluster consists of one or more master nodes and one or more tablet servers. Tablet servers do most of the work in the cluster and typically run on machines known as worker nodes. Tablet servers are responsible for storing and processing data.

Kudu masters don't serve or store table data. They are responsible for the cluster's metadata, such as the schema for every table and on which tablet servers each table's data is stored.

System administrators have the option of setting up a Kudu cluster to be highly available. Therefore, if a single node becomes unavailable, it doesn't cause the entire cluster to be disabled and the applications will always have access to the data.

A highly-available cluster runs in a multi-master mode. A multi-master Kudu cluster has multiple masters, one of which is elected the leader by consensus of all the master nodes. To make sure leader election doesn't result in ties, clusters must have an odd number of masters. Leader masters are responsible for responding to client requests for data access. Masters that are not leaders are called followers. Followers keep their copy of the cluster metadata in sync

with the leader, so that if a leader fails at any time, any of the followers can be elected by the remaining masters as the new leader.

High-availability is an optional feature that is useful when continuous access to the data is critical.

The following diagram shows a Kudu cluster with three masters and multiple tablet servers, each serving multiple tablets. It illustrates how Raft consensus is used to allow for both leaders and followers for both the masters and tablet servers. In addition, a tablet server can be a leader for some tablets and a follower for others. Leaders are shown in gold, while followers are shown in gray.

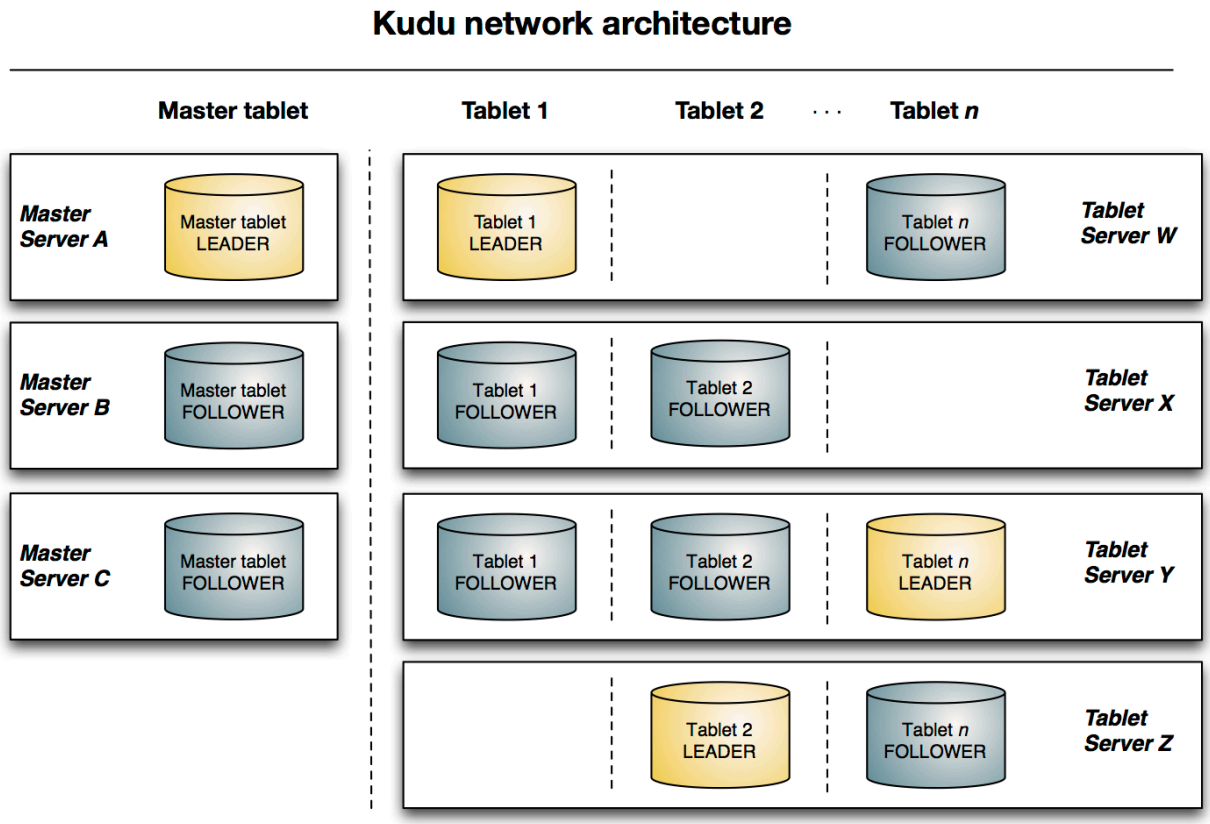

If the master fails when multi-master mode is disabled, tablet servers can continue to serve data as long as the required metadata from the master is cached. However, any requests to access data in tables for which the tablet servers don't have cached metadata will fail until a master is available again. Kudu masters are lightweight processes and can restart quickly.

A key feature to support fault tolerance is tablet replication, which ensures that data in a Kudu table will never be lost. By default, each tablet is replicated on three different servers. Each table can have its own setting for how many times its tablets should be replicated. Note that there must be enough tablet servers in the cluster to allow each replica to be stored on a separate server.

Among all of a tablet's replicas, one is elected leader by the tablet servers, the same way the cluster's Kudu masters elect a leader master. All other replicas are considered followers. Both followers and leaders can service reads, but only leaders can service a write operation.

The leader also propagates these operations to each of the follower replicas, who then perform the same operation with their own data, so that the follower and leader replicas are always in sync. In case a leader replica becomes unavailable, then the followers elect a new leader.

The default replication factor of three allows for continued operation if just one replica is unavailable. A replication factor of five can tolerate up to two unavailable replicas while still guaranteeing data availability.

Kudu-Impala integration

Apache Kudu has tight integration with Apache Impala, allowing you to use Impala to insert, query, update, and delete data from Kudu tablets using Impala's SQL syntax, as an alternative to using the Kudu APIs to build a custom Kudu application. In addition, you can use JDBC or ODBC to connect existing or new applications written in any language, framework, or business intelligence tool to your Kudu data, using Impala as the broker.

- CREATE/ALTER/DROP TABLE - Impala supports creating, altering, and dropping tables using Kudu as the persistence layer. The tables follow the same internal/external approach as other tables in Impala, allowing for flexible data ingestion and querying.
- INSERT - Data can be inserted into Kudu tables from Impala using the same mechanisms as any other table with HDFS or HBase persistence.
- UPDATE/DELETE - Impala supports the UPDATE and DELETE SQL commands to modify existing data in a Kudu table row-by-row or as a batch. The syntax of the SQL commands is designed to be as compatible as possible with existing solutions. In addition to simple DELETE or UPDATE commands, you can specify complex joins in the FROM clause of the query, using the same syntax as a regular SELECT statement.
- Flexible Partitioning - Similar to partitioning of tables in Hive, Kudu allows you to dynamically pre-split tables by hash or range into a predefined number of tablets, in order to distribute writes and queries evenly across your cluster. You can partition by any number of primary key columns, with any number of hashes, a list of split rows, or a combination of these. A partition scheme is required.
- Parallel Scan - To achieve the highest possible performance on modern hardware, the Kudu client used by Impala parallelizes scans across multiple tablets.
- High-efficiency queries - Where possible, Impala pushes down predicate evaluation to Kudu, so that predicates are evaluated as close as possible to the data. Query performance is comparable to Parquet in many workloads.

Example use cases

Kudu gives you capability to stream inputs with near real-time availability. You can run time-series applications with varying access patterns, and develop predective learning models. Kudu gives you freedom to access and query data from any legacy sources or formats using Impala.

Streaming Input with Near Real Time Availability

A common business challenge is one where new data arrives rapidly and constantly, and the same data needs to be available in near real time for reads, scans, and updates. Kudu offers the powerful combination of fast inserts and updates with efficient columnar scans to enable real-time analytics use cases on a single storage layer.

Time-Series Application with Widely Varying Access Patterns

A time-series schema is one in which data points are organized and keyed according to the time at which they occurred. This can be useful for investigating the performance of metrics over time or attempting to predict future behavior based on past data. For instance, time-series customer data might be used both to store purchase click-stream history and to predict future purchases, or for use by a customer support representative. While these different types of analysis are occurring, inserts and mutations might also be occurring individually and in bulk, and become available immediately to read workloads. Kudu can handle all of these access patterns simultaneously in a scalable and efficient manner.

Kudu is a good fit for time-series workloads for several reasons. With Kudu's support for hash-based partitioning, combined with its native support for compound row keys, it is simple to set up a table spread across many servers without the risk of "hotspotting" that is commonly observed when range partitioning is used. Kudu's columnar storage engine is also beneficial in this context, because many time-series workloads read only a few columns, as opposed to the whole row.

In the past, you might have needed to use multiple datastores to handle different data access patterns. This practice adds complexity to your application and operations, and duplicates your data, doubling (or worse) the amount of storage required. Kudu can handle all of these access patterns natively and efficiently, without the need to off-load work to other datastores.

Predictive Modeling

Data scientists often develop predictive learning models from large sets of data. The model and the data might need to be updated or modified often as the learning takes place or as the situation being modeled changes. In addition, the scientist might want to change one or more factors in the model to see what happens over time. Updating a large set of data stored in files in HDFS is resource-intensive, as each file needs to be completely rewritten. In Kudu, updates happen in near real time. The scientist can tweak the value, re-run the query, and refresh the graph in seconds or minutes, rather than hours or days. In addition, batch or incremental algorithms can be run across the data at any time, with near-real-time results.

Combining Data In Kudu With Legacy Systems

Companies generate data from multiple sources and store it in a variety of systems and formats. For instance, some of your data might be stored in Kudu, some in a traditional RDBMS, and some in files in HDFS. You can access and query all of these sources and formats using Impala, without the need to change your legacy systems.

Kudu concepts

In order to use Kudu efficiently, you have to familiarize yourself with some basic concepts related to Kudu.

Columnar datastore

Kudu is a columnar datastore. A columnar datastore stores data in strongly-typed columns. With a proper design, a columnar store can be superior for analytical or data warehousing workloads for the following reasons:

Read efficiency

For analytical queries, you can read a single column, or a portion of that column, while ignoring other columns. This means you can fulfill your request while reading a minimal number of blocks on disk. With a row-based store, you need to read the entire row, even if you only return values from a few columns.

Data compression

Because a given column contains only one type of data, pattern-based compression can be orders of magnitude more efficient than compressing mixed data types, which are used in row-based solutions. Combined with the efficiencies of reading data from columns, compression allows you to fulfill your query while reading even fewer blocks from disk.

Raft consensus algorithm

The Raft consensus algorithm provides a way to elect a leader for a distributed cluster from a pool of potential leaders. If a follower cannot reach the current leader, it transitions itself to become a candidate. Given a quorum of voters, one candidate is elected to be the new leader, and the others transition back to being followers.

A full discussion of Raft is out of scope for this documentation, but it is a robust algorithm.

Kudu uses the Raft consensus algorithm for the election of masters and leader tablets, as well as determining the success or failure of a given write operation.

Table

A tablet is where your data is stored in Kudu. A table has a schema and a totally ordered primary key. A table is split into segments called tablets, by primary key.

Tablet

A tablet is a contiguous segment of a table, similar to a partition in other data storage engines or relational databases. A given tablet is replicated on multiple tablet servers, and at any given point in time, one of these replicas is considered the leader tablet. Any replica can service reads. Writes require consensus among the set of tablet servers serving the tablet.

Tablet server

A tablet server stores and serves tablets to clients. For a given tablet, one tablet server acts as a leader and the others serve follower replicas of that tablet. Only leaders service write requests, while leaders or followers each service read requests. Leaders are elected using Raft consensus. One tablet server can serve multiple tablets, and one tablet can be served by multiple tablet servers.

Master

The master keeps track of all the tablets, tablet servers, the catalog table, and other metadata related to the cluster. At a given point in time, there can only be one acting master (the leader). If the current leader disappears, a new master is elected using Raft consensus.

The master also coordinates metadata operations for clients. For example, when creating a new table, the client internally sends the request to the master. The master writes the metadata for the new table into the catalog table, and coordinates the process of creating tablets on the tablet servers.

All the master's data is stored in a tablet, which can be replicated to all the other candidate masters.

Tablet servers heartbeat to the master at a set interval (the default is once per second).

Catalog table

The catalog table is the central location for metadata of Kudu. It stores information about tables and tablets. The catalog table is accessible to clients through the master, using the client API. The catalog table cannot be read or written directly. Instead, it is accessible only through metadata operations exposed in the client API.

The catalog table stores two categories of metadata:

Table 1: Contents of the catalog table

Content	Description
Tables	Table schemas, locations, and states
Tablets	The list of existing tablets, which tablet servers have replicas of each tablet, the tablet's current state, and start and end keys.

Logical replications

Kudu replicates operations, not on-disk data. This is referred to as logical replication, as opposed to physical replication.

Logical replication has several advantages:

- Although inserts and updates transmit data over the network, deletes do not need to move any data. The delete operation is sent to each tablet server, which performs the delete locally.
- Physical operations, such as compaction, do not need to transmit the data over the network in Kudu. This is different from storage systems that use HDFS, where the blocks need to be transmitted over the network to fulfill the required number of replicas.
- Tablets do not need to perform compactions at the same time or on the same schedule. They do not even need to remain in sync on the physical storage layer. This decreases the chances of all tablet servers experiencing high latency at the same time, due to compactions or heavy write loads.

Related Information

Raft consensus algorithm

Apache Kudu usage limitations

We recommend that you review the usage limitations of Kudu with respect to schema design, scaling, server management, cluster management, replication and backup, security, and integration with Spark and Impala.

Schema design limitations

Kudu currently has some known limitations that may factor into schema design.

Primary key

- The primary key cannot be changed after the table is created. You must drop and recreate a table to select a new primary key.
- The columns which make up the primary key must be listed first in the schema.
- The primary key of a row cannot be modified using the UPDATE functionality. To modify a row's primary key, the row must be deleted and re-inserted with the modified key. Such a modification is non-atomic.
- Columns with DOUBLE, FLOAT, or BOOL types are not allowed as part of a primary key definition. Additionally, all columns that are part of a primary key definition must be NOT NULL.
- Auto-generated primary keys are not supported.
- Cells making up a composite primary key are limited to a total of 16KB after internal composite-key encoding is done by Kudu.

Cells

No individual cell may be larger than 64KB before encoding or compression. The cells making up a composite key are limited to a total of 16KB after the internal composite-key encoding done by Kudu. Inserting rows not conforming to these limitations will result in errors being returned to the client.

Columns

- By default, Kudu tables can have a maximum of 300 columns. We recommend schema designs that use fewer columns for best performance.
- CHAR, and complex types such as ARRAY, MAP, and STRUCT are not supported.
- Type, nullability, and type attributes (i.e. precision and scale of DECIMAL, length of VARCHAR) of the existing columns cannot be changed by altering the table.
- Dropping a column does not immediately reclaim space. Compaction must run first.
- Kudu does not allow the type of a column to be altered after the table is created.

Tables

- Tables must have an odd number of replicas, with a maximum of 7.
- There is no way to run compaction manually, but dropping a table will reclaim the space immediately.
- Kudu does not allow you to change how a table is partitioned after creation, with the exception of adding or dropping range partitions.
- Partitions cannot be split or merged after table creation.

Other usage limitations

- Secondary indexes are not supported.
- Multi-row transactions are not supported.

- Relational features, such as foreign keys, are not supported.
- Identifiers such as column and table names are restricted to be valid UTF-8 strings. Additionally, a maximum length of 256 characters is enforced.

  If you are using Apache Impala to query Kudu tables, refer to the section on *Impala integration limitations* as well.
- Deleted row disk space cannot be reclaimed. The disk space occupied by a deleted row is only reclaimable via compaction, and only when the deletion's age exceeds the "tablet history maximum age" which is controlled by the --tablet_history_max_age_sec flag. Currently, Kudu only schedules compactions in order to improve read/write performance. A tablet will never be compacted purely to reclaim disk space. As such, range partitioning should be used when it is expected that large swaths of rows will be discarded. With range partitioning, individual partitions may be dropped to discard data and reclaim disk space. See KUDU-1625 for more details.

Partitioning limitations

Here are some of the limitations that you must consider before partitionoing tables.

- Tables must be manually pre-split into tablets using simple or compound primary keys. Automatic splitting is not yet possible. Kudu does not allow you to change how a table is partitioned after creation, with the exception of adding or dropping range partitions.
- Data in existing tables cannot currently be automatically repartitioned. As a workaround, create a new table with the new partitioning and insert the contents of the old table.
- Tablets that lose a majority of replicas (such as 1 left out of 3) require manual intervention to be repaired.

Scaling recommendations and limitations

Here are some of the recommended vales that you should consider while deciding the size of the tablet servers, masters, and other storage resources to get optimum performance.

Kudu can seamlessly run across a wide array of environments and workloads with minimal expertise and configuration at the following scale:

- Recommended maximum number of masters is 3.
- Recommended maximum number of tablet servers is 100.
- Recommended maximum amount of stored data, post-replication and post-compression, per tablet server is 8 TiB.
- Recommended number of tablets per tablet server is 1000 (post-replication) with 2000 being the maximum number of tablets allowed per tablet server.
- Maximum number of tablets per table for each tablet server is 60, post-replication (assuming the default replication factor of 3), at table-creation time.
- Recommended maximum amount of data per tablet is 50 GiB. Going beyond this can cause issues such a reduced performance, compaction issues, and slow tablet startup times.

  The recommended target size for tablets is under 10 GiB.

Staying within these limits will provide the most predictable and straightforward Kudu experience. However, experienced users who run on modern hardware, use the latest versions of Kudu, test and tune Kudu for their use case, and work closely with the community, can achieve much higher scales comfortably. Following are some anecdotal values that have been seen in real world production clusters:

- Number of master servers: 3
- More than 300 tablet servers
- 10+ TiB of stored data per tablet server, post-replication and post-compression
- More than 4000 tablets per tablet server, post-replication
- 50 GiB of stored data per tablet. Going beyond this can cause issues such a reduced performance, compaction issues, and slower tablet startup time.

Server management limitations

Here are some of the server management guidelines that you should consider before implementing Kudu.

- Production deployments should configure a least 4 GiB of memory for tablet servers, and ideally more than 16 GiB when approaching the data and tablet scale limits.
- Write ahead logs (WALs) can only be stored on one disk.
- Data directories cannot be removed. You must reformat the data directories to remove them.
- Tablet servers cannot change their address or port.
- Kudu has a hard requirement on having an up-to-date NTP. Kudu masters and tablet servers will crash when out of sync.

Cluster management limitations

When managing Kudu clusters, review the following limitations and recommended maximum point-to-point latency and bandwidth values.

- Recommended maximum point-to-point latency within a Kudu cluster is 20 milliseconds.
- Recommended minimum point-to-point bandwidth within a Kudu cluster is 10 Gbps.
- If you intend to use the location awareness feature to place tablet servers in different locations, it is recommended that you measure the bandwidth and latency between servers to ensure they fit within the above guidelines.
- All masters must be started at the same time when the cluster is started for the very first time.

Impala integration limitations

Here are the limitations when integrating Kudu with Impala, and list of Impala keywords that are not supported for creating Kudu tables.

- When creating a Kudu table, the CREATE TABLE statement must include the primary key columns before other columns, in primary key order.
- Impala cannot update values in primary key columns.
- Kudu tables with a name containing upper case or non-ASCII characters must be assigned an alternate name when used as an external table in Impala.
- Kudu tables with a column name containing upper case or non-ASCII characters cannot be used as an external table in Impala. Columns can be renamed in Kudu to work around this issue.
- != and LIKE predicates are not pushed to Kudu, and instead will be evaluated by the Impala scan node. This may decrease performance relative to other types of predicates.
- Updates, inserts, and deletes using Impala are non-transactional. If a query fails part of the way through, its partial effects will not be rolled back.
- The maximum parallelism of a single query is limited to the number of tablets in a table. For good analytic performance, aim for 10 or more tablets per host or use large tables.

Impala keywords not supported for creating Kudu tables

- PARTITIONED
- LOCATION
- ROWFORMAT

Spark integration limitations

Here are the limitations that you should consider while integrating Kudu and Spark.

- Spark 2.2 (and higher) requires Java 8 at runtime even though Kudu Spark 2.x integration is Java 7 compatible. Spark 2.2 is the default dependency version as of Kudu 1.5.0.
- Kudu tables with a name containing upper case or non-ASCII characters must be assigned an alternate name when registered as a temporary table.
- Kudu tables with a column name containing upper case or non-ASCII characters must not be used with SparkSQL. Columns can be renamed in Kudu to work around this issue.
- <> and ORpredicates are not pushed to Kudu, and instead will be evaluated by the Spark task. Only LIKE predicates with a suffix wildcard are pushed to Kudu. This means LIKE "FOO%" will be pushed, but LIKE "FOO%BAR" won't.
- Kudu does not support all the types supported by Spark SQL. For example, Date and complex types are not supported.
- Kudu tables can only be registered as temporary tables in SparkSQL.
- Kudu tables cannot be queried using HiveContext.

Kudu security limitations

Here are some limitations related to data encryption and authorization in Kudu.

- Kudu uses an internal PKI system to issue X.509 certificates to servers in the cluster. As a result, you cannot run Kudu with public IPs.
- Server certificates generated by Kudu IPKI are incompatible with bouncycastle version 1.52 and earlier.
- When you are creating a new Kudu service using the Ranger web UI, the Test Connection button is displayed. However, the TestConnection tab is not implemented in the Kudu Ranger plugin. As a result, if you try to use it with Kudu, it fails. But that does not mean that the service is not working.

Other known issues

Some of the known bugs and issues with the current release of Kudu are listedmin this topic. These will be addressed in later releases. Note that this list is not exhaustive, and is meant to communicate only the most important known issues.

- If the Kudu master is configured with the -log_force_fsync_all option, the tablet servers and the clients will experience frequent timeouts, and the cluster may become unusable.
- If a tablet server has a very large number of tablets, it may take several minutes to start up. It is recommended to limit the number of tablets per server to 1000 or fewer. Consider this limitation when pre-splitting your tables. If you notice slow start-up times, you can monitor the number of tablets per server in the web UI.

More resources for Apache Kudu

The following is a list of resources that may help you to understand some of the architectural features of Apache Kudu and columnar data storage. The links further down tend toward the academic and are not required reading in order to understand how to install, use, and administer Kudu.

Resources

Kudu Project

Read the official Kudu documentation and learn how you can get involved.

Kudu Documentation

Read the official Kudu documentation, which includes more in-depth information about installation and configuration choices.

Kudu Github Repository

Examine the Kudu source code and contribute to the project.

Kudu-Examples Github Repository

View and run several Kudu code examples, as well as the Kudu Quickstart VM.

Kudu White Paper

Read draft of the white paper discussing Kudu's architecture, written by the Kudu development team.

In Search Of An Understandable Consensus Algorithm*, Diego Ongaro and John Ousterhout, Stanford University. 2014.*

The original whitepaper describing the Raft consensus algorithm.

Support

Bug reports and feedback can be submitted through the public JIRA, our Cloudera Community Kudu forum, and a public mailing list monitored by the Kudu development team and community members. In addition, a public Slack instance is available to communicate with the team.

Planning for Apache Kudu

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Kudu schema design....4
The perfect schema....4
Column design....4
Decimal type....5
Varchar type....5
Column encoding....5
Column compression....6
Primary key design....6
Primary key index....7
Non-unique primary key index....7
Considerations for backfill inserts....7
Partitioning....8
Range partitioning....8
Hash partitioning....9
Multilevel partitioning....9
Partition pruning....9
Partitioning examples....9
Schema alterations....12
Schema design limitations....13
Partitioning limitations....14

Kudu transaction semantics....14
Single tablet write operations....15
Writing to multiple tablets....15
Read operations (scans)....16
Known issues and limitations....17
Writes....17
Reads (scans)....18

Scaling Kudu....18
Terms....18
Example workload....18
Memory....19
Verifying if a memory limit is sufficient....19
File descriptors....20
Threads....20
Scaling recommendations and limitations....21

Kudu schema design

Kudu tables have a structured data model similar to tables in a traditional relational database. With Kudu, schema design is critical for achieving the best performance and operational stability. Every workload is unique, and there is no single schema design that is best for every table. This topic outlines effective schema design philosophies for Kudu, and how they differ from approaches used for traditional relational database schemas.

There are three main concerns when creating Kudu tables: column design, primary key design, and partitioning.

The perfect schema

The perfect schema would accomplish the following:

- Data would be distributed such that reads and writes are spread evenly across tablet servers. This can be achieved by effective partitioning.
- Tablets would grow at an even, predictable rate, and load across tablets would remain steady over time. This can be achieved by effective partitioning.
- Scans would read the minimum amount of data necessary to fulfill a query. This is impacted mostly by primary key design, but partitioning also plays a role via partition pruning.

The perfect schema depends on the characteristics of your data, what you need to do with it, and the topology of your cluster. Schema design is the single most important thing within your control to maximize the performance of your Kudu cluster.

Column design

A Kudu table consists of one or more columns, each with a defined type. Columns that are not part of the primary key may be nullable.

Supported column types include:

- boolean
- 8-bit signed integer
- 16-bit signed integer
- 32-bit signed integer
- 64-bit signed integer
- date (32-bit days since the Unix epoch)
- unixtime_micros (64-bit microseconds since the Unix epoch)
- single-precision (32-bit) IEEE-754 floating-point number
- double-precision (64-bit) IEEE-754 floating-point number
- decimal
- varchar
- UTF-8 encoded string (up to 64KB uncompressed)
- binary (up to 64KB uncompressed)
- VARCHAR type with configurable maximum length (up to 64KB uncompressed)

Kudu takes advantage of strongly-typed columns and a columnar on-disk storage format to provide efficient encoding and serialization. To make the most of these features, columns should be specified as the appropriate type, rather than simulating a 'schemaless' table using string or binary columns for data which could otherwise be structured. In addition to encoding, Kudu allows compression to be specified on a per-column basis.

Attention: Kudu does not provide a version or timestamp column to track changes to a row. If version or timestamp information is needed, the schema should include an explicit version or timestamp column.

Decimal type

The decimal type is a numeric data type with fixed scale and precision suitable for financial and other arithmetic calculations where the imprecise representation and rounding behavior of float and double make those types impractical. The decimal type is also useful for integers larger than int64 and cases with fractional values in a primary key.

The decimal type is a parameterized type that takes precision and scale type attributes.

Precision represents the total number of digits that can be represented by the column, regardless of the location of the decimal point. This value must be between 1 and 38 and has no default. For example, a precision of 4 is required to represent integer values up to 9999, or to represent values up to 99.99 with two fractional digits. You can also represent corresponding negative values, without any change in the precision. For example, the range -9999 to 9999 still only requires a precision of 4.

Scale represents the number of fractional digits. This value must be between 0 and the precision. A scale of 0 produces integral values, with no fractional part. If precision and scale are equal, all of the digits come after the decimal point. For example, a decimal with precision and scale equal to 3 can represent values between -0.999 and 0.999.

Performance considerations:

Kudu stores each value in as few bytes as possible depending on the precision specified for the decimal column. For that reason it is not advised to just use the highest precision possible for convenience. Doing so could negatively impact performance, memory and storage.

Before encoding and compression:

- Decimal values with precision of 9 or less are stored in 4 bytes.
- Decimal values with precision of 10 through 18 are stored in 8 bytes.
- Decimal values with precision greater than 18 are stored in 16 bytes.

Note: The precision and scale of decimal columns cannot be changed by altering the table.

Varchar type

The varchar type is a UTF-8 encoded string (up to 64KB uncompressed) with a fixed maximum character length. This type is especially useful when migrating from or integrating with legacy systems that support the varchar type. If a maximum character length is not required the string type should be used instead.

The varchar type is a parameterized type that takes a length attribute.

Length represents the maximum number of UTF-8 characters allowed. Values with characters greater than the limit will be truncated. This value must be between 1 and 65535 and has no default. Note that some other systems may represent the length limit in bytes instead of characters. That means that Kudu may be able to represent longer values in the case of multi-byte UTF-8 characters.

Column encoding

Depending on the type of the column, Kudu columns can be created with the following encoding types.

Plain Encoding

Data is stored in its natural format. For example, int32 values are stored as fixed-size 32-bit little-endian integers.

Bitshuffle Encoding

A block of values is rearranged to store the most significant bit of every value, followed by the second most significant bit of every value, and so on. Finally, the result is LZ4 compressed. Bitshuffle encoding is a good choice for columns that have many repeated values, or values that change by small amounts when sorted by primary key. The bitshuffle project has a good overview of performance and use cases.

Run Length Encoding

Runs (consecutive repeated values) are compressed in a column by storing only the value and the count. Run length encoding is effective for columns with many consecutive repeated values when sorted by primary key.

Dictionary Encoding

Dictionary encoding can be used for BINARY or STRING columns. A dictionary of unique values is built, and each column value is encoded as its corresponding index in the dictionary. Dictionary encoding is effective for columns with low cardinality. If the column values of a given row set are unable to be compressed because the number of unique values is too high, Kudu will transparently fall back to plain encoding for that row set. This is evaluated during flush.

Prefix Encoding

Common prefixes are compressed in consecutive column values. Prefix encoding can be effective for values that share common prefixes, or the first column of the primary key, since rows are sorted by primary key within tablets.

Each column in a Kudu table can be created with an encoding, based on the type of the column. Starting with Kudu 1.3, default encodings are specific to each column type.

Column Type	Encoding	Default
int8, int16, int32, int64	plain, bitshuffle, run length	bitshuffle
date, unixtime_micros	plain, bitshuffle, run length	bitshuffle
float, double	plain, bitshuffle	bitshuffle
bool	plain, run length	run length
string, varchar, binary	plain, prefix, dictionary	dictionary

Column compression

Kudu allows per-column compression using the LZ4, Snappy, or zlib compression codecs.

By default, columns that are Bitshuffle-encoded are inherently compressed with the LZ4 compression. Otherwise, columns are stored uncompressed. Consider using compression if reducing storage space is more important than raw scan performance.

Every data set will compress differently, but in general LZ4 is the most efficient codec, while zlib will compress to the smallest data sizes. Bitshuffle-encoded columns are automatically compressed using LZ4, so it is not recommended to apply additional compression on top of this encoding.

Primary key design

Every Kudu table must declare a primary key comprised of one or more columns. Like an RDBMS primary key, the Kudu primary key enforces a uniqueness constraint. Attempting to insert a row with the same primary key values as an existing row results in a duplicate key error.

Primary key columns must be non-nullable, and may not be a boolean or floating-point type.

Once set during table creation, the set of columns in the primary key may not be altered.

Unlike an RDBMS, Kudu does not provide an explicit auto-incrementing column feature, so the application must always provide the full primary key during insert.

Columns which do not satisfy the uniqueness constraint can still be used as primary keys, by specifying them as non-unique primary keys.

Row delete and update operations must also specify the full primary key of the row to be changed. Kudu does not natively support range deletes or updates.

The primary key values of a column may not be updated after the row is inserted. However, the row may be deleted and re-inserted with the updated value.

Primary key index

As with many traditional relational databases, Kudu's primary key is in a clustered index. All rows within a tablet are sorted by its primary key.

When scanning Kudu rows, use equality or range predicates on primary key columns to efficiently find the rows.

Attention: Primary key indexing optimizations apply to scans on individual tablets. See the *Partitioning pruning* topic for details on how scans can use predicates to skip entire tablets.

Related Information

Partition pruning

Non-unique primary key index

Learn how Kudu internally creates an auto-incrementing column when you specify columns as non-unique primary key. The specified columns and the auto-incrementing column form the effective primary key. Also learn about the supported and unsupported operations on such tables.

Note: The auto-incrementing counter which is used to assign value for the auto-incrementing column is managed by Kudu. The counter values are monotonically increasing for every tablet.

Non-unique primary key columns must be non-nullable, and need not be a boolean, float, or double type.

Once configured during table creation, the set of columns in the non-unique primary key and the auto-incrementing column can not be altered.

Considerations related to operations supported on tables with non-unique primary key are as follows:

- For insert operations, provide values for the non-unique primary key columns without specifying the values for auto-incrementing column. The auto-incrementing column is populated on the server side automatically.
- For update or delete operations, provide the complete set of key columns. You need to perform a scan before an update or a delete operation to get the auto-incrementing value.

The non-unique primary key values of a column might not be updated after the row is inserted. However, the row may be deleted and re-inserted with the updated value; moreover, a new auto-incrementing counter value is assigned to the auto-incrementing column during insertion operation.

Operations not supported on tables with non-unique primary key are as follows:

- Restoring tables with non-unique primary keys is not supported currently.
- Upsert operation is not supported on tables with non-unique primary key.

For more details on how to use non-unique primary key, see the examples on https://github.com/apache/kudu/tree/master/examples.

Considerations for backfill inserts

This section discuss a primary key design consideration for timeseries use cases where the primary key is a timestamp, or the first column of the primary key is a timestamp.

Each time a row is inserted into a Kudu table, Kudu looks up the primary key in the primary key index storage to check whether that primary key is already present in the table. If the primary key exists in the table, a "duplicate key" error is returned. In the typical case where data is being inserted at the current time as it arrives from the data source, only a small range of primary keys are "hot". So, each of these "check for presence" operations is very fast. It finds the cached primary key storage in memory and doesn't require going to disk.

In the case when you load historical data, which is called "backfilling", from an offline data source, each row that is inserted is likely to reach a cold area of the primary key index which is not resident in memory and will cause one or more HDD disk seeks. For example, in a normal ingestion case where Kudu sustains a few million inserts per second, the "backfill" use case might sustain only a few thousand inserts per second.

To alleviate the performance issue during backfilling, consider the following options:

- Make the primary keys more compressible.

  For example, with the first column of a primary key being a random ID of 32-bytes, caching one billion primary keys would require at least 32 GB of RAM to stay in cache. If caching backfill primary keys from several days ago, you need to have several times 32 GB of memory. By changing the primary key to be more compressible, you increase the likelihood that the primary keys can fit in cache and thus reducing the amount of random disk I/Os.
- Use SSDs for storage as random seeks are orders of magnitude faster than spinning disks.
- Change the primary key structure such that the backfill writes find a continuous range of primary keys.

Partitioning

In order to provide scalability, Kudu tables are partitioned into units called tablets, and distributed across many tablet servers. A row always belongs to a single tablet. The method of assigning rows to tablets is determined by the partitioning of the table, which is set during table creation.

Choosing a partitioning strategy requires understanding the data model and the expected workload of a table. For write-heavy workloads, it is important to design the partitioning such that writes are spread across tablets in order to avoid overloading a single tablet. For workloads involving many short scans, where the overhead of contacting remote servers dominates, performance can be improved if all of the data for the scan is located on the same tablet. Understanding these fundamental trade-offs is central to designing an effective partition schema.

Important: Kudu does not provide a default partitioning strategy when creating tables. It is recommended that new tables which are expected to have heavy read and write workloads have at least as many tablets as tablet servers.

Kudu provides two types of partitioning: range partitioning and hash partitioning. Tables may also have multilevel partitioning, which combines range and hash partitioning, or multiple instances of hash partitioning.

Range partitioning

Range partitioning distributes rows using a totally-ordered range partition key. Each partition is assigned a contiguous segment of the range partition keyspace. The key must be comprised of a subset of the primary key columns. If the range partition columns match the primary key columns, then the range partition key of a row will equal its primary key. In range partitioned tables without hash partitioning, each range partition will correspond to exactly one tablet.

The initial set of range partitions is specified during table creation as a set of partition bounds and split rows. For each bound, a range partition will be created in the table. Each split will divide a range partition in two. If no partition bounds are specified, then the table will default to a single partition covering the entire key space (unbounded below and above). Range partitions must always be non-overlapping, and split rows must fall within a range partition.

Adding and Removing Range Partitions

Kudu allows range partitions to be dynamically added and removed from a table at runtime, without affecting the availability of other partitions. Removing a partition will delete the tablets belonging to the partition, as well as the data contained in them. Subsequent inserts into the dropped partition will fail. New partitions can be added, but they must not overlap with any existing range partitions. Kudu allows dropping and adding any number of range partitions in a single transactional alter table operation.

Dynamically adding and dropping range partitions is particularly useful for time series use cases. As time goes on, range partitions can be added to cover upcoming time ranges. For example, a table storing an event log could add a month-wide partition just before the start of each month in order to hold the upcoming events. Old range partitions can be dropped in order to efficiently remove historical data, as necessary.

Hash partitioning

Hash partitioning distributes rows by hash value into one of many buckets. In single-level hash partitioned tables, each bucket will correspond to exactly one tablet. The number of buckets is set during table creation. Typically the primary key columns are used as the columns to hash, but as with range partitioning, any subset of the primary key columns can be used.

Hash partitioning is an effective strategy when ordered access to the table is not needed. Hash partitioning is effective for spreading writes randomly among tablets, which helps mitigate hot-spotting and uneven tablet sizes.

Multilevel partitioning

Kudu allows a table to combine multiple levels of partitioning on a single table. Zero or more hash partition levels can be combined with an optional range partition level. The only additional constraint on multilevel partitioning beyond the constraints of the individual partition types, is that multiple levels of hash partitions must not hash the same columns.

When used correctly, multilevel partitioning can retain the benefits of the individual partitioning types, while reducing the downsides of each. The total number of tablets in a multilevel partitioned table is the product of the number of partitions in each level.

Partition pruning

Kudu scans will automatically skip scanning entire partitions when it can be determined that the partition can be entirely filtered by the scan predicates. To prune hash partitions, the scan must include equality predicates on every hashed column. To prune range partitions, the scan must include equality or range predicates on the range partitioned columns. Scans on multilevel partitioned tables can take advantage of partition pruning on any of the levels independently.

Partitioning examples

To illustrate the factors and tradeoffs associated with designing a partitioning strategy for a table, we will walk through some different partitioning scenarios.

Consider the following table schema for storing machine metrics data (using SQL syntax and date-formatted timestamps for clarity):

```
CREATE TABLE metrics (
    host STRING NOT NULL,
    metric STRING NOT NULL,
    time INT64 NOT NULL,
    value DOUBLE NOT NULL,
    PRIMARY KEY (host, metric, time)
);
```

Range partitioning

A natural way to partition the metrics table is to range partition on the time column. Let’s assume that we want to have a partition per year, and the table will hold data for 2014, 2015, and 2016. There are at least two ways that the table could be partitioned: with unbounded range partitions, or with bounded range partitions.

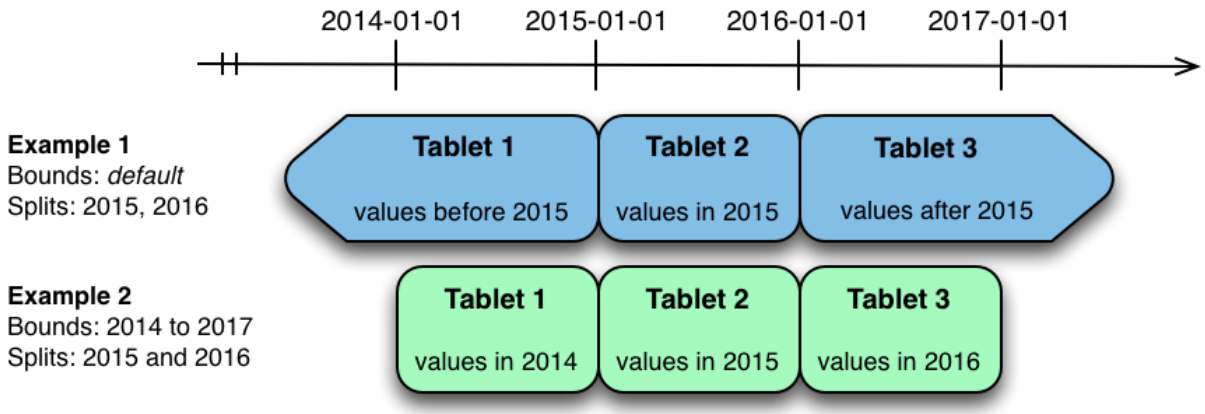

The image above shows the two ways the metrics table can be range partitioned on the time column. In the first example (in blue), the default range partition bounds are used, with splits at 2015-01-01 and 2016-01-01. This results in three tablets: the first containing values before 2015, the second containing values in the year 2015, and the third containing values after 2016. The second example (in green) uses a range partition bound of [(2014-01-01), (2017-01-01)], and splits at 2015-01-01 and 2016-01-01. The second example could have equivalently been expressed through range partition bounds of [(2014-01-01), (2015-01-01)], [(2015-01-01), (2016-01-01)], and [(2016-01-01), (2017-01-01)], with no splits. The first example has unbounded lower and upper range partitions, while the second example includes bounds.

Each of the range partition examples above allows time-bounded scans to prune partitions falling outside of the scan's time bound. This can greatly improve performance when there are many partitions. When writing, both examples suffer from potential hot-spotting issues. Because metrics tend to always be written at the current time, most writes will go into a single range partition.

The second example is more flexible, because it allows range partitions for future years to be added to the table. In the first example, all writes for times after 2016-01-01 will fall into the last partition, so the partition may eventually become too large for a single tablet server to handle.

Hash partitioning

Another way of partitioning the metrics table is to hash partition on the host and metric columns.

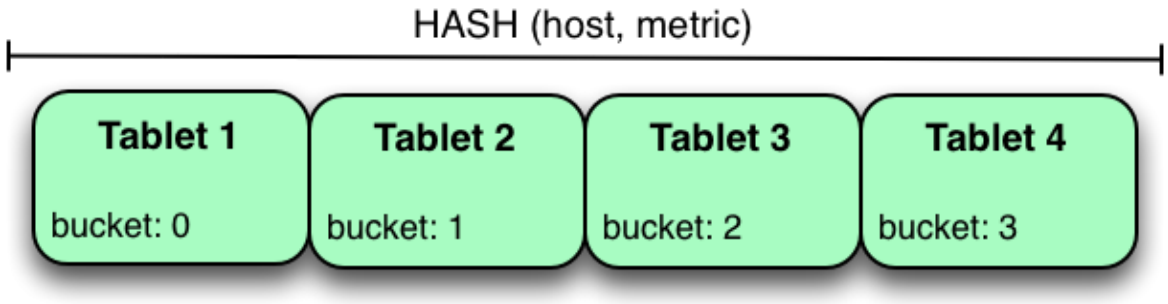

In the example above, the metrics table is hash partitioned on the host and metric columns into four buckets. Unlike the range partitioning example earlier, this partitioning strategy will spread writes over all tablets in the table evenly, which helps overall write throughput. Scans over a specific host and metric can take advantage of partition pruning by specifying equality predicates, reducing the number of scanned tablets to one. One issue to be careful of with a pure hash partitioning strategy, is that tablets could grow indefinitely as more and more data is inserted into the table. Eventually tablets will become too big for an individual tablet server to hold.

Hash and range partitioning

The previous examples showed how the metrics table could be range partitioned on the time column, or hash partitioned on the host and metric columns. These strategies have associated strength and weaknesses:

Table 1: Partitioning strategies

Strategy	Writes	Reads	Tablet Growth
range(time)	# - all writes go to latest partition	# - time-bounded scans can be pruned	# - new tablets can be added for future time periods
hash(host, metric)	# - writes are spread evenly among tablets	# - scans on specific hosts and metrics can be pruned	# - tablets could grow too large

Hash partitioning is good at maximizing write throughput, while range partitioning avoids issues of unbounded tablet growth. Both strategies can take advantage of partition pruning to optimize scans in different scenarios. Using multilevel partitioning, it is possible to combine the two strategies in order to gain the benefits of both, while minimizing the drawbacks of each.

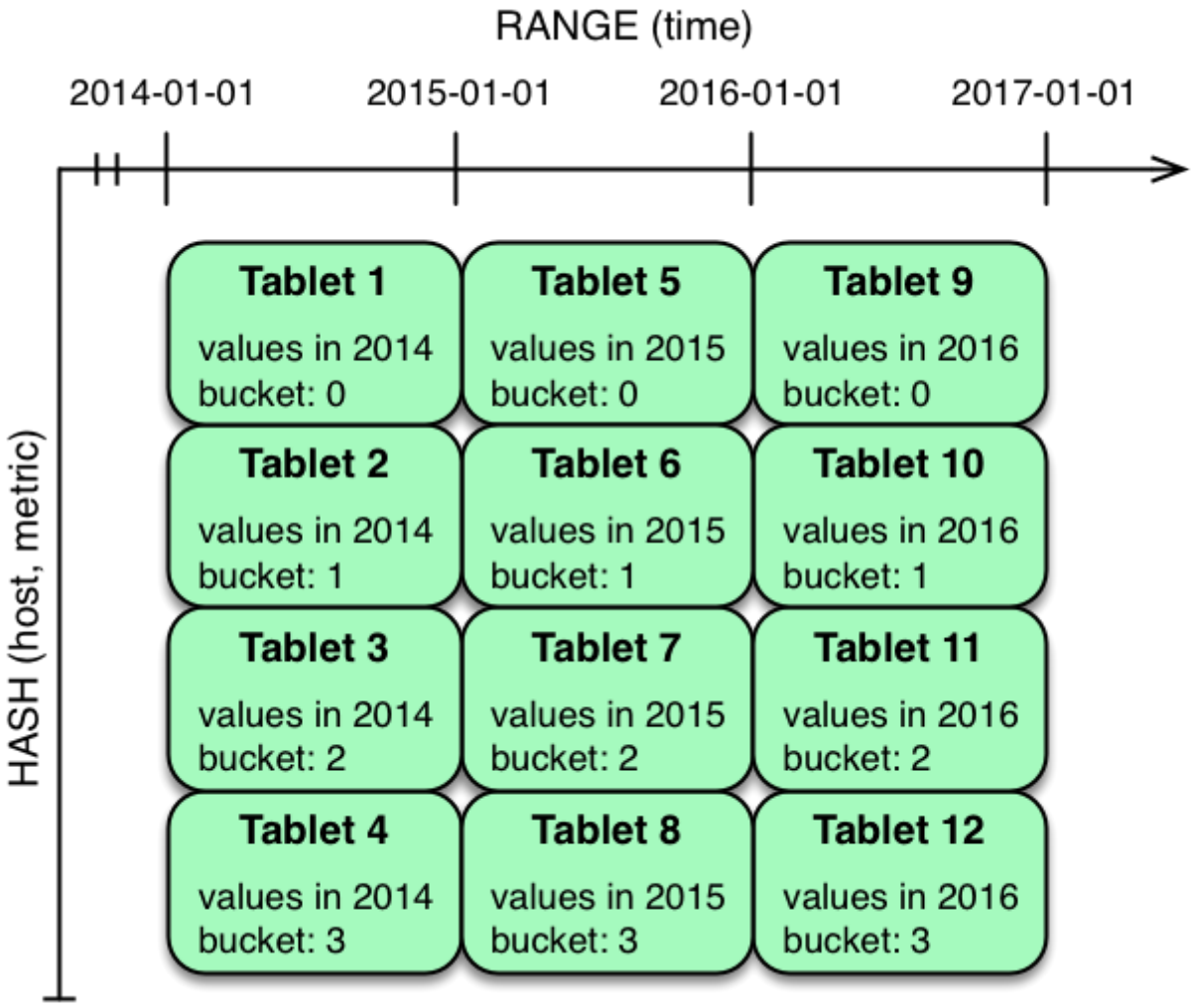

In the example above, range partitioning on the time column is combined with hash partitioning on the host and metr ic columns. This strategy can be thought of as having two dimensions of partitioning: one for the hash level and one for the range level. Writes into this table at the current time will be parallelized up to the number of hash buckets, in this case 4. Reads can take advantage of time bound and specific host and metric predicates to prune partitions. New

range partitions can be added, which results in creating 4 additional tablets (as if a new column were added to the diagram).

Hash and hash partitioning

Kudu can support any number of hash partitioning levels in the same table, as long as the levels have no hashed columns in common.

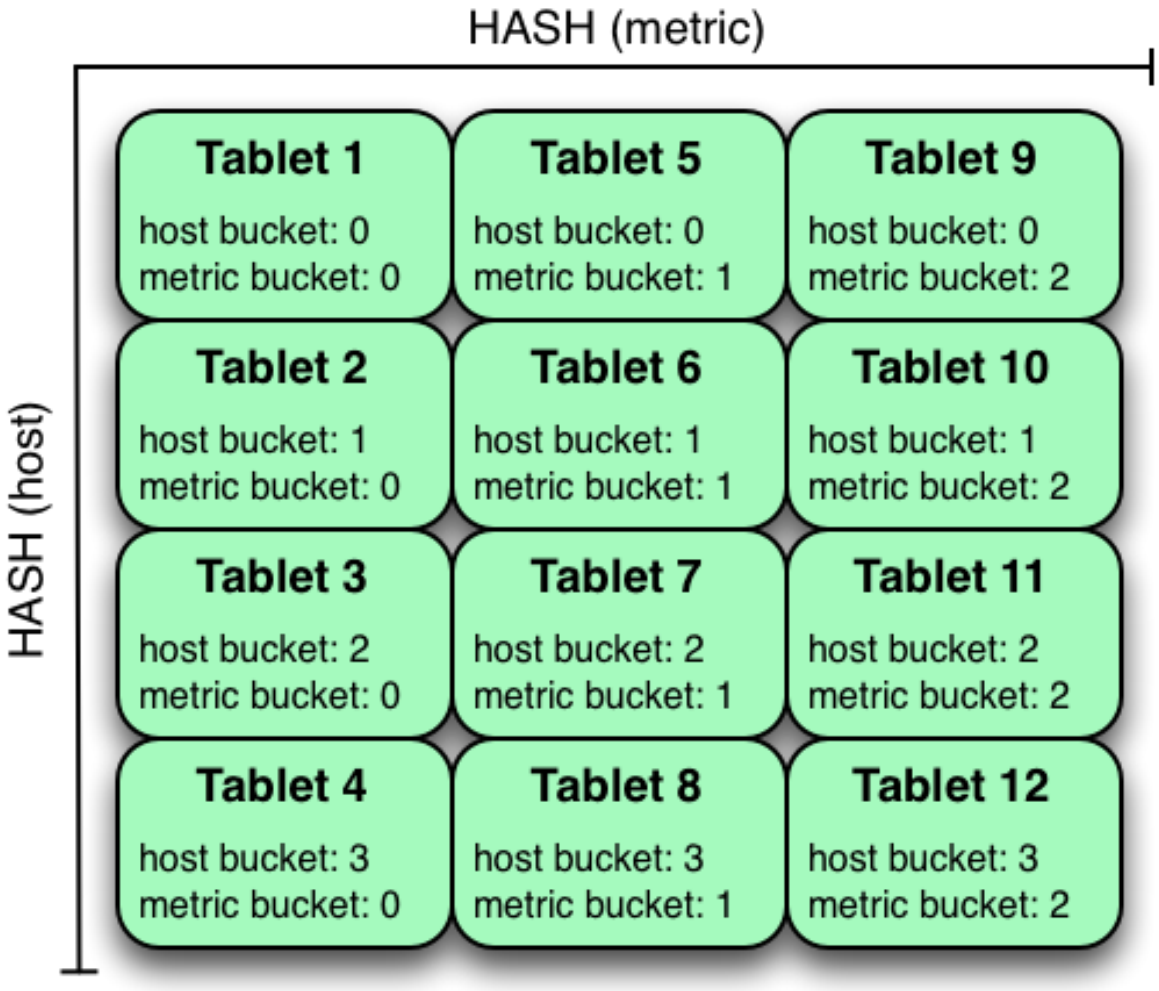

In the example above, the table is hash partitioned on host into 4 buckets, and hash partitioned on metric into 3 buckets, resulting in 12 tablets. Although writes will tend to be spread among all tablets when using this strategy, it is slightly more prone to hot-spotting than when hash partitioning over multiple independent columns, since all values for an individual host or metric will always belong to a single tablet. Scans can take advantage of equality predicates on the host and metric columns separately to prune partitions.

Multiple levels of hash partitioning can also be combined with range partitioning, which logically adds another dimension of partitioning.

Schema alterations

This topic lists the ways in which you can alter a table's schema.

- Rename the table

- Rename primary key columns
- Rename, add, or drop non-primary key columns
- Add and drop range partitions

Multiple alteration steps can be combined in a single transactional operation.

Schema design limitations

Kudu currently has some known limitations that may factor into schema design.

Primary key

- The primary key cannot be changed after the table is created. You must drop and recreate a table to select a new primary key.
- The columns which make up the primary key must be listed first in the schema.
- The primary key of a row cannot be modified using the UPDATE functionality. To modify a row's primary key, the row must be deleted and re-inserted with the modified key. Such a modification is non-atomic.
- Columns with DOUBLE, FLOAT, or BOOL types are not allowed as part of a primary key definition. Additionally, all columns that are part of a primary key definition must be NOT NULL.
- Auto-generated primary keys are not supported.
- Cells making up a composite primary key are limited to a total of 16KB after internal composite-key encoding is done by Kudu.

Cells

No individual cell may be larger than 64KB before encoding or compression. The cells making up a composite key are limited to a total of 16KB after the internal composite-key encoding done by Kudu. Inserting rows not conforming to these limitations will result in errors being returned to the client.

Columns

- By default, Kudu tables can have a maximum of 300 columns. We recommend schema designs that use fewer columns for best performance.
- CHAR, and complex types such as ARRAY, MAP, and STRUCT are not supported.
- Type, nullability, and type attributes (i.e. precision and scale of DECIMAL, length of VARCHAR) of the existing columns cannot be changed by altering the table.
- Dropping a column does not immediately reclaim space. Compaction must run first.
- Kudu does not allow the type of a column to be altered after the table is created.

Tables

- Tables must have an odd number of replicas, with a maximum of 7.
- There is no way to run compaction manually, but dropping a table will reclaim the space immediately.
- Kudu does not allow you to change how a table is partitioned after creation, with the exception of adding or dropping range partitions.
- Partitions cannot be split or merged after table creation.

Other usage limitations

- Secondary indexes are not supported.
- Multi-row transactions are not supported.
- Relational features, such as foreign keys, are not supported.

- Identifiers such as column and table names are restricted to be valid UTF-8 strings. Additionally, a maximum length of 256 characters is enforced.

  If you are using Apache Impala to query Kudu tables, refer to the section on *Impala integration limitations* as well.
- Deleted row disk space cannot be reclaimed. The disk space occupied by a deleted row is only reclaimable via compaction, and only when the deletion's age exceeds the "tablet history maximum age" which is controlled by the --tablet_history_max_age_sec flag. Currently, Kudu only schedules compactions in order to improve read/write performance. A tablet will never be compacted purely to reclaim disk space. As such, range partitioning should be used when it is expected that large swaths of rows will be discarded. With range partitioning, individual partitions may be dropped to discard data and reclaim disk space. See KUDU-1625 for more details.

Related Information

Impala integration limitations

Partitioning limitations

Here are some of the limitations that you must consider before partitionoing tables.

- Tables must be manually pre-split into tablets using simple or compound primary keys. Automatic splitting is not yet possible. Kudu does not allow you to change how a table is partitioned after creation, with the exception of adding or dropping range partitions.
- Data in existing tables cannot currently be automatically repartitioned. As a workaround, create a new table with the new partitioning and insert the contents of the old table.
- Tablets that lose a majority of replicas (such as 1 left out of 3) require manual intervention to be repaired.

Kudu transaction semantics

This is a brief introduction to Kudu’s transaction and consistency semantics. Kudu's core philosophy is to provide transactions with simple, strong semantics, without sacrificing performance or the ability to tune to different requirements. Kudu’s transactional semantics and architecture are inspired by state-of-the-art systems such as Spanner and Calvin. For an in-depth technical exposition of what is mentioned here, see the *Technical Report: HybridTime - Accessible Global Consistency with High Clock Uncertainty*.

Kudu currently allows the following operations:

- Scans are read operations that can traverse multiple tablets and read information with different levels of consistency or correctness guarantees. Scans can also perform time-travel reads. That is, you can set a scan timestamp from the past and get back results that reflect the state of the storage engine at that point in time.
- Write operations are sets of rows to be inserted, updated, or deleted in the storage engine, in a single tablet with multiple replicas. Write operations do not have separate "read sets", that is, they do not scan existing data before performing the write. Each write is only concerned with the previous state of the rows that are about to change. Writes are not "committed" explicitly by the user. Instead, they are committed automatically by the system, after completion.

While Kudu is designed to eventually be fully ACID (Atomic, Consistent, Isolated, Durable), multi-tablet transactions have not yet been implemented. As such, the following discussion focuses on single-tablet write operations, and only briefly touches multi-tablet reads.

Related Information

Spanner

Calvin

Technical Report: HybridTime - Accessible Global Consistency with High Clock Uncertainty

Single tablet write operations

Kudu employs Multiversion Concurrency Control (MVCC) and the Raft consensus algorithm. Each write operation in Kudu must go through the following order of operations:

1. The tablet's leader acquires all locks for the rows that it will change.
2. The leader assigns the write a timestamp before the write is submitted for replication. This timestamp will be the write's *TAG* in MVCC.
3. After a majority of replicas have acknowledged the write, the rows are changed.
4. After the changes are complete, they are made visible to concurrent writes and reads, atomically.

All replicas of a tablet observe the same process. Therefore, if a write operation is assigned timestamp n, and changes row x, a second write operation at timestamp m > n is guaranteed to see the new value of x.

This strict ordering of lock acquisition and timestamp assignment is enforced to be consistent across all replicas of a tablet through consensus. Therefore, write operations are ordered with regard to clock-assigned timestamps, relative to other writes in the same tablet. In other words, writes have strict-serializable semantics.

In case of multi-row write operations, while they are Isolated and Durable in an ACID sense, they are not yet fully Atomic. The failure of a single write in a batch operation will not roll back the entire operation, but produce per-row errors.

Writing to multiple tablets

Kudu does not support transactions that span multiple tablets. However, consistent snapshot reads are possible (with caveats, as explained below). Writes from a Kudu client are optionally buffered in memory until they are flushed and sent to the tablet server. When a client's session is flushed, the rows for each tablet are batched together, and sent to the tablet server which hosts the leader replica of the tablet. Since there are no inter-tablet transactions, each of these batches represents a single, independent write operation with its own timestamp. However, the client API provides the option to impose some constraints on the assigned timestamps and on how writes to different tablets are observed by clients.

Kudu was designed to be externally consistent, that is, preserving consistency when operations span multiple tablets and even multiple data centers. In practice this means that if a write operation changes item x at tablet A, and a following write operation changes item y at tablet B, you might want to enforce that if the change to y is observed, the change to x must also be observed. There are many examples where this can be important. For example, if Kudu is storing clickstreams for further analysis, and two clicks follow each other but are stored in different tablets, subsequent clicks should be assigned subsequent timestamps so that the causal relationship between them is captured.

- CLIENT_PROPAGATED Consistency

  Kudu's default external consistency mode is called CLIENT_PROPAGATED. This mode causes writes from a single client to be automatically externally consistent. In the clickstream scenario above, if the two clicks are submitted by different client instances, the application must manually propagate timestamps from one client to the other for the causal relationship to be captured. Timestamps between clients a and b can be propagated as follows:

  Java Client

  Call AsyncKuduClient#getLastPropagatedTimestamp() on client a, propagate the timestamp to client b, and call AsyncKuduClient#setLastPropagatedTimestamp() on client b.

  C++ Client

  Call KuduClient::GetLatestObservedTimestamp() on client a, propagate the timestamp to client b, and call KuduClient::SetLatestObservedTimestamp() on client b.

- COMMIT_WAIT Consistency

  Kudu also has an experimental implementation of an external consistency model (used in Google's Spanner), called COMMIT_WAIT. COMMIT_WAIT works by tightly synchronizing the clocks on all machines in the cluster. Then, when a write occurs, timestamps are assigned and the results of the write are not made visible

until enough time has passed so that no other machine in the cluster could possibly assign a lower timestamp to a following write.

When using this mode, the latency of writes is tightly tied to the accuracy of clocks on all the cluster hosts, and using this mode with loose clock synchronization causes writes to either take a long time to complete, or even time out.

The COMMIT_WAIT consistency mode may be selected as follows:

Java Client

Call KuduSession#setExternalConsistencyMode(ExternalConsistencyMode.COMMIT_WAIT)

C++ Client

Call KuduSession::SetExternalConsistencyMode(COMMIT_WAIT)

Warning:

COMMIT_WAIT consistency is an experimental feature. It may return incorrect results, exhibit performance issues, or negatively impact cluster stability. Its use in production environments is discouraged.

Read operations (scans)

Scans are read operations performed by clients that may span one or more rows across one or more tablets. When a server receives a scan request, it takes a snapshot of the MVCC state and then proceeds in one of two ways depending on the read mode selected by the user.

The mode may be selected as follows:

Java Client

Call KuduScannerBuilder#ReadMode(…)

C++ Client

Call KuduScanner::SetReadMode()

The following modes are available in both clients:

READ_LATEST

This is the default read mode. The server takes a snapshot of the MVCC state and proceeds with the read immediately. Reads in this mode only yield 'Read Committed' isolation.

READ_AT_SNAPSHOT

In this read mode, scans are consistent and repeatable. A timestamp for the snapshot is selected either by the server, or set explicitly by the user through KuduScanner::SetSnapshotMicros(). Explicitly setting the timestamp is recommended.

The server waits until this timestamp is 'safe'; that is, until all write operations that have a lower timestamp have completed and are visible). This delay, coupled with an external consistency method, will eventually allow Kudu to have full strict-serializable semantics for reads and writes. However, this is still a work in progress and some anomalies are still possible. Only scans in this mode can be fault-tolerant.

READ_YOUR_WRITES

This read mode relies on the state of a Kudu client to issue subsequent scan requests. When issuing a scan request in this read mode, a Kudu client provides the latest timestamp it observed so far. The server selects a timestamp higher than the timestamp provided by the client, that is also guaranteed to have all prior write operations committed and applied to the data. That translates into read-your-writes and read-your-reads behavior which is useful in scenarios where subsequent scan requests

should contain the data the client has seen so far while reading and writing during its current session. To summarize, this read mode

- ensures read-your-writes and read-your-reads session guarantees
- minimizes the latency caused by waiting for outstanding write operations at the server side to complete
- does not guarantee linearizability

Selecting between read modes requires balancing the trade-offs and making a choice that fits your workload. For instance, a reporting application that needs to scan the entire database might need to perform careful accounting operations, so that scan may need to be fault-tolerant, but probably doesn't require a to-the-microsecond up-to-date view of the database. In that case, you might choose READ_AT_SNAPSHOT and select a timestamp that is a few seconds in the past when the scan starts. On the other hand, a machine learning workload that is not ingesting the whole data set and is already statistical in nature might not require the scan to be repeatable, so you might choose READ_LATEST instead for better scan performance.

Note:

Kudu also provides replica selection API for you to choose at which replica the scan should be performed:

Java Client

Call KuduScannerBuilder#replicaSelection(...)

C++ Client

Call KuduScanner::SetSelection(...)

This API is a means to control locality and, in some cases, latency. The replica selection API has no effect on the consistency guarantees, which will hold no matter which replica is selected.

Related Information

Known issues and limitations

Known issues and limitations

There are several gaps and corner cases that currently prevent Kudu from being strictly-serializable in certain situations.

Writes

Support for COMMIT_WAIT is experimental and requires careful tuning of the time-synchronization protocol, such as NTP (Network Time Protocol). Its use in production environments is discouraged.

Recommendation

If external consistency is a requirement and you decide to use COMMIT_WAIT, the time-synchronization protocol needs to be tuned carefully. Each transaction will wait 2x the maximum clock error at the time of execution, which is usually in the 100 msec. to 1 sec. range with the default settings, maybe more. Thus, transactions would take at least 200 msec. to 2 sec. to complete when using the default settings and may even time out.

- A local server should be used as a time server. We have performed experiments using the default NTP time source available in a Google Compute Engine data center and were able to obtain a reasonable tight max error bound, usually varying between 12-17 milliseconds.
- The following parameters should be adjusted in /etc/ntp.conf to tighten the maximum error:
  - server my_server.org iburst minpoll 1 maxpoll 8
  - tinker dispersion 500
  - tinker allan 0

Reads (scans)

On a leader change, READ_AT_SNAPSHOT scans at a snapshot whose timestamp is beyond the last write, may yield non-repeatable reads (see KUDU-1188).

Recommendation

If repeatable snapshot reads are a requirement, use READ_AT_SNAPSHOT with a timestamp that is slightly in the past (between 2-5 seconds, ideally). This will circumvent the anomaly described above. Even when the anomaly has been addressed, back-dating the timestamp will always make scans faster, since they are unlikely to block.

Impala scans are currently performed as READ_LATEST and have no consistency guarantees.

In AUTO_BACKGROUND_FLUSH mode, or when using "async" flushing mechanisms, writes applied to a single client session may get reordered due to the concurrency of flushing the data to the server. This is particularly noticeable if a single row is quickly updated with different values in succession. This phenomenon affects all client API implementations. Workarounds are described in the respective API documentation for FlushMode or AsyncKud uSession. See KUDU-1767.

Scaling Kudu

This section describes in detail how Kudu scales with respect to various system resources, including memory, file descriptors, and threads. See *Scaling recommendations and limitations* for the maximum recommended parameters of a Kudu cluster. They can be used to estimate roughly the number of servers required for a given quantity of data.

Attention: The recommendations and conclusions here are only approximations. Appropriate numbers depend on use case. There is no substitute for measurement and monitoring of resources used during a representative workload.

Related Information

Scaling recommendations and limitations

Terms

Here are some of the terms used in this topic.

- hot replica: A tablet replica that is continuously receiving writes. For example, in a time series use case, tablet replicas for the most recent range partition on a time column would be continuously receiving the latest data, and would be hot replicas.
- cold replica: A tablet replica that is not hot, i.e. a replica that is not frequently receiving writes, for example, once every few minutes. A cold replica may be read from. For example, in a time series use case, tablet replicas for previous range partitions on a time column would not receive writes at all, or only occasionally receive late updates or additions, but may be constantly read.
- data on disk: The total amount of data stored on a tablet server across all disks, post-replication, post-compression, and post-encoding.

Example workload

The sections below help you perform sample calculations using the following parameters:

- 200 hot replicas per tablet server
- 1600 cold replicas per tablet server
- 8TB of data on disk per tablet server (about 4.5GB/replica)
- 512MB block cache
- 40 cores per server
- limit of 32000 file descriptors per server

- a read workload with 1 frequently-scanned table with 40 columns

This workload resembles a time series use case, where the hot replicas correspond to the most recent range partition on time.

Memory

The flag --memory_limit_hard_bytes determines the maximum amount of memory that a Kudu tablet server may use. The amount of memory used by a tablet server scales with data size, write workload, and read concurrency.

The following table provides numbers that can be used to compute a rough estimate of memory usage:

Table 2: Tablet server memory usage

Type	Multiplier	Description
Memory required per TB of data on disk	1.5GB per 1TB data on disk	Amount of memory per unit of data on disk required for basic operation of the tablet server.
Hot Replicas' MemRowSets and DeltaMemStores	minimum 128MB per hot replica	Minimum amount of data to flush per MemRowSet flush. For most use cases, updates should be rare compared to inserts, so the DeltaMemStores should be very small.
Scans	256KB per column per core for read-heavy tables	Amount of memory used by scanners, and which will be constantly needed for tables which are constantly read.
Block Cache	Fixed by --block_cache_capacity_mb(default 512MB)	Amount of memory reserved for use by the block cache.

Using this information for the example load gives the following breakdown of memory usage:

Table 3: Example tablet server memory usage

Type	Amount
8TB data on disk	8TB * 1.5GB / 1TB = 12GB
200 hot replicas	200 * 128MB = 25.6GB
1 40-column, frequently-scanned table	40 * 40 * 256KB = 409.6MB
Block Cache	--block_cache_capacity_mb=512 = 512MB
Expected memory usage	38.5GB
Recommended hard limit	52GB

Using this as a rough estimate of Kudu's memory usage, select a memory limit so that the expected memory usage of Kudu is around 50-75% of the hard limit.

Verifying if a memory limit is sufficient

After configuring an appropriate memory limit with --memory_limit_hard_bytes, run a workload and monitor the Kudu tablet server process's RAM usage. The memory usage should stay around 50-75% of the hard limit, with occasional spikes above 75% but below 100%. If the tablet server runs above 75% consistently, the memory limit should be increased.

Additionally, it's also useful to monitor the logs for memory rejections, which look like:

```
Service unavailable: Soft memory limit exceeded (at 96.35% of capacity)
```

And watch the memory rejections metrics:

- leader_memory_pressure_rejections
- follower_memory_pressure_rejections
- transaction_memory_pressure_rejections

Occasional rejections due to memory pressure are fine and act as backpressure to clients. Clients will transparently retry operations. However, no operations should time out.

File descriptors

Processes are allotted a maximum number of open file descriptors (also referred to as fds). If a tablet server attempts to open too many fds, it will typically crash with a message saying something like "too many open files".

The following table summarizes the sources of file descriptor usage in a Kudu tablet server process:

Table 4: Tablet server file descriptor usage

Type	Multiplier	Description
File cache	Fixed by --block_manager_max_open_files (default 40% of process maximum)	Maximum allowed open fds reserved for use by the file cache.
Hot replicas	2 per WAL segment, 1 per WAL index	Number of fds used by hot replicas. See below for more explanation.
Cold replicas	3 per cold replica	Number of fds used per cold replica: 2 for the single WAL segment and 1 for the single WAL index.

Every replica has at least one WAL segment and at least one WAL index, and should have the same number of segments and indices; however, the number of segments and indices can be greater for a replica if one of its peer replicas is falling behind. WAL segment and index fds are closed as WALs are garbage collected.

Using this information for the example load gives the following breakdown of file descriptor usage, under the assumption that some replicas are lagging and using 10 WAL segments:

Table 5: Example tablet server file descriptor usage

Type	Amount
file cache	40% * 32000 fds = 12800 fds
1600 cold replicas	1600 cold replicas * 3 fds / cold replica = 4800 fds
200 hot replicas	(2 / segment * 10 segments/hot replica * 200 hot replicas) + (1 / index * 10 indices / hot replica * 200 hot replicas) = 6000 fds
Total	23600 fds

So for this example, the tablet server process has about 32000 - 23600 = 8400 fds to spare.

There is typically no downside to configuring a higher file descriptor limit if approaching the currently configured limit.

Threads

Processes are allotted a maximum number of threads by the operating system, and this limit is typically difficult or impossible to change. Therefore, this section is more informational than advisory.

If a Kudu tablet server’s thread count exceeds the OS limit, it will crash, usually with a message in the logs like "pthread_create failed: Resource temporarily unavailable". If the system thread count limit is exceeded, other processes on the same node may also crash.

Threads and thread pools are used all over Kudu for various purposes, but the number of threads found in nearly all of these does not scale with load or data/tablet size; instead, the number of threads is either a hard coded constant, a constant defined by a configuration parameter, or based on a static dimension (such as the number of CPU cores).

The only exception to this is the WAL append thread, one of which exists for every "hot" replica.

Note that all replicas may be considered hot at startup, so tablet servers' thread usage will generally peak when started and settle down thereafter.

Scaling recommendations and limitations

Here are some of the recommended vales that you should consider while deciding the size of the tablet servers, masters, and other storage resources to get optimum performance.

Kudu can seamlessly run across a wide array of environments and workloads with minimal expertise and configuration at the following scale:

- Recommended maximum number of masters is 3.
- Recommended maximum number of tablet servers is 100.
- Recommended maximum amount of stored data, post-replication and post-compression, per tablet server is 8 TiB.
- Recommended number of tablets per tablet server is 1000 (post-replication) with 2000 being the maximum number of tablets allowed per tablet server.
- Maximum number of tablets per table for each tablet server is 60, post-replication (assuming the default replication factor of 3), at table-creation time.
- Recommended maximum amount of data per tablet is 50 GiB. Going beyond this can cause issues such a reduced performance, compaction issues, and slow tablet startup times.

  The recommended target size for tablets is under 10 GiB.

Staying within these limits will provide the most predictable and straightforward Kudu experience. However, experienced users who run on modern hardware, use the latest versions of Kudu, test and tune Kudu for their use case, and work closely with the community, can achieve much higher scales comfortably. Following are some anecdotal values that have been seen in real world production clusters:

- Number of master servers: 3
- More than 300 tablet servers
- 10+ TiB of stored data per tablet server, post-replication and post-compression
- More than 4000 tablets per tablet server, post-replication
- 50 GiB of stored data per tablet. Going beyond this can cause issues such a reduced performance, compaction issues, and slower tablet startup time.

Backing up and recovering Apache Kudu

Date published:
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Kudu backup **4**
Back up tables 4
Backup tools 4
Generate a table list 5
Backup directory structure 5
Physical backups of an entire node 5

Kudu recovery **6**
Restore tables from backups 6
Recover from disk failure 7
Recover from full disks 8
Bring a tablet that has lost a majority of replicas back online 8
Rebuild a Kudu filesystem layout 9

Kudu backup

Kudu supports both full and incremental table backups through a job implemented using Apache Spark.

Kudu backup and restore jobs use Apache Spark. Therefore, ensure that you install Apache Spark in your environment. To download Apache Spark, see the *Apache Spark documentation*. You can also review the *Submitting Spark applications* topics.

Related Information

Apache Spark

Submitting Spark applications

Back up tables

You can use the KuduBackup Spark job to backup one or more Kudu tables.g

When you first run the job for a table, a full backup is run. Additional runs will perform incremental backups which will only contain the rows that have changed since the initial full backup. A new set of full backups can be forced at anytime by passing the --forceFull flag to the backup job.

Following are some of the common flags that you can use while taking a backup:

- --kuduMasterAddresses: This is used to specify a comma-separated addresses of Kudu masters. The default value is localhost.
- --rootPath: The root path is used to output backup data. It accepts any Spark-compatible path.

Note: You can see the full list of the job options by passing the --help flag.

The following code run a KuduBackup job which backs up the tables foo and bar to an HDFS directory:

```
spark-submit --class org.apache.kudu.backup.KuduBackup [***FULL PATH TO kudu
-backup2_2.11-1.12.0.jar***] \
  --kuduMasterAddresses [***KUDU MASTER HOSTNAME 1***]:7051,[***KUDU MASTER
 HOSTNAME 2***]:7051 \
  --rootPath hdfs:///[***DIRECTORY TO USE FOR BACKUP***] \
   impala::[***DATABASE NAME***].foo impala::[***DATABASE NAME***].bar
```

Backup tools

An additional kudu-backup-tools JAR is available to provide some backup exploration and garbage collection capabilities. This jar does not use Spark directly, but instead only requires the Hadoop classpath to run.

Commands:

- list: Lists the backups in the rootPath
- clean: Cleans up old backed up data in the rootPath

Note: You can see the full list of the job options by passing the --help flag.

Following is an example execution which prints the command options:

```
java -cp $(hadoop classpath):kudu-backup-tools-1.12.0.jar org.apache.kudu.ba
ckup.KuduBackupCLI --help
```

Generate a table list

To generate a list of tables to backup using the kudu table list tool along with grep can be useful.

Following is an example that generates a list of all tables that start with my_db.:

```
kudu table list <master_addresses> | grep "^my_db\.*" | tr '\n' ' '
```

Note: This list could be saved as a part of your backup process so that it can be used while restoring.

Backup directory structure

The backup directory structure in the rootPath is considered an internal detail and could change in future versions of Kudu. Additionally, the format and content of the data and metadata files is meant for the backup and restore process only and could change in future versions of Kudu. That said, understanding the structure of the backup rootPath and how it is used can be useful when working with Kudu backups.

The backup directory structure in the rootPath is as follows:

```
/<rootPath>/<tableId>-<tableName>/<backup-id>/
   .kudu-metadata.json
   part-*.<format>
```

- rootPath: Can be used to distinguish separate backup groups, jobs, or concerns
- tableId: The unique internal ID of the table being backed up
- tableName: The name of the table being backed up
  - Note: Table names are URL encoded to prevent pathing issues
- backup-id: A way to uniquely identify/group the data for a single backup run
- .kudu-metadata.json: Contains all of the metadata to support recreating the table, linking backups by time, and handling data format changes

  Written last so that failed backups will not have a metadata file and will not be considered at restore time or backup linking time.
- part-*.<format>: The data files containing the tables data.
  - Currently 1 part file per Kudu partition
  - Incremental backups contain an additional “RowAction” byte column at the end
  - Currently the only supported format/suffix is parquet

Physical backups of an entire node

Kudu does not provide a built-in physical backup and restore functionality yet. However, it is possible to create a physical backup of a Kudu node (either tablet server or master) and restore it later.

Note:

The node to be backed up must be offline during the procedure, or else the backed up (or restored) data will be inconsistent.

Certain aspects of the Kudu node (such as its hostname) are embedded in the on-disk data. As such, it’s not yet possible to restore a physical backup of a node onto another machine.

Procedure

1. Stop all Kudu processes in the cluster. This prevents the tablets on the backed up node from being rereplicated elsewhere unnecessarily.
2. If creating a backup, make a copy of the WAL, metadata, and data directories on each node to be backed up. It is important that this copy preserve all file attributes as well as sparseness.
3. If restoring from a backup, delete the existing WAL, metadata, and data directories, then restore the backup via move or copy. As with creating a backup, it is important that the restore preserve all file attributes and sparseness.
4. Start all Kudu processes in the cluster.

Kudu recovery

Kudu supports restoring tables from full and incremental backups through a restore job implemented using Apache Spark.

Kudu backup and restore jobs use Apache Spark. Therefore, ensure that you install Apache Spark in your environment. To download Apache Spark, see the *Apache Spark documentation*. You can also review the *Submitting Spark applications* topics.

Related Information

Apache Spark

Submitting Spark applications

Restore tables from backups

You can use the KuduRestore Spark job to restore one or more Kudu tables. For each backed up table, the KuduRestore job restores the full backup and each associated incremental backup until the full table state is restored.

Restoring the full series of full and incremental backups is possible because the backups are linked via the from_ms and to_ms fields in the backup metadata. By default the restore job will create tables with the same name as the table that was backed up. If you want to side-load the tables without affecting the existing tables, you can pass the --tableSuffix flag to append a suffix to each restored table.

Following are the common flags that are used when restoring the tables:

- --rootPath: The root path to the backup data. Accepts any Spark-compatible path.

  See *Backup directory structure* for the directory structure used in the rootPath.
- --kuduMasterAddresses: Comma-separated addresses of Kudu masters. The default value is localhost.
- --createTables: If set to true, the restore process creates the tables. Set it to false if the target tables already exist. The default value is true.
- --tableSuffix: If set, it adds a suffix to the restored table names. Only used when createTables is true.
- --timestampMs: A UNIX timestamp in milliseconds that defines the latest time to use when selecting restore candidates. The default is System.currentTimeMillis().
- <table>…: A list of tables to restore.

Note: You can see the full list of the job options by passing the --help flag.

Following is an example of a KuduRestore job execution which restores the tables foo and bar from the HDFS directory kudu-backups:

```
spark-submit --class org.apache.kudu.backup.KuduRestore kudu-backup2_2.11-1.
12.0.jar \
  --kuduMasterAddresses master1-host,master-2-host,master-3-host \
  --rootPath hdfs:///kudu-backups \
```

```
  foo bar
```

Related Information

357f9ab119556938b3c92cfe8b2d61b3e1b99caa.xml

Recover from disk failure

Kudu nodes can only survive failures of disks on which certain Kudu directories are mounted. For more information about the different Kudu directory types, see the *Directory configuration* topic.

About this task

The following table summarizes the resilience to disk failure in different releases of Apache Kudu.

Table 1: Kudu disk failure behavior

Node Type	Kudu directory type	Kudu releases that crash on disk failure
Master	All	All
Tablet Server	Directory containing WALs	All
Tablet Server	Directory containing tablet metadata	All
Tablet Server	Directory containing data blocks only	Pre-1.6.0

When a disk failure occurs that does not lead to a crash, Kudu will stop using the affected directory, shut down tablets with blocks on the affected directories, and automatically re-replicate the affected tablets to other tablet servers. The affected server will remain alive and print messages to the log indicating the disk failure, for example:

```
E1205 19:06:24.163748 27115 data_dirs.cc:1011] Directory /data/8/kudu/data m
arked as failed
E1205 19:06:30.324795 27064 log_block_manager.cc:1822] Not using report from
 /data/8/kudu/data: IO error: Could not open container 0a6283cab82d4e75848f4
9772d2638fe: /data/8/kudu/data/0a6283cab82d4e75848f49772d2638fe.metadata: Re
ad-only file system (error 30)
E1205 19:06:33.564638 27220 ts_tablet_manager.cc:946] T 4957808439314e0d9
7795c1394348d80 P 70f7ee61ead54b1885d819f354eb3405: aborting tablet bootstra
p: tablet has data in a failed directory
```

While in this state, the affected node will avoid using the failed disk, leading to lower storage volume and reduced read parallelism. The administrator can remove the failed directory from the --fs_data_dirs gflag to avoid seeing these errors.

When the disk is repaired, remounted, and ready to be reused by Kudu, take the following steps:

Procedure

1. Make sure that the Kudu portion of the disk is completely empty.
2. Stop the tablet server.
3. Update the --fs_data_dirs gflag to add /data/3, on the server with the disk failure. For example,

   ```
   $ sudo -u kudu kudu fs update_dirs --force --fs_wal_dir=/wals --fs_data_
   dirs=/data/1,/data/2,/data/3
   ```

4. Start the tablet server.
5. Run ksck to verify cluster health, on any kudu server in the cluster. For example:

   ```
   $ sudo -u kudu kudu cluster ksck master-01.example.com
   ```

What to do next

Note: Note that existing tablets will not stripe to the restored disk, but any new tablets will stripe to the restored disk.

potentially using the update_dirs tool if on a version of Kudu that is below 1.12

Related Information

Directory configuration

Changing directory configuration

Recover from full disks

By default, Kudu reserves a small amount of space, 1% by capacity, in its directories. Kudu considers a disk full if there is less free space available than the reservation. Kudu nodes can only tolerate running out of space on disks on which certain Kudu directories are mounted.

The following table describes this behavior for each type of directory. The behavior is uniform across masters and tablet servers.

Kudu Directory Type	Crash on Full Disk?
Directory containing WALs	Yes
Directory containing tablet metadata	Yes
Directory containing data blocks only	No (see below)

Prior to Kudu 1.7.0, Kudu stripes tablet data across all directories, and will avoid writing data to full directories. Kudu will crash if all data directories are full.

In 1.7.0 and later, new tablets are assigned a disk group consisting of data directories. The number of data directories are as specified by the -fs_target_data_dirs_per_tablet flag with the default being 3. If Kudu is not configured with enough data directories for a full disk group, all data directories are used. When a data directory is full, Kudu will stop writing new data to it and each tablet that uses that data directory will write new data to other data directories within its group. If all data directories for a tablet are full, Kudu will crash. Periodically, Kudu will check if full data directories are still full, and will resume writing to those data directories if space has become available.

If Kudu does crash because its data directories are full, freeing space on the full directories will allow the affected daemon to restart and resume writing. Note that it may be possible for Kudu to free some space by running:

```
$ sudo -u kudu kudu fs check --repair
```

However, the above command may also fail if there is too little space left.

It is also possible to allocate additional data directories to Kudu in order to increase the overall amount of storage available. Note that existing tablets will not use new data directories, so adding a new data directory does not resolve issues with full disks.

Related Information

Directory configuration

Changing directory configuration

Bring a tablet that has lost a majority of replicas back online

If a tablet has permanently lost a majority of its replicas, it cannot recover automatically and operator intervention is required. If the tablet servers hosting a majority of the replicas are down (i.e. ones reported as "TS unavailable" by ksck), they should be recovered instead if possible.

Attention: The steps below may cause recent edits to the tablet to be lost, potentially resulting in permanent data loss. Only attempt the procedure below if it is impossible to bring a majority back online.

Suppose a tablet has lost a majority of its replicas. The first step in diagnosing and fixing the problem is to examine the tablet's state using ksck:

```
$ sudo -u kudu kudu cluster ksck --tablets=e822cab6c0584bc0858219d1539a17e6
master-00,master-01,master-02
Connected to the Master
Fetched info from all 5 Tablet Servers
Tablet e822cab6c0584bc0858219d1539a17e6 of table 'my_table' is unavailable:
 2 replica(s) not RUNNING
  638a20403e3e4ae3b55d4d07d920e6de (tserver-00:7150): RUNNING
  9a56fa85a38a4edc99c6229cba68aeaa (tserver-01:7150): bad state
    State:       FAILED
    Data state:  TABLET_DATA_READY
    Last status: <failure message>
  c311fef7708a4cf9bb11a3e4cbcaab8c (tserver-02:7150): bad state
    State:       FAILED
    Data state:  TABLET_DATA_READY
    Last status: <failure message>
```

This output shows that, for tablet e822cab6c0584bc0858219d1539a17e6, the two tablet replicas on tserver-01 and tser ver-02 failed. The remaining replica is not the leader, so the leader replica failed as well. This means the chance of data loss is higher since the remaining replica on tserver-00 may have been lagging. In general, to accept the potential data loss and restore the tablet from the remaining replicas, divide the tablet replicas into two groups:

1. Healthy replicas: Those in RUNNING state as reported by ksck
2. Unhealthy replicas

For example, in the above ksck output, the replica on tablet server tserver-00 is healthy while the replicas on tserver-01 and tserver-02 are unhealthy. On each tablet server with a healthy replica, alter the consensus configuration to remove unhealthy replicas. In the typical case of 1 out of 3 surviving replicas, there will be only one healthy replica, so the consensus configuration will be rewritten to include only the healthy replica.

```
$ sudo -u kudu kudu remote_replica unsafe_change_config tserver-00:7150 <tab
let-id> <tserver-00-uuid>
```

where <tablet-id> is e822cab6c0584bc0858219d1539a17e6 and <tserver-00-uuid> is the uuid of tserver-00, 638a 20403e3e4ae3b55d4d07d920e6de.

Once the healthy replicas' consensus configurations have been forced to exclude the unhealthy replicas, the healthy replicas will be able to elect a leader. The tablet will become available for writes though it will still be under-replicated. Shortly after the tablet becomes available, the leader master will notice that it is under-replicated, and will cause the tablet to re-replicate until the proper replication factor is restored. The unhealthy replicas will be tombstoned by the master, causing their remaining data to be deleted.

Rebuild a Kudu filesystem layout

In the event that critical files are lost, i.e. WALs or tablet-specific metadata, all Kudu directories on the server must be deleted and rebuilt to ensure correctness. Doing so will destroy the copy of the data for each tablet replica hosted on the local server. Kudu will automatically re-replicate tablet replicas removed in this way, provided the replication factor is at least three and all other servers are online and healthy.

About this task

Note: These steps use a tablet server as an example, but the steps are the same for Kudu master servers.

Warning: If multiple nodes need their FS layouts rebuilt, wait until all replicas previously hosted on each node have finished automatically re-replicating elsewhere before continuing. Failure to do so can result in permanent data loss.

Attention: Before proceeding, ensure the contents of the directories are backed up, either as a copy or in the form of other tablet replicas.

Procedure

1. The first step to rebuilding a server with a new directory configuration is emptying all of the server's existing directories. For example, if a tablet server is configured with --fs_wal_dir=/data/0/kudu-tserver-wal, --fs_metadata_dir=/data/0/kudu-tserver-meta, and --fs_data_dirs=/data/1/kudu-tserver,/data/2/kudu-tserver, the following commands will remove the WAL directory's and data directories' contents:

```
# Note: this will delete all of the data from the local tablet server.
$ rm -rf /data/0/kudu-tserver-wal/* /data/0/kudu-tserver-meta/* /data/1/kudu-tserver/* /data/2/kudu-tserver/*
```

2. If using Cloudera Manager, update the configurations for the rebuilt server to include only the desired directories. Make sure to only update the configurations of servers to which changes were applied, rather than of the entire Kudu service.
3. After directories are deleted, the server process can be started with the new directory configuration. The appropriate sub-directories will be created by Kudu upon starting up.

Managing Apache Kudu Security

Date published:
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Kudu security considerations 4
Proxied RPCs in Kudu 4

Kudu security limitations 5

Kudu authentication 5
Kudu authentication with Kerberos 5
Kudu authentication tokens 6
Client authentication to secure Kudu clusters 6
JWT authentication for Kudu 6
Configuring server side JWT authentication for Kudu 7
Configuring client side JWT authentication for Kudu 7

Kudu coarse-grained authorization 8

Kudu fine-grained authorization 9
Kudu and Apache Ranger integration 9
Kudu authorization tokens 10
Specifying trusted users 10
Kudu authorization policies 11
Ranger policies for Kudu 12

Disabling redaction 12

Configuring a secure Kudu cluster using Cloudera Manager 13
Enabling Kerberos authentication and RPC encryption 13
Configuring custom Kerberos principal for Kudu 14
Configuring coarse-grained authorization with ACLs 14
Configuring TLS/SSL encryption for Kudu using 15
Enabling Ranger authorization 15
Configuring HTTPS encryption 16
Configuring data at rest encryption 17

Kudu security considerations

Kudu includes security features that allow Kudu clusters to be hardened against access from unauthorized users.

Kudu uses strong authentication with Kerberos, and authorization with Ranger. Communication between Kudu clients and servers, and between servers to other servers, can be encrypted with TLS. Kudu also supports HTTPS on the web UI.

These security features should work seamlessly in Impala as well, as long as Impala's user is given permission to access Kudu.

With support of JWT for authenticating Kudu clients and RPC forwarding through a TCP proxy, more options are available for deploying secure Kudu clusters. For example, you can deploy a secure (for example, Kerberised) Kudu cluster in an internal or firewalled network and allow Kudu clients running outside of the network to connect to the cluster by authenticating themselves through JWT to Kudu servers, where all the RPC traffic is forwarded through a TCP proxy. In such a deployment, all the communications between Kudu clients running outside of the internal network and the Kudu cluster running in the internal network are protected by TLS, where the clients are provided with the cluster's IPKI CA certificate to authenticate the server side they are communicating with.

Proxied RPCs in Kudu

You can configure RPC endpoints for Kudu servers to separate internal and external traffic in a Kudu cluster to enable inter-cluster communications.

You can separate the internal and the external traffic in a Kudu cluster while providing the connectivity for Kudu clients running in external networks where the internal traffic is never routed through a proxy's or a loadbalancer's endpoint. Essentially, it allows for the internal traffic (for example, the traffic between tablet servers and masters) to bypass advertised RPC addresses, using alternative addresses for inter-cluster communications.

You can configure additional RPC endpoints for Kudu servers to handle traffic proxied from external networks. So, when a Kudu server receives an RPC request, it possesses enough information to decide whether to handle the request as arriving from an internal network or an external network. All the communications of Kudu components within the cluster's internal network are routed through the standard RPC endpoints. But the requests proxied from external networks are routed through those dedicated RPC endpoints.When a Kudu server receives an RPC request through such an endpoint, it sends a response where the internal RPC addresses of Kudu servers are substituted with corresponding RPC addresses reachable to Kudu clients outside the cluster through a TCP proxy.

To support proxied RPCs, the following flags are introduced; both flags accepting comma-separated list of strings in the <hostname>:<port> form:

- --rpc_proxy_advertised_addresses

  Sets the server's RPC endpoints exposed to the external network through a TCP proxy.
- --rpc_proxied_addresses

  Defines RPC endpoints in the internal network to handle RPC requests forwarded or proxied from external networks. It is possible to use a wild card for IP address (for example, 0.0.0.0) and the port number (for example, 0) for the elements of this address list.

The --rpc_proxy_advertised_addresses flag is orthogonal to the --rpc_advertised_addresses flag, so it is possible to use both simultaneously. For example, it might be useful if the network environment for docker containers in the private internal network is configured in some special way that mandates using the RPC-advertised addresses instead of the real RPC addresses for Kudu servers.

Kudu security limitations

Here are some limitations related to data encryption and authorization in Kudu.

- Kudu uses an internal PKI system to issue X.509 certificates to servers in the cluster. As a result, you cannot run Kudu with public IPs.
- Server certificates generated by Kudu IPKI are incompatible with bouncycastle version 1.52 and earlier.
- When you are creating a new Kudu service using the Ranger web UI, the Test Connection button is displayed. However, the TestConnection tab is not implemented in the Kudu Ranger plugin. As a result, if you try to use it with Kudu, it fails. But that does not mean that the service is not working.

Kudu authentication

Configure Kudu to enforce secure authentication among servers, and between clients and servers.

Authentication prevents untrusted actors from gaining access to Kudu, and securely identifies connecting users or services for authorization checks. Authentication in Kudu is designed to interoperate with other secure Hadoop components by utilizing Kerberos.

Scalability

Kudu authentication is designed to scale to thousands of nodes, which means it must avoid unnecessary coordination with a central authentication authority (such as the Kerberos KDC) for each connection.

Instead, Kudu servers and clients use Kerberos to establish initial trust with the Kudu master, and then use alternate credentials for subsequent connections. The Kudu master issues internal X.509 certificates to tablet servers on startup, and temporary authentication tokens to clients on first contact.

Internal private key infrastructure (PKI)

Kudu uses an internal PKI to issue X.509 certificates to servers in the cluster. Connections between peers who have both obtained certificates will use TLS for authentication. In such cases, neither peer needs to contact the Kerberos KDC.

X.509 certificates are only used for internal communication among Kudu servers, and between Kudu clients and servers. These certificates are never presented in a public facing protocol. By using internally-issued certificates, Kudu offers strong authentication which scales to huge clusters, and allows TLS encryption to be used without requiring you to manually deploy certificates on every node.

Kudu authentication with Kerberos

You can configure authentication on Kudu servers. Authentication in Kudu is designed to interoperate with other secure Hadoop components by utilizing Kerberos.

The --rpc_authentication flag is used to configure authentication on Kudu servers. In a Cloudera Manager managed cluster that flag is configured using the Enable Secure Authentication And Encryption property. This property can have the following values:

- Optional: Kudu will attempt to use strong authentication, but will allow unauthenticated connections. The connection between servers will be encrypted. This is the default value which is indicated by a clear checkbox.
- Required: Kudu will reject connections from clients and servers who lack authentication credentials. If you want to secure your cluster you have to set the --rpc_authentication flag to this value, meaning that you have to select the Enable Secure Authentication And Encryption property.

Kudu authentication tokens

After authenticating to a secure cluster, the Kudu client will automatically request an authentication token from the Kudu master. An authentication token encapsulates the identity of the authenticated user and carries the Kudu master's RSA signature so that its authenticity can be verified. This token will be used to authenticate subsequent connections.

By default, authentication tokens are only valid for seven days, so that even if a token were compromised, it cannot be used indefinitely. For the most part, authentication tokens should be completely transparent to users. By using authentication tokens, Kudu is able to take advantage of strong authentication, without paying the scalability cost of communicating with a central authority for every connection.

When used with distributed compute frameworks such as Apache Spark, authentication tokens can simplify configuration and improve security. For example, the Kudu Spark connector will automatically retrieve an authentication token during the planning stage, and distribute the token to tasks. This allows Spark to work against a secure Kudu cluster where only the planner node has Kerberos credentials.

Client authentication to secure Kudu clusters

Users running client Kudu applications must first run the kinit command to obtain a Kerberos ticket-granting ticket.

For example: kinit admin@EXAMPLE.COM

Once authenticated, you use the same client code to read from and write to Kudu servers with and without the Kerberos configuration.

JWT authentication for Kudu

JSON Web Token (JWT) authentication is a standard way of securely transmitting signed information between two parties. Learn how Kudu clients can use JWT to authenticate to a Kudu cluster.

JWT is a compact, URL-safe mechanism of representing claims to be transferred between two parties (for example, Kudu server and Kudu client). The claims in a JWT token are encoded either as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext format of a JSON Web Encryption (JWE) structure. This encoding enables the claims to be digitally signed, integrity protected with a Message Authentication Code (MAC), and/or encrypted. The structure of JWT enables you to verify whether the content is tampered.

Kudu servers accept JWTs for authentication when run with the --enable_jwt_token_auth flag (by default, it is set to false). Correspondingly, all servers in a Kudu cluster should be able to access the JWT issuer's JSON Web Key Set (JWKS) to verify the JWTs presented to them by connecting clients. Kudu servers fetch JWKS from the location specified by the --jwks_url flag.

When to use JWT authentication

JWT authentication is an alternative to Kerberos authentication, and you can use it in situations where Kerberos authentication is not a viable option but authentication is required, as in the following examples,

- A Kudu client runs on a node outside the Kerberos realm used in the cluster's internal network, and cross-realm authentication cannot be configured.
- A Kudu client cannot access Kerberos servers in the Kudu cluster's internal network, but it can talk to those Kudu servers through a TCP proxy.
- A Kudu client runs on a node without any Kerberos environment configured.

How JWT authentication works

In Kudu, a client sends a signed and non-encrypted JWT to the Kudu server when negotiating a new RPC connection between them. The JWTs are sent over communication channels protected by TLS. The Kudu servers check the

validity of the JWTs by verifying their signatures. In its turn, a Kudu client confirms the authenticity of a Kudu server by verifying the signature of the server's TLS certificate. So, for JWT-based authentication scenarios in Kudu, TLS certificates used by servers should be trusted by clients. If Kudu is deployed with its own certificate authority (that is a part of the Kudu's Internal Private Key Infrastructure, or IPKI), the Kudu's IPKI CA certificate should be added as trusted by using certain Kudu client API calls. For example, use KuduClientBuilder::trusted_certificate(const std::string&) in C++ Kudu applications to add a certificate in PEM format in the client's chain of trusted certificates. The Kudu's IPKI CA certificate can be retrieved from the /ipki-ca-cert HTTP/HTTPS endpoint of the embedded master's web server.

Configuring server side JWT authentication for Kudu

Learn how to configure server-side changes for JSON Web Token (JWT) authentication for Kudu.

Before you begin

JWT authentication is an experimental feature. To change the flags mentioned in the following steps, you need to add --unlock_experimental_flags to the Kudu safety valve configuration.

Procedure

1. Log in to Cloudera Manager.
2. Select the Kudu service.
3. Click Configurations.
4. Search for Kudu Service Advanced Configuration Snippet (Safety Valve) for gflagfile, and configure the required configuration parameters for JWT authentication:

   ```
   --unlock_experimental_flags
   --enable_jwt_token_auth=true
   --jwks_url=...
   ```

   The following list describes the configuration flags available:

   enable_jwt_token_auth

   Valid option: Boolean

   Default value: false

   Description: This enables JWT authentication. The server expects a valid JWT to be sent by the client which is verified when the connection is being established. When true, reads the JWT token out of the RPC and extracts user name from the token payload.

   Either --jwks_file_path or --jwks_url (but not both) must be set when --enable_jwt_token_auth is set to true.

   jwks_file_path

   Valid option: String

   Description: File path of the pre-installed JSON Web Key Set (JWKS) for JWT verification.

   jwks_url

   Valid option: String

   Description: URL of the JWKS for JWT verification.
5. Add --trusted_certificate_file=... flag as well, if the JWKS server you are using does not have an SSL certificate signed by a CA that is trusted on the system level.

Configuring client side JWT authentication for Kudu

Learn how to configure client-side changes for JSON Web Token (JWT) authentication for Kudu.

Procedure

Set JWT and trusted certificate:

For C++ Client

```
 return KuduClientBuilder()
      .master_server_addrs(master_addrs)
      .default_admin_operation_timeout(MonoDelta::FromSeconds(20))
      .jwt(cJwt)
      .trusted_certificate(cKuduIpkiCaCert)
      .Build(client);
}
```

For Python Client

```
client = kudu.connect(self.master_hosts,
                              self.master_ports,
                              require_authentication=True,
                              jwt=jwtToken,
                              trusted_certificates=certs)
```

For Java Client

```
KuduClient client = new KuduClient.KuduClientBuilder(KUDU_MASTERS).build
();
client.jwt(jwtToken);
client.trustedCertificates(Arrays.asList(ByteString.copyFrom(ipkiCaCertDer
Bytes)));
```

Note: The Java client expects the trusted certificate to be in DER format, while the C++ client expects it to be in PEM format. The master server's Internal Private Key Infrastructure (IPKI) certificates are accessible on its webserver's /ipki-ca-cert-der and /ipki-ca-cert endpoints respectively.

Kudu coarse-grained authorization

Kudu supports coarse-grained authorization checks for client requests based on the client's authenticated Kerberos principal (user or service). Access levels are granted based on whitelist-style Access Control Lists (ACLs), one for each level. Each ACL specifies a comma-separated list of users, or may be set to '*' to indicate that all authenticated users have access rights at the specified level.

The two levels of access which can be configured are:

- Superuser - Principals authorized as a superuser can perform certain administrative functions such as using the kudu command line tool to diagnose and repair cluster issues.
- User - Principals authorized as a user are able to access and modify all data in the Kudu cluster. This includes the ability to create, drop, and alter tables, as well as read, insert, update, and delete data. The default value for the User ACL is '*', which allows all users access to the cluster. However, if authentication is enabled, this will restrict access to only those users who are able to successfully authenticate using Kerberos. Unauthenticated users on the same network as the Kudu servers will be unable to access the cluster.

Note: Internally, Kudu has a third access level for the daemons themselves called Service. This is used to ensure that users cannot connect to the cluster and pose as tablet servers.

Kudu fine-grained authorization

Kudu can be configured to enforce fine-grained authorization across servers.

Fine-grained authorization ensures that users can see only the data they are explicitly authorized to see. Kudu supports this by leveraging policies defined in Apache Ranger.

Note: Fine-grained authorization policies are not enforced when accessing the web UI. User data may appear on various pages of the web UI (for example, in logs, metrics, scans). As such, it is recommended to either limit access to the web UI ports, or redact or disable the web UI entirely, as desired.

Kudu and Apache Ranger integration

Learn about how Apache Ranger is integrated with Kudu in order to provide fine-grained authorization across servers.

The Ranger models tabular objects are stored in a Kudu cluster in the following hierarchy: Database, Table, Column.

Note: Ranger allows you to add separate service repositories to manage privileges for different Kudu clusters. Depending on the value of the ranger.plugin.kudu.service.name configuration in the Ranger client, Kudu knows which service repository to connect to. For more details about Ranger service repository, see the Apache Ranger documentation.

Database: Kudu does not have the concept of a database. Therefore, a database is indicated as a prefix of table names with the format <database>.<table>. Since Kudu's only restriction on table names is that they be valid UTF-8 encoded strings, Kudu considers special characters to be valid parts of database or table names. For example, if a managed Kudu table created from Impala is named impala::bar.foo, its database will be impala::bar.

Table: Is a single Kudu table.

Column: Is a column within a Kudu table.

In Ranger, privileges are also associated with specific actions. Access to Kudu tables may rely on privileges on the following actions:

- ALTER
- CREATE
- DELETE
- DROP
- INSERT
- UPDATE
- SELECT

There are two additional access types:

- ALL
- METADATA

If a user has the ALL privilege on a resource, they implicitly have privileges to perform any action on that resource that does not require the users to be a delegated admin.

If a user is granted any privilege, they are able to perform actions requiring METADATA (for example, opening a table) without having to explicitly grant them METADATA privileges.

Ranger supports a delegate admin flag which is independent of the action type. It is not implied by ALL and does not imply METADATA. This is similar to the GRANT OPTION part of the ALL WITH GRANT OPTION in SQL as it is required to modify privileges in Ranger and change the owner of a Kudu table.

Table creation requires CREATE ON DATABASE privilege. If the user creates a table with a different owner, ALL and delegate admin are required.

Warning: A user with delegate admin privilege on a resource can grant any privilege to themselves and others.

While the action types are hierarchical, in terms of privilege evaluation, Ranger does not have the concept of hierarchy. For instance, if a user has SELECT privilege on a database, it does not imply that the user has SELECT privilege on every table belonging to that database.

However, Ranger supports privilege wildcard matching. For example, db=a->table=* matches all the tables that belong to database a. Therefore, in Ranger users actually need the SELECT privilege granted on db=a->table=*->c olumn=* to allow SELECT on every table and every column in database a.

Nevertheless, with Ranger integration, when a Kudu master receives a request, it consults Ranger to determine what privileges a user has. And the required policies documented in the <<security.adoc#policy-for-kudu-masters, policy section>> are enforced to determine whether the user is authorized to perform the requested action or not.

Note: Even though Kudu table names remain case sensitive with Ranger integration, policies authorization is considered case-insensitive.

In addition to granting privileges to a user by username, privileges can also be granted to table owners using the special {OWNER} username. These policies are evaluated only when a user tries to perform an action on a table that they own. For example, a policy can be defined for the {OWNER} user and db=*->table=* resource, and it will automatically be applied when any table is accessed by its owner. This way administrators do not need to choose between creating policies one by one for each table, and granting access to a wide range of users.

Warning: If a user has ALL and delegate admin privileges on a table only by ownership and no privileges by username, they can effectively lock themselves out by giving away the ownership.

Related Information

Ranger User Guide

Kudu authorization tokens

Learn about authorization tokens which are used by Ranger to propagate and check privileges.

Rather than having every tablet server communicate directly with the underlying authorization service (Ranger), privileges are propagated and checked via authorization tokens. These tokens encapsulate what privileges a user has on a given table. Tokens are generated by the master and returned to Kudu clients upon opening a Kudu table. Kudu clients automatically attach authorization tokens when sending requests to tablet servers.

Authorization tokens are a means to limit the number of nodes directly accessing the authorization service to retrieve privileges. As such, since the expected number of tablet servers in a cluster is much higher than the number of Kudu masters, they are only used to authorize requests sent to tablet servers. Kudu masters fetch privileges directly from the authorization service or cache.

Similar to the validity interval for authentication tokens, to limit the window of potential unwanted access if a token becomes compromised, authorization tokens are valid for five minutes by default. The acquisition and renewal of a token is hidden from the user, as Kudu clients automatically retrieve new tokens when existing tokens expire.

When a tablet server that has been configured to enforce fine-grained access control receives a request, it checks the privileges in the attached token, rejecting it if the privileges are not sufficient to perform the requested operation, or if it is invalid (e.g. expired).

Specifying trusted users

You can specify which users can view and modify data stored in Kudu. Additionally, some services that interact with Kudu may authorize requests on behalf of their end users.

About this task

It may be desirable to allow certain users to view and modify any data stored in Kudu. Such users can be specified via the --trusted_user_acl master configuration. Trusted users can perform any operation that would otherwise require fine-grained privileges, without Kudu consulting the authorization service.

Furthermore, some services that interact with Kudu can authorize requests on behalf of their end users. For example, Apache Impala authorizes queries on behalf of its users, and sends requests to Kudu as the Impala service user, commonly "impala". Since Impala authorizes requests on its own, to avoid extraneous communication between the authorization service and Kudu, the Impala service user should be listed as a trusted user.

Note: When accessing Kudu through Impala, Impala enforces its own fine-grained authorization policy. This policy is similar to Kudu's and can be found in the Impala authorization documentation.

Procedure

1. In Cloudera Manager, navigate to Kudu Configuration .
2. Find the Master Advanced Configuration Snippet (Safety Valve) for gflagfile propertz.
3. Click View as XML and add the following configuration:

```
--trusted_user_acl=impala,hive,kudu,rangeradmin,[***TRUSTED USER***]
```

4. Click Save Changes.
5. Restart the Kudu service.

Kudu authorization policies

Review the authorization policies that are enforced by Kudu masters and Kudu tablet servers.

Policy for Kudu masters

The following authorization policy is enforced by Kudu masters:

Table 1: Authorization Policy for Masters

Operation	Required Privilege
CreateTable	CREATE ON DATABASE
CreateTable with a different owner specified than the requesting user	ALL ON DATABASE with the Sentry GRANT OPTION.
DeleteTable	DROP ON TABLE
AlterTable (with no rename)	ALTER ON TABLE
AlterTable (with rename)	ALL ON TABLE <old-table> and CREATE ON DATABASE <new-database>
IsCreateTableDone	METADATA ON TABLE
IsAlterTableDone	METADATA ON TABLE
ListTables	METADATA ON TABLE
GetTableLocations	METADATA ON TABLE
GetTableSchema	METADATA ON TABLE
GetTabletLocations	METADATA ON TABLE

Policy for Kudu tablet servers

The following authorization policy is enforced by Kudu tablet servers:

Table 2: Authorization Policy for Tablet Servers

Operation	Required Privilege
Scan	SELECT ON TABLE, or METADATA ON TABLE and SELECT ON COLUMN for each projected column and each predicate column
Scan (no projected columns, equivalent to COUNT(*))	SELECT ON TABLE, or SELECT ON COLUMN for each column in the table
Scan (with virtual columns)	SELECT ON TABLE, or SELECT ON COLUMN for each column in the table
Scan (in ORDERED mode)	<privileges required for a Scan> and SELECT ON COLUMN for each primary key column
Insert	INSERT ON TABLE
Update	UPDATE ON TABLE
Upsert	INSERT ON TABLE and UPDATE ON TABLE
Delete	DELETE ON TABLE
SplitKeyRange	SELECT ON COLUMN for each primary key column and SELECT ON COLUMNfor each projected column
Checksum	User must be configured in --superuser_acl
ListTablets	User must be configured in --superuser_acl

Note: Unlike Impala, Kudu only supports all-or-nothing access to a table's schema, rather than showing only authorized columns.

Ranger policies for Kudu

There are two Kudu related Ranger policies which are applied based on how you are accessing Kudu.

There are two resource-based services in Ranger that are used in relation to Kudu: cm_kudu and Hadoop SQL.

The Kudu service and its connected clients, such as Spark, native C++, and Java clients, use the cm_kudu resource-based service.

The Hadoop SQL resource-based service is used by Hive and Impala when Kudu is accessed through them.

When Kudu is accessed by Impala, the Impala service performs actions as the impala user in Kudu. The impala user is set as a trusted user in Kudu, meaning that privilege checks are completely bypassed and the impala user is granted full access. As a result, the cm_kudu resource-based service is not applied, only the Hadoop SQL resource-based service is used to check for permission and privileges.

As a result, when you are accessing Kudu through Hive or Impala, you must ensure that all applicable permission and privileges are configured in the Hadoop SQL resource-based service.

Note: Ranger provides a client side cache that uses privileges and can periodically poll the privilege store for any changes. When a change is detected, the cache is automatically updated.

Disabling redaction

Data redaction is used to prevent sensitive data from being included in Kudu server logs or in the web UI. Data redaction is enabled by default but you can disable it using the –redact or the –webserver_enabled flag.

Log redaction

To prevent sensitive data from being included in Kudu server logs, all row data is redacted. You can turn off log redaction using the --redact flag.

Web UI redaction

To prevent sensitive data from being included in the web UI, all row data is redacted. Table metadata, such as table names, column names, and partitioning information is not redacted. Alternatively, you can choose to completely disable the web UI by setting the --webserver_enabled flag to false on the Kudu servers.

Note: Disabling the web UI will also disable REST endpoints such as /metrics. Monitoring systems rely on these endpoints to gather metrics data.

Procedure

1. In Cloudera Manager, navigate to Kudu Configuration .
2. Find the Kudu Service Advanced Configuration Snippet (Safety Valve) for gflagfile property.
3. If you want to disable log redaction, add the following configuration:

```
--redact=false
```

4. If you want to Web UI disable redaction, add the following configuration:

```
--webserver_enabled=false
```

5. Click Save Changes.
6. Restart the Kudu service.

Configuring a secure Kudu cluster using Cloudera Manager

You can configure a secure Kudu cluster using Cloudera Manager For that you need enabled Kerberos authentication and RPC encryption, configure coarse-grained authorization, and configure HTTPS encryption. Optionally you can configure custom Kerberos principal, TLS/SSL encryption or fine-grained authorization using Ranger.

Enabling Kerberos authentication and RPC encryption

You must already have a secure Cloudera Manager cluster with Kerberos authentication enabled.

Procedure

1. In Cloudera Manager, navigate to Kudu Configuration .
2. In the Search field, type Kerberos to show the relevant properties.
3. Find and edit the following properties according to your cluster configuration:

Field	Usage Notes
Kerberos Principal	Set to the default principal, kudu. Currently, Kudu does not support configuring a custom service principal for Kudu processes.
Enable Secure Authentication And Encryption	Select this checkbox to enable authentication and RPC encryption between all Kudu clients and servers, as well as between individual servers. Only enable this property after you have configured Kerberos. If this is not selected, security is not enforced but secured clients are not rejected either and the connection will be encrypted if both sides support it.

4. Click Save Changes.
   An error message displayed that tells you the Kudu keytab is missing.
5. Navigate to Administration Security .
6. Select the Kerberos Credentials tab.

   On this page you will see a list of the existing Kerberos principals for services running on the cluster.
7. Click Generate Missing Credentials
   Once the Generate Missing Credentials command has finished running, you will see the Kudu principal added to the list.

Configuring custom Kerberos principal for Kudu

You can configure a custom Kerberos principal for Kudu using .

Procedure

1. In , navigate to Kudu Configuration .
2. Search for kerberos.
3. Find the Kerberos principal property and set it to the required name.
4. Click Save changes.
   Custom Kerberos principal for Kudu is applied for not just the Kudu server but also for the Hive and Impala clients.
5. For components that are not controlled by , configure the SASL protocol name.

   You must add the SASL protocol name configuration wherever the Kudu client properties, such as master addresses, are configured.

   For example, if the principal name is kudu/_HOST set the protocol to kudu.

Configuring coarse-grained authorization with ACLs

The coarse-grained authorization can be configured with the following two ACLs: the Superuser Access Control List and the User Access Control List. The Superuser ACL is the list of all the superusers that can access the cluster. User-level access can be controlled by using the User ACL. By default, all the users can access the clusters. But when you enable authentication using Kerberos, only the users who are able to authenticate successfully can access the cluster.

Procedure

1. Go to the Kudu service.
2. Click the Configuration tab.
3. Select Category Security .
4. In the Search field, type ACL to show the relevant properties.
5. Edit the following properties according to your cluster configuration:

Field	Usage Notes
Superuser Access Control List	Add a comma-separated list of superusers who can access the cluster. By default, this property is left blank. '*' indicates that all authenticated users will be given superuser access.

Field	Usage Notes
User Access Control List	Add a comma-separated list of users who can access the cluster. By default, this property is set to '*'. The default value of '*' allows all users access to the cluster. However, if authentication is enabled, this will restrict access to only those users who are able to successfully authenticate using Kerberos. Unauthenticated users on the same network as the Kudu servers will be unable to access the cluster. Add the impala user to this list to allow Impala to query data in Kudu. You might choose to add any other relevant usernames if you want to give access to Spark Streaming jobs.

6. Click Save Changes.

Configuring TLS/SSL encryption for Kudu using

TLS/SSL encryption is enabled between Kudu servers and clients by default. You can enable TLS/SSL encryption for Kudu web UIs or configure the encryption using .

Procedure

1. In , navigate to Kudu Configuration .
2. In the Search field, type TLS/SSL to show the relevant properties.
3. Edit the following properties according to your cluster configuration:

Table 3: TLS/SSL Kudu properties

Property	Description
Master TLS/SSL Server Private Key File (PEM Format)	Set to the path containing the Kudu master host's private key (Privacy Enhanced Mail (PEM)-format). This is used to enable TLS/SSL encryption (over HTTPS) for browser-based connections to the Kudu master web UI.
Tablet Server TLS/SSL Server Private Key File (PEM Format)	Set to the path containing the Kudu tablet server host's private key (PEM-format). This is used to enable TLS/SSL encryption (over HTTPS) for browser-based connections to Kudu tablet server web UIs.
Master TLS/SSL Server Certificate File (PEM Format)	Set to the path containing the signed certificate (PEM-format) for the Kudu master host's private key (set in Master TLS/SSL Server Private Key File). The certificate file can be created by concatenating all the appropriate root and intermediate certificates required to verify trust.
Tablet Server TLS/SSL Server Certificate File (PEM Format)	Set to the path containing the signed certificate (PEM-format) for the Kudu tablet server host's private key (set in Tablet Server TLS/SSL Server Private Key File). The certificate file can be created by concatenating all the appropriate root and intermediate certificates required to verify trust.
Enable TLS/SSL for Master Server	Enables HTTPS encryption on the Kudu master web UI.
Enable TLS/SSL for Tablet Server	Enables HTTPS encryption on the Kudu tablet server web UIs.
Enable Secure Authentication, Encryption, and Web UI	Enables Kerberos for Kudu servers, clients and web servers, and enables encryption for Kudu servers and clients.

4. Click Save Changes.

Enabling Ranger authorization

You can configure fine-grained authorization using Apache Ranger. This topic provides the steps to enable Kudu's integration with Ranger from Cloudera Manager.

Procedure

1. From Cloudera Manager, go to Clusters Kudu Configurations .
2. Select the Ranger Service with which Kudu should authorize requests.
3. If Ranger high-availability is enabled for the cluster, add a Kudu service repository with the following configurations through the Ranger Admin web UI is required:

```
# This example setup configures the Kudu service user as a privileged user
 to be
# able to retrieve authorization policies stored in Ranger.

<property>
  <name>policy.download.auth.users</name>
  <value>kudu</value>
</property>
```

The name of the added Kudu service repository needs to match the one specified in ranger.plugin.kudu.service.name of the Ranger client ranger-kudu-security.xml configuration file.

Note: When a Kudu client opens a table, the Kudu Master will authorize all possible actions the user may want to perform on the given table (ALL, and if it's not allowed, then INSERT, SELECT, UDPATE, DELETE). This results in auditing these requests when a client opens a table, even if they'll never do any of these operations.

Configuring HTTPS encryption

Lastly, you enable TLS/SSL encryption (over HTTPS) for browser-based connections to both the Kudu master and tablet server web UIs.

Procedure

1. Go to the Kudu service.
2. Click the Configuration tab.
3. Select Category Security .
4. In the Search field, type TLS/SSL to show the relevant properties.
5. Edit the following properties according to your cluster configuration:

Field	Usage Notes
Master TLS/SSL Server Private Key File (PEM Format)	Set to the path containing the Kudu master host's private key (PEM-format). This is used to enable TLS/SSL encryption (over HTTPS) for browser-based connections to the Kudu master web UI.
Tablet Server TLS/SSL Server Private Key File (PEM Format)	Set to the path containing the Kudu tablet server host's private key (PEM-format). This is used to enable TLS/SSL encryption (over HTTPS) for browser-based connections to Kudu tablet server web UIs.
Master TLS/SSL Server Certificate File (PEM Format)	Set to the path containing the signed certificate (PEM-format) for the Kudu master host's private key (set in Master TLS/SSL Server Private Key File). The certificate file can be created by concatenating all the appropriate root and intermediate certificates required to verify trust.
Tablet Server TLS/SSL Server Certificate File (PEM Format)	Set to the path containing the signed certificate (PEM-format) for the Kudu tablet server host's private key (set in Tablet Server TLS/SSL Server Private Key File). The certificate file can be created by concatenating all the appropriate root and intermediate certificates required to verify trust.
Enable TLS/SSL for Master Server	Enables HTTPS encryption on the Kudu master web UI.
Enable TLS/SSL for Tablet Server	Enables HTTPS encryption on the Kudu tablet server Web UIs.

6. Click Save Changes.

Configuring data at rest encryption

You can enable data at rest encryption using Cloudera Manager However, you can enable it only for a fresh installation and once Kudu directories exist on the cluster you cannot disable the encryption.

About this task

Kudu supports data at rest encryption, specifically AES-128, AES-192 and AES-256 ciphers in CTR mode are supported to encrypt data.

When data is encrypted at rest, the following keys are used for encryption:

- File Key: The unique key used to encrypt a physical file
- Server Key: A server's own key used to encrypt the File Key
- Cluster Key: A key that is used to encrypt the Server Key and stored in a third-party Key Management Service (KMS). Kudu supports Apache Ranger KMS and Apache Hadoop KMS. Both are API-compatible

Before you begin

If Kudu was started before encryption was enabled, all Kudu data directories, WAL directories, and metadata directories (usually the same as WAL directories) must be deleted. This will result in a completely clean slate, so if there is any data in Kudu, it must be backed up using the backup tool before doing so. For more information, see *Kudu backup*.

Procedure

1. In Cloudera Manager navigate to Kudu Configuration .
2. Search for data at rest.
3. Find the Encrypt data at rest property and select it to enabe data at rest encryption.
4. Find the Data at rest encryption cipher property and select the cipher for the encryption.

   Its defaault value is AES-128-CTR.
5. Find the Data at rest encryption key name property and set its value to the name of the data at rest encryption key.

   Its default value is kudu_encryption_key.
6. Create a key in Ranger KMS with the same name as the value of the Data at rest encryption key name property.

   For more information, see *List and create keys*.
7. Ensure that the TLS certificate used by Ranger KMS is trusted on the OS-level by all Kudu hosts. This is required even if you use AutoTLS.

   For more information, consult your OS documentation.
8. Start the Kudu cluster.

What to do next

If you enabled data at encryption for a pre-existing cluster, restore the data you backed up before enabling encryption. For more information, see *Kudu recovery*.

Related Information

Kudu backup

Kudu recovery

Cloudera Runtime 7.2.11

Managing Apache HBase

Date published: 2020-02-29
Date modified: 2024-05-16

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Starting and stopping HBase using Cloudera Manager.....5
Start HBase.....5
Stop HBase.....5
Graceful HBase shutdown.....6
Gracefully shut down an HBase RegionServer.....6
Gracefully shut down the HBase service.....6

Importing data into HBase.....7
Choose the right import method.....7
Use snapshots.....7
Use CopyTable.....9
Use BulkLoad.....10
Use cases for BulkLoad.....11
Use cluster replication.....12
Use Sqoop.....12
Use Spark.....13
Use a custom MapReduce job.....14
Use HashTable and SyncTable Tool.....14
HashTable/SyncTable tool configuration.....14
Synchronize table data using HashTable/SyncTable tool.....15

Writing data to HBase.....16
Variations on Put.....16
Versions.....16
Deletion.....17
Examples.....17

Reading data from HBase.....18
Perform scans using HBase Shell.....19

HBase filtering.....20
Dynamically loading a custom filter.....20
Logical operators, comparison operators and comparators.....20
Compound operators.....21
Filter types.....21
HBase Shell example.....24
Java API example.....24

HBase online merge.....28

Move HBase Master Role to another host.....29

Expose HBase metrics to a Ganglia server **29**

HBase metrics **30**

Using JMX for accessing HBase metrics 30

Accessing HBase metrics in Prometheus format 30

Starting and stopping HBase using Cloudera Manager

You can start and stop your HBase cluster using Cloudera Manager.

Start HBase

You can start HBase clusters or individual hosts following these instructions.

Procedure

1. In Cloudera Manager, select the HBase service.
2. Click the Actions button.
3. Select Start.

If you want to restart a running cluster, do the following:

4. Click the Actions button.
5. Select Restart or Rolling Restart.
   A rolling restart restarts each RegionServer, on at a time, after a a grace period.

If you want to restart the Thrift service, do the following:

6. Click Instances.
7. Select the HBase Thrift Server instance.
8. Select Actions for Selected.
9. Select Restart.

Related Information

Configure the graceful shutdown timeout property

Stop HBase

You can stop HBase clusters or individual hosts following these instructions.

Procedure

If you want to stop a single RegionServer, do the following:

1. In Cloudera Manager, select the HBase service.
2. Click the Instance tab.
3. From the list of Role Instances, select the RegionServer or RegionServers you want to stop.
4. Select Actions for Selected.
5. Select Stop.
   Stop happens immediately and does not redistribute the regions. It issues a SIGTERM(kill -5) signal.

If you want to stop a single HMaster, do the following:

6. In Cloudera Manager, select the HBase service.
7. Click the Instance tab.
8. From the list of Role Instances, select the HMaster or HMasters you want to stop.
9. Select Actions for Selected.
10. Select Stop.
    Stop happens immediately and does not redistribute the regions. It issues a SIGTERM(kill -5) signal.

If you want to stop the entire cluster, do the following:

11. In Cloudera Manager, select the HBase service.
12. Click the Actions button.
13. Select Stop.

Related Information

Configure the graceful shutdown timeout property

Graceful HBase shutdown

Graceful Shutdown causes the regions to be redistributed to other RegionServers, increasing availability during the RegionServer outage.

A graceful shutdown of an HBase RegionServer allows the regions hosted by that RegionServer to be moved to other RegionServers before stopping the RegionServer. You can perform a graceful shutdown of either an HBase RegionServer or the entire service.

Cloudera Manager waits for an interval determined by the Graceful Shutdown timeout interval, which defaults to three minutes. If the graceful stop does not succeed within this interval, the RegionServer is stopped with a SIGKILL (kill -9) signal. Recovery will be initiated on affected regions.

To increase the speed of a rolling restart of the HBase service, set the Region Mover Threads property to a higher value. This increases the number of regions that can be moved in parallel, but places additional strain on the HMaster.

Related Information

Configure the graceful shutdown timeout property

Gracefully shut down an HBase RegionServer

You can gracefully shut down an HBase RegionServer using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the HBase service.
2. Click the Instance tab.
3. From the list of Role Instances, select the RegionServer or RegionServers you want to shut down gracefully.
4. Click Actions for Selected.
5. Select Decommission (Graceful Stop).

   If you cancel the graceful shutdown before the Graceful Shutdown Timeout expires, you can still manually stop a RegionServer by selecting Actions for Selected Stop , which sends a SIGTERM (kill -15) signal.

Related Information

Configure the graceful shutdown timeout property

Gracefully shut down the HBase service

You can gracefully shut down the HBase Service using Cloudera Manager.

Procedure

1. In Clouder Manager, select the HBase service.
2. Click the Actions button.
3. Select Stop.
   Cloudera Manager tries to perform an HBase Master-driven graceful shutdown for the length of the configured Graceful Shutdown Timeout, after which it abruptly shuts down the whole service. The default value of Graceful Shutdown Timeout is three minutes.

Importing data into HBase

In different scenarios different methods can be used to import data into HBase.

Choose the right import method

Learn about how to choose the right data import method.

The method you use for importing data into HBase depends on several factors:

- The location, size, and format of your existing data
- Whether you need to import data once or periodically over time
- Whether you want to import the data in bulk or stream it into HBase regularly
- How fresh the HBase data needs to be

This topic helps you choose the correct method or composite of methods and provides example workflows for each method.

Always run HBase administrative commands as the HBase user (typically hbase).

If the data is already in an HBase table:

- To move the data from one HBase cluster to another, use snapshot and either the clone_snapshot or ExportSnapsh ot utility; or, use the CopyTable utility.
- To move the data from one HBase cluster to another without downtime on either cluster, use replication.

If the data currently exists outside HBase:

- If possible, write the data to HFile format, and use a BulkLoad to import it into HBase. The data is immediately available to HBase and you can bypass the normal write path, increasing efficiency.
- If you prefer not to use bulk loads, and you are using a tool such as Pig, you can use it to import your data.

If you need to stream live data to HBase instead of import in bulk:

- Write a Java client using the Java API, or use the Apache Thrift Proxy API to write a client in a language supported by Thrift.
- Stream data directly into HBase using the REST Proxy API in conjunction with an HTTP client such as wget or curl.
- Use Flume or Spark.

Most likely, at least one of these methods works in your situation. If not, you can use MapReduce directly. Test the most feasible methods with a subset of your data to determine which one is optimal.

Use snapshots

A snapshot captures the state of a table at the time the snapshot was taken.

Cloudera recommends snapshots instead of CopyTable where possible. Because no data is copied when a snapshot is taken, the process is very quick. As long as the snapshot exists, cells in the snapshot are never deleted from HBase, even if they are explicitly deleted by the API. Instead, they are archived so that the snapshot can restore the table to its state at the time of the snapshot.

You can export snapshots from CDH 5 to , and from to CDH 5, if the the version of is 7.1 or higher.

After taking a snapshot, use the clone_snapshot command to copy the data to a new (immediately enabled) table in the same cluster, or the Export utility to create a new table based on the snapshot, in the same cluster. This is a copy-on-write operation. The new table shares HFiles with the original table until writes occur in the new table but not the

old table, or until a compaction or split occurs in either of the tables. This can improve performance in the short term compared to CopyTable.

To export the snapshot to a new cluster, use the ExportSnapshot utility, which uses MapReduce to copy the snapshot to the new cluster. Run the ExportSnapshot utility on the source cluster, as a user with HBase and HDFS write permission on the destination cluster, and HDFS read permission on the source cluster. This creates the expected amount of IO load on the destination cluster. Optionally, you can limit bandwidth consumption, which affects IO on the destination cluster. After the ExportSnapshot operation completes, you can see the snapshot in the new cluster using the list_snapshot command, and you can use the clone_snapshot command to create the table in the new cluster from the snapshot.

For full instructions for the snapshot and clone_snapshot HBase Shell commands, run the HBase Shell and type help snapshot. The following example takes a snapshot of a table, uses it to clone the table to a new table in the same cluster, and then uses the ExportSnapshot utility to copy the table to a different cluster, with 16 mappers and limited to 200 Mb/sec bandwidth.

```
$ bin/hbase shell
   hbase(main):005:0> snapshot 'TestTable', 'TestTableSnapshot'
   0 row(s) in 2.3290 seconds

   hbase(main):006:0> clone_snapshot 'TestTableSnapshot', 'NewTestTable'
   0 row(s) in 1.3270 seconds

   hbase(main):007:0> describe 'NewTestTable'
   DESCRIPTION                                          ENABLED
   'NewTestTable', {NAME => 'cf1', DATA_BLOCK_ENCODING true
   => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE
   => '0', VERSIONS => '1', COMPRESSION => 'NONE', MI
   N_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_C
   ELLS => 'false', BLOCKSIZE => '65536', IN_MEMORY =>
   'false', BLOCKCACHE => 'true'}, {NAME => 'cf2', DA
   TA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW',
   REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESS
   ION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER
   ', KEEP_DELETED_CELLS => 'false', BLOCKSIZE => '655
   36', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
   1 row(s) in 0.1280 seconds
   hbase(main):008:0> quit

   $ hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot TestTa
bleSnapshot -copy-to file:///tmp/hbase -mappers 16 -bandwidth 200
   14/10/28 21:48:16 INFO snapshot.ExportSnapshot: Copy Snapshot Manifest
   14/10/28 21:48:17 INFO client.RMProxy: Connecting to ResourceManager at a
1221.example.com/192.0.2.121:8032
   14/10/28 21:48:19 INFO snapshot.ExportSnapshot: Loading Snapshot 'TestT
ableSnapshot' hfile list
   14/10/28 21:48:19 INFO Configuration.deprecation: hadoop.native.lib is d
eprecated. Instead, use io.native.lib.available
   14/10/28 21:48:19 INFO util.FSVisitor: No logs under directory:hdfs://a12
21.example.com:8020/hbase/.hbase-snapshot/TestTableSnapshot/WALs
   14/10/28 21:48:20 INFO mapreduce.JobSubmitter: number of splits:0
   14/10/28 21:48:20 INFO mapreduce.JobSubmitter: Submitting tokens for job:
 job_1414556809048_0001
   14/10/28 21:48:20 INFO impl.YarnClientImpl: Submitted application appli
cation_1414556809048_0001
   14/10/28 21:48:20 INFO mapreduce.Job: The url to track the job: http://
a1221.example.com:8088/proxy/application_1414556809048_0001/
   14/10/28 21:48:20 INFO mapreduce.Job: Running job: job_1414556809048_0001
   14/10/28 21:48:36 INFO mapreduce.Job: Job job_1414556809048_0001 running
 in uber mode : false
   14/10/28 21:48:36 INFO mapreduce.Job:  map 0% reduce 0%
   14/10/28 21:48:37 INFO mapreduce.Job: Job job_1414556809048_0001 comple
ted successfully
```

```
   14/10/28 21:48:37 INFO mapreduce.Job: Counters: 2
   Job Counters
   Total time spent by all maps in occupied slots (ms)=0
   Total time spent by all reduces in occupied slots (ms)=0
   14/10/28 21:48:37 INFO snapshot.ExportSnapshot: Finalize the Snapshot
Export
   14/10/28 21:48:37 INFO snapshot.ExportSnapshot: Verify snapshot integrity
   14/10/28 21:48:37 INFO Configuration.deprecation: fs.default.name is depr
ecated. Instead, use fs.defaultFS
   14/10/28 21:48:37 INFO snapshot.ExportSnapshot: Export Completed: TestT
ableSnapshot
```

The url to track the job: contains the URL from which you can track the ExportSnapshot job. When it finishes, a new set of HFiles, comprising all of the HFiles that were part of the table when the snapshot was taken, is created at the HDFS location you specified.

You can use the SnapshotInfo command-line utility included with HBase to verify or debug snapshots.

Use CopyTable

CopyTable uses HBase read and write paths to copy part or all of a table to a new table in either the same cluster or a different cluster.

CopyTable causes read load when reading from the source, and write load when writing to the destination. Region splits occur on the destination table in real time as needed. To avoid these issues, use snapshot and export commands instead of CopyTable. Alternatively, you can pre-split the destination table to avoid excessive splits. The destination table can be partitioned differently from the source table. See this section of the Apache HBase documentation for more information.

Using CopyTable, you can copy a part or all of a table in a CDH 5 cluster to a given table in a cluster where the version is 7.1 or higher. However, you have to pass the Dhbase.meta.replicas.use=true command in the command line to make it work. For example:

```
hbase org.apache.hadoop.hbase.mapreduce.CopyTable -Dhbase.meta.replicas.use=
true ... <other options>
```

Using CopyTable to copy a part or all of a table from a cluster to a CDH 5 cluster does not work.

Edits to the source table after the CopyTable starts are not copied, so you may need to do an additional CopyTable operation to copy new data into the destination table. Run CopyTable as follows, using --help to see details about possible parameters.

```
$ ./bin/hbase org.apache.hadoop.hbase.mapreduce.CopyTable --help
    Usage: CopyTable [general options] [--starttime=X] [--endtime=Y] [--new
.name=NEW] [--peer.adr=ADR] <tablename>
```

The starttime/endtime and startrow/endrow pairs function in a similar way: if you leave out the first of the pair, the first timestamp or row in the table is the starting point. Similarly, if you leave out the second of the pair, the operation continues until the end of the table. To copy the table to a new table in the same cluster, you must specify --new.name, unless you want to write the copy back to the same table, which would add a new version of each cell (with the same data), or just overwrite the cell with the same value if the maximum number of versions is set to 1. To copy the table to a new table in a different cluster, specify --peer.adr and optionally, specify a new table name.

The following example creates a new table using HBase Shell in non-interactive mode, and then copies data in two ColumnFamilies in rows starting with timestamp 1265875194289 and including the last row before the CopyTable started, to the new table.

```
echo create 'NewTestTable', 'cf1', 'cf2', 'cf3' | bin/hbase shell --non-inte
ractive
```

```
    bin/hbase org.apache.hadoop.hbase.mapreduce.CopyTable --starttime=126587
5194289  --families=cf1,cf2,cf3 --new.name=NewTestTable TestTable
```

Snapshots are recommended instead of CopyTable for most situations.

Use BulkLoad

In many situations, writing HFiles programmatically with your data, and bulk-loading that data into HBase on the RegionServer, has advantages over other data ingest mechanisms.

HBase uses an internal file format called HFile to store its data on disk. BulkLoad operations bypass the write path completely, providing the following benefits:

- The data is available to HBase immediately but does cause additional load or latency on the cluster when it appears.
- BulkLoad operations do not use the write-ahead log (WAL) and do not cause flushes or split storms.
- BulkLoad operations do not cause excessive garbage collection.

If you use BulkLoads with HBase, your workflow is similar to the following:

1. Extract your data from its existing source. For instance, if your data is in a MySQL database, you might run the mysqldump command. The process you use depends on your data. If your data is already in TSV or CSV format, skip this step and use the included ImportTsv utility to process your data into HFiles. See the ImportTsv documentation for details.
2. Process your data into HFile format. See http://hbase.apache.org/book.html#_hfile_format_2 for details about HFile format. Usually you use a MapReduce job for the conversion, and you often need to write the Mapper yourself because your data is unique. The job must to emit the row key as the Key, and either a KeyValue, a Put, or a Delete as the Value. The Reducer is handled by HBase; configure it using HFileOutputFormat.configureIncrementalLoad() and it does the following:
   - Inspects the table to configure a total order partitioner
   - Uploads the partitions file to the cluster and adds it to the DistributedCache
   - Sets the number of reduce tasks to match the current number of regions
   - Sets the output key/value class to match HFileOutputFormat requirements
   - Sets the Reducer to perform the appropriate sorting (either KeyValueSortReducer or PutSortReducer)
3. One HFile is created per region in the output folder. Input data is almost completely re-written, so you need available disk space at least twice the size of the original data set. For example, for a 100 GB output from mysq ldump, you should have at least 200 GB of available disk space in HDFS. You can delete the original input file at the end of the process.
4. Load the files into HBase. Use the LoadIncrementalHFiles command (more commonly known as the completebulkload tool), passing it a URL that locates the files in HDFS. Each file is loaded into the relevant region on the RegionServer for the region. You can limit the number of versions that are loaded by passing the --versions= N option, where N is the maximum number of versions to include, from newest to oldest (largest timestamp to smallest timestamp).

   If a region was split after the files were created, the tool automatically splits the HFile according to the new boundaries. This process is inefficient, so if your table is being written to by other processes, you should load as soon as the transform step is done.

The following illustration shows the full BulkLoad process.

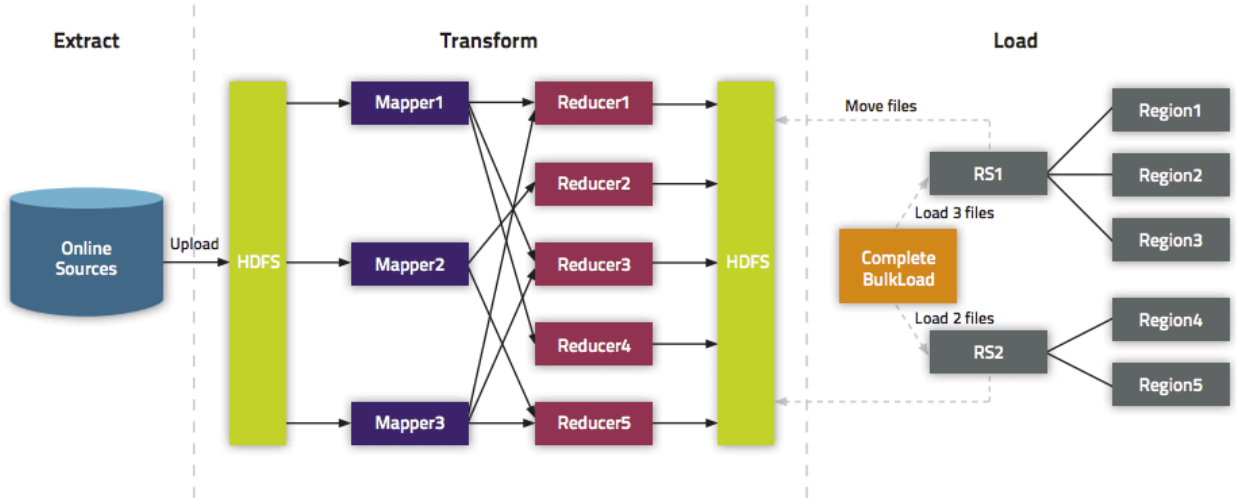

For an explanation of how secure BulkLoad works, see Bulk Loading .

Extra Steps for BulkLoad With Encryption Zones

When using BulkLoad to import data into HBase in the a cluster using encryption zones, the following information is important.

- Both the staging directory and the directory into which you place your generated HFiles need to be within HBase's encryption zone (generally under the /hbase directory). Before you can do this, you need to change the permissions of /hbase to be world-executable but not world-readable (rwx--x--x, or numeric mode 711).
- You also need to configure the HMaster to set the permissions of the HBase root directory correctly. If you use Cloudera Manager, edit the Master Advanced Configuration Snippet (Safety Valve) for hbase-site.xml. Otherwise, edit hbase-site.xml on the HMaster. Add the following:

  ```
  <property>
        <name>hbase.rootdir.perms</name>
        <value>711</value>
        </property>
  ```

  If you skip this step, a previously-working BulkLoad setup will start to fail with permission errors when you restart the HMaster.

Use cases for BulkLoad

There are three typical use cases when using BulkLoad can be advantageous.

- Loading your original dataset into HBase for the first time - Your initial dataset might be quite large, and bypassing the HBase write path can speed up the process considerably.
- Incremental Load - To load new data periodically, use BulkLoad to import it in batches at your preferred intervals. This alleviates latency problems and helps you to achieve service-level agreements (SLAs). However, one trigger for compaction is the number of HFiles on a RegionServer. Therefore, importing a large number of HFiles at frequent intervals can cause major compactions to happen more often than they otherwise would, negatively impacting performance. You can mitigate this by tuning the compaction settings such that the maximum number of HFiles that can be present without triggering a compaction is very high, and relying on other factors, such as the size of the Memstore, to trigger compactions.
- Data needs to originate elsewhere - If an existing system is capturing the data you want to have in HBase and needs to remain active for business reasons, you can periodically BulkLoad data from the system into HBase so that you can perform operations on it without impacting the system.

Use cluster replication

If your data is already in an HBase cluster, replication is useful for getting the data into additional HBase clusters.

In HBase, cluster replication refers to keeping one cluster state synchronized with that of another cluster, using the write-ahead log (WAL) of the source cluster to propagate the changes. Replication is enabled at column family granularity. Before enabling replication for a column family, create the table and all column families to be replicated, on the destination cluster.

Cluster replication uses an active-push methodology. An HBase cluster can be a source (also called active, meaning that it writes new data), a destination (also called passive, meaning that it receives data using replication), or can fulfill both roles at once. Replication is asynchronous, and the goal of replication is consistency.

When data is replicated from one cluster to another, the original source of the data is tracked with a cluster ID, which is part of the metadata. All clusters that have already consumed the data are also tracked. This prevents replication loops.

Related Information

Using HBase Replication

Use Sqoop

Sqoop can import records into a table in HBase. It has an out-of-the-box support for HBase

There are two mandatory options you must specify when using the sqoop import command to import data into HBase using Sqoop:

- --hbase-table: Specifies the name of the table in HBase to which you want to import your data.
- --column-family: Specifies into which column family Sqoop imports the data of your tables.

For example, you can import the table cities into an already existing HBase table with the same name and use the column family name world:

```
sqoop import --connect JDBC:MYSQL://MYSQL.EXAMPLE.COM/SQOOP --username SQOOP
 --password SQOOP --table CITIES --hbase-table CITIES --column-family WORLD
```

If the target table and column family do not exist, the Sqoop job will exit with an error. You must create the target table and column family before running an import. If you specify --hbase-create-table, Sqoop creates the target table and column family if they do not exist, using the default parameters from your HBase configuration.

Sqoop needs to identify which RDBMS column is used as row key column in the HBase table. There are three ways to do this:

- By default, with the column name specified in the --split-by option
- With the primary key of the table, if it is available
- With the --hbase-row-key parameter, which overrides both the --split-by option and the primary key of the table

For more information on data insertion into HBase, see Sqoop User Guide.

Import NULL Column Updates into HBase

You can specify how Sqoop handles RDBMS table column updated to NULL during incremental import.

There are two modes for this, ignore and delete. You can specify the mode using the --hbase-null-incrementel-mode option:

- -ignore: This is the default value. If the source table's column is updated to NULL, the target HBase table will still show the previous value for that column.
- -delete: If the source table's column is updated to NULL, all previous versions of the column will be deleted from HBase. When checking the column in HBase using the Java API, a null value will be displayed.

Examples:

Run an incremental import to an HBase table and ignore the columns which were updated to NULL in the relational database:

```
sqoop import --connect $CONN --username $USER --password $PASS --t
able "HBASE_TEST" --hbase-table HBASE_TEST --column-family DATA -m 1 --incr
emental lastmodified --check-column DATE_MODIFIED --last-value "2017-12-15
 10:58:44.0" --merge-key ID --hbase-null-incremental-mode ignore
```

Run an incremental import to an HBase table and delete all the versions of the columns which were updated to NULL in the relational database:

```
                sqoop import --connect $CONN --username $USER --passw
ord $PASS --table "HBASE_TEST" --hbase-table HBASE_TEST --column-family DATA
 -m 1 --incremental lastmodified --check-column DATE_MODIFIED --last-valu
e "2017-12-15 10:58:44.0" --merge-key ID --hbase-null-incremental-mode d
elete
```

Use Spark

You can write data to HBase from Apache Spark using def saveAsHadoopDataset(conf: JobConf): Unit.

This example is adapted from a post on the spark-users mailing list.

```
// Note: mapred package is used, instead of the
                    // mapreduce package which contains new hadoop APIs.

                    import org.apache.hadoop.hbase.mapred.TableOutputFormat
                    import org.apache.hadoop.hbase.client
                    // ... some other settings

                    val conf = HBaseConfiguration.create()

                    // general hbase settings
                    conf.set("hbase.rootdir",
                    "hdfs://" + nameNodeURL + ":" + hdfsPort + "/hbase")
                    conf.setBoolean("hbase.cluster.distributed", true)
                    conf.set("hbase.zookeeper.quorum", hostname)
                    conf.setInt("hbase.client.scanner.caching", 10000)
                    // ... some other settings

                    val jobConfig: JobConf = new JobConf(conf, this.getClass
)

                    // Note:  TableOutputFormat is used as deprecated code
                    // because JobConf is an old hadoop API
                    jobConfig.setOutputFormat(classOf[TableOutputFormat])
                    jobConfig.set(TableOutputFormat.OUTPUT_TABLE, outputTabl
e)
```

Next, provide the mapping between how the data looks in Spark and how it should look in HBase. The following example assumes that your HBase table has two column families, col_1 and col_2, and that your data is formatted in sets of three in Spark, like (row_key, col_1, col_2).

```
def convert(triple: (Int, Int, Int)) = {
                val p = new Put(Bytes.toBytes(triple._1))
                p.add(Bytes.toBytes("cf"),
                Bytes.toBytes("col_1"),
```

```
                    Bytes.toBytes(triple._2))
                    p.add(Bytes.toBytes("cf"),
                    Bytes.toBytes("col_2"),
                    Bytes.toBytes(triple._3))
                    (new ImmutableBytesWritable, p)
                    }
```

To write the data from Spark to HBase, you might use:

```
new PairRDDFunctions(localData.map(convert)).saveAsHadoopDataset(jobConfig)
```

Use a custom MapReduce job

Many of the methods to import data into HBase use MapReduce implicitly. If none of those approaches fit your needs, you can use MapReduce directly to convert data to a series of HFiles or API calls for import into HBase

In this way, you can import data from Avro, Parquet, or another format into HBase, or export data from HBase into another format, using API calls such as TableOutputFormat, HFileOutputFormat, and TableInputFormat.

Use HashTable and SyncTable Tool

HashTable/SyncTable is a two steps tool for synchronizing table data without copying all cells in a specified row key/time period range.

The HashTable/SyncTable tool can be used for partial or entire table data synchronization, under the same or remote cluster. Both the HashTable and the SyncTable step are implemented as a MapReduce job.

The first step, HashTable, creates hashed indexes for batch of cells on the source table and output those as results. The source table is the table whose state is copied to its counterpart.

The second step, SyncTable, scans the target table and calculates hash indexes for table cells. Then these hashes are compared to the HashTable step outputs. So, SyncTable scans and compares cells for diverging hashes and updating only the mismatching cells.

This results in less network traffic or data transfers than other methods, for example CopyTable, which can impact performance when large tables are synchronized on remote clusters.

Remote clusters are often deployed on different Kerberos Realms. SyncTable support cross realm authentication, allowing a SyncTable process running on the target cluster to connect to the source cluster and read both the HashTable output files and the given HBase table when performing the required comparisons.

HashTable/SyncTable tool configuration

You can configure the HashTable/SyncTable tool for your specific needs.

Using the batchsize option

You can define the amount of cell data for a given region that is hashed together in a single hash value using the batchsize option, which sets the batchsize property. Sizing this property has a direct impact on the synchronization efficiency. If the batch size is increased, larger chunks are hashed.

If only a few differences are expected between the two tables, using a bit larger batch size can be beneficial, as less scans are executed by mapper tasks of SyncTable.

However, if relatively frequent differences are expected between the tables, using a large batch size can cause frequent mismatches of hash values, as the probability of finding at least one mismatch in a batch is increased.

The following is an example of sizing this property:

```
$ hbase org.apache.hadoop.hbase.mapreduce.HashTable --batchsize=32000 --numh
ashfiles=50 --starttime=1265875194289 --endtime=1265878794289 --families=cf2
,cf3 TESTTABLEA /HASHES/TESTTABLE
```

Creating a read-only report

You can use the dryrun option in the second, SyncTable, step to create a read only report. It produces only COUNTERS indicating the differences between the two tables, but does not perform any actual changes. It can be used as an alternative of the VerifyReplication tool.

The following is an example of using this option:

```
$ hbase org.apache.hadoop.hbase.mapreduce.SyncTable --dryrun=true --sourcezk
cluster=ZK1.EXAMPLE.COM,ZK2.EXAMPLE.COM,ZK3.EXAMPLE.COM:2181:/HBASE HDFS://
NN:8020/HASHES/TESTTABLE TESTTABLEA TESTTABLEB
```

Handling missing cells

By default, SyncTable changes the target table to make it an exact copy of the source table, at least for the specified startrow-stoprow or starttime-endtime.

Setting doDeletes to false modifies the default behaviour to not delete target cells that are missing on the source table. Similarly, setting doPuts to false modifies the default behaviour to not add missing cells to target table. If you set both doDeletes and doPuts to false, the result will be the same as setting dryrun to true.

In the case of two-way replication or other scenarios where both source and target clusters can have data ingested, Cloudera recommends to set doDeletes to false. Otherwise any additional cells inserted on the SyncTable target cluster and not yet replicated to the source cluster would be deleted, and potentially lost permanently.

Synchronize table data using HashTable/SyncTable tool

The HashTable/SyncTable tool can be used for partial or entire table data synchronization, under the same or remote cluster.

Before you begin

- Ensure that all RegionServers/DataNodes on the source cluster is accessible by the NodeManagers on the target cluster where SyncTable job tasks will be running.
- In the case of secured clusters, the user on the target cluster who executes the SyncTable job must be able to do the following on the HDFS and HBase services of the source cluster:
  - Authenticate: for example, using centralized authentication or cross-realm setup.
  - Be authorized: having at least read permission.

Procedure

1. Run HashTable on the source table cluster: HashTable [options] <tablename> <outputpath>

   The following is an example to hash the TesTable in 32kB batches for a 1 hour window into 50 files:

   ```
   $ hbase org.apache.hadoop.hbase.mapreduce.HashTable --batchsize=32000 --n
   umhashfiles=50 --starttime=1265875194289 --endtime=1265878794289 --famil
   ies=CF2,CF3 TESTTABLEA /HASHES/TESTTABLE
   ```

2. Run SyncTable on the target cluster: SyncTable [options] <sourcehashdir> <sourcetable> <targettable>

   The following is an example for a dry run SyncTable of tableA from a remote source cluster to a local tableB on the target cluster:

```
$ hbase org.apache.hadoop.hbase.mapreduce.SyncTable --dryrun=true --sour
cezkcluster=ZK1.EXAMPLE.COM,ZK2.EXAMPLE.COM,ZK3.EXAMPLE.COM:2181:/HBASE
 HDFS://NN:8020/HASHES/TESTTABLE TESTTABLEA TESTTABLEB
```

   Note:

   For more detailed information regarding HashTable/SynTable tool options, use hbase org.apache.hadoop.hbase.mapreduce.SyncTable --help.

Writing data to HBase

To write data to HBase, you use methods of the Table class.

You can use the Java API directly, or use the HBase Shell, the REST API, the Thrift API, , or another client which uses the Java API indirectly. When you issue a Put, the coordinates of the data are the row, the column, and the timestamp. The timestamp is unique per version of the cell, and can be generated automatically or specified programmatically by your application, and must be a long integer.

Variations on Put

There are several different ways to write data into HBase.

- A Put operation writes data into HBase.
- A Delete operation deletes data from HBase. What actually happens during a Delete depends upon several factors.
- A CheckAndPut operation performs a Scan before attempting the Put, and only does the Put if a value matches what is expected, and provides row-level atomicity.
- A CheckAndDelete operation performs a Scan before attempting the Delete, and only does the Delete if a value matches what is expected.
- An Increment operation increments values of one or more columns within a single row, and provides row-level atomicity.

Refer to the API documentation for a full list of methods provided for writing data to HBase.Different methods require different access levels and have other differences.

Versions

When you put data into HBase, a timestamp is required.

The timestamp can be generated automatically by the RegionServer or can be supplied by you. The timestamp must be unique per version of a given cell, because the timestamp identifies the version. To modify a previous version of a cell, for instance, you would issue a Put with a different value for the data itself, but the same timestamp.

HBase's behavior regarding versions is highly configurable. The maximum number of versions defaults to 1. You can change the default value for HBase by configuring hbase.column.max.version in hbase-site.xml, either using an advanced configuration snippet if you use Cloudera Manager, or by editing the file directly otherwise.

You can also configure the maximum and minimum number of versions to keep for a given column, or specify a default time-to-live (TTL), which is the number of seconds before a version is deleted. The following examples all use alter statements in HBase Shell to create new column families with the given characteristics, but you can use the

same syntax when creating a new table or to alter an existing column family. This is only a fraction of the options you can specify for a given column family.

```
hbase> alter ‘t1#, NAME => ‘f1#, VERSIONS => 5
    hbase> alter ‘t1#, NAME => ‘f1#, MIN_VERSIONS => 2
    hbase> alter ‘t1#, NAME => ‘f1#, TTL => 15
```

HBase sorts the versions of a cell from newest to oldest, by sorting the timestamps lexicographically. When a version needs to be deleted because a threshold has been reached, HBase always chooses the "oldest" version, even if it is in fact the most recent version to be inserted. Keep this in mind when designing your timestamps. Consider using the default generated timestamps and storing other version-specific data elsewhere in the row, such as in the row key. If MIN_VERSIONS and TTL conflict, MIN_VERSIONS takes precedence.

Deletion

When you request for HBase to delete data, either explicitly using a Delete method or implicitly using a threshold such as the maximum number of versions or the TTL, HBase does not delete the data immediately

Instead, it writes a deletion marker, called a tombstone, to the HFile, which is the physical file where a given RegionServer stores its region of a column family. The tombstone markers are processed during major compaction operations, when HFiles are rewritten without the deleted data included.

Even after major compactions, "deleted" data may not actually be deleted. You can specify the KEEP_DELETED_CELLS option for a given column family, and the tombstones will be preserved in the HFile even after major compaction. One scenario where this approach might be useful is for data retention policies.

Another reason deleted data may not actually be deleted is if the data would be required to restore a table from a snapshot which has not been deleted. In this case, the data is moved to an archive during a major compaction, and only deleted when the snapshot is deleted. This is a good reason to monitor the number of snapshots saved in HBase.

Examples

These abbreviated example writes data to an HBase table.

This abbreviated example writes data to an HBase table using HBase Shell and then scans the table to show the result

```
hbase> put 'test', 'row1', 'cf:a', 'value1'
0 row(s) in 0.1770 seconds

hbase> put 'test', 'row2', 'cf:b', 'value2'
0 row(s) in 0.0160 seconds

hbase> put 'test', 'row3', 'cf:c', 'value3'
0 row(s) in 0.0260 seconds
hbase> scan 'test'
ROW                   COLUMN+CELL
 row1                 column=cf:a, timestamp=1403759475114, value=value1
 row2                 column=cf:b, timestamp=1403759492807, value=value2
 row3                 column=cf:c, timestamp=1403759503155, value=value3
3 row(s) in 0.0440 seconds
```

This abbreviated example uses the HBase API to write data to an HBase table, using the automatic timestamp created by the Region Server.

```
publicstaticfinalbyte[] CF = "cf".getBytes();
publicstaticfinalbyte[] ATTR = "attr".getBytes();
...
Put put = new Put(Bytes.toBytes(row));
put.add(CF, ATTR, Bytes.toBytes( data));
```

```
htable.put(put);
```

This example uses the HBase API to write data to an HBase table, specifying the timestamp.

```
publicstaticfinalbyte[] CF = "cf".getBytes();
publicstaticfinalbyte[] ATTR = "attr".getBytes();
...
Put put = new Put( Bytes.toBytes(row));
long explicitTimeInMs = 555;  // just an example
put.add(CF, ATTR, explicitTimeInMs, Bytes.toBytes(data));
htable.put(put);
```

Reading data from HBase

The Get and Scan are the two ways to read data from HBase, aside from manually parsing HFiles.

A Get is simply a Scan limited by the API to one row. A Scan fetches zero or more rows of a table. By default, a Scan reads the entire table from start to end. You can limit your Scan results in several different ways, which affect the Scan's load in terms of IO, network, or both, as well as processing load on the client side. This topic is provided as a quick reference. Refer to the API documentation for Scan for more in-depth information. You can also perform Get and Scan using the HBase Shell, the REST API, or the Thrift API.

- Specify a startrow or stoprow or both. Neither startrow nor stoprow need to exist. Because HBase sorts rows lexicographically, it will return the first row after startrow would have occurred, and will stop returning rows after stoprow would have occurred.The goal is to reduce IO and network.
  - The startrow is inclusive and the stoprow is exclusive. Given a table with rows a, b, c, d, e, f, and startrow of c and stoprow of f, rows c-e are returned.
  - If you omit startrow, the first row of the table is the startrow.
  - If you omit the stoprow, all results after startrow (including startrow) are returned.
  - If startrow is lexicographically after stoprow, and you set Scan setReversed(boolean reversed) to true, the results are returned in reverse order. Given the same table above, with rows a-f, if you specify c as the stoprow and f as the startrow, rows f, e, and d are returned.

  ```
  Scan()
       Scan(byte[] startRow)
       Scan(byte[] startRow, byte[] stopRow)
  ```

- Specify a scanner cache that will be filled before the Scan result is returned, setting setCaching to the number of rows to cache before returning the result. By default, the caching setting on the table is used. The goal is to balance IO and network load.

  ```
  public Scan setCaching(int caching)
  ```

- To limit the number of columns if your table has very wide rows (rows with a large number of columns), use setBatch(int batch) and set it to the number of columns you want to return in one batch. A large number of columns is not a recommended design pattern.

  ```
  public Scan setBatch(int batch)
  ```

- To specify a maximum result size, use setMaxResultSize(long), with the number of bytes. The goal is to reduce IO and network.

  ```
  public Scan setMaxResultSize(long maxResultSize)
  ```

- When you use setCaching and setMaxResultSize together, single server requests are limited by either number of rows or maximum result size, whichever limit comes first.

- You can limit the scan to specific column families or columns by using addFamily or addColumn. The goal is to reduce IO and network. IO is reduced because each column family is represented by a Store on each RegionServer, and only the Stores representing the specific column families in question need to be accessed.

```
public Scan addColumn(byte[] family,
     byte[] qualifier)

     public Scan addFamily(byte[] family)
```

- You can specify a range of timestamps or a single timestamp by specifying setTimeRange or setTimestamp.

```
public Scan setTimeRange(long minStamp,
     long maxStamp)
     throws IOException

     public Scan setTimeStamp(long timestamp)
     throws IOException
```

- You can retrieve a maximum number of versions by using setMaxVersions.

```
public Scan setMaxVersions(int maxVersions)
```

- You can use a filter by using setFilter. .

```
public Scan setFilter(Filter filter)
```

- You can disable the server-side block cache for a specific scan using the API setCacheBlocks(boolean). This is an expert setting and should only be used if you know what you are doing.

Related Information

HBase filtering

Perform scans using HBase Shell

You can perform scans using HBase Shell, for testing or quick queries.

Use the following guidelines or issue the scan command in HBase Shell with no parameters for more usage information. This represents only a subset of possibilities.

```
# Display usage information
hbase> scan
# Scan all rows of table 't1'
hbase> scan 't1'
# Specify a startrow, limit the result to 10 rows, and only return selected
 columns
hbase> scan 't1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'}

# Specify a timerange
hbase> scan 't1', {TIMERANGE => [1303668804, 1303668904]}

# Specify a custom filter
hbase> scan 't1', {FILTER => org.apache.hadoop.hbase.filter.ColumnPaginatio
nFilter.new(1, 0)}

# Specify a row prefix filter and another custom filter
hbase> scan 't1', {ROWPREFIXFILTER => 'row2',
                          FILTER => (QualifierFilter (>=, 'binary:xyz')) AND
 (TimestampsFilter ( 123, 456))}

# Disable the block cache for a specific scan (experts only)
hbase> scan 't1', {COLUMNS => ['c1', 'c2'], CACHE_BLOCKS => false}
```

HBase filtering

When reading data from HBase using Get or Scan operations, you can use custom filters to return a subset of results to the client.

While this does not reduce server-side IO, it does reduce network bandwidth and reduces the amount of data the client needs to process. Filters are generally used using the Java API, but can be used from HBase Shell for testing and debugging purposes.

HBase filters take zero or more arguments, in parentheses. Where the argument is a string, it is surrounded by single quotes ('string').

Related Information

Reading data from HBase

Dynamically loading a custom filter

Cloudera Runtime by default has the ability to dynamically load a custom filter.

It adds a JAR with your filter to the directory specified by the hbase.dynamic.jars.dir property (which defaults to the lib/ directory under the HBase root directory).

Go to Cloudera Manager and configure hbase.use.dynamic.jars property accordingly in the advanced configuration snippet for hbase-site.xml or to hbase-site.xml otherwise.

- To disable automatic loading of dynamic JARs, set hbase.use.dynamic.jars to false.
- To use any dynamic JARs, for example, custom filters or UDFs, set hbase.use.dynamic.jars to true.

Related Information

UDF concepts

Logical operators, comparison operators and comparators

Filters can be combined together with logical operators.

Some filters take a combination of comparison operators and comparators. Following is the list of each.

Logical Operators

- AND - the key-value must pass both the filters to be included in the results.
- OR - the key-value must pass at least one of the filters to be included in the results.
- SKIP - for a particular row, if any of the key-values do not pass the filter condition, the entire row is skipped.
- WHILE - For a particular row, it continues to emit key-values until a key-value is reached that fails the filter condition.
- Compound Filters - Using these operators, a hierarchy of filters can be created. For example:

```
(Filter1 AND Filter2)OR(Filter3 AND Filter4)
```

Comparison Operators

- LESS (<)
- LESS_OR_EQUAL (<=)
- EQUAL (=)
- NOT_EQUAL (!=)
- GREATER_OR_EQUAL (>=)

- GREATER (>)
- NO_OP (no operation)

Comparators

- BinaryComparator - lexicographically compares against the specified byte array using the Bytes.compareTo(byte [], byte[]) method.
- BinaryPrefixComparator - lexicographically compares against a specified byte array. It only compares up to the length of this byte array.
- RegexStringComparator - compares against the specified byte array using the given regular expression. Only EQUAL and NOT_EQUAL comparisons are valid with this comparator.
- SubStringComparator - tests whether or not the given substring appears in a specified byte array. The comparison is case insensitive. Only EQUAL and NOT_EQUAL comparisons are valid with this comparator.

Examples

```
Example1: >, 'binary:abc' will match everything that is lexicographically gr
eater than "abc"
   Example2: =, 'binaryprefix:abc' will match everything whose first 3 char
acters are lexicographically equal to "abc"
   Example3: !=, 'regexstring:ab*yz' will match everything that doesn't begi
n with "ab" and ends with "yz"
   Example4: =, 'substring:abc123' will match everything that begins with
 the substring "abc123"
```

Compound operators

Within an expression, parentheses can be used to group clauses together, and parentheses have the highest order of precedence.

SKIP and WHILE operators are next, and have the same precedence.

The AND operator is next.

The OR operator is next.

Examples

```
A filter string of the form: “Filter1 AND Filter2 OR Filter3” will be evalua
ted as: “(Filter1 AND Filter2) OR Filter3”
   A filter string of the form: “Filter1 AND SKIP Filter2 OR Filter3” will
 be evaluated as: “(Filter1 AND (SKIP Filter2)) OR Filter3”
```

Filter types

HBase includes several filter types, as well as the ability to group filters together and create your own custom filters.

- KeyOnlyFilter - takes no arguments. Returns the key portion of each key-value pair.

  ```
  Syntax: KeyOnlyFilter ()
  ```

- FirstKeyOnlyFilter - takes no arguments. Returns the key portion of the first key-value pair.

  ```
  Syntax: FirstKeyOnlyFilter ()
  ```

- PrefixFilter - takes a single argument, a prefix of a row key. It returns only those key-values present in a row that start with the specified row prefix

```
Syntax:  PrefixFilter (‘<row_prefix>’)

      Example: PrefixFilter (‘Row’)
```

- ColumnPrefixFilter - takes a single argument, a column prefix. It returns only those key-values present in a column that starts with the specified column prefix.

```
Syntax:  ColumnPrefixFilter (‘<column_prefix>’)

      Example: ColumnPrefixFilter (‘Col’)
```

- MultipleColumnPrefixFilter - takes a list of column prefixes. It returns key-values that are present in a column that starts with any of the specified column prefixes.

```
Syntax:  MultipleColumnPrefixFilter (‘<column_prefix>’, ‘<column_prefix>’,
 …, ‘<column_prefix>’)

      Example: MultipleColumnPrefixFilter (‘Col1’, ‘Col2’)
```

- ColumnCountGetFilter - takes one argument, a limit. It returns the first limit number of columns in the table.

```
Syntax:  ColumnCountGetFilter (‘<limit>’)

      Example: ColumnCountGetFilter (4)
```

- PageFilter - takes one argument, a page size. It returns page size number of rows from the table.

```
Syntax:  PageFilter (‘<page_size>’)

      Example: PageFilter (2)
```

- ColumnPaginationFilter - takes two arguments, a limit and offset. It returns limit number of columns after offset number of columns. It does this for all the rows.

```
Syntax:  ColumnPaginationFilter (‘<limit>’, ‘<offset>’)

      Example: ColumnPaginationFilter (3, 5)
```

- InclusiveStopFilter - takes one argument, a row key on which to stop scanning. It returns all key-values present in rows up to and including the specified row.

```
Syntax:  InclusiveStopFilter (‘<stop_row_key>’)

      Example: InclusiveStopFilter (‘Row2’)
```

- TimeStampsFilter - takes a list of timestamps. It returns those key-values whose timestamps matches any of the specified timestamps.

```
Syntax:  TimeStampsFilter (<timestamp>, <timestamp>, ... ,<timestamp>)

      Example: TimeStampsFilter (5985489, 48895495, 58489845945)
```

- RowFilter - takes a compare operator and a comparator. It compares each row key with the comparator using the compare operator and if the comparison returns true, it returns all the key-values in that row.

```
Syntax:  RowFilter (<compareOp>, ‘<row_comparator>’)

      Example: RowFilter (<=, ‘binary:xyz)
```

- FamilyFilter - takes a compare operator and a comparator. It compares each family name with the comparator using the compare operator and if the comparison returns true, it returns all the key-values in that family.

```
Syntax:  FamilyFilter (<compareOp>, ‘<family_comparator>’)

      Example: FamilyFilter (>=, ‘binaryprefix:FamilyB’)
```

- QualifierFilter - takes a compare operator and a comparator. It compares each qualifier name with the comparator using the compare operator and if the comparison returns true, it returns all the key-values in that column.

```
Syntax:  QualifierFilter (<compareOp>, ‘<qualifier_comparator>’)

      Example: QualifierFilter (=, ‘substring:Column1’)
```

- ValueFilter - takes a compare operator and a comparator. It compares each value with the comparator using the compare operator and if the comparison returns true, it returns that key-value.

```
Syntax:  ValueFilter (<compareOp>, ‘<value_comparator>’)

      Example: ValueFilter (!=, ‘binary:Value’)
```

- DependentColumnFilter - takes two arguments required arguments, a family and a qualifier. It tries to locate this column in each row and returns all key-values in that row that have the same timestamp. If the row does not contain the specified column, none of the key-values in that row will be returned.

  The filter can also take an optional boolean argument, dropDependentColumn. If set to true, the column used for the filter does not get returned.

  The filter can also take two more additional optional arguments, a compare operator and a value comparator, which are further checks in addition to the family and qualifier. If the dependent column is found, its value should also pass the value check. If it does pass the value check, only then is its timestamp taken into consideration.

```
Syntax:  DependentColumnFilter (‘<family>’, ‘<qualifier>’, <boolean>, <c
ompare operator>, ‘<value comparator’)
        DependentColumnFilter (‘<family>’, ‘<qualifier>’, <boolean>)
        DependentColumnFilter (‘<family>’, ‘<qualifier>’)

        Example: DependentColumnFilter (‘conf’, ‘blacklist’, false, >=, ‘
zebra’)
        DependentColumnFilter (‘conf’, ‘blacklist’, true)
        DependentColumnFilter (‘conf’, ‘blacklist’)
```

- SingleColumnValueFilter - takes a column family, a qualifier, a compare operator and a comparator. If the specified column is not found, all the columns of that row will be emitted. If the column is found and the comparison with the comparator returns true, all the columns of the row will be emitted. If the condition fails, the row will not be emitted.

  This filter also takes two additional optional boolean arguments, filterIfColumnMissing and setLatestVersion Only.

  If the filterIfColumnMissing flag is set to true, the columns of the row will not be emitted if the specified column to check is not found in the row. The default value is false.

  If the setLatestVersionOnly flag is set to false, it will test previous versions (timestamps) in addition to the most recent. The default value is true.

  These flags are optional and dependent on each other. You must set neither or both of them together.

```
Syntax:  SingleColumnValueFilter (‘<family>’, ‘<qualifier>’, <compare op
erator>, ‘<comparator>’, <filterIfColumnMissing_boolean>, <latest_versio
n_boolean>)
          Syntax:  SingleColumnValueFilter (‘<family>’, ‘<qualifier>’, <
compare operator>, ‘<comparator>’)
```

```
          Example: SingleColumnValueFilter (‘FamilyA’, ‘Column1’, <=, ‘abc
’, true, false)
          Example: SingleColumnValueFilter ('FamilyA’, ‘Column1’, <=, ‘a
bc’)
```

- SingleColumnValueExcludeFilter - takes the same arguments and behaves same as SingleColumnValueFilter. However, if the column is found and the condition passes, all the columns of the row will be emitted except for the tested column value.

```
Syntax:  SingleColumnValueExcludeFilter (<family>, <qualifier>, <compare
 operators>, <comparator>, <latest_version_boolean>, <filterIfColumnMiss
ing_boolean>)
      Syntax:  SingleColumnValueExcludeFilter (<family>, <qualifier>, <com
pare operator> <comparator>)

      Example: SingleColumnValueExcludeFilter (‘FamilyA’, ‘Column1’, ‘<=’,
 ‘abc’, ‘false’, ‘true’)
      Example: SingleColumnValueExcludeFilter (‘FamilyA’, ‘Column1’, ‘<=’,
 ‘abc’)
```

- ColumnRangeFilter - takes either minColumn, maxColumn, or both. Returns only those keys with columns that are between minColumn and maxColumn. It also takes two boolean variables to indicate whether to include the minColumn and maxColumn or not. If you don’t want to set the minColumn or the maxColumn, you can pass in an empty argument.

```
Syntax:  ColumnRangeFilter (‘<minColumn >’, <minColumnInclusive_bool>, ‘
<maxColumn>’, <maxColumnInclusive_bool>)

      Example: ColumnRangeFilter (‘abc’, true, ‘xyz’, false)
```

- Custom Filter - You can create a custom filter by implementing the Filter class. The JAR must be available on all RegionServers.

HBase Shell example

This example scans the 'users' table for rows where the contents of the cf:name column equals the string 'abc'.

```
hbase>
          scan 'users', { FILTER =>
 SingleColumnValueFilter.new(Bytes.toBytes('cf'),
          Bytes.toBytes('name'), CompareFilter::CompareOp.valueOf('EQUAL'),
          BinaryComparator.new(Bytes.toBytes('abc')))}
```

Java API example

This example shows how to use the Java API to implement several different filters.

This example is taken from the HBase unit test found in hbase-server/src/test/java/org/apache/hadoop/hbase/filter/TestSingleColumnValueFilter.java.

```
/**
      *
      * Licensed to the Apache Software Foundation (ASF) under one
      * or more contributor license agreements.  See the NOTICE file
      * distributed with this work for additional information
      * regarding copyright ownership.  The ASF licenses this file
```

```
      * to you under the Apache License, Version 2.0 (the
      * "License"); you may not use this file except in compliance
      * with the License.  You may obtain a copy of the License at
      *
      *     http://www.apache.org/licenses/LICENSE-2.0
      *
      * Unless required by applicable law or agreed to in writing, software
      * distributed under the License is distributed on an "AS IS" BASIS,
      * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or im
plied.
      * See the License for the specific language governing permissions and
      * limitations under the License.
      */
      package org.apache.hadoop.hbase.filter;

      import static org.junit.Assert.assertFalse;
      import static org.junit.Assert.assertTrue;

      import java.util.regex.Pattern;

      import org.apache.hadoop.hbase.KeyValue;
      import org.apache.hadoop.hbase.SmallTests;
      import org.apache.hadoop.hbase.filter.CompareFilter.CompareOp;
      import org.apache.hadoop.hbase.util.Bytes;
      import org.junit.Before;
      import org.junit.Test;
      import org.junit.experimental.categories.Category;

      /**
      * Tests the value filter
      */
      @Category(SmallTests.class)
      public class TestSingleColumnValueFilter {
      private static final byte[] ROW = Bytes.toBytes("test");
      private static final byte[] COLUMN_FAMILY = Bytes.toBytes("test");
      private static final byte [] COLUMN_QUALIFIER = Bytes.toBytes("foo");
      private static final byte[] VAL_1 = Bytes.toBytes("a");
      private static final byte[] VAL_2 = Bytes.toBytes("ab");
      private static final byte[] VAL_3 = Bytes.toBytes("abc");
      private static final byte[] VAL_4 = Bytes.toBytes("abcd");
      private static final byte[] FULLSTRING_1 =
      Bytes.toBytes("The quick brown fox jumps over the lazy dog.");
      private static final byte[] FULLSTRING_2 =
      Bytes.toBytes("The slow grey fox trips over the lazy dog.");
      private static final String QUICK_SUBSTR = "quick";
      private static final String QUICK_REGEX = ".+quick.+";
      private static final Pattern QUICK_PATTERN = Pattern.compile("QuIcK",
 Pattern.CASE_INSENSITIVE | Pattern.DOTALL);

      Filter basicFilter;
      Filter nullFilter;
      Filter substrFilter;
      Filter regexFilter;
      Filter regexPatternFilter;

      @Before
      public void setUp() throws Exception {
      basicFilter = basicFilterNew();
      nullFilter = nullFilterNew();
      substrFilter = substrFilterNew();
      regexFilter = regexFilterNew();
      regexPatternFilter = regexFilterNew(QUICK_PATTERN);
      }
```

```
      private Filter basicFilterNew() {
      return new SingleColumnValueFilter(COLUMN_FAMILY, COLUMN_QUALIFIER,
      CompareOp.GREATER_OR_EQUAL, VAL_2);
      }

      private Filter nullFilterNew() {
      return new SingleColumnValueFilter(COLUMN_FAMILY, COLUMN_QUALIFIER,
CompareOp.NOT_EQUAL,
      new NullComparator());
      }

      private Filter substrFilterNew() {
      return new SingleColumnValueFilter(COLUMN_FAMILY, COLUMN_QUALIFIER,
      CompareOp.EQUAL,
      new SubstringComparator(QUICK_SUBSTR));
      }

      private Filter regexFilterNew() {
      return new SingleColumnValueFilter(COLUMN_FAMILY, COLUMN_QUALIFIER,
      CompareOp.EQUAL,
      new RegexStringComparator(QUICK_REGEX));
      }

      private Filter regexFilterNew(Pattern pattern) {
      return new SingleColumnValueFilter(COLUMN_FAMILY, COLUMN_QUALIFIER,
      CompareOp.EQUAL,
      new RegexStringComparator(pattern.pattern(), pattern.flags()));
      }

      private void basicFilterTests(SingleColumnValueFilter filter)
      throws Exception {
      KeyValue kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_
2);
      assertTrue("basicFilter1", filter.filterKeyValue(kv) == Filter.ReturnC
ode.INCLUDE);
      kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_3);
      assertTrue("basicFilter2", filter.filterKeyValue(kv) == Filter.ReturnC
ode.INCLUDE);
      kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_4);
      assertTrue("basicFilter3", filter.filterKeyValue(kv) == Filter.ReturnC
ode.INCLUDE);
      assertFalse("basicFilterNotNull", filter.filterRow());
      filter.reset();
      kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_1);
      assertTrue("basicFilter4", filter.filterKeyValue(kv) == Filter.Retu
rnCode.NEXT_ROW);
      kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_2);
      assertTrue("basicFilter4", filter.filterKeyValue(kv) == Filter.ReturnC
ode.NEXT_ROW);
      assertFalse("basicFilterAllRemaining", filter.filterAllRemaining());
      assertTrue("basicFilterNotNull", filter.filterRow());
      filter.reset();
      filter.setLatestVersionOnly(false);
      kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_1);
      assertTrue("basicFilter5", filter.filterKeyValue(kv) == Filter.ReturnC
ode.INCLUDE);
      kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_2);
      assertTrue("basicFilter5", filter.filterKeyValue(kv) == Filter.ReturnC
ode.INCLUDE);
      assertFalse("basicFilterNotNull", filter.filterRow());
      }

      private void nullFilterTests(Filter filter) throws Exception {
      ((SingleColumnValueFilter) filter).setFilterIfMissing(true);
```

```
      KeyValue kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, FUL
LSTRING_1);
      assertTrue("null1", filter.filterKeyValue(kv) == Filter.ReturnCode.INC
LUDE);
      assertFalse("null1FilterRow", filter.filterRow());
      filter.reset();
      kv = new KeyValue(ROW, COLUMN_FAMILY, Bytes.toBytes("qual2"), FULLST
RING_2);
      assertTrue("null2", filter.filterKeyValue(kv) == Filter.ReturnCode.IN
CLUDE);
      assertTrue("null2FilterRow", filter.filterRow());
      }

      private void substrFilterTests(Filter filter)
      throws Exception {
      KeyValue kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER,
      FULLSTRING_1);
      assertTrue("substrTrue",
      filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);
      kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER,
      FULLSTRING_2);
      assertTrue("substrFalse", filter.filterKeyValue(kv) == Filter.ReturnC
ode.INCLUDE);
      assertFalse("substrFilterAllRemaining", filter.filterAllRemaining());
      assertFalse("substrFilterNotNull", filter.filterRow());
      }

      private void regexFilterTests(Filter filter)
      throws Exception {
      KeyValue kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER,
      FULLSTRING_1);
      assertTrue("regexTrue",
      filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);
      kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER,
      FULLSTRING_2);
      assertTrue("regexFalse", filter.filterKeyValue(kv) == Filter.Return
Code.INCLUDE);
      assertFalse("regexFilterAllRemaining", filter.filterAllRemaining());
      assertFalse("regexFilterNotNull", filter.filterRow());
      }

      private void regexPatternFilterTests(Filter filter)
      throws Exception {
      KeyValue kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER,
      FULLSTRING_1);
      assertTrue("regexTrue",
      filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);
      assertFalse("regexFilterAllRemaining", filter.filterAllRemaining());
      assertFalse("regexFilterNotNull", filter.filterRow());
      }

      private Filter serializationTest(Filter filter)
      throws Exception {
      // Decompose filter to bytes.
      byte[] buffer = filter.toByteArray();

      // Recompose filter.
      Filter newFilter = SingleColumnValueFilter.parseFrom(buffer);
      return newFilter;
      }

      /**
      * Tests identification of the stop row
      * @throws Exception
```

```
    */
    @Test
    public void testStop() throws Exception {
    basicFilterTests((SingleColumnValueFilter) basicFilter);
    nullFilterTests(nullFilter);
    substrFilterTests(substrFilter);
    regexFilterTests(regexFilter);
    regexPatternFilterTests(regexPatternFilter);
    }

    /**
    * Tests serialization
    * @throws Exception
    */
    @Test
    public void testSerialization() throws Exception {
    Filter newFilter = serializationTest(basicFilter);
    basicFilterTests((SingleColumnValueFilter)newFilter);
    newFilter = serializationTest(nullFilter);
    nullFilterTests(newFilter);
    newFilter = serializationTest(substrFilter);
    substrFilterTests(newFilter);
    newFilter = serializationTest(regexFilter);
    regexFilterTests(newFilter);
    newFilter = serializationTest(regexPatternFilter);
    regexPatternFilterTests(newFilter);
    }

    }
```

HBase online merge

Cloudera Runtime supports online merging of regions.

HBase splits big regions automatically but does not support merging small regions automatically. To complete an online merge of two regions of a table, use the HBase shell to issue the online merge command. By default, both regions to be merged should be neighbors; that is, one end key of a region should be the start key of the other region. Although you can "force merge" any two regions of the same table, this can create overlaps and is not recommended.

The Master and RegionServer both participate in online merges. When the request to merge is sent to the Master, the Master moves the regions to be merged to the same RegionServer, usually the one where the region with the higher load resides. The Master then requests the RegionServer to merge the two regions. The RegionServer processes this request locally. Once the two regions are merged, the new region will be online and available for server requests, and the old regions are taken offline.

For merging two consecutive regions use the following command:

```
hbase> merge_region 'ENCODED_REGIONNAME', 'ENCODED_REGIONNAME'
```

For merging regions that are not adjacent, passing true as the third parameter forces the merge.

```
hbase> merge_region 'ENCODED_REGIONNAME', 'ENCODED_REGIONNAME', true
```

Note:

- This command is slightly different from other region operations. You must pass the encoded region name (ENCODED_REGIONNAME), not the full region name . The encoded region name is the hash suffix on region names. For example, if the region name is TestTable,0094429456,1289497600452.527db22f95c8a9e0116f0cc13c680396, the encoded region name portion is 527db22f95c8a9e0116f0cc13c680396.
- Cloudera recommends that you do not perform region merge operations on the Phoenix slated tables. This might affect the region boundaries and produce incorrect query results.

Move HBase Master Role to another host

You can move the HBase Master Role from one host to another using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the HBase service.
2. Click the Instances tab.
3. Click Add Role Instances and add role instances to HBase.
4. Choose the new host under Master.
5. Click Continue.
6. Start the newly added HBase Master role.
   The state of the role becomes Started.
7. Wait until the type of the newly added HBase Master role bacomes Master (Backup).
8. Stop any other non-active HBase Master role instances.

   This step does not impact HBase. It is required to ensure that the newly created HBase Master role backup will be chosen to be the new active HBase Master role.
9. Stop the remaining active HBase Master role.
   The type of the newly added HBase Master role automatically bacomes Master (Active).
10. Delete the old HBase Master role instances on hosts that are not wanted.

Expose HBase metrics to a Ganglia server

You can expose HBase metrics to Ganglia instance so that Ganglia, an open-source monitoring framework, can detect potential problems with your HBase cluster.

Procedure

1. In Cloudera Manager, select the HBase service.
2. Click the Configuration tab.
3. Select the HBase Master or RegionServer role.

Configure each role to monitor both:

4. Search for metrics2.
5. Find the Hadoop Metrics2 Advanced Configuration Snippet (Safety Valve) property.
6. Add the following snippet to the property, substituting the server information with your own:

```
hbase.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.Ganglia
Sink31
hbase.sink.ganglia.servers=<Ganglia server>:<port>
hbase.sink.ganglia.period=10
```

7. To apply this configuration property to other role groups as needed, edit the value for the appropriate role group.
8. Click Save Changes.
9. Restart the role.
10. Restart the HBase service.

HBase metrics

Use Java Management Extensions (JMX) APIs to collect the metrics exposed by the various HBase services. You can also export the metrics in Prometheus format.

You can use various HBase metrics to understand the state of your cluster.

Using JMX for accessing HBase metrics

You can access HBase metrics over Java Management Extensions (JMX) through either the HBase web interface or by directly accessing the JMX remote agent.

About this task

You can access JMX metrics through the HBase web interface.

Procedure

1. In Cloudera Manager, select the HBase service.
2. Click the HBase Web UI tab.
   The HBase portal is launched in a separate window.
3. Click Metrics JMX to view the HBase metrics using JMX.
   Alternatively, use the following command format to access the JMX remote agent.

   ```
   curl -i http://<localhost>:<port>/jmx
   ```

   You can use the qry parameter to fetch only a particular key.

   ```
   curl -i http://<localhost>:<port>/jmx?qry=Hadoop:service=NameNode,name=N
   ameNodeInfo
   ```

   a) Click Metrics JMX with description to view the HBase metrics along with the descriptions for the items.
      Alternatively, use the following command to access the HBase metrics with descriptions.

      ```
      curl -i http://<localhost>:<port>/jmx?description=true
      ```

Results

The HBase metrics are displayed in the JMX format.

Accessing HBase metrics in Prometheus format

You can access HBase metrics in Prometheus format through the HBase web interface.

Procedure

1. In Cloudera Manager, select the HBase service.

2. Click the HBase Web UI tab.
   The HBase portal is launched in a separate window.
3. Click Metrics Prometheus to view the HBase metrics in Prometheus format.

   Alternatively, use the following command to access the metrics.

   ```
   curl -i http://<localhost>:<port>/prometheus
   ```

   a) Click Metrics Prometheus with description to view HBase metrics along with the descriptions for the items.

      Alternatively, use the following command to access the metrics with descriptions.

      ```
      curl -i http://<localhost>:<port>/prometheus?description=true
      ```

Results

The HBase metrics are displayed in the Prometheus format.

Cloudera Runtime 7.3.1

Managing Apache Hive

Date published: 2019-08-21
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ACID operations in Cloudera Data Hub.....4
Configuring partitions for transactions.....4
Options to monitor transactions.....4
Options to monitor transaction locks.....6

Data compaction.....8
Compaction tasks.....8
Initiating automatic compaction in Cloudera Manager.....9
Starting compaction manually.....10
Options to monitor compactions.....11
Disabling automatic compaction.....12
Configuring compaction using table properties.....13
Configuring compaction in Cloudera Manager.....13
Configuring the compaction check interval.....14
Compactor properties.....14

Compaction observability in Cloudera Manager.....17
Configuring compaction health monitoring.....17
Monitoring compaction health in Cloudera Manager.....18
Hive ACID metric properties for compaction observability.....18

Query vectorization.....20
Vectorization default.....21

ACID operations in Cloudera Data Hub

Apache Hive supports ACID (atomicity, consistency, isolation, and durability) v2 transactions at the row level without any configuration. Knowing what this support entails helps you determine the table type you create.

By default, managed tables are ACID tables. You cannot disable ACID transactions on managed tables, but you can change the Hive default behavior to create external tables by default to mimic legacy releases. Application development and operations are simplified with strong transactional guarantees and simple semantics for SQL commands. You do not need to bucket ACID v2 tables, so maintenance is easier. With improvements in transactional semantics, advanced optimizations, such as materialized view rewrites and automatic query cache, are available. With these optimizations, you can deploy new Hive application types.

A Hive operation is atomic. The operation either succeeds completely or fails; it does not result in partial data. A Hive operation is also consistent: After an application performs an operation, the results are visible to the application in every subsequent operation. Hive operations are isolated. Your operations do not cause unexpected side effects for other users. Finally, a Hive operation is durable. A completed operation is preserved in the event of a failure.

Hive operations are atomic at the row level instead of the table or partition level. A Hive client can read from a partition at the same time another client adds rows to the partition. Transaction streaming rapidly inserts data into Hive tables and partitions.

Configuring partitions for transactions

You set a couple of parameters, to prevent or permit dynamic partitioning, that inserts, updates, or deletes data into partitions implicitly created on the table.

About this task

Configuring partitioning involves changing the following parameters to meet your needs:

- hive.exec.max.dynamic.partitions
- hive.exec.max.dynamic.partitions.pernode

You set hive.exec.dynamic.partition.mode to strict to prevent dynamic partitioning or to nonstrict (the default) to include INSERT, UPDATE, and DELETE statements in your transaction applications.

Procedure

1. In Cloudera Manager Clusters select the Hive service. Click Configuration, and search for hive-site.xml.
2. In HiveServer2 Advanced Configuration Snippet (Safety Valve) for hive-site.xml, click + and add the hive.exe c.dynamic.partition.mode property.
3. Set the value to nonstrict.
4. Save the changes and restart the Hive service.

Related Information

Hive Configuration Properties documentation on the Apache wiki

Options to monitor transactions

As a Hive administrator, you can view the list of all currently open and aborted transactions using the SHOW TRA NSACTIONS statement or by querying the TRANSACTIONS view within the SYS database.

The query statements display the following details about transactions:

- TXN_ID: Unique internal transaction ID
- STATE: Transaction state

- STARTED: Timestamp when the transaction was started
- LAST_HEARTBEAT: Timestamp of the latest heartbeat
- USER: Hive user who initiated the transaction
- HOST: Host machine or virtual machine where the transaction was initiated
- HEARTBEAT_COUNT: Total number of heartbeats
- TYPE: Transaction type
  - DEFAULT
  - REPL_CREATED
  - READ_ONLY
  - COMPACTION
- TC_DATABASE: Hive database name
- TC_TABLE: Table name
- TC_PARTITION: Partition name (if the table is partitioned)
- TC_OPERATION_TYPE:
  - SELECT
  - INSERT
  - UPDATE
  - COMPACT
- TC_WRITEID: Unique internal write ID

The following sections describe the various options that you can use to monitor transactions.

SHOW TRANSACTIONS

You can run the SHOW TRANSACTIONS statement to view details about all open and aborted transactions.

SHOW TRANSACTIONS;

This statement lists all the transactions and you cannot filter or limit the results as required. Alternatively, you can use the SYS database to query and filter specific transactions.

Querying the SYS database

You can query the TRANSACTIONS view within the SYS database to filter and view specific transactions.

For example, you can run the following query to view transactions in a particular state:

```
SELECT *
FROM SYS.TRANSACTIONS
WHERE STATE='aborted';
```

Important: Currently, you cannot retrieve information about old or aborted transactions from the SYS database that is preventing the Cleanup phase. This is because the MIN_HISTORY_LEVEL table is not exposed in the SYS database. As a workaround, you can run the following query directly on the backend database:

```
SELECT * from TXNS where txn_id = (
    SELECT MIN(res.id) FROM (
        SELECT ntxn_next id from next_txn_id
        UNION ALL
        SELECT MIN(mhl_min_open_txnid) id FROM min_history_level
        UNION ALL
        SELECT MIN(txn_id) id FROM txns WHERE txn_state = 'a'
) res);
```

Note: Similarly, you can also query the INFORMATION_SCHEMA database for details about transactions. You must use the Ranger service and set up access policies for Hive users on this database to make it accessible.

Options to monitor transaction locks

As a Hive administrator, you can view information about locks on a table, partition, or schema that are created as a result of transactions. You can either use the SHOW LOCKS statement or query the LOCKS view within the SYS database to view transaction locks.

Hive transactions, enabled by default, disable ZooKeeper locking. DbLockManager stores and manages all transaction lock information in the Hive Metastore. Heartbeats are sent regularly from lock holders and transaction initiators to the Hive metastore to prevent stale locks and transactions. The lock or transaction is aborted if the metastore does not receive a heartbeat within the amount of time specified by the hive.txn.timeout configuration property.

The query statements display the following details about transaction locks unless ZooKeeper or in-memory lock managers are used:

- LOCK_EXT_ID: Unique internal ID of a lock request that might be requesting multiple lock entries on several resources (tables or partitions).
- LOCK_INT_ID: Unique internal ID of a lock entry that has been requested by a LOCK_EXT_ID
- TXNID: Transaction ID associated with the lock, if one exists
- DB: Hive database name
- TABLE: Table name
- PARTITION: Partition name (if the table is partitioned)
- LOCK_STATE:
  - acquired: transaction initiator holds the lock
  - waiting: transaction initiator is waiting for the lock
  - aborted: the lock has timed out but has not yet been cleaned
- LOCK_TYPE:
  - exclusive: lock cannot be shared. No one else can hold the lock at the same time.
  - shared_read: any number of other shared_read locks can lock the same resource at the same time
  - shared_write: any number of shared_read locks can lock the same resource at the same time, but no other shared_write locks are allowed
- LAST_HEARTBEAT: Last time the holder of this lock sent a heartbeat
- ACQUIRED_AT: Time when the lock was acquired, if it has been acquired
- USER: Hive user who requested the lock
- HOST: Host machine or virtual machine on which the Hive user is running a Hive client
- HEARTBEAT_COUNT: Total number of heartbeats
- BLOCKEDBY_EXT_ID: ID of the lock (LOCK_EXT_ID) causing current lock to be in “waiting” mode, if the lock is in this mode
- BLOCKEDBY_INT_ID: ID of the lock (LOCK_INT_ID) causing current lock to be in “waiting” mode, if the lock is in this mode

The following sections describe the various options that you can use to monitor transaction locks.

SHOW LOCKS

You can run the SHOW LOCKS statement to view details about all transaction locks. Ensure that transactions are enabled

SHOW LOCKS;

The following examples illustrate some sample queries that you can run:

Query to check table locks

SHOW LOCKS mytable EXTENDED;

Query to check partition locks

SHOW LOCKS mytable PARTITION(ds='2018-05-01', hr='12') EXTENDED;

Query to check schema locks

SHOW LOCKS SCHEMA mydatabase;

The SHOW LOCKS SCHEMA cannot be used with ZooKeeper or in-memory lock managers.

The SHOW LOCKS statement lists all the transaction locks and you cannot filter or limit the results as required. Alternatively, you can use the SYS database to query and filter specific locks.

Querying the SYS database

You can query the LOCKS view within the SYS database to filter and view specific locks.

The following examples illustrate how you can run queries on the SYS.LOCKS view to monitor transaction locks:

Query to view locks requested on a particular resource (table or partition)

```
SELECT *
FROM SYS.LOCKS
WHERE db='default'
AND table='tab_acid';
```

Query to view list of acquired locks

```
SELECT *
 FROM SYS.LOCKS
 WHERE lock_state='acquired';
```

Query to view blocking transactions that are preventing locks requested by a user- defined transaction, from being acquired

```
SELECT *
FROM SYS.TRANSACTIONS
WHERE txn_id IN (
  SELECT txnid
  FROM SYS.LOCKS
    INNER JOIN (
  SELECT blockedby_ext_id, blockedby_int_id
  FROM SYS.LOCKS
    WHERE txnid=4534) b
  ON lock_ext_id = b.blockedby_ext_id
    AND lock_int_id = b.blockedby_int_id
);
```

Note: Similarly, you can also query the INFORMATION_SCHEMA database for details about transaction locks. You must use the Ranger service and set up access policies for Hive users on this database to make it accessible.

Related Information

Apache wiki transaction configuration documentation

Data compaction

As administrator, you need to manage compaction of delta files that accumulate during data ingestion. Compaction is a process that performs critical cleanup of files.

Hive creates a set of delta files for each transaction that alters a table or partition. By default, compaction of delta and base files occurs at regular intervals. Compactions occur in the background without affecting concurrent reads and writes.

There are two types of compaction:

- Minor

  Rewrites a set of delta files to a single delta file for a bucket.
- Major

  Rewrites one or more delta files and the base file as a new base file for a bucket.

  Carefully consider the need for a major compaction as this process can consume significant system resources and take a long time. Base and delta files for a table or partition are compacted.

  You can configure automatic compactions or do manual compactions. Start a major compaction during periods of low traffic. You use an ALTER TABLE statement to start compaction manually. A manual compaction either returns the accepted compaction request ID or shows the ID (and current state) of a compaction request for the very same target. The request is stored in the COMPACTION_QUEUE table.

The compactor initiator must run on only one HMS instance at a time.

Related Information

Apache Wiki transactions and compaction documentation

Compaction tasks

Compaction in Hive goes hand-in-hand with Hive ACID. Compaction is not, however, necessarily required for Hive ACID. You need to understand when you want, or do not want, compaction to occur.

If you confine your ACID operations tables to full reloads and some delta merges, the performance is steady. As tables are always rewritten (dropped and recreated), there is no need for compaction. Consider disabling compaction by

Compaction occurs for the following reasons:

- You explicitly trigger compaction on a table or partition.
- Automatic compaction finds something to compact.

You run an ALTER TABLE statement to start explicit compaction. Automatic compaction happens without your intervention when Hive periodically crawls through the transaction information, finds tables/partitions affected and those fit for the pre-defined conditions, and marks them for compaction. Conditions are based on the number of deltas, amount of change, and so on.

Compaction of sorted tables is not supported.

In a data pipeline, creating staging or temporary tables can significantly increase the compaction throughput. Avoid compaction of these tables.

Initiating automatic compaction in Cloudera Manager

Several properties in the Hive and Hive metastore service configurations must be set to enable automatic compaction. You need to check that the property settings are correct and to add one of the properties to the Hive on Tez service. Automatic compaction will then occur at regular intervals, but only if necessary.

About this task

Initiator threads should run in only one Hive Metastore server (even in high-availability / HA configurations). The Hive metastore instances will elect a single leader among themselves. There is no need to override the hive.compact or.initiator.on on the Hive metastore instance level. For more information, see *Hive Metastore leader election*.

Disable Initiator threads in all the Cloudera Data Hub clusters' Hive service, the compaction initiator thread can be run by the leader HMS in the DataLake cluster. Set the following properties:

- In Hive metastore (Hive-1) service:
  - hive.compactor.initiator.on = true (default)
- In Hive on Tez service:
  - hive.compactor.worker.threads = <a value greater than 0> (default and recommended value = 5)
  - hive.metastore.runworker.in = hs2 (default)

Before you begin

Tables or partitions you are compacting must be full ACID or insert-only ACID tables.

Procedure

1. In Cloudera Manager, select the Hive metastore service: Clusters Hive-1 Configuration .
2. Search for compact.

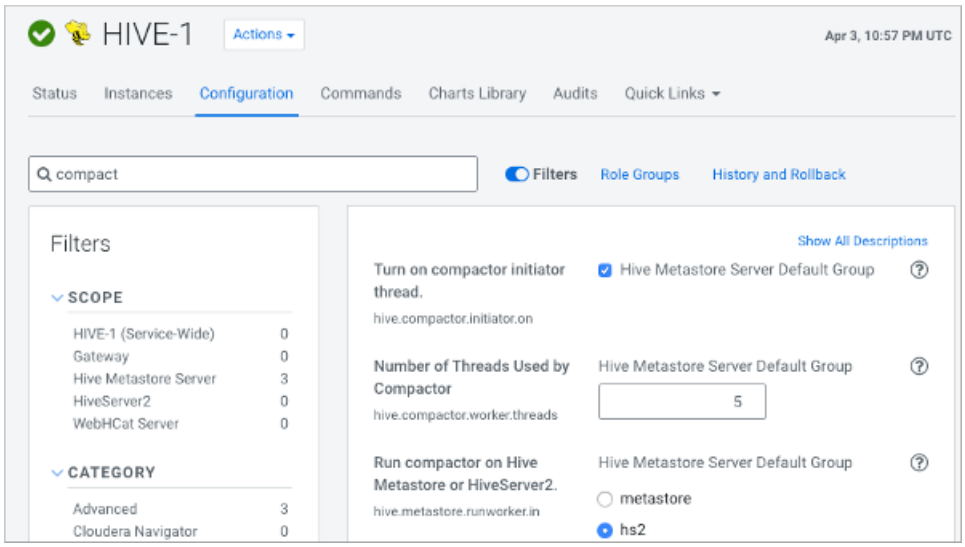

3. Check that Turn on Compactor Initiator Thread (hive.compactor.initiator.on), Number of Threads Used by Compactor (hive.compactor.worker.threads), and Run Compactor on Hive Metastore or HiveServer2 (hive.met astore.runworker.in) are set to the values shown above.
4. Save the changes.
5. In Cloudera Manager, select the Hive on Tez service: Clusters HIVE_ON_TEZ-1 Configuration .

6. Search for compact.

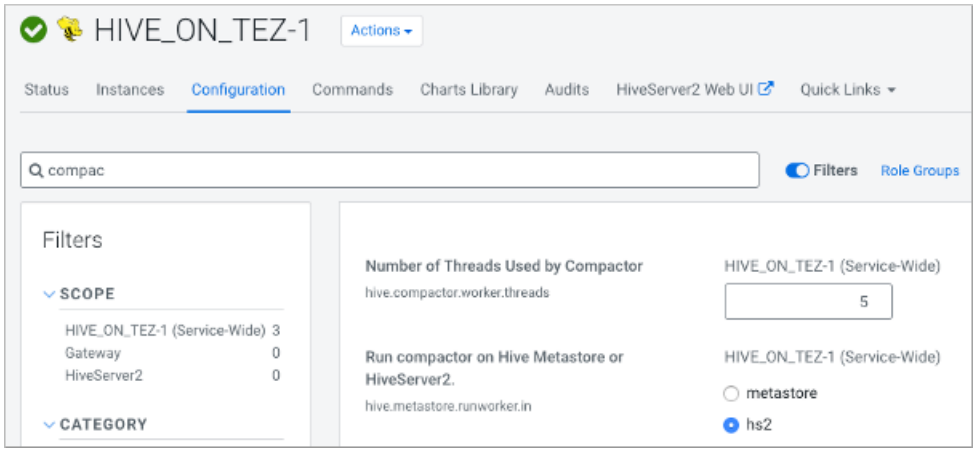

7. Check that the Number of Threads Used by Compactor (hive.compactor.worker.threads), and Run compactor on Hive Metastore or HiveServer2 (hive.metastore.runworker.in) is set to hs2.
8. Save the changes and restart the Hive on Tez and Hive (HIVE-1) metastore services at an appropriate time.

Related Information

Hive Metastore leader election

Starting compaction manually

You manually start compaction when automatic compaction fails for some reason. You can start compaction by running a Hive statement.

About this task

You can run compaction pseudo-synchronously using the AND WAIT clause. Compaction actually occurs asynchronously, but seems synchronous. The compaction request is recorded and queued, and remains in a waiting cycle, querying the status of the compaction in the background until a failure, success, or timeout occurs. The hive .compactor.wait.timeout (default: 300s) property sets the timeout.

Start compaction using a query

You use the following syntax to issue a query that starts compaction:

```
 ALTER TABLE tablename [PARTITION (partition_key='partition_value' [,...])]
COMPACT 'compaction_type'
```

Before you begin

- Tables or partitions you are compacting must be full ACID or insert-only ACID tables.
- Compaction must be enabled (initiator hive.compactor.initiator.on=true)

Procedure

1. Run a query to start a major compaction of a table.

   ```
   ALTER TABLE mytable COMPACT 'major'
   ```

   Use the COMPACT 'minor' clause to run a minor compaction. ALTER TABLE compacts tables even if the NO_AUTO_COMPACTION table property is set.

2. Start compaction in a pseudo-synchronous way.

```
ALTER TABLE mydb.mytable PARTITION (mypart='myval') COMPACT 'MAJOR' AND
WAIT;
```

Options to monitor compactions

You can view the progress of compactions using the SHOW COMPACTIONS statement or by querying the COMP ACTIONS view within the SYS database.

The query statement displays the following details about compactions:

- C_ID: Unique internal compaction ID
- C_DATABASE: Hive database name
- C_TABLE: Table name
- C_PARTITION: Partition name (if the table is partitioned)
- C_TYPE: Major or minor compaction
- C_STATE: Compaction state
  - initiated: waiting in queue to be compacted
  - working: currently being compacted
  - ready for cleaning: compaction completed and old files scheduled for removal
  - failed: compaction job failed. Details are printed to the metastore log.
  - succeeded: compaction job completed successfully
  - attempted: initiator attempted to schedule a compaction but failed or hive.compactor.initiator.failed.compacts .threshold is breached and compaction is skipped . Details are printed to the metastore log.
- C_WORKER_HOST: Thread ID of the worker thread performing the compaction
- C_WORKER_ID: Unique internal worker ID
- C_WORKER_VERSION: Version of distribution with running worker
- C_ENQUEUE_TIME: Time when compaction request was placed
- C_START: Start time of the compaction job
- C_DURATION: Duration (in ms) of the compaction job
- C_HADOOP_JOB_ID: ID of the submitted MapReduce job for MR compaction. 'None' for query-based compaction.
- C_INITIATOR_HOST: Host where initiator is configured
- C_INITIATOR_ID: Unique internal initiator ID
- C_INITIATOR_VERSION: Version of distribution with active initiator

Note: Details about "initiators" and "workers" are new additions that you can use to investigate the compaction setup in an environment. If you are unable to retrieve these details using the options listed here, you can run the following query on the backend database:

```
SELECT activity.pid,
       activity.usename,
       activity.client_addr,
       activity.client_hostname,
       activity.query,
       blocking.pid AS blocking_pid,
       blocking.query AS blocking_query,
       blocking.client_addr AS blocking_client_addr,
       blocking.client_hostname AS blocking_client_hostname
FROM pg_stat_activity AS activity
JOIN pg_stat_activity AS blocking
    ON blocking.pid = ANY(pg_blocking_pids(activity.pid))
WHERE blocking.query like '%AUX_TABLE%';
```

The following sections describe the various options that you can use to monitor compactions.

SHOW COMPACTIONS

You can run the SHOW COMPACTIONS statement to view details about all the compaction jobs.

SHOW COMPACTIONS;

Authorization is not required to use the SHOW COMPACTIONS statement. Therefore, every user can view compactions and their current state.

Since the statement lists all the compaction jobs, you cannot filter or limit the results as required. Alternatively, you can use the SYS database to query and filter specific compactions.

Querying the SYS database

The SYS database in the Hive metastore contains the following sources that you can use to monitor compactions:

- COMPACTION_QUEUE: Stores information about compaction requests - both explicit user submitted and compactions discovered by the initiator and enqueued.
- COMPLETED_COMPACTIONS: Stores the same information present in COMPACTION_QUEUE along with the compaction's end time. Records are moved to the completed table when the cleaner finishes the job on the compaction request.
- COMPACTIONS: A view over the COMPACTION_QUEUE and COMPLETED_COMPACTIONS tables that resolves the column values to human readable format.

The SYS database tables and views are treated as normal Hive external tables or views, and therefore, standard access policies can be configured against these sources.

The following examples illustrate how you can run queries on the SYS.COMPACTIONS view to monitor compactions:

Query to display the last 10 failed compactions

```
SELECT *
FROM SYS.COMPACTIONS
WHERE C_STATE='failed'
ORDERBY C_ID DESC
LIMIT 10;
```

Query to check the status of a specific compaction using the compaction ID

```
SELECT *
 FROM SYS.COMPACTIONS
 WHERE C_ID='1234';
```

Query to view the total number of compactions in a particular state

```
SELECT COUNT(*)
 FROM SYS.COMPACTIONS
 WHERE C_STATE='ready for cleaning';
```

Note: Similarly, you can also query the INFORMATION_SCHEMA database for details about compactions. You must use the Ranger service and set up access policies for Hive users on this database to make it accessible.

Disabling automatic compaction

You can disable automatic compaction of a particular Hive ACID table by setting a Hive table property. By default, compaction is enabled, so you must enter an ALTER TABLE command to disable it.

About this task

Compaction of a full ACID table is skipped under the following conditions:

- Another compaction is either running or already initiated for the target.
- The compaction target table is already dropped.
- Table is explicitly configured to be skipped by the auto-compaction

Compaction of an insert-only, ACID table is skipped if hive.compactor.compact.insert.only is set to false (turned off). The default is true. Although you can disable automatic compaction, tables still can be compacted if you explicitly request compaction.Disabling automatic compaction does not prevent you from performing manual compaction.

The compaction auto-initiator can be disabled on service instance level (disabled by default). You can independently enable or disable compaction workers on service instance level. Compaction merges only the bucket files with the same index and keeps the same number of bucket files in base. Compaction does not rebalance the rows between the buckets.

Procedure

At the Hive JDBC client prompt, in the database of the target table, alter the TBLPROPERTIES.

```
ALTER TABLE my_t SET TBLPROPERTIES ('NO_AUTO_COMPACTION'='true');
```

Configuring compaction using table properties

You see how to configure compaction using table properties and learn about the advantage of using this method of configuration.

About this task

You can configure compaction using table properties. Using table properties, you can group all table and partition compactions into a specific queue to match your use case. You can also size the compactor job based on the tables parameters like size and compression.

Procedure

Set table properties to adjust the compaction initiator properties.

```
ALTER TABLE mydb.mytable
SET TBLPROPERTIES (
'compactorthreshold.hive.compactor.delta.pct.threshold'='0.2f',
'compactorthreshold.hive.compactor.delta.num.threshold'='20');
```

These properties change thresholds, as the names imply: the deltas/base size percentage override threshold and the number of deltas threshold.

Configuring compaction in Cloudera Manager

You see how to configure compaction properties exposed in Cloudera Manager. You rarely need to configure other compaction properties, but need to know how to do this.

About this task

You can configure Hive metastore (HMS) and HiveServer (HS2) properties. Compaction properties you typically configure appear in Cloudera Manager. You search for a property to change it. Occasionally, you might add a property to hive-site or core-site using the Cloudera Manager Safety Valve.

Procedure

1. In Cloudera Manager, click Clusters Hive Configuration , and search for a property, such as hive.compactor.worker.threads.

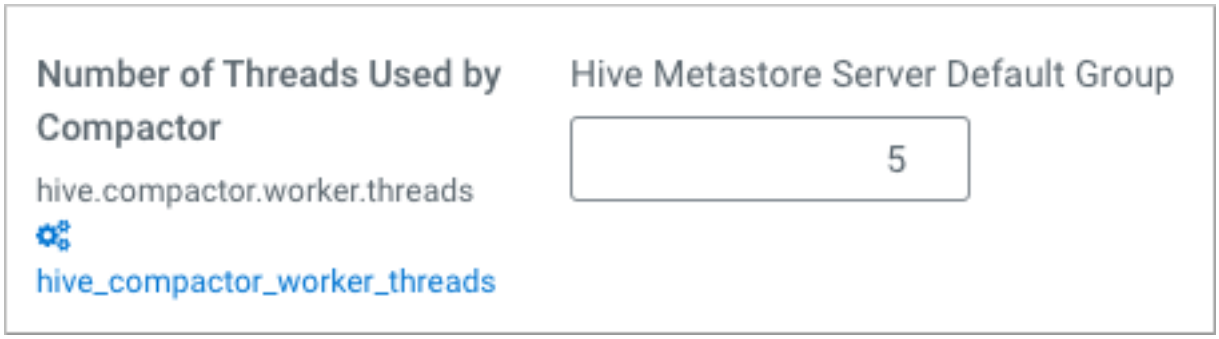

2. Change the value, and save.
3. Restart the Hive cluster.

Configuring the compaction check interval

You need to know when and how to control the compaction process checking, performed in the background, of the file system for changes that require compaction.

When you turn on the compaction initiator, consider setting the hive.compactor.check.interval property. This property determines how often the initiator should search for possible tables, partitions, or compaction. By default, the value for this property is set to 300 seconds. Decreasing hive.compactor.check.interval has the following effect:

- Reduces the time it takes for compaction to be started for a table or partition that requires compaction.
- Requires several calls to the file system for each table or partition that has undergone a transaction since the last major compaction, resulting in increases to the load on the filesystem.

The compaction initiator first checks the completed transactions (COMPLETED_TXN_COMPONENTS), excluding those that already have completed compactions, searching for potential compaction targets. The search of the first iteration includes all transactions. Further searching of iterations are limited to the time-frame since the last iteration.

To configure the compaction check interval, set the hive.compactor.check.interval. For example:

From Cloudera Manager, go to Clusters Hive Metastore Configuration and set the value for the hive.compactor.check.interval parameter in the Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml file.

Compactor properties

You check and change a number of Apache Hive properties to configure the compaction of delta files that accumulate during data ingestion. You need to know the defaults and valid values.

Basic compactor properties

hive.compactor.initiator.on

Default=false

Whether to run the initiator threads on this metastore instance or not. Set in only one instance.

hive.compactor.cleaner.on

Default=false

Whether to run the cleaner threads on this metastore instance or not. Set in only one instance.

Note: Before HIVE-26908, the cleaner thread could only be enabled or disabled through the hive.compactor.initiator.on property. With the introduction of the new cleaner property, you can enable or disable the cleaner and initiator threads independently.

hive.compactor.worker.threads

Default=0

Set this to a positive number to enable Hive transactions, which are required to trigger transactions. Worker threads spawn jobs to perform compactions, but do not perform the compactions themselves. Increasing the number of worker threads decreases the time that it takes tables or partitions to be compacted. However, increasing the number of worker threads also increases the background load on the Cloudera cluster because they cause more jobs to run in the background.

hive.metastore.runworker.in

Default=HS2

Specifies where to run the Worker threads that spawn jobs to perform compactions. Valid values are HiveServer (HS2) or Hive metastore (HMS).

hive.compactor.abortedtxn.threshold

Default=1000 aborts

The number of aborted transactions that triggers compaction on a table/partition.

hive.compactor.aborted.txn.time.threshold

Default=12 hours

The hours of aborted transactions that trigger compaction on a table/partition.

Advanced compactor properties

hive.compactor.worker.timeout

Default=86400s

Expects a time value using d/day, h/hour, m/min, s/sec, ms/msec, us/usec, or ns/nsec (default is sec). Time in seconds after which a compaction job will be declared failed and the compaction re-queued.

hive.compactor.check.interval

Default=300s

A valid value is a time with unit (d/day, h/hour, m/min, s/sec, ms/msec, us/usec, ns/nsec), which is sec if not specified.

Time in seconds between checks to see if any tables or partitions need to be compacted. This value should be kept high because each check for compaction requires many calls against the NameNode. Decreasing this value reduces the time it takes to start compaction for a table or partition that requires it. However, checking if compaction is needed requires several calls to the NameNode for each table or partition involved in a transaction done since the last major compaction. Consequently, decreasing this value increases the load on the NameNode.

hive.compactor.delta.num.threshold

Default=10

Number of delta directories in a table or partition that triggers a minor compaction.

hive.compactor.delta.pct.threshold

Default=0.1

Percentage (fractional) size of the delta files relative to the base that triggers a major compaction. (1.0 = 100%, so the default 0.1 = 10%.)

hive.compactor.max.num.delta

Default=500

Maximum number of delta files that the compactor attempts to handle in a single job.

hive.compactor.wait.timeout

Default=300000

The value must be greater than 2000 milliseconds.

Time out in milliseconds for blocking compaction.

hive.compactor.initiator.failed.compacts.threshold

Default=2

A valid value is between 1 and 20, and must be less than hive.compactor.history.retention.failed.

The number of consecutive compaction failures (per table/partition) after which automatic compactions are not scheduled any longer.

hive.compactor.cleaner.run.interval

Default=5000ms

A valid value is a time with unit (d/day, h/hour, m/min, s/sec, ms/msec, us/usec, ns/nsec), which is msec if not specified.

The time between runs of the cleaner thread.

hive.compactor.job.queue

Specifies the Hadoop queue name to which compaction jobs are submitted. If the value is an empty string, Hadoop chooses the default queue to submit compaction jobs.

Providing an invalid queue name results in compaction job failures.

hive.compactor.compact.insert.only

Default=true

The compactor compacts insert-only tables, or not (false). A safety switch.

hive.compactor.crud.query.based

Default=false

Performs major compaction on full CRUD tables as a query, and disables minor compaction.

hive.split.grouping.mode

Default=query

A valid value is either query or compactor.

This property is set to compactor from within the query-based compactor. This setting enables the Tez SplitGrouper to group splits based on their bucket number, so that all rows from different bucket files for the same bucket number can end up in the same bucket file after the compaction.

hive.compactor.history.retention.succeeded

Default=3

A valid value is between 0 and 100.

Determines how many successful compaction records are retained in compaction history for a given table/partition.

hive.compactor.history.retention.failed

Default=3

A valid value is between 0 and 100.

Determines how many failed compaction records are retained in compaction history for a given table/partition.

hive.compactor.history.retention.attempted

Default=2

A valid value is between 0 and 100.

Determines how many attempted compaction records are retained in compaction history for a given table/partition.

hive.compactor.history.reaper.interval

Default=2m

A valid value is a time with unit (d/day, h/hour, m/min, s/sec, ms/msec, us/usec, ns/nsec), which is msec if not specified.

Determines how often the compaction history reaper runs.

Compaction observability in Cloudera Manager

Compaction observability is a notification and information system based on metrics about the health of the compaction process. A healthy compaction process is critical to query performance, availability, and uptime of your data warehouse. You learn how to use compaction observability to prevent serious problems from developing.

Compaction runs in the background. At regular intervals, Hive accesses the health of the compaction process, and logs an error in the event of a problem. The assessment is based on metrics, such as the number of rows in the metadata table TXN_TO_WRITE_ID and the age of the longest running transaction (oldest_open_txn_age_in_sec). For example, if compaction is not running, the TXN_TO_WRITE_ID table in the HMS backend database becomes bloated and queries slow down.

Compaction health monitoring provides the following information to help you proactively address the problems before the problems become an emergency:

- Warnings and errors indicating problems in compaction processes
- Charts of various metrics that provide information about compaction health
- Recommended actions to address suboptimal configurations

Compaction observability does not attempt to do root cause analysis (RCA) and does not attempt to fix the underlying problem. Compaction observability helps you quickly respond to symptoms of compaction problems.

Using Cloudera Manager, you can view compaction health checks for the Hive Metastore and Hive on Tez services, view actions and advice related to configurations and thresholds, and use the Compaction tab from the Hive Metastore service to view compaction-related charts based on the collected metrics.

Configuring compaction health monitoring

As an administrator, you can use Cloudera Manager to enable or disable compaction health tests for the Hive Metastore (HMS) and HiveServer (HS2) services, configure how those health tests factor into the overall health of the service, and modify thresholds for the status of certain health tests.

About this task

You can configure HMS and HS2 compaction properties that are exposed in Cloudera Manager by searching for the property. Occasionally, you might add a property to hive-site.xml or core-site.xml using the Hive Service Advanced Configuration Snippet (Safety Valve).

Procedure

1. In Cloudera Manager, click Clusters Hive Configuration to navigate to the configuration page for HMS.
2. Search for Hive Compaction Health Test, enable the property, and save the changes.
3. Enable the Hive Compaction Health Test property for the HS2 service by going to Clusters Hive on Tez Configuration .
4. To configure the Health Check thresholds, go to Clusters Hive Configuration .

5. Select the Category Monitoring filter.
6. Modify the warning and critical (error) threshold values for the required compaction-related thresholds and save the changes.

Results

The **Health Tests** panel of HMS and HS2 service displays the compaction health test results, typically with specific metrics that the test returned.

Note: For the HS2 service, if there is no delta metric data (metric values are too low to be collected), then the health check displays a "Disabled" status.

Monitoring compaction health in Cloudera Manager

Using Cloudera Manager, you can view compaction health checks for the Hive Metastore (HMS) and Hive on Tez services, view actions and advice related to configurations and thresholds, and use the Compaction tab from the Hive Metastore service to view compaction-related charts based on the collected metrics.

Before you begin

- You must have enabled compaction health checks.
- You must have enabled the Hive ACID metrics service to collect ACID-related metrics.

Procedure

1. Sign in to Cloudera Manager as an Administrator.
2. Go to the HMS service by clicking Clusters Hive .
   The **Health Tests** panel in the **Status** tab displays the compaction health test results, typically with the specific metrics that the test returned.

   Similarly, you can view the compaction health test results for the HiveServer (HS2) service by going to Clusters Hive on Tez .
3. From the HMS service page, click the Compactions tab to view dashboards of compaction-related charts of various metrics.

   You can create triggers for the charts that allow you to define actions to be taken when a specified condition is met.

Hive ACID metric properties for compaction observability

As an administrator, you must configure certain properties related to the Hive ACID metrics service to enable the collection of Hive ACID metrics that are required to display compaction-related alerts and charts in Cloudera Manager.

Basic properties

You must configure the following metrics to enable Hive ACID metrics collection:

hive.metastore.metrics.enabled

Default value: True

Enables or disables Hive metrics subsystem for the Hive Metastore (HMS) role.

hive.server2.metrics.enabled

Default value: True

Enables or disables Hive metrics subsystem for the HiveServer (HS2) role.

hive.metastore.acidmetrics.ext.on

Default value: True

Set this to property to True to collect additional acid related metrics outside of the Hive ACID metrics service. Enable this property for both HMS and HS2 roles.

Note: Ensure that you have also enabled the hive.metastore.metrics.enabled property for HMS and the hive.server2.metrics.enabled property for HS2.

metastore.acidmetrics.thread.on

Default value: True

Specifies whether to run Hive ACID related metrics collection on this metastore instance. Enable this property in only one HMS instance and disable the property in all other instances.

Advanced properties

Set these additional properties to configure how Hive ACID metrics service collects metric data:

The following properties have to be configured in the HMS instance where metastore.acidmetrics.thread.on is enabled:

metastore.acidmetrics.check.interval

Default value: 300 seconds

Specifies the time (in seconds) between Hive ACID metric collection jobs.

metastore.acidmetrics.table.aborted.txns.threshold

Default value: 1500

Hive metrics subsystem collects the number of tables that have a large number of aborted transactions. This property specifies the minimum number of aborted transactions required to consider a table.

metastore.compactor.acid.metrics.logger.frequency

Default value: 360m

Specifies the logging frequency (in minutes) of Hive ACID metrics. Set this property in only one HMS instance and set the property to 0 in all other HMS instances to disable logging.

The following properties have to be configured in the HS2 (Hive on Tez) service:

hive.txn.acid.metrics.cache.size

Default value: 100

Specifies the size of the ACID metrics cache, which is the maximum number of partitioned and unpartitioned tables with the most deltas that is included in the list of active, obsolete, and small deltas. Valid values are between 0 to 500.

hive.txn.acid.metrics.cache.duration

Default value: 7200 seconds

Specifies the maximum lifetime (in seconds) for an entry in the ACID metrics cache.

hive.txn.acid.metrics.reporting.interval

Default value: 30 seconds

Specifies the reporting period (in seconds) for ACID metrics.

hive.txn.acid.metrics.delta.num.threshold

Default value: 100

Specifies the minimum number of active delta files that a table or partition must have to be included in the ACID metrics report.

hive.txn.acid.metrics.obsolete.delta.num.threshold

Default value: 100

Specifies the minimum number of obsolete delta files that a table or partition must have to be included in the ACID metrics report.

hive.txn.acid.metrics.delta.check.threshold

Default value: 300 seconds

Specifies the minimum age (in seconds) for delta files to be included in the ACID metrics report. Delta files less than this age are not included in the report.

hive.txn.acid.metrics.delta.pct.threshold

Default value: 0.01

Specifies the percentage size of the delta files relative to the base directory. Delta files whose size is smaller than this threshold value are considered as small deltas and are not included in the ACID metrics report. (1 = 100%, so the default 0.01 = 1%)

Query vectorization

You can use vectorization to improve instruction pipelines for certain data and queries and to optimize how Hive uses the cache. Vectorization processes batches of primitive types on the entire column rather than one row at a time.

Unsupported functionality on vectorized data

Some functionality is not supported on vectorized data:

- DDL queries
- DML queries other than single table, read-only queries
- Formats other than Optimized Row Columnar (ORC)

Supported functionality on vectorized data

The following functionality is supported on vectorized data:

- Single table, read-only queries

  Selecting, filtering, and grouping data is supported.
- Partitioned tables
- The following expressions:
  - Comparison: >, >=, <, <=, =, !=
  - Arithmetic plus, minus, multiply, divide, and modulo
  - Logical AND and OR
  - Aggregates sum, avg, count, min, and max

Supported data types

You can query data of the following types using vectorized queries:

- tinyint
- smallint
- int
- bigint
- date
- boolean
- float
- double
- timestamp

- stringchar
- varchar
- binary

Vectorization default

Vectorized query execution can affect performance. You need to be aware of the Boolean default value of hive.vectorized.execution.enabled.

Vectorized query execution is enabled by default (true). Vectorized query execution processes Hive data in batch, channeling a large number of rows of data into columns, foregoing intermediate results. This technique is more efficient than the MapReduce execution process that stores temporary files.

Cloudera Runtime 7.2.11

Managing Apache ZooKeeper

Date published: 2019-08-21
Date modified: 2021-09-09

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Add a ZooKeeper service.....4

Use multiple ZooKeeper services.....4

Replace a ZooKeeper disk.....4

Replace a ZooKeeper role with ZooKeeper service downtime.....5

Replace a ZooKeeper role without ZooKeeper service downtime.....6

Replace a ZooKeeper role on an unmanaged cluster.....7

Confirm the election status of a ZooKeeper service.....7

Add a ZooKeeper service

If the Add Service wizard does not finish successfully, you can add the ZooKeeper service manually.

About this task

In a production environment, you should deploy ZooKeeper as an ensemble with an odd number of servers. As long as a majority of the servers in the ensemble are available, the ZooKeeper service will be available. The minimum recommended ensemble size is three ZooKeeper servers, and Cloudera recommends that each server run on a separate machine. In addition, the ZooKeeper server process should have its own dedicated disk storage if possible.

When adding the ZooKeeper service, the Add Service wizard automatically initializes the data directories.

When you add Zookeeper servers to an existing ensemble, a rolling restart of all zookeeper is required in order to allow all zookeeper servers to have the same configurations.

If you quit the Add Service wizard or it does not finish successfully, you can initialize the directories outside the wizard by following these steps:

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click Action.
3. Select Initialize.
4. Click Initialize again to confirm.

   If the data directories are not initialized, the ZooKeeper servers cannot be started.

Use multiple ZooKeeper services

Only services that are not dependent can use different ZooKeeper services.

Cloudera Manager requires dependent services within Cloudera Runtime to use the same ZooKeeper service. If you configure dependent Cloudera Runtime services to use different ZooKeeper services, Cloudera Manager reports the following error:

```
com.cloudera.cmf.command.CmdExecException:java.lang.RuntimeException: java.l
ang.IllegalStateException: Assumption violated:
getAllDependencies returned multiple distinct services of the same type
at SeqFlowCmd.java line 120
in com.cloudera.cmf.command.flow.SeqFlowCmd run()
```

Because of a known limitation in Cloudera Manager, Cloudera Runtime services that are depending on each other can not use different ZooKeeper services. For example, Kafka does not depend on any services other than ZooKeeper, so Kafka can use one ZooKeeper cluster and the rest of the Cloudera Runtime services can use another one. At the same time, since Ranger also depends on ZooKeeper, if Kafka gets configured to use Ranger, then both Kafka and Ranger must use the same ZooKeeper cluster.

Replace a ZooKeeper disk

You can replace a ZooKeeper Disk but first you must stop the ZooKeeper role.

About this task

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Search for data directory.
4. Find the Data Directory property and update as applicable.
5. Find the Transaction Log Directory property and update as applicable.
6. Click Save Chages.
7. Click the Instances tab.
8. Select one ZooKeeper role.
9. Click the Actions for Selected button.
10. Select Stop.
11. Move the contents to the new disk location and modify mounts as needed.
    Ensure that the permissions and ownership are correct.
12. Select the same ZooKeeper role.
13. Click the Actions for Selected button.
14. Select Start.
15. Repeat step 8-14 for any remaining ZooKeeper roles.

Replace a ZooKeeper role with ZooKeeper service downtime

You can replace a ZooKeeper role with ZooKeeper service downtime.

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Instances tab.
3. Select the ZooKeeper role on the old host.
4. Click the Actions for Selected button.
5. Select Stop.
6. Click the Actions for Selected button.
7. Select Delete.
8. Click the Add Role Instances button.
9. Select the new host.
10. Click Continue.
11. Confirm the changes.
12. Click Continue.
13. Restart the old ZooKeeper servers that have outdated configuration.
14. Select the ZooKeeper service.
15. Click the Status tab.
16. Confirm the ZooKeeper service elected one of the restarted hosts as a leader.
17. Restart the newly added ZooKeeper server.

18. Restart any dependent services, such as HBase, HDFS, YARN or Hive, that are marked to have stale configuration.

Replace a ZooKeeper role without ZooKeeper service downtime

If server to server SASL authentication is not enabled, you can replace a ZooKeeper role without ZooKeeper service downtime.

About this task

This process is valid only if the SASL authentication is not enabled between the ZooKeeper servers. You can check this in Cloudera Manager, if you see the Enable Server to Server SASL Authentication under the Configuration of ZooKeeper.

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Instances tab.
3. Select the ZooKeeper role on the old host.
4. Click the Actions for Selected button.
5. Select Stop.
6. Wait for 1 minute.
7. Click the Status tab.
8. Confirm that ZooKeeper service elected one of the remaining hosts as a leader.
9. Click the Instances tab.
10. Click the Add Role Instances button.
11. Select the new host.
12. Click Continue.
13. Confirm the changes.
14. Click Continue.
15. Click the newly added Server instance.
16. Click the Configuration tab.
17. Find the ZooKeeper Server ID property.
18. Change its value, so that it is the highest ZooKeeper Service ID set in the cluster.
19. Delete the old ZooKeeper role.
    a) Go back to the ZooKeeper cluster.
    b) Click Instances.
    c) Select the old ZooKeeper role.
    d) Click the Actions for Selected button.
    e) Click Delete.
20. Click the Status tab.
21. Click Actions.
22. Select Start this Server.
    The new ZooKeeper role is started.
23. Restart the ZooKeeper server that has an outdated configuration and is a follower.
24. Restart the leader ZooKeeper server that has an outdated configuration.
25. Confirm that a leader is elected after the restart, and the whole ZooKeeper service is in green state.

26. Restart any dependent services, such as HBase, HDFS, YARN or Hive, that are marked to have stale configuration.

Replace a ZooKeeper role on an unmanaged cluster

You can replace a ZooKeeper Role on an unmanaged cluster, using command line.

About this task

These instructions assume you are using ZooKeeper from the command line.

Procedure

1. Stop the ZooKeeper role on the old host.
2. Confirm the ZooKeeper Quorum elected a leader.
3. Add a new ZooKeeper role on the new server.
4. Identify the dataDir locatiom from the zoo.cfg file.

   This default to /var/lib/zookeeper.
5. Identify the ID number for the ZooKeeper Server from the myid file in the configuration: cat*/VAR/LIB/ZOOKEEPER*/myid.
6. On all the ZooKeeper hosts, edit the zoo.cfg file, so the server ID references the new server hostname.

   For example:

   ```
   server.1=zk1.example.org:3181:4181 server.2=zk2.example.org:3181:4181 se
   rver.4=zk4.example.org:3181:4181
   ```
7. Restart the ZooKeeper hosts.
8. Confirm the ZooKeeper Quorum elected a leader and the other hosts are followers.
9. Restart any dependent services, such as HBase, HDFS Failover Controllers with HDFS High Availability, or YARN with High Availability.
10. Perform a failover to make on HDFS NameNode active.

Confirm the election status of a ZooKeeper service

You can determine the election status of a ZooKeeper host if you have installed telnet ot netcat and it runs from a host with network access to the ZooKeeper host.

The default ZooKeeper client port is 2181. Run the following command against each ZooKeeper host:

```
echo "stat" | nc server.example.org 2181 | grep Mode
```

For example, a follower host would return the message:

```
Mode: follower
```

You can use telnet, if you prefer:

```
$ telnet server.example.org 2181
```

Sample output would be similar to the following:

```
Trying 10.1.2.154...
```

```
Connected to server.example.org.
Escape character is '^]'.
stat
Zookeeper version: 3.4.5-cdh5.4.4--1, built on 07/06/2015 23:54 GMT
...

Latency min/avg/max: 0/1/40
Received: 631
Sent: 677
Connections: 7
Outstanding: 0
Zxid: 0x30000011a
Mode: follower               <----
Node count: 40
Connection closed by foreign host.
```

Cloudera Runtime 7.3.1

Apache Hive Materialized View Commands

Date published: 2019-08-21
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ALTER MATERIALIZED VIEW REBUILD..... 4

ALTER MATERIALIZED VIEW REWRITE..... 4

CREATE MATERIALIZED VIEW.....5

DESCRIBE EXTENDED and DESCRIBE FORMATTED.....6

DROP MATERIALIZED VIEW.....8

SHOW MATERIALIZED VIEWS.....8

ALTER MATERIALIZED VIEW REBUILD

You must rebuild the materialized view to keep it up-to-date when changes to the data occur.

Syntax

```
ALTER MATERIALIZED VIEW [db_name.]materialized_view_name REBUILD;
```

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by the name of the materialized view in dot notation.

Description

A rewrite of a query based on a stale materialized view does not occur automatically. If you want a rewrite of a stale or possibly stale materialized view, you can force a rewrite. For example, you might want to use the contents of a materialized view of a non-transactional table because the freshness of such a table is unknown. To enable rewriting of a query based on a stale materialized view, you can run the rebuild operation periodically and set the following property: hive.materializedview.rewriting.time.window. For example, SET hive.materializedview.rewriting.time.window=10min;

Example

ALTER MATERIALIZED VIEW mydb.mv1 REBUILD;

Related Information

Using materialized views

Periodically rebuild a materialized view

ALTER MATERIALIZED VIEW REWRITE

You can enable or disable the rewriting of queries based on a particular materialized view.

Syntax

```
ALTER MATERIALIZED VIEW [db_name.]materialized_view_name ENABLE|DISABLE REWR
ITE;
```

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by the name for the materialized view in dot notation.

Description

To optimize performance, by default, Hive rewrites a query based on materialized views. You can change this behavior to manage query planning and execution manually. By setting the hive.materializedview.rewriting global property, you can manage query rewriting based on materialized views for all queries.

Example

ALTER MATERIALIZED VIEW mydb.mv1 DISABLE REWRITE;

Related Information

Using materialized views

CREATE MATERIALIZED VIEW

If you are familiar with the CREATE TABLE AS SELECT (CTAS) statement, you can quickly master how to create a materialized view.

Syntax

```
CREATE MATERIALIZED VIEW [IF NOT EXISTS] [db_name.]materialized_view_name
   [DISABLE REWRITE]
   [COMMENT materialized_view_comment]
   [PARTITIONED ON (column_name, ...)]
   [
     [ROW FORMAT row_format]
     [STORED AS file_format]
        | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (ser
de_property_name=serde_property_value, ...)]
   ]
   [LOCATION file_path]
   [TBLPROPERTIES (tbl_property_name=tbl_property_value, ...)]
   AS
   <query>;
```

Required Parameters

QUERY

The query to run for results that populate the contents of the materialized view.

Optional Parameters

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by a name, unique among materialized view names, for the materialized view. The name must be a valid a table name, including case-insensitive alphanumeric and underscore characters.

MATERIALIZED_VIEW_COMMENT

A string literal enclosed in single quotation marks.

COLUMN_NAME

A key that determines how to do the partitioning, which divides the view of the table into parts.

'STORAGE.HANDLER.CLASS.NAME'

The name of a storage handler, such as org.apache.hadoop.hive.druid.DruidStorageHandler, that conforms to the Apache Hive specifications for storage handlers in a table definition that uses the STORED BY clause. The default is hive.materializedview.fileformat.

SERDE_PROPERTY_NAME

A property supported by SERDEPROPERTIES that you specify as part of the STORED BY clause. The property is passed to the serde provided by the storage handler. When not specified, Hive uses the default hive.materializedview.serde.

SERDE_PROPERTY_VALUE

A value of the SERDEPROPERTIES property.

FILE_PATH

The location on the file system for storing the materialized view.

TBL_PROPERTY_NAME

A key that conforms to the Apache Hive specification for TBLPROPERTIES keys in a table.

TBL_PROPERTY_VALUE

The value of a TBLPROPERTIES key.

Usage

The materialized view creation statement meets the criteria of being atomic: it does not return incomplete results. By default, the optimizer uses materialized views to rewrite the query. You can store a materialized view in an external storage system using the STORED AS clause followed by a valid storage handler class name. You can set the DISABLE REWRITE option to alter automatic rewriting of the query at materialized view creation time. The table on which you base the materialized view, src in the example below, must be an ACID, managed table.

Examples

```
CREATE MATERIALIZED VIEW druid_t
  STORED BY 'org.apache.hadoop.hive.druid.DruidStorageHandler'
  AS SELECT a, b, c
  FROM src;
```

```
CREATE MATERIALIZED VIEW mv4
  LOCATION '/user/csso_max'
  AS SELECT empid, deptname, hire_date
  FROM emps JOIN depts
  ON (emps.deptno = depts.deptno)
  WHERE hire_date >= '2017-01-01';
```

Note: Renaming materialized view in Hive is currently not supported.

Related Information

Apache Hive Wiki Hive Data Definition Language > Create Table and CTAS

Apache Hive Wiki StorageHandlers > DDL

Using materialized views

DESCRIBE EXTENDED and DESCRIBE FORMATTED

You can get extensive formatted and unformatted information about a materialized view.

Syntax

```
DESCRIBE [EXTENDED | FORMATTED] [db_name.]materialized_view_name;
```

DB_NAME

The database name.

MATERIALIZED_VIEW_NAME

The name of the materialized view.

Examples

Get summary, details, and formatted information about the materialized view in the default database and its partitions.

```
DESCRIBE FORMATTED default.partition_mv_1;
```

Example output is:

col_name	data_type	comment
# col_name	data_type	comment
name	varchar(256)	
	NULL	NULL
# Partition Information	NULL	NULL
# col_name	data_type	comment
deptno	int	
	NULL	NULL
# Detailed Table Information	NULL	NULL
Database:	default	NULL
OwnerType:	USER	NULL
Owner:	hive	NULL
CreateTime:	Wed Aug 24 19:46:08 UTC 2022	NULL
LastAccessTime:	UNKNOWN	NULL
Retention:	0	NULL
Location:	hdfs://myserver:8020/warehouse/ tables pace/managed/hive/partition_mv_1	NULL
Table Type:	MATERIALIZED_VIEW	NULL
Table Parameters:	NULL	NULL
	COLUMN_STATS_ACCURATE	{\"BASIC_STATS\":\"true\"}
	bucketing_version	2
	numFiles	2
	numPartitions	2
	numRows	4
	rawDataSize	380
	totalSize	585
	transient_lastDdlTime	1534967168
	NULL	NULL
# Storage Information	NULL	NULL
SerDe Library:	org.apache.hadoop.hive.ql.io.orc.OrcSerde	NULL
InputFormat:	org.apache.hadoop.hive.ql.io.orc.OrcInputFor mat	NULL
OutputFormat:	org.apache.hadoop.hive.ql.io.orc.OrcOutputFo rmat	NULL
Compressed:	No	NULL
Num Buckets:	-1	NULL
Bucket Columns:	[]	NULL
Sort Columns:	[]	NULL
	NULL	NULL
# Materialized View Information	NULL	NULL
Original Query:	SELECT hire_date, deptno FROM emps W HERE deptno > 100 AND deptno < 200	NULL

col_name	data_type	comment
Expanded Query:	SELECT `hire_date`, `deptno` FROM (S ELECT `emps`.`hire_date`, `emps`.`dept no` FROM `default`.`emps` WHERE `e mps`.`deptno` > 100 AND `emps`.`deptno ` < 200) `default.partition_mv_1`	NULL
Rewrite Enabled:	Yes	NULL
Outdated for Rewriting:	No	NULL

Related Information

Using materialized views

DROP MATERIALIZED VIEW

You can avoid making a table name unusable by dropping a dependent materialized view before dropping a table.

Syntax

```
DROP MATERIALIZED VIEW [db_name.]materialized_view_name;
```

DB_NAME.MATERIALIZED_VIEW_NAME

The database name followed by a name for the materialized view in dot notation.

Description

Dropping a table that is used by a materialized view is not allowed and prevents you from creating another table of the same name. You must drop the materialized view before dropping the tables.

Example

DROP MATERIALIZED VIEW mydb.mv1;

Related Information

Using materialized views

SHOW MATERIALIZED VIEWS

You can list all materialized views in the current database or in another database. You can filter a list of materialized views in a specified database using regular expression wildcards.

Syntax

```
SHOW MATERIALIZED VIEWS [IN db_name];
```

DB_NAME

The database name.

'IDENTIFIER_WITH_WILDCARDS'

The name of the materialized view or a regular expression consisting of part of the name plus wildcards. The asterisk and pipe (* and |) wildcards are supported. Use single quotation marks to enclose the identifier.

Examples

SHOW MATERIALIZED VIEWS;

```
+-----------------+--------------------+-----------------+
|     mv_name     |  rewrite_enabled   |      mode       |
+-----------------+--------------------+-----------------+
| # MV Name       | Rewriting Enabled  | Mode            |
| partition_mv_1  | Yes                | Manual refresh  |
| partition_mv_2  | Yes                | Manual refresh  |
| partition_mv_3  | Yes                | Manual refresh  |
+-----------------+--------------------+-----------------+
```

SHOW MATERIALIZED VIEWS '*1';

```
+-----------------+--------------------+-----------------+
|     mv_name     |  rewrite_enabled   |      mode       |
+-----------------+--------------------+-----------------+
| # MV Name       | Rewriting Enabled  | Mode            |
| partition_mv_1  | Yes                | Manual refresh  |
|                 | NULL               | NULL            |
+-----------------+--------------------+-----------------+
```

Related Information

Using materialized views

Cloudera Runtime 7.2.11

Migrating Data Using Sqoop

Date published: 2019-08-21
Date modified: 2021-09-08

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Data migration to Apache Hive 4

Sqoop enhancements to the Hive import process 4
- Configuring custom Beeline arguments 4
- Configuring custom Hive JDBC arguments 6
- Configuring a custom Hive CREATE TABLE statement 8
- Configuring custom Hive table properties 9

Secure options to provide Hive password during a Sqoop import 11
- Providing the Hive password through a prompt 11
- Providing the Hive password through a file 12
- Providing the Hive password through an alias 12
- Providing the Hive password through an alias in a file 13
- Saving the password to Hive Metastore 14

Imports into Hive 16
- Creating a Sqoop import command 16
- Importing RDBMS data into Hive 18

Import command options 19

Data migration to Apache Hive

To migrate data from an RDBMS, such as MySQL, to Hive, you should consider Apache Sqoop in Cloudera with the Teradata connector. Apache Sqoop client CLI-based tool transfers data in bulk between relational databases and HDFS or cloud object stores including Amazon S3 and Microsoft ADLS.

The source of legacy system data that needs to undergo an extract, transform, and load (ETL) process typically resides on the file system or object store. You can also import data in delimited text (default) or SequenceFile format, and then convert data to ORC format recommended for Hive. Generally, for querying the data in Hive, ORC is the preferred format because of the performance enhancements ORC provides.

Teradata Connector for Sqoop

Cloudera does not support the Sqoop exports using the Hadoop jar command (the Java API). The connector documentation from Teradata includes instructions that include the use of this API. Cloudera users have reportedly mistaken the unsupported API commands, such as -forcestage, for the supported Sqoop commands, such as –-staging-force. Cloudera supports the use of Sqoop only with commands documented in Using the Cloudera Connector Powered by Teradata. Cloudera does not support using Sqoop with Hadoop jar commands, such as those described in the Teradata Connector for Hadoop Tutorial.

Apache Sqoop documentation on the Cloudera web site

To access the latest Sqoop documentation, go to Sqoop Documentation 1.4.7.7.1.6.0.

Sqoop enhancements to the Hive import process

Learn how you can leverage the various Sqoop enhancements that enable you to configure how Sqoop imports data from relational databases into Hive.

Sqoop had limited capabilities in executing Hive processes that often resulted in a lack of control of the imported data and the corresponding Hive table properties. With the enhancements, you can now specify custom Beeline arguments, define custom Hive JDBC arguments, choose how tables are created in Hive using custom CREATE TABLE statements, and configure custom Hive table properties. The changes allow users to control the imported data according to their specific requirements.

In addition, the enhancements also address the variability across Hive versions and their corresponding table creation semantics. Depending on the Hive version, a CREATE TABLE statement can have different implications, such as creating an external table with purge or creating a managed table. Also, the ability to set custom configurations globally not only streamlines the import process but also simplifies the modification of a large number of Sqoop commands.

Configuring custom Beeline arguments

Learn how to configure Sqoop to enable users to include additional Beeline arguments while importing data into Hive using Sqoop.

About this task

You can configure the required property either through Cloudera Manager or by using the --beeline-args argument in your Sqoop import command.

Order of precedence

The configuration set through the Sqoop argument in the command line takes precedence over the configuration specified through Cloudera Manager. If custom Beeline arguments are specified

through both the command line and Cloudera Manager, the command line Sqoop argument overwrites the Cloudera Manager configuration.

Limitations

You must be aware of certain Beeline arguments that cannot be specified. This limitation ensures the integrity and proper functioning of Sqoop's core processes. The following Beeline arguments are not supported:

- -e
- -f
- -n
- -p
- -w
- --password-file
- -a (if the HADOOP_TOKEN_FILE_LOCATION is not specified)

Procedure

1. If you are specifying the custom Beeline arguments through Cloudera Manager, perform the following steps:
   a) In Cloudera Manager, click Clusters and then select the SQOOP_CLIENT-1 service.
   b) From the Sqoop service, go to the **Configuration** tab and search for sqoop.beeline-args.
   c) Click ⊕ and specify the custom Beeline argument and its value.

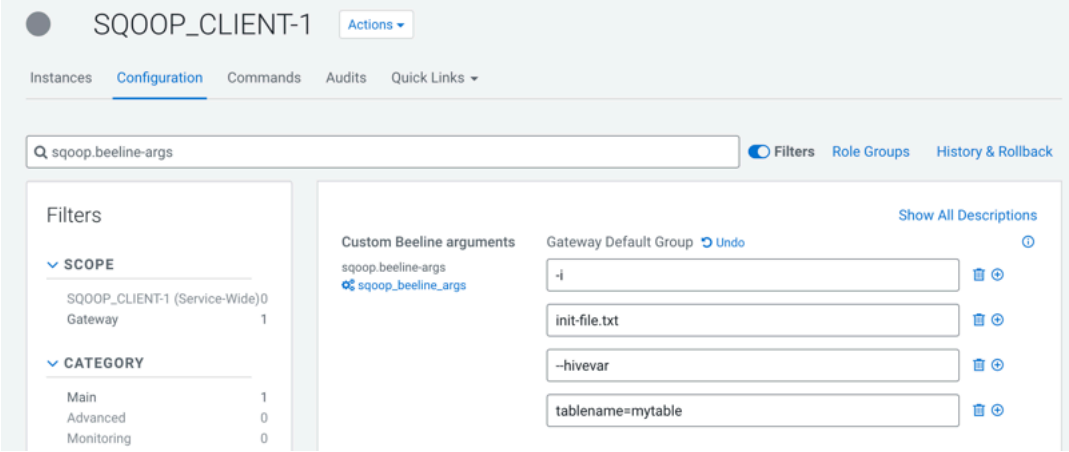

   Important: Each argument and value must be added in a new line using ⊕.

   d) Click Save Changes.
2. If you are specifying the custom Beeline arguments through the Sqoop argument, specify the required argument using the --beeline-args argument while constructing the Sqoop import command.

```
/opt/cloudera/parcels/CDH/bin/sqoop import \
  -Dsqoop.beeline.env.preserve=KRB5CCNAME \
  --connection-manager org.apache.sqoop.manager.MySQLManager \
  --connect jdbc:mysql://db.foo.com:3306/employees \
  --username [***USERNAME***] \
  --password [***PASSWORD***] \
  --table employees \
  --warehouse-dir /user/hrt_qa/test-sqoop \
  --hive-import \
  --delete-target-dir \
  --hive-overwrite \
```

```
--beeline-args -i init_file2.txt --hivevar tablename=mytable2 -r
```

Important: Every argument that you provide must line up following the --beeline-args key.

Note: As part of the upgrade, Sqoop imports into Hive using Beeline instead of the Hive CLI. This change requires that Ranger policies be in place for the target-dir input location, as Cloudera mandates explicit permissions on the target directory to ensure successful data import.

Configuring custom Hive JDBC arguments

Learn how to configure Sqoop to enable users to specify custom Hive JDBC arguments while importing data into Hive using Sqoop. This allows users to set Hive session variables, Hive configuration values, and Hive user variables.

About this task

You can configure the required property either through Cloudera Manager or by using the --jdbc-arg argument in your Sqoop import command. Use the hiveconf: prefix to specify Hive configuration values and the hivevar: prefix to specify Hive user variables.

Order of precedence

The configuration set through the Sqoop argument in the command line takes precedence over the configuration specified through Cloudera Manager. If custom Hive JDBC arguments are specified through both the command line and Cloudera Manager, the command line Sqoop argument does not entirely overwrite the Cloudera Manager configuration. Instead, the distinct values are retained and the values with matching keys are replaced with arguments specified through the command line Sqoop argument.

Procedure

1. If you are specifying the custom Hive JDBC arguments through Cloudera Manager, perform the following steps:
   a) In Cloudera Manager, click Clusters and then select the SQOOP_CLIENT-1 service.
   b) From the Sqoop service, go to the **Configuration** tab and search for sqoop.hive-jdbc-params.
   c) Click ⊕ and specify the custom Hive JDBC argument and its value.

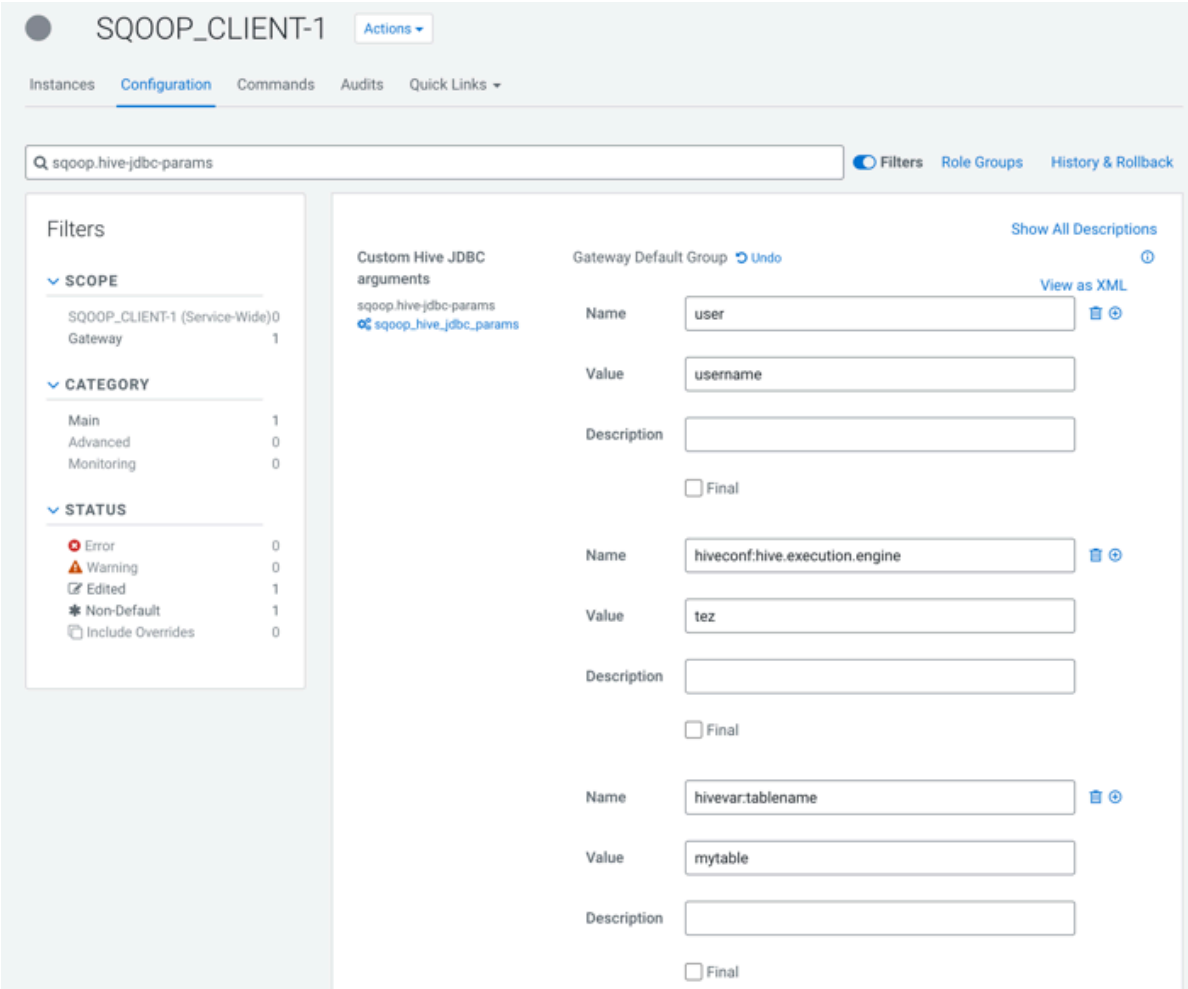

Important: Each key-value pair must be added to a new key-value field using ⊕.

   d) Click Save Changes.

2. If you are specifying the custom Hive JDBC arguments through the Sqoop argument, specify the required key-value pair using the --hive-jdbc-arg argument while constructing the Sqoop import command.

```
/opt/cloudera/parcels/CDH/bin/sqoop import \
  -Dsqoop.beeline.env.preserve=KRB5CCNAME \
  --connection-manager org.apache.sqoop.manager.MySQLManager \
  --connect jdbc:mysql://db.foo.com:3306/employees \
  --username [***USERNAME***] \
  --password [***PASSWORD***] \
  --table employees \
  --warehouse-dir /user/hrt_qa/test-sqoop \
  --hive-import \
  --delete-target-dir \
  --hive-overwrite \
```

```
  --hs2-url "jdbc:hive2://[***HOST***]:[***PORT***];serviceDiscoveryMode=z
ooKeeper;zooKeeperNamespace=hiveserver2;transportMode=http;httpPath=clis
ervice;ssl=true;sslTrustStore=[***TRUSTSTORE PATH***];trustStorePassword
=[***TRUSTSTORE PASSWORD***]" \
  --hive-jdbc-arg user=username \
  --hive-jdbc-arg hiveconf:hive.execution.engine=tez \
  --hive-jdbc-arg hivevar:tablename=mytable
```

Important: Every key-value pair should be added in the following format: --hive-jdbc-arg key1=value1 --hive-jdbc-arg key2=value2

Configuring a custom Hive CREATE TABLE statement

Learn how to configure a custom CREATE TABLE statement that Sqoop uses during the Hive table creation process. For example, if you have configured Sqoop to use the CREATE EXTERNAL TABLE statement, then external tables are created by default during the table creation process.

About this task

You can configure the required property either through Cloudera Manager or by using the --hive-create-table-statement argument in your Sqoop import command.

Order of precedence

The configuration set through the Sqoop argument in the command line takes precedence over the configuration specified through Cloudera Manager.

Procedure

1. If you are specifying the custom CREATE TABLE statement through Cloudera Manager, perform the following steps:
   a) In Cloudera Manager, click Clusters and then select the SQOOP_CLIENT-1 service.
   b) From the Sqoop service, go to the **Configuration** tab and search for sqoop.hive-create-table-statement.
   c) Enter the custom CREATE TABLE statement.

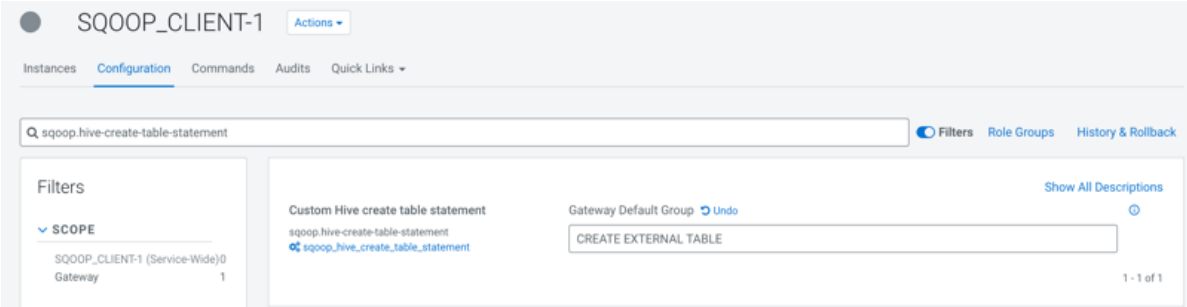

   d) Click Save Changes.
2. If you are specifying the custom CREATE TABLE statement through the Sqoop argument, specify the required statement using the --hive-create-table-statement argument while constructing the Sqoop import command.

```
/opt/cloudera/parcels/CDH/bin/sqoop import \
  -Dsqoop.beeline.env.preserve=KRB5CCNAME \
  --connection-manager org.apache.sqoop.manager.MySQLManager \
  --connect jdbc:mysql://db.foo.com:3306/employees \
  --username [***USERNAME***] \
  --password [***PASSWORD***] \
  --table employees \
  --warehouse-dir /user/hrt_qa/test-sqoop \
  --hive-import \
  --delete-target-dir \
  --hive-overwrite \
```

```
--hive-create-table-statement "CREATE EXTERNAL TABLE"
```

Configuring custom Hive table properties

Learn how to specify custom Hive table properties for the CREATE TABLE statement that Sqoop uses during the Hive table creation process.

About this task

You can configure the required property either through Cloudera Manager or by using the --hive-table-property argument in your Sqoop import command.

For example, if you have specified the custom Hive table properties with the "transactional=true" and "transactional_properties=insert_only" key-value pairs, the Hive CREATE TABLE statement is constructed as follows:

```
CREATE TABLE foo ....
....
TBLPROPERTIES ('transactional'='true', 'transactional_properties'='insert
_only');
```

Order of precedence

The configuration set through the Sqoop argument in the command line takes precedence over the configuration specified through Cloudera Manager. If custom Hive table properties are specified through both the command line and Cloudera Manager, the command line Sqoop argument does not entirely overwrite the Cloudera Manager configuration. Instead, the distinct values are retained and the values with matching keys are replaced with arguments specified through the command line Sqoop argument.

Procedure

1. If you are specifying the custom Hive table properties through Cloudera Manager perform the following steps:
   a) In Cloudera Manager, click Clusters and then select the SQOOP_CLIENT-1 service.
   b) From the Sqoop service, go to the **Configuration** tab and search for sqoop.hive-table-properties.
   c) Click ⊕ and specify the custom Hive table property and its value.

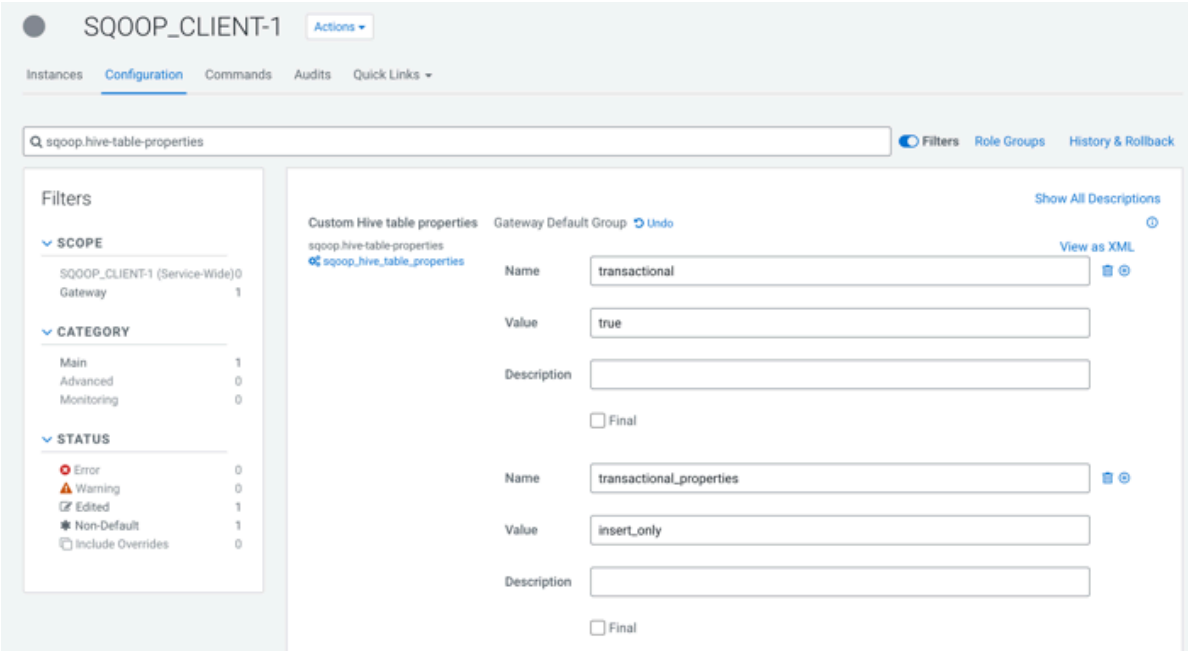

   Important: Each key-value pair must be added to a new key-value field using ⊕.

   d) Click Save Changes.
2. If you are specifying the custom Hive table properties through the Sqoop argument, specify the required table properties using the --hive-table-property argument while constructing the Sqoop import command.

```
/opt/cloudera/parcels/CDH/bin/sqoop import \
  -Dsqoop.beeline.env.preserve=KRB5CCNAME \
  --connection-manager org.apache.sqoop.manager.MySQLManager \
  --connect jdbc:mysql://db.foo.com:3306/employees \
  --username [***USERNAME***] \
  --password [***PASSWORD***] \
  --table employees \
  --warehouse-dir /user/hrt_qa/test-sqoop \
  --hive-import \
  --delete-target-dir \
  --hive-overwrite \
  --hive-table-property transactional=true
  --hive-table-property transactional_properties=insert_only
```

Important: Every key-value pair should be added in the following format: --hive-table-property key1=value1 --hive-table-property key2=value2

Secure options to provide Hive password during a Sqoop import

Learn about the secure options that you can use to provide the Hive password during Sqoop-Hive imports instead of the earlier way of providing the password as plaintext in the command-line interface.

When importing data into Hive using Sqoop and if LDAP authentication is enabled for Hive, the necessity to set the Hive password parameter directly in the command-line poses a potential vulnerability. Passwords provided in plaintext within command-line interfaces are susceptible to unauthorized access or interception, compromising sensitive credentials and, subsequently, the security of the entire data transfer process.

You can use the following Sqoop arguments in your Sqoop import command:

The following Sqoop arguments are introduced that allow you to provide the Hive password in a secure way during the Sqoop-Hive import process:

Sqoop argument	Description
-promptHivePassword	Prompts the user to enter the Hive password
--hive-password-file <***FILE PATH***>	Stores the Hive password in a file and uses it during the import
--hive-password-alias <***ALIAS NAME***>	Stores the Hive password in a Credential Provider facility with an alias associated with the actual value in the credential storage

Along with securely providing the Hive password to Sqoop, it is essential that the password is safely persisted in the Hive metastore when saving a Sqoop job related to Hive.

Providing the Hive password through a prompt

Use the -promptHivePassword argument in the Sqoop import command to prompt users to enter the Hive password manually in the command-line for Sqoop-Hive import processes when LDAP authentication is enabled.

About this task

Procedure

1. Specify the -promptHivePassword argument while constructing the Sqoop import command.

```
/opt/cloudera/parcels/CDH/bin/sqoop import \
  -Dsqoop.beeline.env.preserve=KRB5CCNAME \
  --connection-manager org.apache.sqoop.manager.PostgresqlManager \
  --connect "jdbc:postgresql://db.foo.com:5432/employees" \
  --username [***USERNAME***] \
  --password [***PASSWORD***] \
  --table employees \
  --warehouse-dir \
  /user/hrt_qa/test-sqoop \
  --hive-import \
  --delete-target-dir \
  --hive-overwrite \
  --external-table-dir hdfs:///warehouse/tablespace/external/hive/employee
s \
  --hs2-url "jdbc:hive2://[***HOST***]:[***PORT***];serviceDiscoveryMode
=zooKeeper;zooKeeperNamespace=hiveserver2;transportMode=http;httpPath=cl
```

```
iservice;ssl=true;sslTrustStore=[***TRUSTSTORE PATH***];trustStorePasswo
rd=[***TRUSTSTORE PASSWORD***]" \
  --hive-user guest \
  -promptHivePassword \
  -m 1
```

2. Enter the Hive password in the command-line when you are prompted with the "Enter Hive password:" message.

Providing the Hive password through a file

You can save the Hive password in a file and then set up Sqoop to use this password for Sqoop-Hive import processes when LDAP authentication is enabled.

Procedure

1. Create a file containing the appropriate Hive password. Ensure to set '400' permission on the file so that only the user or owner of the file has read permissions.

   You can save this file either in a local file system or on HDFS.
2. While creating the Sqoop import command, specify the --hive-password-file argument along with the path of the password file you created in the previous step.

```
/opt/cloudera/parcels/CDH/bin/sqoop import \
  -Dsqoop.beeline.env.preserve=KRB5CCNAME \
  --connection-manager org.apache.sqoop.manager.PostgresqlManager \
  --connect "jdbc:postgresql://db.foo.com:5432/employees" \
  --username [***USERNAME***] \
  --password [***PASSWORD***] \
  --table employees \
  --warehouse-dir \
  /user/hrt_qa/test-sqoop \
  --hive-import \
  --delete-target-dir \
  --hive-overwrite \
  --external-table-dir hdfs:///warehouse/tablespace/external/hive/employee
s \
  --hs2-url "jdbc:hive2://[***HOST***]:[***PORT***];serviceDiscoveryMode
=zooKeeper;zooKeeperNamespace=hiveserver2;transportMode=http;httpPath=cl
iservice;ssl=true;sslTrustStore=[***TRUSTSTORE PATH***];trustStorePasswo
rd=[***TRUSTSTORE PASSWORD***]" \
  --hive-user guest \
  --hive-password-file /user/hrt_qa/hivepasswd-storefile \
  -m 1
```

Providing the Hive password through an alias

Learn how you can use an alias to represent the Hive password during Sqoop-Hive import processes when LDAP authentication is enabled.

About this task

The Hive password is stored in a Credential Provider facility and is associated with the alias. During the import, Sqoop resolves the alias and uses the linked password.

Procedure

1. Using the CredentialProvider API, store the Hive password in a user specified provider path and associate it with an alias.

```
/opt/cloudera/parcels/CDH/lib/hadoop/bin/hadoop credential \
  create sqoophive.password.alias \
  -value guest-password \
  -provider jceks://hdfs/user/hive/sqoophivepasswd.jceks
```

2. Add the provider path property in the Sqoop import command pointing to the credential provider URI that should be considered while resolving the credential alias.

   -D hadoop.security.credential.provider.path=<***PROVIDER PATH***>

```
-D hadoop.security.credential.provider.path=jceks://hdfs/user/hive/sqoop
hivepasswd.jceks \
```

3. While creating the Sqoop import command, specify the --hive-password-alias argument with the alias name that you want to resolve.

```
/opt/cloudera/parcels/CDH/bin/sqoop import \
  -Dsqoop.beeline.env.preserve=KRB5CCNAME \
  -D hadoop.security.credential.provider.path=jceks://hdfs/user/hive/
sqoophivepasswd.jceks \
  --connection-manager org.apache.sqoop.manager.PostgresqlManager \
  --connect "jdbc:postgresql://db.foo.com:5432/employees" \
  --username [***USERNAME***] \
  --password [***PASSWORD***] \
  --table employees \
  --warehouse-dir \
  /user/hrt_qa/test-sqoop \
  --hive-import \
  --delete-target-dir \
  --hive-overwrite \
  --external-table-dir hdfs:///warehouse/tablespace/external/hive/employee
s \
  --hs2-url "jdbc:hive2://[***HOST***]:[***PORT***];serviceDiscoveryMode
=zooKeeper;zooKeeperNamespace=hiveserver2;transportMode=http;httpPath=cl
iservice;ssl=true;sslTrustStore=[***TRUSTSTORE PATH***];trustStorePasswo
rd=[***TRUSTSTORE PASSWORD***]" \
  --hive-user guest \
  --hive-password-alias sqoophive.password.alias \
  -m 1
```

Providing the Hive password through an alias in a file

Learn how you can use an alias stored in a file to represent the Hive password during Sqoop-Hive import processes when LDAP authentication is enabled.

About this task

The Hive password is stored in a Credential Provider facility and is associated with the alias. During the import, Sqoop resolves the alias from the file and uses the linked password.

Procedure

1. Using the CredentialProvider API, store the Hive password in a user specified provider path and associate it with an alias.

```
/opt/cloudera/parcels/CDH/lib/hadoop/bin/hadoop credential \
  create sqoophive.password.alias \
  -value guest-password \
  -provider jceks://hdfs/user/hive/sqoophivepasswd.jceks
```

2. Create a file containing the alias that you created in the previous step. Ensure to set '400' permission on the file so that only the user or owner of the file has read permissions.

   You can save this file either in a local file system or on HDFS.

3. Add the provider path property in the Sqoop import command pointing to the credential provider URI that should be considered while resolving the credential alias.

   -D hadoop.security.credential.provider.path=<***PROVIDER PATH***>

```
-D hadoop.security.credential.provider.path=jceks://hdfs/user/hive/sqoop
hivepasswd.jceks \
```

4. Include the following argument in the Sqoop import command to ensure that the content of the file you created will be handled as an alias.

   -D org.apache.sqoop.credentials.loader.class=org.apache.sqoop.util.password.CredentialProviderPasswordLoader

5. While creating the Sqoop import command, specify the --hive-password-file argument along with the path of the alias file you created earlier.

```
/opt/cloudera/parcels/CDH/bin/sqoop import \
  -Dsqoop.beeline.env.preserve=KRB5CCNAME \
  -D hadoop.security.credential.provider.path=jceks://hdfs/user/hive/
sqoophivepasswd.jceks \
  -D
 org.apache.sqoop.credentials.loader.class=org.apache.sqoop.util.password.CredentialP
  --connection-manager org.apache.sqoop.manager.PostgresqlManager \
  --connect "jdbc:postgresql://db.foo.com:5432/employees" \
  --username [***USERNAME***] \
  --password [***PASSWORD***] \
  --table employees \
  --warehouse-dir \
  /user/hrt_qa/test-sqoop \
  --hive-import \
  --delete-target-dir \
  --hive-overwrite \
  --external-table-dir hdfs:///warehouse/tablespace/external/hive/employee
s \
  --hs2-url "jdbc:hive2://[***HOST***]:[***PORT***];serviceDiscoveryMode
=zooKeeper;zooKeeperNamespace=hiveserver2;transportMode=http;httpPath=cl
iservice;ssl=true;sslTrustStore=[***TRUSTSTORE PATH***];trustStorePasswo
rd=[***TRUSTSTORE PASSWORD***]" \
  --hive-user guest \
  --hive-password-file /user/hrt_qa/hivepasswd-storefile \
  -m 1
```

Saving the password to Hive Metastore

Learn how Sqoop stores the password in the Hive Metastore (HMS) when the Sqoop job is saved and how this password is retrieved from the metastore when the job is run.

Along with securely providing the Hive password to Sqoop, it is essential that the password is safely persisted in the Hive metastore when saving a Sqoop job related to Hive.

The following sections describe how Sqoop stores the Hive password in HMS based on the secure option that you have chosen to provide the password during the Sqoop-Hive import process:

Hive password is provided through a file

When you save the Hive password in a file and then set up Sqoop to use this password for Sqoop-Hive import processes, Sqoop stores the path of the password file in HMS when the Sqoop job is saved.

On running the Sqoop job, the file path is retrieved from HMS and the content of the file is used as the Hive password during the job execution.

Note: The file might include the actual Hive password or an alias linked to the password in a Credential Provider facility, provided that the CredentialProviderPasswordLoader and the Credential Provider path is configured correctly.

Hive password is provided through an alias

When you use an alias to represent the Hive password during the Sqoop-Hive import process, Sqoop stores the alias in the HMS when the Sqoop job is saved.

On running the Sqoop job, the value of the alias is retrieved from the HMS and the value is read by Sqoop. This value is then used to obtain the actual Hive password that is stored in a Credential Provider facility and referenced by a provider path. Once the password is acquired, it is used as the Hive password during the job execution.

Hive password is provided in an insecure way

When the Hive password is provided using one of the following arguments, Sqoop does not store the raw or unencrypted password in the HMS for security reasons and instead prompts the user to enter the Hive password.

- --hive-password
- --hs2-password
- -promptHivePassword

However, if you still want to save the password in HMS although it is insecure, you can set the sqoop.hive.store-raw-password to true either by using this as an argument in your Sqoop import command or by adding this to Cloudera Manager Advanced Configuration Snippet (Safety Valve) for sqoop-site.xml. This ensures that the password is persisted in HMS and you can run the Sqoop job without having to enter the password.

Important: The configuration set through the Sqoop argument in the command-line takes precedence over the configuration specified through Cloudera Manager.

Setting the sqoop.hive.store-raw-password property through Cloudera Manager

From Cloudera Manager, go to Clusters > SQOOP CLIENT > Configuration and search for 'Gateway Advanced Configuration Snippet (Safety Valve) for sqoop-site.xml'.

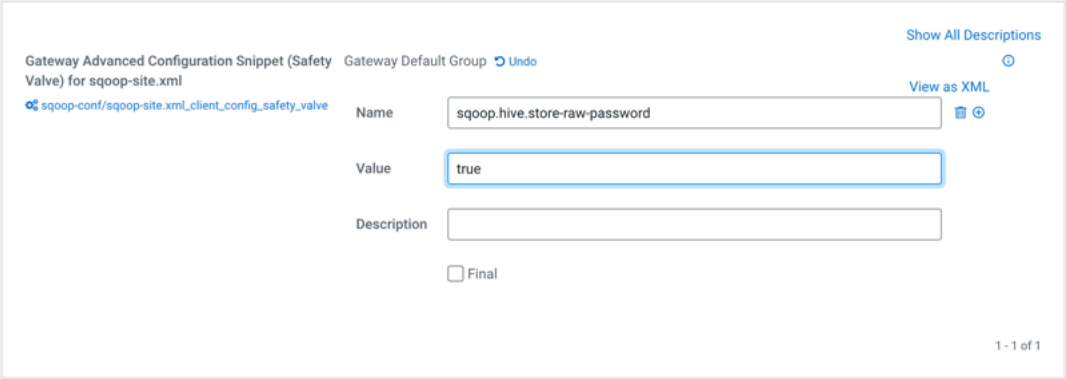

Setting the sqoop.hive.store-raw-password property through the Sqoop command

```
/opt/cloudera/parcels/CDH/bin/sqoop job \
 -Dsqoop.hive.store-raw-password=true \
 --meta-connect "jdbc:postgresql://db.foo.com:5432/employees" \
 --meta-username [***USERNAME***] \
 --meta-password [***PASSWORD***] \
 --create myjob -- import \
 --connection-manager org.apache.sqoop.manager.PostgresqlManager
\
 --connect "jdbc:postgresql://db.foo.com:5432/employees" \
 --username [***USERNAME***] \
 --password [***PASSWORD***] \
 --table employees \
 --warehouse-dir /user/hrt_qa/test-sqoop \
 --hive-import \
 --delete-target-dir \
 --hive-overwrite \
 --external-table-dir hdfs:///warehouse/tablespace/external/hive/
employees \
 --hs2-url "jdbc:hive2://[***HOST***]:[***PORT***];serviceDiscov
eryMode=zooKeeper;zooKeeperNamespace=hiveserver2;transportMode=h
ttp;httpPath=cliservice;ssl=true;sslTrustStore=[***TRUSTSTORE PA
TH***];trustStorePassword=[***TRUSTSTORE PASSWORD***]" \
 --hive-user guest \
 --hive-password [***PASSWORD***] \
 -m 1
```

Imports into Hive

You can use Sqoop to import data from a relational database into for use with Hive. You can import the data directly to Hive. Assuming the Sqoop client service is available in your cluster, you can issue import and export commands from the command line.

Related Information

Apache Sqoop Documentation (v1.4.7.7.1.6)

Creating a Sqoop import command

You create a single Sqoop import command that imports data from diverse data sources, such as a relational database on a different network, into Apache Hive using Apache Sqoop.

About this task

You enter the Sqoop import command on the command line of your Hive cluster to import data from a data source into the cluster file system and Hive. The import can include the following information, for example:

- Database connection information: database URI, database name, and connection protocol, such as jdbc:mysql:
- The data to import
- Parallel processing directives for fast data transfer
- Destination for imported data

Sqoop is tested to work with Connector/J 5.1. If you have upgraded to Connector/J 8.0, and want to use the zeroDate TimeBehavior property to handle values of '0000-00-00\' in DATE columns, explicitly specify zeroDateTimeBeha vior=CONVERT_TO_NULL in the connection string. For example, --connect jdbc:mysql://<MySQL host>/<DB>? zeroDateTimeBehavior=CONVERT_TO_NULL.

Before you begin

It is recommended that you familiarize yourself with the Sqoop configurations that can enable you to control the imported data according to specific requirements.

Procedure

1. Create an import command that specifies the Sqoop connection to the RDBMS.
   - To enter a password for the data source on the command line, use the -P option in the connection string.
   - To specify a file where the password is stored, use the --password-file option.

   Password on command line:

   ```
   sqoop import --connect jdbc:mysql://db.foo.com:3306/bar \
   <data to import> \
   --username <username> \
   -P
   ```

   Specify password file:

   ```
   sqoop import --connect jdbc:mysql://db.foo.com:3306/bar \
   --table EMPLOYEES \
   --username <username> \
   --password-file ${user.home}/.password
   ```

2. Specify the data to import in the command.
   - Import an entire table.
   - Import a subset of the columns.
   - Import data using a free-form query.

   Entire table:

   ```
   sqoop import \
   --connect jdbc:mysql://db.foo.com:3306/bar \
   --table EMPLOYEES
   ```

   Subset of columns:

   ```
   sqoop import
   --connect jdbc:mysql://db.foo.com:3306/bar \
   --table EMPLOYEES \
   --columns "employee_id,first_name,last_name,job_title"
   ```

   Free-form query to import the latest data:

   ```
   sqoop import \
   --connect jdbc:mysql://db.foo.com:3306/bar \
   --table EMPLOYEES \
   --where "start_date > '2018-01-01'"
   ```

3. Optionally, specify write parallelism in the import statement to run a number of map tasks in parallel:
   - Set mappers: If the source table has a primary key, explicitly set the number of mappers using --num-mappers.
   - Split by: If primary keys are not evenly distributed, provide a split key using --split-by
   - Sequential: If you do not have a primary key or split key, import data sequentially using --num-mappers 1 or --autoreset-to-one-mapper in query.
   - Set mappers:

     ```
     sqoop import --connect jdbc:mysql://db.foo.com:3306/bar \
      --table EMPLOYEES \
     ```

```
 --num-mappers 8 \
```

- Split by:

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/bar \
 --table EMPLOYEES \
 --split-by dept_id
```

- Setting mappers evenly splits the primary key range of the source table.
- Split by evenly splits the data using the split key instead of a primary key.

4. Specify importing the data into Hive using Hive default delimiters --hive-import.
5. Specify the Hive destination of the data.
   - If you think the table does not already exist in Hive, name the database and table, and use the --create-hive-table option.
   - If you want to insert the imported data into an existing Hive external table, name the database and table, but do not use the --create-hive-table option.

   This command imports the MySQL EMPLOYEES table to a new Hive table named in the warehouse.

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/corp \
--table EMPLOYEES \
--hive-import \
--create-hive-table \
--hive-database 'mydb' \
--hive-table 'newtable'
```

   This command imports the MySQL EMPLOYEES table to an external table in HDFS.

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/corp \
--table EMPLOYEES \
--hive-import \
--hive-database 'mydb' \
--hive-table 'myexternaltable'
```

   Specify the database and table names, enclosed in single quotation marks, on separate lines (recommended) as shown above. Alternatively specify the database and table names on one line, and enclose the database and table names in backticks.

   --hive-table `mydb`.`myexternaltable`

   Due to the Hive-16907 bug fix, Hive rejects `db.table` in SQL queries. A dot (.) is no longer allowed in table names. You need to change queries that use such references to prevent Hive from interpreting the entire db.table string as the table name.

Related Information

Apache Sqoop Documentation (v1.4.7.7.1.6)

Sqoop enhancements to the Hive import process

Importing RDBMS data into Hive

You can test the Apache Sqoop import command and then run the command to import relational database tables into Apache Hive.

About this task

You enter the Sqoop import command on the command line of your Hive cluster to import data from a data source to Hive. You can test the import statement before actually executing it.

Before you begin

- The Apache Sqoop client service is available.
- The Hive Metastore and Hive services are available.

Procedure

1. Optionally, test the import command before execution using the eval option.

```
sqoop eval --connect jdbc:mysql://db.foo.com/bar \
--query "SELECT * FROM employees LIMIT 10"
```

   The output of the select statement appears listing 10 rows of data from the RDBMS employees table.

2. Rn a Sqoop import command that specifies the Sqoop connection to the RDBMS, the data you want to import, and the destination Hive table name.
   This command imports the MySQL EMPLOYEES table to a new Hive table named in the warehouse.

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/corp \
--table EMPLOYEES \
--hive-import \
--create-hive-table \
--hive-table mydb.newtable
```

Related Information

Apache Sqoop Documentation (v1.4.7.7.1.6)

Import command options

You can use Sqoop command options to import data into Apache Hive.

Table 1: Sqoop Command Options for Importing Data into Hive

Sqoop Command Option	Description
--hive-home <directory>	Overrides $HIVE_HOME.
--hive-import	Imports tables into Hive using Hive's default delimiters if none are explicitly set.
--hive-overwrite	Overwrites existing data in the Hive table.
--create-hive-table	Creates a hive table during the operation. If this option is set and the Hive table already exists, the job will fail. Set to false by default.
--hive-create-table-statement	Specifies a custom CREATE TABLE statement that Sqoop uses during the Hive table creation process.
--hive-table <table_name>	Specifies the table name to use when importing data into Hive.
--hive-table-property	Specifies custom Hive table properties for the CREATE TABLE statement that Sqoop uses during the Hive table creation process.
--hive-drop-import-delims	Drops the delimiters \n, \r, and \01 from string fields when importing data into Hive.
--hive-delims-replacement	Replaces the delimiters \n, \r, and \01 from strings fields with a user-defined string when importing data into Hive.
--hive-partition-key	Specifies the name of the Hive field on which a sharded database is partitioned.
--hive-partition-value <value>	A string value that specifies the partition key for data imported into Hive.

Sqoop Command Option	Description
--map-column-hive <map>	Overrides the default mapping from SQL type to Hive type for configured columns.
--beeline-args	Enables you to include additional Beeline arguments while importing data into Hive.
--hive-jdbc-arg	Enables you to specify custom Hive JDBC arguments, such as Hive session variables, user variables, and configuration values.

Cloudera Runtime 7.2.6

Apache Oozie Reference

Date published: 2021-01-12
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Submit Oozie Jobs in Cloudera Data Engineering Cluster.....4

Submit Oozie Jobs in Cloudera Data Engineering Cluster

You can submit Apache Hive jobs or queries through JDBC, ODBC or Oozie in a Cloudera Data Engineering cluster. Automated batch jobs require a non-interactive mechanism. You can use Cron, Cloudera AI, Amazon Web Services (AWS) Lambda or Fargate, Azure Functions, and many other mechanisms for automated batch jobs. Apache Oozie has built-in integration with the Cloudera Data Engineering's cluster and is a better mechanism for these purposes.

This document explains how to create and use a Cloudera machine user account to submit Oozie jobs to a Cloudera Data Engineering cluster through Knox Oozie endpoint.

Pre-requisites

- You must be a Cloudera user with a PowerUser role. For information about user roles and cluster access, see Enabling admin and user access to environments.
- You must have a Cloudera environment running. For information about the Cloudera environment, see Accessing an environment. For information about CDP CLI, see CDP CLI .

Steps

1. Create a Cloudera machine user account.
2. Set the workload password for the Cloudera machine user.
3. Give the Cloudera machine user the EnvironmentUser role for your Cloudera environment.
4. Perform synchronization of users.
5. Create a Cloudera Data Engineering cluster.
6. Get the Oozie endpoint from the gateway tab in the Cloudera Data Engineering cluster details page.
7. Setup Oozie artifacts.
8. Configure IDBroker mappings.
9. Submit the Oozie job.

- Submit the Oozie job to the Knox endpoint.
- Submit the Oozie job using the CLI.

You can also see an example of how to submit a Spark job using Oozie.

Note: In the above procedure, the first six steps are performed by you with the PowerUser role of a Cloudera tenant. The seventh and eighth steps are performed by the machine user.

1. Create a machine user.

   Create a machine user named *JOB-SUBMITTER* and assign *SRV_JOB-SUBMITTER* to the workloadUsername parameter.

```
$ cdp iam create-machine-user --machine-user-name job-submitter

{
    "machineUser": {
        "machineUserName": "job-submitter",
        "crn": "crn:altus:iam:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9ed
bb73d:machineUser:job-submitter/f622d289-56a1-4cc3-a14e-9c42b7d4988f",
        "creationDate": "2020-04-24T22:19:13.889000+00:00",
        "workloadUsername": "srv_job-submitter"
    }
}
```

2. Set the workload password for the machine user.

   Set the workload password for the machine user. The following example sets the password as *PASSWORD@123*. Use the *ACTOR CRN* from Step 1.

```
$ cdp iam set-workload-password --password 'Password@123' --actor-crn 'c
rn:altus:iam:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:machineUser:
job-submitter/f622d289-56a1-4cc3-a14e-9c42b7d4988f'
```

3. Assign the machine user the EnvironmentUser role for your Cloudera environment. The Resource role crn does not change. Resource crn is the Environment crn.

```
$ cdp iam assign-machine-user-resource-role --machine-user-name job-subm
itter --resource-crn 'crn:cdp:environments:us-west-1:9d74eee4-1cad-45d7-
b645-7ccf9edbb73d:environment:d84ca408-3dbd-4849-946f-c0c194f15429' --re
source-role-crn 'crn:altus:iam:us-west-1:altus:resourceRole:EnvironmentU
ser'
```

4. Perform synchronization of users.

   Synchronize the machine user to be part of the environment users. You must use the Environment name as the Environment crn in the below command.

```
cdp environments sync-all-users --environment-name 'crn:cdp:environments
:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:environment:d84ca408-3db
d-4849-946f-c0c194f15429'
```

5. Create a Cloudera Data Engineering cluster.

```
cdp datahub create-aws-cluster --cluster-name 'gk1de' --environment-name
 'spark-ql5h6p' --cluster-definition-name '7.2.6 - Data Engineering for
AWS'
```

6. Get the Oozie endpoint.

```
cdp datahub describe-cluster --cluster-name gk1de | jq '.cluster.endpoin
ts.endpoints[] | select( .serviceName == "OOZIE_SERVER" ) | .serviceUrl'
```

```
"https://GK1DE-MASTER3.GK1PRIV.A465-9Q4K.CLOUDERA.SITE/GK1DE/cdp-proxy-ap
i/oozie/"
```

7. Setup Oozie artifacts.

   Set up the hive.hql and workflow.xml files in AWS S3.

   - hive.hql contains the hive query you want to execute periodically.
   - workflow.xml contains a boilerplate template to execute hive.hql.
   - The parameters are controlled by job.xml submitted in Step 9.

   The value of the credential name="aws_auth" type="filesystem property is required for the Oozie logs to be saved in cloud storage.

```
<workflow-app name="My Workflow" xmlns="uri:oozie:workflow:1.0">
  <credentials>
   <credential name="aws_auth" type="filesystem">
      <property>
        <name>filesystem.path</name>
        <value>s3a://gk1priv-cdp-bucket</value>
      </property>
    </credential>
    <credential name="hive2" type="hive2">
      <property>
        <name>hive2.jdbc.url</name>
```

```
        <value>${jdbcUrl}</value>
      </property>
      <property>
        <name>hive2.server.principal</name>
        <value>${hs2principal}</value>
      </property>
    </credential>
  </credentials>

    <start to="hive"/>

    <kill name="Kill">
        <message>Action failed, error message[${wf:errorMessage(wf:lastErr
orNode())}]</message>
    </kill>

    <action name="hive" cred="hive2,aws_auth">
        <hive2 xmlns="uri:oozie:hive2-action:1.0">
            <resource-manager>${resourceManager}</resource-manager>
            <name-node>${nameNode}</name-node>
            <jdbc-url>${jdbcUrl}</jdbc-url>
            <script>s3a://gk1priv-cdp-bucket/oozie/hive/hive.hql</script>
        </hive2>
        <ok to="End"/>
        <error to="Kill"/>
    </action>

    <end name="End"/>
</workflow-app>
```

In this example, the hive.hql file is setup at the *S3A://GK1PRIV-CDP-BUCKET/OOZIE/HIVE/HIVE.HQL* location and this *WORKFLOW.XML* file is uploaded to *S3A://GK1PRIV-CDP-BUCKET/OOZIE/HIVE/*. The input and output S3 objects are in the same bucket. So, set the *S3A://GK1PRIV-CDP-BUCKET* in the credentials section of the bucket.

8. Configure IDBroker Mappings.
   a. Select your environment in Cloudera Management Console > Environment.
   b. Click Actions > Manage Access > IDBroker Mappings.
   c. Click Edit.
   d. Add srv_job-submitter in User or Group.
   e. In Role, add the same role listed in the Data Access Role field.
   f. Click Save and Sync.
9. Submit the Oozie job

   You can submit the Oozie job using the knox endpoint or CLI.

   - Submit the Oozie job to the Knox endpoint.

     Submit the *SRV_JOB-SUBMITTER* Oozie job. This is the workloadUserName from Step 1 and the password you have set in Step 2.

     For resourceManager, hs2principal, and jdbcUrl replace the hostname with the hostname in the Oozie endpoint. Specifically for the hs2principal the domain name is the upper cased sub-string (first label is not included) of the hostname in the Oozie endpoint.

```
$ curl --request POST --location https://gk1de-master3.gk1priv.a465-9q4k
.cloudera.site/gk1de/cdp-proxy-api/oozie/v1/jobs?action=start --header '
Content-Type: application/xml;charset=UTF-8' -u 'srv_job-submitter:Passw
ord@123' --data-raw '<?xml version="1.0" encoding="UTF-8"?>
> <configuration>
>     <property>
>         <name>user.name</name>
```

```
>         <value>srv_job-submitter</value>
>     </property>
>     <property>
>         <name>security_enabled</name>
>         <value>True</value>
>     </property>
>     <property>
>         <name>oozie.wf.application.path</name>
>         <value>s3a://gk1priv-cdp-bucket/oozie/hive</value>
>     </property>
>     <property>
>         <name>oozie.use.system.libpath</name>
>         <value>true</value>
>     </property>
>     <property>
>         <name>nameNode</name>
>         <value>will-be-replaced-by-knox</value>
>     </property>
>     <property>
>         <name>resourceManager</name>
>         <value>will-be-replaced-by-knox</value>
>     </property>
>     <property>
>         <name>jobTracker</name>
>         <value>will-be-replaced-by-knox</value>
>     </property>
>     <property>
>         <name>hs2principal</name>
>         <value>hive/gk1de-master3.gk1priv.a465-9q4k.cloudera.site@GK1
PRIV.A465-9Q4K.CLOUDERA.SITE</value>
>     </property>
>     <property>
>         <name>jdbcUrl</name>
>         <value>jdbc:hive2://gk1de-master3.gk1priv.a465-9q4k.clouder
a.site:2181/default;httpPath=cliservice;serviceDiscoveryMode=zooKeeper;s
sl=true;transportMode=http;zooKeeperNamespace=hiveserver2</value>
>     </property>
> </configuration>'
{"id":"0000002-200605195726045-oozie-oozi-W"}
```

Track the id 0000002-200605195726045-oozie-oozi-W for result

```
curl --request GET --location https://gk1de-master3.gk1priv.a465-9q4k.cl
oudera.site/gk1de/cdp-proxy-api/oozie/v1/job/0000002-200605195726045-ooz
ie-oozi-W?show=info -u 'srv_job-submitter:Password@123'
{"appName":"My Workflow","externalId":null,"conf":"<configuration>\r\n
 <property>\r\n    <name>oozie.action.sharelib.for.hive</name>\r\n    <v
alue>hive</value>\r\n  </property>\r\n  <property>\r\n    <name>oozie.wf
.application.path</name>\r\n    <value>s3a://gk1priv-cdp-bucket/oozie/hi
ve</value>\r\n  </property>\r\n  <property>\r\n    <name>oozie.use.syste
m.libpath</name>\r\n    <value>true</value>\r\n  </property>\r\n  <prope
rty>\r\n    <name>security_enabled</name>\r\n    <value>True</value>\r\n
  </property>\r\n  <property>\r\n    <name>jdbcUrl</name>\r\n    <value>
jdbc:hive2://gk1de-master3.gk1priv.a465-9q4k.cloudera.site:2181/default;
httpPath=cliservice;serviceDiscoveryMode=zooKeeper;ssl=true;transportMod
e=http;zooKeeperNamespace=hiveserver2</value>\r\n  </property>\r\n  <pro
perty>\r\n    <name>hs2principal</name>\r\n    <value>hive/gk1de-master3
.gk1priv.a465-9q4k.cloudera.site@GK1PRIV.A465-9Q4K.CLOUDERA.SITE</value>
\r\n  </property>\r\n  <property>\r\n    <name>user.name</name>\r\n    <
value>srv_job-submitter</value>\r\n  </property>\r\n  <property>\r\n
<name>jobTracker</name>\r\n    <value>gk1de-master3.gk1priv.a465-9q4k.cl
oudera.site:8032</value>\r\n  </property>\r\n  <property>\r\n    <name>m
apreduce.job.user.name</name>\r\n    <value>srv_job-submitter</value>\r\
```

```
n  </property>\r\n  <property>\r\n    <name>resourceManager</name>\r\n
  <value>gk1de-master3.gk1priv.a465-9q4k.cloudera.site:8032</value>\r\n
 </property>\r\n  <property>\r\n    <name>nameNode</name>\r\n    <value>
hdfs://gk1de-master3.gk1priv.a465-9q4k.cloudera.site:8020</value>\r\n  <
/property>\r\n</configuration>","run":0,"acl":null,"appPath":"s3a://gk1p
riv-cdp-bucket/oozie/hive","parentId":null,"lastModTime":"Sat, 06 Jun 20
20 00:25:15 GMT","consoleUrl":"https://gk1de-master3.gk1priv.a465-9q4k.c
loudera.site:11443/oozie?job=0000002-200605195726045-oozie-oozi-W","crea
tedTime":"Sat, 06 Jun 2020 00:25:02 GMT","startTime":"Sat, 06 Jun 2020 0
0:25:02 GMT","toString":"Workflow id[0000002-200605195726045-oozie-oozi-
W] status[SUCCEEDED]","id":"0000002-200605195726045-oozie-oozi-W","endTi
me":"Sat, 06 Jun 2020 00:25:15 GMT","user":"srv_job-submitter","actions"
:[{"cred":null,"userRetryMax":0,"trackerUri":"-","data":null,"errorMessa
ge":null,"userRetryCount":0,"externalChildIDs":null,"externalId":"-","er
rorCode":null,"conf":"","type":":START:","transition":"hive","retries":0
,"consoleUrl":"-","stats":null,"userRetryInterval":10,"name":":start:","
startTime":"Sat, 06 Jun 2020 00:25:02 GMT","toString":"Action name[:star
t:] status[OK]","id":"0000002-200605195726045-oozie-oozi-W@:start:","end
Time":"Sat, 06 Jun 2020 00:25:02 GMT","externalStatus":"OK","status":"OK
"},{"cred":"hive2","userRetryMax":0,"trackerUri":"gk1de-master3.gk1priv.
a465-9q4k.cloudera.site:8032","data":null,"errorMessage":null,"userRetry
Count":0,"externalChildIDs":null,"externalId":"application_1591385267488
_0010","errorCode":null,"conf":"<hive2 xmlns=\"uri:oozie:hive2-action:0.
1\">\r\n  <job-tracker>gk1de-master3.gk1priv.a465-9q4k.cloudera.site:803
2</job-tracker>\r\n  <name-node>hdfs://gk1de-master3.gk1priv.a465-9q4k.c
loudera.site:8020</name-node>\r\n  <jdbc-url>jdbc:hive2://gk1de-master3.
gk1priv.a465-9q4k.cloudera.site:2181/default;httpPath=cliservice;service
DiscoveryMode=zooKeeper;ssl=true;transportMode=http;zooKeeperNamespace=h
iveserver2</jdbc-url>\r\n  <script>s3a://gk1priv-cdp-bucket/oozie/hive/h
ive.hql</script>\r\n  <configuration />\r\n</hive2>","type":"hive2","tra
nsition":"End","retries":0,"consoleUrl":"https://gk1de-master3.gk1priv.a
465-9q4k.cloudera.site:8090/proxy/application_1591385267488_0010/","stat
s":null,"userRetryInterval":10,"name":"hive","startTime":"Sat, 06 Jun 20
20 00:25:02 GMT","toString":"Action name[hive] status[OK]","id":"0000002
-200605195726045-oozie-oozi-W@hive","endTime":"Sat, 06 Jun 2020 00:25:15
 GMT","externalStatus":"SUCCEEDED","status":"OK"},{"cred":null,"userRetr
yMax":0,"trackerUri":"-","data":null,"errorMessage":null,"userRetryCount
":0,"externalChildIDs":null,"externalId":"-","errorCode":null,"conf":"",
"type":":END:","transition":null,"retries":0,"consoleUrl":"-","stats":nu
ll,"userRetryInterval":10,"name":"End","startTime":"Sat, 06 Jun 2020 00:
25:15 GMT","toString":"Action name[End] status[OK]","id":"0000002-200605
195726045-oozie-oozi-W@End","endTime":"Sat, 06 Jun 2020 00:25:15 GMT","e
xternalStatus":"OK","status":"OK"}],"status":"SUCCEEDED","group":null}
```


- For Cloudera Data Engineering HA clusters, you can use multiple ZooKeeper services with retries:

```
jdbc:hive2://imb6t-master0.gk1de.xcu2-8y8x.dev.cldr.work:2181,imb6t-mast
er1.gk1de.xcu2-8y8x.dev.cldr.work:2181,imb6t-masterx0.gk1de.xcu2-8y8x.de
v.cldr.work:2181/default;httpPath=cliservice;serviceDiscoveryMode=zooKee
per;ssl=true;transportMode=http;zooKeeperNamespace=hiveserver2;retries=5
```

- Submit the Oozie job using the CLI.

  You can also submit the Oozie job from inside the cluster using the CLI.

  a. Setup the job.properties file:

```
user.name=srv_job-submitter
security_enabled=True
oozie.wf.application.path=s3a://gk1priv-cdp-bucket/oozie/hive
oozie.use.system.libpath=true
nameNode=[Knox will replace this automatically]
resourceManager=[Knox will replace this automatically]
```

```
hs2principal=hive/gk1de-master3.gk1priv.a465-9q4k.cloudera.site@GK1
PRIV.A465-9Q4K.CLOUDERA.SITE
jdbcUrl=jdbc:hive2://gk1de-master3.gk1priv.a465-9q4k.cloudera.site:
2181/default;httpPath=cliservice;serviceDiscoveryMode=zooKeeper;ssl=
true;transportMode=http;zooKeeperNamespace=hiveserver2
```

b. Run the Oozie command line. The workload username is *SRV_JOB-SUBMITTER* and password is the same as that of the machine user.

```
oozie job -oozie {oozieKnoxUrl} -auth BASIC -username {workloadUser}
 -password {workloadPassword} -config job.properties -run
```

c. Check the status.

```
oozie job -oozie {oozieKnoxUrl} -auth BASIC -username {workloadUser}
 -password {workloadPassword} -info 0000002-200605195726045-oozie-oo
zi-W
```

Example - Spark submit using Oozie

```
<workflow-app name="My Workflow" xmlns="uri:oozie:workflow:1.0">
    <start to="spark-8c75"/>
    <kill name="Kill">
        <message>Action failed, error message[${wf:errorMessage(wf:lastErro
rNode())}]</message>
    </kill>
    <action name="spark-8c75">
        <spark xmlns="uri:oozie:spark-action:1.0">
           <resource-manager>${resourceManager}</resource-manager>
            <name-node>${nameNode}</name-node>
            <master>yarn</master>
            <mode>client</mode>
            <name></name>
              <class>org.apache.spark.examples.SparkPi</class>
            <jar>spark-examples_2.11-2.4.5.7.2.2.0-244.jar</jar>
            <file>s3a://gk1priv-cdp-bucket/spark-examples_2.11-2.4.5.7.2.2.
0-244.jar#spark-examples_2.11-2.4.5.7.2.2.0-244.jar</file>
        </spark>
        <ok to="End"/>
        <error to="Kill"/>
    </action>
    <end name="End"/>
</workflow-app>
```

To quickly get started with Oozie for other applications, you could try the Hue Oozie editor. For more information about the Hue Oozie editor, see Hue Oozie Workflow Editor.

Cloudera Runtime 7.2.7

Operational Database Overview

Date published: 2020-02-29
Date modified: 2023-06-05

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to Operational Database..... 4
Introduction to Apache HBase..... 6
Introduction to Apache Phoenix..... 6
Apache Phoenix and SQL..... 6

Introduction to Operational Database

Cloudera Operational Database delivers a real-time, always available, scalable operational database that serves traditional structured data alongside unstructured data within a unified operational and warehousing platform.

Cloudera Operational Database is a Cloudera Public Cloud service for self-service creation of an operational database. Cloudera Operational Database is designed to be used by developers. You do not need a dedicated team of system administrators to run an on-premise cluster of physical servers. You can go directly to Cloudera and provision a new database, therefore letting you rapidly prototype and deploy new applications onto the public cloud.

Apache HBase

Apache HBase is a column-oriented database management system that runs on top of the Hadoop Distributed File System (HDFS), a main component of Apache Hadoop. It enables random, strictly consistent, and real-time access to petabytes of data.

Apache Phoenix

Apache Phoenix is a SQL layer for Apache HBase that provides a programmatic ANSI SQL interface. Apache Phoenix implements best-practice optimizations to enable software engineers to develop HBase based next-generation applications that operationalize big data.

Apache Kafka

Apache Kafka is a distributed streaming platform that is used to build real-time applications.

Apache Spark

Apache Spark is a general framework for distributed computing that offers high performance for both batch and interactive processing. It exposes APIs for Java, Python, and Scala and consists of Spark core and several related projects.

Apache Nifi

Apache Nifi is used to develop, manage, and monitor data flows.

Apache Hive

Apache Hive is a data warehouse software that facilitates reading, writing, and managing large datasets residing in distributed storage using SQL.

Apache Hue

Apache Hue is a SQL assistant for databases and data warehouses.

Apache Impala

Apache Impala is a modern, open source, distributed SQL query engine for Apache Hadoop.

Apache Knox

Apache Know Gateway provides access to Apache Hadoop through proxying of HTTP resources. It provides proxying services, authentication services, and client DSL or SDK services.

Cloudera Operational Database is powered by Apache HBase and Apache Phoenix. In Cloudera Operational Database, you use object store such as Amazon S3 and Microsoft ADLS Gen2 to store the Apache HBase files. You have the choice to either develop applications using one of the native Apache HBase API, or you can use Apache Phoenix for data access. Apache Phoenix is a SQL layer that provides a programmatic ANSI SQL interface. It works on top of Apache HBase, and it makes it possible to handle data using standard SQL queries and Apache Phoenix commands. You can use Cloudera Operational Database in the public cloud or on-premises.

Public Cloud

Cloudera Operational Database that is a managed database platform as a service (dbPaaS).

Private Cloud

- Cloudera Private Cloud Base

Cloudera Operational Database plays the crucial role of a data store in the enterprise data lifecycle. The following graphic shows some of the key elements in a typical operational database deployment.

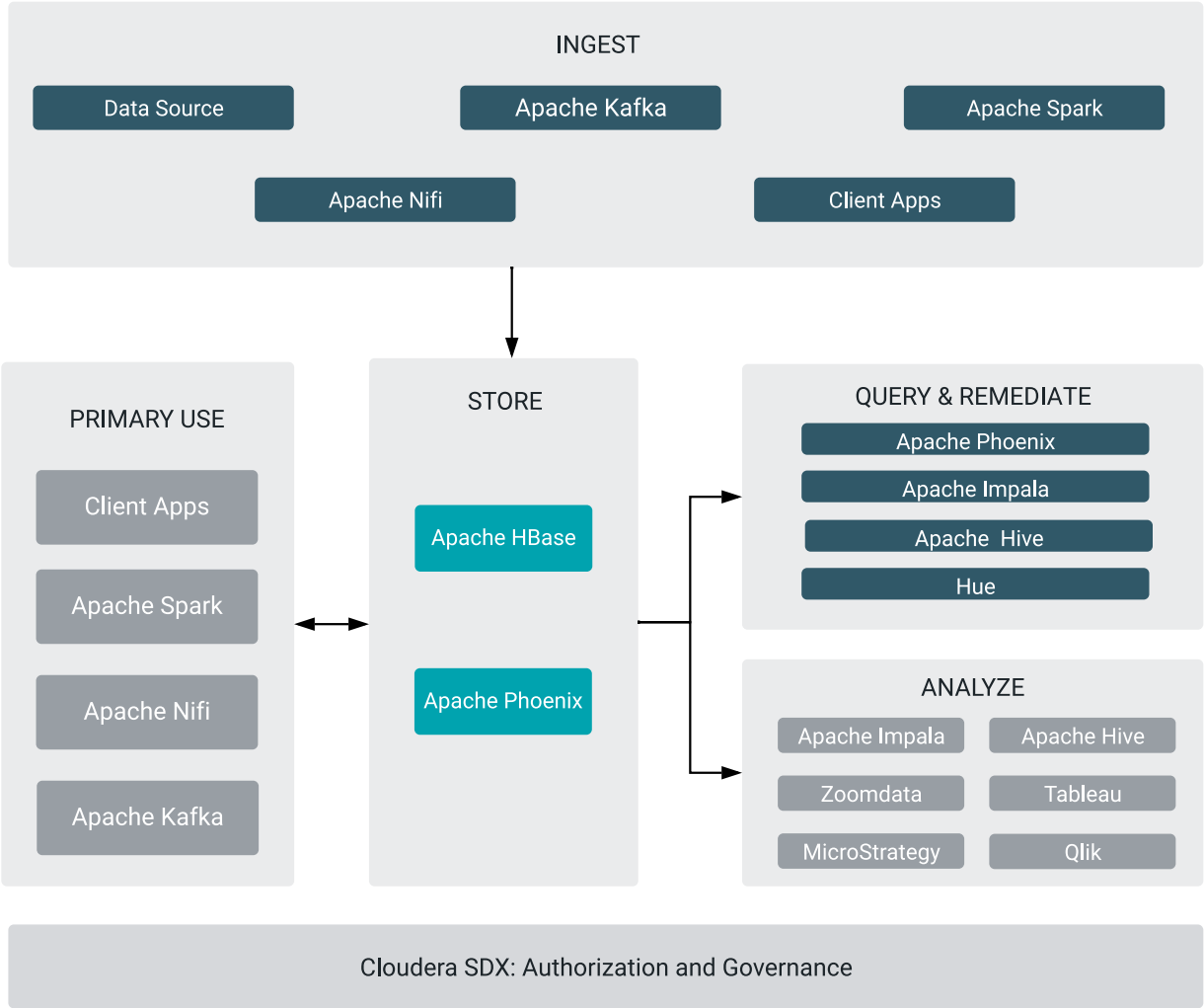

The operational database has the following components:

- *Apache Phoenix* provides a SQL interface that runs on top of Apache HBase.
- *Apache HBase* provides the key-value stores with massive scalability, so you can store unlimited amounts of data in a single platform and handle growing demands for serving data.
- *Apache ZooKeeper* provides a distributed configuration service, a synchronization service, and a naming registry.
- *Apache Knox Gateway* provides perimeter security so that the enterprise can confidently extend access to new users.
- *Apache HDFS* is used to write the Apache HBase WALs.
- *Hue* provides a web-based editor to create and browse Apache HBase tables.
- Object store such as Amazon S3 and Microsoft ADLS Gen2 is used to store the Apache HBase HFiles.
- Cloudera Shared Data Experience is used for security and governance capabilities. Security and governance policies are set once and applied across all data and workloads.
- *IDBroker* provides an authentication mechanism that is built as part of the Apache Knox authentication service. It allows an authenticated and authorized user to exchange a set of credentials or a token for cloud vendor access tokens.

Introduction to Apache HBase

Apache HBase is a scalable, distributed, column-oriented datastore. Apache HBase provides real-time read/write random access to very large datasets hosted on HDFS.

HBase is sparse, distributed, persistent, multidimensional sorted map, which is indexed by rowkey, column key,and timestamp. Fundamentally, it is a platform for storing and retrieving data with random access. HBase is designed to run on a cluster of computers, built using commodity hardware. HBase is also horizontally scalable, which means you can add any number of columns anytime.

Introduction to Apache Phoenix

Apache Phoenix is is a SQL layer for Apache HBase that provides a programmatic ANSI SQL interface.

Apache Phoenix implements best-practice optimizations to enable software engineers to develop HBase based next-generation applications that operationalize big data. Using Phoenix, you can create and interact with tables in the form of typical DDL/DML statements using the Phoenix standard JDBC API.

Apache HBase timestamps are the core part of the Apache Phoenix functionality. It is used as part of the *Travel in time* functionality and by the Transactional engines. Due to this, we expect that the data must have valid timestamps that signifies the time when the record is created. Invalid timestamps cause incorrect behavior for most Apache Phoenix functionalities, such as secondary indexes or CRUD operations. If your data is relying on the special behavior of the record's timestamps or future timestamps, you must not use such data with Apache Phoenix.

Apache Phoenix and SQL

You can use the Apache Phoenix SQL commands to create, drop, or modify an Apache HBase table. You can also create Apache Phoenix views that are virtual tables that share the same Apache HBase table.

The Apache Phoenix commands enable you to use standard SQL data definition language (DDL) and data manipulation language (DML) commands to interact with Apache HBase tables. You can create a new Apache HBase table using the standard CREATE TABLE SQL command or create a view on an existing Apache HBase table using the VIEW command. View enables you to have multiple virtual tables that share the same physical Apache HBase table.

Apache HBase tables and Apache Phoenix tables have a one-to-one relationship. This means each Apache HBase table is associated with a corresponding Apache Phoenix table.

Warning: Although you can modify Apache Phoenix tables using the Apache HBase native APIs, this is not supported and results in errors, inconsistent indexes, incorrect query results, and sometimes corrupt data.

Using Apache Phoenix commands you can do the following tasks:

- Create or alter Apache HBase tables using DDL commands like CREATE TABLE
- Modify contents in an Apache HBase table using DML commands like UPSERT VALUES

Cloudera Runtime 7.2.11

Using Apache Phoenix to Store and Access Data

Date published: 2020-02-29
Date modified: 2024-01-24

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Mapping Apache Phoenix schemas to Apache HBase namespaces..... 4
Enable namespace mapping..... 4

Associating tables of a schema to a namespace.....5
Associate table in a customized Kerberos environment..... 5
Associate a table in a non-customized environment without Kerberos.....6

Using secondary indexing in Apache Phoenix.....6
Use strongly consistent indexing with Apache Phoenix.....7
Migrate existing tables to use strongly consistent indexing..... 8

Using transactions in Apache Phoenix..... 9
Configure transaction support..... 9
Create and modify tables with transaction support.....11

Using JDBC API with Apache Phoenix..... 13
Connecting to Apache Phoenix Query Server using the JDBC client.....14
Connect to Phoenix Query Server.....14
Connect to Phoenix Query Server through Apache Knox.....15
Launching Apache Phoenix Thin Client.....16

Using the Phoenix JDBC Driver.....17
Configuring the Phoenix classpath.....17
Adding the Phoenix JDBC driver jar.....17
Adding the HBase or Hadoop configuration files..... 18
Understanding the Phoenix JDBC URL..... 18

Using non-JDBC drivers with Apache Phoenix..... 20

Using Apache Phoenix-Spark connector.....21
Configure Phoenix-Spark connector using Cloudera Manager.....21
Phoenix-Spark connector usage examples..... 22

Using Apache Phoenix-Hive connector..... 24
Configure Phoenix-Hive connector using Cloudera Manager..... 25
Apache Phoenix-Hive usage examples..... 25
Limitations of Phoenix-Hive connector..... 27

Mapping Apache Phoenix schemas to Apache HBase namespaces

You can map a Apache Phoenix schema to an Apache HBase namespace to gain multitenancy features in Apache Phoenix.

Important: You must configure this feature only in a Cloudera Private Cloud Base deployment. This feature is configured automatically in a Cloudera Public Cloud deployment.

Apache HBase namespaces are a logical grouping of tables, similar to a database in a relational database system. Apache HBase, the underlying storage engine for Apache Phoenix, has namespaces to support multi-tenancy features. Multitenancy helps an Apache HBase user or administrator to perform access control and quota management tasks. Namespaces enable tighter control of where a particular data set is stored on the Apache HBase RegionsServers.

Enable namespace mapping

You can enable namespace mapping by configuring a set of properties using Cloudera Manager.

About this task

After you set the properties to enable the mapping of Phoenix schemas to HBase namespaces, reverting the property settings renders the Phoenix database unusable. Test or carefully plan the Phoenix to HBase namespace mappings before implementing them.

Important: Cloudera recommends that you enable namespace mapping. If you decide not to enable this feature, you can skip the following steps.

To enable Phoenix schema mapping to a non-default HBase namespace:

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope (Service-Wide) .
4. Locate the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the Search box.
5. Add the following property values:

   Name: phoenix.schema.isNamespaceMappingEnabled

   Description: Enables mapping of tables of a Phoenix schema to a non-default HBase namespace. To enable mapping of a schema to a non-default namespace, set the value of this property to true. The default setting for this property is false.

   Value: true

   Name: phoenix.schema.mapSystemTablesToNamespace

   Description: With true setting (default): After namespace mapping is enabled with the other property, all system tables, if any, are migrated to a namespace called system. With false setting: System tables are associated with the default namespace.

   Value: true
6. Select Scope Gateway .

7. Locate the HBase Client Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the Search box.
8. Add the following property values:

   Name: phoenix.schema.isNamespaceMappingEnabled

   Description: Enables mapping of tables of a Phoenix schema to a non-default HBase namespace. To enable mapping of the schema to a non-default namespace, set the value of this property to true. The default setting for this property is false.

   Value: true

   Name: phoenix.schema.mapSystemTablesToNamespace

   Description: With true setting (default): After namespace mapping is enabled with the other property, all system tables, if any, are migrated to a namespace called system.With false setting: System tables are associated with the default namespace.

   Value: true
9. Enter a Reason for change, and then click Save Changes to commit the changes.
10. Restart the role and service when Cloudera Manager prompts you to restart.

Note: If you do not want to map Phoenix system tables to namespaces because of compatibility issues with your current applications, set the phoenix.schema.mapSystemTablesToNamespace property to false.

Associating tables of a schema to a namespace

Learn how to associate tables of a Phoenix schema.

Important: You must use this feature only in a Cloudera Private Cloud Base deployment. This feature is configured automatically in a Cloudera Public Cloud deployment.

After you enable namespace mapping on a Phoenix schema that already has tables, you can migrate the tables to an HBase namespace. The namespace directory that contains the migrated tables inherits the schema name.

For example, if the schema name is store1, then the full path to the namespace is $hbase.rootdir/data/store1. System tables are migrated to the namespace automatically during the first connection after enabling namespace properties.

Associate table in a customized Kerberos environment

You can run a command to associate a table in a customized environment without Kerberos.

Before you begin

In a Kerberos-secured environment, you must have admin privileges (user hbase) to complete the following task.

Procedure

- Run a command to migrate a table of a schema to a namespace, using the following command syntax for the options that apply to your environment:

```
phoenix-psql
ZooKeeper_hostnames:2181
:zookeeper.znode.parent
:principal_name
:HBase_headless_keytab_location
;TenantId=tenant_Id
;CurrentSCN=current_SCN
-m
```

```
schema_name.table_name
```

Associate a table in a non-customized environment without Kerberos

You can run a command to associate a table in a non-customized environment without Kerberos.

Procedure

- Run the following command to associate a table:

```
phoenix-psql ZooKeeper_hostname -m Schema_name.table_name
```

Using secondary indexing in Apache Phoenix

Apache Phoenix uses a secondary index to serve queries. An index table is an Apache Phoenix table that stores the reference copy of some or all the data in the main table.

You can use a secondary index to access data from its primary data access path. When you use a secondary index, the indexed column qualifiers or rows form a unique row key that allows you to do point lookups and range scans.

Secondary index types

Apache Phoenix supports global and local secondary indexes. Global indexes is used for all typical use cases. You can use local indexes for specific use cases where you want the primary and the index table to be present in the same Apache HBase region.

- Use global indexes for read-heavy use cases. Use the covered-global index to save on read-time overheads. Global indexes are used to co-locate related information.
- Use local indexes for write-heavy use cases. Use the functional-local index on arbitrary expressions to query specific combinations of index queries. A local index is an in-partition index that is optimized for writes but requires more data to be read to answer a query.

Note: If your table is large, you can use the ASYNC keyword with CREATE INDEX to create an index asynchronously. ASYNC uses a MapReduce job to distribute index workload, and every single mapper works with each data table region and writes to the index region. Therefore freeing up critical resources during indexing.

The following tables list the index type and index scope with a description and example for each index type:

Table 1:

Index type	Description
Covered	Include the data that you want to access from the primary table in the index rows. The query does not have to access the primary table once the index entry is found. Benefits: Save read-time overhead by only accessing the index entry. In the following example, column v3 is included in the index to avoid the query to access the primary table to retrieve this information. Example The following command creates indexes on the v1 and v2 columns and include the v3 column as well: `CREATE INDEX my_index ON exp_table ( v1,v2) INCLUDE(v3);`

Index type	Description

Functional	Create an index on arbitrary expressions. When your query uses the expression, the index is used to retrieve the results instead of the data table.
	Benefits: Useful for certain combinations of index queries.
	Example
	Run the following command to create a functional index so that you can perform case insensitive searches on the combined first name and last name of a person:
	`CREATE INDEX UPPER_NAME ON EMP (UPPER(FIRST_NAME||' '||LAST_NAME));`
	Search on the combined first name and last name using the following command:
	`SELECT EMP_ID FROM EMP WHERE UPPER(FIRST_NAME||' '||LAST_NAME)='EXP_NAME';`

Table 2:

Index scope	Description
Global	You can use this when you have read-heavy use cases. Each global index is stored in its own table, and therefore it is not co-located with the data table. A Global index is a covered index. It is used for queries only when all columns in that query are included in that index. Example Run the following command to create a global index: `CREATE INDEX my_index ON exp_table (v1);`
Local	You can use this for write-heavy use cases. Each local index is stored within the data table. Example Run the following command to create a local index: `CREATE LOCAL INDEX my_index ON exp_table (v1);`

Use strongly consistent indexing with Apache Phoenix

Strongly consistent indexing is a three-phase indexing algorithm that ensures that the index table is always in sync with the data table.

Strongly consistent indexing ensures that the data read from the index table is consistent at read time for read operations. An index table row is repaired using the corresponding data table row if necessary at read time. When the writes to a table are fast, an additional write phase is used without impacting the write performance.

The following table illustrates the design of the three-phase indexing:

Table 3:

Operation	Strongly consistent index
READ	**1.** Reads the index rows and checks its status **2.** Repairs unverified rows using the underlying data table
WRITE	**1.** Sets the status of the existing index rows as unverified and writes the new index rows with status as unverified **2.** Writes to the underlying data table rows **3.** Deletes the exisiting index rows and sets the status oif the new rows as verified
DELETE	**1.** Sets the status of index table rows as unverified **2.** Deletes the underlying data table rows **3.** Deletes the index table rows

Newly created Apache Phoenix tables in Cloudera Runtime 7.1.6 and higher use strongly consistent indexing by default. You can use the index upgrade tool to upgrade your Apache Phoenix tables to use the strongly consistent indexing.

Migrate existing tables to use strongly consistent indexing

You must use the index upgrade tool to rebuild your indexes. After you run this tool, your existing tables use the strongly consistent global index.

About this task

If you have Apache Phoenix tables that you migrated from Cloudera Runtime 7.1.5 or lower to Cloudera Runtime 7.1.6 or higher, you must convert the tables to use the strongly consistent index using the index upgrade tool.

Procedure

1. Perform the following as the HBase user, or as a user with HBase root privileges (global RXWCA).
2. Run the following command for each table that has global indexes.

```
hbase org.apache.phoenix.mapreduce.index.IndexUpgradeTool -o [***UPGRADE /
 ROLLBACK***] -tb [***TABLE NAME***]
-lf [***/TMP/INDEX-UPGRADE-TOOL.LOG***]
```

Parameters:

- -o: Type whether you want to upgrade or rollback.
- -tb: Type the table name that you want to upgrade or rollback.
- -lf: Type the path where you want to store the *index upgrade tool* log file.

3. After the conversion is finished, it is highly recommended to rebuild the indexes. Use the following commands for each index.

```
hbase org.apache.phoenix.mapreduce.index.IndexTool -s [***SCHEMA NAME***]
 -dt [***DATA TABLE NAME***] -it[***INDEX NAME***] -op /tmp/phoenix_index
```

```
hbase org.apache.phoenix.mapreduce.index.IndexTool -fi -s [***SCHEMA
 NAME***] -dt [***DATA TABLE NAME***] -it[***INDEX NAME***] -op /tmp/ph
oenix_index
```

Parameters:

- -s: Type the schema name. Optional for default schema.
- -dt: Type the data table name.
- -it: Type the index name.
- -op: Type the path where you want to store the *index upgrade tool* log file.

Note: These commands start the MapReduce jobs in the background, which might take a longer time and cause significant load on the cluster.

Using transactions in Apache Phoenix

X-row, X-table transaction support enables support for complex distributed transactions. Transaction support enables you to run atomic database operations, meaning your database operations must either be completed or terminated. A transaction may contain a set of database operations where each one is carried out atomically without interfering with others.

You can perform complex transactions that span multiple rows and tables and implement traditional star schema, wide-columns, or both. Transactions are lock-free and handle any write-write conflicts for you without your intervention.

This two-phase commit capability is implemented using ANSI SQL semantics which is the same as used in other databases like MySQL or PostgreSQL, but this implementation benefits from the near-linear scalability of Apache HBase and Apache Phoenix, and do not have upper-bound on the number of transactions, size of rows/table, or size per table like other databases.

Note: You cannot convert a transactional table back to a non-transactional table. Ensure that you only convert those tables that you know will use transactions.

Configure transaction support

Before you can start using transactions, you must first add the Apache Omid service and configure Apache Phoenix to enable transactions. Apache Omid provides the transaction management capability in Apache Phoenix.

Before you begin

- If your cluster is Kerberized, run the kinit command as hbase user before adding the Apache Omid service to the cluster. Ensure you have valid Kerberos credentials. You can list the Kerberos credentials using the klist command.

  or

  If you do not want to kinit as a hbase user, you can use Apache Ranger in the Cloudera Shared Data Experience Data Lake to grant the user permissions to CREATE, READ and WRITE on the commit table (default table name OMID_COMMIT_TABLE) and timestamp table (default table name OMID_TIMESTAMP_TABLE).

  Apache Omid creates and writes to OMID_COMMIT_TABLE and OMID_TIMESTAMP_TABLE during a transaction.
- Ensure that the Apache Phoenix client doing transactions should have minimum WRITE permissions on commit table (default table name OMID_COMMIT_TABLE).

About this task

Add the Apache Omid service, and configure Apache Phoenix using Cloudera Manager.

Procedure

1. On the Home Status tab, click to the right of the cluster name and select Add a Service. A list of service types display. You can add one type of service at a time.
2. Select Omid and click Continue.
3. Click Finish to complete adding the Omid service. Return to the Cloudera Manager home page.

   The Phoenix OMID TSO service listens to the 54758 port by default, which is in the ephemeral port range. Because of this, the service might randomly fail to start if the port is already used.

   To avoid this, change the TSO port to 24758. Follow these steps to change the default port.

   a) Click on the Omid service.
   b) Go to Configuration and search for OMID TSO server port.
   c) Modify the port to 24758.
   d) Enter a Reason for change, and then click Save Changes.
   e) Click (Stale configuration).
   f) Restart the Omid service when Cloudera Manager prompts you to restart.

   Note: You need to change the port manually in all the Omid configuration files that are not managed by Cloudera Manager.

4. Verify the new service is started properly by checking the health status of the new service. If the Health Status is Good, then the service started properly.

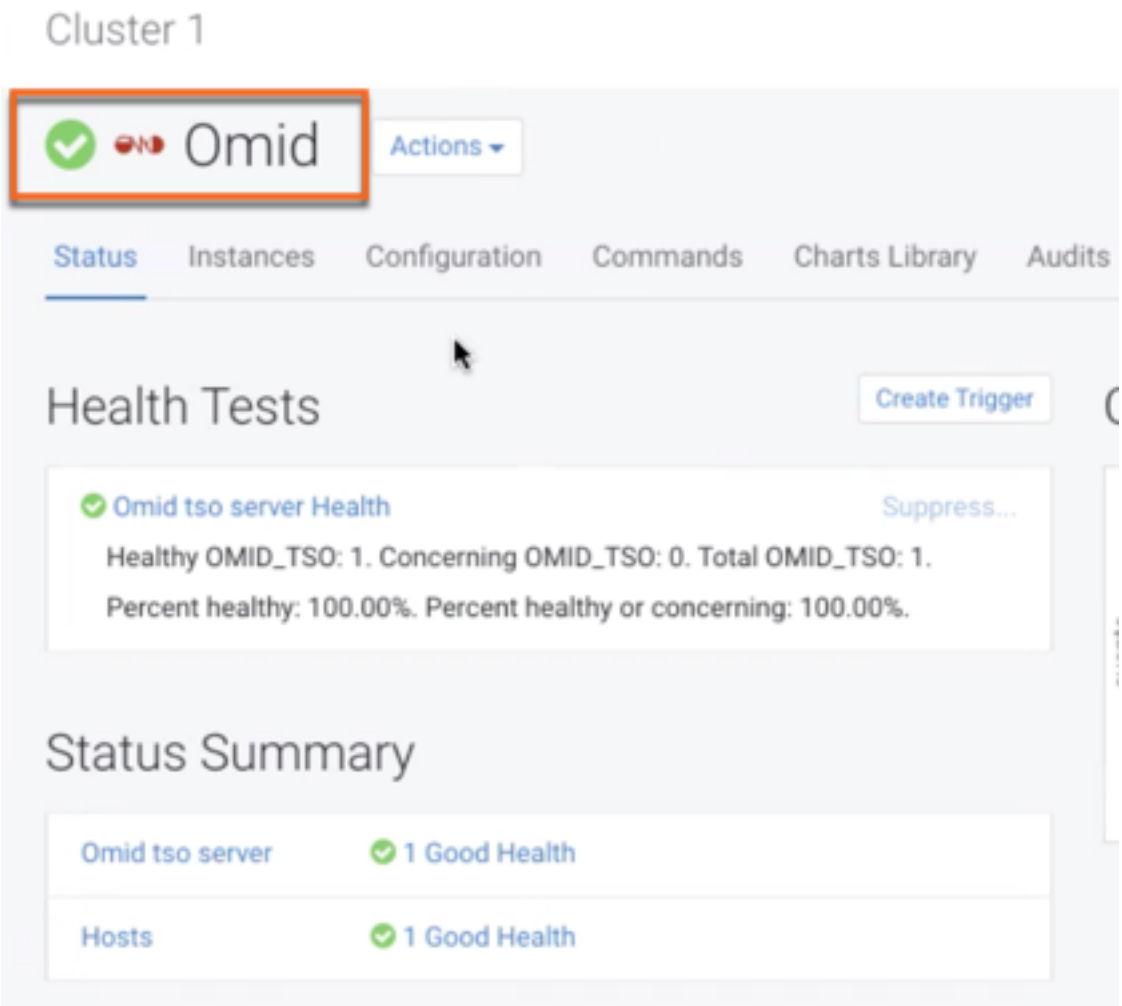

5. After the Omid service has started, go to the HBase service.
6. Click the Configuration tab.
7. Select Scope > Gateway.
8. Select Category > Advanced.
9. Locate the HBase Client Advanced Configuration Snippet property or search for HBase Client Advanced Configuration Snippet (Safety Valve) for hbase-site.xml:

   Click View as XML and add the following property value:

```
<property>
  <name>phoenix.transactions.enabled</name>
  <value>true</value>
</property>
```

10. Enter a Reason for change, and then click Save Changes to commit the changes.
11. Restart the role and service when Cloudera Manager prompts you to restart.

Create and modify tables with transaction support

You can create new tables with transaction support enabled or alter an existing table to enable transaction support on that table.

About this task

To enable transaction support, you must use the TRANSACTIONAL=true parameter when creating a table, followed by the COMMIT command at the end of your transaction.

Procedure

- Create a table with transaction support enabled.

```
CREATE TABLE NEW_TABLE (ID INTEGER PRIMARY KEY, v1 varchar) TRANSACTIONA
L=true, TRANSACTION_PROVIDER='OMID';
```

> **Note:** TRANSACTION_PROVIDER is an optional parameter. Omid is used as the default transaction provider if this parameter is not specified.

- Set autocommit to off before you start a transaction.

  The following example shows you how to turn off autocommit and then upsert a value into NEW_TABLE.

```
jdbc:phoenix:> !autocommit off
Autocommit status: false
jdbc:phoenix:> upsert into NEW_TABLE values(1,'ph');
1 row affected (0.015 seconds)
jdbc:phoenix:> select * from NEW_TABLE;
+---+----+
| A | B  |
+---+----+
| 1 | ph |
+---+----+
1 row selected (0.144 seconds)
jdbc:phoenix:> upsert into NEW_TABLE values(2,'ph');
1 row affected (0.001 seconds)
jdbc:phoenix:> !commit
```

- Modify a table to enable transactions on the table.

```
ALTER TABLE EXISTING_TABLE SET TRANSACTIONAL=true;
```

Example

The following example shows you how to do a transaction on a table when transaction support is enabled:

```
!autocommit off -- Turn off autocommit
SELECT * FROM example_table; -- Transaction starts here. Other transactions
 cannot see your updates yet
UPSERT INTO example_table VALUES (1,'A');
SELECT count(*) FROM example_table WHERE e=1; -- You can see an uncommited
row
DELETE FROM my_other_table WHERE e=2;
!commit -- Other transactions can now see the updates
```

You will see an exception if you try to commit a row that is conflicting with another transaction. In the following example, you will see an exception when you try and commit the first transaction after committing the second transaction because the second transaction updates the same row.

```
UPSERT INTO NEW_TABLE VALUES (1,'a');
```

Now, when you do a commit on the same row. Row 1 in the following example, you will see an error.

```
UPSERT INTO NEW_TABLE VALUES (1,'b');
!commit
```

You see an exception similar to the following:

```
java.sql.SQLException: ERROR 523 (42900): Transaction aborted due to conflic
t with other mutations. Conflict detected for transaction 155612256680000000
0.
```

Using JDBC API with Apache Phoenix

You can create and interact with Apache HBase tables using Phoenix DDL/DML statements through its standard JDBC API. Apache Phoenix JDBC driver can be easily embedded in any application that supports JDBC.

Apache Phoenix enables you to use the standard JDBC API to create and access Apache HBase tables. You can use JDBC APIs with Apache Phoenix instead of native Apache HBase client APIs to create tables, insert, and query data.

Apache Phoenix tables have a 1:1 relationship with Apache HBase tables. You can choose to create a new table using an Apache Phoenix DDL statement such as CREATE TABLE, or create a view on an existing Apache HBase table using the VIEW statement.

Important: Modifying an Apache Phoenix table using Apache HBase native API is not supported. Doing this leads to errors, inconsistent indexes, incorrect query results, or sometimes to corrupt data.

To use the Apache Phoenix JDBC driver, you must embed the driver in your application that supports JDBC. Apache Phoenix has two kinds of JDBC drivers.

- A *thick driver* communicates directly with Apache HBase. The thick driver, therefore, needs access to all the nodes in the Apache HBase cluster.
- A *thin driver* communicates with Apache HBase through Phoenix Query Server (PQS) and requires access only to PQS. Use the thin driver to connect to PQS through Apache Knox or connect to PQS directly.

In an Cloudera Operational Database Cloudera Data Hub cluster, Data Lake (Cloudera Shared Data Experience cluster) provides security dependencies such as Apache Knox. Your JDBC URL string would depend on whether you want to connect directly or through Apache Knox. Before you try connecting to Apache Phoenix, check if you are in the list of allowed users in Apache Ranger allowed to access Apache Phoenix and Apache HBase.

Based on whether you want to use the thick or thin driver, you need the JAR files for the Apache HBase client, the Apache Phoenix client, and the PQS client.

For the thick driver, you need:

- hbase-client-[****VERSION****].jar
- hbase-site.xml

Note: You must add the cluster's current hbase-site.xml to the application classpath. You can get the hbase-site.xml by doing an SSH to the cluster node with the hbase-gateway role. You can copy the hbase-site.xml file from the following path /etc/hbase/hbase-site.xml or /etc/hbase/conf/hbase-site.xml.

For the thin driver, you need:

- phoenix-queryserver-client-[****VERSION****].jar

You can get these JAR files from the following locations:

- Go to /opt/cloudera/parcels/CDH/lib/phoenix/ on an operational database cluster node with the phoenix-gateway role

  or
- Download the JAR files from the Cloudera Repository

When using the thin driver, your applications interact with the Phoenix Query Server using the Avatica API and Google Protocol Buffers serialization format.

JDBC driver location

Use the /opt/cloudera/parcels/CDH/lib/phoenix/[***PHOENIX VERSION***].jar file present in your deployment location. For example, /opt/cloudera/parcels/CDH/lib/phoenix/phoenix-5.0.0.7.2.0.0-128-client.jar

URL syntax for the thick JDBC driver

To connect to Apache Phoenix using the thick JDBC driver, you must use the following JDBC URL syntax:

```
jdbc:phoenix:[***ZOOKEEPER_QUORUM***]:[***ZOOKEEPER_PORT***]:
[***ZOOKEEPER_HBASE_PATH***]
```

The zookeeper_quorum and zookeeper_port parameters are optional if you have added the operational database Apache HBase cluster's current hbase-site.xml to the application classpath.

Apart from the JDBC driver, the following drivers are supported:

- ODBC driver
- Python driver for Phoenix

Connecting to Apache Phoenix Query Server using the JDBC client

You can interact with Apache Phoenix using your client and Apache Phoenix Query Server (PQS).

PQS is automatically configured when you create an Operational Database Data Hub cluster. There are two ways in which you can use the thin client to interact with Phoenix:

- Connect to PQS directly
- Connect to PQS using the Apache Knox gateway

Connect to Phoenix Query Server

You can connect to Phoenix Query Server (PQS) using the JDBC thin client without using a gateway such as Apache Knox. You must use the JDBC URL syntax to form the URL and connect to PQS.

Before you begin

Ensure that you have access to Apache Phoenix and Apache HBase, and you have the required permissions configured in Ranger to connect to PQS.

Ensure that you have safely noted down your Kerberos principal and keytab information used to connect to PQS.

About this task

You can use the JDBC URL syntax to form the URL and connect to PQS.

Note: When you execute the first query on the PQS, the system creates the Phoenix system tables. In some cases, this table creation process might not finish within the default Avatica timeout period. As a result of this issue, you might experience a timeout, and the query might fail while attempting to execute the initial query through PQS. In this case, rerun the PQS query.

Procedure

- To connect to the PQS directly, you must use the JDBC URL syntax as shown here: jdbc:phoenix:thin:[key=value [;key=value...]]

  You must provide the correct URL, serialization, and authentication key-values to interact with the Phoenix Query Server. For more information about optional parameters, see Client Reference.

Example

```
jdbc:phoenix:thin:url=http://[PHOENIX_QUERY_SERVER_FQDN]:8765;serialization=
PROTOBUF; authentication=SPNEGO;
principal=[***PRINCIPAL@EXAMPLE.COM***];keytab=[***PATH TO THE KEYTAB
 FILE***]
```

Connect to Phoenix Query Server through Apache Knox

You can connect to Phoenix Query Server (PQS) using the JDBC thin client through the Apache Knox gateway. Apache Knox requires your thin client connection to be over HTTPS.

Before you begin

Ensure that you have access to Apache Phoenix and Apache HBase, and you have the required permissions configured in Ranger to connect to PQS.

Procedure

- Get the PQS Knox endpoint to connect to PQS from the Cloudera Data Hub user interface. Go to Data Hub Clusters Endpoints Cloudera Manager Info Endpoints Phoenix Query Server .

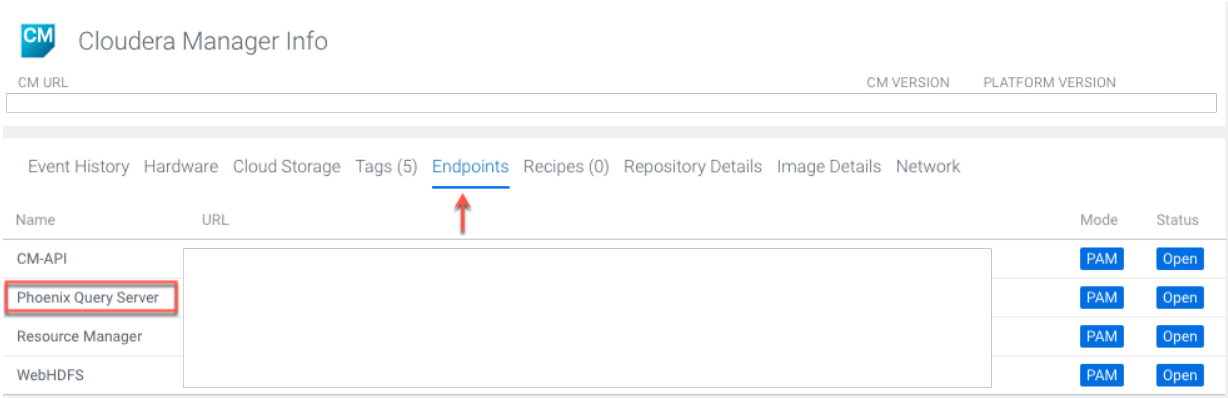

- Use the JDBC URL in the following sytax to connect to PQS through the Apache Knox gateway:

  jdbc:phoenix:thin:url=https://[***KNOX ENDPOINT***]:[***PORT***]/[***KNOX TOPOLOGY NAME***]/cdp-proxy-api/avatica/;serialization=PROTOBUF;authentication=BASIC;avatica_user=[***WORKLOAD USERNAME*];avatica_password=[***WORKLOAD PASSWORD***];truststore=[***PATH TO THE KNOX TRUSTSTORE .JKS FILE***];truststore_password=[***TRUSTSTORE PASSWORD***]

  The standard Oracle Java JDK distribution includes a default truststore (cacerts) that contains root certificates for many well-known CAs, including Symantec. Rather than using the default truststore, Cloudera recommends using the alternative truststore, jssecacerts. The alternative truststore is created by copying cacerts to that filename (jssecacerts). Certificates can be added to this truststore when needed for additional roles or services. This alternative truststore is loaded by Hadoop daemons at startup. Password (if there is one for the truststore) stored in a plaintext file readable by all (OS filesystem permissions set to 0440). For more information about truststores, see Understanding Keystores and Truststores.

  Note: The user name and password are for the Apache Knox gateway, and the authentication must always be set to BASIC. The truststore and truststore_password the Knox public certificate.

Example

```
jdbc:phoenix:thin:url=https://ccycloud-1.ni-717x-cj.root.comops.site:8443/ga
teway/cdp-proxy-api/avatica:8765;serialization=PROTOBUF;
authentication=BASIC;avatica_user=joe
```

```
;avatica_password=joe_password
;truststore=/home/path/truststore.jks;truststore_password=the_truststore_p
asswords
```

Launching Apache Phoenix Thin Client

You need to launch the Apache Phoenix thin client before connecting to the Phoenix Query Server (PQS). Learn to use the command that launches the Apache Phoenix thin client.

Procedure

1. Log in to the terminal.
2. Run the following command to launch the Apache Phoenix thin client.

   `phoenix-sqlline-thin` *<URL>* *<SQLFILE>* *<OPTIONAL_ARGUMENTS>*

   - *URL*: The URL to the Phoenix Query Server.
   - *SQLFILE*: A file of the SQL commands to execute.

Optional arguments	Description
-a (--authentication)	Mechanism for HTTP authentication. Possible values: • SPNEGO • BASIC • DIGEST • NONE The default value is NONE.
-s (--serialization)	Serialization type for HTTP API. Possible values: • PROTOBUF • JSON (Must match server setting)
-au (--auth-user)	Username for HTTP authentication.
-ap (--auth-password)	Password for HTTP authentication.
-p (--pricipal)	Kerberos principal for SPNEGO authentication from keytab.
-kt (--keytab)	Kerberos keytab file for SPNEGO authentication from keytab.
-t (--truststore)	Truststore file that contains the TLS certificate of the server.
-tp (--truststore-password)	Password of the truststore file that contains the TLS certificate of the server.
-v (--verbose)	Verbosity setting for sqlline.
-c (--color)	Color setting for sqlline.

Optional arguments	Description
-fc (--fastconnect)	Do not fetch all schemas on initial connection (for sqlline).

Use the following command to connect to an unsecured server.

```
phoenix-sqlline-thin http://unsecure.server:8765
```

Use the following command to connect to a secure Kerberized server.

```
phoenix-sqlline-thin https://secure.server:8765 -a SPNEGO --principal user
 -kt /path/to/user.kt - t /path/to/truststore -kt trustsore_passsord
```

Use the following command to connect through Knox with BASIC authentication.

```
phoenix-sqlline-thin https://konx.host:8765/avatica -a BASIC --au knox_u
ser user -ap konx_password - t /path/to/truststore -kt trustsore_passsord
```

Using the Phoenix JDBC Driver

Learn how to use the Phoenix thick client. The Phoenix JDBC Driver enables you to connect directly with your Phoenix data from any applications supporting JDBC connectivity. The Phoenix JDBC driver involves two basic aspects, the Phoenix classpath and the Phoenix JDBC URL.

The thin client for Phoenix Query Server is described on the Apache site. For more information, see Phoenix Query Server.

Related Information

Phoenix Query Server

Configuring the Phoenix classpath

Learn how to configure the Phoenix classpath. The classpath is the location where the JVM looks for the user-defined classes, packages, and resources for the Java programs.

To use Phoenix, both the JDBC driver jar and the hbase-site.xml must be added to the application classpath.

Adding the Phoenix JDBC driver jar

Learn how to add the Phoenix JDBC driver jar to the classpath.

About this task

The Phoenix JDBC client is built on top of the HBase client and has an unusually high number of dependencies. To make this manageable, Phoenix provides a single shaded uberjar that can be added to the classpath.

Phoenix uses some private and semi-public HBase APIs, which may change between HBase versions, and provides separate binary distributions for use with different HBase versions.

Before you begin

You must consider the driver JARs from the binary distribution (Phoenix downloads) or maven artifact corresponding to the HBase version on the cluster. See *Binary distribution: Phoenix downloads*.

Procedure

1. Download the driver JAR from the binary distribution location.

   Copy the corresponding phoenix-client-embedded-hbase-[hbase.profile]-[phoenix.version].jar to the application classpath.
2. Add the dependency through Maven.

```
<dependencies>
    ...
    <dependency>
        <groupId>org.apache.phoenix</groupId>
        <artifactId>phoenix-client-embedded-hbase-[hbase.profile]</artifac
tId>
        <version>[phoenix.version]</version>
    </dependency>
    ...
  </dependencies>
```

Related Information

Binary distribution: Phoenix Downloads

Adding the HBase or Hadoop configuration files

Learn how to add the HBase or Hadoop configuration files to the classpath.

About this task

As Phoenix is built on top of the HBase client, it needs the HBase configuration files for the right operation. For some configurations, it may also need other Hadoop or HDFS configuration files like core-site.xml.

Procedure

1. Download the hbase-site.xml from the cluster, and copy it to a directory on the classpath.

   The hbase-site.xml usually resides in /etc/hbase/conf on the client side.

   Important: Add the directory containing hbase-site.xml, and not the complete path of hbase-site.xml, to the classpath.
2. Pack hbase-site.xml into the root directory of a JAR file, and add that to the classpath.

   If hbase-site.xml changes on the cluster, ensure to copy the latest one to the application classpath.

   For some development clusters, that use the default configuration, Phoenix might work without this; however, ensure that the hbase-site.xml on the classpath is correct.

Understanding the Phoenix JDBC URL

Know the usage and different parts of the Phoenix JDBC URL.

The Phoenix URL contains two main parts. The first one describes the connection to the HBase cluster, while the second part can specify additional options for Phoenix.

jdbc:<protocol variant>[:<server list>[:<port>[:<zk root node>[:<principal>[:<keytab file>]]]]][;<option>=<value>]*

For example,

```
jdbc : phoenix:cod-1ijq6ne9nicgn-leader0.cod-7218.xcu2-8y8x.dev.cldr.work,co
d-1ijq6ne9nicgn-master0.cod-7218.xcu2-8y8x.dev.cldr.work,cod-1ijq6ne9nicgn-m
aster1.cod-7218.xcu2-8y8x.dev.cldr.work:2181:/hbase
```

- *PROTOCOL VARIANT*: The HBase connection registry to use.
- *SERVER LIST*: A comma-separated list of hostnames or IPv4 addresses. It is also possible to specify the port in ZK quorum string for each host. For more information, see *HBASE-12706*. In this case the colon ':' characters must be escaped with a backslash '\'. It may be necessary to repeat the escape character, for example in the Java source code.
- *PORT*: An integer number specifying a port. Ports specified in the server list take precedence.
- *ZK ROOT NODE*: The root ZNode for the HBase instance. Must be empty for non-ZK registries.
- *PRINCIPAL*: The Kerberos principal to use for authentication.

  If only the principal is specified, then this defines the user name with each distinct user having their own dedicated HBase connection (HConnection). This provides a means of having multiple, different connections, each with different configuration properties on the same JVM.
- *KEYTAB*: The Kerberos keytab to be used for authentication. Must be specified together with the principal.
- *OPTION*: A connection option. This can be any additional JDBC parameter. This is optional.
- *VALUE*: Value for the connection option.

Parameters from the end of the connection definition can be omitted. Use empty strings for missing parameters in the middle of the URL. For example, the jdbc:phoenix::::principal:/home/user/keytab URL can be used to specify the Kerberos principal and keytab while using the default connection specified in the hbase-site.xml file.

The default connection

This section highlights the client configurations and dependencies connecting to an HBase cluster.

The underlying HBase client identifies the HBase cluster based on parameters specified in the hbase-site.xml file. While Phoenix makes it possible to override this, in the majority of cases it is advisable not to do that, and to use the cluster connection definition from the hbase-site.xml file. You must specify the connection directly only when switching between identically configured HBase instances, for example, a production and a disaster recovery cluster.

To use the defaults from the hbase-site.xml file, use the jdbc:phoenix URL or jdbc:phoenix;option=value if additional options are needed.

For more information, see *HBase documentation on Client configuration and dependencies*.

The jdbc:phoenix: protocol variant

If the default protocol variant is specified, then Phoenix selects the variant based on the value of the hbase.client.registry.impl property.

If the hbase.client.registry.impl property is not defined, then it chooses a default value based on the version of the HBase client it includes.

The jdbc:phoenix+zk: protocol variant

This uses the original Zookeeper-based HBase connection registry, which is always available in HBase. The server list and port specify the ZooKeeper (ZK) quorum. For more information, see *HBASE-12706*, this is supported now. In this case the colon ':' characters must be escaped with a backslash ''.

Following are a few examples:

- jdbc:phoenix+zk:localhost:2181:/hbase:principal:keytab: Fully specified.
- jdbc:phoenix+zk:host1\:2181,host1\:2182,host2\:2183 : Heterogeneous ports, default ZooKeeper node.
- jdbc:phoenix+zk : Use the default ZooKeeper parameters from the hbase-site.xml file or the HBase defaults. (using jdbc:phoenix is preferred in most scenarios).

The jdbc:phoenix+master: protocol variant

This uses the master-based connection registry that is added by providing an option for clients to find the server hosting META that does not involve the ZooKeeper client, and is available from HBase 2.3.0. The "zk root node" parameter must never be specified. For more information, see *HBASE-18095*.

Following are a few examples:

- jbdc:phoenix+master:master1\:16001,master2\:16002::principal:/path/to/keytab : Fully specified.
- jbdc:phoenix+master:master1,master2 : Uses master1 and master2 with the default port.

Note:

- The support for master and RPC registries is only available in Phoenix 5.1.4 and later and 5.2.0 and later. The earlier version only supports the jdbc:phoenix: protocol variant implementing the original HBase ZooKeeper Connection Registry.
- The support for the registry variants is only available for the HBase versions that support them. If an unsupported HBase client version variant is specified, Phoenix throws an error.

Related Information

HBase client configuration and dependencies
HBASE-12706
HBASE-18095

Using non-JDBC drivers with Apache Phoenix

Based on your application development requirements, you can obtain one of the other non-JDBC drivers.

The list of supported programming languages to access data stored in your operational database:

- Apache Groovy
- C
- C++
- Go
- Java
- Jython
- Python
- PHP
- Scala

You can also use REST for API support.

The following drivers are supported:

ODBC driver

Use the Database Open Database Connectivity (ODBC) interface to access the operational database. The ODBC driver is provided by Cloudera as an additional download, but you can also use ODBC drivers from third-party providers.

You can download the Phoenix ODBC Driver from here: Phoenix ODBC Connector for Cloudera Operational Database.

Note

You must have a Cloudera Enterprise Support Subscription to download the ODBC driver.

Python driver for Phoenix

Download the Python Driver for Apache Phoenix from the Apache Phoenix website. For more information, see Python Driver for Phoenix.

Other drivers that are not supported by Cloudera

You can use other non-JDBC Drivers for Phoenix as add-ons, but they are not supported by Cloudera. You can find compatible client drivers by searching on the web for avatica and the name of an application programming language

that you want to use. For example, Apache Phoenix/Avatica SQL driver for Go language. For more information and links to driver download, see Apache Calcite Avatica.

You can use the Phoenix ODBC driver and the C# client library to develop .Net applications that interact with Phoenix Query Server.

The applications that you develop will interact with the Phoenix Query Server using the Avatica API and Google Protocol Buffers serialization format.

Using Apache Phoenix-Spark connector

You can use the Apache Phoenix-Spark connector on your secure clusters to perform READ and WRITE operations. The Phoenix-Spark connector allows Spark to load Phoenix tables as Resilient Distributed Datasets (RDDs) or DataFrames and lets you save them back to Phoenix.

Configure Phoenix-Spark connector using Cloudera Manager

Learn how to configure Phoenix-Spark connector using Cloudera Manager.

Procedure

1. Follow Step 1 to Step 7(b) as mentioned in *Configure HBase-Spark connector using Cloudera Manager*.
2. Add the required properties to ensure that all required Phoenix and HBase platform dependencies are available on the classpath for the Spark executors and drivers.

   Add the Phoenix-Spark connector to the spark.executor.extraClassPath and spark.driver.extraClassPath parameters:

   Important: Ensure that you add the Phoenix-Spark connector Jars to the configuration properties and you do not remove any of these files added in Step 1.

   Spark3:

   ```
   spark.executor.extraClassPath=/opt/cloudera/parcels/CDHlib/phoenix_conne
   ctors/phoenix5-spark3-shaded-6.x.y.VERSION.jar
   spark.driver.extraClassPath=/opt/cloudera/parcels/CDHlib/phoenix_connecto
   rs/phoenix5-spark3-shaded-6.x.y.VERSION.jar
   ```
3. Enter a Reason for change, and then click Save Changes to commit the changes.
4. Restart the role and service when Cloudera Manager prompts you to restart.

What to do next

Build a Spark application using the dependencies that you provide when you run your application. If you follow the previous instructions, Cloudera Manager automatically configures the connector for Spark. If you have not:

Consider the following example while using a Spark application:

```
spark3-shell --conf spark.executor.extraClassPath=/opt/cloudera/parcels/CDHl
ib/phoenix_connectors/phoenix5-spark3-shaded-6.x.y.VERSION.jar --conf spark.
jars=hdfs:///path/hbase_jars_common/hbase-site.xml.jar,hdfs:///path/hbase_ja
rs_spark3/hbase-spark3-protocol-shaded.jar,hdfs:///path/hbase_jars_spark3/hb
ase-spark3.jar,hdfs:///path/hbase_jars_common/hbase-shaded-mapreduce-***VERS
ION NUMBER***.jar,hdfs:///path/hbase_jars_common/opentelemetry-api-***VERSIO
N NUMBER***.jar,hdfs:///path/hbase_jars_common/opentelemetry-context-***VERS
ION NUMBER***.jar,hdfs:////path/hbase_jars_spark3/phoenix5-spark3-shaded.jar
```

If you run the `spark3-submit` command in the Yarn cluster mode, update the example as below.

```
spark3-submit --conf spark.driver.extraClassPath=/opt/cloudera/parcels/CDHli
b/phoenix_connectors/phoenix5-spark3-shaded-6.x.y.VERSION.jar  --conf spark.
executor.extraClassPath=/opt/cloudera/parcels/CDHlib/phoenix_connectors/phoe
nix5-spark3-shaded-6.x.y.VERSION.jar  spark.jars=hdfs:///path/hbase_jars_com
mon/hbase-site.xml.jar,hdfs:///path/hbase_jars_spark3/hbase-spark3-protocol-
shaded.jar,hdfs:///path/hbase_jars_spark3/hbase-spark3.jar,hdfs:///path/hbas
e_jars_common/hbase-shaded-mapreduce**VERSION NUMBER*.jar,hdfs:///path/hbase
_jars_common/opentelemetry-api-*VERSION NUMBER*.jar,hdfs:///path/hbase_jars_
common/opentelemetry-context-*VERSION NUMBER**.jar,hdfs:////path/hbase_jars_
spark3/phoenix5-spark3-shaded.jar"
```

Related Information

Configure HBase-Spark connector using Cloudera Manager

Phoenix-Spark connector usage examples

Learn the various usages of the Phoenix-Spark connector.

Examples

You can refer to the following Phoenix-Spark connector examples.

- Reading Phoenix tables
- Saving Phoenix tables
- Using PySpark to READ and WRITE tables

Reading Phoenix tables

For example, you have a Phoenix table with the following DDL, you can use one of the following methods to load the table:

- As a DataFrame using the Data Source API
- As a DataFrame using a configuration object
- As an RDD using a Zookeeper URL

```
CREATE TABLE TABLE1 (ID BIGINT NOT NULL PRIMARY KEY, COL1 VARCHAR);
UPSERT INTO TABLE1 (ID, COL1) VALUES (1, 'test_row_1');
UPSERT INTO TABLE1 (ID, COL1) VALUES (2, 'test_row_2');
```

Example: Load a DataFrame using the Data Source API

```
import org.apache.spark.SparkContext
import org.apache.spark.sql.SQLContext
import org.apache.phoenix.spark._


//delete the following line when running fom spark-shell
val sc = new SparkContext("local", "phoenix-test")
val sqlContext = new SQLContext(sc)
val df = sqlContext.load(
  "org.apache.phoenix.spark",
  //replace "phoenix-server:2181" with the real ZK quorum
  Map("table" -> "TABLE1", "zkUrl" -> "phoenix-server:2181")
)

df
  .filter(df("COL1") === "test_row_1" && df("ID") === 1L)
  .select(df("ID"))
```

```
  .show
```

Example: Load as a DataFrame directly using a Configuration object

```
import org.apache.hadoop.conf.Configuration
import org.apache.spark.SparkContext
import org.apache.spark.sql.SQLContext
import org.apache.phoenix.spark._

val configuration = new Configuration()
// Can set Phoenix-specific settings, requires 'hbase.zookeeper.quorum'

//delete the following line when running fom spark-shell
val sc = new SparkContext("local", "phoenix-test")
val sqlContext = new SQLContext(sc)
// Loads the columns 'ID' and 'COL1' from TABLE1 as a DataFrame
val df = sqlContext.phoenixTableAsDataFrame(
  "TABLE1", Array("ID", "COL1"), conf = configuration
)

df.show
```

Example: Load as an RDD using a Zookeeper URL

```
import org.apache.spark.SparkContext
import org.apache.spark.sql.SQLContext
import org.apache.phoenix.spark._
import org.apache.spark.rdd.RDD

//delete the following line when running fom spark-shell
val sc = new SparkContext("local", "phoenix-test")

// Loads the columns 'ID' and 'COL1' from TABLE1 as an RDD
val rdd: RDD[Map[String, AnyRef]] = sc.phoenixTableAsRDD(
//replace "phoenix-server:2181" with the real ZK quorum
  "TABLE1", Seq("ID", "COL1"), zkUrl = Some("phoenix-server:2181")
)

rdd.count()

val firstId = rdd.first()("ID").asInstanceOf[Long]
val firstCol = rdd.first()("COL1").asInstanceOf[String]
```

Saving Phoenix tables

You can refer to the following examples for saving RDDs and DataFrames.

Example: Saving RDDs

For example, you have a Phoenix table with the following DDL, you can save it as an RDD.

```
CREATE TABLE OUTPUT_TEST_TABLE (id BIGINT NOT NULL PRIMARY KEY, col1 VARCHAR
, col2 INTEGER);
```

The saveToPhoenix method is an implicit method on RDD[Product], or an RDD of Tuples. The data types must correspond to one of the Java types supported by Phoenix.

```
import org.apache.spark.SparkContext
import org.apache.phoenix.spark._

//delete the following line when running fom spark-shell
val sc = new SparkContext("local", "phoenix-test")
```

```
val dataSet = List((1L, "1", 1), (2L, "2", 2), (3L, "3", 3))

sc
  .parallelize(dataSet)
  .saveToPhoenix(
    "OUTPUT_TEST_TABLE",
    Seq("ID","COL1","COL2"),
    //replace "phoenix-server:2181" with the real ZK quorum
    zkUrl = Some("phoenix-server:2181")
  )
```

Example: Saving DataFrames

The save is method on DataFrame allows passing in a data source type. You can use org.apache.phoenix.spark, and must also pass in a table and zkUrl parameter to specify which table and server to persist the DataFrame to. The column names are derived from the DataFrame's schema field names, and must match the Phoenix column names. The save method also takes a SaveMode option, for which only SaveMode.Overwrite is supported. For example, you have a two Phoenix tables with the following DDL, you can save it as a DataFrames.

Using PySpark to READ and WRITE tables

With Spark's DataFrame support, you can use pyspark to READ and WRITE from Phoenix tables.

Example: Load a DataFrame

Given a table TABLE1 and a Zookeeper url of localhost:2181, you can load the table as a DataFrame using the following Python code in pyspark:

```
//replace "phoenix-server:2181" with the real ZK quorum
df = sqlContext.read \
  .format("org.apache.phoenix.spark") \
  .option("table", "TABLE1") \
  .option("zkUrl", "phoenix-server:2181") \
  .load()
```

Example: Save a DataFrame

Given the same table and Zookeeper URLs above, you can save a DataFrame to a Phoenix table using the following code:

```
//replace "phoenix-server:2181" with the real ZK quorum
df.write \
  .format("org.apache.phoenix.spark") \
  .mode("overwrite") \
  .option("table", "TABLE1") \
  .option("zkUrl", "phoenix-server:2181") \
  .save()
```

Note: The functions phoenixTableAsDataFrame, phoenixTableAsRDD and saveToPhoenix all support optionally specifying a conf Hadoop configuration parameter with custom Phoenix client settings, as well as an optional zkUrl parameter for the Phoenix connection URL. If zkUrl isn't specified, it's assumed that the hbase.zookeeper.quorum property has been set in the conf parameter. Similarly, if no configuration is passed in, zkUrl must be specified.

Using Apache Phoenix-Hive connector

This connector enables you to access the Phoenix data from Hive without any data transfer. So the Business Intelligence (BI) logic in Hive can access the operational data available in Phoenix.

Using this connector, you can run a certain type of queries in Phoenix more efficiently than using Hive or other applications, however, this is not a universal tool that can run all types of queries. In some cases, Phoenix can run queries faster than the Phoenix Hive integration and vice versa. In others, you can run this tool to perform operations like many to many joins and aggregations which Phoenix would otherwise struggle to effectively run on its own. This integration is better suited for performing online analytical query processing (OLAP) operations than Phoenix.

Another use case for this connector is transferring the data between these two systems. You can use this connector to simplify the data movement between Hive and Phoenix, since an intermediate form of the data (for example, a .CSV file) is not required. The automatic movement of structured data between these two systems is the major advantage of using this tool. You should be aware that for moving a large amount of data from Hive to Phoenix CSV bulk load is preferable due to performance reasons.

Configure Phoenix-Hive connector using Cloudera Manager

You must configure Phoenix-Hive connector before you can access Phoenix data from Hive.

Before you begin

Ensure that you have the phoenix-hive-version.jar. You can find this JAR file in the /opt/cloudera/parcels/CDH/lib/phoenix_connectors/ path on the Apache Phoenix node, or you can download it from the Cloudera respository.

About this task

To configure the Phoenix-Hive connector using Cloudera Manager:

Procedure

1. Go to the Hive or Hive on Tez service.
2. Click the Configuration tab.
3. Ensure that the HBase service is selected in Hive Service as a dependency.
4. Select Scope Hive Cluster (Service-Wide) .
5. Select Category Advanced .
6. Locate the Hive Auxiliary JARs Directory property or search for it by typing its name in the Search box.
7. Add the following auxiliary path directory: */USR/LOCAL/PHOENIX-HIVE*

   Important: You must manually create the */USR/LOCAL/PHOENIX-HIVE* directory, and copy the /opt/cloudera/parcels/CDH/lib/phoenix_connectors/phoenix5-hive-[****VERSION****]-shaded.jar on every node in the cluster that runs Hive-on-Tez Server or Hive Metastore.

   Ensur that you have the required permissions the access and write to the phoenix-hive directory and the JAR file that you copy into it must be accessible by hive:hive user.

   Note: You can use any directory instead of */USR/LOCAL/PHOENIX-HIVE* that Hive can read or place the JAR file in the existing Hive Auxiliary Directory if one already exists.

8. Enter a Reason for change, and then click Save Changes to commit the changes.
9. Restart the role and service when Cloudera Manager prompts you to restart.

What to do next

If you have installed Hive-Tez, repeat the configuration steps for Hive-Tez as well.

Apache Phoenix-Hive usage examples

You can refer to the following Phoenix-Hive connector examples:

- Creating a table

- Loading data
- Querying data

Creating a table

Creating an external Hive table requires an existing table in Phoenix. Hive manages only the Hive metadata. Dropping an external table from Hive deletes only the Hive metadata, but the Phoenix table is not deleted.

Use the create external table command to create an EXTERNAL Hive table.

```
create external table ext_table (
  i1 int,
  s1 string,
  f1 float,
  d1 decimal
)
STORED BY 'org.apache.phoenix.hive.PhoenixStorageHandler'
TBLPROPERTIES (
  "phoenix.table.name" = "ext_table",
  "phoenix.zookeeper.quorum" = "localhost",
  "phoenix.zookeeper.znode.parent" = "/hbase",
  "phoenix.zookeeper.client.port" = "2181",
  "phoenix.rowkeys" = "i1",
  "phoenix.column.mapping" = "i1:i1, s1:s1, f1:f1, d1:d1"
);
```

Following are some of the recommended parameters that you could use when creating an external table:

Parameter	Default Value	Description
phoenix.table.name	The same name as the Hive table	Name of the existing Phoenix table
phoenix.zookeeper.quorum	localhost	Specifies the ZooKeeper quorum for HBase
phoenix.zookeeper.znode.parent	/hbase	Specifies the ZooKeeper parent node for HBase
phoenix.zookeeper.client.port	2181	Specifies the ZooKeeper port
phoenix.rowkeys	N/A	The list of columns to be the primary key in a Phoenix table
phoenix.column.mapping	N/A	Mappings between column names for Hive and Phoenix

Loading data

Use insert statement to load data to the Phoenix table through Hive.

```
insert into table T values (....);
insert into table T select c1,c2,c3 from source_table;
```

Querying data

You can use HiveQL for querying data in a Phoenix table.

Following are some of the recommended parameters that you could use when querying the data:

Parameter	Default Value	Description
hbase.scan.cache	100	Read row size for a unit request
hbase.scan.cacheblock	false	Whether or not cache block
split.by.stats	false	If true, mappers use table statistics. One mapper per guide post.

Parameter	Default Value	Description
[hive-table-name].reducer.count	1	Number of reducers. In Tez mode, this affects only single-table queries. See Limitations.
[phoenix-table-name].query.hint	N/A	Hint for Phoenix query (for example, NO_INDEX)

Limitations of Phoenix-Hive connector

Following are some of the limitations of Phoenix-Hive connector:

- Only 4K character specification is allowed to specify a full table. If the volume of the data is huge, then there is a possibility to lose the metadata information.
- There is a difference in the way timestamp is saved in Phoenix and Hive. Phoenix uses binary format, whereas Hive uses a text format to store data.
- Hive LLAP is not supported.

Cloudera Runtime 7.2.11

Apache Phoenix SQL Command Reference

Date published: 2020-02-29
Date modified: 2021-09-09

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Phoenix SQL command reference..... 4

Apache Phoenix SQL command reference

You can use these data definition language (DDL) and data manipulation language (DML) commands to perform database operations on Apache Phoenix tables.

Table 1: Apache Phoenix commands

Command	Description	Simple example
CREATE TABLE	Creates a new table	`CREATE TABLE your_schema.TEST_TABLE ( id BIGINT not null primary key, date Date);`
DROP TABLE	Drops an existing table	`DROP TABLE your_schema.your_table;`
CREATE VIEW	Creates a new view over an existing Apache HBase or Phoenix table	`CREATE VIEW "hbase table" ( k VARCHAR primary key, "h" UNSIGNED_LONG) default_column_family='cf1';`
DROP VIEW	Drops an exisiting view. When you drop a view, the underlying table data is not affected; the index data for the view is deleted	`DROP VIEW your_view;`
SELECT	Selects data from one or more tables	`SELECT * FROM TEST_TABLE LIMIT 1000;`
UPSERT VALUES	Inserts value if the value is not present in the table and updates the value in the table if the value is already present. The list of columns is optional and if not present, the data is mapped to the column in the order they are declared in the schema	`UPSERT INTO TEST_TABLE VALUES('A','B',4);`
UPSERT SELECT	Inserts value if the value is not already present and updates the rows in the table based on the results of running the next query	`UPSERT INTO test.targetTable(cq1, cq2) SELECT cq3, cq4 FROM test.sourceTable WHERE cq5 < 100`

Cloudera Runtime 7.2.11

Configuring HBase for Phoenix

Date published: 2020-02-29
Date modified: 2021-09-09

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configure HBase for use with Phoenix..... 4

Configure HBase for use with Phoenix

Before you start using Phoenix with HBase, you must configure HBase to recognize Phoenix operations.

About this task

Important: You must configure HBase for use with Phoenix only for the Cloudera Private Cloud Base deployments. This configuration is done automatically for Cloudera Public Cloud deployments.

Configure the following HBase properties using Cloudera Manager:

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope RegionServer .
4. Select Category Advanced .
5. Locate WAL Codec class property or search for hbase.regionserver.wal.codec.

   Click View as XML and add the following property value:

   ```
   org.apache.hadoop.hbase.regionserver.wal.IndexedWALEditCodec
   ```

6. Set the HBase Service Advanced Configuration Snippet and HBase Client Advanced Configuration Snippet properties to enable user-defined functions. Locate the property or search for HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml and HBase Client Advanced Configuration Snippet (Safety Valve) for hbase-site.xml.

   Add the following property value:

   ```
   <property>
     <name>phoenix.functions.allowUserDefinedFunctions</name>
     <value>true</value>
   </property>
   ```

7. Enter a Reason for change, and then click Save Changes to commit the changes.
8. Restart the role and service when Cloudera Manager prompts you to restart.

Cloudera Runtime 7.2.11

Phoenix Frequently Asked Questions

Date published: 2020-02-29
Date modified: 2021-09-09

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Frequently asked questions..... 4

Frequently asked questions

Find answers to frequently asked questions around Apache Phoenix and its deployment.

Can Phoenix be used for ETL use cases?

Yes. Apache Phoenix is used for OLTP (Online Transactional Processing) use cases and not OLAP (Online Analytical Processing) use cases. Although, you can use Phoenix for real-time data ingest as a primary use case.

What is the typical architecture for a Phoenix deployment?

A typical Phoenix deployment has the following:

- Application
- Phoenix Client/JDBC driver
- HBase client

A Phoenix client/JDBC driver is essentially a Java library that you should include in your Java code. Phoenix uses HBase as storage similar to how HBase uses HDFS as storage. However, the abstraction for Phoenix is not yet complete, for example, for implementing access controls, you need to set ACLs on the underlying HBase tables that contain the Phoenix data.

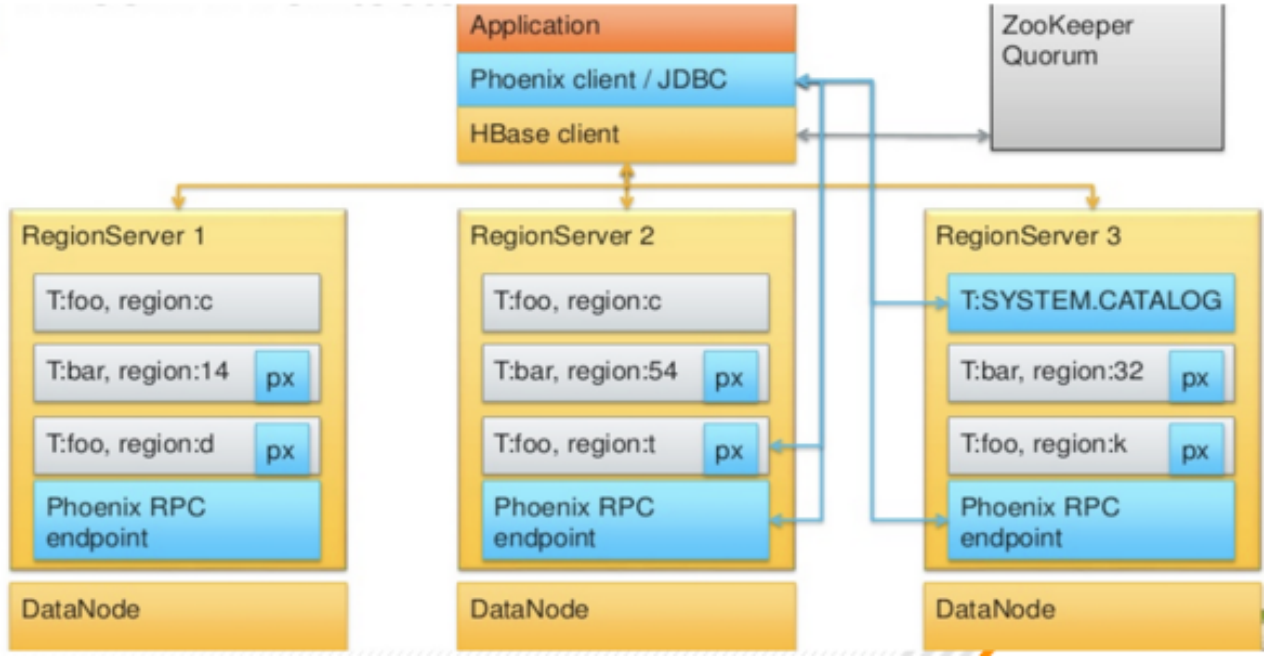

Are there sizing guidelines for Phoenix JDBC servers?

For Phoenix applications, you must follow the same sizing guidelines that you follow for HBase. For more information about Phoenix performance tuning, https://phoenix.apache.org/tuning_guide.html.

Can I govern access to the Phoenix servers?

Yes, you can use Kerberos for authentication. You can configure authorization using HBase authorization.

Can I see the individual cell timestamps in Phoenix tables? Is this something that's commonly used?

You can map HBase's native row timestamp to a Phoenix column. By doing this, you can take advantage of the various optimizations that HBase provides for time ranges on the store files as well as various query optimization capabilities built within Phoenix.

For more information, see https://phoenix.apache.org/rowtimestamp.html

What if the Phoenix index is being built asynchronously and data is added to a table during indexing?

Phoenix does local Indexing for deadlock prevention during global index maintenance.: Phoenix also atomically rebuild index partially when index update fails (PHOENIX-1112).

How do sequences work in Phoenix?

Sequences are a standard SQL feature that allows for generating monotonically increasing numbers typically used to form an ID.

For more information, see https://phoenix.apache.org/sequences.html.

What happens to a Phoenix write when a RegionServer fails?

Writes are durable and durability is defined by a WRITE that is committed to disk (in the Write Ahead Log). So in case of a RegionServer failure, the write is recoverable by replaying the WAL. A "complete" write is one that has been flushed from the WAL to an HFile. Any failures will be represented as exceptions.

Can I do bulk data loads in Phoenix?

Yes, you can do bulk inserts in Phoenix. For more information see https://phoenix.apache.org/bulk_dataload.html.

Can I access a table created by Phoenix using the standard HBase API?

Yes, but is it is not recommended or supported. Data is encoded by Phoenix, so you have to decode the data for reading. Writing to the HBase tables directly would result in corruption in Phoenix.

Can I map a Phoenix table over an existing HBase table?

Yes, as long as Phoenix data types are used. You have to use asynchronous indexes and manually update them since Phoenix won't be aware of any updates.

What are guideposts?

For information about guideposts, see https://phoenix.apache.org/update_statistics.html.

Cloudera Runtime 7.2.11

Managing Apache Phoenix Security

Date published: 2020-02-29
Date modified: 2023-05-09

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Phoenix is FIPS compliant.....4

Managing Apache Phoenix security..... 4

Enable Phoenix ACLs..... 4

Configure TLS encryption manually for Phoenix Query Server.....5

Phoenix is FIPS compliant

Phoenix is now Federal Information Processing Standards (FIPS) compliant.

FIPS are publicly announced standards developed by the National Institute of Standards and Technology for use in computer systems by non-military American government agencies and government contractors. Phoenix Query Server (PQS) is compatible with an FIPS-enabled environment..

PQS can run on an OS with FIPS turned on and can use FIPS-compliant crypto libraries.

OMID is also compatible with an FIPS-enabled environment.

For more information, see *Installing and Configuring Cloudera with FIPS*.

Related Information

Installing and Configuring Cloudera with FIPS

Managing Apache Phoenix security

Apache Ranger manages authorization and access control through a user interface that ensures consistent policy administration for both Apache Phoenix and Apache HBase.

Apache Phoenix namespaces, tables, column family, and columns use the same access control parameters set in Apache HBase. You must first enable Apache Phoenix ACLs support using Cloudera Manager before you can define permissions for your Apache HBase tables if you are using Apache HBase ACLs.

Cloudera Shared Data Experience Data Lake helps you configure and manage authorization and access control through the Apache Ranger user interface that ensures consistent policy administration for Apache HBase. Apache Phoenix security derives policies applied to the underlying Apache HBase tables in Ranger. You can grant read or write permissions to an Apache HBase table for a specific user using the Apache Ranger user interface.

Auto-TLS is enabled by default in Cloudera. But you can also manually configure TLS for Phoenix Query Server. See the related inforamtion section to learn more about security in Cloudera.

Related Information

Configure TLS encryption manually for Phoenix Query Server

Cloudera Security

Enable Phoenix ACLs

To enable Phoenix ACLs using Cloudera Manager, edit the HBase Service advanced configuration snippet for the cluster.

Procedure

1. Go to the HBase service.
2. Click Configuration.
3. Search for the property HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml.
4. Paste your configuration into the Value field and save your changes.

```
<property>
    <name>phoenix.acls.enabled</name>
    <value>true</value>
```

```
  </property>
```

5. Restart your cluster for the changes to take effect.

Configure TLS encryption manually for Phoenix Query Server

You can encrypt communication between clients and the Phoenix Query Server using Transport Layer Security (TLS) formerly known as Secure Socket Layer (SSL). You must follow these steps to manually configure TLS for Phoenix Query Server.

Before you begin

- Keystores containing certificates bound to the appropriate domain names must be accessible on all hosts running the Phoenix Query Server role of the Phoenix service.
- Keystores for Phoenix must be owned by the phoenix group, and have 0440 file permissions (that is, the file must be readable by the owner and group).
- Absolute paths to the keystore and truststore files must be specified. These settings apply to all hosts on which daemon roles of the Phoenix service run. Therefore, the paths you choose must be valid on all hosts.
- The Cloudera Manager version must support the TLS/SSL configuration for Phoenix at the service level. Ensure you specify absolute paths to the keystore and truststore files. These settings apply to all hosts on which daemon roles of the service in question run. Therefore, the paths you choose must be valid on all hosts.

  An implication of this is that the keystore file names for a given service must be the same on all hosts. If, for example, you have obtained separate certificates for Phoenix daemons on hosts node1.example.com and node2.example.com, you might have chosen to store these certificates in files called phoenix-node1.keystore and phoenix-node2.keystore (respectively). When deploying these keystores, you must give them both the same name on the target host — for example, phoenix.keystore.

Procedure

1. In Cloudera Manager, select the Phoenix service.
2. Click the Configuration tab.
3. Use the Scope / Query Server filter.
4. Search for tls.
5. Select Enable TLS/SSL for Query Server.
6. Edit the following TLS/SSL properties according to your configuration.

Table 1:

Property	Description
Query Server TLS/SSL Server JKS Keystore File Location	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when Query Server is acting as a TLS/SSL server. The keystore must be in JKS format.
Query Server TLS/SSL Server JKS Keystore File Password	The password for the Query Server JKS keystore file.
Query Server TLS/SSL Client Trust Store File	The location on disk of the truststore file, in .jks format, used to confirm the authenticity of TLS/SSL servers to which the Query Server might connect. This is used when Query Server is the client in a TLS/SSL connection. This truststore file must contain the certificate(s) used to sign the connected service(s). If this parameter is not specified, the default list of known certificate authorities is used instead.

Property	Description
Query Server TLS/SSL Client Trust Store Password	The password for the Query Server TLS/SSL Certificate Trust Store File. This password is not mandatory to access the truststore; this field is optional. This password provides optional integrity checking of the file. The contents of truststores are certificates, and certificates are public information.

7. Click Save Changes.
8. Restart the Phoenix service.

Related Information

Managing Apache Phoenix security

Cloudera Runtime 7.2.11

Apache Phoenix

Date published: 2020-02-29
Date modified: 2021-09-09

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to Apache Phoenix.....4

Apache Phoenix and SQL.....4

Using secondary indexing in Apache Phoenix.....4

Introduction to Apache Phoenix

Apache Phoenix is is a SQL layer for Apache HBase that provides a programmatic ANSI SQL interface.

Apache Phoenix implements best-practice optimizations to enable software engineers to develop HBase based next-generation applications that operationalize big data. Using Phoenix, you can create and interact with tables in the form of typical DDL/DML statements using the Phoenix standard JDBC API.

Apache HBase timestamps are the core part of the Apache Phoenix functionality. It is used as part of the *Travel in time* functionality and by the Transactional engines. Due to this, we expect that the data must have valid timestamps that signifies the time when the record is created. Invalid timestamps cause incorrect behavior for most Apache Phoenix functionalities, such as secondary indexes or CRUD operations. If your data is relying on the special behavior of the record's timestamps or future timestamps, you must not use such data with Apache Phoenix.

Apache Phoenix and SQL

You can use the Apache Phoenix SQL commands to create, drop, or modify an Apache HBase table. You can also create Apache Phoenix views that are virtual tables that share the same Apache HBase table.

The Apache Phoenix commands enable you to use standard SQL data definition language (DDL) and data manipulation language (DML) commands to interact with Apache HBase tables. You can create a new Apache HBase table using the standard CREATE TABLE SQL command or create a view on an existing Apache HBase table using the VIEW command. View enables you to have multiple virtual tables that share the same physical Apache HBase table.

Apache HBase tables and Apache Phoenix tables have a one-to-one relationship. This means each Apache HBase table is associated with a corresponding Apache Phoenix table.

Warning: Although you can modify Apache Phoenix tables using the Apache HBase native APIs, this is not supported and results in errors, inconsistent indexes, incorrect query results, and sometimes corrupt data.

Using Apache Phoenix commands you can do the following tasks:

- Create or alter Apache HBase tables using DDL commands like CREATE TABLE
- Modify contents in an Apache HBase table using DML commands like UPSERT VALUES

Using secondary indexing in Apache Phoenix

Apache Phoenix uses a secondary index to serve queries. An index table is an Apache Phoenix table that stores the reference copy of some or all the data in the main table.

You can use a secondary index to access data from its primary data access path. When you use a secondary index, the indexed column qualifiers or rows form a unique row key that allows you to do point lookups and range scans.

Secondary index types

Apache Phoenix supports global and local secondary indexes. Global indexes is used for all typical use cases. You can use local indexes for specific use cases where you want the primary and the index table to be present in the same Apache HBase region.

- Use global indexes for read-heavy use cases. Use the covered-global index to save on read-time overheads. Global indexes are used to co-locate related information.

- Use local indexes for write-heavy use cases. Use the functional-local index on arbitrary expressions to query specific combinations of index queries. A local index is an in-partition index that is optimized for writes but requires more data to be read to answer a query.

Note: If your table is large, you can use the ASYNC keyword with CREATE INDEX to create an index asynchronously. ASYNC uses a MapReduce job to distribute index workload, and every single mapper works with each data table region and writes to the index region. Therefore freeing up critical resources during indexing.

The following tables list the index type and index scope with a description and example for each index type:

Table 1:

Index type	Description
Covered	Include the data that you want to access from the primary table in the index rows. The query does not have to access the primary table once the index entry is found. Benefits: Save read-time overhead by only accessing the index entry. In the following example, column v3 is included in the index to avoid the query to access the primary table to retrieve this information. Example The following command creates indexes on the v1 and v2 columns and include the v3 column as well: `CREATE INDEX my_index ON exp_table ( v1,v2) INCLUDE(v3);`

Functional	Create an index on arbitrary expressions. When your query uses the expression, the index is used to retrieve the results instead of the data table.
	Benefits: Useful for certain combinations of index queries.
	Example
	Run the following command to create a functional index so that you can perform case insensitive searches on the combined first name and last name of a person:
	`CREATE INDEX UPPER_NAME ON EMP (UPPER(FIRST_NAME||' '||LAST_NAME));`
	Search on the combined first name and last name using the following command:
	`SELECT EMP_ID FROM EMP WHERE UPPER(FIRST_NAME||' '||LAST_NAME)='EXP_NAME';`

Table 2:

Index scope	Description
Global	You can use this when you have read-heavy use cases. Each global index is stored in its own table, and therefore it is not co-located with the data table. A Global index is a covered index. It is used for queries only when all columns in that query are included in that index. Example Run the following command to create a global index: `CREATE INDEX my_index ON exp_table ( v1);`
Local	You can use this for write-heavy use cases. Each local index is stored within the data table. Example Run the following command to create a local index: `CREATE LOCAL INDEX my_index ON exp_t able (v1);`

Cloudera Runtime 7.2.11

Phoenix Performance Tuning

Date published: 2020-02-29
Date modified: 2021-09-09

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Performance tuning..... 4

Performance tuning

You can use the following configuration properties to tune Phoenix to work optimally on your cluster. You can tune your Phoenix deployment by configuring certain Phoenix specific properties that are configured both on the client and server side hbase-site.xml files. For a full list of Phoenix Tuning properties that are available, see the Apache Phoenix tuning guide.

Table 1:

Property	Description	Default
phoenix.query.threadPoolSize	The number of threads in client-side thread pool executor. As the number of machines/ cores in the cluster grows, this value should be increased.	128
phoenix.query.queueSize	Max queue depth of the bounded round robin backing the client side thread pool executor, beyond which an attempt to queue additional work is rejected. If zero, a SynchronousQueue is used instead of the bounded round-robin queue. The default value is 5000.	5000
phoenix.stats.guidepost.width	The server-side parameter that specifies the number of bytes between guideposts. A smaller amount increases parallelization, but also increases the number of chunks which must be merged on the client side. The default value is 100 MB.	104857600
phoenix.stats.guidepost.per.region	The server-side parameter that specifies the number of guideposts per region. If set to a value greater than zero, then the guidepost width is determined by MAX_FILE_SIZE of table/phoenix .stats.guidepost.per.region. Otherwise, if not set, then the phoenix.stats.guidepost.width parameter is used. No default value.	None
phoenix.stats.updateFrequency	The server-side parameter that determines the frequency in milliseconds for which statistics will be refreshed from the statistics table and subsequently used by the client. The default value is 15 min.	900000
phoenix.query.spoolThresholdBytes	Threshold size in bytes after which results from parallelly executed query results are spooled to disk. Default is 20 mb.	20971520
phoenix.query.maxSpoolToDiskBytes	Threshold size in bytes up to which results from parallelly executed query results are spooled to disk above which the query will fail. Default is 1 GB.	1024000000
phoenix.query.maxGlobalMemoryPercentage	Percentage of total heap memory (i.e. Runtime.getRuntime().maxMemory()) that all threads may use. Only course grain memory usage is tracked, mainly accounting for memory usage in the intermediate map built during group by aggregation. When this limit is reached, the clients' block attempting to get more memory, essentially throttling memory usage. Defaults to 15%	15

Property	Description	Default
phoenix.query.maxGlobalMemorySize	Max size in bytes of total tracked memory usage. By default it is not specified, however, if present, the lower of this parameter and the phoenix.query.maxGlobalMemoryPercentage will be used.	
phoenix.query.maxGlobalMemoryWaitMs	The maximum amount of time that a client will block while waiting for more memory to become available. After this amount of time, an InsufficientMemoryException error is displayed. The default value is 10 seconds.	10000
phoenix.query.maxTenantMemoryPercentage	The maximum percentage of phoenix.query.maxGlobalMemoryPercentage that a tenant is allowed to consume. After this percentage, an InsufficientMemoryException error is displayed. Default is 100%	100
phoenix.mutate.maxSize	The maximum number of rows that may be batched on the client before a commit or rollback must be called.	500000
phoenix.mutate.batchSize	The number of rows that are batched together and automatically committed during the execution of an UPSERT SELECT or DELETE statement. This property may be overridden at connection time by specifying the UpsertBatchSize property value. Note that the connection property value does not affect the batch size used by the coprocessor when these statements are executed completely on the server side.	1000
phoenix.query.maxServerCacheBytes	Maximum size (in bytes) of a single subquery result (usually the filtered result of a table) before compression and conversion to a hash map. Attempting to hash an intermediate subquery result of a size bigger than this setting will result in a MaxServerCacheSizeExceededException. Default 100MB.	104857600
phoenix.coprocessor.maxServerCacheTimeToLiveMs	Maximum living time (in milliseconds) of server caches. A cache entry expires after this amount of time has passed since last access. Consider adjusting this parameter when a server-side IOException("Could not find a hash cache for joinId") happens. Getting warnings like "Earlier hash cache(s) might have expired on servers" might also be a sign that this number should be increased.	30000
phoenix.query.useIndexes	Client-side property determining whether or not indexes are considered by the optimizer to satisfy a query. Default is true	true
phoenix.index.failure.handling.rebuild	Server-side property determining whether or not a mutable index is rebuilt in the background in the event of a commit failure. Only applicable for indexes on mutable, non-transactional tables. Default is true	true
phoenix.groupby.maxCacheSize	Size in bytes of pages cached during GROUP BY spilling. Default is 100Mb	102400000
phoenix.groupby.estimatedDistinctValues	Number of estimated distinct values when a GROUP BY is performed. Used to perform initial sizing with the growth of 1.5x each time reallocation is required. Default is 1000	1000

Property	Description	Default
phoenix.distinct.value.compress.threshold	Size in bytes beyond which aggregate operations which require tracking distinct value counts (such as COUNT DISTINCT) will use Snappy compression. Default is 1Mb	1024000
phoenix.index.maxDataFileSizePerc	The percentage used to determine the MAX_FILESIZE for the shared index table for views relative to the data table MAX_FILESIZE. The percentage should be estimated based on the anticipated average size of a view index row versus the data row. Default is 50%.	50
phoenix.coprocessor.maxMetaDataCacheTimeToLiveMs	Time in milliseconds after which the server-side metadata cache for a tenant will expire if not accessed. Default is 30mins	180000
phoenix.coprocessor.maxMetaDataCacheSize	Max size in bytes of total server-side metadata cache after which evictions will begin to occur based on least recent access time. Default is 20Mb	20480000
phoenix.client.maxMetaDataCacheSize	Max size in bytes of total client-side metadata cache after which evictions will begin to occur based on least recent access time. Default is 10Mb	10240000
phoenix.sequence.cacheSize	Number of sequence values to reserve from the server and cache on the client when the next sequence value is allocated. Only used if not defined by the sequence itself. Default is 100	100

Cloudera Runtime 7.3.1

Release Notes

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Overview 6

What's New In Cloudera Runtime 7.3.1 6
- Atlas 6
- What's New in Cloud Connectors 8
- HBase 8
- Hive 10
- Hue 10
- Iceberg 11
- Impala 11
- Kafka 11
- Livy 12
- Oozie 12
- Phoenix 12
- Ranger 13
- Schema Registry 14
- Solr 14
- Spark 14
- SMM 15
- YARN and YARN Queue Manager 16
- Unaffected Components in this release 16

What's new in Platform Support 17

Cloudera Runtime Component Versions 17

Using the Cloudera Runtime Maven repository 7.3.1 18
- Cloudera Runtime 7.3.1.0-197 19

Fixed Issues In Cloudera Runtime 7.3.1 37
- Atlas 37
- Avro 38
- Cloud Connectors 38
- Cruise Control 38
- Hadoop 38
- HDFS 38
- HBase 39
- Hive 40
- Hue 40
- Impala 40
- Fixed Issues in Apache Iceberg 41
- Kafka 42
- Kudu 42
- Knox 43

Livy.... 44
Oozie.... 45
Parquet.... 45
Phoenix.... 45
Ranger.... 46
Schema Registry.... 50
Solr.... 50
Spark.... 51
SMM.... 51
SRM.... 51
Tez.... 51
YARN and YARN Queue Manager.... 52
Zookeeper.... 52

Known Issues In Cloudera Runtime 7.3.1.... 52
Atlas.... 52
Avro.... 57
Known Issues in Cloud Connectors.... 57
Cruise Control.... 57
Hadoop.... 58
HBase.... 58
HDFS.... 59
Hive.... 62
Hue.... 62
Iceberg.... 65
Impala.... 66
Kafka.... 72
Kerberos.... 77
Knox.... 77
Kudu.... 77
Oozie.... 78
Parquet.... 78
Phoenix.... 78
Ranger.... 78
Schema Registry.... 79
Solr.... 80
Spark.... 88
Sqoop.... 88
SMM.... 89
SRM.... 90
MapReduce, YARN and YARN Queue Manager.... 91
Zookeeper.... 94

Behavioral Changes In Cloudera Runtime 7.3.1.... 94
Behavioral Changes in Atlas.... 94
Behavioral Changes in Hive.... 95
Behavioral Changes in Impala.... 95
Behavioral Changes in Knox.... 95
Behavioral Changes in Livy.... 96
Behavioral Changes in Ranger.... 96
Behavioral Changes in Spark.... 97

Deprecation Notices In Cloudera Runtime 7.3.1.... 97

Platform and OS.....98
Deprecation Notices for Apache Kafka.....99
Deprecation Notices for Apache Livy.....99
Deprecation Notices for Apache Oozie.....100
Deprecation Notices for Apache Spark.....100
Deprecation Notices for Apache Zeppelin.....100

Fixed Common Vulnerabilities and Exposures 7.3.1.....100

Overview

This document provides you with the latest information about Cloudera Runtime 7.3.1. It includes improvements and describes new features, bug fixes, tech previews and more. For detailed information about the runtime components, see Cloudera documentation.

What's New In Cloudera Runtime 7.3.1

This version of Cloudera Runtime provides you with several new capabilities. Learn how the new features and improvements benefit you.

Spark 2 removed from Cloudera Runtime

Spark 3 is the default Spark version in Cloudera Runtime. Spark 2 has been removed and no longer available in 7.3.1.

Important:

Spark 3 contains a large number of changes from Spark 2.

Refer to *Upgrading Spark* for more information on upgrading Spark clusters to 7.3.1, and *Migrating Spark Applications* for more information on migrating your existing Spark applications between versions 2 and 3.

Upgrade Cloudera 7.2.17.200 through Cloudera 7.2.17.500 to Cloudera 7.3.1.

You can perform an in-place upgrade from 7.2.17.200 through Cloudera 7.2.17.500 to Cloudera 7.3.1. For more information, see the Upgrading documentation of Cloudera Public Cloud.

Upgrade Cloudera 7.2.18.0 through Cloudera 7.2.18.200 to Cloudera 7.3.1

You can perform an in-place upgrade from Cloudera 7.2.18.0 through Cloudera 7.2.18.200 to Cloudera 7.3.1. For more information, see the Upgrading documentation of Cloudera Public Cloud.

Rollback Cloudera 7.3.1 to a previous version

Rolling back to a previous version after upgrading to 7.3.1 is not supported. There is a recovery process for Data Lakes to return the cluster to its pre-upgrade state in case the upgrade fails. For more information, see Recovering from failed upgrades.

Related Information

Upgrading Spark
Migrating Spark Applications

What's New in Apache Atlas

Learn about the new features of Apache Atlas in Cloudera Runtime 7.3.1.0.

Cloudera Navigator Data Management upgrade to Apache Atlas

In Cloudera, Apache Atlas fulfills the metadata collection role that in CDH was filled by Cloudera Navigator Data Management. The upgrade to Cloudera provides a method to migrate Navigator content, including technical and business metadata, to Atlas. For information, see Migrating Navigator content to Atlas. For the cluster audit functionality handled by Navigator, see the (production version of) access auditing provided by Apache Ranger see Ranger Audit Overview.

Business Metadata: Entity model extensions

This release of Atlas provides the ability for data stewards to add custom attributes to existing entity types and set their values on existing entities. This functionality allows an organization to extend its enterprise data model with curated master data attributes that have specific meaning for the business. Business Metadata attributes are defined centrally and can be used on designated entity types. Administrators can control who can view, add values to, and create or update set collections of Business Metadata attributes. Privileged users can add free-form values or select from predefined values to the attribute for a given entity. For more information, see Leveraging Business Metadata.

Bulk import of Business Metadata attribute associations

Atlas provides an interface to import a list of assignments of Business Metadata attributes to entities. The list includes information to uniquely identify the Business Metadata attribute and the targeted entity. The list can be formatted as comma-separated values (.CSV) or Microsoft Excel (.XLS) formatted file. For more information, see Importing Business Metadata associations in bulk

Bulk import of Glossary terms

Atlas provides an interface to import a list of terms into existing Glossaries. The list can include any or all of the metadata associated with a given term. The list can be formatted as comma-separated values (.CSV) or Microsoft Excel (.XLS) formatted file. For more information, see Importing Glossary terms in bulk

Administrator features have a home in the Atlas UI

The Atlas UI now contains an Administration section available to users with administrator privileges:

- Review system-level audits, such as created by entity purge events. See Auditing purged entities.
- Create enumerations for use as attribute values. See Defining Apache Atlas enumerations.
- Create Business Metadata attributes. See Creating Business Metadata.

Open the Administration section from the user menu at the top right of the Atlas UI.

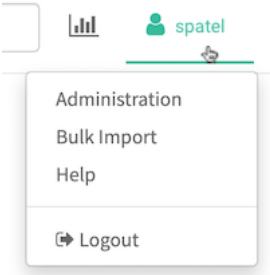

Purge of deleted entities

Atlas now provides the ability to clear the metadata for entities that represent data assets and operation that no longer exist on the cluster. The purge functionality is available to users with administrator privilege; run a REST API command that lists one or more GUID values for the deleted entities. For more information, see Purging deleted entities.

Enhancements to Basic Search in Atlas

Atlas Basic Search includes a filter to allow users to search for entities based on values of entity attributes. In this release, the search filter includes access to system attributes, labels, classifications, and user-defined properties. The filter allows users to build logical combinations of search criteria, including multiple classifications. For more information, see Using Basic Search.

System attributes filter searches

Atlas **Basic** and **Advanced** search now allow you to filter based on system attribute values, including when and by whom an entity was created. Classifications are also modeled as system attributes, so this change allows you to filter on the classifications assigned to an entity and to distinguish between classifications and propagated classifications. System attributes are available in the search filter.

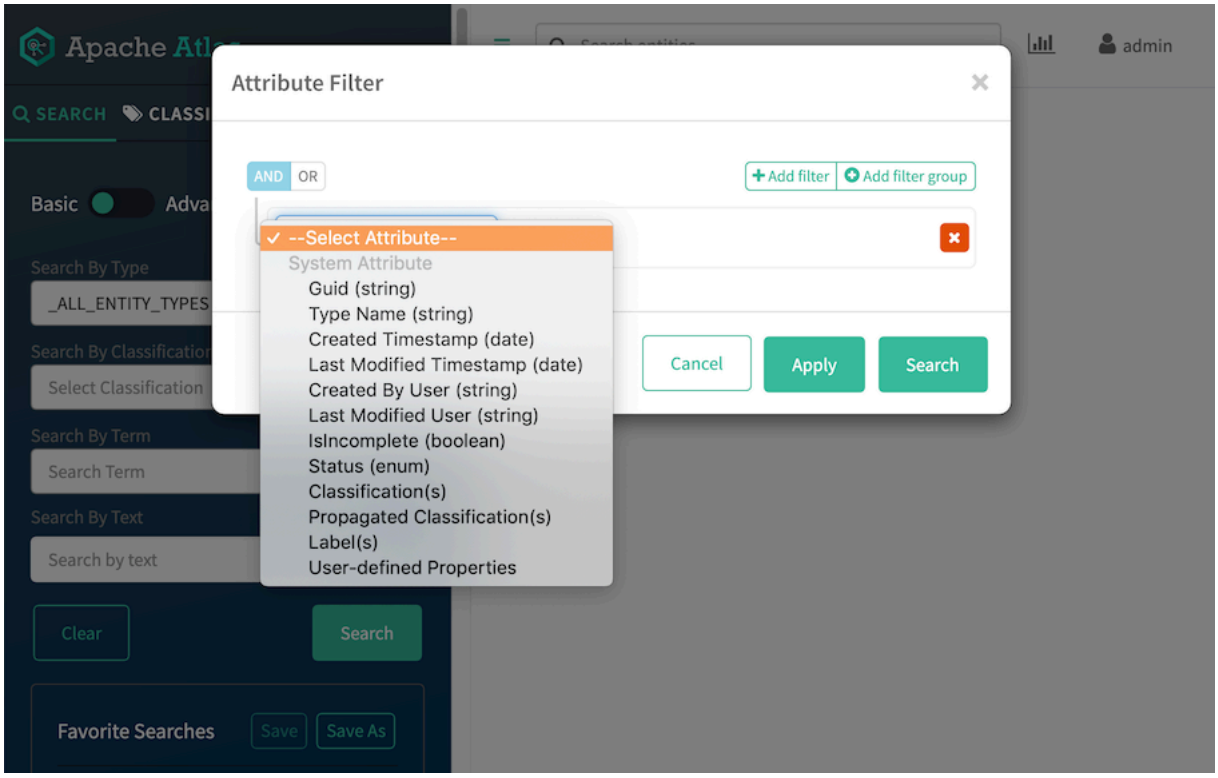

For information on using system attributes in **Advanced Search**, see Apache Atlas metadata attributes.

What's New in Cloud Connectors

Learn about the new features of Cloud Connectors in Cloudera Runtime 7.3.1.

Migration to AWS V2 SDK

The following improvements and enhancements have been introduced for AWS V2 SDK migration:

- Dual-layer server-side encryption (DSSE) has been enabled with AWS KMS keys
- AWS SDK V2 has been upgraded to 2.25.53

What's New in Apache HBase

Learn about the new features of Apache HBase in Cloudera Runtime 7.3.1.0.

HBase supports JDK 17

HBase supports Oracle JDK version 17.0.6 starting from CDP Runtime 7.1.9. For more information on JDK 17, see Java Requirements.

HBase rebase to 2.4.17

CDP Private Cloud Base is updated to use Apache HBase version 2.4.17 and Apache HBase Thirdparty to base version 4.1.1 for a smoother and better functionality. Upgrade your HBase client applications for seamless connectivity.

HBase supports Snappy with /tmp directory mounted with noexec option

In Cloudera Manager, the Snappy temporary directory configuration item is added to HBase Master and HBase RegionServer to allow Snappy compression when /tmp directory is mounted with noexec option.

Operating system support

HBase is now supported on the following operating systems:

- RHEL-9.1
- RHEL-8.8
- RHEL-8.8 FIPS
- Oracle-8.8 UEK
- SLES-15 SP4 for x86

HBase supports load balancing using a cache-aware load balancer

The HBase balancer now supports the cache-aware load balancer that enhances the capability of HBase to enable the balancer to consider the cache allocation of each region on region servers while calculating a new assignment plan. This balancer also uses the region or region server cache allocation information reported by the region servers to calculate the percentage of HFiles cached for each region on the hosting server, and then use that as an additional factor while deciding an optimal new assignment plan.

HBase supports Snappy with /tmp directory mounted with noexec option

In Cloudera Manager, the Snappy temporary directory configuration item is added to HBase Master and HBase RegionServer to allow Snappy compression when /tmp directory is mounted with noexec option.

HBase supports Netty native libraries with /tmp directory mounted with noexec option

In Cloudera Manager, the Netty native library working directory configuration item is added to HBase Master and HBase RegionServer to support HBase with /tmp directory mounted with noexec option.

HBase shows cached percentage for region data on RegionServer UI

An important feature for Cloudera Operational Database (COD) over S3 with ephemeral cache is the process of warming up the cache at region opening (also known as *cache prefetch*). The goal is to load the most of the dataset before any client reads, so that a reduced latency and optimal performance can be achieved for the application requests. This *prefetch* process takes several hours on very large datasets, and the operators might want to monitor the progress of this cache loading. To handle this, HBase has introduced new metrics about the percentage of individual regions data currently cached, and it also added this information to the Storefile Metrics tab in the Regions section of the RegionServer UI.

Related Apache JIRA: HBASE-28246

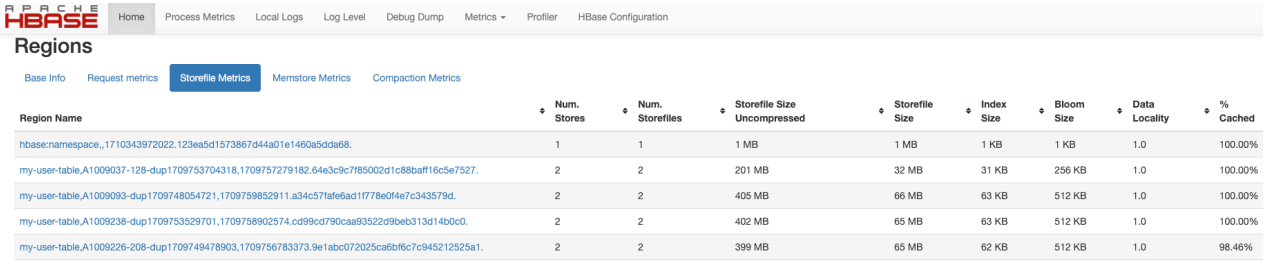

Region Name	Num. Stores	Num. Storefiles	Storefile Size Uncompressed	Storefile Size	Index Size	Bloom Size	Data Locality	% Cached
hbase:namespace,,1710343972022.123ea5d1573867d44a01e1460a5dda68.	1	1	1 MB	1 MB	1 KB	1 KB	1.0	100.00%
my-user-table,A1009037-128-dup1709753704318,1709757279182.64e3c9c7f85002d1c88baff16c5e7527.	2	2	201 MB	32 MB	31 KB	256 KB	1.0	100.00%
my-user-table,A1009093-dup1709748054721,1709759852911.a34c57fafe6ad1f778e0f4e7c343579d.	2	2	405 MB	66 MB	63 KB	512 KB	1.0	100.00%
my-user-table,A1009238-dup1709753529701,1709758902574.cd99cd790caa93522d9beb313d14b0c0.	2	2	402 MB	65 MB	63 KB	512 KB	1.0	100.00%
my-user-table,A1009226-208-dup1709749478903,1709756783373.9e1abc072025ca6bf6c7c945212525a1.	2	2	399 MB	65 MB	62 KB	512 KB	1.0	98.46%

HBase supports disabling the caching for the individual column families

In some use cases, not all tables in the dataset have the same SLA requirements. If the total cache capacity is much smaller than the whole dataset, an alternative is to restrict the cache usage by the tables with critical response times. In HBase, you can now implement this by disabling the cache on individual column families.

On an hbase shell, perform the following `alter` command for each column family that does not require caching.

```
alter 'NAMESPACE:TABLENAME', {NAME=>'CF_NAME', BLOCKCACHE => 'false'}
```

HBase supports truncating the regions in a table

You can now truncate individual regions of an HBase table using the `truncate_region` command.

The command syntax is as follows.

`truncate_region` '*REGIONNAME*'

`truncate_region` '*ENCODED_REGIONNAME*'

For example,

```
hbase:008:0> list_regions 'employee'
                                                    SERVER_NAME |
                                       REGION_NAME |  START_KEY |    END_KEY |
 SIZE |   REQ |   LOCALITY |
 ------------------------------------------------------------ | ---------
------------------------------------------------- | ---------- | ----------
| ----- | ----- | ---------- |
 ccycloud-4.nightly-7x-by.root.comops.site,22101,1718869191555 |  employe
e,,1718877308795.66828b0fe6ceda3e28608617eb6f6b3f. |            |          2
 |     1 |     2 |        1.0 |
 ccycloud-2.nightly-7x-by.root.comops.site,22101,1718869191308 | employee,2,
1718877308795.ff9b19452fecea6353694583e3473b5b. |          2 |            |
    1 |     2 |        1.0 |
 2 rows
Took 0.1088 seconds
hbase:014:0> truncate_region 'employee,2,1718877308795.ff9b19452fecea6353694
583e3473b5b.'
Took 0.6236 seconds
hbase:010:0> truncate_region 'ff9b19452fecea6353694583e3473b5b'
Took 0.6500 seconds
```

What's New in Apache Hive

There are no new features for Apache Hive in Cloudera Runtime 7.3.1.

What's New in Hue

Learn about the new features of Hue in Cloudera Runtime 7.3.1

General availability (GA) of the SQL AI Assistant

Hue leverages the power of Large Language Models (LLM) to help you generate SQL queries from natural language prompts and also provides options to optimize, explain, and fix queries, promoting efficient and accurate practices for accessing and manipulating data. You can use several AI services and models such as OpenAI's GPT service, Amazon Bedrock, and Azure's OpenAI service to run the Hue SQL AI assistant.

- To learn more about the supported models and services, limitations, and what data is shared with the LLMs, see About the Hue SQL AI Assistant.

- To set up and enable the SQL AI Assistant, see About setting up the Hue SQL AI Assistant.
- To see how to generate, edit, explain, optimize, and fix queries, see Starting the SQL AI Assistant in Hue.

Hue supports Python 3.9 on RHEL 8.8 and RHEL 8.10

Starting from the 7.3.1 release, Hue supports only Python 3.9 for RHEL 8.8 and RHEL 8.10. Before upgrading to Cloudera Runtime 7.3.1, you must install Python 3.9 on all the Hue servers, as Hue requires a Python 3.9 version and does not start without it. For information about migrating from Python 3.8 to Python 3.9, see Migrating from Python 3.8 to Python 3.9 on RHEL 8.8 or RHEl 8.10.

What's New in Apache Iceberg

There are no new features for Apache Iceberg in Cloudera Runtime 7.3.1. You can continue to use the Iceberg capabilities that were available in the earlier version.

What's New in Apache Impala

Learn about the new features of Apache Impala in Cloudera Runtime 7.3.1.

Enhanced event processing information in /events web UI

The /events page of Catalogd now includes enhanced metrics, such as event processing lag and details on the current event batch. Error messages are highlighted at the top, if event processing stops. They will disappear after global INVALIDATE METADATA.

Apache Jira: IMPALA-12782

Improved query timeline with disk, network, and memory usage metrics

This update enhances the query timeline in the WebUI by adding disk and network usage metrics alongside CPU utilization. Metrics now display in human-readable formats like KB, MB, and GB. The update also introduces resizable and closable charts, zoom controls for easy navigation, and auto-scaling for timeticks during horizontal zoom. This enhanced display makes monitoring resource usage more intuitive for users.

Apache Jira: IMPALA-12364

Unicode column name support in Impala

Impala now supports Unicode characters in column names, aligning with Hive's support for non-ASCII characters. This enhancement leverages Hive's validateColumnName() function, which removes restrictions on column names at the metadata level. With this update, Impala allows greater flexibility for column naming while remaining consistent with Hive's metadata validation standards.

Apache Jira: IMPALA-12465

Support custom hash partitions at range level in Kudu tables

Impala now supports specifying custom hash partitions at the range level in Kudu tables. You can define hash schemas within specific partitions using the updated CREATE TABLE and ALTER TABLE syntax, and view them with the new SHOW HASH SCHEMA statement. This update aligns hash partitioning more closely with range partitioning, enhancing flexibility while maintaining backward compatibility.

Apache Jira: IMPALA-11430

What's New in Apache Kafka

Learn about the new features of Apache Kafka in Cloudera Runtime 7.3.1.

Kafka Rolling Restart check—all partitions fully replicated

A new broker rolling restart check option, all partitions fully replicated has been introduced. Selecting this option ensures that all partitions are in a fully synchronized state when a broker is stopped. For more information, see Rolling restart checks.

What's New in Livy

There are no new features for Livy in Cloudera Runtime 7.3.1.0.

What's New in Apache Oozie

Learn about the new features of Apache Oozie in Cloudera Runtime 7.3.1.0.

OpenJPA 3 upgrade in Apache Oozie

Third-party library OpenJPA is upgraded from version 2 to version 3 in Apache Oozie. This upgrade includes the following updates:

- New configuration properties
- Deprecated configuration properties
- Enhanced error handling

For more information, see OpenJPA upgrade.

What's New in Apache Phoenix

Learn about the new features of Phoenix in Cloudera Runtime 7.3.1.0.

Phoenix now supports alternate HBase connection registries

You can now use the Phoenix thick client using additional HBase connection registries. Phoenix now supports a Zookeeper-less connection strategy using a Master Registry implementation.

Phoenix now introduces the following JDBC URL variants.

- jdbc:phoenix+zk: Uses Zookeeper. This is the original registry supported since the inception of HBase and Phoenix.
- jdbc:phoenix+rpc: Uses RPC to connect to the specified HBase Region Server or Master nodes.
- jdbc:phoenix+master: Uses RPC to connect to the specified HBase Master nodes.

For more information, see Using the Phoenix JDBC Driver.

Phoenix FIPS support

Phoenix is now Federal Information Processing Standards (FIPS) compliant. For more information, see Phoenix is FIPS compliant.

Phoenix supports rolling restart

If Phoenix service instances are running on multiple nodes, while performing a rolling restart the Phoenix services are restarted one after another ensuring zero downtime.

The OMID service also supports rolling restart and the High Availability (HA) mode for the OMID TSO server. If OMID service instances are running on multiple nodes, while performing a rolling restart the OMID services are restarted one after another ensuring zero downtime. For more information, see Preparing for a Zero Downtime Upgrade.

Phoenix supports JDK 17

Phoenix supports Oracle JDK version 17.0.6 starting from CDP Runtime 7.1.9. For more information on JDK 17, see Java Requirements.

Operating system support

Phoenix is now supported on the following operating systems:

- RHEL-9.1
- RHEL-8.8
- RHEL-8.8 FIPS
- Oracle-8.8 UEK
- SLES-15 SP4 for x86

What's New in Apache Ranger

Learn about the new features of Apache Ranger in Cloudera Runtime 7.3.1.

ZooKeeper SSL/TLS support for Ranger

Ranger and Ranger plugin audit to Solr supports ZooKeeper-SSL enabled connection.

Support multiple columns policy creation in Ranger for Grant/Revoke request

This enhancement supports multiple columns policy creation in Ranger for Grant/Revoke requests for Impala.

Ranger REST API improvements

Ranger REST APIs have the following changes:

- The following APIs have been removed:
  - assets/credstores - GET, POST, PUT
  - credstores/count - GET
  - credstores/{id} - GET
  - /xusers/auditmaps - GET
  - /xusers/auditmaps/count - GET
  - /xusers/permmaps - GET
  - /resource/{id} - GET
  - assets/policyList/{repository}
  - /groupgroups/* (All methods)
- The following APIs were not returning any access code when request is denied; now they suppose to 403:
  - /tags/tags
  - /tags/types
  - /tags/resources APIs
- Earlier When a non admin user makes a DELETE request to below endpoint, it was returning 405 method not allowed. However, now it returns 403.
  - /assets/resources/{resource_id}
- Earlier the API was not accessible for the keyadmin role users, but now it shall be accessible.
  - /xaudit/trx_log
- Earlier the below mentioned API was returning {OWNER} and {USER} users in the response but now onwards it will not return because access to the users list will be based on which role user is having permissions to which role user.
  - /service/xusers/users

- The API endpoint /xaudit/trx_log/{trx_log_id} was not accessible by keyadmin users. keyadmin users can access the transaction logs using the endpoint /xaudit/trx_log, hence, the keyadmin users should also be allowed to access the endpoint /xaudit/trx_log/{trx_log_id} for transaction log ids related to KMS audits.

What's New in Schema Registry

Learn about the new features of Schema Registry in Cloudera Runtime 7.3.1.

Enable SMM principal as trusted proxy user in Schema Registry

SMM usually connects to Schema Registry on behalf of an end user. For requests coming from SMM, Schema Registry can now extract and authorize the end user to authorize the request.

What's New in Apache Solr

Learn about the new features of Apache Solr in Cloudera Runtime 7.3.1.

Data Discovery and Exploration (Technical Preview)

A new Data Discovery and Exploration cluster definition is available in Cloudera Data Hub. It lets you explore and discover data sets ad-hoc; doing relevance-based analytics over unstructured data (logs, images, text, PDFs, etc). The cluster definition deploys HDFS, Hue, Solr, Spark, Yarn, and ZooKeeper services. The cluster definition is available for AWS.

- ZooKeeper SSL can now be configured for Solr and HBase Indexer. For more information, see Enabling ZooKeeper SSL/TLS for Solr and HBase Indexer.
- Solr now supports rolling upgrades. This means that rolling upgrades are available from release 7.3.1 to higher. Upgrades to release 7.3.1 still involve service downtime.
- Apache Solr is updated from 8.4.1 to 8.11.2 in this release of Cloudera Runtime. For more information, see Apache Solr Release Notes in the upstream documentation. For the list of notable unsupported features, see Unsupported features.
- Using Local File System (LFS) for both MapReduce Indexer Tool (MRIT) and HBase MRIT is now supported.
- Spark 3 is now supported.
- Spark 2 is deprecated in this release and support will be dropped in an upcoming release.
- This release introduces the following two health checks for the Solr service which give information about the status of the cores hosted on different hosts:

  Recovering cores

  By default this check reports concerning health if any of the hosted cores are in recovering status. This threshold can be modified in the configurations with the solr_recovering_core_thresholds confiuration parameter.

  Critical cores

  By default this check reports "bad health" if any of the hosted cores are in down or recovery failed status. This threshold can be modified in the configurations with the solr_critical_core_thresholds config.

  These checks are enabled by default for the Infra Solr service but disabled by default for the Workload Solr services (Cloudera Search).
- Critical CVE fixes.

What's New in Apache Spark

Learn about the new features of Apache Spark in Cloudera Runtime 7.3.1.0.

Spark 2 removed from Cloudera Runtime

Spark 3 is the default Spark version in Cloudera Runtime. Spark 2 has been removed and no longer available in 7.3.1.0

Important:

Spark 3 contains a large number of changes from Spark 2.

Refer to *Upgrading Spark* for more information on upgrading Spark clusters to 7.3.1.0, and *Migrating Spark Applications* for more information on migrating your existing Spark applications between versions 2 and 3.

Related Information

Upgrading Spark
Migrating Spark Applications

What's new in Streams Messaging Manager

Learn about the new features of Streams Messaging Manager (SMM) in Cloudera Runtime 7.3.1.

Validation for duplicate property keys in Kafka Connect connector configuration

When validating Kafka Connect connector configurations, a warning is displayed if the configuration contains duplicate property keys. Duplicate property keys are highlighted with orange. The form can still be validated with the warnings present, but if there are duplicates, you are notified that only the value of the last occurrence is used.

Search supports regular expressions

The search component on the Topics, Brokers, Consumers, Producers page can now perform a regexp search.

Visual clue when restarting on Kafka Connect

When clicking restart on Kafka Connect tasks or connectors, a loading circle is displayed in case of synchronous calls. The loading circle disappears once a response is received. For asynchronous calls, a pop-up is displayed, stating that the task or connector is restarted.

UX improvements

- Fixed text overflow in the side panel column headers
- Listing page table headers are now sticky of the nested table headers
- Listing page table styling has been improved for readability
- Filter selector drop-downs are now styled consistently
- Sidebar menu pop-ups are no longer hidden under tables
- Class names on the Kafka Connect popup are now wrapped into the containing pop-ups
- The password field is no longer obfuscated when using a file provider as a password
- Fixed the alignment of values on the Connector metrics page
- Source and sink connectors are now separate tabs on the connector creation modal
- Fixed visual issues on the topic creation modal
- Increased consistency in element contrast and text style throughout the UI
- Active and Inactive statuses now have high contrast
- The expand icon is now consistent throughout the UI

Expand security-related headers set by SMM

The following security related headers were added to SMM UI endpoints:

- Referrer-Policy
- Cross-Origin-Embedder-Policy

- Cross-Origin-Opener-Policy
- Cross-Origin-Resource-Policy

SMM uses trusted proxy authentication when connecting to Schema Registry

You can only interact with schemas through SMM if the necessary Ranger policies are set up for Schema Registry. For SMM UI, you must have the correct permissions to check messages deserialized with Avro on Data Explorer.

What's New in Apache Hadoop YARN

Learn about the new features of Apache Hadoop Yarn in Cloudera Runtime 7.3.1.

Queue Manager

YARN Queue Manager is the queue management graphical user interface for Apache Hadoop YARN Capacity Scheduler. You can use YARN Queue Manager UI to manage your cluster capacity using queues to balance resource requirements of multiple applications from various users. Using YARN Queue Manager UI, you can set scheduler level properties and queue level properties. You can also view, sort, search, and filter queues using the YARN Queue Manager UI.

For more information about Queue Manager, see Manage Queues.

FPGA as a resource type

You can use FPGA as a resource type. For more information, see Use FPGA scheduling.

New configuration property to enable or disable the YARN recommendation engine APIs

The YARN Recommendation API now recommends scaling cluster nodes up or down based on the demand and idle state of cluster resources. This feature can be turned on/off using the YARN configuration property yarn.cluster.scaling.recommendation.enable.

Unaffected Components in this release

There are no new features for the following components in Cloudera Runtime 7.3.1.

- Avro
- Cruise Control
- Hadoop
- Hive
- HDFS
- Iceberg
- Knox
- Kudu
- Livy
- MapReduce
- Navigator Encrypt
- Ozone
- Parquet
- Ranger KMS
- Sqoop
- Streams Replication Manager (SRM)
- Tez
- Zookeeper

What's new in Platform Support

You must be aware of the platform support for the Cloudera Runtime 7.3.1 release.

This section describes the platform support changes for the Cloudera Runtime 7.3.1 associated with Cloudera Public Cloud 7.3.1.

Platform Support Enhancements

- Default for all new environments:
  - OS support: RHEL 8.10
  - JDK version: JDK 17
  - Database version: PostgreSQL 14
  - Python version: 3.9
- No longer supported:
  - JDK 11 no longer supported (removed)

Cloudera Runtime Component Versions

List of the official component versions for Cloudera Runtime. To know the component versions for compatibility with other applications, you must be familiar with the latest component versions in Cloudera Runtime. You should also be aware of the available Technical Preview components and use them only in a testing environment.

Apache Components

Component	Version
Apache Arrow	0.11.1.7.3.1.0-197
Apache Atlas	2.1.0.7.3.1.0-197
Apache Calcite	1.25.0.7.3.1.0-197
Apache Avatica	1.22.0.7.3.1.0-197
Apache Avro	1.11.1.7.3.1.0-197
Apache Hadoop (Includes YARN and HDFS)	3.1.1.7.3.1.0-197
Apache HBase	2.4.17.7.3.1.0-197
Apache Flink	1.19.1.14.0.0
Apache Hive	3.1.3000.7.3.1.0-197
Apache Iceberg	1.3.1.7.3.1.0-197
Apache Impala	4.0.0.7.3.1.0-197
Apache Kafka	3.4.1.7.3.1.0-197
Apache Knox	2.0.0.7.3.1.0-197
Apache Kudu	1.17.0.7.3.1.0-197
Apache Livy	0.7.23000.7.3.1.0-197
Apache MapReduce	3.1.1.7.3.1.0-197
Apache NiFi	1.28.1.2.2.9.0
Apache NiFi Registry	1.28.1.2.2.9.0
Apache NiFi [Technical Preview]	2.0.0.4.2.1.0

Component	Version
Apache NiFi Registry [Technical Preview]	2.0.0.4.2.1.0
Apache Oozie	5.1.0.7.3.1.0-197
Apache ORC	1.8.3.7.3.1.0-197
Apache Parquet	1.12.3.7.3.1.0-197
Apache Phoenix	5.1.3.7.3.1.0-197
Apache Ranger	2.4.0.7.3.1.0-197
Apache Solr	8.11.2.7.3.1.0-197
Apache Spark	3.4.1.7.3.1.0-197
Apache Sqoop	1.4.7.7.3.1.0-197
Apache Tez	0.9.1.7.3.1.0-197
Apache ZooKeeper	3.8.1.7.3.1.0-197

Other Components

Component	Version
Cruise Control	2.5.116.7.3.1.0-197
Data Analytics Studio	1.4.2.7.3.1.0-197
GCS Connector	2.1.2.7.3.1.0-197
Hue	4.5.0.7.3.1.0-197
Search	1.0.0.7.3.1.0-197
Schema Registry	0.10.0.7.3.1.0-197
Streams Messaging Manager	2.3.0.7.3.1.0-197
Streams Replication Manager	1.1.0.7.3.1.0-197
Data Discovery and Exploration	Technical Preview

Connectors and Encryption Components

Component	Version
HBase connectors	1.0.0.7.3.1.0-197
Hive Meta Store (HMS)	1.0.0.7.3.1.0-197
Hive on Tez	1.0.0.7.3.1.0-197
Hive Warehouse Connector	1.0.0.7.3.1.0-197
Spark Atlas Connector	3.4.1.7.3.1.0-197
Spark Schema Registry	3.4.1.7.3.1.0-197

Note: Cloudera Ozone version 1.3.0 code is equivalent to Apache Ozone 1.4.0 in the 7.3.1 release. However, the version number will be reset in the next release.

Using the Cloudera Runtime Maven repository 7.3.1

Information about using Maven to build applications with Cloudera Runtime components.

If you want to build applications or tools for use with Cloudera Runtime components and you are using Maven or Ivy for dependency management, you can pull the Cloudera Runtime artifacts from the Cloudera Maven repository. The repository is available at https://repository.cloudera.com/artifactory/cloudera-repos/.

Important: When you build an application JAR, do not include CDH JARs, because they are already provided. If you do, upgrading CDH can break your application. To avoid this situation, set the Maven dependency scope to provided. If you have already built applications which include the CDH JARs, update the dependency to set scope to provided and recompile.

The following is a sample POM (pom.xml) file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM
/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <repositories>
    <repository>
      <id>cloudera</id>
      <url>https://repository.cloudera.com/artifactory/cloudera-repos/</url>
    </repository>
  </repositories>
</project>
```

Cloudera Runtime 7.3.1.0-197

The following table lists the project name, groupId, artifactId, and version required to access each RUNTIME artifact.

Project	groupId	artifactId	version
Atlas	org.apache.atlas	atlas-authorization	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-aws-s3-bridge	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-azure-adls-bridge	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-classification-updater	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-client-common	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-client-v1	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-client-v2	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-client-v2-shaded	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-common	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-distro	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-docs	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-graphdb-api	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-graphdb-common	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-graphdb-janus	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-hdfs-bridge	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-index-repair-tool	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-intg	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-janusgraph-hbase2	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-notification	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-plugin-classloader	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-repository	2.1.0.7.3.1.0-197

Project	groupId	artifactId	version
Atlas	org.apache.atlas	atlas-server-api	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	atlas-testtools	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	hbase-bridge	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	hbase-bridge-shim	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	hbase-testing-util	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	hdfs-model	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	hive-bridge	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	hive-bridge-shim	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	impala-bridge	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	impala-bridge-shim	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	impala-hook-api	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	kafka-bridge	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	kafka-bridge-shim	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	navigator-to-atlas	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	sample-app	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	sqoop-bridge	2.1.0.7.3.1.0-197
Atlas	org.apache.atlas	sqoop-bridge-shim	2.1.0.7.3.1.0-197
Avro	org.apache.avro	avro	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-android	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-codegen-test	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-compiler	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-grpc	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-ipc	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-ipc-jetty	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-ipc-netty	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-mapred	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-maven-plugin	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-perf	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-protobuf	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-service-archetype	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-test-custom-conversions	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-thrift	1.11.1.7.3.1.0-197
Avro	org.apache.avro	avro-tools	1.11.1.7.3.1.0-197
Avro	org.apache.avro	trevni-avro	1.11.1.7.3.1.0-197
Avro	org.apache.avro	trevni-core	1.11.1.7.3.1.0-197
Calcite	org.apache.calcite	calcite-babel	1.25.0.7.3.1.0-197
Calcite	org.apache.calcite	calcite-core	1.25.0.7.3.1.0-197
Calcite	org.apache.calcite	calcite-druid	1.25.0.7.3.1.0-197
Calcite	org.apache.calcite	calcite-linq4j	1.25.0.7.3.1.0-197

Project	groupId	artifactId	version
Calcite	org.apache.calcite	calcite-server	1.25.0.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-aliyun	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-annotations	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-archive-logs	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-archives	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-assemblies	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-auth	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-aws	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-azure	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-azure-datalake	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-benchmark	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-build-tools	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-client	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-client-api	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-client-integration-tests	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-client-minicluster	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-client-runtime	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-cloud-storage	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-common	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-datajoin	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-distcp	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-extras	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-fs2img	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-gridmix	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-hdfs	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-hdfs-client	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-hdfs-httpfs	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-hdfs-native-client	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-hdfs-nfs	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-hdfs-rbf	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-kafka	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-kms	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-mapreduce-client-app	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-mapreduce-client-common	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-mapreduce-client-core	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-mapreduce-client-hs	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-mapreduce-client-hs-plugins	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-mapreduce-client-jobclient	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-mapreduce-client-nativetask	3.1.1.7.3.1.0-197

Project	groupId	artifactId	version
Hadoop	org.apache.hadoop	hadoop-mapreduce-client-shuffle	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-mapreduce-client-uploader	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-mapreduce-examples	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-maven-plugins	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-minicluster	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-minikdc	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-nfs	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-openstack	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-resourceestimator	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-rumen	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-sls	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-streaming	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-tools-dist	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-api	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-applications-distributedshell	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-applications-unmanaged-am-launcher	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-client	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-common	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-registry	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-applicationhistoryservice	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-common	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-nodemanager	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-resourcemanager	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-router	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-sharedcachemanager	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-tests	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-timeline-pluginstorage	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-timelineservice	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-timelineservice-hbase-client	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-timelineservice-hbase-common	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-timelineservice-hbase-server-2	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-timelineservice-hbase-tests	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-server-web-proxy	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-services-api	3.1.1.7.3.1.0-197
Hadoop	org.apache.hadoop	hadoop-yarn-services-core	3.1.1.7.3.1.0-197
HBase	org.apache.hbase	hbase-annotations	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-asyncfs	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-checkstyle	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-client	2.4.17.7.3.1.0-197

Project	groupId	artifactId	version
HBase	org.apache.hbase	hbase-client-project	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-common	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-endpoint	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-examples	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-external-blockcache	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-hadoop-compat	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-hadoop2-compat	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-hbtop	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-http	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-it	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-logging	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-mapreduce	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-metrics	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-metrics-api	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-procedure	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-protocol	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-protocol-shaded	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-replication	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-resource-bundle	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-rest	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-rsgroup	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-server	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-shaded-client	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-shaded-client-byo-hadoop	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-shaded-client-project	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-shaded-mapreduce	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-shaded-testing-util	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-shaded-testing-util-tester	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-shell	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-testing-util	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-thrift	2.4.17.7.3.1.0-197
HBase	org.apache.hbase	hbase-zookeeper	2.4.17.7.3.1.0-197
Hive	org.apache.hive	catalogd-unit	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-beeline	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-blobstore	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-classification	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-cli	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-common	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-contrib	3.1.3000.7.3.1.0-197

Project	groupId	artifactId	version
Hive	org.apache.hive	hive-exec	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-hbase-handler	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-hcatalog-it-unit	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-hplsql	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-iceberg-catalog	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-iceberg-handler	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-iceberg-shading	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-impala	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-it-custom-serde	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-it-iceberg	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-it-impala	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-it-minikdc	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-it-qfile	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-it-qfile-kudu	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-it-test-serde	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-it-unit	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-it-unit-hadoop2	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-it-util	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-jdbc	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-jdbc-handler	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-jmh	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-kudu-handler	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-llap-client	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-llap-common	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-llap-ext-client	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-llap-server	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-llap-tez	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-metastore	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-parser	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-pre-upgrade	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-serde	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-service	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-service-rpc	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-shims	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-standalone-metastore	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-storage-api	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-streaming	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-testutils	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	hive-udf	3.1.3000.7.3.1.0-197

Project	groupId	artifactId	version
Hive	org.apache.hive	hive-vector-code-gen	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	kafka-handler	3.1.3000.7.3.1.0-197
Hive	org.apache.hive	patched-iceberg-api	patched-1.3.1.7.3.1.0-197-3.1.3000.7
Hive	org.apache.hive	patched-iceberg-core	patched-1.3.1.7.3.1.0-197-3.1.3000.7
Hive Warehouse Connector	com.hortonworks.hive	hive-warehouse-connector-spark3_2.12	1.0.0.7.3.1.0-197
Kafka	org.apache.kafka	ci	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-api	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-basic-auth-extension	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-cloudera-authorization-extension	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-cloudera-common	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-cloudera-secret-storage	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-cloudera-security-policies	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-file	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-json	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-mirror	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-mirror-client	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-runtime	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	connect-transforms	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	generator	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-clients	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-cloudera-metrics-reporter_2.12	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-cloudera-metrics-reporter_2.13	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-cloudera-plugins	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-examples	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-group-coordinator	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-log4j-appender	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-metadata	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-raft	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-server-common	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-shell	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-storage	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-storage-api	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-examples	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-scala_2.12	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-scala_2.13	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-test-utils	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-0100	3.4.1.7.3.1.0-197

Project	groupId	artifactId	version
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-0101	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-0102	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-0110	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-10	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-11	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-20	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-21	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-22	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-23	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-24	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-25	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-26	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-27	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-28	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-30	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-31	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-32	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-streams-upgrade-system-tests-33	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka-tools	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka_2.12	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	kafka_2.13	3.4.1.7.3.1.0-197
Kafka	org.apache.kafka	trogdor	3.4.1.7.3.1.0-197
Knox	org.apache.knox	gateway-adapter	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-admin-ui	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-applications	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-cloud-bindings	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-demo-ldap	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-demo-ldap-launcher	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-discovery-ambari	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-discovery-cm	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-docker	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-i18n	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-i18n-logging-log4j	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-i18n-logging-sl4j	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-openapi-ui	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-performance-test	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-ha	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-identity-assertion-common	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-identity-assertion-concat	2.0.0.7.3.1.0-197

Project	groupId	artifactId	version
Knox	org.apache.knox	gateway-provider-identity-assertion-hadoop-groups	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-identity-assertion-no-doas	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-identity-assertion-pseudo	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-identity-assertion-regex	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-identity-assertion-switchcase	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-jersey	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-rewrite	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-rewrite-common	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-rewrite-func-hostmap-static	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-rewrite-func-inbound-query-param	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-rewrite-func-service-registry	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-rewrite-step-encrypt-uri	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-rewrite-step-secure-query	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-authc-anon	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-authz-acls	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-authz-composite	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-authz-path-acls	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-clientcert	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-hadoopauth	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-jwt	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-pac4j	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-preauth	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-shiro	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-provider-security-webappsec	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-release	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-server	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-server-launcher	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-server-xforwarded-filter	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-admin	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-as	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-auth	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-definitions	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-hashicorp-vault	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-hbase	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-health	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-hive	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-idbroker	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-idbroker-plugins	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-impala	2.0.0.7.3.1.0-197

Project	groupId	artifactId	version
Knox	org.apache.knox	gateway-service-jkg	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-knoxsso	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-knoxssout	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-knoxtoken	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-livy	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-metadata	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-nifi	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-nifi-registry	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-remoteconfig	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-rm	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-session	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-storm	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-test	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-tgs	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-vault	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-service-webhdfs	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-shell	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-shell-launcher	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-shell-release	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-shell-samples	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-spi	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-spi-common	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-test	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-test-idbroker	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-test-release-utils	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-test-utils	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-topology-hadoop-xml	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-topology-simple	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-util-common	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-util-configinjector	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-util-launcher	2.0.0.7.3.1.0-197
Knox	org.apache.knox	gateway-util-urltemplate	2.0.0.7.3.1.0-197
Knox	org.apache.knox	hadoop-examples	2.0.0.7.3.1.0-197
Knox	org.apache.knox	knox-cli-launcher	2.0.0.7.3.1.0-197
Knox	org.apache.knox	knox-homepage-ui	2.0.0.7.3.1.0-197
Knox	org.apache.knox	knox-token-generation-ui	2.0.0.7.3.1.0-197
Knox	org.apache.knox	knox-token-management-ui	2.0.0.7.3.1.0-197
Knox	org.apache.knox	knox-webshell-ui	2.0.0.7.3.1.0-197
Knox	org.apache.knox	webhdfs-kerb-test	2.0.0.7.3.1.0-197

Project	groupId	artifactId	version
Knox	org.apache.knox	webhdfs-test	2.0.0.7.3.1.0-197
Kudu	org.apache.kudu	kudu-backup-tools	1.17.0.7.3.1.0-197
Kudu	org.apache.kudu	kudu-backup3_2.12	1.17.0.7.3.1.0-197
Kudu	org.apache.kudu	kudu-client	1.17.0.7.3.1.0-197
Kudu	org.apache.kudu	kudu-hive	1.17.0.7.3.1.0-197
Kudu	org.apache.kudu	kudu-spark3-tools_2.12	1.17.0.7.3.1.0-197
Kudu	org.apache.kudu	kudu-spark3_2.12	1.17.0.7.3.1.0-197
Kudu	org.apache.kudu	kudu-test-utils	1.17.0.7.3.1.0-197
Livy	org.apache.livy	livy-api	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-client-common	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-client-http	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-core_2.12	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-examples	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-integration-test	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-repl_2.12	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-rsc	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-scala-api_2.12	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-server	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-test-lib	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-thriftserver	0.7.23000.7.3.1.0-197
Livy	org.apache.livy	livy-thriftserver-session	0.7.23000.7.3.1.0-197
Lucene	org.apache.lucene	lucene-analyzers-common	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-analyzers-icu	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-analyzers-kuromoji	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-analyzers-morfologik	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-analyzers-nori	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-analyzers-opennlp	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-analyzers-phonetic	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-analyzers-smartcn	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-analyzers-stempel	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-backward-codecs	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-benchmark	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-classification	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-codecs	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-core	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-demo	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-expressions	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-facet	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-grouping	8.11.2.7.3.1.0-197

Project	groupId	artifactId	version
Lucene	org.apache.lucene	lucene-highlighter	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-join	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-memory	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-misc	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-monitor	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-queries	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-queryparser	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-replicator	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-sandbox	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-spatial-extras	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-spatial3d	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-suggest	8.11.2.7.3.1.0-197
Lucene	org.apache.lucene	lucene-test-framework	8.11.2.7.3.1.0-197
Oozie	org.apache.oozie	oozie-client	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-core	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-distro	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-examples	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-fluent-job-api	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-fluent-job-client	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-server	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-sharelib-distcp	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-sharelib-git	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-sharelib-hcatalog	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-sharelib-hive	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-sharelib-hive2	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-sharelib-oozie	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-sharelib-spark3	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-sharelib-sqoop	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-sharelib-streaming	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-tools	5.1.0.7.3.1.0-197
Oozie	org.apache.oozie	oozie-zookeeper-security-tests	5.1.0.7.3.1.0-197
ORC	org.apache.orc	orc-core	1.8.3.7.3.1.0-197
ORC	org.apache.orc	orc-examples	1.8.3.7.3.1.0-197
ORC	org.apache.orc	orc-mapreduce	1.8.3.7.3.1.0-197
ORC	org.apache.orc	orc-shims	1.8.3.7.3.1.0-197
ORC	org.apache.orc	orc-tools	1.8.3.7.3.1.0-197
Ozone	org.apache.ozone	hdds-annotation-processing	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-client	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-common	1.3.0.7.3.1.0-197

Project	groupId	artifactId	version
Ozone	org.apache.ozone	hdds-config	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-container-service	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-crypto-api	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-crypto-default	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-docs	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-erasurecode	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-hadoop-dependency-client	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-hadoop-dependency-server	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-hadoop-dependency-test	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-interface-admin	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-interface-client	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-interface-server	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-managed-rocksdb	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-rocks-native	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-server-framework	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-server-scm	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-test-utils	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	hdds-tools	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	mini-chaos-tests	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-client	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-common	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-csi	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-datanode	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-dist	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-filesystem	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-filesystem-common	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-filesystem-hadoop2	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-filesystem-hadoop3	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-filesystem-shaded	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-httpfsgateway	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-insight	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-integration-test	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-interface-client	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-interface-storage	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-manager	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-network-tests	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-recon	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-reconcodegen	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-s3-secret-store	1.3.0.7.3.1.0-197

Project	groupId	artifactId	version
Ozone	org.apache.ozone	ozone-s3gateway	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	ozone-tools	1.3.0.7.3.1.0-197
Ozone	org.apache.ozone	rocksdb-checkpoint-differ	1.3.0.7.3.1.0-197
Parquet	org.apache.parquet	parquet-avro	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-cascading	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-cascading3	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-column	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-common	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-encoding	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-format-structures	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-generator	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-hadoop	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-hadoop-bundle	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-jackson	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-pig	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-pig-bundle	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-protobuf	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-scala_2.12	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-thrift	1.12.3.7.3.1.0-197
Parquet	org.apache.parquet	parquet-tools	1.12.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-client-embedded-hbase-2.4	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-client-hbase-2.4	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-core	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-hbase-compat-2.1.6	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-hbase-compat-2.2.5	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-hbase-compat-2.3.0	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-hbase-compat-2.4.0	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-hbase-compat-2.4.1	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-hbase-compat-2.5.0	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-hbase-compat-2.5.4	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-pherf	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-queryserver	6.0.0.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-queryserver-client	6.0.0.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-queryserver-it	6.0.0.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-queryserver-load-balancer	6.0.0.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-queryserver-orchestrator	6.0.0.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-server-hbase-2.4	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix-tracing-webapp	5.1.3.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix5-hive	6.0.0.7.3.1.0-197

Project	groupId	artifactId	version
Phoenix	org.apache.phoenix	phoenix5-hive-shaded	6.0.0.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix5-spark3	6.0.0.7.3.1.0-197
Phoenix	org.apache.phoenix	phoenix5-spark3-shaded	6.0.0.7.3.1.0-197
Ranger	org.apache.ranger	conditions-enrichers	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	credentialbuilder	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	embeddedwebserver	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	jisql	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ldapconfigcheck	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-adls-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-atlas-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-atlas-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-authn	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-common-ha	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-distro	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-examples-distro	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-gs-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-hbase-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-hbase-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-hdfs-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-hdfs-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-hive-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-hive-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-intg	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-kafka-connect-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-kafka-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-kafka-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-kms	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-kms-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-kms-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-knox-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-knox-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-kudu-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-kylin-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-kylin-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-metrics	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-nifi-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-nifi-registry-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-ozone-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-ozone-plugin-shim	2.4.0.7.3.1.0-197

Project	groupId	artifactId	version
Ranger	org.apache.ranger	ranger-plugin-classloader	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-plugins-audit	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-plugins-common	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-plugins-cred	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-plugins-installer	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-policymigration	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-raz-adls	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-raz-chained-plugins	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-raz-hook-abfs	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-raz-hook-s3	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-raz-intg	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-raz-processor	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-raz-s3	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-raz-s3-lib	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-rms-common	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-rms-hive	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-rms-plugins-common	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-rms-tools	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-rms-webapp	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-s3-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-sampleapp-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-schema-registry-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-solr-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-solr-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-sqoop-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-sqoop-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-storm-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-storm-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-tagsync	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-tools	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-trino-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-util	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-yarn-plugin	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ranger-yarn-plugin-shim	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	sample-client	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	sampleapp	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	shaded-raz-hook-abfs	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	shaded-raz-hook-s3	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	ugsync-util	2.4.0.7.3.1.0-197

Project	groupId	artifactId	version
Ranger	org.apache.ranger	unixauthclient	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	unixauthservice	2.4.0.7.3.1.0-197
Ranger	org.apache.ranger	unixusersync	2.4.0.7.3.1.0-197
Solr	org.apache.solr	solr-analysis-extras	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-analytics	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-cell	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-core	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-dataimporthandler	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-dataimporthandler-extras	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-gcs-repository	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-jaegertracer-configurator	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-langid	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-ltr	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-prometheus-exporter	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-s3-repository	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-security-util	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-solrj	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-test-framework	8.11.2.7.3.1.0-197
Solr	org.apache.solr	solr-velocity	8.11.2.7.3.1.0-197
Spark	org.apache.spark	spark-avro_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-catalyst_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-connect-client-jvm_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-connect-common_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-connect_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-core_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-graphx_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-hadoop-cloud_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-hive_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-kubernetes_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-kvstore_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-launcher_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-mllib-local_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-mllib_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-network-common_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-network-shuffle_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-network-yarn_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-protobuf_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-repl_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-shaded-raz	3.4.1.7.3.1.0-197

Project	groupId	artifactId	version
Spark	org.apache.spark	spark-sketch_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-sql-kafka-0-10_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-sql_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-streaming-kafka-0-10-assembly_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-streaming-kafka-0-10_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-streaming_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-tags_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-token-provider-kafka-0-10_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-unsafe_2.12	3.4.1.7.3.1.0-197
Spark	org.apache.spark	spark-yarn_2.12	3.4.1.7.3.1.0-197
Sqoop	org.apache.sqoop	sqoop	1.4.7.7.3.1.0-197
Sqoop	org.apache.sqoop	sqoop-test	1.4.7.7.3.1.0-197
Tez	org.apache.tez	hadoop-shim	0.9.1.7.3.1.0-197
Tez	org.apache.tez	hadoop-shim-2.8	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-api	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-aux-services	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-common	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-dag	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-examples	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-ext-service-tests	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-history-parser	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-javadoc-tools	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-job-analyzer	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-mapreduce	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-protobuf-history-plugin	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-runtime-internals	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-runtime-library	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-tests	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-yarn-timeline-cache-plugin	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-yarn-timeline-history	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-yarn-timeline-history-with-acls	0.9.1.7.3.1.0-197
Tez	org.apache.tez	tez-yarn-timeline-history-with-fs	0.9.1.7.3.1.0-197
Zeppelin	org.apache.zeppelin	zeppelin-angular	0.8.2.7.3.1.0-197
Zeppelin	org.apache.zeppelin	zeppelin-display	0.8.2.7.3.1.0-197
Zeppelin	org.apache.zeppelin	zeppelin-interpreter	0.8.2.7.3.1.0-197
Zeppelin	org.apache.zeppelin	zeppelin-jdbc	0.8.2.7.3.1.0-197
Zeppelin	org.apache.zeppelin	zeppelin-jupyter	0.8.2.7.3.1.0-197
Zeppelin	org.apache.zeppelin	zeppelin-livy	0.8.2.7.3.1.0-197
Zeppelin	org.apache.zeppelin	zeppelin-markdown	0.8.2.7.3.1.0-197

Project	groupId	artifactId	version
Zeppelin	org.apache.zeppelin	zeppelin-server	0.8.2.7.3.1.0-197
Zeppelin	org.apache.zeppelin	zeppelin-shaded-raz	0.8.2.7.3.1.0-197
Zeppelin	org.apache.zeppelin	zeppelin-shell	0.8.2.7.3.1.0-197
Zeppelin	org.apache.zeppelin	zeppelin-zengine	0.8.2.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper	3.8.1.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper-contrib-fatjar	3.8.1.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper-contrib-loggraph	3.8.1.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper-contrib-rest	3.8.1.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper-contrib-zooinspector	3.8.1.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper-it	3.8.1.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper-jute	3.8.1.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper-prometheus-metrics	3.8.1.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper-recipes-election	3.8.1.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper-recipes-lock	3.8.1.7.3.1.0-197
ZooKeeper	org.apache.zookeeper	zookeeper-recipes-queue	3.8.1.7.3.1.0-197

Fixed Issues In Cloudera Runtime 7.3.1

Fixed issues represent issues reported by Cloudera customers that are addressed in this release.

Fixed Issues in Atlas

Review the list of Atlas issues that are resolved in Cloudera Runtime 7.3.1

CDPD-69962: fetchType as "incremental" does full export instead of "CONNECTED"

Earlier, the first incremental export operation performed on a target entity used to fetch all entities even if they were not related to the targeted entity. This affected the performance as it imports more data than what was expected.

Now, the first incremental export will only fetch the entities which are related to the target entity. Also, if the target entity is connected to a lineage, then only the immediately connected entities in the lineage will get exported and not the whole lineage.

CDPD-67654: [Atlas] [navigator2atlas] Status of deleted table is ACTIVE in Atlas after navigator2atlas migration

Deleted hive tables migrated via the Navigator to Atlas transition may shown as active in Apache Atlas. Changes done in the Nav2Atlas module to set the relationType as hive_table_storagedesc of relationship attribute table for evey entity of hive_storagedesc.

CDPD-72732: [UCL] Incorrect Atlas audits generated for updates with atlas.hook.hive.skip.dml.messages set to true/false in 7.3.0.1 Cloudera Private Cloud Base

The Apache Atlas property atlas.hook.hive.skip.dml.messages = true can be used to reduce the number of audits that are generated for any DML command executed over a hive entity.

The default value for hive.split.update is set to true in 7.3.1 causing two audits to be generated for one update command: one delete and one insert. This will impact Apache Atlas when atlas.hook.hive.skip.dml.messages = false (Atlas is processing Data Manipulation events) and atlas.entity.audit.differential = false (Atlas logs the full entity metadata during every update).

CDPD-71516: Temporarily disable the tasks tab on Entity Detail page

The Entity Detail page was showing "Something went wrong". This is occurring because on loading the Entity Detail page, an API call (`/api/atlas/admin/tasks`) is made to get all the tasks that are created when deferred actions features are enabled. The Entity Detail page task tab and task API will display in UI depending upon the server side property atlas.tasks.ui.tab.enabled. Initially, this is set to false. Therefore, temporarily the task tab on entity detail page in UI is disabled.

Apache Jira:ATLAS-4880

OPSAPS-64385: Atlas's client.auth.enabled configuration is not configurable

In customer environments where user certifications are required to authenticate to services, the Apache Atlas web UI will constantly prompt for certifications. To solve this, the client.auth.enabled parameter is set to true by default. If it is needed to set it false, then you need to override the setting from safety-valve with a configuration snippet. Once it set to false, then no more certificate prompts will be displayed.

OPSAPS-68461: Update GC and JVM options for Atlas service for supporting JDK17 in main Atlas CSD

The issue of existing ATLAS OPTS not working for JDK17 is fixed.

Fixed Issues in Apache Avro

There are no fixed issues for Apache Avro in Cloudera Runtime 7.3.1.

Apache patch information

None

Fixed issues in Cloud Connectors

Review the list of Cloud Connectors issues that are resolved in Cloudera Runtime 7.3.1.

Apache patch information

- HADOOP-18855 - Tuning and stabilization of Vector IO
- MAPREDUCE-7474 - Improve commit resilience and performance in Manifest Committer for ABFS

Fixed issues in Cruise Control

Review the list of Cruise Control issues that are resolved in Cloudera Runtime 7.3.1.

OPSAPS-69978: Cruise Control capacity.py script fails on Python 3

The script querying the capacity information is now fully compatible with Python 3.

Fixed Issues in Hadoop

There are no fixed issues for Hadoop in Cloudera Runtime 7.3.1.

Fixed Issues in HDFS

Review the list of HDFS issues that are resolved in Cloudera Runtime 7.3.1.

OPSAPS-71677: When you are upgrading from Cloudera Private Cloud Base 7.1.9 SP1 to Cloudera Private Cloud Base 7.3.1, upgrade-rollback execution fails during HDFS rollback due to missing directory.

This issue is now resolved. The HDFS meta upgrade command is executed by creating the previous directory due to which the rollback does not fail.

OPSAPS-71390: Cloudera Operational Database cluster creation is failing on INT and displays the Failed to create HDFS directory /tmp error.

This issue is now resolved. Export options for jdk17 is added now.

OPSAPS-71188: Modify default value of dfs_image_transfer_bandwidthPerSec from 0 to a feasible value to mitigate RPC latency in the namenode.

This issue is now resolved.

OPSAPS-58777: HDFS Directories are created with root as user.

This issue is now resolved by fixinf service.sdl.

CDPD-67823: Ranger RMS gives all permissions to the user through the Create permission.

This issue is now resolved. An additional check is added to ensure that the user attempting to alter any HDFS directory that maps to the Hive database is the owner of the Hive database.

Fixed Issues in HBase

Review the list of HBase issues that are resolved in Cloudera Runtime 7.3.1.

CDPD-67520: JWT authentication expects [sub] claim in the payload

A JWT payload can have a custom claim for Subject/Principal instead of the standard sub claim.

You can set the hbase.security.oauth.jwt.token.principal.claim configuration property in Cloudera Manager under HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml to define the custom Subject/Principal claim.

CDPD-66387: RegionServer should be aborted when WAL.sync throws TimeoutIOException

This fix adds additional logic for WAL.sync. If WAL.sync gets a timeout exception, HBase wraps TimeoutIOException as a special WALSyncTimeoutIOException. When the upper layer such as HRegion.doMiniBatchMutate called by HRegion.batchMutation catches this special exception, HBase aborts the region server.

Apache Jira: HBASE-27230

CDPD-65373: Make delay prefetch property dynamically configurable

This change allows you to dynamically configure the hbase.hfile.prefetch.delay property using the Cloudera Manager. You need to update the value and refresh the HBase service. The new value is applied to the HBase service automatically.

Apache Jira: HBASE-28292

CDPD-74494: JVM crashes intermittently on ARM64 machines

After noticing the JVM crashes in the HBase service that is based on arm64 architecture and uses JDK 17, the fix is applied that refactors the module and the large implementation function into multiple smaller functions. The issue was observed in a specific module that had a very large member function.

Apache Jira: HBASE-28206

CDPD-73118: Bucket cache validation fails after a rolling restart resulting in an empty bucket cache without running the prefetch operations

During the retrieval of bucket cache from persistence, it was observed that, if an exception, other than the IOException occurs, the exception is not logged, and also the retrieval thread exits leaving the bucket cache in an uninitialized state, leaving it unusable.

This change enables the retrieval thread to print all types of exceptions and also reinitializes the bucket cache and makes it reusable.

Fixed Issues in Hive

Review the list of Hive issues that are resolved in Cloudera Runtime 7.3.1.

CDPD-57121: ThreadPoolExecutorWithOomHook handling OutOfMemoryError

The ThreadPoolExecutorWithOomHook wasn't effectively handling OutOfMemoryError when executing tasks, as the exception was wrapped in ExecutionException, making it harder to detect.

The issue was fixed by updating ThreadPoolExecutorWithOomHook to properly invoke OutOfMemoryError hooks and stop HiveServer2 when required.

Apache Jira: HIVE-24411, HIVE-26955, IMPALA-8518

Fixed Issues in Hue

Review the list of Hue issues that are resolved in Cloudera Runtime 7.3.1

CDPD-65034: Receiving Error " TypeError: 'NoneType' object is not callable" in TCLIService.py when custom headers are being set

When XSRF (Cross-Site Request Forgery) or CSRF (Cross-Site Request Forgery) is enabled in Hive or Impala, you might encounter the error "Error " TypeError: 'NoneType' object is not callable” in TCLIService.py. You can resolve this issue by upgrading to 7.1.9 SP1 CHF3 or 7.3.1 versions.

Fixed Issues in Impala

Review the list of Impala issues that are resolved in Cloudera Runtime 7.3.1

CDPD-69856: SIGSEGV crash while accessing query state from concurrent access during query execution

A crash can occur due to concurrent updates and reads of execution state, such as through the WebUI, during query processing.

Ensured atomic updates of execution state to prevent conflicts and crashes during concurrent operations.

Apache Jira: IMPALA-12747

Missing txnId in tableWriteIds Mapping during AllocWriteIdEvent Processing

Fix issue of txnId not being added to tableWriteIds mapping in Catalog.

Apache Jira:: IMPALA-12851

CDPD-73442: Resolution of potential deadlock

This fix addresses a deadlock issue in long-running sessions with an active idle_query_timeout, which caused new queries to hang and prevented existing queries from expiring.

Apache Jira: IMPALA-13313

CDPD-71288: Retry HMS fetch failures to keep event-processor active

Metastore event processor enters an error state due to failures in creating a MetaStoreClient or fetching events, which should be retriable instead.

This issue is now fixed.

Apache Jira: IMPALA-12561

Some local file descriptors not released when using remote spilling

The issue that occurred during remote spilling when writing spilled data to local buffers has been fixed. The disk space occupied by the file can now be reclaimed.

Apache Jira:: IMPALA-12681

Handle empty string in StringValue::LargestSmallerString

StringValue::LeastSmallerString() did not account for empty strings, causing potential exceptions by using an invalid length.

The function now checks if the string is empty and returns an empty string if so. The function was also renamed to LargestSmallerString() to clarify its purpose.

Apache Jira:: IMPALA-12478

CDPD-67493 : ALTER_PARTITION self-event detection for partitions created via INSERT

Fix for incorrect identification on self events.

Apache Jira:: IMPALA-12356

CDPD-67912 : Failures in processing AbortTxnEvent

Fixes event processing errors when write IDs of an AbortTxnEvent are cleaned up by the HMS cleaner housekeeping threads.

Apache Jira:: IMPALA-12827

CDPD-67493: Implicit invalidate metadata on event failures

Implicitly invalidates a table instead of resulting in an ERROR state during event processing.

Apache Jira:: IMPALA-12832

IMPALA-12831: HdfsTable.toMinimalTCatalogObject() failed by concurrent modification

Fix race condition when a table is being invalidated and updated concurrently.

Apache Jira:: IMPALA-12831

Release JNI array if DeserializeThriftMsg failed

Fix conditional JVM heap leak in array allocation on deserialization failures.

Apache Jira:: IMPALA-12969

NPE in executing RELOAD events

Fixes the possibility of encountering a NullPointerException when refreshing a partition that has just been dropped.

Apache Jira:: IMPALA-12969

Event processing without hms_event_incremental_refresh_transactional_table

Fix event processor, which is not synching file metadata for non-partitioned ACID tables when incremental refresh on transactional tables is turned off.

Note:

- This issue only occurs when hms_event_incremental_refresh_transactional_table is set to 'false'
- This issue occurs on non-partitioned tables. Partitioned tables are not affected.

Apache Jira:: IMPALA-12835

Fixed Issues in Apache Iceberg

Review the list of iceberg issues that are resolved in Cloudera Runtime 7.3.1

CDPD-48395: Upgrade the Parquet version to 1.12.3 for Hive

This fix upgrades the Parquet version for Hive to 1.12.3, which is the same Parquet version that is used for Iceberg.

Fixed Issues in Kafka

Review the list of Kafka issues that are resolved in Cloudera Runtime 7.3.1.

CDPD-65649: ReplicaAlterLogDirs stuck with Offset mismatch for the future replica

This is a backported fix, see KAFKA-9087 for more information.

CDPD-66986: Mirrormaker 2 auto.offset.reset=latest not working

This is a backported fix, see KAFKA-13988 for more information.

OPSAPS-71258: Kafka, SRM, and SMM cannot process messages compressed with Zstd or Snappy is /tmp is mounted as noexec

The issue is fixed by using JVM flags that point to a different temporary folder for extracting the native library.

CDPD-71433: Connect logical type null values are not handled in AvroConnectTranslator

When the time.precision.mode property is set to connect for the Debezium connector, the connect logical types are used and null values are now handled.

OPSAPS-69481: Some Kafka Connect metrics missing from Cloudera Manager due to conflicting definitions

Cloudera Manager now registers the metrics kafka_connect_connector_task_metrics_batch_size_avg and kafka_connect_connector_task_metrics_batch_size_max correctly.

Fixed Issues in Kudu

Review the list of Kudu issues that are resolved in Cloudera Runtime 7.3.1

KUDU-3576: Fix the Connection timeout After Tablet Server Restart

In Kudu, if a Java client application maintains an open connection to a tablet server and the tablet server is restarted or encounters a network error, the client cannot re-establish communication with the tablet server even after it comes back online. The fix resolves the issue by updating the Kudu Java client.

Apache Jira: KUDU-3576

KUDU-3496: Support for SPNEGO dedicated keytab

Kudu now supports configuring a dedicated Spnego keytab.

Apache Jira: KUDU-3496

KUDU-3524: Fix crash when sending periodic keep-alive requests

The fix ensures that Kudu clients do not crash when sending keep-alive requests.

Apache Jira: KUDU-3524

KUDU-3497: Optimize leader rebalancer algorithm

Optimized the leader balancing algorithm to effectively handle corner cases detailed in the Jira.

Apache Jira: KUDU-3497

KUDU-3447: Limit the usage of network bandwidth of tablet copying

Two new flags are introduced in Kudu CLI tools to copy tablets kudu tablet copy_from_remote command to limit the speed of the copy task and avoid resource contention.

Apache Jira: KUDU-3447

KUDU-3353: Add an immutable attribute to column schema

Introduced Immutable column. It's useful to represent a semantically constant entity.

Apache Jira: KUDU-3353

KUDU-3351: Add insert error count metrics in WriteResponsePB

Statistics on various write operations are now available via Kudu client API at the session level.

Apache Jira: KUDU-3351

KUDU-3526: Scanner should bound with a tserver in java client

The scanner in the Kudu Java client now binds with the Kudu tablet server. This prevents scanning failures that occur when scanning from the leader replica and leadership changes to a different replica.

Apache Jira: KUDU-3526

Fixed Issues in Apache Knox

Review the list of Knox issues that are resolved in Cloudera Runtime 7.3.1.

CDPD-73275: HTTP 404 responses while Knox is redeploying topologies

While you were redeploying topologies, Knox returned HTTP 404 responses.

Knox no longer returns HTTP 404 responses during topology redeployment, but returns HTTP 503 instead.

CDPD-70313: KNOX did not send Authentication header on FIPS configuration

KNOX neither sent the authentication header nor hadoop.auth cookie that was why the SMM UI sent back the HTTP 401 response and set the "www-authenticate": "Negotiate" header. Because of this, the SMM UI was inaccessible through Knox.

This issue is fixed now.

CDPD-60630: Knox redirecting Yarn Node Manager URLs to http instead of https

While viewing the yarn application logs on YARN RM UI via Knox, we can see that Knox is redirecting the NM URL to HTTP instead of HTTPS, as YARN is running on TLS/SSL.

```
https://<knox-gateway>/gateway/cdp-proxy/yarn/nodemanager/node?s
cheme=http&host=some.url&port=8044
```

CDPD-69305: /plugins/policies/importPoliciesFromFile API returns 500 service connectivity error through Knox Proxy

The fix imports large policy files using the Ranger importPoliciesFromFile API through Knox.

Apache patch information

- KNOX-3073
- KNOX-3058
- KNOX-3055
- KNOX-3054
- KNOX-3053
- KNOX-3052
- KNOX-3050
- KNOX-3049
- KNOX-3045
- KNOX-3040
- KNOX-3038
- KNOX-3037
- KNOX-3036
- KNOX-3029
- KNOX-3028

- KNOX-3026
- KNOX-3024
- KNOX-3023
- KNOX-3022
- KNOX-3020
- KNOX-3019
- KNOX-3018
- KNOX-3017
- KNOX-3016
- KNOX-3012
- KNOX-3007
- KNOX-3006
- KNOX-3005
- KNOX-3002
- KNOX-3001
- KNOX-3000
- KNOX-2994
- KNOX-2985
- KNOX-2983
- KNOX-2980
- KNOX-2979
- KNOX-2978
- KNOX-2976
- KNOX-2975
- KNOX-2974
- KNOX-2973
- KNOX-2972
- KNOX-2971
- KNOX-2970
- KNOX-2969
- KNOX-2968
- KNOX-2966
- KNOX-2963
- KNOX-2961
- KNOX-2960
- KNOX-2959
- KNOX-2958
- KNOX-2955
- KNOX-2951
- KNOX-2949
- KNOX-2948
- KNOX-2947
- KNOX-2946
- KNOX-2929
- KNOX-2896
- KNOX-2881

Fixed Issues in Apache Livy

Review the list of Livy issues that are resolved in Cloudera Runtime 7.3.1

OPSAPS-71873 - UCL | CKP4| livyfoo0 kms proxy user is not allowed to access HDFS in 7.3.1.0

In the kms-core.xml file, the Livy proxy user is taken from Livy for Spark 3's configuration in Cloudera Public Cloud version 7.3.1 and above.

CDPD-73324 - LIVY_FOR_SPARK3 goes into down with Invalid Keystore format error in FIPS cluster

Fixed an issue that caused LIVY_FOR_SPARK3 to go into a bad state with Invalid Keystore format error in a 7.3.1 FIPS cluster.

Apache patch information

None

Fixed Issues in Oozie

Review the list of Oozie issues that are resolved in Cloudera Runtime 7.3.1.0.

CDPD-70422: Cannot enforce Oozie parameter oozie.http.hostname

A new property named oozie.http.hostname.override is now introduced to specify the interface that the Oozie Server must be using.

CDPD-71117: Oozie server does not pass action start time to action conf causes a restarting launcher doesn't find child apps

Whenever Yarn restarted the Oozie Launcher AM, Oozie could not find the previously started child jobs due to a missing original start timestamp from the Oozie Server. And the previously started child Jobs were not terminated when the Launcher AM was restarted. This issue is now resolved.

CDPD-48664: Retry mechanism anomaly in Oozie with High Availability enabled

There was an issue with the retry mechanism in Oozie when High Availability was enabled. This issue is now resolved.

CDPD-49745: Expand app_path column in *_JOBS tables to allow HDFS paths longer than 255 characters

The APP_PATH column now supports storing paths longer than 255 characters.

Fixed Issues in Apache Parquet

There are no fixed issues for Parquet in Cloudera Runtime 7.3.1.

Apache Patch Information

None

Fixed Issues in Phoenix

Review the list of Phoenix issues that are resolved in Cloudera Runtime 7.3.1.

OPSAPS-71553: OMID TSO server role fails during cluster installation

High Availability (HA) configuration generation is based on the number of nodes instead of a parameter in OMID.

OPSAPS-70507: ZDU runtime upgrade from 719 CHF4 to 7.3.0.1 fails for OMID

The OMID update issue is fixed.

OPSAPS-69838: OMID cannot connect contact the ZooKeeper cluster

OMID now connects to a secure port of ZooKeeper if TLS is enabled. Earlier it failed to communicate because port unification was missing.

OPSAPS-68583: ZooKeeper SSL/TLS support for OMID

OMID supports a secure connection to ZooKeeper if AutoTLS and ZooKeeper TLS are enabled.

OPSAPS-57949: OMID integration to Cloudera Manager

OMID now supports integration into the Cloudera Manager.

Fixed Issues in Ranger

Review the list of Ranger issues that are resolved in Cloudera Runtime 7.3.1.

CDPD-73663: RMS server threw ConcurrentModificationException

The original ConcurrentModificationException was likely thrown when the resource-mappings were modified in response to changes in the Hive metadata while they were being serialized for downloading to the NameNode (or secondary-namenode).

The fix is to create a shallow copy of resource-mappings before applying deltas which ensures that resource-mappings are not modified while they are being serialized for downloading to the NameNode.

CDPD-73326: Reduce memory needed to create Ranger policy engine

Ranger policy engine creates a RangerPolicyResourceMatcher object for every resource specified either in policy or in a tag association. PolicyResourceMatcher, for the services that have more than one level in their resource hierarchy, consists of RangerResourceMatcher objects for each level in the resource-level hierarchy for the resource. In many cases, this leads to creation of multiple RangerResourceMatchers with identical resource specification.

The fix for this issue avoids creation of multiple RangerResourceMatcher objects by maintaining a cache of them in the RangerPluginContext object associated with the Ranger policy engine, thereby reducing policy engine's memory needs.

CDPD-73144: Trie to support processing of evaluators during traversal

Ranger policy engine uses trie data structure to organize resources for faster retrieval of policies/tags/zones associated with a given resource. When a resource consists of multiple elements, like database/table/column, as many trie instances are consulted to retrieve policies/tags/zones associated with the resource. Such multi-trie retrieval can be optimized with a 2-pass traversal - first pass to get count and the second pass to get the actual objects. Trie data structure used in Ranger policy engine should be updated to support this optimization.

Now, Trie to support processing of evaluators during traversal is enhanced.

CDPD-73102: Access issues for s3 express buckets

Fixed S3 Express bucket access with RAZ enabled in all regions.

CDPD-72203: Users observing role change from ROLE_SYS_ADMIN to ROLE_USER

Fixes role reset (to USER role) for users in usersync paged requests to ranger-admin.

CDPD-71719: Ranger override policy was not working

Ranger override policy was not allowing the access even though all permissions were given to the user.

This fix ensures that once all of the requested accesses are successfully allowed by (possibly multiple) Ranger policies, the access evaluation terminates with access allowed as the result.

CDPD-70081: "Drop database cascade" resulted in dropping of a table on which the user did not have access

Drop database cascade failed if the user did not have access to one or more of the underlying tables. It deleted the tables the user had access to but not others which caused the database to be not dropped as well.

This issue is fixed now.

CDPD-69488: Upgrade failure due to NPE in PatchForUpdatingServiceDefJson_J10058

Patch upgrade error failure in non-default service-def is fixed now.

CDPD-69305: /plugins/policies/importPoliciesFromFile API returns 500 service connectivity error through Knox Proxy

The fix imports large policy files using the Ranger importPoliciesFromFile API through Knox.

CDPD-68921: Exclude flag not taking effect for Ozone key resource in Ranger policy

Fix for exclude flag not taking effect for Ozone key resource in Ranger policy has been added.

CDPD-68853: Create function and Drop function commands are not supported when Ranger plugin is enabled

Support for Create and Drop function commands in Ranger trino plugin has been added.

CDPD-68827: Alter materialized view command is not working when Ranger plugin is enabled

Added support for Alter materialized view command in Ranger trino plugin.

CDPD-68826: Refresh materialized view command is not working when Ranger plugin is enabled

Added support for Refresh materialized view command in Ranger trino plugin.

CDPD-68376: Enable policy and tag deltas for Ranger admin and plugins by default

Policy and tag deltas for Ranger admin and plugins are enabled by default.

CDPD-68238: Update operations are not supported when Ranger plugin is enabled

The fix enables support for the update statement in the Ranger Trino plugin.

CDPD-67823: Ranger RMS gives all permissions to the user through the Create permission

An additional check is now made to ensure that the user attempting to alter a HDFS directory that maps to the Hive database is owner of the Hive database for the attempted operation is allowed.

CDPD-67193: Issue with inactivityTimeout getting reset

The inactivityTimeout was getting reset when a user updated its profile from the UserProfile page.

Fixed issue of not resetting inactivityTimeout to a default value of 15 minutes when user updates its profile from UserProfile page on Ranger Admin UI.

CDPD-66842: Ranger Admin server gives empty response

Ranger Admin server gave an empty response when a user with user-role tried to update lastname or email address.

The issue is fixed now. Error response with message will be shown when a user with user-role tries to add/update last name or email address.

CDPD-66839: Enhance perf-tracer to get CPU time when possible

Ranger module is instrumented with performance measurement code. It enables performance logging for the module and helps in measuring the amount of time spent during execution of various methods/functions during its operation. For achieving more precise time measurement, this feature supports nanosecond precision when the JVM version supports it.

CDPD-66624: Transform URLs with or without “/” at the end issue

The fix enables the transformation step handle “/” at the end of the path.

CDPD-66404: Merging apache ranger jiras for handling local storage data for column show/hide functionality

Implemented Column Hide/Show functionality in Audit Plugin Status tab.

CDPD-66358: HS2 logs having a huge number of WARN logs

HS2 logs had a huge number of WARN logs from RangerHiveAuthorizer regarding connection to HMS for fetching Hive object owner.

This fix addresses the issue where HS2 logs have a huge number of WARN logs.

CDPD-66136: Display of query information for Show databases/schemas command on Ranger Admin UI

In Ranger React UI, if the resource type for certain commands were logged as "null" in the audits, then in the access audits, the information of the query/operations performed would not be displayed.

This ticket addresses the issue and displays the query/operation information for access audits where the resource type was "null".

CDPD-66092: Ranger Javapatch failure even if service-defs do not exist in Ranger DB

Added support to upgrade non-default service-defs in Ranger.

CDPD-65923: Audit logs for Mask and Row policy does not show policy condition under policy item

The fix now shows policy conditions under policy items for Mask and Row policy Audit logs.

CDPD-65650: Pagination missing on the Ranger Admin - Plugin Status page

This fix offers the following:

- Sorting works properly after this patch.
- Pagination added.

CDPD-63891: Backport the ranger-trino changes from upstream to downstream

Trino support in Ranger has been added.

OPSAPS-70838: Flink user should be add by default in ATLAS_HOOK topic policy in Ranger >> cm_kafka

The "flink" service user is granted publish access on the ATLAS_HOOK topic by default in the Kafka Ranger policy configuration.

OPSAPS-69411: Update AuthzMigrator GBN to point to latest non-expired GBN

Users will now be able to export sentry data only for given Hive objects (databases and tables and the respective URLs) by using the config "authorization.migration.export.migration_objects" during export.

OPSAPS-68252: "Ranger RMS Database Full Sync" option was not visible on mow-int cluster setup for hrt_qa user (7.13.0.0)

The fix makes the command visible on cloud clusters when the user has minimum EnvironmentAdmin privilege.

Apache Patch information

- RANGER-4973
- RANGER-4972
- RANGER-4960
- RANGER-4933
- RANGER-4912
- RANGER-4905
- RANGER-4893
- RANGER-4833
- RANGER-4823
- RANGER-4819
- RANGER-4818
- RANGER-4802
- RANGER-4799
- RANGER-4798
- RANGER-4797
- RANGER-4796
- RANGER-4791
- RANGER-4786
- RANGER-4782

- RANGER-4781
- RANGER-4780
- RANGER-4774
- RANGER-4767
- RANGER-4753
- RANGER-4747
- RANGER-4745
- RANGER-4737
- RANGER-4729
- RANGER-4722
- RANGER-4720
- RANGER-4718
- RANGER-4717
- RANGER-4710
- RANGER-4699
- RANGER-4698
- RANGER-4690
- RANGER-4689
- RANGER-4688
- RANGER-4681
- RANGER-4673
- RANGER-4668
- RANGER-4653
- RANGER-4641
- RANGER-4611
- RANGER-4609
- RANGER-4607
- RANGER-4598
- RANGER-4597
- RANGER-4596
- RANGER-4595
- RANGER-4594
- RANGER-4593
- RANGER-4591
- RANGER-4590
- RANGER-4589
- RANGER-4588
- RANGER-4586
- RANGER-4578
- RANGER-4577
- RANGER-4576
- RANGER-4575
- RANGER-4574
- RANGER-4573
- RANGER-4568
- RANGER-4555
- RANGER-4554
- RANGER-4553
- RANGER-4552
- RANGER-4551

- RANGER-4550
- RANGER-4549
- RANGER-4548
- RANGER-4547
- RANGER-4546
- RANGER-4545
- RANGER-4544
- RANGER-4532
- RANGER-4515
- RANGER-4513
- RANGER-4492
- RANGER-4370
- RANGER-4303
- RANGER-4278
- RANGER-4261
- RANGER-4229
- RANGER-4221
- RANGER-4172
- RANGER-4010
- RANGER-3805
- RANGER-3772
- RANGER-3759
- RANGER-3745
- RANGER-3657
- RANGER-3182
- RANGER-3174

Fixed Issues in Schema Registry

Review the list of Schema Registry issues that are resolved in Cloudera Runtime 7.3.1.

OPSAPS-68708: Schema Registry might fail to start if a load balancer address is specified in Ranger

Schema Registry now always ensures that the address it uses to connect to Ranger ends with a trailing slash (/). As a result, Schema Registry no longer fails to start if Ranger has a load balancer address configured that does not end with a trailing slash.

Fixed Issues in Apache Solr

Review the list of Solr issues that are resolved in Cloudera Runtime 7.3.1.

OPSAPS-71690: Update control group V2 configuration parameters

The default values of the control group (CGroup) V2 configuration parameters are updated in Cloudera Manager for the Solr service. The following table describes the default values of the corresponding V2 parameters.

Parameter name	Default values
memory.high	-1
memory.max	-1
io.weight	100
cpu.weight	100

For more information on CGroup V2 parameters, see Configuring Resource Parameters.

Fixed Issues in Spark

Review the list of Spark issues that are resolved in Cloudera Runtime 7.3.1.0.

CDPD-74697 - Spark Iceberg vectorized Parquet read of decimal column is incorrect

CDPD-72774 - Use common versions of commons-dbcp2 and commons-pool2

CDPD-70114 - Redirect spark-submit, spark-shell etc. scripts to their Spark 3 counterparts

CDPD-58844 - Spark - Upgrade Janino to 3.1.10 due to CVE-2023-33546

CDPD-48171 - Spark3 - Upgrade snakeyaml due to CVE-2022-1471

Apache patch information

None

Fixed Issues in Streams Messaging Manager

Review the list of Streams Messaging Manager (SMM) issues that are resolved in Cloudera Runtime 7.3.1.

OPSAPS-71258: Kafka, SRM, and SMM cannot process messages compressed with Zstd or Snappy if /tmp is mounted as noexec

The issue is fixed by using JVM flags that point to a different temporary folder for extracting the native library.

CDPD-72543: Security headers are not set for static files in SMM

SMM now applies the following security-related headers to static files:

- Content-Security-Policy
- X-XSS-PROTECTION
- X-Content-Type-Options
- X-Frame-Options
- Strict-Transport-Security

CDPD-73643: Unused CM_USER parameter is visible in /cm-configs internal endpoint

The unused CM_USER field has been removed from the /cm-configs internal endpoint

CDPD-70313: KNOX does not send Authentication header on FIPS configuration

KNOX now sends the Authentication header on FIPS clusters.

Fixed Issues in Streams Replication Manager

Review the list of Streams Replication Manager (SRM) issues that are resolved in Cloudera Runtime 7.3.1.

OPSAPS-71258: Kafka, SRM, and SMM cannot process messages compressed with Zstd or Snappy if /tmp is mounted as noexec

The issue is fixed by using JVM flags that point to a different temporary folder for extracting the native library.

Fixed Issues in Apache Tez

There are no fixed issues for Tez in Cloudera Runtime 7.3.1.

Fixed Issues in YARN and YARN Queue Manager

Review the list of YARN and YARN Queue Manager issues that are resolved in Cloudera Runtime 7.3.1.

COMPX-17702: Backport - YARN-10345 - HsWebServices containerlogs does not honor ACLs for completed jobs

The following rest APIs now have ACL authorization:

- /ws/v1/history/containerlogs/{containerid}/{filename}
- /ws/v1/history/containers/{containerid}/logs

Apache Jira: YARN-10345

COMPX-16285: Optimize system credentials sent in node heartbeat responses

Previously, the heartbeat responses set all application's tokens even though all applications were not active on a node. Hence, for each node and each heartbeat too many SystemCredentialsFor AppsProto objects were created. This issue is now resolved and the system credentials sent in node heartbeat responses are optimized..

Apache Jira: YARN-6523

CDPD-73754: Yarn Application Master Node web link is broken on yarnuiv2 page

Previously, the RM did not open the Yarn application manager node web link on the **yarnuiv2** page because the URL ended with a /. This issue is now resolved and the last character / is now removed from the URL.

Apache Jira: YARN-11729

Fixed Issues in Zookeeper

Review the list of ZooKeeper issues that are resolved in Cloudera Runtime 7.3.1.

CDPD-67821: Zookeeper - Information disclosure in persistent watcher handling(CVE-2024-23944)

There was information disclosure in persistent watchers handling in Apache ZooKeeper due to CVE-2024-23944. This issue is now fixed.

Apache Jira: ZOOKEEPER-4799

CDPD-66977: Backport ZOOKEEPER-4804 Use daemon threads for Netty client

Previously, when the Netty client was used, the Java process did not respond on System.exit when the Zookeeper connection was open. This issue was caused by the non-daemon threads created by Netty. This issue is now resolved.

Apache Jira: ZOOKEEPER-4804

Known Issues In Cloudera Runtime 7.3.1

This topic describes known issues and workarounds in this release of Cloudera Runtime.

Known Issues in Apache Atlas

Learn about the known issues in Atlas, the impact or changes to the functionality, and the workaround.

CDPD-67450: Table name renaming operation is not updating or creating iceberg_table entity

Renaming an Iceberg table does not update the corresponding Atlas entity.

None.

CDPD-59413: Plugin is not supported with older Atlas server versions for Iceberg tables

Copy the model file 1130-iceberg_table_model.json to the directory: /opt/cloudera/parcels/CDH/lib/atlas/models/1000-Hadoop.

Proceed to restart the Atlas Service using Cloudera Manager.

CDPD-56590: Create table "like" from Iceberg table creates a hive_table instead of iceberg_table

By default, for tables created using the "like" command, lineage is not generated in Atlas. The destination like table should be of the same type as source table. Instead an iceberg_table for source and hive_table for destination are getting created.

CDPD-56085: [Impala Iceberg] LOAD DATA INPATH to Iceberg_table creates a temporary hive_table with name <iceberg_table_name>_tmp* and then marks it as DELETED in Atlas

Running a query like "LOAD DATA INPATH to iceberg_table", creates a temporary hive_table with name <iceberg_table_name>_tmp* and then marks it as DELETED in Atlas. So in Atlas, a deleted entity is created corresponding to the temporary table "<iceberg_table_name>_tmp*".

Tag added to the File system (HDFS) entity will not be propagated to the Iceberg table, user has to manually add to the iceberg_table, since the tag propagation is broken due to the deleted table in the flow.

CDPD-67112: Import transforms do not work as expected when replacing a string which already has ":"

None

CDPD-65806: After upgrading from Cloudera Runtime 7.2.17 to 7.2.18, not all Iceberg table relationships are visible in the entity details page

None

CDPD-62973: Change in audits behavior in Cloudera Runtime 7.2.18 deployment.

When the value of differential audits is set as true, the audit information is not segregated based on the user which is firing the query. The HMS service user information includes details of the service user. When differential audit is enabled, only the difference between the two subsequent audits is logged, but in this case, there is no change in the data which is retrieved from HS2 and HMS, which does not create the audit. The user information is audited fine when differential audit is disabled

CDPD-63397: During Data Lake upgrade, Atlas authorization is denied

When rolling upgrade is performed, there might be a scenario where Ranger Admin could be undergoing upgrade by itself and hence the policy download could be affected.

During this period, access might be denied for certain Atlas entities. This issue is resolved once Ranger Admin is up and the policies are downloaded.

CDPD-55301: The ddlQueries and ALTERTABLE_* lineage are missing for Spark tables created using spark3-shell

The ddlQueries and outputFromProcesses (lineage) is missing for the alter queries.

CDPD-40346: The ddlQueries and ALTERTABLE_ADDCOLS lineage missing for Impala tables

The ALTERTABLE_ADDCOLS lineage has some issue when an Impala table is altered and the corresponding lineage is not created.

CDPD-55671: When one Atlas server host is not reachable (stopped), the GET request does multiple failover for approximately 4 minutes and takes around 2 minutes for every failover and finally the request fails.

None

CDPD-55122: Any user with ssh access can view the downloaded results

None

CDPD-45642: When REST Notification server is down, messages from hooks are lost

None

CDPD-46940: REST notification need to be disabled when running import scripts

None

ATLAS-3921: Currently there is no migration path from AWS S3 version 1 to AWS S3 version 2

None

CDPD-11941: Table creation events missed when multiple tables are created in the same Hive command

When multiple Hive tables are created in the same database in a single command, the Atlas audit log for the database may not capture all the table creation events. When there is a delay between creation commands, audits are created as expected.

None

CDPD-11940: Database audit record misses table delete

When a hive_table entity is created, the Atlas audit list for the parent database includes an update audit. However, at this time, the database does not show an audit when the table is deleted.

None

CDPD-11692: Navigator table creation time not converted to Atlas

In converting content from Navigator to Atlas, the create time for Hive tables is not moved to Atlas.

None

CDPD-11338: Cluster names with upper case letters may appear in lower case in some process names

Atlas records the cluster name as lower case in qualifiedNames for some process names. The result is that the cluster name may appear in lower case for some processes (insert overwrite table) while it appears in upper case for other queries (ctas) performed on the same cluster.

None

CDPD-10576: Deleted Business Metadata attributes appear in Search Suggestions

Atlas search suggestions continue to show Business Metadata attributes even if the attributes have been deleted.

None

CDPD-10574: Suggestion order doesn't match search weights

At this time, the order of search suggestions does not honor the search weight for attributes.

None

CDPD-9095: Duplicate audits for renaming Hive tables

Renaming a Hive table results in duplicate ENTITY_UPDATE events in the corresponding Atlas entity audits, both for the table and for its columns.

None

CDPD-7982: HBase bridge stops at HBase table with deleted column family

Bridge importing metadata from HBase fails when it encounters an HBase table for which a column family was previously dropped. The error indicates:

```
Metadata service API org.apache.atlas.AtlasClientV2$API_V2@58112
bc4 failed with status 404 (Not Found) Response Body
({""errorCode"":""ATLAS-404-00-007"",""errorMessage"":""Invalid
 instance creation/updation parameters passed :
hbase_column_family.table: mandatory attribute value missing in
type hbase_column_family""})
```

None

CDPD-7781: TLS certificates not validated on Firefox

Atlas is not checking for valid TLS certificates when the UI is opened in FireFox browsers.

None

CDPD-6675: Irregular qualifiedName format for Azure storage

The qualifiedName for hdfs_path entities created from Azure blog locations (ABFS) doesn't have the clusterName appended to it as do hdfs_path entities in other location types.

None

CDPD-4762: Spark metadata order may affect lineage

Atlas may record unexpected lineage relationships when metadata collection from the Spark Atlas Connector occurs out of sequence from metadata collection from HMS. For example, if an ALTER TABLE operation in Spark changing a table name and is reported to Atlas before HMS has processed the change, Atlas may not show the correct lineage relationships to the altered table.

None

CDPD-4545: Searches for Qualified Names with "@" doesn't fetch the correct results

When searching Atlas qualifiedName values that include an "at" character (@), Atlas does not return the expected results or generate appropriate search suggestions.

Consider leaving out the portion of the search string that includes the @ sign, using the wildcard character * instead.

CDPD-3208: Table alias values are not found in search

When table names are changed, Atlas keeps the old name of the table in a list of aliases. These values are not included in the search index in this release, so after a table name is changed, searching on the old table name will not return the entity for the table.

None

CDPD-3160: Hive lineage missing for INSERT OVERWRITE queries

Lineage is not generated for Hive INSERT OVERWRITE queries on partitioned tables. Lineage is generated as expected for CTAS queries from partitioned tables.

None

CDPD-3125: Logging out of Atlas does not manage the external authentication

At this time, Atlas does not communicate a log-out event with the external authentication management, Apache Knox. When you log out of Atlas, you can still open the instance of Atlas from the same web browser without re-authentication.

To prevent access to Atlas after logging out, close all browser windows and exit the browser.

CDPD-1892: Ranking of top results in free-text search not intuitive

The Free-text search feature ranks results based on which attributes match the search criteria. The attribute ranking is evolving and therefore the choice of top results may not be intuitive in this release.

If you don't find what you need in the top 5 results, use the full results or refine the search.

CDPD-1884: Free text search in Atlas is case sensitive

The free text search bar in the top of the screen allows you to search across entity types and through all text attributes for all entities. The search shows the top 5 results that match the search terms at any place in the text (*term* logic). It also shows suggestions that match the search terms that begin with the term (term* logic). However, in this release, the search results are case-sensitive.

If you don't see the results you expect, repeat the search changing the case of the search terms.

CDPD-1823: Queries with ? wildcard return unexpected results

DSL queries in Advanced Search return incorrect results when the query text includes a question mark (?) wildcard character. This problem occurs in environments where trusted proxy for Knox is enabled, which is always the case for Cloudera.

None

CDPD-1664: Guest users are redirected incorrectly

Authenticated users logging in to Atlas are redirected to the Cloudera Knox-based login page. However, if a guest user (without Atlas privileges) attempts to log in to Atlas, the user is redirected instead to the Atlas login page.

To avoid this problem, open the Atlas Dashboard in a private or incognito browser window.

CDPD-922: IsUnique relationship attribute not honored

The Atlas model includes the ability to ensure that an attribute can be set to a specific value in only one relationship entity across the cluster metadata. For example, if you wanted to add metadata tags to relationships that you wanted to make sure were unique in the system, you could design the relationship attribute with the property "IsUnique" equal true. However, in this release, the IsUnique attribute is not enforced.

None

CDPD-65619: Newly created Iceberg tables do not show up under hive_db entity

When creating hive_table as well as iceberg_table, only the tables with the hive_table typename are shown under the Tables tab. Both Iceberg and Hive tables cannot be shown when they are created in the same hive_db entity.

CDPD-75994: Post DL regular upgrade (non ZDU) to 7.3.1, "Exception in getKafkaConsumer ,WakeupException: null" is seen

After the data lake is upgraded to 7.3.1, sometimes Atlas Hook does not function when Apache Atlas and Apache Kafka are started at the same time, thus Atlas is unable to connect to Kafka while Kafka is still being set up. Atlas performs only three attempts.

Restart the cluster, after the upgrade to trigger to reconnect to Apache Kafka. The Kafka consumer creation should be retried if the Kafka service is unavailable during Atlas startup.

CDPD-76035: Resource lookup for Atlas service is failing

Once the Atlas configuration snippet atlas.authentication.method.file is enabled and a classification is created, these do not synchronize correctly to the Type Category resource field setting of Apache Ranger. The newly created classification won't be able to be selected as the Type Name.

CDPD-74180: Export/Import : If Shell entities have a lineage, it is not exported

If there is a shell entity which has lineage, while using the Export API, that shell entity will not be exported in the zip file.

CDPD-66938: [Analyze] [Atlas] [FIPS] test_time_range tests fail

When the Apache Atlas server is running on a node which has time zone other than UTC, there might be a time of day when the search results might differ if the relative **CreateTime** date range filters of TODAY, YESTERDAY, etc. are used.

Use explicit date range filters instead of using relative date range filters, such as, TODAY, YESTERDAY.

CDPD-70321: Atlas Parallel import is failing with various errors

During a parallel import-export activity with six iceberg table policies with exportOption as db1.* for all six exports, all import fail after the exports.

CDPD-76269: POST Rolling Upgrade performed from 7.1.7.3000 to 7.3.1.0 , and then downgraded from 7.3.1.0 to 7.1.7.3000 , updating edge to enable tag propagation is failing

When performing a rolling upgrade from 7.1.7.3000 to 7.3.1.0, and then downgrading from 7.3.1.0 to 7.1.7.3000, updating edge to enable tag propagation fails.

Tag propagation works well after a restart.

CDPD-76536: Iceberg entities created in 7.2.17 Atlas-Hook (Hive, Impala, Spark) are not ingested by 7.2.18/7.3.1 Atlas Server

Cloudera Data Warehouse version 2024.0.18.2-4 has the Cloudera Data Hub version 7.2.17 which supports an older Iceberg model for the Atlas hook. Apache Atlas in 7.3.1 data lake already supports the latest Iceberg model causing incompatibilities.

Known Issues in Apache Avro

Learn about the known issues in Avro, the impact or changes to the functionality, and the workaround.

CDPD-23451: Avro library depends on the already EOL jackson-mapper-asl 1.9.13-cloudera.1 that also contains a couple of CVEs. The jackson library is part of the Avro API so cannot be changed without a complete rebase.

None.

Known Issues in Cloud Connectors

Learn about the known issues in Cloud Connectors, the impact or changes to the functionality, and the workaround.

AWS SDK 2.25.53 warning about transfer manager not using CRT client

Due to the AWS SDK 2.25.53 upgrade, the following warning might be seen:

```
5645:2024-09-13 16:29:17,375 [setup] WARN  s3.S3TransferManager
          (LoggerAdapter.java:warn(225)) - The provided S3AsyncC
lient is an instance of
          MultipartS3AsyncClient, and thus multipart download fe
ature is not enabled. To benefit
          from all features, consider using S3AsyncClient.crtBu
ilder().build() instead
```

This error message is completely harmless and should be ignored. For more information, see HADOOP-19272.

None

Known issues in Cruise Control

Learn about the known issues in Cruise Control, the impact or changes to the functionality, and the workaround.

CDPD-44676: Rebalancing with Cruise Control does not work if the metric reporter fails to report the CPU usage metric

If the CPU usage metric is not reported, the numValidWindows in Cruise Control will be 0 and proposal generation as well as partition rebalancing will not work. If this issue is present, the following message will be included in the Kafka logs:

```
WARN com.linkedin.kafka.cruisecontrol.metricsreporter.CruiseCont
rolMetricsReporter:
       [CruiseControlMetricsReporterRunner]: Failed reporting CPU
 util.
```

```
java.io.IOException: Java Virtual Machine recent CPU usage is not
 available.
```

This issue is only known to affect Kafka broker hosts that have the following specifications:

- CPU: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
- OS: Linux 4.18.5-1.el7.elrepo.x86_64 #1 SMP Fri Aug 24 11:35:05 EDT 2018 x86_64
- Java version: 8-18

Move the broker to a different machine where the CPU is different. This can be done by moving the host to a different cluster. For more information, see Moving a Host Between Clusters

Note: Cluster nodes affected by this issue are not displayed as unhealthy.

Known Issues in Apache Hadoop

There are no known issues for Hadoop in Cloudera Runtime 7.3.1.

Known Issues in Apache HBase

Learn about the known issues in HBase, the impact or changes to the functionality, and the workaround.

CDPD-60862: Rolling restart fails during ZDU when DDL operations are in progress

During a Zero Downtime Upgrade (ZDU), the rolling restart of services that support Data Definition Language (DDL) statements might fail if DDL operations are in progress during the upgrade. As a result, ensure that you do not run DDL statements during ZDU.

The following services support DDL statements:

- Impala
- Hive – using HiveQL
- Spark – using SparkSQL
- HBase
- Phoenix
- Kafka

Data Manipulation Lanaguage (DML) statements are not impacted and can be used during ZDU. Following the successful upgrade, you can resume running DDL statements.

None. Cloudera recommends modifying applications to not use DDL statements for the duration of the upgrade. If the upgrade is already in progress, and you have experienced a service failure, you can remove the DDLs in-flight and resume the upgrade from the point of failure.

OpDB Data Hub cluster fails to initialize if you are reusing a cloud storage location that was used by an older OpDB Data Hub cluster

Workaround: Stop HBase using Cloudera Manager before deleting an OpDB Data Hub cluster.

IntegrationTestReplication fails if replication does not finish before the verify phase begins

During IntegrationTestReplication, if the verify phase starts before the replication phase finishes, the test will fail because the target cluster does not contain all of the data. If the HBase services in the target cluster does not have enough memory, long garbage-collection pauses might occur.

Workaround: Use the -t flag to set the timeout value before starting verification.

HDFS encryption with HBase

Cloudera has tested the performance impact of using HDFS encryption with HBase. The overall overhead of HDFS encryption on HBase performance is in the range of 3 to 4% for both read and update workloads. Scan performance has not been thoroughly tested.

Workaround: N/A

Snappy compression with /tmp directory mounted with noexec option

Using the HBase client applications such as hbase hfile on the cluster with Snappy compression could result in UnsatisfiedLinkError.

Add -Dorg.xerial.snappy.tempdir=/var/hbase/snappy-tempdir to Client Java Configuration Options in Cloudera Manager that points to a directory where exec option is allowed.

AccessController postOperation problems in asynchronous operations

When security and Access Control are enabled, the following problems occur:

- If a Delete Table fails for a reason other than missing permissions, the access rights are removed but the table may still exist and may be used again.

- If hbaseAdmin.modifyTable() is used to delete column families, the rights are not removed from the Access Control List (ACL) table. The portOperation is implemented only for postDeleteCo lumn().
- If Create Table fails, full rights for that table persist for the user who attempted to create it. If another user later succeeds in creating the table, the user who made the failed attempt still has the full rights.

Workaround: N/A

Apache Issue: HBASE-6992

HBase shutdown can lead to inconsistencies in META

Cloudera Manager uses an incorrect shutdown command. This prevents graceful shutdown of the HBase service and forces Cloudera Manager to kill the processes instead. It can lead to inconsistencies in Meta.

Workaround: Run the following command instead of shutting down the HBase service using Cloudera Manager.

```
hbase master stop --shutDownCluster
```

The command output must end with Closing master protocol: MasterService phrase. You can verify the command execution by checking the master logs. The log must contain Cluster shutdown req uested of master=xxx and the closing of regions. Upon successful execution, the RegionServers start shutting down.

Note: The command does not stop the *REST Server* and the *Thrift Server* role instances. You can safely shut down them from Cloudera Manager later.

If you find any inconsistencies, please contact Cloudera Support.

Bulk load is not supported when the source is the local HDFS

The bulk load feature (the completebulkload command) is not supported when the source is the local HDFS and the target is an object store, such as S3/ABFS.

Workaround: Use distcp to move the HFiles from HDFS to S3 and then run bulk load from S3 to S3.

Apache Issue: N/A

Storing Medium Objects (MOBs) in HBase is currently not supported

Storing MOBs in HBase relies on bulk loading files, and this is not currently supported when HBase is configured to use cloud storage (S3).

Workaround: N/A

Apache Issue: N/A

Known Issues in HDFS

Learn about the known issues in HDFS, the impact or changes to the functionality, and the workaround.

CDPD-65530: HDFS requests throw UnknownHostException during OS upgrade

During the VM replacement as part of OS upgrade, every new node gets a new IP Address, and if the old IP address is cached somewhere, HDFS requests fail with UnknownHostException and it recovers after sometime (10 mins max).

The issue is seen during COD and DL ZDU.

None.

CDPSDX-5302: Avoiding long delay on the HBase master does not happen during upgrade.

1. Log in to Cloudera Manager
2. Select the HDFS service
3. Select Configurations tab
4. Search for hdfs-site.xml.
5. Set ipc.client.connect.timeout = 5000
6. Set ipc.client.connect.max.retries.on.timeouts = 5
7. Click Save

The above configuration changes ensures that:

1. The long delay on the HBase master does not happen during upgrade.
2. The long delay on the HBase master recovery does not happen during upgrade.

CDPD-67230: Rolling restart can cause failed writes on small clusters

In a rolling restart, if the cluster has less than 10 datanodes existing writers can fail with an error indicating a new block cannot be allocated and all nodes are excluded. This is because you have attempted to use all the datanodes in the cluster, and failed to write to each of them as they were restarted. This only happen on small clusters of less than 10 datanodes, because larger clusters have more spare nodes to allow the write to continue.

None.

CDPD-60873: java.io.IOException:Encountered "status=ERROR, status message, ack with firstBadLink" while fixing the HDFS corrupt file during rollback.

Increase the value of dfs.client.block.write.retries to the number of nodes in the cluster and perform Deploy client configuration procedure for rectification.

CDPD-60431: Configuration difference between 7.1.7 SP2 and 7.1.9.0 results

Component	Configuration	Old Value	New Value	Description
HDFS	dfs.permissions.ContentSummary.subAccess	Not Set	True	Performance optimization for NameNode content summary API
HDFS	dfs.datanode.handler.count	3	10	Optimal value for DN server threads on large clusters

None.

CDPD-60387: Configuration difference between 7.1.8.3 and 7.1.9.0 results

Component	Configuration	Old Value	New Value	Description
HDFS	dfs.namenode.accesstime.precision	Not set	0	Optimal value for NameNode performance on large clusters
HDFS	dfs.datanode.handler.count	3	10	Optimal value for DN server threads on large clusters

None.

OPSAPS-64307: When the JournalNodes on a cluster are restarted, the Add new NameNode wizard for HDFS service might fail to bootstrap the new NameNode. If there was no new fsImage created from the time JournalNodes restarted, during the restart the edit logs were rolled in the system.

If the bootstraping fails during the Add new NameNode wizard, then perform the following steps:

1. Delete the newly added NameNode and FailoverController
2. Move the active HDFS NameNode to safe mode
3. Perform the Save Namespace operation on the active HDFS NameNode
4. Leave safe mode on the active HDFS NameNode
5. Add the new NameNode again

Note: Entering safe mode disables writes to HDFS which causes a service disruption. If you cannot enter the safe mode, delete the newly added NameNode and FailoverController in the HDFS service and wait until HDFS automatically creates a new fsImage and then add the new NameNode again with the wizard.

OPSAPS-64363: Deleting of additional Standby Namenode does not delete the ZKFC role and this has to be done manually.

None.

CDPD-28390: Rolling restart of the HDFS JournalNodes may time out on Ubuntu20.

If the restart operation times out, you can manually stop and restart the Name Node and Journal Node services one by one.

OPSAPS-55788: WebHDFS is always enabled. The Enable WebHDFS option does not take effect.

None.

OPSAPS-63299: Disable HA command for a nameservice does not work if the nameservice has more than 2 NameNodes defined.

None.

OPSAPS-63301: Deleting nameservice command does not delete all the NameNodes belonging to the nameservice, if there are more than two NameNodes that are assigned to the nameservice.

None.

Unsupported features

The following HDFS features are currently not supported in Cloudera:

- ACLs for the NFS gateway (HADOOP-11004)
- Aliyun Cloud Connector (HADOOP-12756)
- Allow HDFS block replicas to be provided by an external storage system (HDFS-9806)
- Consistent standby Serving reads (HDFS-12943)
- Cost-based RPC FairCallQueue (HDFS-14403)
- HDFS Router Based Federation (HDFS-10467)
- NameNode Federation (HDFS-1052)
- NameNode Port-based Selective Encryption (HDFS-13541)
- Non-Volatile Storage Class Memory (SCM) in HDFS Cache Directives (HDFS-13762)
- OpenStack Swift (HADOOP-8545)
- SFTP FileSystem (HADOOP-5732)
- Storage policy satisfier (HDFS-10285)

Technical Service Bulletins

TSB 2022-549: Possible HDFS Erasure Coded (EC) data loss when EC blocks are over-replicated

Cloudera has detected a bug that can cause loss of data that is stored in HDFS Erasure Coded (EC) files in an unlikely scenario.

Some EC blocks may be inadvertently deleted due to a bug in how the NameNode chooses excess or over-replicated block replicas for deletion. One possible cause of over-replication is running the HDFS balancer soon after a NameNode goes into failover mode.

In a rare situation, the redundant blocks can be placed in such a way that one replica is in one rack, and few redundant replicas are in the same rack. Such placement causes a counting bug (HDFS-16420) to be triggered. Instead of deleting just the redundant replicas, the original replica may also be deleted.

Usually this is not an issue, because the lost replica can be detected and reconstructed from the remaining data and parity blocks. However, if multiple blocks in an EC Block Group are affected by this counting bug within a short time, the block cannot be reconstructed anymore. For example, 4 blocks are affected out of 9 for the RS(6,3) policy.

Another situation is recommissioning multiple nodes back into the same rack of the cluster where the current live replica exists.

Upstream JIRA

HDFS-16420

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2022-549: Possible HDFS Erasure Coded (EC) data loss when EC blocks are over-replicated

Known Issues in Apache Hive

Learn about the known issues in Hive, the impact or changes to the functionality, and the workaround.

DAG not retried after failure

When executing a Hive query, if the ApplicationMaster container fails, Hive does not retry the DAG if the failure message contains some diagnostic information including a line break, leading to query failure (instead of retry).

None

Known Issues in Hue

Learn about the known issues in Hue, the impact or changes to the functionality, and the workaround.

CDPD-58978: Batch query execution using Hue fails with Kerberos error

When you run Impala queries in a batch mode, you enounter failures with a Kerberos error even if the keytab is configured correctly. This is because submitting Impala, Sqoop, Pig, or pyspark queries in a batch mode launches a shell script Oozie job from Hue and this is not supported on a secure cluster.

There is no workaround. You can submit the queries individually.

CDPD-54376: Clicking the home button on the File Browser page redirects to HDFS user directory

When you are previewing a file on any supported filesystem, such as S3 or ABFS, and you click on the Home button, you are redirected to the HDFS user home directory instead of the user home directory on the said filesystem.

None.

CDPD-43293: Unable to import Impala table using Importer

Creating Impala tables using the Hue Importer may fail.

If you have both Hive and Impala services installed on your cluster, then you can import the table using by selecting the Hive dialect from Tables Sources .

If only Impala service is installed on your cluster, then go to Cloudera Manager Clusters Hue Configurations and add the following line in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[beeswax]
max_number_of_sessions=1
```

CDPD-41136: Importing files from the local workstation is disabled by default

Cloudera has disabled the functionality to import files from your local workstation into Hue because it may cause errors. You may not see the Local File option in the Type drop-down menu on the **Importer** page by default.

You can enable the functionality to import files from your local workstation by specifying the following parameter in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field in Hue configurations in Cloudera Manager:

```
[indexer]
enable_direct_upload=true
```

INSIGHT-3707: Query history displays "Result Expired" message

You see the "Result Expired" message under the Query History column on the **Queries** tab for queries which were run back to back. This is a known behaviour.

None.

CDPD-64541, CDPD-63617: Creating managed tables using Hue Importer fails on RAZ-enabled GCP environments

On Google Cloud Platform (GCP) environments, creating managed tables in both Hive and Impala dialects fails and temporary (tmp) tables are dumped (created). This is most likely because Hive and Impala cannot load data inpath from Google Storage (outside of Hue).

None.

CDPD-56888: Renaming a folder with special characters results in a duplicate folder with a new name on AWS S3.

On AWS S3, if you try to rename a folder with special characters in its name, a new folder is created as a copy of the original folder with its contents. Also, you may not be able to delete the folder containing special characters.

You can rename or delete a directory having special characters using the HDFS commands as follows:

1. SSH into your Cloudera environment host.
2. To delete a directory within your S3 bucket, run the following command:

   ```
   hdfs dfs -rm -r [***COMPLETE-PATH-TO-S3-BUCKET***]/[***DIREC
   TORY-NAME***]
   ```

3. To rename a folder, create a new directory and run the following command to move files from the source directory to the target directory:

   ```
   hdfs dfs -mkdir [***DIRECTORY-NAME***]
   ```

   ```
   hdfs dfs -mv [***COMPLETE-PATH-TO-S3-BUCKET***]/[***SOURCE-D
   IRECTORY***] [***COMPLETE-PATH-TO-S3-BUCKET***]/[***TARGET-D
   IRECTORY***]
   ```

CDPD-48146: Error while browsing S3 buckets or ADLS containers from the left-assist panel

You may see the following error while trying to access the S3 buckets or ADLS containers from the left-assist panel in Hue: Failed to retrieve buckets: :1:0: syntax error.

Access the S3 buckets or ADLS containers using the File Browser.

CDPD-41136: Importing files from the local workstation is disabled by default

Cloudera has disabled the functionality to import files from your local workstation into Hue because it may cause errors. You may not see the Local File option in the Type drop-down menu on the Importer page by default.

You can enable the functionality to import files from your local workstation by specifying the following parameter in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field using Cloudera Manager:

```
[indexer]
enable_direct_upload=true
```

CDPD-42619: Unable to import a large CSV file from the local workstation

You may see an error message while importing a CSV file into Hue from your workstation, stating that you cannot import files of size more than 200 KB.

Upload the file to S3 or ABFS and then import it into Hue using the Importer.

Hue Importer is not supported in the Data Engineering template

When you create a Cloudera Data Hub cluster using the Cloudera Data Engineering template, the Importer application is not supported in Hue:

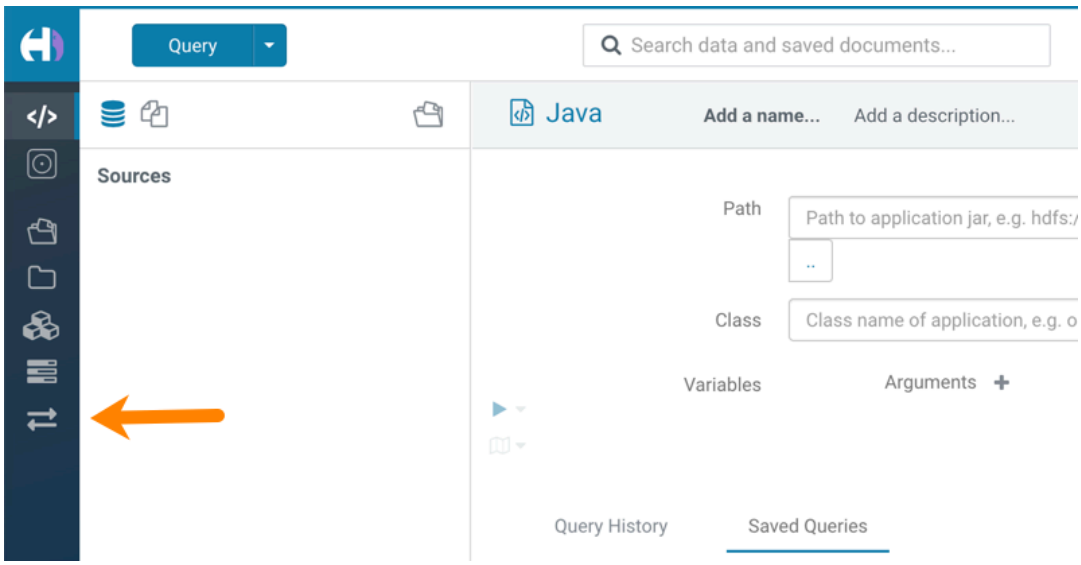

Unsupported features

CDPD-59595: Spark SQL does not work with all Livy servers that are configured for High Availability

SparkSQL support in Hue with Livy servers in HA mode is not supported. Hue does not automatically connect to one of the Livy servers. You must specify the Livy server in the Hue Advanced Configuration Snippet as follows:

```
[desktop]
[spark]
livy_server_url=http(s)://[***LIVY-FOR-SPARK3-SERVER-HOST***]:
[***LIVY-FOR-SPARK3-SERVER-PORT***]
```

Moreover, you may see the following error in Hue when you submit a SparkSQL query: Expecting value: line 2 column 1 (char 1). This happens when the Livy server does not respond to the request from Hue.

Specify all different Livy servers in the livy_server_url property one at a time and use the one which does not cause the issue.

Importing and exporting Oozie workflows across clusters and between different CDH versions is not supported

You can export Oozie workflows, schedules, and bundles from Hue and import them only within the same cluster if the cluster is unchanged. You can migrate bundle and coordinator jobs with their workflows only if their arguments have not changed between the old and the new cluster. For example, hostnames, NameNode, Resource Manager names, YARN queue names, and all the other parameters defined in the workflow.xml and job.properties files.

Using the import-export feature to migrate data between clusters is not recommended. To migrate data between different versions of CDH, for example, from CDH 5 to Cloudera 7, you must take the dump of the Hue database on the old cluster, restore it on the new cluster, and set up the database in the new environment. Also, the authentication method on the old and the new cluster should be the same because the Oozie workflows are tied to a user ID, and the exact user ID needs to be present in the new environment so that when a user logs into Hue, they can access their respective workflows.

Note: Migrating Oozie workflows from HDP clusters is not supported.

Known Issues in Apache Iceberg

Learn about the known issues in Iceberg, the impact or changes to the functionality, and the workaround.

CDPD-75667: Querying an Iceberg table with a TIMESTAMP_LTZ column can result in data loss

When you query an Iceberg table that has a TIMESTAMP_LTZ column, the query could result in data loss.

When creating Iceberg tables from Spark, set the following Spark configuration to avoid creating columns with the TIMESTAMP_LTZ type:

```
spark.sql.timestampType=TIMESTAMP_NTZ
```

Apache JIRA: IMPALA-13484

CDPD-75088: Iceberg tables in azure cannot be partitioned by strings ending in '.'

In an Azure environment, you cannot create Iceberg tables from Spark that are partitioned by string columns having a partition value that contains the period (.) character. The query fails with the following error:

```
24/10/08 18:14:12 WARN  scheduler.TaskSetManager: [task-result-g
etter-2]: Lost task 0.0 in stage 2.0 (TID 2) (spark-sfvq0t-compu
te0.spark-r9.l2ov-m7vs.int.cldr.work executor 1): java.lang.Ille
galArgumentException: ABFS does not allow files or directories to
 end with a dot.
```

None.

CDPD-72942: Unable to read Iceberg table from Hive after writing data through Apache Flink

If you create an Iceberg table with default values using Hive and insert data into the table through Apache Flink, you cannot then read the Iceberg table from Hive using the Beeline client, and the query fails with the following error:

```
Error while compiling statement: java.io.IOException: java.io.IO
Exception: Cannot create an instance of InputFormat class org.ap
ache.hadoop.mapred.FileInputFormat as specified in mapredWork!
```

The issue persists even after you use the ALTER TABLE statement to set the engine.hive.enabled table property to "true".

None.

Apache JIRA: HIVE-28525

CDPD-71962: Hive cannot write to a Spark Iceberg table bucketed by date column

If you have used Spark to create an Iceberg table that is bucketed by the "date" column and then try inserting or updating this Iceberg table using Hive, the query fails with the following error:

```
Error: Error while compiling statement: FAILED: RuntimeException
 org.apache.hadoop.hive.ql.exec.UDFArgumentException:  ICEBERG_B
UCKET() only takes STRING/CHAR/VARCHAR/BINARY/INT/LONG/DECIMAL/F
LOAT/DOUBLE types as first argument, got DATE (state=42000,code=
40000)
```

This issue does not occur if the Iceberg table is created through Hive.

None.

CDPD-66305: Do not turn on the optimized Iceberg V2 operator

The optimized Iceberg V2 operator is disabled by default due to a correctness issue. The correct setting for the property that turns off the operator is DISABLE_OPTIMIZED_ICEBERG_V2_READ=true.

Accept the default setting of the V2 operator. Do not change the setting from true to false.

CDPD-64629: Performance degradation of Iceberg tables compared to Hive tables

Cloudera testing of Iceberg and Hive tables using the Hive TPC-DS 1 Tb dataset (Parquet) revealed a slower performance executing a few of the queries in TPCDS. Overall performance of Iceberg executing queries on Hive external tables of Iceberg is faster than Hive.

CDPD-57551: Performance issue can occur on reads after writes of Iceberg tables

Hive might generate too many small files, which causes performance degradation.

Maintain a relatively small number of data files under the iceberg table/partition directory to have efficient reads. To alleviate poor performance caused by too many small files, run the following queries:

```
TRUNCATE TABLE target;
INSERT OVERWRITE TABLE target select * from target FOR SYSTEM_VER
SION AS OF <preTruncateSnapshotId>;
```

Known Issues in Apache Impala

Learn about the known issues in Impala, the impact or changes to the functionality, and the workaround.

IMPALA-532: Impala should tolerate bad locale settings

If the LC_* environment variables specify an unsupported locale, Impala does not start.

Add LC_ALL="C" to the environment settings for both the Impala daemon and the Statestore daemon.

IMPALA-691: Process mem limit does not account for the JVM's memory usage

Some memory allocated by the JVM used internally by Impala is not counted against the memory limit for the impalad daemon.

To monitor overall memory usage, use the top command, or add the memory figures in the Impala web UI /memz tab to JVM memory usage shown on the /metrics tab.

IMPALA-635: Avro Scanner fails to parse some schemas

The default value in Avro schema must match type of first union type, e.g. if the default value is null, then the first type in the UNION must be "null".

Swap the order of the fields in the schema specification. For example, use ["null", "string"] instead of ["string", "null"]. Note that the files written with the problematic schema must be rewritten with the new schema because Avro files have embedded schemas.

IMPALA-1024: Impala BE cannot parse Avro schema that contains a trailing semi-colon

If an Avro table has a schema definition with a trailing semicolon, Impala encounters an error when the table is queried.

Remove trailing semicolon from the Avro schema.

IMPALA-1652: Incorrect results with basic predicate on CHAR typed column

When comparing a CHAR column value to a string literal, the literal value is not blank-padded and so the comparison might fail when it should match.

Use the RPAD() function to blank-pad literals compared with CHAR columns to the expected length.

IMPALA-1821: Casting scenarios with invalid/inconsistent results

Using a CAST() function to convert large literal values to smaller types, or to convert special values such as NaN or Inf, produces values not consistent with other database systems. This could lead to unexpected results from queries.

None

IMPALA-2005: A failed CTAS does not drop the table if the insert fails

If a CREATE TABLE AS SELECT operation successfully creates the target table but an error occurs while querying the source table or copying the data, the new table is left behind rather than being dropped.

Drop the new table manually after a failed CREATE TABLE AS SELECT

IMPALA-3509: Breakpad minidumps can be very large when the thread count is high

The size of the breakpad minidump files grows linearly with the number of threads. By default, each thread adds 8 KB to the minidump size. Minidump files could consume significant disk space when the daemons have a high number of threads.

Add -\-minidump_size_limit_hint_kb=size to set a soft upper limit on the size of each minidump file. If the minidump file would exceed that limit, Impala reduces the amount of information for each thread from 8 KB to 2 KB. (Full thread information is captured for the first 20 threads, then 2 KB per thread after that.) The minidump file can still grow larger than the "hinted" size. For example, if you have 10,000 threads, the minidump file can be more than 20 MB.

IMPALA-4978: Impala requires FQDN from hostname command on Kerberized clusters

The method Impala uses to retrieve the host name while constructing the Kerberos principal is the gethostname() system call. This function might not always return the fully qualified domain name, depending on the network configuration. If the daemons cannot determine the FQDN, Impala does not start on a Kerberized cluster.

Test if a host is affected by checking whether the output of the `hostname` command includes the FQDN. On hosts where `hostname`, only returns the short name, pass the command-line flag ##hostname=*FULLY_QUALIFIED_DOMAIN_NAME* in the startup options of all Impala-related daemons.

IMPALA-6671: Metadata operations block read-only operations on unrelated tables

Metadata operations that change the state of a table, like COMPUTE STATS or ALTER RECOVE R PARTITIONS, may delay metadata propagation of unrelated unloaded tables triggered by statements like DESCRIBE or SELECT queries.

None

IMPALA-7072: Impala does not support Heimdal Kerberos

None

CDPD-28139: Set spark.hadoop.hive.stats.autogather to false by default

As an Impala user, if you submit a query against a table containing data ingested using Spark and you are concerned about the quality of the query plan, you must run COMPUTE STATS against such a table in any case after an ETL operation because numRows created by Spark could be incorrect. Also, use other stats computed by COMPUTE STATS, e.g., Number of Distinct Values (NDV) and NULL count for good selectivity estimates.

For example, when a user ingests data from a file into a partition of an existing table using Spark, if spark.hadoop.hive.stats.autogather is not set to false explicitly, numRows associated with this partition would be 0 even though there is at least one row in the file. To avoid this, the workaround is to set "spark.hadoop.hive.stats.autogather=false" in the "Spark Client Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-defaults.conf" in Spark's CM Configuration section.

IMPALA-2422: % escaping does not work correctly when occurs at the end in a LIKE clause

If the final character in the RHS argument of a LIKE operator is an escaped \% character, it does not match a % final character of the LHS argument.

None

IMPALA-2603: Crash: impala::Coordinator::ValidateCollectionSlots

A query could encounter a serious error if includes multiple nested levels of INNER JOIN clauses involving subqueries.

None

IMPALA-3094: Incorrect result due to constant evaluation in query with outer join

An OUTER JOIN query could omit some expected result rows due to a constant such as FALSE in another join clause. For example:

```
explain SELECT 1 FROM alltypestiny a1
  INNER JOIN alltypesagg a2 ON a1.smallint_col = a2.year AND fals
e
  RIGHT JOIN alltypes a3 ON a1.year = a1.bigint_col;
+-\-\-\
```

Impala known limitation when querying compacted tables

When the compaction process deletes the files for a table from the underlying HDFS location, the Impala service does not detect the changes as the compactions does not allocate new write ids. When the same table is queried from Impala it throws a 'File does not exist' exception that looks something like this:

```
Query Status: Disk I/O error on <node>:22000: Failed to open HDF
S file hdfs://nameservice1/warehouse/tablespace/managed/hive/<da
tabase>/<table>/xxxxx
Error(2): No such file or directory Root cause: RemoteException:
 File does not exist: /warehouse/tablespace/managed/hive/<data
base>/<table>/xxxx
```

Use the REFRESH/INVALIDATE statements on the affected table to overcome the 'File does not exist' exception.

TSB 2021-502: Impala logs the session / operation secret on most RPCs at INFO level

Impala logs contain the session / operation secret. With this information a person who has access to the Impala logs might be able to hijack other users' sessions. This means the attacker is able to execute statements for which they do not have the necessary privileges otherwise. Impala deployments where Apache Sentry or Apache Ranger authorization is enabled may be vulnerable to privilege escalation. Impala deployments where audit logging is enabled may be vulnerable to incorrect audit logging.

Restricting access to the Impala logs that expose secrets will reduce the risk of an attack. Additionally, restricting access to trusted users for the Impala deployment will also reduce the risk of an attack. Log redaction techniques can be used to redact secrets from the logs. For more information, see the *Cloudera Manager documentation*.

For log redaction, users can create a rule with a search pattern: secret \(string\) [=:].*And the replacement could be for example: secret=LOG-REDACTED

This vulnerability is fixed upstream under IMPALA-10600

Severity

7.5 (High) CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H

Releases affected

- Cloudera Private Cloud Base 7.0.3, 7.1.1, 7.1.2, 7.1.3, 7.1.4, 7.1.5 and 7.1.6
- Cloudera Public Cloud 7.0.0, 7.0.1, 7.0.2, 7.1.0, 7.2.0, 7.2.1, 7.2.2, 7.2.6, 7.2.7, and 7.2.8
- All CDH 6.3.4 and lower releases

Impact

Unauthorized access

Users affected

Impala users of the affected releases

Action required

Upgrade to a Cloudera Private Cloud Base or Cloudera Public Cloud version containing the fix.

Addressed in patch/release/hotfix

- Cloudera Private Cloud Base 7.1.7
- Cloudera Public Cloud 7.2.9 or higher versions

Knowledge article

For the latest update on this issue see the corresponding Knowledge article: TSB 2021-502: Impala logs the session / operation secret on most RPCs at INFO level

HADOOP-15720: Queries stuck on failed HDFS calls and not timing out

In Impala 3.2 and higher, if the following error appears multiple times in a short duration while running a query, it would mean that the connection between the impalad and the HDFS NameNode is in a bad state.

```
"hdfsOpenFile() for <filename> at backend <hostname:port> failed
 to finish before the <hdfs_operation_timeout_sec> second timeout
 "
```

In Impala 3.1 and lower, the same issue would cause Impala to wait for a long time or not respond without showing the above error message.

Restart the impalad.

IMPALA-5605: Configuration to prevent crashes caused by thread resource limits

Impala could encounter a serious error due to resource usage under very high concurrency. The error message is similar to:

```
F0629 08:20:02.956413 29088 llvm-codegen.cc:111] LLVM hit fatal
 error: Unable to allocate section memory!
terminate called after throwing an instance of 'boost::exception_
detail::clone_impl<boost::exception_detail::error_info_injector<
boost::thread_resource_error> >'
```

To prevent such errors, configure each host running an `impalad` daemon with the following settings:

```
echo 2000000 > /proc/sys/kernel/threads-max
echo 2000000 > /proc/sys/kernel/pid_max
echo 8000000 > /proc/sys/vm/max_map_count
```

Add the following lines in /etc/security/limits.conf:

```
impala soft nproc 262144
impala hard nproc 262144
```

IMPALA-9350: Ranger audit logs for applying column masking policies missing

Impala is not producing these logs.

None

IMPALA-1792: ImpalaODBC: Can not get the value in the SQLGetData(m-x th column) after the SQLBindCol(m th column)

If the ODBC SQLGetData is called on a series of columns, the function calls must follow the same order as the columns. For example, if data is fetched from column 2 then column 1, the SQLGetDa ta call for column 1 returns NULL.

Fetch columns in the same order they are defined in the table.

Technical Service Bulletins

TSB 2021-479: Impala can return incomplete results through JDBC and ODBC clients in all Cloudera offerings

In Cloudera, we introduced a timeout on queries to Impala defaulting to 10 seconds. The timeout setting is called FETCH_ROWS_TIMEOUT_MS. Due to this setting, JDBC, ODBC, and Beeswax clients running Impala queries believe the data returned at 10 seconds is a complete dataset and present it as the final output. However, in cases where there are still results to return after this timeout has passed, when the driver closes the connection, based on the timeout, it results in a scenario where the query results are incomplete.

Upstream JIRA

IMPALA-7561

Impact

Potential incorrect query results, due to incomplete dataset.

Action required

- **Upgrade (recommended)**

This is fixed in the newest versions of the Impala JDBC driver and the Impala ODBC driver, available at the following locations:

- Impala ODBC 2.6.12 - https://www.cloudera.com/downloads/connectors/impala/odbc/2-6-12.html
- Impala JDBC 2.6.20 - https://www.cloudera.com/downloads/connectors/impala/jdbc/2-6-20.html
- **Workaround**

Set the property "FETCH_ROWS_TIMEOUT_MS" to 0 if you are unable to use one of the newer versions of the respective drivers listed above. This way, the client can fetch the complete set of data without any issues. Setting the timeout to 0 effectively turns the fetch call into a blocking request which will not timeout and will wait till all the results are fetched. It will wait until all the results come through and/or the network layer timeouts.

This can be set at the Impala server level (via Impala Daemon Query Options safety valve), or in a pool used with Admission Control, or at the session level, or at the query level.

- On the command line, it can be changed at the server level as follows:

```
bin/start-impala-cluster.py
                                --impalad_args="--default_quer
y_options=fetch_rows_timeout_ms=0"
```

- To set the query option at the user session level: SET fetch_rows_timeout_ms=0;

- To set it in admission controls pools via Cloudera Manager:
  1. Click Clusters Impala Admission Control Configuration .

     You will see a screen similar to the following:

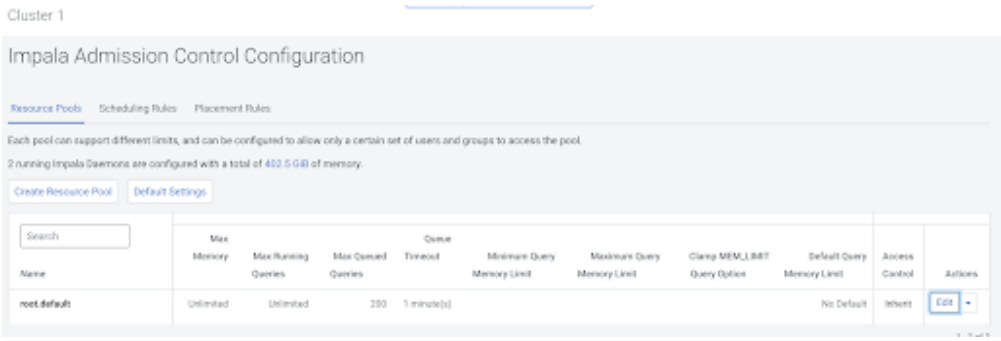

  2. Click Edit and scroll down to Default Query Options.
  3. Click + and add the query option FETCH_ROWS_TIMEOUT_MS = 0
- In Cloudera Data Warehouse, users can configure their Impala VW by the following steps:
  1. Click the three dots and choose the Edit option.
  2. Select the Impala coordinator tab within the Configurations tab.
  3. Select flagfile in the drop-down menu.
  4. Find the default_query_options key and add the following to the end of the value string: FETCH_ROWS_TIMEOUT_MS = 0. Make sure to add a comma before adding this to the value string.

Knowledge article

For the latest update on this issue, see the corresponding Knowledge article: TSB-2021 479: Impala can return incomplete results through JDBC and ODBC clients in all Cloudera offerings

TSB 2022-543: Impala query with predicate on analytic function may produce incorrect results

Apache Impala may produce incorrect results for a query which has all of the following conditions:

- There are two or more analytic functions (for example, row_number()) in an inline view
- Some of the functions have partition-by expression while the others do not
- There is a predicate on the inline view's output expression corresponding to the analytic function

Upstream JIRA

IMPALA-11030

Impact

Incorrect results returned from certain Impala queries.

Action required

- **Preferred Solution/Upgrade**

Please contact Cloudera Support for raising a Hotfix request until a release with the fix is available.

- **Workaround**

None

Knowledge article

For the latest update on this issue, see the corresponding Knowledge article: TSB 2022-543: Impala query with predicate on analytic function may produce incorrect results

Known Issues in Apache Kafka

Learn about the known issues in Kafka, the impact or changes to the functionality, and the workaround.

Known Issues

OPSAPS-59553: SMM's bootstrap server config should be updated based on Kafka's listeners

SMM does not show any metrics for Kafka or Kafka Connect when multiple listeners are set in Kafka.

Workaround: SMM cannot identify multiple listeners and still points to bootstrap server using the default broker port (9093 for SASL_SSL). You need to override the bootstrap server URL by performing the following steps:

1. In Cloudera Manager, go to SMM Configuration Streams Messaging Manager Rest Admin Server Advanced Configuration Snippet (Safety Valve)
2. Override bootstrap server URL (hostname:port as set in the listeners for broker) for streams-messaging-manager.yaml.
3. Save your changes.
4. Restart SMM.

The offsets.topic.replication.factor property must be less than or equal to the number of live brokers

The offsets.topic.replication.factor broker configuration is now enforced upon auto topic creation. Internal auto topic creation will fail with a GROUP_COORDINATOR_NOT_AVAILABLE error until the cluster size meets this replication factor requirement.

None

Requests fail when sending to a nonexistent topic with auto.create.topics.enable set to true

The first few produce requests fail when sending to a nonexistent topic with auto.create.topics.e nable set to true.

Increase the number of retries in the producer configuration setting retries.

KAFKA-2561: Performance degradation when SSL Is enabled

In some configuration scenarios, significant performance degradation can occur when SSL is enabled. The impact varies depending on your CPU, JVM version, Kafka configuration, and message size. Consumers are typically more affected than producers.

Configure brokers and clients with ssl.secure.random.implementation = SHA1PRNG. It often reduces this degradation drastically, but its effect is CPU and JVM dependent.

CDPD-45183: Kafka Connect active topics might be visible to unauthorised users

The Kafka Connect active topics endpoint (/connectors/*[***CONNECTOR NAME***]*/topics) and the Connect Cluster page on the SMM UI disregard the user permissions configured for the Kafka service in Ranger. As a result, all active topics of connectors might become visible to users who do not have permissions to view them. Note that user permission configured for Kafka Connect in Ranger are not affected by this issue and are correctly applied.

None.

RANGER-3809: Idempotent Kafka producer fails to initialize due to an authorization failure

Kafka producers that have idempotence enabled require the Idempotent Write permission to be set on the cluster resource in Ranger. If permission is not given, the client fails to initialize and an error similar to the following is thrown:

```
org.apache.kafka.common.KafkaException: Cannot execute transacti
onal method because we are in an error state
              at org.apache.kafka.clients.producer.internals.Tr
ansactionManager.maybeFailWithError(TransactionManager.java:1125)
              at org.apache.kafka.clients.producer.internals.T
ransactionManager.maybeAddPartition(TransactionManager.java:442)
              at org.apache.kafka.clients.producer.KafkaProduce
r.doSend(KafkaProducer.java:1000)
              at org.apache.kafka.clients.producer.KafkaProduc
er.send(KafkaProducer.java:914)
```

```
                at org.apache.kafka.clients.producer.KafkaProducer
.send(KafkaProducer.java:800)
                .
                .
                .
                Caused by: org.apache.kafka.common.errors.Cluste
rAuthorizationException: Cluster authorization failed.
```

Idempotence is enabled by default for clients in Kafka 3.0.1, 3.1.1, and any version after 3.1.1. This means that any client updated to 3.0.1, 3.1.1, or any version after 3.1.1 is affected by this issue.

This issue has two workarounds, do either of the following:

- Explicitly disable idempotence for the producers. This can be done by setting enable.idempotence to false.
- Update your policies in Ranger and ensure that producers have Idempotent Write permission on the cluster resource.

CDPD-49304: AvroConverter does not support composite default values

AvroConverter cannot handle schemas containing a STRUCT type default value.

None.

DBZ-4990: The Debezium Db2 Source connector does not support schema evolution

The Debezium Db2 Source connector does not support the evolution (updates) of schemas. In addition, schema change events are not emitted to the schema change topic if there is a change in the schema of a table that is in capture mode. For more information, see DBZ-4990.

None.

CFM-3532: The Stateless NiFi Source, Stateless NiFi Sink, and HDFS Stateless Sink connectors cannot use Snappy compression

This issue only affects Stateless NiFi Source and Sink connectors if the connector is running a dataflow that uses a processor that uses Hadoop libraries and is configured to use Snappy compression. The HDFS Stateless Sink connector is only affected if the Compression Codec or Compression Codec for Parquet properties are set to SNAPPY.

If you are affected by this issue, errors similar to the following will be present in the logs.

```
Failed to write to HDFS due to java.lang.UnsatisfiedLinkError: o
rg.apache.hadoop.util.NativeCodeLoader.buildSupportsSnappy()
```

```
Failed to write to HDFS due to java.lang.RuntimeException: nativ
e snappy library not available: this version of libhadoop was bu
ilt without snappy support.
```

Download and deploy missing libraries.

Important: Ensure that you complete steps 1-11 on all Kafka Connect hosts. Additionally, ensure that the advanced configuration snippet in step 12 is configured for all Kafka Connect role instances.

1. Create the /opt/nativelibs directory.

```
mkdir /opt/nativelibs
```

2. Change the owner to kafka.

```
chown kafka:kafka /opt/nativelibs
```

3. Locate the directory containing the Hadoop native libraries and copy its contents to the directory you created.

```
cp /opt/cloudera/parcels/CDH/lib/hadoop/lib/native/* /opt/na
tivelibs
```

4. Verify that libsnappy.so was copied to the directory you created.
5. Remove the following from /opt/nativelibs.

```
libhadoop.a
                  libhadoop.so
                  libhadoop.so.1.0.0
```

6. Run the following command.

```
hadoop version
```

The command returns the Hadoop version running in the cluster. Note down the first three digits in the version.

7. Go to https://archive.apache.org/dist/hadoop/common/ and download the Hadoop version that matches the first three digits of the version running in the cluster.

For example, if your Hadoop version is 3.1.1.7.1.9.0-296, then you need to download Hadoop 3.1.1.

8. Extract the downloaded archive.
9. Copy the following libraries from the downloaded archive to /opt/nativelibs on the cluster host.

```
libhadoop.a
                  libhadoop.so.1.0.0
```

The libraries are located in hadoop-*[***VERSION***]*/lib/native.

10. Create a symlink named libhadoop.so and point it to /opt/nativelibs/libhadoop.so.1.0.0.

```
ln -s /opt/nativelibs/libhadoop.so.1.0.0 /opt/nativelibs/lib
hadoop.so
```

11. Change the owner of every entry within /opt/nativelibs to kafka.

```
chown -h kafka:kafka /opt/nativelibs/*
```

12. In Cloudera Manager, go to Kafka service Configuration .
13. Add the following key-value pair to Kafka Connect Environment Advanced Configuration Snippet (Safety Valve).
    - Key: LD_LIBRARY_PATH
    - Value: /opt/nativelibs
14. Click Save Changes.
15. Restart the Kafka service.

OPSAPS-69317: Kafka Connect Rolling Restart Check fails if SSL Client authentication is required

The rolling restart action does not work in Kafka Connect when the ssl.client.auth option is set to required. The health check fails with a timeout which blocks restarting the subsequent Kafka Connect instances.

You can set ssl.client.auth to requested instead of required and initiate a rolling restart again. Alternatively, you can perform the rolling restart manually by restarting the Kafka Connect instances one-by-one and checking periodically whether the service endpoint is available before starting the next one.

Unsupported Features

The following Kafka features are not supported in Cloudera Data Platform:

- Only Java and .Net based clients are supported. Clients developed with C, C++, Python, and other languages are currently not supported.
- The Kafka default authorizer is not supported. This includes setting ACLs and all related APIs, broker functionality, and command-line tools.
- SASL/SCRAM is only supported for delegation token based authentication. It is not supported as a standalone authentication mechanism.
- Kafka KRaft in this release of Cloudera Runtime is in technical preview and does not support the following:
  - Deployments with multiple log directories. This includes deployments that use JBOD for storage.
  - Delegation token based authentication.
  - Migrating an already running Kafka service from ZooKeeper to KRaft.
  - Atlas Integration.

Limitations

Collection of Partition Level Metrics May Cause Cloudera Manager's Performance to Degrade

If the Kafka service operates with a large number of partitions, collection of partition level metrics may cause Cloudera Manager's performance to degrade.

If you are observing performance degradation and your cluster is operating with a high number of partitions, you can choose to disable the collection of partition level metrics.

Important: If you are using SMM to monitor Kafka or Cruise Control for rebalancing Kafka partitions, be aware that both SMM and Cruise Control rely on partition level metrics. If partition level metric collection is disabled, SMM will not be able to display information about partitions. In addition, Cruise Control will not operate properly.

Complete the following steps to turn off the collection of partition level metrics:

1. Obtain the Kafka service name:
   a. In Cloudera Manager, Select the Kafka service.
   b. Select any available chart, and select Open in Chart Builder from the configuration icon drop-down.
   c. Find $SERVICENAME= near the top of the display.

      The Kafka service name is the value of $SERVICENAME.
2. Turn off the collection of partition level metrics:
   a. Go to Hosts Hosts Configuration .
   b. Find and configure the Cloudera Manager Agent Monitoring Advanced Configuration Snippet (Safety Valve) configuration property.

      Enter the following to turn off the collection of partition level metrics:

      ```
      [KAFKA_SERVICE_NAME]_feature_send_broker_topic_partition_ent
      ity_update_enabled=false
      ```

      Replace [KAFKA_SERVICE_NAME] with the service name of Kafka obtained in step 1. The service name should always be in lower case.
   c. Click Save Changes.

Known Issues in Kerberos

There are no known issues for Kerberos in Cloudera Runtime 7.3.1.0.

Known Issues in Apache Knox

Learn about the known issues in Knox, the impact or changes to the functionality, and the workaround.

CDPD-71305: Concurrent impala shell connection failure

If a user makes a concurrent impala-shell connection through Knox, then the connection fails.

Use only one Knox role.

CDPD-68146: Unable to update the log level for Knox from Cloudera Manager

Users are not able to change the log level for Knox from Cloudera Manager. Hence, it impacts debugging in case of any issue.

Change the level for the org.apache.knox.gateway logger in /var/lib/knox/gateway/conf/gateway-log4j2.xml file and restart Knox.

CDPD-64652: During CDH + OS rolling upgrade knox admin api access fails with 403 ACL authorization failures

During OS upgrades, attempts to access Knox on the host being upgraded may produce occasional 403 HTTP responses.

Since the cause is the unavailability of underlying OS service(s), wait and retry the failed request(s).

CDPD-60379: During rolling upgrade of Knox service, access fails with 503/500/404/403 error code

The user operation which is performed during the rolling upgrade of knox might fail with 503/500/404/403 error code.

Retry the user operation.

CDPD-60376: Cloud loadbalancer takes 20-30 secs to failover to the next available knox host

If Knox is in HA and one of the Knox server is down, then accessing of service via Control plane endpoint url(i.e. via cloud loadbalancer) will take ~ 30secs to failover the request to available knox instance.

Retry the request after 30 seconds.

CDPD-3125: Logging out of Atlas does not manage the external authentication

At this time, Atlas does not communicate a log-out event with the external authentication management, Apache Knox. When you log out of Atlas, you can still open the instance of Atlas from the same web browser without re-authentication.

To prevent additional access to Atlas, close all browser windows and exit the browser.

Known Issues in Apache Kudu

Learn about the known issues in Kudu, the impact or changes to the functionality, and the workaround.

- Kudu HMS Sync is disabled and is not yet supported

Kudu supports both coarse-grain and fine-grain authorization, but Kudu does not yet support integration with Atlas.

None

KUDU-3619: Major delta compaction for a tablet might fail for particular workloads due to a bug introduced with KUDU-3367

A bug has been introduced with KUDU-3367 functionality. The bug manifests itself when a tablet server's maintenance thread attempts to run a major delta compaction on a tablet where many rows have been deleted, and the attempt fails with an error. To know more about the error message pattern, see KUDU-3619. If that happens, the corresponding tablet might accumulate a lot of updates that cannot be compacted and later garbage collected. In extreme cases, it could lead to running out of disk space when many tablet replicas hosted at the same tablet server hit the issue.

If a tablet server is affected by the issue, messages like the below are present in the tablet server's logs, where <tabletUUID> and <rowsetID> placeholders are populated with corresponding identifiers:

Major delta compaction failed on <tabletUUID>: Corruption: Failed major delta compaction on RowSet(<rowsetID>): No min key found: CFile base data in RowSet(<rowsetID>).

Set the --all_delete_op_delta_file_cnt_for_compaction flag to a very high value (e.g. 1000000) using the Tablet Server Advanced Configuration Snippet (Safety Valve) for gflagfile in the Cloudera Manager UI and restart all the tablet servers in the Kudu cluster.

Apache Jira: KUDU-3619

Known Issues in Apache Oozie

Learn about the known issues in Oozie, the impact or changes to the functionality, and the workaround.

Oozie jobs fail (gracefully) on secure YARN clusters when JobHistory server is down

If the JobHistory server is down on a YARN (MRv2) cluster, Oozie attempts to submit a job, by default, three times. If the job fails, Oozie automatically puts the workflow in a SUSPEND state.

Workaround: When the JobHistory server is running again, use the resume command to inform Oozie to continue the workflow from the point at which it left off.

CDPD-5340: The resourceManager property defined in an Oozie workflow might not work properly if the workflow is submitted through Knox proxy.

An Oozie workflow defined to use the resourceManager property might not work as expected in situations when the workflow is submitted through Knox proxy.

Workaround: Define the jobTracker property with the same value as that of the resourceManager property.

Unsupported Feature

The following Oozie features are currently not supported in Cloudera:

- Non-support for Pig action (CDPD-1070)
- Conditional coordinator input logic

Known Issues in Apache Parquet

There are no known issues for Parquet in Cloudera Runtime 7.3.1.0.

Known Issues in Apache Phoenix

There are no known issues for Phoenix in Cloudera Runtime 7.3.1.

Known Issues in Apache Ranger

Learn about the known issues in Ranger, the impact or changes to the functionality, and the workaround.

CDPD-3296: Audit files for Ranger plugin components do not appear immediately in S3 after cluster creation

For Ranger plugin components (Atlas, Hive, HBase, etc.), audit data is updated when the applicable audit file is rolled over. The default Ranger audit rollover time is 24 hours, so audit data appears 24 hours after cluster creation.

To see the audit logs in S3 before the default rollover time of 24 hours, use the following steps to override the default value in the Cloudera Manager safety valve for the applicable service.

1. On the Configuration tab in the applicable service, select Advanced under CATEGORY.
2. Click the + icon for the <service_name> Advanced Configuration Snippet (Safety Valve) for ranger-<service_name>-audit.xml property.
3. Enter the following property in the Name box:

   xasecure.audit.destination.hdfs.file.rollover.sec.
4. Enter the desired rollover interval (in seconds) in the Value box. For example, if you specify 180, the audit log data is updated every 3 minutes.
5. Click Save Changes and restart the service.

Known Issues in Schema Registry

Learn about the known issues in Schema Registry, the impact or changes to the functionality, and the workaround.

CDPD-40380: Authorization checking issue when Kerberos is disabled

Due to an issue in Ranger, when Kerberos is disabled then it is not possible to check authorization.

1. Open Schema Registry configuration in Cloudera Manager.
2. Find the ranger.plugin.schema-registry.service.name field.
3. Replace GENERATED_RANGER_SERVICE_NAME with the actual name of the service.
4. Restart the Schema Registry service.

CDPD-49304: AvroConverter does not support composite default values

AvroConverter cannot handle schemas containing a STRUCT type default value.

None.

OPSAPS-70971: Schema Registry does not have permissions to use Atlas after an upgrade

Following an upgrade, Schema Registry might not have the required permissions in Ranger to access Atlas. As a result, Schema Registry's integration with Atlas might not function in secure clusters where Ranger authorization is enabled.

1. Access the Ranger Console (Ranger Admin web UI).
2. Click the cm_atlas resource-based service.
3. Add the schemaregistry user to the all - * policies.
4. Click Manage Service Edit Service .
5. Add the schemaregistry user to the default.policy.users property.

OPSAPS-69317: Kafka Connect Rolling Restart Check fails if SSL Client authentication is required

The rolling restart action does not work in Kafka Connect when the ssl.client.auth option is set to required. The health check fails with a timeout which blocks restarting the subsequent Kafka Connect instances.

You can set ssl.client.auth to requested instead of required and initiate a rolling restart again. Alternatively, you can perform the rolling restart manually by restarting the Kafka Connect instances one-by-one and checking periodically whether the service endpoint is available before starting the next one.

Known Issues in Apache Solr

Learn about the known issues in Apache Solr, the impact or changes to the functionality, and the workaround.

HBase indexer does not load netty and snappy libraries

The HBase indexer loads the netty and snappy libraries and these libraries are necessary for the Key-Value Indexer to work. However, the Key-Value Indexer cannot use these libraries if /tmp is mounted with the noexec property. To address this issue, you have to manually specify another directory instead of the default /tmp.

Perform the following steps to resolve this issue:

1. Go to the Key-Value Store Indexer service Configuration .
2. Search for the Key-Value Store Indexer Service Environment Advanced Configuration Snippet (Safety Valve) property.
3. If the HBASE_INDEXER_OPTS key is already present in the configuration, append the following value else add the following key and value:

```
Name: HBASE_INDEXER_OPTS
Value: -Dorg.apache.hbase.thirdparty.io.netty.native.workdir=/
var/hbase-solr/netty-workdir -Dorg.xerial.snappy.tempdir=/var/
hbase-solr/snappy-tempdir
```

Note:

- If the /var/hbase-solr/netty-workdir and /var/hbase-solr/snappy-tempdir file system locations do not exist, create the directories and ensure that the "hbase" user has permissions to write into these directories.
- Run the chown command on the directories. For example,

```
chown -R hbase:hbase /var/hbase-solr/netty-workdir
```

4. Restart the Key-Value Store Indexer service by clicking Key-Value Store Indexer service Actions Restart .

HBase Indexer does not work with JDK 17

Depending on the Cloudera Manager version used with Cloudera, HBase Indexer (KS Indexer) may have compatibility issues with JDK 17.

You have the following options to fix this issue:

- Upgrade Cloudera Manager to version 7.11.3 or higher.
- If upgrading Cloudera Manager is not an option, you can manually add the following to HBase Indexer Java options in Cloudera Manager:

```
--add-opens java.base/java.nio=ALL-UNNAMED --add-opens java.
base/java.util.concurrent.atomic=ALL-UNNAMED --add-opens jav
a.base/java.lang=ALL-UNNAMED --add-opens java.base/java.lang
.reflect=ALL-UNNAMED
```

Splitshard operation fails after CDH 6 to Cloudera upgrade

Collections are not reindexed during an upgrade from CDH 6 to Cloudera 7 because Lucene 8 (Cloudera) can read Lucene 7 (CDH 6) indexes.

If you try to execute a SPLITSHARD operation against such a collection, it fails with a similar error message:

```
o.a.s.h.a.SplitOp ERROR executing split: => java.lang.IllegalArg
umentException: Cannot merge a segment t
```

```
hat has been created with major version 7 into this index which
 has been created by major version 8
        at org.apache.lucene.index.IndexWriter.validateMergeRea
der(IndexWriter.java:3044)
java.lang.IllegalArgumentException: Cannot merge a segment that h
as been created with major version 7 into this index which has b
een created by major version 8
        at org.apache.lucene.index.IndexWriter.validateMergeReade
r(IndexWriter.java:3044) ~[lucene-core-8.11.2.7.1.9.3-2.jar:8.11
.2.7.1.9.3-2 a6ff93f9665115dffbdad0ad7f222fd1978d495d - jenkins -
 2023-12-02 00:05:23]
        at org.apache.lucene.index.IndexWriter.addIndexes(IndexWr
iter.java:3110) ~[lucene-core-8.11.2.7.1.9.3-2.jar:8.11.2.7.1.9.
3-2 a6ff93f9665115dffbdad0ad7f222fd1978d495d - jenkins - 2023-12
-02 00:05:23]
        at org.apache.solr.update.SolrIndexSplitter.doSplit(So
lrIndexSplitter.java:318) ~[solr-core-8.11.2.7.1.9.3-2.jar:8.11.
2.7.1.9.3-2 a6ff93f9665115dffbdad0ad7f222fd1978d495d - jenkins -
 2023-12-02 00:16:28]
        at org.apache.solr.update.SolrIndexSplitter.split(Solr
IndexSplitter.java:184) ~[solr-core-8.11.2.7.1.9.3-2.jar:8.11.2.
7.1.9.3-2 a6ff93f9665115dffbdad0ad7f222fd1978d495d - jenkins - 2
023-12-02 00:16:28]
        at org.apache.solr.update.DirectUpdateHandler2.split(Dir
ectUpdateHandler2.java:922) ~[solr-core-8.11.2.7.1.9.3-2.jar:8.1
1.2.7.1.9.3-2 a6ff93f9665115dffbdad0ad7f222fd1978d495d - jenkins
 - 2023-12-02 00:16:28]
        at org.apache.solr.handler.admin.SplitOp.execute(SplitOp
.java:165) ~[solr-core-8.11.2.7.1.9.3-2.jar:8.11.2.7.1.9.3-2 a6f
f93f9665115dffbdad0ad7f222fd1978d495d - jenkins - 2023-12-02 00:
16:28]
        at org.apache.solr.handler.admin.CoreAdminOperation.execu
te(CoreAdminOperation.java:367) ~[solr-core-8.11.2.7.1.9.3-2.jar
:8.11.2.7.1.9.3-2 a6ff93f9665115dffbdad0ad7f222fd1978d495d - jen
kins - 2023-12-02 00:16:28]
```

This happens because the segment created using a Lucene 7 index cannot be merged into a Lucene 8 index.

Drop the entire collection, delete the data in HDFS and recreate the collection with Solr 8 configs.

Changing the default value of Client Connection Registry HBase configuration parameter causes HBase MRIT job to fail

If the value of the HBase configuration property Client Connection Registry is changed from the default ZooKeeper Quorum to Master Registry then the Yarn job started by HBase MRIT fails with a similar error message:

```
Caused by: org.apache.hadoop.hbase.exceptions.MasterRegistryFetc
hException: Exception making rpc to masters [quasar-bmyccr-2.qua
sar-bmyccr.root.hwx.site,22001,-1]
        at org.apache.hadoop.hbase.client.MasterRegistry.lambda$g
roupCall$1(MasterRegistry.java:244)
        at org.apache.hadoop.hbase.util.FutureUtils.lambda$addLi
stener$0(FutureUtils.java:68)
        at java.util.concurrent.CompletableFuture.uniWhenCompl
ete(CompletableFuture.java:774)
        at java.util.concurrent.CompletableFuture.uniWhenComplet
eStage(CompletableFuture.java:792)
        at java.util.concurrent.CompletableFuture.whenComplete(Co
mpletableFuture.java:2153)
        at org.apache.hadoop.hbase.util.FutureUtils.addListener(F
utureUtils.java:61)
```

```
        at org.apache.hadoop.hbase.client.MasterRegistry.groupCa
ll(MasterRegistry.java:228)
        at org.apache.hadoop.hbase.client.MasterRegistry.call(Ma
sterRegistry.java:265)
        at org.apache.hadoop.hbase.client.MasterRegistry.getMetaR
egionLocations(MasterRegistry.java:282)
        at org.apache.hadoop.hbase.client.ConnectionImplementati
on.locateMeta(ConnectionImplementation.java:900)
        at org.apache.hadoop.hbase.client.ConnectionImplementat
ion.locateRegion(ConnectionImplementation.java:867)
        at org.apache.hadoop.hbase.client.ConnectionImplementati
on.relocateRegion(ConnectionImplementation.java:850)
        at org.apache.hadoop.hbase.client.ConnectionImplementat
ion.locateRegionInMeta(ConnectionImplementation.java:981)
        at org.apache.hadoop.hbase.client.ConnectionImplementa
tion.locateRegion(ConnectionImplementation.java:870)
        at org.apache.hadoop.hbase.client.RpcRetryingCallerWith
ReadReplicas.getRegionLocations(RpcRetryingCallerWithReadReplica
s.java:319)
        ... 21 more
Caused by: org.apache.hadoop.hbase.client.RetriesExhaustedExcept
ion: Failed contacting masters after 1 attempts.
Exceptions:
java.io.IOException: Call to address=quasar-bmyccr-2.quasar-bmy
ccr.root.hwx.site/172.27.19.4:22001 failed on local exception: j
ava.io.IOException: java.lang.RuntimeException: Found no valid a
uthentication method from options
        at org.apache.hadoop.hbase.client.MasterRegistry.lambda
$groupCall$1(MasterRegistry.java:243)
        ... 35 more
```

Add the following line to the MRIT command line:

```
-D 'hbase.client.registry.impl=org.apache.hadoop.hbase.client.ZK
ConnectionRegistry'
```

Solr does not support rolling upgrade to release 7.2.18 or lower

Solr supports rolling upgrades from release 7.2.18 and higher. Upgrading from a lower version means that all the Solr Server instances are shut down, parcels upgraded and activated and then the Solr Servers are started again. This causes a service interruption of several minutes, the actual value depending on cluster size.

Services like Atlas and Ranger that depend on Solr, may face issues because of this service interruption.

None.

Unable to see single valued and multivalued empty string values when querying collections after upgrade to Cloudera

After upgrading from CDH or HDP to Cloudera, you are not able to see single valued and multi Valued empty string values in Cloudera.

This behavior in Cloudera is due to the remove-blank processor present in solrconfig.xml in Solr 8.

Remove the remove-blank processor from solrconfig.xml.

Cannot create multiple heap dump files because of file name error

Heap dump generation fails with a similar error message:

```
java.lang.OutOfMemoryError: Java heap space
Dumping heap to /data/tmp/solr_solr-SOLR_SERVER-fc9dacc265fabfc5
00b92112712505e3_pid{{PID}}.hprof ...
```

```
Unable to create /data/tmp/solr_solr-SOLR_SERVER-fc9dacc265fab
fc500b92112712505e3_pid{{PID}}.hprof: File exists
```

The cause of the problem is that {{PID}} does not get substituted during dump file creation with an actual process ID and because of that, a generic file name is generated. This causes the next dump file creation to fail, as the existing file with the same name cannot be overwritten.

You need to manually delete the existing dump file.

Solr coreAdmin status throws Null Pointer Exception

You get a Null Pointer Exception with a similar stacktrace:

```
Caused by: java.lang.NullPointerException
    at org.apache.solr.core.SolrCore.getInstancePath(SolrCore.
java:333)
    at org.apache.solr.handler.admin.CoreAdminOperation.getCor
eStatus(CoreAdminOperation.java:324)
    at org.apache.solr.handler.admin.StatusOp.execute(StatusOp.
java:46)
    at org.apache.solr.handler.admin.CoreAdminOperation.execute
(CoreAdminOperation.java:362)
```

This is caused by an error in handling solr admin core STATUS after collections are rebuilt.

Restart the Solr server.

Applications fail because of mixed authentication methods within dependency chain of services

Using different types of authentication methods within a dependency chain, for example, configuring your indexer tool to authenticate using Kerberos and configuring your Solr Server to use LDAP for authentication may cause your application to time out and eventually fail.

Make sure that all services in a dependency chain use the same type of authentication.

API calls fail with error when used with alias, but work with collection name

API calls fail with a similar error message when used with an alias, but they work when made using the collection name:

```
[   ] o.a.h.s.t.d.w.DelegationTokenAuthenticationFilter Authenti
cation exception: User: xyz@something.example.com is not allowed
 to impersonate xyz@something.example.com
  [c:RTOTagMetaOdd s:shard3 r:core_node11 x:RTOTagMetaOdd_shar
d3_replica_n8] o.a.h.s.t.d.w.DelegationTokenAuthenticationFilter
 Authentication exception: User: xyz@something.example.com is not
 allowed to impersonate xyz@something.example.com
```

Make sure there is a replica of the collection on every host.

CrunchIndexerTool does not work out of the box if /tmp is mounted noexec mode

When you try to run CrunchIndexerTool with the /tmp directory mounted in noexec mode, It throws a snappy-related error.

Create a separate directory for snappy temp files which is mounted with EXEC privileges and set this directory as the value of the org.xerial.snappy.tempdir java property as a driver java option.

For example:

```
export myDriverJarDir=/opt/cloudera/parcels/CDH//lib/solr/contri
b/crunch;export myDependencyJarDir=/opt/cloudera/parcels/CDH//
lib/search/lib/search-crunch;export myDriverJar=$(find $myDriv
erJarDir -maxdepth 1 -name 'search-crunch-*.jar' ! -name '*-job.
jar' ! -name '*-sources.jar');export myDependencyJarFiles=$(find
 $myDependencyJarDir -name '*.jar' | sort | tr '\n' ',' | head
 -c -1);export myDependencyJarPaths=$(find $myDependencyJarDir
```

```
 -name '*.jar' | sort | tr '\n' ':' | head -c -1);export HADOOP_
CONF_DIR=;spark-submit --master local --deploy-mode client --
driver-library-path /opt/cloudera/parcels/CDH//lib/hadoop/lib/
native/ --jars $myDependencyJarFiles --driver-java-options ' -
Dorg.xerial.snappy.tempdir=/home/systest/tmp ' --class org.apa
che.solr.crunch.CrunchIndexerTool $myDriverJar --input-file-form
at=avroParquet --input-file-reader-schema search-parquetfile/par
quet-schema.avsc --morphline-file /tmp/mrTestBase.conf --pipelin
e-type spark --chatty hdfs://[***HOSTNAME***]:8020/tmp/parquetfi
leparsertest-input
```

Mergeindex operation with --go-live fails after CDH 6 to Cloudera upgrade

During an upgrade from CDH6 to Cloudera, collections are not reindexed because Lucene 8 (Cloudera) can read Lucene 7 (CDH6) indexes.

If you try to execute MapReduceIndexerTool (MRIT) or HBase Indexer MRIT with --go-live against such a collection, you get a similar error message:

```
Caused by: java.lang.IllegalArgumentException: Cannot merge a se
gment that has been created with major version 8 into this index
 which has been created by major version 7
        at org.apache.lucene.index.IndexWriter.validateMergeReade
r(IndexWriter.java:2894)
        at org.apache.lucene.index.IndexWriter.addIndexes(Index
Writer.java:2960)
        at org.apache.solr.update.DirectUpdateHandler2.mergeIn
dexes(DirectUpdateHandler2.java:570)
        at org.apache.solr.update.processor.RunUpdateProcessor.
processMergeIndexes(RunUpdateProcessorFactory.java:95)
        at org.apache.solr.update.processor.UpdateRequestProcesso
r.processMergeIndexes(UpdateRequestProcessor.java:63)
```

This happens because Cloudera MRIT and HBase indexer use Solr 8 as embedded Solr, which creates a Lucene 8 index. It cannot be merged (using MERGEINDEXES) into an older Lucene 7 index.

In the case of MRIT the only way to move past this issue is to drop the entire collection, delete the data in HDFS and recreate the collection with Solr 8 configs.

For HBase Indexer MRIT an alternative workaround is setting the number of reducers to 0 (--re ducers 0) because in this case documents are sent directly from the mapper tasks to live Solr servers instead of using MERGEINDEXES.

Apache Tika upgrade may break morphlines indexing

The upgrade of Apache Tika from 1.27 to 2.3.0 brought potentially breaking changes for morphlines indexing. Duplicate/triplicate keys names were removed and certain parser class names were changed (For example, org.apache.tika.parser.jpeg.JpegParser changed to org.apache.tika.parser.i mage.JpegParser).

To avoid morphline commands failing after the upgrade, do the following:

- Check if key name changes affect your morphlines. For more information, see *Removed duplicate/triplicate keys* in Migrating to Tika 2.0.0.
- Check if the name of any parser you use has changed. For more information, see the Apache Tika API documentation.

Update your morphlines if necessary.

CDPD-28006: Solr access via Knox fails with impersonation error though auth_to_local and proxy user configs are set

Currently the names of system users which are impersonating users with Solr should match with the names of their respective Kerberos principals.

If, for some reason, this is not feasible, you must add the user name you want to associate with the custom Kerberos principal to Solr configuration via the Solr Service Environment Advanced Configuration Snippet (Safety Valve) environment variable in Cloudera Manager.

For more information, see Configuring custom Kerberos principals and custom system users.

CDH-77598: Indexing fails with socketTimeout

Starting from CDH 6.0, the HTTP client library used by Solr has a default socket timeout of 10 minutes. Because of this, if a single request sent from an indexer executor to Solr takes more than 10 minutes to be serviced, the indexing process fails with a timeout error.

This timeout has been raised to 24 hours. Nevertheless, there still may be use cases where even this extended timeout period proves insufficient.

If your MapreduceIndexerTool or HBaseMapreduceIndexerTool batch indexing jobs fail with a timeout error during the go-live (Live merge, MERGEINDEXES) phase (This means the merge takes longer than 24 hours).

Use the --go-live-timeout option where the timeout can be specified in milliseconds.

CDPD-12450: CrunchIndexerTool Indexing fails with socketTimeout

The http client library uses a socket timeout of 10 minutes. The Spark Crunch Indexer does not override this value, and in case a single batch takes more than 10 minutes, the entire indexing job fails. This can happen especially if the morphlines contain DeleteByQuery requests.

Try the following workarounds:

- Check the batch size of your indexing job. Sending too large batches to Solr might increase the time needed on the Solr server to process the incoming batch.
- If your indexing job uses deleteByQuery requests, consider using deleteById wherever possible as deleteByQuery involves a complex locking mechanism on the Solr side which makes processing the requests slower.
- Check the number of executors for your Spark Crunch Indexer job. Too many executors can overload the Solr service. You can configure the number of executors by using the --mappers parameter
- Check that your Solr installation is correctly sized to accommodate the indexing load, making sure that the number of Solr servers and the number of shards in your target collection are adequate.
- The socket timeout for the connection can be configured in the morphline file. Add the solrClientSocketTimeout parameter to the solrLocator command

  Example

```
SOLR_LOCATOR :
{
  collection : test_collection
  zkHost : "zookeeper1.example.corp:2181/solr"
# 10 minutes in milliseconds
  solrClientSocketTimeout: 600000
  # Max number of documents to pass per RPC from morphline to
 Solr Server
  # batchSize : 10000
}
```

CDPD-29289: HBaseMapReduceIndexerTool fails with socketTimeout

The http client library uses a socket timeout of 10 minutes. The HBase Indexer does not override this value, and in case a single batch takes more than 10 minutes, the entire indexing job fails.

You can overwrite the default 600000 millisecond (10 minute) socket timeout in HBase indexer using the --solr-client-socket-timeout optional argument for the direct writing mode (when the value of the --reducers optional argument is set to 0 and mappers directly send the data to the live Solr).

CDPD-20577: Splitshard operation on HDFS index checks local filesystem and fails

When performing a shard split on an index that is stored on HDFS, SplitShardCmd still evaluates free disk space on the local file system of the server where Solr is installed. This may cause the command to fail, perceiving that there is no adequate disk space to perform the shard split.

Run the following command to skip the check for sufficient disk space altogether:

- On nonsecure clusters:

```
curl 'http://$[***SOLR_SERVER_HOSTNAME***]:8983/so
lr/admin/collections?action=SPLITSHARD&collectio
n=[***COLLECTION_NAME***]&shard=[***SHARD_TO_SPLIT***]&skipFre
eSpaceCheck=true'
```

- On secure clusters:

```
curl -k -u : --negotiate 'http://
$[***SOLR_SERVER_HOSTNAME***]:8985/solr/admin/collections
?action=SPLITSHARD&collection=[***COLLECTION_NAME***]&sha
rd=[***SHARD_TO_SPLIT***]&skipFreeSpaceCheck=true'
```

Replace *[***SOLR_SERVER_HOSTNAME***]* with a valid Solr server hostname, *[***COLLECTION_NAME***]* with the collection name, and *[***SHARD_TO_SPLIT***]* with the ID of the to split.

To verify that the command executed succesfully, check overseer logs for a similar entry:

```
2021-02-02 12:43:23.743 INFO  (OverseerThreadFactory-9-thread-5-
processing-n:myhost.example.com:8983_solr) [c:example s:shard1
  ] o.a.s.c.a.c.SplitShardCmd Skipping check for sufficient disk
 space
```

CDH-22190: CrunchIndexerTool which includes Spark indexer requires specific input file format specifications

If the --input-file-format option is specified with CrunchIndexerTool, then its argument must be text, avro, or avroParquet, rather than a fully qualified class name.

None

CDH-26856: Field value class guessing and Automatic schema field addition are not supported with the MapReduceIndexerTool nor with the HBaseMapReduceIndexerTool

The MapReduceIndexerTool and the HBaseMapReduceIndexerTool can be used with a Managed Schema created via NRT indexing of documents or via the Solr Schema API. However, neither tool supports adding fields automatically to the schema during ingest.

Define the schema before running the MapReduceIndexerTool or HBaseMapReduceIndexerTool. In non-schemaless mode, define in the schema using the schema.xml file. In schemaless mode, either define the schema using the Solr Schema API or index sample documents using NRT indexing before invoking the tools. In either case, Cloudera recommends that you verify that the schema is what you expect, using the List Fields API command.

Users with insufficient Solr permissions may encounter a blank Solr Web Admin UI

Users who are not authorized to use the Solr Admin UI are not given a page explaining that access is denied to them, instead they receive a blank Admin UI with no information.

None

CDH-15441: Using MapReduceIndexerTool or HBaseMapReduceIndexerTool multiple times may produce duplicate entries in a collection

Repeatedly running the MapReduceIndexerTool on the same set of input files can result in duplicate entries in the Solr collection. This occurs because the tool can only insert documents

and cannot update or delete existing Solr documents. This issue does not apply to the HBaseMapReduceIndexerTool unless it is run with more than zero reducers.

To avoid this issue, use HBaseMapReduceIndexerTool with zero reducers.

Note: This workaround is only valid for HBaseMapReduceIndexerTool. There is no workaround for MapReduceIndexerTool

CDH-58694: Deleting collections might fail if hosts are unavailable

It is possible to delete a collection when hosts that host some of the collection are unavailable. After such a deletion, if the previously unavailable hosts are brought back online, the deleted collection may be restored.

Ensure all hosts are online before deleting collections.

CDPD-13923: Every Configset is Untrusted Without Kerberos

Solr 8 introduces the concept of 'untrusted configset', denoting configsets that were uploaded without authentication. Collections created with an untrusted configset will not initialize if <lib> directives are used in the configset.

Select one of the following options if you would like to use untrusted configsets with <lib> directives:

- If the configset contains external libraries, but you do not want to use them, simply upload the configsets after deleting the <lib> directives.
- If the configset contains external libraries, and you want to use them, choose one from the following options:
  - Secure your cluster before reuploading the configset.
  - Add the libraries to Solr's classpath, then reupload the configset without the <lib> directives.

CDPD-71422: Solr went into an unhealthy state after the data lake upgrade

After the Data Lake upgrade to the 7.3.1.0 version, the Solr service becomes unhealthy for the public cloud environments (AWS, Azure, and GCP). This is an intermittent issue.

Manually restart the Solr service in the Data Lake after an upgrade.

Unsupported features

The following Solr features are currently not supported in Cloudera:

- Panel with security info in admin UI's dashboard
- Incremental backup mode
- Schema Designer UI
- Package Management System
- HTTP/2
- Solr SQL/JDBC
- Graph Traversal
- Cross Data Center Replication (CDCR)
- SolrCloud Autoscaling
- HDFS Federation
- Saving search results
- Solr contrib modules

  (Spark, MapReduce, and Lily HBase indexers are not contrib modules but part of Cloudera's distribution of Solr itself, therefore they are supported)

Limitations

Enabling blockcache writing may result in unusable indexes

It is possible to create indexes with solr.hdfs.blockcache.write.enabled set to true. Such indexes may appear corrupt to readers, and reading these indexes may irrecoverably corrupt them. Because of this, blockcache writing is disabled by default.

Default Solr core names cannot be changed

Although it is technically possible to give user-defined Solr core names during core creation, it is to be avoided in the context of Cloudera's distribution of Apache Solr. Cloudera Manager expects core names in the default "collection_shardX_replicaY" format. Altering core names results in Cloudera Manager being unable to fetch Solr metrics for the given core and this may corrupt data collection for co-located core, or even shard, and server level charts.

Lucene index handling limitation

The Lucene index can only be upgraded by one major version. Solr 8 will not open an index that was created with Solr 6 or earlier. Because of this, you need to reindex collections that were created with Solr 6 or earlier.

Known Issues in Apache Spark

Learn about the known issues in Spark, the impact or changes to the functionality, and the workaround.

Spark 3: RAPIDS Accelerator is not available

The RAPIDS Accelerator for Apache Spark is currently not available in 7.3.1.0.

Workaround: None

The CHAR(n) type handled inconsistently, depending on whether the table is partitioned or not.

In upstream Spark 3 the spark.sql.legacy.charVarcharAsString configuration was introduced, but it does not solve all incompatibilities with Spark 2.

Workaround: None. A new configuration spark.cloudera.legacy.charVarcharLegacyPadding will be introduced in a future version to keep compatibility with Spark 2, but it isn't available in 7.3.1.

Note: The CHAR type is legacy in SQL, and using it is discouraged. Cloudera recommends using VARCHAR or STRING instead.

Apache Jira: SPARK-33480

Known Issues for Apache Sqoop

Learn about the known issues in Apache Sqoop, the impact or changes to the functionality, and the workaround.

CDPD-44431: Using direct mode causes problems

Using direct mode has several drawbacks:

- Imports can cause an intermittent and overlapping input split.
- Imports can generate duplicate data.
- Many problems, such as intermittent failures, can occur.
- Additional configuration is required.

Stop using direct mode. Do not use the --direct option in Sqoop import or export commands.

Sqoop direct mode is disabled by default. However, if you still want to use it, enable it by either setting the sqoop.enable.deprecated.direct property globally in Cloudera Manager for Sqoop or by specifying it in the command-line through -Dsqoop.enable.deprecated.direct=true.

CDPD-3089: Avro, S3, and HCat do not work together properly

Importing an Avro file into S3 with HCat fails with Delegation Token not available.

Parquet columns inadvertently renamed

Problem: Column names that start with a number are renamed when you use the --as-parquetfile option to import data.

Workaround: Prepend column names in Parquet tables with one or more letters or underscore characters.

Apache JIRA: None

Importing Parquet files might cause out-of-memory (OOM) errors

Problem: Importing multiple megabytes per row before initial-page-run check (ColumnWriter) can cause OOM. Also, rows that vary significantly by size so that the next-page-size check is based on small rows, and is set very high, followed by many large rows can also cause OOM.

PARQUET-99

Known issues in Streams Messaging Manager

Learn about the known issues in Streams Messaging Manager (SMM), the impact or changes to the functionality, and the workaround.

OPSAPS-59597: SMM UI logs are not supported by Cloudera Manager

Cloudera Manager does not display a Log Files menu for SMM UI role (and SMM UI logs cannot be displayed in the Cloudera Manager UI) because the logging type used by SMM UI is not supported by Cloudera Manager.

View the SMM UI logs on the host.

CDPD-39313: Some numbers are not rendered properly in SMM UI

Very large numbers can be imprecisely represented on the UI. For example, bytes larger than 8 petabytes would lose precision.

None.

OPSAPS-59553: SMM's bootstrap server config should be updated based on Kafka's listeners

SMM does not show any metrics for Kafka or Kafka Connect when multiple listeners are set in Kafka.

SMM cannot identify multiple listeners and still points to bootstrap server using the default broker port (9093 for SASL_SSL). You need to override bootstrap server URL (hostname:port as set in the listeners for broker). Add the bootstrap server details in SMM safety valve in the following path:

1. In Cloudera Manager, go to SMMConfigurationStreams Messaging Manager Rest Admin Server Advanced Configuration Snippet (Safety Valve) for streams-messaging-manager.yaml.
2. Add the following value for bootstrap servers.

```
streams.messaging.manager.kafka.bootstrap.servers=<comma-sep
arated list of brokers>
```

3. Save your changes.
4. Restart SMM.

CDPD-45183: Kafka Connect active topics might be visible to unauthorised users

The Kafka Connect active topics endpoint (/connectors/*[***CONNECTOR NAME***]*/topics) and the Connect Cluster page on the SMM UI disregard the user permissions configured for the Kafka service in Ranger. As a result, all active topics of connectors might become visible to users who do not have permissions to view them. Note that user permission configured for Kafka Connect in Ranger are not affected by this issue and are correctly applied.

None.

Limitations

CDPD-36422: 1MB flow.snapshot freezes Safari

While importing large connector configurations, flow.snapshots reduces the usability of the Streams Messaging Manager when using Safari browser.

Use a different browser (Chrome/Firefox/Edge).

Known Issues in Streams Replication Manager

Learn about the known issues in Streams Replication Manager (SRM), the impact or changes to the functionality, and the workaround.

Known Issues

CDPD-22089: SRM does not sync re-created source topics until the offsets have caught up with target topic

Messages written to topics that were deleted and re-created are not replicated until the source topic reaches the same offset as the target topic. For example, if at the time of deletion and re-creation there are a 100 messages on the source and target clusters, new messages will only get replicated once the re-created source topic has 100 messages. This leads to messages being lost.

None

CDPD-11079: Blacklisted topics appear in the list of replicated topics

If a topic was originally replicated but was later disallowed (blacklisted), it will still appear as a replicated topic under the /remote-topics REST API endpoint. As a result, if a call is made to this endpoint, the disallowed topic will be included in the response. Additionally, the disallowed topic will also be visible in the SMM UI. However, it's Partitions and Consumer Groups will be 0, its Throughput, Replication Latency and Checkpoint Latency will show N/A.

None

CDPD-30275: SRM may automatically re-create deleted topics on target clusters

If auto.create.topics.enable is enabled, deleted topics might get automatically re-created on target clusters. This is a timing issue. It only occurs if remote topics are deleted while the replication of the topic is still ongoing.

1. Remove the topic from the topic allowlist with srm-control. For example:

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGE
T_CLUSTER] --remove [TOPIC1]
```

2. Wait until SRM is no longer replicating the topic.
3. Delete the remote topic in the target cluster.

Limitations

SRM cannot replicate Ranger authorization policies to or from Kafka clusters

Due to a limitation in the Kafka-Ranger plugin, SRM cannot replicate Ranger policies to or from clusters that are configured to use Ranger for authorization. If you are using SRM to replicate data to or from a cluster that uses Ranger, disable authorization policy synchronization in SRM. This can be achieved by clearing the Sync Topic Acls Enabled (sync.topic.acls.enabled) checkbox.

SRM cannot ensure the exactly-once semantics of transactional source topics

SRM data replication uses at-least-once guarantees, and as a result cannot ensure the exactly-once semantics (EOS) of transactional topics in the backup/target cluster.

Note: Even though EOS is not guaranteed, you can still replicate the data of a transactional source, but you must set isolation.level to read_committed for SRM's internal consumers. This can be done by adding *[***SOURCE CLUSTER ALIAS***]->[***TARGET CLUSTER ALIAS***]*.consumer.isolation.level=read_committed to the Streams Replication Manager's Replication Configs SRM service property in Cloudera Manger. The isolation.level property can be set on a global connector or replication level. For example:

```
#Global connector level
                  connectors.consumer.isolation.leve
l=read_committed
                  #Replication level
                  uswest->useast.consumer.isolation.le
vel=read_committed
```

SRM checkpointing is not supported for transactional source topics

SRM does not correctly translate checkpoints (committed consumer group offsets) for transactional topics. Checkpointing assumes that the offset mapping function is always increasing, but with transactional source topics this is violated. Transactional topics have control messages in them, which take up an offset in the log, but they are never returned on the consumer API. This causes the mappings to decrease, causing issues in the checkpointing feature. As a result of this limitation, failover operations for transactional topics is not possible.

Known Issues in YARN, YARN Queue Manager and MapReduce

Learn about the known issues in YARN, YARN Queue Manager and MapReduce, the impact or changes to the functionality, and the workaround.

Known Issues

A fresh install of 7.2.18 of YARN Queue Manager does not allow user to bypass the Setup Database screen for YARN Queue Manager

YARN Queue Manager in Cloudera Private Cloud Base 7.2.18 does not require you to install a PostGres database, therefore users should not see the Setup Database screen and should be able to skip the Setup Database screen. With this known issue, users who are conducting a fresh install of 7.2.18 are not able to bypass the Setup Database screen as expected.

1. When conducting a fresh install of YARN Queue Manager in 7.2.18, you must ensure that you have both Cloudera and Cloudera Manager upgraded to 7.2.18.
2. When you reach the Setup Database screen in the Cloudera Manager installation wizard for Queue Manager, enter any dummy values for the following fields:
   a. Database name: configstore
   b. Database Username: dbuser
   c. Database Password: dbpassword

   YARN Queue Manager will not connect to PostGres with the above details and will fall back to the embedded database.
3. Run the following script command in a browser console to enable the Continue button:

   document.querySelector('.btn.next').removeAttribute('disabled');
4. Click Continue and proceed with the YARN Queue Manager installation.

5. After installation is complete, SSH into the host that has Queue Manager installed, and run this command: sed -i 's/migrationCompleted=true/migrationCompleted=false/' /var/lib/hadoop-yarn/migration.properties

   Note: Enable Queue Manager in the YARN configurations, and restart YARN.

6. Restart YARN Queue Manager.

CDPD-46685 Nodemanager logs are filled with logs similar to: 2022-11-28 03:42:39,587 WARN org.apache.hadoop.ipc.Client: Address change detected. Old: deh-34631355-niv-master1.e2e-797.dze1-y40r.int.cldr.work/10.114.128.84:8031 New: deh-34631355-niv-master1.e2e-797.dze1-y40r.int.cldr.work/10.114.128.63:8031 2022-11-28 03:43:01,425 WARN org.apache.hadoop.ipc.Client: Address change detected. Old: deh-34631355-niv-master0.e2e-797.dze1-y40r.int.cldr.work/10.114.128.79:8031 New: deh-34631355-niv-master0.e2e-797.dze1-y40r.int.cldr.work/10.114.128.65:8031.

Restart all YARN NodeManagers, they should come up without issues and Cloudera Manager must recognize them as healthy nodes once the status of them is refreshed upon restart.

YARN cannot start if Kerberos principal name is changed

If the Kerberos principal name is changed in Cloudera Manager after launch, YARN will not be able to start. In such case the keytabs can be correctly generated but YARN cannot access ZooKeeper with the new Kerberos principal name and old ACLs.

There are two possible workarounds:

- Delete the znode and restart the YARN service.
- Use the reset ZK ACLs command. This also sets the znodes below /rmstore/ZKRMStateRoot to world:anyone:cdrwa which is less secure.

Third party applications do not launch if MapReduce framework path is not included in the client configuration

MapReduce application framework is loaded from HDFS instead of being present on the NodeManagers. By default the mapreduce.application.framework.path property is set to the appropriate value, but third party applications with their own configurations will not launch.

Set the mapreduce.application.framework.path property to the appropriate configuration for third party applications.

JobHistory URL mismatch after server relocation

After moving the JobHistory Server to a new host, the URLs listed for the JobHistory Server on the ResourceManager web UI still point to the old JobHistory Server. This affects existing jobs only. New jobs started after the move are not affected.

For any existing jobs that have the incorrect JobHistory Server URL, there is no option other than to allow the jobs to roll off the history over time. For new jobs, make sure that all clients have the updated mapred-site.xml that references the correct JobHistory Server.

CDH-6808: Routable IP address required by ResourceManager

ResourceManager requires routable host:port addresses for yarn.resourcemanager.scheduler.address, and does not support using the wildcard 0.0.0.0 address.

Set the address, in the form host:port, either in the client-side configuration, or on the command line when you submit the job.

CDH-49165: History link in ResourceManager web UI broken for killed Spark applications

When a Spark application is killed, the history link in the ResourceManager web UI does not work.

To view the history for a killed Spark application, see the Spark HistoryServer web UI instead.

COMPX-3329: Autorestart is not enabled for Queue Manager in Data Hub

In a Data Hub cluster, Queue Manager is installed with autorestart disabled. Hence, if Queue Manager goes down, it will not restart automatically.

If Queue Manager goes down in a Data Hub cluster, you must go to the Cloudera Manager Dashboard and restart the Queue Manager service.

COMPX-5817: Queue Manager UI will not be able to present a view of pre-upgrade queue structure. Cloudera Manager Store is not supported and therefore Yarn will not have any of the pre-upgrade queue structure preserved.

When a Data Hub cluster is deleted, all saved configurations are also deleted. All YARN configurations are saved in Cloudera Manager Store and this is yet to be supported in Data Hub and Cloudera Manager. Hence, the YARN queue structure also will be lost when a Data Hub cluster is deleted or upgraded or restored.

CDPD-67150: During the restore procedure it could happen that a node becomes unhealthy as the default YARN Capacity Scheduler configuration is loaded onto the node during the restart.

Perform an extra restart on the Resource Manager role that has the incorrect configuration to restore the correct configuration.

CDPD-75652: Reverse DNS lookup fails for YARN but works for HDFS

Submitting a YARN application from a host without proper DNS setup (reverse DNS does not work for the YARN ResourceManager's host) results in the Server has invalid Kerberos principal error.

Add the following to YARN Service Advanced Configuration Snippet (Safety Valve) for the yarn-site.xml file:

```
<property>
<name>yarn.resourcemanager.principal.pattern</name>
<value>*</value>
</property>
```

Unsupported Features

The following YARN features are currently not supported in Cloudera:

- Application Timeline Server (ATSv2 and ATSv1)
- Container Resizing
- Distributed or Centralized Allocation of Opportunistic Containers
- Distributed Scheduling
- Docker on YARN (DockerContainerExecutor) on Data Hub clusters
- Fair Scheduler
- GPU support for Docker
- Hadoop Pipes
- Native Services
- Pluggable Scheduler Configuration
- Queue Priority Support
- Reservation REST APIs
- Resource Estimator Service
- Resource Profiles
- (non-Zookeeper) ResourceManager State Store
- Rolling Log Aggregation
- Shared Cache
- YARN Federation
- Moving jobs between queues

Known Issues in Apache ZooKeeper

Learn about the known issues in Zookeeper, the impact or changes to the functionality, and the workaround.

Zookeeper-client does not use ZooKeeper TLS/SSL automatically

The command-line tool 'zookeeper-client' is installed to all Cloudera Nodes and it can be used to start the default Java command line ZooKeeper client. However even when ZooKeeper TLS/SSL is enabled, the zookeeper-client command connects to localhost:2181, without using TLS/SSL.

Workaround:

Manually configure the 2182 port, when zookeeper-client connects to a ZooKeeper cluster.The following is an example of connecting to a specific three-node ZooKeeper cluster using TLS/SSL:

```
CLIENT_JVMFLAGS="-Dzookeeper.clientCnxnSocket=org.apache.zoo
keeper.ClientCnxnSocketNetty -Dzookeeper.ssl.keyStore.locati
on=<PATH TO YOUR CONFIGURED KEYSTORE> -Dzookeeper.ssl.keyStor
e.password=<THE PASSWORD YOU CONFIGURED FOR THE KEYSTORE>  -
Dzookeeper.ssl.trustStore.location=<PATH TO YOUR CONFIGURED
 TRUSTSTORE> -Dzookeeper.ssl.trustStore.password=<THE PASSWORD
 YOU CONFIGURED FOR THE TRUSTSTORE> -Dzookeeper.client.secu
re=true" zookeeper-client -server <YOUR.ZOOKEEPER.SERVER-1>:218
2,<YOUR.ZOOKEEPER.SERVER-2>:2182,<YOUR.ZOOKEEPER.SERVER-3>:2182
```

Behavioral Changes In Cloudera Runtime 7.3.1

Behavioral changes denote a marked change in behavior from the previously released version to this version of Cloudera Runtime.

Behavioral Changes in Atlas

Behavioral changes denote a marked change in behavior from the previously released version to this version of Apache Atlas.

Summary:

The Exclude SubTypes and Exclude Sub-classifications filters were removed from the **Table** tab of entity details.

Previous behavior:

Previously, the Exclude SubTypes and Exclude Sub-classifications filters were available from the **Table** tab in entity details. There were no properties being passed to these filters when you visited the entity details of the page.

New behavior:

The two unused filter checkboxes Exclude SubTypes and Exclude Sub-classifications from the **Table** tab of entity detail page were removed.

Summary:

Special character validation was added to glossary, term and category names in Apache Atlas.

Previous behavior:

The special characters ('@', '.', '<', '>') could be used in glossary, term and category name fields.

New behavior:

The special characters ('@', '.', '<', '>') are no longer accepted in glossary, term and category name fields by the validation introduced. Avoid using these characters when creating glossary names, glossary terms and category names.

Behavioral Changes in Hive

Behavioral changes denote a marked change in behavior from the previously released version to this version of Apache Hive.

Summary:

Change in the way compaction initiator and cleaner threads are handled

Previous behavior:

The compaction initiator and cleaner threads are enabled and disabled by setting the hive.compactor.initiator.on property to 'true' or 'false'.

New behavior:

A new property hive.compactor.cleaner.on is introduced that allows you to selectively enable or disable the cleaner thread.

This property is not listed and is set to 'false' by default. Add the property to Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml in Cloudera Manager to have the same out-of-the-box experience as in the previous version.

Also, ensure that you set the property to 'true' for the compactor to run on the HMS instance.

Behavioral Changes in Impala

Behavioral changes denote a marked change in behavior from the previously released version to this version of Apache Impala.

Summary:

Impala now unregisters timed-out queries promptly to free memory, retaining error messages for clients that return later.

Previous behavior:

Timed-out queries remained registered until the session closed, keeping memory occupied and sometimes leaving failed queries in an active state if not explicitly closed.

New behavior:

Timed-out queries are unregistered immediately to free memory, while error messages are kept in a new structure so clients can still receive an error message if they return later.

Apache Jira: IMPALA-12602

Behavioral Changes in Knox

Behavioral changes denote a marked change in behavior from the previously released version to this version of Apache Knox.

Knox token impersonation config

Summary

Knox token service has been changed to use the identity assertion provider configuration for impersonation.

Previous behaviour

The token service had its own impersonation configuration.

New behaviour

The token service relies on the identity assertion provider for impersonation configuration.

PEM file name change

Summary

The name of the pem file generated through knoxcli.sh has been changed.

Previous behaviour

The name of the file was gateway-identity.pem.

New behaviour

The name of the file is now gateway-client-trust.pem.

Composite authorization provider misconfiguration

Summary

Composite authorization provider misconfiguration behavior

Previous behaviour

1. If composite.provider.names is empty, the topology would fail deployment.
2. If composite.provider.names has an invalid value, the topology would fail deployment.

New behaviour

1. Deployment succeeds, and Knox allows access with no authorization since none is configured.
2. Deployment succeeds, but Knox rejects requests with a HTTP 403 response because the configuration is present (indicating that authorization is expected) but invalid.

Inactive topologies

Summary

Knox distinguishes inactive topologies from undeployed topologies.

Previous behaviour

Requests for topologies which are not yet fully deployed result in HTTP 404 responses.

New behaviour

Requests for topologies which are not yet fully deployed result in HTTP 503 responses.

Behavioral Changes in Livy

Behavioral changes denote a marked change in behavior from the previously released version to this version of Apache Livy.

Summary:

The Livy proxy user is taken from Livy for Spark 3's configuration.

Previous behavior:

The custom Kerberos principal configuration was updated via the Livy service.

New behavior:

The Livy proxy user is taken from Livy for Spark 3's configuration, as the Livy service has been replaced with Livy for Spark3 in Cloudera Public Cloud version 7.3.1.

Behavioral Changes in Ranger

Behavioral changes denote a marked change in behavior from the previously released version to this version of Apache Ranger.

Summary: Ranger access audit behavior changes.

Previous behavior:

When you ran `hdfs dfs -copyFromLocal` command, audit logs were generated for the following:

- "write" Access Type and "write" permission.
- "rename" Access Type and "write" permission.
- "rename" Access Type and "write" permission.

When you ran `hdfs dfs -touch` command, audit log was generated for the following:

- "write" Access Type and "write" permission.

New behavior:

When you run `hdfs dfs -copyFromLocal` command, audit logs are generated for the following:

- "create" Access Type and "write" permission.
- "rename" Access Type and "write" permission.

When you run `hdfs dfs -touch` command, audit log is generated for the following:

- "create" Access Type and "write" permission.

Behavioral Changes in Spark

Behavioral changes denote a marked change in behavior from the previously released version to this version of Apache Spark.

Summary:

Spark 2 has been removed from Cloudera Runtime.

Previous behavior:

Spark 2 was the default version in Cloudera Runtime, Spark 3 was available as an add-on parcel.

New behavior:

Spark 3 is the default Spark version in Cloudera Runtime. Spark 2 has been removed and no longer available in 7.3.1.0.

Important:

Spark 3 contains a large number of changes from Spark 2.

Refer to *Upgrading Spark* for more information on upgrading Spark clusters to 7.3.1.0, and *Migrating Spark Applications* for more information on migrating your existing Spark applications between versions 2 and 3.

Related Information

Upgrading Spark

Migrating Spark Applications

Deprecation Notices In Cloudera Runtime 7.3.1

Components and features that will be deprecated or removed in this release or a future release.

Terminology

Items in this section are designated as follows:

Deprecated

Technology that Cloudera is removing in a future Cloudera release. Marking an item as deprecated gives you time to plan for removal in a future Cloudera release.

Moving

Technology that Cloudera is moving from a future Cloudera release and is making available through an alternative Cloudera offering or subscription. Marking an item as moving gives you time to plan for removal in a future Cloudera release and plan for the alternative Cloudera offering or subscription for the technology.

Removed

Technology that Cloudera has removed from Cloudera and is no longer available or supported as of this release. Take note of technology marked as removed since it can potentially affect your upgrade plans.

Removed Components and Product Capabilities

- Apache Spark 2

  Spark 3 is the default Spark version in Cloudera Runtime. Spark 2 (and Livy 2) has been removed and no longer available in 7.3.1

  Important:

  Spark 3 contains a large number of changes from Spark 2.

  Refer to *Upgrading Spark* for more information on upgrading Spark clusters to 7.3.1.0, and *Migrating Spark Applications* for more information on migrating your existing Spark applications between versions 2 and 3.
- Apache Livy 2 (see Deprecation Notices for Apache Livy)
- Apache Zeppelin (see Deprecation Notices for Apache Zeppelin)

Please contact Cloudera Support or your Cloudera Account Team if you have any questions.

Related Information

Upgrading Spark
Migrating Spark Applications

Platform and OS

The listed Operating Systems, databases, and instant client library are deprecated or removed from the 7.3.1 release.

Database Support:

The listed databases are deprecated from the 7.3.1 release:

- None

The following database is removed and no longer supported from the 7.3.1 release:

- PostgreSQL 11

Operating System

The listed operating systems are deprecated from the 7.3.1 release:

- None

The following operating system is removed and no longer supported from the 7.3.1 release:

- RHEL 8.6
- RHEL 7.9

- RHEL 7.9 (FIPS)
- CentOS 7.9
- SLES 12 SP5
- CentOS

Note: CentOS Linux 7 has reached end of life. Ensure to migrate to RHEL/Oracle Linux or any supported operating system before upgrading to 7.3.1.

Deprecation Notices for Apache Kafka

Certain features and functionality in Apache Kafka are deprecated or removed in Cloudera Runtime 7.3.1. You must review these changes along with the information about the features in Kafka that will be removed or deprecated in a future release.

Important: The following list of deprecated and removed items is not exhaustive and only contains items that have a direct and immediate effect on Kafka in CDP. For a full list of deprecation and/or removals in the version Apache Kafka shipped with Runtime, review the *Notable Changes* as well as the *Release Notes* on https://kafka.apache.org/.

Deprecated

MirrorMaker (MM1)

MirrorMaker is deprecated. Cloudera recommends that you use Streams Replication Manager (SRM) instead.

--zookeeper

The --zookeeper option is only supported for the kafka-configs tool and should be only used when updating SCRAM Credential configurations. The --zookeeper option is either deprecated in or removed from other Kafka command line tools. Cloudera recommends that you use the --bootstrap-server option instead.

Deprecation Notices for Apache Livy

Certain features and functionality in Apache Livy are deprecated or removed in Cloudera Runtime 7.3.1. You must review these changes along with the information about the features in Livy that will be removed or deprecated in a future release.

Removed

Apache Livy 2

As Spark 3 is the default Spark version in Cloudera Runtime, Livy 2 has been removed, alongside with Spark 2, and no longer available in 7.3.1

Important:

Spark 3 contains a large number of changes from Spark 2.

For more information on upgrading to Spark 3, refer to Upgrading Spark for more information on upgrading Spark clusters to 7.3.1, and Migrating Spark Applications for more information on migrating your existing Spark applications between versions 2 and 3.

Deprecation Notices for Apache Oozie

Certain features and functionality in Apache Oozie are deprecated or removed in Cloudera Runtime 7.3.1. You must review these changes along with the information about the features in Oozie that will be removed or deprecated in a future release.

Deprecated

Oozie's Spark action

Due to the discontinuation and deprecation of Spark 2 in Cloudera 7.3.1, Cloudera decided to deprecate Oozie Spark actions, which are based on Spark 2. Consequently, Oozie's Spark actions will no longer be available, and if you attempt to execute a Spark action, an error will be raised.

Starting from 7.3.1, you must migrate to Spark 3 to use Spark actions. For more information, see Spark 3 support in Oozie.

Deprecation Notices for Apache Spark

Certain features and functionality in Apache Spark 2 are deprecated or removed in Cloudera Runtime 7.3.1. You must review these changes along with the information about the features in Spark 2 that will be removed or deprecated in a future release.

Removed

Apache Spark 2

Spark 3 is the default Spark version in Cloudera Runtime. Spark 2 has been removed and no longer available in 7.3.1

Important:

Spark 3 contains a large number of changes from Spark 2.

Refer to Upgrading Spark for more information on upgrading Spark clusters to 7.3.1, and Migrating Spark Applications for more information on migrating your existing Spark applications between versions 2 and 3.

Deprecation Notices for Apache Zeppelin

Certain features and functionality in Apache Zeppelin are deprecated or removed in Cloudera Runtime 7.3.1. You must review these changes along with the information about the features in Zeppelin that will be removed or deprecated in a future release.

Removed

Apache Zeppelin

Apache Zeppelin is removed from Cloudera Public Cloud.

Cloudera recommends you back up all existing Zeppelin notebooks before upgrading to version 7.3.1.

Fixed Common Vulnerabilities and Exposures 7.3.1

Common vulnerabilities and Exposures (CVEs) fixed in this release.

- CVE-2023-6378 Logback
- CVE-2023-6481 Logback
- CVE-2023-2976 Google Guava
- CVE-2020-8908 Google Guava
- CVE-2018-10237 Google Guava
- CVE-2023-52428 Nimbus-jose-jwt
- CVE-2023-45865 Akka-actor
- CVE-2021-42697 Akka-http-core
- CVE-2021-23339 Akka-http-core
- CVE-2022-31023 Akka-http-server
- CVE-2021-26291 Apache Maven
- CVE-2022-46337 Apache Derby
- CVE-2023-22006 Graal-sdk
- CVE-2023-50386 Apache Solr
- CVE-2023-50291 Apache Solr
- CVE-2023-50292 Apache Solr
- CVE-2023-50298 Apache Solr
- CVE-2023-1932 Hibernate Validator
- CVE-2024-22201 Eclipse Jetty
- CVE-2024-21634 Amazon Ion
- CVE-2017-7525 Jackson-mapper-asl
- CVE-2019-10172 Jackson-mapper-asl
- CVE-2023-51775 Jose4j
- CVE-2020-15522 Bouncycastle
- CVE-2020-0187 Bouncycastle
- CVE-2022-1471 Snakeyaml
- CVE-2022-25857 Snakeyaml
- CVE-2022-38749 Snakeyaml
- CVE-2022-38751 Snakeyaml
- CVE-2022-38752 Snakeyaml
- CVE-2022-41854 Snakeyaml
- CVE-2022-38750 Snakeyaml
- CVE-2021-31684 Json-smart
- CVE-2023-1370 Json-smart
- CVE-2021-27568 Json-smart
- CVE-2021-4178 Fabric 8 Kubernetes client
- CVE-2023-3635 Okio
- CVE-2024-1597 Postgresql
- CVE-2023-45857 Axios
- CVE-2022-4244 Plexus-utils
- CVE-2022-4245 Plexus-utils
- CVE-2023-34453 Snappy-java
- CVE-2023-34454 Snappy-java
- CVE-2023-34455 Snappy-java
- CVE-2023-43642 Snappy-java
- CVE-2023-34042 Spring Security
- CVE-2024-22257 Spring Security
- CVE-2023-20859 Spring Vault
- CVE-2024-22243 Spring Framework
- CVE-2024-22262 Spring Framework

- CVE-2024-22259 Spring Framework
- CVE-2024-1300 Vertx-core
- CVE-2023-44483 Xmlsec
- CVE-2024-31573 Xmlunit-core
- CVE-2024-38998 Requirejs
- CVE-2024-38999 Requirejs
- CVE-2023-4759 Eclipse Jgit

CDP Public Cloud 7

Ranger API

Date published: 2019-08-22
Date modified: 2022-12-15

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Ranger API Overview **4**
Ranger Admin Metrics API 4

Ranger REST API documentation **7**

Ranger API Overview

The Ranger REST API provides a framework to enable, monitor and manage comprehensive data security across the Cloudera platform.

Apache Ranger currently provides a centralized security administration, fine grain access control and detailed auditing for user access within Apache Hadoop, Apache Hive, Apache HBase and other Apache components. The Ranger REST API documentation is served on the same host and port as the Ranger Admin Web UI, and does not require an extra process or extra configuration. The API supports HTTP Basic Authentication, accepting the same users and credentials as the Ranger Admin Web UI. For more information about REST APIs available for collecting Ranger metrics, see:

Related Information

Apache Ranger REST API

Ranger Admin Metrics API

You can access two APIs that fetch Ranger Admin metrics.

One Ranger Admin API returns a response in JSON format. The other Ranger Admin API returns a response in prometheus compatible format. Neither API that fetches Ranger Admin metrics requires authentication as it does not provide any sensitive information. Both APIs return the same value. You can reduce (or increase) the metrics collection interval using the hadoop-metrics2.properties file.

JSON compatible API

accessible at a url with format:

http(s)://<ADMIN_HOST>:<ADMIN_PORT>/service/metrics/json

Example: https://kg-rzmtr-1.kg-rzmtr.root.hwx.site:6182/service/metrics/json

Prometheus compatible API

accessible at a url with format:

http(s)://<ADMIN_HOST>:<ADMIN_PORT>/service/metrics/prometheus

Example: https://kg-rzmtr-1.kg-rzmtr.root.hwx.site:6182/service/metrics/prometheus

Response

The response is grouped into policy, context enrichers, service, deny conditions, user group and Ranger JVM categories. Since both APIs return the same value, only json responses are discussed in this document.

Policy:

The policy group metrics gives all metrics related to policies. Further it is divided into following sub-categories.

Sample json response follows:

```
"Policy": {
        "RowFilteringCount": 0,
        "MaskingCount": 0,
        "TagAccessCount": 1,
        "ResourceAccessCount": 95,
        "ResourceAccessCountHIVE": 10,
        "ResourceAccessCountKAFKA": 12,
```

```
        "ResourceAccessCountATLAS": 9,
        "ResourceAccessCountKUDU": 3,
        "ResourceAccessCountHDFS": 3,
        ….
    }
```

Policy metric response can further be divided into row filtering, masking, resource access, and tag policies.

Row Filtering: This group of metrics indicates row filtering policy metrics. Currently we only support total row filtering count metrics.

Masking: This group of metrics indicates masking policy metrics. Currently we only support total masking count metrics.

Resource Access: This group of metrics indicates resource access policy metrics. Other than total resource access policy count, it also gives component specific resource access policy count.

Sample json follows:

```
"Policy": {
        ….
        "ResourceAccessCount": 95,
        "ResourceAccessCountHIVE": 10,
        "ResourceAccessCountKAFKA": 12,
        "ResourceAccessCountATLAS": 9,
        "ResourceAccessCountKUDU": 3,
        "ResourceAccessCountHDFS": 3,
        ….
    }
```

ContextEnrichers:

This metrics group represents the available metrics for context enrichers. Similar to the d deny conditions and policy metrics group, this metric also shares count for each service definition/type available other than total count.

Sample json follows:

```
 "ContextEnrichers": {
        "ContextEnricherCountTAG": 1,
        "ContextEnricherCount": 1
    }
```

Service:

This metrics group represents the available metrics for service. Similar to the context enrichers and policy metrics group, this metric also shares count for each service definition/type available other than total count.

Sample json follows:

```
"Service": {
        "ServiceCount": 16,
        "ServiceCountHIVE": 1,
        "ServiceCountTAG": 1,
        "ServiceCountHBASE": 1,
        "ServiceCountADLS": 1,
        "ServiceCountKAFKA": 1,
        "ServiceCountS3": 1,
        ….
    }
```

DenyConditions:

This metrics group represents the available metrics for all deny conditions. Similar to the service and policy metrics group, this metric also shares count for each service definition/type available other than total count.

Sample json follows:

```
"DenyConditions": {
        "DenyConditionCountTAG": 1,
        "DenyConditionCount": 1
    }
```

User Group:

This metrics group represents the available metrics for all users and groups. Other than total user and groups it also includes role specific user counts.

Sample json follows:

```
"UserGroup": {
        "UserCount": 62,
        "GroupCount": 97,
        "UserCountKeyAdmin": 1,
        "UserCountKeyAdminAuditor": 0,
        "UserCountSysAdmin": 8,
        "UserCountAdminAuditor": 0,
        "UserCountUser": 53
    }
```

Ranger JVM:

This metrics group represents the JVM metrics needed to get internal state of admin.

Sample json follows:

```
 "RangerJvm": {
        "GcTimeTotal": 5315,
        "SystemLoadAvg": 3.91,
        "ThreadsBusy": 2250,
        "GcCountTotal": 472,
        "MemoryMax": 1066401792,
        "MemoryCurrent": 179763416,
        "ThreadsWaiting": 2248,
        "ProcessorsAvailable": 4,
        "GcTimeMax": 5002,
        "ThreadsBlocked": 25,
        "ThreadsRemaining": 4
    }
}
```

Table 1: Ranger JVM Admin Metrics Descriptions

Metric	Descriptions
GcTimeTotal	Total time taken by JVM Garbage Collector.
GcTimeMax	Maximum time taken by a single JVM Garbage Collection process.
GcCountTotal	Number of times JVM Garbage Collector runs.
SystemLoadAvg	Average load on the host on which the service is running.
ThreadsBusy	Total JVM threads which are busy.

Metric	Descriptions
ThreadsWaiting	Total JVM threads which are waiting.
ThreadsBlocked	Total JVM threads which are blocked.
ThreadsRemaining	Total remaining (not busy/ waiting/ blocked) JVM threads.
MemoryMax	Maximum memory (RAM) JVM can get from the host.
MemoryCurrent	Currently occupied memory (RAM) by JVM.
ProcessorsAvailable	Currently available processor cores on host.

Ranger REST API documentation

The Ranger Admin Web UI includes REST API documentation which describes the framework used to enable, monitor and manage comprehensive data security across the Hadoop platform. The Ranger REST API is published as html, using the Swagger UI, at a url that has the following format:

Ranger Admin SSL-enabled (Default Port: 6182):

https://<Ranger-host-name>:6182/apidocs/index.html

Ranger Admin not SSL-enabled (Default Port: 6080):

http://<Ranger-host-name>:6080/apidocs/index.html

You can access the Ranger API documentation using the Swagger user interface from Cloudera Manager. From Cloudera Manager, go to Ranger Ranger Admin Web UI .

To view the Ranger REST API documentation:

1. Log in to the Ranger Admin Web UI.
2. Open the Ranger Admin menu.
3. Select API Documentation.

Ranger Rest API documentation displays as html output in your web browser.

Cloudera Runtime 7.3.1

Running Apache Spark Applications

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction 5

Apache Spark 3.4 Requirements 5

Running Apache Spark 3.4 applications 6

Running your first Spark application 7

Running sample Spark applications 8

Configuring Spark Applications 9
- Configuring Spark application properties in spark-defaults.conf 10
- Configuring Spark application logging properties 11

Submitting Spark applications 11
- spark-submit command options 12
- Spark cluster execution overview 12
- Canary test for pyspark command 13
- Fetching Spark Maven dependencies 13
- Accessing the Spark History Server 14

Running Spark applications on YARN 14
- Spark on YARN deployment modes 14
- Submitting Spark Applications to YARN 16
- Monitoring and Debugging Spark Applications 16
- Example: Running SparkPi on YARN 17
- Configuring Spark on YARN Applications 17
- Dynamic allocation 18

Submitting Spark applications using Livy 19
- Using the Livy API to run Spark jobs 19
- Submitting a Spark job to a cluster using Livy 20
- Running an interactive session with the Livy REST API 20
  - Livy objects for interactive sessions 21
  - Setting Python path variables for Livy 23
  - Livy API reference for interactive sessions 23
- Submitting batch applications using the Livy REST API 25
  - Livy batch object 26
  - Livy API reference for batch jobs 26
- Submitting a Spark job to a cluster using Livy 27
- Configuring the Livy Thrift Server 28

Connecting to the Apache Livy Thrift Server.....28
Using Livy with Spark.....30

Using PySpark.....30
Running PySpark in a virtual environment.....30
Running Spark Python applications.....31

Automating Spark Jobs with Oozie Spark Action.....33

Introduction

You can run Spark applications locally or distributed across a cluster, either by using an interactive shell or by submitting an application. Running Spark applications interactively is commonly performed during the data-exploration phase and for ad hoc analysis.

You can:

- Submit interactive statements through the Scala, Python, or R shell, or through a high-level notebook such as Zeppelin.
- Use APIs to create a Spark application that runs interactively or in batch mode, using Scala, Python, R, or Java.

Because of a limitation in the way Scala compiles code, some applications with nested definitions running in an interactive shell may encounter a Task not serializable exception. Cloudera recommends submitting these applications.

To run applications distributed across a cluster, Spark requires a cluster manager. supports only the YARN cluster manager. When run on YARN, Spark application processes are managed by the YARN ResourceManager and NodeManager roles. Spark Standalone is not supported.

To launch Spark applications on a cluster, you can use the spark-submit script in the /bin directory on a gateway host. You can also use the API interactively by launching an interactive shell for Scala (spark-shell), Python (pyspark), or SparkR. Note that each interactive shell automatically creates SparkContext in a variable called sc, and SparkSession in a variable called spark. For more information about spark-submit, see the Apache Spark documentation Submitting Applications.

Alternately, you can use Livy to submit and manage Spark applications on a cluster. Livy is a Spark service that allows local and remote applications to interact with Apache Spark over an open source REST interface. Livy offers additional multi-tenancy and security functionality. For more information about using Livy to run Spark Applications, see Submitting Spark applications using Livy on page 19.

Related Information

Migrating Spark applications

Deprecation Notices in

Apache Spark 3.4 Requirements

The following sections describe software and hardware requirements for Spark 3.4

Software requirements

Each cluster host must have the following software installed:

Java

JDK 8 or JDK 17. Remove other JDK versions from all cluster and gateway hosts to ensure proper operation.

Scala

Scala 2.12

Python

Python 3.7 and higher

Hardware requirements

Spark 3 has no specific hardware requirements on top of what is required for deployments.

Running Apache Spark 3.4 applications

You can run Apache Spark 3 applications locally or distributed across a cluster, either by using an interactive shell or by submitting an application. Running Spark applications interactively is commonly performed during the data-exploration phase and for ad hoc analysis.

Note: Spark 3.3 is the first Spark version with the log4j2 dependency. Previous versions contained the log4j1 dependency. If you are using custom logging related changes, you must rewrite the original log4j properties files using log4j2 syntax such as XML, JSON, YAML, or properties format.

Unsupported connectors

This release does not support the following connectors:

- SparkR

Limitations of Spark in

Limitations of Spark (in comparison to Apache Spark 3.3) in are described below:

- spark.sql.orc.compression.codec config doesn't accept zsdt value.

Canary test for pyspark command

The following example shows a simple pyspark session that refers to the SparkContext, calls the collect() function which runs a Spark 3 job, and writes data to HDFS. This sequence of operations helps to check if there are obvious configuration issues that prevent Spark 3 jobs from working at all. For the HDFS path for the output directory, substitute a path that exists on your own system.

```
$ hdfs dfs -mkdir /user/jdoe/spark
$ pyspark
...
SparkSession available as 'spark'.
>>> strings = ["one","two","three"]
>>> s2 = sc.parallelize(strings)
>>> s3 = s2.map(lambda word: word.upper())
>>> s3.collect()
['ONE', 'TWO', 'THREE']
>>> s3.saveAsTextFile('hdfs:///user/jdoe/spark/canary_test')
>>> quit()
$ hdfs dfs -ls /user/jdoe/spark
Found 1 items
drwxr-xr-x   - jdoe spark-users  0 2016-08-26 14:41 /user/jdoe/spark/cana
ry_test
$ hdfs dfs -ls /user/jdoe/spark/canary_test
Found 3 items
-rw-r--r--   3 jdoe spark-users  0 2016-08-26 14:41 /user/jdoe/spark/canar
y_test/_SUCCESS
-rw-r--r--   3 jdoe spark-users  4 2016-08-26 14:41 /user/jdoe/spark/canary_
test/part-00000
-rw-r--r--   3 jdoe spark-users 10 2016-08-26 14:41 /user/jdoe/spark/canary_
test/part-00001
$ hdfs dfs -cat /user/jdoe/spark/canary_test/part-00000
ONE
$ hdfs dfs -cat /user/jdoe/spark/canary_test/part-00001
TWO
THREE
```

Fetching Spark 3 Maven Dependencies

The Maven coordinates are a combination of groupId, artifactId and version. The groupId and artifactId are the same as for the upstream Apache Spark project. For example, for spark-core, groupId is org.apache.spark, and artifactId is spark-core_2.12, both the same as the upstream project. The version is different for the packaging, it contains the upstream Spark version as well as the version: *[**ARTIFACTID***]-[***UPSTREAM SPARK VERSION***].[***CDP VERSION***]-[***CLOUDERA BUILD NUMBER***]*. For example, spark-core_2.12-3.3.0.7.2.16.0-123.

Accessing the Spark 3 History Server

The Spark 3 history server is available on port 18089, rather than port 18088 as with the Spark 2 history server.

Running your first Spark application

About this task

Important:

By default, is configured to permit any user to access the Hive Metastore. However, if you have modified the value set for the configuration property hadoop.proxyuser.hive.groups, which can be modified in by setting the Hive Metastore Access Control and Proxy User Groups Override property, your Spark application might throw exceptions when it is run. To address this issue, make sure you add the groups that contain the Spark users that you want to have access to the metastore when Spark applications are run to this property in :

1. In the Home page, click the Hive service.
2. On the Hive service page, click the Configuration tab.
3. In the Search well, type hadoop.proxyuser.hive.groups to locate the Hive Metastore Access Control and Proxy User Groups Override property.
4. Click the plus sign (+), enter the groups you want to have access to the metastore, and then click Save Changes. You must restart the Hive Metastore Server for the changes to take effect by clicking the restart icon at the top of the page.

The simplest way to run a Spark application is by using the Scala or Python shells.

Procedure

1. To start one of the shell applications, run one of the following commands:
   - Scala:

```
$ /bin/spark-shell
...
Spark context Web UI available at ...
Spark context available as 'sc' (master = yarn, app id = ...).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version ...
      /_/
Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_191)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

- Python:

```
$ /bin/pyspark
Python 2.7.5 (default, Sep 12 2018, 05:31:16)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
...
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version ...
      /_/

Using Python version 2.7.5 (default, Sep 12 2018 05:31:16)
SparkSession available as 'spark'.
>>>
```

SPARK_HOME defaults to /opt/cloudera/parcels/CDH/lib/spark in parcel installations. In a deployment, the shells are also available from /usr/bin.

For a complete list of shell options, run spark-shell or pyspark with the -h flag.

2. To run the classic Hadoop word count application, copy an input file to HDFS:

```
hdfs dfs -copyFromLocal INPUT s3a://<BUCKET_NAME>/
```

3. Within a shell, run the word count application using the following code examples, substituting for *NAMENODE_HOST*, *PATH/TO/INPUT*, and *PATH/TO/OUTPUT*:
Scala:

```
scala> val myfile = sc.textFile("s3a://<BUCKET_NAME>/PATH/TO/INPUT")
scala> val counts = myfile.flatMap(line => line.split(" ")).map(word => (
word, 1)).reduceByKey(_ + _)
scala> counts.saveAsTextFile("s3a://<BUCKET_NAME>/PATH/TO/OUTPUT")
```

Python:

```
>>> myfile = sc.textFile("s3a://BUCKET_NAME/PATH/TO/INPUT")
>>> counts = myfile.flatMap(lambda line: line.split(" ")).map(lambda word:
 (word, 1)).reduceByKey(lambda v1,v2: v1 + v2)
>>> counts.saveAsTextFile("s3a://<BUCKET_NAME>/PATH/TO/OUTPUT")
```

Running sample Spark applications

About this task

You can use the following sample Spark Pi and Spark WordCount sample programs to validate your Spark installation and explore how to run Spark jobs from the command line and Spark shell.

Spark Pi

You can test your Spark installation by running the following compute-intensive example, which calculates pi by “throwing darts” at a circle. The program generates points in the unit square ((0,0) to (1,1)) and counts how many points fall within the unit circle within the square. The result approximates pi.

Follow these steps to run the Spark Pi example:

1. Authenticate using kinit:

```
kinit <USERNAME>
```

2. Run the Apache Spark Pi job in yarn-client mode, using code from org.apache.spark:

```
spark-submit --class org.apache.spark.examples.SparkPi \
    --master yarn-client \
    --num-executors 1 \
    --driver-memory 512m \
    --executor-memory 512m \
    --executor-cores 1 \
    examples/jars/spark-examples*.jar 10
```

Commonly used options include the following:

--class

The entry point for your application: for example, org.apache.spark.examples.SparkPi.

--master

The master URL for the cluster: for example, spark://23.195.26.187:7077.

--deploy-mode

Whether to deploy your driver on the worker nodes (cluster) or locally as an external client (default is client).

--conf

Arbitrary Spark configuration property in key=value format. For values that contain spaces, enclose “key=value” in double quotation marks.

<application-jar>

Path to a bundled jar file that contains your application and all dependencies. The URL must be globally visible inside of your cluster: for instance, an hdfs:// path or a file:// path that is present on all nodes.

<application-arguments>

Arguments passed to the main method of your main class, if any.

Your job should produce output similar to the following. Note the value of pi in the output.

```
17/03/22 23:21:10 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.s
cala:38, took 1.302805 s
Pi is roughly 3.1445191445191445
```

You can also view job status in a browser by navigating to the YARN ResourceManager Web UI and viewing job history server information. (For more information about checking job status and history, see "Tuning Spark" in this guide.)

Configuring Spark Applications

You can specify Spark application configuration properties as follows:

- Pass properties using the --conf command-line switch; for example:

```
spark-submit \
--class com.cloudera.example.YarnExample \
--master yarn \
```

```
--deploy-mode cluster \
--conf "spark.eventLog.dir=hdfs:///user/spark/eventlog" \
lib/yarn-example.jar \
10
```

- Specify properties in spark-defaults.conf.
- Pass properties directly to the SparkConf used to create the SparkContext in your Spark application; for example:
  - Scala:

    ```
    val conf = new SparkConf().set("spark.dynamicAllocation.initialExecutors
    ", "5")
    val sc = new SparkContext(conf)
    ```

  - Python:

    ```
    from pyspark import SparkConf, SparkContext
    from pyspark.sql import SQLContext
    conf = (SparkConf().setAppName('Application name'))
    conf.set('spark.hadoop.avro.mapred.ignore.inputs.without.extension',
    'false')
    sc = SparkContext(conf = conf)
    sqlContext = SQLContext(sc)
    ```

The order of precedence in configuration properties is:

1. Properties passed to SparkConf.
2. Arguments passed to spark-submit, spark-shell, or pyspark.
3. Properties set in spark-defaults.conf.

For more information, see Spark Configuration.

Configuring Spark application properties in spark-defaults.conf

Specify properties in the spark-defaults.conf file in the form property=value.

To create a comment, add a hash mark (#) at the beginning of a line. You cannot add comments to the end or middle of a line.

This example shows an example excerpt of a spark-defaults.conf file:

```
spark.authenticate=false
spark.driver.log.dfsDir=/user/spark/driverLogs
spark.driver.log.persistToDfs.enabled=true
spark.dynamicAllocation.enabled=true
spark.dynamicAllocation.executorIdleTimeout=60
spark.dynamicAllocation.minExecutors=0
spark.dynamicAllocation.schedulerBacklogTimeout=1
spark.eventLog.enabled=true
spark.io.encryption.enabled=false
spark.lineage.enabled=true
spark.network.crypto.enabled=false
spark.serializer=org.apache.spark.serializer.KryoSerializer
spark.shuffle.service.enabled=true
spark.shuffle.service.port=7447
spark.ui.enabled=true
spark.ui.killEnabled=true
spark.kerberos.access.hadoopFileSystems=s3a://bucket1,s3a://bucket2
```

recommends placing configuration properties that you want to use for every application in spark-defaults.conf. See Application Properties for more information.

Configuring Spark application logging properties

Before you begin

Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

About this task

To configure only the logging threshold level, follow the procedure in Configuring Logging Thresholds. To configure any other logging property, do the following:

Procedure

1. In the , go to Data Hub Clusters.
2. Find and select the cluster you want to configure.
3. Click the link for the URL.
4. Go to the Spark service.
5. Click the Configuration tab.
6. Select Scope Gateway .
7. Select Category Advanced .
8. Locate the Spark Client Advanced Configuration Snippet (Safety Valve) for spark-conf/log4j.properties.
9. Specify log4j properties. If more than one role group applies to this configuration, edit the value for the appropriate role group. See *Modifying Configuration Properties Using* .
10. Enter a Reason for change, and then click Save Changes to commit the changes.
11. Deploy the client configuration.

Submitting Spark applications

To submit an application consisting of a Python file or a compiled and packaged Java or Spark JAR, use the spark-submit script.

spark-submit Syntax

```
spark-submit --OPTION VALUE \
  APPLICATION JAR | PYTHON FILE [APPLICATION ARGUMENTS]
```

Example: Running SparkPi on YARN demonstrates how to run one of the sample applications, SparkPi, packaged with Spark. It computes an approximation to the value of pi.

Table 1: spark-submit Arguments

Option	Description
APPLICATION JAR	Path to a JAR file containing a Spark application. For the client deployment mode, the path must point to a local file. For the cluster deployment mode, the path can be either a local file or a URL globally visible inside your cluster; see Advanced Dependency Management.
PYTHON FILE	Path to a Python file containing a Spark application. For the client deployment mode, the path must point to a local file. For the cluster deployment mode, the path can be either a local file or a URL globally visible inside your cluster; see Advanced Dependency Management.
APPLICATION ARGUMENTS	Arguments to pass to the main method of your application.

spark-submit command options

You specify spark-submit options using the form --*OPTION VALUE* instead of --*OPTION=VALUE* . (Use a space instead of an equals sign.)

Option	Description
class	For Java and Scala applications, the fully qualified classname of the class containing the main method of the application. For example, org.apache.spark.examples.SparkPi.
conf	Spark configuration property in *KEY=VALUE* format. For values that contain spaces, surround "*KEY=VALUE*" with quotes (as shown).
deploy-mode	Deployment mode: *cluster* and *client*. In cluster mode, the driver runs on worker hosts. In client mode, the driver runs locally as an external client. Use cluster mode with production jobs; client mode is more appropriate for interactive and debugging uses, where you want to see your application output immediately. To see the effect of the deployment mode when running on YARN, see Deployment Modes. Default: client.
driver-class-path	Configuration and classpath entries to pass to the driver. JARs added with --jars are automatically included in the classpath.
driver-cores	Number of cores used by the driver in cluster mode. Default: 1.
driver-memory	Maximum heap size (represented as a JVM string; for example 1024m, 2g, and so on) to allocate to the driver. Alternatively, you can use the spark.driver.memory property.
files	Comma-separated list of files to be placed in the working directory of each executor. For the client deployment mode, the path must point to a local file. For the cluster deployment mode, the path can be either a local file or a URL globally visible inside your cluster; see Advanced Dependency Management.
jars	Additional JARs to be loaded in the classpath of drivers and executors in cluster mode or in the executor classpath in client mode. For the client deployment mode, the path must point to a local file. For the cluster deployment mode, the path can be either a local file or a URL globally visible inside your cluster; see Advanced Dependency Management.
master	The location to run the application.
packages	Comma-separated list of Maven coordinates of JARs to include on the driver and executor classpaths. The local Maven, Maven central, and remote repositories specified in repositories are searched in that order. The format for the coordinates is groupId:artifactId:version.
py-files	Comma-separated list of .zip, .egg, or .py files to place on PYTHONPATH. For the client deployment mode, the path must point to a local file. For the cluster deployment mode, the path can be either a local file or a URL globally visible inside your cluster; see Advanced Dependency Management.
repositories	Comma-separated list of remote repositories to search for the Maven coordinates specified in packages.

Table 2: Master Values

Master	Description
local	Run Spark locally with one worker thread (that is, no parallelism).
local[*K*]	Run Spark locally with *K* worker threads. (Ideally, set this to the number of cores on your host.)
local[*]	Run Spark locally with as many worker threads as logical cores on your host.
yarn	Run using a YARN cluster manager. The cluster location is determined by *HADOOP_CONF_DIR* or *YARN_CONF_DIR*. See Configuring the Environment.

Spark cluster execution overview

Spark orchestrates its operations through the driver program. When the driver program is run, the Spark framework initializes executor processes on the cluster hosts that process your data. The following occurs when you submit a Spark application to a cluster:

1. The driver is launched and invokes the main method in the Spark application.
2. The driver requests resources from the cluster manager to launch executors.
3. The cluster manager launches executors on behalf of the driver program.
4. The driver runs the application. Based on the transformations and actions in the application, the driver sends tasks to executors.
5. Tasks are run on executors to compute and save results.
6. If dynamic allocation is enabled, after executors are idle for a specified period, they are released.
7. When driver's main method exits or calls SparkContext.stop, it terminates any outstanding executors and releases resources from the cluster manager.

Canary test for pyspark command

The following example shows a simple `pyspark` session that refers to the SparkContext, calls the collect() function which runs a Spark 2 job, and writes data to HDFS. This sequence of operations helps to check if there are obvious configuration issues that prevent Spark jobs from working at all. For the HDFS path for the output directory, substitute a path that exists on your own system.

```
$ hdfs dfs -mkdir /user/systest/spark
$ pyspark
...
SparkSession available as 'spark'.
>>> strings = ["one","two","three"]
>>> s2 = sc.parallelize(strings)
>>> s3 = s2.map(lambda word: word.upper())
>>> s3.collect()
['ONE', 'TWO', 'THREE']
>>> s3.saveAsTextFile('hdfs:///user/systest/spark/canary_test')
>>> quit()
$ hdfs dfs -ls /user/systest/spark
Found 1 items
drwxr-xr-x   - systest supergroup          0 2016-08-26 14:41 /user/systest/
spark/canary_test
$ hdfs dfs -ls /user/systest/spark/canary_test
Found 3 items
-rw-r--r--   3 systest supergroup          0 2016-08-26 14:41 /user/systest/
spark/canary_test/_SUCCESS
-rw-r--r--   3 systest supergroup          4 2016-08-26 14:41 /user/systest/
spark/canary_test/part-00000
-rw-r--r--   3 systest supergroup         10 2016-08-26 14:41 /user/systest/
spark/canary_test/part-00001
$ hdfs dfs -cat /user/systest/spark/canary_test/part-00000
ONE
$ hdfs dfs -cat /user/systest/spark/canary_test/part-00001
TWO
THREE
```

Fetching Spark Maven dependencies

The Maven coordinates are a combination of groupId, artifactId and version. The groupId and artifactId are the same as for the upstream Apache Spark project. For example, for spark-core, groupId is org.apache.spark, and artifactId is spark-core_2.12, both the same as the upstream project. The version is different for the packaging: see the release notes for the exact name depending on which release you are using.

Accessing the Spark History Server

You can access the Spark History Server for your Spark cluster from the interface.

Procedure

1. In the , navigate to your Spark cluster: Data Hub ClustersCluster Name>
2. Select the Gateway tab.
3. Click the URL for Spark History Server.

Running Spark applications on YARN

When Spark applications run on a YARN cluster manager, resource management, scheduling, and security are controlled by YARN.

Spark on YARN deployment modes

In YARN, each application instance has an ApplicationMaster process, which is the first container started for that application. The application is responsible for requesting resources from the ResourceManager. Once the resources are allocated, the application instructs NodeManagers to start containers on its behalf. ApplicationMasters eliminate the need for an active client: the process starting the application can terminate, and coordination continues from a process managed by YARN running on the cluster.

Cluster Deployment Mode

In cluster mode, the Spark driver runs in the ApplicationMaster on a cluster host. A single process in a YARN container is responsible for both driving the application and requesting resources from YARN. The client that launches the application does not need to run for the lifetime of the application.

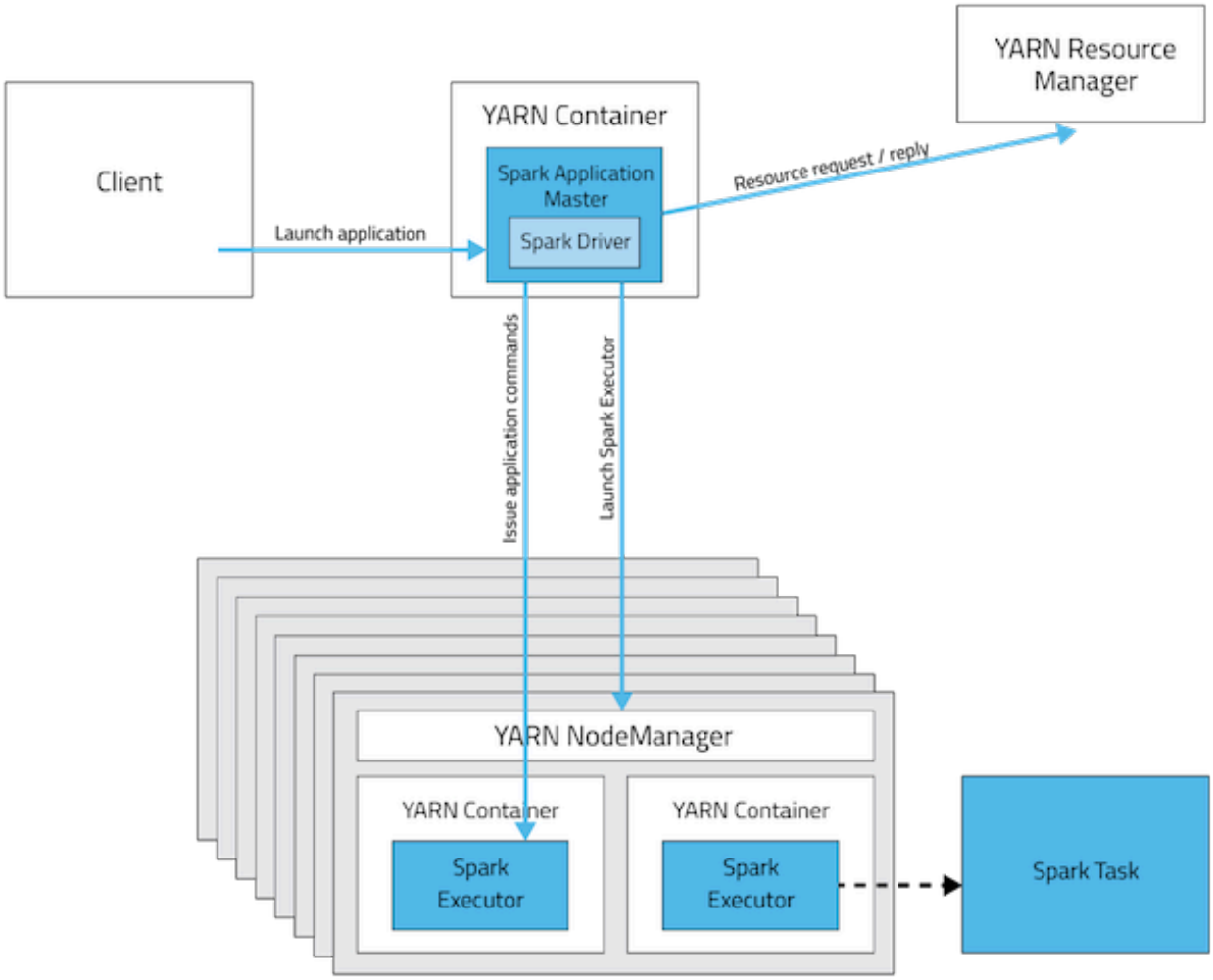

Cluster mode is not well suited to using Spark interactively. Spark applications that require user input, such as spark-shell and pyspark, require the Spark driver to run inside the client process that initiates the Spark application.

Client Deployment Mode

In client mode, the Spark driver runs on the host where the job is submitted. The ApplicationMaster is responsible only for requesting executor containers from YARN. After the containers start, the client communicates with the containers to schedule work.

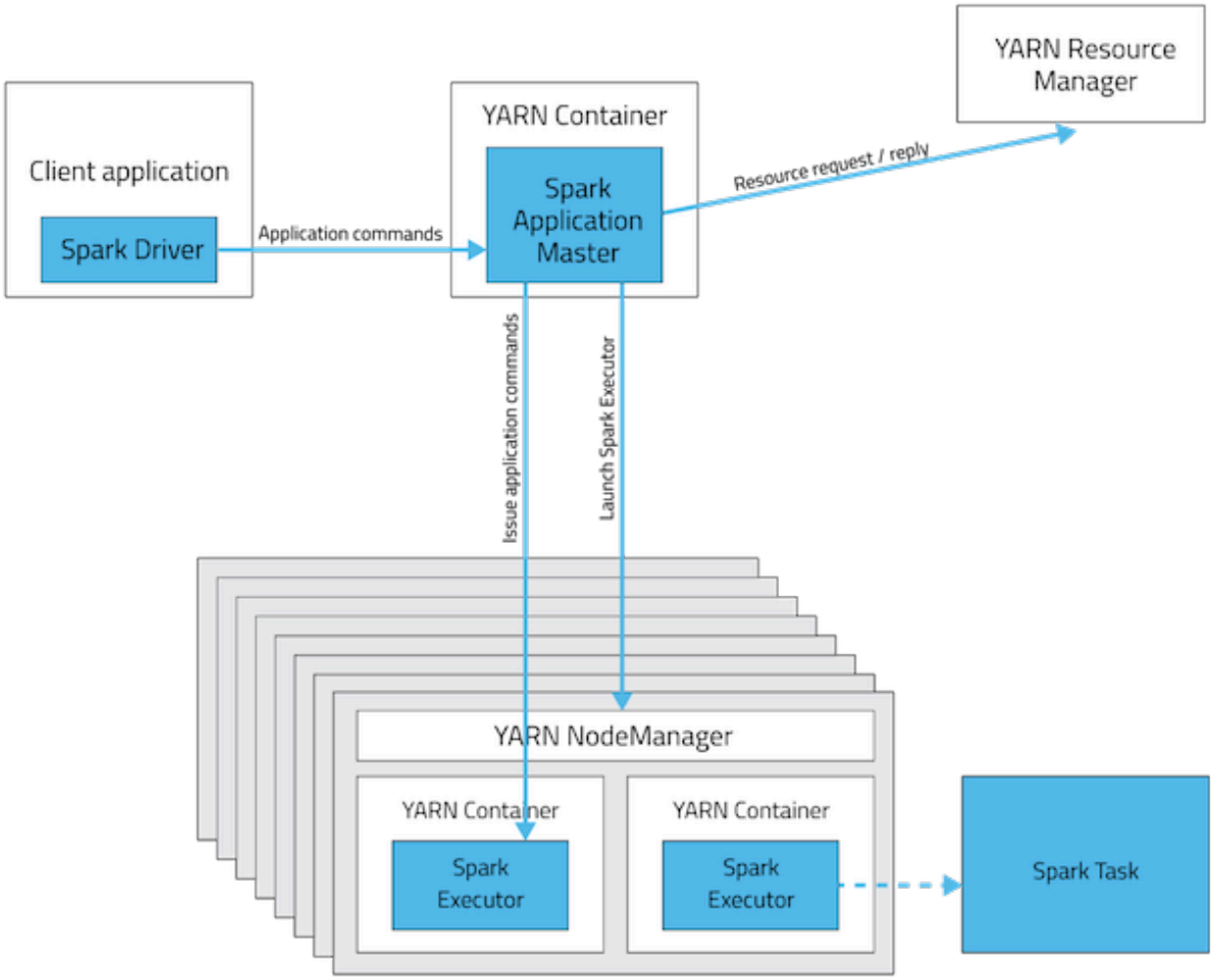

Table 3: Deployment Mode Summary

Mode	YARN Client Mode	YARN Cluster Mode
Driver runs in	Client	ApplicationMaster
Requests resources	ApplicationMaster	ApplicationMaster
Starts executor processes	YARN NodeManager	YARN NodeManager
Persistent services	YARN ResourceManager and NodeManagers	YARN ResourceManager and NodeManagers
Supports Spark Shell	Yes	No

Submitting Spark Applications to YARN

To submit an application to YARN, use the spark-submit script and specify the --master yarn flag. For other spark-submit options, see spark-submit command options on page 12.

Monitoring and Debugging Spark Applications

To obtain information about Spark application behavior you can consult YARN logs and the Spark web application UI. These two methods provide complementary information. For information how to view logs created by Spark applications and the Spark web application UI, see Monitoring Spark Applications.

Example: Running SparkPi on YARN

These examples demonstrate how to use spark-submit to submit the SparkPi Spark example application with various options. In the examples, the argument passed after the JAR controls how close to pi the approximation should be.

In a deployment, *SPARK_HOME* defaults to /opt/cloudera/parcels/CDH/lib/spark. The shells are also available from /bin.

Running SparkPi in YARN Cluster Mode

To run SparkPi in cluster mode:

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \
--deploy-mode cluster /opt/cloudera/parcels/CDH/jars/spark-examples*.jar 10
```

The command prints status until the job finishes or you press `control-C`. Terminating the spark-submit process in cluster mode does not terminate the Spark application as it does in client mode. To monitor the status of the running application, run yarn application -list.

Running SparkPi in YARN Client Mode

To run SparkPi in client mode:

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \
--deploy-mode client SPARK_HOME/lib/spark-examples.jar 10
```

Running Python SparkPi in YARN Cluster Mode

1. Unpack the Python examples archive:

```
sudo su gunzip SPARK_HOME/lib/python.tar.gz
sudo su tar xvf SPARK_HOME/lib/python.tar
```

2. Run the pi.py file:

```
spark-submit --master yarn --deploy-mode cluster SPARK_HOME/lib/pi.py 10
```

Configuring Spark on YARN Applications

In addition to spark-submit Options, options for running Spark applications on YARN are listed in spark-submit on YARN Options.

Table 4: spark-submit on YARN Options

Option	Description
archives	Comma-separated list of archives to be extracted into the working directory of each executor. For the client deployment mode, the path must point to a local file. For the cluster deployment mode, the path can be either a local file or a URL globally visible inside your cluster; see Advanced Dependency Management.
executor-cores	Number of processor cores to allocate on each executor. Alternatively, you can use the spark.executor.cores property.
executor-memory	Maximum heap size to allocate to each executor. Alternatively, you can use the spark.executor.memory property.
num-executors	Total number of YARN containers to allocate for this application. Alternatively, you can use the spark.executor.instances property. If dynamic allocation is enabled, the initial number of executors is the greater of this value or the spark.dynamicAllocation.initialExecutors value.

Option	Description
queue	YARN queue to submit to. For more information, see Assigning Applications and Queries to Resource Pools. Default: default.

During initial installation, tunes properties according to your cluster environment.

In addition to the command-line options, the following properties are available:

Property	Description
spark.yarn.driver.memoryOverhead	Amount of extra off-heap memory that can be requested from YARN per driver. Combined with spark.driver.memory, this is the total memory that YARN can use to create a JVM for a driver process.
spark.yarn.executor.memoryOverhead	Amount of extra off-heap memory that can be requested from YARN, per executor process. Combined with spark.executor.memory, this is the total memory YARN can use to create a JVM for an executor process.

Dynamic allocation

Dynamic allocation allows Spark to dynamically scale the cluster resources allocated to your application based on the workload. When dynamic allocation is enabled and a Spark application has a backlog of pending tasks, it can request executors. When the application becomes idle, its executors are released and can be acquired by other applications.

In , dynamic allocation is enabled by default. The table below describes properties to control dynamic allocation.

To disable dynamic allocation, set spark.dynamicAllocation.enabled to false. If you use the --num-executors command-line argument or set the spark.executor.instances property when running a Spark application, the number of initial executors is the greater of spark.executor.instances or spark.dynamicAllocation.initialExecutors.

For more information on how dynamic allocation works, see resource allocation policy.

When Spark dynamic resource allocation is enabled, all resources are allocated to the first submitted job available causing subsequent applications to be queued up. To allow applications to acquire resources in parallel, allocate resources to pools and run the applications in those pools and enable applications running in pools to be preempted.

If you are using Spark Streaming, see the recommendation in Spark Streaming and Dynamic Allocation.

Table 5: Dynamic Allocation Properties

Property	Description
spark.dynamicAllocation.executorIdleTimeout	The length of time executor must be idle before it is removed. Default: 60 s.
spark.dynamicAllocation.enabled	Whether dynamic allocation is enabled. Default: true.
spark.dynamicAllocation.initialExecutors	The initial number of executors for a Spark application when dynamic allocation is enabled. If spark.executor.instances (or its equivalent command-line argument, --num-executors) is set to a higher number, that number is used instead. Default: 1.
spark.dynamicAllocation.minExecutors	The lower bound for the number of executors. Default: 0.
spark.dynamicAllocation.maxExecutors	The upper bound for the number of executors. Default: Integer.MAX_VALUE.

Property	Description
spark.dynamicAllocation.schedulerBacklogTimeout	The length of time pending tasks must be backlogged before new executors are requested. Default: 1 s.

Submitting Spark applications using Livy

Apache Livy is a Spark service that allows local and remote applications to interact with Apache Spark over a REST interface.

You can use Livy to submit and manage Spark jobs on a cluster. Livy extends Spark capabilities, offering additional multi-tenancy and security features. Applications can run code inside Spark without needing to maintain a local Spark context.

Features include the following:

- Jobs can be submitted from anywhere, using the REST API.
- Livy supports user impersonation: the Livy server submits jobs on behalf of the user who submits the requests. Multiple users can share the same server ("user impersonation" support). This is important for multi-tenant environments, and it avoids unnecessary permission escalation.
- Livy supports security features such as Kerberos authentication and wire encryption.
  - REST APIs are backed by SPNEGO authentication, which the requested user should get authenticated by Kerberos at first.
  - RPCs between Livy Server and Remote SparkContext are encrypted with SASL.
  - The Livy server uses keytabs to authenticate itself to Kerberos.

Livy supports programmatic and interactive access to Spark with Scala. For example, you can:

- Use an interactive notebook to access Spark through Livy.
- Develop a Scala, Java, or Python client that uses the Livy API. The Livy REST API supports full Spark functionality including SparkSession, and SparkSession with Hive enabled.
- Run an interactive session, provided by spark-shell, PySpark, or SparkR REPLs.
- Submit batch applications to Spark.

Code runs in a Spark context, either locally or in YARN; YARN cluster mode is recommended.

Using the Livy API to run Spark jobs

Using the Livy API to run Spark jobs is similar to using the original Spark API. Livy provides programmatic API and REST API.

Programmatic API

Livy provides a programmatic Java/Scala and Python API that allows applications to run code inside Spark without having to maintain a local Spark context. For more information, see *Using the Programmatic API* documentation.

REST API

Architecturally, the client creates a remote Spark cluster, initializes it, and submits jobs through REST APIs. The Livy server unwraps and rewraps the job, and then sends it to the remote SparkContext through RPC. While the job runs the client waits for the result, using the same path. The following diagram illustrates the process:

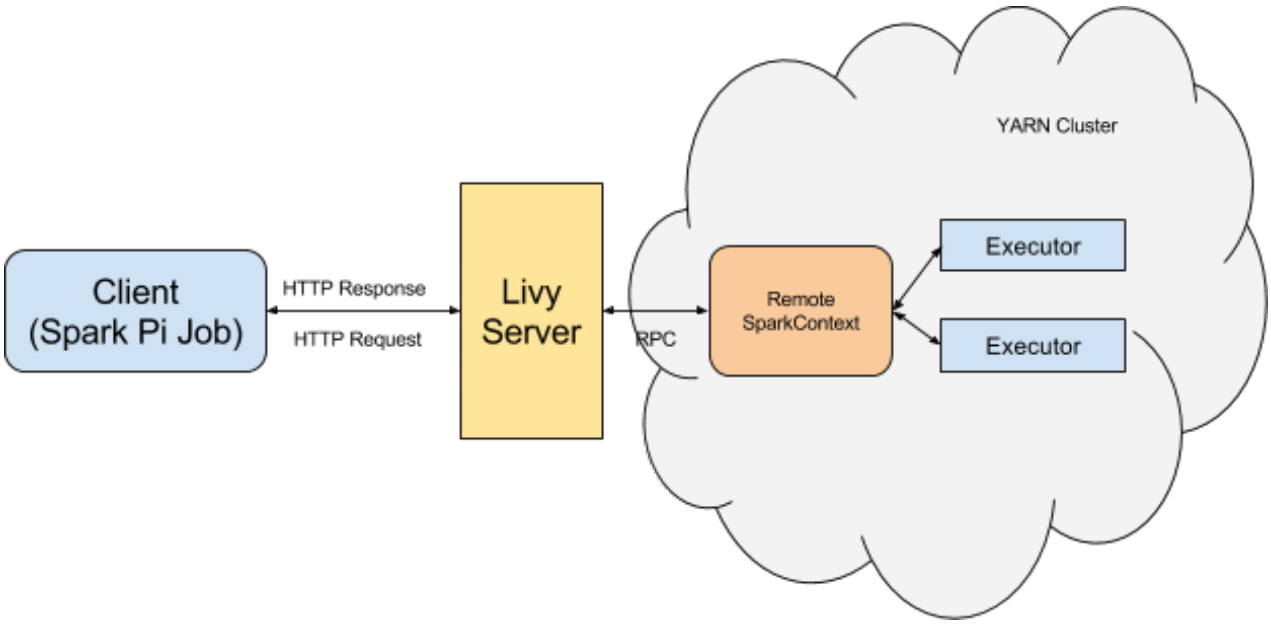

Related Information

Using the Programmatic API

Submitting a Spark job to a cluster using Livy

clusters using definitions such as can accept Spark job execution requests using Livy.

Example

For an example of how to submit a request from a Windows computer to a cluster running on AWS, see How to submit a Spark job to a cluster using Livy.

Running an interactive session with the Livy REST API

About this task

Running an interactive session with Livy is similar to using Spark shell or PySpark, but the shell does not run locally. Instead, it runs in a remote cluster, transferring data back and forth through a network.

The Livy REST API supports GET, POST, and DELETE calls for interactive sessions.

The following example shows how to create an interactive session, submit a statement, and retrieve the result of the statement; the return ID could be used for further queries.

Procedure

1. Create an interactive session. The following POST request starts a new Spark cluster with a remote Spark interpreter; the remote Spark interpreter is used to receive and run code snippets, and return the result.

```
POST /sessions
        host = 'http://localhost:8998'
        data = {'kind': 'spark'}
        headers = {'Content-Type': 'application/json'}
        r = requests.post(host + '/sessions', data=json.dumps(data), heade
rs=headers)
        r.json()
```

```
{u'state': u'starting', u'id': 0, u'kind': u'spark'}
```

2. Submit a statement. The following POST request submits a code snippet to a remote Spark interpreter, and returns a statement ID for querying the result after execution is finished.

```
POST /sessions/{sessionId}/statements
        data = {'code': 'sc.parallelize(1 to 10).count()'}
        r = requests.post(statements_url, data=json.dumps(data), header
s=headers)
        r.json()
{u'output': None, u'state': u'running', u'id': 0}
```

3. Get the result of a statement. The following GET request returns the result of a statement in JSON format, which you can parse to extract elements of the result.

```
GET /sessions/{sessionId}/statements/{statementId}
        statement_url = host + r.headers['location']
        r = requests.get(statement_url, headers=headers)
        pprint.pprint(r.json())

{u'id': 0,
   u'output': {u'data': {u'text/plain': u'res0: Long = 10'},
               u'execution_count': 0,
               u'status': u'ok'},
   u'state': u'available'}
```

The remainder of this section describes Livy objects and REST API calls for interactive sessions.

Livy objects for interactive sessions

See the following tables for Livy objects properties for interactive sessions.

Session Object

A session object represents an interactive shell:

Table 6: Session Object

Property	Description	Type
name	Name of the session	string
id	A non-negative integer that represents a specific session of interest	int
appId	Application ID for this session	string
owner	Remote user who submitted this session	string
proxyUser	User ID to impersonate when running	string
kind	Session kind (see the "kind" table below for values)	session kind
log	Log file data	list of strings
state	Session state (see the "state" table below for values)	string
appInfo	Detailed application information	key=value map

The following values are valid for the kind property in a session object:

Table 7: Session Kind

Value	Description
spark	Interactive Scala Spark session
pyspark	Interactive Python 2 Spark session
pyspark3	Interactive Python 3 Spark session
sparkr	Interactive R Spark session

The following values are valid for the state property in a session object:

Table 8: Session Object State

Value	Description
not_started	Session has not started
starting	Session is starting
idle	Session is waiting for input
busy	Session is executing a statement
shutting_down	Session is shutting down
error	Session terminated due to an error
dead	Session exited
success	Session successfully stopped

A statement object represents the result of an execution statement.

Table 9: Statement Object

Property	Description	Type
id	A non-negative integer that represents a specific statement of interest	integer
state	Execution state (see the following "state" table for values)	statement state
output	Execution output (see the following "output" table for values)	statement output

The following values are valid for the state property in a statement object:

Table 10: Statement Object State

value	Description
waiting	Statement is queued, execution has not started
running	Statement is running
available	Statement has a response ready
error	Statement failed
cancelling	Statement is being cancelled
cancelled	Statement is cancelled

The following values are valid for the output property in a statement object:

Table 11: Statement Object Output

Property	Description	Type
status	Execution status, such as "starting", "idle", or "available".	string
execution_count	Execution count	integer (monotonically increasing)
data	Statement output	An object mapping a mime type to the result. If the mime type is application/json, the value is a JSON value.

Setting Python path variables for Livy

To change the Python executable used by a Livy session, follow the instructions for your version of Python.

pyspark

Livy reads the path from the PYSPARK_PYTHON environment variable (this is the same as PySpark).

- If Livy is running in local mode, simply set the environment variable (this is the same as PySpark).
- If the Livy session is running in yarn-cluster mode, setspark.yarn.appMasterEnv.PYSPARK_PYTHON in the SparkConf file, so that the environment variable is passed to the driver.

pyspark3

Livy reads the path from environment variable PYSPARK3_PYTHON.

- If Livy is running in local mode, simply set the environment variable.
- If the Livy session is running in yarn-cluster mode, setspark.yarn.appMasterEnv.PYSPARK3_PYTHON in Spar kConf file, so that the environment variable is passed to the driver.

Livy API reference for interactive sessions

GET

GET /sessions returns all active interactive sessions.

Request Parameter	Description	Type
from	Starting index for fetching sessions	int
size	Number of sessions to fetch	int

Response	Description	Type
from	Starting index of fetched sessions	int
total	Number of sessions fetched	int
sessions	Session list	list

The following response shows zero active sessions:

```
{"from":0,"total":0,"sessions":[]}
```

GET /sessions/{sessionId} returns information about the specified session.

GET /sessions/{sessionId}/state returns the state of the specified session:

Response	Description	Type
id	A non-negative integer that represents a specific session	int
state	Current state of the session	string

GET /sessions/{sessionId}/logs retrieves log records for the specified session.

Request Parameters	Description	Type
from	Offset	int
size	Maximum number of log records to retrieve	int

Response	Description	Type
id	A non-negative integer that represents a specific session	int
from	Offset from the start of the log file	int
size	Number of log records retrieved	int
log	Log records	list of strings

GET /sessions/{sessionId}/statements returns all the statements in a session.

Response	Description	Type
statements	List of statements in the specified session	list

GET /sessions/{sessionId}/statements/{statementId} returns a specified statement in a session.

Response	Description	Type
statement object (for more information see "Livy Objects for Interactive Sessions")	Statement	statement object

POST

POST /sessions creates a new interactive Scala, Python, or R shell in the cluster.

Request Parameter	Description	Type
kind	Session kind (required)	session kind
proxyUser	User ID to impersonate when starting the session	string
jars	Jar files to be used in this session	list of strings
pyFiles	Python files to be used in this session	list of strings
files	Other files to be used in this session	list of strings
driverMemory	Amount of memory to use for the driver process	string
driverCores	Number of cores to use for the driver process	int
executorMemory	Amount of memory to use for each executor process	string
executorCores	Number of cores to use for each executor process	int
numExecutors	Number of executors to launch for this session	int
archives	Archives to be used in this session	list of strings
queue	The name of the YARN queue to which the job should be submitted	string
name	Name of the session. This is an optional parameter. The session cannot be created if another session with the same name already exists.	string
conf	Spark configuration properties	Map of key=value

Request Parameter	Description	Type
heartbeatTimeoutInSecond	Timeout in second to which session be orphaned	int

Response	Description	Type
session object (for more information see "Livy Objects for Interactive Sessions")	The created session	session object

The following response shows a PySpark session in the process of starting:

```
{"id":0,"state":"starting","kind":"pyspark","log":[]}
```

POST /sessions/{sessionId}/statements runs a statement in a session.

Request Parameter	Description	Type
code	The code to run	string

Response	Description	Type
statement object (for more information see "Livy Objects for Interactive Sessions")	Result of an execution statement	statement object

POST /sessions/{sessionId}/statements/{statementId}/cancel cancels the specified statement in the session.

Response	Description	Type
cancellation message	Reports "cancelled"	string

DELETE

DELETE /sessions/{sessionId} terminates the session.

Submitting batch applications using the Livy REST API

About this task

Spark provides a spark-submit command for submitting batch applications. Livy provides equivalent functionality through REST APIs, using job specifications specified in a JSON document.

The following example shows a spark-submit command that submits a SparkPi job, followed by an example that uses Livy POST requests to submit the job. The remainder of this subsection describes Livy objects and REST API syntax. For additional examples and information, see the readme.rst file at https://github.com/hortonworks/livy-release/releases/tag/HDP-2.6.0.3-8-tag.

The following command uses spark-submit to submit a SparkPi job:

```
./bin/spark-submit \
    --class org.apache.spark.examples.SparkPi \
    --master yarn \
    --deploy-mode cluster \
    --executor-memory 20G \
    /path/to/examples.jar 1000
```

To submit the SparkPi job using Livy, complete the following steps. Note: the POST request does not upload local jars to the cluster. You should upload required jar files to HDFS before running the job. This is the main difference between the Livy API and spark-submit.

Procedure

1. Form a JSON structure with the required job parameters:

```
{ "className": "org.apache.spark.examples.SparkPi",
        "executorMemory": "20g",
        "args": [2000],
        "file": "/path/to/examples.jar"
        }
```

2. Specify master and deploy mode in the livy.conf file.
3. To submit the SparkPi application to the Livy server, use the a POST /batches request.
4. The Livy server helps launch the application in the cluster.

Livy batch object

Batch session APIs operate on batch objects, defined as follows:

Property	Description	Type
id	A non-negative integer that represents a specific batch session	int
appId	The application ID for this session	String
appInfo	Detailed application info	Map of key=value
log	Log records	list of strings
state	Batch state	string

Livy API reference for batch jobs

GET /batches returns all active batch sessions.

Request Parameters	Description	Type
from	Starting index used to fetch sessions	int
size	Number of sessions to fetch	int

Response	Description	Type
from	Starting index of fetched sessions	int
total	Number of sessions fetched	int
sessions	List of active batch sessions	list

GET /batches/{batchId} returns the batch session information as a batch object.

GET /batches/{batchId}/state returns the state of batch session:

Response	Description	Type
id	A non-negative integer that represents a specific batch session	int
state	The current state of batch session	string

GET /batches/{batchId}/log retrieves log records for the specified batch session.

Request Parameters	Description	Type
from	Offset into log file	int
size	Max number of log lines to return	int

Response	Description	Type
id	A non-negative integer that represents a specific batch session	int
from	Offset from start of the log file	int
size	Number of log records returned	int
log	Log records	list of strings

POST /batches creates a new batch environment and runs a specified application:

Request Body	Description	Type
file	File containing the application to run (required)	path
proxyUser	User ID to impersonate when running the job	string
className	Application Java or Spark main class	string
args	Command line arguments for the application	list of strings
jars	Jar files to be used in this session	list of strings
pyFiles	Python files to be used in this session	list of strings
files	Other files to be used in this session	list of strings
driverMemory	Amount of memory to use for the driver process	string
driverCores	Number of cores to use for the driver process	int
executorMemory	Amount of memory to use for each executor process	string
executorCores	Number of cores to use for each executor	int
numExecutors	Number of executors to launch for this session	int
archives	Archives to be used in this session	list of strings
queue	The name of the YARN queue to which the job should be submitted	string
name	Name of this session	string
conf	Spark configuration properties	Map of key=val

Response	Description	Type
batch object (for more information see "Livy Batch Object")	The created batch object	batch object

DELETE /batches/{batchId} terminates the Batch job.

Submitting a Spark job to a cluster using Livy

clusters using definitions such as can accept Spark job execution requests using Livy.

Example

For an example of how to submit a request from a Windows computer to a cluster running on AWS, see How to submit a Spark job to a cluster using Livy.

Configuring the Livy Thrift Server

The Apache Livy Thrift Server provides a JDBC endpoint for submitting Spark SQL queries. The Livy Thrift Server is disabled by default. The following instructions demonstrate how to enable and configure it using .

Before you begin

The Livy Thrift Server is supported in 7.1.0 and higher. The Thrift Server is part of the Livy Server. There is no separate process.

The Livy Thrift Server inherits the security configuration of the Livy Server. It uses Kerberos for authentication and TLS for encryption if configured. There is no separate security configuration for the Thrift server.

Procedure

1. On the home page, click the three-dot menu icon next to the Livy service, and then click Configuration.
2. In the Search field, enter thrift to filter the configuration options and display the Thrift Server configuration settings.
3. Enable the Livy Thrift Server by checking the box labeled Enable Livy Thrift Server.
4. If you want to change the transport mode or port number, modify the Livy Thrift Server Transport Mode and Livy Thrift Server Port settings.
5. Enter a reason for the change in the provided field at the bottom of the page, and then click Save Changes.
6. Return to the home page, and then click the stale configuration icon next to the Livy service.
7. Click the Restart Stale Services button.
8. Make sure that Re-deploy client configuration is checked, and then click Restart Now.
9. After the command completes, click Finish.

What to do next

If you want to further configure the Livy Thrift Server, use the Livy Server Advanced Configuration Snippet (Safety Valve) for livy-conf/livy.conf:

1. On the home page, click the three-dot menu icon next to the Livy service, and then click Configuration.
2. In the Search field, enter livy.conf to filter the configuration options and display the advanced configuration text box.
3. Enter configuration key pairs, one per line, using the syntax <*PARAMETER*> = <*VALUE*>. For example:

   ```
   livy.server.thrift.limit.connections.per.user = 10
   livy.server.thrift.limit.connections.per.ipaddress = 100
   livy.server.thrift.limit.connections.per.user.ipaddress = 10
   ```

   These parameters limit the number of connections to the Thrift server by user, by IP address, and by the combination of both.
4. Enter a reason for the change in the provided field at the bottom of the page, and then click Save Changes.
5. Return to the home page, and then click the stale configuration icon next to the Livy service.
6. Click the Restart Stale Services button.
7. Make sure that Re-deploy client configuration is checked, and then click Restart Now.
8. After the command completes, click Finish.

Connecting to the Apache Livy Thrift Server

You can connect to the Apache Livy Thrift Server using the Beeline client that is included with Apache Hive.

Before you begin

The Livy Thrift Server is disabled by default. Make sure that you have enabled it before proceeding. For instructions, see Configuring the Livy Thrift Server.

The information required to form a JDBC URL is included in the Livy client configuration file, /etc/livy/conf/livy-client.conf. Make sure that you are on a gateway cluster host managed by to ensure that the client configuration is included. You can designate a cluster host as a Livy client by assigning it the Gateway role in the Livy service configuration. For more information about role assignments, see Managing Roles.

Procedure

1. On a host designated as a Livy Gateway, view the /etc/livy/conf/livy-client.conf file and note the values of the following configuration parameters:

   livy.uri

   This is the URI for the Livy server. The hostname in this URI is the hostname to use for the Thrift server connection.

   livy.server.thrift.port

   This is the port to use in the JDBC connection URL.

   livy.server.thrift.transport.mode

   This specifies the transport mode (either binary or http).

2. **Note:** If you are using a cluster with security enabled, you must authenticate with your user principal using kinit before running beeline. If Kerberos authentication is not used, then the “-u <user> -p <password>” options for beeline should be used to connect as a particular user.

   Determine the JDBC connection URL using the following table:

Connection Mode	JDBC Connection URL
binary	jdbc:hive2://<livy_host>:<port>/<database>
binary with Kerberos	jdbc:hive2://<livy_host>:<port>/<database>;principal=livy/<livy_host>@<REALM>
binary with Kerberos and TLS	jdbc:hive2://<livy_host>:<port>/<database>;principal=livy/<livy_host>@<REALM>;ssl=true;sslTrustStore=<path_to_truststore>
http	jdbc:hive2://<livy_host>:<port>/<database>;transportMode=http;httpPath=cliservice
http with Kerberos	jdbc:hive2://<livy_host>:<port>/<database>;principal=HTTP/<livy_host>@<REALM>;transportMode=http;httpPath=cliservice
http with Kerberos and TLS	jdbc:hive2://<livy_host>:<port>/<database>;principal=HTTP/<livy_host>@<REALM>;ssl=true;sslTrustStore=<path_to_truststore>;\ trustStorePassword=<truststore_password>;transportMode=http;httpPath=cliservice

3. Run the beeline command.

4. At the Beeline prompt, enter !connect <jdbc_url>

   You can set configuration options by appending ?<configuration_list> to the JDBC connection URL. Separate multiple options with a semi-colon (;). Supported options are:

   - livy.session.driverMemory
   - livy.session.driverCores
   - livy.session.executorMemory
   - livy.session.executorCores
   - livy.session.numExecutors
   - livy.session.queue
   - livy.session.name
   - livy.session.heartbeatTimeoutInSecond

   For example:

```
beeline> !connect jdbc:hive2://cdp01.example.com:10090/db01?livy.session
.driverMemory=4g;livy.session.driverCores=2
```

Using Livy with Spark

Scala Support

Livy supports Scala version 2.12.

For default Scala builds, Spark 3 with Scala 2.12, Livy automatically detects the correct Scala version and associated jar files.

If you require a different Spark-Scala combination, set livy.spark.scalaVersion to the desired version so that Livy uses the right jar files.

Using PySpark

Apache Spark provides APIs in non-JVM languages such as Python. Many data scientists use Python because it has a rich variety of numerical libraries with a statistical, machine-learning, or optimization focus.

Running PySpark in a virtual environment

For many PySpark applications, it is sufficient to use --py-files to specify dependencies. However, there are times when --py-files is inconvenient, such as the following scenarios:

- A large PySpark application has many dependencies, including transitive dependencies.
- A large application needs a Python package that requires C code to be compiled before installation.
- You want to run different versions of Python for different applications.

For these situations, you can create a virtual environment as an isolated Python runtime environment. supports VirtualEnv for PySpark in both local and distributed environments, easing the transition from a local environment to a distributed environment.

Note:

This feature is currently only supported in YARN mode.

Running Spark Python applications

Accessing Spark with Java and Scala offers many advantages: platform independence by running inside the JVM, self-contained packaging of code and its dependencies into JAR files, and higher performance because Spark itself runs in the JVM. You lose these advantages when using the Spark Python API.

Managing dependencies and making them available for Python jobs on a cluster can be difficult. To determine which dependencies are required on the cluster, you must understand that Spark code applications run in Spark executor processes distributed throughout the cluster. If the Python transformations you define use any third-party libraries, such as NumPy or nltk, Spark executors require access to those libraries when they run on remote executors.

Setting the Python Path

After the Python packages you want to use are in a consistent location on your cluster, set the appropriate environment variables to the path to your Python executables as follows:

- Specify the Python binary to be used by the Spark driver and executors by setting the PYSPARK_PYTHON environment variable in spark-env.sh. You can also override the driver Python binary path individually using the PYSPARK_DRIVER_PYTHON environment variable. These settings apply regardless of whether you are using yarn-client or yarn-cluster mode.

  Make sure to set the variables using the export statement. For example:

  ```
  export PYSPARK_PYTHON=${PYSPARK_PYTHON:-<PATH_TO_PYTHON_EXECUTABLE>}
  ```

  This statement uses shell parameter expansion to set the PYSPARK_PYTHON environment variable to *<PATH_TO_PYTHON_EXECUTABLE>* if it is not set to something else already. If it is already set, it preserves the existing value.

  Here are some example Python binary paths:

  - Anaconda parcel: /opt/cloudera/parcels/Anaconda/bin/python
  - Virtual environment: */PATH/TO/VIRTUALENV*/bin/python
- If you are using yarn-cluster mode, in addition to the above, also set spark.yarn.appMasterEnv.PYSPARK_PYTHON and spark.yarn.appMasterEnv.PYSPARK_DRIVER_PYTHON in spark-defaults.conf (using the safety valve) to the same paths.

In , set environment variables in spark-env.sh and spark-defaults.conf as follows:

Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

1. Go to the Spark service.
2. Click the Configuration tab.
3. Search for Spark Service Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-env.sh.
4. Add the spark-env.sh variables to the property.
5. Search for Spark Client Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-defaults.conf.
6. Add the spark-defaults.conf variables to the property.
7. Enter a Reason for change, and then click Save Changes to commit the changes.
8. Restart the service.
9. Deploy the client configuration.

Self-Contained Dependencies

In a common situation, a custom Python package contains functionality you want to apply to each element of an RDD. You can use a map() function call to make sure that each Spark executor imports the required package, before calling any of the functions inside that package. The following shows a simple example:

```
def import_my_special_package(x):
```

```
  import my.special.package
  return x

int_rdd = sc.parallelize([1, 2, 3, 4])
int_rdd.map(lambda x: import_my_special_package(x))
int_rdd.collect()
```

You create a simple RDD of four elements and call it int_rdd. Then you apply the function import_my_specia l_package to every element of the int_rdd. This function imports my.special.package and then returns the original argument passed to it. Calling this function as part of a map() operation ensures that each Spark executor imports my.special.package when needed.

If you only need a single file inside my.special.package, you can direct Spark to make this available to all executors by using the --py-files option in your spark-submit command and specifying the local path to the file. You can also specify this programmatically by using the sc.addPyFiles() function. If you use functionality from a package that spans multiple files, you can make an egg for the package, because the --py-files flag also accepts a path to an egg file.

If you have a self-contained dependency, you can make the required Python dependency available to your executors in two ways:

- If you depend on only a single file, you can use the --py-files command-line option, or programmatically add the file to the SparkContext with sc.addPyFiles(path) and specify the local path to that Python file.
- If you have a dependency on a self-contained module (a module with no other dependencies), you can create an egg or zip file of that module and use either the --py-files command-line option or programmatically add the module to theSparkContext with sc.addPyFiles(path) and specify the local path to that egg or zip file.

Note: Libraries that are distributed using the Python “wheel” mechanism cannot be used with the --py-files option.

Complex Dependencies

Some operations rely on complex packages that also have many dependencies. Although such a package is too complex to distribute as a *.py file, you can create an egg for it and all of its dependencies, and send the egg file to executors using the --py-files option.

Limitations of Distributing Egg Files on Heterogeneous Clusters

If you are running a heterogeneous cluster, with machines of different CPU architectures, sending egg files is impractical because packages that contain built-in code must be compiled for a single specific CPU architecture. Therefore, distributing an egg for complex, compiled packages like NumPy, SciPy, and pandas often fails. Instead of distributing egg files, install the required Python packages on each host of the cluster and specify the path to the Python binaries for the worker hosts to use.

Installing and Maintaining Python Environments

Installing and maintaining Python environments can be complex but allows you to use the full Python package ecosystem. Ideally, a sysadmin installs the Anaconda distribution or sets up a virtual environment on every host of your cluster with your required dependencies.

If you are using , you can deploy the Anaconda distribution as a parcel as follows:

Minimum Required Role: Cluster Administrator (also provided by Full Administrator)

1. Add the following URL https://repo.anaconda.com/pkgs/misc/parcels/archive/ to the Remote Parcel Repository URLs as described in "Parcel Configuration Settings."
2. Download, distribute, and activate the parcel as described in "Managing Parcels."

Anaconda is installed in *PARCEL DIRECTORY*/Anaconda, where *PARCEL DIRECTORY* is /opt/cloudera/parcels by default, but can be changed in parcel configuration settings. The Anaconda parcel is supported by Continuum Analytics.

If you are not using , you can set up a virtual environment on your cluster by running commands on each host using Cluster SSH, Parallel SSH, or Fabric. Assuming each host has Python and pip installed, use the following commands to set up the standard data stack (NumPy, SciPy, scikit-learn, and pandas) in a virtual environment on a RHEL 6-compatible system:

```
# Install python-devel:
yum install python-devel

# Install non-Python dependencies required by SciPy that are not installed
 by default:
yum install atlas atlas-devel lapack-devel blas-devel

# install virtualenv:
pip install virtualenv

# create a new virtualenv:
virtualenv MYNEWENV

# activate the virtualenv:
source MYNEWENV/bin/activate

# install packages in MYNEWENV:
pip install numpy
pip install scipy
pip install scikit-learn
pip install pandas
```

Related Information

Spark Python Supportability Matrix

Automating Spark Jobs with Oozie Spark Action

You can use Apache Spark as part of a complex workflow with multiple processing steps, triggers, and interdependencies. You can automate Apache Spark jobs using Oozie Spark action.

Before you begin

Spark3 (Gateway) must be installed on the node where the Oozie server is installed.

About Oozie Spark Action

If you use Apache Spark as part of a complex workflow with multiple processing steps, triggers, and interdependencies, consider using Apache Oozie to automate jobs. Oozie is a workflow engine that executes sequences of actions structured as directed acyclic graphs (DAGs). Each action is an individual unit of work, such as a Spark job or Hive query.

The Oozie "Spark action" runs a Spark job as part of an Oozie workflow. The workflow waits until the Spark job completes before continuing to the next action.

For additional information about Spark action, see the latest Oozie Spark3 Action Extension documentation, available on the Oozie UI. For general information about Oozie, see Overview of Oozie.

Important:

Oozie Spark3 actions support both yarn-cluster and yarn-client execution in general. However, there could be use cases which won't work in yarn-client mode.

For Spark3 actions both master and mode tags have to be provided. If master=yarn then mode must be either cluster or client.

Configure Oozie Spark Action for Spark

1. Set up .jar file exclusions.

   Oozie distributes its own libraries on the ShareLib, which are included on the classpath. These .jar files may conflict with each other if some components require different versions of a library. You can use the oozie.action .sharelib.for.<action_type>.exclude=<value> property to address these scenarios.

   Exclusion pattern works well for Spark3 actions, use spark3 as the action type. Jackson should not be an issue for Spark3 actions.

   Examples

   The following examples show how to use a ShareLib exclude on a Java action.

   Actual ShareLib content:

   ```
   * /user/oozie/share/lib/lib_20180701/oozie/lib-one-1.5.jar
   * /user/oozie/share/lib/lib_20180701/oozie/lib-two-1.5.jar
   * /user/oozie/share/lib/lib_20180701/java/lib-one-2.6.jar
   * /user/oozie/share/lib/lib_20180701/java/lib-two-2.6.jar
   * /user/oozie/share/lib/lib_20180701/java/component-connector.jar
   ```

   Setting the oozie.action.sharelib.for.java.exclude property to oozie/lib-one.*= results in the following distributed cache content:

   ```
   * /user/oozie/share/lib/lib_20180701/oozie/lib-two-1.5.jar
   * /user/oozie/share/lib/lib_20180701/java/lib-one-2.6.jar
   * /user/oozie/share/lib/lib_20180701/java/lib-two-2.6.jar
   * /user/oozie/share/lib/lib_20180701/java/component-connector.jar
   ```

   Setting the oozie.action.sharelib.for.java.exclude property to oozie/lib-one.*|component-connector.jar= results in the following distributed cache content:

   ```
   * /user/oozie/share/lib/lib_20180701/oozie/lib-two-1.5.jar
   * /user/oozie/share/lib/lib_20180701/java/lib-one-2.6.jar
   * /user/oozie/share/lib/lib_20180701/java/lib-two-2.6.jar
   ```

2. Run the Oozie shareliblist command to verify the configuration. You should see spark in the results.

   ```
   oozie admin –shareliblist spark
   ```

The following examples show a workflow definition XML file, an Oozie job configuration file, and a Python script for running a Spark3-Pi job.

Sample Workflow.xml file for spark3-Pi:

```
<workflow-app xmlns='uri:oozie:workflow:1.0' name='SparkPythonPi'>
          <start to='spark-node' />

          <action name='spark-node'>
            <spark3 xmlns="uri:oozie:spark3-action:1.0">
              <resource-manager>${resourceManager}</resource-manager>
              <name-node>${nameNode}</name-node>
              <master>${master}</master>
              <mode>${mode}</mode>
              <name>Python-Spark-Pi</name>
              <jar>pi.py</jar>
            </spark3>
            <ok to="end" />
            <error to="fail" />
          </action>
```

```
          <kill name="fail">
            <message>Workflow failed, error message [${wf:errorMessage(wf
:lastErrorNode())}]</message>
          </kill>
          <end name='end' />
        </workflow-app>
```

Sample Job.properties file for spark3-Pi:

```
nameNode=hdfs://host:8020
resourceManager=host:8050
queueName=default
examplesRoot=examples
oozie.use.system.libpath=true
oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/ap
ps/pyspark
master=yarn
mode=cluster
```

Sample Python script, lib/pi.py:

```
import sys
from random import random
from operator import add
from pyspark import SparkContext

if __name__ == "__main__":
"""
Usage: pi [partitions]
"""
sc = SparkContext(appName="Python-Spark-Pi")
partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2
n = 100000 * partitions

def f(_):
x = random() * 2 - 1
y = random() * 2 - 1
return 1 if x ** 2 + y ** 2 < 1 else 0

count = sc.parallelize(range(1, n + 1), partitions).map(f).reduce(add)
print("Pi is roughly %f" % (4.0 * count / n))

sc.stop()
```

Troubleshooting .jar file conflicts with Oozie Spark action

When using Oozie Spark action, Oozie jobs may fail with an error similar to below if there are .jar file conflicts between the "oozie" ShareLib and the "spark3" ShareLib. In such case, you need to remove the conflicting JAR file.

```
2018-06-04 13:27:32,652 WARN SparkActionExecutor:523 - SERVER[XXXX] USER[XXX
X] GROUP[-] TOKEN[] APP[XXXX] JOB[0000000-<XXXXX>-oozie-oozi-W] ACTION[00000
00-<XXXXXX>-oozie-oozi-W@spark2] Launcher exception: Attempt to add (hdfs://
XXXX/user/oozie/share/lib/lib_XXXXX/oozie/aws-java-sdk-kms-1.10.6.jar) multi
ple times to the distributed cache.
java.lang.IllegalArgumentException: Attempt to add (hdfs://XXXXX/user/oozie/
share/lib/lib_20170727191559/oozie/aws-java-sdk-kms-1.10.6.jar) multiple ti
mes to the distributed cache.
at org.apache.spark.deploy.yarn.Client$anonfun$prepareLocalResources$13$a
nonfun$apply$8.apply(Client.scala:632)
at org.apache.spark.deploy.yarn.Client$anonfun$prepareLocalResources$13$anon
fun$apply$8.apply(Client.scala:623)
at scala.collection.mutable.ArraySeq.foreach(ArraySeq.scala:74)
```

```
at org.apache.spark.deploy.yarn.Client$anonfun$prepareLocalResources$13.ap
ply(Client.scala:623)
at org.apache.spark.deploy.yarn.Client$anonfun$prepareLocalResources$13.a
pply(Client.scala:622)
at scala.collection.immutable.List.foreach(List.scala:381)
at org.apache.spark.deploy.yarn.Client.prepareLocalResources(Client.scala:62
2)
at org.apache.spark.deploy.yarn.Client.createContainerLaunchContext(Client.s
cala:895)
at org.apache.spark.deploy.yarn.Client.submitApplication(Client.scala:171)
at org.apache.spark.deploy.yarn.Client.run(Client.scala:1231)
at org.apache.spark.deploy.yarn.Client$.main(Client.scala:1290)
at org.apache.spark.deploy.yarn.Client.main(Client.scala)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.ja
va:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorI
mpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit
$runMain(SparkSubmit.scala:750)
at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:187)
at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:212)
at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:126)
at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
at org.apache.oozie.action.hadoop.SparkMain.runSpark(SparkMain.java:311)
at org.apache.oozie.action.hadoop.SparkMain.run(SparkMain.java:232)
at org.apache.oozie.action.hadoop.LauncherMain.run(LauncherMain.java:58)
at org.apache.oozie.action.hadoop.SparkMain.main(SparkMain.java:62)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.jav
a:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.apache.oozie.action.hadoop.LauncherMapper.map(LauncherMapper.java:2
37)
at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:54)
at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:453)
at org.apache.hadoop.mapred.MapTask.run(MapTask.java:343)
at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:170)
at java.security.AccessController.doPrivileged(Native Method)
at javax.security.auth.Subject.doAs(Subject.java:422)
at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformatio
n.java:1866)
at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:164)
```

Run the following commands to resolve this issue.

Note:

You may need to perform a backup before running the rm commands.

```
hadoop fs -rm /user/oozie/share/lib/lib_<ts>/spark3/aws*
hadoop fs -rm /user/oozie/share/lib/lib_<ts>/spark3/azure*
hadoop fs -rm /user/oozie/share/lib/lib_<ts>/spark3/hadoop-aws*
hadoop fs -rm /user/oozie/share/lib/lib_<ts>/spark3/hadoop-azure*
hadoop fs -rm /user/oozie/share/lib/lib_<ts>/spark3/ok*
hadoop fs -mv /user/oozie/share/lib/lib_<ts>/oozie/jackson* /user/oozie/sh
are/lib/lib_<ts>/oozie.old
```

Next, run the following command to update the Oozie ShareLib:

```
oozie admin -oozie http://<oozie-server-hostname>:11000/oozie -sharelibu
pdate
```

Cloudera Runtime 7.3.1

Managing Data Storage

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Optimizing data storage....6

Balancing data across disks of a DataNode....6
- Plan the data movement across disks....6
- Run the Disk Balancer plan....7
- Disk Balancer commands....8

Erasure coding overview....10
- Understanding erasure coding policies....11
- Comparing replication and erasure coding....12
- Best practices for rack and node setup for EC....14
- Prerequisites for enabling erasure coding....15
- Limitations of erasure coding....15
- Using erasure coding for existing data....15
- Using erasure coding for new data....15
- Advanced erasure coding configuration....16
- Erasure coding CLI command....17
- Erasure coding examples....18

Increasing storage capacity with HDFS compression....20
- Enable GZipCodec as the default compression codec....20
- Use GZipCodec with a one-time job....21

Set HDFS quotas....21
- Setting HDFS quotas in Cloudera Manager....22

Configuring heterogeneous storage in HDFS....22
- HDFS storage types....22
- HDFS storage policies....22
- Commands for configuring storage policies....23
- Set up a storage policy for HDFS....23
- Set up SSD storage using Cloudera Manager....24
- Configure archival storage....24
- The HDFS mover command....25

Balancing data across an HDFS cluster....26
- Why HDFS data becomes unbalanced....26
- Configurations and CLI options for the HDFS Balancer....26
- Configuring and running the HDFS balancer using Cloudera Manager....30
- Cluster balancing algorithm....32
- Exit statuses for the HDFS Balancer....34

HDFS heap sizing....34

Optimizing performance....35

Improving performance with centralized cache management....35
- Benefits of centralized cache management in HDFS....35
- Use cases for centralized cache management....36
- Centralized cache management architecture....36
- Caching terminology....37
- Properties for configuring centralized caching....37
- Commands for using cache pools and directives....39

Specifying racks for hosts....42
- Viewing racks assigned to cluster hosts....43
- Editing rack assignments for hosts....43

Customizing HDFS....43
Customize the HDFS home directory....43
Properties to set the size of the NameNode edits directory....44
Optimizing NameNode disk space with Hadoop archives....44
Overview of Hadoop archives....44
Hadoop archive components....44
Creating a Hadoop archive....45
List files in Hadoop archives....46
Format for using Hadoop archives with MapReduce....47
Detecting slow DataNodes....47
Enable disk IO statistics....47
Enable detection of slow DataNodes....48
Allocating DataNode memory as storage....48
HDFS storage types....49
LAZY_PERSIST memory storage policy....49
Configure DataNode memory as storage....49
Improving performance with short-circuit local reads....50
Prerequisites for configuring short-ciruit local reads....50
Properties for configuring short-circuit local reads on HDFS....50
Configure mountable HDFS....52
Add HDFS system mount....53
Optimize mountable HDFS....54

Configuring Proxy Users to Access HDFS....54

Using DistCp to copy files....55
Using DistCp....56
Distcp syntax and examples....56
Using DistCp with Highly Available remote clusters....57
Using DistCp with Amazon S3....60
Using a credential provider to secure S3 credentials....60
Examples of DistCp commands using the S3 protocol and hidden credentials....61
Kerberos setup guidelines for Distcp between secure clusters....62
Distcp between secure clusters in different Kerberos realms....63
Configure source and destination realms in krb5.conf....63
Configure HDFS RPC protection....64
Specify truststore properties....64
Set HADOOP_CONF to the destination cluster....65
Launch distcp....65
Copying data between a secure and an insecure cluster using DistCp and WebHDFS....65
Post-migration verification....66
Using DistCp between HA clusters using Cloudera Manager....67

Using the NFS Gateway for accessing HDFS....68
Install the NFS Gateway....68
Configure the NFS Gateway....69
Start and stop the NFS Gateway services....70
Start the NFS Gateway services....71
Stop the NFS Gateway services....71
Verify validity of the NFS services....71
Access HDFS from the NFS Gateway....72
How NFS Gateway authenticates and maps users....72

APIs for accessing HDFS.....73
Set up WebHDFS on a secure cluster.....73

Using HttpFS to provide access to HDFS.....74
Add the HttpFS role.....74
Using Load Balancer with HttpFS.....75

HttpFS authentication.....75
Use curl to access a URL protected by Kerberos HTTP SPNEGO.....75

Data storage metrics.....76
Using JMX for accessing HDFS metrics.....76

HDFS Metrics.....77

Using HdfsFindTool to find files.....78
Downloading Hdfsfindtool from the CDH archives.....78

Optimizing data storage

You can consider various options to optimize data storage, such as increasing storage space through erasure coding, using codecs for compressing data, balancing data across an HDFS cluster, and so on.

To know more about HDFS heap calculations, see HDFS heap sizing.

Balancing data across disks of a DataNode

The HDFS Disk Balancer is a command-line tool that evenly distributes data across all the disks of a DataNode to ensure that the disks are effectively utilized. Unlike the HDFS Balancer that balances data across DataNodes of a cluster, the Disk Balancer balances data at the level of individual DataNodes.

Disks on a DataNode can have an uneven spread of data because of reasons such as large amount of writes and deletes or disk replacement. To ensure uniformity in the distribution of data across disks of a specified DataNode, the Disk Balancer operates against the DataNode and moves data blocks from one disk to another.

For a specified DataNode, the Disk Balancer determines the ideal amount of data to store per disk based on its total capacity and used space, and computes the amount of data to redistribute between disks. The Disk Balancer captures this information about data movement across the disks of a DataNode in a plan. On executing the plan, the Disk Balancer moves the data as specified.

Plan the data movement across disks

A Disk Balancer plan identifies the amount of data that should move between the disks of a specified DataNode. The plan contains move steps, where each move step specifies the source and destination disks for data movement and the number of bytes to move.

Before you begin

You must set the dfs.disk.balancer.enabled property in hdfs-site.xml to *TRUE*.

Procedure

Run the hdfs diskbalancer -plan command by specifying the path to the DataNode.

For example, hdfs diskbalancer -plan node1.mycluster.com.

The specified command generates the following two JSON documents as output in the HDFS namespace: <*NODENAME*>.before.json that indicates the state of the cluster before running Disk Balancer, and <*NODENAME*>.plan.json that details the data movement plan for the DataNode.

By default, the JSON documents are placed in the following location: /system/diskbalancer/<Creation_Timestamp>. Here, <Creation_Timestamp> indicates the folder named according to the date and time of creation of the JSON documents.

What to do next

Run the generated plan.

Related Information

Run the Disk Balancer plan

Parameters to configure the Disk Balancer

You must configure various parameters through HDFS Service Advanced Configuration Snippet (Safety Valve) in hdfs-site.xml for effectively planning and executing the Disk Balancer.

Parameter	Description
dfs.disk.balancer.enabled	Controls whether the Disk Balancer is enabled for the cluster. The Disk Balancer executes only if this parameter is set to *TRUE*. The default value is *FALSE*.
dfs.disk.balancer.max.disk.throughputInMBperSec	The maximum disk bandwidth that Disk Balancer consumes while transferring data between disks. The default value is 10 MB/s.
dfs.disk.balancer.max.disk.errors	The maximum number of errors to ignore for a move operation between two disks before abandoning the move. The default value of the maximum errors to ignore is 5. For example, if a plan specifies data move operation between three pairs of disks, and if the move between the first pair encounters more than five errors, that move is abandoned and the next move between the second pair of disks starts.
dfs.disk.balancer.block.tolerance.percent	Specifies a threshold value in percentage to consider a move operation successful, and stop moving further data. For example, setting this value to 20% for an operation requiring 10 GB data movement indicates that the movement will be considered successful only after 8 GB of data is moved.
dfs.disk.balancer.plan.threshold.percent	The ideal storage value for a set of disks in a DataNode indicates the amount of data each disk should have for achieving perfect data distribution across those disks. The threshold percentage defines the value at which disks start participating in data redistribution or balancing operations. Minor imbalances are ignored because normal operations automatically correct some of these imbalances. The default threshold percentage for a disk is 10%; indicating that a disk is used in balancing operations only if the disk contains 10% more or less data than the ideal storage value.

Run the Disk Balancer plan

For a DataNode, the HDFS Disk Balancer moves data according to the details specified in the JSON document generated as the Disk Balancer plan.

Before you begin

You must have generated the data movement plan for the DataNode.

Procedure

Run the hdfs diskbalancer -execute command by specifying the path to the JSON plan document.

```
hdfs diskbalancer -execute /system/diskbalancer/<Creation_Timestamp>/nodename.plan.json
```

The command reads the DataNode's address from the JSON file and executes the plan.

Related Information

Plan the data movement across disks

Disk Balancer commands

In addition to planning for data movement across disks and executing the plan, you can use hdfs diskbalancer sub-commands to query the status of the plan, cancel the plan, identify at a cluster level the DataNodes that require balancing, or generate a detailed report on a specific DataNode that can benefit from running the Disk Balancer.

Planning the data movement for a DataNode

Command:hdfs diskbalancer -plan <datanode>

Argument	Description
<DATANODE>	Fully qualified name of the DataNode for which you want to generate the plan.

hdfs diskbalancer -plan node1.mycluster.com

The following table lists the additional options that you can use with the hdfs diskbalancer -plan command.

Option	Description
-out	Specify the location within the HDFS namespace where you want to save the output JSON documents that contain the generated plans.
-bandwidth	Specify the maximum bandwidth to use for running the Disk Balancer. This option helps in minimizing the amount of data moved by the Disk Balancer on an operational DataNode. Disk Balancer uses the default bandwidth of 10 MB/s if you do not specify this value.
-thresholdPercentage	The ideal storage value for a set of disks in a DataNode indicates the amount of data each disk should have for achieving perfect data distribution across those disks. The threshold percentage defines the value at which disks start participating in data redistribution or balancing operations. Minor imbalances are ignored because normal operations automatically correct some of these imbalances. The default value of -thresholdPercentage for a disk is 10%; indicating that a disk is used in balancing operations only if the disk contains 10% more or less data than the ideal storage value.
-maxerror	Specify the number of errors to ignore for a move operation between two disks before abandoning the move. Disk Balancer uses the default if you do not specify this value.
-v	Verbose mode. Specify this option for Disk Balancer to display a summary of the plan as output.
-fs	Specify the NameNode to use.

Option	Description
	Disk Balancer uses the default NameNode from the configuration if you do not specify this value.

Executing the plan

Command:hdfs diskbalancer -execute <JSON file path>

Argument	Description
<JSON FILE PATH>	Path to the JSON document that contains the generated plan (nodename.plan.json).

hdfs diskbalancer -execute /system/diskbalancer/nodename.plan.json

Querying the current status of execution

Command:hdfs diskbalancer -query <datanode>

Argument	Description
<DATANODE>	Fully qualified name of the DataNode for which the plan is running.

hdfs diskbalancer -query nodename.mycluster.com

Cancelling a running plan

Commands:hdfs diskbalancer -cancel <JSON file path>

Argument	Description
<JSON FILE PATH>	Path to the JSON document that contains the generated plan (nodename.plan.json).

hdfs diskbalancer -cancel /system/diskbalancer/nodename.plan.json

OR

hdfs diskbalancer -cancel <planID> -node <nodename>

Argument	Description
PLANID	ID of the plan to cancel. You can get this value from the output of the `hdfs diskbalancer -query` command.
NODENAME	The fully qualified name of the DataNode on which the plan is running.

Viewing detailed report of DataNodes that require Disk Balancer

Commands:hdfs diskbalancer -fs http://namenode.uri -report -node <file://>

Argument	Description
<FILE://>	Hosts file listing the DataNodes for which you want to generate the reports.

OR

hdfs diskbalancer -fs http://namenode.uri -report -node [<DataNodeID|IP|Hostname>,...]

Argument	Description
[<DATANODEID|IP|HOSTNAME>,...]	Specify the DataNode ID, IP address, and the host name of the DataNode for which you want to generate the report. For multiple DataNodes, provide the details using comma-separated values.

Viewing details of the top DataNodes in a cluster that require Disk Balancer

Command:hdfs diskbalancer -fs http://namenode.uri -report-node -top <topnum>

Argument	Description
<TOPNUM>	The number of the top DataNodes that require Disk Balancer to be run.

Erasure coding overview

Data durability describes how resilient data is to loss. When data is stored in HDFS, Cloudera provides two options for data durability. You can use replication, which HDFS was originally built on, or Erasure Coding (EC).

Note: The comparisons between EC and replication use a replication factor of 3 (three copies of data are maintained) since that is the default.

Replication

HDFS creates two copies of data, resulting in three total instances of data. These copies are stored on separate DataNodes to guard against data loss when a node is unreachable. When the data stored on a node is lost or inaccessible, it is replicated from one of the other nodes to a new node so that there are always multiple copies. The number of replications is configurable, but the default is three. Cloudera recommends keeping the replication factor to at least three when you have three or more DataNodes. A lower replication factor leads to a situation where the data is more vulnerable to DataNode failures since there are fewer copies of data spread out across fewer DataNodes..

When data is written to an HDFS cluster that uses replication, additional copies of the data are automatically created. No additional steps are required.

Replication supports all data processing engines that Cloudera supports.

Erasure Coding (EC)

EC is an alternative to replication. When an HDFS cluster uses EC, no additional direct copies of the data are generated. Instead, data is striped into blocks and encoded to generate parity blocks. If there are any missing or corrupt blocks, HDFS uses the remaining data and parity blocks to reconstruct the missing pieces in the background. This process provides a similar level of data durability to 3x replication but at a lower storage cost.

Additionally, EC is applied when data is written. This means that to use EC, you must first create a directory and configure it for EC. Then, you can either replicate existing data or write new data into this directory.

EC supports the following data processing engines:

- Hive
- MapReduce
- Spark

With both data durability schemes, replication and EC, recovery happens in the background and requires no direct input from a user.

HDFS clusters can be configured with a single data durability scheme (3x replication or EC), or with a hybrid data durability scheme where EC enabled directories co-exist on a cluster with other directories that are protected with the traditional 3x replication model. This decision should be based on the temperature of the data (how often the data is accessed) stored in HDFS. Hot data, data that is accessed frequently, should use replication. Cold data, data that is accessed less frequently, can take advantage of EC's storage savings.

Understanding erasure coding policies

The EC policy determines how data is encoded and decoded. An EC policy is made up of the following parts: codec-number of data blocks-number of parity blocks-cell size.

- Codec: The erasure codec that the policy uses. Cloudera currently supports Reed-Solomon (RS).
- Number of Data Blocks: The number of data blocks per stripe. The higher this number, the more nodes that need to be read when accessing data because HDFS attempts to distribute the blocks evenly across DataNodes.
- Number of Parity Blocks: The number of parity blocks per stripe. Even if a file does not use up all the data blocks available to it, the number of parity blocks will always be the total number listed in the policy.
- Cell Size: The size of one basic unit of striped data.

For example, a RS-6-3-1024k policy has the following attributes:

- Codec: Reed-Solomon
- Number of Data Blocks: 6
- Number of Parity Blocks: 3
- Cell Size: 1024k

The sum of the number of data blocks and parity blocks is the data-stripe width. When you make hardware plans for your cluster, the number of racks should at least equal the stripe width in order for the data to be resistant to rack failures.

The following image compares the data durability and storage efficiency of different RS codecs and replication:

	Data Durability	Storage Efficiency
Single copy of data	0	100%
3x replication	2	33%
RS(6,3)	3	67%
RS(3,2)	2	60%

Storage efficiency is the ratio of data blocks to total blocks as represented by the following formula: data blocks / (data blocks + parity blocks)

Comparing replication and erasure coding

You must consider factors such as data temperature, i/o cost, storage cost, and file size when comparing replication and erasure coding.

Data Temperature

Data temperature refers to how often data is accessed. EC works best with cold data that is accessed and modified infrequently. There is no data locality, and all reads are remote. Replication is more suitable for hot data, data that is accessed and modified frequently because data locality is a part of replication.

I/O Cost

EC has higher I/O costs than replication for the following reasons:

- EC spreads data across nodes and racks, which means reading and writing data comes at a higher network cost.
- A parity block is generated when data is written, thus impacting write speed. This can be slower than writing to a file when the replication factor is one but is faster than writing two or more replicas.
- If data is missing or corrupt, a DataNode reads the remaining data and parity blocks in order to reconstruct the data. This process requires CPU and network resources.

Cloudera recommends at least a 10 GB network connection if you want to use EC.

Storage Cost

EC has a lower storage overhead than replication because multiple copies of data are not maintained. Instead, a number of parity blocks are generated based on the EC policy. For the same

amount of data, EC will store fewer blocks than 3x replication in most cases. For example with a Reed-Solomon (6,3), HDFS stores three parity blocks for each set of 6 data blocks. With replication, HDFS stores 12 replica blocks for every six data blocks, the original block and three replicas. The case where 3x replication requires fewer blocks is when data is stored in small files.

File Size

Erasure coding works best with larger files. The total number of blocks is determined by data blocks + parity blocks, which is the data-stripe width discussed earlier.

128 MB is the default block size. With RS (6,3), each block group can hold up to (128 MB * 6) = 768 MB of data. Inside each block group, there will be 9 total blocks, 6 data blocks, each holding up to 128 MB, and 3 parity blocks. This is why EC works best with larger files. For a chunk of data less than the block size, replication writes one data block to three DataNodes; EC, on the other hand, still needs to stripe the data to data blocks and calculate parity blocks. This leads to a situation where erasure coded files will generate more blocks than replication because of the parity blocks required. The bytes/blocks ratio is worse for small files which increases the memory usage of the NameNode.

The following two figures show how replication (with a replication factor of 3) compares to EC based on the number of blocks relative to file size.

For very small files of sizes smaller than the value of data blocks * cell size (in case of RS(6, 3) it is (6 * 1) MB), the number of actual data blocks is less than the data blocks defined by the erasure coding policy, though the number of parity blocks is always the same. For example, in the case of RS(6,3) with a cell size of 1 MB, a 1MB file consists of one data block, rather than six, but it still has three parity blocks. A 1MB file would therefore require four block objects in total. If 3x replication were used, the same file would require only three block objects.

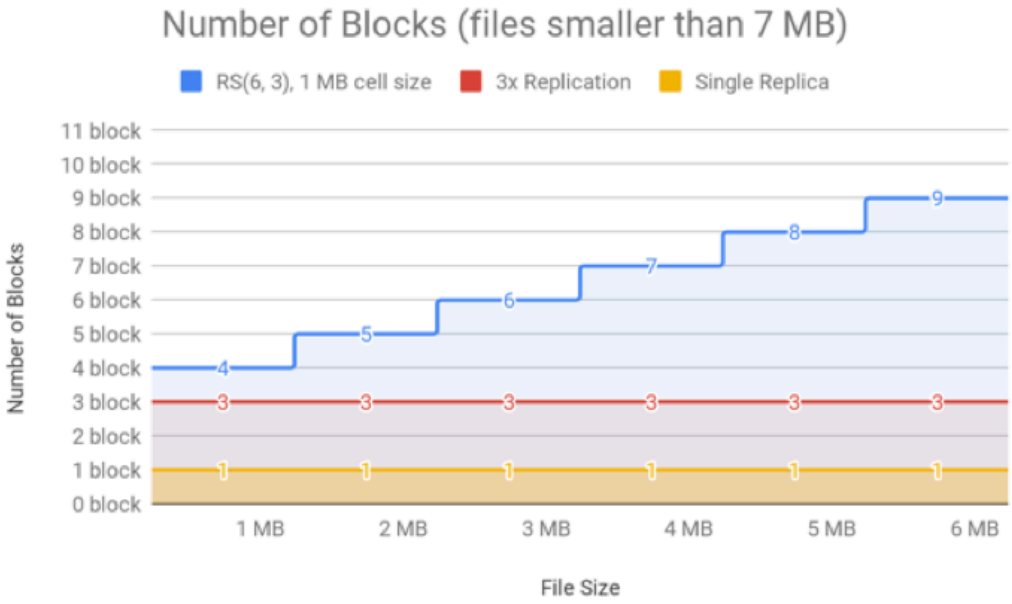

As shown in the following figure, the RS(6,3) EC policy ensures that the number of data blocks continues to remain less than that used by 3x replication for larger file sizes.

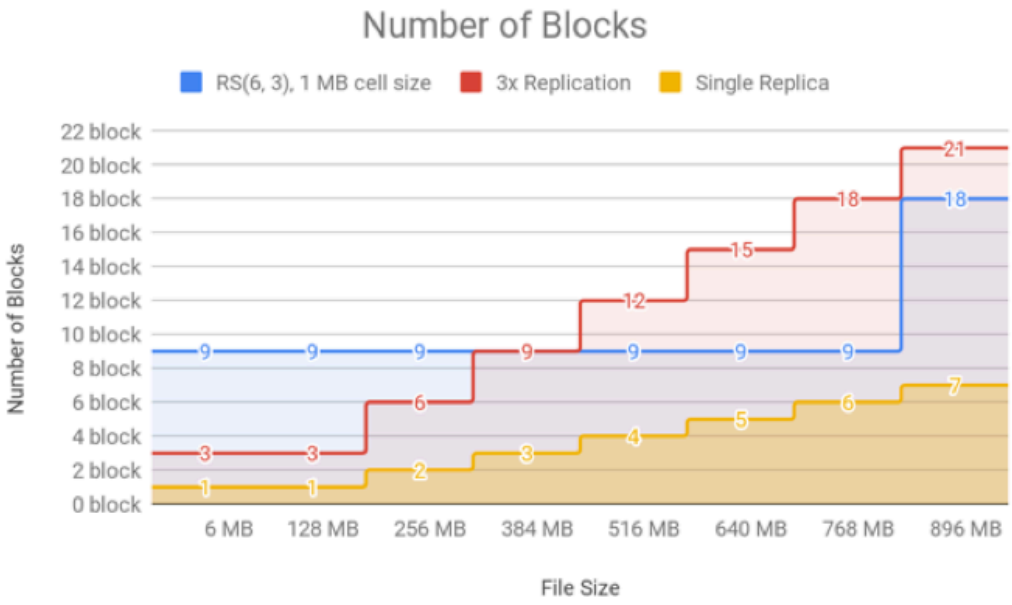

Supported Engines

Replication supports all data processing engines that Cloudera supports.

EC supports the following data processing engines: Hive, MapReduce, and Spark.

Unsupported Features

The XOR codec for EC is not supported. Additionally, certain HDFS functions are not supported with EC: hflush, hsync, concat, setReplication, truncate and append. For more information, see Erasure Coding Limitations. and HDFS Erasure Coding Limitations.

Best practices for rack and node setup for EC

In a Cloudera Private Cloud Base deployment, when setting up a cluster to take advantage of EC, consider the number of racks and nodes in your setup.

Ideally, the number of racks exceed the data-stripe width to account for downtime and outages. If there are fewer racks than the data-stripe width, HDFS spreads data blocks across multiple nodes to maintain fault tolerance at the node level. When distributing blocks to racks, HDFS attempts to distribute the blocks evenly across all racks. Because of this behavior, Cloudera recommends setting up each rack with a similar number of DataNodes. Otherwise, racks with fewer DataNodes may be filled up faster than racks with more DataNodes.

To achieve node-level fault tolerance, the number of nodes needs to equal the data-stripe width. For example, in order for a RS-6-3-1024k policy to be node failure tolerant, you need at least nine nodes. Note that the write will fail if the number of DataNodes is less than the policy's number of data blocks. The write will succeed but show a warning message if the number of DataNodes is less than the policy's number of data blocks + parity blocks. For example, with RS(6,3), if there are six to eight DataNodes, the write will succeed but show a warning message. If there are less than six DataNodes, the write fails.

For rack-level fault tolerance, there must be enough number of racks such that each rack contains at most the same number of blocks as the erasure coding parity blocks. The number of racks is calculated as (data stripe width) / (number of parity blocks). For example, with RS(6,3), the minimum number of racks must be (6+3)/3 = 3. Cloudera recommends at least nine racks, which leads to one data or parity block on each rack. Ideally, the number of racks exceeds the sum of the number of data and parity blocks.

If you decide to use three racks, spread the nine nodes evenly across three racks. The data and parity blocks, when distributed evenly, lead to the following placement on the racks:

- Rack 1 with three nodes: Three blocks

- Rack 2 with three nodes: Three blocks
- Rack 3 with three nodes: Three blocks

A policy with a wide data-stripe width like RS-6-3-1024k comes with a tradeoff though. Data must be read from six blocks, increasing the overall network load. Choose your EC policy based on your network settings and expected storage efficiency. Note, the larger the cluster and colder the data, the more appropriate it is to use EC policies with large data-stripe widths. Larger data-stripe widths have the benefit of a better storage efficiency.

Prerequisites for enabling erasure coding

Before enabling erasure coding on your data, you must consider various factors such as the type of policy to use, the type of data, and the rack or node requirements.

- Note the limitations for EC.
- Determine which EC policy you want to use:
- Determine if you want to use EC for existing data or new data. If you want to use EC for existing data, you need to replicate that data with distcp or BDR.
- Verify that your cluster setup meets the rack and node requirements.

Limitations of erasure coding

The limitations of erasure coding include non-support of XOR codecs and certain HDFS functions.

EC does not support:

- XOR codecs
- certain HDFS functions such as hflush, hsync, concat, setReplication, and truncate. For more information, see Erasure Coding Limitations.

Using erasure coding for existing data

You must set a supported EC policy for a directory and copy the existing data to the directory.

Procedure

1. Create a new directory or choose an existing directory.
2. View the supported EC policies.

```
hdfs ec -listPolicies
```

3. Enable a supported EC policy.

```
hdfs ec -enablePolicy -policy <POLICY>
```

4. Set the EC policy for the directory you want to use.

```
hdfs ec -setPolicy -path <DIRECTORY> [-policy <POLICYNAME>]
```

   - path. Required. Specify the HDFS directory you want to apply the EC policy to.
   - policy. Optional. The EC policy you want to use for the directory you specified. If you do not provide this parameter, the EC policy you specified in the Default Policy when Setting Erasure Coding setting from Cloudera Manager is used.
5. Copy the data to the directory you set an EC policy for.

   You can use the distcp tool or Cloudera Manager's Backup and Disaster Recovery process.

Using erasure coding for new data

You must create a new directory and then set a supported EC policy for the directory.

Procedure

1. Create a new directory or choose an existing directory.
2. View the supported EC policies.

```
hdfs ec -listPolicies
```

3. Enable a supported EC policy.

```
hdfs ec -enablePolicy -policy <POLICY>
```

4. Set the EC policy for the directory you want to use.

```
hdfs ec -setPolicy -path <DIRECTORY> [-policy <POLICYNAME>]
```

   - path. Required. Specify the HDFS directory you want to apply the EC policy to.
   - policy. Optional. The EC policy you want to use for the directory you specified. If you do not provide this parameter, the EC policy you specified in the Fallback Erasure Coding Policy setting from Cloudera Manager is used.
5. Set the destination for the data to the directory you enabled EC for. No action beyond that is required.

   When data is written to the directory, it will be erasure coded based on the policy you set.

Advanced erasure coding configuration

You can customize the behavior of EC through a combination of the hdfs ec subcommand and the Cloudera Admin Console.

About this task

This procedure provides information about configuring certain properties for EC with the Cloudera Admin Console.

For information about the hdfs ec see Erasure coding CLI command on page 17.

Procedure

1. Select Clusters and choose the HDFS cluster you want to configure.
2. Navigate to the Configuration tab and select the Erasure Coding category.
3. Configure the advanced EC properties.
   - DataNode Striped Read Timeout: The timeout for striped reads during background data reconstruction in milliseconds.
   - DataNode Striped Read Threads: The number of threads that a DataNode can use during background data reconstruction.
   - Erasure Coding Reconstruction Weight: The relative weight of resources used by EC for data recovery. The number of blocks that must be read is based on the EC policy used. For example, RS-6-3-1024k requires six blocks to be read. Replication only requires one block to be read. Higher values result in fewer reconstruction tasks being able to run concurrently. The number of blocks required to be read to recover data is multiplied by this weight to determine the total weight of the recovery task. The total weight of the recovery task counts against the limit set with the dfs.namenode.replication.max-streams property.
   - Fallback Erasure Coding Policy: The fallback Erasure Coding policy that HDFS uses if no policy is specified when you run the -setPolicy command.

     Note: You must disable erasure coding in small clusters to prevent potential data loss. If the cluster falls back to an Erasure Coding policy value that requires a greater number of DataNodes or racks than available, this situation could result in a potential data loss. As a result, the Erasure Coding Policy Verification Test returns "Concerning" health. To prevent the issue, set the value of the fallback Erasure Coding policy to No Default Erasure Coding Policy. In addition, you must disable all the erasure coding policies that are enabled on the cluster.

Erasure coding CLI command

Use the hdfs ec command to set erasure coding policies on directories.

```
hdfs ec [generic options]
     [-setPolicy -path <path> [-policy <policyName>] [-replicate]]
     [-getPolicy -path <path>]
     [-unsetPolicy -path <path>]
     [-listPolicies]
     [-addPolicies -policyFile <file>]
     [-listCodecs]
     [-removePolicy -policy <policyName>]
     [-enablePolicy -policy <policyName>]
     [-disablePolicy -policy <policyName>]
     [-help [cmd ...]]
```

Options:

- [-setPolicy [-p <policyName>] <path>]: Sets an EC policy on a directory at the specified path. The following EC policies are supported: RS-3-2-1024k (Reed-Solomon with 3 data blocks, 2 parity blocks and 1024 KB cell size), RS-6-3-1024k, RS-LEGACY-6-3-1024k, and XOR-2-1-1024k.

  <path>: A directory in HDFS. This is a mandatory parameter. Setting a policy only affects newly created files, and does not affect existing files.

  <policyName>: The EC policy to be used for files under the specified directory. This is an optional parameter, specified using the -p flag. If no policy is specified, the system default Erasure Coding policy is used. The default policy is RS-6-3-1024k.
- -replicate: Forces a directory to use the default 3x replication scheme.

  Note: You cannot specify -replicate and -policy <policyName> at the same time. Both the arguments are optional.
- -getPolicy -path <path>: Gets details of the EC policy of a file or directory for the specified path.
- [-unsetPolicy -path <path>]: Removes an EC policy already set by a setPolicy on a directory. This option does not work on a directory that inherits the EC policy from a parent directory. If you run this option on a directory that does not have an explicit policy set, no error is returned.
- [-addPolicies -policyFile <file>]: Adds a list of EC policies. HDFS allows you to add 64 policies in total, with the policy ID in the range of 64 to 127. Adding policies fails if there are already 64 policies.
- [-listCodecs]: Lists all the supported EC erasure coding codecs and coders in the system.
- [-removePolicy -policy <policyName>]: Removes an EC policy.
- [-enablePolicy -policy <policyName>]: Enables an EC policy.
- [-disablePolicy -policy <policyName>]: Disables an EC policy.
- [-help]: Displays help for a given command, or for all commands if none is specified.

Erasure coding background recovery work on the DataNodes can be tuned using the following configuration parameters in hdfs-site.xml.

- dfs.datanode.ec.reconstruction.stripedread.timeout.millis: Timeout for striped reads. Default value is 5000 ms.
- dfs.datanode.ec.reconstruction.threads: Number of threads used by the DataNode for the background recovery task. The default value is 8 threads.
- dfs.datanode.ec.reconstruction.stripedread.buffer.size: Buffer size for reader service. Default value is 64 KB.
- dfs.datanode.ec.reconstruction.xmits.weight: The relative weight of xmits used by the EC background recovery task when compared to replicated block recovery. The default value is 0.5.

  If the parameter is set to 0, the EC task always has one xmit. The xmits of an erasure coding recovery task are calculated as the maximum value between the number of read streams and the number of write streams. For example, if an EC recovery task needs to read from six nodes and write to two nodes, the xmit value is max(6, 2) * 0.5 = 3.

Erasure coding examples

You can use the hdfs ec command with its various options to set erasure coding policies on directories.

Viewing the list of erasure coding policies

The following example shows how you can view the list of available erasure coding policies:

```
hdfs ec -listPolicies
Erasure Coding Policies:
ErasureCodingPolicy=[Name=RS-10-4-1024k, Schema=[ECSchema=[Codec=rs,
 numDataUnits=10, numParityUnits=4]], CellSize=1048576, Id=5], State=DISABLED
ErasureCodingPolicy=[Name=RS-3-2-1024k, Schema=[ECSchema=[Codec=rs,
 numDataUnits=3, numParityUnits=2]], CellSize=1048576, Id=2], State=DISABLED
ErasureCodingPolicy=[Name=RS-6-3-1024k, Schema=[ECSchema=[Codec=rs,
 numDataUnits=6, numParityUnits=3]], CellSize=1048576, Id=1], State=ENABLED
ErasureCodingPolicy=[Name=RS-LEGACY-6-3-1024k, Schema=[ECSchema=[Codec=rs-
legacy, numDataUnits=6, numParityUnits=3]], CellSize=1048576, Id=3],
 State=DISABLED
ErasureCodingPolicy=[Name=XOR-2-1-1024k, Schema=[ECSchema=[Codec=xor,
 numDataUnits=2, numParityUnits=1]], CellSize=1048576, Id=4], State=DISABLED
```

Enabling an erasure coding policy

In the previous example, the list of erasure coding policies indicates that RS-6-3-1024k is already enabled. If required, you can enable additional policies as mentioned in the following example:

```
hdfs ec -enablePolicy -policy RS-3-2-1024k
Erasure coding policy RS-3-2-1024k is enabled
```

Setting an erasure coding policy

The following example shows how you can set the erasure coding policy RS-6-3-1024k on a particular directory:

```
hdfs ec -setPolicy -path /data/dir1 -policy RS-6-3-1024k
Set erasure coding policy RS-6-3-1024k on /data/dir1
```

To confirm whether the specified directory has the erasure coding policy applied, run the hdfs ec -getPolicy command:

```
hdfs ec -getPolicy -path /data/dir1
RS-6-3-1024k
```

Checking the block status on an erasure-coded directory

After enabling erasure coding on a directory, you can check the block status by running the hdfs fsck command. The following example output shows the status of the erasure-coded blocks:

```
hdfs fsck /data/dir1
.
.
.
Erasure Coded Block Groups:
Total size: 434424 B
Total files: 1
Total block groups (validated): 1 (avg. block group size 434424 B)
Minimally erasure-coded block groups: 1 (100.0 %)
Over-erasure-coded block groups: 0 (0.0 %)
Under-erasure-coded block groups: 0 (0.0 %)
Unsatisfactory placement block groups: 0 (0.0 %)
Average block group size: 4.0
```

```
Missing block groups: 0
Corrupt block groups: 0
Missing internal blocks: 0 (0.0 %)
FSCK ended at Fri Mar 21 19:39:11 UTC 2018 in 1 milliseconds

The filesystem under path '/data/dir1' is HEALTHY
```

Changing the erasure coding policy

You can use the hdfs ec setPolicy command to change the erasure coding policy applied on a particular directory.

```
hdfs ec -setPolicy -path /data/dir1 -policy RS-3-2-1024k
Set erasure coding policy RS-3-2-1024k on /data/dir1
```

You can check the block status after applying the new policy. The following example output shows the status of the erasure-coded blocks for a directory that has the RS-3-2-1024k policy:

```
hdfs fsck /data/dir1
.
.
.
Erasure Coded Block Groups:
Total size: 68644 B
Total files: 2
Total block groups (validated): 2 (avg. block group size 34322 B)
Minimally erasure-coded block groups: 2 (100.0 %)
Over-erasure-coded block groups: 0 (0.0 %)
Under-erasure-coded block groups: 0 (0.0 %)
Unsatisfactory placement block groups: 0 (0.0 %)
Average block group size: 2.5
Missing block groups:  0
Corrupt block groups:  0
Missing internal blocks: 0 (0.0 %)
FSCK ended at Mon Apr 09 10:11:06 UTC 2018 in 3 milliseconds


The filesystem under path '/data/dir1' is HEALTHY
```

You can apply the default 3x replication policy and check the block status as specified in the following examples:

```
hdfs ec -setPolicy -path /data/dir1 -replicate
Set erasure coding policy replication on /tmp/data1/
Warning: setting erasure coding policy on a non-empty directory will not
 automatically convert existing files to replication
```

```
hdfs fsck /data/dir1
.
.
.
Erasure Coded Block Groups:
Total size: 34322 B
Total files: 1
Total block groups (validated): 1 (avg. block group size 34322 B)
Minimally erasure-coded block groups: 1 (100.0 %)
Over-erasure-coded block groups: 0 (0.0 %)
Under-erasure-coded block groups: 0 (0.0 %)
Unsatisfactory placement block groups: 0 (0.0 %)
Average block group size: 2.0
Missing block groups:  0
Corrupt block groups:  0
Missing internal blocks: 0 (0.0 %)
FSCK ended at Tue Apr 10 04:34:14 UTC 2018 in 2 milliseconds
```

```
The filesystem under path '/data/dir1' is HEALTHY
```

Increasing storage capacity with HDFS compression

Linux supports GzipCodec, DefaultCodec, BZip2Codec, LzoCodec, and SnappyCodec. Typically, GzipCodec is used for HDFS compression.

To configure data compression, you can either enable a data compression codec, for example, GZipCodec, as the default or use the codec from the command line with a one-time job.

Enable GZipCodec as the default compression codec

For the MapReduce framework, update relevant properties in core-site.xml and mapred-site.xml to enable GZipCodec as the default compression codec.

Procedure

1. Edit the core-site.xml file on the NameNode host machine.

```
<property>
  <name>io.compression.codecs</name>
  <value>org.apache.hadoop.io.compress.GzipCodec,
    org.apache.hadoop.io.compress.DefaultCodec,com.hadoop.compression.lzo.
    LzoCodec,org.apache.hadoop.io.compress.SnappyCodec</value>
  <description>A list of the compression codec classes that can be used
    for compression/decompression.</description>
</property>
```

2. Edit the mapred-site.xml file on the JobTracker host machine.

```
<property>
  <name>mapreduce.map.output.compress</name>
  <value>true</value>
</property>

<property>
  <name>mapreduce.map.output.compress.codec</name>
  <value>org.apache.hadoop.io.compress.GzipCodec</value>
</property>

<property>
  <name>mapreduce.output.fileoutputformat.compress.type</name>
  <value>BLOCK</value>
</property>
```

3. Enable the following two configuration parameters to enable job output compression. Edit the mapred-site.xml file on the Resource Manager host machine.

```
<property>
  <name>mapreduce.output.fileoutputformat.compress</name>
  <value>true</value>
</property>

<property>
  <name>mapreduce.output.fileoutputformat.compress.codec</name>
  <value>org.apache.hadoop.io.compress.GzipCodec</value>
</property>
```

4. Restart the cluster.

Use GZipCodec with a one-time job

You can configure GZipcodec to compress the output of a MapReduce job.

Procedure

To use GzipCodec with a one-time only job, add the options to configure compression for the MapReduce job and configure GZipCodec for the output of the job.

```
hadoop jar hadoop-examples-1.1.0-SNAPSHOT.jar sort sbr"-Dmapred.compress.map
.output=true"
sbr"-Dmapred.map.output.compression.codec=org.apache.hadoop.io.compress.G
zipCodec"
sbr "-Dmapred.output.compress=true"
sbr"-Dmapred.output.compression.codec=org.apache.hadoop.io.compress.GzipC
odec"sbr -outKey org.apache.hadoop.io.Textsbr
-outValue org.apache.hadoop.io.Text input output
```

Set HDFS quotas

As an administrator, you can set HDFS quotas for the number of file and directory names used and the amount of space used by directories after the Reports Manager service is installed as part of Cloudera Management Service. Before you can set a quota, the Reports Manager must index the file or directory.

Considerations for working with HDFS quotas

- The quotas for names and the quotas for space are independent of each other.
- You cannot create more files and directories if their creation would cause the quotas to exceed.
- Block allocation fails if the quota prevents a full block from being written.
- If you are using replication, each replica of a block counts against the quota.

File count limits

- The file count quota is a limit on the number of file and directory names in the directory configured.
- A directory counts against its own quota. Therefore, a quota value of 1 forces the directory to remain empty.
- File counts are based on the intended replication factor for the files. Changing the replication factor for a file will increase or decrease the corresponding quota values.

Disk space limits

- The space quota is a hard limit on the number of bytes used by files in the tree rooted at the directory being configured.
- Each replica of a block counts against the quota.
- The disk space quota calculation takes replication into account. Therefore, the calculation uses the replicated size of each file and not the user-facing size.
- The disk space quota calculation includes open files (that are being written) and files that already written.
- Block allocations for files being written fail if the quota does not allow a full block to be written.

Related Information

HDFS Quotas Guide

Set up a storage policy for HDFS

Setting HDFS quotas in Cloudera Manager

You can set the file count or space quotas for a directory using Cloudera Manager. Ensure that you have administrator privileges before you set the quotas.

Procedure

1. Go to the Cloudera Manager Clusters *HDFS SERVICE* File Browser tab.
2. Browse the file system to find the directory for which you want to set quotas.
3. Click the directory name so that it appears in the gray panel above the listing of its contents and in the detail section to the right of the File Browser table.
4. Click the Edit Quota option for the directory.

   Tip: The Edit Quota option appears if the Reports Manager service is installed as part of Cloudera Management Service.

5. Set the file count or disk space limits for the directory in the Manage Quota dialog box.

   Tip: Before you can set a quota, the Reports Manager must index the file or directory.

6. Click OK.

Configuring heterogeneous storage in HDFS

A variety of storage types are supported with HDFS. You can choose which storage type to assign to each DataNode Data Directory. Specifying a storage type allows you to optimize your data usage and lower your costs, based on your data usage frequency.

Each DataNode in a cluster is configured with a set of data directories. You can configure each data directory with a storage type. The storage policy dictates which storage types to use when storing the file or directory.

Some reasons to consider using different types of storage are as follows:

- You have datasets with temporal locality (for example, time-series data). The latest data can be loaded initially into SSD for improved performance, then migrated out to disk as it ages.
- You need to move cold data to denser archival storage because the data will rarely be accessed and archival storage is much cheaper. This could be done with simple age-out policies: for example, moving data older than six months to archival storage.

HDFS storage types

The storage type associated with HDFS identifies the underlying storage media.

HDFS supports the following storage types:

- ARCHIVE - Archival storage is for very dense storage and is useful for rarely accessed data. This storage type is typically cheaper per TB than normal hard disks.
- DISK - Hard disk drives are relatively inexpensive and provide sequential I/O performance. This is the default storage type.
- SSD - Solid state drives are useful for storing hot data and I/O-intensive applications.
- RAM_DISK - This special in-memory storage type is used to accelerate low-durability, single-replica writes.

When you add the DataNode Data Directory, you can specify which type of storage it uses, by prefixing the path with the storage type, in brackets. If you do not specify a storage type, it is assumed to be DISK.

HDFS storage policies

A storage policy contains information that describes the type of storage to use. This policy also defines the fallback storage type if the primary type is out of space or out of quota. If a target storage type is not available, HDFS attempts to place replicas on the default storage type.

Each storage policy consists of a policy ID, a policy name, a list of storage types, a list of fallback storage types for file creation, and a list of fallback storage types for replication.

HDFS has six preconfigured storage policies.

- Hot - All replicas are stored on DISK.
- Cold - All replicas are stored ARCHIVE.
- Warm - One replica is stored on DISK and the others are stored on ARCHIVE.
- All_SSD - All replicas are stored on SSD.
- One_SSD - One replica is stored on SSD and the others are stored on DISK.
- Lazy_Persist - The replica is written to RAM_DISK and then lazily persisted to DISK.

Note: You cannot create your own storage policy. You must use one of the six pre-configured policies. HDFS clients such as HBase may support different storage policies.

Commands for configuring storage policies

Depending on your requirements, use the `hdfs storagepolicy` sub-commands to set storage policies or list the storage policies applied on a file or a directory.

Setting storage policies

Command: `hdfs storagepolicies -setStoragePolicy -path <path> -policy <policyName>`

Argument	Description
<path>	The path to a directory or file.
<policyName>	The name of the storage policy.

hdfs storagepolicies -setStoragePolicy -path /cold1 -policy COLD

Listing storage policies

Command: `hdfs storagepolicies -getStoragePolicy -path <path>`

Argument	Description
<path>	The path to a directory or file.

hdfs storagepolicies -getStoragePolicy -path /cold1

Unsetting/clearing storage policies

Command: `hdfs storagepolicies -unsetStoragePolicy -path <path>`

Argument	Description
<path>	The path to a directory or file.

hdfs storagepolicies -unsetStoragePolicy -path /cold1

Set up a storage policy for HDFS

You can use Cloudera Manager to set up storage policy on a DataNode data directory.

Procedure

1. Check the HDFS Service Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml to be sure that dfs .storage.policy.enabled has not been changed from its default value of true.

2. Specify the storage types for each DataNode Data Directory that is not a standard disk, by adding the storage type in brackets at the beginning of the directory path.

   For example:

   ```
   [SSD]/dfs/dn1
   [DISK]/dfs/dn2
   [ARCHIVE]/dfs/dn3
   ```

3. Open a terminal session on any HDFS host and run the `hdfs storagepolicies -setStoragePolicy` command for each path on which you want to set a storage policy.

   ```
   hdfs storagepolicies -setStoragePolicy -path <path> -policy <policy>
   path_to_file_or_directory -policy policy_name
   ```

4. To move the data to the appropriate storage based on the current storage policy, use the `mover` utility, from any HDFS host.

   Use `mover -h` to get a list of available options. To migrate all data at once (this may take a long time), you can set the path to /.

   ```
   hdfs mover -p <path>
   ```

   Note: Quotas are enforced at the time you set the storage policy or when writing the file, not when quotas are changed. The Mover tool does not recognize quota violations. It only verifies that a file is stored on the storage types specified in its policy.

Related Information

Set HDFS quotas

Set up SSD storage using Cloudera Manager

You can use Cloudera Manager to set up SSD storage for your data directories.

Procedure

1. Set up your cluster normally, but customize your DataNodes with the [ssd] prefix for data directories.

   Adding [ssd] can also be done after initial setup (which requires an extra HDFS restart).
2. Stop HBase.
3. Using the HDFS client, move /hbase to /hbase_backup.
4. Re-create /hbase using the Cloudera Manager command in the HBase service (this ensures that proper permissions are used).
5. Using the HDFS client, set the storage policy for /hbase to be SSD only.
6. Use the DistCp to copy /hbase_backup to /hbase.

   ```
   hadoop distcp /hbase_backup /hbase
   ```

7. Start HBase.

Configure archival storage

To configure archival storage for a DataNode, you must assign the ARCHIVE storage type to the DataNode, set storage policies, and move blocks that violate the storage policy to the appropriate storage type.

Procedure

1. Shut down the DataNode.

2. Assign the ARCHIVE Storage Type to the DataNode.

   You can use the dfs.datanode.data.dir property in the /etc/hadoop/conf/hdfs-site.xml file to assign the ARCHIVE storage type to a DataNode.

   The dfs.datanode.data.dir property determines where on the local filesystem a DataNode should store its blocks.

   If you specify a comma-delimited list of directories, data will be stored in all named directories, typically on different devices. Directories that do not exist are ignored. You can specify that each directory resides on a different type of storage: DISK, SSD, ARCHIVE, or RAM_DISK.

   To specify a DataNode as DISK storage, specify [DISK] and a local file system path. For example:

```
<property>
  <name>dfs.datanode.data.dir</name>
  <value>[DISK]/grid/1/tmp/data_trunk</value>
</property>
```

   To specify a DataNode as ARCHIVE storage, insert [ARCHIVE] at the beginning of the local file system path. For example:

```
<property>
  <name>dfs.datanode.data.dir</name>
  <value>[ARCHIVE]/grid/1/tmp/data_trunk</value>
</property>
```

3. Depending on your requirements, either set a storage policy or list the already applied storage policies on a specified file or directory.

To	Run this command
Set a storage policy	hdfs storagepolicies -setStoragePolicy <path> <policyname>
List storage policies	hdfs storagepolicies -getStoragePolicy <path>

   Note: When you update a storage policy setting on a file or directory, the new policy is *not* automatically enforced. You must use the HDFS mover data migration tool to actually move blocks as specified by the new storage policy.

4. Start the DataNode.
5. Use the HDFS mover tool according to the specified storage policy.

   The HDFS mover data migration tool scans the specified files in HDFS and verifies if the block placement satisfies the storage policy. For the blocks that violate the storage policy, the tool moves the replicas to a different storage type in order to fulfill the storage policy requirements.

The HDFS mover command

You can use the `hdfs mover` command to move replicas of data blocks that violate the storage policy set on a file or a directory to a storage type that fulfills the policy requirements.

Command: `hdfs mover [-p <files/dirs> | -f <local file name>]`

Argument	Description
-f <local file>	Specify a local file containing a list of HDFS files or directories to migrate.
-p <files/dirs>	Specify a space-separated list of HDFS files or directories to migrate.

Note: When both -p and -f options are omitted, the default path is the root directory.

Balancing data across an HDFS cluster

The HDFS Balancer is a tool for balancing the data across the storage devices of a HDFS cluster.

You can also specify the source DataNodes, to free up the spaces in particular DataNodes. You can use a block distribution application to pin its block replicas to particular DataNodes so that the pinned replicas are not moved for cluster balancing.

Why HDFS data becomes unbalanced

Factors such as addition of DataNodes, block allocation in HDFS, and behavior of the client application can lead to the data stored in HDFS clusters becoming unbalanced.

Addition of DataNodes

When new DataNodes are added to a cluster, newly created blocks are written to these DataNodes from time to time. The existing blocks are not moved to them without using the HDFS Balancer.

Behavior of the client application

In some cases, a client application might not write data uniformly across the DataNode machines. A client application might be skewed in writing data, and might always write to some particular machines but not others. HBase is an example of such a client application. In other cases, the client application is not skewed by design, for example, MapReduce or YARN jobs.

The data is skewed so that some of the jobs write significantly more data than others. When a Datanode receives the data directly from the client, it stores a copy to its local storage for preserving data locality. The DataNodes receiving more data generally have higher storage utilization.

Block allocation in HDFS

HDFS uses a constraint satisfaction algorithm to allocate file blocks. Once the constraints are satisfied, HDFS allocates a block by randomly selecting a storage device from the candidate set uniformly. For large clusters, the blocks are essentially allocated randomly in a uniform distribution, provided that the client applications write data to HDFS uniformly across the DataNode machines. Uniform random allocation might not result in a uniform data distribution because of randomness. This is generally not a problem when the cluster has sufficient space. The problem becomes serious when the cluster is nearly full.

Configurations and CLI options for the HDFS Balancer

You can configure the HDFS Balancer by changing various configuration options or by using the command line.

Properties for configuring the Balancer

Depending on your requirements, you can configure various properties for the HDFS Balancer.

dfs.datanode.balance.max.concurrent.moves

Limits the maximum number of concurrent block moves that a DataNode is allowed for balancing the cluster. If you set this configuration in a DataNode, the DataNode throws an exception when the limit is exceeded. If you set this configuration in the HDFS Balancer, the HDFS Balancer schedules concurrent block movements within the specified limit. The DataNode setting and the HDFS Balancer setting can be different. As both settings impose a restriction, an effective setting is the minimum of them.

Cloudera recommends you to set this to the highest possible value in DataNodes and adjust the runtime value in the HDFS Balancer to gain the flexibility. The default value is 100.

You can reconfigure without DataNode restart. Follow these steps to reconfigure a DataNode:

1. Change the value of *DFS.DATANODE.BALANCE.MAX.CONCURRENT.MOVES* from the Configuration tab of the HDFS service from Cloudera Manager.
2. Refresh the cluster.

You can use the default value of 100 as the maximum number of concurrent block moves in most of the situations. If you want to set it to a lower value, you can consider a value between 25 and 50. The recommended maximum value for this parameter is 200.

dfs.datanode.balance.bandwidthPerSec

Limits the bandwidth in each DataNode using for balancing the cluster. Changing this configuration requires restarting DataNodes.

The default is 100 MB/s.

To dynamically change balancer bandwidth, use the following command:

```
hdfs dfsadmin -setBalancerBandwidth <bandwidth in bytes per seco
nd>
```

dfs.balancer.moverThreads

Limits the number of total concurrent moves for balancing in the entire cluster. Set this property to the number of threads in the HDFS Balancer for moving blocks. Each block move requires a thread.

The default is 1000.

dfs.balancer.max-size-to-move

With each iteration, the HDFS Balancer chooses DataNodes in pairs and moves data between the DataNode pairs. Limits the maximum size of data that the HDFS Balancer moves between a chosen DataNode pair. If you increase this configuration when the network and disk are not saturated, increases the data transfer between the DataNode pair in each iteration while the duration of an iteration remains about the same.

The default is 10GB.

dfs.balancer.getBlocks.size

Specifies the total data size of the block list returned by a getBlocks(..).

When the HDFS Balancer moves a certain amount of data between source and destination DataNodes, it repeatedly invokes the getBlocks(..) rpc to the Namenode to get lists of blocks from the source DataNode until the required amount of data is scheduled.

The default is 2GB.

dfs.balancer.getBlocks.min-block-size

Specifies the minimum block size for the blocks used to balance the cluster.

The default is 10MB.

Note: If the majority of files in the cluster have block size smaller than 10MB, then set a lower value for this parameter, because the HDFS Balancer might not be able to find enough blocks greater than this size to move.

dfs.datanode.block-pinning.enabled

Specifies if block-pinning is enabled. When you create a file, a user application can specify a list of favorable DataNodes by way of the file creation API in DistributedFileSystem. The NameNode uses its best effort, allocating blocks to the favorable DataNodes. If dfs.datanode.block-pinning.enabled is set to true, if a block replica is written to a favorable DataNode, it is “pinned” to that DataNode. The pinned replicas are not moved for cluster balancing to keep them stored in the

specified favorable DataNodes. This feature is useful for block distribution aware user applications such as HBase.

The default is false.

Balancer commands

You can use various command line options with the `hdfs balancer` command to work with the HDFS Balancer.

Balancing policy, threshold, and blockpools

[-policy <policy>]

Specifies which policy to use to determine if a cluster is balanced.

The two supported policies are blockpool and datanode. Setting the policy to blockpool means that the cluster is balanced if each pool in each node is balanced while datanode means that a cluster is balanced if each DataNode is balanced.

The default policy is datanode.

[-threshold <threshold>]

Specifies a number in [1.0, 100.0] representing the acceptable threshold of the percentage of storage capacity so that storage utilization outside the average +/- the threshold is considered as over/under utilized.

The default threshold is 10.0.

[-blockpools <comma-separated list of blockpool ids>]

Specifies a list of block pools on which the HDFS Balancer runs. If the list is empty, the HDFS Balancer runs on all existing block pools.

The default value is an empty list.

Include and exclude lists

[-include [-f <hosts-file> | <comma-separated list of hosts>]]

When the include list is non-empty, only the DataNodes specified in the list are balanced by the HDFS Balancer. An empty include list means including all the DataNodes in the cluster. The default value is an empty list.

[-exclude [-f <hosts-file> | <comma-separated list of hosts>]]

The DataNodes specified in the exclude list are excluded so that the HDFS Balancer does not balance those DataNodes. An empty exclude list means that no DataNodes are excluded. When a DataNode is specified in both in the include list and the exclude list, the DataNode is excluded. The default value is an empty list.

Idle-iterations and run during upgrade

[-idleiterations <idleiterations>]

Specifies the number of consecutive iterations in which no blocks have been moved before the HDFS Balancer terminates with the NO_MOVE_PROGRESS exit status.

Specify -1 for infinite iterations. The default is 5.

[-runDuringUpgrade]

If specified, the HDFS Balancer runs even if there is an ongoing HDFS upgrade. If not specified, the HDFS Balancer terminates with the UNFINALIZED_UPGRADE exit status.

When there is no ongoing upgrade, this option has no effect. It is usually not desirable to run HDFS Balancer during upgrade. To support rollback, blocks being deleted from HDFS are moved to the

internal trash directory in DataNodes and not actually deleted. Running the HDFS Balancer during upgrading cannot reduce the usage of any DataNode storage.

Source DataNodes

[-source [-f <hosts-file> | <comma-separated list of hosts>]]

Specifies the source DataNode list. The HDFS Balancer selects blocks to move from only the specified DataNodes. When the list is empty, all the DataNodes are chosen as a source. The option can be used to free up the space of some particular DataNodes in the cluster. Without the -source option, the HDFS Balancer can be inefficient in some cases.

The default value is an empty list.

The following table shows an example, where the average utilization is 25% so that D2 is within the 10% threshold. It is unnecessary to move any blocks from or to D2. Without specifying the source nodes, HDFS Balancer first moves blocks from D2 to D3, D4 and D5, since they are under the same rack, and then moves blocks from D1 to D2, D3, D4 and D5. By specifying D1 as the source node, HDFS Balancer directly moves blocks from D1 to D3, D4 and D5.

Table 1: Example of utilization movement

Datanodes (with the same capacity)	Utilization	Rack
D1	95%	A
D2	30%	B
D3, D4, and D5	0%	B

Recommended configurations for the Balancer

The HDFS Balancer can run in either Background or Fast modes. Depending on the mode in which you want the Balancer to run, you can set various properties to recommended values.

Background and fast modes

HDFS Balancer runs as a background process. The cluster serves other jobs and applications at the same time.

Fast Mode

HDFS Balancer runs at maximum (fast) speed.

Table 2: DataNode configuration properties

Property	Default	Background Mode	Fast Mode
dfs.datanode.balance.max.concurrent.moves	50	4 x (# of disks)	4 x (# of disks)
dfs.datanode.balance.bandwidthPerSec	10485760 (10 MB)	use default	10737418240 (10 GB)

Table 3: Balancer configuration properties

Property	Default	Background Mode	Fast Mode
dfs.datanode.balance.max.concurrent.moves	50	# of disks	4 x (# of disks)
dfs.balancer.moverThreads	1000	use default	20,000
dfs.balancer.max-size-to-move	10737418240 (10 GB)	1073741824 (1GB)	107374182400 (100 GB)
dfs.balancer.getBlocks.min-block-size	10485760 (10 MB)	use default	104857600 (100 MB)

Troubleshooting tip

When the block report length exceeds the configured limit, you might receive the following exception in the NameNode logs:

```
java.io.IOException: Requested data length 141557760 is longer than maximum
configured RPC length 134217728
```

Causes

When there are too many blocks in one or more Datanodes, for example in the range of several millions or 10M+, it leads to a large size of block report from those particular Datanodes, and the block report length exceeds the default value of 128M. In that case the NameNode only receives a part of block report instead of the entire bock report as the block report gets truncated at the default configured length of 128M. Since the HDFS Balancer receives block information from NameNode, with wrong information of block number the balancer might not move the blocks from the Datanode with the above issue.

Solutions

Increase the value of ipc.maximum.data.length to 256M or above the reported data length from the above exception, so that the NameNode can receive a large size of block report. The balancer would then be able to progress further.

Note: If the above mentioned exception is sustained in the NameNode logs for a prolonged period of time, then the property modification would be required regardless of whether the balancer needs to be run or not and even for normal and correct HDFS operations.

Configuring and running the HDFS balancer using Cloudera Manager

Learn how to run HDFS Balancer by using Cloudera Manager. You configure the balancer threshold, concurrent moves, and block size. Then you run the HDFS Balancer. You also learn about recommended values for the HDFS Balancer.

In Cloudera Manager, the HDFS balancer utility is implemented by the Balancer role. The Balancer role usually shows a health of None on the HDFS Instances tab because it does not run continuously.

The Balancer role is normally added (by default) when the HDFS service is installed. If it has not been added, you must add a Balancer role to rebalance HDFS and to see the Rebalance action.

Configuring the balancer threshold

Learn how to configure the Balancer Threshold property, before using HDFS Balancer feature.

About this task

The Balancer has a default threshold of 10%, which ensures that disk usage on each DataNode differs from the overall usage in the cluster by no more than 10%. For example, if overall usage across all the DataNodes in the cluster is 40% of the cluster's total disk-storage capacity, the script ensures that DataNode disk usage is between 30% and 50% of the DataNode disk-storage capacity. To change the threshold, perform the following steps:

Procedure

1. Got to the HDFS service in Cloudera Manager.
2. Select the Configuration tab.
3. Select Scope Balancer .
4. Select Category Main .
5. Set the Rebalancing Threshold property.

   To apply this configuration property to other role groups as needed, edit the value for the appropriate role group. For more information, see Modifying Configuration Properties Using Cloudera Manager.

6. Enter a reason for change, and then click Save Changes.

Configuring concurrent moves

Learn how to configure concurrent moves for the DataNode and Balancer to prevent the balancer from taking too many resources from the DataNode and interfering with normal cluster operations.

About this task

The property dfs.datanode.balance.max.concurrent.moves sets the maximum number of threads used by the DataNode balancer for pending moves. It is a throttling mechanism to prevent the balancer from taking too many resources from the DataNode and interfering with normal cluster operations. Increasing the value allows the balancing process to complete more quickly. Decreasing the value allows rebalancing to complete more slowly, but is less likely to compete for resources with other tasks on the DataNode. To use this property, you need to set the value on both the DataNode and the Balancer.

Procedure

1. Configure concurrent moves on the DataNode.
   a) Go to the HDFS service.
   b) Select the Configuration tab.
   c) Search for DataNode Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml.
   d) Add the following code to the configuration field, for example, setting the value to 50.

```
<property>
  <name>dfs.datanode.balance.max.concurrent.moves</name>
  <value>50</value>
</property>
```

   e) Click Save Changes.
   f) Restart the DataNode.
2. Configure concurrent moves on the Balancer.
   a) Go to the HDFS service.
   b) Select the Configuration tab.
   c) Search for Balancer Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml.
   d) Add the following code to the configuration field, for example, setting the value to 50.

```
<property>
  <name>dfs.datanode.balance.max.concurrent.moves</name>
  <value>50</value>
</property>
```

   e) Click Save Changes.

Recommended configurations for the balancer

Depending on whether you want the HDFS Balancer to run in the background or at maximum speed, you can configure the values of certain properties through safety valves.

Property	Values for Running the Balancer in the Background	Value for Running the Balancer at Maximum Speed
DataNode		
dfs.datanode.balance.bandwidthPerSec	10 MB	10 GB
Balancer		
dfs.balancer.moverThreads	1000	20000
dfs.balancer.max-size-to-move	10 GB	100 GB
dfs.balancer.getBlocks.min-block-size	10 MB	100 MB

Running the balancer

Learn how to run the HDFS Balancer.

Procedure

1. Go to the HDFS service.
2. Ensure that the service has a Balancer role.
3. Select Actions Rebalance .
4. Click Rebalance to confirm.

   If you see a Finished status, the Balancer ran successfully.

Configuring block size

Learn how to configure the block metadata batch size and minimum block size for HDFS.

About this task

The Block Metadata Batch Size property configures the amount of block metadata that gets retrieved. The Minimum Block Size property configures the smallest block to consider for moving. Tuning these properties can improve performance during balancing.

Procedure

1. In the Cloudera Admin Console, select Clusters <HDFS cluster> .
2. Select the Configuration tab.
3. Search for the following properties:
   - Block Metadata Batch Size (dfs.balancer.getBlocks.size)
   - Minimum Block Size (dfs.balancer.getBlocks.min-block-size)
4. Set the values for each, and click Save Changes.

Cluster balancing algorithm

The HDFS Balancer runs in iterations. Each iteration contains the following four steps: storage group classification, storage group pairing, block move scheduling, and block move execution.

Storage group classification

The HDFS Balancer first invokes the `getLiveDatanodeStorageReport` rpc to the Namenode to the storage report for all the storages in all Datanodes. The storage report contains storage utilization information such as capacity, dfs used space, remaining space, and so forth, for each storage in each DataNode.

A Datanode can contain multiple storages and the storages can have different storage types. A storage group Gi,T is defined to be the group of all the storages with the same storage type T in Datanode i. For example, Gi,DISK is the storage group of all the DISK storages in Datanode i. For each storage type T in each DataNode i, HDFS Balancer computes Storage Group Utilization (%)

Ui,T = 100% (storage group used space)/(storage group capacity),

and Average Utilization (%)

Uavg,T = 100% * (sum of all used spaces)/(sum of all capacities).

Let # be the threshold parameter (default is 10%) and GI,T be the storage group with storage type T in DataNode I.

```
                    Over-Utilized:       {Gi,T :  Uavg,T + # < Ui,T},
Average + Threshold ----------------------------------------------------
----------------------
                    Above-Average: {Gi,T :  Uavg,T < Ui,T <= Uavg,T + #},
Average -------------------------------------------------------------
----------------------
```

```
                     Below-Average: {Gi,T :  Uavg,T - # <= Ui,T <= Uavg,T},
Average - Threshold ----------------------------------------------------
----------------------
                     Under-Utilized:      {Gi,T :  Ui,T < Uavg,T - # }.
```

A storage group is over-utilized or under-utilized if its utilization is larger or smaller than the difference between the average and the threshold. A storage group is above-average or below-average if its utilization is larger or smaller than average but within the threshold.

If there are no over-utilized storages and no under-utilized storages, the cluster is said to be balanced. The HDFS Balancer terminates with a SUCCESS state. Otherwise, it continues with storage group pairing.

Storage group pairing

The HDFS Balancer selects over-utilized or above-average storage as source storage, and under-utilized or below-average storage as target storage. It pairs a source storage group with a target storage group (source # target) in a priority order depending on whether or not the source and the target storage reside in the same rack.

The Balancer uses the following priority orders for pairing storage groups from the source and the target.

- Same-Rack (where the source and the target storage reside in the same rack)

  Over-Utilized # Under-Utilize

  Over-Utilized # Below-Average

  Above-Average # Under-Utilized
- Any (where the source and target storage do not reside in the same rack)

  Over-Utilized # Under-Utilized

  Over-Utilized # Below-Average

  Above-Average # Under-Utilized

Block move scheduling

For each source-target pair, the HDFS Balancer chooses block replicas from the source storage groups and schedules block moves.

A block replica in a source DataNode is a good candidate if it satisfies all of the following conditions:

- The storage type of the block replica in the source DataNode is the same as the target storage type.
- The storage type of the block replica is not already scheduled.
- The target does not already have the same block replica.
- The number of racks of the block is not reduced after the move.

Logically, the HDFS Balancer schedules a block replica to be “moved” from a source storage group to a target storage group. In practice, a block usually has multiple replicas. The block move can be done by first copying the replica from a proxy, which can be any storage group containing one of the replicas of the block, to the target storage group, and then deleting the replica in the source storage group.

After a candidate block in the source DataNode is specified, the HDFS Balancer selects a storage group containing the same replica as the proxy. The HDFS Balancer selects the closest storage group as the proxy in order to minimize the network traffic.

When it is impossible to schedule a move, the HDFS Balancer terminates with a NO_MOVE_BLOCK exit status.

Block move execution

The HDFS Balancer dispatches a scheduled block move by invoking the DataTransferProtocol.replaceBlock(..) method to the target DataNode.

The Balancer specifies the proxy, and the source as delete-hint in the method call. The target DataNode copies the replica directly from the proxy to its local storage. When the copying process has been completed, the target DataNode reports the new replica to the NameNode with the delete-hint. NameNode uses delete-hint to delete the extra replica, that is, delete the replica stored in the source.

After all block moves are dispatched, the HDFS Balancer waits until all the moves are completed. Then, the HDFS Balancer continues running a new iteration and repeats all of the steps. If the scheduled moves fail for 5 consecutive iterations, the HDFS Balancer terminates with a NO_MOVE_PROGRESS exit status.

Exit statuses for the HDFS Balancer

The HDFS Balancer concludes a cluster balancing operation with a specific exit status that indicates whether the operation succeeded or failed, with supporting reasons.

Table 4: Exit statuses for the HDFS balancer

Status	Value	Description
SUCCESS	0	The cluster is balanced. There are no over or under-utilized storages, with regard to the specified threshold.
ALREADY_RUNNING	-1	Another HDFS Balancer is running.
NO_MOVE_BLOCK	-2	The HDFS Balancer is not able to schedule a move.
NO_MOVE_PROGRESS	-3	All of the scheduled moves have failed for 5 consecutive iterations.
IO_EXCEPTION	-4	An IOException occurred.
ILLEGAL_ARGUMENTS	-5	An invalid argument in the command or configuration occurred.
INTERUPTED	-6	The HDFS Balancer process was interrupted.
UNFINALIZED_UPGRADE	-7	The cluster is being upgraded.

HDFS heap sizing

You can provision an HDFS cluster for optimal performance based on the desired storage capacity.

Component	Memory	CPU	Disk
JournalNode	1 GB (default) Set this value using the Java Heap Size of JournalNode in Bytes HDFS configuration property.	1 core minimum	1 dedicated disk
NameNode	• Minimum: 1 GB (for proof-of-concept deployments) • Add an additional 1 GB for each additional 1,000,000 blocks Snapshots and encryption can increase the required heap memory. See *Sizing NameNode Heap Memory*. Set this value using the Java Heap Size of NameNode in Bytes HDFS configuration property.	Minimum of 4 dedicated cores; more may be required for larger clusters	• Minimum of 2 dedicated disks for metadata • 1 dedicated disk for log files (This disk may be shared with the operating system.) • Maximum disks: 4

Component	Memory	CPU	Disk
DataNode	Minimum: 4 GB Maximum: 8 GB Increase the memory for higher replica counts or a higher number of blocks per DataNode. When increasing the memory, Cloudera recommends an additional 1 GB of memory for every 1 million replicas above 4 million on the DataNodes. For example, 5 million replicas require 5 GB of memory. Set this value using the Java Heap Size of DataNode in Bytes HDFS configuration property.	Minimum: 4 cores. Add more cores for highly active clusters.	Minimum: 4 Maximum: 24 The maximum acceptable size will vary depending upon how large average block size is. The DN's scalability limits are mostly a function of the number of replicas per DN, not the overall number of bytes stored. That said, having ultra-dense DNs will affect recovery times in the event of machine or rack failure. Cloudera does not support exceeding 100 TB per data node. You could use 12 x 8 TB spindles or 24 x 4TB spindles. Cloudera does not support drives larger than 8 TB. Configure the disks in JBOD mode. Do not use RAID/LVM/ZFS.

Warning: Running Cloudera Runtime on storage platforms other than direct-attached physical disks can provide suboptimal performance. Cloudera Enterprise and the majority of the Hadoop platform are optimized to provide high performance by distributing work across a cluster that can utilize data locality and fast local I/O.

Optimizing performance

You can consider the following options to optimize the performance of an HDFS cluster: swapping disk drives on a DataNode, caching data, specifying racks for hosts, customizing HDFS, optimizing NameNode disk space with Hadoop archives, identifying slow DataNodes and improving them, optimizing small write operations by using DataNode memory as storage, and implementing short-circuit reads.

Improving performance with centralized cache management

Centralized cache management enables you to specify paths to directories that are cached by HDFS, thereby improving performance for applications that repeatedly access the same data.

Centralized cache management in HDFS is an explicit caching mechanism. The NameNode communicates with DataNodes that have the required data blocks available on disk, and instructs the DataNodes to cache the blocks in off-heap caches.

Benefits of centralized cache management in HDFS

Centralized cache management in HDFS offers many significant advantages such as explicit pinning, querying cached blocks for task placement, and improving cluster memory utilization.

- Explicit pinning prevents frequently used data from being evicted from memory. This is particularly important when the size of the working set exceeds the size of main memory, which is common for many HDFS workloads.
- Because DataNode caches are managed by the NameNode, applications can query the set of cached block locations when making task placement decisions. Co-locating a task with a cached block replica improves read performance.
- When a block has been cached by a DataNode, clients can use a more efficient zero-copy read API. Since checksum verification of cached data is done once by the DataNode, clients can incur essentially zero overhead when using this new API.

- Centralized caching can improve overall cluster memory utilization. When relying on the operating system buffer cache on each DataNode, repeated reads of a block will result in all n replicas of the block being pulled into the buffer cache. With centralized cache management, you can explicitly pin only m of the n replicas, thereby saving n-m memory.

Use cases for centralized cache management

Centralized cache management is useful for files that are accessed repeatedly and for mixed workloads that have performance SLAs.

- Files that are accessed repeatedly: For example, a small fact table in Hive that is often used for joins is a good candidate for caching. Conversely, caching the input of a once-yearly reporting query is probably less useful, since the historical data might only be read once.
- Mixed workloads with performance SLAs: Caching the working set of a high priority workload ensures that it does not compete with low priority workloads for disk I/O.

Centralized cache management architecture

In a centralized cache management, the NameNode is responsible for coordinating all of the DataNode off-heap caches in the cluster. The NameNode periodically receives a cache report from each DataNode. The cache report describes all of the blocks cached on the DataNode. The NameNode manages DataNode caches by piggy-backing cache and uncache commands on the DataNode heartbeat.

The following figure illustrates the centralized cached management architecture.

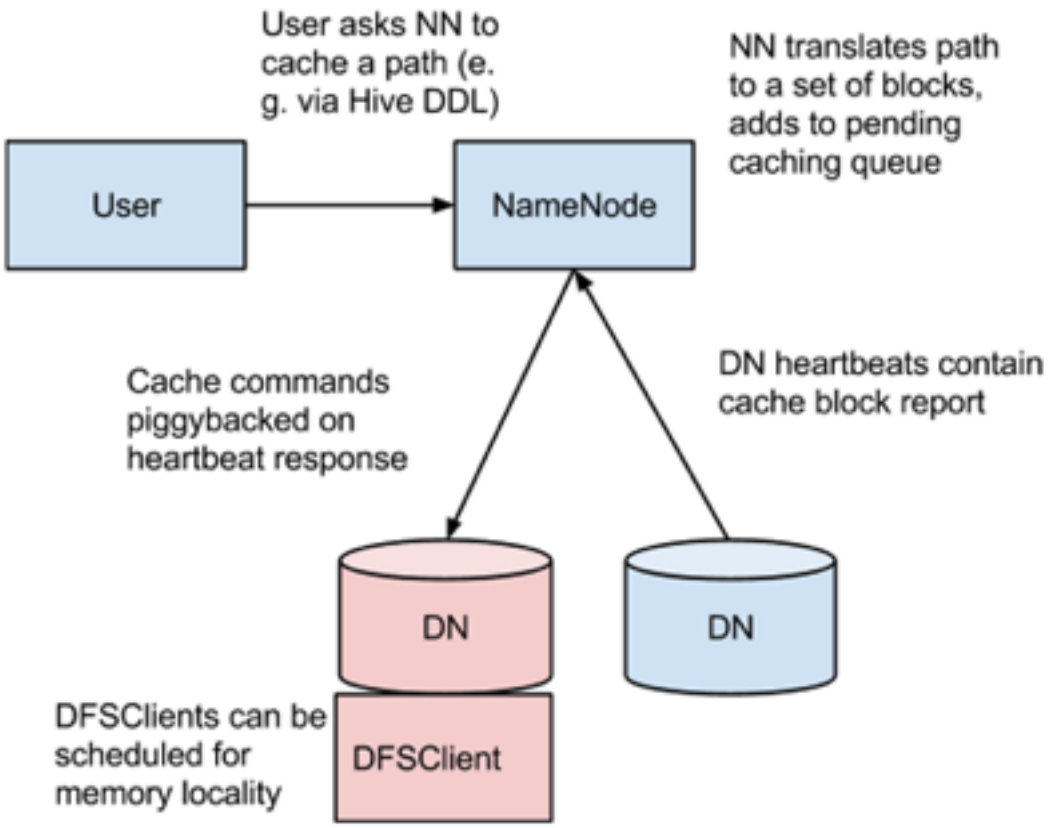

The NameNode queries its set of cache directives to determine which paths should be cached. Cache directives are persistently stored in the fsimage and edit logs, and can be added, removed, and modified through Java and command-line APIs. The NameNode also stores a set of cache pools, which are administrative entities used to group cache directives together for resource management, and to enforce permissions.

The NameNode periodically re-scans the namespace and active cache directives to determine which blocks need to be cached or uncached, and assigns caching work to DataNodes. Re-scans can also be triggered by user actions such as adding or removing a cache directive or removing a cache pool.

Cache blocks that are under construction, corrupt, or otherwise incomplete are not cached. If a Cache directive covers a symlink, the symlink target is not cached.

Caching can only be applied to directories and files.

Related Information

Caching terminology

Caching terminology

A cache directive defines the path to cache while a cache pool manages groups of cache directives.

Cache directive

Defines the path to be cached. Paths can point either directories or files. Directories are cached non-recursively, meaning only files in the first-level listing of the directory will be cached.

Cache directives also specify additional parameters, such as the cache replication factor and expiration time. The replication factor specifies the number of block replicas to cache. If multiple cache directives refer to the same file, the maximum cache replication factor is applied.

The expiration time is specified on the command line as a time-to-live (TTL), which represents a relative expiration time in the future. After a cache directive expires, it is no longer taken into consideration by the NameNode when making caching decisions.

Cache pool

An administrative entity that manages groups of cache circtives. Cache pools have UNIX-like permissions that restrict which users and groups have access to the pool. Write permissions allow users to add and remove cache directives to the pool. Read permissions allow users to list the Cache Directives in a pool, as well as additional metadata. Execute permissions are unused.

Cache pools are also used for resource management. Cache pools can enforce a maximum memory limit, which restricts the aggregate number of bytes that can be cached by directives in the pool. Normally, the sum of the pool limits will approximately equal the amount of aggregate memory reserved for HDFS caching on the cluster. Cache pools also track a number of statistics to help cluster users track what is currently cached, and to determine what else should be cached.

Cache pools can also enforce a maximum time-to-live. This restricts the maximum expiration time of directives being added to the pool.

Related Information

Centralized cache management architecture

Properties for configuring centralized caching

You must enable JNI to use centralized caching. In addition, you must configure various properties and consider the locked memory limit for configuring centralized caching.

Native libraries

In order to lock block files into memory, the DataNode relies on native JNI code found in libhadoop.so. Be sure to enable JNI if you are using HDFS centralized cache management.

Configuration properties

Configuration properties for centralized caching are specified in the hdfs-site.xml file.

Required properties

Only the following property is required:

- dfs.datanode.max.locked.memory This property determines the maximum amount of memory (in bytes) that a DataNode will use for caching. The "locked-in-memory size" ulimit (ulimit -l) of the DataNode user also needs to be increased to exceed this parameter. When setting this value, remember that you will need space in memory for other things as well, such as the DataNode and application JVM heaps, and the operating system page cache. Example:

```
<property>
  <name>dfs.datanode.max.locked.memory</name>
  <value>268435456</value>
</property>
```

Optional properties

The following properties are not required, but can be specified for tuning.

- dfs.namenode.path.based.cache.refresh.interval.ms The NameNode will use this value as the number of milliseconds between subsequent cache path re-scans. By default, this parameter is set to 300000, which is five minutes. Example:

```
<property>
  <name>dfs.namenode.path.based.cache.refresh.interval.ms</name>
  <value>300000</value>
</property>
```

- dfs.time.between.resending.caching.directives.ms The NameNode will use this value as the number of milliseconds between resending caching directives. Example:

```
<property>
  <name>dfs.time.between.resending.caching.directives.ms</name>
  <value>300000</value>
</property>
```

- dfs.datanode.fsdatasetcache.max.threads.per.volume The DataNode will use this value as the maximum number of threads per volume to use for caching new data. By default, this parameter is set to 4. Example:

```
<property>
  <name>dfs.datanode.fsdatasetcache.max.threads.per.volume</name>
  <value>4</value>
</property>
```

- dfs.cachereport.intervalMsec The DataNode will use this value as the number of milliseconds between sending a full report of its cache state to the NameNode. By default, this parameter is set to 10000, which is 10 seconds. Example:

```
<property>
  <name>dfs.cachereport.intervalMsec</name>
  <value>10000</value>
</property>
```

- dfs.namenode.path.based.cache.block.map.allocation.percent The percentage of the Java heap that will be allocated to the cached blocks map. The cached blocks map is a hash map that uses chained hashing. Smaller maps may be accessed more slowly if the number of cached blocks is large. Larger maps will consume more memory. The default value is 0.25 percent. Example:

```
<property>
  <name>dfs.namenode.path.based.cache.block.map.allocation.percent</name>
```

```
  <value>0.25</value>
</property>
```

OS limits

If you get the error "Cannot start datanode because the configured max locked memory size...is more than the datanode's available RLIMIT_MEMLOCK ulimit," this means that the operating system is imposing a lower limit on the amount of memory that you can lock than what you have configured. To fix this, you must adjust the ulimit -l value that the DataNode runs with. This value is usually configured in /etc/security/limits.conf, but this may vary depending on what operating system and distribution you are using.

You have correctly configured this value when you can run ulimit - l from the shell and get back either a higher value than what you have configured or the string "unlimited", which indicates that there is no limit. Typically, ulimit -l returns the memory lock limit in kilobytes (KB), but dfs.datanode.max.locked.memory must be specified in bytes.

For example, if the value of dfs.datanode.max.locked.memory is set to 128000 bytes:

```
<property>
  <name>dfs.datanode.max.locked.memory</name>
  <value>128000</value>
</property>
```

Set the memlock (max locked-in-memory address space) to a slightly higher value. For example, to set memlock to 130 KB (130,000 bytes) for the hdfs user, you would add the following line to /etc/security/limits.conf.

```
hdfs - memlock 130
```

Commands for using cache pools and directives

You can use the Command Line Interface (CLI) to create, modify, and list cache pools and cache directives using the hdfs cacheadmin subcommand.

Cache Directives are identified by a unique, non-repeating, 64-bit integer ID. IDs will not be reused even if a Cache Directive is removed.

Cache Pools are identified by a unique string name.

You must first create a Cache Pool, and then add Cache Directives to the Cache Pool.

Cache Pool Commands

- addPool -- Adds a new Cache Pool.

  Usage:

  ```
  hdfs cacheadmin -addPool <name> [-owner <owner>] [-group <group>]
  [-mode <mode>] [-limit <limit>] [-maxTtl <maxTtl>]
  ```

  Options:

  #### Table 5: Cache Pool Add Options

Option	Description
<name>	The name of the pool.
<owner>	The user name of the owner of the pool. Defaults to the current user.
<group>	The group that the pool is assigned to. Defaults to the primary group name of the current user.
<mode>	The UNIX-style permissions assigned to the pool. Permissions are specified in octal (for example, 0755). Pool permissions are set to 0755 by default.

Option	Description
<limit>	The maximum number of bytes that can be cached by directives in the pool, in aggregate. By default, no limit is set.
<maxTtl>	The maximum allowed time-to-live for directives being added to the pool. This can be specified in seconds, minutes, hours, and days (e.g. 120s, 30m, 4h, 2d). Valid units are [smhd]. By default, no maximum is set. A value of "never" specifies that there is no limit.

- modifyPool -- Modifies the metadata of an existing Cache Pool.

  Usage:

```
hdfs cacheadmin -modifyPool <name> [-owner <owner>] [-group <group>]
[-mode <mode>] [-limit <limit>] [-maxTtl <maxTtl>]
```

  Options:

Table 6: Cache Pool Modify Options

Option	Description
<name>	The name of the pool to modify.
<owner>	The user name of the owner of the pool.
<group>	The group that the pool is assigned to.
<mode>	The UNIX-style permissions assigned to the pool. Permissions are specified in octal (for example, 0755).
<limit>	The maximum number of bytes that can be cached by directives in the pool, in aggregate.
<maxTtl>	The maximum allowed time-to-live for directives being added to the pool. This can be specified in seconds, minutes, hours, and days (for example, 120s, 30m, 4h, 2d). Valid units are [smdh]. By default, no maximum is set. A value of "never" specifies that there is no limit.

- removePool -- Removes a Cache Pool. This command also "un-caches" paths that are associated with the pool.

  Usage:

```
hdfs cacheadmin -removePool <name>
```

  Options:

Table 7: Cache Pool Remove Options

Option	Description
<name>	The name of the Cache Pool to remove.

- listPools -- Displays information about one or more Cache Pools, such as name, owner, group, permissions, and so on.

  Usage:

  ```
  hdfs cacheadmin -listPools [-stats] [<name>]
  ```

  Options:

  Table 8: Cache Pools List Options

Option	Description
-stats	Displays additional Cache Pool statistics.
<name>	If specified, lists only the named Cache Pool.

- help -- Displays detailed information about a command.

  Usage:

  ```
  hdfs cacheadmin -help <command-name>
  ```

  Options:

  Table 9: Cache Pool Help Options

Option	Description
<command-name	Displays detailed information for the specified command name. If no command name is specified, detailed help is displayed for all commands.

Cache Directive Commands

- addDirective -- Adds a new Cache Directive.

  Usage:

  ```
  hdfs cacheadmin -addDirective -path <path> -pool <pool-name> [-force]
  [-replication <replication>] [-ttl <time-to-live>]
  ```

  Options:

  Table 10: Cache Pool Add Directive Options

Option	Description
<path>	The path to the cache directory or file.
<pool-name>	The Cache Pool to which the Cache Directive will be added. You must have Write permission for the Cache Pool in order to add new directives.
-force	Skips checking of the Cache Pool resource limits.
<replication>	The cache replication factor to use. Default setting is 1.
<time-to-live>	How long the directive is valid. This can be specified in minutes, hours and days (for example, 30m, 4h, 2d). Valid units are [smdh]. A value of "never" indicates a directive that never expires. If unspecified, the directive never expires.

- removeDirective -- Removes a Cache Directive.

  Usage:

  ```
  hdfs cacheadmin -removeDirective <id>
  ```

  Options:

  Table 11: Cache Pool Remove Directive Options

Option	Description
<id>	The ID of the Cache Directive to remove. You must have Write permission for the pool that the directive belongs to in order to remove it. You can use the -listDirectives command to display a list of Cache Directive IDs.

- removeDirectives -- Removes all of the Cache Directives in a specified path.

  Usage:

  ```
  hdfs cacheadmin -removeDirectives <path>
  ```

  Options:

  Table 12: Cache Pool Remove Directives Options

Option	Description
<path>	The path of the Cache Directives to remove. You must have Write permission for the pool that the directives belong to in order to remove them. You can use the -listDirectives command to display a list of Cache Directives.

- listDirectives -- Returns a list of Cache Directives.

  Usage:

  ```
  hdfs cacheadmin -listDirectives [-stats] [-path <path>] [-pool <pool>]
  ```

  Options:

  Table 13: Cache Pools List Directives Options

Option	Description
<path>	Lists only the Cache Directives in the specified path. If there is a Cache Directive in the <path> that belongs to a Cache Pool for which you do not have Read access, it will not be listed.
<pool>	Lists on the Cache Directives in the specified Cache Pool.
-stats	Lists path-based Cache Directive statistics.

Specifying racks for hosts

HDFS uses the network location of your deployment, such as hosts and racks, to place block replicas more intelligently and trade off performance and resilience. When placing jobs on hosts, Cloudera prefers within-rack transfers, where there is more bandwidth available, to off-rack transfers. These computations are performed with the assistance of rack awareness scripts. Cloudera Manager includes internal rack awareness scripts, but you must specify the racks where the hosts in your cluster are located.

Procedure

1. In Cloudera Manager, go to Hosts All Hosts .
2. Select the checkboxes corresponding to the hosts for which you want to specify the racks.
3. Select Actions for Selected Assign Rack .
4. Enter a rack name or ID that begins with a slash (/); for example, /rack123 or /aisle1/rack123, and click Confirm.
5. Restart the affected services.

Viewing racks assigned to cluster hosts

You can view details of the racks assigned to the cluster hosts from the All Hosts page of Cloudera Manager.

Procedure

1. In Cloudera Manager, go to Hosts All Hosts .
2. Expand the Columns drop-down list, and select Rack.

   The table displaying the host details now includes a Rack column next to Name, and lists the racks assigned to every host.

Editing rack assignments for hosts

You can edit individual host rack assignments from Cloudera Manager.

Procedure

1. In Cloudera Manager, go to Hosts All Hosts .
2. Select the checkboxes corresponding to the hosts for which you want to specify the racks.
3. Select Actions for Selected Assign Rack .
4. Enter a rack name different from that assigned to the selected hosts, and click Confirm.
5. Restart the affected services.

Customizing HDFS

You can use the dfs.user.home.base.dir property to customize the HDFS home directory. In addition, you can configure properties to control the size of the directory that holds the NameNode edits directory.

Customize the HDFS home directory

By default, the HDFS home directory is set to /user/<user_name>. Use the dfs.user.home.dir.prefix property to customize the HDFS home directory.

Procedure

In hdfs-site.xml file, set the value of the dfs.user.home.dir.prefix property.

```
<property>
  <name>dfs.user.home.dir.prefix</name>
  <value>/user</value>
  <description>Base directory of user home.</description>
</property>
```

In the example, <value> is the path to the new home directory.

Properties to set the size of the NameNode edits directory

You can configure the dfs.namenode.num.checkpoints.retained and dfs.namenode.num.extra.edits.retained properties to control the size of the directory that holds the NameNode edits directory.

- dfs.namenode.num.checkpoints.retained: The number of image checkpoint files that are retained in storage directories. All edit logs necessary to recover an up-to-date namespace from the oldest retained checkpoint are also retained.
- dfs.namenode.num.extra.edits.retained: The number of extra transactions that should be retained beyond what is minimally necessary for a NameNode restart. This can be useful for audit purposes, or for an HA setup where a remote Standby Node may have been offline for some time and require a longer backlog of retained edits in order to start again.

Optimizing NameNode disk space with Hadoop archives

Hadoop Archives (HAR) are special format archives that efficiently pack small files into HDFS blocks.

The Hadoop Distributed File System (HDFS) is designed to store and process large data sets, but HDFS can be less efficient when storing a large number of small files. When there are many small files stored in HDFS, these small files occupy a large portion of the namespace. As a result, disk space is under-utilized because of the namespace limitation.

Hadoop Archives (HAR) can be used to address the namespace limitations associated with storing many small files. A Hadoop Archive packs small files into HDFS blocks more efficiently, thereby reducing NameNode memory usage while still allowing transparent access to files. Hadoop Archives are also compatible with MapReduce, allowing transparent access to the original files by MapReduce jobs.

Overview of Hadoop archives

Storing a large number of small files in HDFS leads to inefficient utilization of space – the namespace is overutilized while the disk space might be underutilized. Hadoop Archives (HAR) address this limitation by efficiently packing small files into large files without impacting the file access.

The Hadoop Distributed File System (HDFS) is designed to store and process large (terabytes) data sets. For example, a large production cluster may have 14 PB of disk space and store 60 million files.

However, storing a large number of small files in HDFS is inefficient. A file is generally considered to be "small" when its size is substantially less than the HDFS block size. Files and blocks are name objects in HDFS, meaning that they occupy namespace (space on the NameNode). The namespace capacity of the system is therefore limited by the physical memory of the NameNode.

When there are many small files stored in the system, these small files occupy a large portion of the namespace. As a consequence, the disk space is underutilized because of the namespace limitation. In one real-world example, a production cluster had 57 million files less than 256 MB in size, with each of these files taking up one block on the NameNode. These small files used up 95% of the namespace but occupied only 30% of the cluster disk space.

Hadoop Archives (HAR) can be used to address the namespace limitations associated with storing many small files. HAR packs a number of small files into large files so that the original files can be accessed transparently (without expanding the files).

HAR increases the scalability of the system by reducing the namespace usage and decreasing the operation load in the NameNode. This improvement is orthogonal to memory optimization in the NameNode and distributing namespace management across multiple NameNodes.

Hadoop Archive is also compatible with MapReduce — it allows parallel access to the original files by MapReduce jobs.

Hadoop archive components

You can use the Hadoop archiving tool to create Hadoop Archives (HAR). The Hadoop Archive is integrated with the Hadoop file system interface. Files in a HAR are exposed transparently to users. File data in a HAR is stored in multipart files, which are indexed to retain the original separation of data.

Hadoop archiving tool

Hadoop Archives can be created using the Hadoop archiving tool. The archiving tool uses MapReduce to efficiently create Hadoop Archives in parallel. The tool can be invoked using the following command:

```
hadoop archive -archiveName name -p <parent> <src>* <dest>
```

A list of files is generated by traversing the source directories recursively, and then the list is split into map task inputs. Each map task creates a part file (about 2 GB, configurable) from a subset of the source files and outputs the metadata. Finally, a reduce task collects metadata and generates the index files. For more information, see Creating a Hadoop archive on page 45.

HAR file system

Most archival systems, such as tar, are tools for archiving and de-archiving. Generally, they do not fit into the actual file system layer and hence are not transparent to the application writer in that the archives must be expanded before use.

The Hadoop Archive is integrated with the Hadoop file system interface. The HarFileSystem implements the FileSystem interface and provides access via the har:// scheme. This exposes the archived files and directory tree structures transparently to users. Files in a HAR can be accessed directly without expanding them.

For example, if we have the following command to copy an HDFS file to a local directory:

```
hdfs dfs –get hdfs://namenode/foo/file-1 localdir
```

Suppose a Hadoop Archive bar.har is created from the foo directory. With the HAR, the command to copy the original file becomes:

```
hdfs dfs –get har://namenode/bar.har/foo/file-1 localdir
```

Users only need to change the URI paths. Alternatively, users may choose to create a symbolic link (from hdfs://namenode/foo to har://namenode/bar.har/foo in the example above), and then even the URIs do not need to be changed. In either case, HarFileSystem will be invoked automatically to provide access to the files in the HAR. Because of this transparent layer, HAR is compatible with the Hadoop APIs, MapReduce, the FS shell command-line interface, and higher-level applications such as Pig, Zebra, Streaming, Pipes, and DistCp.

HAR format data model

The Hadoop Archive data format has the following layout:

```
foo.har/_masterindex //stores hashes and offsets
foo.har/_index //stores file statuses
foo.har/part-[1..n] //stores actual file data
```

The file data is stored in multipart files, which are indexed in order to retain the original separation of data. Moreover, the file parts can be accessed in parallel by MapReduce programs. The index files also record the original directory tree structures and file status.

Creating a Hadoop archive

Hadoop Archives can be created using the Hadoop archiving tool. The archiving tool uses MapReduce to efficiently create Hadoop Archives in parallel. Use the "hadoop archive" command to invoke the Hadoop archiving tool.

Procedure

1. Run the `hadoop archive` command by specifying the archive name to create, the parent directory relative to the archive location, the source files to archive, and the destination archive location.

   ```
   hadoop archive -archiveName name -p <parent> <src>* <dest>
   ```

   The archive name must have a .har extension

   Note:

   - Archiving does not delete the source files. If you want to delete the input files after creating an archive to reduce namespace, you must manually delete the source files.
   - Although the `hadoop archive` command can be run from the host file system, the archive file is created in the HDFS file system from directories that exist in HDFS. If you reference a directory on the host file system and not HDFS, the system displays the following error:

     ```
     The resolved paths set is empty. Please check whether the srcPaths
      exist, where srcPaths
      = [</directory/path>]
     ```

2. Use the -Dhar.partfile.size=[***enter part file size***] parameter to configure the part file size based on your requirements.

Example

```
hadoop archive -Dhar.partfile.size=2147483648 -archiveName foo.har -p /user/
hadoop dir1 dir2 /user/zoo
```

This example creates an archive using /user/hadoop as the relative archive directory. The directories /user/hadoop/dir1 and /user/hadoop/dir2 are archived in the /user/zoo/foo.har archive and a part file of 2 GB is created.

List files in Hadoop archives

Use the hdfs dfs -ls command to list files in Hadoop archives.

Procedure

Run the `hdfs dfs -ls` command by specifying the archive directory location.
To specify the directories in an archive directory foo.har located in /usr/zoo, run the following command:

```
hdfs dfs -ls har:///user/zoo/foo.har/
```

Assuming the archive directory foo.har contains two directories dir1 and dir2, the command returns the following

```
har:///user/zoo/foo.har/dir1
har:///user/zoo/foo.har/dir2
```

Note:

Consider an archive created using the following command:

```
hadoop archive -archiveName foo.har -p /user/ hadoop/dir1 hadoop/dir2 /
user/zoo
```

If you list the files of the archive created in the preceding command, the command returns the following:

```
har:///user/zoo/foo.har/hadoop
har:///user/zoo/foo.har/hadoop/dir1
har:///user/zoo/foo.har/hadoop/dir2
```

Note that the modified parent argument causes the files to be archived relative to /user/.

Format for using Hadoop archives with MapReduce

To use Hadoop Archives with MapReduce, you must reference files differently than you would with the default file system. If you have a Hadoop Archive stored in HDFS in /user/zoo/foo.har, you must specify the input directory as har:///user/zoo/foo.har to use it as a MapReduce input.

Because Hadoop Archives are exposed as a file system, MapReduce can use all of the logical input files in Hadoop Archives as input.

Detecting slow DataNodes

Slow DataNodes in a Cloudera Private Cloud Base cluster can negatively impact the cluster performance. Therefore, HDFS provides a mechanism to detect and report slow DataNodes that have a negative impact on the performance of the cluster.

HDFS is designed to detect and recover from complete failure of DataNodes:

- There is no single point of failure.
- Automatic NameNode failover takes only a few seconds.
- Because data replication can be massively parallelized in large clusters, recovery from DataNode loss occurs within minutes.
- Most jobs are not affected by DataNode failures.

However, partial failures can negatively affect the performance of running DataNodes:

- Slow network connection due to a failing or misconfigured adapter.
- Bad OS or JVM settings that affect service performance.
- Slow hard disk.
- Bad disk controller.

Slow DataNodes can have a significant impact on cluster performance. A slow DataNode may continue sending heartbeats successfully, and the NameNode will keep redirecting clients to slow DataNodes. HDFS DataNode monitoring provides detection and reporting of slow DataNodes that negatively affect cluster performance.

Enable disk IO statistics

Disk IO statistics are disabled by default. To enable disk IO statistics on a Cloudera Private Cloud Base cluster, you must set the file IO sampling percentage to a non-zero value in the hdfs-site.xml file.

Procedure

1. Set the dfs.datanode.fileio.profiling.sampling.percentage property to a non-zero value in hdfs-site.xml.

```
<property>
```

```
  <name>dfs.datanode.fileio.profiling.sampling.percentage</name>
  <value>25</value>
</property>
```

Note: Sampling disk IO might have a minor impact on cluster performance.

2. Access the disk IO statistics from the NameNode JMX page at http://<namenode_host>:50070/jmx.
   In the following JMX output example, the time unit is milliseconds, and the disk is healthy because the IO latencies are low:

```
    "name" : "Hadoop:service=DataNode,name=DataNodeVolume-/data/disk2/dfs/
data/",
    "modelerType" : "DataNodeVolume-/data/disk2/dfs/data/",
    "tag.Context" : "dfs",
    "tag.Hostname" : "n001.hdfs.example.com",
    "TotalMetadataOperations" : 67,
    "MetadataOperationRateAvgTime" : 0.08955223880597014,
  ...
    "WriteIoRateNumOps" : 7321,
    "WriteIoRateAvgTime" : 0.050812730501297636
```

Enable detection of slow DataNodes

When slow DataNode detection is enabled, DataNodes collect latency statistics on their peers during write pipelines, and use periodic outlier detection to determine slow peers. The NameNode aggregates reports from all DataNodes and flags potentially slow nodes. Slow DataNode detection is disabled by default.

Procedure

1. To enable slow DataNode detection, set the value of the dfs.datanode.peer.stats.enabled property to true in hdfs-site.xml.

```
<property>
  <name>dfs.datanode.peer.stats.enabled</name>
  <value>true</value>
</property>
```

2. Access the slow DataNode statistics either from the NameNode JMX page at http://<namenode_host>:50070/jmx or from the DataNode JMX page at http://<datanode_host>:50075/jmx.
   In the following JMX output example, the time unit is milliseconds, and the peer DataNodes are healthy because the latencies are in milliseconds:

```
"name" : "Hadoop:service=DataNode,name=DataNodeInfo",
"modelerType" : "org.apache.hadoop.hdfs.server.datanode.DataNode",    "Sen
dPacketDownstreamAvgInfo" : "{
        \"[192.168.7.202:50075]RollingAvgTime\" : 1.4476967370441458,
        \"[192.168.7.201:50075]RollingAvgTime\" : 1.5569170444798432
}"
```

Allocating DataNode memory as storage

HDFS supports efficient writes of large data sets to durable storage, and also provides reliable access to the data. This works well for batch jobs that write large amounts of persistent data. Emerging classes of applications are driving use cases for writing smaller amounts of temporary data. Using DataNode memory as storage addresses the use case of applications that want to write relatively small amounts of intermediate data sets with low latency.

Writing block data to memory reduces durability, as data can be lost due to process restart before it is saved to disk. HDFS attempts to save replica data to disk in a timely manner to reduce the window of possible data loss.

DataNode memory is referenced using the RAM_DISK storage type and the LAZY_PERSIST storage policy.

HDFS storage types

The storage type associated with HDFS identifies the underlying storage media.

HDFS supports the following storage types:

- ARCHIVE - Archival storage is for very dense storage and is useful for rarely accessed data. This storage type is typically cheaper per TB than normal hard disks.
- DISK - Hard disk drives are relatively inexpensive and provide sequential I/O performance. This is the default storage type.
- SSD - Solid state drives are useful for storing hot data and I/O-intensive applications.
- RAM_DISK - This special in-memory storage type is used to accelerate low-durability, single-replica writes.

When you add the DataNode Data Directory, you can specify which type of storage it uses, by prefixing the path with the storage type, in brackets. If you do not specify a storage type, it is assumed to be DISK.

LAZY_PERSIST memory storage policy

Use the LAZY_PERSIST storage policy to store data blocks on the configured DataNode memory.

For LAZY_PERSIST, the first replica is stored on RAM_DISK (DataNode memory), and the remaining replicas are stored on DISK. The fallback storage for both creation and replication is DISK.

The following table summarizes these replication policies:

Policy ID	Policy Name	Block Placement (for n replicas)	Fallback storage for creation	Fallback storage for replication
15	LAZY_PERSIST	RAM_DISK: 1, DISK:n-1	DISK	DISK

Configure DataNode memory as storage

Configuring memory on a DataNode as storage requires you to shut down the particular DataNode, set RAM_DISK as the storage type, set the LAZY_PERSIST storage policy to store data, and then start the DataNode.

Procedure

1. Shut down the DataNode.
2. Use required mount commands to allocate a certain portion of the DataNode memory as storage.
   The following example shows how you can allocate 2GB memory for use by HDFS.

   ```
   sudo mkdir -p /mnt/hdfsramdisk
   sudo mount -t tmpfs -o size=2048m tmpfs /mnt/hdfsramdisk
   sudo mkdir -p /usr/lib/hadoop-hdfs
   ```

3. Assign the RAM_DISK storage type to ensure that HDFS can assign data to the DataNode memory configured as storage.

   To specify the DataNode as RAM_DISK storage, insert [RAM_DISK] at the beginning of the local file system mount path and add it to the dfs.name.dir property in hdfs-default.xml.

   The following example shows the updated mount path values for dfs.datanode.data.dir

   ```
   <property>
     <name>dfs.datanode.data.dir</name>
     <value>file:///grid/3/aa/hdfs/data/,[RAM_DISK]file:///mnt/hdfsramdisk/</
   value>
   </property>
   ```

4. Set the LAZY_PERSIST storage policy to store data on the configured DataNode memory.
   The following example shows how you can use the `hdfs dfsadmin -getStoragepolicy` command to configure the LAZY_PERSIST storage policy:

   ```
   hdfs dfsadmin -getStoragePolicy /memory1 LAZY_PERSIST
   ```

   Note: When you update a storage policy setting on a file or directory, the new policy is *not* automatically enforced. You must use the HDFS mover data migration tool to actually move blocks as specified by the new storage policy.
5. Start the DataNode.
6. Use the HDFS mover tool to move data blocks according to the specified storage policy.

   The HDFS mover data migration tool scans the specified files in HDFS and verifies if the block placement satisfies the storage policy. For the blocks that violate the storage policy, the tool moves the replicas to a different storage type in order to fulfill the storage policy requirements.

Improving performance with short-circuit local reads

In HDFS, reads normally go through the DataNode. Thus, when a client asks the DataNode to read a file, the DataNode reads that file off of the disk and sends the data to the client over a TCP socket. "Short-circuit" reads bypass the DataNode, allowing the client to read the file directly. This is only possible in cases where the client is co-located with the data. Short-circuit reads provide a substantial performance boost to many applications.

Prerequisites for configuring short-ciruit local reads

To configure short-circuit local reads, you must enable libhadoop.so.

See the Native Libraries Guide for details on enabling this library.

Related Information

Native Libraries Guide

Properties for configuring short-circuit local reads on HDFS

To configure short-circuit local reads, you must add various properties to the hdfs-site.xml file. Short-circuit local reads must be configured on both the DataNode and the client.

Property Name	Property Value	Description
dfs.client.read.shortcircuit	true	Set this to true to enable short-circuit local reads.
dfs.domain.socket.path	/var/lib/hadoop-hdfs/ dn_socket	The path to the domain socket. Short-circuit reads make use of a UNIX domain socket. This is a special path in the file system that allows the client and the DataNodes to communicate. You will need to set a path to this socket. The DataNode needs to be able to create this path. On the other hand, it should not be possible for any user except the hdfs user or root to create this path. For this reason, paths under /var/run or /var/lib are often used. In the file system that allows the client and the DataNodes to communicate. You will need to set a path to this socket. The DataNode needs to be able to create this path. On the other hand, it should not be possible for any user except the hdfs user or root to create this path. For this reason, paths under /var/run or /var/lib are often used.

Property Name	Property Value	Description
dfs.client.domain.socket.data.traffic	false	This property controls whether or not normal data traffic will be passed through the UNIX domain socket. It is recommended that you set the value of this property to false. Unusual data traffic will be passed through the UNIX domain socket.
dfs.client.use.legacy.blockreader.local	false	Setting this value to false specifies that the new version (based on HDFS-347) of the short-circuit reader is used. Setting this value to true would mean that the legacy short-circuit reader would be used.
dfs.datanode.hdfs-blocks-metadata.enabled	true	Boolean which enables back-end DataNode-side support for the experimental DistributedFileSystem#getFile VBlockStorageLocationsAPI.
dfs.client.file-block-storage-locations.timeout	60	Timeout (in seconds) for the parallel RPCs made in DistributedFileSystem #getFileBlockStorageLocations(). This property is deprecated but is still supported for backward compatibility
dfs.client.file-block-storage-locations.timeout.millis	60000	Timeout (in milliseconds) for the parallel RPCs made in DistributedFileSystem #getFileBlockStorageLocations(). This property replaces dfs.client.file-block-storage-locations.timeout, and offers a finer level of granularity.
dfs.client.read.shortcircuit.skip.checksum	false	If this configuration parameter is set, short-circuit local reads will skip checksums. This is normally not recommended, but it may be useful for special setups. You might consider using this if you are doing your own checksumming outside of HDFS.
dfs.client.read.shortcircuit.streams.cache.size	256	The DFSClient maintains a cache of recently opened file descriptors. This parameter controls the size of that cache. Setting this higher will use more file descriptors, but potentially provide better performance on workloads involving many seeks.
dfs.client.read.shortcircuit.streams.cache.expiry.ms	300000	This controls the minimum amount of time (in milliseconds) file descriptors need to sit in the client cache context before they can be closed for being inactive for too long.

The XML for these entries:

```
<configuration>
<property>
  <name>dfs.client.read.shortcircuit</name>
  <value>true</value>
```

```
</property>

<property>
  <name>dfs.domain.socket.path</name>
  <value>/var/lib/hadoop-hdfs/dn_socket</value>
</property>

<property>
  <name>dfs.client.domain.socket.data.traffic</name>
  <value>false</value>
</property>

<property>
  <name>dfs.client.use.legacy.blockreader.local</name>
  <value>false</value>
</property>

<property>
  <name>dfs.datanode.hdfs-blocks-metadata.enabled</name>
  <value>true</value>
</property>

<property>
  <name>dfs.client.file-block-storage-locations.timeout.millis</name>
  <value>60000</value>
</property>

<property>
  <name>dfs.client.read.shortcircuit.skip.checksum</name>
  <value>false</value>
</property>

<property>
  <name>dfs.client.read.shortcircuit.streams.cache.size</name>
  <value>256</value>
</property>

<property>
  <name>dfs.client.read.shortcircuit.streams.cache.expiry.ms</name>
  <value>300000</value>
</property>

</configuration>
```

Configure mountable HDFS

Cloudera includes a FUSE (Filesystem in UserSpace) interface into HDFS. The hadoop-hdfs-fuse command enables you to use your HDFS cluster as if it were a traditional filesystem on Linux.

Before you begin

You must have a working HDFS cluster and know the hostname and port that your NameNode exposes.

Procedure

1. Set up and test your mount point depending on whether you have an HA or a non-HA configuration.
   - To set up and test your mount point in a non-HA installation:

     ```
     $ mkdir -p <mount_point>
     ```

```
$ hadoop-fuse-dfs dfs://<name_node_hostname>:<namenode_port> <mount_poi
nt>
```

where *NAMENODE_PORT* is the NameNode's RPC port, dfs.namenode.servicerpc-address.

- To set up and test your mount point in an HA installation:

```
mkdir -p <mount_point>
hadoop-fuse-dfs dfs://<nameservice_id> <mount_point>
```

- where *NAMESERVICE_ID* is the value of fs.defaultFS. In this case the port defined for dfs.namenode.rpc-address.[nameservice ID].[name node ID] is used automatically.

You can now run operations as if they are on your mount point.

Note:

To find its configuration directory, hadoop-fuse-dfs uses the HADOOP_CONF_DIR configured at the time the mount command is invoked.

2. Clean up your test mount by using the umount command.

```
umount <mount_point>
```

What to do next

You can now add a permanent HDFS mount which persists through reboots.

Add HDFS system mount

After installing hadoop-hdfs-fuse and configuring the test mount, you can add a persistent HDFS system mount.

About this task

Procedure

1. Open /etc/fstab and append information about the hostname, the NameNode port, and the mount point in the specified format.

   The format is as follows:

```
hadoop-fuse-dfs#dfs://<name_node_hostname>:<namenode_port> <mount_point>
 fuse allow_other,usetrash,rw 2 0
```

   For example:

```
hadoop-fuse-dfs#dfs://localhost:8020 /mnt/hdfs fuse allow_other,usetrash
,rw 2 0
```

   Note:

   In an HA deployment, use the HDFS nameservice instead of the NameNode URI; that is, use the value of dfs.nameservices in hdfs-site.xml.

2. Test to make sure the mount point is working properly

```
mount <mount_point>
```

Results

Your system is now configured to allow you to use the ls command and use that mount point as if it were a normal system disk.

For more information, see the help for hadoop-fuse-dfs:

```
hadoop-fuse-dfs --help
```

Optimize mountable HDFS

You can consider certain recommendations to optimize the mountable HDFS.

- Cloudera recommends that you use the -obig_writes option on kernels later than 2.6.26. This option allows for better performance of writes.
- By default, the package installation creates the /etc/default/hadoop-fuse file with a maximum heap size of 128 MB. You might need to change the JVM minimum and maximum heap size for better performance. For example:

  ```
  export LIBHDFS_OPTS="-Xms64m -Xmx256m"
  ```

  Be careful not to set the minimum to a higher value than the maximum.

Configuring Proxy Users to Access HDFS

Hadoop allows you to configure proxy users to submit jobs or access HDFS on behalf of other users; this is called *impersonation*. When you enable impersonation, any jobs submitted using a proxy are executed with the impersonated user's existing privilege levels rather than those of a superuser (such as hdfs).

All proxy users are configured in one location, core-site.xml, for Hadoop administrators to implement centralized access control.

To configure proxy users, set the hadoop.proxyuser.<proxy_user>.hosts, hadoop.proxyuser.<proxy_group>.groups, and hadoop.proxyuser.<proxy_user>.users in core-site.xml properties.

For example, to allow user alice to impersonate a user belonging to group_a and group_b, set hadoop.proxyuser.<proxy_group>.groups as follows:

```
  <property>
    <name>hadoop.proxyuser.alice.groups</name>
    <value>group_a,group_b</value>
  </property>
```

To limit the hosts from which impersonated connections are allowed, use hadoop.proxyuser.<proxy_user>.hosts. For example, to allow user alice impersonated connections only from host_a and host_b:

```
<property>
   <name>hadoop.proxyuser.alice.hosts</name>
   <value>host_a,host_b</value>
</property>
```

If the configuration properties described are not present, impersonation is not allowed and connections will fail.

For looser restrictions, use a wildcard (*) to allow impersonation from any host and of any user. For example, to allow user bob to impersonate any user belonging to any group, and from any host, set the properties as follows:

```
  <property>
    <name>hadoop.proxyuser.bob.hosts</name>
    <value>*</value>
  </property>
  <property>
    <name>hadoop.proxyuser.bob.groups</name>
    <value>*</value>
```

```
  </property>
```

The hadoop.proxyuser.<proxy_user>.hosts property also accepts comma-separated lists of IP addresses, IP address ranges in CIDR format, or host names. For example, to allow user kate access from hosts in the range 10.222.0.0-15 and 10.113.221.221, to impersonate user_a and user_b, set the proxy user properties as follows:

```
<property>
     <name>hadoop.proxyuser.kate.hosts</name>
     <value>10.222.0.0/16,10.113.221.221</value>
</property>
<property>
     <name>hadoop.proxyuser.kate.users</name>
     <value>user1,user2</value>
</property>
```

Note: If you require proxy user to be setup on HTTPFS, you also need to configure them through the safety valve for the httpfs-site.xml. The properties required are the same as for the general proxy user setup, only they are prefixed with httpfs instead of hadoop. For example,

```
  <property>
    <name>httpfs.proxyuser.#USER#.hosts</name>
    <value>*</value>
    <description>
      List of hosts the '#USER#' user is allowed to perform 'doAs'
      operations.
      The '#USER#' must be replaced with the username o the user who is
      allowed to perform 'doAs' operations.

      The value can be the '*' wildcard or a list of hostnames.

      For multiple users copy this property and replace the user name
      in the property name.
    </description>
  </property>

  <property>
    <name>httpfs.proxyuser.#USER#.groups</name>
    <value>*</value>
    <description>
      List of groups the '#USER#' user is allowed to impersonate users
      from to perform 'doAs' operations.

      The '#USER#' must be replaced with the username o the user who is
      allowed to perform 'doAs' operations.
      The value can be the '*' wildcard or a list of groups.

      For multiple users copy this property and replace the user name
      in the property name.
    </description>
  </property>
```

The content of the properties should be the same as for the hadoop equivalent. For HTTPFS, you need to go into HttpFS Advanced Configuration Snippet (Safety Valve) for httpfs-site.xml.

Using DistCp to copy files

Hadoop DistCp (distributed copy) can be used to copy data between Cloudera clusters (and also within a Cloudera cluster).

DistCp uses MapReduce to implement its distribution, error handling, and reporting. It expands a list of files and directories into map tasks, each of which copies a partition of the files specified in the source list.

Using DistCp

The distributed copy command, distcp, is a general utility for copying large data sets between distributed filesystems within and across clusters. You can also use distcp to copy data to and from an Amazon S3 bucket. The distcp command submits a regular MapReduce job that performs a file-by-file copy.

To see the distcp command options, run the built-in help:

```
hadoop distcp
```

Important:

- Do not run distcp as the hdfs user which is disallowed for MapReduce jobs by default.
- Do not use Hadoop shell commands (such as cp, copyfromlocal, put, get) for large copying jobs or you may experience I/O bottlenecks.

Distcp syntax and examples

You can use distcp for copying data between Cloudera clusters. In addition, you can also use it to copy data between a Cloudera cluster and Amazon S3 or Azure Data Lake Storage Gen 2.

Common use of distcp

The most common use of distcp is an inter-cluster copy:

```
hadoop distcp hdfs://nn1:8020/source hdfs://nn2:8020/destination
```

Where hdfs://nn1:8020/source is the data source, and hdfs://nn2:8020/destination is the destination. This will expand the name space under /source on NameNode "nn1" into a temporary file, partition its contents among a set of map tasks, and start copying from "nn1" to "nn2". Note that DistCp requires absolute paths.

You can also specify multiple source directories:

```
hadoop distcp hdfs://nn1:8020/source/a hdfs://nn1:8020/source/b hdfs://nn2:8
020/destination
```

Or specify multiple source directories from a file with the -f option:

```
hadoop distcp -f hdfs://nn1:8020/srclist hdfs://nn2:8020/destination
```

Where srclist contains:

```
hdfs://nn1:8020/source/a
hdfs://nn1:8020/source/b
```

Copying between major versions

Run the distcp command on the cluster that runs the higher version of Cloudera, which should be the destination cluster. Use the following syntax:

```
hadoop distcp webhdfs://<namenode>:<port> hdfs://<namenode>
```

Note the webhdfs prefix for the remote cluster, which should be your source cluster. You must use webhdfs when the clusters run different major versions. When clusters run the same version, you can use the hdfs protocol for better performance.

For example, the following command copies data from a Cloudera source cluster named example-source to another Cloudera version destination cluster named example-dest:

```
hadoop distcp webhdfs://example-source.cloudera.com:8020 hdfs://example-dest
.cloudera.com
```

Copying to/from Amazon S3

The following syntax for distcp shows how to copy data to/from S3:

```
#Copying from S3
hadoop distcp s3a://<bucket>/<data> hdfs://<namenode>/<directory>/
#Copying to S3
hadoop distcp hdfs://<namenode>/<directory> s3a://<bucket>/<data>
```

This is a basic example of using distcp with S3.

Copying to/from ADLS Gen 2

The following syntax for distcp shows how to copy data to/from ADLS Gen 2:

```
#Copying from ABFS
hadoop distcp abfs[s]://<file_system>@<account_name>.dfs.core.windows.net/<p
ath>/<file_name> hdfs://HDFS_DESTINATION_PATH
#Copying to ADLS Gen2
hadoop distcp hdfs://HDFS_DESTINATION_PATH abfs[s]://<file_system>@<accoun
t_name>.dfs.core.windows.net/<path>/<file_name>
```

Using DistCp with Highly Available remote clusters

You can use distcp to copy files between highly available clusters by configuring access to the remote cluster with the nameservice ID.

Procedure

1. Create a new directory and copy the contents of the /etc/hadoop/conf directory on the local cluster to this directory.

   The local cluster is the cluster where you plan to run the distcp command.

   Specify the new directory for the --config parameter when you run the distcp command in step 5.

   Note: The following steps use distcpConf as the directory name. Substitute the name of the directory you created for distcpConf.

2. In the hdfs-site.xml file in the distcpConf directory, add the nameservice ID for the remote cluster to the dfs.name services property.

   Note: If the remote cluster has the same nameservice ID as the local cluster, change the remote cluster's nameservice ID. Nameservice names must be unique.

   For example, if the nameservice name for both clusters is nameservice1, change the nameservice ID of the remote cluster to a different ID, such as externalnameservice:

```
<property>
 <name>dfs.nameservices</name>
 <value>nameservice1,externalnameservice</value>
 </property>
```

3. On the remote cluster, find the hdfs-site.xml file and copy the properties that refer to the nameservice ID to the end of the hdfs-site.xml file in the distcpConf directory you created in step 1.

   - dfs.ha.namenodes.<nameserviceID>
   - dfs.client.failover.proxy.provider.<remote nameserviceID>
   - dfs.ha.automatic-failover.enabled.<remote nameserviceID>
   - dfs.namenode.rpc-address.<nameserviceID>.<namenode1>
   - dfs.namenode.servicerpc-address.<nameserviceID>.<namenode1>
   - dfs.namenode.http-address.<nameserviceID>.<namenode1>
   - dfs.namenode.https-address.<nameserviceID>.<namenode1>
   - dfs.namenode.rpc-address.<nameserviceID>.<namenode2>
   - dfs.namenode.servicerpc-address.<nameserviceID>.<namenode2>
   - dfs.namenode.http-address.<nameserviceID>.<namenode2>
   - dfs.namenode.https-address.<nameserviceID>.<namenode2>

   By default, you can find the hdfs-site.xml file in the /etc/hadoop/conf directory on a node of the remote cluster.

4. If you changed the nameservice ID for the remote cluster in step 2, update the nameservice ID used in the properties you copied in step 3 with the new nameservice ID, accordingly.

   The following example shows the properties copied from the remote cluster with the following values:

   - A remote nameservice called externalnameservice
   - NameNodes called namenode1 and namenode2
   - A host named remotecluster.com

```
 <property>
<name>dfs.ha.namenodes.externalnameservice</name>
<value>namenode1,namenode2</value>
</property>

<property>
<name>dfs.client.failover.proxy.provider.externalnameservice</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProx
yProvider</value>
</property>

<property>
<name>dfs.ha.automatic-failover.enabled.externalnameservice</name>
<value>true</value>
</property>

<property>
<name>dfs.namenode.rpc-address.externalnameservice.namenode1</name>
<value>remotecluster.com:8020</value>
</property>

<property>
```

```
<name>dfs.namenode.servicerpc-address.externalnameservice.namenode1</name>
<value>remotecluster.com:8022</value>
</property>
<property>
<name>dfs.namenode.http-address.externalnameservice.namenode1</name>
<value>remotecluster.com:20101</value>
</property>
<property>
<name>dfs.namenode.https-address.externalnameservice.namenode1</name>
<value>remotecluster.com:20102</value>
</property>

<property>
<name>dfs.namenode.rpc-address.externalnameservice.namenode2</name>
<value>remotecluster.com:8020</value>
</property>

<property>
<name>dfs.namenode.servicerpc-address.externalnameservice.namenode2</name>
<value>remotecluster.com:8022</value>
</property>
<property>
<name>dfs.namenode.http-address.externalnameservice.namenode2</name>
<value>remotecluster.com:20101</value>
</property>
<property>
<name>dfs.namenode.https-address.externalnameservice.namenode2</name>
<value>remotecluster.com:20102</value>
</property>
```

At this point, the hdfs-site.xml file in the distcpConf directory should have both the clusters and four NameNode IDs.

5. Depending on the use case, the options specified when you run the distcp may differ.

Note: The remote cluster can be either the source or the target. The examples provided specify the remote cluster as the source.

To copy data from an insecure cluster, run the following command:

```
hadoop --config distcpConf distcp hdfs://<nameservice>/<source_directory>
 <target directory>
```

To copy data from a secure cluster, run the following command:

```
hadoop --config distcpConf distcp -Dmapreduce.job.hdfs-servers.token-ren
ewal.exclude=<nameservice>  hdfs://<nameservice>/<source_directory> <tar
get directory>
```

For example:

```
hadoop --config distcpConf distcp -Dmapreduce.job.hdfs-servers.token-ren
ewal.exclude=ns1 hdfs://ns1/xyz /tmp/test
```

If the distcp source or target are in encryption zones, include the following distcp options: -skipcrccheck -update. The distcp command may fail if you do not include these options when the source or target are in encryption zones because the CRC for the files may differ.

For distcp between clusters that both use HDFS Transparent Encryption, you must include the exclude parameter.

Using DistCp with Amazon S3

You can copy HDFS files to and from an Amazon S3 instance. You must provision an S3 bucket using Amazon Web Services and obtain the access key and secret key.

You can pass these credentials on the distcp command line, or you can reference a credential store to "hide" sensitive credentials so that they do not appear in the console output, configuration files, or log files.

Amazon S3 block and native filesystems are supported with the s3a:// protocol.

Example of an Amazon S3 Block Filesystem URI: s3a://*BUCKET_NAME*/path/to/file

S3 credentials can be provided in a configuration file (for example, core-site.xml):

```
<property>
    <name>fs.s3a.access.key</name>
    <value>...</value>
</property>
<property>
    <name>fs.s3a.secret.key</name>
    <value>...</value>
</property>
```

You can also enter the configurations in the Advanced Configuration Snippet for core-site.xml, which allows Cloudera Manager to manage this configuration.

You can also provide the credentials on the command line:

```
hadoop distcp -Dfs.s3a.access.key=... -Dfs.s3a.secret.key=... s3a://
```

For example:

```
hadoop distcp -Dfs.s3a.access.key=myAccessKey -Dfs.s3a.secret.key=mySecretKe
y /user/hdfs/mydata s3a://myBucket/mydata_backup
```

Important: Entering secrets on the command line is inherently insecure. These secrets may be accessed in log files and other artifacts. Cloudera recommends that you use a credential provider to store secrets.

Note: Using the -diff option with the distcp command requires a Distributed FileSystem on both the source and destination and is not supported when using distcp to copy data to or from Amazon S3.

Using a credential provider to secure S3 credentials

You can run the distcp command without having to enter the access key and secret key on the command line. This prevents these credentials from being exposed in console output, log files, configuration files, and other artifacts.

About this task

Running the distcp command in this way requires that you provision a credential store to securely store the access key and secret key. The credential store file is saved in HDFS.

Note: Using a Credential Provider does not work with MapReduce v1 (MRV1).

Procedure

1. Provision the credentials by running the following commands:

```
hadoop credential create fs.s3a.access.key -value ACCESS_KEY -provider j
ceks://hdfs/PATH_TO_CREDENTIAL_STORE_FILE
```

```
hadoop credential create fs.s3a.secret.key -value SECRET_KEY -provider j
ceks://hdfs/PATH_TO_CREDENTIAL_STORE_FILE
```

For example:

```
hadoop credential create fs.s3a.access.key -value foobar -provider jceks
://hdfs/user/alice/home/keystores/aws.jceks
hadoop credential create fs.s3a.secret.key -value barfoo -provider jceks
://hdfs/user/alice/home/keystores/aws.jceks
```

You can omit the -value option and its value and the command will prompt the user to enter the value.

For more details on the hadoop credential command, see Credential Management (Apache Software Foundation).

Note: If you configure the hadoop.security.credential.provider.path property to jceks://hdfs/path_to_cre dential_store_file, ZKFC might fail to start and NameNodes might remain in standby state. So, Cloudera recommends not to add or change this property in Cloudera Manager.

2. Copy the contents of the /etc/hadoop/conf directory to a working directory.
3. Add the following to the core-site.xml file in the working directory:

```
<property>
<name>hadoop.security.credential.provider.path</name>
<value>jceks://hdfs/PATH_TO_CREDENTIAL_STORE_FILE</value>
</property>
```

4. Set the HADOOP_CONF_DIR environment variable to the location of the working directory:

```
export HADOOP_CONF_DIR=PATH_TO_WORKING_DIRECTORY
```

What to do next

After completing these steps, you can run the distcp command using the following syntax:

```
hadoop distcp SOURCE_PATH s3a://DESTINATION_PATH
```

You can also reference the credential store on the command line, without having to enter it in a copy of the core-sit e.xml file. You also do not need to set a value for HADOOP_CONF_DIR. Use the following syntax:

```
hadoop distcp SOURCE_PATH s3a://BUCKET_NAME/DESTINATION_PATH
-Dhadoop.security.credential.provider.path=jceks://
hdfsPATH_TO_CREDENTIAL_STORE_FILE
```

Examples of DistCp commands using the S3 protocol and hidden credentials

You can various distcp command options to copy files between your Cloudera clusters and Amazon S3.

Copying files to Amazon S3

```
hadoop distcp /user/hdfs/mydata s3a://myBucket/mydata_backup
```

Copying files from Amazon S3

```
hadoop distcp s3a://myBucket/mydata_backup //user/hdfs/mydata
```

Copying files to Amazon S3 using the -filters option to exclude specified source files

You specify a file name with the -filters option. The referenced file contains regular expressions, one per line, that define file name patterns to exclude from the distcp job. The pattern specified in the regular expression should match the fully-qualified path of the intended files, including the

scheme (hdfs, webhdfs, s3a, etc.). For example, the following are valid expressions for excluding files:

```
hdfs://x.y.z:8020/a/b/c
webhdfs://x.y.z:50070/a/b/c
s3a://bucket/a/b/c
```

Reference the file containing the filter expressions using -filters option. For example:

```
hadoop distcp -filters /user/joe/myFilters /user/hdfs/mydata s
3a://myBucket/mydata_backup
```

Contents of the sample myFilters file:

```
.*foo.*
.*/bar/.*
hdfs://x.y.z:8020/tmp/.*
hdfs://x.y.z:8020/tmp1/file1
```

The regular expressions in the myFilters exclude the following files:

- .*foo.* – excludes paths that contain the string "foo".
- .*/bar/.* – excludes paths that include a directory named bar.
- hdfs://x.y.z:8020/tmp/.* – excludes all files in the /tmp directory.
- hdfs://x.y.z:8020/tmp1/file1 – excludes the file /tmp1/file1.

Copying files to Amazon S3 with the -overwrite option.

The -overwrite option overwrites destination files that already exist.

```
hadoop distcp -overwrite /user/hdfs/mydata  s3a://user/mydata_ba
ckup
```

For more information about the -filters, -overwrite, and other options, see DistCp Guide: Command Line Options (Apache Software Foundation).

Kerberos setup guidelines for Distcp between secure clusters

There are specific guidelines to consider while setting up Kerberos on secure Cloudera clusters for successfully performing distcp between them.

The guidelines mentioned here are only applicable for the following example deployment:

- You have two clusters, each in a different Kerberos realm (SOURCE and DESTINATION in this example)
- You have data that needs to be copied from SOURCE to DESTINATION
- A Kerberos realm trust exists, either between SOURCE and DESTINATION (in either direction), or between both SOURCE and DESTINATION and a common third realm (such as an Active Directory domain).

If your environment matches the one described above, use the following table to configure Kerberos delegation tokens on your cluster so that you can successfully distcp across two secure clusters. Based on the direction of the trust between the SOURCE and DESTINATION clusters, you can use the mapreduce.job.hdfs-servers.token-renewal .exclude property to instruct ResourceManagers on either cluster to skip or perform delegation token renewal for NameNode hosts.

Note: You must use the mapreduce.job.hdfs-servers.token-renewal.exclude parameter if both clusters use the HDFS Transparent Encryption feature.

Environment Type		Kerberos Delegation Token Setting
SOURCE trusts DESTINATION	Distcp job runs on the DESTINATION cluster	You do not need to set the mapreduce.job.hdfs-servers.token-renewal.exclude property.
	Distcp job runs on the SOURCE cluster	Set the mapreduce.job.hdfs-servers.token-renewal.exclude property to a comma-separated list of the hostnames of the NameNodes of the DESTINATION cluster.
DESTINATION trusts SOURCE	Distcp job runs on the DESTINATION cluster	Set the mapreduce.job.hdfs-servers.token-renewal.exclude property to a comma-separated list of the hostnames of the NameNodes of the SOURCE cluster.
	Distcp job runs on the SOURCE cluster	You do not need to set the mapreduce.job.hdfs-servers.token-renewal.exclude property.
Both SOURCE and DESTINATION trust each other	Set the mapreduce.job.hdfs-servers.token-renewal.exclude property to a comma-separated list of the hostnames of the NameNodes of the DESTINATION cluster.	
Neither SOURCE nor DESTINATION trusts the other	If a common realm is usable (such as Active Directory), set the mapreduce.job.hdfs-servers.token-renewal.exclude property to a comma-separated list of hostnames of the NameNodes of the cluster that is not running the distcp job. For example, if you are running the job on the DESTINATION cluster: **1.** kinit on any DESTINATION YARN Gateway host using an AD account that can be used on both SOURCE and DESTINATION. **2.** Run the distcp job as the hadoop user: `$ hadoop distcp -Ddfs.namenode.kerberos.principal.pattern=* \ -Dmapreduce.job.hdfs-servers.token-renewal.exclude=SOURCE-nn-host1,SOURCE-nn-host2 \ hdfs://source-nn-nameservice/source/path \ /destination/path` By default, the YARN ResourceManager renews tokens for applications. The mapreduce.job.hdfs-servers.token-renewal.exclude property instructs ResourceManagers on either cluster to skip delegation token renewal for NameNode hosts.	

Distcp between secure clusters in different Kerberos realms

To use distcp between two secure clusters in different Kerberos realms, you must use a single Kerberos principal that can authenticate to both realms. In other words, a Kerberos realm trust relationship must exist between the source and destination realms. This can be a one-way trust (in either direction), a bi-directional trust, or even multiple one-way trusts where both the source and destination realms trust a third realm (such as an Active Directory domain).

Before you begin

Ensure that both the secure clusters run a supported version of JDK. For more information about the supported JDK versions, see Java Requirements.

About this task

If there is no trust relationship between the source and destination realms, you cannot use distcp to copy data between the clusters, but you can use Replication Manager. For details, see Enabling Replication Between Clusters with Kerberos Authentication.

Configure source and destination realms in krb5.conf

You must ensure that the krb5.conf file on the client from which the distcp job is submitted includes realm definitions and mappings for both source and destination realms.

Procedure

Include the realm definitions and mappings for both source and destination realms in the krb5.conf file, as specified in the following example.

```
[realms]
 QA.EXAMPLE.COM = {
  kdc = kdc01.qa.example.com:88
  admin_server = kdc01.qa.example.com:749
 }

 DEV.EXAMPLE.COM = {
  kdc = kdc01.dev.example.com:88
  admin_server = kdc01.dev.example.com:749
 }

[domain_realm]
.qa.example.com = QA.EXAMPLE.COM
qa.example.com = QA.EXAMPLE.COM
.dev.example.com = DEV.EXAMPLE.COM
dev.example.com = DEV.EXAMPLE.COM
```

Configure HDFS RPC protection

Set the hadoop.rpc.protection property to authentication in both the clusters between which you want to run distcp. You can modify the property either in hdfs-site.xml, or using Cloudera Manager.

About this task

This procedure explains how you can configure the hadoop.rpc.protection property using Cloudera Manager. You must perform the procedure on both the clusters.

Procedure

1. Open the Cloudera Admin Console.
2. Go to the HDFS service.
3. Click the Configuration tab.
4. Select Scope > HDFS-1 (Service-Wide).
5. Select Category > Security.
6. Locate the Hadoop RPC Protection property and select authentication.
7. Enter a Reason for Change and then click Save Changes.

Specify truststore properties

If TLS/SSL is enabled, then you must configure specific properties in the ssl-client.xml file on the client submitting the distcp job to establish trust between the target and destination clusters.

Procedure

Configure the ssl.client.truststore.location, ssl.client.truststore.password, and ssl.client.truststore.password properties in ssl-client.xml as specified.

```
<property>
<name>ssl.client.truststore.location</name>
<value>path_to_truststore</value>
</property>

<property>
<name>ssl.client.truststore.password</name>
<value>XXXXXX</value>
```

```
</property>

<property>
<name>ssl.client.truststore.type</name>
<value>jks</value>
</property>
```

Set HADOOP_CONF to the destination cluster

Set the HADOOP_CONF path to be the destination environment. If you are not using HFTP, set the HADOOP_CONF path to the source environment instead.

Alternatively, you can point the hadoop distcp client to a configuration file that includes the parameters that allow it to point to the destination cluster. This can be done on the command line with the --config option. That option must be pointed to a new copy of the contents of the /etc/hadoop/conf directory with the following parameters from the destination /remote cluster added to the hdfs-site.xml file:

```
dfs.ha.namenodes.<nameserviceID>
dfs.client.failover.proxy.provider.<remote nameserviceID>
dfs.ha.automatic-failover.enabled.<remote nameserviceID>
dfs.namenode.rpc-address.<nameserviceID>.<namenode1>
dfs.namenode.servicerpc-address.<nameserviceID>.<namenode1>
dfs.namenode.http-address.<nameserviceID>.<namenode1>
dfs.namenode.https-address.<nameserviceID>.<namenode1>
dfs.namenode.rpc-address.<nameserviceID>.<namenode2>
dfs.namenode.servicerpc-address.<nameserviceID>.<namenode2>
dfs.namenode.http-address.<nameserviceID>.<namenode2>
dfs.namenode.https-address.<nameserviceID>.<namenode2>
```

These parameters can be found in the /etc/hadoop/conf/hdfs-site.xml path on the remote cluster.

For example, the command would look something like the following:

```
hadoop --config distcpConf distcp -Dmapreduce.job.hdfs-servers.token-renewal
.exclude=<nameservice>  hdfs://<nameservice>/<source_directory> <target dire
ctory>
```

For more details, see Using DistCp with Highly Available remote clusters.

Launch distcp

Authenticate using kinit on the client and then launch the distcp job.

Consider the following example:

```
hadoop distcp hdfs://xyz01.dev.example.com:8020/user/alice hdfs://abc01.qa.e
xample.com:8020/user/alice
```

If launching distcp fails, force Kerberos to use TCP instead of UDP by adding the following parameter to the krb5.conf file on the client:

```
[libdefaults]
udp_preference_limit = 1
```

Copying data between a secure and an insecure cluster using DistCp and WebHDFS

You can use distcp and WebHDFS to copy data between a secure cluster and an insecure cluster.

About this task

When copying data, ensure that you run the distcp commands from the secure cluster.

Procedure

1. On the secure cluster, set ipc.client.fallback-to-simple-auth-allowed to true in core-site.xml.

```
<property>
  <name>ipc.client.fallback-to-simple-auth-allowed</name>
  <value>true</value>
</property>
```

   Alternatively, you can also pass this as a parameter when you run the distcp command. If you want to do that, move onto step 2.

2. On the insecure cluster, add the secured cluster's realm name to the insecure cluster's configuration.
   a) In the Cloudera Admin Console for the insecure cluster, navigate to Clusters <HDFS cluster> .
   b) On the Configuration tab, search for Trusted Kerberos Realms and add the secured cluster's realm name.

   Note: This does not require Kerberos to be enabled but is a necessary step to allow the simple auth fallback to happen in the hdfs:// protocol.

   c) Save the change.

3. Use commands such as the following only from the secure cluster side.

```
#This example uses the insecure cluster as the source and the secure clu
ster as the destination
distcp webhdfs://<INSECURE_NAMENODE>:9870 webhdfs://<SECURE_NAMENODE>:9871

#This example uses the sefcure cluster as the source and the insecure cl
uster as the destination
distcp webhdfs://<SECURE_NAMENODE>:9871 webhdfs://<INSECURE_NAMENODE>:9870
```

   If TLS is enabled, replace webhdfs with swebhdfs.

   If you did not configure ipc.client.fallback-to-simple-auth-allowed and want to pass it as a parameter, run commands such as the following from the secure cluster:

```
#This example uses the insecure cluster as the source and the secure clu
ster (with TLS enabled) as the destination cluster. swebhdfs is used ins
tead of webhdfs when TLS is enabled.
hadoop distcp -D ipc.client.fallback-to-simple-auth-allowed=true webhdf
s://<INSECURE_NAMENODE>:9870 swebhdfs://<SECURE_NAMENODE>:9871

#This example uses the secure cluster (with TLS enabled) as the source cl
uster and the insecure cluster as the destination. swebhdfs is used inst
ead of webhdfs when TLS is enabled.
hadoop distcp -D ipc.client.fallback-to-simple-auth-allowed=true swebhdf
s://<SECURE_NAMENODE>:9871 webhdfs://<INSECURE_NAMENODE>:9870
```

Post-migration verification

After migrating data between the two clusters, you can use hadoop fs -ls /basePath to verify the permissions, ownership and other aspects of your files, and correct any problems before using the files in your new cluster.

Using DistCp between HA clusters using Cloudera Manager

To copy data between HA clusters using distcp, you must configure specific name service properties to ensure that the HDFS clients in a cluster can access a remote HA cluster. You can configure these values through advanced configuration snippets in Cloudera Manager.

About this task

This procedure explains how you can configure the name service properties from Cloudera Manager to enable copying of data between two example clusters A and B. Here, A is the source cluster while B is the remote cluster.

Procedure

1. Select Clusters and choose the source HDFS cluster where you want to configure the properties.
2. Navigate to the Configuration tab and search for HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml.

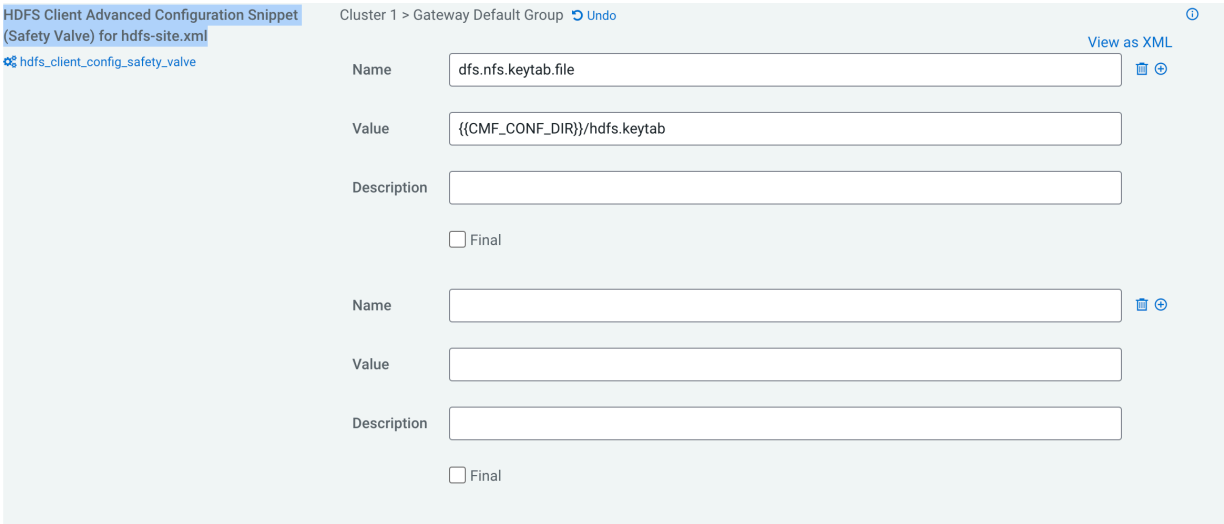

3. Add the various properties as specified:
   a) Add the dfs.internal.nameservices property as the name service of the source cluster.

      dfs.internal.nameservices = HAA
   b) Add the dfs.nameservices property as the name services of both the clusters.

      dfs.nameservices = HAA,HAB
   c) Add the dfs.client.failover.proxy.provider.<nameservice> property for the remote cluster.

      dfs.client.failover.proxy.provider.HAB = org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider
   d) Add the dfs.ha.namenodes.<nameservice> property to reference the NameNodes of the remote cluster.

      dfs.ha.namenodes.HAB = nn1,nn2
   e) Add the dfs.namenode.rpc-address and the dfs.namenode.servicerpc-address properties as the fully qualified RPC addresses on which each NameNode of the remote cluster can listen.

      dfs.namenode.rpc-address.HAB.nn1 = <NN1_fqdn>:8020

      dfs.namenode.rpc-address.HAB.nn2 = <NN2_fqdn>:8020

      dfs.namenode.servicerpc-address.HAB.nn1 = <NN1_fqdn>:8022

      dfs.namenode.servicerpc-address.HAB.nn2 = <NN2_fqdn>:8022
   f) Add the dfs.namenode.http-address and the dfs.namenode.https-address properties as the fully qualified HTTP and HTTPS addresses on which each NameNode of the remote cluster can listen.

      dfs.namenode.http-address.HAB.nn1 = <NN1_fqdn>:9870

      dfs.namenode.http-address.HAB.nn2 = <NN2_fqdn>:9870

      dfs.namenode.https-address.HAB.nn1 = <NN1_fqdn>:9871

      dfs.namenode.https-address.HAB.nn2 = <NN2_fqdn>:9871
4. Specify the Reason for Change, and click Save Changes.

   This saves the property values as client-side configurations.
5. Restart the HDFS service, then run the distcp command using the NameService. For example:

   hadoop distcp hdfs://HAA/tmp/testDistcp hdfs://HAB/tmp/

   Note: When you try to connect a NameNode from one cluster to another cluster which resides in the same REALM and same KDC, you must add the above HDFS HA configurations on the client side to connect to the remote cluster.

Using the NFS Gateway for accessing HDFS

Available in a Cloudera Private Cloud Base deployment, the NFS Gateway for HDFS allows clients to mount HDFS and interact with it through NFS, as if it were part of their local file system. The Gateway supports NFSv3.

After mounting HDFS, a client user can perform the following tasks:

- Browse the HDFS file system through their local file system on NFSv3 client-compatible operating systems.
- Upload and download files between the HDFS file system and their local file system.
- Stream data directly to HDFS through the mount point. File append is supported, but random write is not supported.

Install the NFS Gateway

The NFS Gateway is part of HDFS service. The NFS Gateway can be installed on any host managed by Cloudera Manager, including DataNode, NameNode, or Cloudera client machine. Start the NFS server on that machine.

About this task

Perform the following steps to install NFS Gateway role:

Procedure

1. In the Cloudera Admin Console, go to Clusters HDFS service .
2. Navigate to the Instances tab and click Add Role Instances.
3. Click NFS Gateway to choose the hosts to install the NFS Gateway role.
   The NFS Gateway is added to the host.

Configure the NFS Gateway

You must ensure that the proxy user for the NFS Gateway can proxy all the users accessing the NFS mounts. In addition, you must configure settings specific to the Gateway.

Procedure

1. Ensure that the proxy user for the NFS Gateway can proxy all the users accessing the NFS mounts.

   In non-secure mode, the user running the Gateway is the proxy user, while in secure mode the user in Kerberos keytab is the proxy user.

   If a user nfsserver is running the Gateway and there are users belonging to groups nfs-users1 and nfs-users2, then set the following values in core-site.xml on the NameNode.

   Note: Replace nfsserver with the user account that starts the Gateway in your cluster.

   a) Set the value of the hadoop.proxyuser.nfsserver.groups property in NFS Gateway Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml.

      For example, nfs-users1,nfs-users2. The nfsserver user is allowed to proxy all members of the nfs-users1 and nfs-users2 groups. Set this to * to allow nfsserver user to proxy any group.

   b) Set the value of the hadoop.proxyuser.nfsserver.hosts property in NFS Gateway Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml.

      For example, nfs-client-host1.com is the host where the NFS Gateway is running. Set this to * to allow requests from any hosts to be proxied.

2. Configure settings for the NFS Gateway.

   The NFS Gateway uses the same settings that are used by the NameNode and DataNode. Configure various properties based on your application's requirements:

   a) In the Cloudera Admin Console, go to Clusters HDFS service .
   b) Navigate to the Configurations tab and search for the Access Time Precision property.
   c) Specify the access time precision. For example, 1 hour.

      Note: If the export is mounted with access time update allowed, ensure that this property is not disabled in the configuration file. Only the NameNode needs to restart after this property is changed. If you have disabled access time update by mounting with noatime, you do NOT have to change this property nor restart your NameNode.

   d) Search for the Temporary Dump Directory property, and set the value as /tmp/.hdfs-nfs.

      Note: The NFS client often reorders writes. Sequential writes can arrive at the NFS Gateway in a random order. This directory is used to temporarily save out-of-order writes before writing to HDFS. Ensure that the directory has enough space. For example, if the application uploads 10 files with each having 100 MB, it is recommended for this directory to have 1GB space in case a write reorder happens to every file.

   e) Search for the Allowed Hosts and Privileges property, and set the value as * rw.

      Note: By default, the export can be mounted by any client. You must update this property to control access. The value string contains the machine name and access privilege, separated by whitespace characters. The machine name can be in single host, wildcard, or IPv4 network format. The access privilege uses rw or ro to specify readwrite or readonly access to exports. If you do not specify an access privilege, the default machine access to exports is readonly. Separate machine entries by ;. For example, 192.168.0.0/22 rw ; host*.example.com ; host1.test.org ro;.

   f) Restart the NFS Gateway.
   g) Customize log settings by modifying the log4j.property file.

      - To change the trace level, go to Configuration NFS Gateway Logging Advanced Configuration Snippet (Safety Valve) , and add log4j.logger.org.apache.hadoop.hdfs.nfs=DEBUG
      - To view more information about ONCRPC requests, go to Configuration NFS Gateway Logging Advanced Configuration Snippet (Safety Valve) , and add log4j.logger.org.apache.hadoop.oncrpc=DEBUG

3. Specify JVM heap space (HADOOP_NFS3_OPTS) for the NFS Gateway.

   You can increase the JVM heap allocation for the NFS Gateway using this option. To set this option, go to Configuration Java Heap Size of NFS Gateway in Bytes , and enter the value.

4. To improve the performance of large file transfers, you can increase the values of the dfs.nfs.rtmax and dfs.nfs.wtmax properties.

   These properties are configuration settings on the NFS Gateway server that change the maximum read and write request size supported by the Gateway. The default value for both settings is 1MB.

Start and stop the NFS Gateway services

You must start the following daemons to run the NFS services on the Gateway: rpcbind (or portmap), mountd, and nfsd. The NFS Gateway process includes both nfsd and mountd. Although NFS Gateway works with portmap included with most Linux distributions, you must use the portmap included in the NFS Gateway package on some Linux systems such as SLES 11 and RHEL 6.2.

About this task

Perform the following steps to use the bundled portmap:

Procedure

1. Stop the nfs/rpcbind/portmap services provided by the platform.

```
service nfs stop
service rpcbind stop
```

2. Start the included portmap service using one of the following commands:

```
hdfs portmap
OR
hdfs --daemon start portmap
```

Note: You must have root privileges to run these commands.

Start the NFS Gateway services

Learn how to start the NFS Gateway services.

About this task

Perform the following steps to start the NFS Gateway:

Procedure

1. In the Cloudera Admin Console, go to Clusters HDFS service .
2. Navigate to the Instances tab and search for the NFS Gateway.
3. Select Start from the Actions for Selected dropdown menu.

Note: The NFS Gateway log file directory is configured by the property NFS Gateway Log Directory. For example, if NFS Gateway Log Directory is configured as /var/log/hadoop-hdfs and the NFS Gateway runs on host host1.example.com, the log file would be /var/log/hadoop-hdfs/hadoop-hdfs-hdfs-nfs3-host1 .example.com.out.

What to do next

Verify the validity of the NFS services.

Stop the NFS Gateway services

Learn how to stop the NFS Gateway services. Before stopping the NFS Gateway services, all NFS mount points must be unmounted.

Procedure

1. Bring down all NFS mount points.

```
sudo umount {mountPoint}
```

2. Perform the following steps to stop the NFS Gateway services:
   a) In the Cloudera Admin Console, go to Clusters HDFS service .
   b) Navigate to the Instances tab and search for NFS Gateway.
   c) Select Stop from the drop down menu.

Verify validity of the NFS services

After starting the NFS services on the Gateway, you must verify whether all the services are running. Additionally, you must ensure that the HDFS namespace is exported and can be mounted.

Procedure

1. Run the `rpcinfo` command to verify whether all the NFS services are running.

   rpcinfo -p $nfs_server_ip

   The command returns an output similar to the following:

```
    program vers proto   port
    100005    1   tcp   4242  mountd
    100005    2   udp   4242  mountd
    100005    2   tcp   4242  mountd
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100005    3   udp   4242  mountd
    100005    1   udp   4242  mountd
    100003    3   tcp   2049  nfs
    100005    3   tcp   4242  mountd
```

2. Verify that the HDFS namespace is exported and can be mounted.

   showmount -e $nfs_server_ip

   The command returns an exports list similar to the following:

   Exports list on $nfs_server_ip : / (everyone)

Access HDFS from the NFS Gateway

To access HDFS, you must first mount the namespace and then set up the NFS client hosts to interact with HDFS through the Gateway.

Procedure

1. Mount the HDFS namespace.

   mount -t nfs -o vers=3,proto=tcp,nolock,sync,rsize=1048576,wsize=1048576 $server:/ $mount_point

   Access HDFS as part of the local file system, except that hard link or symbolic link and random write are not supported.

   Note: Because NLM is not supported, the mount option nolock is required.

   You can use the sync option for improving the performance and reliability when writing large files. If the sync option is specified, the NFS client flushes write operations to the NFS Gateway before returning control to the client application. Additionally, the client does not issue reordered writes. This reduces buffering requirements on the NFS gateway.

2. Set up the NFS client users to interact with HDFS through the NFS Gateway.

Related Information

How NFS Gateway authenticates and maps users

How NFS Gateway authenticates and maps users

The manner in which NFS Gateway authenticates and maps users determines how NFS client users access HDFS through the NFS Gateway.

NFS Gateway uses AUTH_UNIX-style authentication, which means that the user logged on the client is the same that NFS passes to HDFS.

For example, suppose that the NFS client has current user as admin. When the user accesses the mounted HDFS directory, NFS gateway accesses HDFS as the user admin. To access HDFS as hdfs user, you must first switch the current user to hdfs on the client system before accessing the mounted directory.

For the NFS client users accessing HDFS, the Gateway converts the User Identifier (UID) to a username, which is used by HDFS to check permissions. Therefore, you must ensure that the user on the NFS client host has the same name and UID as that on the NFS Gateway host.

Note: If you use the same user management system such as LDAP/NIS to create and deploy users to the Cloudera nodes and to the client host, then the user names and UIDs are usually identical on both the client and the Gateway.

If the user is created manually on different hosts, you might need to modify the UID, as specified in the following example, on either the client or the NFS Gateway to make them the same:

usermod -u 123 $myusername

The following diagram illustrates how the UID and name are communicated between the NFS client, NFS gateway, and NameNode.

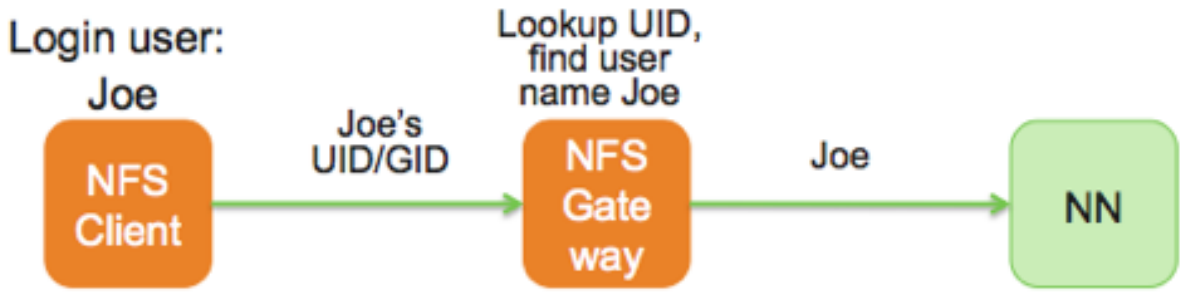

Related Information

Access HDFS from the NFS Gateway

APIs for accessing HDFS

Use the WebHDFS REST API to access an HDFS cluster from applications external to the cluster. WebHDFS supports all HDFS user operations including reading files, writing to files, making directories, changing permissions and renaming. In addition, WebHDFS uses Kerberos and delegation tokens for authenticating users.

Set up WebHDFS on a secure cluster

Setting up WebHDFS on a secure cluster requires you to update certain properties on hdfs-site.xml, create an HTTP service user principal, create a keytab for the principal, and verify the association of the principal and keytab with the correct HTTP service.

Procedure

1. Create an HTTP service user principal.

```
kadmin: addprinc -randkey HTTP/$<Fully_Qualified_Domain_Name>@$<Realm_Na
me>.COM
```

where:

- Fully_Qualified_Domain_Name: Host where the NameNode is deployed.
- Realm_Name: Name of your Kerberos realm.

2. Create a keytab file for the HTTP principal.

```
kadmin: xst -norandkey -k /etc/security/spnego.service.keytab HTTP/$<Ful
ly_Qualified_Domain_Name>
```

3. Verify that the keytab file and the principal are associated with the correct service.

```
klist –k -t /etc/security/spnego.service.keytab
```

4. Add the dfs.web.authentication.kerberos.principal and dfs.web.authentication.kerberos.keytab properties to hdfs-site.xml.

```
<property>
  <name>dfs.web.authentication.kerberos.principal</name>
  <value>HTTP/$<Fully_Qualified_Domain_Name>@$<Realm_Name>.COM</value>
</property>
<property>
  <name>dfs.web.authentication.kerberos.keytab</name>
  <value>/etc/security/spnego.service.keytab</value>
</property>
```

5. Restart the NameNode and the DataNodes.

Using HttpFS to provide access to HDFS

Apache Hadoop HttpFS is a service that provides HTTP access to HDFS. HttpFS has a REST HTTP API supporting all HDFS filesystem operations (both read and write).

Common HttpFS use cases are:

- Read and write data in HDFS using HTTP utilities (such as curl or wget) and HTTP libraries from languages other than Java (such as Perl).
- Transfer data between HDFS clusters running different versions of Hadoop (overcoming RPC versioning issues), for example using Hadoop DistCp.
- Accessing WebHDFS using the NameNode Web UI port (default port 9870). Access to all data hosts in the cluster is required, because WebHDFS redirects clients to the DataNode port (default port 9864). If the cluster is behind a firewall, and you use WebHDFS to read and write data to HDFS, then Cloudera recommends you use the HttpFS server. The HttpFS server acts as a gateway. It is the only system that is allowed to send and receive data through the firewall.

HttpFS supports Hadoop pseudo-authentication, HTTP SPNEGO Kerberos, and additional authentication mechanisms using a plugin API. HttpFS also supports Hadoop proxy user functionality.

The webhdfs client file system implementation can access HttpFS using the Hadoop filesystem command (hadoop fs), by using Hadoop DistCp, and from Java applications using the Hadoop file system Java API.

The HttpFS HTTP REST API is interoperable with the WebHDFS REST HTTP API.

Add the HttpFS role

You can use Cloudera Manager to add the HttpFS role.

1. Go to the HDFS service.
2. Click the Instances tab.
3. Click Add Role Instances.
4. Click the text box below the HttpFS field. The Select Hosts dialog box displays.

5. Select the host on which to run the role and click OK.

   Note: You can select any node other than the master nodes.

6. Click Continue.
7. Check the checkbox next to the HttpFS role and select Actions for Selected Start .

Using Load Balancer with HttpFS

Configure the HttpFS Service to work with the load balancer that you configured for the service.

1. In the Cloudera Admin Console, navigate to Cluster <HDFS service> .
2. On the Configuration tab, search for the following property:

```
HttpFS Load Balancer
```

3. Enter the hostname and port for the load balancer in the following format:

```
<HOSTNAME>:<PORT>
```

4. Save the changes.

Note: When you set this property, Cloudera Manager regenerates the keytabs for HttpFS roles. The principal in these keytabs contains the load balancer hostname. If there is a Hue service that depends on this HDFS service, the Hue service has the option to use the load balancer as its HDFS Web Interface Role.

HttpFS authentication

To enable HttpFS to work with Kerberos security on your Cloudera cluster, ensure that you have configured Kerberos for the cluster.

Important: If the NameNode, Secondary NameNode, DataNode, JobTracker, TaskTrackers, ResourceManager, NodeManagers, HttpFS, or Oozie services are configured to use Kerberos HTTP SPNEGO authentication, and two or more of these services are running on the same host, then all of the running services must use the same HTTP principal and keytab file used for their HTTP endpoints.

Use curl to access a URL protected by Kerberos HTTP SPNEGO

To access a URL protected by Kerberos HTTP SPNEGO, ensure that your version of curl supports GSS and is capable of running curl -V.

Procedure

1. Run curl -V.

```
$ curl -V
curl 7.19.7 (universal-apple-darwin10.0) libcurl/7.19.7 OpenSSL/0.9.8l
zlib/1.2.3
Protocols: tftp ftp telnet dict ldap http file https ftps
Features: GSS-Negotiate IPv6 Largefile NTLM SSL libz
```

2. Log on to the KDC using kinit.

```
$ kinit
Please enter the password for username@LOCALHOST:
```

3. Use curl to fetch the protected URL.

```
$ curl --cacert
              /path/to/truststore.pem --negotiate -u : -b ~/cookiejar.txt
 -c ~/cookiejar.txt https://localhost:14000/webhdfs/v1/?op=liststatus
```

where:

- The --cacert option is required if you are using TLS/SSL certificates that curl does not recognize by default.
- The --negotiate option enables SPNEGO in curl.
- The -u: option is required but the username is ignored (the principal that has been specified for kinit is used).
- The -b and -c options are used to store and send HTTP cookies.
- Cloudera does not recommend using the -k or --insecure option as it turns off curl's ability to verify the certificate.

Data storage metrics

Use Java Management Extensions (JMX) APIs to collect the metrics exposed by the various HDFS daemons.

You can use various HDFS metrics to understand the state of your cluster.

Using JMX for accessing HDFS metrics

You can access HDFS metrics over Java Management Extensions (JMX) through either the web interface of an HDFS daemon or by directly accessing the JMX remote agent.

Using the HDFS daemon web interface

You can access JMX metrics through the web interface of an HDFS daemon. This is the recommended method.

For example, use the following command format to access the NameNode JMX:

```
curl -i http://localhost:50070/jmx
```

You can use the qry parameter to fetch only a particular key:

```
curl -i http://localhost:50070/jmx?qry=Hadoop:service=NameNode,name=NameNode
Info
```

Directly accessing the JMX remote agent

This method requires that the JMX remote agent is enabled with a JVM option when starting HDFS services.

For example, the following JVM options in hadoop-env.sh are used to enable the JMX remote agent for the NameNode. It listens on port 8004 with SSL disabled. The user name and password are saved in the mxremote.pas sword file.

```
export HADOOP_NAMENODE_OPTS="-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.password.file=$HADOOP_CONF_DIR/jmxremote.pass
word
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.port=8004 $HADOOP_NAMENODE_OPTS"
```

See the Oracle Java SE documentation for more information about the related settings.

You can also use the jmxquery tool to retrieve information through JMX.

Hadoop has a built-in JMX query tool, jmxget. For example:

```
hdfs jmxget -server localhost -port 8004 -service NameNode
```

Note: jmxget requires that authentication be disabled, as it does not accept a user name and password.

Using JMX can be challenging for operations personnel who are not familiar with JMX setup, especially JMX with SSL and firewall tunnelling. Therefore, we recommend that you collect JMX information through the web interface of HDFS daemons rather than directly accessing the JMX remote agent.

Related Information

JMX Query

Monitoring and Management Using JMX Query

HDFS Metrics

Many aggregate metrics are available in addition to base metrics. If an entity type has parents defined, you can formulate all possible aggregate metrics using the formula base_metric_across_parents.

In addition, metrics for aggregate totals can be formed by adding the prefix total_ to the front of the metric name.

Use the type-ahead feature in the Cloudera Manager chart browser to find the exact aggregate metric name, in case the plural form does not end in "s".

For example, the following metric names may be valid for HDFS:

- alerts_rate_across_clusters
- total_alerts_rate_across_clusters

Some metrics, such as alerts_rate, apply to nearly every metric context. Others only apply to a certain service or role.

Metric Name	Description	Unit	Parents
`alerts_rate`	The number of alerts.	events per second	cluster
`block_capacity`	The block capacity of the NameNode	blocks	cluster
`blocks_total`	Blocks total	blocks	cluster
`blocks_with_corrupt_replicas`	Blocks with corrupt replicas	blocks	cluster
`canary_duration`	Duration of the last or currently running canary job	ms	cluster
`cm_time_since_last_fsimage_fetch`	Time since last FsImage was fetched by Cloudera Reports Manager	seconds	cluster
`cm_time_since_last_fsimage_index`	Time since last FsImage was indexed By Cloudera Reports Manager	seconds	cluster
`dfs_capacity`	Total configured HDFS storage capacity	bytes	cluster
`dfs_capacity_used`	Storage space used by HDFS files	bytes	cluster
`dfs_capacity_used_non_hdfs`	Storage space used by non-HDFS files	bytes	cluster
`events_critical_rate`	The number of critical events.	events per second	cluster
`events_important_rate`	The number of important events.	events per second	cluster
`events_informational_rate`	The number of informational events.	events per second	cluster
`excess_blocks`	The total number of excess blocks	blocks	cluster

Metric Name	Description	Unit	Parents
`expired_heartbeats`	The number of expired heartbeats	heartbeats	cluster
`files_total`	The number of files and directories in the HDFS	files	cluster
`health_bad_rate`	Percentage of Time with Bad Health	seconds per second	cluster
`health_concerning_rate`	Percentage of Time with Concerning Health	seconds per second	cluster
`health_disabled_rate`	Percentage of Time with Disabled Health	seconds per second	cluster
`health_good_rate`	Percentage of Time with Good Health	seconds per second	cluster
`health_unknown_rate`	Percentage of Time with Unknown Health	seconds per second	cluster
`missing_blocks`	Missing blocks	blocks	cluster
`pending_deletion_blocks`	The number of replicas pending deletion.	replicas	cluster
`pending_replication_blocks`	The number of blocks with replication pending	blocks	cluster
`scheduled_replication_blocks`	The number of blocks with replication currently scheduled	blocks	cluster
`under_replicated_blocks`	Under-replicated blocks	blocks	cluster
`xceivers`	Transceivers	transceivers	cluster

Using HdfsFindTool to find files

HDFSFindTool is essentially the HDFS version of the Linux filesystem find command. You can use the tool to search one or more HDFS directory trees, find all HDFS files that match the specified expression, and apply selected actions to them.

Note: HDFSFindtool is *not* available for download with Cloudera. If you have the required paywall credentials to access the CDH archives, you can download the JAR file for HDFSFindtool and use it.

Downloading Hdfsfindtool from the CDH archives

If you have access to CDH credentials and want to use HDFSFindtool with your Cloudera deployment, then you can download the tool from the archives.

Before you begin

You must have the required paywall credentials to access the CDH archives.

About this task

You can download the CDH5 jar to run HDFSFindtool commands from the CDH archives.

Procedure

1. Log into the CDH archives with the help of your paywall credentials and download the search tarball.

```
$ wget --user=<your_paywall_username> --ask-password https://archive.clou
dera.com/p/cdh5/cdh/5/search-1.0.0-cdh5.16.2.tar.gz
```

   Here, <your_paywall_username> indicates your Paywall username. In addition, enter your password when prompted.

2. Extract the downloaded tarball file.

```
$ tar -xvf search-1.0.0-cdh5.16.2.tar.gz
```

3. Run HDFSFindtool.

```
$ hadoop jar cloudera-search-1.0.0-cdh5.16.2/dist/search-mr-1.0.0-cdh5.1
6.2.jar org.apache.solr.hadoop.HdfsFindTool -find /tmp/ -type f
```

Example

The following example shows how you can use HDFSFindtool:

```
$ hadoop jar cloudera-search-1.0.0-cdh5.16.2/dist/search-mr-1.0.0-cdh5.16.2.
jar org.apache.solr.hadoop.HdfsFindTool -find /tmp/ -type f
WARNING: Use "yarn jar" to launch YARN applications.
hdfs://ns1/tmp/logs/systest/logs/application_1575879043308_0001/file1
hdfs://ns1/tmp/my_table.avsc
```

Cloudera Runtime 7.3.1

Schema Registry Overview

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Schema Registry overview.....4
Examples of interacting with Schema Registry.....5
Schema Registry use cases.....6
Registering and querying a schema for a Kafka topic.....6
Deserializing and serializing data from and to a Kafka topic.....7
Dataflow management with schema-based routing.....7
Schema Registry component architecture.....7
Schema Registry concepts.....8
Schema entities.....8
Compatibility policies.....10
Importance of logical types in Avro.....12

Schema Registry overview

Learn about the basic features of Schema Registry and how it integrates into Cloudera Streams Processing.

Schema Registry provides a shared repository of schemas that allows applications to flexibly interact with each other.

As displayed in the following diagram, Schema Registry is part of the enterprise services that power streams processing.

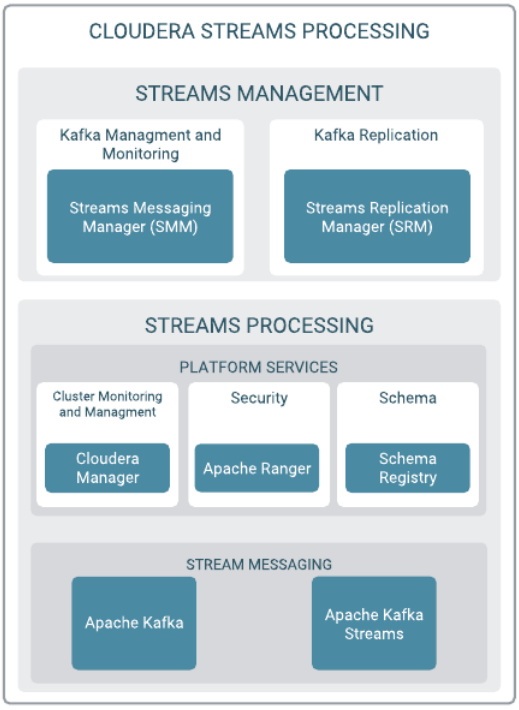

Applications built often need a way to share metadata across three dimensions:

- Data format
- Schema
- Semantics or meaning of the data

The Schema Registry design principle is to provide a way to tackle the challenges of managing and sharing schemas between components. The schemas are designed to support evolution such that a consumer and producer can understand different versions of those schemas but still read all information shared between both versions and safely ignore the rest.

Hence, the value that Schema Registry provides and the applications that integrate with it are the following:

- Centralized registry

  Provide reusable schema to avoid attaching schema to every piece of data
- Version management

  Define relationship between schema versions so that consumers and producers can evolve at different rates
- Schema validation

  Enable generic format conversion, generic routing and data quality

The following image displays Schema Registry usage in Flow and Streams Management:

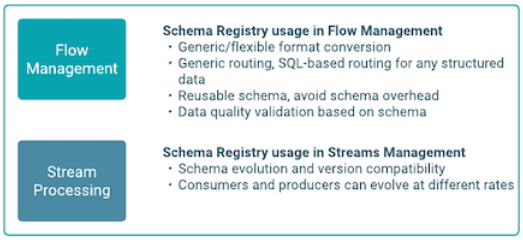

Examples of interacting with Schema Registry

Learn about different ways of interacting with Schema Registry.

Schema Registry UI

You can use the Schema Registry UI to create schema groups, schema metadata, and add schema versions.

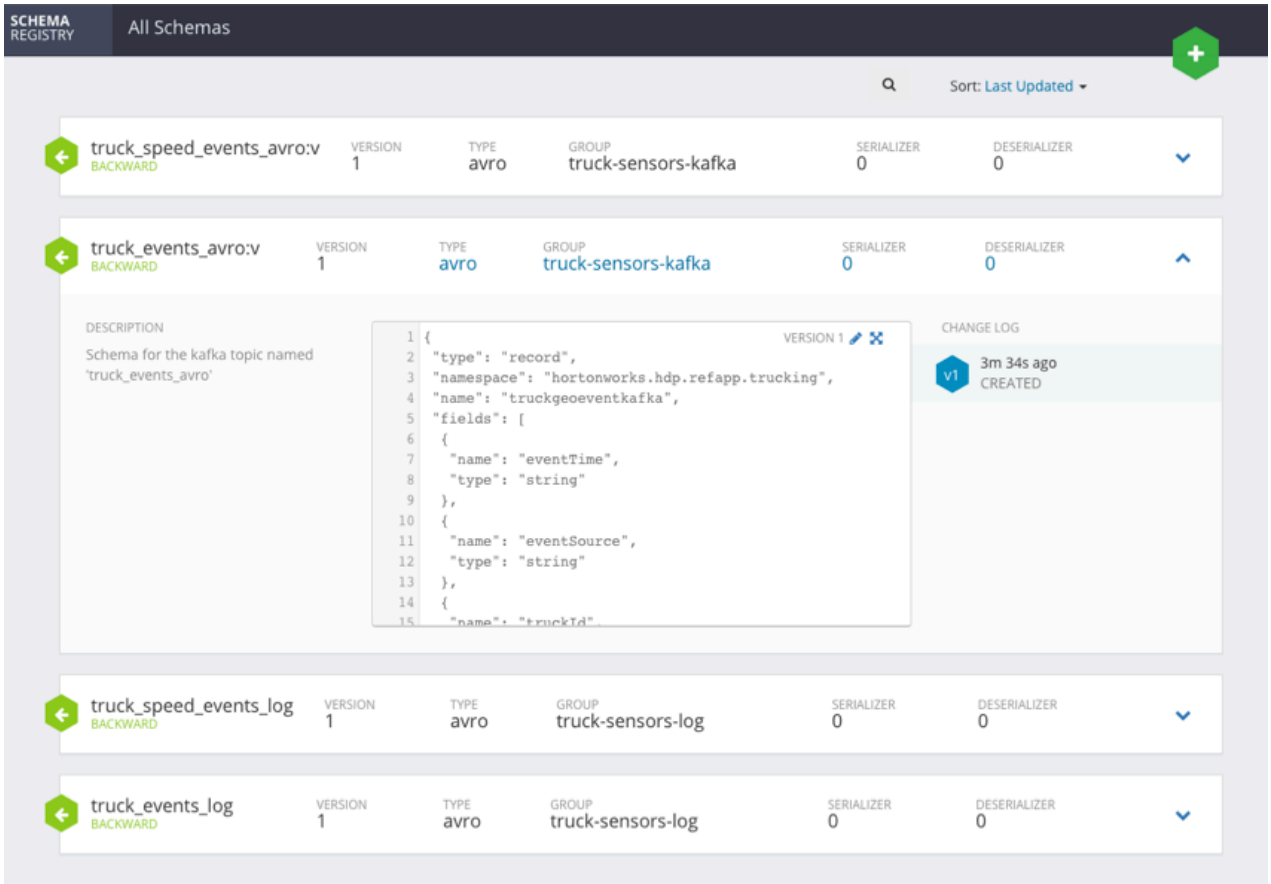

Schema Registry API

You can access the Schema Registry API swagger directly from the UI.

To do this, you need to append your URL with /swagger/. For example: https://localhost:7790/swagger/.

For example, you can create a schema by using Swagger, as shown in the following steps:

1. Navigate to the Swagger UI: http:<sr-host>:<sr-port>/swagger/
2. Go to the Schema section and open POST /api/v1/schemaregistry/schemas.
3. Click Try it out.

4. Enter the input for your schema in the body field.

   There is already an example input in the body. You can also edit it.
5. Click Execute.

   After the successful execution of the command, you see the newly created schema in you Schema Registry UI

For more information about the tasks you can perform through Swagger, see Cloudera Schema Registry REST API Reference.

Java client

You can review the following GitHub repositories for examples of how to interact with the Schema Registry Java client:

- https://github.com/georgevetticaden/cdf-ref-app/blob/master/csp-trucking-schema/src/main/java/cloudera/cdf/csp/schema/refapp/trucking/schemaregistry/TruckSchemaRegistryLoader.java#L62
- https://github.com/hortonworks/registry/blob/0.9.0/examples/schema-registry/avro/src/main/java/com/hortonworks/registries/schemaregistry/examples/avro/SampleSchemaRegistryClientApp.java

Java and Scala

See the following examples of using schema related API:

https://github.com/hortonworks/registry/blob/HDF-3.4.1.0-5-tag/examples/schema-registry/avro/src/main/java/com/hortonworks/registries/schemaregistry/examples/avro/SampleSchemaRegistryClientApp.java

https://github.com/hortonworks/registry/blob/HDF-2.1.0.0/schema-registry/README.md

Kafka SerDes

See the following example of using the Schema Registry Kafka SerDes:

https://github.com/hortonworks/registry/blob/0.9.0/examples/schema-registry/avro/src/main/java/com/hortonworks/registries/schemaregistry/examples/avro/KafkaAvroSerDesApp.java

Schema Registry also supports serializing objects as JSON. To enable JSON serialization or deserialization, set the following properties:

- value.serializer=com.hortonworks.registries.schemaregistry.serdes.json.kafka.KafkaJsonSerial
- value.deserializer=com.hortonworks.registries.schemaregistry.serdes.json.kafka.KafkaJsonDeserializer

Jackson 2 is used for serialization, so any Java object which can be processed with this library is also going to be processed by Schema Registry.

.NET client

See the following examples of interacting with the Schema Registry .NET client:

.NET client examples

Schema Registry use cases

Learn about different use cases of using Schema Registry.

Registering and querying a schema for a Kafka topic

Learn how to use Schema Registry to track metadata for a Kafka topic.

When Kafka is integrated into enterprise organization deployments, you typically have many different Kafka topics used by different applications and users. With the adoption of Kafka within the enterprise, some key questions that often come up are the following:

- What are the different events in a given Kafka topic?
- What do I put into a given Kafka topic?
- Do all Kafka events have a similar type of schema?
- How do I parse and use the data in a given Kafka topic?

While Kafka topics do not have a schema, having an external store that tracks this metadata for a given Kafka topic helps to answer these common questions. Schema Registry addresses this use case.

One important point to note is that Schema Registry is not just a metastore for Kafka. Schema Registry is designed to be a generic schema store for any type of entity or store (log files, or similar.)

Deserializing and serializing data from and to a Kafka topic

Learn how to use Schema Registry to store metadata on reading or deserializing and writing or serializing data from and to Kafka topics.

You can store metadata for the format of how data should be read and how it should be written. Schema Registry supports this use case by providing capabilities to store JAR files for serializers and deserializers, and then mapping the SerDes to the schema.

Dataflow management with schema-based routing

Learn how to use Schema Registry to support NiFi dataflow management.

If you are using NiFi to move different types of syslog events to downstream systems, you have data movement requirements where you need to parse the syslog event to extract the event type and route the event to a certain downstream system (for example, different Kafka topics) based on the event type.

Without Schema Registry, NiFi uses regular expressions or other utilities to parse the event type value from the payload and stores it into a flowfile attribute. Then NiFi uses routing processors (for example, RouteOnAttribute) to use the parsed value for routing decisions. If the structure of the data changes considerably, this type of extract and routing pattern is brittle and requires frequent changes.

With the introduction of Schema Registry, NiFi queries the registry for schema and then retrieves the value for a certain element in the schema. In this case, even if the structure changes, as long as compatibility policies are adhered to, NiFi's extract and routing rules do not change. This is another common use case for Schema Registry.

Schema Registry component architecture

Learn about the component architecture of Schema Registry.

Schema Registry has the following main components:

- Registry web server

  Web application exposing the REST endpoints you can use to manage schema entities. You can use a web proxy and load balancer with multiple web servers to provide high availability and scalability.
- Pluggable storage

  Schema Registry uses the following types of storages:

  - Schema metadata storage

    Relational store that holds the metadata for the schema entities. MySQL, PostgreSQL, and Oracle databases are supported.
  - SerDes storage

    File storage for the serializer and deserializer jars. Local file system and HDFS storage are supported. Local file system storage is the default.
- Schema Registry client

  A java client that components can use to interact with the RESTful services.

The following diagram represents the component architecture of Schema Registry.

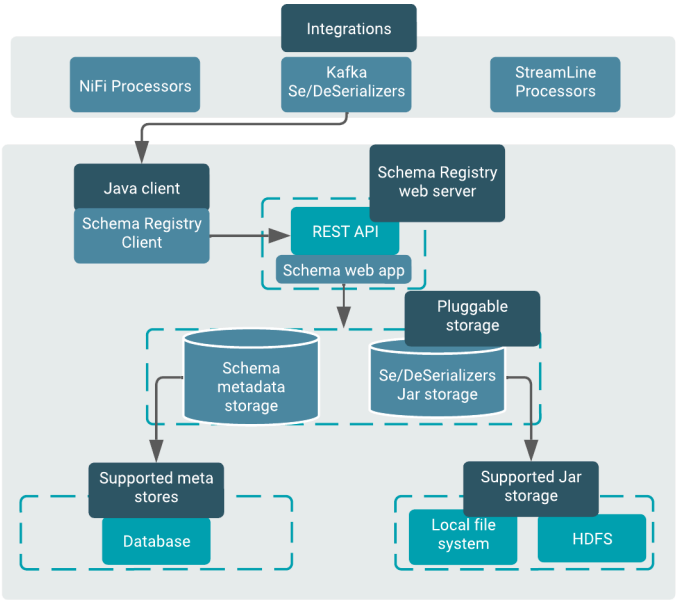

There are two integration points:

- Custom NiFi processors

  New processors and controller services in NiFi that interact with the Schema Registry.
- Kafka serializer and deserializer

  A Kafka serializer or deserializer that uses Schema Registry. The Kafka SerDes can be found on GitHub.

Schema Registry concepts

Learn about the basic concepts of Schema Registry.

Schema entities

Learn about the three types of schema entity types: Schema Group, Schema Metadata, Schema Version.

You can use Schema Registry to work with three types of schema entities. The following image shows the types of Schema entities:

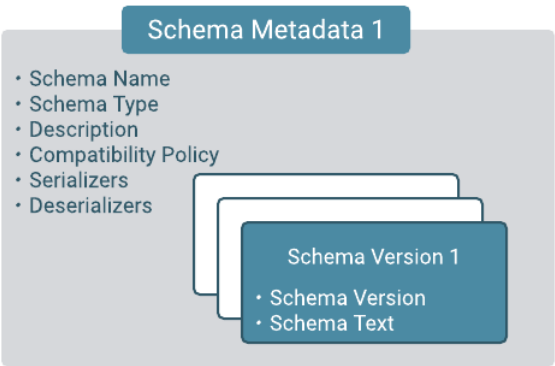

The following table provides a more detailed description of the schema entity types:

Table 1: Schema entity types

Entity Type	Description	Example
Schema Group	A logical grouping of similar schemas. A Schema Group can be based on any criteria you have for managing schemas. Schema Groups can have multiple Schema Metadata definitions.	• Group Name – truck-sensors-log • Group Name – truck-sensors-kafka

Entity Type	Description	Example
Schema Metadata	Metadata associated with a named schema. A metadata definition is applied to all the schema versions that are assigned to it. Key metadata elements include: • Schema Name – A unique name for each schema. Used as a key to look up schemas. • Schema Type – The format of the schema. The supported formats are Avro and JSON. • Compatibility Policy – The compatibility rules that exist when the new schemas are registered. • Serializers/Deserializers – A set of serializers and deserializers that you can upload to the registry and associate with schema metadata definitions.	• Schema Name – truck_events_avro:v • Schema Type – avro • Compatibility Policy – SchemaCompatibility.BACKWARD
Schema Version	The versioned schema associated a schema metadata definition.	`{ "type" : "record", "namespace" : "hortonworks.hdp.refapp.trucking", "name" : "truckgeoevent", "fields" : [{ "name" : "eventTime" , "type" : "string" }, { "name" : "eventSource" , "type" : "string" }, { "name" : "truckId" , "type" : "int" }, { "name" : "driverId" , "type" : "int"}, { "name" : "driverName" , "type" : "string"}, { "name" : "routeId" , "type" : "int"}, { "name" : "route" , "type" : "string"}, { "name" : "eventType" , "type" : "string"}, { "name" : "longitude" , "type" : "double"}, { "name" : "latitude" , "type" : "double"}, { "name" : "correlationId" , "type" : "long"}] }`

Compatibility policies

Learn about the compatibility policies to be able to manage versioning of schemas in Schema Registry.

A key Schema Registry feature is the ability to version schemas as they evolve. Compatibility policies are created at the schema metadata level, and define evolution rules for each schema.

After a policy has been defined for a schema, any subsequent version updates must honor the schema's original compatibility, otherwise you experience an error.

Compatibility of schemas can be configured with any of the following values:

Backward Compatibility

Indicates that a new version of a schema would be compatible with earlier versions of that schema. This means the data written from an earlier version of the schema can be deserialized with a new version of the schema.

When you have a Backward Compatibility policy on your schema, you can evolve schemas by deleting portions, but you cannot add information. New fields can be added only if a default value is also provided for them.

Forward Compatibility

Indicates that an existing schema is compatible with subsequent versions of the schema. That means the data written from a new version of the schema can still be read with an old version of the schema.

When you have a Forward Compatibility policy on your schema, you can evolve schemas by adding new information, but you cannot delete existing portions.

Full Compatibility

Indicates that a new version of the schema provides both backward and forward compatibilities.

None

Indicates that no compatibility policy is in place.

Note: The default value is Backward. You can set the compatibility policy when you are adding a schema. Once set, you cannot change it.

Validation level

Validation level limits the scope of the compatibility check when you add a new version of the schema. It checks compatibility only against the latest version or all previous versions based on the validation level value configured. Validation level only makes sense when coupled with compatibility. When compatibility is set to None then the validation is also ignored.

The validation level cannot be set from the Schema Registry UI. You can set it through the Swagger REST API. For example,

```
SchemaMetadata.Builder builder = new SchemaMetadata.Builder("Blah")
        .type("avro")
        .schemaGroup("Kafka")
        .description("test")
        .compatibility(SchemaCompatibility.BACKWARD)
        .validationLevel(SchemaValidationLevel.ALL);
```

When using the Schema Registry UI, the validation level defaults to ALL.

The supported validation levels are as follows:

- All

  Schemas are compatible across multiple versions, for example, with Backward Compatibility, data written with version 1 can be read with version 7 too. All is the default validation level.
- Latest

  There is no transient compatibility, only the latest version is used, for example, with Backward Compatibility, data written with version 1 can only be read with version 2, not with version 7.

Allowed schema changes for different compatibilities

The following table presents a summary of the types of schema changes allowed for the different compatibility types and validation levels, for a given subject.

Compatibility types	Validation levels	Changes allowed	Check against which schemas	Upgrade first
Backward	Latest	• Delete fields • Add optional fields	Last version	Consumers
Backward	All	• Delete fields • Add optional fields	All previous versions	Consumers
Forward	Latest	• Add fields • Delete optional fields	Last version	Producers
Forward	All	• Add fields • Delete optional fields	All previous versions	Producers
Full	All	• Add optional fields • Delete optional fields	Last version	Any order
None	All	• All changes are accepted	Compatibility checking disabled	Depends

Importance of logical types in Avro

The importance of logical types in Avro format for Schema Registry can be represented with an example using a decimal logical type.

Huge numbers that cannot be stored in a long type with 8 bytes can be stored as byte arrays with decimal logical type using Avro. This means that the data will be stored as a byte array, which is flexible, but it also means that when you parse the data from Avro, you need additional information, the logical type, to interpret the raw data, the byte array, properly. By default, you will only get a byte array for this kind of field value, and you have to parse that byte array for the required logical type. Each Avro logical type has their own Java classes that they can be parsed for. Decimal logical type has BigDecimal in Java.

If you use generated classes, such as specific classes, on the classpath for Avro, the conversion happens automatically based on the types in the classes. If you do not use specific classes, Cloudera provides configuration to enable automatic conversion for logical type record fields during serialization and deserialization. Check the logical.type .conversion.enabled flag. This flag has no effect if specific.avro.reader is set to true, which means specific classes are used.

For more information, see the Logical Types section in the official Avro documentation.

Cloudera Runtime 7.3.1

Schema Registry Reference

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

SchemaRegistryClient properties reference.....4

KafkaAvroSerializer properties reference.....6

KafkaAvroDeserializer properties reference.....7

SchemaRegistryClient properties reference

Review the following reference for a comprehensive list of the SchemaRegistryClient properties.

Property Name	Description	Type	Default Value
schema.registry.url	The URL of the Schema Registry server which this client connects to.	String	http://localhost:9090/api/v1
schema.registry.client.local.jars.path	The local directory path to which downloaded JARs are copied to.	String	/tmp/schema-registry/local-jars
schema.registry.client.class.loader.cache.size	The maximum size of the classloader cache.	Int	1024
schema.registry.client.class.loader.cache.expiry.interval	The expiry interval (in seconds) of an entry in the classloader cache.	Int	3600 sec
schema.registry.client.schema.version.text.cache.size	The maximum size of schema text cache.	Int	1024
schema.registry.client.schema.version.text.cache.expiry.interval	The expiry interval (in seconds) of an entry in schema version cache.	Int	300 sec
schema.registry.client.schema.metadata.cache.size	Maximum size of schema metadata cache.	Int	1024
schema.registry.client.schema.metadata.cache.expiry.interval	Expiry interval (in seconds) of an entry in schema metadata cache.	Int	300 sec
schema.registry.client.schema.text.cache.size	Maximum size of schema text cache.	Int	1024
schema.registry.client.schema.text.cache.expiry.interval	Expiry interval (in seconds) of an entry in schema text cache.	Int	300 sec
schema.registry.client.url.selector	Schema Registry URL selector class.	String	com.hortonworks.registries.schemaregistry.client.Fail
sasl.jaas.config	Schema Registry Dynamic JAAS configuration for SASL connection.	String	null
schema.registry.auth.username	Username for basic authentication.	String	null
schema.registry.auth.password	Password for basic authentication.	String	null
schema.registry.hash.function	Hashing algorithm for generating schema fingerprints.	String	MD5
chema.registry.auth.type	The type of authentication the client should use. If the value is oauth2, it will be configured for oauth login, otherwise it will use Kerberos.	String	kerberos
schema.registry.oauth.client.id	Client ID for OAuth server in case of oauth login.	String	empty
schema.registry.oauth.secret	Secret for OAuth server in case of oauth login.	String	empty
schema.registry.oauth.server.url	OAuth server URL in case of oauth login.	String	empty
schema.registry.oauth.scope	OAuth scope in case of oauth login.	String	empty
schema.registry.oauth.request.method	HTTP method for requesting the oauth token	String	post

Property Name	Description	Type	Default Value
connector.provider.class	Classname of a Jersey connector provider. (For example: org.glassfish.jersey.apache.connector.ApacheConnectorProvider) Make sure the class is on classpath. If this is set, Backoff policy might need to be set to com.hortonworks.registries.schemaregistry.retry.policy.ExponentialBackoffPolicy.	String	empty
schema.registry.client.ssl	Schema Registry SSL configuration[1]	Object	null
schema.registry.client.retry.policy	Schema Registry Client retry configuration[2]	Object	`ExponentialBackoffPolicy` with the following default properties: `sleepTimeMs=1000` `maxAttempts=5` `timeoutMs=90000`

Related Information

Compatibility policies

[1] The schema.registry.client.ssl property expects a map of SSL configuration properties. The following snippet lists all valid properties.

```
protocol: SSL
hostnameVerifierClass: com.hortonworks.registries.schemaregistry.avro.util
.AcceptAllHostnameVerifier
keyStoreType: JKS
keyStorePath: ./src/test/resources/jks/client.jks
keyStorePassword: clientpwd
keyPassword:
keyStoreProvider:
keyManagerFactoryProvider:
keyManagerFactoryAlgorithm:
trustStoreType: JKS
trustStorePath: ./src/test/resources/jks/client.jks
trustStorePassword: clientpwd
trustStoreProvider:
trustManagerFactoryProvider:
trustManagerFactoryAlgorithm:
```

[2] The schema.registry.client.retry.policy expects a retry policy and a number of configuration properties related to the retry policy.

The valid policies are as follows:

```
com.hortonworks.registries.schemaregistry.retry.policy.NOOPBackoffPolicy
com.hortonworks.registries.schemaregistry.retry.policy.FixedTimeBackoffPoli
cy
com.hortonworks.registries.schemaregistry.retry.policy.ExponentialBackoff
Policy
```

The valid properties of the policies are as follows:

```
sleepTimeMs=[***SLEEPTIME VALUE***]
maxAttempts=[***MAX ATTEMPT VALUE***]
timeoutMs=[***TIMEOUT VALUE***]
```

KafkaAvroSerializer properties reference

Review the following reference for a comprehensive list of the KafkaAvroSerializer properties.

Property Name	Description	Type	Default Value
schema.compatibility	The default compatibility of the new schemas created by the serializer. For more information about the compatibility options, see the *Compatibility policies* documentation.	String	BACKWARD
schema.group	The default group of the new schemas created by the serializer.	String	kafka
schema.name.key.suffix	The provided string will be appended to the end of the schema name created for record keys. The schema name is the topic name initially, so as a result, you will have topic name + provided suffix as schema name.	String	:k
schema.name.value.suffix	The provided string will be appended to the end of the schema name created for record values. The schema name is the topic name initially, so as a result, you will have topic name + provided suffix as schema name.	String	null
null.passthrough.enabled	If this is enabled, the serializer will return immediately with null value, without putting anything in the serialized output. Having null values can be important in compact topics. Having null as the latest value for a key indicates for Kafka that the corresponding key can be deleted during log compaction.	Boolean	false
slogical.type.conversion.enabled	If this is enabled, Avro record fields can be serialized that are instances of Java classes (BigDecimal, UUID, LocalDate, and so on) that can be serialized in Avro using logical types. For more information, see the Logical Types section in the official Avro documentation. This does not have to be enabled if SpecificRecords are used as part of generated classes. SpecificDatumWriter is used for serializing SpecificRecord instances and that writer handles logical types automatically.	Boolean	false

Property Name	Description	Type	Default Value
serdes.protocol.version	The protocol version to be used. The available default registered protocols are the following: • CONFLUENT_VERSION_PROTOCOL = 0x0; • METADATA_ID_VERSION_PROTOCOL = 0x1; • VERSION_ID_AS_LONG_PROTOCOL = 0x2; • VERSION_ID_AS_INT_PROTOCOL = 0x3; • JSON_PROTOCOL = 0x4;	Byte	0x3 (VERSION_ID_AS_INT_PROTOCOL)
store.schema.version.id.in.header	If this is set to true, the schema version ID will be stored in the record header instead of the value.	Boolean	false
key_schema_version_id_header_name	If the schema of the record key is stored in the header, the header name can be configured.	String	key.schema.version.id
value_schema_version_id_header_name	If the schema of the record value is stored in the header, the header name can be configured.	String	value.schema.version.id

The KafkaAvroSerializer contains an embedded SchemaRegistryClient. The configuration for SchemaRegistryClient should also be provided for the KafkaAvroSerializer. For more information about configurable properties for the *Schema Registry Client properties* reference documentation.

Related Information

Schema Registry Client properties reference

KafkaAvroDeserializer properties reference

Review the following reference for a comprehensive list of the KafkaAvroDeserializer properties.

Property Name	Description	Type	Default Value
schemaregistry.reader.schema.versions	This is for setting which version of the schema should be used for which topic.[3]	Map<String, Integer>	Empty map
specific.avro.reader	If this is enabled, SpecificDatumReader will be used for deserialization.	Boolean	false

[3] Example:

```
Map<String, Object> configs = new HashMap<>()
Map<String, Integer> readerVersions = new HashMap<>();
readerVersions.put("clicks", 2);
readerVersions.put("users", 1);
configs.put(“schemaregistry.reader.schema.versions”, readerVersions);
```

Property Name	Description	Type	Default Value
logical.type.conversion.enabled	If this is enabled, Avro record fields that have logical types can be deserialized directly into their corresponding Java classes (BigDecimal, UUID, LocalDate, and so on). If not enabled, the record field will be deserialized as raw type, without handling the logical type. In this case, probably the same logical type related conversion will be needed to get a valid value. If specific.avro.reader is set to true, this does not have any effect.	Boolean	false
key_schema_version_id_header_name	If the schema of the record key is stored in the header, the header name can be configured.	String	key.schema.version.id
value_schema_version_id_header_name	If the schema of the record value is stored in the header, the header name can be configured.	String	value.schema.version.id

The KafkaAvroDeserializer contains an embedded SchemaRegistryClient. The configuration for SchemaRegistryClient should also be provided for the KafkaAvroDeserializer. For more information about configurable properties for the *Schema Registry Client properties* reference documentation.

Related Information

Schema Registry Client properties reference

Cloudera Runtime 7.3.1

Schema Registry Security

Date published: 2019-08-22
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Schema Registry authorization through Ranger access policies.....4
Predefined access policies for Schema Registry.....4
Adding the user or group to a predefined access policy.....5
Creating a custom access policy.....7
Schema Registry authentication through OAuth2 JWT tokens.....8
JWT algorithms.....10
Public key and secret storage.....10
Authentication using OAuth2 with Kerberos.....10
Schema Registry server configuration.....11
Configuring the Schema Registry client.....13

Schema Registry authorization through Ranger access policies

User and group access to various Schema Registry functions is controlled through Apache Ranger.

Predefined access policies for Schema Registry allow the administrator to quickly add a user or user group to specify:

- Who can add or evolve schemas to a schema metadata.
- Who can view and edit schemas within a schema metadata.
- Who can upload the SerDes JAR files.

If a higher level of granularity is necessary, the administrator can create an access policy and add the user or user group to this custom policy.

Related Information

Predefined access policies for Schema Registry

Adding the user or group to a predefined access policy

Creating a custom access policy

Predefined access policies for Schema Registry

Based on a user's responsibilities, you can add users or user groups to one or more of the predefined access policies for Schema Registry and you can specify if they have the permission to create, read, update, or delete access policies..

The following image shows the predefined access policies for Schema Registry:

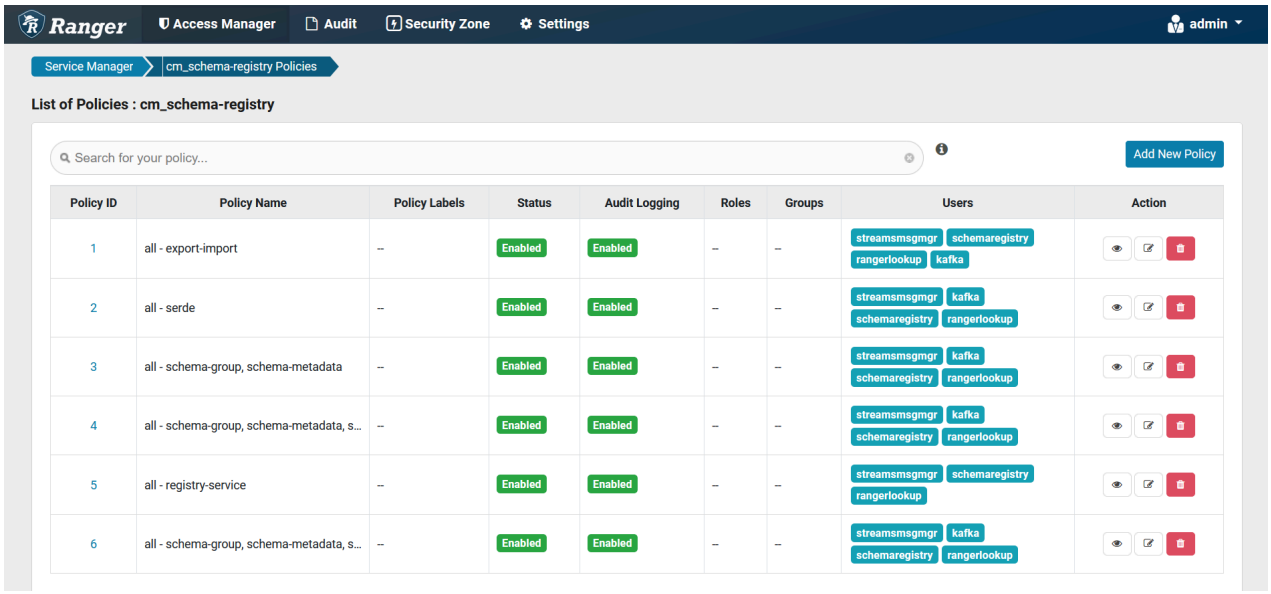

The following table describes the predefined access policies for Schema Registry:

Access Policy	Description
all - export-import	Allows users to import and export schemas to or from the Schema Registry service. For example, a user can import a JSON file with schemas from a Confluent Kafka topic to Cloudera Schema Registry.
all - serde	Allows users to store metadata regarding the format of how data should be read and how it should be written. Users can store JAR files for serializers and deserializers and then map the SerDes to the schema.

Access Policy	Description
all - schema-group, schema-metadata	Allows users to access the schema groups and schema metadata.
all - schema-group, schema-metadata, schema-branch	Allows users to access the schema groups, schema metadata, and schema branch.
all - registry-service	Allows users to access the Schema Registry service. If a user is added to this policy, the user can access all Schema Registry entities.
all - schema-group, schema-metadata, schema-branch, schema-version	Allows users to access the schema groups, schema metadata, schema branch, and schema version.

Related Information

Schema Registry authorization through Ranger access policies
Adding the user or group to a predefined access policy
Creating a custom access policy

Adding the user or group to a predefined access policy

When an authenticated user attempts to view, create, edit, or delete a Schema Registry entity, the system checks whether the user has privileges to perform that action. These privileges are determined by the Ranger access policies that a user is associated with.

Before you begin

For Ranger policies to work, you must have a user group named schemaregistry. If you use UNIX PAM, the sche maregistry user group must be on the node that hosts Schema Registry.

About this task

Determine the permissions required by a user or user group, and accordingly add the user or group to the appropriate predefined access policy.

Each predefined access policy controls access to one or more Schema Registry entities.

Procedure

1. From the Cloudera Manager home page, click the Ranger link.
   The **Ranger** management page appears.

2. Click Ranger Admin Web UI.

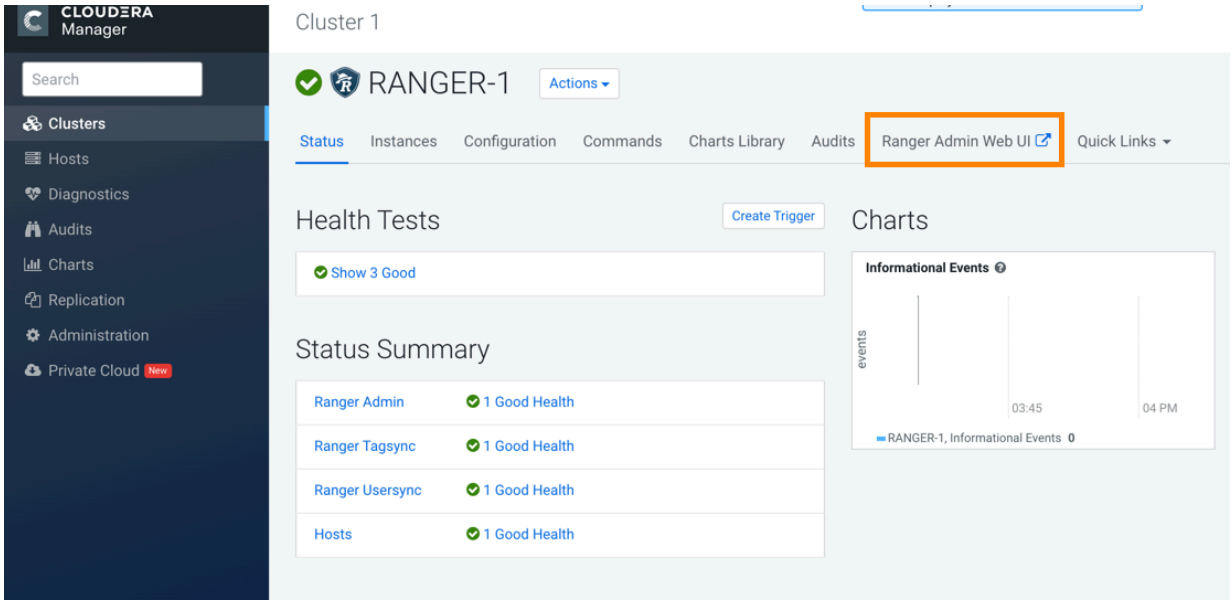

The **Ranger Log In** page appears.

3. Enter your user name and password to log in.
The **Ranger Service Manager** page appears.

The page is organized by service. Each cluster is listed under its respective service. For example, the Schema Registry clusters in the environment are listed under Schema Registry.

4. Select a cluster from the Schema Registry section.
The **List of Policies** page appears.

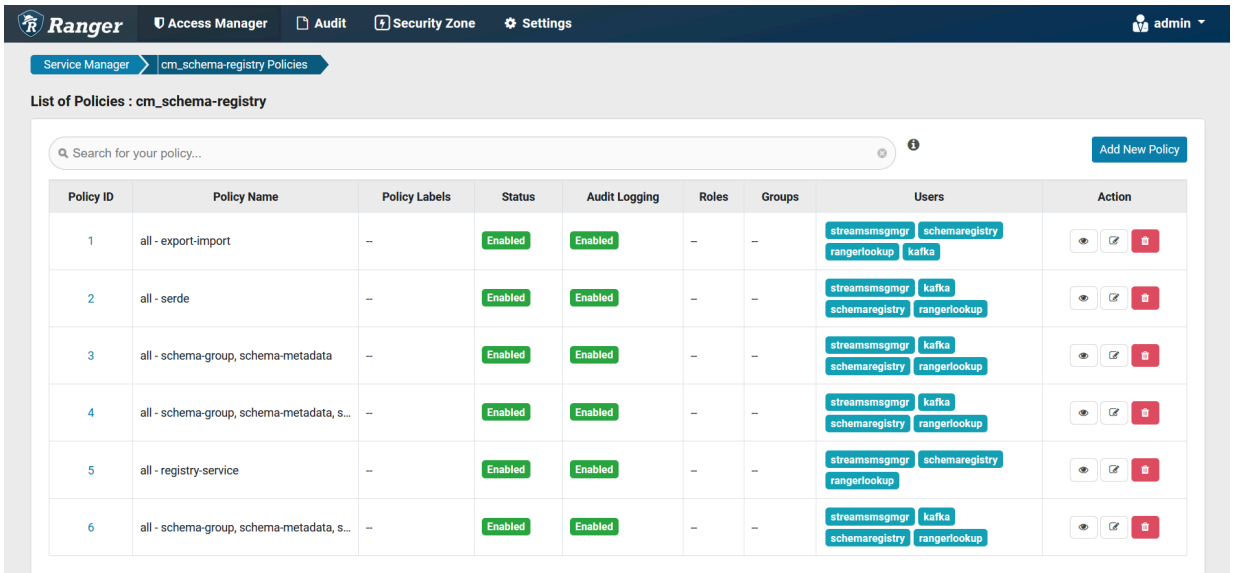

5. Click the ID of a policy.
The **Edit Policy** page appears.

6. In the Allow Conditions section, add the user or group to the respective Select User or Select Group field.

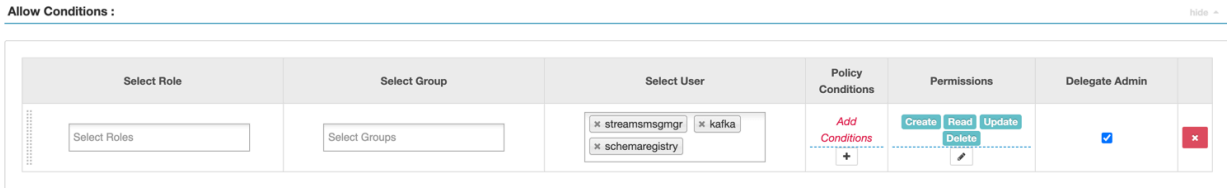

7. In the Policy Conditions field, enter the appropriate IP address.
8. From the Permissions field, select the appropriate permission.
9. Click Save.

Results

The user now has the rights according to the policy and the permissions you assigned to the user. These rights apply to all objects in the entities unless you specified otherwise in the Policy Conditions field.

Related Information

Schema Registry authorization through Ranger access policies

Predefined access policies for Schema Registry

Creating a custom access policy

Creating a custom access policy

You can create a custom access policy for a specific Schema Registry entity, specify an access type, and add a user or user group to the policy.

Before you begin

Determine and note down the following information:

- The schema registry entity that the user needs access to.
- Whether the user requires all objects in the entity or specific objects.
- Whether the user needs read, view, edit, or delete permissions to the entity.
- If there are any IP addresses to include or exclude from the user's access.

About this task

With a custom policy you can specify the Schema Registry entity and the type of access the user requires.

Procedure

1. Go to the **Ranger List of Policies** page.

2. Click Add New Policy.

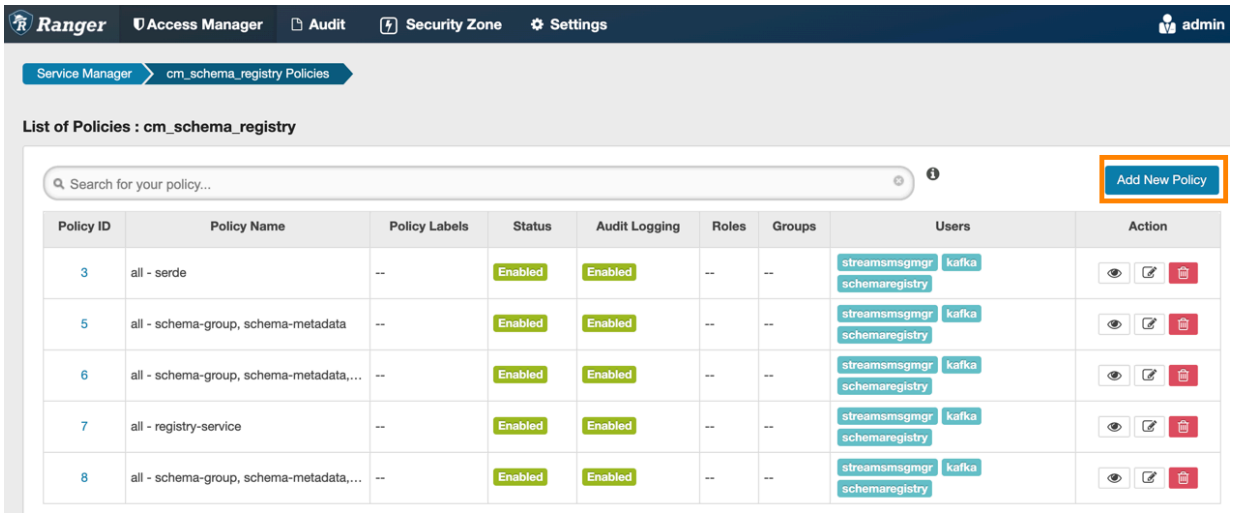

The **Create Policy** page appears.

3. Enter a unique name for the policy.
4. Optionally, enter a keyword in the Policy Label field to aid in searching for a policy.
5. Select a Schema Registry entity. You can choose the Schema Registry service, schema group, or SerDe. Then, perform one of the following tasks:
   - If you want the user to access all the objects in the entity, enter *.
   - If you want to specify the objects in the entity that a user can access, enter the name of the object in the text field.
6. Optionally, enter a description.
7. In the Allow Conditions section, add the user or group to the respective Select User or Select Group field.

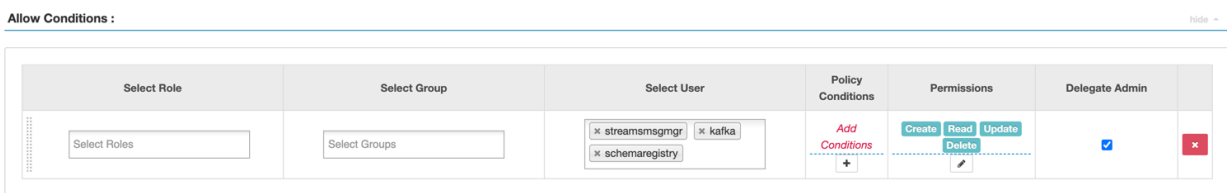

8. Optionally, from the Policy Conditions field, enter the appropriate IP address.
9. From the Permissions field, select the appropriate permission.
10. Click Save.

Results

The user now has the rights according to the policy and the permissions you assigned to the user.

Related Information

Schema Registry authorization through Ranger access policies

Predefined access policies for Schema Registry

Adding the user or group to a predefined access policy

Schema Registry authentication through OAuth2 JWT tokens

You can use OAuth2 JSON Web Token (JWT) in Schema Registry for authentication. Authorization continues to be implemented in Ranger; however, you can obtain the principal from a JWT token.

The flow for authenticating with OAuth2 tokens is as follows:

1. A client requests a token from the OAuth2 service.

   During Schema Registry startup the application obtains the public keys needed for validating the incoming tokens.
2. The client sends the HTTP requests to Schema Registry and these requests contain the bearer token in the HTTP header.
3. Schema Registry validates the token.

The following image shows the authentication flow with OAuth2 tokens:

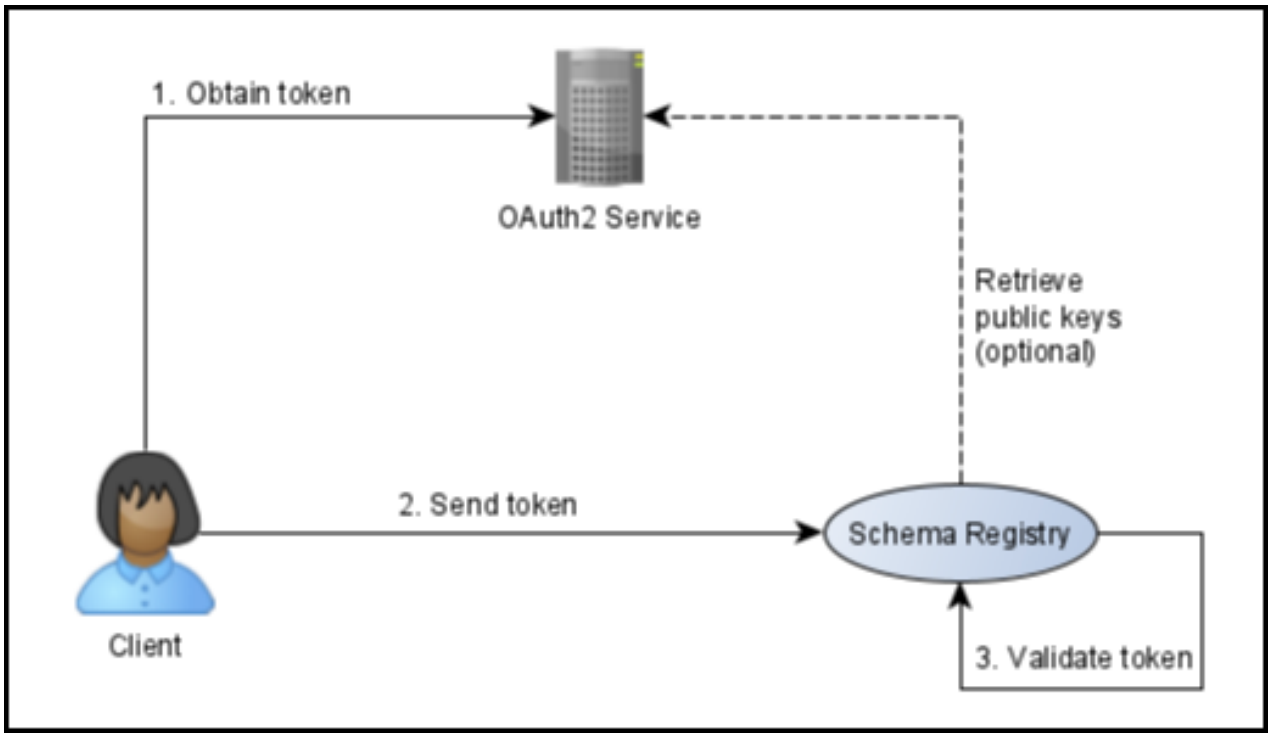

Note: Schema Registry currently supports JWT tokens only. There is no support for opaque tokens. The following is an example of a JWT token:

```
{
  "kid": "3",
  "alg": "HS256"
}
--------------------------------------------------
{
  "iss": "sender",
  "aud": "receiver",
  "exp": 1644492815,
  "jti": "5vgglGQCjC9_WZJMJg7mHQ",
  "iat": 1644492515,
  "sub": "abigel"
}
--------------------------------------------------
<signature>
```

The flow for authorization is as follows:

1. Once the token is validated, the principal is extracted from the JWT token. By default, the principal is stored in the sub field.
2. The principal is passed to Ranger which performs the authorization.

JWT algorithms

Similarly to Kafka, Schema Registry also uses Jose4J for validating the JWT tokens and their signatures.

This library supports a range of signing algorithms: HS256, HS384, HS512, RS256, RS384, and RS512.

For more information, see *Bitbucket jose4j Wiki*.

Related Information

Bitbucket jose4j Wiki

Public key and secret storage

Learn about public key, private key, and secret in JSON Web Token (JWT). Also learn about JSON Web Key (JWK), keystore, and property that Schema Registry supports for storing the public key or the secret.

When JWTs are signed with RSA, there is a private and public key pair. The private key is located on the OAuth2 server and is hidden from you. Schema Registry uses the public key for validating the signature of the JWT token.

When JWTs are signed with HMAC, there is a secret which is shared by all parties. The secret is used for signing the token and also for verifying it.

Schema Registry supports the following ways to store the public key or the secret:

- JWK

  JSON Web Key is a data structure that describes a key. When you have multiple keys collected in a set, that data structure is named JWKS. A JWKS contains a collection of keys.

  Usually, there is a public web service that exposes the JWKS. You can obtain the JWKS through an HTTP request. Other transportation methods are possible, for example, the keys can be stored in a file or on a network storage.

  The keys are usually short lived (depending on the provider the validity period ranges from one day to one week). For this reason, Schema Registry runs a thread every 5 minutes to refresh the keys. The interval can be customized.
- Keystore

  The keys can be stored in a Java keystore file. You need to ensure that Schema Registry has access to the file and permission to read the key.
- Property

  The public key or secret can be stored directly in Schema Registry. In this case, you enter the key in Cloudera Manager and Schema Registry loads it during startup. This option is useful when the public key expires rarely and you do not want to depend on an external JWK service for managing the keys.

Authentication using OAuth2 with Kerberos

It is possible to have both Kerberos and OAuth2 enabled at the same time for Schema Registry.

OAuth2 is added as yet another authentication layer to Schema Registry. It is possible to have both Kerberos and OAuth2 enabled at the same time. In this scenario, if either one of them succeeds in authenticating the client, the client is given a pass.

This setup can be useful for cases when you have different services communicating with Schema Registry and some of them use Kerberos while others rely on OAuth2.

Schema Registry server configuration

Learn how to configure general settings for Schema Registry server. Also learn about the extra parameters which you can set when storage type is JWK, keystore, or property.

General settings

Property	Data type	Description
schema.registry.oauth.enabled	Boolean	Select this option to enable OAuth2 authentication.
schema.registry.oauth.key.store.type	Enum	Select the type of the key storage where the public key is read from. Possible values are: property, keystore, jwk. Depending on the chosen value, additional configuration might be necessary. These are detailed in the following sections.
schema.registry.oauth.jwt.principal.claim.name	String	The JWT token needs to contain the principal which is used during Ranger authorization. By default, it is assumed that the sub claim contains the principal, but this can be modified with this parameter.
schema.registry.oauth.jwt.expected.audience	String	The JWT token can optionally contain an audience aud claim. When this claim is present, the same audience value needs to be expected on the server side, otherwise the token is considered invalid.
schema.registry.oauth.jwt.expected.issuer	String	The JWT token can optionally contain an issuer iss claim. You can configure Schema Registry to only accept tokens issued by a specific issuer.
schema.registry.oauth.clock.skew	Integer	The clock of the server issuing the token might not be in sync with the clock where Schema Registry is running. You can adjust this value to tolerate a certain difference between the two clocks (in seconds).

JWK configuration settings

When storage type is JSON Web Key (JWK), you can also apply the following parameters.

Property	Data type	Description
schema.registry.oauth.jwks.url	String	URL to the server issuing the JWK keys. This can also be a local file if the URL starts with file://.
schema.registry.oauth.jwks.refresh.ms	Long	Refresh interval for reading the keys from the JWK server. Default value is 30000 ms (30 seconds).

The following parameters are optional. When the keys are downloaded from a remote server, you might need special configuration for accessing the server.

Property	Data type	Description
schema.registry.oauth.jwks.httpClient.basic.user	String	If the JWK server requires basic authentication, then you can provide the username.

Property	Data type	Description
schema.registry.oauth.jwks.httpClient.basic.password	String	If the JWK server requires basic authentication, then you can provide the password.
schema.registry.oauth.jwks.httpClient.keyStorePath	String	If a key is required for accessing the JWK server, then you can provide the keystore path.
schema.registry.oauth.jwks.httpClient.keyStoreType	String	Schema Registry keystore type of HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.keyPassword	String	Schema Registry key password of HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.keyStorePassword	String	Schema Registry keystore password of HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.keyStoreProvider	String	Schema Registry keystore provider of HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.keyManagerFactoryAlgorithm	String	Schema Registry algorithm of KeyManagerFactory for HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.keyManagerFactoryProvider	String	Schema Registry KeyManagerFactory provider for HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.trustStorePath	String	You can add the certificate of the JWK server to a truststore.
schema.registry.oauth.jwks.httpClient.trustStoreType	String	Schema Registry truststore type of HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.trustStorePassword	String	Schema Registry truststore password of HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.trustStoreProvider	String	Schema Registry truststore provider of HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.trustManagerFactoryAlgorithm	String	Schema Registry TrustManagerFactory algorithm for HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.trustManagerFactoryProvider	String	Schema Registry TrustManagerFactory provider for HTTP client used for JWK OAuth2. This can be required when keystore type is jwk.
schema.registry.oauth.jwks.httpClient.protocol	String	HTTPS security protocol. By default, it is TLS.

Keystore configuration settings

When storage type is keystore, you can also apply the following parameters.

Property	Data type	Description
schema.registry.oauth.keystore.public.key.keystore	String	Path to the keystore file. Ensure the file is readable by Schema Registry.
schema.registry.oauth.keystore.public.key.keystore.alias	String	The alias of the key within the keystore.

Property	Data type	Description
schema.registry.oauth.keystore.public.key.keystore.pass	String password	Password for reading the keystore.

Property configuration settings

When storage type is property, you can also apply the following parameters.

Property	Data type	Description
schema.registry.oauth.property.public.key.property	String	The public key or the secret.
schema.registry.oauth.property.key.algorithm	Enum	The algorithm of the key. The values are: RS256, HS256.

Configuring the Schema Registry client

Learn how to configure the Schema Registry client to access the server.

When running together with Kafka, the existing parameters still apply.

The client first sends a request to the OAuth2 auth server and requests a token. Configure the settings required to access the server.

Property	Data type	Description
schema.registry.auth.type	String	It needs to be set to oauth2.
schema.registry.oauth.server.url	String	URL of the server issuing the tokens.
schema.registry.oauth.client.id	String	ID of the client.
schema.registry.oauth.secret	String	Secret.
schema.registry.oauth.scope	String	Scope (optional).
schema.registry.oauth.request.method	String	HTTP request method. By default, it is post.

Cloudera Runtime 7.2.18

Deployment Planning for Cloudera Search

Date published: 2015-05-05
Date modified: 2024-04-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Things to consider before deploying Cloudera Search.....4

Dimensioning guidelines.....4

Schemaless mode overview and best practices.....7

Advantages of defining a schema for production use.....8

Things to consider before deploying Cloudera Search

Deployment planning for Cloudera Search is a complex procedure, covering several aspects from resource considerations to defining a schema.

Dimensioning guidelines

Think about the type of workload your future Solr deployment is expected to handle and plan resources accordingly.You need to consider several aspects from the amount of data ingested and the frequency of updates, through the number and complexity of concurrent queries to security and resilience.

Cloudera Search provides interactive search and scalable indexing. Before you begin installing Cloudera Search:

- Select the machines on which you want to install Cloudera Search, as well as any other services you are collocating with Search.
- Identify the tasks and workloads you need to manage and the types of data you need to search. This information can help guide the deployment process.

Important: Cloudera Search does not support contrib modules, such as DataImportHandler.

Spark, MapReduce and Lily HBase indexers are not contrib modules but part of the Cloudera Search product itself, therefore they are supported.

Sizing considerations

Default Cloudera deployments use a Java virtual machine (JVM) size of 1 GB, but this is insufficient for most use cases. Consider the following when determining an optimal JVM size for production usage:

- The more searchable data you have, the more memory you need. In general, 10 TB of searchable data requires more memory than 1 TB of searchable data.
- Indexing more fields requires more memory. For example, indexing all fields in a collection of logs, email messages, or Wikipedia entries requires more memory than indexing only the Date Created field.
- If the system must be stable and respond quickly, more memory might help. If slow responses are acceptable, you might be able to use less memory.

Consider the information given here as guidance instead of absolute requirements.

Table 1: Solr sizing guidelines

Java Heap	CPU	Disk
• Small workloads, or evaluations: 16 GB • Smaller production environments: 32 GB • Larger production environments: 96 GB is sufficient for most clusters. Set this value using the Java Heap Size of Solr Server in Bytes Solr configuration property.	• Minimum: 4 • Recommended: 16 for production workloads	No requirement if Solr uses HDFS for storage. If Solr uses local file system: • Faster disks, such as SSD can provide a significant performance improvement. • Occasionally a node may need disk space equal to 2-2.5 times the size of shards on the node for storage and overhead (for segment merging and shard recovery). • For heavy ingest and query loads Solr typically performs best on dedicated nodes, partially due to available OS cache for Solr files.

Define the use case

To determine how best to deploy Search in your environment, define use cases. You can use a sample application to benchmark different use cases and data types and sizes to help you identify the most important performance factors.

The same Solr index can have different hardware requirements, depending on the types of queries performed. The most common variation in hardware requirements is memory. For example, the memory requirements for faceting vary depending on the number of unique terms in the faceted field. Suppose you want to use faceting on a field that has 10 unique values. In this case, only 10 logical containers are required for counting. No matter how many documents are in the index, memory overhead is almost nonexistent.

Conversely, the same index could have unique timestamps for every entry, and you want to facet on that field with a : -type query. In this case, each index requires its own logical container. With this organization, if you had a large number of documents—500 million, for example—then faceting across 10 fields would increase the RAM requirements significantly.

For this reason, you must consider use cases and data characteristics before you can estimate hardware requirements. Important parameters to consider are:

- Number of documents. For Cloudera Search, sharding is almost always required.
- Approximate word count for each potential field.
- What information is stored in the Solr index and what information is only for searching? Information stored in the index is returned with the search results.
- Foreign language support:
  - How many different languages appear in your data?
  - What percentage of documents are in each language?
  - Is language-specific search supported? This determines whether accent folding and storing the text in a single field is sufficient.
  - What language families will be searched? For example, you could combine all Western European languages into a single field, but combining English and Chinese into a single field is not practical. Even with similar languages, using a single field for different languages can be problematic. For example, sometimes accents alter the meaning of a word, and in such cases, accent folding loses important distinctions.
- Faceting requirements:
  - Be wary of faceting on fields that have many unique terms. For example, faceting on timestamps or free-text fields typically has a high cost. Faceting on a field with more than 10,000 unique values is typically not useful. Ensure that any such faceting requirement is necessary.
  - What types of facets are needed? You can facet on queries as well as field values. Faceting on queries is often useful for dates. For example, “in the last day” or “in the last week” can be valuable. Using Solr Date Math to facet on a bare “NOW” is almost always inefficient. Facet-by-query is not memory-intensive because the number of logical containers is limited by the number of queries specified, no matter how many unique values are in the underlying field. This can enable faceting on fields that contain information such as dates or times, while avoiding the problem described for faceting on fields with unique terms.
- Sorting requirements:
  - Sorting requires one integer for each document (maxDoc), which can take up significant memory. Additionally, sorting on strings requires storing each unique string value.
- Paging requirements. End users rarely look beyond the first few pages of search results. For use cases requiring deep paging (paging through a large number of results), using cursors can improve performance and resource utilization. For more information, see Pagination of Results on the Apache Solr wiki.
- Is advanced search capability planned? If so, how will it be implemented? Significant design decisions depend on user requirements:
  - Can users be expected to learn about the system? Advanced screens can intimidate e-commerce users, but these screens can be more effective if users can be expected to learn them.
  - How long will users wait for results? Data mining or other design requirements can affect response times.
- How many simultaneous users must your system accommodate?

- Update requirements. An update in Solr refers both to adding new documents and changing existing documents:
  - Loading new documents:
    - Bulk. Will the index be rebuilt from scratch periodically, or will there only be an initial load?
    - Incremental. At what rate will new documents enter the system?
  - Updating documents:
    - Can you characterize the expected number of modifications to existing documents?
    - How much latency is acceptable between when a document is added to Solr and when it is available in Search results?
- Security requirements. Solr has no built-in security options, although Cloudera Search supports authentication using Kerberos. In Solr, document-level security is usually best accomplished by indexing authorization tokens with the document. The number of authorization tokens applied to a document is largely irrelevant; for example, thousands are reasonable but can be difficult to administer. The number of authorization tokens associated with a particular user should be no more than 100 in most cases. Security at this level is often enforced by appending an fq clause to the query, and adding thousands of tokens in an fq clause is expensive.
  - A post filter, also known as a no-cache filter, can help with access schemes that cannot use an fq clause. These are not cached and are applied only after all less-expensive filters are applied.
  - If grouping, faceting is not required to accurately reflect true document counts, so you can use some shortcuts. For example, ACL filtering is expensive in some systems, sometimes requiring database access. If completely accurate faceting is required, you must completely process the list to reflect accurate facets.
- Required query rate, usually measured in queries-per-second (QPS):
  - At a minimum, deploy machines with sufficient hardware resources to provide an acceptable response rate for a single user. Some types of queries can utilize the system so much that performance for even a small number of users is unacceptable. In this case, resharding can help.
  - If QPS is only somewhat lower than required and you do not want to reshard, you can often improve performance by adding replicas to each shard.
  - As the number of shards in your deployment increases, the general QPS rate starts to slowly decrease. This typically occurs when there are hundreds of shards.

Lucene index size limitation

A Solr collection can consist of multiple Lucene indexes because of sharding and replication. Splitting a collection into multiple shards (based on the order of magnitude of the number of documents to be indexed) is a very common setup to scale out Solr.

This is illustrated in figure *Solr scaling*. Each shard and replica represents a Lucene index. One such component can contain a maximum of ~2 billion documents.

Figure 1: Solr scaling

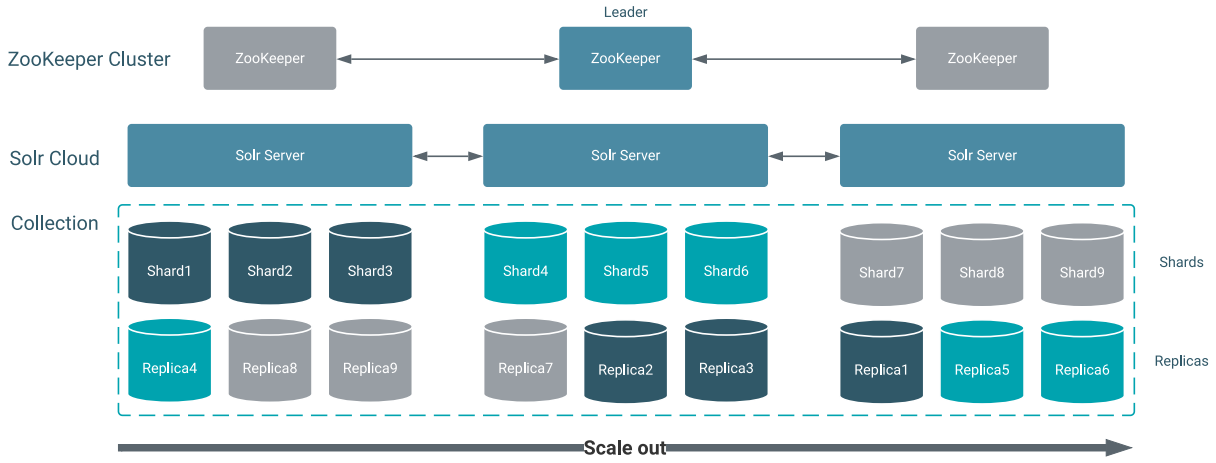

This allows Solr to handle more than 2 billion documents. Although for collections with billions of documents, there is a good chance that you will run into some other bottlenecks that require further tuning; especially if the size of each document is large or the schema is complex.

Related Information

Package org.apache.lucene.codecs.lucene40 Limitations

Schemaless mode overview and best practices

Schemaless mode removes the need to design a schema before beginning to use Cloudera Search. This can help you begin using Search more quickly, but schemaless mode is typically less efficient and effective than using a deliberately designed schema.

Note: Cloudera recommends pre-defining a schema before moving to production.

With the default non-schemaless mode, you create a schema by writing a schema.xml file before loading data into Solr so it can be used by Cloudera Search. You typically write a different schema definition for each type of data being ingested, because the different data types usually have different field names and values. Writing a custom schema is done by examining the data to be imported so its structure can be understood, and then creating a schema that accommodates that data. For example, emails might have a field for recipients and log files might have a field for IP addresses for machines reporting errors. Conversely, emails typically do not have an IP address field and log files typically do not have recipients. Therefore, the schema you use to import emails is different from the schema you use to import log files.

Cloudera Search offers schemaless mode to help facilitate sample deployments without the need to pre-define a schema. While schemaless is not suitable for production environments, it can help demonstrate the functionality and features of Cloudera Search. Schemaless mode operates based on three principles:

1. The schema is automatically updated using an API. When not using schemaless mode, users manually modify the schema.xml file or use the Schema API.
2. As data is ingested, it is analyzed and a guess is made about the type of data in the field. Supported types include Boolean, Integer, Long, Float, Double, Date, and Text.
3. When a new field is encountered, the schema is automatically updated using the API. The update is based on the guess about the type of data in the field.

For example, if schemaless encounters a field that contains "6.022", this would be determined to be type Float, whereas "Mon May 04 09:51:52 CDT 2009" would be determined to be type Date.

By combining these techniques, Schemaless:

1. Starts without a populated schema.
2. Intakes and analyzes data.
3. Modifies the schema based on guesses about the data.
4. Ingests the data so it can be searched based on the schema updates.

To generate a configuration for use in Schemaless mode, use solrctl instancedir --generate path -schemaless. Then, create the instancedir and collection as with non-schemaless mode.

Note:

The Browse and Spell Request Handlers require certain fields to be present in the schema. Since those fields cannot be guaranteed to exist in a Schemaless setup, the Browse and Spell Request Handlers are not enabled by default.

If you require the Browse and Spell Request Handlers, add them to the solrconfig.xml configuration file. Generate a non-schemaless configuration to see the usual settings and modify the required fields to fit your schema.

Best practices

Give each collection its own unique Instancedir

Solr supports using the same instancedir for multiple collections. In schemaless mode, automatic schema field additions actually change the underlying instancedir. Thus, if two collections are using the same instancedir, schema field additions meant for one collection will actually affect the other one as well. Therefore, Cloudera recommends that each collection have its own instancedir.

Advantages of defining a schema for production use

Although schemaless mode facilitates quick adoption and exploratory use, you are recommended to implement a schema for production use of Cloudera Search. This ensures that ingestion of data is consistent with expectations and query results are reliable.

Schemaless Solr is useful for getting started quickly and for understanding the underlying structure of the data you want to search. However, Schemaless Solr is not recommended for production use cases. Because the schema is automatically generated, a mistake like misspelling the name of the field alters the schema, rather than producing an error. The mistake may not be caught until much later and once caught, may require re-indexing to fix. Also, an unexpected input format may cause the type guessing to pick a field type that is incompatible with data that is subsequently ingested, preventing further ingestion until the incompatibility is manually addressed. Such a case is rare, but could occur. For example, if the first instance of a field was an integer, such as '9', but subsequent entries were text such as '10 Spring Street', the schema would make it impossible to properly ingest those subsequent entries. Therefore, Schemaless Solr may be useful for deciding on a schema during the exploratory stage of development, but Cloudera recommends defining the schema in the traditional way before moving to production.

When constructing a schema, use data types that most accurately describe the data that the fields will contain. For example:

- Use the tdate type for dates. Do this instead of representing dates as strings.
- Consider using the text type that applies to your language, instead of using String. For example, you might use text_en. Text types support returning results for subsets of an entry. For example, querying on "john" would find "John Smith", whereas with the string type, only exact matches are returned.
- For IDs, use the string type.

Cloudera Runtime 7.2.18

Cloudera Search ETL Using Morphlines

Date published: 2019-11-19
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Extracting, transforming, and loading data with Cloudera Morphlines.....4

Using Morphlines to index Avro.....6

Using Morphlines with Syslog.....9

Extracting, transforming, and loading data with Cloudera Morphlines

Cloudera Morphlines is an open-source framework that reduces the time and skills required to build or change Search indexing applications.

A morphline is a rich configuration file that simplifies defining an extraction, transformation, and loading (ETL) transformation chain. Use these chains to consume any kind of data from any data source, process the data, and load the results into Cloudera Search. Executing in a small, embeddable Java runtime system, morphlines can be used for near real-time applications as well as batch processing applications. The following diagram shows the process flow:

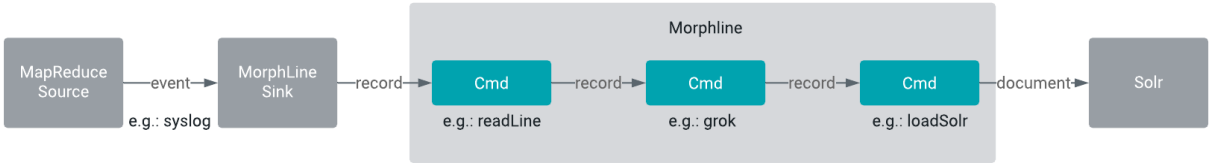

Morphlines can be seen as an evolution of Unix pipelines, where the data model is generalized to work with streams of generic records, including arbitrary binary payloads. Morphlines can be embedded into Hadoop components such as Search, MapReduce, Hive, and Sqoop.

The framework ships with a set of frequently used high-level transformation and I/O commands that can be combined in application-specific ways. The plug-in system allows you to add new transformations and I/O commands and integrates existing functionality and third-party systems.

This integration enables the following:

- Rapid Hadoop ETL application prototyping
- Complex stream and event processing in real time
- Flexible log file analysis
- Integration of multiple heterogeneous input schemas and file formats
- Reuse of ETL logic building blocks across Search applications

The high-performance Cloudera Runtime compiles a morphline, processing all commands for a morphline in the same thread and adding no artificial overhead. For high scalability, you can deploy many morphline instances on a cluster in many MapReduce tasks.

The following components run morphlines:

- MapReduceIndexerTool
- Lily HBase Indexer

Cloudera also provides a corresponding Cloudera Search Tutorial.

Data Morphlines support

Morphlines manipulate continuous or arbitrarily large streams of records. The data model can be described as follows: A record is a set of named fields where each field has an ordered list of one or more values. A value can be any Java Object. That is, a record is essentially a hash table where each hash table entry contains a String key and a list of Java Objects as values. (The implementation uses Guava's ArrayListMultimap, which is a ListMultimap). Note that a field can have multiple values and any two records need not use common field names. This flexible data model corresponds exactly to the characteristics of the Solr/Lucene data model, meaning a record can be seen as a SolrInpu tDocument. A field with zero values is removed from the record - fields with zero values effectively do not exist.

Not only structured data, but also arbitrary binary data can be passed into and processed by a morphline. By convention, a record can contain an optional field named _attachment_body, which can be a Java java.io.InputStream

or Java byte[]. Optionally, such binary input data can be characterized in more detail by setting the fields named _attachment_mimetype (such as application/pdf) and _attachment_charset (such as UTF-8) and _attachment_name (such as cars.pdf), which assists in detecting and parsing the data type.

This generic data model is useful to support a wide range of applications.

Important: Cloudera Search does not support contrib modules, such as DataImportHandler.

Spark, MapReduce and Lily HBase indexers are not contrib modules but part of the Cloudera Search product itself, therefore they are supported.

How Morphlines act on data

A command transforms a record into zero or more records. Commands can access all record fields. For example, commands can parse fields, set fields, remove fields, rename fields, find and replace values, split a field into multiple fields, split a field into multiple values, or drop records. Often, regular expression based pattern matching is used as part of the process of acting on fields. The output records of a command are passed to the next command in the chain. A command has a Boolean return code, indicating success or failure.

For example, consider the case of a multi-line input record: A command could take this multi-line input record and divide the single record into multiple output records, one for each line. This output could then later be further divided using regular expression commands, splitting each single line record out into multiple fields in application specific ways.

A command can extract, clean, transform, join, integrate, enrich and decorate records in many other ways. For example, a command can join records with external data sources such as relational databases, key-value stores, local files or IP Geo lookup tables. It can also perform tasks such as DNS resolution, expand shortened URLs, fetch linked metadata from social networks, perform sentiment analysis and annotate the record accordingly, continuously maintain statistics for analytics over sliding windows, compute exact or approximate distinct values and quantiles.

A command can also consume records and pass them to external systems. For example, a command can load records into Solr or write them to a MapReduce Reducer or pass them into an online dashboard. The following diagram illustrates some pathways along which data might flow with the help of morphlines:

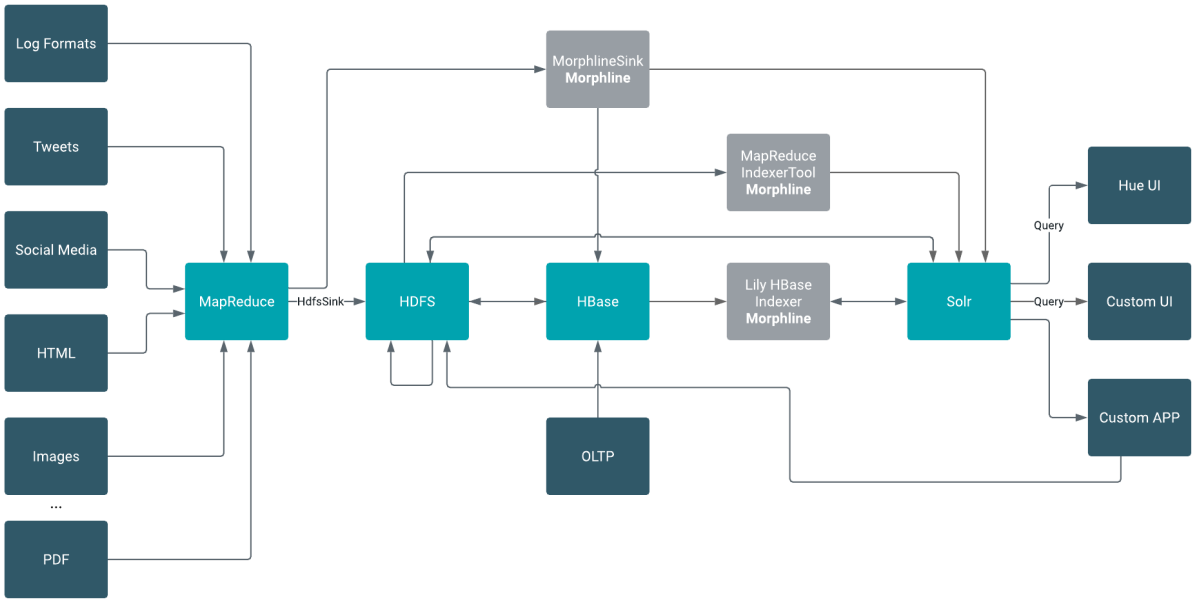

Morphline characteristics

A command can contain nested commands. Thus, a morphline is a tree of commands, akin to a push-based data flow engine or operator tree in DBMS query execution engines.

A morphline has no notion of persistence, durability, distributed computing, or host failover. A morphline is basically just a chain of in-memory transformations in the current thread. There is no need for a morphline to manage multiple processes, hosts, or threads because this is already addressed by host systems such as MapReduce or Storm. However, a morphline does support passing notifications on the control plane to command subtrees. Such notifications include BEGIN_TRANSACTION, COMMIT_TRANSACTION, ROLLBACK_TRANSACTION, SHUTDOWN.

The morphline configuration file is implemented using the HOCON format (Human-Optimized Config Object Notation). HOCON is basically JSON slightly adjusted for configuration file use cases. HOCON syntax is defined at HOCON github page and is also used by Akka and Play.

How Morphlines are implemented

Cloudera Search includes several maven modules that contain morphline commands for integration with Apache Solr including SolrCloud, flexible log file analysis, single-line records, multi-line records, CSV files, regular expression based pattern matching and extraction, operations on record fields for assignment and comparison, operations on record fields with list and set semantics, if-then-else conditionals, string and timestamp conversions, scripting support for dynamic Java code, a small rules engine, logging, metrics and counters, integration with Avro, integration with Apache Tika parsers, integration with Apache Hadoop Sequence Files, auto-detection of MIME types from binary data using Apache Tika, and decompression and unpacking of arbitrarily nested container file formats, among others.

Using Morphlines to index Avro

This example illustrates using a morphline to index an Avro file with a schema.

1. View the content of the Avro file to understand the data:

```
$ wget http://archive.apache.org/dist/avro/avro-1.7.4/java/avro-tools-1.
7.4.jar
$ java -jar avro-tools-1.7.4.jar tojson \
/opt/cloudera/parcels/CDH/share/doc/search-*/examples/test-documents/sampl
e-statuses-20120906-141433.avro
```

2. Inspect the schema of the Avro file:

```
$ java -jar avro-tools-1.7.4.jar getschema /opt/cloudera/parcels/CDH/sha
re/doc/search-*/examples/test-documents/sample-statuses-20120906-141433.
avro

{
  "type" : "record",
  "name" : "Doc",
  "doc" : "adoc",
  "fields" : [ {
    "name" : "id",
    "type" : "string"
  }, {
    "name" : "user_statuses_count",
    "type" : [ "int", "null" ]
  }, {
    "name" : "user_screen_name",
    "type" : [ "string", "null" ]
  }, {
    "name" : "created_at",
    "type" : [ "string", "null" ]
  }, {
    "name" : "text",
    "type" : [ "string", "null" ]
  }
```

```
  ...
  ]
}
```

3. Extract the id, user_screen_name, created_at, and text fields from the Avro records, and then store and index them in Solr, by adding the username, created_at, and text field definitions to the default managed-schema:

```
<fields>
  <field name="id" type="string" indexed="true" stored="true" required=
"true" multiValued="false" />
  <field name="username" type="text_en" indexed="true" stored="true" />
  <field name="created_at" type="tdate" indexed="true" stored="true" />
  <field name="text" type="text_en" indexed="true" stored="true" />

  <field name="_version_" type="long" indexed="true" stored="true"/>
  <dynamicField name="ignored_*" type="ignored"/>
</fields>
```

The Solr output schema omits some Avro input fields, such as user_statuses_count. If your data includes Avro input fields that are not included in the Solr output schema, you may want to make changes to data as it is ingested. For example, suppose you need to rename the input field user_screen_name to the output field username. Also suppose that the time format for the created_at field is yyyy-MM-dd'T'HH:mm:ss'Z'. Finally, suppose any unknown fields present are to be removed. Recall that Solr throws an exception on any attempt to load a document that contains a field that is not specified in managed-schema.

4. These transformation rules that make it possible to modify data so it fits your particular schema can be expressed with morphline commands called `readAvroContainer`, `extractAvroPaths`, `convertTimeStamp`, `sanitizeUnknownSolrFields`, and `sanitizeUnknownSolrFields` by editing a morphline.conf file.

```
# Specify server locations in a SOLR_LOCATOR variable; used later in
# variable substitutions:
SOLR_LOCATOR : {
  # Name of solr collection
  collection : collection1

  # ZooKeeper ensemble
  zkHost : "127.0.0.1:2181/solr"
}

# Specify an array of one or more morphlines, each of which defines an ETL
# transformation chain. A morphline consists of one or more potentially
# nested commands. A morphline is a way to consume records such as Flume e
vents,
# HDFS files or blocks, turn them into a stream of records, and pipe the
stream
# of records through a set of easily configurable transformations on its
way to
# Solr.
morphlines : [
  {
    # Name used to identify a morphline. For example, used if there are mu
ltiple
    # morphlines in a morphline config file.
    id : morphline1
    # Import all morphline commands in these java packages and their su
bpackages.
    # Other commands that may be present on the classpath are not visible
 to this
    # morphline.
    importCommands : ["org.kitesdk.**", "org.apache.solr.**"]
    commands : [
      {
        # Parse Avro container file and emit a record for each Avro object
```

```
        readAvroContainer {
          # Optionally, require the input to match one of these MIME ty
pes:
          # supportedMimeTypes : [avro/binary]
          # Optionally, use a custom Avro schema in JSON format inline:
          # readerSchemaString : """<json can go here>"""

          # Optionally, use a custom Avro schema file in JSON format:
          # readerSchemaFile : /path/to/syslog.avsc
        }
      }
      {
        # Consume the output record of the previous command and pipe an
other
        # record downstream.
        #
        # extractAvroPaths is a command that uses zero or more Avro path
        # excodeblockssions to extract values from an Avro object. Each ex
codeblockssion
        # consists of a record output field name, which appears to the le
ft of the
        # colon ':' and zero or more path steps, which appear to the right
.
        # Each path step is separated by a '/' slash. Avro arrays are
        # traversed with the '[]' notation.
        #
        # The result of a path excodeblockssion is a list of objects, each
 of which
        # is added to the given record output field.
        #
        # The path language supports all Avro concepts, including nested
        # structures, records, arrays, maps, unions, and others, as well a
s a flatten
        # option that collects the primitives in a subtree into a flat l
ist. In the
        # paths specification, entries on the left of the colon are the
target Solr
        # field and entries on the right specify the Avro source paths.
Paths are read
        # from the source that is named to the right of the colon and wri
tten to the
        # field that is named on the left.
        extractAvroPaths {
          flatten : false
          paths : {
            id : /id
            username : /user_screen_name
            created_at : /created_at
            text : /text
          }
        }
      }

      # Consume the output record of the previous command and pipe another
      # record downstream.
      #
      # convert timestamp field to native Solr timestamp format
      # such as 2012-09-06T07:14:34Z to 2012-09-06T07:14:34.000Z
      {
        convertTimestamp {
          field : created_at
          inputFormats : ["yyyy-MM-dd'T'HH:mm:ss'Z'", "yyyy-MM-dd"]
          inputTimezone : America/Los_Angeles
          outputFormat : "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'"
```

```
          outputTimezone : UTC
        }
      }

      # Consume the output record of the previous command and pipe another
      # record downstream.
      #
      # This command deletes record fields that are unknown to Solr
      # managed-schema.
      #
      # Recall that Solr throws an exception on any attempt to load a do
cument
      # that contains a field that is not specified in managed-schema.
      {
        sanitizeUnknownSolrFields {
          # Location from which to fetch Solr schema
          solrLocator : ${SOLR_LOCATOR}
        }
      }

      # log the record at DEBUG level to SLF4J
      { logDebug { format : "output record: {}", args : ["@{}"] } }

      # load the record into a Solr server or MapReduce Reducer
      {
        loadSolr {
          solrLocator : ${SOLR_LOCATOR}
        }
      }
    ]
  }
]
```

Related Information

Morphlines Reference

Using Morphlines with Syslog

This example illustrates using a morphline to extract information from a syslog file.

A syslog file contains semi-structured lines of the following form:

```
<164>Feb  4 10:46:14 syslog sshd[607]: listening on 0.0.0.0 port 22.
```

The program extracts the following record from the log line and loads it into Solr:

```
syslog_pri:164
syslog_timestamp:Feb  4 10:46:14
syslog_hostname:syslog
syslog_program:sshd
syslog_pid:607
syslog_message:listening on 0.0.0.0 port 22.
```

Use the following rules to create a chain of transformation commands, which are expressed with the readLine, grok, and logDebug morphline commands, by editing a morphline.conf file.

```
# Specify server locations in a SOLR_LOCATOR variable; used later in
# variable substitutions:
SOLR_LOCATOR : {
```

```
  # Name of solr collection
  collection : collection1

  # ZooKeeper ensemble
  zkHost : "127.0.0.1:2181/solr"
}

# Specify an array of one or more morphlines, each of which defines an ETL
# transformation chain. A morphline consists of one or more potentially
# nested commands. A morphline is a way to consume records such as Flume e
vents,
# HDFS files or blocks, turn them into a stream of records, and pipe the
stream
# of records through a set of easily configurable transformations on the
way to
# a target application such as Solr.
morphlines : [
  {
    id : morphline1
    importCommands : ["org.kitesdk.**"]

    commands : [
      {
        readLine {
          charset : UTF-8
        }
      }
      {
        grok {
          # a grok-dictionary is a config file that contains prefabricated r
egular expressions
          # that can be referred to by name.
          # grok patterns specify such a regex name, plus an optional output
 field name.
          # The syntax is %{REGEX_NAME:OUTPUT_FIELD_NAME}
          # The input line is expected in the "message" input field.
          dictionaryFiles : [target/test-classes/grok-dictionaries]
          expressions : {
            message : """<%{POSINT:syslog_pri}>%{SYSLOGTIMESTAMP:syslog_ti
mestamp} %{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:\[%{POSINT:s
yslog_pid}\])?: %{GREEDYDATA:syslog_message}"""
          }
        }
      }

      # Consume the output record of the previous command and pipe another
      # record downstream.
      #
      # This command deletes record fields that are unknown to Solr
      # managed-schema.
      #
      # Recall that Solr throws an exception on any attempt to load a docum
ent
      # that contains a field that is not specified in managed-schema.
      {
        sanitizeUnknownSolrFields {
          # Location from which to fetch Solr schema
          solrLocator : ${SOLR_LOCATOR}
        }
      }

      # log the record at DEBUG level to SLF4J
      { logDebug { format : "output record: {}", args : ["@{}"] } }
```

```
      # load the record into a Solr server or MapReduce Reducer
      {
        loadSolr {
          solrLocator : ${SOLR_LOCATOR}
        }
      }
    ]
  }
]
```

Related Information

Morphlines Reference

Cloudera Runtime 7.2.18

Indexing Data Using Morphlines

Date published: 2019-11-19
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Indexing data using Morphlines 4

Lily HBase near real time indexing using Morphlines 4
- Adding the Lily HBase indexer service 4
- Starting the Lily HBase NRT indexer service 5
- Using the Lily HBase NRT indexer service 5
  - Enable replication on HBase column families 5
  - Create a Collection in Cloudera Search 5
  - Creating a Lily HBase Indexer Configuration File 6
  - Creating a Morphline Configuration File 6
  - Understanding the extractHBaseCells Morphline Command 7
  - Registering a Lily HBase Indexer Configuration with the Lily HBase Indexer Service 8
  - Verifying that Indexing Works 9
  - Using the indexer HTTP interface 10
- Configuring Lily HBase Indexer Security 10
  - Configure Lily HBase Indexer to use TLS/SSL 10
  - Configure Lily HBase Indexer Service to use Kerberos authentication 11

Batch indexing using Morphlines 11
- Spark indexing using morphlines 12
- MapReduce indexing 18
  - MapReduceIndexerTool 18
  - Lily HBase batch indexing for Cloudera Search 28

Indexing data using Morphlines

There are generally two approaches to indexing data using Cloudera Search:

1. Near real time (NRT) indexing
2. Batch indexing

Near real time indexing is generally used when new data needs to be returned in query results in time frames measured in seconds, whereas batch indexing is useful for situations where large amounts of data is indexed at regular intervals, or for indexing a new dataset for the first time.

Near real time indexing generally uses a framework such as Apache Kafka to continuously ingest and index data. The Lily HBase Indexer can also be used for NRT indexing on Apache HBase tables.

Batch indexing usually relies on MapReduce/YARN jobs to periodically index large datasets. The Lily HBase Indexer can also be used for batch indexing HBase tables.

Related Concepts

Lily HBase near real time indexing using Morphlines

Batch indexing using Morphlines

Lily HBase near real time indexing using Morphlines

You can use the Lily HBase Indexer for near real time (NRT) indexing updates to HBase tables.

The Lily HBase NRT Indexer service is a flexible, scalable, fault-tolerant, transactional, NRT system for processing a continuous stream of HBase cell updates into live search indexes. Typically it takes seconds for data ingested into HBase to appear in search results; this duration is tunable. The Lily HBase Indexer uses SolrCloud to index data stored in HBase. As HBase applies inserts, updates, and deletes to HBase table cells, the indexer keeps Solr consistent with the HBase table contents, using standard HBase replication. The indexer supports flexible custom application-specific rules to extract, transform, and load HBase data into Solr. Solr search results can contain columnFamily:qualifier links back to the data stored in HBase. This way, applications can use the Search result set to directly access matching raw HBase cells. Indexing and searching do not affect operational stability or write throughput of HBase because the indexing and searching processes are separate and asynchronous to HBase.

To accommodate the HBase ingest load, you can run as many Lily HBase Indexer services on different hosts as required. Because the indexing work is shared by all indexers, you can scale the service by adding more indexers. The recommended number of indexer is 1 for each HBase RegionServer but in a High Availability environment five worker nodes is the minimum for acceptable performance and reliability. You can co-locate Lily HBase Indexer services with Solr servers on the same set of hosts. RegionServers can also be co-locate with Lily HBase Indexer on the same host to improve performance.

Note: Specific workloads and usage patterns might require additional fine-tuning beyond these general recommendations.

The Lily HBase NRT Indexer service must be deployed in an environment with a running HBase cluster, a running SolrCloud cluster (the Solr service in Cloudera Manager), and at least one ZooKeeper quorum.

Adding the Lily HBase indexer service

In Cloudera Manager, the Lily HBase Indexer service is called Key-Value Store Indexer, and the service role is called Lily HBase Indexer.

Starting the Lily HBase NRT indexer service

Use Cloudera Manager to start the Lily HBase indexer service.

You can use Cloudera Manager to start the Lily HBase Indexer Service (Key-Value Store Indexer service Actions Start).

Once the service is running, you can create and manage indexers.

Using the Lily HBase NRT indexer service

To index for column families of tables in an HBase cluster:

- Enable replication on HBase column families
- Create collections and configurations
- Register a Lily HBase Indexer configuration with the Lily HBase Indexer Service
- Verify that indexing is working

Enable replication on HBase column families

Apache HBase Replication allows you to copy data from one HBase cluster to a different and possibly distant HBase cluster. This can be used for disaster recovery or when you want to run load intensive MapReduce jobs on your HBase cluster.

Procedure

1. For every existing table, set the REPLICATION_SCOPE on every column family that you want to index:

```
hbase shell
hbase shell> disable 'sample_table'
hbase shell> alter 'sample_table', {NAME => 'columnfamily1', REPLICATION
_SCOPE => 1}
hbase shell> enable 'sample_table'
```

2. For every new table, set the REPLICATION_SCOPE on every column family that you want to index using a command such as the following:

```
hbase shell
hbase shell> create 'test_table', {NAME => 'testcolumnfamily', REPLICATI
ON_SCOPE => 1}
```

Create a Collection in Cloudera Search

About this task

A collection in Search used for HBase indexing must have a Solr schema that accommodates the types of HBase column families and qualifiers that are being indexed. To begin, consider adding the all-inclusive data field to a default schema.

Procedure

Once you decide on a schema, create a collection using commands similar to the following:

```
solrctl instancedir --generate $HOME/hbase_collection_config
## Edit $HOME/hbase_collection_config/conf/managed-schema as needed ##
solrctl config --upload hbase_collection_config $HOME/hbase_collection_confi
g
```

```
solrctl collection --create hbase_collection -s <numShards> -c hbase_colle
ction_config
```

Creating a Lily HBase Indexer Configuration File

About this task

Configure individual Lily HBase Indexers using the hbase-indexer command-line utility. Typically, there is one Lily HBase Indexer configuration file for each HBase table, but there can be as many Lily HBase Indexer configuration files as there are tables, column families, and corresponding collections in Search. Each Lily HBase Indexer configuration is defined in an XML file, such as morphline-hbase-mapper.xml.

An indexer configuration XML file must refer to the MorphlineResultToSolrMapper implementation and point to the location of a Morphline configuration file, as shown in the following morphline-hbase-mapper.xml indexer configuration file.

Procedure

- Set morphlineFile to the relative path morphlines.conf. Make sure the file is readable by the HBase system user (hbase by default).

```
$ cat $HOME/morphline-hbase-mapper.xml

<?xml version="1.0"?>
<indexer table="sample_table"
mapper="com.ngdata.hbaseindexer.morphline.MorphlineResultToSolrMapper">
   <!-- The relative path on the local file system to the
   morphline configuration file. -->

   <param name="morphlineFile" value="morphlines.conf"/>

   <!-- The optional morphlineId identifies a morphline if there are multi
ple
   morphlines in morphlines.conf -->
   <!-- <param name="morphlineId" value="morphline1"/> -->
</indexer>
```

  The Lily HBase Indexer configuration file also supports the standard attributes of any HBase Lily Indexer on the top-level <indexer> element. It does not support the <field> element and <extract> elements.

Creating a Morphline Configuration File

About this task

After creating an indexer configuration XML file, you can configure morphline ETL transformation commands in a morphlines.conf configuration file. The morphlines.conf configuration file can contain any number of morphline commands. Typically, an extractHBaseCells command is the first command. The readAvroContainer or readAvro morphline commands are often used to extract Avro data from the HBase byte array. This configuration file can be shared among different applications that use morphlines.

Note: To function properly, the morphline must not contain a loadSolr command. The Lily HBase Indexer must load documents into Solr, instead of the morphline itself.

Procedure

You can edit the morphlines.conf file within Cloudera Manager (Key-Value Store Indexer service Configuration Category Morphlines Morphlines File).

Understanding the extractHBaseCells Morphline Command

The extractHBaseCells morphline command extracts cells from an HBase result and transforms the values into a Solr InputDocument. The command consists of an array of zero or more mapping specifications.

Each mapping has:

- The inputColumn parameter, which specifies the data from HBase for populating a field in Solr. It has the form of a column family name and qualifier, separated by a colon. The qualifier portion can end in an asterisk, which is interpreted as a wildcard. In this case, all matching column-family and qualifier expressions are used. The following are examples of valid inputColumn values:
  - mycolumnfamily:myqualifier
  - mycolumnfamily:my*
  - mycolumnfamily:*
- The outputField parameter specifies the morphline record field to which to add output values. The morphline record field is also known as the Solr document field. Example: first_name.
- Dynamic output fields are enabled by the outputField parameter ending with a wildcard (*). For example:

  ```
  inputColumn : "mycolumnfamily:*"
  outputField : "belongs_to_*"
  ```

  In this case, if you make these puts in HBase:

  ```
  put 'table_name' , 'row1' , 'mycolumnfamily:1' , 'foo'
  put 'table_name' , 'row1' , 'mycolumnfamily:9' , 'bar'
  ```

  Then the fields of the Solr document are as follows:

  ```
  belongs_to_1 : foo
  belongs_to_9 : bar
  ```

- The type parameter defines the data type of the content in HBase. All input data is stored in HBase as byte arrays, but all content in Solr is indexed as text, so a method for converting byte arrays to the actual data type is required. The type parameter can be the name of a type that is supported by org.apache.hadoop.hbase.util.Bytes.to* (which currently includes byte[], int, long, string, boolean, float, double, short, and bigdecimal). Use type byte[] to pass the byte array through to the morphline without conversion.
  - type:byte[] copies the byte array unmodified into the record output field
  - type:int converts with org.apache.hadoop.hbase.util.Bytes.toInt
  - type:long converts with org.apache.hadoop.hbase.util.Bytes.toLong
  - type:string converts with org.apache.hadoop.hbase.util.Bytes.toString
  - type:boolean converts with org.apache.hadoop.hbase.util.Bytes.toBoolean
  - type:float converts with org.apache.hadoop.hbase.util.Bytes.toFloat
  - type:double converts with org.apache.hadoop.hbase.util.Bytes.toDouble
  - type:short converts with org.apache.hadoop.hbase.util.Bytes.toShort
  - type:bigdecimal converts with org.apache.hadoop.hbase.util.Bytes.toBigDecimal

  Alternatively, the type parameter can be the name of a Java class that implements the com.ngdata.hbaseindexer.parse.ByteArrayValueMapper interface.

  HBase data formatting does not always match what is specified by org.apache.hadoop.hbase.util.Bytes.*. For example, this can occur with data of type float or double. You can enable indexing of such HBase data by converting the data. There are various ways to do so, including:

  - Using Java morphline command to parse input data, converting it to the expected output. For example:

    ```
    {
     imports : "import java.util.*;" code: """ // manipulate the contents of
     a record field
    ```

```
 String stringAmount = (String) record.getFirstValue("amount");
 Double dbl = Double.parseDouble(stringAmount); record.replaceValues("
amount",dbl);
 return child.process(record); // pass record to next command in chain
 """
}
```

- Creating table fields with binary format and then using types such as double or float in a morphline.conf. You could create a table in HBase for storing doubles using commands similar to:

```
CREATE TABLE sample_lily_hbase ( id string, amount double, ts timestamp
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,ti:amount#b,ti:
ts,')
TBLPROPERTIES ('hbase.table.name' = 'sample_lily');
```

- The source parameter determines which portion of an HBase KeyValue is used as indexing input. Valid choices are value or qualifier. When value is specified, the HBase cell value is used as input for indexing. When qualifier is specified, then the HBase column qualifier is used as input for indexing. The default is value.

Registering a Lily HBase Indexer Configuration with the Lily HBase Indexer Service

About this task

When the content of the Lily HBase Indexer configuration XML file is satisfactory, register it with the Lily HBase Indexer Service. Register the Lily HBase Indexer configuration file by uploading the Lily HBase Indexer configuration XML file to ZooKeeper. For example:

Procedure

1. If your cluster has security enabled, create a Java Authentication and Authorization Service (JAAS) configuration file named jaas.conf in your home directory with the following contents:

```
Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=false
  useTicketCache=true
    principal="jdoe@EXAMPLE.COM";
};
```

   Replace jdoe@EXAMPLE.COM with your user principal. Your user account must have WRITE permission to create an indexer. For more information, see Configuring Lily HBase Indexer Security on page 10.

2. If your cluster has security enabled, authenticate with the user principal specified in your jaas.conf file:

```
kinit jdoe@EXAMPLE.COM
```

3. Run the following command to add the JAAS configuration to the system properties:

```
export HBASE_INDEXER_OPTS=-Djava.security.auth.login.config=jaas.conf
```

4. Run the following command to register your indexer configuration file with the indexer service:

```
hbase-indexer add-indexer \
--name myIndexer \
--indexer-conf $HOME/morphline-hbase-mapper.xml \
--connection-param solr.zk=zk01.example.com,zk02.example.com,zk03.examp
le.com/solr \
--connection-param solr.collection=hbase_collection \
```

```
--zookeeper zk01.example.com:2181,zk02.example.com:2181,zk03.example.com
:2181
```

5. Verify that the indexer was successfully created as follows:

```
hbase-indexer list-indexers -zookeeper zk01.example.com:2181,zk02.exampl
e.com:2181,zk03.example.com:2181
Number of indexes: 1

myIndexer
  + Lifecycle state: ACTIVE
  + Incremental indexing state: SUBSCRIBE_AND_CONSUME
  + Batch indexing state: INACTIVE
  + SEP subscription ID: Indexer_myIndexer
  + SEP subscription timestamp: 2013-06-12T11:23:35.635-07:00
  + Connection type: solr
  + Connection params:
    + solr.collection = hbase-collection1
    + solr.zk = localhost/solr
  + Indexer config:
      110 bytes, use -dump to see content
  + Batch index config:
      (none)
  + Default batch index config:
      (none)
  + Processes
    + 1 running processes
    + 0 failed processes
```

Use the update-indexer and delete-indexer command-line options of the hbase-indexer utility to manipulate existing Lily HBase Indexers.

For more help, use the following commands:

```
hbase-indexer add-indexer --help
hbase-indexer list-indexers --help
hbase-indexer update-indexer --help
hbase-indexer delete-indexer --help
```

Morphline configuration files can be changed without re-creating the indexer itself, but you must restart the Lily HBase Indexer service for the changes to take effect.

Verifying that Indexing Works

Procedure

1. Add rows to the indexed HBase table. For example:

```
hbase shell
hbase(main):001:0> put 'sample_table', 'row1', 'data', 'value'
hbase(main):002:0> put 'sample_table', 'row2', 'data', 'value2'
```

2. If the put operation succeeds, wait a few seconds, go to the SolrCloud UI query page, and query the data. Note the updated rows in Solr.

3. To print diagnostic information, such as the content of records as they pass through the morphline commands, enable the TRACE log level:
   a) Go to Key-Value Store Indexer service Configuration Category Advanced .
   b) Find the Lily HBase Indexer Logging Advanced Configuration Snippet (Safety Valve) property or search for it by typing its name in the Search box.
   c) Add the following to the text box:

```
log4j.logger.org.kitesdk.morphline=TRACE
log4j.logger.com.ngdata=TRACE
```

   d) Click Save Changes.
   e) Restart the service (Key-Value Store Indexer service Actions Restart).
4. Examine the log files in /var/log/hbase-solr/lily-hbase-indexer-* for details.

Using the indexer HTTP interface

Lily HBase Indexer includes an HTTP interface for the list-indexers, create-indexer, update-indexer, and delete-indexer commands.

This interface can be secured with Kerberos for authentication and Apache Ranger for authorization. For information on configuring security for the Lily HBase Indexer service, see Configuring Lily HBase Indexer Security on page 10.

By default, the hbase-indexer command line client does not use the HTTP interface. Use the HTTP interface to take advantage of the features it provides, such as Kerberos authentication and Ranger integration. The hbase-indexer command supports two additional parameters to the list-indexers, create-indexer, delete-indexer, and update-indexer commands:

- --http: An HTTP URI for the HTTP interface. By default, this URI is of the form http://lily01.example.com:11060/indexer/. If this parameter is specified, the Lily HBase Indexer uses the HTTP API. If this parameter is not specified, the indexer communicates directly with ZooKeeper.
- --jaas: Specifies a Java Authentication and Authorization Service (JAAS) configuration file. This is only necessary for Kerberos-enabled deployments.

Note: Make sure that you use fully qualified domain names (FQDN) when specifying hostnames for both the Lily HBase Indexer host and the ZooKeeper hosts. Using FQDNs helps ensure proper Kerberos realm mapping.

For example:

```
hbase-indexer list-indexers --http http://lily01.example.com:11060/indexer/
\
--jaas $HOME/jaas.conf --zookeeper zk01.example.com:2181,zk02.example.com:
2181,zk03.example.com:2181
```

Related Concepts

Configuring Lily HBase Indexer Security

Configuring Lily HBase Indexer Security

The Lily HBase Indexer includes an HTTP interface for the list-indexers, create-indexer, update-indexer, and delete-indexer commands. This interface can be secured with Kerberos for authentication and Apache Ranger for authorization.

Configure Lily HBase Indexer to use TLS/SSL

Although Cloudera recommends using AutoTLS, you also have the option to set up TLS manually for the Lily HBase Indexer.

About this task

To configure and enable Hadoop TLS/SSL for the Lily HBase Indexer (Key-Value Store Indexer) perform the following steps.

Procedure

1. Open the Cloudera Manager Admin Console and go to the Key-Value Store Indexer.
2. Click the Configuration tab.
3. Select Scope All .
4. Select Category All .
5. In the Search field, type TLS/SSL to show the Solr TLS/SSL properties.
6. Edit the following TLS/SSL properties according to your cluster configuration.

   Note: These values must be the same for all hosts running the Key-Value Store Indexer role.

   Table 1: Key-Value Store TLS/SSL Properties

Property	Description
HBase Indexer TLS/SSL Certificate Trust Store File	The location on disk of the truststore, in .jks format, used to confirm the authenticity of TLS/SSL servers that HBase Indexer might connect to. This is used when HBase Indexer is the client in a TLS/SSL connection. This truststore must contain the certificate(s) used to sign the service(s) being connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
HBase Indexer TLS/SSL Certificate Trust Store Password (Optional)	The password for the HBase Indexer TLS/SSL Certificate Trust Store File. This password is not required to access the truststore: this field can be left blank. This password provides optional integrity checking of the file. The contents of truststores are certificates, and certificates are public information.

7. Restart the service.

Configure Lily HBase Indexer Service to use Kerberos authentication

To enable Kerberos authentication for the Lily HBase Indexer service, perform the following steps.

Procedure

1. In the Cloudera Manager admin console, go to Key-Value Store Indexer service Configuration Category Security .
2. Select the kerberos option for HBase Indexer Secure Authentication.
3. Click Save Changes.
4. Go to Administration Security Kerberos Credentials .
5. Click Generate Missing Credentials.
6. Restart the indexer service (Key-Value Store Indexer service Actions Restart).

Batch indexing using Morphlines

Batch indexing usually relies on MapReduce/YARN or Spark jobs to periodically index large datasets, or to index new datasets for the first time. The Lily HBase indexer, also called HBaseMapReduceIndexerTool, can be used for batch indexing HBase tables.

Spark indexing using morphlines

If you are using Apache Spark, you can batch index data using the CrunchIndexerTool.

CrunchIndexerTool is a Spark or MapReduce ETL batch job that pipes data from HDFS files into Apache Solr through a morphline for extraction and transformation. The program is designed for flexible, scalable, fault-tolerant batch ETL pipeline jobs. It is implemented as an Apache Crunch pipeline, allowing it to run on MapReduce or Spark execution engines. The tool supports both Spark 2 and Spark 3. Spark 2 support will be discontinued in a future release.

CrunchIndexerTool requires a working MapReduce or Spark cluster, such as one installed using Cloudera Manager.

Note: This command requires a morphline file, which must include a SOLR_LOCATOR directive. The snippet that includes the SOLR_LOCATOR might appear as follows:

```
SOLR_LOCATOR : {
  # Name of solr collection
  collection : collection_name

  # ZooKeeper ensemble
  zkHost :
 "zk01.example.com:2181,zk02.example.com:2181,zk03.example.com:2181/
solr"
}

morphlines : [
  {
    id : morphline1
    importCommands : ["org.kitesdk.**", "org.apache.solr.**"]
    commands : [
      { generateUUID { field : id } }

      { # Remove record fields that are unknown to Solr managed-schema.
        # Recall that Solr throws an exception on any attempt to load a
 document that
        # contains a field that isn't specified in managed-schema.
        sanitizeUnknownSolrFields {
          solrLocator : ${SOLR_LOCATOR} # Location from which to fetch
 Solr schema
        }
      }

      { logDebug { format : "output record: {}", args : ["@{}"] } }

      {
        loadSolr {
          solrLocator : ${SOLR_LOCATOR}
        }
      }
    ]
  }
]
```

You can see the usage syntax CrunchIndexerTool by running the job with the -help argument. Unlike other Search indexing tools, the CrunchIndexerTool jar does not contain all dependencies. If you try to run the job without addressing this, you get an error such as the following:

```
hadoop jar /opt/cloudera/parcels/CDH/lib/solr/contrib/crunch/search-crunch.j
ar org.apache.solr.crunch.CrunchIndexerTool -help
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/crun
ch/types/PType
        at java.lang.Class.forName0(Native Method)
```

```
        at java.lang.Class.forName(Class.java:348)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:214)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
Caused by: java.lang.ClassNotFoundException: org.apache.crunch.types.PType
        at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
        ... 4 more
```

To see the command usage (or to run the job), you must first add the dependencies to the classpath:

```
export HADOOP_CLASSPATH="/opt/cloudera/parcels/CDH/lib/search/lib/search-cru
nch/*"
hadoop jar /opt/cloudera/parcels/CDH/lib/solr/contrib/crunch/search-crunc
h.jar org.apache.solr.crunch.CrunchIndexerTool -help
```

For reference, here is the command usage syntax:

Important: The command usage help incorrectly notes the following:

NOTE: MapReduce does not require extra steps for communicating with kerberos-enabled Solr

To run the MapReduce job on a Kerberos-enabled cluster, you must create and specify a jaas.conf file.

For example:

```
HADOOP_OPTS="-Djava.security.auth.login.config=/path/to/jaas.conf" \
hadoop jar /opt/cloudera/parcels/CDH/lib/solr/contrib/crunch/search-cru
nch.jar \
org.apache.solr.crunch.CrunchIndexerTool [...]
```

```
MapReduceUsage: export HADOOP_CLASSPATH=$myDependencyJarPaths; hadoop jar $m
yDriverJar
org.apache.solr.crunch.CrunchIndexerTool --libjars $myDependencyJarFiles
[MapReduceGenericOptions]...
        [--input-file-list URI] [--input-file-format [text|avro|avroParquet
]]
        [--input-file-projection-schema FILE]
        [--input-file-reader-schema FILE] --morphline-file FILE
        [--morphline-id STRING] [--pipeline-type STRING] [--xhelp]
        [--mappers INTEGER] [--parallel-morphline-inits INTEGER]
        [--dry-run] [--log4j FILE] [--chatty] [HDFS_URI [HDFS_URI ...]]

SparkUsage: spark-submit [SparkGenericOptions]... --master local|yarn --de
ploy-mode client|cluster
--jars $myDependencyJarFiles --class org.apache.solr.crunch.CrunchIndexerTo
ol $myDriverJar
        [--input-file-list URI] [--input-file-format [text|avro|avroParquet]
]
        [--input-file-projection-schema FILE]
        [--input-file-reader-schema FILE] --morphline-file FILE
        [--morphline-id STRING] [--pipeline-type STRING] [--xhelp]
        [--mappers INTEGER] [--parallel-morphline-inits INTEGER]
        [--dry-run] [--log4j FILE] [--chatty] [HDFS_URI [HDFS_URI ...]]
Spark or MapReduce ETL batch job  that  pipes data from (splittable or non-
splittable) HDFS files into Apache Solr,  and  along  the way runs the data
through a Morphline  for  extraction  and  transformation.  The  program is
designed for  flexible,  scalable  and  fault-tolerant  batch  ETL pipeline
jobs. It is implemented as an  Apache  Crunch  pipeline and as such can run
on either the Apache Hadoop MapReduce or Apache Spark execution engine.

The program proceeds in several consecutive phases, as follows:
```

```
1) Randomization phase: This (parallel)  phase  randomizes the list of HDFS
input files in order to spread ingestion  load more evenly among the mapper
tasks of the  subsequent  phase.  This  phase  is  only  executed  for non-
splittables files, and skipped otherwise.

2) Extraction phase: This  (parallel)  phase  emits  a  series of HDFS file
input streams (for non-splittable files) or  a series of input data records
(for splittable files).

3) Morphline  phase:  This  (parallel)  phase  receives  the  items  of the
previous phase, and  uses  a  Morphline  to  extract  the relevant content,
transform  it  and  load  zero  or   more  documents  into  Solr.  The  ETL
functionality is  flexible  and  customizable  using  chains  of  arbitrary
morphline commands that pipe  records  from  one  transformation command to
another. Commands to parse and  transform  a  set  of standard data formats
such as Avro, Parquet,  CSV,  Text,  HTML,  XML,  PDF,  MS-Office, etc. are
provided out of the box,  and  additional  custom  commands and parsers for
additional file or data formats can  be added as custom morphline commands.
Any kind of data format can be processed  and any kind output format can be
generated by any custom Morphline ETL  logic.  Also, this phase can be used
to send data  directly  to  a  live  SolrCloud  cluster  (via  the loadSolr
morphline command).

The program  is  implemented  as  a  Crunch  pipeline  and  as  such Crunch
optimizes the logical phases  mentioned  above  into  an efficient physical
execution plan that runs a single  mapper-only job, or as the corresponding
Spark equivalent.

Fault Tolerance: Task attempts  are  retried  on  failure  per the standard
MapReduce or Spark semantics. If the  whole  job fails you can retry simply
by rerunning the program again using the same arguments.
Comparison with MapReduceIndexerTool:

1) CrunchIndexerTool can also run on  the  Spark execution engine, not just
on MapReduce.
2)  CrunchIndexerTool  enables  interactive  low  latency  prototyping,  in
particular in Spark 'local' mode.
3) CrunchIndexerTool supports updates  (and  deletes) of existing documents
in Solr, not just inserts.
4) CrunchIndexerTool can exploit data  locality for splittable Hadoop files
(text, avro, avroParquet).
We recommend  MapReduceIndexerTool  for  large  scale  batch  ingestion use
cases where updates (or  deletes)  of  existing  documents  in Solr are not
required, and we recommend CrunchIndexerTool for all other use cases.
CrunchIndexerOptions:
  HDFS_URI               HDFS URI of  file  or  directory  tree  to ingest.
                         (default: [])
  --input-file-list URI, --input-list URI
                         Local URI or  HDFS  URI  of  a  UTF-8 encoded file
                         containing a list of HDFS  URIs to ingest, one URI
                         per line in the  file.  If  '-' is specified, URIs
                         are read  from  the  standard  input.  Multiple --
                         input-file-list arguments can be specified.
  --input-file-format [text|avro|avroParquet]
                         The Hadoop FileInputFormat  to  use for extracting
                         data from splittable HDFS  files.  Can  be one  of

                         ['text', 'avro', 'avroParquet']. If this option is
                         present the extraction phase will emit a series of
                         input data records rather  than  a series  of HDFS
                         file input streams.
  --input-file-projection-schema FILE
                         Relative or absolute path  to  an Avro schema file
                         on the local file  system.  This  will  be used as
```

```
                           the projection schema for Parquet input files.
  --input-file-reader-schema FILE
                           Relative or absolute path  to  an Avro schema file
                           on the local file  system.  This  will  be used as
                           the  reader  schema  for  Avro  or  Parquet  input
                           files.      Example:      src/test/resources/test-
                           documents/strings.avsc
  --morphline-file FILE    Relative or absolute path  to  a local config file
                           that contains one  or  more  morphlines.  The file
                           must be UTF-8  encoded.  It  will  be  uploaded to
                           each remote task. Example: /path/to/morphline.conf
  --morphline-id STRING    The identifier  of  the  morphline  that  shall be
                           executed  within   the   morphline   config   file
                           specified by --morphline-file. If the --morphline-
                           id option is  omitted  the  first  (i.e. top-most)
                           morphline  within  the   config   file   is  used.
                           Example: morphline1
  --pipeline-type STRING
                           The engine to use  for  executing  the job. Can be
                           'mapreduce' or 'spark'. (default: mapreduce)
  --xhelp, --help, -help
                           Show this help message and exit
  --mappers INTEGER        Tuning knob that indicates  the  maximum number of
                           MR mapper tasks to use.  -1  indicates use all map
                           slots available  on  the  cluster.  This parameter
                           only  applies   to   non-splittable   input  files
                           (default: -1)
  --parallel-morphline-inits INTEGER
                           Tuning knob that indicates  the  maximum number of
                           morphline instances  to  initialize  at  the  same
                           time. This kind of rate  limiting on rampup can be
                           useful  to  avoid  overload   conditions  such  as
                           ZooKeeper connection limits  or  DNS lookup limits
                           when using  many  parallel  mapper  tasks  because
                           each  such   task   contains   one   morphline.  1
                           indicates initialize  each  morphline  separately.
                           This feature  is  implemented  with  a distributed
                           semaphore. The default is to  use no rate limiting
                           (default: 2147483647)
  --dry-run                Run the pipeline  but  print  documents  to stdout
                           instead of loading  them  into  Solr.  This can be
                           used for quicker turnaround  during  early trial &
                           debug sessions. (default: false)
  --log4j FILE             Relative or absolute  path  to  a log4j.properties
                           config file on the  local  file  system. This file
                           will be uploaded  to  each  remote  task. Example:
                           /path/to/log4j.properties
  --chatty                 Turn on verbose output. (default: false)

SparkGenericOptions:       To print all options run 'spark-submit --help'

MapReduceGenericOptions: Generic options supported are
  --conf <configuration file>
                           specify an application configuration file
  -D <property=value>      use value for given property
  --fs <local|namenode:port>
                           specify a namenode
  --jt <local|resourcemanager:port>
                           specify a ResourceManager
  --files <comma separated list of files>
                           specify comma separated files to  be copied to the
                           map reduce cluster
  --libjars <comma separated list of jars>
                           specify comma separated  jar  files  to include in
```

```
                           the classpath.
  --archives <comma separated list of archives>
                           specify comma separated archives  to be unarchived
                           on the compute machines.
The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]


Examples:

# Prepare - Copy input files into HDFS:
export myResourcesDir=src/test/resources # for build from git
export myResourcesDir=/opt/cloudera/parcels/CDH/share/doc/search-*/search-cr
unch # for CDH with parcels
export myResourcesDir=/usr/share/doc/search-*/search-crunch # for CDH with p
ackages
hadoop fs -copyFromLocal $myResourcesDir/test-documents/hello1.txt hdfs:/use
r/systest/input/

# Prepare variables for convenient reuse:
export myDriverJarDir=target # for build from git
export myDriverJarDir=/opt/cloudera/parcels/CDH/lib/solr/contrib/crunch #
for CDH with parcels
export myDriverJarDir=/usr/lib/solr/contrib/crunch # for CDH with packages
export myDependencyJarDir=target/lib # for build from git
export myDependencyJarDir=/opt/cloudera/parcels/CDH/lib/search/lib/search-
crunch # for CDH with parcels
export myDependencyJarDir=/usr/lib/search/lib/search-crunch # for CDH with
 packages
export myDriverJar=$(find $myDriverJarDir -maxdepth 1 -name 'search-crunch-
*.jar' ! -name '*-job.jar' ! -name '*-sources.jar')
export myDependencyJarFiles=$(find $myDependencyJarDir -name '*.jar' | sort
 | tr '\n' ',' | head -c -1)
export myDependencyJarPaths=$(find $myDependencyJarDir -name '*.jar' | sort
 | tr '\n' ':' | head -c -1)
export myJVMOptions="-DmaxConnectionsPerHost=10000 -DmaxConnections=10000 -
Djava.io.tmpdir=/my/tmp/dir/" # connection settings for solrj, also custom t
mp dir

# MapReduce on Yarn - Ingest text file line by line into Solr:
export HADOOP_CLIENT_OPTS="$myJVMOptions"; export HADOOP_CLASSPATH=$myDep
endencyJarPaths; hadoop \
  --config /etc/hadoop/conf.cloudera.YARN-1 \
  jar $myDriverJar org.apache.solr.crunch.CrunchIndexerTool \
  --libjars $myDependencyJarFiles \
  -D mapreduce.map.java.opts="-Xmx500m $myJVMOptions" \
  -D morphlineVariable.ZK_HOST=$(hostname):2181/solr \
  --files $myResourcesDir/test-documents/string.avsc \
  --morphline-file $myResourcesDir/test-morphlines/loadSolrLine.conf \
  --pipeline-type mapreduce \
  --chatty \
  --log4j $myResourcesDir/log4j.properties \
  /user/systest/input/hello1.txt
# Spark in Local Mode (for rapid prototyping) - Ingest into Solr:
spark-submit \
  --master local \
  --deploy-mode client \
  --jars $myDependencyJarFiles \
  --executor-memory 500M \
  --conf "spark.executor.extraJavaOptions=$myJVMOptions" \
  --driver-java-options "$myJVMOptions" \
  # --driver-library-path /opt/cloudera/parcels/CDH/lib/hadoop/lib/native #
 for Snappy on CDH with parcels\
```

```
  # --driver-library-path /usr/lib/hadoop/lib/native # for Snappy on CDH
with packages \
  --class org.apache.solr.crunch.CrunchIndexerTool \
  $myDriverJar \
  -D morphlineVariable.ZK_HOST=$(hostname):2181/solr \
  --morphline-file $myResourcesDir/test-morphlines/loadSolrLine.conf \
  --pipeline-type spark \
  --chatty \
  --log4j $myResourcesDir/log4j.properties \
  /user/systest/input/hello1.txt

# Spark on Yarn in Client Mode (for testing) - Ingest into Solr:
Same as above, except replace '--master local' with '--master yarn'

# View the yarn executor log files (there is no GUI yet):
yarn logs --applicationId $application_XYZ

# Spark on Yarn in Cluster Mode (for production) - Ingest into Solr:
spark-submit \
  --master yarn \
  --deploy-mode cluster \
  --jars $myDependencyJarFiles \
  --executor-memory 500M \
  --conf "spark.executor.extraJavaOptions=$myJVMOptions" \
  --driver-java-options "$myJVMOptions" \
  --class org.apache.solr.crunch.CrunchIndexerTool \
  --files $(ls $myResourcesDir/log4j.properties),$(ls $myResourcesDir/test-m
orphlines/loadSolrLine.conf)\
  $myDriverJar \
  -D hadoop.tmp.dir=/tmp \
  -D morphlineVariable.ZK_HOST=$(hostname):2181/solr \
  --morphline-file loadSolrLine.conf \
  --pipeline-type spark \
  --chatty \
  --log4j log4j.properties \
  /user/systest/input/hello1.txt

# Spark on Yarn in Cluster Mode (for production) - Ingest into Secure (Ker
beros-enabled) Solr:
# Spark requires two additional steps compared to non-secure solr:
# (NOTE: MapReduce does not require extra steps for communicating with kerbe
ros-enabled Solr)
# 1) Create a delegation token file
#    a) kinit as the user who will make solr requests
#    b) request a delegation token from solr and save it to a file:
#       e.g. using curl:
#       "curl --negotiate -u: http://solr-host:port/solr/admin?op=GETDELEGA
TIONTOKEN > tokenFile.txt"
# 2) Pass the delegation token file to spark-submit:
#    a) add the delegation token file via --files
#    b) pass the file name via -D tokenFile
#       spark places this file in the cwd of the executor, so only list the
 file name, no path
spark-submit \
  --master yarn \
  --deploy-mode cluster \
  --jars $myDependencyJarFiles \
  --executor-memory 500M \
  --conf "spark.executor.extraJavaOptions=$myJVMOptions" \
  --driver-java-options "$myJVMOptions" \
  --class org.apache.solr.crunch.CrunchIndexerTool \
  --files $(ls $myResourcesDir/log4j.properties),$(ls $myResourcesDir/test-
morphlines/loadSolrLine.conf),tokenFile.txt\
  $myDriverJar \
```

```
  -D hadoop.tmp.dir=/tmp \
  -D morphlineVariable.ZK_HOST=$(hostname):2181/solr \
  -DtokenFile=tokenFile.txt \
  --morphline-file loadSolrLine.conf \
  --pipeline-type spark \
  --chatty \
  --log4j log4j.properties \
  /user/systest/input/hello1.txt
```

MapReduce indexing

Cloudera Search provides the ability to batch index documents using MapReduce jobs.

Running an example indexing job

For examples of running a MapReduce job to index documents, see Cloudera Search Tutorial

MapReduceIndexerTool

MapReduceIndexerTool (MRIT) is a MapReduce batch job driver that takes a morphline and creates a set of Solr index shards from a set of input files and writes the indexes into HDFS or local FS in a flexible, scalable, and fault-tolerant manner. MRIT also supports merging the output shards into a set of live customer-facing Solr servers, typically a SolrCloud.

Important: Merging output shards into live customer-facing Solr servers can only be completed if all replicas are online.

The indexer creates an offline index on HDFS in the output directory specified by the --output-dir parameter. If the --go-live parameter is specified, Solr merges the resulting offline index into the live running service. Thus, the Solr service must have read access to the contents of the output directory to complete the go-live step. In an environment with restrictive permissions, such as one with an HDFS umask of 077, the Solr user may not be able to read the contents of the newly created directory. To address this issue, the indexer automatically applies the HDFS ACLs to enable Solr to read the output directory contents. These ACLs are only applied if HDFS ACLs are enabled on the HDFS NameNode.

The indexer only makes ACL updates to the output directory and its contents. If the output directory's parent directories do not include the run permission, the Solr service is not able to access the output directory. Solr must have run permissions from standard permissions or ACLs on the parent directories of the output directory.

Note: Using --libjars parameter in dry-run mode does not work. Instead, specify the JAR files using the HADOOP_CLASSPATH environmental variable.

Related Information

Extracting, transforming, and loading data with Cloudera Morphlines

Using morphlines to index Avro

Using morphlines with syslog

HDFS ACLs

MapReduceIndexerTool input splits

Different from some other indexing tools, the MapReduceIndexerTool does not operate on HDFS blocks as input splits. This means that when indexing a smaller number of large files, fewer hosts may be involved. For example, indexing two files that are each one GB results in two hosts acting as mappers. If these files were stored on a system with a 128 MB block size, other mappers might divide the work on the two files among 16 mappers, corresponding to the 16 HDFS blocks that store the two files.

This intentional design choice aligns with MapReduceIndexerTool supporting indexing non-splittable file formats such as JSON, XML, jpg, or log4j.

In theory, this could result in inefficient use of resources when a single host indexes a large file while many other hosts sit idle. In reality, this indexing strategy typically results in satisfactory performance in production environments because in most cases the number of files is large enough that work is spread throughout the cluster.

While dividing tasks by input splits does not present problems in most cases, users may still want to divide indexing tasks along HDFS splits. In that case, use the CrunchIndexerTool, which can work with Hadoop input splits using the input-file-format option.

MapReduceIndexerTool metadata

The MapReduceIndexerTool generates metadata fields for each input file when indexing. These fields can be used in morphline commands. These fields can also be stored in Solr, by adding definitions like the following to your Solr managed-schema file. After the MapReduce indexing process completes, the fields are searchable through Solr.

```
<!-- file metadata -->
<field name="file_download_url" type="string" indexed="false" stored="tru
e" />
<field name="file_upload_url" type="string" indexed="false" stored="true" />
<field name="file_scheme" type="string" indexed="true" stored="true" />
<field name="file_host" type="string" indexed="true" stored="true" />
<field name="file_port" type="int" indexed="true" stored="true" />
<field name="file_path" type="string" indexed="true" stored="true" />
<field name="file_name" type="string" indexed="true" stored="true" />
<field name="file_length" type="tlong" indexed="true" stored="true" />
<field name="file_last_modified" type="tlong" indexed="true" stored="true" /
>
<field name="file_owner" type="string" indexed="true" stored="true" />
<field name="file_group" type="string" indexed="true" stored="true" />
<field name="file_permissions_user" type="string" indexed="true" stored="
true" />
<field name="file_permissions_group" type="string" indexed="true" stored="t
rue" />
<field name="file_permissions_other" type="string" indexed="true" stored="tr
ue" />
<field name="file_permissions_stickybit" type="boolean" indexed="true" st
ored="true" />
```

Example output:

```
"file_upload_url":"foo/test-documents/sample-statuses-20120906-141433.avro",
"file_download_url":"hdfs://host1.mycompany.com:8020/user/foo/ test-documen
ts/sample-statuses-20120906-141433.avro",
"file_scheme":"hdfs",
"file_host":"host1.mycompany.com",
"file_port":8020,
"file_name":"sample-statuses-20120906-141433.avro",
"file_path":"/user/foo/test-documents/sample-statuses-20120906-141433.avro",
"file_last_modified":1357193447106,
"file_length":1512,
"file_owner":"foo",
"file_group":"foo",
"file_permissions_user":"rw-",
"file_permissions_group":"r--",
"file_permissions_other":"r--",
"file_permissions_stickybit":false,
```

MapReduceIndexerTool usage syntax

Learn about the use of the MapReduceIndexer command line tool.

Important: You must run the indexer tool with the following command-line argument:

```
-D 'mapreduce.job.user.classpath.first=true'
```

Running the tool without this argument triggers the following error:

```
ERROR [main] org.apache.hadoop.mapred.YarnChild: Error running child :
 java.lang.NoSuchMethodError:
com.codahale.metrics.MetricRegistry.meter(Ljava/lang/String;Lcom/cod
ahale/metrics/MetricRegistry$MetricSupplier;)Lcom/codahale/metrics/M
eter;
```

Note:

For HBase batch indexing to work with JDK 17, you also need to add the following Hadoop option:

```
HADOOP_OPTS="--add-opens java.base/java.nio=ALL-UNNAMED"
```

Important:

If Solr uses a custom system user, you need to pass this username to MapReduceIndexerTool jobs using the following argument:

```
-Dsolr.authorization.superuser=<<customSolrSystemUser>>
```

This is necessary for setting up ACLs of the MapReduce jobs directory correctly.

To view the usage syntax in a default parcel-based deployment, run:

```
hadoop jar /opt/cloudera/parcels/CDH/jars/search-mr-*-job.jar \
org.apache.solr.hadoop.MapReduceIndexerTool --help
```

usage:

```
hadoop [GenericOptions]... jar search-mr-*-job.jar org.apache.solr.hadoop.Ma
pReduceIndexerTool
       [--help] --output-dir HDFS_URI [--input-list URI]
       --morphline-file FILE [--morphline-id STRING] [--solr-home-dir DIR]
       [--update-conflict-resolver FQCN] [--mappers INTEGER]
       [--reducers INTEGER] [--max-segments INTEGER]
       [--fair-scheduler-pool STRING] [--dry-run] [--log4j FILE]
       [--verbose] [--show-non-solr-cloud] [--zk-host STRING] [--go-live]
       [--collection STRING] [--go-live-min-replication-factor INTEGER]
       [--go-live-threads INTEGER] [--go-live-timeout INTEGER]
       [--filesystem STRING] [--private-key FILE]
       [--known-hosts FILE] [--local-merge-dir DIR]
       [--keytab FILE]
       [HDFS_URI [HDFS_URI ...]]
```

MapReduce batch job driver that takes a morphline and creates a set of Solr index shards from a set of input files and writes the indexes into HDFS, in a flexible, scalable and fault-tolerant manner. It also supports merging the output shards into a set of live customer facing Solr servers, typically a SolrCloud. The program proceeds in several consecutive MapReduce based phases, as follows:

1. Randomization phase: This (parallel) phase randomizes the list of input files in order to spread indexing load more evenly among the mappers of the subsequent phase.
2. Mapper phase: This (parallel) phase takes the input files, extracts the relevant content, transforms it and hands SolrInputDocuments to a set of reducers. The ETL functionality is flexible and customizable using chains of arbitrary morphline commands that pipe records from one transformation command to another. Commands to parse and transform a set of standard data formats such as Avro, CSV, Text, HTML, XML, PDF, Word, Excel, etc. are provided out of the box, and additional custom commands and parsers for additional file or data formats

can be added as morphline plugins. This is done by implementing a simple Java interface that consumes a record (e.g. a file in the form of an InputStream plus some headers plus contextual metadata) and generates as output zero or more records. Any kind of data format can be indexed and any Solr documents for any kind of Solr schema can be generated, and any custom ETL logic can be registered and executed. Record fields, including MIME types, can also explicitly be passed by force from the CLI to the morphline, for example: hadoop ... -D morphlineField ._attachment_mimetype=text/csv

3. Reducer phase: This (parallel) phase loads the mapper's SolrInputDocuments into one EmbeddedSolrServer per reducer. Each such reducer and Solr server can be seen as a (micro) shard. The Solr servers store their data in HDFS.
4. Mapper-only merge phase: This (parallel) phase merges the set of reducer shards into the number of solr shards expected by the user, using a mapper-only job. This phase is omitted if the number of shards is already equal to the number of shards expected by the user.
5. Go-live phase: This optional (parallel) phase merges the output shards of the previous phase into a set of live customer facing Solr servers, typically a SolrCloud. If this phase is omitted you can explicitly point each Solr server to one of the HDFS output shard directories. Fault Tolerance: Mapper and reducer task attempts are retried on failure per the standard MapReduce semantics. On program startup all data in the -- output-dir is deleted if that output directory already exists. If the whole job fails you can retry simply by rerunning the program again using the same arguments.

positional arguments:

HDFS_URI

HDFS URI of file or directory tree to index. (default: [])

optional arguments:

--help, -help, -h

Show help message and exit

--input-list URI

Local URI or HDFS URI of a UTF-8 encoded file containing a list of HDFS URIs to index, one URI per line in the file. If '-' is specified, URIs are read from the standard input. Multiple -- input-list arguments can be specified.

--morphline-id STRING

The identifier of the morphline that shall be executed within the morphline config file specified by --morphline-file. If the --morphline-id option is ommitted the first (i.e. top-most) morphline within the config file is used. Example: morphline1

--solr-home-dir DIR

Optional relative or absolute path to a local dir containing Solr conf/ dir and in particular conf/ solrconfig.xml and optionally also lib/ dir. This directory will be uploaded to each MR task. Example: src/test/resources/solr/minimr

--update-conflict-resolver FQCN

Fully qualified class name of a Java class that implements the UpdateConflictResolver interface. This enables deduplication and ordering of a series of document updates for the same unique document key. For example, a MapReduce batch job might index multiple files in the same job where some of the files contain old and new versions of the very same document, using the same unique document key.

Typically, implementations of this interface forbid collisions by throwing an exception, or ignore all but the most recent document version, or, in the general case, order colliding updates ascending from least recent to most recent (partial) update. The caller of this interface (i.e. the Hadoop Reducer) will then apply the updates to Solr in the order returned by the orderUpdates() method.

The default RetainMostRecentUpdateConflictResolver implementation ignores all but the most recent document version, based on a configurable numeric Solr field, which defaults to the file_last_modified timestamp (default: org.apache.solr.hadoop.dedup.

--mappers INTEGER

RetainMostRecentUpdateConflictResolver) Tuning knob that indicates the maximum number of MR mapper tasks to use. -1 indicates use all map slots available on the cluster. (default: -1)

--reducers INTEGER

Tuning knob that indicates the number of reducers to index into. 0 is reserved for a mapper-only feature that may ship in a future release. -1 indicates use all reduce slots available on the cluster. -2 indicates use one reducer per output shard, which disables the mtree merge MR algorithm. The mtree merge MR algorithm improves scalability by spreading load (in particular CPU load) among a number of parallel reducers that can be much larger than the number of solr shards expected by the user. It can be seen as an extension of concurrent lucene merges and tiered lucene merges to the clustered case. The subsequent mapper-only phase merges the output of said large number of reducers to the number of shards expected by the user, again by utilizing more available parallelism on the cluster. (default: -1)

--max-segments INTEGER

Tuning knob that indicates the maximum number of segments to be contained on output in the index of each reducer shard. After a reducer has built its output index it applies a merge policy to merge segments until there are <= maxSegments lucene segments left in this index. Merging segments involves reading and rewriting all data in all these segment files, potentially multiple times, which is very I/O intensive and time consuming. However, an index with fewer segments can later be merged faster, and it can later be queried faster once deployed to a live Solr serving shard. Set maxSegments to 1 to optimize the index for low query latency. Set maxSegments to 0 or -1 to skip the entire optimize phase. In a nutshell, a small maxSegments value trades indexing latency for subsequently improved query latency. This can be a reasonable trade-off for batch indexing systems. (default: 1)

--dry-run

Run in local mode and print documents to stdout instead of loading them into Solr. This executes the morphline in the client process (without submitting a job to MR) for quicker turnaround during early trial & debug sessions. (default: false)

--log4j FILE

Relative or absolute path to a log4j.properties config file on the local file system. This file will be uploaded to each MR task. Example: /path/to/log4j.properties

--verbose, -v

Turn on verbose output. (default: false)

--show-non-solr-cloud

Also show options for Non-SolrCloud mode as part of --help. (default: false)

Required arguments:

--output-dir HDFS_URI

HDFS directory to write Solr indexes to. Inside there one output directory per shard will be generated. Example: hdfs://c2202.mycompany.com/user/$USER/test

--morphline-file FILE

Relative or absolute path to a local config file that contains one or more morphlines. The file must be UTF-8 encoded. Example: /path/to/morphline.conf

Cluster arguments:

Arguments that provide information about your Solr cluster.

--zk-host STRING

The address of a ZooKeeper ensemble being used by a SolrCloud cluster. This ZooKeeper ensemble will be examined to determine the number of output shards to create as well as the Solr URLs to

merge the output shards into when using the --go- live option. Requires that you also pass the --collection to merge the shards into.

The --zk-host option implements the same partitioning semantics as the standard SolrCloud Near-Real-Time (NRT) API. This enables to mix batch updates from MapReduce ingestion with updates from standard Solr NRT ingestion on the same SolrCloud cluster, using identical unique document keys.

Format is: a list of comma separated host:port pairs, each corresponding to a zk server. Example: '127.0.0.1:2181,127.0.0.1:2182,127.0.0.1: 2183' If the optional chroot suffix is used the example would look like: '127.0.0.1:2181/solr, 127.0.0.1:2182/solr,127.0.0.1:2183/solr' where the client would be rooted at '/solr' and all paths would be relative to this root - i.e. getting/setting/etc... '/foo/bar' would result in operations being run on '/solr/foo/bar' (from the server perspective).

If --solr-home-dir is not specified, the Solr home directory for the collection may be downloaded from this ZooKeeper ensemble.

Go live arguments:

Arguments for merging the shards that are built into a live Solr cluster. Also see the Cluster arguments.

--go-live

Allows you to optionally merge the final index shards into a live Solr cluster after they are built. You can pass the ZooKeeper address with -- zk-host and the relevant cluster information will be auto detected. (default: false)

-collection STRING

The SolrCloud collection to merge shards into when using --go-live and --zk-host. Example: collection1

--go-live-min-replication-factor INTEGER

The minimum number of SolrCloud replicas to successfully merge any final index shard into. The go-live job phase attempts to merge final index shards into all SolrCloud replicas. Some of these merge operations may fail, for example if some SolrCloud servers are down. This option enables indexing jobs to succeed even if some such merge operations fail on SolrCloud followers. Successful merge operations into all leaders are always required for job success, regardless of the value of --go-live-min- replication-factor. -1 indicates require successful merge operations into all replicas. 1 indicates require successful merge operations only into leader replicas. (default: -1)

--go-live-threads INTEGER

Tuning knob that indicates the maximum number of live merges to run in parallel at one time. (default: 1000)

--go-live-timeout INTEGER

Timeout in ms to wait for the merge to complete before the connection times out and the tool fails.

--filesystem STRING

Allows you to change whether you want to merge indexes into a collection on HDFS or on localfs. Possible values are 'hdfs' and 'localfs'. By default MRIT will use hdfs for merging the index. In case of 'localfs' make sure that the target collection uses localfs. This option also needs --use-zk-solrconfig.xml and --private-key parameters as well. (default: hdfs)

--private-key FILE

Path to a private key using which the user running the MRIT job is able to SSH into all Solr hosting nodes. Must be provided together with the '--filesystem localfs' option

--known-hosts FILE

Path to the known hosts file which contains keys to all the Solr hosting nodes. Used only with '--filesystem localfs' option

Always specify this in production use! If not provided strict host checking will be skipped which is unsafe and only recommended for testing purposes

--local-merge-dir DIR

Path to a directory on all Solr hosts to temporarily copy index to before mergeindexes action.

Used only with '--filesystem localfs' option. The user running the MRIT job needs to have permission to read and write into this directory. This directory needs to be relative to SOLR_HOME or SOLR_DATA_HOME or needs to be specified in the Solr system property `-Dsolr.allowPaths`

--keytab FILE

Path to the keytab file for the user running the MRIT job on all Solr hosting nodes. Used only with '--filesystem localfs' option. If not provided kinit is skipped and MRIT will expect an unsecure environment or a valid kerberos ticket already present on all Solr hosting nodes for the current user.

Generic options supported are:

--conf <configuration file>

specify an application configuration file

-D <property=value>

define a value for a given property

-fs <file:///|hdfs://namenode:port>

specify default filesystem URL to use, overrides 'fs.defaultFS' property from configurations.

--jt <local|resourcemanager:port>

specify a ResourceManager

--files <file1,...>

specify a comma-separated list of files to be copied to the map reduce cluster

--libjars <jar1,...>

specify a comma-separated list of jar files to be included in the classpath

--archives <archive1,...>

specify a comma-separated list of archives to be unarchived on the compute machines

The general command line syntax is:

```
command [genericOptions] [commandOptions]
```

Examples:

(Re)index an Avro based Twitter tweet file:

```
sudo -u hdfs hadoop \
  --config /etc/hadoop/conf.cloudera.mapreduce1 \
  jar target/search-mr-*-job.jar org.apache.solr.hadoop.MapReduceIndexerTool
 \
  -D 'mapred.child.java.opts=-Xmx2g' \
  -D 'mapreduce.map.java.opts=-Xmx2g' \
  --log4j src/test/resources/log4j.properties \
  --morphline-file ../search-core/src/test/resources/test-morphlines/tutor
ialReadAvroContainer.conf \
  --solr-home-dir src/test/resources/solr/minimr \
  --output-dir hdfs://c2202.mycompany.com/user/$USER/test \
  --shards 1 \
  hdfs:///user/$USER/test-documents/sample-statuses-20120906-141433.avro
```

Note: If the suggested Java heap size of 2 Gigabytes (2g) proves insufficient, increase it to 4g or 6g.

(Re)index all files that match all of the following conditions:

1. File is contained in dir tree hdfs:///user/$USER/solrloadtest/twitter/tweets
2. file name matches the glob pattern 'sample-statuses*.gz'
3. file was last modified less than 100000 minutes ago
4. file size is between 1 MB and 1 GB

Also include extra library jar file containing JSON tweet Java parser:

```
hadoop fs \
  -find hdfs:///user/$USER/solrloadtest/twitter/tweets \
  -type f \
  -name 'sample-statuses*.gz' \
  -mmin -1000000 \
  -size -100000000c \
  -size +1000000c \
| sudo -u hdfs hadoop \
  --config /etc/hadoop/conf.cloudera.mapreduce1 \
  jar target/search-mr-*-job.jar org.apache.solr.hadoop.MapReduceIndexerTool
 \
  --libjars /path/to/kite-morphlines-twitter-0.10.0.jar \
  -D 'mapred.child.java.opts=-Xmx2g' \
  -D 'mapreduce.map.java.opts=-Xmx2g' \
  --log4j src/test/resources/log4j.properties \
  --morphline-file ../search-core/src/test/resources/test-morphlines/tutori
alReadJsonTestTweets.conf \
  --solr-home-dir src/test/resources/solr/minimr \
  --output-dir hdfs://c2202.mycompany.com/user/$USER/test \
  --shards 100 \
  --input-list -
```

Note: If the suggested Java heap size of 2 Gigabytes (2g) proves insufficient, increase it to 4g or 6g.

Go live by merging resulting index shards into a live Solr cluster (explicitly specify Solr URLs - for a SolrCloud cluster see next example):

```
sudo -u hdfs hadoop \
  --config /etc/hadoop/conf.cloudera.mapreduce1 \
  jar target/search-mr-*-job.jar org.apache.solr.hadoop.MapReduceIndexerTool
 \
  -D 'mapred.child.java.opts=-Xmx2g' \
  -D 'mapreduce.map.java.opts=-Xmx2g' \
  --log4j src/test/resources/log4j.properties \
  --morphline-file ../search-core/src/test/resources/test-morphlines/tuto
rialReadAvroContainer.conf \
  --solr-home-dir src/test/resources/solr/minimr \
  --output-dir hdfs://c2202.mycompany.com/user/$USER/test \
  --shard-url http://solr001.mycompany.com:8983/solr/collection1 \
  --shard-url http://solr002.mycompany.com:8983/solr/collection1 \
  --go-live \
  hdfs:///user/foo/indir
```

Note: If the suggested Java heap size of 2 Gigabytes (2g) proves insufficient, increase it to 4g or 6g.

Go live by merging resulting index shards into a live SolrCloud cluster (discover shards and Solr URLs through ZooKeeper):

```
sudo -u hdfs hadoop \
```

```
  --config /etc/hadoop/conf.cloudera.mapreduce1 \
  jar target/search-mr-*-job.jar org.apache.solr.hadoop.MapReduceIndexerTool
 \
  -D 'mapred.child.java.opts=-Xmx2g' \
  -D 'mapreduce.map.java.opts=-Xmx2g' \
  --log4j src/test/resources/log4j.properties \
  --morphline-file ../search-core/src/test/resources/test-morphlines/tutor
ialReadAvroContainer.conf \
  --output-dir hdfs://c2202.mycompany.com/user/$USER/test \
  --zk-host zk01.mycompany.com:2181/solr \
  --collection collection1 \
  --go-live \
  hdfs:///user/foo/indir
```

Note: If the suggested Java heap size of 2 Gigabytes (2g) proves insufficient, increase it to 4g or 6g.

MapReduce on Yarn - Pass custom JVM arguments (including a custom tmp directory)

```
HADOOP_CLIENT_OPTS='-DmaxConnectionsPerHost=10000 -DmaxConnections=10000 -Dj
ava.io.tmpdir=/my/tmp/dir/'; \
sudo -u hdfs hadoop \
  --config /etc/hadoop/conf.cloudera.mapreduce1 \
  jar target/search-mr-*-job.jar org.apache.solr.hadoop.MapReduceIndexerT
ool \
  -D 'mapreduce.map.java.opts=-DmaxConnectionsPerHost=10000 -DmaxConnectio
ns=10000' \
  -D 'mapreduce.reduce.java.opts=-DmaxConnectionsPerHost=10000 -DmaxConnecti
ons=10000' \
  --log4j src/test/resources/log4j.properties \
  --morphline-file ../search-core/src/test/resources/test-morphlines/tutori
alReadAvroContainer.conf \
  --solr-home-dir src/test/resources/solr/minimr \
  --output-dir hdfs://c2202.mycompany.com/user/$USER/test \
  --shards 1 \
  hdfs:///user/$USER/test-documents/sample-statuses-20120906-141433.avro
```

Indexing data with MapReduceIndexerTool in Solr backup format

MapReduceIndexerTool (MRIT) is capable of batch indexing a dataset and providing the output in the format of Solr backups, using morphlines. This backup can then be ingested into Solr using a backup operation.

About this task

The MapReduceIndexerTool (MRIT) backup format feature addresses the dilemma of ingesting indexes produced by MRIT jobs into Solr:

- Near-real-time (NRT) ingestion using the --go-live option is resource-intensive and involves merging indexes.
- Batch indexing requires shutting down the Solr server.

MRIT backup format takes the best of both worlds: by creating the index in the Solr backup format, it can be ingested into Solr as a restore operation, using the solrctl command line utility. This method is significantly less resource intensive on the part of Solr compared to NRT with --go-live. Restoring the backup results in a new collection which can be queried directly or put behind an alias.

Procedure

1. To perform a batch indexing job on MRIT with the output in Solr backup format, run the following command:

```
hadoop jar /opt/cloudera/parcels/CDH/lib/solr/contrib/mr/
search-mr-*-job.jar  org.apache.solr.hadoop.MapReduceIndexe
rTool --morphline-file [***MORPHLINE_FILE***]  --output-dir
```

```
 "[***ABSOLUTE/PATH/TO/OUTPUT/DIRECTORY***]" --use-backup-format
 --backup-name [***USER_SPECIFIED_NAME_FOR_THE_BACKUP***] --zk-ho
st [***HOSTNAME***]:2181/solr --collection [***COLLECTION_NAME***]
 "[***ABSOLUTE/PATH/TO/INPUT/FILE***]"
```

Replace *[***MORPHLINE_FILE***]*, *[***ABSOLUTE/PATH/TO/OUTPUT/DIRECTORY***]*, *[***USER_SPECIFIED_NAME_FOR_THE_BACKUP***]*, *[***HOSTNAME***]*, *[***COLLECTION_NAME***]*, and *[***ABSOLUTE/PATH/TO/INPUT/FILE***]* with values applicable in your environment.

For example:

To parse the contents of hdfs://ns1:8020/tmp/inputfile using the morphline file morphlines.conf and write the resulting index to hdfs://ns1:8020/tmp/output/results/backupName:

```
hadoop jar /opt/cloudera/parcels/CDH/lib/solr/contrib/mr/search-mr-*-job
.jar  org.apache.solr.hadoop.MapReduceIndexerTool --morphline-file morph
lines.conf  --output-dir "hdfs://ns1:8020/tmp/output" --use-backup-forma
t --backup-name backupName --zk-host zk-server:2181/solr --collection co
llection "hdfs://ns1:8020/tmp/inputfile"
```

2. To create a new collection with the contents of the backup:

```
solrctl collection --restore [***USER_DEFINED_COLLECTION_NAME***] -
b [***NAME_OF_THE_INDEX_IN_BACKUP_FORMAT***] -l [***ABSOLUTE/PATH/TO/
RESTORE/TARGET/DIRECTORY***] -i [***REQUEST_ID***]
```

Make sure that you use a unique <*REQUESTID*> each time you run this command.

Note:

Statuses of historic job runs are stored in ZooKeeper and can be retrieved using the solrctl collection --req uest-status *[***REQUEST_ID***]* command. The number of async call responses stored in a cluster is limited to 10,000.

Status information can be removed from ZooKeeper using the DELETESTATUS API call.

Replace *[***USER_DEFINED_COLLECTION_NAME***]*, *[***NAME_OF_THE_INDEX_IN_BACKUP_FORMAT***]*, *[***ABSOLUTE/PATH/TO/RESTORE/TARGET/DIRECTORY***]* with values applicable in your environment.

For example:

To create the collection finalcollectionName from the backup backupName to the directory hdfs://ns1:8020/tmp/output/results with the request ID 1234:

```
solrctl collection --restore finalcollectionName -b backupName -l hdfs://
ns1:8020/tmp/output/results -i 1234
```

3. To monitor the status of the restore step, run the following command:

```
solrctl collection --request-status [***REQUEST_ID***]
```

Replace *[***REQUEST_ID***]* with the ID of the task you want to monitor.

For example:

```
solrctl collection --request-status 1234
```

Related Information

Collection aliasing

Asynchronous calls

Lily HBase batch indexing for Cloudera Search

You can batch index HBase tables using the Lily HBase batch indexer MapReduce job (HBaseMapReduceIndexerTool). This batch indexing does not require HBase replication or the Lily HBase Indexer Service. Subsequently you do not need to register a Lily HBase Indexer configuration with the Lily HBase Indexer Service.

The indexer supports flexible, custom, application-specific rules to extract, transform, and load HBase data into Solr. Solr search results can contain columnFamily:qualifier links back to the data stored in HBase. This way, applications can use the search result set to directly access matching raw HBase cells.

The following procedures demonstrate creating a small HBase table and using the HBaseMapReduceIndexerTool to index the table into a collection:

Important: Do not use the Lily HBase Batch Indexer during a rolling upgrade. The indexer requires all replicas be hosted on the same HBase version. If an indexing job is running during a rolling upgrade, different nodes may be running pre- and post-upgrade versions of HBase.

Populating an HBase Table

Procedure

After configuring and starting your system, create an HBase table and add rows to it. For example:

```
hbase shell
hbase(main):002:0> create 'sample_table', {NAME => 'data'}
hbase(main):002:0> put 'sample_table', 'row1', 'data', 'value'
hbase(main):001:0> put 'sample_table', 'row2', 'data', 'value2'
```

Create a Collection in Cloudera Search

About this task

A collection in Search used for HBase indexing must have a Solr schema that accommodates the types of HBase column families and qualifiers that are being indexed. To begin, consider adding the all-inclusive data field to a default schema.

Procedure

Once you decide on a schema, create a collection using commands similar to the following:

```
solrctl instancedir --generate $HOME/hbase_collection_config
## Edit $HOME/hbase_collection_config/conf/managed-schema as needed ##
solrctl config --upload hbase_collection_config $HOME/hbase_collection_confi
g
solrctl collection --create hbase_collection -s <numShards> -c hbase_colle
ction_config
```

Creating a Lily HBase Indexer Configuration File

About this task

Configure individual Lily HBase Indexers using the hbase-indexer command-line utility. Typically, there is one Lily HBase Indexer configuration file for each HBase table, but there can be as many Lily HBase Indexer configuration files as there are tables, column families, and corresponding collections in Search. Each Lily HBase Indexer configuration is defined in an XML file, such as morphline-hbase-mapper.xml.

An indexer configuration XML file must refer to the MorphlineResultToSolrMapper implementation and point to the location of a Morphline configuration file, as shown in the following morphline-hbase-mapper.xml indexer configuration file.

Procedure

- Set morphlineFile to the relative path morphlines.conf. Make sure the file is readable by the HBase system user (hbase by default).

```
$ cat $HOME/morphline-hbase-mapper.xml

<?xml version="1.0"?>
<indexer table="sample_table"
mapper="com.ngdata.hbaseindexer.morphline.MorphlineResultToSolrMapper">
   <!-- The relative path on the local file system to the
   morphline configuration file. -->

   <param name="morphlineFile" value="morphlines.conf"/>

   <!-- The optional morphlineId identifies a morphline if there are multi
ple
   morphlines in morphlines.conf -->
   <!-- <param name="morphlineId" value="morphline1"/> -->
</indexer>
```

The Lily HBase Indexer configuration file also supports the standard attributes of any HBase Lily Indexer on the top-level <indexer> element. It does not support the <field> element and <extract> elements.

Creating a Morphline Configuration File

About this task

After creating an indexer configuration XML file, you can configure morphline ETL transformation commands in a morphlines.conf configuration file. The morphlines.conf configuration file can contain any number of morphline commands. Typically, an extractHBaseCells command is the first command. The readAvroContainer or readAvro morphline commands are often used to extract Avro data from the HBase byte array. This configuration file can be shared among different applications that use morphlines.

Note: To function properly, the morphline must not contain a loadSolr command. The Lily HBase Indexer must load documents into Solr, instead of the morphline itself.

Procedure

You can edit the morphlines.conf file within Cloudera Manager (Key-Value Store Indexer service Configuration Category Morphlines Morphlines File).

Understanding the extractHBaseCells Morphline Command

The extractHBaseCells morphline command extracts cells from an HBase result and transforms the values into a Solr InputDocument. The command consists of an array of zero or more mapping specifications.

Each mapping has:

- The inputColumn parameter, which specifies the data from HBase for populating a field in Solr. It has the form of a column family name and qualifier, separated by a colon. The qualifier portion can end in an asterisk, which is interpreted as a wildcard. In this case, all matching column-family and qualifier expressions are used. The following are examples of valid inputColumn values:
  - mycolumnfamily:myqualifier
  - mycolumnfamily:my*
  - mycolumnfamily:*
- The outputField parameter specifies the morphline record field to which to add output values. The morphline record field is also known as the Solr document field. Example: first_name.

- Dynamic output fields are enabled by the outputField parameter ending with a wildcard (*). For example:

```
inputColumn : "mycolumnfamily:*"
outputField : "belongs_to_*"
```

  In this case, if you make these puts in HBase:

```
put 'table_name' , 'row1' , 'mycolumnfamily:1' , 'foo'
put 'table_name' , 'row1' , 'mycolumnfamily:9' , 'bar'
```

  Then the fields of the Solr document are as follows:

```
belongs_to_1 : foo
belongs_to_9 : bar
```

- The type parameter defines the data type of the content in HBase. All input data is stored in HBase as byte arrays, but all content in Solr is indexed as text, so a method for converting byte arrays to the actual data type is required. The type parameter can be the name of a type that is supported by org.apache.hadoop.hbase.util.Bytes.to* (which currently includes byte[], int, long, string, boolean, float, double, short, and bigdecimal). Use type byte[] to pass the byte array through to the morphline without conversion.
  - type:byte[] copies the byte array unmodified into the record output field
  - type:int converts with org.apache.hadoop.hbase.util.Bytes.toInt
  - type:long converts with org.apache.hadoop.hbase.util.Bytes.toLong
  - type:string converts with org.apache.hadoop.hbase.util.Bytes.toString
  - type:boolean converts with org.apache.hadoop.hbase.util.Bytes.toBoolean
  - type:float converts with org.apache.hadoop.hbase.util.Bytes.toFloat
  - type:double converts with org.apache.hadoop.hbase.util.Bytes.toDouble
  - type:short converts with org.apache.hadoop.hbase.util.Bytes.toShort
  - type:bigdecimal converts with org.apache.hadoop.hbase.util.Bytes.toBigDecimal

  Alternatively, the type parameter can be the name of a Java class that implements the com.ngdata.hbaseindexer.parse.ByteArrayValueMapper interface.

  HBase data formatting does not always match what is specified by org.apache.hadoop.hbase.util.Bytes.*. For example, this can occur with data of type float or double. You can enable indexing of such HBase data by converting the data. There are various ways to do so, including:

  - Using Java morphline command to parse input data, converting it to the expected output. For example:

```
{
 imports : "import java.util.*;" code: """ // manipulate the contents of
 a record field
 String stringAmount = (String) record.getFirstValue("amount");
 Double dbl = Double.parseDouble(stringAmount); record.replaceValues("
amount",dbl);
 return child.process(record); // pass record to next command in chain
 """
}
```

  - Creating table fields with binary format and then using types such as double or float in a morphline.conf. You could create a table in HBase for storing doubles using commands similar to:

```
CREATE TABLE sample_lily_hbase ( id string, amount double, ts timestamp
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,ti:amount#b,ti:
ts,')
TBLPROPERTIES ('hbase.table.name' = 'sample_lily');
```

- The source parameter determines which portion of an HBase KeyValue is used as indexing input. Valid choices are value or qualifier. When value is specified, the HBase cell value is used as input for indexing. When qualifier is specified, then the HBase column qualifier is used as input for indexing. The default is value.

Running the HBaseMapReduceIndexerTool

HBaseMapReduceIndexerTool is a MapReduce batch job driver that takes input data from an HBase table, creates Solr index shards, and writes the indexes to HDFS or local file system in a flexible, scalable, and fault-tolerant manner. It also supports merging the output shards into a set of live customer-facing Solr servers in SolrCloud.

Before you begin

Important: Merging output shards into live customer-facing Solr servers can only be completed if all replicas are online.

Important: You must run the indexer tool with the following command-line argument:

```
-D 'mapreduce.job.user.classpath.first=true'
```

Running the tool without this argument triggers the following error:

```
ERROR [main] org.apache.hadoop.mapred.YarnChild: Error running child :
 java.lang.NoSuchMethodError:
com.codahale.metrics.MetricRegistry.meter(Ljava/lang/String;Lcom/cod
ahale/metrics/MetricRegistry$MetricSupplier;)Lcom/codahale/metrics/M
eter;
```

Note:

For HBase batch indexing to work with JDK 17, you also need to add the following Hadoop option:

```
HADOOP_OPTS="--add-opens java.base/java.nio=ALL-UNNAMED"
```

Procedure

- Run the command as follows:

```
hadoop --config /etc/hadoop/conf \
jar /opt/cloudera/parcels/CDH/lib/hbase-solr/tools/hbase-indexer-mr-*-job
.jar \
--conf /etc/hbase/conf/hbase-site.xml -D 'mapreduce.job.user.classpath.fi
rst=true' \
-Dmapreduce.map.java.opts="-Xmx512m" -Dmapreduce.reduce.java.opts="-Xmx5
12m" \
--hbase-indexer-file $HOME/morphline-hbase-mapper.xml \
--zk-host 127.0.0.1/solr --collection hbase-collection1 \
```

```
--go-live --log4j src/test/resources/log4j.properties
```

Note: For development purposes, use the --dry-run option to run in local mode and print documents to stdout, instead of loading them to Solr. Using this option causes the morphline to run in the client process without submitting a job to MapReduce. Running in the client process provides quicker results during early trial and debug sessions.

To print diagnostic information, such as the content of records as they pass through morphline commands, enable TRACE log level diagnostics by adding the following to your log4j.properties file:

```
log4j.logger.org.kitesdk.morphline=TRACE
log4j.logger.com.ngdata=TRACE
```

The log4j.properties file can be passed using the --log4j command-line option.

To invoke the command-line help, use:

```
hadoop jar /opt/cloudera/parcels/CDH/jars/hbase-indexer-mr-*-job.jar --h
elp
```

Related reference

HBaseMapReduceIndexerTool command line reference

HBaseMapReduceIndexerTool command line reference

Command line syntax, examples and list of parameters.

Important:

If Solr uses a custom system user, you need to pass this username to MapReduceIndexerTool jobs using the following argument:

```
-Dsolr.authorization.superuser=<<customSolrSystemUser>>
```

This is necessary for setting up ACLs of the MapReduce jobs directory correctly.

The general command line syntax is:

```
command [genericOptions] [commandOptions]
```

usage:

```
hadoop [GenericOptions]... jar hbase-indexer-mr-*-job.jar
       [--hbase-indexer-zk STRING] [--hbase-indexer-name STRING]
       [--hbase-indexer-file FILE]
       [--hbase-indexer-component-factory STRING]
       [--hbase-table-name STRING] [--hbase-start-row BINARYSTRING]
       [--hbase-end-row BINARYSTRING] [--hbase-start-time STRING]
       [--hbase-end-time STRING] [--hbase-timestamp-format STRING]
       [--zk-host STRING] [--go-live] [--collection STRING]
       [--go-live-min-replication-factor INTEGER]
       [--go-live-threads INTEGER] [--go-live-timeout INTEGER]
       [--filesystem STRING] [--private-key FILE]
       [--known-hosts FILE] [--local-merge-dir DIR]
       [--keytab FILE] [--help] [--output-dir HDFS_URI]
       [--overwrite-output-dir] [--morphline-file FILE]
       [--morphline-id STRING] [--solr-home-dir DIR]
       [--update-conflict-resolver FQCN] [--reducers INTEGER]
       [--max-segments INTEGER] [--fair-scheduler-pool STRING] [--dry-run]
       [--log4j FILE] [--verbose] [--clear-index] [--show-non-solr-cloud]
```

Examples:

(Re)index a table in GoLive mode based on a local indexer config file:

```
hadoop --config /etc/hadoop/conf \
  jar hbase-indexer-mr-*-job.jar \
  --conf /etc/hbase/conf/hbase-site.xml \
  -D 'mapreduce.job.user.classpath.first=true' \
  -Dmapreduce.map.java.opts="-Xmx512m" \
  -Dmapreduce.reduce.java.opts="-Xmx512m" \
  --hbase-indexer-file indexer.xml \
  --zk-host 127.0.0.1/solr \
  --collection collection1 \
  --go-live \
  --log4j src/test/resources/log4j.properties
```

(Re)index a table in GoLive mode using a local morphline-based indexer config file. Also include extra library jar file containing JSON tweet Java parser:

```
hadoop --config /etc/hadoop/conf \
  jar hbase-indexer-mr-*-job.jar \
  --conf /etc/hbase/conf/hbase-site.xml \
  --libjars /path/to/kite-morphlines-twitter-0.10.0.jar \
  -D 'mapreduce.job.user.classpath.first=true' \
  -Dmapreduce.map.java.opts="-Xmx512m" \
  -Dmapreduce.reduce.java.opts="-Xmx512m" \
  --hbase-indexer-file src/test/resources/morphline_indexer_without_zk.xml \
  --zk-host 127.0.0.1/solr \
  --collection collection1 \
  --go-live \
  --morphline-file src/test/resources/morphlines.conf \
  --output-dir hdfs://c2202.mycompany.com/user/$USER/test \
  --overwrite-output-dir \
  --log4j src/test/resources/log4j.properties
```

(Re)index a table in GoLive mode:

```
hadoop --config /etc/hadoop/conf \
  jar hbase-indexer-mr-*-job.jar \
  --conf /etc/hbase/conf/hbase-site.xml \
  -D 'mapreduce.job.user.classpath.first=true' \
  -Dmapreduce.map.java.opts="-Xmx512m" \
  -Dmapreduce.reduce.java.opts="-Xmx512m" \
  --hbase-indexer-file indexer.xml \
  --zk-host 127.0.0.1/solr \
  --collection collection1 \
  --go-live \
  --log4j src/test/resources/log4j.properties
```

(Re)index a table with direct writes to SolrCloud:

```
hadoop --config /etc/hadoop/conf \
  jar hbase-indexer-mr-*-job.jar \
  --conf /etc/hbase/conf/hbase-site.xml \
  -D 'mapreduce.job.user.classpath.first=true' \
  -Dmapreduce.map.java.opts="-Xmx512m" \
  -Dmapreduce.reduce.java.opts="-Xmx512m" \
  --hbase-indexer-file indexer.xml \
  --zk-host 127.0.0.1/solr \
  --collection collection1 \
  --reducers 0 \
  --log4j src/test/resources/log4j.properties
```

(Re)index a table based on a indexer config stored in ZK:

```
hadoop --config /etc/hadoop/conf \
  jar hbase-indexer-mr-*-job.jar \
  --conf /etc/hbase/conf/hbase-site.xml \
  -D 'mapreduce.job.user.classpath.first=true' \
  -Dmapreduce.map.java.opts="-Xmx512m" \
  -Dmapreduce.reduce.java.opts="-Xmx512m" \
  --hbase-indexer-zk zk01 \
  --hbase-indexer-name docindexer \
  --go-live \
  --log4j src/test/resources/log4j.properties
```

MapReduce on Yarn - Pass custom JVM arguments:

```
HADOOP_CLIENT_OPTS='-DmaxConnectionsPerHost=10000 -DmaxConnections=10000'; \
hadoop --config /etc/hadoop/conf \
  jar hbase-indexer-mr-*-job.jar \
  --conf /etc/hbase/conf/hbase-site.xml \
  -D 'mapreduce.map.java.opts=-DmaxConnectionsPerHost=10000 -DmaxConnection
s=10000' \
  -D 'mapreduce.reduce.java.opts=-DmaxConnectionsPerHost=10000 -DmaxConne
ctions=10000' \
  --hbase-indexer-zk zk01 \
  --hbase-indexer-name docindexer \
  --go-live \
  --log4j src/test/resources/log4j.properties
```

HBase Indexer Parameters

Parameters for specifying the HBase indexer definition and/or where it should be loaded from.

Table 2: HBase Indexer parameters

Parameter	Type	Description	Example
--hbase-indexer-zk	STRING	The address of the ZooKeeper ensemble from which to fetch the indexer definition named --hbase-indexer-name. Format is: a list of comma separated host:port pairs, each corresponding to a zk server.	'127.0.0.1:2181,127.0.0.1:2182,127.0.0.1:2183'
--hbase-indexer-name	STRING	The name of the indexer configuration to fetch from the ZooKeeper ensemble specified with --hbase-indexer-zk.	myIndexer
--hbase-indexer-file	FILE	Relative or absolute path to a local HBase indexer XML configuration file. If supplied, this overrides --hbase-indexer-zk and --hbase-indexer-name.	/path/to/morphline-hbase-mapper.xml
--hbase-indexer-component-factory	STRING	Classname of the hbase indexer component factory.	

HBase Scan Parameters

Parameters for specifying what data is included while reading from HBase.

Table 3: HBase scan parameters

Parameter	Type	Description	Example
--hbase-table-name	STRING	Optional name of the HBase table containing the records to be indexed. If supplied, this overrides the value from the --hbase-inde xer-* options.	myTable
--hbase-start-row	BINARYSTRING	Binary string representation of start row from which to start indexing (inclusive). The format of the supplied row key should use two-digit hex values prefixed by \x for non-ascii characters (e.g. 'row\x00'). The semantics of this argument are the same as those for the HBase Scan#setStartRow method. The default is to include the first row of the table.	AAAA
--hbase-end-row	BINARYSTRING	Binary string representation of end row prefix at which to stop indexing (exclusive). See the description of --hbase-start-row for more information. The default is to include the last row of the table.	CCCC
--hbase-start-time	STRING	Earliest timestamp (inclusive) in time range of HBase cells to be included for indexing. The default is to include all cells.	0
--hbase-end-time	STRING	Latest timestamp (exclusive) of HBase cells to be included for indexing. The default is to include all cells.	123456789
--hbase-timestamp-format	STRING	Timestamp format to be used to interpret --hbase- start-time and --hbase-end-time. This is a java .text.SimpleDateFormat compliant format. If this parameter is omitted then the timestamps are interpreted as number of milliseconds since the standard epoch (Unix time).	yyyy-MM-dd'T'HH:mm:ss.SSSZ

Solr Cluster Arguments

Arguments that provide information about your Solr cluster.

Table 4: Solr cluster arguments

Argument	Type	Description	Example
--zk-host	STRING	The address of a ZooKeeper ensemble being used by a SolrCloud cluster. This ZooKeeper ensemble will be examined to determine the number of output shards to create as well as the Solr URLs to merge the output shards into when using the --go-live option. Requires that you also pass the --collection to merge the shards into. The format is a list of comma separated host:port pairs, each corresponding to a zk server. The --zk-host option implements the same partitioning semantics as the standard SolrCloud Near-Real-Time (NRT) API. This enables to mix batch updates from MapReduce ingestion with updates from standard Solr NRT ingestion on the same SolrCloud cluster, using identical unique document keys. If --solr-home-dir is not specified, the Solr home directory for the collection will be downloaded from this ZooKeeper ensemble.	'127.0.0.1:2181,127.0.0.1:2182,127.0.0.1:2183' If the optional chroot suffix is used the example would look like: '127.0.0.1:2181/solr127.0.0.1:2182/solr127.0.0.1:2183/solr' where the client would be rooted at '/solr' and all paths would be relative to this root - i.e.: getting/setting/etc... '/foo/bar' would result in operations being run on '/solr/foo/bar' (from the server perspective).
--solr-client-socket-timeout	INTEGER	Solr socket timeout in milliseconds This optional argument overwrites the default 10 minute socket timeout in HBase indexer for the direct writing mode (when the value of the --reducers optional argument is set to 0 and mappers directly send the data to the live Solr). Default value: 600000	

Go Live Arguments

Arguments for merging the shards that are built into a live Solr cluster. Also see the Cluster arguments.

Table 5: Go live arguments

Argument	Type	Description	Example
--go-live		Allows you to optionally merge the final index shards into a live Solr cluster after they are built. You can pass the ZooKeeper address with --zk-host and the relevant cluster information will be auto detected. (default: false)	
--collection	STRING	The SolrCloud collection to merge shards into when using --go-live and --zk-host.	collection1

Argument	Type	Description	Example
--go-live-min-replication-factor	INTEGER	The minimum number of SolrCloud replicas to successfully merge any final index shard into. The go-live job phase attempts to merge final index shards into all SolrCloud replicas. Some of these merge operations may fail, for example if some SolrCloud servers are down. This option enables indexing jobs to succeed even if some such merge operations fail on SolrCloud followers. Successful merge operations into all leaders are always required for job success, regardless of the value of --go -live-min- replication-factor. -1 indicates require successful merge operations into all replicas. 1 indicates require successful merge operations only into leader replicas. (default: -1)	
--go-live-threads	INTEGER	Tuning knob that indicates the maximum number of live merges to run in parallel at one time. (default: 1000)	
--go-live-timeout	INTEGER	Timeout in milliseconds (ms) to wait for the merge to complete before the connection times out and the tool fails.	
--filesystem	STRING	Allows you to change whether you want to merge indexes into a collection on HDFS or on localfs. Possible values are: • hdfs (default) • localfs In case of localfs make sure that the target collection uses localfs. This option also requires the --us e-zk-solrconfig.xml and --private-key arguments.	
--private-key	FILE	Path to the private key that allows the user running the MRIT job to SSH into all Solr hosting nodes. This must be provided together with the --filesystem localfs argument.	

Argument	Type	Description	Example
--known-hosts	FILE	Path to the known hosts file which contains keys to all the Solr hosting nodes. Use with the --filesystem localfs argument. **Important:** Always specify this in localfs mode in production use. If not provided, strict host checking is skipped. This is unsafe and Cloudera only recommends it for testing purposes.	
--local-merge-dir	DIR	Path to a directory on all Solr hosts to temporarily copy index to before mergeindexes action. Used only with the --filesystem localfs option. The user running the MRIT job needs to have permission to read and write into this directory. This directory needs to be relative to SOLR_HOME or SOLR_DATA_HOME or needs to be specified in the Solr system property -Dsolr.allowPaths. (default: /var/lib/solr)	
--keytab	FILE	Path to the keytab file for the user running the MRIT job on all Solr hosting nodes. Used only with the --filesystem localfs argument. If not provided, `kinit` is skipped and MRIT expects an unsecure environment or a valid kerberos ticket already present on all Solr hosting nodes for the current user.	

Table 6: Optional arguments

Argument	Type	Description	Example
--help -help -h		Show the help message and exit	
--output-dir	HDFS_URI	HDFS directory to write Solr indexes to. Inside there one output directory per shard will be generated.	hdfs://c2202.mycompany. com/user/$USER/test
--overwrite-output-dir		Overwrite the directory specified by --output-dir if it already exists. Using this parameter will result in the output directory being recursively deleted at job startup. (default: false)	

Argument	Type	Description	Example
--morphline-file	FILE	Relative or absolute path to a local config file that contains one or more morphlines. The file must be UTF-8 encoded. The file will be uploaded to each MR task. If supplied, this overrides the value from the --hbase-indexer-* options.	/path/to/morphlines.conf
--morphline-id	STRING	The identifier of the morphline that shall be executed within the morphline config file, e.g. specified by --morphline-file. If the --morphline- id option is ommitted the first (i.e. top-most) morphline within the config file is used. If supplied, this overrides the value from the --hbase-inde xer-* options.	morphline1
--solr-home-dir	DIR	Optional relative or absolute path to a local dir containing Solr conf/ dir and in particular conf /solrconfig.xml and optionally also lib/ dir. This directory will be uploaded to each MR task.	src/test/resources/solr/minimr
--update-conflict-resolver	FQCN	Fully qualified class name of a Java class that implements the UpdateConflictResolver interface. This enables deduplication and ordering of a series of document updates for the same unique document key. For example, a MapReduce batch job might index multiple files in the same job where some of the files contain old and new versions of the very same document, using the same unique document key. Typically, implementations of this interface forbid collisions by throwing an exception, or ignore all but the most recent document version, or, in the general case, order colliding updates ascending from least recent to most recent (partial) update. The caller of this interface (i. e. the Hadoop Reducer) will then apply the updates to Solr in the order returned by the orderUpdates() method. The default RetainMostRe centUpdateConflictResolver implementation ignores all but the most recent document version, based on a configurable numeric Solr field, which defaults to the file_last_modified timestamp. (default: org.apache. solr.hadoop. dedup. RetainMostRecentU pdateConflictResolver)	

Argument	Type	Description	Example
--reducers	INTEGER	Tuning knob that indicates the number of reducers to index into. • 0 indicates that no reducers should be used, and documents should be sent directly from the mapper tasks to live Solr servers. • -1 indicates use all reduce slots available on the cluster. • -2 indicates use one reducer per output shard, which disables the mtree merge MR algorithm. The mtree merge MR algorithm improves scalability by spreading load (in particular CPU load) among a number of parallel reducers that can be much larger than the number of solr shards expected by the user. It can be seen as an extension of concurrent lucene merges and tiered lucene merges to the clustered case. The subsequent mapper-only phase merges the output of said large number of reducers to the number of shards expected by the user, again by utilizing more available parallelism on the cluster. (default: -1)	
--max-segments	INTEGER	Tuning knob that indicates the maximum number of segments to be contained on output in the index of each reducer shard. After a reducer has built its output index it applies a merge policy to merge segments until there are <= maxSegments lucene segments left in this index. Merging segments involves reading and rewriting all data in all these segment files, potentially multiple times, which is very I/O intensive and time consuming. However, an index with fewer segments can later be merged faster, and it can later be queried faster once deployed to a live Solr serving shard. Set maxSegments to 1 to optimize the index for low query latency. In a nutshell, a small maxSegments value trades indexing latency for subsequently improved query latency. This can be a reasonable trade-off for batch indexing systems. (default: 1)	

Argument	Type	Description	Example
--dry-run		Run in local mode and print documents to stdout instead of loading them into Solr. This executes the morphline in the client process (without submitting a job to MR) for quicker turnaround during early trial and debug sessions. (default: false)	
--log4j	FILE	Relative or absolute path to a log4j.properties config file on the local file system. This file will be uploaded to each MR task.	/path/to/log4j.properties
--verbose -v		Turn on verbose output. (default: false)	
--clear-index		Will attempt to delete all entries in a solr index before starting batch build. This is not transactional so if the build fails the index will be empty. (default: false)	
--show-non-solr-cloud		Also show options for Non-SolrCloud mode as part of --help. (default: false)	

Supported Generic Options

The following generic options are supported:

Table 7: Supported generic options

Option	Description
--conf <configuration file>	Specify an application configuration file.
-D <property=value>	Define a value for a given property.
-fs <file:///|hdfs://namenode:port>	Specify default filesystem URL to use, overrides the fs.defaultFS property from configurations.
--jt <local|resourcemanager:port>	Specify a ResourceManager.
--files <file1,...>	Specify a comma-separated list of files to be copied to the map reduce cluster.
--libjars <jar1,...>	Specify a comma-separated list of jar files to be included in the classpath.
--archives <archive1,...>	Specify a comma-separated list of archives to be unarchived on the compute machines.

Related Information

Java SimpeDateFormat

Using --go-live with SSL or Kerberos

Establish trust between the indexer client and Solr server(s).

About this task

The go-live phase of the indexer jobs sends a MERGEINDEXES request from the indexer client (the node from which the MR job was submitted) to the live Solr servers. If the Solr server has SSL enabled, you need to ensure that the indexer client trusts the certificate presented by the Solr server(s), otherwise you get an SSLPeerUnverifiedExc ption.

Procedure

1. Specify the location of the trust store by setting the following HADOOP_OPTS variable before launching the indexer job:

```
HADOOP_OPTS="-Djavax.net.ssl.trustStore=/etc/cdep-ssl-conf/CA_STANDARD/t
ruststore.jks "
```

2. If the Solr servers have Kerberos authentication enabled, you need to ensure that the indexer client can authenticate via Kerberos to the Solr servers. For this, you need to create a Java Authentication and Authorization Service configuration (JAAS) file locally on the node where the indexing job is launched:

   - If you are authenticating using kinit to obtain credentials, you can configure the client to use your credential cache by creating a jaas.conf file with the following contents:

```
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=false
 useTicketCache=true
 principal="<USER>@EXAMPLE.COM";
 };
```

     Replace *<USER>* with your username, and *EXAMPLE.COM* with your Kerberos realm.

   - If you want the client application to authenticate using a keytab, modify jaas-client.conf as follows:

```
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=true
 keyTab="/PATH/TO/USER.KEYTAB"
 storeKey=true
 useTicketCache=false
 principal="<USER>@EXAMPLE.COM";
};
```

     Replace */PATH/TO/USER.KEYTAB* with the keytab file you want to use and *<USER>@EXAMPLE.COM* with the principal in the keytab. If you are using a service principal that includes the hostname, make sure that it is included in the jaas.conf file (for example, solr/solr01.example.com@EXAMPLE.COM).

3. If you are using a ticket cache, you need to do a kinit to acquire a ticket for the configured principal before launching the indexer.
4. Specify the authentication configuration in the HADOOP_OPTS environment variable:

   Note:

   For HBase batch indexing to work with JDK 17, you also need to add the following Hadoop option:

```
HADOOP_OPTS="--add-opens java.base/java.nio=ALL-UNNAMED"
```

```
HADOOP_OPTS="-Djava.security.auth.login.config=jaas.conf -Djavax.net.ssl
.trustStore=/etc/cdep-ssl-conf/CA_STANDARD/truststore.jks" \
hadoop --config /etc/hadoop/conf \
jar /opt/cloudera/parcels/CDH/lib/hbase-solr/tools/hbase-indexer-mr-*-jo
b.jar \
```

```
--conf /etc/hbase/conf/hbase-site.xml -Dmapreduce.map.java.opts="-Xmx512m"
 -Dmapreduce.reduce.java.opts="-Xmx512m" \
--hbase-indexer-file /home/systest/hbasetest/morphline-hbase-mapper.xml \
--zk-host 127.0.0.1/solr \
--collection hbase-collection1 \
--go-live --log4j src/test/resources/log4j.properties
```

Note:

Communication to Solr servers is only occurring in the go-live phase, not from the MapReduce jobs. Therefore it is enough to place the jaas.conf and the SSL trust store on the node from which the indexer client is started as it will be the one that communicates to Solr.

Understanding --go-live and HDFS ACLs

When run with a reduce phase, as opposed to as a mapper-only job, the indexer creates an offline index on HDFS in the output directory specified by the --output-dir parameter. If the --go-live parameter is specified, Solr merges the resulting offline index into the live running service. Thus, the Solr service must have read access to the contents of the output directory in order to complete the --go-live step. If --overwrite-output-dir is specified, the indexer deletes and recreates any existing output directory; in an environment with restrictive permissions, such as one with an HDFS umask of 077, the Solr user may not be able to read the contents of the newly created directory. To address this issue, the indexer automatically applies the HDFS ACLs to enable Solr to read the output directory contents. These ACLs are only applied if HDFS ACLs are enabled on the HDFS NameNode.

The indexer only makes ACL updates to the output directory and its contents. If the output directory's parent directories do not include the execute permission, the Solr service cannot access the output directory. Solr must have execute permissions from standard permissions or ACLs on the parent directories of the output directory.

Cloudera Runtime 7.2.18

Indexing Data Using Spark-Solr Connector

Date published: 2020-09-16
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Batch indexing to Solr using SparkApp framework.....4
Create indexer Maven project.....5
Run the spark-submit job.....8

Batch indexing to Solr using SparkApp framework

The Maven project presented here is provided as an example on using the Spark-Solr connector to batch-index data from a CSV file in HDFS into a Solr collection

The Spark-Solr connector framework comes bundled with Cloudera Search. It enables Extraction, Transformation, and Loading (ETL) of large datasets to Solr. You can use spark-submit with a Spark job to batch index HDFS files into Solr. For this you need to create a class which implements the SparkApp.RDDProcesor interface.

To use the SparkApp framework, you must create a Maven project with the spark-solr dependency.

```
<dependencies>
        <dependency>
            <groupId>com.lucidworks.spark</groupId>
            <artifactId>spark-solr</artifactId>
            <version>{LATEST_VERSION}</version>
        </dependency>
    </dependencies>
```

This project needs to have at a minimum one class, which implements the SparkApp.RDDProcessor interface. This class can be written either in Java or Scala. This documentation uses a Java class to demonstrate how to use the framework.

The SparkApp.RDDProcessor interface has three functions which need to be overwritten:

- getName()
- getOptions()
- run

getName()

The getName() function returns the short name of the application as a string. When running your spark-submit job, this is the name you pass as a parameter allowing the job to find your class.

public String getName() { return "csv"; }

getOptions()

In the getOptions() function you may specify parameters that are specific to your application. Certain parameters, for example zkHost, collection, or batchSize are present by default. You do not need to specify those here.

```
public Option[] getOptions() {
     return new Option[]{
          OptionBuilder
               .withArgName("PATH").hasArgs()
               .isRequired(true)
               .withDescription("Path to the CSV file to index")
               .create("csvPath")
     };
}
```

run

The run function is the core of the application. This returns an integer, and has two parameters, SparkConf and Comm andLine.

You can create a JavaSparkContext class with the use of the SparkConf parameter, and use this to open the CSV file as a JavaRDD<String> class:

```
JavaSparkContext jsc = new JavaSparkContext(conf);
JavaRDD<String> textFile = jsc.textFile(cli.getOptionValue("csvPath"));
```

You now have to convert these String values to SolrInputDocument, and create a JavaRDD class . To achieve this the script uses a custom-made map function which splits the CSV file upon commas and adds the records to the SolrInpu tDocument document. You must specify the schema used in the CSV file in advance.

```
JavaRDD<SolrInputDocument> jrdd = textFile.map(new Function<String, SolrInpu
tDocument>() {
            @Override
            public SolrInputDocument call(String line) throws Exception {
                 SolrInputDocument doc = new SolrInputDocument();
                 String[] row = line.split(",");

                 if (row.length != schema.length)
                       return null;
                 for (int i=0;i<schema.length;i++){
                       doc.setField(schema[i], row[i]);
                 }
                 return doc;
            }
});
```

After this, the script requires the CommandLine instance options to perform indexing:

```
String zkhost = cli.getOptionValue("zkHost", "localhost:9983");
String collection = cli.getOptionValue("collection", "collection1");
int batchSize = Integer.parseInt(cli.getOptionValue("batchSize", "100"));
```

Finally, the job indexes data into the Solr cluster:

```
SolrSupport.indexDocs(zkhost, collection, batchSize, jrdd.rdd());
```

If the function is succesfully called, 0 is returned.

Create indexer Maven project

As a prerequisite to using the SparkApp framework, you need to create a Maven project with the Spark-Solr dependency and at least one class, implementing the SparkApp.RDDProcessor interface.

About this task

You can either write a Java or a Scala class implementation. The examples show implementation with a Java class.

Procedure

1. Create the indexer Maven project.
2. Edit the .pom file, add the following spark-solr-dependency:

```
<dependencies>
        <dependency>
            <groupId>com.lucidworks.spark</groupId>
            <artifactId>spark-solr</artifactId>
            <version>[***LATEST VERSION***]</version>
            <classifier>shaded</classifier>
```

```
        </dependency>
    </dependencies>
```

Replace *[***LATEST VERSION***]* with an actual version number:

- To use Spark 2, replace it with 3.9.0.*[***CDP VERSION***]-[***BUILD***]*
- To use Spark 3, replace it with 3.9.0.3000.*[***CDP VERSION***]-[***BUILD***]*

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://m
aven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>indexer</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <repositories>
      <repository>
        <id>cdh.repo</id>
        <url>https://repository.cloudera.com/artifactory/cloudera-repos/</
url>
        <name>Cloudera Repositories</name>
        <snapshots>
          <enabled>true</enabled>
        </snapshots>
      </repository>
    </repositories>

    <dependencies>
        <dependency>
            <groupId>com.lucidworks.spark</groupId>
            <artifactId>spark-solr</artifactId>
            <version>3.9.0.3000.3.4.7190.1-123</version>
            <classifier>shaded</classifier>
        </dependency>
    </dependencies>

</project>
```

3. Create a CSVIndexer.java file that implements the SparkApp.RDDProcessor interface in the main/java/com/lucidworks/spark directory.
   For example:

```
package com.lucidworks.spark;

import com.lucidworks.spark.SparkApp;
import com.lucidworks.spark.util.SolrSupport;
import shaded.apache.commons.cli.CommandLine;
import shaded.apache.commons.cli.Option;
import shaded.apache.commons.cli.OptionBuilder;
import org.apache.solr.common.SolrInputDocument;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
```

```
import org.apache.spark.api.java.function.Function;

public class CSVIndexer implements SparkApp.RDDProcessor {
    @Override
    public String getName() {
        return "csv";
    }

    @Override
    public Option[] getOptions() {
        return new Option[]{
                OptionBuilder
                        .withArgName("PATH").hasArgs()
                        .isRequired(true)
                        .withDescription("Path to the CSV file to index")
                        .create("csvPath")
        };
    }

    private String[] schema = "vendor_id,pickup_datetime,dropoff_datetim
e,passenger_count,trip_distance,pickup_longitude,pickup_latitude,rate_co
de_id,store_and_fwd_flag,dropoff_longitude,dropoff_latitude,payment_type
,fare_amount,extra,mta_tax,tip_amount,tolls_amount,improvement_surcharge
,total_amount".split(",");

    @Override
    public int run(SparkConf conf, CommandLine cli) throws Exception {
        JavaSparkContext jsc = new JavaSparkContext(conf);
        JavaRDD<String> textFile = jsc.textFile(cli.getOptionValue("csvP
ath"));
        JavaRDD<SolrInputDocument> jrdd = textFile.map(new Function<Stri
ng, SolrInputDocument>() {
            @Override
            public SolrInputDocument call(String line) throws Exception {
                SolrInputDocument doc = new SolrInputDocument();
                String[] row = line.split(",");
                if (row.length != schema.length)
                    return null;
                for (int i=0;i<schema.length;i++){
                    doc.setField(schema[i], row[i]);
                }
                return doc;
            }
        });

        String zkhost = cli.getOptionValue("zkHost", "localhost:9983");
        String collection = cli.getOptionValue("collection", "collection
1");
        int batchSize = Integer.parseInt(cli.getOptionValue("batchSize",
 "100"));

        SolrSupport.indexDocs(zkhost, collection, batchSize, jrdd.rdd());

        return 0;
    }
}
```

4. Create a JAR file:

```
mvn clean install
```

The indexer.jar file is created.

Run the spark-submit job

After you create an indexer.jar file, you need to run a spark-submit job on a Solr worker node to index your input file.

Before you begin

- You have prepared the indexer.jar file and it is available on your local machine.
- A DDE Data Hub cluster is up and running.
- You have sufficient rights to SSH into one of the cluster nodes.
- Your user has a role assigned that provides 'write' rights on S3.
- You have retrieved the keytab for your environment.

Procedure

1. SSH to one of the worker nodes in your Data Hub cluster.
2. Copy your keytab file to the working directory:

```
scp [***KEYTAB***] [***USER***]@[***IP OF WORKER NODE***]:/[***PATH/TO/
WORKING/DIRECTORY***]
```

   For example:

```
scp sampleuser.keytab sampleuser@1.1.1.1:/tmp
```

3. Create a JAAS file with the following content:

```
Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  useTicketCache=false
  doNotPrompt=true
  debug=true
  keyTab="SAMPLEUSER.KEYTAB"
  principal="SAMPLEUSER@EXAMPLE.COM";
};
```

   Replace *SAMPLEUSER@EXAMPLE.COM* with your user principal.
4. Copy the indexer JAR file to the working directory:

```
scp [***INDEXER***].jar [***USER***]@[***IP OF WORKER NODE***]:/[***PATH/
TO/WORKING/DIRECTORY***]
```

   For example:

```
scp indexer-1.0-SNAPSHOT.jar sampleuser@1.1.1.1:/tmp
```

5. Copy the input CSV file to the working directory:

```
scp [***INPUT FILE***] [***USER***]@[***IP OF WORKER NODE***]:/[***PATH/
TO/WORKING/DIRECTORY***]
```

   For example:

```
scp nyc_yellow_taxi_sample_1k.csv sampleuser@1.1.1.1:/tmp
```

6. Add the input file to HDFS:

```
hdfs dfs -put [***INPUT FILE***]
```

For example:

```
hdfs dfs -put nyc_yellow_taxi_sample_1k.csv
```

7. Create a Solr configuration and a collection:

```
solrctl config --create [***CONFIG NAME***] [***BASE CONFIG***] -p immu
table=false
solrctl collection --create [***COLLECTION NAME***] -s [***NUMBER OF
 SHARDS***] -c [***COLLECTION CONFIG NAME***]
```

For example:

```
solrctl config --create testConfig managedTemplate -p immutable=false
solrctl collection --create testcollection -s 2 -c testConfig
```

8. Submit your spark job:

```
spark-submit --jars /opt/cloudera/parcels/CDH/jars/[***SPARK-SOLR-*-
SHADED.JAR***] \
--files [***KEYTAB***],[***JAAS CONFIGURATION FILE***] --name [***SPARK
 JOB NAME***] \
--driver-java-options="-Djavax.net.ssl.trustStoreType=[***TRUSTSTORE
 TYPE***] \
-Djavax.net.ssl.trustStore=[***ABSOLUTE/PATH/TO/TRUSTSTORE/FILE***] \
-Djavax.net.ssl.trustStorePassword=" --class com.lucidworks.spark.SparkAp
p [***INDEXER.JAR***] csv -zkHost [***ZOOKEEPER ENSEMBLE***] \
-collection [***TARGET SOLR COLLECTION***] -csvPath [***INPUT CSV FILE***]
 \
```

```
-solrJaasAuthConfig=[***JAAS CONFIGURATION FILE***]
```

Note:

To use Spark-Solr with an SSL-enabled ZooKeeper, you need to add extra Java options. Add the following JVM parameters as both driver and executor java options:

```
-Dzookeeper.client.secure=true \
-Dzookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocket
Netty \
-Dzookeeper.ssl.trustStore.location=[***TRUSTSTORE_LOCATION***] \
-Dzookeeper.ssl.trustStore.password=[***TRUSTSTORE_PASSWORD***] \
```

Replace *[***TRUSTSTORE_LOCATION***]* with the Java truststore location and *[***TRUSTSTORE_PASSWORD***]* with the Java truststore password.

Replace

[SPARK-SOLR-*-SHADED.JAR***]***

with the name of the shaded.jar file under /opt/cloudera/parcels/CDH/jars/

[KEYTAB***]***

with the keytab file of your user

[JAAS CONFIGURATION FILE***]***

with the JAAS file you created

[SPARK JOB NAME***]***

with the name of the job you want to run

[TRUSTSTORE TYPE***]***

with the type of the truststore used. If you use the default jks type, you do not need to specify -Dja vax.net.ssl.trustStoreType. In every other case it is mandatory.

[ABSOLUTE/PATH/TO/TRUSTSTORE/FILE***]***

with the absolute path to the truststore file

[INDEXER.JAR***]***

with the indexer.jar file you created

[ZOOKEEPER ENSEMBLE***]***

with the address of the ZooKeeper ensemble used by the Solr cluster.

[TARGET SOLR COLLECTION***]***

with the name of the Solr collection you created

[INPUT CSV FILE***]***

with the name of the file that you want to index into the *[***TARGET SOLR COLLECTION***]*

For example:

```
spark-submit --jars /opt/cloudera/parcels/CDH/jars/spark-solr-3.9.0.spar
k3.7.2.18.0-33-shaded.jar \
--files sampleuser.keytab,jaas-client.conf --name spark-solr \
--driver-java-options="-Djavax.net.ssl.trustStoreType=bcfks \
-Djavax.net.ssl.trustStore=/var/lib/cloudera-scm-agent/agent-cert/cm-auto-
global_truststore.jks \
-Djavax.net.ssl.trustStorePassword=" --class com.lucidworks.spark.SparkApp
 indexer-1.0-SNAPSHOT.jar csv \
-zkHost sampleuser-leader2.sampleuser.work:2181,sampleuser.work:2181,sa
mpleuser-master7.work:2181/solr-dde \
-collection testcollection -csvPath nyc_yellow_taxi_sample_1k.csv
```

Related Information

Retrieving keytabs for workload users
Generating collection configuration using configs
Creating a Solr collection
Enabling ZooKeeper SSL/TLS for Solr

Cloudera Runtime 7.2.18

Managing Cloudera Search

Date published: 2019-11-19
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Viewing and modifying log levels for Cloudera Search and related services 5

Viewing and modifying Solr configuration using Cloudera Manager 5
- Setting the Solr Critical State Cores Percentage parameter 5
- Setting the Solr Recovering Cores Percentage parameter 6

Managing collection configuration using configs or instance directories 6
- Cloudera Search config templates 7
- Generating collection configuration using configs 7
- Securing configs with ZooKeeper ACLs and Ranger 8
- Generating Solr collection configuration using instance directories 9
- Modifying a collection configuration generated using an instance directory 10
- Converting instance directories to configs 11
- Using custom JAR files with Cloudera Search 12
- Retrieving the clusterstate.json file 13

Managing collections in Cloudera Search 14
- Creating a Solr collection 14
- Viewing existing collections 15
- Deleting all documents in a collection 15
- Deleting a collection 16
- Updating the schema in a collection 16

Creating a replica of an existing shard 17

Migrating Solr replicas 17

Splitting a shard on HDFS 20

Backing up a collection from HDFS 20

Backing up a collection from local file system 22

Restoring a collection 25

Defining a backup target in solr.xml 27

Cloudera Search log files **28**

Cloudera Search configuration files **29**

Viewing and modifying log levels for Cloudera Search and related services

Learn about viewing and modifying log levels for Cloudera Search and related services, such as Apache HBase.

Procedure

1. Go to the service configuration page. For example: Solr service Configuration .
2. Select the CATEGORY Logs filter.
3. Modify the logging threshold for the service roles. Available options (in descending verbosity) are:
   - TRACE
   - DEBUG
   - INFO
   - WARN
   - ERROR
   - FATAL
4. Enter a Reason for change, and click Save Changes.
5. Restart the Solr service (Solr service Actions Restart) and any dependent services.

Viewing and modifying Solr configuration using Cloudera Manager

Learn about viewing and editing configuration parameters for the Infra and Workload Solr services in Cloudera Manager.

Procedure

1. Go to Solr service Configuration .
2. Use the SCOPE and CATEGORY filters to restrict the displayed configuration parameters. You can also enter text in the Search field to dynamically filter configuration parameters.
3. After making changes, enter a Reason for change, and click Save Changes.
4. Restart the Solr service (Solr service Actions Restart) and any dependent services. You can also restart dependent services using the Restart Stale Services wizard.

Setting the Solr Critical State Cores Percentage parameter

Configure this health check for the Solr service to receive information about the status of the cores hosted on different hosts.

About this task

By default the Solr Critical State Cores Percentage check reports 'bad health' if any of the hosted cores are in 'down' or 'recovery failed' status. This threshold can be modified in the configurations with the solr_critical_core_thresholds configuration parameter.

The value of solr_critical_core_thresholds determines the minimum percentage of Solr cores which need to be in 'critical' state in order for this check to report bad health. For example, with solr_critical_core_thresholds set to 10 (percent) and with 20 Solr cores in total you will not get bad health reported until you have 2 or more cores in 'critical' state.

This check is enabled by default for the Infra Solr service but disabled by default for the Workload Solr services (Cloudera Search).

For information on configuring this parameter, see Viewing and modifying Solr configuration using Cloudera Manager.

Related Information

Solr Critical State Cores Percentage

Setting the Solr Recovering Cores Percentage parameter

This health check for the Solr service gives information about the number of recovering cores hosted on different hosts.

About this task

By default this check reports concerning health if any of the hosted cores are in recovering status. This threshold can be modified in the configurations with the solr_recovering_core_thresholds configuration parameter.

The value of solr_recovering_core_thresholds determines the minimum percentage of Solr cores which need to be in 'recovering' state in order for this check to report 'concerning health'. For example, with solr_recovering_core_thresholds set to 10 (percent) and with 20 Solr cores in total you do not get 'concerning health' reported until you have 2 or more cores in recovering state.

This check is enabled by default for the Infra Solr service but disabled by default for the Workload Solr services (Cloudera Search).

For information on configuring this parameter, see Viewing and modifying Solr configuration using Cloudera Manager.

Related Information

Solr Recovering Cores Percentage

Managing collection configuration using configs or instance directories

Both configs and instance directories are configuration sets for Solr collections that can be referenced by their respective names. Although configs and instance directories are functionally identical from the perspective of the Solr server, there are a number of important administrative differences between these two implementations.

The solrctl utility includes the config and instancedir commands for managing configuration. Configs and instance directories refer to the same thing: named configuration sets used by collections, as specified by the solrctl collection --create -c *[***CONFIG NAME***]* command.

Table 1: Config and instance directory comparison

Attribute	Config	Instance Directory
Security	• In a Kerberos-enabled cluster, the ZooKeeper znodes associated with configurations created using the solrctl config command automatically have proper ZooKeeper ACLs.	• No ZooKeeper security support. Any user can create, delete, or modify an instance dir directly in ZooKeeper. • Because instancedir updates ZooKeeper directly, it is the client's responsibility to add the proper ACLs, which can be cumbersome.
Creation method	Generated from existing configs or instance directories in ZooKeeper using the ConfigSets API.	Manually edited locally and re-uploaded directly to ZooKeeper using solrctl utility.

Attribute	Config	Instance Directory
Template support	• Predefined templates are available. These can be used as the basis for creating additional configs. Additional templates can be created by creating configs that are immutable. • Mutable configs that use a managed schema can only be modified using the Schema API as opposed to being manually edited. As a result, configs are less flexible, but they are also less error-prone than instance directories.	One standard template.

Cloudera Search config templates

Config templates are immutable configuration templates that you can use as a starting point when creating configs for Solr collections. Cloudera Search contains templates by default and you can define new ones based on existing configs.

Configs can be declared as immutable, which means they cannot be deleted or have their Schema updated by the Schema API. Immutable configs are uneditable config templates that are the basis for additional configs. After a config is made immutable, you cannot change it back without accessing ZooKeeper directly as the solr (or solr@EXAMPLE.COM principal, if you are using Kerberos) super user.

Solr provides a set of immutable config templates. These templates are only available after Solr initialization, so templates are not available in upgrades until after Solr is initialized or re-initialized. Templates include:

Table 2: Available Config Templates and Attributes

Template Name	Supports Schema API	Uses Schemaless Solr
managedTemplate	Yes	No
schemalessTemplate	Yes	Yes

Note: schemalessTemplate is the same as the template generated by the solrctl instancedir --generate command.

Config templates are managed using the solrctl config command. For example:

- To create a new config based on the managedTemplate template:

  ```
  solrctl config --create [***NEW CONFIG***] managedTemplate -p immutable=false
  ```

  Replace *[***NEW CONFIG***]* with the name of the config you want to create.
- To create a new template (immutable config) from an existing config:

  ```
  solrctl config --create [***NEW TEMPLATE***] [***EXISTING CONFIG***] -p immutable=true
  ```

  Replace *[***NEW TEMPLATE***]* with a name for the new template you want to create and *[***EXISTING CONFIG***]* with the name of the existing config that you want to base *[***NEW TEMPLATE***]* on.

Generating collection configuration using configs

You must create a collection configuration prior to creating a Solr collection. The configuration files are created in ZooKeeper based on existing templates using the ConfigSets API. Learn how to create one using configs.

About this task

Configs are named configuration sets that you can reference when creating collections.

You can manage configuration objects directly using the solrctl config command, which is a wrapper script for the Solr ConfigSets API.

```
solrctl config --create [***NEW CONFIG***] [***TEMPLATE***] [-
p [***NAME***]=[***VALUE***]]
```

Procedure

1. If you are using Kerberos, kinit as a user with permission to create the collection configuration:

   ```
   kinit solradmin@EXAMPLE.COM
   ```

   Replace EXAMPLE.COM with your Kerberos realm name.
2. To generate configuration files for a collection, run the following command:

   ```
   solrctl config --create [***NEW CONFIG***] [***TEMPLATE***] -p immutabl
   e=false
   ```

   where

   [NEW CONFIG***]***

   is the user-specified name of the config

   [TEMPLATE***]***

   is the name of an existing config template

   To list all available config templates, use the solrctl instancedir --list command.

   -p ***[***NAME***]=[***VALUE***]***

   Overrides a *[***TEMPLATE***]* setting. The only config property that you can override is immu table, so the possible options are -p immutable=true and -p immutable=false. If you are copying an immutable config, such as a template, use -p immutable=false to make sure that you can edit the new config.

   For example, to create the configuration logs_config based on managedTemplate:

   ```
   solrctl config --create logs_config managedTemplate -p immutable=false
   ```

Securing configs with ZooKeeper ACLs and Ranger

Learn how you can restrict access to configuration sets by setting ZooKeeper Acces control Lists (ACLs) on all znodes under and including the /solr directory and using Ranger to control access to the ConfigSets API.

Before you begin

Ranger requires Kerberos authentication.

About this task

The solrctl instancedir command interacts directly with ZooKeeper, and therefore cannot be protected by Ranger. Because the solrctl config command is a wrapper script for the ConfigSets API, it can be protected by Ranger.

To force users to use the ConfigSets API, you must set all ZooKeeper znodes under and including /solr to read-only (except for the solr user).

After completing these steps, you cannot run commands such as solrctl instancedir --create or solrctl instancedir --delete without first authenticating as the solr@EXAMPLE.COM super user principal. Unauthenticated users can

still run solrctl instancedir --list and solrctl instancedir --get, because those commands only perform read operations against ZooKeeper.

Procedure

1. Create a jaas.conf file containing the following:

```
Client {
           com.sun.security.auth.module.Krb5LoginModule required
           useKeyTab=false
           useTicketCache=true
           principal="solr@[***EXAMPLE.COM***]";
           };
```

   Replace *[***EXAMPLE.COM***]* with your Kerberos realm name.

2. Set the LOG4J_PROPS environment variable so that it points to a log4j.properties file:

```
export LOG4J_PROPS=/etc/zookeeper/conf/log4j.properties
```

3. Set the ZKCLI_JVM_FLAGS environment variable:

```
export ZKCLI_JVM_FLAGS="-Djava.security.auth.login.config=[***PATH TO
 JAAS.CONF FILE***] \
             -DzkACLProvider=org.apache.solr.common.cloud.SaslZkACLProvid
er \
             -Droot.logger=INFO,console"
```

   Replace *[***PATH TO JAAS.CONF FILE***]* with the path pointing to the jaas.conf file you just created.

4. Authenticate as the solr user:

```
kinit solr@[***EXAMPLE.COM***]
```

   Replace *[***EXAMPLE.COM***]* with your Kerberos realm name.

5. Run the zkcli.sh script as follows:

```
/opt/cloudera/parcels/CDH/lib/solr/bin/zkcli.sh -zkhost [***ZOOKEEPER
 SERVER HOSTNAME***]:2181 -cmd updateacls /solr
```

   Replace *[***ZOOKEEPER SERVER HOSTNAME***]* with the hostname of a ZooKeeper server.

Generating Solr collection configuration using instance directories

You must create a collection configuration prior to creating a Solr collection. The configuration files for a Solr collection are stored in a directory called instance directories. Learn how to create the directory and make it available to Solr by uploading the contents to Zookeeper.

Before you begin

Note:

If you want to control access to configuration sets, you must enable ZooKeeper ACLs and use configs instead.

Important: Although you can create a collection directly in /var/lib/solr, Cloudera recommends using the solrctl utility instead.

About this task

In this case, configuration files for a collection are contained in a directory called an instance directory. An instance directory is a named set of configuration files. You can generate an instance directory template locally, edit the configuration, and then upload the directory to ZooKeeper as a named configuration set. You can then reference this named configuration set when creating a collection.

Procedure

1. If you are using Kerberos, kinit as a user with permission to create the collection configuration:

   ```
   kinit solradmin@[***EXAMPLE.COM***]
   ```

   Replace *[***EXAMPLE.COM***]* with your Kerberos realm name.
2. To generate a template instance directory, run the following command:

   ```
   solrctl instancedir --generate $HOME/solr_configs
   ```
3. Customize the collection by directly editing the solrconfig.xml and schema.xml files created in $HOME/solr_configs/conf.
4. After completing the configuration, make it available to Solr by running the following command, which uploads the contents of the instance directory to ZooKeeper:

   ```
   solrctl config --upload [***COLLECTION_NAME***] $HOME/solr_configs
   ```

   For example:

   ```
   solrctl config --upload WEBLOGS $HOME/solr_configs
   ```
5. Use the solrctl utility to verify that your instance directory uploaded successfully and is available to ZooKeeper. List the uploaded instance directories as follows:

   ```
   solrctl instancedir --list
   ```

   If you used the --create command to create a collection named weblogs, the --list command returns weblogs.

Related Information

solrctl Reference

Modifying a collection configuration generated using an instance directory

The configuration files for a Solr collection are stored in a directory called instance directories. Learn how to modify the directory and make it available to Solr by uploading the contents to ZooKeeper.

About this task

In this case, configuration files for a collection are contained in a directory called an instance directory. An instance directory is a named set of configuration files. You can download and edit the configuration locally and then upload the directory to ZooKeeper as a named configuration set. If your instance directory is already referenced by a collection, make sure to reload it for configuration changes to take effect.

Procedure

1. If you are using Kerberos, kinit as a user with permission to create the collection configuration:

   ```
   kinit solradmin@[***EXAMPLE.COM***]
   ```

   Replace *[***EXAMPLE.COM***]* with your Kerberos realm name.

2. To download a template instance directory, run the following command:

```
solrctl instancedir --get $HOME/solr_configs
```

3. Customize the collection by directly editing the solrconfig.xml and schema.xml files.
4. After completing the configuration, make it available to Solr by running the following command, which uploads the contents of the instance directory to ZooKeeper:

```
solrctl config --upload [***COLLECTION_NAME***] $HOME/solr_configs
```

For example:

```
solrctl config --upload WEBLOGS $HOME/solr_configs
```

5. Use the solrctl utility to verify that your instance directory uploaded successfully and is available to ZooKeeper. List the uploaded instance directories as follows:

```
solrctl instancedir --list
```

If you used the --create command to create a collection named weblogs, the --list command returns weblogs.

6. After uploading the updated collection configuration. you need to reload every collection referencing that instancce directory for the updates to take effect.

```
solrctl collection --reload [***COLLECTION_NAME***]
```

Converting instance directories to configs

Cloudera Search supports converting existing deployments that use instance directories to use configs. This allows you to implement access control using ZooKeeper Access Control Lists and Ranger.

Procedure

1. Create a temporary config based on the existing instance directory.

```
solrctl config --create [***TEMPORARY NEW CONFIG***] [***EXISTING
 INSTANCEDIR***] \
-p immutable=false
```

Replace *[***TEMPORARY NEW CONFIG***]* with a config name and *[***EXISTING INSTANCEDIR***]* with the name of the instancedir you want to convert.

For example, if the instance directory name is weblogs_config:

```
solrctl config --create weblogs_config_temp weblogs_config \
-p immutable=false
```

2. Delete the existing instance directory.

```
solrctl instancedir --delete [***EXISTING INSTANCEDIR***]
```

For example:

```
solrctl instancedir --delete weblogs_config
```

3. Create a config using the same name as the instance directory you just deleted, based on the temporary config you created earlier.

```
solrctl config --create [***NEW CONFIG***] [***TEMPORARY NEW CONFIG***] \
```

```
-p immutable=false
```

Replace *[***NEW CONFIG***]* with the name of the instancedir you just deleted and *[***TEMPORARY NEW CONFIG***]* with the name of the temporary config created earlier in this procedure.

For example:

```
solrctl config --create weblogs_config weblogs_config_temp \
-p immutable=false
```

4. Delete the temporary config:

```
solrctl config --delete [***TEMPORARY NEW CONFIG***]
```

For example:

```
solrctl config --delete weblogs_config_temp
```

5. Reload the affected collection:

```
solrctl collection --reload [***COLLECTION NAME***]
```

Replace *[***COLLECTION NAME***]* with the name of the collection you want to reload.

For example:

```
solrctl collection --reload weblogs
```

Using custom JAR files with Cloudera Search

Cloudera Search supports custom plug-in code. You can load classes into JAR files and then configure Cloudera Search to find these files.

About this task

To correctly deploy custom JARs, ensure that:

- Custom JARs are pushed to the same location on all hosts in your cluster that are hosting Cloudera Search (Solr Service).
- Supporting configuration files direct Cloudera Search to find the custom JAR files.
- Any required configuration files such as schema.xml or solrconfig.xml reference the custom JAR code.

The following procedure describes how to use custom JARs. Some cases may not require completion of every step. For example, indexer tools that support passing JARs as arguments may not require modifying xml files. However, completing all configuration steps helps ensure the custom JARs are used correctly in all cases.

Procedure

1. Copy your custom JAR file in the same location on all hosts in your cluster.

2. For all collections where custom JARs will be used, modify solrconfig.xml to include references to the new JAR files. These directives can include explicit or relative references and can use wildcards. In the solrconfig.xml file, add <lib> directives to indicate the JAR file locations or <path> directives for specific jar files.
   For example:

   ```
   <lib path="/usr/lib/solr/lib/MyCustom.jar" />
   ```

   or

   ```
   <lib dir="/usr/lib/solr/lib" />
   ```

   or

   ```
   <lib dir="../../../myProject/lib" regex=".*\.jar" />
   ```

3. For all collections in which custom JARs will be used, reference custom JAR code in the appropriate Solr configuration file. The two configuration files that most commonly reference code in custom JARs are solrconfig.xml and schema.xml.
4. For all collections in which custom JARs will be used, use solrctl to update ZooKeeper's copies of configuration files such as solrconfig.xml and schema.xml
   For example:

   ```
   solrctl instancedir --update [***NAME***] [***PATH***]
   ```

   - *[***NAME***]* specifies the instancedir associated with the collection using solrctl instancedir --create.
   - *[***PATH***]* specifies the directory containing the collection's configuration files.

   For example:

   ```
   solrctl instancedir --update collection1 $HOME/solr_configs
   ```

5. For all collections in which custom JARs will be used, use `RELOAD` to refresh information. When the `RELOAD` command is issued to any host that hosts a collection, that host sends subcommands to all replicas in the collection. All relevant hosts refresh their information, so this command must be issued once per collection.

   ```
   http://example.com:8983/solr/admin/collections?action=RELOAD&name=collec
   tion1
   ```

6. Ensure that the class path includes the location of the custom JAR file.
   a) For example, if you store the custom JAR file in /opt/myProject/lib/, add that path as a line to the ~/.profile for the Solr user.
   b) Restart the Solr service to reload the PATH variable.
   c) Repeat this process of updating the PATH variable for all hosts.

What to do next

The system is now configured to find custom JAR files. Some command-line tools included with Cloudera Search support specifying JAR files. For example, when using MapReduceIndexerTool, use the --libjars option to specify JAR files to use. Tools that support specifying custom JARs include:

- MapReduceIndexerTool
- Lily HBase Indexer
- CrunchIndexerTool

Retrieving the clusterstate.json file

You can access the clusterstate.json file for Solr monitoring purposes through the Solr webUI.

Procedure

1. Go to the Solr webUI.
2. Select Cloud Tree .
3. From the tree view select /clusterstate.json.

Managing collections in Cloudera Search

A collection in Cloudera Search refers to a repository for indexing and querying documents. Collections typically contain the same types of documents with similar schemas.

To start using Solr and indexing data, you must configure a collection to hold the index.

A collection requires the following configuration files:

- solrconfig.xml
- schema.xml
- Any additional files referenced in the xml files

The solrconfig.xml file contains all of the Solr settings for a given collection, and the schema.xml file specifies the schema that Solr uses when indexing documents. For more details on how to configure a collection, see *SchemaXml*.

A typical deployment workflow with solrctl consists of:

1. Establishing a configuration.
   - If using configs, creating a config object from a template.
   - If using instance directories, generating an instance directory and uploading it to ZooKeeper.
2. Creating a collection associated with the name of the config or instance directory.

Collections are managed using the solrctl commandline utility.

Related Concepts

Managing collection configuration using configs or instance directories

Related Information

solrctl Reference

SchemaXml

Creating a Solr collection

Learn how to create a collection so that you can start indexing data with Solr.

Before you begin

- If you have enabled Ranger for authorization, you must have Solr Admin permission to be able to create collections.
- Before you can create a Solr collection you need to generate a collection configuration using either a config or an instance directory.

About this task

Note: Although it is not currenly strictly enforced, you are strongly recommended to observe the following limitations on collection names:

- Use only ASCII alphanumeric characters (A-Za-z0-9), hyphen (-), or underscore (_).
- Avoid using the strings shard and replica.

Procedure

1. If you are using Kerberos, kinit as a user with permission to create the collection:

```
kinit solradmin@[***EXAMPLE.COM***]
```

   Replace *[***EXAMPLE.COM***]* with your Kerberos realm name.

2. On a host running a Solr server, make sure that the SOLR_ZK_ENSEMBLE environment variable is set in /etc/solr/conf/solr-env.sh. For example:

```
cat /etc/solr/conf/solr-env.sh
export SOLR_ZK_ENSEMBLE=zk01.example.com:2181,zk02.example.com:2181,zk03.
example.com:2181/solr
```

   This is automatically set on hosts with a Solr Server or Gateway role in Cloudera Manager.

3. Create a new collection using the following command:

```
solrctl collection --create [***COLLECTION NAME***] -s [***NUMBER OF
 SHARDS***] -c [***COLLECTION CONFIGURATION***]
```

   where

*[***COLLECTION NAME***]*	User-defined name of the collection. **Note:** Although it is not currenly strictly enforced, you are strongly recommended to observe the following limitations on collection names: • Use only ASCII alphanumeric characters (A-Za-z0-9), hyphen (-), or underscore (_). • Avoid using the strings shard and replica.
*[***NUMBER OF SHARDS***]*	The number of shards you want to split your collection into.
*[***COLLECTION CONFIGURATION***]*	The name of an existing collection configuration.

   For example:

```
solrctl collection --create logs -s 3 -c logs_config
```

Related Tasks

Generating collection configuration using configs

Generating Solr collection configuration using instance directories

Viewing existing collections

Learn how you can list existing Solr collections.

You can view existing collections using the solrctl collection --list command.

Deleting all documents in a collection

Deleting all documents in a Solr collection does not delete the collection or its configuration files. It only deletes the index. This can be useful for rapid prototyping of configuration changes in test environments.

Before you begin

If you have enabled Ranger for authorization, you must have Solr Admin permission to be able to delete documents in a collection.

Procedure

1. If you are using Kerberos, kinit as a user with permission to delete the collection:

```
kinit solradmin@[***EXAMPLE.COM***]
```

   Replace *[***EXAMPLE.COM***]* with your Kerberos realm name.
2. On a host running Solr Server, make sure that the SOLR_ZK_ENSEMBLE environment variable is set in /etc/solr/conf/solr-env.sh. For example:

```
$ cat /etc/solr/conf/solr-env.sh
export SOLR_ZK_ENSEMBLE=zk01.example.com:2181,zk02.example.com:2181,zk03.example.com:2181/solr
```

   If you are using Cloudera Managerr, this is automatically set on hosts with a Solr Server or Gateway role.
3. Delete the documents:

```
solrctl collection --deletedocs logs
```

Deleting a collection

Deleting a Solr collection deletes the collection and its index, but does not delete its configuration files.

Before you begin

If you have enabled Ranger for authorization, you must have Solr Admin permission to be able to delete collections.

About this task

Procedure

1. If you are using Kerberos, kinit as a user with permission to delete the collection:

```
kinit solradmin@[***EXAMPLE.COM***]
```

   Replace *[***EXAMPLE.COM***]* with your Kerberos realm name.
2. On a host running Solr Server, make sure that the SOLR_ZK_ENSEMBLE environment variable is set in /etc/solr/conf/solr-env.sh. For example:

```
$ cat /etc/solr/conf/solr-env.sh
export SOLR_ZK_ENSEMBLE=zk01.example.com:2181,zk02.example.com:2181,zk03.example.com:2181/solr
```

   If you are using Cloudera Manager, this is automatically set on hosts with a Solr Server or Gateway role.
3. Delete the collection:

```
solrctl collection --delete [***COLLECTION NAME***]
```

   Replace *[***COLLECTION NAME***]* with the name of the collection you want to delete.

Updating the schema in a collection

It is sometimes necessary to change the underlaying schema behind a Solr collection. Find out how to do it for collections configured through instance directories and configs.

If your collection was configured using an instance directory, you can download the instance directory, edit sche ma.xml, then re-upload it to ZooKeeper. For instructions, see Generating Solr collection configuration using instance directories on page 9.

If your collection was configured using a config, you can update the schema using the Schema API. For information on using the Schema API, see Schema API in the Apache Solr Reference Guide.

Creating a replica of an existing shard

You can create additional replicas of existing shards using the solrctl utility. Replicating shards boosts query throughput and prevents data loss.

Procedure

To create additional replicas of existing shards, use the following command:

```
solrctl core --create [***NEW CORE***] -p collection=[***COLLECTION NAME***]
 \
-p shard=[***SHARD TO REPLICATE***]
```

For example, to create a new replica of the collection named collection1 that is comprised of shard1, use the following command:

```
solrctl core --create collection1_shard1_replica2 \
-p collection=collection1 -p shard=shard1
```

Migrating Solr replicas

When you replace a host, migrating replicas from that host to the new host, instead of depending on failure recovery, can help ensure optimal performance.

About this task

Where possible, the Solr service routes requests to the proper host. Both `ADDREPLICA` and `DELETEREPLICA` Collections API calls can be sent to any host in the cluster.

- For adding replicas, the node parameter ensures the new replica is created on the intended host. If no host is specified, Solr selects a host with relatively fewer replicas.
- For deleting replicas, the request is routed to the host that hosts the replica to be deleted.

Adding replicas can be resource intensive. For best results, add replicas when the system is not under heavy load. For example, do not add replicas when heavy indexing is occurring or when MapReduceIndexerTool jobs are running.

Cloudera recommends using API calls to create and unload cores. Do not use the Cloudera Manager Admin Console or the Solr Admin UI for these tasks.

This procedure uses the following names:

- Host names:
  - Origin: solr01.example.com.
  - Destination: solr02.example.com.
- Collection name: email

- Replicas:
  - The original replica email_shard1_replica1, which is on solr01.example.com.
  - The new replica email_shard1_replica2, which will be on solr02.example.com.

Procedure

1. If you want to add a replica to a particular node, review the contents of the live_nodes directory on ZooKeeper to find all nodes available to host replicas. Open the Solr Administration User interface, click Cloud Tree live_nodes .

   The Solr Administration User Interface, including live_nodes, appears.

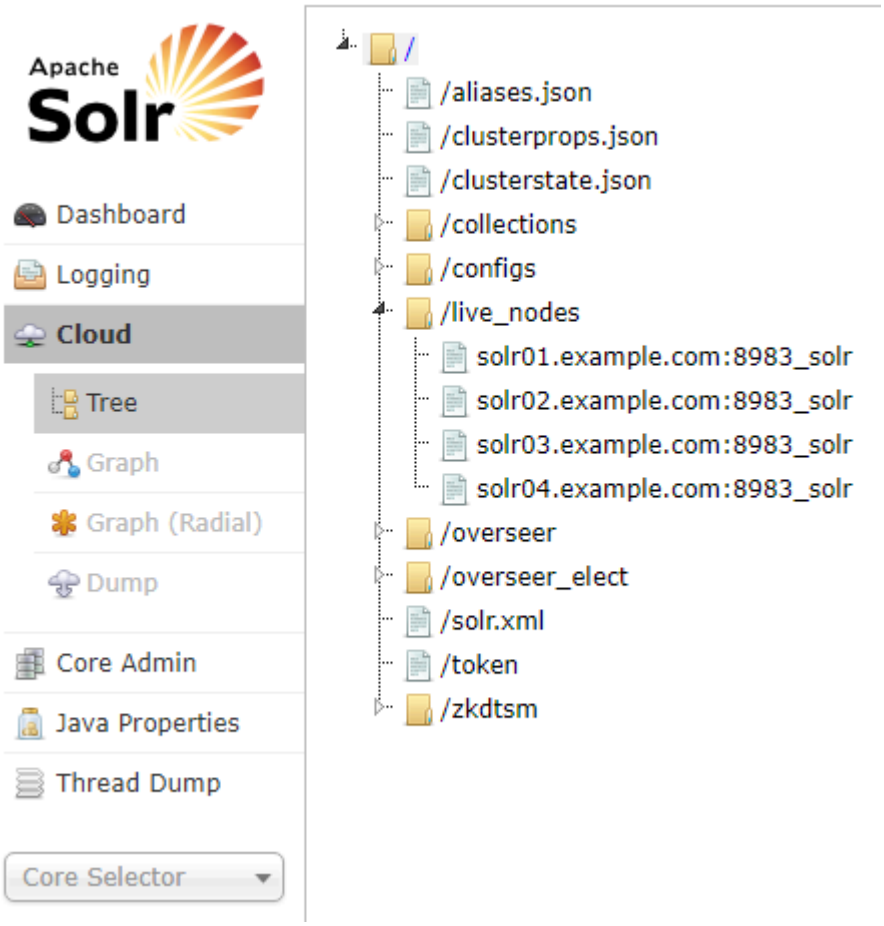

   Note: Information about Solr nodes can also be found in clusterstate.json, but that file only lists nodes currently hosting replicas. Nodes running Solr but not currently hosting replicas are not listed in the clus terstate.json file.

2. Add the new replica on solr02.example.com using the `ADDREPLICA` API call.

```
http://solr01.example.com:8983/solr/admin/collections?action=ADDREPLICA&
collection=email&shard=shard1&node=solr02.example.com:8983_solr
```

3. Verify that the replica creation succeeds and moves from recovery state to ACTIVE.

   You can check the replica status in the Cloud view, which can be found at a URL similar to: http://solr02.example.com:8983/solr/#/~cloud.

   Note: Do not delete the original replica until the new one is in the ACTIVE state. When the newly added replica is listed as ACTIVE, the index has been fully replicated to the newly added replica. The total time to replicate an index varies according to factors such as network bandwidth and the size of the index. Replication times on the scale of hours are not uncommon and do not necessarily indicate a problem.

   You can use the details command to get an XML document that contains information about replication progress. Use curl or a browser to access a URI similar to:

   ```
   http://solr02.example.com:8983/solr/email_shard1_replica2/replication?command=details
   ```

   Accessing this URI returns an XML document that contains content about replication progress. A snippet of the XML content might appear as follows:

   ```
   ...
   <str name="numFilesDownloaded">126</str>
   <str name="replication StartTime">Tue Jan 21 14:34:43 PST 2014</str>
   <str name="timeElapsed">457s</str>
   <str name="currentFile">4xt_Lucene41_0.pos</str>
   <str name="currentFileSize">975.17 MB</str>
   <str name="currentFileSizeDownloaded">545 MB</str>
   <str name="currentFileSizePercent">55.0</str>
   <str name="bytesDownloaded">8.16 GB</str>
   <str name="totalPercent">73.0</str>
   <str name="timeRemaining">166s</str>
   <str name="downloadSpeed">18.29 MB</str>
   ...
   ```

4. Use the `CLUSTERSTATUS` API call to retrieve information about the cluster, including current cluster status:

   ```
   http://solr01.example.com:8983/solr/admin/collections?action=clusterstatus&wt=json&indent=true
   ```

   Review the returned information to find the correct replica to remove. An example of the JSON file might appear as follows:

5. Delete the old replica on solr01.example.com server using the DELETEREPLICA API call:

```
http://solr01.example.com:8983/solr/admin/collections?action=DELETEREPLI
CA&collection=email&shard=shard1&replica=core_node2
```

The `DELETEREPLICA` call removes the datadir.

Related Information

Collections API in Apache Solr Reference Guide

Splitting a shard on HDFS

When performing a shard split on an index that is stored on HDFS, the SplitShardCmd evaluates free disk space on the local file system of the server where Solr is installed. This may cause the command to fail, perceiving that there is no adequate disk space to perform the shard split.

Procedure

- Run the following command with the skipFreeSpaceCheck=true' argument to skip the check for sufficient disk space:
  - On nonsecure clusters:

    ```
    curl 'http://$[***SOLR_SERVER_HOSTNAME***]:[***SOLR_SERVER_PORT***]/
    solr/admin/collections?action=SPLITSHARD&collec
    tion=[***COLLECTION_NAME***]&shard=[***SHARD_TO_SPLIT***]&skipFreeSpa
    ceCheck=true
    ```

  - On secure clusters:

    ```
    curl -k -u : --negotiate 'http://
    $[***SOLR_SERVER_HOSTNAME***]:[***SOLR_SERVER_PORT***]/solr/admin/
    collections?action=SPLITSHARD&collection=[***COLLECTION_NAME***]&sha
    rd=[***SHARD_TO_SPLIT***]&skipFreeSpaceCheck=true
    ```

  Replace *[***SOLR_SERVER_HOSTNAME***]* with a valid Solr server hostname, *[***SOLR_SERVER_PORT***]* with the Solr server port number, *[***COLLECTION_NAME***]* with the collection name, and *[***SHARD_TO_SPLIT***]* with the ID of the shard to split.

Results

To verify that the command executed successfully, check overseer logs for a similar entry:

```
2024-02-02 12:43:23.743 INFO  (OverseerThreadFactory-9-thread-5-processing-n
:myhost.example.com:8983_solr) [c:example s:shard1  ] o.a.s.c.a.c.SplitShard
Cmd Skipping check for sufficient disk space
```

Backing up a collection from HDFS

You can back up Solr collections to your local cluster or a remote cluster using the solrctl utility to minimize data loss caused by accidental or malicious administrative actions. Learn how to create, prepare, and export the Solr collection snapshot to create a backup of the Solr collection.

About this task

Note:

Creating backups and performing restore operations using the solrctl CLI tool exclusively works on HDFS. If you want to create a backup to, or perform a restore from a different FS, consult the Apache Solr documentation.

Before you begin

If you are using a secure (Kerberos-enabled) cluster, specify your jaas.conf file by adding the following parameter to each command:

```
--jaas [***/PATH/TO/JAAS.CONF***]
```

If TLS is enabled for the Solr service, specify the truststore and password using the ZKCLI_JVM_FLAGS environment variable before you begin the procedure:

```
export ZKCLI_JVM_FLAGS="-Djavax.net.ssl.trustStore=[***/PATH/TO/TRUSTSTORE \
-Djavax.net.ssl.trustStorePassword=[***TRUST_STORE_PASSWORD***]"
```

Procedure

1. Create a snapshot. On a host running Solr Server, run the following command:

   ```
   solrctl collection --create-snap
   shot [***USER_DEFINED_NAME_OF_THE_SNAPSHOT***] -
   c [***NAME_OF_THE_COLLECTION_TO_BE_BACKED_UP***]
   ```

   For example, for a collection named tweets the command:

   ```
   solrctl collection --create-snapshot tweets-$(date +%Y%m%d%H%M) -c tweets
   ```

   creates the snapshot tweets-202103281043.

2. If you are backing up the Solr collection to a remote cluster, prepare the snapshot for export. If you are backing up the Solr collection to the local cluster, skip this step.

   The destination HDFS directory path (*[***DESTINATION_DIRECTORY***]*, specified by the -d option) must exist on the local cluster before you run this command. Make sure that the Solr superuser (solr by default) has permission to write to this directory.

   ```
   solrctl collection --prepare-snapshot-ex
   port [***NAME_OF_THE_SNAPSHOT_TO_BE_EXPORTED***] -
   c [***COLLECTION_NAME***] -d [***DESTINATION_DIRECTORY***]
   ```

   For example:

   ```
   hdfs dfs -mkdir -p /path/to/backup-staging/tweets-202103281043
   hdfs dfs -chown :solr /path/to/backup-staging/tweets-202103281043
   solrctl collection --prepare-snapshot-export tweets-202103281043 -c twe
   ets \
   -d /path/to/backup-staging/tweets-202103281043
   ```

3. Export the snapshot. This step uses the DistCp utility to back up the collection metadata as well as the corresponding index files. The destination directory must exist and be writable by the Solr superuser (solr by default).

   To export the snapshot to a remote cluster, run the following command:

```
solrctl collection --export-snap
shot [***NAME_OF_THE_SNAPSHOT_TO_BE_EXPORTED***] -
s [***SOURCE_DIRECTORY***] -
d [***PROTOCOL***]://[***NAMENODE***]:[***PORT***]/[***DESTINATION_DIRECTORY***]
```

   For example:

   For HDFS

   HDFS protocol:

```
solrctl collection --export-snapshot tweets-202103281043 -s /path/to/bac
kup-staging/tweets-202103281043 \
-d hdfs://nn01.example.com:8020/path/to/backups
```

   For webHDFS

   WebHDFS protocol:

```
solrctl collection --export-snapshot tweets-202103281043 -s /path/to/bac
kup-staging/tweets-202103281043 \
-d webhdfs://nn01.example.com:20101/path/to/backups
```

   To export the snapshot to the local cluster, run the following command:

```
solrctl collection --export-snap
shot [***NAME_OF_THE_SNAPSHOT_TO_BE_EXPORTED***] -
c [***COLLECTION_NAME***] -d [***DESTINATION_DIRECTORY***]
```

   For example:

```
solrctl collection --export-snapshot tweets-202103281043 -c tweets -d /p
ath/to/backups/
```

4. Delete the snapshot after exporting:

```
solrctl collection --delete-snap
shot [***NAME_OF_THE_SNAPSHOT_TO_BE_DELETED***] -c [***COLLECTION_NAME***]
```

   For example:

```
solrctl collection --delete-snapshot tweets-202103281043 -c tweets
```

Related Information

solrctl Reference

Backing up a collection from local file system

Back up Solr collections to a shared file system to minimize data loss caused by accidental or malicious administrative actions. Learn how to create backup of a collection from the local file system (FS).

About this task

Note:

Creating backups and performing restore operations using the solrctl CLI tool exclusively works on HDFS. If you want to create a backup to, or perform a restore from a different FS, consult the Apache Solr documentation.

If you use local FS to store backups, each Solr host stores its backup directory locally, that is, server X contains the backup directory snapshots.shard1, server Y contains snapshots.shard2 and you need to copy those to a shared location in order to be able to restore them later. Because of this, Cloudera recommends to target backups to a shared file system, even if your Solr collection uses local FS.

Before you begin

If you are using a secure (Kerberos-enabled) cluster, specify your jaas.conf file by adding the following parameter to each command:

```
--jaas [***/PATH/TO/JAAS.CONF***]
```

Note:

Procedure

1. Create a snapshot. On a host running Solr Server, run the following command:

   ```
   solrctl collection --create-snap
   shot [***USER_DEFINED_NAME_OF_THE_SNAPSHOT***] -
   c [***NAME_OF_THE_COLLECTION_TO_BE_BACKED_UP***]
   ```

   This step is optional. You can back up this snapshot by specifying the *[***USER_DEFINED_NAME_OF_THE_SNAPSHOT***]* as the value of commitName parameter. If you do not create and specify a snapshot, the backup exports the index state corresponding to the current latest finished commit.

   For example, for a collection named tweets the command:

   ```
   solrctl collection --create-snapshot tweets-$(date +%Y%m%d%H%M) -c tweets
   ```

   creates the snapshot tweets-202103281043.

2. Create the backup. The destination directory must exist and be writable by the Solr superuser (solr by default).

   - To back up a snapshot, use the following command:

     ```
     curl -k --negotiate -u : 'http://[***HOST***]:[***PORT***]/solr/
     admin/collections?action=BACKUP&name=[***BACKUP_NAME***]&commitN
     ```

```
ame=[***SNAPSHOT_NAME***]&collection=[***COLLECTION_NAME***]&locatio
n=***BACKUP_LOCATION***'
```

For example:

```
curl -k --negotiate -u : 'http://host1.example.com:8983/solr/admin/colle
ctions?action=BACKUP&name=mybackup&commitName=tweets-202103281043&collec
tion=tweets&location=/tmp'
```

The example URL targets one (any one) of the Solr servers and creates a backup of the entire collection.

- To back up the current state of the index:

```
curl -k --negotiate -u : 'http://[***HOST***]:[***PORT***]/solr/admin/
collections?action=BACKUP&name=[***BACKUP_NAME***]&collection=tweets&l
ocation=/tmp'
```

For example:

```
curl -k --negotiate -u : 'http://host1.example.com:8983/solr/admin/colle
ctions?action=BACKUP&name=mybackup&collection=tweets&location=/tmp'
```

The example URL targets one (any one) of the Solr servers and creates a backup of the entire collection.

[HOST***]***

is a host name or IP address valid in your environment

[PORT***]***

is the port where you can access your Solr service

[BACKUP_LOCATION***]***

specifies the directory (for example, /tmp) of the backup target defined in solr.xml where the backup is to be stored. If you have defined a HDFS target backup repository, the backup is stored on HDFS at *[***BACKUP_LOCATION***]*

[BACKUP_NAME***]***

specifies the name of the backup - the backup is created in the subdirectory *[***BACKUP_NAME***]* of the backup repository directory *[***BACKUP_LOCATION***]*.

[SNAPSHOT_NAME***]***

is the name of the snapshot you want to back up

[COLLECTION_NAME***]***

specifies the collection that you want to back up

Tip: To use a specific repository as a backup target, use the repository parameter.

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<lst name="responseHeader"><int name="status">0</int><int name="QTime">363
6</int></lst>
</response>
```

After completing a backup, the data is stored in the standard backup format:

3. To check the backup files, run the following command:

```
hdfs dfs -ls /tmp/mybackup
```

```
Found 4 items
```

```
-rw-rw-rw-   2 solr supergroup        181 2021-01-13 21:33 /tmp/mybackup/b
ackup.properties
drwxrwxrwx   - solr supergroup          0 2021-01-13 21:33 /tmp/mybackup/
snapshot.shard1
drwxrwxrwx   - solr supergroup          0 2021-01-13 21:33 /tmp/mybackup/s
napshot.shard2
drwxrwxrwx   - solr supergroup          0 2021-01-13 21:33 /tmp/mybackup/
zk_backup
```

4. Delete the snapshot after exporting:

```
solrctl collection --delete-snap
shot [***NAME_OF_THE_SNAPSHOT_TO_BE_DELETED***] -c [***COLLECTION_NAME***]
```

For example:

```
solrctl collection --delete-snapshot tweets-202103281043 -c tweets
```

Related Tasks

Defining a backup target in solr.xml

Related Information

Backup/Restore Storage Repositories

BACKUP: Backup Collection

Restoring a collection

You can restore a Solr collection from a backup stored on either a remote cluster or the local cluster using the solrctl utility. You must pass a unique request identifier as part of the restore command in the solrctl utility while initiating the restore operation for tracking the process.

About this task

Note:

Creating backups and performing restore operations using the solrctl CLI tool exclusively works on HDFS. If you want to create a backup to, or perform a restore from a different FS, consult the Apache Solr documentation.

Before you begin

If you are using a secure (Kerberos-enabled) cluster, specify your jaas.conf file by adding the following parameter to each command:

```
-jaas [***/PATH/TO/JAAS.CONF***]
```

If TLS is enabled for the Solr service, specify the truststore and password by using the ZKCLI_JVM_FLAGS environment variable before you begin the procedure:

```
export ZKCLI_JVM_FLAGS="-Djavax.net.ssl.trustStore=[***/PATH/TO/TRUSTSTORE \
-Djavax.net.ssl.trustStorePassword=[***TRUST_STORE_PASSWORD***]"
```

Procedure

1. If you are restoring from a backup stored on a remote cluster, copy the backup from the remote cluster to the local cluster. If you are restoring from a local backup, skip this step.

   Run the following commands on the cluster to which you want to restore the collection:

   ```
   hdfs dfs -mkdir -p [***PATH/TO/RESTORE/STAGING***]hadoop distc
   p [***PROTOCOL***]://[***NAMENODE***]:[***PORT***]/[***PATH/TO/
   BACKUP***] [***/PATH/TO/RESTORE-STAGING***]
   ```

   For example:

   For HDFS

   HDFS protocol:

   ```
   hadoop distcp hdfs://nn01.example.com:8020/path/to/backups/tweet
   s-202103281043 /path/to/restore-staging
   ```

   For webHDFS

   WebHDFS protocol:

   ```
   hadoop distcp webhdfs://nn01.example.com:20101/path/to/backups/t
   weets-202103281043 /path/to/restore-staging
   ```

2. Start the restore procedure. Run the following command:

   ```
   solrctl collection --restore [***NAME_OF_THE_RESTORED_COLLECTION***] -
   l [***BACKUP_LOCATION***] -b [***NAME_OF_THE_SNAPSHOT_TO_BE_RESTORED***] -
   i [***REQUEST_ID***]
   ```

   Make sure that you use a unique *[***REQUEST_ID***]* each time you run this command.

   Note:

   Statuses of historic job runs are stored in ZooKeeper and can be retrieved using the solrctl collection --req uest-status *[***REQUEST_ID***]* command. The number of async call responses stored in a cluster is limited to 10,000.

   Status information can be removed from ZooKeeper using the DELETESTATUS API call.

   For example:

   ```
   solrctl collection --restore tweets -l /path/to/restore-staging -b tweet
   s-202103281043 -i restore-tweets
   ```

3. Monitor the status of the restore operation. Run the following command periodically:

   ```
   solrctl collection --request-status [***REQUEST_ID***]
   ```

   Look for <str name="state"> in the output. For example (emphasis added):

   ```
   solrctl collection --request-status restore-tweets
    <?xml version="1.0" encoding="UTF-8"?> <response> <lst name="responseHea
   der"> <int name="status"> 0</int> <int name="QTime"> 1</int> </lst> \
   ```

```
<lst name="status"> <str name="state"> completed</str> <str name="msg"> fo
und restore-tweets in completed tasks</str> </lst> </response>
```

The state parameter can be one of the following:

- running: The restore operation is running.
- completed: The restore operation is complete.
- failed: The restore operation failed.
- notfound: The specified *[***REQUEST_ID***]* does not exist.

Related Information

solrctl Reference

Defining a backup target in solr.xml

If you want to to define or modify a backup target, you can do it by downloading, editing, and reuploading the solr.xml file from ZooKeeper.

About this task

The solr.xml file of your Solr installation, which is stored in ZooKeeper, can define a backup target repository, and depending on your installation it likely has a default target pointing to HDFS.

HDFS as a backup target is still fine even if your Solr collection uses a local file system (FS) / NRTCachingDirectoryFactory, so even with local FS collections you can store your backups on HDFS.

Similarly to this, other repositories like a LocalFileSystemRepository can also be defined in the solr.xml if you want to store the backups on a location other than HDFS.

Important: If you use a local FS to store backups, each Solr host stores its backup directory locally. That is, server X contains the backup directory snapshots.shard1, server Y contains snapshots.shard2 and you need to copy them to a shared location in order to be able to restore them later. Because of this, Cloudera recommends you to target backups to a shared file system.

If the solr.xml does not have a backup repository at all, it defaults to the local FS repository.

Note: If you use a HDFS backup repository, the backup also works if the Solr servers are located on nodes which do not have HDFS data node roles, they just need to have a HDFS client (gateway) role.

Procedure

1. To define or modify a backup target, download the solr.xml file from ZooKeeper using the following solrctl command:

```
solrctl cluster --get-solrxml solr.xml
```

2. Edit the contents of the solr.xml file.
   This is an example of defining a HDFS backup target in the solr.xml file:

```
<backup>
  <repository name="hdfs" class="org.apache.solr.core.backup.repository
.HdfsBackupRepository" default="false">
    <str name="location">${solr.hdfs.default.backup.path}</str>
    <str name="solr.hdfs.home">${solr.hdfs.home:}</str>
    <str name="solr.hdfs.confdir">${solr.hdfs.confdir:}</str>
  </repository>
</backup>
```

3. Reupload the modified solr.xml file:

```
solrctl cluster --put-solrxml solr.xml
```

Cloudera Search log files

Cloudera Search has several log files, stored on each Solr Server host. See the following for log file locations and a brief description of each. The referenced configuration parameters can be viewed or modified as described in Cloudera Manager.

Logs under /var/log/solr/

/var/log/solr/ is the parent directory for most Search logs.

Table 3: Logs and directories under /var/log/solr/

Log or Directory	Description	Configuration Options
audit/	Audit log directory.	The path may be specified by the Audit Log Directory configuration parameter.
stacks/	The directory in which stacks logs are placed.	Logs are collected if the Stacks Collection Enabled option is checked. Stacks Collection Directory - The directory in which stacks logs are placed. If not set, stacks are logged into a stacks subdirectory of the role's log directory.
solr-cmf-*SOLR-1*-SOLR_SERVER-*HOSTNAME.EXAMPLE.COM*.log.out	Solr server log file. The *SOLR-1* and *HOSTNAME.EXAMPLE.COM* portion varies depending on your environment.	Solr Server Logging Threshold parameter sets log level for Solr Server roles. Gateway Logging Threshold parameter sets log level for Gateway roles.
solr_gc_log.*	Java garbage collection (GC) logs for the Solr Server process.	

Logs under /var/run/cloudera-scm-agent/process/<*PROCESS_DIR_ID*>-*SOLR*-SOLR_SERVER/

/var/run/cloudera-scm-agent/process/<*PROCESS_DIR_ID*>-*SOLR*-SOLR_SERVER/ is the configuration directory for the currently running Solr Server role. The <*PROCESS_DIR_ID*> portion changes each time the Solr Server is restarted.

Warning: Do not modify any of the files in this directory. These files are automatically generated by Cloudera Manager. To modify any configuration parameters, use Cloudera Manager.

Table 4: Logs under /var/run/cloudera-scm-agent/process/<*PROCESS_DIR_ID*>-*SOLR*-SOLR_SERVER/

Log Directory	Description
logs/	Log directory containing the stderr and stdout logs for the Solr Server process.

To identify the current or most recent directory, list the /var/run/cloudera-scm-agent/process/*solr* directories sorted by time in reverse as follows:

```
sudo ls -ltrd /var/run/cloudera-scm-agent/process/*solr*
```

The entry at the bottom is the current or most recent process directory.

Cloudera Search configuration files

Cloudera Search configuration is primarily controlled by several configuration files, that are mostly stored in Apache ZooKeeper.

Configuration file	Description
solr.xml	This file is stored in ZooKeeper, and controls global properties for Apache Solr. To edit this file, you must download it from ZooKeeper, make your changes, and then upload the modified file back to ZooKeeper using the solrctl cluster command.
solrconfig.xml	Each collection in Solr uses a solrconfig.xml file, stored in ZooKeeper, to control collection behavior.
managed-schema or schema.xml	Cloudera recommends using a managed schema, and making schema changes using the Schema API. Collections use either a managed schema or the legacy schema.xml file. These files, also stored in ZooKeeper and assigned to a collection, define the schema for the documents you are indexing. For example, they specify which fields to index, the expected data type for each field, the default field to query when the field is unspecified, and so on.
core.properties	Unlike other configuration files, this file is stored in the local filesystem rather than ZooKeeper, and is used for core discovery.
Additional files	Any additional files referenced in the xml files, for example, custom JAR files.

Related Information

Solr configuration files
Solr cores and solr.xml
Configuring solrconfig.xml
Schema API
Schema factory definition in SolrConfig
Defining core.properties

Cloudera Runtime 7.2.18

Morphlines Reference Guide

Date published: 2015-05-05
Date modified: 2024-04-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Implementing your own Custom Command 5

Morphline commands overview 5

kite-morphlines-core-stdio 10

kite-morphlines-core-stdlib 14

kite-morphlines-avro 42

kite-morphlines-json 46

kite-morphlines-hadoop-core 48

kite-morphlines-hadoop-parquet-avro 49

kite-morphlines-hadoop-rcfile 50

kite-morphlines-hadoop-sequencefile 51

kite-morphlines-maxmind 52

kite-morphlines-metrics-servlets 54

kite-morphlines-protobuf 61

kite-morphlines-tika-core 63

kite-morphlines-tika-decompress 64

kite-morphlines-saxon 65

kite-morphlines-solr-core..... 70

kite-morphlines-solr-cell.....75

kite-morphlines-useragent.....80

Implementing your own Custom Command

Before we dive into the currently available commands, it is worth noting again that perhaps the most important property of the Morphlines framework is how easy it is to add new transformations and I/O commands and integrate existing functionality and third party systems. If none of the existing commands match your use case, you can easily write your own command and plug it in.

Simply implement the Java interface Command or subclass AbstractCommand, have it handle a Record and add the resulting Java class to the classpath, along with a CommandBuilder implementation that defines the name(s) of the command and serves as a factory. Here are two example implementations: toString and readLine. No registration or other administrative action is required. Indeed, none of the standard commands are special or intrinsically known per se. All commands are implemented like this, even including standard commands such as pipe, if, and tryRules. This means your custom commands can even replace any standard commands, if desired. When writing your own custom Command implementation you can take advantage of lifecycle methods: you compile regexes, read config files and perform other expensive setup stuff once in the constructor of the command, then reuse the resulting optimized representation any number of times in the execution phase on method process(). Putting it all together, you can download and run this working example Maven project that demonstrates how to unit test Morphline config files and custom Morphline commands. With that said, the following tables provide a short description of each available command and a link to the complete documentation.

Morphline commands overview

Morphlines provides a set of frequently-used high-level transformation and I/O commands that can be combined in application specific ways. This is a short description of each available command and a link to the complete documentation.

kite-morphlines-core-stdio

readBlob

Converts a byte stream to a byte array in main memory.

readClob

Converts a byte stream to a string.

readCSV

Extracts zero or more records from the input stream of bytes representing a Comma Separated Values (CSV) file.

readLine

Emits one record per line in the input stream.

readMultiLine

Log parser that collapses multiple input lines into a single record, based on regular expression pattern matching.

kite-morphlines-core-stdlib

addCurrentTime

Adds the result of System.currentTimeMillis() to a given output field.

addLocalHost

Adds the name or IP of the local host to a given output field.

addValues

Adds a list of values (or the contents of another field) to a given field.

addValuesIfAbsent

Adds a list of values (or the contents of another field) to a given field if not already contained.

callParentPipe

Implements recursion for extracting data from container data formats.

contains

Returns whether or not a given value is contained in a given field.

convertTimestamp

Converts the timestamps in a given field from one of a set of input date formats to an output date format.

decodeBase64

Converts a Base64 encoded String to a byte[].

dropRecord

Silently consumes records without ever emitting any record. Think /dev/null.

equals

Succeeds if all field values of the given named fields are equal to the the given values and fails otherwise.

extractURIComponents

Extracts subcomponents such as host, port, path, query, etc from a URI.

extractURIComponent

Extracts a particular subcomponent from a URI.

extractURIQueryParameters

Extracts the query parameters with a given name from a URI.

findReplace

Examines each string value in a given field and replaces each substring of the string value that matches the given string literal or grok pattern with the given replacement.

generateUUID

Sets a universally unique identifier on all records that are intercepted.

grok

Uses regular expression pattern matching to extract structured fields from unstructured log or text data.

head

Ignores all input records beyond the N-th record, akin to the Unix head command.

if

Implements if-then-else conditional control flow.

java

Scripting support for Java. Dynamically compiles and executes the given Java code block.

logTrace, logDebug, logInfo, logWarn, logError

Logs a message at the given log level to SLF4J.

not

Inverts the boolean return value of a nested command.

pipe

Pipes a record through a chain of commands.

removeFields

Removes all record fields for which the field name matches a blacklist but not a whitelist.

removeValues

Removes all record field values for which the field name and value matches a blacklist but not a whitelist.

replaceValues

Replaces all record field values for which the field name and value matches a blacklist but not a whitelist.

sample

Forwards each input record with a given probability to its child command.

separateAttachments

Emits one separate output record for each attachment in the input record's list of attachments.

setValues

Assigns a given list of values (or the contents of another field) to a given field.

split

Divides a string into substrings, by recognizing a separator (a.k.a. "delimiter") which can be expressed as a single character, literal string, regular expression, or grok pattern.

splitKeyValue

Splits key-value pairs where the key and value are separated by the given separator, and adds the pair's value to the record field named after the pair's key.

startReportingMetricsToCSV

Starts periodically appending the metrics of all commands to a set of CSV files.

startReportingMetricsToJMX

Starts publishing the metrics of all commands to JMX.

startReportingMetricsToSLF4J

Starts periodically logging the metrics of all morphline commands to SLF4J.

toByteArray

Converts a String to the byte array representation of a given charset.

toString

Converts a Java object to it's string representation; optionally also removes leading and trailing whitespace.

translate

Replace a string with the replacement value defined in a given dictionary aka lookup hash table.

tryRules

Simple rule engine for handling a list of heterogeneous input data formats.

kite-morphlines-avro

readAvroContainer

Parses an Apache Avro binary container and emits a morphline record for each contained Avro datum.

readAvro

Parses containerless Avro and emits a morphline record for each contained Avro datum.

extractAvroTree

Recursively walks an Avro tree and extracts all data into a single morphline record.

extractAvroPaths

Extracts specific values from an Avro object, akin to a simple form of XPath.

toAvro

Converts a morphline record to an Avro record.

writeAvroToByteArray

Serializes Avro records into a byte array.

kite-morphlines-json

readJson

Parses JSON and emits a morphline record for each contained JSON object, using the Jackson library.

extractJsonPaths

Extracts specific values from a JSON object, akin to a simple form of XPath.

kite-morphlines-hadoop-core

downloadHdfsFile

Downloads, on startup, zero or more files or directory trees from HDFS to the local file system.

openHdfsFile

Opens an HDFS file for read and returns a corresponding Java InputStream.

kite-morphlines-hadoop-parquet-avro

readAvroParquetFile

Parses a Hadoop Parquet file and emits a morphline record for each contained Avro datum.

kite-morphlines-hadoop-rcfile

readRCFile

Parses an Apache Hadoop RCFile and emits morphline records row-wise or column-wise.

kite-morphlines-hadoop-sequencefile

readSequenceFile

Parses an Apache Hadoop SequenceFile and emits a morphline record for each contained key-value pair.

kite-morphlines-maxmind

geoIP

Returns Geolocation information for a given IP address, using an efficient in-memory Maxmind database lookup.

kite-morphlines-metrics-servlets

registerJVMMetrics

Registers metrics that are related to the Java Virtual Machine with the MorphlineContext.

startReportingMetricsToHTTP

Exposes liveness status, health check status, metrics state and thread dumps via a set of HTTP URLs served by Jetty, using the AdminServlet.

kite-morphlines-protobuf

readProtobuf

Parses an InputStream that contains protobuf data and emits a morphline record containing the protobuf object as an attachment.

extractProtobufPaths

Extracts specific values from a protobuf object, akin to a simple form of XPath.

kite-morphlines-tika-core

detectMimeType

Uses Apache Tika to autodetect the MIME type of binary data.

kite-morphlines-tika-decompress

decompress

Decompresses gzip and bzip2 format.

unpack

Unpacks tar, zip, and jar format.

kite-morphlines-saxon

convertHTML

Converts any HTML to XHTML, using the TagSoup Java library.

xquery

Parses XML and runs the given W3C XQuery over it, using the Saxon Java library.

xslt

Parses XML and runs the given W3C XSL Transform over it, using the Saxon Java library.

kite-morphlines-solr-core

solrLocator

Specifies a set of configuration parameters that identify the location and schema of a Solr server or SolrCloud.

loadSolr

Inserts, updates or deletes records into a Solr server or MapReduce Reducer.

generateSolrSequenceKey

Assigns a unique key that is the concatenation of a field and a running count of the record number within the current session.

sanitizeUnknownSolrFields

Removes record fields that are unknown to Solr schema.xml, or moves them to fields with a given prefix.

tokenizeText

Uses the embedded Solr/Lucene Analyzer library to generate tokens from a text string, without sending data to a Solr server.

kite-morphlines-solr-cell

solrCell

Uses Apache Tika to parse data, then maps the Tika output back to a record using Apache SolrCell.

kite-morphlines-useragent

userAgent

Parses a user agent string and returns structured higher level data like user agent family, operating system, version, and device type.

kite-morphlines-core-stdio

readBlob

The readBlob command (source code) converts a byte stream to a byte array in main memory. It emits one record for the entire input stream of the first attachment, interpreting the stream as a Binary Large Object (BLOB), i.e. emits a corresponding Java byte array. The BLOB is put as a Java byte array into the _attachment_body output field by default.

The command provides the following configuration options:

Property Name	Default	Description
supportedMimeTypes	null	Optionally, require the input record to match one of the MIME types in this list.
outputField	_attachment_body	Name of the output field where the BLOB will be stored.

Example usage:

```
readBlob {}
```

readClob

The readClob command (source code) converts bytes to a string. It emits one record for the entire input stream of the first attachment, interpreting the stream as a Character Large Object (CLOB). The CLOB is put as a string into the message output field by default.

The command provides the following configuration options:

Property Name	Default	Description
supportedMimeTypes	null	Optionally, require the input record to match one of the MIME types in this list.

Property Name	Default	Description
charset	null	he character encoding to use, for example, UTF-8. If none is specified the charset specified in the _attachment_charset input field is used instead.
outputField	message	Name of the output field where the CLOB will be stored.

Example usage:

```
readClob {
  charset : UTF-8
}
```

readCSV

The readCSV command (source code) extracts zero or more records from the input stream of the first attachment of the record, representing a Comma Separated Values (CSV) file.

For the format see this article.

Some CSV files contain a header line that contains embedded column names. This command does not support reading and using such embedded column names as output field names because this is considered unreliable for production systems. If the first line of the CSV file is a header line, you must set the ignoreFirstLine option to true. You must explicitly define the columns configuration parameter in order to name the output fields.

The command provides the following configuration options:

Property Name	Default	Description
supportedMimeTypes	null	Optionally, require the input record to match one of the MIME types in this list.
separator	","	The character separating any two fields. Must be a string of length one.
columns	n/a	The name of the output fields for each input column. An empty string indicates omit this column in the output. If more columns are contained in the input than specified here, those columns are automatically named columnN.
ignoreFirstLine	false	Whether to ignore the first line. This flag can be used for CSV files that contain a header line.
trim	true	Whether leading and trailing whitespace shall be removed from the output fields.
addEmptyStrings	true	Whether or not to add zero length strings to the output fields.
charset	null	The character encoding to use, for example, UTF-8. If none is specified the charset specified in the _attachment_charset input field is used instead.
quoteChar	""	Must be a string of length zero or one. If this parameter is a String containing a single character then a quoted field can span multiple lines in the input stream. To disable quoting and multiline fields set this parameter to the empty string "".

Property Name	Default	Description
commentPrefix	""	Must be a string of length zero or one, for example "#". If this parameter is a String containing a single character then lines starting with that character are ignored as comments. To disable the comment line feature set this parameter to the empty string "".
maxCharactersPerRecord	1000000	Records longer than maxCharactersPerRecord characters are handled according to the policy specified in the onMaxCharactersPerRecord parameter described below.
onMaxCharactersPerRecord	throwException	Records longer than maxCharactersPerRecord characters are handled according to the policy specified in the onMaxCharactersPerRecord parameter. Must be one of ignoreRecord or throwException. A value of ignoreRecord indicates to ignore such records and continue with the following record (warnings about such events are emitted to the log file). This value is typically used in production. A value of throwException indicates to throw an exception and fail hard in such cases. This value is typically used for testing.

If the parameter quoteChar is a String containing a single character then a quoted field can span multiple lines in the input stream, for example as shown in the following example CSV input containing a single record with three columns:

```
column0,"Look, new hot tub under redwood tree!
All bubbly!",column2
```

The above example can be parsed by specifying a double-quote character for the parameter quoteChar, using backslash syntax per the JSON specification, as follows:

```
readCSV {
  ...
  quoteChar : "\""
```

If the parameter commentPrefix is a String containing a single character then lines starting with that character are ignored as comments. Example:

```
#This is a comment line. It is ignored.
```

Example usage for CSV (Comma Separated Values):

```
readCSV {
  separator : ","
  columns : [Age,"",Extras,Type]
  ignoreFirstLine : false
  quoteChar : ""
  commentPrefix : ""
  trim : true
  charset : UTF-8
}
```

Example usage for TSV (Tab Separated Values):

```
readCSV {
  separator : "\t"
  columns : [Age,"",Extras,Type]
  ignoreFirstLine : false
```

```
  quoteChar : ""
  commentPrefix : ""
  trim : true
  charset : UTF-8
}
```

Example usage for SSV (Space Separated Values):

```
readCSV {
  separator : " "
  columns : [Age,"",Extras,Type]
  ignoreFirstLine : false
  quoteChar : ""
  commentPrefix : ""
  trim : true
  charset : UTF-8
}
```

Example usage for Apache Hive (Values separated by non-printable CTRL-A character):

```
readCSV {
  separator : "\u0001" # non-printable CTRL-A character
  columns : [Age,"",Extras,Type]
  ignoreFirstLine : false
  quoteChar : ""
  commentPrefix : ""
  trim : false
  charset : UTF-8
}
```

readLine

The readLine command (source code) emits one record per line in the input stream of the first attachment. The line is put as a string into the message output field. Empty lines are ignored.

The command provides the following configuration options:

Property Name	Default	Description
supportedMimeTypes	null	Optionally, require the input record to match one of the MIME types in this list.
ignoreFirstLine	false	Whether to ignore the first line. This flag can be used for CSV files that contain a header line.
commentPrefix	""	A character that indicates to ignore this line as a comment for example, "#". To disable the comment line feature set this parameter to the empty string "".
charset	null	The character encoding to use, for example, UTF-8. If none is specified the charset specified in the _attachment_charset input field is used instead.

Example usage:

```
readLine {
  ignoreFirstLine : true
  commentPrefix : "#"
  charset : UTF-8
}
```

readMultiLine

The readMultiLine command (source code) is a multiline log parser that collapses multiple input lines into a single record, based on regular expression pattern matching. It supports regex, what, and negate configuration parameters similar to logstash. The line is put as a string into the message output field.

For example, this can be used to parse log4j with stack traces. Also see https://gist.github.com/smougenot/3182192 and http://logstash.net/docs/1.1.13/filters/multiline.

The input stream or byte array is read from the first attachment of the input record.

The command provides the following configuration options:

Property Name	Default	Description
supportedMimeTypes	null	Optionally, require the input record to match one of the MIME types in this list.
regex	n/a	This parameter should match what you believe to be an indicator that the line is part of a multi-line record.
what	previous	This parameter must be one of "previous" or "next" and indicates the relation of the regex to the multi-line record.
negate	false	This parameter can be true or false. If true, a line not matching the regex constitutes a match of the multiline filter and the previous or next action is applied. The reverse is also true.
charset	null	The character encoding to use, for example, UTF-8. If none is specified the charset specified in the _attachment_charset input field is used instead.

Example usage:

```
# parse log4j with stack traces
readMultiLine {
  regex : "(^.+Exception: .+)|(^\\s+at .+)|(^\\s+\\.\\.\\. \\d+ more)|(^\\s*
Caused by:.+)"
  what : previous
  charset : UTF-8
}

# parse sessions; begin new record when we find a line that starts with "S
tarted session"
readMultiLine {
  regex : "Started session.*"
  what : next
  charset : UTF-8
}
```

kite-morphlines-core-stdlib

This maven module contains standard transformation commands, such as commands for flexible log file analysis, regular expression based pattern matching and extraction, operations on fields for assignment and comparison, operations on fields with list and set semantics, if-then-else conditionals, string and timestamp conversions, scripting support for dynamic java code, a small rules engine, logging, and metrics and counters.

addCurrentTime

The addCurrentTime command (source code) adds the result of System.currentTimeMillis() as a Long integer to a given output field. Typically, a convertTimestamp command is subsequently used to convert this timestamp to an application specific output format.

The command provides the following configuration options:

Property Name	Default	Description
field	timestamp	The name of the field to set.
preserveExisting	true	Whether to preserve the field value if one is already present.

Example usage:

```
addCurrentTime {}
```

addLocalHost

The addLocalHost command (source code) adds the name or IP of the local host to a given output field.

The command provides the following configuration options:

Property Name	Default	Description
field	host	The name of the field to set.
preserveExisting	true	Whether to preserve the field value if one is already present.
useIP	true	Whether to add the IP address or fully-qualified hostname.

Example usage:

```
addLocalHost {
  field : my_host
  useIP : false
}
```

addValues

The addValues command (source code) adds a list of values (or the contents of another field) to a given field. The command takes a set of outputField : values pairs and performs the following steps: For each output field, adds the given values to the field. The command can fetch the values of a record field using a field expression, which is a string of the form @{fieldname}.

Example usage:

```
addValues {
  # add values "text/log" and "text/log2" to the source_type output field
  source_type : [text/log, text/log2]
  # add integer 123 to the pid field
  pid : [123]
  # add all values contained in the first_name field to the name field
  name : "@{first_name}"
}
```

addValuesIfAbsent

The addValuesIfAbsent command (source code) adds a list of values (or the contents of another field) to a given field if not already contained. This command is the same as the addValues command, except that a given value is only added to the output field if it is not already contained in the output field.

Example usage:

```
addValuesIfAbsent {
  # add values "text/log" and "text/log2" to the source_type output field
  # unless already present
  source_type : [text/log, text/log2]

  # add integer 123 to the pid field, unless already present
  pid : [123]

  # add all values contained in the first_name field to the name field
  # unless already present
  name : "@{first_name}"
}
```

callParentPipe

The callParentPipe command (source code) implements recursion for extracting data from container data formats. The command routes records to the enclosing pipe object. Recall that a morphline is a pipe. Thus, unless a morphline contains nested pipes, the parent pipe of a given command is the morphline itself, meaning that the first command of the morphline is called with the given record. Thus, the callParentPipe command effectively implements recursion, which is useful for extracting data from container data formats in elegant and concise ways. For example, you could use this to extract data from tar.gz files. This command is typically used in combination with the commands detectMimeType, tryRules, decompress, unpack, and possibly solrCell.

Example usage:

```
callParentPipe {}
```

For a real world example, see the solrCell command.

contains

The contains command (source code) returns whether or not a given value is contained in a given field. The command succeeds if one of the field values of the given named field is equal to one of the the given values, and fails otherwise. Multiple fields can be named, in which case the results are ANDed.

Example usage:

```
# succeed if the _attachment_mimetype field contains a value "avro/binary"
# fail otherwise
contains { _attachment_mimetype : [avro/binary] }

# succeed if the tags field contains a value "version1" or "version2",
# fail otherwise
contains { tags : [version1, version2] }
```

convertTimestamp

The convertTimestamp command (source code) converts the timestamps in a given field from one of a set of input date formats (in an input timezone) to an output date format (in an output timezone), while respecting daylight savings time rules. The command provides reasonable defaults for common use cases.

The command provides the following configuration options:

Property Name	Default	Description
field	timestamp	The name of the field to convert.
inputFormats	A list of common input date formats	A list of SimpleDateFormat or "unixTimeInMillis" or "unixTimeInSeconds". "unixTimeInMillis" and "unixTimeInSeconds" indicate the difference, measured in milliseconds and seconds, respectively, between a timestamp and midnight, January 1, 1970 UTC. Multiple input date formats can be specified. If none of the input formats match the field value then the command fails.
inputTimezone	UTC	The time zone to assume for the input timestamp.
inputLocale	""	The Java Locale to assume for the input timestamp.
outputFormat	"yyyy-MM-dd'T'HH:mm:ss.SSS'Z'"	The SimpleDateFormat to which to convert. Can also be "unixTimeInMillis" or "unixTimeInSeconds". "unixTimeInMillis" and "unixTimeInSeconds" indicate the difference, measured in milliseconds and seconds, respectively, between a timestamp and midnight, January 1, 1970 UTC.
outputTimezone	UTC	The time zone to assume for the output timestamp.
outputLocale	""	The Java Locale to assume for the output timestamp.

Example usage with plain SimpleDateFormat:

```
# convert the timestamp field to "yyyy-MM-dd'T'HH:mm:ss.SSSZ"
# The input may match one of "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'"
# or "yyyy-MM-dd'T'HH:mm:ss" or "yyyy-MM-dd".
convertTimestamp {
  field : timestamp
  inputFormats : ["yyyy-MM-dd'T'HH:mm:ss.SSS'Z'", "yyyy-MM-dd'T'HH:mm:ss",
 "yyyy-MM-dd"]
  inputTimezone : America/Los_Angeles
  outputFormat : "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'"
  outputTimezone : UTC
}
```

Example usage with Solr date rounding:

A SimpleDateFormat can also contain a literal string for Solr date rounding down to, say, the current hour, minute or second. For example: '/MINUTE' to round to the current minute. This kind of rounding results in fewer distinct values and improves the performance of Solr several ways:

- it uses less memory for many functions, e.g. sorting by time, restricting by date ranges etc.
- it improves speed of range queries based on time, e.g. "restrict documents to those from the last 7 days"
- In the case of faceting by the values in the field it will improve both memory requirements and speed.

For these reasons, it's advisable to store dates in the coarsest granularity that's appropriate for your application.

Example usage with Solr date rounding:

```
# convert the timestamp field to "yyyy-MM-dd'T'HH:mm:ss.SSSZ"
# and indicate to Solr that it shall round the time down to the current mi
nute
# per http://lucene.apache.org/solr/4_4_0/solr-core/org/apache/solr/util/Da
teMathParser.html
```

```
convertTimestamp {
  ...
  outputFormat : "yyyy-MM-dd'T'HH:mm:ss.SSS'Z/MINUTE'"
  ...
}
```

decodeBase64

The decodeBase64 command (source code) converts a Base64 encoded String to a byte[] per Section 6.8. "Base64 Content-Transfer-Encoding" of RFC 2045. The command converts each value in the given field and replaces it with the decoded value.

The command provides the following configuration options:

Property Name	Default	Description
field	n/a	The name of the field to modify.

Example usage:

```
decodeBase64 {
  field : screenshot_base64
}
```

dropRecord

The dropRecord command (source code) silently consumes records without ever emitting any record. This is much like piping to /dev/null in Unix.

Example usage:

```
dropRecord {}
```

equals

The equals command (source code) succeeds if all field values of the given named fields are equal to the the given values and fails otherwise. Multiple fields can be named, in which case a logical AND is applied to the results.

Example usage:

```
# succeed if the _attachment_mimetype field contains the value "avro/binary"
# and nothing else, fail otherwise
equals { _attachment_mimetype : [avro/binary] }
# succeed if the tags field contains nothing but the values "version1"
# and "highPriority", in that order, fail otherwise
equals { tags : [version1, highPriority] }
```

extractURIComponents

The extractURIComponents command (source code) extracts the following subcomponents from the URIs contained in the given input field and adds them to output fields with the given prefix: scheme, authority, host, port, path, query, fragment, schemeSpecificPart, userInfo.

The command provides the following configuration options:

Property Name	Default	Description
inputField	n/a	The name of the input field that contains zero or more URIs.
outputFieldPrefix	""	A prefix to prepend to output field names.

Property Name	Default	Description
failOnInvalidURI	false	If an URI is syntactically invalid (i.e. throws a URISyntaxException on parsing), fail the command (true) or ignore this URI (false).

Example usage:

```
extractURIComponents {
  inputField : my_uri
  outputFieldPrefix : uri_component_
}
```

For example, given the input field myUri with the value http://userinfo@www.bar.com:8080/errors.log?foo=x&bar=y&foo=z#fragment the expected output is as follows:

Name	Value
myUri	http://userinfo@www.bar.com:8080/errors.log?foo=x&bar=y&foo=z#fragment
uri_component_authority	userinfo@www.bar.com:8080
uri_component_fragment	fragment
uri_component_host	www.bar.com
uri_component_path	/errors.log
uri_component_port	8080
uri_component_query	foo=x&bar=y&foo=z
uri_component_scheme	http
uri_component_schemeSpecificPart	//userinfo@www.bar.com:8080/errors.log?foo=x&bar=y&foo=z
uri_component_userInfo	userinfo

extractURIComponent

The extractURIComponent command (source code) extracts a subcomponent from the URIs contained in the given input field and adds it to the given output field. This is the same as the extractURIComponents command, except that only one component is extracted.

The command provides the following configuration options:

Property Name	Default	Description
inputField	n/a	The name of the input field that contains zero or more URIs.
outputField	n/a	The field to add output values to.
failOnInvalidURI	false	If an URI is syntactically invalid (i.e. throws a URISyntaxException on parsing), fail the command (true) or ignore this URI (false).
component	n/a	The type of information to extract. Can be one of scheme, authority, host, port, path, query, fragment, schemeSpecificPart, userInfo

Example usage:

```
extractURIComponent {
  inputField : my_uri
  outputField : my_scheme
  component : scheme
```

```
}
```

extractURIQueryParameters

The extractURIQueryParameters command (source code) extracts the query parameters with a given name from the URIs contained in the given input field and appends them to the given output field.

The command provides the following configuration options:

Property Name	Default	Description
parameter	n/a	The name of the query parameter to find.
inputField	n/a	The name of the input field that contains zero or more URI values.
outputField	n/a	The field to add output values to.
failOnInvalidURI	false	If an URI is syntactically invalid (i.e. throws a URISyntaxException on parsing), fail the command (true) or ignore this URI (false).
maxParameters	1000000000	The maximum number of values to append to the output field per input field.
charset	UTF-8	The character encoding to use, for example, UTF-8.

Example usage:

```
extractURIQueryParameters {
  parameter : foo
  inputField : myUri
  outputField : my_query_params
}
```

For example, given the input field myUri with the value http://userinfo@www.bar.com/errors.log?foo=x&bar=y&foo=z#fragment the expected output record is:

```
my_query_params:x
my_query_params:z
```

findReplace

The findReplace command (source code) examines each string value in a given field and replaces each substring of the string value that matches the given string literal or grok pattern with the given replacement.

This command also supports grok dictionaries and regexes in the same way as the grok command.

The command provides the following configuration options:

Property Name	Default	Description
field	n/a	The name of the field to modify.
pattern	n/a	The search string to match.
isRegex	false	Whether or not to interpret the pattern as a grok pattern (true) or string literal (false).
dictionaryFiles	[]	A list of zero or more local files or directory trees from which to load dictionaries. Only applicable if isRegex is true. See grok command.

Property Name	Default	Description
dictionaryString	null	An optional inline string from which to load a dictionary. Only applicable if isRegex is true. See grok command.
replacement	n/a	The replacement pattern (isRegex is true) or string literal (isRegex is false).
replaceFirst	false	For each field value, whether or not to skip any matches beyond the first match.

Example usage with grok pattern:

```
findReplace {
  field : message
  dictionaryFiles : [kite-morphlines-core/src/test/resources/grok-dictionar
ies]
  pattern : """%{WORD:myGroup}"""
  #pattern : """(\b\w+\b)"""
  isRegex : true
  replacement : "${myGroup}!"
  #replacement : "$1!"
  #replacement : ""
  replaceFirst : false
}
```

Input: "hello world"

Expected output: "hello! world!"

generateUUID

The generateUUID command (source code) sets a universally unique identifier on all records that are intercepted. An example UUID is b5755073-77a9-43c1-8fad-b7a586fc1b97, which represents a 128bit value.

The command provides the following configuration options:

Property Name	Default	Description
field	id	The name of the field to set.
preserveExisting	true	Whether to preserve the field value if one is already present.
prefix	""	The prefix string constant to prepend to each generated UUID.
type	secure	This parameter must be one of "secure" or "nonSecure" and indicates the algorithm used for UUID generation. Unfortunately, the cryptographically "secure" algorithm can be comparatively slow - if it uses /dev/random on Linux, it can block waiting for sufficient entropy to build up. In contrast, the "nonSecure" algorithm never blocks and is much faster. The "nonSecure" algorithm uses a secure random seed but is otherwise deterministic, though it is one of the strongest uniform pseudo random number generators known so far.

Example usage:

```
generateUUID {
  field : my_id
}
```

grok

The grok command (source code) uses regular expression pattern matching to extract structured fields from unstructured log data.

This is well suited for syslog logs, apache, and other webserver logs, mysql logs, and in general, any log format that is generally written for humans and not computer consumption.

A grok command can load zero or more dictionaries. A dictionary is a file, file on the classpath, or string that contains zero or more REGEXNAME to REGEX mappings, one per line, separated by space. Here is an example dictionary:

```
INT (?:[+-]?(?:[0-9]+))
HOSTNAME \b(?:[0-9A-Za-z][0-9A-Za-z-]{0,62})(?:\.(?:[0-9A-Za-z][0-9A-Za-z-]{
0,62}))*(\.?|\b)
```

In this example, the regex named "INT" is associated with the following regex pattern:

```
[+-]?(?:[0-9]+)
```

and matches strings like "123", whereas the regex named "HOSTNAME" is associated with the following regex pattern:

```
\b(?:[0-9A-Za-z][0-9A-Za-z-]{0,62})(?:\.(?:[0-9A-Za-z][0-9A-Za-z-]{0,62}))*(
\.?|\b)
```

and matches strings like "www.cloudera.com".

Morphlines ships with several standard grok dictionaries. Dictionaries may be loaded from a file or directory of files on the local filesystem (see "dictionaryFiles"), files found on the classpath (see "dictionaryResources"), or literal inline strings in the morphlines configuration file (see "dictionaryString").

A grok command can contain zero or more grok expressions. Each grok expression refers to a record input field name and can contain zero or more grok patterns. The following is an example grok expression that refers to the input field named "quot;message" and contains two grok patterns:

```
expressions : {
  message : """\s+%{INT:pid} %{HOSTNAME:my_name_servers}"""
}
```

The syntax for a grok pattern is

```
%{REGEX_NAME:GROUP_NAME}
```

for example

```
%{INT:pid}
```

or

```
%{HOSTNAME:my_name_servers}
```

The REGEXNAME is the name of a regex within a loaded dictionary.

The GROUPNAME is the name of an output field.

If all expressions of the grok command match the input record, then the command succeeds and the content of the named capturing group is added to this output field of the output record. Otherwise, the record remains unchanged and the grok command fails, causing backtracking of the command chain.

Note: The morphline configuration file is implemented using the HOCON format (Human Optimized Config Object Notation). HOCON is basically JSON slightly adjusted for the configuration file use case. HOCON syntax is defined

at HOCON github page and as such, multi-line strings are similar to Python or Scala, using triple quotes. If the three-character sequence """ appears, then all Unicode characters until a closing """ sequence are used unmodified to create a string value.

In addition, the grok command supports the following parameters:

Property Name	Default	Description
dictionaryFiles	[]	A list of zero or more local files or directory trees from which to load dictionaries.
dictionaryResources	[]	A list of zero or more classpath resources (i.e. dictionary files on the classpath) from which to load dictionaries. Unlike "dictionaryFiles" it is not possible to specify directories.
dictionaryString	null	An optional inline string from which to load a dictionary.
extract	true	Can be "false", "true", or "inplace". Add the content of named capturing groups to the input record ("inplace"), to a copy of the input record ("true"), or to no record ("false").
numRequiredMatches	atLeastOnce	Indicates the minimum and maximum number of field values that must match a given grok expression for each input field name. Can be "atLeastOnce" (default), "once", or "all".
findSubstrings	false	Indicates whether the grok expression must match the entire input field value or merely a substring within.
addEmptyStrings	false	Indicates whether zero length strings stemming from empty (but matching) capturing groups shall be added to the output record.

Example usage:

```
# Index syslog formatted files
#
# Example input line:
#
# <164>Feb  4 10:46:14 syslog sshd[607]: listening on 0.0.0.0 port 22.
#
# Expected output record fields:
#
# syslog_pri:164
# syslog_timestamp:Feb  4 10:46:14
# syslog_hostname:syslog
# syslog_program:sshd
# syslog_pid:607
# syslog_message:listening on 0.0.0.0 port 22.
#
grok {
  dictionaryFiles : [kite-morphlines-core/src/test/resources/grok-dictionar
ies]
  expressions : {
    message : """<%{POSINT:syslog_pri}>%{SYSLOGTIMESTAMP:syslog_timestamp}
 %{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:\[%{POSINT:syslog_p
id}\])?: %{GREEDYDATA:syslog_message}"""

    #message2 : "(?<queue_field>.*)"
    #message4 : "%{NUMBER:queue_field}"
  }
}
```

More example usage:

```
# Split a line on one or more whitespace into substrings,
# and add the substrings to the "columns" output field.
#
# Example input line with tabs:
#
# "hello\t\tworld\tfoo"
#
# Expected output record fields:
#
# columns:hello
# columns:world
# columns:foo
#
grok {
  expressions : {
    message : """(?<columns>.+?)(\s+|\z)"""
  }
  findSubstrings : true
}
```

Even more example usage:

```
# Index a custom variant of syslog files where subfacility is optional.
#
# Dictionaries in this example are loaded from three places:
# * The my-jar-dictionaries/my-commands file found on the classpath.
# * The local file kite-morphlines-core/src/test/resources/grok-dictionaries
.
# * The inline definition shown in dictionaryString.
#
# Example input line:
#
# <179>Jun 10 04:42:51 www.foo.com Jun 10 2013 04:42:51 : %myproduct-3-mysu
bfacility-123456: Health probe failed
#
# Expected output record fields:
#
# my_message_code:%myproduct-3-mysubfacility-123456
# my_product:myproduct
# my_level:3
# my_subfacility:mysubfacility
# my_message_id:123456
# syslog_message:%myproduct-3-mysubfacility-123456: Health probe failed
#
grok {
  dictionaryResources : [my-jar-dictionaries/my-commands]
  dictionaryFiles : [kite-morphlines-core/src/test/resources/grok-dictionari
es]
  dictionaryString : """
    MY_CUSTOM_TIMESTAMP %{MONTH} %{MONTHDAY} %{YEAR} %{TIME}
  """
  expressions : {
    message : """<%{POSINT}>%{SYSLOGTIMESTAMP} %{SYSLOGHOST} %{MY_CUSTOM_
TIMESTAMP} : (?<syslog_message>(?<my_message_code>%%{\w+:my_product}-%{\w+:m
y_level}(-%{\w+:my_subfacility})?-%{\w+:my_message_id}): %{GREEDYDATA})"""
  }
}
```

Note: An easy way to test grok out is to use an online grok debugger.

head

The head command (source code) ignores all input records beyond the N-th record, thus emitting at most N records, akin to the Unix head command. This can be helpful to quickly test a morphline with the first few records from a larger dataset.

The command provides the following configuration options:

Property Name	Default	Description
limit	-1	The maximum number of records to emit. -1 indicates never ignore any records.

Example usage:

```
# emit only the first 10 records
head {
  limit : 10
}
```

if

The if command (source code) implements if-then-else conditional control flow. It consists of a chain of zero or more conditions commands, as well as an optional chain of zero or or more commands that are processed if all conditions succeed ("then commands"), as well as an optional chain of zero or more commands that are processed if one of the conditions fails ("else commands").

If one of the commands in the then chain or else chain fails, then the entire if command fails and any remaining commands in the then or else branch are skipped.

The command provides the following configuration options:

Property Name	Default	Description
conditions	[]	A list of zero or more commands.
then	[]	A list of zero or more commands.
else	[]	A list of zero or more commands.

Example usage:

```
if {
  conditions : [
    { contains { _attachment_mimetype : [avro/binary] } }
  ]
  then : [
    { logInfo { format : "processing then..." } }
  ]
  else : [
    { logInfo { format : "processing else..." } }
  ]
}
```

More example usage - Ignore all records that don't have an id field:

```
if {
  conditions : [
    { equals { id : [] } }
  ]
  then : [
    { logTrace { format : "Ignoring record because it has no id: {}", args :
 ["@{}"] } }
    { dropRecord {} }
```

```
  ]
}
```

More example usage - Ignore all records that contain at least one value in the malformed field:

```
if {
  conditions : [
    { not { equals { malformed : [] } } }
  ]
  then : [
    { logTrace { format : "Ignoring record containing at least one malfor
med value: {}", args : ["@{}"] } }
    { dropRecord {} }
  ]
}
```

java

The java command (source code) provides scripting support for Java. The command compiles and executes the given Java code block, wrapped into a Java method with a Boolean return type and several parameters, along with a Java class definition that contains the given import statements.

The following enclosing method declaration is used to pass parameters to the Java code block:

public static boolean evaluate(Record record, com.typesafe.config.Config config, Command parent, Command child, MorphlineContext context, org.slf4j.Logger logger) {

```
  // your custom java code block goes here...
}
```

Compilation is done in main memory, meaning without writing to the filesystem.

The result is an object that can be executed (and reused) any number of times. This is a high performance implementation, using an optimized variant of JSR 223 Java Scripting". Calling eval() just means calling Method.i nvoke(), and, as such, has the same minimal runtime cost. As a result of the low cost, this command can be called on the order of 100 million times per second per CPU core on industry standard hardware.

The command provides the following configuration options:

Property Name	Default	Description
imports	A default list sufficient for typical usage.	A string containing zero or more Java import declarations.
code	[]	A Java code block as defined in the Java language specification. Must return a Boolean value.

Example usage:

```
java {
  imports : "import java.util.*;"
  code: """
    // Update some custom metrics - see http://metrics.codahale.com/getting-
started/
    context.getMetricRegistry().counter("myMetrics.myCounter").inc(1);
    context.getMetricRegistry().meter("myMetrics.myMeter").mark(1);
    context.getMetricRegistry().histogram("myMetrics.myHistogram").update(1
00);
    com.codahale.metrics.Timer.Context timerContext = context.getMetricRegi
stry().timer("myMetrics.myTimer").time();

    // manipulate the contents of a record field
```

```
    List tags = record.get("tags");
    if (!tags.contains("hello")) {
      return false;
    }
    tags.add("world");

    logger.debug("tags: {} for record: {}", tags, record); // log to SLF4J
    timerContext.stop(); // measure how much time the code block took
    return child.process(record); // pass record to next command in chain
        """
}
```

The main disadvantage of the scripting "java" command is that you can't reuse things like compiled regexes across command invocations so you end up having to compile the same regex over and over again, for each record again. The main advantage is that you can implement your custom logic exactly the way you want, without recourse to perhaps overly generic features of certain existing commands.

logTrace, logDebug, logInfo, logWarn, logError

These commands log a message at the given log level to SLF4J. The command can fetch the values of a record field using a field expression, which is a string of the form @{fieldname}. The special field expression @{} can be used to log the entire record.

Example usage:

```
# log the entire record at DEBUG level to SLF4J
logDebug { format : "my record: {}", args : ["@{}"] }
```

More example usage:

```
# log the timestamp field and the entire record at INFO level to SLF4J
logInfo {
  format : "timestamp: {}, record: {}"
  args : ["@{timestamp}", "@{}"]
}
```

To automatically print diagnostic information such as the content of records as they pass through the morphline commands, consider enabling TRACE log level, for example by adding the following line to your log4j.properties file:

```
log4j.logger.org.kitesdk.morphline=TRACE
```

not

The not command (source code) inverts the boolean return value of a nested command. The command consists of one nested command, the Boolean return value of which is inverted.

Example usage:

```
if {
  conditions : [
    {
      not {
        grok {
          ... some grok expressions go here
        }
      }
    }
  ]
  then : [
    { logDebug { format : "found no grok match: {}", args : ["@{}"] } }
```

```
    { dropRecord {} }
  ]
  else : [
    { logDebug { format : "found grok match: {}", args : ["@{}"] } }
  ]
}
```

pipe

The pipe command (source code) pipes a record through a chain of commands. The pipe command has an identifier and contains a chain of zero or more commands, through which records get piped. A command transforms the record into zero or more records. The output records of a command are passed to the next command in the chain. A command has a Boolean return code, indicating success or failure. If any command in the pipe fails (meaning that it returns false), the whole pipe fails (meaning that it returns false), which causes backtracking of the command chain.

Because a pipe is itself a command, a pipe can contain arbitrarily nested pipes. A morphline is a pipe. "Morphline" is simply another name for the pipe at the root of the command tree.

The command provides the following configuration options:

Property Name	Default	Description
id	n/a	An identifier for this pipe.
importCommands	[]	A list of zero or more import specifications, each of which makes all morphline commands that match the specification visible to the morphline. A specification can import all commands in an entire Java package tree (specification ends with ".**"), all commands in a Java package (specification ends with ".*"), or the command of a specific fully qualified Java class (all other specifications). Other commands present on the Java classpath are not visible to this morphline.
commands	[]	A list of zero or more commands.

Example usage demonstrating a pipe with two commands, namely addValues and logDebug:

```
pipe {
  id : my_pipe

  # Import all commands in these java packages, subpackages and classes.
  # Other commands on the Java classpath are not visible to this morphline.
  importCommands : [
    "org.kitesdk.**",   # package and all subpackages
    "org.apache.solr.**", # package and all subpackages
    "com.mycompany.mypackage.*", # package only
    "org.kitesdk.morphline.stdlib.GrokBuilder" # fully qualified class
  ]

  commands : [
    { addValues { foo : bar }}
    { logDebug { format : "output record: {}", args : ["@{}"] } }
  ]
}
```

removeFields

The removeFields command (source code) removes all record fields for which the field name matches at least one of the given blacklist predicates, but matches none of the given whitelist predicates.

A predicate can be a regex pattern (e.g. "regex:foo.*") or POSIX glob pattern (e.g. "glob:foo*") or literal pattern (e.g. "literal:foo") or "*" which is equivalent to "glob:*".

The command provides the following configuration options:

Property Name	Default	Description
blacklist	"*"	The blacklist predicates to use. If the blacklist specification is absent it defaults to MATCH ALL.
whitelist	""	The whitelist predicates to use. If the whitelist specification is absent it defaults to MATCH NONE.

Example usage:

```
# Remove all fields where the field name matches at least one of foo.* or ba
r* or baz,
# but matches none of foobar or baro*
removeFields {
  blacklist : ["regex:foo.*", "glob:bar*", "literal:baz"]
  whitelist: ["literal:foobar", "glob:baro*"]
}
```

Input record:

```
foo:data
foobar:data
barx:data
barox:data
baz:data
hello:data
```

Expected output:

```
foobar:data
barox:data
hello:data
```

removeValues

The removeValues command (source code) removes all record field values for which all of the following conditions hold:

1) the field name matches at least one of the given nameBlacklist predicates but none of the given nameWhitelist predicates.

2) the field value matches at least one of the given valueBlacklist predicates but none of the given valueWhitelist predicates.

A predicate can be a regex pattern (e.g. "regex:foo.*") or POSIX glob pattern (e.g. "glob:foo*") or literal pattern (e.g. "literal:foo") or "*" which is equivalent to "glob:*".

This command behaves in the same way as the replaceValues command except that maching values are removed rather than replaced.

The command provides the following configuration options:

Property Name	Default	Description
nameBlacklist	"*"	The blacklist predicates to use for entry names (i.e. entry keys). If the blacklist specification is absent it defaults to MATCH ALL.

Property Name	Default	Description
nameWhitelist	""	The whitelist predicates to use for entry names (i.e. entry keys). If the whitelist specification is absent it defaults to MATCH NONE.
valueBlacklist	"*"	The blacklist predicates to use for entry values. If the blacklist specification is absent it defaults to MATCH ALL.
valueWhitelist	""	The whitelist predicates to use for entry values. If the whitelist specification is absent it defaults to MATCH NONE.

Example usage:

```
# Remove all field values where the field name and value matches at least on
e of foo.* or bar* or baz,
# but matches none of foobar or baro*
removeValues {
  nameBlacklist : ["regex:foo.*", "glob:bar*", "literal:baz", "literal:xxxx
"]
  nameWhitelist: ["literal:foobar", "glob:baro*"]
  valueBlacklist : ["regex:foo.*", "glob:bar*", "literal:baz", "literal:xxx
x"]
  valueWhitelist: ["literal:foobar", "glob:baro*"]
}
```

Input record:

```
foobar:data
foo:[foo,foobar,barx,barox,baz,baz,hello]
barx:foo
barox:foo
baz:[foo,foo]
hello:foo
```

Expected output:

```
foobar:data
foo:[foobar,barox,hello]
barox:foo
hello:foo
```

replaceValues

The replaceValues command (source code) replaces all record field values for which all of the following conditions hold:

1) the field name matches at least one of the given nameBlacklist predicates but none of the given nameWhitelist predicates.

2) the field value matches at least one of the given valueBlacklist predicates but none of the given valueWhitelist predicates.

A predicate can be a regex pattern (e.g. "regex:foo.*") or POSIX glob pattern (e.g. "glob:foo*") or literal pattern (e.g. "literal:foo") or "*" which is equivalent to "glob:*".

This command behaves in the same way as the removeValues command except that maching values are replaced rather than removed.

The command provides the following configuration options:

Property Name	Default	Description
nameBlacklist	"*"	The blacklist predicates to use for entry names (i.e. entry keys). If the blacklist specification is absent it defaults to MATCH ALL.
nameWhitelist	""	The whitelist predicates to use for entry names (i.e. entry keys). If the whitelist specification is absent it defaults to MATCH NONE.
valueBlacklist	"*"	The blacklist predicates to use for entry values. If the blacklist specification is absent it defaults to MATCH ALL.
valueWhitelist	""	The whitelist predicates to use for entry values. If the whitelist specification is absent it defaults to MATCH NONE.
replacement	n/a	The replacement string to use for matching entry values.

Example usage:

```
# Replace with "myReplacement" all field values where the field name and val
ue
# matches at least one of foo.* or bar* or baz, but matches none of foobar
or baro*
replaceValues {
  nameBlacklist : ["regex:foo.*", "glob:bar*", "literal:baz", "literal:xxxx
"]
  nameWhitelist: ["literal:foobar", "glob:baro*"]
  valueBlacklist : ["regex:foo.*", "glob:bar*", "literal:baz", "literal:xxx
x"]
  valueWhitelist: ["literal:foobar", "glob:baro*"]
  replacement : "myReplacement"
}
```

Input record:

```
foobar:data
foo:[foo,foobar,barx,barox,baz,baz,hello]
barx:foo
barox:foo
baz:[foo,foo]
hello:foo
```

Expected output:

```
foobar:data
foo:[myReplacement,foobar,myReplacement,barox,myReplacement,myReplacement,he
llo]
barox:foo
baz:[myReplacement,myReplacement]
hello:foo
```

sample

The sample command (source code) forwards each input record with a given probability to its child command, and silently ignores all other input records. Sampling is based on a random number generator. This can be helpful to easily test a morphline with a random subset of records from a large dataset.

The command provides the following configuration options:

Property Name	Default	Description
probability	1.0	The probability that any given record will be forwarded; must be in the range 0.0 (forward no records) to 1.0 (forward all records).
seed	null	An optional long integer that ensures that the series of random numbers will be identical on each morphline run. This can be helpful for deterministic unit testing and debugging. If the seed parameter is absent the pseudo random number generator is initialized with a seed that's obtained from a cryptographically "secure" algorithm, leading to a different sample selection on each morphline run.

Example usage:

```
sample {
  probability : 0.0001
  seed : 12345
}
```

separateAttachments

The separateAttachments command (source code) emits one output record for each attachment in the input record's list of attachments. The result is many records, each of which has at most one attachment.

Example usage:

```
separateAttachments {}
```

setValues

The setValues command (source code) assigns a given list of values (or the contents of another field) to a given field. This command is the same as the addValues command, except that it first removes all values from the given output field, and then it adds new values.

Example usage:

```
setValues {
  # assign values "text/log" and "text/log2" to source_type output field
  source_type : [text/log, text/log2]

  # assign the integer 123 to the pid field
  pid : [123]

  # remove the url field
  url : []

  # assign all values contained in the first_name field to the name field
  name : "@{first_name}"
}
```

split

The split command (source code) divides strings into substrings, by recognizing a separator (a.k.a. "delimiter") which can be expressed as a single character, literal string, regular expression, or grok pattern. This class provides the functionality of Guava's Splitter class as a morphline command, plus it also supports grok dictionaries and regexes in the same way as the grok command, except it doesn't support the grok extraction features.

The command provides the following configuration options:

Property Name	Default	Description
inputField	n/a	The name of the input field.
outputField	null	The name of the field to add output values to, i.e. a single string. Example: tokens. One of outputField or outputFields must be present, but not both.
outputFields	null	The names of the fields to add output values to, i.e. a list of strings. Example: [firstName, lastName, "", age]. An empty string in a list indicates omit this column in the output. One of outputField or outputFields must be present, but not both.
separator	n/a	The delimiting string to search for.
isRegex	false	Whether or not to interpret the separator as a grok pattern (true) or string literal (false).
dictionaryFiles	[]	A list of zero or more local files or directory trees from which to load dictionaries. Only applicable if isRegex is true. See grok command.
dictionaryString	null	An optional inline string from which to load a dictionary. Only applicable if isRegex is true. See grok command.
trim	true	Whether or not to apply the String.trim() method on the output values to be added.
addEmptyStrings	false	Whether or not to add zero length strings to the output field.
limit	-1	The maximum number of items to add to the output field per input field value. -1 indicates unlimited.

Example usage with multiple output field names and literal string as separator:

```
split {
  inputField : message
  outputFields : [first_name, last_name, "", age]
  separator : ","
  isRegex : false
  #separator : """\s*,\s*"""
  #isRegex : true
  addEmptyStrings : false
  trim : true
}
```

Input record:

```
message:"Nadja,Redwood,female,8"
```

Expected output:

```
first_name:Nadja
last_name:Redwood
age:8
```

More example usage with one output field and literal string as separator:

```
split {
  inputField : message
  outputField : substrings
```

```
  separator : ","
  isRegex : false
  #separator : """\s*,\s*"""
  #isRegex : true
  addEmptyStrings : false
  trim : true
}
```

Input record:

```
message:"_a ,_b_ ,c__"
```

Expected output contains a "substrings" field with three values:

```
substrings:_a
substrings:_b_
substrings:c__
```

More example usage with grok pattern or normal regex:

```
split {
  inputField : message
  outputField : substrings
  # dictionaryFiles : [kite-morphlines-core/src/test/resources/grok-dicti
onaries]
  dictionaryString : """COMMA_SURROUNDED_BY_WHITESPACE \s*,\s*"""
  separator : """%{COMMA_SURROUNDED_BY_WHITESPACE}"""
  # separator : """\s*,\s*"""
  isRegex : true
  addEmptyStrings : true
  trim : false
}
```

splitKeyValue

The splitKeyValue command (source code) iterates over the items in a given record input field, interprets each item as a key-value pair where the key and value are separated by the given separator, and adds the pair's value to the record field named after the pair's key. Typically, the input field items have been placed there by an upstream split command with a single output field.

The command provides the following configuration options:

Property Name	Default	Description
inputField	n/a	The name of the input field.
outputFieldPrefix	""	A string to be prepended to each output field name.
separator	"="	The string separating the key from the value.
isRegex	false	Whether or not to interpret the separator as a grok pattern (true) or string literal (false).
dictionaryFiles	[]	A list of zero or more local files or directory trees from which to load dictionaries. Only applicable if isRegex is true. See grok command.
dictionaryString	null	An optional inline string from which to load a dictionary. Only applicable if isRegex is true. See grok command.

Property Name	Default	Description
trim	true	Whether or not to apply the String.trim() method on the output keys and values to be added.
addEmptyStrings	false	Whether or not to add zero length strings to the output field.

Example usage:

```
splitKeyValue {
  inputField : params
  separator : "="
  outputFieldPrefix : "/"
}
```

Input record:

```
params:foo=x
params: foo = y
params:foo
params:fragment=z
```

Expected output:

```
/foo:x
/foo:y
/fragment:z
```

Example usage that extracts data from iptables log file

```
# read each line in the file
{
  readLine {
    charset : UTF-8
  }
}

# extract timestamp and key value pair string
{
  grok {
    dictionaryFiles : [target/test-classes/grok-dictionaries/grok-patterns]
    expressions : {
      message : """%{SYSLOGTIMESTAMP:timestamp} %{GREEDYDATA:key_value_pair
s_string}"""
    }
  }
}

# split key value pair string on blanks into an array of key value pairs
{
  split {
    inputField : key_value_pairs_string
    outputField : key_value_array
    separator : " "
  }
}

# split each key value pair on '=' char and extract its value into record fi
elds named after the key
{
  splitKeyValue {
```

```
    inputField : key_value_array
    outputFieldPrefix : ""
    separator : "="
    addEmptyStrings : false
    trim : true
  }
}

# remove temporary work fields
{
  setValues {
    key_value_pairs_string : []
    key_value_array : []
  }
}
```

Input file:

```
Feb  6 12:04:42 IN=eth1 OUT=eth0 SRC=1.2.3.4 DST=6.7.8.9 ACK DF WINDOW=0
```

Expected output record:

```
timestamp:Feb  6 12:04:42
IN:eth1
OUT:eth0
SRC:1.2.3.4
DST:6.7.8.9
WINDOW:0
```

startReportingMetricsToCSV

The startReportingMetricsToCSV command (source code) starts periodically appending the metrics of all morphline commands to a set of CSV files. The CSV files are named after the metrics.

The command provides the following configuration options:

Property Name	Default	Description
outputDir	n/a	The relative or absolute path of the output directory on the local file system. The directory and it's parent directories will be created automatically if they don't yet exist.
frequency	"10 seconds"	The amount of time between reports to the output file, given in HOCON duration format.
locale	JVM default Locale	Format numbers for the given Java Locale. Example: "en_US"
defaultDurationUnit	milliseconds	Report output durations in the given time unit. One of nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days.
defaultRateUnit	seconds	Report output rates in the given time unit. One of nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days. Example output: events/second
metricFilter	null	Only report metrics which match the given (optional) filter, as described in more detail below. If the filter is absent all metrics match.

metricFilter

A metricFilter uses pattern matching with include/exclude specifications to determine if a given metric shall be reported to the output destination.

A metric consists of a metric name and a metric class name. A metric matches the filter if the metric matches at least one include specification, but matches none of the exclude specifications. An include/exclude specification consists of zero or more expression pairs. Each expression pair consists of an expression for the metric name, as well as an expression for the metric's class name. Each expression can be a regex pattern (e.g. "regex:foo.*") or POSIX glob pattern (e.g. "glob:foo*") or literal string (e.g. "literal:foo") or "*" which is equivalent to "glob:*". Each expression pair defines one expression for the metric name and another expression for the metric class name.

If the include specification is absent it defaults to MATCH ALL. If the exclude specification is absent it defaults to MATCH NONE.

Example startReportingMetricsToCSV usage:

```
startReportingMetricsToCSV {
  outputDir : "mytest/metricsLogs"
  frequency : "10 seconds"
  locale : en_US
}
```

More example startReportingMetricsToCSV usage:

```
startReportingMetricsToCSV {
  outputDir : "mytest/metricsLogs"
  frequency : "10 seconds"
  locale : en_US
  defaultDurationUnit : milliseconds
  defaultRateUnit : seconds
  metricFilter : {
    includes : { # if absent defaults to match all
      "literal:foo" : "glob:foo*"
      "regex:.*" : "glob:*"
    }
    excludes : { # if absent defaults to match none
      "literal:foo.bar" : "*"
    }
  }
}
```

Example output log file:

```
t,count,mean_rate,m1_rate,m5_rate,m15_rate,rate_unit
1380054913,2,409.752100,0.000000,0.000000,0.000000,events/second
1380055913,2,258.131131,0.000000,0.000000,0.000000,events/second
```

startReportingMetricsToJMX

The startReportingMetricsToJMX command (source code) starts publishing the metrics of all morphline commands to JMX.

The command provides the following configuration options:

Property Name	Default	Description
domain	metrics	The name of the JMX domain (aka category) to publish to.
durationUnits	null	Report output durations of the given metrics in the given time units. This optional parameter is a JSON object where the key is the metric name and the value is a time unit. The time unit can be one of nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days.

Property Name	Default	Description
defaultDurationUnit	milliseconds	Report all other output durations in the given time unit. One of nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days.
rateUnits	null	Report output rates of the given metrics in the given time units. This optional parameter is a JSON object where the key is the metric name and the value is a time unit. The time unit can be one of nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days.
defaultRateUnit	seconds	Report all other output rates in the given time unit. One of nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days. Example output: events/second
metricFilter	null	Only report metrics which match the given (optional) filter, as described in more detail at metricFilter. If the filter is absent all metrics match.

Example startReportingMetricsToJMX usage:

```
startReportingMetricsToJMX {
  domain : myMetrics
}
```

More example startReportingMetricsToJMX usage:

```
startReportingMetricsToJMX {
  domain : myMetrics
  durationUnits : {
    myMetrics.myTimer : minutes
  }
  defaultDurationUnit : milliseconds
  rateUnits : {
    myMetrics.myTimer : milliseconds
    morphline.logDebug.numProcessCalls : milliseconds
  }
  defaultRateUnit : seconds
  metricFilter : {
    includes : { # if absent defaults to match all
      "literal:foo" : "glob:foo*"
      "regex:.*" : "glob:*"
    }
    excludes : { # if absent defaults to match none
      "literal:foo.bar" : "*"
    }
  }
}
```

startReportingMetricsToSLF4J

The startReportingMetricsToSLF4J command (source code) starts periodically logging the metrics of all morphline commands to SLF4J.

The command provides the following configuration options:

Property Name Default Description

logger metrics The name of the SLF4J logger to write to. marker null The optional name of the SLF4J marker object to associate with each logging request. frequency "10 seconds" The amount of time between reports to the output file, given in HOCON duration format. defaultDurationUnit milliseconds Report output durations in the given time unit.

One of nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days. defaultRateUnit seconds Report output rates in the given time unit. One of nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days. Example output: events/second metricFilter null Only report metrics which match the given (optional) filter, as described in more detail at metricFilter. If the filter is absent all metrics match.

Example startReportingMetricsToSLF4J usage:

```
startReportingMetricsToSLF4J {
  logger : "org.kitesdk.morphline.domain1"
  frequency : "10 seconds"
}
```

More example startReportingMetricsToSLF4J usage:

```
startReportingMetricsToSLF4J {
  logger : "org.kitesdk.morphline.domain1"
  frequency : "10 seconds"
  defaultDurationUnit : milliseconds
  defaultRateUnit : seconds
  metricFilter : {
    includes : { # if absent defaults to match all
      "literal:foo" : "glob:foo*"
      "regex:.*" : "glob:*"
    }
    excludes : { # if absent defaults to match none
      "literal:foo.bar" : "*"
    }
  }
}
```

Example output log line:

```
457  [metrics-logger-reporter-thread-1] INFO  org.kitesdk.morphline.domain1
 - type=METER, name=morphline.logDebug.numProcessCalls, count=2, mean_rate=1
44.3001443001443, m1=0.0, m5=0.0, m15=0.0, rate_unit=events/second
```

toByteArray

The toByteArray command (source code) converts the Java objects in a given field via Object.toString() to their string representation, and then via String.getBytes(Charset) to their byte array representation. If the input Java objects are already byte arrays the command does nothing.

The command provides the following configuration options:

Property Name	Default	Description
field	n/a	The name of the field to convert.
charset	UTF-8	The character encoding to use.

Example usage:

```
toByteArray { field : _attachment_body }
```

toString

The toString command (source code) converts the Java objects in a given field using the Object.toString() method to their string representation, and optionally also applies the String.trim() method to remove leading and trailing whitespace.

The command provides the following configuration options:

Property Name	Default	Description
field	n/a	The name of the field to convert.
trim	false	Whether or not to apply the String.trim() method.

Example usage:

```
toString { field : source_type }
```

translate

The translate command (source code) examines each value in a given field and replaces it with the replacement value defined in a given dictionary aka lookup hash table.

The command provides the following configuration options:

Property Name	Default	Description
field	n/a	The name of the field to modify.
dictionary	n/a	The lookup hash table to use for finding matches and replacement values
fallback	null	The fallback value to use as replacement if no match is found. If no fallback is defined and no match is found then the command fails.

Example usage to translate Syslog severity level numeric codes to string labels:

```
translate {
  field : level
  dictionary : {
     0 : Emergency
     1 : Alert
     2 : Critical
     3 : Error
     4 : Warning
     5 : Notice
     6 : Informational
     7 : Debug
  }
  fallback : Unknown # if no fallback is defined and no match is found then
 the command fails
}
```

Input: level:0

Expected output: level:Emergency

Input: level:999

Expected output: level:Unknown

tryRules

The tryRules command (source code) is a simple rule engine for handling a list of heterogeneous input data formats. The command consists of zero or more rules. A rule consists of zero or more commands.

The rules of a tryRules command are processed in top-down order. If one of the commands in a rule fails, the tryRules command stops processing this rule, backtracks and tries the next rule, and so on, until a rule is found that runs all its commands to completion without failure (the rule succeeds). If a rule succeeds, the remaining rules of the current

tryRules command are skipped. If no rule succeeds the record remains unchanged, but a warning may be issued or an exception may be thrown.

Because a tryRules command is itself a command, a tryRules command can contain arbitrarily nested tryRules commands. By the same logic, a pipe command can contain arbitrarily nested tryRules commands and a tryRules command can contain arbitrarily nested pipe commands. This helps to implement complex functionality for advanced usage.

The command provides the following configuration options:

Property Name	Default	Description
catchExceptions	false	Whether Java exceptions thrown by a rule shall be caught, with processing continuing with the next rule (true), or whether such exceptions shall not be caught and consequently propagate up the call chain (false).
throwExceptionIfAllRulesFailed	true	Whether to throw a Java exception if no rule succeeds.

Example usage:

```
tryRules {
  catchExceptions : false
  throwExceptionIfAllRulesFailed : true
  rules : [
    # next rule of tryRules cmd:
    {
      commands : [
        { contains { _attachment_mimetype : [avro/binary] } }
        ... handle Avro data here
        { logDebug { format : "output record: {}", args : ["@{}"] } }
      ]
    }

    # next rule of tryRules cmd:
    {
      commands : [
        { contains { _attachment_mimetype : [text/csv] } }
        ... handle CSV data here
        { logDebug { format : "output record: {}", args : ["@{}"] } }
      ]
    }

    # if desired, the last rule can serve as a fallback mechanism
    # for records that don't match any rule:
    {
      commands : [
        { logWarn { format : "Ignoring record with unsupported input format:
 {}", args : ["@{}"] } }
        { dropRecord {} }
      ]
    }
  ]
}
```

kite-morphlines-avro

This maven module contains morphline commands for reading, extracting, and transforming Avro files and Avro objects.

readAvroContainer

The readAvroContainer command (source code) parses an InputStream or byte array that contains Apache Avro binary container file data. For each Avro datum, the command emits a morphline record containing the datum as an attachment in the field _attachment_body.

The Avro schema that was used to write the Avro data is retrieved from the Avro container. Optionally, the Avro schema that shall be used for reading can be supplied with a configuration option; otherwise it is assumed to be the same as the writer schema.

Note: Avro uses Schema Resolution if the two schemas are different, e.g. if the reader schema is a subset of the writer schema for the purpose of efficient column projection.

The input stream or byte array is read from the first attachment of the input record.

The command provides the following configuration options:

Property Name	Default	Description
supportedMimeTypes	null	Optionally, require the input record to match one of the MIME types in this list.
readerSchemaFile	null	An optional Avro schema file in JSON format on the local file system to use for reading.
readerSchemaString	null	An optional Avro schema in JSON format given inline to use for reading.

Example usage:

```
# Parse Avro container file and emit a record for each avro object
readAvroContainer {
  # Optionally, require the input to match one of these MIME types:
  # supportedMimeTypes : [avro/binary]

  # Optionally, use this Avro schema in JSON format inline for reading:
  # readerSchemaString : """<json can go here>"""

  # Optionally, use this Avro schema file in JSON format for reading:
  # readerSchemaFile : /path/to/syslog.avsc
}
```

readAvro

The readAvro command (source code) parses containerless Avro. This command is the same as the readAvroContainer command except that the Avro schema that was used to write the Avro data must be explicitly supplied to the readAvro command because it expects raw Avro data without an Avro container and hence without a built-in writer schema.

Optionally, the Avro schema that shall be used for reading can be supplied with a configuration option; otherwise it is assumed to be the same as the writer schema.

Note: Avro uses Schema Resolution if the two schemas are different, e.g. if the reader schema is a subset of the writer schema for the purpose of efficient column projection.

Note: For the readAvro command to work correctly, each Avro event must have been written with the same writer schema by the ingesting app. That is, you cannot parse two Avro events with two different writer schemas A and B within the same readAvro command. The readAvroContainer command doesn't have that limitation, of course, because the writer schema comes embedded inside each Avro container, per the standard Avro container specification.

The command provides the following configuration options:

Property Name	Default	Description
supportedMimeTypes	null	Optionally, require the input record to match one of the MIME types in this list.
readerSchemaFile	null	An optional Avro schema file in JSON format on the local file system to use for reading.
readerSchemaString	null	An optional Avro schema in JSON format given inline to use for reading.
writerSchemaFile	null	The Avro schema file in JSON format that was used to write the Avro data.
writerSchemaString	null	The Avro schema file in JSON format that was used to write the Avro data, given inline.
isJson	false	Whether the Avro input data is encoded as JSON or binary.

Example usage:

```
# Parse Avro and emit a record for each avro object
readAvro {
  # supportedMimeTypes : [avro/binary]
  # readerSchemaString : """<json can go here>"""
  # readerSchemaFile : test-documents/sample-statuses-20120906-141433-subsch
ema.avsc
  # writerSchemaString : """<json can go here>"""
  writerSchemaFile : test-documents/sample-statuses-20120906-141433.avsc
}
```

extractAvroTree

The extractAvroTree command (source code) converts an attached Avro datum to a morphline record by recursively walking the Avro tree and extracting all data into a single morphline record, with fields named by their path in the Avro tree.

The Avro input object is expected to be contained in the field _attachment_body, and typically placed there by an upstream readAvroContainer or readAvro command.

This kind of mapping is useful for simple Avro schemas, but for more complex schemas, this approach may be overly simplistic and expensive.

The command provides the following configuration options:

Property Name	Default	Description
outputFieldPrefix	""	A string to be prepended to each output field name.

Example usage:

```
extractAvroTree {
  outputFieldPrefix : ""
}
```

extractAvroPaths

The extractAvroPaths command (source code) extracts specific values from an Avro object, akin to a simple form of XPath. The command uses zero or more Avro path expressions to extract values from an Avro object.

The Avro input object is expected to be contained in the field _attachment_body, and typically placed there by an upstream readAvroContainer or readAvro command.

Each path expression consists of a record output field name (on the left side of the colon ':') as well as zero or more path steps (on the right hand side), each path step separated by a '/' slash, akin to a simple form of XPath. Avro arrays are traversed with the '[]' notation.

The result of a path expression is a list of objects, each of which is added to the given record output field.

The path language supports all Avro concepts, including such concepts as nested structures, records, arrays, maps, and unions. The path language supports a flatten option that collects the primitives in a subtree into a flat output list.

The command provides the following configuration options:

Property Name	Default	Description
flatten	true	Whether to collect the primitives in a subtree into a flat output list.
paths	[]	Zero or more Avro path expressions.

Example usage:

```
extractAvroPaths {
  flatten : true
  paths : {
    my_price : /price

    my_docId : /docId
    my_links : /links
    my_links_backward : "/links/backward"
    my_links_forward : "/links/forward"
    my_name_language_code : "/name[]/language[]/code"
    my_name_language_country : "/name[]/language[]/country"
    my_name : /name

    /mymapField/foo/label : /mapField/foo/label/
  }
}
```

Alternatively, if the extractAvroPaths and extractAvroTree commands don't fit your needs you can instead implement your own custom morphline command or script a java command config that uses the Generic Avro Java API to arbitrarily traverse and process the Avro tree that is emitted by the readAvroContainer and readAvro commands. For example, along the following lines:

```
{
  readAvroContainer { }
}

{
  java {
    imports : """
      import org.apache.avro.generic.GenericRecord;
      import org.kitesdk.morphline.base.Fields;
      // import com.cloudera.cdk.morphline.base.Fields; // use this for CDK
    """
    code : """
      GenericRecord root = (GenericRecord) record.getFirstValue(Fields.AT
TACHMENT_BODY);
```

```
      GenericRecord links = (GenericRecord) root.get("links"); // traverse v
ia Avro Tree API
      String forwardLinks = links.get("forward").toString(); // traverse via
 Avro Tree API
      record.put("forwardLinks", forwardLinks);
      logger.debug("My output record: {}", record);
      return child.process(record);
    """
    }
  }
}
```

toAvro

The toAvro command (source code) converts a morphline record to an Avro record of Java class org.apache.avro.generic.IndexedRecord.

The conversion supports all Avro concepts, including such concepts as nested structures, records, arrays, maps, and unions.

The Avro output record object is added to the morphline field _attachment_body.

The command provides the following configuration options:

Property Name	Default	Description
schemaFile	null	An optional Avro schema file in JSON format on the local file system to use for writing.
schemaString	null	An optional Avro schema in JSON format given inline to use for writing.
schemaField	null	An optional org.apache.avro.Schema object fetched from the given record input field. One of schemaFile or schemaString or schemaField must be present, but not more than one.
mappings	[]	An optional JSON object containing zero or more mappings from morphline record field names to Avro record field names. Each mapping consists of an Avro output field name (on the left side of the colon ':') as well as a Morphline field name (on the right hand side). Example mapping: avroPrice : morphlinePrice. Any such mappings are optional - by default data is extracted from the morphline fields that carry the same name as the Avro fields defined in the Avro schema.

Example usage:

```
toAvro {
  #schemaFile : /path/to/interop.avsc
  #schemaField : _dataset_descriptor_schema
  schemaString : """
    {
      "type" : "record",
      "name" : "Rating",
      "fields" : [
        {
          "name" : "userId",
          "type" : "int"
        },
        {
          "name" : "rating",
          "type" : ["int","null"]
```

```
        },
        {
          "name" : "reviews",
          "type" : {"type": "array", "items": "string"}
        },
        {
          "name" : "history",
          "type" : ["null", {"type": "map", "values":
                                {"type": "record", "name": "Foo",
                                     "fields": [{"name": "timestamp", "type":
 "long"}]}}]
        }
      ]
    }
  """
  mappings : {
    userId : morphlineUserId
  }
}
```

writeAvroToByteArray

The writeAvroToByteArray command (source code) serializes the Avro records contained in the _attachment_body field into a byte array and replaces the _attachment_body field with that byte array. The records must share an identical Avro schema. Often, the records were originally generated by the toAvro command.

The command provides the following configuration options:

Property Name	Default	Description
format	container	Indicates the type of Avro output format that shall be written. Must be one of container (serialize into a byte array that contains Apache Avro binary container file data) or containerlessBinary (serialize into a byte array that contains Apache Avro without an Avro container and hence without a built-in writer schema) or containerlessJSON (same as containerlessBinary except that Avro output is encoded as JSON).
codec	null	Optional parameter that specifies the compression algorithm to use. Must be one of null or snappy or deflate or bzip2. This parameter only applies if format = container.

Example usage:

```
writeAvroToByteArray {
  format : container
  codec : snappy
}
```

kite-morphlines-json

This maven module contains morphline commands for reading, extracting, and transforming JSON files and JSON objects.

readJson

The readJson command (source code) parses an InputStream or byte array that contains JSON data, using the Jackson library. For each top level JSON object, the command emits a morphline record containing the top level object as an attachment in the field _attachment_body.

The input stream or byte array is read from the first attachment of the input record.

The command provides the following configuration options:

Property Name	Default	Description
outputClass	com.fasterxml.jackson.databind.JsonNode	The fully qualified name of a Java class that Jackson shall convert to.

Example usage:

```
readJson {}
```

Example usage with conversion from JSON to java.util.Map objects:

```
readJson {
  outputClass : java.util.Map
}
```

extractJsonPaths

The extractJsonPaths command (source code) extracts specific values from a JSON object, akin to a simple form of XPath. The command uses zero or more JSON path expressions to extract values from a Jackson JSON object of outputClass com.fasterxml.jackson.databind.JsonNode.

The JSON input object is expected to be contained in the field _attachment_body, and typically placed there by an upstream readJson command with outputClass : com.fasterxml.jackson.databind.JsonNode.

Each path expression consists of a record output field name (on the left side of the colon ':') as well as zero or more path steps (on the right hand side), each path step separated by a '/' slash, akin to a simple form of XPath. JSON arrays are traversed with the '[]' notation.

The result of a path expression is a list of objects, each of which is added to the given record output field.

The path language supports all JSON concepts, including such concepts as nested objects, arrays, etc. The path language supports a flatten option that collects the primitives in a subtree into a flat output list.

The command provides the following configuration options:

Property Name	Default	Description
flatten	true	Whether to collect the primitives in a subtree into a flat output list.
paths	[]	Zero or more JSON path expressions.

Example usage:

```
extractJsonPaths {
  flatten : true
  paths : {
    my_price : /price

    my_docId : /docId
    my_links : /links
    my_links_backward : "/links/backward"
    my_links_forward : "/links/forward"
    my_name_language_code : "/name[]/language[]/code"
```

```
    my_name_language_country : "/name[]/language[]/country"
    my_name : /name
  }
}
```

Alternatively, if the extractJsonPaths command doesn't fit your needs you can instead implement your own Custom Morphline Command or script a java command config that uses the com.fasterxml.jackson.databind.JsonNode Java API to arbitrarily traverse and process the Jackson Json tree that is emitted by the readJson command. For example, along the following lines:

```
{
  readJson { }
}
{
  java {
    imports : """
      import com.fasterxml.jackson.databind.JsonNode;
      import org.kitesdk.morphline.base.Fields;
      // import com.cloudera.cdk.morphline.base.Fields; // use this for CDK
    """
    code : """
      JsonNode rootNode = (JsonNode) record.getFirstValue(Fields.ATTACHMENT_
BODY);
      String forwardLinks = rootNode.get("links").get("forward").asText()
; // traverse via Jackson Tree API
      record.put("forwardLinks", forwardLinks);
      logger.debug("My output record: {}", record);
      return child.process(record);
    """
    }
  }
}
```

kite-morphlines-hadoop-core

downloadHdfsFile

The downloadHdfsFile command (source code) downloads, on startup, zero or more files or directory trees from HDFS to the local file system. These files are typically static configuration files that are required by downstream morphline commands, e.g. Avro schema files, XML join tables, grok dictionaries, etc. Storing such configuration files in HDFS can help with consistent centralized configuration management across a set of cluster nodes.

The output directory on the local file system defaults to the current working directory of the current process. If the effective output file or directory already exists it will be deleted and overwritten.

The command provides the following configuration options:

Property Name	Default	Description
inputFiles		The HDFS files or directories to download, in the form of a list of HDFS URIs.
outputDir	"."	The relative or absolute path of the destination directory on the local file system. Parent directories of that directory will be created automatically. Defaults to the current working directory of the current process.

Example usage:

```
downloadHdfsFile {
  inputFiles : ["hdfs://c2202.mycompany.com/user/foo/configs/sample-schem
a.avsc"]
  outputDir : "myconfigs"
}
```

openHdfsFile

The openHdfsFile command (source code) opens an HDFS file for read and returns a corresponding Java InputStream.

The morphline record input field _attachment_body must contain the HDFS Path of the file to read. The command replaces the HDFS Path in this field with the corresponding Java InputStream. Said InputStream can then be parsed with other commands, such as readLine on page 13 or similar.

The command automatically handles gzip files if the file path ends with the ".gz" file name extensions.

Example usage:

```
openHdfsFile {}
```

kite-morphlines-hadoop-parquet-avro

This maven module contains morphline commands for handling Hadoop Avro Parquet files.

readAvroParquetFile

The readAvroParquetFile command (source code) parses a Hadoop Parquet file and emits a morphline record for each contained Avro datum.

The morphline record input field file_upload_url must contain the HDFS Path of the Parquet file to read. (This field is already provided out of the box with MapReduceIndexerTool).

For each Avro datum, the command emits a morphline record containing the datum as an attachment in the field _attachment_body. Typically, the emitted Avro datum is further post-processed with downstream commands such as extractAvroPaths.

Optionally, an Avro schema that shall be used for projecting parquet columns can be supplied with a configuration option.

The command provides the following configuration options:

Property Name	Default	Description
decimalConversionEnabled	false	When set to true, decimal Parquet data is correctly read instead of returning raw bytes.
projectionSchemaFile	null	An optional Avro schema file in JSON format on the local file system to use for projection. This Avro schema is converted to a parquet schema before applying the projection.
projectionSchemaString	null	An optional Avro schema in JSON format given inline to use for projection. This Avro schema is converted to a parquet schema before applying the projection.

Property Name	Default	Description
readerSchemaFile	null	This optional parameter is available in CDH 5.0 and beyond. This optional parameter specifies an Avro schema file in JSON format on the local file system to use for reading, as discussed above.
readerSchemaString	null	This optional parameter is available in CDH 5.0 and beyond. This optional parameter specifies an optional Avro schema in JSON format given inline to use for reading. Has identical behaviour as the readerSchemaFile parameter described above.

Example usage:

```
readAvroParquetFile {
  # Optionally, use this Avro schema in JSON format inline for projection:
  # projectionSchemaString : """<json can go here>"""

  # Optionally, use this Avro schema file in JSON format for projection:
  # projectionSchemaFile : /path/to/syslog.avsc
}
```

kite-morphlines-hadoop-rcfile

readRCFile

The readRCFile command (source code) parses an Apache Hadoop RCFile. An RCFile can be read Row-wise or Column-wise, as follows:

- Row-wise: One morphline record is emitted for each row in the RCFile. Each record will contain fields for all the columns configured in the columns parameter. For example, with an RCFile with 10 rows and 5 columns, Row-wise mode would emit 10 morphline records.
- Column-wise: For every row-split (block) in the RC File, Emits one record for each column specified in the colu mns parameter. This record will contain a list of row values for that column as a list. The order of columns is as specified in the columns parameter. For example, with an RCFile with 10 rows and 5 columns, Column-wise mode would emit 5 morphline records each with a list of 10 values assuming there are no row-splits.

The InputStream of the RCFile is read from the _attachment_body field of the input record. Optionally, the name of the RCFile is read from the _attachment_name field of the input record. Providing a name for the InputStream will, in case of errors, result in error messages containing said name for better debugging and diagnostics.

The command automatically handles compressed RCFiles.

The command provides the following configuration options:

Property Name	Default	Description
readMode	row	Valid values: row or column. Defines the reading strategy. Affects the structure and number of output records that will be emitted, as discussed above.
includeMetaData	false	Whether or not the RCFile metadata shall be included in the output record.
columns	n/a	A list of column configurations. Since an RCFile does not store meta information about the columns itself, this configuration is necessary to read the RCFile.

The columns configuration has the following configuration options:

Property Name	Default	Description
inputField	n/a	Non-negative integer index of the RCFile column to read.
outputField	n/a	The name of the field to add output values to. The output record will have the value of the column added to this field.
writableClass	n/a	Fully Qualified Class Name of a sub-class of org.apache.hadoop.io.Writable. Instances of this class are used to read the value of the column bytes, and said instances are added to the outputField. For example org.apache.hadoop.io.Text or org.apache.hadoop.io.LongWritable or org.apache.hadoop.hive.serde2.columnar.BytesRefWritable

Example usage:

```
readRCFile {
  readMode: row
  includeMetaData: false
  columns: [
    {
      inputField: 0
      outputField: name
      writableClass: "org.apache.hadoop.io.Text"
    }
    {
      inputField: 3
      outputField: age
      writableClass: "org.apache.hadoop.io.LongWritable"
    }
    {
      inputField: 1000
      outputField: photo
      writableClass: "org.apache.hadoop.hive.serde2.columnar.BytesRefWrita
ble"
    }
  ]
}
```

kite-morphlines-hadoop-sequencefile

readSequenceFile

The readSequenceFile command (source code) parses an Apache Hadoop SequenceFile and emits a morphline record for each contained key-value pair. The sequence file is read from the input stream of the first attachment of the record.

The command automatically handles Record-Compressed and Block-Compressed SequenceFiles.

The command provides the following configuration options:

Property Name	Default	Description
keyField	_attachment_name	The name of the output field to store the SequenceFile Record key.

Property Name	Default	Description
valueField	_attachment_body	The name of the output field to store the SequenceFile Record value.

Example usage:

```
readSequenceFile {
  keyField : "key"
  valueField : "value"
}
```

kite-morphlines-maxmind

geoIP

The geoIP command (source code) returns Geolocation information for a given IP address, using an efficient in-memory Maxmind database lookup. The command stores a corresponding Jackson JsonNode Java object into the _attachment_body record field. The most recent version of the Maxmind GeoLite2 database can be downloaded as a flat data file from Maxmind.

Often, the geoIP command is combined with commands such as extractJsonPaths.

The command provides the following configuration options:

Property Name	Default	Description
inputField	n/a	The name of the input field that contains zero or more IP addresses.
database	GeoLite2-City.mmdb	The relative or absolute path of a Maxmind database file on the local file system. Example: /path/to/GeoLite2-City.mmdb

Example usage:

```
# extract geolocation info into a Jackson JsonNode Java object
# and store it into the _attachment_body field:
geoIP {
  inputField : ip
  database : "target/test-classes/GeoLite2-City.mmdb"
}

# extract parts of the geolocation info from the Jackson JsonNode Java
# object contained in the _attachment_body field and store the parts in
# the given record output fields:
extractJsonPaths {
  flatten : false
  paths : {
    /country/iso_code : /country/iso_code
    /country/names/en : /country/names/en
    /country/names/zh-CN : /country/names/zh-CN
    "/subdivisions[]/names/en" : "/subdivisions[]/names/en"
    "/subdivisions[]/iso_code" : "/subdivisions[]/iso_code"
    /city/names/en : /city/names/en
    /postal/code : /postal/code
    /location/latitude : /location/latitude
    /location/longitude : /location/longitude
    /location/latitude_longitude : /location/latitude_longitude
    /location/longitude_latitude : /location/longitude_latitude
```

```
  }
}
```

Example geoIP JSON output with extractJsonPaths:

Input: ip: 128.101.101.101

Expected output:

```
ip: 128.101.101.101
/country/iso_code: US
/country/names/en: United States
/country/names/zh-CN: ##
/subdivisions[]/names/en: Minnesota
/subdivisions[]/iso_code: MN
/city/names/en: Minneapolis
/postal/code: 55455
/location/latitude: 44.9733
/location/longitude: -93.2323
/location/latitude_longitude: 44.9733,-93.2323
/location/longitude_latitude: -93.2323,44.9733
```

Example geoIP JSON output:

Input: ip: 128.101.101.101

Expected output:

```
{
   "city":{
      "geoname_id":5037649,
      "names":{
         "de":"Minneapolis",
         "en":"Minneapolis",
         "es":"Mineápolis",
         "fr":"Minneapolis",
         "ja":"######",
         "pt-BR":"Minneapolis",
         "ru":"###########",
         "zh-CN":"######"
      }
   },
   "continent":{
      "code":"NA",
      "geoname_id":6255149,
      "names":{
         "de":"Nordamerika",
         "en":"North America",
         "es":"Norteamérica",
         "fr":"Amérique du Nord",
         "ja":"#####",
         "pt-BR":"América do Norte",
         "ru":"######## #######",
         "zh-CN":"###"
      }
   },
   "country":{
      "geoname_id":6252001,
      "iso_code":"US",
      "names":{
         "de":"USA",
         "en":"United States",
         "es":"Estados Unidos",
         "fr":"États-Unis",
```

```
            "ja":"#######",
            "pt-BR":"Estados Unidos",
            "ru":"###",
            "zh-CN":"##"
         }
      },
      "location":{
         "latitude":44.9733,
         "longitude":-93.2323,
         "metro_code":"613",
         "time_zone":"America/Chicago"
         "latitude_longitude":"44.9733,-93.2323"
         "longitude_latitude":"-93.2323,44.9733"
      },
      "postal":{
         "code":"55455"
      },
      "registered_country":{
         "geoname_id":6252001,
         "iso_code":"US",
         "names":{
            "de":"USA",
            "en":"United States",
            "es":"Estados Unidos",
            "fr":"États-Unis",
            "ja":"#######",
            "pt-BR":"Estados Unidos",
            "ru":"###",
            "zh-CN":"##"
         }
      },
      "subdivisions":[
         {
            "geoname_id":5037779,
            "iso_code":"MN",
            "names":{
               "en":"Minnesota",
               "es":"Minnesota",
               "ja":"#####",
               "ru":"########"
            }
         }
      ]
}
```

kite-morphlines-metrics-servlets

registerJVMMetrics

The registerJVMMetrics command (source code) registers metrics that are related to the Java Virtual Machine with the MorphlineContext of the morphline. For example, this includes metrics for garbage collection events, buffer pools, threads and thread deadlocks.

Often, the registerJVMMetrics command is combined with commands such as startReportingMetricsToHTTP or startReportingMetricsToJMX or startReportingMetricsToSLF4J or startReportingMetricsToCSV.

Example usage:

```
registerJVMMetrics {}
```

startReportingMetricsToHTTP

Status: EXPERIMENTAL

The startReportingMetricsToHTTP command (source code) exposes liveness status, health check status, metrics state and thread dumps via a set of HTTP URLs served by Jetty, using the AdminServlet.

On startup, a Jetty HTTP server is created that listens on a configurable port. If an HTTP server isn't required for your use case, and reporting metrics to JMX (or SLF4J or CSV) is sufficient, consider command such as startReportingMetricsToJMX or startReportingMetricsToSLF4J or startReportingMetricsToCSV.

Often, the startReportingMetricsToHTTP command is combined with the registerJVMMetrics command.

The following HTTP URLs are provided:

URL Path	Servlet	Description
/	AdminServlet	an HTML admin menu, for example at http://foo.com:8080/, with links to the following servlets:
/ping	PingServlet	PingServlet responds to GET requests with a text/plain/200 OK response of pong. This is useful for determining liveness for load balancers, etc.
/healthcheck	HealthCheckServlet	HealthCheckServlet responds to GET requests by running all the health checks registered with the morphline context, and returning 501 Not Implemented if no health checks are registered, 200 OK if all pass, or 500 Internal Service Error if one or more fail. The results are returned as a human-readable text/plain JSON entity.
/metrics	MetricsServlet	MetricsServlet exposes the state of the metrics registered with the morphline context as a JSON object.
/threads	ThreadDumpServlet	ThreadDumpServlet responds to GET requests with a text/plain representation of all the live threads in the JVM, their states, their stack traces, and the state of any locks they may be waiting for.

The command provides the following configuration options:

Property Name	Default	Description
port	8080	The port on which the HTTP server shall listen.
defaultDurationUnit	milliseconds	Report output durations in the given time unit. One of nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days.
defaultRateUnit	seconds	Report output rates in the given time unit. One of nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days. Example output: events/second

Example startReportingMetricsToHTTP Usage:

```
startReportingMetricsToHTTP {
  port : 8080
}
```

Example startReportingMetricsToHTTP ping output:

Here is the response to an HTTP GET to http://localhost:8080/ping for liveness check:

```
pong
```

Example startReportingMetricsToHTTP healthcheck output:

Example output of running healthchecks via a HTTP GET to http://localhost:8080/healthcheck

```
{"deadlocks":{"healthy":true}}
```

Example startReportingMetricsToHTTP metrics output:

For an example on how to update user defined custom metrics such as counters, meters, timers and histograms, see the java command. Here is an example output of the JSON metrics reported by an HTTP GET to http://localhost:8080/metrics?pretty=true

```
{
 "version" : "3.0.0",
 "gauges" : {
   "jvm.gc.ConcurrentMarkSweep.count" : {
     "value" : 0
   },
   "jvm.gc.ConcurrentMarkSweep.time" : {
     "value" : 0
   },
   "jvm.gc.ParNew.count" : {
     "value" : 4
   },
   "jvm.gc.ParNew.time" : {
     "value" : 29
   },
   "jvm.memory.heap.committed" : {
     "value" : 85000192
   },
   "jvm.memory.heap.init" : {
     "value" : 0
   },
   "jvm.memory.heap.max" : {
     "value" : 129957888
   },
   "jvm.memory.heap.usage" : {
     "value" : 0.1319703810514372
   },
   "jvm.memory.heap.used" : {
     "value" : 17150592
   },
   "jvm.memory.non-heap.committed" : {
     "value" : 24711168
   },
   "jvm.memory.non-heap.init" : {
     "value" : 24317952
   },
   "jvm.memory.non-heap.max" : {
     "value" : 136314880
   },
   "jvm.memory.non-heap.usage" : {
     "value" : 0.16856530996469352
   },
   "jvm.memory.non-heap.used" : {
     "value" : 22978104
   },
   "jvm.memory.pools.CMS-Old-Gen.usage" : {
     "value" : 0.05705025643464222
```

```
    },
    "jvm.memory.pools.CMS-Perm-Gen.usage" : {
      "value" : 0.25629341311571074
    },
    "jvm.memory.pools.Code-Cache.usage" : {
      "value" : 0.018703460693359375
    },
    "jvm.memory.pools.Par-Eden-Space.usage" : {
      "value" : 0.5095581972509399
    },
    "jvm.memory.pools.Par-Survivor-Space.usage" : {
      "value" : 0.9115804036458334
    },
    "jvm.memory.total.committed" : {
      "value" : 109711360
    },
    "jvm.memory.total.init" : {
      "value" : 24317952
    },
    "jvm.memory.total.max" : {
      "value" : 266272768
    },
    "jvm.memory.total.used" : {
      "value" : 40248344
    },
    "jvm.threads.blocked.count" : {
      "value" : 2
    },
    "jvm.threads.count" : {
      "value" : 22
    },
    "jvm.threads.daemon.count" : {
      "value" : 4
    },
    "jvm.threads.deadlocks" : {
      "value" : [ ]
    },
    "jvm.threads.new.count" : {
      "value" : 0
    },
    "jvm.threads.runnable.count" : {
      "value" : 10
    },
    "jvm.threads.terminated.count" : {
      "value" : 0
    },
    "jvm.threads.timed_waiting.count" : {
      "value" : 8
    },
    "jvm.threads.waiting.count" : {
      "value" : 2
    }
  },
  "counters" : {
    "myMetrics.myCounter" : {
      "count" : 1
    }
  },
  "histograms" : {
    "myMetrics.myHistogram" : {
      "count" : 1,
      "max" : 100,
      "mean" : 100.0,
      "min" : 100,
```

```
      "p50" : 100.0,
      "p75" : 100.0,
      "p95" : 100.0,
      "p98" : 100.0,
      "p99" : 100.0,
      "p999" : 100.0,
      "stddev" : 0.0
    }
  },
  "meters" : {
    "morphline.java.numNotifyCalls" : {
      "count" : 1,
      "m15_rate" : 0.16929634497812282,
      "m1_rate" : 0.19779007785878447,
      "m5_rate" : 0.1934432200964012,
      "mean_rate" : 0.06666243138019297,
      "units" : "events/second"
    },
    "morphline.java.numProcessCalls" : {
      "count" : 1,
      "m15_rate" : 0.16929634497812282,
      "m1_rate" : 0.19779007785878447,
      "m5_rate" : 0.1934432200964012,
      "mean_rate" : 0.06666191145031655,
      "units" : "events/second"
    },
    "morphline.logDebug.numNotifyCalls" : {
      "count" : 3,
      "m15_rate" : 0.5078890349343685,
      "m1_rate" : 0.5933702335763534,
      "m5_rate" : 0.5803296602892035,
      "mean_rate" : 0.1999690981087056,
      "units" : "events/second"
    },
    "morphline.logDebug.numProcessCalls" : {
      "count" : 3,
      "m15_rate" : 0.5078890349343685,
      "m1_rate" : 0.5933702335763534,
      "m5_rate" : 0.5803296602892035,
      "mean_rate" : 0.19996765856402157,
      "units" : "events/second"
    },
    "morphline.logWarn.numNotifyCalls" : {
      "count" : 2,
      "m15_rate" : 0.33859268995624564,
      "m1_rate" : 0.39558015571756894,
      "m5_rate" : 0.3868864401928024,
      "mean_rate" : 0.11979779569659962,
      "units" : "events/second"
    },
    "morphline.logWarn.numProcessCalls" : {
      "count" : 2,
      "m15_rate" : 0.33859268995624564,
      "m1_rate" : 0.39558015571756894,
      "m5_rate" : 0.3868864401928024,
      "mean_rate" : 0.11979702071997292,
      "units" : "events/second"
    },
    "morphline.pipe.numNotifyCalls" : {
      "count" : 1,
      "m15_rate" : 0.16929634497812282,
      "m1_rate" : 0.19779007785878447,
      "m5_rate" : 0.1934432200964012,
      "mean_rate" : 0.05774150456461029,
```

```
      "units" : "events/second"
    },
    "morphline.pipe.numProcessCalls" : {
      "count" : 1,
      "m15_rate" : 0.16929634497812282,
      "m1_rate" : 0.19779007785878447,
      "m5_rate" : 0.1934432200964012,
      "mean_rate" : 0.05766282424671017,
      "units" : "events/second"
    },
    "morphline.registerJVMMetrics.numNotifyCalls" : {
      "count" : 1,
      "m15_rate" : 0.16929634497812282,
      "m1_rate" : 0.19779007785878447,
      "m5_rate" : 0.1934432200964012,
      "mean_rate" : 0.059902898599428295,
      "units" : "events/second"
    },
    "morphline.registerJVMMetrics.numProcessCalls" : {
      "count" : 1,
      "m15_rate" : 0.16929634497812282,
      "m1_rate" : 0.19779007785878447,
      "m5_rate" : 0.1934432200964012,
      "mean_rate" : 0.059902518235973874,
      "units" : "events/second"
    },
    "morphline.startReportingMetricsToHTTP.numNotifyCalls" : {
      "count" : 3,
      "m15_rate" : 0.5078890349343685,
      "m1_rate" : 0.5933702335763534,
      "m5_rate" : 0.5803296602892035,
      "mean_rate" : 0.19066147711547488,
      "units" : "events/second"
    },
    "morphline.startReportingMetricsToHTTP.numProcessCalls" : {
      "count" : 3,
      "m15_rate" : 0.5078890349343685,
      "m1_rate" : 0.5933702335763534,
      "m5_rate" : 0.5803296602892035,
      "mean_rate" : 0.19066014422550162,
      "units" : "events/second"
    },
    "myMetrics.myMeter" : {
      "count" : 1,
      "m15_rate" : 0.18400888292586468,
      "m1_rate" : 0.19889196960097935,
      "m5_rate" : 0.1966942907643235,
      "mean_rate" : 0.0698670918312095,
      "units" : "events/second"
    }
  },
  "timers" : {
    "myMetrics.myTimer" : {
      "count" : 1,
      "max" : 1.4000000000000001E-5,
      "mean" : 1.4000000000000001E-5,
      "min" : 1.4000000000000001E-5,
      "p50" : 1.4000000000000001E-5,
      "p75" : 1.4000000000000001E-5,
      "p95" : 1.4000000000000001E-5,
      "p98" : 1.4000000000000001E-5,
      "p99" : 1.4000000000000001E-5,
      "p999" : 1.4000000000000001E-5,
      "stddev" : 0.0,
```

```
      "m15_rate" : 0.18400888292586468,
      "m1_rate" : 0.19889196960097935,
      "m5_rate" : 0.1966942907643235,
      "mean_rate" : 0.0698417274708746,
      "duration_units" : "seconds",
      "rate_units" : "calls/second"
    },
    "myMetrics.myTimer2" : {
      "count" : 1,
      "max" : 0.0,
      "mean" : 0.0,
      "min" : 0.0,
      "p50" : 0.0,
      "p75" : 0.0,
      "p95" : 0.0,
      "p98" : 0.0,
      "p99" : 0.0,
      "p999" : 0.0,
      "stddev" : 0.0,
      "m15_rate" : 0.18400888292586468,
      "m1_rate" : 0.19889196960097935,
      "m5_rate" : 0.1966942907643235,
      "mean_rate" : 0.0698418152725891,
      "duration_units" : "seconds",
      "rate_units" : "calls/second"
    }
  }
}
```

Example startReportingMetricsToHTTP thread dump output:

Here is the response to an HTTP GET to http://localhost:8080/threads for a thread dump:

```
main id=1 state=TIMED_WAITING
    at java.lang.Thread.sleep(Native Method)
    at org.kitesdk.morphline.metrics.servlets.HttpMetricsMorphlineTest.te
stBasic(HttpMetricsMorphlineTest.java:51)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl
.java:39)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAcc
essorImpl.java:25)
    at java.lang.reflect.Method.invoke(Method.java:597)
    at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(Framew
orkMethod.java:45)
    at org.junit.internal.runners.model.ReflectiveCallable.run(Reflective
Callable.java:15)
    at org.junit.runners.model.FrameworkMethod.invokeExplosively(Framework
Method.java:42)
    at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMet
hod.java:20)
    at org.junit.internal.runners.statements.RunBefores.evaluate(RunBefores
.java:28)
    at org.junit.internal.runners.statements.RunAfters.evaluate(RunAfters.
java:30)
    at org.junit.runners.ParentRunner.runLeaf(ParentRunner.java:263)
    at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassR
unner.java:68)
    at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4Class
Runner.java:47)
    at org.junit.runners.ParentRunner$3.run(ParentRunner.java:231)
    at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:60)
    at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:229)
```

```
    at org.junit.runners.ParentRunner.access$000(ParentRunner.java:50)
    at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:222)
    at org.junit.runners.ParentRunner.run(ParentRunner.java:300)
    at org.eclipse.jdt.internal.junit4.runner.JUnit4TestReference.run(JUn
it4TestReference.java:50)
    at org.eclipse.jdt.internal.junit.runner.TestExecution.run(TestExecuti
on.java:38)
    at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(Remot
eTestRunner.java:467)
    at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(Rem
oteTestRunner.java:683)
    at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.run(RemoteTest
Runner.java:390)
    at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.main(RemoteTe
stRunner.java:197)

Reference Handler id=2 state=WAITING
    - waiting on <0x7418e252> (a java.lang.ref.Reference$Lock)
    - locked <0x7418e252> (a java.lang.ref.Reference$Lock)
    at java.lang.Object.wait(Native Method)
    at java.lang.Object.wait(Object.java:485)
    at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:116)

... and so on
```

kite-morphlines-protobuf

This maven module contains morphline commands for reading, extracting, and transforming protocol buffer objects.

readProtobuf

The readProtobuf command (source code) parses an InputStream or byte array that contains protobuf data. For each protobuf object, the command emits a morphline record containing the top level object as an attachment in the field _attachment_body.

The input stream or byte array is read from the first attachment of the input record.

The command provides the following configuration options:

Property Name	Default	Description
protobufClass	[]	The fully qualified name of a Java class that was generated by the protoc compiler. This Java class contains protobuf message definitions.
outputClass	[]	The name of an inner Java class (within protobufClass) for deserializing data to.

Example usage:

```
readProtobuf {
  protobufClass : org.kitesdk.morphline.protobuf.Protos
  outputClass : RepeatedLongs
}
```

And protobuf schema for protoc:

```
option java_package = "org.kitesdk.morphline.protobuf";
option java_outer_classname = "Protos";
option java_generate_equals_and_hash = true;
```

```
option optimize_for = SPEED;

message RepeatedLongs {
  repeated sint64 longVal = 1;
}
message Complex {

  message Name {
    optional uint32 intVal = 1;
    optional uint64 longVal = 2;
    optional double doubleVal = 3;
    optional float floatVal = 4;
    repeated string stringVal = 5;
    optional RepeatedLongs repeatedLong = 6;
  }

  message Link {
    repeated string language = 1;
    required string url = 2;
  }

  enum Type {
    QUERY = 1;
    UPDATE = 2;
  }

  required sint32 docId = 1;
  required Name name = 2;
  repeated Link link = 3;
  required Type type = 4;
}
```

extractProtobufPaths

The extractProtobufPaths command (source code) extracts specific values from a protobuf object, akin to a simple form of XPath. The command uses zero or more path expressions to extract values from a protobuf instance object.

The protobuf input object is expected to be contained in the field _attachment_body, and typically placed there by an upstream readProtobuf command.

Each path expression consists of a record output field name (on the left side of the colon ':') as well as zero or more path steps (on the right hand side), each path step separated by a '/' slash, akin to a simple form of XPath. Repeated values(Lists) are traversed with the '[]' notation.

The result of a path expression is a list of objects, each of which is added to the given record output field. To check if the property is set and serialized in protobuf message is used the has<PropertyName>() method and if the property isn't set then there is no result of a path expression. That means the output field is not passed to next command.

The command provides the following configuration options:

Property Name	Default	Description
objectExtractMethod	toByteArray	Java method that is called on the protobuf object to get a value to pass to the next command if the type of value on a path is a protobuf object. Options are: toByteArray - the "toByteArray()" method is called to get serialized bytes from the protobuf object. toString - the "toString()" method is called to get a String representation of a protobuf object. none - no method is called and the whole protobuf object is passed to the next command.

Property Name	Default	Description
enumExtractMethod	name	Java method that is called to get a value to pass to the next command if the type of value on a path is an enum object. Options are: name - the "name()" method is called to get a String representation of the enum object. getNumber - the "getNumber()" method is called to get an int representation of the enum object, none - no method is called and the whole enum object is passed to the next command.
paths	[]	Zero or more protobuf path expressions.

Example usage:

```
extractProtobufPaths {
  objectExtractMethod : toByteArray
  enumExtractMethod : name
  paths : {
    "docId" : "/docId"
    "name" : "/name"
    "intVal" : "/name/intVal"
    "longVal" : "/name/longVal"
    "doubleVal" : "/name/doubleVal"
    "floatVal" : "/name/floatVal"
    "stringVals" : "/name/stringVal[]"
    "longVals" : "/name/repeatedLong/longVal[]"
    "links" : "/link[]"
    "languages" : "/link[]/language"
    "urls" : "/link[]/url"
    "type" : "/type"
  }
}
```

kite-morphlines-tika-core

This maven module contains morphline commands for autodetecting MIME types from binary data. Depends on tika-core.

detectMimeType

The detectMimeType command (source code) uses Apache Tika to autodetect the MIME type of the first attachment from the binary data. The detected MIME type is assigned to the _attachment_mimetype field.

The command provides the following configuration options:

Property Name	Default	Description
includeDefaultMimeTypes	true	Whether to include the Tika default MIME types file that ships embedded in tika-core.jar (see http://github.com/apache/tika/blob/trunk/tika-core/src/main/resources/org/apache/tika/mime/tika-mimetypes.xml)
mimeTypesFiles	[]	The relative or absolute path of zero or more Tika custom-mimetypes.xml files to include.
mimeTypesString	null	The content of an optional custom-mimetypes.xml file embedded directly inside of this morphline configuration file.

Property Name	Default	Description
preserveExisting	true	Whether to preserve the _attachment_mimetype field value if one is already present.
includeMetaData	false	Whether to pass the record fields to Tika to assist in MIME type detection.
excludeParameters	true	Whether to remove MIME parameters from output MIME type.

Example usage:

```
detectMimeType {
  includeDefaultMimeTypes : false
  #mimeTypesFiles : [src/test/resources/custom-mimetypes.xml]
  mimeTypesString :
    """
    <mime-info>
      <mime-type type="text/space-separated-values">
        <glob pattern="*.ssv"/>
      </mime-type>

      <mime-type type="avro/binary">
        <magic priority="50">
          <match value="0x4f626a01" type="string" offset="0"/>
        </magic>
        <glob pattern="*.avro"/>
      </mime-type>

      <mime-type type="mytwittertest/json+delimited+length">
        <magic priority="50">
          <match value="[0-9]+(\r)?\n\\{&quot;" type="regex" offset="0:16"/>
        </magic>
      </mime-type>
    </mime-info>
    """
}
```

kite-morphlines-tika-decompress

This maven module contains morphline commands for decompressing and unpacking files. Depends on tika-core and commons-compress.

decompress

The decompress command (source code) decompresses the first attachment, and supports gzip and bzip2 format.

Example usage:

```
decompress {}
```

unpack

The unpack command (source code) unpacks the first attachment, and supports tar, zip, and jar format. The command emits one record per contained file.

Example usage:

```
unpack {}
```

kite-morphlines-saxon

This maven module contains morphline commands for reading, extracting and transforming XML and HTML with XPath, XQuery and XSLT.

convertHTML

The convertHTML command (source code) converts any HTML to XHTML, using the TagSoup Java library.

Instead of parsing well-formed or valid XML, this command parses HTML as it is found in the wild: poor, nasty and brutish, though quite often far from short. TagSoup (and hence this command) is designed for people who have to process this stuff using some semblance of a rational application design. By providing this converter, it allows standard XML tools to be applied to even the worst malformed HTML.

The command reads an InputStream or byte array from the first attachment (field _attachment_body) of the input record, parses it as HTML and replaces the field with UTF-8 encoded XHTML.

The command provides the following configuration options:

Property Name	Default	Description
supportedMimeTypes	null	Optionally, require the input record to match one of the MIME types in this list.
charset	null	The character encoding to use for parsing input, for example, UTF-8. If none is specified the charset specified in the _attachment_charset input field is used instead.
noNamespaces	true	A value of false indicates namespace URIs and unprefixed local names for element and attribute names will be available.
noCDATA	false	A value of true indicates that the parser will treat CDATA elements specially.
noBogons	false	A value of true indicates that the parser will ignore unknown elements.
emptyBogons	false	A value of true indicates that the parser will give unknown elements a content model of EMPTY; a value of false, a content model of ANY.
noRootBogons	false	A value of true indicates that the parser will allow unknown elements to be the root element.
noDefaultAttributes	false	A value of true indicates that the parser will return default attribute values for missing attributes that have default values.
noColons	false	A value of true indicates that the parser will translate colons into underscores in names.
noRestart	false	A value of true indicates that the parser will attempt to restart the restartable elements.
suppressIgnorableWhitespace	true	A value of false indicates that the parser will transmit whitespace in element-only content via the SAX ignorableWhitespace callback.

Example usage:

```
convertHTML {
  charset : UTF-8
}
```

xquery

The xquery command (source code) parses an InputStream that contains an XML document and runs the given W3C XQuery over the XML document, using the Saxon Java library. For each item in the query result sequence, the command emits a corresponding morphline record.

The command reads an InputStream or byte array from the first attachment (field _attachment_body) of the input record.

Per the W3C specs, every valid XPath (e.g. //tweets/tweet[@color='blue']) is also a valid XQuery. If you are comfortable with XPath you are already almost there.

An XQuery result sequence contains zero or more items such as element nodes, attribute nodes, text nodes, atomic values, etc. For each item in the query result sequence, the morphline command converts the item to a record and pipes that record to the next morphline command. For an attribute node the attribute's XPath string value is filled into the record field named after the attribute name. For an element node the attributes and children of the element are treated as follows: The XPath string value of the attribute or child is filled into the record field named after the child's name.

For example, in order to generate two morphline records, the first morphline record with a firstName field that contains Joe, as well as a lastName field that contains Bubblegum, and the second morphline record with a firs tName field that contains Alice, as well as a lastName field that contains Pellegrino, your xquery command should be formulated such that it outputs two XML fragments like this:

```
<record>
 <firstName>Joe</firstName>
 <lastName>Bubblegum</lastName>
</record>

<record>
 <firstName>Alice</firstName>
 <lastName>Pellegrino</lastName>
</record>
```

The xquery command provides the following configuration options:

Property Name	Default	Description
supportedMimeTypes	null	Optionally, require the input record to match one of the MIME types in this list.
languageVersion	"1.0"	Must be "1.0" for XQuery 1.0 or "3.0" for XQuery 3.0.
features	null	An optional JSON object containing zero or more name-value pairs that represent configuration properties for Saxon features.
extensionFunctions	[]	An optional list of Java class names that implement custom Saxon extension functions. Each such Java class must implement net.sf.saxon.s9api.ExtensionFunction as described in the Saxon documentation.
fragments	n/a	An array containing exactly one fragment JSON object, as described below.
Each fragment provides the following configuration options:		

Property Name	Default	Description
fragmentPath	n/a	Currently must be "/"
externalVariables	null	An optional JSON object containing zero or more name-value pairs that are bound and passed in as external variables to the query. Example: myVar : "hello world"
externalFileVariables	null	An optional JSON object containing zero or more name-path pairs that refer to XML files on the local file system, and are bound and passed in as external variables to the query. These files are loaded once on program startup and subsequently remain memory resident across queries. This can be used for efficient joins where the join table is static and fits into main memory. Example: myDoc : src/test/resources/testdocuments/helloworld.xml
queryFile	null	A relative or absolute path of a local file from which to load the query.
queryString	null	An inline string from which to load the query. One of queryFile or queryString must be present, but not both. Example: """/tweets/tweet"""

Note: The morphline configuration file is implemented using the HOCON format (Human Optimized Config Object Notation). HOCON is basically JSON slightly adjusted for the configuration file use case. HOCON syntax is defined at HOCON github page and as such, multi-line strings are similar to Python or Scala, using triple quotes. If the three-character sequence """ appears, then all Unicode characters until a closing """ sequence are used unmodified to create a string value.

Example usage:

```
xquery {
  fragments : [
    {
      fragmentPath : "/"
      externalVariables : {
        myVariable : "hello world"
      }
      queryString : """
        (: Example test xquery :)

        declare variable $myVariable as xs:string external;

        for $tweet in /tweets/tweet
        return
        <record>
          {$tweet/@id}
          {$tweet/user/@screen_name}
          <myStatusCounts>{string($tweet/user/@statuses_count)}</myStatusC
ounts>
          <text>{$tweet/text}</text>
          <greeting>{$myVariable}</greeting>
        </record>
      """
    }
  ]
}
```

Here is an example output record for the query above:

```
id:11111112
```

```
screen_name:fake_user1
myStatusCounts:11111
text:Come, see new hot tub under Redwood Tree!
greeting:hello world
```

More example usage:

```
xquery {
  fragments : [
    {
      fragmentPath : "/"
      queryString : """
        (: Example xquery :)
        for $req in /request
        return
          <record>
            <date> { string($req/data/agreementDate) } </date>
            <tradeId> { string($req/trade/@tradeId) } </tradeId>

            <partyId>
              {
                for $keyword in $req/trade/keyWords/keyword
                  where $keyword/name = "memberId"
                    return string($keyword/value)
              }
            </partyId>
            <fullText> { $req } </fullText>
        </record>
      """
    }
  ]
}
```

More examples can be found in the unit tests.

Here is an example extension function along with a corresponding example xquery.

For more background, see resources such as the XQuery Primer and XQuery FLOWR Tutorial and XQuery: A Guided Tour and Wikipedia.

xslt

The xslt command (source code) parses an InputStream that contains an XML document and runs the given W3C XSL Transform over the XML document, using the Saxon Java library. For each item in the query result sequence, the command emits a corresponding morphline record.

The command reads an InputStream or byte array from the first attachment (field _attachment_body) of the input record.

An XSLT result sequence contains zero or more items such as element nodes, attribute nodes, text nodes, atomic values, etc. For each item in the query result sequence, the morphline command converts the item to a record and pipes that record to the next morphline command. For an attribute node the attribute's XPath string value is filled into the record field named after the attribute name. For an element node the attributes and children of the element are treated as follows: The XPath string value of the attribute or child is filled into the record field named after the child's name.

For example, in order to generate two morphline records, the first morphline record with a firstName field that contains Joe, as well as a lastName field that contains Bubblegum, and the second morphline record with a firs tName field that contains Alice, as well as a lastName field that contains Pellegrino, your xslt command should be formulated such that it outputs two XML fragments like this:

```
<record>
```

```
 <firstName>Joe</firstName>
 <lastName>Bubblegum</lastName>
</record>

<record>
 <firstName>Alice</firstName>
 <lastName>Pellegrino</lastName>
</record>
```

The command provides the following configuration options:

Property Name	Default	Description
supportedMimeTypes	null	Optionally, require the input record to match one of the MIME types in this list.
features	null	An optional JSON object containing zero or more name-value pairs that represent configuration properties for Saxon features.
extensionFunctions	[]	An optional list of Java class names that implement custom Saxon extension functions. Each such Java class must implement net.sf.saxon.s9api.ExtensionFunction as described in the Saxon documentation.
fragments	n/a	An array containing exactly one fragment JSON object, as described below.
Each fragment provides the following configuration options:		
fragmentPath	n/a	Currently must be "/"
parameters	null	An optional JSON object containing zero or more name-value pairs that are bound and passed in as XSLT parameters to the query. Example: myVar : "hello world"
fileParameters	null	An optional JSON object containing zero or more name-path pairs that refer to XML files on the local file system, and are bound and passed in as external variables to the query. These files are loaded once on program startup and subsequently remain memory resident across queries. This can be used for efficient joins where the join table is static and fits into main memory. Example: myDoc : src/test/resources/testdocuments/helloworld.xml
queryFile	null	A relative or absolute path of a local file from which to load the query.
queryString	null	An inline string from which to load the query. One of queryFile or queryString must be present, but not both.

Note: The morphline configuration file is implemented using the HOCON format (Human Optimized Config Object Notation). HOCON is basically JSON slightly adjusted for the configuration file use case. HOCON syntax is defined at HOCON github page and as such, multi-line strings are similar to Python or Scala, using triple quotes. If the three-character sequence """ appears, then all Unicode characters until a closing """ sequence are used unmodified to create a string value.

Example usage:

```
xslt {
  fragments : [
    {
      fragmentPath : "/"
      parameters : {
```

```
        myVariable : "hello world"
      }
      queryString : """
        <!-- Example XSLT identity transformation -->
        <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" ver
sion="1.0">

        <xsl:template match="@*|node()">
          <xsl:copy>
            <xsl:apply-templates select="@*|node()"/>
          </xsl:copy>
        </xsl:template>

        </xsl:stylesheet>
      """
    }
  ]
}
```

More examples can be found in the unit tests.

For more background, see resources such as the XSLT Tutorial and Wikipedia.

kite-morphlines-solr-core

This maven module contains morphline commands for Solr that higher level modules such as kite-morphlines-solr-cell, search-mr, and search-flume depend on for indexing.

solrLocator

A solrLocator is a set of configuration parameters that identify the location and schema of a Solr server or SolrCloud. Based on this information a morphline Solr command can fetch the Solr index schema and send data to Solr. A solr Locator is not actually a command but rather a common parameter of many morphline Solr commands, and thus described separately here.

Example usage:

```
solrLocator : {
  # Name of solr collection
  collection : collection1

  # ZooKeeper ensemble
  zkHost : "127.0.0.1:2181/solr"
  # Max number of documents to pass per RPC from morphline to Solr Server
  # batchSize : 10000
}
```

loadSolr

The loadSolr command (source code) inserts, updates or deletes records into a Solr server or MapReduce Reducer.

The command provides the following configuration options:

Property Name	Default	Description
solrLocator	n/a	Solr location parameters as described separately above.

Property Name	Default	Description
boosts	[]	An optional JSON object containing zero or more fieldName-boostValue mappings where the fieldName is a String and the boostValue is a float. The default boost is 1.0.

Examples:

- loadSolrUpdate
- loadSolrPartialUpdate
- loadSolrDeleteById
- loadSolrDeleteByQuery
- loadSolrChildDocuments

Example loadSolr usage to insert a document or update an existing document stored in Solr ("update")

```
loadSolr {
  solrLocator : {
    # Name of solr collection
    collection : collection1
    # ZooKeeper ensemble
    zkHost : "127.0.0.1:2181/solr"

    # Max number of docs to pass per RPC from morphline to Solr Server
    # batchSize : 10000
  }
  boosts : {
    id : 2.0 # assign to the id field a boost value 2.0
  }
}
```

Example loadSolr usage to update a subset of fields of an existing document stored in Solr ("partial document update"):

```
java { code : """
  // specify the unique key of the document stored in Solr that shall be
updated
  record.put("id", 123);

  // set "first_name" field of stored Solr document to "Nadja"; retain ot
her fields as-is
  record.put("first_name", Collections.singletonMap("set", "Nadja"));

  // set "tags" field of stored Solr document to multiple values ["smart", "
creative"]; retain other fields as-is
  record.put("tags", Collections.singletonMap("set", Arrays.asList("smart",
 "creative")));

  // add "San Francisco" to the existing values of the cities field of the
 stored Solr document; retain other fields as-is
  record.put("cities", Collections.singletonMap("add", "San Francisco"));

  // remove the "text" field from a document stored in Solr; retain other
fields as-is
  record.put("text", Collections.singletonMap("set", null));

  // increment user_friends_count by 5; retain other fields of stored Solr
document as-is
  record.put("user_friends_count", Collections.singletonMap("inc", 5));

  // pass record to next command in chain
```

```
  return child.process(record);
              """
}

loadSolr {
  <solrLocator goes here>
}
```

Note:

A partial document update requires that all Solr fields be configured as stored="true" in schema.xml, not just the field that shall be updated. This is because this kind of "update" is implemented as a document delete followed by a document insert: the old values of all fields are fetched from Solr (requires stored="true" markers), then the old document is deleted and then a new document is inserted with the old values plus the new field value. For details see this article and http://stackoverflow.com/questions/12183798/solrj-api-for-partial-document-update

Example loadSolr usage for deleteById:

```
# Tell loadSolr command to delete the documents for which the unique key fie
ld equals 123 or 456.
setValues {
  _loadSolr_deleteById:[123, 456]
}

loadSolr {
  <solrLocator goes here>
}
```

Example loadSolr usage for deleteByQuery:

```
# Tell loadSolr command to delete all documents for which the following cond
itions hold:
# The city field starts with "Paris" AND the color field equals "blue" OR
# The city field starts with "London" AND the color field equals "purple"
setValues {
  _loadSolr_deleteByQuery:["(city:Paris*)AND(color:blue)", "(city:London*)
AND(color:purple)"]
}

loadSolr {
  <solrLocator goes here>
}
```

Example loadSolr usage for child documents (aka nested documents):

A record can contain (arbitrarily nested) child documents (aka nested documents aka nested records) in the "_loadSolr_childDocuments" morphline record field. If present, these are recognized and indexed by the loadSolr command, and the parent-child relationships become available to Solr queries, as shown below:

```
java {
  code: """
    // Index a document that has a foo child document, which in turn has a b
ar child document
    record.put("id", "12345");
    record.put("content_type", "parent");
    Record childDoc = new Record();
    childDoc.put("id", "foo");
    childDoc.put("content_type", "child");
    Record childDoc2 = new Record();
    childDoc2.put("id", "bar");
    childDoc2.put("content_type", "child");
```

```
    childDoc.put("_loadSolr_childDocuments", childDoc2); // mark as child
doc
    record.put("_loadSolr_childDocuments", childDoc); // mark as child doc
    return child.process(record);
        """
  }
}
loadSolr {
  <solrLocator goes here>
}
```

Example Solr parent block join that returns the parent records for records where the child documents contain "bar" in the id field:

```
{!parent which='content_type:parent'}id:bar
```

For more background see this article.

generateSolrSequenceKey

The generateSolrSequenceKey command (source code) assigns a record unique key that is the concatenation of the given baseIdField record field, followed by a running count of the record number within the current session. The count is reset to zero whenever a startSession notification is received.

For example, assume a CSV file containing multiple records but no unique ids, and the base_id field is the filesystem path of the file. Now this command can be used to assign the following record values to Solr's unique key field: $path#0, $path#1, ... $path#N.

The name of the unique key field is fetched from Solr's schema.xml file, as directed by the solrLocator configuration parameter.

The command provides the following configuration options:

Property Name	Default	Description
solrLocator	n/a	Solr location parameters as described separately above.
baseIdField	baseid	The name of the input field to use for prefixing keys.
preserveExisting	true	Whether to preserve the field value if one is already present.solrLocator n/a Solr location parameters as described separately above. baseIdField baseid The name of the input field to use for prefixing keys. preserveExisting true Whether to preserve the field value if one is already present.

Example usage:

```
generateSolrSequenceKey {
  baseIdField: ignored_base_id
  solrLocator : ${SOLR_LOCATOR}
}
```

sanitizeUnknownSolrFields

The sanitizeUnknownSolrFields command (source code) sanitizes record fields that are unknown to Solr schema.xml by either deleting them (renameToPrefix parameter is absent or a zero length string) or by moving them to a field prefixed with the given renameToPrefix (for example, to use typical dynamic Solr fields).

Recall that Solr throws an exception on any attempt to load a document that contains a field that is not specified in schema.xml.

The command provides the following configuration options:

Property Name	Default	Description
solrLocator	n/a	Solr location parameters as described separately above.
renameToPrefix	""	Output field prefix for unknown fields.

Example usage:

```
sanitizeUnknownSolrFields {
  solrLocator : ${SOLR_LOCATOR}
}
```

tokenizeText

The tokenizeText command (source code) uses the embedded Solr/Lucene Analyzer library to generate tokens from a text string, without sending data to a Solr server.

This is useful for prototyping and debugging Solr applications. It is also useful for standalone usage outside of Solr, e.g. for extracting text features from documents for use with recommendation systems, clustering and classification applications.

The command provides the following configuration options:

Property Name	Default	Description
solrLocator	n/a	Solr location parameters as described separately above.
inputField	n/a	The name of the input field.
outputField	n/a	The name of the field to add output values to.
solrFieldType	n/a	The name of the Solr field type in schema.xml to use for text analysis and tokenization. This parameter specifies the algorithmic extraction rules. Example: "text_en"

Example usage:

```
tokenizeText {
  inputField : message
  outputField : tokens
  solrFieldType : text_en
  solrLocator : {
    # Name of solr collection
    collection : collection1
    # ZooKeeper ensemble
    zkHost : "127.0.0.1:2181/solr"

    # solrHomeDir : "example/solr/collection1"
  }
}
```

For example, given the input field message with the value Hello World!\nFoo@Bar.com #%()123 the expected output record is:

```
tokens:hello
tokens:world
tokens:foo
```

```
tokens:bar.com
tokens:123
```

This example assumes the Solr field type named "text_en" is defined in schema.xml as shown in the following snippet:

```
...
<fieldType name="text_en" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory"
       ignoreCase="true"
       words="lang/stopwords_en.txt"
       enablePositionIncrements="true"
    />
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.tx
t"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
</fieldType>
```

kite-morphlines-solr-cell

This maven module contains morphline commands for using SolrCell with Tika parsers. This includes support for types including HTML, XML, PDF, Word, Excel, Images, Audio, and Video.

solrCell

The solrCell command (source code) pipes the first attachment of a record into one of the given Apache Tika parsers, then maps the Tika output back to a record using Apache SolrCell.

The Tika parser is chosen from the configurable list of parsers, depending on the MIME type specified in the input record. Typically, this requires an upstream detectMimeType command.

The command provides the following configuration options:

Property Name	Default	Description
solrLocator	n/a	Solr location parameters as described separately above.
capture	[]	List of XHTML element names to extract from the Tika output. For instance, it could be used to grab paragraphs (<p>) and index them into a separate field. Note that content is also still captured into the overall "content" field.
fmaps	[]	Maps (moves) one field name to another. See the example below.

Property Name	Default	Description
uprefix	null	The uprefix option indicates that the command shall prefix all fields that are not defined in the Solr schema.xml with the given prefix. Recall that Solr throws an exception on any attempt to load a document that contains a field that is not specified in schema.xml. The uprefix option is very useful when combined with dynamic field definitions. For example, uprefix : ignored_ would effectively ignore all unknown fields generated by Tika if the schema.xml contains the following dynamic field definition: dynamicField name="ignored_*" type="ignored"
captureAttr	false	Whether to index attributes of the Tika XHTML elements into separate fields, named after the element. For example, when extracting from HTML, Tika can return the href attributes in <a> tags as fields named "a".
xpath	null	When extracting, only return Tika XHTML content that satisfies the XPath expression. See http://tika.apache.org/1.4/parser.html for details on the format of Tika XHTML. See also http://wiki.apache.org/solr/TikaExtractOnlyExampleOutput.
lowernames	false	Map all field names to lowercase with underscores. For example, Content-Type would be mapped to content_type.
solrContentHandlerFactory	org.kitesdk.morphline.solrcell.TrimSolrContentHandlerFactory	A Java class to handle bridging from Tika to SolrCell.
parsers	[]	List of fully qualified Java class names of one or more Tika parsers.

Example usage:

```
solrCell {
  solrLocator : ${SOLR_LOCATOR}

  # extract some fields
  capture : [content, a, h1, h2]

  # rename exif_image_height field to text field
  # rename a field to anchor field
  # rename h1 field to heading1 field
  fmap : { exif_image_height : text, a : anchor, h1 : heading1 }

  # xpath : "/xhtml:html/xhtml:body/xhtml:div/descendant:node()"

  parsers : [ # one or more nested Tika parsers
    { parser : org.apache.tika.parser.jpeg.JpegParser }
  ]
}
```

Here is a complex morphline that demonstrates integrating multiple heterogenous input file formats via a tryR ules command, including Avro and SolrCell, using auto detection of MIME types via detectMimeType command, recursion via the callParentPipe command for unwrapping container formats, and automatic UUID generation:

```
morphlines : [
  {
    id : morphline1
    importCommands : ["org.kitesdk.**", "org.apache.solr.**"]
```

```
    commands : [
      {
        # emit one output record for each attachment in the input
        # record's list of attachments. The result is a list of
        # records, each of which has at most one attachment.
        separateAttachments {}
      }

      {
        # used for auto-detection if MIME type isn't explicitly supplied
        detectMimeType {
          includeDefaultMimeTypes : true
          mimeTypesFiles : [target/test-classes/custom-mimetypes.xml]
        }
      }

      {
        tryRules {
          throwExceptionIfAllRulesFailed : true
          rules : [
            # next rule of tryRules cmd:
            {
              commands : [
                { logDebug { format : "hello unpack" } }
                { unpack {} }
                { generateUUID {} }
                { callParentPipe {} }
              ]
            }

            # next rule of tryRules cmd:
            {
              commands : [
                { logDebug { format : "hello decompress" } }
                { decompress {} }
                { callParentPipe {} }
              ]
            }
            # next rule of tryRules cmd:
            {
              commands : [
                {
                  readAvroContainer {
                    supportedMimeTypes : [avro/binary]
                    # optional, avro json schema blurb for getSchema()
                    # readerSchemaString : "<json can go here>"
                    # readerSchemaFile : /path/to/syslog.avsc
                  }
                }

                { extractAvroTree {} }

                {
                  setValues {
                    id : "@{/id}"
                    user_screen_name : "@{/user_screen_name}"
                    text : "@{/text}"
                  }
                }

                {
                  sanitizeUnknownSolrFields {
                    solrLocator : ${SOLR_LOCATOR}
```

```
                }
              }
            ]
          }

          # next rule of tryRules cmd:
          {
            commands : [
              {
                readJsonTestTweets {
                  supportedMimeTypes : ["mytwittertest/json+delimited+len
gth"]
                }
              }
              {
                sanitizeUnknownSolrFields {
                  solrLocator : ${SOLR_LOCATOR}
                }
              }
            ]
          }

          # next rule of tryRules cmd:
          {
            commands : [
              { logDebug { format : "hello solrcell" } }
              {
                # wrap SolrCell around an Tika parsers
                solrCell {
                  solrLocator : ${SOLR_LOCATOR}

                  capture : [
                    # twitter feed schema
                    user_friends_count
                    user_location
                    user_description
                    user_statuses_count
                    user_followers_count
                    user_name
                    user_screen_name
                    created_at
                    text
                    retweet_count
                    retweeted
                    in_reply_to_user_id
                    source
                    in_reply_to_status_id
                    media_url_https
                    expanded_url
                   ]

                  # rename "content" field to "text" fields
                  fmap : { content : text, content-type : content_type }

                  lowernames : true
                  # Tika parsers to be registered:
                  parsers : [
                    # { parser : org.apache.tika.parser.AutoDetectParser }
                    { parser : org.apache.tika.parser.asm.ClassParser }
                    { parser : org.gagravarr.tika.FlacParser }
                    { parser : org.apache.tika.parser.audio.AudioParser }
                    { parser : org.apache.tika.parser.audio.MidiParser }
                    { parser : org.apache.tika.parser.crypto.Pkcs7Parser }
                    { parser : org.apache.tika.parser.dwg.DWGParser }
```

```
                        { parser : org.apache.tika.parser.epub.EpubParser }
                        { parser : org.apache.tika.parser.executable.Executabl
eParser }
                        { parser : org.apache.tika.parser.feed.FeedParser }
                        { parser : org.apache.tika.parser.font.AdobeFontMetric
Parser }
                        { parser : org.apache.tika.parser.font.TrueTypeParser
 }
                        { parser : org.apache.tika.parser.xml.XMLParser }
                        { parser : org.apache.tika.parser.html.HtmlParser }
                        { parser : org.apache.tika.parser.image.ImageParser }
                        { parser : org.apache.tika.parser.image.PSDParser }
                        { parser : org.apache.tika.parser.image.TiffParser }
                        { parser : org.apache.tika.parser.iptc.IptcAnpaParse
r }
                        { parser : org.apache.tika.parser.iwork.IWorkPackagePa
rser }
                        { parser : org.apache.tika.parser.jpeg.JpegParser }
                        { parser : org.apache.tika.parser.mail.RFC822Parser }
                        { parser : org.apache.tika.parser.mbox.MboxParser,
                            additionalSupportedMimeTypes : [message/x-emlx] }
                        { parser : org.apache.tika.parser.microsoft.OfficePa
rser }
                        { parser : org.apache.tika.parser.microsoft.TNEFPar
ser }
                        { parser : org.apache.tika.parser.microsoft.ooxml.OO
XMLParser }
                        { parser : org.apache.tika.parser.mp3.Mp3Parser }
                        { parser : org.apache.tika.parser.mp4.MP4Parser }
                        { parser : org.apache.tika.parser.hdf.HDFParser }
                        { parser : org.apache.tika.parser.netcdf.NetCDFParser
}
                        { parser : org.apache.tika.parser.odf.OpenDocumentPa
rser }
                        { parser : org.apache.tika.parser.pdf.PDFParser }
                        { parser : org.apache.tika.parser.pkg.CompressorPar
ser }
                        { parser : org.apache.tika.parser.pkg.PackageParser }
                        { parser : org.apache.tika.parser.rtf.RTFParser }
                        { parser : org.apache.tika.parser.txt.TXTParser }
                        { parser : org.apache.tika.parser.video.FLVParser }
                        { parser : org.apache.tika.parser.xml.DcXMLParser }
                        { parser : org.apache.tika.parser.xml.FictionBookPar
ser }
                        { parser : org.apache.tika.parser.chm.ChmParser }
                      ]
                    }
                  }

                  { generateUUID { field : ignored_base_id } }
                  {
                    generateSolrSequenceKey {
                      baseIdField: ignored_base_id
                      solrLocator : ${SOLR_LOCATOR}
                    }
                  }
                ]
              }
            ]
          }
        }

        {
          loadSolr {
```

```
          solrLocator : ${SOLR_LOCATOR}
        }
      }

      {
        logDebug {
          format : "My output record: {}"
          args : ["@{}"]
        }
      }
    ]
  }
]
```

Note: More information on SolrCell can be found here: http://wiki.apache.org/solr/ExtractingRequestHandler

kite-morphlines-useragent

userAgent

The userAgent command (source code) parses a user agent string and returns structured higher level data like user agent family, operating system, version, and device type, using the underlying API and regexes.yaml BrowserScope database from ua-parser.

The command provides the following configuration options:

Property Name	Default	Description
inputField	n/a	The name of the input field that contains zero or more user agent strings.
outputFields	[]	A JSON object containing zero or more user agent mappings. Each mapping consists of a record output field name (on the left side of the colon ':') as well as an expression (on the right hand side). An expression consists of a concatenation of zero or more literal strings or components of the form @{componentName}. Example mapping: myOutputField : "@{ua_family}/@{ua_major}.@{ua_minor}.@{ua_patch}". The following components are available: ua_family, ua_major, ua_minor, ua_patch, os_family, os_major, os_minor, os_patch, os_patch_minor, device_family. If a component resolves to null or the empty string the preceding string separator, if any, is suppressed.
database	null	The (optional) relative or absolute path of a regexes.yaml database file on the local file system. The default is to use the standard regexes.yaml database file that ships embedded inside of the ua-parser-*.jar. Example: /path/to/regexes.yaml

Example usage:

```
userAgent {
  inputField : user_agents
  outputFields : {
    ua_family : "@{ua_family}"
    device_family : "@{device_family}"
```

```
    ua_family_and_version : "@{ua_family}/@{ua_major}.@{ua_minor}.@{ua_patch
}"
    os_family_and_version : "@{os_family} @{os_major}.@{os_minor}.@{os_pa
tch}"
  }
}
```

Example input:

```
user_agents : Mozilla/5.0 (iPhone; CPU iPhone OS 5_1_1 like Mac OS X) AppleW
ebKit/534.46 (KHTML, like Gecko) Version/5.1 Mobile/9B206 Safari/7534.48.3
```

Expected output:

```
ua_family : Mobile Safari
device_family : iPhone
ua_family_and_version : Mobile Safari/5.1
os_family_and_version : iOS 5.1.1
```

Cloudera Runtime 7.2.18

Cloudera Search Overview

Date published: 2015-05-05
Date modified: 2024-04-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

What is Cloudera Search....4

How Cloudera Search works....5

Cloudera Search and Cloudera....6

Cloudera Search and other Cloudera Runtime components.... 6

Cloudera Search architecture....8

Local file system support....11

Cloudera Search tasks and processes....13

Ingestion....13
Indexing....13
Querying....14

Backing up and restoring data using Cloudera Search....15

What is Cloudera Search

Learn about how Cloudera Search is different from Apache Solr and the added value it provides.

Cloudera Search is Apache Solr fully integrated in the Cloudera platform, taking advantage of the flexible, scalable, and robust storage system and data processing frameworks included in Cloudera. This eliminates the need to move large data sets across infrastructures to perform business tasks. It further enables a streamlined data pipeline, where search and text matching is part of a larger workflow.

Cloudera Search provides easy, natural language access to data stored in or ingested into Hadoop, HBase, or cloud storage. End users and other web services can use full-text queries and faceted drill-down to explore text, semi-structured, and structured data as well as quickly filter and aggregate it to gain business insight without requiring SQL or programming skills.

Using Cloudera Search with the Cloudera infrastructure provides:

- Simplified infrastructure
- Better production visibility and control
- Quicker insights across various data types
- Quicker problem resolution
- Simplified interaction and platform access for more users and use cases beyond SQL
- Scalability, flexibility, and reliability of search services on the same platform used to run other types of workloads on the same data
- A unified security model across all processes with access to your data
- Flexibility and scale in ingest and pre-processing options

The following table describes Cloudera Search features.

Table 1: Cloudera Search Features

Feature	Description
Unified management and monitoring with Cloudera Manager	Cloudera Manager provides unified and centralized management and monitoring for Cloudera Runtime and Cloudera Manager. Cloudera Manager simplifies deployment, configuration, and monitoring of your search services. Many existing search solutions lack management and monitoring capabilities and fail to provide deep insight into utilization, system health, trending, and other supportability aspects.
Simple cluster and collection management using the solrctl tool	Solrctl is a command line tool that allows convenient, centralized management of: • Solr instance configurations and schemas in ZooKeeper • Solr collection life cycle including low level control (Solr cores) • Collection snapshots, Backup and restore operations • SolrCloud cluster initialization and configuration
Index storage in HDFS	Cloudera Search is integrated with HDFS for robust, scalable, and self-healing index storage. Indexes created by Solr/Lucene are directly written in HDFS with the data, instead of to local disk, thereby providing fault tolerance and redundancy. Cloudera Search is optimized for fast read and write of indexes in HDFS while indexes are served and queried through standard Solr mechanisms. Because data and indexes are co-located, data processing does not require transport or separately managed storage.
Batch index creation through MapReduce	To facilitate index creation for large data sets, Cloudera Search has built-in MapReduce jobs for indexing data stored in HDFS or HBase. As a result, the linear scalability of MapReduce is applied to the indexing pipeline, off-loading Solr index serving resources.
Easy interaction and data exploration through Hue	A Cloudera Search GUI is provided as a Hue plug-in, enabling users to interactively query data, view result files, and do faceted exploration. Hue can also schedule standing queries and explore index files. This GUI uses the Cloudera Search API, which is based on the standard Solr API. The drag-and-drop dashboard interface makes it easy for anyone to create a Cloudera Search dashboard.

Feature	Description
Simplified data processing for Search workloads	Cloudera Search can use Apache Tika for parsing and preparation of many of the standard file formats for indexing. Additionally, Cloudera Search supports Avro, Hadoop Sequence, and Snappy file format mappings, as well as Log file formats, JSON, XML, and HTML. Cloudera Search also provides Morphlines, an easy-to-use, pre-built library of common data preprocessing functions. Morphlines simplifies data preparation for indexing over a variety of file formats. Users can easily implement Morphlines for Kafka, and HBase, or re-use the same Morphlines for other applications, such as MapReduce or Spark jobs.
HBase search	Cloudera Search integrates with HBase, enabling full-text search of HBase data without affecting HBase performance or duplicating data storage. A listener monitors the replication event stream from HBase RegionServers and captures each write or update-replicated event, enabling extraction and mapping (for example, using Morphlines). The event is then sent directly to Solr for indexing and storage in HDFS, using the same process as for other indexing workloads of Cloudera Search. The indexes can be served immediately, enabling free-text searching of HBase data.
Spark-Solr connector	You can use the Spark-Solr connector to index data into Solr in multiple ways, both batch and streaming. With the connector you get the benefits of Spark while you can access Solr easily and in a familiar way, from both Scala and Java.

How Cloudera Search works

Learn about the operation of Cloudera Search

In near real-time indexing use cases, such as log or event stream analytics, Cloudera Search indexes events that are streamed through Apache Kafka, Spark Streaming, or HBase. Fields and events are mapped to standard Solr indexable schemas. Lucene indexes the incoming events and the index is written and stored in standard Lucene index files in HDFS.

The indexes by default are loaded from HDFS to Solr cores, but Solr is also able to read from local disk. The difference in the design of Cloudera Search is the robust, distributed, and scalable storage layer of HDFS, which helps eliminate costly downtime and allows for flexibility across workloads without having to move data. Search queries can then be submitted to Solr through either the standard Solr API, or through a simple search GUI application, included in Cloudera Search, which can be deployed in Hue.

Cloudera Search batch-oriented indexing capabilities can address needs for searching across batch uploaded files or large data sets that are less frequently updated and less in need of near-real-time indexing. It can also be conveniently used for re-indexing (a common pain point in stand-alone Solr) or ad-hoc indexing for on-demand data exploration. Often, batch indexing is done on a regular basis (hourly, daily, weekly, and so on) as part of a larger workflow.

For such cases, Cloudera Search includes a highly scalable indexing workflow based on MapReduce or Spark. A MapReduce or Spark workflow is launched for specified files or folders in HDFS, or tables in HBase, and the field extraction and Solr schema mapping occurs during the mapping phase. Reducers use embedded Lucene to write the data as a single index or as index shards, depending on your configuration and preferences. After the indexes are stored, they can be queried by using standard Solr mechanisms, as previously described above for the near-real-time indexing use case. You can also configure these batch indexing options to post newly indexed data directly into live, active indexes, served by Solr. This GoLive option enables a streamlined data pipeline without interrupting service to process regularly incoming batch updates.

The Lily HBase Indexer Service is a flexible, scalable, fault tolerant, transactional, near real-time oriented system for processing a continuous stream of HBase cell updates into live search indexes. The Lily HBase Indexer uses Solr to index data stored in HBase. As HBase applies inserts, updates, and deletes to HBase table cells, the indexer keeps Solr consistent with the HBase table contents, using standard HBase replication features. The indexer supports flexible custom application-specific rules to extract, transform, and load HBase data into Solr. Solr search results can contain columnFamily:qualifier links back to the data stored in HBase. This way applications can use the Search result set to directly access matching raw HBase cells. Indexing and searching do not affect operational stability or write throughput of HBase because the indexing and searching processes are separate and asynchronous to HBase.

Cloudera Search and Cloudera

Discover how Search fits into Cloudera's offering.

Cloudera Search fits into the broader set of solutions available for analyzing information in large data sets. Cloudera provides both the means and the tools to store the data and run queries. You can explore data through:

- MapReduce or Spark jobs
- Impala queries
- Cloudera Search queries

Cloudera provides storage for and access to large data sets by using MapReduce jobs, but creating these jobs requires technical knowledge, and each job can take minutes or more to run. The longer run times associated with MapReduce jobs can interrupt the process of exploring data.

To provide more immediate queries and responses and to eliminate the need to write MapReduce applications, you can use Apache Impala. Impala returns results in seconds instead of minutes.

Although Impala is a fast, powerful application, it uses SQL-based querying syntax. Using Impala can be challenging for users who are not familiar with SQL. If you do not know SQL, you can use Cloudera Search. Although Impala, Apache Hive, and Apache Pig all require a structure that is applied at query time, Search supports free-text search on any data or fields you have indexed.

How Search uses existing infrastructure

Any data already in a Cloudera deployment can be indexed and made available for query by Cloudera Search. For data that is not stored in Cloudera, Cloudera Search provides tools for loading data into the existing infrastructure, and for indexing data as it is moved to HDFS or written to Apache HBase.

By leveraging existing infrastructure, Cloudera Search eliminates the need to create new, redundant structures. In addition, Cloudera Search uses services provided by Cloudera and Cloudera Manager in a way that does not interfere with other tasks running in the same environment. This way, you can reuse existing infrastructure without the cost and problems associated with running multiple services in the same set of systems.

Cloudera Search and other Cloudera Runtime components

Cloudera Search interacts with other Cloudera Runtime components to solve different problems. Learn about Cloudera Runtime components that contribute to the Search process and see how they interact with Cloudera Search.

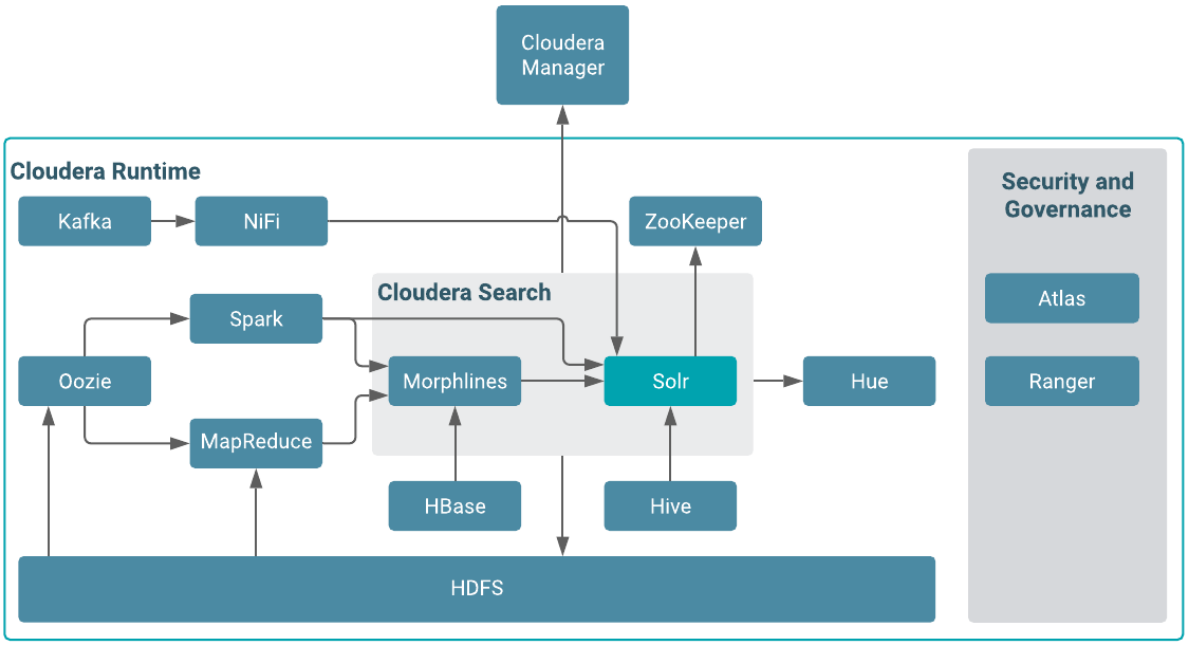

Component	Contribution	Applicable To
HDFS	Stores source documents. Search indexes source documents to make them searchable. Files that support Cloudera Search, such as Lucene index files and write-ahead logs, are also stored in HDFS. Using HDFS provides simpler provisioning on a larger base, redundancy, and fault tolerance. With HDFS, Cloudera Search servers are essentially stateless, so host failures have minimal consequences. HDFS also provides snapshotting, inter-cluster replication, and disaster recovery.	All cases
MapReduce	Cloudera Search includes a pre-built MapReduce-based job. This job can be used for on-demand or scheduled indexing of any supported data set stored in HDFS. This job uses cluster resources for scalable batch indexing.	Many cases
Hue	Hue includes a GUI-based Search application that uses standard Solr APIs and can interact with data indexed in HDFS. The application provides support for the Solr standard query language and visualization of faceted search functionality.	Many cases
Morphlines	A morphline is a rich configuration file that defines an ETL transformation chain. Morphlines can consume any kind of data from any data source, process the data, and load the results into Cloudera Search. Morphlines run in a small, embeddable Java runtime system, and can be used for near real-time applications such as the Lily HBase Indexer as well as batch processing applications such as a Spark job.	Many cases
ZooKeeper	Coordinates distribution of data and metadata, also known as shards. It provides automatic failover to increase service resiliency.	Many cases
Spark	The CrunchIndexerTool can use Spark to move data from HDFS files into Apache Solr, and run the data through a morphline for extraction and transformation. The Spark-Solr connector framework enables ETL of large datasets to Solr using spark-submit with a Spark job to batch index HDFS files into Solr.	Some cases
NiFi	Used for real time and often voluminous incoming event streams that need to be explorable (e.g. logs, twitter feeds, file appends etc).	
HBase	Supports indexing of stored data, extracting columns, column families, and key information as fields. Although HBase does not use secondary indexing, Cloudera Search can facilitate full-text searches of content in rows and tables in HBase.	Some cases
Cloudera Manager	Deploys, configures, manages, and monitors Cloudera Search processes and resource utilization across services on the cluster. Cloudera Manager helps simplify Cloudera Search administration.	Some cases
Atlas	Atlas classifications drive data access control through Ranger.	Some cases

Component	Contribution	Applicable To
Ranger	Enables role-based, fine-grained authorization for Cloudera Search. Ranger can apply restrictions to various actions, such as accessing data, managing configurations through config objects, or creating collections. Restrictions are consistently enforced, regardless of how users attempt to complete actions. Ranger offers both resource-based and tag-based access control policies.	Some cases
Oozie	Automates scheduling and management of indexing jobs. Oozie can check for new data and begin indexing jobs as required.	Some cases
Hive	Further analyzes search results.	Some cases
Kafka	Search uses this message broker project to increase throughput and decrease latency for handling real-time data.	Some cases
Sqoop	Ingests data in batch and enables data availability for batch indexing.	Some cases

Cloudera Search architecture

Cloudera Search runs as a distributed service on a set of servers, and each server is responsible for a portion of the searchable data. The data is split into smaller pieces, copies are made of these pieces, and the pieces are distributed among the servers. This provides two main advantages:

- Dividing the content into smaller pieces distributes the task of indexing the content among the servers.
- Duplicating the pieces of the whole allows queries to be scaled more effectively and enables the system to provide higher levels of availability.

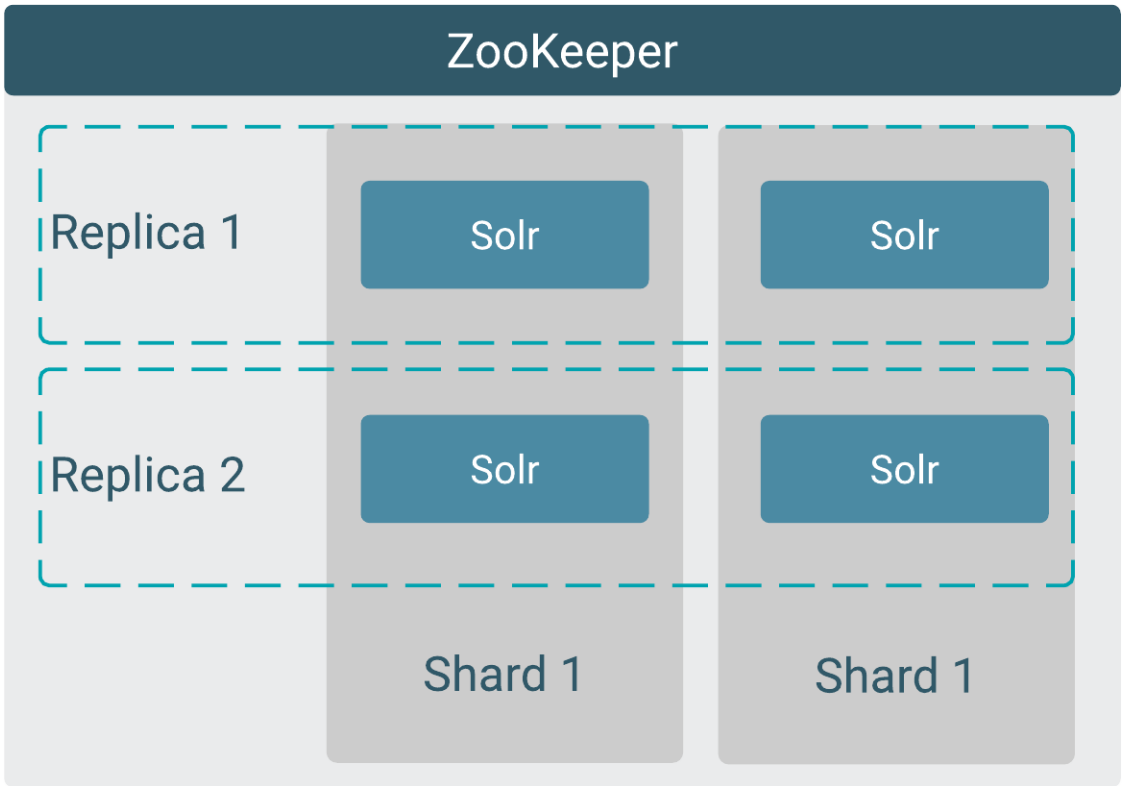

Each Cloudera Search server can handle requests independently. Clients can send requests to index documents or perform searches to any Search server, and that server routes the request to the correct server.

Each Search deployment requires:

- ZooKeeper on at least one host. You can install ZooKeeper, Search, and HDFS on the same host.
- HDFS on at least one, but as many as all hosts. HDFS is commonly installed on all cluster hosts.
- Solr on at least one but as many as all hosts. Solr is commonly installed on all cluster hosts.

More hosts with Solr and HDFS provides the following benefits:

- More Search servers processing requests.
- More Search and HDFS co-location increasing the degree of data locality. More local data provides faster performance and reduces network traffic.

The following graphic illustrates some of the key elements in a typical deployment.

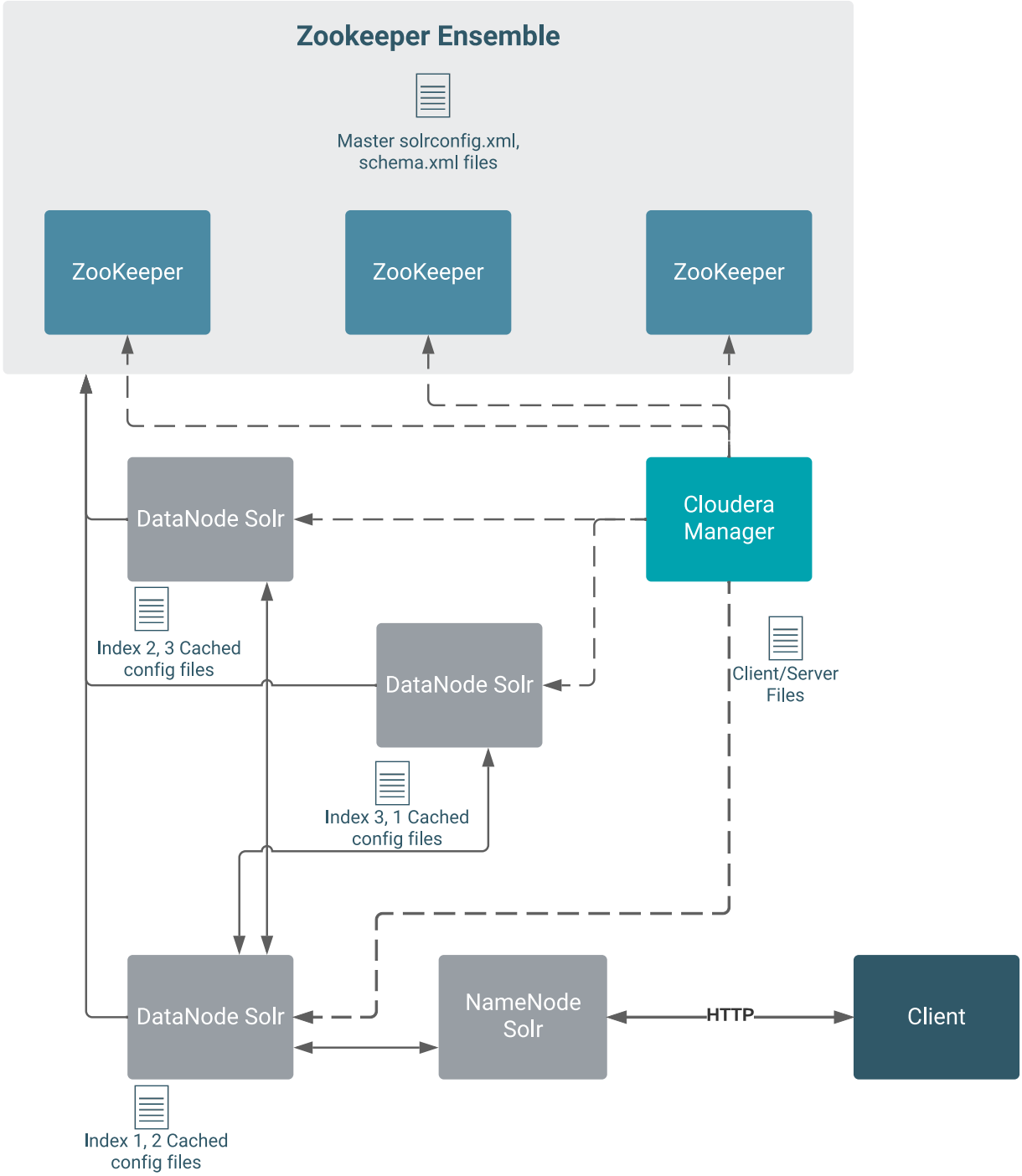

This graphic illustrates:

1. A client submits a query over HTTP.
2. The response is received by the NameNode and then passed to a DataNode.
3. The DataNode distributes the request among other hosts with relevant shards.
4. The results of the query are gathered and returned to the client.

Also notice that the:

- Cloudera Manager provides client and server configuration files to other servers in the deployment.
- ZooKeeper server provides information about the state of the cluster and the other hosts running Solr.

The information a client must send to complete jobs varies:

- For queries, a client must have the hostname of the Solr server and the port to use.
- For actions related to collections, such as adding or deleting collections, the name of the collection is required as well.
- Indexing jobs, such as MapReduceIndexer jobs, use a MapReduce driver that starts a MapReduce job. These jobs can also process morphlines and index the results to Solr.

Local file system support

Learn about the advantages and limitations of using local File System (FS) to store indexes of Cloudera Search.

Apache Lucene-Solr supports various file systems to store index data, including the Hadoop Distributed File System (HDFS) or file systems which are local to the server.

Solr queries perform the best when the index data is available in memory. The HDFS data storage implementation includes various performance tweaks, for example data locality, Short Circuit Data Reads and caching of the HDFS data blocks in memory. For most of the cases this yields acceptable performance, however there are some cases where sub-second response time of queries is critical. In these cases a local index storage (especially using performant Solid State Disks, SSDs) paired with the OS caching, can yield better performance. In the upstream Apache Solr, local FS storage is a widely used and supported feature that is considered stable.

Cloudera curently supports local FS storage for new installations only. The global default behavior of Cloudera Search remains storing indexes on HDFS. If you want to store your indexes locally, you can configure it at collection level, by generating a collection configuration using the solrctl instancedir --generate command with the -localfs argument. This replaces HdfsDirectoryFactory with NRTCachingDirectoryFactory, and HDFS lock factory with native lock factory in the generated solrconfig.xml file. When Solr loads such a collection, it writes the index into a subdirectory of the local data directory (by default var/lib/solr).

Note: If you opt for storing indexes on local FS, it is your responsibility to set quotas on OS level, preventing Solr from consuming all available space under /var, negatively impacting the performance of other services.

Benefits of using local file system for index storage

No need to worry about data locality

The HDFS directory factory attempts to place each HDFS data block on a Data Node which is local to a given Solr server, provided that such a Data Node is up and operational. When the data node goes down, or an HDFS rebalancing operation is triggered in the cluster, the overall data locality of Solr can be ruined which can negatively affect performance. At the moment there is no easy way to restore locality, either you can optimize an index which can have bad side effects, or you use the ADDREPLICA of the Collections API to re-add each replica and then apply DELETEREPLICA on the old replica with bad locality.

With local FS there are no such issues.

Faster disk reads and better caching

The HDFS block cache provides a great help when you have to read data blocks which are not available locally. For the fast execution of a query it is critically important to ensure that a block is available locally - a cache miss can be costly due to the network round trips involved.

Better Query Response Times

Cloudera performed benchmarking to measure document indexing performance, and found that the indexing was roughly two times faster on local FS than on HDFS even when using a magnetic spinning disk.

Table 2: Indexing time in seconds for 10 million records

	HDFS	Local FS	Percentage
2x5 million records, 2x1 thread	3265	2853	87.38
2x5 million records, 2x5 thread	1645	856	52.04
2x5 million records, 2x10 thread	1603	715	44.6
2x5 million records, 2x20 thread	1524	695	45.6
Spark Crunch Indexer, 20 executors	1652	766	46.37
Spark Crunch Indexer, 40 executors	1559	762	48.88
Spark Crunch Indexer, 100 executors	1471	752	51.12

In another case NVMe SSDs were used with a local FS to store the Solr index. In this case the query executions turned out to be three times faster than on HDFS.

Note:

The downsides of this configuration are that replacing an NVMe SSD might require full server downtime (since they are connected directly to the PCI Express bus).

You must take care regarding the RAID configurations, as the local FS does not provide an FS level replication, so the data redundancy has to be provided at the disk level by using a RAID setup and/or at the Solr level by adding more replicas to the collection.

No “Small files problem”

HDFS is sensitive to having a large number of small files stored. Even in spite of Solr periodically merging the segments in the background, an index directory can have many small files. NameNode RPC queues may be periodically overloaded when having a Solr collection with a large number of shards (100+).

Dynamic caching

When using a local FS, caching of the files is managed by the underlying operating system. The operating system can dynamically manage the amount of memory used for caching based on the current memory pressure, whereas the HDFS block cache allocation is more static.

Less impact from other services

By storing the files locally, query performance is not affected by the load on HDFS caused by other services. This is especially important if you have certain query response time SLAs to meet.

Less storage overhead

The index files on HDFS are stored with the same replication factor as configured for the underlying HDFS service. The current Solr implementation allows specifying a custom replication factor for the transaction log files, but for the index files it always uses the replication factor configured for the entire HDFS service even if you want to go with a smaller replication factor.

Cloudera features that are only available on HDFS

Index data encryption

Cloudera provides robust, transparent encryption of the HDFS data where the encryption key management is external to HDFS. Only the authorized HDFS clients can access the data, even

HDFS and the operating system interact using encrypted data only. The solution also supports various Hardware Security Modules and CPU instruction sets to accelerate the encryption.

When storing data on a local FS, encryption is also possible but needs to be implemented and managed external to Cloudera.

Quota support

Cloudera have built in solutions to define quotas for the data stored on HDFS.

When using a local FS, the disk space taken by Solr indexes is not controlled by Cloudera, so monitoring the disk usage and ensuring that index files do not eat up all the space has to be done externally.

File system level redundancy

Solr allows collection level fault tolerance using the replica concept. Provided that the replicas of each shard are stored on different machines, this provides enough protection against the failure of a single server. Additionally to this, HDFS is distributed by nature and it provides block level redundancy. This provides more opportunities for disaster recovery - in case of any server going down, the affected replicas can be restarted on any other server, pointing to the same data set.

Flexible cross-cluster backups

The backup implementation provided by Cloudera Search is relying on the hadoop distcp tool to export index snapshots to HDFS, and it also allows transferring them to remote clusters.

Related Information

Generating Solr collection configuration using instance directories

solrctl Reference

Cloudera Search tasks and processes

For content to be searchable, it must exist in Cloudera and be indexed. Content can either already exist in Cloudera and be indexed on demand, or it can be updated and indexed continuously. To make content searchable, first ensure that it is ingested or stored in Cloudera.

Ingestion

Ingestion is about making data available in Cloudera.

You can move content to Cloudera by using:

- Apache NiFi, a flexible data streaming framework.
- A copy utility such as distcp for HDFS.
- Sqoop, a structured data ingestion connector.

Related Information

Extracting, transforming, and loading data with Cloudera Morphlines

Indexing

Content must be indexed before it can be searched. Learn about how indexing in Cloudera Search happens.

Indexing comprises the following steps:

1. Extraction, transformation, and loading (ETL) - Use existing engines or frameworks such as Apache Tika or Cloudera Morphlines.
   a. Content and metadata extraction
   b. Schema mapping
2. Index creation using Lucene.
   a. Index creation
   b. Index serialization

Indexes are typically stored on a local file system. Lucene supports additional index writers and readers. One HDFS-based interface implemented as part of Apache Blur is integrated with Cloudera Search and has been optimized for Cloudera-stored indexes. All index data in Cloudera Search is stored in and served from HDFS.

You can index content in the following ways:

Batch indexing using MapReduce

To use MapReduce to index documents, run a MapReduce job on content in HDFS to produce a Lucene index. The Lucene index is written to HDFS, and this index is subsequently used by Search services to provide query results.

Batch indexing is most often used when bootstrapping a Search cluster. The Map phase of the MapReduce task parses input into indexable documents, and the Reduce phase indexes the documents produced by the Map. You can also configure a MapReduce-based indexing job to use all assigned resources on the cluster, utilizing multiple reducing steps for intermediate indexing and merging operations, and then writing the reduction to the configured set of shard sets for the service. This makes the batch indexing process as scalable as MapReduce workloads.

NRT indexing using the API

Other clients can complete NRT indexing. This is done when the client first writes files directly to HDFS and then triggers indexing using the Solr REST API. Specifically, the API does the following:

1. Extract content from the document contained in HDFS, where the document is referenced by a URL.
2. Map the content to fields in the search schema.
3. Create or update a Lucene index.

This is useful if you index as part of a larger workflow. For example, you could trigger indexing from an Oozie workflow.

Indexing using the Spark-Solr connector

Using the Spark-Solr connector you have two options to index data into Solr: batch index documents or index streaming data.

For batch indexing you have multiple options, firstly you can use the spark-shell tool to reach Solr with Scala commands. Cloudera recommends this solution mostly for experimenting or smaller indexing jobs.

The second option is to use spark-submit with your spark job, for this you need to create a class which implements SparkApp.RDDProcesor interface. This way you can access the robustness of Spark with the phrases and concepts well-known from Solr. This works in Scala and Java as well.

If you want to index streaming data, you can do it by implementing the SparkApp.StreamingProcessor interface. With this solution you gain access to all the benefits of SparkStreaming and send on the data to Solr.

Querying

After data is available as an index, you can run queries against it.

The query API provided by the Cloudera Search service allows direct queries to be completed or to be facilitated through a command-line tool or graphical interface. Hue includes a simple GUI-based Search application, or you can

create a custom application based on the standard Solr API. Any application that works with Solr is compatible and runs as a search-serving application for Cloudera Search because Solr is at its core.

Backing up and restoring data using Cloudera Search

Cloudera Search includes a backup and restore mechanism primarily designed to provide disaster recovery capability for Apache Solr. You can create a backup of a Solr collection by creating and exporting a snapshot and restoring from this backup to recover from scenarios, such as accidental or malicious data modification or deletion.

Important: Backing up and restoring Cloudera Search is based on the SolrCloud implementation of Apache Solr. Cloudera Search does not support backups using the Solr replication handler.

Cloudera Search includes a backup/restore mechanism primarily designed to provide disaster recovery capability for Apache Solr. You can create a backup of a Solr collection and restore from this backup if the index is corrupted due to a software bug, or if an administrator accidentally or maliciously deletes a collection or a subset of documents. This procedure can also be used as part of a cluster migration (for example, if you are migrating to a cloud environment), or to recover from a failed upgrade.

At a high level, the steps to back up a Solr collection are as follows:

1. Create a snapshot.
2. If you are exporting the snapshot to a remote cluster, prepare the snapshot for export.
3. Export the snapshot to either the local cluster or a remote one. This step uses the Hadoop DistCP utility.

The backup operation uses the built-in Solr snapshot capability to capture a point-in-time, consistent state of index data for a specified Solr collection. You can then use the Hadoop DistCp utility to copy the index files and the associated metadata for that snapshot to a specified location in HDFS or a cloud object store (for example, Amazon S3).

The Solr snapshot mechanism is based on the Apache Lucene IndexDeletionPolicy abstraction, which enables applications such as Cloudera Search to manage the lifecycle of specific index commits. A Solr snapshot assigns a user-specified name to the latest hard-committed state. After the snapshot is created, the Lucene index files associated with the commit are retained until the snapshot is explicitly deleted. The index files associated with the snapshot are preserved regardless of document updates and deletions, segment merges during index optimization, and so on.

Creating a snapshot does not take much time because it only preserves the snapshot metadata and does not copy the associated index files. A snapshot does have some storage overhead corresponding to the size of the index because the index files are retained indefinitely until the snapshot is deleted.

Solr snapshots can help you recover from some scenarios, such as accidental or malicious data modification or deletion. They are insufficient to address others, such as index corruption and accidental or malicious administrative action (for example, deleting a collection, changing collection configuration, and so on). To address these situations, export snapshots regularly and before performing non-trivial administrative operations such as changing the schema, splitting shards, or deleting replicas.

Exporting a snapshot exports the collection metadata as well as the corresponding Lucene index files. This operation internally uses the Hadoop DistCp utility to copy the Lucene index files and metadata, which creates a full backup at the specified location. After the backup is created, the original Solr snapshot can be safely deleted if necessary.

Important: If you create a snapshot and do not export it, you do not have a complete backup, and cannot restore index files if they are accidentally or maliciously deleted.

Related Information

Apache Lucene IndexDeletionPolicy

Backups using the Solr replication handler

Cloudera Runtime 7.2.18

Securing Cloudera Search

Date published: 2015-05-05
Date modified: 2024-04-03

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Search security aspects 4

Enabling ZooKeeper SSL/TLS for Solr and HBase Indexer 4
- Enabling SSL for Solr ZooKeeper 4
- Enabling SSL for HBase Indexer ZooKeeper 4

Enable LDAP authentication in Solr 5

Creating a JAAS configuration file 6

Manage Ranger authorization in Solr 7

Configuring Ranger authorization for Solr in a DDE Cloudera Data Hub cluster 7

Enable Ranger document-level authorization for a Solr collection 9
- Modify the schema and solrconfig files to enable document-level authorization 9
- solrconfig.xml.secure file example 11

Cloudera Search security aspects

Cloudera Search security covers the following security aspects:

- Securing network communication

  Cloudera Search supports TLS for encrypting communications over a network.
- Authentication

  Cloudera Search supports Kerberos and LDAP for authentication.
- Authorization

  Cloudera Search supports Apache Ranger for authorization.

Related Tasks

Configuring Ranger authorization for Solr in a DDE Cloudera Data Hub cluster

Enabling ZooKeeper SSL/TLS for Solr and HBase Indexer

Learn about configuring Solr and HBase Indexer to communicate with ZooKeeper in a secure way.

Related Information

Configure ZooKeeper TLS/SSL using Cloudera Manager

Enabling SSL for Solr ZooKeeper

Learn about enabling secure communications between Solr and ZooKeeper.

Before you begin

- The Enable TLS/SSL for ZooKeeper and Client Port Unification options must be enabled in the ZooKeeper configuration (These are the default settings for a secure cluster).

Procedure

1. In Cloudera Manager, select the Solr service for which you want to enable secure communication.
2. Click the Configuration tab.
3. Search for SSL.
4. Find the Enable TLS/SSL for Solr ZooKeeper property and select it to enable TLS/SSL.
5. Click Save Changes.
6. Restart.

Enabling SSL for HBase Indexer ZooKeeper

Learn about enabling secure communication between HBase Indexer and ZooKeeper.

Before you begin

- The Enable TLS/SSL for ZooKeeper and Client Port Unification options must be selected in the ZooKeeper configuration (These are the default settings for a secure cluster).
- The Enable TLS/SSL for Solr Zookeeper must be selected for the Solr service used with HBase Indexer. When you disable SSL/TLS for HBase indexer ZooKeepeer, make sure that you disable it for the Solr service as well.

Procedure

1. In Cloudera Manager, select the HBase service for which you want to enable secure communication.
2. Click the Configuration tab.
3. Search for SSL.
4. Find the Enable TLS/SSL for HBase ZooKeeper property and select it to enable TLS/SSL.
5. Click Save Changes.
6. Restart.

Enable LDAP authentication in Solr

You can configure LDAP-based authentication using Cloudera Manager at the Solr service level.

Before you begin

Note: Depending on how your environment is set up, you may have issues with DNS resolution. You can avoid it by using IP addresses instead of hostnames.

About this task

Solr supports LDAP authentication for external Solr clients including:

- Command-line tools
- curl
- Web browsers
- Solr Java clients

In some cases, Solr does not support LDAP authentication. Use Kerberos authentication instead in these cases. Solr does not support LDAP authentication with:

- Search indexing components including the MapReduce indexer and Lily HBase indexer.
- Solr internal requests such as those for replication or querying.
- Hadoop delegation token management requests such as GETDELEGATIONTOKEN or RENEWDELEGAT IONTOKEN.

Before you begin

- Configuring LDAP authentication requires that Kerberos authentication is already configured and enabled in Solr.
- For secure LDAP connections, it is a prerequisite that TLS/SSL has been configured and enabled in Solr.

Procedure

1. In Cloudera Manager, select the Solr service.
2. Click the Configuration tab.
3. Select Scope Solr .
4. Select Category Security .
5. Select Enable LDAP Authentication.

6. Enter the LDAP URL in the LDAP URL property.

   To configure a TLS encrypted LDAP connection, select one of the following options:

   - ldaps://*<LDAP_SERVER>*:*<PORT>*

     The default port is 636.

   OR

   - ldap://*<LDAP_SERVER>*:*<PORT>*

     The default port is 389.

     Select Enable LDAP TLS. This is not required when using an LDAP URL with prefix ldaps://, because that already specifies TLS.

   To configure LDAP with unencrypted transmission of usernames and passwords, set ldap://<ldap_server>:<port>, without setting Enable LDAP TLS.
7. Configure only one of following mutually exclusive parameters:
   - LDAP BaseDN: Replaces the username with a "distinguished name" (DN) of the form: uid=userid,ldap_base DN. Typically used for OpenLDAP server installation.
   - Active Directory Domain: Replaces the username with a string username@ldap_domain. Typically used for Active Directory server installation.
8. Launch the Stale Configuration wizard to restart the Solr service and any dependent services.

Creating a JAAS configuration file

Certain applications, such as those using the SolrJ library, require a Java Authentication and Authorization Service (JAAS) configuration file.

- If you are authenticating using kinit to obtain credentials, you can configure the client to use your credentials cache by creating a JAAS file with the following contents:

```
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=false
 useTicketCache=true
 principal="[***USER***]@[***REALM***]";
 };
```

- If you want the client application to authenticate using a keytab, create a JAAS file with the following contents:

```
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=true
 keyTab="[***PATH/TO/USER.KEYTAB***]"
 storeKey=true
 useTicketCache=false
 principal="[***USER***]/[***HOST NAME***]@[***REALM***]";
};
```

[USER***]***

is a valid user name in your environment

/[HOST NAME***]***

If you use a service principal that includes the host name, make sure that it is included in the jaas .conf file (for example, solr/solr01.example.com@EXAMPLE.COM).

[REALM***]***

is your Kerberos realm

[PATH/TO/USER.KEYTAB***]***

is the path to the keytab file you want to use

Manage Ranger authorization in Solr

Using Cloudera Manager, you can turn on and off Ranger authorization for your Solr service.

About this task

Ranger restrictions are consistently applied regardless of the way users attempt to complete actions. For example, restricting access to data in a collection consistently restricts that access, whether queries come from the command line, from a browser, or through the admin console.

Procedure

1. In Cloudera Manager, select the Solr service.
2. Select Configuration and find the RANGER Service property.
3. To turn on Ranger authorization, select the Ranger service that you want the Solr service to depend on. To turn off Ranger authorization, unselect the Ranger service.
4. Click Save Changes.
5. Restart the Solr service.

Results

Ranger authorization for Solr is enabled. The Solr service depends on the selected Ranger service for authorization.

Configuring Ranger authorization for Solr in a DDE Cloudera Data Hub cluster

By default, the Solr service deployed with a Data Discovery and Experience (DDE) cluster is not configured for authorization by Ranger. You can configure authorization manually, using Cloudera Management Console.

Before you begin

Minimum Required Role: environmentAdmin

Procedure

1. Navigate to the Ranger service of the data lake where DDE is running.
2. On the **Service Manager** page, click the Add icon next to Solr.
3. Enter the following information on the **Create Service** page:

   Service Details

Field name	Description
Service Name	Assign a name to the Solr service you want to create. Note down the value you define here. You need to enter it later, when specifying the ranger.plugin.solr.service.name parameter in the Solr Service Advanced Configuration Snippet (Safety Valve) for ranger-solr-security.xml option.

Field name	Description
Active Status	Enabled
Select Tag Service	Select cm_tag.

Configuration Properties

Field name	Description
Username	Assign a placeholder value. This property is not used in case of Kerberos authentication.
Password	Assign a placeholder value. This property is not used in case of Kerberos authentication.
Solr URL	Assign a placeholder value. This property is not used in case of Kerberos authentication.
Add new configurations	Add the following new configurations: • policy.download.auth.users = solr • tag.download.auth.users = solr

Note:

Do not click Test connection. In this use case it is not necessary and it does not work.

4. Click Save.
5. Click on the newly added service.
   The **List of Policies** page opens.
6. Under Action click the Edit icon.
7. In Allow Conditions Select User dropdown select hue.

   This is necessary because the Hue service breaks if it has no permission to access Solr.

   To keep the cluster accessible to non-admin users, you can add other users to the default policy or you can define additional policies.
8. Grant full admin privileges to users that you want to be able to access the Solr Admin UI.
   a) Click Add under Allow Conditions to add a new condition, then add the user or users from the Select User drop-down.
   b) Click Add Permissions then select the Select/Deselect All option.
   c) Accept the selection, then click Save.
9. Navigate to the Cloudera Manager UI on the DDE cluster.
10. From Clusters select Solr.
11. Select the **Configuration** tab.
12. In the Search field start typing ‘safety’
13. Click Add under Solr Service Advanced Configuration Snippet (Safety Valve).
14. Define the following:

    Key

    ranger.plugin.solr.service.name

    Value

    The Solr Service Name you assigned when creating the Solr service in Ranger.
15. Click Save Changes.
16. Click the Status tab to refresh the window.
17. Click the Stale Configuration: Restart needed indicator on top of the page.
18. Click Restart Stale Services.
19. Click Restart Now.

Enable Ranger document-level authorization for a Solr collection

By default, Ranger authorization works on collection level. Ranger allows you to configure document level security for individual Solr collections. This requires updating the solrconfig.xml file belonging to the particular collection. For the authorization to work on existing collections, you need to update the collection as well, by adding the rang er_auth parameter with an appropriate value to individual documents.

Important:

Document-level authorization does not prevent users from modifying documents or performing other update operations on the collection. Update operations are only governed by collection-level authorization.

Document-level authorization can exclusively be used to prevent documents being returned in query results. If users are not granted access to a document, those documents are not returned even if that user submits a query that matches those documents. This does not affect attempted updates.

Consequently, it is possible for a user to not have access to a set of documents based on document-level security, but to still be able to modify the documents using their collection-level authorization update rights. This means that a user can delete all documents in the collection. Similarly, a user might modify all documents, adding their authorization token to each one. After such a modification, the user could access any document using querying.

Therefore, if you restrict access using document-level security, consider granting collection-level update rights only to those users you trust and assume they will be able to access every document in the collection.

Prerequisites

1. Define or edit roles on Ranger UI and assign them to users/groups.
2. Create a new collection configuration or download an existing one for editing.

Enabling document-level authorization

1. Add ranger_auth field to the schema file and add hooks to solrconfig.xml that trigger document-level authorization.
2. Disable Ranger authorization.
3. Upload the configuration to ZooKeeper.
4. If you updated an existing collection to enable document-level authorization, add appropriate values to the newly created ranger_auth field before you turn on Ranger authorization. For example, you can reindex your collection, using one of the batch-indexing options offered by Cloudera Search.
5. To take document-level authorization into use, create a collection using the updated configuration (new collections) or reload your collection (updating an existing collection).
6. Enable Ranger authorization.

Modify the schema and solrconfig files to enable document-level authorization

To enable document-level Ranger authorization, you need to add a field in the schema file determining the roles that can access a particular document in the collection. You also need to edit the solrconfig file to include the hooks that trigger Ranger document level authorization based on the schema field values.

Before you begin

1. Disable Ranger authorization.

2. Define or edit roles on Ranger UI and assign them to users/groups.
3. Create a new collection configuration or download an existing one for editing.

Procedure

1. If you are using Kerberos, kinit as a user with sufficient rights to create or modify collections:

```
kinit [***KERBEROS PRINCIPAL***]@[***EXAMPLE.COM***]
```

   Replace *[***KERBEROS PRINCIPAL***]@[***EXAMPLE.COM***]* with your Kerberos principal and your Kerberos realm name respectively.

2. Go to the conf subdirectory of the newly created/downloaded folder to edit the schema file. (In our example it is mycollection/conf)

   The file name is either managed-schema or schema.xml based on the schema factory used.

3. Add a field that determines which roles have access to a particular document. In this example we name this field ranger_auth.

   Add the following to the list of fields in your schema, making sure the value of the type property is string; the values of the indexed, stored, and multiValued properties are true:

```
<field name="ranger_auth" type="string" indexed="true" stored="true" re
quired="false" multiValued="true"/>
```

4. Open the solrconfig.xml file for editing
5. Locate the section which contains the list of search components. In the default configuration it starts with a comment block similar to this:

```
<!-- Search Components
       Search components are registered to SolrCore and used by
       instances of SearchHandler (which can access them by name)
       ...
  -->
```

6. Add a new SearchComponent:

```
<searchComponent name="queryDocAuthorization" class="org.apache.ranger.a
uthorization.solr.authorizer.RangerSolrAuthorizer">
    <str name="enabled">true</str>
    <!-- The field which contains the role or list of roles which are all
owed to query a particular document -->
    <str name="rangerAuthField">ranger_auth</str>
    <!-- If the rangerAuthField contains this value, all roles will be all
owed to query that particular document -->
    <str name="allRolesToken">*</str>
 </searchComponent>
```

7. Add queryDocAuthorization to the first-components array of the /query, /get, /browse, /tvrh, /terms, and /elevate request handlers as well, in case they are present in your solrconfig.xml.

   Locate the request handler /select section:

```
<requestHandler name="/select" class="solr.SearchHandler">
    <!-- default values for query parameters can be specified, these
         will be overridden by parameters in the request
       -->
    <lst name="defaults">
      <str name="echoParams">explicit</str>
```

```
      <int name="rows">10</int>
```

To the end of this section, before the closing tag, insert queryDocAuthorization to the first-components array:

```
<arr name="first-components">
        <str>queryDocAuthorization</str>
      </arr>
  </requestHandler>
```

Note: /get (real-time get request handler) is implicitly defined even if it does not appear in solrconfig.xml. In case of using document level security, Cloudera recommends to also protect the /get handler. To do that, you need to explicitly add it to the solrconfig.xml and define queryDocAuthorization as a first-co mponent:

```
<requestHandler name="/get" class="solr.RealTimeGetHandler">
  <lst name="defaults">
    <str name="omitHeader">true</str>
    <str name="wt">json</str>
    <str name="indent">true</str>
  </lst>
 <arr name="first-components">
        <str>queryDocAuthorization</str>
      </arr>
</requestHandler>
```

8. Locate the requestParsers section:

```
<requestParsers enableRemoteStreaming="true"
                    multipartUploadLimitInKB="2048000"
                    ...
                    />
```

Ensure that this section has a boolean attribute called addHttpRequestToContext with a value of true:

```
<requestParsers enableRemoteStreaming="true"
                    multipartUploadLimitInKB="2048000"
                    formdataUploadLimitInKB="2048"
                    addHttpRequestToContext="true"/>
```

What to do next

1. Upload the configuration metadata to ZooKeeper.
2. Populate the ranger_auth field for each document with roles you have defined in Ranger.
3. Create a new collection using the the updated configuration metadata, or update the configuration of an existing collection.
4. Enable Ranger authorization.

Related Concepts

Enable Ranger document-level authorization for a Solr collection

solrconfig.xml.secure file example

You can copy this xml file for editing by clicking the Copy to clipboard icon.

solconfig.xml.secure

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
```

```
 Licensed to the Apache Software Foundation (ASF) under one or more
 contributor license agreements.  See the NOTICE file distributed with
 this work for additional information regarding copyright ownership.
 The ASF licenses this file to You under the Apache License, Version 2.0
 (the "License"); you may not use this file except in compliance with
 the License.  You may obtain a copy of the License at

     http://www.apache.org/licenses/LICENSE-2.0
 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.
-->
<!--
     For more details about configurations options that may appear in
     this file, see http://wiki.apache.org/solr/SolrConfigXml.
-->
<config>
  <!-- In all configuration below, a prefix of "solr." for class names
       is an alias that causes solr to search appropriate packages,
       including org.apache.solr.(search|update|request|core|analysis)

       You may also specify a fully qualified Java classname if you
       have your own custom plugins.
    -->
  <!-- Controls what version of Lucene various components of Solr
       adhere to.  Generally, you want to use the latest version to
       get all bug fixes and improvements. It is highly recommended
       that you fully re-index after changing this setting as it can
       affect both how text is indexed and queried.
  -->
  <luceneMatchVersion>8.4.1</luceneMatchVersion>

  <!-- <lib/> directives can be used to instruct Solr to load any Jars
       identified and use them to resolve any "plugins" specified in
       your solrconfig.xml or schema.xml (ie: Analyzers, Request
       Handlers, etc...).

       All directories and paths are resolved relative to the
       instanceDir.

       Please note that <lib/> directives are processed in the order
       that they appear in your solrconfig.xml file, and are "stacked"
       on top of each other when building a ClassLoader - so if you have
       plugin jars with dependencies on other jars, the "lower level"
       dependency jars should be loaded first.

       If a "./lib" directory exists in your instanceDir, all files
       found in it are included as if you had used the following
       syntax...
              <lib dir="./lib" />
    -->

  <!-- A 'dir' option by itself adds any files found in the directory
       to the classpath, this is useful for including all jars in a
       directory.
       When a 'regex' is specified in addition to a 'dir', only the
       files in that directory which completely match the regex
       (anchored on both ends) will be included.

       If a 'dir' option (with or without a regex) is used and nothing
       is found that matches, a warning will be logged.
```

```
       The example below can be used to load a solr-contrib along
       with their external dependencies.
    -->
    <!-- <lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-ltr-
\d.*\.jar" /> -->

  <!-- an exact 'path' can be used instead of a 'dir' to specify a
       specific jar file.  This will cause a serious error to be logged
       if it can't be loaded.
    -->
  <!--
     <lib path="../a-jar-that-does-not-exist.jar" />
  -->

  <!-- Data Directory

       Used to specify an alternate directory to hold all index data
       other than the default ./data under the Solr home.  If
       replication is in use, this should match the replication
       configuration.
    -->
  <dataDir>${solr.data.dir:}</dataDir>


  <!-- The DirectoryFactory to use for indexes.
       solr.StandardDirectoryFactory is filesystem
       based and tries to pick the best implementation for the current
       JVM and platform.  solr.NRTCachingDirectoryFactory, the default,
       wraps solr.StandardDirectoryFactory and caches small files in memory
       for better NRT performance.

       One can force a particular implementation via solr.MMapDirectoryFa
ctory,
       solr.NIOFSDirectoryFactory, or solr.SimpleFSDirectoryFactory.
       solr.RAMDirectoryFactory is memory based and not persistent.
    -->

  <directoryFactory name="DirectoryFactory" class="${solr.directoryFactory
:org.apache.solr.core.HdfsDirectoryFactory}">
    <str name="solr.hdfs.home">${solr.hdfs.home:}</str>
    <str name="solr.hdfs.confdir">${solr.hdfs.confdir:}</str>
    <str name="solr.hdfs.security.kerberos.enabled">${solr.hdfs.security.ker
beros.enabled:false}</str>
    <str name="solr.hdfs.security.kerberos.keytabfile">${solr.hdfs.securi
ty.kerberos.keytabfile:}</str>
    <str name="solr.hdfs.security.kerberos.principal">${solr.hdfs.securit
y.kerberos.principal:}</str>
    <bool name="solr.hdfs.blockcache.enabled">${solr.hdfs.blockcache.enable
d:true}</bool>
    <!-- Enable/Disable using one global cache for all SolrCores.
    The settings used will be from the first HdfsDirectoryFactory created.
-->
    <str name="solr.hdfs.blockcache.global">${solr.hdfs.blockcache.global:tr
ue}</str>
    <int name="solr.hdfs.blockcache.slab.count">${solr.hdfs.blockcache.sla
b.count:1}</int>
    <bool name="solr.hdfs.blockcache.direct.memory.allocation">${solr.hdfs.
blockcache.direct.memory.allocation:true}</bool>
    <int name="solr.hdfs.blockcache.blocksperbank">${solr.hdfs.blockcache.b
locksperbank:16384}</int>
    <bool name="solr.hdfs.blockcache.read.enabled">${solr.hdfs.blockcache.
read.enabled:true}</bool>
    <bool name="solr.hdfs.blockcache.write.enabled">${solr.hdfs.blockcache
.write.enabled:false}</bool>
```

```
    <!-- the buffercount is actually the total size in bytes to used for bu
ffer caching -->
    <int name="solr.hdfs.blockcache.bufferstore.buffercount">${solr.hdfs.bl
ockcache.bufferstore.buffercount:0}</int>
    <bool name="solr.hdfs.nrtcachingdirectory.enable">${solr.hdfs.nrtcachi
ngdirectory.enable:true}</bool>
    <int name="solr.hdfs.nrtcachingdirectory.maxmergesizemb">${solr.hdfs.nrt
cachingdirectory.maxmergesizemb:16}</int>
    <int name="solr.hdfs.nrtcachingdirectory.maxcachedmb">${solr.hdfs.nrtc
achingdirectory.maxcachedmb:192}</int>
    <!-- HDFS Block Locality Reporter can be toggled on and off -->
    <bool name="solr.hdfs.locality.metrics.enabled">${solr.hdfs.locality.
metrics.enabled:false}</bool>
  </directoryFactory>
  <!-- The CodecFactory for defining the format of the inverted index.
       The default implementation is SchemaCodecFactory, which is the offici
al Lucene
       index format, but hooks into the schema to provide per-field custom
ization of
       the postings lists and per-document values in the fieldType element
       (postingsFormat/docValuesFormat). Note that most of the alternative
 implementations
       are experimental, so if you choose to customize the index format, it
's a good
       idea to convert back to the official format e.g. via IndexWriter.ad
dIndexes(IndexReader)
       before upgrading to a newer version to avoid unnecessary reindexing.
       A "compressionMode" string element can be added to <codecFactory> to
 choose
       between the existing compression modes in the default codec: "BEST_S
PEED" (default)
       or "BEST_COMPRESSION".
  -->
  <codecFactory class="solr.SchemaCodecFactory"/>

  <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       Index Config - These settings control low-level behavior of indexing
       Most example settings here show the default value, but are commented
       out, to more easily see where customizations have been made.
       Note: This replaces <indexDefaults> and <mainIndex> from older versi
ons
       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 -->
  <indexConfig>
    <!-- maxFieldLength was removed in 4.0. To get similar behavior, include
 a
         LimitTokenCountFilterFactory in your fieldType definition. E.g.
     <filter class="solr.LimitTokenCountFilterFactory" maxTokenCount="10000
"/>
    -->
    <!-- Maximum time to wait for a write lock (ms) for an IndexWriter. Defa
ult: 1000 -->
    <!-- <writeLockTimeout>1000</writeLockTimeout>  -->

    <!-- Expert: Enabling compound file will use less files for the index,
         using fewer file descriptors on the expense of performance decrea
se.
         Default in Lucene is "true". Default in Solr is "false" (since 3.6)
 -->
    <!-- <useCompoundFile>false</useCompoundFile> -->

    <!-- ramBufferSizeMB sets the amount of RAM that may be used by Lucene
         indexing for buffering added documents and deletions before they
 are
```

```
         flushed to the Directory.
         maxBufferedDocs sets a limit on the number of documents buffered
         before flushing.
         If both ramBufferSizeMB and maxBufferedDocs is set, then
         Lucene will flush based on whichever limit is hit first.  -->
    <ramBufferSizeMB>128</ramBufferSizeMB>
    <!-- <maxBufferedDocs>1000</maxBufferedDocs> -->

    <!-- Expert: ramPerThreadHardLimitMB sets the maximum amount of RAM that
 can be consumed
         per thread before they are flushed. When limit is exceeded, this
 triggers a forced
         flush even if ramBufferSizeMB has not been exceeded.
         This is a safety limit to prevent Lucene's DocumentsWriterPerThread
 from address space
         exhaustion due to its internal 32 bit signed integer based memory
addressing.
         The specified value should be greater than 0 and less than 2048MB.
When not specified,
         Solr uses Lucene's default value 1945. -->
    <!-- <ramPerThreadHardLimitMB>1945</ramPerThreadHardLimitMB> -->

    <!-- Expert: Merge Policy
         The Merge Policy in Lucene controls how merging of segments is done
.
         The default since Solr/Lucene 3.3 is TieredMergePolicy.
         The default since Lucene 2.3 was the LogByteSizeMergePolicy,
         Even older versions of Lucene used LogDocMergePolicy.
      -->
    <!--
        <mergePolicyFactory class="org.apache.solr.index.TieredMergePolic
yFactory">
          <int name="maxMergeAtOnce">10</int>
          <int name="segmentsPerTier">10</int>
          <double name="noCFSRatio">0.1</double>
        </mergePolicyFactory>
      -->

    <!-- Expert: Merge Scheduler
         The Merge Scheduler in Lucene controls how merges are
         performed.  The ConcurrentMergeScheduler (Lucene 2.3 default)
         can perform merges in the background using separate threads.
         The SerialMergeScheduler (Lucene 2.2 default) does not.
     -->
    <!--
       <mergeScheduler class="org.apache.lucene.index.ConcurrentMergeSched
uler"/>
       -->

    <!-- LockFactory

         This option specifies which Lucene LockFactory implementation
         to use.

         single = SingleInstanceLockFactory - suggested for a
                  read-only index or when there is no possibility of
                  another process trying to modify the index.
         native = NativeFSLockFactory - uses OS native file locking.
                  Do not use when multiple solr webapps in the same
                  JVM are attempting to share a single index.
         simple = SimpleFSLockFactory  - uses a plain file for locking

         Defaults: 'native' is default for Solr3.6 and later, otherwise
                   'simple' is the default
```

```
        More details on the nuances of each LockFactory...
        http://wiki.apache.org/lucene-java/AvailableLockFactories
  -->
  <lockType>${solr.lock.type:hdfs}</lockType>

  <!-- Commit Deletion Policy
        Custom deletion policies can be specified here. The class must
        implement org.apache.lucene.index.IndexDeletionPolicy.

        The default Solr IndexDeletionPolicy implementation supports
        deleting index commit points on number of commits, age of
        commit point and optimized status.

        The latest commit point should always be preserved regardless
        of the criteria.
  -->
  <!--
  <deletionPolicy class="solr.SolrDeletionPolicy">
  -->
  <!-- The number of commit points to be kept -->
  <!-- <str name="maxCommitsToKeep">1</str> -->
  <!-- The number of optimized commit points to be kept -->
  <!-- <str name="maxOptimizedCommitsToKeep">0</str> -->
  <!--
      Delete all commit points once they have reached the given age.
      Supports DateMathParser syntax e.g.
    -->
  <!--
     <str name="maxCommitAge">30MINUTES</str>
     <str name="maxCommitAge">1DAY</str>
  -->
  <!--
  </deletionPolicy>
  -->

  <!-- Lucene Infostream

        To aid in advanced debugging, Lucene provides an "InfoStream"
        of detailed information when indexing.

        Setting The value to true will instruct the underlying Lucene
        IndexWriter to write its debugging info the specified file
    -->
  <!-- <infoStream file="INFOSTREAM.txt">false</infoStream> -->
</indexConfig>


<!-- JMX

     This example enables JMX if and only if an existing MBeanServer
     is found, use this if you want to configure JMX through JVM
     parameters. Remove this to disable exposing Solr configuration
     and statistics to JMX.

     For more details see http://wiki.apache.org/solr/SolrJmx
  -->
<jmx />
<!-- If you want to connect to a particular server, specify the
     agentId
  -->
<!-- <jmx agentId="myAgent" /> -->
<!-- If you want to start a new MBeanServer, specify the serviceUrl -->
<!-- <jmx serviceUrl="service:jmx:rmi:///jndi/rmi://localhost:9999/solr"/>
```

```
    -->

  <!-- The default high-performance update handler -->
  <updateHandler class="solr.DirectUpdateHandler2">

    <!-- Enables a transaction log, used for real-time get, durability, and
         and solr cloud replica recovery.  The log can grow as big as
         uncommitted changes to the index, so use of a hard autoCommit
         is recommended (see below).
         "dir" - the target directory for transaction logs, defaults to the
                solr data directory.
         "numVersionBuckets" - sets the number of buckets used to keep
                track of max version values when checking for re-ordered
                updates; increase this value to reduce the cost of
                synchronizing access to version buckets during high-volume
                indexing, this requires 8 bytes (long) * numVersionBuckets
                of heap space per Solr core.
    -->
    <updateLog>
      <str name="dir">${solr.ulog.dir:}</str>
      <int name="tlogDfsReplication">${solr.ulog.tlogDfsReplication:3}</int>
      <int name="numVersionBuckets">${solr.ulog.numVersionBuckets:65536}</
int>
    </updateLog>

    <!-- AutoCommit
         Perform a hard commit automatically under certain conditions.
         Instead of enabling autoCommit, consider using "commitWithin"
         when adding documents.

         http://wiki.apache.org/solr/UpdateXmlMessages

         maxDocs - Maximum number of documents to add since the last
                   commit before automatically triggering a new commit.

         maxTime - Maximum amount of time in ms that is allowed to pass
                   since a document was added before automatically
                   triggering a new commit.
         openSearcher - if false, the commit causes recent index changes
           to be flushed to stable storage, but does not cause a new
           searcher to be opened to make those changes visible.

         If the updateLog is enabled, then it's highly recommended to
         have some sort of hard autoCommit to limit the log size.
      -->
    <autoCommit>
      <maxTime>${solr.autoCommit.maxTime:60000}</maxTime>
      <openSearcher>false</openSearcher>
    </autoCommit>
    <!-- softAutoCommit is like autoCommit except it causes a
         'soft' commit which only ensures that changes are visible
         but does not ensure that data is synced to disk.  This is
         faster and more near-realtime friendly than a hard commit.
      -->

    <autoSoftCommit>
      <maxTime>${solr.autoSoftCommit.maxTime:15000}</maxTime>
    </autoSoftCommit>

    <!-- Update Related Event Listeners

         Various IndexWriter related events can trigger Listeners to
         take actions.
```

```
         postCommit - fired after every commit or optimize command
         postOptimize - fired after every optimize command
      -->
    <!-- The RunExecutableListener executes an external command from a
         hook such as postCommit or postOptimize.

         exe - the name of the executable to run
         dir - dir to use as the current working directory. (default=".")
         wait - the calling thread waits until the executable returns.
                (default="true")
         args - the arguments to pass to the program.  (default is none)
         env - environment variables to set.  (default is none)
      -->
    <!-- This example shows how RunExecutableListener could be used
         with the script based replication...
         http://wiki.apache.org/solr/CollectionDistribution
      -->
    <!--
       <listener event="postCommit" class="solr.RunExecutableListener">
         <str name="exe">solr/bin/snapshooter</str>
         <str name="dir">.</str>
         <bool name="wait">true</bool>
         <arr name="args"> <str>arg1</str> <str>arg2</str> </arr>
         <arr name="env"> <str>MYVAR=val1</str> </arr>
       </listener>
      -->

  </updateHandler>

  <!-- IndexReaderFactory

       Use the following format to specify a custom IndexReaderFactory,
       which allows for alternate IndexReader implementations.

       ** Experimental Feature **

       Please note - Using a custom IndexReaderFactory may prevent
       certain other features from working. The API to
       IndexReaderFactory may change without warning or may even be
       removed from future releases if the problems cannot be
       resolved.

       ** Features that may not work with custom IndexReaderFactory **
       The ReplicationHandler assumes a disk-resident index. Using a
       custom IndexReader implementation may cause incompatibility
       with ReplicationHandler and may cause replication to not work
       correctly. See SOLR-1366 for details.

    -->
  <!--
  <indexReaderFactory name="IndexReaderFactory" class="package.class">
    <str name="someArg">Some Value</str>
  </indexReaderFactory >
  -->

  <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       Query section - these settings control query time things like caches
       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 -->
  <query>

    <!-- Maximum number of clauses allowed when parsing a boolean query stri
ng.
```

```
         This limit only impacts boolean queries specified by a user as part
 of a query string,
         and provides per-collection controls on how complex user specified
 boolean queries can
         be.  Query strings that specify more clauses then this will result
 in an error.

         If this per-collection limit is greater then the global `maxBoolea
nClauses` limit
         specified in `solr.xml`, it will have no effect, as that setting al
so limits the size
         of user specified boolean queries.
      -->
    <maxBooleanClauses>${solr.max.booleanClauses:1024}</maxBooleanClauses>
    <!-- Solr Internal Query Caches

         There are two implementations of cache available for Solr,
         LRUCache, based on a synchronized LinkedHashMap, and
         FastLRUCache, based on a ConcurrentHashMap.

         FastLRUCache has faster gets and slower puts in single
         threaded operation and thus is generally faster than LRUCache
         when the hit ratio of the cache is high (> 75%), and may be
         faster under other scenarios on multi-cpu systems.
    -->

    <!-- Filter Cache

         Cache used by SolrIndexSearcher for filters (DocSets),
         unordered sets of *all* documents that match a query.  When a
         new searcher is opened, its caches may be prepopulated or
         "autowarmed" using data from caches in the old searcher.
         autowarmCount is the number of items to prepopulate.  For
         LRUCache, the autowarmed items will be the most recently
         accessed items.
         Parameters:
           class - the SolrCache implementation LRUCache or
               (LRUCache or FastLRUCache)
           size - the maximum number of entries in the cache
           initialSize - the initial capacity (number of entries) of
               the cache.  (see java.util.HashMap)
           autowarmCount - the number of entries to prepopulate from
               and old cache.
           maxRamMB - the maximum amount of RAM (in MB) that this cache is a
llowed
                      to occupy. Note that when this option is specified,
 the size
                      and initialSize parameters are ignored.
      -->
    <filterCache class="solr.FastLRUCache"
                 size="512"
                 initialSize="512"
                 autowarmCount="0"/>

    <!-- Query Result Cache

         Caches results of searches - ordered lists of document ids
         (DocList) based on a query, a sort, and the range of documents r
equested.
         Additional supported parameter by LRUCache:
            maxRamMB - the maximum amount of RAM (in MB) that this cache is
 allowed
                       to occupy
      -->
```

```
    <queryResultCache class="solr.LRUCache"
                      size="512"
                      initialSize="512"
                      autowarmCount="0"/>

    <!-- Document Cache

         Caches Lucene Document objects (the stored fields for each
         document).  Since Lucene internal document ids are transient,
         this cache will not be autowarmed.
      -->
    <documentCache class="solr.LRUCache"
                   size="512"
                   initialSize="512"
                   autowarmCount="0"/>

    <!-- custom cache currently used by block join -->
    <cache name="perSegFilter"
           class="solr.search.LRUCache"
           size="10"
           initialSize="0"
           autowarmCount="10"
           regenerator="solr.NoOpRegenerator" />

    <!-- Field Value Cache

         Cache used to hold field values that are quickly accessible
         by document id.  The fieldValueCache is created by default
         even if not configured here.
      -->
    <!--
       <fieldValueCache class="solr.FastLRUCache"
                        size="512"
                        autowarmCount="128"
                        showItems="32" />
      -->

    <!-- Custom Cache
         Example of a generic cache.  These caches may be accessed by
         name through SolrIndexSearcher.getCache(),cacheLookup(), and
         cacheInsert().  The purpose is to enable easy caching of
         user/application level data.  The regenerator argument should
         be specified as an implementation of solr.CacheRegenerator
         if autowarming is desired.
      -->
    <!--
       <cache name="myUserCache"
              class="solr.LRUCache"
              size="4096"
              initialSize="1024"
              autowarmCount="1024"
              regenerator="com.mycompany.MyRegenerator"
              />
      -->

    <!-- Lazy Field Loading

         If true, stored fields that are not requested will be loaded
         lazily.  This can result in a significant speed improvement
         if the usual case is to not load all stored fields,
         especially if the skipped fields are large compressed text
         fields.
    -->
    <enableLazyFieldLoading>true</enableLazyFieldLoading>
```

```
    <!-- Use Filter For Sorted Query

         A possible optimization that attempts to use a filter to
         satisfy a search.  If the requested sort does not include
         score, then the filterCache will be checked for a filter
         matching the query. If found, the filter will be used as the
         source of document ids, and then the sort will be applied to
         that.

         For most situations, this will not be useful unless you
         frequently get the same search repeatedly with different sort
         options, and none of them ever use "score"
      -->
    <!--
       <useFilterForSortedQuery>true</useFilterForSortedQuery>
      -->

    <!-- Result Window Size
         An optimization for use with the queryResultCache.  When a search
         is requested, a superset of the requested number of document ids
         are collected.  For example, if a search for a particular query
         requests matching documents 10 through 19, and queryWindowSize is
 50,
         then documents 0 through 49 will be collected and cached.  Any fu
rther
         requests in that range can be satisfied via the cache.
      -->
    <queryResultWindowSize>20</queryResultWindowSize>

    <!-- Maximum number of documents to cache for any entry in the
         queryResultCache.
      -->
    <queryResultMaxDocsCached>200</queryResultMaxDocsCached>
    <!-- Query Related Event Listeners
         Various IndexSearcher related events can trigger Listeners to
         take actions.

         newSearcher - fired whenever a new searcher is being prepared
         and there is a current searcher handling requests (aka
         registered).  It can be used to prime certain caches to
         prevent long request times for certain requests.

         firstSearcher - fired whenever a new searcher is being
         prepared but there is no current registered searcher to handle
         requests or to gain autowarming data from.

      -->
    <!-- QuerySenderListener takes an array of NamedList and executes a
         local query request for each NamedList in sequence.
      -->
    <listener event="newSearcher" class="solr.QuerySenderListener">
      <arr name="queries">
        <!--
           <lst><str name="q">solr</str><str name="sort">price asc</str></
lst>
           <lst><str name="q">rocks</str><str name="sort">weight asc</str></
lst>
          -->
      </arr>
    </listener>
    <listener event="firstSearcher" class="solr.QuerySenderListener">
      <arr name="queries">
        <!--
```

```
        <lst>
          <str name="q">static firstSearcher warming in solrconfig.xml</str>
        </lst>
        -->
      </arr>
    </listener>

    <!-- Use Cold Searcher

         If a search request comes in and there is no current
         registered searcher, then immediately register the still
         warming searcher and use it.  If "false" then all requests
         will block until the first searcher is done warming.
      -->
    <useColdSearcher>false</useColdSearcher>
    <!-- Slow Query Request Logging

         Any queries that take longer than the specified threshold
         will be logged as "slow" queries.
         To disable slow request logging for this Solr config,
         set the value to -1
      -->
    <slowQueryThresholdMillis>5000</slowQueryThresholdMillis>

  </query>


  <!-- Request Dispatcher

       This section contains instructions for how the SolrDispatchFilter
       should behave when processing requests for this SolrCore.

    -->
  <requestDispatcher>
    <!-- Request Parsing

         These settings indicate how Solr Requests may be parsed, and
         what restrictions may be placed on the ContentStreams from
         those requests
         enableRemoteStreaming - enables use of the stream.file
         and stream.url parameters for specifying remote streams.

         multipartUploadLimitInKB - specifies the max size (in KiB) of
         Multipart File Uploads that Solr will allow in a Request.

         formdataUploadLimitInKB - specifies the max size (in KiB) of
         form data (application/x-www-form-urlencoded) sent via
         POST. You can use POST to pass request parameters not
         fitting into the URL.

         addHttpRequestToContext - if set to true, it will instruct
         the requestParsers to include the original HttpServletRequest
         object in the context map of the SolrQueryRequest under the
         key "httpRequest". It will not be used by any of the existing
         Solr components, but may be useful when developing custom
         plugins.
         *** WARNING ***
         Before enabling remote streaming, you should make sure your
         system has authentication enabled.

      -->
    <requestParsers enableRemoteStreaming="true"
                    multipartUploadLimitInKB="2048000"
                    formdataUploadLimitInKB="2048"
```

```
                    addHttpRequestToContext="true"/>

  <!-- HTTP Caching

       Set HTTP caching related parameters (for proxy caches and clients).

       The options below instruct Solr not to output any HTTP Caching
       related headers
    -->
  <httpCaching never304="true" />
  <!-- If you include a <cacheControl> directive, it will be used to
       generate a Cache-Control header (as well as an Expires header
       if the value contains "max-age=")
       By default, no Cache-Control header is generated.
       You can use the <cacheControl> option even if you have set
       never304="true"
    -->
  <!--
     <httpCaching never304="true" >
       <cacheControl>max-age=30, public</cacheControl>
     </httpCaching>
    -->
  <!-- To enable Solr to respond with automatically generated HTTP
       Caching headers, and to response to Cache Validation requests
       correctly, set the value of never304="false"

       This will cause Solr to generate Last-Modified and ETag
       headers based on the properties of the Index.

       The following options can also be specified to affect the
       values of these headers...

       lastModFrom - the default value is "openTime" which means the
       Last-Modified value (and validation against If-Modified-Since
       requests) will all be relative to when the current Searcher
       was opened.  You can change it to lastModFrom="dirLastMod" if
       you want the value to exactly correspond to when the physical
       index was last modified.

       etagSeed="..." is an option you can change to force the ETag
       header (and validation against If-None-Match requests) to be
       different even if the index has not changed (ie: when making
       significant changes to your config file)
       (lastModifiedFrom and etagSeed are both ignored if you use
       the never304="true" option)
    -->
  <!--
     <httpCaching lastModifiedFrom="openTime"
                  etagSeed="Solr">
       <cacheControl>max-age=30, public</cacheControl>
     </httpCaching>
    -->
</requestDispatcher>

<!-- Request Handlers

     http://wiki.apache.org/solr/SolrRequestHandler

     Incoming queries will be dispatched to a specific handler by name
     based on the path specified in the request.

     If a Request Handler is declared with startup="lazy", then it will
     not be initialized until the first request that uses it.
```

```
    -->
  <!-- SearchHandler
       http://wiki.apache.org/solr/SearchHandler

       For processing Search Queries, the primary Request Handler
       provided with Solr is "SearchHandler" It delegates to a sequent
       of SearchComponents (see below) and supports distributed
       queries across multiple shards
    -->


<requestHandler name="/get" class="solr.RealTimeGetHandler">
  <lst name="defaults">
    <str name="omitHeader">true</str>
    <str name="wt">json</str>
    <str name="indent">true</str>
  </lst>
 <arr name="first-components">
        <str>queryDocAuthorization</str>
      </arr>
</requestHandler>



 <requestHandler name="/select" class="solr.SearchHandler">
    <!-- default values for query parameters can be specified, these
         will be overridden by parameters in the request
      -->
    <lst name="defaults">
      <str name="echoParams">explicit</str>
      <int name="rows">10</int>
      <!-- Default search field
         <str name="df">text</str>
        -->
      <!-- Change from JSON to XML format (the default prior to Solr 7.0)
         <str name="wt">xml</str>
        -->
    </lst>
    <!-- In addition to defaults, "appends" params can be specified
         to identify values which should be appended to the list of
         multi-val params from the query (or the existing "defaults").
      -->
    <!-- In this example, the param "fq=instock:true" would be appended to
         any query time fq params the user may specify, as a mechanism for
         partitioning the index, independent of any user selected filtering
         that may also be desired (perhaps as a result of faceted searching
).

         NOTE: there is *absolutely* nothing a client can do to prevent thes
e
         "appends" values from being used, so don't use this mechanism
         unless you are sure you always want it.
      -->
    <!--
       <lst name="appends">
         <str name="fq">inStock:true</str>
       </lst>
      -->
    <!-- "invariants" are a way of letting the Solr maintainer lock down
         the options available to Solr clients.  Any params values
         specified here are used regardless of what values may be specified
         in either the query, the "defaults", or the "appends" params.

         In this example, the facet.field and facet.query params would
```

```
         be fixed, limiting the facets clients can use.  Faceting is
         not turned on by default - but if the client does specify
         facet=true in the request, these are the only facets they
         will be able to see counts for; regardless of what other
         facet.field or facet.query params they may specify.

         NOTE: there is *absolutely* nothing a client can do to prevent th
ese
         "invariants" values from being used, so don't use this mechanism
         unless you are sure you always want it.
      -->
    <!--
       <lst name="invariants">
         <str name="facet.field">cat</str>
         <str name="facet.field">manu_exact</str>
         <str name="facet.query">price:[* TO 500]</str>
         <str name="facet.query">price:[500 TO *]</str>
       </lst>
      -->
    <!-- If the default list of SearchComponents is not desired, that
         list can either be overridden completely, or components can be
         prepended or appended to the default list.  (see below)
      -->
    <!--
       <arr name="components">
         <str>nameOfCustomComponent1</str>
         <str>nameOfCustomComponent2</str>
       </arr>
      -->
      <arr name="first-components">
        <str>queryDocAuthorization</str>
      </arr>
  </requestHandler>

  <!-- A request handler that returns indented JSON by default -->
  <requestHandler name="/query" class="solr.SearchHandler">
    <lst name="defaults">
      <str name="echoParams">explicit</str>
      <str name="wt">json</str>
      <str name="indent">true</str>
    </lst>
    <arr name="first-components">
        <str>queryDocAuthorization</str>
    </arr>
  </requestHandler>

  <initParams path="/update/**,/query,/select,/spell">
    <lst name="defaults">
      <str name="df">_text_</str>
    </lst>
  </initParams>

  <!-- Search Components

       Search components are registered to SolrCore and used by
       instances of SearchHandler (which can access them by name)

       By default, the following components are available:

       <searchComponent name="query"     class="solr.QueryComponent" />
       <searchComponent name="facet"     class="solr.FacetComponent" />
       <searchComponent name="mlt"       class="solr.MoreLikeThisComponen
t" />
       <searchComponent name="highlight" class="solr.HighlightComponent" />
```

```
       <searchComponent name="stats"     class="solr.StatsComponent" />
       <searchComponent name="debug"     class="solr.DebugComponent" />

       Default configuration in a requestHandler would look like:

       <arr name="components">
         <str>query</str>
         <str>facet</str>
         <str>mlt</str>
         <str>highlight</str>
         <str>stats</str>
         <str>debug</str>
       </arr>

       If you register a searchComponent to one of the standard names,
       that will be used instead of the default.

       To insert components before or after the 'standard' components, use:

       <arr name="first-components">
         <str>myFirstComponentName</str>
       </arr>
       <arr name="last-components">
         <str>myLastComponentName</str>
       </arr>

       NOTE: The component registered with the name "debug" will
       always be executed after the "last-components"

     -->

  <searchComponent name="queryDocAuthorization" class="org.apache.ranger.a
uthorization.solr.authorizer.RangerSolrAuthorizer">
    <str name="enabled">true</str>
    <!-- The field which contains the role or list of roles which are allow
ed to query a particular document -->
    <str name="rangerAuthField">ranger_auth</str>
    <!-- If the rangerAuthField contains this value, all roles will be allo
wed to query that particular document -->
    <str name="allRolesToken">*</str>
  </searchComponent>

  <!-- Spell Check

       The spell check component can return a list of alternative spelling
       suggestions.

       http://wiki.apache.org/solr/SpellCheckComponent
    -->
  <searchComponent name="spellcheck" class="solr.SpellCheckComponent">
    <str name="queryAnalyzerFieldType">text_general</str>

    <!-- Multiple "Spell Checkers" can be declared and used by this
         component
      -->

    <!-- a spellchecker built from a field of the main index -->
    <lst name="spellchecker">
      <str name="name">default</str>
      <str name="field">_text_</str>
      <str name="classname">solr.DirectSolrSpellChecker</str>
      <!-- the spellcheck distance measure used, the default is the internal
 levenshtein -->
      <str name="distanceMeasure">internal</str>
```

```
      <!-- minimum accuracy needed to be considered a valid spellcheck su
ggestion -->
      <float name="accuracy">0.5</float>
      <!-- the maximum #edits we consider when enumerating terms: can be 1
 or 2 -->
      <int name="maxEdits">2</int>
      <!-- the minimum shared prefix when enumerating terms -->
      <int name="minPrefix">1</int>
      <!-- maximum number of inspections per result. -->
      <int name="maxInspections">5</int>
      <!-- minimum length of a query term to be considered for correction --
>
      <int name="minQueryLength">4</int>
      <!-- maximum threshold of documents a query term can appear to be co
nsidered for correction -->
      <float name="maxQueryFrequency">0.01</float>
      <!-- uncomment this to require suggestions to occur in 1% of the do
cuments
        <float name="thresholdTokenFrequency">.01</float>
      -->
    </lst>

    <!-- a spellchecker that can break or combine words.  See "/spell" handl
er below for usage -->
    <!--
    <lst name="spellchecker">
      <str name="name">wordbreak</str>
      <str name="classname">solr.WordBreakSolrSpellChecker</str>
      <str name="field">name</str>
      <str name="combineWords">true</str>
      <str name="breakWords">true</str>
      <int name="maxChanges">10</int>
    </lst>
    -->
  </searchComponent>

  <!-- A request handler for demonstrating the spellcheck component.

       NOTE: This is purely as an example.  The whole purpose of the
       SpellCheckComponent is to hook it into the request handler that
       handles your normal user queries so that a separate request is
       not needed to get suggestions.
       IN OTHER WORDS, THERE IS REALLY GOOD CHANCE THE SETUP BELOW IS
       NOT WHAT YOU WANT FOR YOUR PRODUCTION SYSTEM!
       See http://wiki.apache.org/solr/SpellCheckComponent for details
       on the request parameters.
    -->
  <requestHandler name="/spell" class="solr.SearchHandler" startup="lazy">
    <lst name="defaults">
      <!-- Solr will use suggestions from both the 'default' spellchecker
           and from the 'wordbreak' spellchecker and combine them.
           collations (re-written queries) can include a combination of
           corrections from both spellcheckers -->
      <str name="spellcheck.dictionary">default</str>
      <str name="spellcheck">on</str>
      <str name="spellcheck.extendedResults">true</str>
      <str name="spellcheck.count">10</str>
      <str name="spellcheck.alternativeTermCount">5</str>
      <str name="spellcheck.maxResultsForSuggest">5</str>
      <str name="spellcheck.collate">true</str>
      <str name="spellcheck.collateExtendedResults">true</str>
      <str name="spellcheck.maxCollationTries">10</str>
      <str name="spellcheck.maxCollations">5</str>
    </lst>
```

```
    <arr name="last-components">
      <str>spellcheck</str>
    </arr>
  </requestHandler>
  <!-- Terms Component
       http://wiki.apache.org/solr/TermsComponent

       A component to return terms and document frequency of those
       terms
    -->
  <searchComponent name="terms" class="solr.TermsComponent"/>

  <!-- A request handler for demonstrating the terms component -->
  <requestHandler name="/terms" class="solr.SearchHandler" startup="lazy">
    <lst name="defaults">
      <bool name="terms">true</bool>
      <bool name="distrib">false</bool>
    </lst>
    <arr name="components">
      <str>terms</str>
    </arr>
  </requestHandler>

  <!-- Highlighting Component
       http://wiki.apache.org/solr/HighlightingParameters
    -->
  <searchComponent class="solr.HighlightComponent" name="highlight">
    <highlighting>
      <!-- Configure the standard fragmenter -->
      <!-- This could most likely be commented out in the "default" case -->
      <fragmenter name="gap"
                  default="true"
                  class="solr.highlight.GapFragmenter">
        <lst name="defaults">
          <int name="hl.fragsize">100</int>
        </lst>
      </fragmenter>
      <!-- A regular-expression-based fragmenter
           (for sentence extraction)
        -->
      <fragmenter name="regex"
                  class="solr.highlight.RegexFragmenter">
        <lst name="defaults">
          <!-- slightly smaller fragsizes work better because of slop -->
          <int name="hl.fragsize">70</int>
          <!-- allow 50% slop on fragment sizes -->
          <float name="hl.regex.slop">0.5</float>
          <!-- a basic sentence pattern -->
          <str name="hl.regex.pattern">[-\w ,/\n\&quot;&apos;]{20,200}</str>
        </lst>
      </fragmenter>

      <!-- Configure the standard formatter -->
      <formatter name="html"
                 default="true"
                 class="solr.highlight.HtmlFormatter">
        <lst name="defaults">
          <str name="hl.simple.pre">&lt;![CDATA[&lt;em&gt;]]&gt;</str>
          <str name="hl.simple.post">&lt;![CDATA[&lt;/em&gt;]]&gt;</str>
        </lst>
      </formatter>

      <!-- Configure the standard encoder -->
      <encoder name="html"
```

```
                class="solr.highlight.HtmlEncoder" />

      <!-- Configure the standard fragListBuilder -->
      <fragListBuilder name="simple"
                       class="solr.highlight.SimpleFragListBuilder"/>

      <!-- Configure the single fragListBuilder -->
      <fragListBuilder name="single"
                       class="solr.highlight.SingleFragListBuilder"/>

      <!-- Configure the weighted fragListBuilder -->
      <fragListBuilder name="weighted"
                       default="true"
                       class="solr.highlight.WeightedFragListBuilder"/>

      <!-- default tag FragmentsBuilder -->
      <fragmentsBuilder name="default"
                        default="true"
                        class="solr.highlight.ScoreOrderFragmentsBuilder">
        <!--
        <lst name="defaults">
          <str name="hl.multiValuedSeparatorChar">/</str>
        </lst>
        -->
      </fragmentsBuilder>

      <!-- multi-colored tag FragmentsBuilder -->
      <fragmentsBuilder name="colored"
                        class="solr.highlight.ScoreOrderFragmentsBuilder">
        <lst name="defaults">
          <str name="hl.tag.pre"><![CDATA[
               <b style="background:yellow">,<b style="background:lawgreen"
>,
               <b style="background:aquamarine">,<b style="background:mag
enta">,
               <b style="background:palegreen">,<b style="background:coral">
,
               <b style="background:wheat">,<b style="background:khaki">,
               <b style="background:lime">,<b style="background:deepskyblue"
>]]></str>
          <str name="hl.tag.post">&lt;![CDATA[&lt;/b&gt;]]&gt;</str>
        </lst>
      </fragmentsBuilder>

      <boundaryScanner name="default"
                       default="true"
                       class="solr.highlight.SimpleBoundaryScanner">
        <lst name="defaults">
          <str name="hl.bs.maxScan">10</str>
          <str name="hl.bs.chars">.,!? &#9;&#10;&#13;</str>
        </lst>
      </boundaryScanner>

      <boundaryScanner name="breakIterator"
                       class="solr.highlight.BreakIteratorBoundaryScanner">
        <lst name="defaults">
          <!-- type should be one of CHARACTER, WORD(default), LINE and SEN
TENCE -->
          <str name="hl.bs.type">WORD</str>
          <!-- language and country are used when constructing Locale obje
ct.  -->
          <!-- And the Locale object will be used when getting instance of
BreakIterator -->
          <str name="hl.bs.language">en</str>
```

```
          <str name="hl.bs.country">US</str>
        </lst>
      </boundaryScanner>
    </highlighting>
  </searchComponent>

  <!-- Update Processors
       Chains of Update Processor Factories for dealing with Update
       Requests can be declared, and then used by name in Update
       Request Processors
       http://wiki.apache.org/solr/UpdateRequestProcessor

    -->
  <!-- Add unknown fields to the schema

       Field type guessing update processors that will
       attempt to parse string-typed field values as Booleans, Longs,
       Doubles, or Dates, and then add schema fields with the guessed
       field types. Text content will be indexed as "text_general" as
       well as a copy to a plain string version in *_str.

       These require that the schema is both managed and mutable, by
       declaring schemaFactory as ManagedIndexSchemaFactory, with
       mutable specified as true.

       See http://wiki.apache.org/solr/GuessingFieldTypes
    -->
  <updateProcessor class="solr.UUIDUpdateProcessorFactory" name="uuid"/>
  <updateProcessor class="solr.RemoveBlankFieldUpdateProcessorFactory" na
me="remove-blank"/>
  <updateProcessor class="solr.FieldNameMutatingUpdateProcessorFactory" name
="field-name-mutating">
    <str name="pattern">[^\w-\.]</str>
    <str name="replacement">_</str>
  </updateProcessor>
  <updateProcessor class="solr.ParseBooleanFieldUpdateProcessorFactory" name
="parse-boolean"/>
  <updateProcessor class="solr.ParseLongFieldUpdateProcessorFactory" name
="parse-long"/>
  <updateProcessor class="solr.ParseDoubleFieldUpdateProcessorFactory" name=
"parse-double"/>
  <updateProcessor class="solr.ParseDateFieldUpdateProcessorFactory" name="
parse-date">
    <arr name="format">
      <str>yyyy-MM-dd['T'[HH:mm[:ss[.SSS]][z</str>
      <str>yyyy-MM-dd['T'[HH:mm[:ss[,SSS]][z</str>
      <str>yyyy-MM-dd HH:mm[:ss[.SSS]][z</str>
      <str>yyyy-MM-dd HH:mm[:ss[,SSS]][z</str>
      <str>[EEE, ]dd MMM yyyy HH:mm[:ss] z</str>
      <str>EEEE, dd-MMM-yy HH:mm:ss z</str>
      <str>EEE MMM ppd HH:mm:ss [z ]yyyy</str>
    </arr>
  </updateProcessor>
  <updateProcessor class="solr.AddSchemaFieldsUpdateProcessorFactory" name=
"add-schema-fields">
    <lst name="typeMapping">
      <str name="valueClass">java.lang.String</str>
      <str name="fieldType">text_general</str>
      <lst name="copyField">
        <str name="dest">*_str</str>
        <int name="maxChars">256</int>
      </lst>
      <!-- Use as default mapping instead of defaultFieldType -->
      <bool name="default">true</bool>
```

```
    </lst>
    <lst name="typeMapping">
      <str name="valueClass">java.lang.Boolean</str>
      <str name="fieldType">booleans</str>
    </lst>
    <lst name="typeMapping">
      <str name="valueClass">java.util.Date</str>
      <str name="fieldType">pdates</str>
    </lst>
    <lst name="typeMapping">
      <str name="valueClass">java.lang.Long</str>
      <str name="valueClass">java.lang.Integer</str>
      <str name="fieldType">plongs</str>
    </lst>
    <lst name="typeMapping">
      <str name="valueClass">java.lang.Number</str>
      <str name="fieldType">pdoubles</str>
    </lst>
  </updateProcessor>

  <!-- The update.autoCreateFields property can be turned to false to dis
able schemaless mode -->
  <updateRequestProcessorChain name="add-unknown-fields-to-the-schema" defa
ult="${update.autoCreateFields:true}"
           processor="uuid,remove-blank,field-name-mutating,parse-boolean,
parse-long,parse-double,parse-date,add-schema-fields">
    <processor class="solr.LogUpdateProcessorFactory"/>
    <processor class="solr.DistributedUpdateProcessorFactory"/>
    <processor class="solr.RunUpdateProcessorFactory"/>
  </updateRequestProcessorChain>
  <!-- Deduplication

       An example dedup update processor that creates the "id" field
       on the fly based on the hash code of some other fields.  This
       example has overwriteDupes set to false since we are using the
       id field as the signatureField and Solr will maintain
       uniqueness based on that anyway.
    -->
  <!--
     <updateRequestProcessorChain name="dedupe">
       <processor class="solr.processor.SignatureUpdateProcessorFactory">
         <bool name="enabled">true</bool>
         <str name="signatureField">id</str>
         <bool name="overwriteDupes">false</bool>
         <str name="fields">name,features,cat</str>
         <str name="signatureClass">solr.processor.Lookup3Signature</str>
       </processor>
       <processor class="solr.LogUpdateProcessorFactory" />
       <processor class="solr.RunUpdateProcessorFactory" />
     </updateRequestProcessorChain>
    -->

  <!-- Response Writers

       http://wiki.apache.org/solr/QueryResponseWriter

       Request responses will be written using the writer specified by
       the 'wt' request parameter matching the name of a registered
       writer.

       The "default" writer is the default and will be used if 'wt' is
       not specified in the request.
    -->
  <!-- The following response writers are implicitly configured unless
```

```
       overridden...
    -->
  <!--
     <queryResponseWriter name="xml"
                          default="true"
                          class="solr.XMLResponseWriter" />
     <queryResponseWriter name="json" class="solr.JSONResponseWriter"/>
     <queryResponseWriter name="python" class="solr.PythonResponseWriter"/>
     <queryResponseWriter name="ruby" class="solr.RubyResponseWriter"/>
     <queryResponseWriter name="php" class="solr.PHPResponseWriter"/>
     <queryResponseWriter name="phps" class="solr.PHPSerializedResponseWri
ter"/>
     <queryResponseWriter name="csv" class="solr.CSVResponseWriter"/>
     <queryResponseWriter name="schema.xml" class="solr.SchemaXmlResponseWr
iter"/>
    -->
  <queryResponseWriter name="json" class="solr.JSONResponseWriter">
    <!-- For the purposes of the tutorial, JSON responses are written as
     plain text so that they are easy to read in *any* browser.
     If you expect a MIME type of "application/json" just remove this over
ride.
    -->
    <str name="content-type">text/plain; charset=UTF-8</str>
  </queryResponseWriter>
  <!-- Query Parsers

       https://lucene.apache.org/solr/guide/query-syntax-and-parsing.html

       Multiple QParserPlugins can be registered by name, and then
       used in either the "defType" param for the QueryComponent (used
       by SearchHandler) or in LocalParams
    -->
  <!-- example of registering a query parser -->
  <!--
     <queryParser name="myparser" class="com.mycompany.MyQParserPlugin"/>
    -->

  <!-- Function Parsers

       http://wiki.apache.org/solr/FunctionQuery
       Multiple ValueSourceParsers can be registered by name, and then
       used as function names when using the "func" QParser.
    -->
  <!-- example of registering a custom function parser  -->
  <!--
     <valueSourceParser name="myfunc"
                        class="com.mycompany.MyValueSourceParser" />
    -->

  <!-- Document Transformers
       http://wiki.apache.org/solr/DocTransformers
    -->
  <!--
     Could be something like:
     <transformer name="db" class="com.mycompany.LoadFromDatabaseTransform
er" >
       <int name="connection">jdbc://....</int>
     </transformer>

     To add a constant value to all docs, use:
     <transformer name="mytrans2" class="org.apache.solr.response.transfor
m.ValueAugmenterFactory" >
       <int name="value">5</int>
     </transformer>
```

```
     If you want the user to still be able to change it with _value:somethi
ng_ use this:
     <transformer name="mytrans3" class="org.apache.solr.response.transfor
m.ValueAugmenterFactory" >
       <double name="defaultValue">5</double>
     </transformer>

      If you are using the QueryElevationComponent, you may wish to mark d
ocuments that get boosted.  The
      EditorialMarkerFactory will do exactly that:
     <transformer name="qecBooster" class="org.apache.solr.response.trans
form.EditorialMarkerFactory" />
    -->
</config>
```

Cloudera Runtime 7.2.18

Cloudera Search solrctl Reference

Date published: 2020-06-01
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

solrctl Reference.....4

Using solrctl with an HTTP proxy..... 11

solrctl Reference

The solrctl utility is a wrapper shell script included with Cloudera Search for managing collections, instance directories, configs, and more.

Make sure that the host on which you are running the solrctl utility has either a Gateway or Solr Server role assigned.

In general, if an operation succeeds, solrctl exits silently with a success exit code. If an error occurs, solrctl prints a diagnostics message combined with a failure exit code. solrctl supports specifying a log4j.properties file by setting the LOG4J_PROPS environment variable. By default, the LOG4J_PROPS setting specifies the log4j.properties in the Solr configuration directory (for example, /etc/solr/conf/log4j.properties). Many solrctl commands redirect stderr to /dev/null, so Cloudera recommends that your log4j properties file specify a location other than stderr for log output.

You can run solrctl on any host that is configured as part of the SolrCloud deployment (the Solr service in Cloudera Manager environments) . To run any solrctl command on a host outside of SolrCloud deployment, ensure that SolrCloud hosts are reachable and provide --zk and --solr command line options.

If you are using solrctl to manage your deployment in an environment that requires Kerberos authentication, you must have a valid Kerberos ticket, which you can get using kinit.

For collection configuration, users have the option of interacting directly with ZooKeeper using the instancedir option or using the Solr ConfigSets API using the config option. For more information, see Managing Configuration Using Configs or Instance Directories.

Command syntax

The general solrctl command syntax is:

solrctl [options] command [command-arg] [command [command-arg]] ...

Each element and its possible values are described in the following sections.

solrctl options

If used, the following options must precede commands:

Table 1: solrctl options

Option	Description
--solr *<SOLR_URI>*	Directs solrctl to a SolrCloud web API available at the specified URI. This option is required for hosts running outside of SolrCloud. A sample URI might be: http://search01.example.com:8983/solr.
--zk *<ZK_ENSEMBLE>*	Directs solrctl to a particular ZooKeeper quorum. This option is required for hosts running outside of SolrCloud. For example: zk01.example.com:2181,zk02.example.com:2181,zk03.example.com:2181/solr. Output from solrctl commands that use the --zk option is sent to /dev/null, so no results are displayed.
--jaas */PATH/TO/JAAS.CONF*	Used to identify a JAAS configuration that specifies the principal with permissions to modify Solr metadata. The principal is typically solr@EXAMPLE.COM. In Kerberos-enabled environments where ZooKeeper ACLs protect Solr metadata, you must use this parameter when modifying metadata.
--help	Prints help.
--quiet	Suppresses most solrctl messages.
--debug	Prints errors to stdout.
--debug-dump	Prints the current state of ZooKeeper
--trace	Prints the executed commands to stdout.

Commands and arguments

The solrctl commands init, instancedir, config, collection and cluster affect the entire SolrCloud deployment and only need to be run once per required operation.

The solrctl core command affects a single SolrCloud host.

Table 2: solrctl commmands and arguments

Command	Argument	Description
init		Initializes a SolrCloud deployment. Must be run before starting solr-server daemons for the first time. The command has a built-in security check that prevents it from running on a deployment that has already been initialized.
	[--force]	Allows you to re-initialize an already initialized SolrCloud deployment. Use this command cautiously because it erases all SolrCloud deployment state information from ZooKeeper, including all configuration files. It does not delete collections.
instancedir		Manipulates instance directories. **Note:** solrctl instancedir commands talk directly to ZooKeeper. Starting from CDH version 6.0, the ZooKeeper node that stores the instance configuration files (e.g.: /solr/configs) is only writable by the user 'solr'. This means that in a Kerberized environment you will have to create a Java Authentication and Authorization Service (JAAS) configuration file and a keytab for the solr user, and specifying this jaas.conf file when using solrctl. For example: solrctl --jaas jaas.conf instancedir --create myInstanceDir /tmp/myInstanceDir. Cloudera recommends using the solrctl config --upload command instead, as it does not require a keytab of the solr user (a simple kinit is enough) and additionally it can utilize Ranger to control who is authorized to modify instance configurations.
	--generate <*PATH*> [-schemaless] [-localfs]	--generate <*PATH*>: Generates an instance directory template on the local filesystem at <*PATH*>. The configuration files are located in the conf subdirectory under <*PATH*>. • If used with the -schemaless option, it generates a schemaless instance directory template. For more information on schemaless support, see Schemaless Mode Overview and Best Practices. • If used with the -localfs option, the default HdfsDirectoryFactory is overwritten with NRTCachingDirectoryFactory in the generated solrconfig.xml. The index of a collection using this configuration is written to the local file system, not to HDFS.

Command	Argument	Description
	--create *<NAME>* *<PATH>*	Uploads a copy of the instance directory from *<PATH>* on the local filesystem to ZooKeeper. If an instance directory with the specified *<NAME>* already exists, this command fails. Use --update to modify existing instance directories.
	--update *<NAME>* *<PATH>*	Overwrites an existing instance directory in ZooKeeper using the specified files on the local filesystem. This command is analogous to first running --delete *<NAME>* followed by --create *<NAME>* *<PATH>*.
	--get *<NAME>* *<PATH>*	Downloads the specified instance directory from ZooKeeper to the specified path on the local filesystem. You can then edit the configuration and then re-upload it using --update.
	--delete *<NAME>*	Deletes the specified instance directory from ZooKeeper.
	--list	Lists existing instance directories, including configs created by the solrctl config command.
config		Manipulates configs.
	--create name *<BASECONFIG>* [-p *<NAME>*=*<VALUE>*	Creates a new config based on an existing config. The config is created with the specified *NAME*, using *<BASECONFIG>* as the template. For more information about config templates, see Config Templates. The -p name=value option overrides a *<BASECONFIG>* setting. The only config property that you can override is immutable, so the possible options are -p immutable=true and -p immutable=false. If you are copying an immutable config, such as a template, use -p immutable=false to make sure that you can edit the new config.
	--upload *NAME* *PATH*	Uploads a new configset in zip file format. You cannot use this option to update an existing config. The script will ask you to delete the existing version before allowing you to upload the new one. The *PATH* argument of this command needs to point to the local directory containing the instance configuration (meaning it has a conf subdirectory and the config files like conf/solrconfig.xml). This can also be an instance configuration directory generated using solrctl instancedir --generate *NAME* or downloaded using solrctl instancedir --get *NAME* *PATH*. The underlying Solr API requires a .zip archive to be created, this is automatically performed by the command. **Note:** This function needs the zip binary to be installed and executable by the user running the solrctl config --upload command.
	--delete name	Deletes the specified config. You cannot delete an immutable config without accessing ZooKeeper directly as the solr super user.
collection		Manipulates collections.

Command	Argument	Description
	--create *<NAME>* -s *<NUMSHARDS>* [-a] [-c *<CONFIGNAME>*] [-r *<REPLICATIONFACTOR>*] [-m *<MAXSHARDSPERHOST>*] [-n *<HOSTLIST>*]]	Creates a new collection with *<NUMSHARDS>* shards. The -a option enables automatic addition of replicas (autoAddReplicas=true) if machines hosting existing shards become unavailable. The collection uses the specified *<CONFIGNAME>* for its configuration set, and the specified *<REPLICATIONFACTOR>* controls the number of replicas for the collection. Keep in mind that this replication factor is on top of the HDFS replication factor. The maximum shards per host is determined by *<MAXSHARDSPERHOST>*, and you can specify specific hosts for the collection in the *<HOSTLIST>*. The only required parameters are *<NAME>* and -s *<NUMSHARDS>*. If -c *<CONFIGNAME>* is not provided, it is assumed to be the same as the name of the collection.
	--delete *<NAME>*	Deletes a collection.
	--reload *<NAME>*	Reloads a collection.
	--stat *<NAME>*	Outputs SolrCloud specific run-time information for a collection.
	--deletedocs *<NAME>*	Purges all indexed documents from a collection.
	--list	Lists all collections.
	--create-snapshot *<SNAPSHOTNAME>* -c *<COLLECTIONNAME>*	Creates a named snapshot for the specified collection.
	--delete-snapshot *<SNAPSHOTNAME>* -c *<COLLECTIONNAME>*	Deletes the specified snapshot for the specified collection.
	--describe-snapshot *<SNAPSHOTNAME>* -c *<COLLECTIONNAME>*	Provides detailed information about a snapshot for the specified collection.
	--list-snapshots *<COLLECTIONNAME>*	Lists all snapshots for the specified collection.

Command	Argument	Description
	--prepare-snapshot-export *<SNAPSHOTNAME>* -c *<COLLECTIONNAME>* -d *<DESTDIR>*	Prepares the snapshot for export to a remote cluster. If you are exporting the snapshot to the local cluster, you do not need to run this command. This command generates collection metadata as well as information about the Lucene index files corresponding to the snapshot. The destination HDFS directory path (specified by the -d option) must exist on the local cluster before you run this command. Make sure that the Solr superuser (solr by default) has permission to write to this directory. If you are running the snapshot export command on a remote cluster, specify the HDFS protocol (such as WebHDFS or HFTP) to be used for accessing the Lucene index files corresponding to the snapshot on the source cluster. This configuration is driven by the -p option which expects a fully qualified URI for the root filesystem on the source cluster, for example webhdfs://namenode.example.com:20101/. **Note:** Creating backups and performing restore operations using the solr ctl CLI tool exclusively works on HDFS. If you want to create a backup to, or perform a restore from a different FS, consult the Apache Solr documentation.
	--export-snapshot *<SNAPSHOTNAME>* -c *<COLLECTIONNAME>*|-s <sourceDir> -d *<DESTDIR>*	Creates a backup copy of the Solr collection metadata as well as the associated Lucene index files at the specified location. If exporting to a local cluster, use the -c *<COLLECTIONNAME>* argument. If exporting to a remote cluster, use the -s *<SOURCEDIR>* argument. The -d configuration option specifies the directory path where this backup copy is be created. This directory must exist before exporting the snapshot, and the Solr superuser (solr, by default) must be able to write to it. **Note:** Creating backups and performing restore operations using the solr ctl CLI tool exclusively works on HDFS. If you want to create a backup to, or perform a restore from a different FS, consult the Apache Solr documentation.

Command	Argument	Description
	--restore <*RESTORECOLLECTIONNAME*> -l <*BACKUPLOCATION*> -b <*SNAPSHOTNAME*> -i <*REQUESTID*> [-a] [-c <*CONFIGNAME*>] [-r <*REPLICATIONFACTOR*>] [-m <*MAXSHARDSPERHOST*>] [-n <*SELECTEDNODES*>]	Restores the state of an earlier created backup as a new Solr collection. Run this command on the cluster on which you want to restore the backup. The -l configuration option specifies the local HDFS directory where the backup is stored. If the backup is stored on a remote cluster, you must copy it to the local cluster before restoring it. The Solr superuser (solr by default) must have permission to read from this directory. The -b configuration option specifies the name of the backup to be restored. Because the restore operation can take a long time to complete depending on the size of the exported snapshot, it is run asynchronously. The -i configuration parameter specifies a unique identifier for tracking the operation. The optional -a configuration option enables the autoAddReplicas feature for the new Solr collection. The optional -c configuration option specifies the configName for the new Solr collection. If this option is not specified, the configNa me of the original collection at the time of backup is used. If the specified configName does not exist, the restore operation creates a new configuration from the backup. The optional -r configuration option specifies the replication factor for the new Solr collection. If this option is not specified, the replication factor of the original collection at the time of backup is used. The optional -m configuration option specifies the maximum number of replicas (maxShard sPerNode) to create on each Solr Server. If this option is not specified, the maxShardsPerNode configuration of the original collection at the time of backup is used. The optional -n configuration option specifies the nodes to spread the restored collection across. If this option is not specified, the operation will create shard-replica spread across all live Solr nodes. **Note:** Creating backups and performing restore operations using the solr ctl CLI tool exclusively works on HDFS. If you want to create a backup to, or perform a restore from a different FS, consult the Apache Solr documentation.

Command	Argument	Description
	--readonly *NAME*	Puts the collection in read-only mode, in which any index update requests are rejected. Other collection-level actions (e.g., adding / removing / moving replicas) are still available in this mode. The transition from the (default) read-write to read-only mode consists of the following steps: • the readOnly flag is changed in collection state, • any new update requests are rejected with 403 FORBIDDEN error code (ongoing long-running requests are aborted, too), • a forced commit is performed to flush and commit any in-flight updates. **Note:** This may potentially take a long time if there are still major segment merges running in the background. • a collection RELOAD action is executed. Issuing the solrctl collection --readwrite *NAME* command enables the processing of updates and reloads the collection.
	--readwrite *NAME*	puts the collection in the default read-write mode, in which any index update requests are allowed.
	--request-status <*REQUESTID*>	Displays the status of the specified operation. The status can be one of the following: • running: The restore operation is running. • completed: The restore operation is complete. • failed: The restore operation failed. • notfound: The specified requestID is not found.
core		Manipulates cores.
	--create <*NAME*> [-p <*NAME*>=<*VALUE*>]	Creates a new core. The core is configured using <*NAME*>=<*VALUE*> pairs. For more information about configuration options, see Solr documentation
	--reload <*NAME*>	Reloads a core.
	--unload <*NAME*>	Unloads a core.
	--status <*NAME*>	Prints the status of a core.
cluster		Manages cluster configuration.
	--get-solrxml <*FILE*>	Downloads the cluster configuration file solr.xml from ZooKeeper to the local system.
	--put-solrxml <*FILE*>	Uploads the specified file to ZooKeeper as the cluster configuration file solr.xml.
	--set-property <*NAME*> <*VALUE*>	Sets property names and values. For example, to configure a cluster to use TLS/SSL: `solrctl cluster --set-property urlScheme https`

Command	Argument	Description
	--remove-property <*NAME*>	Removes the specified property.
	--get-clusterstate <*FILE*>	Downloads the clusterstate.json file from ZooKeeper to the local system.

Using solrctl with an HTTP proxy

About this task

Using solrctl to manage a deployment in an environment that uses an HTTP proxy fails because solrctl uses curl, which attempts to use the proxy. You can disable the proxy so solrctl succeeds:

Procedure

- Modify the settings for the current shell by exporting the NO_PROXY environment variable. For example:

```
export NO_PROXY='*'
```

- Modify the settings for single commands by prefacing solrctl with NO_PROXY='*'. For example:

```
NO_PROXY='*' solrctl collection --create <COLLECTIONNAME> -s 3
```

Cloudera Runtime 7.2.18

Troubleshooting Cloudera Search

Date published: 2019-11-19
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Identifying problems in a Cloudera Search deployment.....4

Troubleshooting Cloudera Search.....6

Identifying problems in a Cloudera Search deployment

Learn about common issues affecting Search performance and what you can do about them.

To investigate your Cloudera Search deployment for performance problems or misconfigurations, inspect the log files, schema files, and the actual index for issues. If possible, connect to the live Solr instance while watching log files so you can compare the schema with the index. For example, the schema and the index can be out of sync in situations where the schema is changed, but the index was never rebuilt.

Issue	Recommendation
A high number or proportion of 0-match queries	This indicates that the user-facing part of the application is making it easy for users to enter queries for which there are no matches. In Cloudera Search, given the size of the data, this should be an extremely rare event.
Queries that match an excessive number of documents	All documents that match a query have to be scored, and the cost of scoring a query goes up as the number of finds increases. Examine any frequent queries that match millions of documents. An exception to this case is “constant score queries”. Queries, such as those of the form ":" bypass the scoring process entirely.
Overly complex queries	Defining what constitutes overly complex queries is difficult to do, but a very general rule is that queries over 1024 characters in length are likely to be overly complex.
High autowarm times	Autowarming is the process of filling caches. Some queries are run before a new searcher serves the first live user request. This keeps the first few users from having to wait. Autowarming can take many seconds or can be instantaneous. Excessive autowarm times often indicate excessively generous autowarm parameters. Excessive autowarming usually has limited benefit, with longer runs effectively being wasted work. • Cache autowarm. Each Solr cache has an autowarm parameter. You can usually set this value to an upper limit of 128 and tune from there. • FirstSearcher/NewSearcher. The solrconfig.xml file contains queries that can be fired when a new searcher is opened (the index is updated) and when the server is first started. Particularly for firs tSearcher, it can be valuable to have a query that sorts relevant fields. **Note:** The aforementioned flags are available from solrconf ig.xml
Exceptions	The Solr log file contains a record of all exceptions thrown. Some exceptions, such as exceptions resulting from invalid query syntax are benign, but others, such as Out Of Memory, require attention.
Excessively large caches	The size of caches such as the filter cache are bounded by maxDoc/8. Having, for instance, a filterCache with 10,000 entries is likely to result in Out Of Memory errors. Large caches occurring in cases where there are many documents to index is normal and expected.

Issue	Recommendation
Caches with low find (hit) ratios, particularly filterCache	Each cache takes up some space, consuming resources. There are several caches, each with its own rate of finding stored information. • filterCache. This cache should have a relatively high ratio of finding stored information (hit ratio), typically around 80%. • queryResultCache. This is primarily used for paging so it can have a very low ratio of finding stored information (hit ratio). Each entry is quite small as it is basically composed of the raw query as a string for a key and perhaps 20-40 ints. While useful, unless users are experiencing paging, this requires relatively little attention. • documentCache. This cache is a bit tricky. It's used to cache the document data (stored fields) so various components in a request handler don't have to re-read the data from the disk. It's an open question how useful it is when using MMapDirectory to access the index.
Very deep paging	Users seldom go beyond the first page and very rarely to go through 100 pages of results. A &start=<pick your number> query indicates unusual usage that should be identified. Deep paging may indicate some agent is completing scraping. **Note:** Solr is not built to return full result sets no matter how deep. If returning the full result set is required, explore alternatives to paging through the entire result set.
Range queries should work on trie fields	Trie fields (numeric types) store extra information in the index to aid in range queries. If range queries are used, it's almost always a good idea to be using trie fields.
fq clauses that use bare NOW	fq clauses are kept in a cache. The cache is a map from the fq clause to the documents in your collection that satisfy that clause. Using bare NOW clauses virtually guarantees that the entry in the filter cache is not to be re-used.
Multiple simultaneous searchers warming	This is an indication that there are excessively frequent commits or that autowarming is taking too long. This usually indicates a misunderstanding of when you should issue commits, often to simulate Near Real Time (NRT) processing or an indexing client is improperly completing commits. With NRT, commits should be quite rare, and having more than one simultaneous autowarm should not happen.
Stored fields that are never returned (fl= clauses)	Examining the queries for fl= and correlating that with the schema can tell if stored fields that are not used are specified. This mostly wastes disk space. And fl=* can make this ambiguous. Nevertheless, it's worth examining.
Indexed fields that are never searched	This is the opposite of the case where stored fields are never returned. This is more important in that this has real RAM consequences. Examine the request handlers for "edismax" style parsers to be certain that indexed fields are not used.
Queried but not analyzed fields	It's rare for a field to be queried but not analyzed in any way. Usually this is only valuable for "string" type fields which are suitable for machine-entered data, such as part numbers chosen from a pick-list. Data that is not analyzed should not be used for anything that humans enter.
String fields	String fields are completely unanalyzed. Unfortunately, some people confuse string with Java's String type and use them for text that should be tokenized. The general expectation is that string fields should be used sparingly. More than just a few string fields indicates a design flaw.

Issue	Recommendation
The schema and the index are out of sync	Solr uses the schema to set expectations about the index. When schemas are changed, there's no attempt to retrofit the changes to documents that are currently indexed, but any new documents are indexed with the new schema definition. So old and new documents can have the same field stored in vastly different formats (for example, String and TrieDate) making your index inconsistent. This can be detected by examining the raw index. Whenever the schema is changed, re-index the entire data set.
Query stats	Query stats can be extracted from the logs. Statistics can be monitored on live systems, but it is more common to have log files. Here are some of the statistics you can gather: • Longest running queries • 0-length queries • average/mean/min/max query times • You can get a sense of the effects of commits on the subsequent queries over some interval (time or number of queries) to see if commits are the cause of intermittent slowdowns
Unsatisfactory performance	Too-frequent commits have historically been the cause of unsatisfactory performance. This is not so important with NRT processing, but it is valuable to consider.

Troubleshooting Cloudera Search

After installing and deploying Cloudera Search, use the information in this section to troubleshoot problems.

Troubleshooting

The following table contains some common troubleshooting techniques.

Note:

In the URLs in the following table, you must replace entries such as <server:port> with values from your environment. The port defaults value is 8983, but see /etc/default/solr or /opt/cloudera/parcels/CDH-*/etc/default/solr for the port if you are in doubt.

Symptom	Explanation	Recommendation
All	Varied	Examine Solr log. By default, the log can be found at /var/log/solr/solr.out.
No documents found	Server may not be running	Browse to http://*SERVER*:*PORT*/solr to see if the server responds. Check that cores are present. Check the contents of cores to ensure that numDocs is more than 0.
No documents found	Core may not have documents	Browsing http://server:port/solr/[collection name]/select?q=*:*&wt=json&indent=true should show numFound, which is near the top, to be more than 0.
The secure Solr Server fails to respond to Solrj requests, but other clients such as curl can communicate successfully	This may be a version compatibility issue. Httpclient 4.2.3, which ships with solrj in Search 1.x, has a dependency on commons-codec 1.7. If an earlier version of commons-codec is on the classpath, httpclient may be unable to communicate using Kerberos.	Ensure your application is using commons-codec 1.7 or higher. Alternatively, use httpclient 4.2.5 instead of version 4.2.3 in your application. Version 4.2.3 behaves correctly with earlier versions of commons-codec.

Dynamic Solr Analysis

Any JMX-aware application can query Solr for information and display results dynamically. For example, Zabbix, Nagios, and many others have been used successfully. When completing Static Solr Log Analysis, many of the items related to extracting data from the log files can be seen from querying Solr, at least the last value (as opposed to the history which is available from the log file). These are often useful for status boards. In general, anything available from the Solr admin page can be requested on a live basis from Solr. Some possibilities include:

- numDocs/maxDoc per core. This can be important since the difference between these numbers indicates the number of deleted documents in the index. Deleted documents take up disk space and memory. If these numbers vary greatly, this may be a rare case where optimizing is advisable.
- Cache statistics, including:
  - Ratios that measures how effective a cache is at fulfilling requests for content
  - Autowarm times
  - Evictions
- Almost anything available on the admin page. Note that drilling down into the “schema browser” can be expensive.

Other Troubleshooting Information

Since the use cases for Solr and search vary, there is no single solution for all cases. That said, here are some common challenges that many Cloudera Search users have encountered:

- Testing with unrealistic data sets. For example, a users may test a prototype that uses faceting, grouping, sorting, and complex schemas against a small data set. When this same system is used to load of real data, performance issues occur. Using realistic data and use-cases is essential to getting accurate results.
- If the scenario seems to be that the system is slow to ingest data, consider:
  - Upstream speed. If you have a SolrJ program pumping data to your cluster and ingesting documents at a rate of 100 docs/second, the gating factor may be upstream speed. To test for limitations due to upstream speed, comment out only the code that sends the data to the server (for example, SolrHttpServer.add(doclist)) and time the program. If you see a throughput bump of less than around 10%, this may indicate that your system is spending most or all of the time getting the data from the system-of-record.
  - This may require pre-processing.
  - Indexing with a single thread from the client. ConcurrentUpdateSolrServer can use multiple threads to avoid I/O waits.
  - Too-frequent commits. This was historically an attempt to get NRT processing, but with SolrCloud hard commits this should be quite rare.
  - The complexity of the analysis chain. Note that this is rarely the core issue. A simple test is to change the schema definitions to use trivial analysis chains and then measure performance.
  - When the simple approaches fail to identify an issue, consider using profilers.

Cloudera Runtime 7.2.18

Tuning Cloudera Search

Date published: 2020-01-27
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Solr server tuning categories **4**

Setting Java system properties for Solr **5**

Enable multi-threaded faceting **5**

Tuning garbage collection **6**

Enable garbage collector logging **6**

Solr and HDFS - the block cache **7**

Tuning replication **9**

Adjust the Solr replication factor for index files stored in HDFS 10

Solr server tuning categories

Solr performance tuning is a complex task. Learn about available tuning options that youcan perform either during deployment or at a later stage.

The following tuning categories can be completed either during deployment or at a later stage. It is less important to implement these changes before taking your system into use.

Table 1: Tuning steps

Tuning option	Description
Configure the Java heap size	Set the Java heap size for the Solr Server to at least 16 GB for production environments. For more information on memory requirements, see Deployment Planning for Cloudera Search.
Enable multi-threaded faceting	Enabling multi-threaded faceting can provide better performance for field faceting. It has no effect on query faceting.
Enable garbage collector (GC) logging	To help identify any garbage collector (GC) issues, enable GC logging in production. The overhead is low.
Configure garbage collection	Select the garbage collection option that offers best performance in your environment.
Configure index caching	Cloudera Search enables Solr to store indexes in an HDFS filesystem. To maintain performance, an HDFS block cache has been implemented using Least Recently Used (LRU) semantics. This enables Solr to cache HDFS index files on read and write, storing the portions of the file in JVM direct memory (off heap) by default, or optionally in the JVM heap.
Tune commit values	Changing commit values may improve performance in certain situations. These changes result in tradeoffs and may not be beneficial in all cases. • For hard commit values, the default value of 60000 (60 seconds) is typically effective, though changing this value to 120 seconds may improve performance in some cases. Note, that setting this value to higher values, such as 600 seconds may result in undesirable performance tradeoffs. • Consider increasing the auto-soft-commit value from 15000 (15 seconds) to 120000 (120 seconds). You may increase this to the largest value that still meets your requirements.
Tune sharding	In some cases, oversharding can help improve performance including intake speed. If your environment includes massively parallel hardware and you want to use these available resources, consider oversharding. You might increase the number of replicas per host from 1 to 2 or 3. Making such changes creates complex interactions, so you should continue to monitor your system's performance to ensure that the benefits of oversharding outweigh the costs.
Minimize swappiness	For better performance, Cloudera recommends setting the Linux swap space on all Solr server hosts as shown below: `sudo sysctl vm.swappiness=1`
Consider collection aliasing to deal with massive amounts of timestamped data in streaming-style applications	If you need to index and near real time query huge amounts of timestamped data in Solr, such as logs or IoT sensor data, you may consider aliasing as a massively scalable solution. This approach allows for indefinite indexing of data without degradation of performance otherwise experienced due to the continuous growth of a single index.

Additional tuning resources

Practical tuning tips outside the Cloudera Search documentation:

- General information on Solr caching is available under Query Settings in SolrConfig in the Apache Solr Reference Guide.
- Information on issues that influence performance is available on the SolrPerformanceFactors page on the Solr Wiki.
- Resource Management describes how to use Cloudera Manager to manage resources, for example with Linux cgroups.

- For information on improving querying performance, see How to make searching faster.
- For information on improving indexing performance, see How to make indexing faster.
- For information on aliasing, see Collection Aliasing: Near Real-Time Search for Really Big Data on Cloudera Blog and Time Routed Aliases in the Apache Solr Reference Guide.

Related Concepts

Tuning garbage collection

Solr and HDFS - the block cache

Tuning replication

Related Tasks

Enable multi-threaded faceting

Enable garbage collector logging

Related Information

Apache Solr Memory Tuning for Production

Solr Memory Tuning for Production (part 2)

Deployment Planning for Cloudera Search

Resource Management

Setting Java system properties for Solr

Several tuning steps require adding or modifying Java system properties. This is how you do it in Cloudera Manager.

Procedure

1. In Cloudera Manager, select the Solr service.
2. Click the Configuration tab.
3. In the Search box, type Java Configuration Options for Solr Server.
4. Add the property to Java Configuration Options for Solr Server using the format -D<property_name>=<*VALUE*>.

   Garbage collection options, such as -XX:+PrintGCTimeStamps, can also be set here. Use spaces to separate multiple parameters.
5. Click Save Changes.
6. Restart the Solr service (Solr service Actions Restart).

Enable multi-threaded faceting

Enabling multi-threaded faceting can provide better performance for field faceting.

About this task

- When multi-threaded faceting is enabled, field faceting tasks are completed in parallel with a thread working on every field faceting task simultaneously. Performance improvements do not occur in all cases, but improvements are likely when all of the following are true:
  - The system uses highly concurrent hardware.
  - Faceting operations apply to large data sets over multiple fields.
  - There is not an unusually high number of queries occurring simultaneously on the system. Systems that are lightly loaded or that are mainly engaged with ingestion and indexing may be helped by multi-threaded faceting; for example, a system ingesting articles and being queried by a researcher. Systems heavily loaded

by user queries are less likely to be helped by multi-threaded faceting; for example, an e-commerce site with heavy user-traffic.

Note: Multi-threaded faceting only applies to field faceting and not to query faceting.

- Field faceting identifies the number of unique entries for a field. For example, multi-threaded faceting could be used to simultaneously facet for the number of unique entries for the fields, "color" and "size". In such a case, there would be two threads, and each thread would work on faceting one of the two fields.
- Query faceting identifies the number of unique entries that match a query for a field. For example, query faceting could be used to find the number of unique entries in the "size" field that are between 1 and 5. Multi-threaded faceting does not apply to these operations.

Procedure

To enable multi-threaded faceting, add facet-threads to queries.

If facet-threads is omitted or set to 0, faceting is single-threaded. If facet-threads is set to a negative value, such as -1, multi-threaded faceting will use as many threads as there are fields to facet up to the maximum number of threads possible on the system.

For example, to use up to 1000 threads, use a similar query:

```
http://localhost:8983/solr/collection1/select?q=*:*&facet=true&fl=id&facet.f
ield=f0_ws&facet.threads=1000
```

Tuning garbage collection

Choose different garbage collection options for best performance in different environments.

Some garbage collection options typically chosen include:

- Concurrent low pause collector: Use this collector in most cases. This collector attempts to minimize "Stop the World" events. Avoiding these events can reduce connection timeouts, such as with ZooKeeper, and may improve user experience. This collector is enabled using the Java system property -XX:+UseConcMarkSweepGC.
- Throughput collector: Consider this collector if raw throughput is more important than user experience. This collector typically uses more "Stop the World" events so this may negatively affect user experience and connection timeouts such as ZooKeeper heartbeats. This collector is enabled using the Java system property -XX:+UseParallelGC. If UseParallelGC "Stop the World" events create problems, such as ZooKeeper timeouts, consider using the UseParNewGC collector as an alternative collector with similar throughput benefits.

You can also affect garbage collection behavior by increasing the Eden space to accommodate new objects. With additional Eden space, garbage collection does not need to run as frequently on new objects.

Related Tasks

Setting Java system properties for Solr

Enable garbage collector logging

To help identify any garbage collector (GC) issues, enable GC logging in production.

Procedure

1. In Cloudera Manager, select the Solr service.
2. Click the Configuration tab.

3. In the Search box, type Java Configuration Options for Solr Server.
4. Add arguments controlling GC logging behavior.
   - The minimum recommended GC logging flags are: -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+PrintGCDetails.
   - To rotate the GC logs: -Xloggc: -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles= -XX:GCLogFileSize=.

Solr and HDFS - the block cache

Cloudera Search enables Solr to store indexes in an HDFS filesystem. To maintain performance, an HDFS block cache has been implemented using Least Recently Used (LRU) semantics. This enables Solr to cache HDFS index files on read and write, storing the portions of the file in JVM direct memory (off heap) by default, or optionally in the JVM heap.

Warning: Do not enable the Solr HDFS write cache, because it can lead to index corruption.

Cloudera Search enables Solr to store indexes in an HDFS filesystem. To maintain performance, an HDFS block cache has been implemented using Least Recently Used (LRU) semantics. This enables Solr to cache HDFS index files on read and write, storing the portions of the file in JVM direct memory (off heap) by default, or optionally in the JVM heap.

Batch jobs typically do not use the cache, while Solr servers (when serving queries or indexing documents) should. When running indexing using MapReduce (MR), the MR jobs themselves do not use the block cache. Block write caching is turned off by default and should be left disabled.

Tuning of this cache is complex and best practices are continually being refined. In general, allocate a cache that is about 10-20% of the amount of memory available on the system. For example, when running HDFS and Solr on a host with 96 GB of memory, allocate 10-20 GB of memory using solr.hdfs.blockcache.slab.count. As index sizes grow you may need to tune this parameter to maintain optimal performance.

Note: Block cache metrics are currently unavailable.

Configure Index Caching

The following parameters control caching. They can be configured at the Solr process level by setting the respective Java system property or by editing solrconfig.xml directly.

If the parameters are set at the collection level (using solrconfig.xml), the first collection loaded by the Solr server takes precedence, and block cache settings in all other collections are ignored. Because you cannot control the order in which collections are loaded, you must make sure to set identical block cache settings in every collection solrconfig.xml. Block cache parameters set at the collection level in solrconfig.xml also take precedence over parameters at the process level.

Parameter	Cloudera Manager Setting	Default	Description
solr.hdfs.blockcache.enabled	HDFS Block Cache	true	Enable the block cache.

Parameter	Cloudera Manager Setting	Default	Description
solr.hdfs.blockcache.read.enabled	Not directly configurable. If the block cache is enabled, Cloudera Manager automatically enables the read cache. To override this setting, you must use the Solr Service Environment Advanced Configuration Snippet (Safety Valve).	true	Enable the read cache.
solr.hdfs.blockcache.blocksperbank	HDFS Block Cache Blocks per Slab	16384	Number of blocks per cache slab. The size of the cache is 8 KB (the block size) times the number of blocks per slab times the number of slabs.
solr.hdfs.blockcache.slab.count	HDFS Block Cache Number of Slabs	1	Number of slabs per block cache. The size of the cache is 8 KB (the block size) times the number of blocks per slab times the number of slabs.

To configure index caching using Cloudera Manager:

1. Go to the Solr service.
2. Click the Configuration tab.
3. In the Search box, type
   HDFS Block Cache

   to toggle the value of solr.hdfs.blockcache.enabled and enable or disable the block cache.

   HDFS Block Cache Blocks per Slab

   to configure solr.hdfs.blockcache.blocksperbank and set the number of blocks per cache slab.

   HDFS Block Cache Number of Slabs

   to configure solr.hdfs.blockcache.slab.count and set the number of slabs per block cache.
4. Set the new parameter value.
5. Restart Solr servers after editing the parameter.

Note:

Increasing the direct memory cache size may make it necessary to increase the maximum direct memory size (MaxDirectMemorySize) allowed by the JVM. Each Solr slab allocates memory, which is 128 MB by default, as well as allocating some additional direct memory overhead. Therefore, ensure that the MaxDirectMemorySize is set comfortably above the value expected for slabs alone. The amount of additional memory required varies according to multiple factors, but for most cases, setting MaxDirectMemorySize to at least 10-20% more than the total memory configured for slabs is sufficient. Setting MaxDirectMemorySize to the number of slabs multiplied by the slab size does not provide enough memory.

For example, for a direct memory of 20 GB the slab count should be no more than 140 with the default slab size, so there are 7 * 128 MB to store per GB of memory.

To increase the maximum direct memory size (MaxDirectMemorySize) using Cloudera Manager

1. Go to the Solr service.
2. Click the Configuration tab.
3. In the Search box, type Java Direct Memory Size of Solr Server in Bytes.
4. Set the new direct memory value.
5. Restart Solr servers after editing the parameter.

Solr HDFS optimizes caching when performing NRT indexing using Lucene's NRTCachingDirectory.

Lucene caches a newly created segment if both of the following conditions are true:

- The segment is the result of a flush or a merge and the estimated size of the merged segment is <= solr.hdfs.nrtcachingdirectory.maxmergesizemb.
- The total cached bytes is <= solr.hdfs.nrtcachingdirectory.maxcachedmb.

The following parameters control NRT caching behavior:

Parameter	Default	Description
solr.hdfs.nrtcachingdirectory.enable	true	Whether to enable the NRTCachingDirectory.
solr.hdfs.nrtcachingdirectory.maxcachedmb	192	Size of the cache in megabytes.
solr.hdfs.nrtcachingdirectory.maxmergesizemb	16	Maximum segment size to cache.

This is an example solrconfig.xml file with defaults:

```
 <directoryFactory name="DirectoryFactory">
    <bool name="solr.hdfs.blockcache.enabled">${solr.hdfs.blockcache.enabled
:true}</bool>
    <int name="solr.hdfs.blockcache.slab.count">${solr.hdfs.blockcache.sla
b.count:1}</int>
    <bool name="solr.hdfs.blockcache.direct.memory.allocation">${solr.hdfs.
blockcache.direct.memory.allocation:true}</bool>
    <int name="solr.hdfs.blockcache.blocksperbank">${solr.hdfs.blockcache.b
locksperbank:16384}</int>
    <bool name="solr.hdfs.blockcache.read.enabled">${solr.hdfs.blockcache.
read.enabled:true}</bool>
    <bool name="solr.hdfs.nrtcachingdirectory.enable">${solr.hdfs.nrtcachi
ngdirectory.enable:true}</bool>
    <int name="solr.hdfs.nrtcachingdirectory.maxmergesizemb">${solr.hdfs.nrt
cachingdirectory.maxmergesizemb:16}</int>
    <int name="solr.hdfs.nrtcachingdirectory.maxcachedmb">${solr.hdfs.nrtc
achingdirectory.maxcachedmb:192}</int>
</directoryFactory>
```

Related Tasks

Setting Java system properties for Solr

Related Information

Query Settings in SolrConfig - Apache Solr Reference Guide

Tuning replication

If you have sufficient additional hardware, you may add more replicas for a linear boost of query throughput and to prevent data loss.

Note: Do not adjust HDFS replication settings for Solr in most cases.

Note, that adding replicas may slow write performance on the first replica, but otherwise this should have minimal negative consequences.

Transaction log replication

Cloudera Search supports configurable transaction log replication levels for replication logs stored in HDFS. Cloudera recommends leaving the value unchanged at 3 or, barring that, setting it to at least 2.

Configure the transaction log replication factor for a collection by modifying the tlogDfsReplication setting in solr config.xml. The tlogDfsReplication is a setting in the updateLog settings area. An excerpt of the solrconfig.xml file where the transaction log replication factor is set is as follows:

```
 <updateHandler class="solr.DirectUpdateHandler2">

    <!-- Enables a transaction log, used for real-time get, durability, and
         solr cloud replica recovery.  The log can grow as big as
         uncommitted changes to the index, so use of a hard autoCommit
         is recommended (see below).
         "dir" - the target directory for transaction logs, defaults to the
                solr data directory.  -->
    <updateLog>
      <str name="dir">${solr.ulog.dir:}</str>
      <int name="tlogDfsReplication">${solr.ulog.tlogDfsReplication:3}</int>
      <int name="numVersionBuckets">${solr.ulog.numVersionBuckets:65536}</
int>
    </updateLog>
```

The default replication level is 3. For clusters with fewer than three DataNodes (such as proof-of-concept clusters), reduce this number to the amount of DataNodes in the cluster. Changing the replication level only applies to new transaction logs.

Initial testing shows no significant performance regression for common use cases.

Related Tasks

Adjust the Solr replication factor for index files stored in HDFS

Adjust the Solr replication factor for index files stored in HDFS

You can adjust the degree to which different data is replicated.

Procedure

1. Go to Solr service Configuration Category Advanced .
2. Click the plus sign next to Solr Service Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml to add a new property with the following values:

   Name: dfs.replication

   Value: 2
3. Click Save Changes.
4. Restart the Solr service (Solr service Actions Restart).

Cloudera Runtime 7.2.18

Cloudera Search Tutorial

Date published: 2019-11-19
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Cloudera Search tutorial 4

Validating the Cloudera Search deployment 4
- Create a test collection 4
- Index sample data 5
- Query sample data 5

Indexing sample tweets with Cloudera Search 6
- Create a collection for tweets 7
- Copy sample tweets to HDFS 8
- Using MapReduce batch indexing to index sample Tweets 9
  - Batch indexing into offline Solr shards 9

Cloudera Search tutorial

This tutorial introduces you to Cloudera Search and demonstrates some of its basic capabilities to help you become familiar with the concepts and components involved in Search. It demonstrates creating a simple test collection to validate your Cloudera Search installation, and then continues on to more advanced use cases of batch indexing.

The topics in this tutorial assume you have deployed Cloudera Search. The examples in this tutorial use two shards, so make sure that your deployment includes at least two Solr servers.

This tutorial uses modified schema.xml and solrconfig.xml configuration files. In the versions of these files included with the tutorial, unused fields have been removed for simplicity. Original versions of these files include many additional options. For information on all available options, see the Apache Solr wiki:

- SchemaXml
- SolrConfigXml

Validating the Cloudera Search deployment

After installing and deploying Cloudera Search, validate the deployment by indexing and querying sample documents.

Important: This tutorial does not account for Apache Ranger authorization. If you are using Ranger authorization, you must create policies to allow the users and actions described in the tutorial.

You can think of this as a type of "Hello, World!" for Cloudera Search to make sure that everything is installed and working properly.

Before beginning this process, make sure you have access to the Apache Solr admin web console. If your cluster is Kerberos-enabled, make sure you have access to the solr@EXAMPLE.COM Kerberos principal (where EXAMPLE.COM is your Kerberos realm name).

Create a test collection

Generate configuration files and upload the generated configuration to ZooKeeper, so that you can create a collection where you can index sample data.

Procedure

1. Make sure that the SOLR_ZK_ENSEMBLE environment variable is set in /etc/solr/conf/solr-env.sh.
   For example:

   ```
   cat /etc/solr/conf/solr-env.sh
   ```

   ```
   export SOLR_ZK_ENSEMBLE=zk01.example.com:2181,zk02.example.com:2181,zk03
   .example.com:2181/solr
   ```

   This is automatically set on hosts with a Solr Server or Gateway role in Cloudera Manager.
2. If you are using Kerberos, kinit as the user that has privileges to create the collection.

   For example:

   ```
   kinit SOLR@EXAMPLE.COM
   ```

   Replace *SOLR@EXAMPLE.COM* with your user name and Kerberos realm name respectively.

3. Generate configuration files for the collection:

```
solrctl instancedir --generate $HOME/test_collection_config
```

4. Upload the configuration to ZooKeeper:

```
solrctl config --upload test_collection_config $HOME/test_collection_con
fig
```

5. Create a new collection with two shards (specified by the -s parameter) using the named configuration (specified by the -c parameter).

```
solrctl collection --create test_collection -s 2 -c test_collection_config
```

Index sample data

Cloudera Search includes sample data for testing and validation. Run the relevant command to index this data for searching.

About this task

Replace *SEARCH01.EXAMPLE.COM* in the example below with the name of any host running the Solr Server process.

Note: The default static port 8985 in the examples is only valid when you access the host from within the Data Hub cluster. When you want to access the Solr server from outside the cluster, you need to address it via the cdp-proxy-api endpoint. To find out the URL of the Solr server, navigate to the Data Hub Clusters service, or to Cloudera Management Console Data Hub Clusters , and click on the tile representing your cluster. This brings you to the cluster details page where the URLs to cluster UIs and endpoints are listed. Select the **Endpoints** tab to find the service endpoint URLs.

For more information, see Accessing Cloudera Manager, cluster UIs, and endpoints.

Procedure

1. SSH to a Solr host.
2. Run the following command:

```
cd /opt/cloudera/parcels/CDH/share/doc/solr-doc*/example/exampledocs
                    find *.xml -exec curl -i -k --negotiate -u: https:
//SEARCH01.EXAMPLE.COM:8985/solr/test_collection/update -H "Content-Type:
text/xml" --data-binary @{} \;
```

Related Information

Accessing Data Hub cluster via SSH

Query sample data

Run a query to verify that the sample data is successfully indexed and that you are able to search it.

About this task

Note: The default static port 8985 in the examples is only valid when you access the host from within the Data Hub cluster. When you want to access the Solr server from outside the cluster, you need to address it via the cdp-proxy-api endpoint. To find out the URL of the Solr server, navigate to the Data Hub Clusters service, or to Cloudera Management Console Data Hub Clusters , and click on the tile representing your cluster. This brings you to the cluster details page where the URLs to cluster UIs and endpoints are listed. Select the **Endpoints** tab to find the service endpoint URLs.

For more information, see Accessing Cloudera Manager, cluster UIs, and endpoints.

Procedure

1. SSH into a cluster node.
2. Open the Solr admin web interface in a browser.

   https://*SEARCH01.EXAMPLE.COM*:8985/solr

   Replace *SEARCH01.EXAMPLE.COM* with the name of any host running the Solr Server process. If you have security enabled on your cluster, enter the credentials for the Kerberos principal when prompted.
3. Select Cloud from the left panel.
4. From the Collection Selector drop-down menu in the left panel, select the test_collection collection.
5. Select Query from the left panel and click Execute Query. If you see results such as the following, indexing was successful:


```
"response":{"numFound":32,"start":0,"maxScore":1.0,"docs":[
      {
        "id":"SP2514N",
        "name":["Samsung SpinPoint P120 SP2514N - hard drive - 250 GB -
ATA-133"],
        "manu":["Samsung Electronics Co. Ltd."],
        "manu_id_s":"samsung",
        "cat":["electronics",
          "hard drive"],
        "features":["7200RPM, 8MB cache, IDE Ultra ATA-133",
          "NoiseGuard, SilentSeek technology, Fluid Dynamic Bearing (FDB)
 motor"],
        "price":[92.0],
        "popularity":[6],
        "inStock":[true],
        "manufacturedate_dt":"2006-02-13T15:26:37Z",
        "store":["35.0752,-97.032"],
        "manu_str":["Samsung Electronics Co. Ltd."],
        "_version_":1650678864820568064,
        "cat_str":["electronics",
          "hard drive"],
        "name_str":["Samsung SpinPoint P120 SP2514N - hard drive - 250 GB
 - ATA-133"],
        "features_str":["7200RPM, 8MB cache, IDE Ultra ATA-133",
          "NoiseGuard, SilentSeek technology, Fluid Dynamic Bearing (FDB)
 motor"],
        "store_str":["35.0752,-97.032"]},
```


Indexing sample tweets with Cloudera Search

After you have verified that Cloudera Search is installed and running properly, you can experiment with other methods of ingesting and indexing data. This tutorial uses tweets to demonstrate batch indexing.

Related Information

To learn more about Solr, see the Apache Solr Tutorial

Create a collection for tweets

In this part of the Cloudera Search tutorial, you create a collection for tweets.

About this task

The remaining examples in the tutorial use the same collection, so make sure that you follow these instructions carefully.

Procedure

1. On a host with Solr Server installed, make sure that the SOLR_ZK_ENSEMBLE environment variable is set in /etc/solr/conf/solr-env.sh.

   For example:

   ```
   cat /etc/solr/conf/solr-env.sh
   ```

   ```
   export SOLR_ZK_ENSEMBLE=zk01.example.com:2181,zk02.example.com:2181,zk03
   .example.com:2181/solr
   ```

   This is automatically set on hosts with a Solr Server or Gateway role in Cloudera Manager

2. If you are using Kerberos, kinit as the user that has privileges to create the collection.

   For example:

   ```
   kinit SOLR@EXAMPLE.COM
   ```

   Replace *SOLR@EXAMPLE.COM* with your user name and Kerberos realm name respectively.

3. Generate the configuration files for the collection, including the tweet-specific managed-schema:

   ```
   solrctl instancedir --generate $HOME/cloudera_tutorial_tweets_config
                    cp /opt/cloudera/parcels/CDH/share/doc/search*/search-
   crunch/solr/collection1/conf/schema.xml $HOME/cloudera_tutorial_tweets_c
   onfig/conf/managed_schema
   ```

   To the overwrite confirmation prompt:

   ```
   cp: overwrite 'cloudera_tutorial_tweets_config/conf/managed-schema'?
   ```

   type 'y'.

4. Upload the configuration to ZooKeeper:

   ```
   solrctl config --upload cloudera_tutorial_tweets_config $HOME/cloudera_t
   utorial_tweets_config
   ```

5. Create a new collection with two shards (specified by the -s parameter) using the named configuration (specified by the -c parameter):

   ```
   solrctl collection --create cloudera_tutorial_tweets -s 2 -c cloudera_tu
   torial_tweets_config
   ```

6. Verify that the collection is live. Open the Solr admin web interface in a browser by accessing the relevant URL:
   - https://*SEARCH01.EXAMPLE.COM*:8985/solr/#/~cloud

   If you have Kerberos authentication enabled on your cluster, enter the credentials for the solr@EXAMPLE.COM principal when prompted. Replace *SEARCH01.EXAMPLE.COM* with the name of any host running the Solr Server process. Look for the cloudera_tutorial_tweets collection to verify that it exists.
7. Prepare the configuration for use with MapReduce:

```
cp -r $HOME/cloudera_tutorial_tweets_config $HOME/cloudera_tutorial_twee
ts_mr_config
```

Copy sample tweets to HDFS

Copy the provided sample tweets to HDFS. These tweets are used to demonstrate the batch indexing capabilities of Cloudera Search.

Procedure

1. Copy the provided sample tweets to HDFS:

   Security Enabled:

   a.
   ```
   kinit [***HDFS@EXAMPLE.COM***]
   ```
   b.
   ```
   hdfs dfs -mkdir -p /user/[***USER***]
   ```
   c.
   ```
   hdfs dfs -chown [***USER***]:[***GROUP***] /user/[***USER***]
   ```
   d.
   ```
   kinit [***USER@EXAMPLE.COM***]
   ```
   e.
   ```
   hdfs dfs -mkdir -p /user/[***USER***]/indir
   ```
   f.
   ```
   hdfs dfs -put /opt/cloudera/parcels/CDH/share/doc/search*/examples/test-
   documents/sample-statuses-*.avro /user/[***USER***]/indir/
   ```
   g.
   ```
   hdfs dfs -ls /user/[***USER***]/indir
   ```

   Security Disabled: Run the following commands as *[***USER***]*:

```
sudo -u hdfs hdfs dfs -mkdir -p /user/[***USER***]
```

```
sudo -u hdfs hdfs dfs -chown [***USER***]:[***GROUP***] /user/[***USER***]
```

```
hdfs dfs -mkdir -p /user/[***USER***]/indir
```

```
hdfs dfs -put /opt/cloudera/parcels/CDH/share/doc/search*/examples/test-
documents/sample-statuses-*.avro /user/[***USER***]/indir/
```

```
hdfs dfs -ls /user/[***USER***]/indir
```

2. Ensure that outdir is empty and exists in HDFS:

```
hdfs dfs -rm -r -skipTrash /user/[***USER***]/outdir
```

```
hdfs dfs -mkdir /user/[***USER***]/outdir
```

```
hdfs dfs -ls /user/[***USER***]/outdir
```

What to do next

The sample tweets are now in HDFS and ready to be indexed. Continue to the next section to index the sample tweets.

Using MapReduce batch indexing to index sample Tweets

Batch indexing is useful for periodically indexing large amounts of data, or for indexing a dataset for the first time.

Before continuing, make sure that you have completed the procedures earlier in the tutorial.

Batch indexing into offline Solr shards

Batch indexing into offline Solr shards is mainly intended for offline use-cases by advanced users. Use cases requiring read-only indexes for searching can be handled by using batch indexing without the --go-live option. By not using GoLive, you can avoid copying datasets between segments, thereby reducing resource utilization.

About this task

Running the MapReduce job without GoLive causes the job to create a set of Solr index shards from a set of input files and write the indexes to HDFS. You can then explicitly point each Solr server to one of the HDFS output shard directories.

Procedure

1. If you are working with a secured cluster, configure your client JAAS file ($HOME/jaas.conf) as follows:

```
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=false
 useTicketCache=true
 principal="solr@EXAMPLE.COM";
 };
```

2. If you are using Kerberos, kinit as the user that has privileges to update the collection:

```
kinit jdoe@EXAMPLE.COM
```

   Replace EXAMPLE.COM with your Kerberos realm name.

3. Delete any existing documents in the cloudera_tutorial_tweets collection. If your cluster does not have security enabled, run the following commands as the solr user by adding sudo -u solr before the command:

```
solrctl collection --deletedocs cloudera_tutorial_tweets
```

4. Delete the contents of the outdir directory:

```
hdfs dfs -rm -r -skipTrash /user/JDOE/outdir/*
```

5. Run the MapReduce job as follows, replacing *NN01.EXAMPLE.COM* in the command with your NameNode hostname.
   - Security enabled:

```
YARN_OPTS="-Djava.security.auth.login.config=/PATH/TO/jaas.conf" yarn
 jar /opt/cloudera/parcels/CDH/lib/solr/contrib/mr/search-mr-*-job.jar
 org.apache.solr.hadoop.MapReduceIndexerTool -D 'mapred.child.java.opt
s=-Xmx500m' -D 'mapreduce.job.user.classpath.first=true' --log4j /opt/
cloudera/parcels/CDH/share/doc/search*/examples/solr-nrt/log4j.propert
ies --morphline-file /opt/cloudera/parcels/CDH/share/doc/search*/exa
mples/solr-nrt/test-morphlines/tutorialReadAvroContainer.conf --outp
ut-dir hdfs://NN01.EXAMPLE.COM:8020/user/JDOE/outdir --verbose --zk-h
ost ZK01.EXAMPLE.COM:2181/solr --collection cloudera_tutorial_tweets --s
hards 2 hdfs://NN01.EXAMPLE.COM:8020/user/JDOE/indir
```

6. Check the job status at:

```
http://RM01.EXAMPLE.COM:8088/ui2/#/yarn-apps/apps
```

   For secure clusters, replace http with https and port 8088 with 8090.

7. After the job is completed, check the generated index files. Individual shards are written to the results directory with names of the form part-00000, part-00001, part-00002, and so on. This example has two shards:

```
hdfs dfs -ls /user/JDOE/outdir/results
```

```
hdfs dfs -ls /user/JDOE/outdir/results/part-00000/data/index
```

8. In the Cloudera Manager web console for the cluster, stop the Solr service (Solr service Actions Stop).
9. Identify the paths to each Solr core:

```
hdfs dfs -ls /solr/cloudera_tutorial_tweets
```

```
Found 2 items
drwxr-xr-x   - solr solr          0 2017-03-13 06:20 /solr/cloudera_tutori
al_tweets/core_node1
drwxr-xr-x   - solr solr          0 2017-03-13 06:20 /solr/cloudera_tut
orial_tweets/core_node2
```

10. Move the index shards into place.
    a) (Kerberos only) Switch to the solr user:

```
kinit solr@EXAMPLE.COM
```

    b) Remove outdated files. If your cluster does not have security enabled, run the following commands as the solr user by adding sudo -u solr before the command:

```
hdfs dfs -rm -r -skipTrash /solr/cloudera_tutorial_tweets/core_node1/dat
a/index
hdfs dfs -rm -r -skipTrash /solr/cloudera_tutorial_tweets/core_node1/dat
a/tlog
hdfs dfs -rm -r -skipTrash /solr/cloudera_tutorial_tweets/core_node2/
data/index
hdfs dfs -rm -r -skipTrash /solr/cloudera_tutorial_tweets/core_node2/
data/tlog
```

c) Change ownership of the results directory to solr. If your cluster has security enabled, kinit as the HDFS superuser (hdfs by default) before running the following command. If your cluster does not have security enabled, run the command as the HDFS superuser by adding sudo -u hdfs before the command:

```
hdfs dfs -chown -R solr /user/JDOE/outdir/results
```

d) (Kerberos only) Switch to the solr user:

```
kinit solr@EXAMPLE.COM
```

e) Move the two index shards into place. If your cluster does not have security enabled, run the following commands as the solr user by adding sudo -u solr before the command:

```
hdfs dfs -mv /user/JDOE/outdir/results/part-00000/data/index /solr/clou
dera_tutorial_tweets/core_node1/data
```

```
hdfs dfs -mv /user/JDOE/outdir/results/part-00001/data/index /solr/clou
dera_tutorial_tweets/core_node2/data
```

11. In the Cloudera Manager web console for the cluster, start the Solr service (Solr service Actions Start).

12. Run some Solr queries. For example, for a Solr server running on search01.example.com, go to one of the following URLs in a browser, depending on whether you have enabled security on your cluster:

- Security enabled: https://search01.example.com:8985/solr/cloudera_tutorial_tweets/select?q=*:*

If indexing was successful, this page displays the first 10 query results.

Cloudera Runtime 7.3.1

Securing Apache Hive

Date published: 2019-08-21
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Hive access authorization.....4

Transactional table access..... 5

External table access..... 5

HWC authorization..... 5

Hive access authorization

As administrator, you need to understand that the Hive default authorization for running Hive queries is insecure and what you need to do to secure your data. You need to set up Apache Ranger.

To limit Apache Hive access to approved users, Cloudera recommends and supports only Ranger. Authorization is the process that checks user permissions to perform select operations, such as creating, reading, and writing data, as well as editing table metadata. Apache Ranger provides centralized authorization for all Cloudera Runtime Services.

You can set up Ranger to protect managed, ACID tables or external tables using a Hadoop SQL policy. You can protect external table data on the file system by using an HDFS policy in Ranger.

Preloaded Ranger Policies

In Ranger, preloaded Hive policies are available by default. Users covered by these policies can perform Hive operations. All users need to use the default database, perform basic operations such as listing database names, and query the information schema. To provide this access, preloaded default database tables columns and information_schema database policies are enabled for group public (all users). Keeping these policies enabled for group public is recommended. For example, if the default database tables columns policy is disabled preventing use of the default database, the following error appears:

```
hive> USE default;
Error: Error while compiling statement: FAILED: HiveAccessControlException
Permission denied: user [hive] does not have [USE] privilege on [default]
```

Apache Ranger policy authorization

Apache Ranger provides centralized policy management for authorization and auditing of all Cloudera Runtime services, including Hive. All Cloudera Runtime services are installed with a Ranger plugin used to intercept authorization requests for that service, as shown in the following illustration.

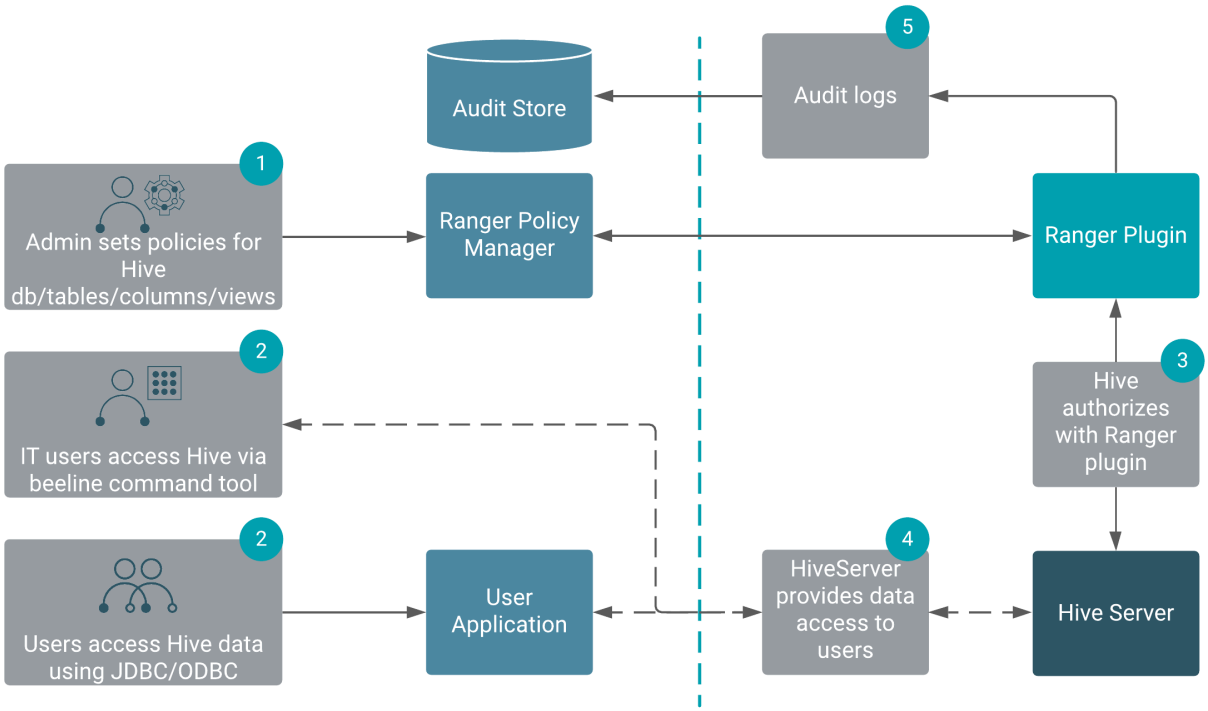

The following table compares authorization models:

Authorization model	Secure?	Fine-grained authorization (column, row level)	Privilege management using GRANT/REVOKE statements	Centralized management GUI
Apache Ranger	Secure	Yes	Yes	Yes
Hive default	Not secure. No restriction on which users can run GRANT statements	Yes	Yes	No

When you run grant/revoke commands and Apache Ranger is enabled, a Ranger policy is created/removed.

Transactional table access

As administrator, you must enable the Apache Ranger service to authorize users who want to work with transactional tables. These types of tables are the default, ACID-compliant tables in Hive 3 and later.

ACID tables reside by default in /warehouse/tablespace/managed/hive. Only the Hive service can own and interact with files in this directory. Ranger is the only available authorization mechanism that Cloudera recommends for ACID tables.

External table access

As administrator, you must set up Apache Ranger to allow users to access external tables.

External tables reside by default in /warehouse/tablespace/external on your object store. To specify some other location of the external table, you need to include the specification in the table creation statement as shown in the following example:

```
CREATE EXTERNAL TABLE my_external_table (a string, b string)
LOCATION '/users/andrena';
```

Hive assigns a default permission of 777 to the hive user, sets a umask to restrict subdirectories, and provides a default ACL to give Hive read and write access to all subdirectories. External tables must be secured using Ranger.

HWC authorization

The way you configure Hive Warehouse Connector (HWC) affects the query authorization process and your security. There are a number of ways to access Hive through HWC, and not all operations go through HiveServer (HS2). Some operations, such as Spark Direct Reader and Hive Streaming, go to Hive directly through HMS where storage-based permissions generally apply.

As a client user, you must be logged in using kerberos before using HWC. You need appropriate storage permissions to write to destination partition or table location. You need to configure an HWC read option. HWC read configuration options are shown in the following table:

Table 1:

Capabilities	JDBC mode	Spark Direct Reader mode	Secure Access mode
Ranger integration with fine-grained access control	#	N/A	#
Hive ACID reads	#	#	#

Capabilities	JDBC mode	Spark Direct Reader mode	Secure Access mode
Workloads handled	Non-production workloads, small datasets	Production workloads, ETL without fine-grained access control	Large workloads with fine-grained access control, row filtering, and column masking

These read configuration options require connections to different Hive components:

- Direct Reader configuration: Connects to Hive Metastore (HMS)
- JDBC configuration: Connects to HiveServer (HS2)
- Secure Access configuration: Connects to HiveServer (HS2)

Ranger authorizes access to Hive tables from Spark through HiveServer (HS2) or the Hive metastore API (HMS API).

To write ACID managed tables from Spark to Hive, you must use HWC. To write external tables from Spark to Hive, you can use native Spark or HWC.

The following diagram shows the typical read authorization process:

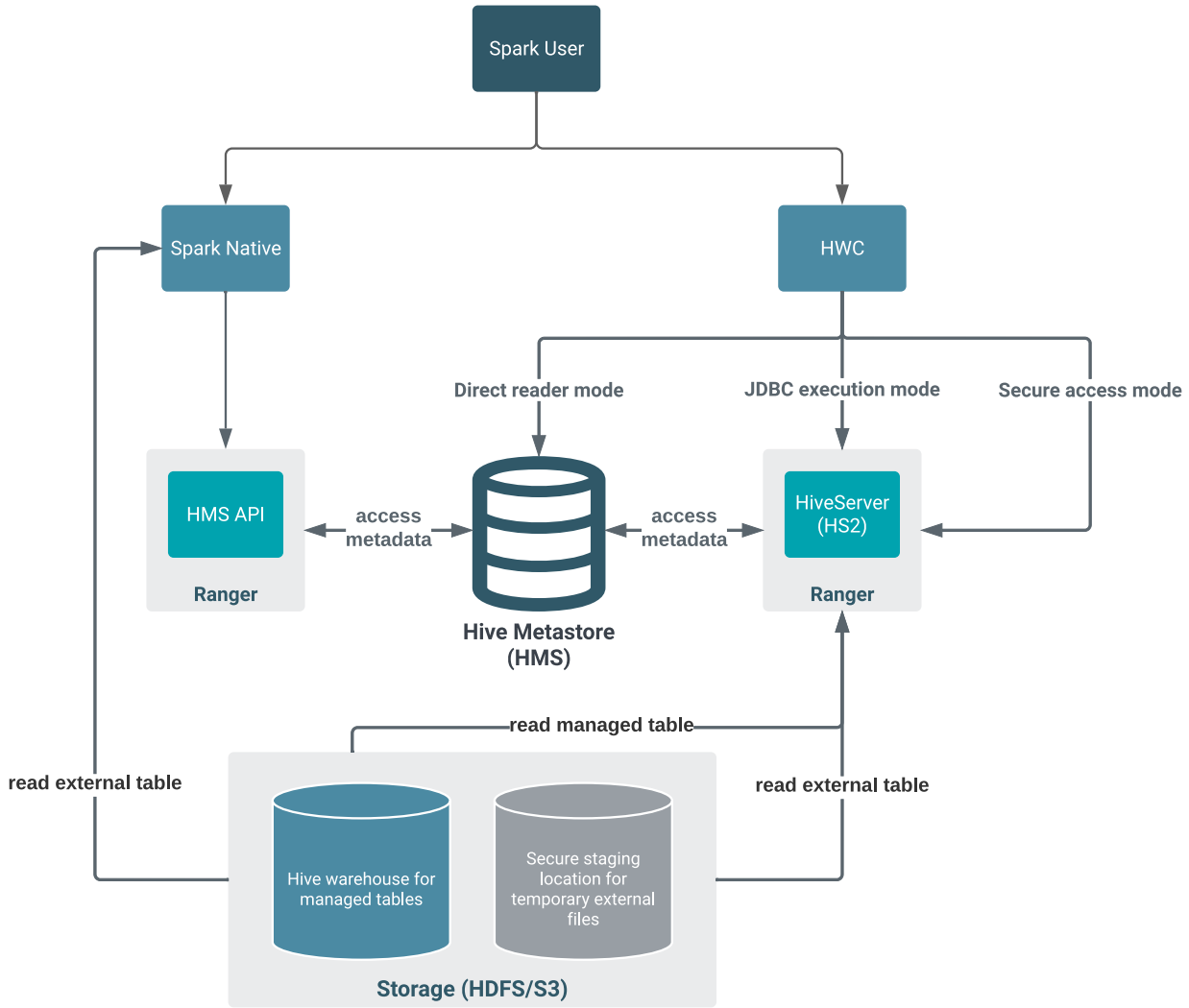

The following diagram shows the typical write authorization process:

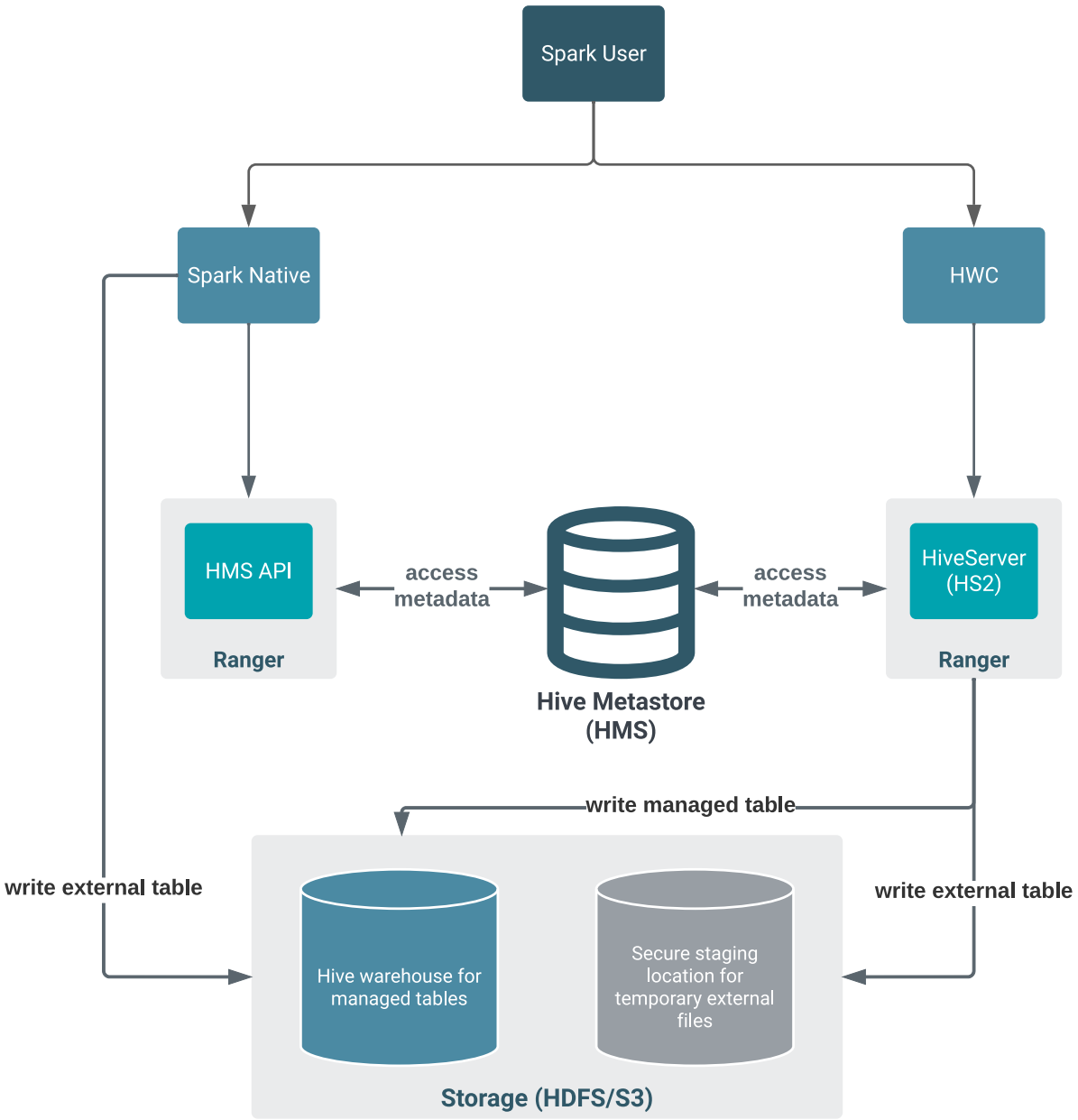

When writing, HWC always enforces authorization through HiveServer (HS2). Reading managed tables in JDBC mode or Secure access mode enforces Ranger authorization, including fine-grained features, such as row masking and column mapping. In Direct Reader mode, the Ranger and HMS integration provides authorization.

External table queries go through the HMS API, which is also integrated with Ranger. If you do not use HWC, the Hive metastore (HMS) API, integrated with Ranger, authorizes external table access. HMS API-Ranger integration enforces the Ranger Hive ACL in this case. When you use HWC, queries such as DROP TABLE affect file system data as well as metadata in HMS.

Using the Direct Reader option, SparkSQL queries read managed table metadata directly from the HMS, but only if you have permission to access files on the file system. You cannot write to managed tables using the Direct Reader option.

Using the Secure access mode, you can enable fine-grained access control (FGAC) column masking and row filtering to secure managed (ACID), or even external, Hive table data that you read from Spark.

Managed table authorization

A Spark job impersonates the end user when attempting to access an Apache Hive managed table. As an end user, you do not have permission to access, managed files in the Hive warehouse. Managed tables have default file system permissions that disallow end user access, including Spark user access.

As Administrator, you set permissions in Ranger to access the managed tables when you configure HWC for JDBC mode or Secure access mode reads. You can fine-tune Ranger to protect specific data. For example, you can mask data in certain columns, or set up tag-based access control.

When you configure HWC for Direct Reader mode, you cannot use Ranger in this way. You must set read access to the file system location for managed tables. You must have Read and Execute permissions on the Hive warehouse location (hive.metastore.warehouse.dir).

External table authorization

Ranger authorization of external table reads and writes is supported. You need to configure a few properties in Cloudera Manager for authorization of external table writes. You must be granted file system permissions on external table files to allow Spark direct access to the actual table data instead of just the table metadata.

Direct Reader Authorization Limitation

As Spark allows users to run arbitrary code, Ranger fine grained access control, such as row level filtering or column level masking, is not possible within Spark itself. This limitation extends to data read using Direct Reader.

To restrict data access at a fine-grained level, use a read option that supports Ranger. Only consider using the Direct Reader option to read Hive data from Spark if you do not require fine-grained access. For example, use Direct Reader for ETL use cases.

Cloudera Runtime 7.3.1

Securing Hue

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

User management in Hue 5
Understanding Hue users and groups 5
Finding the list of Hue superusers 6
Creating a Hue user 7
Restricting user login 8
LDAP import and sync options 9
Import and sync LDAP users and groups 10
Locking an account after invalid login attempts 11
Unlocking locked out user accounts in Hue 11
Creating a group in Hue 12
Managing Hue permissions 12
Resetting Hue user password 12
Assigning superuser status to an LDAP user 13

Configuring file and directory permissions for Hue 13

User authentication in Hue 14
Authenticating Hue users with Kerberos 14
Authenticating Hue users with LDAP 15
Configuring authentication with LDAP and Search Bind 18
Configuring authentication with LDAP and Direct Bind 19
Configuring Hue for authentication against multiple LDAP/Active Directory servers 20
Testing the LDAP configuration 22
Configuring group permissions 23
Enabling LDAP authentication with HiveServer2 and Impala 23
LDAP properties 24
Configuring LDAP on unmanaged clusters 26
Authenticating Hue users with SAML 27
Configuring SAML authentication on managed clusters 27
Manually configuring SAML authentication 28
Integrating your identity provider's SAML server with Hue 30
SAML properties 30
Troubleshooting SAML authentication 32
Authenticating Hue users with Knox SSO 33
Authentication using PAM 35

Applications and permissions reference 36

Securing Hue passwords with scripts 37

Directory permissions when using PAM authentication backend 38

Configuring TLS/SSL for Hue 39

Creating a truststore file in PEM format.....39
Configuring Hue as a TLS/SSL client.....40
Enabling Hue as a TLS/SSL client.....40
Configuring Hue as a TLS/SSL server.....40
Enabling Hue as a TLS/SSL server using Cloudera Manager.....40
Enabling TLS/SSL for Hue Load Balancer.....41
Enabling TLS/SSL communication with HiveServer2.....42
Enabling TLS/SSL communication with Impala.....42
Securing database connections with TLS/SSL.....43
Disabling CA Certificate validation from Hue.....43

Enforcing TLS version 1.2 for Hue.....44

Securing sessions.....45

Specifying HTTP request methods.....48

Restricting supported ciphers for Hue.....48

Specifying domains or pages to which Hue can redirect users.....48

Securing Hue from CWE-16.....49

Setting Oozie permissions.....49

Configuring secure access between Solr and Hue.....51

User management in Hue

Hue is a gateway to Cloudera cluster services and both have completely separate permissions. Being a Hue superuser does not grant access to HDFS, Hive, and so on.

Users who log on to the Hue UI must have permission to use Hue and to each Cloudera service accessible within Hue.

A common configuration is for “Hue users” to be authenticated with an LDAP server and “Cloudera users” with Kerberos. These users can differ. For example, Cloudera services do not authenticate each user who logs on to Hue. Rather, they authenticate “Hue” and trust that Hue has authenticated “its” users.

Once Hue is authenticated by a service such as Hive, Hue impersonates the user requesting use of that service. For example, to create a Hive table. The service uses Apache Ranger to ensure the group to which that user belongs is authorized for that action.

Hue user permissions are at the application level only. For example, a Hue superuser can filter Hue user access to a Cloudera service but cannot authorize the use of its features. Again, Ranger does that.

Understanding Hue users and groups

There are two types of users in Hue - superusers and general users referred to as users, each with specific privileges. These users can be a part of certain groups. Groups enable you to control which Hue applications and features your users can view and access when they log into Hue.

On a non-secure Cloudera cluster, the first user logging into Hue after the initial installation becomes the first superuser. Superusers have the permissions to perform the following administrative functions:

- Add and delete users
- Add and delete groups
- Assign permissions to groups
- Change a user into a superuser
- Import users and groups from an LDAP server

If a user is part of the superuser LDAP group in Hue, then that user is also a part of the group of superusers in Hue.

Users can only change their name, e-mail address, and password. They can log in to Hue and run Hue applications, subject to the permissions provided by the Hue groups to which they belong. This is different from how Cloudera perceives the Hue application when you submit a Hive or an Impala query from the Hue user interface (UI). Hue is a server between the users and the Cloudera services. Hue is considered as a single ‘hue’ user by the other services in the Cloudera cluster.

For example, when a user ‘bob’ submits a query from Hue, Hue also sends the username of this user to the corresponding service in Cloudera. The HIVE_ON_TEZ service in Cloudera considers ‘bob’ as the owner of the query and not ‘hue’. This is illustrated in the following graphic:

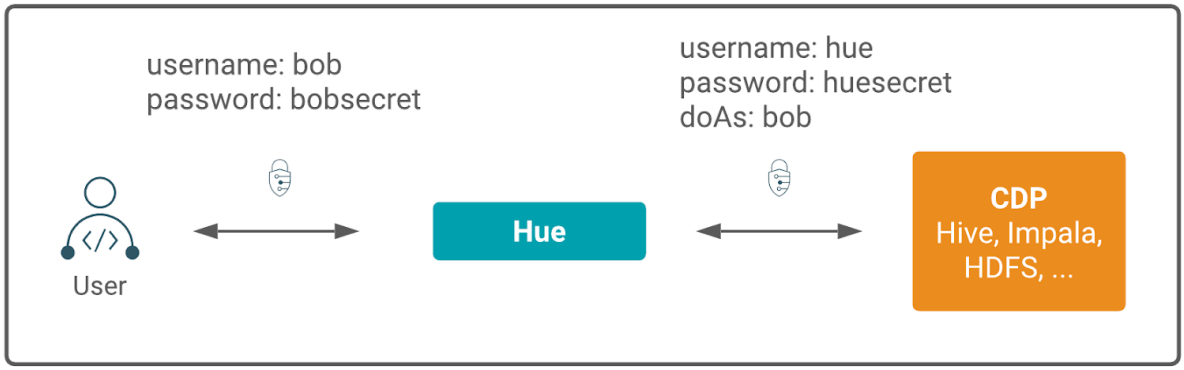

Hue is a gateway to Cloudera cluster services and both have separate permissions. A Hue superuser is not granted access to HDFS, Hive, and other Cloudera cluster services. Apache Ranger governs access to the Cloudera cluster services.

Note: Groups in Hue are different from groups in Ranger.

Hue user permissions are at the application level only. For example, a Hue superuser can filter Hue user access to a Cloudera service but cannot authorize the use of its features. Users who log on to the Hue UI must have permission to use Hue and to each Cloudera service accessible within Hue.

Finding the list of Hue superusers

You can fetch the list of superusers by using the Hue shell with Python code or by running a SQL query on the auth _user table.

Using the Hue shell and Python code to find Hue superusers

1. Connecting to Hue shell by running the following command:

```
/opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue shell --cm-managed
```

2. Enter the Python code as follows:

```
from django.contrib.auth.models import User
print "%s" % User.objects.filter(is_superuser = True)
```

Sample output:

```
<QuerySet [<User: admin>]>
```

Running a SQL query on the auth_user table to find Hue superusers

1. Connect to Hue database shell by running the following command:

```
/opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue dbshell --cm-managed
```

2. Run the following SQL query:

```
select username, is_superuser from auth_user where is_superuser=1;
```

Sample output:

```
----------------------+
username is_superuser
----------------------+

admin 1
----------------------+
1 row in set (0.00 sec)
```

Creating a Hue user

You can create new Hue users and superusers from the Hue web UI and assign them to groups so that they can view and access Hue as per the permissions granted to them.

About this task

Note: You can create a username for Hue 150 characters long.

Before you begin

Per user home directories are created on HDFS when you create a new user in Hue, which the user can use to store and retrieve files using the Hue File Browser. To prevent unauthorized access by other Hue users, add the following lines in Cloudera Manager Clusters Hue Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini :

```
[useradmin]
home_dir_permissions=0700
use_home_dir_permissions=true
```

Set use_home_dir_permissions to false to use the HDFS umask.

Procedure

1. Sign in to the Hue UI as a superuser.

2. From the left assist panel, point your cursor to the user profile icon and click Manage Users.

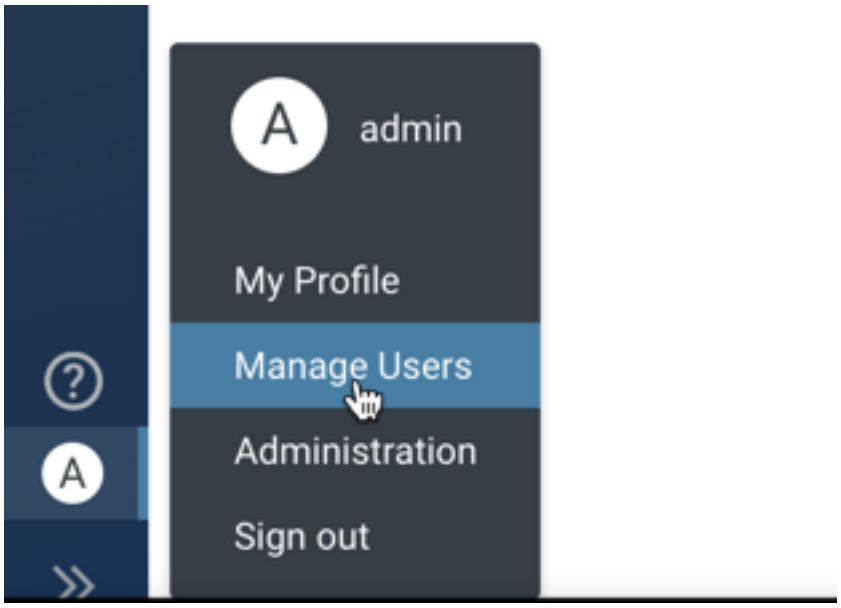

The **User Admin** page is displayed.

3. On the **User Admin** page, click Add User.

   The **Create user** page is displayed.

4. Enter the username and password for the user that you are adding on the Credentials tab.

   To create a separate Hue home directory for the user, select the Create home directory option.

   Click Next.

5. On the Profile and Group tab, create a profile for the user by entering the details such as name and email address.

   At this point, if you have already created a group(s) that you want to assign to the user, then select it from the list displayed in the Groups field.

   A user can be a part of more than one group.

   Click Next.

6. (Optional) On the Advanced tab, select the Superuser status option to make this user a superuser and click Add user.

   The new user is displayed on the **Users** page.

Related Information

Configuring file and directory permissions for Hue

Controlling user access to Hue

Administrators can control users who can access Hue by creating different LDAP login groups, adding users needing access to these login groups, and then specifying the login groups in the Advanced Configuration Snippet in Cloudera Manager.

Before you begin

Ensure that you have added users needing access to Hue to a login group.

Procedure

1. Log in to Cloudera Manager as an Administrator.

2. Go to Clusters Hue service Configuration and enter the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[desktop]
[[ldap]]
login_groups=[***LDAP-GRP1***], [***LDAP-GRP2***], [***LDAP-GRP3***]
```

   Where, *[***LDAP-GRP1***], [***LDAP-GRP2***], [***LDAP-GRP3***]* are the login groups containing users needing access to Hue.

3. Click Save Changes.
4. Restart the Hue service.

Options for importing and syncing LDAP users and groups in Hue

Configuring Hue for Lightweight Directory Access Protocol (LDAP) enables you to import users and groups from a directory service, synchronize group membership manually or automatically at login, and authenticate users with LDAP.

There are four options to import and sync LDAP users and groups in Hue:

LDAP sync option	Description
Add/Sync LDAP user	Import and synchronize one user at a time
Sync LDAP users/groups	Synchronize user memberships in all groups
Add/Sync LDAP group	Import and synchronize all users in one group
sync_groups_on_login	Automatically synchronize group membership at login

The following flowchart shows the process and the options to import and synchronize new users or groups in Hue:

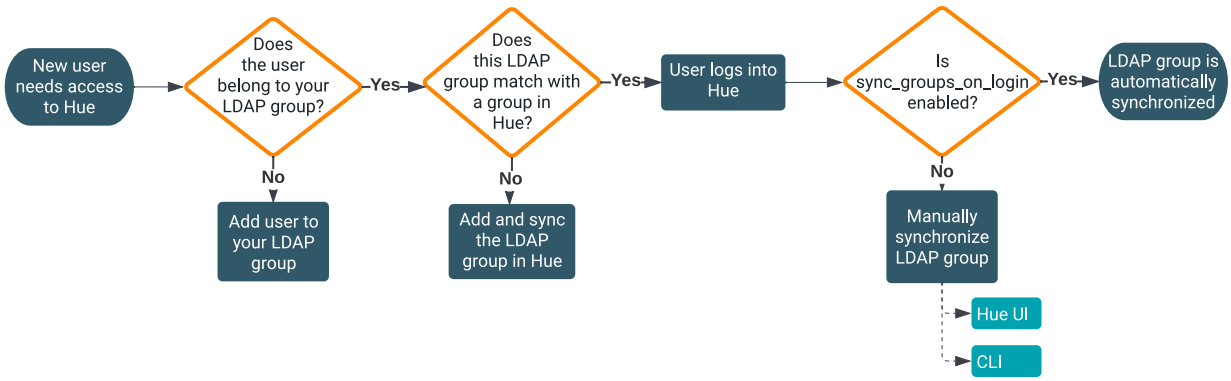

Importing a group from LDAP creates a group in Hue. When you synchronize a group, Hue checks the user's group membership in LDAP and synchronizes it to the corresponding group in Hue. To synchronize an LDAP group with Hue, the group must be imported in the Hue database.

For example, if a user belongs to 10 LDAP groups, but only 5 groups are present in Hue, then only these 5 groups are synced when new users are added to these groups. This mechanism helps to avoid including irrelevant group data in the Hue database.

If you have multiple LDAP groups that are synchronized with Hue, then you can automate the synchronization process by turning on the sync_groups_on_login option in the Hue Advanced Configuration Snippet. However, this process can be burdensome if you have a large number of users logging in and authenticating simultaneously or new users getting added to the LDAP group, as multiple synchronization requests are triggered which could cause collisions on database writes. An alternative approach is to synchronize users using the command line option, which

you can script and automate as a cron job. To manually synchronize LDAP groups having the newly added users that need to be added to Hue, run the following command separately for each LDAP group:

```
$HUE_HOME/build/env/bin/hue import_ldap_group --import-members [***LDAP-GROU
P-NAME***] --cm-managed
```

Importing and synchronizing users and groups with an LDAP server in Hue

You can import and synchronize one user at a time, synchronize all user memberships in all groups, import and synchronize all users in one group, or enable synchronization of group memberships automatically when users in those groups log in to Hue.

Before you begin

To synchronize your Hue users and groups with your LDAP server:

- Hue must be configured to authenticate with LDAP.
- The logged in user must have Hue superuser permissions.

Note: If the backend is Isilon, then pre-create the users home folder /user/[***USERNAME***] and assign the necessary permissions to the folder owner/group. Failing to do this,causes the LDAP sync to fail because Hue assigns the home folder's group permissions to be the username, but Isilon does not have a group with the username existing (unless it was pre-created.)

About this task

Important: Hue does not support importing all groups at once.

Procedure

1. Log in to Hue as a superuser.
2. Go to User Admin Users .

   The **User Admin** page is displayed.
3. To import and synchronize one LDAP user in Hue:
   a) Click Add/Sync LDAP user.
   b) Add a username, check Create home directory, and click Add/Sync user.
4. To synchronize group memberships for LDAP users who have already been imported to Hue:
   a) Click Sync LDAP users/groups.
   b) Select the Create home directories option and click Sync.

   Note: This synchronizes group memberships with the LDAP server.
5. To import and synchronize one LDAP group containing its users:
   a) Click Add/Sync LDAP group.
   b) Check Create home directories, and click Sync.
6. To configure Hue to automatically synchronize LDAP groups and their users when they log in to Hue:
   a) Log in to Cloudera Manager as an Administrator.
   b) Go to Clusters Hue service Configuration and enter the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

   ```
   [desktop]
   [[ldap]]
     sync_groups_on_login=true
   ```

   c) Click Save Changes.

d) Restart the Hue service.

Note: LDAP sync_groups_on_login only works with search bind authentication. This process can be burdensome if you have a large number of users logging in and authenticating simultaneously or new users getting added to the LDAP group, as multiple synchronization requests are triggered which could cause collisions on database writes.

7. To synchronize LDAP groups having the newly added users that need to be added to Hue, run the following command separately for each LDAP group:

```
$HUE_HOME/build/env/bin/hue import_ldap_group --import-members [***LDAP-
GROUP-NAME***] --cm-managed
```

You can script and automate this process using a cron job.

Locking an account after invalid login attempts

As a security measure, you can configure Hue to lock an account after a set number of unsuccessful or invalid login attempts by specifying the number of attempts in the login_failure_limit parameter and setting the login_lock_out_at_failure parameter to true in the Hue Advanced Configuration Snippet in Cloudera Manager.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue Configuration and add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[desktop]
[[auth]]
login_failure_limit=[***NUMBER-OF-ATTEMPTS-BEFORE-ACCOUNT-LOCKOUT***]
login_lock_out_at_failure=true
```

Replace [***NUMBER-OF-ATTEMPTS-BEFORE-ACCOUNT-LOCKOUT***] with the number of unsuccesful login attempts after which you want to lock an account. For example, if you set login_failure_limit to 3, then the user will be locked out on the third invalid attempt.

3. Click Save Changes.
4. Restart the Hue service to apply stale configurations.

   You can use the Rolling Restart option to minimize downtime.

Results

A user's account will be locked out after crossing the set number of invalid login attempts and the following message is displayed "Account locked: too many login attempts. Contact an admin to unlock your account."

Unlocking locked out user accounts in Hue

If Hue is configured to lock out a user account after a set number of invalid login attempts, then the users get an "Account locked: too many login attempts. Contact an admin to unlock your account." error message. As an Admin, you can unlock user accounts from the Hue web interface.

Procedure

1. Log in to the Hue web interface as an Administrator or a Hue superuser.
2. Expand your user name on the left assist panel and click Administer Users.

   The **User Admin** page is displayed.
3. Click on the username that you want to unlock.

4. Go to the **Step 3. Advanced** tab and select the Unlock Account option.
5. Click Update User.

Results

The user should be able to log in to Hue.

Creating a group in Hue

By creating groups, you can club certain permissions that you want to assign to specific users in your organization.

Procedure

1. Sign in to the Hue UI as a superuser.
2. Go to the Groups tab.

   The **Groups** page displays the list of existing groups, if any.
3. Click Add group.

Managing Hue permissions

Permissions for Hue applications are granted to groups, with users gaining permissions based on their group membership. Group permissions define the Hue applications visible to group members when they log in to Hue and the application features available to them. There is a fixed set of Hue permissions. You cannot add or modify permissions. However, you can apply permission to group(s).

Procedure

1. Sign in to the Hue UI as a superuser.
2. From the **User Admin** page, go to the Permissions tab.

   The **Permissions** page displays the list of all the available permissions.
3. Click a permission that you want to assign to a group(s).

   The **Edit [permission name]** page is displayed.
4. Select the group(s) on which you want to apply the permission and click Update permission.

   The “Permission information updated successfully” message is displayed.

Resetting Hue user password

The first user logging into Hue after its initial installation becomes the first superuser. Even if a user does not log into the Hue UI, the first security scan may log in creating an initial user and therefore resulting in an unknown username and password. You can change the password for a user if you know the username or you can create a new superuser user and then use it to log in to Hue and change the password for a user.

Procedure

1. Sign in to the Hue server as the root user and go to the Hue home directory.
2. If you know the user ID of the currently logged in user, then reset the password by running the following command:

   ```
   build/env/bin/hue changepassword [***USER-ID***] --cm-managed
   ```

   Replace the *USER-ID* with the actual ID of the user.

3. If you do not know the user ID of the user whose password you want to change, then create a new Hue admin user by running the following command:

```
build/env/bin/hue createsuperuser --cm-managed
```

After creating a new admin user, log in to Hue and reset the password for a given user ID.

Assigning superuser status to an LDAP user

The Hue User Admin application provides two levels of privileges: users and superusers. The superusers have administrative privileges.

About this task

Users can change their name, email address, and password. They can log in to Hue and run Hue applications according to their group permissions.

Superusers can perform administrative functions such as:

- Add and delete users and groups
- Import and sync users and groups from an LDAP server
- Assign group permissions
- Promote users to superusers and vice versa.

Hue superusers have no special privileges to the underlying Cloudera cluster services. Ranger is used to add those privileges.

Important: On a non-secure cluster, the first user to log in to Hue without LDAP authentication becomes the first superuser.

Procedure

In a secure cluster with LDAP deployed, there are three ways to assign superuser status to a user:

- With desktop.auth.backend.AllowAllBackend set for the Authentication Backend property in Cloudera Manager temporarily enabled, assign superuser status and synchronize one user to the LDAP server.
- With desktop.auth.backend.LdapBackend set for the Authentication Backend property in Cloudera Manager, run a Hue shell command to apply superuser status.
- Enable multiple backends so that the first user to log on still works when integrated with LDAP.

Configuring file and directory permissions for Hue

Per user home directories are created on HDFS when you create a new user in Hue, which the user can use to store and retrieve files using the Hue File Browser. You must either configure Hue to use the same permissions of the HDFS umask or use different permissions when a user creates files and directories from the Hue File Browser. It is recommended to define the home directory permissions before creating users in Hue.

About this task

When you set up Cloudera, you may have set a umask for all files and directories that would be created on HDFS using the “fs.permissions.umask-mode” parameter. The default HDFS umask is 0022. At this point, you can allow Hue to use the permisions as defined by HDFS umask or you can configure Hue to use a different set of permissions. This is controlled by the following two parameters in Hue: “home_dir_permissions” and “use_home_dir_permiss ions”. The “use_home_dir_permissions” parameter is set to true by default.

To allow Hue to create files and directories with permissions different than the HDFS umask, set the following in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field in Cloudera Manager:

```
[useradmin]
home_dir_permissions=[***HUE-FILE-DIR-PERMISSIONS***]
use_home_dir_permissions=true
```

For example:

```
[useradmin]
home_dir_permissions=0700
use_home_dir_permissions=true
```

To allow Hue to use the same file and directory permissions as the HDFS umask, set the following in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field in Cloudera Manager:

```
[useradmin]
use_home_dir_permissions=false
```

If you have not defined the home directory permissions in Hue by setting the value of the home_dir_permissions property in the Hue Advanced Configuration before creating users in Hue, then you can change the permissions for those users later to prevent unauthorized access.

Procedure

1. SSH in to the Hue server host as an Administrator.
2. Run the following command to change the permission for a user:

```
hdfs dfs -chmod 700 /user/[***USERNAME***]
```

User authentication in Hue

Cloudera services do not authenticate each user that logs in to Hue. The Cloudera services authenticate Hue and trust that Hue has authenticated its users. In a most typical configuration, Hue users can be authenticated with an LDAP server and the Cloudera users can be authenticated with Kerberos. You can also use SAML for Single Sign-on (SSO) authentication.

After Hue is authenticated by a service such as Hive, Hue impersonates the user requesting the use of that service, for example, to create a Hive table. In this case, the Hive service uses Apache Ranger to ensure that the group to which the user belonged is authorized for that action (to create a Hive table).

Note: By default, the Hue session uses a secure cookie protocol.

Authenticating Hue users with Kerberos

For Hue to work properly with a Cloudera cluster that uses Kerberos for authentication, the Kerberos Ticket Renewer role must be added to the Hue service.

About this task

Use the Cloudera Manager Admin Console to add the Kerberos Ticket Renewer role to each host with a Hue Server role instance. The Hue Kerberos Ticket Renewer renews only those tickets created for the Hue service principal: hue/*HOSTNAME@REALM-NAME*. The Hue principal impersonates other users for applications within Hue such as the Job Browser, File Browser, and so on. Other services, such as HDFS and MapReduce, do not use

the Hue Kerberos Ticket Renewer. Instead these other services handle ticket renewal as needed by using their own mechanisms.

Procedure

1. On the Cloudera Manager home page, select the Hue service.
2. On the Hue service page, click the Instances tab.
3. On the Instances page, click Add Role Instances on the right side of the page. This launches the Add Role Instances wizard.
4. To add a Kerberos Ticket Renewer role instance to the same host that has the Hue server on your Cloudera cluster, click Select hosts under Kerberos Ticket Renewer:

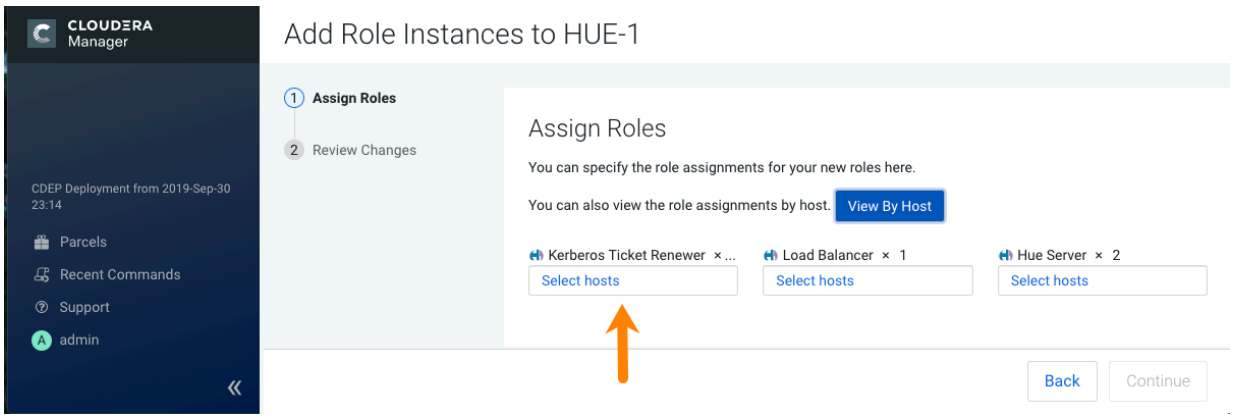

   To check which host has the Hue Server role instance, click View By Host, which launches a table that lists all the hosts in your Cloudera cluster and shows all the roles each host already has.
5. In the host selection dialog box, after selecting the host where you want to add the Kerberos Ticket Renewer role instance, click OK, and Cloudera Manager adds the role instance.
6. After processing the request to add the role instance, Cloudera Manager returns you to the Instances page and prompts you to restart the service. Click the Restart the service (or the instance)... link so the configuration change can take effect.
7. After the services have restarted, click Finish to return to the Instances page.

   Repeat these steps for each Hue Server role on your cluster.

What to do next

Troubleshooting the Kerberos Ticket Renewer:

If the Hue Kerberos Ticket Renewer does not start, check the configuration of your Kerberos Key Distribution Center (KDC). Look at the ticket renewal property, maxrenewlife, to ensure that the principals, hue/*<HOST_NAME>* and krbtgt, are renewable. If these principals are not renewable, run the following commands on the KDC to enable them:

```
kadmin.local: modprinc -maxrenewlife 90day krbtgt/<YOUR_REALM.COM>
kadmin.local: modprinc -maxrenewlife 90day +allow_renewable hue/
<HOST_NAME>@<YOUR_REALM>
```

Authenticating Hue users with LDAP

Configuring Hue for Lightweight Directory Access Protocol (LDAP) enables you to import users and groups from a directory service, synchronize group membership manually or automatically at login, and authenticate with an LDAP server.

Hue supports Microsoft Active Directory (AD) and open standard LDAP such as OpenLDAP and Forgerock OpenDJ Directory Services.

Integrating Hue with LDAP

When Hue is integrated with LDAP, users can use their existing credentials to authenticate and inherit their existing groups transparently. There is no need to save or duplicate any employee password in Hue.

When authenticating using LDAP, Hue validates login credentials against an LDAP directory service if Hue is configured with the LDAP authentication backend (desktop.auth.backend.LdapBackend) in Cloudera Manager.

The LDAP authentication backend automatically creates users that do not exist in Hue by default. Hue needs to import users to properly perform the authentication. Passwords are never imported when importing users. You can disable automatic import of users by setting the create_users_on_login property in the Cloudera Manager Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field to false.

```
[desktop]
[[ldap]]
create_users_on_login=false
```

The purpose of disabling the automatic import is to allow only a predefined list of manually imported users to login.

Preserving the case of the usernames

If you are using mixed case, upper case, or Camel case for usernames in your LDAP directory, then you must add the following configurations in Hue's Advanced Configuration Snippet (Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini), so that the user names are not over written in lower case in the Hue's database. Set the values of these properties to true or false depending on your requirement:

```
[desktop]
[[auth]]
ignore_username_case=true
force_username_uppercase=true/false
force_username_lowercase=true/false
[[ldap]]
ignore_username_case=true
force_username_uppercase=true/false
force_username_lowercase=true/false
```

Binding Hue with LDAP

There are two ways to bind Hue with an LDAP directory service:

Search Bind

The search bind mechanism for authenticating will perform an ldapsearch against the directory service and bind using the found distinguished name (DN) and password provided. This is the default method of authentication used by Hue with LDAP.

You can restrict the search process by configuring the following two properties under the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini [desktop] > [[ldap]] > [[[users]]] section.

Property	Description
user_filter	General LDAP filter to restrict the search. Default: "objectclass=*"
user_name_attr	The attribute that will be considered the username to be searched against. Typical attributes to search for include: uid, sAMAccountName. Default: sAMAccountName

With the above configuration, the LDAP search filter takes the following form:

```
(&(objectClass=*)(sAMAccountName=[***USERNAME-ENTERED-BY-USER***
]))
```

Note: Setting search_bind_authentication=true tells Hue to perform an LDAP search using the bind credentials specified for the bind_dn and bind_password configuration properties. Hue will start searching the subtree starting from the base DN specified for the base_dn property. It will then search the base DN for an entry whose attribute, specified in user_name_attr, has the same value as the short name provided on login. The search filter, defined in user_filter will also be used to limit the search.

Direct Bind

The direct bind mechanism for authenticating binds to the LDAP server using the username and password provided at login.

Hue authenticates (without searching) in one of two ways:

- NT Domain (nt_domain): (Only for use with Microsoft Active Directory) Hue binds to the AD with username@domain using the User Principal Names (UPN) to bind to the LDAP service. This AD-specific property allows Hue to authenticate with AD without having to follow LDAP references to other partitions. This typically maps to the email address of the user or the user's ID in conjunction with the domain. Default: mycompany.com.
- Username Pattern (ldap_username_pattern): Bind to open standard LDAP with full path of directory information tree (DIT). It provides a template for the DN that is ultimately sent to the directory service when authenticating. The [***USERNAME***] parameter is replaced with the username provided at login.

  Default:

  ```
  "uid=[***USERNAME***],ou=People,dc=mycompany,dc=com"
  ```

  Note: Setting search_bind_authentication=false tells Hue to perform a direct bind to LDAP using the credentials provided (not bind_dn and bind_password specified in the Advanced Configuration Snippet). There are two ways direct bind works depending on whether the nt_domain property is specified in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini :

  - nt_domain is specified: This is used to connect to an Active Directory service. In this case, the UPN is used to perform a direct bind. Hue forms the UPN by concatenating the short name provided at login with the nt_domain. For example, [***SHORT-NAME***]@[***NT-DOMAIN***]. The ldap_username_pattern property is ignored.
  - nt_domain is not specified: This is used to connect to all other directory services (can handle AD, but nt_domain is the preferred way for AD). In this case, ldap _username_pattern is used and takes the following form:

    ```
    cn=[***USERNAME-PROVIDED-AT-LOGIN***],dc=example,dc=
    com
    ```

Note: Username pattern does not work with AD because AD inserts spaces into the UID which Hue cannot process.

Encryption

To prevent credentials from transmitting in the clear, encrypt with LDAP over SSL, using the LDAPS protocol on the LDAPS port, which uses port 636 by default. An alternative, is to encrypt with the StartTLS operation using

the standard LDAP protocol, which uses port 389 by default. Cloudera recommends LDAPS. You must have a CA Certificate in either case.

Table 1: Hue Supported LDAP authentication and encryption methods

LDAP Auth Action	Encrypted (LDAPS)	Encrypted (LDAP+TLS)	Not Encrypted (LDAP)
Search Bind	AD, LDAP	AD, LDAP	AD, LDAP
Direct Bind - NT Domain	AD	AD	AD
Direct Bind - User Pattern	LDAP	LDAP	LDAP

Prerequisites

To authenticate Hue users with LDAP, you must have:

- LDAP server
- Bind account (or support for anonymous binds)
- Cloudera Manager access with Full Administrator permissions
- [optional] LDAP server with LDAPS or StartTLS encryption.

Important: To authenticate securely, configure your LDAP server with either LDAP over SSL (LDAPS) or StartTLS encryption. Both methods require a Certificate Authority (CA) chain in a .pem file.

Configuring authentication with LDAP and Search Bind

Search Bind authentication executes ldapsearch against one or more directory services and binds with the distinguished name (DN) and password. Hue searches the subtree from the base distinguished name. If the LDAP Username Attribute is set, Hue looks for an entry whose attribute has the same value as the short name given at login.

About this task

Important: Search Binding works with all directory service types. It is also the only method that allows synchronizing groups at login (set with sync_groups_on_login in a safety-valve).

Video: Authenticate Hue with LDAP and Search Bind

Figure 1: Video: Authenticate Hue with LDAP and Search Bind

https://www.youtube.com/embed/pCgUxQ8CU4o

Procedure

1. Log on to Cloudera Manager and click Hue.
2. Click the Configuration tab and filter by scope=Service-wide and category=Security.
3. Set the following required properties:

Authentication Backend	desktop.auth.backend.LdapBackend
LDAP URL	• ldaps://<ldap_server>:636 if using Secure LDAP • ldap://<ldap_server>:389 if not using encryption Note: If ldaps:// is specified in the LDAP URL, then do not set LDAP TLS.
Enable LDAP TLS	• TRUE if not using Secure LDAP (LDAPS) but want to establish a secure connection using TLS • FALSE if using LDAPS or not encrypting
LDAP Server CA Certificate	/path_to_certificate/cert.pem
LDAP Search Base	DC=mycompany,DC=com

LDAP Bind User Distinguished Name	username@domain
LDAP Bind Password	bind_user_password
Use Search Bind Authentication	TRUE
Create LDAP users on login	TRUE

Note: To encrypt with TLS, set LDAP URL to ldap://<ldap_server>:389 and check Enable LDAP TLS. For a proof of concept without encryption, use ldap://<ldap_server>:389, remove the value for LDAP Server CA Certificate, and uncheck Enable LDAP TLS.

4. You can optionally improve search performance with attributes and filters:

LDAP User Filter	objectclass=user (default = *)
LDAP Username Attribute	sAMAccountName (AD default), uid (LDAP default)
LDAP Group Filter	objectclass=group (default = *)
LDAP Group Name Attribute	cn (default)
LDAP Group Membership Attribute	member (default)

Note: With the user settings in the table above, the LDAP search filter has the form: (&(objectClass=user)(sAMAccountName=<user entered username>)).

5. Add any valid user and/or valid group to quickly test your LDAP configuration:

LDAP Username for Test LDAP Configuration	Any valid user
LDAP Group Name for Test LDAP Configuration	Any valid group

6. Click Save Changes.
7. Test your LDAP configuration, and when successful click Restart Hue.

Note: The syntax of Bind Distinguished Name differs per bind method:

- Search Bind: username@domain
- Direct Bind with NT Domain: username
- Direct Bind with Username Pattern: DN string (full DIT path)

Do not use if anonymous binding is supported.

You can test ldapsearch at the command line as follows:

```
LDAPTLS_CACERT=/<path_to_cert>/<ca_certificate> ldapsearch -H ldaps://<l
dap_server>:636 \
-D "<bind_dn>" -w <bind_password> -b <base_dn> "samaccountname=<user>"
```

Note: To run ldapsearch with a CA certificate, you may need to install ldap_utils on Debian/Ubuntu and openldap-clients on RHEL/CentOS.

Configuring authentication with LDAP and Direct Bind

To authenticate with Direct Binding, Hue needs either the User Principal Name (UPN) for Active Directory, or the full path to the LDAP user in the Directory Information Tree (DIT) for open standard LDAP.

About this task

Important: Direct binding only works with one domain. For multiple directories, use search bind.

Video: Authenticate Hue with LDAP and Direct Bind

Figure 2: Video: Authenticate Hue with LDAP and Direct Bind

https://www.youtube.com/embed/w9PQKytKr1A

To directly bind to an Active Directory/LDAP server with NT domain:

Procedure

1. Log in to Cloudera Manager and click Hue.
2. Click the Configuration tab and filter by scope=Service-wide and category=Security.
3. Set the following LDAP properties:

Authentication Backend	desktop.auth.backend.LdapBackend
LDAP URL	• ldaps://<ldap_server>:636 if using Secure LDAP • ldap://<ldap_server>:389 if not using encryption Note: If ldaps:// is specified in the LDAP URL, then do not set LDAP TLS.
Enable LDAP TLS	• TRUE if not using Secure LDAP (LDAPS) but want to establish a secure connection using TLS • FALSE if using LDAPS or not encrypting
LDAP Server CA Certificate	/path_to_certificate/cert.pem
LDAP Search Base	DC=mycompany,DC=com
LDAP Bind User Distinguished Name	<username> Only the username is required for Direct Bind. There is no need to specify the domain.
LDAP Bind Password	bind_user_password
Active Directory Domain	<your NT domain>
Use Search Bind Authentication	FALSE
Create LDAP users on login	TRUE

4. Click Save Changes
5. Test your LDAP configuration, and when successful, click Restart Hue.

   To directly bind to an open standard LDAP server with a username pattern:

   a. Remove the value for the Active Directory Domain.
   b. Set both LDAP Username Pattern and LDAP Bind User Distinguished Name to a DN string that represents the full path of the directory information tree, from UID to top level domain.

   Note: When using Direct Bind, set the LDAP Search Base property. This is not for authentication because you can log in to Hue without it, but to synchronize Hue with the LDAP server.

Configuring Hue for authentication against multiple LDAP/Active Directory servers

Some organizations have more than one LDAP or Active Directory (AD), and users in one LDAP/AD may not be present in the other. Hue supports the ability to authenticate users against multiple LDAP/AD servers.

Before you begin

Before attempting LDAP authentication against multiple servers, ensure you have configured LDAP synchronization for each server, as described in the previous sections. As long as users and groups are synchronised across LDAP and Hue, authentication should work.

Procedure

1. Log in to Cloudera Manager as an Administrator.

2. Go to Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini , and add the following lines for each LDAP server:

```
[desktop]
[[ldap]]
[[[ldap_servers]]]

[[[[AD1.TEST.COM]]]]
ldap_url=ldap://w2k8-ad1
search_bind_authentication=true
create_users_on_login=true
base_dn="cn=users,dc=ad1,dc=test,dc=com"
bind_dn="cn=Administrator,cn=users,dc=ad1,dc=test,dc=com"
bind_password="[***PASSWORD***]"
[[[[AD2.TEST.COM]]]]
ldap_url=ldap://w2k8-ad2
search_bind_authentication=true
create_users_on_login=true
base_dn="cn=users,dc=ad2,dc=test,dc=com"
bind_dn="cn=Administrator,cn
bind_password="[***PASSWORD***]"
```

   The above code snippet is for a demo LDAP server called "AD1.TEST.COM" and "AD2.TEST.COM".

3. Click Save Changes.
4. Restart the Hue service.

Results

When you log into Hue or while synchronizing LDAP users from the Hue web interface, you need to select the LDAP/AD server from the dropdown list.

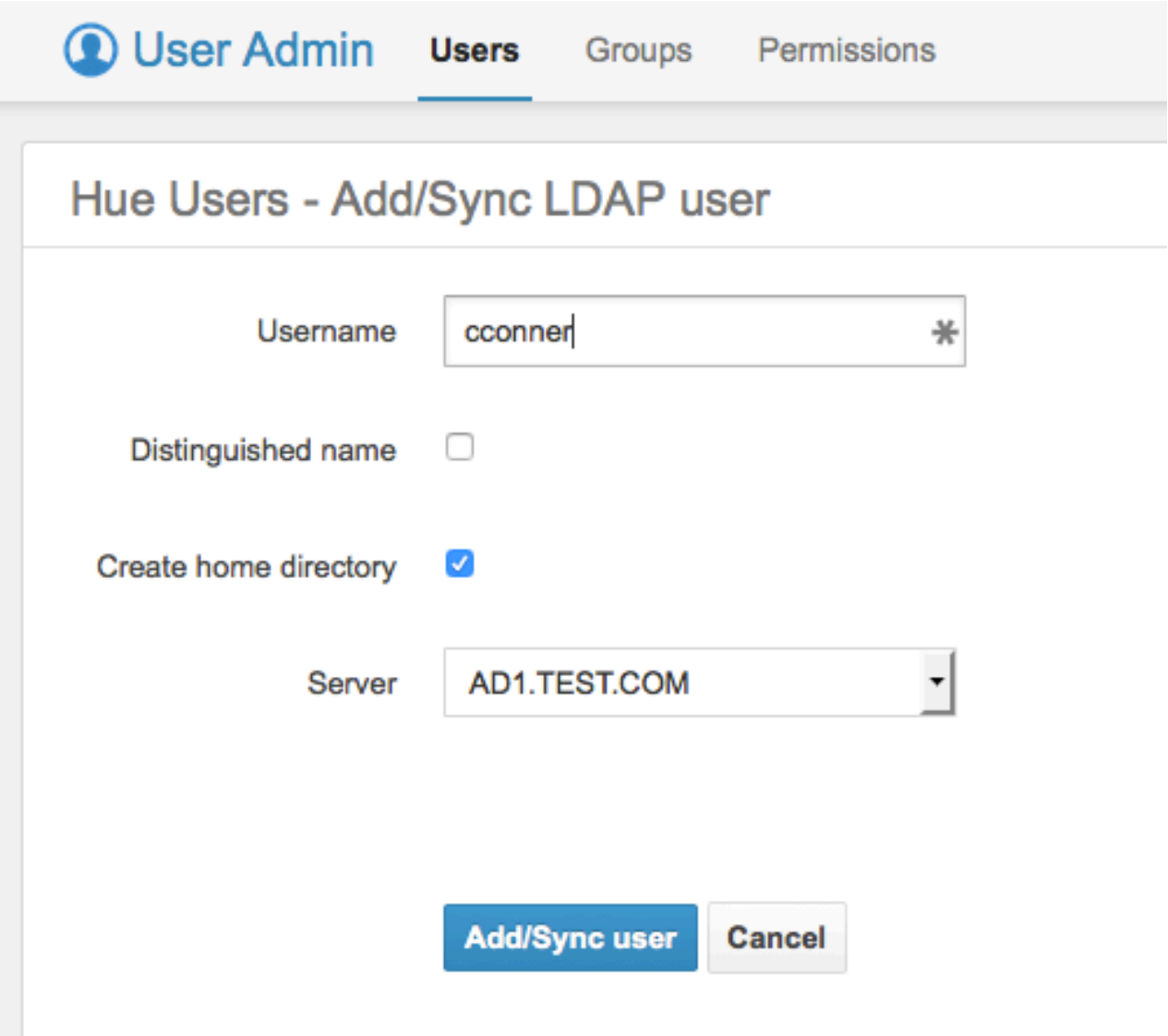

Testing the LDAP configuration

You can test your Hue LDAP configuration without restarting the Hue service. Add the values and save your changes.

Procedure

1. Configure Hue to authenticate with LDAP by using search bind or direct bind.
2. Add a user and group name for Test LDAP Configuration.
3. Click Save Changes.
4. Select ActionsTest LDAP Configuration.

5. Click Test LDAP Configuration:

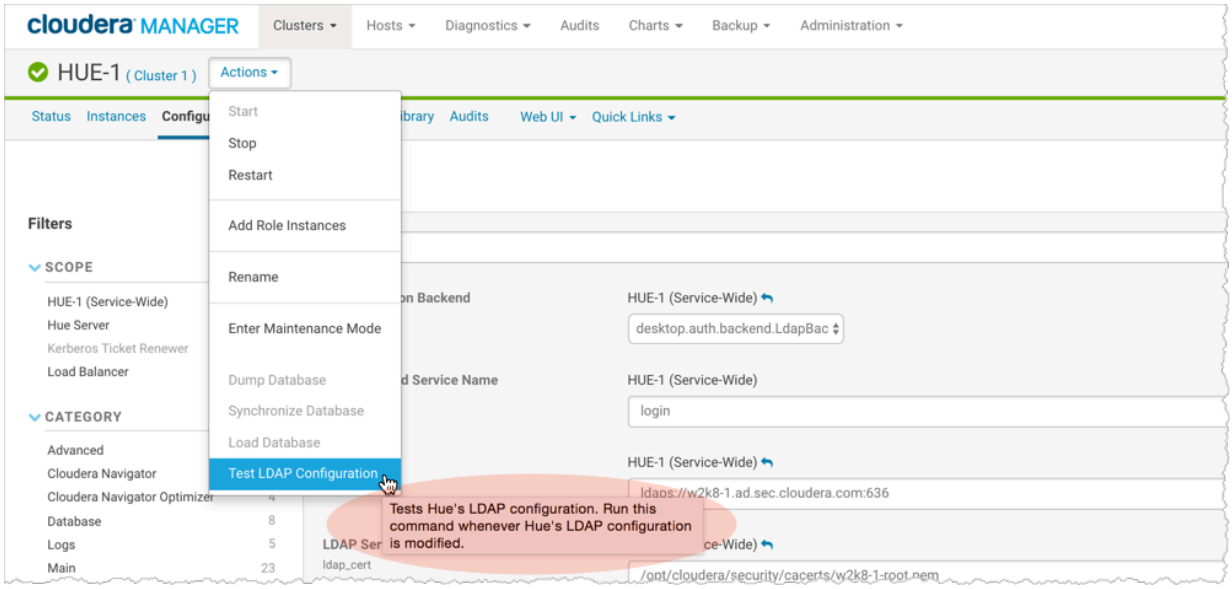

6. Click Restart Hue. When the test succeeds, log in to the Hue Web UI.

Configuring group permissions

You can configure permissions for members of groups on the Groups tab of the Hue User Admin application.

About this task

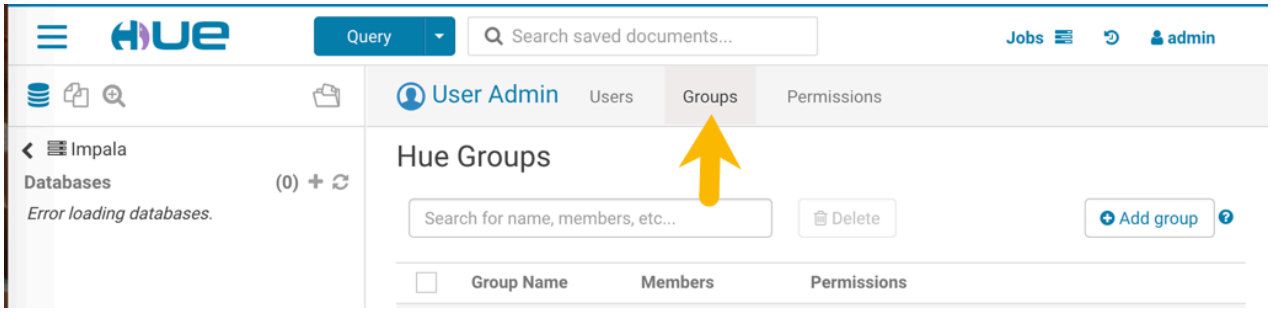

Note: A best practice is to remove all permissions from the default group and assign permissions as appropriate to your own groups.

Procedure

1. Log on to the Hue UI as a superuser.
2. Go to User AdminGroups.
3. Click the name of the group you want to alter.
4. Deselect any users that you do not want to change (all users in the group are selected by default).
5. Select or deselect the permissions you want to apply or remove.
6. Click Update Group.

Enabling LDAP authentication with HiveServer2 and Impala

LDAP authentication with HiveServer2 and Impala can be enabled by setting the auth_username and auth_password properties under the [beeswax] section for Hive and the [impala] section for Impala in a Cloudera Manager safety valve configuration property.

Important: Set these properties in the Cloudera Manager Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini property.

auth_username	LDAP username of the Hue user to be authenticated to the server.
auth_password	LDAP password of the Hue user to be authenticated to the server.

For example:

```
[beeswax]
   auth_username=<HIVESERVER2_LDAP_USER_NAME>
   auth_password=<HIVESERVER2_LDAP_PASSWORD>
[impala]
   auth_username=<IMPALA_LDAP_USER_NAME>
   auth_password=<IMPALA_LDAP_PASSWORD>
```

These login details are only used by Impala and Hive to authenticate to the LDAP server. The Impala and Hive services trust Hue to have already validated the user being impersonated instead of passing on the credentials.

LDAP properties

These are the properties you can use to configure LDAP for Hue in Cloudera Manager or in the hue.ini file for unmanaged clusters.

Property Name	Description and Syntax
General Hue LDAP Properties	
Authentication Backend backend	Authentication Mode. Select desktop.auth.backend.LdapBackend. Multiple backends are allowed. Create a list and add it to the Hue safety-valve.
LDAP URL ldap_url	URL for the LDAP server. Syntax: ldaps://<ldap_server>:<636> or ldap://<ldap_server>:<389> Important: To prevent usernames and passwords from transmitting in the clear, use ldap s:// or ldap:// + "Enable LDAP TLS".
Create LDAP users on login create_users_on_login	Flag to create new LDAP users at Hue login. If true, any user who logs into Hue is automatically created. If false, only users that exist in useradmin can log in.
Direct Bind Properties	
Active Directory Domain nt_domain	For direct binding with Microsoft Active Directory only. Typically maps to the user email address or ID in conjunction with the domain. Allows Hue to authenticate without having to follow LDAP references to other partitions. Hue binds with User-Principal-N ame (UPN) if provided. Example: ad.<mycompany>.com Important: Do not use nt_domain with LDAP Username Pattern or Search Bind.
LDAP Username Pattern ldap_username_pattern	For direct binding with LDAP (non-Active Directory) only (because AD uses UPNs which have a space in them). Username Pattern finds the user attempting to login into LDAP by adding the username to a predefined DN string. Use <username> to reference the user logging in. An example is "uid=<username>,ou=people,dc=mycompany,dc=com".
Search Bind Properties	
Use Search Bind Authentication search_bind_authentication	Flag to enable/disable Search Bind.
LDAP Search Base base_dn	Distinguished name to use as a search base for finding users and groups. Syntax: dc=ad, dc=sec, dc=mycompany,dc=com

Property Name	Description and Syntax
Encryption Properties	
LDAP Server CA Certificate ldap_cert	Full path to .pem file with Certificate Authority (CA) chain used to sign the LDAP server certificate. If left blank, all certificates are trusted and otherwise encrypted usernames and passwords are vulnerable to attack.
Enable LDAP TLS use_start_tls	Flag to enable/disable encryption with the StartTLS operation.
Import / Sync Properties	
LDAP Bind User Distinguished Name bind_dn	Bind user. Only use if LDAP/AD does not support anonymous binds. (Typically, LDAP supports anonymous binds and AD does not.) Bind User differs per auth type: • Search Bind: username@domain • Direct Bind with NT Domain: username • Direct Bind with Username Pattern: DN string (and same as LDAP Username Pattern)
LDAP Bind Password bind_password	Bind user password.
Filter Properties	
LDAP User Filter user_filter	General LDAP text search filter to restrict search of valid users. Only used by Search Bind authentication and LDAP Sync. The default is objectclass=* but can differ. For example, some LDAP environments support Posix objects for *nix authentication and the user filter might need to be objectclass=posixAccount.
LDAP Username Attribute user_name_attr	Username to search against (the attribute in LDAP that contains the username). Typical attributes include sAMAccountName (default for AD/LDAP) and uid (LDAP default). Maintain case sensitivity when setting attributes for AD/LDAP.
LDAP Group Filter group_filter	General LDAP text search filter to restrict search of valid groups. Only used by LDAP Sync (not authentication). If left blank, no filtering is used and all groups in LDAP are synced. The default is objectclass=* but can differ. For example, some LDAP environments support Posix objects for *nix authentication and the user filter might need to be objectclass=posixGroup.
LDAP Group Name Attribute group_name_attr	Group name to search against (the attribute in LDAP that contains the groupname). If left blank, the default is "cn" (common name), that typically works with AD/LDAP. Maintain case sensitivity when setting attributes for AD/LDAP.
LDAP Group Membership Attribute group_member_attr	Attribute in the group that contains DNs of all the members.(Optional) - If left blank, the default is "memberOf" or "member", that typically works with Active Directory/LDAP.
Test Properties	
LDAP Username for Test LDAP Configuration test_ldap_user	Any user (ideally with low privileges) used to verify the LDAP configuration.
LDAP Group Name for Test LDAP Configuration test_ldap_group	Any group (and not necessarily one that includes the test user) used to verify the LDAP configuration.

Configuring LDAP on unmanaged clusters

If your clusters are not managed with Cloudera Manager, you must manually set the LDAP configuration properties in the hue.ini file.

Refer to the following examples of LDAP configurations in the hue.ini file:

Example of a Search Bind configuration encrypted with LDAPS:

```
[[custom]]
[[auth]]
backend=desktop.auth.backend.LdapBackend

[[ldap]]
ldap_url=ldaps://<hostname>.ad.sec.<domain_name>.com:636
search_bind_authentication=true
ldap_cert=/<path_to_cacert>/<cert_filename>.pem
use_start_tls=false
create_users_on_login=true
base_dn="DC=ad,DC=sec,DC=<domain_name>,DC=com"
bind_dn="<username>@ad.sec.<domain_name>.com"
bind_password_script=<path_to_password_script>/<script.sh>
test_ldap_user="testuser1"
test_ldap_group="testgroup1"

[[[users]]]
user_filter="objectclass=user"
user_name_attr="sAMAccountName"

[[[groups]]]
group_filter="objectclass=group"
group_name_attr="cn"
group_member_attr="member"
```

Example of a Direct Bind configuration for Active Directory encrypted with LDAPS:

```
[[ldap]]
ldap_url=ldaps://<hostname>.ad.sec.<domain_name>.com:636
search_bind_authentication=false
nt_domain=ad.sec.<domain_name>.com
ldap_cert=/<path_to_cacert>/<cert_filename>.pem
use_start_tls=false
create_users_on_login=true
base_dn="DC=ad,DC=sec,DC=<domain_name>,DC=com"
bind_dn="<username>"
bind_password_script=<path_to_password_script>/<script.sh>
...
```

Example of a Direct Bind configuration for Active Directory encrypted with StartTLS:

```
[[ldap]]
ldap_url=ldap://<hostname>.ad.sec.<domain_name>.com:389
search_bind_authentication=false
nt_domain=ad.sec.<domain_name>.com
ldap_cert=/opt/cloudera/security/cacerts/<cert_filename>.pem
use_start_tls=true
create_users_on_login=true
base_dn="DC=ad,DC=sec,DC=<domain_name>,DC=com"
bind_dn="<username>"
bind_password_script=<path_to_password_script>/<script.sh>
...
```

Authenticating Hue users with SAML

Hue supports SAML (Security Assertion Markup Language) for Single Sign-on (SSO) authentication.

The SAML 2.0 Web Browser SSO profile has three components:

- User Agent - Browser that represents you, the user, seeking resources.
- Service Provider (SP) - Service (Hue) that sends authentication requests to SAML.
- Identity Provider (IdP) - SAML service that authenticates users.

When a user requests access to an application, the Service Provider (Hue) sends an authentication request from the User Agent (browser) to the identity provider. The identity provider authenticates the user, sends a response, and redirects the browser back to Hue as shown in the following diagram:

Figure 3: SAML SSO protocol flow in a web browser

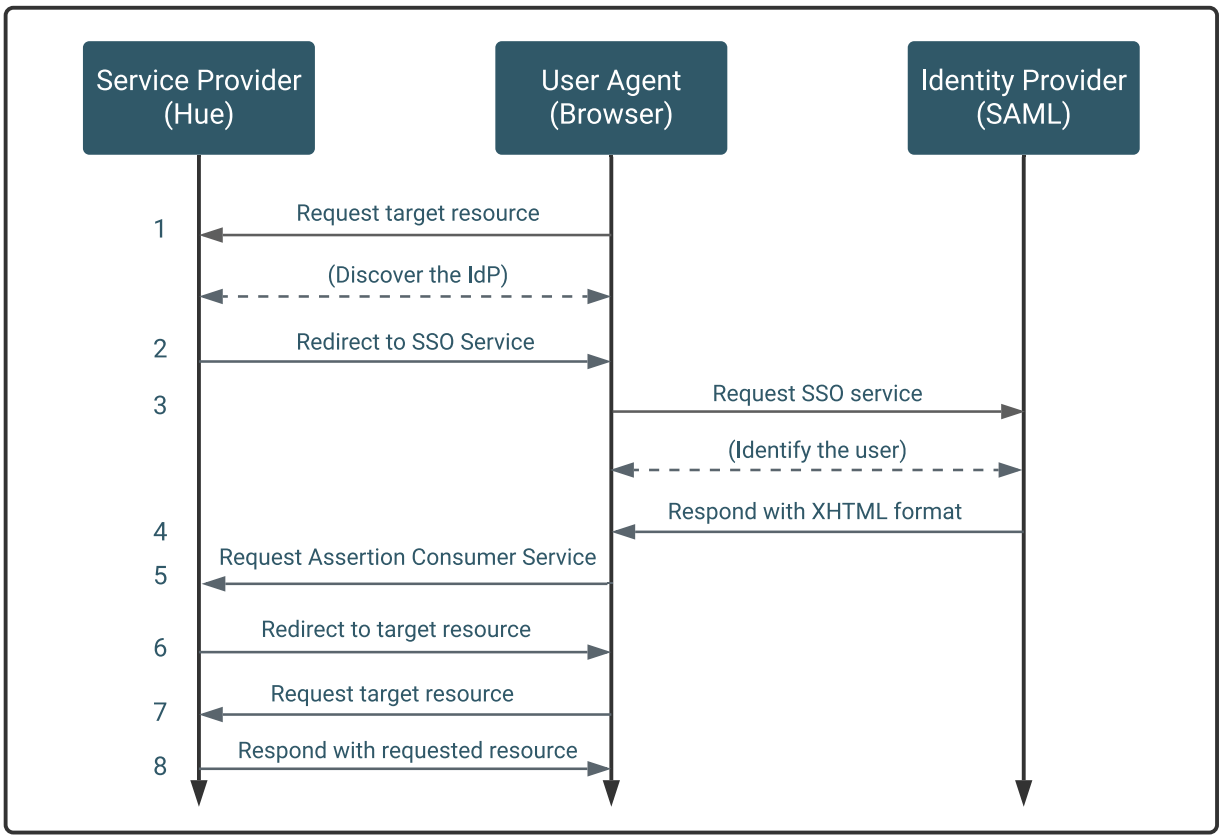

The Service Provider (Hue) and the identity provider use a metadata file to confirm each other's identity. Hue stores metadata from the SAML server, and the identity provider stores metadata from the Hue server.

Configuring SAML authentication on managed clusters

To configure Hue for SAML authentication on managed clusters, you must add the SAML authentication properties to the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini in Cloudera Manager.

Before you begin

These instructions assume that you have an Identity Provider set up and running. You can use any identity provider of your choice. For example, Okta, Ping Identity, and OpenAM.

Procedure

1. Log on to Cloudera Manager and go to Hue Configuration .
2. In the search text box, enter hue_safety_valve.ini to locate the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini.
3. Enter the SAML parameters into the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini text box. For example:

```
## Example Settings using Open AM:
[desktop]
redirect_whitelist="^\/.*$,^https:\/\/idp.example.com:8080\/.*$"
[[auth]]
backend=libsaml.backend.SAML2Backend
[libsaml]
want_response_signed=True
want_assertions_signed=True
xmlsec_binary=/usr/bin/xmlsec1
metadata_file=/opt/cloudera/security/saml/idp-metadata.xml
key_file=/opt/cloudera/security/saml/host.key
cert_file=/opt/cloudera/security/saml/host.pem
key_file_password=Config(
  key="key_file_password",
  help=_t("key_file_password password of the private key"),
  default=None) ## If using encrypted private key
username_source=nameid
name_id_format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
entity_id=[***HOST-BASE-NAME***]
logout_enabled=false
```

   If you are using an encrypted private key file, then you must specify the password in the key_file_password property. Or you can use an unencrypted private key file.

   To create an unencrypted private key file from an encrypted key:

   a. SSH into a terminal as a root user.
   b. Change to the directory where you have stored the ssl certificate key.
   c. Run the following command:

```
openssl rsa -in ssl_certificate.key -out ssl_certificate-nocrypt.key
```

   d. When prompted, enter the password that you use to access the ssl_certificate.key file.

      The output file (ssl_certificate-nocrypt.key) is an unencrypted PEM-formatted key.

   Note: For SLES distributions, the xmlsec binary may be located in /usr/local/bin/. If so:

   - Set xmlsec_binary=/usr/local/bin/xmlsec1 in the Hue Service Advanced Configuration Snippet.
   - Set LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/ in the Hue Service Environment Advanced Configuration Snippet.

4. Go to Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini) and comment or remove any Knox-SSO configurations, if present.

   Caution: Knox-SSO and SAML are incompatible and mutually exclusive. Hue authetication may fail with a redirection loop if you have the Knox-SSO and SAML configurations present in the Hue Advanced Configuration Snippet at the same time, as it confuses the authentication redirect to the IdP and back to Hue.

5. Click Save Changes, then select, ActionsRestart Hue.

Manually configuring SAML authentication

To manually configure Hue for SAML authentication on unmanaged clusters, you must add GCC Python libraries and install xmlsec1 tools on all the hosts in your cluster.

Before you begin

These instructions assume that you have an Identity Provider set up and running. You can use any identity provider of your choice. For example, Okta, Ping Identity, and OpenAM.

Important: You may need to disable cipher algorithms before manually configuring Hue for SAML authentication.

Procedure

1. Install the following libraries on all hosts in your cluster:

```
## RHEL/CentOS
yum install git gcc python-devel swig openssl
```

```
## Ubuntu/Debian
apt-get install git gcc python-dev swig openssl
```

```
## SLES
zypper install git gcc python-devel swig openssl make libxslt-devel libl
tdl-devel
```

2. Install xmlsec1 and xmlsec1-openssl on all hosts in the cluster:

   Important: Ensure that the xmlsec1 package is executable by the user, hue.

```
## RHEL/CentOS
yum install xmlsec1 xmlsec1-openssl
```

   Note: If xmlsec libraries are not available, use the appropriate epel repository:

```
## For RHEL/CentOS 7
wget http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-relea
se-7-6.noarch.rpm
rpm -ivh epel-release-7-6.noarch.rpm
```

```
## Ubuntu/Debian
apt-get install xmlsec1 libxmlsec1-openssl
```

```
## SLES (get latest version)
wget http://www.aleksey.com/xmlsec/download/xmlsec1-1.2.24.tar.gz
tar -xvzf xmlsec1-1.2.24.tar.gz
cd xmlsec1-1.2.24
./configure && make
make install
```

3. Copy metadata from your identity provider's SAML server and save it as an XML file on every host with a Hue server. For example, if your identity provider is Shibboleth, visit https://<idp_host>:8443/idp/shibboleth, copy the metadata content, and paste it into an XML file. Read the documentation of your identity provider for details on how to copy the XML of the SAML server metadata.

   Note: You may have to edit the copied metadata; for example, the identity provider's port number (8443) might be missing from its URL.

```
mkdir -pm 755 /opt/cloudera/security/saml/
```

```
cd /opt/cloudera/security/saml/
```

```
vim idp-<your idp provider>-metadata.xml
# Paste IdP SAML here and save
```

4. Add the files that the key_file and cert_file SAML properties point to for encrypted assertions and make sure you add this properties to the hue.ini configuration file.
   - The key_file parameter points to the location of the private key that is used to encrypt metadata. Its file format must be <file_name>.PEM.
   - The cert_file parameter points to the location of the X.509 certificate that is sent with encrypted metadata. Its file format must be <file_name>.PEM.

   Warning:

   Add the key and cert files even if you are not encrypting assertions. Hue checks for the existence and validity of these files even if they are not needed. They cannot be empty files. This is a known issue. If necessary, create a valid self-signed certificate:

   ```
   openssl req -x509 -newkey rsa:2048 -sha256 -days 3560 -nodes -keyout
    host.key -out host.pem -subj '/CN=Hue SAML'
   ```

Integrating your identity provider's SAML server with Hue

After Hue is configured for SAML authentication and restarted, copy the metadata that is generated by the Hue server and send it to your identity provider so they can configure the SAML server.

Before you begin

Ensure that you have configured Hue for SAML authentication and restarted it before you integrate your identity provider's SAML server with Hue.

Procedure

1. Ensure Hue is configured, restarted, and running.
2. Go to http://<hue_fqdn>:8889/saml2/metadata.
3. Copy the metadata and send it to your identity provider.
4. Ensure that your identity provider configures the SAML server with the Hue metadata. It is the same process you used to configure the Hue server with SAML metadata.

SAML properties

You can set SAML properties in the hue.ini file for unmanaged clusters. A subset of them can be set in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini for managed clusters.

Table 2: Table of SAML parameters

SAML parameter	Description
authn_requests_signed	Boolean, that when True, signs Hue-initiated authentication requests with X.509 certificate.
backend	Hard-coded value set to SAML backend library packaged with Hue (libsaml.backend.SAML2Backend).
base_url	URL that SAML Identity Provider uses for responses. Typically used in Load balanced Hue environments.
cert_file	Path to X.509 certificate sent with encrypted metadata. File format must be .PEM.
create_users_on_login	Boolean, that when True, creates users from OpenId, upon successful login.
entity_id	Service provider ID. Can also accept pattern where '<base_url>' is replaced with server URL base.

SAML parameter	Description
key_file	Path to private key used to encrypt metadata. File format must be .PEM.
key_file_password	Password used to decrypt the X.509 certificate in memory.
logout_enabled	Boolean, that when True, enables single logout.
logout_requests_signed	Boolean, that when True, signs Hue-initiated logout requests with an X.509 certificate.
metadata_file	Path to readable metadata XML file copied from Identity Provider.
name_id_format	Format of NameID that Hue requests from SAML server.
optional_attributes	Comma-separated list of optional attributes that Hue requests from Identity Provider.
required_attributes	Comma-separated list of required attributes that Hue requests from Identity Provider. For example, uid and email.
redirect_whitelist	Fully qualified domain name of SAML server: "^\/.*$,^https:\/\/<SAML_server_FQDN>\/.*$".
user_attribute_mapping	Map of Identity Provider attributes to Hue django user attributes. For example, {'uid':'username', 'email':'email'}.
username_source	Declares source of username as nameid or attributes.
want_response_signed	A boolean parameter, when set to True, requires SAML response wrapper returned by an IdP to be digitally signed by the IdP. The default value is False.
want_assertions_signed	A boolean parameter, when set to True, requires SAML assertions returned by an IdP to be digitally signed by the IdP. The default value is False.
xmlsec_binary	Path to xmlsec_binary that signs, verifies, encrypts/decrypts SAML requests and assertions. Must be executable by user, hue.

SAML properties that can be set for managed clusters

- redirect_whitelist [desktop]

  Set to the fully qualified domain name of the SAML server so that Hue can redirect to the SAML server for authentication.

  ```
  [desktop]
  redirect_whitelist=^\/.$,^https:\/\/<SAML_server_fully_qualified_domain_name>\/.$
  ```

  Note: Hue uses redirect_whitelist to protect itself from redirecting to unapproved URLs.

- backend [desktop]>[[auth]]

  Point to the SAML backend that is packaged with Hue:

  ```
  backend=libsaml.backend.SAML2Backend
  ```

- xmlsec_binary [libsaml]

  Point to the xmlsec1 library path:

  ```
  xmlsec_binary=/usr/bin/xmlsec1
  ```

  Note: To find the path, run: which xmlsec1

- metadata_file [libsaml]

  Point to the path of the XML file you created from the identity provider's metadata:

  ```
  metadata_file=/path/to/<your_idp_metadata_file>.xml
  ```

- key_file and cert_file [libsaml]

  To encrypt communication between Hue and the Identity Provider (IdP), you need a private key and certificate. The private key signs requests sent to the IdP, and decrypts messages from the IdP. The certificate is used to encrypt messages to Hue from the IdP, and must be provided to the IdP. Typically, the cert_file is shared by providing Hue's Service Provider metadata XML to the IdP admins, but you may also share a copy of the cert_file itself.

  The SAML certificate and private key must be the same on all Hue Server hosts, and can be self-signed, obtained from a commercial CA vendor, or from your internal PKI administrators. Both key_file and cert_file must be in PEM format.

  Users with password-protected certificates can set the property, key_file_password in the hue.ini file. Hue uses the password to decrypt the SAML certificate in memory and passes it to xmlsec1 through a named pipe. The decrypted certificate never touches the disk. This only works for POSIX-compatible platforms.

Troubleshooting SAML authentication

Before troubleshooting your SAML authentication configuration in Hue, enable DEBUG for the Hue Django logs that are located in /var/log/hue. In the Hue Web UI, go to the Home page, select Server Logs, and check Force Debug Level. For managed clusters, you can use Cloudera Manager to enable DEBUG by navigating to the Hue service, selecting the Configuration tab, check Enable Django Debug mode, click Save Changes, and then Restart.

SAML SSL error

OpenSSL might fail with this message:

```
SSLError: [Errno bad handshake] [('SSL routines', 'SSL3_CHECK_CERT_AND_ALGOR
ITHM', 'dh key too small')]
```

To resolve, append the following code to the file, /usr/java/<your_jdk_version>-cloudera/jre/lib/securi ty/java.security:

```
jdk.tls.disabledAlgorithms=MD5, RC4, DH
```

SAML decrypt error

The following error is an indication that you are using a slightly different SAML protocol from what Hue expects:

```
Error: ('failed to decrypt', -1)
```

To resolve:

1. Download and rename the fix-xmlsec1.txt Python script.

   ```
   wget https://www.cloudera.com/documentation/other/shared/fix-xmlsec1.txt -
   O fix-xmlsec1.py
   ```

2. Change permissions as appropriate, for example:

   ```
   chmod 755 fix-xmlsec1.py
   ```

3. In hue.ini, set xmlsec_binary=<path_to_script>/fix-xmlsec1.py.
4. Run fix-xmlsec1.py.

This script repairs the known issue whereby xmlsec1 is not compiled with RetrievalMethod and cannot find the location of the encrypted key. SAML2 responses would sometimes place EncryptedKey outside of the EncryptedData tree. This script moves EncryptedKey under EncryptedData.

Authenticating Hue users with Knox SSO

You can use the Apache Knox Gateway to interact with Hue REST APIs and the Hue user interface, along with other Cloudera components and services. To set up Knox Single Sign-on (SSO) to authenticate users, you must configure the KnoxSpnegoDjangoBackend property using Cloudera Manager.

Before you begin

To authenticate users using Knox SSO, you must have Knox installed on your Cloudera cluster, also known as a secure cluster.

Procedure

1. Sign in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configurations and search for the Authentication Backend field.
3. Select desktop.auth.backend.KnoxSpnegoDjangoBackend from the dropdown.

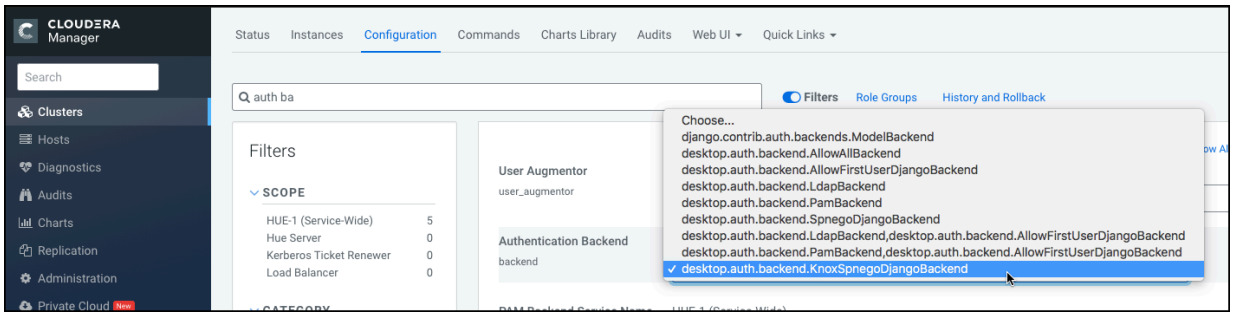

4. Go to Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini) and comment or remove any SAML-specific configurations, if present.

   Caution: Knox-SSO and SAML are incompatible and mutually exclusive. Hue authetication may fail with a redirection loop if you have the Knox-SSO and SAML configurations present in the Hue Advanced Configuration Snippet at the same time, as it confuses the authentication redirect to the IdP and back to Hue.

5. Click Save Changes.
6. Go to Clusters $Knox service Instances and note down the hostnames of the Knox Gateways.

   You must provide these details in the next step.

   If you have set up Knox in High-Availablity (HA) mode, then you can see more than one Knox Gateways listed on the Instances tab.
7. Go back to Clusters Hue service Configurations and search for the Knox Proxy Hosts field.
8. Enter the hostname of the Knox Gateway that you noted earlier.

   If you have set up Knox HA, then click + to add another hostname.

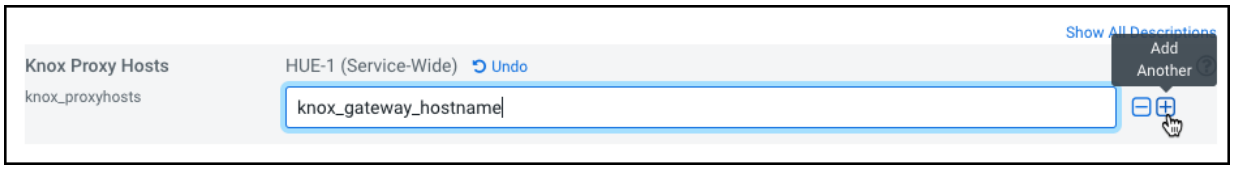

9. If you have deployed a Hue Load Balancer, then you must specify the Load Balancer hostname in the Knox Proxy Hosts field by clicking +.

10. Click Save Changes.

    You would see the following warning:

```
Role is missing Kerberos keytab. Go to the Kerberos Credentials page and
 click the Generate Missing Credentials button.
```

11. Click Administration on the Cloudera Manager left navigation panel and select Security.
12. Go to the Kerberos Credentials tab and click Generate Missing Credentials.

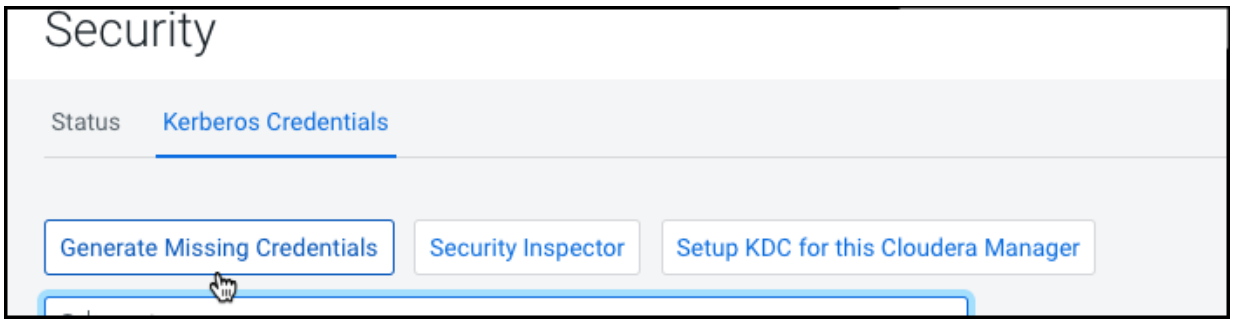

A pop-up showing the status is displayed.

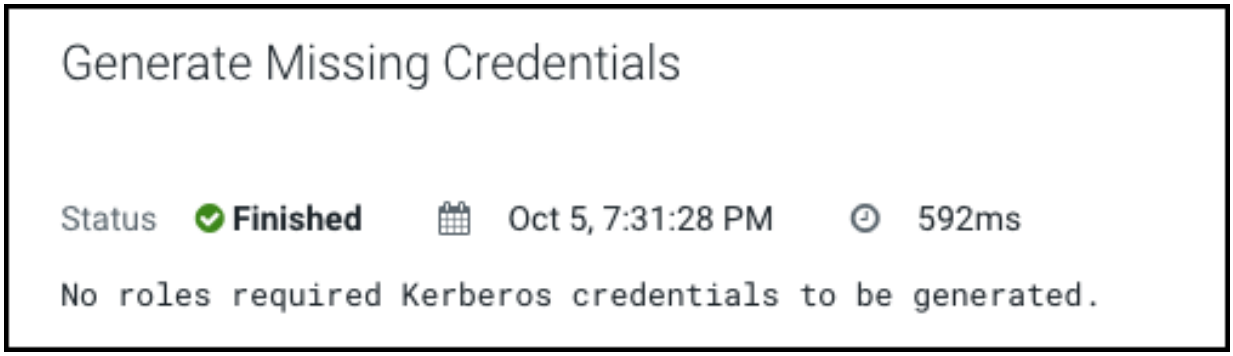

13. Go to Clusters Hue service and click Restart next to Actions.

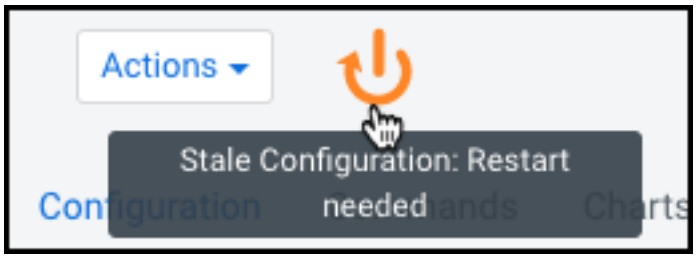

14. On the **Stale Configurations** page, click Restart Stale Services.

    The **Restart Stale Services** wizard is displayed.
15. On the Review Changes page, select Redeploy client configuration, and click Restart Now.

    The **Command Details** page shows the live status as the service restarts.

    When all the steps are complete, click Finish.

16. From the Hue service page, click Web UI Knox Gateway UI .

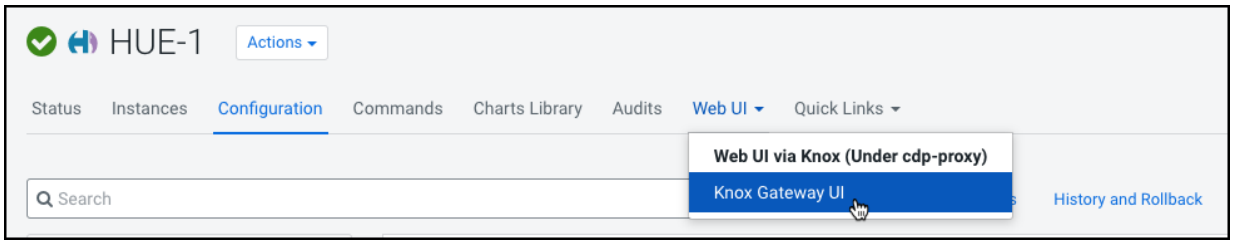

The Knox Gateway UI is displayed.

17. On the **General Proxy Information** page, expand the CDP Proxy topology by clicking + cdp-proxy under Topologies.

    The list of services that are configured with the cdp-proxy topology is displayed.

18. Click on the Hue logo.

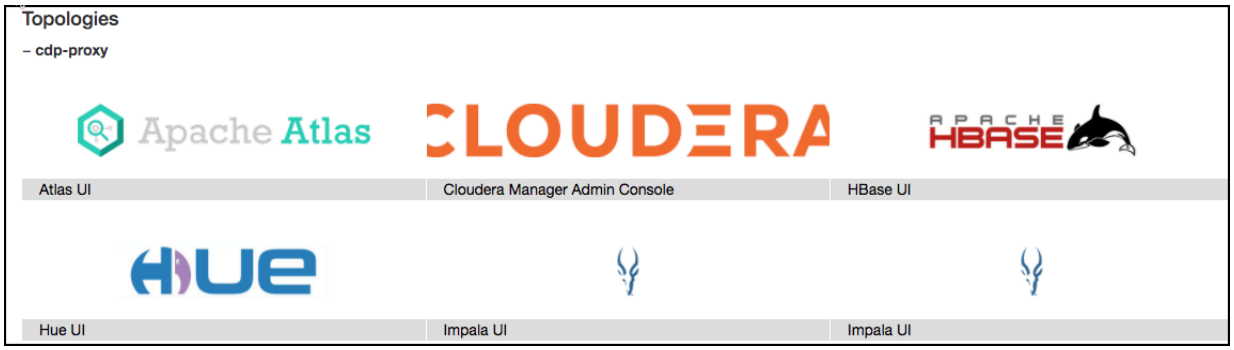

You should be able to log in to the Hue web UI.

You can also log into Hue using the following URL:

```
https://[***HOSTNAME***]:[***PORT***]/gateway/cdp-proxy/hue/
```

19. Go to Clusters Knox Configuration and add the following entries in the Knox Simplified Topology Management - cdp-proxy field:

```
HUE:httpclient.socketTimeout=[***TIMEOUT-IN-MINUTES***]
```

```
HUE:httpclient.connectionTimeout=[***TIMEOUT-IN-MINUTES***]
```

Replace *[***TIMEOUT-IN-MINUTES***]* with the actual timeout value depending on the load on your load on the cluster or environment. For example, to set a timeout of 20 minutes, specify 20m. The default timeout value is 5m.

Note: This step is required to prevent query timeouts, and if you are seeing errors such as “Results have expired, rerun the query if needed” in the Hue logs.

20. Restart the Knox service.

Authenticating Hue users with PAM

You can use Pluggable Authentication Modules (PAM) for authentication in Hue.

Procedure

1. Log in to Cloudera Manager as an Administrator.

2. Go to Clusters Hue Configuration and select desktop.auth.backend.PamBackend from the Authentication Backend drop-down menu.

   The default value of the PAM Backend Service Name property is “login”.

   Note: On RHEL 9, you must change the value of PAM Backend Service Name to sshd.

3. Click Save Changes.
4. Restart the Hue service.

Applications and permissions reference

Hue is a web-based UI for several cluster services that you can access by using Hue applications and their associated permissions.

Hue applications

These Cloudera services are available in Hue. Currently, Spark is only available in the upstream version of Hue.

Hue Application	Application Dependencies
HBase	HBase Browser
HDFS	Core, File Browser
Hive	Metastore Tables, Hive Editor
Impala	Metastore Tables, Impala Editor
MapRed / YARN	Job Browser, Job Designer, Oozie, Hive Editor, Pig, Sqoop
Oozie	Job Designer, Oozie Editor/Dashboard
Solr (Search)	Hadoop Security
Spark	Spark

Hue application permissions

Hue application permissions are composed of name.permission:action.

For example, filebrowser.access:Launch this application(3)

In this example:

- filebrowser = Hue application name
- access = Execute permissions
- Launch this application = Action that is enabled
- (3) = Process ID of the filebrowser application in the Hue database

Hue Application	Permission	Read/Write/Execute	Action Description
about	access	execute	Launch this application
beeswax	access	execute	Launch this application
dashboard	access	execute	Launch this application
filebrowser	access	execute	Launch this application
filebrowser	s3_access	execute	Access to S3 from filebrowser and filepicker
filebrowser	adls_access	execute	Access to ADLS from filebrowser and filepicker

Hue Application	Permission	Read/Write/Execute	Action Description
filebrowser	abfs_access	execute	Access to ABFS from filebrowser and filepicker
filebrowser	gs_access	execute	Access to GS from filebrowser and filepicker
help	access	execute	Launch this application
hive	access	execute	Launch this application
impala	access	execute	Launch this application
indexer	access	execute	Launch this application
jobbrowser	access	execute	Launch this application
jobsub	access	execute	Launch this application
kafka	access	execute	Launch this application
metadata	access	execute	Launch this application
metadata	write	write	Allow edition of metadata like tags
metastore	access	execute	Launch this application
metastore	write	write	Allow DDL operations. Need the app access too
notebook	access	execute	Launch this application
oozie	access	execute	Launch this application
oozie	dashboard_jobs_access	execute	Oozie Dashboard read-only user for all jobs
oozie	disable_editor_access	execute	Disable Oozie Editor access
proxy	access	execute	Launch this application
rdbms	access	execute	Launch this application
search	access	execute	Launch this application
useradmin	access_view:useradmin:edit_user	read/write/execute	Access to profile page on User Admin
useradmin	access_view:useradmin:view_user	read/write/execute	Access to any profile page on User Admin
useradmin	access	execute	Launch this application

Securing Hue passwords with scripts

You can secure passwords in Hue by using one consolidated script, or multiple individual scripts. Hue runs each password script at startup and extracts passwords from stdout.

About this task

Store scripts in a directory that only Hue can read, write, and execute. You can choose password script names but you cannot change hue.ini property names to which you assign those scripts. Add the suffix _script to any password property and set it equal to the script name.

Procedure

1. At the command line, create one or more password scripts. For example, create a consolidated script named my_passwords_script.sh:

```
#!/bin/bash
SERVICE=$1
if [[ ${SERVICE} == "ldap_password" ]]
then
 echo "<YOUR_LDAP_PASSWORD>"
fi

if [[ ${SERVICE} == "ssl_password" ]]
then
 echo "<YOUR_SSL_PASSWORD>"
fi

if [[ ${SERVICE} == "bind_password" ]]
then
 echo "<YOUR_BIND_PASSWORD>"
fi

if [[ ${SERVICE} == "db_password" ]]
then
 echo "<YOUR_DATABASE_PASSWORD>"
fi
```

2. Log on to Cloudera Manager and go to Hue Configuration .
3. Search on Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini.
4. Add script properties. In the following example, the required _script is added to the password property:

```
[desktop]
ldap_username=hueservice
ldap_password_script="/var/lib/hue/password_script.sh ldap_password"
ssl_password_script="/var/lib/hue/password_script.sh ssl_password"

[[ldap]]
bind_password_script="/var/lib/hue/password_script.sh bind_password"
[[database]]
db_password_script="/var/lib/hue/password_script.sh db_password"
```

5. Click Save Changes and Restart Hue.

Directory permissions when using PAM authentication backend

If you are using Pluggable Authentication Modules (PAM) for authenticating Hue users, then ensure that the Hue users have access to the /etc/shadow directory. Use an approach suitable to your organization's security policies.

Making Hue application user a member of the shadow group

This approach involves creating a “shadow” group and adding Hue to this group. Then you must grant the “shadow” group, read permission to the /etc/shadow directory.

Using Access Contol Lists (Recommended)

You can use Linux's ACLs to permit Hue user a read permission to the /etc/shadow directory by using the `setfacl` command. Cloudera recommends this approach.

Configuring TLS/SSL for Hue

You can independently enable TLS/SSL for Hue.

Cloudera recommends that your cluster and the Hue service use Kerberos for authentication. If you enable TLS/SSL for a cluster that has not been configured to use Kerberos, a warning is displayed. You should integrate the cluster with your Kerberos deployment before proceeding.

Creating a truststore file in PEM format

You must create the Hue Truststore by consolidating certificates of all SSL-enabled servers (or a single CA Certificate chain) that Hue communicates with into one file. This generally includes certificates of all the Oozie, HDFS, MapReduce, and YARN daemons, and any other SSL-enabled services.

About this task

Server certificates are stored in Java KeyStore (JKS) format. The Hue Truststore must be in the Privacy Enhanced Mail (PEM) format whereas other services use the JKS format by default. To create the Hue truststore, extract each certificate from Hadoop's Java Keystore with the Java keytool, convert the certificate to PEM format with the OpenSSL.org openssl tool, and then add it to the Hue truststore:

Procedure

1. Extract the certificate from the keystore of each TLS/SSL-enabled server with which Hue communicates. For example, if you have hadoop-server.keystore that contains a server certificate, foo-1.example.com with a password of example123, you would use the following keytool command:

   ```
   keytool -exportcert -keystore hadoop-server.keystore -alias foo-1.exampl
   e.com -storepass example123 -file foo-1.cert
   ```

2. Convert each certificate into a PEM file. Here is what the openssl tool command looks like for the foo-1.cert file that was extracted in Step 1:

   ```
   openssl x509 -inform der -in foo-1.cert > foo-1.pem
   ```

3. Concatenate all the PEM certificates you extracted and converted from the server truststore into one PEM file:

   ```
   cat foo-1.pem foo-2.pem foo-N.pem ... > hue_truststore.pem
   ```

   Concatenate the certificate files in the following order: SSL certificate followed by intermediate certificate, followed by the root CA certificate.

   Important: Ensure the final PEM truststore is deployed in a location that is accessible by the Hue service.

4. Log in to Cloudera Manager as an Administrator.
5. Go to Clusters Hue Configuration and add the following line in the Hue Service Environment Advanced Configuration Snippet (Safety Valve) field:

   ```
   REQUESTS_CA_BUNDLE=[***PATH-TO-HUETRUST.PEM-FIL E***]
   ```

6. Click Save Changes.
7. Restart the Hue service.

Configuring Hue as a TLS/SSL client

Hue acts as a TLS/SSL client when communicating with other services, such as core Hadoop, HBase, Oozie, and cloud providers like Amazon S3 or Azure.

To act as a TLS/SSL client, Hue must authenticate HDFS, MapReduce, YARN daemons, the HBase Thrift server, and so on. To do this, Hue needs to have the certificate chains of these components' hosts in the Hue trust store.

The Hue truststore is a single PEM (Privacy Enhanced Mail) file that contains the certificate authority (CA) root certificate and all intermediate certificates to authenticate the certificate installed on each TLS/SSL-enabled server. These servers host the services with which Hue communicates.

Note: A certificate is specific to a host. It is signed by a CA and tells the requesting client, which is Hue in this case, that the host is the same one as is represented by the host public key. Hue uses a chain of signing authority in its truststore to validate the CA that signed the host certificate.

Enabling Hue as a TLS/SSL client

After you create a Hue truststore file in PEM format, you can configure Hue as a TLS/SSL client by using Cloudera Manager.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration Hue TLS/SSL Server CA Certificate (PEM Format) ssl_cacerts and add the path to the *HUE_TRUSTSTORE*.pem file on the host that is running the Hue web server.
3. Click Save Changes.
4. Restart the Hue service.

Configuring Hue as a TLS/SSL server

Hue and other Python-based services expect certificates and keys to be stored in PEM (Privacy Enhanced Mail) format.

Before you enable TLS/SSL for the Hue server, you must generate a private key and certificate by using the openssl command-line tool and reuse a host's existing Java keystore by converting it to the PEM format.

Enabling Hue as a TLS/SSL server using Cloudera Manager

You can use Cloudera Manager to enable TLS/SSL for the Hue server.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration and filter by SCOPE Hue Server and CATEGORY Security .

3. Edit the following Hue TLS/SSL properties according to your cluster configuration:
   - Enable TLS/SSL for Hue: Select the check box to encrypt communication between clients and Hue with TLS/SSL.
   - Hue TLS/SSL Server Certificate File (PEM Format) ssl_certificate: Specifies the path to the TLS/SSL certificate on the host that is running the Hue web server.

     Ensure that you include the complete chain in the ssl_certificate PEM file.

     The order of the certificates should be as follows from the top to bottom: server, intermediate, root.

     If there are multiple intermediate CA certificates, then you must add them in the correct order. For example:

     ```
     Subject: CN=Hue Server Certificate
     Issuer: CN=Intermediate 2

       Subject: CN=Intermediate 2
       Issuer: CN=Intermediate 1

         Subject: CN=Intermediate 1
         Issuer: CN=RootCA

           Subject: CN=RootCA
           Issuer: CN=RootCA
     ```
   - Hue TLS/SSL Server Private Key File (PEM Format) ssl_private_key: Specifies the path to the TLS/SSL private key on the host running the Hue web server.
   - Hue TLS/SSL Private Key Password ssl_password: Specifies the password for the private key in the Hue TLS/SSL Server Certificate and Private Key file.
   - Hue TLS/SSL Server CA Certificate (PEM Format) ssl_cacerts: Specifies the path to the TLS/SSL certificate authority root certificate on the host that is running the Hue web server.
4. Add the path to the certificate chain PEM file in [desktop] section of the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

   ```
   [desktop]
   ssl_certificate_chain=/[***PATH***]/[***TO***]/[***FULL-CHAIN***].pem
   ```
5. Click Save Changes.
6. Select ActionsRestart to restart the Hue service.

What to do next

Change the permissions for Hue to read the certificates after you have enabled TLS/SSL as follows:

```
chmod 644 [***PATH-WHERE-SSL-FILES-FOR-HUE-ARE-LOCATED***]
```

Enabling TLS/SSL for Hue Load Balancer

To configure the Hue Load Balancer to use HTTPS or operate as a TLS/SSL server, you need a self-signed SSL certificate and a private key file. If the private key file is password protected, then you must configure the Hue Load Balancer to use the corresponding key password.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration Scope Load Balancer .

3. Enter the location of the file that contains the server certificate key for TLS/SSL on the host running Hue Load Balancer in the Hue Load Balancer TLS/SSL Server Certificate File (PEM Format) field.

   The certificate file must be in the Privacy-Enhanced Mail (PEM) format.
4. Enter the location of the TLS/SSL file that contains the private key used for TLS/SSL on the host running Hue Load Balancer, in the Hue Load Balancer TLS/SSL Server Private Key File (PEM Format) field.

   The certificate file must be in PEM format.
5. (Optional) If the private key file is password protected perform the following steps:
   a) Create a password file in your chosen security directory and insert the private key password as shown in the following example:

      ```
      echo "abc123" > /etc/security/password.txt
      ```

      Where abc123 is the private key password and password.txt is the password file.
   b) Set the file ownership and permissions as shown in the following example:

      ```
      chown hue:hue password.txt
      chmod 700 password.txt
      ```
   c) Enter the path to the file containing the passphrase used to encrypt the private key of the Hue Load Balancer server in the Hue Load Balancer TLS/SSL Server SSLPassPhraseDialog field.
6. Click Save Changes.
7. Restart the Hue service.

Enabling TLS/SSL communication with HiveServer2

For Hue to communicate with HiveServer2 using TLS/SSL, Hue needs the Hive certificate and certificate chain.

To enable TLS/SSL communication with HiveServer2, add the following properties in the [beeswax] section under [[ssl]] in the Cloudera Manager Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini configuration property:

Property	Description

`[beeswax]`
`[[ssl]]`
`enabled`
	Valid values: true | false
	Enables or disables TLS/SSL communication for this server.
	Default setting: false
	Example: enabled=true

`[beeswax]`
`[[ssl]]`
`cacerts`
	Valid values: directory path
	Specifies the path to the Certificate Authority certificates.
	Default setting: /etc/hue/cacerts.pem
	Example: cacerts=/opt/cloudera/security/CAcerts/cacerts

`[beeswax]`
`[[ssl]]`
`validate`
	Valid values: true | false
	Specifies whether Hue validates certificates received from the server.
	Default setting: true
	Example: validate=true

Enabling TLS/SSL communication with Impala

For Hue to communicate with Impala using TLS/SSL, Hue needs the Impala certificate and certificate chain.

To enable TLS/SSL communication with Impala, add the following properties in the [impala] section under [[ssl]] in the Cloudera Manager Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini configuration property:

Property	Description

`[impala]`
`[[ssl]]`
`enabled`
	Valid values: true | false
	Enables or disables TLS/SSL communication for this server.
	Default setting: false
	Example: enabled=true

`[impala]`
`[[ssl]]`
`cacerts`
	Valid values: directory path
	Specifies the path to the Certificate Authority certificates.
	Default setting: /etc/hue/cacerts.pem
	Example: cacerts=/opt/cloudera/security/CAcerts/cacerts

`[impala]`
`[[ssl]]`
`validate`
	Valid values: true | false
	Specifies whether Hue validates certificates received from the server.
	Default setting: true
	Example: validate=true

Securing database connections with TLS/SSL

Hue uses different clients to communicate with each database internally. Client-specific options, such as secure connectivity can be configured using Cloudera Manager.

Procedure

1. Log in to Cloudera Manager as an administrator.
2. Go to Clusters Hue service Configuration and add the following section in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[desktop]
  [[database]]
    …
    options={"ssl":{"ca":"/tmp/ca-cert.pem"}}
```

   This identifies the Certificate Authority (CA) certificate for the backend database. You can also identify public and private keys as follows:

```
options='{"ssl": {"ca": "/tmp/newcerts2/ca.pem", "key": "/tmp/newcerts2/
client-key.pem", "cert": “/tmp/newcerts2/client-cert.pem"}}'
```

3. Click Save Changes.
4. Restart the Hue service.

Disabling CA Certificate validation from Hue

By default, Hue validates CA Certificates for Oozie, HTTPFS, Resource Manager, and Job History Server when SSL is enabled for any of these services. If you have not enabled TLS/SSL on your cluster, then you can disable Hue from validating the CA Certificates for other services on your CDP cluster.

Procedure

1. Log in to Cloudera Manager as an Administrator.

2. Go to Clusters Hue Configuration and add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[hadoop]
[[hdfs_clusters]]
[[[default]]]
ssl_cert_ca_verify = False #HTTPFS service
[[yarn_clusters]]
[[[default]]]
ssl_cert_ca_verify = False #Resource Manager/Job History Server
[[[ha]]]
ssl_cert_ca_verify = False #Resource Manager HA
[liboozie]
ssl_cert_ca_verify = False
```

3. Click Save Changes.
4. Restart the hue service.

Enforcing TLS version 1.2 for Hue

Cloudera Data Hub cluster components and services such as the Cloudera Manager web UI, the Hue web UI, and the Impala web UI communicate with each other using TLS 1.2 as the default TLS protocol, and TLS 1.1 or 1.0 if a client requests it. You can enforce these services to only use TLS 1.2 by specifying the SSL protocol in Cloudera Manager.

About this task

Note: The TLS version that is auto-applied depends on the Python version. If your installed Python version is higher than 2.7.9, then both the client and the server use the latest TLS. But if your installed Python version is lower than 2.7.9, then TLS 1.0 is used.

Note: The following steps do not apply to the connections between Hue and its backend database or external identity services such as LDAP and Active Directory.

Procedure

1. Sign in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration Load Balancers Advanced and add the following line in the SSL Protocol field:

```
-all +TLSv1.2
```

3. Click Save Changes.
4. Restart the Hue service.
5. Verify that TLS version 1.2 is used for encryption and all the ciphers used are “strong” by using a security scanner such as Nmap.
   a) Open a CLI console on a machine in your cluster.
   b) Run the following command:

```
nmap -sV --script +ssl-enum-ciphers -p 8889 [***HOSTNAME***] -f
```

   Replace [***HOSTNAME***] with the actual name of the host.

   The following is a sample output. It shows that only TLS 1.2 is available for the handshake and that all the ciphers are “strong”:

```
Starting Nmap 7.80 ( http://nmap.org ) at 2020-30-10 11:16 PDT
```

```
Nmap scan report for hostname.example.com (a.b.c.d)
Host is up (-1800s latency).
PORT STATE SERVICE VERSION
8889/tcp open ssl/http Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.2k-fips)
| ssl-enum-ciphers:
| SSLv3: No supported ciphers found
| TLSv1.2:
| ciphers:
| TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
| TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
| TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 - strong
| TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 - strong
| TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
| TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 - strong
| TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 - strong
| TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
| TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
| TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 - strong
| TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 - strong
| TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
| TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 - strong
| TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 - strong
| TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
| TLS_RSA_WITH_AES_128_CBC_SHA - strong
| TLS_RSA_WITH_AES_128_CBC_SHA256 - strong
| TLS_RSA_WITH_AES_128_GCM_SHA256 - strong
| TLS_RSA_WITH_AES_256_CBC_SHA - strong
| TLS_RSA_WITH_AES_256_CBC_SHA256 - strong
| TLS_RSA_WITH_AES_256_GCM_SHA384 - strong
| compressors:
| NULL
|_ least strength: strong

Service detection performed. Please report any incorrect results at http
://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.43 seconds
You have new mail in /var/spool/mail/root
```

Note: You must perform steps 2 through 5 every time you upgrade Cloudera Manager.

6. Set the SSL_CIPHER_LIST property for the Hue Server in Cloudera Manager.
   a) Sign in to Cloudera Manager as an Administrator.
   b) Go to Clusters Hue service Configuration Hue Server and specify the following in the Hue Server Advanced Configuration Snippet (Safety Valve) for hue_safety_valve_server.ini field:

   ```
   [desktop]
   ssl_cipher_list=DEFAULT:!aNULL:!eNULL:!LOW:!EXPORT:!SSLv2:!SSLv3:!TLSv1
   ```

   The SSL_CIPHER_LIST property is a list of one or more cipher suite strings separated by colons. This restricts the use of the default cipher suite before establishing an encrypted SSL connection.
   c) Click Save Changes.
   d) Restart the Hue service.

Securing sessions

When a Hue session expires, the screen blurs and the user is automatically logged out of the Hue web interface. Logging back on returns the user to the same location in the application.

Session timeout

User sessions are controlled with the ttl (time-to-live) property, which is set in the Cloudera Manager Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini property as follows:

```
[desktop]
     [[session]]
     ttl=[***NUMBER-OF-SECONDS***]
```

The default setting for ttl is 1,209,600 seconds, which equals two weeks. The ttl property determines the length of time that the cookie with the user's session ID lives before expiring. After the ttl setting is reached, the user's session expires whether it is active or not.

Idle session timeout

Idle sessions are controlled with the idle_session_timeout property, which is set in the Cloudera Manager Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini property as follows:

```
[desktop]
     [[auth]]
     idle_session_timeout=[***NUMBER-OF-SECONDS***]
```

Sessions expire that are idle for the number of seconds set for this property. For example, if you set idle_session_timeout=900, sessions expire after being idle for 15 minutes. You can disable the property by setting it to a negative value, like idle-session_timeout=-1.

Secure session login

Session login properties are set under the [desktop] [[auth]] section in the Cloudera Manager Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini property as follows:

```
[desktop]
     [[auth]]
     [***SET-SESSION-LOGIN-PARAMETERS-HERE***]
```

Note: These configuration settings are based on django-axes 1.5.0.

Use the following properties to configure session login behavior:

change_default_password	Valid values: true | false
	If this property is set to true, users must change their passwords on first login attempt.
	Example:
	`[desktop]`
	`[[auth]]`
	`change_default_password=true`
	To use this property, you must enable the AllowFirstUserDjangoBackend in Hue. For example:
	`[desktop]`
	`[[auth]]`
	`backend=desktop.auth.backend.AllowFirstUserDjangoBackend`

expires_after	Use this property to configure the number of seconds after logout that user accounts are disabled. For example, user accounts are disabled 900 seconds or 15 minutes after logout with the following configuration:
	`[desktop]`
	`[[auth]]`
	`expires_after=900`
	If you set this property to a negative value, user sessions never expire. For example, expires_after=-1.

expire_superusers	Use to expire superuser accounts after the specified number of seconds after logout. For example, expire_superusers=900 causes superuser accounts to expire 15 minutes after logging out.
login_cooloff_time	Sets the number of seconds after which failed logins are forgotten. For example, if you set login_cooloff_time=900, a failed login attempt is forgotten after 15 minutes.
login_failure_limit	Sets the number of login attempts allowed before a failed login record is created. For example, if you set login_failure_limit=3, a failed login record is created after 3 login attempts.
login_lock_out_at_failure	Valid values: true | false If set to true: • The IP address that is attempting to log in is locked out after exceeding the limit set for login_failure_limit. • If login_lock_out_by_combination_user_and_ip is also set to true, both the IP address and the user are locked out after exceeding the limit set for login_failure_limit. • If login_lock_out_use_user_agent is also set to true, both the IP address and the agent application (such as a browser) are locked out after exceeding the limit set for login_failure_limit.
login_lock_out_by_combination_user_and_ip	Valid values: true | false If set to true, both the IP address and the user are locked out after exceeding the limit set for login_failure_limit.
login_lock_out_use_user_agent	Valid values: true | false If set to true, the agent application (such as a browser) is locked out after exceeding the limit set for login_failure_limit.

Secure session cookies

Session cookie properties are set under the [desktop] [[session]] section in the Cloudera Manager Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini property as follows:

```
[desktop]
     [[session]]
     [***SET-SESSION-COOKIE-PROPERTIES-HERE***]
```

Use the following properties to configure session cookie behavior:

secure	Valid values: true | false
	If this property is set to true, the user session ID is secured.
	Important: To use this property, HTTPS must be enabled.
	Example:
	`[desktop]`
	`[[session]]`
	`secure=true`
	By default this property is set to false.

http_only	Valid values: true | false
	If this property is set to true, the cookie with the user session ID uses the HTTP only flag.
	Example:
	`[desktop]`
	`[[session]]`
	`http_only=true`
	Important: If the HttpOnly flag is included in the HTTP response header, the cookie cannot be accessed through a client side script.
	By default this property is set to true.

expire_at_browser_close	Valid values: true | false
	If this property is set to true, only session-length cookies are used. Users are automatically logged out when the browser window is closed.
	Example:
	`[desktop]`
	`[[session]]`
	`expire_at_browser_close=true`
	By default this property is set to false.

Specifying HTTP request methods

You can specify the HTTP request methods that the Hue server responds to.

Use the http_allowed_methods property under the [desktop] section in the Cloudera Manager Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini property.

By default, the http_allowed_methods property is set to options, get, head, post, put, delete, connect.

Restricting supported ciphers for Hue

You can configure the list of ciphers that Hue supports with HTTPS.

Use the ssl_cipher_list property under the [desktop] section in the Cloudera Manager Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini property:

```
[desktop]
ssl_cipher_list=[***LIST-OF-ACCEPTED-CIPHERS***]
```

By default, the ssl_cipher_list property is set to !aNULL:!eNULL:!LOW:!EXPORT:!SSLv2. Specify ciphers using the cipher list format described at OpenSSL Cryptograpy and SSL/TLS Toolkit Manpages by selecting the SSL version, and then going to Commands ciphers .

Specifying domains or pages to which Hue can redirect users

You can restrict the domains or pages to which Hue can redirect users.

Use the redirect_whitelist property under the [desktop] section in the Cloudera Manager Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini property:

```
[desktop]
redirect_whitelist=[***REDIRECT-URL***]
```

Specify the redirect_whitelist value with a comma-separated list of regular expressions that match the redirect URL. For example, to restrict redirects to your local domain and fully-qualified domain name (FQDN), use the following value:

```
redirect_whitelist=^\/.*$,^http:\/\/www.mydomain.com\/.*$
```

Securing Hue from CWE-16

Hue may have allowed external domains such as doubleclick.net, .googletagmanager.com, or *.google-analytics.com to run JavaScript scripts, for certain URLs in the Content Security Policy (CSP) headers. This may lead to Common Weakness Enumeration (CWE-16). To secure Hue from CWE-16 class of weaknesses, you can add the X-Content-Type-Options response HTTP header and prevent attacks based on MIME-type confusions in Hue's Advanced Configuration Snippet using Cloudera Manager.

Procedure

1. Log in to Cloudera Manageras an Administrator.
2. Go to Clusters Hue Configuration and add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.in field:

   ```
   [desktop]
   # X-Content-Type-Options: nosniff This is an HTTP response header
   # feature that helps prevent attacks based on MIME-type confusion.

   secure_content_security_policy="script-src 'self' 'unsafe-inline' 'unsafe-
   eval' *.googletagmanager.com *.doubleclick.net data:;img-src 'self' *.doub
   leclick.net http://*.tile.osm.org *.tile.osm.org *.gstatic.com data:;sty
   le-src 'self' 'unsafe-inline' fonts.googleapis.com;connect-src 'self' *.
   google-analytics.com;frame-src *;child-src 'self' data: *.vimeo.com;obje
   ct-src 'none'"
   ```
3. Click Save Changes.
4. Restart the Hue service.

Setting Oozie permissions

You can control access to the Oozie dashboard and editor by using controls in the Hue Web UI.

1. On the Cloudera Manager home page, click the Hue service.
2. On the Hue service page, select Web UIHue Load Balanced - recommended .
3. Log in to the Hue Web UI.

4. In the Hue Web UI, click the admin menu icon in the lower part of the left menu and select Manage Users:

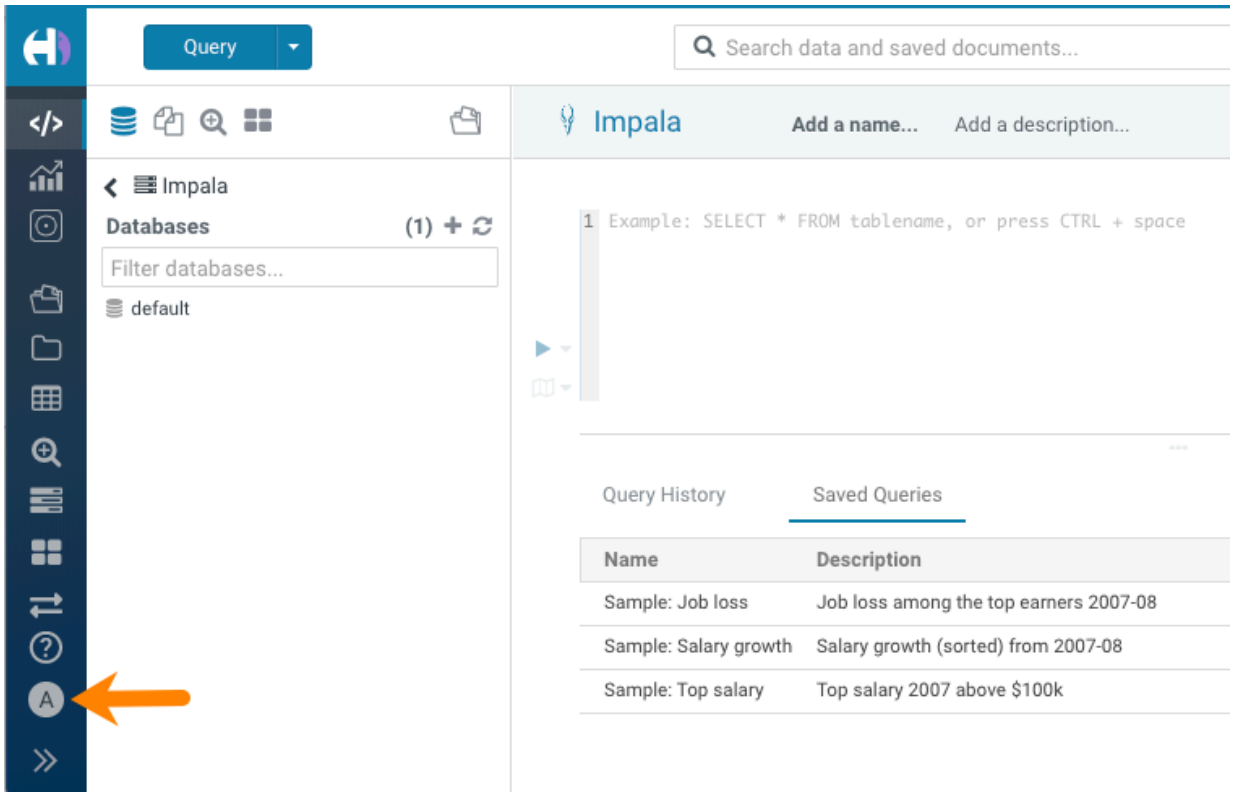

5. On the User Admin page, click the Groups tab.
6. On the Hue Groups page, in the Group Name column, click the default group.

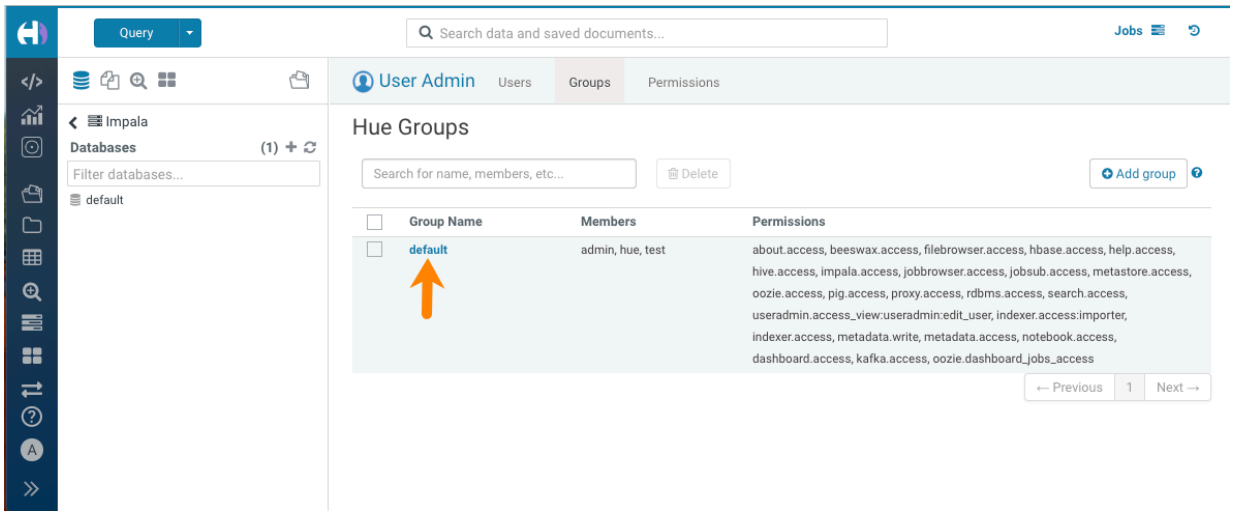

7. On the Hue Groups - Edit group page, scroll down to locate the list of permissions and then scroll further to locate the Oozie permissions:

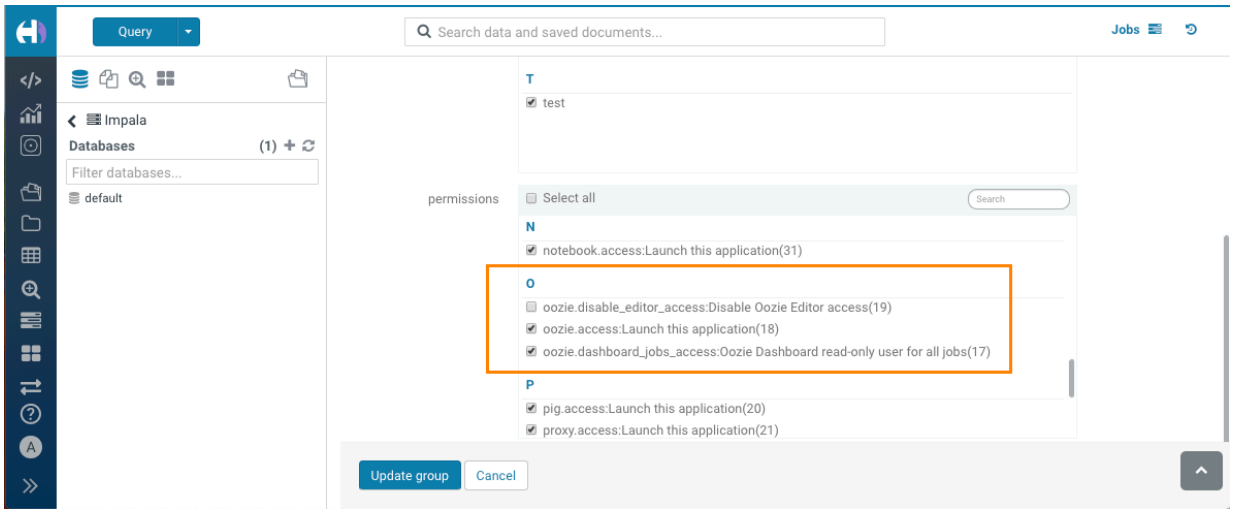

Groups property in UI	Description
oozie.disable_editor_access	Disables access to the Oozie editor for the selected groups Default setting: Unchecked, which disables this permission.
oozie.access	Enables access to the Oozie editor in Hue. Default setting: Checked, which enables access to the Oozie editor.
oozie.dashboard_jobs_access	Enables read-only access for all jobs in the Oozie dashboard. Default setting: Check, which enables this permission.

8. Check or uncheck the permissions as needed and then click Update group to save the permission change.

Configuring secure access between Solr and Hue

If you are using Solr (search) to build dynamic search dashboards and explore data from the Hue web UI, and have secured your Cloudera cluster using Kerberos, then you must enable secure access between the Solr service and Hue by turning on the security_enabled parameter in the hue_safety_valve.ini file.

About this task

The security_enabled parameter is set to false, by default. To enable secure access between Solr and Hue:

Procedure

1. Sign in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration and search for the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field.
3. Go to the [search] section and set the value of the security_enabled parameter to true, as shown in the following sample:

```
[search]
# Requires FQDN in solr_url if enabled
security_enabled=true
# URL of the Solr Server
```

```
solr_url=http://[***FQDN-SOLR-HOST***]:8983/solr
```

Note: If you set security_enabled=true, then you must provide the fully-qualified domain name of the Solr host in the Solr URL.

4. Click Save Changes.
5. Restart the Hue service.

Cloudera Runtime 7.3.1

Encryption reference

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Auto-TLS Requirements and Limitations.....4

Rotate Auto-TLS Certificate Authority and Host Certificates..... 5

Auto-TLS Agent File Locations..... 6

Auto-TLS Requirements and Limitations

Reference information for Auto-TLS requirements, limitations, and component support.

Requirements

- You must install the Cloudera Manager Agent software on the Cloudera Manager Server host.
- You can enable auto-TLS using certificates created and managed by a Cloudera Manager certificate authority (CA), or certificates signed by a trusted public CA or your own internal CA. If you want to use a trusted public CA or your own internal CA, you must obtain all of the host certificates before enabling auto-TLS.

Component support for Auto-TLS

The following Cloudera services support auto-TLS:

- Atlas
- Cloudera Manager Host Monitor Debug Interface
- Cloudera Manager Service Monitor Debug Interface
- Cruise Control
- HBase
- HDFS Client Configuration
- HDFS NameNode Web UI
- Hive-on-Tez
- HiveServer2
- HttpFS
- Hue Client
- Hue Load Balancer
- Hue Server
- Impala Catalog Server
- Impala Server
- Impala StateStore
- Java Keystore Key Management Server (KMS)
- Kafka Broker Server
- Kafka Mirrormaker
- Knox
- Kudu
- Livy
- Oozie
- Ozone
- Phoenix
- Ranger
- Safenet Luna Hardware Security Modules (HSM) KMS
- Schema Registry
- Solr
- Spark History Server
- Streams Messaging Manager
- Streams Replication Manager
- YARN Web UI
- Zeppelin
- ZooKeeper

For unlisted Cloudera services, you must enable TLS manually. See the applicable component guide for more information.

Limitations

- It is not possible to rename hostnames of cluster nodes in an Auto-TLS setup.

Rotate Auto-TLS Certificate Authority and Host Certificates

Your cluster security requirements may require that you rotate the auto-TLS CA and certificates.

Using an internal CA (Use case 1)

1. Navigate to Administration Security . Click Rotate Auto-TLS Certificates to launch the wizard.
2. Complete the wizard.

Using a custom CA (Use case 3)

1. Use the `/cm/commands/addCustomCerts` API command to replace the old certificates with new certificates in CMCA directory for each host. You must run this command for each host separately. An example of a curl command to upload the certificates to Cloudera Manager:

```
curl -u admin:admin -X POST --header 'Content-Type:
                        application/json' --header 'Accept: application/
json' -d '{
                        "location": "/opt/cloudera/AutoTLS",
                        "interpretAsFilenames": true,
                        "hostCerts": [ {
                        "hostname": "ccycloud-10.vcdp71.root.hwx.site",
                        "certificate":
                        "/tmp/auto-tls/certs/ccycloud-10.vcdp71.root.hwx.
site.pem",
                        "key":
                        "/tmp/auto-tls/certs/ccycloud-10.vcdp71.root.hwx.s
ite.pem"
                        } ]
                        }' 'https://ccycloud-7.vcdp71.root.hwx.site:7183/
api/v41/cm/commands/addCustomCerts'
```

   In the example above, the "location" should be omitted if Auto-TLS was enabled or rotated after 7.1, and the file paths should point to files on the Cloudera Manager server host.
2. Use CM API /hosts/{hostId}/commands/generateHostCerts to deploy the new certificates to each host. You must run this command for each host separately. An example curl command :

```
curl -u admin:admin -X POST --header 'Content-Type: application/json' --
header
                            'Accept: application/json' -d '{ "sshPort" :
 22, "userName" : "root", "password" : "cloudera" }'
                            'https://ccycloud-7.vcdp71.root.hwx.site:7183/
api/v41/hosts/250e1bb7-8987-419c-a53f-c852c275d299/commands/generateHost
Certs'
```

   where '250e1bb7-8987-419c-a53f-c852c275d299' in the command above is the hostID.

Auto-TLS Agent File Locations

The certificates, keystores, and password files generated by auto-TLS are stored in /var/lib/cloudera-scm-agent/agent-cert on each Cloudera Manager Agent.

Filenames

Table 1: Auto-TLS Agent Files

Filename	Description
cm-auto-global_cacerts.pem	CA certificate and other trusted certificates in PEM format
cm-auto-global_truststore.jks	CA certificate and other trusted certificates in JKS format
cm-auto-in_cluster_ca_cert.pem	CA certificate in PEM format
cm-auto-in_cluster_truststore.jks	CA certificate in JKS format
cm-auto-host_key_cert_chain.pem	Agent host certificate and private key in PEM format
cm-auto-host_cert_chain.pem	Agent host certificate in PEM format
cm-auto-host_key.pem	Agent host private key in PEM format
cm-auto-host_keystore.jks	Agent host private key in JKS format
cm-auto-host_key.pw	Agent host private key password file

Cloudera Runtime 7.3.1

Ranger Auditing

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Audit Overview 4

Managing Auditing with Ranger 4
- Viewing audit details 6
- Viewing audit metrics 9
- Creating a read-only Admin user (Auditor) 10
- Configuring Ranger audit properties for Solr 11
  - Limiting solr spool directory growth 12
- Configuring Ranger audit properties for HDFS 12
- Triggering HDFS audit files rollover 13
- Configuring Ranger audit log storage to a local file 15

Ranger Audit Filters 17
- Default Ranger audit filters 18
- Configuring a Ranger audit filter policy 20
- How to set audit filters in Ranger Admin Web UI 23
- Filter service access logs from Ranger UI 24

Configuring audit spool alert notifications 26

Charting spool alert metrics 30

Excluding audits for specific users, groups, and roles 30

Changing Ranger audit storage location and migrating data 31

Configuring Ranger audits to show actual client IP address 35

Audit Overview

Apache Ranger provides a centralized framework for collecting access audit history and reporting data, including filtering on various parameters. Ranger enhances audit information obtained from Hadoop components and provides insights through this centralized reporting capability.

Ranger plugins support storing audit data to multiple audit destinations.

Solr

The Solr audit destination is a short term audit destination (with a default TTL of 90 days) managed by Solr which can be configured by a Ranger Admin user. The Ranger Admin Web UI displays the access audit data from the audit data stored in Solr.

HDFS

The HDFS audit destination is a long term audit destination for archival/compliance purposes. The HDFS audit destination has no default retention/purge period. A customer must manage the storage/retention/purge/archival of audit data stored in HDFS manually.

Related Information

Configuring Ranger audit properties for Solr

Configuring Ranger audit properties for HDFS

Managing Auditing with Ranger

You can manage auditing using the Audit page in the Ranger Admin Web UI.

To explore options for auditing policies, click Audits

Figure 1: Ranger Admin Audits Menu

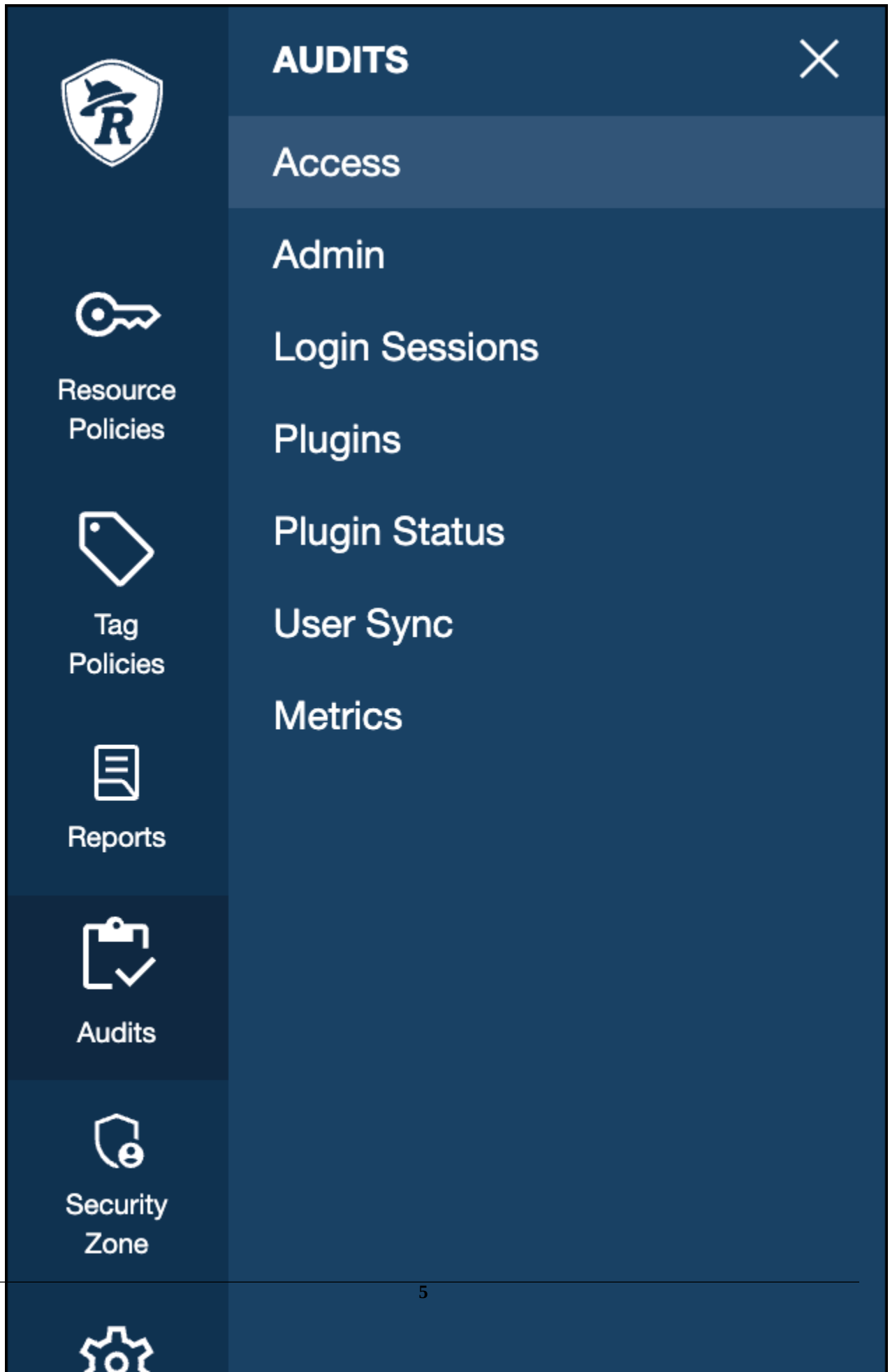
AUDITS
Access
Admin
Login Sessions
Plugins
Plugin Status
User Sync
Metrics
Resource Policies
Tag Policies
Reports
Audits
Security Zone

in the left menu of the Ranger Admin Web UI,

then, choose one of the following options:

- Access
- Admin
- Login sessions
- Plugins
- Plugin Status
- User Sync
- Metrics

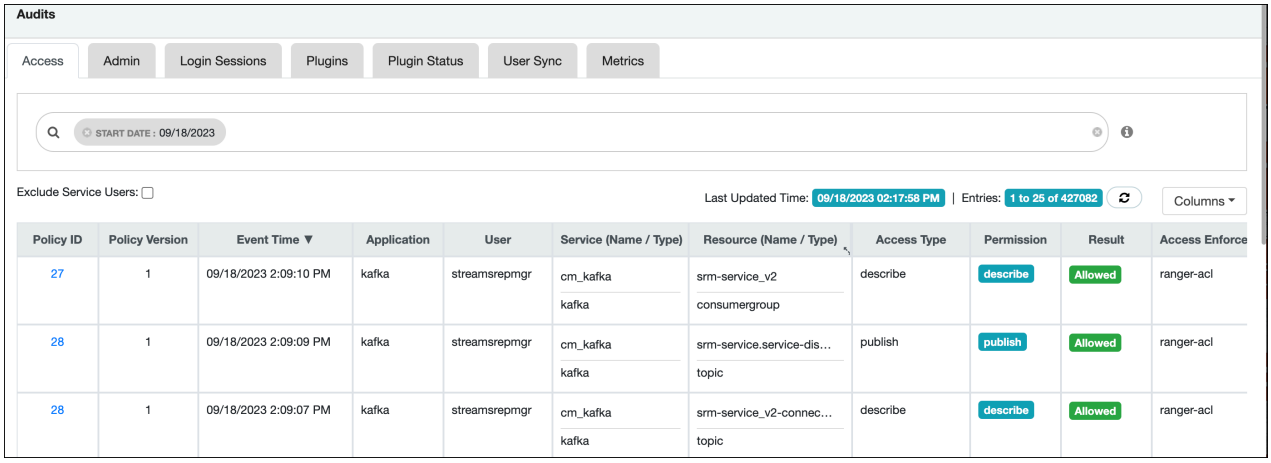

See related topics for more information about using the Audits pages to manage auditing.

Viewing audit details

How to view policy and audit log details in Ranger audits.

Procedure

To view policy details for a specific audit log, click Access Policy ID .

Audit > Access: hbasemaster

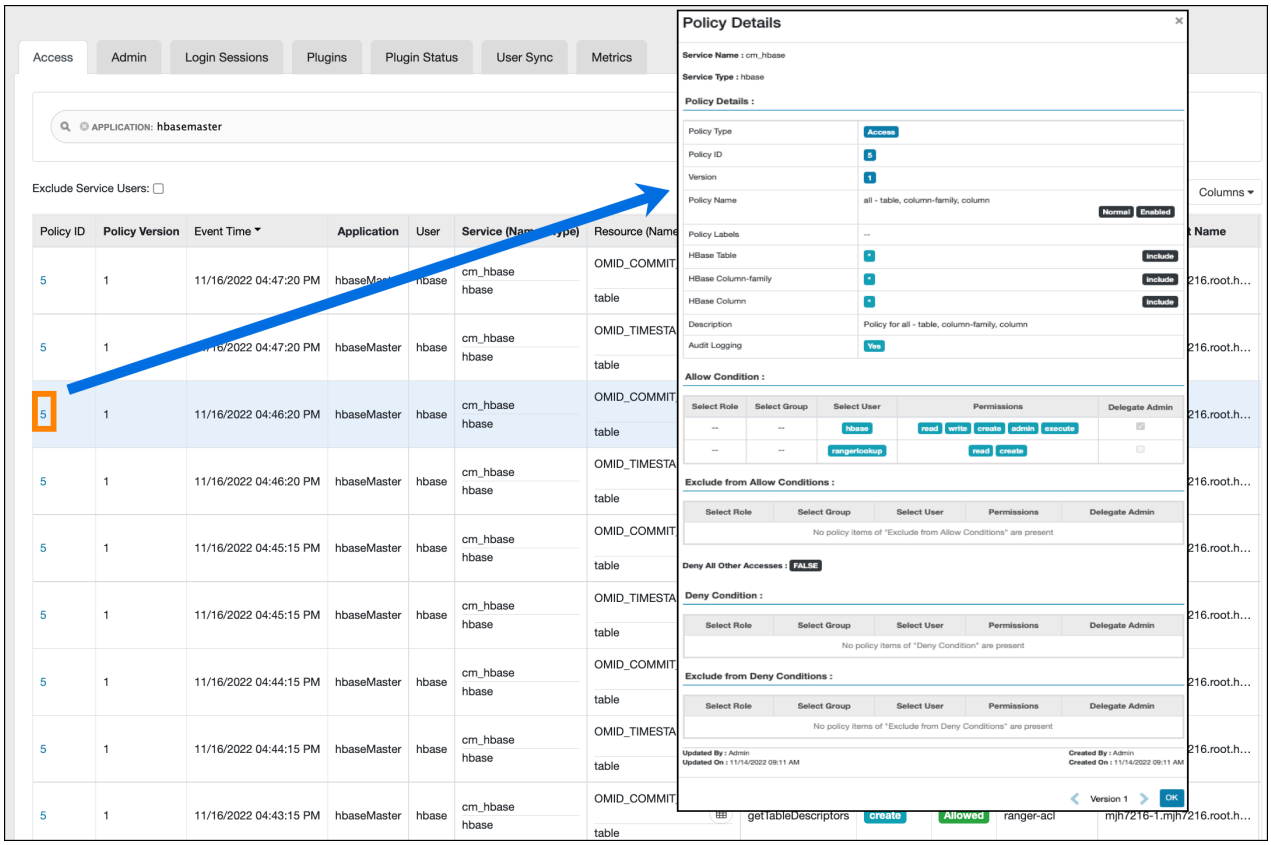

Audit > Access: HadoopSQL

Note: The Hive plugin audit handler now logs UPDATE operations as INSERT, UPDATE, DELETE, and TRUNCATE specifically.

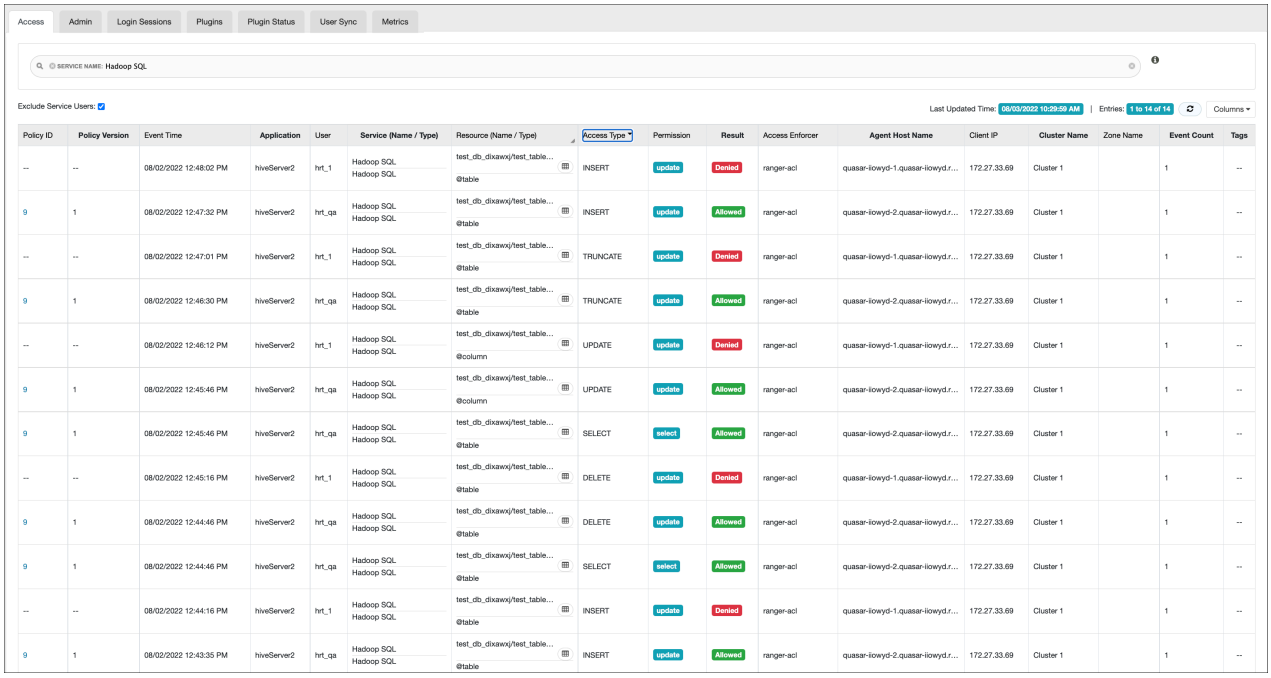

Audit > Admin: Create

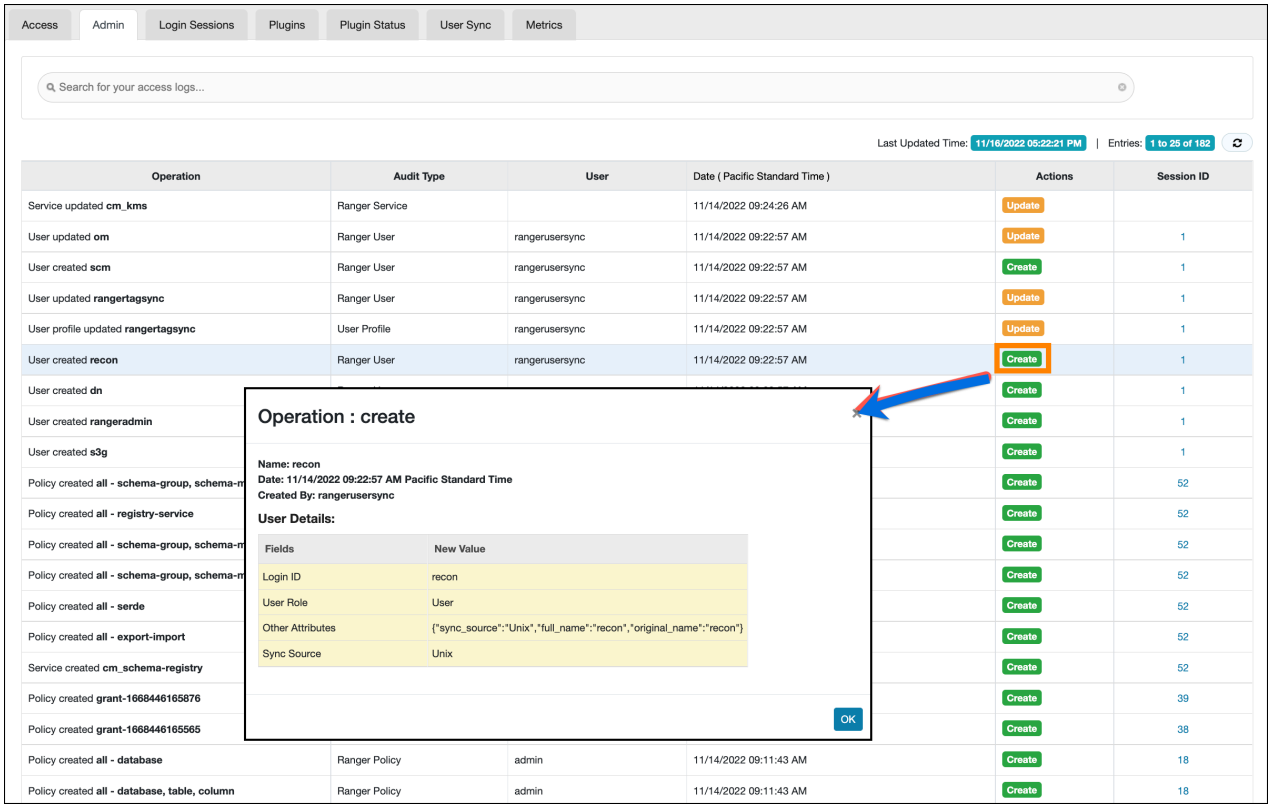

Audit > User Sync: Sync details

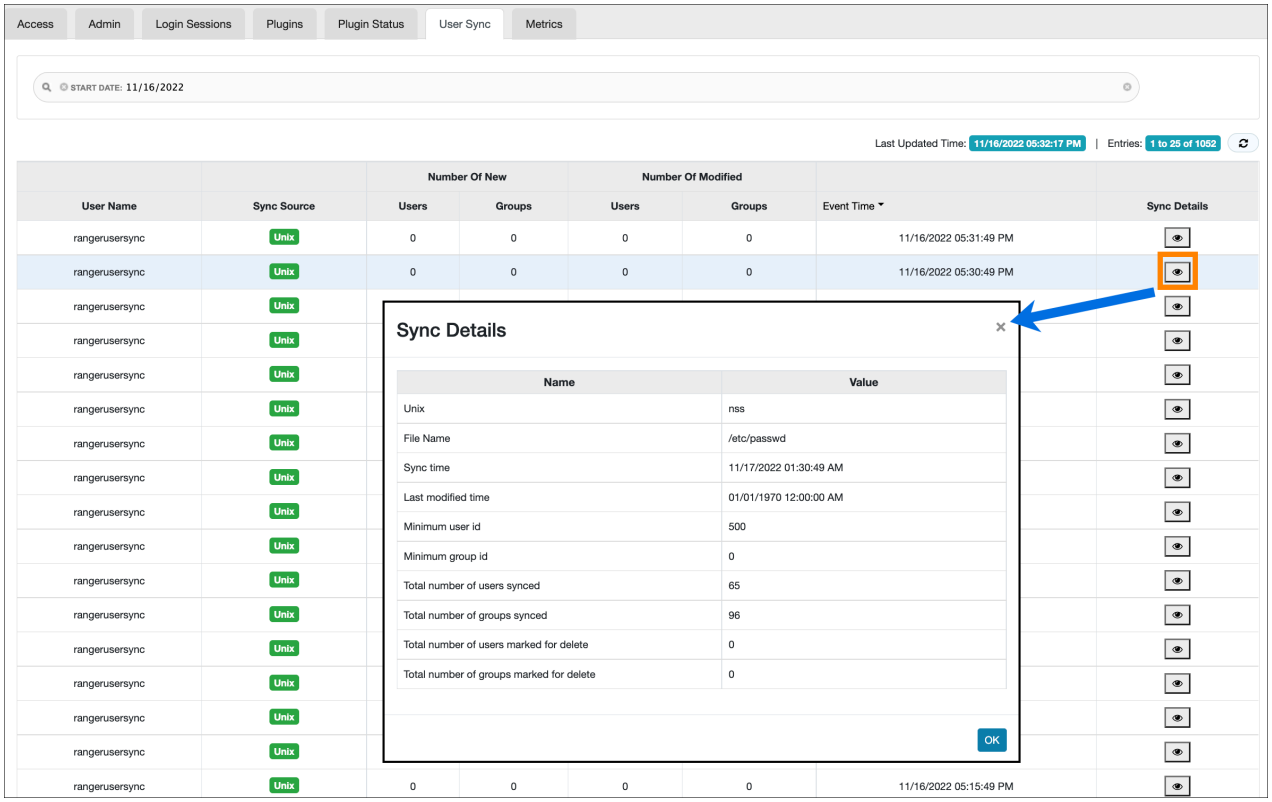

Viewing audit metrics

How to view audit metrics information using the Ranger Admin Web UI.

About this task

Metrics provides a high-level view of audit logs as they generate and update in Ranger. Ranger captures audit metrics throughput from the following Ranger services:

- Atlas
- HBase
- Hdfs
- Hive
- Impala
- Kafka
- Knox
- Kudu
- NiFi
- Schema-registry
- Solr
- Streams Messaging Manager
- Yarn

Procedure

1. To view audit metrics, in the Ranger Admin Web UI, click Audits Metrics .

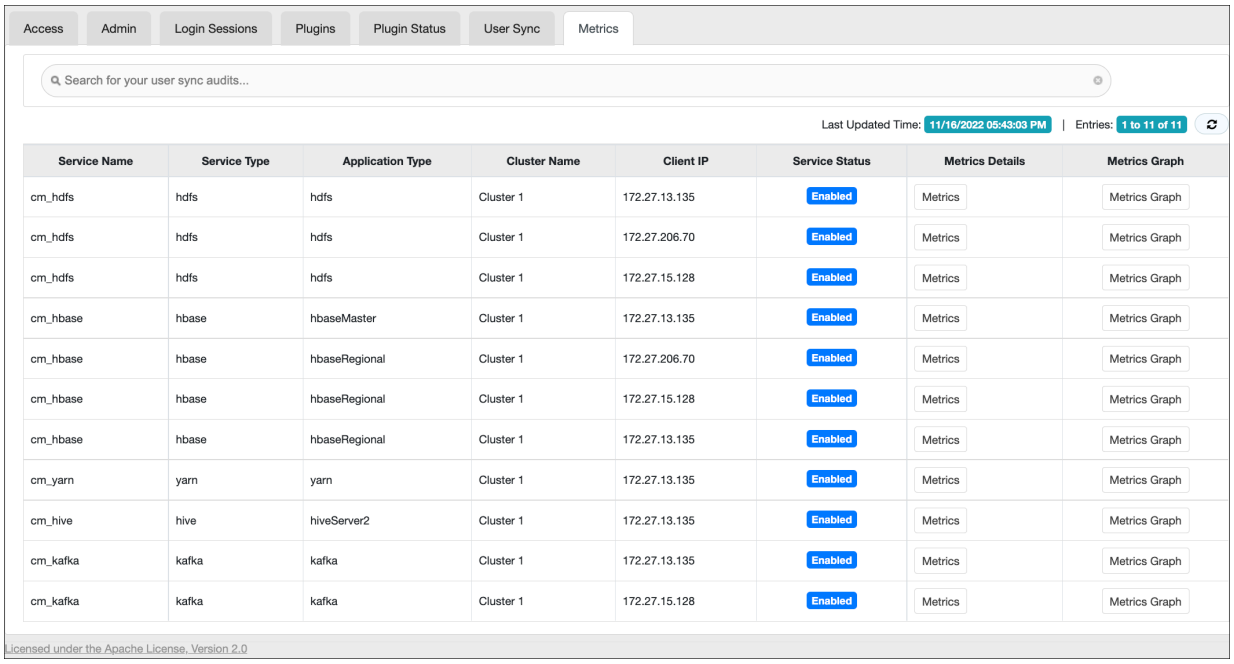

Service Name	Service Type	Application Type	Cluster Name	Client IP	Service Status	Metrics Details	Metrics Graph
cm_hdfs	hdfs	hdfs	Cluster 1	172.27.13.135	Enabled	Metrics	Metrics Graph
cm_hdfs	hdfs	hdfs	Cluster 1	172.27.206.70	Enabled	Metrics	Metrics Graph
cm_hdfs	hdfs	hdfs	Cluster 1	172.27.15.128	Enabled	Metrics	Metrics Graph
cm_hbase	hbase	hbaseMaster	Cluster 1	172.27.13.135	Enabled	Metrics	Metrics Graph
cm_hbase	hbase	hbaseRegional	Cluster 1	172.27.206.70	Enabled	Metrics	Metrics Graph
cm_hbase	hbase	hbaseRegional	Cluster 1	172.27.15.128	Enabled	Metrics	Metrics Graph
cm_hbase	hbase	hbaseRegional	Cluster 1	172.27.13.135	Enabled	Metrics	Metrics Graph
cm_yarn	yarn	yarn	Cluster 1	172.27.13.135	Enabled	Metrics	Metrics Graph
cm_hive	hive	hiveServer2	Cluster 1	172.27.13.135	Enabled	Metrics	Metrics Graph
cm_kafka	kafka	kafka	Cluster 1	172.27.13.135	Enabled	Metrics	Metrics Graph
cm_kafka	kafka	kafka	Cluster 1	172.27.15.128	Enabled	Metrics	Metrics Graph

2. To view metrics details for a specific service, click Metrics.

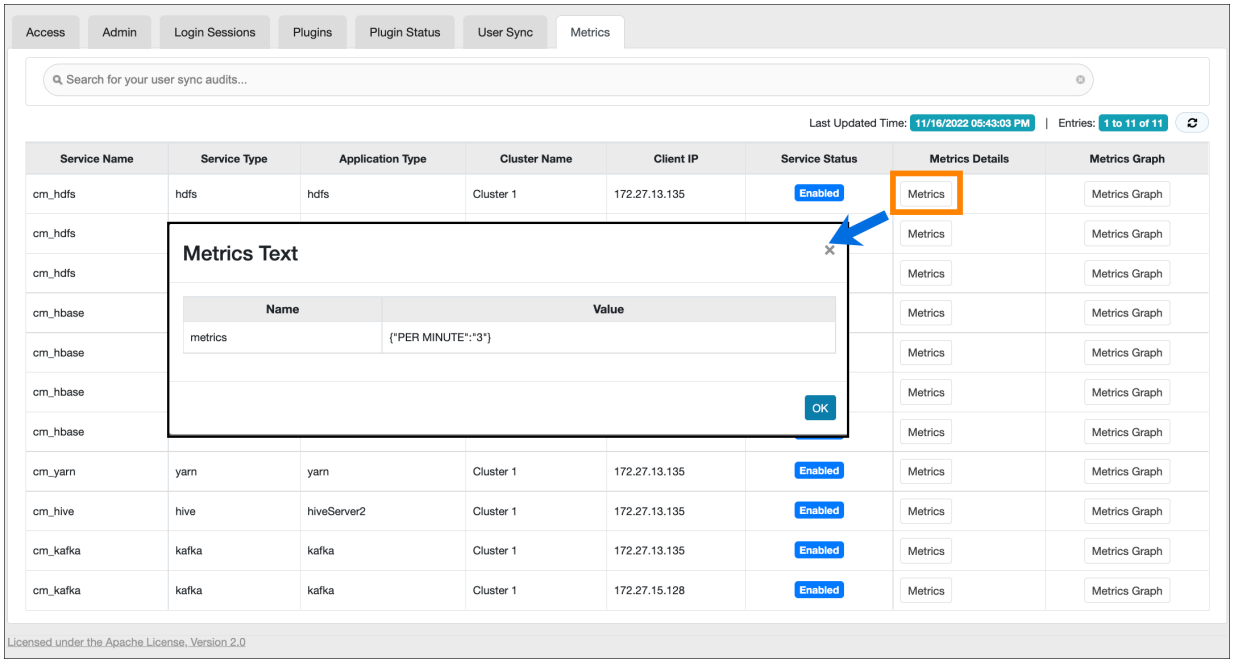

3. To view hourly or daily metrics as a graphic for a specific service, click Metrics Graph.

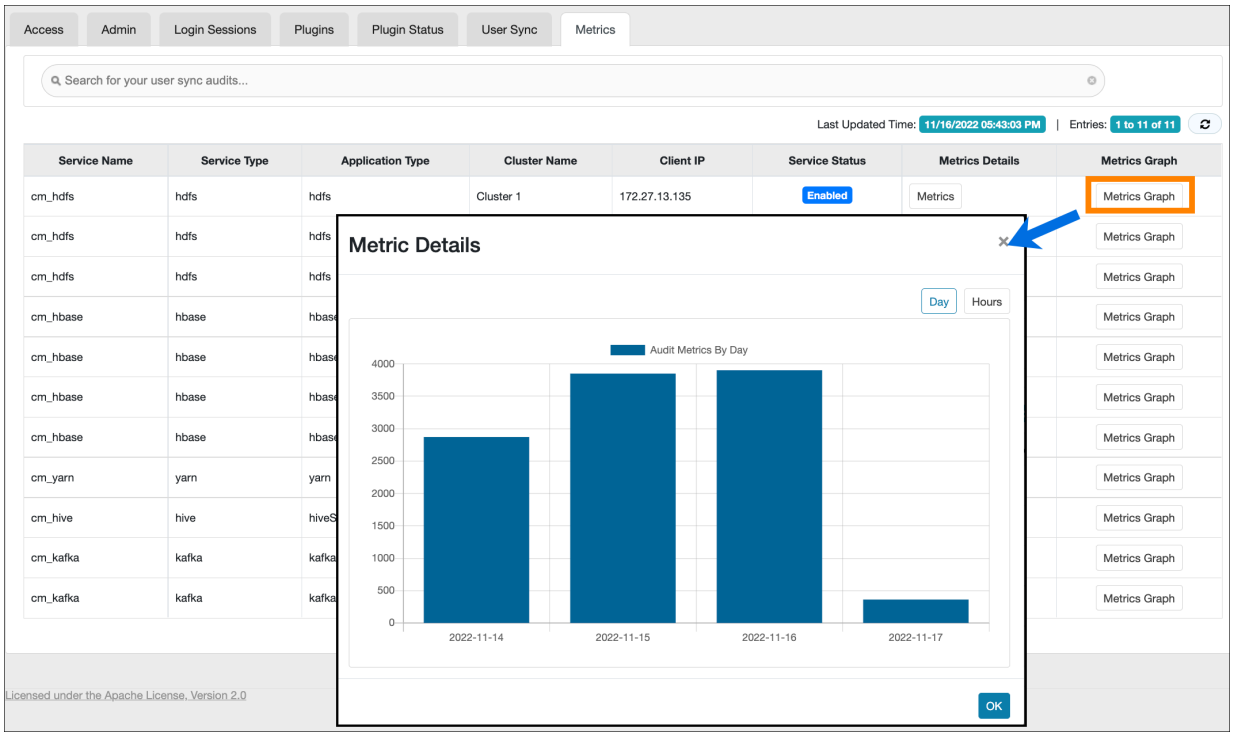

Creating a read-only Admin user (Auditor)

Creating a read-only Admin user (Auditor) enables compliance activities because this user can monitor policies and audit events, but cannot make changes.

About this task

When a user with the Auditor role logs in, they see a read-only view of Ranger policies and audit events. An Auditor can search and filter on access audit events, and access and view all tabs under Audit to understand access events. They cannot edit users or groups, export/import policies, or make changes of any kind.

Procedure

1. In Ranger Admin Web UI, select Settings > Users.
2. Click Add New User.
3. Complete the **User Detail** section, selecting Auditor as the role:

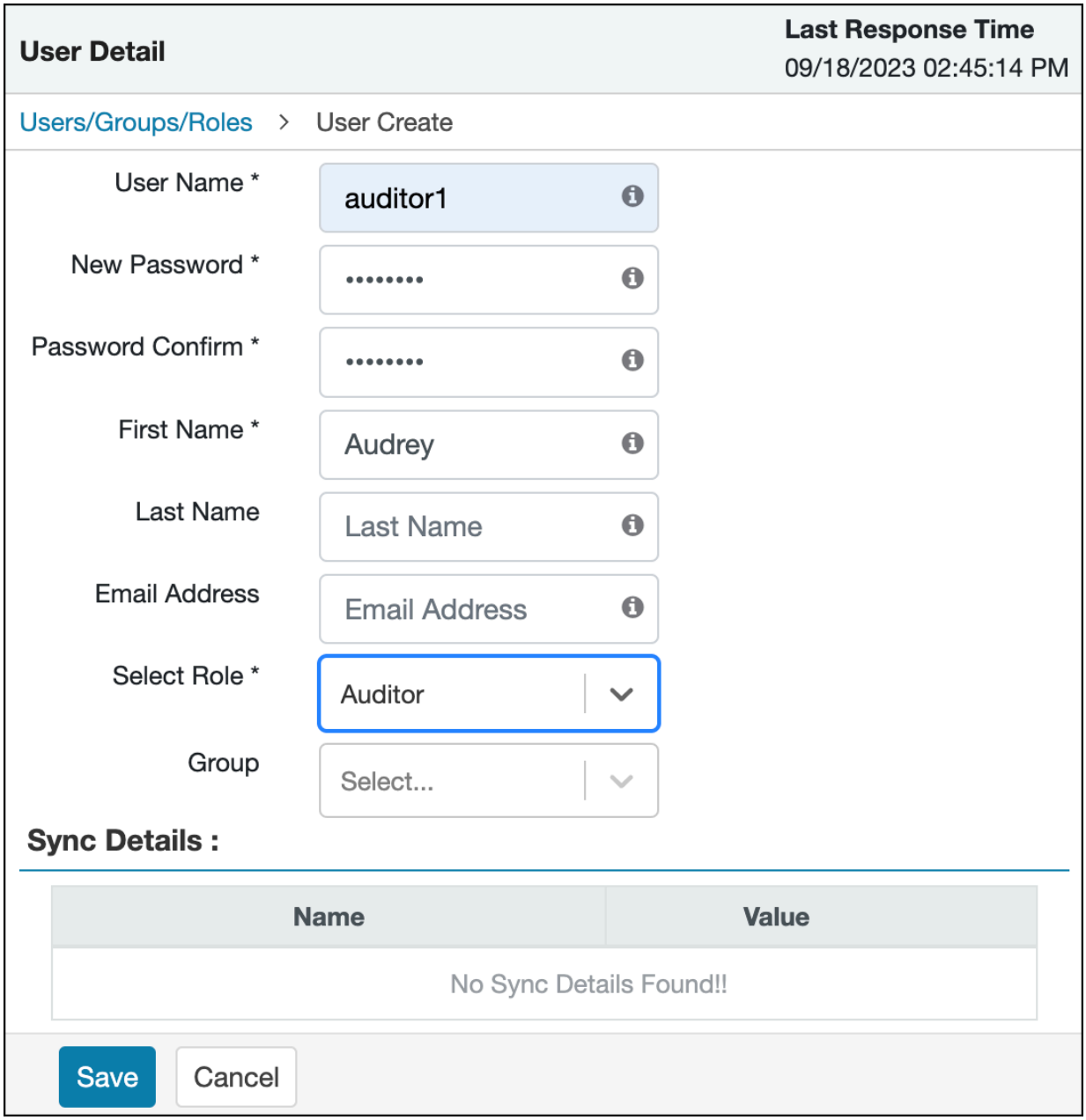

4. Click Save.

Configuring Ranger audit properties for Solr

How to change the default time settings that control how long Ranger keeps audit data collected by Solr.

About this task

The Solr audit destination is intended to store short term audit records. You can configure parameters that control how much data collected by Solr that Ranger will store for auditing purposes.

Table 1: Ranger Audit Configuration Parameters for Solr

Parameter Name	Description	Default Setting	Units
ranger.audit.solr.config.ttl	Time To Live for Solr Collection of Ranger Audits **Note:** "Time To Live for Solr Collection of Ranger Audits" is also known as the Max Retention Days attribute.	90	days
ranger.audit.solr.config.delete.trigger	Auto Delete Period in seconds for Solr Collection of Ranger Audits for expired documents	1	days (configurable)

Procedure

1. From Cloudera Manager choose Ranger Configuration .
2. In Search, type ranger.audit.solr.config, then press Return.
3. In ranger.audit.solr.config.ttl, set the the number of days to keep audit data.
4. In ranger.audit.solr.config.delete.trigger set the number and units (days, minutes, hours, or seconds) to keep data for expired documents.
5. Refresh the configuration:
   a) Click Refresh Configuration, as prompted.
   b) In Actions, click Update Solr config-set for Ranger, then confirm.

Limiting solr spool directory growth

To limit stored audit logs, you may set a maximum limit on the solr spool directory size for each service.

Procedure

1. Manually delete the logs under the archive path for the service.
2. Set the log retention value of the archive path from default 100 to 2 .
   a) In Cloudera Manager <service_name> Configuration <service_name> Service Advanced Configuration Snippet (Safety Valve) for <service_name>.xml
   b) Click +
   c) Add the following parameter:

      xasecure.audit.destination.solr.batch.filespool.archive.max.files

      2
3. Restart the service.

Configuring Ranger audit properties for HDFS

How to change the settings that control how Ranger writes audit records to HDFS.

About this task

The HDFS audit destination is intended to store long-term audit records.

You can configure whether Ranger stores audit records in HDFS and at which location.

You must purge long term audit records stored in HDFS manually.

Table 2: Ranger Audit Configuration Parameters for HDFS

Parameter Name	Description	Default Setting	Units
ranger_plugin_hdfs_audit_enabled	controls whether Ranger writes audit records to HDFS	true	T/F
ranger_plugin_hdfs_audit_url	location at which you can access audit records written to HDFS	<hdfs.host_name>/ranger/audit	string

Note: You can also disable storing ranger audit data to hdfs in each service specifically by setting xasecure.audit.destination.hdfs=false in that service.

Procedure

1. From Cloudera Manager choose Ranger Configuration .
2. In Search, type ranger_plugin, then press Return.
3. In ranger_plugin_hdfs_audit_enabled, check/uncheck RANGER-1 (Service Wide)
4. In ranger_plugin_hdfs_audit_url type a valid directory on the hdfs host.
5. Refresh the configuration, using one of the following two options:
   a) Click Refresh Configuration, as prompted or, if Refresh Configuration does not appear,
   b) In Actions, click Update Solr config-set for Ranger, then confirm.

What to do next

(Optional)

You may want to delete older logs from HDFS. Cloudera provides no feature to do this. You may accomplish this task manually, using a script.

Note:

The following example script is not supported by Cloudera. It is shown for reference only. You must test this successfully in a test environment before implementing it in a production cluster.

You must specify the audit log directory by replacing the 2nd line hdfs dfs -ls /<path_to>/<audit_logs> in the example script.

You may also include the -skipTrash option, if you choose, on 7th line in the script.

```
##################
today=`date +'%s'`
hdfs dfs -ls /<path_to>/<audit_logs> | grep "^d" | while read line ; do
dir_date=$(echo ${line} | awk '{print $6}')
difference=$(( ( ${today} - $(date -d ${dir_date} +%s) ) / ( 24*60*60 ) ))
filePath=$(echo ${line} | awk '{print $8}')

if [ ${difference} -gt 30 ]; then
    hdfs dfs -rm -r $filePath
fi
done
###################
```

Related Information

How to do a cleanup of hdfs files older than a certain date using a bash script

Triggering HDFS audit files rollover

How to configure when HDFS audit files close for each service.

About this task

By default, the Ranger Audit framework closes audit files created in HDFS or other cloud storage inline with audit event triggers. In other words, when an audit event occurs, Ranger checks the configured rollout time and then closes the file if the threshold has reached. Default audit rollout time is 24 hours. If no audit event occurs in a 24 hour period, files remain open beyond the 24 hour period. In some environments, audit log analysis that encounter an audit file open beyond the current date can cause system exceptions. If you want the files to be closed every day, so that the audit log file will have only that day's log and the next day’s log will be in the next day's file, you can configure the audit framework to close files every day. To do this, you must add several configuration parameters to the ranger-<service_name>-audit.xml (safety valve) file for each service, using Cloudera Manager.

Procedure

1. From Cloudera Manager choose <service_name> Configuration .
2. In <service_name> Configuration Search , type ranger-<service_name>, then press Return.
3. In <service_name> Server Advanced Configuration Snippet (Safety Valve) for ranger-<service_name>-audit.xml, do the following steps:
   a) Click + (Add).
   b) In Name, type xasecure.audit.destination.hdfs.file.rollover.enable.periodic.rollover
   c) In Value, type true.

      When this is enabled Ranger Audit Framework will spawn a Scheduler thread which monitors the occurrence of closing threshold and closes the file. By default every night the file gets closed.
   d) Click + (Add another).
   e) In Name, type xasecure.audit.destination.hdfs.file.rollover.sec
   f) In Value, type an integer value in seconds.

      This is the time in seconds when the file has to be closed. The default value is 86400 sec (1 day) which triggers the file to be closed at midnight and opens a new audit log for the next day. You can override the default value

can be overridden by setting this parameter. For example, if you set the value 3600 (1 hr), the file gets closed every hour.

g) Click + (Add another).

h) In Name, type xasecure.audit.destination.hdfs.file.rollover.periodic.rollover.check.sec

i) In Value, type an integer value in seconds.

This is the time frequency of the check to be done whether the threshold time for rollover has occurred. By default the check is done every 60 secs. You can configure this parameter to delay the check time.

Figure 2: Example: Hive service configured to trigger rollover of hdfs audit files

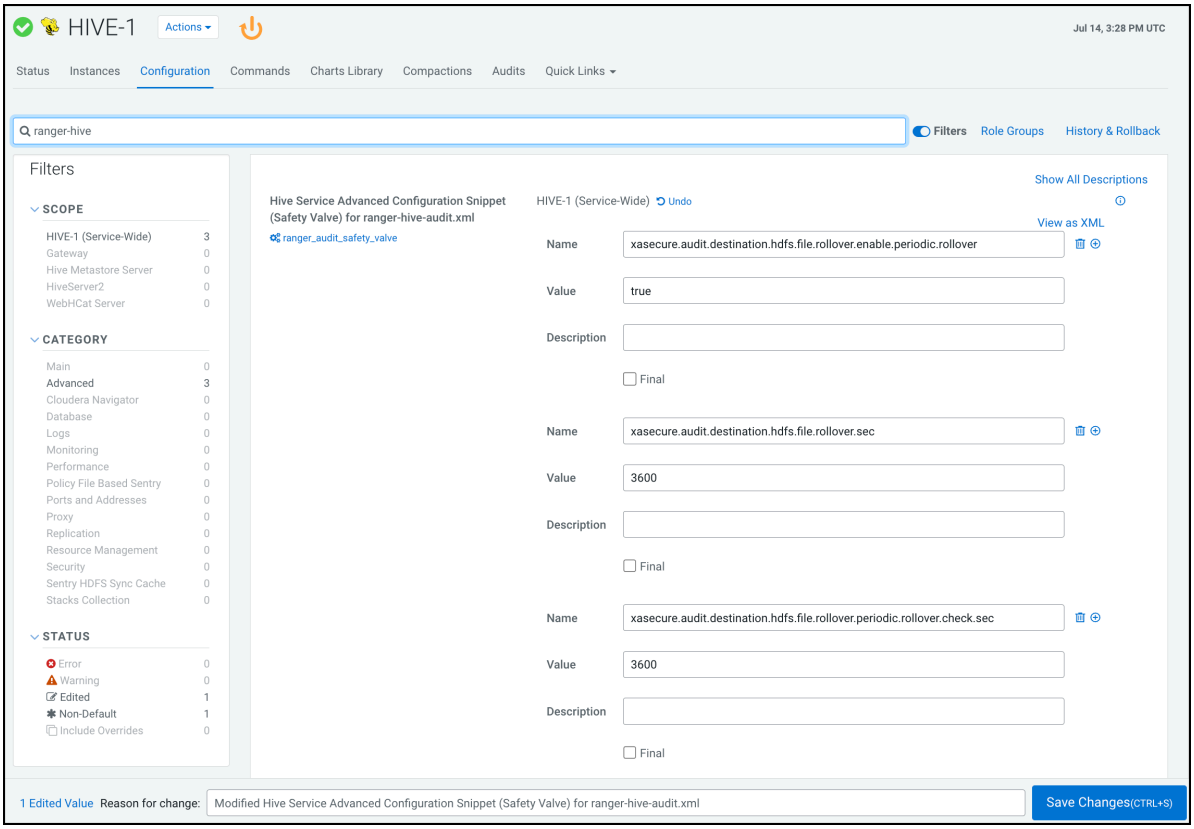

j) Click Save Changes (CTRL+S).

4. Repeat steps 1-3 for each service.
5. Restart the service.

Configuring Ranger audit log storage to a local file

How to configure a service to write Ranger audit log records to a local file, in addition to the default Solr and HDFS locations.

About this task

The Solr and HDFS audit destinations are intended to store short- and long-term audit records respectively, by default. You can configure whether a service also stores Ranger audit records in a local file destination. You may also further configure a service by appending additional, service-specific configuration properties. To do this, first modify ranger-<service_name>-audit.xml file, using Cloudera Manager. Then, add additional appenders to the configuration. This topic shows how to add appenders to the Hive service, for example.

Table 3: Ranger Audit Destination Configuration Parameters for services

Parameter Name	Description	Default Setting	Units
xasecure.audit.destination.log4j	controls whether Ranger writes audit records to a service	true	T/F
xasecure.audit.destination.log4j.logger	location at which you can access audit records written to HDFS	rangerauditlogger	string

Note: You can also disable storing ranger audit data to hdfs in each service specifically by setting xasecure.audit.destination.hdfs=false in that service.

Procedure

1. Configure Ranger Log4JAuditDestination.
   a) From Cloudera Manager choose <service_name> Configuration .
   b) In Search, type ranger-<service_name>-audit, then press Return.
   c) In Hive Service Advanced Configuration Snippet (Safety Valve) for ranger-hive-audit.xml, click +.
   d) Add the following property/value:
      Name
      xasecure.audit.destination.log4j
      Value
      true
   e) Click Save Changes (CTRL+S).
   f) In Hive Service Advanced Configuration Snippet (Safety Valve) for ranger-hive-audit.xml, click +.
   g) Add the following property/value
      Name
      xasecure.audit.destination.log4j.logger
      Value
      rangerauditlogger
   h) Click Save Changes (CTRL+S).
2. Configure HIVE log4 for ranger audit log appender.
   Add the following properties to the configuration: (?? HiveServer2 Logging Advanced Configuration Snippet (Safety Valve) ??)

```
appenders=console, DRFA, redactorForRootLogger, RANGERAUDIT
appenders=console, DRFA, redactorForRootLogger, RANGERAUDIT
appender.RANGERAUDIT.type=RollingRandomAccessFile
appender.RANGERAUDIT.name=RANGERAUDIT
appender.RANGERAUDIT.fileName=${log.dir}/hive-ranger-audit.log
appender.RANGERAUDIT.filePattern=${log.dir}/hive-ranger-audit.log.%d{yyyy-
MM-dd}-%i
appender.RANGERAUDIT.filePermissions=rwxr--r--
appender.RANGERAUDIT.layout.type=PatternLayout
appender.RANGERAUDIT.layout.pattern=%d{ISO8601} %q %5p [%t] %c{2} (%F:%
M(%L)) - %m%n
appender.RANGERAUDIT.policies.type=Policies
appender.RANGERAUDIT.policies.size.type=SizeBasedTriggeringPolicy
appender.RANGERAUDIT.policies.size.size=250MB
appender.RANGERAUDIT.strategy.type=DefaultRolloverStrategy
appender.RANGERAUDIT.strategy.max=10
appender.RANGERAUDIT.strategy.action.type=DELETE
appender.RANGERAUDIT.strategy.action.basepath=${log.dir}
appender.RANGERAUDIT.strategy.action.maxdepth=1
```

```
appender.RANGERAUDIT.strategy.action.PathConditions.glob=hive-ranger-aud
it.*
appender.RANGERAUDIT.strategy.action.PathConditions.type=IfFileName
appender.RANGERAUDIT.strategy.action.PathConditions.nestedConditions.ty
pe=IfAccumulatedFileCount
appender.RANGERAUDIT.strategy.action.PathConditions.nestedConditions.excee
ds=2GB
loggers=Ranger
logger.Ranger.name=rangerauditlogger
logger.Ranger.level=INFO
logger.Ranger.appenderRefs=RANGERAUDIT
logger.Ranger.appenderRef.RANGERAUDIT.ref=RANGERAUDIT
```

3. Restart the service to refresh the configuration.

   The result will be hive-ranger-audit.log getting created in the configured folder. Ranger audit logs for hive operations will be appended.

   Note:

   Change the max number of files to retain the files in the folder. Scripting has to be done to periodically move the hive-ranger-audit.log.* to the processing directory if this feature is used to send these audit logs to other destinations via external applications.

Ranger Audit Filters

You can use Ranger audit filters to control the amount of audit log data collected and stored on your cluster.

About Ranger audit filters

Ranger audit filters allow you to control the amount of audit log data for each Ranger service. Audit filters are defined using a JSON string that is added to each service configuration. The audit filter JSON string is a simplified form of the Ranger policy JSON. Audit filters appear as rows in the Audit Filter section of the Edit Service view for each service. The set of audit filter rows defines the audit log policy for the service. For example, the default audit log policy for the Hadoop SQL service appears in Ranger Admin web UI Service Manager Edit Service when you scroll down to Audit Filter. Audit Filter is checked (enabled) by default. In this example, the top row defines an audit filter that causes all instances of "access denied" to appear in audit logs. The lower row defines a filter that causes no metadata operations to appear in audit logs. These two filters comprise the default audit filter policy for Hadoop SQL service.

Figure 3: Default audit filter policy for the Hadoop SQL service

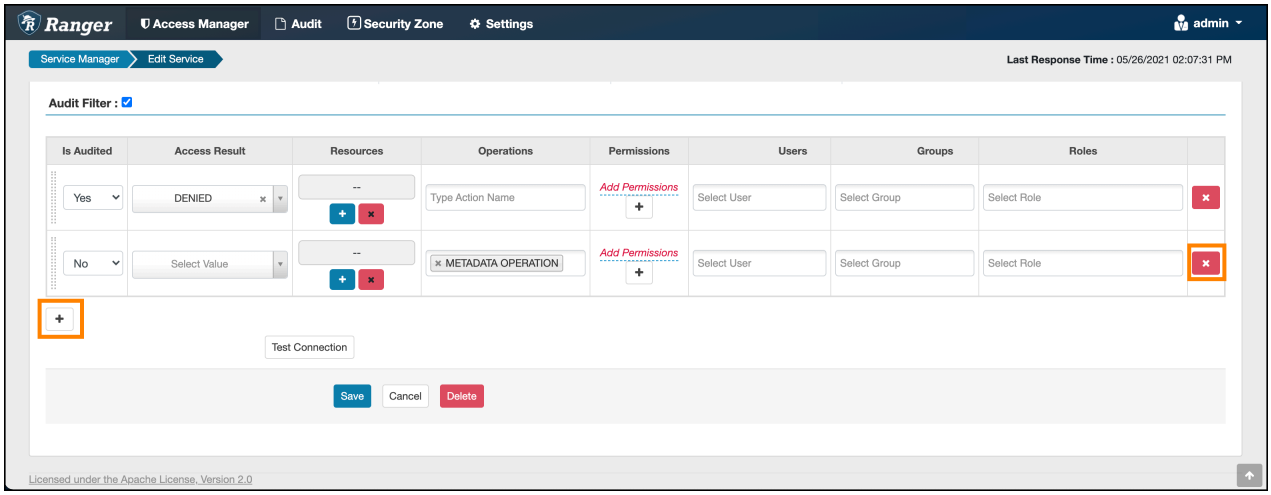

Default Ranger audit filters

Default audit filters for the following Ranger service appear in Edit Services and may be modified as necessary by Ranger Admin users.

HDFS

Figure 4: Default audit filters for HDFS service

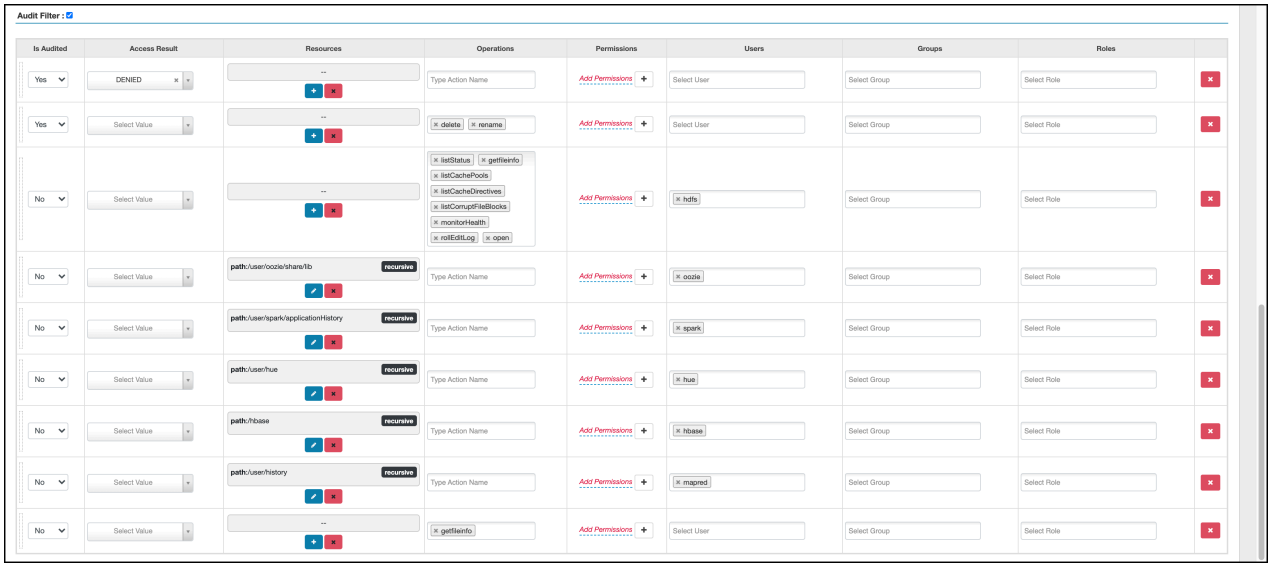

Hbase

Figure 5: Default audit filters for the Hbase service

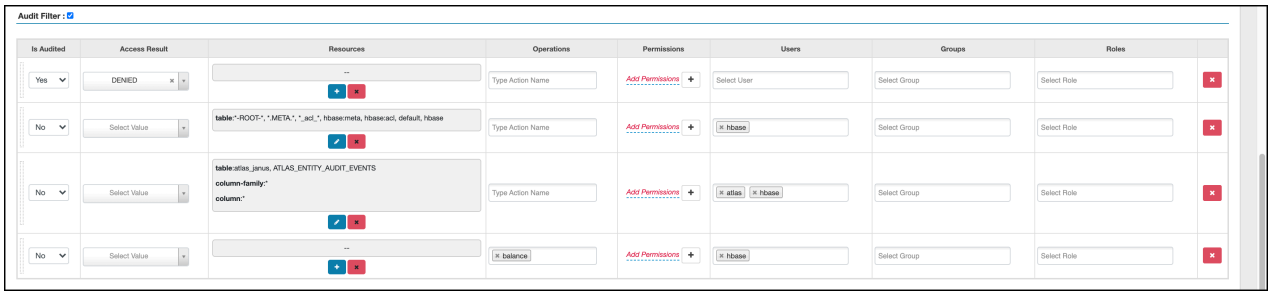

Hadoop SQL

Figure 6: Default audit filters for the Hadoop SQL service

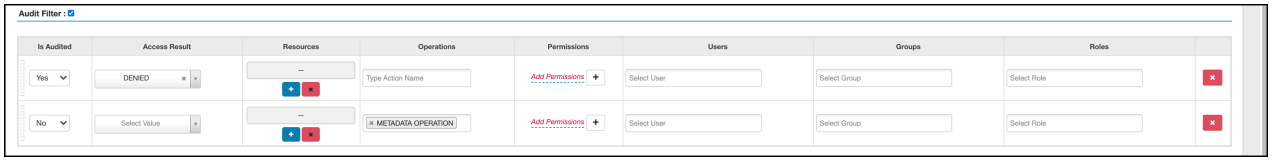

Knox

Figure 7: Default audit filters for the Knox service

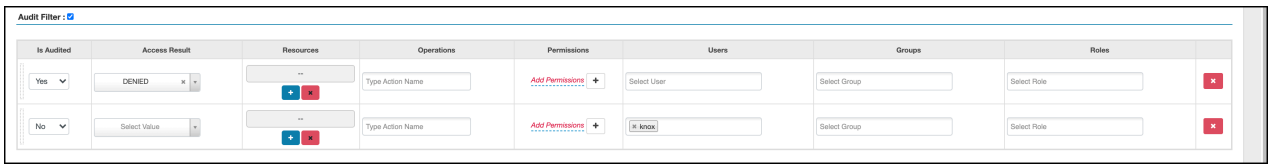

Solr

Figure 8: Default audit filters for the Solr service

Kafka

Figure 9: Default audit filters for the Kafka service

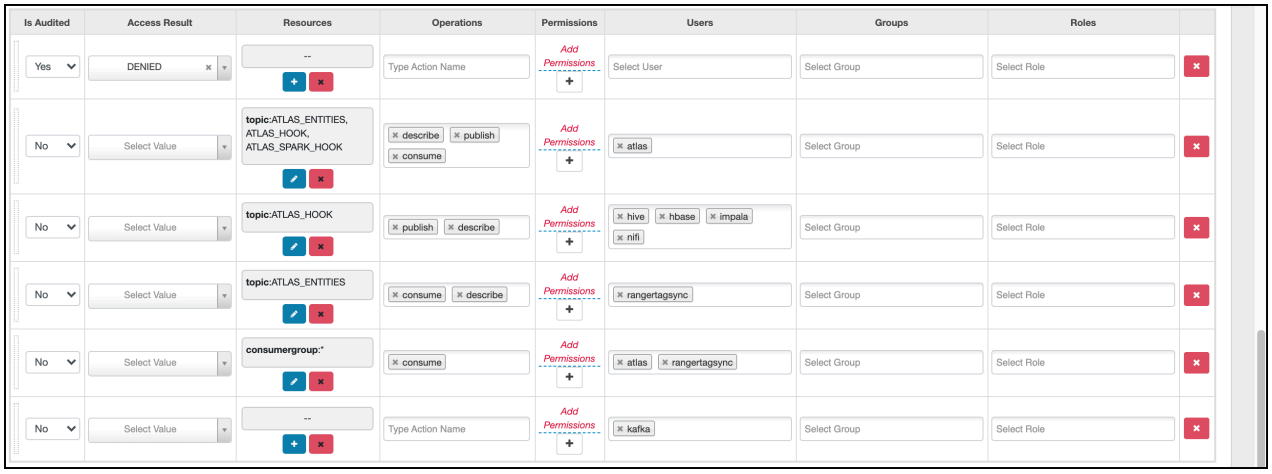

Ranger KMS

Figure 10: Default audit filters for the Ranger KMS service

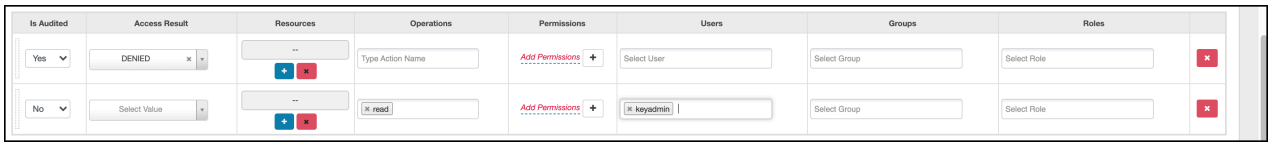

Atlas

Figure 11: Default audit filters for the Atlas service

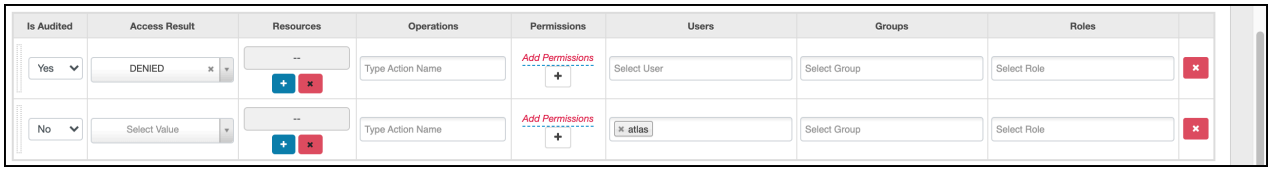

ADLS

Figure 12: Default audit filters for the ADLS service

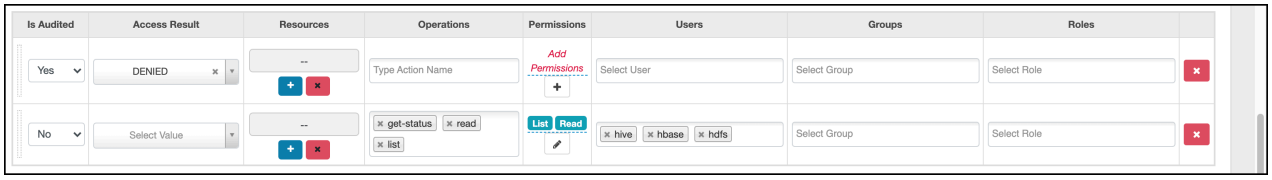

Ozone

Figure 13: Default audit filters for the Ozone service

S3

Figure 14: Default audit filters for the S3 service

Tag-based services

Figure 15: Default audit filters for a tag-based service

Note:

Default audit filter policies do not exist for Yarn, NiFi, NiFi Registry, Kudu, or schema registry services.

Configuring a Ranger audit filter policy

You can configure an audit filter as you add or edit a resource- or tag-based service.

To configure an audit filter policy

1. In Ranger Admin web UI Service Manager clcik Add or Edit for either a resource-, or tag-based service.
2. Scroll down to Audit Filter.
3. Click Audit Filter flag.

   You configure a Ranger audit filter policy by adding (+), deleting (X), or modifying each audit filter row for the service.

4. Use the controls in the filter row to edit filter properties. For example, you can configure:
   Is Audited: choose Yes or No

   to include or not include a filter in the audit logs for a service

   Access Result: choose DENIED, ALLOWED, or NOT_DETERMINED

   to include that access result in the audit log filter

   Resources: Add or Delete a resource item

   to include or remove the resource from the audit log filter

   Operations: Add or Remove an action name

   to include the action/operation in the audit log filter

   (click x to remove an existing operation)

   Permissions: Add or Remove permissions

   a. Click + in Permissions to open the Add dialog.
   b. Select/Unselect required permissions.

   For example, in HDFS service select read, write, execute, or All permissions.

   Users: click Select User to see a list of defined users

   to include one or multiple users in the audit log filter

   Groups: click Select Group to see a list of defined groups

   to include one or multiple groups in the audit log filter

   Roles: click Select Role to see a list of defined roles

   to include one or multiple roles in the audit log filter

Audit filter details

- When you save the UI selections described in the preceding list, audit filters are defined as a JSON list. Each service references a unique list.
- For example, ranger.plugin.audit.filters for the HDFS service includes:

```
[
                {
                "accessResult":"DENIED",
                "isAudited":true
                },
                {
                "users":[
                "unaudited-user1"
                ],
                "groups":[
                "unaudited-group1"
                ],
                "roles":[
                "unaudited-role1"
                ],
                "isAudited":false
                },
                {
                "actions":[
                "listStatus",
                "getfileinfo"
                ],
                "accessTypes":[
                "execute"
                ],
```

```
                "isAudited":false
                },
                {
                "resources":{
                "path":{
                "values":[
                "/audited"
                ],
                "isRecursive":true
                }
                },
                "isAudited":true
                },
                {
                "resources":{
                "path":{
                "values":[
                "/unaudited"
                ],
                "isRecursive":true
                }
                },
                "isAudited":false
                }
                ]
```

- Each value in the list is an audit filter, which takes the format of a simplified Ranger policy, along with access results fields.
- Audit filters are defined with rules on Ranger policy attributes and access result attributes.
  - Policy attributes: resources, users, groups, roles, accessTypes
  - Access result attributes: isAudited, actions, accessResult
- The following audit filter specifies that accessResult=DENIED will be audited.

  The isAudited flag specifies whether or not to audit.

  ```
  {"accessResult":"DENIED","isAudited":true}
  ```
- The following audit filter specifies that “resource => /unaudited” will not be audited.

  ```
  {"resources":{"path":{"values":["/
  unaudited"],"isRecursive":true}},"isAudited":false}
  ```
- The following audit filter specifies that access to resource database=> sys table=> dump by user “use2” will not be audited.

  ```
  {"resources":{"database":{"values":["sys"]},"table":{"values":
  ["dump"]}},"users":["user2"],"isAudited":false}
  ```
- The following audit filter specifies that access result in actions => listStatus, getfileInfo and accessType => execute will not be audited.

  ```
  {"actions":["listStatus","getfileinfo"],"accessTypes":
  ["execute"],"isAudited":false}
  ```
- The following audit filter specifies that access by user "superuser1" and group "supergroup1" will not be audited.

  ```
  {"users":["superuser1"],"groups":["supergroup1"],"isAudited":false}
  ```
- The following audit filter specifies that access to any resource tagged as NO_AUDIT will not be audited.

  ```
  {"resources":{"tag":{"values":["NO_AUDIT"]}},"isAudited":false}
  ```

How to set audit filters in Ranger Admin Web UI

You can set specific audit filter conditions for each service, using Create/Edit Service .

About this task

Creating audit filters for a service using the Ranger Admin Web UI can prevent audit logs from being sent to destinations like SOLR and HDFS.

Procedure

1. In the Ranger Admin Web UI Service Manager , click Add New Service or Edit (existing service).
2. On Create/Edit Service, scroll down to Audit Filters.
   a) Verify that Audit Filter is checked.

      Optionally, define any of the following to include in the filter definition:

      Is Audited

      Defines whether audit logs are stored or not.

      Is Audited=Yes: stores audit records in the defined audit destination.

      Is Audited=No: do not store audit records.

      Access Results

      Denied, Allowed, or Not Determined

      select to filter access=denied, access=allowed or access=Not determined

      Resource

      use Resource Details to include or exclude specific resources such as databases, tables, or columns.

      Operations

      select specific operations to filter

      Note: For Operations field, you can refer to the Ranger access audit page and find out the available access types.

      Permissions

      select specific permissions

      Users, Groups, Roles

      select specific users, groups, and roles

   b) Click Save.

Figure 16: Adding an audit filter that stores user systest, access=Allowed logs for Hive service

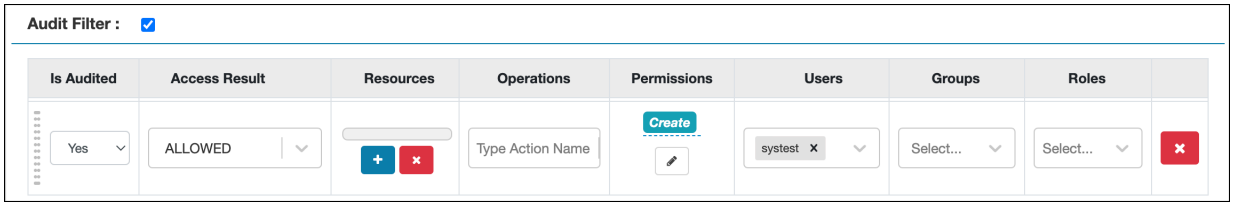

3. Test your filters to verify that defined audit filters perform as expected.

Results

Defining specific filtering properties can prevent access logs for service users from being stored in the configured audit destination, if Is Audited = No.

Filter service access logs from Ranger UI

You can limit display of system access/audit log records generated by service users in each service.

About this task

This topic describes how to limit the display of access log records on the Access tab in the Ranger Admin Web UI.

Procedure

1. Go to Ranger Admin Web UI Audit Access .
2. Check the Exclude Service Users box, as shown in:

Figure 17: Setting the Exclude Service Users flag to true

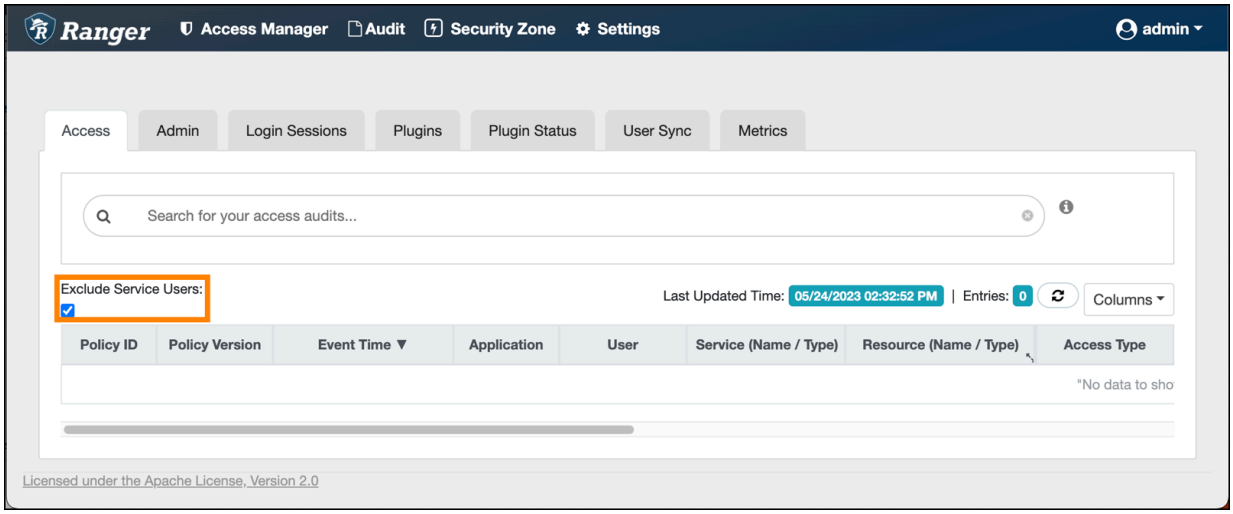

3. Define specific component services and users for access logs to filter out, in ranger-admin-site.xml.
   a) Go to Cloudera Manager Ranger Configuration
   b) In Search, type ranger-admin-site.
   c) Define the following properties:

      Name

      ranger.plugins.<service_name>.serviceuser

      Value

      <service_name>

      Name

      ranger.accesslogs.exclude.users.list

      Value

      user1, user2

Figure 18: Filtering out service and user logs for Hive service

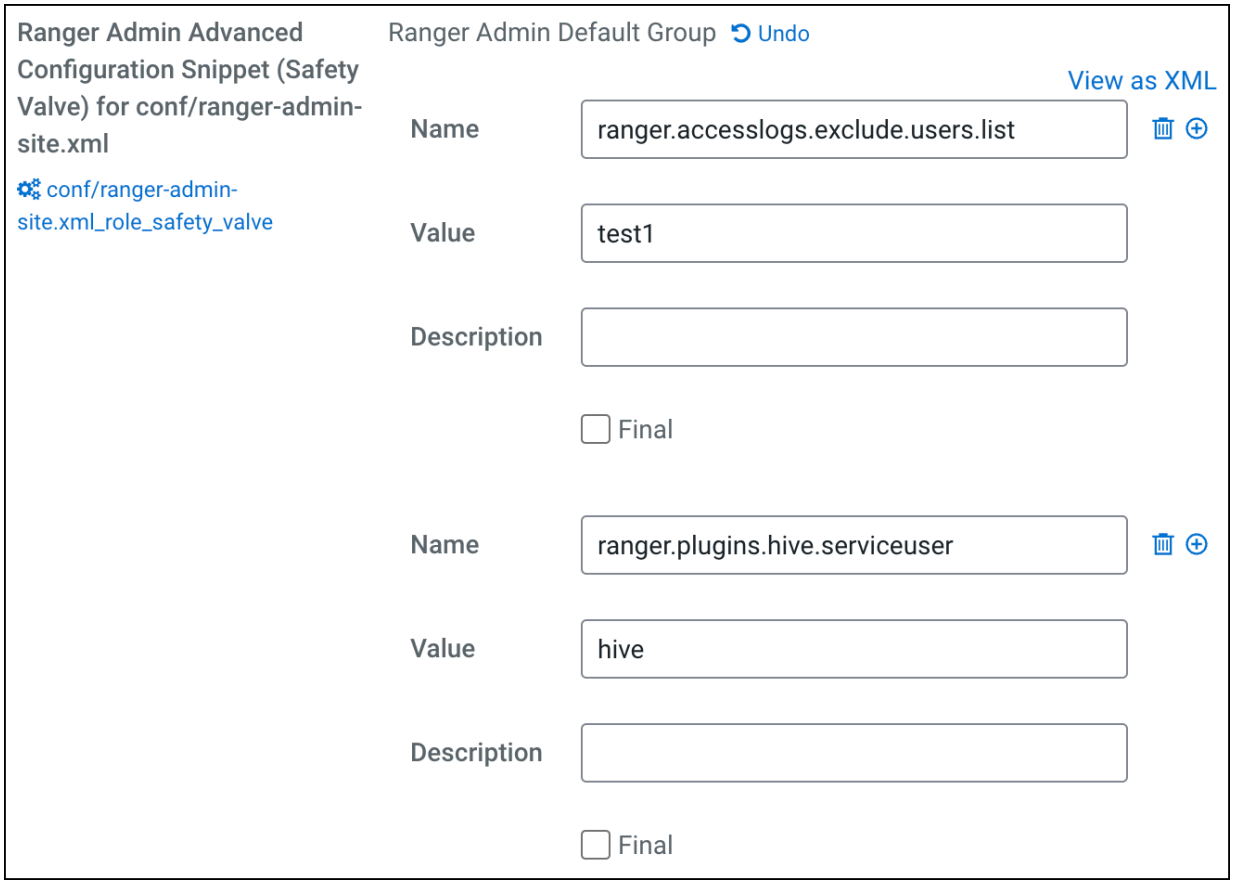

4. Click Save Changes (CTRL+S).
5. Restart the Ranger service.

Results

Setting Exclude Service Users to true and defining specific filtering properties prevents audit logs from service users from appearing on Ranger Admin Web UI Audit Access , but does NOT prevent access logs for service users from being generated in Solr.

Configuring audit spool alert notifications

You can enable and configure alerts for Ranger plugin supported services on Cloudera Manager that notify when audit spool files are accumulated.

About this task

Ranger stores plugin access audit events (audit logs) in Solr and in HDFS. Typically, you store audit logs in Solr for short-term auditing purposes and in HDFS for longer-term purposes. When the Solr server is down, Ranger plugin audit logs are stored in the spool directory as spool files. You configure the spool directory location using the following properties:

For Solr

xasecure.audit.destination.solr.batch.filespool.dir = /var/log/<service_name>/audit/solr/spool

For HDFS

xasecure.audit.destination.hdfs.batch.filespool.dir = /var/log/<service_name>/audit/hdfs/spool

If the Solr server goes down for a long period of time or, if a large number of audit events occur while the Solr server is down; then spool files accumulate in the spool directory. Spool file accumulation consumes system memory. Sometimes, audit records in spool files become corrupted, and may not be restored when the Solr server returns to a running state. Corrupted, un-restored records also cause spool file accumulation. This requires manual cleanup of corrupted spool files. An unnoticed large accumulation or "piling up" of spool files may fill the local filesystem and result in service failure.

After you enable spool directory metric usage for a service, an alert appears on the Cloudera Manager UI which notifies the user when spool files have piled up in the spool directory. The Cloudera Manager agent measures the disk usage of the spool directory and registers it as a metric value. This metric value is compared against a threshold value. The spool alert appears on the Cloudera Manager UI if the metric value is greater than the threshold value.

Single-level metrics for collecting the disk usage of Solr and HDFS Ranger plugin spool directories are registered in Cloudera Manager, using the following disk usage metric names:

Solr

ranger_plugin_solr_spool_directory_size

HDFS

ranger_plugin_hdfs_spool_directory_size

The following table lists Ranger plugin supported service names and roles that support spool alerts.

Table 4: Ranger plugin supported services and their roles supporting audit spool alerts

Services	Roles
HDFS	NAMENODE
HIVE	HIVEMETASTORE
HIVE_ON_TEZ	HIVESERVER2
HBASE	MASTER, REGIONSERVER
YARN	RESOURCEMANAGER
IMPALA	IMPALAD, CATALOGSERVER
ATLAS	ATLAS_SERVER
KAFKA	KAFKA_BROKER
KNOX	KNOX_GATEWAY
KUDU	KUDU_MASTER

Services	Roles
RANGER_KMS	RANGER_KMS_SERVER
RANGER_RAZ	RANGER_RAZ_SERVER
SCHEMAREGISTRY	SCHEMA_REGISTRY_SERVER
STREAMS_MESSAGING_MANAGER	STREAMS_MESSAGING_MANAGER_SERVER

To enable or disable spool directory alerts:

Procedure

1. Go to Cloudera Manager Ranger Plugin Supported Services page Configuration .
2. In Search, type spool directory.

   The following configuration properties display (this example uses the Kafka Broker role from Kafka service)

   Figure 19: Audit Spool Alert Configurations for Kafka Broker

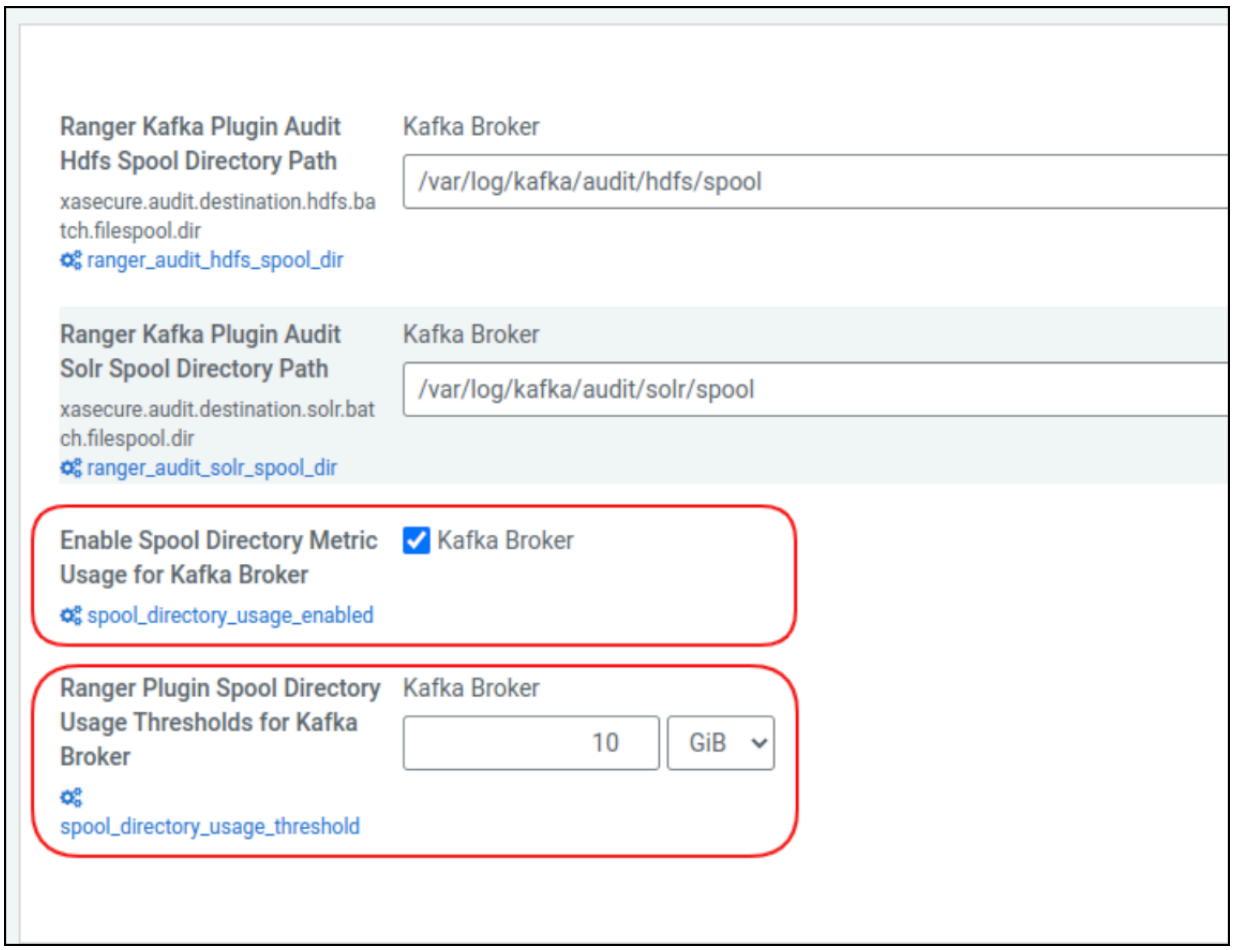

3. In Enable Spool Directory Metric Usage for <service-name>, check the box.

   (un-checking the Enable box disables spool directory metric usage)

   a) Refresh the role.

      For example, the Kafka service which supports Ranger Plugin in Kafka Broker role, go to Cloudera Manager Kafka Instances Kafka Broker Actions Refresh Kafka Broker .

4. In Ranger Plugin Spool Directory Usage Thresholds for <service-name>, type values and select units.

   By default the threshold is set to 1 GB. If the disk usage of the spool directory exceeds this threshold, an alert will be shown.

Results

These spool alert details appear on the Ranger plugin service Role status page in the Cloudera Manager UI, as shown for Kafka Broker role in the following example:

Figure 20: Audit spool details for the Kafka Broker role

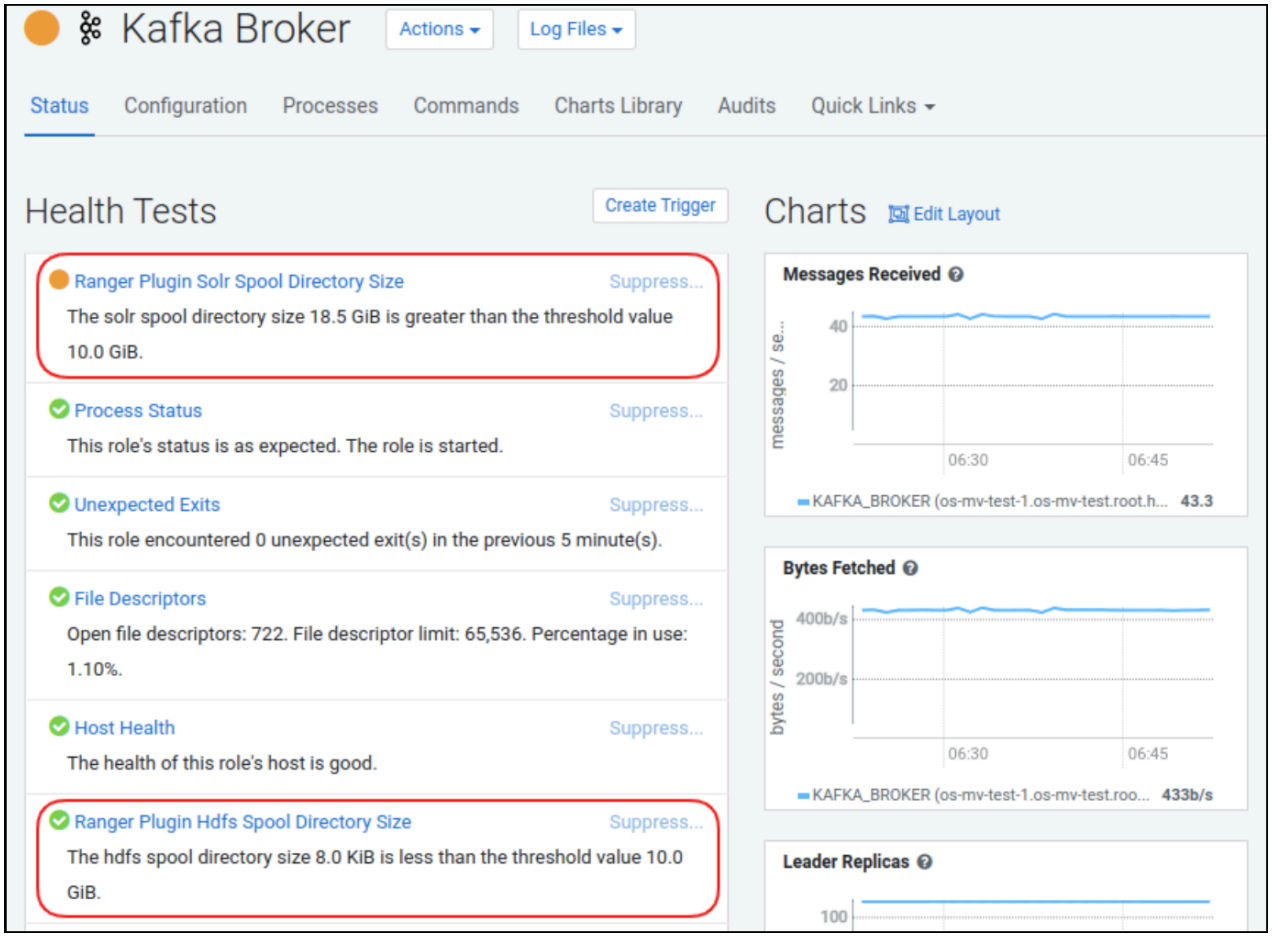

Also, an alert appears on the Cloudera Manager Home page next to the service name, as shown for Kafka service in the following example.

Figure 21: Audit file spool notification for Kafka service plugin on Cloudera Manager home page.

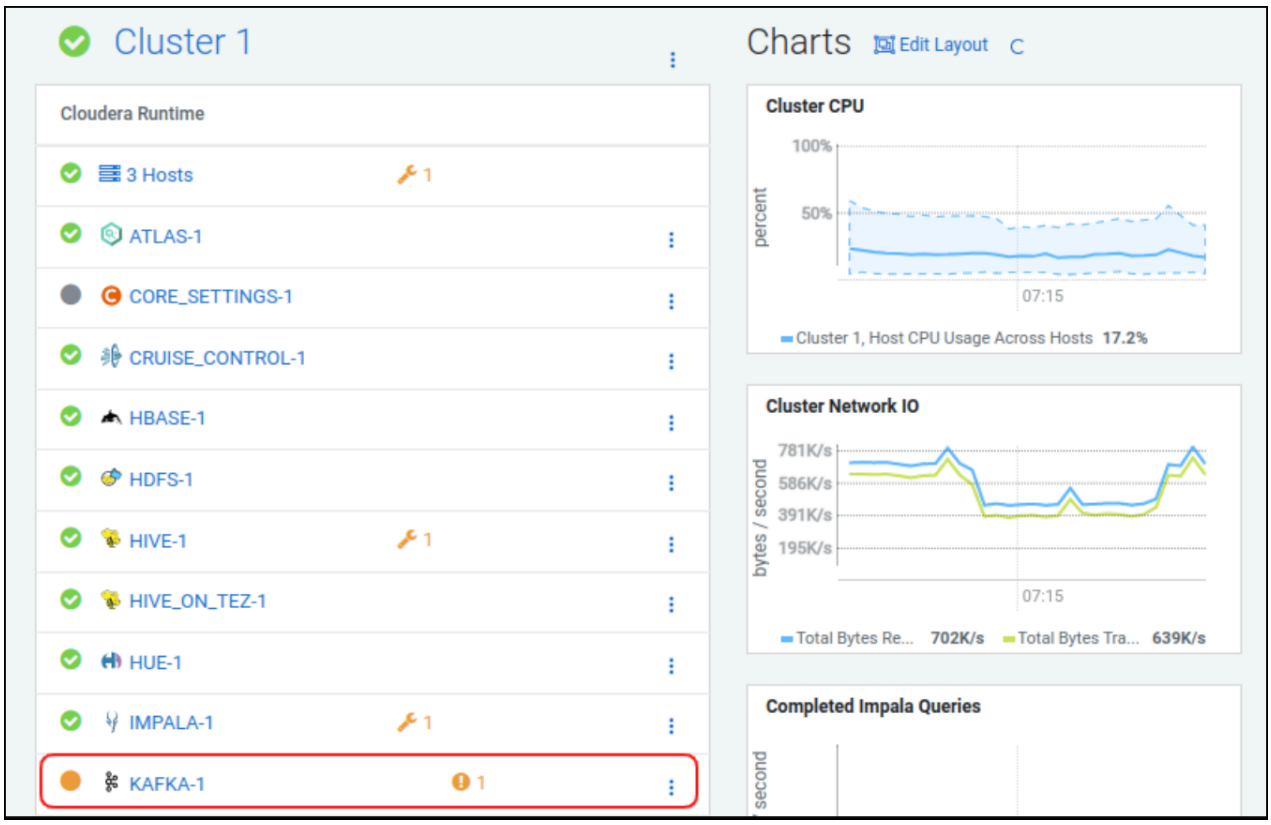

Also, Cloudera Manager shows Cluster Heath Actions and Advice for the role, as shown in the following example:

Figure 22: Health Test For Ranger Plugin Spool Alert

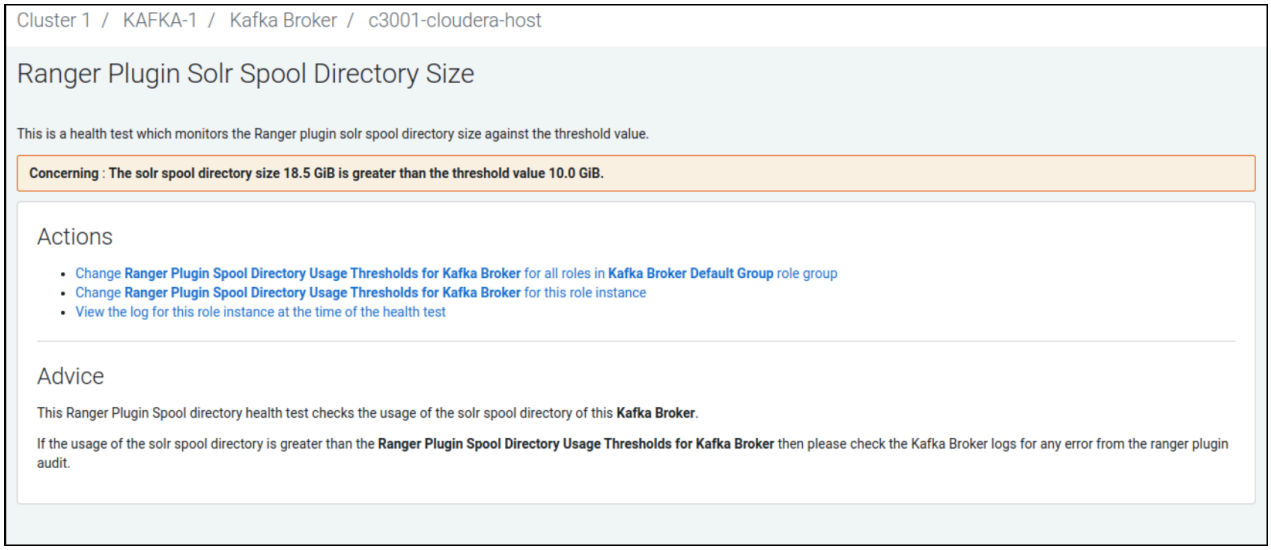

What to do next

Optionally, you can create a graph for each spool alert metric.

Related Information

Charting spool alert metrics

Charting spool alert metrics

Cloudera Manager supports chart visualization of spool directory disk usage metrics.

Single-level metrics for collecting the disk usage of Solr and Hdfs Ranger plugin spool directories are registered in Cloudera Manager, using the following disk usage metric names:

Solr

ranger_plugin_solr_spool_directory_size

HDFS

ranger_plugin_hdfs_spool_directory_size

You can chart disk usage of Solr and HDFS spool directory size as a time-series, using Cloudera Manager Chart Builder .

For example, to chart the disk usage for the Solr spool directory defined for Kafka Broker, use the folllowing query syntax:

SELECT ranger_plugin_solr_spool_directory_size WHERE roleType = KAFKA_BROKER, as shown in

Figure 23: Charting the disk usage of the Ranger Solr plugin spool directory for Kafka Broker

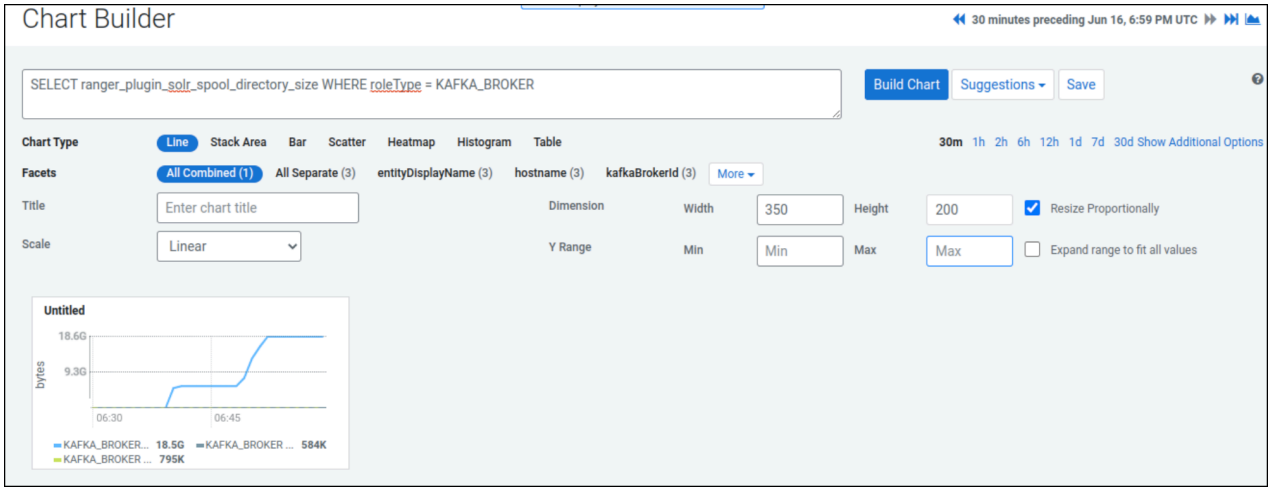

For more specific information about charting time-series data, see:

Related Information

Building a Chart with Time-Series Data

Excluding audits for specific users, groups, and roles

You can exclude audit records for specific users, groups, and roles from each service from appearing in the Ranger UI.

About this task

Ranger default log functionality creates audit log records for access and authorization requests, specifically around service accounts such as hbase, atlas and solr. Writing so much data to solr can limit the availability of Solr for further usage. This topic describes how to exclude audit records for specific users, groups, and roles from each service from appearing in the Ranger UI. Excluding specific users, groups or roles is also known as creating a blacklist for Ranger audits.

Procedure

1. In the Ranger Admin Web UI Service Manager , click Add New Service or Edit (existing service).
2. On Create/Edit Service, scroll down to Config Properties Add New Configurations .
3. Remove all audit filters from the existing service.
4. Click +, then type one of the following property names:
   - ranger.plugin.audit.exclude.users
   - ranger.plugin.audit.exclude.groups
   - ranger.plugin.audit.exclude.roles

   followed by one or more values.

   Note: You can include multiple values for each exclude property using a comma-separated list.

Figure 24: Adding an exclude users property to the HadoopSQL service

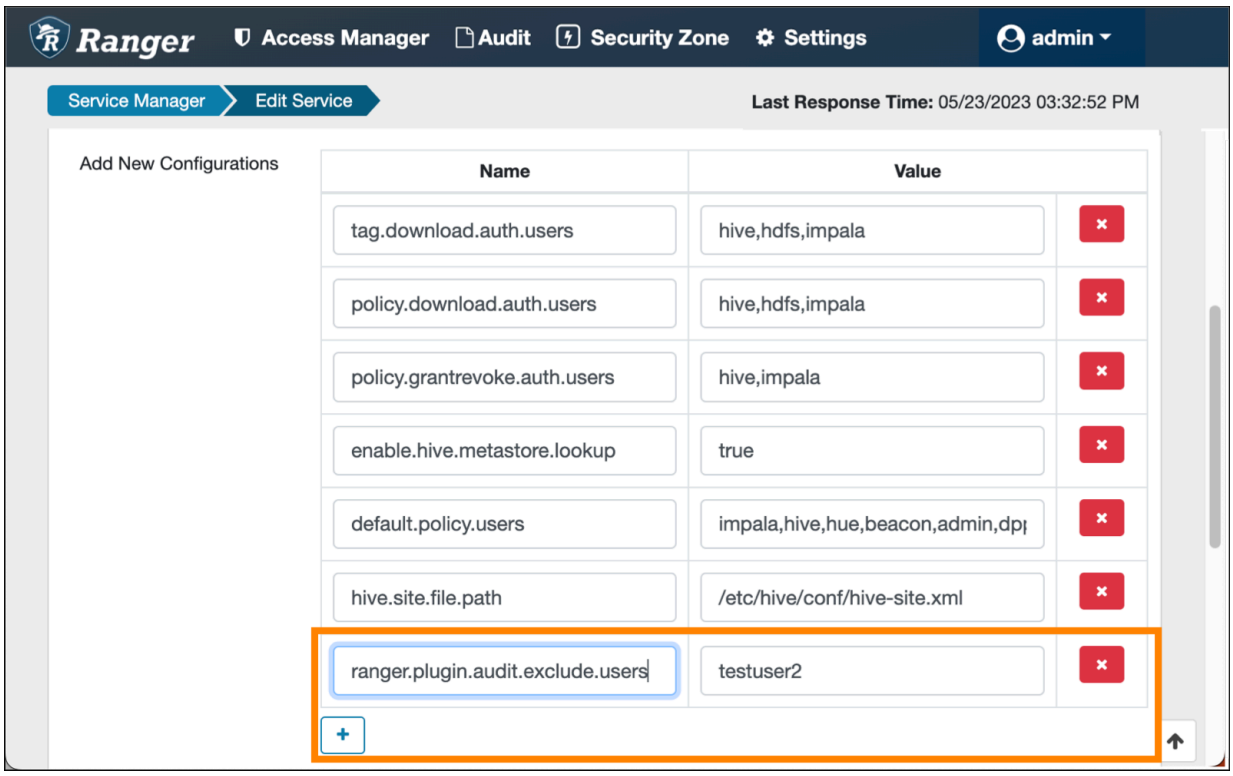

After adding the above configuration; if testuser2 user performs any actions for HadoopSQL service, Audit Access logs will not appear in the Ranger UI, but are still sent to Solr.

Similarly, you can exclude (or blacklist) users belonging to a particular group or role by adding a user-specific or role-specific configuration.

Changing Ranger audit storage location and migrating data

How to change the location of existing and future Ranger audit data collected by Solr from HDFS to a local file system or from a local file system to HDFS.

Before you begin

- Stop Atlas from Cloudera Manager.
- If using Kerberos, set the SOLR_PROCESS_DIR environment variable.

```
# export SOLR_PROCESS_DIR=$(ls -1dtr /var/run/cloudera-scm-agent/process/
*SOLR_SERVER | tail -1)
```

About this task

Starting with Cloudera Runtime version 7.1.4 / 7.2.2, the storage location for ranger audit data collected by Solr changed to local file system from HDFS, as was true for previous versions. The default storage location Ranger audit data storage location for Cloudera Runtime-7.1.4+ and Cloudera Runtime-7.2.2+ installations is local file system. After upgrading from an earlier Cloudera platform version, follow these steps to backup and migrate your Ranger audit data and change the location where Solr stores your future Ranger audit records.

- The default value of the index storage in the local file system is /var/lib/solr-infra. You can configure this, using Cloudera Manager Solr Configuration "Solr Data Directory" .
- The default value of the index storage in HDFS is /solr-infra. You can configure this, using Cloudera Manager Solr Configuration "HDFS Data Directory" .

Procedure

1. Create HDFS Directory to store the collection backups.

   As an HDFS super user, run the following commands to create the backup directory:

   ```
   # hdfs dfs -mkdir /solr-backups
   # hdfs dfs -chown solr:solr /solr-backups
   ```

2. Obtain valid kerberos ticket for Solr user.

   ```
   # kinit -kt solr.keytab solr/$(hostname -f)
   ```

3. Download the configs for the collection.

   ```
   # solrctl instancedir --get ranger_audits /tmp/ranger_audits
   # solrctl instancedir --get atlas_configs /tmp/atlas_configs
   ```

4. Modify the solrconfig.xml for each of the configs for which data needs to be stored in HDFS.

   In /tmp/<config_name>/conf created during Step 3., edit properties in the solrconfig.xml file as follows:

   - When migrating your data storage location from a local file system to HDFS, replace these two lines:

     ```
     <directoryFactory name="DirectoryFactory"
      class="${solr.directoryFactory:solr.NRTCachingDirectoryFactory}">
     <lockType>${solr.lock.type:native}</lockType>
     ```

     with

     ```
     <directoryFactory name="DirectoryFactory"
      class="${solr.directoryFactory:org.apache.solr.core.HdfsDirectoryFactory}">
     <lockType>${solr.lock.type:hdfs}</lockType>
     ```

   - When migrating your data storage location from HDFS to a local file system, replace these two lines:

     ```
     <directoryFactory name="DirectoryFactory"
      class="${solr.directoryFactory:org.apache.solr.core.HdfsDirectoryFactory}">
     <lockType>${solr.lock.type:hdfs}</lockType>
     ```

     with

     ```
     <directoryFactory name="DirectoryFactory"
      class="${solr.directoryFactory:solr.NRTCachingDirectoryFactory}">
     ```

```
<lockType>${solr.lock.type:native}</lockType>
```

5. Backup the Solr collections.

   - When migrating your data storage location from a local file system to HDFS, run:

     ```
     # curl -k --negotiate -u : "https://$(hostname
      -f):8995/solr/admin/collections?action=BACKUP&name=vertex_backup&col
     lection=vertex_index&
     location=hdfs://<Namenode_Hostname>:8020/solr-backups"
     ```

     In the preceding command, the important points are name, collection, and location:

     name

     specifies the name of the backup. It should be unique per collection

     collection

     specifies the collection name for which the backup will be performed

     location

     specifies the HDFS path, where the backup will be stored

     Repeat the curl command for different collections, modifying the parameters as necessary for each collection.

     The expected output would be -

     ```
     "responseHeader":{
      "status":0,
      "QTime":10567},
     "success":{
      "Solr_Server_Hostname:8995_solr":{
        "responseHeader":{
        "status":0,
        "QTime":8959}}}}
     ```

   - When migrating your data storage location from HDFS to a local file system:

     Refer to Back up a Solr collection for specific steps, and make the following adjustments:

     - If TLS is enabled for the Solr service, specify the trust store and password by using the ZKCLI_JVM_FLAGS environment variable before you begin the procedure.

       ```
       # export ZKCLI_JVM_FLAGS="-Djavax.net.ssl.trustStore=/path/to/
       truststore.jks -Djavax.net.ssl.trustStorePassword="
       ```

     - Create Snapshot

       ```
       # solrctl --jaas $SOLR_PROCESS_DIR/jaas.conf collection --create-
       snapshot <snapshot_name> -c <collection_name>
       ```

     - or use the Solr API to take the backup:

       ```
       curl -i -k --negotiate -u : "https://(hostname -f):8995/solr/admin/
       collections?
       action=BACKUP&name=ranger_audits_bkp&collection=ranger_audits&location=/
       path/to/solr-backups"
       ```

     - Export Snapshot

       ```
       # solrctl --jaas $SOLR_PROCESS_DIR/jaas.conf collection
        --export-snapshot <snapshot_name> -c <collection_name> -d
        <destination_directory>
       ```

       Note: The <destination_directory> is a HDFS path. The ownership of this directory should be solr:solr.

6. Update the modified configs in Zookeeper.

```
# solrctl --jaas $SOLR_PROCESS_DIR/jaas.conf instancedir --update
 atlas_configs /tmp/atlas_configs
# solrctl --jaas $SOLR_PROCESS_DIR/jaas.conf instancedir --update
 ranger_audits /tmp/ranger_audits
```

7. Delete the collections from the original location.

   All instances of Solr service should be up, running, and healthy before deleting the collections. Use Cloudera Manager to check for any alerts or warnings for any of the instances. If alerts or warnings exist, fix those before deleting the collection.

```
# solrctl collection --delete edge_index
# solrctl collection --delete vertex_index
# solrctl collection --delete fulltext_index
# solrctl collection --delete ranger_audits
```

8. Verify that the collections are deleted from the original location.

```
# solrctl collection --list
```

   This will give an empty result.

9. Verify that no leftover directories for any of the collections have been deleted.
   - When migrating your data storage location from a local file system to HDFS:

     ```
     # cd /var/lib/solr-infra
     ```

     Get the value of "Solr Data Directory, using Cloudera Manager Solr Configuration .

     ```
     # ls -ltr
     ```

   - When migrating your data storage location from HDFS to a local file system, replace these two lines:

     ```
     # hdfs dfs -ls /solr/<collection_name>
     ```

     Note: If any directory name which starts with the collection name deleted in Step 7. exists, delete/move the directory to another path.

10. Restore the collection from backup to the new location.

    Refer to Restore a Solr collection, for more specific steps.

```
# curl -k --negotiate -u : "https://$(hostname
 -f):8995/solr/admin/collections?
action=RESTORE&name=<Name_of_backup>&location=hdfs:/
<<Namenode_Hostname>:8020/solr-backups&collection=<Collection_Name>"
```

```
# solrctl collection --restore ranger_audits
 -l hdfs://<Namenode_Hostname>:8020/solr-backups
 -b ranger_backup -i ranger1
```

    The request id must be unique for each restore operation, as well as for each retry.

    To check the status of restore operation:

```
# solrctl collection --request-status <requestId>
```

    Note: If the Atlas Collections (vertex_index, fulltext_index and edge_index) restore operations fail, restart the solr service and rerun the restore command. Now, the restart operations should complete successfully.

11. Verify the Atlas and Ranger functionality.

Verify that both Atlas and Ranger audits functions properly, and that you can see the latest audits in Ranger Web UI and latest lineage in Atlas.

- To verify Atlas audits, create a test table in Hive, and then query the collections to see if you are able to view the data.
- You can also query the collections every 20-30 seconds (depending on how other services utilize Atlas/Ranger), and verify if the "numDocs" value increases at every query.

```
# curl -k --negotiate -u : "https://$(hostname -f):8995/solr/edge_index/
select?q=*%3A*&wt=json&ident=true&rows=0"
# curl -k --negotiate -u : "https://$(hostname -f):8995/solr/vertex_index/
select?q=*%3A*&wt=json&ident=true&rows=0"
# curl -k --negotiate -u : "https://$(hostname -f):8995/solr/
fulltext_index/select?q=*%3A*&wt=json&ident=true&rows=0"
# curl -k --negotiate -u : "https://$(hostname -f):8995/solr/
ranger_audits/select?q=*%3A*&wt=json&ident=true&rows=0"
```

Configuring Ranger audits to show actual client IP address

How to forward the actual client IP address to audit logs generated from a Ranger plugin.

About this task

Ranger audit logs record the IP address through which Ranger policies grant/authorize access. When Ranger is set up behind a Knox proxy server, the proxy server IP address appears in the audit logs generated for each Ranger plugin. You can configure each plugin to forward the actual client IP address on which that service runs, so that the audit logs for that service more specifically reflect access/authorization activity. You must configure each plugin individually. This topic uses the Hive (Hadoop SQL) service as an example.

Procedure

1. From Cloudera Manager choose <service_name> Configuration .
2. In <service_name> Configuration Search , type ranger-plugin, then press Return.
3. In Ranger Plugin Use X-Forwarded for IP Address, check the box.
4. In Ranger Plugin Trusted Proxy IP Address, type the IP address of the Knox proxy server host.

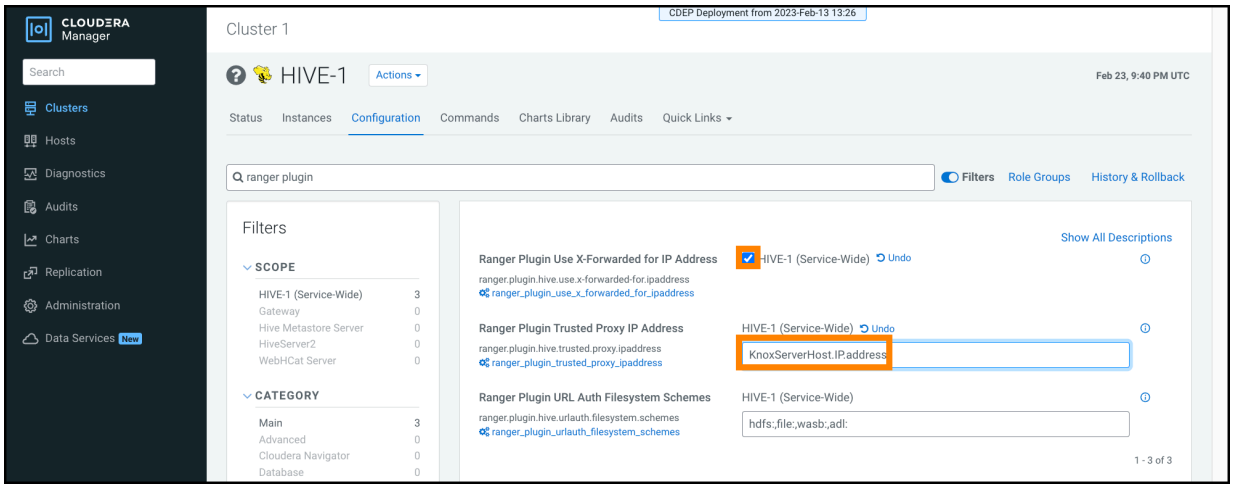

Results

Hive audit logs will now show the IP address of the host on which Hive service runs.

Cloudera Runtime 7.3.1

Configuring Apache Ranger Authentication with UNIX, LDAP, or AD

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring Ranger Authentication with UNIX, LDAP, AD, or PAM..... 4
Configure Ranger authentication for UNIX.....4
Configure Ranger authentication for AD.....6
Configure Ranger authentication for LDAP.....8
Configure Ranger authentication for PAM.....10

Ranger AD Integration..... 11
Ranger UI authentication.....16
Ranger UI authorization..... 19

Configuring Ranger Authentication with UNIX, LDAP, AD, or PAM

This section describes how to configure the authentication method that determines who is allowed to log in to the Ranger web UI. The options are local UNIX, LDAP, AD, or PAM.

Note: In Cloudera Public Cloud, identity management is provided by FreeIPA, and configured using the Cloudera Management Console. Therefore for Cloudera Public Cloud you should leave the Admin Authentication Method set to the UNIX authentication settings. For more information on FreeIPA, see Managing FreeIPA in the Identify Management documentation.

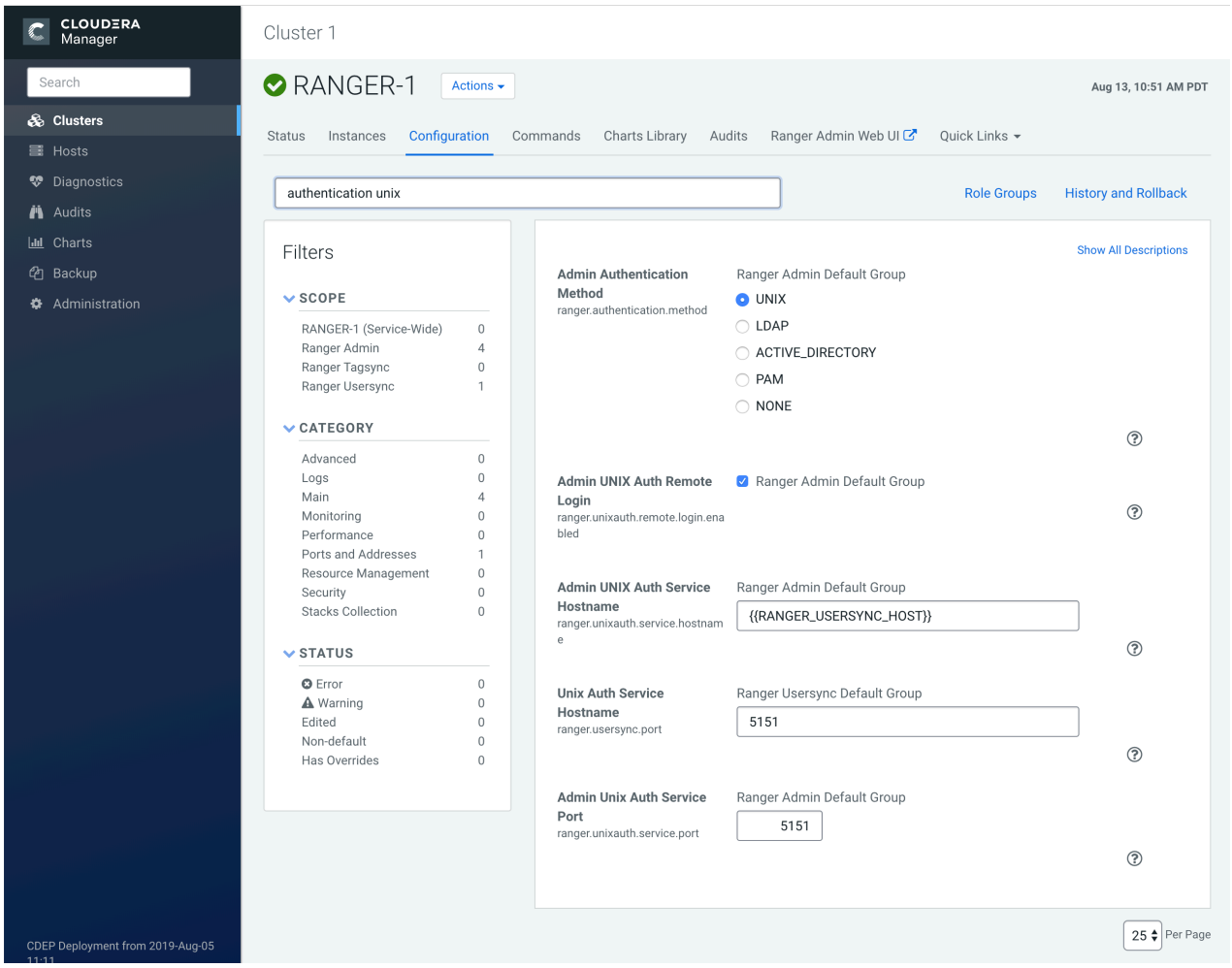

Related Information

Cloudera Management Console

Cloudera Management Console: Managing user access and authorization

Managing FreeIPA

Configure Ranger authentication for UNIX

How to configure Ranger to use UNIX for user authentication.

About this task

Note: In Cloudera Public Cloud, identity management is provided by FreeIPA, and configured using the Cloudera Management Console. Therefore for Cloudera Public Cloud you should leave the Admin Authentication Method set to the UNIX authentication settings. For more information on FreeIPA, see Managing FreeIPA in the Identify Management documentation.

Procedure

1. In Cloudera Manager, select Ranger, then click the Configuration tab.
2. To display the UNIX authentication settings, type "authentication unix" in the Search box.

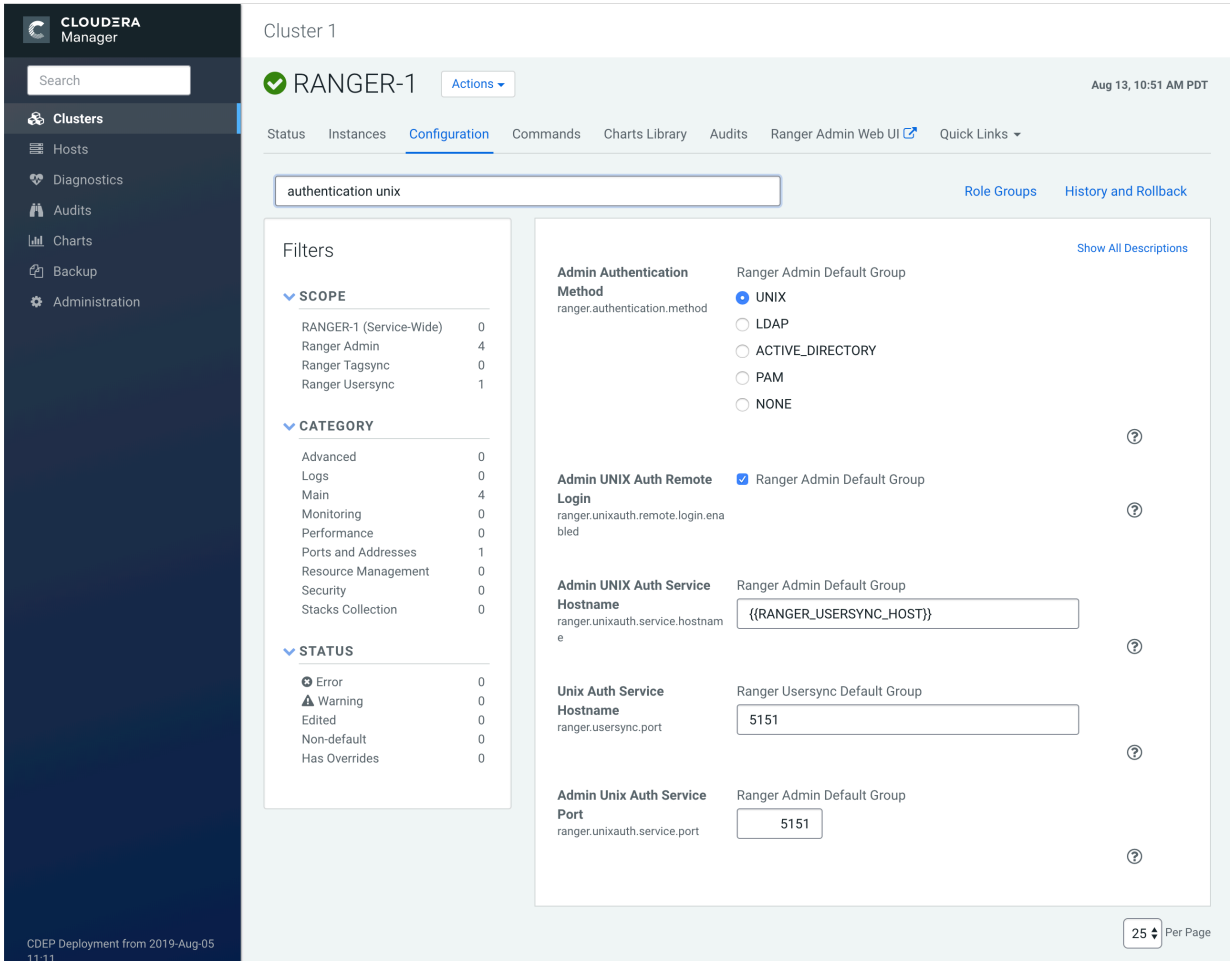

3. Configure the following settings for UNIX authentication, then click Save Changes.

Table 1: UNIX Authentication Settings

Configuration Property	Description	Default Value	Example Value	Requi
Admin Authentication Method	The Ranger authentication method.	UNIX	UNIX	Yes, t auther
Allow remote Login	Flag to enable/disable remote login. Only used if the Authentication method is UNIX.	TRUE	TRUE	No.

Configuration Property	Description	Default Value	Example Value	Requi
ranger.unixauth.service.hostname	The FQDN of the host where the UNIX authentication service is running. Only used if the Authentication method is UNIX. {{RANGER_USERSYNC_HOST}} is a placeholder value that is replaced with the host where Ranger Usersync is installed in the cluster.	localhost	myunixhost.domain.com	Yes, i selecte
ranger.unixauth.service.port	The port number where the ranger-usersync module is running the UNIX Authentication Service.	5151	5151	Yes, i selecte

Related Information

Cloudera Management Console

Configure Ranger authentication for AD

How to configure Ranger to use Active Directory (AD) for user authentication.

About this task

Note: In Cloudera Public Cloud, identity management is provided by FreeIPA, and configured using the Cloudera Management Console. Therefore for Cloudera Public Cloud you should leave the Admin Authentication Method set to the UNIX authentication settings. For more information on FreeIPA, see Managing FreeIPA in the Identify Management documentation.

Procedure

1. Select Cloudera Manager Ranger Configuration , type authentication in Search.
   Ranger authentication property settings display. You may need to scroll down to see the AD settings.

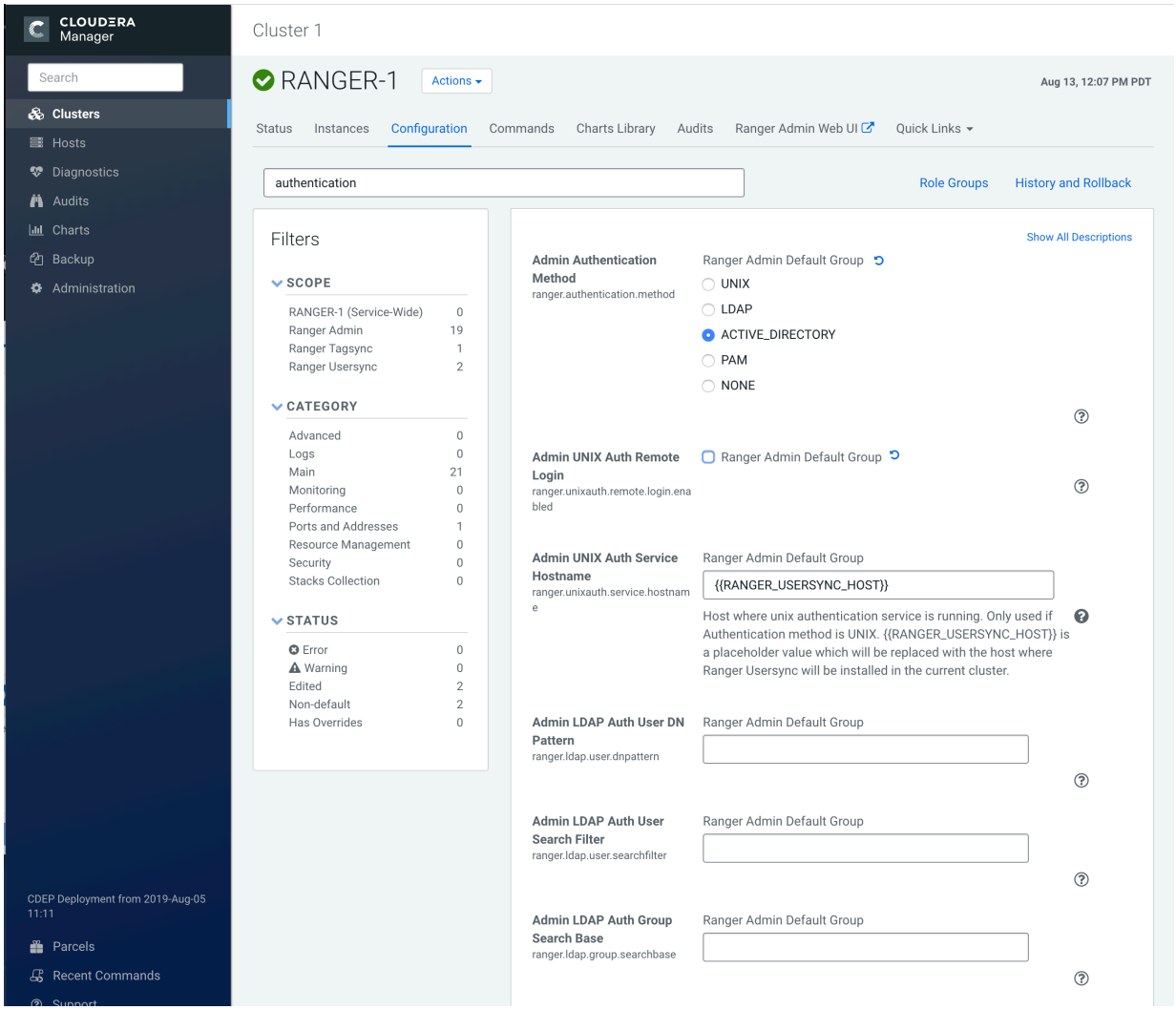

2. Configure the following settings for AD authentication, then click Save Changes.

Property	Description	Default value	Sample values
Admin Authentication Method	The Ranger authentication method.	UNIX	ACTIVE_DIRECTORY
Admin AD Auth Base DN ranger.ldap.ad.base.dn	The Distinguished Name (DN) of the starting point for directory server searches.	N/A	dc=example,dc=com
Admin AD Auth Bind DN ranger.ldap.ad.bind.dn	The full Distinguished Name (DN), including Common Name (CN) of an LDAP user account that has privileges to search for users.	N/A	cn=adadmin,cn=Users,dc=example,dc=com
Admin AD Auth Bind Password ranger.ldap.ad.bind.password	Password for the bind.dn.	N/A	Secret123!
Admin AD Auth Domain Name ranger.ldap.ad.domain	The domain name of the AD Authentication service.	N/A	example.com

Property	Description	Default value	Sample values
Admin AD Auth Referral ranger.ldap.ad.referral*	See below.	ignore	follow | ignore | throw
Admin AD Auth URL ranger.ldap.ad.url	The AD server URL, for example: ldap://<AD-Servername>:Port	N/A	ldap://<AD-Servername>:Port
Admin AD Auth User Search Filter ranger.ldap.ad.user.searchfilter	AD user search filter.	N/A	

* There are three possible values for ranger.ldap.ad.referral:

- follow
- throw
- ignore

The recommended setting is: follow.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

When ranger.ldap.ad.referral is set to follow:

The AD service provider processes all of the normal entries first, and then follows the continuation references.

When ranger.ldap.ad.referral is set to throw:

All of the normal entries are returned in the enumeration first, before theReferralException is thrown.

By contrast, a referral error response is processed immediately when this property is set to follow or throw.

When ranger.ldap.ad.referral is set to ignore:

The server should return referral entries as ordinary entries (or plain text). This might return partial results for the search. In the case of AD, a PartialResultException is returned when referrals are encountered while search results are processed.

Related Information

Cloudera Management Console

Configure Ranger authentication for LDAP

How to configure Ranger to use LDAP for user authentication.

About this task

Note: In Cloudera Public Cloud, identity management is provided by FreeIPA, and configured using the Cloudera Management Console. Therefore for Cloudera Public Cloud you should leave the Admin Authentication Method set to the UNIX authentication settings. For more information on FreeIPA, see Managing FreeIPA in the Identify Management documentation.

Procedure

1. In Cloudera Manager, select Ranger, then click the Configuration tab.

2. To display the authentication settings, type "authentication" in the Search box. You may need to scroll down to see all of the LDAP settings.

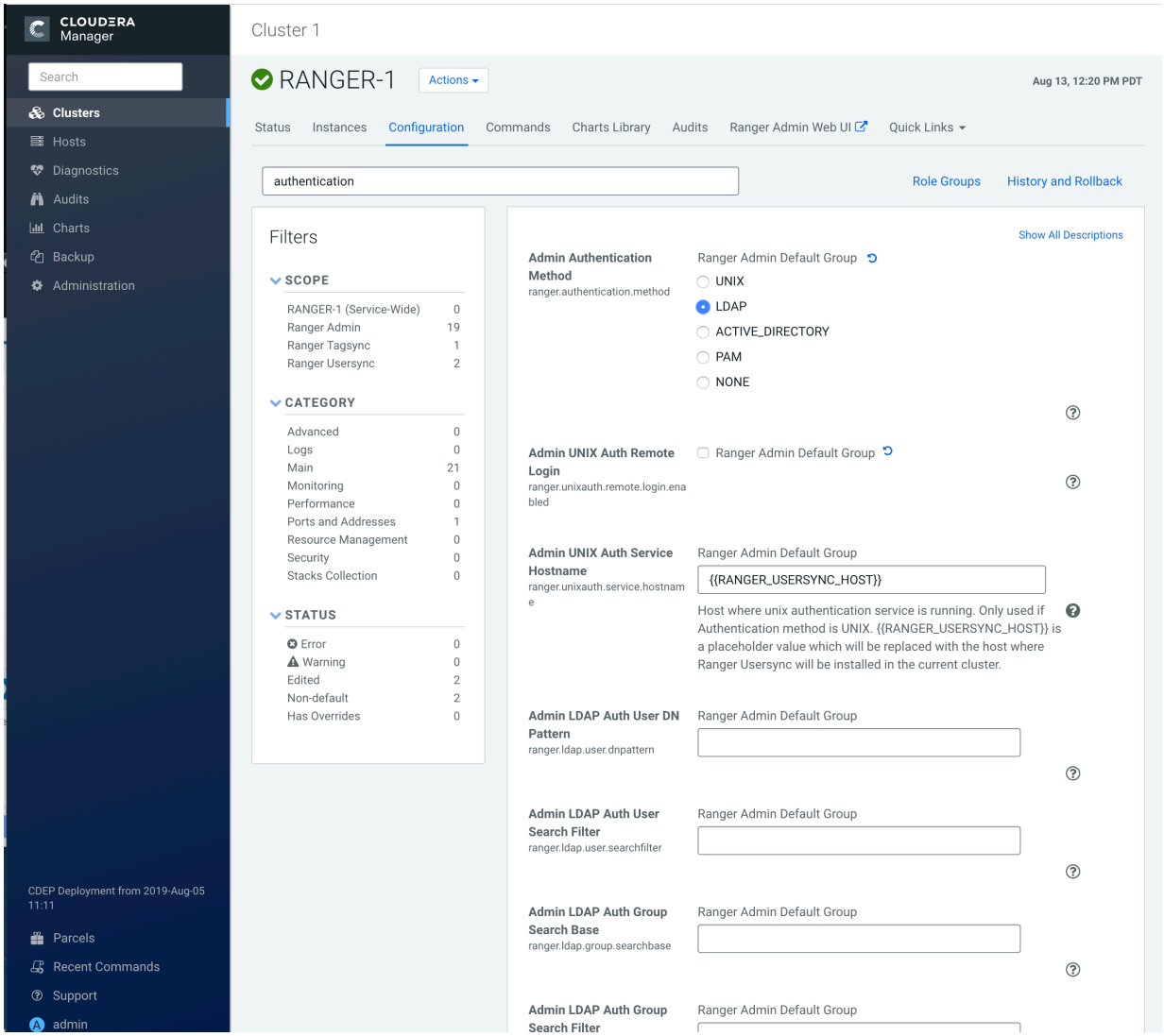

3. Configure the following settings for LDAP authentication, then click Save Changes.

Property	Required ?	Description	Default value	Sample values
Admin Authentication Method	Required	The Ranger authentication method.	UNIX	LDAP
Admin LDAP Auth Group Search Base ranger.ldap.group.searchbase	Optional	The LDAP group search base.	N/A	(|(CN=Hdp_users)(CN=Hdp_admins))
Admin LDAP Auth Group Search Filter ranger.ldap.group.searchfilter	Optional	The LDAP group search filter.	N/A	
Admin LDAP Auth URL ranger.ldap.url	Required	The LDAP server URL	N/A	ldap://localhost:389 or ldaps://localhost:636

Property	Required ?	Description	Default value	Sample values
Admin LDAP Auth Bind User ranger.ldap.bind.dn	Required	Full distinguished name (DN), including common name (CN), of an LDAP user account that has privileges to search for users. This user is used for searching the users. This could be a read-only LDAP user.	N/A	cn=admin,dc=example,dc=com
Admin LDAP Auth Bind User Password ranger.ldap.bind.password	Required	Password for the account that can search for users.	N/A	Secret123!
Admin LDAP Auth User Search Filter ranger.ldap.user.searchfilter	Required	The LDAP user search filter.	N/A	
Admin LDAP Auth Base DN ranger.ldap.base.dn	Required	The Distinguished Name (DN) of the starting point for directory server searches.	N/A	dc=example,dc=com
Admin LDAP Auth Group Role Attribute ranger.ldap.group.roleattribute	Optional	The LDAP group role attribute.	N/A	cn
Admin LDAP Auth Referral ranger.ldap.referral*	Required	See below.	ignore	follow | ignore | throw
Admin LDAP Auth User DN Pattern ranger.ldap.user.dnpattern	Required	The LDAP user DN.	N/A	uid={0},ou=users,dc=xasecure,dc=net

* There are three possible values for ranger.ldap.ad.referral: follow, throw, and ignore. The recommended setting is follow.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to follow, the AD service provider processes all of the normal entries first, and then follows the continuation references.
- When this property is set to throw, all of the normal entries are returned in the enumeration first, before the ReferralException is thrown. By contrast, a "referral" error response is processed immediately when this property is set to follow or throw.
- When this property is set to ignore, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search. In the case of AD, a PartialResultException is returned when referrals are encountered while search results are processed.

Related Information

Cloudera Management Console

Configure Ranger authentication for PAM

How to configure Ranger to use PAM for user authentication.

About this task

Note: In Cloudera Public Cloud, identity management is provided by FreeIPA, and configured using the Cloudera Management Console. Therefore for Cloudera Public Cloud you should leave the Admin Authentication Method set to the UNIX authentication settings. For more information on FreeIPA, see Managing FreeIPA in the Identify Management documentation.

Procedure

1. In Cloudera Manager, select Ranger, then click the Configuration tab.
2. Under Admin Authentication Method, select PAM, then click Save Changes.

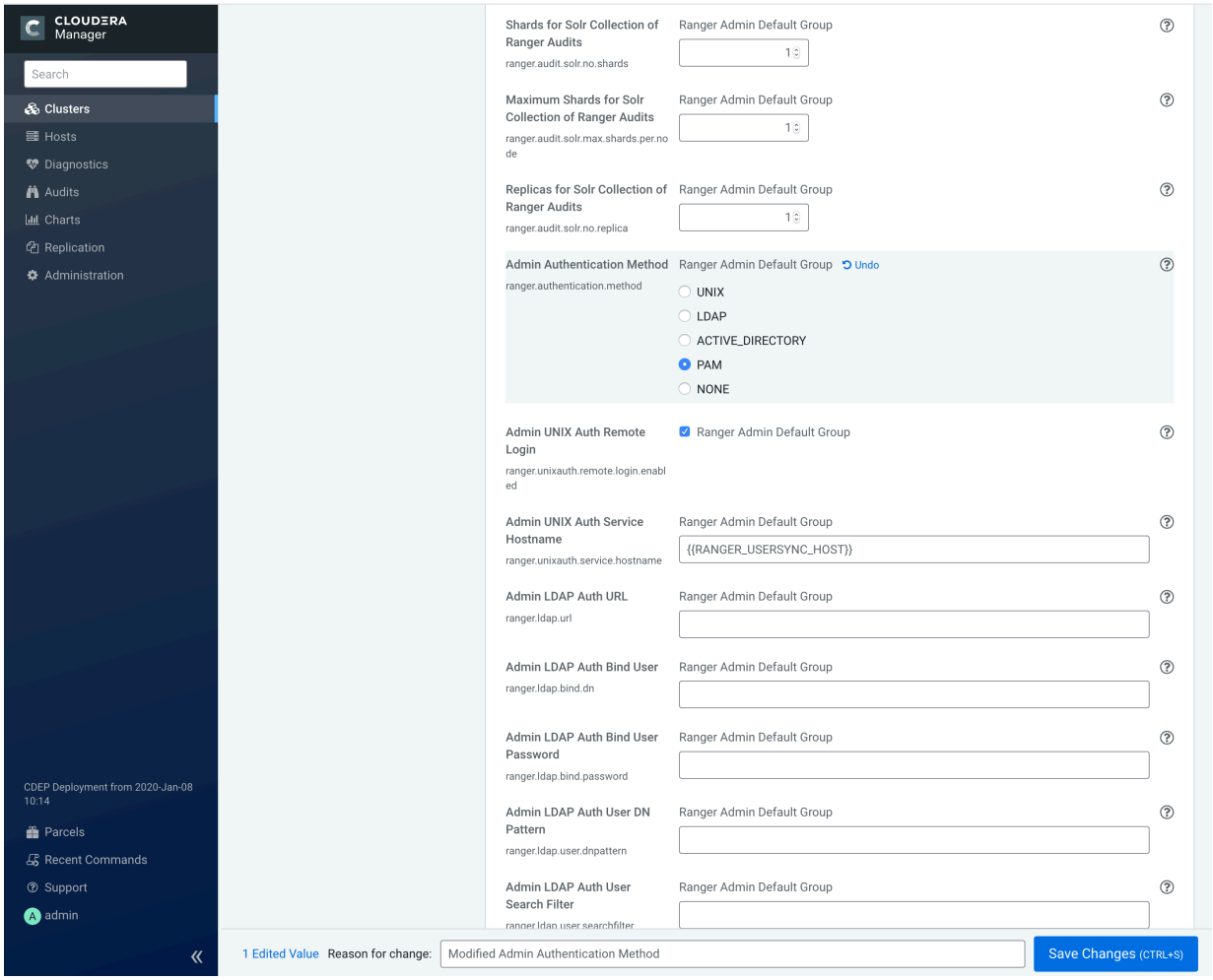

3. Allow the Ranger user to read the /etc/shadow file:

```
groupadd shadow
usermod -a -G shadow ranger
chgrp shadow /etc/shadow
chmod g+r /etc/shadow
```

4. Select Actions > Restart to restart Ranger.

Ranger AD Integration

A conceptual overview of Ranger-AD integration architecture.

Ranger AD Integration: Architecture Overview

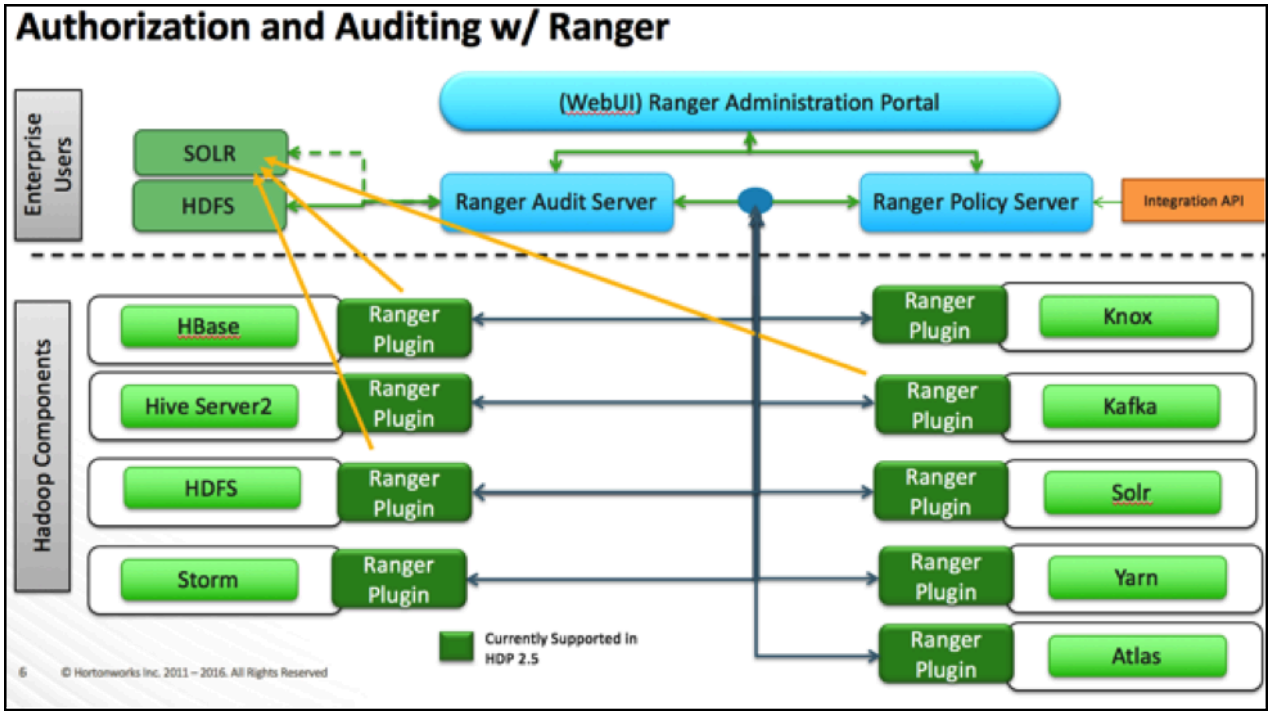

When a Ranger plugin for a component (such as HBase or HDFS) is activated, Ranger is in full control of any access. There is two-way communication between the Ranger plugin and the Ranger (Admin) Policy Server (RPS):

1. Plugins to RPS: Ranger plugins regularly call the RPS to see if new policies were defined in the Ranger Administration Portal (RAP). Generally it takes approximately 30 seconds for a policy to be updated.
2. RPS to components: The RPS queries the component for meta objects that live on the component to base policies upon (this provides the autocomplete and drop-down list when defining policies).

The first communication channel (Plugin to RPS) is essential for the plugin to function, whereas the second (RPS to components) is optional. It would still be possible to define and enforce policies without the second channel, but you would not have autocomplete during policy definition.

Configuration details on both communication channels are configured in both Cloudera Manager and in the Ranger Administration Portal.

Example for HDFS plugin on a kerberized cluster:

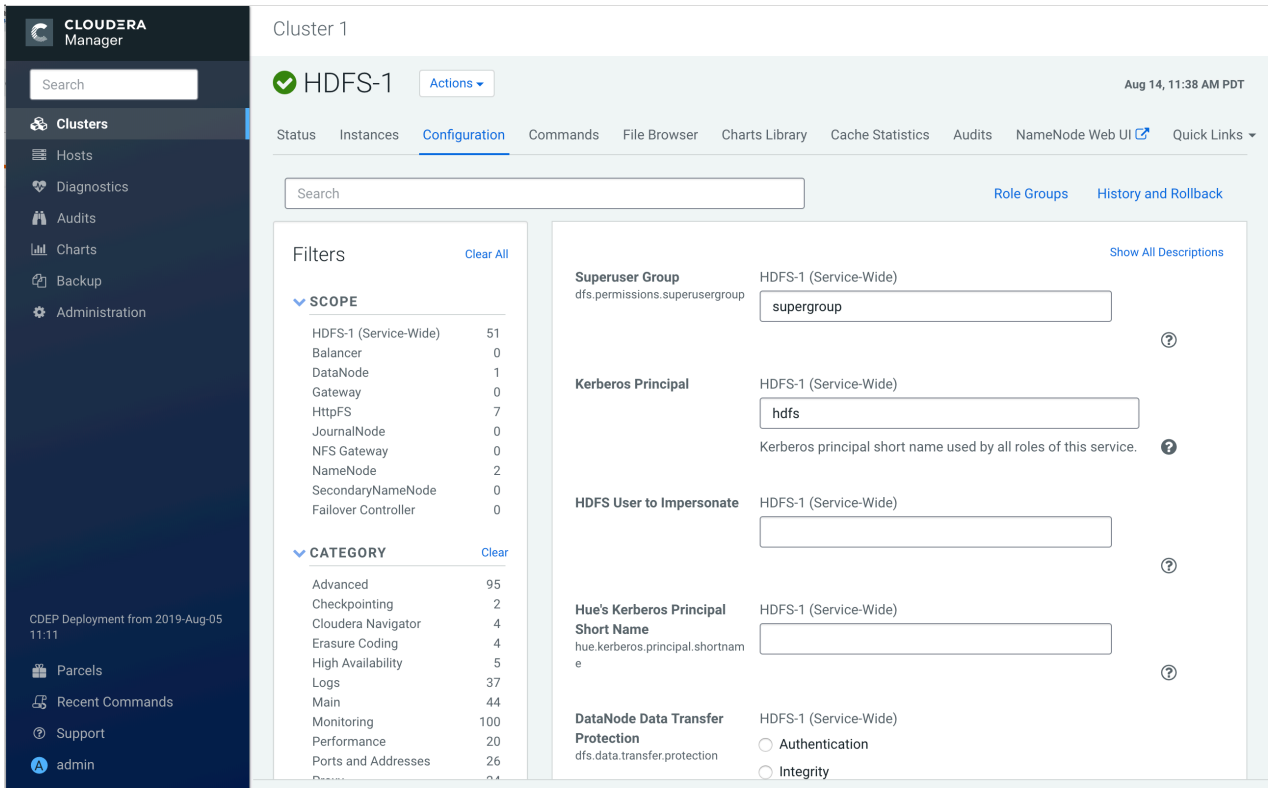

The Kerberos principal short name for the HDFS service,"hdfs", is the one that is involved the second communication channel (RPS to components) for getting metadata from HDFS (such as HDFS folders) across. The settings on the HDFS configuration must match those set in Ranger (by selecting Service Manager Resource Based Policies , then selecting Edit for the HDFS service:

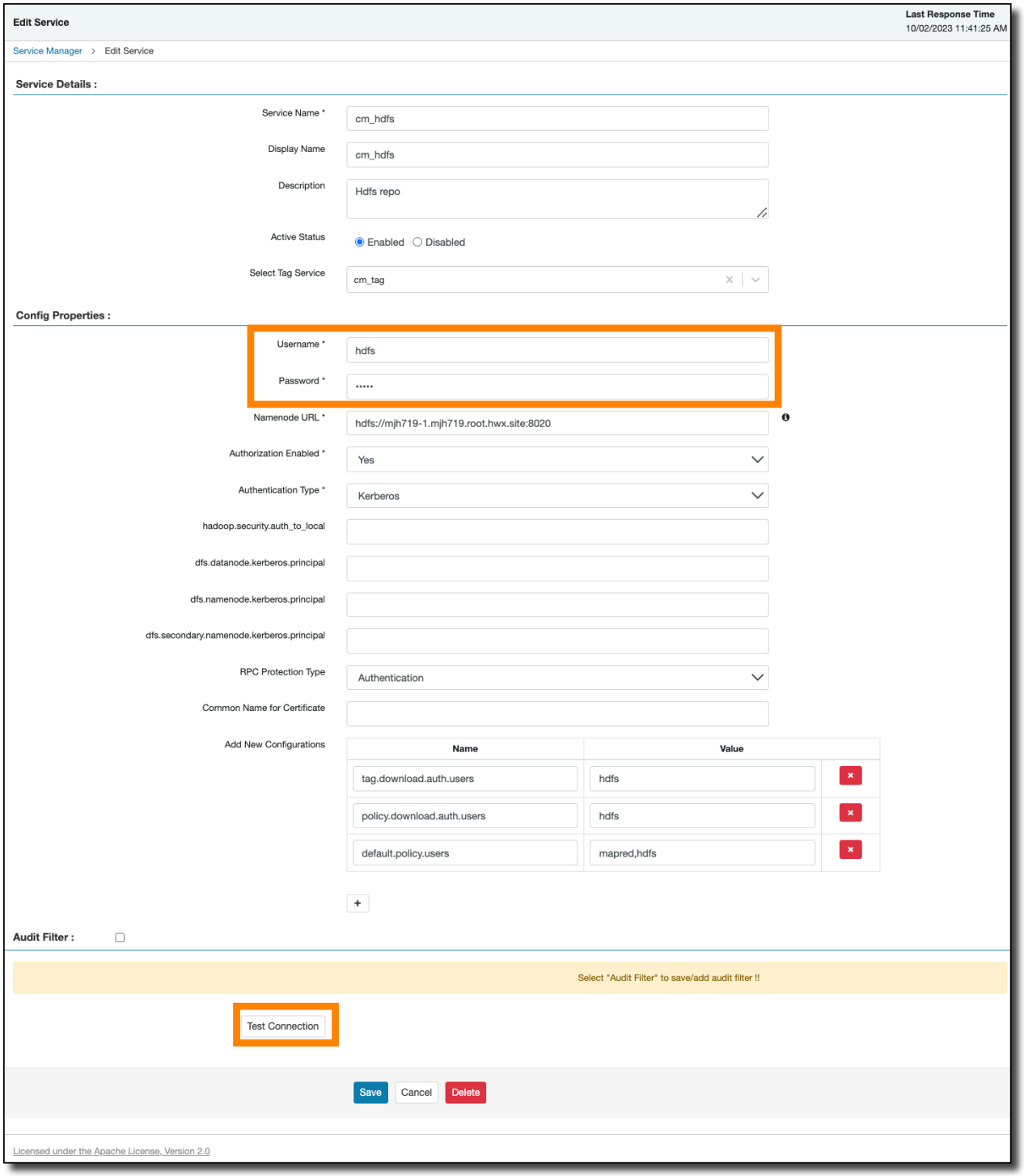

To verify the second communication channel (RPS to components) click Test Connection for the applicable service (as shown above for the HDFS service). A confirmation message appears if the connection works successfully.

To verify if the paramount first communication channel (Plugins to RPS) works, select Service Manager Audits Plugins in Ranger Admin Web UI:

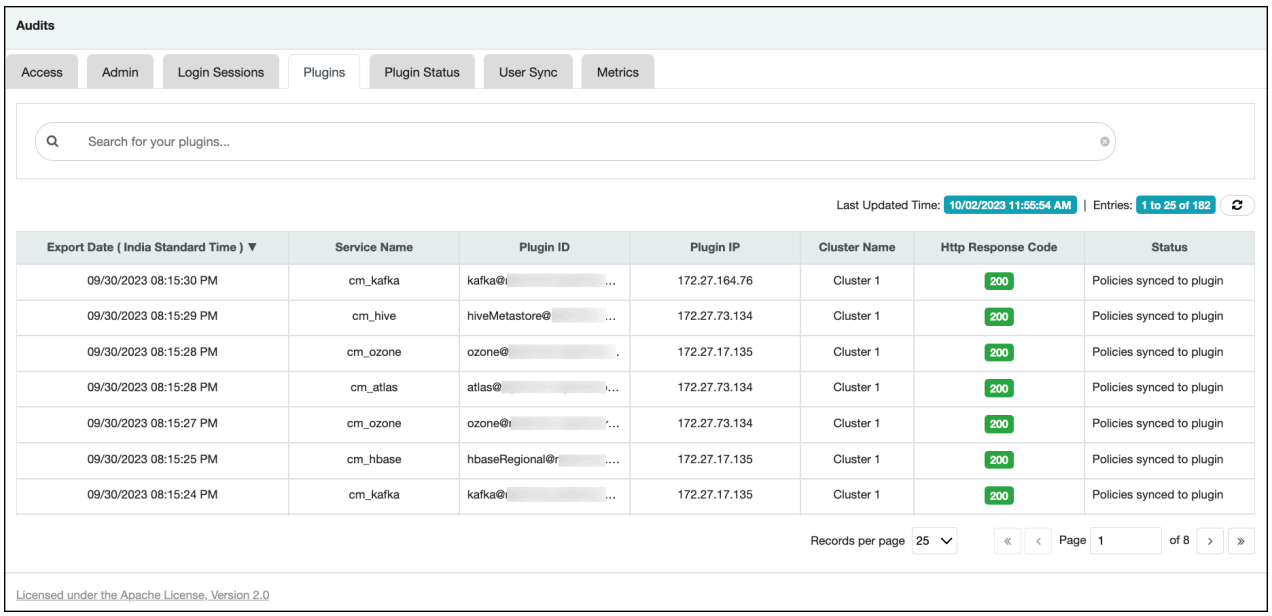

Ranger AD Integration: Ranger Audit

Ranger plugins furthermore send their audit event (whether access was granted or not and based on which policy) directly to the configured sink for audits, which can be HDFS, Solr or both. This is indicated by the yellow arrows in the architectural graph.

The audit access tab on the RAP (Audits Access) is only populated if Solr is used as the sink.

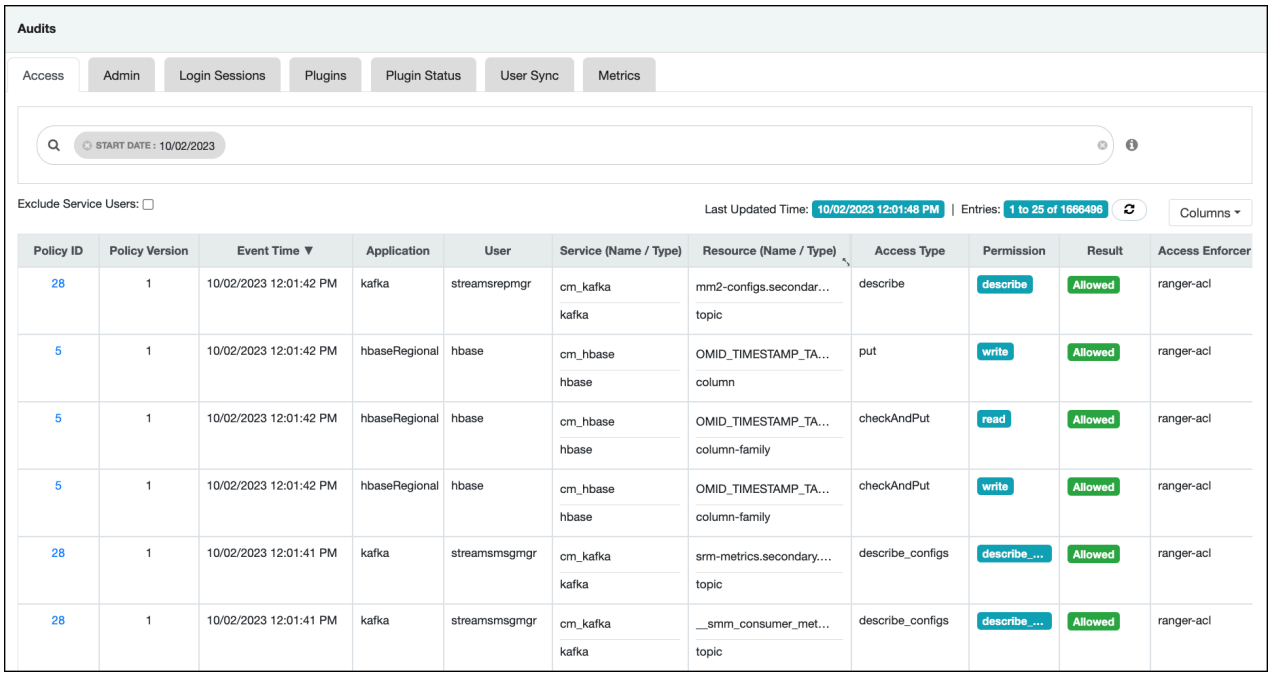

This screen points out an important Ranger feature. When the plugin is enabled AND no specific policy is in place for access to some object, the plugin will fall back to enforcing the standard component-level Access Control Lists (ACLs). For HDFS that would be the user : rwx / group : rwx / other : rwx ACLs on folders and files.

Once this defaulting to component ACLs happens, the audit events list a " - " in the Policy ID column instead of a policy number. If a Ranger policy was in control of allowing/denying access, the policy number is shown.

Ranger AD Integration: Overview

Rangers AD Integration has 2 levels:

1. Ranger UI authentication (which users can log in to Ranger itself).
2. Ranger user/group sync (which users/groups to define policies for)

Ranger UI authentication

Reference information on Ranger UI authentication, when configuring Ranger AD integration.

This is an extra AD level filter option on top of Kerberos authentication that maps to:

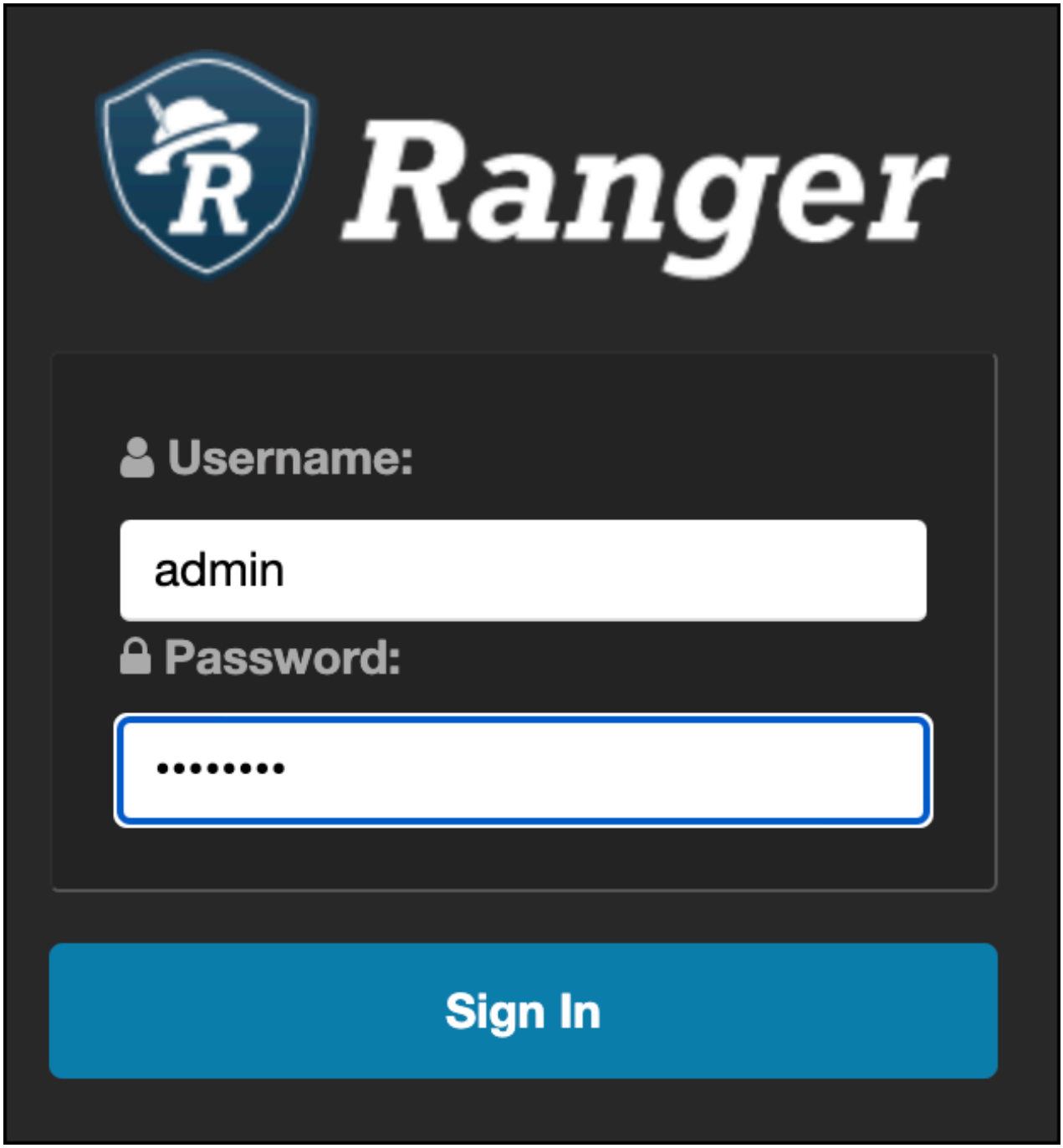

For AD there are two options for defining who can access the Ranger UI: LDAP or ACTIVE_DIRECTORY. There is not a huge amount of difference between them, but they are separate sets of properties.

ACTIVE_DIRECTORY

In Cloudera Manager, select Ranger, then click the Configuration tab.To display the authentication settings, type "authentication" in the Search box. You may need to scroll down to see the AD settings.

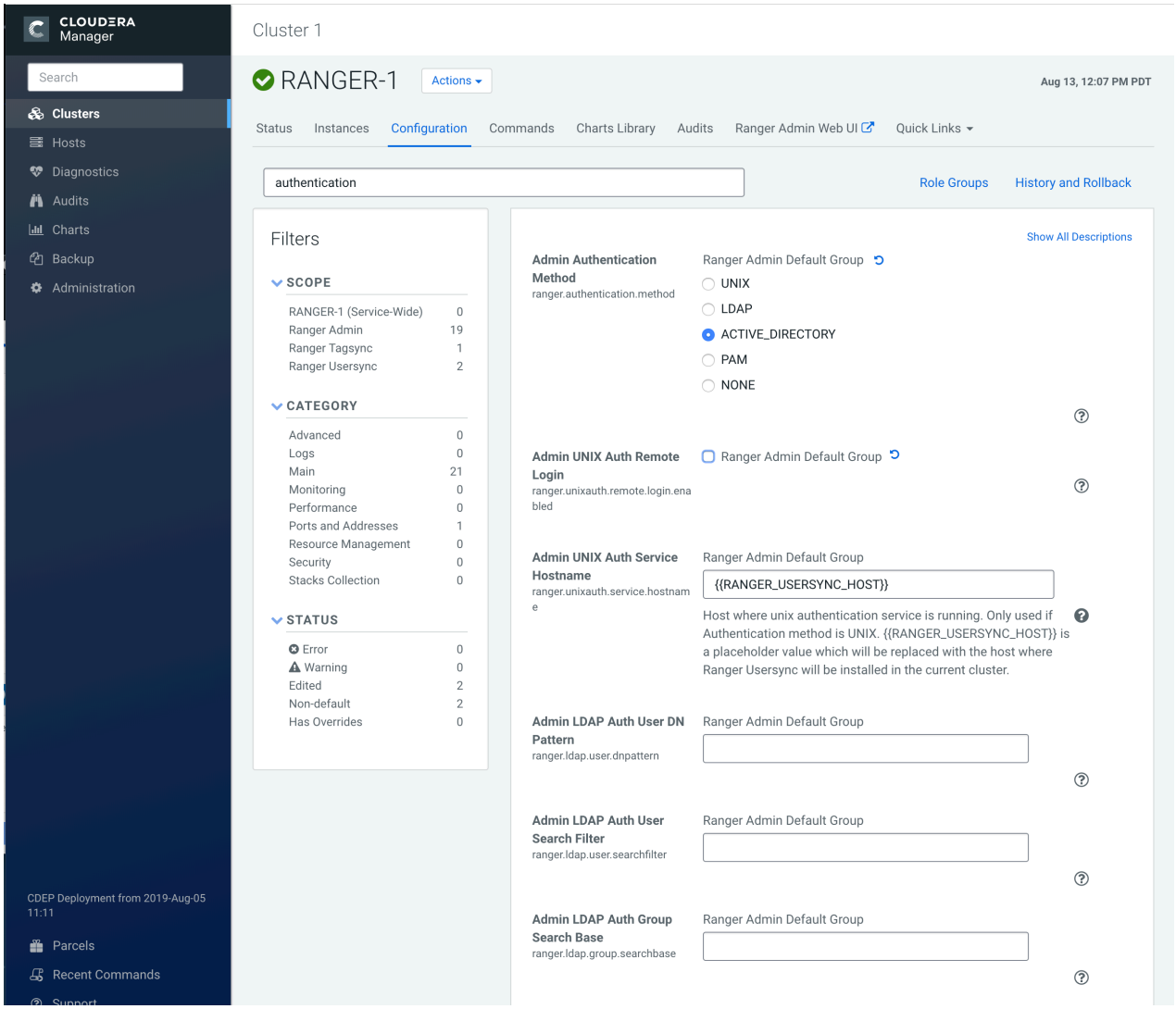

The ranger.ldap.ad.base.dn property determines the base of any search, so users not on this OU tree path can not be authenticated.

The ranger.ldap.ad.user.searchfilter poperty is a dynamic filter that maps the user name in the Ranger web UI login screen to sAMAccountName. For example, the AD sAMAccountName property has example values like k.reshi and d.alora so make sure to enter a matching value for ‘Username’ in the logon dialogue.

LDAP

The LDAP properties allow for more fine tuning.

In Cloudera Manager, select Ranger, then click the Configuration tab.To display the authentication settings, type "authentication" in the Search box. You may need to scroll down to see all of the LDAP settings.

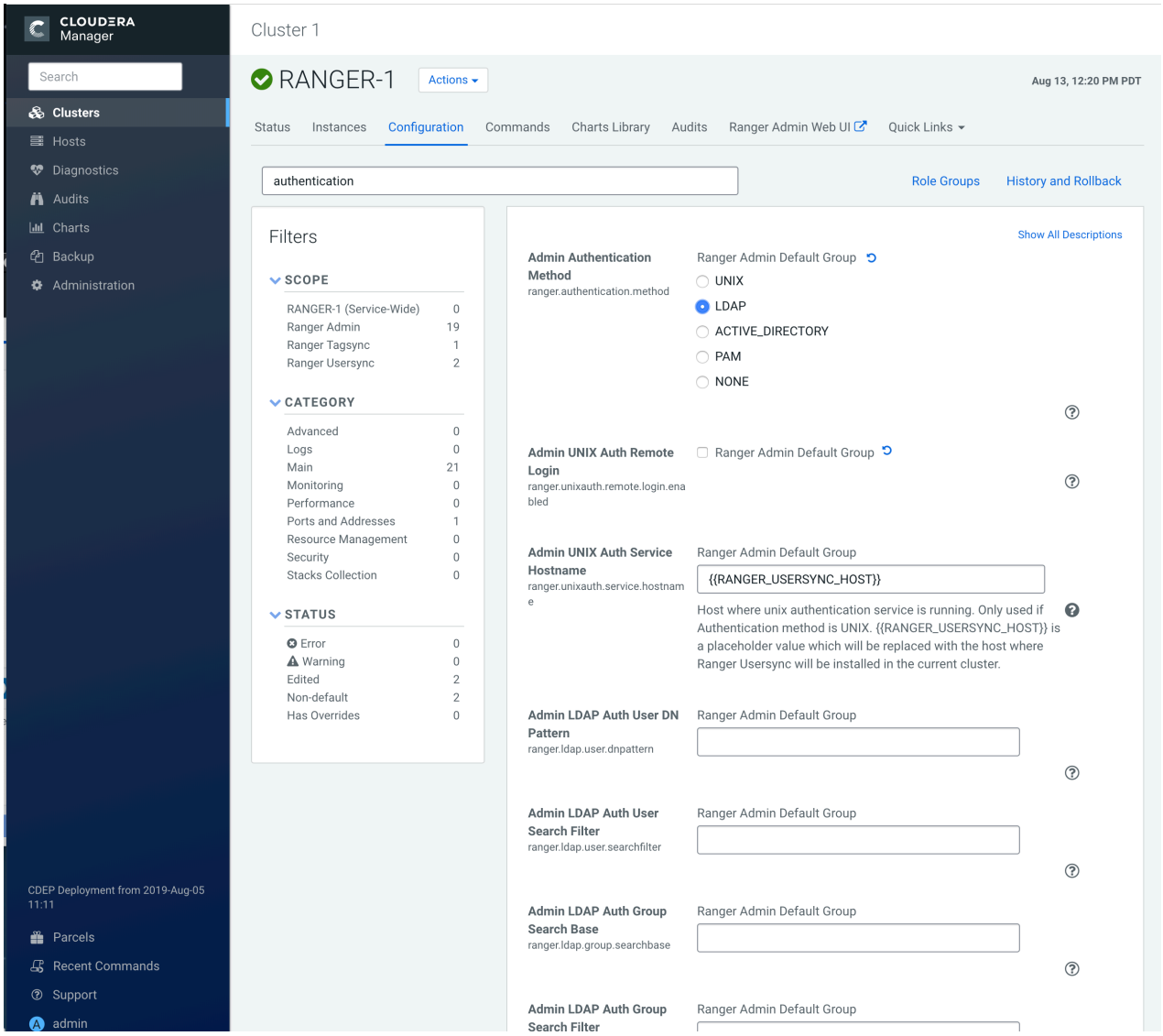

There is one catch: the ranger.ldap.user.dnpattern is evaluated first. Consider the following example value:

CN={0},OU=London,OU=Company,OU=User Accounts,OU=CorpUsers,DC=field,DC=hortonworks,DC=com

This would work, but has two side effects:

- Users would have to log on with their ‘long username’ (like ‘Kvothe Reshi / Denna Alora'), which would also mean that policies would have to be updated using that long name instead of the k.reshi short name variant.
- Traversing AD by DN patterns does not allow for applying group filters at all. In the syntax above, only users directly in OU=London would be able to log on.

This adverse behavior can be avoided by intentionally putting a DN pattern (DC=intentionally,DC=wrong) in the ranger.ldap.user.dnpattern property, AND a valid filter in User Search Filter:

(&(objectclass=user)(memberOf=CN=Hdp_admins,OU=Company,OU=User Accounts,OU=CorpUsers,DC=field,DC=hortonworks,DC=com)(sAMAccountName={0}))

This works because the filter is only applied after the DN pattern query on AD does not return anything. If it does, the User Search Filter is not applied.

Ranger has a very simple approach to the internal user list that is kept in a relational schema. This list contains all users that were synced with AD ever, and all those users can potentially log in to the Ranger UI. But only Admin users can really do any policy-related things in the Ranger UI (see next section).

Be aware that all of this is only about authentication to Ranger. Someone from the ‘Hdp_admins’ group would still not have a Ranger admin role.

Related Information

Configure Ranger authentication for LDAP

Ranger UI authorization

Reference information on Ranger UI authorization, when configuring Ranger AD integration.

To configure the users, groups, and roles that can access the Ranger portal or its services, select Service Manager Settings Users .

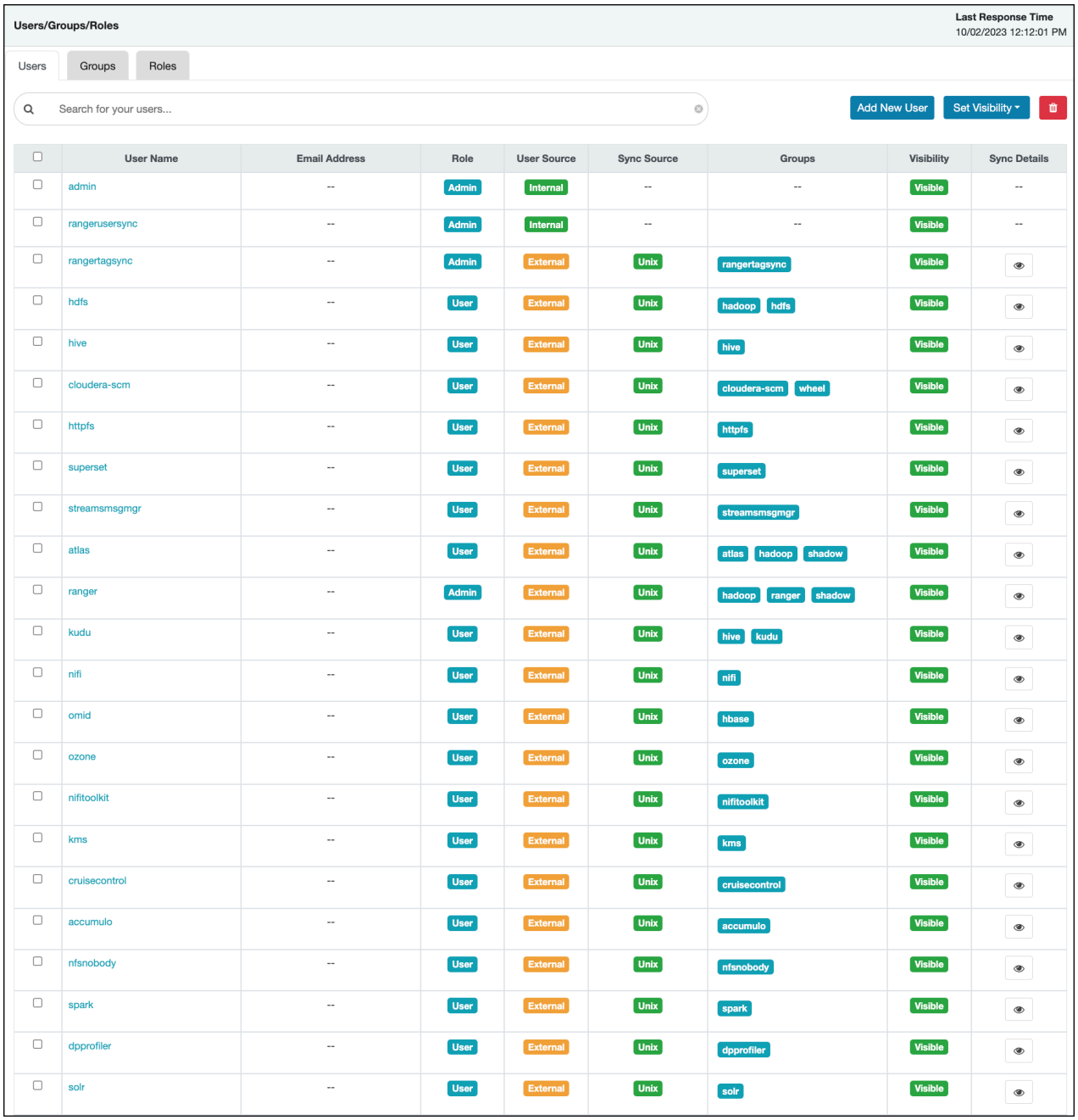

User Name	Email Address	Role	User Source	Sync Source	Groups	Visibility	Sync Details
admin	--	Admin	Internal	--	--	Visible	--
rangerusersync	--	Admin	Internal	--	--	Visible	--
rangertagsync	--	Admin	External	Unix	rangertagsync	Visible	
hdfs	--	User	External	Unix	hadoop hdfs	Visible	
hive	--	User	External	Unix	hive	Visible	
cloudera-scm	--	User	External	Unix	cloudera-scm wheel	Visible	
httpfs	--	User	External	Unix	httpfs	Visible	
superset	--	User	External	Unix	superset	Visible	
streamsmsgmgr	--	User	External	Unix	streamsmsgmgr	Visible	
atlas	--	User	External	Unix	atlas hadoop shadow	Visible	
ranger	--	Admin	External	Unix	hadoop ranger shadow	Visible	
kudu	--	User	External	Unix	hive kudu	Visible	
nifi	--	User	External	Unix	nifi	Visible	
omid	--	User	External	Unix	hbase	Visible	
ozone	--	User	External	Unix	ozone	Visible	
nifitoolkit	--	User	External	Unix	nifitoolkit	Visible	
kms	--	User	External	Unix	kms	Visible	
cruisecontrol	--	User	External	Unix	cruisecontrol	Visible	
accumulo	--	User	External	Unix	accumulo	Visible	
nfsnobody	--	User	External	Unix	nfsnobody	Visible	
spark	--	User	External	Unix	spark	Visible	
dpprofiler	--	User	External	Unix	dpprofiler	Visible	
solr	--	User	External	Unix	solr	Visible	

A user can be a User, Admin, or Auditor:

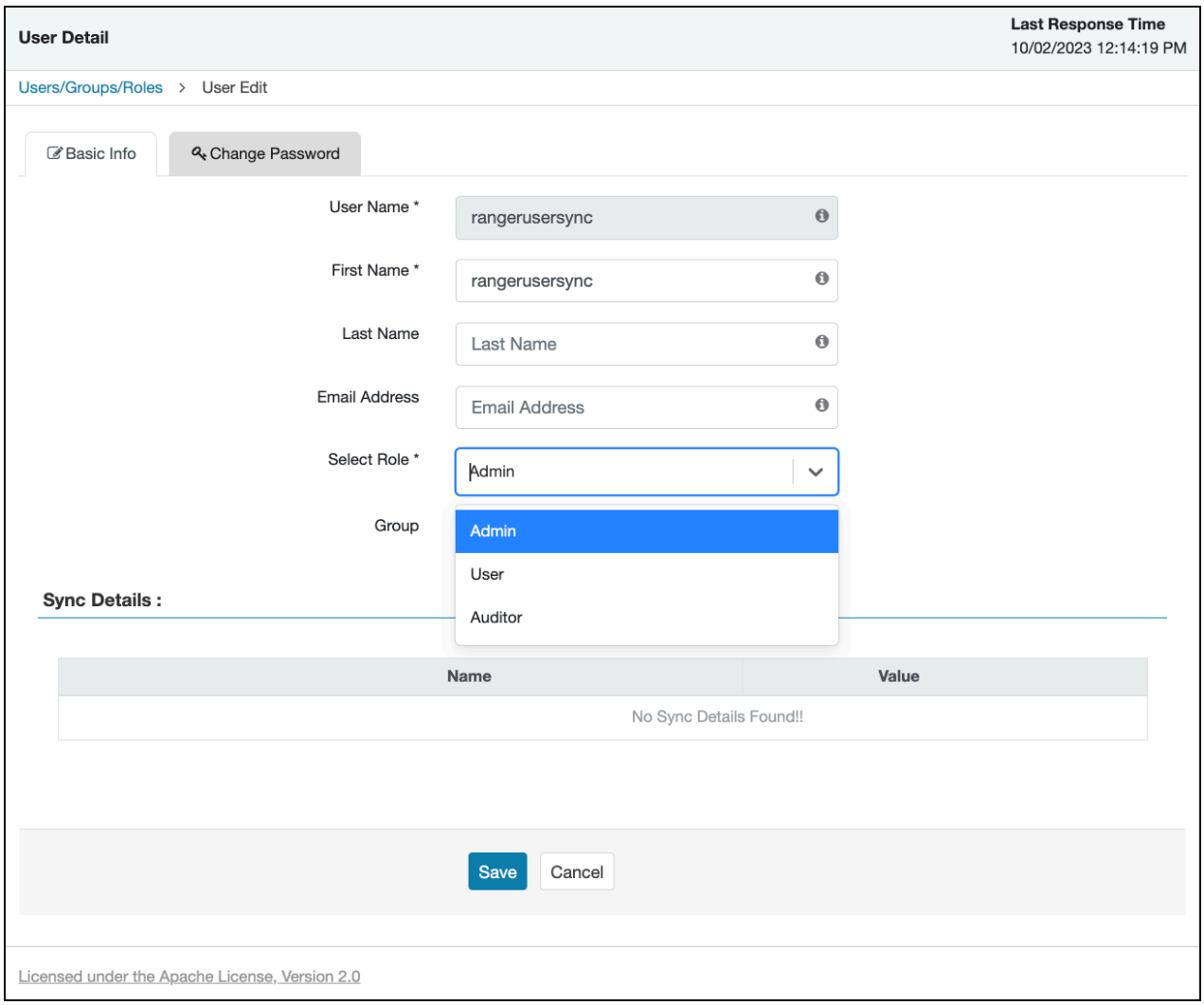

Only users with the Admin role can edit Ranger policies.

Cloudera Runtime 7.3.1

Ranger Authorization

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Ranger to Provide Authorization in CDP 6

Ranger plugin overview 6
- Ranger Hive Plugin 6
- Ranger Kafka Plugin 8
- Ranger-HBase Plugin 9

Ranger special entities 10

Enabling Ranger HDFS plugin manually on a Cloudera Data Hub 11

Ranger Policies Overview 13
- Ranger tag-based policies 14
- Tags and policy evaluation 15
- Ranger access conditions 16

Using the Ranger Admin Web UI 19
- Accessing the Ranger Admin Web UI 19
- Ranger console navigation 21

Resource-based Services and Policies 25
- Configuring resource-based services 26
  - Configure a resource-based service: Atlas 26
  - Configure a resource-based service: HBase 28
  - Configure a resource-based service: HDFS 30
  - Configure a resource-based service: HadoopSQL 34
  - Configure a resource-based service: Kafka 35
  - Configure a resource-based service: Knox 37
  - Configure a resource-based service: NiFi 39
  - Configure a resource-based service: NiFi Registry 41
  - Configure a resource-based service: Solr 43
  - Configure a resource-based service: YARN 45
- Configuring resource-based policies 47
  - Configure a resource-based policy: Atlas 48
  - Configure a resource-based policy: HBase 49
  - Configure a resource-based policy: HDFS 52
  - Configure a resource-based policy: HadoopSQL 54
  - Configure a resource-based storage handler policy: HadoopSQL 57
  - Configure a resource-based policy: Kafka 62
  - Configure a resource-based policy: Knox 63
  - Configure a resource-based policy: NiFi 65
  - Configure a resource-based policy: NiFi Registry 67
  - Configure a resource-based policy: S3 69

Configure a resource-based policy: Solr....71
Configure a resource-based policy: YARN....73
Wildcards and variables in resource-based policies....75
Adding a policy condition to a resource-based policy....77
Adding a policy label to a resource-based policy....79
Preloaded resource-based services and policies....80
Importing and exporting resource-based policies....86
Import resource-based policies for a specific service....89
Import resource-based policies for all services....91
Export resource-based policies for a specific service....94
Export all resource-based policies for all services....95
Row-level filtering and column masking in Hive....97
Row-level filtering in Hive with Ranger policies....97
Dynamic resource-based column masking in Hive with Ranger policies....101
Dynamic tag-based column masking in Hive with Ranger policies....105

Tag-based Services and Policies....108
Adding a tag-based service....108
Adding tag-based policies....110
Using tag attributes and values in Ranger tag-based policy conditions....112
Adding a policy condition to a tag-based policy....113
Adding a tag-based PII policy....114
Default EXPIRES ON tag policy....117
Importing and exporting tag-based policies....119
Import tag-based policies....121
Export tag-based policies....123

Create a time-bound policy....125

Create a Hive authorizer URL policy....127

Showing Role|Grant definitions from Ranger HiveAuthorizer....129

Ranger Security Zones....130
Security Zones Administration....130
Security Zones Example Use Cases....131
Adding a Ranger security zone....133

Administering Ranger Reports....139
View Ranger reports....139
Search Ranger reports....140
Export Ranger reports....141

Using Ranger client libraries....142

Using session cookies to validate Ranger policies....143

Configure optimized rename and recursive delete operations in Ranger Ozone plugin.....143

How to optimally configure Ranger RAZ client performance.....144

Using Ranger to Provide Authorization in CDP

Apache Ranger manages access control through a user interface that ensures consistent policy administration across Cloudera Data Platform (CDP) components. Security administrators can define security policies at the database, table, column, and file levels, and can administer permissions for specific LDAP-based groups or individual users. Rules based on dynamic conditions such as time or geolocation can also be added to an existing policy rule. The Ranger authorization model is pluggable and can be easily extended to any data source using a service-based definition.

Once a user has been authenticated, their access rights must be determined. Authorization defines user access rights to resources. For example, a user may be allowed to create a policy and view reports, but not allowed to edit users and groups. You can use Ranger to set up and manage access to Hadoop services.

Ranger enables you to create services for specific resources (HDFS, HBase, Hive, etc.) and add access policies to those services. Ranger security zones enable you to organize service resources into multiple security zones. You can also create tag-based services and add access policies to those services. Using tag-based policies enables you to control access to resources across multiple components without creating separate services and policies in each component. You can also use Ranger TagSync to synchronize the Ranger tag store with an external metadata service such as Apache Atlas.

Note:

You can configure authorization using the Ranger UI, REST APIs, or client libraries. For more information about:

- Ranger REST APIs, see see https://ranger.apache.org/apidocs/index.html.
- Ranger client libraries, see Using Ranger client libraries.

Ranger plugin overview

Ranger enforces authorization using a plugin model.

Ranger at the core has a centralized web application, which consists of the policy administration. These policies are enforced within the Hadoop ecosystem using lightweight Ranger Java plugins. These plugins run as part of the same process as the namenode (HDFS), HiveServer2(Hive), HiveMetaStore, HBase server (Hbase), Kafka, Solr, NiFi, Raz, RazS3, ADLS, Yarn and Knox server (Knox). Plugins are enabled by default for each of these components except (Solr) and can be disabled individually, using Cloudera Manager.

Ranger plugins exist in the path of the user request. Each plugin decides whether to allow or deny user requests for accessing. Each plugin also collects and stores the access request details as access audit log records.

Ranger plugins enforce the policies defined in the policy database. Ranger Admin users can create a policy for a specific set of resources and assign a specific set of permissions to a specific set of users, groups and roles. Ranger admin users manage policies using the Ranger Admin Web UI.

Ranger policies are independent from native permissions (os permission). Ranger uses native permissions to authorize user access in the case that an applicable Ranger policy does not exist in the policy database.

Ranger Hive Plugin

Describes how the Ranger Hive plugin enforces authorization.

Ranger Hive Plugin is enabled in HiveServer2 which helps in storage-based authorization and SQL-standard authorization. In storage-based authorization when a new table is created by running CREATE TABLE statement in Beeline, which will submit query to HiveServer2 for processing, and before HiveServer2 is able to run the query, it will check the policy cache file and make sure the user who submits the query has the appropriate permission to perform the task. Once the authorization passes, a query is submitted and a table created.

Upon successful creation of the new table, the following will be triggered by Ranger's Hive plugin:

1. Sends audit event to Solr and/or HDFS

SQL standard authorization provides grant/revoke functionality at database, table level. When a grant command is executed in beeline it updates/creates a policy for that user and when revoke is executed the user is added in the deny condition of the policy.

Ranger Hive Plugin Enforcement Example

Prerequisite

1. Create a database, table, column in hive service and also insert some data into it with hive user.
   - create database vehicle;
   - create table vehicle.cars(car_id int, car_name string, car_color string, car_price int);"
   - insert into vehicle.cars(car_id, car_name, car_color, car_price) VALUES (1,'car1','color1',100000), (2,'car2','color2',200000), (3,'car3','color3',300000), (4,'car4','color4',400000);
   - select * from vehicle.cars;
2. Create external user 'externaluser1'

Access Enforcement steps

1. Let's try to access the vehicle.cars table using 'externaluser1'.

   'externaluser1' will be denied access, because 'externaluser1' lacks permission to access the vehicle.cars table.
2. Lets create a policy in ranger-hive for the user:
   - Resource : [database=vehicle, table=cars, column=*]
   - allow policy item : [user='externaluser1', permission=select]
3. Let's try to access the vehicle.cars table using 'externaluser1'.

   'externaluser1' will be allowed access, because 'externaluser1' now has permission to access the vehicle.cars table.
4. You can check the logs related to these actions, using Ranger Admin Web UI Access Audit tab.

Masking Enforcement steps

Suppose you don't want to show the car_price to 'externaluser1' user so we can mask the data of that column for that user.

1. Lets create a masking policy in ranger-hive for the user:
   - Resource : [database=vehicle, table=cars, column=car_price]
   - allow policy item : [user='externaluser1', permission=select, Select Masking Option=Partial mask: show last 4]
2. Now let's try to access the vehicle.cars table using 'externaluser1'

   'externaluser1' will see the car_price - only last 4 digits - because 'externaluser1' has masked access to vehicle.cars table.

Row Level Filtering steps

Suppose you don't want to show the only one row to 'externaluser1' user so we can do it using the row filter policy.

1. Lets create a masking policy in ranger-hive for the user:
   - Resource : [database=vehicle, table=cars]
   - allow policy item : [user='externaluser1', permission=select, Row Level Filter=car_color = 'color4']
2. Now let's try to access the vehicle.cars table using 'externaluser1'.

   'externaluser1' will see only the row whose car_color is 'color4'.

Table 1: Hive Commands to Ranger Permission Mapping

Permission	Action
SELECT	Gives read access to an object.
CREATE	Hive Create Table statement is used to create table.
UPDATE	Gives the ability to run update queries on an object (table).
ALTER	You can rename the table and column of existing Hive tables. You can add a new column to the table. Rename Hive table column. Add or drop table partition. Add Hadoop archive option to Hive table.
DROP	DROP TABLE command in the hive is used to drop a table inside the hive.
INDEX	An Index is nothing but a pointer on a particular column of a table. Creating an index means creating a pointer on a particular column of a table.
LOCK	Is used to lock the table.
Read	Read data from HDFS using hdfs or other cloud locations.
Write	Export Data to a location in hdfs or other cloud locations.
ReplAdmin	ReplAdmin privilege is related to REPL DUMP and REPL LOAD commands.
Service Admin	Enable hive ranger plugin to isolate various admin operations, in this case "Kill Query". "Service Admin" won't be able to do DATABASE/TABLE/COLUMN operations as this will all be taken care by the existing DATABASE/TABLE/COLUMN level permission model.
Temporary UDF Admin	Temporary UDF Admin is needed for creating UDFs.
Refresh	Refresh is used by only impala.
ALL	This is for all the permission mentioned above.

Ranger Kafka Plugin

Describes how the Ranger Kafka plugin enforces authorization.

Ranger Kafka plugin is enabled in master.

Ranger Kafka Plugin Enforcement Example

Prerequisite

1. Create external user 'externaluser3'

Access Enforcement steps

1. Let's try to create a topic and send some data using ‘externaluser3’, he will be denied as he doesn't have permission to create it.
2. Lets create a policy in ranger-kafka for the user
   - Resource : [Topic=topictest01]
   - allow policy item : [user='externaluser3', permission=publish, consume, describe, create]
3. Let's try to create a topic and send some data using ‘externaluser3’, he will be allowed as he gets permission to access it.
4. You can check the logs related to these actions, using Ranger Admin Web UI Access Audit tab.

Table 2: Kafka Commands to Ranger Permission Mapping

Permission	Action
Resource = topic	
Publish, Describe, Create	To produce topic and publish
Describe, Create	To describe topic
Describe	sending message to topic
Publish	To publish topic
Consume	To read data (consume)
Describe	To list topic
Configure	To alter config of topic
Delete	To delete topic
Describe Config	To describe config of topic
Alter Config	To alter config
Resource = consumergroup	
Describe	To describe topic
Consume	To consume topic
Resource = cluster	
Create	To create topic
Describe	To describe topic
Idempotent Write	To write idempotently
Resource = transactionid	
Describe, Publish	To publish and describe

Ranger-HBase Plugin

Describes how the Ranger HBase plugin enforces authorization.

Ranger HBase Plugin is enabled in master which helps in authorization of the column-oriented database management system.

Ranger HBase Plugin Enforcement Example

Prerequisite

1. Create a sample table called "data", with two column families, and add some rows to the table with hbase user.
   - create 'Testtable1', 'personal', 'medical','finance'
   - put 'Testtable1', '1', 'personal:fname', 'Mike'
   - get 'Testtable1','1','personal:fname'
2. Create external user 'externaluser2'

Access Enforcement steps

1. Let's try to access the 'Testtable1' table using 'externaluser2', he will be denied as he don't have permission to access it.

2. Lets create a policy in ranger-hive for the user
   - Resource : [HBase Table=Testtable1, Column-family=*, Column=*]
   - allow policy item : [user='externaluser2', permission=read]
3. Let's try to access the vehicle.cars table using 'externaluser2', he will be allowed as he gets permission to access it.
4. You can check the logs related to these actions, using Ranger Admin Web UI Access Audit tab.

Table 3: HBase Commands to Ranger Permission Mapping

Permission	Action
Read (R)	can read data at the given scope
Write (W)	can write data at the given scope
Execute (X)	can execute coprocessor endpoints at the given scope
Create (C)	can create tables or drop tables (even those they did not create) at the given scope
Admin (A)	can perform cluster operations such as balancing the cluster or assigning regions at the given scope

Ranger special entities

Ranger in Cloudera has specific, internal groups and entities that affect user authorization and access to all services in Cloudera.

In addition to any users, group, roles and permissions that you define using Ranger, you must understand the following Ranger special entities:

"public" group

A special, internal group within Ranger that consists of all users, including future users. Membership is implicit and automatic.

Note: All users belong to "public" group. Any policies granted to this group provide access to everyone.

The following, default policies give permissions to members of group "public":

- all - database > public > create permission
- default database tables columns > public > create permission
- Information_schema database tables columns > public > select permission

You can remove “public” from these default policies to further restrict user access, based on your security requirements.

{OWNER} special entity

A special Ranger entity attached to a user based on their actions. For example, when a user "bob" creates a table, "bob" becomes the {OWNER} of that table and would get any permissions provided to {OWNER} on that table across all the policies. The following default policies have permissions for {OWNER}:

- all - database, table, column > {OWNER} > all permissions
- all - database, table > {OWNER} > all permissions
- all - database, udf > {OWNER} > all permissions
- all - database > {OWNER} > all permissions

Although not recommended, you can modify access to special entity {OWNER}, based on your security requirements. Removing the default {OWNER} permissions may require adding additional,

specific policies for each object owner, which may increase your policy management operational burden.

Enabling Ranger HDFS plugin manually on a Cloudera Data Hub

How to enable an HDFS plugin for Ranger, service-wide, on a Cloudera Data Hub using Cloudera Manager.

About this task

The Ranger HDFS plug-in helps to centralize HDFS authorization policies. Apache Ranger plugins validate the access of a user against the authorization policies defined in the Apache Ranger policy administration server, and stored in the HDFS service instance, also called a repository. When you enable the Ranger HDFS plugin and an HDFS service user attempts access, Ranger checks whether a policy exists granting or denying the user access. If no policy exists, Ranger defaults to use the native permissions model in HDFS. Access control rules configured through this combination of Ranger HDFS plugin and native file system permissions apply.

To enable users define Ranger authorization polices, using an HDFS service plugin:

Procedure

1. In a Cloudera Data Hub, go to Cloudera Manager HDFS Configuration .

2. In Search, type Ranger Service, then click the box to enable the hdfs (service-wide) parameter for Ranger Service.

Figure 1: Enabling the HDFS Ranger plugin parameter on a Cloudera Data Hub

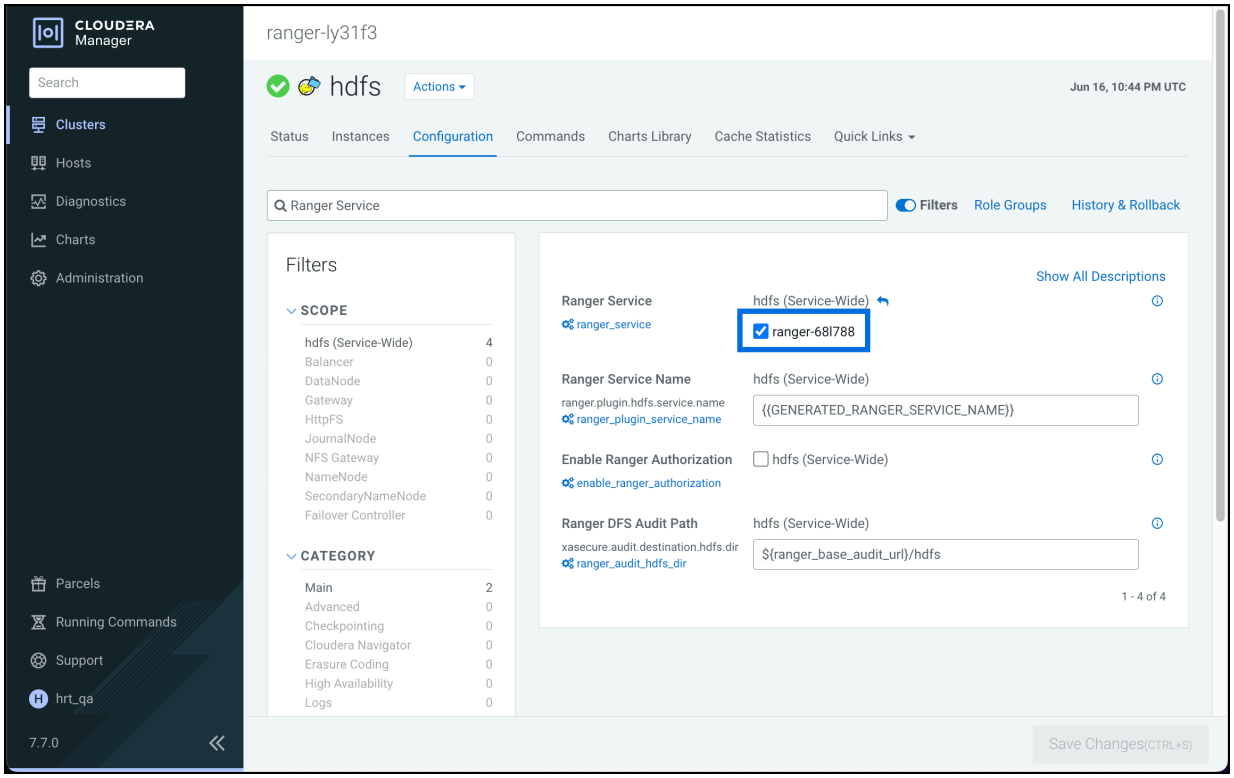

A stale configuration icon displays for the hdfs service.

Figure 2: Result of adding a new configuration parameter

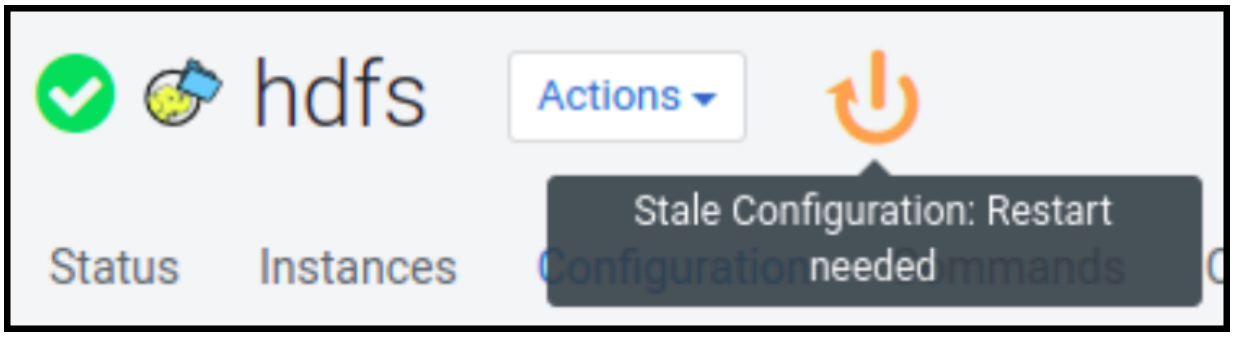

3. Before restarting Hdfs service for stale configurations, choose HDFS Actions Create Ranger Repository . After progress completes, close the Create Repository dialog.
4. Now proceed to restart the Hdfs service. Click HDFS Actions Restart . After progress completes, close the Restart dialog.
5. On the Data Lake, log in to Ranger.
6. Go to Admin Web UI Access Manager .

7. In Service Manager HDFS , confirm that (Cloudera Data Hub cluster name)_hdfs appears.

Figure 3: Confirming HDFS plugin added

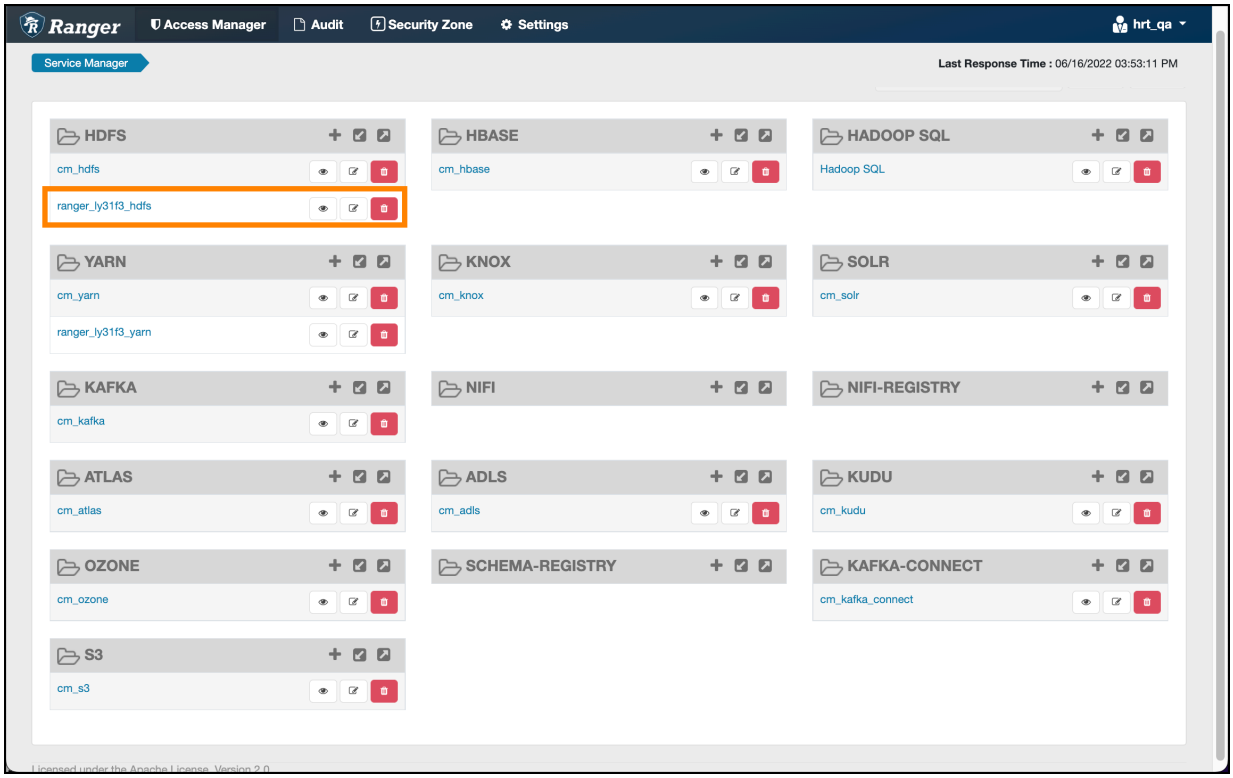

8. Go to Audit Plugins .

Results

Confirm that the Http response code for the Ranger Hdfs plugin enabled on the Cloudera Data Hub Hdfs service displays 200 (successful).

Figure 4: Confirming successful http response

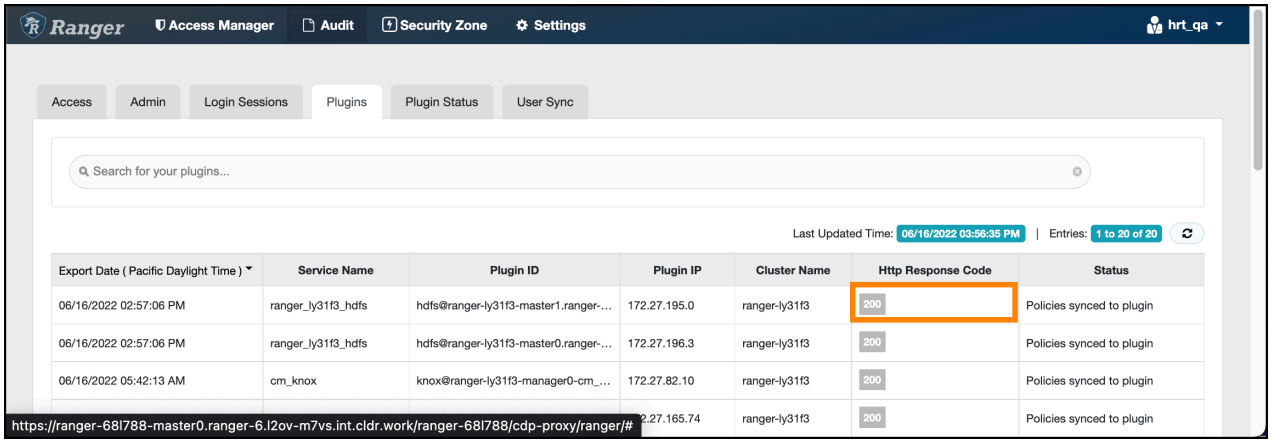

Ranger Policies Overview

Ranger has two types of policies: resource-based and tag-based.

Resource-based policies

Ranger enables you to configure resource-based services (HDFS, HBase, Hive, etc.) and add access policies to those services.

Tag-based policies

Ranger enables you to create tag-based services and add access policies to those services.

Ranger tag-based policies

Ranger enables you to create tag-based services and add access policies to those services.

Tag-Based Policies Overview

- An important feature of Ranger tag-based authorization is the separation of resource-classification from access-authorization. For example, resources (HDFS file/directory, Hive database/table/column etc.) containing sensitive data such as social security numbers, credit card numbers, or sensitive health care data can be tagged with PII/PCI/PHI – either as the resource enters the Hadoop ecosystem or at a later time. Once a resource is tagged, the authorization for the tag would be automatically enforced, thus eliminating the need to create or update policies for the resource.

  Note: Tags applied on a Hive table are propagated to the views created from that table. Hence, if any Ranger tag based access or masking policies are associated with those tags, then the views also have those policies applied.
- Using tag-based policies also enables you to control access to resources across multiple Hadoop components without creating separate services and policies in each component.
- Tag details are stored in a tag store. Ranger TagSync can be used to synchronize the tag store with an external metadata service such as Apache Atlas.

Tag Store

Details of tags associated with resources are stored in a tag store. Apache Ranger plugins retrieve the tag details from the tag store for use during policy evaluation. To minimize the performance impact during policy evaluation (in finding tags for resources), Apache Ranger plugins cache the tags and periodically poll the tag store for any changes. When a change is detected, the plugins update the cache. In addition, the plugins store the tag details in a local cache file – just as the policies are stored in a local cache file. On component restart, the plugins will use the tag data from the local cache file if the tag store is not reachable.

Apache Ranger plugins download the tag details from the store managed by Ranger Admin. Ranger Admin persists the tag details in its policy store and provides a REST interface for the plugins to download the tag details.

Tags

Ranger Tags can have attributes. Tag attribute values can be used in Ranger tag-based policies to influence the authorization decision.

For example, to deny access to a resource after a specific date:

1. Add the EXPIRES_ON tag to the resource.
2. Add an exipry_date tag attribute and set its value to the expiry date.
3. Create a Ranger policy for the EXPIRES_ON tag.
4. Add a condition in this policy to deny access when the date specified the in expiry_date tag attribute is later than the current date.

Note that the EXPIRES_ON tag policy is created as the default policy in tag service instances.

TagSync

Ranger TagSync is used to synchronize the tag store with an external metadata service such as Apache Atlas. TagSync is a daemon process similar to the Ranger UserSync process.

Ranger TagSync receives tag details from Apache Atlas via change notifications. As tags are added to, updated, or deleted from resources in Apache Atlas, Ranger TagSync receives notifications and updates the tag store.

Tags and policy evaluation

When authorizing an access request, an Apache Ranger plugin evaluates applicable Ranger policies for the resource being accessed. The following diagram shows the details of the policy evaluation flow. More details on the steps in this workflow are provided in the subsequent sections.

Apache Ranger Policy Evaluation Flow with Tags

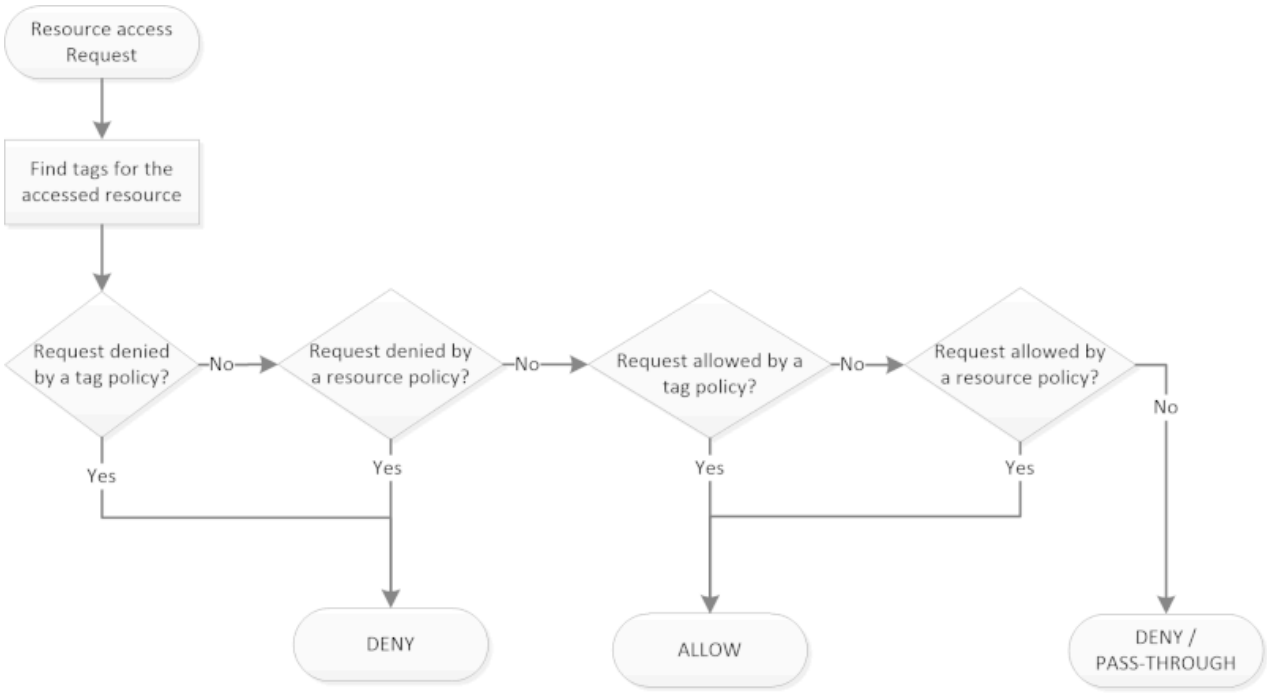

Apache Ranger Policy Evaluation Flow with Tags

Finding Tags

Apache Ranger supports a service to register context enrichers, which are used to update context data to the access request.

The Ranger Tag service, which is part of the tag-based policies feature, adds a context enricher named RangerTagEnricher. This context enricher is responsible for finding tags for the requested resource and adding the tag details to the request context. This context enricher keeps a cache of the available tags; while processing an access request, it finds the tags applicable for the requested resource and adds the tags to the request context. The context enricher keeps the cache updated by periodically polling Ranger Admin for changes.

Evaluating Tag-Based Policies

Once the list of tags for the requested resource is found, the Apache Ranger policy engine evaluates the tag-based policies applicable to the tags. If a policy for one of these tag results in a deny, access will be denied. If none of the tags are denied, and if a policy allows for one of the tags, access will be allowed. If there is no result for any tag, or if there are no tags for the resource, the policy engine will evaluate the resource-based policies to make the authorization decision.

Using Tags in Conditions

Apache Ranger allows the use of custom conditions while evaluating authorization policies. The Apache Ranger policy engine makes various request details – such as user, groups, resource, and context – available to the conditions. Tags in the request context, which are added by the enricher, are available to the conditions and can be used to influence the authorization decision.

The default policy in tag service instances, the EXPIRES_ON tag, uses such condition to check to see if the request date is later than the value specified in tag attribute expiry_date. This default policy does not work unless an EXPIRES_ON tag has been created in Atlas.

Related Information

Apache Ranger Wiki> Context Enrichers

Ranger access conditions

The Apache Ranger access policy model consists of two major components: specification of the resources a policy is applied to, such as HDFS files and directories, Hive databases. tables. and columns, HBase tables, column-families, and columns, and so on; and the specification of access conditions for specific users and groups

Allow Deny and Exclude Conditions

Apache Ranger supports the following access conditions:

- Allow
- Exclude from Allow
- Deny
- Exclude from Deny

These access conditions enable you to set up fine-grained access control policies.

For example, you can allow access to a "finance" database to all users in the "finance" group, but deny access to all users in the "interns" group. Let's say that one of the members of the "interns" group, "scott", needs to work on an assignment that requires access to the "finance" database. In that case, you can add an Exclude from Deny condition that will allow user "scott" to access the "finance" database. The following image shows how this policy would be set up in Apache Ranger:

Policy Evaluation of Access Conditions

Apache Ranger policies are evaluated in a specific order to ensure predictable results (if there is no access policy that allows access, the authorization request will typically be denied). The following diagram shows the policy evaluation work-flow:

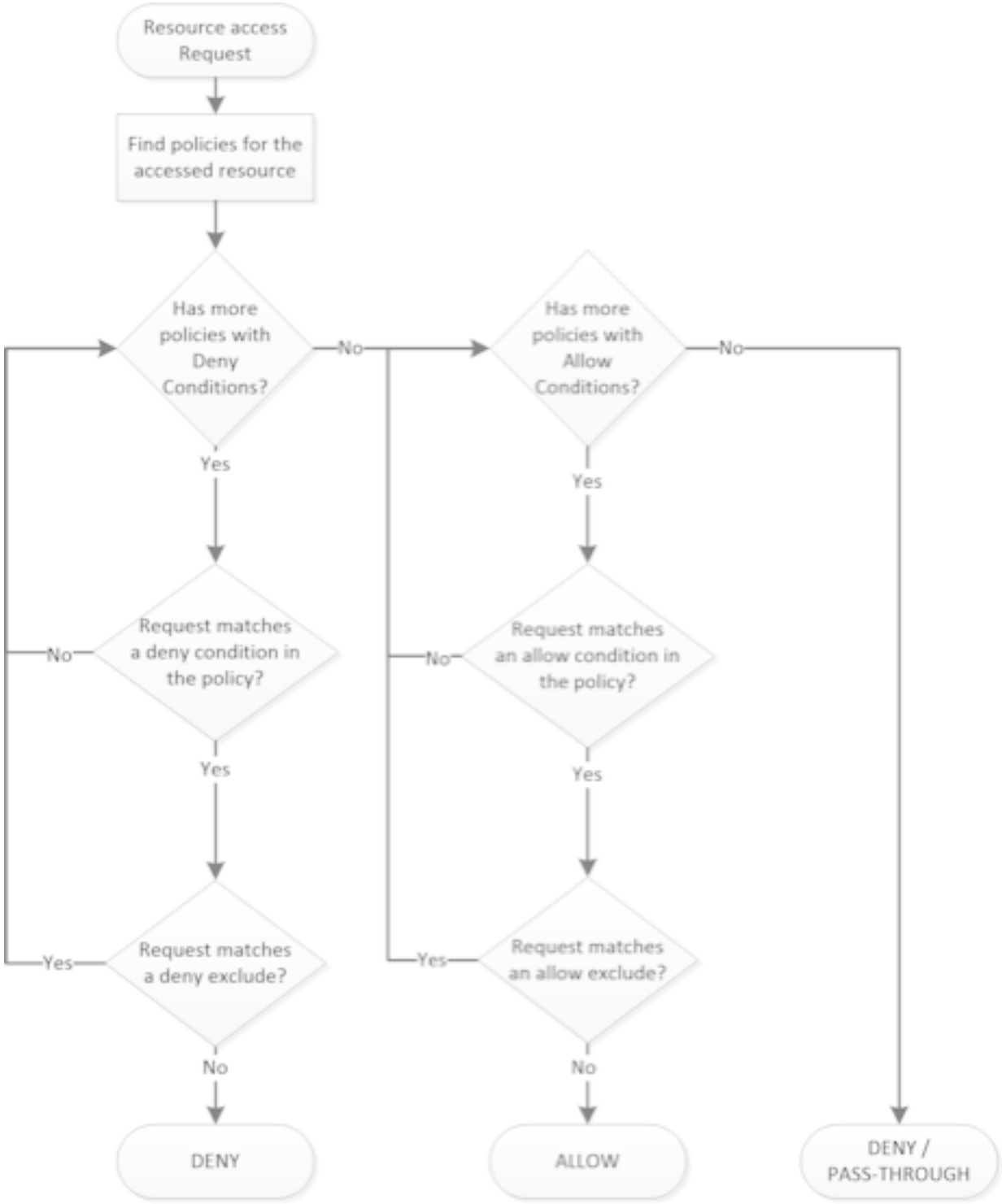

Apache Ranger Policy Evaluation Flow

Using the Ranger Admin Web UI

Log in through the Ranger Admin Web UI as a Ranger administrator user to administer auditing, services for Cloudera resources, access policies for those services and permissions to the Ranger Admin modules for other Ranger users, groups and roles.

Accessing the Ranger Admin Web UI

How to access the Ranger Admin Web UI.

To access the Ranger Admin Web UI, go to Cloudera Manager Ranger Ranger Admin Web UI , type your user name and password, and then click Sign In.

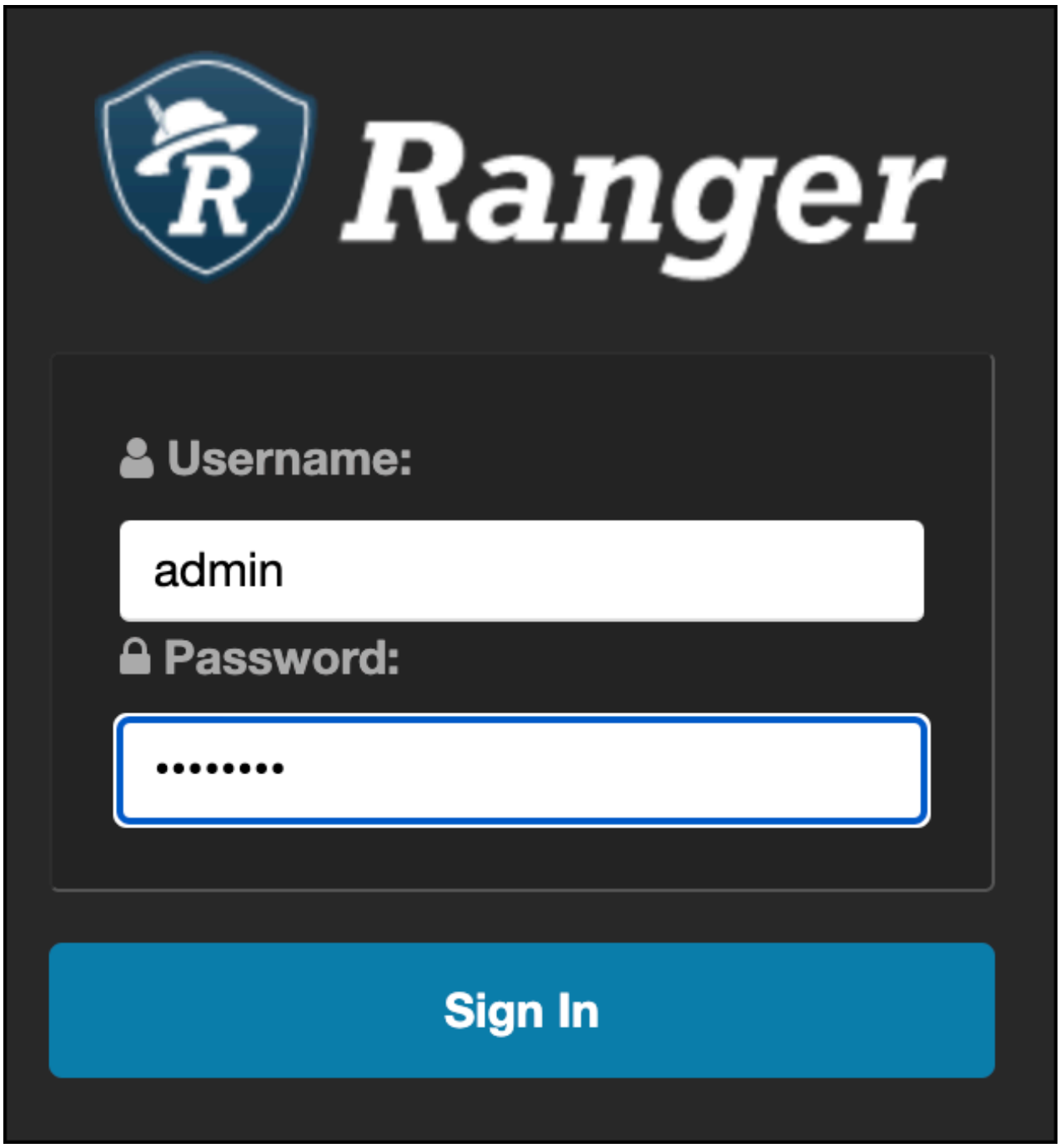

Ranger Admin Web UI Home Page

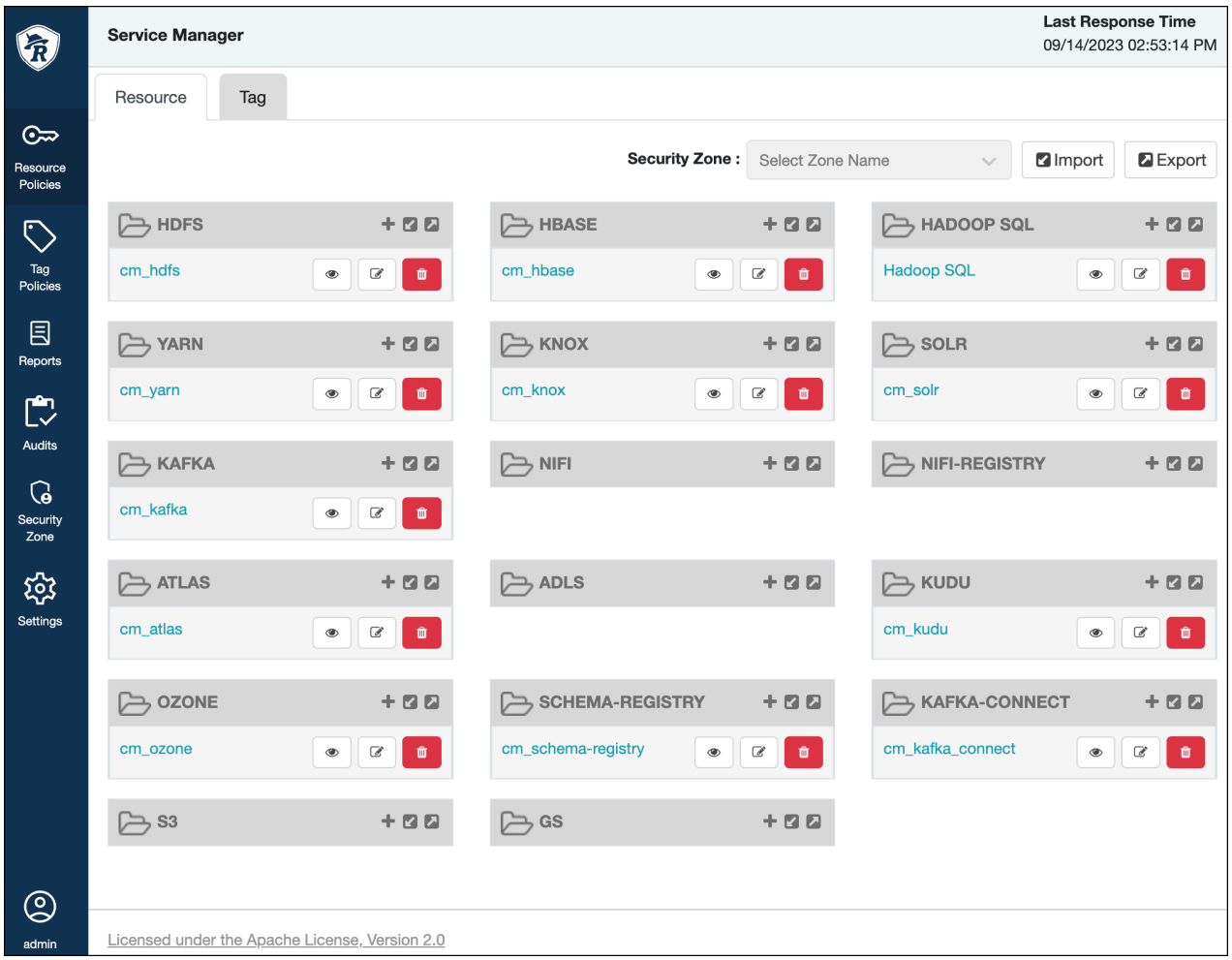

After you log in, your user name is displayed at the lower left of the Ranger Admin Web UI.

Ranger console navigation

Explains the basic Ranger console/GUI.

- The Service Manager for Resource Based Policies page displays when you log in to Ranger Admin Web UI. You can use Service Manager to create services for Cloudera resources (HDFS, HBase, Hive, etc.) and add access policies to those resources.

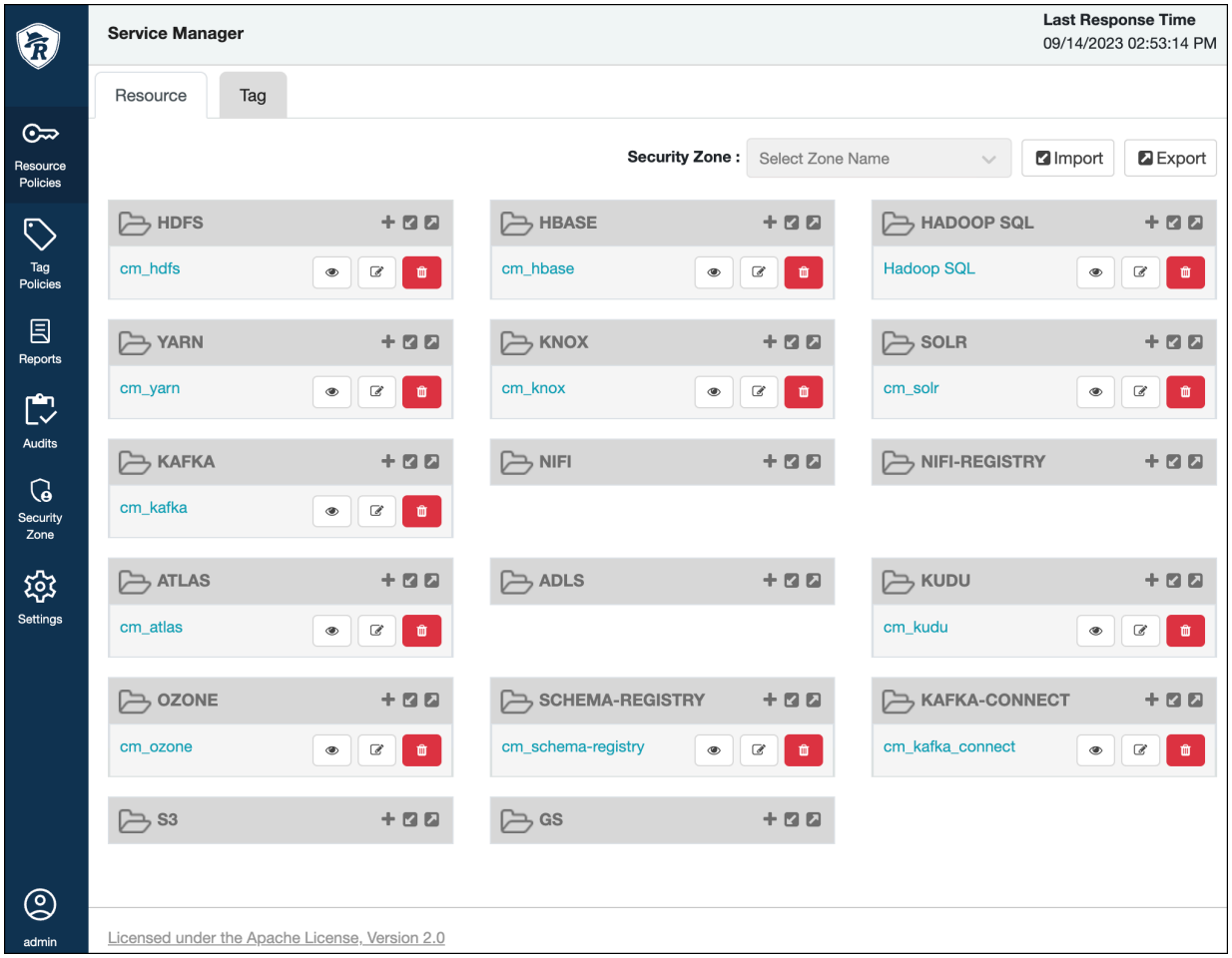

Use the left navigation panel to navigate the Ranger Admin Web UI.

To return to the Service Manager home page, click the Ranger icon at the uppper left corner of the Ranger Admin Web UI page.

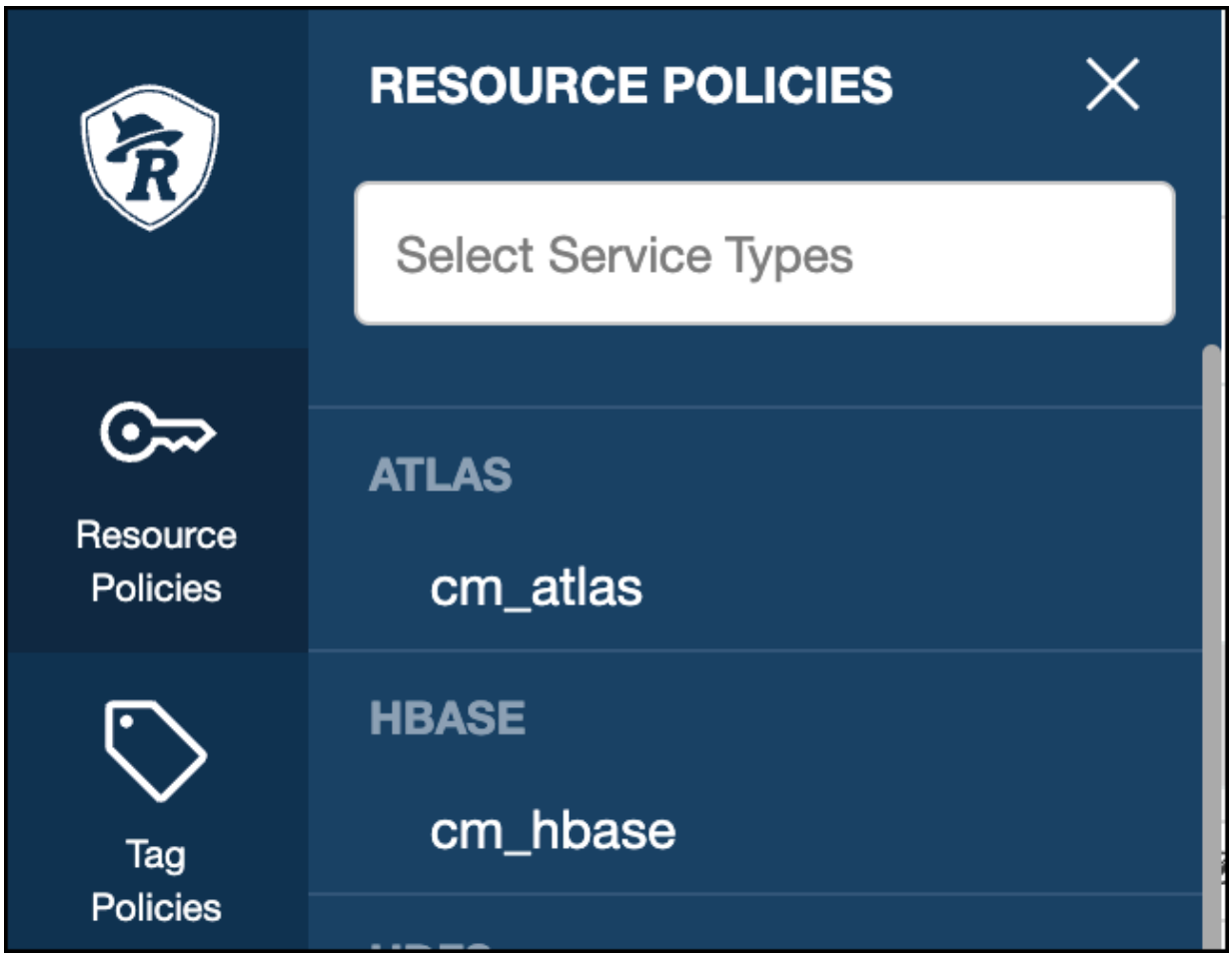

- Resource Policies -- Clicking Resource Polices displays a list of resource-based policies. Click a specific policy name to open policy management page for the selected service. You can use the policy page to administer access policies for that service.
- Tag Policies -- Clicking Tag Polices displays a list of resource-based policies. Click a specific policy name to open policy management page for the selected tag-based policy. You can use the Tag policy page to administer access policies for tag-based policies.
- Reports -- Clicking Reports opens the Reports page. You can use the Reports page to generate user access reports for resource and tag-based policies based on search criteria such as policy name, resource, group, and user name.
- Audits -- Click Audits, then select Access, Admin, Login Sessions, Plugins, Plugin Status or User Sync to access the Audit page Access, Admin, Login Sessions, Plugins, Plugin Status, and User Sync tabs. These UIs provide

administrator access to monitor user activity at the resource level, and also to set up conditional auditing based on users, groups, or time.

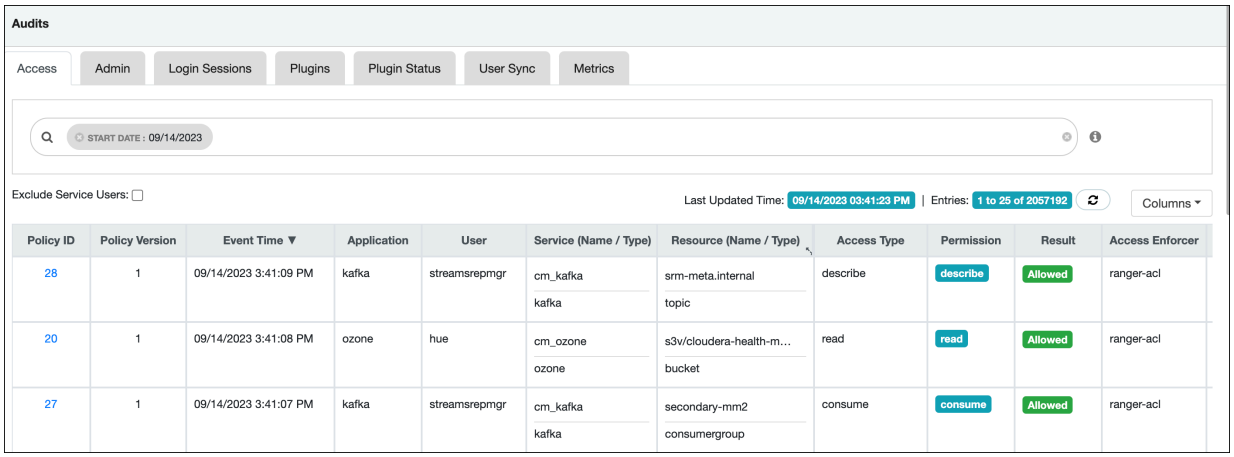

- Security Zone -- Lets you organize resource and tag-based services and policies into separate security zones. You can assign one or more administrators for each security zone. Security zone administrators can then create and update policies for their security zone.

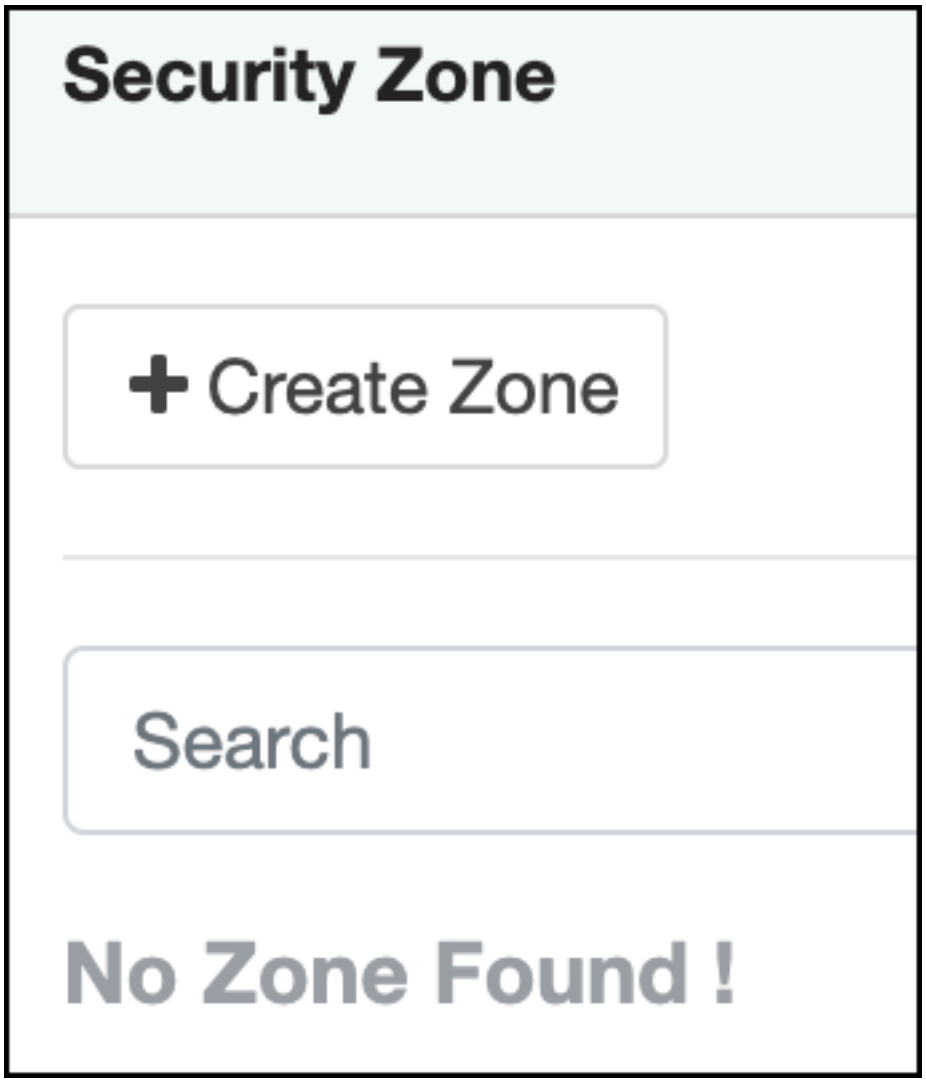

- Settings -- Enables you to manage and assign policy permissions to users and groups. Select the appropirate link to the Users, Groups, Roles, and Permissions pages.

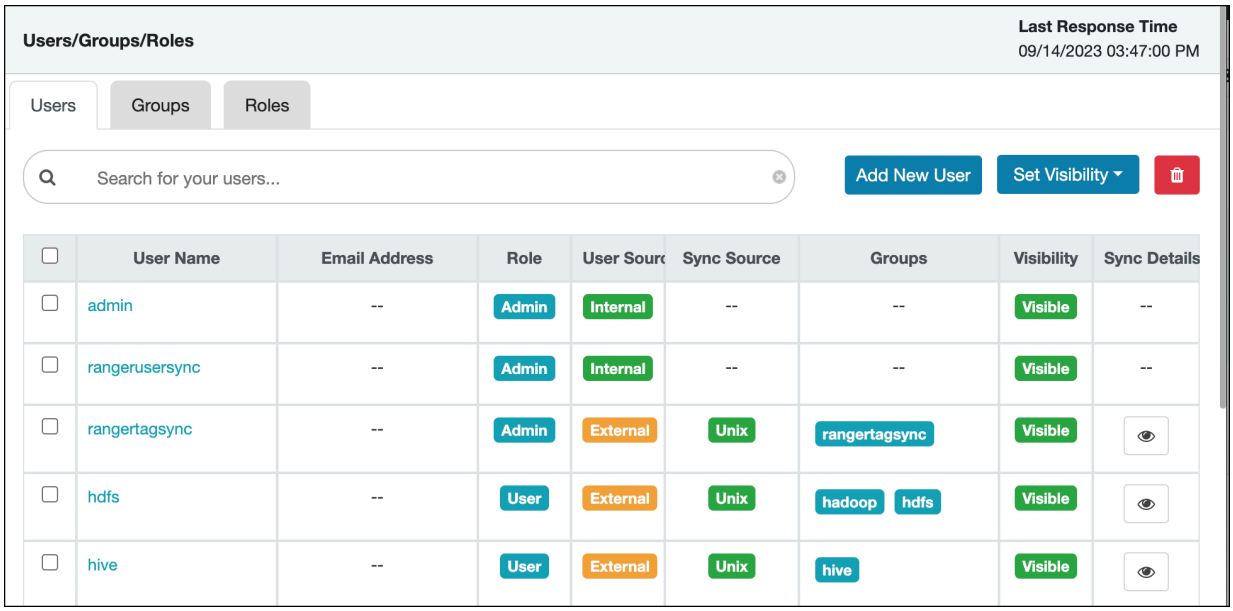

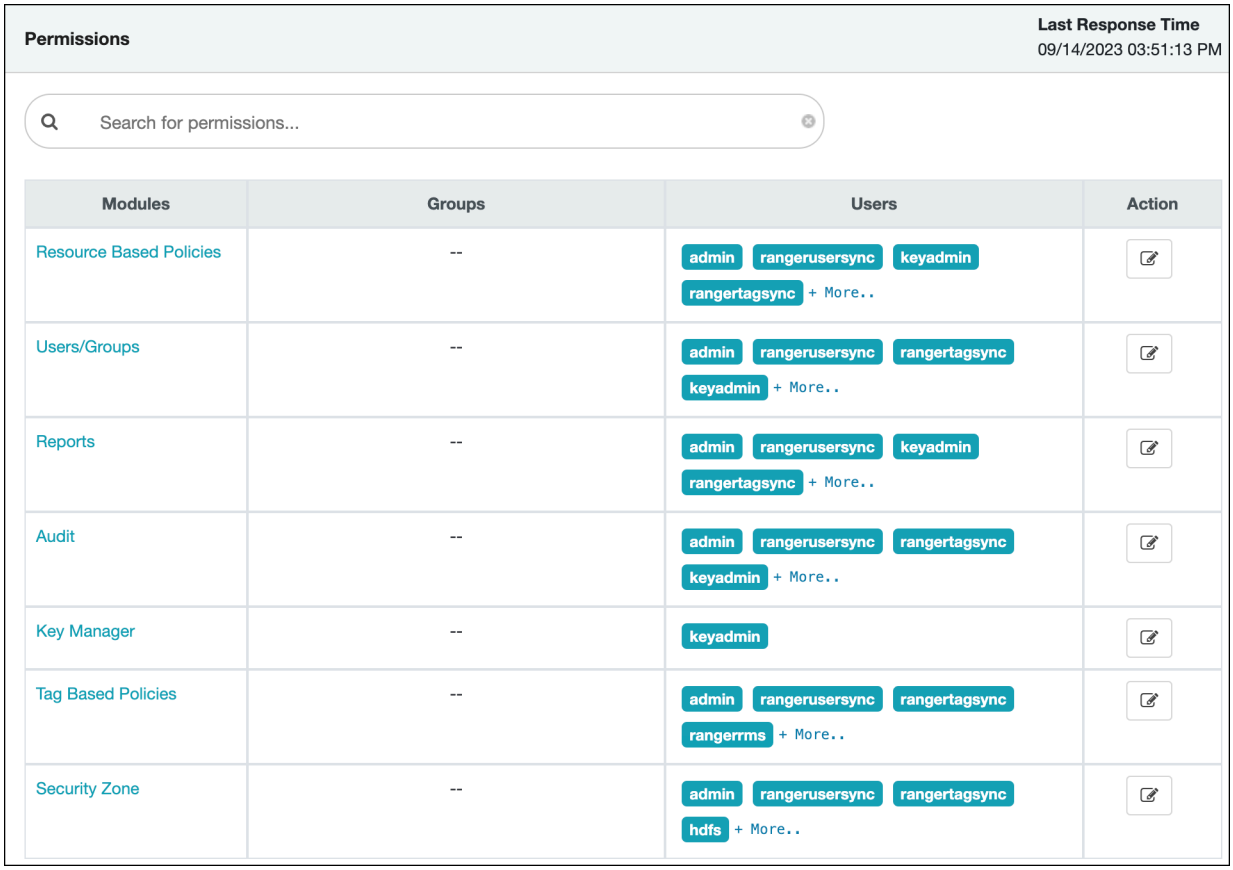

Resource-based Services and Policies

Ranger enables you to configure resource-based services for Hadoop components (e.g. HBase, Kafka, Storm, etc.) and add access policies to those services.

Configuring resource-based services

The Service Manager displays the Resource Policies Resource page when you log in to the Ranger Admin Web UI. You can also access this page by selecting Service Manager Resource Policies , or by clicking the Ranger icon at the upper left of the Ranger Admin Web UI. You can use Resource to add, edit or delete services for Hadoop resources (HDFS, HBase, HadoopSQL, etc.) and add access policies for those resources.

- To add a new resource-based service, click Add ( ) in the applicable box on Service Manager. Enter the required configuration settings, then click Add.
- To edit a resource-based service, click Edit ( ) at the right of the service. Edit the service settings, then click Save to save your changes.
- To delete a resource-based service, click Delete ( ) at the right of the service. Deleting a service also deletes all of the policies for that service.

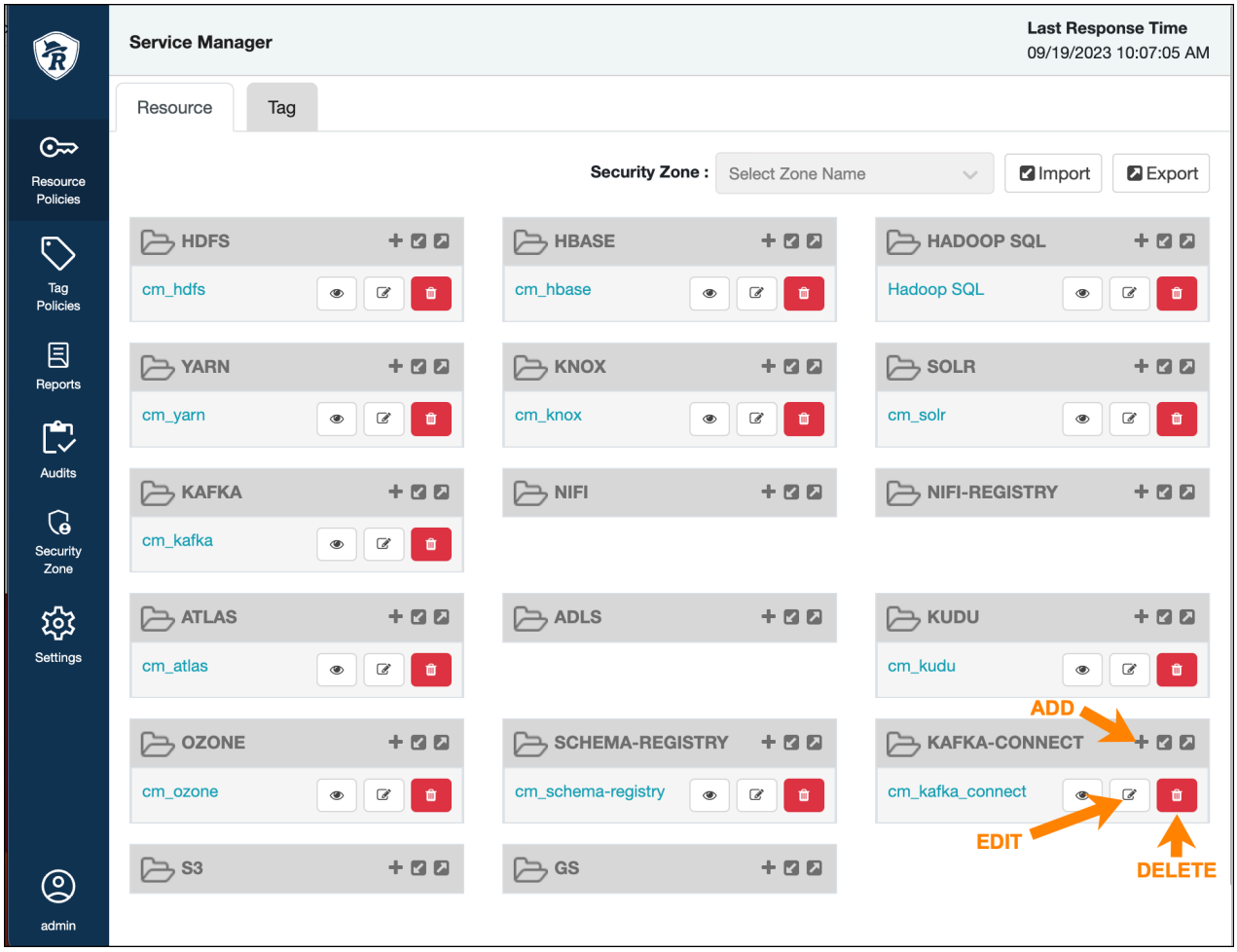

Configure a resource-based service: Atlas

How to add an Atlas service.

Procedure

1. On Service Manager Resource Policies , click Add New Service ( ) next to Atlas.

   The Create Service page appears.

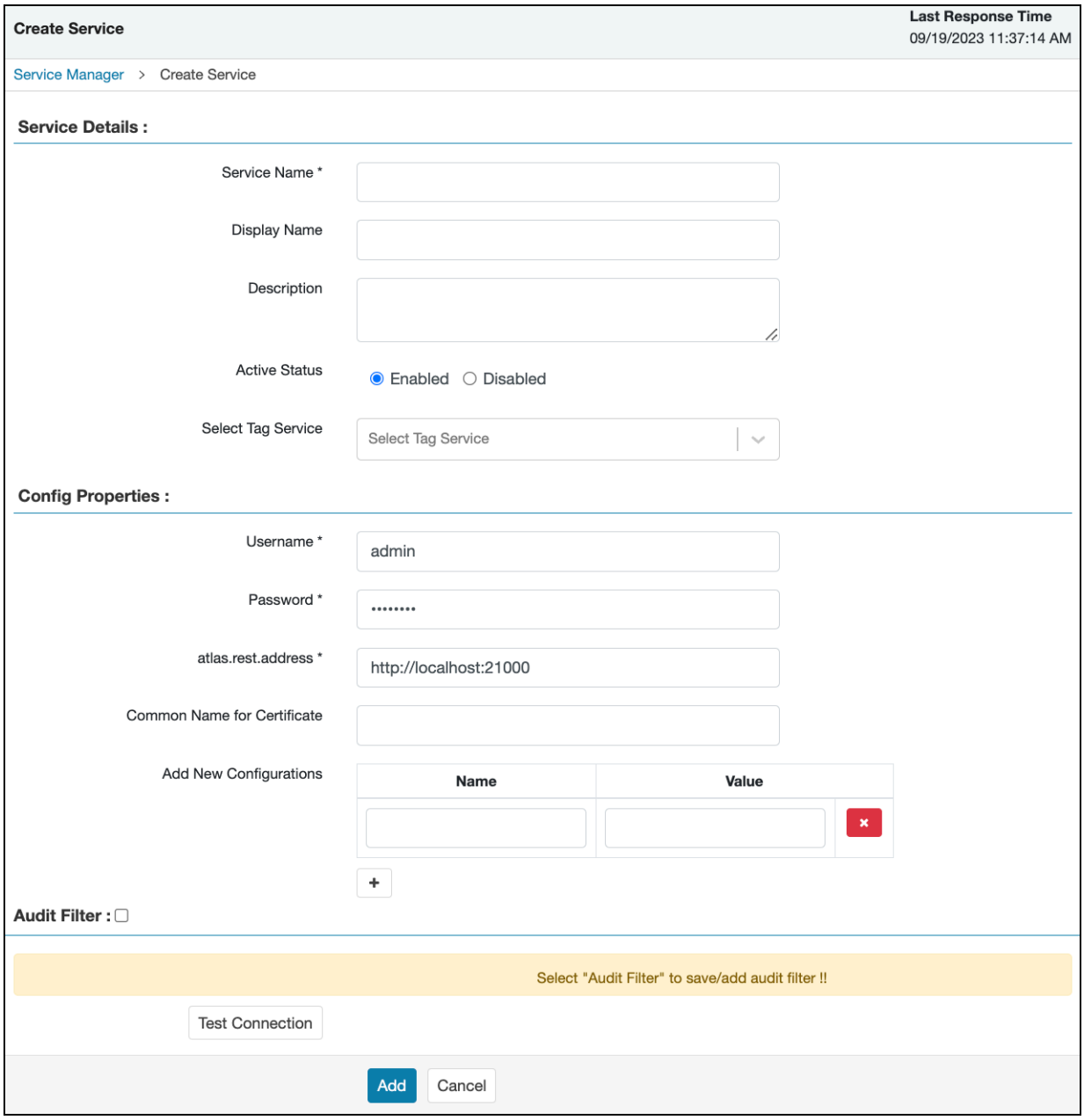

2. On Create Service, enter the following information:

Table 4: Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Display Name	The name which will appear on Service Manager.
Description	A description of the service.

Field name	Description
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Atlas.

Table 5: Configuration Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
atlas.rest.address (SSL environment)	Atlas host and port. Update in the Ranger plugin services to get resource lookup working. Value is https://<Atlas-server>:<https-port>. Default port is 31443.
Common Name For Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click Test Connection.
4. Click Add.

Configure a resource-based service: HBase

How to add an HBase service.

Procedure

1. On Service Manager Resource Policies , click Add New Service ( ) next to HBase.
   The Create Service page appears.

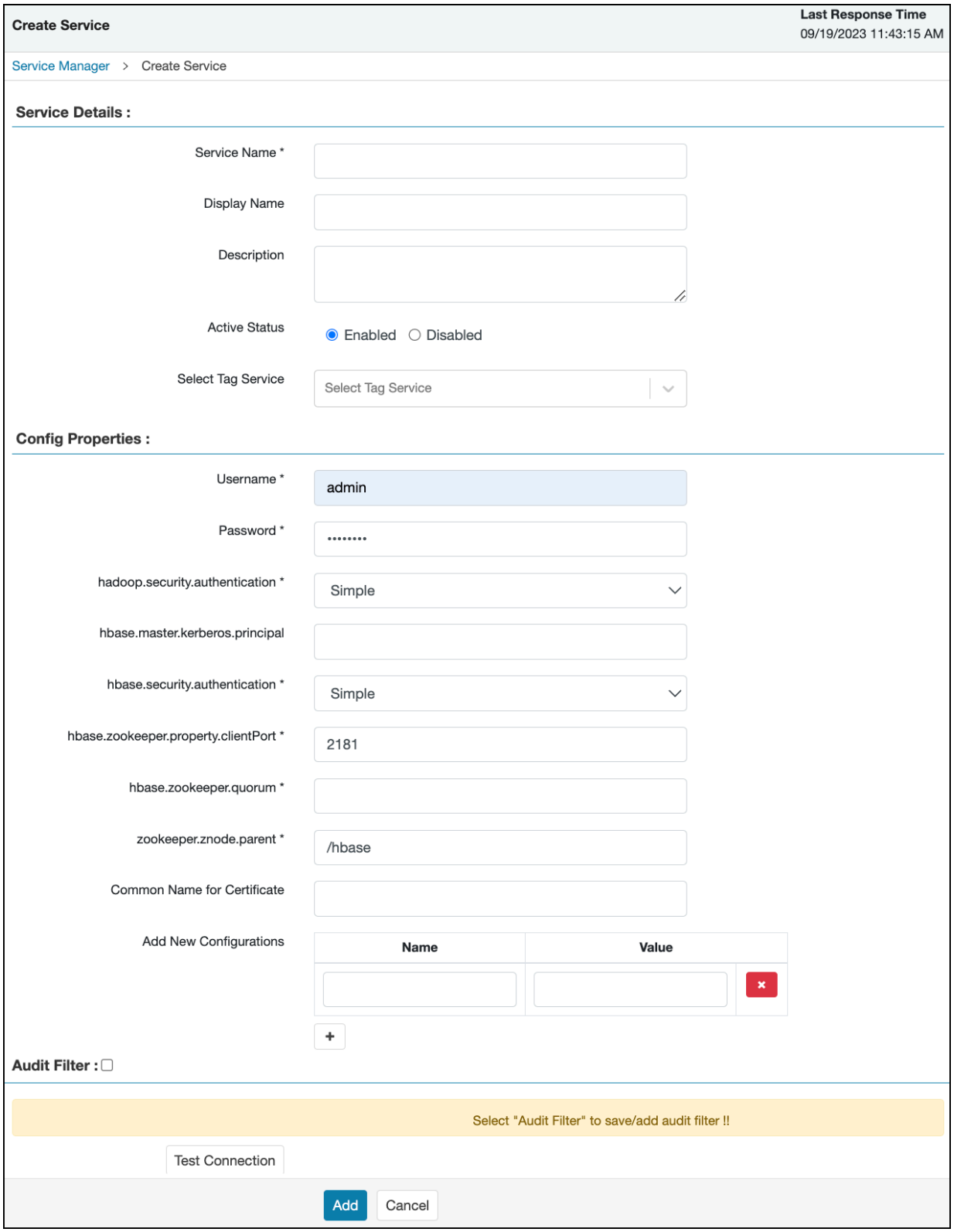

2. On Create Service, enter the following information:

Table 6: Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Display Name	The name which will appear on Service Manager.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to HBase.

Table 7: Configuration Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
hadoop.security.authentication	Values are simple and kerberos.
hbase.master.kerberos.principal	The Kerberos principal for the HBase Master. (Required only if Kerberos authentication is enabled). Update in the Ranger plugin services to get resource lookup working. For example, hbase/_HOST@<REALM>.
hbase.security.authentication	As noted in the hadoop configuration file hbase-site.xml.
hbase.zookeeper.property.clientPort	As noted in the hadoop configuration file hbase-site.xml.
hbase.zookeeper.quorum	As noted in the hadoop configuration file hbase-site.xml.
zookeeper.znode.parent	As noted in the hadoop configuration file hbase-site.xml.
Common Name for Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click Test Connection.
4. Click Add.

Configure a resource-based service: HDFS

How to add an HDFS service.

Procedure

1. On Service Manager Resource Policies , click Add New Service ( ) next to HDFS.

   The Create Service page appears.

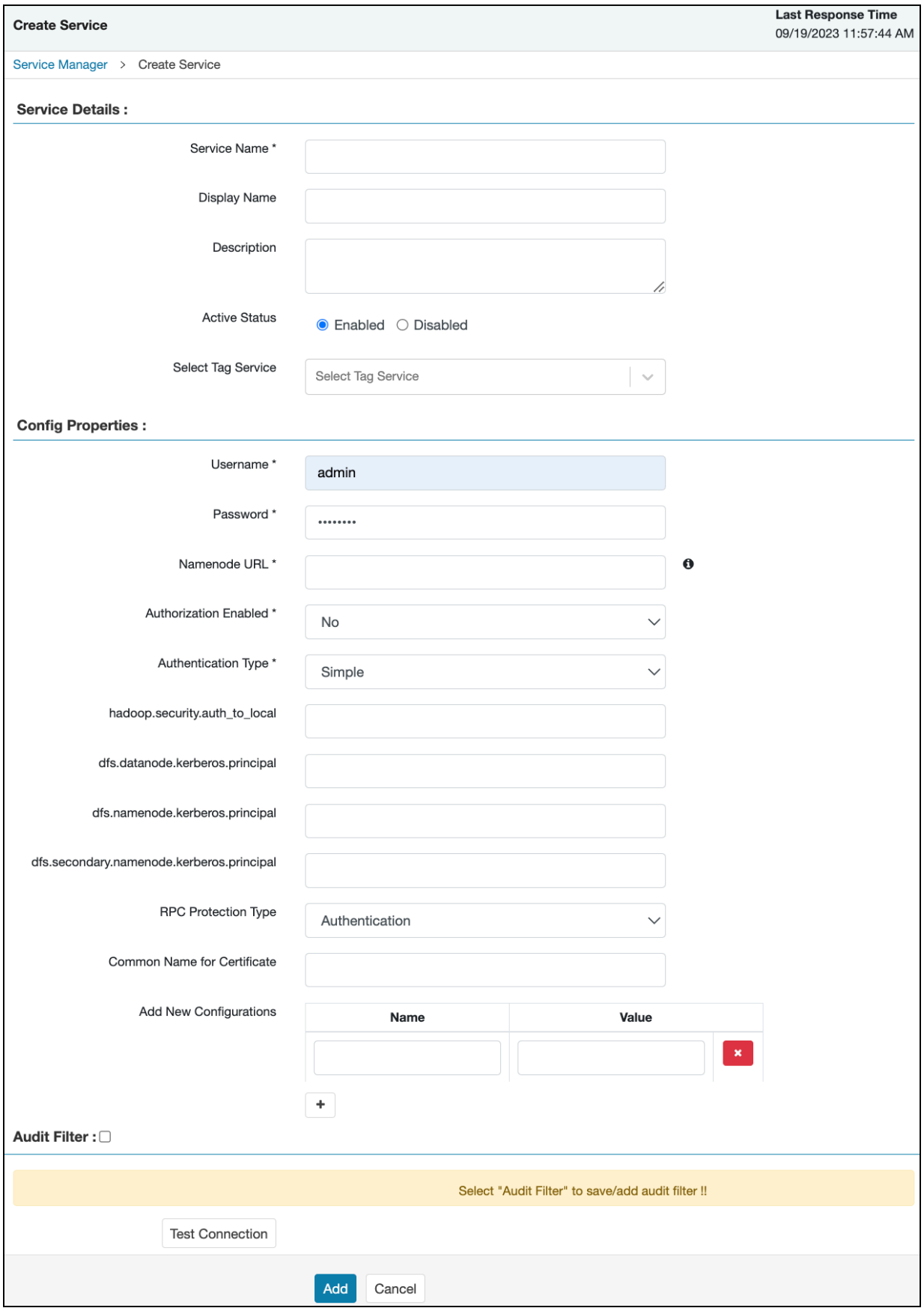
Create Service
Last Response Time
09/19/2023 11:57:44 AM
Service Manager > Create Service
Service Details :
Service Name *
Display Name
Description
Active Status
Enabled
Disabled
Select Tag Service
Select Tag Service
Config Properties :
Username *
admin
Password *
Namenode URL *
Authorization Enabled *
No
Authentication Type *
Simple
hadoop.security.auth_to_local
dfs.datanode.kerberos.principal
dfs.namenode.kerberos.principal
dfs.secondary.namenode.kerberos.principal
RPC Protection Type
Authentication
Common Name for Certificate
Add New Configurations
Name
Value
Audit Filter :
Select "Audit Filter" to save/add audit filter !!
Test Connection
Add
Cancel

2. On Create Service, enter the following information:

Table 8: Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Display Name	The name which will appear on Service Manager.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to HDFS.

Table 9: Configuration Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
NameNode URL	hdfs://NAMENODE_FQDN:8020 The location of the Hadoop HDFS service, as noted in the hadoop configuration file core-site.xml OR (if this is a HA environment) the path for the primary NameNode. This field was formerly named fs.defaultFS.
Authorization Enabled	Authorization involves restricting access to resources. If enabled, user need authorization credentials.
Authentication Type	The type of authorization in use, as noted in the hadoop configuration file core-site.xml; either simple or Kerberos. (Required only if authorization is enabled). This field was formerly named hadoop.security.authorization.
hadoop.security.auth_to_local	Maps the login credential to a username with Hadoop; use the value noted in the hadoop configuration file, core-site.xml.
dfs.datanode.kerberos.principal	The principal associated with the datanode where the service resides, as noted in the hadoop configuration file hdfs-site.xml. (Required only if Kerberos authentication is enabled).
dfs.namenode.kerberos.principal	The principal associated with the NameNode where the service resides, as noted in the hadoop configuration file hdfs-site.xml. (Required only if Kerberos authentication is enabled).
dfs.secondary.namenode.kerberos.principal	The principal associated with the secondary NameNode where the service resides, as noted in the hadoop configuration file hdfs-site.xml. (Required only if Kerberos authentication is enabled).
RPC Protection Type	Only authorised user can view, use, and contribute to a dataset. A list of protection values for secured SASL connections. Values: Authentication, Integrity, Privacy
Common Name For Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click Test Connection.
4. Click Add.

Configure a resource-based service: HadoopSQL

How to add a HadoopSQL service.

Procedure

1. On Service Manager Resource Policies , click Add New Service ( ) next to HadoopSQL.

   Create Service appears.

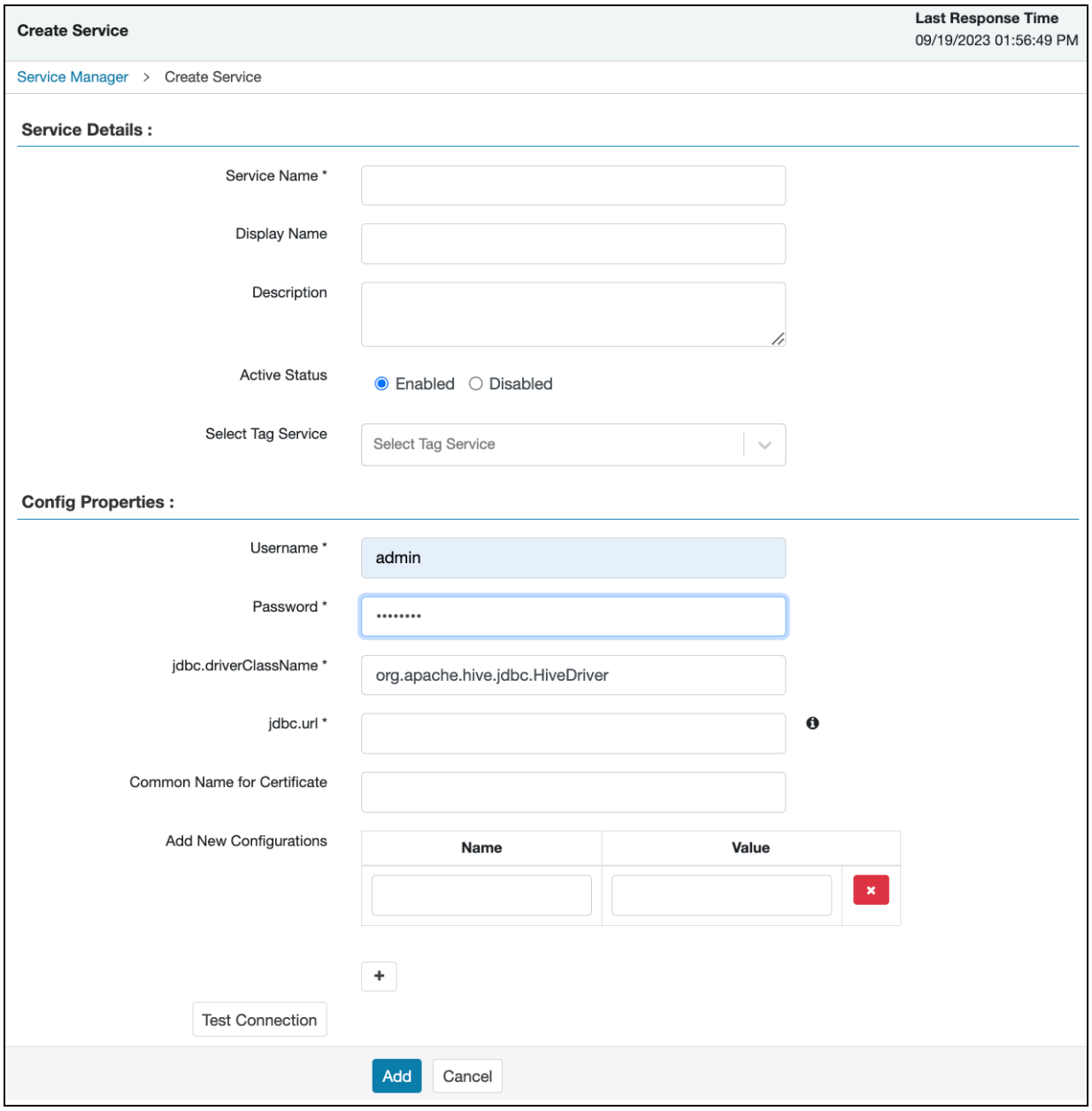

2. On Create Service, enter the following information:

Table 10: Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Display Name	The name which will appear on Service Manager.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Hive.

Table 11: Configuration Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
jdbc.driver ClassName	The full classname of the driver used for Hive connections. Default: org.apache.hive.jdbc.HiveDriver
jdbc.url	The complete connection URL, including port and database name. (Default port: 10000.) For example, on the sandbox, jdbc:hive2://sandbox:10000/.
Common Name For Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click Test Connection.
4. Click Add.

Configure a resource-based service: Kafka

How to add a Kafka service.

Procedure

1. On Service Manager Resource Policies , click Add New Service ( ) next to Kafka.

   The Create Service page appears.

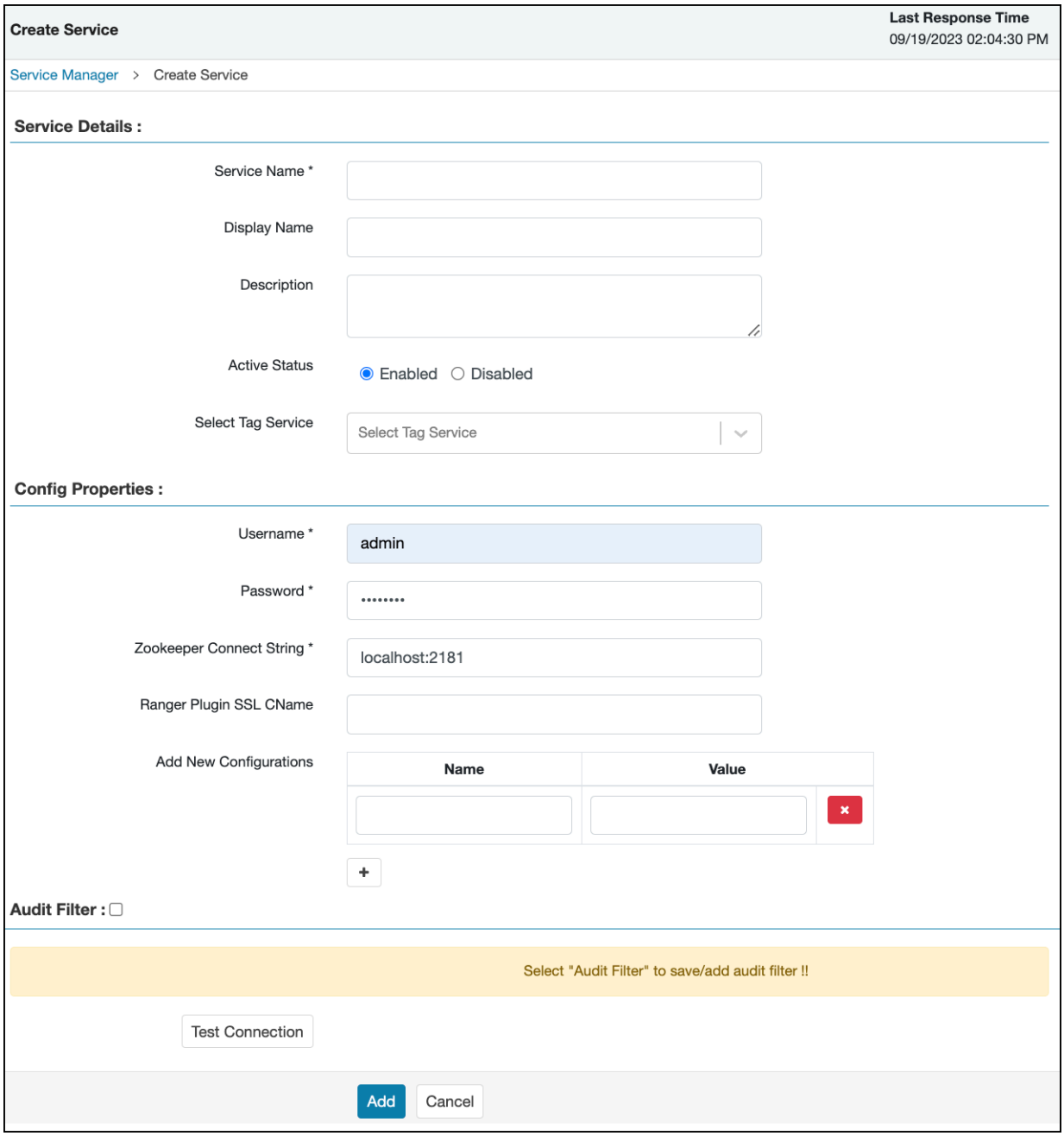

2. On Create Service, enter the following information:

Table 12: Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Display Name	The name which will appear on Service Manager.

Field name	Description
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Kafka.

Table 13: Configuration Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
ZooKeeper Connect String	Defaults to localhost:2181 (Provide FQDN of zookeeper host : 2181).
Ranger Plugin SSL CName	Provide common.name.for.certificate which is registered with Ranger (in Wire Encryption environment). This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).
For non-SSL environment + Kerberos - Update in the Ranger plugin services to get resource lookup working.	
bootstrap.servers	The value is host1:9092,host2:9092,host3:9092.
security.protocol	The value is SASL_PLAINTEXT.
sasl.mechanism	The value is GSSAPI.
kafka.keytab	The value is <path to keytab>. **Note:** It should be present in all the Ranger admin nodes.
kafka.principal	The value is kafka@<REALM>.
For SSL environment + Kerberos - Update in the Ranger plugin services to get resource lookup working.	
bootstrap.servers	The value is host1:9093,host2:9093,host3:9093.
security.protocol	The value is SASL_SSL.
sasl.mechanism	The value is GSSAPI.
kafka.keytab	The value is <path to keytab>. **Note:** It should be present in all the Ranger admin nodes.
kafka.principal	The value is kafka@<REALM>.

3. Click Test Connection.
4. Click Add.

Configure a resource-based service: Knox

How to add a Knox service.

Procedure

1. On Service Manager Resource Policies , click Add New Service ( ) next to Knox.

   The Create Service page appears.

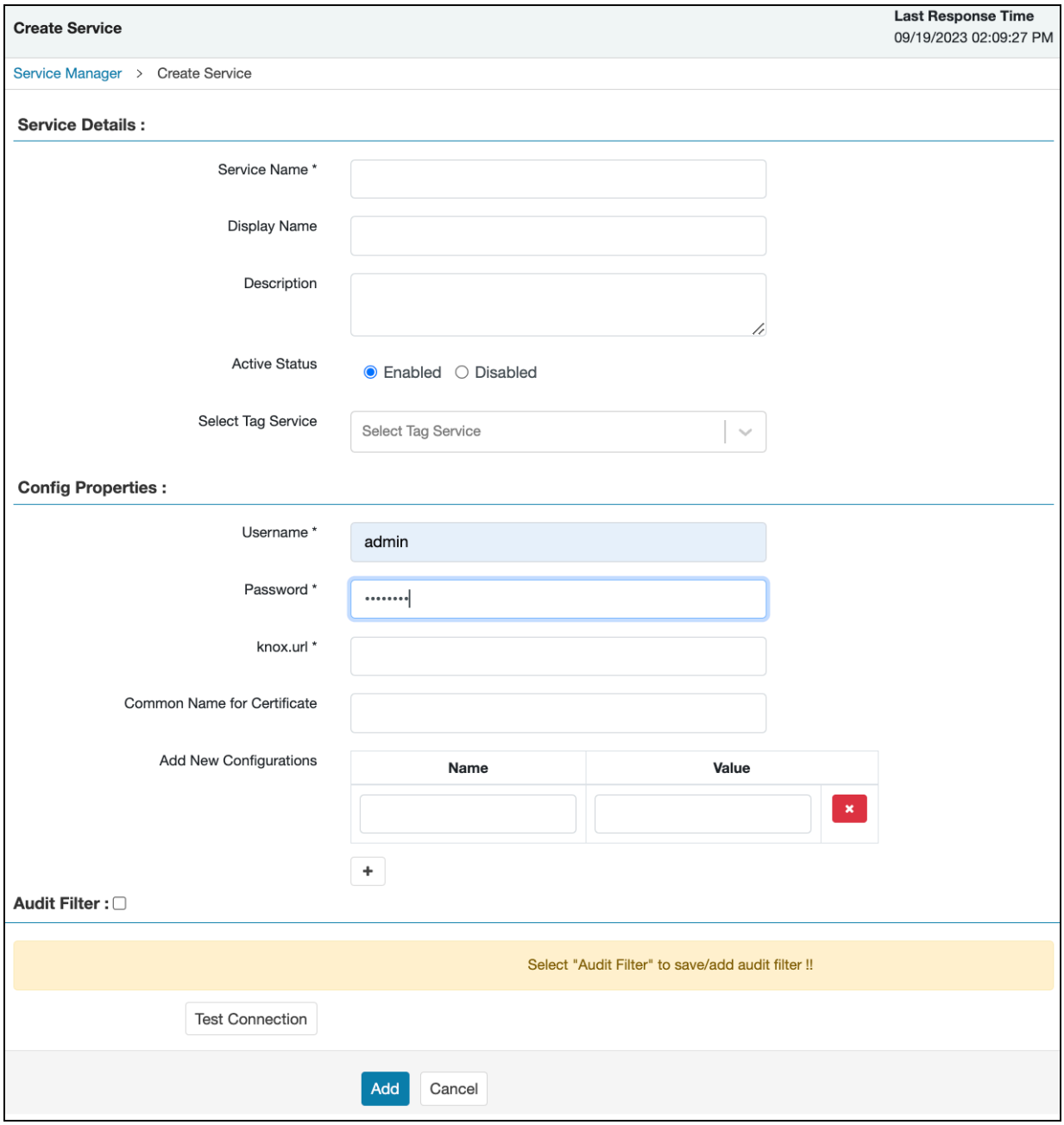

2. On Create Service, enter the following information:

Table 14: Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Display Name	The name which will appear on Service Manager.

Field name	Description
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Knox.

Table 15: Configuration Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
knox.url	The Gateway URL for Knox.
Common Name For Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click Test Connection.
4. Click Add.

Configure a resource-based service: NiFi

How to add a NiFi service.

Procedure

1. On Service Manager Resource Policies , click Add New Service ( ) next to NiFi.
   The Create Service page appears.

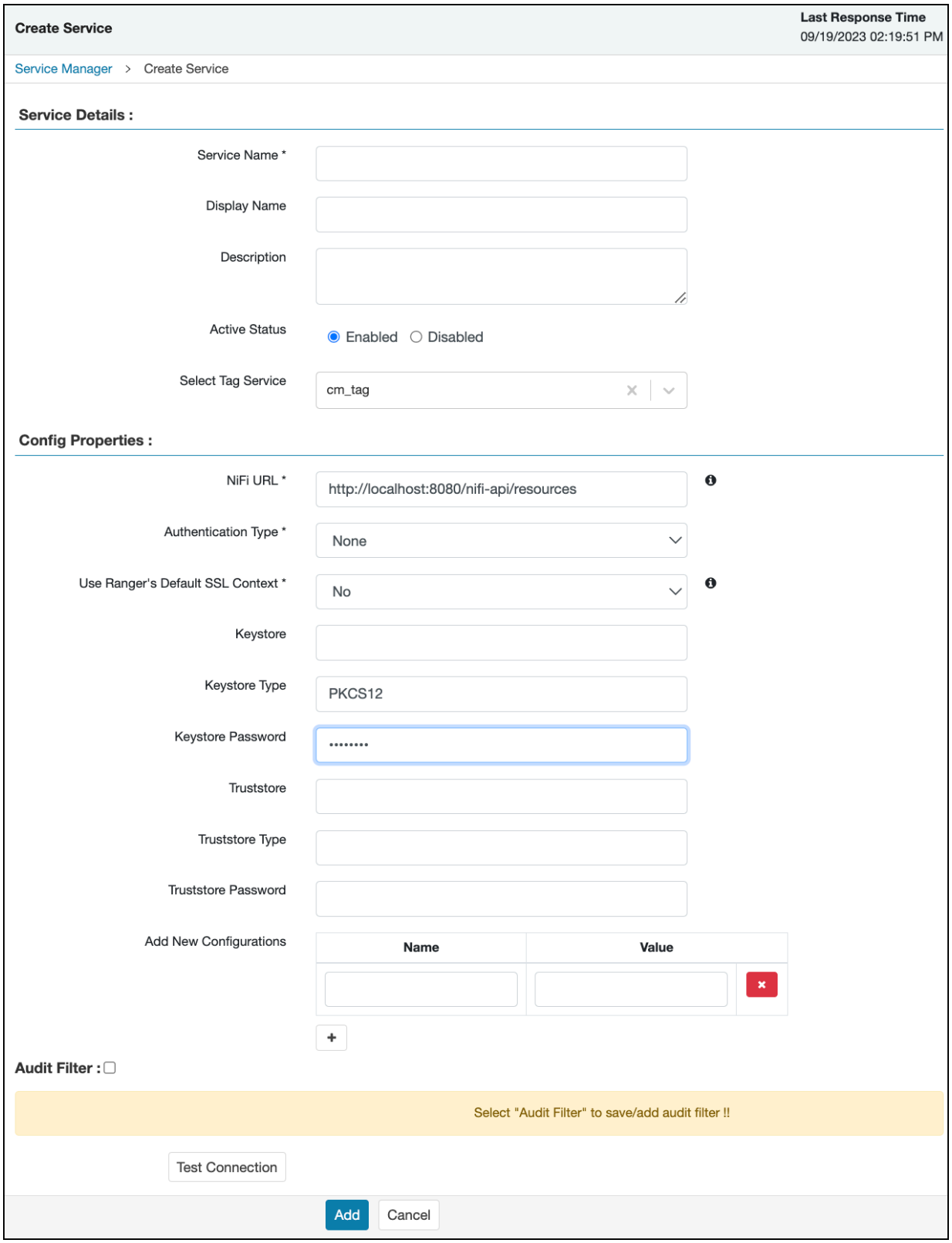

2. On Create Service, enter the following information:

Table 16: Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Display Name	The name which will appear on Service Manager.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to NiFi.

Table 17: Configuration Properties

Field name	Description
NiFi URL	The complete NiFi host URL.
Authentication Type	None or SSL.
Keystore	The keystore to use when Ranger makes an https connection to NiFi. This keystore contains the certificate that represents the Ranger server.
Keystore Type	The keystore type (JKS or PKCS12).
Keystore Password	The keystore password.
Truststore	The truststore to use when Ranger makes an https connection to NiFi. This truststore contains the public key of the certificate authority that signed the NiFi server certificates.
Truststore Type	The truststore type (JKS or PKCS12).
Truststore Password	The truststore password.
Add New Configurations	Add any other new configuration(s).

3. Click Test Connection.
4. Click Add.

Configure a resource-based service: NiFi Registry

How to add a NiFi Registry service.

Procedure

1. On Service Manager Resource Policies , click Add New Service ( ) next to NiFi Registry.
   The Create Service page appears.

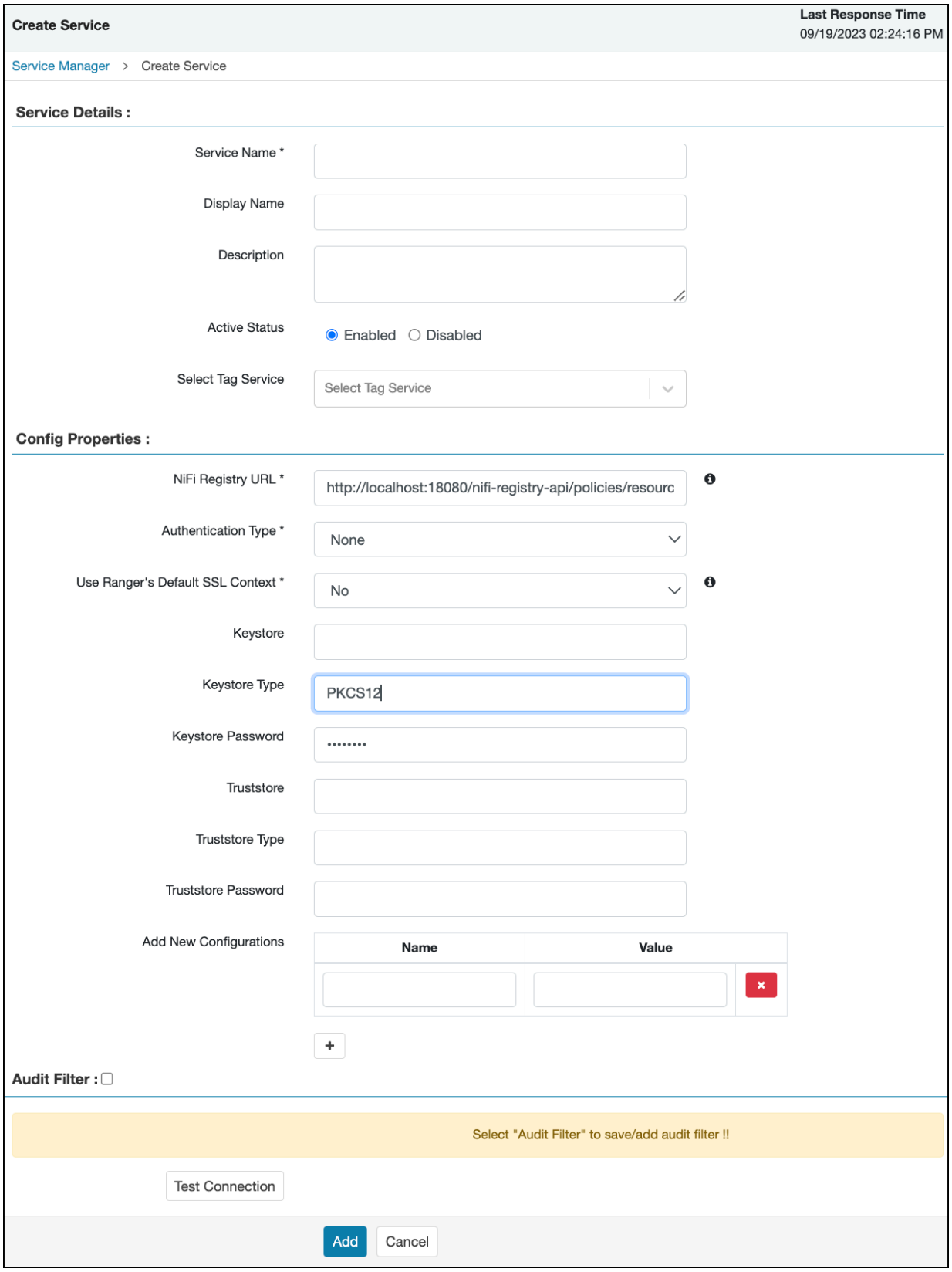

2. On Create Service, enter the following information:

Table 18: Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Display Name	The name which will appear on Service Manager.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to NiFi.

Table 19: Configuration Properties

Field name	Description
NiFi Registry URL	The complete NiFi Registry URL.
Authentication Type	None or SSL.
Keystore	The keystore to use when Ranger makes an https connection to the NiFi Registry. This keystore contains the certificate that represents the Ranger server.
Keystore Type	The keystore type (JKS or PKCS12).
Keystore Password	The keystore password.
Truststore	The truststore to use when Ranger makes an https connection to the NiFi Registry. This truststore contains the public key of the certificate authority that signed the NiFi server certificates.
Truststore Type	The truststore type (JKS or PKCS12).
Truststore Password	The truststore password.
Add New Configurations	Add any other new configuration(s).

3. Click Test Connection.
4. Click Add.

Configure a resource-based service: Solr

How to add a Solr service.

Procedure

1. On Service Manager Resource Policies , click Add New Service ( ) next to Solr.

   The Create Service page appears.

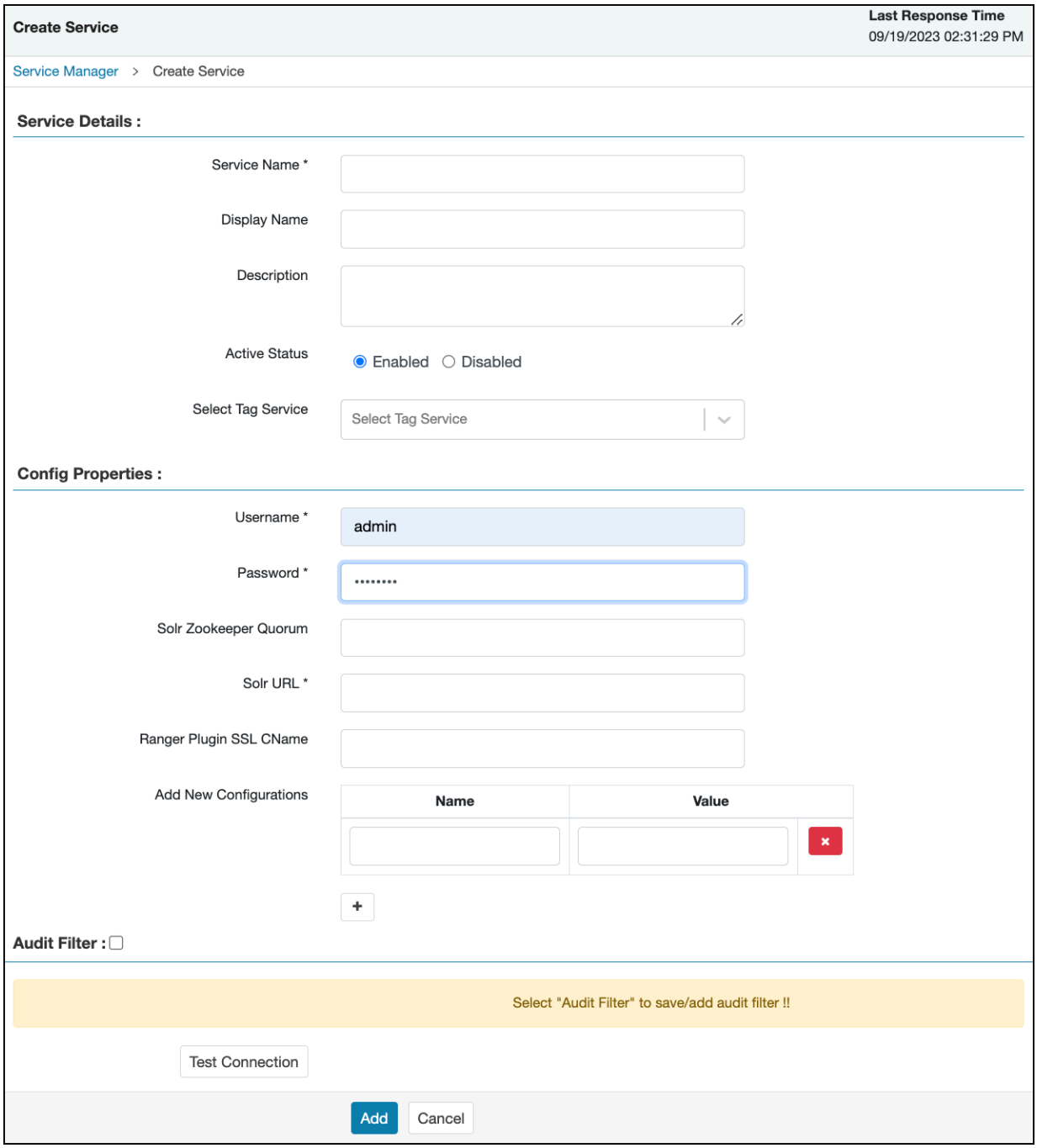

2. On Create Service, enter the following information:

Table 20: Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.

Field name	Description
Display Name	The name which will appear on Service Manager.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Solr.

Table 21: Configuration Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
Solr URL	http://Solr_host:8983
Ranger Plugin SSL CName	Provide common.name.for.certificate which is registered with Ranger (in Wire Encryption environment). This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click Test Connection.
4. Click Add.

Configure a resource-based service: YARN

How to add a YARN service.

Procedure

1. On Service Manager Resource Policies , click Add New Service ( ) next to YARN.

   The Create Service page appears.

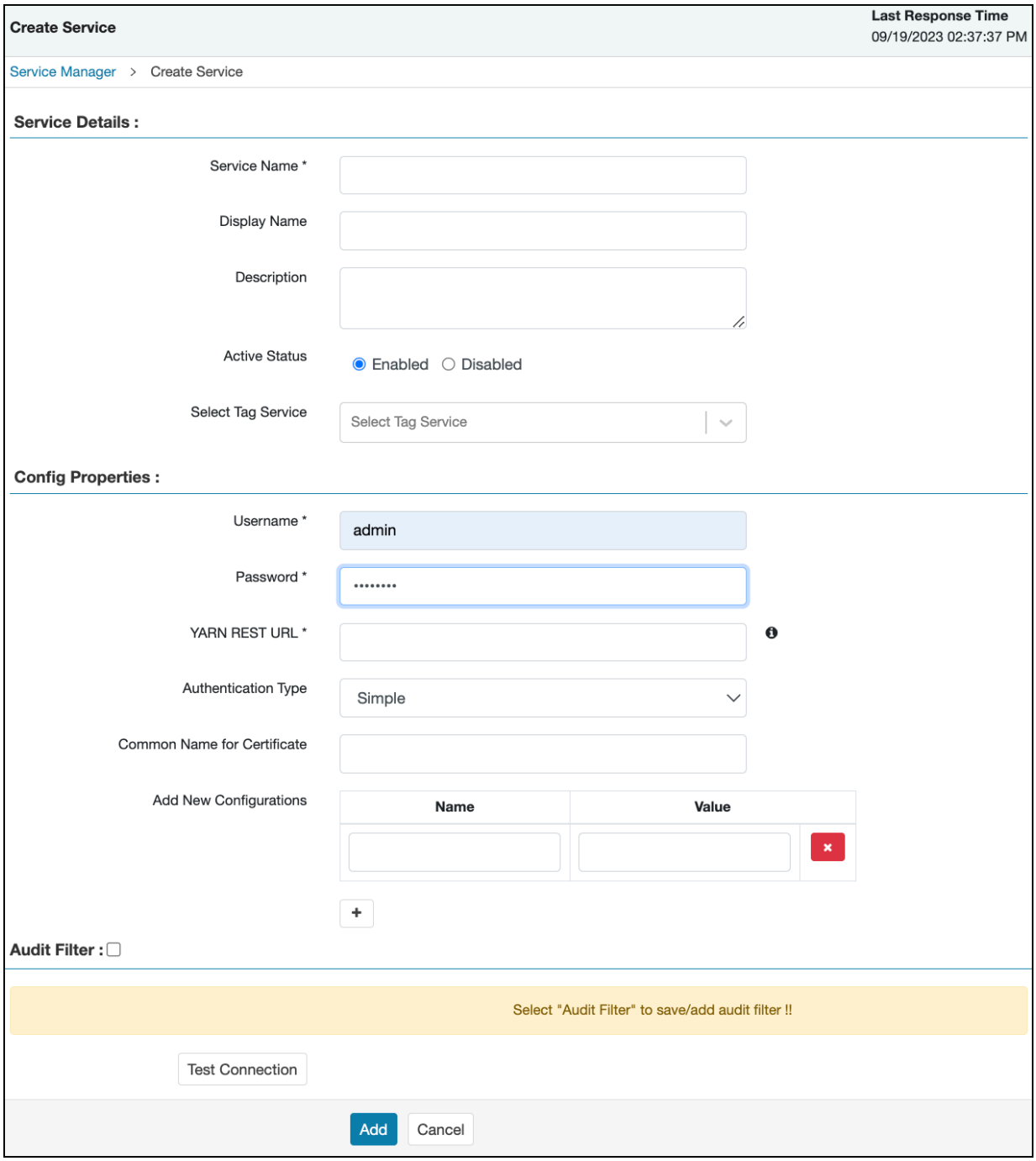

2. On Create Service, enter the following information:

Table 22: Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.

Field name	Description
Display Name	The name which will appear on Service Manager.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to YARN.

Table 23: Configuration Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
YARN REST URL	http://<RESOURCEMANAGER_FQDN>:<http-port> https://<RESOURCEMANAGER_FQDN>:<https-port>
Authentication Type	The type of authorization in use, as noted in the hadoop configuration file core-site.xml; either simple or Kerberos. (Required only if authorization is enabled). This field was formerly named hadoop.security.authorization.
Common Name For Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click Test Connection.
4. Click Add.

Configuring resource-based policies

To view the policies associated with a service, click the service name on Service Manager Resource Policies . List of Policies displays the list of existing policies for that service, along with a search box.

- To add a new resource-based policy to the service, click Add New Policy.
- To edit a resource-based policy, click Edit ( ) for the policy. Edit the policy settings, then click Save to save your changes.
- To delete a resource-based policy, click Delete ( ) for the policy.

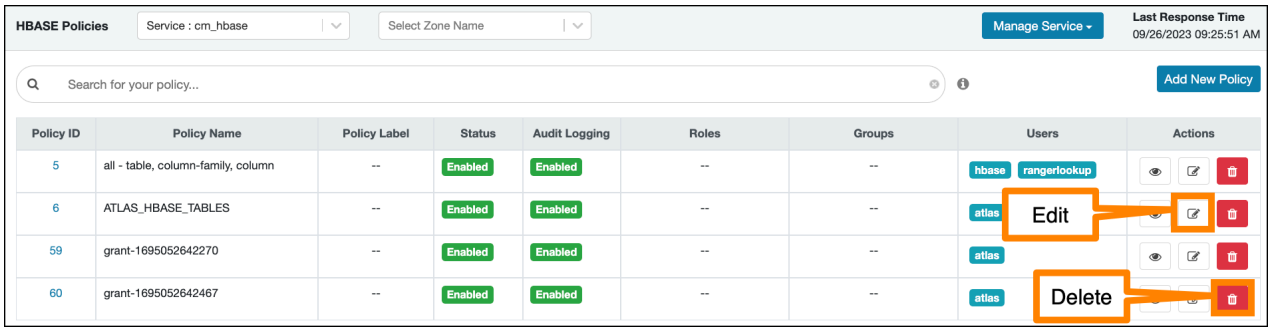

Related Information

Importing and exporting resource-based policies

Configure a resource-based policy: Atlas

How to add a new policy to an existing Atlas service.

Procedure

1. On Service Manager, select an existing Atlas service.

   List of Policies displays a list of the policies defined for Atlas service.

2. Click Add New Policy.
   Create Policy displays controls for creating details for a new policy.

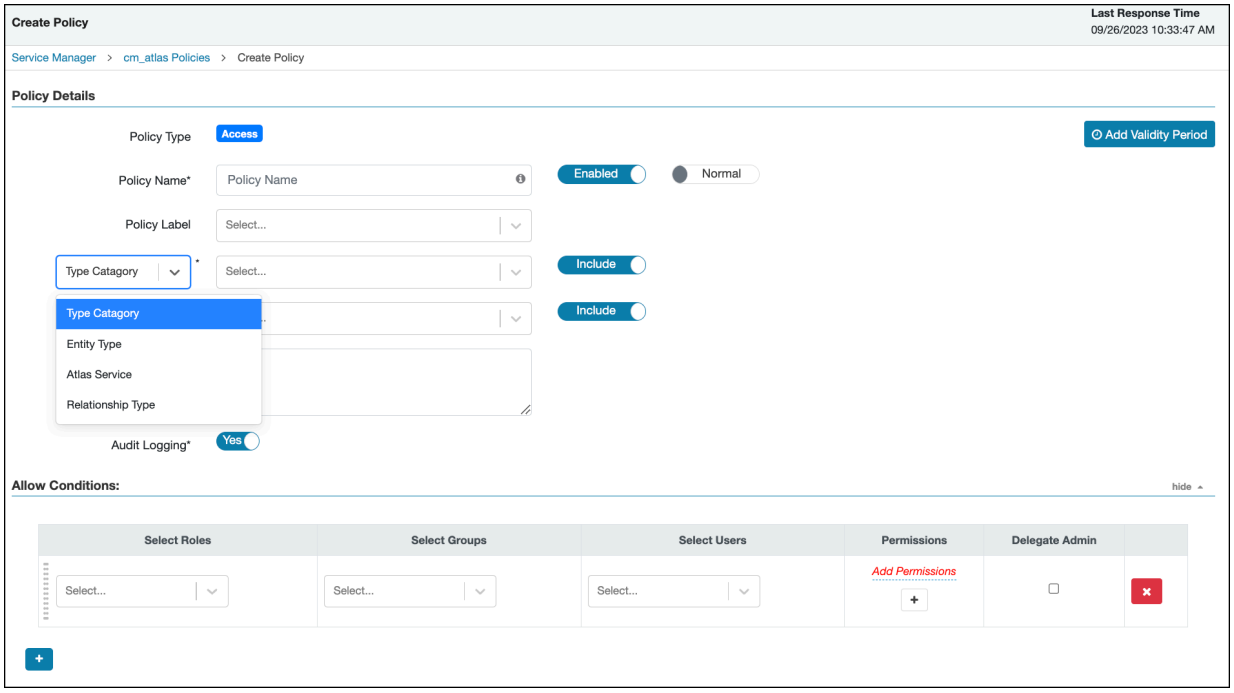

3. Complete the Create Policy page as follows:

Table 24: Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
type-category	Select type-category, entity-type, atlas-service, or relationship-type.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.

Field	Description
Add Validity Period	Specify a start and end time for the policy.

Table 25: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Permissions	Add or edit permissions: Create Type, Update Type, Delete Type, Select/Deselect All.
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

Related Information

Wildcards and variables in resource-based policies

Configure a resource-based policy: HBase

How to add a new policy to an existing HBase service.

Procedure

1. On Service Manager, select an existing HBase service.

   List of Policies displays a list of the policies defined for Hbase service.

2. Click Add New Policy.

   Create Policy displays controls for creating details for a new policy.

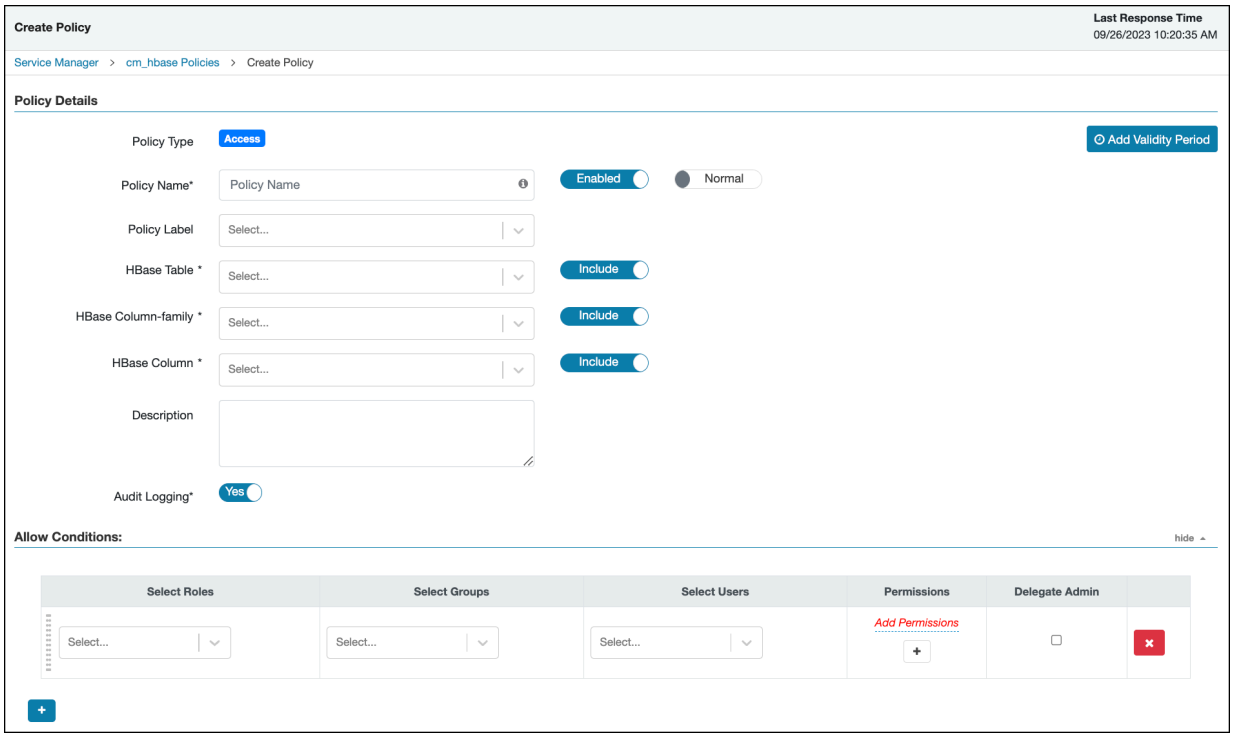

3. Edit fields on Create Policy, as follows:

Table 26: Policy Details

Label	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
HBase Table	Select the appropriate database. Multiple databases can be selected for a particular policy. This field is mandatory. **Note:** Policies must use '<namespace>:*' for the table name to apply to the namespace as a whole. You can define a namespace in the HBase table field. Valid formats for a namespace-specific, HBase policy include: <namespace>:<table> for example, "myNameSpace:table1" Further, note that <namespace>:<tablePrefix>* (default value) does not work, per https://issues.apache.org/jira/browse/RANGER-1226. All other namespaces except the default one work.
HBase Column-family	For the selected table, specify the column families to which the policy applies.
HBase Column	For the selected table and column families, specify the columns to which the policy applies.
Description	(Optional) Describe the purpose of the policy.

Label	Description
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.
Add Validity Period	Specify a start and end time for the policy.

Table 27: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Permissions	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

What to do next

Provide User Access to HBase Database Tables from the Command Line:

HBase provides the means to manage user access to HBase database tables directly from the command line. The most commonly-used commands are:

- GRANT

  Syntax:

  ```
  grant '<user-or-group>','<permissions>','<table>
  ```

  For example, to create a policy that grants user1 read/write permission on the table usertable, the command would be:

  ```
  grant 'user1','RW','usertable'
  ```

  The syntax is the same for granting CREATE and ADMIN rights.
- REVOKE

  Syntax:

  ```
  revoke '<user-or-group>','<usertable>'
  ```

  For example, to revoke the read/write access of user1 to the table usertable, the command would be:

  ```
  revoke 'user1','usertable'
  ```

  Note:

  Unlike Hive, HBase has no specific revoke commands for each user privilege.

Related Information

Wildcards and variables in resource-based policies

Configure a resource-based policy: HDFS

How to add a new policy to an existing HDFS service.

About this task

Through configuration, Apache Ranger enables both Ranger policies and HDFS permissions to be checked for a user request. When the NameNode receives a user request, the Ranger plugin checks for policies set through the Ranger Service Manager. If there are no policies, the Ranger plugin checks for permissions set in HDFS.

We recommend that permissions be created using Service Manager Resource Policies , and to have restrictive permissions at the HDFS level.

Procedure

1. On Service Manager Resource Policies , select an existing HDFS service.

   List of Policies displays a list of the policies defined for HDFS service.

2. Click Add New Policy.
   Create Policy displays controls for creating details for a new policy.

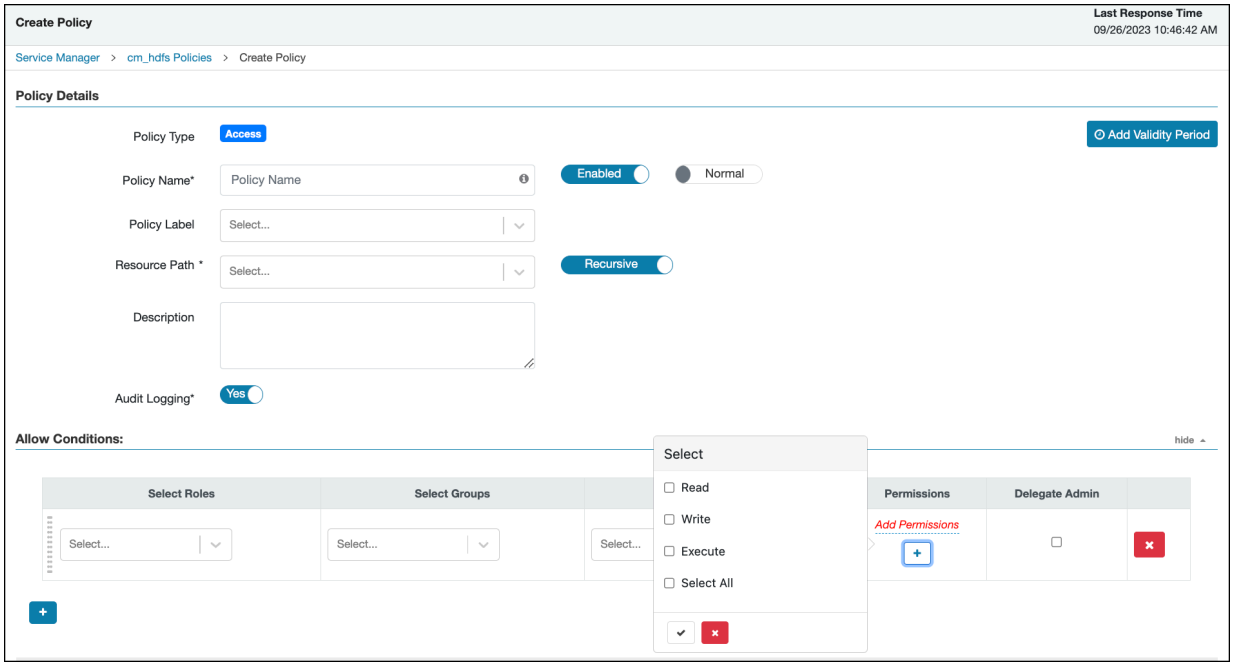

3. Complete the Create Policy page as follows:

Table 28: Policy Details

Field	Description
Policy Name	Enter a unique name for this policy. The name cannot be duplicated anywhere in the system.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
Resource Path	Define the resource path for the policy folder/file. The default Recursive setting specifies that the resource path is recursive; you can also specify a non-recursive path.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.
Add Validity Period	Specify a start and end time for the policy.

Table 29: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

Label	Description
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Permissions	Add or edit permissions: Read, Write, Execute, Select/Deselect All.
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

Related Information

Wildcards and variables in resource-based policies

Configure a resource-based policy: HadoopSQL

How to add a new policy to an existing Hive service.

Procedure

1. On Service Manager Resource Policies , select an existing HadoopSQL service.

   List of Policies displays a list of the policies defined for HadoopSQL service.

   Note: Service_name remains cm_hive. Display name is HadoopSQL.

2. Click Add New Policy.

   Create Policy displays controls for creating details for a new policy.

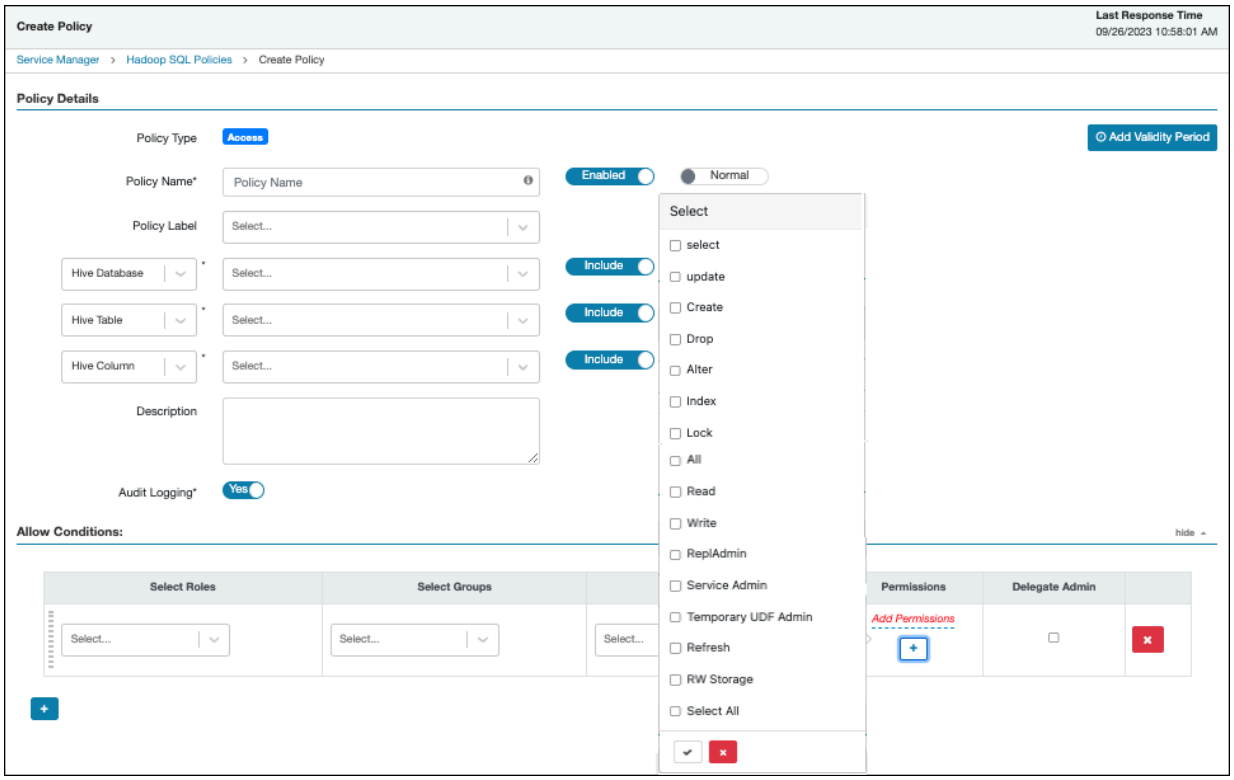

3. Complete the Create Policy page as follows:

Table 30: Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory. The policy is enabled by default.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
Database	Type in the applicable database name. The autocomplete feature displays available databases based on the entered text. Include is selected by default to allow access. Select Exclude to deny access..
table/udf	Specifies a table-based or UDF-based policy. Select table or udf, then type in the applicable table or UDF name. The autocomplete feature displays available tables based on the entered text. Include is selected by default to allow access. Select Exclude to deny access.
column	Type in the applicable column name. The autocomplete feature displays available columns based on the entered text. Include is selected by default to allow access. Select Exclude to deny access.

Field	Description
URL	Specify the cloud storage path (for example s3a://dev-admin/demo/campaigns.txt) where the end-user permission is needed to read/write the Hive data from/to a cloud storage path. Permissions: READ operation on the URL permits the user to perform HiveServer2 operations which use S3 as data source for Hive tables. WRITE operation on the URL permits the user to perform HiveServer2 operations which write data to the specified S3 location.
URI	Hive INSERT OVERWRITE queries require a Ranger URI policy to allow write operations, even if the user has write privilege granted through HDFS policy. Failure to specify this field will result in the following error: Error while compiling statement: FAILED: HiveAccessControlException Permission denied: user [jdoe] does not have [WRITE] privilege on [/tmp/*] (state=42000,code=40000) Example value: /tmp/*
Description	(Optional) Describe the purpose of the policy.
Hive Service Name	hiveservice is used only in conjunction with Permissions=Service Admin. Enables a user who has Service Admin permission in Ranger to run the kill query API: kill query <queryID> . Supported value: *. (Required)
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.
Add Validity Period	Specify a start and end time for the policy.

Table 31: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Permissions	Add or edit permissions: Select, Update, Create, Drop, Alter, Index, Lock, All, ReplAdmin, Service Admin, Temp UDF Admin, Refresh, RW Storage, Select/Deselect All. Service Admin is used in conjunction with Hive Service Name and the kill query API: kill query <queryID> .

Label	Description
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

What to do next

Provide User Access to Hive Database Tables from the Command Line

Hive provides the means to manage user access to Hive database tables directly from the command line. The most commonly-used commands are:

- GRANT

  Syntax:

  ```
  grant <permissions> on table <table> to user <user or group>;
  ```

  For example, to create a policy that grants user1 SELECT permission on the table default-hivesmoke22074, the command would be:

  ```
  grant select on table default.hivesmoke22074 to user user1;
  ```

  The syntax is the same for granting UPDATE, CREATE, DROP, ALTER, INDEX, LOCK, ALL, and ADMIN rights.
- REVOKE

  Syntax:

  ```
  revoke <permissions> on table <table> from user <user or group>;
  ```

  For example, to revoke the SELECT rights of user1 to the table default.hivesmoke22074, the command would be:

  ```
  revoke select on table default.hivesmoke22074 from user user1;
  ```

  The syntax is the same for revoking UPDATE, CREATE, DROP, ALTER, INDEX, LOCK, ALL, and ADMIN rights.

Related Information

Wildcards and variables in resource-based policies

Configure a resource-based storage handler policy: HadoopSQL

How to configure a policy that allows authorized users to create data tables using storage-handlers.

About this task

Ranger includes “storage-type” and “storage-url” resources in HadoopSQL Service that support only the “RW Storage ” permission. Ranger authorizes users to create or alter tables against this resource policy. Users are allowed to create/alter the table in the respective storage if they have the required “RW Storage” permission on the resource representing the storage-type and storage-url .

Procedure

1. On Service Manager Resource Policies , select an existing HadoopSQL service.

   List of Policies displays a list of the policies defined for HadoopSQL service.

   Note: Service_name remains cm_hive. Display name is HadoopSQL.

2. Select Add New Policy to create a new policy.
   a) Within Create Policy, select storage-type as shown in the following example:

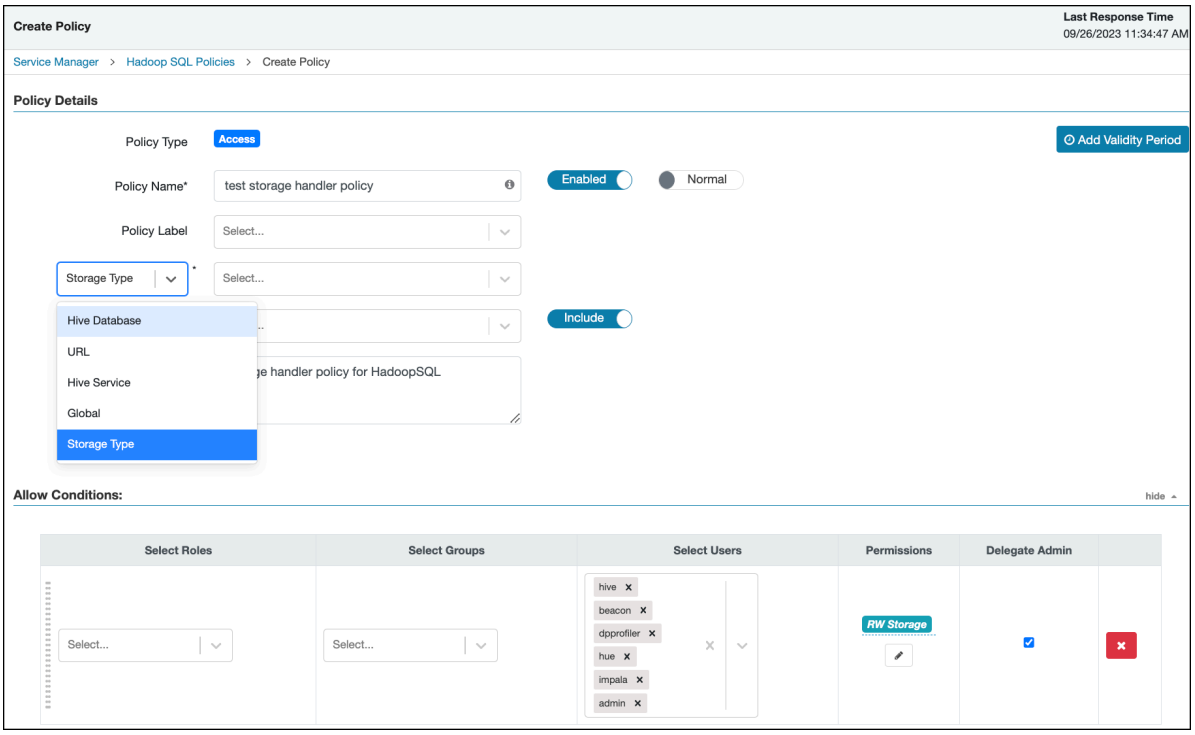

   b) Complete the required* fields.
   c) Under Allow Conditions, select users and add the RW Storage permission, as shown in the preceeding example.
   d) Scroll to the bottom of Create Policy, then click Add.

3. To configure an existing policy named all - storage-type, storage-url, click Edit.

   The Edit Policy page appears.

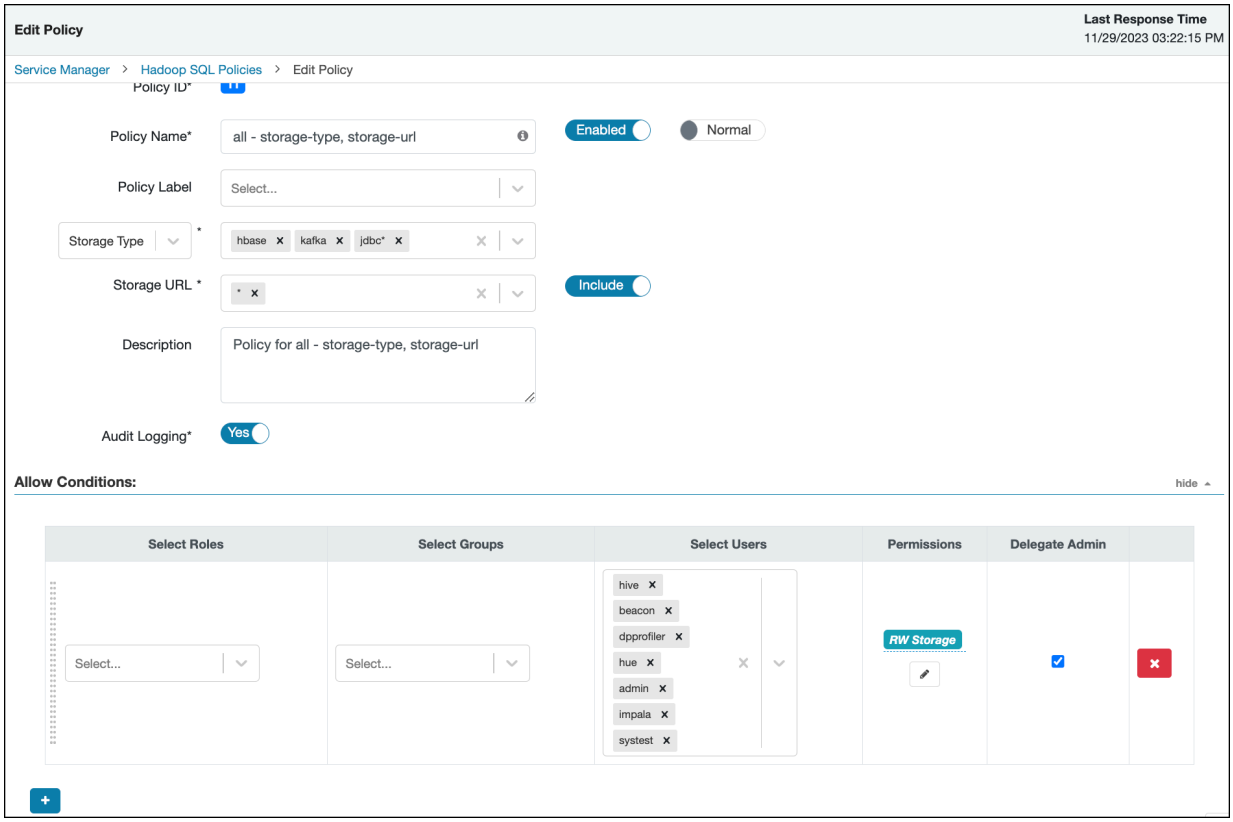

4. Complete the Edit Policy page as shown in the preceding example using the follow policy detail descriptions:

Table 32: Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory. The policy is enabled by default.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.
storage-type	Type in the applicable storage type. * allows athorizes users to create any table in the spcified storage type..
storage url	Type in the applicable storage URL. * allows athorizes users to create any table in the spcified storage URL. Select Exclude to deny access.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Field	Description
Add Validity Period	Specify a start and end time for the policy.

Table 33: Allow Conditions

Label	Description
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Permissions	Add or edit permissions: RW Storage, You can assign read and select permissions to rangerlookup user.
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

Example

Example StorageHandler Policy Definitions and Use Cases:

HBase StorageHandler policy:

Storage Type: hbase

Storage URL: hbase-cluster:port/hbase-table

Storage create table command:

```
CREATE [EXTERNAL] table foo(…)
STORED BY‘org.apache.hadoop.hive.hbase.HBaseStorageHandler’
TBLPROPERTIES (‘hbase.table.name’ = ‘bar’);
```

```
e.g:
CREATE TABLE hive_hbase_test_1(key int, value string) STORED BY
 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPR
OPERTIES ("hbase.columns.mapping" = "cf:string", "hbase.table.na
me" = "hbase_test_1");
```

Iceberg StorageHandler policy:

Storage type: iceberg

Storage URL: DBname/Table* , or

Storage URL: DBname/*

JDBC StorageHandler policy:

Storage Type: jdbc:mysql

Storage URL: mysql-host:port/DBname/Table , or

Storage URL: mysql-host:port/DBname/*

Note:

Policy and table definitions must be in sync regarding the port definition, even for default port numbers. For example, if port number 3306 is defined in the policy for mysql and this port number is left out from the URL as default value for the JDBC Driver, you must use the same reference as defined in the policy when creating the external table.

Using an explicit table name allows only to reference that specific table with hive.sql.table while using * allows not only to reference any tables from the database but also allows you to write a custom query against this database, for example using hive.sql.query.

Storage create table command:

```
CREATE [EXTERNAL] TABLE student_jdbc
(
  name string,
  age int,
  gpa double
)
STORED BY 'org.apache.hive.storage.jdbc.JdbcStorageHandler'
TBLPROPERTIES (
    "hive.sql.database.type" = "MYSQL",
    "hive.sql.jdbc.driver" = "com.mysql.jdbc.Driver",
    "hive.sql.jdbc.url" = "jdbc:mysql://localhost/sample",
    "hive.sql.dbcp.username" = "hive",
    "hive.sql.dbcp.password" = "hive",
    "hive.sql.table" = "STUDENT",
    "hive.sql.dbcp.maxActive" = "1"
);
```

Kafka StorageHandler policy:

Storage Type: kafka

Storage URL: bootstrap-server:port/kafka-topic

Phoenix StorageHandler policy:

Storage Type: phoenix

Storage URL: phoenix-cluster:port/table-name

Storage create table command:

```
CREATE [EXTERNAL] TABLE phoenix_table (
                            s1 string,
                            i1 int,
                            f1 float,
                            d1 double
                            )
                            STORED BY 'org.apache.phoenix.hive.
PhoenixStorageHandler'
                            TBLPROPERTIES (
                            "phoenix.table.name" = "phoenix_table
",
                            "phoenix.zookeeper.quorum" = "localho
st",
                            "phoenix.zookeeper.znode.parent" = "/
hbase",
                            "phoenix.zookeeper.client.port" =
"2181",
                            "phoenix.rowkeys" = "s1, i1",
                            "phoenix.column.mapping" = "s1:s1,
 i1:i1, f1:f1, d1:d1",
```

```
                                "phoenix.table.options" = "SALT_BUCKE
TS=10, DATA_BLOCK_ENCODING='DIFF'"
                                );
```

Configure a resource-based policy: Kafka

How to add a new policy to an existing Kafka service.

Procedure

1. On Service Manager Resource Policies , select an existing Kafka service.

   List of Policies displays a list of the policies defined for Kafka service.

2. Click Add New Policy.
   Create Policy displays controls for creating details for a new policy.

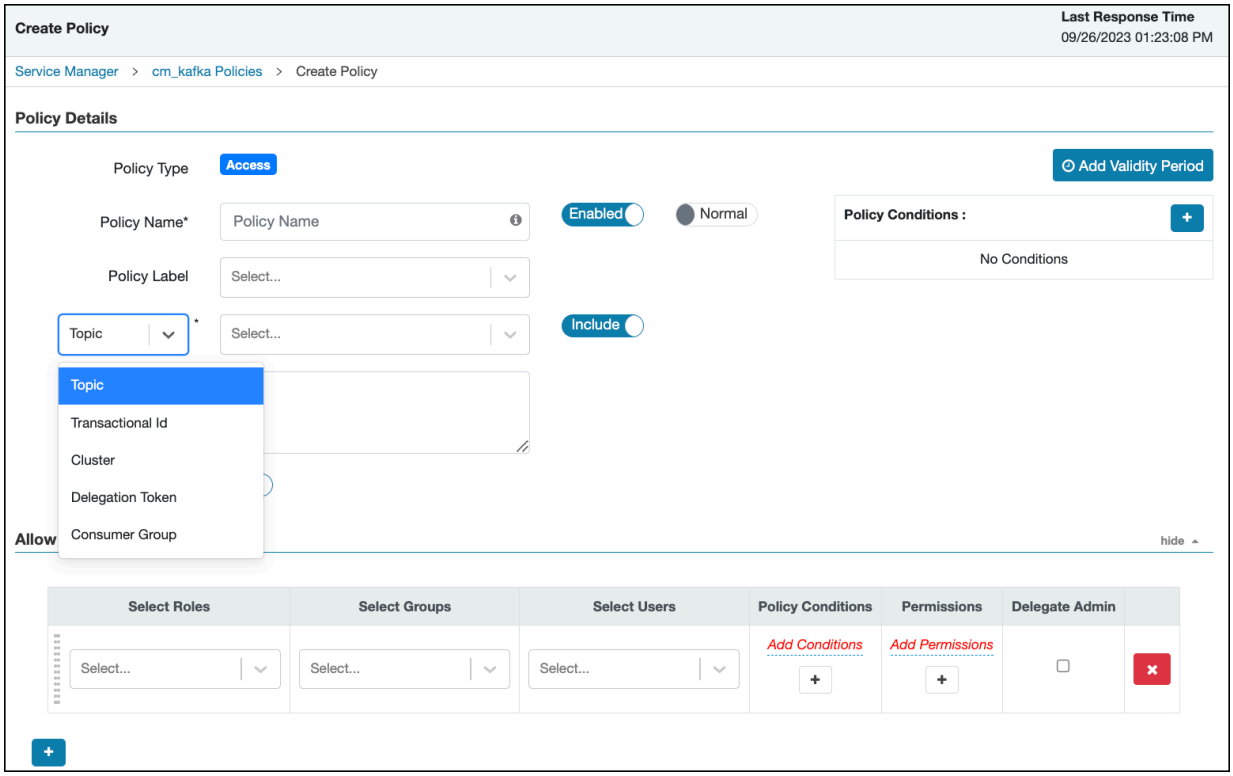

3. Complete the Create Policy page as follows:

Table 34: Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.
Topic	Kafka resource type. A topic is a category or feed name to which messages are published.

Field	Description
Transactional ID	Kafka resource type, uniquely identifies producers in a persistent way.
Cluster	Kafka resource type.
Delegation Token	Kafka resource type for authentication.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Add Validity Period	Specify a start and end time for the policy.
Policy Conditions (applied at the policy level)	Click +, then specify an IP address range.

Table 35: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Policy Conditions (applied at the item level)	Specify an IP address range.
Permissions	Add or edit permissions: Publish, Consume,Configure,Describe, Create, Delete, Describe Configs, Alter Configs, Select/Deselect All.
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

Related Information

Wildcards and variables in resource-based policies

Configure a resource-based policy: Knox

How to add a new policy to an existing Knox service.

Procedure

1. On Service Manager Resource Policies , select an existing Knox service.

   List of Policies displays a list of the policies defined for Knox service.
2. Click Add New Policy.
   Create Policy displays controls for creating details for a new policy.

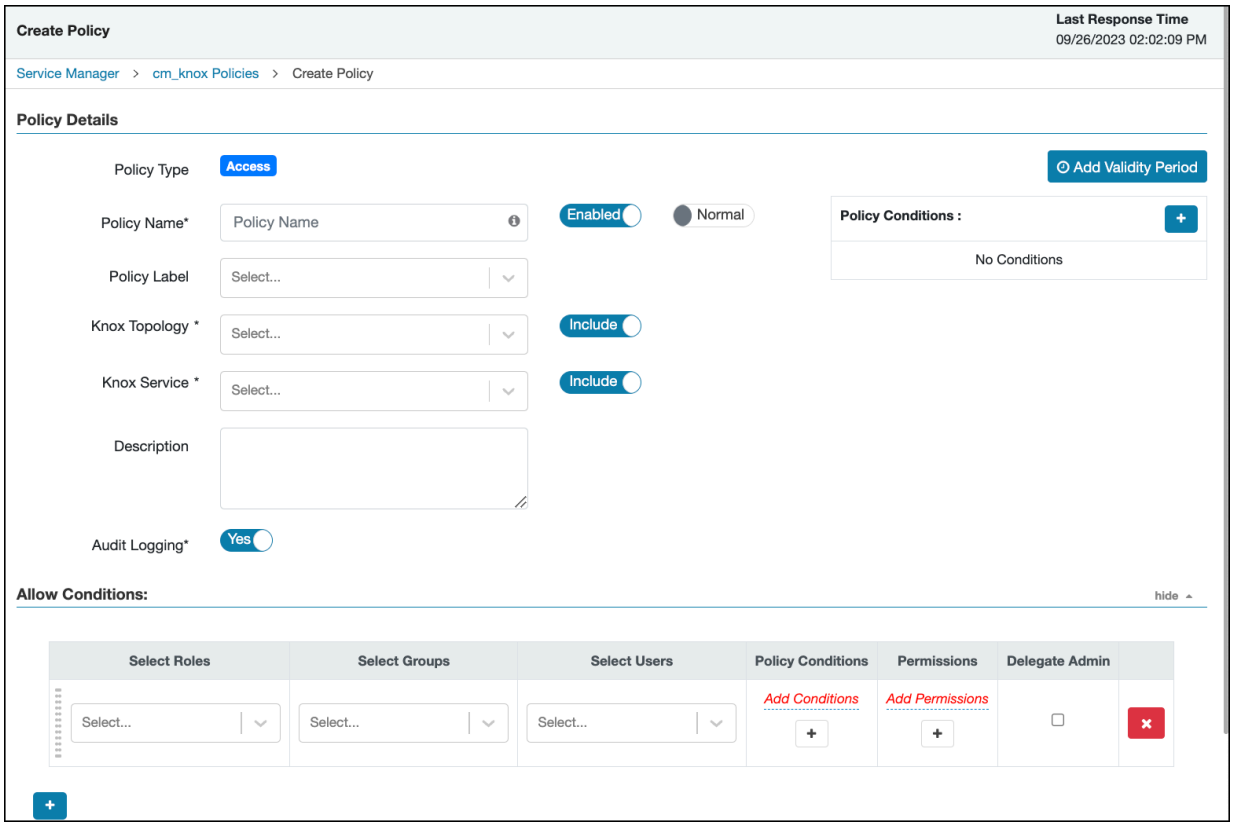

3. Complete the Create Policy page as follows:

Table 36: Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
Knox Topology	Enter an appropriate Topology Name.
Knox Service	Enter an appropriate Service Name.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.
Add Validity Period	Specify a start and end time for the policy.

Field	Description
Policy Conditions (applied at the policy level)	Click +, then specify an IP address range.

Table 37: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Policy Conditions (applied at the item level)	Specify an IP address range.
Permissions	Add or edit permissions: Allow
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

Since Knox does not provide a command line methodology for assigning privileges or roles to users, the User and Group Permissions portion of the Knox Create Policy form is especially important.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

Related Information

Wildcards and variables in resource-based policies

Configure a resource-based policy: NiFi

How to add a new policy to an existing Atlas service.

Procedure

1. On Service Manager Resource Policies , select an existing NiFi service.

   List of Policies displays a list of the policies defined for NiFi service.

2. Click Add New Policy.
   Create Policy displays controls for creating details for a new policy.

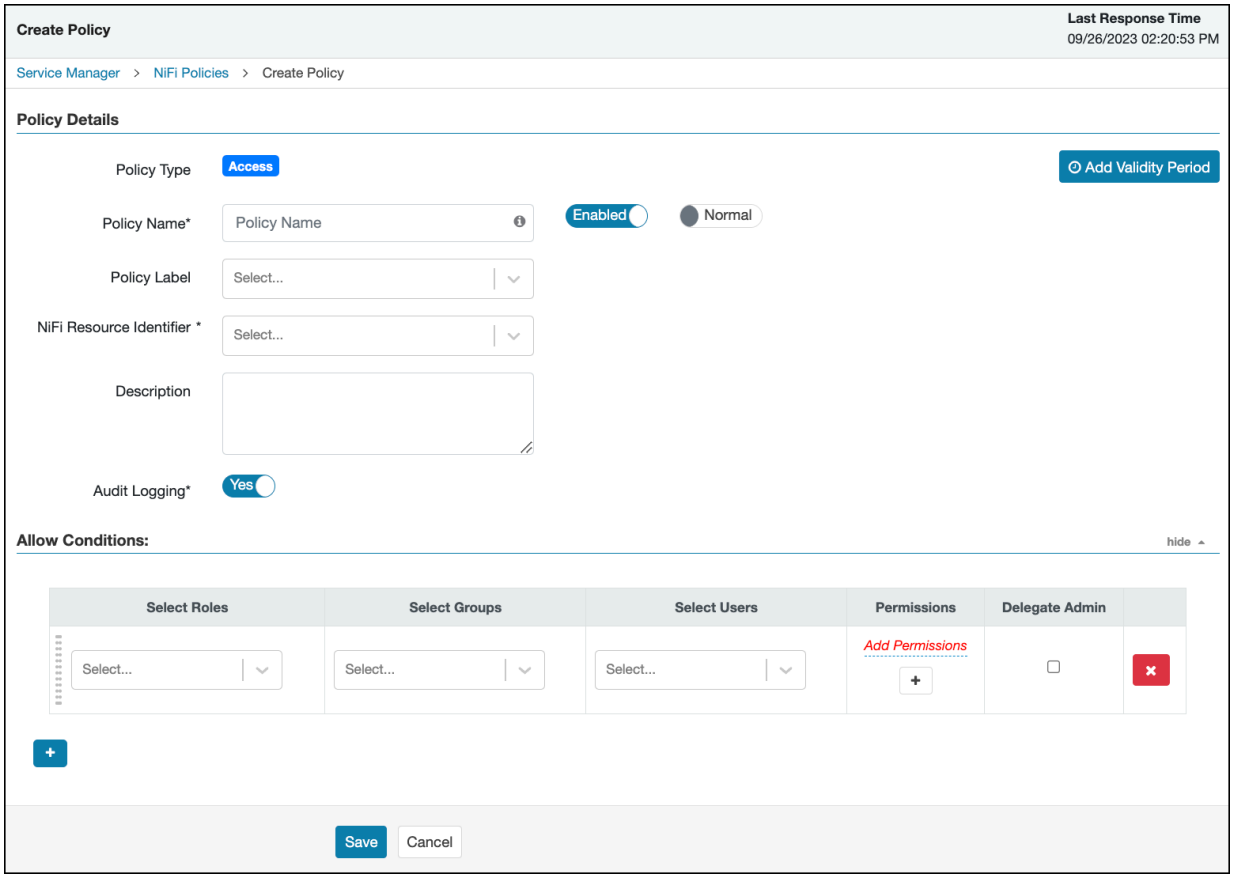

3. Complete the Create Policy page as follows:

Table 38: Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
NiFi Resource Identifier	In a NiFi cluster, all nodes must be granted the ability to view and modify component data in order for user to list or empty queues in processor component outbound connections. With Ranger this can be accomplished by using a wildcard to grant all of the NiFi nodes read and write access to the /data/* NiFi resource.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.

Field	Description
Add Validity Period	Specify a start and end time for the policy.

Table 39: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Permissions	Add or edit permissions: Read, Write, Select/Deselect All.
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

Configure a resource-based policy: NiFi Registry

How to add a new policy to an existing Atlas service.

Procedure

1. On Service Manager Resource Policies , select an existing NiFi Registry service.

   List of Policies displays a list of the policies defined for NiFi Registry service.

2. Click Add New Policy.
   Create Policy displays controls for creating details for a new policy.

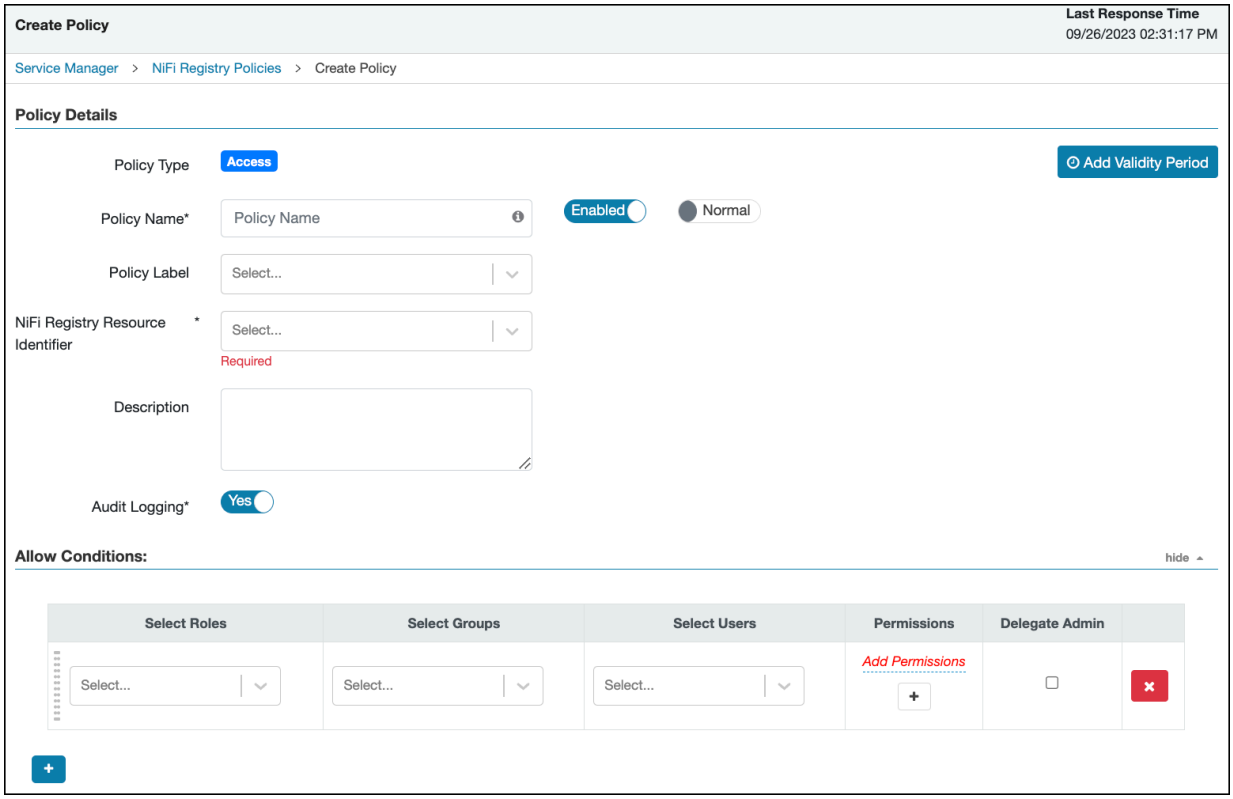

3. Complete the Create Policy page as follows:

Table 40: Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
NiFi Registry Resource Identifier	In a NiFi cluster, all nodes must be granted the ability to view and modify component data in order for user to list or empty queues in processor component outbound connections. With Ranger this can be accomplished by using a wildcard to grant all of the NiFi nodes read and write access to the /data/* NiFi resource.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.

Field	Description
Add Validity Period	Specify a start and end time for the policy.

Table 41: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Permissions	Add or edit permissions: Read, Write, Delete, Select/Deselect All.
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

Related Information

SQL Standard Based Hive Authorization

Configure a resource-based policy: S3

How to add a new policy to an existing S3 service.

Procedure

1. On Service Manager Resource Policies , select an existing S3 service.

   List of Policies displays a list of the policies defined for S3 service.

2. Click Add New Policy.
   Create Policy displays controls for creating details for a new policy

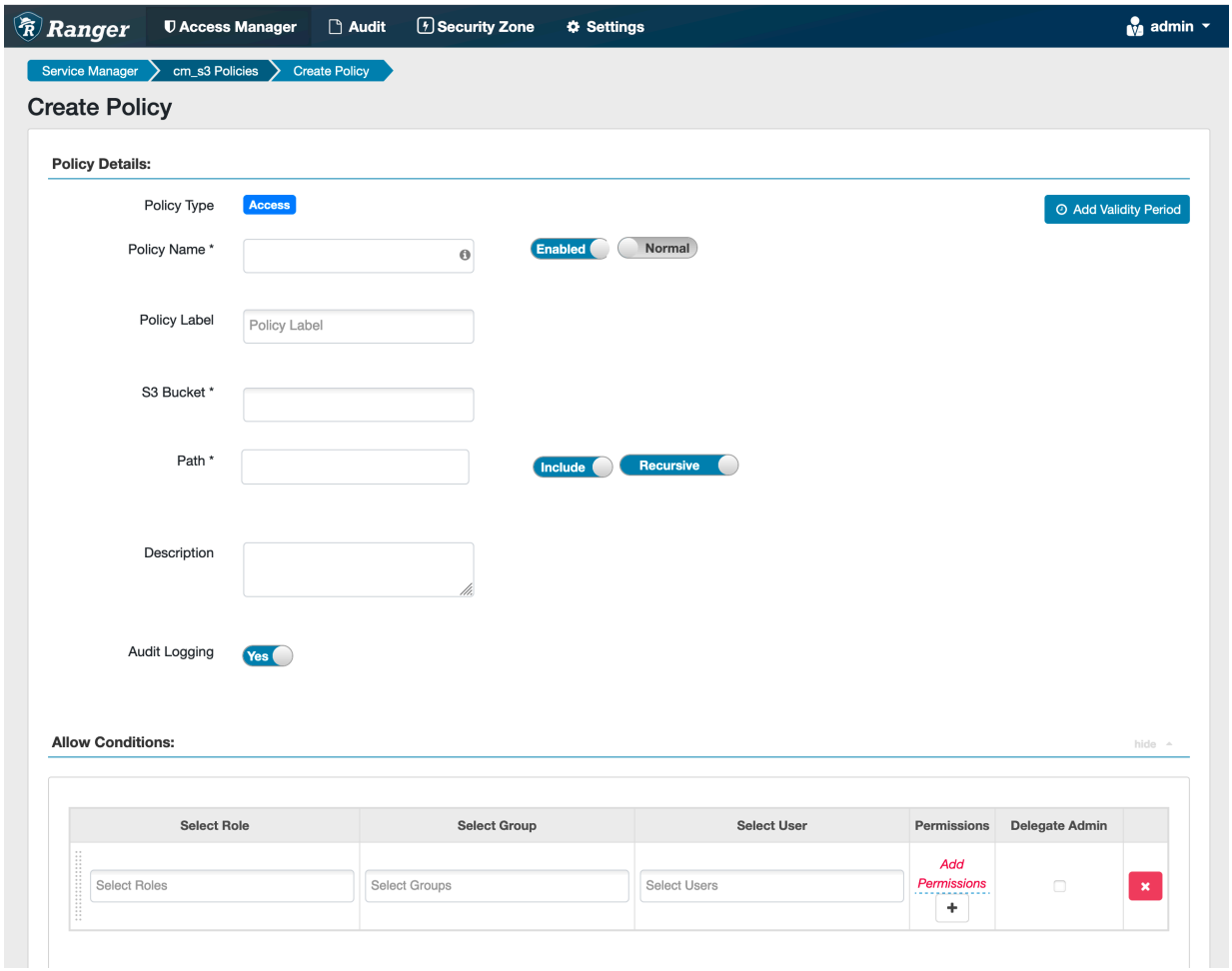

3. Complete the Create Policy page as follows:

Table 42: Policy Details

Field	Description
Policy Name	Enter a unique name for this policy. The name cannot be duplicated anywhere in the system.
Policy Label	An optional label for the policy. You can search reports and filter policies based on these labels.
Enabled/Disabled	Enables or disables the policy.
Normal/Override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
S3 Bucket	The S3 bucket.
Path	Specify the path for the policy. The default Recursive setting specifies that the path is recursive; you can also specify a non-recursive path. The default Include setting specifies that the path is included; you can also exclude the path.
Description	(Optional) Describe the purpose of the policy.

Field	Description
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Add Validity Period	Specify a start and end time for the policy.

Table 43: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Permissions	Add or edit permissions: Read, Write, Select/Deselect All.
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

Configure a resource-based policy: Solr

How to add a new policy to an existing Solr service.

Procedure

1. On Service Manager Resource Policies , select an existing Solr service.

   List of Policies displays a list of the policies defined for Solr service.

2. Click Add New Policy.
   Create Policy displays controls for creating details for a new policy.

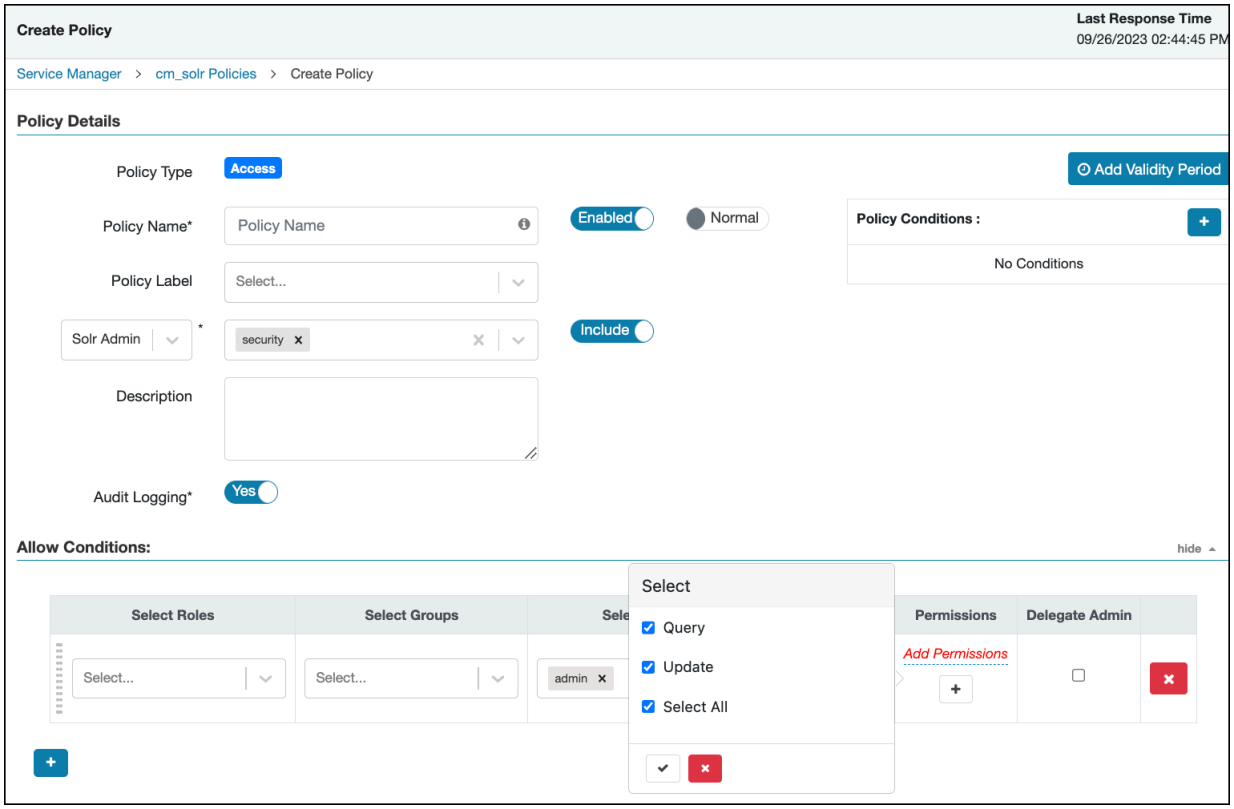

3. Complete the Create Policy page as follows:

Table 44: Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
Resource Type	collection - click to select from a list of dynamic values config - click to select from a list of dynamic values schema - click to select from a list of dynamic values admin - click to select COLLECTIONS, CORES, METRICS, SECURITY or AUTOSCALING
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.
Add Validity Period	Specify a start and end time for the policy.

Field	Description
Policy Conditions (applied at the policy level)	Click +, then specify an IP address range.

Table 45: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Policy Conditions (applied at the item level)	Specify an IP address range.
Permissions	Add or edit permissions: Query, Update
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

Related Information

Wildcards and variables in resource-based policies

Configure a resource-based policy: YARN

How to add a new policy to an existing YARN service.

Procedure

1. On Service Manager Resource Policies , select an existing YARN service.

   List of Policies displays a list of the policies defined for YARN service.

2. Click Add New Policy.
   Create Policy displays controls for creating details for a new policy.

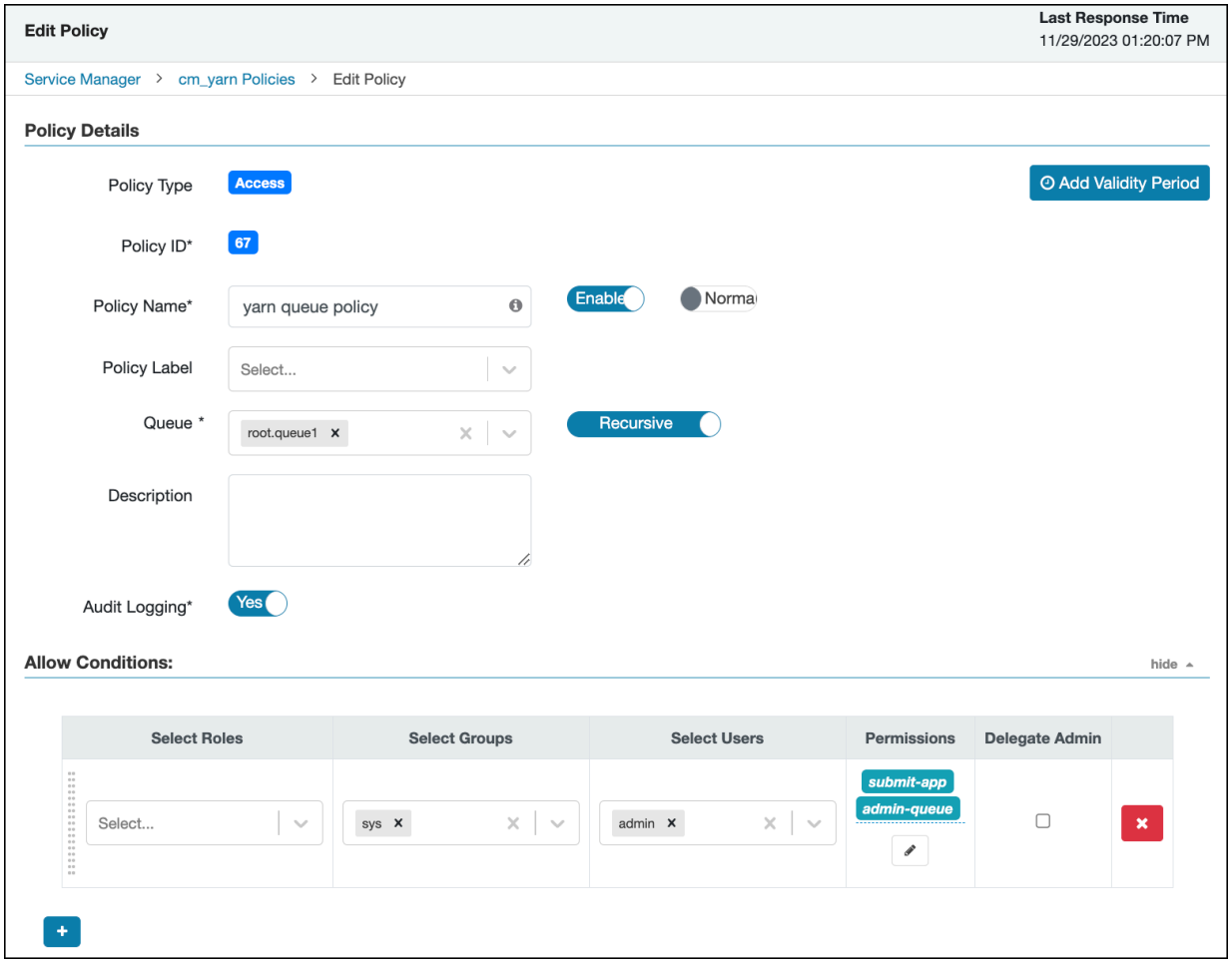

3. Complete the Create Policy page as follows:

Table 46: Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
Queue	The YARN queue to which the policy applies. For example, enter root.xyz for the xyz queue.
Recursive	The default recursive setting specifies that the policy will also be applied to all sub-queues; you can also specify a non-recursive path.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.

Field	Description
Add Validity Period	Specify a start and end time for the policy.

Table 47: Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select Delegate Admin. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Permissions	Add or edit permissions: submit-app, admin-queue, Select/Deselect All.
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.

   Combination of Deny All Other Accesses and Ranger macros (for example, {USER}) are not supported.
6. Click Add.

Related Information

Wildcards and variables in resource-based policies

Wildcards and variables in resource-based policies

Reference for wildcards and variables in resource-based policies.

Ranger Authorization Resource Policy Wildcard Characters

Wildcard characters can be included in the resource path, the database name, the table name, or the column name:

- * indicates zero or more occurrences of characters
- ? indicates a single character

Ranger Authorization Resource Policy {USER} Variable

The variable {USER} can be used to autofill the accessing user, for example:

In Select User, choose {USER}.

In Resource Path, enter data_{USER}.

Ranger Authorization Resource Policy {USER} Variable Recommended Practices and Customizability

Ranger requires that string '{USER}' is used to represent accessing user as the user in the policy-item in a Ranger policy. However, Ranger provides flexible way of customizing the string that is used as shorthand to represent the accessing user's name in the policy resource specification. By default, Ranger policy resource specification expects characters '{' and '}' as delimiters for string 'USER', however, ranger supports customizable way of specifying delimiter characters, escaping those delimiters, and the string 'USER' itself by prefixing it with another, user-specified string on a per resource-level basis in the service definition of each component supported by Ranger.

For example, if for a certain HDFS installation, if the path names may contain '{' or '}' as valid characters, but not '%' character, then the service-definition for HDFS can be specified as:

```
"resources": [
{
      "itemId": 1,
      "name": "path",
      "type": "path",
      "level": 10,
      "parent": "",
      "mandatory": true,
      "lookupSupported": true,
      "recursiveSupported": true,
      "excludesSupported": false,
      "matcher": "org.apache.ranger.plugin.resourcematcher.RangerPathReso
urceMatcher",
      "matcherOptions": {"wildcard": true, "ignoreCase": false}, "replaceT
okens":true, "tokenDelimiterStart":"%", "tokenDelimiterEnd":"%", "tokenDelim
iterPrefix":"rangerToken:"}
      "validationRegEx":"",
      "validationMessage": "",
      "uiHint":"",
      "label": "Resource Path",
      "description": "HDFS file or directory
path"
}
]
```

Corresponding ranger policy for the use case for HDFS will be written as follow:

```
resource: path=/home/%rangerToken:USER%
user: {USER}
permissions: all, delegateAdmin=true
```

The following customizable matcherOptions are available for this feature:

- replaceTokens: true if short-hand for user in resource-spec needs to be replaced at run-time with current-user's name; false if the resource-spec needs to be interpreted as it is. Default value: true.
- tokenDelimiterStart: Identifies start character of short-hand for current-user in resource specification. Default value: {.
- tokenDelimiterEnd: Identifies end character of short-hand for current-user in resource specification. Default value:}.
- tokenDelimiterEscape: Identifies escape character for escaping tokenDelimiterStart or tokenDelimiterEnd values in resource specification. Default value: \.
- tokenDelimiterPrefix: Identifies special prefix which together with string 'USER' makes up short-hand for current-user's name in the resource specification. Default value: .

For all other use-cases or combination of use-cases (for example, combining Ranger macros with Deny All Other Accesses), except mentioned above might result in unwanted behaviour.

Adding a policy condition to a resource-based policy

You can add a condition to a resource-based policy, using Ranger Admin Web UI when creating a new, or editing an existing policy.

About this task

Ranger Admin Web UI supports adding the following policy conditions to a new or existing resource-based policy for Knox, Kafka and Kafka-connect services.

- IP Address Range for example - xx.xxx.xxx, yy.yyy.yy
- Boolean expression for example - Country_Name="XYZ"

The Policy Conditions dialog prompts for inputs using uhint JSON. For populating For populating "IP-range" for example, we are using JSON like this:

```
{
    "itemId": 1,
    "name": "ip-range",
    "evaluator": "org.apache.ranger.plugin.conditionevaluator.RangerIpMat
cher",
    "evaluatorOptions": {},
    "validationRegEx": "",
    "validationMessage": "",
    "uiHint": "{ \"isMultiValue\":true }",
    "label": "IP Address Range",
    "description": "IP Address Range"
}
```

Procedure

1. In Service Manager Resource Policies cm_knox_policies (for example), choose one of the following:

   Add New Policy

   to add a new, tag-based policy.

   Policy ID

   click a policy ID to edit an existing policy.

2. In either Create Policy or Edit Policy Policy Conditions , click +.

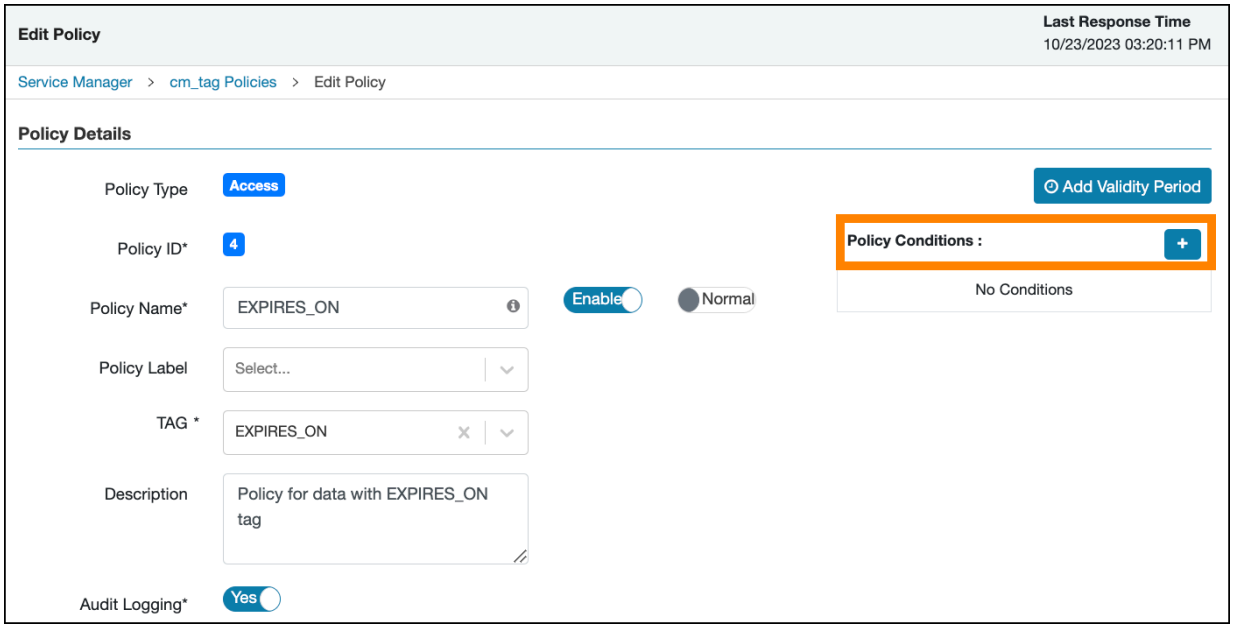

3. In Policy Conditions:
   a) In IP Address Range ?, enter or choose existing ip.address.values .
   b) In Enter boolean expression, enter an expression that evaluates to true or false.

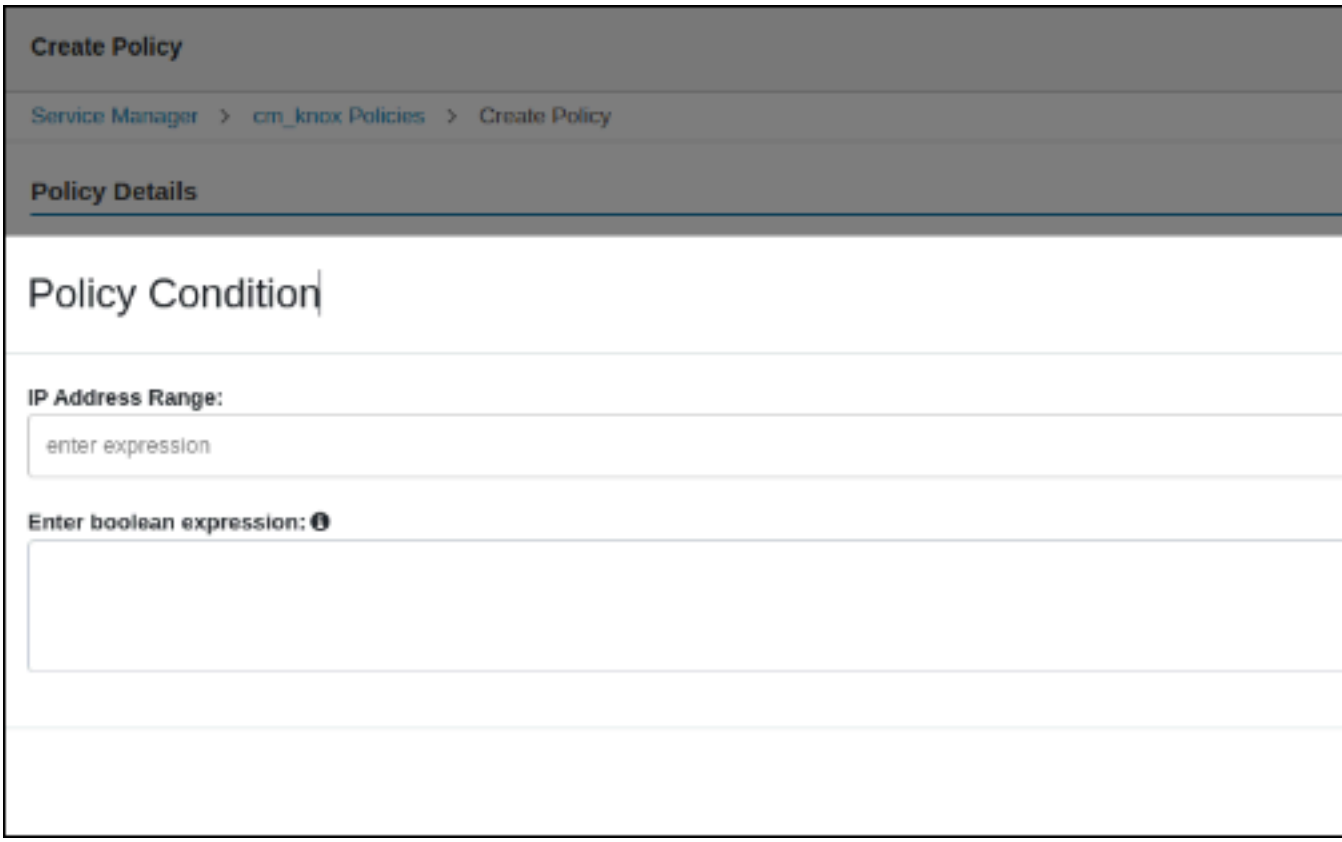

4. Click Save.

Adding a policy label to a resource-based policy

You can add a label to a resource-based policy, using Ranger Admin Web UI when creating a new, or editing an existing policy.

About this task

Ranger Admin Web UI supports adding one or more labels to a new or existing resource-based policy for Cloudera services. The Policy Label field displays all current labels created for a policy. Policy lables:

- Allow users to group sets of policies, using one or more lables.
- Enables users to search (filter) all policies using policy labels from both the Service Manager Policy Details or Service Manager Reports pages.
- Enables users to filter and export the filtered list of policies based on the policy label.

How to create a label for a policy

Procedure

1. In Service Manager Resource Policies cm_hdfs_policies (for example), choose one of the following:

   Add New Policy

   to add a new, tag-based policy.

   Policy ID

   click a policy ID to edit an existing policy.

2. In either Create Policy or Edit Policy Policy Conditions Policy Label , type or select a policy lable string.

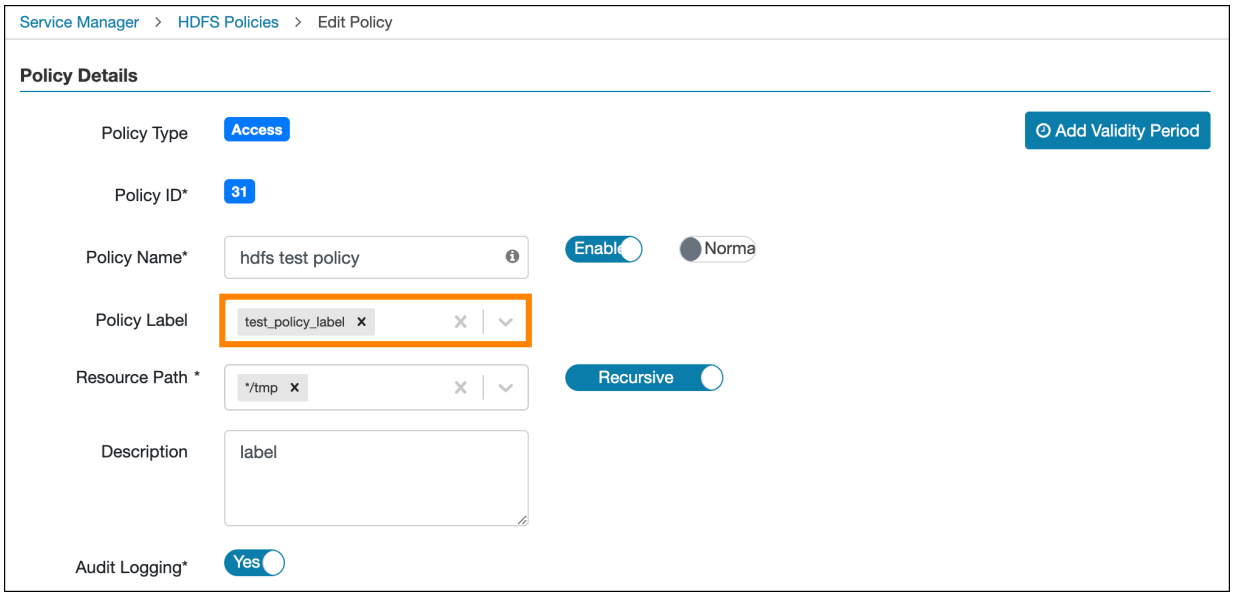

3. Click Save.

Example

To filter existing policies by label from Resource Policies Search , select Policy Label, then type a lable name, as shown in the following example:

Figure 5: Filtering Resource-based policies for HDFS service by label

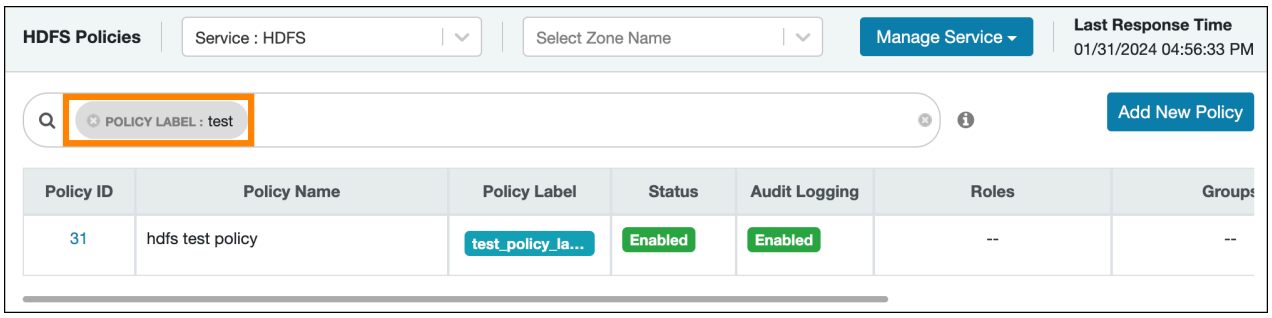

Example

To export a set of policies having the same label, filter as shown, using Service Manager Reports Policy Label (select) Search in the following example. Then, choose Export.

Figure 6: Exporting a list of policies filtered by label

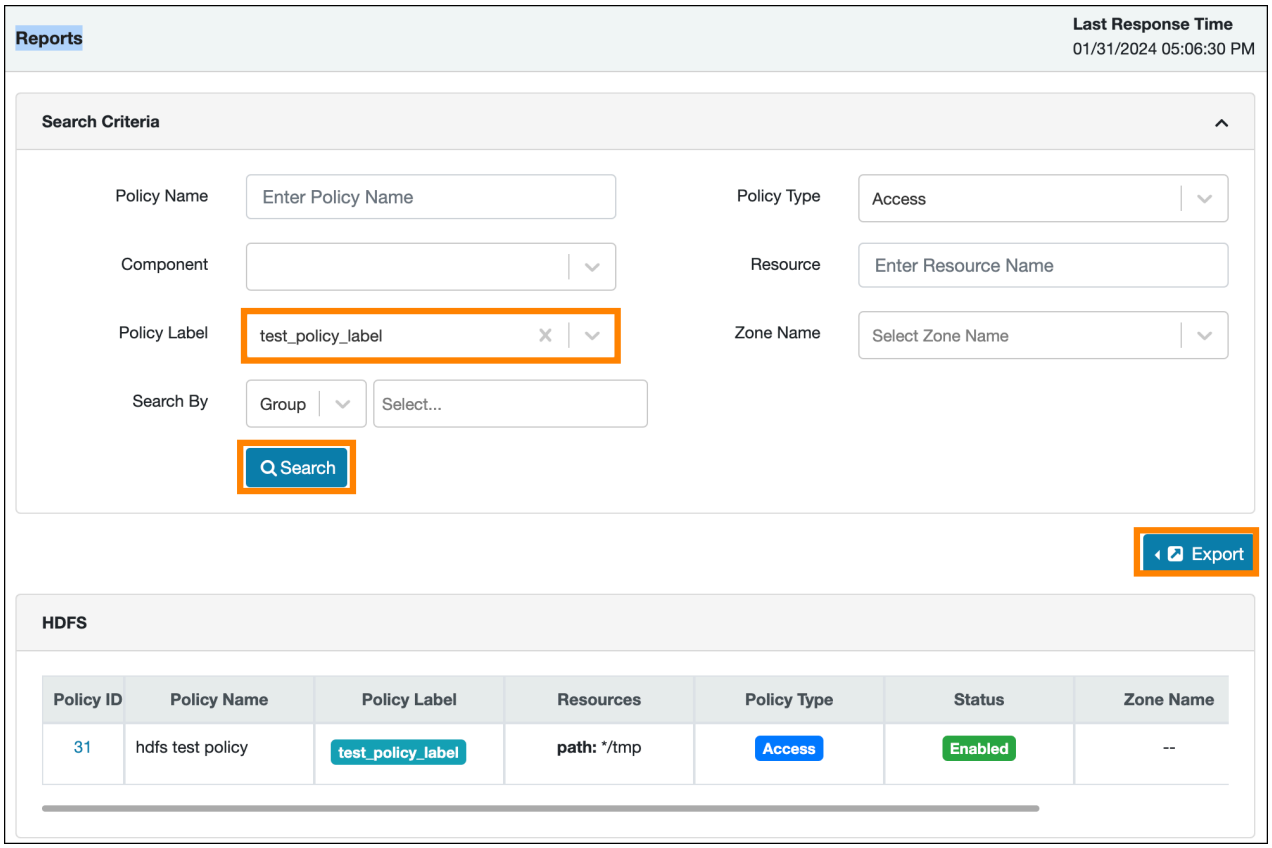

Preloaded resource-based services and policies

Apache Ranger includes preloaded resource-based services and policies.

- The preloaded resource-based services appear on the Service Manager page for resource-based policies Service names are prefixed with "cm_", with the exception of Hadoop SQL, which applies to multiple SQL stack components (Hive, Impala, and Hue).

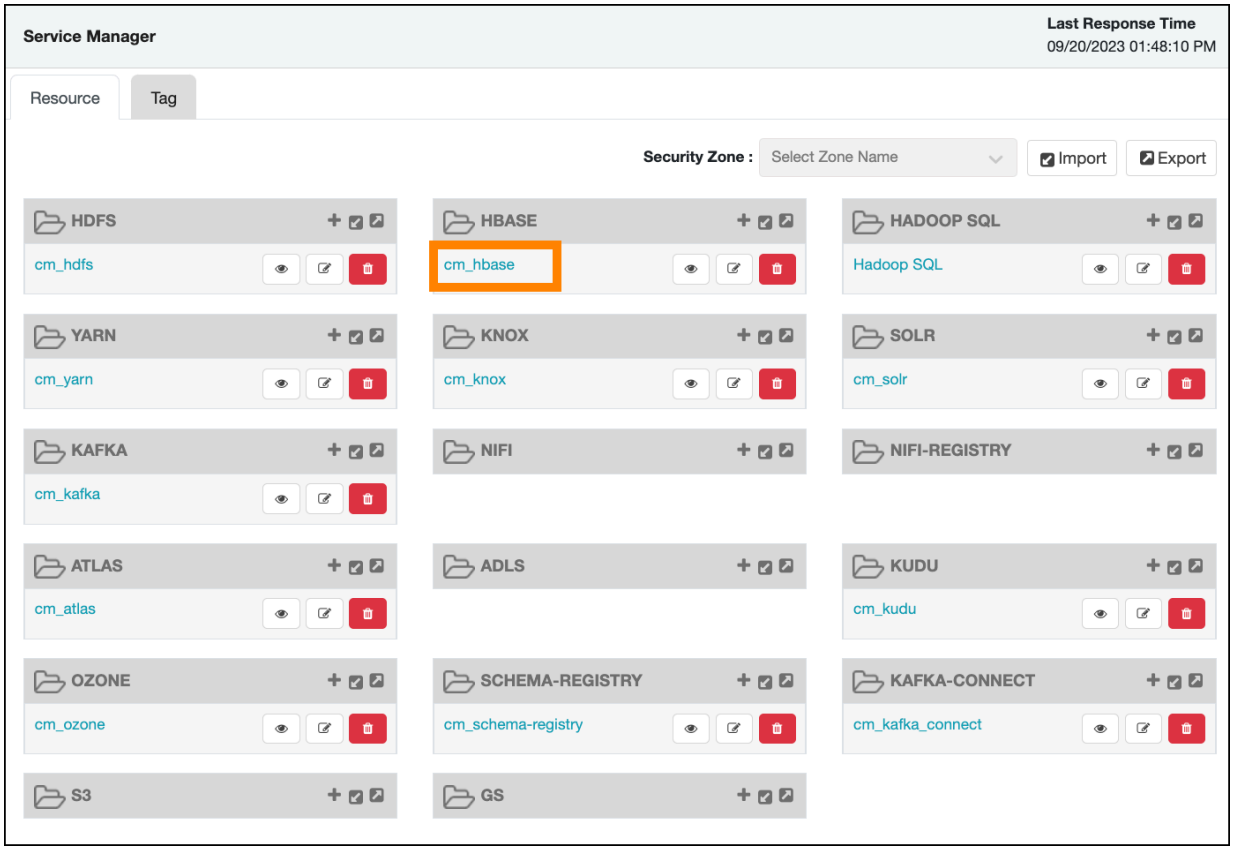

- To view the policies for each preloaded service, click the service name. To view policy details, click the applicable edit icon or policy ID number.

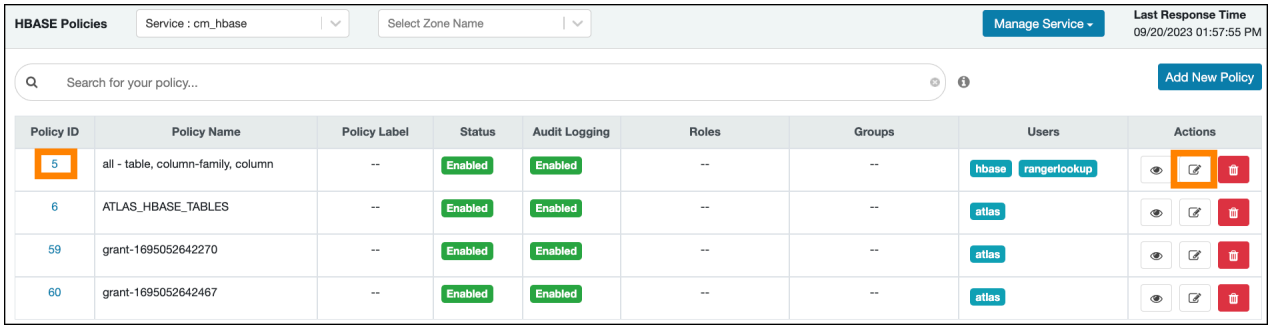

Index

cm_atlas

cm_hbase

cm_hdfs

cm_kafka

cm_knox

cm_nifi

cm_solr

cm_yarn

Hadoop SQL

cm_atlas

all - entity-type, entity-classification, entity, entity-business-metadata

This is a default policy of type "entity" that gives access to all entities and their business metadata attributes for the following users and groups, with the specified permissions:

- admin, dpprofiler, beacon – Update Business Metadata
- rangertagsync, rangerlookup – Read entity
- public group – Read entity

all - entity-type, entity-classification, entity

This is a default policy of type "entity" that gives access to all entities and their classifications for the following users and groups, with the specified permissions:

- admin, dpprofiler, beacon – Read, Create, Update, Delete entity & Add, Update, Remove classification
- rangertagsync, rangerlookup – Read entity
- public group – Read entity

all - entity-type, entity-classification, entity, entity-label

This is a default policy of type "entity" that gives access to all entities and classifications and their labels for the following users and groups, with the specified permissions:

- admin, dpprofiler, beacon – Add, Remove label
- rangertagsync, rangerlookup – Read entity
- public group – Read entity

all - relationship-type, end-one-entity-type, end-one-entity-classification, end-one-entity, end-two-entity-type, end-two-entity-classification, end-two-entity

This is a default policy of type "relationship" that gives access to all to all Entity-Relationships between End1-Entity-Type, End1-Entity-Classification, End1-Entity-ID and End2-Entity-Type, End2-Entity-Classification, End2-Entity-ID for the following users and groups, with the specified permissions:

- admin, dpprofiler, beacon – Add, Update, and Remove relationship
- public group – Add, Update, and Remove relationship

all - atlas-service

This is a default policy of type "atlas-service" that gives access to all atlas-services [export, import, purge, server] for the following users, with the specified permissions:

- admin, dpprofiler, beacon – Admin Export and Admin Import

all - type-category, type

This is a default policy of type "type-category" that gives access to all type categories [ENUM, ENTITY, CLASSIFICATION, RELATIOSHIP, STRUCT] and type names for the following users, with the specified permissions:

- admin, dpprofiler, beacon – Create, Update, and Delete type

Allow users to manage favorite searches

This is a default policy of type "entity-type" that gives access to __AtlasUserProfile and __AtlasUserSavedSearch resources which are internal types for favorite search. This policy provides Read, Create, Update, and Delete Entity permissions to validated users who create a favorite search.

cm_hbase

all - table, column-family, column

Provides access to all HBase tables, column-families, and columns to the following users, with the specified permissions:

- hbase, rangerlookup – Read, Write, Create, Admin

ATLAS_HBASE_TABLES

Provides access to all HBase column-families and columns in the atlas_janus and ATLAS_ENTITY_AUDIT_EVENTS HBase tables, to the following user, with the specified permissions:

- atlas – Read, Write, Create, Admin

cm_hdfs

all - path

Provides access to all HDFS resource paths to the following users, with the specified permissions:

- hdfs, rangerlookup – Read, Write, Execute

kms-audit-path

Provides access to the /ranger/audit/kms resource path to the following user, with the specified permissions:

- keyadmin – Read, Write, Execute

cm_kafka

all - topic

Provides access to all topics to the following users, with the specified permissions:

- kafka, rangerlookup, streamsmsgmgr, streamsrepmgr – Publish, Consume, Configure, Describe, Create, Delete, Describe Configs, Alter Configs

all - cluster

Provides access to all clusters to the following users, with the specified permissions:

- kafka, rangerlookup, streamsmsgmgr, streamsrepmgr – Configure, Describe, Create, Kafka Admin, Idempotent Write, Describe Configs, Alter Configs

all - transactionalid

Provides transactionalid access to the following users, with the specified permissions:

- kafka, rangerlookup, streamsmsgmgr, streamsrepmgr – Publish, Describe

all - delegationtoken

Provides delegationtoken access to the following users, with the specified permissions:

- kafka, rangerlookup, streamsmsgmgr, streamsrepmgr – Describe

ATLAS_HOOK

Provides ATLAS_HOOK topic access to the following users, with the specified permissions:

- hbase, hive, impala, mlgov – Publish
- atlas – Create, Configure, and Consume

ATLAS_ENTITIES

Provides ATLAS_ENTITIES topic access to the following users, with the specified permissions:

- atlas – Create, Configure, and Publish
- rangertagsync – Consume

ATLAS_SPARK_HOOK

Provides ATLAS_SPARK_HOOK topic access to the following user, with the specified permissions:

- atlas – Create, Configure, and Consume

Also provides ATLAS_SPARK_HOOK topic access to the following group, with the specified permissions:

- public – Publish

cm_knox

all - topology, service

Provides access to all Knox topologies and services to the following users, with the specified permissions:

- admin, rangerlookup – Allow

cm_nifi

all - nifi-resource

Provides access to all NiFi resource identifiers to the following user, with the specified permissions:

- rangerlookup – Read, Write

cm_solr

all - collection

Provides access to all Solr collections to the following users, with the specified permissions:

- solr, rangerlookup, ranger, atlas – Query, Update, Others, Solr Admin

RANGER_AUDITS_COLLECTION

Provides access to the RANGER_AUDITS_COLLECTION Solr collection to the following users, with the specified permissions:

- atlas, hbase, hdfs, hive, impala, kafka, knox, nifi, ranger, storm, yarn – Query, Update, Others
- ranger – Query, Update, Others, Solr Admin

cm_yarn

all - queue

Provides access to all YARN queues to the following users, with the specified permissions:

- yarn, rangerlookup – submit-app, admin-queue

Hadoop SQL

all - global

Provides global access to the following users, with the specified permission:

- hive, beacon, dpprofiler, hue, admin, impala, rangerlookup – Temporary UDF Admin

Note: The Ranger web UI may show additional permissions for the all-global policy, but the only valid permission is Temporary UDF Admin.

all - database, table, column

Provides access to all databases, tables, and columns to the following users, with the specified permissions:

- hive, rangerlookup, impala – Select, Update, Create, Drop, Alter, Index, Lock, All, Read, Write, ReplAdmin, Service Admin, Temporary UDF Admin, Refresh
- {OWNER} – All

all - database, table

Provides access to all databases and tables to the following users, with the specified permissions:

- hive, rangerlookup, impala – Select, Update, Create, Drop, Alter, Index, Lock, All, Read, Write, ReplAdmin, Service Admin, Temporary UDF Admin, Refresh
- {OWNER} – All

all - storage-type, storage-url

Ranger introduces new resources “storage-type” and “storage-url” in HadoopSQL Service and supports only one permission “RW Storage ”. When a user creates / alters a table, they will be authorized against this resource policy. Users granted “RW Storage” permission on the resource representing the storage-type + storage-url, can create/alter the table in the respective storage. Provides ccess to all databases to the following users, with the RW Storage permission only:

- hive, rangerlookup, impala, beacon, dpprofiler, hue, admin

Note: {OWNER} macro should NOT be configured for StorageHandler policies.

all - database

Provides access to all databases to the following users, with the specified permissions:

- hive, rangerlookup, impala – Select, Update, Create, Drop, Alter, Index, Lock, All, Read, Write, ReplAdmin, Service Admin, Temporary UDF Admin, Refresh
- {OWNER} – All

Also provides access to all databases to the following group, with the specified permissions:

- public – Create

all - hiveservice

Provides hiveservice access to the following users, with the specified permissions:

- hive, rangerlookup, impala – Select, Update, Create, Drop, Alter, Index, Lock, All, Read, Write, ReplAdmin, Service Admin, Temporary UDF Admin, Refresh

all - database, udf

Provides database and udf access to the following users, with the specified permissions:

- hive, rangerlookup, impala – Select, Update, Create, Drop, Alter, Index, Lock, All, Read, Write, ReplAdmin, Service Admin, Temporary UDF Admin, Refresh
- {OWNER} – All

all - url

Provides url access to the following users, with the specified permissions:

- hive, rangerlookup, impala – Select, Update, Create, Drop, Alter, Index, Lock, All, Read, Write, ReplAdmin, Service Admin, Temporary UDF Admin, Refresh

default database tables columns

Provides access to all tables and columns in the default database to the following user, with the specified permissions:

- impala – Create

Also provides access to all tables and columns in the default database to the following group, with the specified permissions:

- public – Create

information_schema database tables columns

Provides access to all tables and columns in the information_schema database to the following user, with the specified permissions:

- impala – Select

Also provides access to all tables and columns in the information_schema database to the following group, with the specified permissions:

- public – Select

Importing and exporting resource-based policies

You can export and import policies from the Ranger Admin UI for cluster resiliency (backups), during recovery operations, or when moving policies from test clusters to production clusters. You can export/import a specific subset of policies (such as those that pertain to specific resources or user/groups) or clone the entire repository (or multiple repositories) via Ranger Admin UI.

Interfaces

You can import and export policies from Ranger Admin Web UI Service Manager Resource :

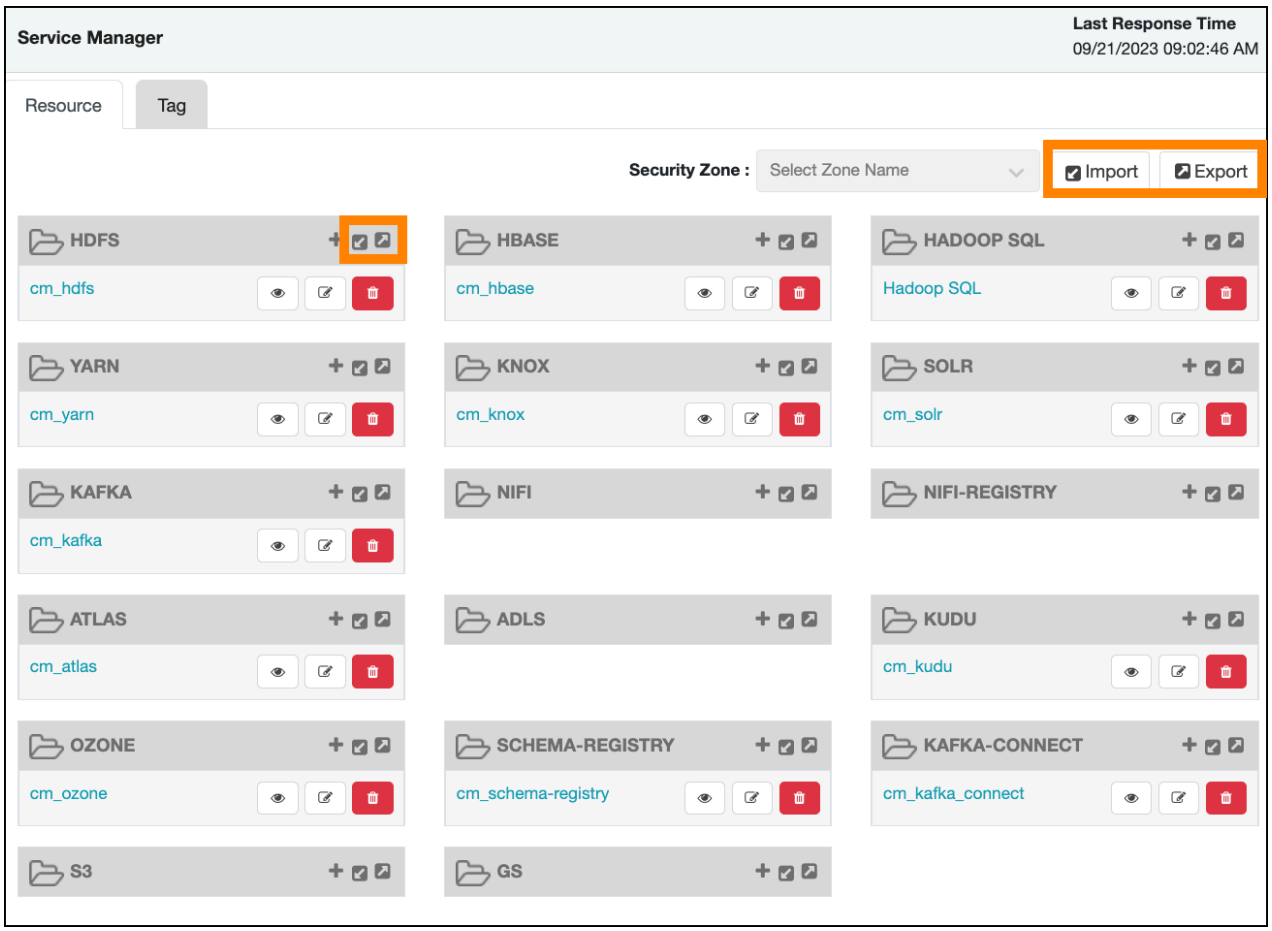

You can also export policies from Ranger Admin Web UI Service Manager Reports :

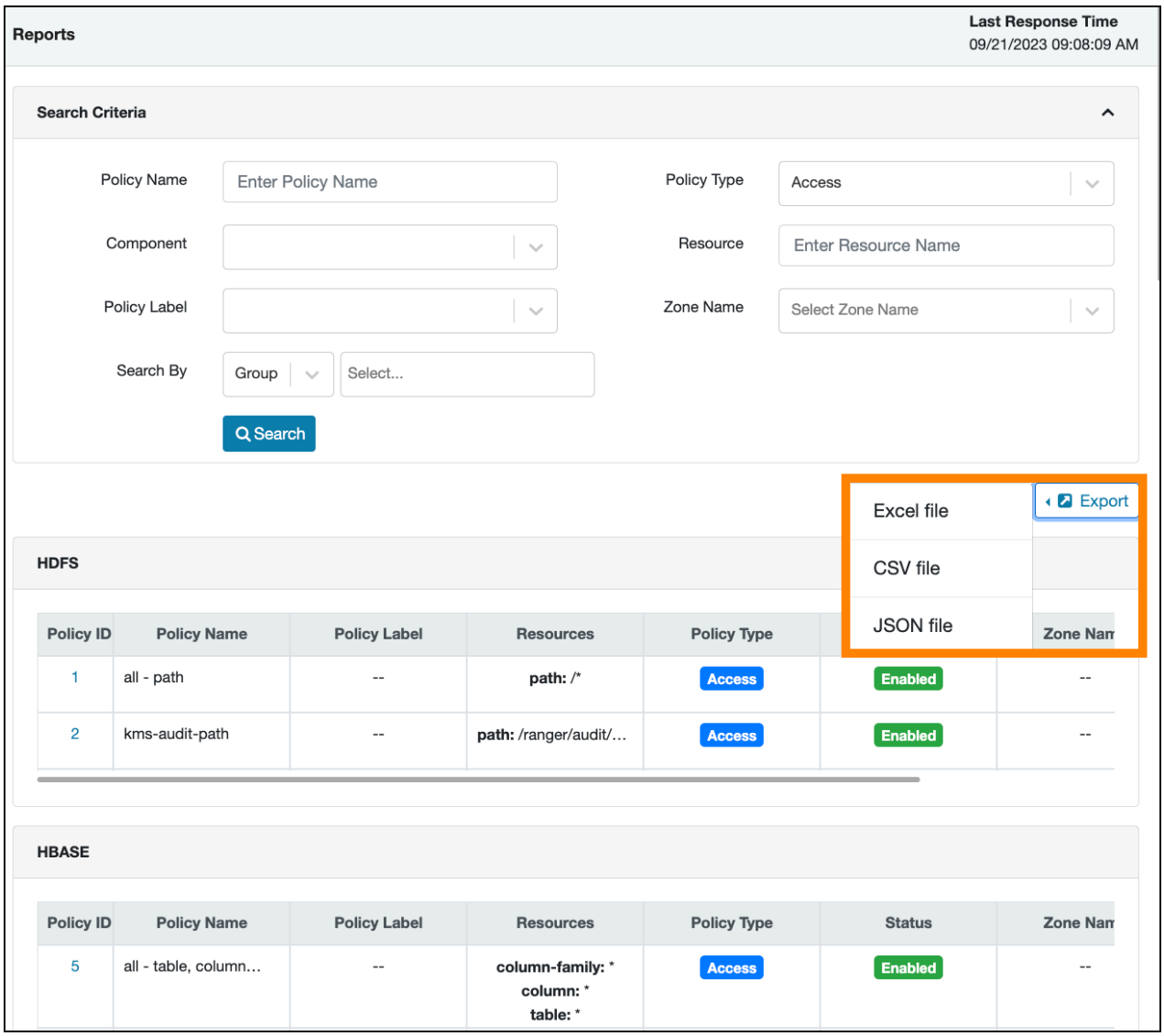

Table 48: Export Policy Options

	Service Manager Page	Reports Page
Formats	JSON	JSON Excel CSV
Filtering Supported	No	Yes
Specific Service Export	Yes	Via filtering

Filtering

When exporting from the Reports page, you can apply filters before saving the file.

Export Formats

You can export policies in the following formats:

- Excel
- JSON

- CSV

  Note: CSV format is not supported for importing policies.

When you export policies from the Service Manager page, the policies are automatically downloaded in JSON format. If you wish to export in Excel or CSV format, export the policies from the Reports page dropdown menu.

Required User Roles

The Ranger admin user can import and export only Resource & Tag based policies. The credentials for this user are set in Ranger Configs > Advanced ranger-env in the fields labeled admin_username (default: admin/admin).

The Ranger KMS keyadmin user can import and export only KMS policies. The default credentials for this user are keyadmin/keyadmin.

Limitations

To successfully import policies, use the following database versions:

- MariaDB: 10.1.16+
- MySQL: 5.6.x+
- Oracle: 11gR2+
- PostgreSQL: 8.4+
- MS SQL: 2008 R2+

Partial import is not supported.

Important: When importing Ranger policies, it is crucial to understand the behavior of the import process regarding users and groups that do not exist in Ranger:

- Automatic user or group creation: The Ranger policy import process automatically creates any non-existing users or groups at the Ranger admin end as external users or groups. This is because the import process does not have access to the passwords of these users or groups, unlike users or groups created through the Ranger UI who are considered internal with known passwords.
- Handling of existing users: If a synchronization process is actively managing users or groups, any users or groups created during the policy import are updated with the details from the synchronization process as soon as they are synchronized. This ensures that users and groups information remains current and consistent with your identity management system.
- Impact on policy enforcement: While the absence of specific Hive resources or HDFS paths in the target system does not impede the policy import process, policies linked to these non-existent resources are not enforced until the resources are created. Therefore, Cloudera recommends to verify the existence of critical resources in the new environment to ensure that all policies function as expected after migration.

Related Information

Importing and exporting tag-based policies

Import resource-based policies for a specific service

How to import resource-based policies for a specific service (HBase, YARN, etc.).

Procedure

1. On the Service Manager page, click the Import icon for the service:

The Import Policy page appears.

2. Select the file to import.

You can only import policies in JSON format.

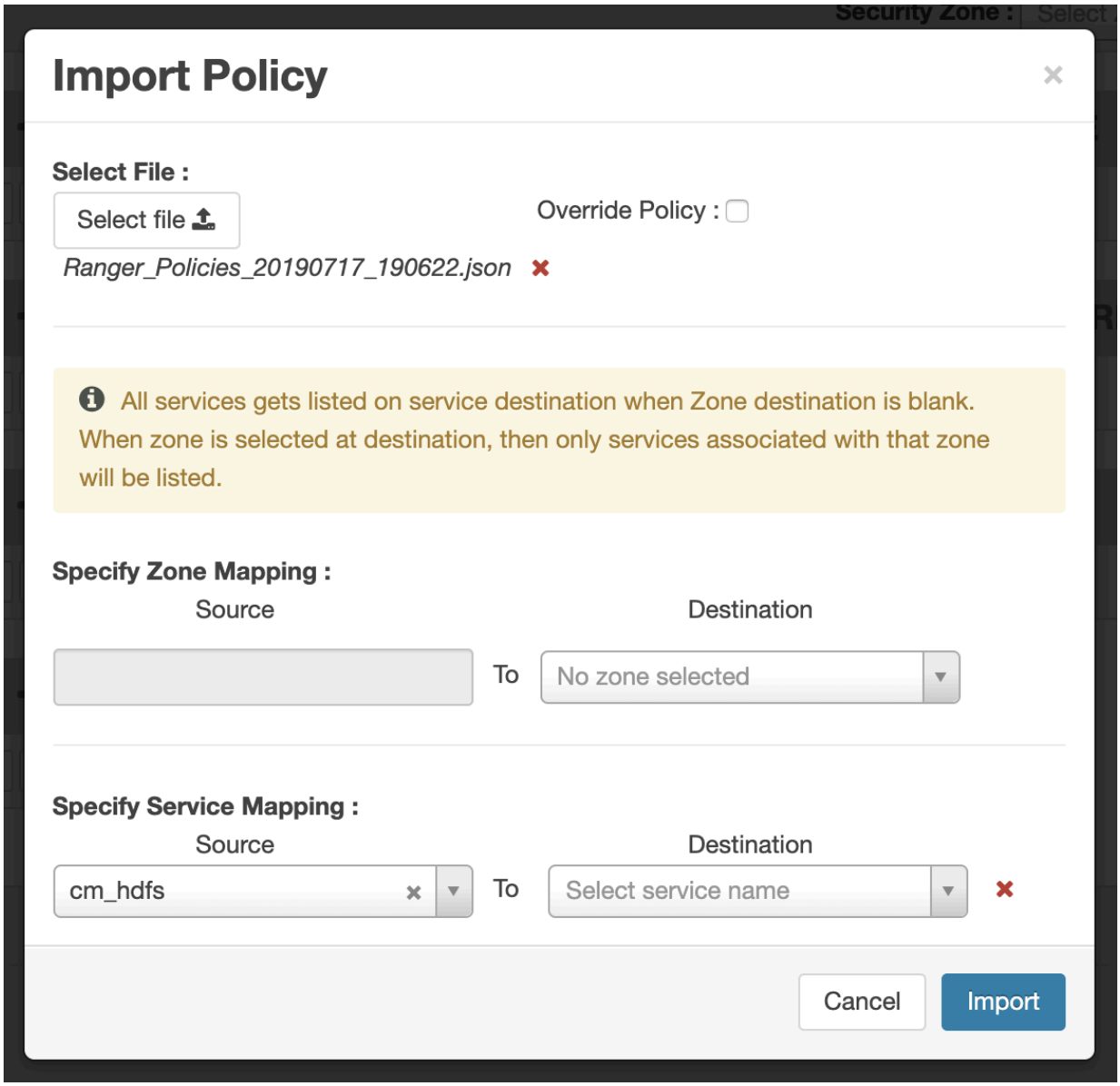

3. (Optional) Configure the import operation:
   a) The Override Policy option deletes all policies of the destination repositories.
   b) Zone Mapping – when no destination is selected, all services are imported. When a destination is selected, only the services associated with that security zone are imported.
   c) Service Mapping maps the downloaded file repository, i.e. source repository to destination repository. You can use the red x symbols to remove services from the import. Scroll down to view all service mappings.

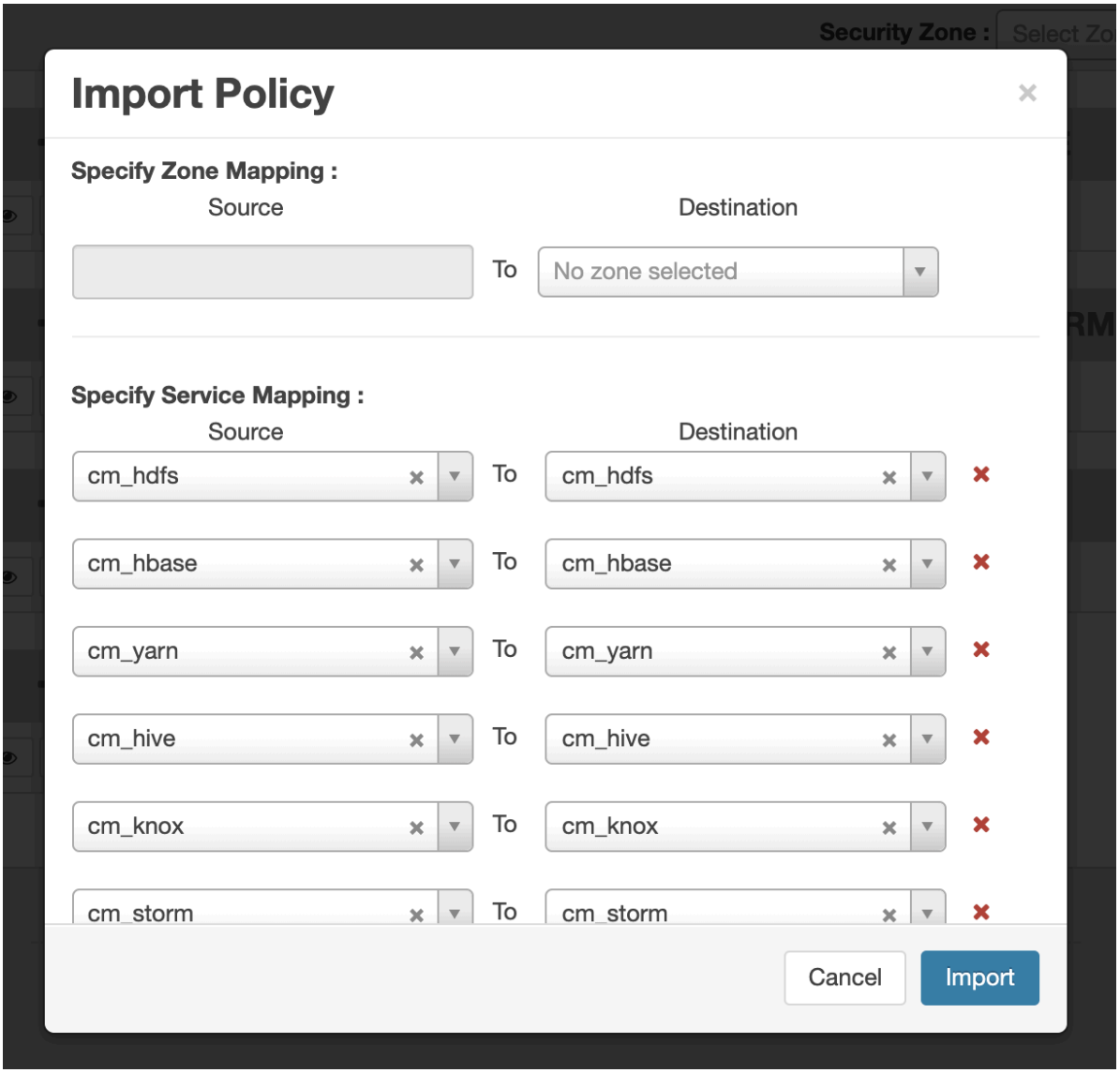

4. Click Import.
   A confirmation message appears after the file is imported.

Related Information

Import resource-based policies for all services

Import resource-based policies for all services

How to import policies for all services.

Procedure

1. On Service Manager Resource , click Import.

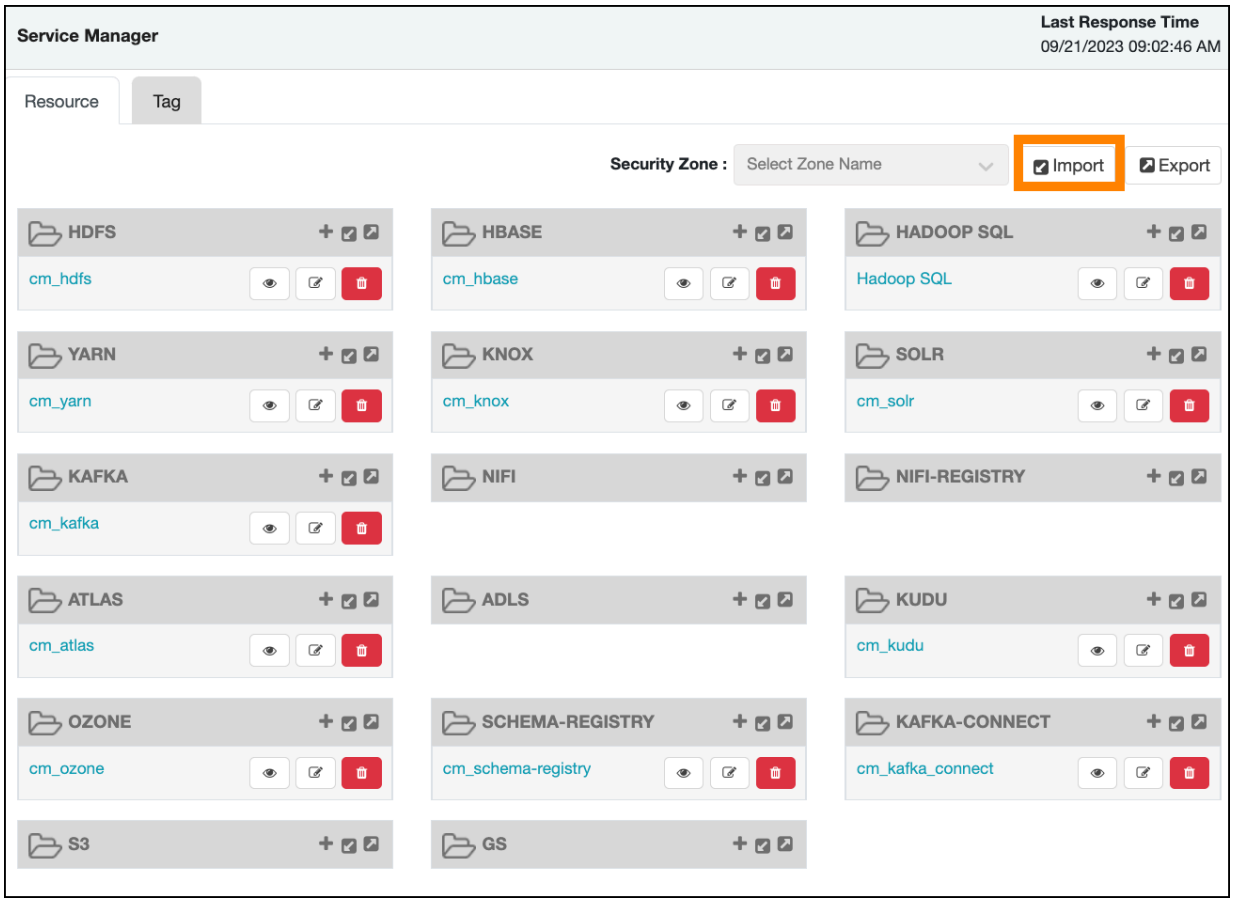

The Import Policy page appears.

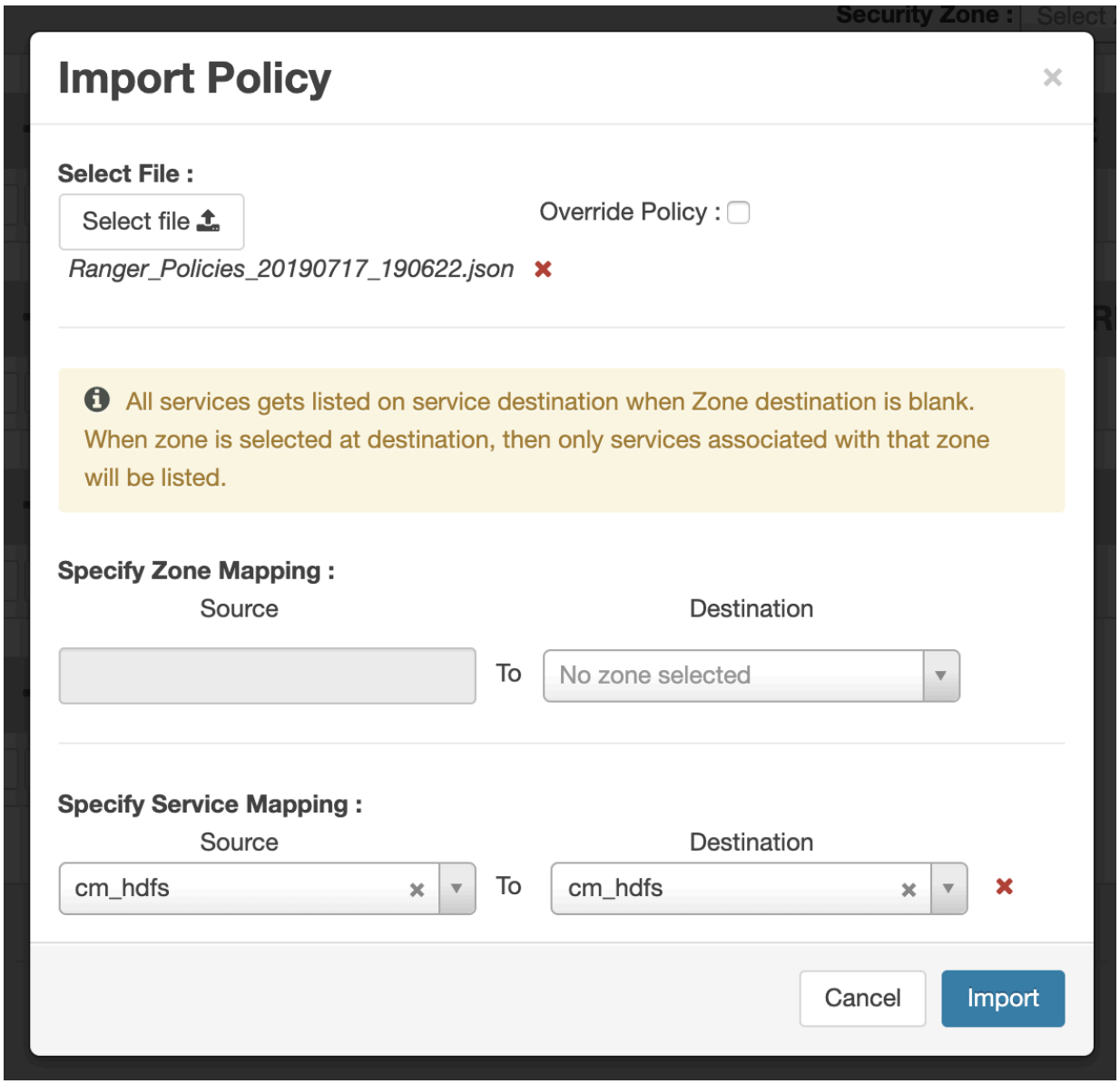

2. Select the file to import.
   You can only import policies in JSON format.

3. (Optional) Configure the import operation:
   a) The Override Policy option deletes all policies of the destination repositories.
   b) Zone Mapping – when no destination is selected, all services are imported. When a destination is selected, only the services associated with that security zone are imported.
   c) Service Mapping maps the downloaded file repository, i.e. source repository to destination repository. You can use the red x symbols to remove services from the import. Scroll down to view all service mappings.

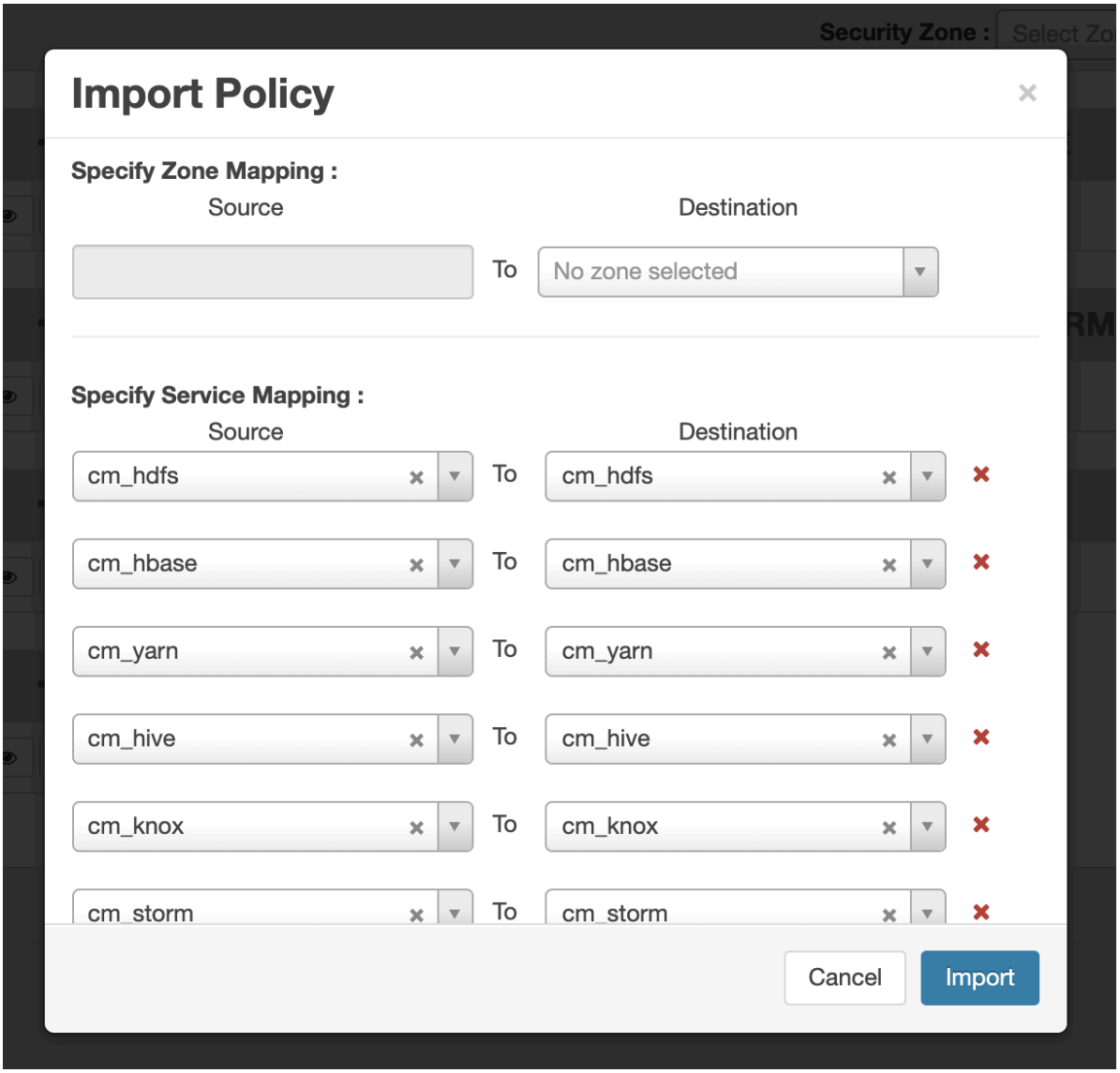

4. Click Import.
   A confirmation message appears after the file is imported.

Related Information

Import resource-based policies for a specific service

Export resource-based policies for a specific service

How to export the policies for a specific service (HBase, YARN, etc).

About this task

If you would like to export in Excel or CSV format, export the policies from the Reports page dropdown menu.

Procedure

1. On Service Manager Resource , click the Export icon for the service:

The Export Policy page appears.

2. Click Export.

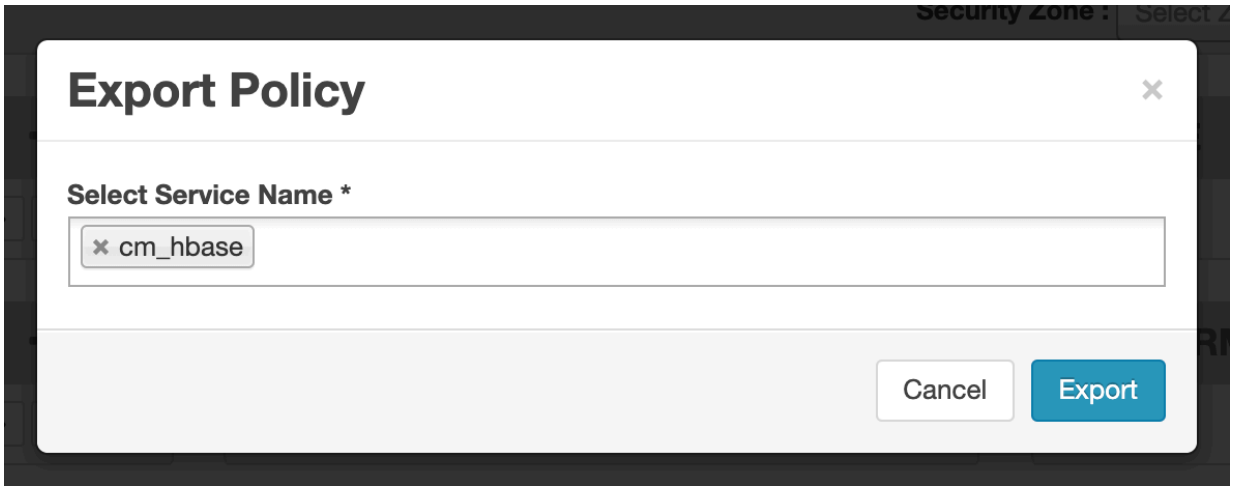

The file downloads in your browser as a JSON file.

Related Information

Export all resource-based policies for all services

Export all resource-based policies for all services

How to export the policies for all services.

About this task

If you would like to export in Excel or CSV format, export the policies from the Reports page drop-down menu.

Procedure

- From Service Manager Resource :
  a) Click Export.
     The Export Policy page appears.
  b) Remove components or specific services, then click Export.

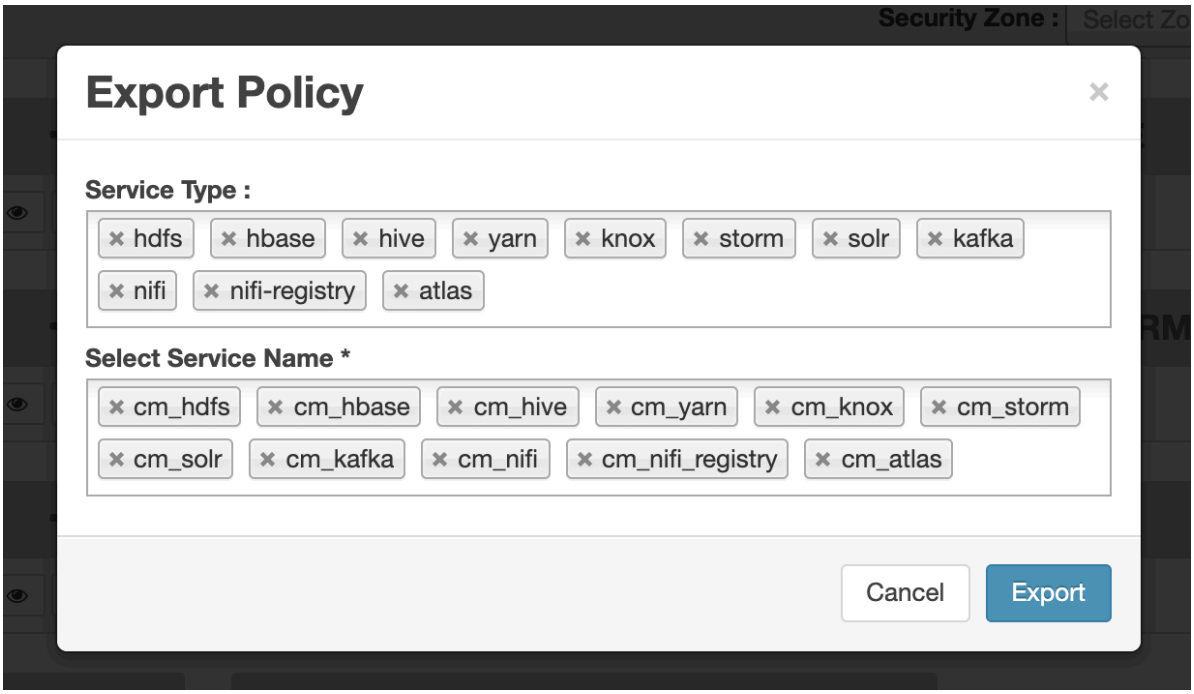

The file downloads in your browser as a JSON file.

- From Service Manager Reports :
  a) Apply filters before exporting the file.
  b) Open the Export drop-down menu:

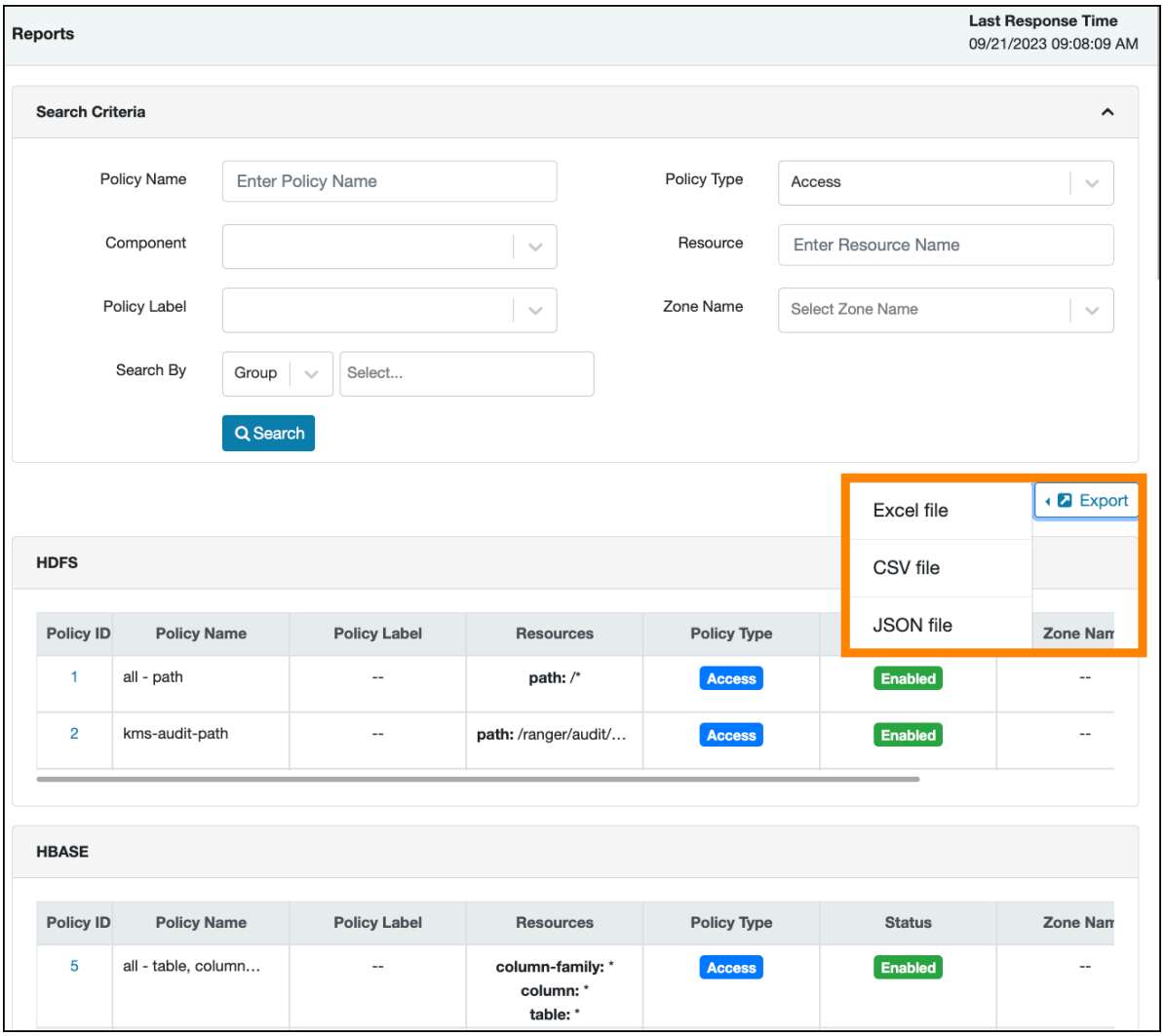

  c) Select the file format.
  The file downloads in your browser.

Related Information

Export resource-based policies for a specific service

Row-level filtering and column masking in Hive

You can use Apache Ranger row-level filters to set access policies for rows in Hive tables. You can also use Ranger column masking to set policies that mask data in Hive columns, for example to show only the first or last four characters of column data.

Row-level filtering in Hive with Ranger policies

Row-level filtering helps simplify Hive queries. By moving the access restriction logic down into the Hive layer, Hive applies the access restrictions every time data access is attempted. This helps simplify authoring of the Hive query, and provides seamless behind-the-scenes enforcement of row-level segmentation without having to add this logic to the predicate of the query.

About this task

Row-level filtering also improves the reliability and robustness of Hadoop. By providing row-level security to Hive tables and reducing the security surface area, Hive data access can be restricted to specific rows based on user characteristics (such as group membership) and the runtime context in which this request is issued.

Typical use cases where row-level filtering can be beneficial include:

- A hospital can create a security policy that allows doctors to view data rows only for their own patients, and that allows insurance claims administrators to view only specific rows for their specific site.
- A bank can create a policy to restrict access to rows of financial data based on the employee's business division, locale, or based on the employee's role (for example: only employees in the finance department are allowed to see customer invoices, payments, and accrual data; only European HR employees can see European employee data).
- A multi-tenant application can create logical separation of each tenant's data so that each tenant can see only their own data rows.

You can use Apache Ranger row-level filters to set access policies for rows in Hive tables. Row-level filter policies are similar to other Ranger access policies. You can set filters for specific users, groups, and conditions.

The following conditions apply when using row-level filters:

- The filter expression must be a valid WHERE clause for the table or view.
- Each table or view should have its own row-level filter policy.
- Wildcard matching is not supported on database or table names.
- Filters are evaluated in the order listed in the policy.
- An audit log entry is generated each time a row-level filter is applied to a table or view.

Procedure

1. On Service Manager Resource , select an existing Hadoop SQL service.
2. Select Row Level Filter , then click Add New Policy.

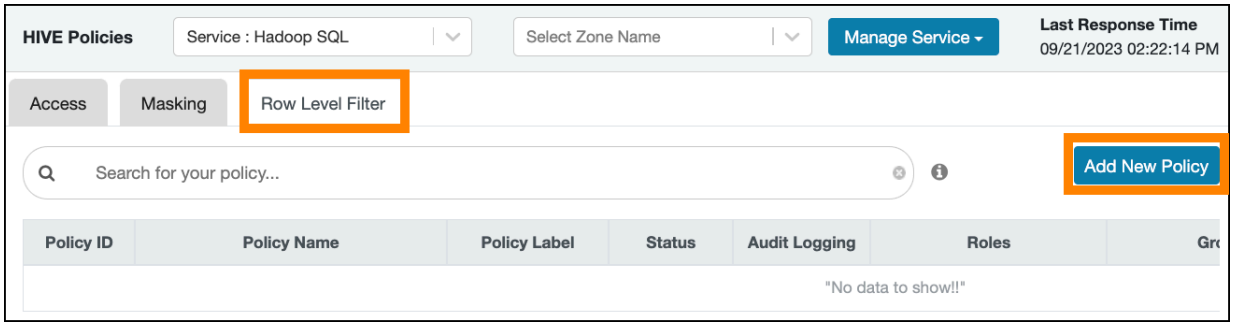

3. On the Create Policy page, add the following information for the row-level filter:

Table 49: Policy Details

Field	Description
Policy Name (required)	Enter an appropriate policy name. This name cannot be duplicated across the system. The policy is enabled by default.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
Hive Database (required)	Type in the applicable database name. The auto-complete feature displays available databases based on the entered text.

Field	Description
Hive Table (required)	Type in the applicable table name. The auto-complete feature displays available tables based on the entered text.
Audit Logging	Audit Logging is set to Yes by default. Select No to turn off audit logging.
Description	Enter an optional description for the policy.
Add Validity Period	Specify a start and end time for the policy.

Table 50: Row Filter Conditions

Label	Description
Select Role	Specify the roles to which this policy applies.
Select Group	Specify the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify one or more users to which this policy applies.
Access Types	Currently select is the only available access type. This will be used in conjunction with the WHERE clause specified in the Row Level Filter field.

Label	Description
Add Row Filter	• To create a row filter for the specified users, groups, and roles, Click Add Row Filter, then type a valid WHERE clause in the Enter filter expression box. • To allow Select access for the specified users and groups without row-level restrictions, do not add a row filter (leave the setting as "Add Row Filter"). • Filters are evaluated in the order listed in the policy. The filter at the top of the Row Filter Conditions list is applied first, then the second, then the third, and so on.

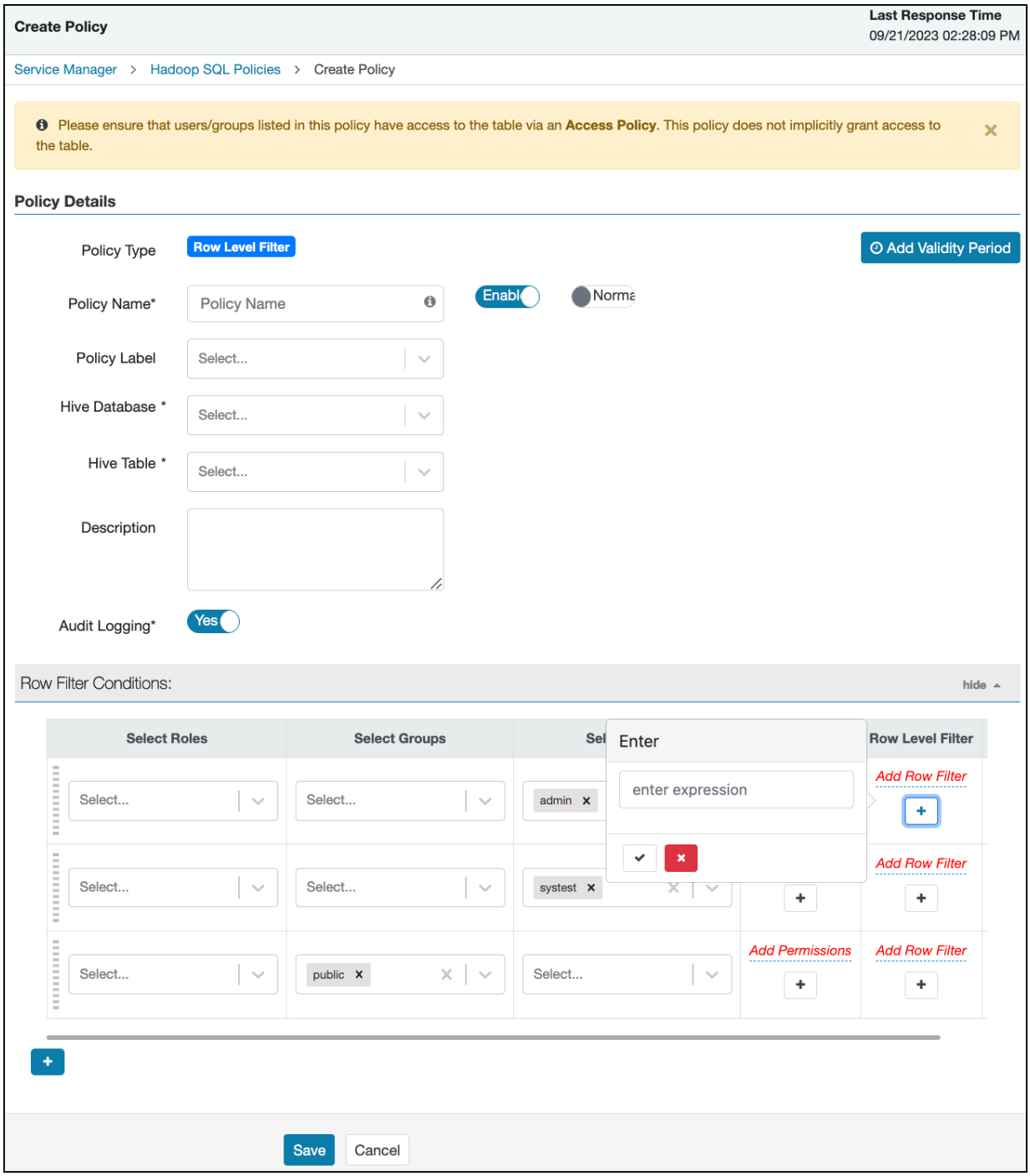

4. To move a condition in the Row Filter Conditions list (and therefore change the order in which the list is evaluated), click the dotted rows icon at the left of the condition row, then drag the condition to a new position in the list.

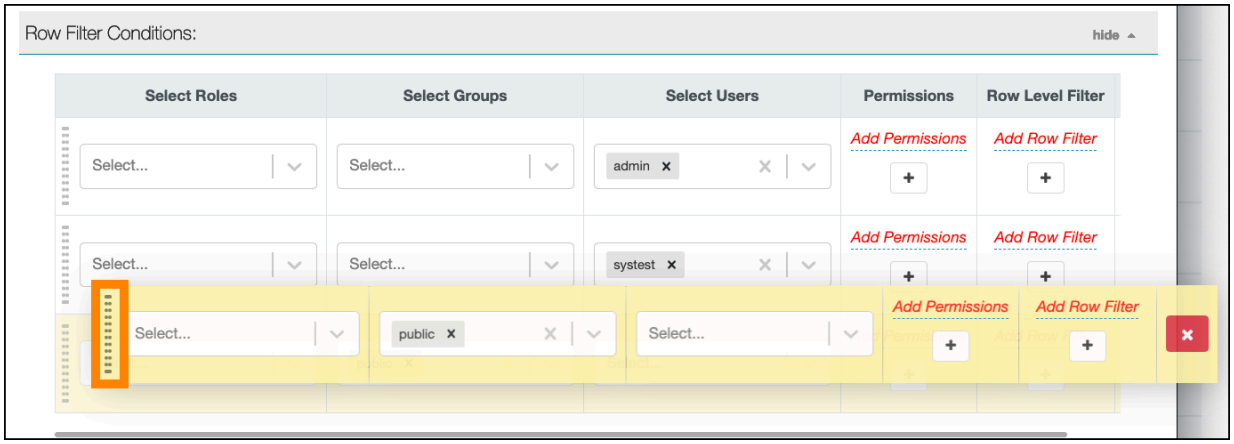

5. Click Add in Create Policy to add the new row-level filter policy.

Dynamic resource-based column masking in Hive with Ranger policies

You can use Apache Ranger dynamic resource-based column masking capabilities to protect sensitive data in Hive in near real-time. You can set policies that mask or anonymize sensitive data columns (such as PII, PCI, and PHI) dynamically from Hive query output. For example, you can mask sensitive data within a column to show only the first or last four characters.

About this task

Dynamic column masking policies are similar to other Ranger access policies for Hive. You can set filters for specific users, groups, and conditions. With dynamic column-level masking, sensitive information never leaves Hive, and no changes are required at the consuming application or the Hive layer. There is also no need to produce additional protected duplicate versions of datasets.

The following conditions apply when using Ranger column masking policies to mask data returned in Hive query results:

- A variety of masking types are available, such as show last 4 characters, show first 4 characters, Hash, Nullify, and date masks (show only year).
- You can specify a masking type for specific users, groups, and conditions.
- Wildcard matching is not supported.
- Each column should have its own masking policy.
- Masks are evaluated in the order listed in the policy.
- An audit log entry is generated each time a masking policy is applied to a column.

Note: Operations like insert/update/delete/export are denied for users if row-filter or column-masking policies are applicable on the table for the user.

Procedure

1. On Service Manager Resource , select an existing Hadoop SQL service.

2. Select Masking , then click Add New Policy.

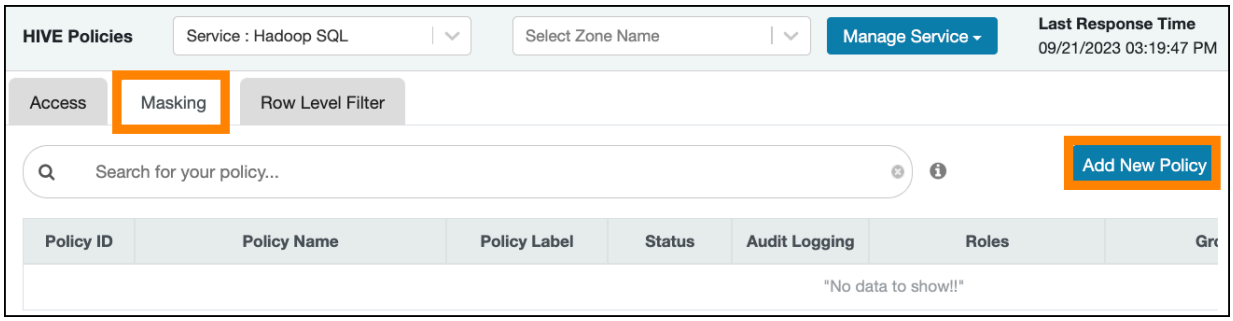

3. On Create Policy, add the following information for the column-masking filter:

Table 51: Policy Details

Field	Description
Policy Name (required)	Enter an appropriate policy name. This name cannot be duplicated across the system. The policy is enabled by default.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
Hive Database (required)	Type in the applicable database name. The auto-complete feature displays available databases based on the entered text.
Hive Table (required)	Type in the applicable table name. The auto-complete feature displays available tables based on the entered text.
Hive Column (required)	Type in the applicable column name. The auto-complete feature displays available columns based on the entered text.
Audit Logging	Audit Logging is set to Yes by default. Select No to turn off audit logging.
Description	Enter an optional description for the policy.
Add Validity Period	Specify a start and end time for the policy.

Table 52: Mask Conditions

Label	Description
Select Role	Specify the roles to which this policy applies.
Select Group	Specify the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify one or more users to which this policy applies.
Access Types	Currently select is the only available access type.

Label	Description
Select Masking Type	To create a row filter for the specified users, groups, and roles, click Select Masking Option, then select a masking type: • Redact – mask all alphabetic characters with "x" and all numeric characters with "n". • Partial mask: show last 4 – Show only the last four characters. • Partial mask: show first 4 – Show only the first four characters. • Hash – Replace all characters with a hash of entire cell value. • Nullify – Replace all characters with a NULL value. • Unmasked (retain original value) – No masking is applied. • Date: show only year – Show only the year portion of a date string and default the month and day to 01/01 • Custom – Specify a custom masked value or expression. Custom masking can use any valid Hive UDF (Hive that returns the same data type as the data type in the column being masked). Masking conditions are evaluated in the order listed in the policy. The condition at the top of the Masking Conditions list is applied first, then the second, then the third, and so on.

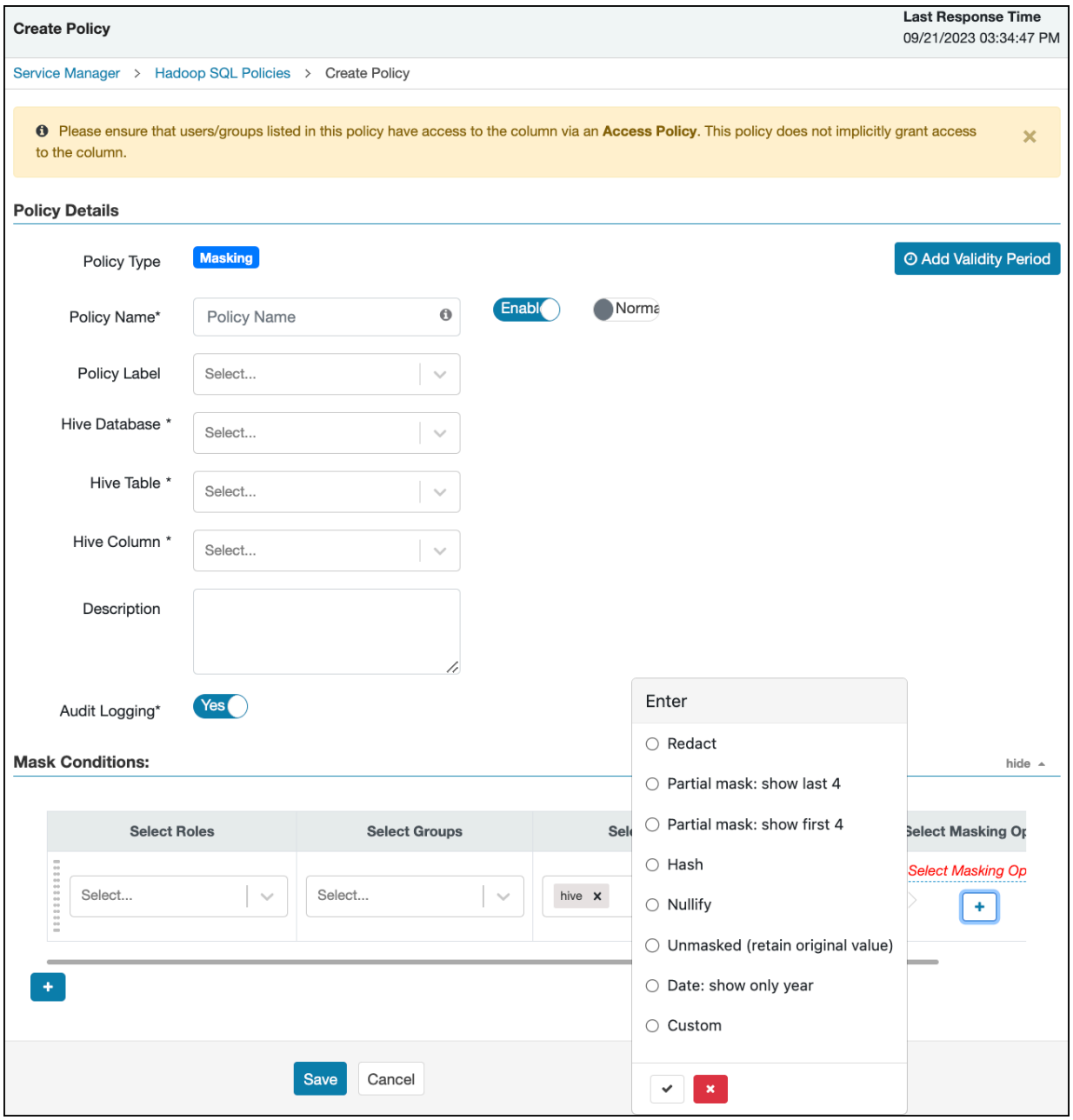
Create Policy
Last Response Time
09/21/2023 03:34:47 PM
Service Manager > Hadoop SQL Policies > Create Policy
Please ensure that users/groups listed in this policy have access to the column via an Access Policy. This policy does not implicitly grant access to the column.
Policy Details
Policy Type
Masking
Add Validity Period
Policy Name*
Policy Name
Enabl
Norma
Policy Label
Select...
Hive Database *
Select...
Hive Table *
Select...
Hive Column *
Select...
Description
Audit Logging*
Yes
Enter
Redact
Partial mask: show last 4
Partial mask: show first 4
Hash
Nullify
Unmasked (retain original value)
Date: show only year
Custom
Mask Conditions:
hide
Select Roles
Select Groups
Select Masking Op
Select...
Select...
hive
Select Masking Op
Save
Cancel

4. To move a condition in the Mask Conditions list (and therefore change the order in which the list is evaluated), click the dotted rows icon at the left of the condition row, then drag the condition to a new position in the list.

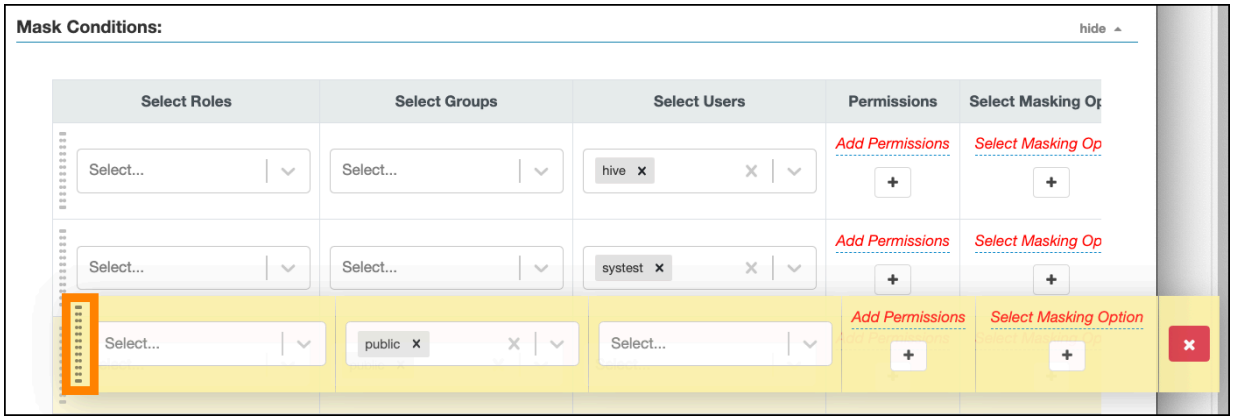

5. On Create Policy, click Add to add the new column masking filter policy.

Dynamic tag-based column masking in Hive with Ranger policies

Where Ranger resource-based masking policy for Hive anonymizes data from a Hive column identified by the database, table, and column, Ranger tag-based masking policy anonymizes Hive column data based on tags and tag attribute values associated with Hive column (usually specified as metadata classification in Atlas).

About this task

The following conditions apply when using Ranger column masking policies to mask data returned in Hive query results:

- A variety of masking types are available, such as show last 4 characters, show first 4 characters, Hash, Nullify, and date masks (show only year).

  Note: Ranger depends on Hive/Impala's hashing functions for hash masking.

  - Impala uses sha512 in FIPS mode, sha256 in non-FIPS mode.
  - Hive uses sha256. Plans to update to sha512 in FIPS mode.
- You can specify a masking type for specific users, groups, and conditions.
- Wildcard matching is not supported.
- If there are multiple tag masking policies applied to the same Hive column, the masking policy with the lexicographically smallest policy-name is chosen for enforcement, E.G., policy "a" is enforced before policy "aa".
- Masks are evaluated in the order listed in the policy.
- An audit log entry is generated each time a masking policy is applied to a column.

Procedure

1. On Service Manager Tag , select a tag-based service.
2. On TAG Policies, select Masking, then click Add New Policy.

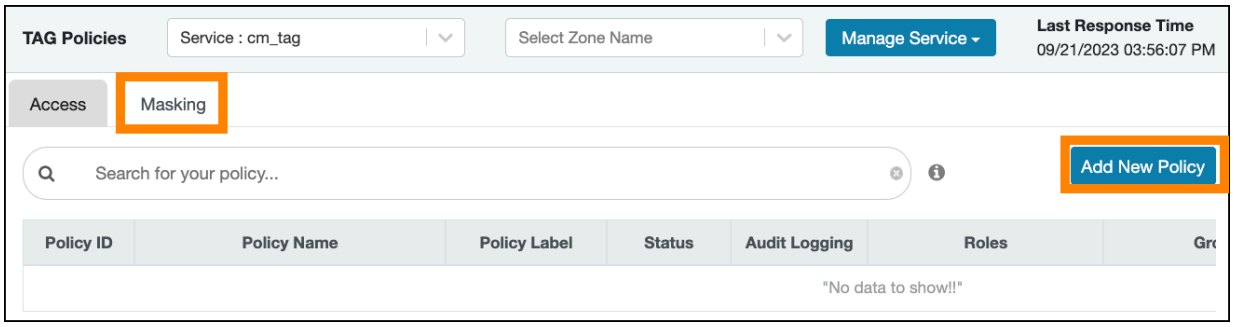

3. In Create Policy, add the following information for the column-masking filter:

Table 53: Policy Details

Field	Description
Policy Type (required)	Set to Masking by default.
Policy Name (required)	Enter an appropriate policy name. This name cannot be duplicated across the system. The policy is enabled by default.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
TAG (required)	Enter the applicable tag name, for example MASK.
Audit Logging	Audit Logging is set to Yes by default. Select No to turn off audit logging.
Description	Enter an optional description for the policy.
Add Validity Period	Specify a start and end time for the policy.
Policy Conditions (applied at the policy level)	Click + to add policy conditions. Currently "Accessed after expiry_date? (yes/no)" is the only available policy condition. "Accessed after expiry_date (yes/no)?": To set this condition, type yes in the text box, then click check mark to add the condition. Enter boolean expression: Available for allow or deny conditions on tag-based policies. For examples and details, see “Using Tag Attributes and Values in Ranger Tag-Based Policy Conditions”. Click Save to save the policy condition.

Table 54: Mask Conditions

Label	Description
Select Role	Specify the roles to which this policy applies.
Select Group	Specify the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify one or more users to which this policy applies.
Policy Conditions (applied at the item level)	Click Add Conditions to add policy conditions. Currently "Accessed after expiry_date? (yes/no)" is the only available policy condition. "Accessed after expiry_date (yes/no)?": To set this condition, type yes in the text box, then click check mark to add the condition. Enter boolean expression: Available for allow or deny conditions on tag-based policies. For examples and details, see “Using Tag Attributes and Values in Ranger Tag-Based Policy Conditions”.
Permissions - Access Types	Currently select is the only available access type for the hive component.

Label	Description
Select Masking Option	To create a row filter for the specified users, groups, and roles, click Select Masking Option, then select a masking type: • Redact – mask all alphabetic characters with "x" and all numeric characters with "n". • Partial mask: show last 4 – Show only the last four characters. • Partial mask: show first 4 – Show only the first four characters. • Hash – Replace all characters with a hash of entire cell value. • Nullify – Replace all characters with a NULL value. • Unmasked (retain original value) – No masking is applied. • Date: show only year – Show only the year portion of a date string and default the month and day to 01/01 • Custom – Specify a custom masked value or expression. Custom masking can use any valid Hive UDF (Hive that returns the same data type as the data type in the column being masked). Masking conditions are evaluated in the order listed in the policy. The condition at the top of the Masking Conditions list is applied first, then the second, then the third, and so on.

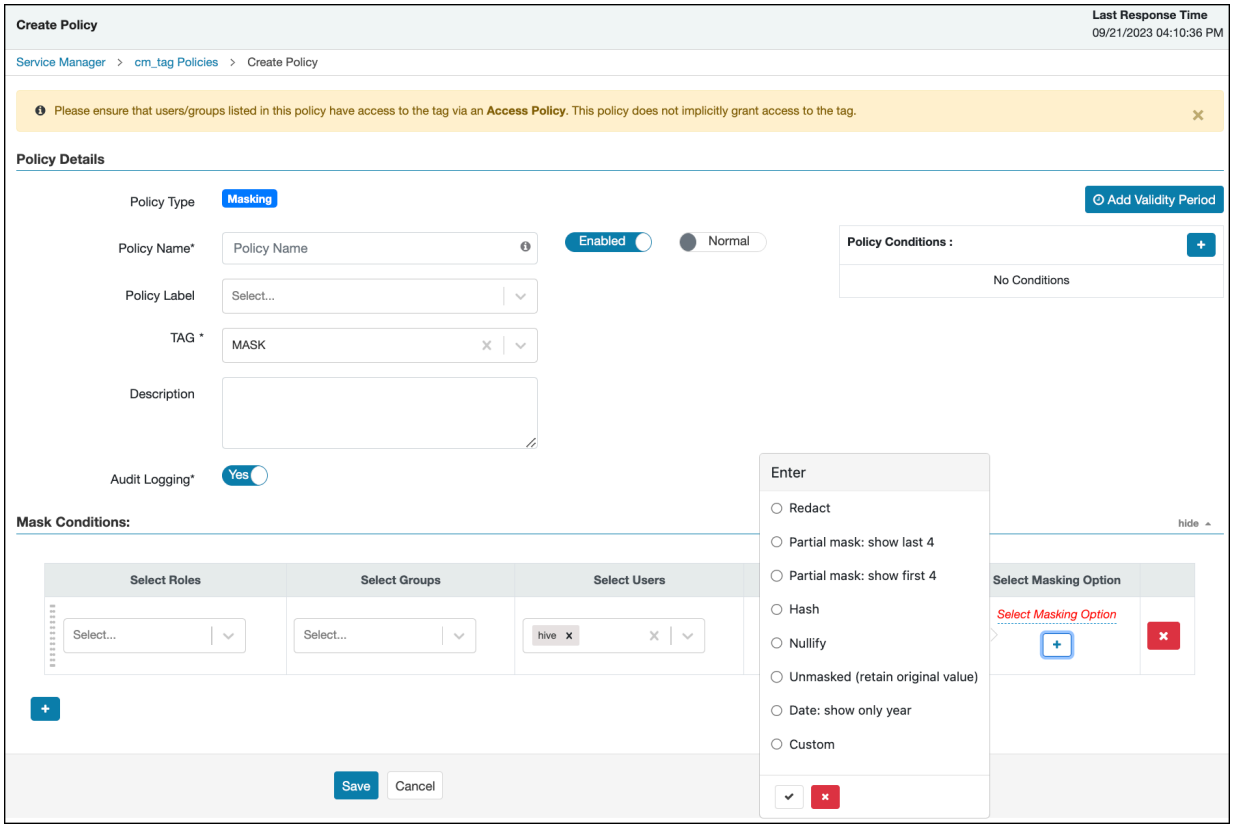

4. Click + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. Click Add to add the new policy.

Related Information

Using tag attributes and values in Ranger tag-based policy conditions

Tag-based Services and Policies

Ranger enables you to create tag-based services and add access policies to those services.

Adding a tag-based service

How to add a tag-based service to Ranger.

About this task

You can use Ranger Admin Web UI Service Manager Tag Policies to create tag-based services and add tag-based access policies that can be applied to Cloudera resources. Using tag-based policies enables you to control access to resources across multiple components without creating separate services and policies in each component. You can also use Ranger TagSync to synchronize the Ranger tag store with an external metadata service such as Apache Atlas.

Procedure

1. Select Ranger Admin Web UI Service Manager Tag Policies , then click +.

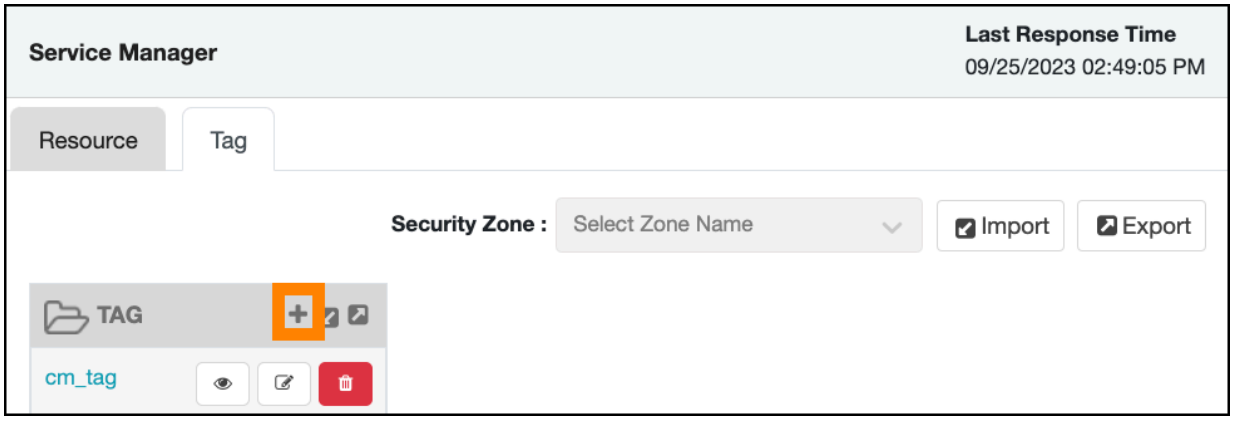

2. On Create Service, type in a service name and an optional description. The service is enabled by default, but you can disable it by selecting Disabled. To add the service, click Add.

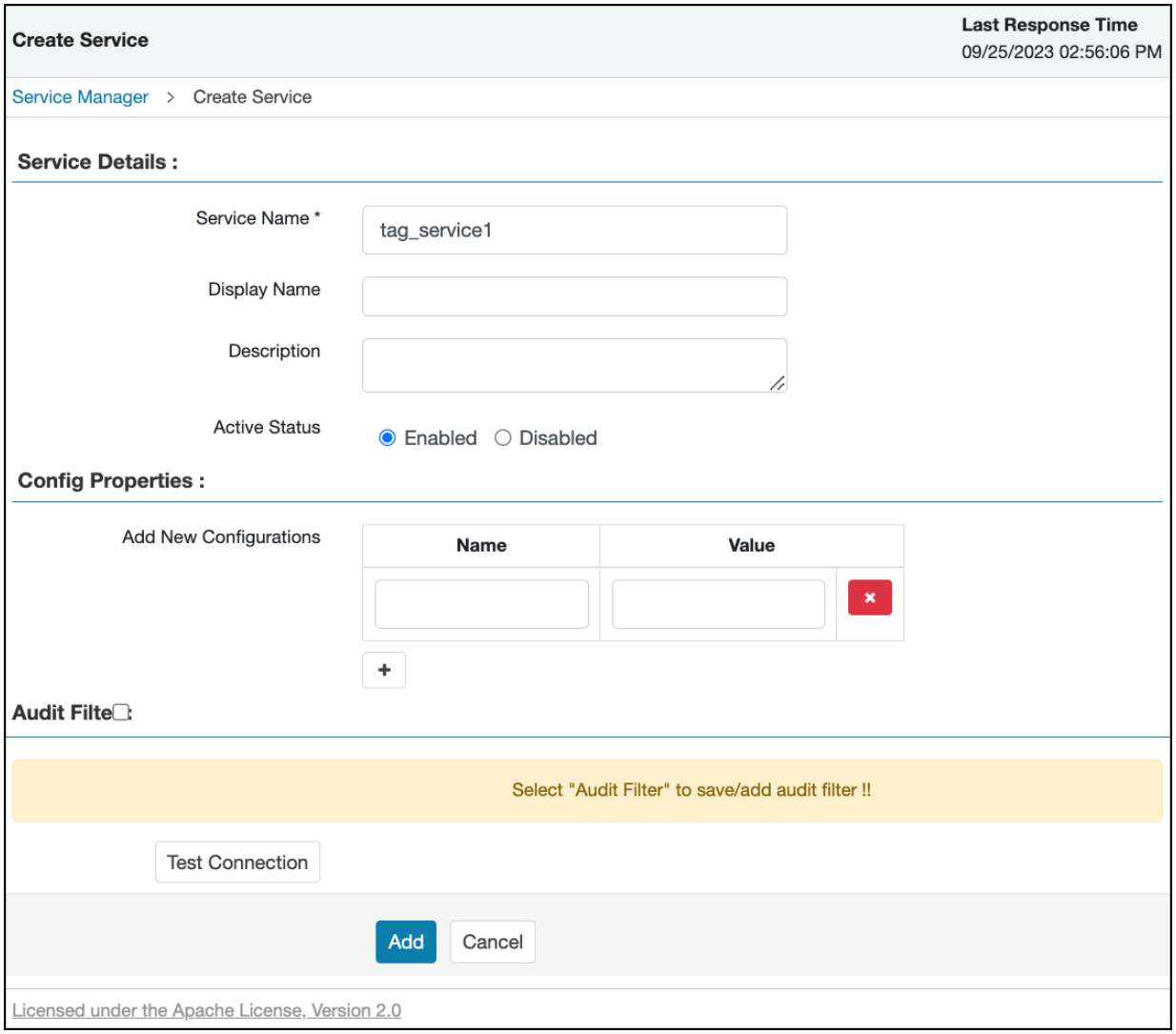

3. The new tag service appears in Service Manager.

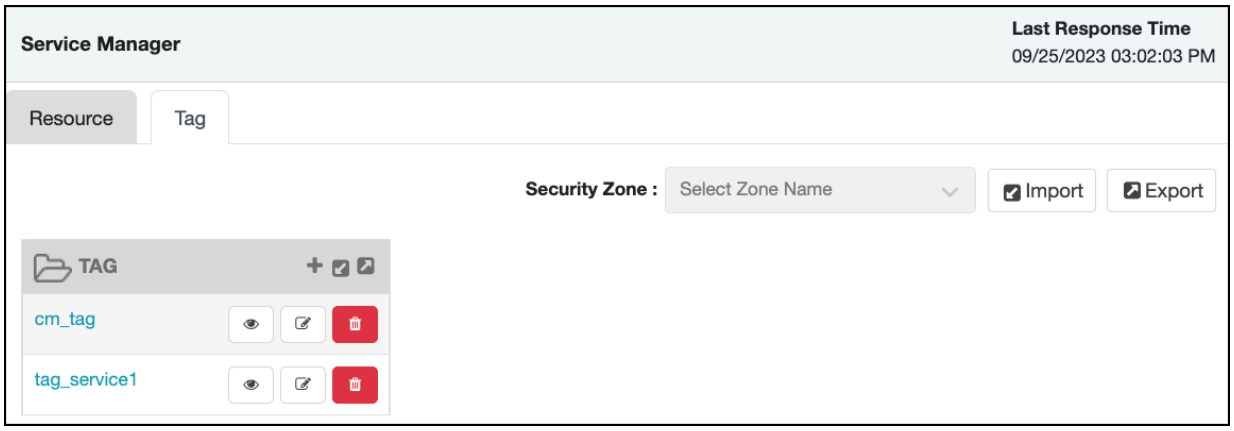

Adding tag-based policies

Tag-based policies enable you to control access to resources across multiple Hadoop components without creating separate services and policies in each component. You can also use Ranger TagSync to synchronize the Ranger tag store with an external metadata service such as Apache Atlas.

Procedure

1. Select Service Manager Tag Policies , then select a tag-based service.
2. List of Policies displays existing Access policies by default. Click Add New Policy.

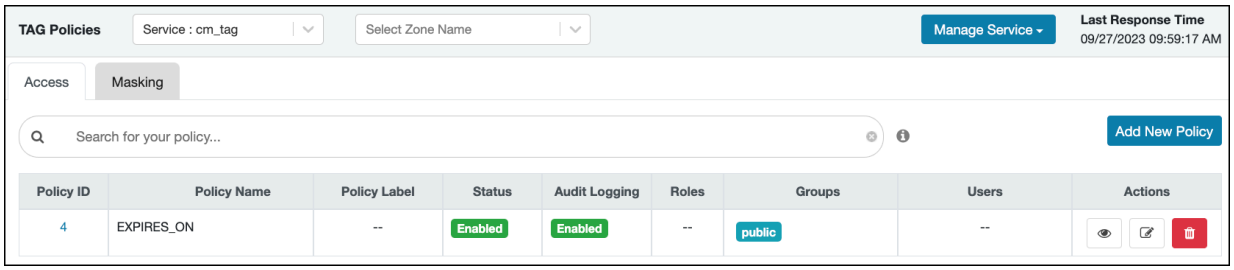

Create Policy displays controls for creating details for a new policy.

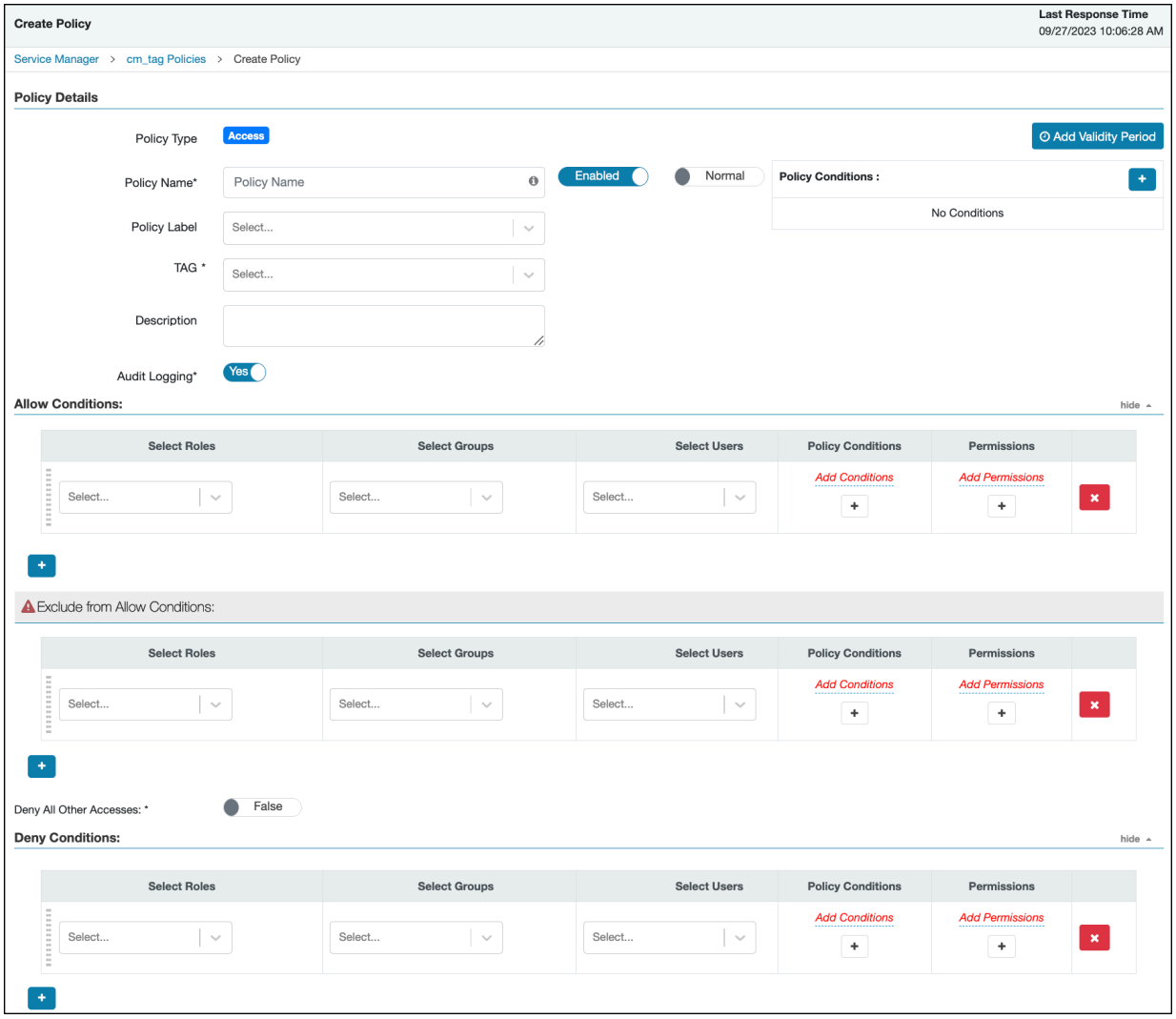

3. Complete the Create Policy page as follows:

Table 55: Policy Details

Field	Description
Policy Type	Set to Access by default.
Policy Name	Enter a unique policy name. This name cannot be duplicated across the system. This field is mandatory.
normal/override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
TAG	Enter the applicable tag name.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).
Policy Label	Specify a label for this policy. You can search reports and filter policies based on these labels.
Add Validity Period	Specify a start and end time for the policy.
Policy Conditions (applied at the policy level)	Click the + icon to add policy conditions. Currently "Accessed after expiry_date? (yes/no)" is the only available policy condition. "Accessed after expiry_date (yes/no)?": To set this condition, type yes in the text box, then click the check mark button to add the condition. Enter boolean expression: Available for allow or deny conditions on tag-based policies. For examples and details, see “Using Tag Attributes and Values in Ranger Tag-Based Policy Conditions”. Click Save to save the policy condition.

Table 56: Allow, Exclude from Allow, Deny, and Exclude from Deny Conditions

Label	Description
Select Role	Specify the roles to which this policy applies.
Select Group	Specify the group to which this policy applies. To designate the group as an Administrator for the chosen resource, specify Admin permissions. (Administrators can create child policies based on existing policies). The public group contains all users, so setting a condition for the public group applies to all users.
Select User	Specify a particular user to which this policy applies (outside of an already-specified group) OR designate a particular user as Admin for this policy. (Administrators can create child policies based on existing policies).
Policy Conditions (applied at the item level)	Click Add Conditions to add policy conditions. Currently "Accessed after expiry_date? (yes/no)" is the only available policy condition. "Accessed after expiry_date (yes/no)?": To set this condition, type yes in the text box, then click the check mark button to add the condition. Enter boolean expression: Available for allow or deny conditions on tag-based policies. For examples and details, see “Using Tag Attributes and Values in Ranger Tag-Based Policy Conditions”.

Label	Description
Component Permissions	Click Add Permissions to add or edit component conditions. To add component permissions, enter the component name in the text box, then use the check boxes to specify component permissions. Click the check mark button to add the chosen component conditions to the policy.

4. You can use + to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. You can use Deny All Other Accesses to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.
6. Click Add to add the new policy.

Related Information

Using tag attributes and values in Ranger tag-based policy conditions

Using tag attributes and values in Ranger tag-based policy conditions

Enter boolean expression allows Ranger to use tag attributes and values when configuring tag-based policy Allow or Deny conditions. It allows admins to provide boolean expression(s) using tag attributes.

The policy condition is introduced in the tag service definition:

```
{
      "itemId":2,
      "name":"expression",
      "evaluator": "org.apache.ranger.plugin.conditionevaluator.RangerScr
iptConditionEvaluator",
      "evaluatorOptions" : {"engineName":"JavaScript", "ui.isMultiline":"tru
e"},
      "label":"Enter boolean expression",
      "description": "Boolean expression"
    }
```

The following variables can be referenced in the boolean expression:

- ctx: Context handler containing APIs to access metadata information from the request.
- tag: Information about the current tag.
- tagAttr: Map containing all the current tag attributes and corresponding values.

The following APIs available from the request:

- getUser(): Returns a string.
- getUserGroups(): Returns a set of strings containing groups.
- getClientIPAddress(): Returns a string containing client IP address.
- getAction(): Returns a string containing information about the action being requested.

For two scenarios:

- User “sam” needs to be denied a policy based on the IP address of the machine from where the resources are accessed.

  Set the deny condition for user sam with the following boolean expression:

  ```
  if ( tagAttr.get('ipAddr').equals(ctx.getClientIPAddress()) ) {
   ctx.result = true;
     }
  ```

- Deny one particular user, “bob” from a group, “users”, only when this user is accessing resources from a particular IP defined as an tag attribute in Atlas.

  Set the deny condition for group users with the following boolean expression:

```
if (tagAttr.get('ipAddr').equals(ctx.getClientIPAddress()) && ctx.getUse
r().equals("bob"))  {
 ctx.result=true;
      }
```

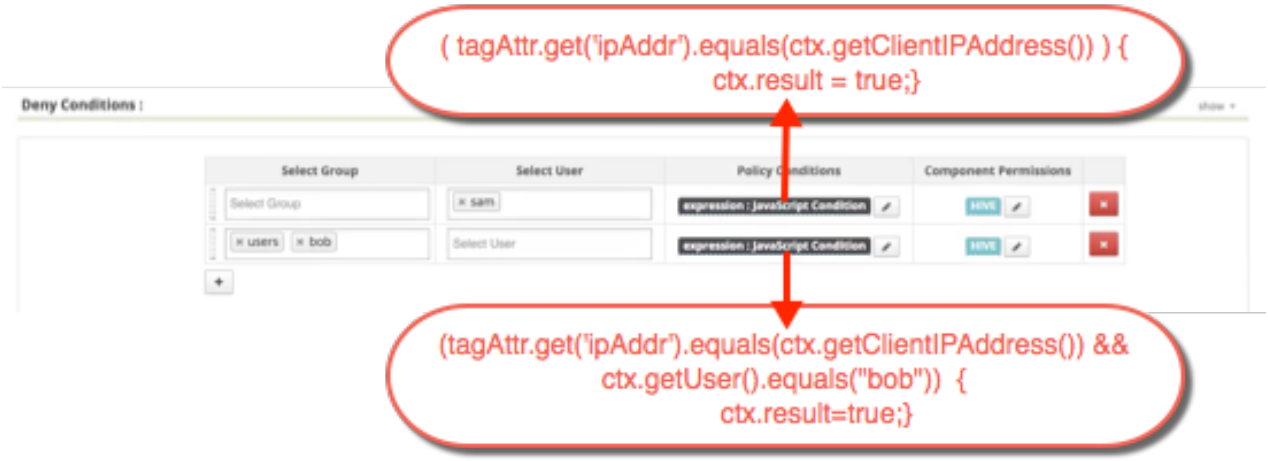

Adding a policy condition to a tag-based policy

You can add a condition to a tag-based policy, using Ranger Admin Web UI when creating a new, or editing an existing policy.

About this task

Ranger Admin Web UI supports adding one of the following policy conditions to a new or existing tag-based policy:

- Accessed after expiry_date ? for example - Yes/No
- Boolean expression for example - Country_Name="XYZ"

The Policy Conditions dialog prompts for inputs using uhint JSON. For populating "Accessed after expiry_date? (yes/no)" for example, we are using JSON like this:

```
{
    "itemId": 1,
    "name": "accessed-after-expiry",
    "evaluator": "org.apache.ranger.plugin.conditionevaluator.RangerScriptTe
mplateConditionEvaluator",
    "evaluatorOptions": {
        "scriptTemplate": "ctx.isAccessedAfter('expiry_date');"
    },
    "uiHint": "{ \"singleValue\":true }",
    "label": "Accessed after expiry_date (yes/no)?",
    "description": "Accessed after expiry_date? (yes/no)"
}
```

Procedure

1. In Service Manager Tag Policies cm_tag_policies , choose one of the following:

   Add New Policy

   to add a new, tag-based policy.

   Policy ID

   click a policy ID to edit an existing policy.

2. In either Create Policy or Edit Policy Policy Conditions , click +.

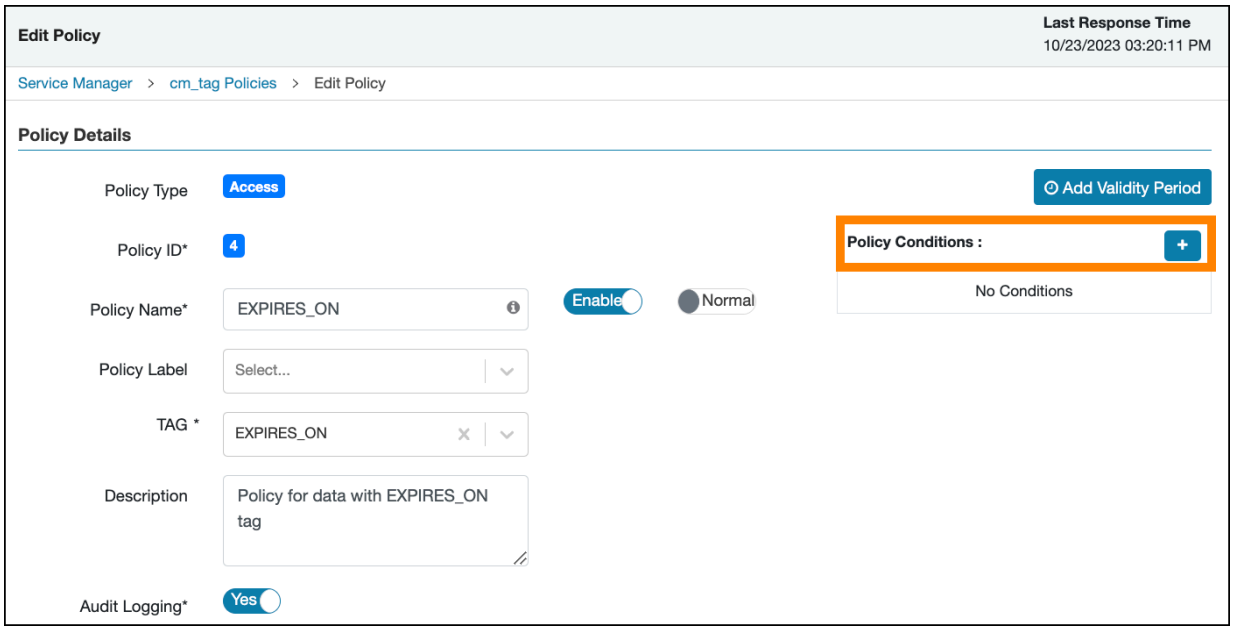

3. In Policy Conditions:
   a) In Accessed after expiry date ?, select Yes or No.
   b) In Enter boolean expression, enter an expression that evaluates to true or false.
      Country_Name="XYZ"
4. Click Save.

Adding a tag-based PII policy

Example of how to add a PII tag-based policy. In this example we create a tag-based policy for objects tagged "PII" in Atlas. Access to objects tagged "PII" is allowed for members of the "audit" group. All other users (the "public" group) are denied access.

Procedure

1. Select Service Manager Tag Policies , then select a tag-based service.

2. List of Policies displays existing Access policies by default. Click Add New Policy.

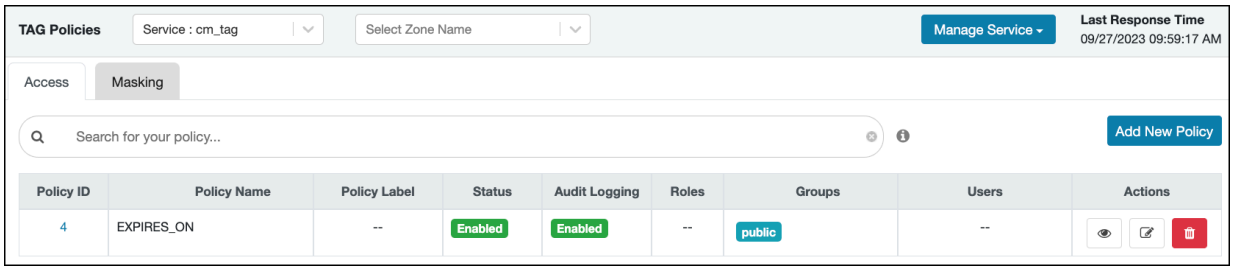

Create Policy displays controls for creating details for a new policy.

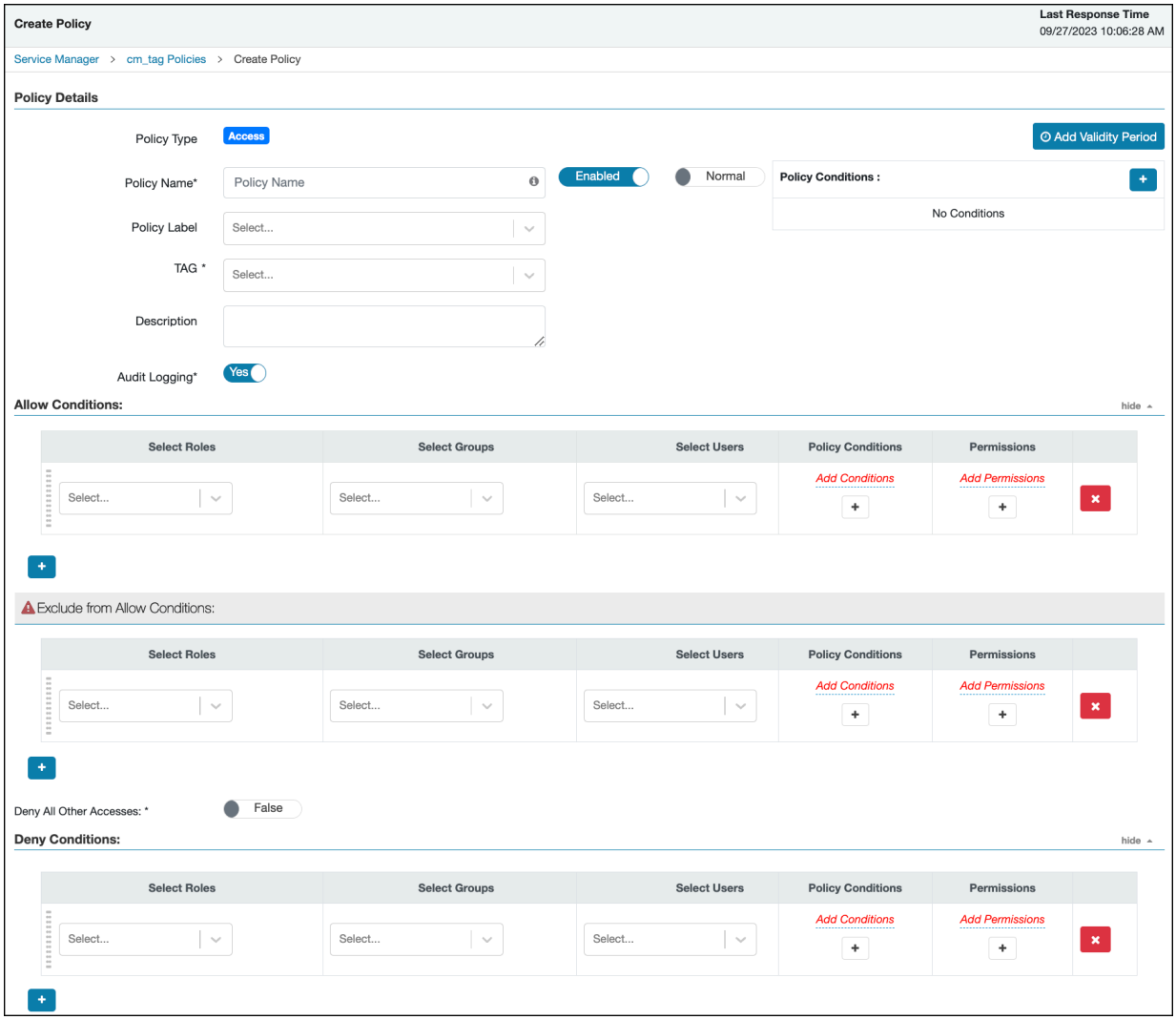

3. Complete the Create Policy page as follows:

Table 57: Policy Details

Field	Description
Policy Type	Set to Access by default.
Policy Name	PII
TAG	PII
Audit Logging	YES

Field	Description
Description	Restrict access to resources with the PII tag.

Table 58: Allow Conditions

Label	Description
Select Group	audit
Select User	<none>
Policy Conditions	<none>
Component Permissions	hive (select all permissions)

Table 59: Deny Conditions

Label	Description
Select Group	public
Select User	<none>
Policy Conditions	<none>
Component Permissions	hive (select all permissions)

Table 60: Exclude from Deny Conditions

Label	Description
Select Group	audit
Select User	<none>
Policy Conditions	<none>

Label	Description
Component Permissions	hive (select all permissions)

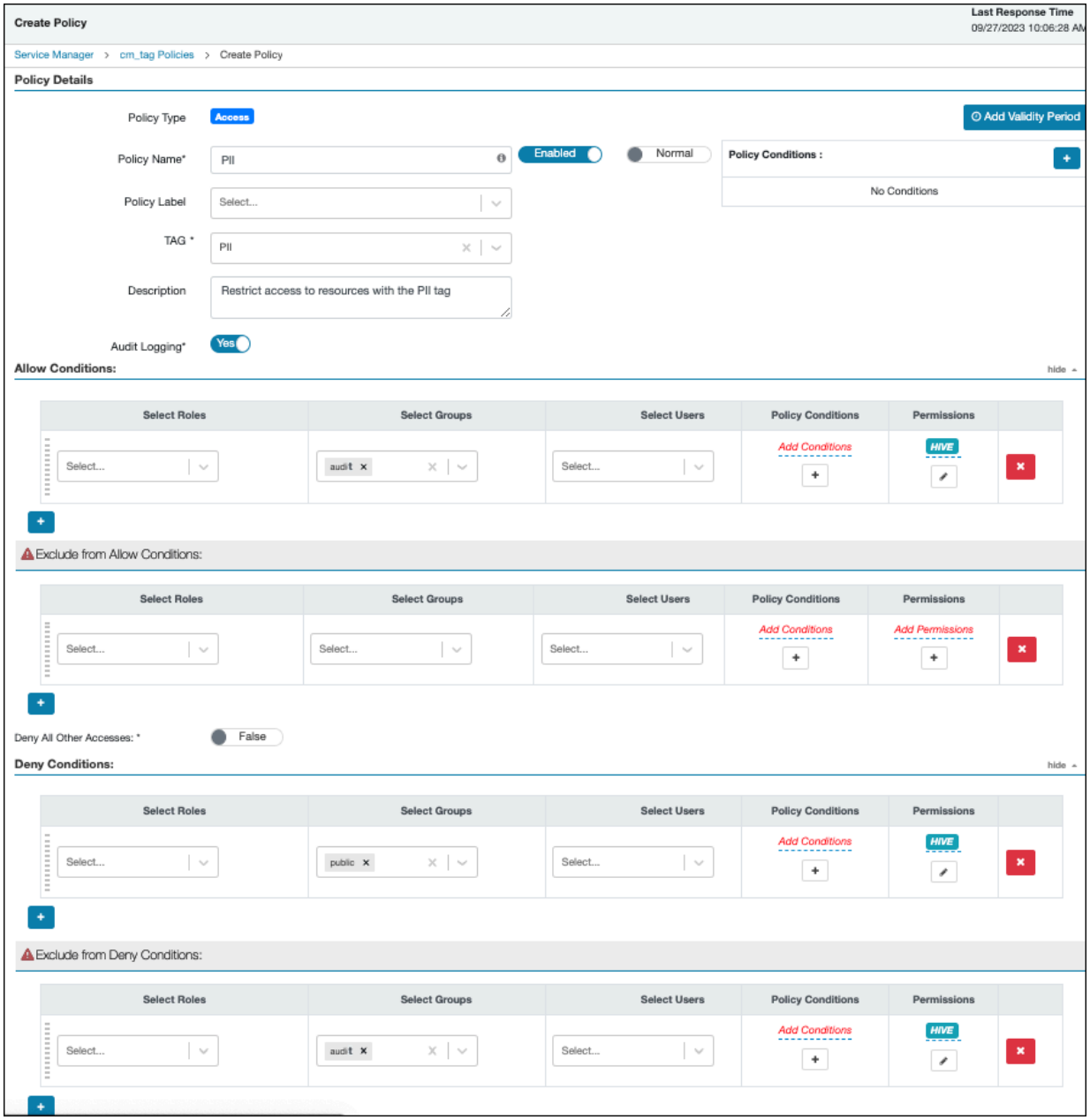

In this example we used Allow Conditions to grant access to the "audit" group, and then used Deny Conditions to deny access to the "public" group. Because the "public" group includes all users, we then used Exclude from Deny Conditions to exclude the "audit" group, in effect reinstating the "audit" group's original Allow access condition.

4. Click Add to add the new policy.

Default EXPIRES ON tag policy

An EXPIRES_ON tag-based policy is created automatically when a tag service instance created. This default policy denies access to objects tagged with EXPIRES_ON after the expiry date specified in the Atlas tag attribute. You can use the following steps to review the default EXPIRES_ON policy.

Procedure

1. Select Service Manager Tag Policies , then select a tag-based service.
2. On List of Policies, click Edit for the default EXIRES_ON policy.

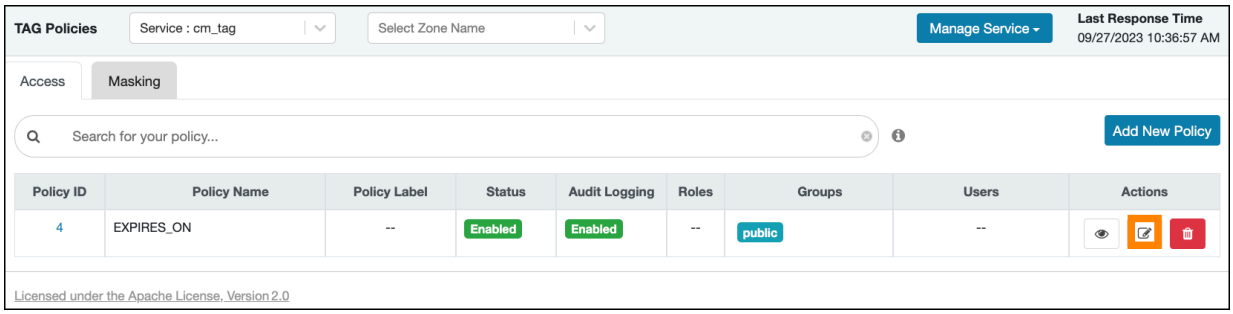

The Edit Policy page appears:

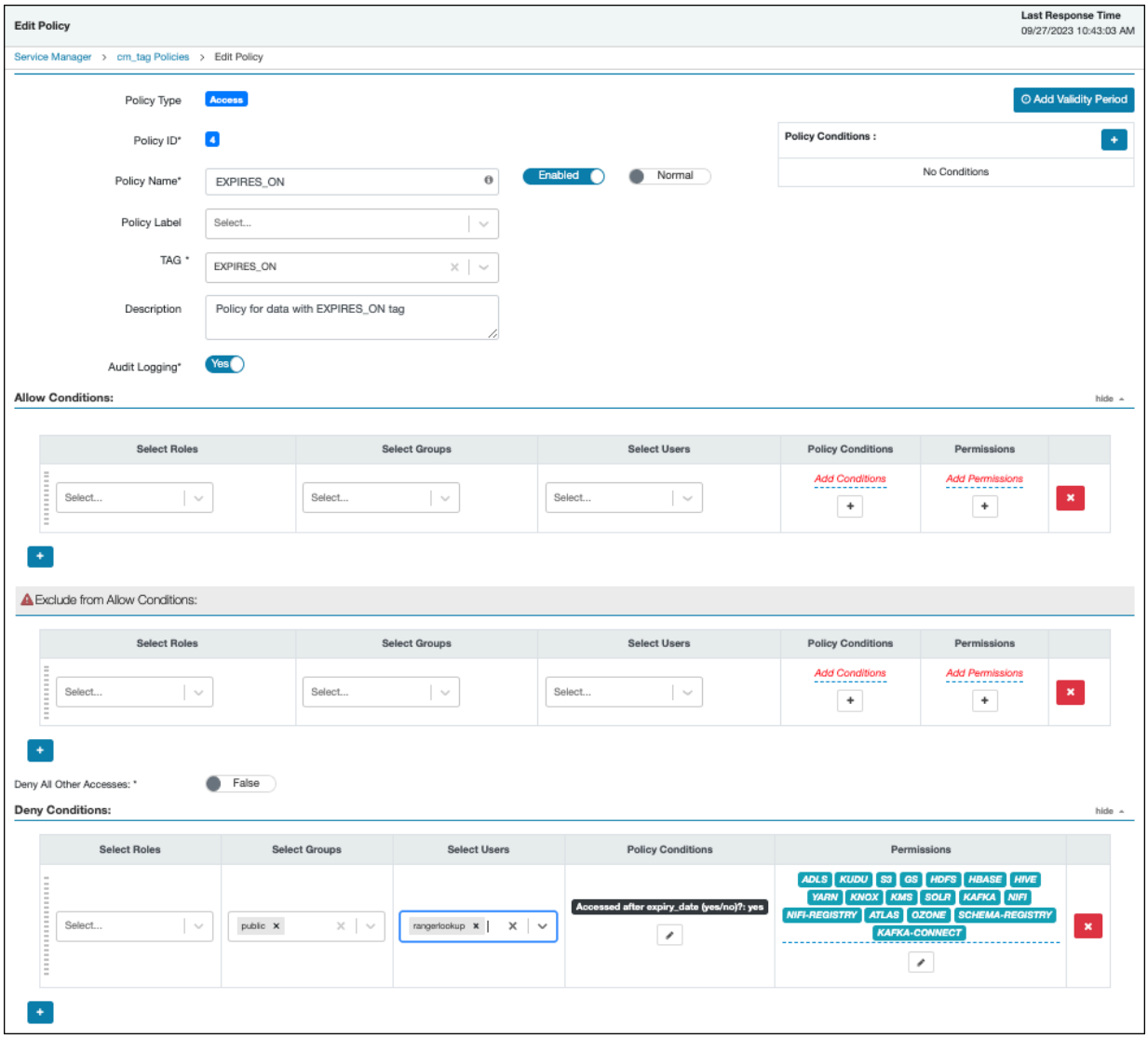

3. We can see that the default EXPIRES_ON policy denies access to all users, and for all components, after the expiry date specified in the Atlas tag attribute.

Importing and exporting tag-based policies

You can export and import policies from the Ranger Admin UI for cluster resiliency (backups), during recovery operations, or when moving policies from test clusters to production clusters. You can import or export a specific subset of policies (such as those that pertain to specific resources or user/groups) or clone the entire repository (or multiple repositories) via the Ranger Admin UI.

Interfaces

You can import and export policies from Service Manager Tag Policies Tag :

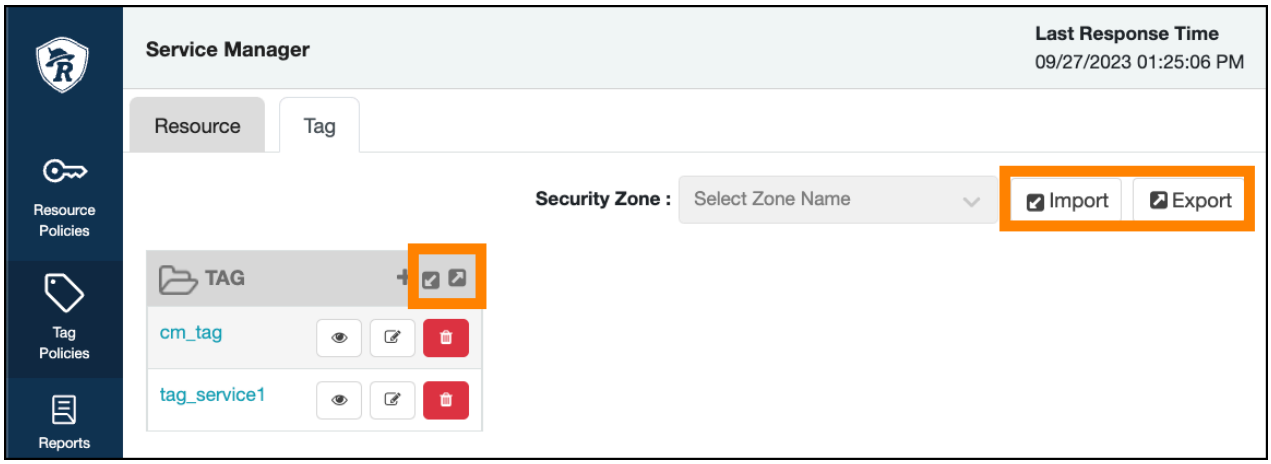

You can also export policies from Reports:

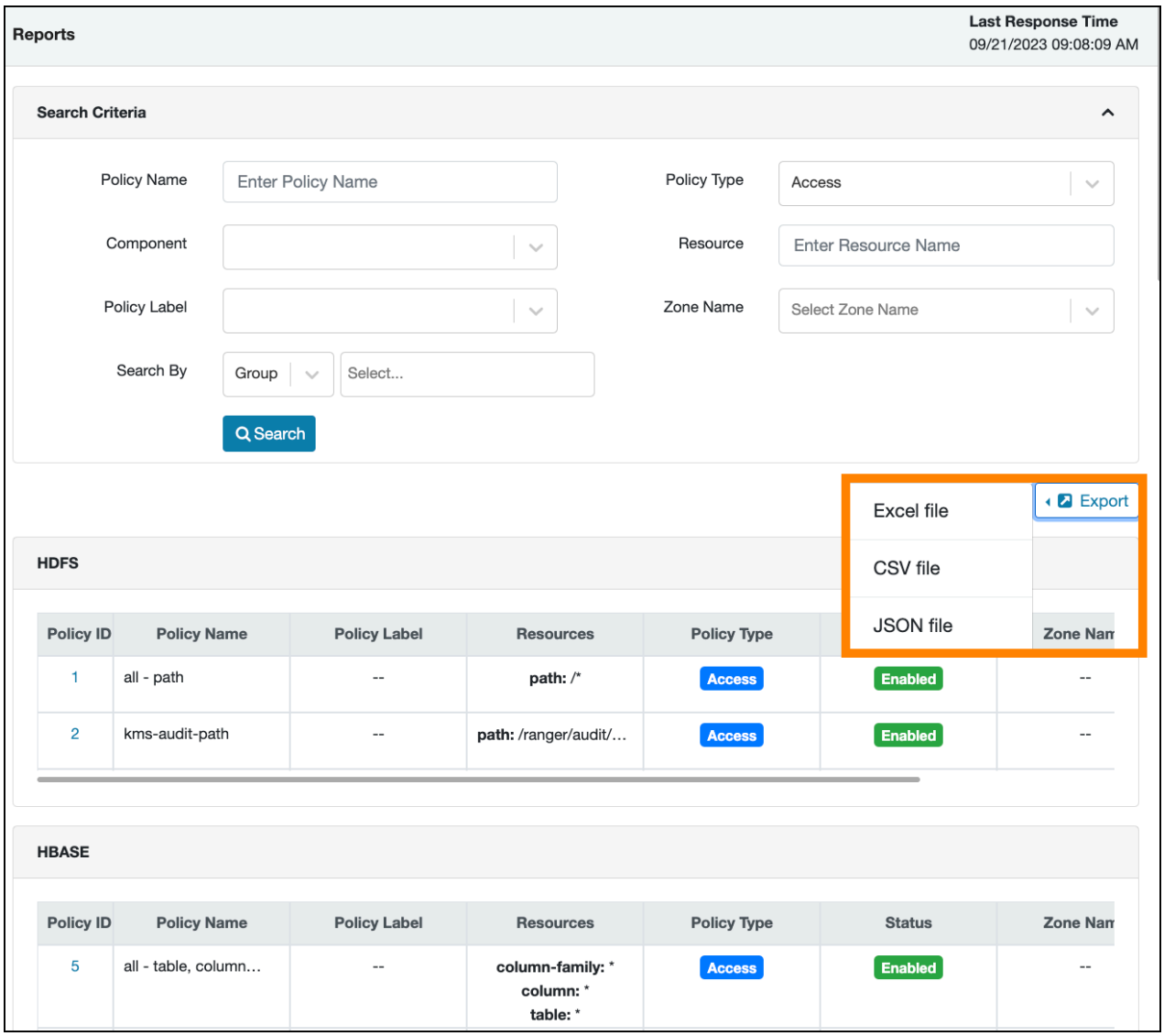

Table 61: Export Policy Options

	Service Manager Page	Reports Page
Formats	JSON	JSON Excel CSV
Filtering Supported	No	Yes
Specific Service Export	Yes	Via filtering

Filtering

When exporting from the Reports page, you can apply filters before saving the file.

Export Formats

You can export policies in the following formats:

- Excel
- JSON

- CSV

> **Note:**
>
> CSV format is not supported for importing policies.

When you export policies from the Service Manager page, the policies are automatically downloaded in JSON format. If you wish to export in Excel or CSV format, export the policies from the Reports page dropdown menu.

Required User Roles

The Ranger admin user can import and export only Resource & Tag based policies. The credentials for this user are set in Ranger Configs Advanced ranger-env in the fields labeled admin_username (default: admin/admin).

The Ranger KMS keyadmin user can import and export only KMS policies. The default credentials for this user are keyadmin/keyadmin.

Limitations

To successfully import policies, use the following database versions:

- MariaDB: 10.1.16+
- MySQL: 5.6.x+
- Oracle: 11gR2+
- PostgreSQL: 8.4+
- MS SQL: 2008 R2+

> **Note:**
>
> Partial policy import is not supported.

Related Information

Importing and exporting resource-based policies

Import tag-based policies

How to import tag-based policies.

Procedure

1. On Service Manager Tag Policies Tag , click one of the Import icons:

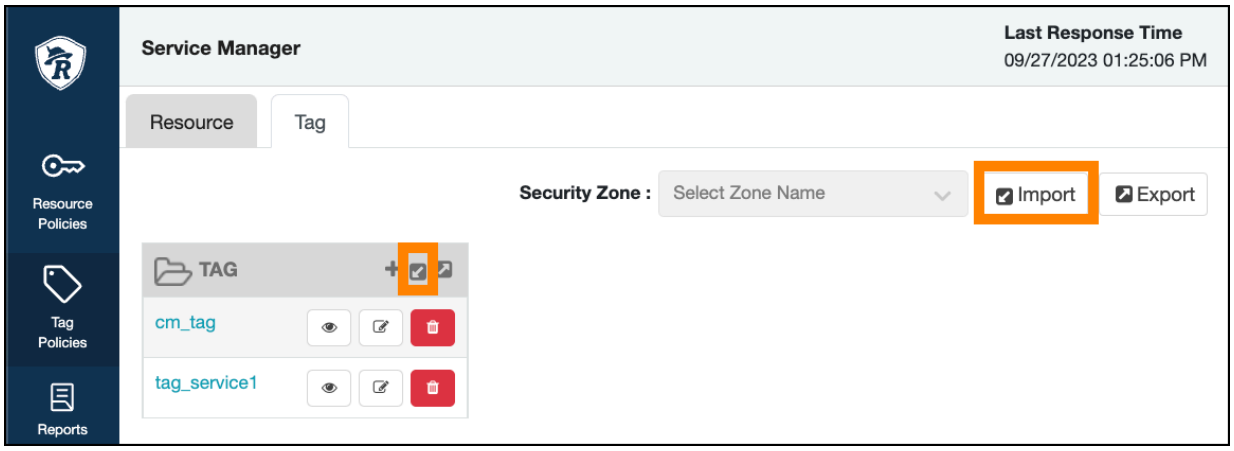

The Import Policy displays options for importing policies.

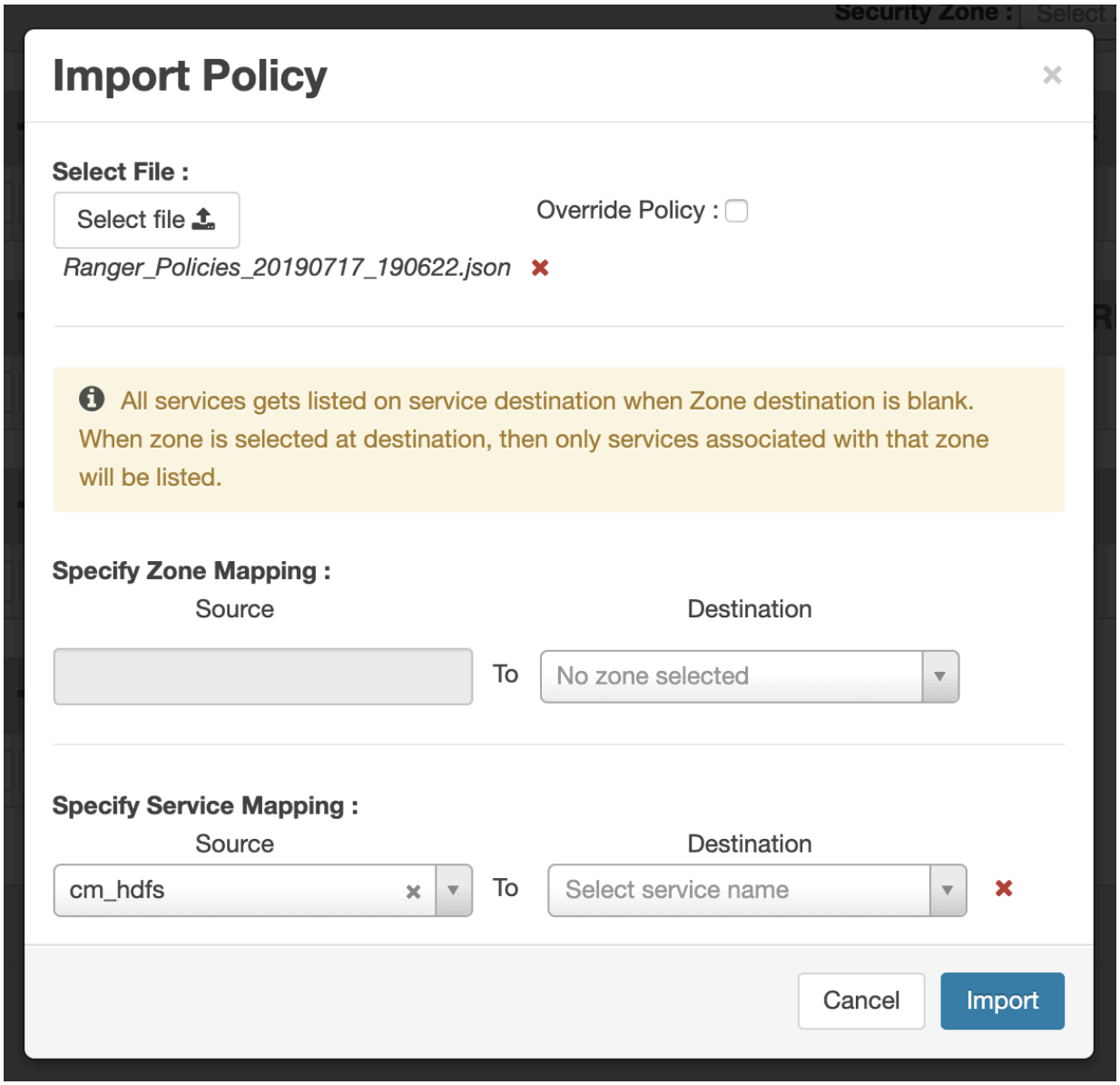

2. Select the file to import.
   You can only import policies in JSON format.
3. (Optional) Configure the import operation:
   a) The Override Policy option deletes all policies of the destination repositories.
   b) Zone Mapping – when no destination is selected, all services are imported. When a destination is selected, only the services associated with that security zone are imported.
   c) Service Mapping maps the downloaded file repository, i.e. source repository to destination repository. You can use the red x symbols to remove services from the import. Scroll down to view all service mappings.

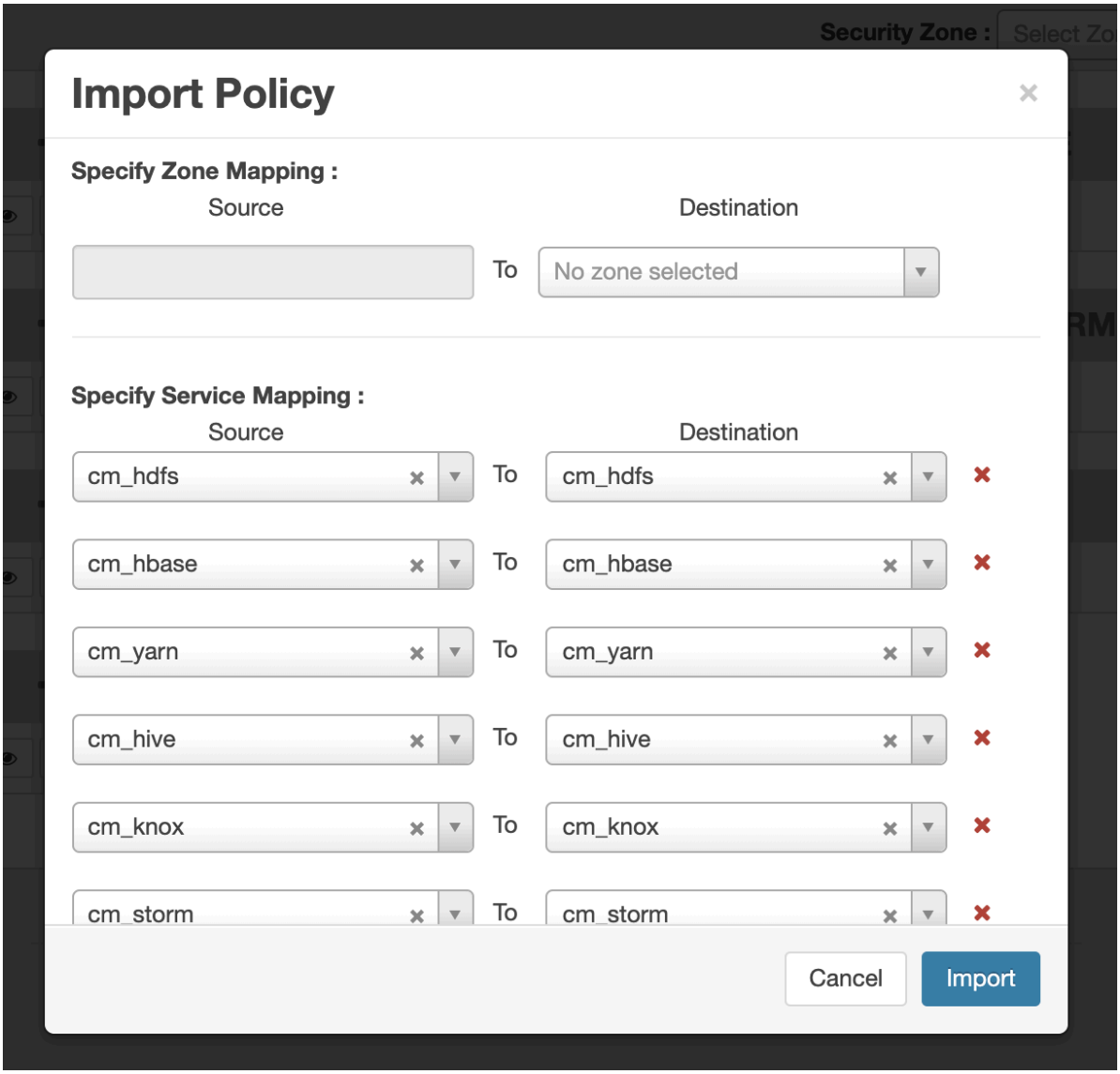

4. Click Import.
   A confirmation message appears after the file is imported.

Related Information

Export tag-based policies

Export tag-based policies

How to export all tag-based policies.

About this task

You can only export policies in JSON format from the Tag-based polices page. If you would like to export in Excel or CSV format, export the policies from the Reports page drop-down menu.

Procedure

- On Service Manager Tag Policies Tag , click one of the Export icons:
  a) Click the Export button or icon:

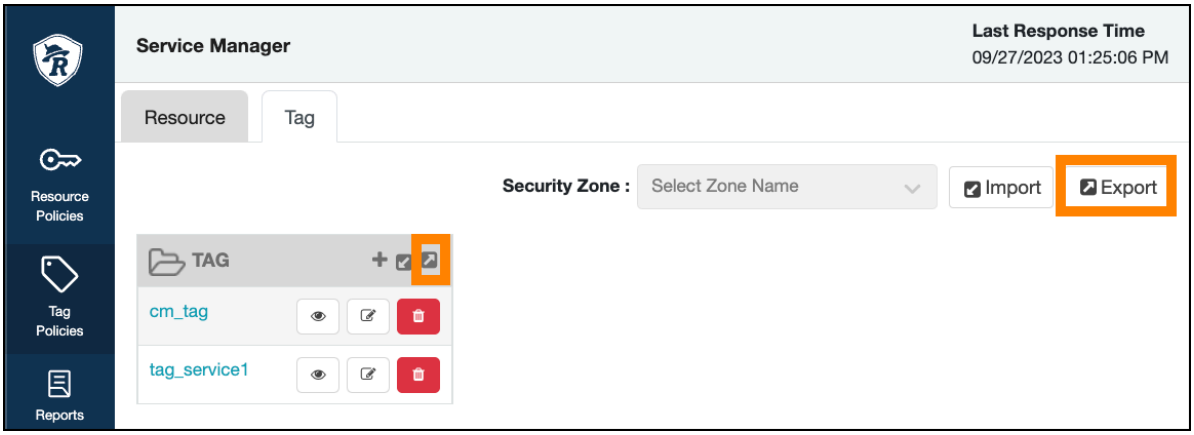

  Export Policy displays options for exporting policies.

  b) Remove components or specific services, then click Export.

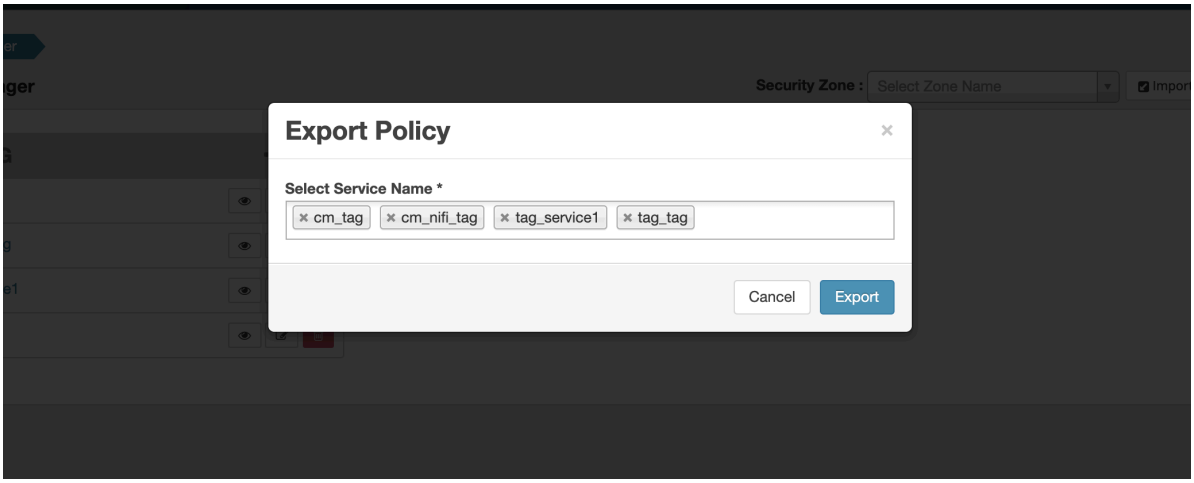

  c) The file downloads in your browser as a JSON file.

- From Reports:
  a) Filter Component to tag and click Search.
  b) (Optional) Apply filters before exporting the file.
  c) Open the Export drop-down menu:

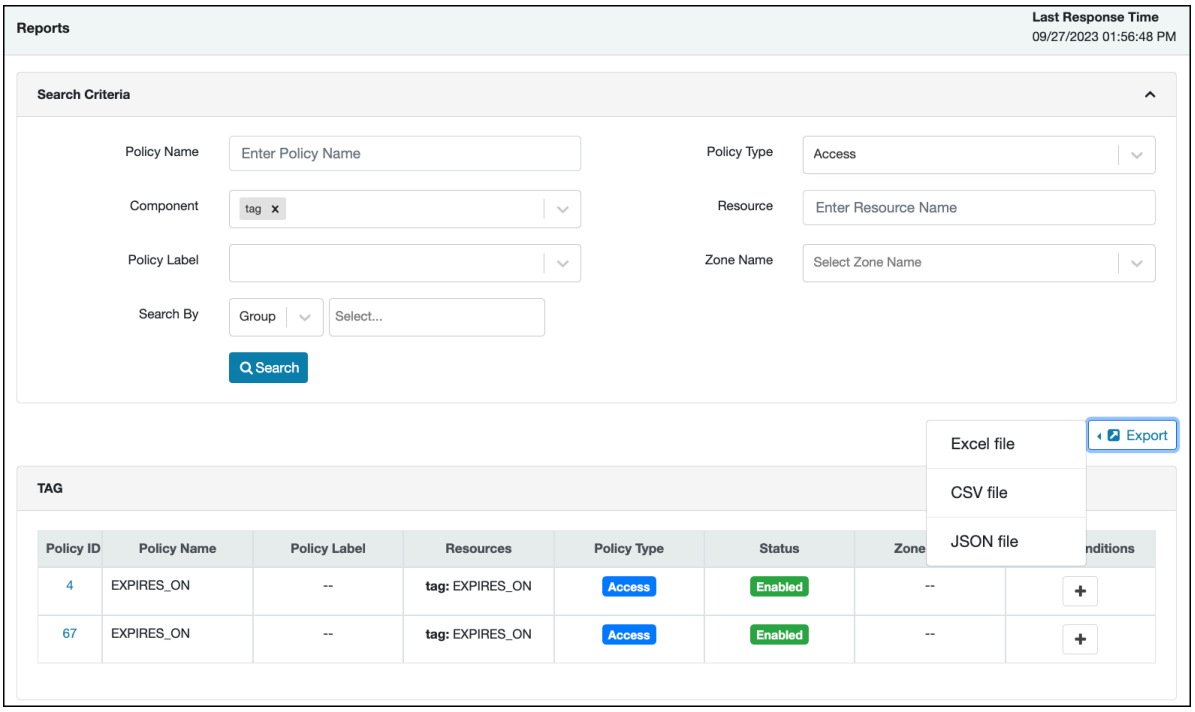

  d) Select the file format.
  The file downloads in your browser.

Create a time-bound policy

Ranger policy validity periods enable you to configure a policy to be effective for a specified time range. You can add a validity period to both resource-based and tag-based policies.

About this task

Time-bound policy use-case examples:

- To restrict access to sensitive financial information until the earnings release date.
- To block a certain user for a specific time period (e.g., a compromised user account being investigated needs to be put on "hold" from accessing resources in Hadoop services).
- To block a certain group for a specific time (e.g., excluding temporary employees from writing on resources during the holiday season).

Note: The following procedure shows how to create a time-bound resource-based policy. The procedure is essentially the same for a tag-based resource policy.

Procedure

1. On Service Manager Resource Policies , select a service.
2. On <Service_name> Policies, click Add New Policy.
3. Complete the fields on Create Policy.
4. Click Add Validity Period.

5. On Policy Validity Period, specify a start time, end time, and time zone. To add additional validity periods, click +. Click Save to save the specified validity periods.

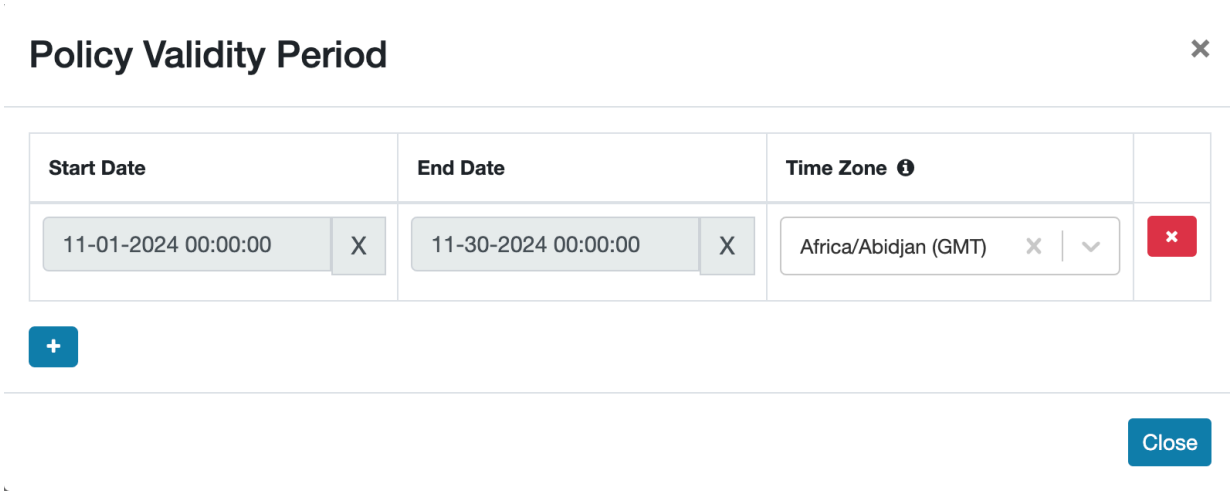

The JSON format for Policy Validity Period appears as follows:

```
"validitySchedules":[{"startTime":"2024/11/01 00:00:00","endTime":"2024/
11/30 00:00:00","timeZone":"Africa/Abidjan"}]
```

6. If you would like the policy to override all other policies during its validity period, select override.

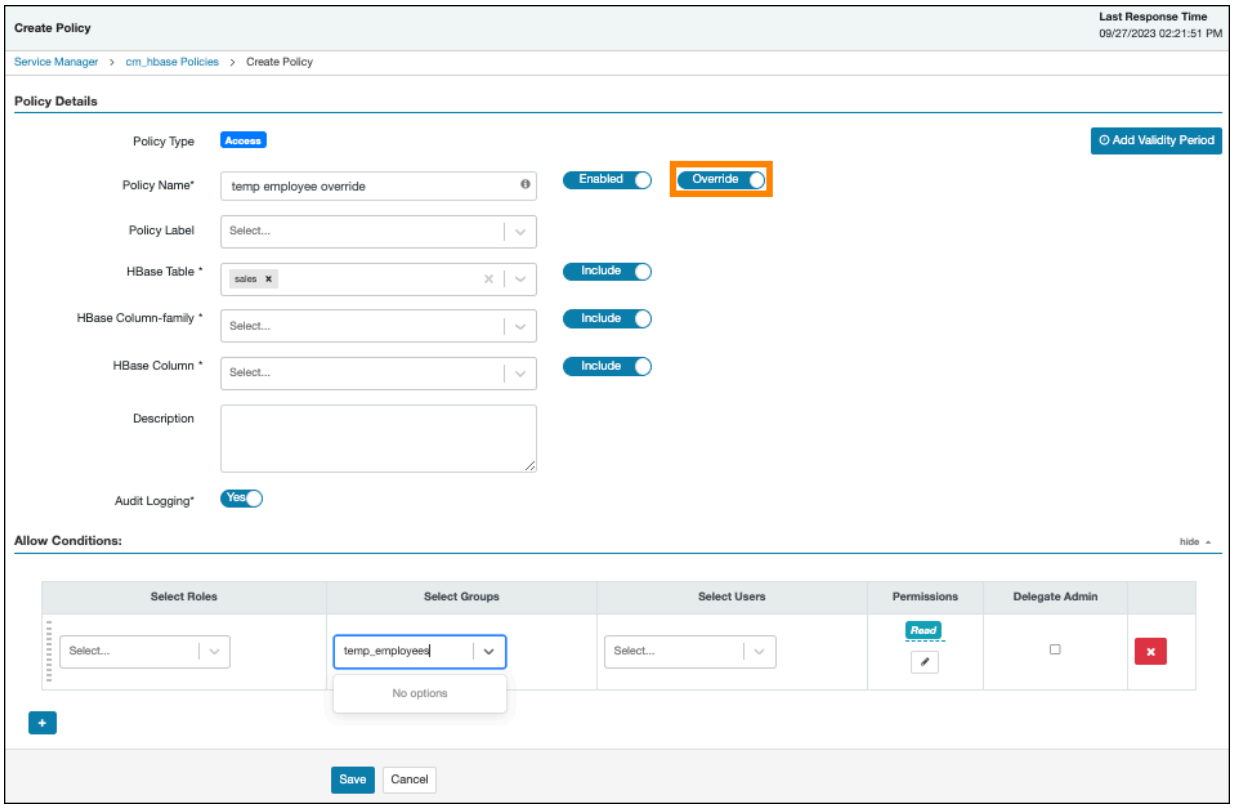

7. Click Add.

Create a Hive authorizer URL policy

You can create a Hive Authorizer URL policy in Ranger that maintains Read and Write permissions for a location or folder.

About this task

Hive supports several commands that include URLs which refer to a current or future data location. Such locations must authorize end user access to that location. Currently, you can create a Ranger HDFS policy that grants "All" permissions for a location, recursively. If no such policy exists, HDFS authorization "falls back" to the current ACL that defines access to a location or folder. By default the value of the parameter is “hdfs:,file:”. If you remove “hdfs:”, access requests will be authorized against the HIVE URL policy and won't check for hdfs plugin or Hadoop ACL. This solution requires maintaining many policies or ACLs at the storage level. You can create a Hive Authorizer URL policy in Ranger that maintains Read and Write permissions for a location or folder.

To create a Hive Authorizer policy:

Procedure

1. In Cloudera Manager HIVE-1 Configuration Search . type ranger-hive.
2. In Hive Service Advanced Configuration Snippet (Safety Valve) for ranger-hive-security.xml, click +.
   a) Under HIVE-1, in Name, type: ranger.plugin.hive.urlauth.filesystem.schemes.
   b) In Value, type: file:
   c) Click Save Changes.
3. In Cloudera Manager Hive_On_Tez-1 Configuration Search . type ranger-hive.
4. In Hive Service Advanced Configuration Snippet (Safety Valve) for ranger-hive-security.xml, click +.
   a) Under HIVE_ON_TEZ-1, in Name, type: ranger.plugin.hive.urlauth.filesystem.schemes.
   b) In Value, type: file:
   c) Click Save Changes.
5. In HIVE-1 Actions , click Restart.
6. In HIVE_ON_TEZ-1 Actions , click Restart.

   By default the value of the parameter is “hdfs:,file:”. If you remove “hdfs:”, access requests will be authorized against the HIVE URL policy and won't check for hdfs plugin or Hadoop ACL.
7. In Ranger Resource Policies Hadoop SQL , click Add New Policy.
8. In Policy Details, select URL, then type the URL represents the location or folder to which you want Ranger to authorize access: hdfs://<hostname>.root.hwx.site:8020/demo/data.

9. In Allow Conditions, select user(s), then choose Read and Write permissions, as shown in the following example:

Figure 7: Creating a Hive Authorizer URL Policy

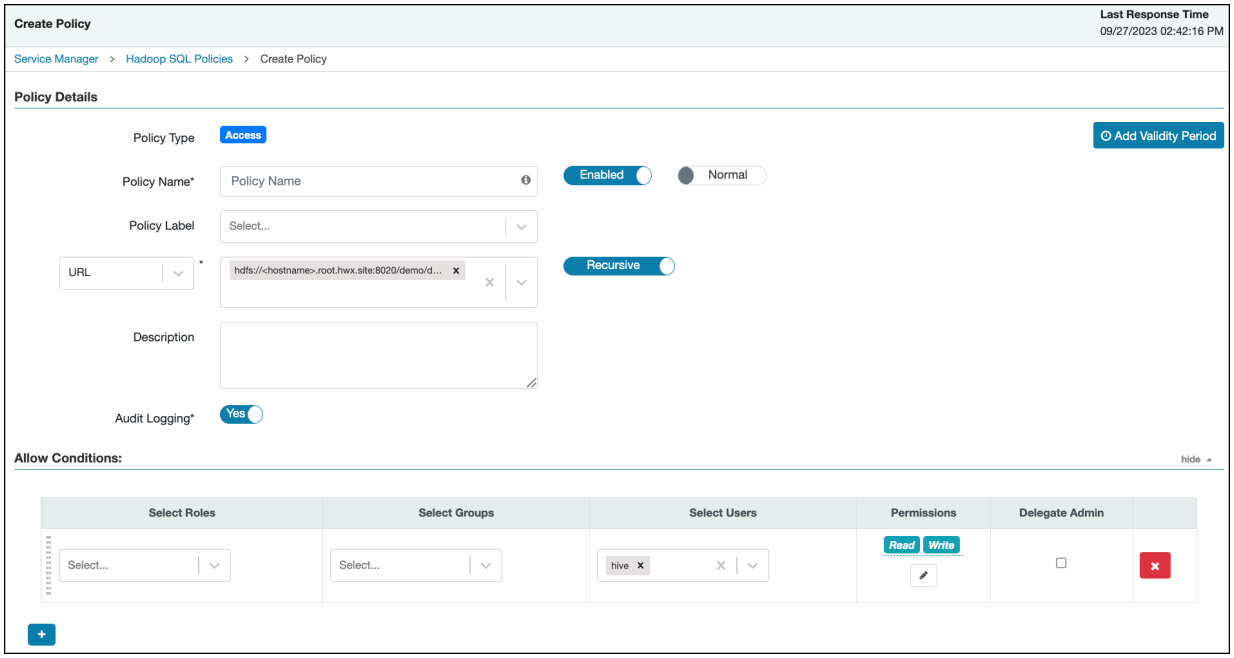

This policy allows the user to READ / WRITE into the location defined by the URL.

CREATE EXTERNAL TABLE IF NOT EXISTS STUDENT (student_ID INT, FirstName STRING, LastName STRING, year STRING, Major STRING) COMMENT 'Student Names' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/demo/data';

This will create a table reading data from the location /demo/data provided user will have the necessary READ permission to the location along with CREATE permission for table STUDENT

If the storage system is S3A or ADFS, then URL policy would be maintained for the scheme. For example, s3a:// <folder>, abfs://<folder>.

Hive supports URL policies for the following commmands that have URLs defined for the respective location:

CREATE TABLE

exernal table location

ALTER TABLE LOCATION

new location

ALTER PARTITION LOCATION

new partition location

ALTER TABLE ADD PARTITION

for partition location

LOAD

input location

For additional information about creating Hive commands with URL, see https://cwiki.apache.org/confluence/display/RANGER/Hive+Commands+to+Ranger+Permission+Mapping .

Showing Role|Grant definitions from Ranger HiveAuthorizer

You can use beeline to show the roles granted to users, groups, and roles.

About this task

You can create roles in Ranger or in Hive. You create roles in HIVE using ROLE commands. such as CREATE ROLE, GRANT / REVOKE ROLE. You can create roles in Ranger, using the Ranger Admin Web UI, if you have Admin permissions. See related links for more information about creating roles. The Hive2 command line infterface Beeline returns role grant defnitions for a specific principal, such as a user, group or role.

Before you begin

Roles must be defined before using beeline to show role|grant definitions.

Procedure

1. Run beeline, (the hive2 command line interface) on the Ranger host.

```
beeline -u jdbc:hive2://<ranger_host_name>
```

2. Enter valid syntax to return the role definitions for a specific principal.

   Syntax

   SHOW ROLE GRANT (USER|GROUP|ROLE) principal_name;

   where

   principal_name is USER | GROUP | ROLE name

Results

Beeline outputs query results, as shown in following examples:

Example

SHOW ROLE GRANT USER HDFS -> show roles for user "hdfs"

```
0: jdbc:hive2://rm-ranger-3.rm-ranger.root.hw> show role grant user hdfs;
INFO  : Compiling command(queryId=hive_20211109235258_b9211cfe-0e78-47d7-8a2a-1b6111ddcd18): show role grant user hdfs
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:role, type:string, comment:from deserializer), FieldSchema(name:grant_option,
_time, type:bigint, comment:from deserializer), FieldSchema(name:grantor, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20211109235258_b9211cfe-0e78-47d7-8a2a-1b6111ddcd18); Time taken: 0.02 seconds
INFO  : Executing command(queryId=hive_20211109235258_b9211cfe-0e78-47d7-8a2a-1b6111ddcd18): show role grant user hdfs
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20211109235258_b9211cfe-0e78-47d7-8a2a-1b6111ddcd18); Time taken: 0.008 seconds
INFO  : OK
+------------+---------------+---------------+----------+
|    role    | grant_option  |   grant_time  | grantor  |
+------------+---------------+---------------+----------+
| ITManager  | false         | 1636501912000 |          |
+------------+---------------+---------------+----------+
```

Example

SHOW ROLE GRANT ROLE -> show roles for role "ITManagers"

```
0: jdbc:hive2://rm-ranger-3.rm-ranger.root.hw> show role grant role ITManager;
INFO  : Compiling command(queryId=hive_20211109235607_2d543c50-7c7b-4d54-bed0-159f67c24079): show role grant role ITManager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:role, type:string, comment:from deserializer), FieldSchema(name:grant_option,
_time, type:bigint, comment:from deserializer), FieldSchema(name:grantor, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20211109235607_2d543c50-7c7b-4d54-bed0-159f67c24079); Time taken: 0.177 seconds
INFO  : Executing command(queryId=hive_20211109235607_2d543c50-7c7b-4d54-bed0-159f67c24079): show role grant role ITManager
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20211109235607_2d543c50-7c7b-4d54-bed0-159f67c24079); Time taken: 0.007 seconds
INFO  : OK
+------------+---------------+----------------+----------+
|    role    | grant_option  |   grant_time   | grantor  |
+------------+---------------+----------------+----------+
| Managers   | false         | 1636502074000  |          |
| TeamLeads  | false         | 1636502122000  |          |
+------------+---------------+----------------+----------+
```

Related Information

Apache documentation on Role operations

Adding a role through Hive

Adding a role through Ranger

Ranger Security Zones

Ranger security zones let you organize service resources into multiple security zones.

Security Zones allow carving/bucketing of resources in a service into multiple zones for better administration of security policies. Defining Security Zones can enable multiple administrators to setup security policies for a service – based on the zones to which they have been granted administration rights.

Security Zones Administration

A Security Zone enables a Ranger administrator to separate resource policies into different administrative zones.

What is a Security Zone?

Security Zones help simplify security policy administration, and allow a limited amount of policies to be checked when doing authorization against certain resources. Only policies under a particular zone that contains the requested resource are loaded and checked by Ranger.

For example, let us consider two security zones: finance and sales:

- Security zone finance includes all content in a Hive database named finance.
- Security zone sales includes all content in a sales database.
- Policies defined in a security zone apply only to resources of that zone.
- A zone can be extended to include resources from multiple services such as HDFS, Hive, HBase, Kafka, etc. Extending a zone across multiple services allows zone administrators to set up policies for resources owned by their organization across multiple services.

For example:

```
Zone: finance
    service: prod_hdfs; path=/finance/*, /taxes/*
    service: prod_hive; database=finance
    service: prod_kafka; topic=FIN_*
    service: test_hadoop; path=/finance/*, /taxes/*
Zone: sales
    service: prod_hadoop; path=/sales/*
    service: prod_hive; database=sales
    service: prod_kafka; topic=SALES_*
```

- As shown above, resources can be specified using wildcards (FIN_*, SALES_*).
- Sets of users and groups are designated as administrators in each security zone.
- Users are allowed to set up policies only in security zones in which they are administrators.
- A resource cannot map to more than one security zone. Ranger does not allow creating security zones that specify resources that match resources in another zone. For example, an attempt to update the finance zone in the above example with the HDFS path /sales/finance/* is not permitted, because this conflicts with the HDFS path /sales/* specified in the sales zone.
- A set of users and groups can be designated as administrators of a security zone. Administrators can create, update, and delete security policies for the resources in that security zone.
- A set of users and groups can be authorized to view audit logs for that security zone's resources. Other users are not allowed to view access-audit logs for that security zone's resources.
- The security zone name appears in the zonename column of the access-audit log.

Security Zone Administration

- Security zones can only be created, updated, or deleted by a user with the ROLE_SYS_ADMIN role in Ranger.
- Users can view, retrieve, and update policies only in security zones in which they have administrator privileges.
- Users can view/retrieve and cannot update zone policies for which they have zone auditor permission.

How are Security Zones Used in Authorization?

When a Ranger authorization plugin authorizes a resource access request, it first determines the zone in which the accessed resource resides. If the resource matches a security zone, only the policies of that security zone are used to authorize the access. If resource does not match any security zone, the policies in the default (unnamed) security zone are used to authorize the access.

Tag-based Policies in Security Zones

In a given service, each security zone can be configured to use tag-based policies from a specific security zone in a tag-service. This enables different tag-based authorization policies to be used, based on the security zone of the resource.

Audit Logs

Audit logs generated by Ranger include the name of the security zone in which the accessed resource resides. Only users who have been assigned as an Admin or Auditor for the security zone are allowed to view the audit logs.

Security Zones Example Use Cases

Four example use cases for admininistering security zones.

Based on the following example:

```
Zone: finance
          service: prod_hdfs; path=/finance/*, /taxes/*
          service: prod_hive; database=finance
          service: prod_kafka; topic=FIN_*
          service: test_hadoop; path=/finance/*, /taxes/*
Zone: sales
          service: prod_hadoop; path=/sales/*
          service: prod_hive; database=sales
          service: prod_kafka; topic=SALES_*
```

Use case 1 : Access HDFS path using zone policy

For example, let us access hdfs path using unixuser1 user from finance zone.

Finance zone resource:

Ranger Service : prod_hdfs

Resource : /finance/*

Finance zone policy:

Resource Path : /finance/*

User : unixuser1

Permission : read, write, execute

Now, when unixuser1 user tries to create dir in /finance dir, Ranger checks for zone with resource /finance and policy for that user in that zone and then allows access for that user. Also, access-audit logs for that operation appear in the Ranger Admin Web UI, Access Audit tab.

Use case 2 : Hive access policy and tag masking policy

For example, we want to manage access policies and masking policy for taxation-related information in multiple finance databases for an organization.

Zone Resource :

Zone Tag service: cm_tag

Ranger Service : prod_hive

Resource :

Database : finance

Zone policy resource

Tag policy

resource:TDS

Hive policy

Resource :

Database : finance

Now, the Admin and security zone admin can create access policies and masking policies for all the resources associated with tag TDS and as and when new tables on Hive/HBase are created for saving any taxation related data. They can associate a TDS tag with a related Hive/HBase column. This will enable zone admin to create policies for masking the confidential data of its organization.

Use case 3 : Knox topologies

For example, suppose we want to manage access to a service. We can mange access to a service using topology.

Zone Resource :

Ranger Service : prod_knox

Resource:

Knox Topology:cdp-proxy-api

Knox Service:WEBHDFS

Zone deny policy Resource:

Knox Topology:cdp-proxy-api

Knox Service:WEBHDFS

Without a security zone, access to webhdfs is allowed since the default policy has a 'public' group in it.

Use case 4 : Import and export of zone policy

We can import and export zone policies from stage to prod.

Suppose we want to have the same policy in production that exists on stage. We can export the zone policy from the stage where the exported json has a zone name as a parameter in the json. While importing, we can map the zone name of stage to prod and then import the policies.

Adding a Ranger security zone

Ranger administrator users can create a Security Zone using the Ranger Admin Web UI.

Procedure

1. In Ranger Admin Web UI Service Manager , click Security Zone.

   Security Zone displays existing security zones. If no zone exists, two options for creating a new zone display.

2. On Security Zone, click + Create (new) Zone.

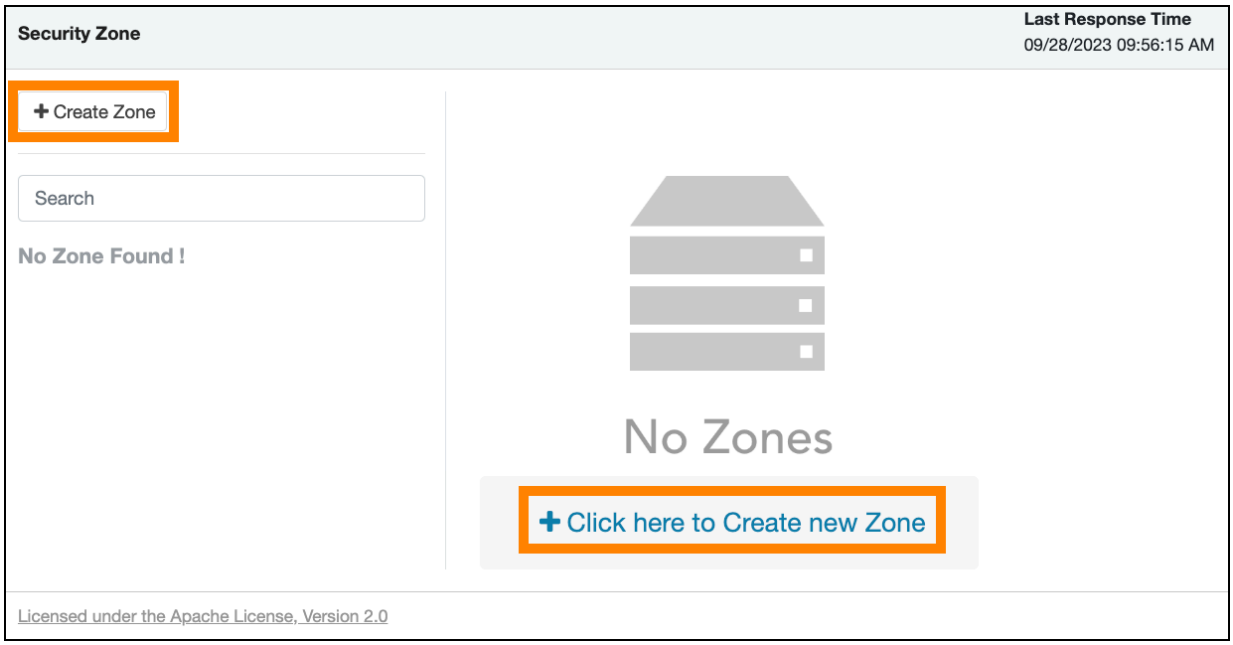

Create Zonedisplays options for creating a new security zone.

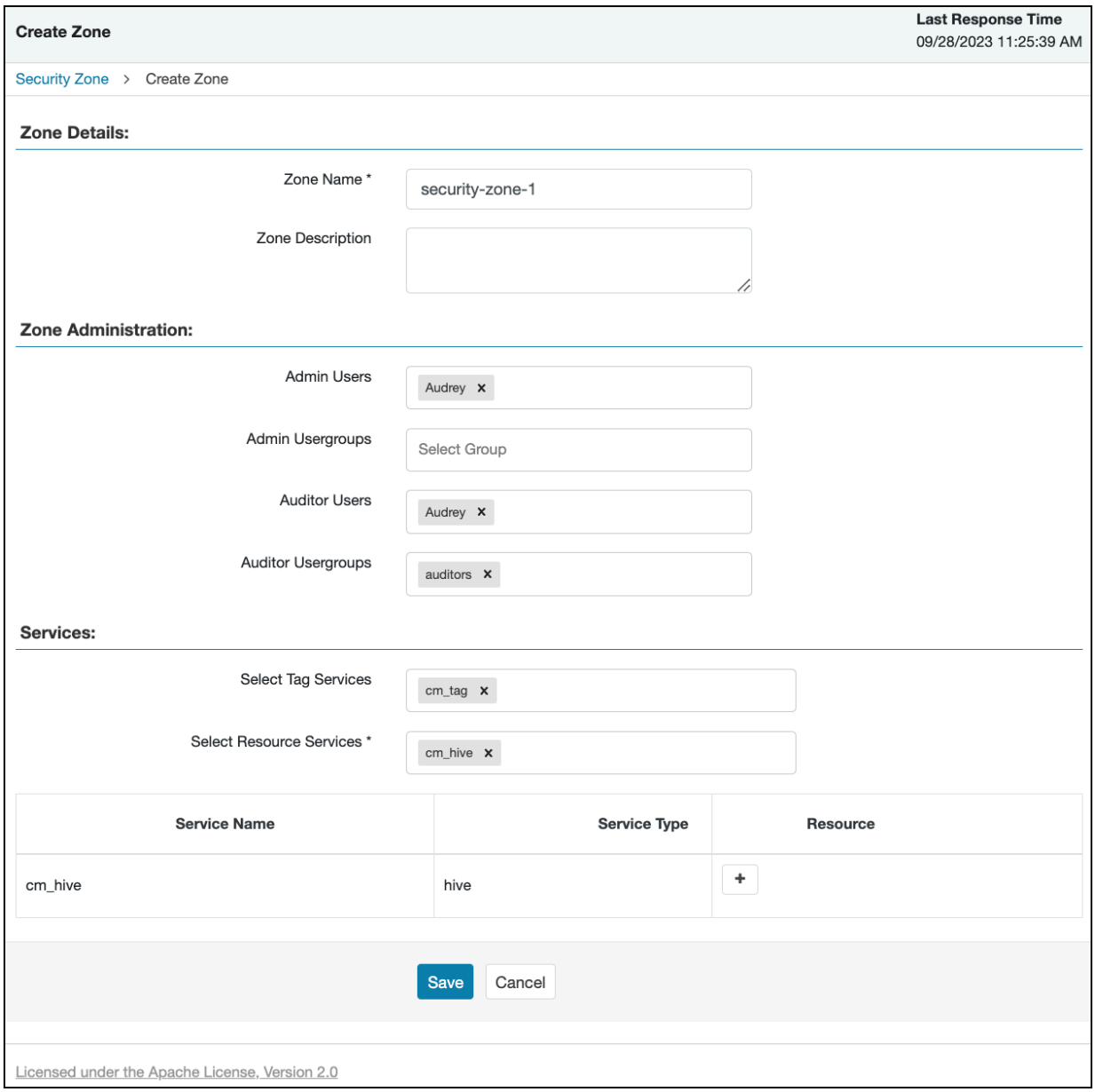

3. On Create Zone, enter the following information:

Table 62: Zone Details

Field	Description
Zone Name	The security zone name.
Zone Description	An optional description.

Table 63: Zone Administration

Field	Description
Admin Users	The Admin users for the security zone.
Admin Usergroups	The Admin user groups for the security zone.
Auditor Users	The Auditor users for the security zone.

Field	Description
Auditor Usergroups	The Auditor user groups for the security zone.

Table 64: Services

Label	Description
Select Tag Services	Select tag-based services for the security zone.
Select Resource Services	Select resource-based services for the security zone.

4. Selected services are listed in Services. To add resources for each selected service, click + in the Resource column for the applicable service.

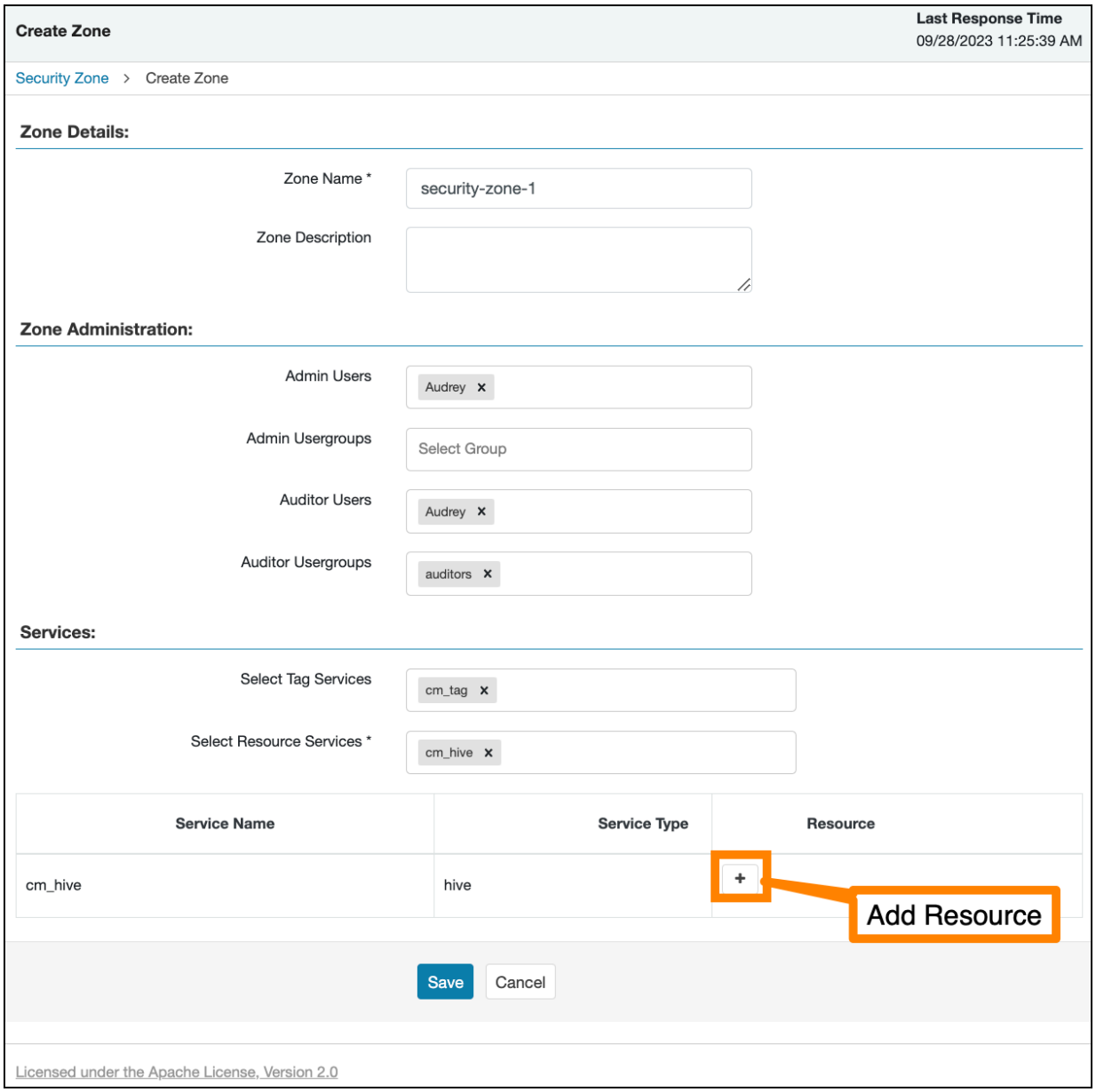

5. Use Resource Details to specify resources for the service, then click Save.

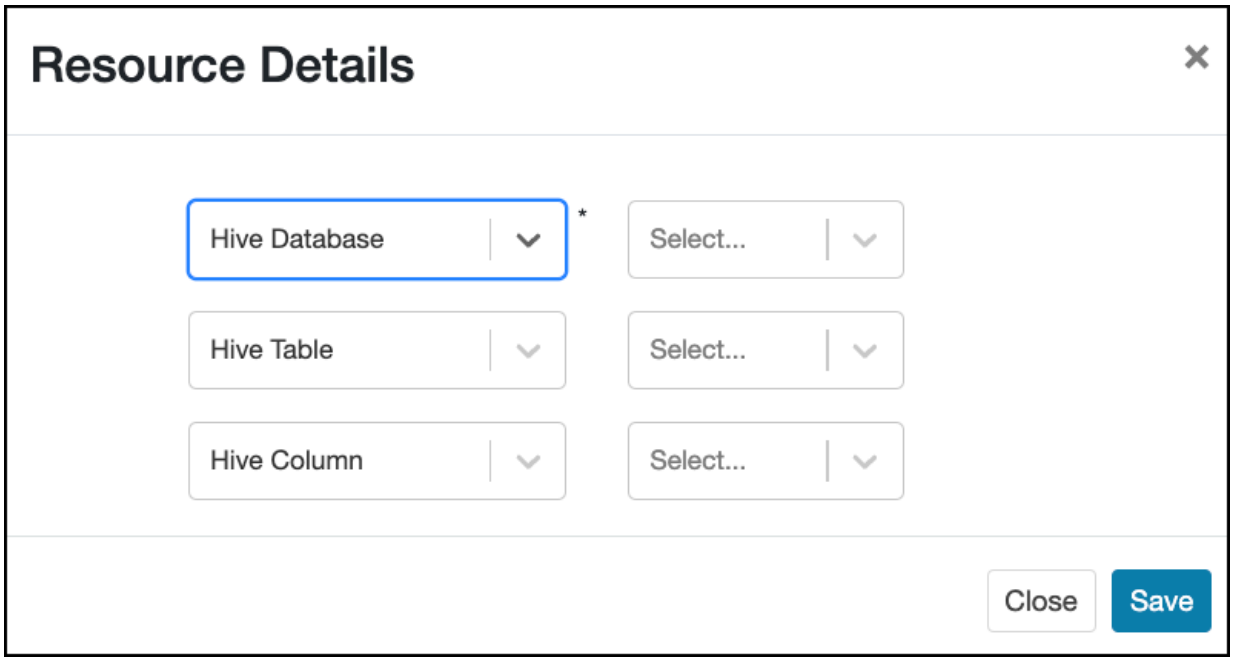

Selected resources appear as Resources for each Service in Create Zone.

Note: The solr plugin supports fine-grained authorization similar to legacy Sentry privileges. A part of this support introduces the following new solr resources: collection, config, schema and admin. To perform any operation on a collection, a user also requires admin-level permission. To create a security zone for with the solr service that includes a collection resource, you must also add an admin resource. Currently, if you use one solr service to create a security zone that has a collection resource (and therefore includes an admin resource) you cannot create another solr security zone using another collection. (currently, only one admin resource can be used per solr security zone). This limitation exists for security zones in Cloudera Runtime-7.1.8.

6. Click Save at the bottom of Create Zone to save the new security zone.

7. The new security zone is listed on the Security Zone page.

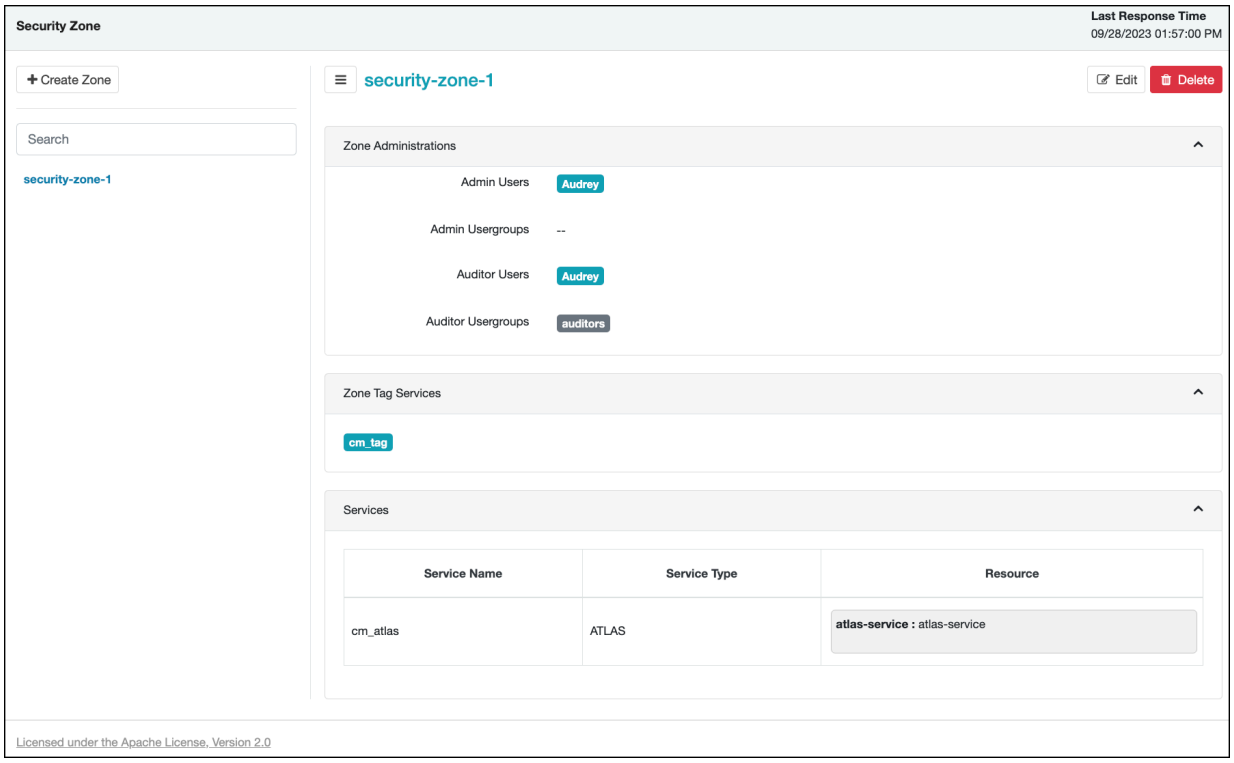

8. To edit a security zone, click the security zone name in the Security Zones list, then click Edit.
9. After security zones have been created, you can use the Security Zone selection box on the Service Manager page to display the services assigned to the selected security zone. A Zone Name column appears in the table on Audit Access , and also in Service Manager Reports .

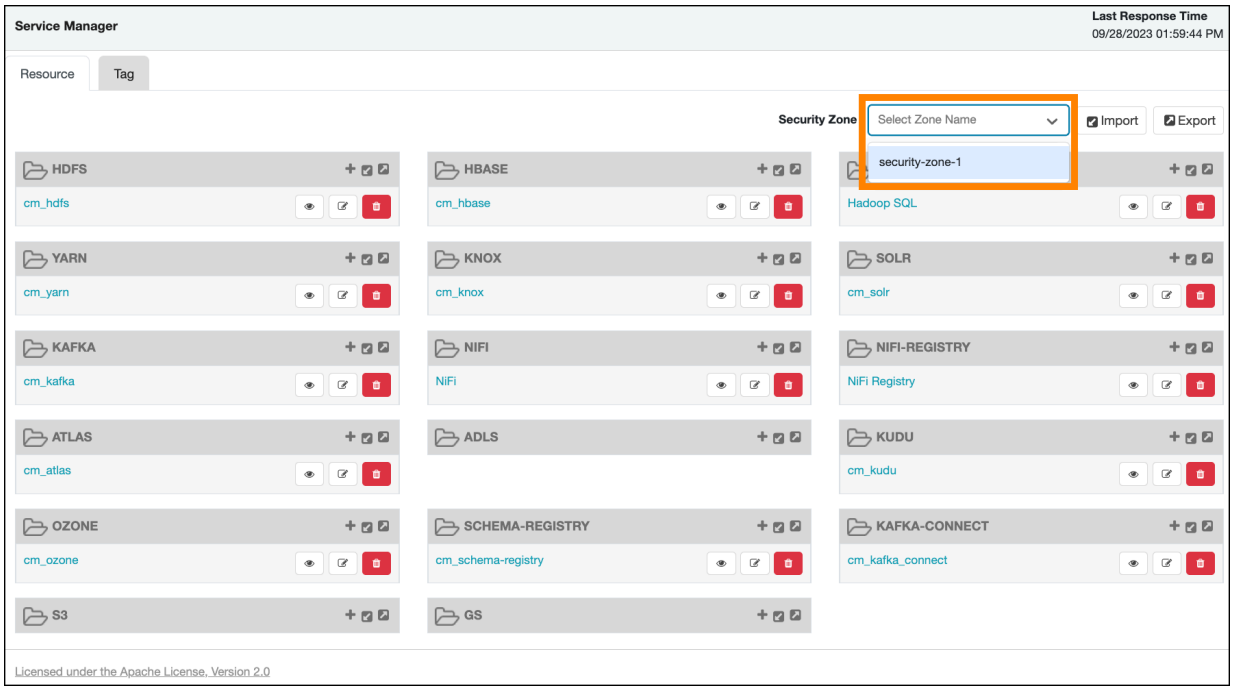

Administering Ranger Reports

You can use Service Manager Reports to help manage policies more efficiently as the number of policies increases. Reports lists all resource-based and tag-based policies.

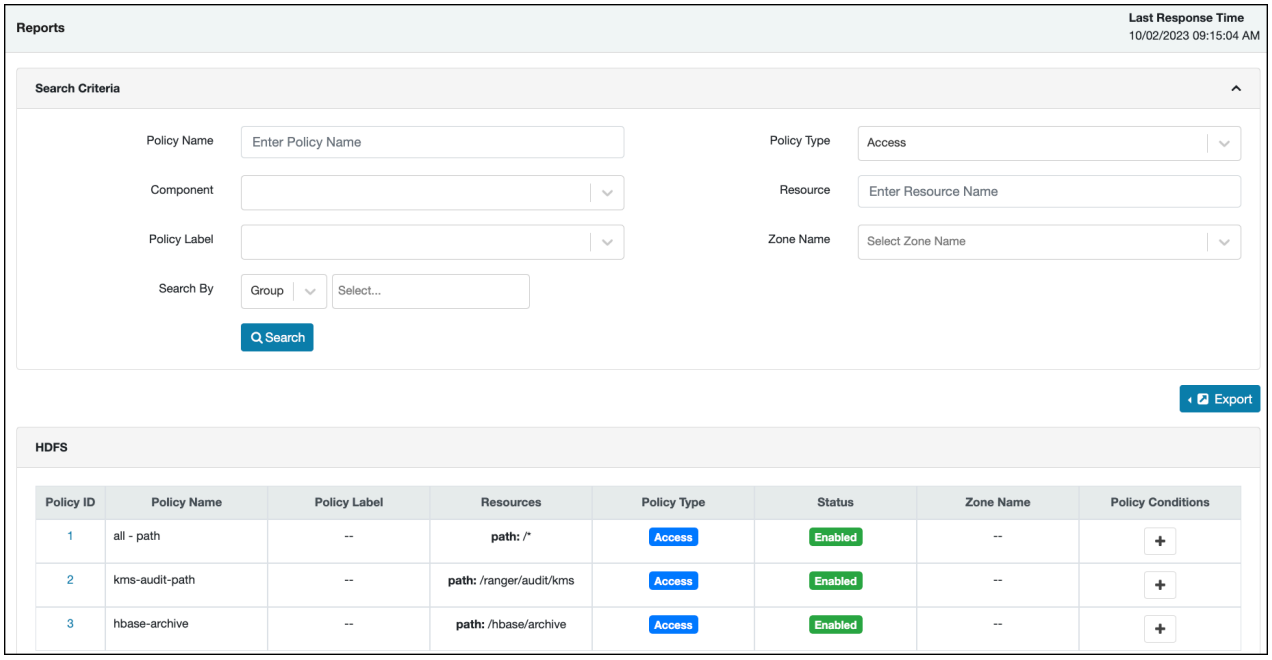

View Ranger reports

How to view reports for Ranger policies.

To view reports for one or more policies, select Service Manager Reports .

- To view Allow Condition details for each policy, click + in the Allow Conditions column. You can use the same method to view details for other policy conditions (Allow Exclude, Deny Conditions, etc.).
- To edit a policy from the Reports page, click the Policy ID.

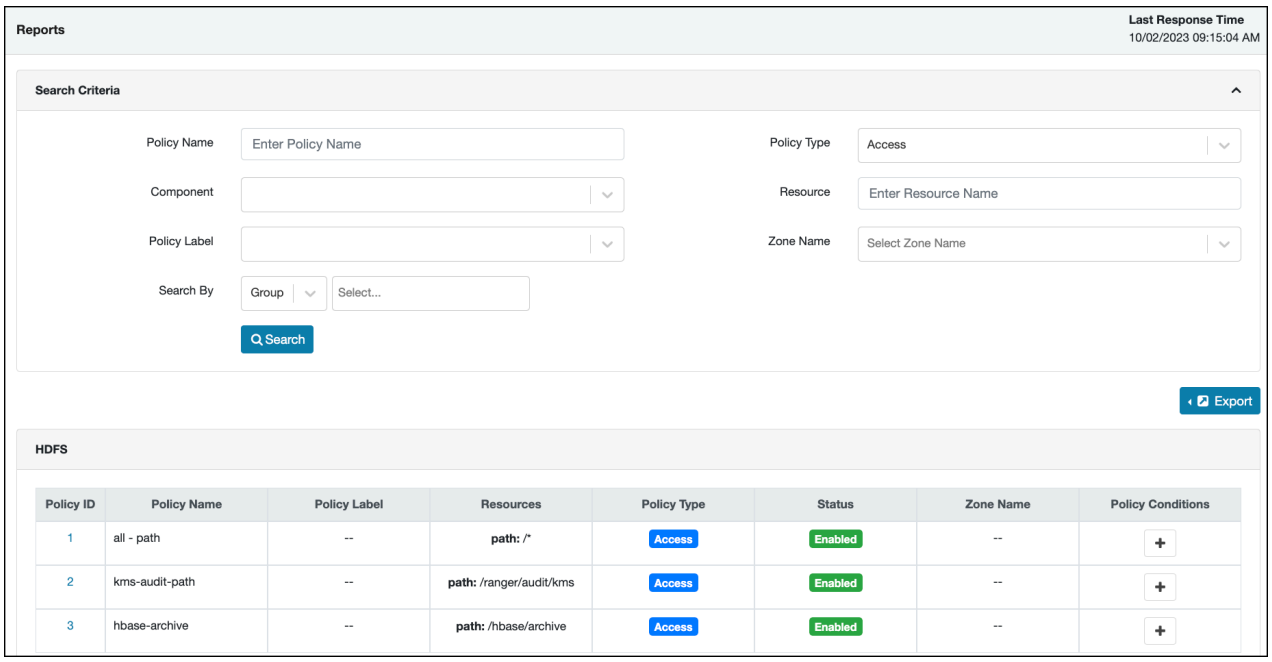

Search Ranger reports

Reference information for searching Ranger reports on one or more policies.

You can search based on:

- Policy Name – The policy name.
- Policy Type – The policy type (Access, Masking, or Row Level Filter).
- Policy Label – The policy label.
- Component – The policy resource or tag component.
- Resource – The resource path used when creating the policy.
- Zone Name – The security zone name.
- Group, Username – The group or user name assigned to the policy.

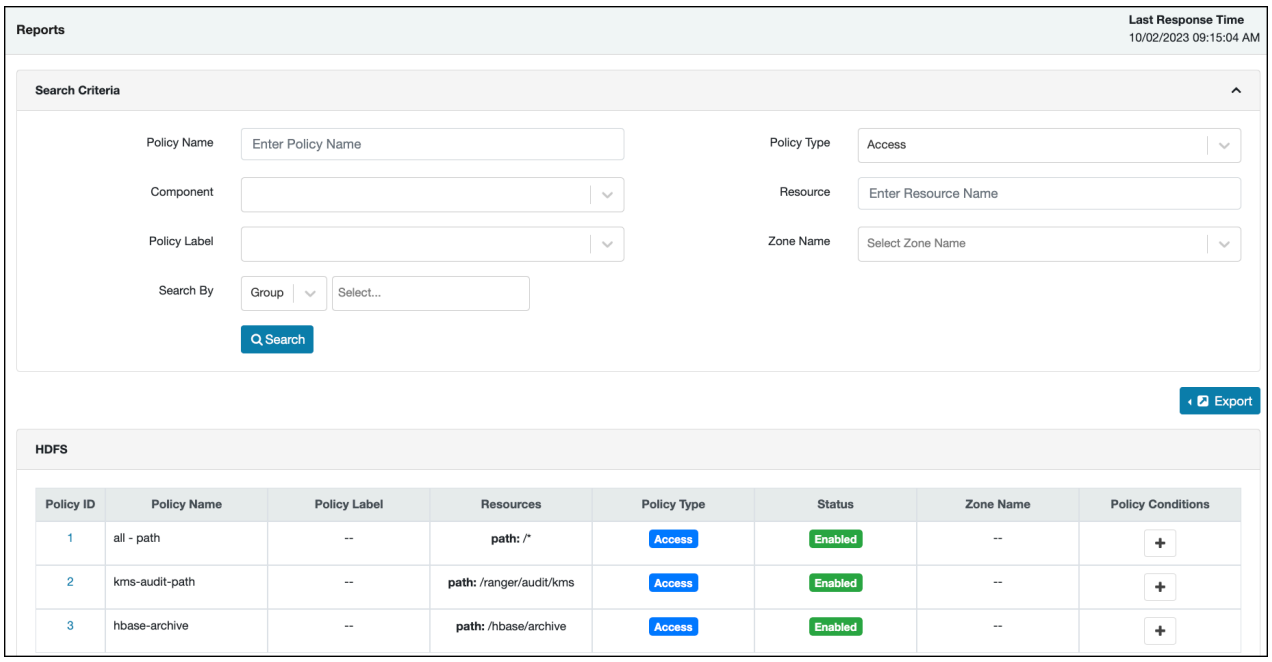

Export Ranger reports

Reference information for exporting Ranger reports on one or more policies.

You can export a list of reports in three file formats:

- CSV file
- Excel file
- JSON

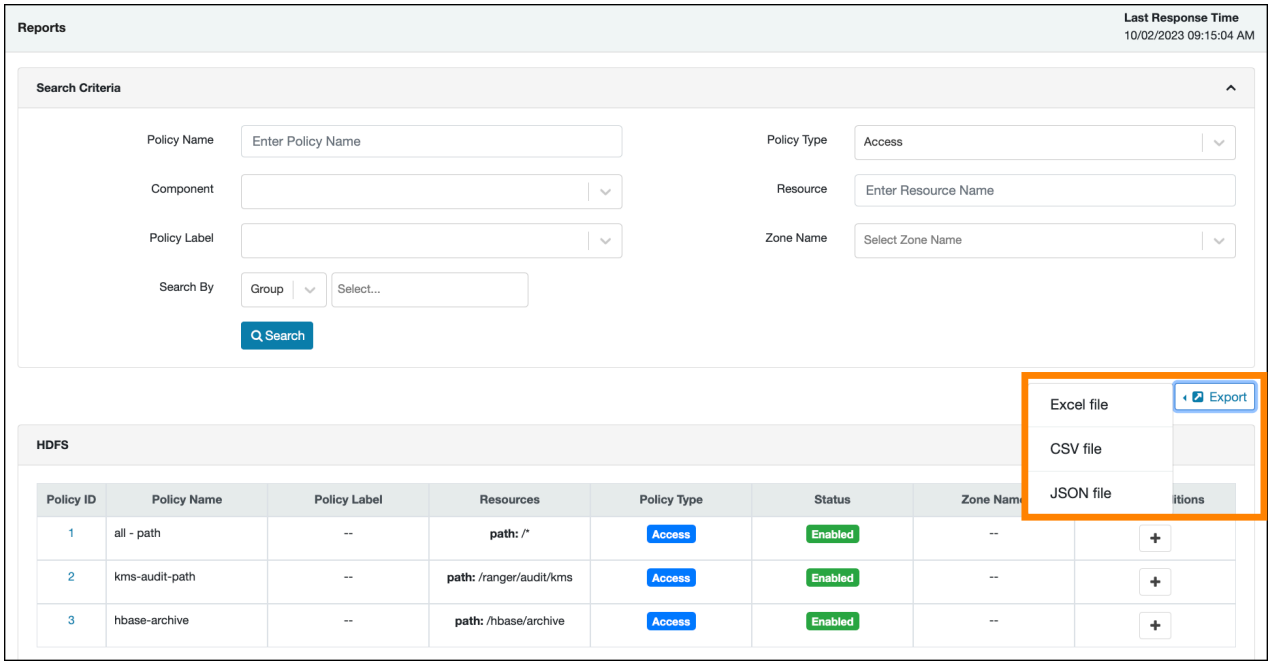

Related Information

Export tag-based policies

Export resource-based policies for a specific service

Export all resource-based policies for all services

Using Ranger client libraries

Ranger now supports clients written in java and python which enable applications to access Ranger REST APIs programmatically. Using client library code simplifies access using java or python, compared with making direct HTTP requests to Ranger REST APIs.

Summary

Ranger client libraries:

- Provide idiomatic, hand-written code in Java and Python, making Ranger REST APIs simple and intuitive to use.
- Handle all low-level details of communication with the server including complexities involved in JSON parsing.
- Support installing the python client using the familiar package management tool pip.

Table 65: Ranger Client Installation Repo and Library Reference Links

Language	Installation	Library Reference
java	github source repository	java library reference
python	github source repository	python library reference

Authentication

The Apache Ranger release 2.2 client supports two authentication types:

- Basic authentication (username/password)
- Kerberos authentication

Java client prompts for the authentication mode to be used at runtime. For Kerberos-based authentications, a principal and keytab file path is required.

SSL

Java and Python clients support SSL/TLS-enabled ranger. To connect to HTTPS ranger using java client, provide the path to the SSL configuration file, as shown in this example:

```
$ ./run-sample-client.sh -n <ranger_admin_url>
SSL Configuration File: /path/to/config.xml
```

Sample SSL configuration file which requires values to be populated:

```
<configuration>
  <property>
    <name>xasecure.policymgr.clientssl.truststore</name>
    <value></value>
  </property>
  <property>
    <name>xasecure.policymgr.clientssl.truststore.credential.file</name>
    <value></value>
  </property>
  <property>
    <name>xasecure.policymgr.clientssl.truststore.type</name>
    <value></value>
  </property>
</configuration>
```

Environment variables

The Java client requires requires that you initialize the following environment variables:

```
$ export JAVA_HOME=/usr/java/<jdk_version>/bin
$ export PATH=$PATH:$JAVA_HOME
$ export HADOOP_CREDSTORE_PASSWORD=<hadoop_credstore_password>
```

Using session cookies to validate Ranger policies

Apache Ranger REST Client uses cookie sessions to download policies, tags and roles from Ranger Admin.

In earlier versions, each Ranger plugin used a kerberos login to request a ticket granting ticket (TGT) from the KDC/AD server in order to download policies, tags and roles. This casued high traffiic levels when mulitple Ranger plugins requested downloads.

Ranger Admin now supports cookie-based sessions. The flag used to enable cookie sessions, ranger.plugin.<service-name>.policy.rest.client.cookie.enabled, where <service-name> is the name of the service for which a Ranger plugin is eanbled, such as hive, solr, or kafka, is set to "enabled" by default.

To check whether the cookie session is used, open the Ranger Admin access.log in the /var/log/ranger/admin folder. Any policy, tag, or role download call to Ranger Admin displays either a 200 or 304 value as response status. A 401 value for response status indicates the call to the KDC server for a TGT for autherntication at service start or when the session cookie expires.

Configure optimized rename and recursive delete operations in Ranger Ozone plugin

You can enable performance optimized authorization approach for rename and recursive delete operations in the Ranger Ozone plugin.

About this task

Ozone introduced support for FSO (FILE_SYSTEM_OPTIMIZED) Bucket layout. FSO Bucket layout is a Hierarchical FileSystem namespace view with directories and files. Similar to HDFS, with FSO bucket layout, Ozone has an efficient directory rename and delete operations. Ranger supports not only authorization for rename and recursive delete operations, but also provides an option to enable performance optimized solution when these operations are performed on directory containing large set of subpaths (directories/files) within it.

Property name - ranger.plugin.ozone.optimized.subaccesspath.enabled

Default is set to false.

To enable authorization for rename and recursive delete operations in the Ranger Ozone plugin:

Procedure

1. In Cloudera Manager Ozone Ozone Manager Configuration Search . type ranger-ozone-security.xml.
2. In Ozone Service Advanced Configuration Snippet (Safety Valve) for ranger-ozone-security.xml, click +.
   a) Under Ozone, in Name, type: ranger.plugin.ozone.optimized.subaccesspath.enabled.
   b) In Value, type: true.
   c) Click Save Changes.
3. In Ozone Actions , click Restart.

Results

Ranger not only authorizes rename and recursive delete operations, but also provides an option to enable performance optimized solution when these operations are performed on a directory containing a large set of subpaths (directories/files) within it.

How to optimally configure Ranger RAZ client performance

How to find and set configurations for RAZ client performance.

About this task

This topic presents a set of best- balanced configs for Ranger RAZ clients, based on our past experience and testing. The majority of the time, the default config set for Ranger RAZ client is sufficient. We do not recommend any update in these default Ranger RAZ client configs, as it might result in unwanted outcomes.

In some cases, you may want to update these configs to optimize/suit your environment and if you are willing to take risks.

The following table lists some useful configs and short descriptions, which will help you optimize.

Procedure

1. Go to Cloudera Manager HDFS Configuration .
2. In Search, type core-site.xml.
3. Add the following configurations in Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml:

Table 66: RAZ client configuration properites and default values

Configuration Name	Description	Default Value
ranger.raz.client.max.retry	Ranger Raz client retries (high layer), must not be negative	3
ranger.raz.client.rest.client.connection.pool.retry-count	Lower layer retries, must not be negative	3
ranger.raz.client.rest.client.connection.timeoutMs	Connection timeout in milliseconds	120000
ranger.raz.client.rest.client.read.timeoutMs	Read timeout in milliseconds	30000

Note: Changes in these default values might result in reduction of stability of a job (for example, yarn/spark) completion.

Cloudera Runtime 7.3.1

Configuring and Using Ranger RMS (Hive-S3 ACL Sync)

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to Ranger RMS.....4
Ranger RMS - HIVE-S3 ACL Sync Overview.....4

Understanding Ranger policies with RMS.....7

How to full sync the Ranger RMS database.....8

Configuring Ranger RMS (Hive-S3 ACL Sync).....9

Ranger RMS (Hive-S3 ACL-Sync) Use Cases.....11

Introduction to Ranger RMS

Ranger Resource Mapping Server (RMS) enables automatic translation of access policies from HIVE to S3. This feature is available only in AWS deployments.

Ranger RMS - HIVE-S3 ACL Sync Overview

Ranger Resource Mapping Server (RMS) enables automatic translation of access policies from HIVE to S3. This feature is available only in AWS deployments.

About HIVE-S3 ACL Sync

It is common to have different workloads use the same data – some require authorizations at the table level (Apache Hive queries) and others at the underlying files (Apache Spark jobs). Unfortunately, in such instances you would have to create and maintain separate Ranger policies for both Hive and S3, that correspond to each other.

As a result, whenever a change is made on a Hive table policy, the data admin should make a consistent change in the corresponding S3 policy. Failure to do so could result in security and/or data exposure issues. Ideally the data admin would set a single table policy, and the corresponding file access policies would automatically be kept in sync along with access audits, referring to the table policy that enforced it.

Legacy CDH users had a feature called the Hive-HDFS ACL sync which had Hive policies in Apache Sentry that automatically linked Hive permissions with HDFS ACLs. This was especially convenient for external table data used by Spark or Hive.

Prior to Cloudera Public Cloud 7.2.18, Ranger only supported manually managing Hive and S3 policies separately. Ranger RMS (Resource Mapping Server) allows you to authorize access to S3 locations using policies defined for Hive tables. RMS is the service that enables Hive-S3 Policy Sync.

RMS periodically connects to the Hive Metastore and pulls Hive metadata (database-name, table-name) to S3 file-name mapping. We have introduced a RAZ-chained plugin (running in the Ranger RAZ service) which has an additional HivePolicyEnforcer module. The RAZ-S3 plugin downloads Hive policies from Ranger Admin, along with the mappings from Ranger RMS. S3 access is determined by both S3 policies and Hive policies.

Figure 1: HIVE-S3 ACL Sync using Ranger RMS

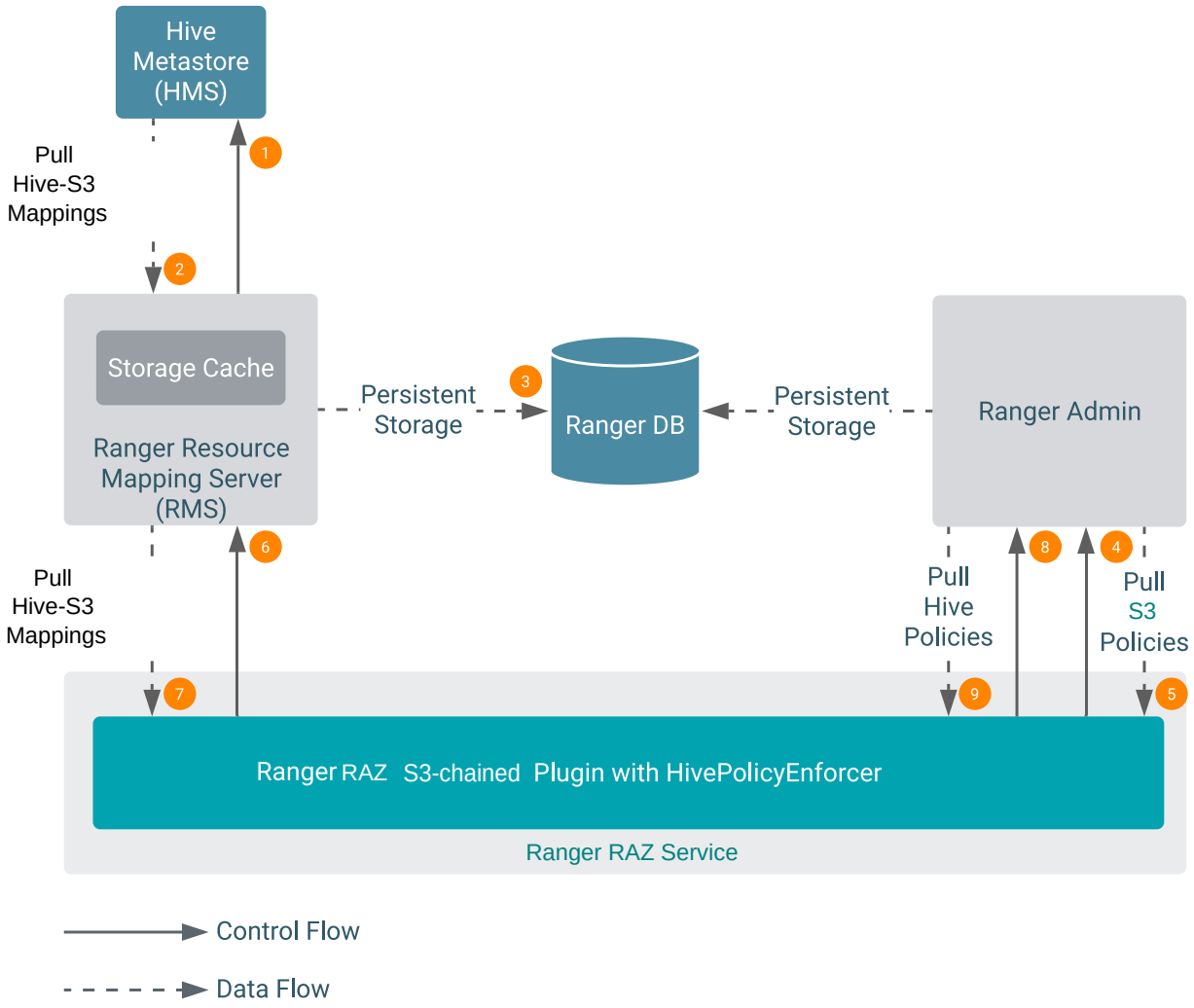

Phase I (items 1-3 above)

Ranger RMS periodically connects to the HIVE Metastore and pulls HIVE metadata (database-name, table-name) to S3 file-name mapping.

Phase II (items 4-9 above)

The Ranger RAZ S3 Chained Plugin (running in the RAZ service) periodically pulls S3 policies from Ranger Admin. With the introduction of Ranger RMS, the Ranger RAZ S3 Chained Plugin (running in the RAZ service) has been extended with an additional HIVEPolicyEnforcer module. It now pulls down the HIVE-S3 mappings from RMS and HIVE Policies from Ranger Admin.

After phase II completes, the requested S3 access is determined in the RAZ service by the S3 and HIVE policies defined by the Ranger Administrator.

About database-level grants feature

Legacy CDH users used HIVE policies in Apache Sentry that automatically linked HIVE permissions with HDFS ACLs. This was especially convenient for external table data used by Spark or HIVE. Specifically, using Sentry, you could make grants at the HIVE database level and HDFS permissions would propagate to the database directory, and to all tables and partitions under it.

In RMS-S3, we have introduced the database-level grant feature also. Ranger Resource Mapping Server (RMS) willl allow you to create a database-level policy in HIVE and have these permissions propagate to the S3 locations and all tables under it. RMS is the service that enables HIVE-S3 ACL Sync.

RMS captures database metadata from the HIVE Meta Store (HMS). After the first, full-synchronization run, RMS downloads mappings for tables and databases present in the HMS.

Whenever you create a new database, RMS synchronizes metadata information from HMS and uses it to update the resource mapping file linking HIVE database resources to their corresponding S3 location. Any user with access permissions on a HIVE database automatically receives similar S3 file-level access permissions on the database's data files. Select/ Read access for any user in the database location is allowed through default HIVE policy for all-databases. This behavior is treated as _any access, which is similar to the HIVE command `show tables`. If a user has no HIVE policy which allows access to the database, then the user is denied access to the corresponding S3 location of that database. Without this feature, users will not be allowed to access the S3 location of a database even if the user had permission to access the database through a HIVE policy. The S3 to HIVE access type mappings follow:

Access Type mapping for S3 to HIVE for Database:

- _any=[_any]
- read=[_any]
- write=[create, drop, alter]

Access Type mapping for S3 to HIVE for Table:

- _any=[_any]
- read=[select]
- write=[update, alter]

If you create tables under a database but the S3 location of the corresponding table does not reside under the S3 location of that database (for example: table locations are external locations), the HIVE policies (database- name, table = *, column= *) translate into S3 access rules and allow the RAZ S3 chained plugin to enforce them. If the policy is created only for the database resource, the same access translates to the S3 location of that database only; not for the tables residing under that database.

Ranger RMS Assumptions and Limitations

- All partitions of a table are assumed to be under the location specified for the table. Therefore, table permissions will not authorize access to partitions that store data outside the location specified for the table. For example, if a table is located in a /warehouse/foo S3 directory, all partitions of the table must have locations that are under the /warehouse/foo directory.
- In Cloudera Public Cloud 7.2.18 or above, RMS service will be available only on AWS deployment for fresh install setup (not for upgrade scenario). A customer with this new RMS entitlement ENABLE_RMS_ON_DATALAKE should be able to create a cluster with RMS as a configurable option (`--enable-ranger-rms`) through a cdp cli command `create-aws-datalake`. When RMS is selected during cluster setup, customers will not be required to install & configure RMS separately.
- The Ranger RMS ACL-sync feature supports a single logical HMS, to evaluate S3 access via HIVE permissions. This is aligned with the Sentry implementation in CDH.
- Permissions granted on views (traditional and materialized) do not extend to S3 access. This is aligned with the Sentry implementation in CDH.
- RMS ACL sync is designed to work on a specific pair of S3<->Hive Ranger service. Ranger RMS supports only one pair of Hive and S3 services. By default cm_s3 is configured as source service and cm_hive as target service.
- If a Cloudera Public Cloud deployment supports multiple logical HMS with a single Ranger, Ranger RMS (Hive-S3 ACL-Sync) works with only one logical HMS. Permissions granted on databases/tables in other logical HMS instances will not be considered to authorize S3 access.
- Ranger RAZ memory requirements must be increased, based on the number of HIVE table mappings downloaded to S3 Ranger plugin. Additionally, maintaining HIVE policies in memory cache will also require additional memory.

- In Cloudera Public Cloud deployments, Ranger RMS service will be installed only on DataLake and it will use the same database as Ranger Admin to store mappings downloaded from HMS.
- Ranger RAZ service running in DataHub will have a S3 chained plugin and it will do the authorization based on the policies and mappings downloaded & stored into policy-cache directory of RAZ service. Even if the RMS service is stopped, authorization will continue to work based on the files available in the policy-cache directory.
- Metrics support for RMS with S3 is not added in the Cloudera Public Cloud 7.2.18 release.
- Expect Ranger RAZ CPU load to increase, due to additional access evaluation performed to enforce HIVE policies and periodic downloading and processing of the HIVE table mappings. The latter increase is proportional to the number of table mappings downloaded to HDFS Ranger plugin.
- When multiple databases are mapped to a single S3 location, and if a HIVE policy allows a user to access one database. Then, users will be able to access its S3 location and all other files & directories under it. This may include table or database directories of other databases and tables. However, users will not be able to access other databases or tables under it through Hive queries.

  For example,

  music_a, music_b, music_c are created at S3 path'/data'.

  Policy-A to allow 'sam' user 'all' access on resource = {database=music_a; table= * ; column= * ; }

  Now, 'sam' user will get all access on S3 path /data and files, directories under it. Therefore, 'sam' user will be able to access S3 location of tables under music_b and music_c databases as long as those locations reside under /data directory.

  However, 'sam' user will not be able to access music_b and music_c databases or any tables under these databases through Hive queries.

  Note: This is the expected behavior when the S3 location for different database/tables are designed in such a way that these entities share the same S3 path or HIVE database/table locations fall under another database S3 location.

Comparison with Sentry HDFS ACL sync

The Ranger RMS (Hive-S3 ACL-Sync) feature resembles the Sentry HDFS ACL Sync feature in the way it downloads and keeps track of the HIVE table to S3 location mapping.

It differs from Sentry in the way it completely and transparently supports all features that Ranger policies express. Therefore, support for tag-based policies, security-zones, masking and row-filtering and audit logging is included with this implementation.

Also, the feature is enabled or disabled by a simple configuration on the Ranger RAZ side, allowing each installation the option of turning this feature on or off.

Understanding Ranger policies with RMS

Ranger RMS for S3 access evaluation workflow

At a high level, the Ranger RMS for S3 access evaluation workflow is as follows:

- Ranger policies for the S3 service are evaluated. If any policy explicitly denies access, access is denied.
- Ranger checks to see if the accessed location maps to a Hive table.
- If it does, Hive policies are evaluated for the mapped Hive table. Otherwise, if there is an S3 policy allowing access, then the access is allowed.
  Requested S3 permission is mapped to Hive permissions as follows:

  S3 ‘read’ ==> Hive ‘select’

  S3 ‘write’ ==> Hive ‘update’ or ‘alter’

- If there is no Hive policy that explicitly allows access to the mapped table, access is denied, otherwise access is allowed.

Appropriate tag policies are considered both during S3 access evaluation and if needed, during Hive access evaluation phases. Also, one or more log records are generated to indicate which policy, if any, made the access decision.

The following scenarios illustrate how the access permissions are determined. All scenarios assume that the S3 location is NOT explicitly denied access by a Ranger S3 policy.

Location does not correspond to a Hive table.

In this case, access will be granted only if a Ranger S3 policy allows access. The audit log will show which policy made the decision.

Location corresponds to a Hive table.

A Ranger Hive policy explicitly denied access to the mapped table for any of the accesses derived from the original S3 request.

- Access will be denied by Hive policy.

There is no matching Ranger Hive policy.

- Access will be denied. Audit log will not specify the policy.

Ranger policy masks some columns in the mapped table.

- Access will be denied. Audit log will show Hive masking policy.

Mapped Hive table has a row-filter policy

- Access will be denied. Audit log will show Hive Row-filter policy.

A Ranger Hive policy allows access to the mapped table for the access derived from the original RAZ-S3 access request.

- Access will be granted. If the access was originally granted by S3 policy, the audit log will show Hive policy.

How to full sync the Ranger RMS database

About this task

Performing a full-sync in Ranger RMS truncates all the existing data from RMS tables in the Ranger database. The sync process initiates a fresh synchronization from the Hive Metastore (HMS). All available tables and database metadata download from HMS and persist into the Ranger database.

Before you begin

Have a (7.2.18) cluster with Ranger, Ranger RAZ, and Ranger RMS service up and running.

Note: You must have Admin privileges to perform the following steps.

Procedure

1. In Cloudera Manager Ranger_RMS Actions , click Stop.
2. In Ranger_RMS Actions , click Ranger RMS Database Full Sync.

3. Confirm that the command completed successfully, as shown in the following example:

Figure 2: Confirming successful Ranger RMS database full sync

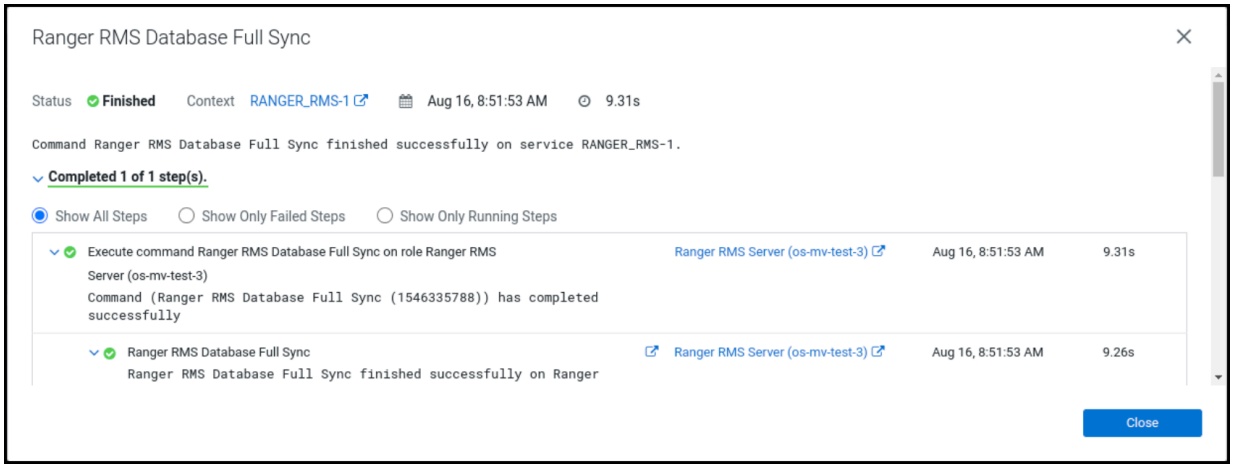

Note: You can also confirm the text - '[I] Truncate operation executed successfully on Ranger RMS tables' from the stderr log of the command from Cloudera Manager UI.

4. In Cloudera Manager Ranger_RMS , click Start.

Configuring Ranger RMS (Hive-S3 ACL Sync)

Ranger Resource Mapping Server (RMS) is fully configured after installing in Cloudera Public Cloud 7.2.18. This topic provides further information about RMS configuration settings and workflows.

Important configuration information - Hive S3

- Ranger RMS enables S3 access via Ranger Hive policies. Ranger RMS is pre-configured with the names of S3 and Hive services (aka Repos). In your installation, there may be multiple Ranger services created for S3 and Hive. These can be seen from the Ranger Admin Web UI. RMS ACL sync is designed to work on a specific pair of S3<->Hive- Ranger services. Therefore, it is important to identify those service names before Ranger RMS is installed. The default value for Ranger S3 service name is cm_s3, and for the Ranger Hive service the default name is cm_hive.
- After Ranger RMS installation, ensure that the Hive service identified in the installation above allows the rangerrms user select access to all tables in all databases in default (no-zone), as well as in all security-zones for the Hive service.
- In case of custom kerberos principal/user, ensure that the Hive service identified in the installation above allows the rangerrmsfoo0 (custom) user select access to all tables in all databases in default (no-zone), as well as in all security-zones for the Hive service.
- In Cloudera Public Cloud, Ranger RMS by default tracks both external and managed tables in Hive. To configure Ranger RMS to track only external Hive tables, add the following configuration setting to Ranger RMS.

```
ranger-rms.HMS.map.managed.tables=false
```

Note:

Locations behind managed tables are granted Read access only, even if the users have Write access at the Hive table level. However, this restriction is not enforced for the hive and impala users.

- To disable RMS for S3 authorization, go to Cloudera Manager Ranger-Ranger RAZ Configuration Advanced Configuration Snippet (Safety Valve) for ranger-raz-conf/ranger-raz-site.xml , then add empty values to the following settings:

```
ranger.raz.service-type.S3.chained.services =
ranger.raz.service-type.S3.chained.services.cm_hive.impl =
```

Note: If any of these configurations are changed after Ranger RMS is started and has synchronized with Hive Metastore, the only way to have Ranger RMS use a new configuration is by performing a “RMS Full Sync”. Please refer to the related topic, "How to full sync the Ranger RMS database" for specific steps.

Advanced configurations

S3 plugin-side configurations

- ranger.raz.service-type.s3.mapping.hive.authorize.with.only.chained.policies
  - true: Enforce strict Sentry semantics.
  - false: If there is no applicable Hive policy, let S3 determine access.
  - Default setting: false
- ranger.raz.service-type.s3.accesstype.mapping.read
  - A comma-separated list of hive access types that S3 "read" maps to.
  - Default setting: select
- ranger.raz.service-type.s3.accesstype.mapping.write
  - A comma-separated list of hive access types that S3 "write" maps to.
  - Default setting: update,alter
- ranger.raz.service-type.s3.privileged.user.names
  - Default setting: admin,dpprofiler,hue,beacon,hive,impala

    Note: The comma-separated lists that you define for S3 privileged user names and service names are users that, based on default hive policies, have all access for all Hive resources. Therefore, for these users, checking Hive policies when they access storage locations which map to Hive resources is unnecessary, and may cause access violations if masking/row-filtering policies are configured for public group.
- ranger.raz.service-type.s3.mapping.source.download.interval
  - The time in milliseconds between mappings download requests from the S3 Ranger plugin to RMS.
  - Default setting: 30 seconds

    By default, the RMS plugin checks for new mapping downloads every 30 seconds, based on this configuration. If you have mapping data (found in the raz_cm_hive_resource_mapping.json file) of approximately 360MB file size; then performing this operation every 30 seconds could cause an excessive load on the NameNode. After enabling performance logs, we can observe that saveToCache takes 11 seconds and loadFromCache operations take 7 seconds to complete. The cacheing process takes approximately 18~19 seconds to complete, as shown in the following example performance logs:

```
DEBUG org.apache.ranger.perf.resourcemapping.init: [PERF] RangerMappingR
efresher.loadFromCache(serviceName=cm_hive): 7449
DEBUG org.apache.ranger.perf.resourcemapping.init: [PERF] RangerMappin
gRefresher.saveToCache(serviceName=cm_hive): 11787
```

    In this case, you should adjust the frequency of download RMS mappings to at least 18*2= 36 seconds. A more conservative value = 45 seconds. In this way, you can tune RMS configurations to optimize performance in the RAZ S3 chained plugin.

- ranger.plugin.hdfs.bypass.chained.plugin.evaluation.if.access.is.determined
  - boolean valued configuration variable, default value = false
  - When it is set to true, the chained-plugin (in our case, RMS code for looking up and enforcing Hive policies) is NOT executed if a matching Ranger HDFS policy was found for the accessed HDFS location and that policy determined the access result (either ALLOW or DENY).

Hive service configuration

- ranger.plugin.audit.exclude.users
  - This configuration, added in the Hive service-configs, lists the users whose access to Hive or Hive Metastore does not generate audit records. There may be a large number of audit records created when "rangerrms" makes requests to the Hive Metastore when downloading Hive table data. By adding the "rangerrms" user to the comma-separated list of users in this configuration, such audit records will not be generated.

RMS side configurations

Note: Changes to any of these requires that RMS is stopped, all rows are deleted from RMS database table x_rms_mapping_provider and RMS is restarted. On restart, RMS downloads complete table data from RMS, which may take a significant amount of time depending on the number of tables in HMS.

- ranger-rms.HMS.source.service.name
  - The Ranger S3 service name (default: cm_s3).
- ranger-rms.HMS.target.service.name
  - The Ranger Hive service name (default: cm_hive).
- ranger-rms.HMS.map.managed.tables
  - true – Track managed and external tables.
  - false – Track only external tables.
  - Default setting: true
- ranger-rms.polling.notifications.frequency.ms
  - The time in milliseconds between polls from RMS to HMS for changes to tables.
  - Default setting: 30 seconds
- ranger-rms.supported.uri.scheme
  - A comma-separated list of uri schemes supported by RMS
  - Default setting : s3a

Ranger RMS (Hive-S3 ACL-Sync) Use Cases

This topic presents a few common use cases for Ranger RMS (Hive-S3 ACL-Sync).

Use Case 1: RMS Hive policies control access to a table's S3 directories

Prerequisites:

1. Create a "Customer" Hive table under the default database.
2. Create a "unixuser1" user.
3. User "unixuser1" does not have any policy to allow it access to table "Customer".
4. User "unixuser1" tries to access the storage location through the hdfs command.

Before setting up RMS:

If S3 policies configured through Ranger Admin allow access to the location for Customer table, access will be granted to "unixuser1". The audit log will have "ramger-acl" as the access enforcer.

After setting up RMS:

Access will not be granted to user "unixuser1". The audit log will not specify denying policy.

Use Case 2: RMS Hive policies propagate tag-based access control on tables to S3 directories

Prerequisites:

1. Create a "Customer" Hive table under the default database.
2. Create a "unixuser1" user.
3. The tag "SPECIAL_ACCESS" is associated with the "Customer" table.
4. A policy for the tag "SPECIAL_ACCESS" provides Hive select access to "unixuser1".
5. User "unixuser1" tries to read the Hive data through the S3 command.

Before setting up RMS:

If S3 policies configureed through Ranger Admin allow access to the location of "Customer" table, access will be granted to ‘"unixuser1". Otherwise, access is denied.

After setting up RMS:

Access will be granted by tag-based policy for "SPECIAL_ACCESS".

Use Case 3: RMS Hive policies propagate tag-based masking on tables and denies access to S3 directories

Prerequisites:

1. Create a "Customer" Hive table under the default database.
2. Create a "unixuser1" user.
3. The tag "SPECIAL_ACCESS" is associated with the "Customer" table.
4. A policy for the tag "SPECIAL_ACCESS" provides Hive select access to "unixuser1".
5. A masking policy for the "Customer" table is set up so that for "unixuser1" a column "SSN" is redacted.
6. User "unixuser1" tries to read the Hive data through the hdfs command.

Before setting up RMS:

If S3 policies configured through Ranger Admin allow access to the location of Customer table, access will be granted to "unixuser1". Otherwise, access is denied.

After setting up RMS:

Access will be denied by the masking policy.

Use Case 4: RMS Hive policies take precedence over S3 policies

Prerequisites:

1. Create a "Customer" Hive table under the default database.
2. Create a "unixuser1" user.
3. User "unixuser1" has a S3 policy allowing read access.
4. User "unixuser1" has a Hive policy to allow it access to the "Customer" table.
5. User "unixuser1" tries to access the Hive data through the hdfs command.

Before setting up RMS:

Access will be granted by the Ranger S3 policy.

After setting up RMS:

Access will be granted to the "unixuser1" user through the Hive policy. The audit log should display the same.

Cloudera Runtime 7.3.1

Apache Ranger User Management

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Administering Ranger Users, Groups, Roles, and Permissions..... 4
Adding a user.....5
Editing a user.....7
Deleting a user.....10
Adding a group.....10
Editing a group.....12
Deleting a group.....13
Adding a role through Ranger.....15
Adding a role through Hive.....17
Editing a role.....19
Deleting a role.....20
Adding or editing module permissions.....21

Ranger Usersync..... 24
Adding default service users and roles for Ranger.....33
Configuring Usersync assignment of Admin users.....33
Configuring nested group hierarchies.....35
Configuring Ranger Usersync for Deleted Users and Groups.....37
Configuring Ranger Usersync for invalid usernames.....41
Setting credentials for Ranger Usersync custom keystore.....42
Enabling Ranger Usersync search to generate internally.....43
Configuring Usersync to sync directly with LDAP/AD (FreeIPA).....44
Configure SASL Bind in Ranger Usersync.....45
Force deletion of external users and groups from the Ranger database.....47

Administering Ranger Users, Groups, Roles, and Permissions

Ranger Admin users can manage users, group, roles and permissions using Ranger Admin Web UI.

Overview: Ranger User/Groups/Roles

Ranger Admin Web UI allows users with Administrator role (permissions) to create new users, groups and roles that define fine-grained access control in Cloudera. This topic presents an overview of the user, group, role, permission management options you can find under Settings.

To list the users, groups, and roles for which Ranger manages access to Cloudera services, select Ranger Admin Web UI Settings Users/Groups/Roles .

Users lists:

- Internal users - created by a Ranger admin user who can log in to the Ranger Admin Web UI.
- External users - created at other systems such as Active Directory, LDAP, or UNIX.
- Admin users - who are the only users with permission to create users and services, run reports, and perform other administrative tasks using Ranger Admin Web UI.
- Visible users - those users created in Ranger Admin Web UI, or in other systems who are "active", in other words, not marked for deletion.
- Hidden users - those users that have been marked for deletion for any reason (for example invalid characters, duplicates, or obsolescence).

Users also shows the Groups to which each user belongs.

The following example shows internal, external, and Admin users listed on Service Manager Users :

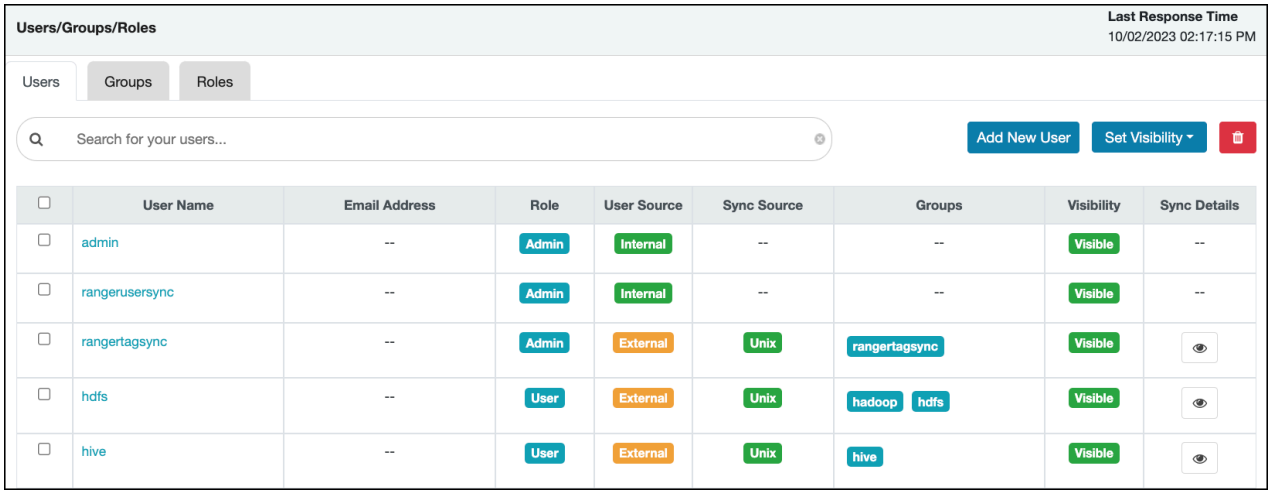

Groups lists:

- Internal groups - created by a Ranger admin.
- External groups - created by other systems.
- On Groups, you can click Users to view the members of a specific group.

The following figure shows internal and external groups listed on Groups:

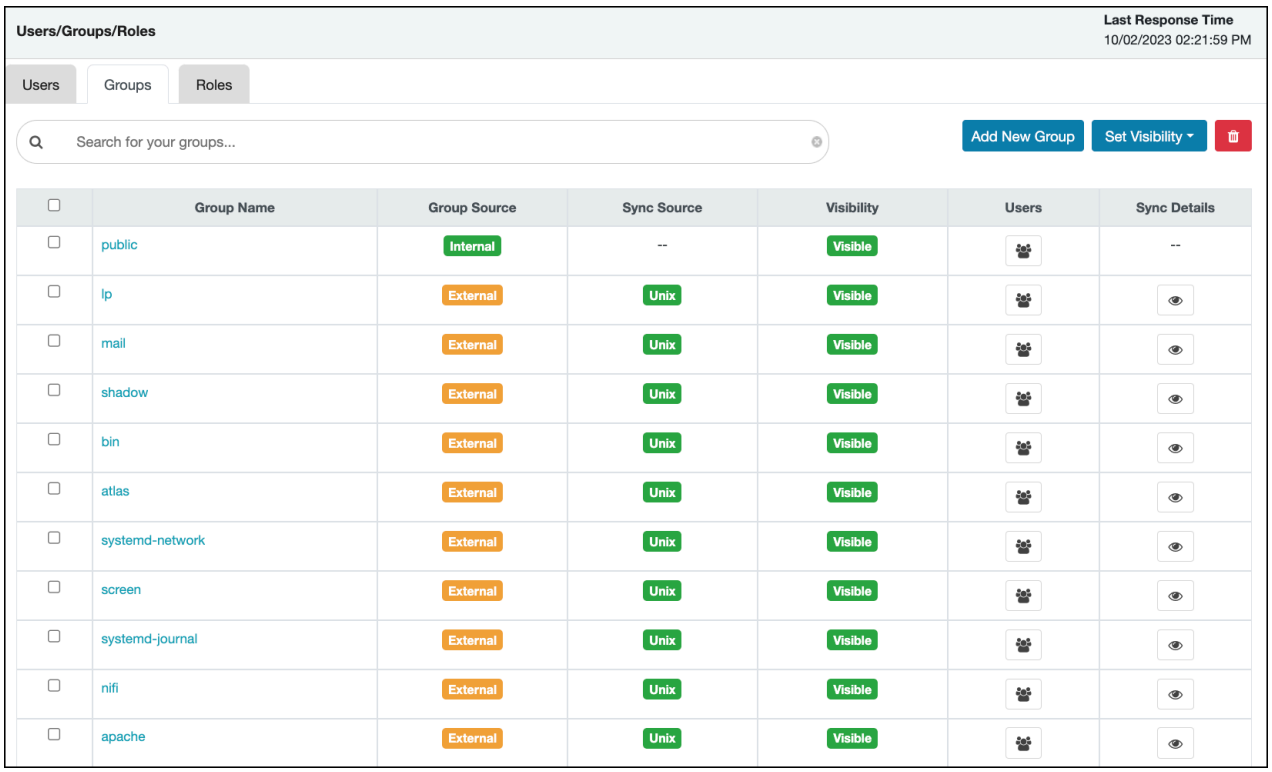

The Users and Groups pages also lists a Sync Source for each user and group. To filter Users and Groups by sync source type, select Sync Source as a search filter, then enter a sync source type, such as Unix or LDAP/AD. To view more information about the sync source, click Sync Details for a user or group.

The following example shows the sync details for the rangertagsync user.

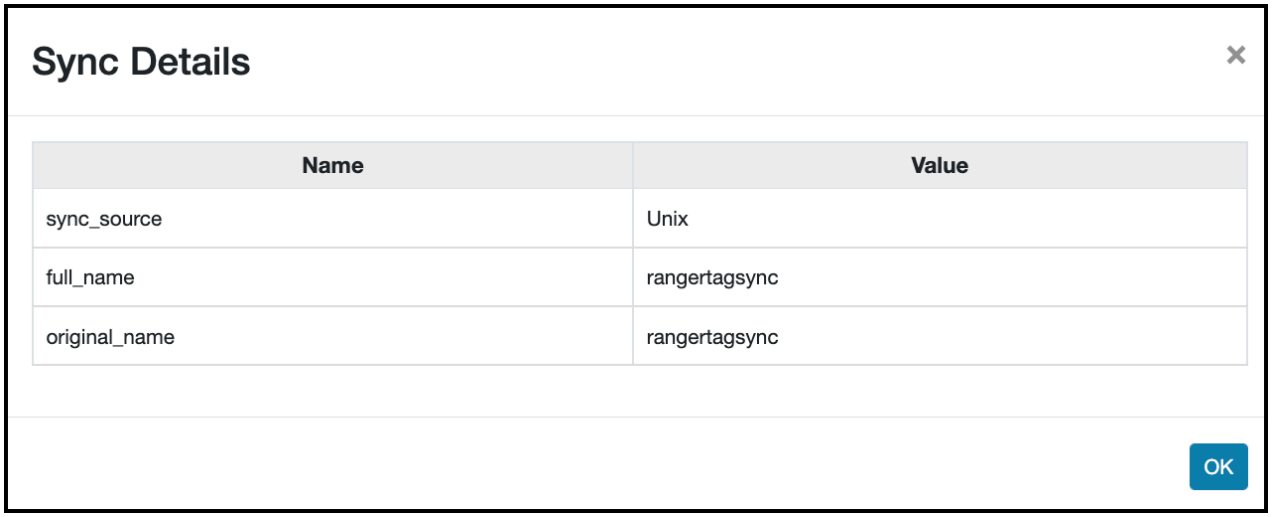

Roles lists:

- Role names, and related mappings to:
- User names
- Group names
- Other role names

Adding a user

How a Ranger Admin user can add new Ranger users.

About this task

Only a Ranger Admin user can create other Admin users, service users, or Auditor users, based on the full permissions to configure Ranger Admin Web UI.

Admin users can create the following user types:

admin

auditor

keyadmin

user

A Ranger Admin user can also import/ export policies for services (for example hdfs, hive, atlas, etc.) other than kms.

KeyAdmin users cannot create users, but can:

Import kms policies

Export kms policies

access Key Manager module functionality with full permissions.

Auditor users cannot create users, but can:

access Audit Manager module functionality with full permissions.

This topic presents the example of logging in to Ranger Admin Web UI, using admin credentials and then creating a new user with Auditor role.

Procedure

1. Log in to Ranger Admin Web UI, using administrator credentials.
2. In the Ranger Admin Web UI, select Settings Users .

   Users/Groups/Roles displays the Users tab active.

   Note: Users displays the Add New Users option only to users logged in with admin permissions.

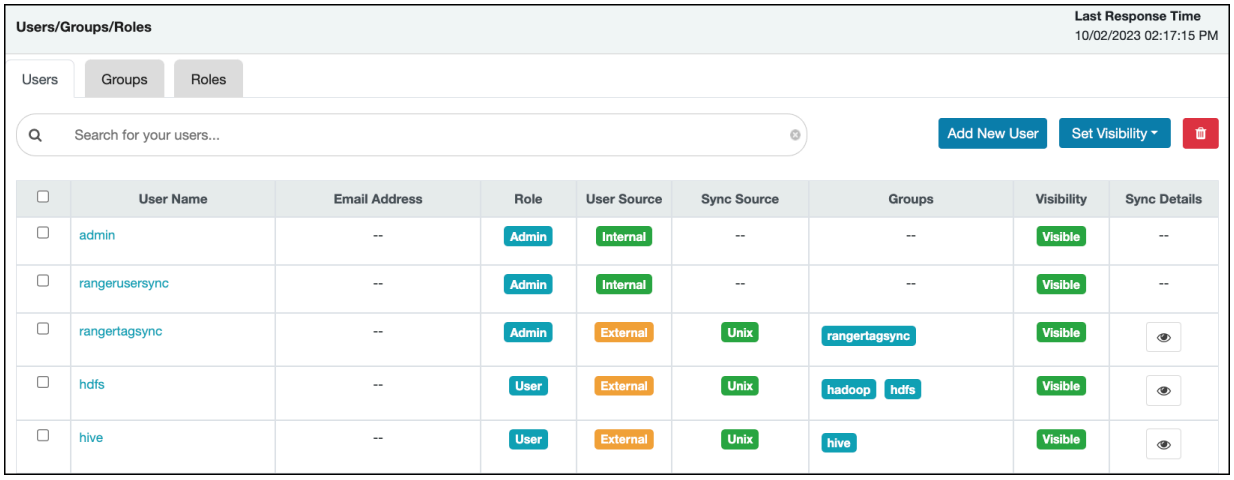

3. Click Add New User .
   User Detail displays User Create options.

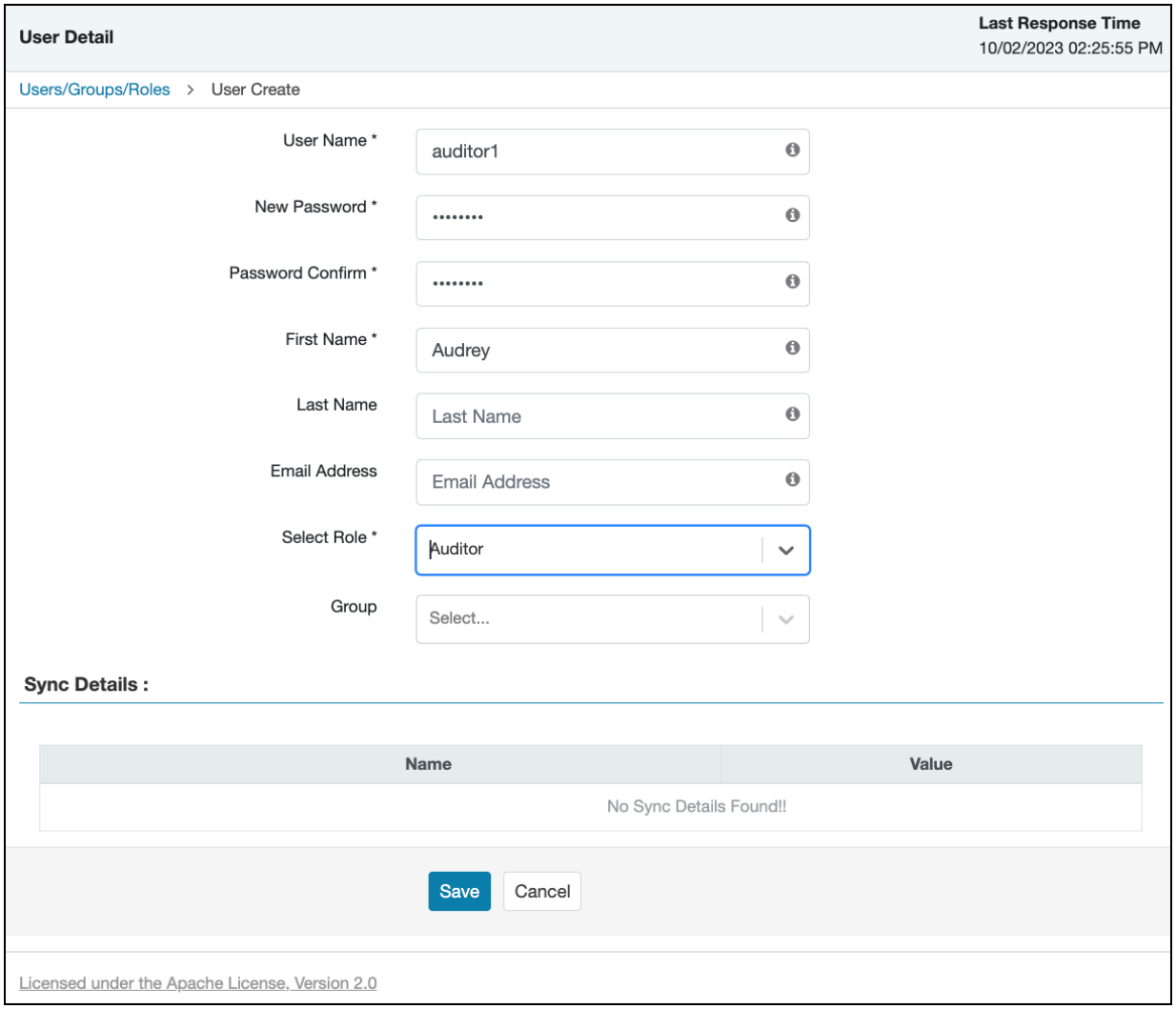

4. On User Create, enter information in ALL REQUIRED (*) fields, then click Save.

   In this example, Auditor has been selected in Select Role. No group has been selected, nor has the password been confirmed.

   After completing all required fields, clicking Save immediately creates the user and adds the user to any selected groups.

Editing a user

How to edit a user in Ranger.

Procedure

1. In the Ranger Admin Web UI, select Settings Users .
   Users/Groups/Roles displays the Users tab active.

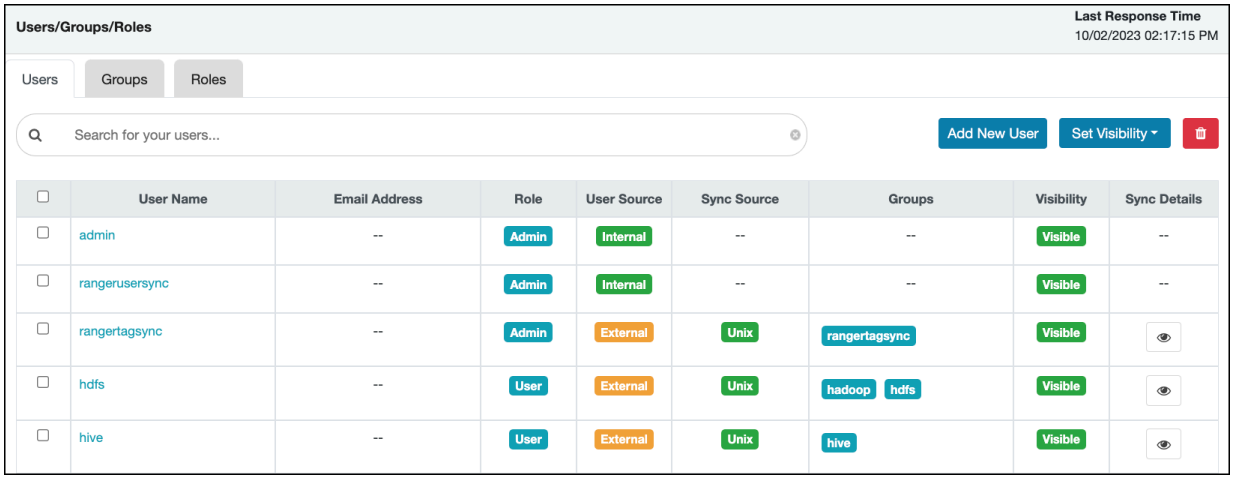

2. Select a user profile to edit. To edit your own profile, select your user name, then click Profile.

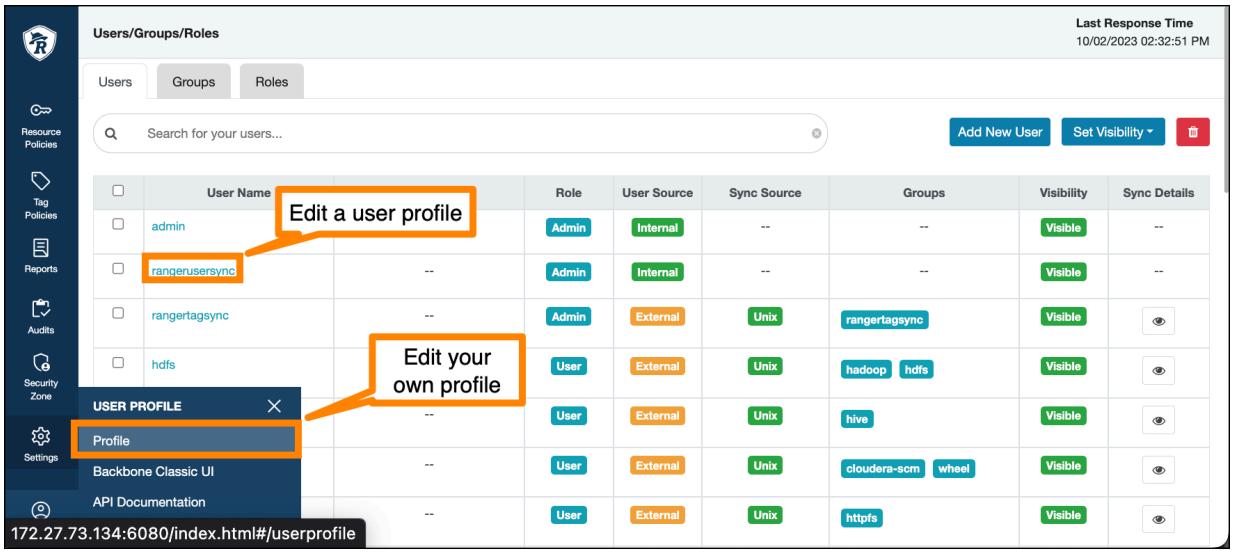

The User Detail page appears.

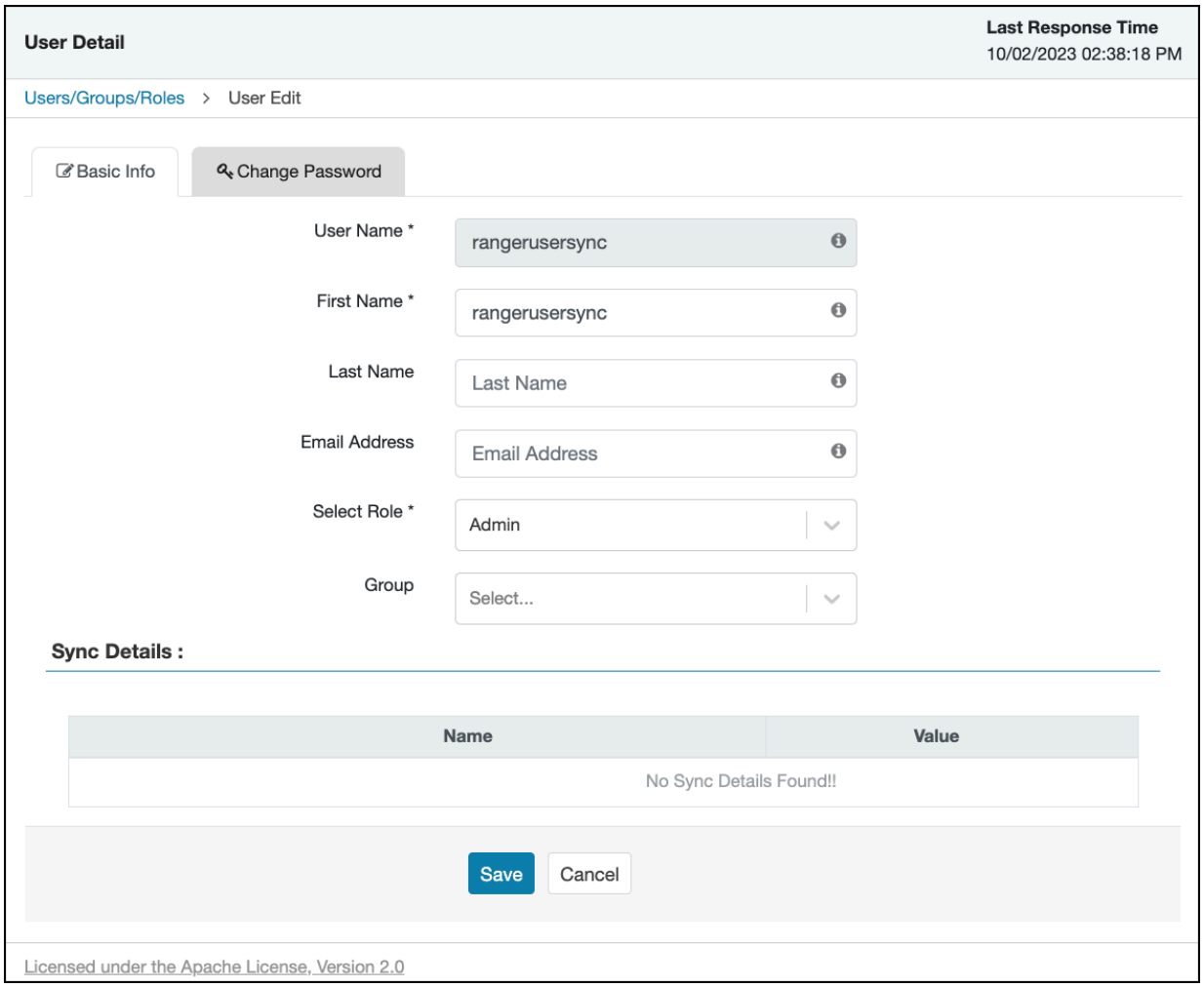

Note:

You can only fully edit internal users. For external users, you can only edit the user role.

3. Edit the user details, then click Save.

Related Tasks

Deleting a group

Deleting a user

How to delete a user in Ranger.

Before you begin

Only users with the "admin" role can delete a user.

Procedure

1. In the Ranger Admin Web UI, select Settings Users .
   Users/Groups/Roles displays the Users tab active.

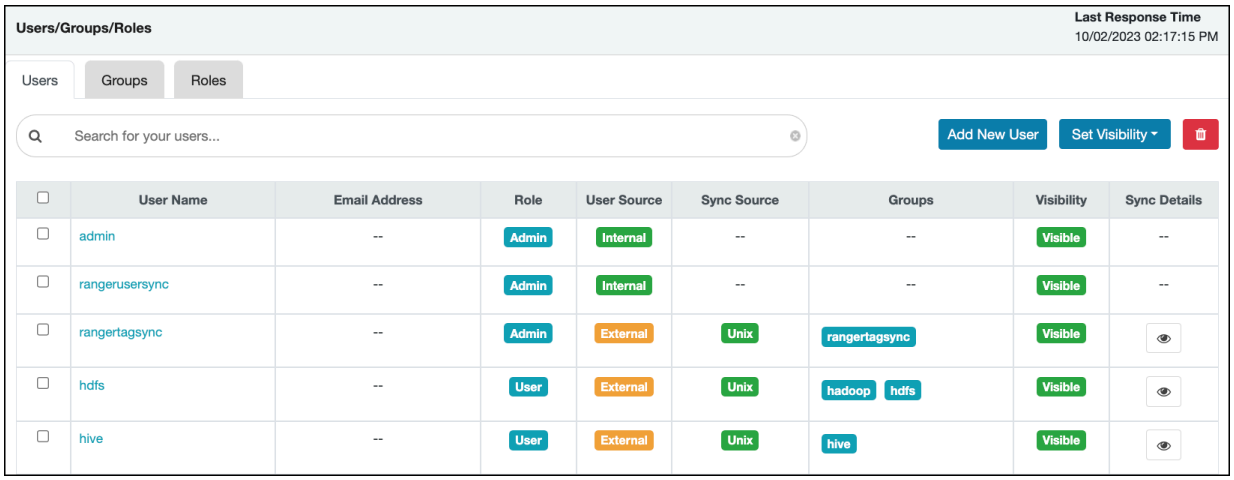

2. Select the check box of the user you want to delete, then click Delete.

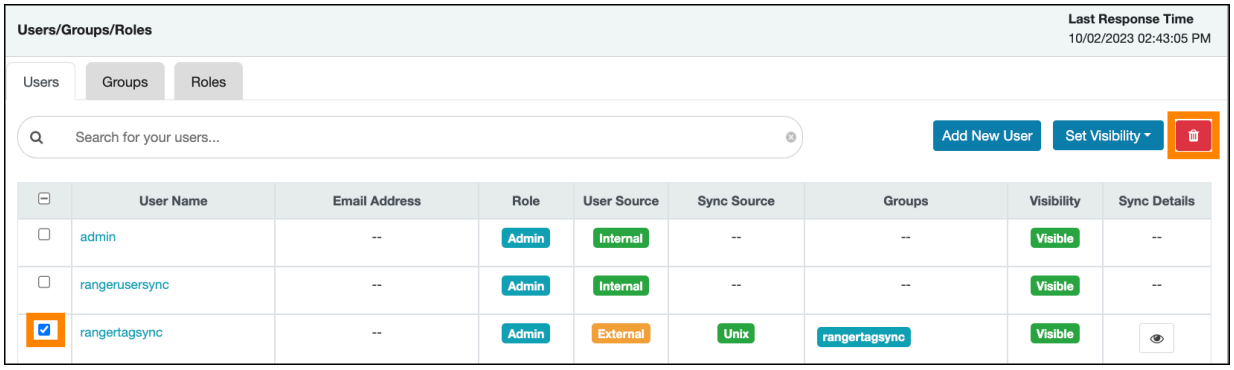

3. Click OK on the confirmation pop-up.

Adding a group

How to add a group in Ranger.

Procedure

1. In the Ranger Admin Web UI, select Settings Groups .
   Users/Groups/Roles displays the Groups tab active.

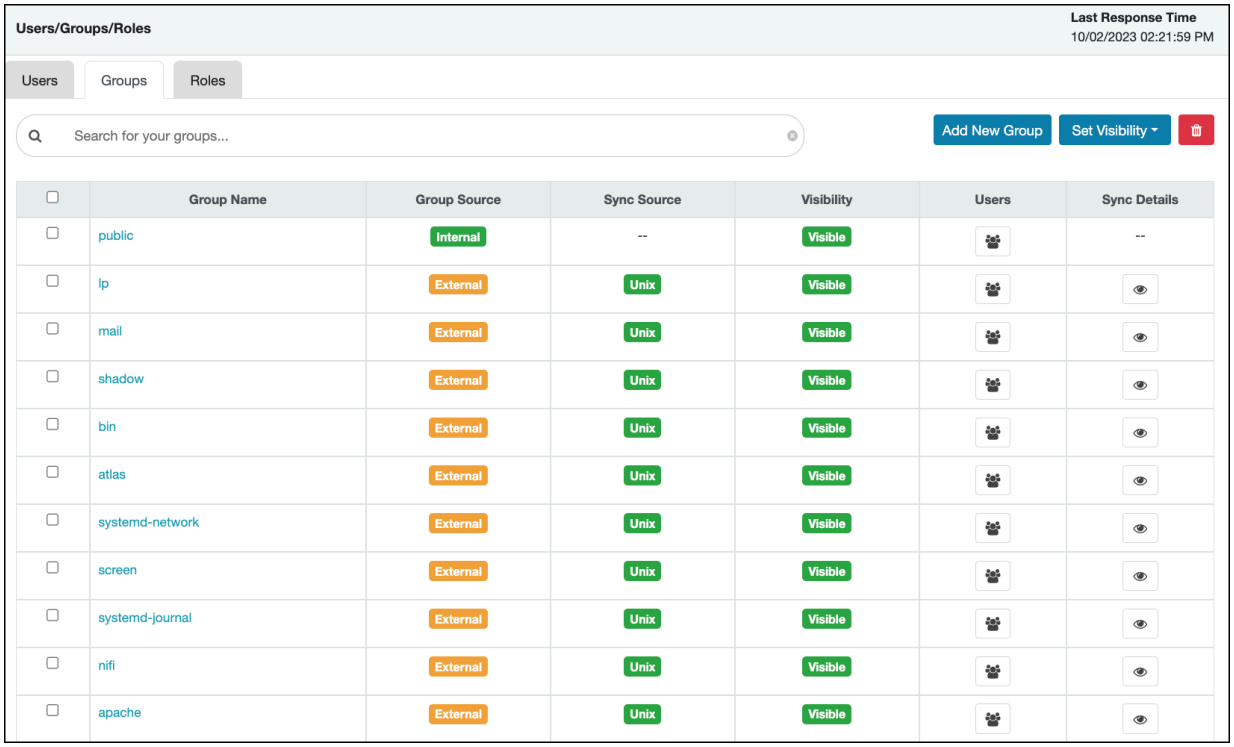

2. Click Add New Group.

   The Group Create page appears.

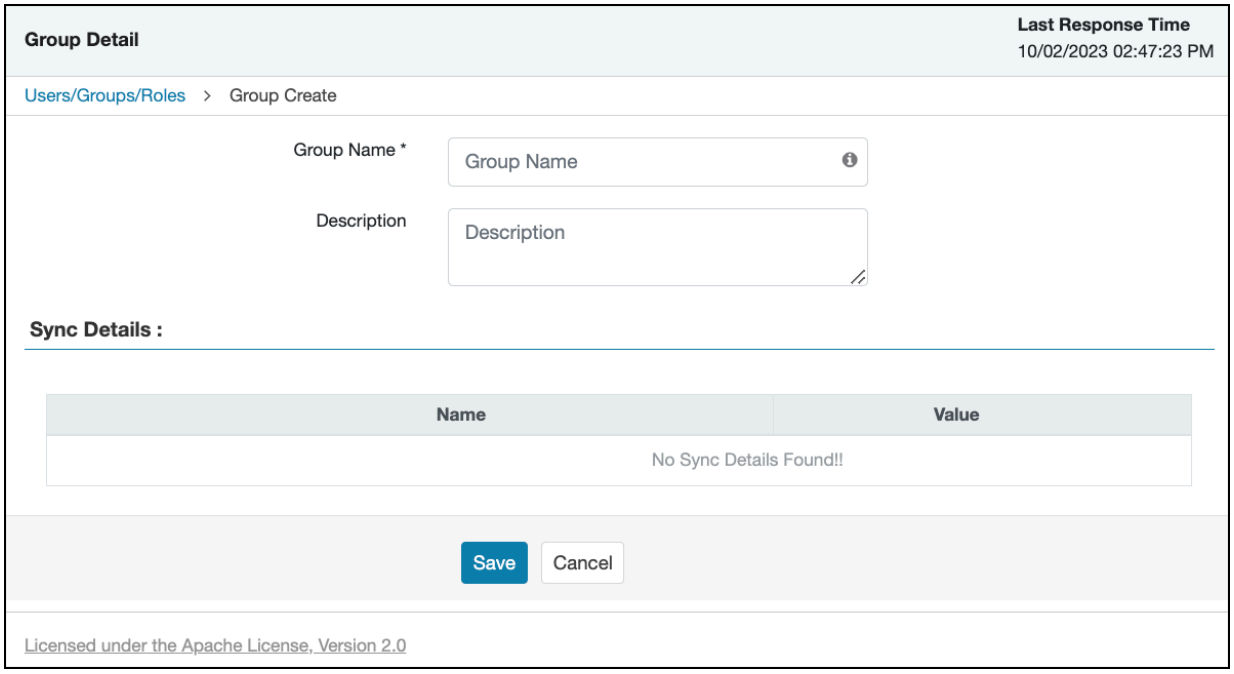

3. Enter a unique name for the group (required) and description (optional), then click Save.

Editing a group

How to edit a group in Ranger.

Procedure

1. In the Ranger Admin Web UI, select Settings Groups .
   Users/Groups/Roles displays the Groups tab active.

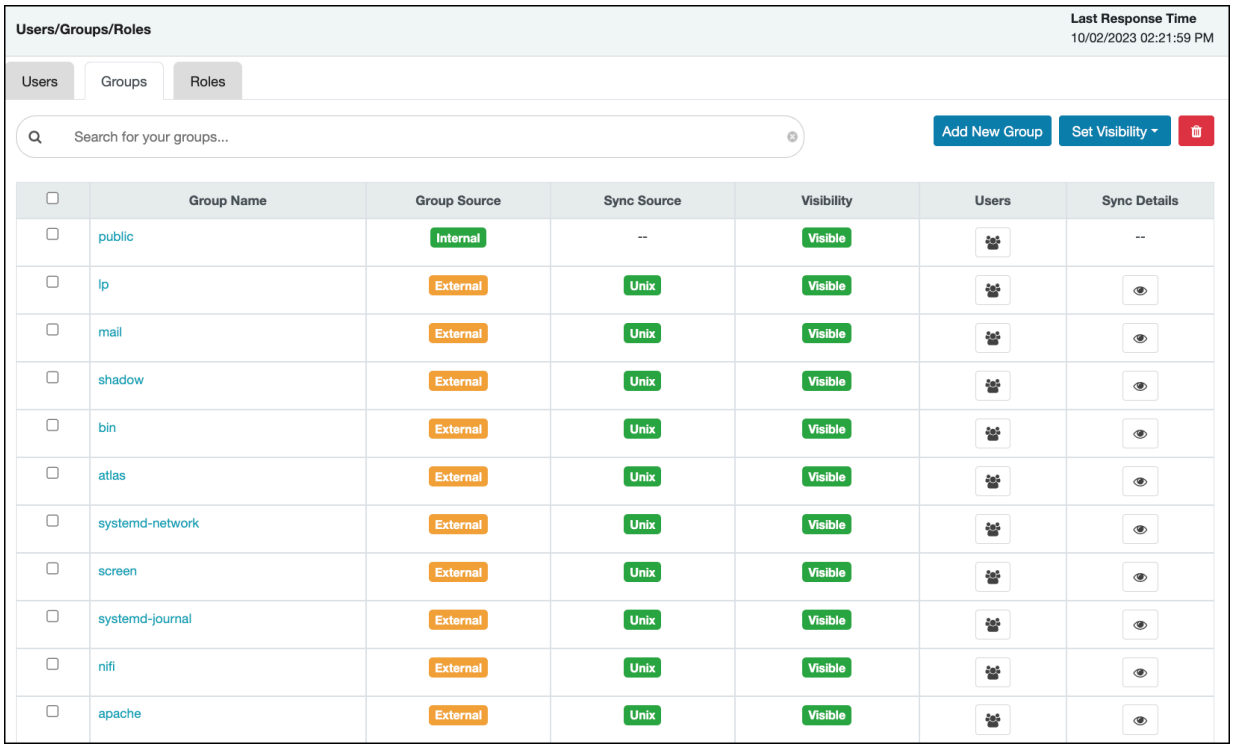

2. Select a group name to edit, then double-click the group name.

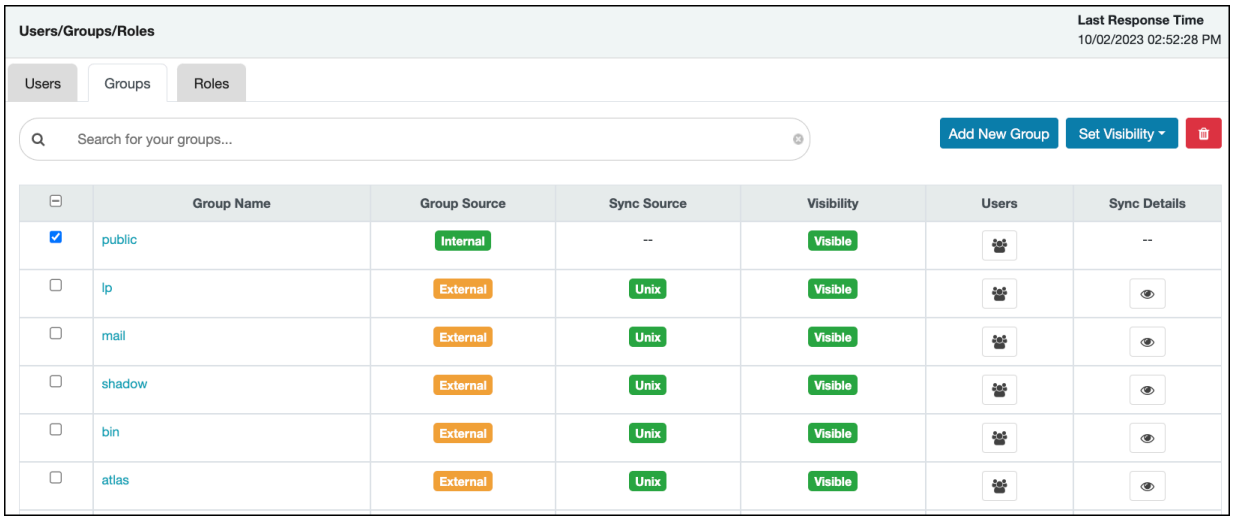

3. The Group Edit page appears.

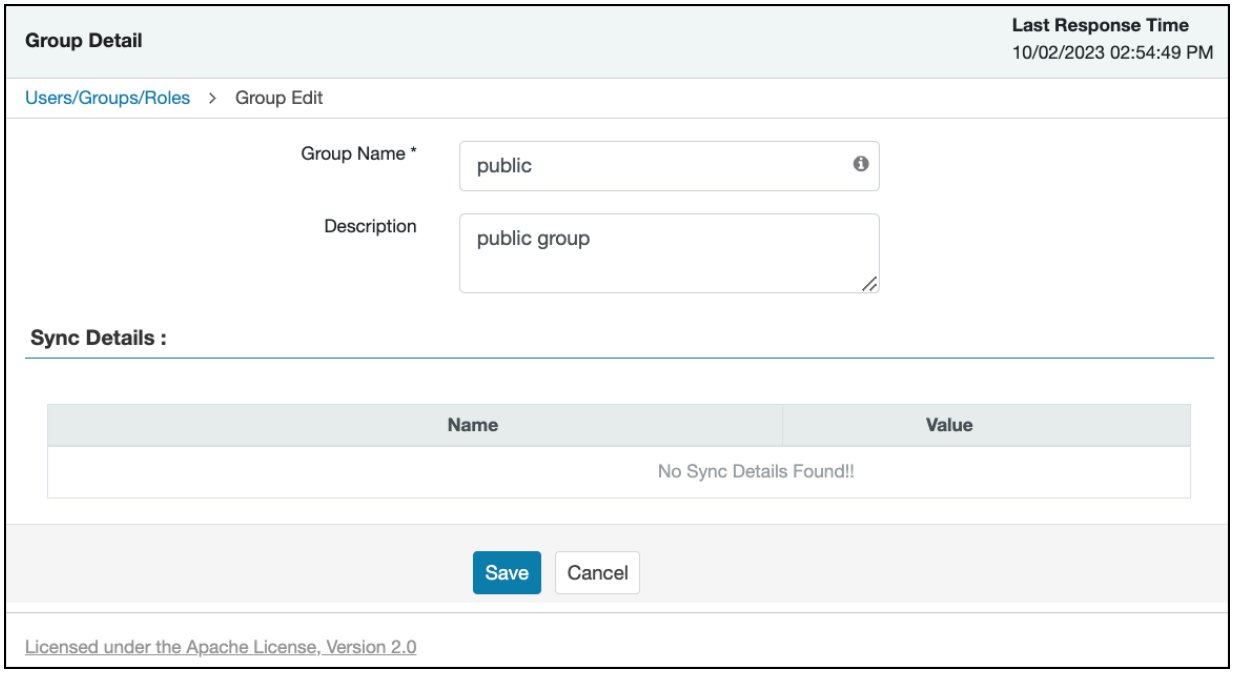

4. Edit the group details, then click Save.

Deleting a group

How to delete a group in Ranger.

Before you begin

Only users with the "admin" role can delete a group.

Procedure

1. In the Ranger Admin Web UI, select Settings Groups .
   Users/Groups/Roles displays the Groups tab active.

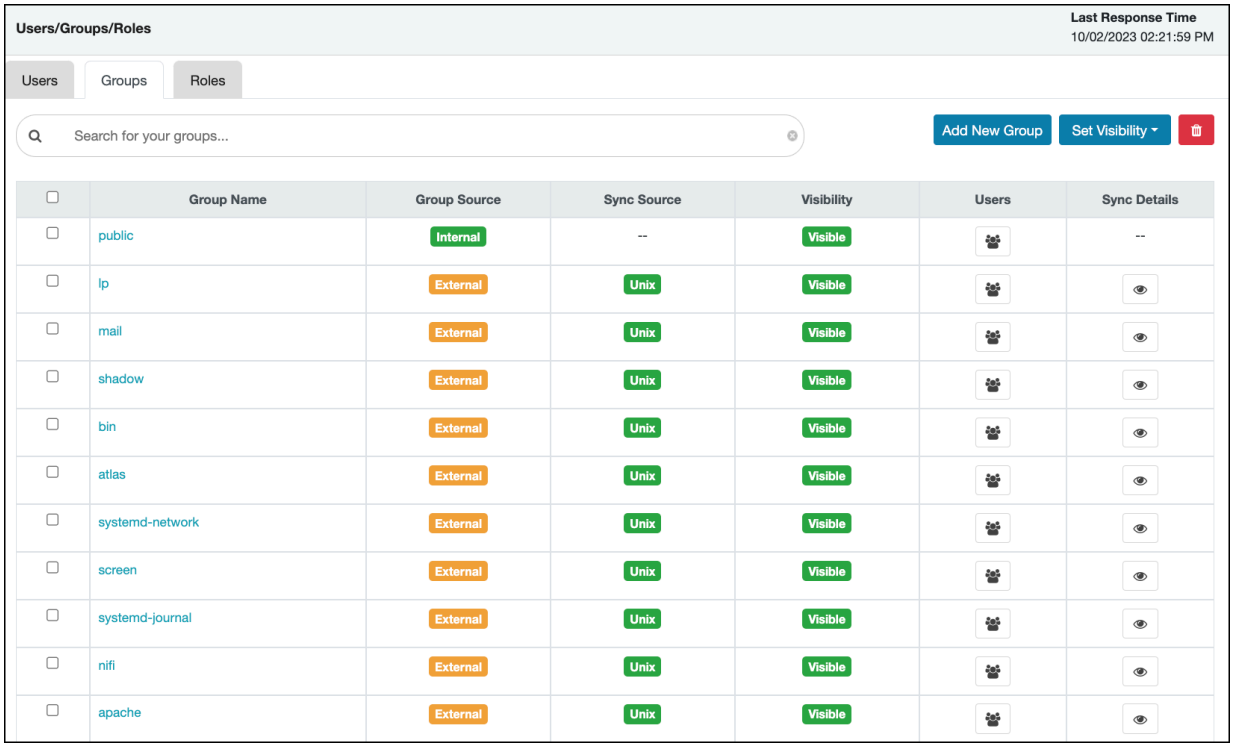

2. Select the check box of the group you want to delete, then click Delete.

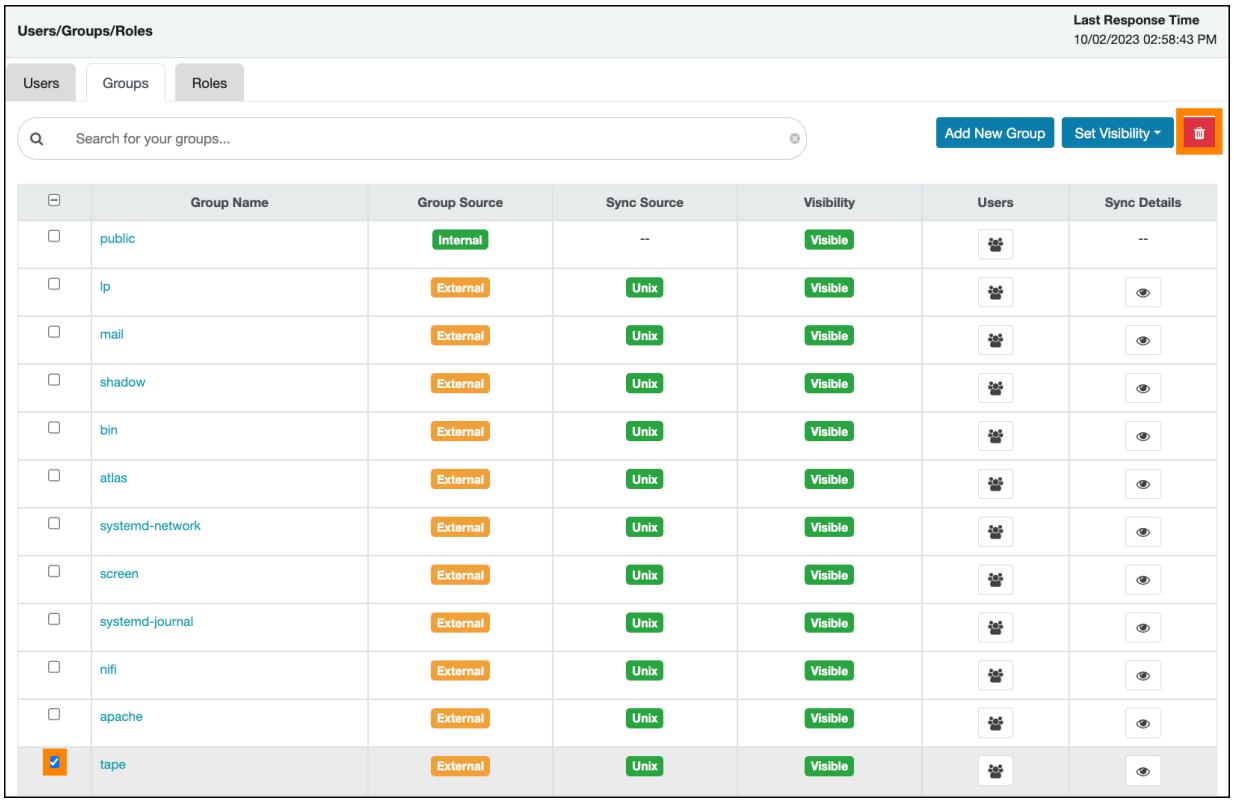

3. Click OK on the confirmation pop-up.

What to do next

Users in a deleted group will be reassigned to no group. You can edit these users and reassign them to other groups.

Related Tasks

Editing a user

Adding a role through Ranger

How to add a role in Ranger.

About this task

A role contains a set of users, groups, or other roles. You assign a role by adding a user, group or role to it. By adding multiple roles, you create a role hierarchy in which you manage permission sets at the role level.

Benefits that roles provide in a large environment:

- A role may include many users or groups, all of which may be updated using a single command.
- Adding or revoking a single permission to or from a role requires a single command, which also applies to all users and groups with that role.
- Roles allow for some documentation about why a permission is granted or revoked.

Conceptually, a role functions as a collection. A group is a collection of users. You create a role and add users to it. Then, you grant that role to a group. Roles present an easier way to manage users and groups, based on specific access criteria.

A simple example of a role hierarchy follows:

- FinReadOnly role, which gives read permission on all tables in the Finance database and is defined by a Ranger policy that grants read on database:Finance, table:* to the FinReadOnly role.
- FinWrite role, which gives write permission on all tables in the Finance database and is defined by a Ranger policy that grants write on database:Finance, table:* to the FinWrite role.
- FinReadWrite role, which role is granted both the FinRead and FinWrite roles and thereby inherits read and write permission to all tables in the Finance database.
- FinReporting group whose users require only read permission to the Finance tables. FinReporting group is added to FinReadOnly role in Ranger.
- FinDataPrep group whose users require only write permission to the Finance tables. FinDataPrep group is added to the FinWrite role in Ranger.
- FinPowerUser group whose users require read and write permission to all Finance tables. FinPowerUsers group is added to the FinReadWrite role in Ranger.

You can create a role either through Ranger, or through Hive.

Before you begin

To add a role, the user must have Admin_Role privilege in Ranger.

Procedure

To create a role through Ranger, using Ranger Admin Web UI:

1. Select Settings Roles .
   Users/Groups/Roles displays the Roles tab active.

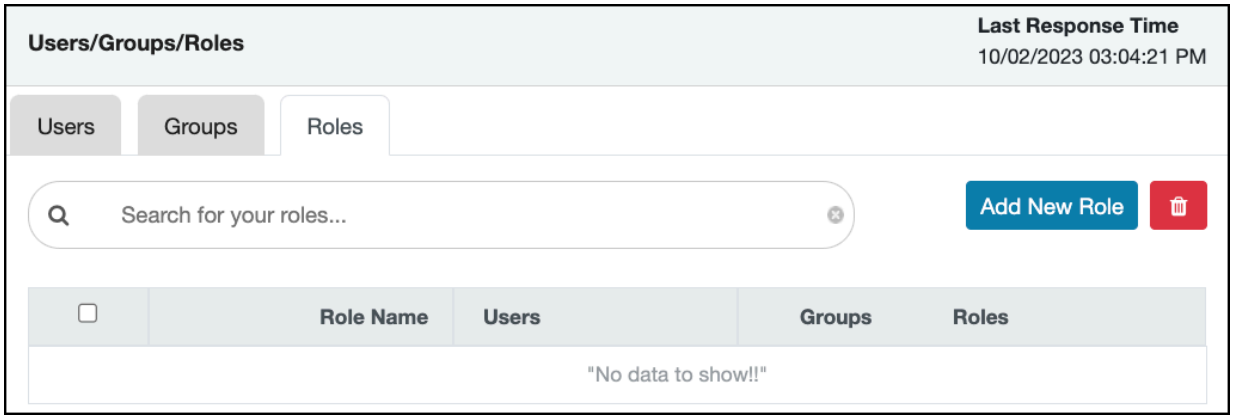

2. Click Add New Role.
   The Role Detail page displays Role Create options.

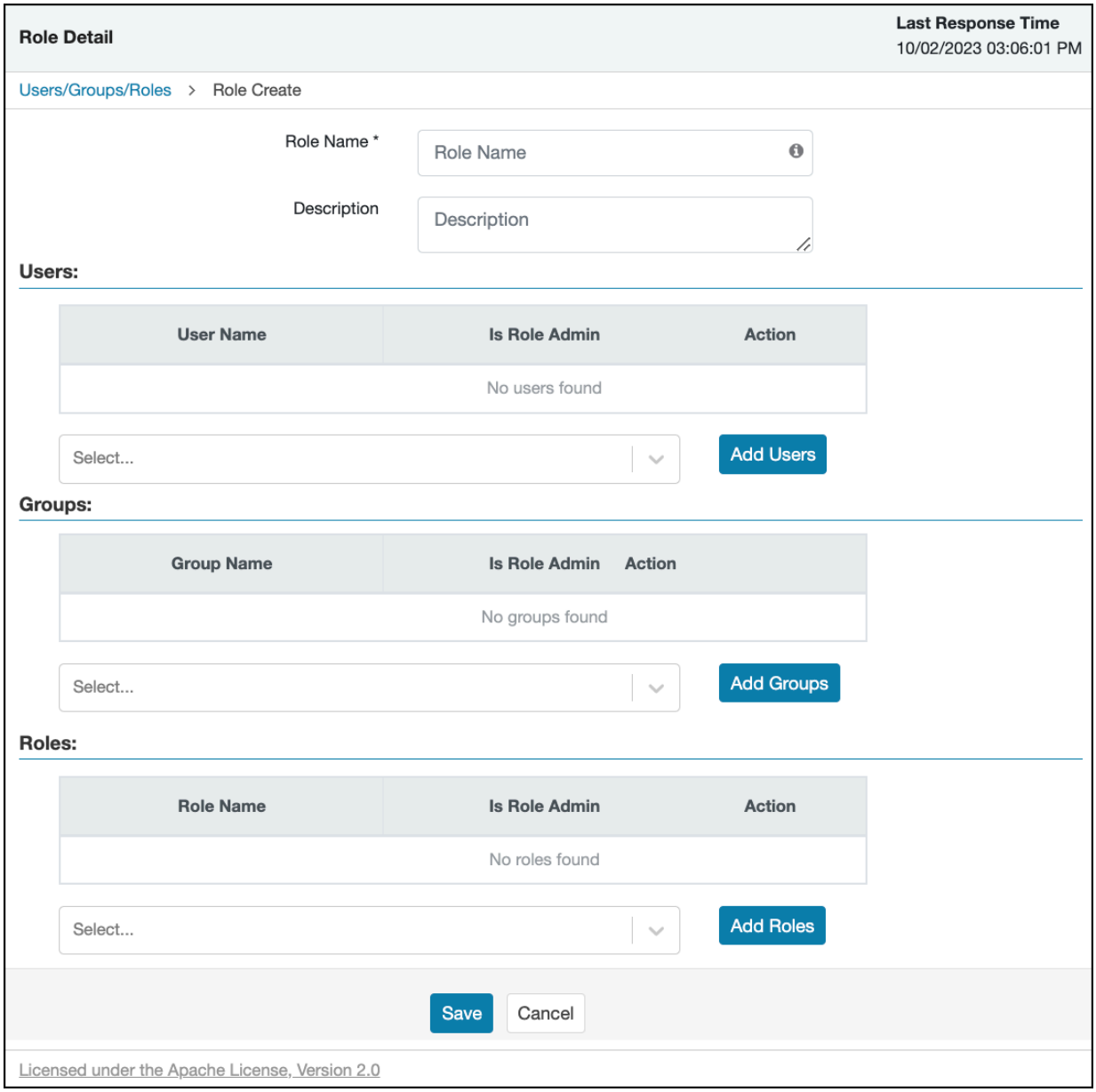

3. Enter a unique name for the role. Optionally, add users, groups and/or roles to be associated with the role, then click Save.

Adding a role through Hive

How to add a role in Hive.

About this task

You can create a role either through Ranger, or through Hive.

Before you begin

To add a role through Hive, the user must have Admin_Role privilege in Ranger.

Procedure

In Hive:

1. Log in as a user with Admin_Role privilege.

2. Type the following command:

```
CREATE ROLE external_hr_role_01;
```

Any user with Is_Role_Admin privilege has the ability to assign the role to other users in Hive.

For example, to grant this new role to the user hr_user01, type:

```
GRANT ROLE external_hr_role_01 TO USER hr_user01;
```

hr_user01 appears in Ranger having the external_hr_role_01 role.

You can also grant Is_Role_Admin privilege to a specific user by typing:

```
GRANT ROLE external_hr_role_01 TO USER hr_user02 WITH ADMIN OPTION;
```

The role you create appears in Ranger and is recognized by Hive. The user that creates the role adds automatically to the list of users having that role. The added user has the Is_Role_Admin privilege, as shown in Ranger:

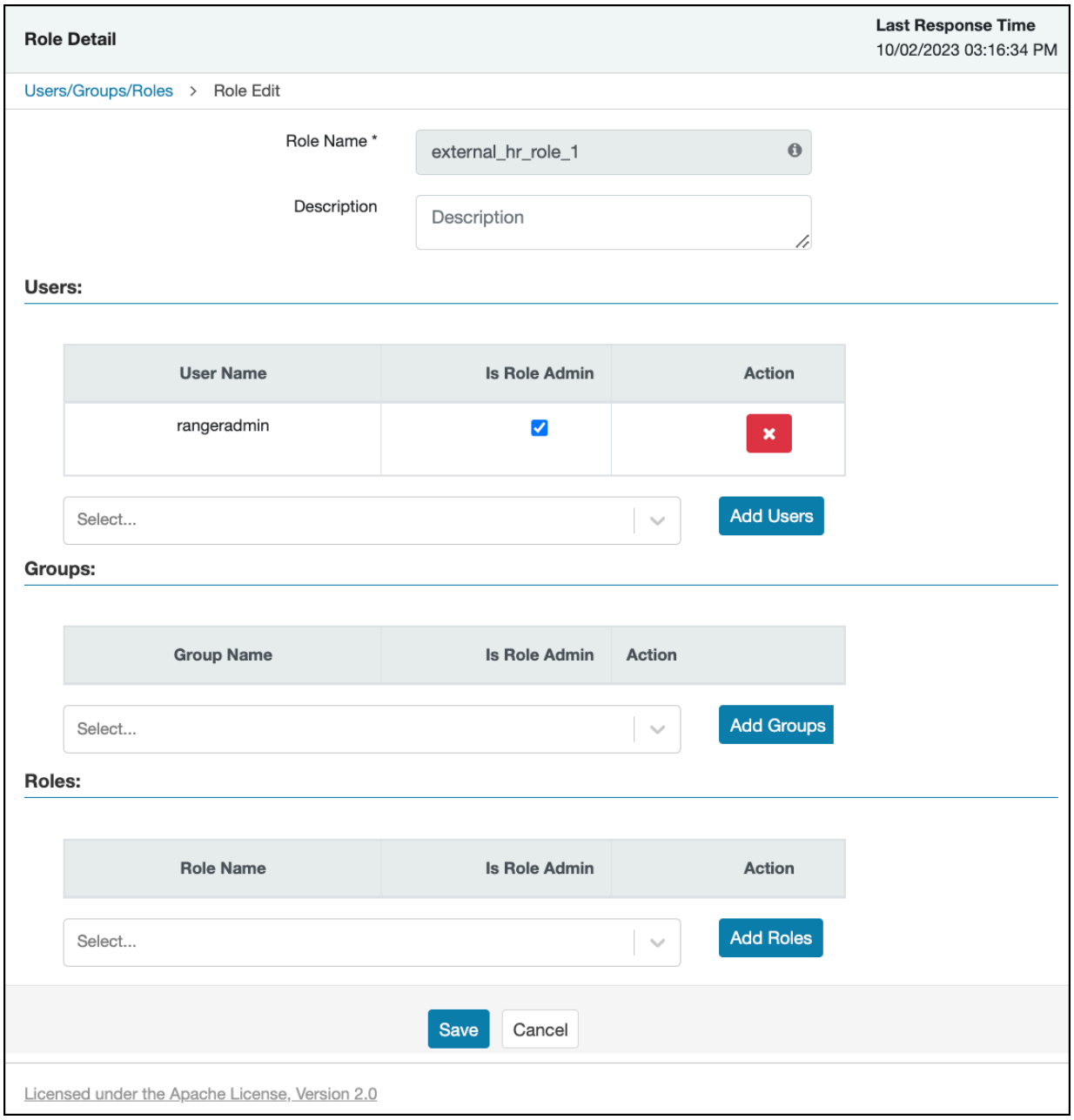

Editing a role

How to edit a role in Ranger.

Procedure

1. Select Settings Roles .
2. Click the Roles tab.
   Users/Groups/Roles displays the Roles tab active.

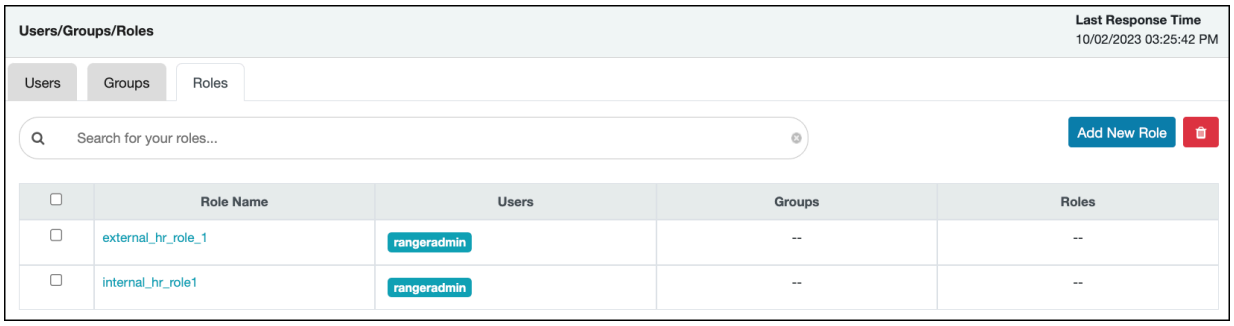

3. Click the role name to edit.
   The selected role opens for editing in Role Detail.

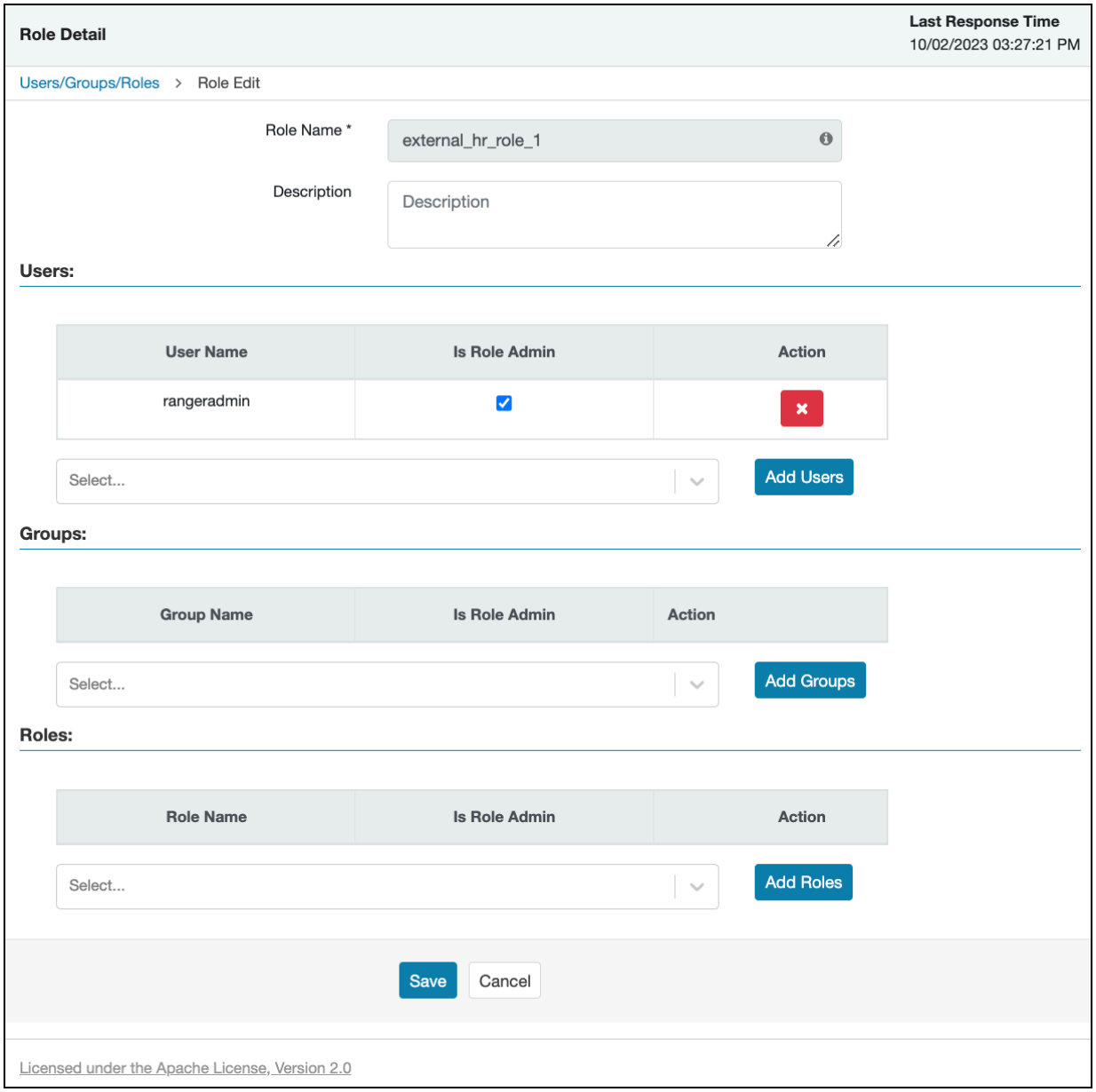

4. Add users, groups and roles to the existing role, then click Save.
   If the role was created in Hive, you can add other users in Hive using the GRANT command:

```
GRANT ROLE external_hr_role_01 TO USER hr_user02;
```

Deleting a role

How to delete a role in Ranger.

Procedure

1. Select Settings Roles .

2. Click the Roles tab.
   Users/Groups/Roles displays the Roles tab active.

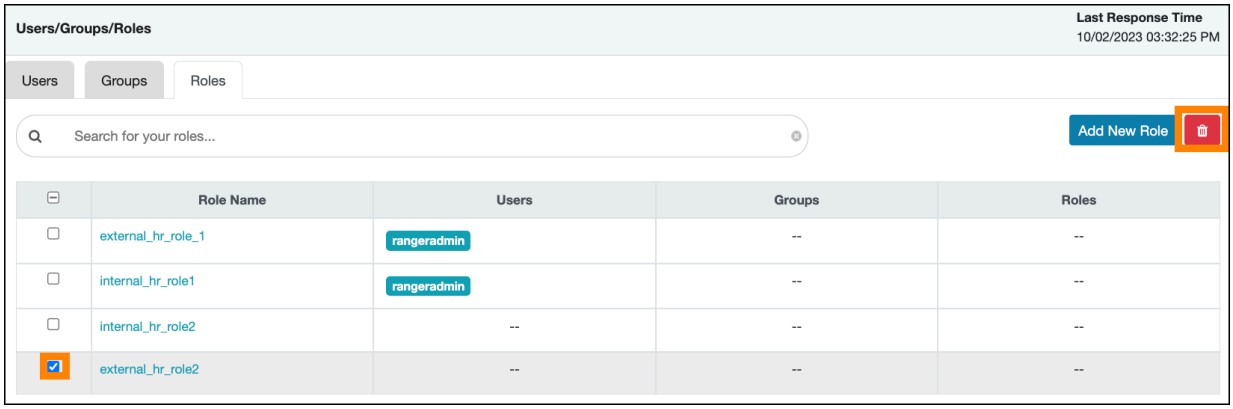

3. Select the role you want to delete, then click Delete.
4. After deleting any roles, click Save .
   If the role was created in Hive, you can delete the role through Hive using the Drop command:

```
DROP ROLE internal_hr_role_02;
```

Adding or editing module permissions

How to add or edit users and groups permission to access Ranger modules.

About this task

Permissions defines the users and groups that can access Ranger modules using Ranger Admin Web UI. Access to a module determines whether the user can see and use options within that module using Ranger Admin Web UI. For example, a user with access to Resource Based Polices module can see Resource Policies option under Service Manager in Ranger Admin Web UI. An admin user can add, import, and export services policies, as well as view, edit and delete permissions for policies, as shown in the following example:

Figure 1: Ranger Admin User View of Resource-based Policies

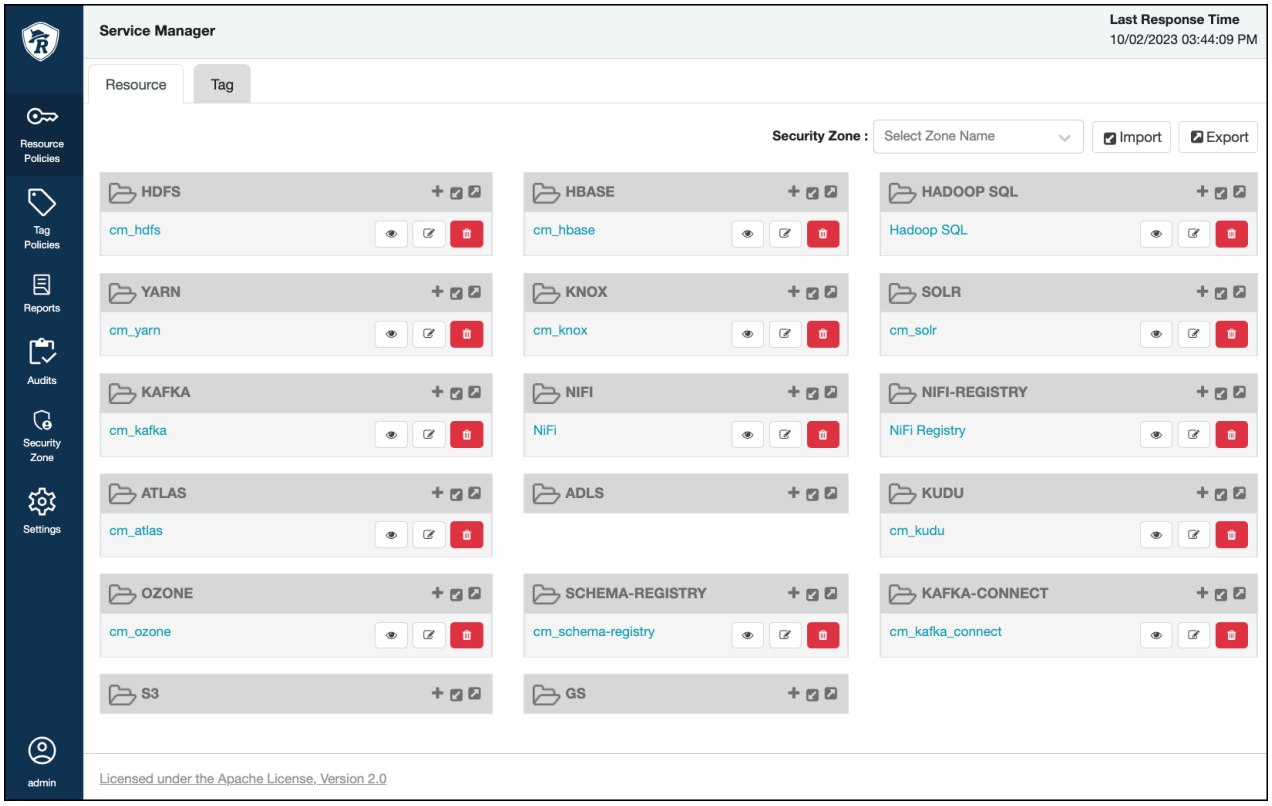

A Ranger user (without admin permissions) can only view the resource-based policies, as shown in the following example:

Figure 2: Ranger User View of Resource-based Policies

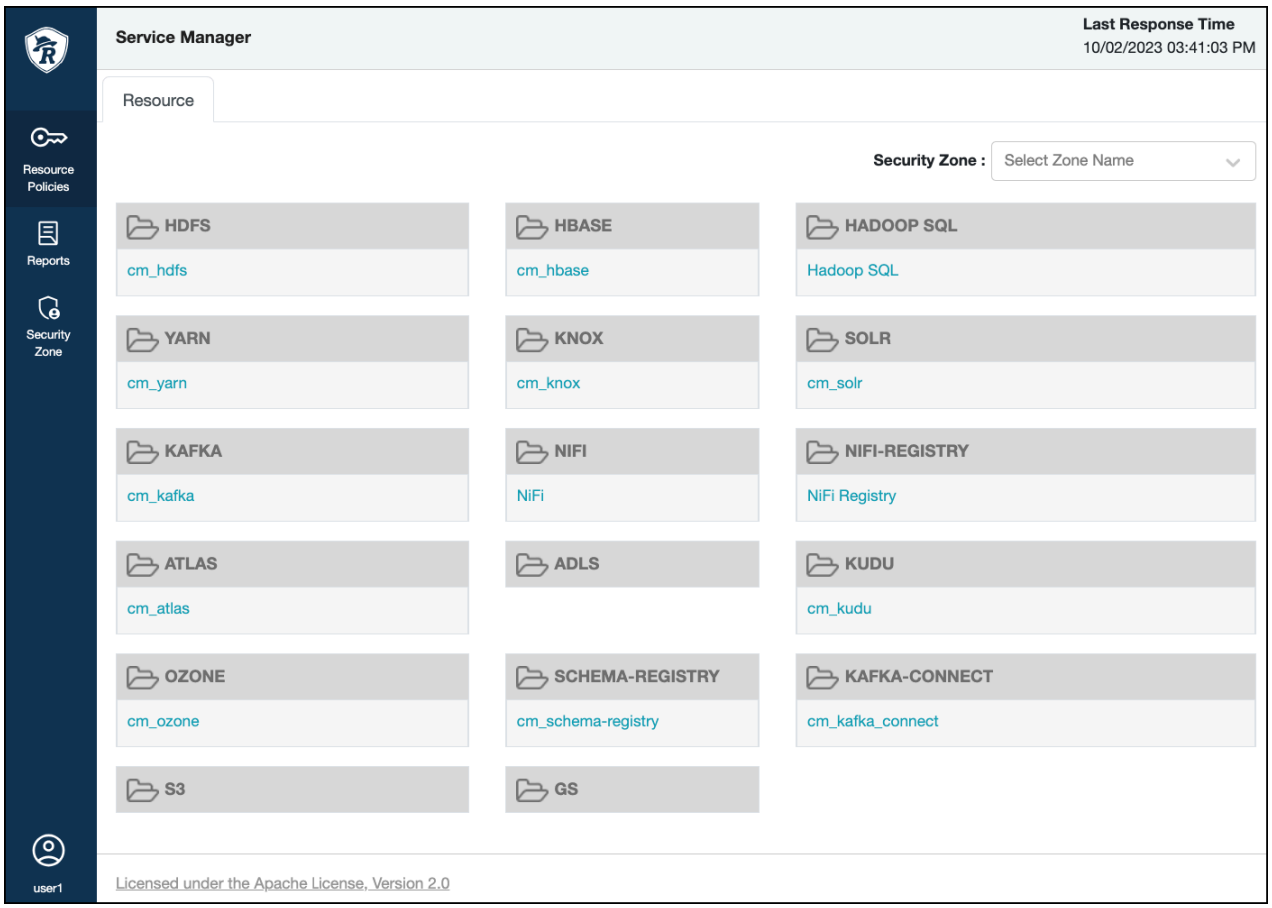

Procedure

1. Select Settings Permissions .
   Permissions displays the Ranger Admin Web UI modules and users that have access permissions to each module.

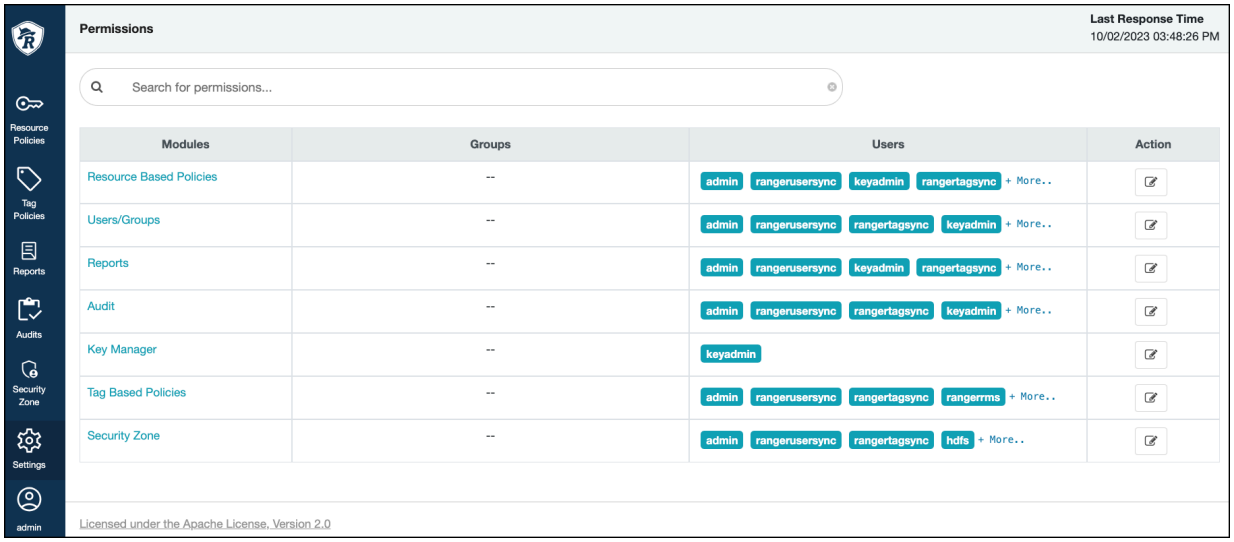

2. Click the Edit icon ( ) for the module you would like to edit.
   Edit Permission page displays options to select and add user and groups to modules.

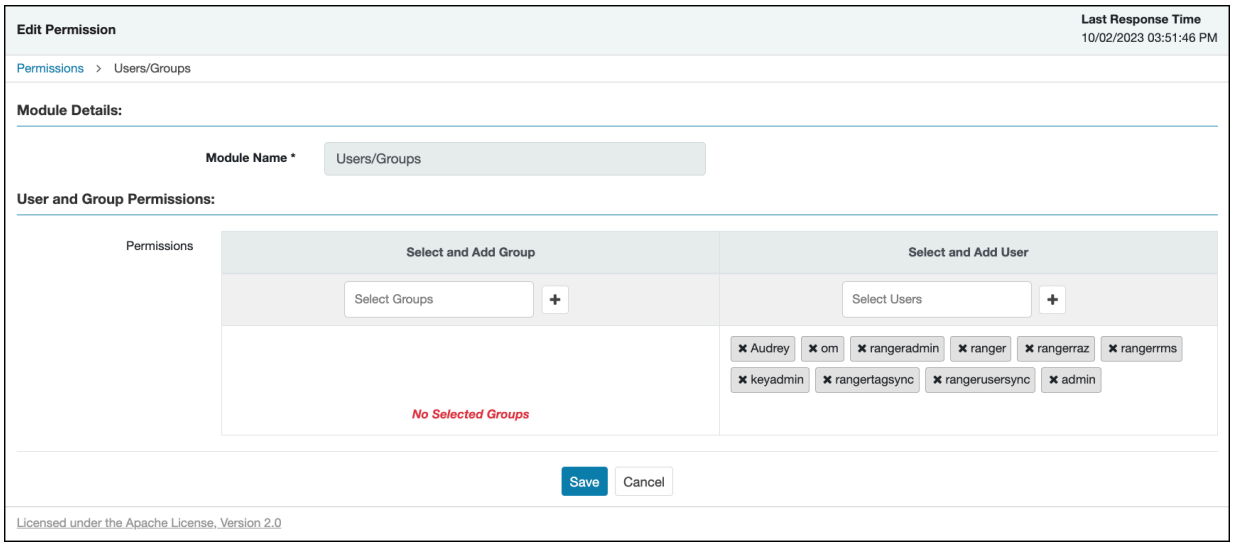

3. Edit the users and groups that get permission to access a module, then click Save.
   You can select multiple users and groups using + .

Ranger Usersync

How to configure Ranger Usersync to sync users and groups from AD/LDAP.

Overview

The Ranger usersync service syncs users, groups, and group memberships from various sources, such as Unix, File, or AD/LDAP into Ranger. Ranger usersync provides a set of rich and flexible configuration properties to sync users, groups, and group memberships from AD/LDAP supporting a wide variety of use cases.

As a Ranger administrator, you will work with users and groups to configure policies in Ranger and administer access to the Ranger UI. You will use group memberships only to administer access to the Ranger UI.

Note: Group memberships stored in Ranger are not used during authorization. Instead, individual components compute the group memberships for a given user on-demand, using utilities like id or group mappings, during authorization. The authority on this is the output of the id or groups command on the Linux OS, which is populated by SSSD from AD (or whichever LDAP provider is used).

For example:

```
# idsp_test1
uid=40002(sp_test1) gid=40006(sp_test1)
groups=40006(sp_test1),40003(cdf_puas),40005(cdf_policy_admins)
# id sp_auditor
uid=40003(sp_auditor) gid=40007(sp_auditor)groups=40007(sp_auditor),40003(cdf_puas)
```

uses id to show that users sp_test 1 and user sp_auditor each belong to three groups, also

```
#groups sp_test1
sp_test1 : sp_test1 cdf_puas cdf_policy_admins
# groups sp_auditor
sp_auditor : sp_auditor cdf_puas
```

uses groups to show the groups that users sp_test1 and sp_auditor belong to.

You must first understand the specific use-case before syncing users, groups, and group memberships from AD/LDAP. For example, if you want to configure only group-level policies, then you must sync groups to Ranger, but syncing users and group memberships to Ranger is not required.

Determining the users and groups to sync to Ranger:

Typically, you must complete a three-step process to define the complete set of users and groups that you will sync to Ranger:

1. Define the customer use-case.

   3 common use cases:

   - A customer Admin or Data Admin wants to configure only group-level policies and restrict access to the Ranger UI to only a few users.
   - A customer's Admin or Data Admin wants to configure only group-level policies and restrict access to the Ranger UI to only members of a group.
   - A customer's Admin or Data Admin wants to configure mostly group-level policies and a few user- level policies.
2. Define all relevant sync source details. For every use-case, at least four key questions must by answered:

   - What groups will sync to Ranger?
   - Which organizational units (OUs) in AD/LDAP contain these groups?
   - What users will sync to Ranger?
   - Which organizational units (OUs) in AD/LDAP contain these users?
3. Configure Usersync properties.

   This topic describes an example set of Usersync configuration properties and values, based on a simple use-case and example AD source repository.

Example Use Case:

First, consider the following use-case, in order to better understand how to configure Usersync properties:

A customers Admin or Data Admin wants to configure only group-level policies and restrict access to the Ranger UI to only members of a group.

Example AD environment:

Configuring Ranger Usersync with AD/LDAP depends highly on the customer environment. You must understand the organization of users and groups in the customer environment. This illustration shows users and groups organized in an Active Directory environment.

Figure 3: Example Active Directory Structure

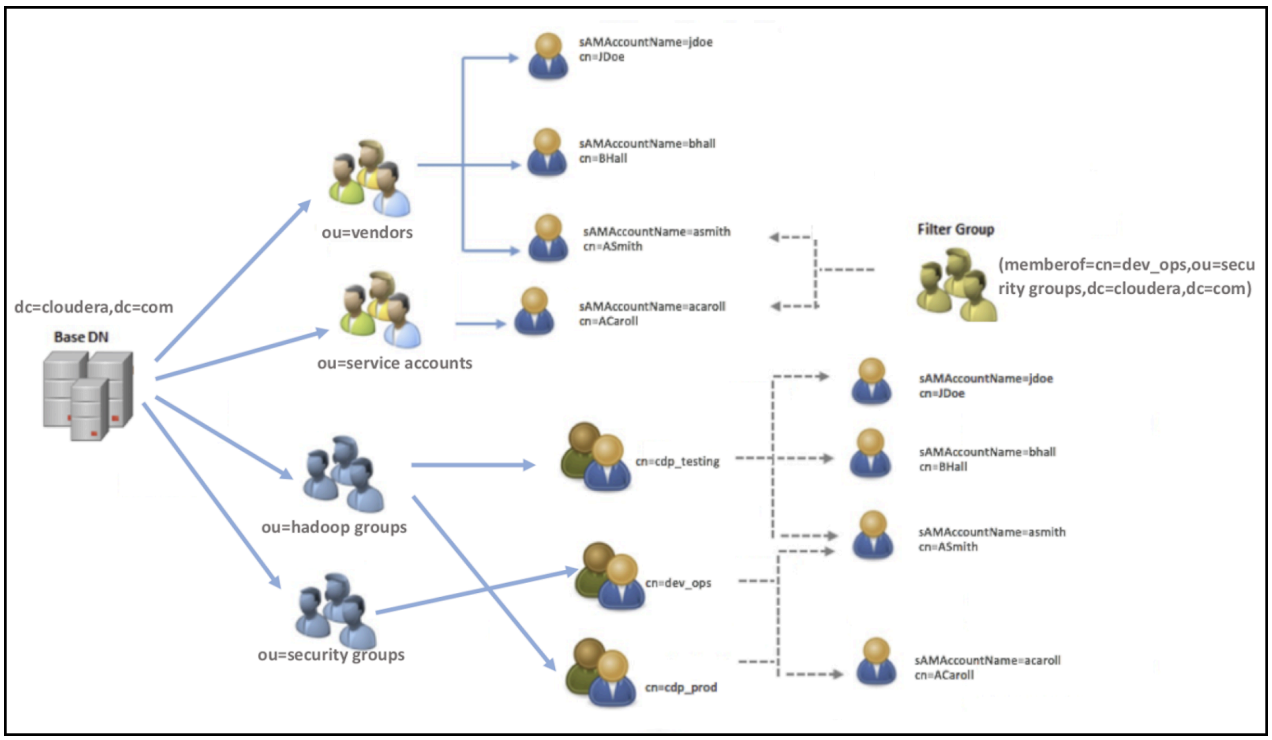

Answering the key user and group questions, based on the example AD structure:

In this example, the customer wants to configure group-level policies for groups cdp_testing and cdp_prod and wants to provide admin access to the Ranger UI only for users in the dev_ops group.

Based on the example Active Directory structure, answers to the four key user/group questions are:

Q1: What groups will be synced to Ranger?

A1: cdp_testing, cdp_prod, and dev_ops

Q2: What OUs contain these groups in AD?

A2: hadoop groups and security groups

Q3: What users will be synced to Ranger?

A3: asmith and acaroll (these users are dev_ops group members)

Q4: What OUs contain these users in AD?

A4: vendors and service accounts

To find the specific answers to these questions in a particular environment, use a tool such as Ldapsearch, as shown in the following examples.

- Example: Ldapsearch command to search a particular group cdp_testing and determine what attributes are available for the group.

  Figure 4: Using Ldapsearch to find a specific group

```
ldapsearch -x -LLL -h 10.10.10.10:389 -D 'cn=administrator,CN=Users,dc=cloudera,dc=com' -W
-b 'ou=Hadoop Groups,dc=cloudera,dc=com' 'cn=cdp_testing'
Enter LDAP Password:
dn: CN=cdp_testing,ou=Hadoop Groups,dc=cloudera,dc=com
objectClass: top
objectClass: group
cn: cdp_testing
member: CN=ASmith,ou=Hadoop Users,dc=cloudera,dc=com
member: CN=BHall,ou=Hadoop Users,dc=cloudera,dc=com
member: CN=JDoe,ou=Hadoop Users,dc=cloudera,dc=com
distinguishedName: CN=cdp_testing,ou=Hadoop Groups,dc=cloudera,dc=com
instanceType: 4
name: cdp_testing
sAMAccountName: cdp_testing
```

  Above output shows all the available attributes for cn=cdp_testing. The highlighted attributes are those of interest for usersync configuration. In this case, cdp_testing has three “member” attributes: ASmith, BHall, and JDoe.

- Example: Ldapsearch command to search a particular user ASmith and determine what attributes are available for the user.

  Figure 5: Using Ldapsearch to find a specific user

```
ldapsearch -x -LLL -h 10.10.10.10:389 -D 'cn=administrator,CN=Users,dc=cloudera,dc=com'
-W -b 'ou=Hadoop Users,dc=cloudera,dc=com' 'samaccountname=ASmith
Enter LDAP Password:
dn: CN=ASmith,ou=Hadoop Users,dc=cloudera,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: ASmith
sn: Smith
givenName: Andy
distinguishedName: CN=ASmith,ou=Hadoop Users,dc=cloudera,dc=com
instanceType: 4
memberOf: CN=cdp_testing,ou=Hadoop Groups,dc=cloudera,dc=com
memberOf: CN=dev_ops,ou=Hadoop Groups,dc=cloudera,dc=com
memberOf: CN=cdp_prod,ou=Hadoop Groups,dc=cloudera,dc=com
primaryGroupID: 513
logonCount: 0
sAMAccountName: ASmith
```

  Above output shows all the available attributes for a user. The highlighted attributes are those of interest for usersync configuration. In this case, ASmith is a “memberof” 3 groups - cdp_testing, dev_ops, and cdp_prod.

How to configure Usersync, based on the illustrated AD environment example:

In Cloudera Manager Ranger Configuration select the Ranger Usersync filter scope.

Figure 6: Filtering the Ranger Configuration Properties for Usersync

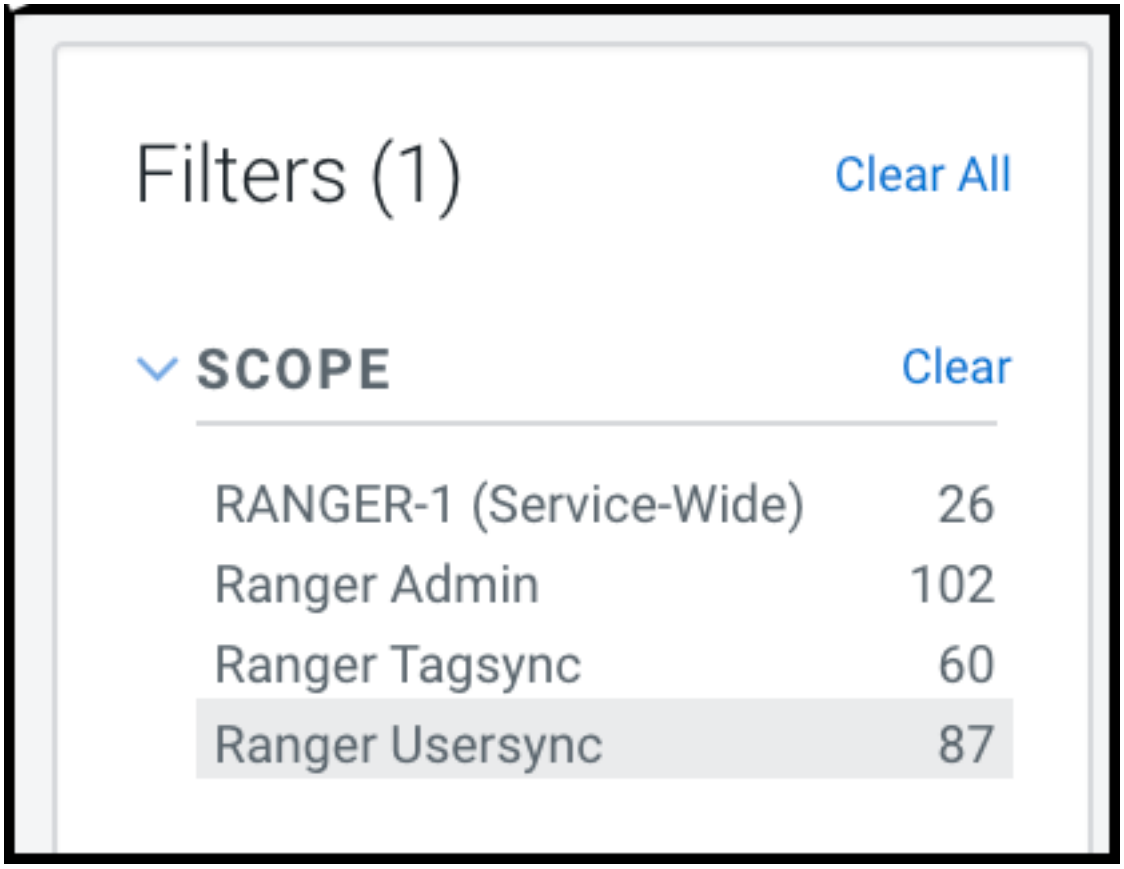

Filtering narrows the list to 87 configuration properties specific to Usersync.

1. To define the common configuration properties that control LDAP URL and bind credentials, scroll to Source for Syncing Users and Groups, then define the configurations properties appropriate for the environment. Configurations shown here match the Example AD environment.

Figure 7: Ranger Usersync common configuration settings

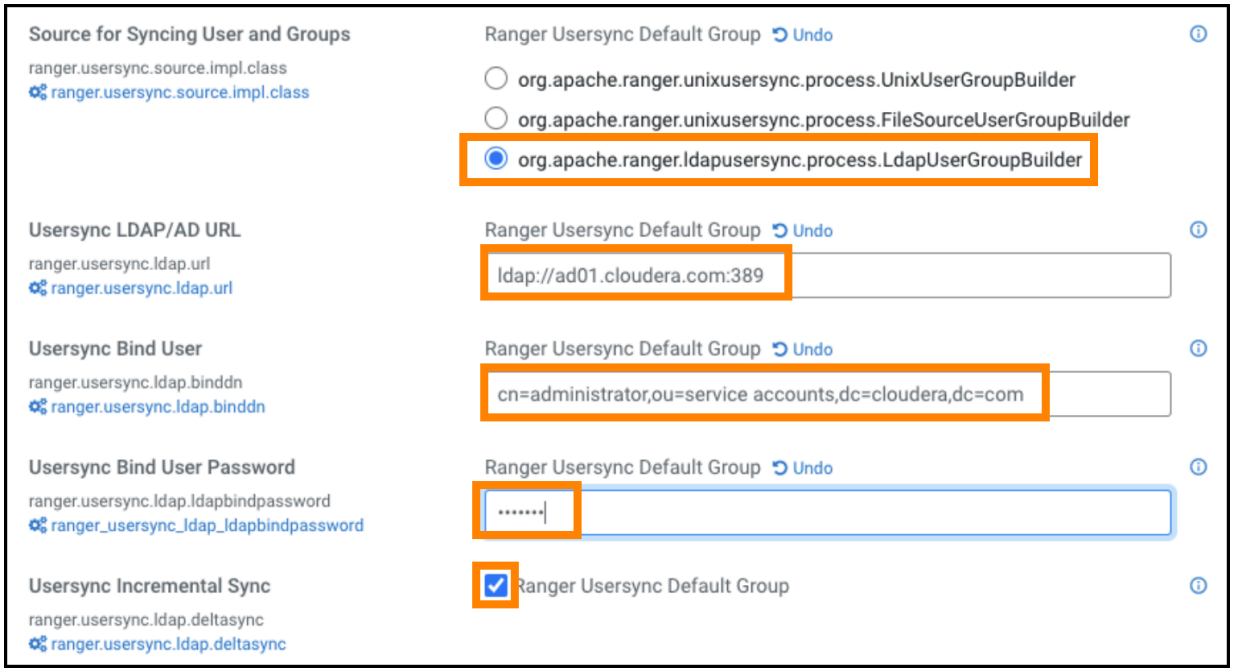

Bind credentials are for the user to query Ldap service for users and groups. Bind credentials contain two configuration properties:

- Usersync Bind User (or bind dn) - specify the username as complete DN (Distinguished Name)
- Usersync Bind User Password

2. To define the required configuration properties that control group synchronization from AD, scroll to Usersync Enable User Search, then define the configurations properties appropriate for the environment. Configurations shown here match the Example AD environment.

Figure 8: Ranger Usersync group configuration settings

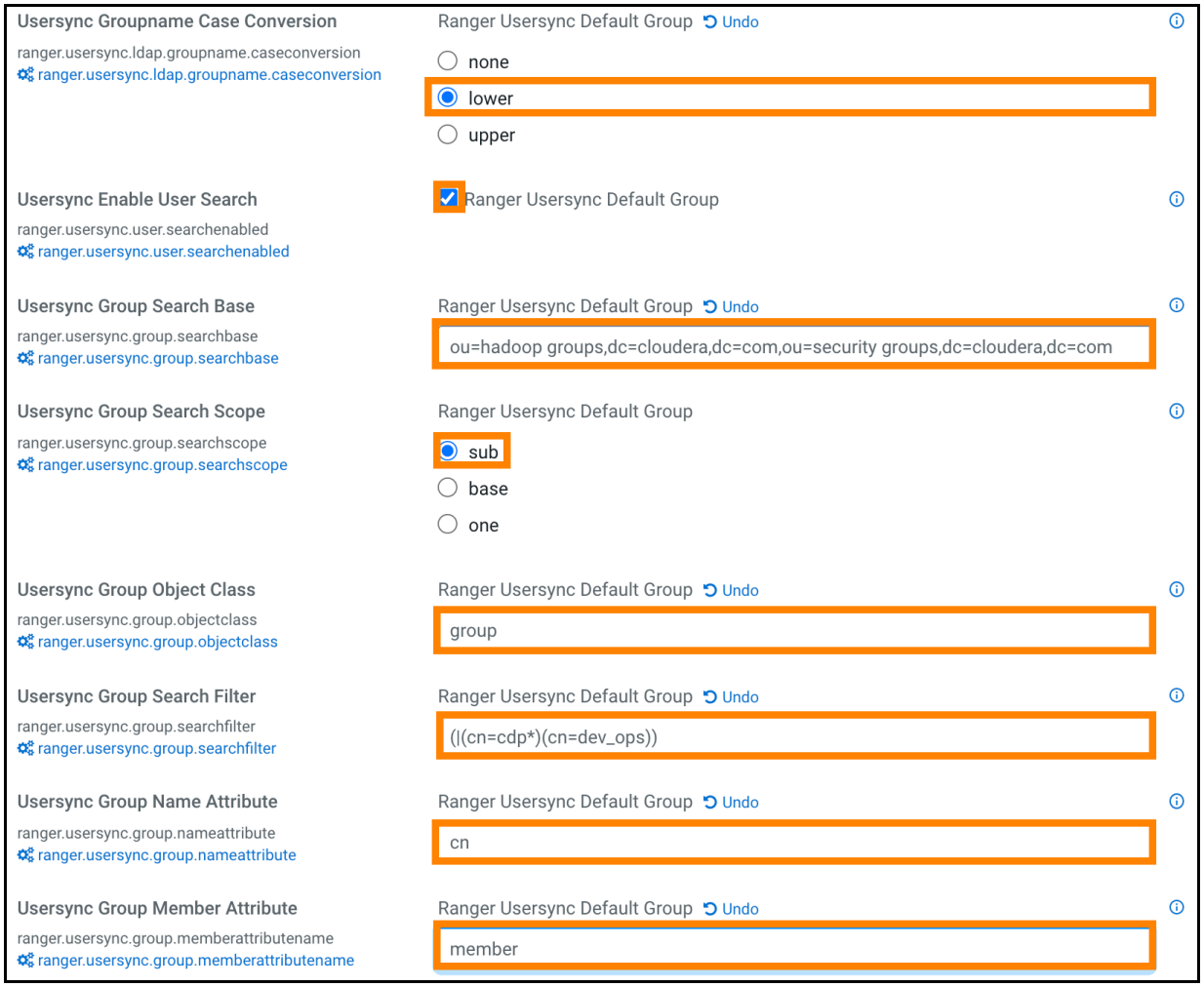

A few specific points to consider about group config settings:

- ranger.usersync.ldap.groupname.caseconversion - Used for converting the case of the groupname. Three possible options are:
  - None - Group names are synced to ranger as is from AD/LDAP. This is the default setting.
  - Lower - All the group names are converted to lowercase while syncing to ranger. This is the recommended setting.
  - Upper - All the group names are converted to uppercase while syncing to ranger

  Note: Policy authorization is case sensitive. Therefore, usernames and groups names synced to ranger must match the exact case of the users and groups resolved by the services such as hdfs, hive, hbase, etc. For example, consider dev_ops (all in lower case). Ranger does not treat this as the same value as Dev_Ops which may have been synced from AD and applied to some policies.

  ranger.usersync.group.searchbase - Used to search a particular OU in AD for groups. Multiple OUs can be specified with ; separated. For example, the example AD shows two OUs that must be searched for groups:

  - ou=hadoop groups,dc=cloudera,dc=com (complete DN for ou=hadoop groups)
  - ou=security groups,dc=cloudera,dc=com (complete DN for ou=security groups)

- ranger.usersync.group.searchfilter - In this example, since only 3 groups exist in hadoop groups OU and security groups OU and since all 3 require sync to Ranger, you can specify the filter as cn=* . The value for this property follows standard ldap search query filter format.

  Note: Later, if a new group is added in AD under these OUs and if the customer wants those groups to be sync'd to ranger, no configuration change to usersync is required.

- ranger.usersync.user.searchenabled - In this example, since the customer wants to sync users from dev_ops groups to provide admin access to Ranger UI, this property is set to true .

3. To define the required configuration properties that control user synchronization from AD, scroll to Usersync User Search Base, then define the configurations properties appropriate for the environment. Configurations shown here match the Example AD environment.

Figure 9: Ranger Usersync user configuration settings

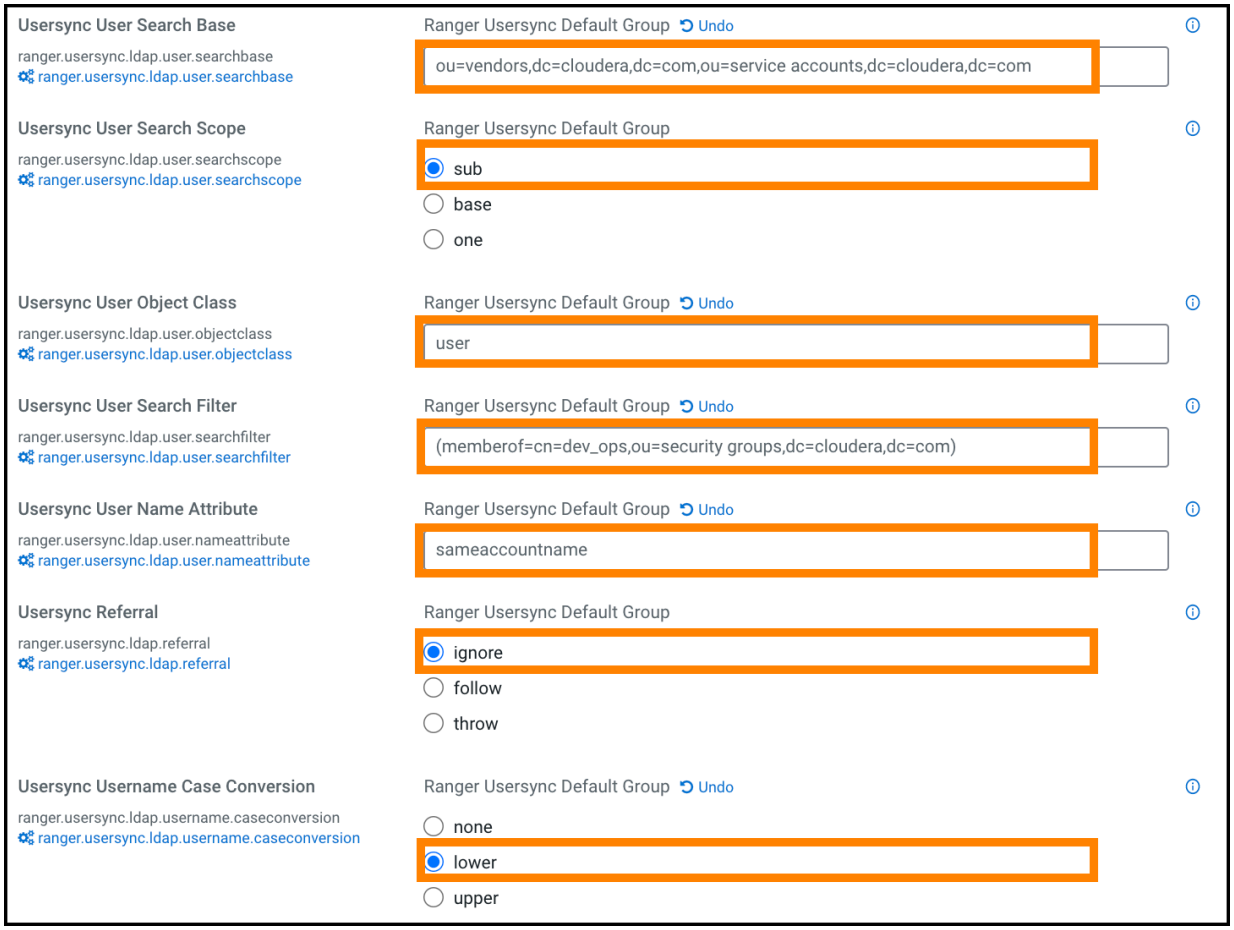

A few specific points to consider about user config settings:

- ranger.usersync.ldap.user.searchbase - This configuration is used to search a particular location in AD for users. Specify multiple OUs with ; separated.

  Note: If users are distributed across several OUs, specifying a base directory, for example, dc=cloudera,dc=com might be convenient and is highly recommended to restrict the search with proper filters.

- ranger.usersync.ldap.user.searchfilter - In this example, since the customer wants to sync only the users that belong to dev_ops, the value for this property is (memberof=cn=dev_ops,ou=security groups,dc=cloudera,dc=com) .

  Note: Wildcards are not supported only when the memberof attribute is used for searching. If you use attributes such as cn or samaccountname for filtering, you can specify wildcards. For example, (| (cn=asm*)(samaccountname=acar*))

- ranger.usersync.ldap.username.caseconversion - Used for converting the case of the username. Three possible options are:
  - None - Usernames are synced to ranger as is from AD/LDAP. This is the default setting.
  - Lower - All the usernames are converted to lowercase while syncing to ranger. This is the recommended setting.
  - Upper - All the usernames are converted to uppercase while syncing to ranger

Note: Policy authorization is case sensitive. Therefore, usernames and groups names synced to ranger must match the exact case of the users and groups resolved by the services such as hdfs, hive, hbase, etc. For example, consider asmith (all in lower case). Ranger does not treat this as the same value as ASmith which may have been synced from AD and applied to some policies.

Adding default service users and roles for Ranger

Cloudera Manager adds a property and default values that define roles for the minimum set of service users by default.

Name

ranger.usersync.whitelist.users.role.assignment.rules

Default Value

&ROLE_SYS_ADMIN:u:admin,rangerusersync,rangertagsync,ranger,rangeradmin,rangerraz,rangerrms&ROLE_KEY_

Go to Cloudera Manager Ranger Configuration , then type whitelist in Search to see the property and assigned values. Ranger Usersync creates roles for each service user during syncronization.

ranger.usersync.whitelist.users.role.assignment.rules uses same format as ranger.usersync.group.based.role.assignment.rules.

If you add any custom principals, you must update the list of values for ranger.usersync.whitelist.users.role.assignment.rules accordingly so that Ranger usersync applies role assignments rules appropriately. Any change to these configuration values requires a restart of Ranger usersync. Ranger usersync applies these rules during restart and every sync cycle, if changed.

If the same service user exists in:

- ranger.usersync.whitelist.users.role.assignment.rules, and
- ranger.usersync.group.based.role.assignment.rules

with different role assignments, then the role assignment from ranger.usersync.whitelist.users.role.assignment.rules takes priority. This is true even if ranger.usersync.group.based.role.assignment.rules has role assignment rules for a group that has service users as members. Any changes to the role assignments made to these service users from Ranger UI or rest API are temporary. The next Ranger usersync sync cycle resets them.

Configuring Usersync assignment of Admin users

How to automatically assign Admin and Key Admin roles for external users

About this task

Ranger provides configuration for defining roles for external users.

Usersync pulls in users/groups from your external user repository, such as LDAP/AD, and populates the Ranger database with these users/groups. Use this procedure to automatically assign roles to specific users/groups. The example properties shown in this topic automatically assign the ADMIN/KEYADMIN role to external users.

Currently, Ranger supports various roles (or privileges) to be assigned to a user:

ROLE_SYS_ADMIN

Has permission to create users, group, roles, services, and policies, run reports, and perform other administrative tasks. Admin users can also create child policies based on the original policy.

ROLE_KEY_ADMIN

Has permission to manage (create, update, or delete) access policies and keys for Ranger KMS.

ROLE_USER

Has least privilege (and default role) assigned to a user. All users are assigned this default role.

ROLE_ADMIN_AUDITOR

An Admin user with read-only permission.

ROLE_KEY_ADMIN_AUDITOR

An Admin user with read-only permission for Ranger KMS.

Auditor and KMS Auditor roles have been introduced in Ranger Admin. Users with these roles have read-only access to all the services, policies, user/groups,audits and reports.

- The Auditor role allows a user with Auditor role to view all information that a user with Admin role can see. A user with Auditor role will have a read-only view of a user with Admin role. In other words, a user with Auditor role user will be blocked from the create/update/delete/import/exportJson of all API in the Ranger UI and curl command.
- The KMS Auditor role allows a user with KMS Auditor role to view all information that a user with Keyadmin role can see on the Ranger UI. A user with KMS Auditor role will have a read-only view of a user with Keyadmin role. In other words, a user with KMS Auditor role will be blocked from create/update/delete/import/exportJson of all API in the Ranger UI and curl command.
- Users with the Auditor or KMSAuditor role, even if delegated as admin in any policies of any services, will be restricted from create/update/delete/import/exportJson. In other words, users with Auditor or KMS Auditor role have view-only access based on their role.
- A user with KMS Auditor role cannot get keys, even if that user is added in policy.
- Users with Auditor or KMS Auditor role can change their password.
- No user has Auditor or KMS Auditor role by default.
- Users with Auditor or KMS Auditor role can export policies to excel and csv file formats.

A user can have only one role, and that role is determined by the last role assigned, depending in part on group membership.

For example, if the role assignment rules are configured as follows:

ROLE_SYS_ADMIN:u:User1, User2&ROLE_SYS_ADMIN:g:Group1, Group2&ROLE_AUDITOR:g:Group3, Group4&ROLE_USER:g:Group5

and if a user belongs to Group1 & Group5, then the role assigned to that user is ROLE_USER.

Similarly, if a user belongs to Group2 & Group3, then the role assigned to that user is ROLE_AUDITOR.

If the user does not belong to any of these groups (Group1, Group2, Group3, Group4, or Group5), then the default role assigned to the user is ROLE_USER.

If the user belongs to only Group1, then the role assigned to the user is ROLE_SYS_ADMIN.

To automatically assign the ADMIN/KEYADMIN role to external users:

Procedure

1. In Ranger Configuration Search , type role.assignment.
2. In Ranger Usersync Default Group: verify that the following default delimiter values appear for each property:

Property Name	Delimiter Value
ranger.usersync.role.assignment.list.delimiter	&
ranger.usersync.users.groups.assignment.list.delimiter	:
ranger.usersync.username.groupname.assignment.list.delimiter	,
ranger.usersync.group.based.role.assignment.rules	

3. In Ranger UserSync Group Based Role Assignment Rules, type the following value as one string:

   ROLE_SYS_ADMIN:u:User1,User2&ROLE_SYS_ADMIN:g:Group1,Group2&

   ROLE_KEY_ADMIN:u:kmsUser&ROLE_KEY_ADMIN:g:kmsGroup&

   ROLE_USER:u:User3,User4&ROLE_USER:g:Group3,Group4&

   ROLE_ADMIN_AUDITOR:u:auditorUsers,auditors&
   ROLE_ADMIN_AUDITOR:g:adminAuditorGroup,rangerAuditors&

   ROLE_KEY_ADMIN_AUDITOR:u:kmsAuditors&ROLE_KEY_ADMIN_AUDITOR:g:kmsAuditorGroup

   where "u" indicates user and "g" indicates group
4. Click Save Changes (CTRL+S).
5. If Usersync requires no other changes, choose Actions Restart Usersync .

Configuring nested group hierarchies

You must enable non-zero group hierarchy levels in Ranger before adding a group as a member to another group.

About this task

"Nesting" groups means adding a group as a member to another group. If a group (groupA) is a member of another group (groupB), then the users belonging to the member group (groupA) are part of the parent group (groupB) as well. Nesting can be very useful when delegating access through inheritance. Many large enterprises have their groups in LDAP/AD nested within other groups. Security admins want the users in those nested groups to be associated in Ranger so that they are available for policy authoring in Ranger Admin. Cloudera Ranger Usersync supports nested group membership representation for policy authoring.

Note: In order to utilize nested group mapping, a Ranger Admin user must set the Usersync Group Hierarchy Levels to a non-zero value.

In the following example directory structure, all the marketing users are under one OU "Marketing Users". All these users are members of different groups based on the location like US, Canada, London, etc… For example, user "Adam Will" from "Marketing Users" OU is a member of "Canada Marketing Group". The example directory structure also contains multiple, nested group levels; for example, "US Marketing Group" is a member of "AMER Marketing Group" which again is a member of "Marketing Group".

Figure 10: Example Active Directory structure with Nested Groups

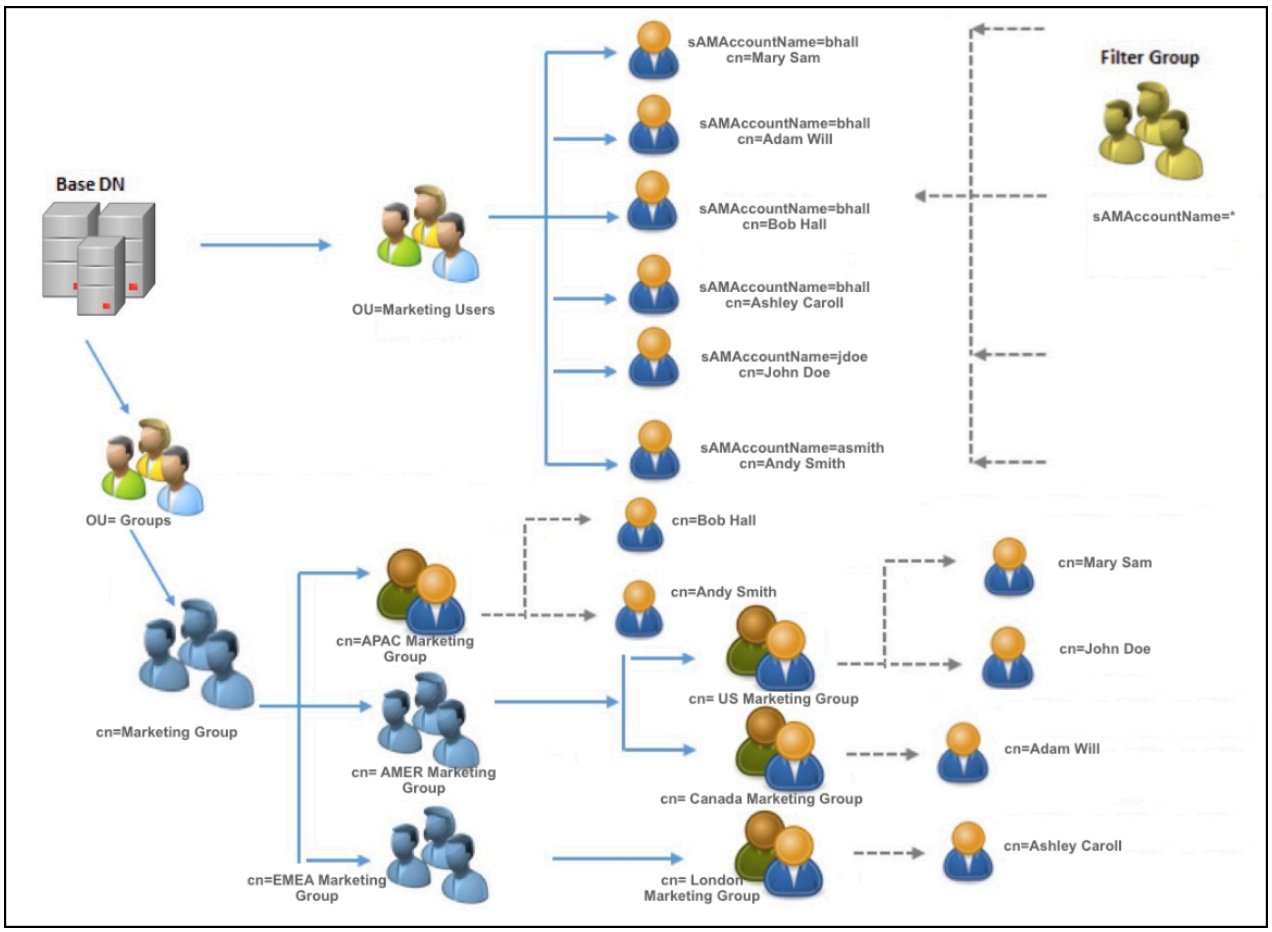

Ranger Usersync, by default, computes only the immediate groups for the users. For example, user "Adam Will" is part of "Canada Marketing Group" and only this information is available in ranger without nested group sync configuration. With this information, if an admin wants to provide access to all the users under "AMER Marketing Group", then all the sub groups - "US Marketing Group" and "Canada Marketing Group" must be added to the policy, using Ranger.

In order to simplify the policy configuration at parent-level groups, Ranger supports evaluating nested group memberships by configuring "ranger.usersync.ldap.grouphierarchylevels". If ranger.usersync.ldap.grouphierarchylevels is set to "3", Ranger Usersync computes the group memberships for user "Adam Will" as "Canada Marketing Group", "AMER Marketing Group", "Marketing Group". This way, admin can configure ranger policy at the parent group level ("AMER Marketing Group") which will be applied for all the users (Mary Sam, John Doe, and Adam Will) under each sub group (US Marketing Group and Canada Marketing Group).

Procedure

1. In Cloudera Manager Ranger Configuration, type hierarchy in Search.
2. In Usersync Group Hierarchy Levels (ranger.usersync.ldap.grouphierarchylevels) configuration, set:
   - In Ranger Usersync Default Group, type: 3
3. Click Save Changes.
4. Restart Ranger.

Note: ranger.usersync.ldap.grouphierarchylevels property is set to 0 by default.

Configuring Ranger Usersync for Deleted Users and Groups

How to configure Ranger Usersync for users and groups that have been deleted from the sync source.

About this task

You can configure Ranger Usersync to update Ranger when users and groups have been deleted from the sync source (UNIX, LDAP, AD or PAM). This ensures that users and groups – and their associated access permissions – do not remain in Ranger when they are deleted from sync source.

Procedure

1. In Cloudera Manager, select Ranger > Configuration, then use the Search box to search for Ranger Usersync Advanced Configuration Snippet (Safety Valve) for conf/ranger-ugsync-site.xml. Use the Add (+) icons to add the following properties, then click Save Changes.

Name	Value	Description
ranger.usersync.deletes.enabled	true	Enables deleted users and groups synchronization. The default setting is false (disabled).
ranger.usersync.deletes.frequency	10	Sets the frequency of delete synchronization. The default setting is 10, or once every 10 Usersync cycles. Delete synchronization consumes cluster resources, so a lower (more frequent) setting may affect performance.

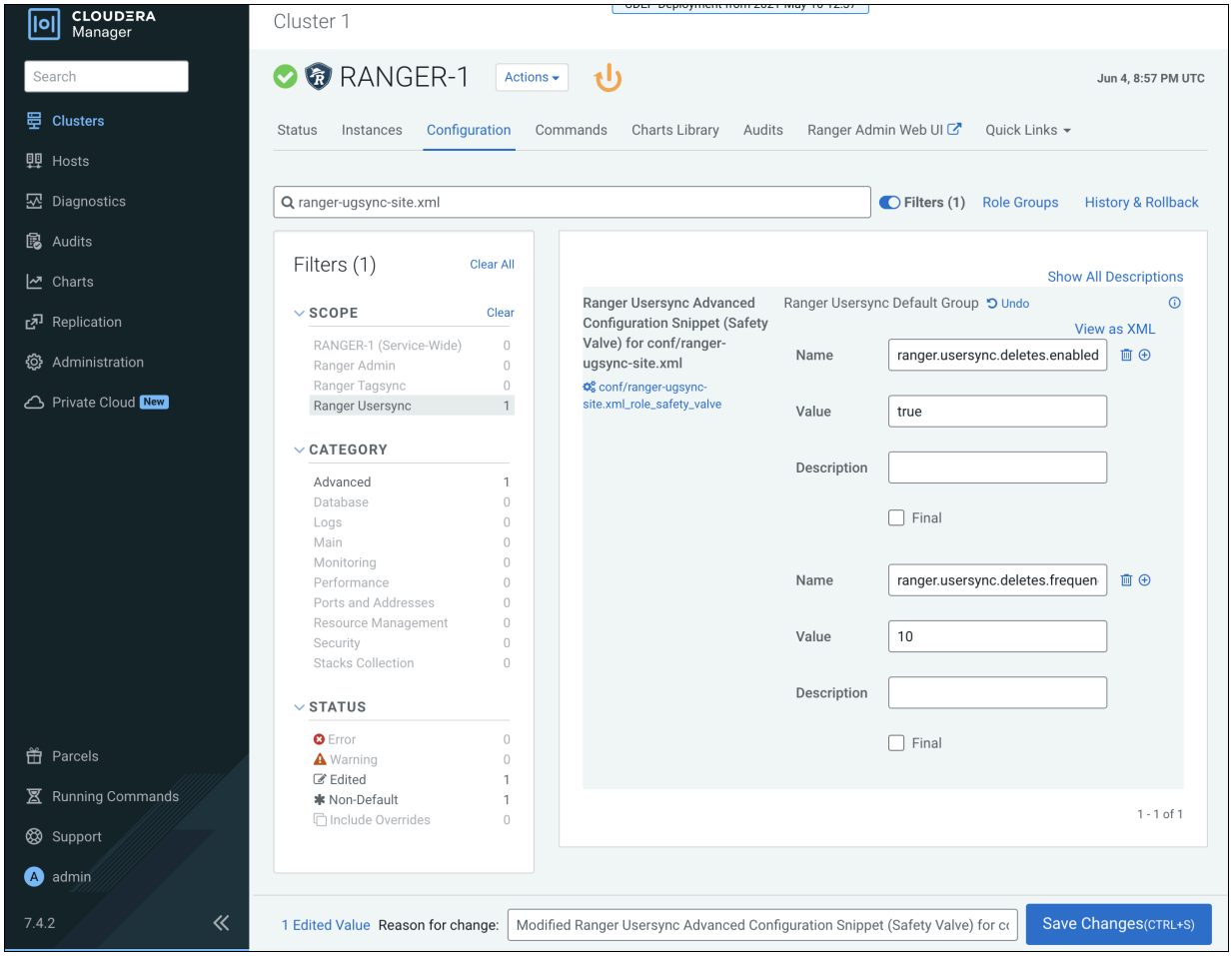

2. Click the Ranger Restart icon.

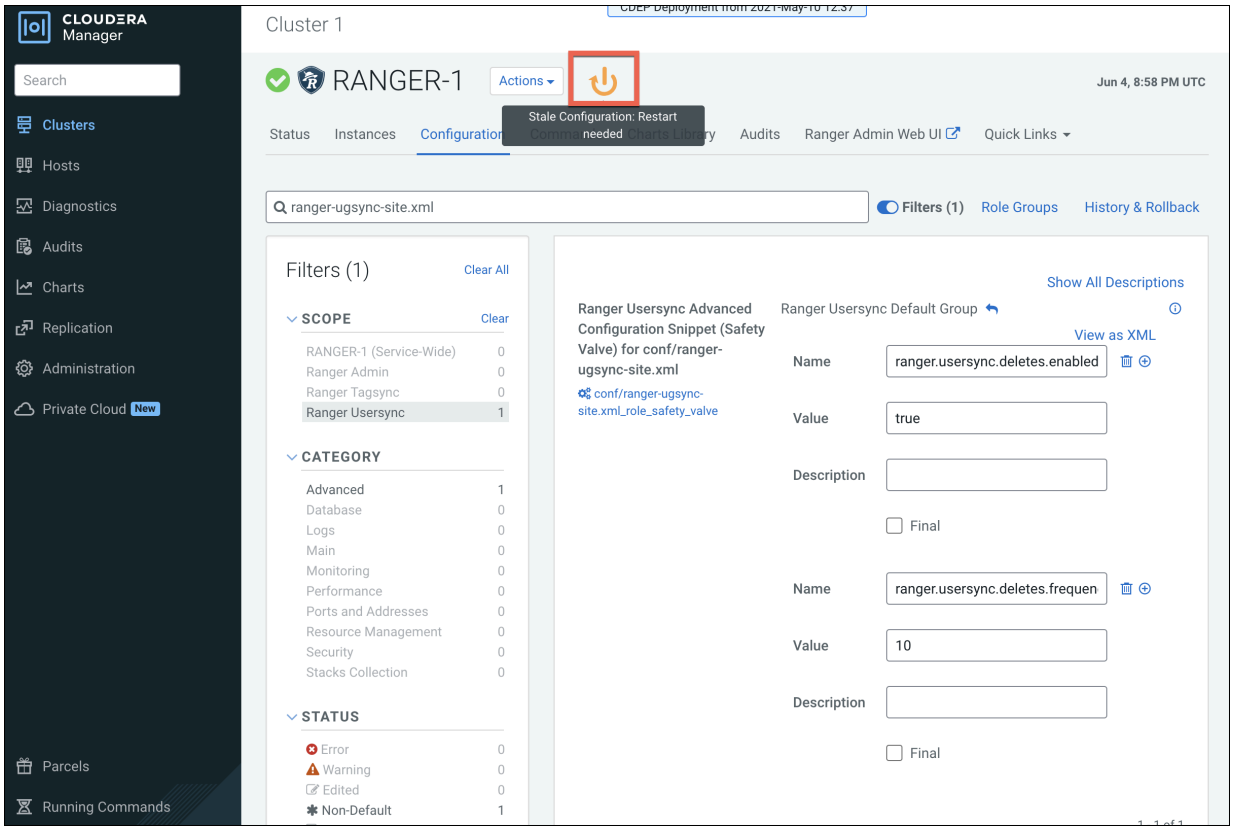

3. On the Stale Configurations page, click Restart Stale Services.

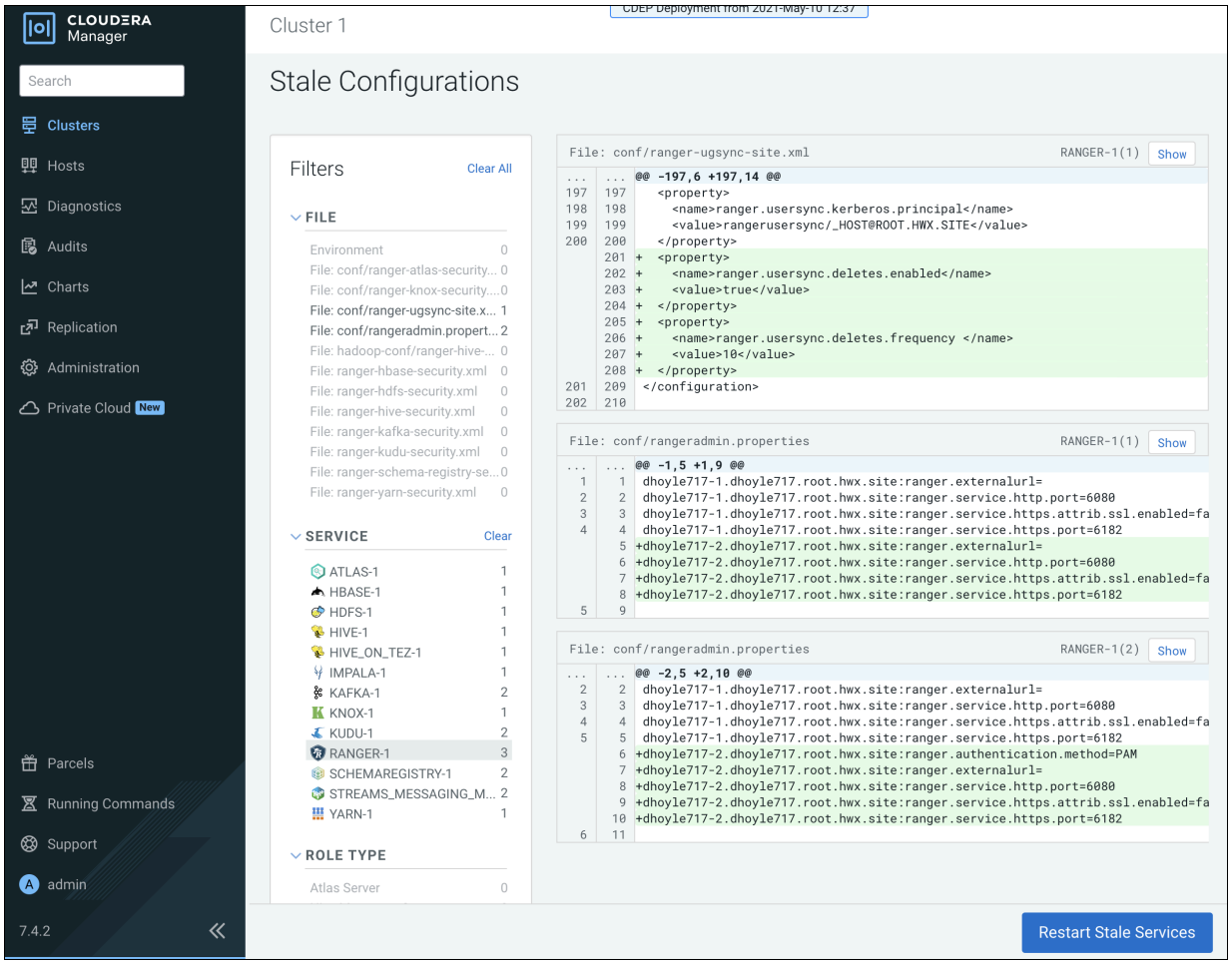

4. On the Restart Stale Services page, select the Re-deploy client configuration check box, then click Restart Now.

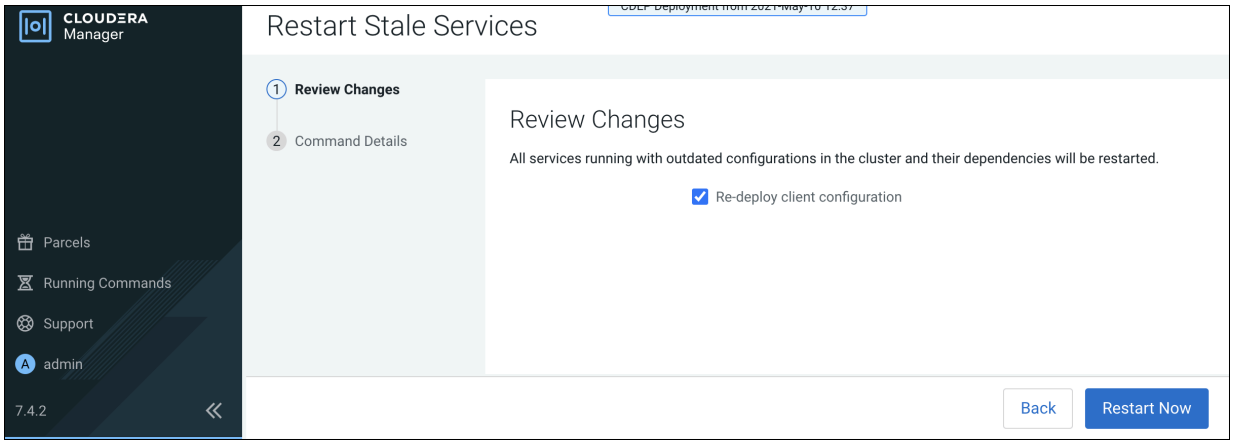

5. A progress indicator page appears while the services are being restarted. When the services have restarted, click Continue.

6. Users that have been deleted in sync source are not automatically deleted in Ranger – they are marked as Hidden and must be manually deleted by the Ranger Admin user, and then Ranger Usersync must be restarted.

   In the Ranger Admin Web UI, select Settings > Users/Groups/Roles. Click in the User List text box, then select Visibility > Hidden.

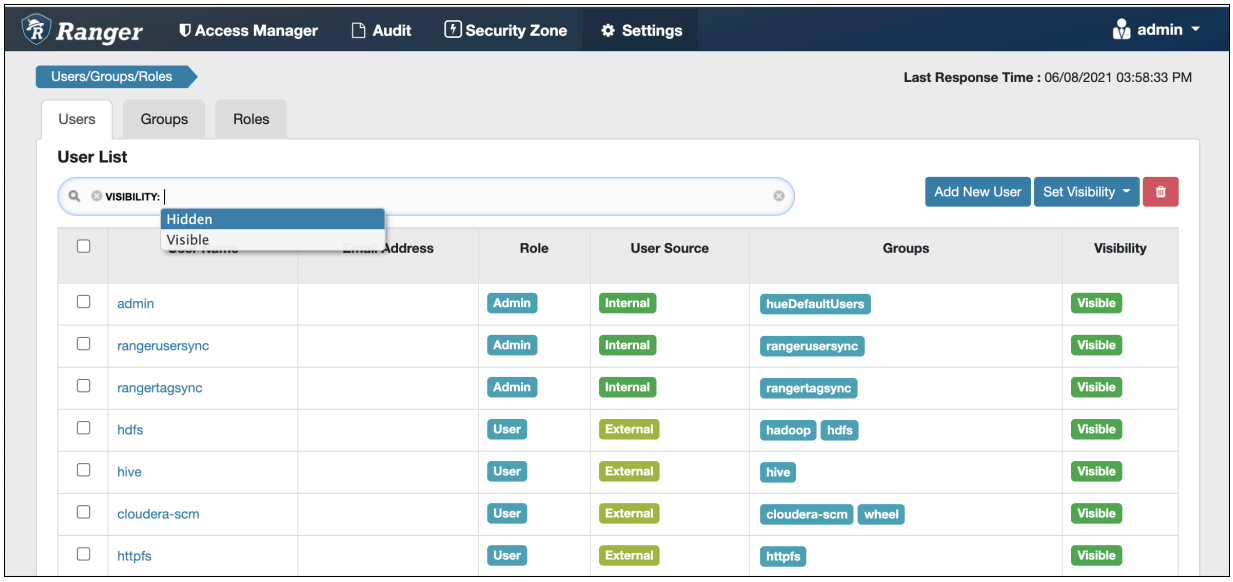

7. To delete a hidden user or group manually, select the applicable check boxes, then click the red Delete icon, as shown in the following example.

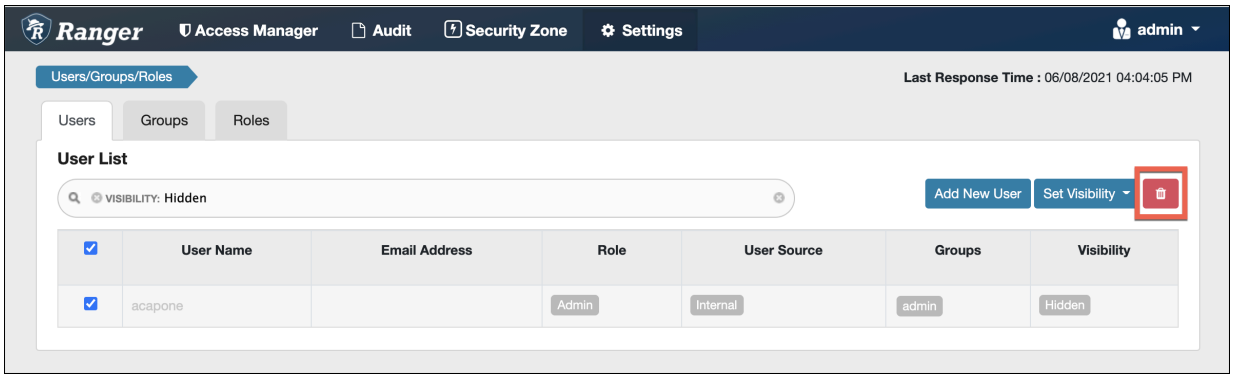

You can delete multiple users or groups by running a "delete" script on the command line interface.

For example:

```
Sample command to delete users:
python deleteUserGroupUtil.py -users <user file path> -admin <ranger admin
 user> -url <rangerhosturl> [-force] [-sslCertPath <cert path>] [-debug]

Sample command to delete groups:
python deleteUserGroupUtil.py -groups <group file path> -admin <ranger
admin user> -url <rangerhosturl> [-force] [-sslCertPath <cert path>] [-d
ebug]
```

Note: The deleteUserGroupUtil.py script installs as part of the Ranger installation on the node where Ranger Admin runs, in the following location: /opt/cloudera/parcels/CDH/lib/ranger-admin/ .

8. In Cloudera Manager, select Ranger > Ranger Usersync, then select Actions > Restart this Ranger Usersync.

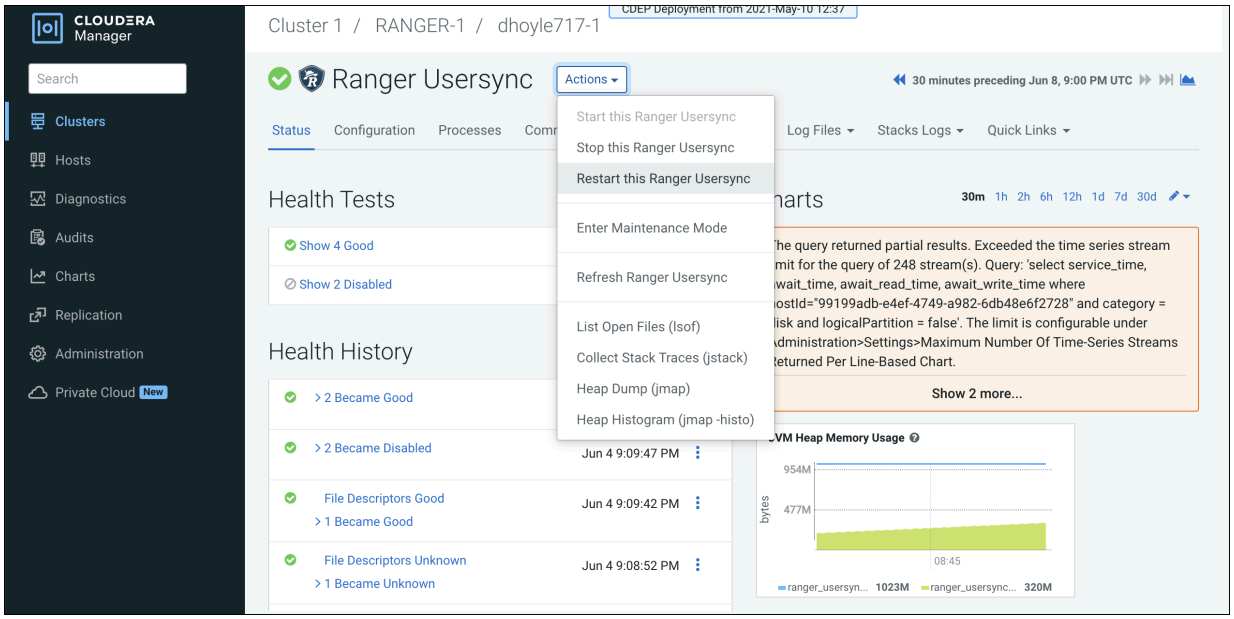

Note:

- Sync source is tracked when processing Ranger users and groups for deletion. If the same user name for a separate sync source already exists in Ranger DB, that user will not be updated or marked as hidden.
- For AD/LDAP sync:
  - After marking a user or group as deleted/hidden in Ranger, the user or group status does not change automatically. The user or group must be manually deleted (or deleted using the cli "delete" script). Usersync must be restarted to reflect any changes to the same user name in the source.
  - For example, a user (Bob) from one OU (say Engineering) is deleted from the source and is marked as deleted in Ranger admin. If the same user name (Bob) is subsequently added back to the same OU, the user status will not be automatically enabled. The user must be manually deleted and Usersync must be restarted to implement the changes.
  - If an identical user name (say Bob) is deleted from one OU (say Engineering) and added to a different OU (say Finance) between the sync cycles, user Bob is marked as hidden/deleted only when the delete cycle is triggered. Until then there is a security risk that user Bob from Finance will be granted the permissions for Bob from Engineering.

Configuring Ranger Usersync for invalid usernames

How to configure Ranger Usersync to manage usernames containing invalid characters.

About this task

Ranger Usersync pulls in users/groups from your external user repository, such as LDAP/AD, and populates the Ranger database with these users/groups.

An invalid username contains at least one invalid character. Ranger fails to create a set of users if an invalid username exists within that set of users. Usersync perpetually tries to recreate this user set without creating Ranger or Cloudera Manager alerts. This error appears in both Usersync and admin logs, but the log output lacks necessary details such as the invalid username. By adding the following configuration, you cause Usersync to recognize invalid characters in a user/group name and then skip synchronization for any names that contain invalid characters.

Procedure

1. In Cloudera Manager Ranger Configuration, type Ranger Usersync Advanced Configuration Snippet (Safety Valve) in Search.
2. In the Ranger Usersync Advanced Configuration Snippet (Safety Valve) for conf/ranger-ugsync-site.xml configuration, select + to include the following property:
   - In Name, type: ranger.usersync.name.validation.enabled
   - In Value, type: true
3. Click Save Changes.
4. Restart Ranger.

Note: This configuration property is set to false by default.

Results

Ranger Usersync now successfully synchronizes all valid usernames from the external user repository and skips any usernames containing invalid characters.

Setting credentials for Ranger Usersync custom keystore

How to set the custom keystore file location and password for a Ranger Usersync custom keystore.

About this task

Ranger Usersync role creates a default keystore file, ranger.usersync.keystore.file during restart. UNIX authentication in Ranger Admin requires this keystore file. The keystore file takes a password from the ranger.usersync.keystore.password configuration, exposed in Cloudera Manager supporting Cloudera 7.1.6 and higher.

Setting custom keystore credentials for Ranger Usersync overrides the default credentials.

Note: Setting custom keystore credentials addresses the issue of using the default, self-signed certificate created for usersync for port 5151. After performing this procedure, you can use your custom, CA-signed certificate.

To set Ranger Usersync custom keystore credentials:

Procedure

1. In Cloudera Manager Ranger Configuration , type Ranger Usersync Advanced Configuration Snippet in the search field.
2. In Ranger Usersync Advanced Configuration Snippet (Safety Valve) for conf/ranger-ugsync-site.xml , enter the following:
   a) In Name, type: ranger.usersync.keystore.file
   b) In Value, type: <keystore_file_path>
3. In Cloudera Manager Ranger Configuration , type Usersync Keystore Password in the search field.
4. In ranger.usersync.keystore.password, type a new password.
5. Click Save Changes.
6. Restart Ranger Usersync.

Results

Ranger uses the custom keystore file location and password values instead of the default values.

Enabling Ranger Usersync search to generate internally

You can configure Ranger Usersync to generate a search filter internally when Search includes a list of group names or group names with a wildcard character.

About this task

When you want to filter users who are members of “cdp_prod”, “cdp_testing”, or “dev_ops” groups, you can add a configuration, ranger.usersync.ldap.groupnames, that accepts each group name, as a domain name, a short name, or as a group name that contains a wildcard character. Usersync only reads ranger.usersync.ldap.groupnames when the sync source is AD/LDAP and ranger.usersync.ldap.user.searchfilter is empty. This also requires that ranger.usersync.group.searchbase is not empty and the configured value for ranger.usersync.group.searchbase must be part of the group searchbase in AD/LDAP. When ranger.usersync.ldap.user.searchfilter is not empty, Usersync ignores the value of ranger.usersync.ldap.groupnames. Values can be either DN of the groups, short name of the groups, or the group names with wildcard character. For example:

- Domain names of the groups
  - memberof=CN=dev_ops,ou=Hadoop Groups,dc=cloudera,dc=com
  - memberof=CN=cdp_prod,ou=Hadoop Groups,dc=cloudera,dc=com
  - memberof=CN=cdp_testing,ou=Hadoop Groups,dc=cloudera,dc=com
- Short names of the groups
  - CN=dev_ops
  - CN=cdp_prod
  - CN=cdp_testing
- Group names with wildcard character
  - CN=cdp*
  - CN=dev_ops

To enable Usersync search to generate an internal search filter for multiple groups names that include wildcard characters:

Procedure

1. In Cloudera Manager Ranger Configuration Search , type ranger.usersync.ldap.usergroups.
2. In Ranger Usersync Default Group, click +1.
3. Type <group_name>.

4. Repeat steps 2 and 3 for each group name.

Figure 11: Example supported group name formats for Usersync LDAP

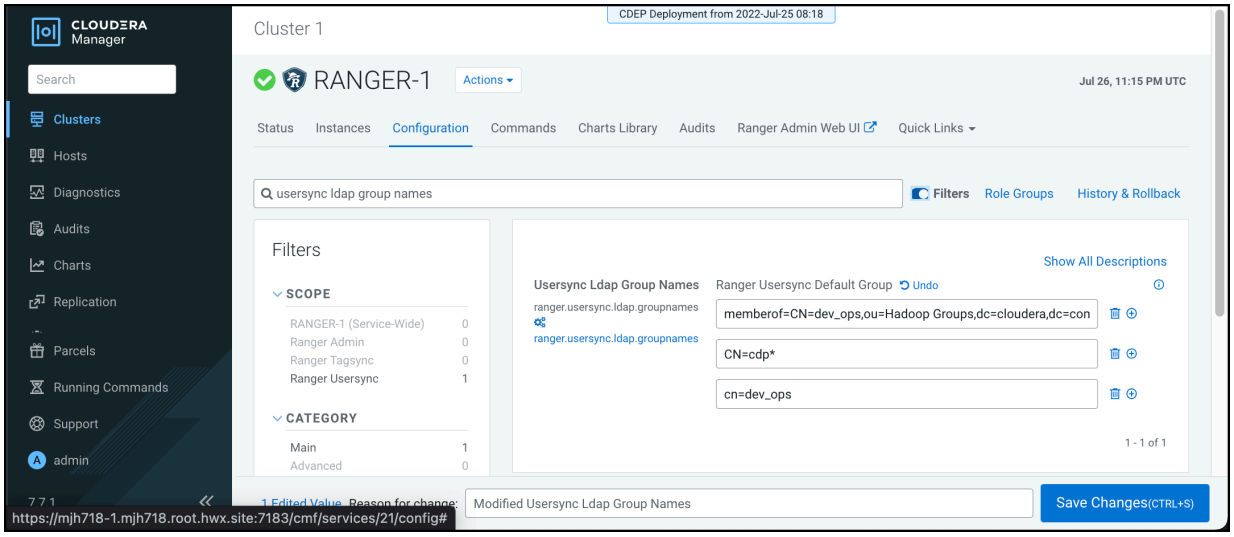

5. Click Save Changes.
6. In Actions, choose Restart Usersync.

Results

The search filter now includes all group names that you saved.

What to do next

To confirm, log in to Ranger Admin Web UI. In Settings Users/Group/Roles Groups , in Groups List, select Group Name. You should see group names that you configured available as search filter values.

Configuring Usersync to sync directly with LDAP/AD (FreeIPA)

Ranger Usersync can be manually configured to sync directly with LDAP/AD , using FreeIPA deployed in public cloud.

About this task

By default, Ranger Usersync uses sssd to sync users and groups from a Unix source. This can affect performance and limit scale. This runtime release supports LDAP/AD as a default sync source. Additionally, Ranger Usersync can be manually configured to:

- update users and groups from multiple (LDAP/AD, Unix and file) sync sources
- customize the default sync interval
- sync directly from FreeIPA deployed in public cloud

Procedure

1. Go to Cloudera Manager Ranger Configuration Filters Ranger Usersync .
2. In Search, type safety valve.
   This filters all Ranger configs to expose only the Usersync safety valves.

3. In Ranger Usersync Advanced Configuration Snippet (Safety Valve) for conf/ranger-ugsync-site.xml, press +.
   a) In Name, type ranger.usersync.syncsource.validation.enabled
   b) In Value, type false
   c) Click Save Changes(CTRL+S)

   Allows sync from multiple source types.
4. In Ranger Usersync Advanced Configuration Snippet (Safety Valve) for conf/ranger-ugsync-site.xml, press +.
   a) In Name, type ranger.usersync.cdp.public
   b) In Value, type true
   c) Click Save Changes(CTRL+S)

   Sets default sync interval to 1 min in public cloud.

   Note: This interval can be customized further using this config: ranger.usersync.sleeptimeinmillisbetweensynccycle
5. In Ranger Usersync Advanced Configuration Snippet (Safety Valve) for conf/ranger-ugsync-extra-attrs.xml, press +.
   a) In Name, type <cloud user.cloud.id.mapping>
   b) In Value, type clouduser1,clouduser2,….
   c) Click Save Changes(CTRL+S)

   Cloud ids for these users and groups are synced to ranger admin.

   Note: This will be populated by default. The above can also be manually overridden.

What to do next

Contact your Cloudera Support Engineer to manually configure Ranger Usersync to sync with a specific FreeIPA instance.

Configure SASL Bind in Ranger Usersync

Learn how to configure GSSAPI/Kerberos SASL Bind in Ranger Usersync.

About this task

Note: GSSAPI SASL Bind in Ranger Usersync is applicable for 7.1.9 SP1 CHF3 and above releases.

Usersync of Ranger supports Simple Bind to authenticate to a directory service and GSSAPI SASL Bind using Kerberos mechanism to authenticate to a directory service from Usersync. This topic describes how to configure GSSAPI SASL Bind in Ranger Usersync.

Procedure

1. In Cloudera Manager, go to Ranger Configuration .
2. Configure the following properties of admin and usersync:

Table 1: LDAP configurations

Property	Values
Admin Configurations	
ranger.authentication.method	ACTIVE_DIRECTORY
ranger.ldap.ad.domain	<ldap-ad-domain>

Property	Values
ranger.ldap.ad.url	ldap://<ldaphost>:389 For SSL: ldaps://<ldaphost>:636
ranger.ldap.ad.base.dn	DC=qe-infra-ad,DC=example,DC=com
ranger.ldap.ad.bind.dn	rangerusersync@<ldap-ad-domain>
ranger.ldap.ad.bind.password	<Ldap-password>
ranger.ldap.ad.referral	ignore
ranger.ldap.ad.user.searchfilter	(sAMAccountName={0})
Usersync Configurations	
ranger.usersync.ldap.user.searchbase	CN=Users,DC=qe-infra-ad,DC=example,DC=com
ranger.usersync.source.impl.class	org.apache.ranger.ldapusersync.process.LdapUserGroupBuilder
ranger.usersync.ldap.url	ldap://<ldaphost>:389 For SSL, ldaps://<ldaphost>:636
ranger.usersync.ldap.binddn	rangerusersync@<ldap-ad-domain>
ranger_usersync_ldap_ldapbindpassword	<Ldap-password>
ranger.usersync.ldap.starttls	false For SSL, true
ranger.usersync.ldap.user.nameattribute	sAMAccountName
ranger.usersync.ldap.user.objectclass	person
ranger.usersync.ldap.user.searchscope	sub
ranger.usersync.ldap.user.searchfilter	cn=*
ranger.usersync.group.nameattribute	cn
ranger.usersync.group.memberattributename	member
ranger.usersync.group.searchfilter	cn=*
ranger.usersync.group.searchbase	OU=groups,DC=qe-infra-ad,DC=example,DC=com
ranger.usersync.group.objectclass	group
ranger.usersync.ldap.authentication.mechanism	GSSAPI **Note:** Add the ranger.usersync.ldap.authentication.mechanism configuration in Safety Valve for conf/ranger-ugsync-site.xml.

3. Click Save Changes.
4. Add truststore details in usersync truststore:
   a) To get the AD cert to pem file:

```
openssl s_client -showcerts -connect <ldap-host>:636 -servername <server
_name> </dev/null 2>/dev/null > my_ldaps_cert.pem
```

   b) To view the content of the pem file:

```
<path_to_keytool>/keytool -printcert -file my_ldaps_cert.pem
```

   c) To add the pem file to usersync truststore:

```
<path_to_keytool>/keytool -import -alias ALIAS -file my_ldaps_cert.pem -
storetype JKS -keystore <path to truststore.jks> -storepass <truststoe_p
assword>
```

5. Restart the service.

Force deletion of external users and groups from the Ranger database

You can force delete external users and groups from a Ranger database using either REST endpoints or python client APIs.

If Ranger Usersync operates without proper configuration, a Ranger database may be (over)-populated with user and group records. Huge user/group and mapping tables may cause Ranger administration issues. To aid in removal of unnecessary users/groups, customers may use this feature to delete specific external user/groups or even all external users/groups if required.

Note: Deleting users and groups manually is not an optimal solution for all customers.

The following 2 REST endpoints are available:

- service/xusers/delete/external/groups
- service/xusers/delete/external/users

The following 2 python client APIs are available:

- force_delete_external_users
- force_delete_external_groups

Invoking REST endpoints (via cURL):

```
# to delete a group named 'group_1'
service/xusers/delete/external/groups?name=group_1
# to delete all hidden groups
service/xusers/delete/external/groups?isVisible=0
# to delete all visible groups
service/xusers/delete/external/groups?isVisible=1
# to delete all groups with sync source as Unix
service/xusers/delete/external/groups?syncSource=Unix
# to delete a user named 'user_1'
service/xusers/delete/external/users?name=user_1
# to delete all hidden users
service/xusers/delete/external/users?isVisible=0
# to delete a user with a particular email id
service/xusers/delete/external/users?emailAddress=xyz5@gmail.com
```

To use the python client APIs:

Note: apache_ranger python package version should be >= 0.0.11

```
$ pip3 install apache_ranger
$ python3
>>> from apache_ranger.client.ranger_client import *
>>> ranger = RangerClientPrivate('<ranger_url>', ('<ranger_user>', '<ranger_
pass>'))
```

Specific use-case examples using the python client APIs:

```
# to delete user_1
>>> ranger.force_delete_external_users("name=user_1")
# to delete group_1
>>> ranger.force_delete_external_groups("name=group_1")
# to delete all users with sync source as Unix
```

```
>>> ranger.force_delete_external_users("syncSource=Unix")
# to delete all users with Auditor Role
>>> ranger.force_delete_external_users("userRole=ROLE_ADMIN_AUDITOR")
# to delete all external users
>>> ranger.force_delete_external_users()
# to delete all external groups
>>> ranger.force_delete_external_groups()
```

Cloudera Runtime 7.3.1

Configuring Streams Messaging Manager

Date published: 2023-09-25
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Installing Streams Messaging Manager in Cloudera Public Cloud.....4

Setting up Prometheus for Streams Messaging Manager..... 4

Prometheus configuration for Streams Messaging Manager..... 6
Prerequisites for Prometheus configuration..... 6
Prometheus properties configuration.....6
Streams Messaging Manager property configuration in Cloudera Manager for Prometheus..... 9
Kafka property configuration in Cloudera Manager for Prometheus..... 9
Kafka Connect property configuration in Cloudera Manager for Prometheus..... 9
Start Prometheus..... 10
Secure Prometheus for Streams Messaging Manager.....10
Nginx proxy configuration over Prometheus.....10
Setting up TLS for Prometheus.....11
Setting up basic authentication with TLS for Prometheus..... 13
Setting up mTLS for Prometheus..... 14
Prometheus for Streams Messaging Manager limitations.....15
Troubleshooting Prometheus for Streams Messaging Manager..... 15
Performance comparison between Cloudera Manager and Prometheus.....15

Installing Streams Messaging Manager in Cloudera Public Cloud

Learn how Streams Messaging Manager is installed and set up in Cloudera Public Cloud with the Streams Messaging cluster templates.

In Cloudera Public Cloud, Streams Messaging Manager is automatically installed on clusters that you deploy with the Streams Messaging Light Duty, Heavy Duty, and High Availability templates available in the Cloudera Data Hub service. A single instance of the Streams Messaging Manager service (Rest Admin Server and UI Server roles) is installed with each of the templates.

In the Light and High Availability templates, Streams Messaging Manager is installed in the **Master Nodes** host group, while in Heavy Duty, Streams Messaging Manager is installed in the **Streams Messaging Manager Nodes** host group. For more information regarding the Streams Messaging cluster templates and cluster deployment with Cloudera Data Hub, see the *Related Information* section below.

Integration with other services

Streams Messaging Manager is capable of integrating with other services. Enabling integration enables additional features and functions on the Streams Messaging Manager UI.

In Cloudera Public Cloud, integration with other, eligible Cloudera Streams Messaging services is set up and configured automatically when you deploy the cluster using one of the Streams Messaging templates. For example, if your Streams Messaging cluster includes Kafka Connect, Streams Messaging Manager is automatically configured to integrate with the Kafka Connect instances running on the cluster. The **Connect** section of the Streams Messaging Manager UI will be available by default, enabling you to manage and monitor Kafka Connect using Streams Messaging Manager.

For Cruise Control, even though the integration with Streams Messaging Manager is automatic, users must have at least Viewer level authorization in Cruise Control. Otherwise, the Cruise Control UI will not be visible in Streams Messaging Manager. For more information, see *Authorizing users to access Cruise Control in Streams Messaging Manager*.

Related Information

Setting up your Streams Messaging cluster

Streams Messaging cluster layout

Monitoring Streams Replication Manager

Setting up Prometheus for Streams Messaging Manager

Get started with setting up and using Prometheus for Streams Messaging Manager. You can use Prometheus as the metrics store for Streams Messaging Manager. Its use is recommended over Cloudera Manager if your deployment includes a significantly large number Kafka entities that you want to monitor.

Cloudera Manager metrics overview

A metric is a property that can be measured to quantify the state of an entity or activity. They include properties such as the number of open file descriptors or CPU utilization percentage across your cluster.

Cloudera Manager monitors a number of performance metrics for services and role instances running on your clusters. These metrics are monitored against configurable thresholds and can be used to indicate whether a host is functioning as expected or not. You can view these metrics in the Cloudera Manager Admin Console.

Cloudera agents collect the metrics from individual brokers and report them to Cloudera Manager once in a minute (through Heartbeat). The collected metrics are stored in the Cloudera Manager database to query or search for historical data.

Cloudera Manager provides a default metric store for Streams Messaging Manager. Streams Messaging Manager fetches the required metrics from Cloudera Manager whenever required and caches the subset of metrics in the Streams Messaging Manager server for which the historical values appear in the Streams Messaging Manager UI. The cache refresh is configurable, the default is 50 seconds.

Prometheus overview

Prometheus is a metrics store that pulls metrics from different endpoints which you configure. Prometheus is not the default metric store for Streams Messaging Manager. If you want to use Prometheus as the metrics store for Streams Messaging Manager, you need to download and configure it. Prometheus supports a larger number of time series entities compared to the Cloudera Manager metrics store.

If you use Prometheus, you can configure the roll up policy, delete specific time series entities, and configure scrape interval and metrics retention period.

For Streams Messaging Manager, you need to configure the following endpoints for Prometheus to pull metrics from:

- Kafka

  Kafka exposes a Prometheus metrics endpoint for Kafka metrics to be pulled.
- Kafka Connect

  Kafka Connect, through configuration, exposes a Prometheus endpoint for Kafka connect metrics to be pulled.
- Prometheus Node Exporter

  You need to configure a separate Node Exporter on each Kafka broker host and enable Prometheus to pull the system metrics.

Streams Messaging Manager queries Prometheus for metrics over time. Prometheus fetches the metrics from the endpoints.

Prometheus relies on external tools for security. For example, you can secure your Prometheus with Nginx in the following scenarios:

- TLS
- TLS with basic authentication
- mTLS

The following image shows the architecture of Prometheus configured for Streams Messaging Manager and secured with Nginx:

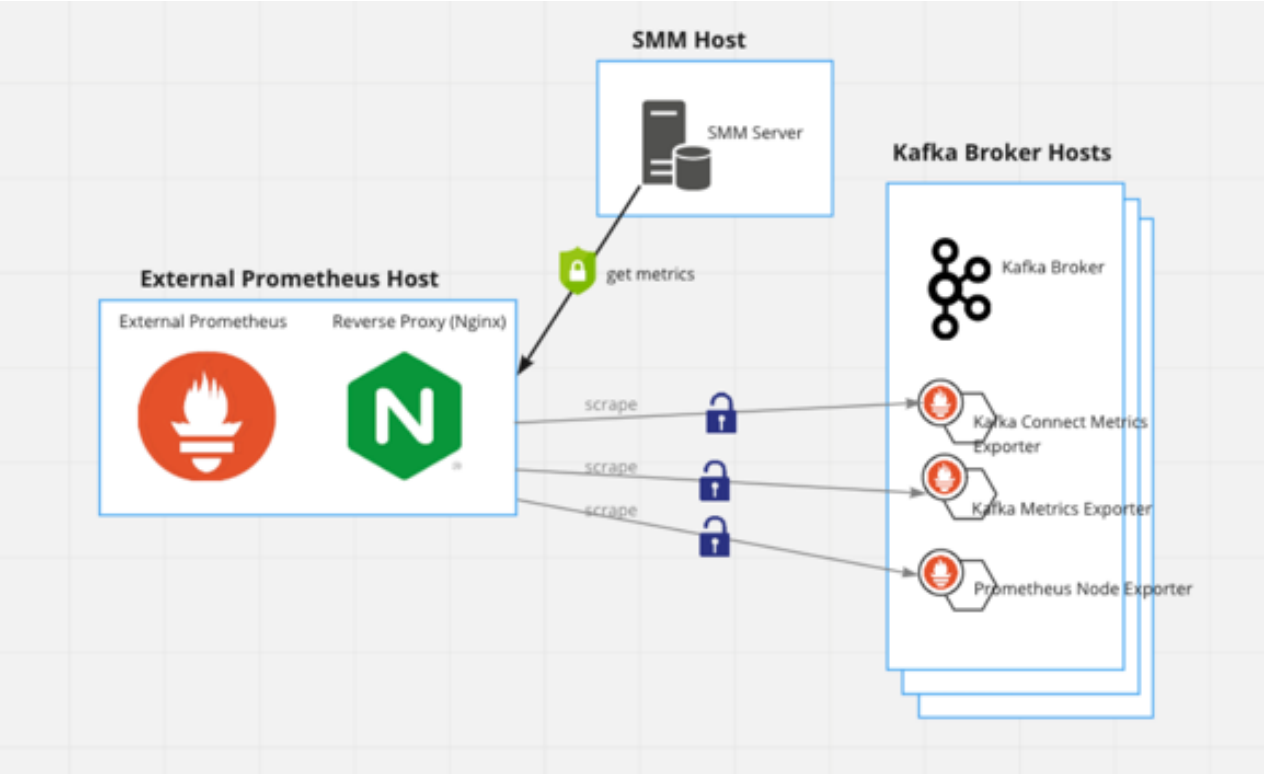

Prometheus configuration for Streams Messaging Manager

Prometheus is not the default metric store for Streams Messaging Manager. If you want to use Prometheus as the metric store for Streams Messaging Manager, you need to download and configure it.

Prerequisites for Prometheus configuration

Learn the prerequisites before you configure Prometheus for Streams Messaging Manager.

- You must download Prometheus and install Prometheus and Prometheus node exporters for each Kafka broker. Streams Messaging Manager requires system-level metrics for each Kafka broker and therefore, Prometheus node exporter must be configured for each Kafka broker. Cloudera only supports Linux node exporters.
- Cloudera recommends using a dedicated Prometheus instance for Streams Messaging Manager because other services querying Prometheus could use the capacity causing Streams Messaging Manager query to timeout.

Prometheus properties configuration

Learn the properties that you need to configure in the prometheus.yml file before you start using the Prometheus metric store for Streams Messaging Manager.

Configure the following properties in the prometheus.yml file:

- Set the scrape_interval property value to 60 seconds in the prometheus.yml file.

```
scrape_interval: 60s
```

- Prometheus automatically assigns the instance label to metrics. However, in case partition leader change reassignment happens, it means an excessive amount of metrics being created. This property is recommended for large clusters.

```
    metric_relabel_configs:
    - source_labels: [__name__]
```

```
      regex: ^(broker_producer_messagesinpersec_total|topic_partition_me
ssagesinpersec_total|topic_partition_bytesinpersec_total|topic_partition
_bytesoutpersec_total)$
      target_label: instance
      replacement: 'no-instance'
```

- Set Kafka host and metrics port values.

```
['luigi-1.luigi.root.hwx.site:24042','luigi-2.luigi.root.hwx.site:24042'
,'luigi-3.luigi.root.hwx.site:24042']
```

  You can find the metric port used by Kafka in Cloudera Manager Kafka service Configuration HTTP Metric Report Port .
- Set Kafka Connect host (deployed on same hosts as Kafka) and the metrics port values.

```
['luigi-1.luigi.root.hwx.site:28087','luigi-1.luigi.root.hwx.site:28087'
,'luigi-1.luigi.root.hwx.site:28087']
```

  The metrics port used by Kafka Connect is set in Cloudera Manager Kafka service Configuration Secure Jetty Metrics Port and Jetty Metrics Port. Use Secure Jetty Metrics Port if the Kafka Connect metrics reporter uses TLS/SSL, otherwise, use Jetty Metrics Port. The value of 28087 in this example is the default secure port.
- Set Prometheus node exporter host (deployed on same hosts as Kafka) and the Prometheus node exporter metrics port values.

```
['luigi-1.luigi.root.hwx.site:9100','luigi-2.luigi.root.hwx.site:9100','
luigi-3.luigi.root.hwx.site:9100']
```

- update=true parameter should be only used by Prometheus. Querying Kafka Prometheus endpoint with this flag set to true updates the internal metrics cache for stateful metrics.

```
update: ['true']
```

- If Basic Authentication is enabled for the Kafka Connect metrics reporter, add the following parameters:

```
basic_auth:
  username: 'email@username.me'
  password: 'password'
```

  Ensure that you specify the username and password that is configured for the Kafka Connect metrics reporter. You can find the username and password in Cloudera Manager Kafka service Configuration Jetty Metrics User Name and Jetty Metrics Password.

Configuration example

The following is an example of a Prometheus configuration YAML. You can use this example as a template and make changes as necessary. Ensure that you replace host and port values as well as the Basic Authentication credentials with your own values. This example uses the default ports.

```
# my global config
global:
  scrape_interval: 60s
  scrape_timeout: 55s

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093
```

```
# Load rules once and periodically evaluate them according to the global '
evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries s
craped from this config.
  - job_name: 'prometheus'
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
    - targets: ['localhost:9090']

  - job_name: 'kafka'

    metrics_path: '/api/prometheus-metrics'

    params:
      update: ['true']

    static_configs:
    - targets: ['luigi-1.luigi.root.hwx.site:24042','luigi-2.luigi.root.hwx.
site:24042','luigi-3.luigi.root.hwx.site:24042']
    metric_relabel_configs:
    - source_labels: [__name__]
      regex: ^(broker_producer_messagesinpersec_total|topic_partition_messa
gesinpersec_total|topic_partition_bytesinpersec_total|topic_partition_byteso
utpersec_total)$
      target_label: instance
      replacement: 'no-instance'

  - job_name: 'kafka_connect'

    metrics_path: '/prometheus-metrics'

    basic_auth:
      username: 'email@username.me'
      password: 'password'

    static_configs:
    - targets: ['luigi-1.luigi.root.hwx.site:28087','luigi-1.luigi.root.hwx.
site:28087','luigi-1.luigi.root.hwx.site:28087']

  - job_name: 'system_metrics'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['luigi-1.luigi.root.hwx.site:9100','luigi-2.luigi.root.hwx
.site:9100','luigi-3.luigi.root.hwx.site:9100']
```

Where:

- ['luigi-1.luigi.root.hwx.site:9100','luigi-2.luigi.root.hwx.site:9100','luigi-3.luigi.root.hwx.site:9100'] = Kafka host + Node exporter metrics port
- ['luigi-1.luigi.root.hwx.site:24042','luigi-2.luigi.root.hwx.site:24042','luigi-3.luigi.root.hwx.site:24042'] = Kafka host + HTTP Metric Report Port

- ['luigi-1.luigi.root.hwx.site:28087','luigi-1.luigi.root.hwx.site:28087','luigi-1.luigi.root.hwx.site:28087'] = Kafka Connect host + Secure Jetty Metrics Port

Streams Messaging Manager property configuration in Cloudera Manager for Prometheus

Learn about the Streams Messaging Manager properties that you need to configure in Cloudera Manager before you start using the Prometheus metric store.

Configure the following Streams Messaging Manager properties in Cloudera Manager:

- metrics.fetcher.class

  Configures Streams Messaging Manager to fetch metrics from Prometheus. Set it to com.hortonworks.smm.kafka.services.metric.prometheus.PrometheusMetricsFetcher.
- prometheus.metrics.url

  Prometheus metrics URL should be configured here in the format of scheme://prometheus_host:prometheus_port. If HTTPS is configured for Prometheus, HTTPS should be specified as the scheme.
- prometheus.metrics.user

  Should be set if Prometheus is configured for TLS with basic authentication.
- prometheus.metrics.password

  Should be set if Prometheus is configured for TLS with basic authentication.

Kafka property configuration in Cloudera Manager for Prometheus

Learn about the Kafka properties that you need to configure in Cloudera Manager before you start using the Prometheus metric store.

Configure the following Kafka property in Cloudera Manager:

- kafka.http.metrics.port

  Used for exposing Cloudera Manager metrics. This port is now also shared to expose Prometheus metrics for Kafka.

Kafka Connect property configuration in Cloudera Manager for Prometheus

Learn about the Kafka Connect properties that you need to configure in Cloudera Manager before you start using the Prometheus metric store.

In order to set up Prometheus as the Streams Messaging Manager metrics store, you need to configure the metrics reporter of Kafka Connect. Configuration is done in Cloudera Manager by setting Kafka Connect properties. Configuration differs depending on whether you want to enable security for the Kafka Connect metrics reporter. All of the following properties are found in Cloudera Manager Kafka service Configuration . Completing the following makes Prometheus-compatible metrics available on the /prometheus-metrics API path on each Kafka Connect worker host.

Do the following to configure an unsecured metrics-scraping endpoint:

1. Add metrics.jetty.server.prometheus.metrics.enable=true to the Kafka Connect Advanced Configuration Snippet (Safety Valve) for connect-distributed.properties advanced configuration snippet.
2. Look up or configure the value of Jetty Metrics Port.

   This is the port where Kafka Connect exposes its metrics when TLS/SSL is not enabled. This is the port that you need to add to prometheus.yml. Only configure this property if you want to change the port.

Do the following to configure a secure metrics scraping endpoint:

1. Add metrics.jetty.server.prometheus.metrics.enable=true to the Kafka Connect Advanced Configuration Snippet (Safety Valve) for connect-distributed.properties advanced configuration snippet.

2. Enable TLS/SSL, Basic Authentication (BA), or both for the metrics reporter by configuring the following properties:
   - Enable TLS/SSL for Kafka Connect
   - Enable Basic Authentication for Metrics Reporter
   - Jetty Metrics User Name
   - Jetty Metrics User Password

   Note the following about these properties:
   - The Enable TLS/SSL for Kafka Connect property is not specific to the metrics reporter. It enables TLS/SSL for Kafka Connect roles including their metrics reporter.
   - You can enable both TLS/SSL and BA on their own, however, Cloudera recommends that you enable both.
   - Jetty Metrics User Name and Jetty Metrics User Password set the username and password that you need to add to prometheus.yml if you enable BA.
3. Look up or configure the value of Secure Jetty Metrics Port or Jetty Metrics Port

   These are the ports where Kafka Connect exposes its metrics. Secure Jetty Metrics Port is only used if TLS/SSL is enabled, otherwise Kafka Connect uses Jetty Metrics Port. You need to add the port being used to prometheus.yml. Only configure this property if you want to change the port.

Start Prometheus

You should start Prometheus with the startup flag options that you need.

For example,

```
./prometheus --storage.tsdb.retention.time 30d --web.enable-admin-api --query.max-samples 5000000000
```

- --storage.tsdb.retention.time 30d

  Prometheus defaults to 15 days. Streams Messaging Manager monitors metrics up to 30 days. Therefore, this flag must be configured.
- --web.enable-admin-api

  Optional. If Admin APIs are needed, such as deleting time series entities, this flag should be set.
- --query.max-samples

  Optional. When many entities or samples exist, bulk query might reach the maximum sample limit.

Secure Prometheus for Streams Messaging Manager

Streams Messaging Manager supports connecting to Prometheus servers behind a TLS proxy (for example, Nginx).

You can connect to Prometheus servers behind a TLS proxy in the following scenarios:

- TLS

  Streams Messaging Manager verifies the proxy's certificate.
- TLS with basic authentication

  Streams Messaging Manager verifies the proxy's certificate and authenticates itself with username and password.
- mTLS

  Streams Messaging Manager verifies the proxy's certificate and authenticates itself with its TLS Certificate (TLS proxy should recognize it).

Nginx proxy configuration over Prometheus

You need to install and configure Nginx before using it to configure proxy over Prometheus.

Nginx installtion

To use Nginx, you need to install it.

For information about how to install Nginx, see Nginx documentation.

Nginx configuration for Prometheus

Prometheus does not, by default, support TLS encryption for connections to Prometheus instances. If you want to enforce TLS encryption for the connections, you can use Prometheus in conjunction with a reverse proxy and apply TLS at the proxy layer. You can use any reverse proxy, but in this guide you see an Nginx example.

For details about securing Prometheus using TLS encryption, see Prometheus documentation.

Note: In the above linked documentation it is presumed that you want to run Ningx in the 443 port. If it is not the case (for example, it runs on 9443 port), it is not necessary to run Nginx as root. However, you must pay attention to the following things:

- Ensure that the nginx user has access to the TLS certificates.
- The web.external-url parameter of the Prometheus start command must contain the port number. For example,

```
--web.external-url=https://myprometheus.com:9443/prometheus
```

After you configure a server for Prometheus, you may want to disable the default server in Nginx configuration. The following example shows the default server commented out:

```
#    server {
#        listen       80 default_server;
#        listen       [::]:80 default_server;
#        server_name  _;
#        root         /usr/share/nginx/html;
#
#        # Load configuration files for the default server block.
#        include /etc/nginx/default.d/*.conf;
#
#        location / {
#        }
#
#        error_page 404 /404.html;
#        location = /404.html {
#        }
#
#        error_page 500 502 503 504 /50x.html;
#        location = /50x.html {
#        }
#    }
```

Setting up TLS for Prometheus

You need to configure Streams Messaging Manager when a TLS proxy is configured over Prometheus.

You need to configure the following:

- Ensure that Cloudera Manager or Streams Messaging Manager recognizes Nginx's TLS certificate. For details, see *Configure Streams Messaging Manager to recognize Prometheus's TLS certificate*.
- Update the Prometheus URL in Streams Messaging Manager settings. You must update the prometheus.metrics.url property to point to the TLS proxy's endpoint (for example, https://myprometheus.com:9443/prometheus) and restart Streams Messaging Manager.

Related Information

Configuring Streams Messaging Manager to recognize Prometheus's TLS certificate

Configuring Streams Messaging Manager to recognize Prometheus's TLS certificate

You can configure Streams Messaging Manager either to use its own keystore or truststore, or to use the auto-TLS feature in Cloudera Manager. Cloudera recommends using the auto-TLS feature for Cloudera clusters.

About this task

If the TLS proxy certificate is not recognized by Streams Messaging Manager, it must be added to the Streams Messaging Manager truststore. The process is different for auto-TLS and the manual TLS setups.

Auto-TLS

If the TLS proxy certificate is not recognized by the cluster, you can add the TLS proxy certificate to the CA truststore of the cluster by triggering a certificate regeneration. This involves restarting the services in the cluster.

1. Go to Administration Security from the left menu bar.

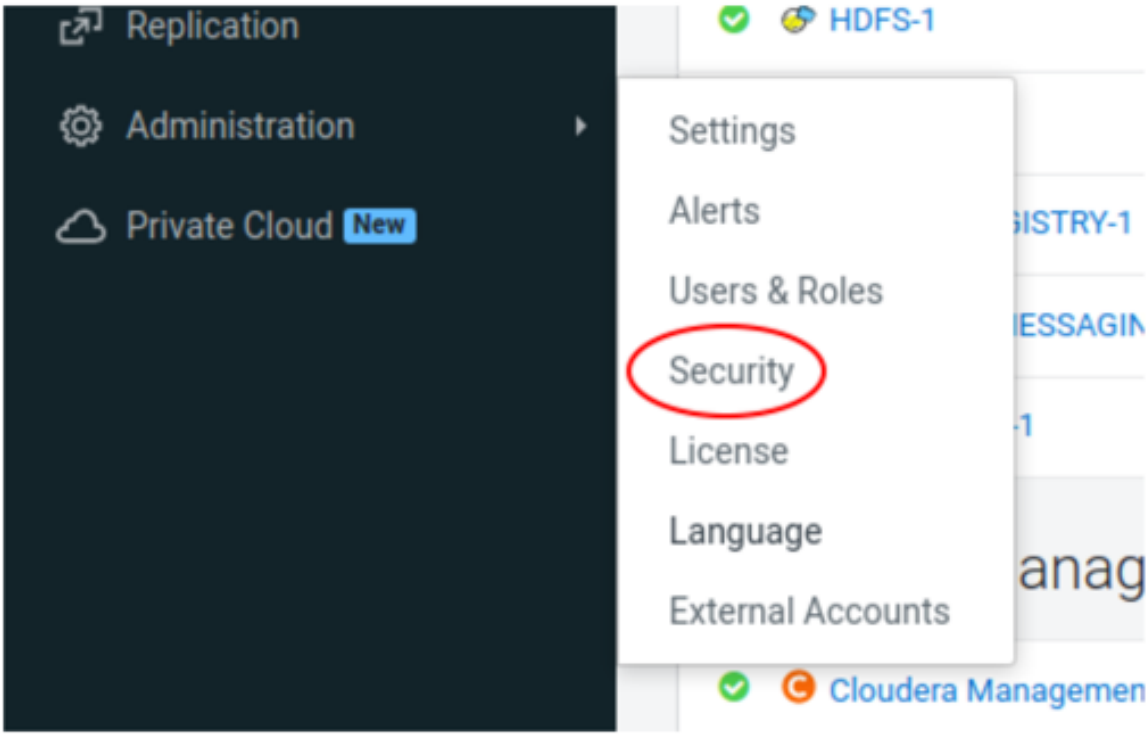

2. Click Rotate Auto-TLS Certificates.
3. In the Trusted CA Certificates Location field, enter the path to the Nginx server's certificate. For example, /etc/nginx/certs/ca-certificate.pem. Ensure that the file is accessible by the cloudera-scm user.
4. Specify the authentication method with other nodes of the cluster (password or certificate).
5. Click Next and follow the instructions in the wizard.

Manual TLS

You can use the keytool command to configure the manual TLS settings.

Keytool is a tool provided by the Java Runtime Environment to manipulate JKS type keystores. You can find it in the bin folder of your JRE installation. For example, /usr/java/default/jre/bin/keytool.

1. Use the following command to add the TLS proxy certificate to the Streams Messaging Manager truststore:

```
keytool -import -file <TLS PROXY OR CA CERTIFICATE> -alias Nginx_for_Pro
metheus -keystore <STREAMS MESSAGING MANAGER TRUSTSTORE> -storepass <TRU
STSTORE PASSWORD>
```

   For example,

```
keytool -import -file /etc/nginx/certs/ca-certificate.pem -alias Nginx_f
or_Prometheus -keystore smm_trusstore.jks
```

   This command creates the truststore if it does not already exist.
2. Create a keystore for Streams Messaging Manager if it does not already exist:

```
keytool -genkey -keystore smm_keystore.jks -alias smm -keyalg RSA -sigalg
 SHA256withRSA -validity 365 -keysize 3072
```

   It creates a keystore with a self-signed key.
3. Set the following Streams Messaging Manager properties in Cloudera Manager:
   - streams.messaging.manager.ssl.keyStorePath/ssl_server_keystore_location
   - ssl_server_keystore_password
   - ssl_server_keystore_keypassword (by default it is the same as the keystore file password)
   - streams.messaging.manager.ssl.trustStorePath/ssl_client_truststore_location
   - ssl_client_truststore_password

Related Information

Setting up TLS for Prometheus

Setting up basic authentication with TLS for Prometheus

To set up TLS with basic authentication, you need to configure Nginx and Streams Messaging Manager.

Configuring Nginx for basic authentication

To configure Nginx for basic authentication, you need to create a file for user and password, update Nginx configurations, and restart Nginx.

Procedure

1. Create an user-password file for Nginx.

```
htpasswd -c /etc/nginx/.htpasswd admin
```

   This requires the Apache HTTP package to be installed on the system.
2. Update your Nginx configuration (/etc/nginx/nginx.conf or a custom configuration file in the /etc/nginx/conf.d directory) with the highlighted portion in the code below:

```
    server {
        listen              9443 ssl;
        server_name         _;
        ssl_certificate     /<PATH TO CERTIFICATE>/nginx-server-crt.pem;
        ssl_certificate_key /<PATH TO CERTIFICATE>/nginx-server-key.pem;
        location /prometheus/ {
            auth_basic           "Prometheus";
            auth_basic_user_file /etc/nginx/.htpasswd;
            proxy_pass http://localhost:9090/;
        }
    }
```

3. Restart Nginx.

Configuring Streams Messaging Manager for basic authentication

To configure Streams Messaging Manager, you need to configure Prometheus username and password, and restart Streams Messaging Manager.

Procedure

1. Set the following configurations:
   - Prometheus User (admin)
   - Prometheus Password (the password you gave in the htpassword tool)
2. Restart Streams Messaging Manager.

Setting up mTLS for Prometheus

Along with or instead of basic authentication, mTLS can also be used for client authentication.

About this task

When using mTLS, both the server and the client authenticate themselves with a TLS certificate. As Streams Messaging Manager is configured to recognize Nginx's certificate, it needs to be configured the other way around.

Procedure

1. Export the certificate or CA certificate of Streams Messaging Manager.
   - In case of Auto-TLS, it is

     ```
     /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem
     ```

     or

     ```
     /var/lib/cloudera-scm-agent/agent-cert/cm-auto-in_cluster_ca_cert.pem
     ```

   - In case of manual TLS, you can use keytool to export the certificate. For example,

     ```
     keytool -exportcert -rfc -keystore /opt/cloudera/smm_keystore.jks -alias
      smm -file smm.cer
     ```

2. Add the highlighted lines to Nginx server configuration (/etc/nginx/nginx.conf or a custom configuration file in the /etc/nginx/conf.d directory).

   ```
       server {
           listen              9443 ssl;
           server_name         _;
           ssl_certificate     /<PATH TO CERTIFICATE>/nginx-server-crt.pem;
           ssl_certificate_key /<PATH TO CERTIFICATE>/nginx-server-key.pem;
           ssl_client_certificate /<PATH TO STREAMS MESSAGING MANAGER
     CERTIFICATE>;
           ssl_verify_client      on;

           location /prometheus/ {
               proxy_pass http://localhost:9090/;
           }
       }
   ```

3. Restart Nginx.

Prometheus for Streams Messaging Manager limitations

Learn about the known issues and limitations, the areas of impact, and workaround while using Prometheus for Streams Messaging Manager.

- Prometheus is not managed by Cloudera Manager. You should start Prometheus right after Kafka startup because certain metrics are stateful within Kafka and might cause the initial scrape metric values for Prometheus to be high. The stateful metrics accumulate the values from several Kafka restarts until the initial scrape, and accumulate the values between consecutive scrapes.
- You need to configure the scrape interval to 60 seconds.
- Streams Messaging Manager supports Prometheus 2.25 and above versions.
- Streams Messaging Manager supports Linux 1.1.2 and above versions of node exporters.
- Depending on the number of entities (topics, partitions, consumers, and producers), memory requirements for Prometheus might vary.

Troubleshooting Prometheus for Streams Messaging Manager

Troubleshooting Prometheus for Streams Messaging Manager requires being able to diagnose and debug problems related to performance, network connectivity, out-of-memory conditions, disk space usage, and crash or non-responsive conditions.

This section provides some troubleshooting solutions for Prometheus issues for Streams Messaging Manager.

Issue: If no metrics are available.

Solution: Examine if all configured Prometheus targets are running.

Issue: You observe incorrect metrics in Streams Messaging Manager.

Solution: Examine up metrics to determine if scraping failed at certain points in time.

Issue: If Prometheus does not start up or scrape right after Kafka startup, or Prometheus is down for some time, scrapped metric values might be unusually high.

Solution: This situation can be avoided by querying the endpoints manually with the update=true parameter and then starting Prometheus. If you already encountered this situation, you can delete that particular time series with the delete API of Prometheus.

Performance comparison between Cloudera Manager and Prometheus

Learn the performance comparison between Cloudera Manager and Prometheus metric stores.

- Prometheus can handle more than twice the amount of TimeSeries entities than Cloudera Manager while maintaining the consistency of the metrics.
- Metrics queries for shorter time periods (30 minutes, 1 hour) take about half the time in case of Prometheus compared to Cloudera Manager.
- Metrics queries for larger time periods (2 weeks, 1 month), Prometheus seems to be about 30% slower than Cloudera Manager.
- Metric inconsistencies are not observed while testing Prometheus.

Cloudera Runtime 7.3.1

Streams Messaging Manager Overview

Date published: 2019-08-22
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to Streams Messaging Manager..... 4

Introduction to Streams Messaging Manager

Streams Messaging Manager is an operations monitoring and management tool that provides end-to-end visibility in an enterprise Apache Kafka® environment.

With Streams Messaging Manager, you can gain clear insights about your Kafka clusters. You can understand the end-to-end flow of message streams from producers to topics to consumers. Streams Messaging Manager helps you troubleshoot your Kafka environment to identify bottlenecks, throughputs, consumer patterns, traffic flow etc. Streams Messaging Manager enables you to analyze the stream dynamics between producers and consumers using various filters. You can optimize your Kafka environment based on the key performance insights gathered from various brokers and topics. With the tight integration of Apache Atlas, you can gain complete data lineage across multiple Kafka hops, producers and consumers with powerful data flow visualization.

Simplifies troubleshooting Kafka environments

Streams Messaging Manager provides intelligence-based filtering that allows a user to select a producer, broker, topic or consumer and see only related entities based on the selection. Streams Messaging Manager is smart enough to show only those producers that are sending data to the selected topics and show only those consumer groups that are consuming from those topics. The filtering works on the selection of any of the four entities. This enables users to quickly hone in on the root cause when troubleshooting and debugging Kafka issues.

Visualizes end-to-end Kafka stream flows

Another powerful feature of Streams Messaging Manager is its ability to visualize all the data streams/flows across all your Kafka clusters. You can select any entity and visualize how data flows with respect to the entity selected.

Extends monitoring and management capabilities with REST API

Streams Messaging Manager offers REST endpoints for all of its capabilities. This enables developers to integrate Streams Messaging Manager with their other enterprise tools such as APM or case or ticketing systems.

Tracks data lineage and governance from edge-to-enterprise

Streams Messaging Manager has been fully integrated with Apache Atlas for governance and data lineage, Apache Ranger for security and access control management, Apache Ambari for infrastructure level monitoring and lifecycle actions for the cluster, and Grafana to be able to graph Kafka metrics over time.

Visualize lineage between producers and consumers

With Streams Messaging Manager, you can visualize the lineage between producers and consumers.

Lineage information helps you to understand how the message is moving from a producer to a consumer group and which topics or partitions are part of that flow. This is just a high level information, intended for basic monitoring. For the fully featured lineage monitoring, see Atlas and its Kafka monitoring capabilities in *Kafka lineage*.

Related Information

Kafka lineage

Cloudera Runtime 7.3.1

Securing Streams Messaging Manager

Date published: 2019-08-22
Date modified: 2024-07-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Securing Streams Messaging Manager..... 4

Verifying the setup..... 6

Securing Streams Messaging Manager

As a cluster administrator, you can combine Kerberos authentication and Ranger authorization to secure the Streams Messaging Manager web user interface (UI). After you secure the Streams Messaging Manager web UI, the login page appears, which does not appear by default.

About this task

If you deploy Streams Messaging Manager without security, the login page is not enabled on the Streams Messaging Manager UI by default. When you enable Kerberos authentication, Streams Messaging Manager uses SPNEGO to authenticate users and allows them to view or create topics within Kafka by administering Ranger Kafka Policies. For information on enabling browsers to use SPNEGO, see How to Configure Browsers for Kerberos Authentication.

After you secure Streams Messaging Manager, anyone within the organization can login to Streams Messaging Manager. However, if they do not have the correct policy configuration in Ranger, then they may not have the necessary privileges to perform their required tasks through Streams Messaging Manager.

Before you begin

- Configure Kafka in Ranger

  For more information, see *Configure a resource-based service: Kafka*.
- Enable Kerberos authentication for Kafka

  For more information, see *Enable Kerberos authentication*.
- Add and configure Streams Messaging Manager

  For more information, see *Creating your first Streams Messaging cluster*.

Note: For the Kafka Client security protocol, it is possible to use INFERRED, SASL_PLAINTEXT, and SASL_SSL for securing Streams Messaging Manager. However, Cloudera recommends using SASL_SSL.

Procedure

1. Go to Cloudera Manager Streams Messaging Manager , and click Configuration.
2. Enable Ranger for Streams Messaging Manager.

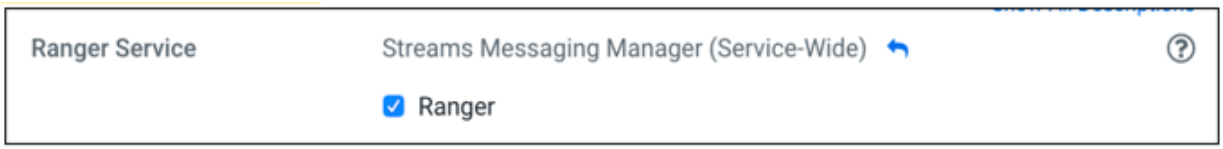

3. Go to the Ranger service UI and configure the Kafka policies.

   Note: Review your Ranger Kafka Policies. Remember to log in to Ranger with a user that has the Ranger Admin role.

4. Click cm_kafka in the Ranger service UI.

   The List of Policies page appears.

5. Click Add New Policy.

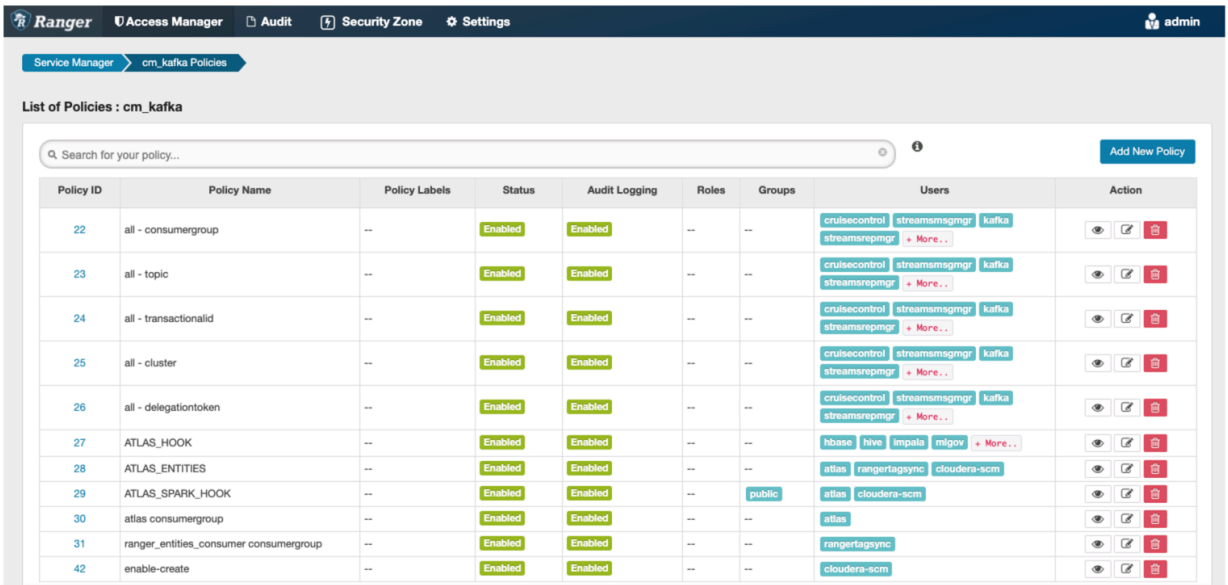

The Policy Details page appears.

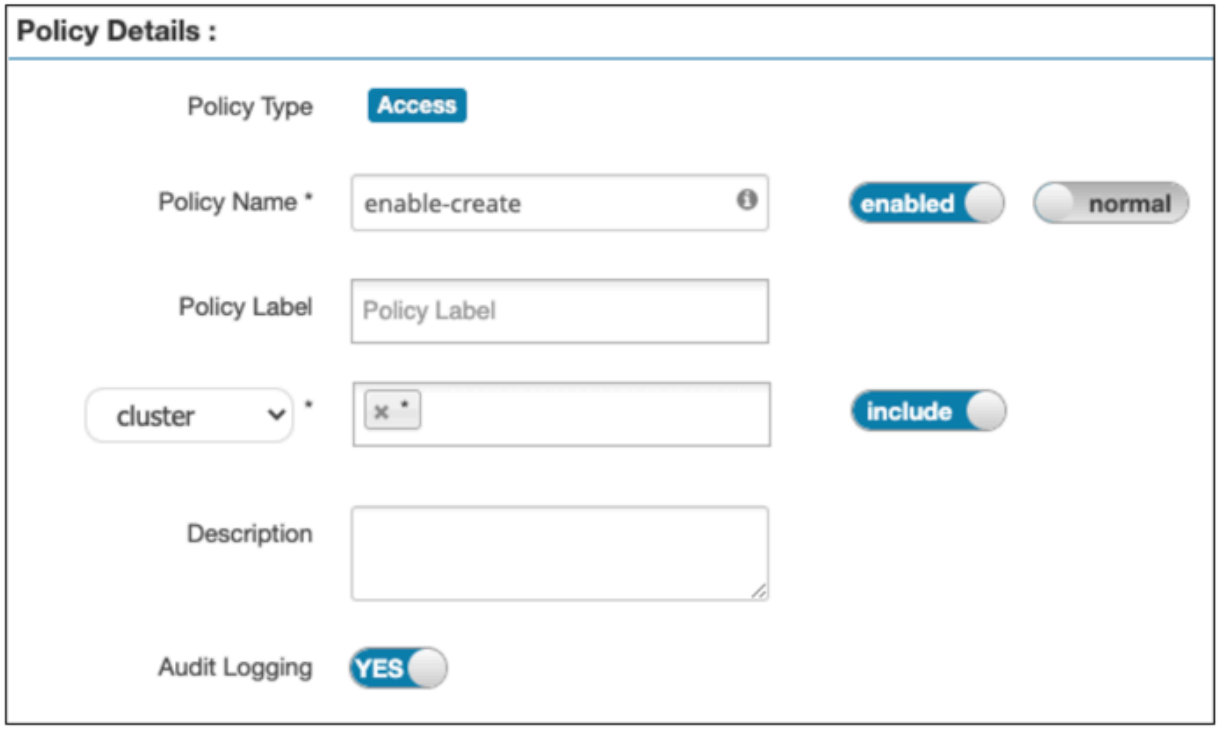

6. Add a policy name and select cluster from the dropdown.

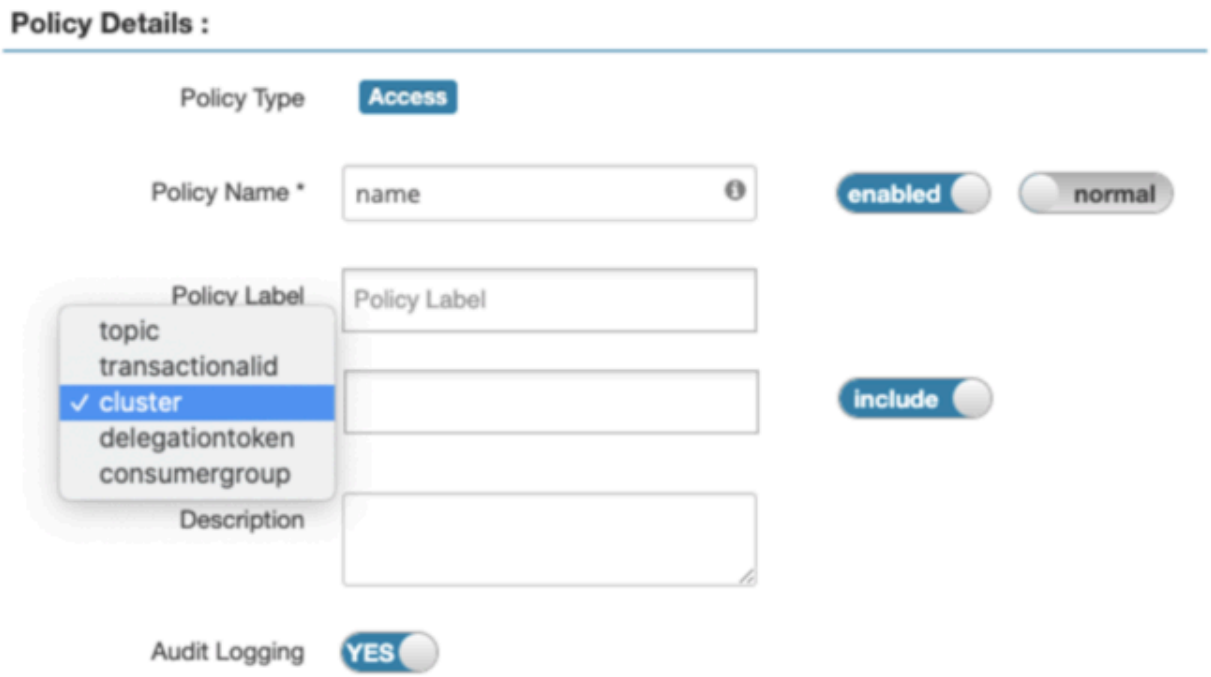

7. Type * in the field beside cluster, and select the * from the values that appear.
8. Go to the Allow Condition section and select the user.
9. Add permissions by clicking the + under Add Permissions.

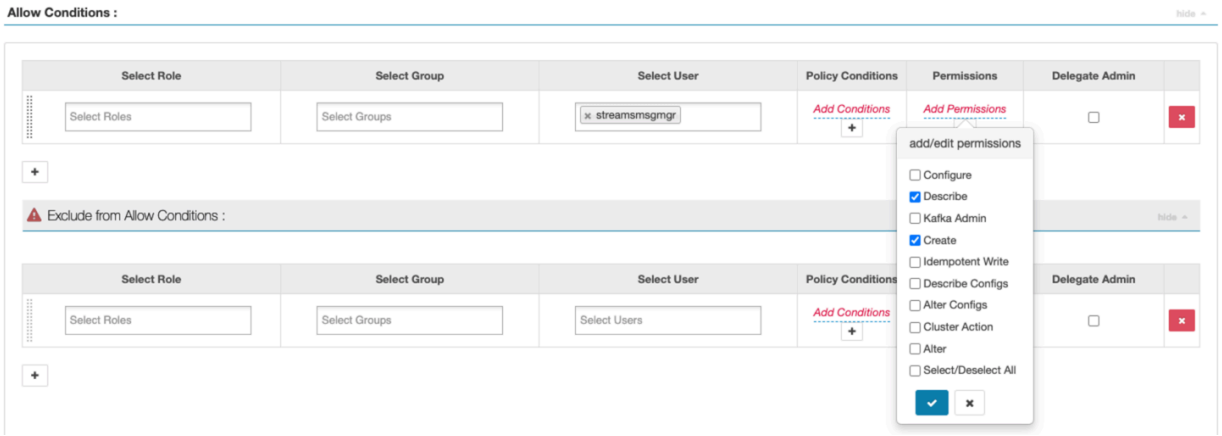

10. Select Create and Describe permissions.
11. Click Add.

Related Information

Configure a resource-based service: Kafka
Enable Kerberos Authentication
Setting up your Streams Messaging cluster

Verifying the setup

After you secure Streams Messaging Manager, you can verify the security setup. You can login to the Streams Messaging Manager web UI and create Kafka topics.

Procedure

1. Go to Cloudera Manager Streams Messaging Manager .

   The login page for Streams Messaging Manager appears.

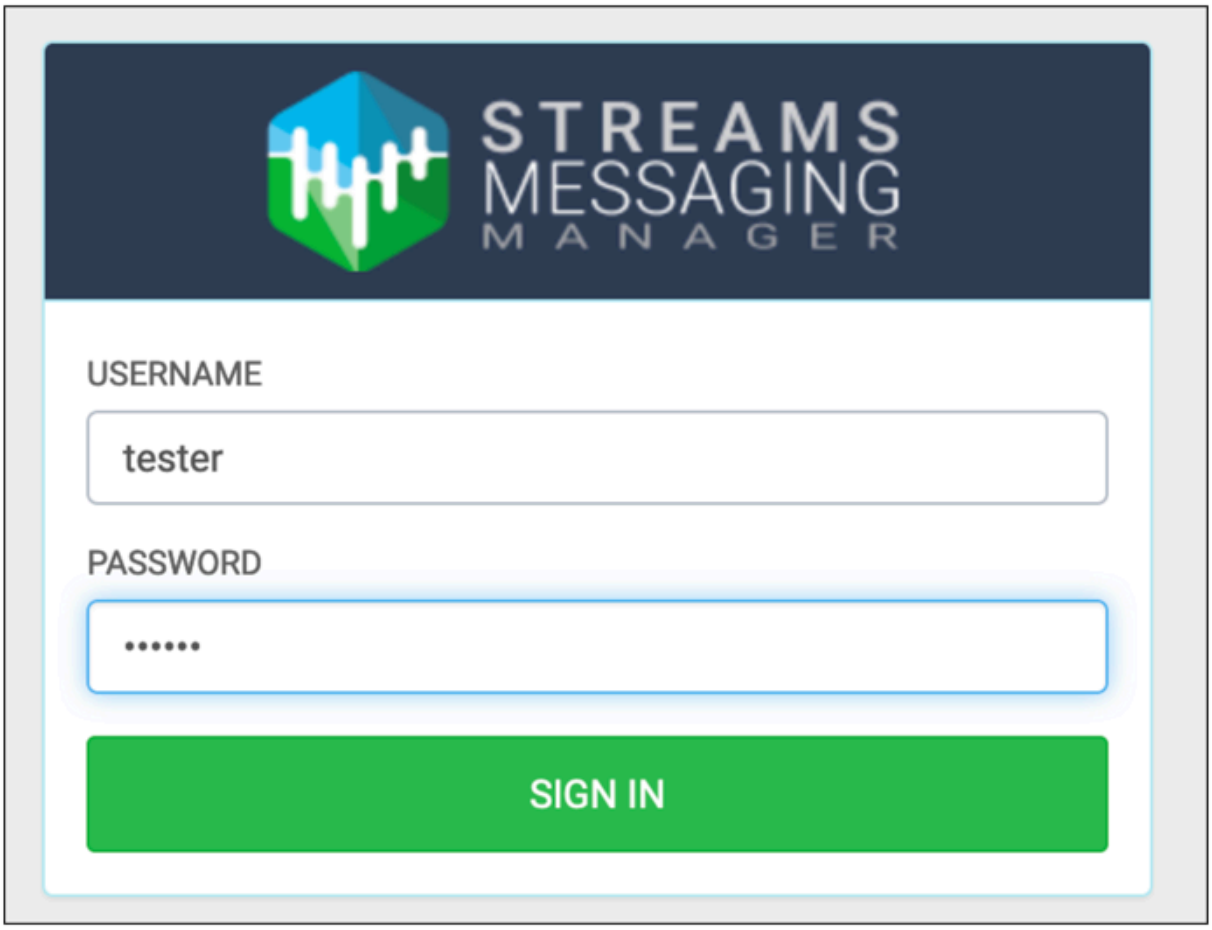

2. Login to the Streams Messaging Manager UI using your regular credentials.

   After you log in, you see the user logout dropdown at the top right corner of your screen. It shows the domain associated with the user.

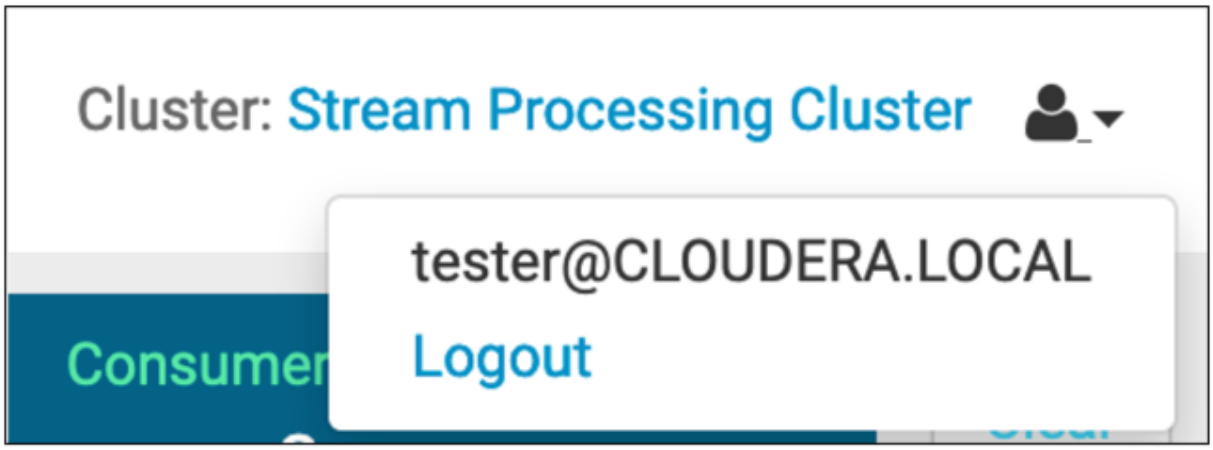

3. Click Streams Messaging Manager Web UI.

4. To add a topic, go to Topics.

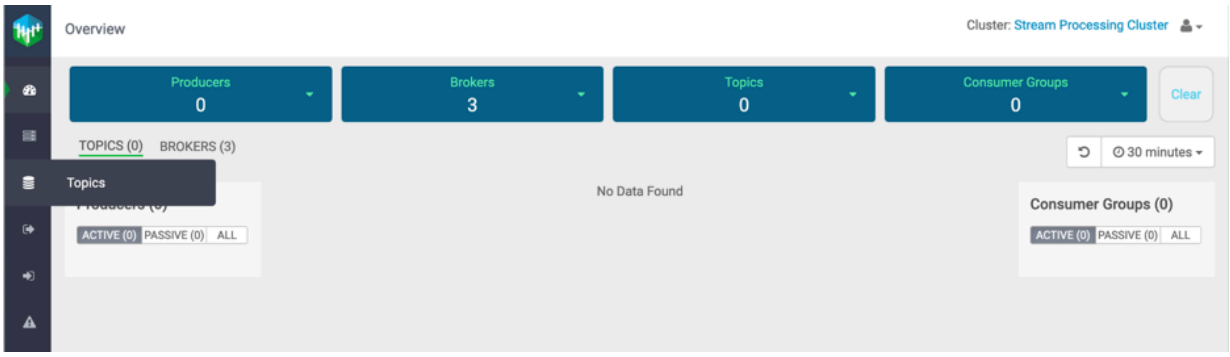

5. Click Add New.

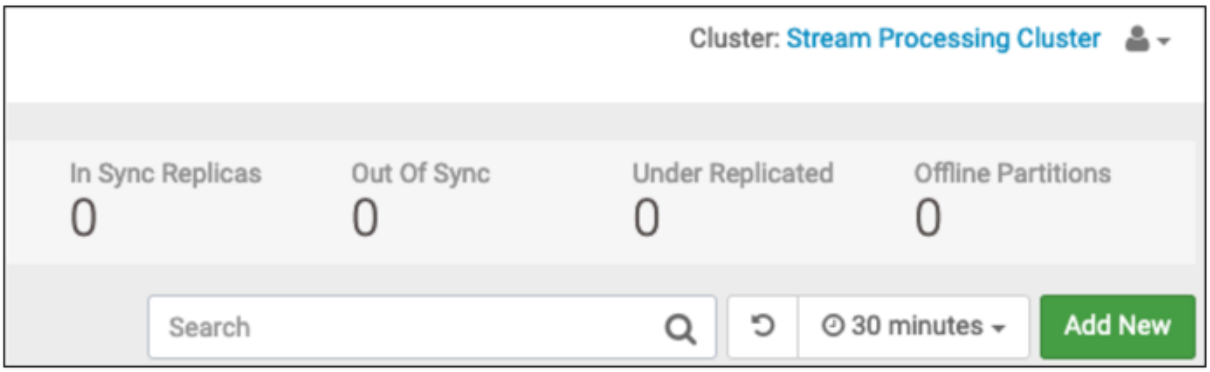

6. Add a topic name, select partitions, and cleanup policy.

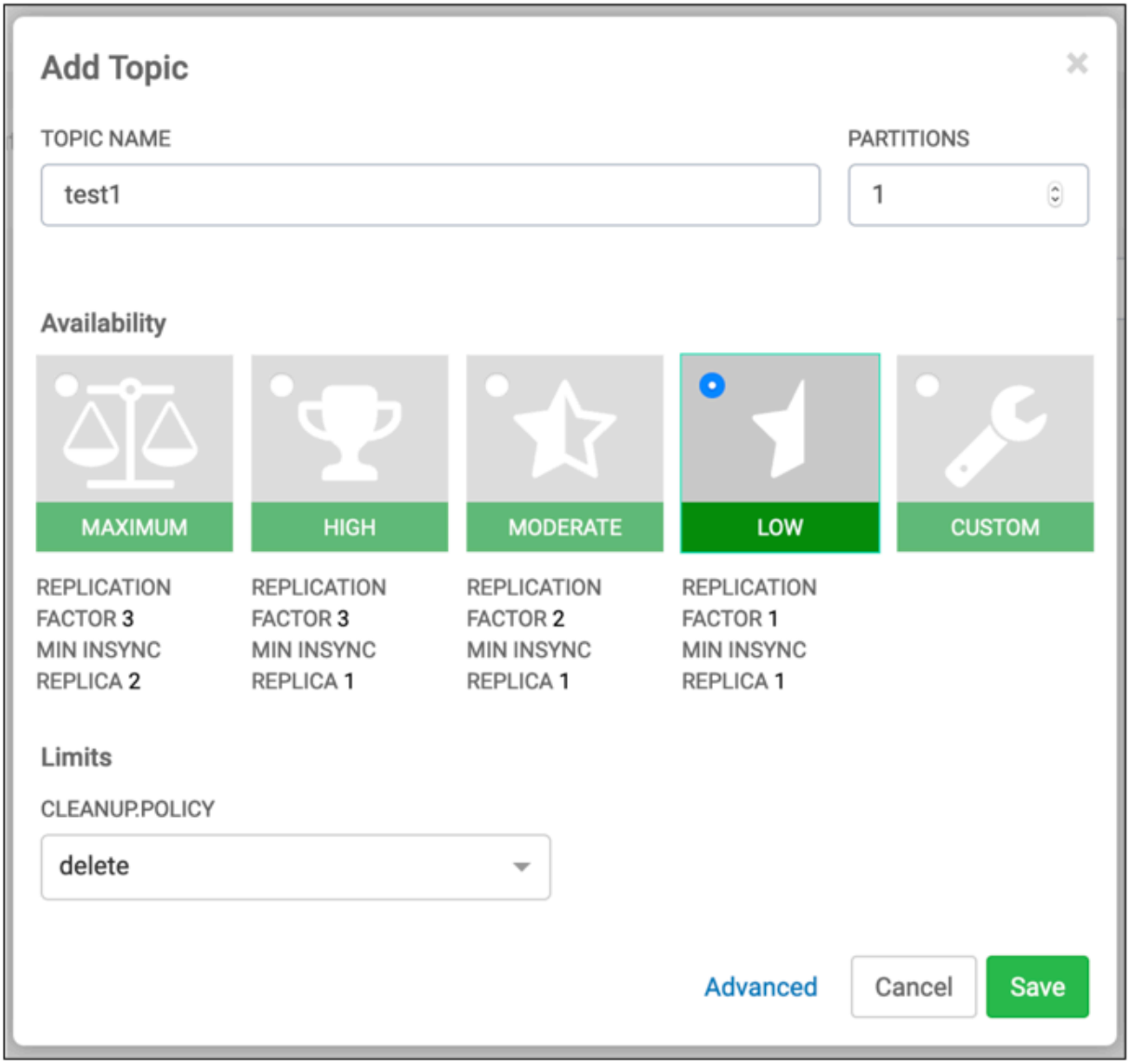

7. Click Save.

   You see the following message in the top right corner of the webpage.

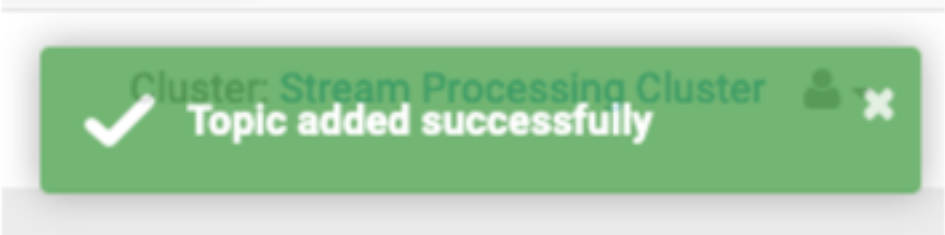

Cloudera Runtime 7.3.1

Using Streams Messaging Manager

Date published: 2023-10-25
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Monitoring Kafka 5
Monitoring Kafka clusters 5
Monitoring Kafka producers 6
Monitoring Kafka topics 9
Monitoring Kafka brokers 15
Monitoring Kafka consumers 17
Monitoring log size information 20
Monitoring lineage information 24

Managing Kafka topics 26
Creating a Kafka topic 26
Modifying a Kafka topic 27
Deleting a Kafka topic 27

Managing Alert Policies and Notifiers 28
Creating a notifier 29
Updating a notifier 30
Deleting a notifier 30
Creating an alert policy 31
Updating an alert policy 32
Enabling an alert policy 32
Disabling an alert policy 33
Deleting an alert policy 33
Component types and metrics for alert policies 33

Monitoring end-to-end latency 37
Enabling interceptors 39
Monitoring end to end latency for Kafka topic 40
End to end latency use case 44

Monitoring Kafka cluster replications (Streams Replication Manager) 49
Viewing Kafka cluster replication details 50
Searching Kafka cluster replications by source 51
Monitoring Kafka cluster replications by quick ranges 51
Monitoring status of the clusters to be replicated 52
Monitoring topics to be replicated 52
Searching by topic name 53
Monitoring throughput for cluster replication 54
Monitoring replication latency for cluster replication 54
Monitoring checkpoint latency for cluster replication 55
Monitoring replication throughput and latency by values 56

Managing and monitoring Kafka Connect using Streams Messaging Manager..... 57
The Kafka Connect UI..... 57
Deploying and managing Kafka Connect connectors in Streams Messaging Manager.....62
Deploying a Kafka Connect connector in Streams Messaging Manager.....62
Pausing, resuming, restarting, and deleting a Kafka Connect connector in Streams Messaging Manager..... 66
Reconfiguring Kafka Connect connectors in Streams Messaging Manager.....67
Connector configuration features in Streams Messaging Manager..... 68

Managing and monitoring Cruise Control rebalance.....73
Authorizing users to access Cruise Control in Streams Messaging Manager.....73
Cruise Control dashboard in Streams Messaging Manager UI.....74
Using the Rebalance Wizard in Cruise Control.....79

Monitoring Kafka

Learn how to monitor Kafka clusters, producers, consumers topics brokers, log size and lineage using Streams Messaging Manager.

Monitoring Kafka clusters

The overview page provides you with tools to see a snapshot of the Kafka cluster you are monitoring. After you select the Kafka cluster to monitor, you can see the total number of producers, brokers, topics, and consumer groups in that cluster. You can also monitor producer and consumer metrics.

Configure Apache Kafka for Streams Messaging Manager

After you have installed and configured Apache Kafka, you must set one configuration parameter to enable Kafka and Streams Messaging Manager to communicate.

1. Select Kafka from your cluster drop-down, and then select the Configuration tab.
2. Ensure that the Enable Producer Metrics check box is selected.

Viewing cluster overview information

You can use the Overview tab to review information about your Kafka cluster. This page gives you information about total number of producers, brokers, topics, and consumer groups. It also provides more detailed metrics about producers and consumers.

Review the Producers, Brokers, Topics, and Consumer Groups information at the top of your page to understand how many of each are contained in your Kafka cluster.

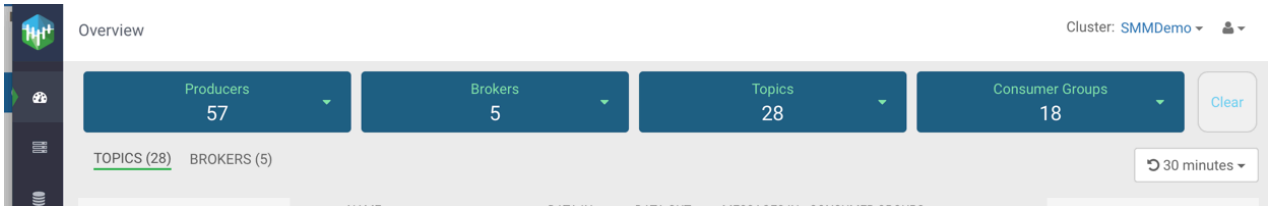

You can click the drop-down arrow in any of the boxes to view a list of Kafka resource. Select one or more Kafka resource to filter your view to just those resource. You can also search for a specific resource. You can click clear at any time to return to the full overview.

Note: Searching supports regular expressions. Invalid regular expressions are interpreted as exact match searches.

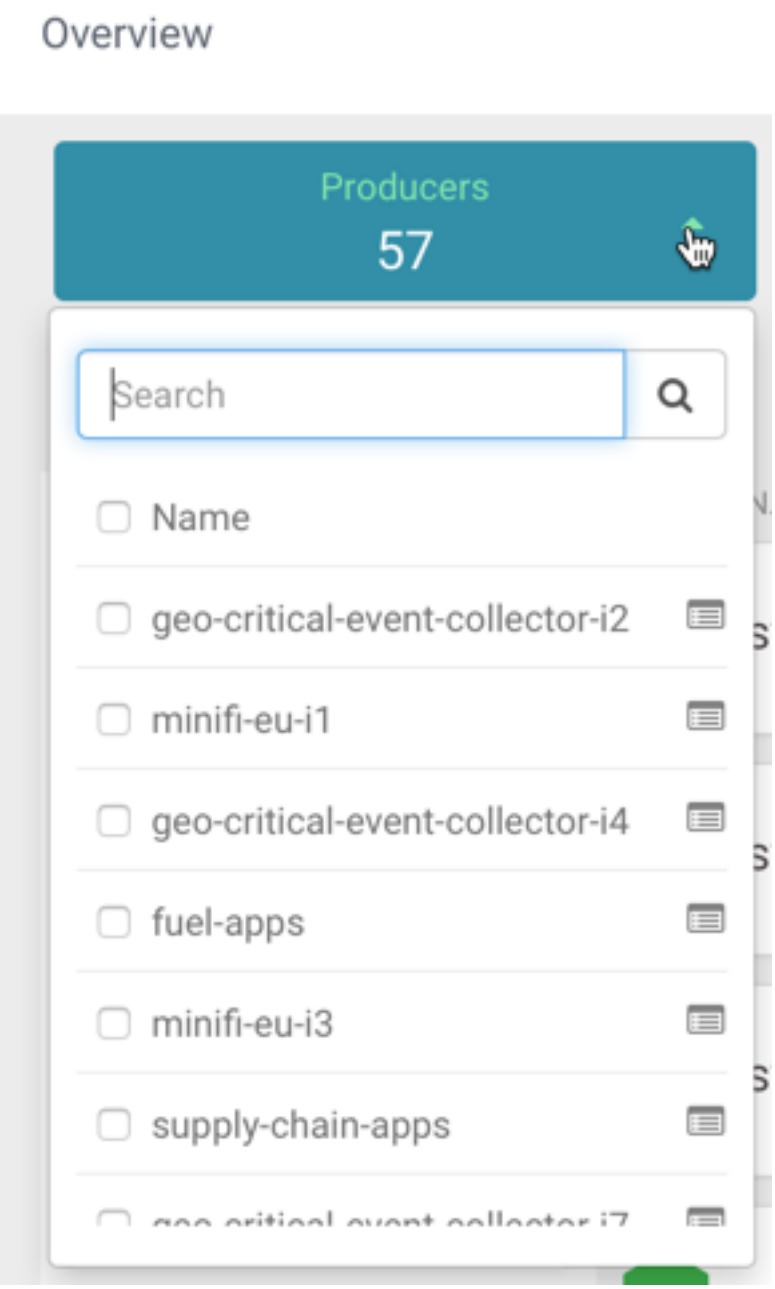

You can select the time period you want to view the metrics for, on the top-right corner of the page. If Cloudera Manager is configured as a metrics backend, the metrics (for example, topic > partition > producermetrics) which are used for time periods larger than 6 hours are calculated asynchronously, and take time to refresh.

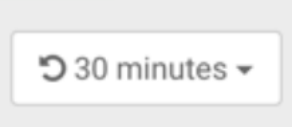

Monitoring Kafka producers

By monitoring Kafka producers, you can track the active and inactive producers in your cluster. You can also change the period of time after which a producer is considered inactive.

Understanding producer naming conventions

The producers you interact with in Streams Messaging Manager are named based on the client.id property you added when creating Kafka producers.

Active vs. passive producers

On the Overview page, producers are referred to as active or passive. Producers are active when they are producing messages over a designated time period.

On the Producers page, passive producers are referred to as inactive.

You can set the period of time after which a producer is considered inactive in the Streams Messaging Manager Configs screen.

1. Select Streams Messaging Manager from the services pane.
2. Click Configs and select Advanced streams-messaging-manager-common from the Advanced tab.
3. Update inactive.producer.timeout.ms to change the period of time after which a producer is considered inactive. This value is specified in milliseconds.

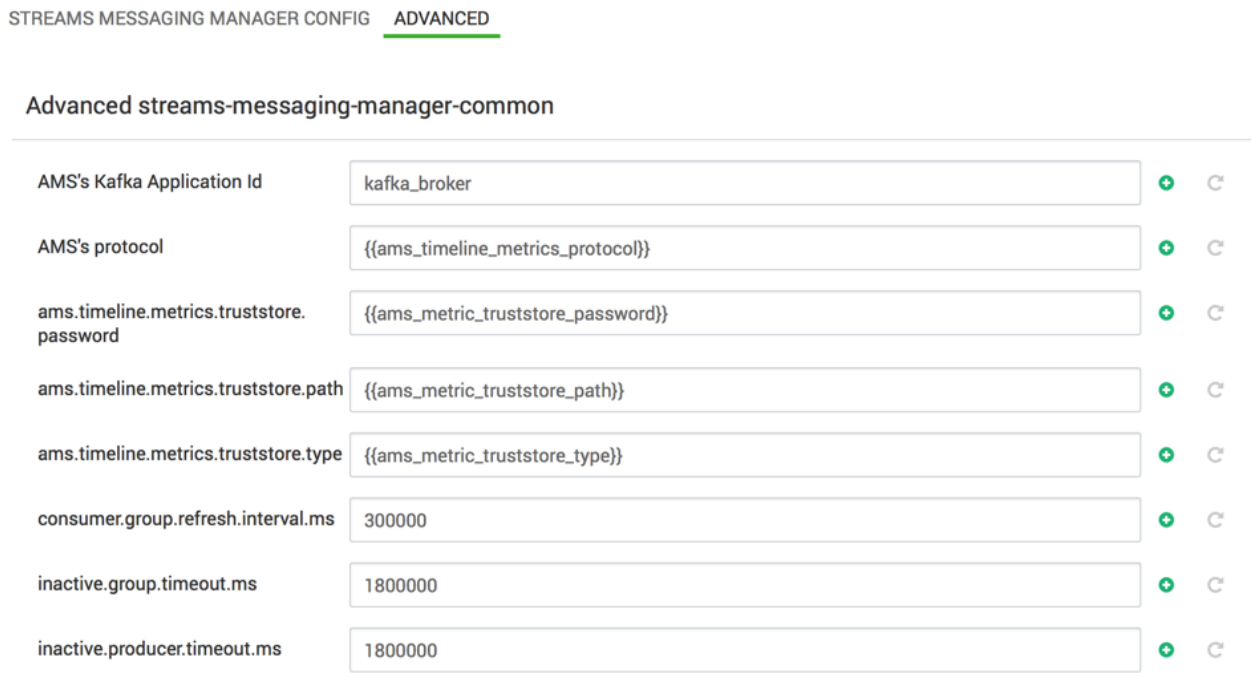

Identifying a producer state

There are two ways to identify whether a producer is active or passive.

From the Producer pane in the Overview page, use the Active, Passive, and All tabs to view only active producers, only passive producers, or all of them. This allows you to see the total number of active and passive producers.

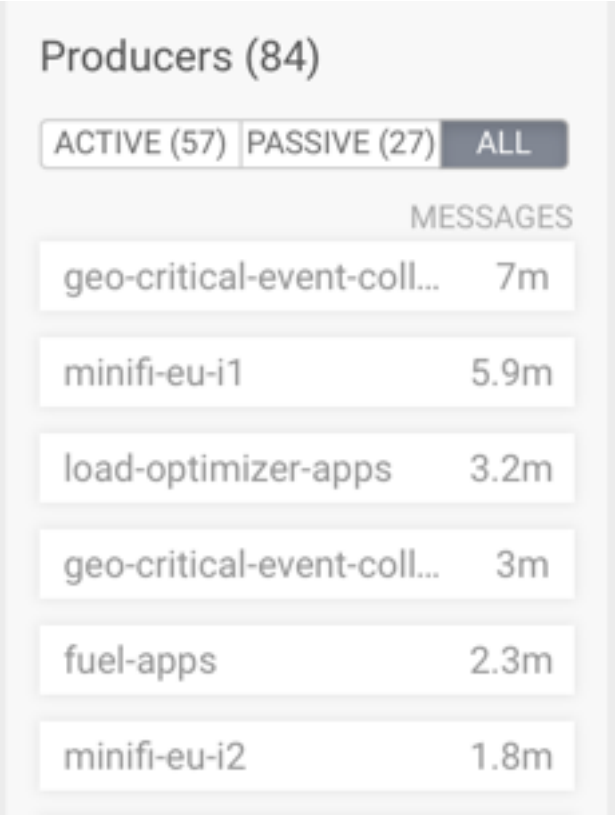

From the Producers page, each producer is listed with the status visible.

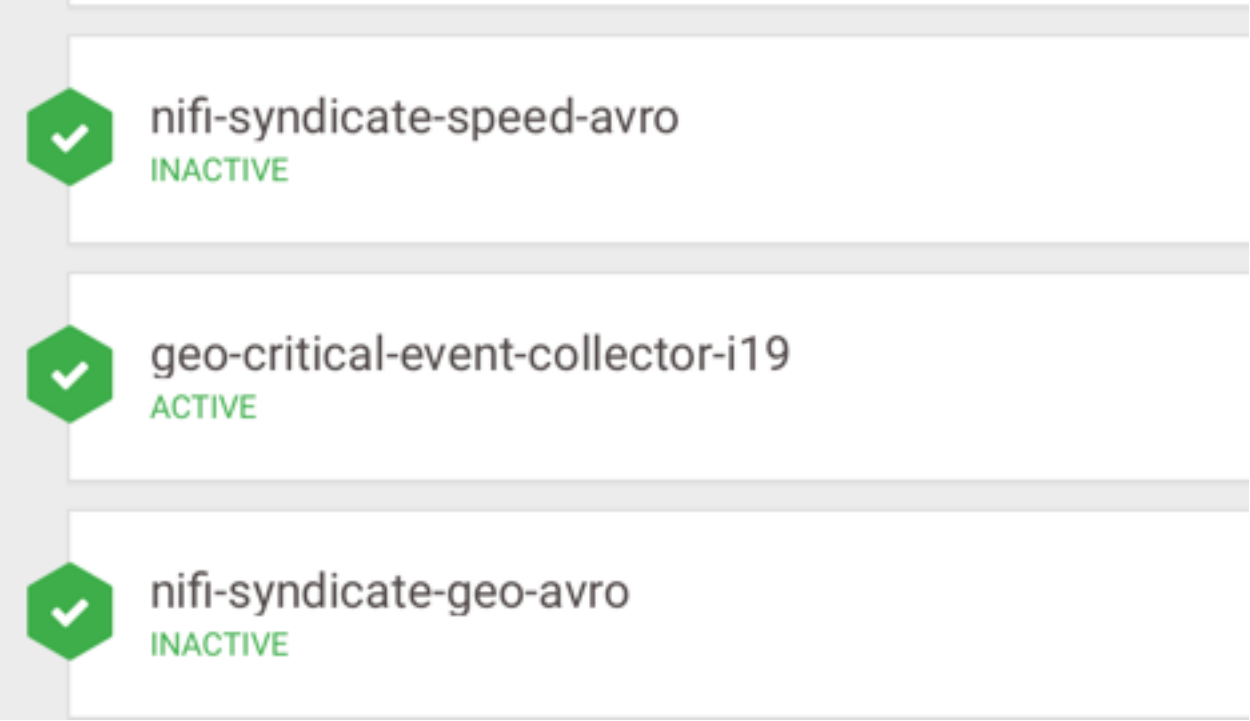

Monitoring Kafka topics

By monitoring Kafka topics, you can track the total number of topics in your cluster and details about the topics. You can also monitor Grafana metrics for the topics in your cluster.

Viewing the total number of topics in your cluster

You can see the total number of topics in your Kafka cluster on the Overview page.

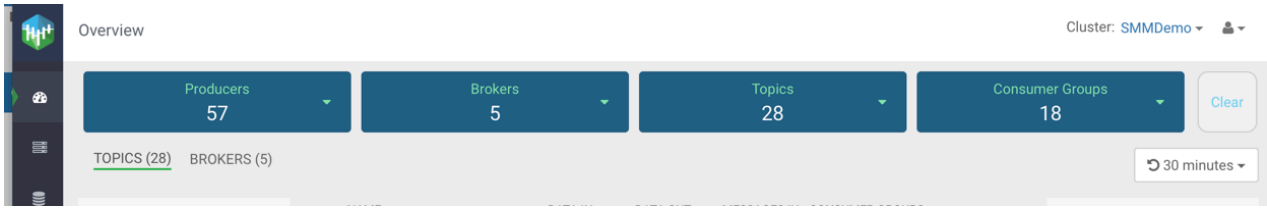

Detailed information about topics

The Topics page contains a number of useful details about your Kafka topics. This page helps you answer the following questions:

- How can I see if the replicas in this topic are in sync?
- How do I see this topic's retention rate?
- How can I see the replication factor for this topic?
- How do I see the producers and consumers that are connected to this topic?
- How do I find the total number of messages going into this topic, over a specified time range?

To access this detailed topic information:

1. From the left navigation pane, click Topics.
2. Identify the topic about which you want information. You can either scroll through the list of topics, or use the Search bar at the top left of the page.
3. Click the green hexagon at the left of the topic to view details.

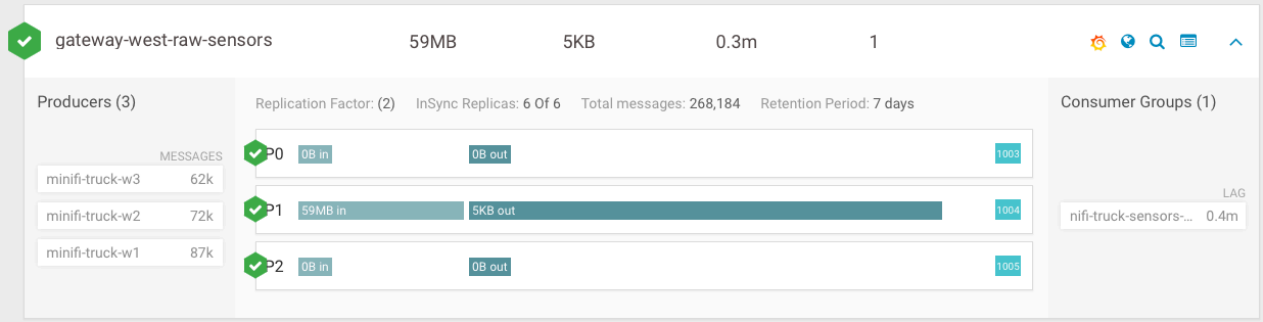

Viewing topic messages using Data Explorer

Data Explorer is a simple Kafka consumer within Streams Messaging Manager. It enables you to view the content of a Kafka topic. You can select any Kafka topic and any partition within that topic, and view messages from the selected partition.

You can reach Data Explorer in two ways. One way is from the **Topics** page, and the other is from the **Overview** page. In both pages, you need to either click the magnifier icon, or navigate to the **Topic Details** page and then select the Data Explorer tab. The following steps describe the process:

1. Log in to the Streams Messaging Manager UI.
2. From the left navigation pane, click Topics.

3. Identify the topic for which you want the message information. You can either scroll through the list of topics, or use the Search bar to find a topic.

4. Click the Data Explorer icon for that topic.

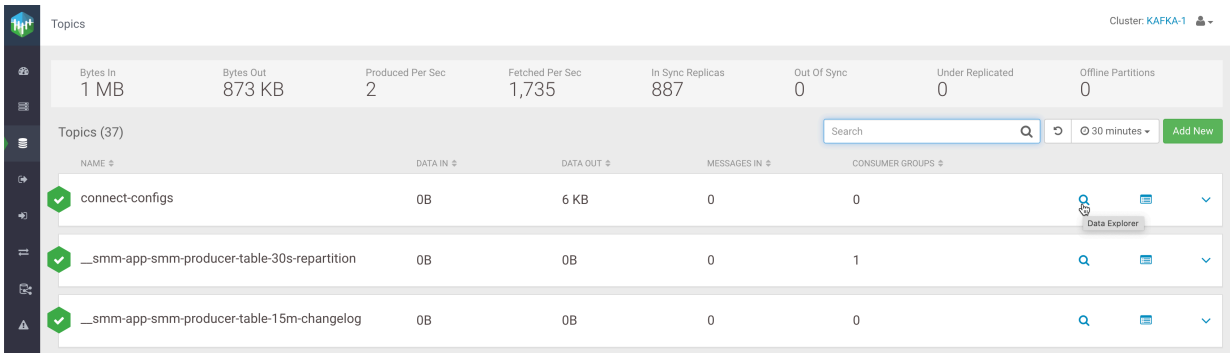

The Data Explorer dialog appears.

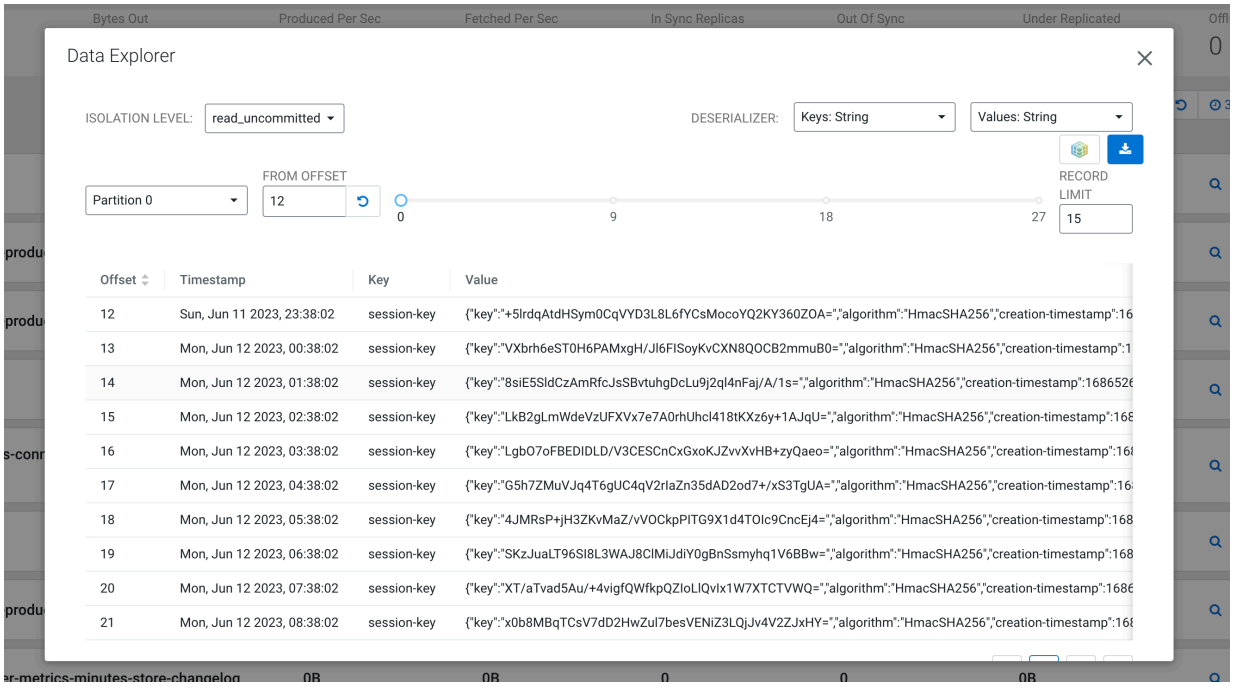

Alternatively, you can click the Profile icon for that topic.

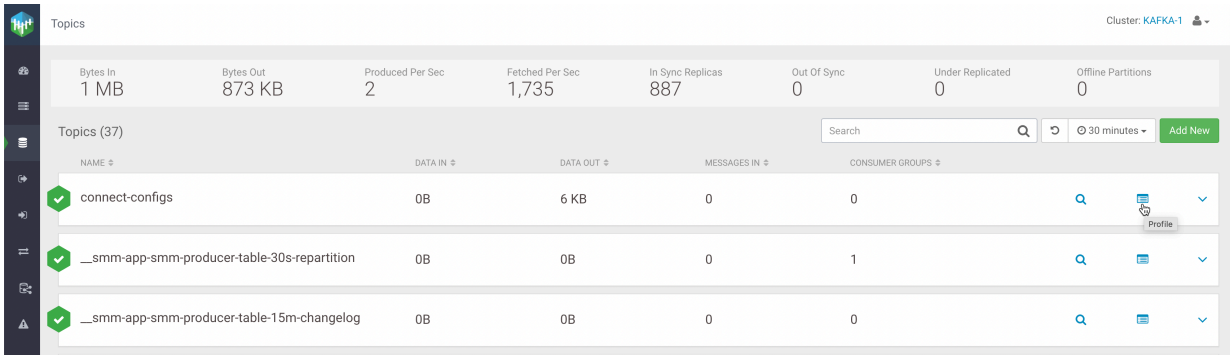

Then go to the Data Explorer tab.

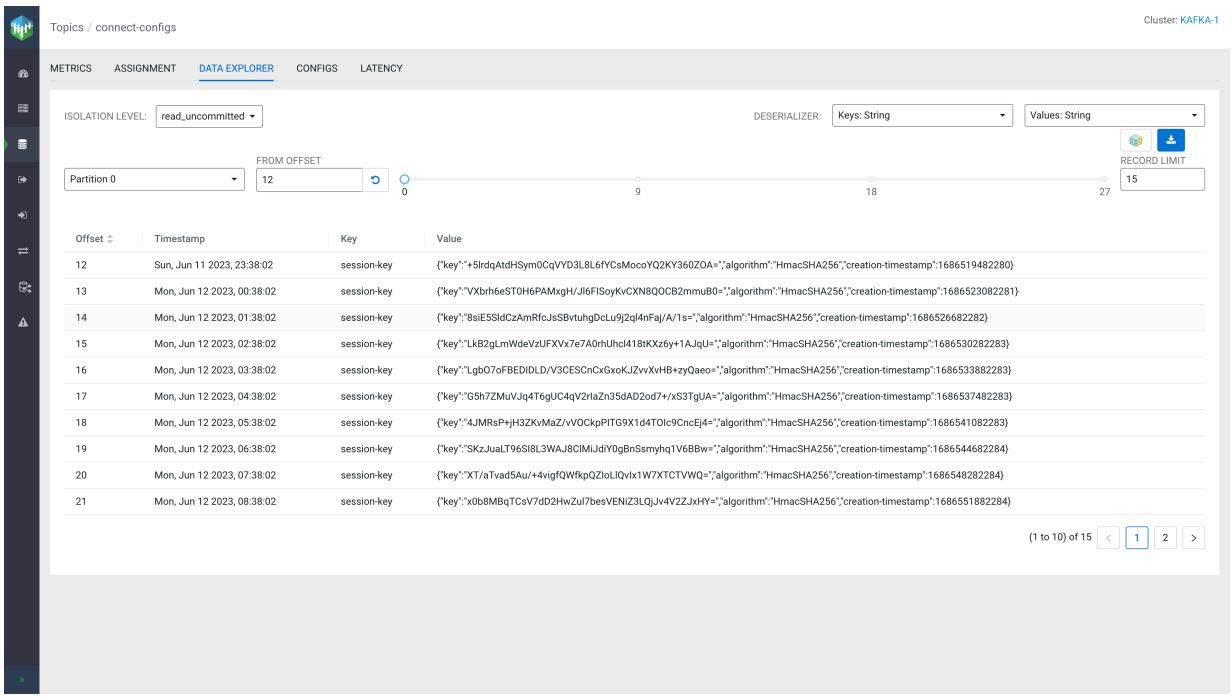

5. Select any of the following modes in the Isolation Level option:

   - read_committed
   - read_uncommitted

   The isolation level specifies whether uncommitted transactional messages should be read. By default, it is set to read_uncommitted.
6. Select the deserializer types for the Keys and Values options.

   For example, if you select Avro, Streams Messaging Manager uses the schema that can be found in Schema Registry to deserialize the messages.
7. Select a Partition.

   The Kafka topic must have partitions to select from.
8. Select a value for the From Offset field.

   You can also use the selection bar to select an offset value. The maximum value is the offset of the last message.

   Click ↺ to refresh the partition offset range and to fetch the latest messages.
9. Select a Record Limit.

   The record limit value is the number of messages that are fetched starting from the message offset number selected in the From Offset field.
10. To see long messages, click show more beside a message.

    The message opens in a dialog or a new tab based on the size of the message.
11. Click the Schema Registry icon to go to the related page in the Schema Registry UI.

Increasing topic partition

You can increase the number of partitions of a topic

Warning: Increasing the partition numbers can impact the use of keys on messages. The data that is already written is not redistributed using the new partition, but remains on the partition on which it was located.

The following steps describe the process:

1. Log in to the Streams Messaging Manager UI.
2. From the left navigation pane, click Topics.
3. Identify the topic for which you want to increase the topic partition number. You can either scroll through the list of topics, or use the Search bar to find a topic.
4. Click the Profile icon for that topic.
5. Go to the Configs tab.

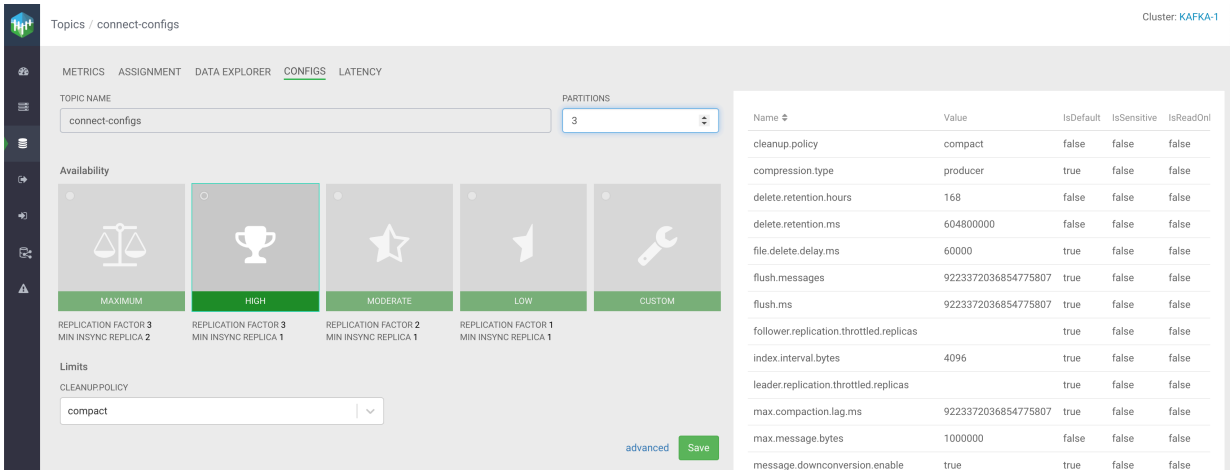

6. Enter the partition number in Partitions or increase it using the arrows.
7. Click Save.

   A dialog appears to ensure the increment.
8. Click Yes.
9. Refresh the page to see the change.

Getting current state of the topic partitions (Tech Preview)

The experimental Assignment tab, on the topic details page, shows the current state of the topic. It shows some topic-level statistics and the replica assignment of all partitions. If rack awareness is used in the Kafka cluster, the replica assignment is shown in a rack-based view. If the rack IDs follow the format of multi-level rack IDs, the rack IDs are rendered as a hierarchy. For more information on rack awareness, see *Kafka rack awareness*.

To go to the Assignment tab, click the Profile icon for a topic from the Overview or Topics page, and then click the Assignment tab. You can view the following statistics there:

- Number of offline partitions

  Shows the number of offline partitions in the topic. A partition is offline if it does not have a leader. Partitions can become offline if all their in-sync replicas are offline.
- Number of under-min-ISR partitions

  Shows the number of under-min-ISR partitions in the topic. A partition is in an under-min-ISR state if the number of in-sync replicas is lower than the value set in the min.insync.replicas property of the topic. The minimum in-sync replicas configuration defines how many replicas must acknowledge a produced message before the produce request is considered successful.

- Number of under-replicated partitions

  Shows the number of under-replicated partitions in the topic. A partition is under-replicated if it has at least one out-of-sync replica.
- Number of unevenly distributed partitions

  This appears if rack awareness is being used.

  Important: This statistics is based on standard rack awareness, and does not respect the multi-level rack IDs. This means that a partition might be shown as evenly distributed while in terms of multi-level rack awareness it should be considered unevenly distributed.

  Shows the number of unevenly distributed partitions in the topic. A partition is unevenly distributed if the difference between the maximum and minimum number of replicas in a rack is greater than one. This typically suggests that the partition does not meet expected durability guarantees or that it causes uneven load on the cluster. If a partition is unevenly distributed, try reassigning them. In most cases, unevenly distributed partitions become evenly distributed across the racks after reassignment.
- Number of unused racks

  This appears if rack awareness is being used.

  Shows the number of racks which are currently not used by this topic. A rack is unused if the topic has no replicas residing in that rack. This typically suggests that the partition does not meet expected durability guarantees because it is not using all available racks (physical locations) to store replicas of the data. This is expected and does not cause issues for non-critical topics that have an intentionally low replication factor.

In the replica assignment table, replicas are shown with different colors:

- Leader replicas are green
- In-sync replicas are blue
- Out-of-sync replicas are red

In addition, offline partitions and out-of-sync partitions are red. Under-replicated partitions, unevenly distributed racks, and unused racks are orange. When a specific partition or rack has one of these issues, a warning sign appears next to it.

Leader and in-sync replicas also act as links to the corresponding broker details page.

In the table header and in the first column, warning icons are shown if the specific column or row is affected by one of the issues listed in the topic statistics.

The following image shows the Assignment tab without racks:

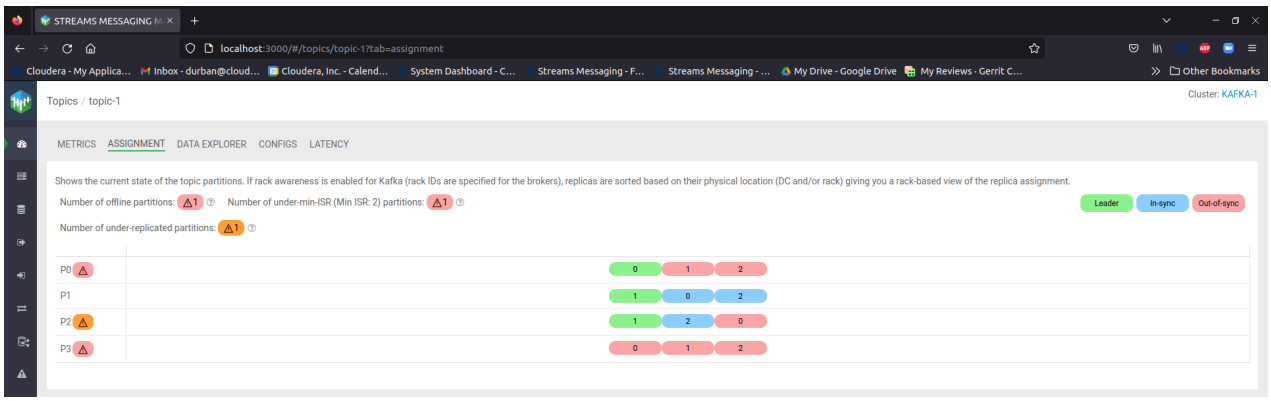

The following image shows the Assignment tab with racks:

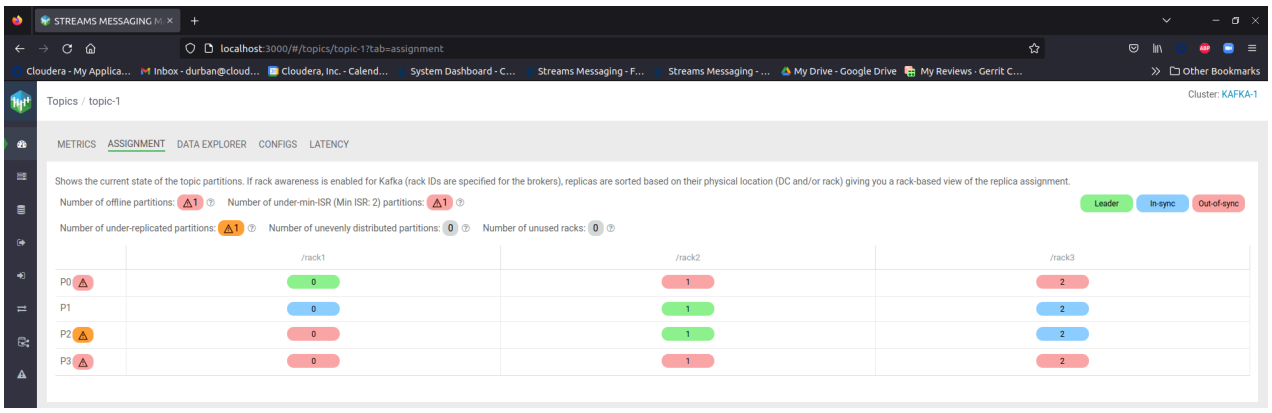

The following image shows the Assignment tab with racks, but with one broker metadata unknown, which adds an extra Unknown column to the table:

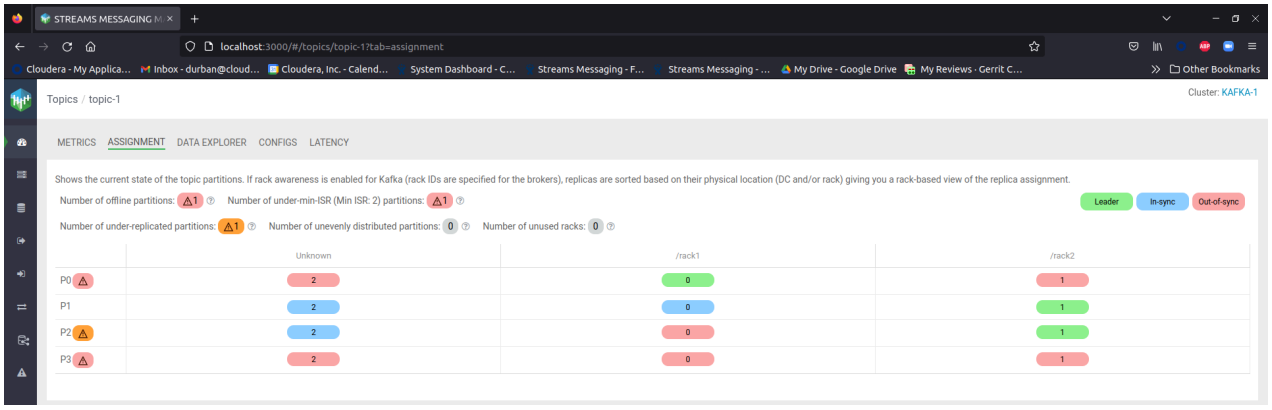

Related Information

Kafka rack awareness

Monitoring Kafka brokers

By monitoring Kafka brokers, you can track various details about brokers including the host where the broker is located, disk space used by the broker, throughput, messages coming in, partitions, and replicas.

Detailed broker information

The Brokers page contains a number of useful details about your Kafka brokers. This page helps you answer the following questions:

- On what host is my broker located?
- Is my broker running out of disk space?

To access detailed broker information:

1. From the left navigation pane, click Brokers.
2. Identify the broker about which you want information. You can either scroll through the list of brokers, or use the Search bar at the top left of the page.
3. Click the green hexagon at the left of the broker to view details.

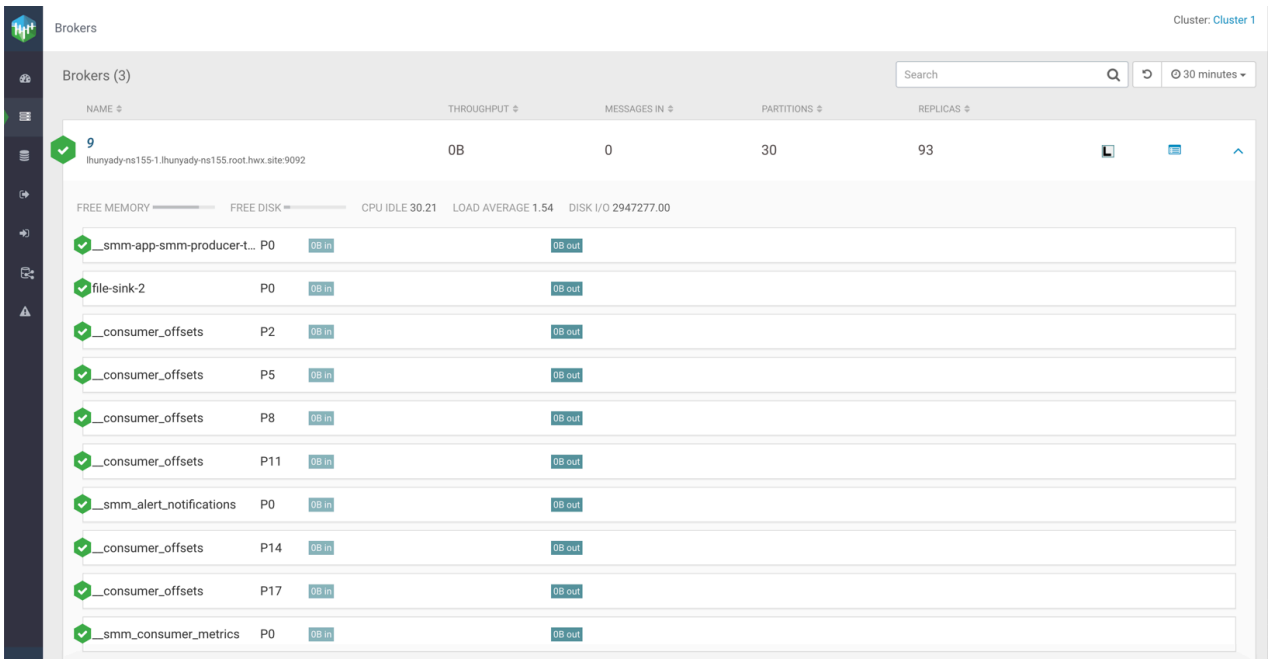

Viewing additional details about the broker host

You can view additional details about the broker host from Cloudera Manager. To access this information perform the following steps:

1. From the left navigation pane, click Brokers.
2. Identify the broker about which you want information. You can either scroll through the list of brokers, or use the Search bar at the top left of the page.
3. Click the Profile icon on the right side of the broker view.

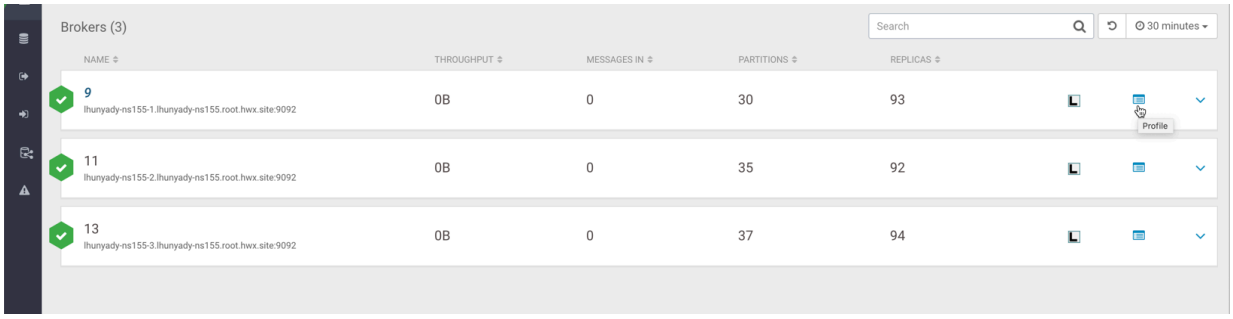

4. Click the Cloudera Manager icon on the right side of the header.

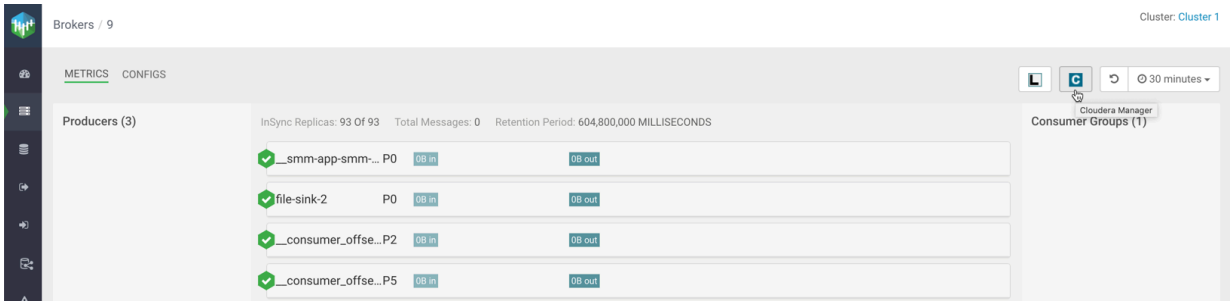

You can track additional details about the broker host in the Metrics and Configs tabs.

Monitoring Kafka consumers

By monitoring Kafka consumer groups, you can track active and passive consumer groups, or all consumer groups, which use the default internal __consumer_offsets topic to store the consumed offset information. You can track additional details about consumer groups. You can also track details including number of consumers and consumer instances included in a group and consumer group lag in the consumer group profile.

Streams Messaging Manager displays consumer groups that have offsets stored in Kafka's internal topic __consumer_offsets, which is also the default store if the auto.commit.enable property is set to true for consumers. Streams Messaging Manager does not display consumer groups that have offsets stored anywhere else other than this default store.

Viewing summary information about consumer groups

The Overview page gives you summary information about consumer groups on the right side of the page. You can use the Active, Passive, and All tabs to view consumer groups only in the Active or Passives, or all of the consumer groups, which use the default internal __consumer_offsets topic to store the consumed offset information. Use the Lag tab to sort consumer groups based on ascending or descending amounts of lag.

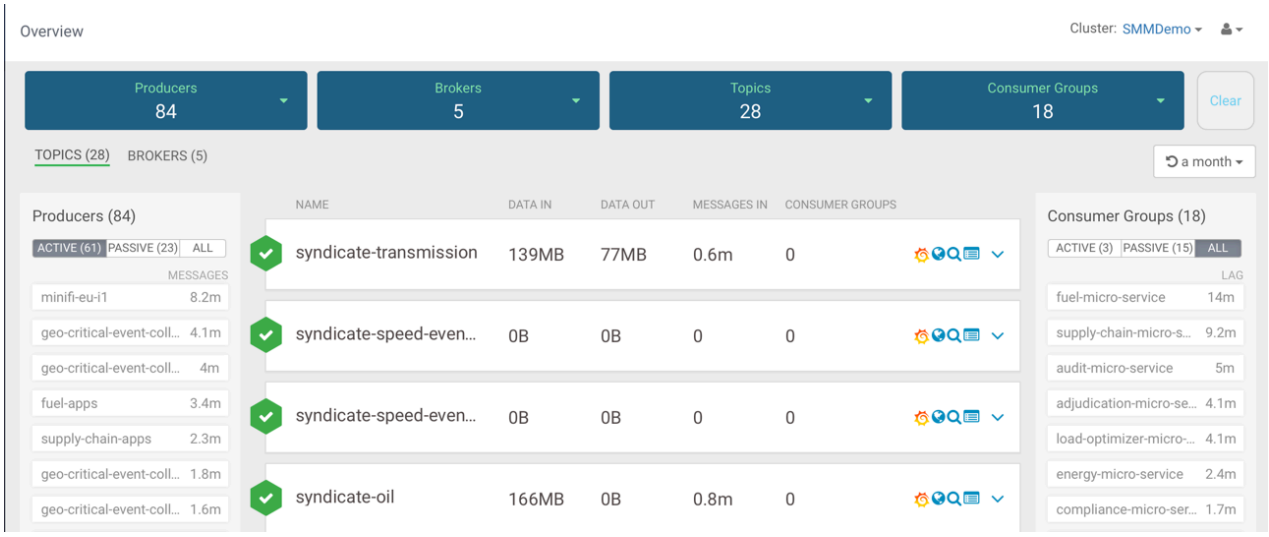

Viewing details about a consumer group

To access detailed consumer group information:

1. From the left navigation pane, click Consumer Group.
2. Identify the consumer group about which you want information. You can either scroll through the list of consumer groups, or use the Search bar at the top left of the page.
3. Click the green hexagon at the left of the consumer group to view details.

Viewing the consumer group profile

The Consumer Group profile displays detailed information about each consumer group, including:

- The number of consumers included in the group.
- The number of consumer instances in the group.
- Details about consumer group lag.

To access the Consumer Group profile:

1. From the Consumer Group page, select the consumer group for which you want to view the profile.
2. Click the profile icon in the upper right of the Consumer Group tile.

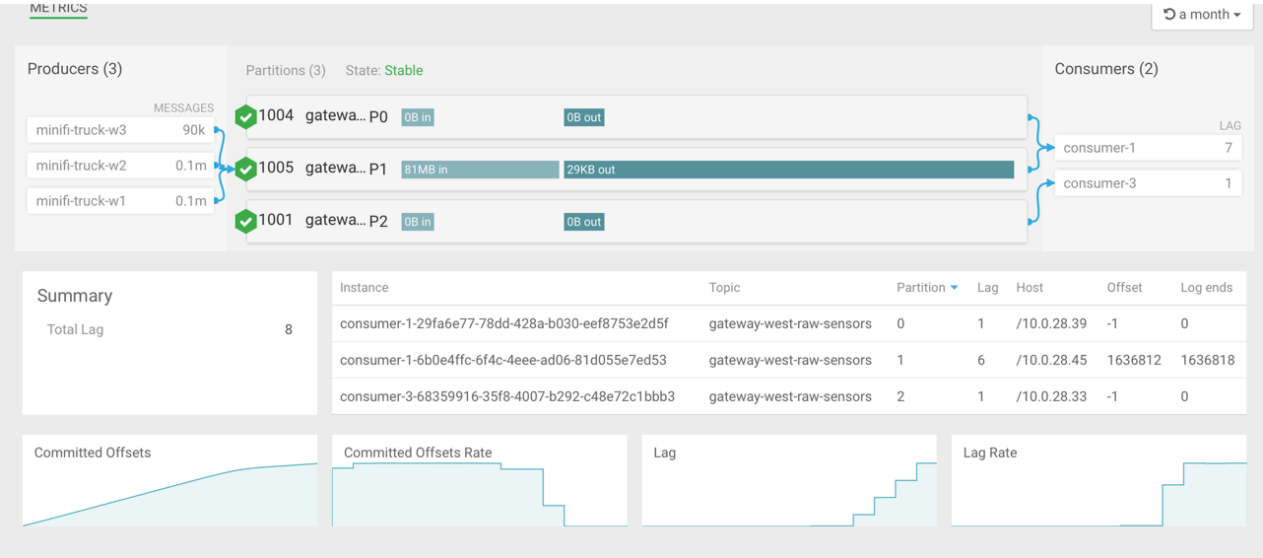

Resetting consumer offset

To reset the offset of a consumer group, perform the following steps:

1. From the left navigation pane, click Consumer Group.
2. Choose the consumer group for which you want to reset the offset, and click the Profile icon.

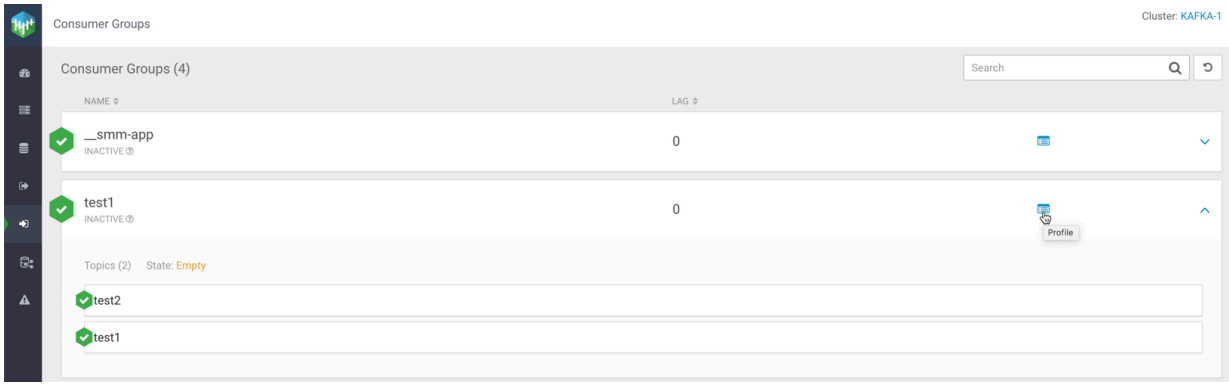

Note: Resetting offsets is only possible for those groups whose state is Empty or Dead. Attempting to reset offset of any other group results in an error.

3. In the Metrics page, click Actions Reset offset .

   The Reset offsets for consumer group dialog appears.

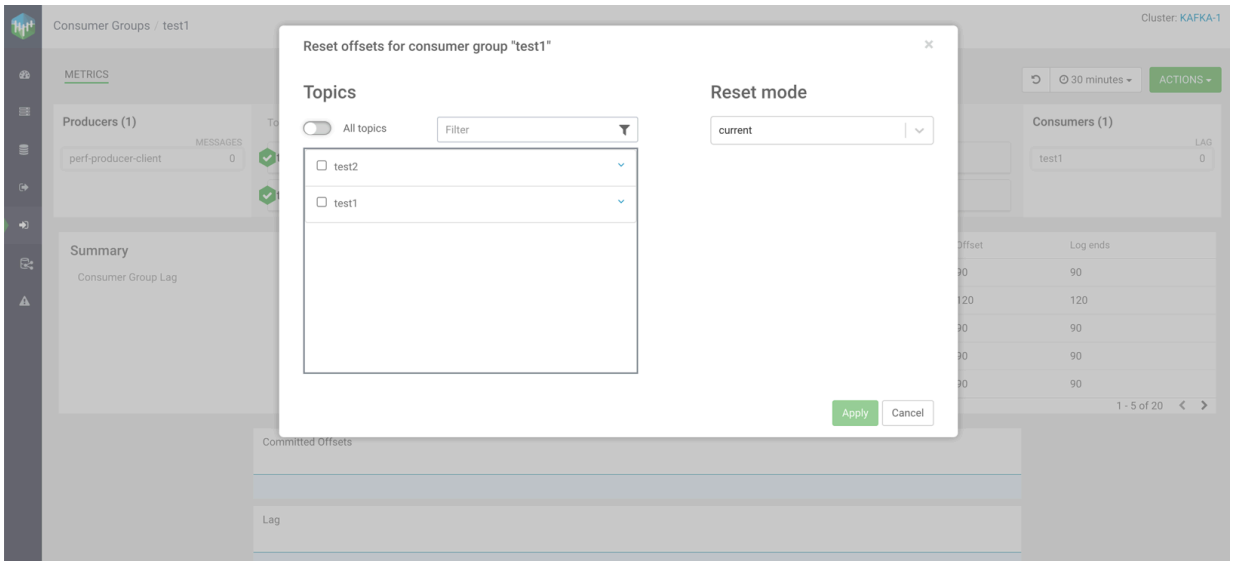

4. Select the topic and partition(s) you want to reset.

   You can use the arrow beside each topic to display the partitions of that topic, and select partition(s) as required. You can also select all topics and all related partitions by selecting the All topics option. You can use the Filter option to find a specific topic.

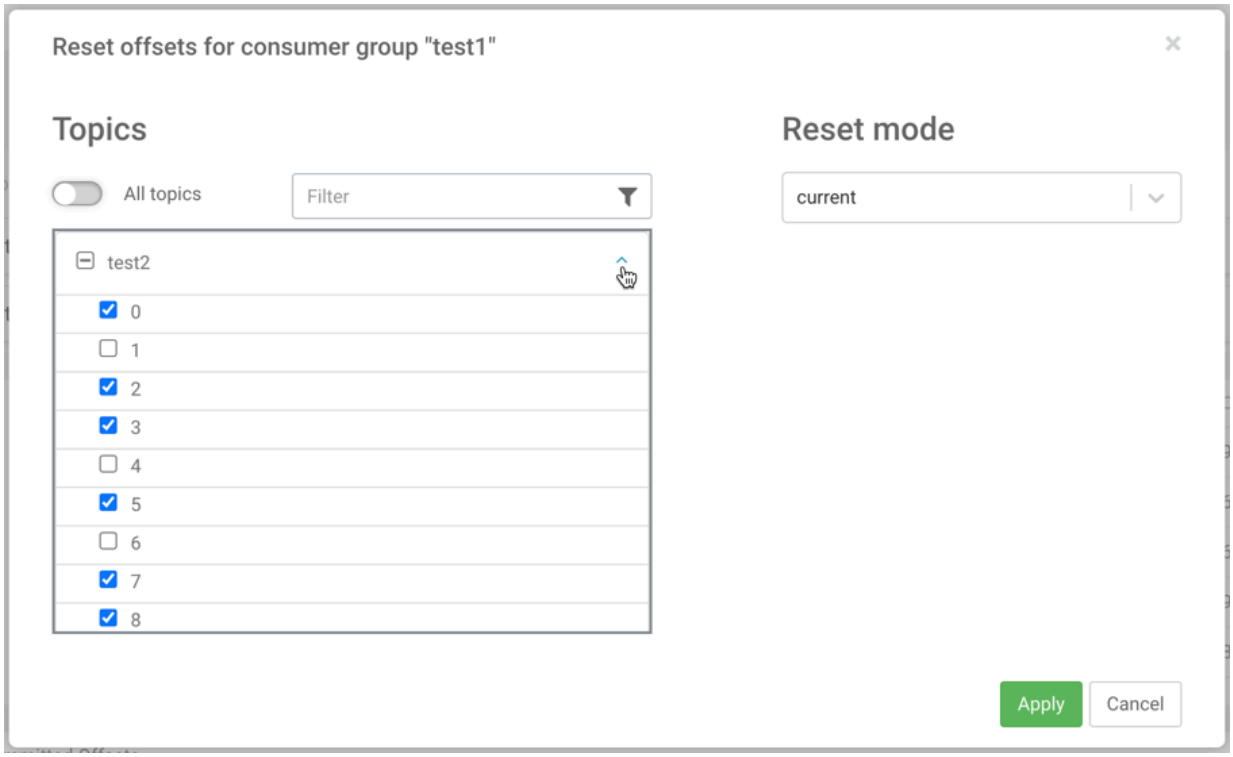

5. Select the reset option from the Reset mode.

   The available options are as shown in the following image:

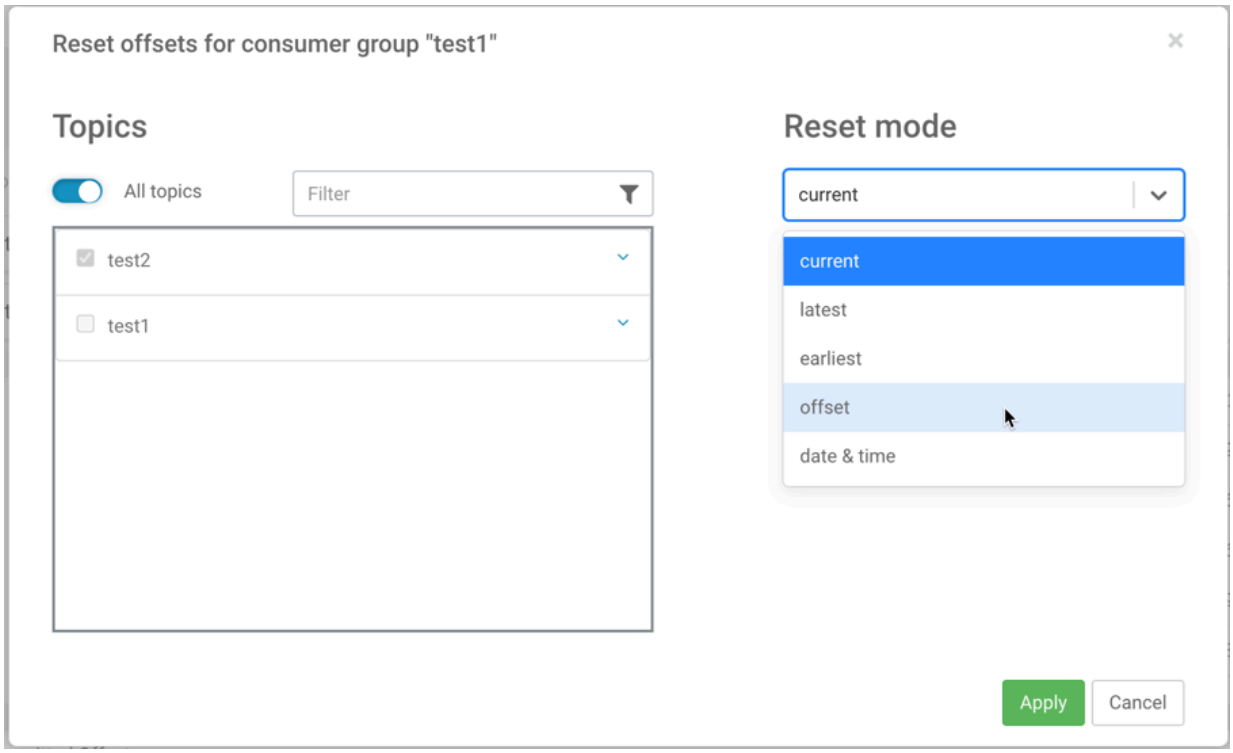

6. Click Apply.

Monitoring log size information

The Streams Messaging Manager UI shows log size related information about brokers, topics, and partitions. Furthermore, warning messages appear when log directory related errors happen.

The feature uses a cache mechanism. The following cache attributes are set by default. You can modify these attributes in the Streams Messaging Manager configuration page of the Cloudera Manager UI:

- Kafka Logdir Cache Expiration ms

  Sets cache expiration time.
- Kafka Logdir Cache Maximum Size

  Sets maximum cache size.
- Kafka Logdir Call Timeout ms

  Sets timeout of the Kafka query.

The following image shows the properties in Cloudera Manager:

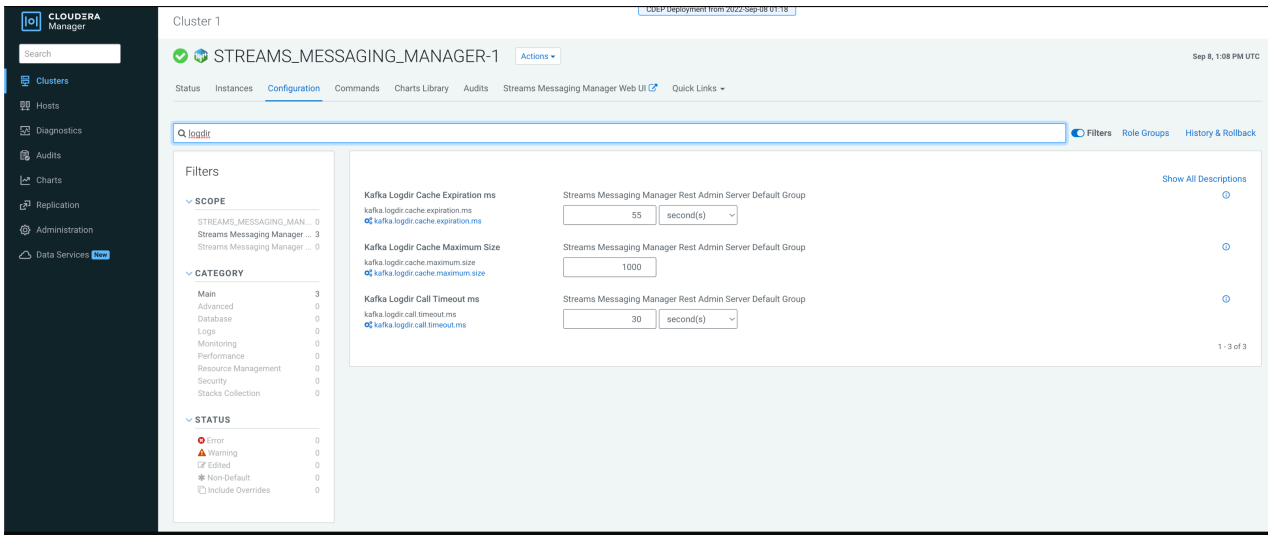

Log size details for topics

On the Overview page of the Streams Messaging Manager UI, the Topics tab contains a sortable column called Current Log Size which shows the data with the measure unit. This log size is calculated by the leader partitions; so, the follower partitions are not included. The column contains a tooltip which is a question-mark symbol. The topic related log-size improvements or information are also available on the Topics page.

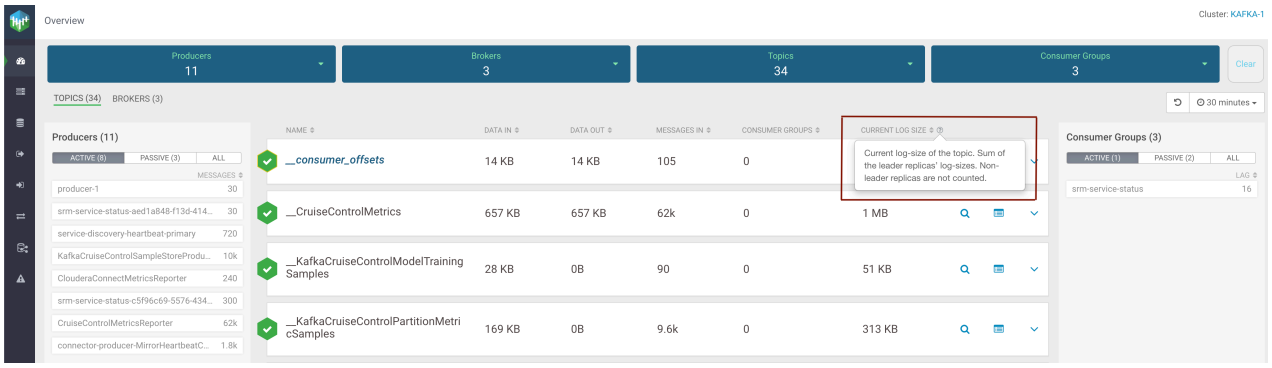

The Topics tab on the Overview page represents topics and you can select one of them. In this case the topic's leader partitions are viewable and here you can find the leader partition log-size. If you click a partition, a pop-up window appears, where log-size information is displayed.

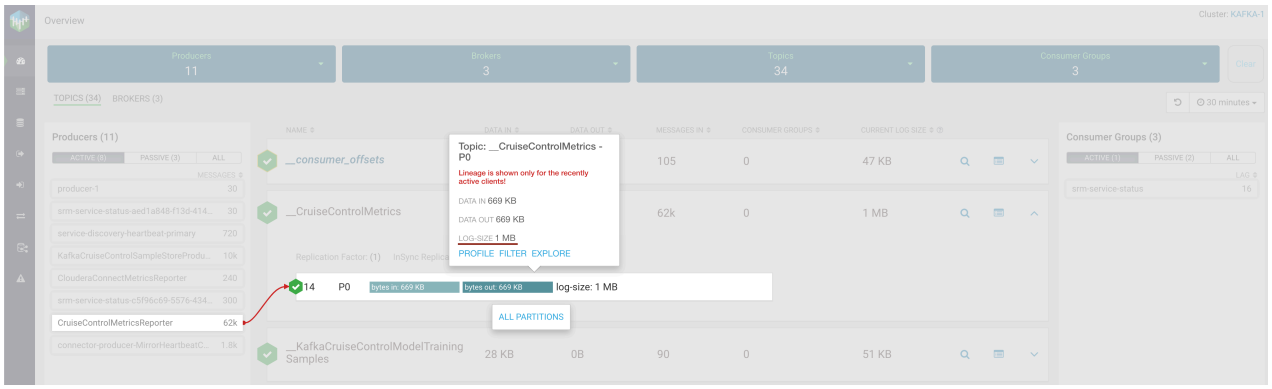

If you click Profile for a topic, log-size of replica-leaders appears, hosted by the broker.

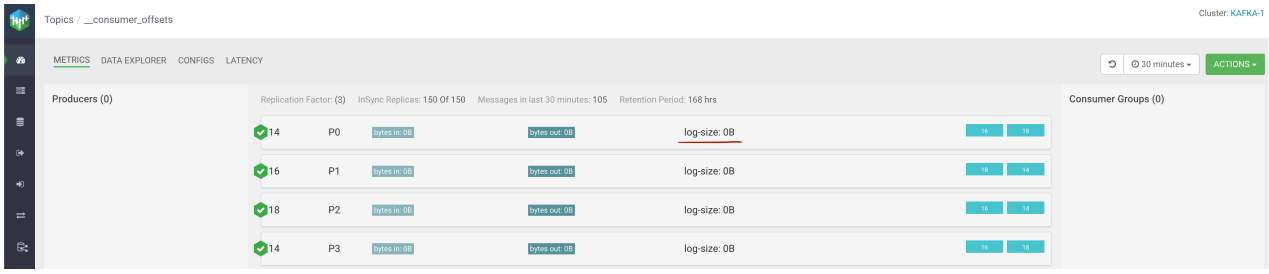

Log size details for brokers

On the Overview page, the Brokers tab contains two sortable columns called Log Size and Remaining Storage. The log-size is calculated for all the partitions including the follower partitions. The remaining storage column contains the available capacity in all of the configured log directories. The broker related log-size improvements or information are also available on the Brokers page.

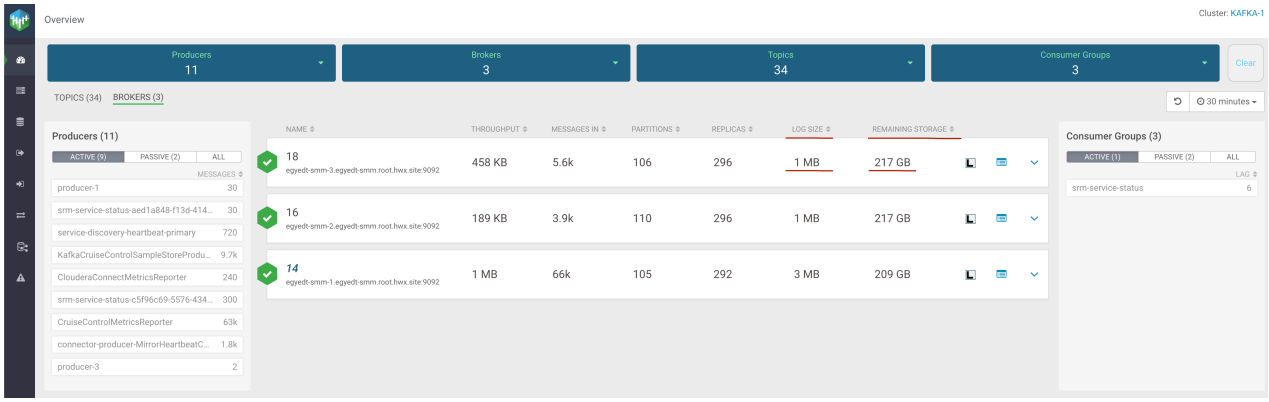

If you click the arrow icon for a broker, the details of that broker are displayed. The Leader Partition Log-Size shows the sum of leader partitions' log-size. The All Partitions Log-Size field's value is equal to the sortable column's value called Log Size (naming is different to make the sortable column names as short as possible). The Remaining Storage sortable column and field is the same (value and name).

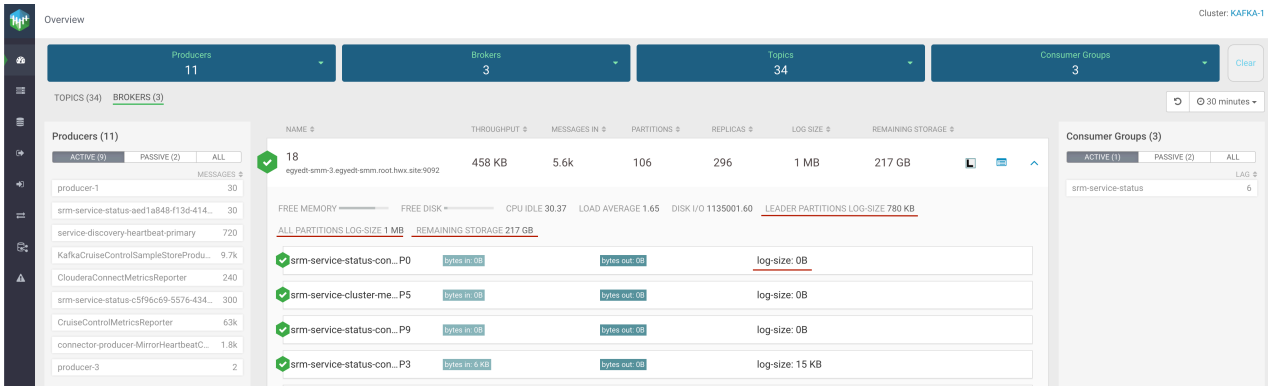

If you click a specific partition, a pop-up window appears where log-size information is displayed.

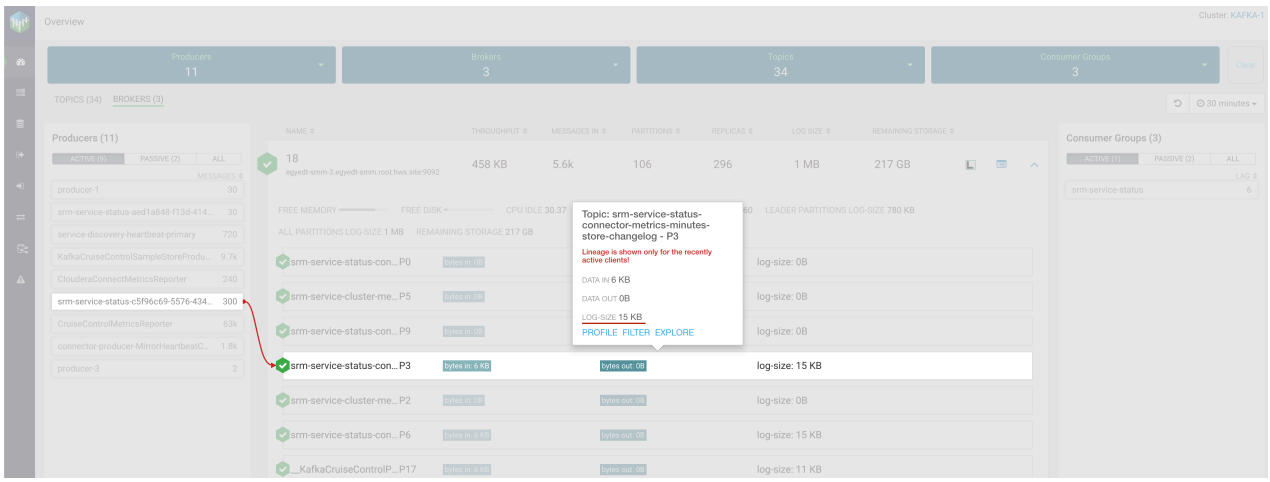

If you click Profile of a broker, then the metrics details for that broker are displayed. The log-size information per partition is available. Furthermore, the sum of leader partitions' log-size, remaining storage size, and all partitions log-size details are also available. The Metrics tab on the Brokers page contains the available storage size per log directory.

Note: There is no advantage of creating more log directories on the same disk, so this should be avoided.

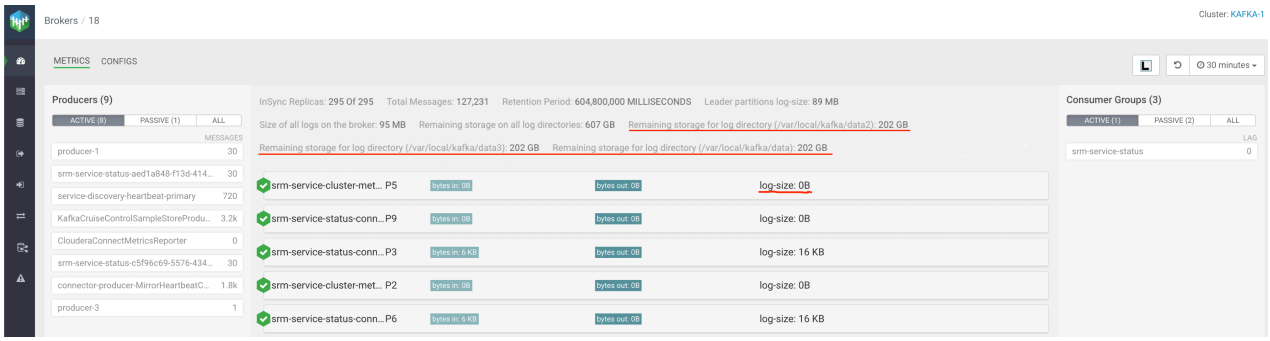

If there is any issue with the query on the log directory information, then a warning message appears. There can be multiple warnings if more than one broker is related. But no new warning message from a specific broker appears until the previous message disappears.

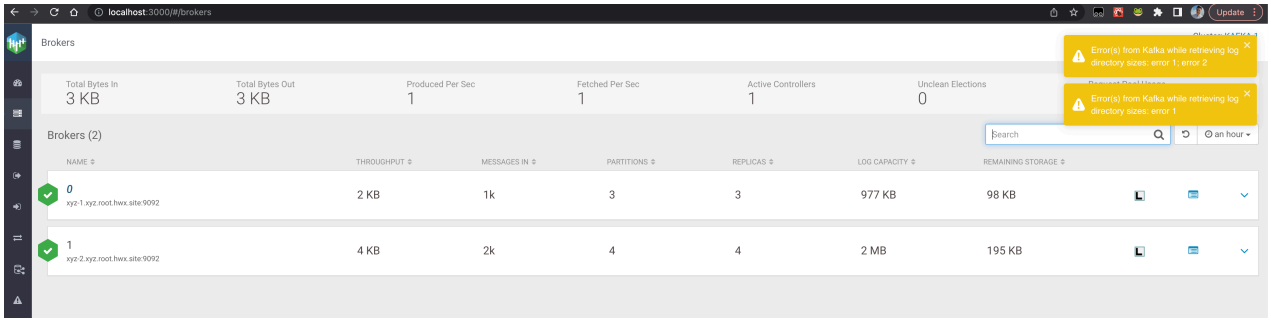

Cloudera Manager chart builder

For historical information, you can use the chart builder provided in Cloudera Manager. The following command and screenshot show Kafka log-size data from the past where the broker ID and topic name with the topic partition are specified.

```
SELECT counter_sum(kafka_bytes_received_by_partition_rate) WHERE serviceName
=KafkaServiceName AND kafkaBrokerId = KafkaBrokerId AND kafkaTopicName = Kaf
kaTopicName AND kafkaPartitionId = kafkapartitionId
```

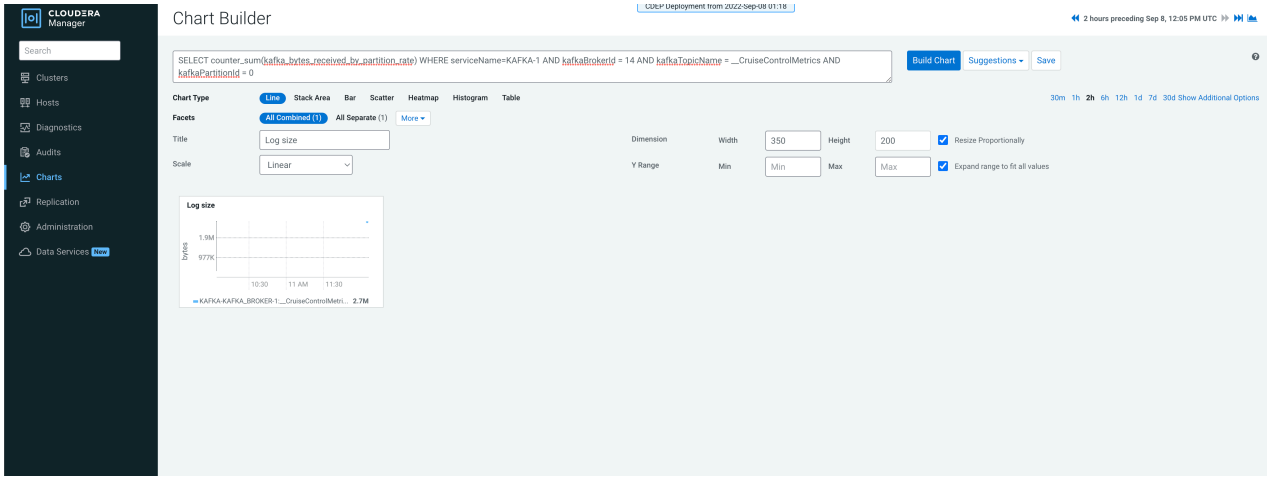

Monitoring lineage information

Learn how you can visualize the lineage between producers and consumers.

To check which topics a producer is producing to, and which consumers consume from those topics, go to the Overview page and click on a single producer on the Producer pane. For example, click producer_1, as shown in the following image:

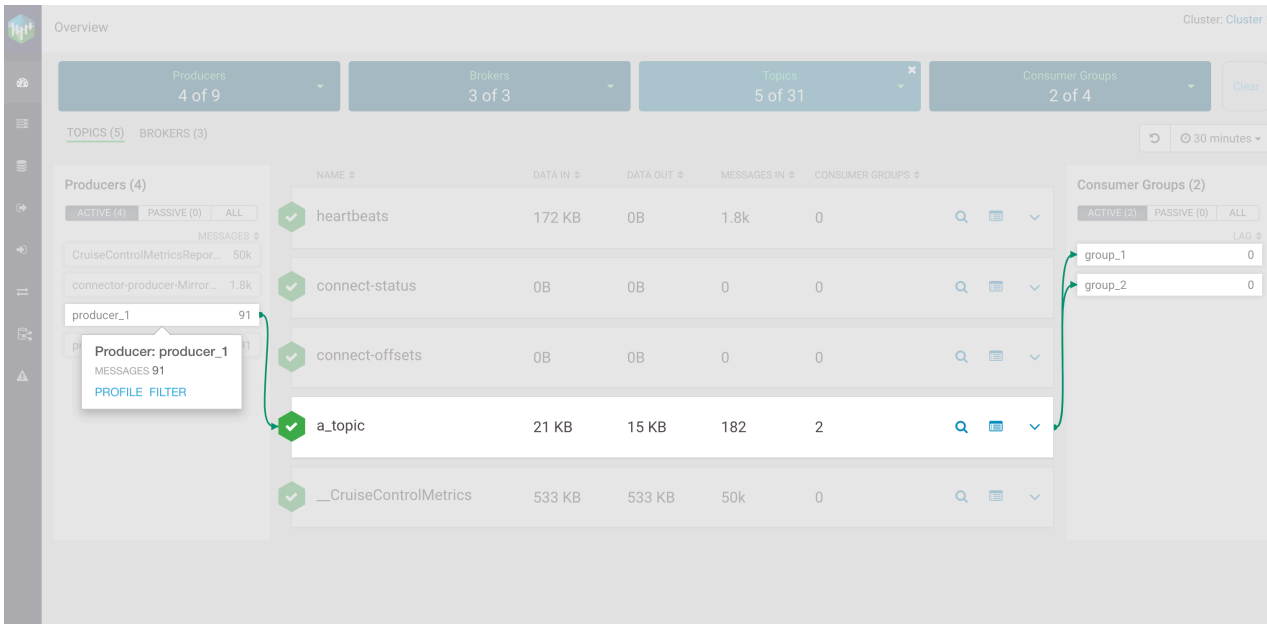

After you click producer_1, you can see that it produces to a topic called a_topic, and that both consumer groups (group_1 and group_2) consume from that topic.

This works the other way round as well. If you click on a single consumer group, you see what topics it consumes from and which producers produce to those topics. For example, click group_1, as shown in the following image:

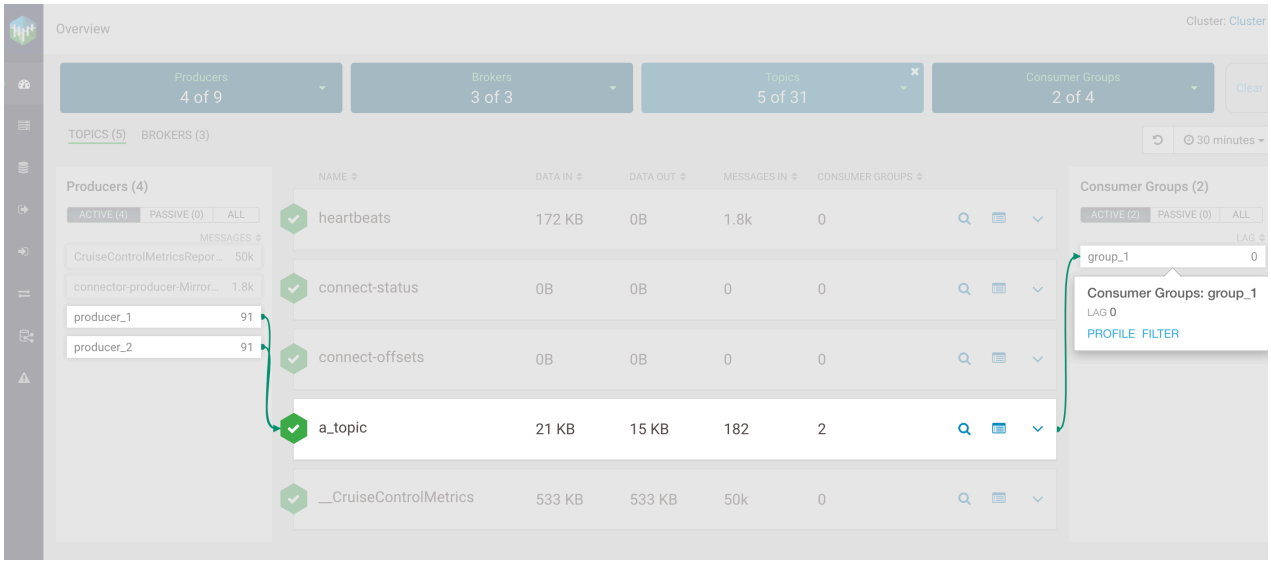

After you click group_1 consumer group, you can see that it consumes from the topic called a_topic, and that two producers produce to that topic (producer_1 and producer_2).

If you are interested in a more detailed view and want to check the lineage information for a single partition, you can do that as well, however, it is important to note that the lineage information is provided exclusively for the last 30 minutes. For example, click P3, as shown in the following image:

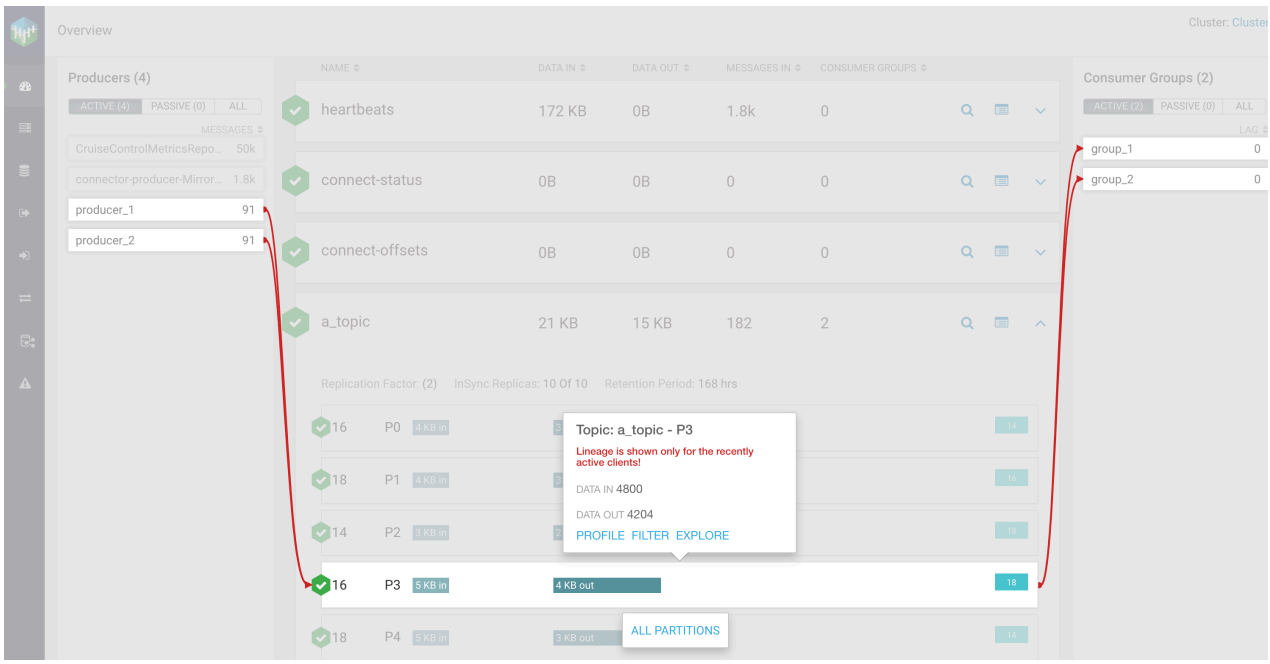

After you click P3 partition in the topic called a_topic, you can see that producer_1 and producer_2 produce to that partition, and group_1 and group_2 consume from it.

If you click the All Partitions button, you are shown the lineage information for every partition in a single topic.

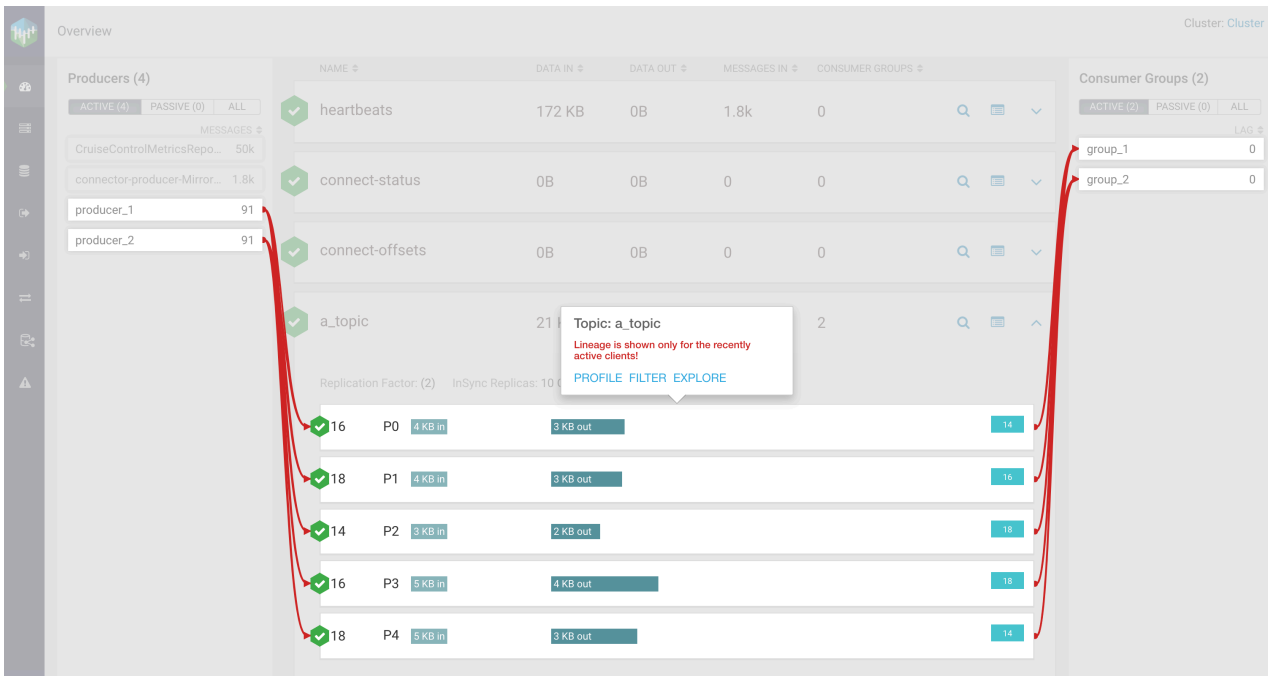

You can also access the lineage information from the experimental endpoints. You can find the endpoints at the *Streams Messaging Manager REST API Reference*.

Related Information

Streams Messaging Manager REST API Reference

Managing Kafka topics

Learn how to create, modify, and delete Kafka topics using Streams Messaging Manager.

Creating a Kafka topic

Learn how to create Kafka topics by using Streams Messaging Manager UI.

About this task

You can create a new Kafka topic by navigating to the Topics page and providing information prompted in the Add New pop-up.

Procedure

1. Navigate to the Topics page.
2. Click Add New.
3. Provide the following information:
   - Topic name
   - Number of partitions
   - Level of availability
   - Cleanup policy

4. Streams Messaging Manager automatically sets Kafka topic properties. You can manually adjust the properties by clicking advanced.

   For more information on configuration properties, see Apache Kafka documentation.
5. Click Save.

Related Information

Apache Kafka documentation: Topic-Level Configs

Modifying a Kafka topic

Learn how to modify Kafka topics by using Streams Messaging Manager UI.

About this task

You can modify Kafka topic properties by navigating to the Topics page, and editing content in the Configs tab of the topic Profile.

Procedure

1. Navigate to the Topics page and select the topic you want to modify.
2. Select the Profile icon.
3. Select the Configs tab.
4. Update the configurations.

   Note: You cannot change the topic name or the replication factor.

   In the simple version, you can update the number of partitions at partitions and the cleanup.policy.

   You can click advanced to access the additional properties. For more information on configuration properties, see Apache Kafka documentation.
5. Click Save.

Related Information

Apache Kafka documentation: Topic-Level Configs

Deleting a Kafka topic

Learn how to delete Kafka topics by using Streams Messaging Manager UI.

About this task

You can delete Kafka topics by navigating to the Topics page, and using the Delete Topic option from the topic Profile.

Procedure

1. Navigate to the Topics page and select the topic you want to delete.
2. Click the Profile icon.
3. Click the Actions button and select Delete Topic.
4. Confirm that you want to delete the topic.

Managing Alert Policies and Notifiers

Learn what Streams Messaging Manager Alert Policies and Notifiers are and how you can manage them.

Alert Policies in Streams Messaging Manager are user configured alerts that automatically trigger when configured conditions are met. An alert contains the details of the policy including the alert message and the conditions that triggered the alert. Once an alert is triggered, a notification is sent out by Streams Messaging Manager using the Notifier associated with the Alert Policy. You can use these alerts and notifications to monitor the health of Kafka, as well as to identify and troubleshoot problems.

For example, assume that you want to monitor your Kafka topics to ensure that data production and consumption is happening without interruptions. To do this you could set up an Alert Policy that triggers if the bytes consumed from or produced to your topics fall below a specified threshold. When the alert is triggered, Streams Messaging Manager sends out a notification using the Notifier associated with the Alert Policy immediately notifying you of a potential issue.

UI overview

Alerts in the navigation sidebar takes you to the **Alerts Overview** page, which you use to view and manage Alert Policies and Notifiers.

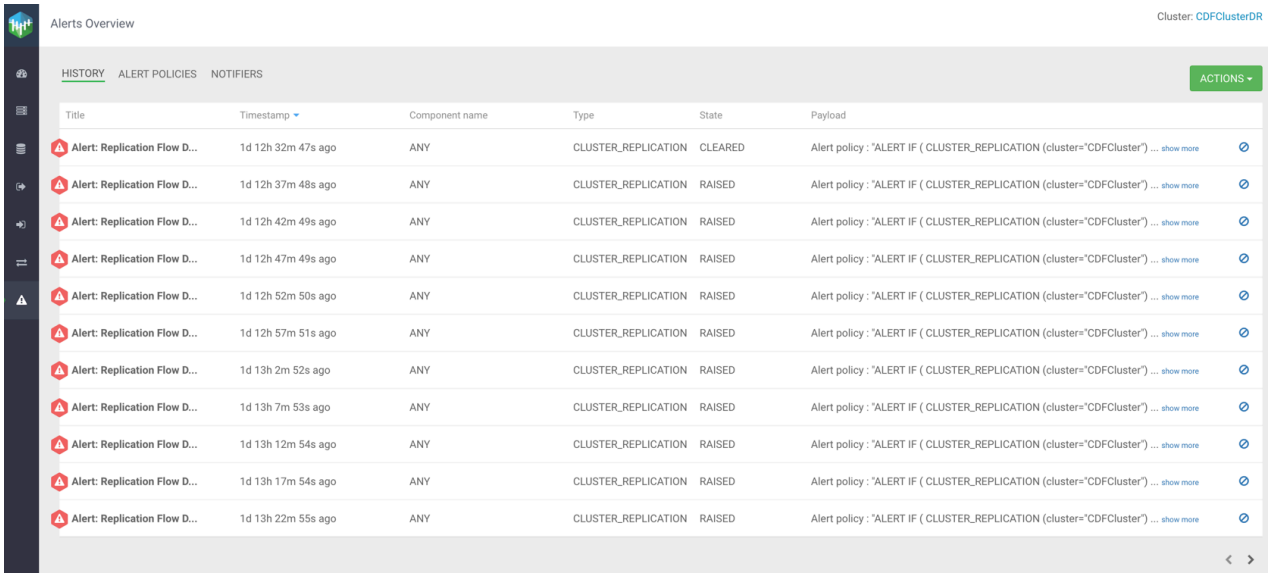

The page consists of three tabs. These are as follows:

For HISTORY

The **HISTORY** tab lists all triggered alerts. You can check the title, timestamp details, component name, type, state, and payload information of an alert. You can click show more to check complete payload details for an alert.

Click ACTIONS Mark All As Read to mark all the alerts as read. You can also click Dismiss to mark an individual alert as read.

For ALERT POLICIES

The **ALERT POLICIES** tab lists available Alert Policies. You can create new Alert Policies with ADD NEW. Additionally, you can manage existing policies.

For NOTIFIERS

The **NOTIFIERS** tab lists available Notifiers. You can create new Notifiers with ADD NEW. Additionally, you can manage existing Notifiers.

Creating a notifier

Learn how to use Streams Messaging Manager to create a notifier in your environment.

About this task

Perform the following steps to create a notifier:

Procedure

1. From the left navigation pane, select Alerts.
   The Alerts Overview page appears.
2. Click NOTIFIERS.
3. Click ADD NEW to create a new notifier.
   The Notifier window appears.
4. Configure the following properties:

Configuration	Property	Description
Common Notifier Configuration	NAME	Enter a unique name for the notifier.
	DESCRIPTION	Enter an optional description for the notifier.
	PROVIDER	Choose one of the following providers: • Email • HTTP
	NOTIFIER RATE LIMIT COUNT	Select the number of allowed notifications.
	NOTIFIER RATE LIMIT DURATION	Select the number of allowed notifications with respect to given duration in SECONDS, MINUTES, or HOURS.
Email Notifier Configuration	FROM ADDRESS	Enter the email address to use for SMTP mail command. Default is admin@localhost.
	TO ADDRESS	Enter one or multiple email addresses that you want to send the notification to.
	USERNAME	Enter the username for SMTP.
	PASSWORD	Enter the password for SMTP. **Note:** The password must be reentered when editing the configuration of existing notifiers.
	SMTP HOSTNAME	Enter the SMTP server that you want to connect to. Default is localhost.
	SMTP PORT	Enter the SMTP server port that you want to connect to. Default is 25.
	ENABLE AUTH	Select to enable authentication.
	ENABLE SSL/STARTTLS	Select to enable SSL. This is applicable when you enable authentication. You can either select SSL or STARTTLS.
	PROTOCOL	Enter the protocol to use to send emails. Default is SMTP.

Configuration	Property	Description
	ENABLE DEBUG	Select to enable debug mode to trace any issue in the email notifier. Disabled by default.
HTTP Notifier Configuration	URL	Enter the target service URL.
	CONNECTION TIMEOUT (MSECS)	Select the connection timeout in milliseconds for creating the initial connection. Default is 30 seconds.
	READ TIMEOUT (MSECS)	Select the read timeout in milliseconds for waiting to read data. Default is 30 seconds.

5. Click Save.

Updating a notifier

Learn how to use Streams Messaging Manager to update a notifier that you create in your environment.

About this task

Perform the following steps to update a notifier:

Procedure

1. From the left navigation pane, select Alerts.

   The Alerts Overview page appears.
2. Click NOTIFIERS.
3. Find the notifier you want to update from the list of available notifiers, and click the pencil icon beside the notifier.

   The Notifier window appears.
4. Edit the properties.

   Note: The password of email notifiers must be reentered when you update the configuration of the notifier.
5. Click Save.

Deleting a notifier

You can use Streams Messaging Manager to delete a notifier that you create in your environment. You can delete a notifier only if the notifier is not mapped to an alert policy.

About this task

Perform the following steps to delete a notifier:

Procedure

1. From the left navigation pane, select Alerts.

   The Alerts Overview page appears.
2. Click NOTIFIERS.
3. Find the notifier you want to delete from the list of available notifiers, and click the delete icon beside the notifier.
4. Click Yes.

Creating an alert policy

You can use Streams Messaging Manager to create an alert policy in your environment.

About this task

Perform the following steps to create an alert policy:

Procedure

1. From the left navigation pane, select Alerts.
   The Alerts Overview page appears.
2. Click ALERT POLICIES.
3. Click ADD NEW to create a new alert policy.
   The Alert Policy window appears.
4. Configure the following properties:

Property	Description
NAME	Enter a unique name for the alert policy.
DESCRIPTION	Enter a description for the alert policy.
EXECUTION INTERVAL IN SECONDS	Enter the execution interval in seconds to run the alert policy periodically after the given time interval.
EXECUTION DELAY IN SECONDS	Enter the execution delay in seconds to delay the execution of the alert policy. This is applicable only when the last execution of the alert policy triggered an alert. Ideally, this value should not be less than the value you enter for the EXECUTION INTERVAL IN SECONDS option.
ENABLE	Choose to enable or disable the alert policy.
COMPONENT TYPE	Select one of the following component types for the IF policy: • Broker • Consumer • Producer • Topic • Latency • Cluster • Cluster replication
TARGET NAME	Select the target name for the IF policy. You can add multiple WITH conditions by clicking the plus icon beside TARGET NAME.
TARGET CLUSTER NAME	Select the name of the target cluster for the IF policy. The property is visible when you select Cluster replication component type.
SOURCE CLUSTER NAME	Select the name of the source cluster for the IF policy. The property is visible when you select Cluster replication component type.
TOPIC NAME	Select the topic name for the IF policy. The property is visible when you select Latency or Cluster replication component type. For Cluster replication, the topic name represents the name of the topic on the target cluster.

Property	Description
CONSUMER NAME	Select the consumer name for the IF policy. The property is visible when you select Latency component type.
ATTRIBUTE	Select the attribute for the policy.
CONDITION	Select the condition for the policy.
VALUE	Select the value for the policy. You can add multiple attributes, conditions, and values by clicking the plus icon beside VALUE.
NOTIFICATION	Select a notifier.
PREVIEW	Displays the alert that you configure. For example, IF [COMPONENT_TYPE]: [TARGET_NAME] has [METRIC] [CONDITION] [VALUE] THEN notify by [NOTIFICATION]

5. Click Save.

Updating an alert policy

You can use Streams Messaging Manager to update an alert policy in your environment.

About this task

Perform the following steps to update an alert policy:

Procedure

1. From the left navigation pane, select Alerts.
   The Alerts Overview page appears.
2. Click ALERT POLICIES.
3. Find the alert policy that you want to update, and click the pencil icon beside the alert policy.
   The Alert Policy window appears.
4. Edit the properties.
5. Click Save.

Enabling an alert policy

You can use Streams Messaging Manager to enable an alert policy in your environment.

About this task

Perform the following steps to enable an alert policy:

Procedure

1. From the left navigation pane, select Alerts.
   The Alerts Overview page appears.
2. Click ALERT POLICIES.
3. Find the alert policy that you want to enable, and click the enable icon beside the alert policy.
   The alert policy is enabled.

Disabling an alert policy

You can use Streams Messaging Manager to disable an alert policy in your environment.

About this task

Perform the following steps to disable an alert policy:

Procedure

1. From the left navigation pane, select Alerts.
   The Alerts Overview page appears.
2. Click ALERT POLICIES.
3. Find the alert policy that you want to disable, and click the enable icon beside the alert policy.
   The alert policy is disabled.

Deleting an alert policy

You can use Streams Messaging Manager to delete an alert policy in your environment.

About this task

Perform the following steps to delete an alert policy:

Procedure

1. From the left navigation pane, select Alerts.
   The Alerts Overview page appears.
2. Click ALERT POLICIES.
3. Find the alert policy that you want to delete, and click the delete icon beside the alert policy.
4. Click Yes.

Component types and metrics for alert policies

You create an alert policy for a component type. The component type drives the list of metrics to select for creating a threshold. Learn the different component types and supported metrics for each component type.

The following table lists the component types and metrics for an alert policy:

Table 1: Component Types and Metrics

Metric	Description	Suggested Alert
Topic		
UNDER REPLICATED PARTITIONS COUNT	Total number of partitions that are under replicated for a topic.	Value > 0.
BYTES IN PER SEC	Bytes per second coming in to a topic.	Two kinds of alert can be configured. • Alert-1: Value = 0, raises an alert when the topic becomes idle. • Alert-2: Value > max_bytes_in_expected, raises an alert when the topic input load is higher than usual.

Metric	Description	Suggested Alert
BYTES OUT PER SEC	Bytes per second going out from a topic. It does not count the internal replication traffic.	Two kinds of alert can be configured. • Alert-1: Value = 0, raises an alert when the topic becomes idle. • Alert-2: Value > max_bytes_out_expected, raises an alert when the topic output load is higher than usual.
OUT OF SYNC REPLICA COUNT	Total number of replicas that are not in sync with the leader for a topic.	Value > 0, raises an alert if there are out of sync replicas for the topic.
TOPIC PARTITION CONSUMPTION PERCENTAGE	Percentage of bytes consumed per topic partition compared according to the configured parameter retention.bytes. If retention.bytes is not configured, any condition involving this metric would be false.	Value > max_expected_value, raises an alert if the topic partition reaches a certain consumption percentage.
TOPIC PARTITION BYTES IN PER SEC	Bytes per second coming in to a topic partition.	Two kinds of alert can be configured. • Alert-1: Value = 0, raises an alert when the topic partition becomes idle. • Alert-2: Value > max_bytes_in_expected, raises an alert when the topic partition input load is higher than usual.
TOPIC PARTITION BYTES OUT PER SEC	Bytes per second coming out of a topic partition.	Two kinds of alert can be configured. • Alert-1: Value = 0, raises an alert when the topic partition becomes idle. • Alert-2: Value > max_bytes_out_expected, raises an alert when the topic partition output load is higher than usual.
Producer		
IS PRODUCER ACTIVE	Checks whether a producer is active.	Value is False.
MILLISECONDS LAPSED SINCE PRODUCER WAS ACTIVE	Milliseconds passed since the producer was last active.	Value > max_producer_idle_time, raises an alert if the producer did not produce for max_producer_idle_time ms.
Cluster		
ACTIVE CONTROLLER COUNT	Number of brokers in the cluster reporting as the active controller in the last interval.	Value != 1.
ONLINE BROKER COUNT	Number of brokers that are currently online.	Depends on the application. For example, you can raise an alert if the number of brokers falls below the min.insync.replicas configured for the producer. • Alert-1: Value < min.insync.replicas, raises an alert when producer could not send any messages. • Alert-2: Value = min.insync.replicas, raises an alert that denotes if any one of the remaining brokers goes down, then producer would not be able to send messages.
UNCLEAN LEADER ELECTION COUNT	Number of unclean partition leader elections in the cluster reported in the last interval.	Value > 0.
UNDER REPLICATED PARTITIONS COUNT	Total number of topic partitions in the cluster that are under replicated.	Value > 0.
LEADER ELECTION PER SEC	Rate of partition leader elections.	Depends on the number of partitions in the application.

Metric	Description	Suggested Alert
OFFLINE PARTITIONS COUNT	Total number of topic partitions, in the cluster, that are offline.	Value > 0.
NETWORK PROCESSOR AVG IDLE PERCENT	Average fraction of time the network processor threads are idle across the cluster.	Two kinds of alert can be configured. • Alert-1: Value = 0, raises an alert when the network processor threads are busy. • Alert-2: Value > network_processor_idle_percentage, raises an alert when the network_processor_idle_percentage is higher than usual. Note: Value range is 0 to 1.
REQUEST HANDLER POOL AVG IDLE PERCENT	Average fraction of time the request handler threads are idle across the cluster.	Two kinds of alert can be configured. • Alert-1: Value = 0, raises an alert when the request handler threads are busy. • Alert-2: Value > request_handler_idle_percentage, raises an alert when the request_handler_idle_percentage is higher than usual. Note: Value range is 0 to 1.
BROKER BYTES IN DEVIATION PERCENTAGE	Percentage by which a broker bytes in per second has deviated from the average bytes in per second of all the alive brokers.	Value > max_byte_in_deviation_percentage, raises an alert if a broker is seeing more than max_byte_in_deviation_percentage incoming traffic compared to average incoming traffic seen by all the brokers.
BROKER BYTES OUT DEVIATION PERCENTAGE	Percentage by which a broker bytes out per second has deviated from the average bytes out per second of all the alive brokers.	Value > max_byte_out_deviation_percentage, raises an alert if a broker is seeing more than max_byte_out_deviation_percentage outgoing traffic compared to average outgoing traffic seen by all the brokers.
ZOOKEEPER SESSION EXPIRATION PER SEC	Average rate at which brokers are experiencing zookeeper session expiration per second.	If this value is high, it can lead to controller fail over and leader changes. Raises an alert if value > 0.
Consumer		
CONSUMER GROUP LAG	How far consumer groups are behind the producers.	Depends on the application.
IS CONSUMER ACTIVE	Checks whether a consumer is active.	Value is False.
MILLISECONDS LAPSED SINCE CONSUMER WAS ACTIVE	Milliseconds passed since the consumer was last active.	Value > max_consumer_idle_time, raises an alert if the consumer did not consume for max_consumer_idle_time ms.
Broker		
BYTES IN PER SEC	Number of bytes per second produced to a broker.	Two kinds of alert can be configured. • Alert-1: Value = 0, raises an alert when the broker becomes idle. • Alert-2: Value > max_bytes_in_expected_per_broker, raises an alert when the broker input load is higher than usual.
ZOOKEEPER SESSION EXPIRATION PER SEC	Rate at which brokers are experiencing Zookeeper session expirations per second.	If this value is high, it can lead to controller fail over and leader changes. Raises an alert if value > 0.

Metric	Description	Suggested Alert
TOTAL PRODUCE REQUESTS PER SEC	Total number of produce requests to a broker per second.	Depends on the application. Two kinds of alert can be configured. • Alert-1: Value =0, raises an alert when there are no produce requests for the last 15 minutes. • Alert-2: Value > usual_num_producer_requests_expected to detect the spike in the number of requests.
PARTITION IMBALANCE PERCENTAGE	The partition imbalance for a broker. It is calculated as: (abs(average_no_of_partitions_per_broker - actual_no_of_partitions_per_broker) / average_no_of_partitions_per_broker) * 100	Value > 10 %
BYTES OUT PER SEC	Number of bytes per second fetched from a broker. It does not count the internal replication traffic.	Two kinds of alert can be configured. • Alert-1: Value = 0, raises an alert when the broker becomes idle. • Alert-2: Value > max_bytes_out_expected_per_broker, raises an alert when the broker output load is higher than usual.
IS BROKER DOWN	Checks whether a broker is down.	Value is True.
TOTAL PRODUCE REQUEST LATENCY	Latency of produce requests to this broker at the 99th percentile (in ms).	Value > max_expected_latency_ms.
ISR SHRINKS PER SEC	Rate at which brokers are experiencing InSync Replica Shrinks (number of shrinks per second).	Value > 0.
TOTAL FETCH CONSUMER REQUEST LATENCY	Latency of fetch consumer requests to this broker at 99th percentile (in ms).	Value > max_expected_latency_ms.
REQUEST HANDLER POOL AVG IDLE PERCENT	Average fraction of time the request handler threads are idle.	Two kinds of alert can be configured. • Alert-1: Value = 0, raises an alert when the request handler threads are busy. • Alert-2: Value > request_handler_idle_percentage, raises an alert when the request_handler_idle_percentage is higher than usual. Note: Value range is 0 to 1.
NETWORK PROCESSOR AVG IDLE PERCENT	Average fraction of time the network processor threads are idle.	Two kinds of alert can be configured. • Alert-1: Value = 0, raises an alert when the network processor threads are busy. • Alert-2: Value > network_processor_idle_percentage, raises an alert when the network_processor_idle_percentage is higher than usual. Note: Value range is 0 to 1.
Cluster Replication		
REPLICATION LATENCY	15 minutes average replication latency in milliseconds.	Value > max_expected_replication_latency, raises an alert if the replication latency is greater than max_expected_replication_latency.
REPLICATION THROUGHPUT	15 minutes average replication throughput in bytes per second.	Value < min_expected_throughput, raises an alert if throughput during replication is low. This could happen because of network issues.

Metric	Description	Suggested Alert
CHECKPOINT LATENCY	15 minutes average checkpoint latency in milliseconds.	Value > max_expected_checkpoint_latency, raises an alert if the checkpoint latency is greater than max_expected_replication_latency.
REPLICATION STATUS	Replication status of a replication pipeline.	Value != ACTIVE, raises an alert if the replication is not active.
Latency		
END TO END LATENCY	15 minutes average of end to end latency in ms.	Value > max_expected_latency, raises an alert if the end to end latency is greater than max_expected_latency.

Monitoring end-to-end latency

You can use Streams Messaging Manager UI to monitor end-to-end latency in Kafka topics. By monitoring latency, you get a complete understanding of the time taken by a consumer to consume a message that is produced in a topic. You can also monitor the number of consumed messages across all the consumer groups for a topic within the selected time range.

Use the latency feature to achieve the following goals:

- Verify whether end-to-end processing time SLAs are met.
- Identify slow or lagging consumers.
- Verify whether messages are overconsumed or underconsumed.

You can find details about the number of messages produced in a topic, the number of messages consumed from a topic, and latency details during the consumption of the messages in the following two graphs in the Streams Messaging Manager UI:

- Messages Consumed. The graph provides you the overall produced message count and consumed message count across all the consumer groups for a topic within the selected time range. Any discrepancy in the produced and consumed message count is highlighted in red.

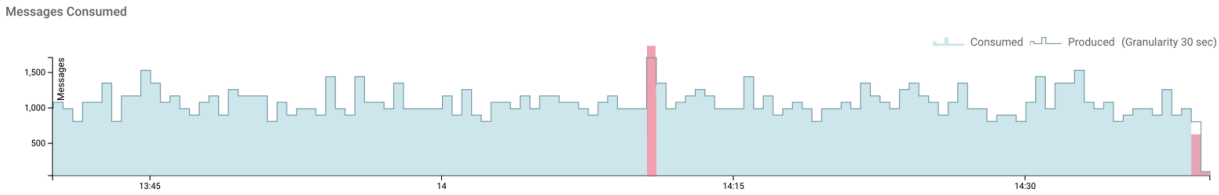

In the preceeding image, the linear formation represents the number of messages produced in last one hour, and the filled area represents the number of messages consumed in last one hour with a granularity of 30 seconds. The blue area signifies that all the produced messages are consumed. The red area represents a discrepancy in the produced and consumed message count and can either mean that messages are overconsumed or underconsumed.

In the image, there are two red areas. The first red area, from the left, indicates that the number of consumed messages is more than the number of produced messages. This represents an overconsumption of messages which

can occur when a consumer group offset is reset to an older offset to reprocess messages, or when producers or consumers are shut down in an unclean manner.

The last red area indicates that the number of consumed messages is less than the number of produced messages. This represents an underconsumption of messages which can occur when the consumer group offset is set to a newer offset causing the consumer group to skip processing some messages.

The far right section of the graph shows the current processing window where consumers are still consuming the produced messages. Therefore this area is expected to be marked red and indicates an undercsonumption of messages.

All other areas in the image are blue, which indicates that all the produced messages are consumed.

- End-to-end Latency. The End-to-end Latency graph provides you the details of the latency range and average latency in consuming the messages, which are produced in a particular topic, in milliseconds within the selected time range.

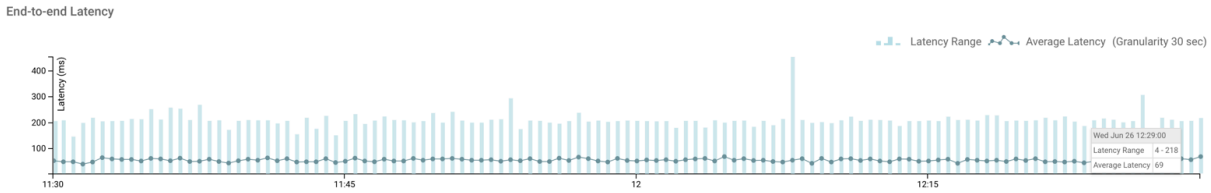

In the above image, the vertical lines represent the latency range, and the dotted line represents the average latency in consuming the produced messages in last one hour with a granularity of 30 seconds. You can see that at 12:29:00 hours on Wed Jun 26, the latency range was between 4 - 218 milliseconds and the average latency was 69 milliseconds.

Note: You can also create alerts to receive notifications based on the conditions that you configure in the alert policy for monitoring latency in your system. For more information about creating alerts to monitor latency, see the *Managing Alert Policies* guide.

Metric granularity

Streams Messaging Manager uses the REST API to display metrics. All the metrics are available for querying at two different granularities: 30 seconds and 15 minutes. The metrics are pre-aggregated for the defined buckets. Depending on how long the data is queried, granularity and varying dimensions of the topic, partition, consumer group ID and client ID, the data is calculated and rendered as JSON. Go through the following details before you start monitoring latency using Streams Messaging Manager:

- When you select a period newer than current time by 24 hours, the data is retrieved from REST server with metrics granularity of 30 seconds.
- When you select a period older than current time by 24 hours, the data is retrieved from REST server with metrics granularity of 15 minutes.
- Streams Messaging Manager UI polls the API for updates periodically (every 30 seconds if the selected period is newer than current time by 24 hours and every 15 minutes otherwise).
- By default, the 30 seconds granularity metrics are stored for 24 hours and the 15 minutes granularity metrics are stored for 2 weeks.

For information on REST APIs used in Streams Messaging Manager, see the *REST API Reference* guide.

Related Information

Managing Alert Policies and Notifiers

Streams Messaging Manager REST API Reference

Enabling interceptors

You need to enable interceptors for consumers, producers, and KafkaStreams applications to enable Streams Messaging Manager to fetch the metrics. If you do not enable the interceptors, you can not see any metrics in Streams Messaging Manager.

Interceptors publish the metrics to Kafka periodically. Metrics include counts on the producer side, and count average latency, and minimum and maximum latencies on the consumer side.

Steps to enable interceptors in your application

Add the following jar to the classpath of the application or as a dependency in the application:

```
<dependency>
 <groupId>com.hortonworks.smm</groupId>
 <artifactId>monitoring-interceptors</artifactId>
</dependency>
```

Steps to enable consumer interceptor

Perform the following steps to enable consumer interceptor:

1. Add the interceptor.classes property to consumer configuration that gets passed to the KafkaConsumer constructor.
2. Configure the client.id property as follows:

```
KafkaConsumer<Integer, String> createKafkaConsumer(String bootstrapServe
rs, String groupId, String clientIdentifier) {
    Properties properties = new Properties();
    properties.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapSe
rvers);
    properties.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, "org.ap
ache.kafka.common.serialization.IntegerDeserializer");
    properties.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, "org.a
pache.kafka.common.serialization.StringDeserializer");
    properties.put(ConsumerConfig.GROUP_ID_CONFIG, groupId);
    properties.put(ConsumerConfig.CLIENT_ID_CONFIG, clientIdentifier);
    //Add ConsumerInterceptor like this
    properties.put(ConsumerConfig.INTERCEPTOR_CLASSES_CONFIG,
       "com.hortonworks.smm.kafka.monitoring.interceptors.MonitoringCon
sumerInterceptor");
    return new KafkaConsumer<Integer, String>(properties);
}
```

Note: We recommend you to configure the client.id property. It helps in identifying the consumer instance. If you do not configure it, the latency metrics fetch the default consumer ID.

Steps to enable producer interceptor

Add the interceptor.classes property to producer configuration that gets passed to the KafkaProducer constructor, as follows:

```
KafkaProducer<Integer, String> createKafkaProducer(String bootstrapServers)
{
    Properties properties = new Properties();
    properties.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServers
);
    properties.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, "org.apach
e.kafka.common.serialization.IntegerSerializer");
```

```
    properties.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, "org.apac
he.kafka.common.serialization.StringSerializer");
    //Add ProducerInterceptor like this
    properties.put(ProducerConfig.INTERCEPTOR_CLASSES_CONFIG,
       "com.hortonworks.smm.kafka.monitoring.interceptors.MonitoringProduce
rInterceptor");
    return new KafkaProducer<Integer, String>(properties);
}
```

Steps to enable interceptors in KafkaStreams applications

Add the producer.interceptor.classes and consumer.interceptor.classes properties to Kafka Streams configurations, as follows:

```
void startKafkaStreams(StreamsBuilder builder) {
    KafkaStreams kstreams = new KafkaStreams(builder.build(), getKafkaStrea
msConfiguration());
    kstreams.start();
}
Properties getKafkaStreamsConfiguration() {
    Properties config = new Properties();
    config.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServers);
    config.put(StreamsConfig.APPLICATION_ID_CONFIG, appId);
    config.put(StreamsConfig.CLIENT_ID_CONFIG, clientId);

    //Add producer interceptor like this
    config.put(
      StreamsConfig.PRODUCER_PREFIX + ProducerConfig.INTERCEPTOR_CLASSES_CO
NFIG,
      "com.hortonworks.smm.kafka.monitoring.interceptors.MonitoringProduce
rInterceptor");
    //Add consumer interceptor like this
    config.put(
      StreamsConfig.CONSUMER_PREFIX + ConsumerConfig.INTERCEPTOR_CLASSES_C
ONFIG,
      "com.hortonworks.smm.kafka.monitoring.interceptors.MonitoringConsum
erInterceptor");

    return config;
}
```

Note: MonitoringProducerInterceptor publishes the producer metrics to the "__smm_producer_metrics" topic and MonitoringConsumerInterceptor publishes the consumer metrics to the "__smm_consumer_metrics" topic from their respective client applications. So if Authorization is enabled on kafka cluster, ensure that your client applications have access to the aforementioned topics.

Monitoring end to end latency for Kafka topic

To monitor end-to-end latency of Kafka topics, you can monitor the count of consumed messages across all Kafka consumer groups and the time taken by all Kafka consumer groups to consume messages that are produced in a Kafka topic, in a graphical way. You can also monitor the same for each Kafka consumer group, each client in a Kafka consumer group, and each partition in a Kafka topic.

About this task

Perform the following steps to monitor end-to-end latency in the Streams Messaging Manager UI:

Procedure

1. Go to Topics in the Streams Messaging Manager UI.
2. Select the topic you want to verify the details about.
3. Click the Profile icon beside the topic you select.

   This takes you to the Metrics page where you can find the Messages Consumed and End-to-end Latency graphs along with other topic details. On the Metrics page, these two graphs provide you an aggregated result of latency and count of consumed messages across all consumer groups.
4. To verify the details by individual consumer groups, clients, and partitions, go to the Latency tab.

   The Latency page appears as shown in the following image:

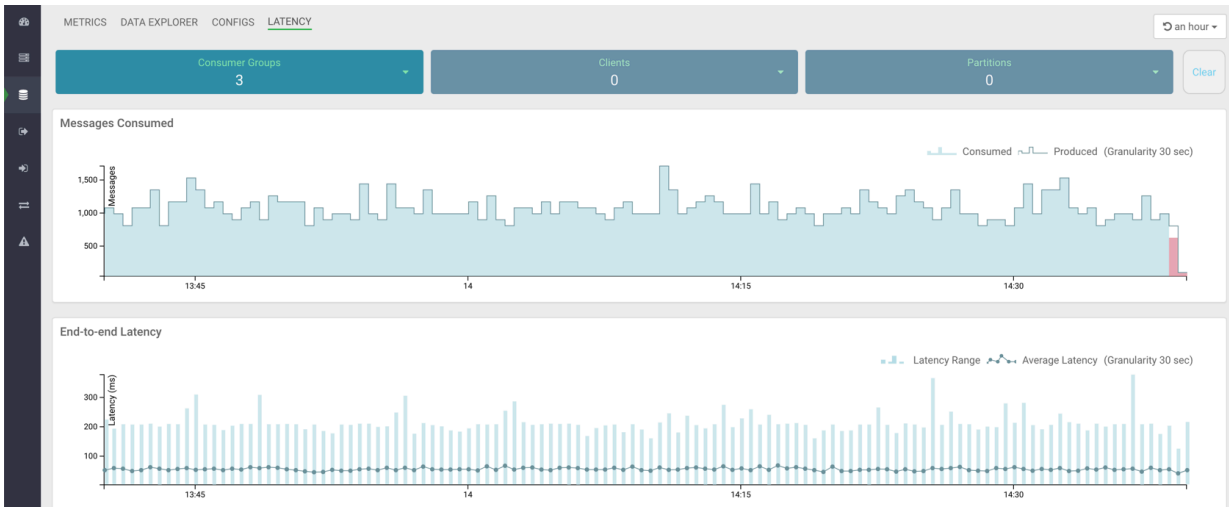

   The Latency view gives you a powerful snapshot of the end-to-end latency scenario: number of consumer groups for a topic, number of clients inside a particular consumer group, and number of partitions in a topic along with the Messages Consumed and End-to-end Latency graphs.
5. Select any consumer group from the Consumer Groups drop-down, as shown in the following image:

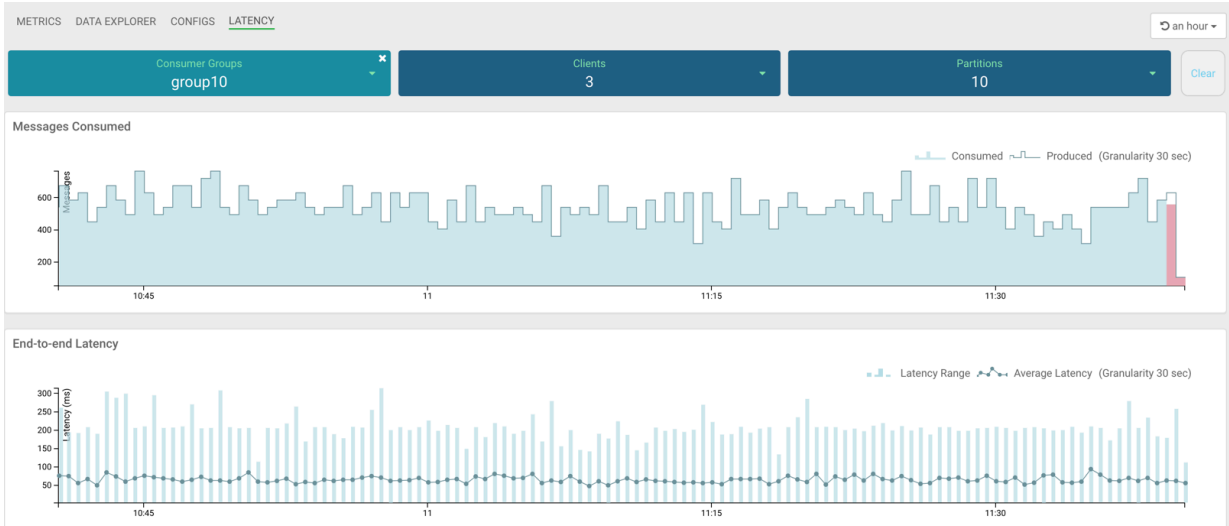

   In the image, group10 consumer group is selected. The Latency tab displays that there are 3 clients in the group10 consumer group, and 10 partitions are available in the topic.

6. Select any client from the Clients drop-down, as shown in the following image:

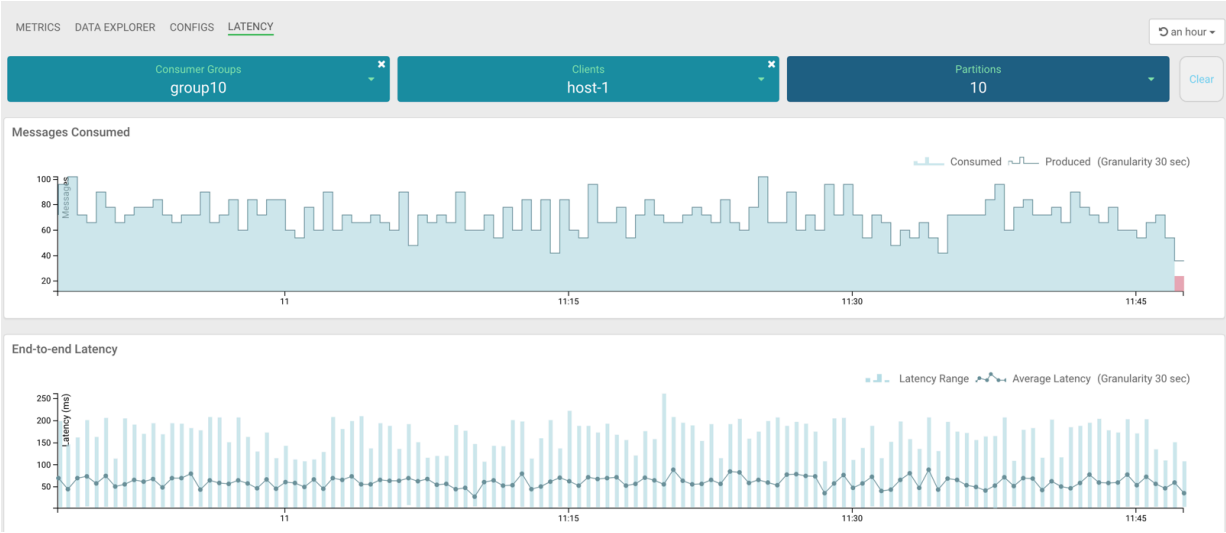

In the image, host-1 client is selected. At this instance, the Messages Consumed and End-to-end Latency graphs display data for only the host-1 client. Here you can monitor the number of messages produced, number of messages consumed, latency range, and average latency for host-1 only. Hover your mouse over the graphs and get data at any point of time in the selected time range. You can see in the Messages Consumed graph that host-1 consumed all the messages that are produced and actively consuming data at the latest time. You can see in the End-to-end Latency graph that the latency range and average latency are under 250 milliseconds.

7. To get details about the partitions from where host-1 is consuming data, click Partitions.

   The list of partitions in the topic appears, as shown in the following image:

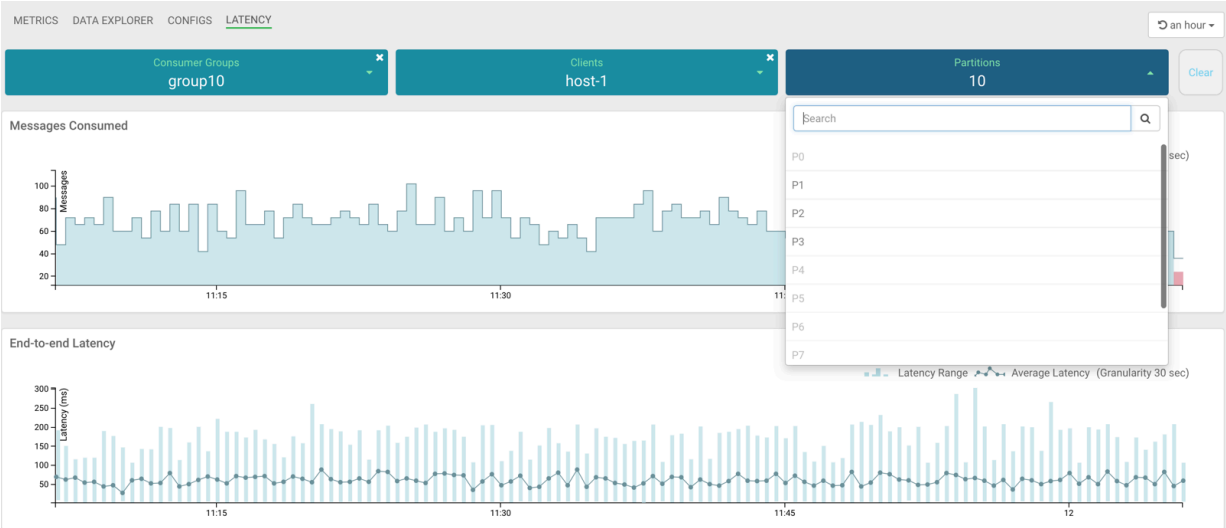

In the image, you can see that host-1 is consuming data from 3 partitions: P1, P2, and P3. Other partitions are inactive for host-1.

8. Select any active partition from the list.

   The Latency tab displays the transaction details between host-1 and the selected partition (for example, P1), as shown in the following image:

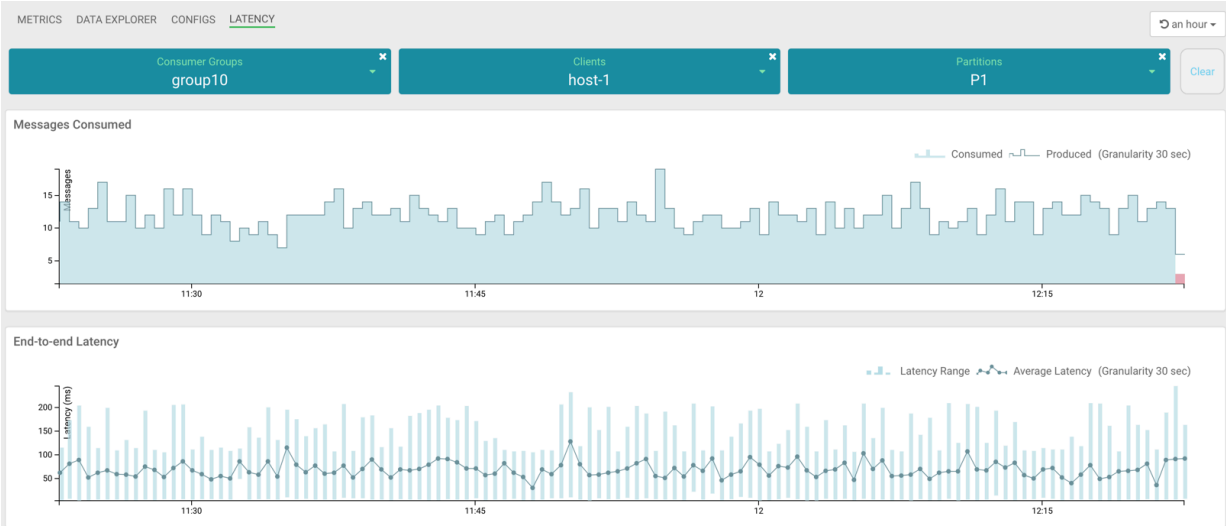

   Now you have got details for host-1 client. Likewise, you can fetch details for other clients.

9. Follow steps 6 through 8 to fetch data for all other clients.

10. Follow steps 5 through 8 to fetch data for all other consumer groups.

- To clear all the selections at once, click the Clear button at the top-right corner of the page.
- To clear selection for Consumer Groups, Clients, or Partitions, click the delete icon located on each drop-down.
- To select a different time range, use the Time Range and Quick Ranges options at the top-right corner of the page, as shown in the following image:

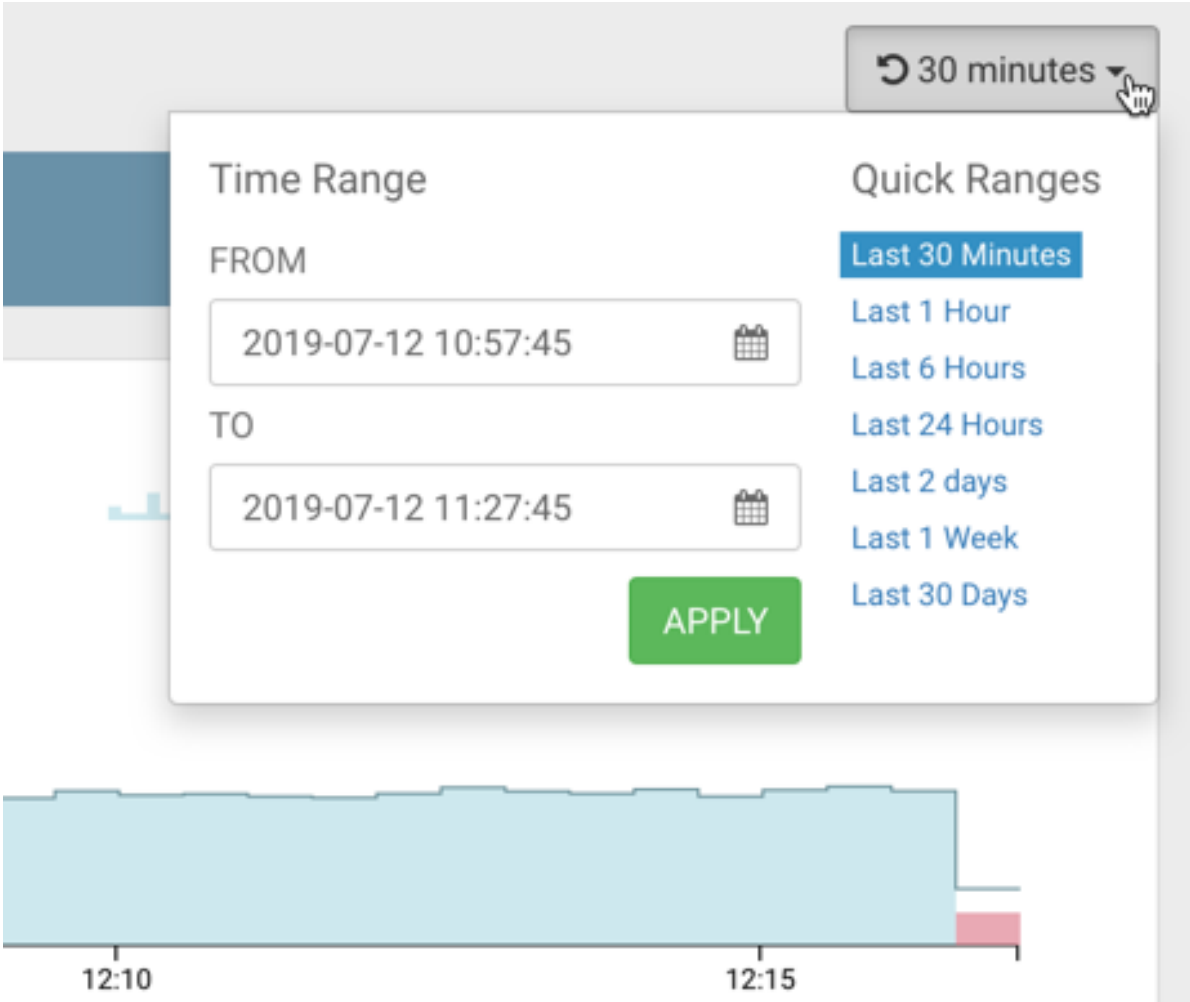

End to end latency use case

You can use end-to-end latency feature available in Streams Messaging Manager to handle real-time scenarios including measurement of end-to-end processing time, identification of slow or lagging consumers, and verification of over-consumption or under-consumption of messages.

Verify whether end-to-end processing time SLAs are met

A service-level agreement (SLA) is a commitment between a service provider and a service user. Particular aspects of the service are agreed between the service provider and the service user. The most common component of SLA is that the services should be provided to the user as agreed upon in the contract. For example, you agreed upon an average latency value and a maximum latency value for message consumption with Cloudera. Therefore, after a producer produces messages, if the messages take the agreed amount of time to be consumed by consumers, the SLA would be met.

1. Go to Topics in the Streams Messaging Manager UI.

2. Select the topic you want to verify the details about.
3. Click the Profile icon beside the topic you select.
4. Check the latency graph and see if the average and maximum latencies are as expected.
5. If the latencies are not as expected, go to the Latency tab.
6. Check whether the number of clients is as expected. If not, then you might want to check the missing client instance.
7. If the number of clients are as expected, then check if there is any spike in the message count. Select a period of 1 week in the Time Range pane and see if there is a surge in incoming messages that could explain the time SLA violation.
8. If the time SLA appears to be breached even after all the checks turned positive, go to Use Case 2.

Identify slow or lagging consumers

A stream based application flow involves application polling for messages, fetching and processing the messages, performing an optional blocking operation like interacting with database or local file system and then application polling for the messages again. However, the delay due to message processing, system bottleneck, or external bottleneck might become very long in some scenarios and you might want to understand which of the process instances is facing issues.

1. Go to Topics in the Streams Messaging Manager UI.
2. Select the topic you want to verify the details about.
3. Click the Profile icon beside the topic you select.
4. Go to the Latency tab.

5. After you select a group, inspect the latency and message counts for each client.

   This could lead you to the slow consumer.

   Let us walk through an example.

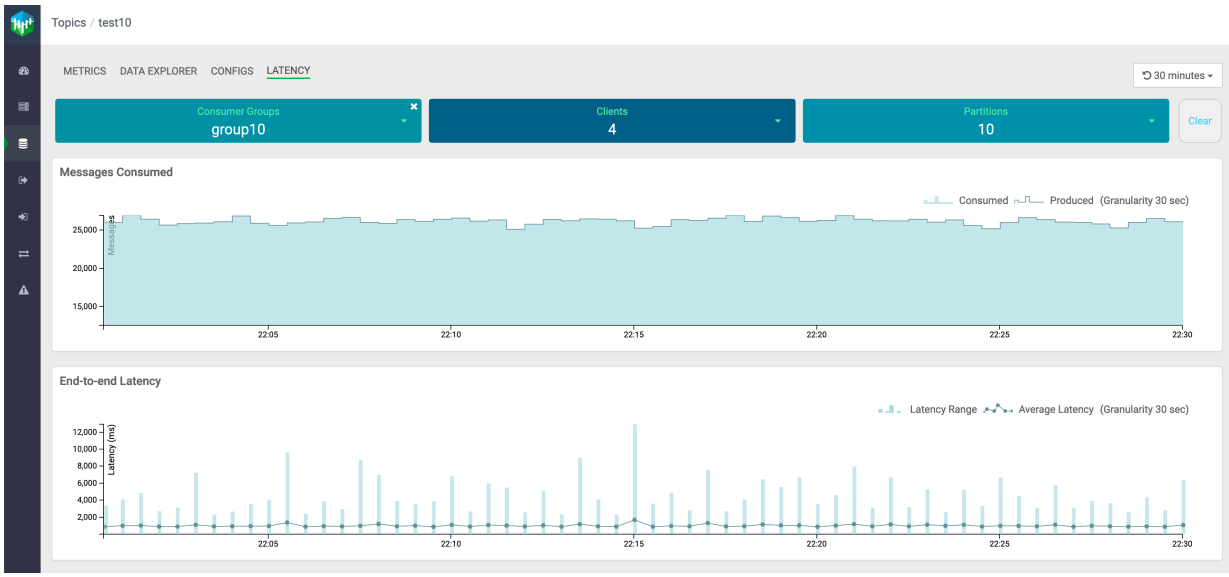

In the above image, you select group10 consumer group to inspect the latency and message counts for each client.

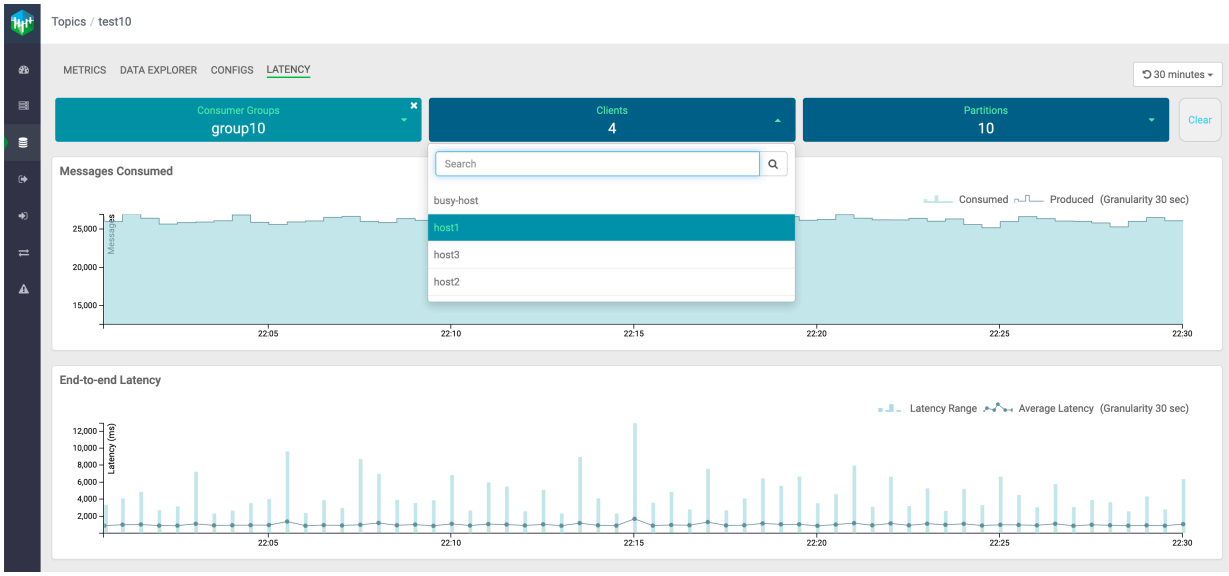

In the above image, you see the list of active clients for group10: host1, host2, host3, and busy-host. Now you need to select each client and inspect the latency and message counts.

In the above image, you can see that host1 consumed all the messages produced, and the average latency and latency range are under a good range.

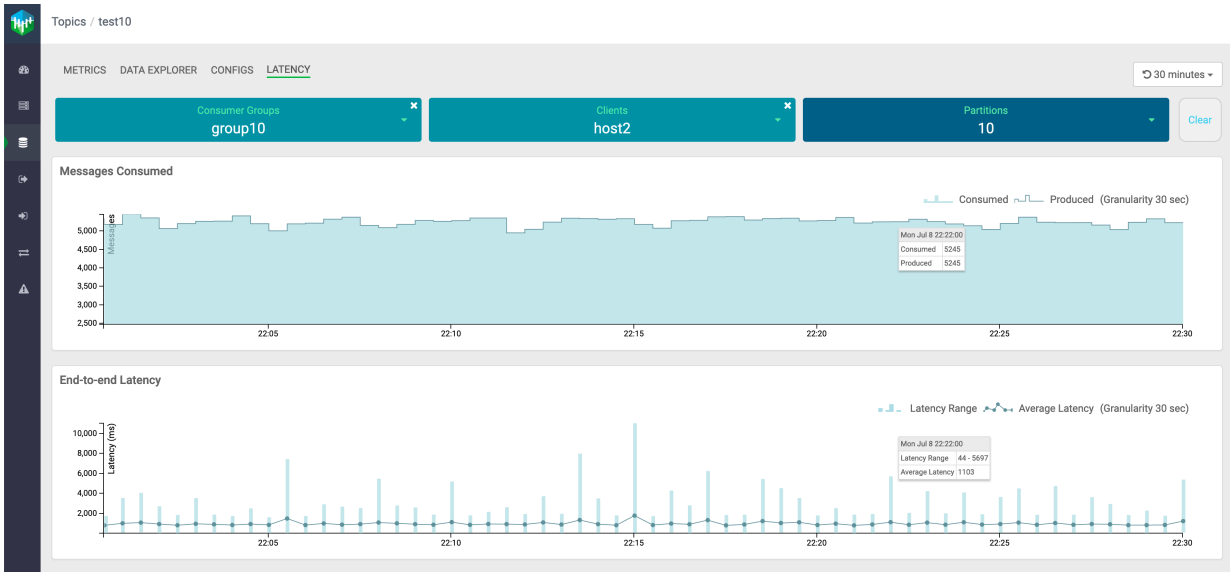

In the above image, you can see that host2 consumed all the messages produced. Also, there are few spikes in latency range, but the average latency is under a good range.

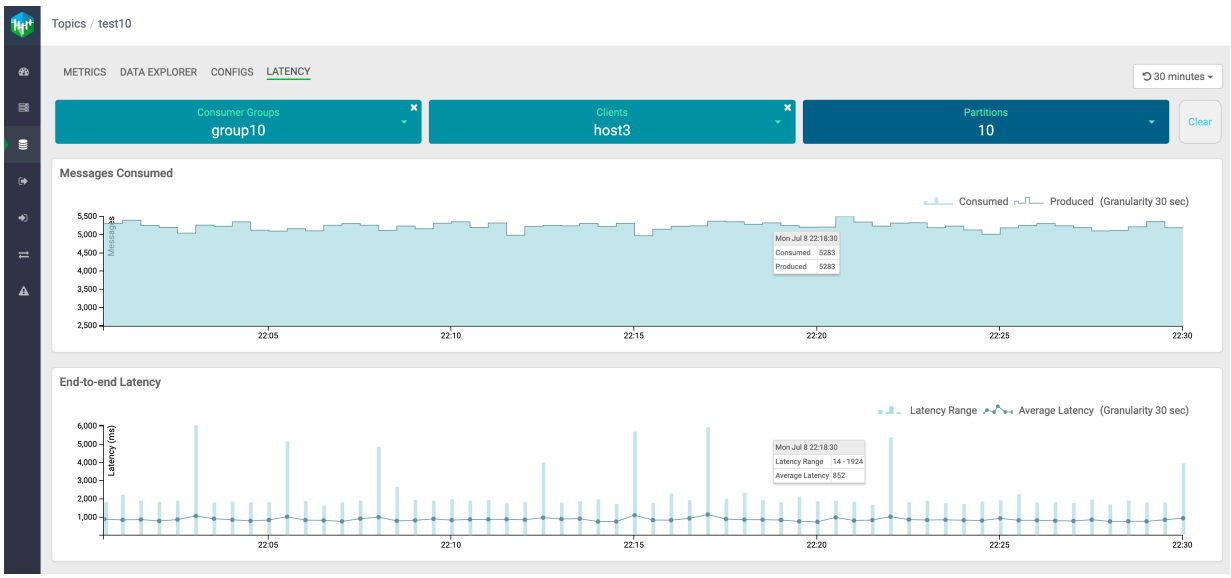

In the above image, you can see that host3 consumed all the messages produced. Also, there are occasional spikes in latency range, but the average latency is under a good range.

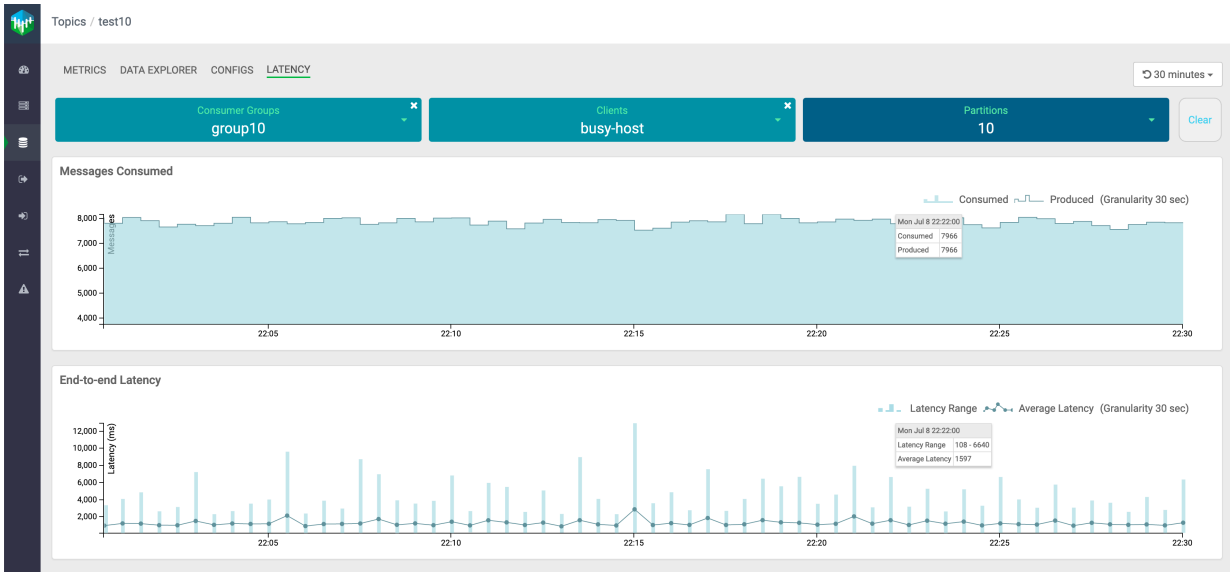

In the above image, you can see that busy-host consumed all the messages produced. There are frequent spikes in latency range, and both latency range and average latency are high. Hence, busy-host is the slow consuming client.

It is possible that all the clients are experiencing longer latencies. This could imply that there is a common bottleneck. For example, clients are interacting with external store over network and having delays in consuming the messages due to network issues.

If there is a single client that is experiencing slowness, you must check the message counts for other clients, and system parameters like CPU and memory.

This addresses your need to identify the slow consuming application.

Verify whether messages are overconsumed or underconsumed

There could be an overconsumption of messages. This might occur due to the following reasons:

- If the producers and consumers are shut down in an unclean manner or the producers and consumers went down in an unexpected manner. For example, Kafka producer produced some messages but it shut down before the

producer received any acknowledgement from the broker. Similarly, Kafka consumer consumed a few messages but got shut down before it could commit the offset at this latest point.

- If the consumer is reset to an older offset (reprocessing scenarios).

If the consumer is reset to a new offset (real-time application requirement), there could be an underconsumption of messages.There could be over or under consumption of messages if the cluster is in an unhealthy state.

1. Go to Topics in the Streams Messaging Manager UI.
2. Select the topic you want to verify the details about.
3. Click the Profile icon beside the topic you select.
4. Go to the Latency tab.
5. After you select a group, inspect the produced message and consumed message counts for each client in Messages Consumed graph.

   This helps you to verify whether the consumers are consuming all the messages that are produced in a topic. You might also identify occurrences of any overconsumption or underconsumption of messages.

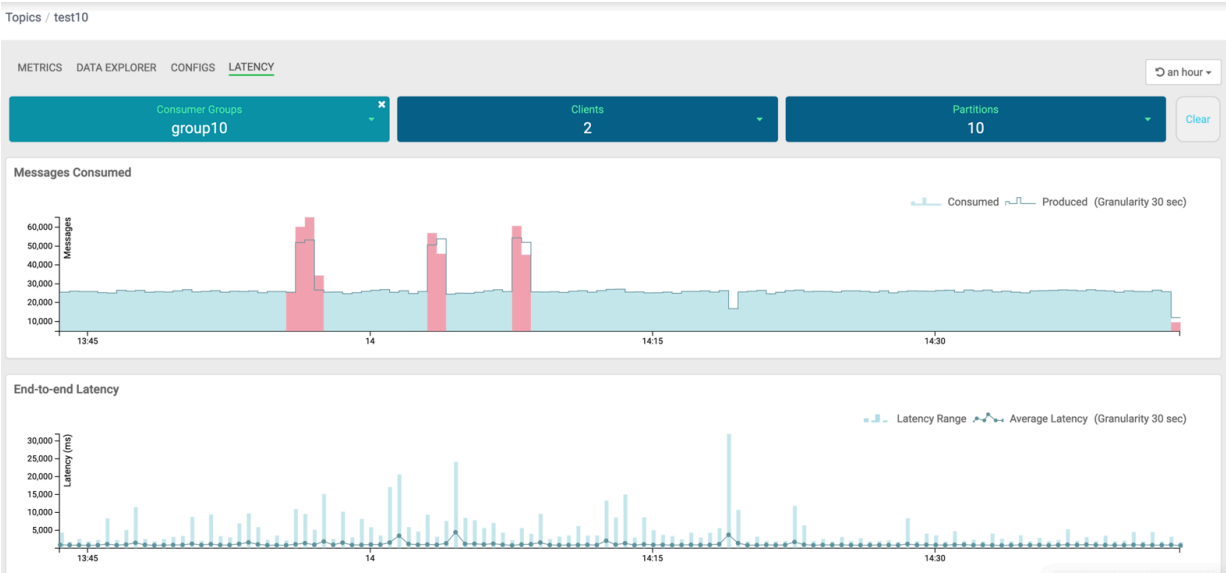

   In the image, you can see that for group10 consumer group, there are three red spikes in the Messages Consumed graph.

   The first spike, from the left, indicates that the number of consumed messages is more than the number of produced messages. So, this is an overconsumption of messages.

   The second and the third spike indicate overconsumption of messages followed by an underconsumption of messages.

Monitoring Kafka cluster replications (Streams Replication Manager)

Get started with monitoring Streams Replication Manager using Streams Messaging Manager.

Streams Replication Manager is an enterprise-grade replication solution that enables fault tolerant, scalable and robust cross-cluster Kafka topic replication. Streams Replication Manager provides the ability to dynamically change configurations and keeps the topic properties in sync across clusters at high performance. Streams Replication Manager also delivers custom extensions that facilitate installation, management and monitoring making Streams Replication Manager a complete replication solution that is built for mission critical workloads.

Streams Messaging Manager is capable of integrating with Streams Replication Manager enabling you to monitor your Streams Replication Manager replications on the Streams Messaging Manager UI. You can monitor status

of the Kafka cluster replications, number of topics associated with the replication, throughput, replication latency, and checkpoint latency. Additionally, the alerting features of Streams Messaging Manager are also supported for replications. This allows you to create receive alerts in connection to your replications.

The following tasks and concepts walk you through the **Cluster Replications** section of the Streams Messaging Manager UI, which is your main hub in Cloudera where you view and monitor details regarding your Streams Replication Manager replications.

Note: The **Cluster Replications** page is only available on the Streams Messaging Manager UI if integration between Streams Messaging Manager and Streams Replication Manager is enabled. Integration is enabled when you add the Streams Replication Manager service to your cluster. However, integration can be enabled separately, after service installation as well.

Related Information

Managing Alert Policies and Notifiers

Integrating Streams Replication Manager with Streams Messaging Manager

Viewing Kafka cluster replication details

You can view and monitor your Streams Replication Manager replications in the Cluster Replications section of the Streams Messaging Manager UI. Learn more about what information you can view on the UI regarding replications.

About this task

Note: Which exact replications are available for monitoring on the UI is determined by the configuration of the Streams Replication Manager Service roles that Streams Messaging Manager integrates with. You will only be able to view the replications that target the cluster(s) that the Streams Replication Manager Service roles target. Replications targeting other clusters in your deployment will not be visible. If you want to view all replications in your deployment using a single Streams Messaging Manager service, you must either enable the Remote Querying feature in Streams Replication Manager (recommended), or reconfigure the Streams Replication Manager Service roles that integrate with Streams Messaging Manager to target all clusters in your deployment.

You can view the status of the replications, source cluster names, target cluster names, number of topics to be replicated, number of consumer groups, throughput of the replication, replication latency, and checkpoint latency. Streams Messaging Manager also displays two graphs, one represents the throughput of the replication and the other displays the replication latency of the replication along with the details of each topic to be replicated.

Perform the following steps to view the details of a cluster replication:

Procedure

In the Cluster Replications page, click a cluster replication or the drop-down icon beside a replication, as shown in the following image:

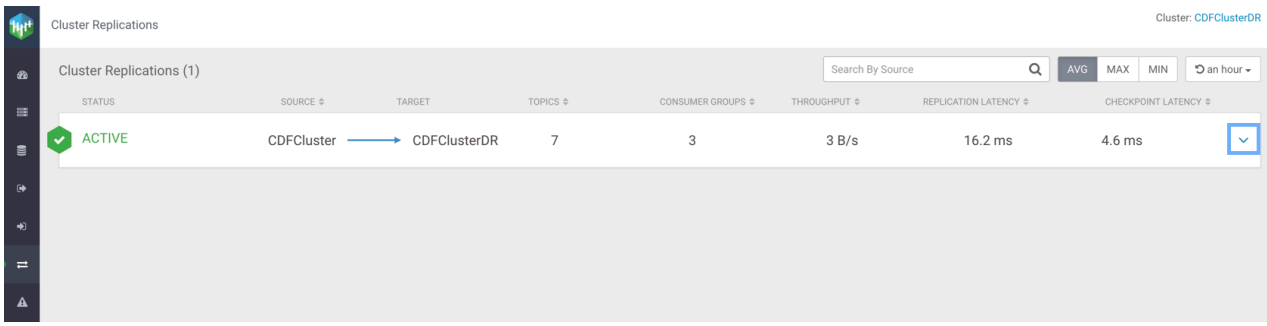

The replication details appear as shown in the following image:

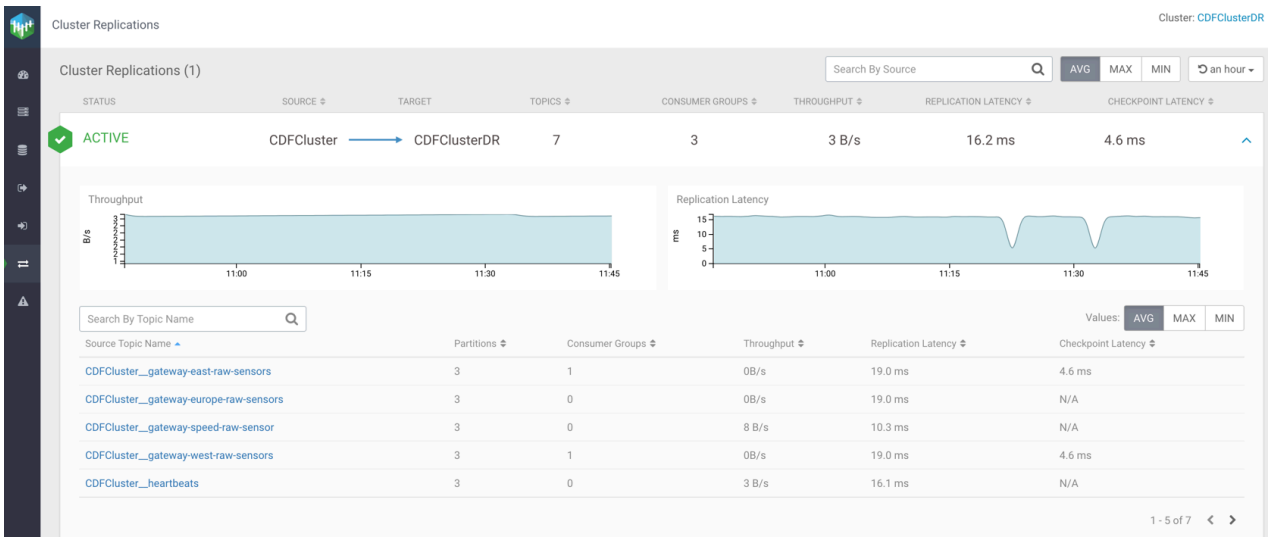

Related Information

Streams Replication Manager Service

Remote Querying

Searching Kafka cluster replications by source

Learn how to filter your Kafka cluster replications by source cluster after you select the Kafka cluster which acts as the target cluster.

About this task

You can use the Search By Source bar at the top right of the page to search cluster replications by the source cluster name. For example, if the destination cluster is amsterdam, and you enter scottsdale in the Search By Source bar, Streams Messaging Manager displays the Kafka cluster replication between scottsdale and amsterdam, as shown in the following image:

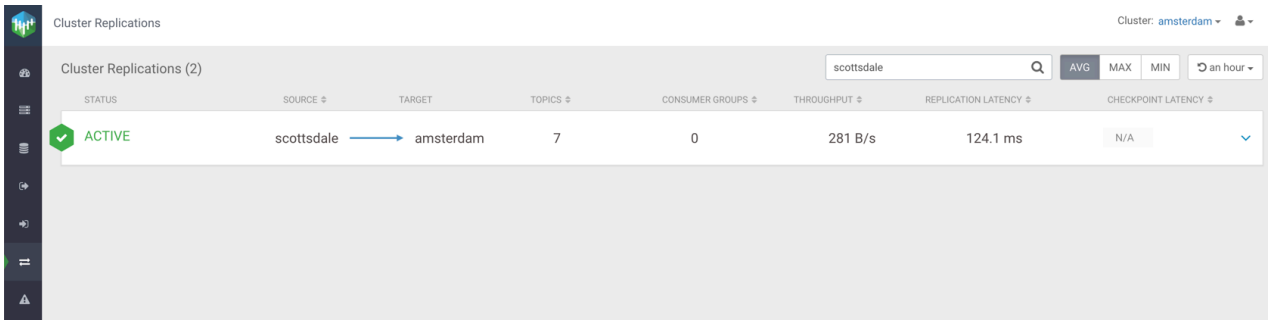

Monitoring Kafka cluster replications by quick ranges

Learn how to filter your cluster replications by specific time periods after you select the Kafka cluster which acts as the target cluster.

About this task

You can filter the cluster replications by time range. You can choose any of the following filter values from the drop-down to display the cluster replication details in the selected time range:

- Last 1 Hour
- Last 6 Hours
- Last 24 Hours

- Last 2 Days

The following image shows the Quick Ranges drop-down:

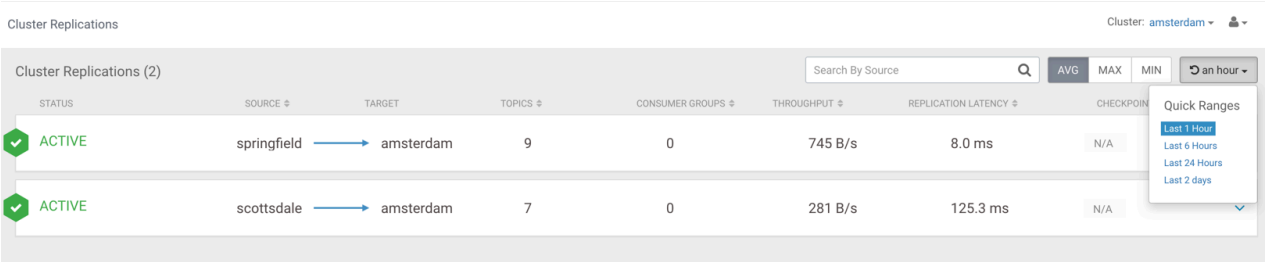

Monitoring status of the clusters to be replicated

You can monitor the status of Kafka cluster replications in the Status column in the Cluster Replications page. You can see whether a cluster replication is active, inactive, or showing a warning.

The status has three variations:

- Active

  Indicates that the cluster replication is running.
- Inactive

  Indicates that the cluster replication is not running.
- Warning

  Indicates that the cluster replication is facing issues.

If the status of a Kafka cluster replication shows inactive or warning, check the logs, and troubleshoot the replication.

Monitoring topics to be replicated

You can monitor the number of topics associated with a Kafka cluster replication in the Topics column in the Clus ter Replications page. You can also monitor details on source topic name, partitions, consumer groups, throughput, replication latency, and checkpoint latency.

Click on the cluster replication to fetch topic details. Streams Messaging Manager displays the following details about the topics:

- Source Topic Name

  Name of the topic at the source.
- Partitions

  Number of partitions of the topic at the source.
- Consumer Groups

  Number of consumer groups consuming data from the topic.
- Throughput

  Data replicated between the source cluster and the target cluster per second from a topic. Throughput is measured in bytes per second. By default, Streams Messaging Manager displays the average throughput. You can also fetch the maximum or minimum throughputs for a topic by clicking the MAX or MIN button located above the topic details.
- Replication Latency

  Amount of time taken for a message of a topic to get replicated from the source cluster to the target cluster. Replication latency is measured in milliseconds. By default, Streams Messaging Manager displays the average

replication latency. You can also fetch the maximum or minimum replication latency for a topic by clicking the MAX or MIN button located above the topic details.

- Checkpoint Latency

  Amount of time taken for a message of a topic to checkpoint on the target cluster after the message is committed on the source cluster. Checkpoint latency is measured in milliseconds. By default, Streams Messaging Manager displays the average checkpoint latency. You can also fetch the maximum or minimum checkpoint latency for a topic by clicking the MAX or MIN button located above the topic details.

The following diagram shows details of topics in a cluster replication:

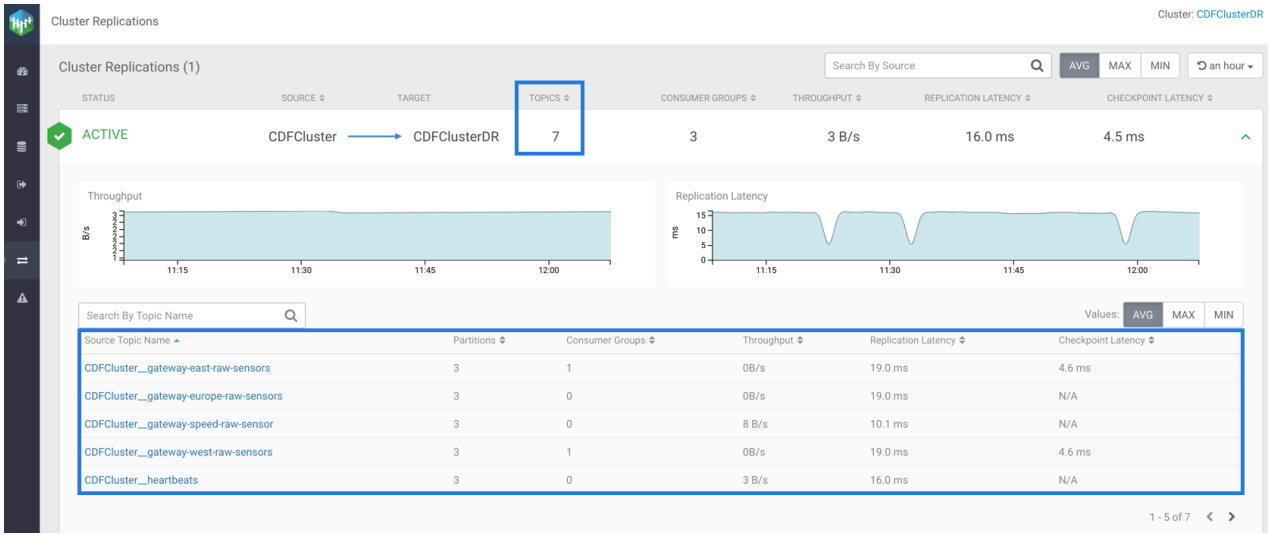

In the image, you can see that the number of topics to be replicated from CDFCluster to CDFClusterDR is 7, and topic details including topic names, number of partitions for the topics at the source cluster, number of consumer groups consuming messages from each topic, throughput, replication latency, and checkpoint latency of each topic.

Searching by topic name

Learn how to search a Kafka topic by name in a Kafka cluster replication and fetch details for that topic. After you find a topic, you can see the topic name, partitions, consumer groups, throughput, replication latency, and checkpoint latency for that topic.

You can use the Search By Topic Name bar to search a Kafka topic by name and get details of that topic. The following image shows details of the CDFCluster__heartbeats topic:

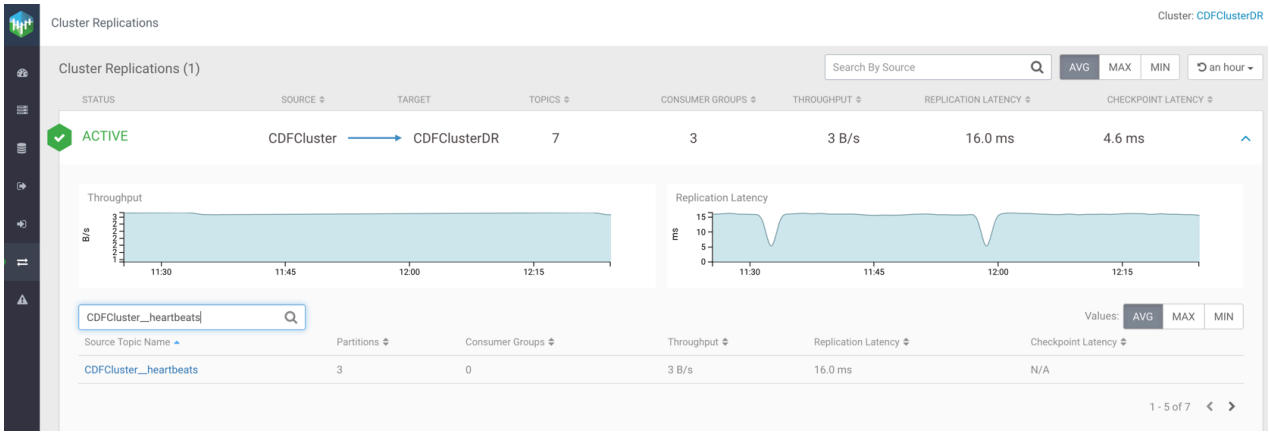

Monitoring throughput for cluster replication

Learn how to monitor the throughput of a Kafka cluster replication in Streams Messaging Manager. You can monitor average, maximum, and minimum throughput of a cluster replication. You can also monitor the throughput of a cluster replication graphically.

Throughput is defined as the data replicated between the source cluster and the destination cluster per second. Throughput is measured in bytes per second.

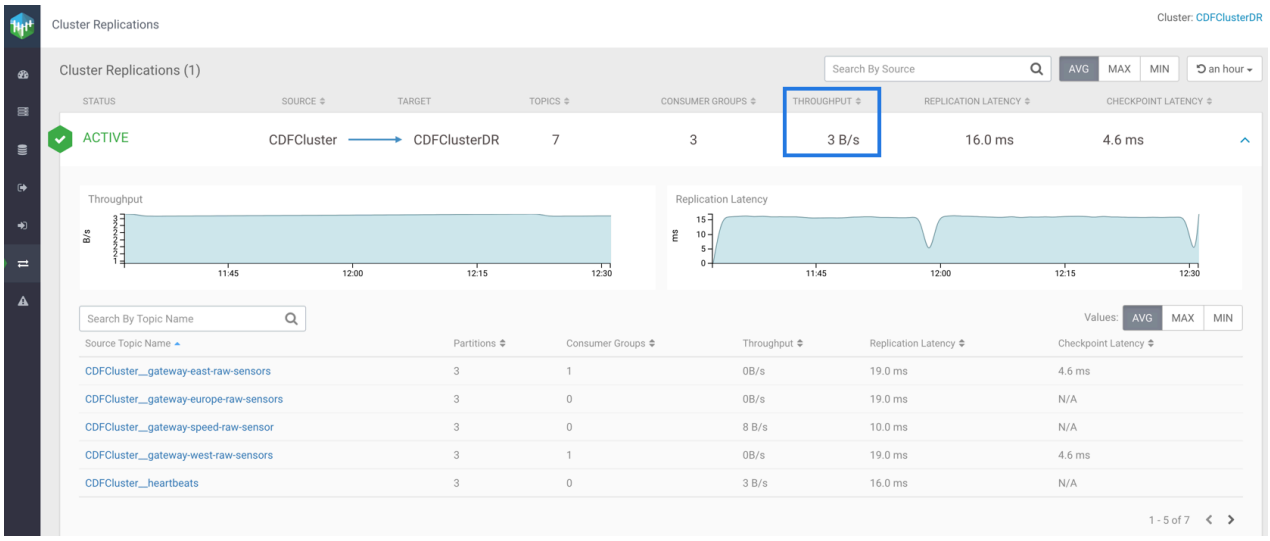

In the image, you can see the average throughput for the CDFCluster to CDFClusterDR replication is 3 bytes per second. You can fetch the maximum or minimum throughput for the replication by clicking the MAX or MIN button located above the cluster replication.

You can also monitor the throughput of a cluster replication graphically. Streams Messaging Manager displays the Throughput graph for each cluster in the cluster details. The following image shows the graph for throughput for CDFCluster to CDFClusterDR replication:

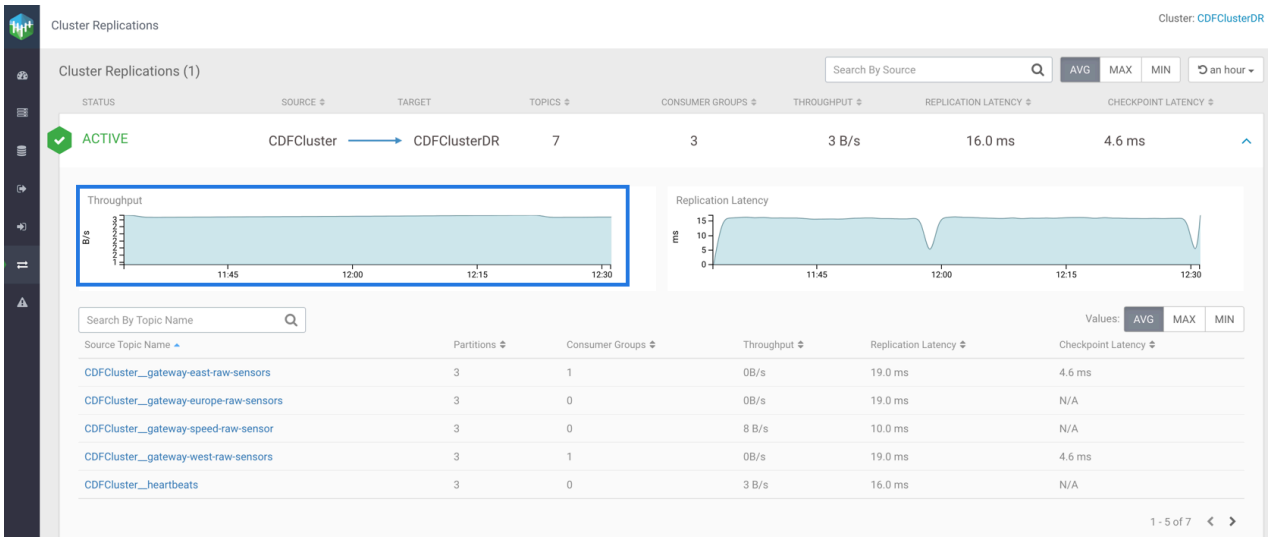

Monitoring replication latency for cluster replication

Learn how to monitor the replication latency of a Kafka cluster replication in Streams Messaging Manager. You can monitor average, maximum, and minimum replication latency of a cluster replication. You can also monitor the replication latency of a cluster replication graphically.

Replication latency is defined as the amount of time taken for a message to get replicated from the source cluster to the destination cluster. Replication latency is measured in milliseconds.

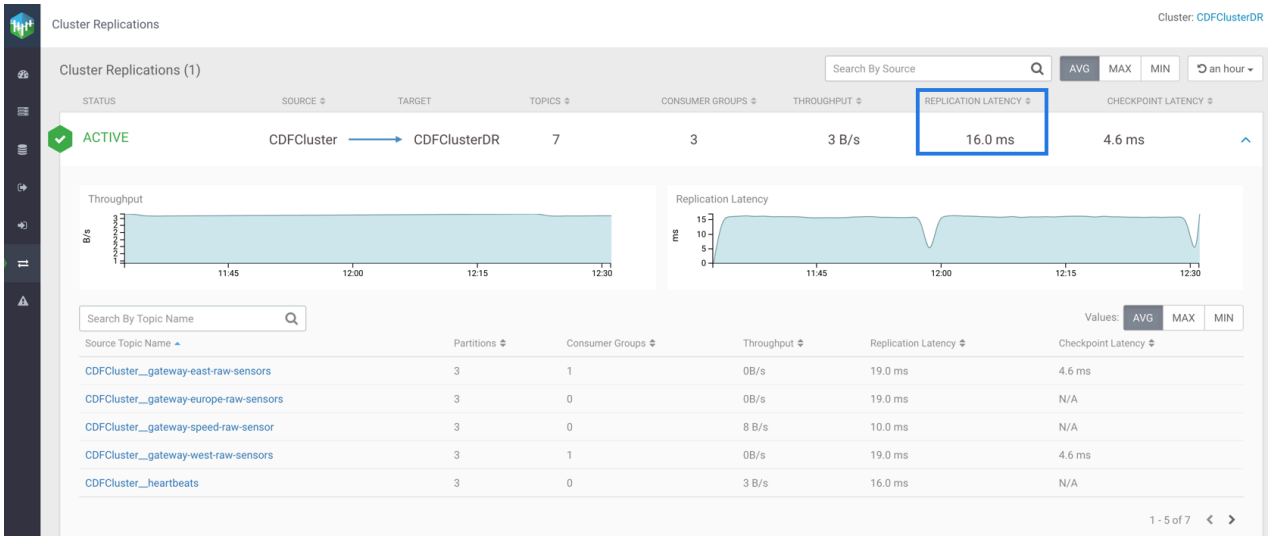

In the image, you can see the average replication latency for the CDFCluster to CDFClusterDR replication is 16.0 milliseconds. You can fetch the maximum or minimum replication latency for the replication by clicking the MAX or MIN button located above the cluster replication.

You can monitor the replication latency of a cluster replication graphically. Streams Messaging Manager displays the Replication Latency graphs for each cluster in the cluster details. The following image shows the graph for replication latency for CDFCluster to CDFClusterDR replication:

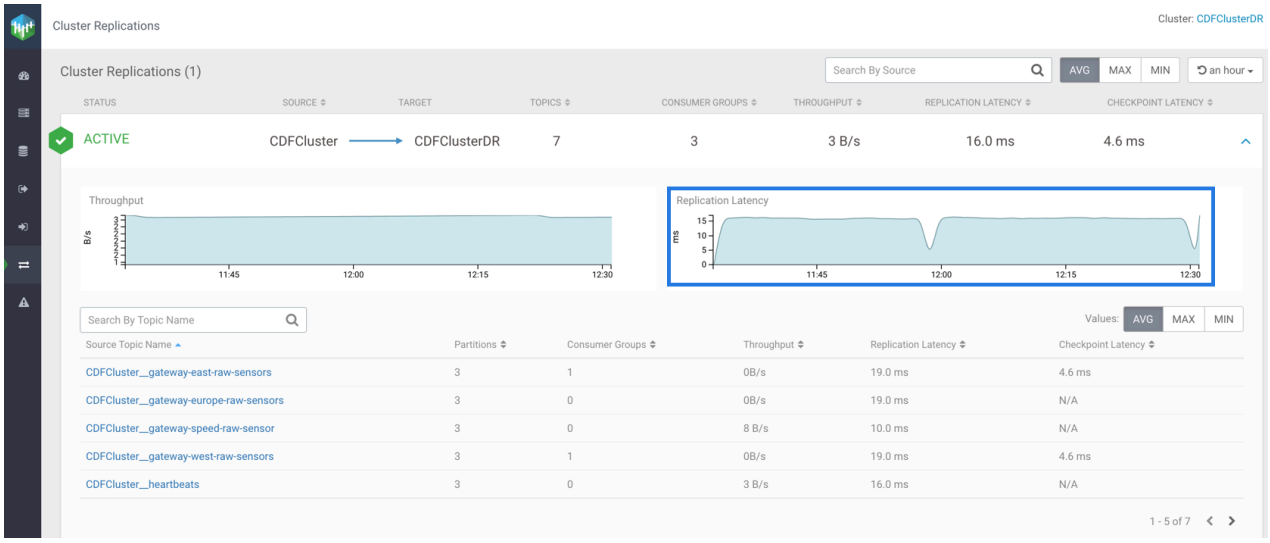

Monitoring checkpoint latency for cluster replication

Learn how to monitor the checkpoint latency of a Kafka cluster replication in Streams Messaging Manager. You can monitor average, maximum, and minimum checkpoint latency of a cluster replication. You can also monitor the checkpoint latency of a cluster replication graphically.

Checkpoint latency is defined as the amount of time taken for a message of a topic to checkpoint on the target cluster after the message is committed on the source cluster. Checkpoint latency is measured in milliseconds.

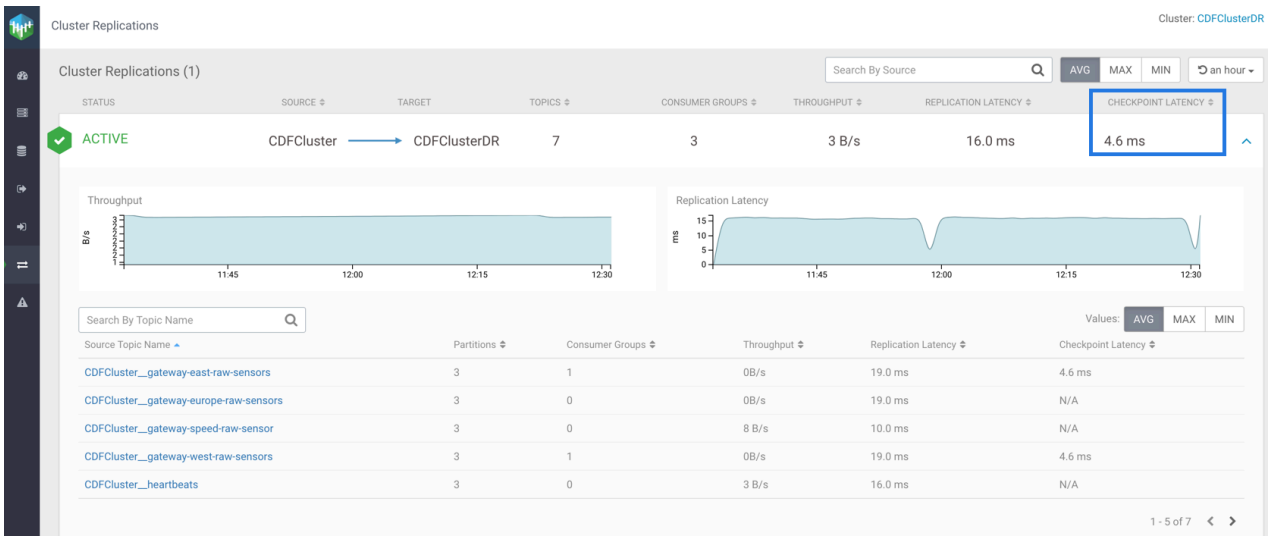

In the image, you can see the average checkpoint latency for the CDFCluster to CDFClusterDR replication is 4.6 milliseconds. You can fetch the maximum or minimum replication latency for the replication by clicking the MAX or MIN button located above the cluster replication.

Note: If the checkpoint latency for a cluster replication or a topic appears as Not Available, then that means there are no consumer groups defined.

Monitoring replication throughput and latency by values

Learn how to fetch the average, maximum, and minimum values for throughput, replication latency, and checkpoint latency of a Kafka cluster replication in Streams Messaging Manager.

You can do this at the following levels:

- Cluster Replication Level

  Click the AVG, MAX, or MIN buttons, as shown in the following image, to fetch average, maximum, or minimum values of throughput, replication latency, and checkpoint latency for cluster replications.

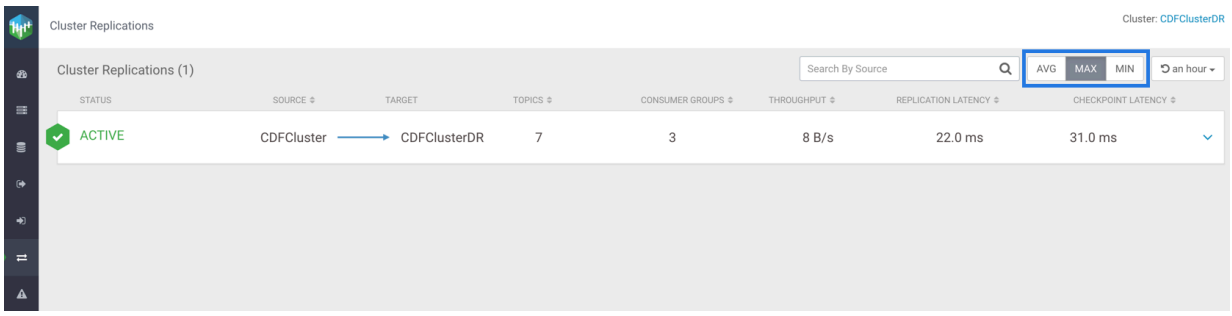

- Topic Level

  Click the AVG, MAX, or MIN buttons, as shown in the following image, to fetch average, maximum, or minimum values of throughput, replication latency, and checkpoint latency for topics.

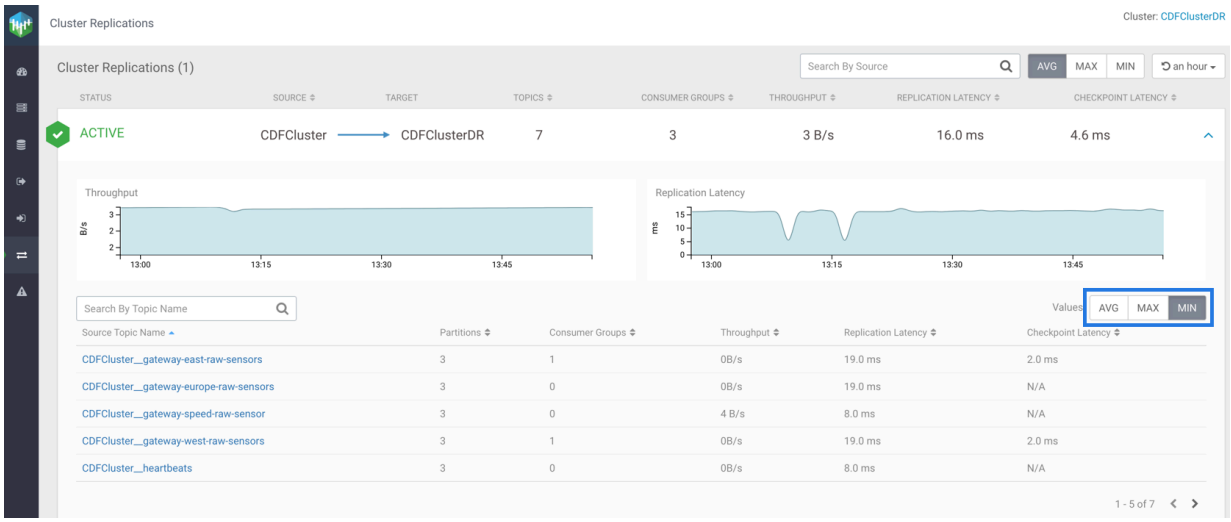

Managing and monitoring Kafka Connect using Streams Messaging Manager

Get started with Kafka Connect in Streams Messaging Manager.

Kafka Connect is a tool for streaming data between Apache Kafka and other systems in a reliable and scalable fashion. Kafka Connect makes it simple to quickly define connectors that move large collections of data into and out of Kafka. Source connectors can ingest entire databases or collect metrics from all your application servers into Kafka topics, making the data available for stream processing with low latency. Sink connectors can deliver data from Kafka topics into secondary storage and query systems or into batch systems for offline analysis.

Kafka Connect in Cloudera is shipped with many differentCloudera-developed as well as publicly available sink and source connectors. Each of which cover a specific use case for streaming data. In addition to the connectors available by default, installing custom developed or third-party connectors is also possible. All connectors can be deployed, managed, and monitored using the Streams Messaging Manager UI (recommended), Streams Messaging Manager REST API, or Kafka Connect REST API.

Related Information

Connectors

The Kafka Connect UI

Learn about the Kafka Connect section in the Streams Messaging Manager UI, which you can use to deploy, manage, and monitor Kafka Connect connectors.

The **Connect** section in Streams Messaging Manager is your main hub in Cloudera where you deploy, manage, and monitor Kafka Connect connectors. This section of the UI is only available if you deployed Kafka Connect role instances under your Kafka service and Streams Messaging Manager is configured to integrate with Kafka Connect. For detailed instructions on how to set up Kafka Connect and integrate it with Streams Messaging Manager, see *Kafka Connect Setup*.

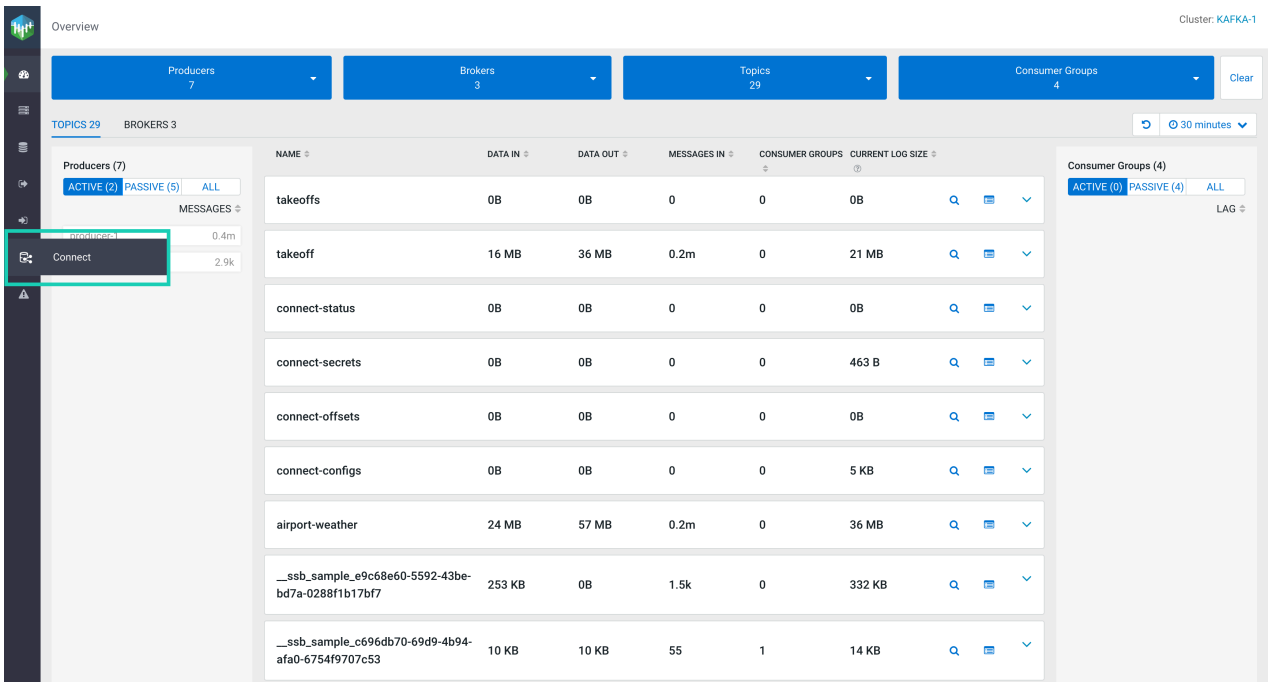

Clicking Connect, opens the **Connect Cluster** page with the **Connectors** tab open. This is the default page of the Kafka Connect UI. It provides a high level overview of the connectors and the connect cluster.

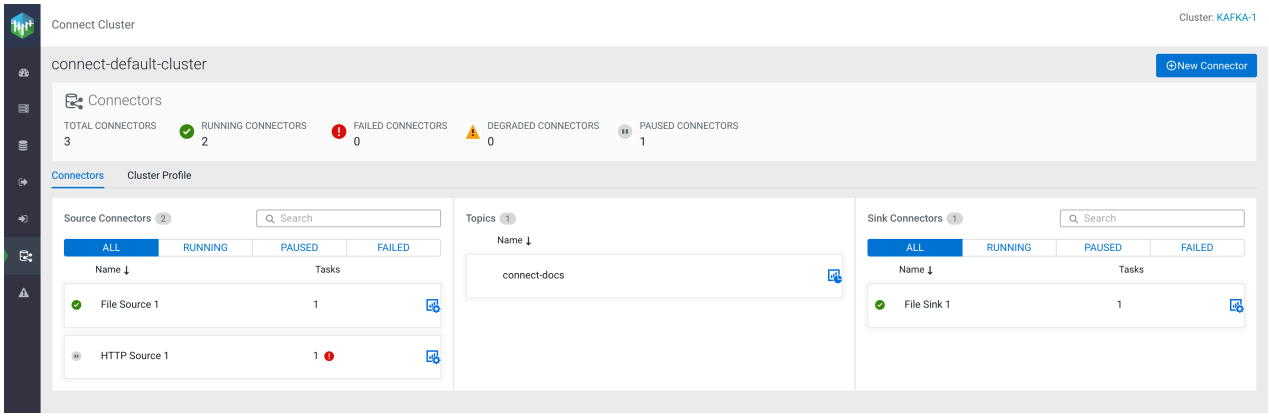

At the top-right corner of the **Connect Cluster** page, you can see the name of the cluster. The New Connector option below the cluster name starts the **Connector Setup** wizard, which you use to deploy connectors in your cluster. For detailed steps on how to deploy a new connector, see *Deploying and Managing connectors* .

The **Connectors** section provides some basic metrics regarding Kafka Connect including total connectors, running connectors, failed connectors, degraded connectors, and paused connectors. Under the Connectors section you can find two tabs, Connectors and Cluster Profile. The two tabs enable you to monitor and manage your connectors and the connect cluster.

Connectors tab

In the Connectors tab, you can view details of the source connectors, topics, and sink connectors in the cluster. The Source Connectors and Sink Connectors sections show all, running, paused, and failed connectors with connector name and associated task details. Both Source Connectors and Sink Connectors sections contain a Search option that enables you to search for particular connector details. The Topics section shows the Kafka topics where data is read from or written to.

Hovering over a specific connector displays a context menu where you can view the class name and status of the connector. The context menu also provides quick options to pause, resume, and restart the connector. Clicking on any of your deployed connectors or topics visualizes how the data is flowing into or out of the selected entities.

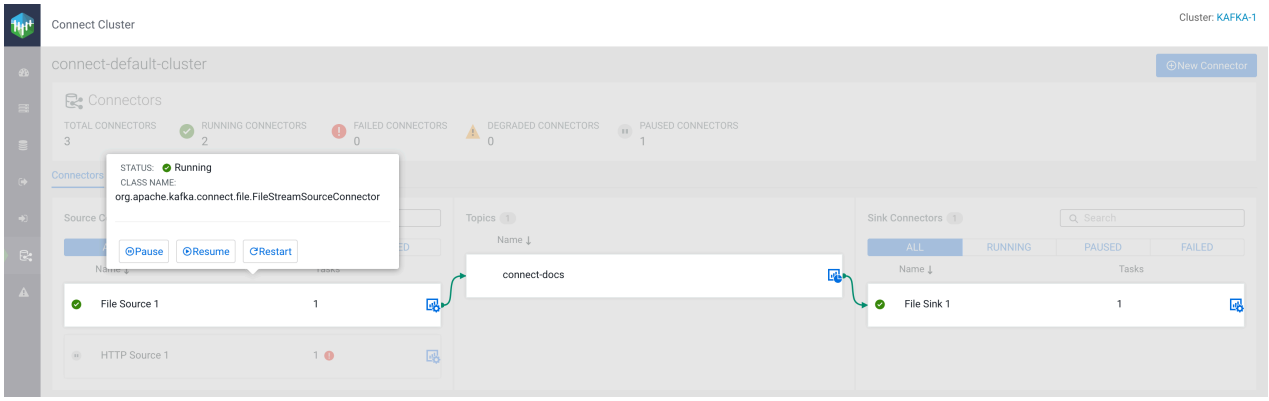

Cluster Profile tab

In the Cluster Profile tab, you can view details of the connect cluster and the workers. The tab includes information regarding your cluster such as the Cloudera Manager URL, the number of workers deployed in the cluster, as well as an uptime counter. Additionally, the **Workers** section lists all worker hosts available in the cluster. This section includes basic worker-level information, such as how many connectors are deployed on a worker, success and failure rates, and more.

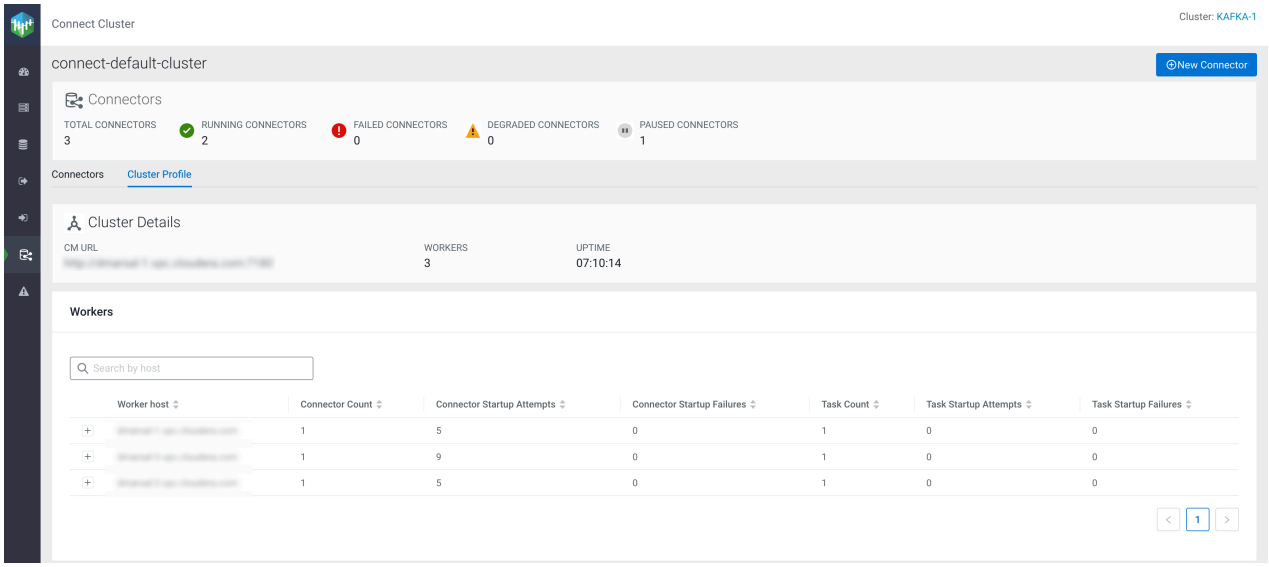

If you click the + option found next to each host, additional details and metrics are displayed that are specific to the worker host you selected. The information includes a list of connectors assigned to the worker, as well detailed connector, task, and worker rebalance metrics. Using the Search by host field, you can search for worker details by host. Additionally, you can filter the list of connectors deployed on the worker host based on their status.

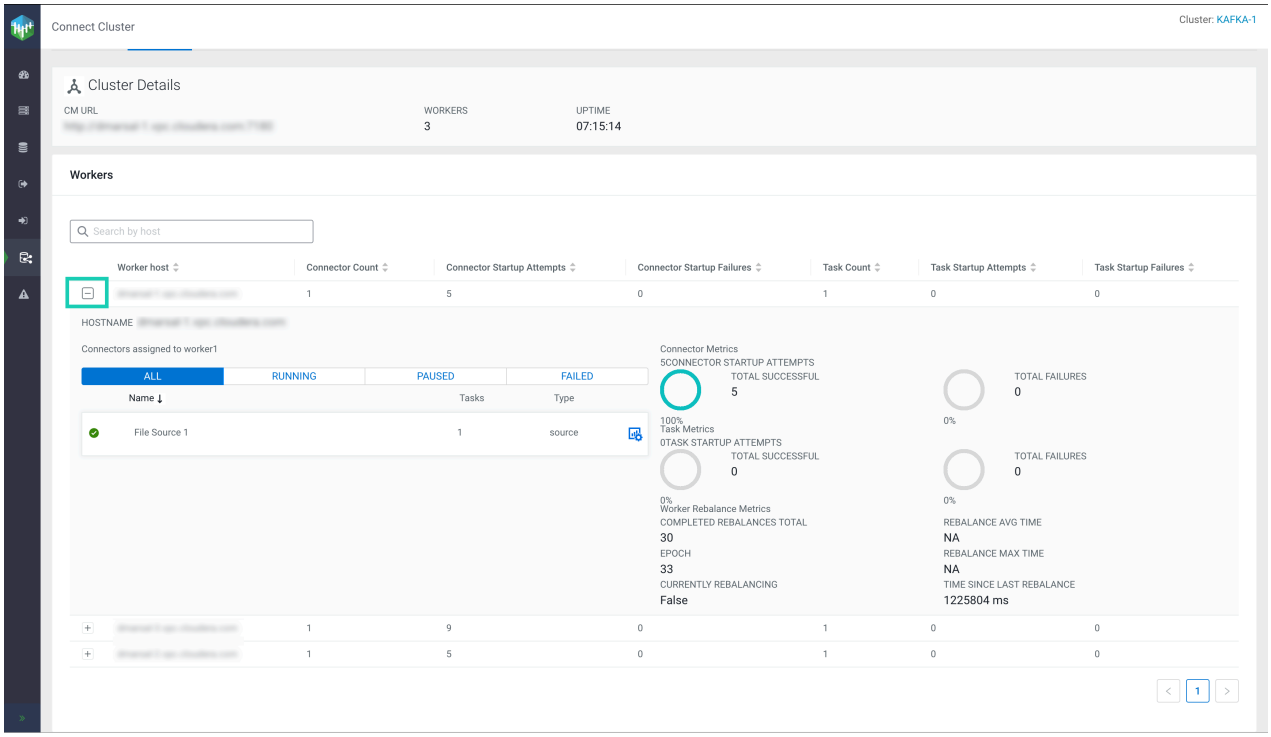

The Connector Profile

After you deploy a connector, you can monitor the details of the connector, modify its configuration, and carry out various management actions.

To monitor or manage a specific connector, click (Connector Profile) next to the connector's name. Clicking the icon redirects you to the **Connector Profile** page of the selected connector. The **Connector Profile** page enables you to monitor the details of the connector, monitor and manage its tasks, and edit its configuration. The page consists of two tabs **Connector Profile** and **Connector Settings**.

Connector Profile tab

The Connector Profile tab enables you to monitor details of the connector and its tasks. Additionally, a few basic management options are also available.

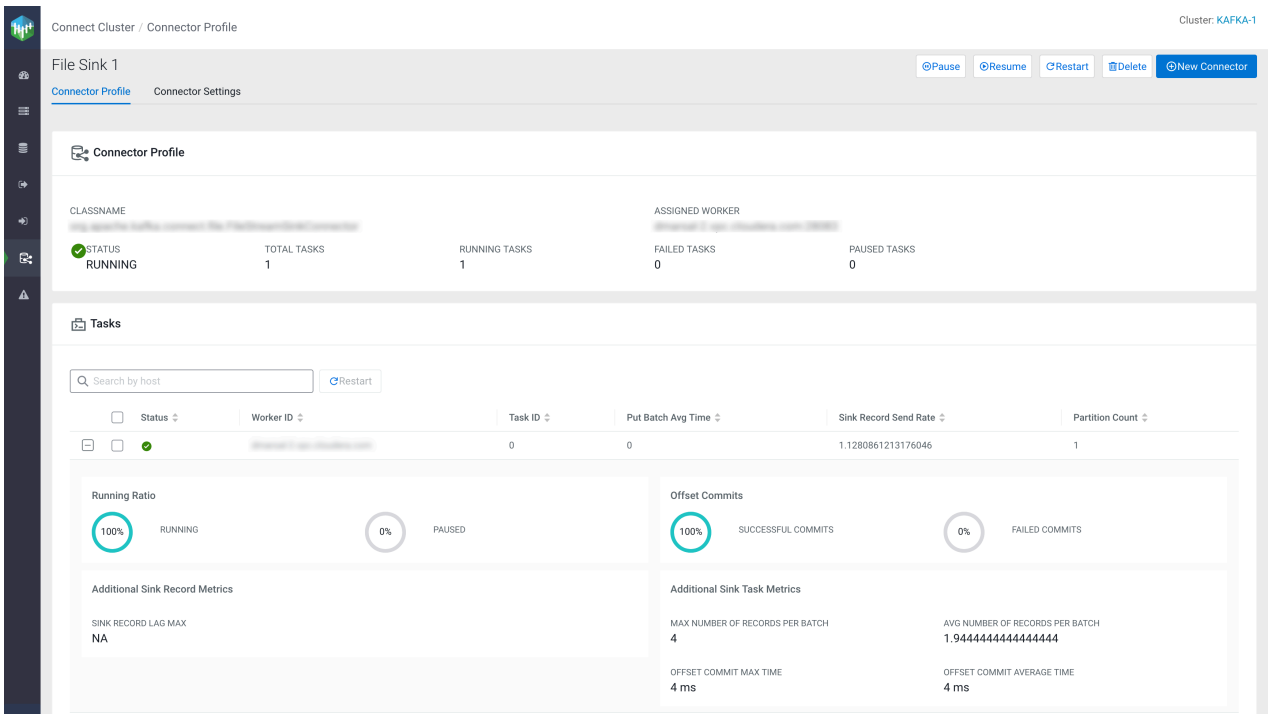

This tab is organized into multiple sections. The various sections enable you to do the following:

- The **Connector Profile** section provides you with details regarding the Classname, Assigned Worker, Status, Total Tasks, Running Tasks, Failed Tasks, and Paused Tasks.
- In the **Tasks** section, you can view and monitor Status, Worker ID, Task ID, and various other details regarding connector tasks.

  Clicking + next to a task displays detailed information and metrics about the selected task. In addition to viewing status and metrics, the **Tasks** section also allows you to restart a particular task. This can be done by selecting the task you want to restart and clicking the Restart option found within the **Tasks** section. The Search by host option enables you to search for particular task details by host.
- Using the buttons in the top right-hand corner you can pause, resume, restart, and delete the connector, or deploy a new connector.

Connector Settings tab

The Connector Settings tab enables you to review and edit the configuration of the connector. By default editing the connector configuration is disabled. To enabled editing, click Edit in the bottom left-hand corner. In addition to reconfiguring the connector, you can pause, resume, restart, and delete the connector, or deploy a new connector with the buttons in the top right-hand corner of the page.

Tip: The **Connector Settings** tab is identical in functionality to the **Connector Configuration** page in the **Connector Setup** wizard. For detailed information on the various options and features available related to configuration, see *Connector configuration features in Streams Messaging Manager*.

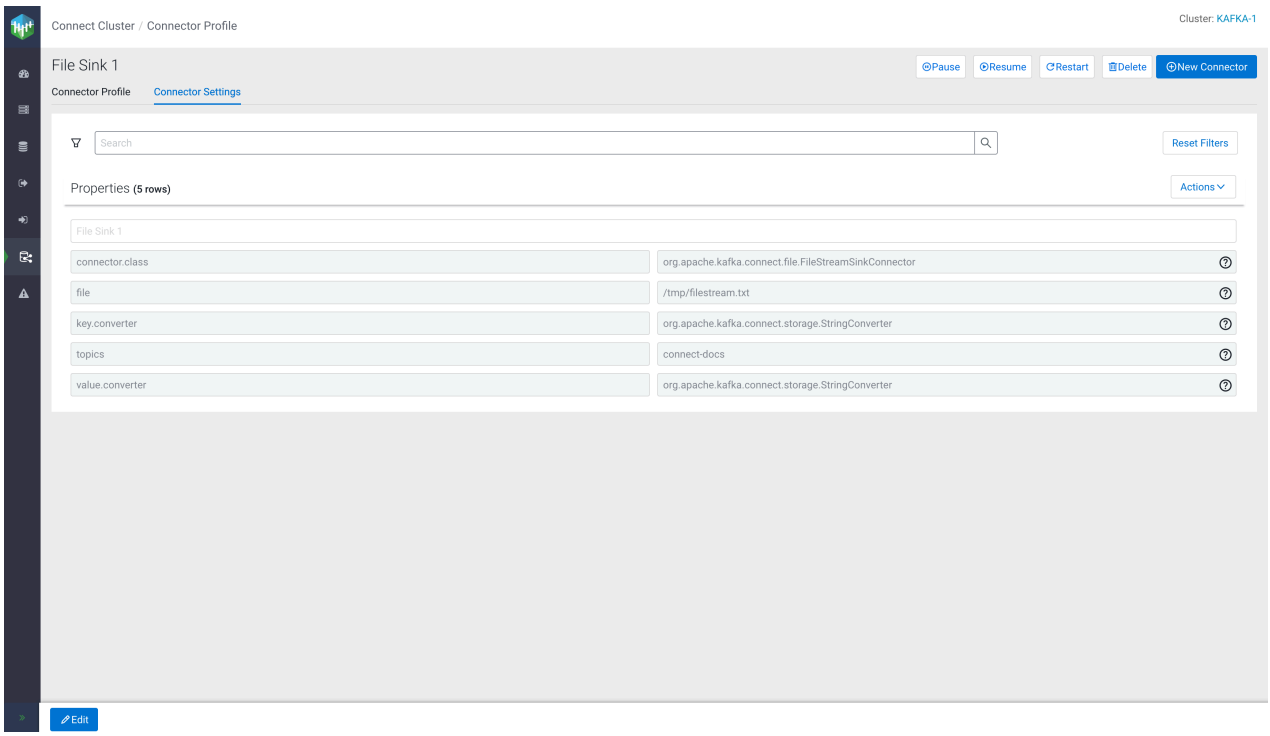

Related Information

Kafka Connect Setup
Integrating Kafka Connect with Streams Messaging Manager
Connectors
Deploying and managing Kafka Connect connectors in Streams Messaging Manager
Connector configuration features in Streams Messaging Manager

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Learn how to use the Streams Messaging Manager UI to deploy new Kafka Connect connectors using the Connector Setup wizard, as well as how to reconfigure, pause, resume, restart, and delete existing connectors. Additionally, learn about the various features and options Streams Messaging Manager provides for editing connector configurations.

Deploying a Kafka Connect connector in Streams Messaging Manager

Kafka Connect connectors are deployed in Streams Messaging Manager using the Connector Setup wizard. Learn how to deploy a new connector using the wizard.

Before you begin

- By default, you can use the Connector Setup wizard to deploy any connector that is shipped with Cloudera Runtime. The only exceptions are the FileStream example connectors (FileStreamSourceConnector and FileStreamSinkConnector). Although these connectors are part of the Kafka distribution and are shipped with Cloudera Runtime, they are not available by default in the Connector Setup wizard. This is because neither connector is considered production ready. If you want to deploy an instance of these connectors, you must install them first. For more information, see Installing Kafka Connect connectors.
- Third-party connectors can be deployed using the wizard, but must be installed first. For more information, see Installing Kafka Connect connectors.
- Before deploying any connector, Cloudera recommends that you review the documentation of the connector. For more information regarding each connector shipped in Cloudera, see Connectors.

Procedure

1. Click **Connect** in the navigation sidebar.
2. Click the New Connector option.

   This option is available on all Kafka Connect related pages of the Streams Messaging Manager UI. Clicking this option starts the **Connector Setup** wizard and redirects you to the **Select A Template** page. This page contains all connectors available for deployment in the form of selectable cards.

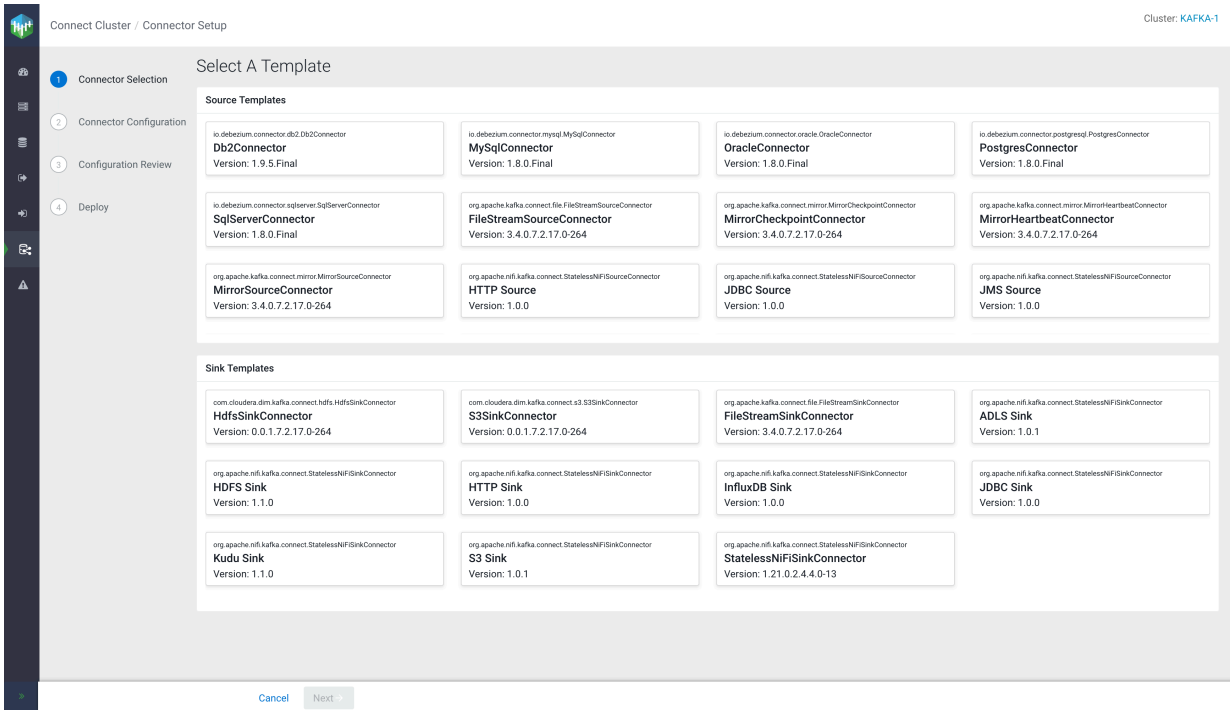

   Each card includes the following information about the connector:

   - The connector's fully qualified class name.
   - The connector's display name. If no display name is available, the card includes the unqualified classname.
   - The version of the connector.

   By default, the page includes all connectors shipped with Cloudera Runtime. Third-party connectors that you install manually are also visible on this page following installation.

3. Select a connector from the **Source Templates** or **Sink Templates** section.

   The Connector Configuration page appears.

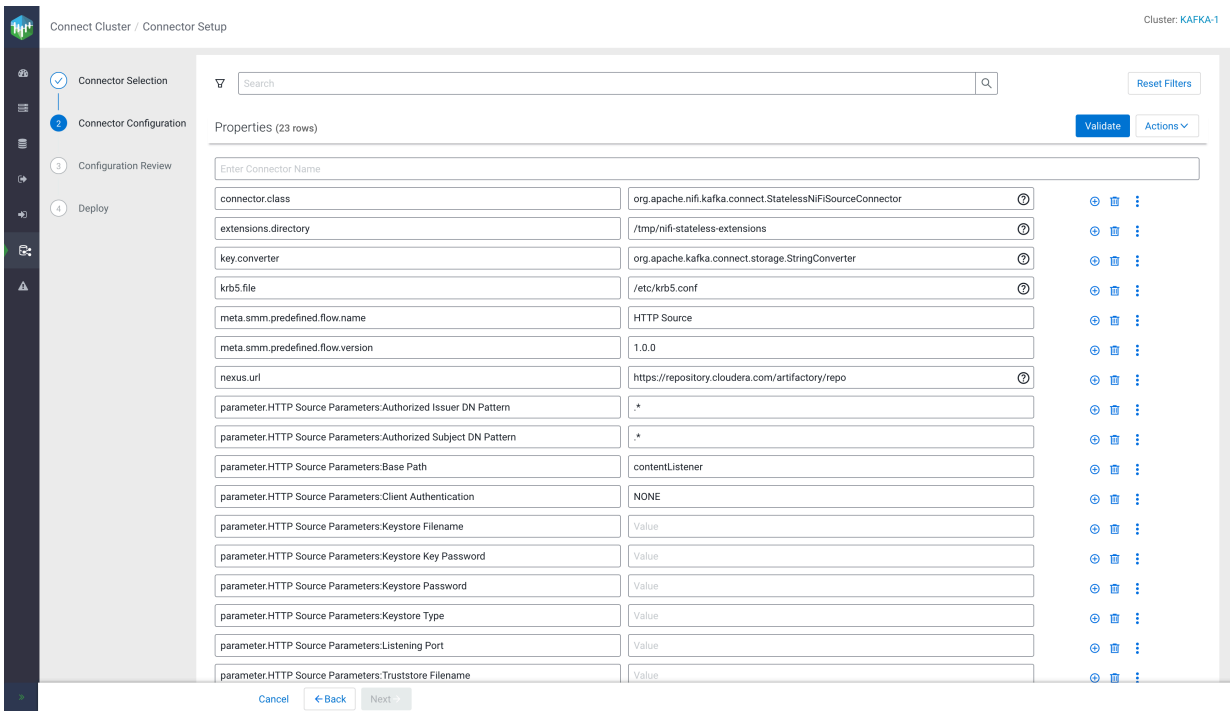

   Most connectors shipped with Cloudera Runtime come with a default configuration template to ease configuration. If a template is available for a specific connector, the property keys and values are automatically populated when you select the connector. The properties and values included in the templates depend on the selected connector. In general, the templates include all mandatory properties that are required for successful deployment. However, most connectors also have a number of additional properties that might not be part of the template. As a result, Cloudera recommends that you always review the documentation for the specific connector that you want to deploy.

4. Configure the properties of the connector.

   The Connector Configuration page includes various features and options that you can use to configure your connector. Each row in the **Properties** section represents the key and value of a property. Using the different buttons and other options available, you can add, delete, import, or otherwise modify the configuration. For more details on the configuration features available on this page, see *Connector configuration features in Streams Messaging Manager*.

5. Click Validate after you are done configuring the connector.

   Validating the configuration is mandatory when you deploy or modify a connector. If Streams Messaging Manager finds any errors in the configuration, the properties that contain errors are highlighted in red, and an error message with the details regarding the configuration issue is displayed. Resolve any errors until validation passes.

6. Click Next.

   The Next option is disabled until validation passes. After you click Next, the **Configuration Review** page appears.

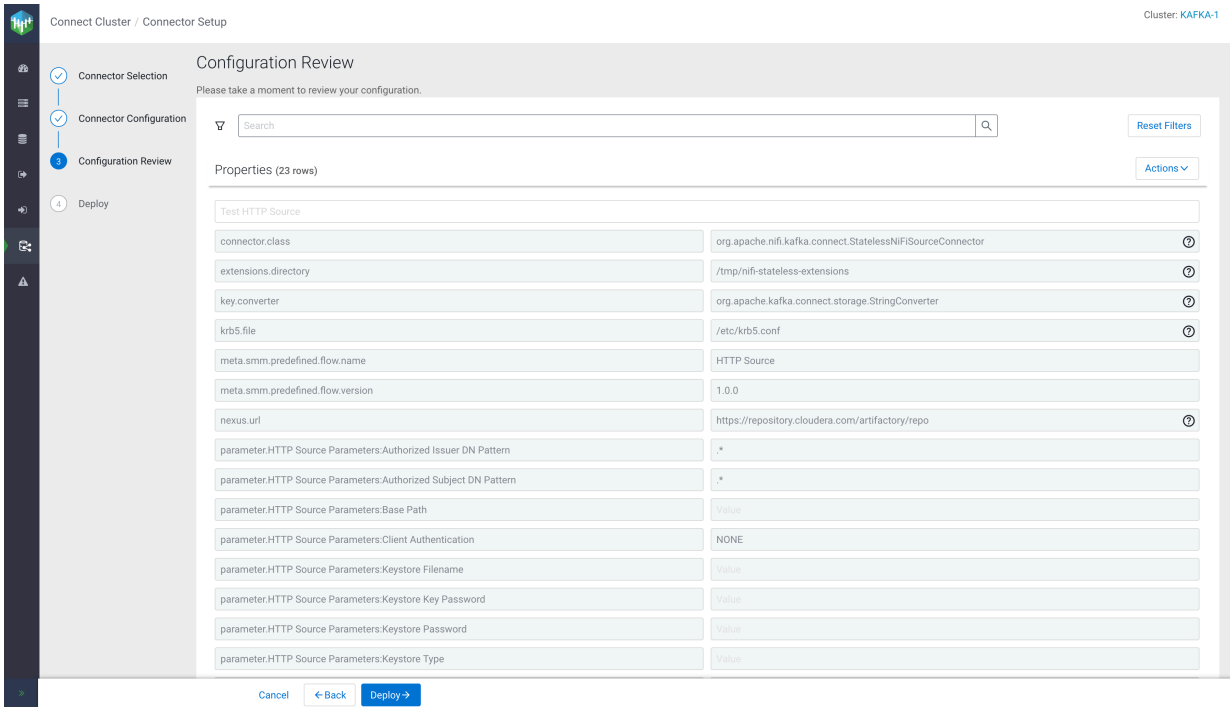

7. Review your connector configuration.

   The **Configuration Review** page allows you to review the connector configuration. Use this page to ensure that all properties are correctly configured for your use case.

   Most configuration options are disabled on this page. However, you can use the search and filtering options to search for properties. Additionally, you can export the connector configuration for later use with Actions Export . If you find any errors, go back to the **Connector Configuration** page, make any necessary changes, and revalidate the configuration before continuing.

8. Click Deploy to deploy the connector.

   After you click Deploy, the **Connector Deployment Status** modal is displayed that shows the status of the deployment.

   Note: Connector deployment is not instantaneous. You might need to wait a few seconds for deployment to finish.

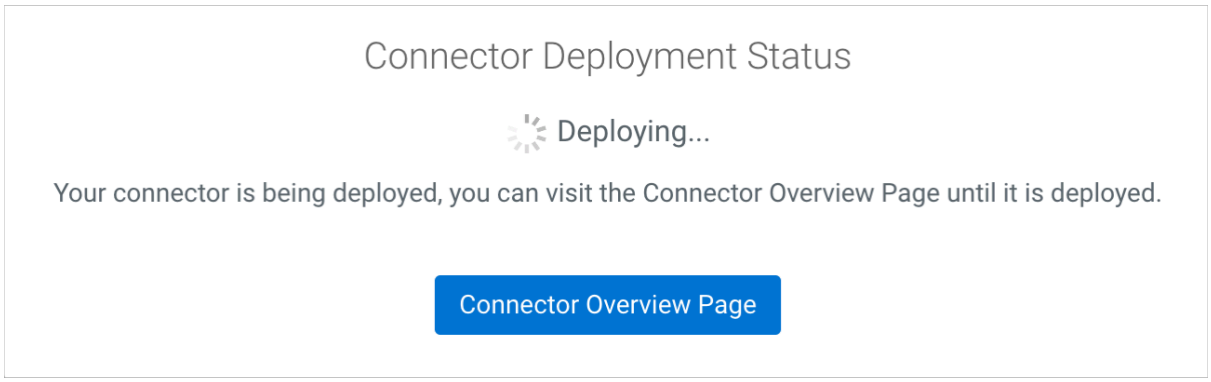

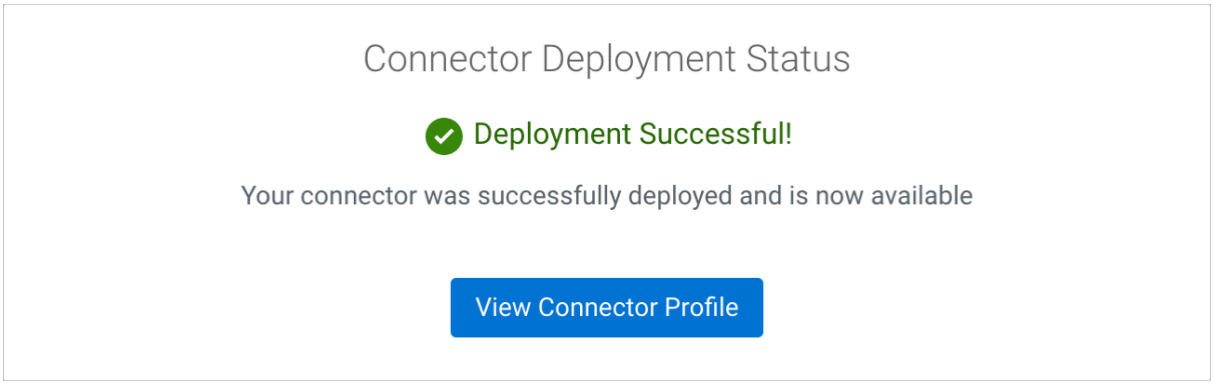

What to do next

After the connector is deployed, you can monitor the connector on the Connect Cluster page or the connector's profile page. For more information, see *The Kafka Connect UI*.

Related Information

Connectors

Streams Messaging Reference

The Kafka Connect UI

Connector configuration features in Streams Messaging Manager

Pausing, resuming, restarting, and deleting a Kafka Connect connector in Streams Messaging Manager

Learn how to pause, resume, restart, or delete an existing Kafka Connect Connector in Streams Messaging Manager.

Procedure

1. Click **Connect** in the navigation sidebar.
2. Identify the connector you want to manage.
3. Click (Profile) beside the connector.

4. Click Pause, Resume, Restart, or Delete.

   If you select Delete, you must confirm the action in a pop-up window.

   Tip: The Pause, Resume, and Restart options are also available if you hover over the connector on the **Connect Cluster** page.

Related Information

Connectors
Streams Messaging Reference
The Kafka Connect UI

Reconfiguring Kafka Connect connectors in Streams Messaging Manager

Learn how to edit the configuration of a running Kafka Connect connector using the Streams Messaging Manager UI.

Procedure

1. Click **Connect** in the navigation sidebar.
2. Identify the connector you want to modify.
3. Click (Profile) beside the connector.
4. Go to the Connector Settings tab.
5. Click Edit at the bottom-left corner of the page.

   All properties except connector name become editable.

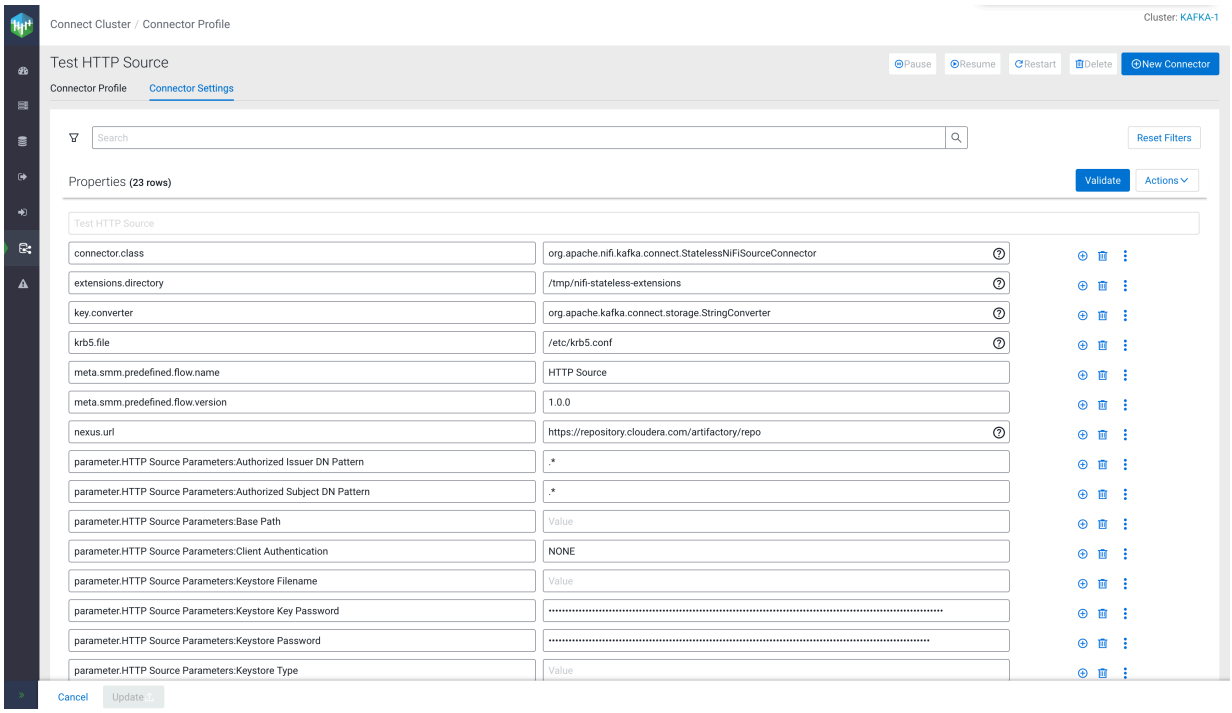

6. Configure the properties of the connector.

   The Connector Settings page includes various features and options that you can use to configure your connector. Each row in the **Properties** section represents the key and value of a property. Using the different buttons and other options available, you can add, delete, import, or otherwise modify the configuration. For more details on

the configuration features available on this page, see *Connector configuration features in Streams Messaging Manager*.

7. Click Validate after you are done configuring the connector.

   Validating the configuration is mandatory when you deploy or modify a connector. If Streams Messaging Manager finds any errors in the configuration, the properties that contain errors are highlighted in red, and an error message with the details regarding the configuration issue is displayed. Resolve any errors until validation passes.

8. Click Update .

Related Information

Connectors

Streams Messaging Reference

The Kafka Connect UI

Connector configuration features in Streams Messaging Manager

Connector configuration features in Streams Messaging Manager

Learn about the various configuration features and options that you can use when configuring Kafka Connect connectors in Streams Messaging Manager.

The Streams Messaging Manager UI includes two pages where you configure Kafka Connect connector properties. The **Connector Configuration** step of the **Connector Setup** wizard and the Connector Profile Connector Settings tab. Both pages include various features and options that are designed to help you with connector configuration. The following sections go over each of the configuration options available on the UI.

Note: The only difference between **Connector Configuration** and **Connector Settings** is that the former is used to configure new connectors, the latter is used to reconfigure already running connectors. Otherwise, the two pages are identical in functionality.

Configuring properties

Each row on the **Connector Configuration** page represents the key (name) and configuration value of a specific property. You can use the icons next to each property to add or remove properties. Clicking next to a property opens a context menu that includes additional configuration options. The options available in the context menu depend on the property. For example, you can edit and reset the connector.class property, but you cannot configure its type.

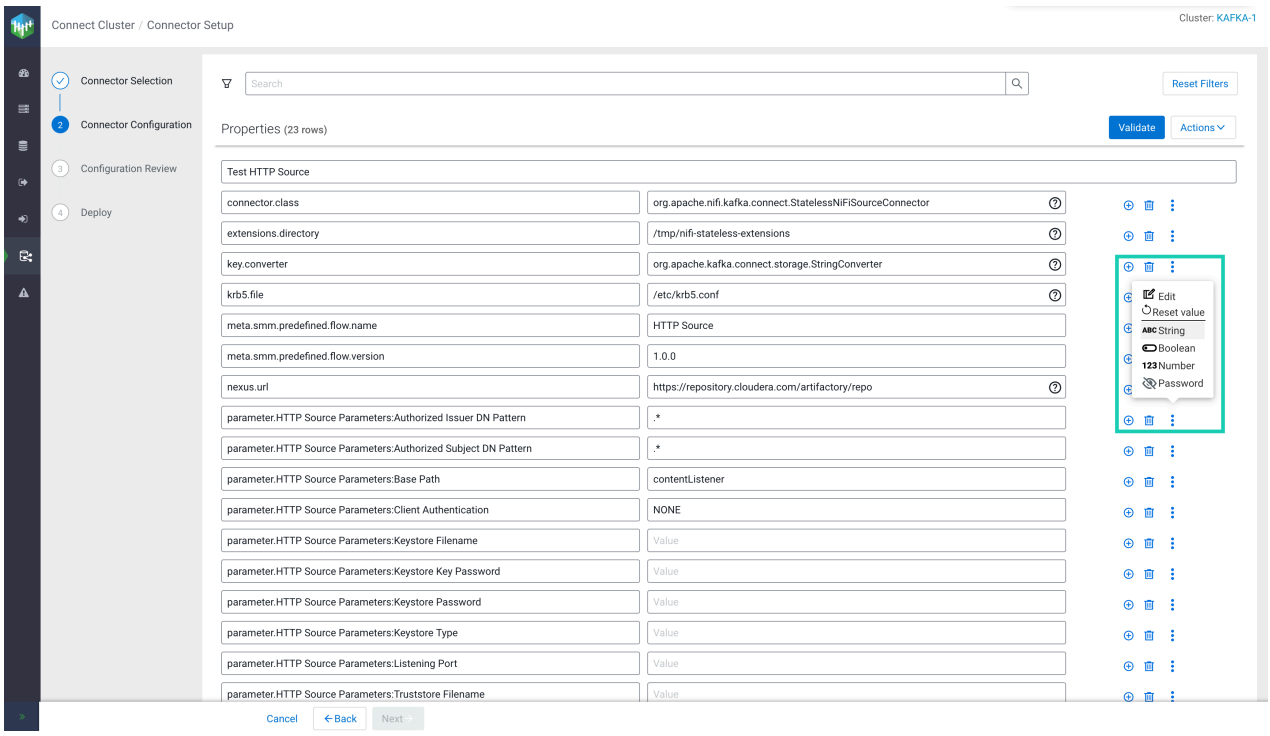

Adding properties

Click ⊕ to add a new property. An empty row appears for the new property. Enter property keys in the left field and property values in the right field. When you select a key field, a list of suggested properties appear that are valid for the connector. The list is automatically filtered when you start typing.

Deleting properties

Click 🗑 to delete a property. Use Actions Remove to remove all properties from the configuration with the exception of connector.class.

Editing properties

Click Edit to edit the configuration value of a property in a separate window. Use this option if the value you add is long, complex, or consists of multiple lines.

If you are deploying a NiFi Stateless Sink or Source connector, the modal that opens with for the flow.snapshot property is unique and includes the Browse... and Save and Enhance options. Browse... enables you to upload a flow definition JSON from your machine, Save and Enhance adds the parameters specified in your flow definition to the connector configuration. For more information on flow.snapshot configuration, see *Configuring flow.snapshot for Stateless NiFi connectors*.

Resetting properties

You can use Reset value to reset the value of a single property. Alternatively, you can use Actions Reset to reset the full configuration. The reset options behave uniquely depending on the configuration page that you use them on.

When deploying new connectors using the **Connector Configuration** page of the wizard:

- Reset value resets the value to the default value set in the configuration template. This option is not available for properties that are not part of the template.

- Actions Reset resets all configuration properties and reverts the configuration to the default configuration template.

When editing the configuration of existing connectors on the Connector Profile Connector Settings page:

- Reset value resets the value to the last value that the connector was deployed with. This option is not available for new properties that you add to the configuration while editing the configuration.
- Actions Reset resets all configuration properties and reverts to the configuration that the connector was last deployed with.

Type configuration

The ABC String , Boolean, 123 Number, and Password options enable you to set the type of the property. Setting the type of a property changes the property's value field depending on the type you select.

- ABC String sets the type of the property to string and changes the input field to a single line text box.
- 123 Number sets the type to number and changes the input field to a text box that only allows negative and positive whole numbers and decimals.
- Boolean sets the type of the property to boolean and changes the input field to a checkbox.
- Password sets the type of the property to password.

  Selecting Password hides (locks) the value on the UI and stores it in a secure manner. Once the connector is deployed, properties marked as passwords are encrypted and stored securely in an internal Kafka topic. The actual values only resolve at runtime. These values cannot be retrieved or otherwise read from the configuration of the connector. In most cases, you use this option to hide passwords or other types of sensitive credentials found in connector configurations, however, you can use this option for any property value that your organization considers sensitive.

  Tip: The password option is the UI implementation of the Kafka Connect Secrets storage feature. For more information on the feature, and how Kafka Connect handles values you mark as sensitive data, see *Kafka Connect Secrets Storage*.

Property search and filtering

At the top of the page you can find a search bar that you can use to search and filter for connector properties. Search only works for property keys, values are disregarded. When you search for a string, the **Properties** section of the page is automatically filtered and only the relevant properties are shown. The number of matches is displayed next to **Properties**.

In addition to searching, you can filter properties based on their group and importance. To do so, click the filter icon and select the relevant group and importance that you want to search for. To clear all applied filters, click Reset Filters.

Viewing property help

Hovering over (Help), found next to some property values, displays information about that property. What information is displayed is property dependent, but at minimum you are presented with a description of the property.

If available, the group, importance, and expected type of a property is also displayed. The icon is only available for properties that have their metadata (description, type, group, and so on) defined.

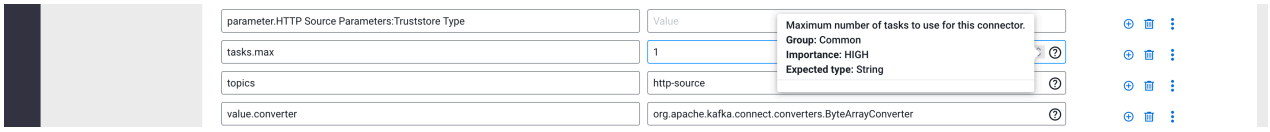

Tip: The information displayed in the help context menu is the property metadata defined with `ConfigDef`. If you are developing your own connector and want help information to be available in Streams Messaging Manager, ensure that you define property metadata in your connector code using the `ConfigDef` class. Additionally, if you specify a display name for the property in `ConfigDef`, Streams Messaging Manager will print the display name above the key field of the property.

Importing configurations

To import a configuration, click Actions Import . Clicking Import opens the **Connector Config…** modal.

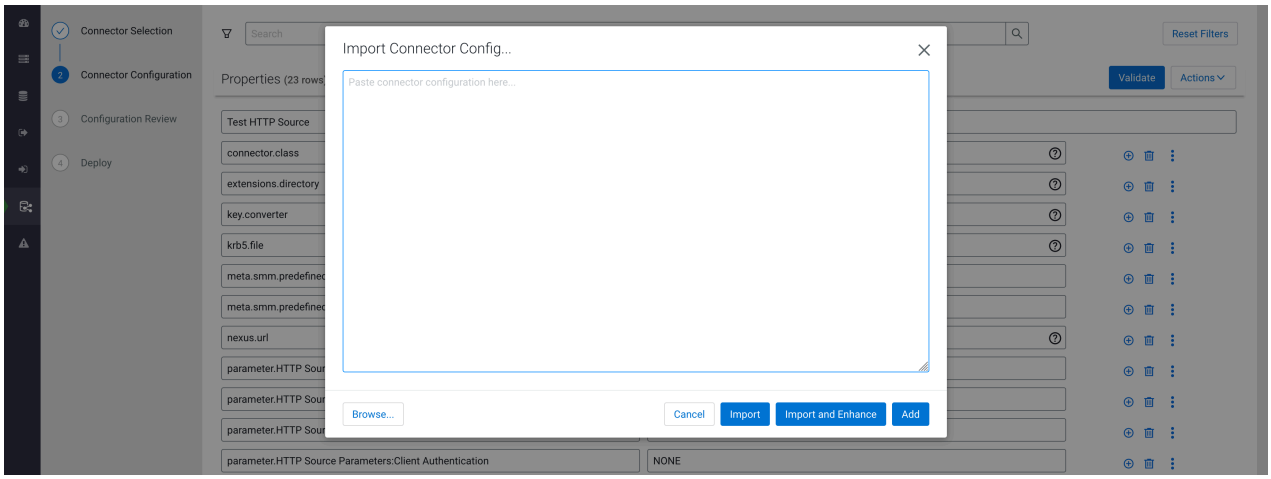

Using the modal you can:

- Type or paste your configuration.
- Use Browse… to upload the contents of a file from your machine.

The configuration that you add can be a full configuration that contains all properties necessary for the connector. Alternatively, it can be a partial configuration that contains a select number of properties that you want to add to the configuration. The configuration you include in the text box must be in JSON format. For example:

```
{
  "tasks.max": 1,
  "key.converter": "org.apache.kafka.connect.storage.StringConverter",
  "value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConvert
er",
  "value.converter.passthrough.enabled": true,
  "value.converter.schema.registry.url": "http://schema-registry:9090/api/v1
",
  "topics": "avro_topic",
  .
  .
  .
```

```
}
```

After adding properties, you can use the Import, Import and Enhance, or Add options to import the configuration. Each option imports the properties in a different way.

- Import

  This option populates the **Properties** section with the keys and values you added in the **Import Connector Config...** modal. This option overwrites all existing properties in the configuration.
- Import and Enhance

  This option populates the **Properties** section with the keys and values you added in the **Import Connector Config...** modal. Additionally, properties that are most likely needed for the respective connector are also added.

  Import and Enhance is used specifically for the NiFi Stateless Source and NiFi Stateless Sink connectors. When this option is used, Streams Messaging Manager parses the dataflow specified in flow.snapshot, extracts all parameters that are available in the dataflow, and adds them to the connector configuration. This option overwrites all existing properties in the configuration.
- Add

  This option amends the existing configuration with the properties from the **Import Connector Config...** modal. Use this option if you want to batch import additional properties to your configuration. For example, you can use this option to batch add security properties.

Exporting Configurations

To export a configuration, click Actions Export . Clicking Export opens the **Configuration of the connector** modal.

- Click Copy to Clipboard to copy the configuration to the clipboard.
- Click Export to file to download the configuration as a JSON file.

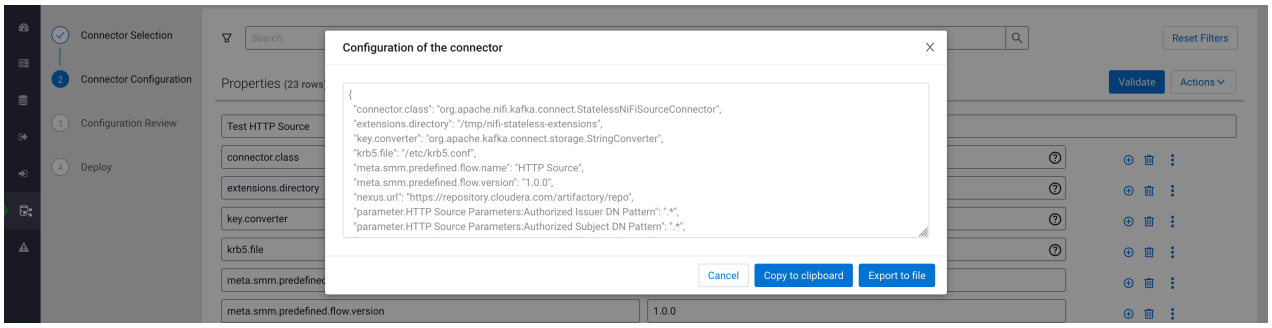

Validating configurations

Before you can deploy or redeploy a connector, you must validate the configuration. This is done using Validate. If Streams Messaging Manager finds any errors in the configuration, the properties that contain errors are highlighted in red, and an error message with the details regarding the configuration issue is displayed.

If the configuration is missing a mandatory property, the name of the missing property is displayed in an **Errors** section on the top of the page. You can add missing mandatory properties by clicking Add missing configurations. This option adds both property keys and values. Values are populated using the following logic.

- If the connector was previously deployed and you are updating the configuration, the default value used will be the value that the connector was last deployed with.
- If you are configuring a new connector, the default value from the sample configuration is used.
- If you are configuring a new connector and there is no sample configuration or the sample configuration does not contain the property, the value defined in the `ConfigDef` class of the connector is used.

If the connector configuration is filtered when you run the validation and Streams Messaging Manager finds errors with properties that are filtered, an error message is displayed on the top of the page notifying you that not all validation errors are visible. In a case like this, click Reset Filters to view all validation errors.

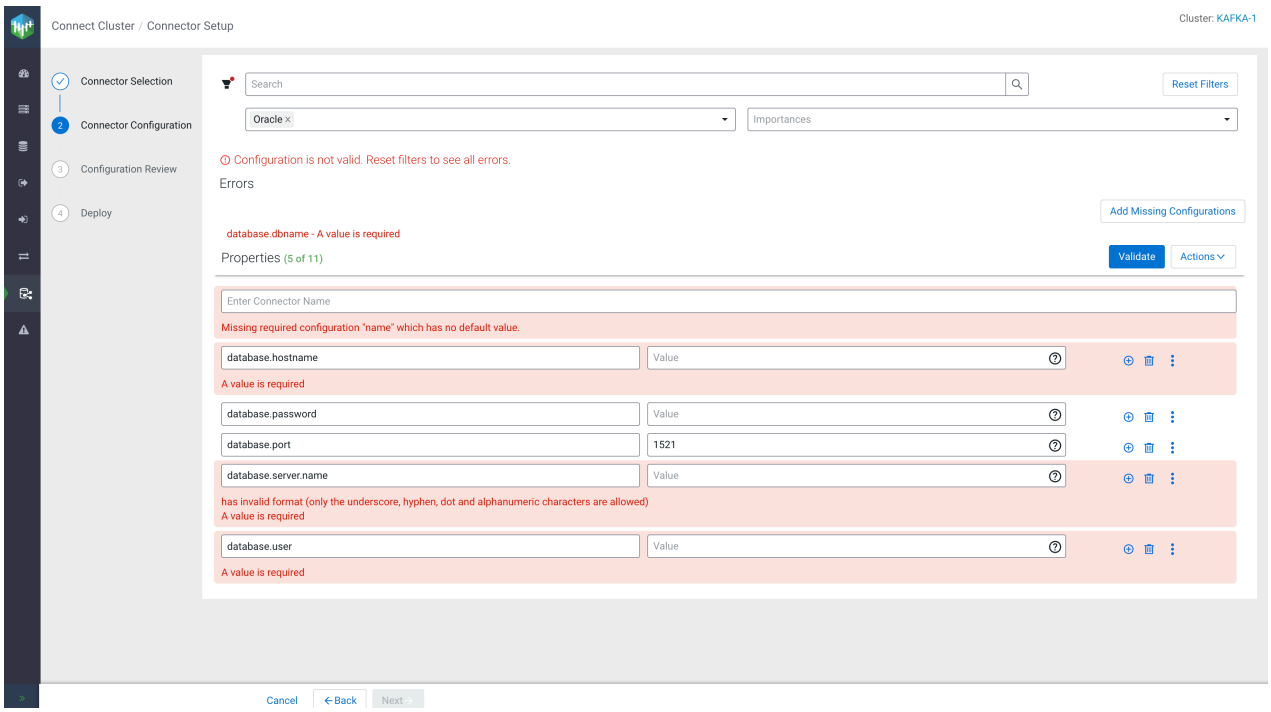

Related Information

Connectors

Streams Messaging Reference

The Kafka Connect UI

Kafka Connect Secrets Storage

Configuring flow.snapshot for Stateless NiFi connectors

Managing and monitoring Kafka rebalancing with Cruise Control

Get started with Cruise Control in Streams Messaging Manager. Cruise Control is automatically integrated with Streams Messaging Manager. It provides user-friendly dashboards for monitoring the rebalancing process and state of the Kafka cluster. You can also manage and customize the rebalancing process, based on the configurations in Cloudera Manager.

Authorizing users to access Cruise Control in Streams Messaging Manager

Learn how to authorize users to access the Cruise Control UI in Streams Messaging Manager. This is required to enable users access the UI or manage rebalancing. You can set access by assigning users the required authorization levels.

About this task

The integration between Streams Messaging Manager and Cruise Control is automatic. The Cruise Control User Interface (UI) can be accessed through an Streams Messaging Manager instance running on the same cluster as Cruise Control. In environments where security is disabled, anyone will be able to access Cruise Control in Streams Messaging Manager. In secured environments, users need to be added to the authorization level groups before they can see the Cruise Control UI. The group a user belongs to determines what they are allowed to do in the UI.

Procedure

1. Select the Cruise Control service in Cloudera Manager.
2. Click Configuration.
3. Search for auth_admins, auth_users or auth_viewers based on which authorization level is needed.
4. Assign users to the required authorization level.

   Based on the authorization level, users have the following access to Cruise Control in Streams Messaging Manager:

   - Viewer level users: can only view the metrics
   - User level users: includes Viewer level permissions and manage goals, estimations
   - Admin level users: includes Viewer and User level permissions, starting and stopping rebalance

   Note: The assigned authorization level for a user can be checked with the permissions endpoint:

   ```
   curl -X GET "http://<cruise_control_hostname>:8899/kafkacruisecontro
   l/kafkacruisecontrol/permissions”
   ```

5. Click Save changes.

Results

Cruise Control is visible in Streams Messaging Manager. Users can access the Cruise Control UI using the on the navigation sidebar.

Cruise Control dashboard in Streams Messaging Manager UI

Learn about the Cruise Control section in the Streams Messaging Manager UI, which you can use to monitor and manage the rebalancing process of Kafka clusters.

The Cruise Control UI in Streams Messaging Manager allows you to monitor the state and operations of the Cruise Control service, including the latest state of the Kafka cluster known to Cruise Control. The Rebalance Wizard allows admins to start rebalance operations, specifying goals, anomaly detectors and estimations that should be used for generating the rebalance proposals. You can access the Cruise Control UI from the left navigation pane of Streams Messaging Manager. Clicking on loads the **Cruise Control Dashboard**.

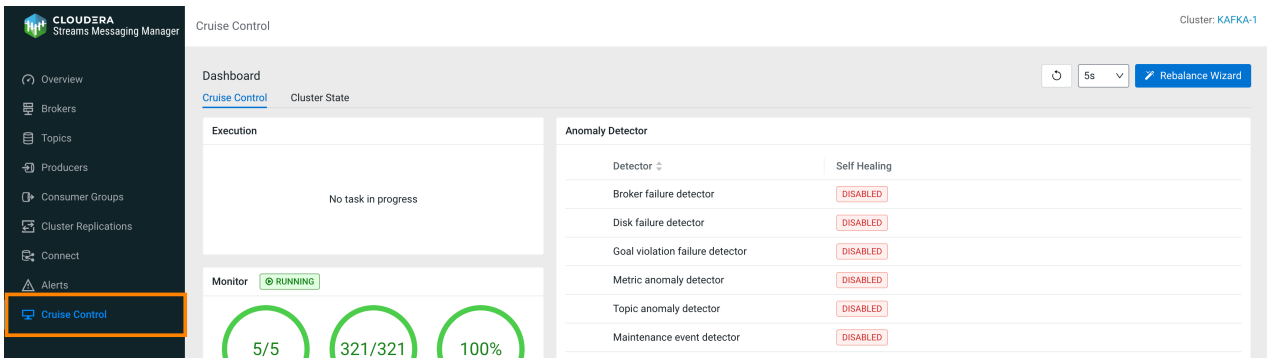

Note: In case Cruise Control is not available in the left navigation pane, review the instruction in *Authorizing users to access Cruise Control in Streams Messaging Manager*.

Cruise Control Dashboard

The **Cruise Control Dashboard** provides an overview of the Cruise Control state, including the following information:

- Any operations that are currently being executed by Cruise Control and their progress

- Status of internal Cruise Control metrics (valid windows, valid partitions and coverage)
- List of goals configured in Cruise Control and their status (ready or not ready)
- List of anomaly detectors and their status (disabled or enabled)

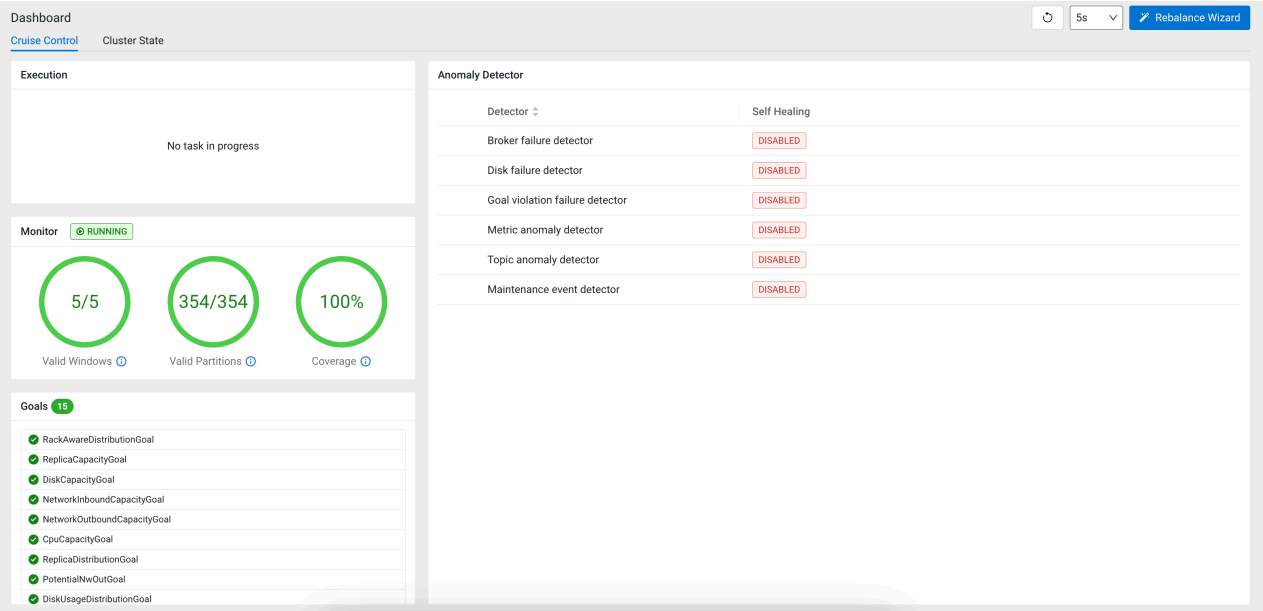

You can manually refresh the **Dashboard** using ↺, or you can set a refresh interval by selecting one of the options available with ⌄.

Execution

The **Execution** tile of the **Dashboard** allows you to track and monitor the tasks being executed by Cruise Control (including rebalancing tasks that are triggered by the enabled goals or other operations started through the REST API). If the goals are fulfilled and there is no ongoing execution, the No task in progress message is shown. Only users with Admin level can stop the execution process.

Monitor

The **Monitor** tile always shows the ratio of **Valid Windows**, **Valid Partitions**, and **Coverage** of the Kafka cluster.

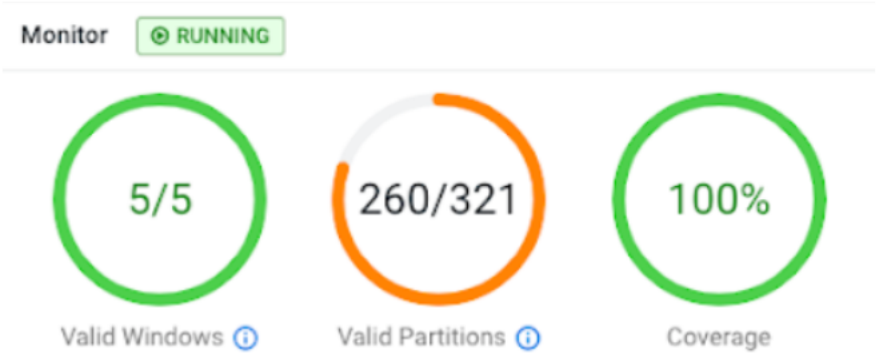

Valid windows

The metrics that Cruise Control collects about brokers and partitions in a Kafka cluster are aggregated in windows of time. A window represents a time range. There is a minimum number of metric samples that a window should have to be considered "valid" by Cruise Control. Only valid windows can be used for balance proposals and anomaly detection.

The validity of a window is determined based on valid brokers, valid partitions and overall valid metric samples count. The parameters of this mechanism can be configured in Cloudera Manager with the following Cruise Control configuration properties:

Property	Description
num.broker.metrics.windows	The maximum number of broker windows the Load Monitor keeps. Each window covers a time window defined by the broker.metrics.window.ms configuration property.
broker.metrics.window.ms	The size of the window in milliseconds to aggregate the Kafka broker metrics. The window must be greater than the value of the metric.sampling.interval.ms configuration property. metric.sampling.interval.ms is the duration until metrics are collected and sampled. When an interval is ready, another interval starts and tries to collect the metrics from the brokers.
min.samples.per.broker.metrics.window	The minimum number of metric samples a valid broker window should have. If a broker does not have enough samples in a broker window, the broker is removed from the window due to insufficient data.
max.allowed.extrapolations.per.broker	The maximum number of extrapolations allowed for each broker. A broker is considered as invalid if the total number extrapolations in all the windows goes above the maximum allowed number. Load Monitor contains information about the status of the linear regression model, which is used to estimate the CPU utilization, number of valid windows and partitions. If there are flawed brokers, not all windows have metrics. Cruise Control extrapolates these metric values.

Valid partitions

Cruise Control aggregates metrics about each partition in the connected Kafka cluster. There is a minimum number of metric samples a valid partition should have. The validity of a partition is determined based on the valid metric samples count. The mechanism can be configured in Cloudera Manager with the following Cruise Control configuration properties:

Property	Description
num.partition.metrics.windows	The maximum number of partition windows the Load Monitor keeps. Each window covers a time window defined by the partition.metrics.window.ms configuration property.

Property	Description
partition.metrics.window.ms	The size of the window in milliseconds to aggregate the Kafka partition metrics. The window must be greater than the value of the metric.sampling.interval .ms configuration property. metric.sampling.interval.ms is the duration until metrics are collected and sampled. When an interval is ready, another interval starts and tries to collect the metrics from the brokers.
min.samples.per.partition.metrics.window	The minimum number of metric samples a valid partition window should have. If a partition does not have enough samples in a partition window, the topic of the partition is removed from the window due to insufficient data.
max.allowed.extrapolations.per.partition	The maximum number of extrapolations allowed for each partition. A partition is considered as invalid if the total number extrapolations in all the windows goes above this number. Load Monitor contains information about the status of the linear regression model, which is used to estimate the CPU utilization, number of valid windows and partitions. If there are flawed partitions, not all windows have metrics. Cruise Control extrapolates these metric values.

For more information on the configuration properties, see the Cruise Control upstream documentation

Coverage

The coverage of this cluster model through monitored partitions ratio, showing how representative the cluster is.

List of Goals

The **Goals** tile shows the list of enabled goals and indicates if the goals have been fulfilled or not with checkmarks. The list of goals can be configured in Cloudera Manager. For more information, see *Setting capacity estimates and goals*.

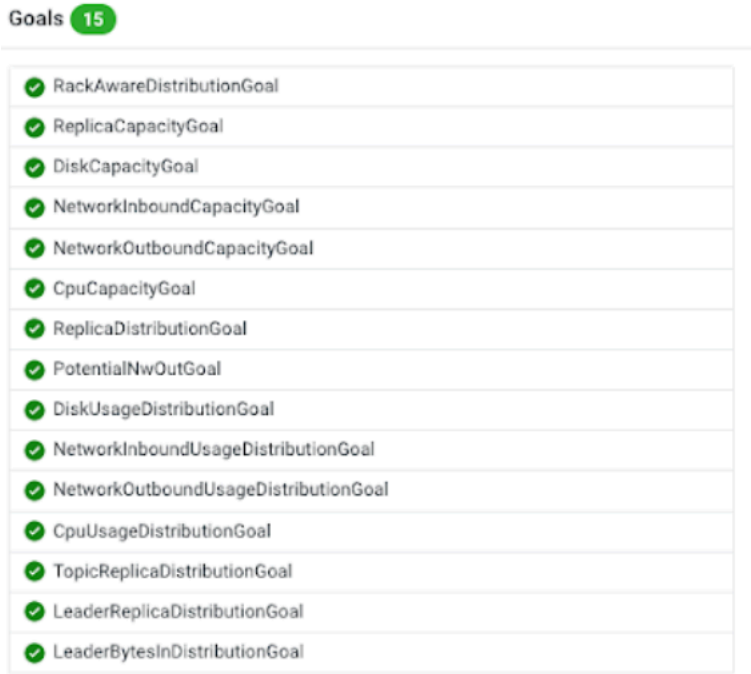

List of Anomaly Detectors

The **Anomaly Detector** tile shows the list of anomaly detectors and whether self-healing is enabled for the given detector. If there are any anomalies found by the detectors, the anomalies are listed under the corresponding detector.

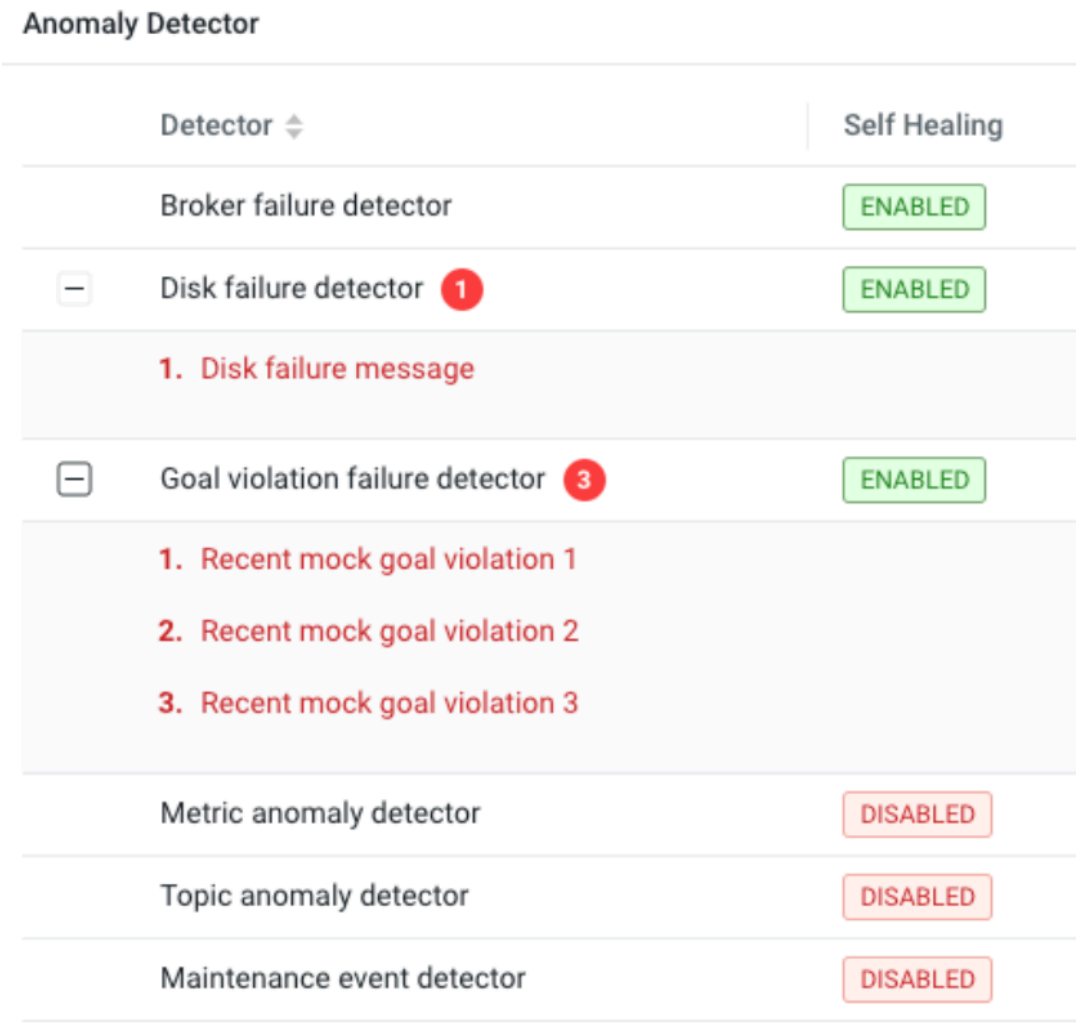

Cluster State Dashboard

The **Cluster State Dashboard** provides an overview of the Kafka cluster specific metrics related to leaders, replicas, and directories. The metrics shown on the **Dashboard** are based on the number of leaders, online, offline and out-of-sync replicas, online and offline directories within the brokers available on the Kafka cluster. You can use these metrics to monitor the Kafka cluster state.

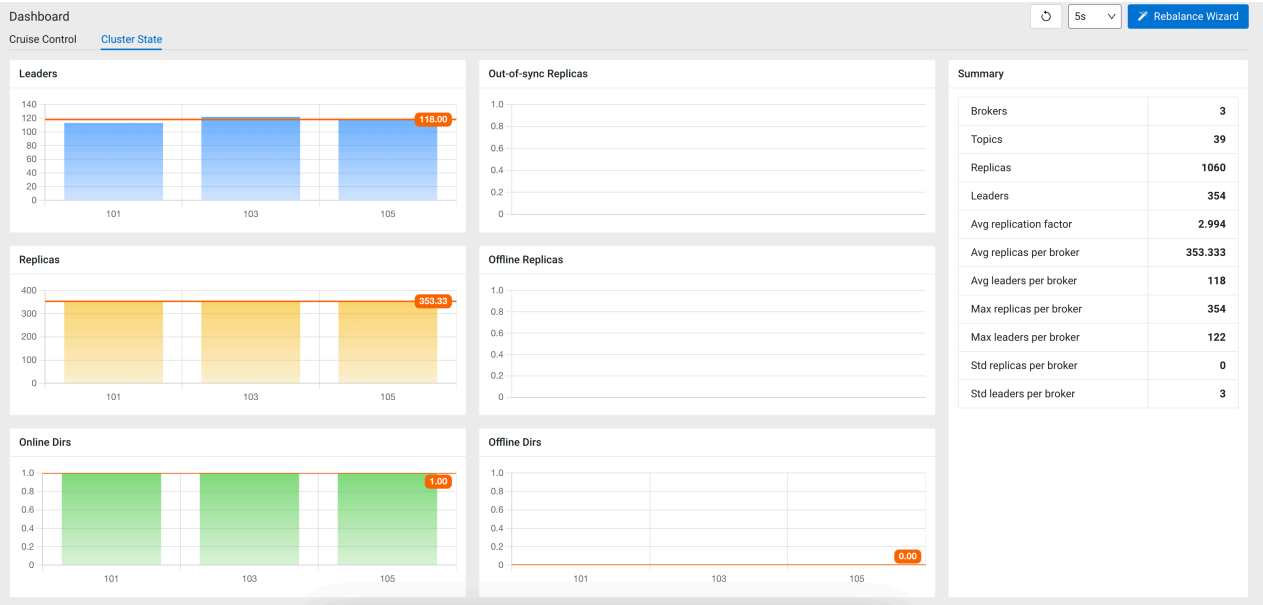

In the **Summary** section, you can review the general information of the number of brokers, topics, replicas and leaders within the Kafka cluster, and more metrics that show the average, maximum and standard ratio of leaders per broker and replicas per broker.

Related Information

Setting capacity estimations and goals

Authorizing users to access Cruise Control in Streams Messaging Manager

Using the Rebalance Wizard in Cruise Control

You can start and manage the rebalancing process of the Kafka cluster using the Rebalance Wizard.

About this task

You can use the Rebalance Wizard, located beside the **Dashboard**, used to trigger the rebalancing process of the Kafka cluster. You must have Admin-level permission to use the Rebalance Wizard.

To restore the default settings, you can use Reset in the Rebalance Wizard, and go back to the Dashboard using the Dashboard link.

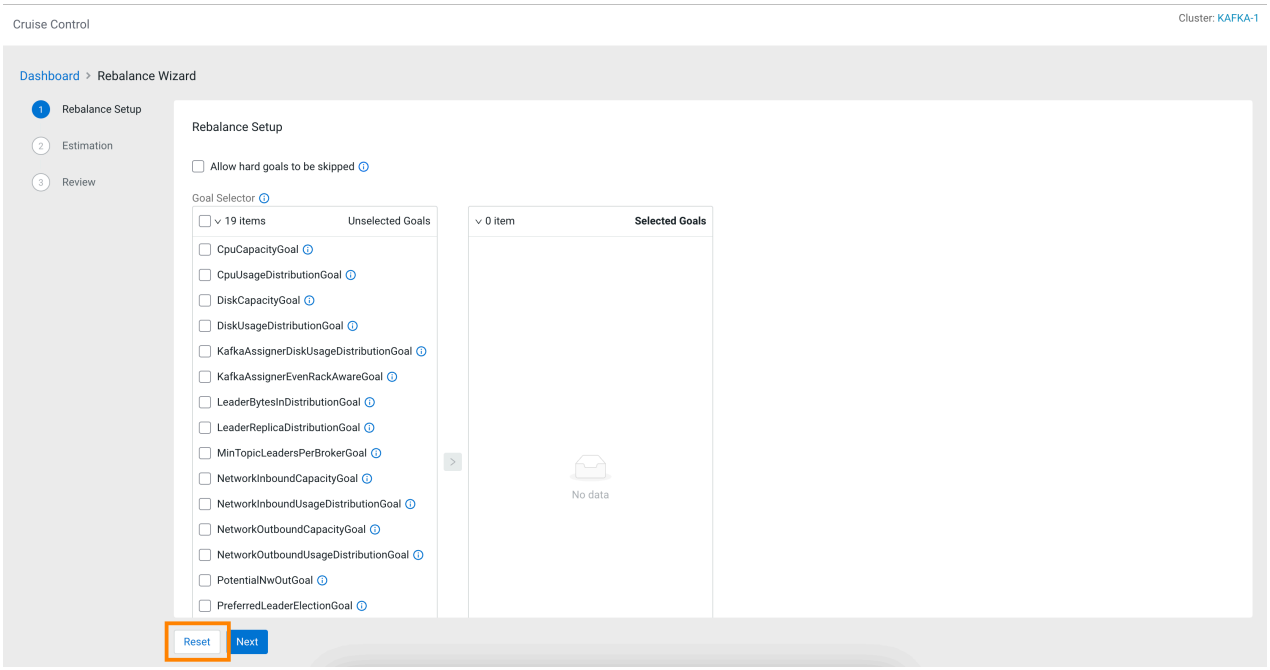

Procedure

1. Click Rebalance Wizard on the **Dashboard**.
2. Configure the Rebalance Setup to determine how the rebalance request is executed.

   The configurations can be used to customize the rebalancing process based on your requirements. If the configurations are not changed and the fields are empty, Cruise Control completes the rebalancing process with the default settings. The rebalance setup and goal list is based on the configuration in Cloudera Manager.

   Tip: Cloudera recommends to configure the following configurations even if the other configurations stay as default:

   - Disk rebalance to ensure that the disks or the entire brokers are rebalanced
   - Excluded topics to run rebalance only for those topics that are explicitly selected
   - Destination brokers to run rebalance only for a group of brokers
3. Click Next to generate the proposal.

   Note: If the proposal generation fails, an error message is displayed with the incorrect configuration. After correcting the configurations, you can move forward to **Estimation**.

4. Review the **Estimation** of the rebalancing proposal.

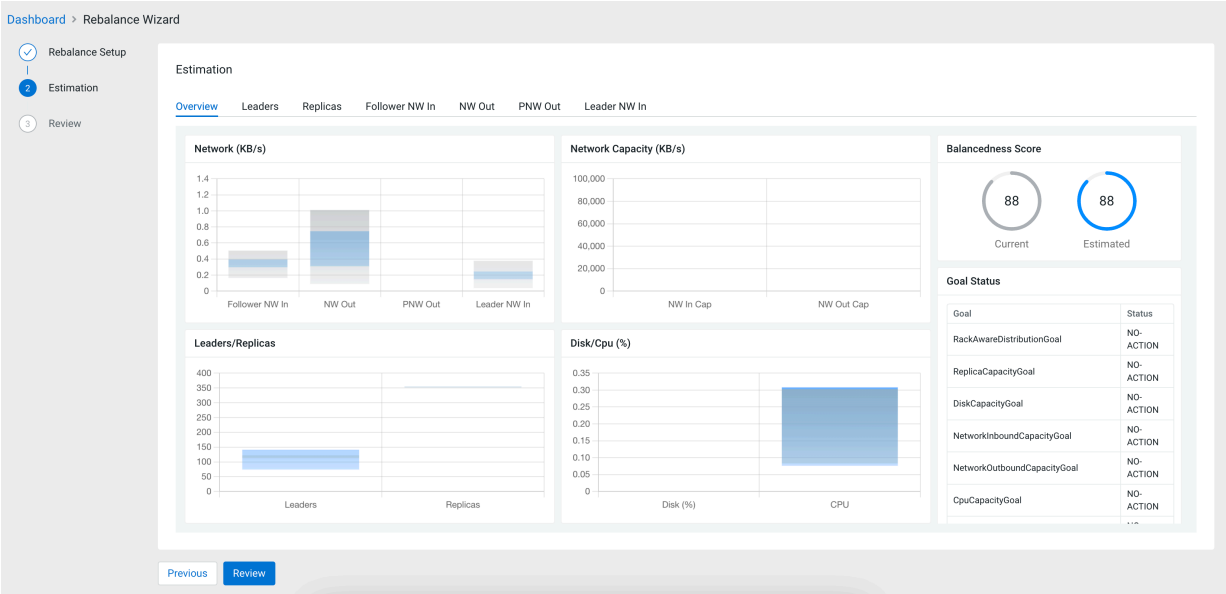

The **Overview** tab contains diagrams about the Network, Network Capacity, Leaders and Replicas ratio and Disk CPU. The **Balancedness Score** shows how efficiently the resources are balanced by comparing the current state and the estimated state that was configured during the **Rebalance Setup**. You can also review the list of goals and the set actions that should be triggered when the specific goal is violated under **Goal Status**. The **Movements** table gives you an overview of the estimated movement to achieve the ideal rebalancement.

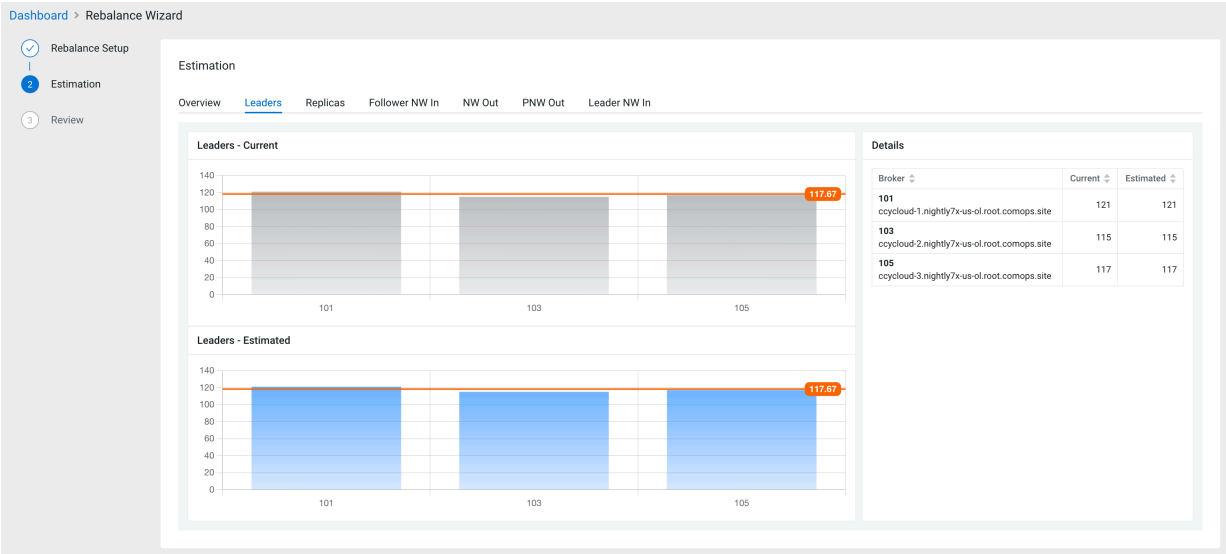

The **Leaders**, **Replicas**, **Follower NW in**, **NW Out**, **PNW Out** and **Leader NW in** tabs compare the current state and the estimated state after applying the new rebalancing configured in the **Rebalance Setup**. On each tab, the **Details** table shows the raw data that is visualized on the bar chart.

As the goal of the Estimation step is to review and assess the result of the rebalancing process, the configurations can be set again at the Rebalance Setup by clicking Previous.

5. Click Review if the estimation overview showed the desired results.

   The **Review** shows the specified configurations as the list of goals for the rebalance request. At this step, you still have the opportunity to change the rebalancing setup.

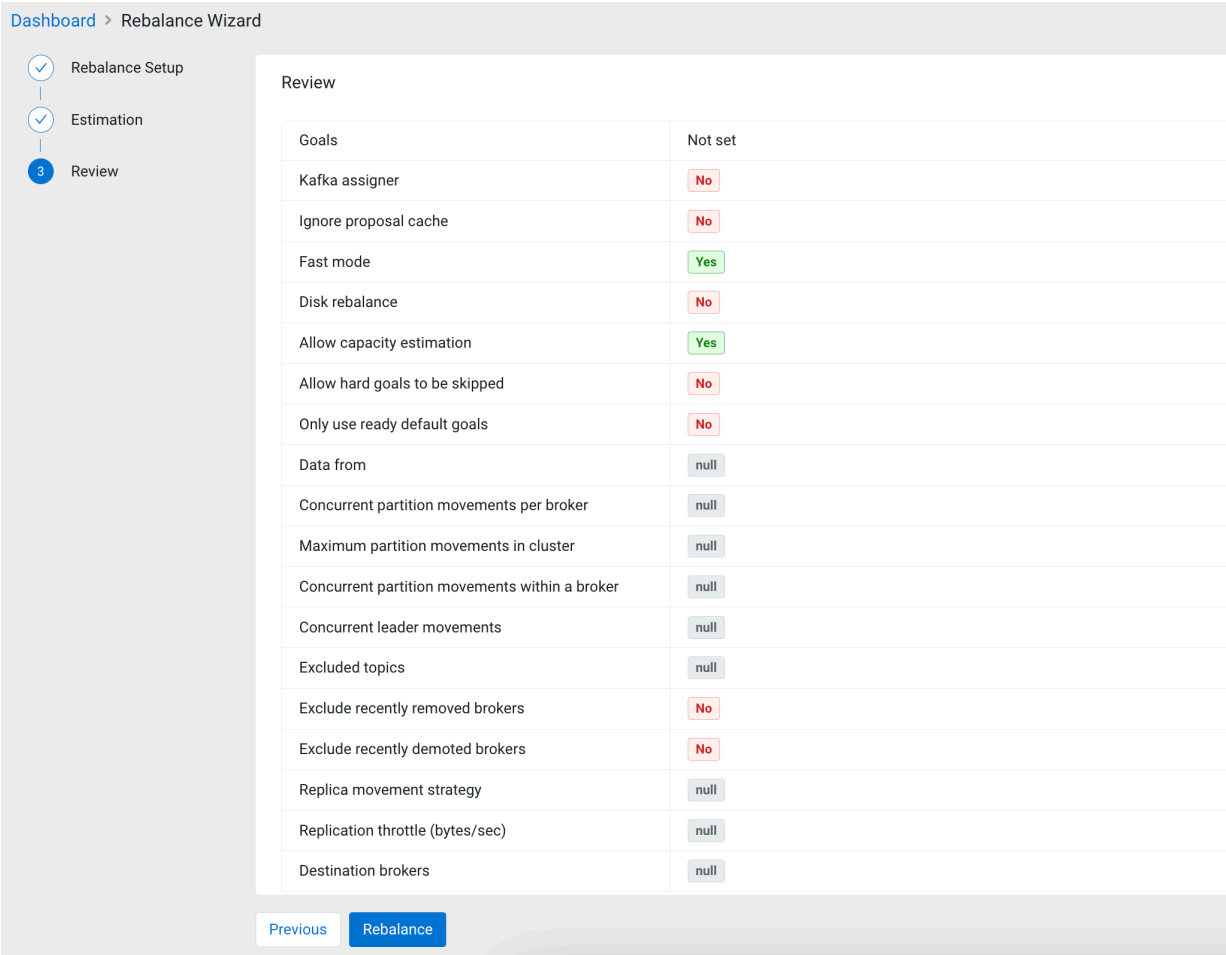

6. Click Rebalance to start the rebalancing process if the configurations are correct.
7. Click Rebalance again to confirm the rebalancing process.

Results

The rebalancing process is triggered based on the configurations provided. You are redirected to the **Cruise Control Dashboard**. The execution of the rebalancing process can be stopped anytime using the Stop button on the **Execution** tile, if you have Admin level permissions.

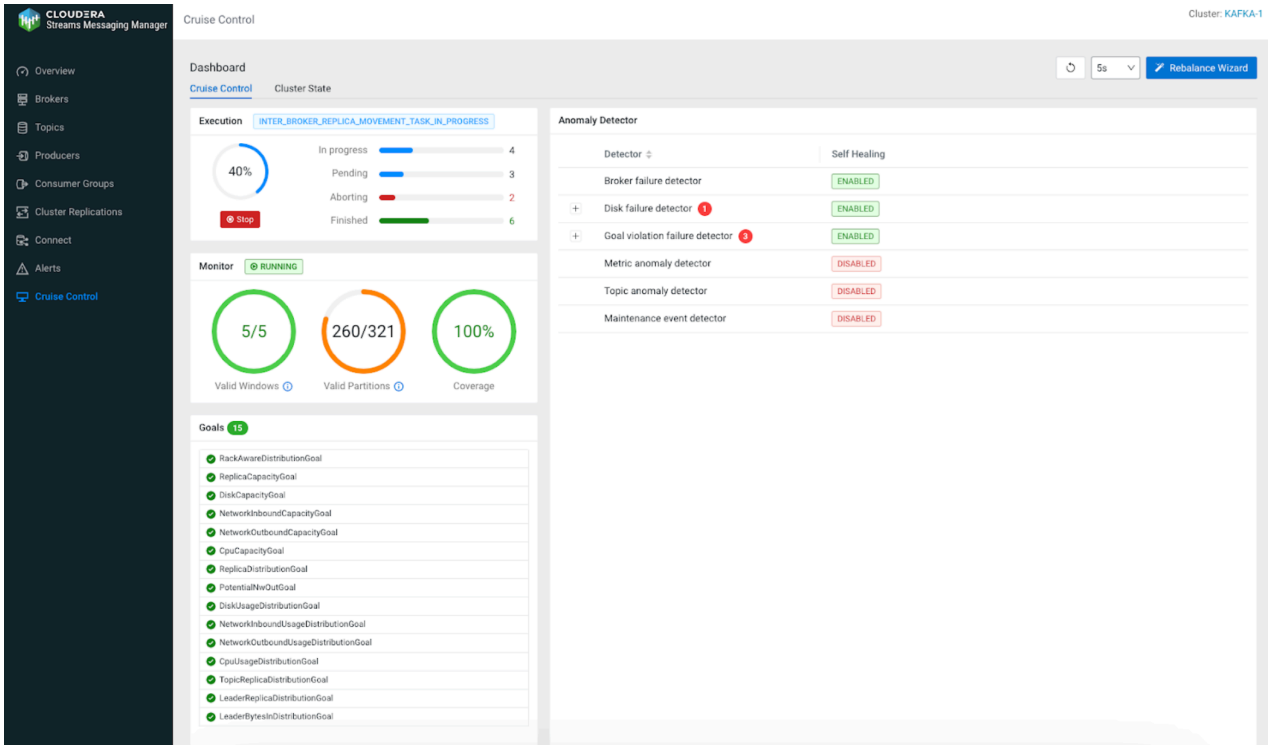
CLOUDERA
Streams Messaging Manager
Cruise Control
Cluster: KAFKA-1
Overview
Brokers
Topics
Producers
Consumer Groups
Cluster Replications
Connect
Alerts
Cruise Control
Dashboard
Cruise Control
Cluster State
5s
Rebalance Wizard
Execution
INTER_BROKER_REPLICA_MOVEMENT_TASK_IN_PROGRESS
40%
Stop
In progress 4
Pending 3
Aborting 2
Finished 6
Monitor
RUNNING
5/5
Valid Windows
260/321
Valid Partitions
100%
Coverage
Goals 15
RackAwareDistributionGoal
ReplicaCapacityGoal
DiskCapacityGoal
NetworkInboundCapacityGoal
NetworkOutboundCapacityGoal
CpuCapacityGoal
ReplicaDistributionGoal
PotentialNwOutGoal
DiskUsageDistributionGoal
NetworkInboundUsageDistributionGoal
NetworkOutboundUsageDistributionGoal
CpuUsageDistributionGoal
TopicReplicaDistributionGoal
LeaderReplicaDistributionGoal
LeaderBytesInDistributionGoal
Anomaly Detector
Detector
Self Healing
Broker failure detector
ENABLED
Disk failure detector 1
ENABLED
Goal violation failure detector 3
ENABLED
Metric anomaly detector
DISABLED
Topic anomaly detector
DISABLED
Maintenance event detector
DISABLED

Cloudera Runtime 7.3.1

Using Apache Iceberg with Spark

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Apache Iceberg with Spark 4
Prerequisites and limitations for using Iceberg in Spark 4
Accessing Iceberg tables 4
Editing a storage handler policy to access Iceberg files on the file system 5
Creating a SQL policy to query an Iceberg table 8
Creating a new Iceberg table from Spark 3 9
Configuring Hive Metastore for Iceberg column changes 10
Importing and migrating Iceberg table in Spark 3 10
Importing and migrating Iceberg table format v2 11
Configuring Catalog 12
Loading data into an unpartitioned table 13
Querying data in an Iceberg table 13
Updating Iceberg table data 13
Iceberg library dependencies for Spark applications 14

Using Apache Iceberg with Spark

Cloudera supports Apache Iceberg which provides a table format for huge analytic datasets. Iceberg enables you to work with large tables, especially on object stores, and supports concurrent reads and writes on all storage media. You can use Spark 3 to interact with Apache Iceberg tables.

Prerequisites and limitations for using Iceberg in Spark

To use Apache Iceberg with Spark, you must be aware of the following limitations:

Limitations

- Iceberg tables with equality deletes do not support partition evolution or schema evolution on Primary Key columns.

  Users should not do partition evolution on tables with Primary Keys or Identifier Fields available, or do Schema Evolution on Primary Key columns, Partition Columns, or Identifier Fields from Spark.
- The use of Iceberg tables as Structured Streaming sources or sinks is not supported.
- PyIceberg is not supported. Using Spark SQL to query Iceberg tables in PySpark is supported.

Iceberg table format version 2

Iceberg table format version 1 and 2 (v1 and v2) is available. Iceberg table format v2 uses row-level UPDATE and DELETE operations that add deleted files to encoded rows that were deleted from existing data files. The DELETE, UPDATE, and MERGE operations function by writing delete files instead of rewriting the affected data files. Additionally, upon reading the data, the encoded deletes are applied to the affected rows that are read. This functionality is called merge-on-read.

With Iceberg table format version 1 (v1), the above-mentioned operations are only supported with copy-on-write where data files are rewritten in their entirety when rows in the files are deleted. Merge-on-read is more efficient for writes, while copy-on-write is more efficient for reads.

Note: Unless otherwise indicated, the operations in the subsequent documentation apply to both v1 and v2 formats.

Accessing Iceberg tables

Cloudera uses Apache Ranger to provide centralized security administration and management. The Ranger Admin UI is the central interface for security administration. You can use Ranger to create two policies that allow users to query Iceberg tables.

How you open the Ranger Admin UI differs from one Cloudera service to another. In Cloudera Management Console, you can select your environment, and then click Environment Details Quick Links Ranger .

You log into the Ranger Admin UI, and the Ranger Service Manager appears.

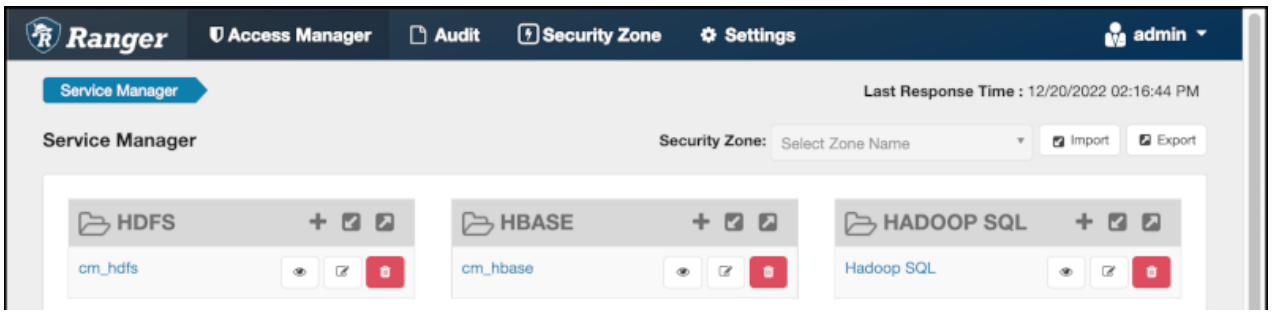

Policies for accessing tables on HDFS

The default policies that appear differ from service to service. You need to set up two Hadoop SQL policies to query Iceberg tables:

- One to authorize users to access the Iceberg files

  Follow steps in "Editing a policy to access Iceberg files" below.
- One to authorize users to query Iceberg tables

  Follow steps in "Creating a policy to query an Iceberg table" below.

Prerequisites

- Obtain the RangerAdmin role.
- Get the user name and password your Administrator set up for logging into the Ranger Admin.

  The default credentials for logging into the Ranger Admin Web UI are admin/admin123.

Editing a storage handler policy to access Iceberg files on the file system

You learn how to edit the existing default Hadoop SQL Storage Handler policy to access files. This policy is one of the two Ranger policies required to use Iceberg.

About this task

The Hadoop SQL Storage Handler policy allows references to Iceberg table storage location, which is required for creating or altering a table. You use a storage handler when you create a file stored as Iceberg on the file system or object store.

In this task, you specify Iceberg as the storage-type and allow the broadest access by setting the URL to *.

The Hadoop SQL Storage Handler policy supports only the RW Storage permission. A user having the required RW Storage permission on a resource, such as Iceberg, that you specify in the storage-type properties, is allowed only to reference the table location (for create/alter operations) in Iceberg. The RW Storage permission does not provide access to any table data. You need to create the Hadoop SQL policy described in the next topic in addition to this Hadoop SQL Storage Handler policy to access data in tables.

For more information about these policy settings, see Ranger Storage Handler documentation.

Procedure

1. Log into Ranger Admin Web UI.
   The Ranger Service Manager appears:

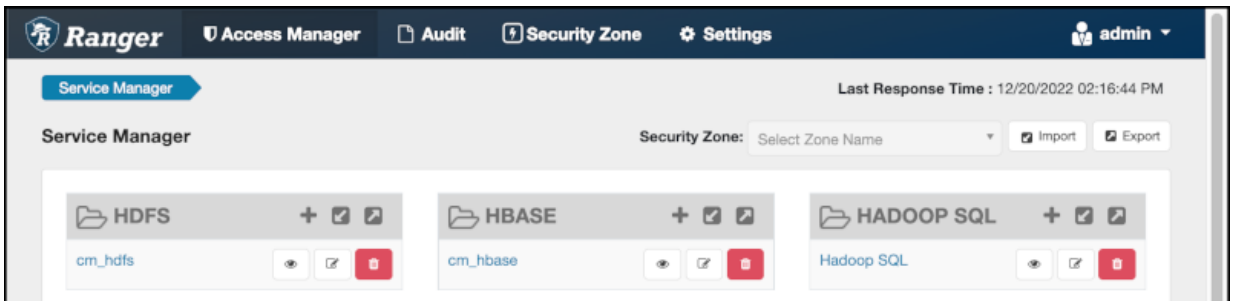

2. In Policy Name, enable the all - storage-type, storage-url policy.

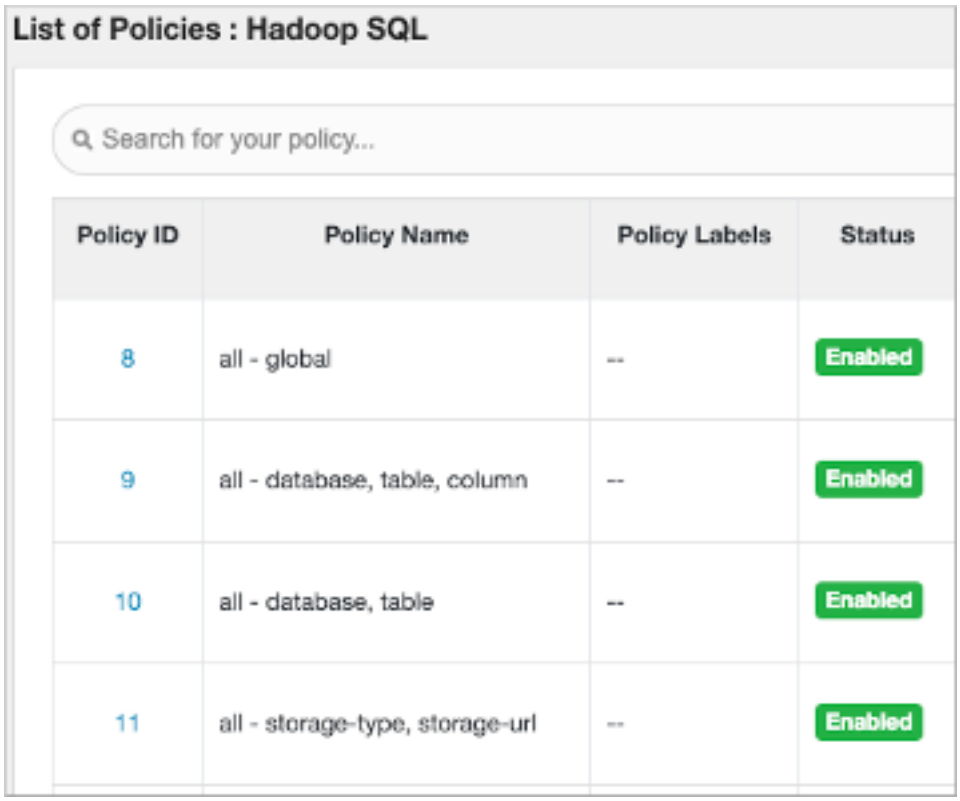

3. In **Service Manager**, in Hadoop SQL, select Edit and edit the all storage-type, storage-url policy.
4. Below Policy Label, select storage-type, and enter iceberg..

5. In Storage URL, enter the value *, enable Include.

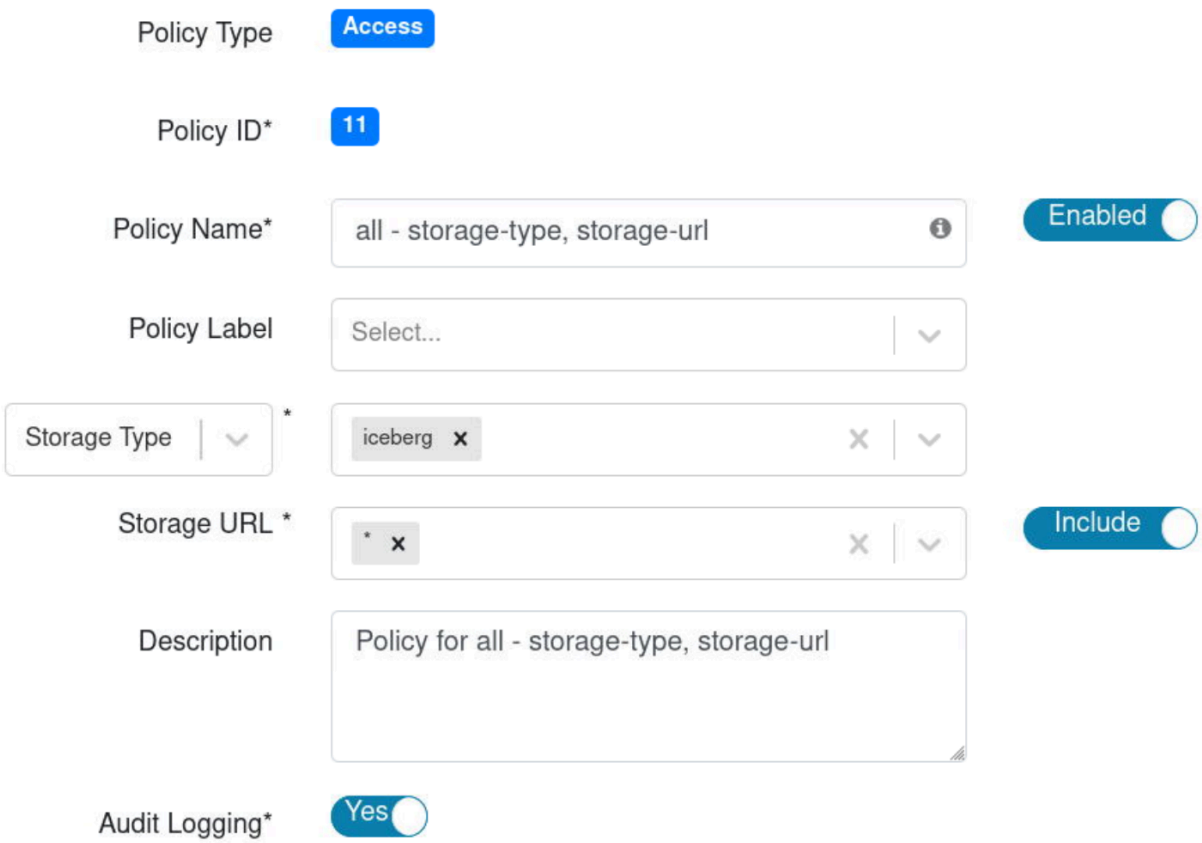

For more information about these policy settings, see Ranger storage handler documentation.

6. In Allow Conditions, specify roles, users, or groups to whom you want to grant RW storage permissions.

   You can specify PUBLIC to grant access to Iceberg tables permissions to all users. Alternatively, you can grant access to one user. For example, add the systest user to the list of users who can access Iceberg:

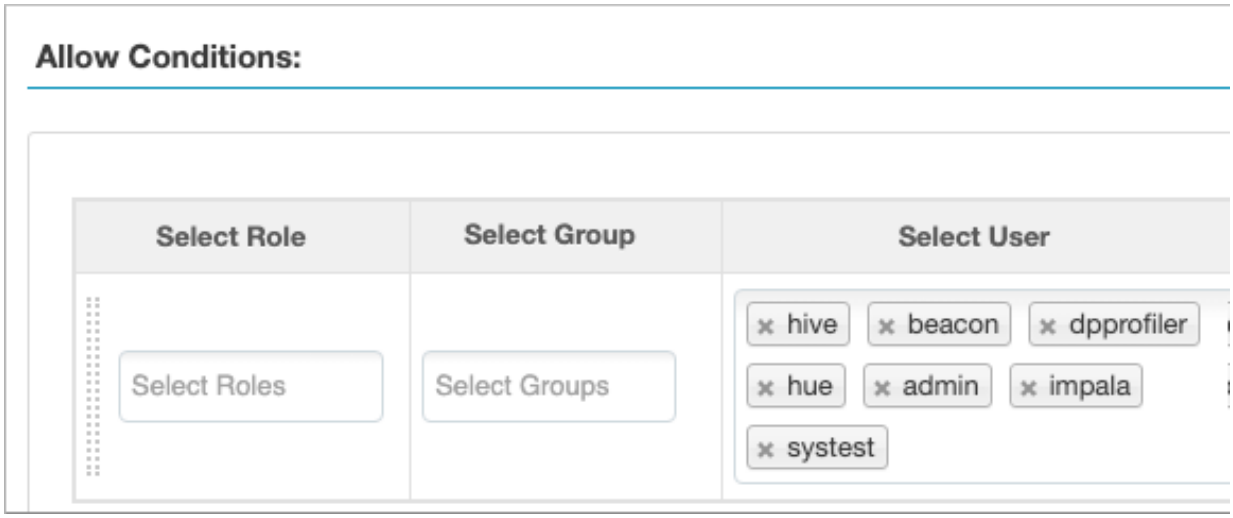

For more information about granting permissions, see Configure a resource-based policy: Hadoop-SQL.

7. Add the RW Storage permission to the policy.
8. Save your changes.

Creating a SQL policy to query an Iceberg table

You learn how to set up the second required policy for using Iceberg. This policy manages SQL query access to Iceberg tables.

About this task

You create a Hadoop SQL policy to allow roles, groups, or users to query an Iceberg table in a database. In this task, you see an example of just one of many ways to configure the policy conditions. You grant (allow) the selected roles, groups, or users the following add or edit permissions on the table: Select, Update, Create, Drop, Alter, and All. You can also deny permissions.

For more information about creating this policy, see Ranger documentation.

Procedure

1. Log into Ranger Admin Web UI.
   The Ranger Service Manager appears.
2. Click Add New Policy.
3. Fill in required fields.
   For example, enter the following required settings:
   - In Policy Name, enter the name of the policy, for example IcebergPolicy1.
   - In database, enter the name of the database controlled by this policy, for example icedb.
   - In table, enter the name of the table controlled by this policy, for example icetable.
   - In columns, enter the name of the column controlled by this policy, for example enter the wildcard asterisk (*) to allow access to all columns of icetable.
   - Accept defaults for other settings.

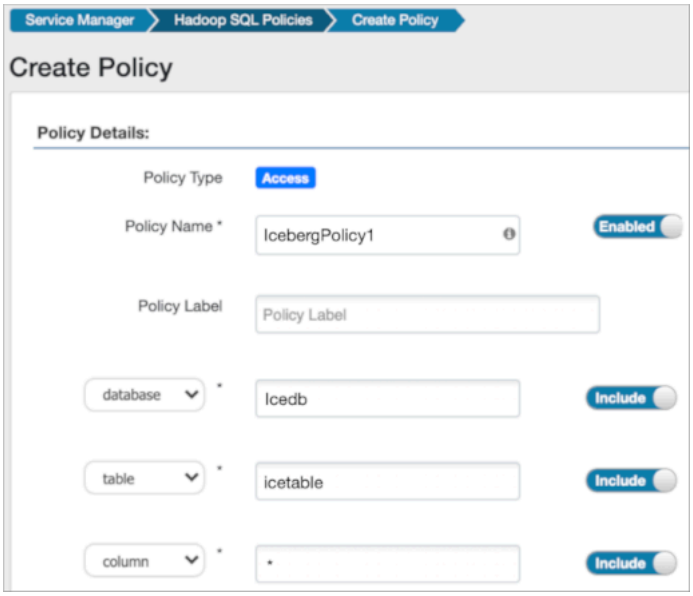

4. Scroll down to Allow Conditions, and select the roles, groups, or users you want to access the table.
   You can use Deny All Other Accesses to deny access to all other roles, groups, or users other than those specified in the allow conditions for the policy.

5. Select permissions to grant.
   For example, select Create, Select, and Alter. Alternatively, to provide the broadest permissions, select All.

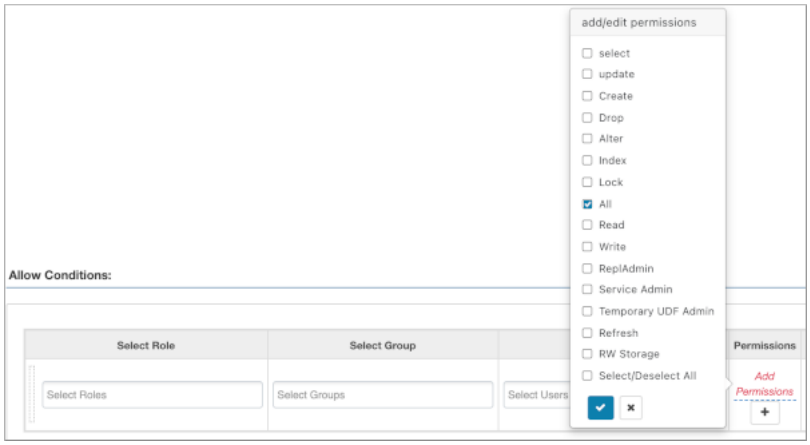

   Ignore RW Storage and other permissions not named after SQL queries. These are for future implementations.
6. Click Add.

Creating a new Iceberg table from Spark 3

You can create an Iceberg table using Spark SQL.

Note: By default, Iceberg tables are created in the v1 format.

An example Spark SQL creation command to create a new Iceberg table is as follows:

```
spark.sql("""CREATE EXTERNAL TABLE ice_t (idx int, name string, state string
)
USING iceberg
PARTITIONED BY (state)""")
```

For information about creating tables, see the Iceberg documentation.

Creating an Iceberg table format v2

To use the Iceberg table format v2, set the format-version property to 2 as shown below:

```
CREATE TABLE logs (app string, lvl string, message string, event_ts timestam
p) USING iceberg TBLPROPERTIES ('format-version' = '2')
```

<delete-mode> <update-mode> and <merge-mode> can be specified during table creation for modes of the respective operation. If unspecified, they default to merge-on-read.

Unsupported Feature: CREATE TABLE ... LIKE

The CREATE TABLE ... LIKE feature is not supported in Spark:

```
CREATE TABLE <target> LIKE <source> USING iceberg
```

Here, `<source>` is an existing Iceberg table. This operation may appear to succeed and does not display errors and only warnings, but the resulting table is not a usable table.

Configuring Hive Metastore for Iceberg column changes

To make schema changes to an existing column of an Iceberg table, you must configure the Hive Metastore of the Data Lake.

Procedure

1. In Cloudera Manager, select the service for the Hive Metastore.
2. Click the Configuration tab.
3. Search for safety valve and find the Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml safety valve.
4. Add the following property:
   - Name: hive.metastore.disallow.incompatible.col.type.changes
   - Value: false
5. Click Save Changes.
6. Restart the service to apply the configuration change.

Importing and migrating Iceberg table in Spark 3

Importing or migrating tables are supported only on existing external Hive tables. When you import a table to Iceberg, the source and destination remain intact and independent. When you migrate a table, the existing Hive table is converted into an Iceberg table. You can use Spark SQL to import or migrate a Hive table to Iceberg.

Importing

Call the snapshot procedure to import a Hive table into Iceberg using a Spark 3 application.

```
spark.sql("CALL
<catalog>.system.snapshot('<src>', '<dest>')")
```

Definitions:

- <src> is the qualified name of the Hive table
- <dest> is the qualified name of the Iceberg table to be created
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.

For example:

```
spark.sql("CALL
spark_catalog.system.snapshot('hive_db.hive_tbl',
'iceberg_db.iceberg_tbl')")
```

For information on compiling Spark 3 application with Iceberg libraries, see Iceberg library dependencies for Spark applications linked below.

Migrating

When you migrate a Hive table to Iceberg, a backup of the table, named <table_name>_backup_, is created.

Ensure that the TRANSLATED_TO_EXTERNAL property, that is located in TBLPROPERTIES, is set to false before migrating the table. This ensures that a table backup is created by renaming the table in Hive metastore (HMS) instead of moving the physical location of the table. Moving the physical location of the table would entail copying files in Amazon s3.

We recommend that you refrain from dropping the backup table, as doing so will invalidate the newly migrated table.

If you want to delete the backup table, set the following:

```
'external.table.purge'='FALSE'
```

Note: For Cloudera Data Engineering Private Cloud 1.5.2 and above, the property will be set automatically.

Deleting the backup table in the manner above will prevent underlying data from being deleted, therefore, only the table will be deleted from the metastore.

To undo the migration, drop the migrated table and restore the Hive table from the backup table by renaming it.

Call the migrate procedure to migrate a Hive table to Iceberg.

```
spark.sql("CALL
<catalog>.system.migrate('<src>')")
```

Definitions:

- <src> is the qualified name of the Hive table
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.

For example:

```
spark.sql(“CALL
spark_catalog.system.migrate(‘hive_db.hive_tbl’)”)
```

Importing and migrating Iceberg table format v2

Importing or migrating Hive tables Iceberg table formats v2 are supported only on existing external Hive tables. When you import a table to Iceberg, the source and destination remain intact and independent. When you migrate a table, the existing Hive table is converted into an Iceberg table. You can use Spark SQL to import or migrate a Hive table to Iceberg.

Importing

Call the snapshot procedure to import a Hive table into Iceberg table format v2 using a Spark 3 application.

```
spark.sql("CALL
<catalog>.system.snapshot(source_table => '<src>',
table => '<dest>',
properties => map('format-version', '2', 'write.delete.mode', '<delete-mo
de>',
'write.update.mode', '<update-mode>',
'write.merge.mode', '<merge-mode>'))")
```

Definitions:

- <src> is the qualified name of the Hive table
- <dest> is the qualified name of the Iceberg table to be created
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.
- <delete-mode> <update-mode> and <merge-mode> are the modes that shall be used to perform the respective operation. If unspecified, they default to 'merge-on-read'

For example:

```
spark.sql("CALL
```

```
spark_catalog.system.snapshot('hive_db.hive_tbl',
'iceberg_db.iceberg_tbl')")
```

For information on compiling Spark 3 application with Iceberg libraries, see Iceberg library dependencies for Spark applications linked below.

Migrating

Call the migrate procedure to migrate a Hive table to Iceberg.

```
spark.sql("CALL
<catalog>.system.migrate('<src>',
map('format-version', '2',
'write.delete.mode', '<delete-mode>',
'write.update.mode', '<update-mode>',
'write.merge.mode', '<merge-mode>'))")
```

Definitions:

- <src> is the qualified name of the Hive table
- <catalog> is the name of the catalog, which you pass in a configuration file. For more information, see Configuring Catalog linked below.
- <delete-mode> <update-mode> and <merge-mode> are the modes that shall be used to perform the respective operation. If unspecified, they default to 'merge-on-read'

For example:

```
spark.sql("CALL
spark_catalog.system.migrate('hive_db.hive_tbl',
map('format-version', '2',
'write.delete.mode', 'merge-on-read',
'write.update.mode', 'merge-on-read',
'write.merge.mode', 'merge-on-read'))")
```

Upgrading Iceberg table format v1 to v2

To upgrade an Iceberg table format from v1 to v2, run an `ALTER TABLE` command as follows:

```
spark.sql("ALTER TABLE <table_name> SET TBLPROPERTIES('merge-on-read', '
2')")
```

<delete-mode>,<update-mode>, and <merge-mode> can be specified as the modes that shall be used to perform the respective operation. If unspecified, they default to ‘merge-on-read'

Configuring Catalog

When using Spark SQL to query an Iceberg table from Spark, you refer to a table using the following dot notation:

```
<catalog_name>.<database_name>.<table_name>
```

The default catalog used by Spark is named spark_catalog. When referring to a table in a database known to spark_catalog, you can omit <catalog_name>. .

Iceberg provides a SparkCatalog property that understands Iceberg tables, and a SparkSessionCatalog property that understands both Iceberg and non-Iceberg tables. The following are configured by default:

```
spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
spark.sql.catalog.spark_catalog.type=hive
```

This replaces Spark's default catalog by Iceberg's SparkSessionCatalog and allows you to use both Iceberg and non-Iceberg tables out of the box.

There is one caveat when using SparkSessionCatalog. Iceberg supports CREATE TABLE … AS SELECT (CTAS) and REPLACE TABLE … AS SELECT (RTAS) as atomic operations when using SparkCatalog. Whereas, the CTAS and RTAS are supported but are not atomic when using SparkSessionCatalog. As a workaround, you can configure another catalog that uses SparkCatalog. For example, to create the catalog named iceberg_catalog, set the following:

```
spark.sql.catalog.iceberg_catalog=org.apache.iceberg.spark.SparkCatalog
spark.sql.catalog.iceberg_catalog.type=hive
```

You can configure more than one catalog in the same Spark job. For more information, see the *Iceberg documentation*.

Related Tasks

Iceberg documentation

Loading data into an unpartitioned table

You can insert data into an unpartitioned table. The syntax to load data into an iceberg table:

```
INSERT INTO table_identifier [ ( column_list ) ]
    VALUES ( { value | NULL } [ , ... ] ) [ , ( ... ) ]
```

Or

```
INSERT INTO table_identifier [ ( column_list ) ]
    query
```

Example:

```
INSERT INTO students VALUES
    ('Amy Smith', '123 Park Ave, San Jose', 111111)
INSERT INTO students VALUES
    ('Bob Brown', '456 Taylor St, Cupertino', 222222),
    ('Cathy Johnson', '789 Race Ave, Palo Alto', 333333)
```

Querying data in an Iceberg table

To read the Iceberg table, you can use SparkSQL to query the Iceberg tables.

Example:

```
spark.sql("select * from ice_t").show(1000, false)
```

Important: When querying Iceberg tables in HDFS, CDS disables locality by default. Because enabling locality generally leads to increased Spark planning time for queries on such tables and often the increase is quite significant. If you wish to enable locality, set the spark.cloudera.iceberg.locality.enabled to true . For example, you can do it by passing --conf spark.cloudera.iceberg.locality.enabled=true to your spark3-submit command.

Updating Iceberg table data

Iceberg table data can be updated using copy-on-write or merge-on-read. The table version you are using will determine how you can update the table data.

v1 format

Iceberg supports bulk updates through MERGE, by defaulting to copy-on-write deletes when using v1 table format.

v2 format

Iceberg table format v2 supports efficient row-level updates and delete operations leveraging merge-on-read.

For more details, refer to *Position Delete Files* linked below.

For updating data examples, see *Spark Writes* linked below.

Iceberg library dependencies for Spark applications

If your Spark application only uses Spark SQL to create, read, or write Iceberg tables, and does not use any Iceberg APIs, you do not need to build it against any Iceberg dependencies. The runtime dependencies needed for Spark to use Iceberg are in the Spark classpath by default. If your code uses Iceberg APIs, then you need to build it against Iceberg dependencies.

Cloudera publishes Iceberg artifacts to a Maven repository with versions matching the Iceberg in CDS.

Note: Use 1.3.0.7.1.9.0-387 iceberg version for compilation. The below iceberg dependencies should only be used for compilation. Including iceberg jars within a Spark application fat jar must be avoided.

```
<dependency>
     <groupId>org.apache.iceberg</groupId>
     <artifactId>iceberg-core</artifactId>
     <version>${iceberg.version}</version>
     <scope>provided</scope>
</dependency>
<!-- for org.apache.iceberg.hive.HiveCatalog -->
<dependency>
     <groupId>org.apache.iceberg</groupId>
     <artifactId>iceberg-hive-metastore</artifactId>
     <version>${iceberg.version}</version>
     <scope>provided</scope>
</dependency>
<!-- for org.apache.iceberg.spark.* classes if used -->
<dependency>
    <groupId>org.apache.iceberg</groupId>
    <artifactId>iceberg-spark</artifactId>
    <version>${iceberg.version}</version>
    <scope>provided</scope>
</dependency>
```

Alternatively, the following dependency can be used:

```
<dependency>
   <groupId>org.apache.iceberg</groupId>
   <artifactId>iceberg-spark-runtime-3.3_2.12</artifactId>
   <version>${iceberg.version}</version>
   <scope>provided</scope>
</dependency>
```

The iceberg-spark3-runtime JAR contains the necessary Iceberg classes for Spark runtime support, and includes the classes from the dependencies above.

Cloudera Runtime 7.3.1

Apache Spark Overview

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Spark Overview **4**

Unsupported Apache Spark Features **4**

Apache Spark Overview

Apache Spark is a distributed, in-memory data processing engine designed for large-scale data processing and analytics.

Apache Spark is a general framework for distributed computing that offers high performance for both batch and interactive processing. It exposes APIs for Java, Python, and Scala and consists of Spark core and several related projects.

You can run Spark applications locally or distributed across a cluster, either by using an interactive shell or by submitting an application. Running Spark applications interactively is commonly performed during the data-exploration phase and for ad hoc analysis.

To run applications distributed across a cluster, Spark requires a cluster manager. Cloudera Data Platform (CDP) supports only the YARN cluster manager. When run on YARN, Spark application processes are managed by the YARN ResourceManager and NodeManager roles. Spark Standalone is not supported.

For detailed API information, see the Apache Spark project site.

Note: Although this document makes some references to the external Spark site, not all the features, components, recommendations, and so on are applicable to Spark when used on CDH. Always cross-check the Cloudera documentation before building a reliance on some aspect of Spark that might not be supported or recommended by Cloudera.

CDP supports Apache Spark, Apache Livy for local and remote access to Spark through the Livy REST API, and Apache Zeppelin for browser-based notebook access to Spark. The Spark LLAP connector is not supported.

Unsupported Apache Spark Features

The following Apache Spark features are not supported in Cloudera Data Platform.

- Apache Spark experimental features/APIs are not supported unless stated otherwise.
- Spark Streaming (DStreams) reading from Kafka topics containing transactions such as idempotent producer being used to publish records.
- Using the JDBC Datasource API to access Hive or Impala.
- Spark with Kudu is not supported for ADLS data.
- IPython / Jupyter notebooks is not supported. The IPython notebook system (renamed to Jupyter as of IPython 4.0) is not supported.
- Certain Spark Streaming features, such as the mapWithState method, are not supported.
- Thrift JDBC/ODBC server (also known as Spark Thrift Server or STS)
- Spark SQL CLI
- GraphX
- SparkR
- GraphFrames
- Structured Streaming is supported, but the following features of it are not:
  - Continuous processing, which is still experimental, is not supported.
  - Stream static joins with HBase have not been tested and therefore are not supported.
- Hudi
- Push-based shuffle
- ZSTD compression in ORC data source (SPARK-33978)
- spark.hadoopRDD.ignoreEmptySplits (SPARK-34809)
- LDAP authentication for livy-server (LIVY-356)
- Thrift ldap authentication, based on ldapurl, basedn, domain (LIVY-678)

Cloudera Runtime 7.3.1

Apache Spark integration with Schema Registry

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Spark 3 integration with Schema Registry..... 4
Configuration.....5
Fetching Spark schema by name..... 5
Building and deploying your app.....6
Running in a Kerberos-enabled cluster.....7
Unsupported features..... 8

Apache Spark 3 integration with Schema Registry

Apache Spark 3 integrated with Schema Registry provides a library to leverage Schema Registry for managing Spark schemas and to serialize and/or de-serialize messages in Spark data sources and sinks.

Note: The Apache Spark 3 integration with Schema Registry is available in CDS 3.3 Powered by Apache Spark from version 3.3.2 for 7.1.9 SP1.

Running the example programs

The examples illustrate the API usage and how to integrate with Schema Registry. The examples can be run from IDE (for example, Intellij) by specifying a master URL or by using spark3-submit.

```
spark3-submit --jars /opt/cloudera/parcels/CDH/lib/spark-schema-registry-for
-spark3/spark-schema-registry-for-spark3_2.12-jar-with-dependencies.jar \
--class com.hortonworks.spark.registry.examples.classname \
/opt/cloudera/parcels/CDH/lib/spark-schema-registry-for-spark3/examples/sp
ark-schema-registry-for-spark3-examples_2.12.jar [***SCHEMA-REGISTRY-URL***]
 \
bootstrap-servers input-topic output-topic checkpoint-location
```

Using the APIs

Typically in a Spark application you define the Spark schema for the data you are going to process:

```
// the schema for truck events
val schema = StructType(Seq(
  StructField("driverId", IntegerType, nullable = false),
  StructField("truckId", IntegerType, nullable = false),
  StructField("miles", LongType, nullable = false),
  StructField("eventType", StringType, nullable = false),
  ...
  ...
)

// read Json string messages from the data source
val messages = spark
      .readStream
      .format(...)
      .option(...)
      .load()

// parse the messages using the above schema and do further operations
val df = messages
      .select(from_json($"value".cast("string"), schema).alias("value"))
      ...
// project (driverId, truckId, miles) for the events where miles > 300
val filtered = df.select($"value.driverId", $"value.truckId", $"value.miles"
)
      .where("value.miles > 300")
```

However, this approach is not practical because the schema information is tightly coupled with the code. The code needs to be changed when the schema changes, and there is no ability to share or reuse the schema between the message producers and the applications that consume the messages.

Using Schema Registry is a better solution because it enables you to manage different versions of the schema and define compatibility policies.

Configuration

The Schema Registry integration comes as a utility method which can be imported into the scope.

```
import com.hortonworks.spark.registry.util._
```

Before invoking the APIs, you need to define an implicit SchemaRegistryConfig which will be passed to the APIs. The main configuration parameter is the schema registry URL.

```
// the schema registry client config
val config = Map[String, Object]("[***SCHEMA.REGISTRY.URL***]" -> schemaR
egistryUrl)
// the schema registry config that will be implicitly passed
implicit val srConfig:SchemaRegistryConfig = SchemaRegistryConfig(config)
```

SSL configuration

SchemaRegistryConfig expects the following SSL configuration properties:

```
"schema.registry.client.ssl.protocol" -> "SSL",
"schema.registry.client.ssl.trustStoreType" -> "JKS",
"schema.registry.client.ssl.trustStorePath" -> "/var/lib/cloudera-scm-agent/
agent-cert/cm-auto-global_truststore.jks",
"schema.registry.client.ssl.trustStorePassword" ->
 "[***CHANGEMECLIENTPWD***]"
```

Fetching Spark schema by name

The API supports fetching the Schema Registry schema as a Spark schema.

- sparkSchema(schemaName: String)

  Returns the spark schema corresponding to the latest version of schema defined in the Schema Registry.
- sparkSchema(schemaName: String, version: Int)

  Returns the spark schema corresponding to the given version of schema defined in the Schema Registry.

Using the Schema Registry integration, the example previously shown can be simplified, as there is no need to explicitly specify the Spark schema in the code:

```
// retrieve the translated "Spark schema" by specifying the schema registry
schema name
val schema = sparkSchema(name)

// parse the messages using the above schema and do further operations
val df = messages
         .select(from_json($"value".cast("string"), schema).alias("value"))
         ...

// project (driverId, truckId, miles) for the events where miles > 300
val filtered = df.select($"value.driverId", $"value.truckId", $"value.miles
")
```

```
                    .where("value.miles > 300")
```

Serializing messages using Schema Registry

The following method can be used to serialize the messages from Spark to Schema Registry binary format using schema registry serializers.

- to_sr(data: Column, schemaName: String, topLevelRecordName: String, namespace: String)

  Converts a Spark column data to binary format of Schema Registry. This looks up a Schema Registry schema for the schemaName that matches the input and automatically registers a new schema, if not found. The topoLevelRecordName and namespace are optional and will be mapped to Avro top level record name and record namespace.

De-serializing messages using Schema Registry

The following methods can be used to de-serialize Schema Registry serialized messages into Spark columns.

- from_sr(data: Column, schemaName: String)

  Converts Schema Registry binary format to Spark column, using the latest version of the schema.
- from_sr(data: Column, schemaName: String, version: Int)

  Converts Schema Registry binary format to Spark column using the given Schema Registry schema name and version.

Serialization - deserialization example

The following is an example that uses the from_sr to de-serialize Schema Registry formatted messages into Spark, transforms and serializes it back to Schema Registry format using to_sr, and writes to a data sink.

This example assumes Spark Structured Streaming use cases, but it should work well for the non-streaming use cases as well (read and write).

```
// Read schema registry formatted messages and deserialize to spark columns.
val df = messages
      .select(from_sr($"value", topic).alias("message"))
// project (driverId, truckId, miles) for the events where miles > 300
val filtered = df.select($"message.driverId", $"message.truckId", $"message.
miles")
      .where("message.miles > 300")
// write the output as schema registry serialized bytes to a sink
// should produce events like {"driverId":14,"truckId":25,"miles":373}
val query = filtered
      .select(to_sr(struct($"*"), outSchemaName).alias("value"))
      .writeStream
      .format(..)
      .start()
```

The output schema outSchemaName is automatically published to the Schema Registry if it does not exist.

Building and deploying your app

Add a Maven dependency in your project to make use of the library and build your application JAR file:

```
<dependency>
    <groupId>com.hortonworks</groupId>
    <artifactId>spark-schema-registry-for-spark3_2.12</artifactId>
    <version>version</version>
```

```
 </dependency>
```

Once the application JAR file is built, deploy it by adding the dependency in spark3-submit using --packages:

```
spark3-submit --packages com.hortonworks:spark-schema-registry-for-spark3_2.
12:version \
--conf spark.jars.repositories=[***HTTPS://REPOSITORY.EXAMPLE.COM***] \
--class YourApp \
your-application-jar \
args ...
```

Make sure the package is published in a local or online available repository.

If the package is not published to an available repository, or your Spark application cannot access external networks, you can use an uber JAR file instead:

```
spark3-submit --master [***MASTER URL****] \
--jars /opt/cloudera/parcels/SPARK3/lib/spark3/spark-schema-registry-for-s
park3/spark-schema-registry-for-spark3_2.12-jar-with-dependencies.jar \
--class YourApp \
your-application-jar \
args ...
```

Running in a Kerberos-enabled cluster

The library works in a Kerberos setup, where Spark and Schema Registry has been deployed on a Kerberos-enabled cluster.

To configure, set up the appropriate JAAS config for RegistryClient (and KafkaClient, if the Spark data source or sink is Kafka).

As an example, to run the SchemaRegistryAvroExample in a Kerberos setup, follow these steps:

1. Create a keytab (for example, app.keytab) with the login user and principal you want to run the application.
2. Create an app_jaas.conf file and specify the keytab and principal created in Step 1.

   If deploying to YARN, the keytab and conf files will be distributed as YARN local resources. They will be placed in the current directory of the Spark YARN container, and the location needs to be specified as ./app.keytab.

   ```
   RegistryClient {
       com.sun.security.auth.module.Krb5LoginModule required
       useKeyTab=true
       keyTab="./app.keytab"
       storeKey=true
       useTicketCache=false
       principal="[***PRINCIPAL***]";
   };

   KafkaClient {
      com.sun.security.auth.module.Krb5LoginModule required
      useKeyTab=true
      keyTab="./app.keytab"
      storeKey=true
      useTicketCache=false
      serviceName="kafka"
      principal="[***PRINCIPAL***]";
   };
   ```

3. Provide the required ACLs for the kafka topics (in-topic, out-topic) for the principal.
4. Use spark3-submit to pass the JAAS configuration file with extraJavaOptions. (And also as local resource files in YARN cluster mode.)

```
spark3-submit --master yarn --deploy-mode cluster \
    --keytab app.keytab --principal [***PRINCIPAL***] \
    --files app_jaas.conf#app_jaas.conf,app.keytab#app.keytab \
    --jars /opt/cloudera/parcels/SPARK3/lib/spark3/spark-schema-registry-f
or-spark3/spark-schema-registry-for-spark3_2.12-jar-with-dependencies.jar
 \
    --conf "spark.executor.extraJavaOptions=-Djava.security.auth.login.c
onfig=./app_jaas.conf" \
    --conf "spark.driver.extraJavaOptions=-Djava.security.auth.login.co
nfig=./app_jaas.conf" \
    --class com.hortonworks.spark.registry.examples.SchemaRegistryAvroEx
ample \
/opt/cloudera/parcels/SPARK3/lib/spark3/spark-schema-registry-for-spark3/
examples/spark-schema-registry-for-spark3-examples_2.12.jar \
    [***SCHEMA-REGISTRY-URL***] bootstrap-server in-topic out-topic che
ckpoint-dir SASL_PLAINTEXT
```

Unsupported features

Apache Spark 3 integration with Schema Registry is not supported in pyspark.

Cloudera Runtime 7.3.1

Upgrading Apache Spark

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Upgrading Spark 2 to Spark 3 for Cloudera Public Cloud 7.3.1.....4
Upgrading from 7.2.18.....5
Upgrading Apache Spark 2.4.8 on 7.2.18 SP2 to Spark 3 on 7.3.1.....5
Upgrading Apach Spark 2.4.8 (with 3.4.1 bundled) on 7.2.18 SP2 to Spark 3 on 7.3.1.....7
Upgrading Apache Spark 3.4.1 (bundled) on 7.2.18 SP2 to Spark 3 on 7.3.1.....9
Upgrading from 7.2.17.....10
Upgrading Apache Spark 2.4.8 on 7.2.17 to Spark 3 on 7.3.1.....10
Upgrading Apache Spark 2.4.8 on 7.2.17 to Spark 3 on 7.3.1.....12
Upgrading Apache Spark 2.4.8 (with 3.3.2 bundled) on 7.2.17 to Spark 3 on 7.3.1.....12
Upgrading Apache Spark 2.4.8 (with 3.3.2 bundled) on 7.2.17 to Spark 3 on 7.3.1.....15
Upgrading Apache Spark 3.3.2 (bundled) on 7.2.17 to Spark 3 on 7.3.1.....15
Migrating Spark applications.....16
Java versions.....17
Scala versions.....17
Python versions.....17
Spark commands.....18
Spark connectors.....18
Logging.....18
Third-party libraries.....18
Spark behavior changes.....18
Apache Spark Migration guides.....19
Spark 2 to Spark 3 workload refactoring.....19
Unsupported features.....22
Post-migration checklist.....22
Benchmark testing.....23
Troubleshooting.....23

Upgrading Spark 2 to Spark 3 for Cloudera Public Cloud 7.3.1

Upgrading Apache Spark from version 2 to version 3 in Cloudera Cloudera Public Cloud is a process that involves:

1. Intermediate in-place cluster upgrade tasks, due to different support for Spark 3 versions of connectors in Cloudera versions.
2. Intermediate Spark application migration tasks, due to minor or maintenance Spark version changes.
3. Sidecar migration tasks for Data Hub clusters, because adjusting existing Data Hub clusters is not possible, but in-place 7.3.1 upgrade can only happen on clusters where Spark 2 is no longer present.
4. Application migration from Spark 2 to Spark 3, due to major Spark version changes.
5. Post-application migration tasks.
6. In-place cluster upgrade tasks.

   Important: Cluster upgrade to version 7.3.1 is only allowed if all Spark 2 applications have been migrated to Spark 3 and tested in the **source cluster**.

7. Spark application migration tasks, due to minor or maintenance Spark version changes.

Warning:

Upgrading to Cloudera Runtime 7.3.1 is only supported for the following service packs:

- 7.2.17.200
- 7.2.17.300
- 7.2.17.400
- 7.2.17.500
- 7.2.18.0
- 7.2.18.100
- 7.2.18.200

If your Cloudera Runtime version is not in the list of supported versions for a direct upgrade, it is possible that you can still upgrade to one of the above versions first, and then upgrade to Cloudera Runtime 7.3.1.

For more information on upgrading Cloudera Runtime 7.2.17.x and the supported upgrade paths, see Upgrading to Cloudera Runtime 7.3.1

Note: The application migration tasks (see *Migrating Spark Applications*) are done on your application codebase.

Note: Depending on your environment and Spark applications, some of these steps are not necessary.

This table below provides links to the upgrade guides based on the version of Cloudera Public Cloud you're using, and the source versions of Spark in your environment.

Important:

If you're using Spark 2 with any of the following connectors, follow the upgrade steps in the relevant *Upgrade guide (with connectors)* pages:

- Oozie
- Solr
- Phoenix
- Hive Warehouse Connector
- Spark Schema Registry

Each upgrade guide contain all steps needed to upgrade Spark 2 to Spark 3 and perform the upgrade to Cloudera Public Cloud version 7.3.1.

Source cluster version	Source cluster Spark 2 version	Source cluster Spark 3 version	Data Hub template	
7.2.18 SP2	2.4.8	None	Data Hub was created with a custom template.	Upgrade guide
	2.4.8	3.4.1 (bundled)		Upgrade guide
	None	3.4.1 (bundled)	Data Hub was created with the *7.2.18 - Data Engineering: Apache Spark3, Apache Hive, Apache Oozie* or a custom template.	Upgrade guide
7.2.17	2.4.8	None	Data Hub was created with the *7.2.17 - Data Engineering: Apache Spark, Apache Hive, Apache Oozie* or a custom template.	Upgrade guide Upgrade (with connectors) guide
	2.4.8	3.3.2 (bundled)	Data Hub was created with a custom template.	Upgrade guide Upgrade guide (with connectors)
	None	3.3.2 (bundled)	Data Hub was created with the *7.2.17 - Data Engineering: Apache Spark3" or a custom template*.	Upgrade guide

Related Information

Migrating Spark applications

Upgrading from 7.2.18

Upgrading Apache Spark 2.4.8 on 7.2.18 SP2 to Spark 3 on 7.3.1

The following steps will help you upgrading from Apache Spark 2.4.8 on Cloudera Public Cloud 7.2.18 SP2 to Spark 3.4.1 on 7.3.1.

Source cluster version	Source cluster Spark 2 version	Source cluster Spark 3 version	Target cluster version	Target cluster Spark 3 version	Spark 2 used with connectors[1]
7.2.18 SP2	2.4.8	none	**7.3.1**	**3.4.1**	no

Sidecar migration of Data Hub clusters

Procedure

Sidecar migration tasks for Data Hub clusters

The new 7.2.18 Data Hub cluster needs to use Spark 3 and Livy 3 instead of Spark 2 and Livy 2.

Depending on the template you used for your existing Data Hub clusters, a new custom template might be needed that contains Spark 3 instead of Spark 2. Alternatively, the built-in *7.2.18 - Data Engineering: Apache Spark3, Apache Hive, Apache Oozie* template can be used, as it contains Spark 3 only.

1. Check the current services in your template, and add the built-in *7.2.18 - : Apache Spark3, Apache Hive, Apache Oozie* template.
2. If the built-in *7.2.18 - : Apache Spark3, Apache Hive, Apache Oozie* template doesn't work, you can create a custom template. Replace all Spark 2 and Livy 2 references with Spark 3 and Livy 3, respectively.
3. Add a new Spark 3-based 7.2.18 cluster to the environment, using your custom template or the built-in *7.2.18 - : Apache Spark3, Apache Hive, Apache Oozie* template.
4. Migrate all non-spark workloads from the old cluster to the new cluster.

Application migration tasks (Spark 2 to 3)

Procedure

1. Follow the Spark application migration documentation to migrate your Apache Spark Applications from version 2.4.8 to 3.4.1
   a) Check the supported Java versions.
   b) Check the supported Scala version.
   c) Check the supported Python versions.
   d) Account for changed or versioned Spark commands in your code. (spark-submit, pyspark, etc.)
   e) Check supported versions for Spark connectors.
   f) Check the logging library used in your code.
   g) Check the compatibility of 3rd-party libraries used in your code.
   h) Check Spark behavior changes and refactor your code.
2. Migrate all Spark 2 applications in the old cluster to Spark 3 applications in the new cluster.

Post-application migration tasks

Procedure

1. Move Spark 2 event logs to the Spark 3 event logs directory.
2. Drop the old cluster.

In-place cluster upgrade

Before you begin

Important: Cluster upgrade to version 7.3.1 is only allowed if all Spark 2 applications have been migrated to Spark 3 and tested in the **source cluster**.

Procedure

1. Upgrade the Data Lake cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Stop all s attached to the environment.
   c) From the , click Data LakesEnvironment Name, scroll to the bottom of the **Data Lake details** page, and click the Upgrade tab.
   d) Click the Target Version drop-down menu to see any available upgrades.
   e) If you want to skip the automatic backup that is taken before the upgrade, uncheck the Automatic backup box.
   f) Click Validate and Prepare to check for any configuration issues and begin the parcel download and distribution.
   g) Click Upgrade to initiate the upgrade.
   h) Click the Event History tab to monitor the upgrade process and verify that it completes successfully.

   For more information, see Data Lake upgrade.

2. Upgrade the new cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Start the cluster.
   c) Check the current version of .
   d) If your cluster uses Streams Replication Manager, export or migrate aggregated metrics.
   e) If you use autoscaling, disable autoscaling on the cluster.
   f) Upgrade the cluster.
   g) Monitor the upgrade progress using the Event History tab.
   h) When the upgrade is complete, verify the new version.
   i) If you disabled autoscaling on the cluster, you can re-enable it after upgrade.

   For more information, see Upgrading s.

Final steps

Procedure

After the upgrade and application migration are complete:

1. Check the status of your Data Lakes, s, and clusters.
2. Perform benchmark testing on your applications. See Spark Application Migration.

Upgrading Apach Spark 2.4.8 (with 3.4.1 bundled) on 7.2.18 SP2 to Spark 3 on 7.3.1

The following steps will help you upgrading from Apache Spark 2.4.8 (with 3.4.1 bundled) on Cloudera Public Cloud 7.2.18 SP2 to Spark 3.4.1 on 7.3.1.

Source cluster version	Source cluster Spark 2 version	Source cluster Spark 3 version	Target cluster version	Target cluster Spark 3 version	Spark 2 used with connectors[2]
7.2.18 SP2	2.4.8	3.4.1 (bundled)	**7.3.1**	**3.4.1**	no

Sidecar migration of Data Hub clusters

Procedure

Sidecar migration tasks for Data Hub clusters

The new 7.2.18 Data Hub cluster needs to use Spark 3 and Livy 3 instead of Spark 2 and Livy 2.

Depending on the template you used for your existing Data Hub clusters, a new custom template might be needed that contains Spark 3 instead of Spark 2. Alternatively, the built-in *7.2.18 - Data Engineering: Apache Spark3, Apache Hive, Apache Oozie* template can be used, as it contains Spark 3 only.

1. Check the current services in your template, and add the built-in *7.2.18 - : Apache Spark3, Apache Hive, Apache Oozie* template.
2. If the built-in *7.2.18 - : Apache Spark3, Apache Hive, Apache Oozie* template doesn't work, you can create a custom template. Replace all Spark 2 and Livy 2 references with Spark 3 and Livy 3, respectively.
3. Add a new Spark 3-based 7.2.18 cluster to the environment, using your custom template or the built-in *7.2.18 - : Apache Spark3, Apache Hive, Apache Oozie* template.
4. Migrate all non-spark workloads from the old cluster to the new cluster.

Application migration tasks (Spark 2 to 3)

Procedure

1. Follow the Spark application migration documentation to migrate your Apache Spark Applications from version 2.4.8 to 3.4.1
   a) Check the supported Java versions.
   b) Check the supported Scala version.
   c) Check the supported Python versions.
   d) Account for changed or versioned Spark commands in your code. (spark-submit, pyspark, etc.)
   e) Check supported versions for Spark connectors.
   f) Check the logging library used in your code.
   g) Check the compatibility of 3rd-party libraries used in your code.
   h) Check Spark behavior changes and refactor your code.
2. Migrate all Spark 2 applications in the old cluster to Spark 3 applications in the new cluster.

Post-application migration tasks

Procedure

1. Move Spark 2 event logs to the Spark 3 event logs directory.
2. Drop the old cluster.

In-place cluster upgrade

Before you begin

Important: Cluster upgrade to version 7.3.1 is only allowed if all Spark 2 applications have been migrated to Spark 3 and tested in the **source cluster**.

Procedure

1. Upgrade the Data Lake cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Stop all s attached to the environment.
   c) From the , click Data LakesEnvironment Name, scroll to the bottom of the **Data Lake details** page, and click the Upgrade tab.
   d) Click the Target Version drop-down menu to see any available upgrades.
   e) If you want to skip the automatic backup that is taken before the upgrade, uncheck the Automatic backup box.
   f) Click Validate and Prepare to check for any configuration issues and begin the parcel download and distribution.
   g) Click Upgrade to initiate the upgrade.
   h) Click the Event History tab to monitor the upgrade process and verify that it completes successfully.

   For more information, see Data Lake upgrade.

2. Upgrade the new cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Start the cluster.
   c) Check the current version of .
   d) If your cluster uses Streams Replication Manager, export or migrate aggregated metrics.
   e) If you use autoscaling, disable autoscaling on the cluster.
   f) Upgrade the cluster.
   g) Monitor the upgrade progress using the Event History tab.
   h) When the upgrade is complete, verify the new version.
   i) If you disabled autoscaling on the cluster, you can re-enable it after upgrade.

   For more information, see Upgrading s.

Final steps

Procedure

After the upgrade and application migration are complete:

1. Check the status of your Data Lakes, s, and clusters.
2. Perform benchmark testing on your applications. See Spark Application Migration.

Upgrading Apache Spark 3.4.1 (bundled) on 7.2.18 SP2 to Spark 3 on 7.3.1

The following steps will help you upgrading from Apache Spark 2.4.8 (with 3.4.1 bundled) on Cloudera Private Cloud 7.2.18 SP2 to Spark 3.4.1 on 7.3.1.

Source cluster version	Source cluster Spark 2 version	Source cluster Spark 3 version	Target cluster version	Target cluster Spark 3 version	Spark 2 used with connectors[3]
7.2.18 SP2	none	3.4.1 (bundled)	**7.3.1**	**3.4.1**	no

In-place cluster upgrade

Procedure

1. Upgrade the Data Lake cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Stop all s attached to the environment.
   c) From the , click Data LakesEnvironment Name, scroll to the bottom of the **Data Lake details** page, and click the Upgrade tab.
   d) Click the Target Version drop-down menu to see any available upgrades.
   e) If you want to skip the automatic backup that is taken before the upgrade, uncheck the Automatic backup box.
   f) Click Validate and Prepare to check for any configuration issues and begin the parcel download and distribution.
   g) Click Upgrade to initiate the upgrade.
   h) Click the Event History tab to monitor the upgrade process and verify that it completes successfully.

   For more information, see Data Lake upgrade.

2. Upgrade the new cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Start the cluster.
   c) Check the current version of .
   d) If your cluster uses Streams Replication Manager, export or migrate aggregated metrics.
   e) If you use autoscaling, disable autoscaling on the cluster.
   f) Upgrade the cluster.
   g) Monitor the upgrade progress using the Event History tab.
   h) When the upgrade is complete, verify the new version.
   i) If you disabled autoscaling on the cluster, you can re-enable it after upgrade.

   For more information, see Upgrading s.

Final steps

Procedure

After the upgrade and application migration are complete:

1. Check the status of your Data Lakes, s, and clusters.
2. Perform benchmark testing on your applications. See Spark Application Migration.

Upgrading from 7.2.17

Upgrading Apache Spark 2.4.8 on 7.2.17 to Spark 3 on 7.3.1

The following steps will help you upgrading from Apache Spark 2.4.8 on Cloudera Public Cloud 7.2.17 to Spark 3.4.1 on 7.3.1.

Source cluster version	Source cluster Spark 2 version	Source cluster Spark 3 version	Target cluster version	Target cluster Spark 3 version	Spark 2 used with connectors[4]
7.2.17	2.4.8	none	**7.3.1**	**3.4.1**	no

Intermediate in-place cluster upgrade

Procedure

Upgrade the cluster OS from Centos 7 to RedHat 8.

Sidecar migration of Data Hub clusters

Procedure

Sidecar migration tasks for Data Hub clusters

The new 7.2.17 Data Hub cluster needs to use Spark 3 and Livy 3 instead of Spark 2 and Livy 2.

Depending on the template you used for your existing Data Hub clusters, a new custom template might be needed that contains Spark 3 instead of Spark 2. Alternatively, the built-in *7.2.17 - Data Engineering: Apache Spark3* template can be used, as it contains Spark 3 only.

1. Check the current services in your template, and add the built-in *7.2.17 - : Apache Spark3* template.
2. If the built-in *7.2.17 - : Apache Spark3* template doesn't work, you can create a custom template. Replace Spark 2 and Livy 2 references with Spark 3 and Livy 3, respectively..

3. Add a new Spark 3-based 7.2.17 cluster to the environment, using your custom template or the built-in *7.2.17 - : Apache Spark3* template.
4. Migrate all non-spark workloads from the old cluster to the new cluster.

Application migration (Spark 2 to 3)

Procedure

1. Follow the Spark application migration documentation to migrate your Apache Spark Applications from version 2.4.8 to 3.3.2.
   a) Check the supported Java versions.
   b) Check the supported Scala version.
   c) Check the supported Python versions.
   d) Account for changed or versioned Spark commands in your code. (spark-submit, pyspark, etc.)
   e) Check supported versions for Spark connectors.
   f) Check the logging library used in your code.
   g) Check the compatibility of 3rd-party libraries used in your code.
   h) Check Spark behavior changes and refactor your code.
2. Migrate all Spark 2 applications in the old cluster to Spark 3 applications in the new cluster.

Post-application migration tasks

Procedure

1. Move Spark 2 event logs to the Spark 3 event logs directory.
2. Drop the old cluster.

In-place cluster upgrade

Before you begin

Important: Cluster upgrade to version 7.3.1 is only allowed if all Spark 2 applications have been migrated to Spark 3 and tested in the **source cluster**.

Procedure

1. Upgrade the Data Lake cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Stop all s attached to the environment.
   c) From the , click Data LakesEnvironment Name, scroll to the bottom of the **Data Lake details** page, and click the Upgrade tab.
   d) Click the Target Version drop-down menu to see any available upgrades.
   e) If you want to skip the automatic backup that is taken before the upgrade, uncheck the Automatic backup box.
   f) Click Validate and Prepare to check for any configuration issues and begin the parcel download and distribution.
   g) Click Upgrade to initiate the upgrade.
   h) Click the Event History tab to monitor the upgrade process and verify that it completes successfully.

   For more information, see Data Lake upgrade.

2. Upgrade the new cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Start the cluster.
   c) Check the current version of .
   d) If your cluster uses Streams Replication Manager, export or migrate aggregated metrics.
   e) If you use autoscaling, disable autoscaling on the cluster.
   f) Upgrade the cluster.
   g) Monitor the upgrade progress using the Event History tab.
   h) When the upgrade is complete, verify the new version.
   i) If you disabled autoscaling on the cluster, you can re-enable it after upgrade.

   For more information, see Upgrading s.

Application migration tasks (Spark 3.x to 3.4.1)

Procedure

Follow the Spark application migration documentation to migrate your Apache Spark Applications from version 3.3.2 to 3.4.1

a) Refactor your Spark application code.

Final steps

Procedure

After the upgrade and application migration are complete:

1. Check the status of your Data Lakes, s, and clusters.
2. Perform benchmark testing on your applications. See Spark Application Migration.

Upgrading Apache Spark 2.4.8 on 7.2.17 to Spark 3 on 7.3.1

The following steps will help you upgrading from Apache Spark 2.4.8 on Cloudera Public Cloud 7.2.17 to Spark 3.4.1 on 7.3.1.

Source cluster version	Source cluster Spark 2 version	Source cluster Spark 3 version	Target cluster version	Target cluster Spark 3 version	Spark 2 used with connectors[5]
7.2.17	2.4.8	none	**7.3.1**	**3.4.1**	yes

Intermediate in-place cluster upgrade

Procedure

1. Upgrade the cluster OS from Centos 7 to RedHat 8.
2. Upgrade your cluster to 7.2.18 SP2.
   a) Identify cluster version details.
   b) Identify your upgrade path.
   c) Review the prerequisites
   d) High-level upgrade steps.

   For more information on upgrading your cluster to 7.2.18 SP2, see:

   - Upgrading to 7.2.18

Upgrading Apache Spark 2.4.8 (with 3.3.2 bundled) on 7.2.17 to Spark 3 on 7.3.1

The following steps will help you upgrading from Apache Spark 2.4.8 (with 3.3.2 bundled) on Cloudera Public Cloud 7.2.17 to Spark 3.4.1 on 7.3.1.

Source cluster version	Source cluster Spark 2 version	Source cluster Spark 3 version	Target cluster version	Target cluster Spark 3 version	Spark 2 used with connectors[6]
7.2.17	2.4.8	3.3.2 (bundled)	**7.3.1**	**3.4.1**	no

Intermediate in-place cluster upgrade

Procedure

Upgrade the cluster OS from Centos 7 to RedHat 8.

Intermediate application migration tasks

Procedure

Follow the Spark application migration documentation to migrate your Apache Spark Applications from version 3.3.2 to 3.4.1

a) Refactor your Spark application code.

Sidecar migration of Data Hub clusters

Procedure

The new 7.2.17 Data Hub cluster needs to use Spark 3 and Livy 3 instead of Spark 2 and Livy 2.

Depending on the template you used for your existing Data Hub clusters, a new custom template might be needed that contains Spark 3 instead of Spark 2. Alternatively, the built-in *7.2.17 - Data Engineering: Apache Spark3* template can be used, as it contains Spark 3 only.

1. Check the current services in your template, and add the built-in *7.2.17 - : Apache Spark3* template.
2. If the built-in *7.2.17 - : Apache Spark3* template doesn't work, you can create a custom template. Replace Spark 2 and Livy 2 references with Spark 3 and Livy 3, respectively..
3. Add a new Spark 3-based 7.2.17 cluster to the environment, using your custom template or the built-in *7.2.17 - : Apache Spark3* template.
4. Migrate all non-spark workloads from the old cluster to the new cluster.

Application migration tasks (Spark 2 to 3)

Procedure

1. Follow the Spark application migration documentation to migrate your Apache Spark Applications from version 2.4.8 to 3.3.2.
   a) Check the supported Java versions.
   b) Check the supported Scala version.
   c) Check the supported Python versions.
   d) Account for changed or versioned Spark commands in your code. (spark-submit, pyspark, etc.)
   e) Check supported versions for Spark connectors.
   f) Check the logging library used in your code.
   g) Check the compatibility of 3rd-party libraries used in your code.
   h) Check Spark behavior changes and refactor your code.
2. Migrate all Spark 2 applications in the old cluster to Spark 3 applications in the new cluster.

Post-application migration tasks

Procedure

1. Move Spark 2 event logs to the Spark 3 event logs directory.
2. Drop the old cluster.

In-place cluster upgrade

Before you begin

Important: Cluster upgrade to version 7.3.1 is only allowed if all Spark 2 applications have been migrated to Spark 3 and tested in the **source cluster**.

Procedure

1. Upgrade the Data Lake cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Stop all s attached to the environment.
   c) From the , click Data LakesEnvironment Name, scroll to the bottom of the **Data Lake details** page, and click the Upgrade tab.
   d) Click the Target Version drop-down menu to see any available upgrades.
   e) If you want to skip the automatic backup that is taken before the upgrade, uncheck the Automatic backup box.
   f) Click Validate and Prepare to check for any configuration issues and begin the parcel download and distribution.
   g) Click Upgrade to initiate the upgrade.
   h) Click the Event History tab to monitor the upgrade process and verify that it completes successfully.

   For more information, see Data Lake upgrade.
2. Upgrade the new cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Start the cluster.
   c) Check the current version of .
   d) If your cluster uses Streams Replication Manager, export or migrate aggregated metrics.
   e) If you use autoscaling, disable autoscaling on the cluster.
   f) Upgrade the cluster.
   g) Monitor the upgrade progress using the Event History tab.
   h) When the upgrade is complete, verify the new version.
   i) If you disabled autoscaling on the cluster, you can re-enable it after upgrade.

   For more information, see Upgrading s.

Application migration tasks (Spark 3.x to 3.4.1)

Procedure

Follow the Spark application migration documentation to migrate your Apache Spark Applications from version 3.3.2 to 3.4.1

a) Refactor your Spark application code.

Final steps

Procedure

After the upgrade and application migration are complete:

1. Check the status of your Data Lakes, s, and clusters.
2. Perform benchmark testing on your applications. See Spark Application Migration.

Upgrading Apache Spark 2.4.8 (with 3.3.2 bundled) on 7.2.17 to Spark 3 on 7.3.1

The following steps will help you upgrading from Apache Spark 2.4.8 (with 3.3.2 bundled) on Cloudera Public Cloud 7.2.17 to Spark 3.4.1 on 7.3.1.

Source cluster version	Source cluster Spark 2 version	Source cluster Spark 3 version	Target cluster version	Target cluster Spark 3 version	Spark 2 used with connectors[7]
7.2.17	2.4.8	3.3.2 (bundled)	**7.3.1**	**3.4.1**	yes

Intermediate in-place cluster upgrade

Procedure

1. Upgrade the cluster OS from Centos 7 to RedHat 8.
2. Upgrade your cluster to 7.2.18 SP2.
   a) Identify cluster version details.
   b) Identify your upgrade path.
   c) Review the prerequisites
   d) High-level upgrade steps.

   For more information on upgrading your cluster to 7.2.18 SP2, see:

   - Upgrading to 7.2.18

Upgrading Apache Spark 3.3.2 (bundled) on 7.2.17 to Spark 3 on 7.3.1

The following steps will help you upgrading from Apache Spark 3.3.2 (bundled) on Cloudera Public Cloud 7.2.17 to Spark 3.4.1 on 7.3.1.

Source cluster version	Source cluster Spark 2 version	Source cluster Spark 3 version	Target cluster version	Target cluster Spark 3 version	Spark 2 used with connectors[8]
7.2.17	none	3.3.2 (bundled)	**7.3.1**	**3.4.1**	no

In-place cluster upgrade

Procedure

1. Upgrade the cluster OS from Centos 7 to RedHat 8.

2. Upgrade the Data Lake cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Stop all s attached to the environment.
   c) From the , click Data LakesEnvironment Name, scroll to the bottom of the **Data Lake details** page, and click the Upgrade tab.
   d) Click the Target Version drop-down menu to see any available upgrades.
   e) If you want to skip the automatic backup that is taken before the upgrade, uncheck the Automatic backup box.
   f) Click Validate and Prepare to check for any configuration issues and begin the parcel download and distribution.
   g) Click Upgrade to initiate the upgrade.
   h) Click the Event History tab to monitor the upgrade process and verify that it completes successfully.

   For more information, see Data Lake upgrade.
3. Upgrade the new cluster to 7.3.1
   a) Check the support matrix for upgrades.
   b) Start the cluster.
   c) Check the current version of .
   d) If your cluster uses Streams Replication Manager, export or migrate aggregated metrics.
   e) If you use autoscaling, disable autoscaling on the cluster.
   f) Upgrade the cluster.
   g) Monitor the upgrade progress using the Event History tab.
   h) When the upgrade is complete, verify the new version.
   i) If you disabled autoscaling on the cluster, you can re-enable it after upgrade.

   For more information, see Upgrading s.

Application migration tasks (Spark 3.x to 3.4.1)

Procedure

Follow the Spark application migration documentation to migrate your Apache Spark Applications from version 3.3.2 to 3.4.1

a) Refactor your Spark application code.

Final steps

Procedure

After the upgrade and application migration are complete:

1. Check the status of your Data Lakes, s, and clusters.
2. Perform benchmark testing on your applications. See Spark Application Migration.

Migrating Spark applications

How to refactor Spark 2 workloads to Spark 3 during the upgrade/migration process due to the removal of Spark 2 in Cloudera Public Cloud.

Introduction

The purpose of this document is to gather all information required to carry out a Spark application migration between different versions.

The necessary set of steps largely depends on the source and target Spark versions, while major version changes require considerable effort, minor and maintenance version changes mostly require only small config or no adjustments.

Major version migration

Migration between major versions requires considerable effort and taking into account many factors.

This documentation focuses on migrating applications from Spark 2 to Spark 3, the two major versions currently supported by Cloudera on different versions of Cloudera Public Cloud.

Java versions

Cloudera currently supports 3 major JDK versions in general:

- 8
- 11
- 17

Refer to *Support Matrix* for a list of supported versions of Java.

Important: For Java 11, setting -Dio.netty.tryReflectionSetAccessible=true is required for the Apache Arrow library. This prevents the java.lang.UnsupportedOperationException: sun.misc.Unsafe or java.nio.DirectByteBuffer.(long, int) not available error when Apache Arrow uses Netty internally.

Related Information

Cloudera Support Matrix

Scala versions

As Cloudera only supports only Spark 2 applications compiled with Scala **2.11**, and Spark 3 applications with Scala **2.12**, a major version change always require:

1. Spark Scala applications to be recompiled with Scala 2.12,
2. adjusting the dependencies to use Spark 3 version binaries provided by Cloudera in the *public maven repository* and the Scala 2.12 version of third-party libraries.

Scala version changes can also require source code changes, for which see the *Scala documentation*.

Related Information

Cloudera public maven repositories

Cloudera Public Cloud | Using the Cloudera Runtime Maven repository

Scala documentation | Scala

Python versions

The supported versions of Python can change between Spark versions. Refer to the table below for details on the supported Python versions for each Spark version, and follow the *Python documentation* to adjust your application.

Spark version	Minimum supported Python version	Maximum supported Python version
3.5.0	3.8	3.11
3.4.0	3.7	3.11
3.3.2	3.7	3.10
3.3.0	3.7	3.10
3.2.3	3.6	3.9
2.4.8	2.7/3.4	3.7

Related Information

Python documentation | Python

Spark commands

Cloudera supports multiple versions of Spark, depending on the version of Cloudera Public CloudCloudera Data Hubs. The general (unversioned) Apache Spark commands (spark-submit, pyspark, etc.) can point to different versions based on the cluster version.

1. The original commands always point to the earliest available version of Spark in the distribution.

   For example, the spark-submit command points to **Spark 2 in version 7.2.18**, but points to **Spark 3 in version 7.3.1**.
2. Other available Spark 3 versions can be used via versioned commands.

   For example, the spark3-submit command points to Spark 3 in all versions.

Spark connectors

Spark 3 supports certain Spark connectors from certain versions.

If Spark 2 connectors are used, please take the connectors into account when choosing the minimum Cloudera Public Cloud version you need to upgrade to when migrating a Spark application to a higher version.

- *Hive Warehouse Connector* for Spark 3 is supported from:
  - Cloudera Public Cloud version 7.2.16
- *HBase connector* for Spark 3 is supported from:
  - Cloudera Public Cloud version 7.2.12
- *Phoenix* connector for Spark 3 is supported from:
  - Cloudera Public Cloud version 7.2.15
- *Oozie* for Spark 3 is supported from:
  - Cloudera Public Cloud version 7.2.18
- *Solr* for Spark 3 is supported from:
  - Cloudera Public Cloud version 7.2.18
- *Spark Schema Registry* is supported from:
  - Cloudera Public Cloud version 7.2.18 SP2

Logging

Since Apache Spark 3 has transitioned from log4j to log4j2, you need to adjust the logging library and/or logging configuration used in your application.

Important: The log4j and reload4j runtime libraries are no longer included in the classpath by default. They have been replaced with the analogous log4j2 libraries.

Third-party libraries

When migrating between versions, ensure that your 3rd-party runtime dependencies align with the Spark versions.

Spark behavior changes

As Apache Spark evolves, its behavior can change between major and minor versions, but many times legacy configurations are added to support the old behavior.

As configurations can be defined at multiple levels, restoring the old behavior might require changing the application itself, the application starting commands/scripts, and/or the default Spark configurations defined on the cluster.

Apache Spark Migration guides

The comprehensive guide of behavior changes between versions are available in the Apache Spark Migration Guides.

Always refer to the following documents to ensure that your Spark application remains compatible with newer Spark versions:

- Migration Guide: Spark Core
- Migration Guide: SQL, Datasets and DataFrame
- Structured Streaming Programming Guide
- Machine Learning Library (MLlib) Guide
- Upgrading PySpark

Spark 2 to Spark 3 workload refactoring

The following list summarizes the most important behavior changes from Spark 2 to Spark 3, and gives examples on how to refactor the Spark 2 application to become Spark 3 compatible.

The list is not exhaustive, refer to the *Apache Spark Migration guides* for the complete list.

Spark Core

Spark Core language/syntactic-level changes

Spark 2	Spark 3	Refactor action
TaskContext.isRunningLocally	Deprecated method, removed.	Remove TaskContext.isRunningLocally if used in code.
ShuffleBytesWritten and shuffleRecordsWritten (ShuffleWriteMetrics class)	bytesWritten and recordsWritten (org.apache.spark.status.api.v1.OutputMetrics class)	Use bytesWritten and recordsWritten, available in class org.apache.spark.status.api.v1.OutputMetrics.
org.apache.spark Class Accumulator	org.apache.spark.util.AccumulatorV2	Replace org.apache.spark.Accumulator with org.apache.spark.util.AccumulatorV2.
For non-struct types, (e.g. int, string, array, Dataset.groupByKey) results in a grouped dataset with key attribute is wrongly named as value.	For non-struct types (e.g. int, string, array, Dataset.groupByKey) results to a grouped dataset with key attribute is named as key.	Refactor the value attribute used in logic to key. To preserve the old behavior, set spark.sql.legacy.dataset.nameNonStructGroupingKeyAsValue to false.

Spark SQL

Spark SQL language/syntactic-level changes

Spark 2	Spark 3	Refactor action
Path option is overwritten if one path parameter is passed to DataFrameReader.load(), DataFrameWriter.save(), DataStreamReader.load(), or DataStreamWriter.start().	Path option cannot coexist when the following methods are called with path parameter(s): DataFrameReader.load(), DataFrameWriter.save(), DataStreamReader.load(), or DataStreamWriter.start().	Remove the path option if it's the same as the path parameter, or add it to the load() parameter if you do want to read multiple paths. To ignore this check, set spark.sql.legacy.pathOptionBehavior.enabled to true.
count(tblName.*) works.	An exception is thrown if count(tblName.*) is used for getting the number of records in the table.	Refactor the code to use count(*), or expand the columns manually. (Example: count(col1, col2).) To restore the old behavior, set spark.sql.legacy.allowStarWithSingleTableIdentifierInCount to true.

Spark SQL configuration-level changes

Spark 2	Spark 3	Refactor action
SET command works for SparkConf entries.	AnalysisException error is thrown if SET command is used to modify the SparkConf entries.	Remove SET commands for SparkConf entries from your code. You can enter SparkConf values at the cluster level by entering them in the cluster's Spark configuration and restarting the cluster. To disable the check, set spark.sql.legacy.setCommandRejectsSparkCoreConfs to false.
The second argument of date_add function (num_days) can be a fraction, as it gets casted to Int internally.	The second argument of date_add function (num_days). If an integer is not provided, an AnalysisException is thrown.	Make sure that in code always integer is passed as the second argument to date_add and date_subtract function.
Fractional and string types are allowed in percentile_approx third argument i.e. accuracy, as it gets casted to Int internally.	percentile_approx third argument accuracy can only be integer. If an integer is not provided, an AnalysisException is thrown.	Make sure that in code always integer is passed as the third argument to percentile_approx function.
Hash expressions can be applied on MapType elements.	Hash expressions are prohibited on MapType elements.	If hash expression is applied on map type, refactor the code to remove it, OR set spark.sql.legacy.allowHashOnMapType to true.
a map can be created with duplicate keys via built-in functions like CreateMap, StringToMap, map_from_arrays etc.	Spark throws RuntimeException when duplicated keys are found in Map. Users may still read map values with duplicate keys from data sources which do not enforce it (for example, Parquet).	If duplicate keys are passed into built in functions to create a map then try to remove duplicate keys OR set spark.sql.mapKeyDedupPolicy to LAST_WIN, the map keys are deduplicated.
the resulting date is adjusted in add_months, when the original date is a last day of months. For example, adding a month to 2019-02-28 results in 2019-03-31.	the add_months function does not adjust the resulting date to a last day of month if the original date is a last day of months. For example, adding a month to 2019-02-28 results in 2019-03-28.	Adjust the code according to logic if required.
multiple from-to units is allowed in Interval literal.	multiple from-to units Interval literal is not allowed.	Remove multiple from-to units is allowed in Interval literal. Adjust the code according to logic if required.
Dataset query success if it contains ambiguous column reference that is caused by self join.	Dataset query fails if it contains ambiguous column reference that is caused by self join. This is because Spark cannot resolve Dataset column references that point to tables being self joined, and df1("a") is exactly the same as df2("a") in Spark.	Use aliases. For example: df2.as("purchases").join(df1.as("devices"), col("devices.key1") === col("purchases.key2")).show()
invalid time zone ids are silently ignored and replaced by GMT timezone.	invalid time zone ids are rejected, and Spark throws java.time.DateTimeException.	rectify to correct Zone ID.
for Parsing and formatting of timestamp and date strings, java.text.SimpleDateFormat is used for timestamp/date string conversions, and the supported patterns are described in SimpleDateFormat.	DateTimeFormatter under the hood or Parsing and formatting of timestamp and date strings. Strict checking of Input is performed.	Refactor the code to correct pattern matching for Input OR set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior, OR set it to CORRECTED and treat it as an invalid datetime string.
datetime pattern letter F is aligned to week of month that represents the concept of the count of weeks within the month where weeks start on a fixed day-of-week.	datetime pattern letter F is aligned to day of week in month that represents the concept of the count of days within the period of a week where the weeks are aligned to the start of the month.	Refactor the code to accommodate new behavior of pattern F.
SparkContext can be created in executors.	an exception will be thrown when creating SparkContext in executor.	Refactor the code to remove the creation of Spark context OR allow it by setting the configuration spark.executor.allowSparkContext when creating SparkContext in executors.
TRANSFORM operator can support alias in inputs.	TRANSFORM operator can't support alias in inputs.	Refactor the code to remove aliases from Inputs.

Spark 2	Spark 3	Refactor action
Loading and saving of timestamps from and to Parquet files does not fail if the timestamps are before 1900-01-01 00:00:00Z.	Loading and saving of timestamps from and to Parquet files fails if the timestamps are before 1900-01-01 00:00:00Z.	Ensure that Input reads do not contain timestamps before 1900-01-01 00:00:00Z. Alternatively, set spark.sql.parquet.int96RebaseModeInWrite to CORRECTED to write the datetime values as it is.
The Char(n) type handled inconsistently, depending on whether the table is partitioned or not.	In upstream Spark 3 the spark.sql.legacy.charVarcharAsString config was introduced, but does not solve all incompatibilities.	A new configuration spark.cloudera.legacy.charVarcharLegacyPadding is introduced in Cloudera to keep the full compatibility. **Note:** Although this workaround is available, using CHAR is discouraged.
The Row field names are sorted alphabetically when constructing with named arguments for Python versions 3.6 and above.	The Row field names are no longer sorted alphabetically.	To enable sorted fields by default as in Spark 2.4, set the environment variable PYSPARK_ROW_FIELD_SORTING_ENABLED to true for both the executors and the driver.

Spark SQL property-level changes

Spark 2	Spark 3	Refactor action
When there is nested CTE with a conflicting name, outer CTE definitions take precedence.	When there is nested CTE with conflicting name, Spark throws an AnalysisException by default, and forces users to choose the specific substitution order they wanted. If the value of spark.sql.legacy.ctePrecedencePolicy is set to CORRECTED (recommended), inner CTE definitions take precedence over outer definitions . If the value of spark.sql.legacy.ctePrecedencePolicy is set to LEGACY, outer CTE definitions take precedence over inner definitions.	Set spark.sql.legacy.ctePrecedencePolicy to CORRECTED.
Type conversions during table insertion are allowed as long as they are valid Cast.	The type coercion is performed as per the ANSI SQL standard.	Ensure the type coercion is performed as per the ANSI SQL standard. Alternatively, set spark.sql.storeAssignmentPolicy to Legacy to restore previous behavior.

Spark storage location configuration changes

To execute workloads in Cloudera Public Cloud, modify the local data storage locations to cloud storage (for example, from HDFS to S3 bucket).

The following example shows a sample workload, with the modified data location **highlighted in bold**.

Spark 3.2 (HDFS)	Spark 3.2 (S3)
spark-shell	spark-shell
`scala> spark.sql("CREATE TABLE IF NOT EXISTS default.sales_spark_2(Region string, Country string,Item_Type string,Sales_Channel string,Order_Priority string,Order_Date date,Order_ID int,Ship_Date date,Units_sold string,Unit_Price string,Unit_cost string,Total_revenue string,Total_Cost string,Total_Profit string) row format delimited fields terminated by ','")` `scala> spark.sql("load data local inpath '/tmp/sales.csv' into table default.sales_spark_3")`	`scala> spark.sql("CREATE TABLE IF NOT EXISTS default.sales_spark_2(Region string, Country string,Item_Type string,Sales_Channel string,Order_Priority string,Order_Date date,Order_ID int,Ship_Date date,Units_sold string,Unit_Price string,Unit_cost string,Total_revenue string,Total_Cost string,Total_Profit string) row format delimited fields terminated by ','")` `scala> spark.sql("load data inpath 's3://[*** BUCKET ***]/sales.csv' into table default.sales_spark_3")`
`scala> spark.sql("select count(*) from default.sales_spark_3").show()`	`scala> spark.sql("select count(*) from default.sales_spark_3").show()`

Unsupported features

Unsupported Spark 3 features in Cloudera Public Cloud.

ZSTD compression in ORC data source

spark.sql.orc.compression.codec config doesn't accept zstd value.

Apache Jira:SPARK-33978

spark.hadoopRDD.ignoreEmptySplits

Causes issues in HBase TableInputFormat.

Apache Jira:SPARK-34809

LDAP authentication for livy-server

Open CVEs in Apache Directory Server dependency, LDAP based authentication is not supported in Livy Server.

Apache Jira:LIVY-356

Thrift ldap authentication, based on ldapurl, basedn, domain

Open CVEs in Apache Directory Server dependency, LDAP based authentication is not supported in Livy Thrift Server.

Apache Jira:LIVY-678

For more information, see *Unsupported Apache Spark Features*.

Related Information

Unsupported Apache Spark Features

Post-migration checklist

Benchmark testing

After all post-migration configurations are performed, perform benchmark testing on the new Apache Spark version.

Troubleshooting

Troubleshoot failed or slow performing workloads by analyzing the application event and driver logs, and fine tune the workloads for better performance.

Cloudera Runtime 7.3.1

Use cases for Streams Replication Manager in Cloudera Public Cloud

Date published: 2019-08-22
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using Streams Replication Manager in Cloudera Public Cloud overview..... 4

Replicating data from on premises to cloud with Streams Replication Manager on premises..... 5

Replicating data from on premises to cloud with Streams Replication Manager in the cloud..... 11

Replicating data between cloud clusters with Streams Replication Manager in the cloud..... 17

Using Streams Replication Manager in Cloudera Public Cloud overview

Learn about the deployment options, prerequisites, and use cases for Streams Replication Managerin a cloud-based context.

Starting with the December 2020 release of Cloudera Public Cloud, Streams Replication Manager is included in the default Streams Messaging cluster definitions. As a result, you can deploy Streams Replication Manager in a Cloudera Data Hub cluster and use it to replicate Kafka data between all types of Cloudera clusters. This includes replicating Kafka data to or from clusters deployed on either Cloudera Public Cloud or Cloudera Private Cloud.

The following sections provide information on how you can deploy Streams Replication Manager in a Cloudera Data Hub cluster, what prerequisites you must meet before using Streams Replication Manager, and the common use cases where you can use Streams Replication Manager in a cloud-based context.

Differences between light and heavy deployments

In Cloudera Public Cloud, you can deploy Streams Replication Manager in Cloudera Data Hub clusters with both the light and heavy duty variants of the Streams Messaging cluster definition. However, there are significant differences in how Streams Replication Manager is deployed with each definition.

Light duty definition:

In the light duty definition, Streams Replication Manager is deployed by default on the broker and master hosts of the cluster. This means that Streams Replication Manager is available for use by default in a Cloudera Data Hub cluster provisioned with the light duty definition.

Heavy duty definition

In the heavy duty definition, Streams Replication Manager has its own host group. However, by default, the Streams Replication Manager host group is not provisioned. When creating a cluster with the heavy duty definition, you must set the instance count of the Srm nodes host group to at least one. Otherwise, Streams Replication Manager is not deployed on the cluster.

For more information on cluster provisioning, see *Creating your first Streams Messaging cluster*. For more information on the default cluster definitions and cluster layouts, see *Streams Messaging cluster layout*.

Note: Deploying Streams Replication Manager in a Cloudera Data Hub cluster requires Cloudera Runtime 7.2.6 or higher.

Prerequisites for using Streams Replication Manager

Streams Replication Manager can be used to replicate Kafka data between all types of Cloudera clusters. However, the following conditions must be met for all deployments and use cases:.

- Streams Replication Manager must be able to access the Kafka hosts of the source and target cluster through the network.
- Streams Replication Manager must trust the TLS certificates of the brokers in the source and target clusters.

  This is required so that Streams Replication Manager can establish a trusted connection.
- Streams Replication Manager must have access to credentials that it can use to authenticate itself in both the source and target clusters.
- Streams Replication Manager must use a principal that is authorized to access Kafka resources (topics) on both source and target clusters.

Cloud-based use cases for Streams Replication Manager

There are three common uses cases when using Streams Replication Manager in a cloud-based context. These are as follows.

Replicating data from on premises to cloud with Streams Replication Manager on premises

In this use case you replicate data from a Cloudera Private Cloud Base cluster to a Cloudera Data Hub cluster with Streams Replication Manager running in the Cloudera Private Cloud Base cluster.

Note: Although in this use case Streams Replication Manager is not deployed in a Cloudera Data Hub cluster, it is still considered a cloud-based use case. This is because the source or the destination of the replicated data is a Cloudera Data Hub cluster.

Replicating data from on premises to cloud with Streams Replication Manager in the cloud

In this use case you replicate data from a Cloudera Private Cloud Base cluster to a Cloudera Data Hub cluster with Streams Replication Manager running in the Cloudera Data Hub cluster.

Replicating data between cloud clusters with Streams Replication Manager in the cloud

In this use case you replicate data between Cloudera Data Hub clusters with Streams Replication Manager running in a Cloudera Data Hub cluster.

Related Information

Setting up your Streams Messaging cluster

Streams Messaging cluster layout

Replicating data from on premises to cloud with Streams Replication Manager on premises

You can set up and configure an instance of Streams Replication Manager running in a Cloudera Private Cloud Base cluster to replicate data between the Cloudera Private Cloud Base cluster and a Cloudera Data Hub cluster. In addition, you can use Streams Messaging Manager to monitor the replication process.

About this task

Consider the following replication scenario.

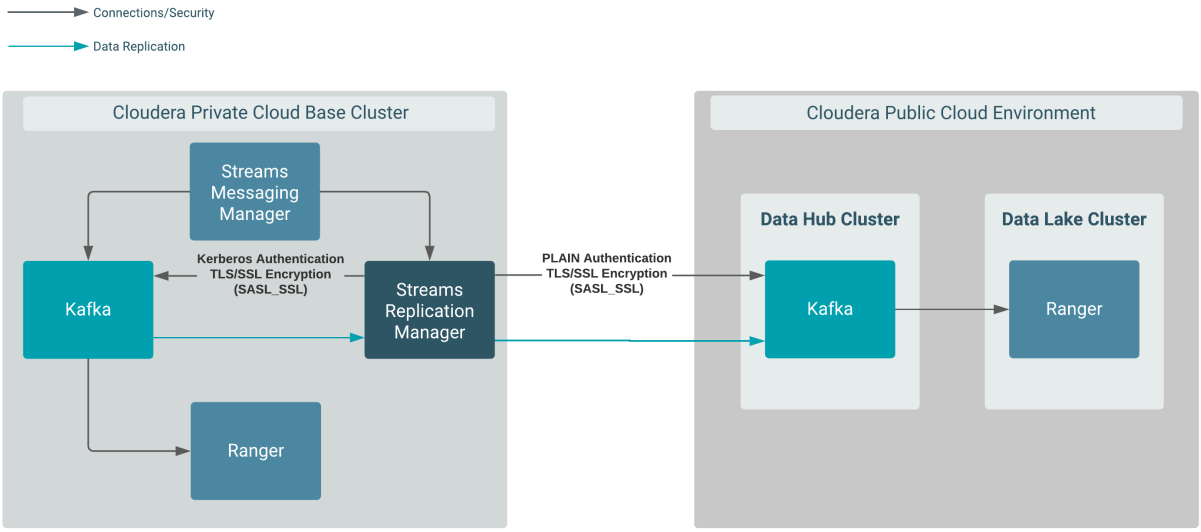

In this scenario, data is replicated from a Cloudera Private Cloud Base cluster that has Kafka, Streams Replication Manager, and Streams Messaging Manager deployed on it. This is a secure cluster that has TLS/SSL encryption and Kerberos authentication enabled. In addition, it uses Ranger for authorization.

Data is being replicated from this cluster by Streams Replication Manager deployed in this cluster to a Cloudera Data Hub cluster.

The Cloudera Data Hub cluster is provisioned with one of the default Streams Messaging cluster definitions.

Before you begin

This example scenario does not go into detail on how to set up the clusters and assumes the following.

- A Cloudera Data Hub provisioned with the Streams Messaging Light Duty or Heavy Duty cluster definition is available.

  For more information, see Setting up your Streams Messaging cluster in the CDF for Data Hub library. Alternatively, you can also review the cloud provider specific cluster creation instructions available in the Cloudera Data Hub library.
- A Cloudera Private Cloud Base Base cluster with Kafka, Streams Replication Manager, and Streams Messaging Manager is available. This cluster is TLS/SSL and Kerberos enabled. In addition, it uses Ranger for authorization.

  For more information, see the Cloudera Private Cloud Base Installation Guide.
- Network connectivity and DNS resolution are established between the clusters.
- This example scenario demonstrates the configuration required to enable replication monitoring of the Cloudera Data Hub cluster with Streams Messaging Manager. This can be done by configuring the Streams Replication Manager Service role to target (monitor) the Cloudera Data Hub cluster. This is done as the last step in the following list of steps and is marked optional. This is because enabling replication monitoring of the Cloudera Data Hub cluster results in a number of caveats. The caveats are the following.
  - The Streams Replication Manager Service role will generate additional cloud traffic.

    Any extra traffic you might have in your cloud deployment can lead to additional cloud costs.
  - The replications tab in Streams Messaging Manager will display all replications targeting the Cloudera Data Hub cluster.

    Although this is expected, you must understand that all other pages in Streams Messaging Manager will display information regarding the Cloudera Private Cloud Base cluster. A setup like this might lead to confusion or mislead users on what this specific instance of Streams Messaging Manager is monitoring.
  - You will lose the ability to monitor the replications targeting the Cloudera Private Cloud Base cluster.

    This is only critical if you have any existing replications that are targeting the Cloudera Private Cloud Base cluster and you are monitoring these replications with the Streams Messaging Manager instance running in the Cloudera Private Cloud Base cluster.

Important: In the following scenario, a new Cloudera machine user is created and set up specifically for Streams Replication Manager. Alternatively, it is also possible to use an existing machine user and skip steps 1 through 3, but this can only be done if the following requirements are met.

- The existing machine user has access to your Cloudera environment.
- The existing machine user has the correct Ranger permissions assigned to it.
- You have access to the existing machine user's credentials.

Procedure

1. Create a machine user for Streams Replication Manager in Cloudera Management Console.

   A machine user is required so that Streams Replication Manager has credentials that it can use to connect to the Kafka service in the Cloudera Data Hub cluster.

   a) Navigate to Management Console User Management.
   b) Click Actions Create Machine User .
   c) Enter a unique name for the user and click Create.

      For example: srm

      After the user is created, you are presented with a page that displays the user details.

      Note:

      The Workload User Name (srv_srm), is different from the actual machine user name (srm). The Workload User Name is the identifier you use to configure Streams Replication Manager.
   d) Click Set Workload Password.
   e) Type a password in the Password and Confirm Password fields. Leave the Environment field blank.
   f) Click Set Workload Password.
      A message appears on successful password creation.

2. Grant the machine user access to your environment.

   You must grant the machine user access to your environment for Streams Replication Manager to connect to the Kafka service with this user.

   a) Navigate to Management Console Environments , and select the environment where your Kafka cluster is located.
   b) Click Actions Manage Access .

      Use the search box to find and select the machine user you want to use.

      A list of **Resource Roles** appears.
   c) Select the EnvironmentUser role and click Update Roles.
   d) Go back to the **Environment Details** page and click Actions Synchronize Users to FreeIPA .
   e) On the **Synchronize Users** page, click Synchronize Users.

      Synchronizing users ensures that the role assignment is in effect for the environment.

      Important: Wait until this process is completed. Otherwise, you will not be able to continue with the next step.

3. Add Ranger permissions for the user you created for Streams Replication Manager in the Cloudera Data Hub cluster.

   You must to grant the necessary privileges to the user so that the user can access Kafka resources. This is configured through Ranger policies.

   a) Navigate to Management Console Environments , and select the environment where your Kafka cluster is located.
   b) Click the Ranger link on the **Environment Details** page.
   c) Select the resource-based service corresponding to the Kafka resource in the Cloudera Data Hub cluster.
   d) Add the Workload User Name of the user you created for Streams Replication Manager to the following Ranger policies.

      - All - consumergroup
      - All - topic
      - All - transactionalid
      - All - cluster
      - All - delegationtoken

4. Ensure that Ranger permissions exist for the streamsrepmgr user in the Cloudera Private Cloud Base cluster.
   a) Access the Cloudera Manager instance of your Cloudera Private Cloud Base cluster.
   b) Go to Ranger Ranger Admin Web UI .
   c) Log in to the Ranger Console (Ranger Admin Web UI).
   d) Ensure that the streamsrepmgr user is added to all required policies.

      If the user is missing, add it. The required policies are as follows.

      - All - consumergroup
      - All - topic
      - All - transactionalid
      - All - cluster
      - All - delegationtoken
5. Create a truststore on the Cloudera Private Cloud Base cluster.

   A truststore is required so that the Streams Replication Manager instance running in the Cloudera Private Cloud Base cluster can trust the secure Cloudera Data Hub cluster. To do this, you extract the FreeIPA certificate from the Cloudera environment, create a truststore that includes the certificate, and copy the truststore to all hosts on the Cloudera Private Cloud Base cluster.
   a) Navigate to Management Console Environments , and select the environment where your Kafka cluster is located.
   b) Go to the FreeIPA tab.
   c) Click Get FreeIPA Certificate.
      The FreeIPA certificate file, *[***ENVIRONMENT NAME***]*.crt, is downloaded to your computer.
   d) Run the following command to create the truststore.

      ```
      keytool \
        -importcert \
        -storetype JKS \
        -noprompt \
        -keystore datahub-truststore.jks \
        -storepass [***PASSWORD***] \
        -alias freeipa-ca \
        -file [***PATH TO FREEIPA CERTIFICATE***]
      ```
   e) Copy the datahub-truststore.jks file to a common location on all the hosts in your Cloudera Private Cloud Base cluster.

      Cloudera recommends that you use the following location: /opt/cloudera/security/datahub-truststore.jks.
   f) Set the correct file permissions.

      Use 751 for the directory and 444 for the truststore file.
6. Access the Cloudera Manager instance of your Cloudera Private Cloud Base cluster.
7. Define the external Kafka cluster (the Cloudera Data Hub cluster).
   a) Go to Administration External Accounts .
   b) Go to the Kafka Credentials tab.

      On this tab you will create a credential for each external cluster taking part in the replication process.
   c) Click Add Kafka credentials.
   d) Configure the Kafka credentials.

      In the case of this example, you must create a single credential representing the Cloudera Data Hub cluster. For example:

      ```
      Name=datahub
      Bootstrap servers=[**MY-CLOUDERA-DATA-HUB-CLUSTER-
      HOST-1.COM:9093***],[***MY-CLOUDERA-DATA-HUB-CLUSTER-HOST-1.COM:9093***]
      Security Protocol=SASL_SSL
      JAAS Secret 1=[***WORKLOAD USER NAME***]
      JAAS Secret 2=[***MACHINE USER PASSWORD***]
      ```

```
JAAS Template=org.apache.kafka.common.security.plain.PlainLoginModule r
equired username="##JAAS_SECRET_1##" password="##JAAS_SECRET_2##";
SASL Mechanism=PLAIN
Truststore Password=[***PASSWORD***]
Truststore Path=/OPT/CLOUDERA/SECURITY/DATAHUB-TRUSTSTORE.JKS
Truststore type=JKS
```

e) Click Add.

If credential creation is successful, a new entry corresponding to the Kafka credential you specified appears on the page.

8. Define the co-located Kafka cluster (the Cloudera Private Cloud Base cluster).

a) In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.

b) Go to Configuration.

c) Find and enable the Kafka Service property.

d) Find and configure the Streams Replication Manager Co-located Kafka Cluster Alias property.

The alias you configure represents the co-located cluster. Enter an alias that is unique and easily identifiable. For example:

```
cdppvc
```

e) Enable relevant security feature toggles.

Because the Cloudera Private Cloud Base cluster is both TLS/SSL and Kerberos enabled, you must enable all feature toggles for both the Streams Replication Manager Driver and Service roles. The feature toggles are the following.

- Enable TLS/SSL for SRM Driver
- Enable TLS/SSL for SRM Service
- Enable Kerberos Authentication

9. Add both clusters to the configuration of Streams Replication Manager.

a) Find and configure the External Kafka Accounts property.

Add the name of all Kafka credentials you created to this property. This can be done by clicking the add button to add a new line to the property and then entering the name of the Kafka credential. For example:

```
datahub
```

b) Find and configure the Streams Replication Manager Cluster alias property.

Add all cluster aliases to this property. This includes the aliases present in both the External Kafka Accounts and Streams Replication Manager Co-located Kafka Cluster Alias properties. Delimit the aliases with commas. For example:

```
datahub,cdppvc
```

10. Configure replications.

In this example data is replicated unidirectionally. As a result, only a single replication must be configured.

a) Find the Streams Replication Manager's Replication Configs property.

b) Click the add button and add new lines for each unique replication you want to add and enable.

c) Add and enable your replications. For example:

```
cdppvc->datahub.enabled=true
```

11. Configure Streams Replication Manager Driver and Service role targets.

a) Find and configure the Streams Replication Manager Service Target Cluster property.

Add the co-located cluster's alias to the property. For example:

```
cdppvc
```

Setting this property to cdppvc does not enable you to monitor the replications targeting the Cloudera Data Hub cluster. It is possible to add the Cloudera Data Hub cluster alias to this property and as a result monitor the Cloudera Data Hub cluster. However, this can lead to unwanted behaviour. See the *Before you begin* section for more information.

b) Find and configure the Streams Replication Manager Driver Target Cluster property.

For example:

```
datahub,cdppvc
```

Note: This property must either contain all aliases or left blank. Leaving the property blank has the same effect as adding all aliases.

12. Configure the srm-control tool.

a) Click Gateway in the **Filters** pane.

b) Find and configure the following properties.

- SRM Client's Secure Storage Password: *[***PASSWORD***]*
- Environment Variable Holding SRM Client's Secure Storage Password: SECURESTOREPASS
- Gateway TLS/SSL Trust Store File: *[***CLOUDERA PRIVATE CLOUD BASE GLOBAL TRUSTSTORE LOCATION***]*
- Gateway TLS/SSL Truststore Password: *[***CLOUDERA PRIVATE CLOUD BASE GLOBAL TRUSTSTORE PASSWORD***]*
- SRM Client's Kerberos Principal Name: *[***MY KERBEROS PRINCIPAL****]*
- SRM Client's Kerberos Keytab Location: *[***PATH TO KEYTAB FILE***]*

Take note of the password you configure in SRM Client's Secure Storage Password and the name you configure in Environment Variable Holding SRM Client's Secure Storage Password. You will need to provide both of these in your CLI session before running the tool.

c) Click Save Changes.

d) Restart the Streams Replication Manager service.

e) Deploy client configuration for Streams Replication Manager.

13. Start the replication process using the srm-control tool.
    a) SSH as an administrator to any of the Streams Replication Manager hosts in the Cloudera Private Cloud Base cluster.

    ```
    ssh [***USER***]@[***MY-CLOUDERA-PRIVATE-CLOUD-BASE-CLUSTER.COM***]
    ```

    b) Set the secure storage password as an environment variable.

    ```
    export [***SECURE STORAGE ENV VAR***]="[***SECURE STORAGE PASSWORD***]"
    ```

    Replace *[***SECURE STORAGE ENV VAR***]* with the name of the environment variable you specified in Environment Variable Holding SRM Client's Secure Storage Password. Replace *[***SECURE STORAGE PASSWORD***]* with the password you specified in SRM Client's Secure Storage Password. For example:

    ```
    export SECURESTOREPASS="mypassword"
    ```

    c) Use the srm-control tool with the topics subcommand to add topics to the allow list.

    ```
    srm-control topics --source cdppvc --target datahub --add [***TOPIC
     NAME***]
    ```

    d) Use the srm-control tool with the groups subcommand to add groups to the allow list.

    ```
    srm-control groups --source cdppvc --target datahub --add ".*"
    ```

14. Configure replication monitoring of the Cloudera Data Hub cluster.
    a) Access the Cloudera Manager instance of your Cloudera Private Cloud Base cluster.
    b) In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
    c) Go to Configuration.
    d) Find and configure the Streams Replication Manager Service Target Cluster property.

    Replace the alias set in the property with the Cloudera Data Hub cluster's alias. For example:

    ```
    datahub
    ```

    e) Click Save Changes.
    f) Restart the Streams Replication Manager service.
    g) Access the Streams Messaging Manager UI in the Cloudera Private Cloud Base cluster and go to the Cluster Replications page.

    The replications you set up will be visible on this page.

    Note: If the topics or groups you added for replication are newly created, they might not be immediately visible. This is due to how frequently Streams Replication Manager checks for newly created topics and consumer groups. By default, this is set to 10 minutes, but can be configured with the Refresh Topics Interval Seconds and Refresh Groups Interval Seconds Streams Replication Manager properties. If at first your topics do not appear, wait a few minutes and refresh the page.

Replicating data from on premises to cloud with Streams Replication Manager in the cloud

You can set up and configure an instance of Streams Replication Manager running in a Cloudera Data Hub cluster to replicate data between the Cloudera Data Hub cluster and a Cloudera Private Cloud Base Base cluster. In addition, you can use Streams Messaging Manager to monitor the replication process.

About this task

Consider the following replication scenario.

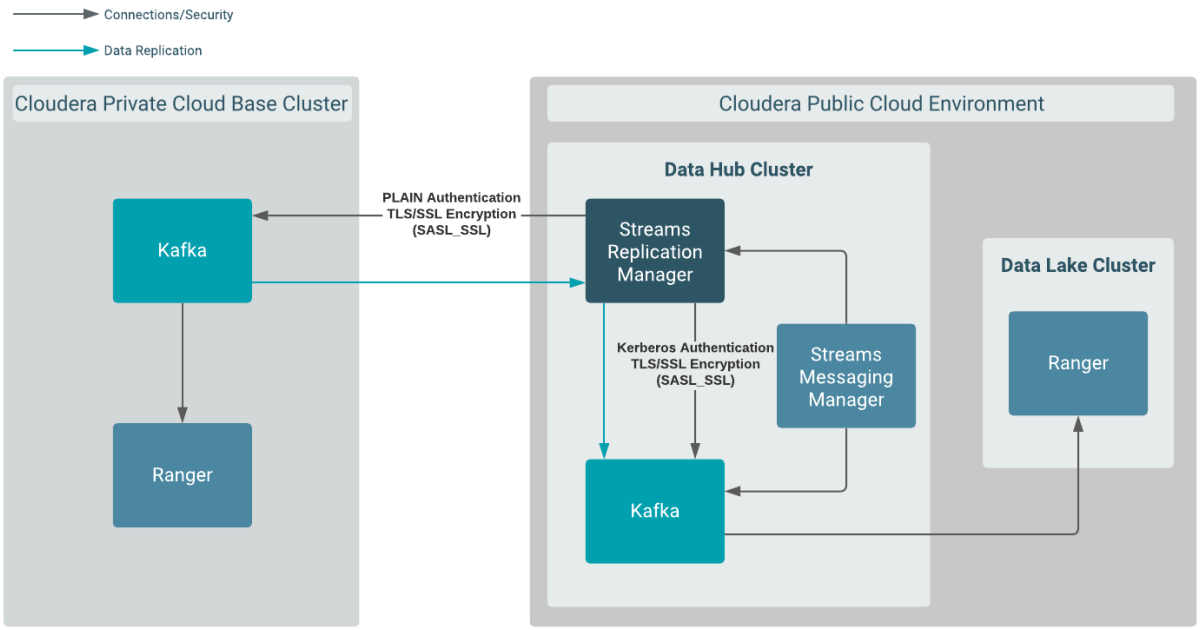

In this scenario, data is replicated from a Cloudera Private Cloud Base cluster to a Cloudera Data Hub cluster by a Streams Replication Manager instance that is deployed in the Cloudera Data Hub cluster.

The Cloudera Private Cloud Base cluster has Kafka deployed on it. It is a secure cluster that has TLS/SSL encryption enabled and uses PLAIN authentication. In addition, it uses Ranger for authorization.

The Cloudera Data Hub cluster is provisioned with the one of the default Streams Messaging cluster definitions.

Before you begin

This example scenario does not go into detail on how to set up the clusters and assumes the following.

- A Cloudera Data Hub provisioned with the Streams Messaging Light Duty or Heavy Duty cluster definition is available.

  For more information, see Setting up your Streams Messaging cluster in the CDF for Data Hub library. Alternatively, you can also review the cloud provider specific cluster creation instructions available in the Cloudera Data Hub library.
- A Cloudera Private Cloud Base cluster with Kafka is available. This cluster has TLS/SSL encryption enabled, uses PLAIN authentication, and has Ranger for authorization. For more information, see the CDP Private Cloud Base Installation Guide.
- Network connectivity and DNS resolution are established between the clusters.

Procedure

1. Obtain PLAIN credentials for Streams Replication Manager.

   The credentials of a PLAIN user that can access the Cloudera Private Cloud Base cluster are required. These credentials are supplied to Streams Replication Manager in a later step. In this example *[***PLAIN USER***]* and *[***PLAIN USER PASSWORD***]* is used to refer to these credentials.

   Note: Typically, Kerberos cannot be easily configured to span across an on-premise and a public cloud environment. Therefore, these instructions assume that these environments use authentication methods that are easy to interoperate in such hybrid environments like LDAP or PAM authentication. For more information on how you can configure your Kafka service to use LDAP or PAM, see *Kafka Authentication*.

2. Add Ranger permissions for the PLAIN user in the Cloudera Private Cloud Base cluster.

   You must ensure that the PLAIN user you obtained has correct permissions assigned to it in Ranger. Otherwise, Streams Replication Manager will not be able to access Kafka resources on the Cloudera Private Cloud Base cluster.

   a) Access the Cloudera Manager instance of your Cloudera Private Cloud Base cluster.
   b) Go to Ranger Ranger Admin Web UI .
   c) Log in to the Ranger Console (Ranger Admin Web UI).
   d) Add the *[***PLAIN USER***]* to the following policies.
      - All - consumergroup
      - All - topic
      - All - transactionalid
      - All - cluster
      - All - delegationtoken

3. Acquire the Cloudera Private Cloud Base cluster truststore and add it to the Cloudera Data Hub cluster.

   The actions you need to take differ depending on how TLS is set up in the Cloudera Private Cloud Base cluster.

   For Auto TLS

   a. Obtain the certificate of the Cloudera Manager root Certificate Authority and its password.

   The Certificate Authority certificate and its password can be obtained using the Cloudera Manager API. The following steps describe how you can retrieve the certificate and password using the Cloudera Manager API Explorer. Alternatively, you can also retrieve the certificate and password by calling the appropriate endpoints in your browser window or using curl.

   1. Access the Cloudera Manager instance of your Cloudera Private Cloud Base cluster.
   2. Go to Support API Explorer .
   3. Find CertManagerResource.
   4. Select the /certs/truststore GET operation and click Try it out.
   5. Enter the truststore type.
   6. Click Execute.
   7. Click Download file under Responses.

      The downloaded file is your certificate.
   8. Select the /certs/truststorePassword GET operation and click Try it out.
   9. Click Execute.

      The password is displayed under Responses.

   b. Run the following command to create the truststore:

```
keytool \
  -importcert \
  -storetype JKS \
  -noprompt \
```

```
-keystore cdppvc-truststore.jks \
-storepass ***PASSWORD*** \
-alias cdppvc-cm-ca \
-file ***PATH TO CM CA CERTIFICATE***
```

Note down the password, it is needed in a later step.

c. Copy the cdpdc-truststore.jks file to a common location on all the hosts in your Cloudera Data Hub cluster.

Cloudera recommends that you use the following location: /opt/cloudera/security/cdppvc-truststore.jks.

d. Set the correct file permissions.

Use 751 for the directory and 444 for the truststore file.

For Manual TLS

a. Note down the truststore location and password of the Cloudera Private Cloud Base cluster. These should be known to you.

b. Copy the truststore file to a common location on all the hosts in your Cloudera Data Hub cluster.

Cloudera recommends that you use the following location: /opt/cloudera/security/truststore.jks.

c. Set the correct file permissions.

Use 751 for the directory and 444 for the truststore file.

4. Access the Cloudera Manager instance of your Cloudera Data Hub cluster.

5. Define the external Kafka cluster (the Cloudera Private Cloud Base cluster).

a) Go to Administration External Accounts .

b) Go to the Kafka Credentials tab.

On this tab you will create a credential for each external cluster taking part in the replication process.

c) Click Add Kafka credentials

d) Configure the Kafka credentials.

In the case of this example, you must create a single credential representing the Cloudera Private Cloud Base cluster. For example:

```
Name=cdppvc
Bootstrap servers=[***MY-CLOUDARE-PRIVATE-CLOUD-BASE-CLUSTER-
HOST-1.COM:9093***],[***MY-CLOUDERA-PRIVATE-CLOUD-BASE-CLUSTER-
HOST-2:9093***]
Security Protocol=SASL_SSL
JAAS Secret 1=[***PLAIN USER***]
JAAS Secret 2=[***PLAIN USER PASSWORD***]
JAAS Template=org.apache.kafka.common.security.plain.PlainLoginModule r
equired username="##JAAS_SECRET_1##" password="##JAAS_SECRET_2##";
SASL Mechanism=PLAIN
Truststore Password=[***PASSWORD***]
Truststore Path=/OPT/CLOUDERA/SECURITY/CDPPVC-TRUSTSTORE.JKS
Truststore type=JKS
```

Note: The properties you specify for the Kafka credential depend on the security configuration of the Cloudera Private Cloud Base cluster. This specific example is for a cluster that has TLS/SSL encryption and PLAIN authentication enabled. You must change these configurations based on the setup of your Cloudera Private Cloud Base cluster.

e) Click Add.

If credential creation is successful, a new entry corresponding to the Kafka credential you specified appears on the page.

6. Define the co-located Kafka cluster (the Cloudera Data Hub cluster).

   Note: Some of the following properties might already be configured by automation.

   a) In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
   b) Go to Configuration.
   c) Find and enable the Kafka Service property.
   d) Find and configure the Streams Replication Manager Co-located Kafka Cluster Alias property.

      The alias you configure represents the co-located cluster. Enter an alias that is unique and easily identifiable. For example:

      ```
      datahub
      ```

   e) Enable relevant security feature toggles.

      Because the Cloudera Data Hub cluster is both TLS/SSL and Kerberos enabled, you must enable all feature toggles for both the Streams Replication Manager Driver and Service roles. The feature toggles are the following.

      - Enable TLS/SSL for SRM Driver
      - Enable TLS/SSL for SRM Service
      - Enable Kerberos Authentication

7. Add both clusters to the configuration of Streams Replication Manager.

   a) Find and configure the External Kafka Accounts property.

      Add the name of all Kafka credentials you created to this property. This can be done by clicking the add button to add a new line to the property and then entering the name of the Kafka credential. For example:

      ```
      cdppvc
      ```

   b) Find and configure the Streams Replication Manager Cluster alias property.

      Add all cluster aliases to this property. This includes the aliases present in both the External Kafka Accounts and Streams Replication Manager Co-located Kafka Cluster Alias properties. Delimit the aliases with commas. For example:

      ```
      datahub,cdppvc
      ```

8. Configure replications.

   In this example data is replicated unidirectionally. As a result, only a single replication must be configured.

   a) Find the Streams Replication Manager's Replication Configs property.
   b) Click the add button and add new lines for each unique replication you want to add and enable.
   c) Add and enable your replications. For example:

      ```
      cdppvc->datahub.enabled=true
      ```

9. Configure Streams Replication Manager Driver and Service role targets.
   a) Find and configure the Streams Replication Manager Service Target Cluster property.

      Add the co-located cluster's alias to the property. For example:

      ```
      datahub
      ```

   b) Find and configure the Streams Replication Manager Driver Target Cluster property.

      For example:

      ```
      datahub,cdppvc
      ```

      Important: If you have another Streams Replication Manager instance configured with the same clusters and is targeting the Cloudera Private Cloud Base cluster, the cdppvc alias should not be configured as a target for this instance of Streams Replication Manager.

      Note: This property must either contain all aliases or left blank. Leaving the property blank has the same effect as adding all aliases.

10. Configure the srm-control tool.
    a) Click Gateway in the **Filters** pane.
    b) Find and configure the following properties.
       - SRM Client's Secure Storage Password: *[***PASSWORD***]*
       - Environment Variable Holding SRM Client's Secure Storage Password: SECURESTOREPASS
       - Gateway TLS/SSL Trust Store File: */OPT/CLOUDERA/SECURITY/DATAHUB-TRUSTSTORE.JKS*
       - Gateway TLS/SSL Truststore Password: *[***PASSWORD***]*
       - SRM Client's Kerberos Principal Name: *[***MY KERBEROS PRINCIPAL****]*
       - SRM Client's Kerberos Keytab Location: *[***PATH TO KEYTAB FILE***]*

       Take note of the password you configure in SRM Client's Secure Storage Password and the name you configure in Environment Variable Holding SRM Client's Secure Storage Password. You will need to provide both of these in your CLI session before running the tool.
    c) Click Save Changes.
    d) Restart the Streams Replication Manager service.
    e) Deploy client configuration for Streams Replication Manager.

11. Start the replication process using the srm-control tool.

a) SSH as an administrator to any of the Streams Replication Manager hosts in the Cloudera Data Hub cluster.

```
ssh [***USER***]@[***MY-CLOUDERA-DATA-HUB-CLUSTER.COM***]
```

b) Set the secure storage password as an environment variable.

```
export [***SECURE STORAGE ENV VAR***]="[***SECURE STORAGE PASSWORD***]"
```

Replace *[***SECURE STORAGE ENV VAR***]* with the name of the environment variable you specified in Environment Variable Holding SRM Client's Secure Storage Password. Replace *[***SECURE STORAGE PASSWORD***]* with the password you specified in SRM Client's Secure Storage Password. For example:

```
export SECURESTOREPASS="mypassword"
```

c) Use the srm-control tool with the topics subcommand to add topics to the allow list.

```
srm-control topics --source cdppvc --target datahub --add [***TOPIC
 NAME***]
```

d) Use the srm-control tool with the groups subcommand to add groups to the allow list.

```
srm-control groups --source cdppvc --target datahub --add ".*"
```

12. Monitor the replication process.

Access the Streams Messaging Manager UI in the Cloudera Data Hub cluster, and go to the Cluster Replications page. The replications you set up will be visible on this page.

Note: If the topics or groups you added for replication are newly created, they might not be immediately visible. This is due to how frequently Streams Replication Manager checks for newly created topics and consumer groups. By default, this is set to 10 minutes, but can be configured with the Refresh Topics Interval Seconds and Refresh Groups Interval Seconds Streams Replication Manager properties. If at first your topics do not appear, wait a few minutes and refresh the page.

Related Information

Kafka Authentication

Replicating data between cloud clusters with Streams Replication Manager in the cloud

You can set up and configure an instance of Streams Replication Manager in a Cloudera Data Hub cluster to replicate data between Cloudera Data Hub clusters. In addition, you can use Streams Messaging Manager to monitor the replication process.

About this task

Consider the following replication scenario.

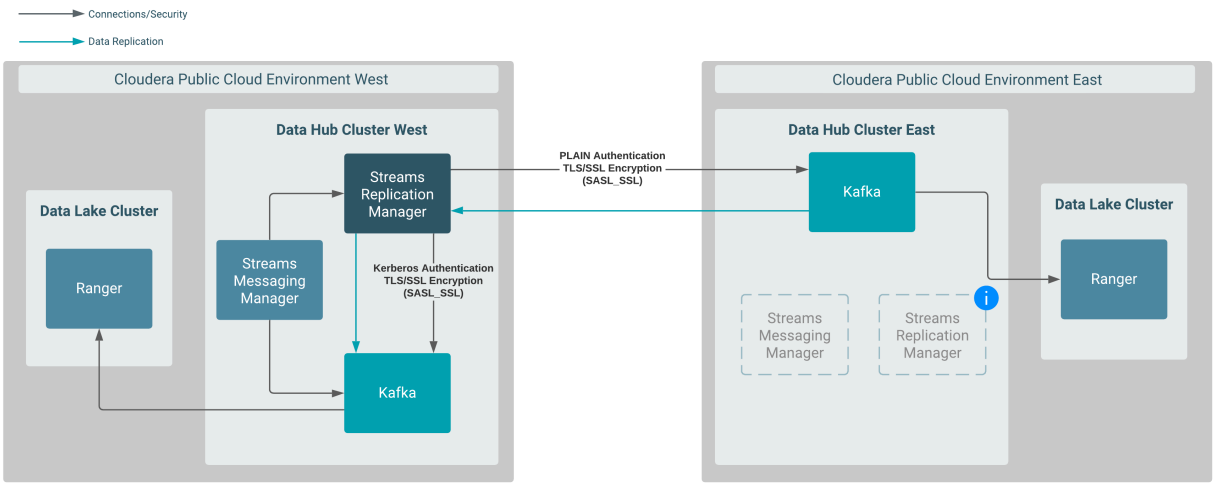

In this scenario, data is replicated between two Cloudera Data Hub clusters that are provisioned in different Cloudera Public Cloud environments. More specifically, data in Cloudera Data Hub East is replicated to Cloudera Data Hub West by an instance of Streams Replication Manager running in Cloudera Data Hub West.

Both Cloudera Data Hub clusters are provisioned with the default Streams Messaging cluster definitions.

Streams Replication Manager and Streams Messaging Manager are available in both clusters, but the instances in Cloudera Data Hub East are not utilized in this scenario.

Before you begin

This example scenario does not go into detail on how to set up the clusters and assumes the following.

- Two Data Hub clusters provisioned with the Streams Messaging Light Duty or Heavy Duty cluster definition are available.

  For more information, see Setting up your Streams Messaging cluster in the CDF for Data Hub library. Alternatively, you can also review the cloud provider specific cluster creation instructions available in the Cloudera Data Hub library.
- Network connectivity and DNS resolution are established between the clusters.

Important: In the following scenario, a new Cloudera machine user is created and set up specifically for Streams Replication Manager. Alternatively, it is also possible to use an existing machine user and skip steps 1 through 3, but this can only be done if the following requirements are met.

- The existing machine user has access to your Cloudera environment.
- The existing machine user has the correct Ranger permissions assigned to it.
- You have access to the existing machine user's credentials.

Procedure

1. Create a machine user for Streams Replication Manager in Cloudera Management Console.

   A machine user is required so that Streams Replication Manager has credentials that it can use to connect to the Kafka service in the Cloudera Data Hub cluster. This step is only required in the environment where

Streams Replication Manager is not running. In the case of this example, this is the Cloudera Public Cloud East environment.

a) Navigate to Management Console User Management.
b) Click Actions Create Machine User .
c) Enter a unique name for the user and click Create.

   For example: srm

   After the user is created, you are presented with a page that displays the user details.

   Note:

   The Workload User Name (srv_srm), is different from the actual machine user name (srm). The Workload User Name is the identifier you use to configure Streams Replication Manager.
d) Click Set Workload Password.
e) Type a password in the Password and Confirm Password fields. Leave the Environment field blank.
f) Click Set Workload Password.
   A message appears on successful password creation.

2. Grant the machine user access to your environment.

You must to grant the machine user access in your environments, otherwise Streams Replication Manager will not be able to connect to the Kafka service with this user. This step is only required in the environments where Streams Replication Manager is not running. In the case of this example this is the Cloudera Public Cloud East environment.

a) Navigate to Management Console Environments , and select the environment where your Kafka cluster is located.
b) Click Actions Manage Access .

   Use the search box to find and select the machine user you want to use.

   A list of **Resource Roles** appears.
c) Select the EnvironmentUser role and click Update Roles.
d) Go back to the **Environment Details** page and click Actions Synchronize Users to FreeIPA .
e) On the **Synchronize Users** page, click Synchronize Users.

   Synchronizing users ensures that the role assignment is in effect for the environment.

   Important: Wait until this process is completed. Otherwise, you will not be able to continue with the next step.

3. Add Ranger permissions for the user you created for Streams Replication Manager.

This step is only required in the environment where Streams Replication Manager is not running. In the case of this example, this is the Cloudera Public Cloud East environment.

a) Navigate to Management Console Environments , and select the environment where your Kafka cluster is located.
b) Click the Ranger link on the **Environment Details** page.
c) Select the resource-based service corresponding to the Kafka resource in the Cloudera Data Hub cluster.
d) Add the Workload User Name of the user you created for Streams Replication Manager to the following Ranger policies.

   - All - consumergroup
   - All - topic
   - All - transactionalid
   - All - cluster
   - All - delegationtoken

4. Establish trust between the clusters.

A truststore is needed so that the Streams Replication Manager instance running in Cloudera Data Hub West can trust Cloudera Data Hub East. To do this, you extract the FreeIPA certificate from the Cloudera Public Cloud East

environment, create a truststore that includes the certificate, and copy the truststore to all hosts on Cloudera Data Hub West.

a) Navigate to Management Console Environments , and select Cloudera Public Cloud East.
b) Go to the FreeIPA tab.
c) Click Get FreeIPA Certificate.
The FreeIPA certificate file, *[***ENVIRONMENT NAME***]*.crt, is downloaded to your computer.
d) Run the following command to create the truststore.

```
keytool \
  -importcert \
  -storetype JKS \
  -noprompt \
  -keystore truststore-east.jks \
  -storepass [***PASSWORD***] \
  -alias freeipa-east-ca \
  -file [***PATH TO FREEIPA CERTIFICATE***]
```

e) Copy the truststore-east.jks file to a common location on all the hosts in your Cloudera Data Hub West cluster.
Cloudera recommends that you use the following location: /opt/cloudera/security/truststore-east.jks.
f) Set the correct file permissions.
Use 751 for the directory and 444 for the truststore file.

5. Access the Cloudera Manager instance of the Cloudera Data Hub West cluster.

6. Define the external Kafka cluster (the Cloudera Data Hub East cluster).

a) Go to Administration External Accounts .
b) Go to the Kafka Credentials tab.
On this tab you will create a credential for each external cluster taking part in the replication process.
c) Click Add Kafka credentials
d) Configure the Kafka credentials.
In the case of this example, you must create a single credential representing the Cloudera Data Hub East cluster. For example:

```
Name=dheast
Bootstrap servers=[***MY-CLOUDERA-DATA-HUB-EAST-CLUSTER-
HOST-1.COM:9093***],[***MY-CLOUDERA-DATA-HUB-EAST-CLUSTER-
HOST-2:9093***]
Security Protocol=SASL_SSL
JAAS Secret 1=[***WORKLOAD USER NAME***]
JAAS Secret 2=[***MACHINE USER PASSWORD***]
JAAS Template=org.apache.kafka.common.security.plain.PlainLoginModule r
equired username="##JAAS_SECRET_1##" password="##JAAS_SECRET_2##";
SASL Mechanism=PLAIN
Truststore Password=[***PASSWORD***]
Truststore Path=/OPT/CLOUDERA/SECURITY/TRUSTSTORE-EAST.JKS
Truststore type=JKS
```

e) Click Add.
If credential creation is successful, a new entry corresponding to the Kafka credential you specified appears on the page.

7. Define the co-located Kafka cluster (the Cloudera Data Hub West cluster).

   Note: Some of the following properties might already be configured by automation.

   a) In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
   b) Go to Configuration.
   c) Find and enable the Kafka Service property.
   d) Find and configure the Streams Replication Manager Co-located Kafka Cluster Alias property.

      The alias you configure represents the co-located cluster. Enter an alias that is unique and easily identifiable. For example:

      ```
      dhwest
      ```

   e) Enable relevant security feature toggles.

      Because the Cloudera Data Hub cluster is both TLS/SSL and Kerberos enabled, you must enable all feature toggles for both the Streams Replication Manager Driver and Service roles. The feature toggles are the following.

      - Enable TLS/SSL for SRM Driver
      - Enable TLS/SSL for SRM Service
      - Enable Kerberos Authentication

8. Add both clusters to the configuration of Streams Replication Manager.

   a) Find and configure the External Kafka Accounts property.

      Add the name of all Kafka credentials you created to this property. This can be done by clicking the add button to add a new line to the property and then entering the name of the Kafka credential. For example:

      ```
      dheast
      ```

   b) Find and configure the Streams Replication Manager Cluster alias property.

      Add all cluster aliases to this property. This includes the aliases present in both the External Kafka Accounts and Streams Replication Manager Co-located Kafka Cluster Alias properties. Delimit the aliases with commas. For example:

      ```
      dheast,dhwest
      ```

9. Configure replications.

   In this example data is replicated unidirectionally. As a result, only a single replication must be configured.

   a) Find the Streams Replication Manager's Replication Configs property.
   b) Click the add button and add new lines for each unique replication you want to add and enable.
   c) Add and enable your replications. For example:

      ```
      dheast->dhwest.enabled=true
      ```

10. Configure Streams Replication Manager Driver and Service role targets.

a) Find and configure the Streams Replication Manager Service Target Cluster property.

Add the co-located cluster's alias to the property. For example:

```
dhwest
```

b) Find and configure the Streams Replication Manager Driver Target Cluster property.

For example:

```
dheast,dhwest
```

Important: If you have another Streams Replication Manager instance configured with the same clusters and is targeting Cloudera Data Hub East, the dheast alias should not be configured as a target for this instance of Streams Replication Manager.

Note: This property must either contain all aliases or left blank. Leaving the property blank has the same effect as adding all aliases.

11. Configure the srm-control tool.

a) Click Gateway in the **Filters** pane.

b) Find and configure the following properties.

Note: Some of the following properties might already be configured by automation.

- SRM Client's Secure Storage Password: *[***PASSWORD***]*
- Environment Variable Holding SRM Client's Secure Storage Password: SECURESTOREPASS
- Gateway TLS/SSL Trust Store File: */OPT/CLOUDERA/SECURITY/TRUSTSTORE-WEST.JKS*
- Gateway TLS/SSL Truststore Password: *[***PASSWORD***]*
- SRM Client's Kerberos Principal Name: *[***MY KERBEROS PRINCIPAL****]*
- SRM Client's Kerberos Keytab Location: *[***PATH TO KEYTAB FILE***]*

Take note of the password you configure in SRM Client's Secure Storage Password and the name you configure in Environment Variable Holding SRM Client's Secure Storage Password. You will need to provide both of these in your CLI session before running the tool.

c) Click Save Changes.

d) Restart the Streams Replication Manager service.

e) Deploy client configuration for Streams Replication Manager.

12. Start the replication process using the srm-control tool.

a) SSH as an administrator to any of the Streams Replication Manager hosts in the Cloudera Data Hub West cluster.

```
ssh [***USER***]@[***MY-CLOUDERA-DATA-HUB-WEST-CLUSTER-HOST-1.COM***]
```

b) Set the secure storage password as an environment variable.

```
export [***SECURE STORAGE ENV VAR***]="[***SECURE STORAGE PASSWORD***]"
```

Replace *[***SECURE STORAGE ENV VAR***]* with the name of the environment variable you specified in Environment Variable Holding SRM Client's Secure Storage Password. Replace *[***SECURE STORAGE PASSWORD***]* with the password you specified in SRM Client's Secure Storage Password. For example:

```
export SECURESTOREPASS="mypassword"
```

c) Use the srm-control tool with the topics subcommand to add topics to the allow list.

```
srm-control topics --source dheast --target dhwest --add [***TOPIC
 NAME***]
```

d) Use the srm-control tool with the groups subcommand to add groups to the allow list.

```
srm-control groups --source dheast --target dhwest --add ".*"
```

13. Monitor the replication process.

Access the Streams Messaging Manager UI in the Cloudera Data Hub West cluster and go to the Cluster Replications page. The replications you set up will be visible on this page.

Note: If the topics or groups you added for replication are newly created, they might not be immediately visible. This is due to how frequently Streams Replication Manager checks for newly created topics and consumer groups. By default, this is set to 10 minutes, but can be configured with the Refresh Topics Interval Seconds and Refresh Groups Interval Seconds Streams Replication Manager properties. If at first your topics do not appear, wait a few minutes and refresh the page.

Cloudera Runtime 7.3.1

Configuring Streams Replication Manager

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enable high availability for Streams Replication Manager..... 5

Enabling prefixless replication..... 5

Defining and adding clusters for replication..... 6

Defining external Kafka clusters.....8
Defining co-located Kafka clusters using a service dependency.....12
Defining co-located Kafka clusters using Kafka credentials.....13
Adding clusters to Streams Replication Manager's configuration..... 17

Configuring replications.....18

Configuring the driver role target clusters.....18

Configuring the service role target cluster..... 19

Configuring properties not exposed in Cloudera Manager.....20

Configuring replication specific Kafka Connect REST servers.....21

Configuring Remote Querying..... 22

Enabling Remote Querying..... 22
Configuring the advertised information of the Streams Replication Manager Service role..... 24

Configuring Streams Replication Manager Driver retry behaviour..... 24

Configuring Streams Replication Manager Driver heartbeat emission..... 25

Configuring automatic group offset synchronization..... 27

Configuring Streams Replication Manager Driver for performance tuning.....28

Hardware..... 28
Task count.....29
Consumer and producer configurations..... 29
Connect worker configurations..... 30

Metric calculation..... 31

New topic and consumer group discovery..... 31

Configuration examples..... 32

Bidirectional replication example of two active clusters..... 32
Cross data center replication example of multiple clusters..... 34

Enable high availability for Streams Replication Manager

Streams Replication Manager is capable of running in high availability mode. This can be enabled by deploying multiple instances of the driver and service role in a cluster.

Streams Replication Manager is capable of running in high availability mode. By enabling high availability mode for Streams Replication Manager, you can ensure that the replication of data and the monitoring of replication continues even in the case of host failure. To enable Streams Replication Manager high availability, you must deploy multiple instances of the driver and service roles on the hosts in a cluster.

In Cloudera Public Cloud this can be done when provisioning a new cluster with Cloudera Data Hub.

Note: Expect an increased load when running Streams Replication Manager in high availability mode.

Enable high availability mode in Cloudera Public Cloud

In Cloudera Public Cloud, high availability mode can be enabled by provisioning a Cloudera Data Hub cluster that has multiple instances of the Streams Replication Manager driver and service. In terms of high availability, the default Streams Messaging cluster definitions behave in the following way:

Streams Messaging Light Duty

High availability mode is enabled by default. Any cluster that you provision with this definition will start Streams Replication Manager in high availability mode. Note that it is only the driver that is started in high availability mode, the service role is only provisioned on a single host. This however, is still considered as a highly available deployment of Streams Replication Manager.

Streams Messaging Heavy Duty

In the heavy duty definition, Streams Replication Manager has its own host group. However, by default, the Streams Replication Manager host group is not provisioned. When provisioning a cluster with the heavy duty definition, you can decide how many nodes this host group should have. To provision Streams Replication Manager in high availability mode with this cluster definition, you have to set the instance count of the SRM nodes host group to at least 2.

For more information on creating a Streams Messaging cluster with Cloudera Data Hub, see Creating your first Streams Messaging cluster.

Related Information

Setting up your Streams Messaging cluster

Streams Replication Manager Driver

Task architecture and load-balancing

Enabling prefixless replication

Learn how to enable prefixless replication with the IdentityReplicationPolicy. If this replication policy is in use, remote topics on target clusters retain their original name and are not prefixed.

About this task

The default replication policy used by Streams Replication Manager is the `DefaultReplicationPolicy`. This replication policy renames remote (replicated) topics on target clusters and adds the source cluster alias as a prefix. For example, useast.topic1. If you do not want your topics to get renamed, you can choose to use `IdentityReplicationPolicy` instead, which does not rename topics during replication. This type of replication is referred to as prefixless replication.

You can configure Streams Replication Manager to use the `IdentityReplicationPolicy` by selecting the Enable Prefixless Replication Streams Replication Manager service property in Cloudera Manager.

Before you begin

- Ensure that the Remote Topics Discovery With Internal Topic property remains selected after you enable prefixless replication. This property must be selected for prefixless replication to function. This property is selected by default and is also automatically selected when you enable prefixless replication with Enable Prefixless Replication.
- Do not use replication.policy.class to set `IdentityReplicationPolicy` as the replication policy. Use Enable Prefixless Replication instead.
- Using a mix of replications policies in a deployment with multiple instances of the Streams Replication Manager service is not supported.
- Transitioning an already running Streams Replication Manager service from the `DefaultReplicationPolicy` or any other replication policy to the `IdentityReplicationPolicy` is not recommended.
- The `IdentityReplicationPolicy` has the following limitations:
  - Replication loop detection is not supported. As a result, you must ensure that topics are **not** replicated in a loop between your source and target clusters.
  - The /v2/topic-metrics/{target}/{downstreamTopic}/{metric} endpoint of Streams Replication Manager Service v2 API does not work properly with prefixless replication. Use the /v2/topic-metrics/{source}/{target}/{upstreamTopic}/{metric} endpoint instead.
  - The replication metric graphs shown on the **Topic Details** page of the Streams Messaging Manager UI do not work with prefixless replication.

  As a result of these limitations, Cloudera recommends that you use the `DefaultReplicationPolicy` whenever possible.

Procedure

1. In Cloudera Manager, select the Streams Replication Manager service.
2. Go to Configuration.
3. Find and select the Enable Prefixless Replication property.
4. Click Save Changes.
5. Restart the service.

Results

Streams Replication Manager is configured to use the `IdentityReplicationPolicy`. Remote topics on target clusters are no longer prefixed with the source cluster alias.

Defining and adding clusters for replication

Before you can start replicating data with Streams Replication Manager, you must define the clusters that take part in the replication process and add them to Streams Replication Manager's configuration. Defining and adding the clusters provides Streams Replication Manager with the necessary connection and security information it needs to replicate data.

In order for Streams Replication Manager to replicate data between clusters, you have to define these clusters in Cloudera Manager and add them to the Streams Replication Manager service's configuration. In order to correctly configure your clusters, you need to complete multiple configuration tasks. The configuration workflow is the following:

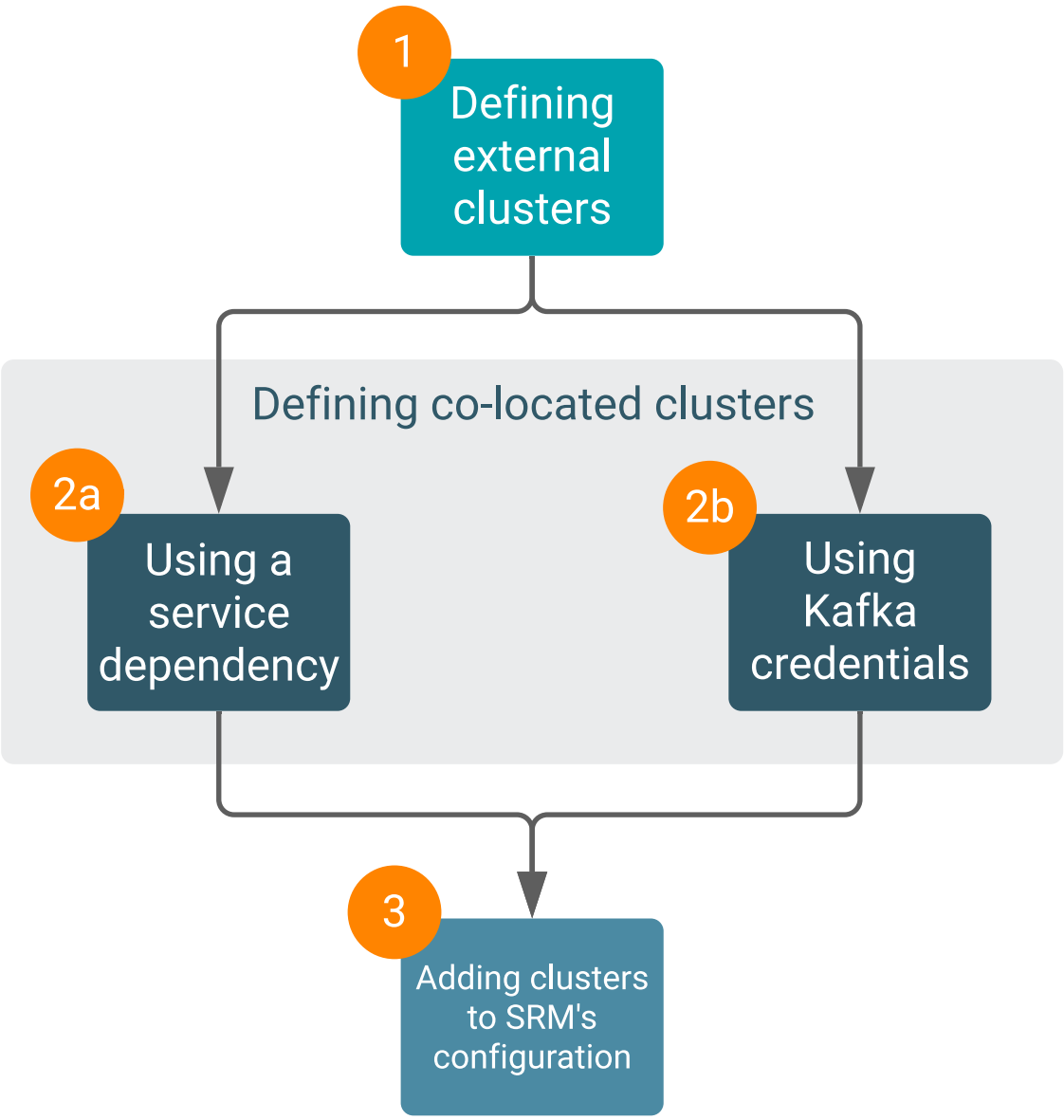

1 Defining external clusters

In any given scenario, you start with defining your external clusters. External clusters are defined by creating and configuring Kafka credentials. A Kafka credential is an item that contains the connection properties required by Streams Replication Manager to establish a connection with a cluster. You can think of a Kafka credential as the definition of a single cluster. It contains the name (alias), address (bootstrap servers), and credentials that Streams Replication Manager can use to access a specific cluster.

Kafka credentials are created and configured in Cloudera Manager on the Administration External Accounts Kafka Credentials page. How you configure each Kafka credential depends on the security configuration of the cluster that it defines. Once a Kafka credential is created, the properties needed to access that cluster become available to Streams Replication Manager.

2 Defining co-located clusters

Co-located Kafka clusters can be defined in either of two ways. You can define these clusters using a service dependency or by using Kafka credentials. Cloudera recommends that you follow the

recommendations provided and choose the method based on the security configuration of the co-located Kafka cluster. Additionally, Cloudera recommends that you use the service dependency method whenever possible. The method you choose also impacts how you configure the srm-control tool.

Note: It is possible that your deployment does not have any co-located clusters, or if there are multiple instances of Streams Replication Manager, only some have a co-located cluster. If there is no co-located cluster, you do not need to define it.

2a Defining co-located clusters using a service dependency

The service dependency method works by enabling a service dependency between the Streams Replication Manager service and the co-located Kafka cluster, specifying an alias for the co-located cluster, and configuring security related properties.

Cloudera recommends that you use this method if:

- The co-located Kafka cluster is not secured.
- The co-located Kafka cluster uses Kerberos for authentication.

2b Defining co-located clusters using Kafka credentials

Kafka credentials can be used to define co-located clusters. The steps to define a co-located cluster with a Kafka credential are identical to the steps you take when defining external clusters.

Cloudera recommends that you use this method if the co-located Kafka cluster uses an authentication mechanism that is not Kerberos, for example PLAIN.

3 Adding clusters to Streams Replication Manager's configuration

Once both external and co-located clusters are defined, you must add all defined clusters to Streams Replication Manager's configuration. This is done by configuring various configuration properties in Cloudera Manager.

Continue reading for step-by-step instructions on how to complete each of the configuration tasks.

Defining external Kafka clusters

Before you can start replicating data with Streams Replication Manager, you must define the external Kafka clusters that take part in the replication process. This is done using Cloudera Manager by creating Kafka credentials.

About this task

The following list of steps walk you through how you can define the external Kafka clusters that Streams Replication Manager connects to.

Before you begin

- Ensure that you have reviewed Defining and adding clusters for replication and understand that the following list of steps are only one part of the full configuration workflow.
- When creating and configuring Kafka Credentials you will be presented with various configuration properties. A comprehensive list of these properties is provided in Kafka credentials property reference. Cloudera recommends having this resource open during configuration.

Procedure

1. In Cloudera Manager, go to Administration External Accounts .
2. Go to the Kafka Credentials tab.

   The Kafka Credentials tab enables you to create, configure, and manage Kafka credentials. On this tab you will create a credential for each external cluster taking part in the replication process.

3. Create and configure Kafka credentials for the external clusters:

   Repeat this step for all external clusters.

   a) Click Add Kafka credentials.
   b) Configure the Kafka credentials:

      The security configuration of the external cluster determines which of the available properties you must set. To see which exact properties are required for your scenario, click one of the following tabs matching the security protocol of the cluster you are defining:

For PLAINTEXT

- Name
- Bootstrap servers
- Security protocol

For SSL

Encryption only

- Name
- Bootstrap servers
- Security protocol
- Truststore Password
- Truststore Path
- Truststore Type

Encryption and authentication

- Name
- Bootstrap servers
- Security protocol
- Truststore Password
- Truststore Path
- Truststore Type
- Key Password
- Keystore Password
- Keystore Path
- Keystore Type

For SASL_PLAINTEXT

- Name
- Bootstrap servers
- Security protocol
- JAAS Secret [1-3]
- JAAS Template
- Kerberos Service Name
- SASL Mechanism

For SASL_SSL

- Name
- Bootstrap servers
- Security protocol
- Truststore Password
- Truststore Path
- Truststore Type

- JAAS Secret [1-3]
- JAAS Template
- Kerberos Service Name
- SASL Mechanism

Note: Click the i icon next to each property in Cloudera Manager to reveal additional information about each property.

c) Click Add.

If credential creation is successful, a new entry corresponding to the Kafka credential you specified appears on the page.

The following tabs collect examples of different Kafka credentials as well as some notes regarding configuration. Review these examples to better understand how to correctly configure a Kafka credential.

For PLAINTEXT

```
name=uswest
Bootstrap servers=uswest-node1.cluster.com:9092,uswest-node2.clu
ster.com:9092,uswest-node3.cluster.com:9092
Security protocol=PLAINTEXT
```

For SSL

Encryption only

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9093,uswest-node2.clu
ster.com:9093,uswest-node3.cluster.com:9093
Security protocol=SSL
Truststore Password=password
Truststore Path=/opt/srm/us-west-truststore.jks
Truststore Type=JKS
```

Encryption and authentication

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9093,uswest-node2.clu
ster.com:9093,uswest-node3.cluster.com:9093
Security protocol=SSL
Truststore Password=password
Truststore Path=/opt/srm/us-west-truststore.jks
Truststore Type=JKS
Keystore Password=password
Keystore Path=/opt/srm/us-west-keystore.jks
Keystore Type=JKS
Key Password=password
```

Note: The truststore and keystore files must be deployed on all Streams Replication Manager hosts and Streams Replication Manager must be able to access these files.

For SASL_PLAINTEXT

Kerberos

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9092,uswest-node2.clu
ster.com:9092,uswest-node3.cluster.com:9092
Security protocol=SASL_PLAINTEXT
```

```
JAAS Template=com.sun.security.auth.module.Krb5LoginModule requ
ired useKeyTab=true keyTab="/opt/srm/streamsrepmgr.keytab" princ
ipal="streamsrepmgr@REALM.COM";
Kerberos Service Name=kafka
SASL Mechanism=GSSAPI
```

PLAIN

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9092,uswest-node2.clu
ster.com:9092,uswest-node3.cluster.com:9092
Security protocol=SASL_PLAINTEXT
JAAS Secret 1=password
JAAS Template=org.apache.kafka.common.security.plain.PlainLoginM
odule required username="username" password="##JAAS_SECRET_1##";
SASL Mechanism=PLAIN
```

Note: The Kerberos principal must not include host names. Additionally, the keytab file must be deployed on all Streams Replication Manager hosts and Streams Replication Manager must be able to access this file.

For SASL_SSL

Kerberos

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9093,uswest-node2.clu
ster.com:9093,uswest-node3.cluster.com:9093
Security protocol=SASL_SSL
Truststore Password=password
Truststore Path=/opt/srm/us-west-truststore.jks
Truststore Type=JKS
JAAS Template=com.sun.security.auth.module.Krb5LoginModule requi
red useKeyTab=true keyTab="/opt/srm/streamsrepmgr.keytab" princi
pal="streamsrepmgr@REALM.COM";
Kerberos Service Name=kafka
SASL Mechanism=GSSAPI
```

PLAIN

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9093,uswest-node2.clu
ster.com:9093,uswest-node3.cluster.com:9093
Security protocol=SASL_SSL
Truststore Password=password
Truststore Path=/opt/srm/us-west-truststore.jks
Truststore Type=JKS
JAAS Secret 1=password
JAAS Template=org.apache.kafka.common.security.plain.PlainLoginMo
dule required username="username" password="##JAAS_SECRET_1##";
SASL Mechanism=PLAIN
```

Note: The Kerberos principal must not include host names. Additionally, the keytab and truststore files must be deployed on all Streams Replication Manager hosts and Streams Replication Manager must be able to access these files.

Results

The external Kafka clusters are defined using Kafka credentials.

What to do next

Define the co-located Kafka cluster either with a Kafka credential or through a service dependency.

Related Information

Defining co-located Kafka clusters using a service dependency

Defining co-located Kafka clusters using Kafka credentials

Defining co-located Kafka clusters using a service dependency

Before you can start replicating data with Streams Replication Manager, you must define the co-located Kafka clusters that take part in the replication process. This can be done by enabling a service dependency between the Streams Replication Manager service and the co-located Kafka cluster, specifying an alias for the co-located cluster, and configuring security related properties.

About this task

The following list of steps walk you through how you can define the co-located Kafka cluster using a service dependency.

Before you begin

Ensure that you have reviewed Defining and adding clusters for replication and understand that the following list of steps are only one part of the full configuration workflow.

Procedure

1. In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
2. Go to Configuration.
3. Configure the co-located cluster:
   a) Find and enable the Kafka Service property.
   b) Find and configure the Streams Replication Manager Co-located Kafka Cluster Alias property.

      The alias you configure represents the co-located cluster. Enter an alias that is unique and easily identifiable. For example:

      ```
      uswest
      ```

   c) If the co-located cluster uses any type of security, enable the relevant security feature toggles.
      - If the cluster uses TLS/SSL, enable the following properties:
        - Enable TLS/SSL for SRM Driver
        - Enable TLS/SSL for SRM Service
      - If the cluster uses Kerberos authentication, enable the Enable Kerberos Authentication property.
      - If the cluster uses TLS/SSL and Kerberos, enable all three properties.

Results

The co-located Kafka cluster is defined using a service dependency.

What to do next

Add both external and co-located Kafka clusters to Streams Replication Manager's configuration.

Related Information

Adding clusters to Streams Replication Manager's configuration

Defining co-located Kafka clusters using Kafka credentials

Before you can start replicating data with Streams Replication Manager, you must define the co-located Kafka clusters that take part in the replication process. This can be done using Cloudera Manager by creating Kafka credentials.

About this task

The following list of steps walk you through how you can define the co-located Kafka cluster using Kafka credentials.

Before you begin

- Ensure that you have reviewed Defining and adding clusters for replication and understand that the following list of steps are only one part of the full configuration workflow.

Procedure

1. In Cloudera Manager, go to Administration External Accounts .
2. Go to the Kafka Credentials tab.

   The Kafka Credentials tab enables you to create, configure, and manage Kafka credentials. On this tab you will create a credential for your co-located Kafka cluster.

3. Create and configure Kafka credentials for the external clusters:
   a) Click Add Kafka credentials.
   b) Configure the Kafka credential:

      The security configuration of the co-located cluster determines which of the available properties you must set. To see which exact properties are required for your scenario, click one of the following tabs matching the security protocol of the cluster you are defining:

For PLAINTEXT

- Name
- Bootstrap servers
- Security protocol

For SSL

Encryption only

- Name
- Bootstrap servers
- Security protocol
- Truststore Password
- Truststore Path
- Truststore Type

Encryption and authentication

- Name
- Bootstrap servers
- Security protocol
- Truststore Password
- Truststore Path
- Truststore Type
- Key Password
- Keystore Password
- Keystore Path
- Keystore Type

For SASL_PLAINTEXT

- Name
- Bootstrap servers
- Security protocol
- JAAS Secret [1-3]
- JAAS Template
- Kerberos Service Name
- SASL Mechanism

For SASL_SSL

- Name
- Bootstrap servers
- Security protocol
- Truststore Password
- Truststore Path
- Truststore Type
- JAAS Secret [1-3]
- JAAS Template

- Kerberos Service Name
- SASL Mechanism

Note: Click the i icon next to each property in Cloudera Manager to reveal additional information about each property.

c) Click Add.

If credential creation is successful, a new entry corresponding to the Kafka credential you specified appears on the page.

The following tabs collect examples of different Kafka credentials as well as some notes regarding configuration. Review these examples to better understand how to correctly configure a Kafka credential.

For PLAINTEXT

```
name=uswest
Bootstrap servers=uswest-node1.cluster.com:9092,uswest-node2.clu
ster.com:9092,uswest-node3.cluster.com:9092
Security protocol=PLAINTEXT
```

For SSL

Encryption only

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9093,uswest-node2.clu
ster.com:9093,uswest-node3.cluster.com:9093
Security protocol=SSL
Truststore Password=password
Truststore Path=/opt/srm/us-west-truststore.jks
Truststore Type=JKS
```

Encryption and authentication

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9093,uswest-node2.clu
ster.com:9093,uswest-node3.cluster.com:9093
Security protocol=SSL
Truststore Password=password
Truststore Path=/opt/srm/us-west-truststore.jks
Truststore Type=JKS
Keystore Password=password
Keystore Path=/opt/srm/us-west-keystore.jks
Keystore Type=JKS
Key Password=password
```

Note: The truststore and keystore files must be deployed on all Streams Replication Manager hosts and Streams Replication Manager must be able to access these files.

For SASL_PLAINTEXT

Kerberos

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9092,uswest-node2.clu
ster.com:9092,uswest-node3.cluster.com:9092
Security protocol=SASL_PLAINTEXT
JAAS Template=com.sun.security.auth.module.Krb5LoginModule requ
ired useKeyTab=true keyTab="/opt/srm/streamsrepmgr.keytab" princ
ipal="streamsrepmgr@REALM.COM";
Kerberos Service Name=kafka
```

```
SASL Mechanism=GSSAPI
```

PLAIN

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9092,uswest-node2.clu
ster.com:9092,uswest-node3.cluster.com:9092
Security protocol=SASL_PLAINTEXT
JAAS Secret 1=password
JAAS Template=org.apache.kafka.common.security.plain.PlainLoginM
odule required username="username" password="##JAAS_SECRET_1##";
SASL Mechanism=PLAIN
```

Note: The Kerberos principal must not include host names. Additionally, the keytab file must be deployed on all Streams Replication Manager hosts and Streams Replication Manager must be able to access this file.

For SASL_SSL

Kerberos

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9093,uswest-node2.clu
ster.com:9093,uswest-node3.cluster.com:9093
Security protocol=SASL_SSL
Truststore Password=password
Truststore Path=/opt/srm/us-west-truststore.jks
Truststore Type=JKS
JAAS Template=com.sun.security.auth.module.Krb5LoginModule requi
red useKeyTab=true keyTab="/opt/srm/streamsrepmgr.keytab" princi
pal="streamsrepmgr@REALM.COM";
Kerberos Service Name=kafka
SASL Mechanism=GSSAPI
```

PLAIN

```
Name=uswest
Bootstrap servers=uswest-node1.cluster.com:9093,uswest-node2.clu
ster.com:9093,uswest-node3.cluster.com:9093
Security protocol=SASL_SSL
Truststore Password=password
Truststore Path=/opt/srm/us-west-truststore.jks
Truststore Type=JKS
JAAS Secret 1=password
JAAS Template=org.apache.kafka.common.security.plain.PlainLoginMo
dule required username="username" password="##JAAS_SECRET_1##";
SASL Mechanism=PLAIN
```

Note: The Kerberos principal must not include host names. Additionally, the keytab and truststore files must be deployed on all Streams Replication Manager hosts and Streams Replication Manager must be able to access these files.

Results

The co-located Kafka cluster is defined using Kafka credentials.

What to do next

Add both external and co-located Kafka clusters to Streams Replication Manager's configuration.

Related Information

Adding clusters to Streams Replication Manager's configuration

Adding clusters to Streams Replication Manager's configuration

After both the external and co-located are defined, you must add them to the Streams Replication Manager service's configuration. This is done by configuring various configuration properties in Cloudera Manager.

About this task

Defining the clusters that you want to replicate is not sufficient. You must add the clusters you defined to the Streams Replication Manager service's configuration. This is required because Streams Replication Manager does not automatically pick up the clusters that you define.

Before you begin

Ensure that you have reviewed Defining and adding clusters for replication and understand that the following list of steps are only one part of the full configuration workflow.

Procedure

1. In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
2. Go to Configuration.
3. Add the clusters defined with Kafka credentials to Streams Replication Manager's configuration:
   a) Find and configure the External Kafka Accounts property.
   b) Click the add button and add new lines for each Kafka credential you created.
   c) Add the names of all Kafka credentials.

      Each Kafka credential must be added to a new line. For example:

      ```
      useast
      usnorth
      ```

      Note: The alias representing the co-located cluster should only be included in this list if the co-located cluster was defined with a Kafka credential. If you used the service dependency method to define the co-located cluster, this list should only contain the external clusters.
4. Find and configure the Streams Replication Manager Cluster alias property.

   Add all cluster aliases including:

   - Aliases present in the External Kafka Accounts property
   - If the co-located cluster was defined with a service dependency, include the alias present in the Streams Replication Manager Co-located Kafka Cluster Alias property.

   Delimit the aliases with commas. For example:

   ```
   useast, usnorth, uswest
   ```
5. Click Save Changes.
6. Restart the Streams Replication Manager service.

Results

Both external and co-located Kafka clusters are added to Streams Replication Manager's configuration.

What to do next

Add and configure replications. Alternatively, if replications are already configured, you can continue with configuring the srm-control tool.

Related Information

Configuring replications

Configuring srm-control

Configuring replications

In order to replicate data between clusters, you must configure replications for Streams Replication Manager. Each replication specifies a source->target cluster pair and the direction in which data is replicated. Replications can be configured in Cloudera Manager.

About this task

Configuring replications does not start the replication of data. When you configure your replications, Streams Replication Manager sets up communication with the clusters specified within each replication, but does not automatically replicate any data. To start replicating data, you must specify which topics and groups you want to replicate. This is done using the srm-control command line tool. Unless the tool is used to add topics and consumer groups to the replication allowlists, the replications you configure remain empty (inactive).

Procedure

1. In Cloudera Manager, select Streams Replication Manager.
2. Go to Configuration.
3. Add and enable replications:
   a) Find the Streams Replication Manager's Replication Configs property.
   b) Click the add button and add new lines for each unique replication you want to add and enable.
   c) Add and enable your replications. For example:

   ```
   primary->secondary.enabled=true
   secondary->primary.enabled=true
   ```

4. Enter a Reason for change, and then click Save Changes to commit the changes.
5. Restart Streams Replication Manager.

Results

Replications are set up and configured for the specified cluster pairs.

What to do next

Use the srm-control tool to start replication by adding topics or groups to the allowlist.

Related Information

srm-control

Configuring the driver role target clusters

The Streams Replication Manager Driver role's target clusters are the clusters that the driver is writing data to. You can configure these target clusters for each instance of the driver with the Streams Replication Manager Driver Target Cluster property. Custom configuration of these targets is only recommended in advanced deployments.

About this task

The Streams Replication Manager Driver role is responsible for connecting to the specified clusters and performing replication between them. The driver can be installed on one or more hosts within a cluster.

The clusters the driver connects to are the clusters that you specify with the Streams Replication Manager Cluster alias and Streams Replication Manager's Replication Configs properties.

Target clusters of the driver are clusters that the driver writes data to. By default when the driver is started it will connect to all clusters, gather data from them, and write to all of them. In other words, by default a driver targets all clusters in your configuration. You can limit the number of clusters that each driver targets. This can be done with the Streams Replication Manager Driver Target Cluster property, which allows you to specify which cluster or clusters the driver targets.

When you specify a driver target, the driver still connects to all clusters and gathers data from them, but will only write to the clusters specified.

However, in order for monitoring to function correctly, the driver has to target all clusters taking part in the replication. That is, it has to contain the actual target, the cluster you want to write data to, as well as the source clusters for that target, where data is being pulled from. If the source clusters are not specified, you will not be able to monitor your replications. As a result of this, configuring driver targets and limiting the number of clusters each instance of the driver writes to is considered an advanced configuration practice. This practice is only viable in complex replication scenarios that involve a high number clusters and replications. Therefore, Cloudera recommends that you either leave this property empty or add all clusters taking part in the replication.

By default the Streams Replication Manager Driver Target Cluster property is left empty, meaning that all clusters are targeted. The property accepts any cluster alias that is specified in Streams Replication Manager Cluster alias. When adding multiple cluster aliases, delimit them with a comma.

Procedure

1. In Cloudera Manager, select Streams Replication Manager.
2. Go to Configuration.
3. Find the Streams Replication Manager Driver Target Cluster property.
4. Add the cluster aliases that you want the driver role to target. For example:

   ```
   primary, secondary
   ```

5. Enter a Reason for change, and then click Save Changes to commit the changes.
6. Restart Streams Replication Manager.

Results

Driver targets are configured. Drivers only write data to the targeted clusters.

Configuring the service role target cluster

The Streams Replication Manager Service role's target cluster is the cluster from which metrics are gathered and exposed. Multiple targets can be configured for each instance of the service with the Streams Replication Manager Service Target Cluster property. Configuration is mandatory.

About this task

The Streams Replication Manager Service role consists of a REST API and a Kafka Streams application that aggregates and exposes cluster, topic, and consumer group metrics. With the help of these metrics, users can monitor replications. The service can be installed on one or more hosts per cluster.

Each instance of the service is associated with at least one target cluster. The target is the cluster that the service gathers and exposes metrics from. Each instance of the service can have multiple targets. As a result, it is possible to use a single instance of the Service role to target and monitor multiple clusters.

Service role target clusters are configured with the Streams Replication Manager Service Target Cluster property. The property accepts any cluster alias that is specified in Streams Replication Manager Cluster alias as long as data is being replicated to that cluster. Configuring at least a single service target is mandatory.

Procedure

1. In Cloudera Manager, select Streams Replication Manager.
2. Go to Configuration.
3. Find the Streams Replication Manager Service Target Cluster property.
4. Add your target cluster aliases.

   The number of aliases you add depends on whether you want the Service role to target and monitor a single cluster or multiple clusters.

   If you want to target a single cluster, add a single cluster alias. For example:

   ```
   secondary
   ```

   If you want to target multiple clusters, add multiple aliases separated by commas. For example:

   ```
   primary,secondary,tertiary
   ```
5. Enter a Reason for change, and then click Save Changes to commit the changes.
6. Restart Streams Replication Manager.

Results

The Service role targets are configured. The Service role gathers and exposes metrics from the specified clusters.

Configuring properties not exposed in Cloudera Manager

There are a number of configuration properties that Streams Replication Manager accepts, but are not exposed directly in Cloudera Manager. You can configure these properties with the Streams Replication Manager's Replication Configs property.

About this task

In addition to the configuration properties exposed directly for configuration through Cloudera Manager, Streams Replication Manager accepts number of additional properties including Streams Replication Manager specific properties, Kafka Connect worker properties, as well as Kafka properties available in the version of Kafka that you are using. Properties not exposed directly in Cloudera Manager can be set through the Streams Replication Manager's Replication Configs property.

The following steps demonstrate how these properties can be configured using the Streams Replication Manager's Replication Configs property. If you want to learn more about the properties that you can configure, see the links provided in the *Related information* section at the bottom of this page.

Before you begin

Make sure that cluster aliases and replications are configured. Otherwise cluster or replication level configuration is not possible.

Procedure

1. In Cloudera Manager, select Streams Replication Manager.
2. Go to Configuration.

3. Configure properties not exposed in Cloudera Manager:
   a) Find the Streams Replication Manager's Replication Configs property.
   b) Click the add button and add new lines for each additional property you want to configure.
   c) Add configuration properties. For example:

```
offset.flush.timeout.ms=20000
```

4. Enter a Reason for change, and then click Save Changes to commit the changes.
5. Restart Streams Replication Manager.

Results

Configuration properties not directly exposed in Cloudera Manager are configured.

Related Information

Configuration Properties Reference for Properties not Available in Cloudera Manager

Configuring replication specific Kafka Connect REST servers

Understanding Streams Replication Manager properties, their configuration and hierarchy

Configuring replication specific Kafka Connect REST servers

The Streams Replication Manager Driver role starts a dedicated Kafka Connect REST server for each replication that you set up and configure. These REST servers make communication possible between the different instances of the Streams Replication Manager driver. Communication between the driver instances in turn ensures that replication does not fail when there are multiple driver instances present in a single cluster.

About this task

If required, you can configure each replication's dedicated REST server that is set up by the driver.

REST server properties can be configured through the Streams Replication Manager's Replication Configs property by using two specific prefixes. These prefixes are the following:

*****ALIAS***->***ALIAS***.worker.**

This prefix can be used to set any Kafka Connect REST server property for a replication's dedicated REST server. The first element of the prefix determines which replication's REST server is being configured. For example, the primary->secondary.worker. prefix can be used to configure the primary->secondary replication's REST server. While configuration of all REST server properties is supported, the main use case for this prefix is to set the ports of the REST servers. The reason for this is that by default the REST servers will bind to port 0, that is, any available port. This behaviour may not be suitable for your deployment.

Important: The rest.host.name and rest.port properties should not be used with this prefix. Use the listeners property instead if you want to configure ports.

listeners.https.

This prefix can be used to configure SSL related properties. You can use this prefix to override the SSL settings that the REST server inherits from the service configuration. For example, if the REST server needs to use a different keystore location than the one provided in the service configuration, you can use the following property:

```
listeners.https.ssl.keystore.location=***CUSTOM KEYSTORE LOCATIO
N***
```

Procedure

1. In Cloudera Manager, select the Streams Replication Manager service.
2. Go to Configuration.
3. Find the Streams Replication Manager's Replication Configs property.
4. Click the add button and add new lines for each additional property you want to configure.
5. Add configuration properties. For example:

```
primary->secondary.worker.listeners=HTTPS://myhost:8084
listeners.https.ssl.keystore.location=***CUSTOM KEYSTORE LOCATION***
```

6. Enter a Reason for change, and then click Save Changes to commit the changes.
7. Restart Streams Replication Manager.

Results

Custom configuration of the Kafka Connect REST server is complete.

Configuring Remote Querying

Remote Querying in Streams Replication Manager refers to the Streams Replication Manager Service's capability of querying other, remote Streams Replication Manager Services to fetch the remote cluster replication metrics. This allows users to monitor all replications of a deployment that has multiple instances of Streams Replication Manager through a single Streams Replication Manager Service.

Related Information

Remote Querying

Enabling Remote Querying

Remote Querying can be enabled for a cluster of Streams Replication Manager Service roles (Streams Replication Manager Service cluster). When this feature is enabled, the configured Streams Replication Manager Service cluster acts as a monitoring gateway and fetches metrics from other, remote Streams Replication Manager Service clusters. This enables you to monitor the replications of your whole deployment using a single Streams Replication Manager Service cluster.

About this task

You enable Remote Querying for the Streams Replication Manager Service clusters that you want to designate as monitoring gateways. Once enabled, the designated Streams Replication Manager Service cluster discovers and communicates with other, remote Streams Replication Manager Service clusters and fetches the metrics collected by the remote Streams Replication Manager Service clusters. After configuration is complete, you can use the designated Streams Replication Manager Service cluster to monitor all replications in your deployment.

Configuration is done in Cloudera Manager by providing a list of external Kafka cluster aliases in the Streams Replication Manager Service Remote Target Clusters property. These clusters must be defined in Cloudera Manager using Kafka credentials.

Before you begin

- If any of the Streams Replication Manager installations are located behind a proxy, complete Configuring the advertised information of the Streams Replication Manager Service role

- Define all external Kafka clusters that you want to monitor.
  - External clusters are defined by creating and configuring Kafka credentials. Kafka credentials are created and configured in Cloudera Manager on the Administration External Accounts Kafka Credentials page. For more information, see Defining external Kafka clusters.
  - Ensure that the Name property of each Kafka credential you create for the external Kafka clusters matches the alias set for that Kafka cluster in the configuration of that Kafka cluster's co-located Streams Replication Manager instance. The alias can be found in the Streams Replication Manager Co-located Kafka Cluster Alias property.
  - Note down the Name property of each Kafka credential you create for the external Kafka clusters. You need to provide these during configuration.
- Ensure that network connectivity and DNS resolution are established between the clusters in your deployment. Additionally, the Streams Replication Manager Service's HTTPS port (default 6671) must have both inbound and outbound traffic enabled.

Procedure

1. In Cloudera Manager, select the Streams Replication Manager service.
2. Go to Configuration.
3. Find and configure the Streams Replication Manager Service Remote Target Clusters property.

   Add a comma separated list of the external Kafka cluster aliases that you want to monitor. For example:

   ```
   remote_1, remote_2
   ```

   Tip: Cluster aliases are specified during Kafka credential creation. A cluster alias is equivalent to the Name of the Kafka credential. Ensure that the aliases you add match the names of the Kafka credentials you created. Additionally, ensure that the aliases are identical with the aliases used on the remote clusters that you want to monitor.
4. Click Save Changes.
5. Restart the Streams Replication Manager service.

Results

Remote Querying is enabled and configured for the selected Streams Replication Manager Service cluster. The Streams Replication Manager Service cluster discovers and then communicates with other, remote Streams Replication Manager Service clusters and fetches metrics available on the remote Streams Replication Manager Service clusters.

What to do next

- If any of the remote Streams Replication Manager Services have Basic Authentication enabled, complete Configuring Basic Authentication for Remote Querying.
- Access the **Replications** page on the Streams Messaging Manager UI. Remote replications will be visible in the UI.
- Query metrics using the Streams Replication Manager REST API. For example:
  1. Go to Streams Replication Manager Web UI SRM Service Swagger UI .
  2. Find and open the /v2/replications endpoint.
  3. Click Try it out then click Execute.

     The response includes all discovered replications, replicated topics, and various other metrics. This includes replications that target remote clusters.

Related Information

Remote Querying

Configuring the advertised information of the Streams Replication Manager Service role

If your Streams Replication Manager deployment is located behind a proxy, you must configure the information advertised by a cluster of Streams Replication Manager Service roles (Streams Replication Manager Service cluster). Otherwise, the Streams Replication Manager Service cluster cannot be discovered by other Streams Replication Manager Service clusters and Remote Querying will not be possible.

About this task

Each Streams Replication Manager Service cluster advertises itself automatically through its target Kafka cluster. Advertised information includes properties like the protocol, host, port, and root API path of the Streams Replication Manager Service cluster. This information is used by Streams Replication Manager Service clusters that have Remote Querying enabled to connect to and communicate with remote Streams Replication Manager Service clusters and fetch the metrics collected by them.

However, if a Streams Replication Manager Service cluster is located behind a proxy, the information advertised by default cannot be used to connect to them. In such a case, the advertised protocol, host, port, and root API path must be configured to match the properties of the proxy. If this configuration is not completed, the Streams Replication Manager Service clusters cannot be discovered.

Procedure

1. In Cloudera Manager, select the Streams Replication Manager service.
2. Go to Configuration.
3. Find and configure the following properties:
   - SRM Service Advertised Protocol For Remote Queries
   - SRM Service Advertised Host For Remote Queries
   - SRM Service Advertised Port For Remote Queries
   - SRM Service Advertised API Root Path For Remote Queries
4. Click Save Changes.
5. Restart the Streams Replication Manager service.

Results

The Streams Replication Manager Service cluster advertises the configured protocol, host, port and API root path.

Related Information

Remote Querying

Configuring Streams Replication Manager Driver retry behaviour

If a target Kafka cluster is not available at startup, the replications targeting that cluster are not set up. In a case like this, the Streams Replication Manager Driver sets up the replications for available clusters and retries replication setup for the unavailable clusters. Retry behaviour can be configured in Cloudera Manager.

About this task

By default the Streams Replication Manager Driver is configured to retry indefinitely with a retry interval of 10000 ms (10 seconds). Both the retry count and retry interval can be configured in Cloudera Manager using the Retry Count for SRM Driver and Retry Delay for SRM Driver properties.

Retry Count for SRM Driver

Configures the maximum number of retries. The default value of -1 means that retries are carried out indefinitely. If the configured retry count is reached, the Streams Replication Manager Driver will stop. Additionally, setting the property to 0 is the same as setting it to 1.

Retry Delay for SRM Driver

Configures the interval at which retries happen, this is the delay (in ms) between each retry.

Note: When retries are happening, at least one of the configured replications is not running as expected. As a result, the DISTRIBUTED_HERDER_STATUS health test returns that replication status is bad. Based on the health test result, Cloudera Manager will display that the Streams Replication Manager Driver is in bad (red) health.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find and configure the following properties:
   - Retry Count for SRM Driver
   - Retry Delay for SRM Driver
4. Click Save Changes.
5. Restart the Streams Replication Manager service.

Results

Retry behaviour is configured for the Streams Replication Manager Driver. The Streams Replication Manager Driver uses the configured values if a Kafka cluster is unavailable at startup.

Configuring Streams Replication Manager Driver heartbeat emission

The Streams Replication Manager Driver spins up a Kafka Connect worker for each possible replication in your deployment. However, some of these workers created for disabled replications might be unnecessary. Unnecessary workers can be deactivated by configuring heartbeat emission. This can help with minimizing unnecessary performance overhead.

About this task

At startup the Streams Replication Manager Driver creates a Kafka Connect worker for each possible replication based on the cluster aliases you have configured. This means that a Connect worker is created not only for the replications that you specifically enable, but also for the ones that are disabled. The difference between an enabled and disabled replication is that for disabled replications not all possible connectors are set up. For disabled replications only the MirrorHeartbeatConnector is created which is responsible for heartbeat emission. For more information on Connect workers and connector instances created by the Streams Replication Manager Driver, see *Streams Replication Manager Architecture*.

In certain scenarios, some of the Connect workers set up for a disabled replication are unnecessary. In such a case, you can configure Streams Replication Manager to not create these Connect workers and to fully deactivate unutilized replications. This can be done by disabling heartbeat emission for these replications. If heartbeat emission is disabled, the Connect worker and MirrorHeartbeatConnector are not created. As a result, the replication associated with that worker is fully deactivated. This can help in minimizing the performance overhead caused by unnecessary Connect workers.

Heartbeat emission is configured with the emit.heartbeats.enabled property. This property can be set on a global as well as replication specific level. On a global level, heartbeat emission can be turned on and off using the Enable Heartbeats Cloudera Manager property. On a replication level, configuration is done by adding emit.heartbeats.enab

led to Streams Replication Manager's Replication Configs with a replication prefix. For example, A->B.emit.heartbeats.enabled=true.

Which disabled replications are unnecessary differs from deployment to deployment. To operate correctly, Streams Replication Manager requires that all clusters that act as a source have a heartbeats topic. The heartbeats topic is created by Connect workers that target the cluster. This means that at minimum, each source cluster must be targeted by at least a single enabled or disabled replication. All other disabled replications can be fully deactivated.

For example, assume that you have three clusters, A, B, and C. The Streams Replication Manager Driver is targeting all three clusters. Because the Streams Replication Manager Driver spins up a Connect worker for each possible replication, the following Connect workers are created:

- B->A
- C->A
- A->B
- C->B
- A->C
- B->C

Now, assume that the enabled replications, the ones that replicate data, are A->B, B->C, and C->A. This means that all three clusters act as sources. Therefore, heartbeat emission must be enabled in at least a single replication that is targeting the clusters. All other replications can be fully deactivated. In this example, the enabled replications A->B, B->C, and C->A target one of the clusters each. As a result, these replications take care of heartbeating. Out of all replications, the following can be deactivated:

- B->A
- C->B
- A->C

This can be achieved by for example enabling heartbeat emission on a global level with Enable Heartbeats and adding the following entries to Streams Replication Manager's Replication Configs:

```
B->A.emit.heartbeats.enabled=false
C->B.emit.heartbeats.enabled=false
A->C.emit.heartbeats.enabled=false
```

Consider another example. Assume that you have the same setup, but the enabled replications are A->B and B->C. Cluster A acts as a source but is not targeted by any enabled replication. In a case like this, you can only fully deactivate replications C->B and A->C. One of the replications targeting cluster A (B->A or C->A) must remain disabled and cannot be deactivated. Otherwise, a heartbeats topic will not be created in cluster A.

This can be achieved by for example enabling heartbeat emission on a global level with Enable Heartbeats and adding the following entries to Streams Replication Manager's Replication Configs:

```
C->B.emit.heartbeats.enabled=false
A->C.emit.heartbeats.enabled=false
B->A.emit.heartbeats.enabled=false
```

In this example, replication B->A was fully deactivated (heartbeat emission disabled) and replication C->A was left disabled. Although replication C->A is disabled, it is still actively emitting heartbeats as a result of Enable Heartbeats being enabled.

Procedure

1. In Cloudera Manager, select the Streams Replication Manager Service.
2. Go to Configuration.

3. Configure heartbeat emission for your replications.

   You have two options. You can enable heartbeat emission on a global level and disable it for the required replications, or disable heartbeat emission on a global level and enable it for the required replications.

   For Enable globally and disable per replication

   a. Find and enable the Enable Heartbeats property.
   b. Find and configure the Streams Replication Manager's Replication Configs property.

      Add the emit.heartbeats.enabled property with a correct replication prefix. For example:

      ```
      A->B.emit.heartbeats.enabled=false
      ```

      Repeat this step for all replications that you want to disable heartbeat emission for.

   For Disable globally and enable per replication

   a. Find and disable the Enable Heartbeats property
   b. Find and configure the Streams Replication Manager's Replication Configs property.

      Add the emit.heartbeats.enabled property with correct replication prefix. For example:

      ```
      A->B.emit.heartbeats.enabled=true
      ```

      Repeat this step for all replications that you want to enable heartbeat emission for.

4. Click Save Changes.
5. Restart Streams Replication Manager.

Results

Heartbeat emission is configured. Unnecessary replications are fully deactivated.

Related Information

Streams Replication Manager Architecture

Configuring automatic group offset synchronization

Learn how to enable automatic group offset synchronization in Streams Replication Manager. Enabling this feature can simplify the manual steps that you need to take to migrate consumer groups in a failover or failback scenario.

About this task

By default, automatic group offset synchronization is disabled. It can be enabled in Cloudera Manager by adding new entries to the Streams Replication Manager's Replication Configs property.

Depending on your use case, you can set up automatic group offset synchronization on a global level or for specific replications only.

Procedure

1. In Cloudera Manager, select Streams Replication Manager.
2. Go to Configuration.
3. Find the Streams Replication Manager's Replication Configs and add the following configuration entries:

   ```
   sync.group.offsets.enabled = true
   ```

```
sync.group.offsets.interval.seconds = [***TIME IN SECONDS***]
```

In this example, all properties are set on a global level. This means that automatic group offset synchronization is applied to all replications. Depending on your setup, you can also choose to set these properties for specific replications using appropriate replication prefixes.

The sync.group.offsets.enabled property enables automatic group offset synchronization. The sync.group.offsets.interval.seconds property controls how frequently offsets are synched. You only need to specify this property if you want to customize synchronization frequency.

4. Enter a Reason for change, and then click Save Changes to commit the changes.
5. Restart Streams Replication Manager.

Results

Automatic group offset synchronization is enabled. From now on, Streams Replication Manager automatically exports the translated offsets of the configured source clusters and applies them to the configured target clusters.

Related Information

Automatic group offset synchronization

Configuring Streams Replication Manager Driver for performance tuning

Learn about the configuration options available for tuning the performance of Streams Replication Manager Driver.

To perform performance tuning for Streams Replication Manager Driver, it is important that you understand the task architecture of Streams Replication Manager. For more information, see Streams Replication Manager Architecture.

It is also important that you are familiar with the task load balancing method used by Streams Replication Manager. For more information, see Task architecture and load-balancing.

You need to tune the following main groups of configurations for the Streams Replication Manager Driver when running a replication flow with a heavy load:

- Hardware
- Task count
- Consumer and producer configurations of MirrorSourceTask
- Connect worker configurations

Hardware

Learn about the hardware-related aspects of Streams Replication Manager performance tuning, such as memory requirements.

For high workloads, Streams Replication Manager Drivers need to run on dedicated nodes. Streams Replication Manager Drivers are typically network-bound, but due to configuration tweaks, memory consumption can also be high.

For small workloads, a maximum heap size of 8 GB is sufficient, however, for larger workloads, Cloudera recommends a higher maximum heap size. Streams Replication Manager Driver performance can significantly degrade when limited by memory or due to garbage collection pauses. Always make sure that Streams Replication Manager Drivers have enough memory. You can define the heap size using the SRM_HEAP_OPTS property in Cloudera Manager.

Task count

Learn about configuring optimal task counts in relation to the number of topic partitions, Streams Replication Manager Drivers, and Streams Replication Manager Driver nodes.

The task count specifies the level of parallelism of the data replication. One task corresponds to (approximately) one worker thread on the Streams Replication Manager Driver cluster. Each task runs a consumer and a producer instance, meaning more connections to the source and target Kafka clusters.

To increase the parallelism of the work, you can increase the Tasks Max property in Cloudera Manager. Since the unit of work is the source topic-partition, there is no need for the value of Tasks Max to exceed the number of replicated topic-partitions, as it has no effect on the throughput of the system.

When Streams Replication Manager Drivers are running on dedicated nodes, Cloudera recommends setting the value of Tasks Max approximately to the number of Streams Replication Manager Drivers times the number of Streams Replication Manager Driver node cores.

Tip: Consider changing this configuration when the consumer and producer batching configurations are already tweaked based on the source data bandwidth, and the CPU is underutilized on the Streams Replication Manager Driver nodes.

However, increasing Tasks Max may affect CPU usage. Make sure to monitor CPU usage metrics when tweaking this property.

Consumer and producer configurations

Learn about consumer and producer properties you can tweak to increase throughput and decrease overhead.

A MirrorSourceTask manages one consumer and one producer to drive the data replication. Similarly to custom Kafka client applications, you need to tweak these consumer and producer instances when expecting a high throughput on the replication flow. The following list is only an overview of properties that are typically tweaked for the consumer and the producer, as well as their application for Streams Replication Manager. On some workloads, you might also need to tweak broker configuration.

Consumer configurations

To configure the consumer inside the MirrorSourceTask, use the cluster->cluster.consumer. prefix in Streams Replication Manager's Replication Configs in Cloudera Manager. You can tweak the following consumer properties to increase throughput:

fetch.max.bytes

The maximum size of a fetch response. You can increase this to increase the consumer throughput and reduce the overhead of fetching.

max.poll.records

The maximum number of records returned in a fetch response. You can increase this to increase the consumer throughput and reduce the overhead of fetching.

receive.buffer.bytes

The size of the TCP receive buffer used by the consumer. When the average response size increases due to tweaking the previous configurations, you also need to increase the receive buffer bytes to reduce the overhead of buffering.

max.partition.fetch.bytes

The maximum amount of data returned for one topic partition in a fetch response. In some cases, where there are source topics with high ingestion rate to be replicated, you need to increase this configuration to allow the consumer to keep up with the ingestion rate. If the source topics have a similar ingestion rate, you do not need to change this.

Producer configurations

To configure the producer inside the MirrorSourceTask, use the cluster->cluster.producer.override. prefix in Streams Replication Manager's Replication Configs in Cloudera Manager. You can tweak the following producer properties to increase throughput:

Producer Batch Size

The maximum batch size created by the producer. Batches are created for each topic partition. You can increase this to increase the producer throughput and reduce the overhead of producing. When tweaking this property, make sure to check the value of max.request.size, as well as message.max.bytes on the broker side.

send.buffer.bytes

The size of the TCP send buffer used by the producer. When the average request size increases due to tweaking the previous configurations, you also need to increase the send buffer bytes to reduce the overhead of buffering.

Producer Compression Type

The compression type to use in the producer. You can use producer side compression to achieve higher throughput.

max.request.size

The maximum request size sent by the producer. The batch size cannot exceed this configuration. When tweaking this property, make sure to check the value of Producer Batch Size, as well as message.max.bytes on the broker side.

Producer Buffer Memory

The maximum amount of memory the producer can use for buffering records. For high throughput replications, heavy batching is necessary on the producer side. To allow heavy batching, you need to increase the buffer memory.

Note: Cloudera does not recommend changing the following Connect producer properties, because the changes might lead to violating Streams Replication Manager guarantees.

- acks (default: all)
- retries (default: int max)
- max.in.flight.requests.per.connection (default: 1)

Related Information

Streams Replication Manager Architecture

Connect worker configurations

Learn about the Connect worker properties you can change to facilitate heavy loads on Streams Replication Manager.

The Connect worker does not significantly influence the throughput of Streams Replication Manager. However, when running a heavy load with Streams Replication Manager, you might need to tweak the following properties.

offset.flush.timeout.ms

Affects the timeout when waiting for in-flight produce requests to finish before committing the source offsets. The default value is five seconds, which is a low timeout. It might be exceeded when there is a high load on a single task. You can safely increase this value, as the timeout does not affect the throughput of the actual flow, and data replication continues even when the Connect framework is waiting for a safe source offset commit.

Tip: Consider changing this when the ERROR level message Failed to flush, timed out while waiting for producer to flush outstanding X messages appears in the Streams Replication Manager Driver logs .

Offset Flush Interval

Controls the interval of the source offset flush. The offset flush does not have a significant impact in the data replication throughput. A possible reason to tweak this configuration is the average offset flush duration getting close to the interval. In such a case, increasing the interval might reduce unnecessary work.

Tip: Consider changing this when the average flush duration seems to be close to the flush interval, based on the DEBUG level message Finished offset commitOffsets successfully in X ms in the Streams Replication Manager Driver logs.

Metric calculation

The calculation of the following metrics might impact performance.

replication-records-lag

You can tweak the performance impact of the replication-records-lag metric by tweaking the values of the following properties in Streams Replication Manager's Replication Configs in Cloudera Manager.

replication.records.lag.calc.enabled

Controls whether the replication-records-lag metric is calculated. This metric provides information regarding the replication lag based on offsets. The metric is available both on the cluster and the topic level. The calculation of this metric might add latency to replications and impact Streams Replication Manager performance. If you are experiencing performance issues, you can try setting this property to false to disable the calculation of replication-records-lag. Alternatively, you can try fine-tuning how Streams Replication Manager calculates replication-records-lag with the replicat ion.records.lag.calc.period.ms and replication.records.lag.end.offset.timeout.ms properties.

replication.records.lag.calc.period.ms

Controls how frequently Streams Replication Manager calculates the replication-records-lag metric. The default value of 0 means that the metric is calculated continuously. Cloudera recommends configuring this property to 15000 ms (15 seconds) or higher if you are experiencing issues related to the calculation of replication-records-lag. A calculation frequency of 15 seconds or more results in the metric being available for consumption without any significant impact on Streams Replication Manager performance.

replication.records.lag.end.offset.timeout.ms

Specifies the Kafka end offset timeout value used for calculating the replication-records-lag metric. Setting this property to a lower value than the default 6000 ms (1 minute) might reduce latency in calculating replication-records-lag, however, replication-records-lag calculation might fail. A value higher than the default can help avoid metric calculation failures, but might increase replication latency and lower Streams Replication Manager performance.

New topic and consumer group discovery

Kafka topics or consumer groups may not get replicated instantly when they are added to white and blacklists. This is due to the default behaviour of how topics and consumer groups are discovered by Streams Replication Manager.

The discovery and replication of newly created topics or consumer groups is not instantaneous. Streams Replication Manager checks source clusters for new topics and consumer groups periodically, as controlled by the Refresh Topics Interval Seconds and Refresh Groups Interval Seconds properties. By default both properties are set to 10 minutes. As a result, the discovery and replication of new topics or groups can take up to 10 minutes.

Cloudera does not recommend using a refresh interval lower than the default value for production environments as it can lead to severe performance degradation.

Related Information

srm-control Topics and Groups Subcommand

Configuration examples

These configuration examples give step-by-step instructions on how you can set up and configure typical deployments of Streams Replication Manager. Reviewing these examples can help you gain a better understanding of how your specific setup can be configured.

Bidirectional replication example of two active clusters

Review the bidirectional replication example to learn how you can configure and start replication with Streams Replication Manager in a deployment with two active clusters configured with bidirectional replication.

About this task

In a typical scenario, you may have two active Kafka clusters within the same region but in separate availability zones. With bidirectional replication, clients can connect to either cluster in case one is temporarily unavailable.

This example demonstrates the steps required to configure the deployment shown below. Additionally, it also provides example commands to start replication between clusters.

Figure 1: Bidirectional Replication of Active Clusters

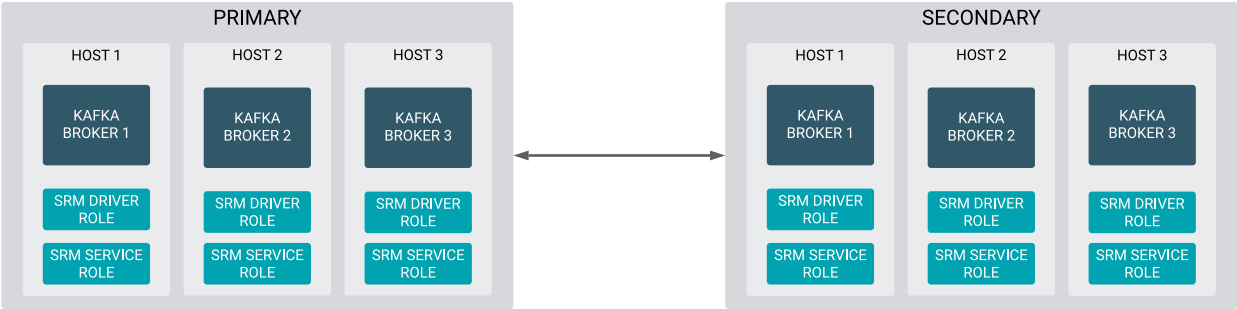

Before you begin

- The following list of steps assume that both clusters are unsecured.
- The following list of steps assume that the Streams Replication Manager Service role is running on all Kafka broker hosts and is targeting its co-located cluster (the cluster it is running in).
- Steps 1 through 6 must be carried out on all clusters for all Streams Replication Manager services. The steps highlight when the configuration differs from cluster to cluster or if it is identical on all clusters.
- The following list of steps assume that the `DefaultReplicationPolicy` is in use.

Procedure

1. Define external clusters:
   a) In Cloudera Manager, go to Administration External Accounts .
   b) Go to the Kafka Credentials tab.

      On this tab you will create a credential for each external cluster taking part in the replication process.
   c) Click Add Kafka credentials.
   d) Configure the Kafka credential:

      On the primary cluster you have to add a credential that defines secondary. For example:

      ```
      Name=secondary
      Bootstrap servers=secondary-1.cloudera.com:9092, secondary-2.cloudera
      .com:9092, secondary-3.cloudera.com:9092
      Security protocol=PLAINTEXT
      ```

      On the secondary cluster you have to add a credential that defines primary. For example:

      ```
      Name=primary
      Bootstrap servers=primary-1.cloudera.com:9092, primary-2.cloudera.com:9
      092, primary-3.cloudera.com:9092
      Security protocol=PLAINTEXT
      ```
   e) Click Add.

      If credential creation is successful, a new entry corresponding to the Kafka credential you specified appears on the page.
2. Define the co-located Kafka cluster:
   a) In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
   b) Go to Configuration.
   c) Find and enable the Kafka Service property.
   d) Find and configure the Streams Replication Manager Co-located Kafka Cluster Alias property.

      The alias you configure represents the co-located cluster. Enter an alias that is unique and easily identifiable. On the primary cluster:

      ```
      primary
      ```

      On the secondary cluster:

      ```
      secondary
      ```
3. Add the clusters defined with Kafka credentials to Streams Replication Manager's configuration:
   a) Find and configure the External Kafka Accounts property.
   b) Click the add button and add new lines for each Kafka credential you created.
   c) Add the names of all Kafka credentials.

      Each Kafka credential must be added to a new line. On the primary cluster:

      ```
      secondary
      ```

      On the secondary cluster:

      ```
      primary
      ```

4. Find and configure the Streams Replication Manager Cluster alias property.

   Add all cluster aliases to this property. This includes the aliases present in both the External Kafka Accounts and Streams Replication Manager Co-located Kafka Cluster Alias properties. Delimit the aliases with commas. In the case of this example the configuration is identical on both clusters:

   ```
   primary, seconadry
   ```

5. Configure Driver role target clusters:
   a) Find the Streams Replication Manager Driver Target Cluster property.
   b) Add the cluster aliases that you want the driver role to target.

      In the case of this example, each Driver should target its co-located cluster. On the primary cluster:

      ```
      primary
      ```

      On the secondary cluster:

      ```
      secondary
      ```

6. Add and enable replications:
   a) Find the Streams Replication Manager's Replication Configs property.
   b) Click the add button and add new lines for each unique replication you want to add and enable.
   c) Add and enable your replications.

      In the case of this example, the configuration will be identical on both clusters:

      ```
      primary->secondary.enabled=true
      secondary->primary.enabled=true
      ```

7. Replicate data between clusters with the following commands:

   Important: The following commands assume that the `DefaultReplicationPolicy` is in use. Running the following commands with the `IdentityReplicationPolicy` would result in replication loops.

   ```
   srm-control topics --source primary --target secondary --add ".*"
   ```

   ```
   srm-control topics --source secondary --target primary --add ".*"
   ```

Cross data center replication example of multiple clusters

Review the cross data center replication example to understand how you can configure and start replication with Streams Replication Manager in a deployment with three data centers that each have two Kafka clusters.

About this task

In more advanced deployments, you may have multiple Kafka clusters in each of several data centers. To prevent creating a fully-connected mesh of all Kafka clusters, Cloudera recommends leveraging a single Kafka cluster in each data center for cross data center replication.

This example demonstrates the steps required to configure the deployment shown below. Additionally, it provides example commands that start bidirectional replication of all topics within each data center as well as example commands that replicate a single topic across all data centers

Figure 2: Cross Data Center Replication of Multiple Clusters

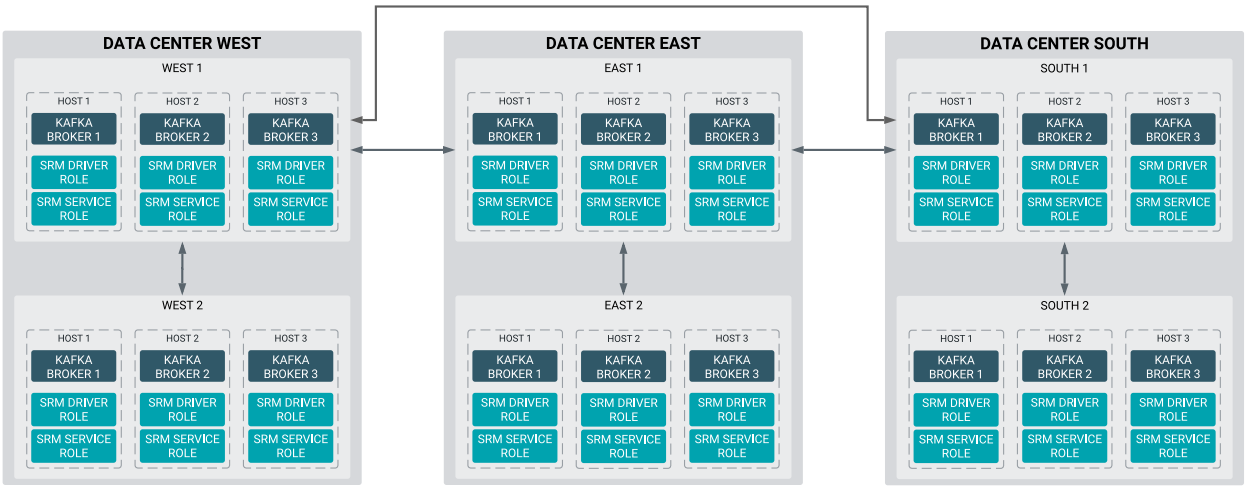

Before you begin

- The following list of steps assume that all clusters are unsecured.
- The following list of steps assume that both Streams Replication Manager Service and Driver roles are running on all Kafka broker hosts. Additionally, both roles have a single target which is their co-located cluster (the cluster they are running in).
- Clusters West 1, East 1, and South 1 are collectively referred to as primary clusters, while clusters West 2, East 2, and South 2 are collectively referred to as secondary clusters.
- Steps 1 through 8 must be carried out on each of the clusters individually. That is, the properties presented in these steps must be configured on all clusters. However, how you configure the properties for each individual cluster (the values that you set) will be different. The steps provide an explanation for each configuration and provide explicit examples for the clusters in Data Center West.
- The following list of steps assume that the `DefaultReplicationPolicy` is in use.

Procedure

1. Define the external Kafka cluster:

External clusters are configured with Kafka credentials. On each primary cluster you need to create a total of three Kafka credentials. Two representing the other two primary clusters and another one representing the secondary cluster that is in the same data center. On each secondary cluster you need to create a single Kafka credential representing the primary cluster that is running in the same data center.

a) In Cloudera Manager, go to Administration External Accounts .

b) Go to the Kafka Credentials tab.

c) Click Add Kafka credentials.

d) Configure the Kafka credentials.

For example, in West 1, you need to create credentials for East 1, South 1, and West 2. In West 2, you need to create a credential for West 1.

- West 1

For East 1 Credential

```
Name=east1
Bootstrap servers=east1-host1.cloudera.com:9092, east1-host2.clou
dera.com:9092, east1-host2.cloudera.com:9092
```

```
Security protocol=PLAINTEXT
```

For South 1 Credential

```
Name=south1
Bootstrap servers=south1-host1.cloudera.com:9092, south1-host2.c
loudera.com:9092, south1-host2.cloudera.com:9092
Security protocol=PLAINTEXT
```

For West 2 Credential

```
Name=west2
Bootstrap servers=west2-host1.cloudera.com:9092, west2-host2.clou
dera.com:9092, west2-host2.cloudera.com:9092
Security protocol=PLAINTEXT
```

- West 2

For West 1 Credential

```
Name=west1
Bootstrap servers=west1-host1.cloudera.com:9092, west1-host2.clou
dera.com:9092, west1-host2.cloudera.com:9092
Security protocol=PLAINTEXT
```

e) Click Add.

If credential creation is successful, a new entry corresponding to the Kafka credential you specified appears on the page.

2. Define the co-located Kafka cluster:

a) In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.

b) Go to Configuration.

c) Find and enable the Kafka Service property.

d) Find and configure the Streams Replication Manager Co-located Kafka Cluster Alias property.

The alias you configure represents the co-located cluster. Enter an alias that is unique and easily identifiable.

For example:

- West 1

```
west1
```

- West 2

```
west2
```

3. Add the clusters defined with Kafka credentials to Streams Replication Manager's configuration:

a) Find and configure the External Kafka Accounts property.

b) Click the add button and add new lines for each Kafka credential you created.

c) Add the names of all Kafka credentials.

Add each Kafka credential that you created in the cluster that you are configuring. When configured correctly, the property will include three credential names in each primary and a single credential name in each secondary cluster. Each Kafka credential must be added to a new line.

For example, in West 1 you must add the names of the Kafka credentials that you created in West 1:

```
east1
south1
```

```
west2
```

In West 2, you must add a the name of the single credential created for West 1:

```
west1
```

4. Find and configure the Streams Replication Manager Cluster alias property.

   Add all cluster aliases to this property. This includes the aliases present in both the External Kafka Accounts and Streams Replication Manager Co-located Kafka Cluster Alias properties. Delimit the aliases with commas. For example:

   - West 1

     ```
     west1, west2, east1, south1
     ```

   - West 2

     ```
     west1, west2
     ```

5. Add and enable replications:

   a) Find the Streams Replication Manager's Replication Configs property.
   b) Click the add button and add new lines for each unique replication you want to add and enable.
   c) Add and enable your replications.

      In the case of this example, you need to enable both cross data center replication and replication within each data center. This can be done by configuring three replications in each primary cluster and a single replication in each of the secondary clusters. Each of the replications that you configure should target the cluster that you are configuring. For example:

      - West 1

        ```
        east1->west1.enabled=true
        south1->west1.enabled=true
        west2->west1.enabled=true
        ```

      - West 2

        ```
        west1->west2.enabled=true
        ```

6. Click Save Changes.
7. Restart the Streams Replication Manager service.
8. Start replication with the srm-control tool:

   The following command examples demonstrate how you can start replication of a single topic across all data centers and how you can replicate all topics within each data center.

   Important: The following commands assume that the `DefaultReplicationPolicy` is in use. Running the following commands with the `IdentityReplicationPolicy` would result in replication loops.

   a) SSH into one of the hosts in West 1 and run the following commands:

      Replicate topic1 across data centers.

      ```
      srm-control topics --source east1 --target west1 --add topic1,east2.topi
      c1
      srm-control topics --source south1 --target west1 --add topic1,south2
      .topic1
      ```

      Replicate all topics within the data center.

      ```
      srm-control topics --source west1 --target west2 --add ".*"
      ```

```
srm-control topics --source west2 --target west1 --add ".*"
```

b) SSH into one of the hosts in East 1 and run the following commands:

Replicate topic1 across data centers.

```
srm-control topics --source west1 --target east1 --add topic1,west2.topi
c1
srm-control topics --source south1 --target east1 --add topic1,south2
.topic1
```

Replicate all topics within the data center.

```
srm-control topics --source east1 --target east2 --add ".*"
srm-control topics --source east2 --target east1 --add ".*"
```

c) SSH into one of the hosts in South 1 and run the following commands:

Replicate topic1 across data centers.

```
srm-control topics --source west1 --target south1 --add topic1,west2.top
ic1
srm-control topics --source east1 --target south1 --add topic1,east2.top
ic1
```

Replicate all topics within the data center.

```
srm-control topics --source south1 --target south2 --add ".*"
srm-control topics --source south2 --target south1 --add ".*"
```

Cloudera Runtime 7.3.1

Streams Streams Replication Manager Overview

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Overview 4

Key Features 5

Main Use Cases 7

Streams Replication Manager use case architectures 7
- Highly available Kafka architectures 7
- Cluster migration architectures 10

Streams Replication Manager Architecture 12
- Streams Replication Manager Driver 12
  - Connect workers 13
  - Connectors 15
  - Task architecture and load-balancing 17
  - Driver inter-node coordination 19
- Streams Replication Manager Service 19
  - Remote Querying 21

Streams Replication Manager monitoring and metrics 24

Streams Replication Manager Service REST API 27

Streams Replication Manager replication flows and replication policies 28

Streams Replication Manager remote topic discovery 32

Automatic group offset synchronization 33

Understanding co-located and external clusters 34

Understanding Streams Replication Manager properties, their configuration and hierarchy 38

Overview

Get familiar with Streams Replication Manager and its components.

Streams Replication Manager is an enterprise-grade replication solution that enables fault tolerant, scalable and robust cross-cluster Kafka topic replication. Streams Replication Manager provides the ability to dynamically change configurations and keeps the topic properties in sync across clusters at high performance. Streams Replication Manager also delivers custom extensions that facilitate installation, management and monitoring making Streams Replication Manager a complete replication solution that is built for mission critical workloads. Streams Replication Manager consists of two main components. The Stream Replication Engine and the Stream Replication Management Services.

Figure 1: Streams Replication Manager Overview

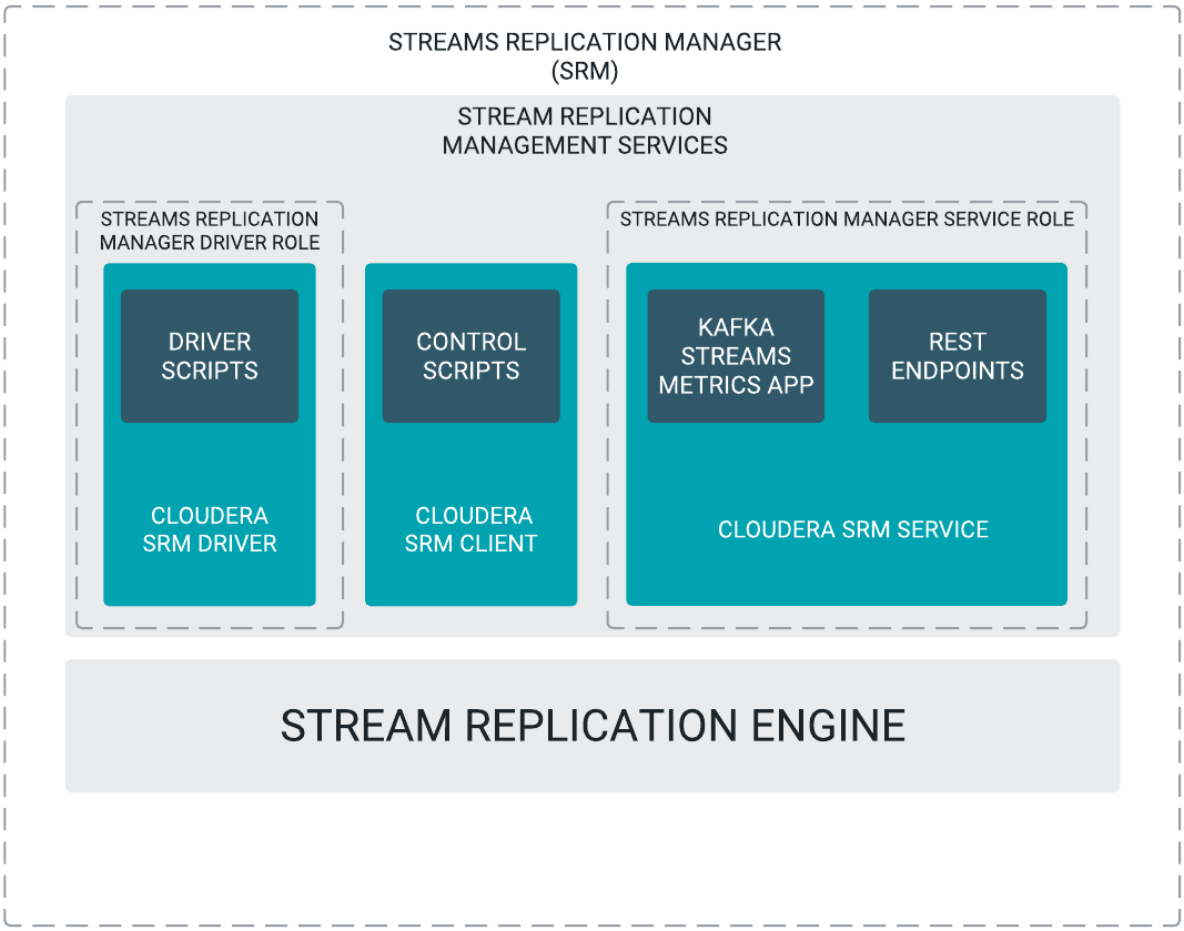

Stream Replication Engine

The Stream Replication Engine is a next generation multi-cluster and cross-datacenter replication engine for Kafka clusters.

Stream Replication Management Services

Stream Replication Management Services are services powered by open source Cloudera extensions which utilize the capabilities of the Stream Replication Engine. These services provide:

- Easy installation

- Lifecycle management
- Management and monitoring of replication flows across clusters

The Stream Replication Management Services includes the following custom extensions:

Cloudera Streams Replication Manager Driver

The Cloudera Streams Replication Manager Driver is a small wrapper around the Stream Replication Engine that adds the extensions provided by Cloudera. It provides the ability to spin up Streams Replication Manager clusters and has a metrics reporter. The driver is managed by Cloudera Manager and is represented by the Streams Replication Manager Driver role.

Cloudera Streams Replication Manager Client

The Cloudera Streams Replication Manager Client provides users with command line tools that enable replication management for topics and consumer groups. The command line tool associated with the Cloudera Streams Replication Manager Client is called srm-control.

Cloudera Streams Replication Manager Service

The Cloudera Streams Replication Manager Service consist of a REST API and a Kafka Streams application to aggregate and expose cluster, topic and consumer group metrics. The service is managed by Cloudera Manager and is represented by the Streams Replication Manager Service role.

Key Features

Streams Replication Manager has the following main features.

Remote topics

Streams Replication Manager replicates Kafka topics from source to target clusters. Remote topics are the replica topics located in target clusters. Remote topics and their source topics are tracked internally by Streams Replication Manager. Additionally, remote topics are by default prefixed with the name (alias) of the source cluster. The naming convention for remote topics is configurable and is determined by the replication policy that is currently in use. For more information, see *Replication flows and replication policies*.

Consistent semantics

Partitioning and record offsets are synchronized between replicated clusters to ensure consumers can migrate from one cluster to another without losing data or skipping records.

Cross cluster configuration

Topic-level configuration properties are synced across clusters. For example, the cleanup policy (cleanup.policy), or the log segment file size (segment.bytes), as well as other topic-level configurations are automatically synched to remote topics. This simplifies managing topics across multiple Kafka clusters.

Consumer group checkpoints

In addition to data and configuration, Streams Replication Manager replicates consumer group progress via periodic checkpoints. At configurable intervals, checkpoint records are emitted to downstream clusters, encoding the latest offsets for whitelisted consumer groups and topic-partitions. As with topics, groups are matched against an allowlist which can be updated dynamically with srm-control. Normally, consumer group offsets are not portable between Kafka clusters, as offsets are not consistent between otherwise identical topic-partitions on different clusters. Streams Replication Manager checkpoint records account for this by including offsets which are automatically translated from one cluster to another. This offset translation feature works in both directions; a consumer group can be migrated from one cluster to another (failover) and then back again (failback) without skipping records or losing progress.

Automatic topic and partition detection

Streams Replication Manager monitors Kafka clusters for new topics, partitions, and consumer groups as they are created. These are compared with configurable whitelists, which may include regular expressions.

Tooling to automate consumer migration

Streams Replication Manager tooling enables operators to translate offsets between clusters and to migrate consumer groups while preserving state.

Centralized configuration for multi-cluster environments

Streams Replication Manager leverages a single top-level configuration file to enable replication across multiple Kafka clusters. Moreover, command-line tooling can alter which topics and consumer groups are replicated in real-time.

Replication monitoring

Since cluster replication will mainly be used for highly critical Kafka applications, it is crucial for customers to be able to easily and reliably monitor the Kafka cluster replications. The Streams Replication Manager Service collects and aggregates Kafka replication metrics and make them available through a REST API. This REST API is used by Streams Messaging Manager to display metrics. Customers could also use the REST API to implement their own monitoring solution or plug it into third party solutions. The metrics make the state of cluster replication visible to end users who then can take corrective action if needed.

Replication policies

The replication policy used by Streams Replication Manager defines the basic rules of how Streams Replication Manager replicates data. Streams Replication Manager ships with two replication policies that are designed for different use cases. These are the `DefaultReplicationPolicy`, which uses topic name prefixes to provide replication loop detection, and the `IdentityReplicationPolicy`, which mimics the behavior of MirrorMaker 1 and provides prefixless replication. Both policies also support the monitoring features provided by the Streams Replication Manager Service.

In addition to the Cloudera provided policies, custom developed replication policies can be used. Developing and using your own replication policy enables you to gain full control over how Streams Replication Manager replicates data.

Important: Cloudera provides limited support for deployments that use custom replication policies. Additionally, some key features including replication monitoring with the Streams Replication Manager Service will not work if a custom replication policy is in use. Whenever possible, Cloudera recommends that you use the `DefaultReplicationPolicy`.

For more information, see *Replication flows and replication policies*.

Remote topic discovery and tracking

Streams Replication Manager uses an internal Kafka topic to track which topics are being replicated in a replication flow. This enables the Streams Replication Manager Service to filter topics from monitoring if they are no longer being replicated. Additionally, this feature enables replication monitoring with the Streams Replication Manager Service even if prefixless replication is being used.

Related Information

Monitoring Kafka cluster replications using Streams Messaging Manager

Streams Replication Manager replication flows and replication policies

Replication flows and replication policies

Main Use Cases

Learn about the main use cases of Streams Replication Manager.

Apache Kafka has become an essential component of enterprise data pipelines and is used for tracking clickstream event data, collecting logs, gathering metrics, and being the enterprise data bus in a microservices based architectures. Kafka supports internal replication to support data availability within a cluster. However with Kafka based applications becoming critical, enterprises require that the data availability and durability guarantees span entire cluster and site failures.

Replication of data across clusters and sites is key for the following use cases:

Disaster Recovery

Common enterprise use cases for cross-cluster replication is for guaranteeing business continuity in the presence of cluster or data center-wide outages.

Aggregation for Analytics

Aggregate data from multiple streaming pipelines possibly across multiple data centers to run batch analytics jobs that provide a holistic view across the enterprise.

Data Deployment after Analytics

This is the opposite of the aggregation use case in which the data generated by the analytics application in one cluster (say the aggregate cluster) is broadcast to multiple clusters possibly across data centers for end user consumption.

Isolation

Due to performance or security reasons, data needs to be replicated between different environments to isolate access. In many deployments the ingestion cluster is isolated from the consumption clusters.

Geo Proximity

In geographically distributed access patterns where low latency is required, replication is used to move data closer to the access location.

Cloud Migration

As more enterprises have an on-premise and cloud presence, Kafka replication can be used to migrate data to the public or private cloud and back.

Legal and Compliance

Much like the isolation uses case, a policy driven replication is used to limit what data is accessible in a cluster to meet legal and compliance requirements.

Streams Replication Manager use case architectures

You can deploy Streams Replication Manager in many different ways and for many different use cases. The following collects example use case architectures where Streams Replication Manager is used to achieve high availability or used for cluster migration.

Highly available Kafka architectures

A highly available Kafka deployment must be able to survive a full single cluster outage while continuing to process events without data loss. With Streams Replication Manager, you can implement highly available Apache Kafka deployments which either follow an Active/Standby or an Active/Active model.

Active/Standby architecture

In an Active/Standby scenario, you set up two Kafka clusters and configure Streams Replication Manager to replicate topics bidirectionally between both clusters. A VIP or load balancer directs your producers to ingest messages into the active cluster from which consumer groups are reading from.

Figure 2: Active/Standby standard operation

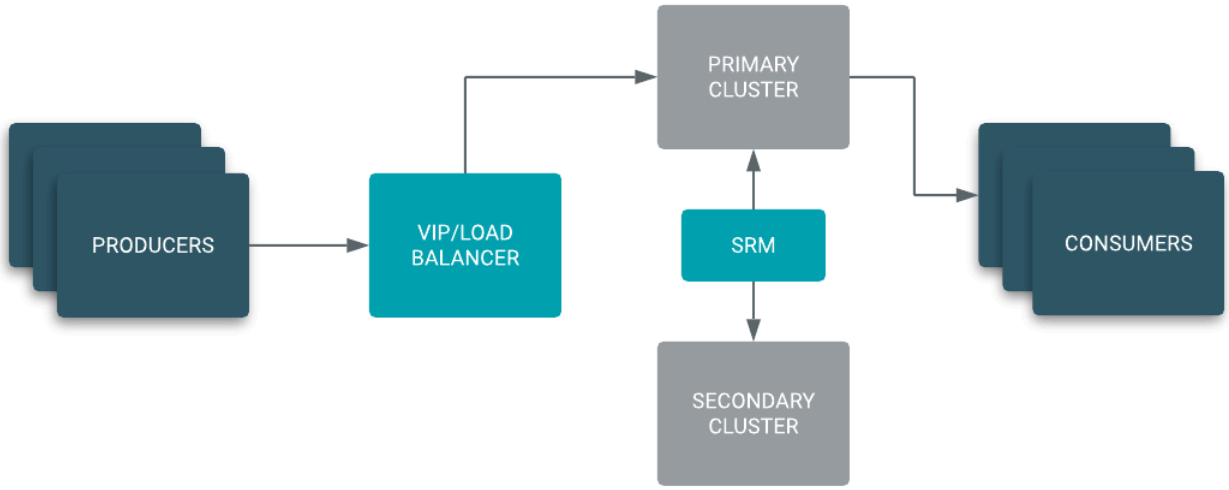

In case of a disaster, the VIP or load balancer directs the producers to the secondary cluster. You can easily migrate your consumer groups to start reading from the secondary cluster or simply wait until the primary cluster is restored if the resulting consumer lag is acceptable for your use case.

While the primary cluster is down, your producers are still able to ingest. Once the primary cluster is restored, Streams Replication Manager automatically takes care of synchronizing both clusters making failback seamless.

Figure 3: Active/Standby cluster failure

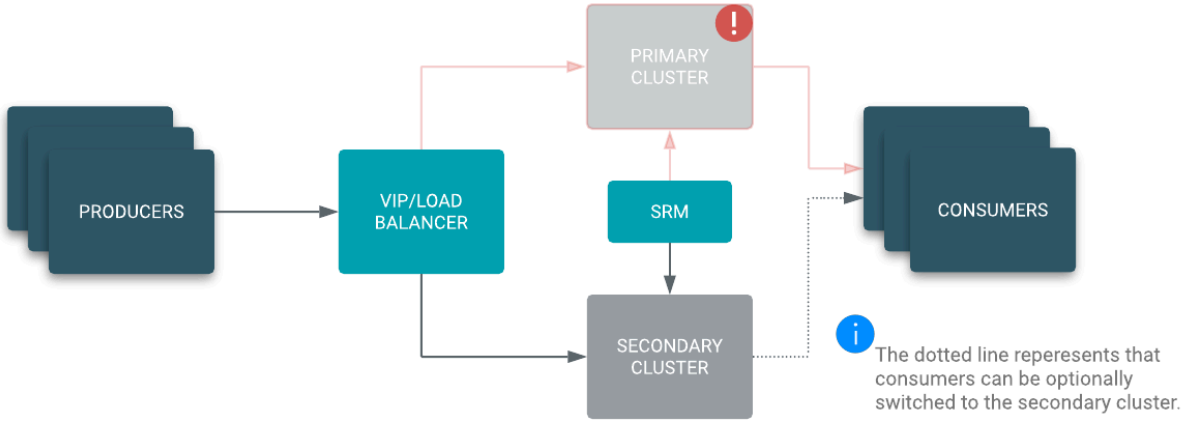

Implementing an Active/Standby architecture is the logical choice when an existing disaster recovery site with established policies is already available, and your goals include not losing ingest capabilities during a disaster and having a backup in your disaster recovery site.

Active/Active architecture

In an Active/Active scenario, your producers can be load balanced to either your primary or secondary cluster. Streams Replication Manager is configured to replicate topics bidirectionally between both clusters. What makes this architecture Active/Active is the fact that you now have consumers reading from both clusters at the same time, essentially acting like a cross-cluster consumer group.

In case of a disaster the VIP or load balancer directs the producers to the secondary cluster and the secondary cluster consumer group is still able to process messages. While the primary cluster is down, your producers are still able to ingest and your consumers are still able to process messages. This results in a zero downtime and hands-off failover in case of a disaster. Once the primary cluster is back online, Streams Replication Manager automatically takes care of synchronizing both clusters and your primary consumer group resumes processing messages.

Figure 4: Active/Active architecture

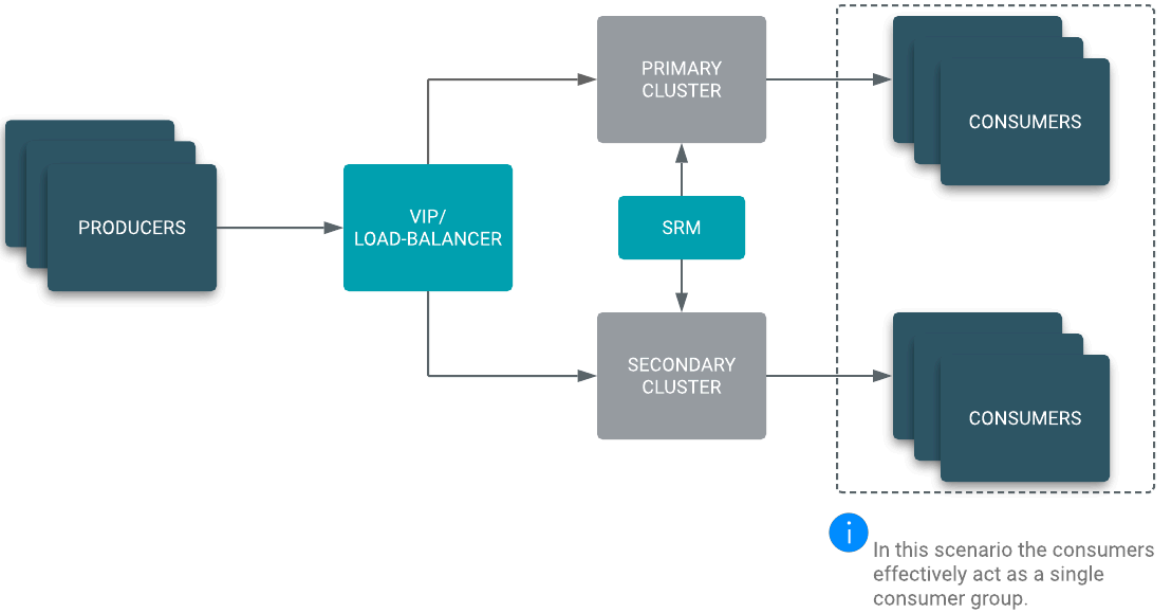

Cross data center replication

Certain applications not only require local high availability within one Data Center (DC) or one Availability Zone (AZ), but have to be highly available across DCs as well. You can use Streams Replication Manager to set up replication between Kafka clusters in different DCs. This results in messages being available to consumers in each of your DCs.

A load balancer directs your producers to the local or closest DC if the primary DC is down. Streams Replication Manager is configured to replicate topics between all DCs. If you are using more than two DCs, Streams Replication Manager is configured to create a “replication circle”, ensuring a single DC failure (for example, us-north in the example below) does not halt replication between the remaining clusters.

Figure 5: Cross DC replication architecture

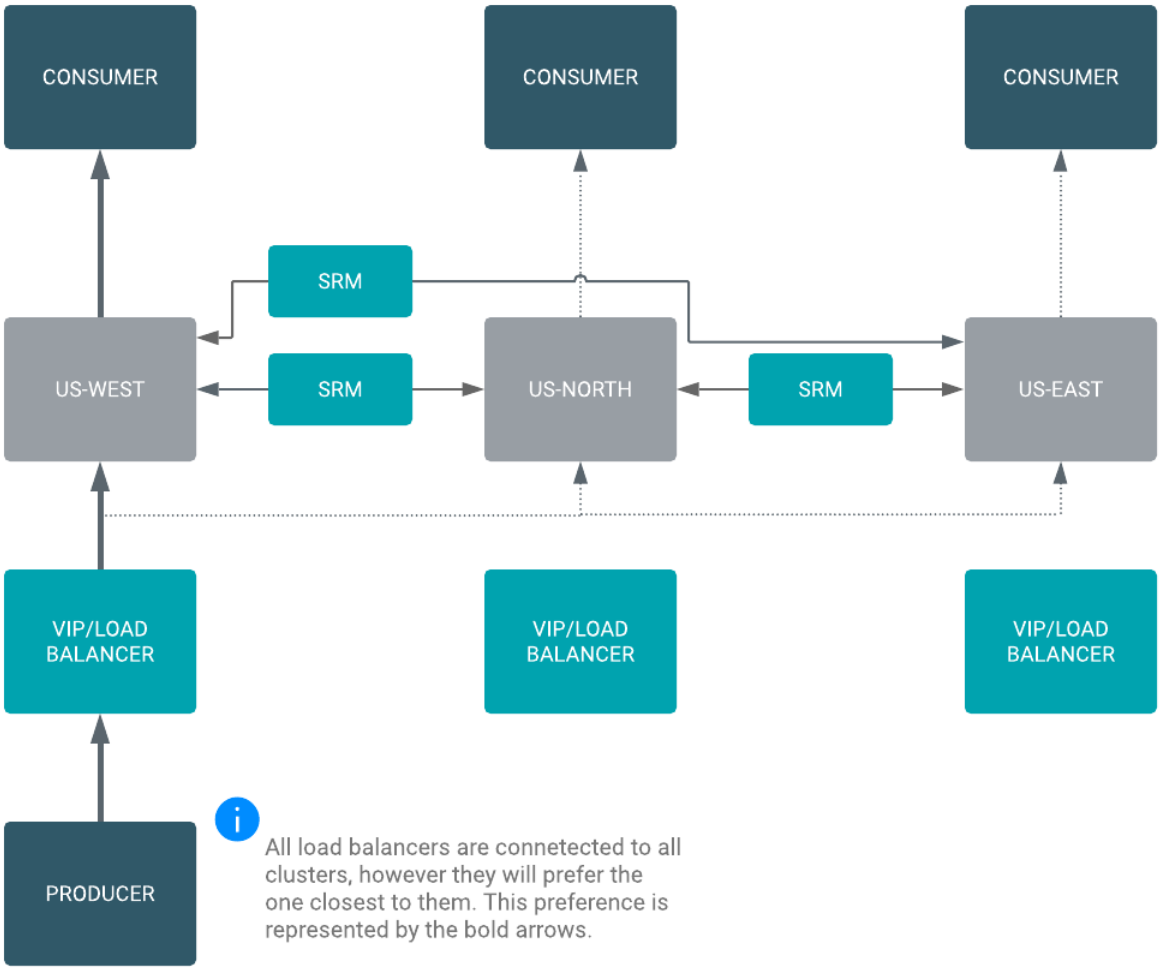

Cluster migration architectures

In addition to high availability use cases, Streams Replication Manager can be used for cluster migration and aggregation scenarios as well.

On-premise to cloud and Kafka version upgrade

If you have an on-premises Apache Kafka cluster that you want to migrate to the cloud, not only do you have to migrate consumers and producers, you also have to migrate topics and their messages to the new cloud-based cluster.

After you have set up replication through Streams Replication Manager, you only need to point your consumers to the new brokers before you can start processing messages from the cloud cluster. This approach ensures that the historical data kept in the on-premises Kafka cluster is migrated to the cloud cluster allowing you to replay messages directly from the cloud without having to go back to your on-premises cluster.

Figure 6: Cluster migration on-premise

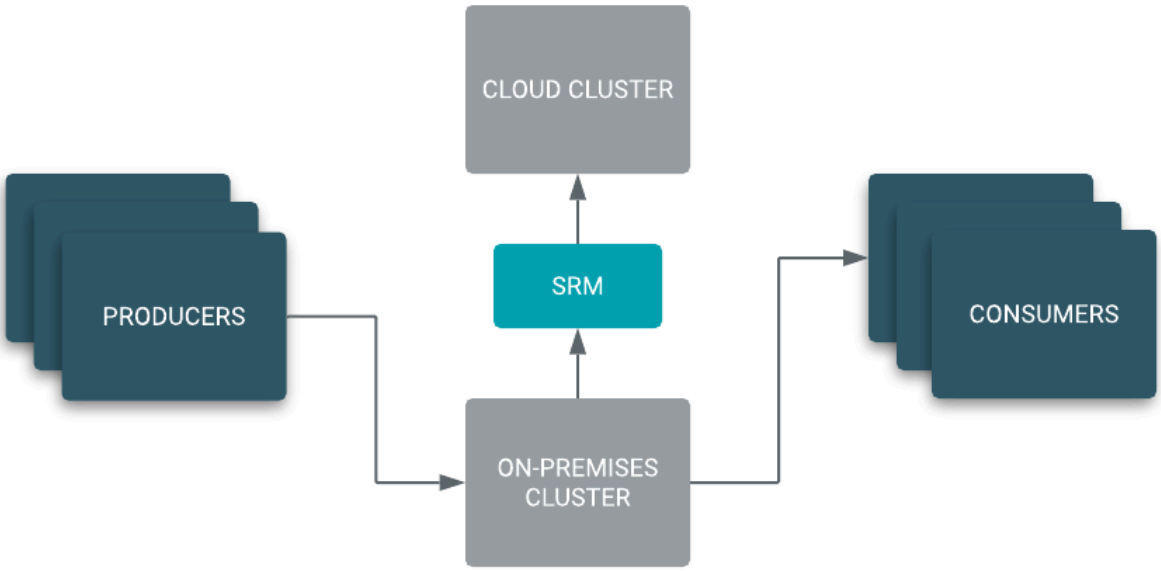

Aggregation for analytics

Streams Replication Manager can be used to aggregate data from multiple streaming pipelines, possibly across multiple data centers (DC), to run batch analytics jobs that provide a holistic view across the enterprise. For aggregation type architectures Cloudera recommends that you use the `IdentityReplicationPolicy`, which provides prefixless replication. When this replication policy is in use, remote (replicated) topics names are not altered. As a result, you can easily aggregate multiple topics from multiple clusters into one. For more information, see *Replication flows and replication policies* or *Enabling prefixless replication*.

Figure 7: Aggregation for Analytics

Related Information

Replication flows and replication policies

Enabling prefixless replication

Streams Replication Manager Architecture

Learn about the architecture of the Streams Replication Manager Driver and Service, which are two main components (roles) that make up Streams Replication Manager

Streams Replication Manager Driver

The Streams Replication Manager Driver is responsible for executing the configured replication work between Kafka clusters. This includes data replication, consumer group offset replication, and heartbeating.

The Streams Replication Manager Driver role is built on top of the Kafka Connect framework and utilizes a group of connectors to execute replication. While Kafka Connect is bound to a single Kafka cluster by design, the Streams Replication Manager Driver must connect to multiple Kafka clusters. The Streams Replication Manager Driver achieves this by wrapping multiple Connect workers in a single driver process. Specifically, for each possible replication between the configured clusters, the Streams Replication Manager Driver spins up a separate Connect worker. The Connect workers join a Connect group which is dedicated to a single replication in the target Kafka cluster. Afterward, each Connect group creates a single MirrorSourceConnector, a single MirrorCheckpointConnector and a single MirrorHeartbeatConnector. These Connectors and the task instances they generate are responsible for different aspects of the replication work.

When Streams Replication Manager Driver High Availability is in use, the Connect workers that are tied to a specific replication, but are running in different Streams Replication Manager Driver processes, coordinate through the Connect group protocol and balance the load.

Note: The following diagram is a visual representation of the Driver's architecture and showcases the different components of the Driver. In a running Driver, a single Connect worker can have a single, multiple, or even no Connectors or tasks assigned to it. As a result of this, the Connector and task distribution shown in this diagram does not fully reflect how Connectors and tasks are distributed in a running Driver. For more information on how Connectors and tasks are distributed, see Task architecture and load-balancing.

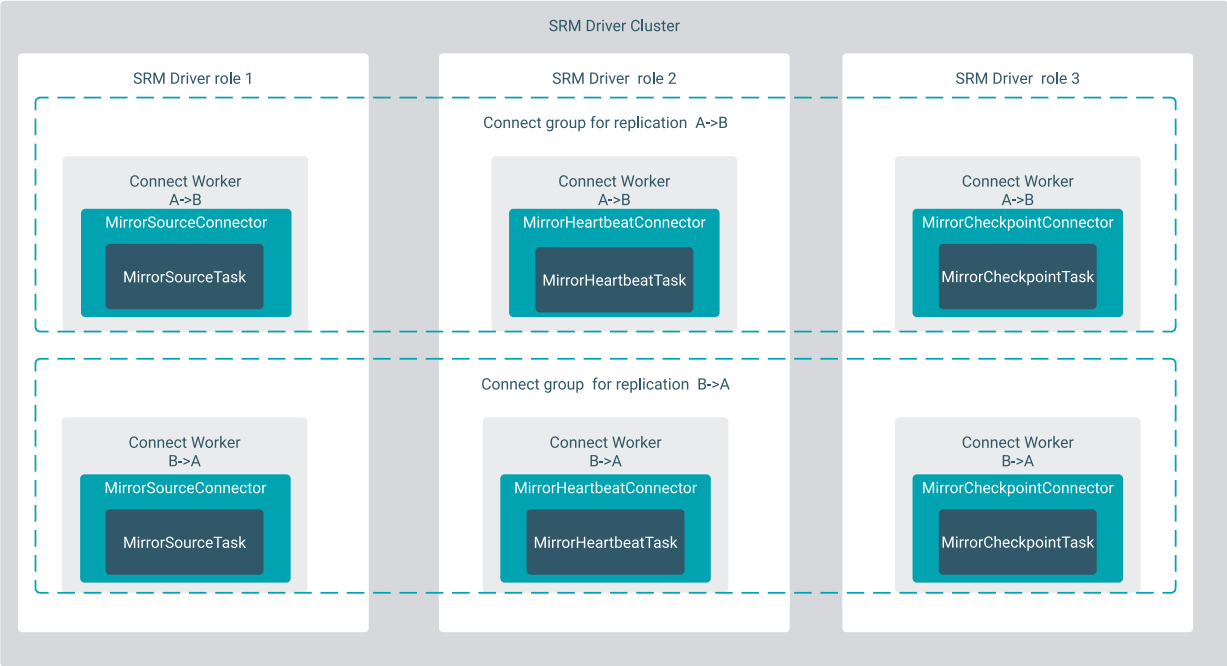

Connect workers

Learn about the Connect workers created by the Streams Replication Manager Driver.

The Streams Replication Manager Driver role wraps multiple Connect workers in its process. Each Connect worker corresponds to a possible replication. At startup, if a target is specified for the replication in the Streams Replication Manager Driver Target Cluster property, a Connect worker is created for each possible cluster pair based on the aliases present in Streams Replication Manager Cluster alias. This means that by default for each possible replication, there is a running Connect worker, regardless of whether the replication is enabled. For enabled replications, the Connect worker creates and manages all three Connectors (MirrorSourceConnector, MirrorCheckpointConnector, and MirrorHeartbeatConnector). For disabled replications, the Connect worker only creates and manages a MirrorHeartbeatConnector. The MirrorHeartbeatConnector is spun up to ensure that the heartbeats topic is created on all clusters which might be the source of a replication.

In certain scenarios however, some of the Connect workers set up for disabled replications are unnecessary. In such a case, you can configure Streams Replication Manager to fully deactivate disabled replications that are unutilized. This can be done by disabling heartbeat emission for these replications. If heartbeat emission is disabled, the Connect worker and MirrorHeartbeatConnector are not created. As a result, the replication associated with that worker is fully deactivated. This can help in minimizing the performance overhead caused by unnecessary Connect workers. For more information on how disable heartbeat emission, see *Configuring Streams Replication Manager Driver heartbeat emission*.

The Connect workers always coordinate using the target Kafka cluster. They join a Connect group which is dedicated to a specific replication. This means that even when there are multiple replications targeting the same cluster, the replications are managed and load-balanced separately, through dedicated Connect groups.

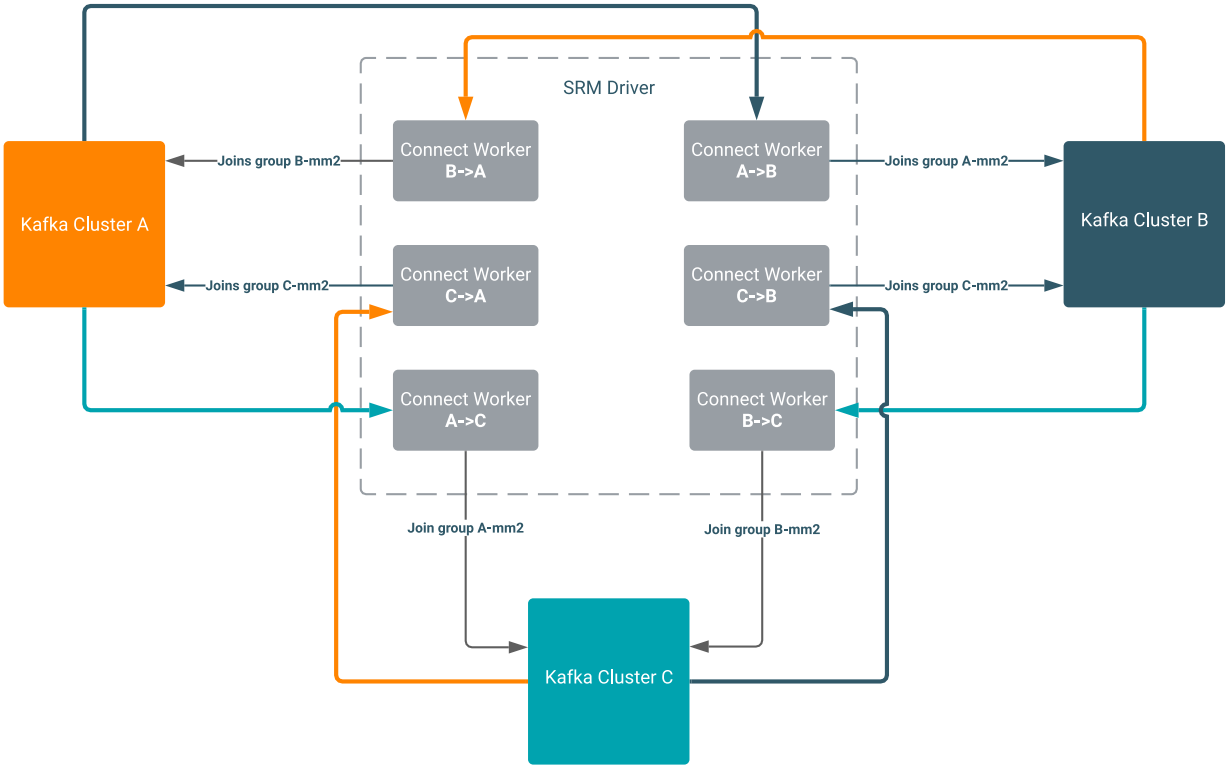

Connect internal topics

Streams Replication Manager creates a separate Connect cluster as well as three internal Kafka topics for each replication. The internal topics are used by the Connect clusters to store their state. These internal topics are all located in the target cluster of the replication. The topic names reference the source cluster alias.

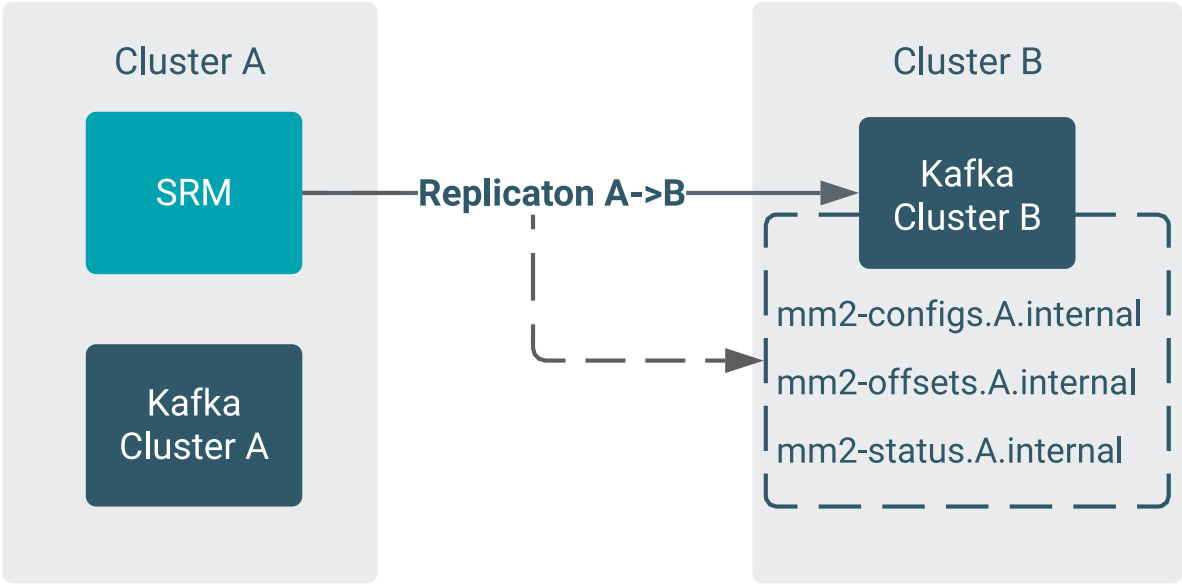

The three internal topics are as follows:

- mm2-configs.*[***SOURCE ALIAS***]*.internal

  Stores the Connector and Task configurations. Expected to be a single partition topic with cleanup.policy=compact. The records of the topic are generated based on Streams Replication Manager configuration at startup. Losing the data does not cause issues for Streams Replication Manager after the service is restarted.
- mm2-offsets.*[***SOURCE ALIAS***]*.internal

  Stores the committed source offsets of Streams Replication Manager. Streams Replication Manager uses this internal topic to track its progress in the replication of the source topic. Expected to be a multi-partition topic with cleanup.policy=compact. The records of the topic are crucial for tracking the state of replication. Losing the data causes Streams Replication Manager to restart the replication of source topics, which leads to data duplication in the target cluster.
- mm2-status.*[***SOURCE ALIAS***]*.internal

  Stores the current status of Connectors and Tasks. Expected to be a multi-partition topic with cleanup.policy=compact. The records of the topic are created for monitoring purposes and do not affect replication. Losing the data does not cause issues for Streams Replication Manager after the service is restarted.

All three internal topics are created by Streams Replication Manager at startup with the expected configurations. Cloudera does not recommend reconfiguring or deleting these topics manually. Doing so can cause issues with replication, which might result in data loss. However, if the topics are created with an incorrect configuration, manual reconfiguration is required. In a case like this, Streams Replication Manager must be stopped, and the topic properties must be updated with correct values. Updating topic properties can be done with the kafka-configs tool.

Related Information

kafka-configs

Configuring Streams Replication Manager Driver heartbeat emission

Connectors

Learn about the different Connector implementations created by the Connect workers of the Streams Replication Manager Driver.

The Streams Replication Manager Driver role employs multiple Connect workers to execute replication, one per each replication. Inside the Connect workers, three separate Connector implementations are utilized. These are the following:

- MirrorSourceConnector
- MirrorHeartbeatConnector
- MirrorCheckpointConnector

MirrorSourceConnector

The MirrorSourceConnector is responsible for replicating topics between the source and the target cluster. The topics to be replicated are defined by allow and deny lists, which can be manipulated using the srm-control tool.

In addition to replicating data, the MirrorSourceConnector also manages the offset sync topic in the source cluster. When production into the target cluster is successful, the mapping between the source offset and the target offset is written into an offset sync topic in the source cluster. The offset sync topic is used by the MirrorCheckpointConnector.

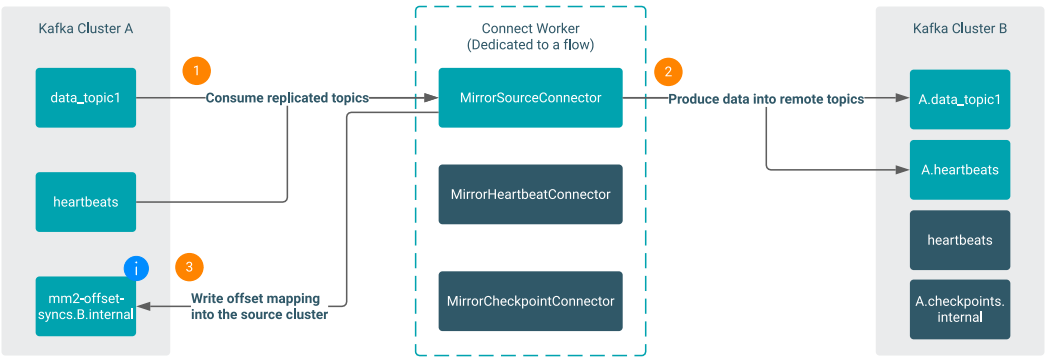

MirrorHeartbeatConnector

The MirrorHeartbeatConnector is responsible for creating the heartbeats topic in the target cluster. It also periodically produces heartbeats into the heartbeats topic. The purpose of the heartbeats topic is twofold:

- It ensures that a topic is available at all times on the source clusters. This way, a replication always has at least a single topic that it can pick up. This functions as a reliable smoke test for the replication.
- It is used by the Streams Replication Manager Service to discover configured replications.

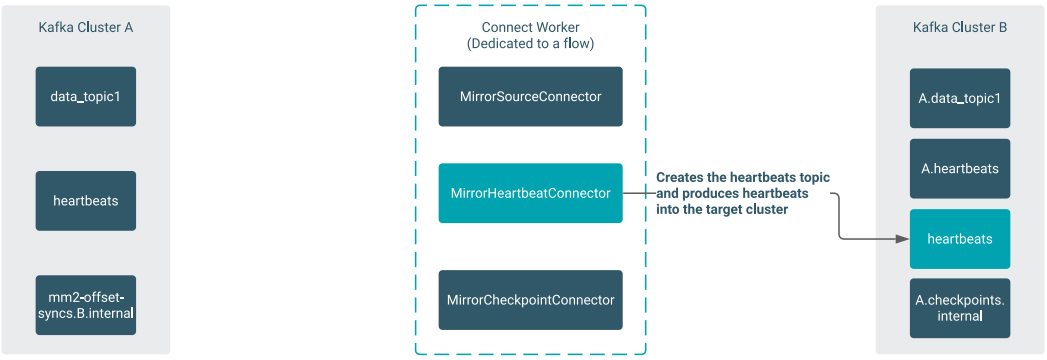

MirrorCheckpointConnector

The MirrorCheckpointConnector is responsible for replicating the committed group offsets between the source and target clusters. The offsets are calculated based on the offset sync topic managed by the MirrorSourceConnector. The offsets of the groups and the topic partitions that are replicated are defined by allow and deny lists, which can be manipulated using the srm-control tool.

The offsets are written into the checkpoint topic in the target cluster. The contents of the checkpoint topic can be exported with srm-control. Exported offsets can then be applied to the consumer groups in the target cluster with the kafka-consumer-groups Kafka tool.

In addition to writing the mapped offsets into the checkpoint topic, the connector is also capable of periodically applying the offsets to the consumer groups in the target cluster. This is done using the Kafka Admin API.

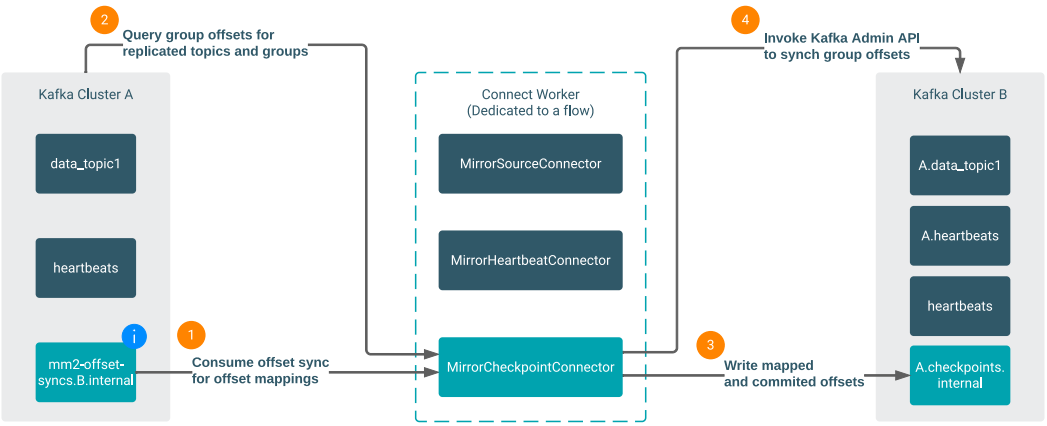

Task architecture and load-balancing

Learn how tasks are distributed and how load is balanced by the Streams Replication Manager Drivers that are running in the same cluster.

Streams Replication Manager Driver roles of the same cluster share the load of the replications among each other. They utilize the task load balancing of the Connect framework. Each Connector in a replication creates a set of task configurations. The number of tasks depend on the replicated topic partitions, groups, and the Tasks Max configuration of Streams Replication Manager. These task instances provide a way to parallelize replication work. After the Connectors create the task configurations, the Connect framework distributes the tasks between the Streams Replication Manager Drivers. More specifically, between the Connect workers running in the Streams Replication Manager Drivers corresponding to a specific replication flow. The task instances are as follows:

MirrorSourceTask

The MirrorSourceTask is created by the MirrorSourceConnector. It is responsible for executing data replication. A MirrorSourceTask wraps a consumer and a producer instance. The consumer instance is managed by the task. The producer instance is managed by the Connect framework.

Each task receives its assignment from the MirrorSourceConnector as a list of topic partitions. These are assigned to the consumer. The fetched records are then forwarded to the producer. The target topic name is generated based on the ReplicationPolicy.

The number of MirrorSourceTasks created in a replication is based on the following formula:

```
min(tasks.max, source_topic_partition_count)
```

In this formula, tasks.max corresponds to what Tasks Max is set to in Streams Replication Manager. The source_topic_partition_count is the number of replicated topic partitions, which is based on the topic allow and deny lists configured with the srm-control tool.

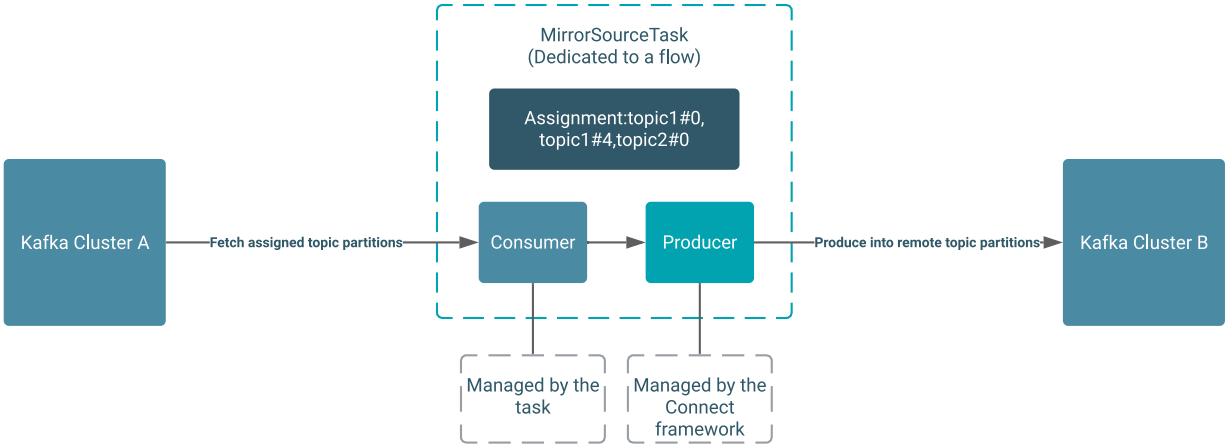

MirrorCheckpointTask

The MirrorCheckpointTask is created by the MirrorCheckpointConnector. It responsible for executing the consumer group offset sync. A MirrorCheckpointTask wraps an admin client and a producer instance. The admin client is managed by the task. The producer instance is managed by the Connect framework.

Each task receives its assignment from the MirrorCheckpointConnector as a list of consumer groups. The assigned consumer group offsets are periodically queried for the replicated topic partitions through the admin client, and are written into the target cluster's checkpoints topic. The replicated offsets can also be applied to the consumer groups in the target cluster by setting sync.group.offsets.enabled to true.

The number of MirrorCheckpointTasks created in a replication flow is based on the following formula:

```
min(tasks.max, source_consumer_group_count)
```

In this formula, tasks.max corresponds to what Tasks Max is set to in Streams Replication Manager. The source_consumer_group_count is the number of replicated consumer groups, which is based on the group allow and deny lists configured with the srm-control tool.

MirrorHeartbeatTask

The MirrorHeartbeatTask is created by the MirrorHeartbeatConnector. It is responsible for producing heartbeats into the target cluster's heartbeats topic. A MirrorHeartbeatTask wraps a producer instance that is managed by the Connect framework. In each replication, there is a single MirrorHeartbeatTask instance.

Task load balancing

Multiple Connect workers participate in each replication. This typically means that each Streams Replication Manager Driver of the cluster has one Connect worker dedicated to a specific replication.

After the Connect group is formed by the Connect workers, the Connectors are instantiated and task configurations are generated. This results in three Connectors, a number of MirrorSourceTasks and MirrorCheckpointTasks, as well as a MirrorHeartbeatTask. For a newly formed group, these Connectors and task instances are assigned to the Connect workers in a round robin fashion. When the group already has an existing assignment, and there is membership change (for example, a Connect worker joins or leaves the group), the Connectors and tasks are assigned in a cooperative and incremental manner. This allows for the majority of the work to continue without interruption.

Unless there are no changes made to the task configurations, meaning that no new topics and groups are added to the replication, the task definitions stay the same. However, based on Connect group membership changes, the tasks can be moved between workers. With classic Connect, these tasks can be managed separately through an admin API, but in the case of Streams Replication Manager, this API is not available, and tasks can onle restarted by restarting the Streams Replication Manager Driver cluster.

Driver inter-node coordination

Learn about how different Streams Replication Manager Driver instances communicate with each other.

As per the Connect framework, members of a Connect group get updates from the group leader through Kafka, using the Connect group protocol. In some cases, followers also need to be able to communicate with the group leader. For this purpose, the Connect framework introduced a REST API. This API allows followers to push notifications to the leader. This step is necessary when dynamic configuration changes occur in a Connector. In case of Streams Replication Manager, configuration changes happen when new topics and groups are added to the replication. This can happen in two ways:

- New topics and groups appear in the source cluster that conform to the allow and deny lists of the replication.
- Allow and deny lists are added using the srm-control tool.

To support this necessary channel of communication inside Streams Replication Manager, each Driver spins up a REST server per replication flow. With this implementation, each nested Connect worker is fully functional, as per the original design of Connect. These replication specific REST servers are configurable, for more information, see *Configuring replication specific Kafka Connect REST servers*.

Related Information

Configuring replication specific Kafka Connect REST servers

Streams Replication Manager Service

The Streams Replication Manager Service is responsible for processing the metrics produced by the Streams Replication Manager Drivers. Additionally, it provides a queryable REST API that you can use to monitor and track replications.

Streams Replication Manager Driver roles produce raw metrics into the target Kafka clusters. Each replication has a separate raw metric topic. Streams Replication Manager Service roles run a Kafka Streams application internally, which aggregates the raw metrics. The Streams application ensures that the metrics processing work is load balanced between the Streams Replication Manager Service instances of the same cluster, and that the members of the cluster can coordinate when serving REST API queries.

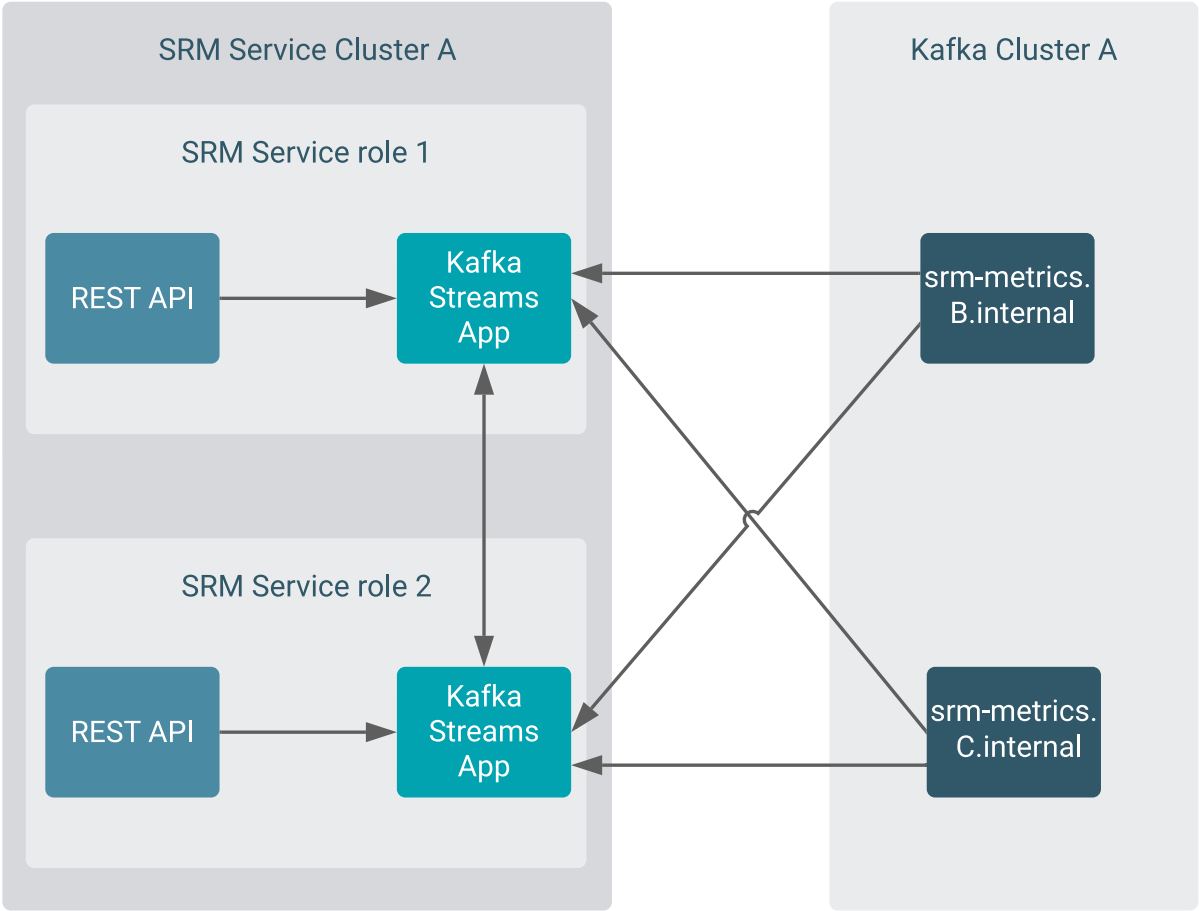

As a result of this architecture, the Streams Replication Manager Service can only report on replication flows targeting the Kafka cluster which is targeted by the Streams Replication Manager Service.

Multi cluster targeting

The Streams Replication Manager Service is capable of targeting multiple Kafka clusters. In this case, a Streams Replication Manager Service instance spins up a Kafka Streams application for each target Kafka cluster. This way, the Streams Replication Manager Service is able to report on all replication flows targeting any of the configured target clusters.

Remote Querying

A Streams Replication Manager Service cluster can be configured to target and gather metrics from multiple clusters (multi cluster targeting). However, a setup like this can result in heavily loaded Service roles, which might not be suitable for your deployment. To circumvent this issue, the Streams Replication Manager Service is capable of connecting to and communicating with other, remote Streams Replication Manager Services to fetch their metrics. This is called Remote Querying. Using Remote Querying makes it possible to designate a Streams Replication Manager Service cluster in your deployment to act as a monitoring gateway. The designated Streams Replication Manager Service cluster then can be used to monitor all clusters and replications in your deployment. For more information, see *Remote Querying*.

Using the Streams Replication Manager Service REST API

The Streams Replication Manager Service offers a Swagger UI for exploring and querying the REST API. For a user-friendly UI solution, Streams Messaging Manager can be integrated with Streams Replication Manager.

When Streams Messaging Manager is configured to connect to Streams Replication Manager, the **Replications** page becomes available on the Streams Messaging Manager UI. This page displays the information available on the Streams Replication Manager Service REST API. For more information on how you can integrate the two services, see *Integrating Streams Replication Manager with Streams Messaging Manager*.

Related Information

Integrating Streams Replication Manager with Streams Messaging Manager

Remote Querying

Remote Querying in Streams Replication Manager refers to the Streams Replication Manager Service's capability of querying other, remote Streams Replication Manager Services to fetch the remote cluster replication metrics. This allows users to monitor all replications of a deployment that has multiple instances of Streams Replication Manager through a single Streams Replication Manager Service.

Overview

The Streams Replication Manager Service role gathers, aggregates, and exposes metrics related to cluster replications. While a single cluster of Streams Replication Manager Service roles (Streams Replication Manager Service cluster) can be configured to target and gather metrics from multiple clusters, a setup like this can result in heavily loaded Service roles, which might not be suitable for your deployment. Instead, you can choose to have a single Streams Replication Manager Service cluster connect to other, remote Streams Replication Manager Service clusters and fetch metrics from them. This is called Remote Querying.

Using Remote Querying makes it possible to designate a Streams Replication Manager Service cluster in your deployment to act as a monitoring gateway. The designated Streams Replication Manager Service cluster can then be used to monitor all clusters and replications in your deployment. This way, a single Streams Replication Manager Service cluster can provide you with information on all clusters and replications. In addition, if you have Streams Messaging Manager integrated with the Streams Replication Manager Service cluster acting as the gateway, information regarding all replications will be available in that Streams Messaging Manager instance's UI.

How it works

Remote Streams Replication Manager Service clusters are discovered through Kafka. Streams Replication Manager Service clusters advertise themselves through their target Kafka cluster by writing data into a heartbeats topic. The information advertised is the Service role's protocol, host, port and root API path. When Remote Querying is configured for a specific Streams Replication Manager Service cluster, that Streams Replication Manager Service cluster connects to the specified external Kafka clusters, consumes the heartbeats topics, and based on the advertised information, discovers the remote Streams Replication Manager Service clusters.

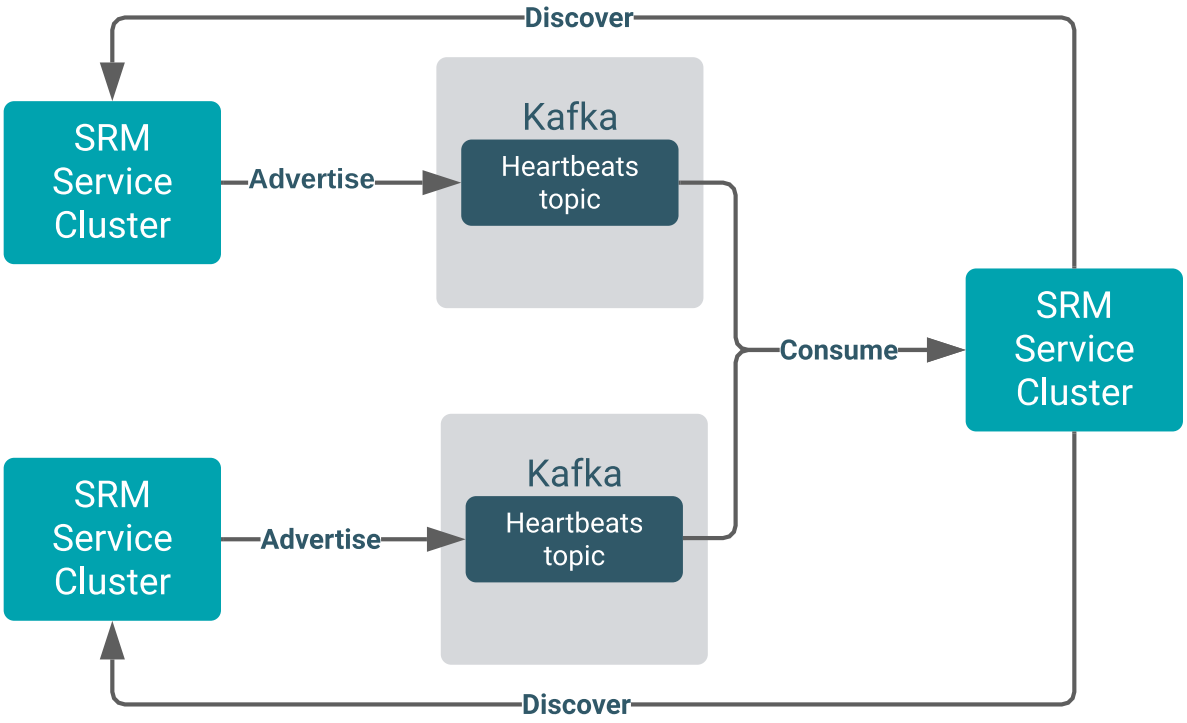

Following discovery, a Streams Replication Manager Service cluster can cooperate with its remote counterparts and fetch the metrics related to remote replications. These metrics can then be queried using the Streams Replication Manager REST API, or viewed on the Streams Messaging Manager UI.

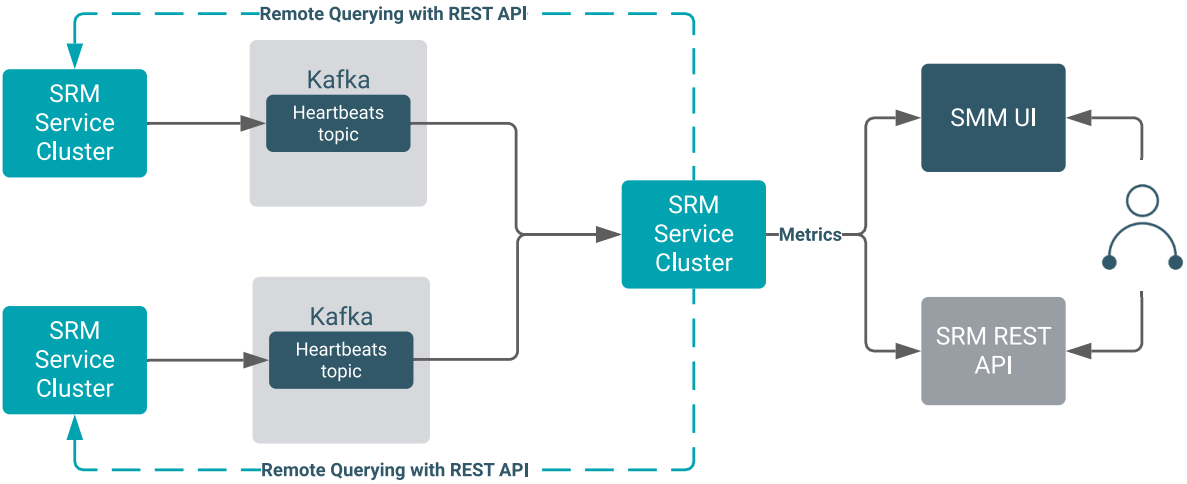

Data locality

When the feature is enabled, all metrics are still processed locally. Each Streams Replication Manager Service cluster processes the metrics of its target Kafka cluster only. The Streams Replication Manager Service cluster configured to be the gateway does not take over and process the metrics of the remote Streams Replication Manager Service clusters. It only communicates with the remote Streams Replication Manager Service clusters to fetch and then serve their metrics.

However, because metrics processing remains local, when you enable the feature, additional traffic is generated between the gateway and remote Streams Replication Manager Service clusters. It is important that you take this into consideration especially if one or more of your Streams Replication Manager installations are located in a public

cloud environment. For more information on the amount of data generated, see *Streams Replication Manager Service data traffic reference*.

Remote Querying example

Consider the following deployment:

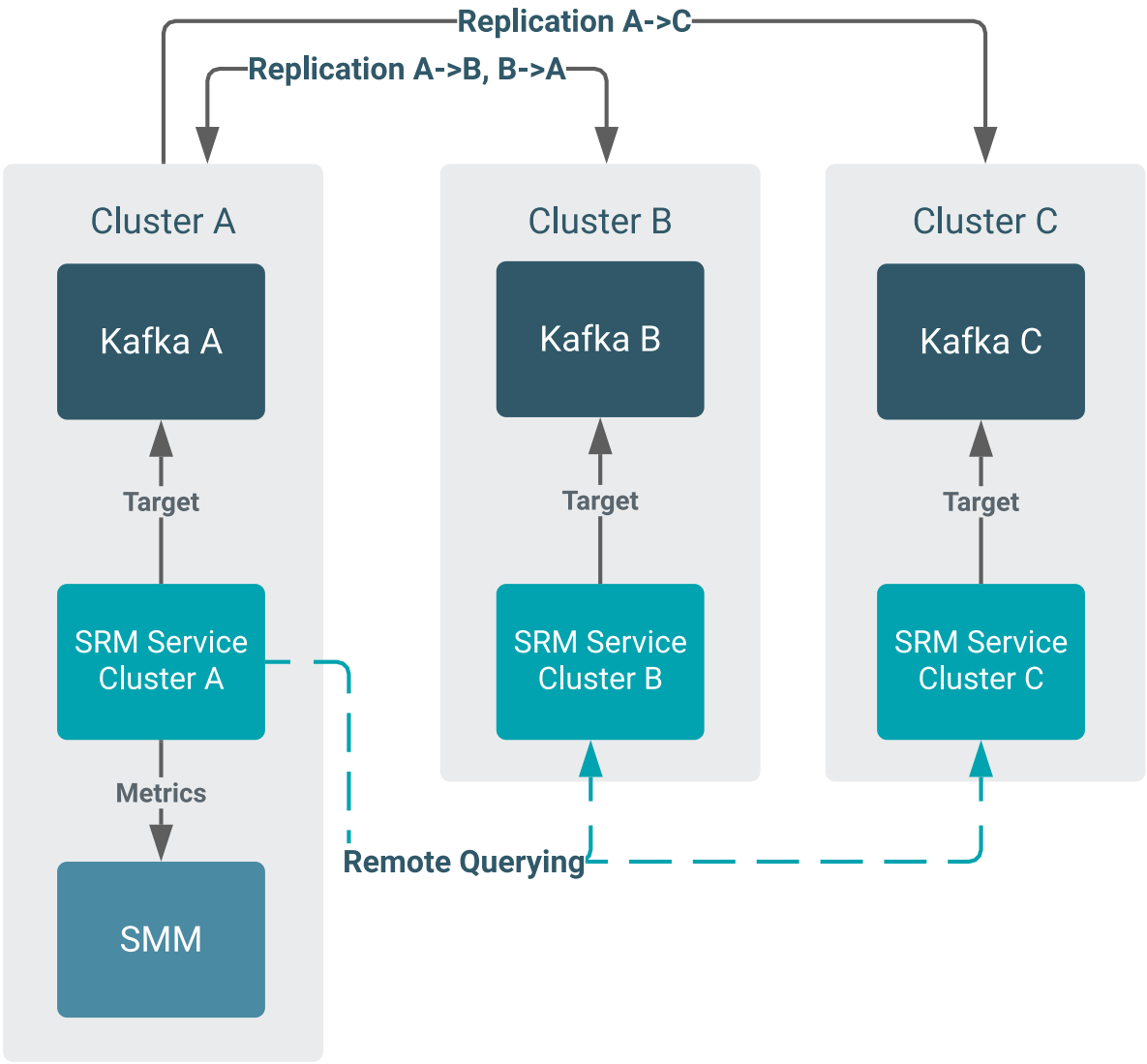

There are three clusters, cluster A, B, and C. All clusters have Kafka and Streams Replication Manager deployed on them. Cluster A has Streams Messaging Manager installed as well. Bidirectional replication is happening between Cluster A and Cluster B. Additionally, unidirectional replication is set up from Cluster A to Cluster C. Each Streams Replication Manager Service cluster is targeting its co-located Kafka.

In this scenario, Remote Querying is configured and enabled for Streams Replication Manager Service cluster A. This enables you to monitor all replications in the deployment using Streams Replication Manager Service cluster A. Additionally, information regarding all replications can be viewed in Streams Messaging Manager deployed in Cluster A.

Without Remote Querying, Streams Messaging Manager would only be able to display replications that are targeting Cluster A. If you wanted to monitor any other replications, you would need to manually query each Streams Replication Manager Service cluster separately using the REST API, or set up separate instances of Streams Messaging Manager on each of the clusters.

Related Information

Streams Replication Manager Service data traffic reference

Streams Replication Manager monitoring and metrics

Streams Replication Manager metrics can give you insight into the status of Streams Replication Manager and the replication process as a whole. Learn how Streams Replication Manager metrics are collected, processed, and exposed on the Streams Replication Manager Service REST API.

Important: While Streams Replication Manager monitoring using the Streams Replication Manager Service and its REST API is supported when the `IdentityReplicationPolicy` is in use, a number of limitations apply, see Streams Replication Manager replication flows and replication policies on page 28.

Streams Replication Manager consists of two main components (roles) that are each responsible for different tasks related to data replication. These are the Streams Replication Manager Driver and Streams Replication Manager Service roles.

The Streams Replication Manager Driver role is based on the Kafka Connect Framework and utilizes Connect connectors and connector tasks to replicate Kafka data. These connectors and tasks produce raw metrics that describe the replication process. The Streams Replication Manager service is responsible for collecting, processing, and exposing these metrics. The metrics can be queried using the Streams Replication Manager Service REST API or viewed in Streams Messaging Manager on the **Replications** page.

Note: For a more in depth look at the internal architecture of Streams Replication Manager as well as how the Driver and Service roles work, see Streams Replication Manager Architecture.

The following sections provide an overview on how raw Streams Replication Manager metrics are collected, how they are processed into metrics that can be used for monitoring.

Raw metric collection

Because the Streams Replication Manager Driver is built on top of the Kafka Connect framework, the connectors and connector tasks of the Driver produce raw Kafka Connect metrics. Although there are many metrics that exist within the Connect framework, Streams Replication Manager only collects the following raw metrics:

- Status metrics are collected from connectors.
- Status, byte-rate, replication-latency-ms, checkpoint-latency-ms, and replication-records-lag metrics are collected from connector tasks.

All of these metrics have two intrinsic properties. They have an origin and a subject. The origin denotes where the metric is collected from. The subject denotes what entity the metric describes. Based on these properties, the raw metrics can be organized into two groups. Connector-originated metrics, which describe the replication flows, and task-originated metrics, which describe the replication of individual topic partitions.

Metrics collected from connectors are connector-originated. Connectors within the framework of Streams Replication Manager represent the replications (also referred to as replication flows). Therefore, the subject of connector-originated metrics are the replications. Because the subject is the replication flow, connector-originated metrics have a wide scope and describe the replication process from a high level. Think of a two cluster replication scenario. The replication flow acts as a bridge between the two clusters. Connector-originated metrics describe the bridge, or in other words, the replication.

Metrics collected from connector tasks are task-originated. Tasks within the framework of Streams Replication Manager represent the actual replication work that is performed. The tasks are the processes that replicate the messages between the source and target clusters from the topic partitions. As a result, the subject of task-originated metrics is the topic partition. Compared to connector-originated metrics, task-originated metrics have a restricted scope and describe the replication process at a low level.

The raw metrics collected by Streams Replication Manager do not provide good insights on the replication process as a whole and cannot be used as is for monitoring. For example, even though connector-originated metrics describe the replications, a replication is always made up of multiple connector instances (or workers), which may even run on different nodes. Therefore, the raw metrics of a single connector instance (or worker) only provide partial data on the replication. Because of this, raw metrics are not exposed by the Streams Replication Manager service. Instead, Streams Replication Manager processes and aggregates them before exposing them for consumption on the REST API.

Metrics processing and aggregation

Evaluating a single metric is rarely useful. For example, looking at the transmitted bytes of an ongoing replication for a single topic partition in the last 30 seconds does not provide good insight on what is actually happening with the topic or the replication. To provide metrics that describe the replication process in a meaningful way, statistics derived from the raw data are required. In other words, the raw metric data must be processed before it can be exposed for consumption. Creating statistics involves gathering data, in this case raw metrics, and deriving their characteristics through processing and aggregation. The result of this are the metrics that Streams Replication Manager exposes on its REST API.

When processing raw data, Streams Replication Manager uses a combination of the following two strategies:

Streams Replication Manager collects and processes metrics over a time period

This is done because having access to the momentary status of a single entity does not provide real value. On the other hand, data gathered over time is descriptive and provides a good filter for temporary or short lived performance issues. Streams Replication Manager gathers data over time using sliding time-window averaging in the case of numeric metrics.

Streams Replication Manager collects and processes metrics across different entities

When monitoring Streams Replication Manager, the entities that you want to monitor are the topics and replications. This is because it is the metrics of the topics and replications that can give you tangible information on the status of Streams Replication Manager. However, there are no raw metrics available that directly describe topics. Additionally, only status metrics are available for the replications; raw metrics measuring byte rate, latency, or lag are not. Nonetheless, Streams Replication Manager can still provide metrics on the topics and replications by way of aggregating available raw metrics across different sets of entities.

Specifically, Streams Replication Manager gathers the raw metrics that describe the topic partitions (the task-originated raw metrics). Next, it aggregates the data over all topic partitions that belong to a topic. That is, over the topic. The resulting data are the metrics that describe the topic.

Metrics that describe the replications are produced in a similar way. Streams Replication Manager gathers the raw partition metrics and aggregates them over a certain replication. To be precise, over all topics that take part in that replication. The resulting data are the metrics that describe the replication.

All resulting metrics reported for the entities include the statistical minimum, maximum, average, and sum of the sample.

Note: Not all statistics are meaningful. For example, the sum of replication speed over all topic partitions does not give good insight on the overall performance of the replication.

Based on how the data is processed, the exposed metrics can be categorized into the following levels:

Cluster-level metrics

Cluster-level metrics are aggregated over a certain time frame and (contrary to the name) over a certain **replication**. These metrics reflect the overall performance of a certain replication.

Topic-level metrics

Topic-level metrics are aggregated over a certain time frame and over a topic. They reflect the overall progress and status of a replication with regards to a single topic.

Available numeric metrics

Table 1: Streams Replication Manager numeric metrics

name	level	description
byte-rate	cluster	The average number of bytes that are being replicated per second, aggregated over a replication
byte-rate	topic	The average number of bytes that are being replicated per second, aggregated over a topic
replication-latency-ms	cluster	Time it takes records to replicate from source to target cluster, aggregated over a replication
replication-latency-ms	topic	Time it takes records to replicate from source to target cluster, aggregated over a topic
checkpoint-latency-ms	cluster	Time it takes consumer group offsets to replicate from source to target cluster, aggregated over a replication
checkpoint-latency-ms	topic	Time it takes consumer group offsets to replicate from source to target cluster, aggregated over a topic
replication-records-lag	cluster	The number of “to be replicated” messages (difference between the source offset of the last replicated message and the log end offset of the source) aggregated over a replication. The “sum” statistics of the metric describes the overall lag of the replication with respect to a replication flow.
replication-records-lag	topic	The number of “to be replicated” messages (difference between the source offset of the last replicated message and the log end offset of the source) aggregated over a topic. The “sum” statistics of the metric describes the overall lag of the replication in a topic.

Notes about calculation of metrics

When monitoring replication and checkpoint latency metrics, keep in mind the following regarding their calculation.

replication-latency-ms

Replication latency is the difference between the record timestamp of the message as saved in the source cluster and the time of the commit in the performing task. Because the source and target clusters in a replication are two distinct clusters, the record timestamp of the message and the commit time will come from two different clusters. As a result, the clocks of the two clusters must be in sync. Otherwise, Streams Replication Manager will calculate the latency incorrectly, which can lead to negative or other misleading values being reported

checkpoint-latency-ms

Checkpoint latency is the difference between the time when the data necessary for creating a checkpoint is fetched (this includes consumer group offsets) and when the checkpoint is eventually committed.

Status metrics

Unlike raw metrics related to latency, record lag, or byte rate, raw status metrics are not numeric. These are simple strings of text that describe the state of the connector or task. For example, connectors will report that they are paused, running, or stopped. Connector tasks will report that they are running, stopped, or failed.

Similarly to other raw metrics, status metrics are not exposed directly by Streams Replication Manager and are also processed and aggregated. Streams Replication Manager uses status metrics as input to create detailed status messages that describe the overall status of a replication and as input for health checks.

The status messages that Streams Replication Manager produces are more complex than the raw metrics used for input. They include the reason and cause of why a specific replication is in the reported state. These messages can be viewed on the **Replications** page of the Streams Messaging Manager UI.

Related Information

Streams Replication Manager Service REST API

Streams Replication Manager Service REST API Reference

Monitoring Kafka cluster replications using Streams Messaging Manager

Streams Replication Manager Service REST API

The Streams Replication Manager Service REST API can be used to monitor your replication workloads. Learn about the endpoints, endpoint versions, and the data that the endpoints expose.

The processed, aggregated, and in part derived metrics produced by the Streams Replication Manager Service are available for consumption through the various endpoints of the Streams Replication Manager Service REST API. Exposed metrics are aggregated over a predefined set of samples and a sliding time window. In addition to aggregated values, the endpoints also return the statistical minimum, maximum, average, and sum of the corresponding sample.

Accessing the API and its endpoints

If you have access to a running instance of Streams Replication Manager, you can find a comprehensive list of the endpoints, their descriptions, as well as example responses for all endpoints on the Streams Replication Manager Service REST API's Swagger UI.

To access the Swagger UI, select the Streams Replication Manager service in Cloudera Manager and go to Web UI SRM Service Swagger UI *[***CLUSTER NAME***]* . Alternatively, you can also view the *Streams Replication Manager REST API Reference* for a comprehensive list of the endpoints.

Tip: If you are accessing the Swagger UI on a running cluster, you can make calls to each endpoint directly on the UI using the Try it out option.

Endpoint versions

REST API endpoints are grouped based on their version. There exists V2 and V1 (legacy) endpoints. Both V1 and V2 endpoints are similar in content and role. Additionally, there is considerable overlap in functionality. However, V1 is provided only for backward compatibility. As a result, Cloudera recommends that you always use V2 endpoints when monitoring Streams Replication Manager. V2 is more sophisticated, offers more endpoints, and allows the target clusters to be specified by the user. The same is not true for V1 endpoints where the target cluster is fixed. It is always the target cluster of the Streams Replication Manager Service instance that serves the REST call.

Internal and non-internal endpoints

In addition to version, the endpoints are also grouped based on whether they are internal or not. Cloudera recommends that you always use non-internal endpoints for monitoring. Internal endpoints are reserved for internal use by Streams Replication Manager. For example, if the Remote Querying feature is set up in your deployment, the Streams Replication Manager Service instances in the deployment use the internal endpoints to query each other's metrics.

Streams Messaging Manager integration

The endpoints made available on the API can also be used by services that integrate with Streams Replication Manager. Streams Messaging Manager, for example, queries the replication metrics using the endpoints and makes the data it collected available for consumption on the **Replications** page of the Streams Messaging Manager UI. In other words, the replication insights that you can view in Streams Messaging Manager are based on the metrics that Streams Replication Manager produces. For more information on how Streams Messaging Manager integrates with Streams Replication Manager as well as the **Replications** page, see *Monitoring Kafka Cluster Replications using Streams Messaging Manager*.

Related Information

Streams Replication Manager Service REST API Reference
Monitoring Kafka cluster replications using Streams Messaging Manager
Streams Replication Manager monitoring and metrics
Streams Replication Manager Service architecture
Remote Querying

Streams Replication Manager replication flows and replication policies

Get familiar with the concepts of replication flows and replication policies. Additionally, learn about the default replication policies shipped with Streams Replication Manager as well as the most commonly used types of replication flows.

Replication flows

Replication involves sending records and consumer group checkpoints from a source cluster to a target cluster. In Streams Replication Manager, a replication flow (also referred to as a replication or flow) specifies a source and target cluster pair, the direction in which data is flowing and the topics that are being replicated. Source and target cluster pairs can be specified in Cloudera Manager; they are notated source->target. Initially, when source->target pairs are set up they are considered inactive, as no data is being replicated between them. To start replication, you must specify which topics to replicate with the srm-control command line tool.

Replication in Streams Replication Manager is configured independently for each source->target cluster pair. Moreover, configuration is done on a per topic basis. This means that each topic in a source cluster can have a different direction or target that it is being replicated to. A set of topics in the source cluster can be replicated to multiple target clusters while others are being replicated to only one target cluster. This allows users to set up powerful, topic specific replication flows.

A basic example of a replication flow is when topics are being sent from one cluster to another cluster in a different geographical location. Note that in this example there is only one replication or source->target pair. Moreover, only one of the two topics on the source cluster are being replicated to the target cluster.

Figure 8: Simple Replication Flow Example

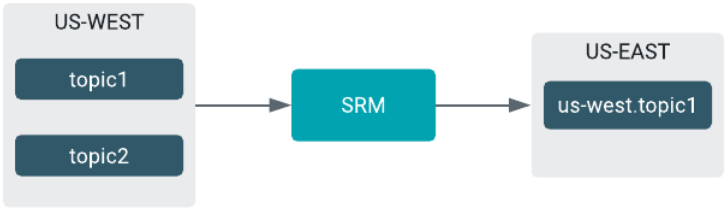

Replication policies

In any replication flow, the selected source topics are replicated to remote (replicated) topics on the target cluster. The basic rules of how these topics are replicated is defined by the replication policy that is used by Streams Replication Manager. Cloudera Runtime includes two replication policies by default. These are the `DefaultReplicationPolicy` and `IdentityReplicationPolicy`. The main difference between the two policies is how they name remote topics. In addition to these policies, Streams Replication Manager also supports the use of custom replication policies.

DefaultReplicationPolicy

The `DefaultReplicationPolicy` is the default and Cloudera-recommended replication policy. This is because the `DefaultReplicationPolicy` is capable of automatically detecting replication loops and supports all monitoring features provided by the Streams Replication Manager Service.

This policy prefixes the remote topic's name with the cluster name (alias) of the source topics. For example, the topic1 topic from the us-west source cluster creates the us-west.topic1 remote topic on the target cluster.

Figure 9: Simple Replication Flow Example

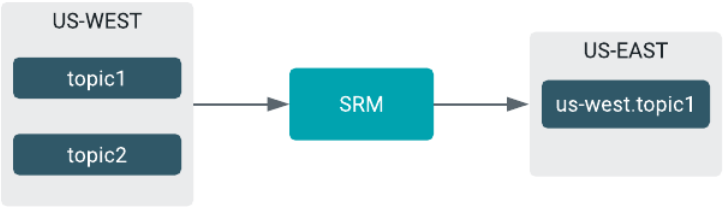

If a remote topic is also replicated, the remote topic references all source and target clusters. The prefix in the name will start with the cluster closest to the final target cluster. For example, the topi c1 topic replicated from the us-west source cluster to the us-east cluster and then to the eu-west cluster will be named us-east.us-west.topic1.

Figure 10: Complex Replication Flow Example

Tip: You might want to have your Kafka consumers read messages from both source and remote topics simultaneously. To achieve this when the `DefaultReplicationPolicy` is in use, Kafka consumers should include a wildcard topic name pattern. For example, suppose that you have a cluster, us-east. This cluster contains a source topic, topic1, and a remote topic from another cluster, us-west.topic1. In such a case, you can use the (.*\.)?topic1 pattern, which matches both topics.

IdentityReplicationPolicy

The `IdentityReplicationPolicy` does not change the names of remote topics. When this policy is in use, topics retain the same name on both source and target clusters. For example, the topic1 topic from the us-west source cluster creates the topic1 remote topic on the target cluster.

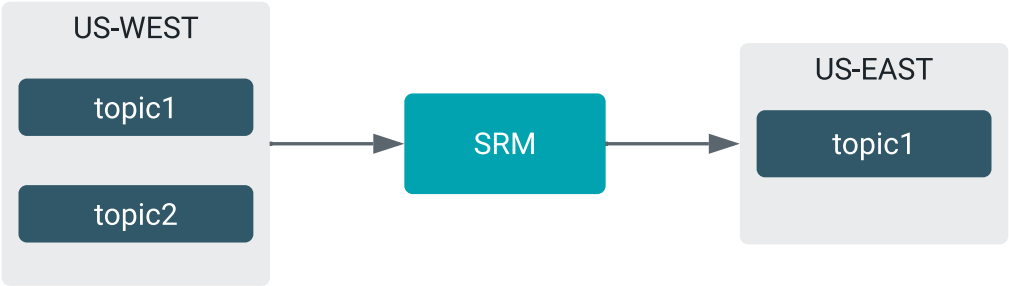

This type of replication is also referred to as prefixless replication. This replication policy is recommended for deployments where Streams Replication Manager is used to aggregate data from multiple streaming pipelines. Alternatively, this replication policy can also be used if the deployment requires MirrorMaker1 (MM1) compatible replication.

The `IdentityReplicationPolicy` has the following limitations:

- Replication loop detection is not supported. As a result, you must ensure that topics are **not** replicated in a loop between your source and target clusters.
- The /v2/topic-metrics/{target}/{downstreamTopic}/{metric} endpoint of Streams Replication Manager Service v2 API does not work properly with prefixless replication. Use the /v2/topic-metrics/{source}/{target}/{upstreamTopic}/{metric} endpoint instead.
- The replication metric graphs shown on the **Topic Details** page of the Streams Messaging Manager UI do not work with prefixless replication.

For more information on how to configure your Streams Replication Manager service to use this policy, see *Enabling prefixless replication*.

Custom replication policies

If neither the `DefaultReplicationPolicy` or `IdentityReplicationPolicy` fit your business requirements, you can develop your own replication policy implementation. Developing and using your own replication policy enables you to gain full control over how Streams Replication Manager replicates data. For more information, see the `ReplicationPolicy` interface in the Kafka Java documentation

Important: Cloudera provides limited support for deployments that use custom replication policies. Additionally, some key features including replication monitoring with the Streams Replication Manager Service will not work if a custom replication policy is in use. Whenever possible, Cloudera recommends that you use the `DefaultReplicationPolicy`.

Bidirectional replication flows

Streams Replication Manager supports bidirectional replication flows. A bidirectional replication flow is a setup where the topics of two or more clusters are mutually replicated. Records sent to one cluster are replicated to the other and the other way around. You can configure any number of clusters in this way.

A common issue with a bidirectional setup is that you can easily create replication loops. A replication loop is when a topic from one cluster is replicated to another and that same topic is replicated back to the source. The `DefaultReplicationPolicy` is capable of detecting replication loops. If this policy is in use, Streams Replication Manager will never replicate records in an infinite loop. As a result, you can freely use regex patterns in your allow and deny lists without having to worry about accidental replication loops.

The `IdentityReplicationPolicy`, on the other hand, cannot detect replication loops. As a result, if using this policy, you must ensure that topic allow and deny lists are correctly set up without any loops in the replication.

For example, consider the following bidirectional setup.

Figure 11: Bidirectional Replication Flow

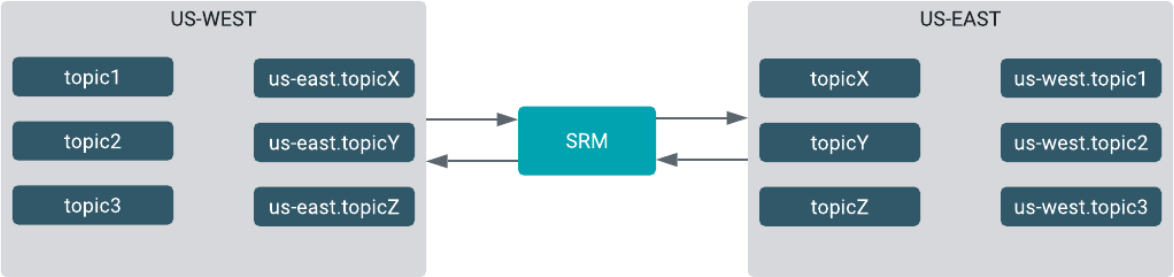

This replication setup can be configured with the srm-control as follows:

```
srm-control topics --source us-west --target us-east --add "topic.*?"
```

```
srm-control topics --source us-east --target us-west --add "topic.*?"
```

These commands add all topics on both clusters to the replication allowlist using a regex pattern. With the `DefaultReplicationPolicy`, these commands would not create replication loops. This is because the `DefaultReplicationPolicy` automatically detects that the topics with the prefixes do not need to be replicated to the source.

Using these srm-control commands with the `IdentityReplicationPolicy` would, however, result in an endless replication loop. Remote topics would be replicated back to their source. In a case like this, the allow and deny lists must be correctly configured so that no loops are present. For example:

```
srm-control topics --source us-west --target us-east --add topic1, topic2, t
opic3
```

```
srm-control topics --source us-east --target us-west --add topicX, topicY, t
opicZ
```

Note: Regex patterns can still be used with the `IdentityReplicationPolicy`, but you must ensure that the regex pattern matches do not create any loops.

Fan-in and fan-out replication flows

You can construct fan-in replication flows, where records from multiple source clusters are aggregated in a single target cluster.

Figure 12: Fan-in Replication Flow

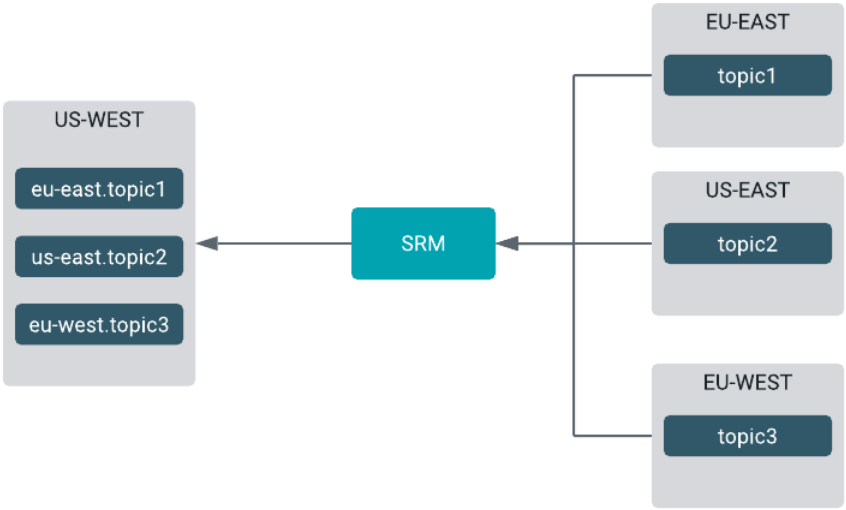

Similarly, you can construct fan-out replication flows as well, where a single cluster is replicated to multiple target clusters.

Figure 13: Fan-out Replication Flow

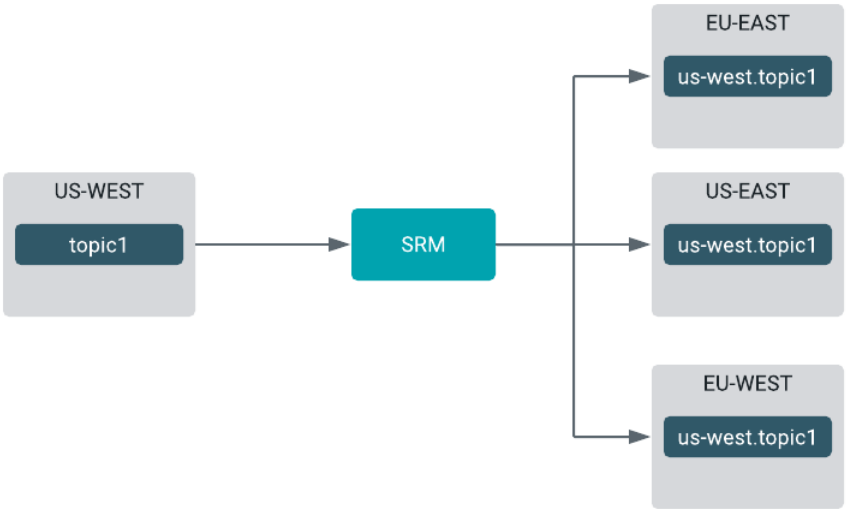

Related Information

Kafka Java documentation - ReplicationPolicy

Streams Replication Manager remote topic discovery

Learn how Streams Replication Manager keeps track of remote topics and how this behavior allows Streams Replication Manager to provide full support for prefixless replication with the IdentityreplicationPolicy.

In order to provide replication monitoring, Streams Replication Manager must discover and keep track of the topics that are replicated. By default, Streams Replication Manager achieves this using an internal Kafka topic, which contains information regarding the currently replicated topics. The behavior is as follows:

1. Streams Replication Manager Drivers periodically check which topics must be replicated.
2. This check results in a list of source – target topic pairs that must be replicated.
3. The Streams Replication Manager Driver writes these source – target pairs into the srm-meta.internal Kafka topic, which is an internal compact topic located in the target cluster.
4. The Streams Replication Manager Service scans srm-meta.internal and consumes the latest message. The latest message contains the currently replicated topics as well as source – target topic name mappings.

Remote topic detection takes into consideration the current state of the replication flow. For example, assume that you have a remote topic, us-east.orders that was replicated at some point in time, but is not currently replicated anymore. Streams Replication Manager correctly identifies that this topic is no longer replicated and will not keep track of it as a remote topic indefinitely. The replication metrics provided by the Streams Replication Manager Service also correctly reflect this state of the topic.

In addition, the use of an internal topic to store replication metadata allows Streams Replication Manager to correctly identify which target maps to which source without relying solely on topic prefixes. This makes it possible for Streams Replication Manager to provide replication monitoring even if a replication policy different than the `DefaultReplicationPolicy` is in use. Most notably, this enables replication monitoring when Streams Replication Manager is configured for prefixless replication with the `IdentityReplicationPolicy`.

Automatic group offset synchronization

Automatic group offset synchronization is a feature in Streams Replication Manager that automates the export and application of translated consumer group offsets. Enabling this feature can simplify the manual steps that you need to take to migrate consumer groups in a failover or failback scenario.

Streams Replication Manager automatically translates consumer group offsets between clusters. While the offset mappings are created by Streams Replication Manager, they are not applied by default to the consumer groups of the target cluster. As a result, by default, migrating consumer groups from one cluster to another involves running the srm-control offsets and kafka-consumer-groups tools. The srm-control offsets tool exports translated offsets, kafka-consumer-groups resets and applies the translated offsets on the target cluster.

You can automate this process by enabling automatic group offset synchronization. If automatic group offset synchronization is enabled, the translated group offsets of the source cluster are automatically exported from the source cluster and are applied on the target cluster (they are written to the __consumer_offsets topic). If you choose to enable this feature, running srm-control offsets and kafka-consumer-groups is not required to migrate consumer groups. You only need to restart and redirect consumers to consume from the new cluster.

Although automatic group offset synchronization can simplify migrating consumer groups, ensure that you understand the following about its behavior:

- Automatic group offset synchronization does not fully automate a failover or failback process. It only allows you to skip certain manual steps. Consumers must be restarted and redirected to the new cluster even if the feature is enabled.
- Offsets are synced at a configured interval. As a result, it is not guaranteed that the latest translated offsets are applied. If you want to have the latest offsets applied, Cloudera recommends that you export and apply consumer group offsets manually. The exact interval depends on sync.group.offsets.interval.seconds and emit.checkpoints .interval.seconds.
- The checkpointing frequency configured for Streams Replication Manager can have an effect on the group offset synchronization frequency. The frequency of group offset synchronization can be configured with sync.gro up.offsets.interval.seconds. However, specifying an interval using this property might not result in the offsets being synchronized at the set frequency. This depends on how emit.checkpoints.interval is configured. The emit .checkpoints.interval property specifies how frequently offset information is fetched (checkpointing). Because offset synchronization can only happen after offset information is available, the frequency configured with the emit.checkpoints.interval property might introduce additional latency. For example, assume that you set offset synchronization to 60 seconds (default), but have checkpointing set to 120 seconds. In such a case, offset synchronization happens every 60 seconds, but because offset information is only refreshed every 120 seconds, in practice offsets are only synchronized every 120 seconds.

- Offsets are only synchronized for the consumers that are inactive in the target cluster. This is done so that the Streams Replication Manager does not override the offsets in the target cluster.
- During a failover-failback scenario, if a consumer group makes progress in the replica topic in the backup cluster and a failback occurs, the progress made in the replica topic is lost, and messages are re-processed from the original topic.

Additionally, Streams Replication Manager checkpointing behavior might affect automatic group offset synchronization in the following ways:

- By default, Streams Replication Manager checkpointing only uses the latest offset sync record, which means that consumer groups behind the latest mapping cannot be checkpointed. If the consumer group is behind by more than the value of offset.lag.max property in replicating the records to the offset sync, the offset translation is skipped because the offset is too far in the past to translate accurately.

  Note: In some cases, Streams Replication Manager checkpointing can also take older synchronized offsets into account.

- Streams Replication Manager checkpointing reads the offset syncs topic to create offset mappings for committed consumer group offsets. In some cases, a mapping might not available in offset syncs. In such a case, Streams Replication Manager copies the source offset, which might not be a valid offset in the replica topic. This can cause issues when auto offset sync is also turned on.
- If sync.group.offsets.enabled is set to true, Kafka only emits checkpoints for partitions that have offset syncs. Partitions for which no offsets are committed (because they cannot be translated) do not appear when describing consumer offsets in the target cluster.

Related Information

Configuring automatic group offset synchronization

Understanding co-located and external clusters

The Kafka clusters that Streams Replication Manager connects to can be categorized into two groups. They can be co-located with or external to Streams Replication Manager. The category dictates how you configure the Streams Replication Manager service and the srm-control tool.

Streams Replication Manager connects to and replicates data between Kafka clusters, which consist of one or more Kafka brokers, deployed on clusters. These Kafka clusters that Streams Replication Manager connects to can be categorized into two groups. They can either be co-located with or external to Streams Replication Manager. Which category a specific cluster falls into is decided based on the relation between that cluster and the Streams Replication Manager service.

A co-located Kafka cluster is the Kafka cluster that is running in the same cluster as the Streams Replication Manager service. Any other Kafka cluster that is remote to Streams Replication Manager either logically or geographically is considered external.

For example, consider the following deployment:

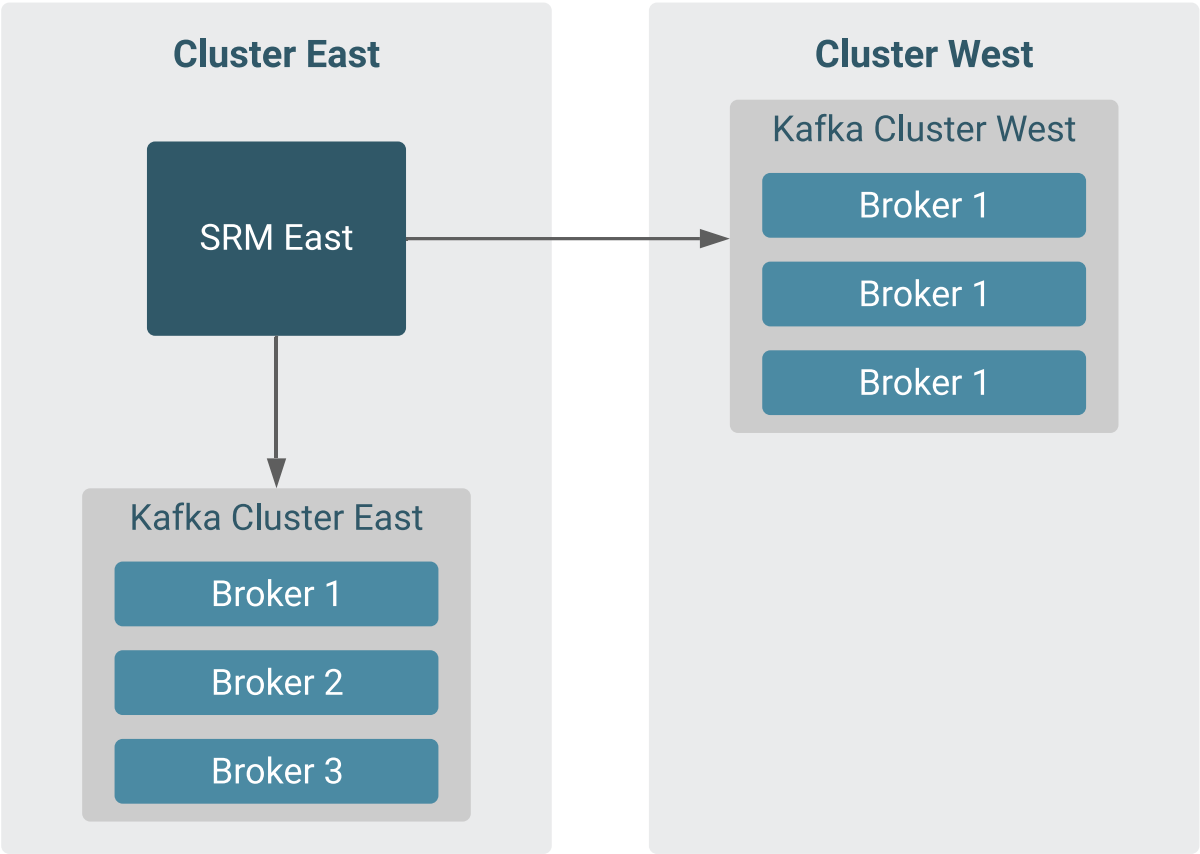

This deployment has two clusters, and both clusters have a Kafka cluster. However, only Cluster East has Streams Replication Manager deployed on it. From the perspective of Streams Replication Manager East, Kafka Cluster East is co-located, while Kafka Cluster West is external.

In a more advanced deployment with multiple Streams Replication Manager services, a single Kafka cluster will fall into both categories. From the perspective of a specific Streams Replication Manager service a cluster will be co-located, while for others it will be external.

For example, consider the following deployment:

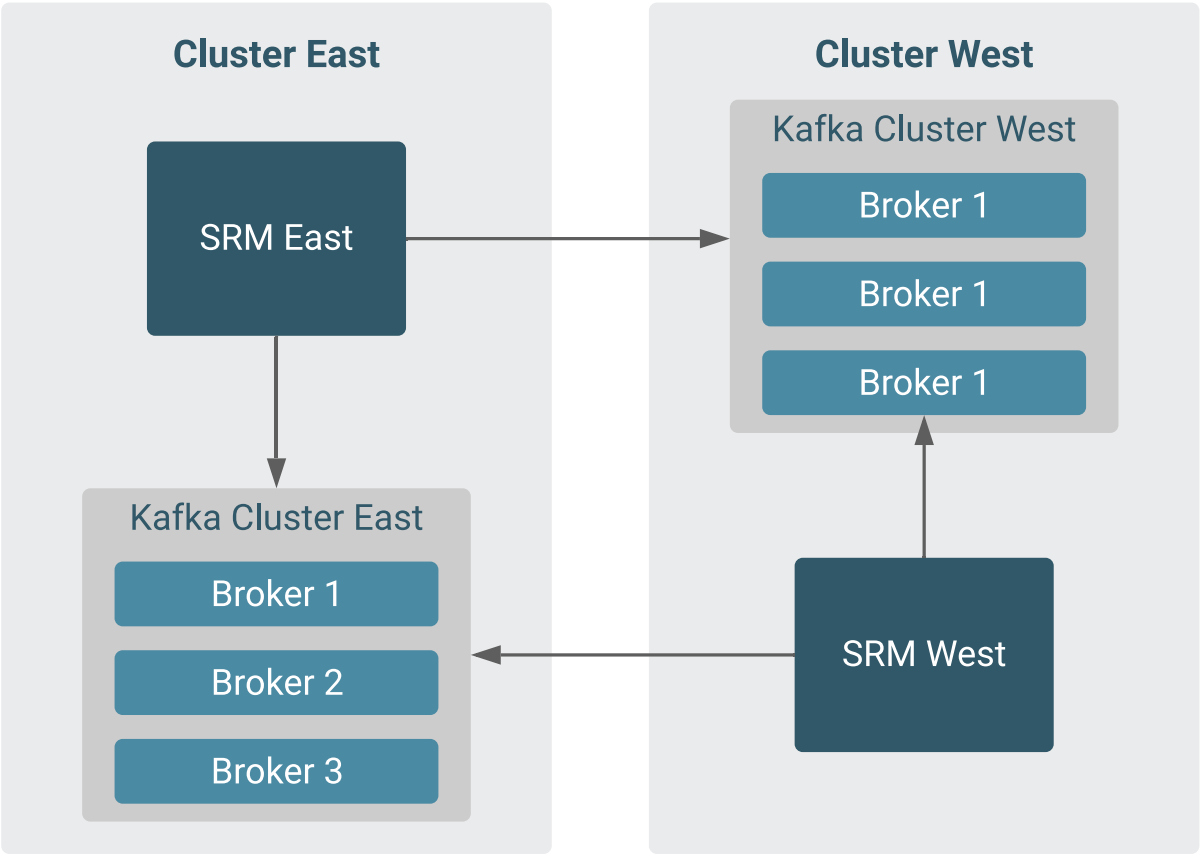

In this example, both clusters have a Kafka cluster as well as Streams Replication Manager. From the perspective of Streams Replication Manager East, Kafka Cluster East is co-located, Kafka Cluster West is external. From the perspective of Streams Replication Manager West, Kafka Cluster West is co-located, Kafka Cluster East is external.

It is also possible to not have co-located Kafka clusters. For example, consider the following deployment:

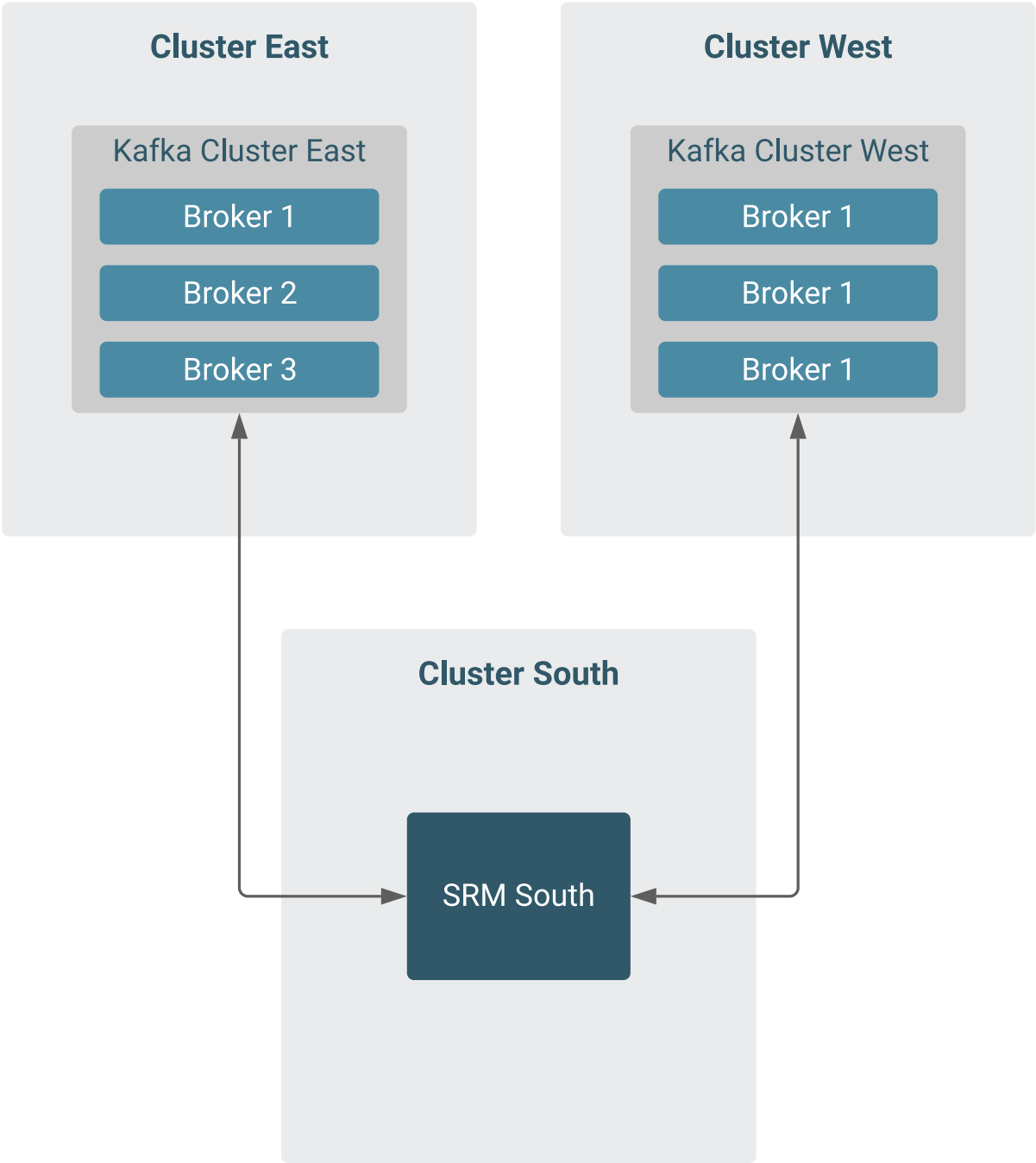

In this example, the clusters that have Kafka deployed on them do not have Streams Replication Manager. Instead, data is replicated by a Streams Replication Manager instance deployed on a separate cluster. From the perspective of Streams Replication Manager South, both Kafka Cluster East and West are external, there is no co-located cluster. In a scenario like this, configuration tasks related to the co-located cluster do not need to be completed.

Being able to correctly identify what category a Kafka cluster falls into is important as the category dictates how you configure each Streams Replication Manager service and the srm-control tool. In general, a co-located Kafka cluster requires less configuration than external Kafka clusters. This is because Cloudera Manageris able to automatically pass certain configuration properties about the co-located Kafka cluster to Streams Replication Manager. External Kafka clusters on the other hand must be fully configured and specified manually.

For more information on how to configure and set up Streams Replication Manager, review any of the configuration examples available in Using Streams Replication Manager in Cloudera Public Cloud overview or Configuration examples.

Understanding Streams Replication Manager properties, their configuration and hierarchy

There are a number of configuration properties that Streams Replication Manager accepts, but are not exposed directly in Cloudera Manager. You can configure these properties with the Streams Replication Manager's Replication Configs property. Additionally, these properties can be configured on different levels allowing for granular control over how and when they are applied.

Streams Replication Manager supports various configuration properties. With the exception of adding and enabling replications, all fundamental properties required to set up and start data replication between clusters are available for configuration directly through Cloudera Manager, or are configured automatically by Cloudera Manager.

In addition to these properties, Streams Replication Manager accepts and supports a number of additional properties including Streams Replication Manager specific properties, Kafka Connect worker properties as well as Kafka properties available in the version of Kafka that you are using. These, however, are not directly available for configuration in Cloudera Manager, and do not have dedicated configuration entries on the UI. Properties like these are configured through the Streams Replication Manager's Replication Configs property.

The configuration properties that you add to Streams Replication Manager's Replication Configs can be set on multiple levels with the help of property prefixes. These prefixes allow you to exercise control over when and where (at what level) a configuration is used by Streams Replication Manager. This gives you fine grained control over how data is replicated.

The prefixes that you can use and the configuration levels they correspond to are as follows:

- Global level - no prefix

  For example:

  ```
  offset.flush.timeout.ms
  ```

- Global worker level - prefixed with the workers. prefix

  For example:

  ```
  workers.scheduled.rebalance.max.delay.ms
  ```

- Global connector level - prefixed with the connectors. prefix

  For example:

  ```
  connectors.producer.override.buffer.memory
  ```

- Cluster level - prefixed with cluster alias

  For example:

  ```
  [***ALIAS***].offset.flush.timeout.ms
  ```

- Replication level - prefixed with a replication name

  For example:

  ```
  [***ALIAS***]->[***ALIAS***].consumer.max.poll.records
  ```

Note: All properties in Streams Replication Manager support prefixes, even the ones that you set directly through the Cloudera Manager UI. However, the prefixes for these properties are set automatically in the background based on your overall configuration. Therefore, from a user perspective, prefixes only become significant when you use Streams Replication Manager's Replication Configs.

Properties follow a hierarchy, which is based on what prefix they have. In general, prefixed properties take precedence over non-prefixed properties, and more specific prefixes take priority over less specific ones. That is, replication level properties take precedence over both cluster and global level configurations. Cluster level properties take precedence over global level configurations. Lastly, global level configurations are overridden by both replication and cluster level properties.

Not all prefixes and configuration levels are valid for all properties. For example, most properties related to Kafka client connections (bootstrap servers, aliases, security configurations and so on) are not relevant on a replication level. While they can be set, they will not have an effect.

The following sections give more information on the available configuration levels/prefixes, give examples of what types of properties can be set at each level, as well as recommendations on when and how to use each.

Global level

Global configuration is achieved by adding the property on its own, without a prefix. This is the most generic configuration, with the lowest priority level. Global level configurations can also be regarded as fallback configurations. This is a result of the property hierarchy. Streams Replication Manager will fall back and use a global level configuration if there are no cluster or replication level configurations set for a specific property.

The following types of properties are accepted at this level by Streams Replication Manager:

Streams Replication Manager service level properties

These configurations are specific to Streams Replication Manager. Additionally, they are applied to Streams Replication Manager as a whole. For example, mm.rest.protocol is a property typically set at this level, as it allows configuring the protocol used by all replication specific Connect REST Servers run internally by the Streams Replication Manager Driver to use the same protocol.

Kafka client connection properties

Client connection properties are used by Streams Replication Manager when it connects to a Kafka cluster. Client connection properties can be set on a global level, however, not all connection related properties are compatible with this level. For example, you can set bootstrap.servers on a global level, but because in the majority of cases you will have a unique bootstrap server set for each individual connection, setting this property on a global level is redundant.

As a result, Cloudera recommends that you only configure security related connection properties at this level and only if you have multiple clusters in your deployment that use the same security configuration. For example, you might have a deployment where Streams Replication Manager connects to many Kafka clusters, and all of them use the same cipher suite. In a case like this you can set ssl.cipher.suites at the global level a single time instead of configuring it on a per cluster basis.

Kafka Connect worker properties

These configurations are applied to all Kafka Connect workers run internally by the Streams Replication Manager Driver. Any Connect worker property is supported at this level as long as it uses one of the following prefixes:

- offset.storage
- config.storage
- status.storage
- key.converter
- value.converter
- header.converter
- task
- worker

- listeners.https

For example, you can configure offset.storage.replication.factor at this level to specify the replication factor of the internal topic used to track the source offsets of Streams Replication Manager in all target clusters.

Other worker properties that do not use the listed prefixes can still be configured globally. However, they must be configured on a global worker level using the workers. prefix.

Kafka Connect connector properties

These configurations are applied to all connectors in all replications run internally by the Streams Replication Manager Driver. The Connect connector properties supported at this level are as follows:

- errors.retry.timeout
- errors.retry.delay.max.ms
- errors.tolerance
- errors.log.enable
- errors.log.include.messages
- config.properties.exclude
- config.properties.blacklist
- config.property.filter.class
- consumer.poll.timeout.ms
- admin.timeout.ms
- refresh.topics.enabled
- refresh.groups.enabled
- sync.topic.configs.enabled
- sync.topic.acls.enabled
- emit.heartbeats.interval.seconds
- emit.checkpoints.interval.seconds
- sync.group.offsets.enabled
- sync.group.offsets.interval.seconds
- replication.policy.class
- replication.policy.separator
- heartbeats.topic.replication.factor
- checkpoints.topic.replication.factor
- offset-syncs.topic.replication.factor
- offset.lag.max
- offset-syncs.topic.location
- connect.start.task.timeout.ms
- disable.source.topic.auto.creation
- copy.source.offset.in.header.enabled

For example, you can configure sync.group.offsets at this level to sync the translated consumer group offsets.

Other connector properties that are not listed can still be configured globally. However, they must be configured on a global connector level using the connectors. prefix.

Global worker level

Global worker level configuration is achieved by using the workers. prefix. This level supports Kafka Connect worker properties only. Properties specified with the workers. prefix are applied to all Connect workers run internally by the Streams Replication Manager Driver.

The difference between the global worker level and the global level (no prefix) is that on a global level only a handful of Connect worker properties are supported. The global worker level on the other hand supports all Connect worker properties.

For example, scheduled.rebalance.max.delay.ms is not supported on a global level, it is not applied to all Connect workers if it is set without a prefix. However, specifying the property on a global worker level is supported. That is, when the property is set with the workers. prefix, it is applied to all workers.

Global connector level

Global connector level configuration can be achieved by prefixing the configuration property with the connectors. prefix. This level supports Kafka Connect connector properties only. Properties that use this prefix are applied to all Connect connectors run internally by the Streams Replication Manager Driver.

The difference between the global connector level and the global level (no prefix) is that on a global level only a handful of Connect connector properties are supported. The global connector level on the other hand supports all Connect connector properties.

The global connector level prefix can also be used to tweak all consumers and producers that are used by the Streams Replication Manager Driver. For example, by using the connectors.consumer. prefix, any consumer property can be specified to tweak the consumers used in the replications. Similarly, the connectors.producer.override. prefix can be used to override producer configurations in the replications.

Cluster level

Cluster level configuration is achieved by prefixing the configuration property with a cluster alias specified in Streams Replication Manager Cluster alias. Cluster level configurations have a higher priority than global configurations. As a result, cluster level configurations take precedence over global configurations.

The following types of configurations are accepted at this level by Streams Replication Manager:

Kafka client connection properties

These configurations are used by Streams Replication Manager to connect to the Kafka cluster identified by the alias. Streams Replication Manager is capable of generating connection configurations securely for most use-cases and Kafka clusters either through automation, or with Kafka Credentials (Administration External Accounts Kafka Credentials). Cloudera recommends that you only use automation or Kafka credentials to configure these properties.

For advanced use-cases, or when a specific connection configuration property is not available through Kafka credentials, the cluster prefix can be used to specify the configuration property. For example, the sasl.login.callback.handler.class property cannot be set using Kafka credentials. If in your deployment you are using a third-party callback handler and need to specify it for Streams Replication Manager, you can do so by adding sasl.login.callback.handler.class with the appropriate cluster prefixes to Streams Replication Manager Replication Configs.

Kafka Connect worker properties

These configurations are applied to Kafka Connect workers which correspond to a replication targeting the Kafka cluster identified by the alias prefix. On this level, all Connect worker configurations are supported. For example, you can use offset.flush.timeout.ms to increase the flush timeout to support high-traffic replication flows.

Replication level

Replication level configuration can be achieved by prefixing the configuration property with the name of the replication. Replication level configurations have a higher priority than cluster or global level configurations. As a result, replication level configurations take precedence over both cluster and global configurations.

The following types of configurations are accepted at this level by Streams Replication Manager:

Kafka Connect worker properties using the cluster->cluster.worker.prefix

These properties are applied to the Kafka Connect workers which correspond to the replication identified by the prefix. On this level, all Connect worker configurations are supported. For example, offset.flush.timeout.ms can be specified to increase the flush timeout to support high-traffic replication flows.

Kafka Connect connector properties

These configurations are applied to all connectors in the replication identified by the prefix.

In addition to the replication specific connector configurations, you can also tweak the consumers and producers that are used within a specific replication. For example, by using the cluster->cluster.consumer. prefix, any consumer property can be specified to tweak the consumers used in the replication. Similarly, the cluster->cluster.producer.override. prefix can be used to override producer configurations in the replication flow.

Related Information

Configuring properties not exposed in Cloudera Manager

Cloudera Runtime 7.3.1

Planning for Streams Replication Manager

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Streams Replication Manager requirements..... 4

Streams Replication Manager recommended deployment architecture..... 5

Streams Replication Manager requirements

When planning and designing your Streams Replication Manager deployment that has multiple instances of the Streams Replication Manager service, there are a number of practices that you must follow. Otherwise, you might encounter issues when you are replicating data with Streams Replication Manager.

Kafka cluster aliases must be uniform across clusters

When you have multiple Streams Replication Manager services, the cluster aliases must be defined identically in all instances. Streams Replication Manager utilizes the cluster alias to detect replication loops.

The heartbeats topic must be present on all source clusters

To be able to properly monitor and discover replications, the heartbeats topic must be present on all source clusters. The heartbeats topic is created by the Streams Replication Manager Driver when a target cluster is set with the Streams Replication Manager Driver Target Cluster property and when heartbeating is enabled for the replication. This means that to ensure that the heartbeats topic is created, and has data written into it periodically, at least a single Streams Replication Manager Driver cluster must target each source cluster.

Driver targets must be managed correctly

The Streams Replication Manager Cluster alias and the Streams Replication Manager Driver Target Cluster properties must be managed correctly when you have multiple Streams Replication Manager services. If a target is specified for the replication in the Streams Replication Manager Driver Target Cluster property, a Connect worker is created for each possible cluster pair based on the aliases present in Streams Replication Manager Cluster alias. As a result of this, it is possible for different Streams Replication Manager Driver clusters to merge through their Connect workers. To avoid this, ensure that different Streams Replication Manager Driver clusters target different Kafka clusters.

Service targets must be managed correctly

The Streams Replication Manager Cluster alias and the Streams Replication Manager Service Target Cluster properties must be managed correctly when you have multiple Streams Replication Manager services. If a target is specified for monitoring in the Streams Replication Manager Service Target Cluster property, a Kafka Streams application instance is created for each cluster present in the Streams Replication Manager Service Target Cluster property. As a result of this, it is possible for different Streams Replication Manager Service clusters to merge through their Kafka Streams application instances. To avoid this, ensure that different Streams Replication Manager Service clusters target different Kafka clusters.

All Streams Replication Manager services must use the same replication policy

Streams Replication Manager ships with and supports multiple replication policies. Using a mix of these policies in a single deployment is not supported. Decide which replication policy you want to use and ensure that all Streams Replication Manager services are configured to use your chosen policy. In addition transitioning from one policy to another is not recommended.

Ensure that you do not create replication loops if using the IdentityReplicationPolicy

The `IdentityReplicationPolicy` cannot detect replication loops. As a result, if you are planning to create a setup with bidirectional replication using the `IdentityReplicationPolicy`, you must ensure that you do not create any replication loops when you set up your allowlists and denylists (topic filters) with srm-control.

Streams Replication Manager recommended deployment architecture

Learn about pull mode, which is the Cloudera-recommended deployment architecture for Streams Replication Manager.

While Streams Replication Manager can be deployed in many different ways, the Cloudera recommended setup is pull mode. Pull mode refers to a Streams Replication Manager deployment where data replication happens by pulling data from remote source clusters, rather than pushing data into remote target clusters. A Streams Replication Manager deployment that is in pull mode conforms to the following:

- An instance of the Streams Replication Manager service (Driver and Service roles) is deployed on all clusters that host a target Kafka cluster. In other words, each target Kafka cluster in the deployment has a co-located Streams Replication Manager service.
- All instances of the Streams Replication Manager service use the same replication policy.
- Streams Replication Manager Driver roles only execute replications that target their co-located Kafka cluster.
- Streams Replication Manager Service roles only target their co-located Kafka cluster.
- If using the `IdentityReplicationPolicy` in a bidirectional setup, no replication loops are present.

The reason why pull mode is recommended is because this is the deployment type that was found to provide the highest amount of resilience against various timeout and network instability issues. For example:

- Streams Replication Manager Driver roles run Connect workers which are coordinating through the target cluster of the replication flow. This means that Connect workers are more closely tied to the target cluster than the source cluster. Having the Streams Replication Manager Drivers and the target Kafka cluster closely located to each other minimizes group membership and rebalance related timeout issues. Additionally, this also minimizes the network instability on the producer side of the Streams Replication Manager Drivers which reduces data duplication in the target cluster.
- Streams Replication Manager Service roles also coordinate through the target cluster. This makes them more sensitive to timeout and network partition issues tied to the target cluster. In addition, the Streams Replication Manager Service reads from and writes to the target Kafka through the Kafka Streams application. Having the Streams Replication Manager Services and the target Kafka clusters closely located to each other can minimize timeout and network partition issues. Additionally, for Kafka clusters in the cloud, hosting the Streams Replication Manager Service in the same data center as the Kafka cluster can help keep cloud costs to a minimum.
- In a situation where there is a network partition, or one of the clusters in the replication is unavailable, it is preferable to let the target cluster pull the data when the connection is established, rather than the source cluster trying to push data indefinitely.

Pull mode deployment example

Note: The following simple example does not go into the detail regarding the exact configuration steps required to set up Streams Replication Manager or your clusters. It is meant to provide you with a general overview of how a pull mode deployment looks like. For configuration how to documentation see, How to Streaming Configuring Streams Replication Manager in the Cloudera Runtime documentation.

Consider a simple deployment that has three clusters. Cluster A, B, and C. Each of them has a Kafka cluster.

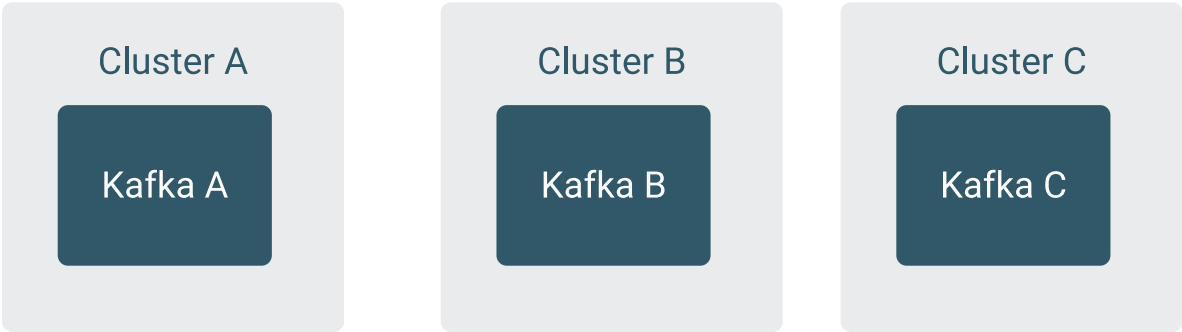

To achieve a pull mode deployment, you must deploy the Streams Replication Manager service (both Driver and Service roles) on all three clusters. The Driver and Service roles must target their co-located Kafka cluster.

To be more precise, in the case of Driver targets, what you must ensure is that each source cluster is targeted by at least a single Driver. This is required so that a heartbeats topic is created in that cluster. This is ensured in the this example because each Kafka has a co-located Driver targeting it. However, if you have a unidirectional replication setup, where the source Kafka cluster does not have a co-located Streams Replication Manager service, you must ensure that one of your Drivers is targeting that Kafka cluster.

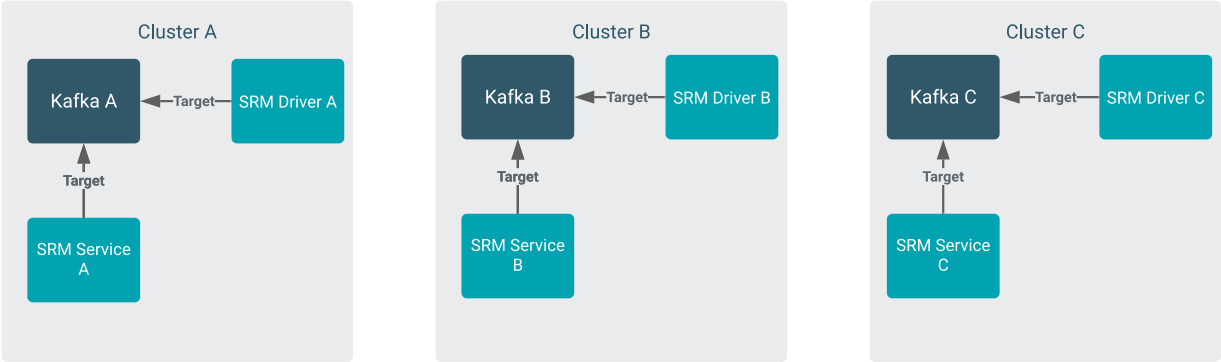

Once setup is complete, you can start configuring your replications. For example, assume that you want to have data replicated from Cluster B and C to cluster A. In this case, you need to enable two separate replications, B->A and C->A. To achieve pull mode, the two replications must be executed by Streams Replication Manager Driver A.

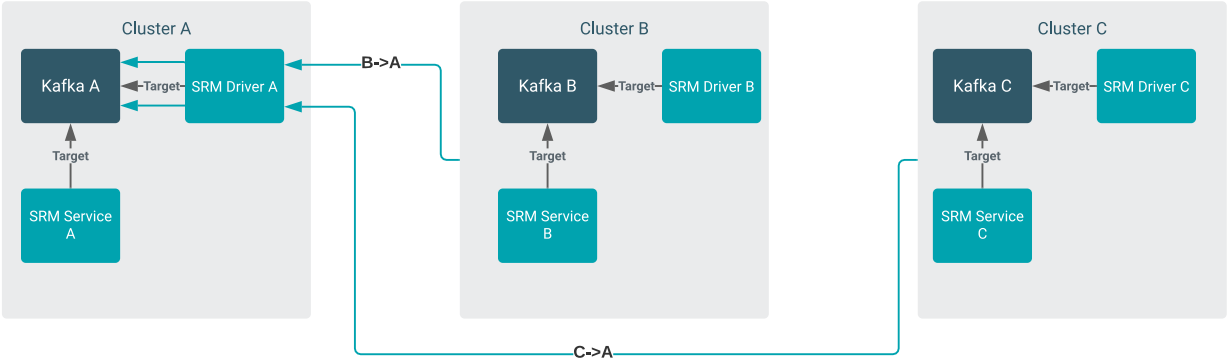

Any number of replications can be set up between the clusters, but you must always ensure that each Driver is only executing the replications targeting their co-located cluster. For example, assume that in addition to the replications set up previously, you also want to set up replication to Cluster B from Cluster A and Cluster C. In this case, the deployment would changes as follows.

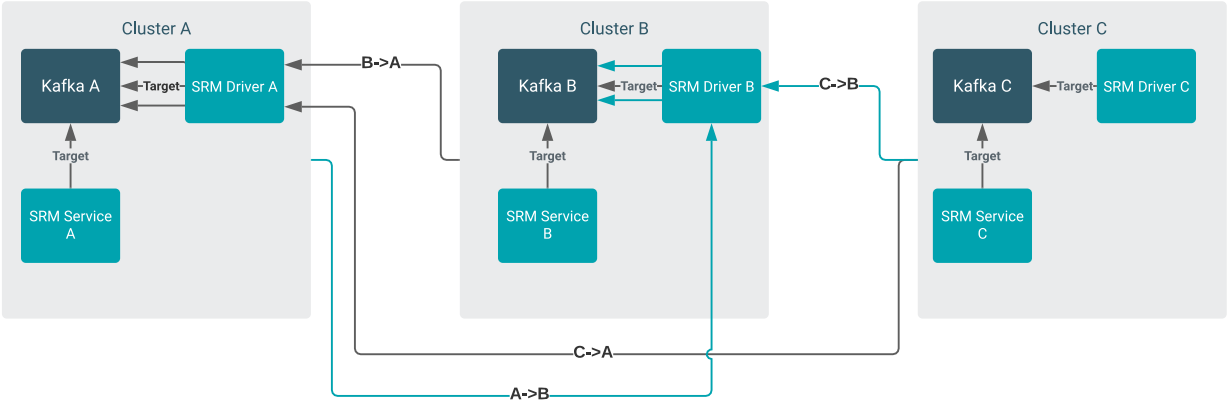
Cluster A
Kafka A
Target
SRM Driver A
Target
SRM Service A
B->A
Cluster B
Kafka B
Target
SRM Driver B
Target
SRM Service B
C->B
Cluster C
Kafka C
Target
SRM Driver C
Target
SRM Service C
C->A
A->B

Cloudera Runtime 7.3.1

Streams Replication Manager Reference

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

srm-control Options Reference..... 4

Configuration Properties Reference for Properties not Available in Cloudera Manager..... 4

Kafka credentials property reference.....6

Streams Replication Manager Service data traffic reference..... 8

srm-control Options Reference

A collection of all options and their descriptions for the srm-control command line tool.

Table 1: Topics and groups subcommand properties

Options	Description
-h, --help	Shows the help message.
--source	Specifies the source cluster
--target	Specifies the target cluster
--config	Specifies the Streams Replication Manager configuration file to use.
--add	Specifies topics or groups to add to the allowlist
--remove	Specifies topics or groups to remove from the allowlist
--add-blacklist	Specifies topics or groups to add to the denylist
--remove-blacklist	Specifies topics or groups to remove from the denylist
--list	Lists current allowlist and denylist
--bootstrap-servers	Specifies the bootstraps servers
--producer-props	Specifies producer configuration properties
--consumer-props	Specifies consumer configuration properties
--props	Specifies client configuration properties

Table 2: Offsets subcommand properties

Option	Description
-h, --help	Shows the help message.
--source	Specifies the source cluster
--target	Specifies the target cluster
--config	Specifies the Streams Replication Manager configuration file to use.
--export	Export translated offsets
--group	Specifies the groups translated offsets should be exported for
--bootstrap-servers	Specifies the bootstraps servers
--props	Specifies client configuration properties

Configuration Properties Reference for Properties not Available in Cloudera Manager

A collection of Streams Replication Manager configuration properties not available in Cloudera Manager

The following table lists all Streams Replication Manager specific configuration properties that are not available directly for configuration via Cloudera Manager.

Property	Default Value	Description
checkpoints.topic.replication.factor	3	Replication factor used for internal checkpoints topics.
emit.checkpoints.enabled	true	Enables periodic emission of consumer offset information.
emit.heartbeats.enabled	true	Enables periodic emission of heartbeats.
emit.heartbeats.interval.seconds	5 (seconds)	The interval at which Streams Replication Manager emits heartbeats.
heartbeats.topic.replication.factor	3	Replication factor used for internal heartbeat topics.
offset-syncs.topic.replication.factor	3	Replication factor used for internal offset-syncs topics.
refresh.groups.enabled	true	Enables a periodical check for new consumer groups on source clusters.
refresh.topics.enabled	true	Enables a periodical check for new topics on source clusters.
replication.factor	2	Replication factor used for remote topics.
replication.policy.class	org.apache.kafka.connect.mirror.DefaultReplicationPolicy	Replication policy to use. Use this property to set a custom replication policy. If you want to enable prefixless replication, use the Enable Prefixless Replication property instead. Enable Prefixless Replication is found in Cloudera Manager Streams Replication Manager Configuration For more information, see *Enabling prefixless replication*.
sync.topic.configs.enabled	true	Enables the monitoring of the source cluster for configuration changes.
config.properties.exclude	follower\.replication\.throttled\.replicas, leader\.replication\.throttled\.replicas, message\.timestamp\.difference\.max\.ms, message\.timestamp\.type, unclean\.leader\.election\.enable, min\.insync\.replicas	A list of topic configuration properties or regexes that must not be replicated.
replication.records.lag.calc.enabled	true	Controls whether the replication-records-lag metric is calculated. This metric provides information regarding the replication lag based on offsets. The metric is available both on the cluster and the topic level. The calculation of this metric might add latency to replications and impact Streams Replication Manager performance. If you are experiencing performance issues, you can try setting this property to false to disable the calculation of replication-records-lag. Alternatively, you can try fine-tuning how Streams Replication Manager calculates replication-records-lag with the replication.records.lag.calc.period.ms and replication.records.lag.end.offset.timeout.ms properties.

Property	Default Value	Description
replication.records.lag.calc.period.ms	0	Controls how frequently Streams Replication Manager calculates the replication-records-lag metric. The default value of 0 means that the metric is calculated continuously. Cloudera recommends configuring this property to 15000 ms (15 seconds) or higher if you are experiencing issues related to the calculation of replication-records-lag. A calculation frequency of 15 seconds or more results in the metric being available for consumption without any significant impact on Streams Replication Manager performance.
replication.records.lag.end.offset.timeout.ms	6000	Specifies the Kafka end offset timeout value used for calculating the replication-records-lag metric. Setting this property to a lower value than the default 6000 ms (1 minute) might reduce latency in calculating replicat ion-records-lag, however, replication-records-lag calculation might fail. A value higher than the default can help avoid metric calculation failures, but might increase replication latency and lower Streams Replication Manager performance.

Related Information

Enabling prefixless replication

Kafka credentials property reference

A Kafka credential allows services and Cloudera tools to securely connect to Kafka clusters that are external to a particular Cloudera Manager instance. Review the Kafka credentials property reference to better understand what options you have when configuring and setting up a Kafka credential.

Name

Description

Specifies the name of the Kafka credential. For Streams Replication Manager, the name you specify here is equivalent to the cluster alias. As a result, this name is used to refer to the cluster that the credential defines when configuring Streams Replication Manager properties and when using the srm-control tool. Cloudera recommends that you specify unique and easily identifiable names.

Boostrap servers

Description

Specifies the addresses of the Kafka brokers. Add a comma-separated list of host and port pairs. The host can be specified as an IP address or an FQDN.

Example

my-kafka-cluster-host-1.com:9093, my-kafka-cluster-host-2.com:9093

JAAS secret [1-3]

Description

There are three JAAS Secret properties, numbered 1-3. All three properties can be used to specify sensitive JAAS configuration values. Values set in these properties can be referenced with placeholders in the JAAS configuration you add to the JAAS template property.

JAAS template

Description

Specifies the JAAS configuration that Streams Replication Manager uses when connecting to this cluster. The configuration you add to this property can contain placeholders. These placeholders can be used to hide sensitive information. The property accepts and resolves the following placeholders:

- ##JAAS_SECRET_1##

  Used to refer to the value set in the JAAS Secret 1 property.
- ##JAAS_SECRET_2##

  Used to refer to the value set in the JAAS Secret 2 property.
- ##JAAS_SECRET_3##

  Used to refer to the value set in the JAAS Secret 3 property.

Example

org.apache.kafka.common.security.plain.PlainLoginModule required username="##JAAS_SECRET_1##" password="##JAAS_SECRET_2##";

In this example ##JAAS_SECRET_1## and ##JAAS_SECRET_2## are used to hide sensitive information. The values of the username and password are defined in the JAAS Secret 1 and JAAS Secret 2 properties.

If for example the JAAS Secret 1 and JAAS Secret 2 properties were set to “username” and “password”, the resolved JAAS configuration would be the following:

```
org.apache.kafka.common.security.plain.PlainLoginModule required
 username="username" password="password";
```

Kerberos Service Name

Description

Specifies the name of the Kafka kerberos principal. Only required if the SASL Mechanism property is set to GSSAPI.

Key Password

Description

Specifies the password used to access the keys stored in the keystore configured in the Keystore Path property.

Keystore Password

Description

Specifies the password used to access the keystore configured in the Keystore Path property.

Keystore Path

Description

Specifies the path to the keystore containing the authorized client’s certificates or keys

Keystore Type

Description

Specifies the type of the keystore configured in the Keystore Path property

SASL Mechanism

Description

Specifies the SASL mechanism used for authentication by the Kafka cluster that the client is connecting to.

Accepted values

- GSSAPI
- PLAIN
- SCRAM-SHA-256
- SCRAM-SHA-512
- OAUTHBEARER

Security Protocol

Description

Specifies the security protocol used by the Kafka cluster that the client is connecting to.

Accepted values

- PLAINTEXT
- SSL
- SASL_PLAINTEXT
- SASL_SSL

Truststore Password

Description

Specifies the password used to access the truststore specified in the Truststore Path property

Truststore Path

Description

Specifies the path to the truststore containing the certificates or keys belonging to the Kafka cluster that the client is connecting to.

Truststore Type

Description

Specifies the type of the truststore configured in the Truststore Path property.

Streams Replication Manager Service data traffic reference

The Streams Replication Manager Service generates data traffic when it collects replication related metrics. Data traffic is generated between the Streams Replication Manager Service and its target Kafka cluster. Additionally, if Remote Querying is set up, data traffic is also generated between different Streams Replication Manager Services that are remote to each other. Review these examples to gain a better understanding of how much data traffic you should expect in your deployment.

Data transferred between the Streams Replication Manager Service and Kafka

The Streams Replication Manager service generates data traffic between itself and its target Kafka cluster. This data traffic is generated as a result of the Streams Replication Manager Service collecting the raw metrics from the target Kafka cluster. In the case of:

- A single replication.
- A single replicated topic with 1000 partitions and 100 messages published per second.
- A single consumer group.

The amount of data traffic generated is approximately 280 kB per second.

Data transferred between Streams Replication Manager Services by Remote Querying

When Remote Querying is enabled, the Streams Replication Manager Service acting as the designated monitoring gateway generates additional data traffic. This data traffic is generated as a result of the gateway communicating with other, remote Streams Replication Manager Services and receiving data on remote replications. In the case of:

- A single replication flow.
- A single replicated topic with 1000 partitions.
- A single consumer group.
- One hour retention time set for the Streams application.

The maximum amount of data traffic generated by Remote Querying (the data returned by other Streams Replication Manager Services) is around 70 kB.

Cloudera Runtime 7.3.1

Securing Streams Replication Manager

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Streams Replication Manager security overview.....4

Enabling TLS/SSL for the Streams Replication Manager service.....5

Enabling Kerberos for the Streams Replication Manager service..... 7

Configuring Basic Authentication for the Streams Replication Manager service.....7

Enabling Basic Authentication for the Streams Replication Manager service..... 7

Configuring Basic Authentication for Streams Replication Manager Services that have Remote Querying enabled.....9

Streams Replication Manager security example..... 12

Streams Replication Manager security overview

Configuring Streams Replication Manager security involves enabling and setting security-related features and properties for the Streams Replication Manager service (Driver and Service roles) and the srm-control command line tool. This permits Streams Replication Manager to access source and target clusters and replicate data between them. In addition, it also enables the subcomponents of Streams Replication Manager that act as servers to function in a secure way.

Streams Replications Manager functions both as a client and a server. When Streams Replication Manager replicates data and connects to Kafka clusters it functions as a client. In addition however, some processes and components of Streams Replication Manager act as servers. For example the Streams Replication Manager Service role spins up a REST server. Similarly, the Streams Replication Manager Driver role has replication specific Connect REST servers which provide background functionality.

As a result of this, configuring security for Streams Replication Manager has two distinct aspects as you can configure security for Streams Replication Manager not only when it acts as client, but also when it acts as a server.

Server configuration

Security for Streams Replication Manager processes and components that act as servers can be configured by enabling the TLS/SSL and/or Kerberos feature toggles as well as configuring key and truststore related properties available in Cloudera Manager. For more information see, *Enable TLS/SSL for the Streams Replication Manager service*or *Enable Kerberos authentication for the Streams Replication Manager service*

Note: While the security feature toggles enable security for the components of Streams Replication Manager that function as servers, they are multi-purpose and also configure the security properties that are needed for Streams Replication Manager to connect, as a client, to the co-located clusters. As a result of this, if you have followed *Defining and adding clusters for replication* and have a secured co-located cluster, it is likely that security is already set up for the components that act as servers.

Client configuration

Configuring security for Streams Replication Manager when it functions as a client involves enabling and setting security-related features and properties for the Streams Replication Manager service and the srm-control command line tool. This permits both the service and the tool to access your source and target clusters and replicate data between them. The configuration workflow and the steps you need to take differ for the service and tool.

Streams Replication Manager service

Before you can start replicating data with Streams Replication Manager, you must define the clusters that take part in the replication process and add them to the configuration of Streams Replication Manager. When you define a cluster, you also specify the required keys, certificates, and credentials needed to access the clusters that you define. This means that when you define a cluster for replication, at the same time, you also configure the necessary security-related properties needed for Streams Replication Manager to connect as a client.

The clusters and their security properties can be defined in multiple ways. What method you use depends on the clusters in your deployment and the type of security these clusters use. For more information regarding the workflow and the available options for configuration, see *Defining and adding clusters for replication*.

srm-control tool

In addition to configuring security for the Streams Replication Manager service, you also need to configure security related properties for the srm-control tool. This is because the srm-cont rol tool also functions as a client and must have access to the clusters in the deployment. The exact configuration steps you need to take depends on how your deployment is set up, how you configured the service, and the type of security used on the clusters. For more information regarding the workflow and the available options for configuration, see *Configuring srm-control*.

Related Information

Enabling TLS/SSL for the Streams Replication Manager service
Defining and adding clusters for replication
Configuring srm-control
Enabling Kerberos for the Streams Replication Manager service

Enabling TLS/SSL for the Streams Replication Manager service

TLS/SSL can be enabled and configured for the Streams Replication Manager service (Driver and Service roles) with various configuration properties available in Cloudera Manager. Configuring these properties affects the security configuration of Streams Replication Manager in multiple ways.

About this task

Both the Driver and Service roles of Streams Replication Manager have a number of TLS/SSL related properties associated with them. A dedicated TLS/SSL feature toggle exists for both roles. These are the Enable TLS/SSL for SRM Driver and Enable TLS/SSL for SRM Service properties. In addition to the feature toggles, there are a number of other properties that can be used to configure key and truststore information.

Configuring the feature toggles and the key/truststore related properties have the following effects on Streams Replication Manager's security configuration:

- The Streams Replication Manager Service role's REST server becomes secured and uses HTTPS.
- The Streams Replication Manager Driver role's replication specific Connect REST servers become secured and use HTTPS. In addition, client authentication will also be required from any client connecting to these servers.

  Note: The replication specific Connect REST servers are for internal use only. They enable communication between Driver role instances. Third party clients should not interface with them.

- If the deployment has a co-located Kafka cluster and that cluster was configured using a service dependency, both the Service and Driver roles will use the keystore and truststore information when they establish a connection with the co-located Kafka cluster.
- Both the Driver and Service roles will use these properties as fallback configurations when establishing a connection to a Kafka cluster.

  That is, if there is a Kafka cluster in your configuration that has its protocol specified as SSL, but no trust or keystore information is set for it, the roles will use the truststore and keystore configured with these properties.

Important: Configuring the feature toggles and key/truststore properties on their own do not enable Streams Replication Manager to connect to or replicate a TLS/SSL enabled external Kafka cluster. They also do not have an effect on the srm-control tool's security configuration.

Configuring these properties is part of the process of defining and adding clusters described in *Defining and adding clusters for replication*. Depending on how you set up your co-located cluster, these properties might already be configured.

If you configured the co-located cluster with a service dependency, then these properties are configured in your deployment and no additional steps are needed to enable or configure TLS/SSL.

However, if you chose to set up the co-located cluster with a Kafka credential, these properties might not be configured. Although in a case like this the properties required by Streams Replication Manager to access the co-located cluster will be set, the server functionality of the roles will not be TLS/SSL enabled. Additionally, while not crucial, no fallback properties will be set up for security either. In a case like this Cloudera recommends that you complete the following steps.

Procedure

1. In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
2. Go to Configuration.
3. Find and configure the following properties based on your cluster and requirements:

Table 1:

Cloudera Manager Property	Description
Enable TLS/SSL for SRM Driver	Encrypt communication between clients and Streams Replication Manager Driver using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).
SRM Driver TLS/SSL Server JKS Keystore File Location	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when Streams Replication Manager Driver is acting as a TLS/SSL server. The keystore must be in JKS format.
SRM Driver TLS/SSL Server JKS Keystore File Password	The password for the Streams Replication Manager Driver JKS keystore file.
SRM Driver TLS/SSL Server JKS Keystore Key Password	The password that protects the private key contained in the JKS keystore used when Streams Replication Manager Driver is acting as a TLS/SSL server.
SRM Driver TLS/SSL Trust Store File	The location on disk of the trust store, in .jks format, used to confirm the authenticity of TLS/SSL servers that Streams Replication Manager Driver might connect to. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
SRM Driver TLS/SSL Trust Store Password	The password for the Streams Replication Manager Driver TLS/SSL Trust Store File. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.
Enable TLS/SSL for SRM Service	Encrypt communication between clients and Streams Replication Manager Service using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL))
SRM Service TLS/SSL Server JKS Keystore File Location	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when Streams Replication Manager Service is acting as a TLS/SSL server. The keystore must be in JKS format.
SRM Service TLS/SSL Server JKS Keystore File Password	The password for the Streams Replication Manager Service JKS keystore file.
SRM Service TLS/SSL Server JKS Keystore Key Password	The password that protects the private key contained in the JKS keystore used when Streams Replication Manager Service is acting as a TLS/SSL server.
SRM Service TLS/SSL Trust Store File	The location on disk of the trust store, in .jks format, used to confirm the authenticity of TLS/SSL servers that Streams Replication Manager Service might connect to. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
SRM Service TLS/SSL Trust Store Password	The password for the Streams Replication Manager Service TLS/SSL Trust Store File. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.

4. Click Save Changes.
5. Restart the Streams Replication Manager service.

Related Information

Defining and adding clusters for replication

Enabling Kerberos for the Streams Replication Manager service

Kerberos can be enabled and configured for the Streams Replication Manager service with the Enable Kerberos Authentication Cloudera Manager property.

About this task

Kerberos authentication can be enabled for the Streams Replication Manager service (Driver and Service roles) with the Enable Kerberos Authentication property. Enabling this property generates a JAAS configuration that is used by default for all SASL connections made by the Streams Replication Manager service. That is, Streams Replication Manager will connect to the co-located Kafka cluster using this JAAS configuration. Additionally, Streams Replication Manager will fall back to using this JAAS configuration for all other SASL connections where a connection (cluster) specific configuration is not configured.

Important: Configuring the Enable Kerberos Authentication on its own does not enable Streams Replication Manager to connect to or replicate a Kerberos enabled external Kafka cluster. Additionally, configuring the property does not have an effect on the srm-control tool's security configuration.

Configuring this property is part of the process of defining and adding clusters described in *Defining and adding clusters for replication*. Depending on how you set up your co-located cluster, the property might already be enabled.

Procedure

1. In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
2. Go to Configuration.
3. Find and enable the Enable Kerberos Authentication property
4. Click Save Changes.
5. Restart the Streams Replication Manager service.

Related Information

Defining and adding clusters for replication

Configuring Basic Authentication for the Streams Replication Manager service

The REST API of the Streams Replication Manager Service can be secured using Basic Authentication (BA). Once enabled, all clients and services will be required to present valid credentials. Configuration is done in Cloudera Manager using Streams Replication Manager configuration properties and external accounts.

Enabling Basic Authentication for the Streams Replication Manager service

Basic Authentication (BA) can be enabled for the Streams Replication Manager Service in Cloudera Manager. After BA is enabled, the REST API of the Streams Replication Manager Service becomes secured. Any clients or services accessing the REST API will need to present valid credentials for access.

About this task

BA is set up for the Streams Replication Manager Service by creating Basic Authentication Credentials. A Basic Authentication Credential is an item that securely stores a username/password pair that can be used by the Streams Replication Manager Service for BA. Once the credential is defined, you must turn on BA for the Streams Replication Manager Service and reference the credentials you created in the configuration of Streams Replication Manager. After configuration is complete, the Streams Replication Manager Service and its Rest API will only be accessible by clients and services that present valid credentials.

In addition to the credentials configured with Basic Authentication Credentials, another username/password pair is generated automatically. This username and password pair is used by other services that depend on Streams Replication Manager and are deployed in the same cluster. These credentials are automatically passed to the dependent services. This way, any co-located dependent service automatically has access to the Streams Replication Manager REST API when BA is enabled. For example, Streams Messaging Manager is a service like this. It provides replication monitoring by accessing and gathering metrics from the Streams Replication Manager REST API. As a result it requires access to the REST API. If required, the automatically generated credentials can be updated. However their configuration is optional.

Procedure

1. Create Basic Authentication Credentials:

Streams Replication Manager supports BA for multiple users. This means that you can create more than one credential. Cloudera recommends that you create at least two. One for any external services and clients, and one that can be used by Streams Replication Manager internally.

a) In Cloudera Manager go to Administration External Accounts Basic Authentication Credentials .
b) Click Add Basic Authentication Credentials.
c) Configure the following properties:

- Name

  Add a unique and easily identifiable name. Note down the name you configure, you will need to provide it in a later step.
- Username and Password

  Add a username and password. These credentials will be accepted by Streams Replication Manager when an external client or service tries to access the REST API. Take note of the username and password that you configure. Depending on your cluster and setup, you might need to provide these credentials when configuring other Streams Replication Manager features or external clients and services that access the Streams Replication Manager REST API.

2. Enable BA for the Streams Replication Manager Service:

a) Go to Clusters and select the Streams Replication Manager Service.
b) Go to Configuration.
c) Find and enable the SRM Service Use Basic Authentication property.
d) Find and configure the External Basic Authentication Accounts Accepted By SRM Service property.

Add the names of all Basic Authentication Credentials that you created in Step 1. For example:

```
external_1
external_2
internal_1
```

e) Find and configure the SRM Service Intra Cluster Account Name property.

This property controls which credential should be used by the individual Streams Replication Manager Service roles within the cluster to communicate with each other. Add one of the Basic Authentication Credential

names that you added to the External Basic Authentication Accounts Accepted By SRM Service property. For example:

```
internal_1
```

3. Configure the automatically generated credentials used by dependent services.

   The automatically generated username and password can be configured with the following properties:

   - SRM Service Co-Located Service Username
   - SRM Service Co-Located Service User Password
4. Click Save Changes.
5. Restart Streams Replication Manager.

Results

BA is configured for the Streams Replication Manager Service

What to do next

- If you have previously enabled Remote Querying for a separate Streams Replication Manager Service that targets this Streams Replication Manager Service (the one that you enabled BA for) with Remote Querying, complete *Configuring Basic Authentication for Remote Querying* for the Streams Replication Manager Service that has Remote Querying enabled.
- Query metrics. You can do either of the following:
  - Access the **Replications** page on the Streams Messaging Manager UI. Replications will be visible in the UI.
  - Query metrics using the Streams Replication Manager REST API. For example:
    1. Go to Streams Replication Manager Web UI SRM Service Swagger UI .
    2. Find and open the /v2/replications endpoint.
    3. Click Try it out then click Execute.

       You are prompted to enter a username and password.
    4. Enter the credentials you configured using Basic Authentication Credentials.

       The response includes all discovered replications, replicated topics, and various other metrics.

Related Information

Configuring Basic Authentication for Remote Querying

Configuring Basic Authentication for Streams Replication Manager Services that have Remote Querying enabled

If a Streams Replication Manager Service has Remote Querying enabled, it will connect to other, remote Streams Replication Manager Services to fetch the remote cluster replication metrics. If the remote Streams Replication Manager Service has Basic Authentication (BA) enabled, the Streams Replication Manager Service that has Remote Querying enabled must be provided with valid credentials that it can use to connect to the remote Streams Replication Manager Service.

About this task

Remote Querying is a feature of the Streams Replication Manager Service that enables you to configure a Streams Replication Manager Service in your deployment to connect to other, remote Streams Replication Manager Services and fetch the remote cluster replication metrics gathered by the remote Service. This allows you to monitor the replications of a whole deployment that has multiple installations of Streams Replication Manager through a single Streams Replication Manager and Streams Messaging Manager service. For more information on Remote Querying, see *Remote Querying*.

If the remote Streams Replication Manager Services are configured to use BA, the Streams Replication Manager Services that have Remote Querying enabled (monitoring gateways) require valid credentials that can be used to access the remote Streams Replication Manager Services. As a result, the credentials required to access the remote Streams Replication Manager Services must be provided to the monitoring gateways.

Configuration is done in Cloudera Manager by creating Streams Replication Manager Service Credentials and then configuring the monitoring gateways to use these credentials for Remote Querying.

Before you begin

- Basic authentication is set up and configured for the remote Streams Replication Manager Services. For more information on how to enable Basic authentication see, Enabling Basic Authentication for the Streams Replication Manager Service.

  Note: Your deployment might include multiple remote Streams Replication Manager Services and some of them might not have BA enabled. In a case like this, you only need to create and configure credentials for the remote services that have BA enabled. Unsecured remote Streams Replication Manager Services can be remotely queried without additional configuration.
- Remote querying is enabled and configured. For more information see, Enabling Remote Querying
- Ensure that you have access to the BA usernames and passwords used by the remote Streams Replication Manager Services. You will need to provide these during configuration. These usernames and passwords were configured using Basic Authentication Credentials when you enabled BA for the remote Streams Replication Manager Services.

Procedure

1. Access the Cloudera Manager instance managing the Streams Replication Manager Service that has Remote Querying enabled (monitoring gateway).
2. Create Streams Replication Manager Credentials:

   A Streams Replication Manager Credential represents a remote Streams Replication Manager Service that has Basic Authentication enabled. The credential stores the information necessary to establish a connection with the remote Streams Replication Manager Service. Create a credential for each BA enabled remote Streams Replication Manager Service.

   a) Go to Administration External Accounts Streams Replication Manager Credentials .
   b) Click Add Streams Replication Manager Service Credentials.
   c) Configure the following properties:

      - Name

        Add a unique and easily identifiable name. Note down the name you configure, you will need to provide it in a later step.
      - Remote Streams Replication Manager Service target Kafka cluster alias

        This is the alias of the Kafka cluster that the remote Streams Replication Manager Service (which this credential represents) is targeting. The alias that you add here must be one of the aliases that you added to the Streams Replication Manager Service Remote Target Clusters property when you enabled Remote Querying.
      - Username and Password

        Add the username and password pair that can be used to access the remote Streams Replication Manager Service that this credential represents. Both the username and password that you add here must be valid. That is, in the Cloudera Manager instance managing the remote Streams Replication Manager Service, a

Basic Authentication credential must exist with this username and password combination. Additionally, the remote Streams Replication Manager Service must be configured to use these credentials.

To better understand how Streams Replication Manager Service Credentials should be configured, review the following two cluster example:

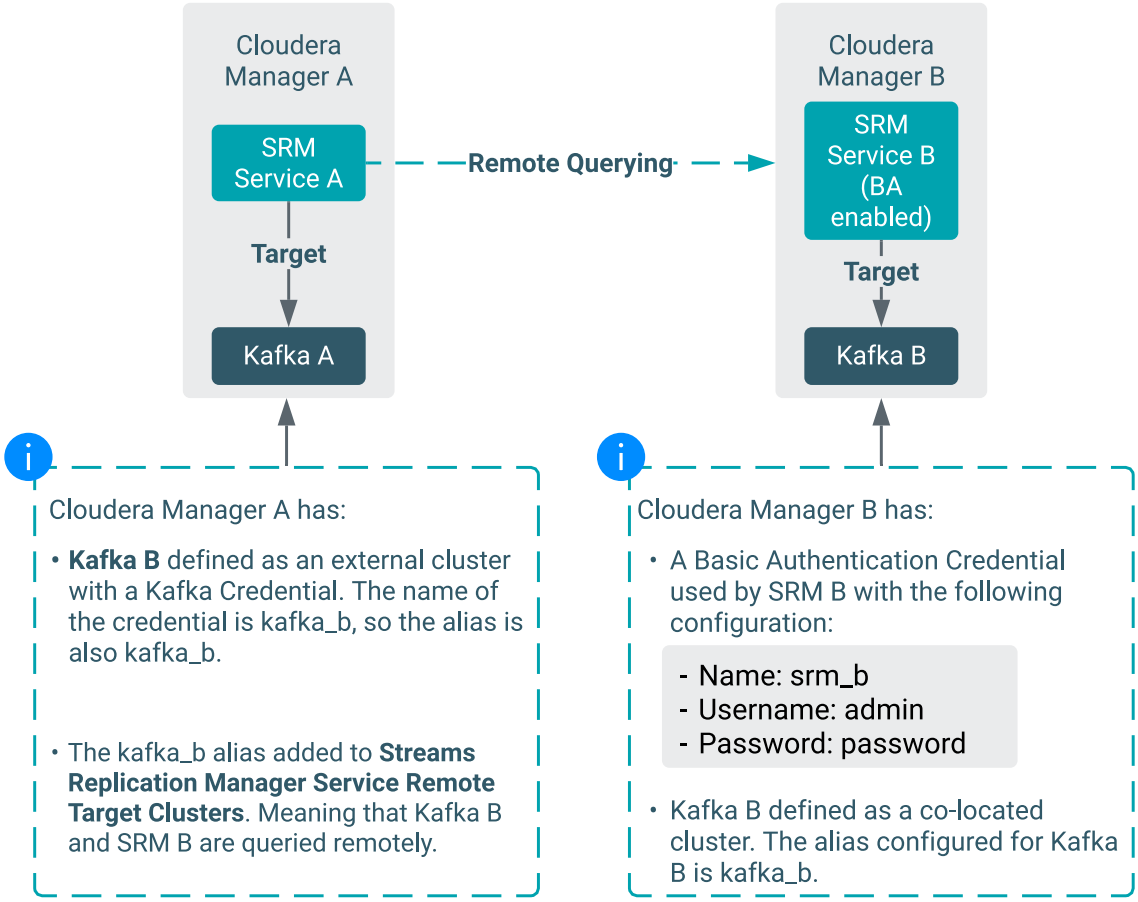

In this specific example, a single Streams Replication Manager Service Credential must be created in Cloudera Manager A. This credential represents Streams Replication Manager Service B and will be used by Streams Replication Manager Service A to access Streams Replication Manager Service B. The credential would be configured as follows:

```
Name: srm_b
Remote Streams Replication Manager Service target Kafka cluster alias: k
afka_b
Username: admin
Password: password
```

3. Configure this Streams Replication Manager Service (monitoring gateway) to use the credentials when querying the remote Streams Replication Manager Service:
   a) Go to Clusters and select the Streams Replication Manager Service.
   b) Go to Configuration.
   c) Find and Configure the External Accounts For Remote Querying SRM Services property.
   d) Add the names of all Streams Replication Manager Service Credentials you created in Step 1. For example:

```
remote_1
```

```
remote_2
remote_3
```

4. Click Save Changes.
5. Restart Streams Replication Manager.

Results

The credentials required to access the remote Streams Replication Manager Services are set up and configured. When this Streams Replication Manager Service establishes a connection with the configured remote Streams Replication Manager Services, it uses the configured credentials.

What to do next

- Access the **Replications** page on the Streams Messaging Manager UI. Remote replications will be visible in the UI.
- Query metrics using the Streams Replication Manager REST API. For example:
  1. Go to Streams Replication Manager Web UI SRM Service Swagger UI.
  2. Find and open the /v2/replications endpoint.
  3. Click Try it out then click Execute.

     The response includes all discovered replications, replicated topics, and various other metrics. This includes replications that target remote clusters.

Related Information

Remote Querying

Streams Replication Manager security example

Streams Replication Manager is capable of replicating data between Cloudera and non-Cloudera clusters that are secured. A setup like this involves configuring security for the Streams Replication Manager service (Driver and Service roles) and the srm-control command line tool.

About this task

Consider the following replication environment.

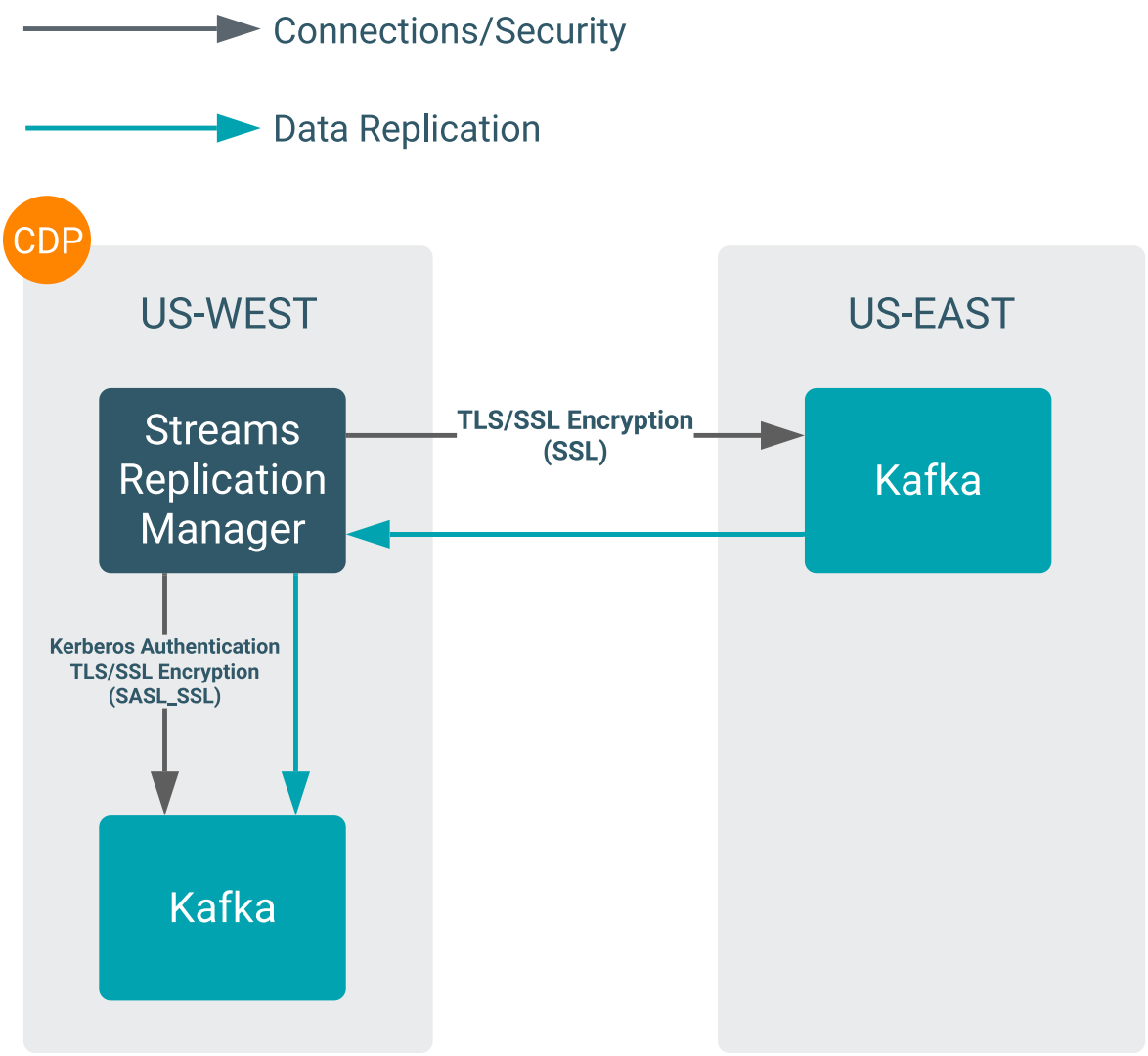

There are two clusters, US-West and US-East. US-West is a Cloudera cluster that has both Kafka and Streams Replication Manager deployed on it. The Kafka service on this cluster is both Kerberos and TLS/SSL enabled. That is, clients connecting to this cluster (including Streams Replication Manager) use the SASL_SSL protocol.

US-East has Kafka deployed on it but not Streams Replication Manager. Kafka on this cluster has TLS/SSL encryption enabled. More importantly, the platform that this cluster is running on is not defined. This is because this example is meant to demonstrate that Streams Replication Manager can connect to and replicate data to or from non-Cloudera clusters. As long as that cluster is running Kafka, Streams Replication Manager will be able to connect to it.

Data replication is unidirectional. The Streams Replication Manager service deployed in US-West is replicating Kafka data from US-East to US-West. From the perspective of the Streams Replication Manager service, US-West is its co-located Kafka cluster, while US-East is an external Kafka cluster.

The following example walks you through the steps required to set up both the Streams Replication Manager service and the srm-control tool for this replication environment.

Before you begin

The following steps assume that the Streams Replication Manager service is already installed and available on US-West. Additionally, it is also assumed that key and truststore files as well as other credentials required to establish a connection with each cluster are known and are available. The instructions do not go into detail on how you can generate or acquire these credentials.

Procedure

1. Define the external Kafka cluster (US-East):
   a) Log into the Cloudera Manager instance managing US-West.
   b) Go to AdministrationExternal Accounts.
   c) Go to the Kafka Credentials tab.
   d) Click Add Kafka credentials.
   e) Create a Kafka credential for US-East:

      The security configuration of the external cluster determines which of the available properties you must set. Because in this example US-East has TLS/SSL encryption enabled, the following properties must be set:

      - Name
      - Bootstrap servers
      - Security protocol
      - Truststore Password
      - Truststore Path
      - Truststore Type

      Note: Click the i icon next to each property in Cloudera Manager to reveal additional information about each property.

   f) Click Add.

      If credential creation is successful, a new entry corresponding to the Kafka credential you specified appears on the page.
2. Define the co-located Kafka cluster (US-West):

   Co-located Kafka clusters can be defined in two separate ways. Either with a service dependency or with Kafka credentials. Because in this example US-West is Kerberos and TLS/SSL enabled, the service dependency method is used. This is in-line with Cloudera best practices.

   a) In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
   b) Go to Configuration.
   c) Find and enable the Kafka Service property.
   d) Find and configure the Streams Replication Manager Co-located Kafka Cluster Alias property.

      The alias you configure represents the co-located cluster. Enter an alias that is unique and easily identifiable. For example:

      ```
      uswest
      ```

   e) Enable relevant security feature toggles

      Because US-West is both TLS/SSL and Kerberos enabled, you must enable all feature toggles for both the Driver and Service roles. The feature toggles are the following:

      - Enable TLS/SSL for SRM Driver
      - Enable TLS/SSL for SRM Service
      - Enable Kerberos Authentication

3. Add both clusters to the configuration of Streams Replication Manager:
   a) Find and configure the External Kafka Accounts property.

      Add the name of the Kafka credential you created for US-East to this property. This can be done by clicking the add button to add a new line to the property and then entering the name of the Kafka credential. For example:

      ```
      useast
      ```

   b) Find and configure the Streams Replication Manager Cluster alias property.

      Add all cluster aliases to this property. This includes the aliases present in both the External Kafka Accounts and Streams Replication Manager Co-located Kafka Cluster Alias properties. Delimit the aliases with commas. For example:

      ```
      useast, uswest
      ```

4. Configure replications:

   In this example, data is replicated unidirectionally from US-West to US-East. As a result, only a single replication must be configured.

   a) Find the Streams Replication Manager's Replication Configs property.
   b) Click the add button and add new lines for each unique replication you want to add and enable.
   c) Add and enable your replications.

      For example:

      ```
      useast->uswest.enabled=true
      ```

5. Configure Driver and Service role targets:
   a) Find and configure the Streams Replication Manager Service Target Cluster property.

      Add the co-located cluster’s alias to the property. In the case of this example:

      ```
      useast
      ```

   b) Find and configure the Streams Replication Manager Driver Target Cluster property.

      This property should contain all cluster aliases. In the case of this example, you can either add the aliases of both US-West and US-East or leave the property empty. Leaving the property empty is the same as adding all aliases.

6. Click Save Changes.
7. Restart the Streams Replication Manager service.

8. Configure and run the srm-control tool:
   a) Click Gateway in the Filters pane.
   b) Find and configure the following properties:

      SRM Client's Secure Storage Password

      The password used to access the secure storage. Take note of the password you configure. You need to provide it in your CLI session before running the tool.

      Environment Variable Holding SRM Client's Secure Storage Password

      The name of the environment variable that stores the secure storage password. Take note of the name that you configure. You need to set it in your CLI session before running the tool.

      Gateway TLS/SSL Trust Store File

      The path to the TLS/SSL truststore file containing the server (co-located cluster's) certificate and public key. Ensure that this file is available on all Streams Replication Manager hosts.

      Gateway TLS/SSL Truststore Password

      The password used to access the truststore file specified in the Gateway TLS/SSL Trust Store File property.

      SRM Client's Kerberos Principal Name

      The kerberos principal that the tool uses for authentication when connecting to the co-located Kafka cluster.

      SRM Client's Kerberos Keytab Location

      The path to the Kerberos keytab file that the tool uses for authentication when connecting to the co-located Kafka cluster.
   c) Click Save Changes.
   d) Re-deploy client configuration.
   e) SSH into one of the Streams Replication Manager hosts in your cluster.
   f) Set the secure storage password as an environment variable.

      ```
      export [***SECURE STORAGE ENV VAR***]="[***SECURE STORAGE PASSWORD***]"
      ```

      Replace *[***SECURE STORAGE ENV VAR***]* with the name of the environment variable you specified in Environment Variable Holding SRM Client's Secure Storage Password. Replace *[***SECURE STORAGE PASSWORD***]* with the password you specified in SRM Client's Secure Storage Password. For example:

      ```
      export SECURESTOREPASS="mypassword
      ```

   g) Run the tool.

      For example:

      ```
      srm-control topics --source USEAST --target USWEST --add TOPIC1,TOPIC2
      ```

Cloudera Runtime 7.3.1

Using Streams Replication Manager

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Streams Replication Manager Command Line Tools.....4
srm-control..... 4
Configuring srm-control..... 4
Topics and Groups Subcommand..... 12
Offsets Subcommand..... 14

Monitoring Replication with Streams Messaging Manager..... 15

Replicating Data.....16

How to Set up Failover and Failback.....16
Configure Streams Replication Manager for Failover and Failback..... 17
Migrating Consumer Groups Between Clusters..... 18

Streams Replication Manager Command Line Tools

Overview of the command line tools shipped with Streams Replication Manager.

Important: Streams Replication Manager ships with the srm-control, srm-driver, srm-service, and srm command line tools. Use the srm-control tool to the manage replication of topics and consumer groups. Do not use the srm-driver, srm-service, and srm tools, use Cloudera Manager instead to manage the Streams Replication Manager Driver and Service.

srm-control

Learn how to use the srm-control command line tool which is used to manage replication of topics and consumer groups.

The srm-control tool enables users to manage replication of topics and consumer groups. The tool has three subcommands topics, groups, and offsets. The topics subcommand is used to control which topics are replicated. The groups subcommand is used to control which consumer groups are replicated. The offsets subcommand is used to export translated offsets for a source->target cluster pair.

The srm-control command line tool is located in /opt/cloudera/parcels/CDH/bin. An alternative is provided for it by default, it is called srm-control. Simply typing srm-control will invoke the tool, you do not need to reference its full path.

Important: Do not use the srm-driver, srm-service, and srm command line tools which are also located in the bin directory. Instead, use Cloudera Manager to manage the Streams Replication Manager driver and service.

If any of the Kafka clusters that Streams Replication Manager connects to use any type of encryption or authentication, you must configure the srm-control tool before use. For more information, see *Configuring srm-control*.

A full list of the available options for the tool can be found in *srm-control Options Reference*.

Related Information

srm-control Options Reference

Configuring srm-control

Cloudera Manager automatically generates a configuration file for the srm-control tool. If both the co-located and external Kafka clusters are unsecured, the default configuration can be used, without making any changes. If however, any of the clusters taking part in the replication process use any type of security, additional configuration is required.

The srm-control tool functions similarly to any Kafka client. It reads data from and writes data to Kafka topics. More specifically, it manipulates the internal configuration topic of Streams Replication Manager that contains the replication allow and deny lists. This internal topic is stored within Kafka.

The srm-control tool requires a properties file which specifies information about all clusters taking part in the replication process. That is, the properties file contains the cluster names (aliases), the bootstrap servers, and security related properties of each cluster.

This information, however, is not unique to the tool, as the Streams Replication Manager service (Driver and Service roles) also requires these properties to be set. Therefore, the configuration of the tool can be viewed as a subset of the Streams Replication Manager service's configuration. Because of this and because the Streams Replication Manager service is configured with Cloudera Manager, Cloudera Manager is capable of automatically creating a configuration file for the tool based on the configuration of the Streams Replication Manager service.

The configuration file is located at /etc/streams_replication_manager/conf/srm.properties. By default the tool uses this configuration file.

Important: Do not edit this configuration file directly. Only use it to verify your configuration or to troubleshoot issues. If you want to make changes manually, always create a copy, edit it as needed, and run the tool with the --config option to specify the location of the copy.

The default configuration generated by Cloudera Manager is dynamic. It is updated any time you deploy the client configuration for Streams Replication Manager. For example, if you add a new cluster for replication or change an existing one, the changes you made are automatically added to the default tool configuration once client configuration is deployed.

This automation simplifies the process of configuring the tool. If the Kafka clusters that Streams Replication Manager is connecting to are not secured, then no additional configuration is needed. The default will contain all necessary properties. In cases like this you only need to ensure that the Streams Replication Manager service is correctly configured.

However, if any of the Kafka clusters use any type of encryption or authentication, additional configuration is required. This is because the generated configuration by default does not contain any sensitive TLS/SSL or SASL properties required to access the Kafka clusters. Providing this information to the tool requires additional configuration by the user.

There are five configuration tasks related to setting up sensitive properties for the tool. These are as follows:

- *Configuring the Streams Replication Manager client's secure storage*
- *Configuring TLS/SSL properties*
- *Configuring Kerberos properties*
- *Configuring properties for non-Kerberos authentication mechanisms*
- *Setting the secure storage password as an environment variable*

Which of these tasks you need to complete depends on the security configuration of the clusters that Streams Replication Manager connects to and the method used to define the co-located clusters. Review the following flowchart to see which of the tasks you need to complete and in what order.

Note: If there is no co-located Kafka cluster in your deployment, choose **No** where the chart asks about the co-located Kafka cluster.

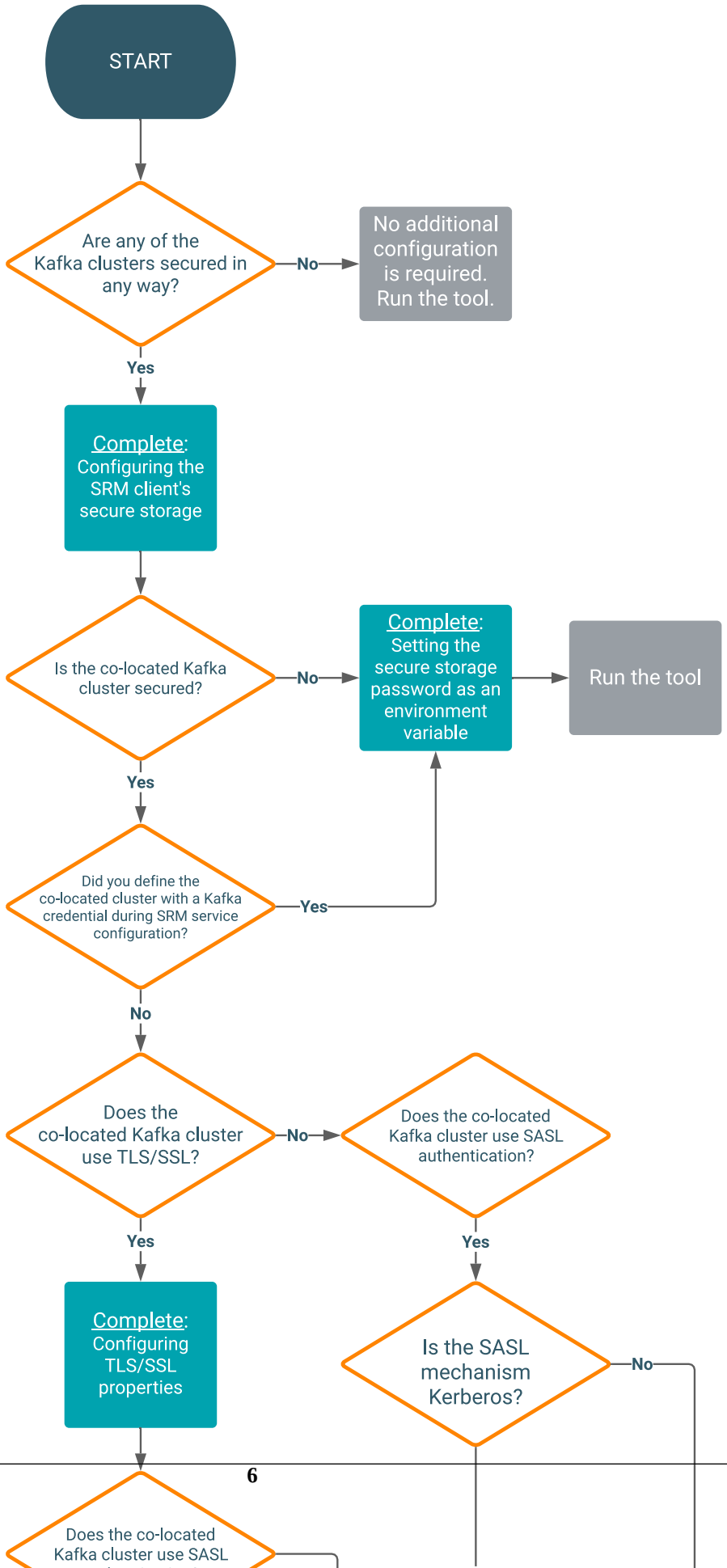
START
Are any of the Kafka clusters secured in any way?
No
No additional configuration is required. Run the tool.
Yes
Complete: Configuring the SRM client's secure storage
Is the co-located Kafka cluster secured?
No
Complete: Setting the secure storage password as an environment variable
Run the tool
Yes
Did you define the co-located cluster with a Kafka credential during SRM service configuration?
Yes
No
Does the co-located Kafka cluster use TLS/SSL?
No
Does the co-located Kafka cluster use SASL authentication?
Yes
Yes
Complete: Configuring TLS/SSL properties
Is the SASL mechanism Kerberos?
No
Does the co-located Kafka cluster use SASL

Configuring the Streams Replication Manager client's secure storage

The Streams Replication Manager client's secure storage is an intermediary keystore used to store sensitive security properties that are required to access the Kafka clusters that Streams Replication Manager connects to. If any of these clusters are secured, you need to set up and configure the secure storage, otherwise the srm-control tool will not function.

About this task

Based on the Streams Replication Manager service's configuration, Cloudera Manager automatically generates a default configuration file that is used by the srm-control tool. The default configuration, however, does not contain any sensitive data (keystores, truststores, passwords) required to access the clusters. Instead, sensitive details are added to Streams Replication Manager's secure storage, which is an intermediary keystore used to store sensitive data. Specifically, this secure storage will contain all the sensitive information required to access the Kafka clusters that Streams Replication Manager connects to and replicates.

The Streams Replication Manager client's secure storage is not automatically created or populated with all the necessary information. As a result, before you can start using the tool, you must set up and configure the secure storage. This is done by configuring various configuration properties of the Streams Replication Manager service in Cloudera Manager.

Before you begin

- Ensure that you have reviewed the information available in Configuring srm-control and understand that the following step list is only one part of the full configuration workflow. Depending on your scenario, completing other configuration tasks might be required.
- Ensure that setup and configuration of the Streams Replication Manager service is complete:
  - The Kafka clusters that Streams Replication Manager connects to are defined and are added to the configuration of the Streams Replication Manager service. This includes both external and co-located clusters.
  - Replications are configured.

Procedure

1. In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
2. Go to Configuration.
3. Click Gateway in the Filters pane.
4. Find and configure the following properties:

   SRM Client's Secure Storage Password

   The password used to access the secure storage. Take note of the password you configure. You need to provide it in your CLI session before running the tool.

   Environment Variable Holding SRM Client's Secure Storage Password

   The name of the environment variable that stores the secure storage password. Take note of the name that you configure. You need to set it in your CLI session before running the tool.
5. Click Save Changes.
6. Re-deploy client configuration.

What to do next

If your co-located cluster is secured, you must manually configure the sensitive properties required to access the co-located cluster. Continue with one of the following tasks depending on the security configuration of the co-located cluster.

- *Configuring TLS/SSL properties*
- *Configuring Kerberos properties*
- *Configuring properties for non-Kerberos authentication mechanisms.*

If the co-located cluster is not secured, continue with:

- *Setting the secure storage password as an environment variable*

Related Information

Configuring TLS/SSL properties

Configuring Kerberos properties

Configuring properties for non-Kerberos authentication mechanisms

Setting the secure storage password as an environment variable

Configuring TLS/SSL properties

The Streams Replication Manager client's secure storage is automatically populated with the sensitive properties needed by the srm-control tool to access clusters defined with Kafka credentials. However, if the co-located cluster uses TLS/SS and was defined through a service dependency, the TLS/SSL properties of the co-located cluster must be configured and added to the secure storage manually.

Before you begin

- Ensure that you have reviewed the information available in Configuring srm-control and understand that the following step list is only one part of the full configuration workflow. Depending on your scenario, completing other configuration tasks might be required.
- Ensure that setup and configuration of the Streams Replication Manager service is complete:
  - The Kafka clusters that Streams Replication Manager connects to are defined and are added to the configuration of the Streams Replication Manager service. This includes both external and co-located clusters.
  - Replications are configured.
- If TLS/SSL encryption is enabled on the co-located Kafka cluster:
  - Generate or acquire a truststore containing the certificate of the Certificate Authority that issued the co-located Kafka cluster's certificate.
  - Note down the password of the truststore. You will need to provide it during configuration.
- If both TLS/SSL encryption and authentication are enabled on the co-located Kafka cluster:
  - Generate or acquire a key and truststore which contain all necessary keys and certificates.
  - Note down the locations and passwords for the key and truststores. You will need to provide these during configuration.
  - If the Certificate Authority that issued the tool's certificate is different from the Certificate Authority of the co-located Kafka cluster's certificate, ensure the tool's Certificate Authority certificate is added to the truststore of the co-located Kafka cluster. Otherwise, the co-located Kafka cluster will not trust the certificates used by the tool and a trusted connection will not be established.

Procedure

1. In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
2. Go to Configuration.
3. Click Gateway in the Filters pane.

4. Find and configure the following properties:

   If the co-located cluster uses SSL/TLS encryption only, and no authentication is required, you do not need to configure the keystore related properties from the following list.

   JKS Keystore file Location for the SRM client's TLS/SSL server

   The path to the TLS/SSL keystore file containing the client certificate and private key. Ensure that this file is available on all Streams Replication Manager hosts.

   SRM Client's TLS/SSL Server JKS Keystore File Password

   The password used to access the keystore file specified in the JKS Keystore file Location for the SRM client's TLS/SSL server property.

   SRM Client's TLS/SSL Server JKS Keystore Key Password

   The password used to access the key stored in the keystore file specified in the JKS Keystore file Location for the SRM client's TLS/SSL server property.

   Gateway TLS/SSL Trust Store File

   The path to the TLS/SSL truststore file containing the server (co-located cluster's) certificate and public key. Ensure that this file is available on all Streams Replication Manager hosts.

   Gateway TLS/SSL Truststore Password

   The password used to access the truststore file specified in the Gateway TLS/SSL Trust Store File property.

   Note: The Gateway TLS/SSL Trust Store File and Gateway TLS/SSL Truststore Password properties are automatically filled if you use Auto-TLS.

5. Click Save Changes.
6. Re-deploy client configuration.

What to do next

- If your co-located cluster uses Kerberos authentication, continue with *Configuring Kerberos properties*.
- If your co-located cluster uses a SASL authentication mechanism different from Kerberos, continue with *Configuring properties for non-Kerberos authentication mechanisms*.
- If the co-located cluster does not use SASL authentication continue with *Setting the secure storage password as an environment variable*.

Related Information

Configuring Kerberos properties

Configuring properties for non-Kerberos authentication mechanisms

Setting the secure storage password as an environment variable

Configuring Kerberos properties

The Streams Replication Manager client's secure storage is automatically populated with the sensitive properties needed by the srm-control tool to access clusters defined with Kafka credentials. However, if the co-located cluster uses Kerberos and was defined with a service dependency, its Kerberos properties must be configured and added to the secure storage manually.

Before you begin

- Ensure that you have reviewed the information available in Configuring srm-control and understand that the following step list is only one part of the full configuration workflow. Depending on your scenario, completing other configuration tasks might be required.
- Ensure that setup and configuration of the Streams Replication Manager service is complete:
  - The Kafka clusters that Streams Replication Manager connects to are defined and are added to the configuration of the Streams Replication Manager service. This includes both external and co-located clusters.
  - Replications are configured.

Procedure

1. In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
2. Go to Configuration.
3. Click Gateway in the Filters pane.
4. Find and configure the following properties:

   SRM Client's Kerberos Principal Name

   The kerberos principal that the tool uses for authentication when connecting to the co-located Kafka cluster.

   SRM Client's Kerberos Keytab Location

   The path to the Kerberos keytab file that the tool uses for authentication when connecting to the co-located Kafka cluster.
5. Click Save Changes.
6. Re-deploy client configuration.

What to do next

Continue with *Setting the secure storage password as an environment variable*.

Related Information

Setting the secure storage password as an environment variable

Configuring properties for non-Kerberos authentication mechanisms

If your co-located Kafka cluster uses an authentication mechanism different from Kerberos, and you defined the co-located cluster through a service dependency, you need to copy and edit the srm-control tool's default configuration file generated by Cloudera Manager so that it includes the SASL properties of the co-located cluster.

About this task

The co-located Kafka cluster's properties for SASL mechanisms other than Kerberos cannot be added to the secure storage using Cloudera Manager. Instead, if your co-located Kafka cluster uses an authentication mechanism different from Kerberos, and you defined the co-located cluster through a service dependency, the SASL properties must be directly added to the configuration file that the srm-control tool uses.

This is done by creating a copy of the default configuration file that Cloudera Manager generates and editing the copy so that it contains the necessary SASL properties.

Important: After you complete the following steps, you must run the srm-control tool with --config option to specify the custom configuration file that you created. Otherwise, the tool uses the default configuration and fails when it tries to establish a connection with the co-located cluster.

Before you begin

- Ensure that you have reviewed the information available in Configuring srm-control and understand that the following step list is only one part of the full configuration workflow. Depending on your scenario, completing other configuration tasks might be required.
- Ensure that setup and configuration of the Streams Replication Manager service is complete:
  - The Kafka clusters that Streams Replication Manager connects to are defined and are added to the configuration of the Streams Replication Manager service. This includes both external and co-located clusters.
  - Replications are configured.

Procedure

1. SSH into one of the Streams Replication Manager hosts in your cluster.

2. Create a copy of the default configuration file.

   For example:

```
cp /etc/streams_replication_manager/conf/srm.properties /tmp
```

   Note: The copy of the configuration file that you create contains sensitive information, Cloudera recommends that you protect the copy by setting appropriate system access rights.

   Important: Do not directly edit the default configuration found at /etc/streams_replication_manager/conf/srm.properties. Always create a copy when making changes.

3. Open the copy of the configuration file with an editor.
4. Add the necessary SASL properties to the file.

   The exact properties and values you add to the file depends on the type of SASL authentication used by the co-located Kafka cluster. The following example is for a cluster that uses PLAIN authentication:

```
[***ALIAS***].sasl.mechanism=PLAIN
[***ALIAS***].sasl.jaas.config=${custom_jaas_config}
```

   Replace *[***ALIAS***]* with the co-located cluster's alias. Additionally, Cloudera does not recommend storing the JAAS configuration directly in the file. Instead, store the JAAS configuration in an environment variable and specify the environment variable as the value of the [***ALIAS***].sasl.jaas.config property. This practice is recommended for any other sensitive data that you add to the configuration.

   In a case like this, the environment variable you use to store the sensitive data must be set in your CLI session before running the tool. For example:

```
export custom_jaas_config  = “org.apache.kafka.common.security.plain.Pla
inLoginModule
          required username="[***USER NAME***]" passwo
rd="[***PASSWORD***]";”
```

5. Save the file.

What to do next

Continue with *Setting the secure storage password as an environment variable*.

Related Information

Setting the secure storage password as an environment variable

Setting the secure storage password as an environment variable

If any of the clusters that Streams Replication Manager connects to are secured, and you have set up and configured the Streams Replication Manager client's secure storage, you must set the secure storage password as an environment variable in your CLI session before running the srm-control tool.

About this task

The Streams Replication Manager client's secure storage is password protected. In order for the srm-control tool to access the secure storage, it must know the password protecting the secure storage. This password is specified by the user in Cloudera Manager, however, it cannot be automatically passed to the tool. This is because the tool is not an entity that is managed by Cloudera Manager. As a result, if any of the Kafka clusters in your replication scenario are secured, and you have set up and configured the Streams Replication Manager client's secure storage, you must configure this password for the srm-control tool. This is done by setting the secure storage password as an environment variable in your CLI session before you run the tool.

Before you begin

- Ensure that you have reviewed the information available in Configuring srm-control and understand that the following step list is only one part of the full configuration workflow. Depending on your scenario, completing other configuration tasks might be required.
- Ensure that setup and configuration of the Streams Replication Manager service is complete:
  - The Kafka clusters that Streams Replication Manager connects to are defined and are added to the configuration of the Streams Replication Manager service. This includes both external and co-located clusters.
  - Replications are configured.

Procedure

1. SSH into one of the Streams Replication Manager hosts in your cluster.
2. Set the secure storage password as an environment variable.

```
export [***SECURE STORAGE ENV VAR***]="[***SECURE STORAGE PASSWORD***]"
```

Replace *[***SECURE STORAGE ENV VAR***]* with the name of the environment variable you specified in Environment Variable Holding SRM Client's Secure Storage Password. Replace *[***SECURE STORAGE PASSWORD***]* with the password you specified in SRM Client's Secure Storage Password. For example:

```
export SECURESTOREPASS="mypassword
```

What to do next

Run the srm-control tool. How you run the tool depends on how it was configured:

- If you did not create a copy of the default configuration file you can run the tool without the --config option.

  For example:

```
srm-control topics --source [***SOURCE_CLUSTER***] --targe
t [***TARGET_CLUSTER***] --add [***TOPIC1***],[***TOPIC2***]
```

- If a custom configuration file was created, you must run the tool using the --config option.

  For example:

```
srm-control topics --config [***PATH TO CUSTOM CONFIGURATION FILE***]
 --source [***SOURCE_CLUSTER***] --target [***TARGET_CLUSTER***] --a
dd [***TOPIC1***],[***TOPIC2***]
```

For more information and command examples, see the documentation on the topics, groups, and offsets subcommands.

Related Information

Topics and Groups Subcommand

Offsets Subcommand

Topics and Groups Subcommand

Learn how to use the topics and groups subcommand of the srm-control command line tool.

Important: If any of the Kafka clusters that Streams Replication Manager connects to use any type of encryption or authentication, you must configure the srm-control tool before use. For more information, see Configuring srm-control.

The topics and groups subcommands are used to manipulate the topic or group allowlist (whitelist) and denylist (blacklist). Both subcommands support the same set of command options.

If you are modifying allow and denylists which target newly created topics or consumer groups, changes made with the srm-control tool may not be instantaneous A topic or group needs to be discovered by Streams Replication Manager before it can be added to or removed from allow or denylists. New topic and group discovery happens every 10 minutes by default. As a result, you may need to wait up to 10 minutes until you can see the changes made.

Add topics or groups to an allowlist:

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --a
dd [TOPIC1],[TOPIC2]
```

```
srm-control groups --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --a
dd [GROUP1],[GROUP2]
```

Remove topics or groups from an allowlist:

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --r
emove [TOPIC1],[TOPIC2]
```

```
srm-control groups --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --r
emove [GROUP1],[GROUP2]
```

Add topics or groups to a denylist (blacklist):

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --a
dd-blacklist [TOPIC1],[TOPIC2]
```

```
srm-control groups --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --a
dd-blacklist [GROUP1],[GROUP2]
```

Remove topics or groups from a denylist:

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --r
emove-blacklist [TOPIC1],[TOPIC2]
```

```
srm-control groups --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --r
emove-blacklist [GROUP1],[GROUP2]
```

In addition to adding or removing items, you can also use the tool to look at the contents of a deny or allowlist.

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --l
ist
```

Using Regular Expressions

Specifying topics or groups is also possible with regular expressions. This is done by specifying a regular expression instead of topic names in the --add, add-blackilist, or remove-blacklist options.

For example, assume that you want to replicate all topics from the source cluster to the target cluster. In a case like this, you must add all topics to the allowlist. This can be done with a single srm-control command using the .* regular expression.

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --add
 ".*"
```

You can also use regular expressions to specify a set of topics. For example, assume that you have many topics in your Kafka cluster, but only want to add the following topics to the allowlist.

```
testTopic1
testTopic2
testTopic3
```

In a case like this, you can use the testTopic.* or ^test.* regular expressions to add the topics.

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --add
 "testTopic.*"
```

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --add
 "^test.*"
```

Note: The regular expression must match the whole topic name. For example, testTopic1 can be matched using test.*, testTopic.*, and .*test.*, but cannot be matched with test, ^test, and so on.

Client Override Options

The topics and groups subcommands support a number of client override options. Client override options allow users to temporarily specify or override configuration properties used for replication. These options also enable users to issue srm-control commands even if the configuration of Streams Replication Manager is not available on the host where the command is being issued. While it is possible to specify a range of properties with the client override options, and they can prove to be a powerful tool in certain scenarios, Cloudera recommends that you use Cloudera Manager to manage client configuration options.

The following client override options are available:

- --bootstrap-servers: Specifies the bootstraps servers.
- --producer-props: Specifies producer configuration properties.
- --consumer-props: Specifies consumer configuration properties.
- --props: Specifies client configuration properties.

Note:

Client override options always take precedence over the configuration set in Cloudera Manager. Additionally, the --producer-props and --consumer-props options take precedence over the --props option.

A simple example of using client override options is when you want to change the bootstrap server. This can be done in two ways.

You can specify the bootstrap server with the --bootstrap-servers option.

```
srm-control --bootstrap-servers localhost:9092 topics --sour
ce [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --list
```

Alternatively, you can also use the --props option together with the bootstrap.servers Kafka property to define the bootstrap server.

```
srm-control --props bootstrap.servers=localhost:9092 topics --so
urce [SOURCE_CLUSTER] --list
```

Related Information

New Topic and Consumer Group Discovery

Offsets Subcommand

Learn how to use the offsets subcommand of the srm-client command line tool.

Important: If any of the Kafka clusters that Streams Replication Manager connects to use any type of encryption or authentication, you must configure the srm-control tool before use. For more information, see Configuring srm-control.

Streams Replication Manager automatically translates consumer group offsets between clusters. The offset mappings are created by Streams Replication Manager, but are not applied to the consumer groups of the target cluster directly. Consumers can be migrated from one cluster to another without losing any progress by using the offsets subcommand on the target cluster to export the translated offsets of the source cluster. For example:

```
srm-control offsets --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --g
roup [GROUP1] --export > OUT.CSV
```

Exported offsets can then be applied to the consumer groups in the target cluster with the kafka-consumer-groups tool. Alternatively, this process can be automated by enabling automatic group offset synchronization. If this feature is enabled, the translated offsets are automatically exported from the source cluster and applied to the target cluster.

For detailed steps on cluster migration, see *Migrating Consumer Groups Between Clusters*. For more information regarding automatic group offset synchronization, see *Configuring automatic group offset synchronization*.

Client Override Options

The offset subcommand supports client override options. Client override options allow users to temporarily specify or override configuration properties. These options also enable users to issue srm-control commands even if the configuration of Streams Replication Manager is not available on the host where the command is being issued. While it is possible to specify a range of properties with the client override options, and they can prove to be a powerful tool in certain scenarios, Cloudera recommends that you use Cloudera Manager to manage client configuration options.

The following client override options are available:

- --bootstrap-servers: Specifies the bootstraps servers.
- --props: Specifies client configuration properties.

Note:

Client override options always take precedence over the configuration set in Cloudera Manager.

A simple example of using client override options is when you want to change the bootstrap server. This can be done in two ways.

You can specify the bootstrap server with the --bootstrap-servers option.

```
srm-control --bootstrap-servers LOCALHOST:9092 offsets --sourc
e [SOURCE_CLUSTER] --group [GROUP] --export > OUT.CSV
```

Alternatively, you can use the --props option together with the bootstrap.servers Kafka property to define the bootstrap server.

```
srm-control --props bootstrap.servers=LOCALHOST:9092 offsets --sourc
e [SOURCE_CLUSTER] --group [GROUP] --export > OUT.CSV
```

Related Information

How to Set up Failover and Failback

Migrating Consumer Groups Between Clusters

Configuring automatic group offset synchronization

Monitoring Replication with Streams Messaging Manager

Learn about monitoring Streams Replication Manager replication with Streams Messaging Manager.

Users have the ability to connect Streams Replication Manager with Streams Messaging Manager and monitor replications through the Streams Messaging Manager UI. This is achieved with the Kafka Streams application and the REST API that come bundled with Streams Replication Manager. The Kafka Streams application calculates and aggregates replication metrics, the REST API exposes these metrics. Streams Messaging Manager uses the REST API to display aggregated metrics to the end users, enabling monitoring as a result. Monitoring replication flows in Streams Messaging Manager is available starting with version 2.0.0.

For more information, see *Monitoring Kafka cluster Replications* in the Streams Messaging Manager documentation.

Related Information

Monitoring Kafka cluster replications

Replicating Data

A step by step guide on how to start replicating data between Kafka clusters with Streams Replication Manager.

About this task

Installing and starting Streams Replication Manager on your cluster does not automatically start data replication. In order to start the replication process, you need to update the allowlists with the srm-control tool.

Before you begin

In Cloudera Manager, verify that the Streams Replication Manager driver role is started and is in good health.

Verify that Streams Replication Manager is configured correctly. Make sure that connection information for each Kafka cluster is added as well as at least one source->target replication is specified and enabled.

Procedure

1. Update the topics allowlist to start data replication.

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --
add [TOPIC1],[TOPIC2]
```

Note: If required, instead of listing the topics that you want to add, you can also use regular expressions to add multiple topics with one command.

2. Verify that the topics have been added to the allowlist.

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --
list
```

Results

The topics you specify with the --add option are added to the topic allowlist and are replicated to the specified target cluster.

How to Set up Failover and Failback

Learn how to prepare for failover and failback scenarios with Streams Replication Manager.

If a primary Kafka cluster is temporarily unavailable, you can migrate mission-critical workloads to a backup Kafka cluster (failover). When the primary cluster is restored, you can migrate back (failback). To prepare for this scenario, ensure Streams Replication Manager is configured with bidirectional replication of mission-critical consumer groups and topics. Then in the case of a disaster scenario you can migrate consumer groups between clusters.

Related Information

Offsets Subcommand

Configure Streams Replication Manager for Failover and Failback

Learn how to configure Streams Replication Manager for failover and failback.

About this task

To prepare for a failover or failback scenario you must set up Streams Replication Manager with bidirectional replication. Additionally, you must ensure that all mission critical topics and consumer groups are added to the allowlist on both the primary and backup clusters. Optionally, you can also choose to enable automatic group offset synchronization, which can simplify the steps you need to take when migrating consumer groups.

Before you begin

- Ensure that the Streams Replication Manager service is configured and all clusters are taking part in the replication process are defined and added to the configuration of Streams Replication Manager. For more information, see Defining and adding clusters for replication.
- Ensure that the srm-control tool is configured. For more information, see Configuring srm-control on page 4.
- If you plan on enabling automatic group offset synchronization, ensure that you review Configuring automatic group offset synchronization. Although the basic steps are provided here, the feature has some limitations that you must be aware of.
- The following steps and examples are for a replication scenario with two clusters. These are referred to as primary and secondary.

Procedure

1. Set up bidirectional replication between clusters:
   a) In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
   b) Go to Configuration
   c) Find the Streams Replication Manager's Replication Configs property.
   d) Click the add button and add new lines for each unique replication you want to add and enable.
   e) Add and enable your replications.

      For example, assume that you have two clusters, primary and secondary. To enable bidirectional replication between these clusters, you must add two unique replications. One that replicates from primary to secondary and another that replicates from secondary to primary.

      ```
      primary->secondary.enabled=true
      secondary->primary.enabled=true
      ```

2. Enable and configure automatic group offset synchronization.

   This can be done by enabling sync.group.offsets.enabled. Optionally, if you want to customize the frequency of offset synchronization, you can also set sync.group.offsets.interval.seconds. Both properties are configured by adding them to Streams Replication Manager's Replication Configs. For example:

   ```
   sync.group.offsets.enabled = true
   sync.group.offsets.interval.seconds = [***TIME IN SECONDS***]
   ```

3. Enter a Reason for change, and then click Save Changes to commit the changes.
4. Restart Streams Replication Manager.

5. Add the required consumer groups and topics to the allowlist on the primary cluster.
   - Add groups:

     ```
     srm-control groups --source [PRIMARY_CLUSTER] --targe
     t [SECONDARY_CLUSTER] --add [GROUP1],[GROUP2]
     ```

   - Add topics:

     ```
     srm-control topics --source [PRIMARY_CLUSTER] --targe
     t [SECONDARY_CLUSTER] --add [TOPIC1],[TOPIC2]
     ```

6. Add the required consumer groups and topics to the allowlist on the secondary cluster.

   Important:

   If remote topics and consumer groups are not added to the allowlist on the secondary cluster, a failback operation will be impossible to carry out.

   - Add groups:

     ```
     srm-control groups --source [SECONDARY_CLUSTER] --targe
     t [PRIMARY_CLUSTER] --add [GROUP1],[GROUP2]
     ```

   - Add topics:

     ```
     srm-control topics --source [SECONDARY_CLUSTER] --target [PRIMARY_CLUSTE
     R] --add [PRIMARY_CLUSTER.TOPIC1],[PRIMARY_CLUSTER.TOPIC2]
     ```

7. Verify that all required topics and consumer groups are added to the allowlist.
   - Verify consumer groups:

     ```
     srm-control groups --source [PRIMARY_CLUSTER] --targe
     t [SECONDARY_CLUSTER] --list
     ```

     ```
     srm-control groups --source [SECONDARY_CLUSTER] --targe
     t [PRIMARY_CLUSTER] --list
     ```

   - Verify topics:

     ```
     srm-control topics --source [PRIMARY_CLUSTER] --targe
     t [SECONDARY_CLUSTER] --list
     ```

     ```
     srm-control topics --source [SECONDARY_CLUSTER] --targe
     t [PRIMARY_CLUSTER] --list
     ```

Results

Streams Replication Manager is set up with bidirectional replication and all mission critical topics and consumer groups are added to the allowlist on both the primary and secondary clusters.

Related Information

Configuration Examples

Migrating Consumer Groups Between Clusters

Learn how to migrate consumers between clusters.

About this task

If a primary Kafka cluster is temporarily unavailable, you can migrate mission-critical workloads to a secondary Kafka cluster (failover). When the primary cluster is restored, you can migrate back (failback). The steps for migrating consumers in a failover or failback scenario are identical. However, depending on the scenario, your source and target clusters will be different. During failover you migrate consumers from primary to secondary, while during failback you migrate consumers from secondary to primary.

Before you begin

- Make sure that the clusters that you are migrating consumers between are set up with bidirectional replication.
- Verify that all mission critical consumer groups and topics, including the ones on the secondary cluster are whitelisted.
- The steps you need to take differ depending on whether automatic group offset synchronization is enabled. Check the configuration of Streams Replication Manager to see if the feature is enabled. For more information, see Configuring automatic group offset synchronization

Procedure

1. Migrate consumer group offsets.

   If automatic group offset synchronization is disabled follow the steps provided under Manual migration. If automatic group offset synchronization is enabled, follow the steps under Automatic migration.

   For Manual migration

   a. Export the translated consumer group offsets of the source cluster.

   ```
   srm-control offsets --source [SOURCE_CLUSTER] --targe
   t [TARGET_CLUSTER] --group [GROUP1] --export > OUT.CSV
   ```

   b. Reset consumer offsets on the target cluster.

   ```
   kafka-consumer-groups --bootstrap-server [TARGET_BROKER:PORT] --reset
   -offsets --group [GROUP1] --execute --from-file OUT.CSV
   ```

   For Automatic migration

   If automatic group offset synchronization is enabled, the translated offsets of the source cluster are already applied on the target cluster. Continue with Step 2 on page 19.

   Important: Although translated offsets are applied, depending on the offset group synchronization frequency configured (sync.group.offsets.interval.seconds and emit.checkpoints.interval.seconds), you might have outdated offsets applied. Cloudera recommends that you follow the steps for manual migration if you are unsure whether the latest offsets were applied or if having the latest offsets is critical for your use case.

2. Start consumers on the target cluster.

Results

Consumers automatically resume processing messages on the target cluster where they left off on the source cluster.

Related Information

Offsets Subcommand

Cloudera Runtime 7.3.1

Starting Apache Hive

Date published: 2019-08-21
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Starting Hive on an insecure cluster 4

Starting Hive using a password 4

Accessing Hive from an external node 8

Running a Hive command in Cloudera Data Hub 9

Configuring graceful shutdown property for HiveServer 10

Starting Hive on an insecure cluster

If you want to use Apache Hive for a quick test, you can do so using default Hive default authorization mode, assuming you are on an insecure cluster (no Kerberos or Ranger policies). In default authorization mode, only user hive can access Hive. Steps to start the Hive shell, not to be mistaken with the Hive CLI that Cloudera does not support, include how to log into a cluster.

About this task

From the cluster command line, you type hive on the command line of your cluster to start the Hive shell. In the background, Beeline launches the Hive shell.

Procedure

1. In Cloudera Manager, click Hosts All Hosts .

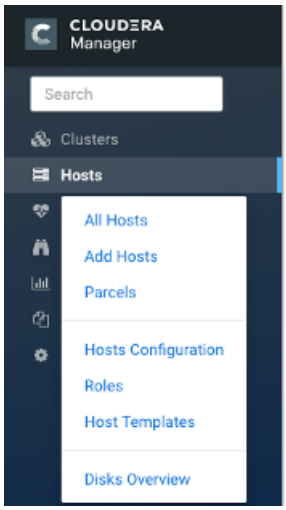

2. Make a note of the IP address or host name of a node in your cluster, for example myhost-vpc.cloudera.com.
3. Use ssh to log into the cluster.
   For example:

```
ssh myhost-vpc.cloudera.com
```

4. Type hive to start Hive from the command line.
5. Enter Hive queries.

```
SHOW DATABASES;
```

```
CREATE TABLE students (name VARCHAR(64), age INT, gpa DECIMAL(3,2));
```

```
INSERT INTO TABLE students VALUES ('fred flintstone', 35, 1.28), ('barney
 rubble', 32, 2.32);
```

Starting Hive using a password

You start the Hive shell using a Beeline command to query Hive as an end user authorized by Apache Ranger. As administrator, you set up the end user in the operating system and in Ranger.

About this task

Before starting Hive for the first time, check that you are covered by Ranger policies required for basic operations as shown in the following steps. All users need to use the default database, perform operations such as listing database names, and query the information schema. The preloaded default database tables columns and information_schema database Ranger policies cover group public (all users). If these policies are disabled, you cannot use the default database, perform basic operations such as listing database names, or query the information schema. For example, if the default database tables columns policy is disabled, the following error appears if you try to use the default database:

```
hive> USE default;
Error: Error while compiling statement: FAILED: HiveAccessControlException
Permission denied: user [hive] does not have [USE] privilege on [default]
```

Procedure

1. Access the Ranger Console: click the Ranger Admin web UI link, enter your user name and password, then click Sign In.
2. On the far right, click Ranger Hadoop SQL , and in Allow Conditions, edit all - database, table, column.

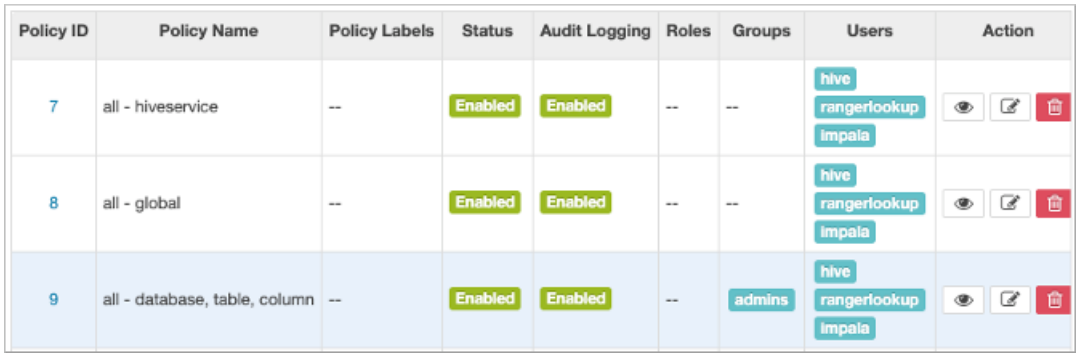

3. Add your user or group name to Hive policies to grant full access to Hive.
   For example, add the admins group name to the list of groups that can access Hive.

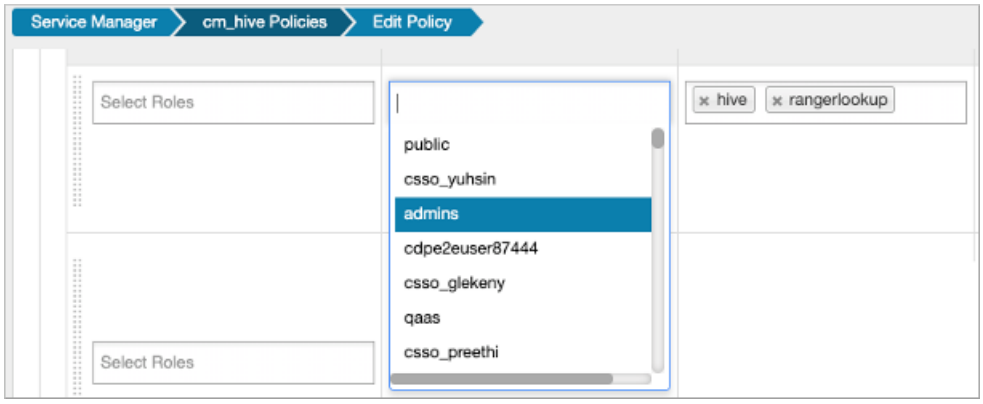

4. Check that the preloaded default database tables columns and information_schema database policies are enabled for group public.

Policy ID	Policy Name	Policy Labels	Status	Audit Logging	Roles	Groups	Users
7	all - global	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
8	all - database, table, column	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
9	all - database, table	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
10	all - database	--	Enabled	Enabled	--	public	hive beacon dpprofiler hue + More..
11	all - hiveservice	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
12	all - database, udf	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
13	all - url	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
14	default database tables columns	--	Enabled	Enabled	--	public	--
15	Information_schema database ...	--	Enabled	Enabled	--	public	--

5. In Environments *YOUR ENVIRONMENT YOUR CLUSTER* , click the CM-URL.
6. In Cloudera Manager, click Hosts All Hosts .

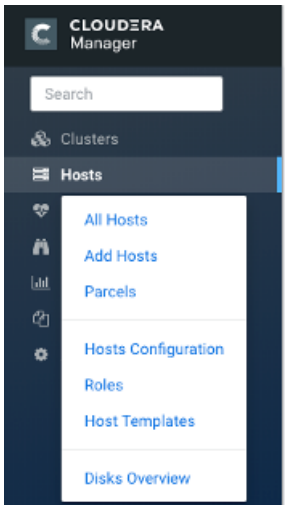

7. Make a note of the IP address or host name of a node in your cluster, for example myhost-vpc.cloudera.com.

Tip: In a Kerberized cluster, ensure the core-site.xml file is correctly configured on all nodes from which you plan to connect to beeline. Include the following property in the core-site.xml:

```
<property>
  <name>hadoop.security.authentication</name>
  <value>kerberos</value>
</property>
```

8. Use ssh to log into the cluster.
   For example:

```
ssh myhost-vpc.cloudera.com
```

   You can get help about starting the Hive shell. On the command line, type

```
hive -h
```

   Output is:

```
Connect using simple authentication to HiveServer2 on localhost:10000
beeline -u jdbc:hive2://localhost:10000 username password

Connect using simple authentication to HiveServer2 on hs.local:10000 us
ing -n for username and -p for password
beeline -n username -p password -u jdbc:hive2://hs2.local:10012

Connect using Kerberos authentication with hive/localhost@mydomain.com as
 HiveServer2 principal
beeline -u "jdbc:hive2://hs2.local:10013/default;principal=hive/localho
st@mydomain.com"

Connect using SSL connection to HiveServer2 on localhost at 10000
beeline "jdbc:hive2://localhost:10000/default;ssl=true;sslTrustStore=/usr/
local/truststore;trustStorePassword=mytruststorepassword"

Connect using LDAP authentication
beeline -u jdbc:hive2://hs2.local:10013/default <ldap-username> <ldap-pas
sword>
```

9. Using the fully qualified domain name or IP address a node in your cluster from Cloudera Manager Hosts Role(s) and looking through the list of roles to find HS2 (Hive on Tez HiveServer2). For example:

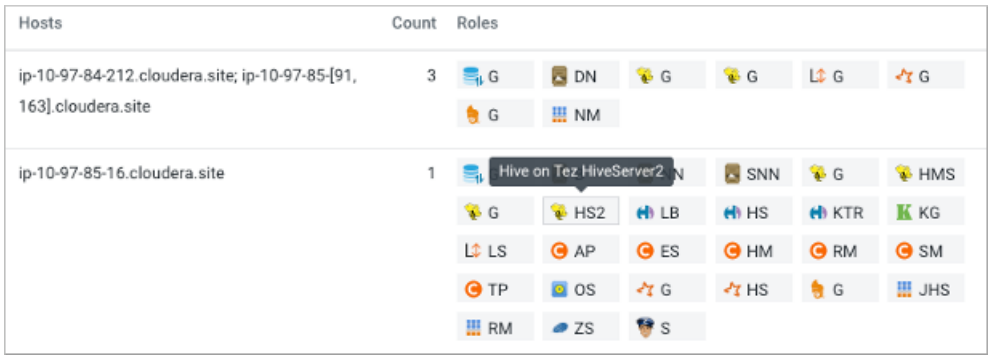

   This node has the HiveServer role, so you can use the name or IP address in Beeline.

10. Start the Hive shell.

- Use your user name if your cluster security is set up.
- Use the user name hive and no password.

Substitute the name or IP address of your HiveServer host for 10.65.13.98.

Simple authentiation:

```
beeline -u jdbc:hive2://10.65.13.98:10000 -n <your user name> -p
```

Kerberos:

```
beeline -u "jdbc:hive2://10.65.13.98:10000/default;principal=hive/_HOST@
CLOUDERA.SITE"
```

11. Enter Hive queries.

```
SHOW DATABASES;
```

```
CREATE TABLE students (name VARCHAR(64), age INT, gpa DECIMAL(3,2));
```

```
INSERT INTO TABLE students VALUES ('fred flintstone', 35, 1.28), ('barney
 rubble', 32, 2.32);
```

Related Information

Configure a Resource-based Policy: Hive

Accessing Hive from an external node

In Cloudera Public Cloud, most of the services are accessible through the Apache Knox gateway including Apache Hive. You need to know the URL of the Cloudera Data Hub cluster to use from an external cluster and the Beeline command to run from an external node.

Before you begin

- In Cloudera Public Cloud, you must be running the Knox, Hive Metastore, and Hive services.
- The external node must be on a cluster that Cloudera Public Cloud can access.
- On an external cluster, you must be authorized to access the Cloudera Data Hub cluster.

Procedure

1. In Cloudera Public Cloud, in Cloudera Managernavigate to Data Hub Clusters Endpoints tab , and copy the Hive Server endpoint.

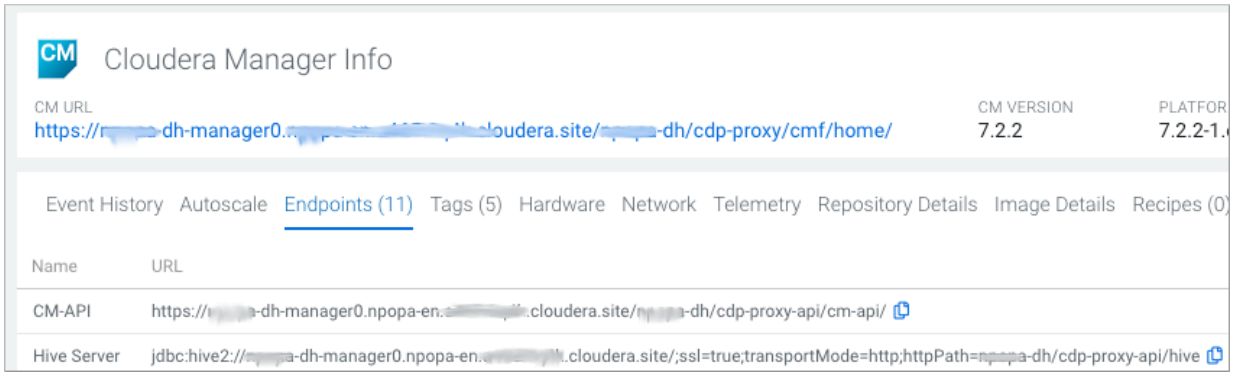

   If you are not running the required services, the URL you need to copy is not listed in Endpoints.

2. On the external node, enter the command to launch Hive.

```
beeline -p -u '<Hive Endpoint URL>;user=<your  user>'
```

   For example:

```
beeline -p -u 'jdbc:hive2://test5-master0.repro.cloudera.site/;ssl=true;
transportMode=http;httpPath=test5/cdp-proxy-api/hive;user=john'
```

   The Hive prompt appears after entering the password, and you can enter Hive queries.

Running a Hive command in Cloudera Data Hub

You can run most Hive commands that push configuration variables to Hive SQL scripts from the command line of a node in your cluster. You use a keyword and options to launch these commands in Beeline.

About this task

Hive supports running Hive commands from the command line. The command you enter launches Beeline in the background. The -e flag followed by a Hive set command lists system variables.

Procedure

On the command line of a node in your Cloudera cluster, enter the hive command to send configuration properties to standard output.

```
> hive -e set
```

Supported commands appear. All obsolete Hive CLI commands are supported by Beeline except set key=value commands that configure Hive Metastore.

The output includes the system variable settings:

```
+-----------------------------------------------------------------+
|                              set                                |
+-----------------------------------------------------------------+
| _hive.hdfs.session.path=/tmp/hive/hive/91ecb...00a              |
| _hive.local.session.path=/tmp/hive/91ecb...00a                  |
                         |
```

```
...
```

Configuring graceful shutdown property for HiveServer

Learn how to configure the graceful shutdown timeout property for HiveServer (HS2), which ensures that HS2 waits for a specified time period before shutting down, thereby allowing queries that are already running to complete before HS2 stops.

About this task

By default, the graceful shutdown timeout property is set to 5 minutes. Queries that are still running beyond the timeout period are aborted and will have to be manually submitted by the user because HS2 cannot automatically resubmit these queries.

Important: If you want to disable HS2 graceful shutdown, set the graceful timeout property value to 0.

You must also know that submitting new queries or starting a new session during a HS2 graceful shutdown will fail. Also, if there are no active queries during the graceful shutdown, HS2 stops in 30 seconds instead of waiting 5 minutes for all the live sessions to close.

You can modify the graceful shutdown timeout property based on a customer's usage, for example, you can choose to increase the timeout property if the customer has long running queries.

The HS2 graceful shutdown is initiated during a Cloudera Runtime Rolling Upgrade, Rolling Restart, can be triggered by stopping HS2 in Cloudera Manager, or by running the hive --service hiveserver2 --graceful_stop command.

Procedure

1. In Cloudera Manager, click Clusters and select the Hive on Tez service.
2. From the Hive on Tez service, go to the **Configuration** tab and search for hive.server2.graceful.stop.timeout.

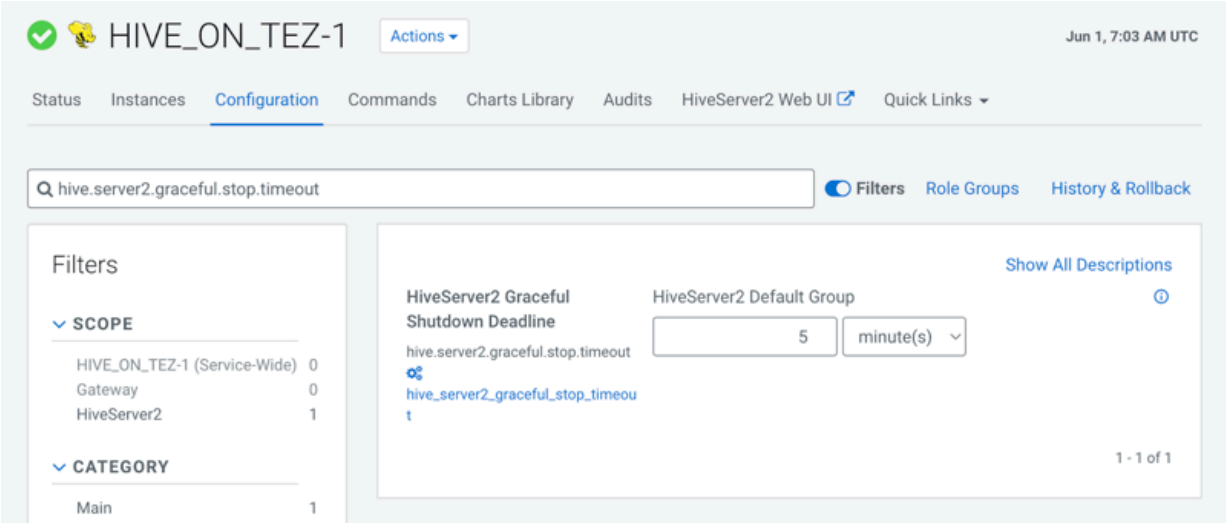

3. Modify the graceful timeout value. By default, the timeout value is set to 5 minutes.
4. Click Save Changes and restart the Hive on Tez service.

Results

The changes do not take effect during the first HS2 restart and the graceful timeout value that was specified prior to the change is considered for the graceful shutdown. The new changes take effect only from the second restart.

For example, if the timeout value was changed from 5 minutes to 10 minutes, the first HS2 restart considers the 5 minute timeout for the graceful shutdown. The second HS2 restart will consider the 10 minute timeout value.

Cloudera Runtime 7.2.11

Troubleshooting HBase

Date published: 2020-02-29
Date modified: 2021-09-09

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Troubleshooting HBase 4

Using the HBCK2 tool to remediate HBase clusters 4

Running the HBCK2 tool 4
Finding issues 5
Fixing issues 6
HBCK2 tool command reference 8

Thrift Server crashes after receiving invalid data 9

HBase is using more disk space than expected 9

Troubleshoot RegionServer grouping 10

Troubleshooting HBase

If you have encountered an error and want to search for speecific keywords in the Apache HBase log, you can use the log search feature in Cloudera Manager. To search for specific keywords in the Apache HBase logs in Cloudera Manager, go to Diagnostics Logs . You can search for specific keywords from various sources, services, role types, and hosts.

Using the HBCK2 tool to remediate HBase clusters

The HBCK2 tool is a repair tool that you can use to remediate Apache HBase clusters. The HBCK2 tool is the next version of the Apache HBase hbck tool.

The HBCK2 tool is a repair tool to remediate Apache HBase clusters in Cloudera. The HBCK2 tool is the next version of the Apache HBase hbck tool.

To identify a list of inconsistencies or blockages in a running HBase cluster, you can view the Master logs. Once you have identified the issue, you can then use the HBCK2 tool to fix the defect or to skip-over a bad state. The HBCK2 tool uses an interactive fix-it process by asking the Master to make the fixes rather than carry out the repair locally.

The HBCK2 performs a single, discrete task each time it is run. The HBCK2 tool does not analyze everything in a running cluster and repair all the problems. Instead, you can use the HBCK2 tool to iteratively find and fix issues in your cluster. The HBCK2 tool lets you use interactive commands to fix one issue at a time.

Important: The HBCK2 tool is specific to internals of Apache HBase. Using this tool requires binaries that are specific to your version of Cloudera Runtime, and you must always use it with the assistance of Cloudera Support and/or Cloudera Professional Services. Please contact Cloudera Support if you believe you have an issue that requires using the HBCK2 tool.

Running the HBCK2 tool

You can run the HBCK2 tool from the command-line interface in your target cluster.

About this task

The HBCK2 tool is a part of the hbase-operator-tools binary. Once you get the hbase-operator-tools binary from Cloudera, upload the binary tarball to the target cluster and extract the tarball. The HBCK2 JAR file is contained in the operator tools tarball provided by Cloudera Support at hbase-operator-tools-<version>/hbase-hbck2/hbase-hbck2-<version>.jar.

Procedure

- Run the HBCK2 tool by specifying the JAR path with the “-j” option as shown here:

  ```
  $ hbase hbck -j $HOME/hbase-operator-tools-<version>/hbase-hbck2/hbase-h
  bck2-<version>.jar
  ```

  When you run the command, the HBCK2 tool command-line menu appears.

  As a Cloudera Support or Professional Services personnel using this tool to remediate an HBase cluster, you can gather useful information using these commands as an HBase super user (typically, hbase), or an HBase principal if Kerberos is enabled:

  ```
  $ hdfs dfs -ls -R /hbase/ 2>&1 | tee /tmp/hdfs-ls.txt
  $ hbase hbck -details 2>&1 | tee /tmp/hbase-hbck.txt
  ```

```
$ echo "scan 'hbase:meta'" | hbase shell 2>&1 | tee /tmp/hbase-meta.txt
```

Finding issues

The HBCK2 tool enables you to use interactive commands to fix one issue at a time. If you have multiple issues, you may have to run the tool iteratively to find and resolve all the issues. Use the following utilities and commands to find the issues.

Find issues using diagnostic tools

Master logs

The Apache HBase Master runs all the cluster start and stop operations, RegionServer assignment, and server crash handling. Everything that the Master does is a procedure on a state machine engine and each procedure has an unique procedure ID (PID). You can trace the lifecycle of a procedure by tracking its PID through the entries in the Master log. Some procedures may spawn sub-procedures and wait for the sub-procedure to complete.

You can trace the sub-procedure by tracking its PID and the parent PID (PPID).

If there is a problem with RegionServer assignment, the Master prints a STUCK log entry similar to the following:

```
2018-09-12 15:29:06,558 WARN
org.apache.hadoop.hbase.master.assignment.AssignmentManager: STUCK
Region-In-Transition rit=OPENING, location=va1001.example.org,00001,10001732
30599,
table=IntegrationTestBigLinkedList_20180626110336,
region=dbdb56242f17610c46ea044f7a42895b
```

Master user interface

Status tables

You can find issues in your HBase tables by looking at the status tables section in the Master user interface home page. Look through the list of tables to identify if a table is ENABLED, ENABLING, DISABLED, or DISABLING. You can also take a look at the regions in transition states: OPEN, CLOSED. For example, there may be an issue if a table is ENABLED, some regions are not in the OPEN state, and the Master log entries do not have any ongoing assignments.

Procedures and locks

When an Apache HBase cluster is started, the Procedures & Locks page in the Master user interface is populated with information about the procedures, locks, and the count of WAL files. After the cluster settles, if the WAL file count does not reduce, it leads to procedure blocks. You can identify those procedures and locks on this page.

You can also get a list of locks and procedures using this command in the HBase shell:

```
$ echo "list_locks"| hbase shell &> /tmp/locks.txt
$ echo "list_procedures"| hbase shell &> /tmp/procedures.txt
```

Apache HBase canary tool

Use the HBase canary tool to verify the state of the assigns in your cluster. You can run this tool to focus on just one table or the entire cluster. You can check the cluster assign using this command:

```
$ hbase canary -f false -t 6000000 &>/tmp/canary.log
```

Use the -f parameter to look for failed region fetches, and set the -t parameter to run for a specified time.

Fixing issues

You can fix issues using the HBCK2 tool.

You must keep these in mind when fixing issues using HBCK2. Ensure that:

- A region is not in the CLOSING state during “assign”, and in the OPENING state during “unassign”. You can change the state using the setRegionState command. See the HBCK2 tool Command Reference section for more information.
- You fix only one table at a time.

Important: Contact Cloudera Support before using any of the HBCK2 tool commands.

Fix assign and unassign issues

You can fix assign and unassign issues by monitoring the current list of outstanding locks. An assign against a locked region will wait till the lock is released. An assignment gets an exclusive lock on the region.

Fix master startup cannot progress error

If you see a master startup cannot progress holding-pattern until region online error in the Master log, it means that the Master is unable to start because there is no procedure to assign hbase:meta. You will see an error message similar to this:

```
2020-04-01 22:07:42,792 WARN org.apache.hadoop.hbase.master.HMaster:
 hbase:meta,,1.1588230740 is NOT online; state={1588230740 state=CLOSING,
ts=1538456302300, server=ve1017.example.org,22101,1234567891012};
ServerCrashProcedures=true. Master startup cannot progress in holding-pat
tern until region onlined.
```

To fix this issue, run the following command:

```
$ hbase hbck -j $HOME/hbase-operator-tools-<version>/hbase-hbck2/hbase-hbck2
-<version>.jar
assigns 1588230740
```

The same issue can occur with a hbase:namespace system table. To fix this issue, run the following command:

```
$ hbase hbck -j $HOME/hbase-operator-tools-<version>/hbase-hbck2/hbase-hbck2
-<version>.jar
assigns <hbase:namespace encoded region id>
```

You can find the namespace encoded region id using this command:

```
$ echo "scan 'hbase:meta',{COLUMNS=>'info:regioninfo',
FILTER=>\"PrefixFilter('hbase:namespace')\"}" | hbase shell
```

The namespace encoded region id is the value under the "ENCODED" field in the results.

Fix missing regions in hbase:meta region/table

If you encounter an issue where table regions have been removed from the hbase:meta table, you can use the addFsRegionsMissingInMeta to resolve this issue. Ensure that the Master is online. This command is not as disruptive as the hbase:meta rebuild command.

To fix this issue, run this command:

```
$ hbase hbck -j $HOME/hbase-operator-tools-<version>/hbase-hbck2/hbase-hbck2
-<version>.jar
addFsRegionsMissingInMeta <NAMESPACE|NAMESPACE:TABLENAME>
```

The command returns an HBCK2 “assigns” command with all the listed re-inserted regions.You must restart the Master, and then run the HBCK2 'assigns' command returned by the addFsRegionsMissingInMeta command to complete your fix.

Example output:

```
Regions re-added into Meta: 2
WARNING:
2 regions were added to META, but these are not yet on Masters cache.
You need to restart Masters, then run hbck2 'assigns' command below:
assigns 7be03127c5e0e2acfc7cae7ddfa9e29e e50b8c1adc38c942e226a8b2976f0c8c
```

Fix extra regions in hbase:meta region/table

If there are extra regions in hbase:meta, it may be because of problems in splitting, deleting/moving the region directory manually, or in rare cases because of the loss of metadata.

To fix this issue, run this command:

```
$ hbase hbck -j $HOME/hbase-operator-tools-<version>/hbase-hbck2/hbase-hbck2
-<version>.jar
extraRegionsInMeta --fix  <NAMESPACE|NAMESPACE:TABLENAME>...
```

Important: Use the --fix option only when the extra regions are overlapping the existing valid regions. Otherwise, use the assigns command to recreate the regions.

Rebuild hbase:meta

If hbase:meta is offline because it is corrupted, you can bring it back online if the corruption is not too critical. If the namespace region is among the mission regions, scan hbase:meta during initialization to check if hbase:meta is online.

To check if hbase:meta is online, run this command in the Apache HBase shell:

```
$ echo "scan 'hbase:meta', {COLUMN=>'info:regioninfo'}" | hbase shell
```

If this scan does not throw any errors, then you can run the following command to validate that the tables are present:

```
$ hbase hbck -j $HOME/hbase-operator-tools-<version>/hbase-hbck2/hbase-hbck2
-<version>.jar
addFsRegionsMissingInMeta <NAMESPACE|NAMESPACE:TABLENAME>
```

The addFsRegionsMissingInMeta command adds regions back to the hbase:meta table if the regioninfo file is present in the storage but the regions were deleted because of an issue.

Fix dropped references and corrupted HFiles

To fix hanging references and corrupt HFiles, run the following command:

```
$ hbase hbck -j $HOME/hbase-operator-tools-<version>/hbase-hbck2/hbase-hbck2
-<version>.jar
filesystem --fix [<TABLENAME>...]
```

HBCK2 tool command reference

You can use this list of HBCK2 commands in your target cluster's command-line interface.

HBCK2 commands

- ```
  addFsRegionsMissingInMeta <NAMESPACE|NAMESPACE:TABLENAME>...
  ```

  Options: -d,--force_disable Use this option to abort fix for table if disable fails.

- ```
  assigns [OPTIONS] <ENCODED_REGIONNAME>...
  ```

  Options:-o,--override Use this option to override ownership by another procedure.

- ```
  bypass [OPTIONS] <PID>...
  ```

  Options: -o,--override Use this option to override if procedure is running/stuck -r,--recursive Use this option to bypass parent and its children.

  -w,--lockWait Use this option to wait (in milliseconds) before giving up; default=1.

- ```
  extraRegionsInMeta <NAMESPACE|NAMESPACE:TABLENAME>...
  ```

  Options:-f, --fix Use this option to fix meta by removing all extra regions found.

- ```
  filesystem [OPTIONS] [<TABLENAME>...]
  ```

  Options:-f, --fix Use this option to sideline corrupt HFiles, bad links, and references.

- ```
  replication [OPTIONS] [<TABLENAME>...]
  ```

  Options:-f, --fix Use this option to fix replication issues.

- ```
  reportMissingRegionsInMeta <NAMESPACE|NAMESPACE:TABLENAME>...
  ```

  Use this command when regions missing from hbase:meta but directories are still present in HDFS.

- ```
  setRegionState <ENCODED_REGIONNAME> <STATE>
  ```

  Possible region states: OFFLINE, OPENING, OPEN, CLOSING, CLOSED, SPLITTING, SPLIT, FAILED_OPEN, FAILED_CLOSE, MERGING, MERGED, SPLITTING_NEW, MERGING_NEW, ABNORMALLY_CLOSED.

  Caution: This command is recommended to be used only as a last resort. Example scenarios include unassigns/assigns that does not happen because the region is in an inconsistent state in hbase:meta.

- ```
  setTableState <TABLENAME> <STATE>
  ```

  Possible table states and representations in hbase:meta table: ENABLED (\x08\x00), DISABLED (\x08\x01), DISABLING (\x08\x02), ENABLING (\x08\x03).

- ```
  scheduleRecoveries <SERVERNAME>...
  ```

  Schedule ServerCrashProcedure(SCP) for list of RegionServers. Format server name as '<HOSTNAME>,<PORT>,<STARTCODE>' .

- ```
  unassigns <ENCODED_REGIONNAME>...
  ```

  Options:-o,--override Use this option to override ownership by another procedure.
  ```

Thrift Server crashes after receiving invalid data

The Thrift server may crash if it receives a large amount of invalid data, due to a buffer overrun.

Why this happens

The Thrift server allocates memory to check the validity of data it receives. If it receives a large amount of invalid data, it may need to allocate more memory than is available. This is due to a limitation in the Thrift library itself.

What to do

To prevent the possibility of crashes due to buffer overruns, use the framed and compact transport protocols. These protocols are disabled by default, because they may require changes to your client code. The two options to add to your hbase-site.xml are hbase.regionserver.thrift.framed and hbase.regionserver.thrift.compact. Set each of these to true, as in the XML below. You can also specify the maximum frame size, using the hbase.regionserver.thrift.framed .max_frame_size_in_mb option.

```
<property>
  <name>hbase.regionserver.thrift.framed</name>
  <value>true</value>
</property>
<property>
  <name>hbase.regionserver.thrift.framed.max_frame_size_in_mb</name>
  <value>2</value>
</property>
<property>
  <name>hbase.regionserver.thrift.compact</name>
  <value>true</value>
</property>
```

HBase is using more disk space than expected

HBase StoreFiles (also called HFiles) store HBase row data on disk. HBase stores other information on disk, such as write-ahead logs (WALs), snapshots, data that would otherwise be deleted but would be needed to restore from a stored snapshot.

Warning: The following information is provided to help you troubleshoot high disk usage only. Do not edit or remove any of this data outside the scope of the HBase APIs or HBase Shell, or your data is very likely to become corrupted.

Table 1: HBase Disk Usage

Location	Purpose	Troubleshooting Notes
/hbase/.snapshots	Contains one subdirectory per snapshot.	To list snapshots, use the HBase Shell command list_snapshots. To remove a snapshot, use delete_snapshot.
/hbase/.archive	Contains data that would otherwise have been deleted (either because it was explicitly deleted or expired due to TTL or version limits on the table) but that is required to restore from an existing snapshot.	To free up space being taken up by excessive archives, delete the snapshots that refer to them. Snapshots never expire so data referred to by them is kept until the snapshot is removed. Do not remove anything from /hbase/.archive manually, or you will corrupt your snapshots.

Location	Purpose	Troubleshooting Notes
/hbase/.logs	Contains HBase WAL files that are required to recover regions in the event of a RegionServer failure.	WALs are removed when their contents are verified to have been written to StoreFiles. Do not remove them manually. If the size of any subdirectory of /hbase/.logs/ is growing, examine the HBase server logs to find the root cause for why WALs are not being processed correctly.
/hbase/logs/.oldWALs	Contains HBase WAL files that have already been written to disk. A HBase maintenance thread removes them periodically based on a TTL.	To tune the length of time a WAL stays in the .oldWALs before it is removed, configure the hbase.master.logcleaner.ttl property, which defaults to 60000 milliseconds, or 1 hour.
/hbase/.logs/.corrupt	Contains corrupted HBase WAL files.	Do not remove corrupt WALs manually. If the size of any subdirectory of /hbase/.logs/ is growing, examine the HBase server logs to find the root cause for why WALs are not being processed correctly.

Troubleshoot RegionServer grouping

When using rsgroup, if you encounter an issue such as the one in the following example, check the log to see what is causing the issue. If a rsgroup operation is unresponsive, you must restart the HBase Master.

An example error of what you will see if you start using rsgroup but the required coprocessor is not found:

```
ERROR: org.apache.hadoop.hbase.exceptions.UnknownProtocolException: No regis
tered master coprocessor service found for name RSGroupAdminService
    at org.apache.hadoop.hbase.master.MasterRpcServices.execMasterService(Ma
sterRpcServices.java:604)
    at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$Mast
erService$2.callBlockingMethod(MasterProtos.java)
    at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:1140)
    at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:133)
    at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:
277)
    at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java
:257)
```

Cloudera Runtime 7.3.1

Hive Troubleshooting

Date published: 2022-12-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Unable to alter S3-backed tables.....4

Unable to alter S3-backed tables

Condition

In Cloudera Public Cloud on AWS or wherever S3-backed tables are used, an ALTER TABLE <table-name> RENAME TO <new-table-name> operation might result in an error stating that the new table location already exists.

Cause

A rename operation on an S3-backed table is not instant. It has to move each file in the table to a new location one at a time. Depending on how many files are in the table directory, this could take more than 5 minutes, which is the default timeout specified.

If the time specified for the connection elapses before the rename completes, Hive tries again and fails because the new directory already exists (from the previous attempt).

Because of this retry behavior, an end-user job or query can result in a misleading error:

```
Caused by: java.sql.SQLException: Error while compiling statement: FAILED: E
xecution Error, return code 1
from org.apache.hadoop.hive.ql.ddl.DDLTask. Unable to alter table. New lo
cation for this table
hive.<dbname>.<tablename> already exists : s3a://<bucket>/warehouse/table
space/managed/hive/<dbname>.db/<tablename>
```

Solution

Procedure

Check the Hive Metastore logs for the time period close to the failed rename and look for the following timeout message:

```
ERROR org.apache.hadoop.hive.metastore.RetryingHMSHandler: [pool-7-thread-31
549]: Error happens in method alter_table_req:
MetaException(message:Timeout when executing method: alter_table_req; 694779
ms exceeds 600000ms)
```

If there is a timeout message, perform the following steps to increase the value of the timeout property for both Hive clients and Hive Metastore:

1. Log in to Cloudera Manager as an administrator.
2. Go to Clusters Hive Configuration and search for the 'Hive Metastore Connection Timeout' property.
3. Modify the value to 1 hour and click Save Changes.
4. Go to Clusters Hive Metastore Configuration and search for 'Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml'.
5. Click ⊕ and add the following property:

   ```
   Name: hive.metastore.client.socket.timeout
   Value: 3600
   ```

6. Click Save Changes.
7. Restart the Hive and Hive Metastore roles.

Cloudera Runtime 7.3.1

Hue Troubleshooting

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

The Hue load balancer not distributing users evenly across various Hue servers.....5

Unable to authenticate users in Hue using SAML.....5

Cleaning up old data to improve performance..... 6

Unable to connect to database with provided credential.....7

Activating Hive query editor on Hue UI.....8

Query execution finished in Hue but shows as executing on Cloudera Manager Impala Queries Page.....9

Finding the list of Hue superusers.....10

'Type' error while accessing Hue from Knox Gateway.....11

Unable to access Hue from Knox Gateway UI.....11

Unable to view Snappy-compressed files..... 14

Impala query fails with invalid query handle error..... 15

Load balancing between Hue and Impala..... 16

Services backed by PostgreSQL fail or stop responding.....19

Invalid method name: 'GetLog' error after submitting Hive queries..... 20

"Authorization Exception" error on submitting queries in Hue.....21

Cannot alter compressed tables in Hue..... 23

MySQL: 1040, 'Too many connections' exception..... 23

Increasing the maximum number of processes for Oracle database..... 24

Fixing authentication issues between HBase and Hue..... 25

Hue Load Balancer does not start due to lengthy BalancerMember Route length..... 26

Enabling access to HBase browser from Hue..... 27

Hue load balancer does not start after enabling TLS..... 27

Unable to log into Hue with Knox..... 28

LDAP search fails with invalid credentials error..... 29

Unable to execute queries due to atomic block..... 29

Query Process fails to start intermittently due to access issues in Java 9 and later..... 30

Disabling the web metric collection for Hue..... 31

Resolving "The user authorized on the connection does not match the session username" error..... 31

Requirements for compressing and extracting files using Hue File Browser..... 32

Fixing a warning related to accessing non-optimized Hue..... 32

Fixing incorrect start time and duration on Hue Job Browser..... 33

The Hue load balancer not distributing users evenly across various Hue servers

The Hue load balancer redirects the new users to the newly added Hue servers and the existing users to the existing Hue servers on your cluster. Even though you add more Hue servers to meet the growing user base, the resources might not be utilized effectively.

About this task

The Hue load balancer is tasked to evenly distribute users across the available Hue servers for effective resource utilization. However, due to session persistence, it does not distribute users evenly. To overcome this issue, you can refresh the cookies from Cloudera Manager.

The load balancer uses the cookie ROUTEID with a random string from your browser and is stored in the hue.conf file. This random string is used to redirect a user to the Hue server. To refresh the cookie and set a new random string every time you add a new Hue server, do the following:

Procedure

1. Go to Cloudera Manager Clusters Hue Configuration .
2. Click Scope Load balancer and select the Hue Load Balancer Cookie Refresh checkbox.

   This refreshes the cookie value in the hue.conf file to rebalance the Hue backend connections.
3. In the Instances tab, select all the Hue services and roles, and click Action for Services Restart .

   This creates a new random string for the cookie which the load balancer can now use to evenly distribute users.

Results

Whenever you restart the Hue server, the load balancer will redistribute users evenly based on the server load.

Unable to authenticate users in Hue using SAML

If you have configured SAML to authenticate users, but your users are unable to log into Hue using Single Sign On (SSO), then it is possible that the RSA key format is not supported. To resolve this issue, you can use an unprotected private key and then specify the private key filename in the safety valve.

Procedure

1. Convert the .key file to an unprotected private key file by using the following command:

```
openssl rsa -in /opt/cloudera/security/<FILE NAME>.key -out /opt/cloudera/
security/<FILE NAME_UNPROTECTED>.key
```

```
openssl rsa -in /opt/cloudera/security/HADOOP-CPI-PROD.key -out /opt/cloud
era/security/HADOOP-CPI-PROD_UNPROTECTED.key
```

2. Update the advanced configuration snippet as shown in the following example:

```
[libsaml]
xmlsec_binary=/usr/bin/xmlsec1
metadata_file=/opt/cloudera/security/saml/idp-openam-metadata.xml
key_file=/opt/cloudera/security/HADOOP-CPI-PROD_UNPROTECTED.key
cert_file=/opt/cloudera/security/HADOOP-CPI-PROD.pem
```

Cleaning up old data to improve performance

Some tables in Hue retain data indefinitely resulting in slower performance or application crashes. Hue does not automatically clean up data from these tables. You can configure Hue to retain the data for a specific number of days and then schedule a cron job to clean up these tables at regular intervals for improved performance.

About this task

Consider cleaning up old data from the backend Hue database if you face the following problems while using Hue:

- Upgrade times out
- Performance is slower than expected
- Long time to log in to Hue
- SQL query shows a large number of documents in tables
- Hue crashes while trying to access saved documents

Important: The clean-up steps only deletes the unsaved documents and workflows. Saved data and information is not cleaned up.

Before you begin

Back up your database before starting the cleanup activity. Check the saved documents such as Queries and Workflows for a few users to prevent data loss. You can also note the sizes of the tables you want to clean up as a reference by running the following queries:

```
select count(*) from desktop_document;
select count(*) from desktop_document2;
select count(*) from beeswax_session;
select count(*) from beeswax_savedquery;
select count(*) from beeswax_queryhistory;
select count(*) from oozie_job;
```

Note: The optimal number of documents that can be stored in a table is less than or equal to 30,000. Consider this number while specifying the cleanup interval.

Procedure

1. SSH into an active Hue instance.
2. Change to the Hue home directory:

```
cd /opt/cloudera/parcels/CDH/lib/hue
```

3. Run the following command as the root user:

```
./build/env/bin/hue desktop_document_cleanup --keep-days X --cm-managed
```

The --keep-days property is used to specify the number of days for which Hue will retain the data in the backend database.

(Optional) Specify DESKTOP_DEBUG=True if you want to log information for troubleshooting purposes.

```
DESKTOP_DEBUG=True ./build/env/bin/hue desktop_document_cleanup --keep-d
ays 30 --cm-managed
```

In this case, Hue will retain data for 30 days.

The logs are displayed on the console because DESKTOP_DEBUG is set to True. Alternatively, you can view the logs from the following location:

/var/log/hue/desktop_document_cleanup.log

The first run can typically take around 1 minute per 1000 entries in each table.

4. Check whether the table size has decreased by running a query as follows:

```
select count(*) from desktop_document;
```

If the desktop_document_cleanup command has run successfully, the table size should decrease.

What to do next

Set up a cron job that runs at regular intervals to automate the database cleanup. For example, you can set up a cron job to run daily and it purges data older than *X* number of days.

Unable to connect to database with provided credential

Cloudera Manager tests the database connection when you add the Hue service to a cluster. The “Test Database Connection” does not work for Oracle databases that require service name instead of the Oracle System ID (SID). This could stop you from adding the Hue service to your cluster.

About this task

If you encounter the following error while adding the Hue service through Cloudera Manager, then follow the workaround as mentioned in this topic:

Unable to connect to database with provided credential. Able to find the Database server, but not the specified dat abase. Please check if the database name is correct and make sure that the user can access the database.

Procedure

1. Install a different database instance, such as MySql temporarily to use with Hue.

   This is referred to as the Hue database.

2. Add the Hue service from Cloudera Manager and specify the Hue database details that you created in the previous step.

   This will allow you to get past the Add Service wizard and add the Hue service to your cluster.

3. Modify the Hue instance to use the actual Oracle database as follows:
   a) Navigate to Cloudera Manager Clusters Hue service Configuration Category Database .

      The database configuration fields are displayed.
   b) Set up the Oracle database by configuring the following fields:
      1. Select Oracle as the Hue Database Type.
      2. In the Hue Database Hostname field, specify the Fully Qualified Domain Name (FQDN) of the host on which you have installed the Oracle database.
      3. In the Hue Database Port field, specify the port on the host on which the Oracle databse is running. Typically, this value is 1521.
      4. In the Hue Database Username field, specify the username to log in to the Oracle database.
      5. In the Hue Database Password field, specify the database password.
      6. In the Hue Database Name field, specify the name of the Hue database in the following format:

         ```
         <HUE_DB_HOST>:1521/<SERVICENAME>
         ```
   c) Navigate to Cloudera Manager Clusters Hue service Configuration Category Advanced and specify the following in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

      ```
      [desktop]
      [[database]]
      port=0
      ```
4. Click Save Changes.
5. Restart the Hue service by clicking Actions Restart .

Activating Hive query editor on Hue UI

You may not see the Hive query editor on the Hue user interface if you have not installed and selected the HIVE_ON_TEZ service on your cluster. The HIVE_ON_TEZ service is needed to configure and use Hive with Hue.

Note:

In CDH 6 and earlier, the Hive service included the Hive Metastore and HiveServer2. In Cloudera Runtime 7.0 and later, this service includes only Hive Metastore. HiveServer2 and other components of the Hive execution engines are part of the HIVE_ON_TEZ service.

About this task

To enable the Hive query editor on the Hue web UI:

Procedure

1. Sign in to Cloudera Manager as an Administrator.
2. Check whether the HIVE_ON_TEZ service is installed on your cluster.

   If it is not installed already, add it as a service.
3. Go to Clusters Hue service Configuration .

   The list of Hue configurations is displyed.
4. Search for the HiveServer2 Service field and select the HIVE_ON_TEZ service.
5. Save the changes and restart Hue.

Results

The Hive query editor should now be available on the Hue user interface.

Query execution finished in Hue but shows as executing on Cloudera Manager Impala Queries Page

Cloudera Manager and the Impala demon web page may show a query in an “executing” or “In Flight” state even though the query has finished executing on the Hue web UI. This can happen due to various reasons.

The three main reasons why the completed Hue query still shows as "executing" are:

- Hue does not close the connection to Impala until you click on the **Results** page.

  Clicking the **Results** page in Hue executes the fetchresults call to Impala.
- Impala queries are client-driven. Therefore, the query still remains in a running state until the client sends a fetch command to complete fetching the entire result set.
- If a query has not been closed or unregistered, Impala shows the same in the **In Flight** section on its web UI. Cloudera Manager shows all In Flight queries in the “Executing” state.

Impala query life cycle

When you submit Impala queries, they are first registered by the system. The system identifies the queries with the help of a coordinator. They also have a state, such as CREATED, INITIALIZED, RUNNING, FINISHED, EXCEPTION, and some metadata.

- FINISHED implies that the rows are available but not all rows are ready to be fetched. It is possible that Impala daemons are still executing the query.
- EXCEPTION implies that an error has occurred. For example, if the system runs out of memory, then the query transitions to the EXCEPTION state.

  The query can also go into an EXCEPTION state if it is cancelled.

Query cancellations may be triggered explicitly with a HiveServer2/Beeswax call or if the query times out. Query time-out may be set through a process-wide impalad argument or with a per-query option.

Currently, Impala does not have a state that explicitly indicates whether all Impala daemons have finished executing the query and that all results have been fetched. Let us call it as End of Statement (EOS), temporarily.

When a query is in the EOS (FINISHED) or EXCEPTION state, the query is not doing any more processing, but the query remains registered. It needs to remain registered because clients may need to access the state.

The query is unregistered only in the following two cases:

- The query is explicitly closed by a Close() API call
- The session associated with the query is closed explicitly or the session time-out is set and the session times out

Note: Hue does not close a query until you explicitly close it. When you close a browser tab on which you are running the query in Hue, the browser sends a JavaScript Close() callback request to close the query. If you leave the query unattended, for example by closing the laptop through which you are accessing Hue or if the browser crashes, then the Close() call is never sent to Hue. The query may eventually time out, but because it was not cancelled explicitly, it does not properly clean up the resources.

To optimize resource utilization, configure the Impala daemon to stop the idle sessions by setting the session timeout value in the --idle_session_timeout impalad argument:

1. Sign in to Cloudera Manager as an Administrator.
2. Go to Clusters Impala service Configuration .

3. Specify the following in the Impala Command Line Argument Advanced Configuration Snippet (Safety Valve) field:

```
--idle_session_timeout=<maximum lifetime of your queries in seconds>
```

For example,

```
--idle_session_timeout=3600
```

In this case, the query will time out after one hour.

Finding the list of Hue superusers

You can fetch the list of superusers by using the Hue shell with Python code or by running a SQL query on the auth _user table.

Using the Hue shell and Python code to find Hue superusers

1. Connecting to Hue shell by running the following command:

```
/opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue shell --cm-managed
```

2. Enter the Python code as follows:

```
from django.contrib.auth.models import User
print "%s" % User.objects.filter(is_superuser = True)
```

Sample output:

```
<QuerySet [<User: admin>]>
```

Running a SQL query on the auth_user table to find Hue superusers

1. Connect to Hue database shell by running the following command:

```
/opt/cloudera/parcels/CDH/lib/hue/build/env/bin/hue dbshell --cm-managed
```

2. Run the following SQL query:

```
select username, is_superuser from auth_user where is_superuser=1;
```

Sample output:

```
----------------------+
username is_superuser
----------------------+

admin 1
----------------------+
1 row in set (0.00 sec)
```

'Type' error while accessing Hue from Knox Gateway

Condition

If you are using KnoxSpenego as an authentication mechanism, and if Knox is set up with Kerberos (HadoopAuth), then you may see the "type" error on the Hue web interface when you try to open Hue from the Knox Gateway UI

Cause

This is because of KNOX-2865.

Solution

Open a Knox support case with Cloudera to request for a hotfix.

Unable to access Hue from Knox Gateway UI

If you are unable to access Hue from the Knox Gateway UI, then it is possible that the KnoxSpnegoDjangoBackend property is not configured for Hue. Cloudera Manager does not set the Authentication Backend to KnoxSpnegoDjangoBackend by default even though Knox is enabled on your cluster.

Procedure

1. Sign in to as an Administrator.
2. Go to Clusters Hue service Configurations and search for the Authentication Backend field.
3. Select desktop.auth.backend.KnoxSpnegoDjangoBackend from the dropdown.

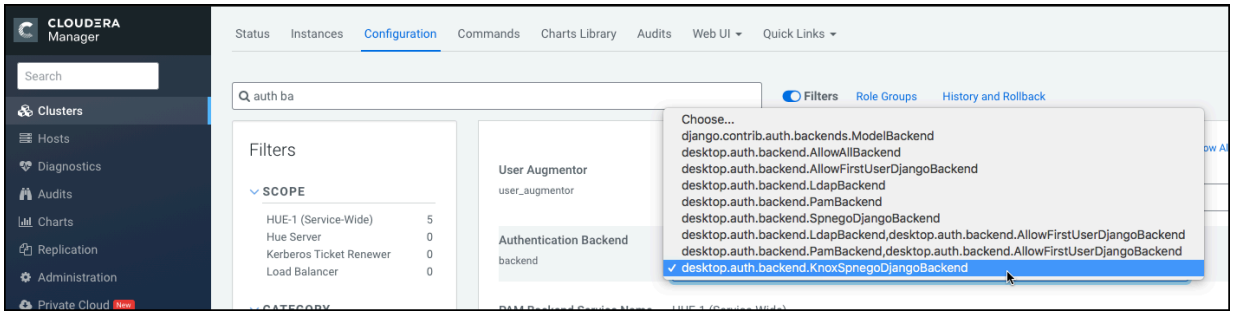

4. Go to Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini) and comment or remove any SAML-specific configurations, if present.

   Caution: Knox-SSO and SAML are incompatible and mutually exclusive. Hue authetication may fail with a redirection loop if you have the Knox-SSO and SAML configurations present in the Hue Advanced Configuration Snippet at the same time, as it confuses the authentication redirect to the IdP and back to Hue.
5. Click Save Changes.
6. Go to Clusters $Knox service Instances and note down the hostnames of the Knox Gateways.

   You must provide these details in the next step.

   If you have set up Knox in High-Availablity (HA) mode, then you can see more than one Knox Gateways listed on the Instances tab.
7. Go back to Clusters Hue service Configurations and search for the Knox Proxy Hosts field.

8. Enter the hostname of the Knox Gateway that you noted earlier.

   If you have set up Knox HA, then click + to add another hostname.

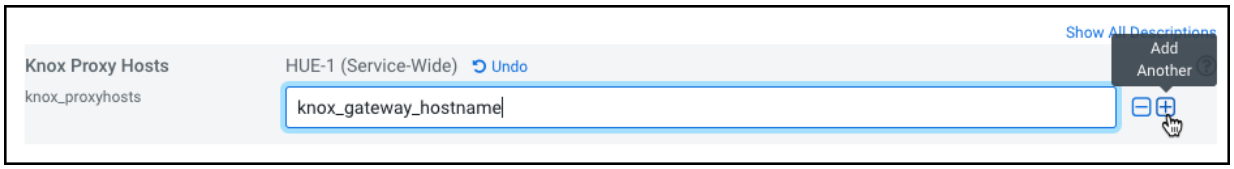

9. If you have deployed a Hue Load Balancer, then you must specify the Load Balancer hostname in the Knox Proxy Hosts field by clicking +.
10. Click Save Changes.

    You would see the following warning:

```
Role is missing Kerberos keytab. Go to the Kerberos Credentials page and
 click the Generate Missing Credentials button.
```

11. Click Administration on the left navigation panel and select Security.
12. Go to the Kerberos Credentials tab and click Generate Missing Credentials.

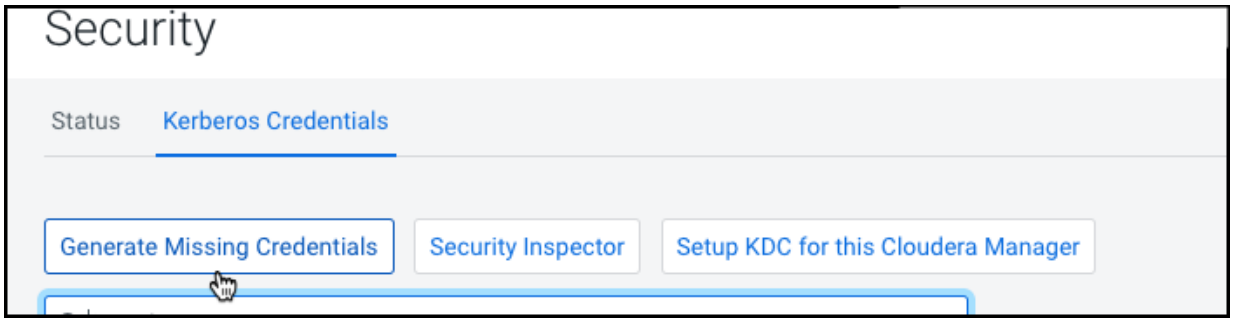

A pop-up showing the status is displayed.

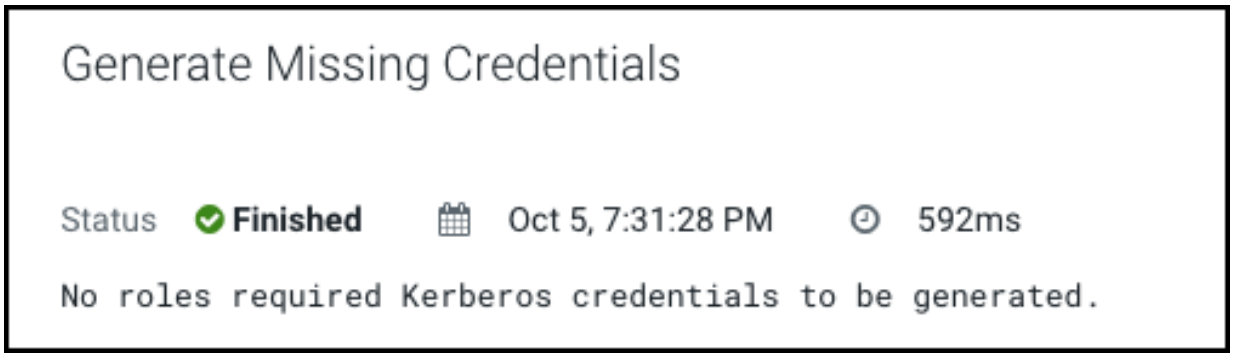

13. Go to Clusters Hue service and click Restart next to Actions.

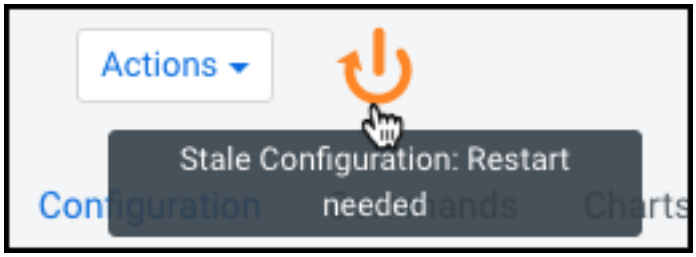

14. On the **Stale Configurations** page, click Restart Stale Services.

    The **Restart Stale Services** wizard is displayed.

15. On the Review Changes page, select Redeploy client configuration, and click Restart Now.

    The **Command Details** page shows the live status as the service restarts.

    When all the steps are complete, click Finish.

16. From the Hue service page, click Web UI Knox Gateway UI .

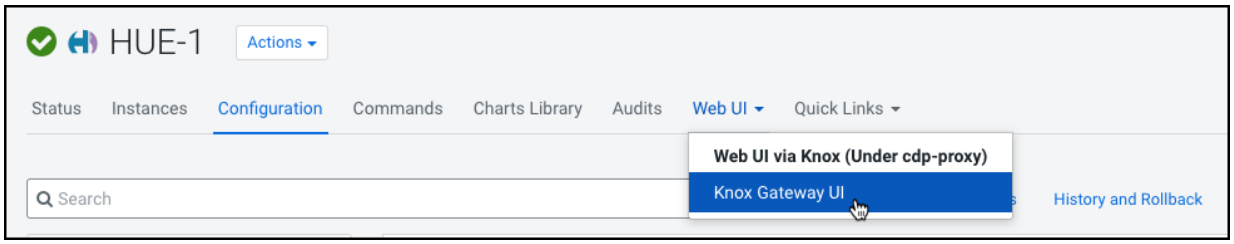

    The Knox Gateway UI is displayed.

17. On the **General Proxy Information** page, expand the CDP Proxy topology by clicking + cdp-proxy under Topologies.

    The list of services that are configured with the cdp-proxy topology is displayed.

18. Click on the Hue logo.

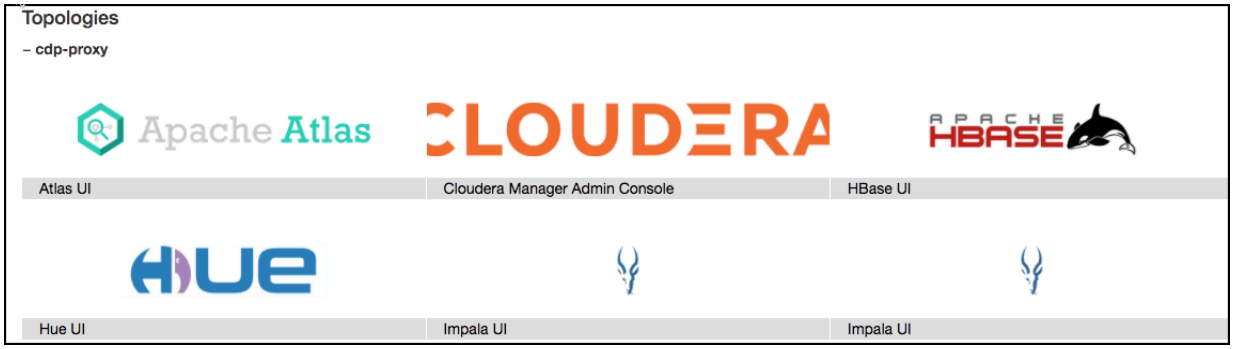

    You should be able to log in to the Hue web UI.

    You can also log into Hue using the following URL:

```
https://[***HOSTNAME***]:[***PORT***]/gateway/cdp-proxy/hue/
```

19. Go to Clusters Knox Configuration and add the following entries in the Knox Simplified Topology Management - cdp-proxy field:

```
HUE:httpclient.socketTimeout=[***TIMEOUT-IN-MINUTES***]
```

```
HUE:httpclient.connectionTimeout=[***TIMEOUT-IN-MINUTES***]
```

    Replace *[***TIMEOUT-IN-MINUTES***]* with the actual timeout value depending on the load on your load on the cluster or environment. For example, to set a timeout of 20 minutes, specify 20m. The default timeout value is 5m.

    Note: This step is required to prevent query timeouts, and if you are seeing errors such as “Results have expired, rerun the query if needed” in the Hue logs.

20. Restart the Knox service.

Unable to view Snappy-compressed files

You must install the python-snappy library on your cluster to view files compressed with Snappy using the Hue File Browser and the HBase Browser. Post-installation, Hue automatically detects and displays the Snappy-compressed files.

Before you begin

The python-snappy library is incompatible with the python library called snappy. You must uninstall snappy if it is present on your cluster.

Run the following command to check whether the snappy library is installed on your cluster:

```
/usr/bin/pip show snappy
```

No output on the console indicates that the snappy library is not installed on your cluster. If you get any results for snappy, then uninstall it by running the following command:

```
/usr/bin/pip uninstall snappy
```

Next, check whether you have the python-snappy library is installed on your cluster by running the following command:

```
/usr/bin/pip show python-snappy
```

Sample output:

```
Name: python-snappy
Version: 0.5.4
Location: /usr/lib64/python2.7/site-packages
```

Procedure

1. Sign in to Cloudera Manager as an Administrator.
2. Stop the Hue service by going to Cluster Hue service Action .
3. Change to the following directory depending on whether you have used parcels or packages to set up your CDH cluster.

   For parcels:

   ```
   cd /opt/cloudera/parcels/CDH/lib/hue
   ```

   For package:

   ```
   cd /usr/lib/hue
   ```

4. Install the python-snappy package by running the following commands:

   ```
   yum install gcc gcc-c++ python-devel snappy-devel
   ./build/env/bin/pip install -U setuptools
   ./build/env/bin/pip install python-snappy
   ```

5. Verify that the python-snappy library is readable by all users by running the following commands:

```
ls -lart `locate snappy.py`
```

The output should be similar to the following:

```
-rw-r--r-- 1 root root 11900 Sep  1 12:25 /usr/lib64/python2.7/site-pack
ages/snappy.py
-rw-r--r-- 1 root root 10344 Sep  1 12:26 /usr/lib64/python2.7/site-packa
ges/snappy.pyc
```

6. Start the Hue service by going to Cluster Hue service Action .
7. Verify that the python-snappy library is working for Hue by running the following command:

```
sudo -u hue /bin/bash -c "echo 'import snappy' | python"
```

If the python-snappy library is working as expected, then no output is displayed for this command.

Results

You should be able to view Snappy-compressed files on the Hue File Browser and the HBase Browser using the Hue web interface.

Impala query fails with invalid query handle error

You encounter an "Invalid query handle" error when running Impala queries from the Hue web interface because the connection between Impala Thrift server and the Hue Load Balancer times out. This is governed by the server_conn_timeout property.

About this task

The default value of the server_conn_timeout property is 30 minutes. You can increase the timeout limit by updating the Hue configuration using Cloudera Manager.

Procedure

1. Log into Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini .
3. Increase the value of the server_conn_timeout property in the Impala section as follows:

```
[impala]
   server_host=[***SERVER-HOST***]
   server_port=[***PORT***]
   server_conn_timeout=[***TIMEOUT-IN-SECONDS***]
```

You can increase the value of the server_conn_timeout property to 2 hours (7200 seconds).

4. Click Save Changes.
5. Restart the Hue service.

Results

You should be able to run Impala queries from the Hue web interface successfully.

Load balancing between Hue and Impala

Condition

You see the following error when running Impala queries from Hue: “"Invalid query handle error or Result have expired , rerun the query if needed"”. You also see either of the following errors in the runcpserver.log file:

- “Invalid query handle”
- “Invalid session id”

Cause

Hue uses a TCP connection pool (10 connections) for all Thrift traffic to Impala. This means that each Impala session is not guaranteed to use the same TCP connection. Load balancers send a single TCP connection to a single Impalad, but without correct persistence, Impala sessions can be sent to the wrong backend server, causing the errors you see.

Solution

Procedure

To solve this issue, you must configure your load balancer that is between Hue and Impala to use Source IP persistence. This is not the load balancer in front of Hue on port 8888/8889, this is the load balancer for Impala, defined in the Impala configuration in Cloudera Manager as Impala Daemons Load Balancer. In addition to Source IP persistence, you must also set the timeout in the load balancer for these connections to a bigger value, otherwise the load balancer can close these connections even though Hue is using them and thinks they are active. Cloudera recommends a minium of 6 hours as the timeout value. 12 hours is ideal.

Cloudera also recommends that you split the VIP configurations into 3 different ports, 21000 for impala-shell users, 21050 for JDBC users and then 21051 for Hue instances. This way you only have to configure the high timeout and Source IP persistence for the Hue port 21051.

Configure the HA Proxy as follows:

1. Open a terminal session and SSH in to the Impala Daemon.
2. Download and install an HAproxy service by running a command based on your operating system. For example:

```
yum install haproxy
```

3. Configure HAProxy for each role as follows:

```
vi /etc/haproxy/haproxy.cfg
```

```
# For impala-shell users on port 21000.
#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
frontend  impala_front
    bind                 *:21000 ssl crt /opt/cloudera/security/x509/certk
eynopw.pem
    mode                 tcp
    option               tcplog
    default_backend      impala-shell

#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend impala-shell
    balance              leastconn
```

```
    mode                 tcp
    server impalad1 impalad-1.example.com:21000 check ssl ca-file /opt/cl
oudera/security/truststore/ca-truststore.pem
    server impalad2 impalad-2.example.com:21000 check ssl ca-file /opt/
cloudera/security/truststore/ca-truststore.pem
    server impalad3 impalad-3.example.com:21000 check ssl ca-file /opt/cl
oudera/security/truststore/ca-truststore.pem


# For JDBC or ODBC version 2.x driver, use port 21050 instead of 21000.
#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
frontend  impala_front
    bind                 *:21050 ssl crt /opt/cloudera/security/x509/ce
rtkeynopw.pem
    mode                 tcp
    option               tcplog
    default_backend      impala-jdbc
#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend impala-jdbc
    balance              leastconn
    mode                 tcp
    server impalad1 impalad-1.example.com:21050 check ssl ca-file /opt/c
loudera/security/truststore/ca-truststore.pem
    server impalad2 impalad-2.example.com:21050 check ssl ca-file /opt/clo
udera/security/truststore/ca-truststore.pem
    server impalad3 impalad-3.example.com:21050 check ssl ca-file /opt/c
loudera/security/truststore/ca-truststore.pem


# Setup for Hue or other JDBC-enabled applications.
# In particular, Hue requires SOURCE IP PERSISTANCE
# The application connects to load_balancer_host:21051, and HAProxy bala
nces
# connections to the associated hosts, where Impala listens for JDBC
# requests on port 21050.
# Notice the timeouts below that do not exist in the other configs
# these are to stop the connections from being killed even though
# hue is using them
#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
frontend  impalajdbc_front
    bind                 *:21051 ssl crt /opt/cloudera/security/x509/cert
keynopw.pem
    mode                 tcp
    option               tcplog
  timeout client    720m
  timeout server    720m
    default_backend      impala-hue

#---------------------------------------------------------------------
# source balancing between the various backends
#---------------------------------------------------------------------
backend impala-hue
    balance              source
    mode                 tcp
    server impalad1 impalad-1.example.com:21050 check ssl ca-file /opt/clo
udera/security/truststore/ca-truststore.pem
    server impalad2 impalad-2.example.com:21050 check ssl ca-file /opt/c
loudera/security/truststore/ca-truststore.pem
```

```
    server impalad3 impalad-3.example.com:21050 check ssl ca-file /opt/clo
udera/security/truststore/ca-truststore.pem
```

4. Create a persistence profile which enables source IP persistence in the F5. Make sure the timeout value is between 6 and 12 hours.

```
ltm persistence source-addr source_addr_12h_idle_timeout {
  app-service none
  defaults-from source_addr
  hash-algorithm default
  map-proxies enabled
  mask none
  match-across-pools disabled
  match-across-services disabled
  match-across-virtuals disabled
  override-connection-limit disabled
  timeout 43200
}
```

5. Create virtual profiles for each VIP, impala-shell, impala JDBC, and impala Hue.

```
impala-shell profile:

ltm virtual vs-impala-21000 {
  description "Impala 21000 for Shell"
  destination <VIPIP>:21000
  ip-protocol tcp
  mask 255.255.255.255
  pool pool-impala-21000
  profiles {
    fastL4 { }
  }
  source 0.0.0.0/0
  source-address-translation {
    type automap
  }
  translate-address enabled
  translate-port enabled
  vs-index 30
}

impala JDBC profile:
ltm virtual vs-impala-jdbc-21050 {
  description "Impala 21050 for JDBC"
  destination <VIPIP>:21050
  ip-protocol tcp
  mask 255.255.255.255
  pool pool-impala-21050
  profiles {
    fastL4 { }
  }
  source 0.0.0.0/0
  source-address-translation {
    type automap
  }
  translate-address enabled
  translate-port enabled
  vs-index 31
}
impala Hue profile:

ltm virtual vs-impala-hue-21051 {
  description "Impala 21051 for Hue"
```

```
  destination <VIPIP>:21051
  ip-protocol tcp
  mask 255.255.255.255
  persist {
    source_addr_12h_idle_timeout {
      default yes
    }
  }
  pool pool-impala-21050
  profiles {
    fastL4 { }
  }
  source 0.0.0.0/0
  source-address-translation {
    type automap
  }
  translate-address enabled
  translate-port enabled
  vs-index 32
}
```

Services backed by PostgreSQL fail or stop responding

When the number of connections between the Cloudera services and the PostgreSQL database exceeds the preset connection limit, it is possible that a new connection fails, Cloudera stops responding, and you cannot log into Hue. The logs show a “FATAL: remaining connection slots are reserved for non-replication superuser connections” error.

About this task

The number of connections between the Cloudera services and the PostgreSQL database is governed by the max_connections setting. By default, the maximum number of available connections to your PostgreSQL database is 115. 15 connections are reserved for the superuser to maintain the state and integrity of your database, and 100 connections are available for Cloudera and other services.

Note: There is a higher probability for this issue to occur in Hue because Cloudera Manager starts the Hue service after starting other Cloudera services. Therefore, the Hue service gets relatively fewer connections to PostgreSQL as compared to other services sharing the same database.

Procedure

1. Check the number of available and idle connections:
   a) SSH into the PostgreSQL database from the command-line client psql as an admin user.
   b) Run the following query to check the number of idle connections:

   ```
   SELECT datname, count(datname) FROM pg_stat_activity WHERE state = 'idle
   ' GROUP BY datname;
   ```

   c) Run the following query to check the number of connections currently in use:

   ```
   SELECT datname, count(datname) FROM pg_stat_activity GROUP BY datname;
   ```

   d) Run the following command to view the maximum number of connections:

   ```
   show max_connections;
   ```

   e) Run the following query to know where the connections are going:

   ```
   SELECT datname, numbackends FROM pg_stat_database;
   ```

2. If most connections are idle and the max_connections value is less than 100, then increase the max_connections value in the postgresql.conf file:
   a) Log into Cloudera Manager and stop all services that use the PostgreSQL database.
   b) SSH into the host on which the PostgreSQL server is running.
   c) Open the postgresql.conf file for editing.

   The postgresql.conf file is typically present in the /var/lib/pgsql/data directory. But this may vary depending on where you have installed the database.
   d) Increase the value of max_connections as per the following recommendation:

   Allow a maximum of 100 connections for each database and add 50 extra connections. For example, for two databases, set the maximum connections to 250.

   If you store five databases on one host (the databases for Cloudera Manager Server, Activity Monitor, Reports Manager, Apache Atlas, and Hive Metastore), set the maximum connections to 550.
   e) Save the changes and exit.
   f) Restart the PostgreSQL database by running the following command:

   ```
   pg_ctl restart
   ```

   g) Restart all the affected services from Cloudera Manager.

What to do next

If increasing the connection limit does not solve your problem and you see a need to scale up, then add new PostgreSQL instances on other hosts and migrate the services to those hosts with the help of your Database Administrator (DBA).

Invalid method name: 'GetLog' error after submitting Hive queries

The Invalid method name: 'GetLog' (code THRIFTAPPLICATION): None error can occur after you submit Hive queries from the Hue editor and while Hue tries to fetch the result set. If you encounter this error, then set the beeswax use_get_log_api property to false using Cloudera Manager.

Procedure

1. Log into Cloudera Manager as an Administrator.

2. Go to Clusters Hue service Configuration .
3. Add/update the beeswax section as follows in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[beeswax]
use_get_log_api=false
```

4. Click Save Changes.
5. Restart the Hue service.

"Authorization Exception" error on submitting queries in Hue

If you have secured your cluster using Ranger, then you must grant the required permissions to your users and groups from the Ranger web UI. If your users do not have proper permissions, then they may not be able to access certain databases or tables from the Hue editor.

About this task

If your users see the "Authorization Exception: User does not have privileges to execute..." error on submitting queries from the Hue editor, then grant them the proper permissions using the Ranger web UI.

Procedure

1. Log into Cloudera Manager as an administrator.
2. Go to Clusters Ranger service Instances tab and note down the hostname corresponding to the “Ranger Usersync” role type.
3. Open the Ranger web UI by clicking Ranger Admin Web UI.
4. SSH into the Ranger Usersync host that you noted in step 2 and add the user or the group as follows:

```
ssh root@example.domain.site useradd [***USERNAME/GROUP-NAME***] passwd
[***PASSWORD***]
```

5. On the Ranger web UI, click Hadoop SQL listed under the HADOOP SQL service.

   The **Hadoop SQL Policies** page is displayed.

6. On the **Hadoop SQL Policies** page, you can grant the new user access to all the databases or to specific databases by adding a new policy.
   - To grant the permission on all databases:
     a. Click the policy ID corresponding to "all - database, table, column".

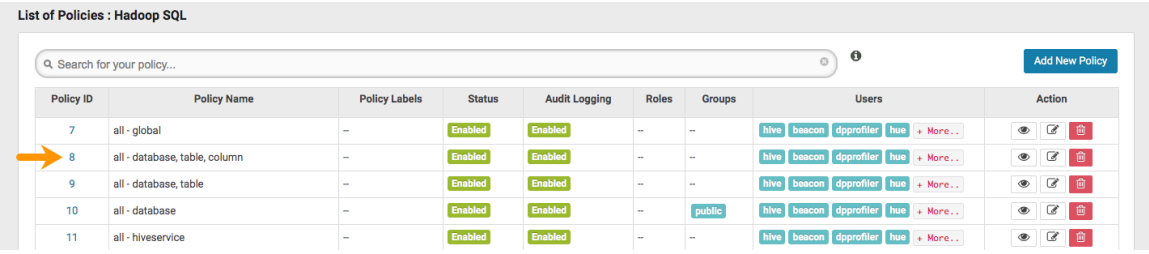

     b. On the **Edit Policy** page, add the user whom you want to grant the permission in the Select User field under the Allow Conditions section as shown in the following image:

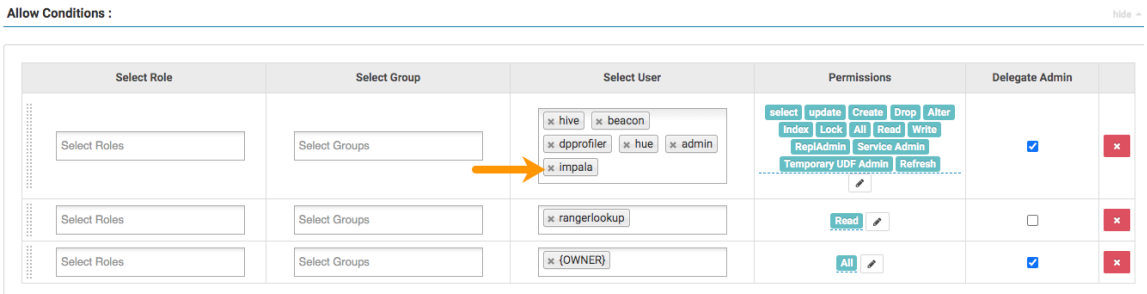

     To grant permissions to a group, enter the group name in the Select Group field.

     c. Click Save.
   - To grant permission on specific database:
     a. Click Add New Policy.

     The **Create Policy** page is displayed.

     b. Under the Policy Details section, specify the policy name and select the database, table, and column that you want your user to access as shown in the following image:

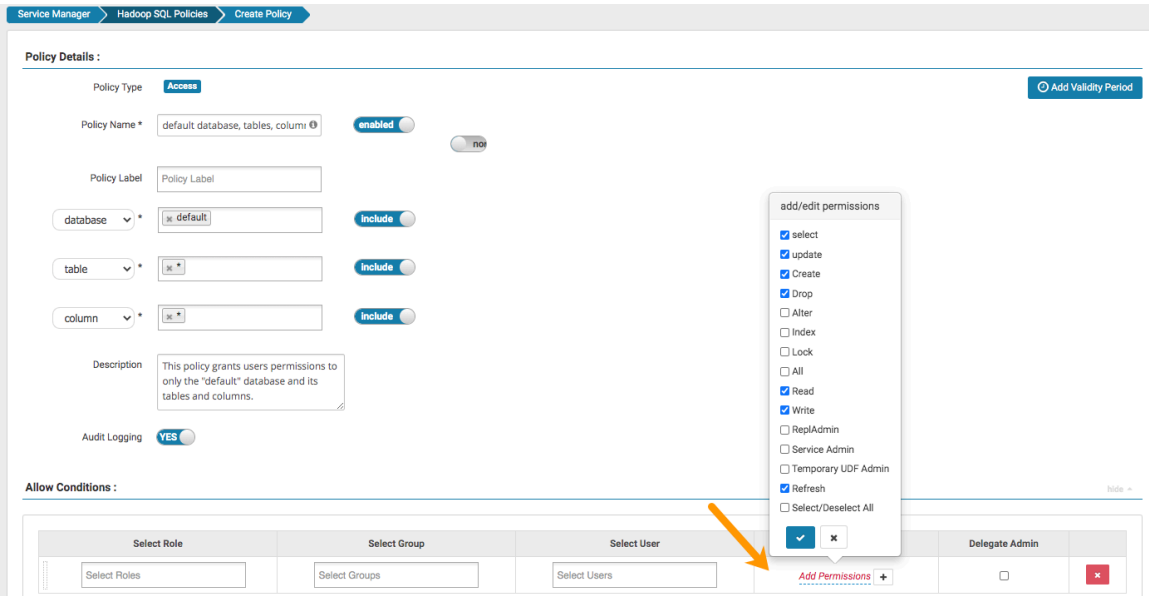

     c. Under the Allow Conditions section, enter the username in the Select User field and click Add Permissions and select the permissions that your user must have.

To grant permissions to a group, enter the group name in the Select Group field.

d. Click Add.

7. Start the Hue service from Cloudera Manager.

Results

The user or the group should be able to run any query on any entities as defined in the policy.

Cannot alter compressed tables in Hue

Due to a known bug in the Oracle database (12c and higher), you cannot perform ALTER TABLE operations (add, delete, drop, modify) on compressed tables. If you have compressed tables in the Hue schema, then you may see the “ORA-39726: unsupported add/drop column operation on compressed tables” error.

About this task

Even if you uncompress an existing table, you may not be allowed to alter the columns. To resolve this issue:

Procedure

1. SSH into the host on which you have installed the Oracle database.
2. Create a new uncompressed table with the same structure as the compressed table.
3. Copy the data from the compressed table to the new uncompressed table.
4. Rename or delete the compressed table.
5. Rename the uncompressed table with the name of the original compressed table.

Results

You should now be able to perform ALTER TABLE operations (add, delete, drop, modify) on the Hue tables.

MySQL: 1040, 'Too many connections' exception

If Hue displays the "1040, Too many connections" exception, then it is possible that the Hue backend database is overloaded and out of maximum available connections. To resolve this issue, you can increase the value of the max_connections property for your database.

About this task

The 1040, 'Too many connections' exception occurs on a MySQL database when it runs out of maximum available connections. If you are using the Impala engine, you may see the following error message on the Hue web interface: OperationalError at /desktop/api2/context/computes/impala("1040: too many connections"). A similar error may be displayed for Hive. The exception is also captured in the Hue server logs.

Before you begin

The max_connections property defines the maximum number of connections that a MySQL instance can accept. Uncontrolled number of connections can crash the server. Following are some guidelines for tuning the value of the max_connections property:

- Set the value of the max_connections property according to the size of your cluster.
- If you have less than 50 hosts, then you can store more than one database (for example, both the Activity Monitor and Service Monitor) on the same host. If you have more than 50 hosts, then use a separate host for each database/host pair. The hosts need not be reserved exclusively for databases, but each database must be on a separate host.

- For less than 50 hosts:
  - Place each database on its own storage volume.
  - Allow 100 maximum connections for each database and then add 50 extra connections. For example, for two databases, set the maximum connections to 250. If you store five databases on one host (the databases for Cloudera Manager Server, Activity Monitor, Reports Manager, Atlas, and Hive MetaStore), then set the maximum connections to 550.

To increase the number of maximium available connections and to resolve the "1040, Too many connections" exception:

Procedure

1. Log in to Cloudera Manager and stop the Hue service.
2. SSH in to your database instance as a root user.
3. Check the number of available connections by running the following command:

   ```
   grep max_conn /etc/my.cnf
   ```

   /etc/my.cnf is the default location of the options file (my.cnf).
4. Set the new value of the max_connections property from the MySQL shell as per the guidelines provided above. For example:

   ```
   mysql> SET GLOBAL max_connections = 550;
   ```
5. Restart the Hue service.

Increasing the maximum number of processes for Oracle database

While using Oracle as a backend database for Hue, if you face issues connecting to the Hue service after restarting the database, then it is possible that the Hue service is not able to get a new database connection. The following error in the Hue logs indicates that the maximum number of connections have exhausted: "ORA-12519: TNS:no appropriate service handler found". This can be resolved by increasing the number of available processes.

After restarting the Oracle database, if you are not able to connect to the Hue service, check the Hue logs for the ORA-12519: TNS:no appropriate service handler found error. If you see the ORA-12519: TNS:no appropriate service handler found error in the logs, then work with your database administrator to check whether the maximum number of processes have exceeded. If the maximum number of processes have exceeded, then you see the following error: ORA-00020: maximum number of processes exceeded. Increase the number of processes to resolve this issue.

How to calculate the number of database processes, transactions, and sessions?

Cloudera recommends that you allow 100 maximum connections for each service that requires a database and then add 50 extra connections. For example, for two services, set the maximum connections to 250. If you have five services that require a database on one host (the databases for Cloudera Manager Server, Activity Monitor, Reports Manager, Cloudera Navigator, and Hive metastore), set the maximum connections to 550.

From the maximum number of connections, you can determine the number of anticipated sessions using the following formula:

```
sessions = (1.1 * maximum_connections) + 5
```

For example, if a host has a database for two services, anticipate 250 maximum connections. If you anticipate a maximum of 250 connections, plan for 280 sessions.

Based on the number of sessions, you can determine the number of anticipated transactions using the following formula:

```
transactions = 1.1 * sessions
```

Continuing with the previous example, if you anticipate 280 sessions, you can plan for 308 transactions.

Work with your Oracle database administrator to apply these derived values to your system.

Using the sample values above, Oracle attributes would be set as follows:

```
alter system set processes=250;
alter system set transactions=308;
alter system set sessions=280;
```

Fixing authentication issues between HBase and Hue

An HBase feature improvement to the Thrift Server (HBASE-19852) may cause authentication issues between HBase and Hue, and you may see the following error while accessing the HBase tables from Hue: “Failed to authenticate to HBase Thrift Server, check authentication configurations.”

About this task

You may also see the following error in the Hue logs: “RestException: Unable to authenticate <Response [401]>”. To fix this issue, update the HBase configurations using Cloudera Manager.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters HBase service Instances and note the hostname of the host on which the HBase Thrift Server is running.

   If multiple Thrift Servers are configured, then find the one that Hue is configured to use.
3. Go to Configuration HBase Thrift Server Advanced Configuration Snippet (Safety Valve) for hbase-site.xml and add the following properties:

Field	Property
Name	hbase.thrift.spnego.principal
Value	HTTP/[***HOSTNAME-FROM-STEP2***]@REALM Substitute @REALM with the actual Kerberos realm.
Name	hbase.thrift.spnego.keytab.file
Value	hbase.keytab

4. Select the following options to enable the properties:
   - Enable HBase Thrift Http Server
   - Enable HBase Thrift Proxy Users
5. Deselect the following properties:
   - Enable HBase Thrift Server Compact Protocol
   - Enable HBase Thrift Server Framed Transport

6. If you have not enabled SSL on your cluster, but if you have Kerberized your cluster, then add the following HBase configurations, without which you may encounter a NullPointerException error while starting the HBase Thrift Server:
   a) Add the following properties in the HBase Client Advanced Configuration Snippet (Safety Valve) for hbase-site.xml field:

Field	Property
Name	hbase.thrift.ssl.enabled
Value	false

   b) Add the following properties in the HBase Thrift Server Advanced Configuration Snippet (Safety Valve) for hbase-site.xml field:

Field	Property
Name	hbase.thrift.ssl.enabled
Value	false

7. Click Save Changes.
8. Go to Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini and add the following lines:

```
[hbase]
thrift_transport=buffered
```

9. Click Save Changes.
10. Restart the HBase and Hue services to apply stale configurations.

Hue Load Balancer does not start due to lengthy BalancerMember Route length

The Hue Load Balancer may not start if the route name in the ROLES table exceeds 64 characters. You must manually reduce the length of the route name for each Hue server in the ROLES table to be less than or equal to 64 characters, and also ensure that it is unique.

About this task

Cloudera Manager creates the Role name in the following format, which may sometimes cause the route name to exceed 64 characters: "Service-name-Role-name-Cluster-name-Unique-hash". You may see the following error while starting the Hue Load Balancer:

```
BalancerMember Route length must be < 64 characters
+ '[' 1 '!=' 0 ']'
+ die '/var/run/cloudera-scm-agent/process/482-hue-HUE_LOAD_BALANCER/http
d.conf is invalid.'
+ echo '/var/run/cloudera-scm-agent/process/482-hue-HUE_LOAD_BALANCER/httpd.
conf is invalid.'
/var/run/cloudera-scm-agent/process/482-hue-HUE_LOAD_BALANCER/httpd.conf is
 invalid.
```

To resolve this issue, modify the route name column in the ROLES table in the scm database for every Hue server to be less than or equal to 64 characters, and ensure that it is unique.

Procedure

1. SSH into the database instance as an administrator.

2. Query the ROLES table to view table content:

```
SELECT * FROM `ROLES`;
```

3. Update the values in the "NAME" column so that the route name is less than or equal to 64 characters:

```
UPDATE ROLES SET NAME='[***HUE-ROLE-NAME***]' WHERE ROLE_ID=[***ROLE-ID-
NUMBER***];
```

```
UPDATE ROLES SET NAME='hue-6c02f47dbd7e181d293c078ea293f3da' WHERE ROLE_
ID=10;
```

Enabling access to HBase browser from Hue

If HBase impersonation is not allowed in HDFS, then you may get an API error while trying to access the HBase browser from Hue. To resolve this issue, you must allow proxy users on the Thrift gateway, and also allow all groups form all hosts in HDFS to impersonate the hbase user.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters HBase service Configuration and search for the hbase.thrift.support.proxyuser property.
3. Enable the Enable HBase Thrift Proxy Users option.
4. Click Save Changes.
5. Go to Clusters HDFS service Configuration .
6. Enter the following lines in the Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml field:

```
<property>
<name>hadoop.proxyuser.hbase.hosts</name>
<value>*</value>
</property>
<property>
<name>hadoop.proxyuser.hbase.groups</name>
<value>*</value>
</property>
```

7. Click Save Changes.
8. Restart the HBase and HDFS services.

Hue load balancer does not start after enabling TLS

If the Hue load balancer does not start after you enable TLS, then check whether your TLS private key file is password protected. To resolve this issue, you must configure the Hue load balancer to use the corresponding key password using Cloudera Manager.

About this task

At startup, the Hue load balancer reads the private key file as defined in the Cloudera Manager Clusters Hue service Configuration Hue Load Balancer TLS/SSL Server Private Key File (PEM Format) configuration property. If the TLS private key file is password protected, then you may see error such as Fatal error initialising mod_ssl, exiting and

Some of your private key files are encrypted for security reasons. In order to read them you have to provide the pass phrases. in the /var/log/hue-httpd/error_log Hue load balancer log file.

Procedure

1. SSH into the Hue host as an Administrator.
2. Create a password file in your chosen security directory and insert the private key password, as shown in the following example:

```
# echo "abc123" > /etc/security/password.txt
# chown hue:hue password.txt
# chmod 700 password.txt
```

3. Log in to Cloudera Manager as an Administrator.
4. Go to Clusters Hue service Configuration and enter the file path and the filename of the password file in the Hue Load Balancer TLS/SSL Server SSLPassPhraseDialog field.
5. Click Save Changes.
6. Restart the Hue service.

Unable to log into Hue with Knox

Opening Hue from Cloudera Manager or Knox Gateway Home may fail if the Kerberos principal name for the Knox service is different from the default Kerberos principal defined in the hue.ini file. To resolve this issue, you must set the value of the knox_principal property to your custom Kerberos principal name for Knox in the Hue Advanced Configuration Snippet, so that Hue's login mechanism (KnoxSpenegoDjangoBackend) can authenticate using Knox's Kerberos principal name.

About this task

If logging into Hue fails due to a mismatch between the custom Kerberos principal name for Knox and the default Kerberos principal name defined in the hue.ini file, then you may see the following error in the access.log file: Failed to verify provided username set(['*KNOX-PRINCIPAL-NAME*']) with set(['knox']).

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Note the Kerberos principal name for the Knox service from Clusters Knox service Configuration Kerberos Principal .
3. Go to Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini and enter the following lines:

```
[desktop]
[[knox]]
knox_principal=[***KNOX-PRINCIPAL-NAME***]
```

   Replace [***KNOX-PRINCIPAL-NAME***] with the Kerberos principal name for the Knox service that you obtained earlier.
4. Click Save Changes.
5. Restart the Hue service.

Results

You should be able to log into Hue from the Knox Gateway UI cdp-proxy topology.

LDAP search fails with invalid credentials error

LDAP authentication fails with an "Invalid credentials" error, even if you input valid login credentials on the Hue login page, and you are unable to log into Hue. To resolve this issue, verify and update the LDAP Bind User credentials using Cloudera Manager.

About this task

This issue may happen if the credentials for the LDAP Bind User for Hue configured in Cloudera Manager are invalid. The invalid credentials could either be the “LDAP Bind Password" or "LDAP Bind User Distinguished Name". If the credentials are valid and the issue persists, verify that LDAP Search Base option in Cloudera Manager Hue Configurations is valid. The LDAP search base should be similar to 'dc=hadoop,dc=mycompany,dc=com'.

Before you begin

This task assumes that the Use Search Bind Authentication option is enabled in Cloudera Manager Hue Configurations . Search Bind Authentication connects to the LDAP server using the credentials provided in the 'bind_dn' and 'bind_password' configurations. If these configurations are not set, then an anonymous search is performed.

If the Use Search Bind Authentication option is not enabled in Cloudera Manager Hue Configurations , then do not set the LDAP Bind User credentials as described in this task. You must use the LDAP Username Pattern field for configuring the LDAP credentials, and verify whether the authentication works as expected.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue Configurations .
3. Set the LDAP Bind User credentials in the following fields:
   - LDAP Bind User Distinguished Name
   - LDAP Bind Password

   You can specify the LDAP Bind User Distinguished Nameeither in the generic LDAPv3 Distinguished Name ("CN=binduser,OU=users,DC=Example,dc=com") format or the Active Directory style (binduser@EXAMPLE.COM) format.
4. Click Save Changes.
5. Restart the Hue service.

Unable to execute queries due to atomic block

Condition

You may see the following error after submitting a query from Hue, while logging into Hue, or while saving documents and workflows: "TransactionManagementError: An error occurred in the current transaction. You can't execute queries until the end of the 'atomic' block."

Cause

If there is a load on Hue's backend database and slower processing of operations such as saving documents or workflows, running INSERT or UPDATE queries from multiple users, then the database puts an atomic block on the table rows. If you are using MySQL or MariaDB as the backend database for Hue, then you can increase the value of the innodb-lock-wait-timeout parameter along with a few other timeout parameters in the my.cnf file.

Solution

Procedure

1. SSH into the database host as an Administrator.
2. Back up the my.cnf configuration file as follows:

```
cp /etc/my.cnf  /[***BACKUP-DIRECTORY***]
```

3. Open the my.cnf file for editing and add the following two lines under the [mysqld] section:

```
vi /etc/my.cnf
```

```
[mysqld]
wait_timeout = 28800
interactive_timeout = 28800
```

4. Save the file and exit from the editor.
5. Restart the database server.

Query Process fails to start intermittently due to access issues in Java 9 and later

Condition

When running a Hue service on Java 9 or later, you may encounter IllegalAccessError exceptions related to module access restrictions. These errors occur because certain internal Java classes (e.g., sun.net.util.IPAddressUtil) are not accessible by default due to Java's module system.

Example Error Messages:

```
[Guice/ErrorInjectingConstructor]: IllegalAccessError: class SecurityUtil$Qu
alifiedHostResolver (in unnamed module @0x31c88ec8)
cannot access class IPAddressUtil (in module java.base)
because module java.base does not export sun.net.util to unnamed module @0
x31c88ec8
```

Cause

Starting with Java 9, the Java Platform Module System restricts access to internal Java classes. If your application relies on these internal classes, the JVM throws an IllegalAccessError unless the module explicitly exports the necessary packages.

Solution

Procedure

1. Log in to the Cloudera Management Console as an Administrator.
2. Go to your environment and click on CM-UI to open Cloudera Manager.
3. Go to Clusters Hue service Configuration query_processor_java_opts option.
4. Add JVM arguments to allow access.

```
--add-exports java.base/sun.net.dns=ALL-UNNAMED --add-exports java.base/
sun.net.util=ALL-UNNAMED
```

5. Click Save Changes.
6. Restart the Hue service.

Disabling the web metric collection for Hue

Condition

You see the following error in the Hue server logs: DatabaseError: ORA-01000: maximum open cursors exceeded, when you are using Oracle as the backend database for Hue.

Cause

Cloudera Manager Agent collects web metrics from the Hue server to monitor its health. At times, the number of metrics that are collected can exhaust the number of available cursors on the Oracle database, resulting in the ORA-01000: maximum open cursors exceeded error.

Solution

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue Configuration and deselect the Web Metric Collection option.

> **Note:** Restarting the Hue service from time to time can also clear the cursors, but temporarily. For a long-term resolution, consider turning off the Web Metric Collection option or reducing the frequency of the metric collection using the Web Metric Collection Duration option in Cloudera Manager.

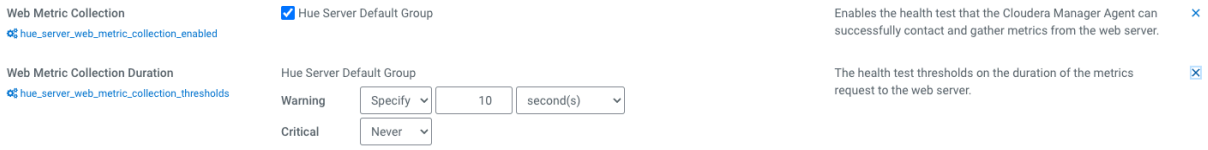

3. Click Save Changes.
4. Restart the Hue service.

Resolving "The user authorized on the connection does not match the session username" error

Condition

In Cloudera, if you try to open two Hue sessions on different ports on the same browser window, then you may see the “The user authorized on the connection does not match the session username” error on the Hue web interface and in the Hue logs.

Cause

On CDH, if you opened Hue web URL on port 8889 on one tab and on port 8888 on another tab, you would be logged out of Hue on the port 8889, and you would be forced to refresh the page. In Cloudera, Hue depends on the ZooKeeper quorum for how it connects to Impala or Hive and how the Hue load balancer distributes the connections to the Hue server nodes and controls switching between the different sesions.

Solution

After migrating/upgrading from CDH to Cloudera, note this change in behaviour and always open Hue web UI using the Hue Load Balanced - recommended option from Cloudera Manager.

Requirements for compressing and extracting files using Hue File Browser

Downloading multiple files from the Hue File Browser is not supported directly. If you want to download multiple files, you can compress them into a zip file and download the zip file. To use the compress and extract function, you must have the Oozie service installed on your cluster, along with other zip and unzip utilities.

Condition

You do not see the Compress option on the **File Browser** page in Hue.

Cause

This can happen because you may not have the Oozie service installed on your cluster, or it is not configured to be used with Hue.

Solution

You must have the Oozie service installed on your cluster. After installing Oozie, go to Cloudera Manager Clusters Hue service Configuration , and select the Oozie Service option. Click Save Changes and restart the Hue service.

You must also install the following utilities for compressing and extracting files on your cluster:

- zip
- unzip
- tar
- bzip2

Note: The option to compress files is not available in Cloudera Data Warehouse.

Fixing a warning related to accessing non-optimized Hue

Condition

You see the following messgae when you open Hue: “You are accessing a non-optimized Hue, please switch to one of the available addresses”.

Cause

This message is displayed when you have a Load Balancer role, but none of the Hue instances are linked to it.

Solution

Procedure

1. Log in to Cloudera Manager as an Administrator.

2. Go to Clusters Hue Configuration .

   If you are using an external load balancer, then you must add the list of the load balancers under the [desktop] section in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field as follows:

   ```
   [desktop]
   hue_load_balancer=[***LOAD-BALANCER-HOST-1***],[***LOAD-BALANCER-HOST-
   2***]
   ```

   Note: If you have added more than one Hue load balancers (Httpd), then restart the Hue service. The hue_load_balancer configuration is automatically updated after restarting Hue.

3. Click Save Changes.
4. Restart the Hue service.

Fixing incorrect start time and duration on Hue Job Browser

Some users may notice incorrect values in the "Started" and "Duration" columns on the Job Browser page in Hue. This could be because the user is unauthorized to view the application according to the Job ACLs. Learn more about this issue and how to fix it.

Condition

Some users see that the "Started" and "Duration" columns show wrong or unexpected values. For example, the Started column displays a value such as: "1 January 1970 00:00" and Duration column displays a value such as: "0s". However, all users do not face this issue. For example, the admin user can see the values fine.

Cause

This issue occurs when Hue accesses the Resource Manager (RM) when Job ACLs are enabled. To confirm this, run the following command and check the output:

```
curl -kiv --negotiate -u : http://[***RM-HTTP-ADDRESS***]:8088/ws/v1/clus
ter/apps/[***APP-ID***]
```

Sample output:

```
{"app":{"id":"appid","user":"testuser","name":"PySparkShell","queue":"root.u
sers.testuser",
"state":"FINISHED","finalStatus":"SUCCEEDED","progress":100.0,
"trackingUI":"History","trackingUrl":"http://[***RM-HTTP-ADDRESS***]:8088/pr
oxy/[***APP-ID***]/",
"diagnostics":"","clusterId":12345678,"applicationType":"SPARK","application
Tags":"",
"startedTime":0,"finishedTime":0,"elapsedTime":0,"allocatedMB":0,"allocate
dVCores":0,
"runningContainers":0,"memorySeconds":175016,"vcoreSeconds":170,
"preemptedResourceMB":0,"preemptedResourceVCores":0,
"numNonAMContainerPreempted":0,"numAMContainerPreempted":0}}
```

"startedTime"=0 in the output indicates that the user is not authorized to view the application according to the following two Job ACLs:

- application VIEW acl (i.e. mapreduce.job.acl-view-job in combination with mapreduce.cluster.acls.enabled)
- queue ADMIN acl (aclAdministerApps for the FS)

Solution

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Go to Clusters YARN Configuration and add the following lines in the YARN Service MapReduce Advanced Configuration Snippet (Safety Valve) and MapReduce Client Advanced Configuration Snippet (Safety Valve) for mapred-site.xml field:

```
<property>
<name>mapreduce.cluster.acls.enabled</name>
<value>true</value>
</property>
<property>
<name>mapreduce.job.acl-view-job</name>
<value>mapred,hue group1,group2</value>
</property>
```

3. Restart the Job History Service and run Deploy Client Configuration.
4. Restart all affected services.

Troubleshooting Apache Kudu

Date published:
Date modified:

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Issues starting or restarting the master or the tablet server....4
Errors during hole punching test....4
Already present: FS layout already exists....4
Troubleshooting NTP stability problems....5

Disk space usage issue....5

Performance issues....6
Kudu tracing....6
Accessing the tracing web interface....6
RPC timeout traces....6
Kernel stack watchdog traces....7
Memory limits....8
Block cache size....8
Heap sampling....9
Slow name resolution and nscd....10

Usability issues....10
ClassNotFoundException: com.cloudera.kudu.hive.KuduStorageHandler....10
Runtime error: Could not create thread: Resource temporarily unavailable (error 11)....10
Tombstoned or STOPPED tablet replicas....10
Corruption: checksum error on CFile block....11

Symbolizing stack traces....11

Recover from a dead Kudu master in a multi-master deployment....12
Prepare for the recovery....13
Perform the recovery....13

Issues starting or restarting the master or the tablet server

You may face issues while starting or restarting the master or the tablet server in case there are errors in the hole punching tests, or if the FS layout already exists, or if the master and tablet server's clocks are not synchronized using NTP.

Errors during hole punching test

Kudu requires hole punching capabilities in order to be efficient. Hole punching support depends upon your operation system kernel version and local filesystem implementation.

- RHEL or CentOS 6.4 or later, patched to kernel version of 2.6.32-358 or later. Unpatched RHEL or CentOS 6.4 does not include a kernel with support for hole punching.
- Ubuntu 14.04 includes version 3.13 of the Linux kernel, which supports hole punching.
- Newer versions of the ext4 and xfs filesystems support hole punching. Older versions that do not support hole punching will cause Kudu to emit an error message such as the following and to fail to start:

```
Error during hole punch test. The log block manager requires a
filesystem with hole punching support such as ext4 or xfs. On el6,
kernel version 2.6.32-358 or newer is required. To run without hole
punching (at the cost of some efficiency and scalability), reconfigure
Kudu with --block_manager=file. Refer to the Kudu documentation for more
details. Raw error message follows.
```

Note:

ext4 mountpoints may actually be backed by ext2 or ext3 formatted devices, which do not support hole punching. The hole punching test will fail when run on such filesystems. There are several different ways to determine whether an ext4 mountpoint is backed by an ext2, ext3, or ext4 formatted device. See the Stack Exchange post for more details.

Without hole punching support, the log block manager is unsafe to use. It won't ever delete blocks, and will consume ever more space on disk.

If you can't use hole punching in your environment, you can still try Kudu. Enable the file block manager instead of the log block manager by adding the --block_manager=file flag to the commands you use to start the master and tablet servers. The file block manager does not scale as well as the log block manager.

Attention:

The file block manager is known to scale and perform poorly, and should only be used for small-scale evaluation and development, and only on systems where hole punching is unavailable.

The file block manager uses one file per block. As multiple blocks are written for each rowset, the number of blocks can be very high, especially for actively written tablets. This can cause performance issues compared to the log block manager even with a small amount of data. And it is impossible to switch between block managers without wiping and reinitializing the tablet servers.

Already present: FS layout already exists

When Kudu starts, it checks each configured data directory, expecting either for all to be initialized or for all to be empty. If a server fails to start with a log message such as the following, then this precondition check has failed.

```
Check failed: _s.ok() Bad status: Already present: Could not create new FS l
ayout: FSManager root is not empty: /data0/kudu/data
```

This could be because Kudu was configured with non-empty data directories on first startup, or because a previously-running, healthy Kudu process was restarted and at least one data directory was deleted or is somehow corrupted, perhaps because of a disk error. If it is the latter, see *Changing directory configuration*.

Troubleshooting NTP stability problems

As of Kudu 1.6.0, Kudu daemons can continue to operate during a brief loss of clock synchronization. If clock synchronization is lost for several hours, the Kudu daemons may crash. If a daemon crashes due to clock synchronization issues, consult the ERROR log for a dump of related information which may help to diagnose the issue.

Note: If using other than link-local NTP server, it may take some time for ntpd to synchronize with one of its reference servers in case of network connectivity issues. In case of a spotty network between the machine and the reference NTP servers, ntpd may become unsynchronized with its reference NTP servers. If that happens, consider finding other set of reference NTP servers: the best bet is to use NTP servers in the local network or *.pool.ntp.org servers.

Disk space usage issue

When using the log block manager (the default on Linux), Kudu uses sparse files to store data. A sparse file has a different apparent size than the actual amount of disk space it uses. This means that some tools may inaccurately report the disk space used by Kudu. For example, the size listed by ls -l does not accurately reflect the disk space used by Kudu data files:

```
$ ls -lh /data/kudu/tserver/data
total 117M
-rw------- 1 kudu kudu 160M Mar 26 19:37 0b9807b8b17d48a6a7d5b16bf4ac4e6d.da
ta
-rw------- 1 kudu kudu 4.4K Mar 26 19:37 0b9807b8b17d48a6a7d5b16bf4ac4e6d
.metadata
-rw------- 1 kudu kudu  32M Mar 26 19:37 2f26eeacc7e04b65a009e2c9a2a8bd20.
data
-rw------- 1 kudu kudu 4.3K Mar 26 19:37 2f26eeacc7e04b65a009e2c9a2a8bd20.m
etadata
-rw------- 1 kudu kudu 672M Mar 26 19:37 30a2dd2cd3554d8a9613f588a8d136ff.da
ta
-rw------- 1 kudu kudu 4.4K Mar 26 19:37 30a2dd2cd3554d8a9613f588a8d136ff
.metadata
-rw------- 1 kudu kudu  32M Mar 26 19:37 7434c83c5ec74ae6af5974e4909cbf82.
data
-rw------- 1 kudu kudu 4.3K Mar 26 19:37 7434c83c5ec74ae6af5974e4909cbf82.m
etadata
-rw------- 1 kudu kudu 672M Mar 26 19:37 772d070347a04f9f8ad2ad3241440090.da
ta
-rw------- 1 kudu kudu 4.4K Mar 26 19:37 772d070347a04f9f8ad2ad3241440090
.metadata
-rw------- 1 kudu kudu 160M Mar 26 19:37 86e50a95531f46b6a79e671e6f5f4151.
data
-rw------- 1 kudu kudu 4.4K Mar 26 19:37 86e50a95531f46b6a79e671e6f5f4151.m
etadata
-rw------- 1 kudu kudu  687 Mar 26 19:26 block_manager_instance
```

Notice that the total size reported is 117MiB, while the first file's size is listed as 160MiB. Adding the -s option to ls will cause ls to output the file's disk space usage.

The du and df utilities report the actual disk space usage by default.

```
$ du -h /data/kudu/tserver/data118M   /data/kudu/tserver/data
```

The apparent size can be shown with the --apparent-size flag to du.

```
$ du -h --apparent-size /data/kudu/tserver/data1.7G  /data/kudu/tserver/data
```

Performance issues

This topic helps you to troubleshoot issues and improve performance using Kudu tracing, memory limits, block size cache, heap sampling, and name service cache daemon (nscd).

Kudu tracing

The Kudu master and tablet server daemons include built-in support for tracing based on the open source Chromium Tracing framework. You can use tracing to diagnose latency issues or other problems on Kudu servers.

Accessing the tracing web interface

The tracing interface is part of the embedded web server in each of the Kudu daemons, and can be accessed using a web browser. Note that while the interface has been known to work in recent versions of Google Chrome, other browsers may not work as expected.

Daemon	URL
Tablet Server	<*TABLET-SERVER-1.EXAMPLE.COM*>:8050/tracing.html
Master	<*MASTER-1.EXAMPLE.COM*>:8051/tracing.html

RPC timeout traces

If client applications are experiencing RPC timeouts, the Kudu tablet server WARNING level logs should contain a log entry which includes an RPC-level trace.

For example:

```
W0922 00:56:52.313848 10858 inbound_call.cc:193] Call kudu.consensus.Consens
usService.UpdateConsensus
from 192.168.1.102:43499 (request call id 3555909) took 1464ms (client tim
eout 1000).
W0922 00:56:52.314888 10858 inbound_call.cc:197] Trace:
0922 00:56:50.849505 (+     0us) service_pool.cc:97] Inserting onto call que
ue
0922 00:56:50.849527 (+    22us) service_pool.cc:158] Handling call
0922 00:56:50.849574 (+    47us) raft_consensus.cc:1008] Updating replica
 for 2 ops
0922 00:56:50.849628 (+    54us) raft_consensus.cc:1050] Early marking co
mmitted up to term: 8 index: 880241
0922 00:56:50.849968 (+   340us) raft_consensus.cc:1056] Triggering prepare
for 2 ops
0922 00:56:50.850119 (+   151us) log.cc:420] Serialized 1555 byte log entry
0922 00:56:50.850213 (+    94us) raft_consensus.cc:1131] Marking committed
 up to term: 8 index: 880241
```

```
0922 00:56:50.850218 (+     5us) raft_consensus.cc:1148] Updating last rece
ived op as term: 8 index: 880243
0922 00:56:50.850219 (+     1us) raft_consensus.cc:1195] Filling consensus
response to leader.
0922 00:56:50.850221 (+     2us) raft_consensus.cc:1169] Waiting on the repl
icates to finish logging
0922 00:56:52.313763 (+1463542us) raft_consensus.cc:1182] finished
0922 00:56:52.313764 (+     1us) raft_consensus.cc:1190] UpdateReplicas() fi
nished
0922 00:56:52.313788 (+    24us) inbound_call.cc:114] Queueing success re
sponse
```

These traces can indicate which part of the request was slow. Make sure you include them when filing bug reports related to RPC latency outliers.

Kernel stack watchdog traces

Each Kudu server process has a background thread called the Stack Watchdog, which monitors other threads in the server in case they are blocked for longer-than-expected periods of time. These traces can indicate operating system issues or bottle-necked storage.

When the watchdog thread identifies a case of thread blockage, it logs an entry in the WARNING log as follows:

```
W0921 23:51:54.306350 10912 kernel_stack_watchdog.cc:111] Thread 10937 stuck
 at /data/kudu/consensus/log.cc:505 for 537ms:
Kernel stack:
[<ffffffffa00b209d>] do_get_write_access+0x29d/0x520 [jbd2]
[<ffffffffa00b2471>] jbd2_journal_get_write_access+0x31/0x50 [jbd2]
[<ffffffffa00fe6d8>] __ext4_journal_get_write_access+0x38/0x80 [ext4]
[<ffffffffa00d9b23>] ext4_reserve_inode_write+0x73/0xa0 [ext4]
[<ffffffffa00d9b9c>] ext4_mark_inode_dirty+0x4c/0x1d0 [ext4]
[<ffffffffa00d9e90>] ext4_dirty_inode+0x40/0x60 [ext4]
[<ffffffff811ac48b>] __mark_inode_dirty+0x3b/0x160
[<ffffffff8119c742>] file_update_time+0xf2/0x170
[<ffffffff8111c1e0>] __generic_file_aio_write+0x230/0x490
[<ffffffff8111c4c8>] generic_file_aio_write+0x88/0x100
[<ffffffffa00d3fb1>] ext4_file_write+0x61/0x1e0 [ext4]
[<ffffffff81180f5b>] do_sync_readv_writev+0xfb/0x140
[<ffffffff81181ee6>] do_readv_writev+0xd6/0x1f0
[<ffffffff81182046>] vfs_writev+0x46/0x60
[<ffffffff81182102>] sys_pwritev+0xa2/0xc0
[<ffffffff8100b072>] system_call_fastpath+0x16/0x1b
[<ffffffffffffffff>] 0xffffffffffffffff
User stack:
    @       0x3a1ace10c4  (unknown)
    @          0x1262103  (unknown)
    @          0x12622d4  (unknown)
    @          0x12603df  (unknown)
    @           0x8e7bfb  (unknown)
    @           0x8f478b  (unknown)
    @           0x8f55db  (unknown)
    @          0x12a7b6f  (unknown)
    @       0x3a1b007851  (unknown)
    @       0x3a1ace894d  (unknown)
    @              (nil)  (unknown)
```

These traces can be useful for diagnosing root-cause latency issues in Kudu especially when they are caused by underlying systems such as disk controllers or filesystems.

Memory limits

Kudu has a hard and soft memory limit. The hard memory limit is the maximum amount a Kudu process is allowed to use, and is controlled by the --memory_limit_hard_bytes flag. The soft memory limit is a percentage of the hard memory limit, controlled by the flag memory_limit_soft_percentage and with a default value of 80%, that determines the amount of memory a process may use before it will start rejecting some write operations.

If the logs or RPC traces contain messages such as the following example, then Kudu is rejecting writes due to memory back pressure. This may result in write timeouts.

```
Service unavailable: Soft memory limit exceeded (at 96.35% of capacity)
```

There are several ways to relieve the memory pressure on Kudu:

- If the host has more memory available for Kudu, increase --memory_limit_hard_bytes.
- Increase the rate at which Kudu can flush writes from memory to disk by increasing the number of disks or increasing the number of maintenance manager threads --maintenance_manager_num_threads. Generally, the recommended ratio of maintenance manager threads to data directories is 1:3.
- Reduce the volume of writes flowing to Kudu on the application side.

Finally, in Kudu versions 1.7 and lower, check the value of the --block_cache_capacity_mb setting. This setting determines the maximum size of Kudu's block cache. While a higher value can help with read and write performance, setting it too high as a percentage of the --memory_limit_hard_bytes setting is harmful. Do not raise --block_cach e_capacity_mb above --memory_pressure_percentage (default 60%) of --memory_limit_hard_bytes, as this will cause Kudu to flush aggressively even if write throughput is low. The recommended value for --block_cache_capacity_mb is below the following:

(50% * --memory_pressure_percentage) * --memory_limit_hard_bytes

With the defaults, this means the --block_cache_capacity_mb should not exceed 30% of --memory_limit_hard_bytes.

In Kudu 1.8 and higher, servers will refuse to start if the block cache capacity exceeds the memory pressure threshold.

Block cache size

Kudu uses an LRU cache for recently read data. On workloads that scan a subset of the data repeatedly, raising the size of this cache can offer significant performance benefits. To increase the amount of memory dedicated to the block cache, increase the value of the --block_cache_capacity_mb flag. The default is 512 MiB.

Kudu provides a set of useful metrics for evaluating the performance of the block cache, which can be found on the /metrics endpoint of the Web UI. The following is an example set:

```
{
  "name": "block_cache_inserts",
  "value": 64
},
{
  "name": "block_cache_lookups",
  "value": 512
},
{
  "name": "block_cache_evictions",
  "value": 0
},
{
  "name": "block_cache_misses",
  "value": 96
},
{
  "name": "block_cache_misses_caching",
```

```
  "value": 64
},
{
  "name": "block_cache_hits",
  "value": 0
},
{
  "name": "block_cache_hits_caching",
  "value": 352
},
{
  "name": "block_cache_usage",
  "value": 6976
}
```

To judge the efficiency of the block cache on a tablet server, first wait until the server has been running and serving normal requests for some time, so the cache is not cold. Unless the server stores very little data or is idle, block_cache_usage should be equal or nearly equal to block_cache_capacity_mb. Once the cache has reached steady state, compare block_cache_lookups to block_cache_misses_caching. The latter metric counts the number of blocks that Kudu expected to read from cache but which weren't found in the cache. If a significant amount of lookups result in misses on expected cache hits, and theblock_cache_evictions metric is significant compared to block_cache_inserts, then raising the size of the block cache may provide a performance boost. However, the utility of the block cache is highly dependent on workload, so it's necessary to test the benefits of a larger block cache.

Attention: Do not raise the block cache size --block_cache_capacity_mb higher than the memory pressure threshold (defaults to 60% of --memory_limit_hard_bytes). As this would cause poor flushing behavior, Kudu servers version 1.8 and higher will refuse to start when misconfigured in this way.

Heap sampling

For advanced debugging of memory usage, administrators may enable heap sampling on Kudu daemons. This allows Kudu developers to associate memory usage with the specific lines of code and data structures responsible. When reporting a bug related to memory usage or an apparent memory leak, heap profiling can give quantitative data to pinpoint the issue.

Caution: Heap sampling is an advanced troubleshooting technique and may cause performance degradation or instability of the Kudu service. Currently it is not recommended to enable this in a production environment unless specifically requested by the Kudu development team.

To enable heap sampling on a Kudu daemon, pass the flag --heap-sample-every-n-bytes=524588. If heap sampling is enabled, the current sampled heap occupancy can be retrieved over HTTP by visiting http://tablet-server.example.com:8050/pprof/heap or http://master.example.com:8051/pprof/heap. The output is a machine-readable dump of the stack traces with their associated heap usage.

Rather than visiting the heap profile page directly in a web browser, it is typically more useful to use the pprof tool that is distributed as part of the gperftools open source project. For example, a developer with a local build tree can use the following command to collect the sampled heap usage and output an SVG diagram:

```
thirdparty/installed/uninstrumented/bin/pprof -svg  'http://localhost:8051/pprof/heap' > /tmp/heap.svg
```

The resulting SVG may be visualized in a web browser or sent to the Kudu community to help troubleshoot memory occupancy issues.

Tip: Heap samples contain only summary information about allocations and do not contain any data from the heap. It is safe to share heap samples in public without fear of exposing confidential or sensitive data.

Slow name resolution and nscd

For better scalability on nodes hosting many replicas, we recommend that you use nscd (name service cache daemon) to cache both DNS name resolution and static name resolution (via /etc/hosts).

When DNS lookups are slow, you will see a log message similar to the following:

```
W0926 11:19:01.339553 27231 net_util.cc:193] Time spent resolve address for
kudu-tserver.example.com: real 4.647s user 0.000s sys 0.000s
```

nscd can alleviate slow name resolution by providing a cache for the most common name service requests, such as for passwords, groups, and hosts.

Refer to your operating system documentation for how to install and enable nscd.

Usability issues

This topic lists some common exceptions and errors that you may encounter while using Kudu and helps you to resolve issues related to usability.

ClassNotFoundException: com.cloudera.kudu.hive.KuduStorageHandler

You will encounter this exception when you try to access a Kudu table using Hive. This is not a case of a missing jar, but simply that Impala stores Kudu metadata in Hive in a format that is unreadable to other tools, including Hive itself. and Spark. Currently, there is no workaround for Hive users. Spark users can work around this by creating temporary tables.

Runtime error: Could not create thread: Resource temporarily unavailable (error 11)

You may encounter this error when Kudu is unable to create more threads, usually on versions older than Kudu 1.7. It happens on tablet servers, and is a sign that the tablet server hosts too many tablet replicas.

To fix the issue, you can raise the nproc ulimit as detailed in the documentation for your operating system or distribution.

However, the better solution is to reduce the number of replicas on the tablet server. This may involve rethinking the table's partitioning schema. For the recommended limits on number of replicas per tablet server, see the *Scaling recommendations and limitations* topic.

Tombstoned or STOPPED tablet replicas

You may notice some replicas on a tablet server are in a STOPPED state and remain on the server indefinitely. These replicas are tombstones. A tombstone indicates that the tablet server once held a bona fide replica of its tablet.

For example, in case a tablet server goes down and its replicas are re-replicated elsewhere, if the tablet server rejoins the cluster, its replicas will become tombstones. A tombstone will remain until the table it belongs to is deleted, or a new replica of the same tablet is placed on the tablet server. A count of tombstoned replicas and details of each one are available on the /tablets page of the tablet server web UI. The Raft consensus algorithm that Kudu uses for replication requires tombstones for correctness in certain rare situations. They consume minimal resources and hold no data. They must not be deleted.

Corruption: checksum error on CFile block

In versions prior to Kudu 1.8.0, if the data on disk becomes corrupt, you will encounter warnings containing "Corruption: checksum error on CFile block" in the tablet server logs and client side errors when trying to scan tablets with corrupt CFile blocks. Fixing this corruption is a manual process.

To fix the issue, first identify all the affected tablets by running a checksum scan on the affected tables or tablets using the ksck tool.

```
sudo -u kudu kudu cluster ksck <master_addresses> -checksum_scan -tables=<ta
bles>
sudo -u kudu kudu cluster ksck <master_addresses> -checksum_scan -tablets=
<tablets>
```

If there is at least one replica for each tablet that does not return a corruption error, you can repair the bad copies by deleting them and forcing them to be re-replicated from the leader using the remote_replica delete tool.

```
sudo -u kudu kudu remote_replica delete <tserver_address> <tablet_id> "Cfile
 Corruption"
```

If all of the replica are corrupt, then some data loss has occurred. Until KUDU-2526 is completed, this can happen if the corrupt replica became the leader and the existing follower replicas are replaced.

If data has been lost, you can repair the table by replacing the corrupt tablet with an empty one using the unsafe_repla ce_tablet tool.

```
sudo -u kudu kudu tablet unsafe_replace_tablet <master_addresses> <table
t_id>
```

From versions 1.8.0 onwards, Kudu will mark the affected replicas as failed, leading to their automatic re-replication elsewhere.

Symbolizing stack traces

This topic helps you to identify whether there is a high contention among the threads to acquire a lock and a way to symbolize stack addresses.

Sometimes you might see the following in the logs:

```
0323 03:59:31.091198 (+607857us) spinlock_profiling.cc:243] Waited 492 ms on
 lock 0x4cb0960. stack: 0000000002398852 0000000000ad8c69 0000000000aa62ba 0
00000000221aaa8 000000000221b1a8 00000000023a8f83 00007fa8b818be24 00007fa8b
646a34c
```

That is usually a sign of high contention among threads to acquire a lock, and in this case the reported time shows how long a thread spent on a CPU before acquiring the lock. The call stack addresses that are listed help to restore the stack trace of the waiting thread and locate the problem in the code.

It is possible to translate the addresses into the name of functions and lines in the code having the binary that produced the output (in this example, it is kudu-master). If the binary is stripped of symbols and debug information, it is possible to do so if the debug information for the binary is available separately.

Assuming both the stripped release binary and the debug information are available as RPMs, unpack them into a directory; for example, sysroot:

```
$ mkdir sysroot && cd sysroot
$ rpm2cpio ../kudu-1.10.0.el7.x86_64.rpm | cpio -idmv
```

```
$ rpm2cpio ../kudu-debuginfo-1.10.0.el7.x86_64.rpm | cpio -idmv
```

Use addr2line to find the line in the code for the stack address. In case if the binary is not stripped of debug information, supply the actual binary with an -e option instead of the debug info file as follows:

```
addr2line -C -f -e usr/lib/debug/usr/lib/kudu/sbin-release/kudu-master.debug
 0x0000000000aa62ba
kudu::master::MasterServiceImpl::ConnectToMaster(kudu::master::ConnectToMast
erRequestPB const*, kudu::master::ConnectToMasterResponsePB*, kudu::rpc::Rpc
Context*)
/usr/src/debug/kudu-1.10.0/src/kudu/master/master_service.cc:504
```

To achieve the same with gdb, first find the address of the .text section in the symbol file (in the example, 00000000 00a2cdb0):

```
$ readelf -S usr/lib/debug/usr/lib/kudu/sbin-release/kudu-master.debug | gre
p .text
  [13] .text             NOBITS           0000000000a2cdb0  000002c0
```

Then start up gdb, pointing it to the kudu-master executable (that's the executable that produced the output in the log file):

```
gdb usr/lib/kudu/sbin-release/kudu-master
```

Now load the .debug symbols into gdb using the address found above. Tell gdb where to find source files and set the sysroot:

```
(gdb) add-symbol-file usr/lib/debug/usr/lib/kudu/sbin-release/kudu-master.de
bug 0x0000000000a2cdb0
(gdb) set substitute-path /usr/src/debug/kudu-1.10.0 usr/src/debug/kudu-1
.10.0
(gdb) set sysroot .
```

To translate the address into line number and function information, use info line * <address>:

```
(gdb) info line * 0x0000000000aa62ba
Line 504 of "/usr/src/debug/kudu-1.10.0/src/kudu/master/master_service.cc"
   starts at address 0xaa62af <kudu::master::MasterServiceImpl::ConnectToMas
ter(kudu::master::ConnectToMasterRequestPB const*, kudu::master::ConnectToMa
sterResponsePB*, kudu::rpc::RpcContext*)+47>
   and ends at 0xaa62bb <kudu::master::MasterServiceImpl::ConnectToMaster(k
udu::master::ConnectToMasterRequestPB const*, kudu::master::ConnectToMasterR
esponsePB*, kudu::rpc::RpcContext*)+59>.
```

Recover from a dead Kudu master in a multi-master deployment

Kudu multi-master deployments function normally in the event of a master loss. However, it is important to replace the dead master. Otherwise a second failure may lead to a loss of availability, depending on the number of available masters. This workflow describes how to replace the dead master.

Due to KUDU-1620, it is not possible to perform this workflow without also restarting the live masters. As such, the workflow requires a maintenance window, albeit a potentially brief one if the cluster was set up with DNS aliases.

Important:

- Replacing a master created without DNS aliases requires an unavailability window when tablet servers are restarted to pick up the replacement master at different hostname. See the multi-master migration workflow in *Migrating to multiple Kudu masters* for more details on deploying with DNS aliases.
- The workflow presupposes at least basic familiarity with Kudu configuration management. If using Cloudera Manager, the workflow also presupposes familiarity with it.
- All of the command line steps below should be executed as the Kudu UNIX user, typically kudu.

Prepare for the recovery

It is crucial to make sure that the master node is truly dead and does not accidentally restart while you are preparing for the recovery.

Procedure

1. If the cluster was configured without DNS aliases perform the following steps. Otherwise move on to step 2:
   a) Establish a maintenance window (one hour should be sufficient). During this time the Kudu cluster will be unavailable.
   b) Shut down all Kudu tablet server processes in the cluster.
2. Ensure that the dead master is well and truly dead. Take whatever steps needed to prevent it from accidentally restarting; this can be quite dangerous for the cluster post-recovery.
3. Choose an unused machine in the cluster where the new master will live. The master generates very little load so it can be co-located with other data services or load-generating processes, though not with another Kudu master from the same configuration. The rest of this workflow will refer to this master as the "replacement" master.
4. Perform the following preparatory steps for the replacement master:
   - If using the same dead master as the replacement master then delete the master's directories.
   - Ensure Kudu is installed on the machine, either via system packages (in which case the kudu and kudu-master packages should be installed), or via some other means.
   - Choose and record the directory where the master's data will live.

Perform the recovery

After you have identified a reference master, you need to copy the master data to the replacement master node. You need to bring the Kudu clusters down. Therefore, identify at least a one-hour maintenance window for this task.

Procedure

1. Remove the dead master from the Raft configuration of the master using the kudu master remove command.

   In the following example dead master master-2 is being recovered:

   ```
   $ sudo -u kudu kudu master remove master-1,master-2 master-2
   ```

2. On the replacement master host, add the replacement master to the cluster using the kudu master add command.

   Look for any success or error messages on the console or the replacement master log file. The command is designed to be idempotent so in case of an error after the issue mentioned in the error messages is fixed, run the same command again to make forward progress. After the completion of the procedure irrespective of whether the procedure is successful, the replacement master is shutdown. In the following example, replacement master mast er-2 is used. In case DNS alias is not being used, use the hostname of the replacement master.

   ```
   $ sudo -u kudu kudu master add master-1 master-2 --fs_wal_dir=/data/kudu/
   master/wal \
   --fs_data_dirs=/data/kudu/master/data
   ```

3. If the cluster was set up with DNS aliases, reconfigure the DNS alias for the dead master to point at the replacement master.
4. If the cluster was set up without DNS aliases, perform the following steps:
   a) Modify the value of the master_addresses configuration parameter for each live master removing the dead master and substituting it with the replacement master.

      The new value must be a comma-separated list of all of the masters. Each entry is a string of the form <hostname>:<port>:

      hostname

      The master's previously recorded hostname or alias

      port

      The master's previously recorded RPC port number
   b) Restart the remaining live masters.
5. Start the replacement master.
6. If the cluster was set up without DNS aliases, perform the following steps for tablet servers:
   a) Modify the value of the tserver_master_addr configuration parameter for each live master removing the dead master and substituting it with the replacement master.

      The new value must be a comma-separated list of all of the masters. Each entry is a string of the form <hostname>:<port>:

      hostname

      The master's previously recorded hostname or alias

      port

      The master's previously recorded RPC port number
   b) Restart all tablet servers.

What to do next

To verify that all masters are working properly, consider performing the following checks:

- Using a browser, visit each master’s web UI and navigate to the /masters page. All the masters should now be listed there with one master in the LEADER role and the others in the FOLLOWER role. The contents of /masters on each master should be the same.
- Run a Kudu system check (ksck) on the cluster using the kudu command line tool.

Spark Troubleshooting

Date published: 2024-09-23
Date modified:

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Spark jobs failing with memory issues.....4

Spark jobs failing with memory issues

This article describes troubleshooting tips for when Spark application fails with memory issues such as java.lang.OutOfMemoryError : *.

Condition

The general symptom for this issue is that Spark applications fail with one of the errors listed below.

This can either be a new application or an application that has been previously working, but is now failing after an increase in the data size being processed or after a configuration change. The specific symptoms for this issue, which are contained in the log output, are exposed in the Spark driver, Spark executor, YARN Application Master, and NodeManager logs.

Symptoms Error messages include:

```
java.lang.OutOfMemoryError: Java heap space
```

```
java.lang.OutOfMemoryError: GC overhead limit exceeded
```

```
java.lang.OutOfMemoryError: unable to create new native thread
```

```
java.lang.OutOfMemoryError: Not enough memory to build and broadcast the tab
le to all worker nodes. As a workaround, you can either disable broadcast by
 setting
            spark.sql.autoBroadcastJoinThreshold to -1 or increase the spark
 driver memory by setting spark.driver.memory to a higher value
```

```
java.lang.OutOfMemoryError: Unable to acquire * bytes of memory, got 0
```

```
java.lang.OutOfMemoryError: Requested array size exceeds VM limit
```

```
ERROR org.apache.spark.executor.Executor: Managed memory leak detected; size
 = * bytes, TID = *
```

```
containerID=container_*] is running [*** VALUE B***] beyond the 'PHYSICAL' m
emory limit. Current usage: * GB of * GB physical memory used; * GB of * GB
virtual memory used. Killing container.
```

Also, the job may report WARN messages such as

```
20/11/24 20:23:20 WARN memory.MemoryStore: Not enough space to cache rdd_343
3_344 in memory! (computed 276.3 MB so far)
```

Cause

When Spark runs on YARN, containers memory is allocated as per the Spark configurations set.

You can configure it using the spark.executor|driver property (the default value is the driver|executor with a minimum of 1024 MB).

When the configured memory is too small to accommodate the needs of a running application, the application containers exceed the memory limit set by YARN and are killed by the framework. When enough containers are killed, the Spark application fails.

Understanding Spark configurations related to memory

This section provides an overview of Spark configuration order of precedence rules and instructions for resolving issues with YARN killing containers for exceeding memory limits.

1. **Understanding the Spark Configuration Order of Precedence**

   When configuring Spark application, you need to understand the order of precedence for configuration values. Configuration values explicitly set in code(SparkConf) take the highest precedence, then flags passed to the spark-submit command, then values in the Spark defaults configuration file.

   The exact order of precedence is:

   - Spark configuration file: Parameters are passed to Spark Conf.
   - Command-line arguments are passed to spark-submit, spark-shell, or pyspark.
   - Cloudera Manager UI properties set in the spark-defaults.conf configuration file.

   a. Available Spark configuration properties

   - spark.executor.memory
   - spark.driver.memory
   - spark.yarn.am.memory
   - spark.yarn.executor.memoryOverhead
   - spark.yarn.driver.memoryOverhead

   b. Using Command-line arguments

   You can set the above configuration properties when launching Spark applications using the spark-submit command:

   ```
   ./bin/spark-submit --conf spark.executor.memory=[*** VALUE ***] --conf
   spark.driver.memory=[*** VALUE ***]
   ```

2. **Identify what type of containers are being killed**

   Review the logs and identify which container is being killed, that is: the driver, executor, or application master container.

   If the container being killed is the executor container, Use the container ID in the NodeManager logs in conjunction with the application logs, to identify the error if not logged in Executor container logs.

3. **Verifying the actual memory values used**

   Before increasing the Driver or executor memory, access the Spark application properties through the Spark UI:

   - For each application, run a job and open the **Spark UI**.
   - Click on the Environment tab.
   - Scroll to the Spark Properties section and view the **Memory** values.

   Note: Refer to *Tuning Spark Applications* for more information about fine tuning Spark.

YARN Capacity Scheduler's container sizing

YARN Capacity Scheduler, the default scheduler in Cloudera, multiplies the size of the minimum allocation for containers. With a minimum scheduler of 1GB memory per container, for a request for a container size of 4.5GB the scheduler rounds up to 5GB. Minimum allocations set to very high values can create large resource waste problems: for example, with a 4GB minimum allocation a request for 5GB would be served 8GB - creating 3 extra GB allocation that is not used! Cloudera recommends that when configuring and minimum and maximum container size, the maximum should be evenly divisible by the minimum.

If you are using the YARN capacity scheduler, memory requests will be rounded up to the nearest multiple of the value of yarn.scheduler.minimum-allocation-mb in the YARN configuration in Cloudera Manager.

For more information, refer to *YARN – The Capacity Scheduler*.

Error 1 : java.lang.OutOfMemoryError: Java heap space

Driver side causes

Possible causes when the driver can go OOM are:

- Running a Collect/Show/ operation on a large file
- Memory leak

Executor side causes

Possible causes when the executor can go OOM are:

- Possible causes data skewness
- Less partitions or the data size is larger than the memory allocated
- Memory leak - where the memory usage increases sharply leading to OOM, even while less or no data is being processed.

Possible solutions

- Increase the memory for the driver or executor depending on the error message. You need to try this until the job succeeds and we recommend using spark-submit command line arguments.
- Check the Spark event log , if the last operation that caused the failure is due to Collect/show , then check with the customer on the necessity to perform this case. This is generally done over testing propose for small data size and is not needed over PROD or larger dataset runs.

Error 2 : java.lang.OutOfMemoryError: GC overhead limit exceeded

Driver side causes

Possible causes when the driver can go OOM are:

- Running a Collect/Show/ operation on a large file
- Memory leak

Executor side causes

- Possible causes data skewness
- Less partitions or the data size is larger than the memory allocated
- Memory leak

Possible solutions

- Increase the memory for the driver or executor depending on the error message. You need to try this until the job succeeds and we recommend using spark-submit command line arguments.
- Check the Spark event log , if the last operation that caused the failure is due to Collect/show , then check with the customer on the necessity to perform this case. This is generally done over testing propose for small data size and is not needed over PROD or larger dataset runs.
- If even after multiple re-tires of increasing the driver or executor memory and job fails with processing small amount of data, then it can be a issues with Memory Leak. Take Heap Dump via steps to take heap dump on Spark application.
- For data skewness issue, please check this article on how to identify and the solutions to resolve the same.

Error 3 : java.lang.OutOfMemoryError: unable to create new native thread

Possible causes when the driver/executor can go OOM are

This indicates that that the application has hit the limit of how many threads it can launch, which is dictated by the OS (the number it will allow your JVM to use). This in turn usually either means that you need to make changes to the code or increase the ulimit (nproc) on all NodeManager hosts.

Possible solutions

In the /etc/security/limits.d/ directory, edit the file *[*** NUMBER ***]*-nproc.conf, and make sure the user who needs higher nproc settings has the appropriate limit configured as follows:

```
userid soft nproc 65536
```

The above is only an example, and should be modified as desired on all NodeManager hosts.

Error 4 : java.lang.OutOfMemoryError: Not enough memory to build and broadcast the table to all worker nodes.

Possible solutions

- Increase the memory for the driver depending on the error message.
- Check the table size being broadcasted. It should be noted that we should always broadcast a smaller table. Users can implicitly specify Broadcast Hint (see *Join Strategy Hints for SQL Queries* in the Apache Spark Documentation) in the query.
- Disable broadcast by setting spark.sql.autoBroadcastJoinThreshold to **-1**.

Error 5 : java.lang.OutOfMemoryError: Unable to acquire * bytes of memory, got 0

Possible causes when the driver can go OOM are:

- Possible causes data skewness
- Less partitions or the data size is larger than the memory allocated
- Memory leak
- System does not have enough memory to allocate resources

Possible solutions

- Increase Executor memory.
- Increase Executor memory overhead:

  ```
  --conf spark.yarn.executor.memoryOverhead=[*** VALUE ***]
  ```

  The *[*** VALUE ***]* is usually 10% of the value specified in --executor-memory.
- Check system setting on Free memory available.

Error 5 : java.lang.OutOfMemoryError: Requested array size exceeds VM limit

Possible causes when the driver can go OOM are:

Shuffle block size is greater than the 2GB Shuffle Size limit . The issue you're experiencing here has to do with the 2 GB block limit in Spark.

Possible solutions

Consider increasing the following:

- the number of partitions for your RDDs in the code by using either the RDD's coalesce() or repartition() method.
- >Executor memory

Error 5 : ERROR org.apache.spark.executor.Executor: Managed memory leak detected; size = * bytes, TID = *

Managed memory leak error in executors issue occurs when the shuffle read requires more memory than what is available for the task. Since the shuffle-read side doubles its memory request each time, it can easily end up acquiring all of the available memory, even if it does not use it.

Note: The job will report with one of the above java.lang.OutOfMemoryError: * reported in the executor logs along with the above Managed memory leak detected* error.

Possible solutions

- To increase Driver memory overhead:

  ```
  --conf spark.yarn.driver.memoryOverhead=[*** VALUE ***]
  ```

  The *[*** VALUE ***]* is usually 10% of the value specified in --executor-memory.
- To Increase Executor memory overhead:

  ```
  --conf spark.yarn.executor.memoryOverhead=[*** VALUE ***]
  ```

  The *[*** VALUE ***]* is usually 10% of the value specified in --executor-memory.

containerID=container_*] is running *[*** VALUE B***]* beyond the 'PHYSICAL' memory limit. Current usage: * GB of * GB physical memory used; * GB of * GB virtual memory used. Killing container.

Possible causes when the executor can go OOM are:

- Possible causes data skewness
- Less partitions or the data size is larger than the memory allocated
- Memory leak

Note: The job may report with one of the above java.lang.OutOfMemoryError: * reported in the executor logs along with the above beyond the 'PHYSICAL' memory limit error.

Possible solutions

Consider increasing Executor memory:

- To Increase Executor memory overhead:

  ```
  --conf spark.yarn.executor.memoryOverhead=[*** VALUE ***]
  ```

  The *[*** VALUE ***]* is usually 10% of the value specified in --executor-memory.

WARN Message: WARN memory.MemoryStore: Not enough space to cache rdd*

Data spilling to disk and no memory to Cache RDD. Indicates memory being easily filled and it can cause performance issues and also application failure with OOM errors as above.

1. To avoid frequent spilling you would need to increase the heap size of Spark executor.
2. Increase the partition count to have more distribution of data. Review *Tuning Partitions in Spark*.
3. Memory leak

Possible solutions

- Increase the executor or driver memory Overhead (spark.executor.memoryOverhead and -spark.driver.memory Overhead).

  The memory overhead defaults to 10% of the memory specified (with a minimum of 384mb). Thus increasing the executor or driver memory can also increase the memory overhead if sufficiently large values are used.
- Decrease the number of executor cores (--executor-cores)

  By decreasing the number of parallel tasks being run in executors, each executor requires less memory overhead.

Related Information

Tuning Partitions in Spark

Join Strategy Hints for SQL Queries | Apache Spark Documentation

Tuning Spark Applications

YARN – The Capacity Scheduler | Cloudera Blog

Cloudera Runtime 7.3.1

Sqoop Troubleshooting

Date published: 2023-05-16
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Unable to read Sqoop metastore created by an older HSQLDB version..... 4

Merge process stops during Sqoop incremental imports.....5

Sqoop Hive import stops when HS2 does not use Kerberos authentication..... 6

Unable to read Sqoop metastore created by an older HSQLDB version

After upgrading to certain Cloudera versions, you may encounter issues in reading the Sqoop metastore that was created using an older version of HyperSQL Database (HSQLDB). Learn how to upgrade the Sqoop metastore to resolve this issue.

Cloudera upgraded the HSQLDB dependency in the following versions to the newest HSQLDB 2.7.1 version:

- Cloudera Private Cloud Base 7.1.7 SP2
- Cloudera Private Cloud Base 7.1.8 Cumulative hotfix 4
- Cloudera Public Cloud 7.2.16

The version upgrade causes incompatibility issues in Sqoop jobs that are stored in HSQLDB. Therefore, the Sqoop metastore should be upgraded after you upgrade to the above-mentioned Cloudera versions or higher.

Note: Sqoop stores its metastore files in the current user's home directory. If the Sqoop metastore is on-demand, then its location is ~/.sqoop/metastore.db.If the Sqoop metastore is a service, then its location is ~/.sqoop/shared-metastore.db.

Choose one of the following procedures based on how the Sqoop metastore is created:

For Sqoop metastore created using Sqoop

Perform the following steps if the Sqoop metastore is created using Sqoop:

1. If the Sqoop metastore is started as a service then stop the service by running this command on the host where the service is running:

```
/opt/cloudera/parcels/CDH/lib/sqoop/stop-metastore.sh -p <path to Sqoop
 metastore pid file>
```

2. Download HSQLDB 2.3.6 and SqlTool JAR files to the host by running the following commands:

```
wget https://repo1.maven.org/maven2/org/hsqldb/sqltool/2.3.6/sqltool-2.3
.6.jar
wget https://repo1.maven.org/maven2/org/hsqldb/hsqldb/2.3.6/hsqldb-2.
3.6.jar
```

3. Run the following command to upgrade and convert the Sqoop metastore database files using SqlTool:

   For on-demand Sqoop metastore:

```
$JAVA_HOME/bin/java -cp "sqltool-2.3.6.jar:hsqldb-2.3.6.jar" org.hsqldb
.cmdline.SqlTool --driver org.hsqldb.jdbc.JDBCDriver  --inlineRc="url=jd
bc:hsqldb:file:~/.sqoop/metastore.db;shutdown=true;user=SA,password=" --
sql="SELECT * FROM SQOOP_ROOT;"
```

   For Sqoop metastore as a service:

```
$JAVA_HOME/bin/java -cp "sqltool-2.3.6.jar:hsqldb-2.3.6.jar" org.hsqldb.
cmdline.SqlTool --driver org.hsqldb.jdbc.JDBCDriver  --inlineRc="url=jdb
```

```
c:hsqldb:file:~/.sqoop/shared-metastore.db;shutdown=true;user=SA,passwor
d=" --sql="SELECT * FROM SQOOP_ROOT;"
```

The Sqoop metastore is upgraded and the database files are converted to a format that can easily be read by HSQLDB 2.7.1.

Note: As part of the Cloudera upgrade, Apache Sqoop installation has the dependent HSQLDB 2.7.1 version by default.

4. If the Sqoop metastore service was stopped earlier, restart the service:

```
/opt/cloudera/parcels/CDH/lib/sqoop/start-metastore.sh -p <path to Sqoop
 metastore pid file> -l <path to Sqoop metastore logs dir>
```

5. You can choose to remove the downloaded HSQLDB JAR files from the host.

For Sqoop metastore created in a separate HSQLDB instance

Perform the following steps if the Sqoop metastore is created in a separate HSQLDB instance:

1. Login to the HSQLDB service host and stop the service. For more information, see the HSQLDB documentation.
2. Work with your IT team to determine the HSQLDB store file that is created to store the Sqoop metastore information.
3. Download HSQLDB 2.3.6 and SqlTool JAR files to the host by running the following commands:

```
wget https://repo1.maven.org/maven2/org/hsqldb/sqltool/2.3.6/sqltool-2.3
.6.jar
wget https://repo1.maven.org/maven2/org/hsqldb/hsqldb/2.3.6/hsqldb-2.
3.6.jar
```

4. Run the following command to upgrade and convert the Sqoop metastore database files using SqlTool:

```
$JAVA_HOME/bin/java -cp "sqltool-2.3.6.jar:hsqldb-2.3.6.jar" org.hsqldb.
cmdline.SqlTool --driver org.hsqldb.jdbc.JDBCDriver  --inlineRc="url=jdb
c:hsqldb:file:<PATH OF SQOOP METASTORE FILE>;shutdown=true;user=SA,pass
word=" --sql="SELECT * FROM SQOOP_ROOT;"
```

5. If the HSQLDB service hosts multiple databases then each of the store files must be upgraded as described in the previous step.

   The Sqoop metastore is upgraded and the database files are converted to a format that can easily be read by HSQLDB 2.7.1.

   Note: As part of the Cloudera upgrade, Apache Sqoop installation has the dependent HSQLDB 2.7.1 version by default.

6. Restart the HSQLDB service. For more information, see the HSQLDB documentation.
7. You can choose to remove the downloaded HSQLDB JAR files from the host.

Related Information

HyperSQL Documentation

Merge process stops during Sqoop incremental imports

During Sqoop incremental import operations, if the target directory is located outside of Hadoop Distributed File System (HDFS), such as in Amazon S3 or Azure Blob Storage, the merge phase of the import process does not take effect.

Condition

Sqoop, by default, creates temporary directories within HDFS. However, you must be aware of certain considerations in choosing the target directory location while working with Sqoop's incremental import modes. By default, Sqoop operates seamlessly when the target directory resides within HDFS. However, the merge phase of the import process does not take effect outside the box if the target directory is located outside of HDFS.

Cause

During an import operation, Sqoop generally imports data to a target directory. If this target directory is a non-HDFS location, the merge process tries to acquire the temporary directory required for the merge on the same non-HDFS file system. Since Sqoop creates the temporary directory in HDFS by default, the merge process checks if the temporary directory exists in the target directory's file system and when it does not find it, the merge process simply stops.

Solution

If the target directory is present outside of HDFS, you must modify the default path of the temporary directory by adding the --temporary-rootdir Sqoop option and pointing to a path on the same file where the target directory is located. By aligning the temporary directory path with the file system of the target directory, Sqoop can effectively complete the import process.

Example:

Include the --temporary-rootdir Sqoop option as shown below:

```
sqoop-import --connect jdbc:mysql://.../transaction --username [***USER NAME
***] --table [***TABLE NAME***] --password [***PASSWORD***] --target-dir abf
s://foo@bar/targetdir -m 1 --temporary-rootdir abfs://foo@bar/_sqoop
```

Sqoop Hive import stops when HS2 does not use Kerberos authentication

Learn how to resolve the issue related to Sqoop Hive imports when either LDAP authentication or no authentication mechanism is enabled for the cluster.

Condition

When running Sqoop commands to import data into Hive from either the CLI or Oozie, the import job stops after the Sqoop import is done and while trying to connect to HiveServer (HS2) through JDBC. The following log is displayed and you will notice that the job stops on the last line:

```
23/07/24 18:10:17 INFO hive.HiveImport: Loading uploaded data into Hive
23/07/24 18:10:17 INFO hive.HiveImport: Collecting environment variables whi
ch need to be preserved for beeline invocation
...
23/07/24 18:10:20 INFO hive.HiveImport: SLF4J: Actual binding is of type
[org.apache.logging.slf4j.Log4jLoggerFactory]
23/07/24 18:10:21 INFO hive.HiveImport: Connecting to jdbc:hive2://HOSTNAME/
default;serviceDiscoveryMode=zooKeeper;ssl=true;sslTrustStore=/var/lib/clo
udera-scm-agent/agent-cert/cm-auto-global_truststore.jks;trustStorePassword=
changeit;zooKeeperNamespace=hiveserver2
```

Cause

This issue occurs when Kerberos is not used in the JDBC connection string, which Sqoop uses to connect to HS2. The issue affects unsecure clusters and clusters where LDAP authentication is enabled, and the beeline-site.xml configuration file does not use Kerberos authentication.

The underlying issue is that Beeline prompts for the username and password for a successful connection and since the Sqoop Hive import is a non-interactive session, you are unable to provide the credentials and therefore the import job stops.

Solution

Procedure

Perform one of the following steps to resolve this issue:

If...	Then...
No authentication is enabled for the cluster	Include the --hs2-url option in the Sqoop import command and provide the JDBC connection string. `--hs2-url <HS2 JDBC string>` This allows for a successful connection without prompting for the credentials.

LDAP authentication is enabled for the cluster	Include the --hs2-user and --hs2-password options in the Sqoop import command and provide the credentials.
	`--hs2-user <username>`
	`--hs2-password <password>`

Cloudera Runtime 7.3.1

Tuning Hue

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Adding a load balancer **4**

Configuring high availability for Hue **4**

Configuring Hive and Impala for high availability with Hue 5

Configuring for HDFS high availability **7**

Configuring dedicated Impala coordinator **8**

Configuring the Hue Query Processor scan frequency **8**

Adding a load balancer

Cloudera recommends you to configure load balancers for Hue to improve performance or if you have more than 10 concurrent users, and access Hue through the Hue load balancer port (8889).

Procedure

1. Log into Cloudera Manager as an Administrator.
2. Go to Clusters Hue service .
3. Click Actions and select Add Role Instances.

   The Add Role Instances wizard is displayed.
4. On the **Assign Roles** page, click Select hosts for Load Balancer.

   Select a host and click OK.
5. [Optional] Add 2 additional Hue servers (for a total of 3) to boost performance:
   a. Click Select hosts for Hue Server.
   b. Select a host and click OKContinue.
6. Click Select Hosts for Kerberos Ticket Renewer if running Kerberos and select each host that you selected earlier.

   Every Hue host must have a Renewer.
7. Click Continue.

   The newly added hosts are displayed on the Instances tab.
8. Select all the newly added hosts.
9. Click Actions for Selected and select Start.

   Note: Hue servers can share hosts with Load Balancers. But Hue servers must be on distinct hosts from other Hue servers, and Load Balancers must be on distinct hosts from other Load Balancers.
10. Restart the Hue service.
11. Click Hue Web UILoad Balanced Hue Web UI.
12. Log on to Hue and ensure the port is 8889.

    Tip: The Load Balancer instance can always be accessed on the Hue Instances tab.

Configuring high availability for Hue

To enable high availabilty for Hue, you must configure at least two Hue servers, each on a different host and a load balancer. For secure connections, enable TLS/SSL and install Kerberos. Configuring Hue for high availability includes configuring Hue, Hive, and Impala.

Before you begin

- Provide SSH network access to host machines with a Hue Server/Kerberos Ticker Renewer role.
- Configure each Hue server to point to an external database. Cloudera recommends to use an external database for clusters with multiple Hue servers so that no matter which server is being used, the data is always accessible. In a multi-server environment with a default embedded database (one per server), the data on server "A" can appear lost when working on server "B" if the server "A" becomes unavailable (which has the Hue backend database) and vice versa.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Add Hue roles:

   Hue HA requires at least two Hue server roles and one Load Balancer role. If the cluster is authenticating with Keberos, you need one Kerberos Ticker Renewer on each host with a Hue Server.

   a) Go to the Hue service and select Actions Add Role Instances .
   b) Click Hue Server, assign to one or more hosts, and click OK Continue .
   c) Click Kerberos Ticket Renewer, assign to each host with a Hue Server, and click OK Continue .
   d) Click Load Balancer, assign to one or more hosts, and click OK Continue .
   e) Check each role and select Actions for Selected Start and click Start.
3. Enable TLS for the Hue Load Balancer:

   Note: You can configure the Load Balancer for TLS/SSL or each endpoint (HS2, Impalad).

   a) Go to Hue Configuration and search on TLS/SSL.
   b) Check Enable TLS/SSL for Hue for the Hue Server Default Group.
   c) Set other TLS/SSL properties appropriate for your setup.

   Some properties to consider are:

   - Hue Load Balancer Port: Apache Load Balancer listens on this port (default is 8889).
   - Path to TLS/SSL Certificate File: Must be multi-domain with CN = Load Balancer in PEM format.
   - Path to TLS/SSL Private Key File: Must be in PEM format.
4. Select the Hue Load Balancer Cookie Refresh option from the Configuration tab to optimize the Hue Load Balancer to distribute users equally across the Hue instances.
5. Click Save Changes.
6. Restart the Hue service.

Configuring Hive and Impala for high availability with Hue

To configure Hive for high availability with Hue, you must have two or more HiveServer2 roles. For Impala, you must have two or more Impala daemon (impalad) roles.

Before you begin

- SSH network access to host machines with a HiveServer2 or Impala Daemon role.
- External database configured for each H2S and Impala Daemon.
- Hue Load Balancer Hive/Impala Load Balancer configured with Source IP Persistence.

Source IP Persistence

Without IP Persistence, you may encounter the error, “Results have expired, rerun the query if needed.

Hue supports High Availability through a "load balancer" to HiveServer2 and Impala. Because the underlying Hue thrift libraries reuse TCP connections in a pool, a single user session may not have the same TCP connection. If a TCP connection is balanced away from a HiveServer2 or Impalad instance, the user session and its queries (running or returned) can be lost and trigger the “Results have expired" error.

To prevent sessions from being lost, configure the Hive/Impala Load Balancer with Source IP Persistence so that each Hue instance sends all traffic to a single HiveServer2/Impala instance. Of course, this is not true load balancing, but a configuration for failover High Availability.

To prevent sessions from timing out while in use, add more Hue Server instances, so that each can be pinned to another HiveServer2/Impala instance. And for both HiveServer2/Impala, set the affinity timeout (that is, the timeout to close persisted sessions) to be longer than the Impala query and session timeouts.

For the best load distribution, create multiple profiles in your load balancer, per port, for both non-Hue clients and Hue clients. Have non-Hue clients distribute loads in a round robin and configure Hue clients with source IP Persistence on dedicated ports, for example, 21000 for impala-shell, 21050 for impala-jdbc, and 21051 for Hue.

Procedure

1. In Cloudera Manager, add roles for HiveServer2 and for Impala daemon:
   a. Configure the cluster with at least two roles for HiveServer2:
      1. Go to the Hive service and select Actions Add Role Instances .
      2. Click HiveServer2, assign one or more hosts, and click OK Continue .
      3. Check each role and select Actions for Selected Start and click Start.
   b. Configure the cluster with at least two roles for Impala Daemon:
      1. Go to the Impala service and select Actions Add Role Instances .
      2. Click Impala Daemon, assign one or more hosts, and click OK Continue .
      3. Check each role and select Actions for Selected Start and click Start.
2. Install a proxy service:

   This is an example of how to add a proxy server for each HiveServer2 and Impala Daemon with multiple profiles using the open source TCP/HTTP load balancer, HAProxy.

   a. Install HAProxy for your operating system:

```
yum install haproxy
```

```
apt-get install haproxy
```

```
zypper addrepo http://download.opensuse.org/repositories/server:http/SLE
_12/server:http.repo
zypper refresh
zypper install haproxy
```

   b. Configure HAProxy for each role, for example:

```
vi /etc/haproxy/haproxy.cfg
```

```
listen impala-shell
   bind :21001
   mode tcp
   option tcplog
   balance roundrobin
   stick-table type ip size 20k expire 5m
server impala_0 shortname-2.domain:21000 check
server impala_1 shortname-3.domain:21000 check

listen impala-jdbc
   bind :21051
   mode tcp
   option tcplog
   balance roundrobin
   stick-table type ip size 20k expire 5m
server impala_0 shortname-2.domain:21050 check
server impala_1 shortname-3.domain:21050 check

listen impala-hue
   bind :21052
   mode tcp
   option tcplog
```

```
    balance source
server impala_0 shortname-2.domain:21050 check
server impala_1 shortname-3.domain:21050 check

listen hiveserver2-jdbc
    bind :10001
    mode http
    option tcplog
    balance roundrobin
    stick-table type ip size 20k expire 5m
server hiveserver2_0 shortname-1.domain:10000 check
server hiveserver2_1 shortname-2.domain:10000 check
listen hiveserver2-hue
    bind :10002
    mode tcp
    option tcplog
    balance source
     stick-table type ip size 20k expire 5m
server hiveserver2_0 host shortname-1.domain:10000 check
server hiveserver2_1 host shortname-2.domain:10000 check
```

Replace shortname-#.domain with those in your environment:

```
sed -i "s/host shortname/your host shortname/g" /etc/haproxy/haproxy.cfg
sed -i "s/domain/your domain/g" /etc/haproxy/haproxy.cfg
```

Hue does not maintain any state for the connection to HS2. If Hue connects to one HS2 instance and the query is being executed on another HS2 instance in an HA setup, then commands including listing tables or databased could fail. Enabling a sticky session using stick table is used to avoid this issue.

When you specify “stick-table type ip size 20k expire 5m”, the stick table tracks the HTTP request rate of each client that passes through the load balancer.

- The table's primary key is of the type ip, which implies that the keys are IP addresses
- The table holds a maximum of 20 thousand records
- A record expires after 5 minutes unless it is accessed during that time

c. Restart haproxy:

```
service haproxy restart
```

d. Run netstat to ensure your proxies are running:

```
netstat | grep LISTEN
```

Configuring for HDFS high availability

You can use Cloudera Manager to configure Hue to use HDFS high availability NameNodes.

Procedure

1. Add the HttpFS role.
2. After the command has completed, go to the Hue service.
3. Click the Configuration tab.
4. Locate the HDFS Web Interface Role property or search for it by typing its name in the Search box.
5. Select the HttpFS role you just created instead of the NameNode role, and save your changes.
6. Restart the Hue service.

Related Information

Using HttpFS to provide access to HDFS

Configuring Hue to use a dedicated Impala coordinator

If you have separate Impala executors and coordinators on your Cloudera cluster, then you can configure Hue to use a dedicated Impala coordinator for running queries.

About this task

Note: Hue must connect to the same Impala daemon for fetching query results on which the query was executed. Therefore, do not configure the load balancer using a simple round-robin algorithm because the Impala daemons are only aware of the queries they handle.

Procedure

1. Log into Cloudera Manager as an Administrator.
2. Go to Clusters Hue Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini and add the following lines under the [impala] section:

```
[impala]
#The Impala coordinator server
server_host=[***COORDINATOR-SERVER***]:[***PORT***]
```

3. Click Save Changes.
4. Restart the Hue service.

Results

Hue uses the dedicated Impala coordinator server to execute queries.

Configuring the Hue Query Processor scan frequency

You can configure the frequency at which the Hue Query Processor fetches the data from a Hadoop Compatible File System by setting the time in milliseconds (ms) in the “hue.query-processor.event-pipeline.folder-scan-delay-millis” property in the Query Processor configurations.

About this task

By default, the value of the hue.query-processor.event-pipeline.folder-scan-delay-millis property is 5 * 60 * 1000 ms (5 minutes). For a faster data refresh rate, you can set this value to less than 5 * 60 * 1000 ms (5 minutes). But this can result in higher storage read costs. However, if you want stronger performance, and a faster refresh rate is not a priority, then you can set it to a higher value to reduce storage read costs.

Procedure

1. Log in to the Cloudera web interface as an Administrator.
2. Go to Data Hub Clusters and select your cluster.
3. Click CM URL or CM-UI to open Cloudera Manager.

4. Go to Clusters Query Processor service Configuration and add the following line in the Query Processor Extra Configurations field:

```
"hue.query-processor.event-pipeline.folder-scan-delay-millis" : [***TIME-IN-MS***]
```

```
"hue.query-processor.event-pipeline.folder-scan-delay-millis" : 2000
```

5. Click Save Changes.
6. Restart the Query Processor service.

Cloudera Runtime 7.3.1

Tuning Apache Spark

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction 4

Check Job Status 4

Check Job History 4

Improving Software Performance 4

Tuning Apache Spark Applications 6

Tuning Spark Shuffle Operations 6
Choosing Transformations to Minimize Shuffles 7
When Shuffles Do Not Occur 8
When to Add a Shuffle Transformation 9
Secondary Sort 10
Tuning Resource Allocation 10
Resource Tuning Example 11
Tuning the Number of Partitions 11
Reducing the Size of Data Structures 13
Choosing Data Formats 13

Introduction

This section provides information about evaluating and tuning Spark performance.

When tuning Apache Spark applications, it is important to understand how Spark works and what types of resources your application requires. For example, machine learning tasks are usually CPU intensive, whereas extract, transform, load (ETL) operations are I/O intensive.

Check Job Status

If a job takes longer than expected or does not finish successfully, check the following to understand more about where the job stalled or failed:

- To list running applications by ID from the command line, use yarn application –list.
- To see a description of a resilient distributed dataset (RDD) and its recursive dependencies (useful for understanding how jobs are executed) use toDebugString() on the RDD.
- To check the query plan when using the DataFrame API, use DataFrame#explain().

Check Job History

You can use the following resources to view job history:

- Spark History Server UI: view information about Spark jobs that have completed.
  1. In the , go to .
  2. Open the .
  3. Find and select the cluster you want to configure.
  4. Click the Spark History Server link.
- YARN web UI: view job history and time spent in various stages of the job:
  1. In the , go to .
  2. Open the .
  3. Find and select the cluster you want to configure.
  4. Click the Resource Manager link.
  5. Select the Applications tab.
- yarn logs command: list the contents of all log files from all containers associated with the specified application:

```
yarn logs -applicationId <app_id>
```

Improving Software Performance

About this task

To improve Spark performance, assess and tune the following operations:

- Minimize shuffle operations where possible.

- Match join strategy (ShuffledHashJoin vs. BroadcastHashJoin) to the table.

  This requires manual configuration.
- Consider switching from the default serializer to the Kryo serializer to improve performance.

  This requires manual configuration and class registration.
- Adjust YARN memory allocation

Configure YARN Memory Allocation for Spark

This section describes how to manually configure YARN memory allocation settings based on node hardware specifications.

YARN evaluates all available compute resources on each machine in a cluster and negotiates resource requests from applications running in the cluster. YARN then provides processing capacity to each application by allocating containers. A container is the basic unit of processing capacity in YARN; it is an encapsulation of resource elements such as memory (RAM) and CPU.

In a Hadoop cluster, it is important to balance the use of RAM, CPU cores, and disks so that processing is not constrained by any one of these cluster resources.

When determining the appropriate YARN memory configurations for Spark, note the following values on each node:

- RAM (amount of memory)
- CORES (number of CPU cores)

When configuring YARN memory allocation for Spark, consider the following information:

- Driver memory does not need to be large if the job does not aggregate much data (as with a collect() action).
- There are tradeoffs between num-executors and executor-memory.

  Large executor memory does not imply better performance, due to JVM garbage collection. Sometimes it is better to configure a larger number of small JVMs than a small number of large JVMs.
- Executor processes are not released if the job has not finished, even if they are no longer in use.

  Therefore, do not overallocate executors above your estimated requirements.

In yarn-cluster mode, the Spark driver runs inside an application master process that is managed by YARN on the cluster. The client can stop after initiating the application. The following example shows starting a YARN client in yarn-cluster mode, specifying the number of executors and associated memory and core, and driver memory. The client starts the default Application Master, and SparkPi runs as a child thread of the Application Master. The client periodically polls the Application Master for status updates and displays them on the console.

```
./bin/spark-submit --class org.apache.spark.examples.SparkPi \
    --master yarn-cluster \
    --num-executors 3 \
    --driver-memory 4g \
    --executor-memory 2g \
    --executor-cores 1 \
    lib/spark-examples*.jar 10
```

In yarn-client mode, the driver runs in the client process. The application master is used only to request resources for YARN. To launch a Spark application in yarn-client mode, replace yarn-cluster with yarn-client. The following example launches the Spark shell in yarn-client mode and specifies the number of executors and associated memory:

```
./bin/spark-shell --num-executors 32 \
    --executor-memory 24g \
    --master yarn-client
```

Tuning Apache Spark Applications

This topic describes various aspects in tuning the performance and scalability of Apache Spark applications. For general Spark tuning advice, consult the upstream Spark documentation. This topic focuses on performance aspects that are especially relevant when using Spark in the context of clusters.

During tuning, monitor application behavior to determine the effect of tuning actions. You might see improvements that are directly relevant to the performance of your job, such as reduction in CPU usage, or reductions in resource usage that improve overall scalability within a multi-tenant cluster.

For information on monitoring Spark applications, see "Monitoring Spark Applications."

Tuning Spark Shuffle Operations

A Spark dataset comprises a fixed number of partitions, each of which comprises a number of records. For the datasets returned by *narrow* transformations, such as map and filter, the records required to compute the records in a single partition reside in a single partition in the parent dataset. Each object is only dependent on a single object in the parent. Operations such as coalesce can result in a task processing multiple input partitions, but the transformation is still considered narrow because the input records used to compute any single output record can still only reside in a limited subset of the partitions.

Spark also supports transformations with *wide* dependencies, such as groupByKey and reduceByKey. In these dependencies, the data required to compute the records in a single partition can reside in many partitions of the parent dataset. To perform these transformations, all of the tuples with the same key must end up in the same partition, processed by the same task. To satisfy this requirement, Spark performs a shuffle, which transfers data around the cluster and results in a new stage with a new set of partitions.

For example, consider the following code:

```
sc.textFile("someFile.txt").map(mapFunc).flatMap(flatMapFunc).filter(filterF
unc).count()
```

It runs a single action, count, which depends on a sequence of three transformations on a dataset derived from a text file. This code runs in a single stage, because none of the outputs of these three transformations depend on data that comes from different partitions than their inputs.

In contrast, this Scala code finds how many times each character appears in all the words that appear more than 1,000 times in a text file:

```
val tokenized = sc.textFile(args(0)).flatMap(_.split(' '))
val wordCounts = tokenized.map((_, 1)).reduceByKey(_ + _)
val filtered = wordCounts.filter(_._2 >= 1000)
val charCounts = filtered.flatMap(_._1.toCharArray).map((_, 1)).reduceByKey(
_ + _)
charCounts.collect()
```

This example has three stages. The two reduceByKey transformations each trigger stage boundaries, because computing their outputs requires repartitioning the data by keys.

A final example is this more complicated transformation graph, which includes a join transformation with multiple dependencies:

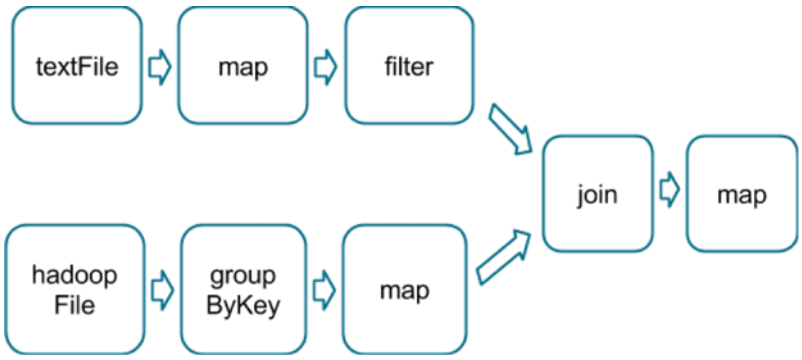

The pink boxes show the resulting stage graph used to run it:

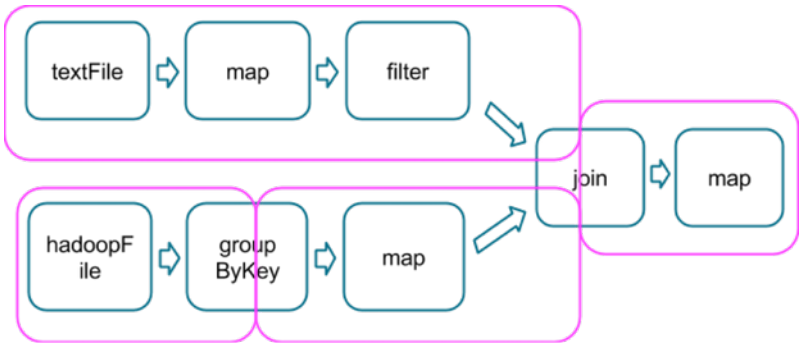

At each stage boundary, data is written to disk by tasks in the parent stages and then fetched over the network by tasks in the child stage. Because they incur high disk and network I/O, stage boundaries can be expensive and should be avoided when possible. The number of data partitions in a parent stage may be different than the number of partitions in a child stage. Transformations that can trigger a stage boundary typically accept a numPartitions argument, which specifies into how many partitions to split the data in the child stage. Just as the number of reducers is an important parameter in MapReduce jobs, the number of partitions at stage boundaries can determine an application's performance. "Tuning the Number of Partitions" describes how to tune this number.

Choosing Transformations to Minimize Shuffles

You can usually choose from many arrangements of actions and transformations that produce the same results. However, not all these arrangements result in the same performance. Avoiding common pitfalls and picking the right arrangement can significantly improve an application's performance.

When choosing an arrangement of transformations, minimize the number of shuffles and the amount of data shuffled. Shuffles are expensive operations; all shuffle data must be written to disk and then transferred over the network. repartition, join, cogroup, and any of the *By or *ByKey transformations can result in shuffles. Not all these transformations are equal, however, and you should avoid the following patterns:

- groupByKey when performing an associative reductive operation. For example, rdd.groupByKey().mapValues(_ .sum) produces the same result as rdd.reduceByKey(_ + _). However, the former transfers the entire dataset across the network, while the latter computes local sums for each key in each partition and combines those local sums into larger sums after shuffling.

- reduceByKey when the input and output value types are different. For example, consider writing a transformation that finds all the unique strings corresponding to each key. You could use map to transform each element into a Set and then combine the Sets with reduceByKey:

  ```
  rdd.map(kv => (kv._1, new Set[String]() + kv._2)).reduceByKey(_ ++ _)
  ```

  This results in unnecessary object creation because a new set must be allocated for each record.

  Instead, use aggregateByKey, which performs the map-side aggregation more efficiently:

  ```
  val zero = new collection.mutable.Set[String]()
  rdd.aggregateByKey(zero)((set, v) => set += v,(set1, set2) => set1 ++= s
  et2)
  ```

- flatMap-join-groupBy. When two datasets are already grouped by key and you want to join them and keep them grouped, use cogroup. This avoids the overhead associated with unpacking and repacking the groups.

When Shuffles Do Not Occur

In some circumstances, the transformations described previously do not result in shuffles. Spark does not shuffle when a previous transformation has already partitioned the data according to the same partitioner. Consider the following flow:

```
rdd1 = someRdd.reduceByKey(...)
rdd2 = someOtherRdd.reduceByKey(...)
rdd3 = rdd1.join(rdd2)
```

Because no partitioner is passed to reduceByKey, the default partitioner is used, resulting in rdd1 and rdd2 both being hash-partitioned. These two reduceByKey transformations result in two shuffles. If the datasets have the same number of partitions, a join requires no additional shuffling. Because the datasets are partitioned identically, the set of keys in any single partition of rdd1 can only occur in a single partition of rdd2. Therefore, the contents of any single output partition of rdd3 depends only on the contents of a single partition in rdd1 and single partition in rdd2, and a third shuffle is not required.

For example, if someRdd has four partitions, someOtherRdd has two partitions, and both the reduceByKeys use three partitions, the set of tasks that run would look like this:

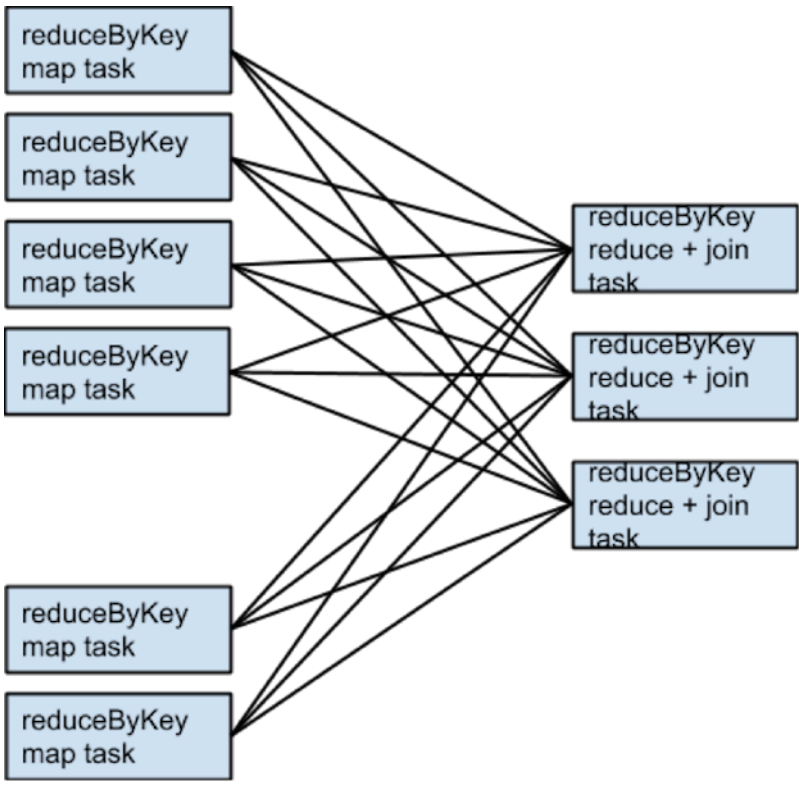

If rdd1 and rdd2 use different partitioners or use the default (hash) partitioner with different numbers of partitions, only one of the datasets (the one with the fewer number of partitions) needs to be reshuffled for the join:

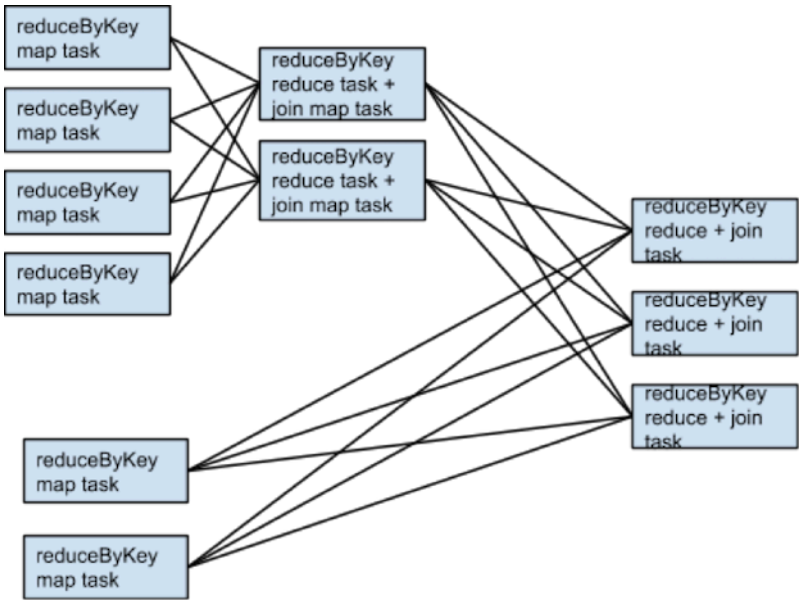

To avoid shuffles when joining two datasets, you can use broadcast variables. When one of the datasets is small enough to fit in memory in a single executor, it can be loaded into a hash table on the driver and then broadcast to every executor. A map transformation can then reference the hash table to do lookups.

When to Add a Shuffle Transformation

The rule of minimizing the number of shuffles has some exceptions.

An extra shuffle can be advantageous when it increases parallelism. For example, if your data arrives in a few large unsplittable files, the partitioning dictated by the InputFormat might place large numbers of records in each partition, while not generating enough partitions to use all available cores. In this case, invoking repartition with a high number of partitions (which triggers a shuffle) after loading the data allows the transformations that follow to use more of the cluster's CPU.

Another example arises when using the reduce or aggregate action to aggregate data into the driver. When aggregating over a high number of partitions, the computation can quickly become bottlenecked on a single thread in the driver merging all the results together. To lighten the load on the driver, first use reduceByKey or aggregat eByKey to perform a round of distributed aggregation that divides the dataset into a smaller number of partitions. The values in each partition are merged with each other in parallel, before being sent to the driver for a final round of aggregation. See treeReduce and treeAggregate for examples of how to do that.

This method is especially useful when the aggregation is already grouped by a key. For example, consider an application that counts the occurrences of each word in a corpus and pulls the results into the driver as a map. One approach, which can be accomplished with the aggregate action, is to compute a local map at each partition and then merge the maps at the driver. The alternative approach, which can be accomplished with aggregateByKey, is to perform the count in a fully distributed way, and then simply collectAsMap the results to the driver.

Secondary Sort

The repartitionAndSortWithinPartitions transformation repartitions the dataset according to a partitioner and, within each resulting partition, sorts records by their keys. This transformation pushes sorting down into the shuffle machinery, where large amounts of data can be spilled efficiently and sorting can be combined with other operations.

For example, Apache Hive on Spark uses this transformation inside its join implementation. It also acts as a vital building block in the secondary sort pattern, in which you group records by key and then, when iterating over the values that correspond to a key, have them appear in a particular order. This scenario occurs in algorithms that need to group events by user and then analyze the events for each user, based on the time they occurred.

Tuning Resource Allocation

For background information on how Spark applications use the YARN cluster manager, see "Running Spark Applications on YARN."

The two main resources that Spark and YARN manage are CPU and memory. Disk and network I/O affect Spark performance as well, but neither Spark nor YARN actively manage them.

Every Spark executor in an application has the same fixed number of cores and same fixed heap size. Specify the number of cores with the --executor-cores command-line flag, or by setting the spark.executor.cores property. Similarly, control the heap size with the --executor-memory flag or the spark.executor.memory property. The cores property controls the number of concurrent tasks an executor can run. For example, set --executor-cores 5 for each executor to run a maximum of five tasks at the same time. The memory property controls the amount of data Spark can cache, as well as the maximum sizes of the shuffle data structures used for grouping, aggregations, and joins.

Dynamic allocation, which adds and removes executors dynamically, is enabled by default. To explicitly control the number of executors, you can override dynamic allocation by setting the --num-executors command-line flag or spar k.executor.instances configuration property.

Consider also how the resources requested by Spark fit into resources YARN has available. The relevant YARN properties are:

- yarn.nodemanager.resource.memory-mb controls the maximum sum of memory used by the containers on each host.
- yarn.nodemanager.resource.cpu-vcores controls the maximum sum of cores used by the containers on each host.

Requesting five executor cores results in a request to YARN for five cores. The memory requested from YARN is more complex for two reasons:

- The --executor-memory/spark.executor.memory property controls the executor heap size, but executors can also use some memory off heap, for example, Java NIO direct buffers. The value of the spark.yarn.executor.memoryOverhead property is added to the executor memory to determine the full memory request to YARN for each executor. It defaults to max(384, .1 * spark.executor.memory).
- YARN may round the requested memory up slightly. The yarn.scheduler.minimum-allocation-mb and yarn.scheduler.increment-allocation-mb properties control the minimum and increment request values, respectively.

The following diagram (not to scale with defaults) shows the hierarchy of memory properties in Spark and YARN:

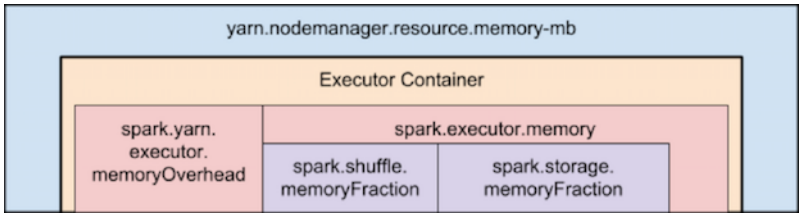

Keep the following in mind when sizing Spark executors:

- The ApplicationMaster, which is a non-executor container that can request containers from YARN, requires memory and CPU that must be accounted for. In *client* deployment mode, they default to 1024 MB and one core. In *cluster* deployment mode, the ApplicationMaster runs the driver, so consider bolstering its resources with the --driver-memory and --driver-cores flags.
- Running executors with too much memory often results in excessive garbage-collection delays. For a single executor, use 64 GB as an upper limit.
- The HDFS client has difficulty processing many concurrent threads. At most, five tasks per executor can achieve full write throughput, so keep the number of cores per executor below that number.
- Running tiny executors (with a single core and just enough memory needed to run a single task, for example) offsets the benefits of running multiple tasks in a single JVM. For example, broadcast variables must be replicated once on each executor, so many small executors results in many more copies of the data.

Resource Tuning Example

Consider a cluster with six hosts running NodeManagers, each equipped with 16 cores and 64 GB of memory.

The NodeManager capacities, yarn.nodemanager.resource.memory-mb and yarn.nodemanager.resource.cpu-vcores, should be set to 63 * 1024 = 64512 (megabytes) and 15, respectively. Avoid allocating 100% of the resources to YARN containers because the host needs some resources to run the OS and Hadoop daemons. In this case, leave one GB and one core for these system processes. Cloudera Manager accounts for these and configures these YARN properties automatically.

You might consider using --num-executors 6 --executor-cores 15 --executor-memory 63G. However, this approach does not work:

- 63 GB plus the executor memory overhead does not fit within the 63 GB capacity of the NodeManagers.
- The ApplicationMaster uses a core on one of the hosts, so there is no room for a 15-core executor on that host.
- 15 cores per executor can lead to bad HDFS I/O throughput.

Instead, use --num-executors 17 --executor-cores 5 --executor-memory 19G:

- This results in three executors on all hosts except for the one with the ApplicationMaster, which has two executors.
- --executor-memory is computed as (63/3 executors per host) = 21. 21 * 0.07 = 1.47. 21 - 1.47 ~ 19.

Tuning the Number of Partitions

Spark has limited capacity to determine optimal parallelism. Every Spark stage has a number of tasks, each of which processes data sequentially. The number of tasks per stage is the most important parameter in determining performance.

As described in "Spark Execution Model," Spark groups datasets into stages. The number of tasks in a stage is the same as the number of partitions in the last dataset in the stage. The number of partitions in a dataset is the same as the number of partitions in the datasets on which it depends, with the following exceptions:

- The coalesce transformation creates a dataset with fewer partitions than its parent dataset.
- The union transformation creates a dataset with the sum of its parents' number of partitions.
- The cartesian transformation creates a dataset with the product of its parents' number of partitions.

Datasets with no parents, such as those produced by textFile or hadoopFile, have their partitions determined by the underlying MapReduce InputFormat used. Typically, there is a partition for each HDFS block being read. The number of partitions for datasets produced by parallelize are specified in the method, or spark.default.parallelism if not specified. To determine the number of partitions in an dataset, call rdd.partitions().size().

If the number of tasks is smaller than number of slots available to run them, CPU usage is suboptimal. In addition, more memory is used by any aggregation operations that occur in each task. In join, cogroup, or *ByKey operations, objects are held in hashmaps or in-memory buffers to group or sort. join, cogroup, and groupByKey use these data structures in the tasks for the stages that are on the fetching side of the shuffles they trigger. reduceByKey and aggr egateByKey use data structures in the tasks for the stages on both sides of the shuffles they trigger. If the records in these aggregation operations exceed memory, the following issues can occur:

- Increased garbage collection, which can lead to pauses in computation.
- Spilling data to disk, causing disk I/O and sorting, which leads to job stalls.

To increase the number of partitions if the stage is reading from Hadoop:

- Use the repartition transformation, which triggers a shuffle.
- Configure your InputFormat to create more splits.
- Write the input data to HDFS with a smaller block size.

If the stage is receiving input from another stage, the transformation that triggered the stage boundary accepts a numP artitions argument:

```
val rdd2 = rdd1.reduceByKey(_ + _, numPartitions = X)
```

Determining the optimal value for *X* requires experimentation. Find the number of partitions in the parent dataset, and then multiply that by 1.5 until performance stops improving.

You can also calculate *X* using a formula, but some quantities in the formula are difficult to calculate. The main goal is to run enough tasks so that the data destined for each task fits in the memory available to that task. The memory available to each task is:

```
(spark.executor.memory * spark.shuffle.memoryFraction * spark.shuffle.safety
Fraction)/
spark.executor.cores
```

memoryFraction and safetyFraction default to 0.2 and 0.8 respectively.

The in-memory size of the total shuffle data is more difficult to determine. The closest heuristic is to find the ratio between shuffle spill memory and the shuffle spill disk for a stage that ran. Then, multiply the total shuffle write by this number. However, this can be compounded if the stage is performing a reduction:

```
(observed shuffle write) * (observed shuffle spill memory) * (spark.executor
.cores)/
(observed shuffle spill disk) * (spark.executor.memory) * (spark.shuffle.me
moryFraction) * (spark.shuffle.safetyFraction)
```

Then, round up slightly, because too many partitions is usually better than too few.

When in doubt, err on the side of a larger number of tasks (and thus partitions). This contrasts with recommendations for MapReduce, which unlike Spark, has a high startup overhead for tasks.

Reducing the Size of Data Structures

Data flows through Spark in the form of records. A record has two representations: a deserialized Java object representation and a serialized binary representation. In general, Spark uses the deserialized representation for records in memory and the serialized representation for records stored on disk or transferred over the network. For sort-based shuffles, in-memory shuffle data is stored in serialized form.

The spark.serializer property controls the serializer used to convert between these two representations. Cloudera recommends using the Kryo serializer, org.apache.spark.serializer.KryoSerializer.

The footprint of your records in these two representations has a significant impact on Spark performance. Review the data types that are passed and look for places to reduce their size. Large deserialized objects result in Spark spilling data to disk more often and reduces the number of deserialized records Spark can cache (for example, at the MEMO RY storage level). The Apache Spark tuning guide describes how to reduce the size of such objects. Large serialized objects result in greater disk and network I/O, as well as reduce the number of serialized records Spark can cache (for example, at the MEMORY_SER storage level.) Make sure to register any custom classes you use with the SparkCon f#registerKryoClasses API.

Choosing Data Formats

When storing data on disk, use an extensible binary format like Avro, Parquet, Thrift, or Protobuf and store in a sequence file.

Cloudera Runtime 7.3.1

Using Apache Hive

Date published: 2019-08-21
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Apache Hive 3 tables.....5

Hive table locations.....6
Refer to a table using dot notation.....6
Understanding CREATE TABLE behavior.....7
Creating a CRUD transactional table.....8
Creating an insert-only transactional table.....8
Creating an S3-based external table.....9
Dropping an external table along with data.....10
Converting a managed non-transactional table to external.....10
Accessing StorageHandler and other external tables.....11
Creating secure external tables.....12
Check for required Ranger features in Cloudera Data Hub.....14
Enable authorization of StorageHandler-based tables in Cloudera Data Hub.....15
Examples of creating secure external tables.....15
Using constraints.....17
Determining the table type.....19

Apache Hive 3 ACID transactions.....19

Apache Hive query basics.....23

Querying the information_schema database.....23
Inserting data into a table.....25
Updating data in a table.....25
Merging data in tables.....26
Deleting data from a table.....26
Using a subquery.....27
Subquery restrictions.....27
Use wildcards with SHOW DATABASES.....27
Aggregating and grouping data.....28
Querying correlated data.....28
Using common table expressions.....29
Use a CTE in a query.....29
Comparing tables using ANY/SOME/ALL.....29
Escaping an invalid identifier.....30
CHAR data type support.....31
ORC vs Parquet formats.....31

Creating a default directory for managed tables.....31

Configuring a table location outside the warehouse root directory.....32

Generating surrogate keys.....33

Partitions and performance.....34

Creating partitions dynamically.....35

Partition refresh and configuration....36
Automating partition discovery and repair.... 37
Managing partition retention time....38
Repairing partitions manually using MSCK repair.... 39

Query scheduling.... 40
Enabling scheduled queries.... 40
Periodically rebuilding a materialized view....41
Getting scheduled query information and monitor the query....42

Materialized views.... 43
Creating and using a materialized view....44
Creating the tables and view....44
Verifing use of a query rewrite....45
Using optimizations from a subquery.... 46
Dropping a materialized view.... 47
Showing materialized views....47
Describing a materialized view.... 48
Managing query rewrites....50
Purposely using a stale materialized view.... 50
Creating and using a partitioned materialized view....50

HPL/SQL stored procedures....53
Setting up a Hive client....54
Creating a function.... 55
Using the cursor to return record sets....56
Stored procedure examples....57

Using functions....58
Reloading, viewing, and filtering functions....58
Create a user-defined function.... 60
Setting up the development environment....60
Creating the UDF class.... 61
Building the project and upload the JAR.... 62
Registering the UDF....63
Calling the UDF in a query.... 64

Apache Hive 3 tables

Table type definitions and a diagram of the relationship of table types to ACID properties clarifies Hive tables. The location of a table depends on the table type. You might choose a table type based on its supported storage format.

You can create ACID (atomic, consistent, isolated, and durable) tables for unlimited transactions or for insert-only transactions. These tables are Hive managed tables. Alternatively, you can create an external table for non-transactional use. Because Hive control of the external table is weak, the table is not ACID compliant.

The following diagram depicts the Hive table types.

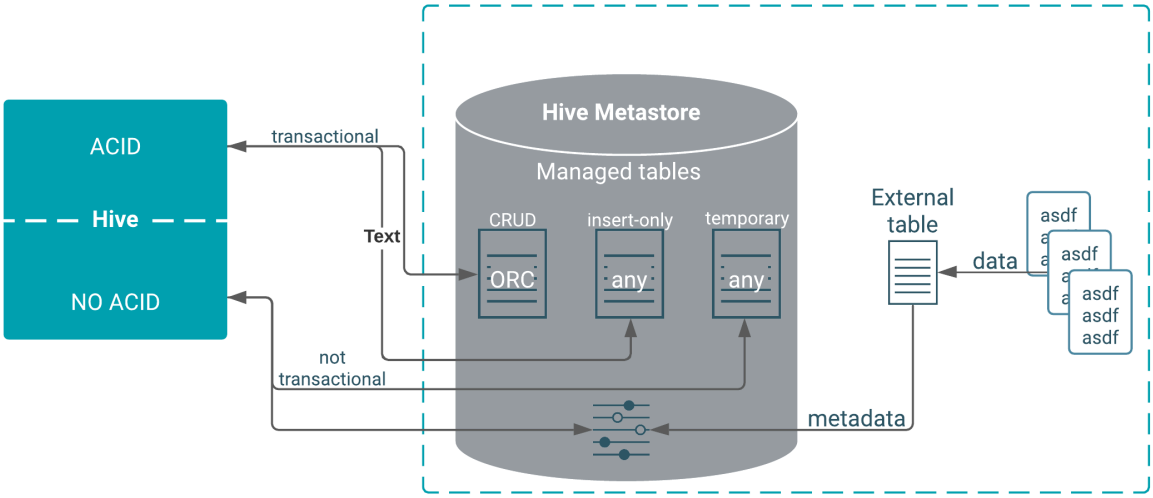

The following matrix includes the types of tables you can create using Hive, whether or not ACID properties are supported, required storage format, and key SQL operations.

Table Type	ACID	File Format	INSERT	UPDATE/DELETE
Managed: CRUD transactional	Yes	ORC	Yes	Yes
Managed: Insert-only transactional	Yes	Any	Yes	No
Managed: Temporary	No	Any	Yes	No
External	No	Any	Yes	No

Although you cannot use the SQL UPDATE or DELETE statements to delete data in some types of tables, you can use DROP PARTITION on any table type to delete the data.

Table storage formats

The data in CRUD tables must be in ORC format. Implementing a storage handler that supports AcidInputFormat and AcidOutputFormat is equivalent to specifying ORC storage.

Insert-only tables support all file formats.

The managed table storage type is Optimized Row Column (ORC) by default. If you accept the default by not specifying any storage during table creation, or if you specify ORC storage, you get an ACID table with insert, update, and delete (CRUD) capabilities. If you specify any other storage type, such as text, CSV, AVRO, or JSON, you get an insert-only ACID table. You cannot update or delete columns in the insert-only table.

Transactional tables

Transactional tables are ACID tables that reside in the Hive warehouse. To achieve ACID compliance, Hive has to manage the table, including access to the table data. Only through Hive can you access and change the data in managed tables. Because Hive has full control of managed tables, Hive can optimize these tables extensively.

Hive is designed to support a relatively low rate of transactions, as opposed to serving as an online analytical processing (OLAP) system. You can use the SHOW TRANSACTIONS command to list open and aborted transactions.

Transactional tables in Hive 3 are on a par with non-ACID tables. No bucketing or sorting is required in Hive 3 transactional tables. Bucketing does not affect performance. These tables are compatible with native cloud storage.

Hive supports one statement per transaction, which can include any number of rows, partitions, or tables.

External tables

External table data is not owned or controlled by Hive. You typically use an external table when you want to access data directly at the file level, using a tool other than Hive. You can also use a storage handler, such as Druid or HBase, to create a table that resides outside the Hive metastore.

Hive 3 does not support the following capabilities for external tables:

- Query cache
- Materialized views, except in a limited way
- Automatic runtime filtering
- File merging after insert
- ARCHIVE, UNARCHIVE, TRUNCATE, MERGE, and CONCATENATE. These statements only work for Hive Managed tables.

When you run DROP TABLE on an external table, by default Hive drops only the metadata (schema). If you want the DROP TABLE command to also remove the actual data in the external table, as DROP TABLE does on a managed table, you need to set the external.table.purge property to true as described later.

Hive table locations

You need a little information about the location of your Hive tables in Cloudera Public Cloud, which you can control. The location depends on the table type that you can determine by running a Hive query.

In Cloudera Public Cloud, you specify the location of managed tables and external table metadata in the Hive warehouse during Cloudera Data Warehouse setup. Hive metastore properties hive.metastore.warehouse.dir and hive .metastore.warehouse.external.dir set the storage locations for Hive tables. For example:

- hive.metastore.warehouse.external.dir = s3a://bucketName/warehouse/tablespace/external/hive
- hive.metastore.warehouse.dir=s3a://bucketName/warehouse/tablespace/managed/hive

In Cloudera Manager, when you launch your cluster, you accept default or specify other Hive metastore variables hive.metastore.warehouse.dir and hive.metastore.warehouse.external.dir that determine storage locations for Hive tables. Managed tables reside in the managed tablespace, which only Hive can access. By default, Hive assumes external tables reside in the external tablespace.

To determine the managed or external table type, you can run the DESCRIBE EXTENDED table_name command.

Refer to a table using dot notation

Hive 3.1 changes to table references using dot notation might require changes to your Hive scripts.

About this task

Hive 3.1 in Cloudera includes SQL compatibility (Hive-16907), which rejects `db.table` in SQL queries. The dot (.) is not allowed in table names. To reference the database and table in a table name, enclosed both in backticks as follows:

```
`db`.`table`
```

Understanding CREATE TABLE behavior

Hive table creation has changed significantly since Hive 3 to improve useability and functionality. You must understand the default behavior of the CREATE TABLE statement in .

Hive has changed table creation in the following ways:

- Creates ACID-compliant table, which is the default in
- Supports simple writes and inserts
- Writes to multiple partitions
- Inserts multiple data updates in a single SELECT statement
- Eliminates the need for bucketing.

If you have an ETL pipeline that creates tables in Hive, the tables will be created as ACID. Hive now tightly controls access and performs compaction periodically on the tables. Using ACID-compliant, transactional tables causes no performance or operational overload. The way you access managed Hive tables from Spark and other clients changes. In , access to external tables requires you to set up security access permissions.

In , by default, the CREATE TABLE statement creates a full ACID transactional table in ORC format or insert-only transactional tables for all other table formats. You can choose to modify the default behavior by configuring certain properties. The order of preference for configuration is as follows:

Modify the default CREATE TABLE behavior

Override default behavior when creating the table

Irrespective of the database, session, or site-level settings, you can override the default table behavior by using the MANAGED or EXTERNAL keyword in the CREATE TABLE statement.

```
CREATE [MANAGED][EXTERNAL] TABLE foo (id INT);
```

Set the default table type at a database level

You can use the database property, defaultTableType=EXTERNAL or ACID to specify the default table type to be created using the CREATE TABLE statement. You can specify this property when creating the database or at a later point using the ALTER DATABASE statement. For example:

```
CREATE DATABASE test_db WITH DBPROPERTIES ('defaultTableType'='E
XTERNAL');
```

In this example, tables created under the test_db database using the CREATE TABLE statement creates external tables with the purge fucntionality enabled (external.table.purge = 'true').

You can also choose to configure a database to allow only external tables to be created and prevent creation of ACID tables. While creating a database, you can set the database property, EXTER NAL_TABLES_ONLY=true to ensure that only external tables are created in the database. For example:

```
CREATE DATABASE test_db WITH DBPROPERTIES ('EXTERNAL_TABLES_ONLY
'='true');
```

Set the default table type at a session level

You can configure the CREATE TABLE behavior within an existing beeline session by setting hive.create.as.external.legacy to true or false. Setting the value to true results in configuring the CREATE TABLE statement to create external tables by default.

When the session ends, the default CREATE TABLE behavior also ends.

Set the default table type at a site level

You can configure the CREATE TABLE behavior at the site level by configuring the hive.create.as.insert.only and hive.create.as.acid properties in under Hive configuration. When configured at the site level, the behavior persists from session to session.

If you are a Spark user, switching to legacy behavior is unnecessary. Calling ‘create table’ from SparkSQL, for example, creates an external table after upgrading to as it did before the upgrade. You can connect to Hive using the Hive Warehouse Connector (HWC) to read Hive ACID tables from Spark. To write ACID tables to Hive from Spark, you use the HWC and HWC API. Spark creates an external table with the purge property when you do not use the HWC API. For more information, see Hive Warehouse Connector for accessing Spark data.

Creating a CRUD transactional table

You create a CRUD transactional table having ACID (atomic, consistent, isolated, and durable) properties when you need a managed table that you can update, delete, and merge. You learn by example how to determine the table type.

About this task

In this task, you create a CRUD transactional table. You cannot sort this type of table. To create a CRUD transactional table, you must accept the default ORC format by not specifying any storage during table creation, or by specifying ORC storage explicitly.

Procedure

1. Start Hive.
2. Enter your user name and password.
   The Hive 3 connection message, followed by the Hive prompt for entering SQL queries on the command line, appears.
3. Create a CRUD transactional table named T having two integer columns, a and b:

   ```
   CREATE TABLE T(a int, b int);
   ```

4. Confirm that you created a managed, ACID table.

   ```
   DESCRIBE FORMATTED T;
   ```

   The table type says MANAGED_TABLE and transactional = true.

Related Information

HMS storage

Creating an insert-only transactional table

You can create a transactional table using any storage format if you do not require update and delete capability. This type of table has ACID properties, is a managed table, and accepts insert operations only. The storage format of an insert-only table is not restricted to ORC.

About this task

In this task, you create an insert-only transactional table for storing text. In the CREATE TABLE statement, specifying a storage type other than ORC, such as text, CSV, AVRO, or JSON, results in an insert-only ACID table. You can explicitly specify insert-only in the table properties clause.

Procedure

1. Start Hive.
2. Enter your user name and password.
   The Hive 3 connection message, followed by the Hive prompt for entering SQL queries on the command line, appears.
3. Create a insert-only transactional table named T2 having two integer columns, a and b:

   ```
   CREATE TABLE T2(a int, b int)
     STORED AS ORC
     TBLPROPERTIES ('transactional'='true',
     'transactional_properties'='insert_only');
   ```

   The 'transactional_properties'='insert_only' is required; otherwise, a CRUD table results. The STORED AS ORC clause is optional (default = ORC).
4. Create an insert-only transactional table for text data.

   ```
   CREATE TABLE T3(a int, b int)
   STORED AS TEXTFILE;
   ```

   The 'transactional_properties'='insert_only' is not required because the storage format is other than ORC.

Related Information

HMS storage

Creating an S3-based external table

You use the LOCATION clause in the CREATE EXTERNAL TABLE statement to create an external data having source data on S3.

About this task

In this task, you create a partitioned, external table and load data from the source on S3. You can use the LOCATION clause in the CREATE TABLE to specify the location of external table data. The metadata is stored in the Hive warehouse.

Before you begin

- Set up Hive policies in Ranger to include S3 URLs.

Procedure

1. Put data source files on S3.
2. Create an external table based on the data source files.

   ```
   CREATE EXTERNAL TABLE `inventory`(
     `inv_item_sk` int,
     `inv_warehouse_sk` int,
     `inv_quantity_on_hand` int)
     PARTITIONED BY (
     `inv_date_sk` int) STORED AS ORC
   ```

```
  LOCATION
  's3a://BUCKET_NAME/tpcds_bin_partitioned_orc_200.db/inventory';
```

Dropping an external table along with data

When you run DROP TABLE on an external table, by default Hive drops only the metadata (schema). If you want the DROP TABLE command to also remove the actual data in the external table, as DROP TABLE does on a managed table, you need to configure the table properties accordingly.

Procedure

1. Create a CSV file of data you want to query in Hive.
2. Start Hive.
3. Create an external table to store the CSV data, configuring the table so you can drop it along with the data.

```
CREATE EXTERNAL TABLE IF NOT EXISTS names_text(
  a INT, b STRING)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  STORED AS TEXTFILE
  LOCATION '<file system>://andrena'
  TBLPROPERTIES ('external.table.purge'='true');
```

4. Run DROP TABLE on the external table.

```
DROP TABLE names_text;
```

   The table is removed from Hive Metastore and the data stored externally. For example, names_text is removed from the Hive Metastore and the CSV file that stored the data is also deleted from the file system.

5. Prevent data in external table from being deleted by a DROP TABLE statement.

```
ALTER TABLE addresses_text SET TBLPROPERTIES ('external.table.purge'='fa
lse');
```

Converting a managed non-transactional table to external

You can easily convert a managed table, if it is not an ACID (transactional) table, to external using the ALTER TABLE statement. You might have a non-ACID, managed table after an upgrade from Hive 1 or 2.

About this task

The following pseudo-code changes a managed table, if it is not transactional, to external. The data and metadata is dropped when the table is dropped.

```
ALTER TABLE ... SET TBLPROPERTIES('EXTERNAL'='TRUE','external.table.purge='
true')
```

Related Information

Before and After Upgrading Table Type Comparison

Accessing StorageHandler and other external tables

Before creating secure external tables based on a StorageHandler, you must configure Hive impersonation. You learn which permissions Hive checks before you attempt to create a secure external table. You understand the policies necessary for accessing HBase from Hive.

When you create a managed versus an external table, including external tables based on a StorageHandler, Hive checks permissions described in the following table:

Table 1:

Table Type	Example	Permissions Checked
Managed	Create table foo(i int);	Does the user have write and execute permission to the table storage location?
External	Create table foo_ext(i int) stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' with serdeproperties ("hbase.columns.mapping"="cf:string", "hbase.table.name"="hbase_table_0”);	Does the user have write and execute permission to the table storage location and does the user have read access to the external table, hbase_table_0, for example?

In Hive 3, you follow recommendations to turn off Hive impersonation (hive.server2.enable.doAs property = false). As shown in the following diagram, any Hive user who can create a table, can also read the data of any external table.

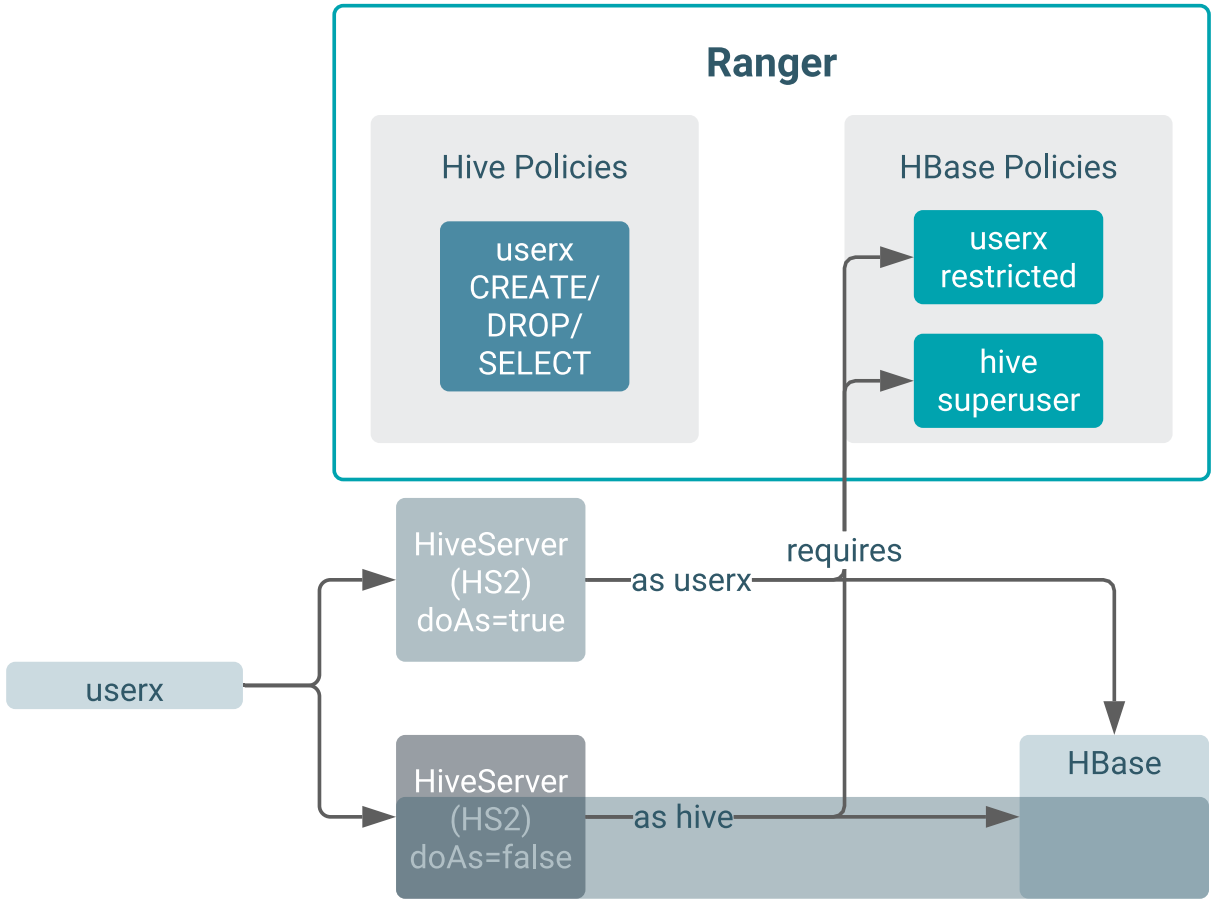

When doas=true, HBase policies restrict access to HBase. When doAs=true, HBase recognizes the end-user who logged into Hive. To access the HBase service, you need to create Ranger policies for end user access, instead of just user hive access to Hive.

When doas=false, any Hive user with CREATE/DROP/SELECT table access in Hive can read, write, or delete any HBase table using the HBaseStorageHandler. When doas=false, HBase see suser hive is accessing the HBase tables. You need to create one policy in Ranger for HBase that allows user hive to read data from any table in HBase. From Hive, any user can read data from any table in HBase.

As a Hive 3 user, you must set doas=false to use Ranger. Ranger is the supported authorization model in Cloudera. You must set up Ranger to secure external tables, such as the HBaseStorageHandler table, as described in the next topic.

Related Information

Configure a resource-based storage handler policy: HadoopSQL

Creating secure external tables

A step-by-step procedure shows you how to create a secure external table using SERDEPROPERTIES or TBLPROPERTIES and Ranger policies.

About this task

You provide SERDEPROPERTIES or TBLPROPERTIES when you create the external table. Hive uses this information to authorize access to the table based on Ranger policies you set up. Continuing with the HBaseStorageHandler example from the previous topic, this procedure shows how to set up Ranger policies for the following table:

```
Create table foo_ext(i int) stored by 'org.apache.hadoop.hive.hbase.HBaseSto
rageHandler'
with serdeproperties ("hbase.columns.mapping"="cf:string", "hbase.table.na
me"="hbase_table_0”);
```

In this case, the URI sent for ranger Authentication is hbase://hostclustername:2181/hbase_table_0/cf. In this procedure, you create a Hadoop SQL policy in Ranger that provides the following authorizations:

- Authorizes the end user against the location of the external table.

  For example, in Ranger create an HDFS policy to give read, write, and execute permissions to hdfs://user/warehouse/tablespace/external/hive/foo.
- Authorizes the end user against the URI used in the table creation statement.

  For example, in Ranger set a SQL policy to give create/alter/drop privileges on hbase://hostname:portnumber/hbase_table_0.
- Authorizes the user hive to access the HBase table.

  For example, in Ranger set an HBase policy to have full access to hbase://hostname/hbase_table_0.

By default, the capability to create StorageHandler-based tables is turned off. Required Ranger features changes might not be available in your environment.

Before you begin

- Check that you have required Ranger features as described in the next topic.
- Enable authorization of StorageHandler based tables as described in the subsequent topic.

Procedure

1. Go to Ranger Service Manager Hadoop SQL Policies , enter a policy name, and click the numerical link for all - storage-type, storage-url.
   For example, click 11 in the screenshot below.

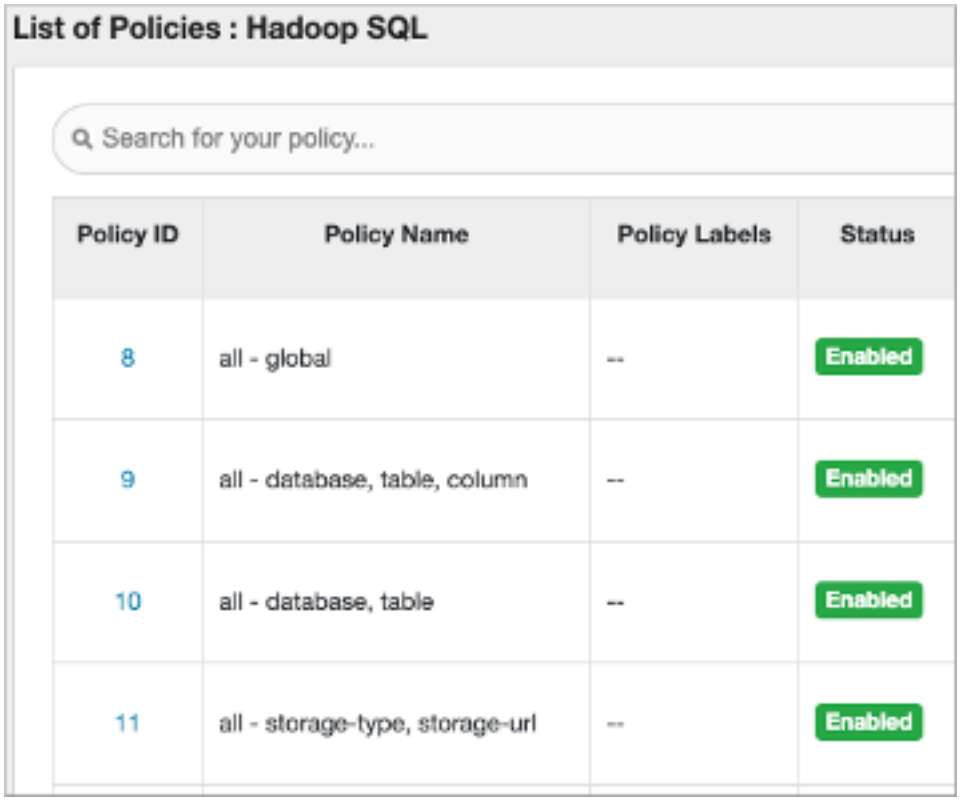

2. In Edit Policy, select storage-type, type hbase, and then select hbase to set the storage type to hbase.
3. In Storage URL, type the URI format for the table.
   For example, type hbase-cluster:port/hbase-table.

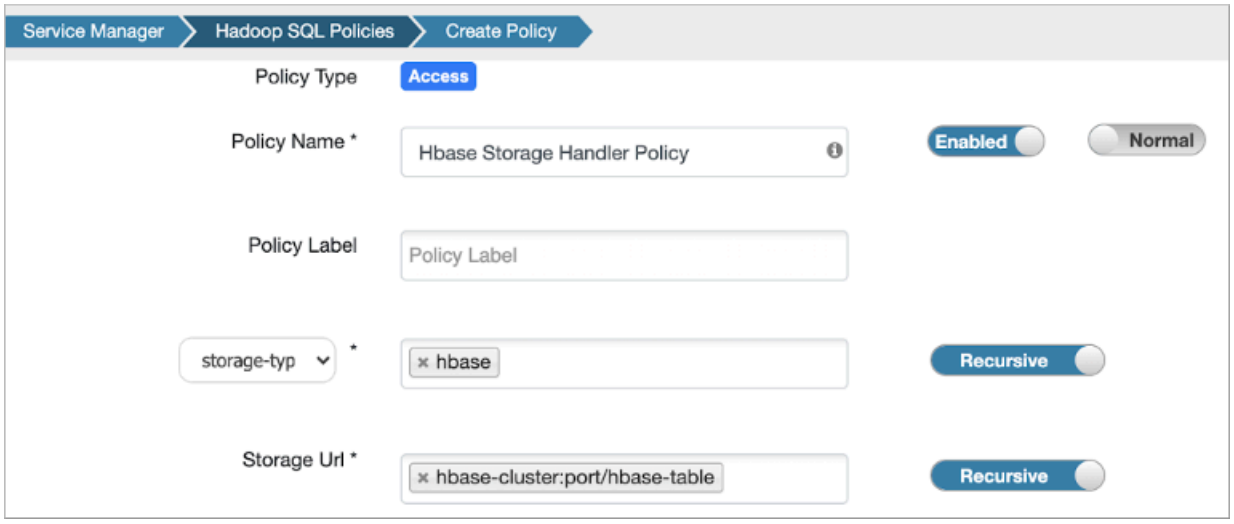

4. Add permissions for users or groups to create a table on the table storage location.
   For example, select Create Table on Storage.

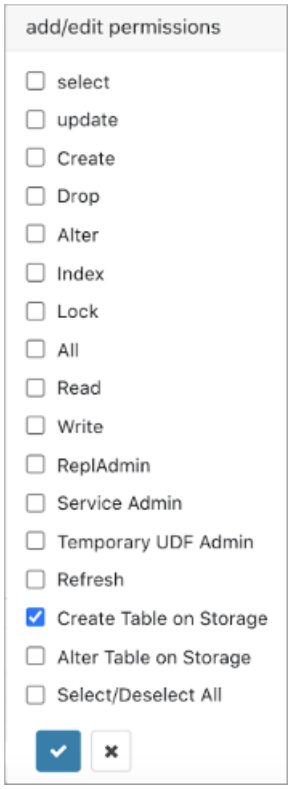

Related Information

Configure a resource-based storage handler policy: HadoopSQL

Check for required Ranger features in Cloudera Data Hub

Your version of Cloudera Data Hub might not have the features necessary to create secure external tables. Learn how to quickly check for the required Ranger feature.

Procedure

1. In Cloudera Manager, click Clusters Ranger Admin web UI .
2. Enter your user name and password, then click Sign In.
3. Select Hadoop SQL.

4. In Hadoop SQL Policies, look for the all - storage-type, storage-url policy.

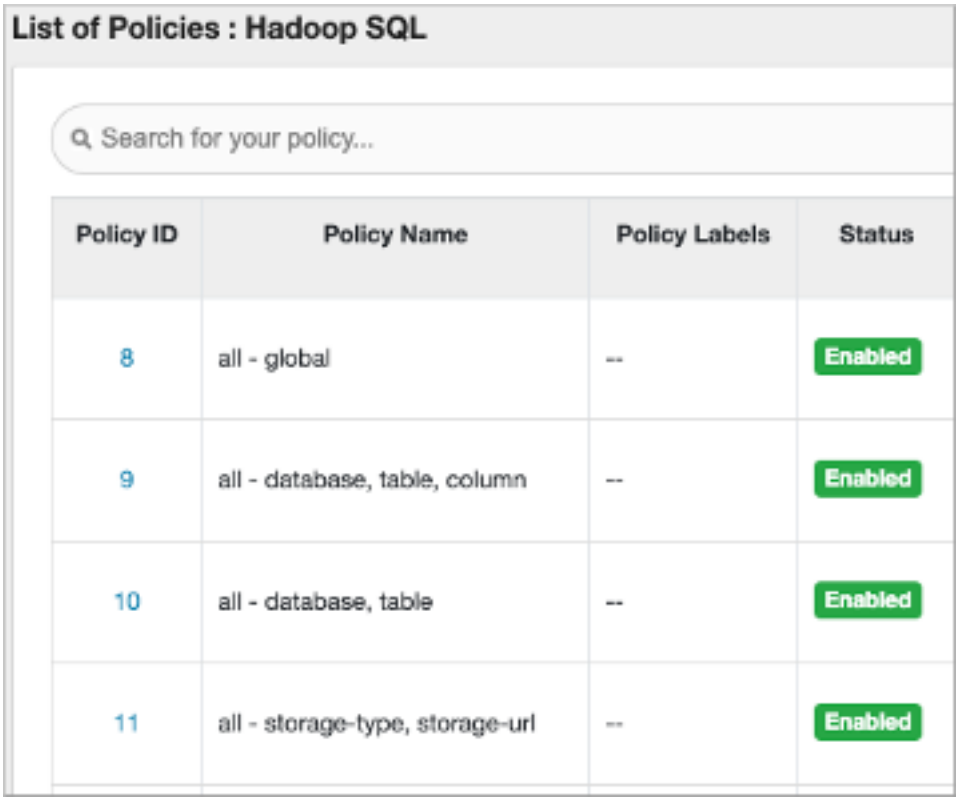

Enable authorization of StorageHandler-based tables in Cloudera Data Hub

You need to know how to enable authorization of StorageHandler-based tables. You follow steps to add a custom property.

About this task

In Cloudera Data Hub, you configure hive.security.authorization.tables.on.storagehandlers = true to enable authorization of StorageHandler-based tables.

Procedure

1. In Cloudera Manager, click Clusters Hive Configurations , and search for hive.security.authorization.tables.on.storagehandlers.
2. Set the value to true.
3. Save changes.

Examples of creating secure external tables

You learn by example how to set up policies to secure HBase-, Kafka- and JDBC-based tables using the Ranger policy properties.

Kafka-based table

```
CREATE EXTERNAL TABLE kafka_table_test (`timestamp` timestamp , `page` strin
g,  `newPage` boolean, added int, deleted bigint, delta double)
STORED BY 'org.apache.hadoop.hive.kafka.KafkaStorageHandler'
TBLPROPERTIES ("kafka.topic" = "test-topic","kafka.bootstrap.servers" = "
hostclustername:2181");
```

In this case, the URI for ranger Authentication is kafka://hostclustername:2181/test-topic/column_names.

The policy definition sets the kafka storage-type and storage Url format bootstrap-server:port/kafka-topic.

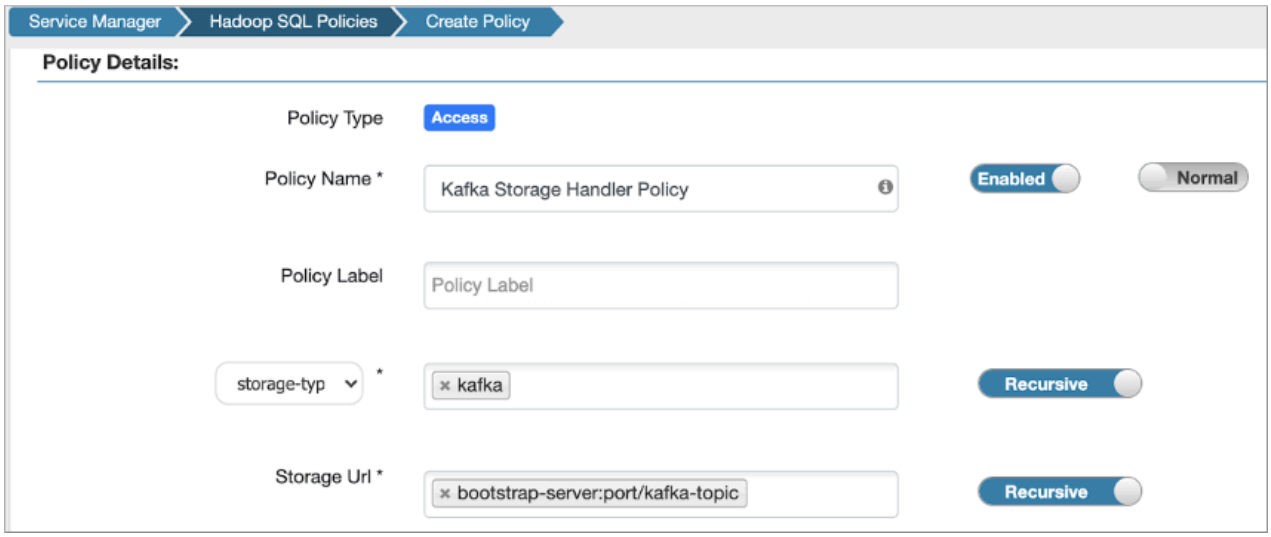

JDBC-based table

```
CREATE EXTERNAL TABLE mytable_jdbc(col1 string, col2 int, col3 double)
STORED BY 'org.apache.hive.storage.jdbc.JdbcStorageHandler'
TBLPROPERTIES ("hive.sql.database.type" = "MYSQL", "hive.sql.jdbc.driver" =
 "com.mysql.jdbc.Driver", "hive.sql.jdbc.url" = "jdbc:mysql://localhost/samp
le", "hive.sql.dbcp.username" = "hive", "hive.sql.dbcp.password" = "hive", "
hive.sql.table" = "MYTABLE");
```

In the query, you need to specify either hive.sql.table or hive.sql.query to tell the storagehandler how to get data from the JDBC database. In this case, the URI for Ranger Authentication is jdbc:mysql://localhost/sample/MYTABLE.

The policy definition sets the com.mysql.jdbc.Driver storage-type and the storage Url format to jdbc:mysql://mysql-host:port/table-name.

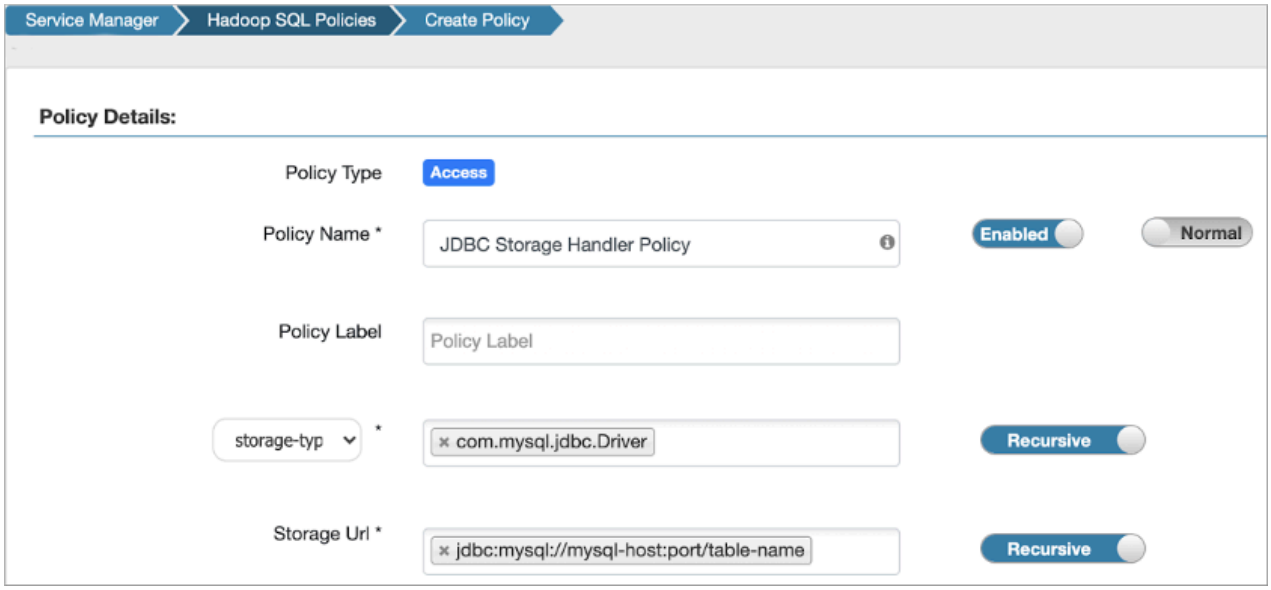

When you create the table, you use the hive.sql.table table property to denote a single table name if your policy defined the Storage Url format using a table name, as shown in the screenshot above.

Alternatively, you can use the wildcard asterisk in the Storage Url format when you define the policy, for example jdbc:mysql://mysql-host:port/* as shown below. In the query, you then use the hive.sql.query table property to denote an arbitrary SQL query:

Related Information

Configure a resource-based storage handler policy: HadoopSQL

Using constraints

You can use SQL constraints to enforce data integrity and improve performance. Using constraints, the optimizer can simplify queries. Constraints can make data predictable and easy to locate. Using constraints and supported modifiers, you can follow examples to constrain queries to unique or not null values, for example.

Hive enforces DEFAULT, NOT NULL and CHECK only, not PRIMARY KEY, FOREIGN KEY, and UNIQUE.

You can use the constraints listed below in your queries. Hive enforces DEFAULT, NOT NULL and CHECK only, not PRIMARY KEY, FOREIGN KEY, and UNIQUE. DEFAULT even if enforced, does not support complex types (array,map,struct). Constraint enforcement is limited to the metadata level. This limitation aids integration with third party tools and optimization of constraints declarations, such as materialized view rewriting.

CHECK

Limits the range of values you can place in a column.

DEFAULT

Ensures a value exists, which is useful in offloading data from a data warehouse.

PRIMARY KEY

Identifies each row in a table using a unique identifier.

FOREIGN KEY

Identifies a row in another table using a unique identifier.

UNIQUE KEY

Checks that values stored in a column are different.

NOT NULL

Ensures that a column cannot be set to NULL.

Supported modifiers

You can use the following optional modifiers:

ENABLE

Ensures that all incoming data conforms to the constraint.

DISABLE

Does not ensure that all incoming data conforms to the constraint.

VALIDATE

Checks that all existing data in the table conforms to the constraint.

NOVALIDATE

Does not check that all existing data in the table conforms to the constraint.

ENFORCED

Maps to ENABLE NOVALIDATE.

NOT ENFORCED

Maps to DISABLE NOVALIDATE.

RELY

Specifies abiding by a constraint; used by the optimizer to apply further optimizations.

NORELY

Specifies not abiding by a constraint.

You use modifiers as shown in the following syntax:

```
( ( ( (ENABLE | DISABLE) (VALIDATE | NOVALIDATE) ) | ( ENFORCED | NOT ENFORC
ED ) ) (RELY | NORELY) )
```

Note: For external tables, the RELY constraint is the only supported constraint.

Default modfiers

The following default modifiers are in place:

- The default modifier for ENABLE is NOVALIDATE RELY.
- The default modifier for DISABLE is NOVALIDATE NORELY.
- If you do not specify a modifier when you declare a constraint, the default is ENABLE NOVALIDATE RELY. The following constraints do not support ENABLE:
  - PRIMARY KEY
  - FOREIGN KEY
  - UNIQUE KEY

  To prevent an error, specify a modfier when using these constraints to override the default.

Constraints examples

The optimizer uses the constraint information to make smart decisions. The following examples show the use of constraints.

The following example shows how to create a table that declares the NOT NULL in-line constraint to constrain a column.

```
CREATE TABLE t(a TINYINT, b SMALLINT NOT NULL ENABLE, c INT);
```

The constrained column b accepts a SMALLINT value as shown in the first INSERT statement.

```
INSERT INTO t values(2,45,5667);
    ...
    ----------------------------------------------------------
    1 row affected ...
```

The constrained column b will not accept a NULL value.

```
INSERT INTO t values(2,NULL,5667);
```

```
    Error: Error running query: org.apache.hadoop.hive.ql.exec.errors.Dat
aConstraintViolationError: /
    Either CHECK or NOT NULL constraint violated! (state=,code=0)
```

The following examples shows how to declare the FOREIGN KEY constraint out of line. You can specify a constraint name, in this case fk, in an out-of-line constraint

```
CREATE TABLE Persons (
     ID INT NOT NULL,
     Name STRING NOT NULL,
     Age INT,
     Creator STRING DEFAULT CURRENT_USER(),
     CreateDate DATE DEFAULT CURRENT_DATE(),
     PRIMARY KEY (ID) DISABLE NOVALIDATE);

     CREATE TABLE BusinessUnit (
     ID INT NOT NULL,
     Head INT NOT NULL,
     Creator STRING DEFAULT CURRENT_USER(),
     CreateDate DATE DEFAULT CURRENT_DATE(),
     PRIMARY KEY (ID) DISABLE NOVALIDATE,
     CONSTRAINT fk FOREIGN KEY (Head) REFERENCES Persons(ID) DISABLE NOVA
LIDATE
     );
```

Determining the table type

You can determine the type of a Hive table, whether it has ACID properties, the storage format, such as ORC, and other information. Knowing the table type is important for a number of reasons, such as understanding how to store data in the table or to completely remove data from the cluster.

Procedure

1. In the Hive shell, get an extended description of the table.
   For example: DESCRIBE EXTENDED mydatabase.mytable;
2. Scroll to the bottom of the command output to see the table type.
   The following output says the table type is managed. transaction=true indicates that the table has ACID properties:

   ```
   ...
   | Detailed Table Information  | Table(tableName:t2, dbName:mydatabase, o
   wner:hdfs, createTime:1538152187, lastAccessTime:0, retention:0, sd:Stor
   ageDescriptor(cols:[FieldSchema(name:a, type:int, comment:null), FieldSc
   hema(name:b, type:int, comment:null)], ...
   ```

Related Information

HMS storage

Apache Hive 3 ACID transactions

Hive 3 achieves atomicity and isolation of operations on transactional tables by using techniques in write, read, insert, create, delete, and update operations that involve delta files. You can obtain query status information from these files and use the files to troubleshoot query problems.

Write and read operations

Hive 3 write and read operations improve the ACID qualities and performance of transactional tables. Transactional tables perform as well as other tables. Hive supports all TPC Benchmark DS (TPC-DS) queries.

Hive 3 and later extends atomic operations from simple writes and inserts to support the following operations:

- Writing to multiple partitions
- Using multiple insert clauses in a single SELECT statement

A single statement can write to multiple partitions or multiple tables. If the operation fails, partial writes or inserts are not visible to users. Operations remain fast even if data changes often, such as one percent per hour. Hive 3 and later does not overwrite the entire partition to perform update or delete operations.

Hive compacts ACID transaction files automatically without impacting concurrent queries. Automatic compaction improves query performance and the metadata footprint when you query many small, partitioned files.

Read semantics consist of snapshot isolation. Hive logically locks in the state of the warehouse when a read operation starts. A read operation is not affected by changes that occur during the operation.

Atomicity and isolation in insert-only tables

When an insert-only transaction begins, the transaction manager gets a transaction ID. For every write, the transaction manager allocates a write ID. This ID determines a path to which data is actually written. The following code shows an example of a statement that creates insert-only transactional table:

```
CREATE TABLE tm (a int, b int) TBLPROPERTIES
('transactional'='true', 'transactional_properties'='insert_only')
```

Assume that three insert operations occur, and the second one fails:

```
INSERT INTO tm VALUES(1,1);
INSERT INTO tm VALUES(2,2); // Fails
INSERT INTO tm VALUES(3,3);
```

For every write operation, Hive creates a delta directory to which the transaction manager writes data files. Hive writes all data to delta files, designated by write IDs, and mapped to a transaction ID that represents an atomic operation. If a failure occurs, the transaction is marked aborted, but it is atomic:

```
tm
___ delta_0000001_0000001_0000
### 000000_0
___ delta_0000002_0000002_0000       //Fails
### 000000_0
___ delta_0000003_0000003_0000
### 000000_0
```

During the read process, the transaction manager maintains the state of every transaction. When the reader starts, it asks for the snapshot information, represented by a high watermark. The watermark identifies the highest transaction ID in the system followed by a list of exceptions that represent transactions that are still running or are aborted.

The reader looks at deltas and filters out, or skips, any IDs of transactions that are aborted or still running. The reader uses this technique with any number of partitions or tables that participate in the transaction to achieve atomicity and isolation of operations on transactional tables.

Atomicity and isolation in CRUD tables

You create a full CRUD (create, retrieve, update, delete) transactional table using the following SQL statement:

```
CREATE TABLE acidtbl (a INT, b STRING);
```

Running SHOW CREATE TABLE acidtbl provides information about the defaults: transactional (ACID) and the ORC data storage format:

```
+----------------------------------------------------+
|                   createtab_stmt                   |
+----------------------------------------------------+
| CREATE TABLE `acidtbl`(                            |
|   `a` int,                                         |
|   `b` string)                                      |
| ROW FORMAT SERDE                                   |
|   'org.apache.hadoop.hive.ql.io.orc.OrcSerde'      |
| STORED AS INPUTFORMAT                              |
|   'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'  |
| OUTPUTFORMAT                                       |
|   'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat' |
| LOCATION                                           |
|   's3://myserver.com:8020/warehouse/tablespace/managed/hive/acidtb
l' |
| TBLPROPERTIES (                                    |
|   'bucketing_version'='2',                         |
|   'transactional'='true',                          |
|   'transactional_properties'='default',            |
|   'transient_lastDdlTime'='1555090610')            |
+----------------------------------------------------+
```

Tables that support updates and deletions require a slightly different technique to achieve atomicity and isolation. Hive runs in append-only mode, which means Hive does not perform in-place updates or deletions. Isolation of readers and writers cannot occur in the presence of in-place updates or deletions. In this situation, a lock manager or some other mechanism, is required for isolation. These mechanisms create a problem for long-running queries.

Instead of in-place updates, Hive decorates every row with a row ID. The row ID is a struct that consists of the following information:

- The write ID that maps to the transaction that created the row
- The bucket ID, a bit-backed integer with several bits of information, of the physical writer that created the row
- The row ID, which numbers rows as they were written to a data file

Metadata Columns	**original_write_id** **bucket_id** **row_id** current_write_id	**ROW__ID**
User Columns	col_1: a : INT col_2: b : STRING	

Instead of in-place deletions, Hive appends changes to the table when a deletion occurs. The deleted data becomes unavailable and the compaction process takes care of the garbage collection later.

Create operation

The following example inserts several rows of data into a full CRUD transactional table, creates a delta file, and adds row IDs to a data file.

```
INSERT INTO acidtbl (a,b) VALUES (100, "oranges"), (200, "apples"), (300, "b
ananas");
```

This operation generates a directory and file, delta_00001_00001/bucket_0000, that have the following data:

ROW_ID	a	b
{1,0,0}	100	"oranges"
{1,0.1}	200	"apples"
{1,0,2}	300	"bananas"

Delete operation

A delete statement that matches a single row also creates a delta file, called the delete-delta. The file stores a set of row IDs for the rows that match your query. At read time, the reader looks at this information. When it finds a delete event that matches a row, it skips the row and that row is not included in the operator pipeline. The following example deletes data from a transactional table:

```
DELETE FROM acidTbl where a = 200;
```

This operation generates a directory and file, delete_delta_00002_00002/bucket_0000 that have the following data:

ROW_ID	a	b
{1,0,1}	null	null

Update operation

An update combines the deletion and insertion of new data. The following example updates a transactional table:

```
UPDATE acidTbl SET b = "pears" where a = 300;
```

One delta file contains the delete event, and the other, the insert event:

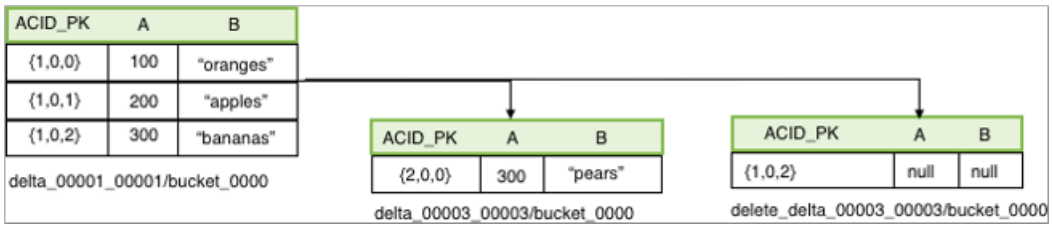

The reader, which requires the AcidInputFormat, applies all the insert events and encapsulates all the logic to handle delete events. A read operation first gets snapshot information from the transaction manager based on which it selects files that are relevant to that read operation. Next, the process splits each data file into the number of pieces that each process has to work on. Relevant delete events are localized to each processing task. Delete events are stored in a sorted ORC file. The compressed, stored data is minimal, which is a significant advantage of Hive 3. You no longer need to worry about saturating the network with insert events in delta files.

Apache Hive query basics

Using Apache Hive, you can query distributed data storage. You need to know ANSI SQL to view, maintain, or analyze Hive data. Examples of the basics, such as how to insert, update, and delete data from a table, helps you get started with Hive.

Hive supports ANSI SQL and atomic, consistent, isolated, and durable (ACID) transactions. For updating data, you can use the MERGE statement, which meets ACID standards. Materialized views optimize queries based on access patterns. Hive supports tables up to 300PB in Optimized Row Columnar (ORC) format. Other file formats are also supported. You can create tables that resemble those in a traditional relational database. You use familiar insert, update, delete, and merge SQL statements to query table data. The insert statement writes data to tables. Update and delete statements modify and delete values already written to Hive. The merge statement streamlines updates, deletes, and changes data capture operations by drawing on co-existing tables. These statements support auto-commit that treats each statement as a separate transaction and commits it after the SQL statement is executed.

Related Information

ORC Language Manual on the Apache wiki

Querying the information_schema database

Hive supports the ANSI-standard information_schema database, which you can query for information about tables, views, columns, and your Hive privileges. The information_schema data reveals the state of the system, similar to sys database data, but in a user-friendly, read-only way. You can use joins, aggregates, filters, and projections in information_schema queries.

About this task

One of the following steps involves changing the time interval for synchronization between HiveServer and the policy. HiveServer responds to any policy changes within this time interval. You can query the information_schema database for only your own privilege information.

Procedure

1. Open Ranger Access Manager, and check that the preloaded default database tables columns and information_schema database policies are enabled for group public.

Policy ID	Policy Name	Policy Labels	Status	Audit Logging	Roles	Groups	Users
7	all - global	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
8	all - database, table, column	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
9	all - database, table	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
10	all - database	--	Enabled	Enabled	--	public	hive beacon dpprofiler hue + More..
11	all - hiveservice	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
12	all - database, udf	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
13	all - url	--	Enabled	Enabled	--	--	hive beacon dpprofiler hue + More..
14	default database tables columns	--	Enabled	Enabled	--	public	--
15	Information_schema database ...	--	Enabled	Enabled	--	public	--

The information schema database is synchronized every half hour by default.

2. Start Hive, and check for the information_schema database:

```
SHOW DATABASES;
...
+----------------------+
|    database_name     |
+----------------------+
| default              |
| information_schema   |
| sys                  |
+----------------------+
```

3. Use the information_schema database to list tables in the database.

```
USE information_schema;
...
SHOW TABLES;
...
+--------------------+
|      tab_name      |
+--------------------+
| column_privileges  |
| columns            |
| schemata           |
| table_privileges   |
| tables             |
| views              |
+--------------------+
```

4. Query the information_schema database to see, for example, information about tables into which you can insert values.

```
SELECT * FROM information_schema.tables WHERE is_insertable_into='YES' l
imit 2;
...
+--------------------+-------------------+-----------------
|tables.table_catalog|tables.table_schema|tables.table_name
+--------------------+-------------------+-----------------
|default             |default            |students2
|default             |default            |t3
```

Inserting data into a table

To insert data into a table you use a familiar ANSI SQL statement. A simple example shows you have to accomplish this basic task.

About this task

To insert data into an ACID table, use the Optimized Row Columnar (ORC) storage format. To insert data into a non-ACID table, you can use other supported formats. You can specify partitioning as shown in the following syntax:

INSERT INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)] VALUES values_row [, values_row...]

where

values_row is (value [, value]) .

A value can be NULL or any SQL literal.

Procedure

1. Create an ACID table to contain student information.
   CREATE TABLE students (name VARCHAR(64), age INT, gpa DECIMAL(3,2));
2. Insert name, age, and gpa values for a few students into the table.
   INSERT INTO TABLE students VALUES ('fred flintstone', 35, 1.28), ('barney rubble', 32, 2.32);
3. Create a table called pageviews and assign null values to columns you do not want to assign a value.

```
CREATE TABLE pageviews (userid VARCHAR(64), link STRING, origin STRING)
PARTITIONED BY (datestamp STRING) CLUSTERED BY (userid) INTO 256 BUCKETS;
INSERT INTO TABLE pageviews PARTITION (datestamp = '2014-09-23') VALUES ('
jsmith', 'mail.com', 'sports.com'), ('jdoe', 'mail.com', null);
INSERT INTO TABLE pageviews PARTITION (datestamp) VALUES ('tjohnson', '
sports.com', 'finance.com', '2014-09-23'), ('tlee', 'finance.com', null,
 '2014-09-21');
```

   The ACID data resides in the warehouse.

Updating data in a table

The syntax describes the UPDATE statement you use to modify data already stored in a table. An example shows how to apply the syntax.

About this task

You construct an UPDATE statement using the following syntax:

UPDATE tablename SET column = value [, column = value ...] [WHERE expression];

Depending on the condition specified in the optional WHERE clause, an UPDATE statement might affect every row in a table. The expression in the WHERE clause must be an expression supported by a SELECT clause. Subqueries are not allowed on the right side of the SET statement. Partition columns cannot be updated.

Before you begin

You must have SELECT and UPDATE privileges to use the UPDATE statement.

Procedure

Create a statement that changes the values in the name column of all rows where the gpa column has the value of 1.0.
UPDATE students SET name = null WHERE gpa <= 1.0;

Merging data in tables

A sample statement shows how you can conditionally insert existing data in Hive tables using the ACID MERGE statement. Additional merge operations are mentioned.

About this task

The MERGE statement is based on ANSI-standard SQL.

Procedure

1. Construct a query to update the customers' names and states in customer target table to match the names and states of customers having the same IDs in the new_customer_stage source table.
2. Enhance the query to insert data from new_customer_stage table into the customer table if none already exists.

   Update or delete data using MERGE in a similar manner.

```
MERGE INTO customer USING (SELECT * FROM new_customer_stage) AS sub ON s
ub.id = customer.id
WHEN MATCHED THEN UPDATE SET name = sub.name, state = sub.state
WHEN NOT MATCHED THEN INSERT VALUES (sub.id, sub.name, sub.state);
```

Note: You can map specific columns in the INSERT clause of the query instead of passing values (including null) for columns in the target table that do not have any data to insert. The unspecified columns in the INSERT clause are either mapped to null or use default constraints, if any.

For example, you can construct the INSERT clause as WHEN NOT MATCHED THEN INSERT VAL UES (customer.id=sub.id, customer.name=sub.name, customer.state=sub.state) instead of WHEN NOT MATCHED THEN INSERT VALUES (sub.id, sub.name, 'null', sub.state).

Related Information

Merge documentation on the Apache wiki

Deleting data from a table

You use the DELETE statement to delete data already written to an ACID table.

About this task

Use the following syntax to delete data from a Hive table. DELETE FROM tablename [WHERE expression];

Procedure

Delete any rows of data from the students table if the gpa column has a value of 1 or 0.
DELETE FROM students WHERE gpa <= 1,0;

Using a subquery

Hive supports subqueries in FROM clauses and WHERE clauses that you can use for many Apache Hive operations, such as filtering data from one table based on contents of another table.

About this task

A subquery is a SQL expression in an inner query that returns a result set to the outer query. From the result set, the outer query is evaluated. The outer query is the main query that contains the inner subquery. A subquery in a WHERE clause includes a query predicate and predicate operator. A predicate is a condition that evaluates to a Boolean value. The predicate in a subquery must also contain a predicate operator. The predicate operator specifies the relationship tested in a predicate query.

Procedure

Select all the state and net_payments values from the transfer_payments table if the value of the year column in the table matches a year in the us_census table.

```
SELECT state, net_payments
FROM transfer_payments
WHERE transfer_payments.year IN (SELECT year FROM us_census);
```

The predicate starts with the first WHERE keyword. The predicate operator is the IN keyword.

The predicate returns true for a row in the transfer_payments table if the year value in at least one row of the us_census table matches a year value in the transfer_payments table.

Subquery restrictions

To construct queries efficiently, you must understand the restrictions of subqueries in WHERE clauses.

- Subqueries must appear on the right side of an expression.
- Nested subqueries are not supported.
- Subquery predicates must appear as top-level conjuncts.
- Subqueries support four logical operators in query predicates: IN, NOT IN, EXISTS, and NOT EXISTS.
- The IN and NOT IN logical operators may select only one column in a WHERE clause subquery.
- The EXISTS and NOT EXISTS operators must have at least one correlated predicate.
- The left side of a subquery must qualify all references to table columns.
- References to columns in the parent query are allowed only in the WHERE clause of the subquery.
- Subquery predicates that reference a column in a parent query must use the equals (=) predicate operator.
- Subquery predicates may refer only to columns in the parent query.
- Correlated subqueries with an implied GROUP BY statement may return only one row.
- All unqualified references to columns in a subquery must resolve to tables in the subquery.
- Correlated subqueries cannot contain windowing clauses.

Use wildcards with SHOW DATABASES

In a SHOW DATABASES LIKE statement, you can use wildcards and specify any character or a single character.

About this task

SHOW DATABASES or SHOW SCHEMAS lists all of the databases defined in Hive metastore. You can use the following wildcards:

%

Matches any single character or multiple characters.

_ (underscore)

Matches any single character.

|

Matches either the part of the pattern on the left or the right side of the pipe.

For example, 'students', 'stu%', 'stu_ents' match the database named students.

Aggregating and grouping data

You use AVG, SUM, or MAX functions to aggregate data, and the GROUP BY clause to group data query results in one or more table columns..

About this task

The GROUP BY clause explicitly groups data. Hive supports implicit grouping, which occurs when aggregating the table in full.

Procedure

1. Construct a query that returns the average salary of all employees in the engineering department grouped by year.

```
SELECT year, AVG(salary)
FROM Employees
WHERE Department = 'engineering' GROUP BY year;
```

2. Construct an implicit grouping query to get the highest paid employee.

```
SELECT MAX(salary) as highest_pay,
AVG(salary) as average_pay
FROM Employees
WHERE Department = 'engineering';
```

Querying correlated data

You can query one table relative to the data in another table.

About this task

A correlated query contains a query predicate with the equals (=) operator. One side of the operator must reference at least one column from the parent query and the other side must reference at least one column from the subquery. An uncorrelated query does not reference any columns in the parent query.

Procedure

Select all state and net_payments values from the transfer_payments table for years during which the value of the state column in the transfer_payments table matches the value of the state column in the us_census table.

```
SELECT state, net_payments
FROM transfer_payments
WHERE EXISTS
   (SELECT year
   FROM us_census
   WHERE transfer_payments.state = us_census.state);
```

This query is correlated because one side of the equals predicate operator in the subquery references the state column in the transfer_payments table in the parent query and the other side of the operator references the state column in the us_census table.

This statement includes a conjunct in the WHERE clause.

A conjunct is equivalent to the AND condition, while a disjunct is the equivalent of the OR condition The following subquery contains a conjunct:

... WHERE transfer_payments.year = "2018" AND us_census.state = "california"

The following subquery contains a disjunct:

... WHERE transfer_payments.year = "2018" OR us_census.state = "california"

Using common table expressions

Using common table expression (CTE), you can create a temporary view that repeatedly references a subquery.

A CTE is a set of query results obtained from a simple query specified within a WITH clause that immediately precedes a SELECT or INSERT keyword. A CTE exists only within the scope of a single SQL statement and not stored in the metastore. You can include one or more CTEs in the following SQL statements:

- SELECT
- INSERT
- CREATE TABLE AS SELECT
- CREATE VIEW AS SELECT

Recursive queries are not supported and the WITH clause is not supported within subquery blocks.

Use a CTE in a query

You can use a common table expression (CTE) to simplify creating a view or table, selecting data, or inserting data.

Procedure

1. Use a CTE to create a table based on another table that you select using the CREATE TABLE AS SELECT (CTAS) clause.

```
CREATE TABLE s2 AS WITH q1 AS (SELECT key FROM src WHERE key = '4') SELECT
 * FROM q1;
```

2. Use a CTE to create a view.

```
CREATE VIEW v1 AS WITH q1 AS (SELECT key FROM src WHERE key='5') SELECT *
 from q1;
```

3. Use a CTE to select data.

```
WITH q1 AS (SELECT key from src where key = '5')
  SELECT * from q1;
```

4. Use a CTE to insert data.

```
CREATE TABLE s1 LIKE src;
WITH q1 AS (SELECT key, value FROM src WHERE key = '5') FROM q1 INSERT OV
ERWRITE TABLE s1 SELECT *;
```

Comparing tables using ANY/SOME/ALL

You learn how to use quantified comparison predicates (ANY/SOME/ALL) in non-correlated subqueries according to the SQL standard. SOME is any alias for ANY.

About this task

You can use one of the following operators with a comparison predicate:

- >
- <
- >=
- <=
- <>
- =

ALL:

- If the table is empty, or the comparison is true for every row in subquery table, the predicate is true for that predicand.
- If the comparison is false for at least one row, the predicate is false.

SOME or ANY:

- If the comparison is true for at least one row in the subquery table, the predicate is true for that predicand.
- If the table is empty or the comparison is false for each row in subquery table, the predicate is false.

If the comparison is neither true nor false, the result is undefined.

For example, you run the following query to match any value in c2 of tbl equal to any value in c1 from the same tbl:

```
select c1 from tbl where c1 = ANY (select c2 from tbl);
```

You run the following query to match all values in c1 of tbl not equal to any value in c2 from the same tbl.

```
select c1 from tbl where c1 <> ALL (select c2 from tbl);
```

Escaping an invalid identifier

When you need to use reserved words, special characters, or a space in a column or partition name, enclose it in backticks (`).

About this task

An identifier in SQL is a sequence of alphanumeric and underscore (_) characters enclosed in backtick characters. In Hive, these identifiers are called quoted identifiers and are case-insensitive. You can use the identifier instead of a column or table partition name.

Before you begin

You have set the following parameter to column in the hive-site.xml file to enable quoted identifiers:

Set the hive.support.quoted.identifiers configuration parameter to column in the hive-site.xml file to enable quoted identifiers in column names. Valid values are none and column. For example, in Hive execute the following command: SET hive.support.quoted.identifiers = column.

Procedure

1. Create a table named test that has two columns of strings specified by quoted identifiers:
   CREATE TABLE test (`x+y` String, `a?b` String);
2. Create a table that defines a partition using a quoted identifier and a region number:
   CREATE TABLE partition_date-1 (key string, value string) PARTITIONED BY (`dt+x` date, region int);
3. Create a table that defines clustering using a quoted identifier:
   CREATE TABLE bucket_test(`key?1` string, value string) CLUSTERED BY (`key?1`) into 5 buckets;

CHAR data type support

Knowing how Hive supports the CHAR data type compared to other databases is critical during migration.

Table 2: Trailing Whitespace Characters on Various Databases

Data Type	Hive	Oracle	SQL Server	MySQL	Teradata
CHAR	Ignore	Ignore	Ignore	Ignore	Ignore
VARCHAR	Compare	Compare	Configurable	Ignore	Ignore
STRING	Compare	N/A	N/A	N/A	N/A

ORC vs Parquet formats

The differences between Optimized Row Columnar (ORC) file format for storing data in SQL engines are important to understand. Query performance improves when you use the appropriate format for your application.

ORC and Parquet capabilities comparison

The following table compares SQL engine support for ORC and Parquet.

Table 3:

Capability	Cloudera Data Warehouse	ORC	Parquet	SQL Engine
Read non-transactional data	Apache Hive	#	#	Hive
Read non-transactional data	Apache Impala	#	#	Impala
Read/Write Full ACID tables	Apache Hive	#		Hive
Read Full ACID tables	Apache Impala	#		Impala
Read Insert-only managed tables	Apache Impala	#	#	Impala
Column index	Apache Hive	#	#	Hive
Column index	Apache Impala		#	Impala
CBO uses column metadata	Apache Hive	#		Hive
Recommended format	Apache Hive	#		Hive
Recommended format	Apache Impala		#	Impala
Vectorized reader	Apache Hive	#	#	Hive
Read complex types	Apache Impala	#	#	Impala
Read/write complex types	Apache Hive	#	#	Hive

Creating a default directory for managed tables

You can specify a top-level directory for managed tables when creating a Hive database.

About this task

Create a default directory for managed tables only after limiting CREATE DATABASE and ALTER DATABASE statements to users having the Admin role, which has hive service user permissions. Permissions to the managed directory must be limited to the hive service user. In addition to restricting permissions to the hive user, you can further secure managed tables using Ranger fine-grained permissions, such as row-level filtering and column masking.

As Admin, you specify a managed location within the default location specified by the hive.metastore.warehouse.dir configuration property to give managed tables a common location for governance policies. The managed location designates a single root directory for all tenant tables, managed and external.

Setting the metastore.warehouse.tenant.colocation property to true allows a common location for managed tables outside the warehouse root directory, providing a tenant-based common root for setting quotas and other policies. To set this property, in Cloudera Manager use the Hive on Tez safety valve for hive-site.xml as described in the next topic.

Use the following syntax to create a database that specifies a location for managed tables:

```
CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name
    [COMMENT database_comment]
    [LOCATION external_table_path]
    [MANAGEDLOCATION managed_table_directory_path]
    [WITH DBPROPERTIES (property_name=property_value, ...)];
```

Do not set LOCATION and MANAGEDLOCATION to the same file system path.

Use the following syntax to set or change a location for managed tables.

```
ALTER (DATABASE|SCHEMA) database_name SET MANAGEDLOCATION [managed_table_dir
ectory_path];
```

Procedure

1. Create a database mydatabase that specifies a top level directory named sales for managed tables.

   ```
   CREATE DATABASE mydatabase MANAGEDLOCATION '/warehouse/tablespace/managed/
   hive/sales';
   ```

2. Change the abc_sales database location to the same location as mydatabase.

   ```
   ALTER DATABASE abc_sales SET MANAGEDLOCATION '/warehouse/tablespace/mana
   ged/hive/sales';
   ```

Configuring a table location outside the warehouse root directory

You need to know how to set the property that allows you to establish a common location for tables.

Procedure

1. In Cloudera Manager, click Clusters Hive on Tez Configuration .
2. In scope, click Hive on Tez (Service-Wide).
3. In Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml, click +.
4. In Name, enter metastore.warehouse.tenant.colocation.

5. In Value, enter true.

6. Save changes.
7. In Cloudera Manager Home, restart Hive on Tez.

Generating surrogate keys

You can use the built-in SURROGATE_KEY user-defined function (UDF) to automatically generate numerical Ids for rows as you enter data into a table. The generated surrogate keys can replace wide, multiple composite keys.

Before you begin

Hive supports the surrogate keys on ACID tables only, as described in the following matrix of table types:

Table Type	ACID	Surrogate Keys	File Format	INSERT	UPDATE/DELETE
Managed: CRUD transactional	Yes	Yes	ORC	Yes	Yes
Managed: Insert-only transactional	Yes	Yes	Any	Yes	No
Managed: Temporary	No	No	Any	Yes	No
External	No	No	Any	Yes	No

The table you want to join using surrogate keys cannot have column types that need casting. These data types must be primitives, such as INT or STRING.

About this task

Joins using the generated keys are faster than joins using strings. Using generated keys does not force data into a single node by a row number. You can generate keys as abstractions of natural keys. Surrogate keys have an advantage over UUIDs, which are slower and probabilistic.

The SURROGATE_KEY UDF generates a unique Id for every row that you insert into a table. It generates keys based on the execution environment in a distributed system, which includes a number of factors, such as internal data structures, the state of a table, and the last transaction id. Surrogate key generation does not require any coordination between compute tasks.

The UDF takes either no arguments or two arguments:

- Write Id bits
- Task Id bits

Procedure

1. Create a students table in the default ORC format that has ACID properties.

```
CREATE TABLE students (row_id INT, name VARCHAR(64), dorm INT);
```

2. Insert data into the table. For example:

```
INSERT INTO TABLE students VALUES (1, 'fred flintstone', 100), (2, 'barney
 rubble', 200);
```

3. Create a version of the students table using the SURROGATE_KEY UDF.

```
CREATE TABLE students_v2
(`ID` BIGINT DEFAULT SURROGATE_KEY(),
 row_id INT,
 name VARCHAR(64),
 dorm INT,
 PRIMARY KEY (ID) DISABLE NOVALIDATE);
```

4. Insert data, which automatically generates surrogate keys for the primary keys.

```
INSERT INTO students_v2 (row_id, name, dorm) SELECT * FROM students;
```

5. Take a look at the surrogate keys.

```
SELECT * FROM students_v2;
```

```
+-----------------+--------------------+------------------+-----------
--------+
| students_v2.id  | students_v2.row_id  | students_v2.name  | students_v2.
dorm  |
+-----------------+--------------------+------------------+-----------
--------+
| 1099511627776   | 1                  | fred flintstone   | 100
      |
| 1099511627777   | 2                  | barney rubble     | 200
      |
+-----------------+--------------------+------------------+-------------
------+
```

6. Add the surrogate keys as a foreign key to another table, such as a student_grades table, to speed up subsequent joins of the tables.

```
ALTER TABLE student_grades ADD COLUMNS (gen_id BIGINT);

MERGE INTO student_grades g USING students_v2 s ON g.row_id = s.row_id
WHEN MATCHED THEN UPDATE SET gen_id = s.id;
```

Now you can achieve fast joins on the surrogate keys.

Partitions and performance

A brief description of partitions and the performance benefits includes characters you must avoid when creating a partition. Examples of creating a partition and inserting data in a partition introduce basic partition syntax. Best practices for partitioning are mentioned.

A table you create without partitioning puts the data in a single directory. Partitioning divides the data into multiple directories. Queries of one or more columns based on the directories can run faster. Lengthy full table scans are avoided. Only data in the relevant directory is scanned. For example, a school_records table partitioned on a year column, segregates values by year into separate directories. A WHERE condiition such as YEAR=2020, YEAR IN (2020,2019), or YEAR BETWEEN 2001 AND 2010 scans only the data in the appropriate directory to resolve the query. Using partitions typically improves query performance.

In a SQL query, you define the partition as shown in the following example:

```
CREATE TABLE sale(id in, amount decimal) PARTITIONED BY (xdate string, state
 string);
```

To insert data into this table, you specify the partition key for fast loading:

```
INSERT INTO sale (xdate='2016-03-08', state='CA') SELECT * FROM staging_tabl
e WHERE xdate='2016-03-08' AND state='CA';
```

You do not need to specify dynamic partition columns. Hive generates a partition specification if you enable dynamic partitions.

Examples of a query on partitioned data

```
INSERT INTO sale (xdate, state)
SELECT * FROM staging_table;
```

Follow these best practices when you partition tables and query partitioned tables:

- Never partition on a unique ID.
- Size partitions to greater than or equal to 1 GB on average.
- Design queries to process not more than 1000 partitions.

Invalid Characters in a Partition Name

When you create a partition, do not use the following characters in a partition name:

- colon
- question mark
- percent

If you use these characters in a partition name, your directories will be named using the URL encoding of these characters, as described in "Why some special characters should not be used in a partition name in Hive/Impala."

Related Information

Why some special characters should not be used in a partition name in Hive/Impala

Creating partitions dynamically

You can configure Hive to create partitions dynamically and then run a query that creates the related directories on the file system or object store. Hive then separates the data into the directories.

About this task

This example assumes you have the following CSV file named employees.csv to use as the data source:

```
1,jane doe,engineer,service
2,john smith,sales rep,sales
3,naoko murai,service rep,service
4,somporn thong,ceo,sales
5,xi singh,cfo,finance
```

Procedure

1. Upload the CSV file to a file system.
2. In the Hive shell, create an unpartitioned table that holds all the data.

```
CREATE EXTERNAL TABLE employees (eid int, name string, position string,
dept string)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  STORED AS TEXTFILE
```

```
  LOCATION '<object store or file system path>';
```

3. Check that the data loaded into the employees table.

```
SELECT * FROM employees;
```

The output, formatted to fit this publication, appears:

```
+------+---------------+-------------+-------+---------+
| eid  |    name       | position    | dept  |         |
+------+---------------+-------------+-------+---------|
| 1    | jane doe      | engineer    | service         |
| 2    | john smith    | sales rep   | sales           |
| 3    | naoko murai   | service rep | service         |
| 4    | somporn thong | ceo         | sales           |
| 5    | xi singh      | cfo         | finance         |
+------+---------------+-------------+-----------------+
```

4. Create a partition table.

```
CREATE EXTERNAL TABLE EMP_PART (eid int, name string, position string)
  PARTITIONED BY (dept string);
```

5. Accept the default dynamic partition mode (nonstrict) to create partitioned directories of data dynamically when data is inserted, or if you changed the default, reset the mode as follows:

```
SET hive.exec.dynamic.partition.mode=nonstrict;
```

6. Insert data from the unpartitioned table (all the data) into the partitioned table , dynamically creating the partitions.

```
INSERT INTO TABLE EMP_PART PARTITION (DEPT)
  SELECT eid,name,position,dept FROM employees;
```

Partitions are created dynamically.

7. Check that the partitions were created.

```
SHOW PARTITIONS emp_part;
```

```
+----------------+
|   partition    |
+----------------+
| dept=finance   |
| dept=sales     |
| dept=service   |
+----------------+
```

Partition refresh and configuration

You can discover partition changes and synchronize Hive metadata automatically. Performing synchronization automatically, instead of manually, can save substantial time, especially when partitioned data, such as logs, changes frequently. You can also configure how long to retain partition data and metadata.

About this task

After creating a partitioned table, Hive does not update metadata about corresponding objects or directories on the file system that you add or drop. The partition metadata in the Hive metastore becomes stale after corresponding objects/directories are added or deleted. You need to synchronize the metastore and the file system.

You can refresh Hive metastore partition information manually or automatically. The time it takes to refresh the partition information is proportional to the number of partitions involved.

- Manually

  You run the MSCK (metastore consistency check) Hive command: MSCK REPAIR TABLE <table_name> ADD/DROP/SYNC PARTITIONS every time you need to synchronize a partition with the file system.

  This command triggers a full repair which might take a longer time if the number of partitions are more. Instead, you can use operators in the command, such as EQUAL, NOTEQUAL, LESSTHANOREQUALTO, LESSTHAN, GREATERTHANOREQUALTO, GREATERTHAN, or LIKE in the partition column so that a larger subset of partitions can be recovered (added/removed) without triggering a full repair.

  Syntax: MSCK REPAIR TABLE <table_name> ADD/DROP/SYNC PARTITIONS (<pcol1><operator><value>,<pcol2><operator><value>);
- Automatically

  You set up partition discovery to occur periodically.

The discover.partitions table property is automatically created and enabled for external partitioned tables. When discover.partitions is enabled for a table, Hive performs an automatic refresh as follows:

- Adds corresponding partitions that are in the file system, but not in the metastore, to the metastore.
- Removes partition schema information from metastore if you removed the corresponding partitions from the file system.

Partition retention

You can configure how long to keep partition metadata and data, and remove it after the retention period elapses.

Limitations

Generally, partition discovery and retention is not recommended for use on managed tables. The Hive metastore acquires an exclusive lock on a table that enables partition discovery that can slow down other queries.

Related Information

Creating partitions dynamically

Apache Wiki: Discover Partitions and Partition Retention

Automating partition discovery and repair

Automated partition discovery and repair is useful for processing log data, and other data, in Spark and Hive catalogs. You learn how to set the partition discovery parameter to suit your use case. An aggressive partition discovery and repair configuration can delay the upgrade process.

About this task

Apache Hive can automatically and periodically discover discrepancies in partition metadata in the Hive metastore and in corresponding directories, or objects, on the file system. After discovering discrepancies, Hive performs synchronization.

The discover.partitions table property enables and disables synchronization of the file system with partitions. In external partitioned tables, this property is disabled (false) by default when you create the table. To a legacy external table (created using a version of Hive that does not support this feature), you need to add discover.partitions to the table properties to enable partition discovery.

By default, discovery and synchronization of partitions occur every 5 minutes. If you are upgrading, this frequency causes performance degradation. Adjust the frequency of these batch routines to hourly or daily, depending on your use case.

Procedure

1. For external partitioned tables and for legacy external tables that are created using a version of Hive that does not support partition discovery, enable partition discovery for the table.

```
ALTER TABLE exttbl SET TBLPROPERTIES ('discover.partitions' = 'true');
```

2. In Cloudera Manager, click Clusters Hive Configuration , search for Hive Server Advanced Configuration Snipp et (Safety Valve) for hive-site.xml.
3. Add the following property and value to hive-site.xml: Property: metastore.partition.management.task.frequency Value: 600.

   The default value of metastore.partition.management.task.frequencyis 300. Changing this to 600 sets synchronization of partitions to occur every 10 minutes expressed in seconds. If you are upgrading to a cloud environment (AWS, AZURE, GCP), consider running discovery and synchonization once every 24 hours by setting the value to 86,400 seconds. Decreasing the value of metastore.partition.management.task.frequency incur higher cloud costs.

   Important: In a multi-Hive Metastore setup the Hive metastore instances will elect a single leader among themselves in the same cluster. There is no need to override the metastore.housekeeping.threads.on on the Hive Metastore instance level. This ensures that partition management (and other housekeeping activities) are done by one instance minimizing locking issues and enhancing performance. For more information, see *Hive Metastore leader election*.

   Disable the metastore.housekeeping.threads.on in all the Cloudera Data Hub clusters' Hive service, the housekeeping can be done by a single Hive metastore in the DataLake cluster.

Related Information

Creating partitions dynamically

Apache Wiki: Discover Partitions and Partition Retention

Hive Metastore leader election

Managing partition retention time

You can keep the size of the Apache Hive metadata and data you accumulate for log processing, and other activities, to a manageable size by setting a retention period for the data.

Before you begin

The table must be configured to automatically synchronize partition metadata with directories or objects on a file system.

About this task

When a partition metadata retention period is specified, Hive will drop the metadata and corresponding data for partitions created after the retention period, and this action will also apply to existing partitions. You express the retention time using a numeral and the following character or characters:

- ms (milliseconds)
- s (seconds)
- m (minutes)
- d (days)

In this task, you configure automatic synchronization of the file system partitions with the metastore and a partition retention period. Assume you already created a partitioned, external table named employees.

Procedure

1. If necessary, enable automatic discovery of partitions for the table employees.

```
ALTER TABLE employees SET TBLPROPERTIES ('discover.partitions'='true');
```

   By default, external partitioned tables already set this table property to true.

2. Configure a partition retention period of one week.

```
ALTER TABLE employees SET TBLPROPERTIES ('partition.retention.period'='7
d');
```

   The partition metadata as well as the actual data for employees in Hive is automatically dropped after a week.

Related Information

Creating partitions dynamically

Apache Wiki: Discover Partitions and Partition Retention

Repairing partitions manually using MSCK repair

The MSCK REPAIR TABLE command was designed to manually add partitions that are added to or removed from the file system, but are not present in the Hive metastore.

About this task

This task assumes you created a partitioned external table named emp_part that stores partitions outside the warehouse. You remove one of the partition directories on the file system. This action renders the metastore inconsistent with the file system. You repair the discrepancy manually to synchronize the metastore with the file system.

Procedure

1. Remove the dept=sales object from the file system.
2. From the Hive command line, look at the emp_part table partitions.

```
SHOW PARTITIONS emp_part;
```

   The list of partitions is stale; it still includes the dept=sales directory.

```
+----------------+
|   partition    |
+----------------+
| dept=finance   |
| dept=sales     |
| dept=service   |
+----------------+
```

3. Repair the partition manually.

```
MSCK REPAIR TABLE emp_part DROP PARTITIONS;
```

Query scheduling

Apache Hive scheduled queries is a simple, secure way to create, manage, and monitor scheduled jobs. For applications that require OS-level schedulers like cron, Apache Oozie, or Apache Airflow, you can use scheduled queries.

Using SQL statements, you can schedule Hive queries to run on a recurring basis, monitor query progress, and optionally disable a query schedule. You can run queries to ingest data periodically, refresh materialized views, replicate data, and perform other repetitive tasks. For example, you can insert data from a stream into a transactional table every 10 minutes, refresh a materialized view used for BI reporting every hour, and replicate data from one cluster to another on a daily basis.

A Hive scheduled query consists of the following parts:

- A unique name for the schedule
- The SQL statement to be executed
- The execution schedule defined by a Quartz cron expression.

Quartz cron expressions are expressive and flexible. For instance, expressions can describe simple schedules such as every 10 minutes, but also an execution happening at 10 AM on the first Sunday of the month in January, February in 2021, 2022. You can describe common schedules in an easily comprehensible format, for example every 20 minutes or every day at ‘3:25:00’.

Operation

A scheduled query belongs to a namespace, which is a collection of HiveServer (HS2) instances that are responsible for executing the query. Scheduled queries are stored in the Hive metastore. The metastore stores scheduled queries, the status of ongoing and previously executed statements, and other information. HiveServer periodically polls the metastore to retrieve scheduled queries that are due to be executed. If you run multiple HiveServer roles, the metastore guarantees that only one of them executes a certain scheduled query at any given time.

You create, alter, and drop scheduled queries using dedicated SQL statements.

Related Information

Apache Hive Language Manual--Scheduled Queries

Enabling scheduled queries

You need to know how to enable and disable scheduled queries and understand how the default state can prevent you from running a query unintentionally.

About this task

Scheduled queries are created in disabled mode by default in Cloudera. This default helps prevent you from running new scheduled queries inadvertantly. You must explicitly enable new scheduled queries. A scheduled query can keep the cluster awake at the wrong time. To enable a particular schedule, for example schedule1, you run the ALTER SCHEDULED QUERY statement:

```
ALTER SCHEDULED QUERY schedule1 ENABLE;
```

To disable this schedule: ALTER SCHEDULED QUERY schedule1 DISABLE;

Related Information

Apache Hive Language Manual--Scheduled Queries

Periodically rebuilding a materialized view

Using materialized views can enhance query performance. You need to update materialized view contents when new data is added to the underlying table. Instead of rebuilding the materialized view manually, you can schedule this task. Automatic rebuilding then occurs periodically and transparently to users.

About this task

This task assumes you created the following schemas for storing employee and departmental information:

```
CREATE TABLE emps (
empid INTEGER,
deptno INTEGER,
name VARCHAR(256),
salary FLOAT,
hire_date TIMESTAMP);

CREATE TABLE depts (
deptno INTEGER,
deptname VARCHAR(256),
locationid INTEGER);
```

Imagine that you add data for a number of employees to the table. Assume many users of your database issue queries to access to data about the employees hired during last year including the department they belong to.

You perform the steps below to create a materialized view of the table to address these queries. Imagine new employees are hired and you add their records to the table. These changes render the materialized view contents outdated. You need to update its contents. You create a scheduled query to perform this task. The scheduled rebuilding will not occur unless there are changes to the input tables. You test the scheduled query by bypassing the schedule and executing the schedule immediately. Finally, you change the schedule to rebuild less often.

Procedure

1. To handle many queries to access recently hired employee and departmental data, create a materialized view.

```
CREATE MATERIALIZED VIEW mv_recently_hired AS
  SELECT empid, name, deptname, hire_date FROM emps
  JOIN depts ON (emps.deptno = depts.deptno)
  WHERE hire_date >= '2020-01-01 00:00:00';
```

2. Use the materialized view by querying the employee data.

```
SELECT empid, name FROM emps
JOIN depts  ON (emps.deptno = depts.deptno)
WHERE hire_date >= '2020-03-01 00:00:00' AND deptname = 'finance';
```

3. Assuming new hiring occurred and you added new records to the emps table, rebuild the materialized view.

```
ALTER MATERIALIZED VIEW mv_recently_hired REBUILD;
```

   The rebuilding updates the contents of the materialized view.

4. Create a scheduled query to invoke the rebuild statement every 10 minutes.

```
CREATE SCHEDULED QUERY scheduled_rebuild
EVERY 10 MINUTES AS
```

```
ALTER MATERIALIZED VIEW mv_recently_hired REBUILD;
```

A rebuild executes every 10 minutes, assuming changes to the emp table occur within that period. If a materialized view can be rebuilt incrementally, the scheduled rebuild does not occur unless there are changes to the input tables.

5. To test the schedule, run a scheduled query immediately.

```
ALTER SCHEDULED QUERY scheduled_rebuild EXECUTE;
```

6. Change the frequency of the rebuilding.

```
ALTER SCHEDULED QUERY scheduled_rebuild EVERY 20 MINUTES;
```

Related Information

Apache Hive Language Manual--Scheduled Queries

Getting scheduled query information and monitor the query

After you create a scheduled query you can access information about it in the scheduled_queries table of the Hive information schema. You can also use the information schema to monitor scheduled query execution.

Procedure

1. Query the information schema to get information about a schedule.

```
SELECT *
FROM information_schema.scheduled_queries
WHERE schedule_name = 'scheduled_rebuild';
```

The following information appears about the scheduled query:

scheduled_query_id
: Unique numeric identifier for a scheduled query.

schedule_name
: Name of the scheduled query.

enabled
: Whether the scheduled query is currently enabled or not.

cluster_namespace
: Namespace that the scheduled query belongs to.

schedule
: Schedule described as a Quartz cron expression.

user
: Owner of the scheduled query.

query
: SQL query to be executed.

next_execution
: When the next execution of this scheduled query is due.

2. Monitor the most recent scheduled query execution.

```
SELECT *
```

```
FROM information_schema.scheduled_executions;
```

You can configure the retention period for this information in the Hive metastore.

scheduled_execution_id

Unique numeric identifier for a scheduled query execution.

schedule_name

Name of the scheduled query associated with this execution.

executor_query_id

Query ID assigned to the execution by HiveServer (HS2).

state

One of the following phases of execution.

- STARTED. A scheduled query is due and a HiveServer instance has retrieved its information.
- EXECUTING. HiveServer is executing the query and reporting progress in configurable intervals.
- FAILED. The query execution was stopped due to an error or exception.
- FINISHED. The query execution was successful.
- TIMED_OUT. HiveServer did not provide an update on the query status for more than a configurable timeout.

start_time

Start time of execution.

end_time

End time of execution.

elapsed

Difference between start and end time.

error_message

If the scheduled query failed, it contains the error message associated with its failure.

last_update_time

Time of the last update of the query status by HiveServer.

Related Information

Apache Hive Language Manual--Scheduled Queries

Materialized views

A materialized view is a Hive-managed database object that holds a query result you can use to speed up the execution of a query workload. If your queries are repetitive, you can reduce latency and resource consumption by using materialized views. You create materialized views to optimize your queries automatically.

Using a materialized view, the optimizer can compare old and new tables, rewrite queries to accelerate processing, and manage maintenance of the materialized view when data updates occur. The optimizer can use a materialized view to fully or partially rewrite projections, filters, joins, and aggregations.

You can perform the following materialized view operations:

- Create a materialized view of queries or subqueries
- Drop a materialized view
- Show materialized views
- Describe a materialized view

- Enable or disable query rewriting based on a materialized view
- Globally enable or disable rewriting based on any materialized view
- Use partitioning to improve the performance of materialized views

Related Information

Materialized view commands

Creating and using a materialized view

You can create a materialized view of a query to calculate and store results of an expensive operation, such as a particular join, on a managed, ACID table that you repeatedly run. When you issue queries specified by that materialized view, the optimizer rewrites the query based on it. This action saves reprocessing. Query performance improves.

About this task

In the tasks that follow, you create and populate example tables. The tables are managed tables. You cannot create a materialized view of an external table. You create a materialized view of a join of the tables. Subsequently, you run a query to join the tables, and the query plan takes advantage of the precomputed join to accelerate processing. These over-simplified tasks show the syntax and output of a materialized view, and do not demonstrate accelerated processing that occurs in a real-world task, processing a large amount of data.

Related Information

Materialized view commands

Creating the tables and view

You see how to create simple tables, insert the data, and join the tables using a materialized view. You run the query, and the optimizer takes advantage of the precomputation performed by the materialized view to speed response time.

Procedure

1. Create two ACID tables:

```
CREATE TABLE emps (
empid INT,
deptno INT,
name VARCHAR(256),
salary FLOAT,
hire_date TIMESTAMP);

CREATE TABLE depts (
deptno INT,
deptname VARCHAR(256),
locationid INT);
```

2. Insert some data into the tables for example purposes:

```
INSERT INTO TABLE emps VALUES (10001,101,'jane doe',250000,'2018-01-10');
INSERT INTO TABLE emps VALUES (10002,100,'somporn klailee',210000,'2017-12
-25');
INSERT INTO TABLE emps VALUES (10003,200,'jeiranan thongnopneua',175000,'
2018-05-05');

INSERT INTO TABLE depts VALUES (100,'HR',10);
INSERT INTO TABLE depts VALUES (101,'Eng',11);
INSERT INTO TABLE depts VALUES (200,'Sup',20);
```

Tables must be ACID (managed) tables.

3. Create a materialized view to join the tables:

```
CREATE MATERIALIZED VIEW mv1
AS SELECT empid, deptname, hire_date
FROM emps JOIN depts
ON (emps.deptno = depts.deptno)
WHERE hire_date >= '2017-01-01';
```

4. Run a query that takes advantage of the precomputation performed by the materialized view:

```
SELECT empid, deptname
FROM emps
JOIN depts
ON (emps.deptno = depts.deptno)
WHERE hire_date >= '2017-01-01'
AND hire_date <= '2019-01-01';
```

Output is:

```
+--------+-----------+
| empid  | deptname  |
+--------+-----------+
| 10003  | Sup       |
| 10002  | HR        |
| 10001  | Eng       |
+--------+-----------+
```

Verifying use of a query rewrite

You can use Apache Hive explain logging to check that a materialized view used a query rewrite.

About this task

By default explain logging is set to false in Cloudera.

Procedure

1. In Cloudera Manager, enable explain logging: Navigate to Clusters HIVE_ON_TEZ-1 Configuration , search for hive.log.explain.output, check HiveServer2 Default Group, and click Save Changes.

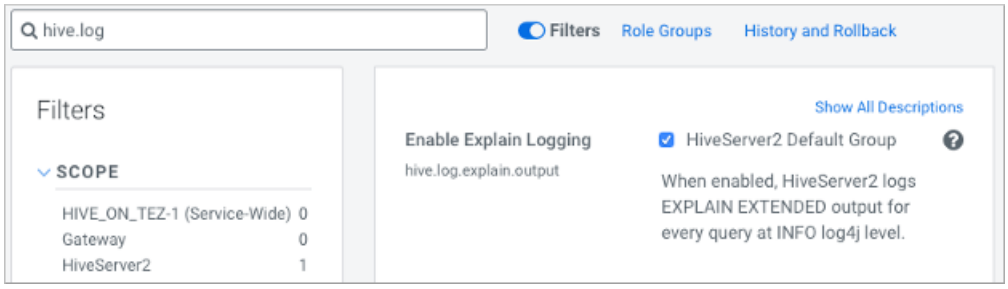

2. Verify that the query rewrite used the materialized view by running an extended EXPLAIN statement:

```
EXPLAIN EXTENDED SELECT empid, deptname
                        FROM emps
                        JOIN depts
                        ON (emps.deptno = depts.deptno)
                        WHERE hire_date >= '2017-01-01'
                        AND hire_date <= '2019-01-01';
```

The output shows the alias default.mv1 for the materialized view in the TableScan section of the plan.

```
OPTIMIZED SQL: SELECT `empid`, `deptname`
```

```
                          FROM `default`.`mv1`

                          WHERE TIMESTAMP '2019-01-01 00:00:00.000000000' >=
 `hire_date`
                          STAGE DEPENDENCIES:

                          Stage-0 is a root stage

                          STAGE PLANS:

                          Stage: Stage-0
                          Fetch Operator
                          limit: -1
                          Processor Tree:
                          TableScan
                          alias: default.mv1
                          filterExpr: (hire_date <= TIMESTAMP'2019-01-01
                          00:00:00') (type: boolean) |
                          GatherStats: false
                          Filter Operator
                          isSamplingPred: false
                          predicate: (hire_date <= TIMESTAMP'2019-01-01
                          00:00:00') (type: boolean)
                          Select Operator
                          expressions: empid (type: int), deptname (type: v
archar(256))
                          outputColumnNames: _col0, _col1
                          ListSink
```

Using optimizations from a subquery

You can create a query having a subquery that the optimizer rewrites based on a materialized view. You create a materialized view, and then run a query that uses the materialized view.

About this task

In this task, you create a materialized view and use it in a subquery to return the number of destination-origin pairs. Suppose the data resides in a table named flights_data that has the following columns:

c_id	dest	origin
1	Chicago	Hyderabad
2	London	Moscow
...		

Procedure

1. Create a table schema definition named flights_data for destination and origin data.

```
CREATE TABLE flights_data(
  c_id INT,
  dest VARCHAR(256),
  origin VARCHAR(256));
```

2. Create a materialized view that counts destinations and origins.

```
CREATE MATERIALIZED VIEW mv1
AS
  SELECT dest, origin, count(*)
  FROM flights_data
  GROUP BY dest, origin;
```

3. Take advantage of the materialized view to speed your queries when you have to count destinations and origins again.

   For example, use a subquery to select the number of destination-origin pairs like the materialized view.

```
SELECT count(*)/2
FROM(
  SELECT dest, origin, count(*)
  FROM flights_data
  GROUP BY dest, origin
) AS t;
```

   Transparently, the SQL engine uses the work already in place since creation of the materialized view instead of reprocessing.

Related Information

Materialized view commands

Dropping a materialized view

You must understand when to drop a materialized view to successfully drop related tables.

About this task

Drop a materialized view before performing a DROP TABLE operation on a related table. You cannot drop a table that has a relationship with a materialized view.

In this task, you drop a materialized view named mv1 from the database named default.

Procedure

Drop a materialized view in my_database named mv1 .
DROP MATERIALIZED VIEW default.mv;

Related Information

Materialized view commands

Showing materialized views

You can list all materialized views in the current database or in another database. You can filter a list of materialized views in a specified database using regular expression wildcards.

About this task

You can use regular expression wildcards to filter the list of materialized views you want to see. The following wildcards are supported:

- Asterisk (*)

  Represents one or more characters.
- Pipe symbol (|)

  Represents a choice.

For example, mv_q* and *mv|q1* match the materialized view mv_q1. Finding no match does not cause an error.

Procedure

1. List materialized views in the current database.
   SHOW MATERIALIZED VIEWS;
2. List materialized views in a particular database.
   SHOW MATERIALIZED VIEWS IN another_database;

Related Information

Materialized view commands

Describing a materialized view

You can get summary, detailed, and formatted information about a materialized view.

About this task

This task builds on the task that creates a materialized view named mv1.

Procedure

1. Get summary information about the materialized view named mv1.

```
DESCRIBE mv1;
```

```
+------------+---------------+----------+
|  col_name  |   data_type   | comment  |
+------------+---------------+----------+
| empid      | int           |          |
| deptname   | varchar(256)  |          |
| hire_date  | timestamp     |          |
+------------+---------------+----------+
```

2. Get detailed information about the materialized view named mv1.

```
DESCRIBE EXTENDED mv1;
```

```
+-----------------------------+---------------------------------...
|          col_name           |                      data_type   ...
+-----------------------------+---------------------------------...
| empid                       | int                                 ...
| deptname                    | varchar(256)                        ...
| hire_date                   | timestamp                           ...
|                             | NULL                                ...
| Detailed Table Information  |Table(tableName:mv1, dbName:default, own
er:hive, createTime:1532466307, lastAccessTime:0, retention:0, sd:Storag
eDescriptor(cols:[FieldSchema(name:empid, type:int, comment:null), Field
Schema(name:deptname, type:varchar(256), comment:null), FieldSchema(name
:hire_date, type:timestamp, comment:null)], location:hdfs://myserver.com
:8020/warehouse/tablespace/managed/hive/mv1, inputFormat:org.apache.hado
op.hive.ql.io.orc.OrcInputFormat, outputFormat:org.apache.hadoop.hive.ql
.io.orc.OrcOutputFormat, compressed:false, numBuckets:-1, serdeInfo:SerD
eInfo(name:null, serializationLib:org.apache.hadoop.hive.ql.io.orc.OrcSe
rde, parameters:{}), bucketCols:[], sortCols:[], parameters:{}, skewedIn
fo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocat
ionMaps:{}), storedAsSubDirectories:false), partitionKeys:[], parameters
:{totalSize=488, numRows=4, rawDataSize=520, COLUMN_STATS_ACCURATE={\"BA
SIC_STATS\":\"true\"}, numFiles=1, transient_lastDdlTime=1532466307, buc
keting_version=2}, viewOriginalText:SELECT empid, deptname, hire_date\nF
ROM emps2 JOIN depts\nON (emps2.deptno = depts.deptno)\nWHERE hire_date >=
 '2017-01-17', viewExpandedText:SELECT `emps2`.`empid`, `depts`.`deptname`
, `emps2`.`hire_date`\nFROM `default`.`emps2` JOIN `default`.`depts`\nON
 (`emps2`.`deptno` = `depts`.`deptno`)\nWHERE `emps2`.`hire_date` >= '20
17-01-17', tableType:MATERIALIZED_VIEW, rewriteEnabled:true, creationMet
adata:CreationMetadata(catName:hive, dbName:default, tblName:mv1, tables
Used:[default.depts, default.emps2], validTxnList:53$default.depts:2:922
3372036854775807::$default.emps2:4:9223372036854775807::, materializatio
nTime:1532466307861), catName:hive, ownerType:USER)
```

3. Get formatting details about the materialized view named mv1.

```
DESCRIBE FORMATTED mv1;
```

```
+-------------------------------+--------------------------------...
|           col_name            |                     data_type  ...
+-------------------------------+--------------------------------...
| # col_name                    | data_type                      ...
| empid                         | int                            ...
| deptname                      | varchar(256)                   ...
| hire_date                     | timestamp                      ...
|                               | NULL                           ...
| # Detailed Table Information  | NULL                           ...
| Database:                     | default                        ...
| OwnerType:                    | USER                           ...
| Owner:                        | hive                           ...
| CreateTime:                   | Tue Jul 24 21:05:07 UTC 2019   ...
| LastAccessTime:               | UNKNOWN                        ...
| Retention:                    | 0                              ...
| Location:                     | hdfs://myserver...
| Table Type:                   | MATERIALIZED_VIEW              ...
| Table Parameters:             | NULL                           ...

|                               | COLUMN_STATS_ACCURATE          ...

|                               | bucketing_version              ...

|                               | numFiles                       ...

|                               | numRows                        ...

|                               | rawDataSize                    ...

|                               | totalSize                      ...

|                               | transient_lastDdlTime          ...
|                               | NULL                           ...

| # Storage Information         | NULL                           ...

| SerDe Library:                | org.apache.hadoop.hive.ql.io.or...

| InputFormat:                  | org.apache.hadoop.hive.ql.io.or...
| OutputFormat:                 | org.apache.hadoop.hive.ql.io.or...
| Compressed:                   | No                             ...

| Num Buckets:                  | -1                             ...

| Bucket Columns:               | []                             ...

| Sort Columns:                 | []                             ...
| # View Information            | NULL                           ...

| View Original Text:           | SELECT empid, deptname, hire_da...

| View Expanded Text:           | SELECT `emps2`.`empid`, `depts`...
| View Rewrite Enabled:         | Yes                            ...
```

Related Information

Materialized view commands

Managing query rewrites

After changes to base tables, the data in a materialized view is stale. You need to know how to prevent the SQL optimizer from rewriting queries in this situation. If you want a query executed without regard to a materialized view, for example to measure the execution time difference, you can disable rewriting and then enable it again.

About this task

As administrator, you can globally enable or disable all query rewrites based on materialized views. By default, the optimizer rewrites a query based on a materialized view.

Procedure

1. Disable rewriting of a query based on a materialized view named mv1 in the default database.

```
ALTER MATERIALIZED VIEW default.mv1 DISABLE REWRITE;
```

2. Enable rewriting of a query based on materialized view mv1.

```
ALTER MATERIALIZED VIEW default.mv1 ENABLE REWRITE;
```

3. Globally enable rewriting of queries based on materialized views by setting a global property.

```
SET hive.materializedview.rewriting=true;
```

Related Information

Materialized view commands

Purposely using a stale materialized view

A rewrite of a query based on a stale materialized view does not occur automatically. If you want a rewrite of a stale or possibly stale materialized view, you can force a rewrite.

About this task

For example, you might want to use the contents of a materialized view of a non-transactional table because the freshness of such a table is unknown. The optimizer cannot determine the data freshness if you use external tables. You can purposely rewrite a query based on a stale materialized views using these techniques:

- Schedule the materialized view for rebuilding. For example, schedule a rebuild to occur every x minutes.
- Adjust the rewriting time window to use stale or possibly stale data for a period of time. For example, schedule the window within which to use stale data for x + y minutes.

Procedure

1. Create a scheduled query to invoke the rebuild statement every 10 minutes.

```
CREATE SCHEDULED QUERY scheduled_rebuild
EVERY 10 MINUTES AS
ALTER MATERIALIZED VIEW mv_recently_hired REBUILD;
```

2. Define the window of time for using stale data.

```
SET hive.materializedview.rewriting.time.window=10min;
```

Creating and using a partitioned materialized view

When creating a materialized view, you can partition selected columns to improve performance. Partitioning separates the view of a table into parts, which often improves query rewrites of partition-wise joins of materialized views with tables or other materialized views.

About this task

This task assumes you created a materialized view of the emps and depts tables and assumes you created these tables. The emps table contains the following data:

empid	deptno	name	salary	hire_date
10001	101	jane doe	250000	2018-01-10
10005	100	somporn klailee	210000	2017-12-25
10006	200	jeiranan thongnopneua	175000	2018-05-05

The depts table contains the following data:

deptno	deptname	locationid
100	HR	10
101	Eng	11
200	Sup	20

In this task, you create two materialized views: one partitions data on department; the other partitions data on hire date. You select data, filtered by department,from the original table, not from either one of the materialized views. The explain plan shows that Hive rewrites your query for efficiency to select data from the materialized view that partitions data by department. In this task, you also see the effects of rebuilding a materialized view.

Procedure

1. Create a materialized view of the emps table that partitions data into departments.

```
CREATE MATERIALIZED VIEW partition_mv_1 PARTITIONED ON (deptno)
AS SELECT hire_date, deptno FROM emps WHERE deptno > 100 AND deptno < 200;
```

2. Create a second materialized view that partitions the data on the hire date instead of the department number.

```
CREATE MATERIALIZED VIEW partition_mv_2 PARTITIONED ON (hire_date)
  AS SELECT deptno, hire_date FROM emps where deptno > 100 AND deptno < 2
00;
```

3. Generate an extended explain plan by selecting data for department 101 directly from the emps table without using the materialized view.

```
EXPLAIN EXTENDED SELECT deptno, hire_date FROM emps where deptno = 101;
```

   The explain plan shows that Hive rewrites your query for efficiency, using the better of the two materialized views for the job: partition_mv_1.

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| OPTIMIZED SQL: SELECT CAST(101 AS INTEGER) AS `deptno`, `hire_date` |
| FROM `default`.`partition_mv_1`                    |
| WHERE 101 = `deptno`                               |
| STAGE DEPENDENCIES:                                |
|   Stage-0 is a root stage
...
```

4. Correct Jane Doe's hire date to February 12, 2018, rebuild one of the materialized views, but not the other, and compare contents of both materialized views.

```
INSERT INTO emps VALUES (10001,101,'jane doe',250000,'2018-02-12');
ALTER MATERIALIZED VIEW partition_mv_1 REBUILD;
```

```
SELECT * FROM partition_mv_1 where deptno = 101;
SELECT * FROM partition_mv_2 where deptno = 101;
```

The output of selecting the rebuilt partition_mv_1 includes the original row and newly inserted row because INSERT does not perform in-place updates (overwrites).

```
+---------------------------+------------------------+
| partition_mv_1.hire_date  | partition_mv_1.deptno  |
+---------------------------+------------------------+
| 2018-01-10 00:00:00.0     | 101                    |
| 2018-02-12 00:00:00.0     | 101                    |
+---------------------------+------------------------+
```

The output from the other partition is stale because you did not rebuild it:

```
+------------------------+---------------------------+
| partition_mv_2.deptno  | partition_mv_2.hire_date  |
+------------------------+---------------------------+
| 101                    | 2018-01-10 00:00:00.0     |
+------------------------+---------------------------+
```

5. Create a second employees table and a materialized view of the tables joined on the department number.

```
CREATE TABLE emps2 AS SELECT * FROM emps;

CREATE MATERIALIZED VIEW partition_mv_3 PARTITIONED ON (deptno) AS
  SELECT emps.hire_date, emps.deptno FROM emps, emps2
  WHERE emps.deptno = emps2.deptno
  AND emps.deptno > 100 AND emps.deptno < 200;
```

6. Generate an explain plan that joins tables emps and emps2 on department number using a query that omits the partitioned materialized view.

```
EXPLAIN EXTENDED SELECT emps.hire_date, emps.deptno FROM emps, emps2
  WHERE emps.deptno = emps2.deptno
  AND emps.deptno > 100 AND emps.deptno < 200;
```

The output shows that Hive rewrites the query to use the partitioned materialized view partition_mv_3 even though your query omitted the materialized view.

7. Verify that the partition_mv_3 sets up the partition for deptno=101 for partition_mv_3.

```
SHOW PARTITIONS partition_mv_3;
```

Output is:

```
+-------------+
|  partition  |
+-------------+
| deptno=101  |
+-------------+
```

Related Information

Creating and using a materialized view

Materialized view commands

HPL/SQL stored procedures

Cloudera supports Hive Hybrid Procedural SQL (HPL/SQL). HPL/SQL is an Apache open source procedural extension for SQL for Hive users. You connect over JDBC to Hive from a client to run HPL/SQL queries.

HPL/SQL includes imperative programming structures (variables, procedures, control flow, and exceptions), and is typically used for ETL. The HPL/SQL language understands the syntax and semantics of most procedural SQL dialects, such as Oracle PL/SQL.

HPL/SQL has its own grammar. For more information about the HPL/SQL language, see the HPL/SQL Reference.

HPL/SQL architecture

HPL/SQL has been re-architected from a command line tool to an integrated part of HiveServer (HS2). From a JDBC client, such as Beeline, you connect to HiveServer in Cloudera Public Cloud or Cloudera Private Cloud Base. The interpreter executes the abstract syntax tree (AST) from the parser. Hive metastore securely stores the function and procedure code permanently. The procedure is loaded and cached on demand to the interpreter's memory when needed. You can close the session or restart Hive without losing the definitions.

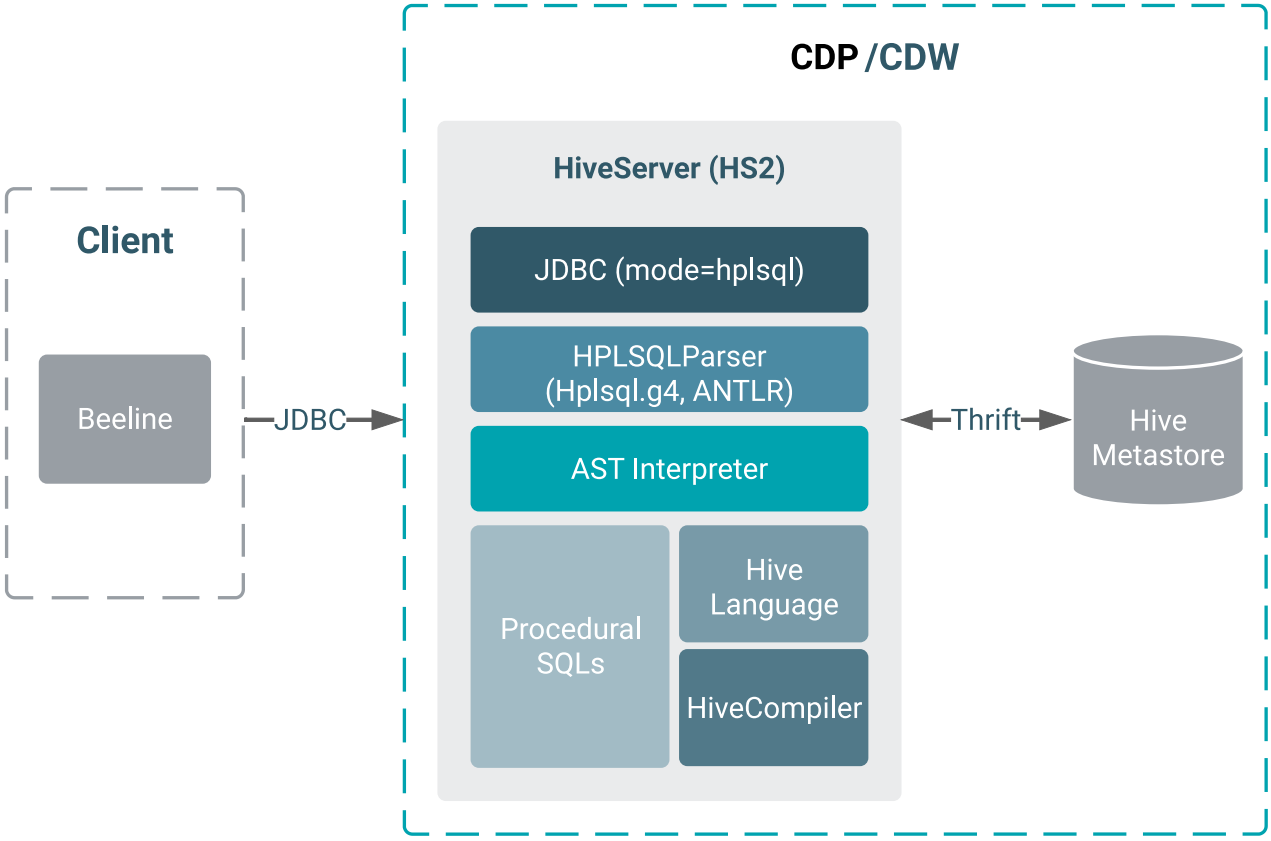

You can enable and use HPL/SQL from any host or third-party tool that can make a JDBC connection to HiveServer. Beeline is a popular client for use with HPL/SQL because other third-party tools do not show you some of the error messages about syntax mistakes.

Enabling HPL/SQL in the beeline connection string

After setting up a client to connect to Hive, you append mode=hplsql to the JDBC URL that connects the client to Hive:

```
beeline -u "jdbc:hive2://<HiveServer host>:10000/default;mode=hplsql"
```

When the client connects to HiveServer configuring this mode, HPL/SQL is enabled; otherwise, HPL/SQL is disabled.

HPL/SQL key features

- Flow of Control Statements (FOR, WHILE, IF, CASE, LOOP, LEAVE, RETURN)
- Functions, procedures, and packages
- Built-in functions (string manipulations, datetime functions, conversions)
- Exception handling and conditions
- Constants and variable, assignment (DECLARE count INT := 1)
- Processing results using a CURSOR
- UDF to run HPL/SQL scripts from Hive queries

  ```
  (SELECT hplsql('mycustomfunc(:1)', name) FROM users;)
  ```

- Bulk data processing with BULK COLLECT statement

HPL/SQL limitations

- Some of the Hive specific CREATE TABLE parameters are missing.
- No colon syntax to parametrize SQL strings.
- No quoted string literals.
- No GOTO and Label.
- EXECUTE does not have output parameters.
- Some complex data types, such as Arrays and Records are not supported.
- No object-oriented extension.
- Data Analytics Studio (DAS) does not support HPL/SQL.

Related Information

HPL/SQL Reference

Setting up a Hive client

As a Hive user who connects to HiveServer2 from a client, you need to know how to set up the client for using HPL/SQL.

Procedure

1. On the client end, download the latest version of the Hive JDBC driver from the Cloudera Downloads page.
2. Install the driver on the client end.

   Typically, you add the JAR file to the Libraries folder.
3. In Cloudera Public Cloud, in Cloudera Manager, navigate to Clusters Endpoints tab, and copy the Hive Server endpoint.
4. Paste the copied JDBC URL into a text file.

   ```
   jdbc:hive2://<host:port>/default;transportMode=http;httpPath=cliservice;
   ssl=true;retries=3
   ```

5. Append mode=hplsql to the end of the JDBC URL for connecting to Hive from the client.

```
jdbc:hive2://<host:port>/default;transportMode=http;httpPath=cliservice;
ssl=true;retries=3;mode=hplsql
```

6. On the client end, use the JDBC URL in Beeline to connect to Hive and enable HPL/SQL.

```
beeline -n <csso_username> -p <password> -u
        "jdbc:hive2://<host:port>/default;transportMode=http;httpPath=cl
iservice;ssl=true;retries=3;mode=hplsql"
```

At the Hive prompt, you can run HPL/SQL. You can use the forward slash (/) as a statement delimiter because a HPL/SQL statement can have multiple rows consisting of multiple SQL statements.

What to do next

Set hive.security.authorization.sqlstd.confwhitelist.append=QUERY_EXECUTOR in Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml before running HPL/SQL queries.

Creating a function

You need to know the syntax of HPL/SQL, which closely resembles Oracle's PL/SQL. An example of creating a function and calling it in a Hive SELECT statement demonstrates the HPL/SQL basics.

Procedure

1. Create a table and populate it with some numbers.

```
create table numbers (n int);

for i in 1..10 loop
  insert into numbers values(i);
end loop;
```

2. Create a function called fizzbuzz to return numbers.

```
create function fizzbuzz(n int) returns string
begin
  if mod(n, 15) == 0 then
     return 'FIZZBUZZ';
  elseif mod(n, 5) == 0 then
     return 'BUZZ';
  elseif mod(n, 3) == 0 then
     return 'FIZZ';
  else
     return n;
  end if;
end;
```

3. Call the function from a Hive select statement.

```
select fizzbuzz(n) from numbers;
```

Output looks something like this:

```
...
1
2
FIZZ
4
```

```
BUZZ
1
2
FIZZ
4
BUZZ
FIZZ
7
8
FIZZ
BUZZ
```

Related Information

HPL/SQL Reference

Using the cursor to return record sets

You use a cursor to return recordsets from stored procedures and functions. A cursor defines a set of rows and row-by-row operations on the set. You need to know how to open a cursor, fetch data into local variables row-by-row, and close a cursor.

About this task

A function called fizzbuzz2 calls another function fizzbuzz, created in the previous topic. The new function declares a cursor of type SYS_REFCURSOR to fetch the contents of the numbers table, also created in the previous topic.

Before you begin

- You created the fizzbuzz function.
- You created the numbers table.

Procedure

Use an OPEN-FOR statement to select the number in each row of the numbers table, iteratively FETCH the numerical content of the each row INTO cursor variable num, close the cursor, and print the results.

```
create function fizzbuzz2() returns string
begin
 declare num int = 0;
 declare result string = '';
 declare curs SYS_REFCURSOR;
 open curs for select n from numbers;
 fetch curs into num;
 while (SQLCODE = 0) do
    result = result || fizzbuzz(num) || ' ';
    fetch curs INTO num;
 end while;
 close curs;
 return result;
end;
print fizzbuzz2();
```

The output looks something like this:

```
...
1 2 FIZZ 4 BUZZ 1 2 FIZZ 4 BUZZ FIZZ 7 8 FIZZ BUZZ
No rows affected
```

Related Information

HPL/SQL Reference
Creating a function

Stored procedure examples

You see by example some of the frequently used and useful HPL/SQL code. HPL/SQL is an Apache open source procedural extension for SQL for Hive users.

Greeting function

The following example creates a function that takes the input of your name and returns "hello <name>":

```
CREATE PROCEDURE greet(name STRING)
BEGIN
  PRINT 'Hello ' || name;
END;
```

Pass a cursor to return records

This example of a procedure, called from another procedure, takes a cursor parameter declared OUT; an OPEN-FOR statement opens the cursor and executes the SELECT query. The query returns a subset of the records to the test_even procedure from the numbers table, created in a previous topic.

```
CREATE PROCEDURE even(cur OUT SYS_REFCURSOR)
BEGIN
OPEN cur FOR
SELECT n FROM numbers
WHERE MOD(n, 2) == 0;
END;
```

The test_even procedure below calls the even procedure above, passing the cursor of type SYS_REFCURSOR to fetch each row containing an even number.

```
CREATE PROCEDURE test_even()
BEGIN
 DECLARE curs SYS_REFCURSOR;
 DECLARE n INT = 0;
 DECLARE result STRING = 'Even numbers are: ';
 even(curs);
 FETCH curs INTO n;
 WHILE (SQLCODE = 0) DO
    result = result || n || ' ';
    FETCH curs INTO n;
 END WHILE;
 CLOSE curs;
 PRINT result;
END;
```

Using table types to index fields in records

You can set up table types and reference fields in records using indexing. This example assumes you created the emp table.

```
-- CREATE TABLE emp (name string, age int);
TYPE emp_type IS TABLE OF emp%ROWTYPE INDEX BY BINARY_INTEGER;
TYPE emp_age_type IS TABLE OF emp.age%TYPE INDEX BY BINARY_INTEGER;
TYPE emp_name_type IS TABLE OF STRING INDEX BY BINARY_INTEGER;
```

```
DECLARE rows emp_type;
DECLARE ages emp_age_type;

SELECT * INTO rows(1) FROM emp WHERE name = 'alice';
PRINT 'name=' || rows(1).name || ' age=' || rows(1).age;

SELECT age INTO ages(1) FROM emp WHERE name = 'alice';
PRINT 'age=' || ages(1);
```

BULK COLLECT

Using BULK COLLECT, you can retrieve multiple rows in a single fetch quickly.

```
TYPE emp_type IS TABLE OF emp%ROWTYPE INDEX BY BINARY_INTEGER;
    DECLARE rows emp_type;

    SELECT * BULK COLLECT INTO rows FROM emp;


    DECLARE idx INT = rows.FIRST;
    WHILE idx IS NOT NULL LOOP
    PRINT rows(idx).name || ' = ' || rows(idx).age;
    idx = rows.NEXT(idx);
    END LOOP;
```

Related Information

HPL/SQL Reference

Using functions

You can call a built-in Hive function to run one of a wide-range of operations instead of performing multiple steps. You use SHOW FUNCTIONS to search for or list available functions. You can create a user-defined function (UDF) when a built-in is not available to do what you need. You might need to reload functions to update the availability of functions created in another session.

Reloading, viewing, and filtering functions

To determine which Hive functions and operators are available, you reload functions, and then use the SHOW FUNCTIONS statement. An optional pattern in the statement filters the list of functions returned by the statement.

About this task

In this task, you first reload functions to make available any user-defined functions that were registered in Hive session after your session started. The syntax is:

```
RELOAD (FUNCTION|FUNCTIONS);
```

Next, you use the SHOW FUNCTIONS statement. The syntax of this statement is:

```
SHOW FUNCTIONS [LIKE "<pattern>"];
```

<pattern> represents search characters that can include regular expression wildcards.

Finally, you get more information about use by issuing the DESCRIBE FUNCTION statement.

Procedure

1. Open the Hive shell.
   Open a client connection, for example:

```
beeline -u jdbc:hive2://mycloudhost-3.com:10000 -n <your user name> -p
```

2. Reload functions to ensure all registered UDFs are available in your session.

```
RELOAD FUNCTIONS;
```

   Use the plural form of the command. RELOAD FUNCTION is for backward compatibility.

3. Generate a list of available built-in and user-defined functions (UDFs).

```
SHOW FUNCTIONS;
```

   The list of built-in functions, operators, and UDFs appear.

```
+-------------------------------+
|            tab_name           |
+-------------------------------+
| !                             |
| !=                            |
| $sum0                         |
| %                             |
...
```

4. Generate a filtered list of functions using the regular expression wildcard %.

```
SHOW FUNCTIONS LIKE "a%";
```

   All available functions that begin with the character a appear.

```
+-------------------------------+
|            tab_name           |
+-------------------------------+
| abs                           |
| acos                          |
| add_months                    |
...
```

5. Get more information about a particular function.

```
DESCRIBE FUNCTION abs;
```

```
+--------------------------------------------+
|                  tab_name                  |
+--------------------------------------------+
| ABS(x) - returns the absolute value of x   |
+--------------------------------------------+
```

6. Get more information about the function.

```
DESCRIBE FUNCTION EXTENDED abs;
```

```
+-----------------------------------------------------+
|                       tab_name                      |
+-----------------------------------------------------+
| ABS(x) - returns the absolute value of x            |
| Synonyms: abs                                       |
```

```
| Example:                                                |
|   > SELECT ABS(0) FROM src LIMIT 1;                     |
|   0                                                     |
|   > SELECT ABS(-5) FROM src LIMIT 1;                    |
|   5                                                     |
| Function class:org.apache.hadoop.hive.ql.udf.generic.GenericUDFAbs |
| Function type:BUILTIN                                   |
+-----------------------------------------------------+
```

Create a user-defined function

You export user-defined functionality (UDF) to a JAR from a Hadoop- and Hive-compatible Java project and store the JAR on your cluster or object store. Using Hive commands, you register the UDF based on the JAR, and call the UDF from a Hive query.

Before you begin

- You have access rights to upload the JAR to the cluster to your cluster or object store.

  Cloudera Manager, Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator).
- Hive on Tez or Hive LLAP is running on the cluster.
- You have installed Java and a Java integrated development environment (IDE) tool on the machine, or virtual machine, where you will create the UDF.

Setting up the development environment

You can create a Hive UDF in a development environment using IntelliJ, for example, and build the UDF. You define the Cloudera Maven Repository in your POM, which accesses necessary JARS hadoop-common-<version>.jar and hive-exec-<version>.jar.

Procedure

1. Open IntelliJ and create a new Maven-based project. Click Create New Project. Select Maven and the supported Java version as the Project SDK. Click Next.
2. Add archetype information.
   For example:
   - GroupId: com.mycompany.hiveudf
   - ArtifactId: hiveudf
3. Click Next and Finish.
   The generated pom.xml appears in sample-hiveudf.
4. To the pom.xml, add properties to facilitate versioning.
   For example:

```
<properties>
   <hadoop.version>TBD</hadoop.version>
   <hive.version>TBD</hive.version>
</properties>
```

5. In the pom.xml, define the repositories.

   Use internal repositories if you do not have internet access.

```
<repositories>
   <repository>
      <releases>
         <enabled>true</enabled>
         <updatePolicy>always</updatePolicy>
```

```
            <checksumPolicy>warn</checksumPolicy>
         </releases>
         <snapshots>
            <enabled>false</enabled>
            <updatePolicy>never</updatePolicy>
            <checksumPolicy>fail</checksumPolicy>
         </snapshots>
         <id>HDPReleases</id>
         <name>HDP Releases</name>
         <url>http://repo.hortonworks.com/content/repositories/releases/</u
rl>
         <layout>default</layout>
      </repository>
      <repository>
         <id>public.repo.hortonworks.com</id>
         <name>Public Hortonworks Maven Repo</name>
         <url>http://repo.hortonworks.com/content/groups/public/</url>
         <snapshots>
            <enabled>false</enabled>
         </snapshots>
      </repository>
      <repository>
            <id>repository.cloudera.com</id>
            <url>https://repository.cloudera.com/artifactory/cloudera-repos/<
/url>
      </repository>
   </repositories>
```

6. Define dependencies.
   For example:

```
<dependencies>
  <dependency>
        <groupId>org.apache.hive</groupId>
        <artifactId>hive-exec</artifactId>
        <version>${hive.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>${hadoop.version}</version>
    </dependency>
</dependencies>
```

Creating the UDF class

You define the UDF logic in a new class that returns the data type of a selected column in a table.

Procedure

1. In IntelliJ, click the vertical project tab, and expand hiveudf: hiveudf src main . Select the java directory, and on the context menu, select New Java Class , and name the class, for example, TypeOf.
2. Extend the GenericUDF class to include the logic that identifies the data type of a column.
   For example:

```
package com.mycompany.hiveudf;

import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDF;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.primitive.\
```

```
PrimitiveObjectInspectorFactory;
import org.apache.hadoop.io.Text;
public class TypeOf extends GenericUDF {
  private final Text output = new Text();
@Override
  public ObjectInspector initialize(ObjectInspector[] arguments) throws U
DFArgumentException {
    checkArgsSize(arguments, 1, 1);
    checkArgPrimitive(arguments, 0);
    ObjectInspector outputOI = PrimitiveObjectInspectorFactory.writableSt
ringObjectInspector;
    return outputOI;
  }

@Override
  public Object evaluate(DeferredObject[] arguments) throws HiveException
 {
    Object obj;
    if ((obj = arguments[0].get()) == null) {
      String res = "Type: NULL";
      output.set(res);
      } else {
      String res = "Type: " + obj.getClass().getName();
      output.set(res);
    }
    return output;
  }

@Override
  public String getDisplayString(String[] children) {
    return getStandardDisplayString("TYPEOF", children, ",");
  }
}
```

Building the project and upload the JAR

You compile the UDF code into a JAR and add the JAR to the classpath on the cluster. You choose one of several methods of configuring the cluster so Hive can find the JAR.

About this task

Cloudera Private Cloud Base

Use one of these methods to configure the cluster to find the JAR:

- Direct reference

  Straight-forward, but recommended for development only.
- Hive aux library directory

  Prevents accidental overwriting of files or functions. Recommended for tested, stable UDFs to prevent accidental overwriting of files or functions.
- Reloadable aux JAR

  Avoids HiveServer restarts. Recommended if you anticipate making frequent changes to the UDF logic.

Cloudera Public Cloud

Use the Direct reference method only.

Procedure

1. Build the IntelliJ project.

```
...
[INFO] Building jar: /Users/max/IdeaProjects/hiveudf/target/TypeOf-1.0-S
NAPSHOT.jar
[INFO] ------------------------------------------------------------------
-------
[INFO] BUILD SUCCESS
[INFO] -------------------------------------------------------------------
--------
[INFO] Total time: 14.820 s
[INFO] Finished at: 2019-04-03T16:53:04-07:00
[INFO] Final Memory: 26M/397M
[INFO] -------------------------------------------------------------------
--------

Process finished with exit code 0
```

2. In IntelliJ, navigate to the JAR in the /target directory of the project.
3. Configure the cluster so Hive can find the JAR using one of the following methods.
   - Direct JAR reference
     a. Upload the JAR to HDFS (Cloudera Private Cloud Base) or S3 (Public Cloud).
     b. Move the JAR into the Hive warehouse. For example, in Cloudera Private Cloud Base:

```
$ hdfs dfs -put TypeOf-1.0-SNAPSHOT.jar /warehouse/tablespace/manage
d/hiveudf-1.0-SNAPSHOT.jar
```

   - Hive aux JARs path (Cloudera Private Cloud Base only)
     a. In Cloudera Private Cloud Base, click Clusters and select the Hive service, for example, HIVE. Click Configuration and search for Hive Auxiliary JARs Directory.
     b. Specify a directory value for the Hive aux JARs property if necessary, or make a note of the path.
     c. Upload the JAR to the specified directory on all HiveServer instances (and all Metastore instances, if separate).
   - Reloadable aux JAR (Cloudera Private Cloud Base only)
     a. Upload the JAR to the /hadoop/hive-udf-dyn directory on all HiveServer instances (and all Metastore instances, if separate). An HDFS location is not supported.
     b. In hive-site.xml, set the following property: hive.reloadable.aux.jars.path=/hadoop/hive-udf-dyn.
4. In IntelliJ, click Save.
5. Click Actions Deploy Client Configuration .
6. Restart the Hive service. For example, restart HIVE.

Registering the UDF

In the cluster, you log into Hive, and run a command from Beeline to make the UDF functional in Hive queries. The UDF persists between HiveServer restarts.

Before you begin

You need to set up UDF access, using a Ranger policy for example.

About this task

In this task, the registration command differs depending on the method you choose to configure the cluster for finding the JAR. If you use the Hive aux library directory method that involves a symbolic link, you need to restart HiveServer after registration. If you use the Direct JAR reference or Reloadable aux JAR methods, you do not need

to restart HiveServer. You must recreate the symbolic link after any patch or maintenance upgrades that deploy a new version of Hive.

Procedure

1. Using Beeline, login to HiveServer as a user who has UDF access.

   HiveServer, for example:

   ```
   beeline -u jdbc:hive2://mycluster.com:10000 -n hive -p
   ```

2. At the Hive prompt, select a database for use.
   USE default;
3. Run the registration command that corresponds to the way you configured the cluster to find the JAR.

   In the case of the direct JAR reference configuration method, you include the JAR location in the command. If you use another method, you do not include the JAR location. The classloader can find the JAR.

   - Direct JAR reference:

     ```
     CREATE FUNCTION udftypeof AS 'com.mycompany.hiveudf.TypeOf01' USING JAR
     'S3:///warehouse/tablespace/managed/TypeOf01-1.0-SNAPSHOT.jar';
     ```

   - Hive aux library directory (Cloudera Private Cloud Base only):

     a. Set up a symbolic link on the command line of the local file system.

     ```
     ln -s /local-apps/hive-udf-aux <path to hive parcel>/hive/auxlib
     ```

     b. In Beeline, run the command to register the UDF.

     ```
     CREATE FUNCTION udftypeof AS 'com.mycompany.hiveudf.Typeof01';
     ```

     c. Restart HiveServer.

   - Reloadable aux JAR (Cloudera Private Cloud Base only):

     ```
     RELOAD;
     CREATE FUNCTION udftypeof AS 'com.mycompany.hiveudf.Typeof01';
     ```

4. Check that the UDF is registered.

   ```
   SHOW FUNCTIONS;
   ```

   You scroll through the output and find default.typeof.

Calling the UDF in a query

After registration of a UDF, you do not need to restart Hive before using the UDF in a query. In this example, you call the UDF you created in a SELECT statement, and Hive returns the data type of a column you specify.

Before you begin

- For the example query in this task, you need to create a table in Hive and insert some data.
- As a user, you need to have permission to call a UDF, which a Ranger policy can provide.

About this task

This task assumes you have the following example table in Hive:

```
+------------------+---------------+---------------+
|  students.name   | students.age  | students.gpa  |
+------------------+---------------+---------------+
| fred flintstone  | 35            | 1.28          |
```

```
| barney rubble    | 32             | 2.32          |
+------------------+---------------+---------------+
```

Procedure

1. Use the database in which you registered the UDF.
   For example:

   ```
   USE default;
   ```

2. Query Hive depending on how you configured the cluster for Hive to find the JAR.

   - Direct JAR reference or Hive aux library directory

     For example:

     ```
     SELECT students.name, udftypeof(students.name) AS type FROM students WHE
     RE age=35;
     ```

   - Reloadable aux JAR

     For example:

     ```
     RELOAD;
     SELECT students.name, udftypeof(students.name) AS type FROM students WHE
     RE age=35;
     ```

   You get the data type of the name column in the students table.

   ```
   +------------------+-----------------------------------------------------
   --------+
   |  students.name   |                        type
         |
   +------------------+-----------------------------------------------------
   --------+
   | fred flintstone  | Type: org.apache.hadoop.hive.serde2.io.HiveVarcharWri
   table |
   +------------------+-----------------------------------------------------
   --------+
   ```

Cloudera Runtime 7.3.1

Using Hue

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

About using Hue 5
Accessing and using Hue 5
Viewing Hive query details 11
Viewing Hive query history 11
Viewing Hive query information 11
Viewing explain plan for a Hive query 12
Viewing Hive query timeline 13
Viewing configurations for a Hive query 13
Viewing DAG information for a Hive query 15
Viewing Impala query details 18
Viewing Impala query history 18
Viewing Impala query information 19
Viewing the Impala query execution plan 19
Viewing the Impala query metrics 20
Viewing Impala profiles in Hue 21
Terminating Hive queries 21
Comparing Hive and Impala queries in Hue 22
Start SQL AI Assistant 24
Generate SQL from NQL 24
Edit query in natural language 25
Explain query in natural language 26
Optimize SQL query 28
Fixing a query in Hue 29
Generate comment for a SQL query 30
Enable stored procedures in Hue 31
Run stored procedure from Hue 32
Using SQL to query HBase from Hue 33
Querying existing HBase tables 34
Enabling the SQL editor autocompleter 34
Rerunning a query from the Job Browser page 35
Using governance-based data discovery 35
Searching metadata tags 36
Using Amazon S3 with Hue 36
Enabling S3 browser for Hue configured with IDBroker 37
Enabling S3 browser for Hue configured without IDBroker 38
Enabling S3 File Browser for Hue with RAZ in DataHub 38
Using Azure Data Lake Storage Gen2 with Hue 40
Enabling ABFS file browser for Hue configured with IDBroker 40
Enabling ABFS file browser for Hue configured without IDBroker 41
Enabling ABFS File Browser in Hue with RAZ in DataHub 42
Using Google Cloud Storage with Hue 43
Prerequisites for enabling GS File Browser 44
Enabling GS File Browser with RAZ 44
Disabling the automatic creation of user home directories 45
Granting permission to access S3, ABFS, GS File Browser in Hue 45
Creating tables in Hue by importing files 46
Supported non-ASCII and special characters in Hue 47
Options to rerun Oozie workflows in Hue 49
Unsupported features in Hue 50

Known limitations in Hue..... 50

About using Hue

Hue provides a one-stop querying experience in Cloudera that leverages Hive and Impala SQL queries.

Accessing and using Hue

Get started using Hue by analyzing and visualizing your data with Impala and Hive SQL query engines.

About this task

To try Hue without having an account, try running sample queries on http://demo.gethue.com/.

Procedure

1. Download and unzip one year of bike trips from the Bay Area Bike Share program. This file is about 80 MB in size.

2. Create a table from the ~/babs_open_data_year_1/201402_babs_open_data/201402_trip_data.csv file found in the unzipped babs_open_data_year_1.zip file:
   a) In the Cloudera Manager, select Hue WebUI Hue Load Balanced to launch Hue.
   b) In the left navigation panel of Hue, make sure the default database is selected, and click the plus sign to create a table as shown in the following image:

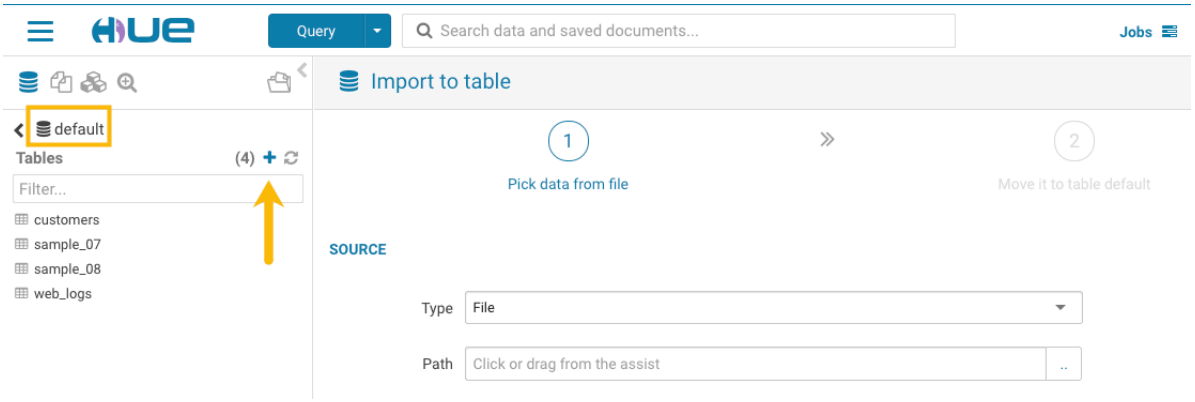

   If the default database is not selected, click the "less than" icon that is next to the database icon in the left panel. This enables you to select the default database.
   c) In the center panel Importer UI, set Type to File.
   d) Drag the 201402_trip_data.csv file to the Path field as shown in the following image:

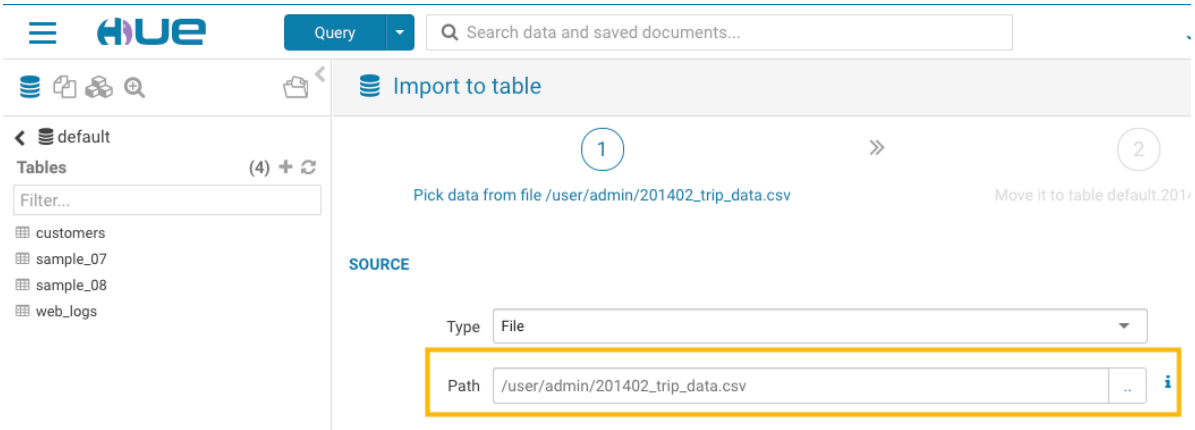

   e) Set the formats as follows:
      - Field Separator = Comma (,)
      - Record Separator = New line
      - Quote Character = Double Quote

      Then click Next at the bottom of the page.
   f) Set the properties Format = Text.
   g) Edit the FIELDS as follows:
      - Rename Bike # to Bike ID
      - Change the data type of ZipCode to string.
      - Remove all of the spaces in the Name fields.

      Then click Submit at the bottom of the page.

3. Click Query at the top of the page and select Editor Hive to open the Hive editor and then create a query.
   - Enter the following query into the editor window:

```
SELECT * FROM default.201402_trip_data
LIMIT 10;
```

   - Click the execute icon to run the query. The following rows are returned:

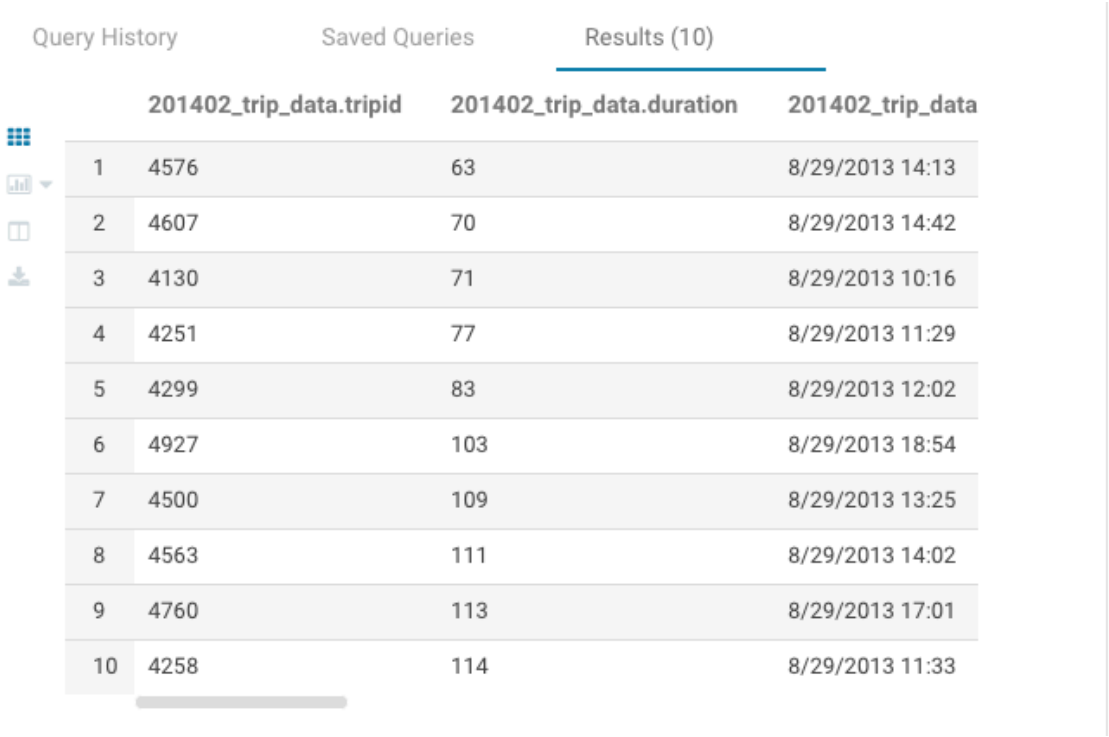

Query History Saved Queries Results (10)

	201402_trip_data.tripid	201402_trip_data.duration	201402_trip_data
1	4576	63	8/29/2013 14:13
2	4607	70	8/29/2013 14:42
3	4130	71	8/29/2013 10:16
4	4251	77	8/29/2013 11:29
5	4299	83	8/29/2013 12:02
6	4927	103	8/29/2013 18:54
7	4500	109	8/29/2013 13:25
8	4563	111	8/29/2013 14:02
9	4760	113	8/29/2013 17:01
10	4258	114	8/29/2013 11:33

4. Click Query at the top of the page and select Editor Impala to open the Impala SQL editor and then create a query.

   a. In the left panel, click the refresh icon and select Perform incremental metadata update to make the new table visible to Impala:

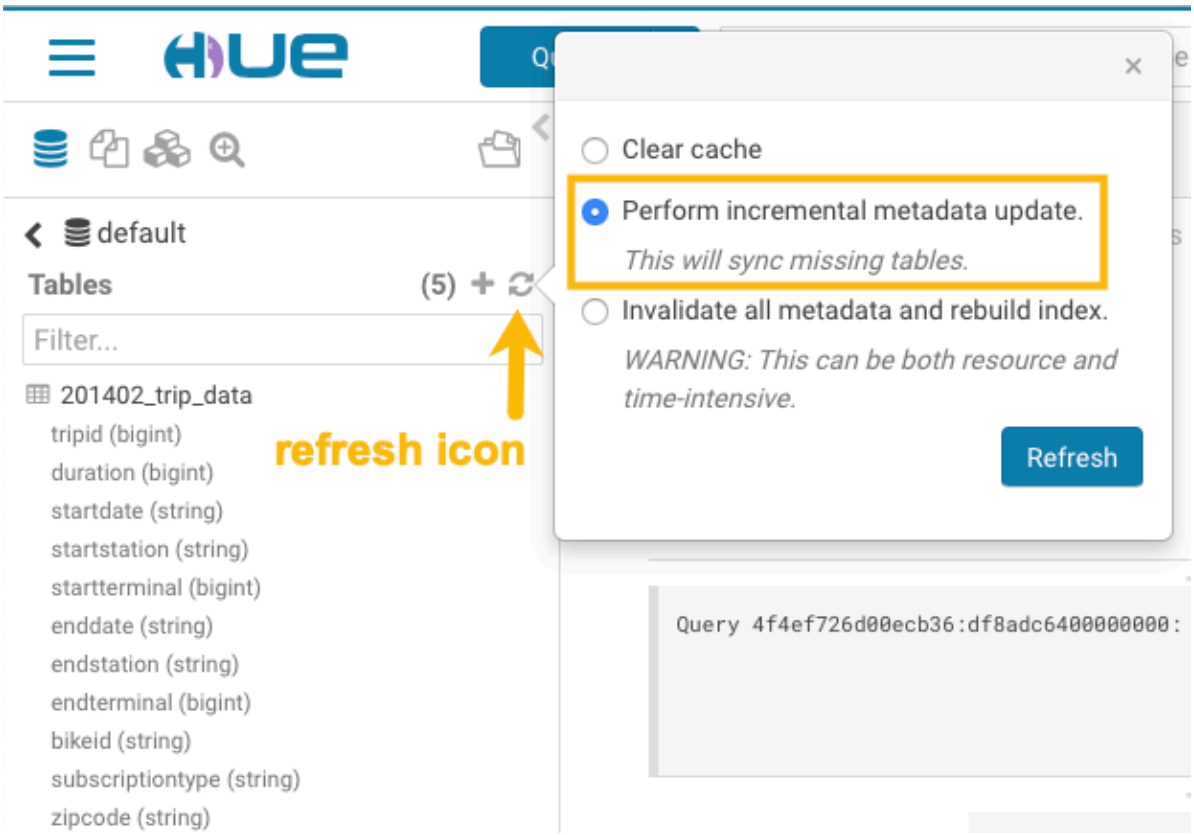

   b. Enter the following query into the editor window:

```
select 'startstation', 'endstation', count(*) as trips from default.'20
1402_trip_data'
group by 'startstation', 'endstation' order by trips desc;
```

c. Click the down arrow just under the execution icon and select Format:

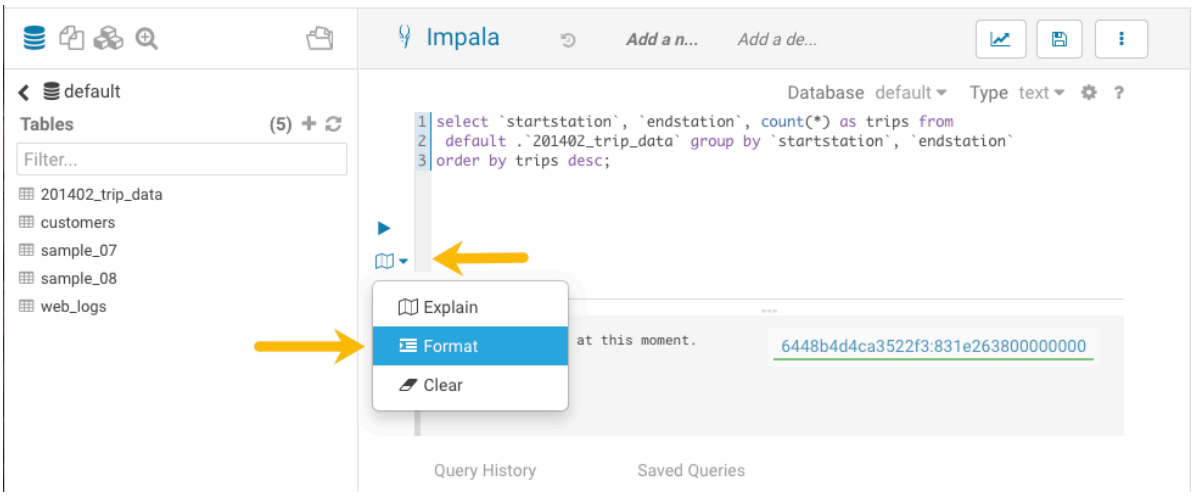

This reformats the query:

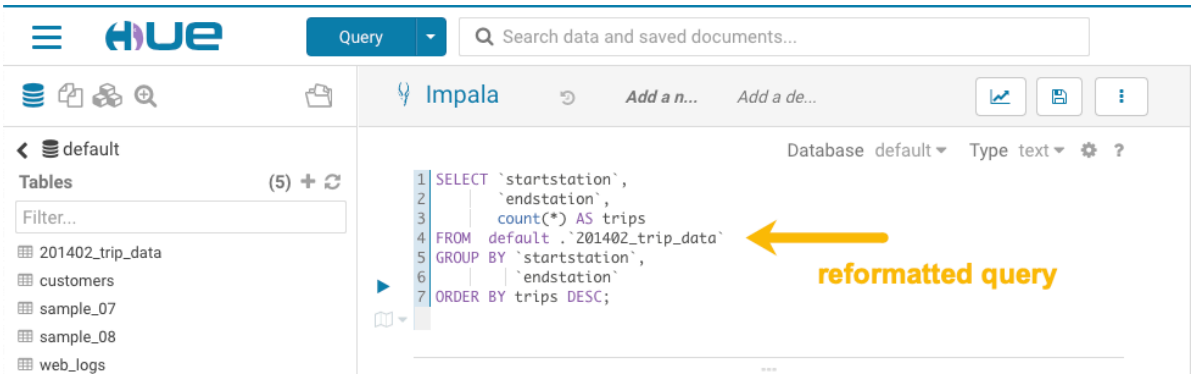

d.

Click the save icon , enter a query name, and click Save.

e.

Click the execute icon to run the query.

5. Create a bar chart that is based on the query results:

   a. Click the chart icon and then select Bars.

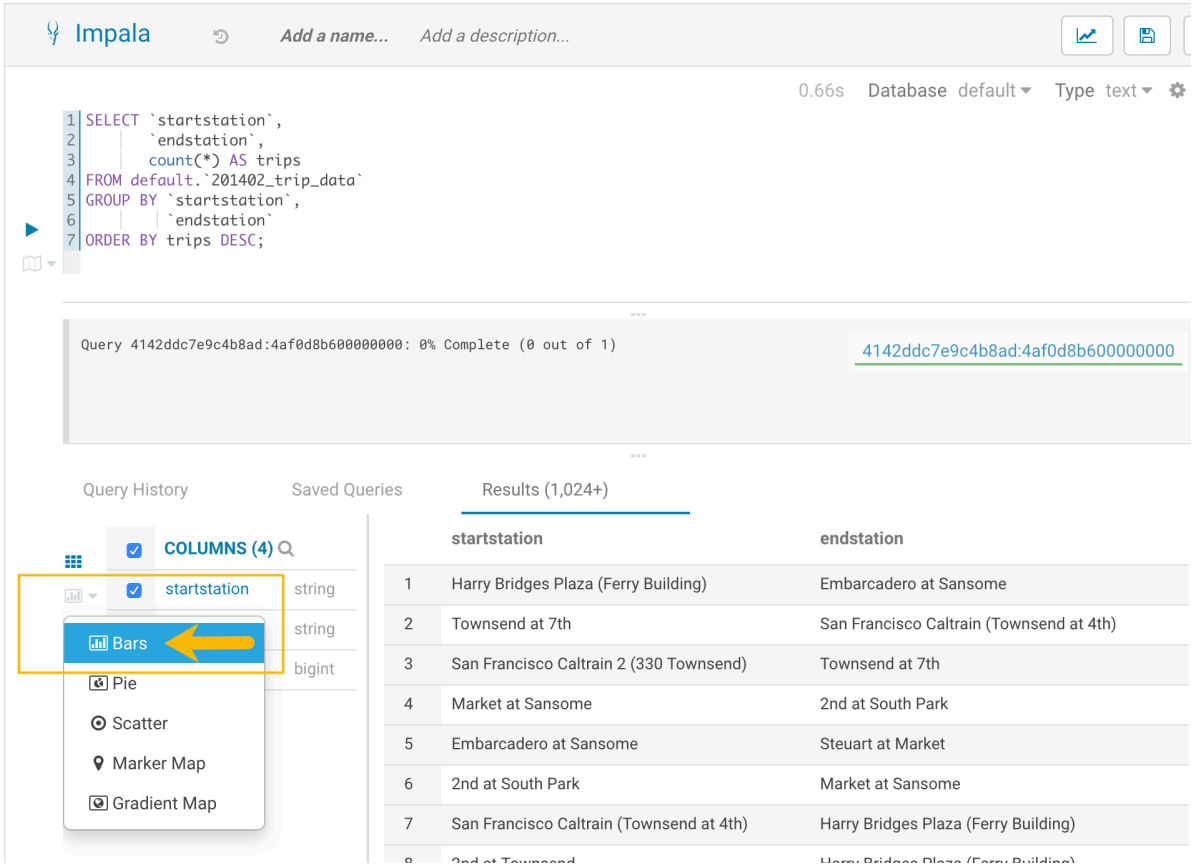

   b. Set the bar chart elements as follows:

   - X-AXIS = startstation
   - Y-AXIS = trips
   - LIMIT = 10

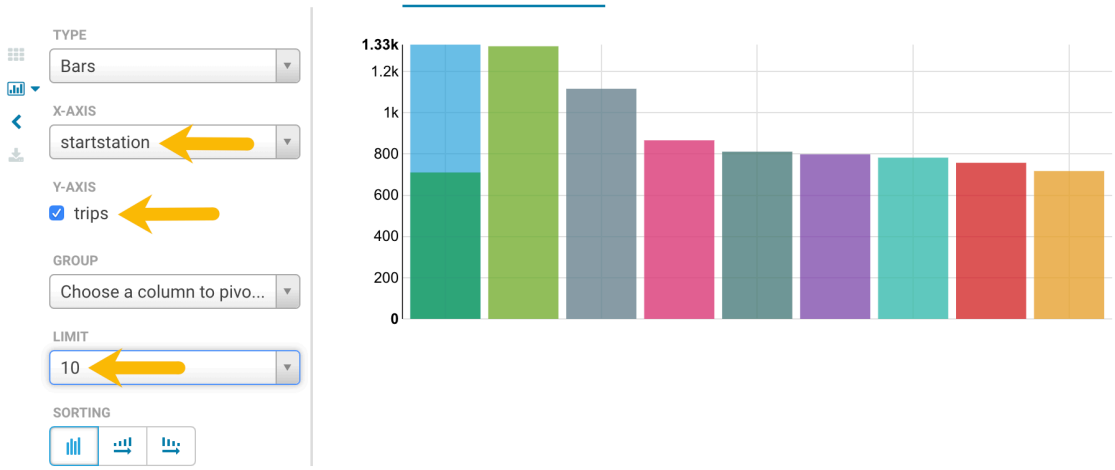

6. Create a pie chart by clicking the chart icon again and then select Pie.
7. Download the query results by clicking and selecting in what format you want to download, copy, or export the results.

Viewing Hive query details

You can search Hive query history, compare two queries, download debug bundles for troubleshooting, and view query details, a graphical representation of the query execution plan, and DAG information on the Job Browser page in Hue.

Viewing Hive query history

The Queries tab on the Job Browser page in Hue displays all the queries that were run from various query interfaces, such as Beeline, Hive Warehouse Connector (HWC), Tableau, Hue, and other JDBC BI clients and tools.

About this task

Only Query Processor Administrators can view historical queries of all users to monitor resource utilization and control costs from the Hue Job Browser. Non-admin users can view only their queries.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Click Queries.

   The Hive queries that were run for the past seven days are displayed. You can select the time period for which you want to view the historical data.

   You can also filter queries by their status.

Related Information

Adding Query Store Administrator users on Cloudera Data Hub clusters

Viewing Hive query information

The Query Info tab provides information such as, the Hive query ID, the user who executed the query, the start time, the end time, the total time taken to execute the query, the tables that were read and written, application ID, Directed Acyclic Graph (DAG) IDs, session ID, LLAP app ID, thread ID, and the queue against which the query was run.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.

3. Go to the **Queries** tab and click on the query for which you want to view the query details.

   The following image shows the **Query Info** tab on the Hue web interface:

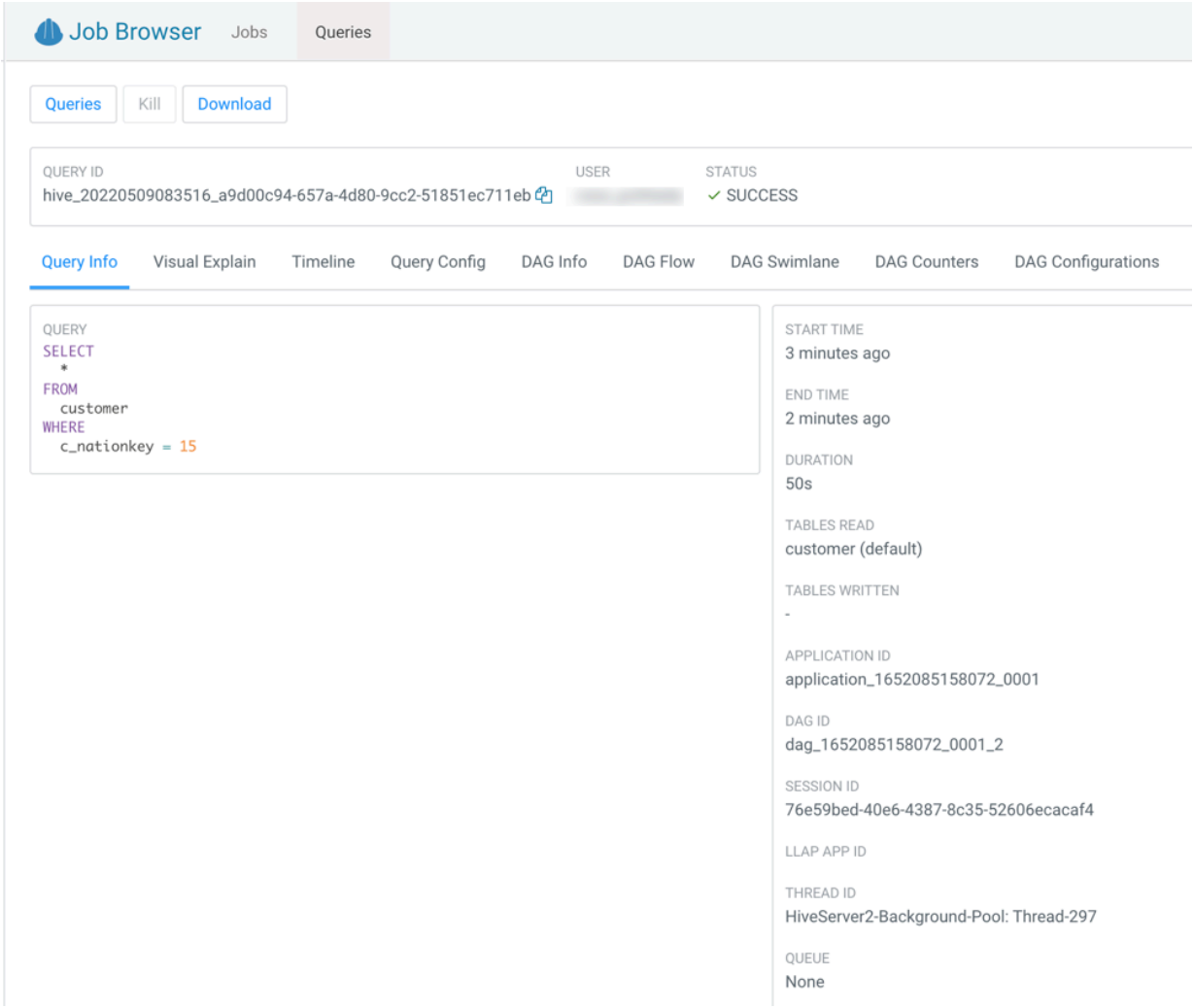

Viewing explain plan for a Hive query

The Visual Explain feature provides a graphical representation of the query execution plan. The Explain plan is read from right to left. It provides details about every stage of query execution.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Queries** tab and click on the query for which you want to view the query details.

4. Click on Visual Explain.

   The following image shows the **Visual Explain** tab on the Hue web interface:

5. (Optional) Click to download the query explain plan in JSON format.

Viewing Hive query timeline

The Timeline tab provides a visual representation of Hive performance logs and shows the time taken by each stage of the query execution.

About this task

Following are the stages in which a query is executed:

- Pre-execution and DAG construction: It is the first phase of query execution and is executed on the Hive engine. It constitutes the time taken to compile, parse, and build the Directed Acyclic Graph (DAG) for the next phase of the query execution.
- DAG submission: It is the second phase in which the DAG that was generated in Hive is submitted to the Tez engine for execution.
- DAG runtime: It shows the time taken by the Tez engine to execute the DAG.
- Post-execution: It is the last phase of query execution in which the files in S3/ABFS are moved or renamed.

Duration data about each phase are distilled into more granular metrics based on query execution logs.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Queries** tab and click on the query for which you want to view the query details.
4. Click on Timeline.

   The following image shows the **Timeline** tab on the Hue web interface:

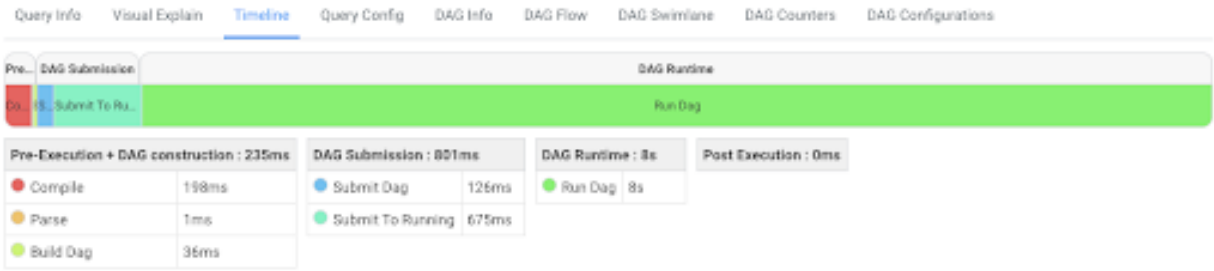

Viewing configurations for a Hive query

The Query Config tab provides the configuration properties and settings that are used in a Hive query. You can use this tab to verify that configuration property values align with your expectations.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.
   The **Job Browser** page is displayed.
3. Go to the **Queries** tab and click on the query for which you want to view the query details.
4. Click on Query Config.
   The following image shows the **Query Config** tab on the Hue web interface:

Query Info | Visual Explain | Timeline | Query Config | DAG Info | DAG Flow | DAG Swimlane

Config Name	Config Value
hadoop.security.group.mapping.ldap.posix.attr...	uidNumber
dfs.block.invalidate.limit	1000
yarn.admin.acl	*
hive.repl.dump.metadata.only.for.external.table	true
hive.exec.stagingdir	.hive-staging
hive.druid.rollup	true
yarn.federation.enabled	false
yarn.app.mapreduce.am.job.committer.cancel-...	60000
hive.druid.broker.address.default	localhost:8082
dfs.disk.balancer.max.disk.throughputInMBper...	10
dfs.qjournal.select-input-streams.timeout.ms	20000
hive.llap.io.orc.time.counters	true
hive.repl.retain.prev.dump.dir	false
hive.vectorized.execution.mapjoin.native.fast....	true
dfs.provided.aliasmap.inmemory.leveldb.dir	/tmp
yarn.nodemanager.process-kill-wait.ms	5000
yarn.minicluster.use-rpc	false
io.map.index.interval	128

Viewing DAG information for a Hive query

Directed Acyclic Graph (DAG) is created by the Hive engine every time you query the Hive Virtual Warehouse. The Hive SQL queries are compiled and converted into a Tez execution graph also known as a DAG. DAG is a collection of vertices where each vertex executes a fragment of the query or script. Hue provides a web interface to view detailed information about DAGs.

About this task

Directed connections between vertices determine the order in which they are executed. For example, the vertex to read a table must be run before a filter can be applied to the rows of that table. As another example, consider a vertex that reads a user table that is very large and distributed across multiple computers and multiple racks. Reading the table is achieved by running many tasks in parallel.

Important: The DAG information tabs (**DAG Info**, **DAG Flow**, **DAG Swimlane**, **DAG Counters**, **DAG Configurations**) are displayed only if the Tez engine is used for query execution. The Tez engine is typically utilized for complex queries.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Queries** tab and click on the query for which you want to view the query details.
4. Click DAG Info to see the DAG ID, DAG name, the status of the query, the time taken to execute the DAG, start time, and end time.

   The following image shows the **DAG Info** tab on the Hue web interface:

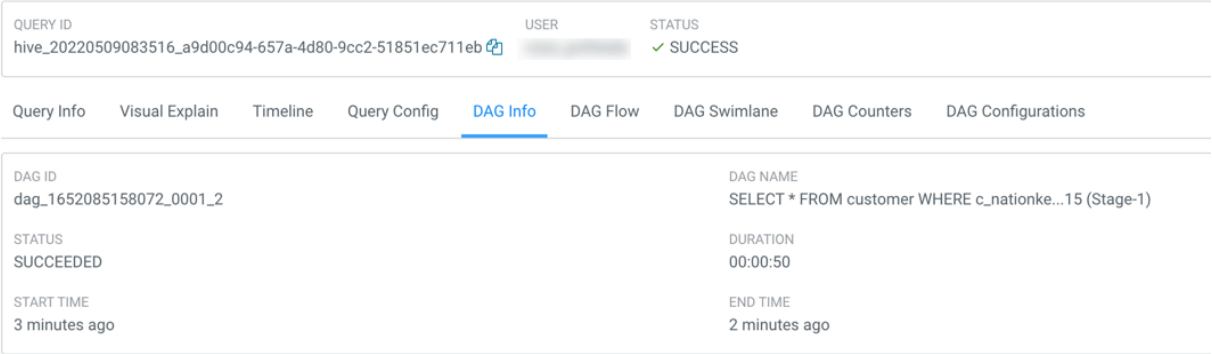

The following table lists and describes the status of the Tez job:

Status	Description
Submitted	The DAG is submitted to Tez but is not running
Running	The DAG is currently running
Succeeded	The DAG was completed successfully
Failed	The DAG failed to complete successfully
Killed	The DAG was stopped manually
Error	An internal error occurred when executing the DAG

5. Click DAG Flow to see the DAG in the form of a flowchart.

   You can gain insight into the complexity and the progress of executing jobs, and investigate the vertices that have failures or are taking a long time to complete.

   The following image shows the **DAG Flow** tab on the Hue web interface::

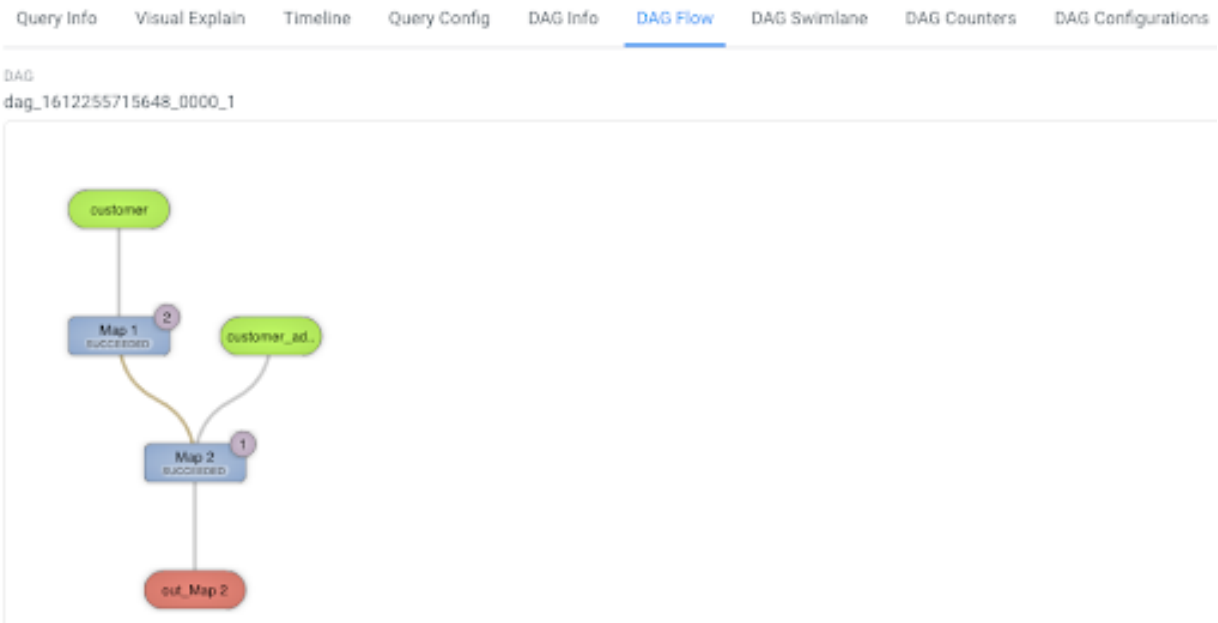

   Here, the input to vertices Map 1 and Map 2 are the tables displayed in green boxes. Next, Map 2 depends on the result set generated by Map 1. Map 2 is the last vertex in the DAG flow and after it completes its execution, the query output is written to a file in a filesystem such as S3 or ABFS.

   There are a few options to change the layout of the DAG flow. You can hide the input and the output nodes to view only the task vertices by clicking the Toggle source/sink visibility button. You can switch between the horizontal and vertical orientation by clicking the Toggle orientation button.

6. Click DAG Swimlane to see the DAG of the vertices against time.

   Each mapping and reducing task is a vertex. Each horizontal bar of the swimlane represents the total time taken by the vertex to complete the execution. The vertical lines indicate the time when the vertex was initialized, the time when the vertex started, the time when the first task started, the time when the last task was completed, and the time when the vertex finished its execution. When you mouse over the vertical line, the bubble displays the stage of the vertex execution and provides a timestamp. The vertical lines connecting two vertices denote the dependency of a vertex on another vertex.

   The following image shows the **DAG Swimlane** tab on the Hue web interface:

   In this example, Map 1 depends on the results of Map 5. Map 1 will finish its execution only when Map 5 finishes its execution successfully. Similarly, Reducer 2 depends on Map 1 to complete its execution.

   The consolidated timeline shows the percentage of time each vertex took to complete executing.

7. Click DAG Counters to see details such as the number of bytes read and written, number of tasks that initiated and ran successfully, amount of CPU and memory consumed, and so on.

   The **DAG Counters** tab provides a way to measure the progress or the number of operations that occur within a generated DAG. Counters are used to gather statistics for quality control purposes or problem diagnosis.

   The following image shows the **DAG Counters** tab on the Hue web interface:

Query Info | Visual Explain | Timeline | Query Config | DAG Info | DAG Flow | DAG Swimlane | DAG Counters | DAG Configurations

Group Name	Counter Name	DAG : dag_1652085158072_0001_2
org.apache.tez.common.counters.DAGCounter	NUM_SUCCEEDED_TASKS	54
org.apache.tez.common.counters.DAGCounter	TOTAL_LAUNCHED_TASKS	54
org.apache.tez.common.counters.DAGCounter	DATA_LOCAL_TASKS	54
org.apache.tez.common.counters.DAGCounter	AM_CPU_MILLISECONDS	5890
org.apache.tez.common.counters.DAGCounter	AM_GC_TIME_MILLIS	22
org.apache.tez.common.counters.FileSystem...	FILE_BYTES_WRITTEN	1074416910
org.apache.tez.common.counters.FileSystem...	S3A_BYTES_READ	6742239795
org.apache.tez.common.counters.FileSystem...	S3A_READ_OPS	1124
org.apache.tez.common.counters.TaskCounter	TASK_DURATION_MILLIS	759357
org.apache.tez.common.counters.TaskCounter	INPUT_RECORDS_PROCESSED	146519
org.apache.tez.common.counters.TaskCounter	INPUT_SPLIT_LENGTH_BYTES	12387638515
HIVE	CREATED_FILES	36
HIVE	RECORDS_IN_Map_1	150000000
HIVE	RECORDS_OUT_0	6003115
HIVE	RECORDS_OUT_OPERATOR_FIL_5	6003115
HIVE	RECORDS_OUT_OPERATOR_FS_7	6003115
HIVE	RECORDS_OUT_OPERATOR_SEL_6	6003115
HIVE	RECORDS_OUT_OPERATOR_TS_0	150000000

8. Click DAG Configurations to see the Tez configuration details for a query that has a DAG associated with it.

   The following image shows the **DAG Configurations** tab on the Hue web interface:

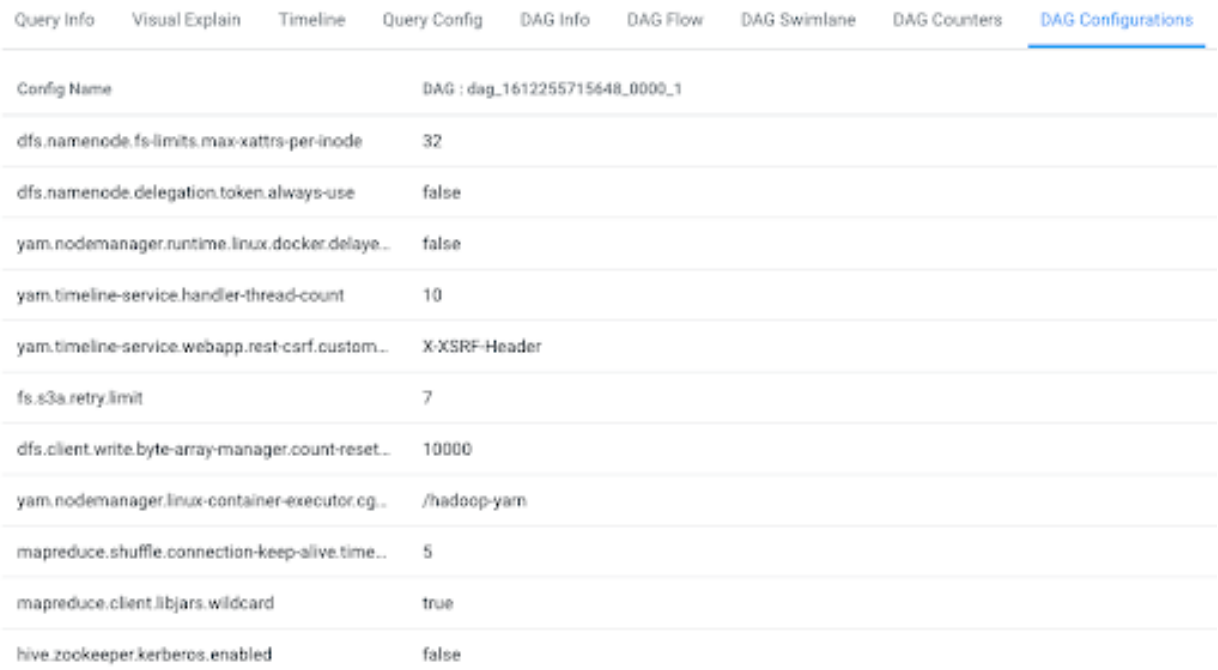

Viewing Impala query details

You can view Impala query details, query plan, execution summary, and query metrics on the new Impala Queries tab on the Job Browser page in Hue, and use this information to tune and optimize your queries. You can also view Impala query profiles on the Impala tab.

Viewing Impala query history

The Impala Queries tab on the Job Browser page in Hue displays all the queries that were run from various query interfaces, such as Impala-shell, Impyla, Hue, and other JDBC BI clients and tools.

About this task

Only Query Processor Administrators can view historical queries of all users to access history of queries run in that Impala Virtual Warehouse from the Hue Job Browser. Non-admin users can only view only their own queries.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Click Queries.

   The Impala queries that were run for the past seven days are displayed. You can select the time period for which you want to view the historical data.

   You can also search using the query ID, sort queries by various parameters such as duration, peak memory, and so on, and filter queries by their status.

Related Information

Adding Query Store Administrator users on Cloudera Data Hub

Viewing Impala query information

The Query Info tab in Hue provides information such as, the Impala query ID, the user who executed the query, the start time, the end time, the total time taken to execute the query, the coordinator that received the query, CPU time, rows produced, peak memory, and HDFS bytes read.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Impala Queries** tab and click on the query for which you want to view the query details.

   The following image shows the **Query Info** tab on the Hue web interface:

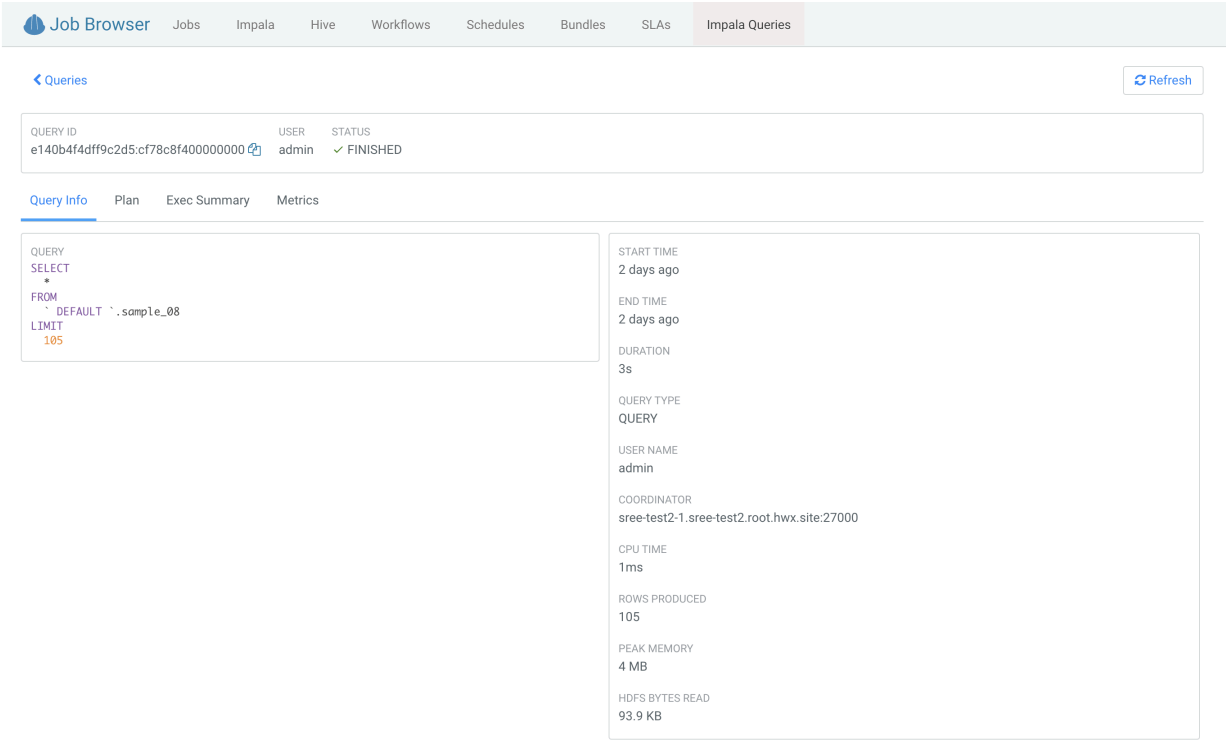

Viewing the Impala query execution plan

The query execution plan in Hue provides details on how the query will be executed, the operators involved, and other information before the query is submitted to the Impala engine.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.

3. Go to the **Impala Queries** tab and click on the query for which you want to view the execution plan.

   The following image shows the **Plan** tab on the Hue web interface:

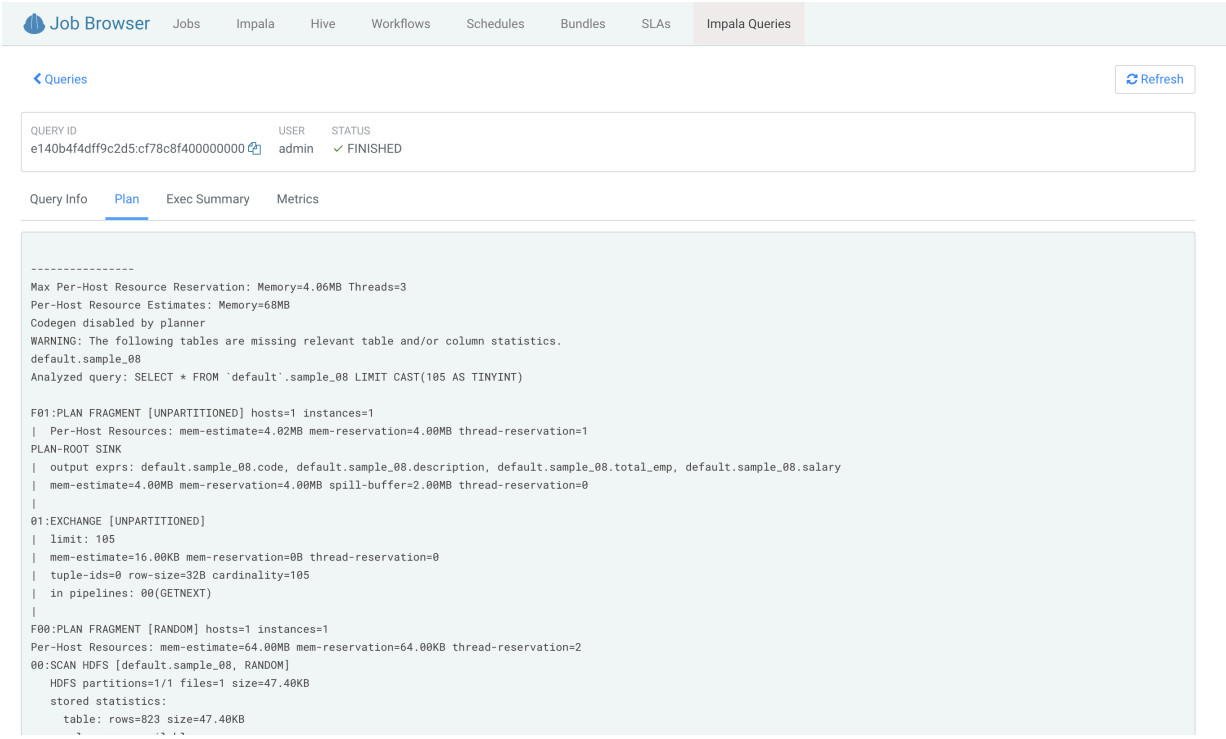

Viewing the Impala query metrics

You can view detailed, aggregated metrics for various counters such as hdfs_bytes_read, memory_per_node_peak, thread_cpu_time, and so on, on the Metrics tab in Hue.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.

3. Go to the **Impala Queries** tab and click on the query for which you want to view the query metrics.

   The following image shows the **Metrics** tab on the Hue web interface:

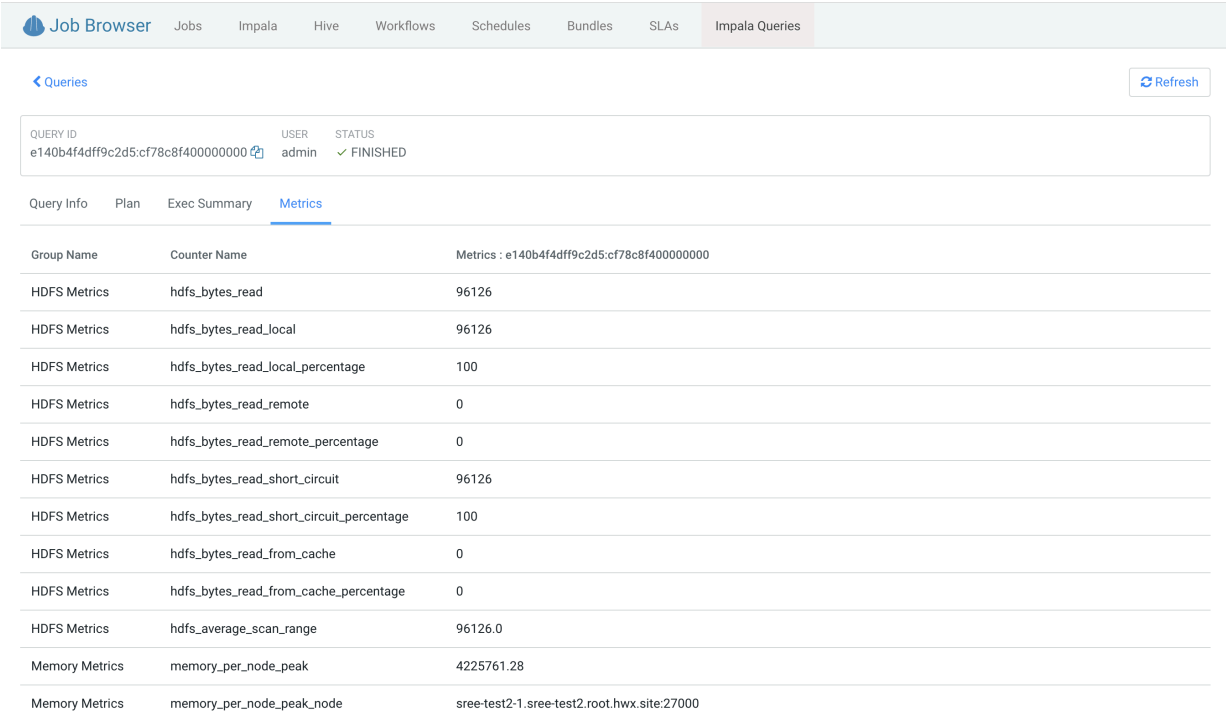

Group Name	Counter Name	Metrics : e140b4f4dff9c2d5:cf78c8f400000000
HDFS Metrics	hdfs_bytes_read	96126
HDFS Metrics	hdfs_bytes_read_local	96126
HDFS Metrics	hdfs_bytes_read_local_percentage	100
HDFS Metrics	hdfs_bytes_read_remote	0
HDFS Metrics	hdfs_bytes_read_remote_percentage	0
HDFS Metrics	hdfs_bytes_read_short_circuit	96126
HDFS Metrics	hdfs_bytes_read_short_circuit_percentage	100
HDFS Metrics	hdfs_bytes_read_from_cache	0
HDFS Metrics	hdfs_bytes_read_from_cache_percentage	0
HDFS Metrics	hdfs_average_scan_range	96126.0
Memory Metrics	memory_per_node_peak	4225761.28
Memory Metrics	memory_per_node_peak_node	sree-test2-1.sree-test2.root.hwx.site:27000

Viewing Impala profiles in Hue

When Impala executes any query, it captures the runtime details of the execution in a query profile. You can now view the Impala query profile from Hue's Job Browser page.

About this task

Query Processor Administrators can view query details of all users unlike the non-admin users who can view query details only for their queries. This enables the Query Processor Administrators to provide a comprehensive report on past queries and running queries.

Procedure

1. Go to the Cloudera Data Warehouse web interface and open Hue from your Impala Virtual Warehouse.
2. Click Jobs on the left-assist panel to go to the **Job Browser** page and then click on the Impala tab.
3. Click on the query for which you want to view the Impala query profile.

   The query execution details are displayed.
4. Click on the Profile tab.

Terminating Hive queries

If a query is running for longer than expected, or you have accidentally triggered it, then you can stop the query to free up the resources. Hue also allows you to stop multiple queries at once.

About this task

Note: This feature is available only for Hive queries. Only admin users or Hue superusers can stop running queries.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Queries** tab.

   A list of queries that were run is displayed.
4. Select the queries that you want to stop and click Kill.

Comparing Hive and Impala queries in Hue

You can compare two queries to know how each query is performing in terms of speed and cost-effectiveness. Hue compares various aspects of the two queries, based on which you can identify what changed between the executions of those two queries, and you can debug performance-related issues between different runs of the same query.

About this task

The query comparison report provides you a detailed side-by-side comparison of your queries.

For Hive queries, it includes recommendations for optimizing each query, metadata about the queries, visual explain for each query, query timeline, query configuration, Directed Acyclic Graph (DAG) information, DAG flows, DAG swimlanes, DAG counters, and DAG configurations.

For Impala queries, the query comparison report includes query details, execution plan details, and the aggregated metrics for both the queries and provides a variance between the two.

Procedure

1. Log in to the Hue web interface.
2. Click Jobs from the left assist panel.

   The **Job Browser** page is displayed.
3. Go to the **Queries** tab.

   A list of queries that were run is displayed.

4. Select the two queries you want to compare and click Compare.

   Query comparison report for Hive queries:

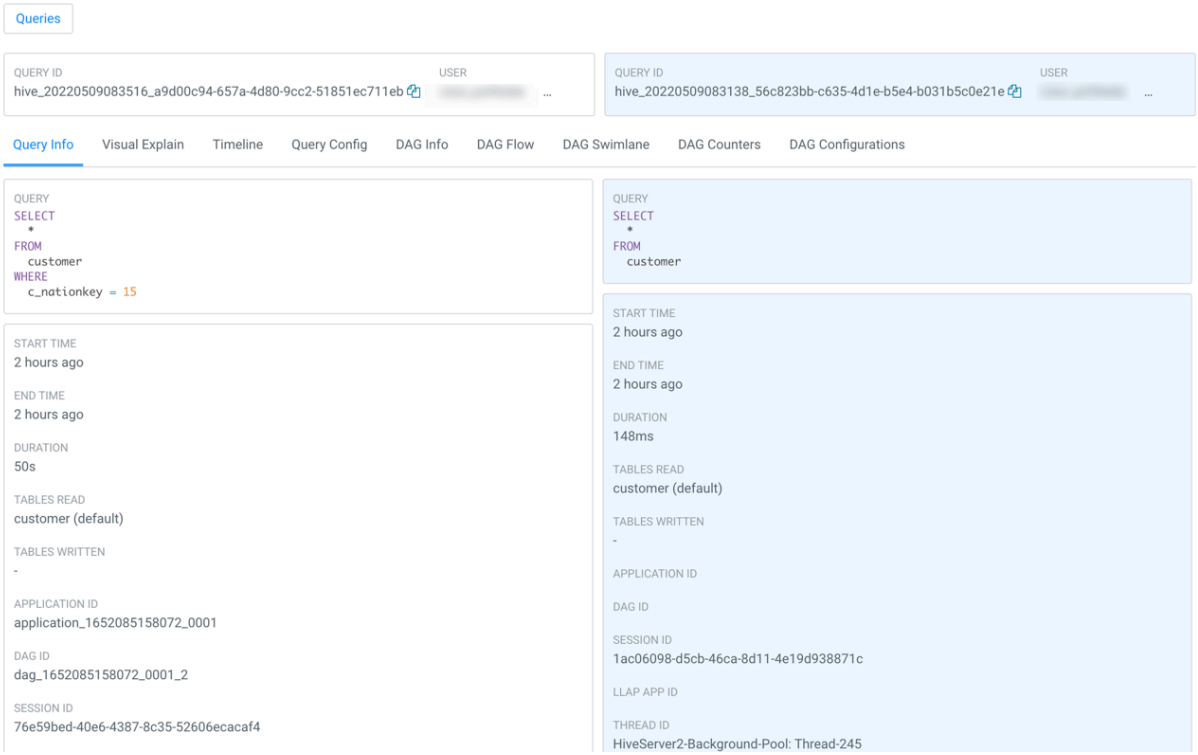

   Query comparison report for Impala queries:

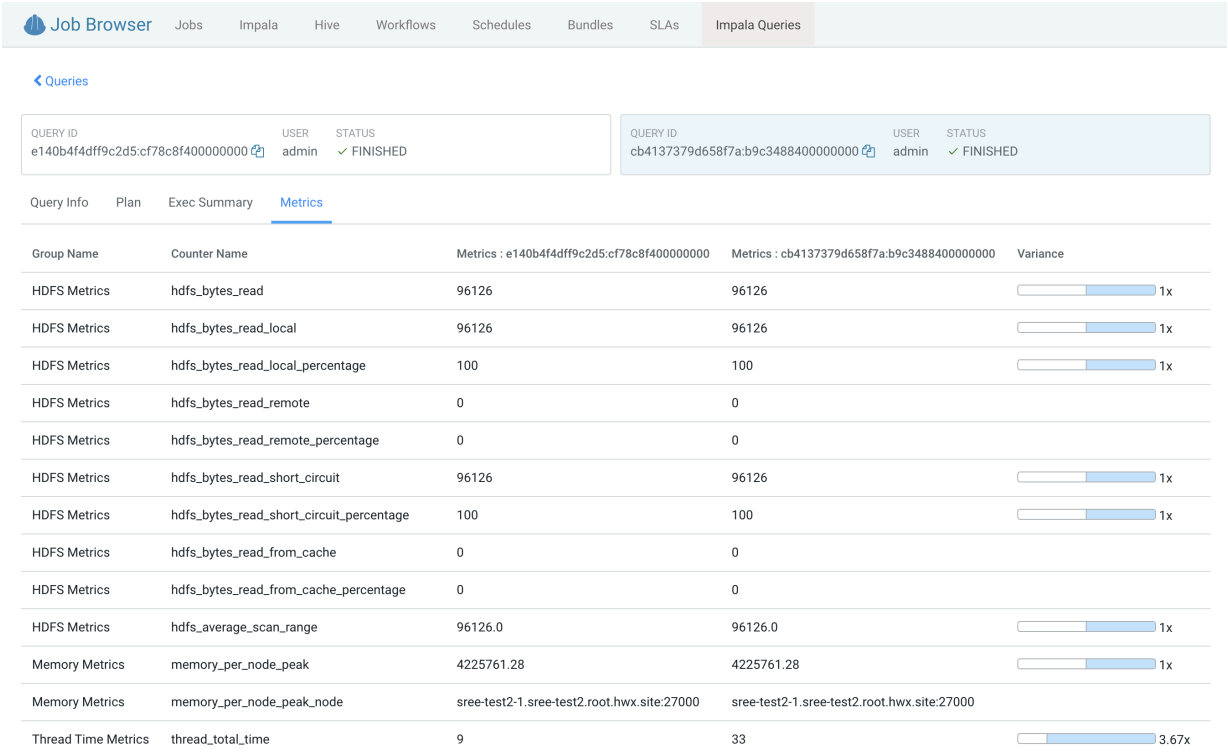

Group Name	Counter Name	Metrics : e140b4f4dff9c2d5:cf78c8f400000000	Metrics : cb4137379d658f7a:b9c3488400000000	Variance
HDFS Metrics	hdfs_bytes_read	96126	96126	1x
HDFS Metrics	hdfs_bytes_read_local	96126	96126	1x
HDFS Metrics	hdfs_bytes_read_local_percentage	100	100	1x
HDFS Metrics	hdfs_bytes_read_remote	0	0	
HDFS Metrics	hdfs_bytes_read_remote_percentage	0	0	
HDFS Metrics	hdfs_bytes_read_short_circuit	96126	96126	1x
HDFS Metrics	hdfs_bytes_read_short_circuit_percentage	100	100	1x
HDFS Metrics	hdfs_bytes_read_from_cache	0	0	
HDFS Metrics	hdfs_bytes_read_from_cache_percentage	0	0	
HDFS Metrics	hdfs_average_scan_range	96126.0	96126.0	1x
Memory Metrics	memory_per_node_peak	4225761.28	4225761.28	1x
Memory Metrics	memory_per_node_peak_node	sree-test2-1.sree-test2.root.hwx.site:27000	sree-test2-1.sree-test2.root.hwx.site:27000	
Thread Time Metrics	thread_total_time	9	33	3.67x

Starting the SQL AI Assistant in Hue

A SQL AI Assistant has been integrated into Hue with the capability to leverage the power of Large Language Models (LLMs) for various SQL tasks. It helps you to create, edit, optimize, fix, and succinctly summarize queries using natural language and makes SQL development faster, easier, and less error-prone. You can also generate comments and insert them into your queries to improve readability.

About this task

Attention: The SQL AI Assistant operates only on the database that you have selected in the Hue editor, and not necessarily on the one that is displayed on the left-assist bar.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant.

Procedure

1. Log in to the Hue web interface.
2. Click Assistant on the Hue SQL editor:

Results

The following options are displayed:

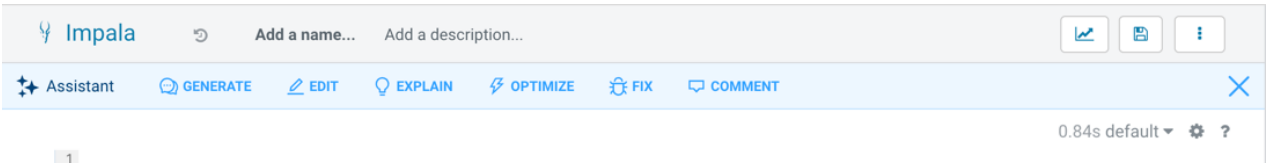

Generating SQL from natural language in Hue

The SQL AI Assistant in Cloudera Data Warehouse helps you to generate SQL queries by entering a prompt in natural language. You can then insert the generated SQL in the Hue SQL editor and run it as usual.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant.

Procedure

1. Log in to the Hue web interface.

2. Click Assistant on the Hue SQL editor:

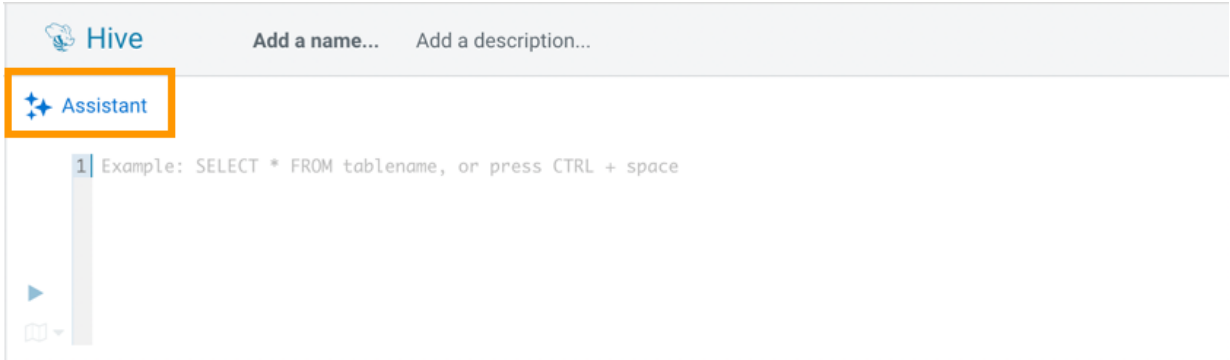

3. Click GENERATE.

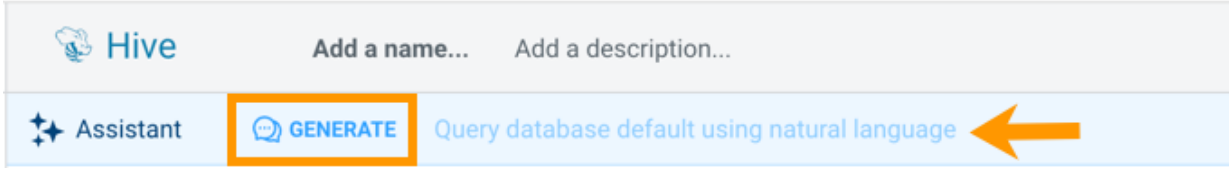

A SQL query is generated based on your input prompt. Click Insert to insert the query into the editor and run it.

Editing the query in natural language in Hue

You can edit a query in natural language and generate it again to fine-tune your results using the Hue SQL AI Assistant. You can then insert the SQL query into the editor and run it as usual.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant.

Procedure

1. Log in to the Hue web interface.
2. Click Assistant on the Hue SQL editor:

3. Click EDIT.

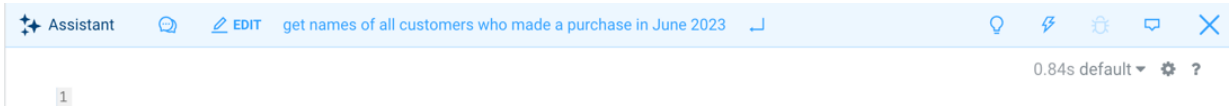

Select the query from the list and edit it as needed. Then press enter to regenerate the SQL.

Getting an explanation of a SQL query in natural language in Hue

You can use the SQL AI Assistant to breakdown and understand a complex SQL query in natural language.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant.

Procedure

1. Log in to the Hue web interface.
2. Insert a SQL query in the Hue editor that you would like to understand in natural language.
3. Click Assistant on the Hue SQL editor:

4. Click EXPLAIN.

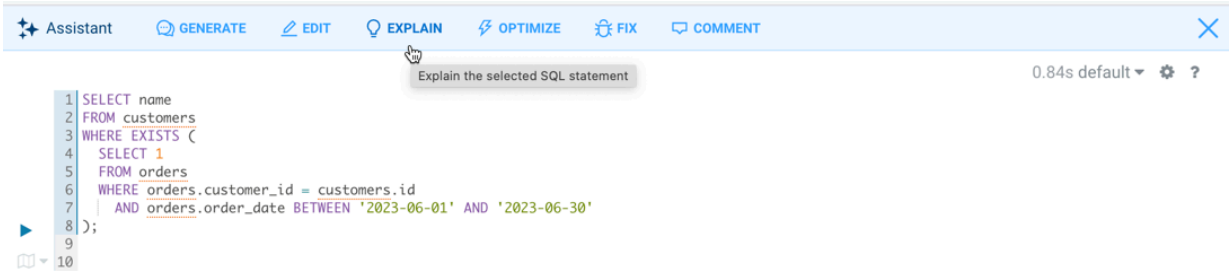

The LLM generates the explanation of the SQL query.

Optimizing a query in Hue

You can use the SQL AI Assistant to optimize a SQL query. Hue identifies the issues in the source query, optimizes it, and provides the optimized version of the SQL query. Hue also summarizes the issues and how it optimized the query in natural language.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant.

Procedure

1. Log in to the Hue web interface.
2. Insert a SQL query in the Hue editor that you would like to optimize.
3. Click Assistant on the Hue SQL editor:

4. Click OPTIMIZE.

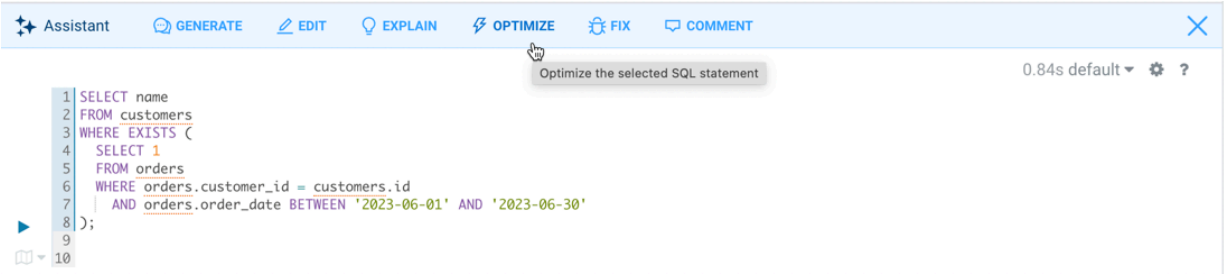

Hue displays the original and the optimized SQL query side-by-side. It also provides an explanation of the issues in the original query and how it was optimized.

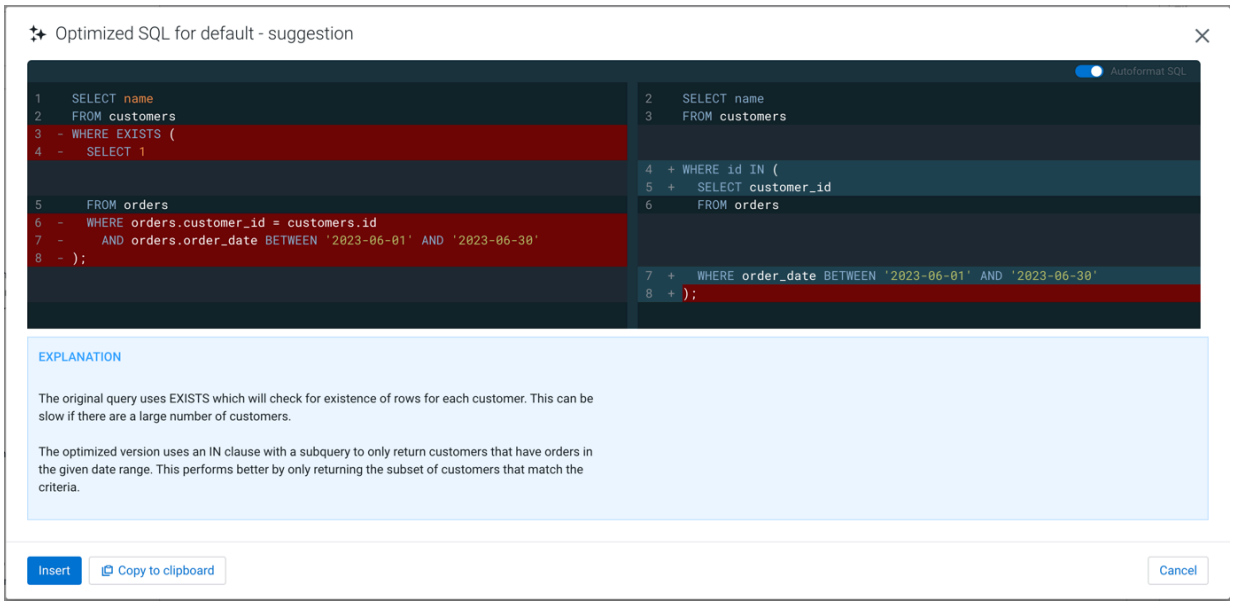

Fixing a query in Hue

You can use the SQL AI Assistant to fix a broken SQL query. Hue identifies the issues in SQL syntax and provides the corrected version.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant.

Procedure

1. Log in to the Hue web interface.
2. Insert a SQL query in the Hue editor that you would like to fix.

3. Click Assistant on the Hue SQL editor:

4. Click FIX.

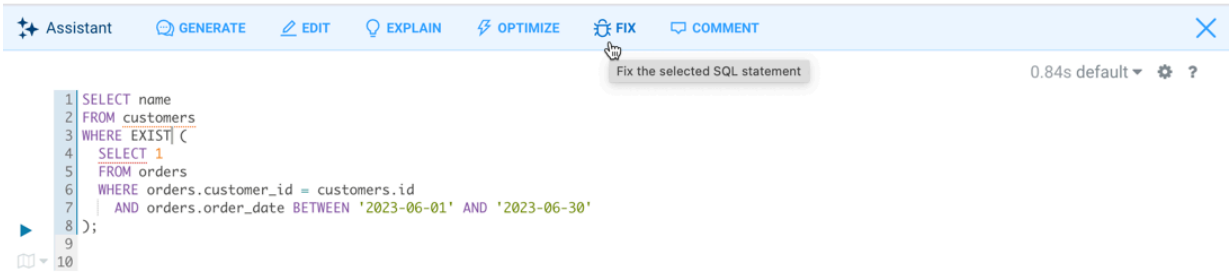

Hue displays the original and the fixed SQL query in a side-by-side comparison.

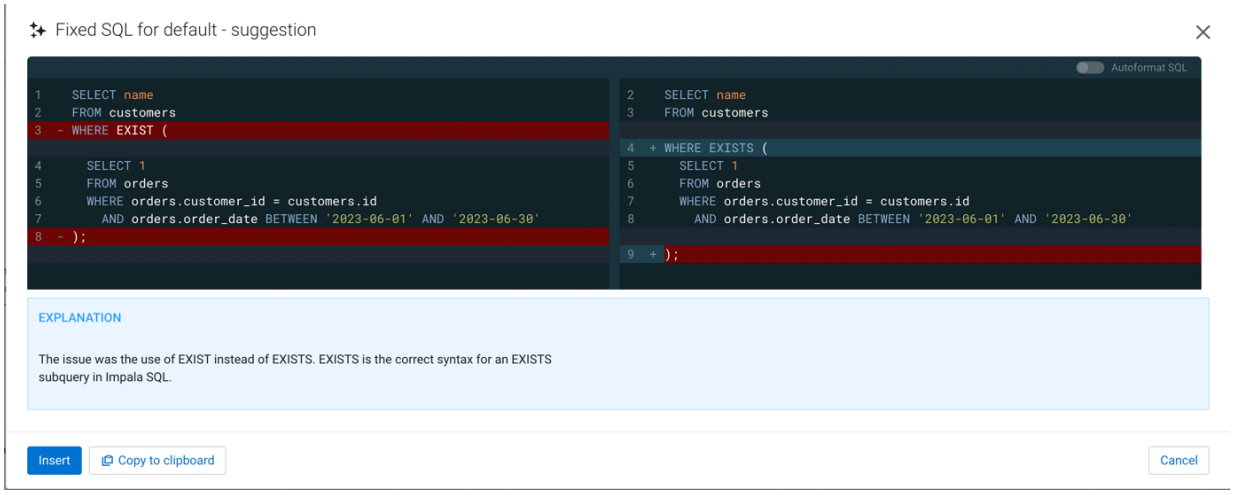

Click Insert to insert the fixed query in the Hue editor and run it.

Generating a comment for a query in Hue

The SQL AI Assistant can generate a comment explaining what SQL query does. You can insert it into the query to improve readability.

Before you begin

Your administrator must have configured and set up the required infrastructure for you to use the SQL AI Assistant. See About setting up the Hue SQL AI Assistant.

Procedure

1. Log in to the Hue web interface.
2. Insert a SQL query in the Hue editor for which you want to generate a comment.

3. Click Assistant on the Hue SQL editor:

4. Click COMMENT.

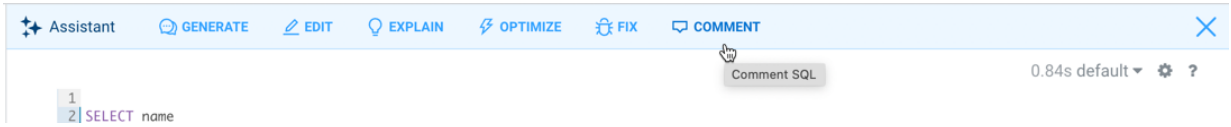

The SQL AI Assistant generates a detailed comment for the input SQL query.

Click Insert to insert the comment into the query.

Enabling stored procedures for Hive in Cloudera Data Hub

To create, edit, and drop procedures and functions that are written in Hive Hybrid Procedural SQL (HPL/SQL) using the Hue query editor, you must enable the hplsql option in the Hue Advanced Configuration Snippet.

About this task

Note: Hue enables you to switch between Hive and HPL/SQL interpreters. By default, the regular Hive interpreter is enabled when you add the Hue service to your cluster. To enable the HPL/SQL interpreter, you must update Hue's Advanced Configuration Snippet in Cloudera Manager. However, updating Hue's Advanced Configuration Snippet overrides the default configuration. Therefore, to use both Hive and HPL/SQL interpreters, you must enable both by updating Hue's Advanced Configuration Snippet.

Procedure

1. Log in to the Cloudera web interface as an Administrator.
2. Go to Cloudera Management Console Environments , select your environment and go to the Data Lake tab.
3. Click on the CM URL to open Cloudera Manager.
4. Go to Clusters Hue Configuration and add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[notebook]
  [[interpreters]]
    [[[hive]]]
      name=Hive
      interface=hiveserver2
    [[[hplsql]]]
      name=Hplsql
      interface=hiveserver2
```

5. Click Save Changes.
6. Restart the Hue service.
7. Go to Clusters Hive on Tez Configuration and add the following property name and its value in the Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml field:

   Name: hive.security.authorization.sqlstd.confwhitelist.append

   Value: QUERY_EXECUTOR|HPLSQL
8. Click Save Changes.
9. Restart the Hive on Tez service.

How to run a stored procedure from Hue in Cloudera Data Hub

HPL/SQL allows you to implement business logic using variables, expressions, flow-of-control statements, and iterations. HPL/SQL makes SQL-on-Hadoop more dynamic. You can leverage your existing procedural SQL skills, and use functions and statements to make your typical ETL development more productive. Hue provides a smart interface to run stored procedures.

Note: This feature is available only for Hive queries.

The following example creates a procedure and returns records by passing a cursor:

```
print 'Hello world';/
CREATE PROCEDURE greet(name STRING)
BEGIN
  PRINT 'Hello ' || name;
END;/
CREATE PROCEDURE even(cur OUT SYS_REFCURSOR)
BEGIN
  OPEN cur FOR
SELECT n FROM NUMBERS
WHERE MOD(n, 2) == 0;
END;/
```

```
CREATE PROCEDURE set_message(IN name STRING, OUT result STRING)
BEGIN
 SET result = 'Hello, ' || name || '!';
END;
-- Call the procedure and print the results
DECLARE str STRING;
CALL set_message('world', str);
PRINT str;
```

Attention: In the hplsql mode, you must terminate the commands using the forward slash (/). The semicolon (;) is used throughout procedure declarations and can no longer be relied upon to terminate a query in the editor.

Note: HPL/SQL does not support all types of Hive statements, such as JOIN or EXPLAIN. Refer to the HPL/SQL Reference for more information.

Related Information

Enabling stored procedures for Hive in Cloudera Data Hub

Using SQL to query HBase from Hue

Hue provides a simple SQL interface to create and manipulate SQL tables that are stored in HBase, and define and manipulate views on HBase tables using Apache Phoenix in addition to HBase shell and database API.

Cloudera does not recommend manipulating Phoenix tables from HBase as this can lead to data loss.

The SQL connector is shipped with Hue so that you do not have to download and configure it yourself.

Following are some examples to create and manipulate the Phoenix SQL tables from the Hue editor:

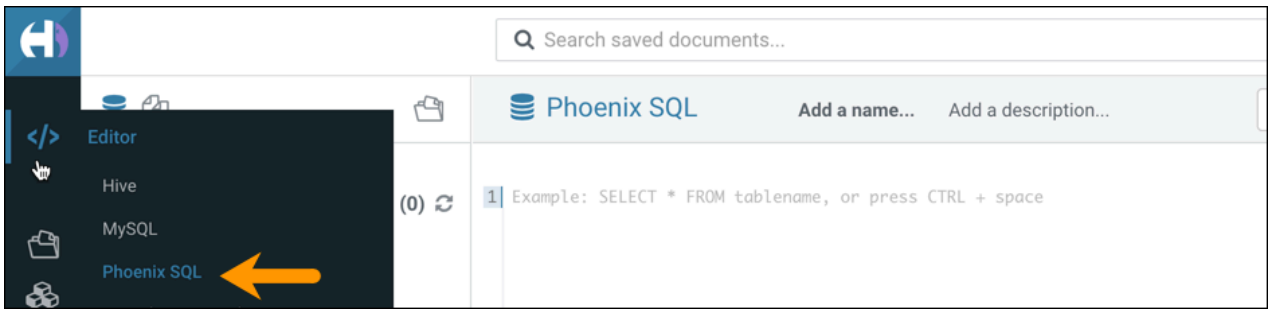

Creating a table

```
CREATE TABLE IF NOT EXISTS Company (company_id INTEGER PRIMARY KEY, name VAR
CHAR(225));
```

Upserting values in the table

```
UPSERT INTO Company VALUES(1, 'Cloudera');
UPSERT INTO Company VALUES(2, 'Apache');
```

Querying the table

```
SELECT * FROM Company;
```

Deleting a record

```
DELETE FROM Company WHERE COMPANY_ID=1;
```

Dropping the table

```
DROP TABLE Company;
```

Related Information

Apache Phoenix SQL command reference

Apache Phoenix Grammar

Querying existing HBase tables

To use SQL for querying data from existing HBase tables, you must create a view in Phoenix pointing to the HBase table.

To map the existing tables to the views, run the following statement using the Phoenix editor on the Hue web interface:

```
CREATE VIEW if not exists "[***HBASE-TABLE-NAME***]" ( pk VARCHAR PRIMARY KE
Y, val VARCHAR );
```

Related Information

Mapping a Phoenix table to an existing HBase table

Enabling the SQL editor autocompleter

Autocompleter provides finely tuned SQL suggestions for Hive and Impala dialects while you enter queries into the editor window. See Brand new Autocompleter for Hive and Impala in the Hue blog.

About this task

Autocompleter is enabled by default. To manually enable or disable it, open the editor configuration panel and edit settings as follows:

Procedure

1. Log in to Hue and go to either the Hive or Impala editor.
2. Place your cursor in the editor window and then use one of the following keyboard shortcuts to open the editor configuration panel:
   - On a Mac system, use the Command key followed by a hyphen and then a comma:

     Command-,
   - On a Windows system, use the Ctrl key followed by a hyphen and then a comma:

     Ctrl-,

   Tip: Type a question mark (?) anywhere but in the active editor window to open a menu of editor keyboard shortcuts.

3. To enable autocompletion, check the box adjacent to Enable Autocompleter. When you check Enable Autocompleter, Enable Live Autocompletion is automatically enabled as well. Place your cursor in the editor window to close the configuration panel.

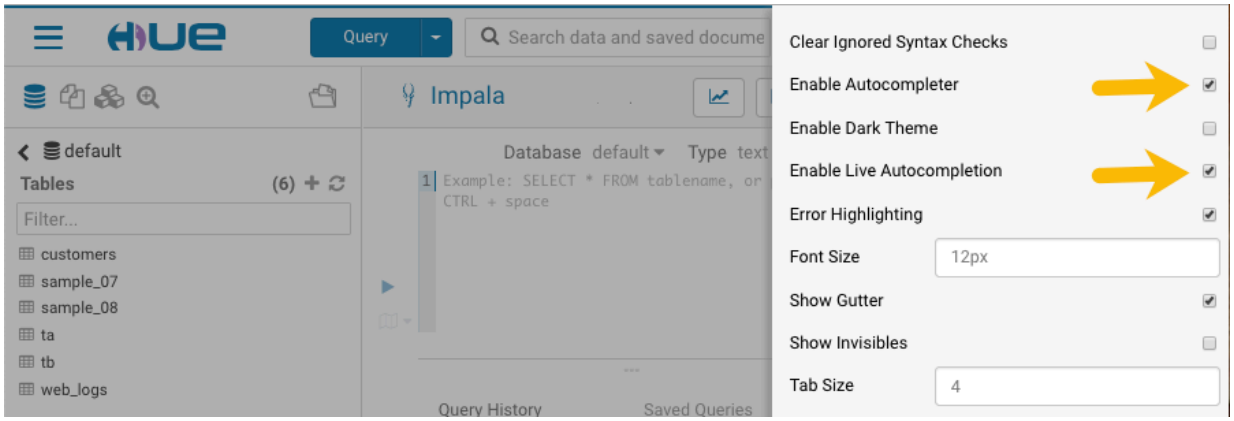

4. To disable autocompletion:
   - Uncheck Enable Live Autocompletion but leave Enable Autocompleter checked, and then place your cursor in the editor window to close the configuration panel. This disables live autocompletion, but if you want to use autocompletion while building your queries in the editor, enter the following key stroke sequence to activate autocompletion: Ctrl + Space Key
   - Uncheck both Enable Autocompleter and Enable Live Autocompletion, and then click in the editor to close the configuration panel. This disables all autocompletion functionality.

Rerunning a query from the Job Browser page

You may need to reconfigure the query parameters post a query execution, based on the results. You can modify and rerun your historical queries from the Job Browser page.

About this task

Note: Only Query Processor Administrators can view and rerun historical queries of all users from the **Job Browser** page. Non-admin users can view and rerun only their queries.

Procedure

1. Log in to the Hue web interface.
2. Select Jobs from the left assist panel and click Queries on the **Job Browser** page.
3. Click on the query that you want to rerun.

   The query details are displayed.
4. Click Re Execute.

   You are redirected to the query editor where you can make the required changes to the pre-populated query.
5. Click to submit the query.

Using governance-based data discovery

Hue can use the metadata tagging, indexing, and search features available in Apache Atlas data management. After integrating Hue with Atlas, classifications and indexed entities can be accessed and viewed in Hue. This topic shows you how to use metadata classifications in Hue.

Integration between Hue and Atlas is enabled by default, but if your administrator has disabled it, it must be re-enabled before you can use governance-based data discovery.

You can create tags to classify your data both from Atlas and Hue.

Searching metadata tags

The SQL Editor in Hue provides a search text box where you can search on the metadata tags or classifications that are associated with your databases, tables, and columns.

About this task

You can search for tags or classifications in either the Hive or the Impala editors.

Note: On clusters that use Apache Ranger for role-based access control, the Search mechanism does not display counts of popular values. Ranger ensures that Hue users can view only entities to which their user role (as configured and managed by Ranger) has been granted specific permissions.

Procedure

1. Go to Query Editor Impala or Hive.
2. To locate the tags or classifications in Apache Atlas, in the metadata search box located just to the right of the Query drop-down menu, type a tag: or classification: facet followed by its name. For example, type classification: wine as shown in the following image:

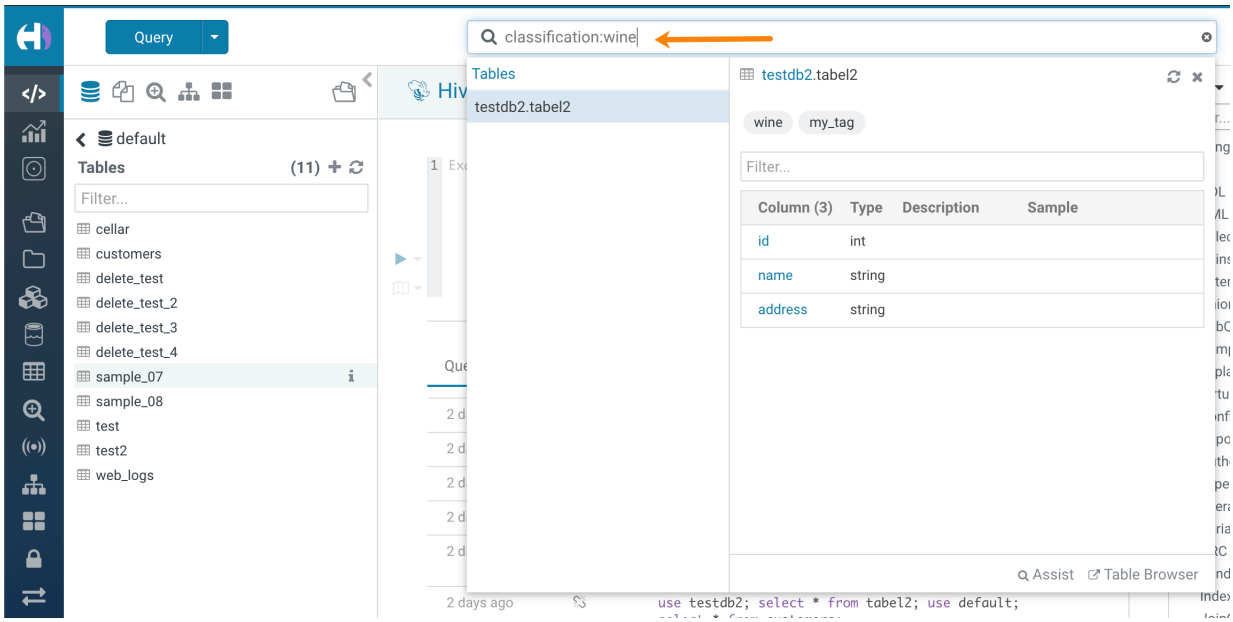

After you type the search facet and the tag or classification name in the search box, the *<DATABASE>.<TABLE>* where the tag or classification is found is returned. Click the *<DATABASE>.<TABLE>* to view the tags and classifications that have been defined for it.

Using Amazon S3 with Hue

Hue can read to and write to an Amazon S3 bucket.

Note:

You can use the S3 browser to store and retrieve data from Amazon S3 buckets into Hue and vice versa. If your Cloudera cluster is secured using Knox, then the Hue users will be able to access the S3 folders as per the access rules set on the individual folders. The access permissions are governed by IDBroker.

In Cloudera, you can enable the S3 File Browser in Hue in the following three ways:

- With IDBroker
- Without IDBroker
- With Ranger Authorization Service

Note: If you have enabled RAZ while registering your AWS environment with Cloudera, then Hue uses RAZ as the default mechanism for enabling the S3 File Browser.

Enabling S3 browser for Hue configured with IDBroker

You can access the S3 buckets from Hue to upload files and tables to S3 and import CSV files as tables directly in Hue by enabling the S3 browser on the Hue user interface.

Before you begin

If you have set up authentication using Knox IDBroker on your cluster, then Hue automatically detects and uses the IDBroker mappings from your cluster's core-site.xml file. Verify that the following property is present in the core-sit e.xml file:

```
<property>
  <name>fs.s3a.ext.cab.address</name>
  <value>https://<IDBROKERURL>:8444/gateway</value>
</property>
```

This property is automatically appended to the core-site.xml file when you enable Knox IDBroker on your cluster.

Procedure

1. Sign in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration .
3. Enter the following in the Hue Server Advanced Configuration Snippet (Safety Valve) for hue_safety_valve_server.ini field:

   ```
   [desktop]
   # Remove the file browser from the blacklisted apps.
   # Tweak the app_blacklist property to suit your app configuration.
   app_blacklist=spark,zookeeper,hive,hbase,search,oozie,jobsub,pig,sqoop,sec
   urity
   [aws]
   has_iam_detection=true
   [[aws_accounts]]
   [[[default]]]
   region=[***AWS-REGION***]
   # Set a particular S3 bucket as the default
   [filebrowser]
   remote_storage_home=s3a://[***S3-BUCKET-NAME***]
   ```

   The custom configuration is stored in the hue_safety_valve_server.ini file.
4. Click Save Changes.
5. Restart the Hue service.

   The S3 file browser icon appears on the left Assist panel as well as on the left navigation bar on the Hue web interface.

What to do next

You must manually grant the following application permission to non-admin users and groups for them to be able to view and access S3 File Browser in Hue: filebrowser.s3_access:Access to S3 from filebrowser and filepicker..

You must also add the Cloudera users and groups to IAM role mappings.

Related Information

Granting permission to access S3 and ABFS File Browser in Hue

Adding Cloudera user/group to IAM role mappings

Enabling S3 browser for Hue configured without IDBroker

You can access the S3 buckets from Hue to upload files and tables to S3 and import CSV files as tables directly in Hue by enabling the S3 browser. You must specify the AWS access key and secret access key information along with the AWS region in the Hue Server Advanced Configuration Snippet if you have not set up authentication using Knox IDBroker.

Procedure

1. Sign in to Cloudera Manager as an Administrator.
2. Go to Clusters Hue service Configuration .
3. Enter the following in the Hue Server Advanced Configuration Snippet (Safety Valve) for hue_safety_valve_server.ini field:

```
[desktop]
# Remove the file browser from the blacklisted apps.
# Tweak the app_blacklist property to suit your app configuration.
app_blacklist=spark,zookeeper,hive,hbase,search,oozie,jobsub,pig,sqoop,sec
urity
[aws]
[[aws_accounts]]
[[[default]]]
access_key_id=[***AWS-ACCESS-KEY***]
secret_access_key=[***SECRET-ACCESS-KEY***]
region=[***AWS-REGION***]

# Set a particular S3 bucket as the default
[filebrowser]
remote_storage_home=s3a://[***S3-BUCKET-NAME***]
```

   The custom configuration is stored in the hue_safety_valve_server.ini file.
4. Click Save Changes.
5. Restart the Hue service.

   The S3 file browser icon appears on the left Assist panel as well as on the left navigation bar on the Hue web interface.

What to do next

You must manually grant the following application permission to non-admin users and groups for them to be able to view and access S3 File Browser in Hue: filebrowser.s3_access:Access to S3 from filebrowser and filepicker.

Related Information

Granting permission to access S3 and ABFS File Browser in Hue

Enabling S3 File Browser for Hue with RAZ in DataHub

Hue offers you the capability to browse S3 buckets, upload files to S3, and create tables by importing files from S3. With RAZ, you can grant fine-grained access to per-user home directories and other directories within the S3 bucket using Ranger policies.

About this task

Hue administrators can create home directories for users by selecting the Create home directory option on the user's profile in Hue. If fine-grained authorization is enabled to access S3 buckets, then user home directories are automatically created when a user logs into Hue. You can disable automatic creation of user directories.

Before you begin

1. Register an AWS environment with the Enable Ranger authorization for AWS S3 option enabled. You can use the Cloudera web interface or the Cloudera CLI to complete this task.
2. Create a Cloudera Data Hub cluster with Data Engineering or Data Mart cluster template.
3. Create the following Ranger policies:
   - Hadoop SQL policy (all - database, table, column, all - url).

     Note: You must grant permissions to individual users or groups in these Ranger policies. To grant permissions to all users, you can specify {USER} in the Permission section.
   - S3 (cm_S3) policy (Default: User Home).

     You must specify the bucket name in the S3 Bucket field and the directory path in the Path field of the cm_S3 Ranger policy.
4. Grant appropriate permissions to the users in Cloudera User Management Service (UMS). For example, EnvironmentUser.

Procedure

1. Log in to the Cloudera web interface as an Administrator.
2. Go to Management Console Environments , select your environment and go to the Data Lake tab.
3. Click on the CM URL to open Cloudera Manager.
4. Go to Clusters Ranger RAZ service Instances RAZ server Processes and note the value of the fs.s3a.ext.raz.rest.host.url property from the core-site.xml file. You need this to specify the value of the api_url property in the Hue configuration.
5. Go to Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini and add the following lines:

```
[desktop]
app_blacklist=spark,zookeeper,hbase,impala,search,sqoop,security,pig
[[raz]]
is_enabled=true
api_url=[***fs.s3a.ext.raz.rest.host.url***]
[aws]
has_iam_detection=false

[[aws_accounts]]
[[[default]]]
region=[***AWS-REGION***]
host=s3.[***AWS-REGION***].amazonaws.com
allow_environment_credentials=false

[filebrowser]
remote_storage_home=s3a://[***S3-BUCKET-NAME***]/user
```

   (RAZ-HA) You can specify a comma-separated list of URLs in the api_url property. For example:

```
api_url=https://[***INSTANCE-1***]:6082/,https://[***INSTANCE-2***]:6082/
```

6. Click Save Changes.
7. Restart the Hue service.

Results

S3 File Browser icon is added to the left assist pane on the Hue web interface. When you click on the S3 File Browser icon, you are redirected to your user home directory.

Related Information

Introduction to RAZ on AWS environments

Disabling the automatic creation of user home directories

Using Azure Data Lake Storage Gen2 with Hue

Hue can read to and write to an Azure Data Lake Storage (ADLS) Gen2.

Note:

You can use the ABFS file browser to store and retrieve data from ADLS Gen2 into Hue and vice versa. If your Cloudera cluster is secured using Knox, then the Hue users will be able to access the ADLS as per the access rules set on the individual components. The access permissions are governed by IDBroker.

Only Hue superusers can view and access the ABFS file browser.

In Cloudera, you can enable the ABFS File Browser in Hue in the following three ways:

- With IDBroker
- Without IDBroker
- With Ranger Authorization Service

Enabling ABFS file browser for Hue configured with IDBroker

You can directly access the Azure file storage from Hue to upload files and tables to Azure file storage and import CSV files as tables directly in Hue by enabling the ABFS file browser on the Hue user interface.

Before you begin

If you have set up authentication using Knox IDBroker on your cluster, then Hue automatically detects and uses the IDBroker mappings from your cluster's core-site.xml file. Verify that the following property is present in the core-sit e.xml file:

```
<property>
  <name>fs.azure.ext.cab.address</name>
  <value>https://<IDBROKERURL>:8444/gateway</value>
</property>
```

This property is automatically appended to the core-site.xml file when you enable Knox IDBroker on your cluster.

You can also check whether there is a mapped role for the group(s) associated with the authenticated user by going to Cloudera Management Console Environments Actions Manage Access IDBroker Mappings . If no role is mapped, then you may see the following error: Failed to obtain storage credentials from IDBroker with error: 403 Client Error: Forbidden for url: <url> { "error": "There is no mapped role for the group(s) associated with the authenticated user.", "auth_id": "<user-id>" }. To resolve this, add the user and the Data Access role to the current mappings on the **IDBroker Mappings** page.

Procedure

1. Sign in to Cloudera Manager as an Admin user.
2. Go to Clusters Hue service Configuration .
3. Specify the following in the Hue Server Advanced Configuration Snippet (Safety Valve) for hue_safety_valve_server.ini field:

```
[desktop]
# Remove the file browser from the blacklisted apps.
# Tweak the app_blacklist property to suit your app configuration.
app_blacklist=spark,zookeeper,hive,hbase,search,oozie,jobsub,pig,sqoop,sec
urity
[azure]
[[abfs_clusters]]
[[[default]]]
```

```
  fs_defaultfs=abfs://<CONTAINER_NAME>@<STORAGE_NAME>.dfs.core.windows.net
 webhdfs_url=https://<STORAGE_NAME>.dfs.core.windows.net/
```

The custom configuration is stored in the hue_safety_valve_server.ini file.

4. (Optional) If the Knox server and Hue are not present on the same host, then add the add the Hue server's FQDN to the trusted origins property in the Hue Server Advanced Configuration Snippet (Safety Valve) for hue_safety_valve_server.ini field:

```
[desktop]
[[session]]
trusted_origins=[***HUE-SERVER-FQDN***]
```

5. Click Save Changes.
6. Restart the Hue service.

   The ABFS file browser icon appears on the left Assist panel as well as on the left navigation bar on the Hue web interface.

What to do next

You must manually grant the following application permission to non-admin users and groups for them to be able to view and access ABFS File Browser in Hue: filebrowser.abfs_access:Access to ABFS from filebrowser and file picker.

You must also add the Cloudera users and groups to IAM role mappings.

Related Information

Granting permission to access S3 and ABFS File Browser in Hue

Adding Cloudera user/group to IAM role mappings

Enabling ABFS file browser for Hue configured without IDBroker

You can directly access the Azure file storage from Hue to upload files and tables to Azure file storage and import CSV files as tables directly in Hue by enabling the ABFS file browser on the Hue user interface. You must specify the Azure client ID and the client secret ID information along with the tenant ID in the Hue Server Advanced Configuration Snippet if you have not set up authentication using Knox IDBroker.

Procedure

1. Sign in to Cloudera Manager as an Admin user.
2. Go to Clusters Hue service Configuration .
3. Specify the following in the Hue Server Advanced Configuration Snippet (Safety Valve) for hue_safety_valve_server.ini field:

```
[desktop]
# Remove the file browser from the blacklisted apps.
# Tweak the app_blacklist property to suit your app configuration.
app_blacklist=spark,zookeeper,hive,hbase,search,oozie,jobsub,pig,sqoop,sec
urity
[azure]
  [[azure_accounts]]
    [[[default]]]
      client_id=<CLIENT_ID>
      client_secret=<CLIENT_SECRET_ID>
      tenant_id=<TENANT_ID>

    [[abfs_clusters]]
      [[[default]]]
  fs_defaultfs=abfs://<CONTAINER_NAME>@<STORAGE_NAME>.dfs.core.windows.net
```

```
webhdfs_url=https://<STORAGE_NAME>.dfs.core.windows.net/
```

The custom configuration is stored in the hue_safety_valve_server.ini file.

4. Click Save Changes.
5. Restart the Hue service.

   The ABFS file browser icon appears on the left Assist panel as well as on the left navigation bar on the Hue web interface.

What to do next

You must manually grant the following application permission to non-admin users and groups for them to be able to view and access S3 File Browser in Hue: filebrowser.abfs_access:Access to ABFS from filebrowser and filepicker.

Related Information

Granting permission to access S3 and ABFS File Browser in Hue

Enabling ABFS File Browser in Hue with RAZ in DataHub

Hue offers you the capability to browse Azure Data Lake Storage (ADLS) Gen2 file system, upload files to ABFS, and create tables by importing files from ABFS containers. With RAZ, you can grant fine-grained access to per-user home directories and other directories within the ADLS containers using Ranger policies.

About this task

Hue administrators can create home directories for users by selecting the Create home directory option on the user's profile in Hue. If fine-grained authorization is enabled to access ADLS containers, then user home directories are automatically created when a user logs into Hue. You can disable automatic creation of user directories.

Note: If you have enabled RAZ while registering your Azure environment with Cloudera, then Hue uses RAZ as the default mechanism for enabling the ABFS File Browser

Before you begin

You must enable RAZ while registering your environment with Cloudera. See Introduction to RAZ on Azure environments” to get started with RAZ and complete the following steps:

1. Register an Azure environment with the Enable Ranger authorization for ADLS Gen2 option enabled. You can use the Cloudera web interface or the Cloudera CLI to complete this task.
2. Create a Cloudera Data Hub cluster with Cloudera Data Engineering or Data Mart cluster template.
3. Create the following Ranger policies:
   - Hadoop SQL policy (all - database, table, column, all - url).

     Note: You must grant permissions to individual users or groups in these Ranger policies. To grant permissions to all users, you can specify {USER} in the Permission section.
   - ABFS (cm_ADLS) policy (Default: User Home)

     You must specify the storage account name in the Storage Account field and the directory path of the container and its sub-directories in the Storage Account Container field of the cm_ADLS Ranger policy.
4. Grant appropriate permissions to the users in Cloudera User Management Service (UMS). For example, EnvironmentUser.

Procedure

1. Log in to the Cloudera web interface as an Administrator.
2. Go to Management Console Environments , select your environment and go to the Data Lake tab.
3. Click on the CM URL to open Cloudera Manager.

4. Go to Clusters Ranger RAZ service Instances RAZ server Processes and note the value of the fs.azure.ext.raz.rest.host.url property from the core-site.xml file. You need this to specify the value of the api_url property in the Hue configuration.
5. Go to Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini and add the following lines:

```
[desktop]
app_blacklist=spark,zookeeper,hbase,impala,search,sqoop,security,pig
[[raz]]
is_enabled=true
api_url=[***fs.azure.ext.raz.rest.host.url***]

[azure]
[[abfs_clusters]]
[[[default]]]
fs_defaultfs=abfs://[***CONTAINER-FOR-DATA-ACCESS***]@[***STORAGE-ACCOUNT
***].dfs.core.windows.net
webhdfs_url=https://[***STORAGE-ACCOUNT***].dfs.core.windows.net

[filebrowser]
remote_storage_home=abfs://[***CONTAINER-FOR-DATA-ACCESS***]/user
```

For example:

- fs_defaultfs=abfs://storage-fs@mystorage.dfs.core.windows.net
- webhdfs_url=https://mystorage.dfs.core.windows.net
- remote_storage_home=abfs://storage-fs/user

(RAZ-HA) You can specify a comma-separated list of URLs in the api_url property. For example:

```
api_url=https://[***INSTANCE-1***]:6082/,https://[***INSTANCE-2***]:6082/
```

6. Click Save Changes.
7. Restart the Hue service.

Results

ABFS File Browser icon is added to the left assist pane on the Hue web interface. When you click on the ABFS File Browser icon, you are redirected to your user home directory.

Related Information

Introduction to RAZ on Azure environments

Disabling the automatic creation of user home directories

Using Google Cloud Storage with Hue

Hue can read to and write to Google Cloud Storage (GS) buckets.

Hue offers you the capability to browse GS buckets, upload files to GS, and create tables by importing files from GS. Using Ranger Authorization Service (RAZ), you can grant fine-grained access to per-user home directories and other directories within the GS bucket using Ranger policies.

Hue administrators can create home directories for users by selecting the Create home directory option on the user's profile in Hue. If fine-grained authorization is enabled to access GS buckets, then user home directories are automatically created when a user logs into Hue. You can disable the automatic creation of user directories.

Related Information

Disabling the automatic creation of user home directories

Prerequisites for enabling GS File Browser in Hue with RAZ in Cloudera Data Hub

First, you must register your Google Cloud environment with Cloudera with the Enable Ranger authorization for Google Cloud Storage option enabled. Next, you must set up Ranger policies for fine-grained access control for your user and Google Cloud Storage (GS) buckets.

Procedure

1. Log in to the Cloudera Management Console as an Administrator.
2. Register a GCP environment with the Enable Ranger authorization for Google Cloud Storage option enabled. You can complete this task using the Cloudera web interface or the Cloudera CLI.
3. Create a Cloudera Data Hub cluster with a Cloudera Data Engineering or Data Mart cluster template.
4. Create the following Ranger policies:
   a) Hadoop SQL policy (all - database, table, column, all - url).

      Note: You must grant permissions to individual users or groups in these Ranger policies. To grant permissions to all users, specify {USER} in the **Permission** section.

   b) GS (cm_gs) policy (Default: User Home).

      You must specify the bucket name in the GS Bucket field and the directory path in the Path field of the cm_gs Ranger policy.
5. Grant appropriate permissions to the users in the Cloudera User Management Service (UMS). For example, EnvironmentUser--grants permission to view Cloudera Data Hub clusters and set the workload password for the environment.

What to do next

Enable the GS File Browser in Hue by configuring [desktop] properties in the Hue Advanced Configuration Snippet using Cloudera Manager.

Related Information

Introduction to RAZ on GCP environments

Enabling GS File Browser for Hue with RAZ in Cloudera Data Hub

After you have registered your Google Cloud environment with Cloudera with the Enable Ranger authorization for Google Cloud Storage (GS) option enabled, created a Cloudera Data Hub cluster, created the required Ranger policies, and granted access to your users, you can enable the GS File Browser using Cloudera Manager.

Procedure

1. Log in to the Cloudera web interface as an Administrator.
2. Go to Cloudera Management Console Environments , select your environment and go to the Data Lake tab.
3. Click on the link under CM URL to open Cloudera Manager.
4. Go to Clusters Ranger RAZ service Instances RAZ server Processes and note the value of the fs.s3a.ext.raz.rest.host.url property from the core-site.xml file. You need this to specify the value of the api_url property in the Hue configuration.
5. Go to Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini and add the following lines:

```
[desktop]
app_blacklist=spark,zookeeper,hbase,impala,search,sqoop,security,pig
[[raz]]
is_enabled=true
is_raz_gs_enabled=true
api_url=[***FS.S3A.EXT.RAZ.REST.HOST.URL***]

[filebrowser]
```

```
remote_storage_home=gs://[***GS-BUCKET-NAME***]/user
```

(RAZ-HA) You can specify a comma-separated list of URLs in the api_url property. For example:

```
api_url=https://[***INSTANCE-1***]:6082/,https://[***INSTANCE-2***]:6082/
```

6. Click Save Changes.
7. Restart the Hue service.

Results

GS File Browser icon is added to the left assist panel on the Hue web interface. When you click the GS File Browser icon, you are redirected to your user home directory.

Disabling the automatic creation of user home directories

If you have enabled fine-grained authorization to access S3 buckets, ADLS Gen 2 storage, or Google Cloud Storage buckets, then Hue is configured to automatically create user home directories, by default. You can disable the automatic creation of user home directories by setting the autocreate_user_dir flag to false in the Hue Service Advanced Configuration Snippet.

Procedure

1. Log in to the Cloudera web interface as an Administrator.
2. Go to Data Hub Clusters and select your cluster.
3. Click on the CM URL to open Cloudera Manager.
4. Go to Clusters Hue service Configuration Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini and add the following lines:

```
[desktop]
[[raz]]
autocreate_user_dir=false
```

5. Click Save Changes.
6. Restart the Hue service.

Granting permission to access S3, ABFS, GS File Browser in Hue

Only admin users can view and access S3, ABFS, and GS File Browser in Hue after enabling it. You must manually grant application permissions to non-admin users and groups for them to be able to view and access S3, ABFS, and GS File Browsers in Hue.

About this task

The following table lists the application permissions for each cloud storage type:

Cloud storage	Hue application permission
S3	filebrowser.s3_access:Access to S3 from filebrowser and filepicker.
ABFS (ADLS Gen2)	filebrowser.abfs_access:Access to ABFS from filebrowser and filepicker.
ADLS (Gen1)	filebrowser.adls_access:Access to ADLS from filebrowser and filepicker.
GS	Access to GS from filebrowser and filepicker.

Before you begin

You can only assign Hue application permissions to groups or users within a group. Add the users to a group to whom you want to grant access to the S3, ABFS, or GS File Browsers.

Important: The "default" group in Hue does not have the permissions required to view S3, ABFS, or GS File Browsers, by default. If your users belong to the default group, then you must manually grant the required permissions.

Procedure

1. Sign in to Hue as an Administrator.
2. Go to admin Manage Users Groups .
3. Click on the group to whom you want to grant the filebrowser application permissions.
4. On the **Edit [***GROUP-NAME***]** page, select the required permission under the permission section and click Update group.

Creating tables in Hue by importing files

Using Hue Importer, you can create Hive, Impala, and Iceberg tables from CVS and XLSX files. After enabling the File Browser for your cloud provider, you can import the file into Hue to create tables.

About this task

You can upload and import a 200 KB file from your local computer, or import a file up to 3 GB from cloud storage (S3, ADLS Gen2, and Google Cloud Storage (GS) buckets) using the Importer.

Note: To enable uploading a small local file from your computer, you must go to Cloudera Manager Clusters Hue Configuration and add the following lines in the Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini field:

```
[indexer]
  # Flag to turn on the direct upload of a small file.
  enable_direct_upload=true
```

You can create managed and external tables. When you create managed tables, the data is moved to HDFS in the Cloudera Data Warehouse workspace. When you create external tables, your data remains in its original location.

Note: Superusers can browse all directories up to the root. Other users can only browse and access buckets and directories that they are granted access to.

Before you begin

- Enable user access to cloud storage buckets/containers in Ranger
- Enable the File Browser for your cloud provider

Procedure

1. Log in to the Hue web interface.
2. Click Importer from the left-assist panel.

   Alternatively, click + to create tables under the Databases Tables list view. This opens the Hue Importer.
3. Under SOURCE, select Remote File from the Type drop-down menu.
4. Click .. at the end of the Path field.

   The **Choose a file** modal is displayed.

5. (Non-RAZ deployment only) Click on your cloud provider, type the following depending on your cloud provider, and press enter:

   (AWS): s3a://[***BUCKET-NAME***]

   (Azure): abfs://[***CONTAINER-NAME***]

   (GCS): gs://[***BUCKET-NAME***]
6. Browse and select the file you want to use to create a table.

   Hue displays the preview of the table along with the format.

   Hue automatically detects the field separator, record separator, and the quote character from the file. If you want to override a specific setting, then you can change it by selecting a different value from the drop-down menu.
7. Click Next.

   The table destination and properties are displayed.
8. Under Extras, deselect Store in Default location and Transaction table options.

   This is required to create external tables.
9. Select the Copy file option.

   Selecting this option retains the source file in the original location and creates a copy of the file to create a table.

   If you do not select this option, then Hue moves the file from the source location to a new folder, making it unavailable in the original location.
10. Set the table destination, partitions, and change the column data types.
11. Verify the settings and click Submit to create the table.

    The CREATE TABLE query is triggered.

    Hue displays the logs and opens the Table Browser from which you can view the newly created table when the operation completes successfully.

Related Information

Enabling S3 File Browser for Hue with RAZ in Cloudera Data Hub

Enabling ABFS File Browser in Hue with RAZ in Cloudera Data Hub

Enabling GS File Browser for Hue with RAZ in Cloudera Data Hub

Supported non-ASCII and special characters in Hue

Auto-generated files may often introduce non-alphanumeric characters in the file and directory names that Hue does not support. This might cause the files or directories to not appear on the Hue File Browser. Review the list of non-alphanumeric, non-ASCII, diacritics (accents) characters supported in Hue for the following operations: upload, create, list in folder, view, and rename.

Table 1: Supported characters on HDFS

Special character symbol	Description	Filename support	Folder name support
~	Tilde	Fully supported	Fully supported
@	Ampersat	Fully supported	Fully supported
#	Hash	Partially supported. Not supported for rename operations.	Unsupported
$	Dollar sign	Fully supported	Fully supported
&	Ampersand	Fully supported	Fully supported
(	Left paranthesis	Fully supported	Fully supported
)	Right paranthesis	Fully supported	Fully supported
*	Asterisk	Fully supported	Fully supported
!	Exclamation mark	Fully supported	Fully supported

Special character symbol	Description	Filename support	Folder name support
+	Plus	Fully supported	Fully supported
=	Equal	Fully supported	Fully supported
:	Colon	Unsupported	Unsupported
;	Semicolon	Fully supported	Fully supported
,	Comma	Fully supported	Fully supported
.	Period	Fully supported	Fully supported
?	Question mark Not supported with Knox.	Fully supported	Fully supported
/	Forward slash	Unsupported	Unsupported
\	Backslash	Unsupported	Unsupported
'	Apostrophe or single quote	Fully supported	Fully supported
%	Percentage	Fully supported	Fully supported
T######-ääö	Non-standard alphabets with diacritics and accents.	Fully supported	Fully supported

Table 2: Supported characters on S3

Special character symbol	Description	Filename support	Folder name support
~	Tilde	Fully supported	Fully supported
@	Ampersat	Fully supported	Fully supported
#	Hash	Partially supported. Not supported for view and rename operations.	Unsupported
$	Dollar sign	Fully supported	Fully supported
&	Ampersand	Fully supported	Fully supported
(	Left paranthesis	Fully supported	Fully supported
)	Right paranthesis	Fully supported	Fully supported
*	Asterisk	Fully supported	Fully supported
!	Exclamation mark	Fully supported	Fully supported
+	Plus	Fully supported. Not supported for create operations on RAZ-enabled environments.	Fully supported
=	Equal	Fully supported	Fully supported
:	Colon	Unsupported	Unsupported
;	Semicolon	Fully supported	Fully supported
,	Comma	Fully supported	Fully supported
.	Period	Fully supported	Fully supported
?	Question mark Not supported with Knox.	Fully supported	Partially supported. Not supported for list, upload, and view operations.
/	Forward slash	Unsupported	Unsupported
\	Backslash	Partially supported. Not supported for upload operations.	Partially supported. Not supported for upload operations.
'	Apostrophe or single quote	Fully supported	Fully supported

Special character symbol	Description	Filename support	Folder name support
%	Percentage	Fully supported	Fully supported
T######-ääö	Non-standard alphabets with diacritics and accents.	Fully supported	Fully supported

Table 3: Supported characters on ABFS

Special character symbol	Description	Filename support	Folder name support
~	Tilde	Fully supported	Fully supported
@	Ampersat	Fully supported	Fully supported
#	Hash	Partially supported. Not supported for view and rename operations.	Unsupported
$	Dollar sign	Fully supported	Fully supported
&	Ampersand	Fully supported	Fully supported
(	Left paranthesis	Fully supported	Fully supported
)	Right paranthesis	Fully supported	Fully supported
*	Asterisk	Fully supported	Fully supported
!	Exclamation mark	Fully supported	Fully supported
+	Plus	Fully supported	Fully supported
=	Equal	Fully supported	Fully supported
:	Colon	Unsupported	Unsupported
;	Semicolon	Fully supported	Fully supported
,	Comma	Fully supported	Fully supported
.	Period	Fully supported	Fully supported
?	Question mark Not supported with Knox.	Partially supported. Not supported for view and rename operations.	Partially supported. Not supported for list, rename, and view operations.
/	Forward slash	Unsupported	Unsupported
\	Backslash	Unsupported	Unsupported
'	Apostrophe or single quote	Fully supported	Fully supported
%	Percentage	Fully supported	Fully supported
T######-ääö	Non-standard alphabets with diacritics and accents.	Fully supported	Fully supported

Options to rerun Oozie workflows in Hue

Oozie workflows consume time and resources to run. You can optimize the rerun of a failed Oozie workflow by selecting one of the following two options: “All or skip successful” or “Only failed”. These options enable you to select individual actions within your workflow that you want to rerun.

Important: The option to rerun the an Oozie workflow is enabled in the following conditions:

- In case of an external workflow, the rerun button is enabled if the workflow was run by the same user who is logged in, or if the user is a Hue superuser.
- If the workflow is created in Hue, then the rerun button is enabled only for the user who originally ran the workflow, or for the Hue superuser.

The All or skip successful option enables you to either rerun both the successful and failed actions or skip the actions that ran successfully and run only the failed ones. Using this option, you have more control over selecting the actions that you want to rerun from the list of successfully run actions.

If you select the All or skip successful option but do not select any or all of the successfully run actions, then Hue reruns the whole Oozie workflow.

If you select the All or skip successful option and select some of the successfully run actions, then Hue reruns the selected actions and the failed actions.

Select the Only failed option to only rerun the failed actions within your workflow.

The following image shows the available rerun options on the Hue web interface:

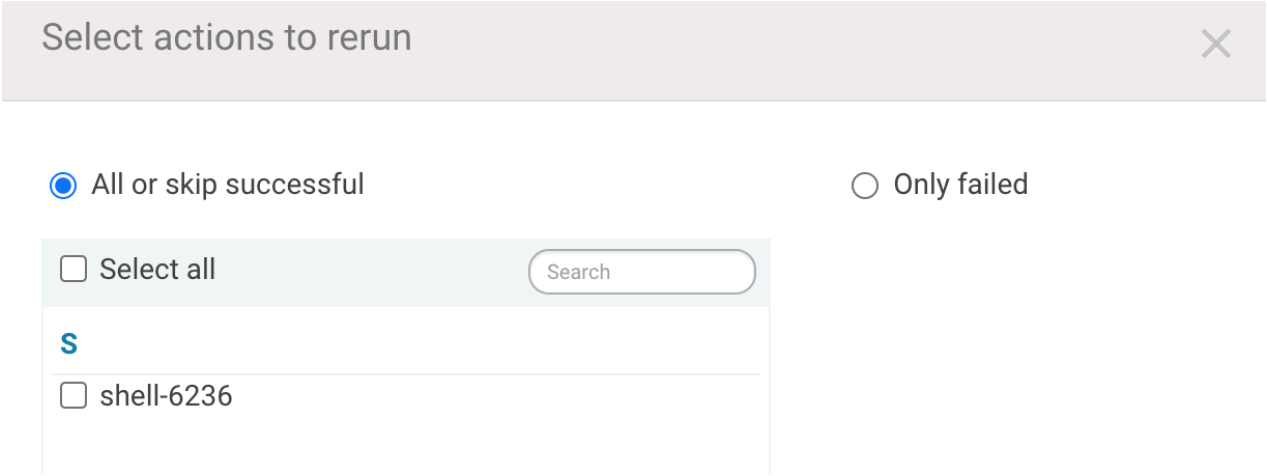

Unsupported features in Hue

Learn about the Hue features that are not supported by Cloudera.

Unsupported options in Hue Importer

The following options are displayed on the Hue **Importer** page under SOURCE Path , but are not supported:

- External Database

  Creating an external database using the Hue Importer is not supported. Cloudera recommends that you create a database using a SQL query.
- Manually

Known limitations in Hue

Review the known limitations in Hue.

Hue has the following limitations:

- Node depth for graphing Oozie workflows because of performance issues. See Improved Oozie Workflow display of large Graphs.
- You must use the Cloudera-provided Apache Load balancer to serve static content, because:
  - It serves static JavaScript, CSS, and Webpack files for client requests and reduces the load from the backend Python web server.
  - The Hue load balancer uses a sticky cookie session to route requests to the same backend as the Python web server, which talks to the same coordinator.
- Hue can only show logs from either Spark1 or Spark2, not both at a time.

- Spark notebook is not supported.
- External RDBMS in the query editor is not supported out of the box by default. Cloudera support will assist on a best-effort basis. Cloudera recommends that you raise issues in the open-source github community.
- Impala queries stay in the “executing” state so that Hue can display results when users are ready
- We need to limit the amount of data available to download from Hive/Impala because massive downloads cause performance degradation. Multiple simultaneous downloads of result sets could also degrade performance.
- Upstream features and connectors may not function properly in Cloudera. Cloudera recommends that you raise issues in the open-source github community.

Cloudera Runtime 7.3.1

Using Schema Registry

Date published: 2019-08-22
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Adding a new schema 4

Querying a schema 5

Evolving a schema 5

Deleting a schema 7

Importing Confluent Schema Registry schemas into Schema Registry 7

Exporting and importing schemas 8

Exporting schemas using Schema Registry API 9
Importing schemas using Schema Registry API 9

ID ranges in Schema Registry 11

Setting a Schema Registry ID range 11

Load balancer in front of Schema Registry instances 12

Configurations required to use load balancer with Kerberos enabled 12
Configurations required to use load balancer with SSL enabled 13

Adding a new schema

Learn how to add a new schema to Schema Registry through the Schema Registry UI.

About this task

A schema is a collection of information that describes your data. The metadata of the schema includes group, version, and description. To add a new schema to Schema Registry, provide information about the schema, select a compatibility policy, and upload the schema text from a file.

Before you begin

Ensure that you understand compatibility policies. Once selected, you cannot change the compatibility policy for a schema.

Procedure

1. From the Schema Registry UI, click the + icon.

   The Add New Schema dialog appears.

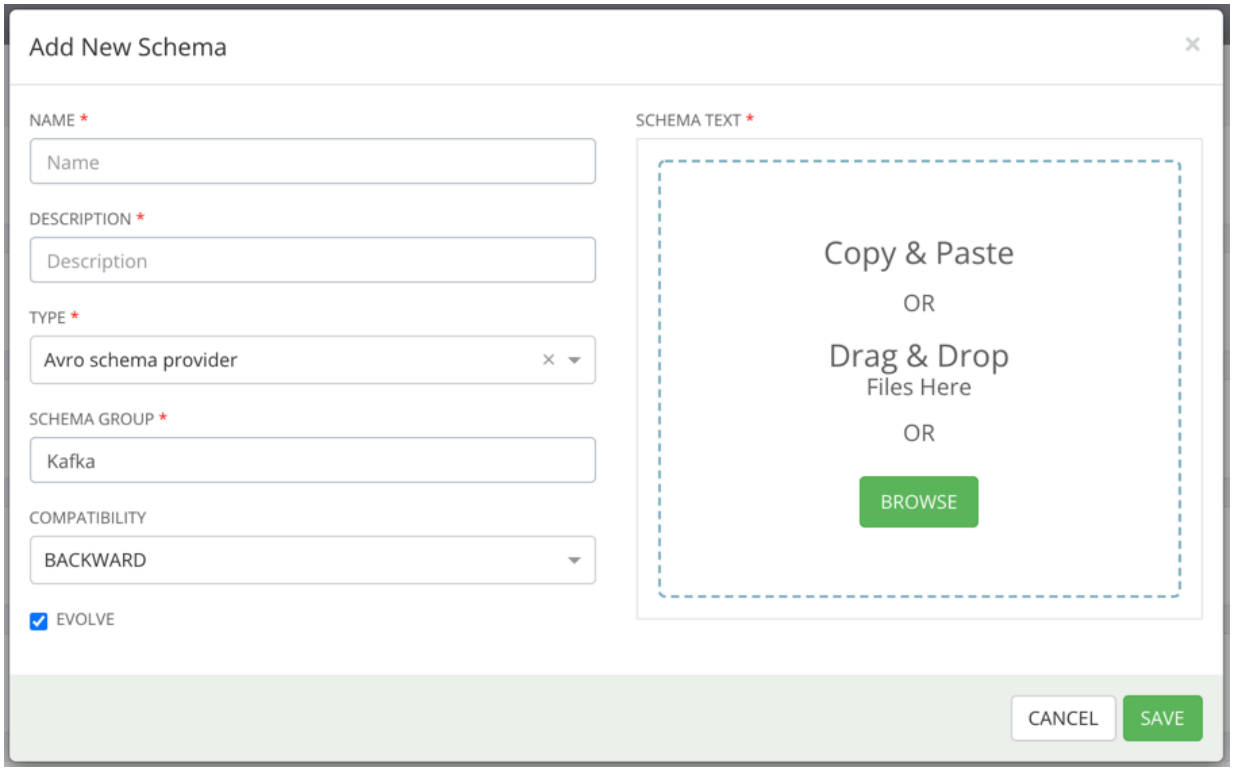

2. Add the schema metadata as follows:
   - Name

     A unique name for each schema. It is used as a key to look up schemas.
   - Description

     A short description of the schema.
   - Schema Type

     The schema format. Select one of the following formats:
     - Avro schema provider
     - JSON schema provider
   - Schema Group

     Allows you to group schemas in any logical order.
   - Compatibility

     This option appears only when the Avro schema provider option is selected. Sets the compatibility policy for the schema. Once set, this cannot be changed. Select one of the following options:
     - Backward
     - Forward
     - Both
     - None

     For more information about compatibility, see *Compatibility Policies*.
   - Schema Text
3. To allow schema to evolve over time by creating multiple versions, select the Evolve checkbox.

   Note:

   Unselecting Evolve means that you can only have one version of a schema.
4. Click Choose File to upload a new schema.

Related Information

Compatibility Policies

Querying a schema

Learn how to search for a particular schema in the Schema Registry UI.

You can use the Search box at the top of the Schema Registry UI to search for schemas by name, description, or schema text. To search, you can enter a schema name, key words, or any text string to return schemas with that value.

To return to your full list of schemas, clear the search box and press Enter on your keyboard.

Evolving a schema

Learn how to save each version of your schema as you make changes to it.

About this task

You evolve a schema when you create a new version. Schema Registry tracks the changes made to your schema, and stores each set of changes in a separate version of the schema. When multiple versions exist, you can select which version you want to use. Ensure that you understand compatibility policies, as they determine how you can evolve your schemas.

Before you begin

- You have selected the Evolve checkbox when initially adding the schema.
- You have saved the schema you want to evolve in a file.

Procedure

1. From the Schema Registry UI, identify the schema that you want to version.
2. Click the pencil icon to open the Edit Version dialog.

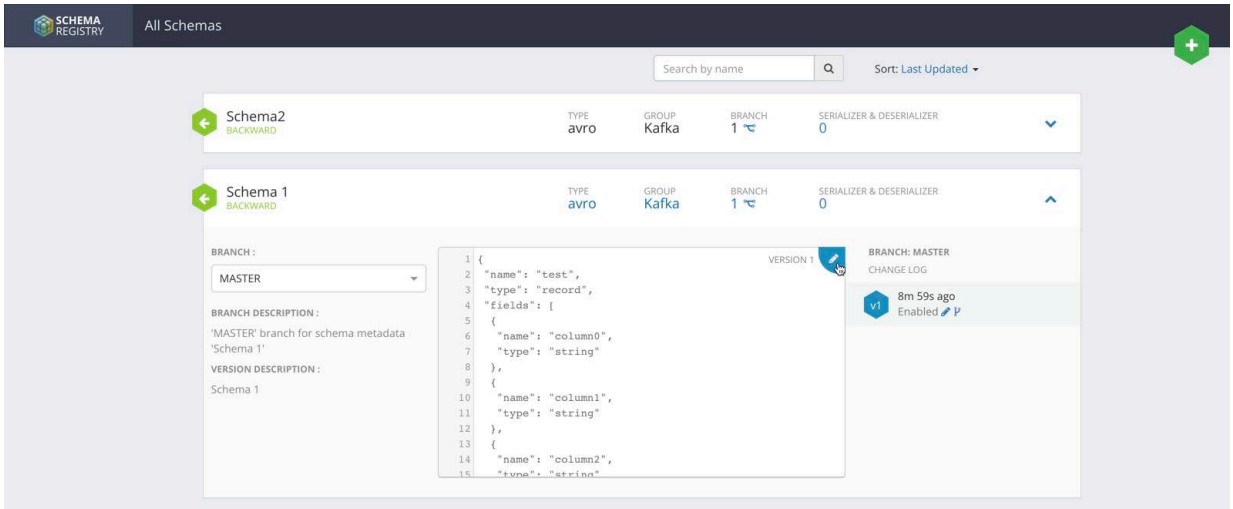

The Edit Version dialog appears.

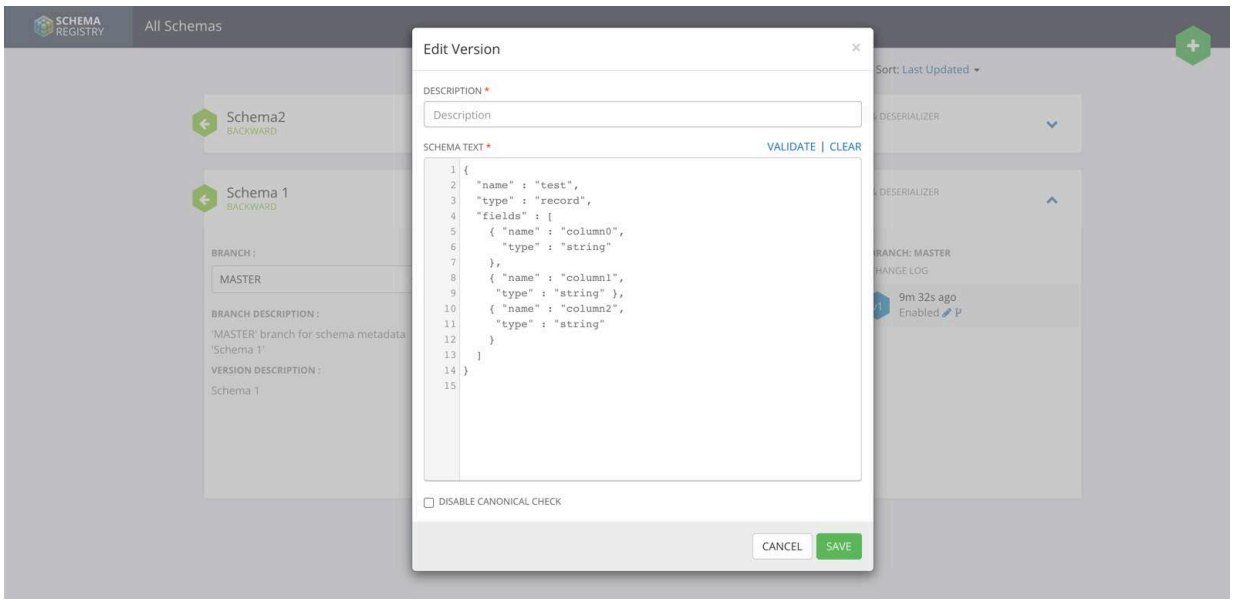

3. Add a description of what has changed in this new version of the schema.

   You can view the description in the Schema Registry UI to easily understand what has changed in each version of the schema. Cloudera recommends that you add as much detail as you can.
4. Optional. Select the Disable Canonical Check checkbox if the schema version should be added despite being canonically similar to an existing schema version.

   The canonical check checkbox only applies to versions of schemas and not to a new schema.
5. Click Choose File to upload the schema you want to evolve.
6. Click OK.

Related Information

Compatibility Policies

Deleting a schema

You can use the Schema Registry REST API to delete schemas.

About this task

You can access the Schema Registry API Swagger documentation directly from the UI. To do this, append your URL with: /swagger/

For example: https://localhost:7790/swagger/.

Procedure

1. Find the DELETE /api/v1/schemaregistry/schemas/{schema_name} API under Schema.
2. Replace {schema_name} with the schema you wish to delete.
3. Invoke the API.

Importing Confluent Schema Registry schemas into Schema Registry

If you are migrating away from Confluent's Schema Registry, you can use the schema-import tool to import your schemas to the Schema Registry service.

About this task

Confluent Schema Registry stores all schemas in a Kafka topic usually named _schemas. To import these schemas, copy the content to a text file and upload to Schema Registry service.

Before you begin

You must have the export-import access policy with the Create permission assigned to your user account.

For more information on Schema Registry access policies, see *Schema Registry Authorization through Ranger Access Policies*.

Procedure

1. Copy the content of the Kafka topic (usually named _schemas) to a text file using the following command:

   ```
   kafka-console-consumer --from-beginning --topic _schemas --formatter kaf
   ka.tools.DefaultMessageFormatter --property print.key=true --property pr
   int.value=true --bootstrap-server `hostname`:9092
   ```

   The following is an example of a text file with the content of the Kafka topic:

   ```
   {"keytype":"SCHEMA","subject":"Kafka-key","version":1,"magic":1}
   {"subject":"Kafka-key","version":1,"id":1,"schema":"\"string\"","deleted
   ":false}
   {"keytype":"SCHEMA","subject":"Car","version":1,"magic":1}      {"subje
   ct":"Car","version":1,"id":2,"schema":"{\"type\":\"record\",\"name\":\"C
   ar\",\"namespace\":\"com.cloudera\",\"fields\":[{\"name\":\"model\",\"type
   ```

```
\":\"string\"},{\"name\":\"color\",\"type\":\"string\",\"default\":\"blue
\"},{\"name\":\"price\",\"type\":\"string\",\"default\":\"0\"},{\"name\":
\"year\",\"type\":[\"null\",\"string\"],\"default\":null}]}","deleted":f
alse}
{"keytype":"SCHEMA","subject":"Car","version":2,"magic":1}      {"subject"
:"Car","version":2,"id":3,"schema":"{\"type\":\"record\",\"name\":\"Car\",
\"namespace\":\"com.cloudera\",\"fields\":[{\"name\":\"model\",\"type\":\"
string\"},{\"name\":\"color\",\"type\":\"string\",\"default\":\"blue\"},
{\"name\":\"price\",\"type\":\"string\",\"default\":\"0\"}]}","deleted":
false}
```

2. Upload the text file to Schema Registry by using the following endpoint: /api/v1/schemaregistry/import.

   Use the following command:

```
curl -u : --negotiate --form file='@<filename>' http://`hostname`:7788/a
pi/v1/schemaregistry/import?format=1&failOnError=true
```

   Where,

   file='<filename>'

   Enter the name of the file.

   format

   Can be 0 or 1. However, for schema content from Confluent, the file must be 1.

   failOnError=true

   This setting is useful if the process fails to import all the schemas when one or more schemas already exist in the target Schema Registry.

   The following is an example of the command:

```
curl -u : --negotiate --form file='@schemadump.json' http://`hostname`:7
788/api/v1/schemaregistry/import?format=1&failOnError=true
```

   The following is an example of the response:

```
{"successCount":45,"failedCount":0,"failedIds":[]}
```

Related Information

Schema Registry Authorization through Ranger Access Policies

Exporting and importing schemas

Learn how to export and import schemas between different Schema Regitry instances allowing services to exchange data without the challenge of managing and sharing schemas between them.

Schemas stored in Schema Registry can be exported to a JSON file. The exported JSON file can then be imported into another Schema Registry database.

During an import, SchemaMetadata, SchemaBranch, and SchemaVersion objects are put into the database. These objects retain their ID as well as a number of other properties that are available in the JSON file used for the import. This way, serializing and deserializing protocols can continue to function without any change and Schema Registry clients can seamlessly switch between different Schema Registry instances. Both import and export operations can be done either by using curl through the command line or by using the Schema Registry API.

Exporting schemas using Schema Registry API

One of the methods of exporting schemas registered in Schema Registry into a JSON file is using the Schema Registry API. The file that you export can be used to import the exported schemas into a different Schema Registry. You can also export schemas using curl through the command line.

About this task

The following steps walk you through how you can export schemas by using the Schema Registry Swagger UI.

Before you begin

- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that Schema Registry is commissioned and running.
- Ensure that you have access to all credentials that are required to access and use Schema Registry.
- Ensure that you are logged in as a user with access to the Cloudera Public Cloud environment containing the cluster with Schema Registry.

Procedure

1. In Cloudera Management Console, go to Data Hub Clusters.
2. Find and select the Cloudera Data Hub cluster you want to export schemas from.
3. Under Services, click Schema Registry.

   The Schema Registry web UI opens in a new tab.
4. In the tab that has the Schema Registry web UI open, replace /ui/# at the end of the URL with /swagger.
5. Select GET /api/v1/schemaregistry/schemas/aggregated.
6. Click Try it out.
7. Click Execute.
8. Click Download found at the bottom right of the Response body pane to download the exported schemas.

Results

All schemas stored in the Schema Registry server are exported to a JSON file.

Importing schemas using Schema Registry API

One of the methods of importing schemas that were exported from one Schema Registry instance into another Schema Registry instance is using the Schema Registry API. You can also import schemas using curl through the command line.

About this task

The following steps describe how to import schemas by using the Schema Registry Swagger UI.

Important: Schema import is only recommended to an empty Schema Registry server or a Schema Registry server that has previously imported schemas. Importing schemas to a Schema Registry that contains manually added schemas is only possible if the Schema Registry instance has correctly configured ID ranges. Otherwise, importing schemas might result in overlapping ID ranges and a failed import. For more information on how to set up ID ranges, see *ID ranges in Schema Registry*.

Before you begin

- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that Schema Registry is commissioned and running.

- Ensure that you have access to all credentials that are required to access and use Schema Registry.
- Ensure that you have access to a JSON file that contains the exported schemas that you want to import. For more information on how to export schemas, see *Exporting schemas using Schema Registry API*.
- Ensure that you are logged in as a user with access to the Cloudera Public Cloud environment containing the cluster with Schema Registry.

Procedure

1. In Cloudera Management Console, go to Data Hub Clusters.
2. Find and select the Cloudera Data Hub cluster you want to import schemas to.
3. Under Services, click Schema Registry.

   The Schema Registry web UI opens in a new tab.
4. In the tab that has the Schema Registry web UI open, replace /ui/# at the end of the URL with /swagger.
5. Click 3. Export/Import.
6. Click POST /api/v1/schemaregistry/import.
7. Click Try it out.
8. Set format to 0.
9. Set failOnError to true.
10. Click Choose file and select a JSON file containing the schema data that you want to import.
11. Click Execute.

Results

Schemas are successfully imported into the target Schema Registry instance.

On successful import, a number of SchemaMetadata, SchemaBranch, and SchemaVersionInfo objects are added to the Schema Registry database.

Objects added to the database retain the following property values from the import file:

- Schema Metadata objects retain the ID, type, schemaGroup, name, description, compatibility, validationLevel, and evolve properties.
- SchemaBranch objects retain the ID, name, schemaMetadataName, and description properties.
- SchemaVersionInfo objects retain the ID, schemaMetadataId, name, description, version, and schemaText properties.

A successful import also results in a JSON file that contains the following:

```
{
  "successCount": 0,
  "failedCount": 0,
  "failedIds": [
    0
  ]
}
```

Where,

- successCount is the number of successfully imported SchemaVersionInfo objects.
- failedCount is the number of SchemaVersionInfo objects that failed to import.
- failedIds contains the IDs of failed SchemaVersionInfo objects.

Schemas (metadata/branch/version) that are already present in the database are skipped during an import. Skipped schemas are not counted in successCount or failedCount.

Related Information

ID ranges in Schema Registry

Exporting schemas using Schema Registry API

ID ranges in Schema Registry

You can avoid overlapping IDs for Schema Registry entities across different Schema Registry instances by setting ID ranges in each instance.

If your deployment has many Schema Registry instances in different clusters, you can ensure that each Schema Registry instance operates in a preset ID range so that there are no overlapping IDs. To achieve this, you need to set an ID range.

When an ID range is set, each Schema Registry entity (schema metadata, branches, versions, and so on) is created with an ID that falls within the configured range. Configuring a range can simplify schema export and import between Schema Registry instances where conflicting IDs can cause the import process to fail.

Consequently, having non-overlapping ID ranges allocated to each Schema Registry makes data replication easier when schema IDs are embedded in the data. This is important because replicated data can only be deserialized with compatible schemas on the replication target site.

When IDs reach the range maximum value, adding more schema fails until a new ID range is configured. The new range must be larger than the current one. The new range must also not overlap with any other ranges.

Important: Cloudera does not recommend that you import schemas into a Schema Registry's ID range. IDs assigned sequentially from the range start value might reach the imported IDs.

Setting a Schema Registry ID range

Learn how to set an ID range for schemas in Schema Registry using Cloudera Manager.

About this task

An ID range that is unique to a Schema Registry instance can be set in Cloudera Manager on the Schema Registry service's configuration page. Configuring an ID range does not alter already existing schemas.

Before you begin

- Ensure that the cluster, its hosts, and all its services are healthy.
- Ensure that Schema Registry is commissioned and running.
- Ensure that you have access to all credentials that are required to access and use Schema Registry.
- Plan ahead and designate ID ranges for each Schema Registry instance. Ensure that the planned ranges do not overlap.
- Ensure that the ID range you plan to configure has enough IDs. Base the size of your range on the approximate number of schemas you expect to have.

Procedure

1. In Cloudera Manager, select the Schema Registry service.
2. Go to Configuration.
3. Find and configure the following properties:
   - ID Offset Range Minimum
   - ID Offset Range Maximum

   Configure the properties according to your planned ID ranges. For example, if you want to have an ID range of 1-100, set ID Offset Range Minimum to 1 and ID Offset Range Maximum to 100.
4. Click Save Changes.
5. Restart Schema Registry.

Results

When new schemas are added, either through the UI or the API, the schema IDs are allocated from the configured range.

Load balancer in front of Schema Registry instances

It is common to have multiple instances of the same application, for example Schema Registry, and have a load balancer in front of them. This can be useful for failover reasons in HA environments, and it can also help sharing the load between instances.

A load balancer can be any application (HAProxy, NGINX, and so on) that listens on a specific port and forwards requests (create schema, get schema, and so on) to Schema Registry instances.

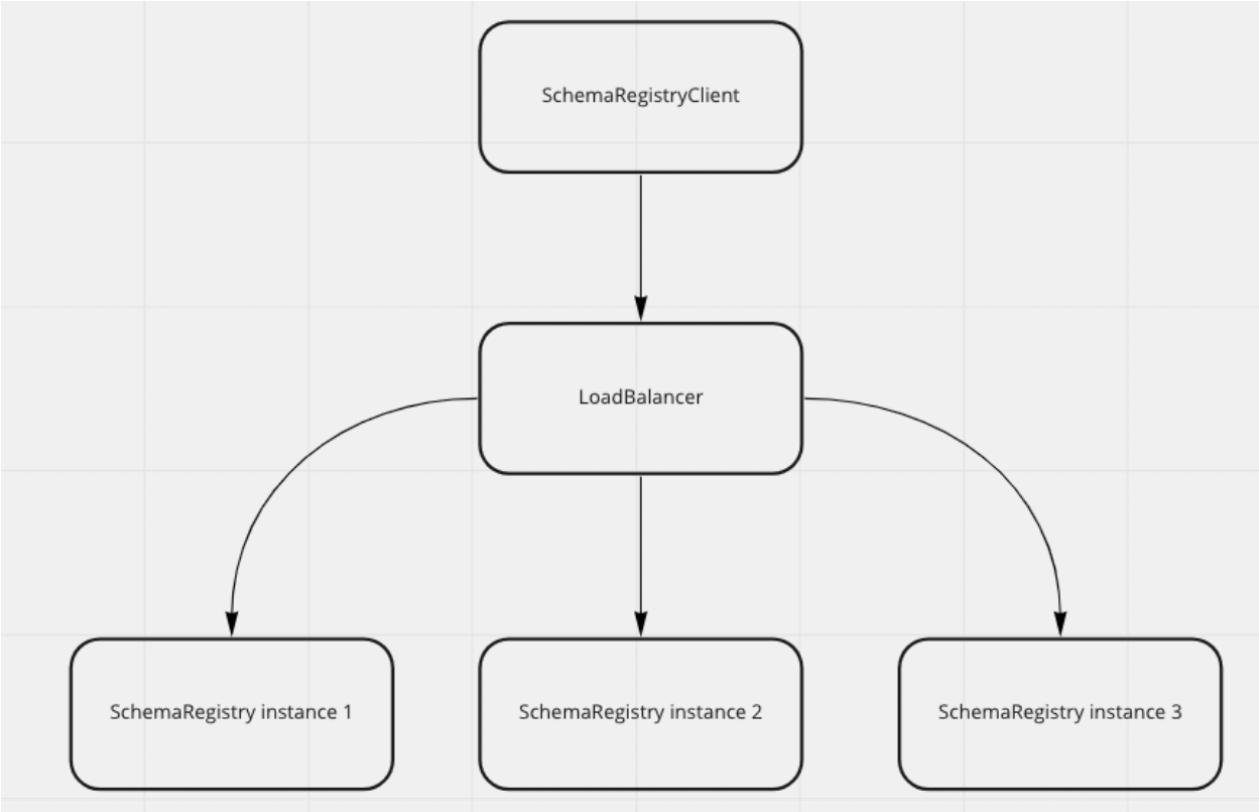

A load balancer not only balances requests using round robin method, but also tracks application instances whether they are usable or not. Having a load balancer minimizes the need to change client-side properties with instance hostnames, because the only host that the client needs to know is the load balancer host. In case the environment is completely insecure, you do not need to perform any additional steps. But in case of SSL or Kerberos is enabled, some defending mechanisms (for example, hostname verification) can cause issues, which requires additional setup to make the load balancing work.

Note: Cloudera recommends using Knox as load balancer in Public Cloud. It performs round robin method to load balance between multiple Schema Registry instances. Knox can be used differently than what is documented here, because it can handle authentication, so requests are not forwarded as is. In case Knox is used as load balancer, the setup mentioned in this document is not needed.

Configurations required to use load balancer with Kerberos enabled

Learn how to use load balancer in front of Schema Registry instances in an environment where Kerberos is enabled.

In this case, the clients connect to the load balancer, so they acquire a Kerberos ticket including the load balancer host. In the meantime, Schema Registry instances are logged in with their service principal including their FQDN. Because the load balancer host and the FQDN are probably not the same (if it is assumed that load balancer is not deployed to Schema Registry host), requests result in ticket mismatch to avoid man-in-the-middle attacks.

To ensure that requests forwarded by a load balancer are accepted, you need to do the following:

1. Log into Cloudera Manager and go to the Schema Registry service.
2. Set schema_registry_load_balancer_host to the host of the load balancer.
3. Restart the Schema Registry service.

This generates a new principal HTTP/<loadbalancerhost>@REALM. Each Schema Registry instance not only logs in with its FQDN related service principal, but also logs in with the load balancer related one, so forwarded requests do not fail.

Configurations required to use load balancer with SSL enabled

Learn how to use a load balancer in front of Schema Registry instances in an environment where SSL is enabled.

When SSL is enabled in your environment, the clients automatically check the SSL certificate of the Schema Registry servers and compare it with where they connected. This is called hostname verification and it fails because the clients connect to the load balancer host, but the Schema Registry server SSL certificate contains the Schema Registry server related host FQDN. To make the load-balancing work with forwarded requests, you can have multiple options to choose from:

- You can modify the server certificate and add a SAN (Subject Alternative Name) certificate including the load balancer host. This is the best option because it ensures that hostname verification only passes if the original FQDN or the load balancer host is used; otherwise it fails.
- If changing the certificates is a big effort, you can disable hostname verification on the Schema Registry client side. The Schema Registry client holds all SSL related properties in a map that contains the schema.registry.client.ssl key in the properties. In this map, it is possible to set an entry with the hostnameVerifierClass key. The value of this key points to a class that implements the Java specific HostnameVerifier interface. An implementation to allow every hostname during hostname verification is shown in the following example:

```
public static class HostnameVerificationDisabled implements HostnameVerifier {

 @Override
 public boolean verify(String hostname, SSLSession session) {
   // Allowing everything is the same as disabling hostname verification.
   return true;
 }
}
```

An example setting of this property is as follows:

```
// Create map for all client properties
HashMap<String, String> clientProperties = new HashMap<>();
// Create SSL related map
HashMap<String, String> sslProperties = new HashMap<>();
// Set hostname verifier in SSL properties
sslProperties.put("hostnameVerifierClass", HostnameVerificationDisabled.class.getName());

// Set other SSL related properties
sslProperties.put(...) ;
// Set SSL map on client properties map
clientProperties.put(“schema.registry.client.ssl”, sslProperties);
```

Important: This solution is less secure, because it disables an SSL client-side defending mechanism. So, Cloudera recommends to use it carefully.

Managing and Allocating Cluster Resources using Capacity Scheduler

Date published:
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Resource scheduling and management.....6
YARN resource allocation of multiple resource-types.....6
Hierarchical queue characteristics.....6
Scheduling among queues.....7
Application reservations.....7
Resource distribution workflow.....7
Resource allocation overview.....9
Use CPU scheduling.....10
Configure CPU scheduling and isolation.....10
Use CPU scheduling with distributed shell.....10
Use GPU scheduling.....11
Configure GPU scheduling and isolation.....11
Use GPU scheduling with distributed shell.....12
Use FPGA scheduling.....12
Configure FPGA scheduling and isolation.....13
Use FPGA with distributed shell.....14
Limit CPU usage with Cgroups.....14
Use Cgroups.....15
Enable Cgroups.....15

Managing YARN Queue Manager.....17
Configuring YARN Queue Manager dependency.....17
Updating YARN Queue Manager Database Password.....18
Accessing the YARN Queue Manager UI.....19
Providing read-only access to Queue Manager UI.....21
Configuring the embedded Jetty Server in Queue Manager.....24

Managing queues.....24
Adding queues using YARN Queue Manager UI.....25
Configuring cluster capacity with queues.....30
Configuring the resource capacity of root queue in absolute mode.....31
Mixed resource allocation mode (Technical Preview).....34
Setting capacity using mixed resource allocation mode (Technical Preview).....35
Changing resource allocation mode.....36
Starting and stopping queues.....37
Deleting queues.....37
Setting queue priorities.....38

Configuring scheduler properties at the global level.....38
Setting global maximum application priority.....41
Configuring preemption.....41
Enabling Intra-Queue preemption.....42
Enabling LazyPreemption.....43
Setting global application limits.....44
Setting default Application Master resource limit.....44
Enabling asynchronous scheduler.....44

Configuring queue mapping to use the user name from the application tag using Cloudera Manager.....45
Configuring NodeManager heartbeat.....46
Configuring data locality.....46
Setting Maximum Parallel Application.....47
Setting maximum parallel application limits.....49

Configuring per queue properties.....49
Setting user limits within a queue.....52
Setting Maximum Application limit for a specific queue.....54
Setting Application-Master resource-limit for a specific queue.....55
Setting maximum parallel application limits for a specific queue.....55
Controlling access to queues using ACLs.....55
Enabling preemption for a specific queue.....56
Enabling Intra-Queue Preemption for a specific queue.....57
Setting ordering policies within a specific queue.....57
Configure queue ordering policies.....58

Autoscaling clusters.....59
Autoscaling behavior.....60
Configuring autoscaling.....61

Dynamic Queue Scheduling.....64
Creating a new Dynamic Configuration.....65
Managing Dynamic Configurations.....70
How to read the Configurations table.....70
Handling Dynamic Configuration conflicts.....71
Revalidating Dynamic Configurations.....72
Dynamic Configurations execution log.....73

Managing placement rules.....73
Placement rule policies.....74
How to read the Placement Rules table.....76
Creating placement rules.....77
Example - Placement rules creation.....81
Reordering placement rules.....82
Editing placement rules.....83
Deleting placement rules.....83
Enabling override of default queue mappings.....83

Managing dynamic queues.....85
Managed Parent Queues.....85
Converting a queue to a Managed Parent Queue.....85
Enabling dynamic child creation in weight mode.....87
Disabling dynamic child creation in weight mode.....88
Managing dynamic child creation enabled parent queues.....88
Managing dynamically created child queues.....89
Deleting dynamically created child queues.....89
Disabling auto queue deletion globally.....90
Disabling queue auto removal on a queue level.....90
Configuring the queue auto removal expiration time.....91
Deleting dynamically created child queues manually.....91

Partition configuration.....92
Enabling node labels on a cluster to configure partition.....94
Creating partitions.....95
Assigning or unassigning a node to a partition.....95
Viewing partitions.....96
Associating partitions with queues.....97
Disassociating partitions from queues.....100
Deleting partitions.....101
Setting a default partition expression.....104
Using partitions when submitting a job.....104

Resource scheduling and management

You can manage resources for the applications running on your cluster by allocating resources through scheduling, limiting CPU usage by configuring cgroups, and partitioning the cluster into subclusters using partitions, and launching applications on Docker containers.

The *CapacityScheduler* is responsible for scheduling. The *CapacityScheduler* is used to run Hadoop applications as a shared, multi-tenant cluster in an operator-friendly manner while maximizing the throughput and the utilization of the cluster.

The *ResourceCalculator* is part of the YARN *CapacityScheduler*. If you have only one type of resource, typically a CPU virtual core (vcore), use the DefaultResourceCalculator. If you have multiple resource types, use the Dominant ResourceCalculator.

YARN resource allocation of multiple resource-types

You can manage your cluster capacity using the Capacity Scheduler in YARN. You can use the Capacity Scheduler's DefaultResourceCalculator or the DominantResourceCalculator to allocate available resources.

The fundamental unit of scheduling in YARN is the queue. The capacity of each queue specifies the percentage of cluster resources available for applications submitted to the queue. You can set up queues in a hierarchy that reflects the database structure, resource requirements, and access restrictions required by the organizations, groups, and individuals who use the cluster resources.

You can use the default resource calculator when you want the resource calculator to consider only the available memory for resource calculation. When you use the default resource calculator (DefaultResourceCalculator), resources are allocated based on the available memory.

If you have multiple resource types, use the dominant resource calculator (DominantResourceCalculator) to enable CPU, , and memory scheduling. The dominant resource calculator is based on the Dominant Resource Fairness (DRF) model of resource allocation. DRF is designed to fairly distribute CPU, and memory resources among different types of processes in a mixed-workload cluster.

For example, if User A runs CPU-heavy tasks and User B runs memory-heavy tasks, the DRF allocates more CPU and less memory to the tasks run by User A, and allocates less CPU and more memory to the tasks run by User B. In a single resource case, in which all jobs are requesting the same resources, the DRF reduces to max-min fairness for that resource.

Hierarchical queue characteristics

You must consider the various characteristics of the Capacity Scheduler hierarchical queues before setting them up.

There are two types of queues: parent queues and leaf queues.

- Parent queues enable the management of resources across organizations and sub- organizations. They can contain more parent queues or leaf queues. They do not themselves accept any application submissions directly.
- Leaf queues are the queues that live under a parent queue and accept applications. Leaf queues do not have any child queues, and therefore do not have any configuration property that ends with ".queues".
- There is a top-level parent root queue that does not belong to any organization, but instead represents the cluster itself.
- Using parent and leaf queues, administrators can specify capacity allocations for various organizations and sub-organizations.

Scheduling among queues

Hierarchical queues ensure that guaranteed resources are first shared among the sub-queues of an organization before any remaining free resources are shared with queues belonging to other organizations. This enables each organization to have control over the utilization of its guaranteed resources.

- At each level in the hierarchy, every parent queue keeps the list of its child queues in a sorted manner based on demand. The sorting of the queues is determined by the currently used fraction of each queue's capacity (or the full-path queue names if the reserved capacity of any two queues is equal).
- The root queue understands how the cluster capacity needs to be distributed among the first level of parent queues and invokes scheduling on each of its child queues.
- Every parent queue applies its capacity constraints to all of its child queues.
- Leaf queues hold the list of active applications (potentially from multiple users) and schedules resources in a FIFO (First In, First Out) manner, while at the same time adhering to capacity limits specified for individual users.

Application reservations

For a resource-intensive application, the Capacity Scheduler creates a reservation on a cluster node if the node's free capacity can meet the particular application's requirements. This ensures that the resources are utilized only by that particular application until the application reservation is fulfilled.

The Capacity Scheduler is responsible for matching free resources in the cluster with the resource requirements of an application. Many times, a scheduling cycle occurs such that even though there are free resources on a node, they are not sized large enough to satisfy the application waiting for a resource at the head of the queue. This typically happens with high-memory applications whose resource demand for Containers is much larger than the typical application running in the cluster. This mismatch can lead to starving these resource-intensive applications.

The Capacity Scheduler reservations feature addresses this issue as follows:

- When a node reports in with a finished Container, the Capacity Scheduler selects an appropriate queue to utilize the newly available resources based on capacity and maximum capacity settings.
- Within that selected queue, the Capacity Scheduler looks at the applications in a FIFO order along with the user limits. Once a needy application is found, the Capacity Scheduler tries to see if the requirements of that application can be met by the node's free capacity.
- If there is a size mismatch, the Capacity Scheduler immediately creates a reservation on the node for the application's required Container.
- Once a reservation is made for an application on a node, those resources are not used by the Capacity Scheduler for any other queue, application, or Container until the application reservation is fulfilled.
- The node on which a reservation is made reports back when enough Containers finish running such that the total free capacity on the node now matches the reservation size. When that happens, the Capacity Scheduler marks the reservation as fulfilled, removes it, and allocates a Container on the node.
- In some cases another node fulfills the resources required by the application, so the application no longer needs the reserved capacity on the first node. In this situation, the reservation is simply cancelled.

To prevent the number of reservations from growing in an unbounded manner, and to avoid any potential scheduling deadlocks, the Capacity Scheduler maintains only one active reservation at a time on each node.

Resource distribution workflow

During scheduling, queues at any level in the hierarchy are sorted in the order of their current used capacity, and the available resources are distributed among them starting with queues that are currently the most under-served.

With respect to capacities alone, the resource scheduling has the following workflow:

- The more under-served a queue is, the higher the priority it receives during resource allocation. The most under-served queue is the queue with the least ratio of used capacity as compared to the total cluster capacity.
  - The used capacity of any parent queue is defined as the aggregate sum of used capacity of all of its descendant queues, recursively.
  - The used capacity of a leaf queue is the amount of resources used by the allocated Containers of all of the applications running in that queue.
- Once it is decided to give a parent queue the currently available free resources, further scheduling is done recursively to determine which child queue gets to use the resources -- based on the previously described concept of used capacities.
- Further scheduling happens inside each leaf queue to allocate resources to applications in a FIFO order.
  - This is also dependent on locality, user level limits, and application limits.
  - Once an application within a leaf queue is chosen, scheduling also happens within the application. Applications may have different priorities of resource requests.
- To ensure elasticity, capacity that is configured but not utilized by any queue due to lack of demand is automatically assigned to the queues that are in need of resources.

Resource Distribution Process in Relative Mode - Example

To understand the resource distribution workflow, consider the example of a 100-node cluster, each with 10 GB of memory allocated for YARN containers, for a total cluster capacity of 1000 GB (1 TB).

For example, in the Relative allocation mode, the "engineering" organization is assigned 60% of the cluster capacity, that is, an absolute capacity of 600 GB. Similarly, the "support" organization is assigned 100 GB, and the "marketing" organization gets 300 GB.

Under the "engineering" organization, capacity is distributed between the "development" team and the "qa" team in a 1:4 ratio. So "development" gets 120 GB, and 480 GB is assigned to "qa".

Now consider the following timeline of events:

- Initially, the entire "engineering" queue is free with no applications running, while the "support" and "marketing" queues are utilizing their full capacities.
- Users Sid and Hitesh first submit applications to the "development" leaf queue. Their applications are elastic and can run with either all of the resources available in the cluster, or with a subset of cluster resources (depending upon the state of the resource-usage).
  - Even though the "development" queue is allocated 120 GB, Sid and Hitesh are each allowed to occupy 120 GB, for a total of 240 GB.
  - This can happen despite the fact that the "development" queue is configured to be run with a capacity of 120 GB. Capacity Scheduler allows elastic sharing of cluster resources for better utilization of available cluster resources. Since there are no other users in the "engineering" queue, Sid and Hitesh are allowed to use the available free resources.
- Next, users Jian, Zhijie and Xuan submit more applications to the "development" leaf queue. Even though each is restricted to 120 GB, the overall used capacity in the queue becomes 600 GB -- essentially taking over all of the resources allocated to the "qa" leaf queue.
- User Gupta now submits an application to the "qa" queue. With no free resources available in the cluster, the user's application must wait.
  - Given that the "development" queue is utilizing all of the available cluster resources, Gupta may or may not be able to immediately get back the guaranteed capacity of his "qa" queue -- depending upon whether or not preemption is enabled.
- As the applications of Sid, Hitesh, Jian, Zhijie, and Xuan finish running and resources become available, the newly available Containers will be allocated to Gupta’s application.

This will continue until the cluster stabilizes at the intended 1:4 resource usage ratio for the "development" and "qa" queues.

From this example, you can see that it is possible for abusive users to submit applications continuously, and thereby lock out other queues from resource allocation until Containers finish running or get preempted. To avoid this scenario, Capacity Scheduler supports limits on the elastic growth of any queue. For example, you can restrict the "development" queue from monopolizing the "engineering" queue capacity by setting the maximum capacity property.

To set the maximum capacity property based on this example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the development queue and select Edit Child Queues option.
3. Enter 40 in the Configured Value field.
4. Click Save.

Once this is set, users of the "development" queue can still go beyond their capacity of 120 GB, but they will not be allocated any more than 40% of the "engineering" parent queue's capacity (that is, 40% of 600 GB = 240 GB).

The capacity and maximum-capacity properties can be used to control sharing and elasticity across the organizations and sub-organizations utilizing a YARN cluster. Administrators should balance these properties to avoid strict limits that result in a loss of utilization, and to avoid excessive cross-organization sharing.

Resource allocation overview

You can allocate resources to queues by using either the Absolute mode, in which the actual units of vCores and memory resources are allocated, the Relative mode, in which resources are allocated as a percentage of total resources, or in the Weight mode, in which resources are allocated as a fraction of total resources or in the Mixed mode, in which you specify the resources in a combination of these allocation modes.

- Relative mode: In the YARN Queue Manager UI, the capacity value is shown as a percentage. When editing queue resources, capacities are specified in percentages. The capacity allocated to a set of sibling queues must equal 100% of the parent.
- Absolute mode: In the YARN Queue Manager UI, the capacity value is shown in units. When editing queue resources, a separate tab is displayed for each resource type. The total allocated capacity must be less than or equal to the capacity of the parent.
- Weight mode: In the YARN Queue Manager UI, the capacity value is shown as a numerical value and is suffixed with “w”. For example, 5w. When editing queue resources, capacities are specified in weights or fractions of total resources. The resources are divided based on how the queue's weight relates to the sum of configured weights under the parent.

  Note: If you have an existing managed queue in Relative mode, then conversion to Weight mode is not allowed. You must delete the managed parent queue and then convert from Relative mode to Weight mode.
- Mixed mode (Technical Preview): In this mode, you can specify the resources in mixed types by updating the XML configuration usingCloudera Manager. You can specify the actual units of vcores and memory resources for each queue or specify the percentage of the total resources used by each queue or specify a weight for each queue. You can use a combination of these allocation modes. The queues under one parent can also mix their modes. For more information, see *Setting capacity using mixed resource allocation mode*.

  Important: This is a technical preview feature and is considered under development. Do not use this in your production systems. If you have feedback, contact Support by logging a case on our Cloudera Support Portal. Technical preview features are not guaranteed troubleshooting and fixes.

Related Information

Changing resource allocation mode

Setting capacity using mixed resource allocation mode (Technical Preview)

Cloudera Support Portal

Adding queues using YARN Queue Manager UI

Configuring cluster capacity with queues

Use CPU scheduling

Cgroups with CPU scheduling helps you effectively manage mixed workloads.

Note: You should use CPU scheduling only in a Linux environment, because there is no isolation mechanism (cgroups equivalent) for Windows.

MapReduce jobs only

If you primarily run MapReduce jobs on your cluster, enabling CPU scheduling does not change performance much. The dominant resource for MapReduce is memory, so the DRF scheduler continues to balance MapReduce jobs in a manner similar to the default resource calculator. In the case of a single resource, the DRF reduces to max-min fairness for that resource.

Mixed workloads

An example of a mixed workload is a cluster that runs both MapReduce and Storm on YARN. MapReduce is not CPU-constrained, but Storm on YARN is; its containers require more CPU than memory. As you add Storm jobs along with MapReduce jobs, the DRF scheduler tries to balance memory and CPU resources, but you might see some performance degradation in as a result. As you add more CPU-intensive Storm jobs, individual jobs start to take longer to run as the cluster CPU resources are consumed.

To solve this problem, you can use cgroups along with CPU scheduling. Using cgroups provides isolation for CPU-intensive processes such as Storm on YARN, thereby enabling you to predictably plan and constrain the CPU-intensive Storm containers.

You can also use partitions in conjunction with CPU scheduling and cgroups to restrict Storm on YARN jobs to a subset of cluster nodes.

Configure CPU scheduling and isolation

You can configure CPU scheduling on your cluster to allocate the best possible nodes having the required CPU resources for application containers.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for Resource Calculator Class.
4. Select the org.apache.hadoop.yarn.util.resource.DominantResourceCalculator option.
5. Search for yarn.nodemanager.resource.cpu-vcores and set the number of vcores to match the number of physical CPU cores on the NodeManager host by providing the number of physical cores.

Related Information

Limit CPU usage with Cgroups

Using GPU on YARN

Enable Cgroups

Use CPU scheduling with distributed shell

You can run the distributed shell by specifying resources other than memory and vCores. The following is an example for distributed shell but you can use CPU scheduling with other frameworks as well.

Procedure

- Use the following command to allocate two cores for two containers:

```
yarn jar </opt/cloudera/parcels/<CDH-version>/lib/hadoop-yarn/hadoop-yarn-
applications-distributedshell.jar> \
                   -jar </opt/cloudera/parcels/<CDH-version>/lib/hadoop-ya
rn/hadoop-yarn-applications-distributedshell.jar> \
                   -shell_command "sleep 120" \
                   -container_resources memory-mb=3072,vcores=2 \
                   -num_containers 2
```

Use GPU scheduling

On your cluster, you can configure GPU scheduling and isolation. Currently only Nvidia GPUs are supported in YARN. You can useCloudera Manager to configure GPU scheduling on your cluster.

Configure GPU scheduling and isolation

You can configure GPU scheduling and isolation on your cluster. Currently only Nvidia GPUs are supported in YARN.

Before you begin

- YARN NodeManager must be installed with the Nvidia drivers.

Procedure

1. In Cloudera Manager, navigate to Hosts Hosts Configuration .
2. Search for cgroup.
3. Select the Enable Cgroup-based Resource Management checkbox.
4. Click Save Changes.
5. Navigate to YARN Configuration
6. Search for cgroup.
7. Find the Use CGroups for Resource Management property and enable it for the applicable clusters.
8. Find the Always use Linux Container Executor property and enable it for the applicable clusters.
9. Search for gpu.
10. Find the Enable GPU Usage property and select the NodeManager Default Group checkbox.

    Note:

    GPU usage is not supported when using cgroup v2.

11. Find the NodeManager GPU Devices Allowed property and define the GPU devices that are managed by YARN using one of the following ways.
    - Use the default value, auto, for auto detection of all GPU devices. In this case all GPU devices are managed by YARN.
    - Manually define the GPU devices that are managed by YARN.
12. Find the NodeManager GPU Detection Executable property and define the location of nvidia-smi. By default, this property has no value and it means that YARN checks the following paths to find nvidia-smi:
    - /usr/bin
    - /bin
    - /usr/local/nvidia/bin
13. Click Save Changes.
14. Click the Stale Configuration: Restart needed button on the top of the page.

15. Click Restart Stale Services.

Note that this step restarts all services with stale configurations.

16. Select Re-deploy client configuration and click Restart Now.

If the NodeManager fails to start, the following error is displayed:

```
INFO gpu.GpuDiscoverer (GpuDiscoverer.java:initialize(240)) - Trying to disc
over GPU information ... WARN gpu.GpuDiscoverer (GpuDiscoverer.java:initiali
ze(247)) - Failed to discover GPU information from system, exception message
:ExitCodeException exitCode=12: continue...
```

Fix the error by exporting the LD_LIBRARY_PATH in the yarn -env.sh using the following command:
`export LD_LIBRARY_PATH=/ usr/local/nvidia/lib:/usr/local/nvidia/lib64: $LD_LIBRARY_PATH`

Use GPU scheduling with distributed shell

You can run the distributed shell by specifying resources other than memory and vCores. The following is an example for distributed shell but you can use GPU scheduling with other frameworks as well.

Procedure

- Use the following command to run the distributed shell and GPU without a Docker container:

```
yarn jar </opt/cloudera/parcels/<CDH-version>/lib/hadoop-yarn/hadoop-yarn-
applications-distributedshell.jar> \
                   -jar </opt/cloudera/parcels/<CDH-version>/lib/hadoop-ya
rn/hadoop-yarn-applications-distributedshell.jar> \
                   -shell_command /usr/local/nvidia/bin/nvidia-smi \
                   -container_resources memory-mb=3072,vcores=1,yarn.io/g
pu=2 \
                   -num_containers 2
```

You receive output similar to the following:

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.66                 Driver Version: 375.66                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 0000:04:00.0     Off |                    0 |
| N/A   30C    P0    24W / 250W |      0MiB / 12193MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla P100-PCIE...  Off  | 0000:82:00.0     Off |                    0 |
| N/A   34C    P0    25W / 250W |      0MiB / 12193MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Use FPGA scheduling

You can use FPGA as a resource type.

A Field Programmable Gate Array (FPGA) is a card that contains a matrix of configurable logic blocks, mostly logic gates. FPGA can be reprogrammed, meaning that the interconnections among the gates and integrated devices can be re-wired. That is in contrast to a regular CPU, as the internal wiring and connections inside a CPU cannot be changed: only the software that runs on it can be changed, but the hardware is always the same. FPGA, on the other hand, allows changes on a circuit level.

An FPGA device does not contain only rewirable logic gates (like AND or OR gates), but also memory, DSP, chip, external connectors and more.

Programming FPGAs

FPGAs can be programmed in various ways. The most popular methods are VHDL and Verilog.

Intel FPGA cards allow the users to upload so-called OpenCL kernels to perform certain computations simultaneously and therefore rapidly. OpenCL is a framework and a set of standards that is currently developed and maintained by the Khronos Group. For more information, see Khronos' OpenCL documentation.

Configure FPGA scheduling and isolation

You can enable and configure FPGA as a resource type using Cloudera Manager.

Before you begin

Ensure that the FPGA Cloudera Runtime or SDK is installed in such a way that the aocl command is executable by the NodeManager. For more information about how to install FPGA, see Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Select FPGA Management category under Filters.
4. Find the Enable FPGA Usage property and select the NodeManagers that can provide FPGA devices to YARN applications that request them.
5. Use the Path to FPGA (aocl) tool property to specify the full local path to the Intel aocl tool installed on the applicable nodes.
6. Use the Allowed FPGA devices to specify the FPGA devices which can be managed by YARN NodeManager.

   Valid values are the following:

   - Auto: Default value. All available FPGA devices are allowed.
   - Comma separated list: List the minor number of the allowed FPGAs. For example: 0,1
7. Use the List of available FPGA devices property to manually specify the available FPGA devices.

   Provide the value in the following format: deviceA/major_number:minor_number,deviceB/major_number:minor_number

   For example: acl0/238:0,acl1/238:1

   The acl numbers are displayed using the aocl diagnose command.

   The major and minor device numbers can usually be determined by listing the device files under /dev. Every card has a corresponding device file. The ls -l command shows both numbers.
8. Use the FPGA device major number property to provide the major device number of the FPGA card.

   This property is used in the container-executor.cfg file.

   Important: Only one major number can be specified for each node, meaning that a node can have only one kind of FPGA card.

9. Use the FPGA initializer script property to provide the path to the shell script that sets up the environment for the FPGA device.

   The initializer script sets up various environment variables, and sources other scripts allowing the aocl tool to run properly. The contents of this script depends heavily on the installation. For example, in case of Intel Processing Accelerator Card (PAC), it is necessary to source init_env.sh and init_opencl.sh.

   The following environment variables are possibly required to be exported:

   - AOCL_BOARD_PACKAGE_ROOT
   - CL_CONTEXT_COMPILER_MODE_INTELFPGA

   Please consult the vendor's documentation for further details.
10. Click Save Changes.
11. Restart the affected NodeManagers.

Results

FPGA support is enabled and configured.

What to do next

Request FPGA resource by specifying yarn.io/fpga as a resource.

Use FPGA with distributed shell

Once FPGA support is enabled, an FPGA resource can be requested by specifying yarn.io/fpga as a resource.

The following is a distributed shell example:

```
yarn jar /path/to/hadoop-yarn-applications-distributedshell.jar
-jar /path/to/hadoop-yarn-applications-distributedshell.jar
-shell_command "date"
-container_resources memory-mb=2048,vcores=1,yarn.io/fpga=1
-num_containers 1
```

This command runs the date command on a node which has an FPGA card installed.

Limit CPU usage with Cgroups

You can use cgroups to limit CPU usage in a Hadoop Cluster.

You can use cgroups to isolate CPU-heavy processes in a Hadoop cluster. If you are using CPU Scheduling, you should also use cgroups to constrain and manage CPU processes. If you are not using CPU Scheduling, do not enable cgroups.

When you enable CPU Scheduling, queues are still used to allocate cluster resources, but both CPU and memory are taken into consideration using a scheduler that utilizes Dominant Resource Fairness (DRF). In the DRF model, resource allocation takes into account the dominant resource required by a process. CPU-heavy processes (such as Storm-on-YARN) receive more CPU and less memory. Memory-heavy processes (such as MapReduce) receive more memory and less CPU. The DRF scheduler is designed to fairly distribute memory and CPU resources among different types of processes in a mixed- workload cluster.

Cgroups compliments CPU Scheduling by providing CPU resource isolation. It enables you to set limits on the amount of CPU resources granted to individual YARN containers, and also lets you set a limit on the total amount of CPU resources used by YARN processes.

Cgroups represents one aspect of YARN resource management capabilities that includes CPU Scheduling, partitions, archival storage, and memory as storage. If CPU Scheduling is used, cgroups should be used along with it to constrain and manage CPU processes.

Related Information

Configure CPU scheduling and isolation

Use Cgroups

You can use strict cgroups CPU limits to constrain CPU processes in mixed workload clusters.

One example of a mixed workload is a cluster that runs both MapReduce and Storm-on-YARN. MapReduce is not CPU-constrained (MapReduce containers do not ask for much CPU). Storm-on-YARN is CPU-constrained: its containers ask for more CPU than memory. As you start adding Storm jobs along with MapReduce jobs, the DRF scheduler attempts to balance memory and CPU resources, but as more CPU-intensive Storm jobs are added, they may begin to take up the majority of the cluster CPU resources.

You can use cgroups along with CPU scheduling to help manage mixed workloads. cgroups provide isolation for CPU-heavy processes such as Storm-on-YARN, thereby enabling you to predictably plan and constrain the CPU-intensive Storm containers.

When you enable strict cgroup CPU limits, each resource gets only what it asks for, even if there is extra CPU available. This is useful for scenarios involving charge-backs or strict SLA enforcement, where you always need to know exactly what percentage or CPU is being used.

Also, enabling strict CPU limits would make job performance predictable, whereas without setting strict limits a CPU-intensive job would run faster when the cluster was not under heavy use, but slower when more jobs were running in the cluster. Strict CPU limits would therefore also be useful for benchmarking.

You can also use partitions in conjunction with cgroups and CPU scheduling to restrict Storm-on-YARN jobs to a subset of cluster nodes.

If you are using cgroups and want more information on CPU performance, you can review the statistics available in the /cgroup/cpu/yarn/cpu.stat file.

Enable Cgroups

You can enable CPU Scheduling to enable cgroups. You must configure certain properties in yarn-site.xml on the ResourceManager and NodeManager hosts to enable cgroups.

About this task

cgroups is a Linux kernel feature. cgroups is supported on the following Linux operating systems:

- CentOS
- RHEL
- SUSE
- Ubuntu

Important:

The yarn.nodemanager.linux-container-executor.cgroups.mount property must be set to false. Setting this value to true is not currently supported.

Procedure

Enable cgroups

The following commands must be run on every reboot of the NodeManager hosts to set up the cgroup hierarchy. Note that operating systems use different mount points for the cgroup interface. Replace /sys/fs/cgroup with your operating system equivalent.

```
mkdir -p /sys/fs/cgroup/cpu/yarn
chown -R yarn /sys/fs/cgroup/cpu/yarn
mkdir -p /sys/fs/cgroup/memory/yarn
chown -R yarn /sys/fs/cgroup/memory/yarn
mkdir -p /sys/fs/cgroup/blkio/yarn
```

```
chown -R yarn /sys/fs/cgroup/blkio/yarn
mkdir -p /sys/fs/cgroup/net_cls/yarn
chown -R yarn /sys/fs/cgroup/net_cls/yarn
mkdir -p /sys/fs/cgroup/devices/yarn
chown -R yarn /sys/fs/cgroup/devices/yarn
```

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for Always Use Linux Container Executor and select YARN-1 (Service-Wide) option.
4. Search for NodeManager Advanced Configuration and set the below property in the NodeManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml field.

   ```
   Name:  yarn.nodemanager.linux-container-executor.group
   Value: hadoop
   yarn.nodemanager.linux-container-executor.cgroups.v2.enabled=true
   yarn.nodemanager.linux-container-executor.cgroups.mount-path=/sys/fs/cgro
   up
   ```

5. Search for CGroups and select YARN-1 (Service-Wide) option in the Use CGroups for Resource Management field.
6. Search for CGroups Hierarchy and set the NodeManager Default Group value to /hadoop-yarn.
7. Search for NodeManager Advanced Configuration and set the below property in the NodeManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml field.

   ```
   Name:  yarn.nodemanager.linux-container-executor.cgroups.mount
   Value: false
   ```

   ```
   Name:  yarn.nodemanager.linux-container-executor.cgroups.mount-path
   Value: /sys/fs/cgroup
   ```

8. (Optional) Set the percentage of CPU to be used by YARN. Search for Containers CPU Limit and set the value in the Containers CPU Limit Percentage field.

   Set the percentage of CPU that can be allocated for YARN containers. In most cases, the default value of 100% should be used. If you have another process that needs to run on a node that also requires CPU resources, you can lower the percentage of CPU allocated to YARN to free up resources for the other process.
9. (Optional) Set flexible or strict CPU limits. Search for Strict CGroup Resource Usage and select the NodeManager Default Group field.

   CPU jobs are constrained with CPU scheduling and cgroups enabled, but by default these are flexible limits. If spare CPU cycles are available, containers are allowed to exceed the CPU limits set for them. With flexible limits,

the amount of CPU resources available for containers to use can vary based on cluster usage -- the amount of CPU available in the cluster at any given time.

You can use cgroups to set strict limits on CPU usage. When strict limits are enabled, each process receives only the amount of CPU resources it requests. With strict limits, a CPU process will receive the same amount of cluster resources every time it runs.

Strict limits are not enabled (set to false) by default.

Note: Irrespective of whether this property is true or false, at no point will total container CPU usage exceed the limit set in yarn.nodemanager.resource.percentage-physical-cpu-limit.

Note: CPU resource isolation leverages advanced features in the Linux kernel. At this time, setting yarn.nodemanager.linux-container-executor.cgroups.strict-resource-usage to true is not recommended due to known kernel panics. In addition, with some kernels, setting yarn.nodemanager.resource.percentage-physical-cpu-limit to a value less than 100 can result in kernel panics. If you require either of these features, you must perform scale testing to determine if the in-use kernel and workloads are stable. As a starting point, Linux kernel version 4.8.1 works with these features. However, testing the features with the desired workloads is very important.

Related Information

Configure CPU scheduling and isolation

Cloudera Support Matrix

Managing YARN Queue Manager

YARN Queue Manager manages Capacity Scheduler configuration It also provides a user interface (YARN Queue Manager UI) that makes it easier for Hadoop operators to create, configure and manage YARN queues on a global or queue level.

As Capacity Scheduler is the supported scheduler in Cloudera, it is recommended to use YARN Queue Manager to manage the scheduler configuration.

Configuring YARN Queue Manager dependency

If you add the YARN Queue Manager service to the cluster after installing the cluster, you must configure the YARN Queue Manager dependency.

1. In Cloudera Manager, select the YARN service.
2. Click Configuration.
3. Search for Queue Manager service.

4. Select the YARN Queue Manager checkbox.

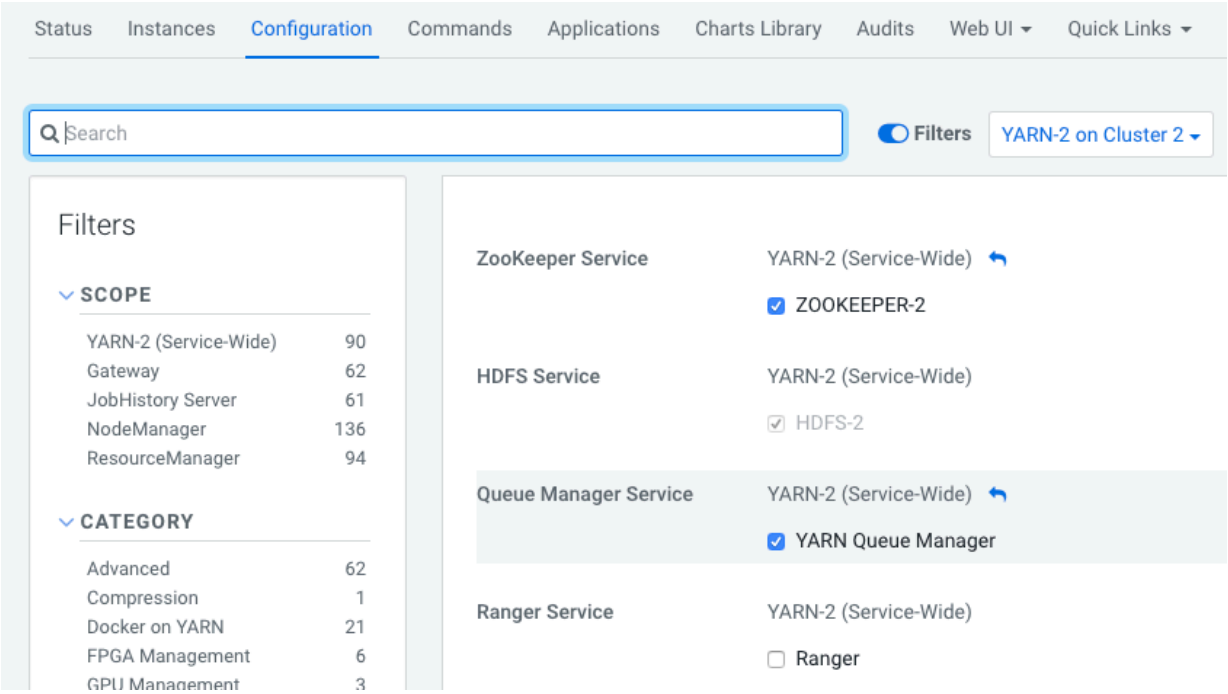

5. Click Save Changes.
6. Restart the YARN and the YARN Queue Manager services.

Updating YARN Queue Manager Database Password

Once you have configured Queue Manager, you can update the YARN Queue Manager database password manually.

Procedure

1. In Cloudera Manager navigate to Clusters YARN Queue Manager UI
2. Click Configuration.

3. Search for password.

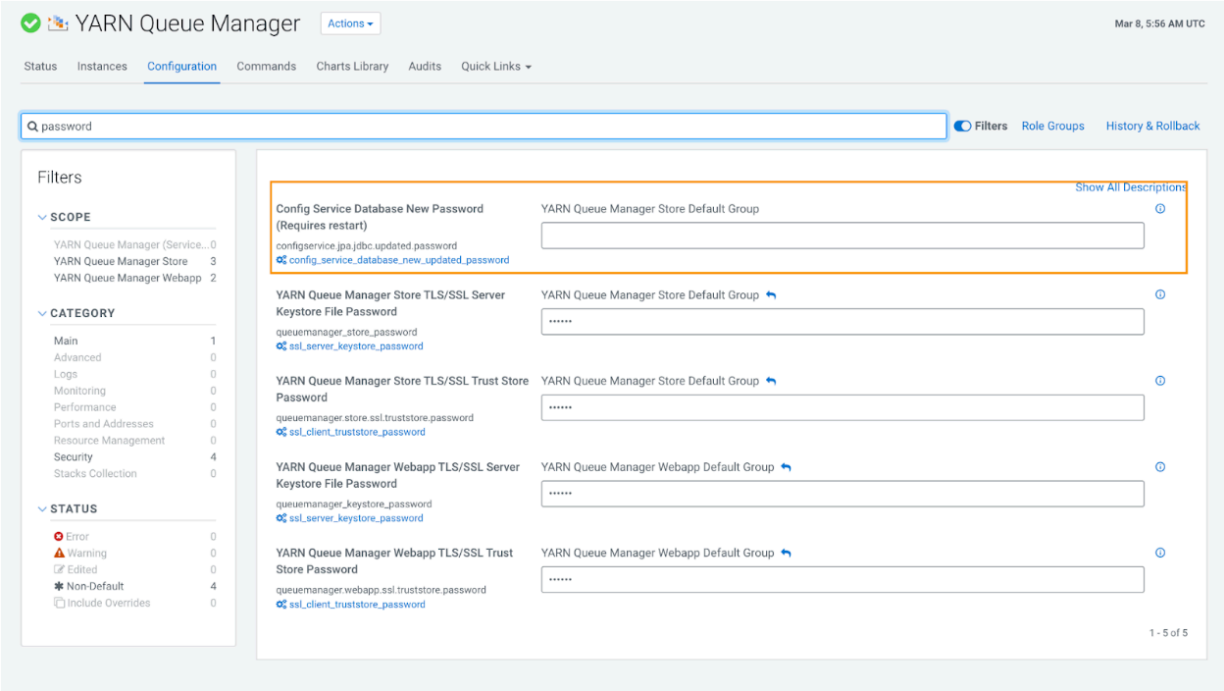

The Config Service Database New Password (Requires restart) field is displayed.

4. Enter the new password into the input field.
5. Click Save Changes.
6. Restart the YARN Queue Manager service.
   On restarting, the password of the database is updated to the new password.
7. On restarting YARN Queue Manager, a stale configuration icon is displayed.

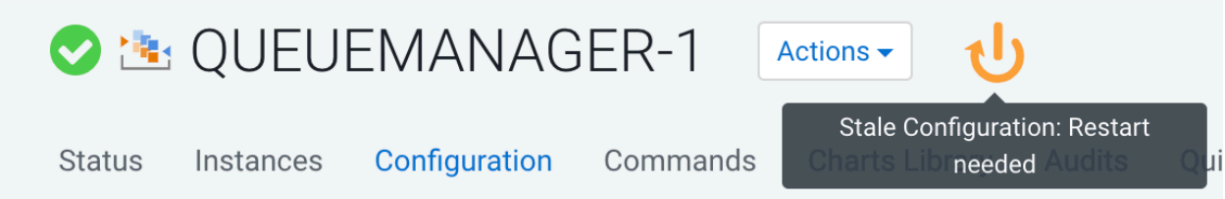

8. Restart the YARN Queue Manager service again to address this.

Accessing the YARN Queue Manager UI

Once you have enabled the Queue Manager UI, you can use it to manage your scheduler configuration.

About this task

Before you begin

- Add the YARN Queue Manager service to your cluster
- Configure the YARN Queue Manager dependency

Procedure

In Cloudera Manager navigate to Clusters YARN Queue Manager UI

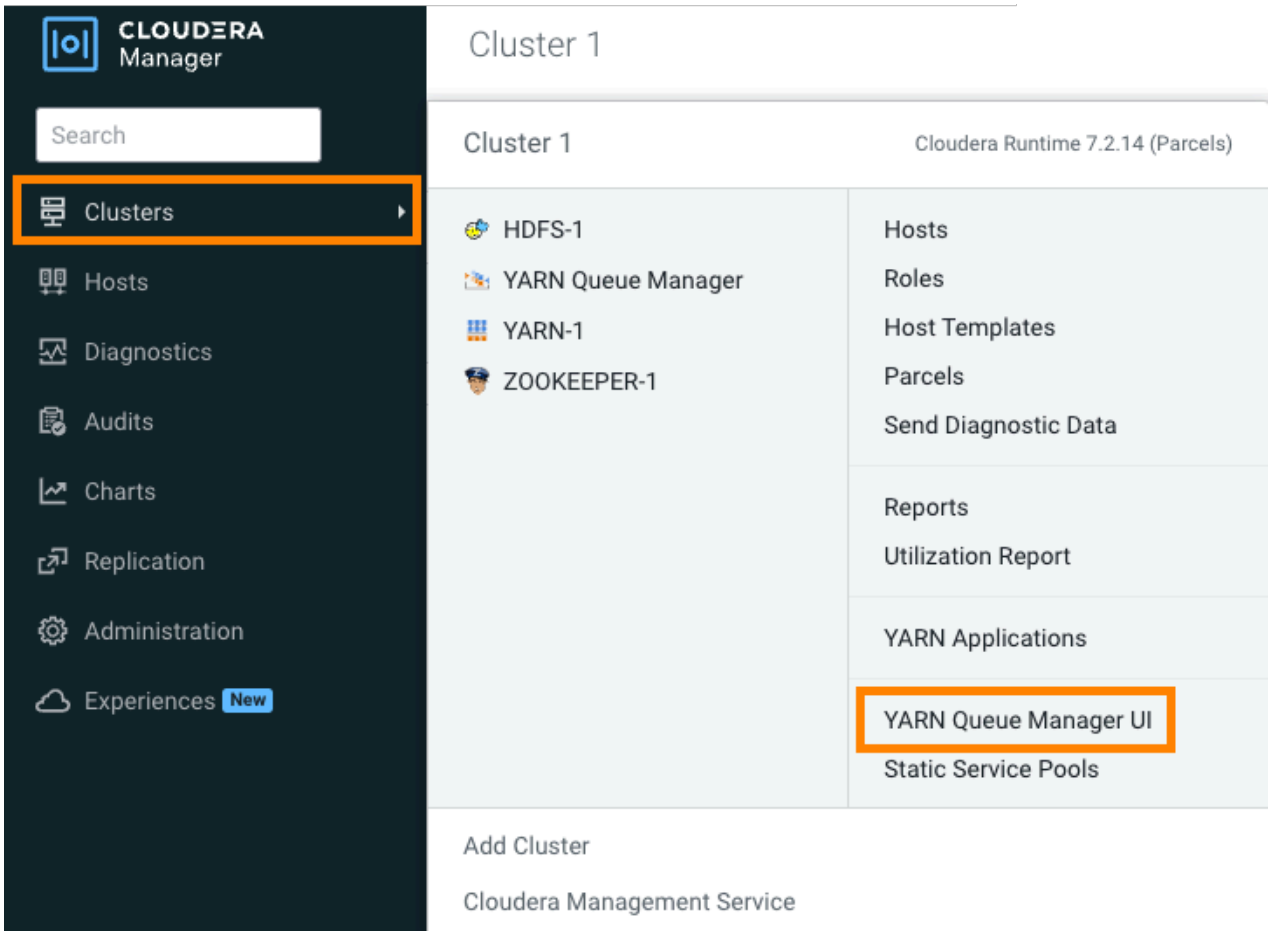

The Overview tab is opened, displaying a hierarchical view of the queues:

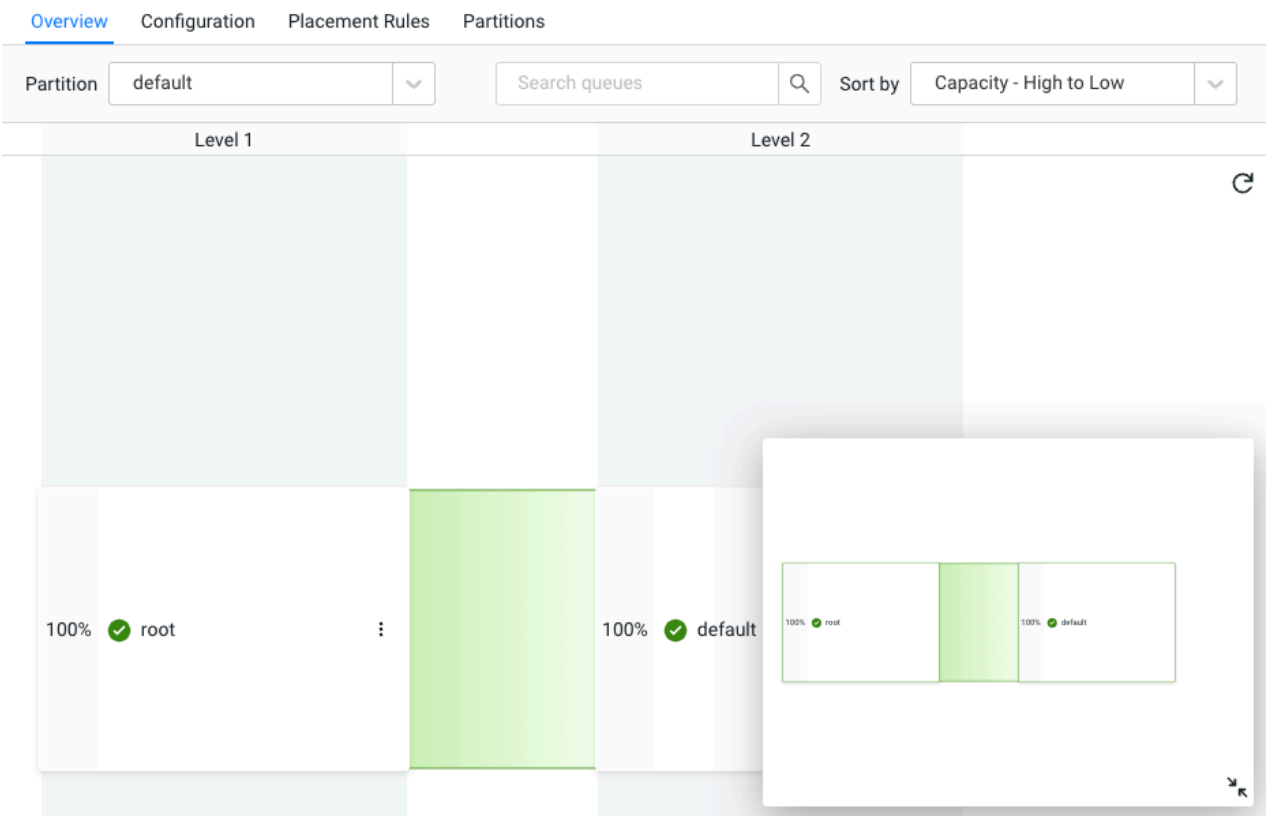

What to do next

Navigate between the difference tabs to manage queues, configure queue and global level properties, and to manage YARN features.

Related Information

Managing queues

Configuring scheduler properties at the global level

Configuring per queue properties

Managing placement rules

Partition configuration

Providing read-only access to Queue Manager UI

You can now allow non-admin users to access YARN Queue Manager in a read-only mode. You can either create a new user account with read-only role or use any existing user account with read-only role in Cloudera Manager to access YARN Queue Manager UI. In the read-only access mode, the user can view all the configurations but cannot make any changes to the configurations.

1. In Cloudera Manager, click the Administration > Users & Roles.

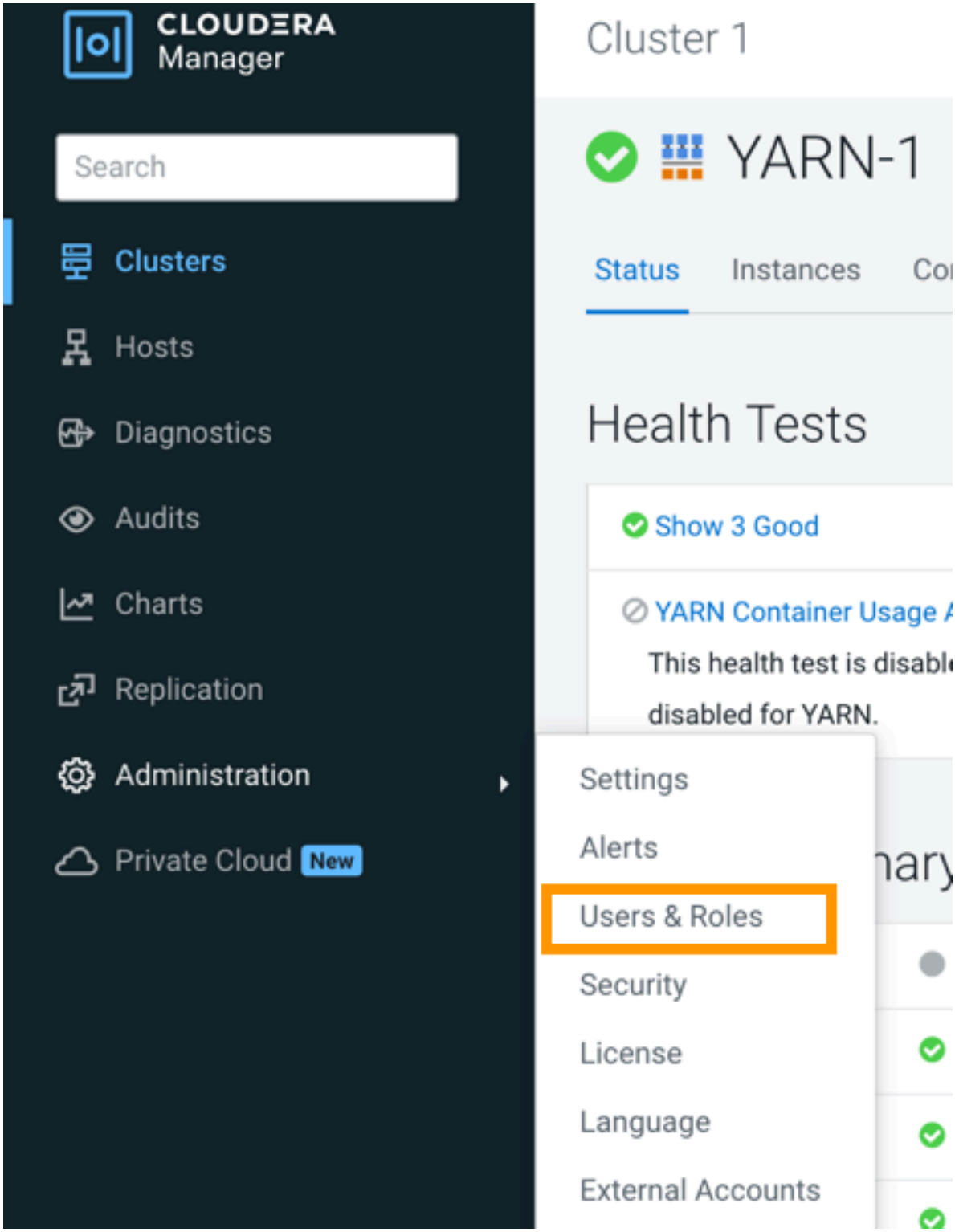

2. Provide username, password, and select Read-Only from the Roles dropdown list.

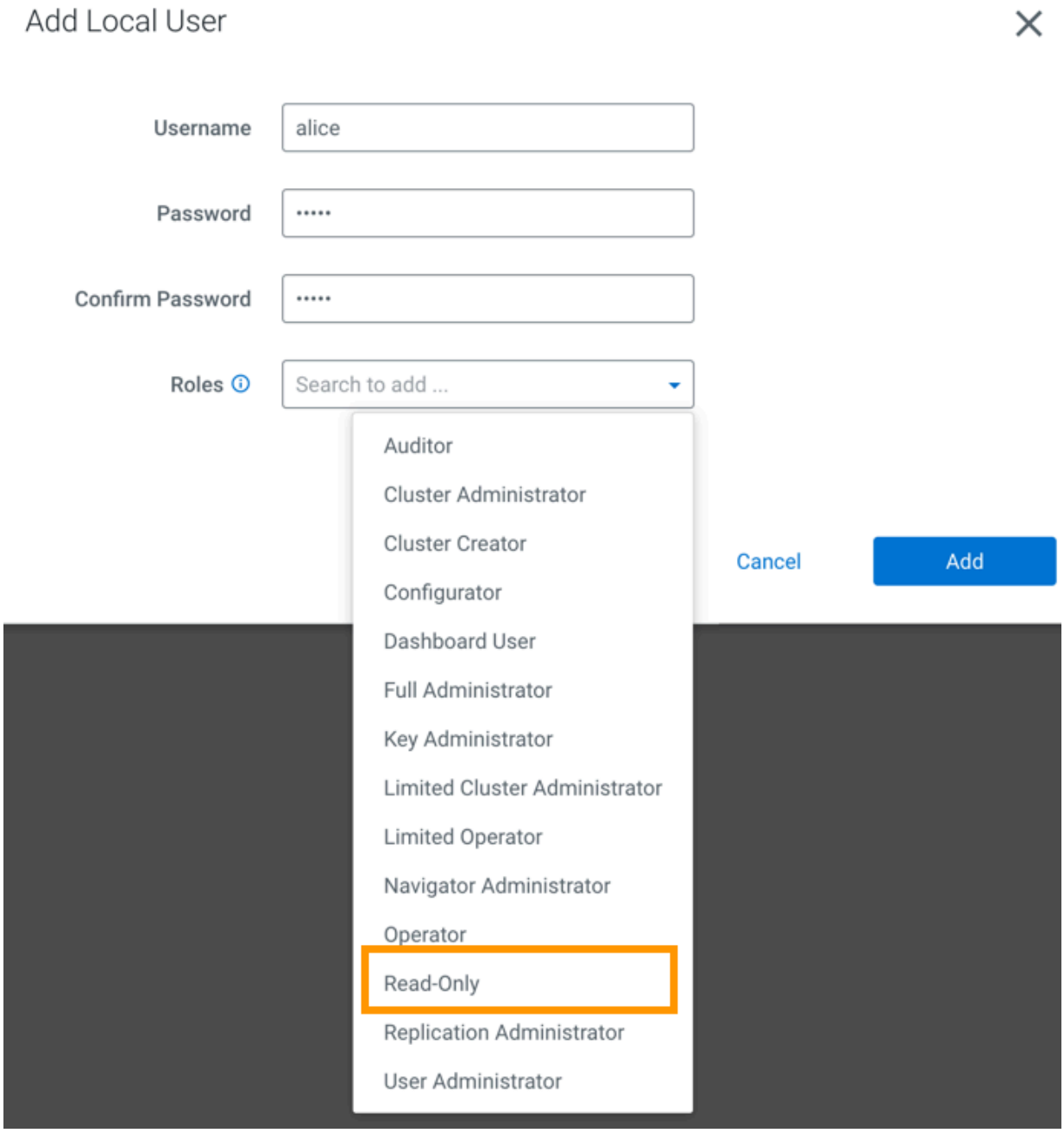

3. Click Add.

   For more information about assigning user roles, see Cloudera Manager User Roles.
4. Click on your username on the left navigation pane and select Sign Out.
5. Log in to Cloudera Manager as the newly created Read Only user.

6. Click Clusters > YARN Queue Manager UI service. The YARN Queue Manager UI is displayed without edit options.

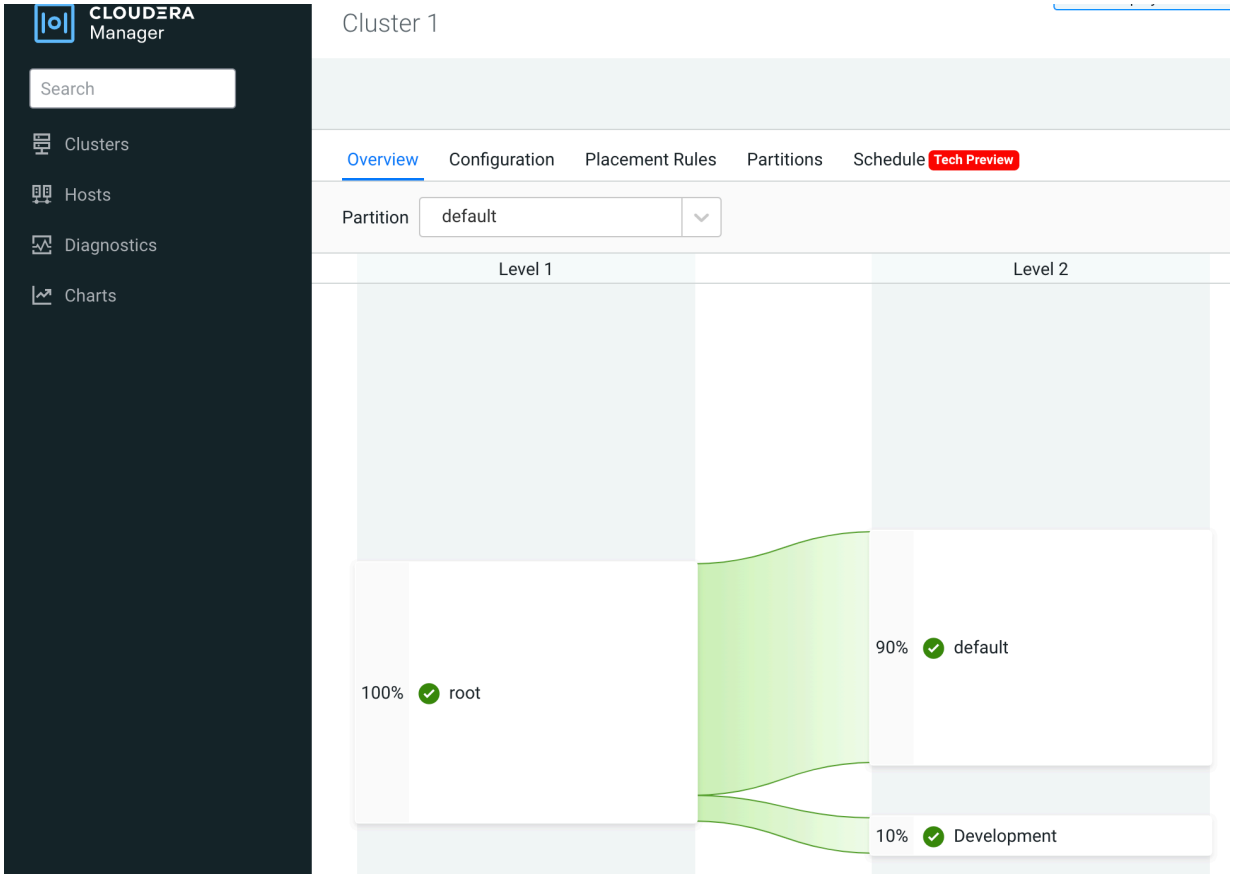

Configuring the embedded Jetty Server in Queue Manager

The configuration properties of the Jetty Server embedded in the Queue Manager can be configured using Cloudera Manager. The configuration of these properties can facilitate debugging.

Procedure

1. In Cloudera Manager, select the YARN Queue Manager service.
2. Click Configuration.
3. Search for Jetty property to list all the available Jetty Server configuration properties.

   For a comprehensive list of Jetty Server configuration properties, see *YARN Queue Manager properties reference* in the Cloudera Manager documentation.
4. Configure the Jetty Server configuration properties according to your use case.
5. Click Save Changes.
6. Restart the YARN Queue Manager service.

Managing queues

YARN Queue Manager is the queue management graphical user interface for Apache Hadoop YARN Capacity Scheduler. You can use YARN Queue Manager UI to manage your cluster capacity using queues to balance resource requirements of multiple applications from various users. Using YARN Queue Manager UI, you can set scheduler level properties and queue level properties.

You can view, sort, search, and filter queues using YARN Queue Manager UI. Queue Manager stores history of previous changes and provides the ability to view the changes of each version in the Overview and Scheduler Configuration tabs. The previous versions will be in the read-only mode and you must select the latest version to make changes.

The fundamental unit of scheduling in YARN is a queue. The capacity of each queue specifies the percentage of cluster resources that are available for applications submitted to the queue. Capacity Scheduler queues can be set up in a hierarchy that reflects the database structure, resource requirements, and access restrictions required by the various organizations, groups, and users that utilize cluster resources.

For example, suppose that a company has three organizations: Engineering, Support, and Marketing. The Engineering organization has two sub-teams: Development and QA. The Support organization has two sub-teams: Training and Services. And finally, the Marketing organization is divided into Sales and Advertising. The following image shows the queue hierarchy for this example:

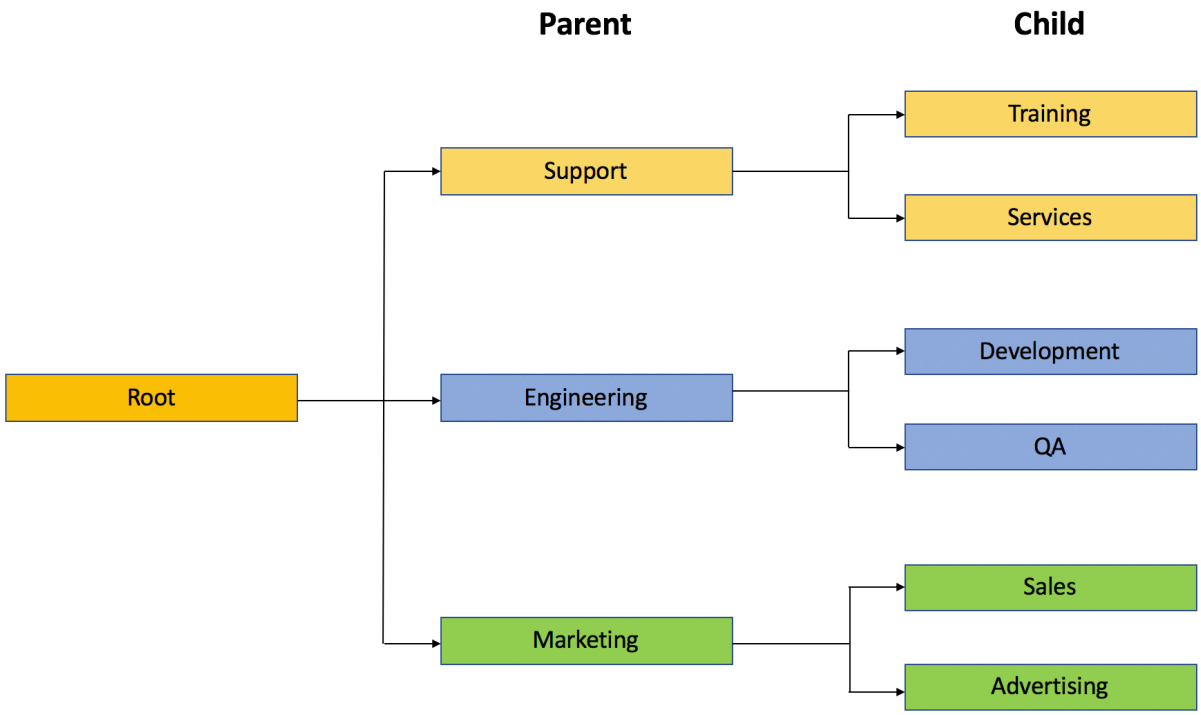

Each child queue is tied to its parent queue and the top-level "support", "engineering", and "marketing" queues would be tied to the "root" queue.

Adding queues using YARN Queue Manager UI

You can add queues to the predefined queue called root from the Yarn Queue Manager UI. The Capacity Scheduler has a predefined queue called root. All queues in the system are children of the root queue. Each child queue is tied to its parent queue, but children queues do not inherit properties directly from the parent queue unless otherwise specified.

Procedure

1. In Cloudera Manager, navigate to Clusters YARN Queue Manager UI .

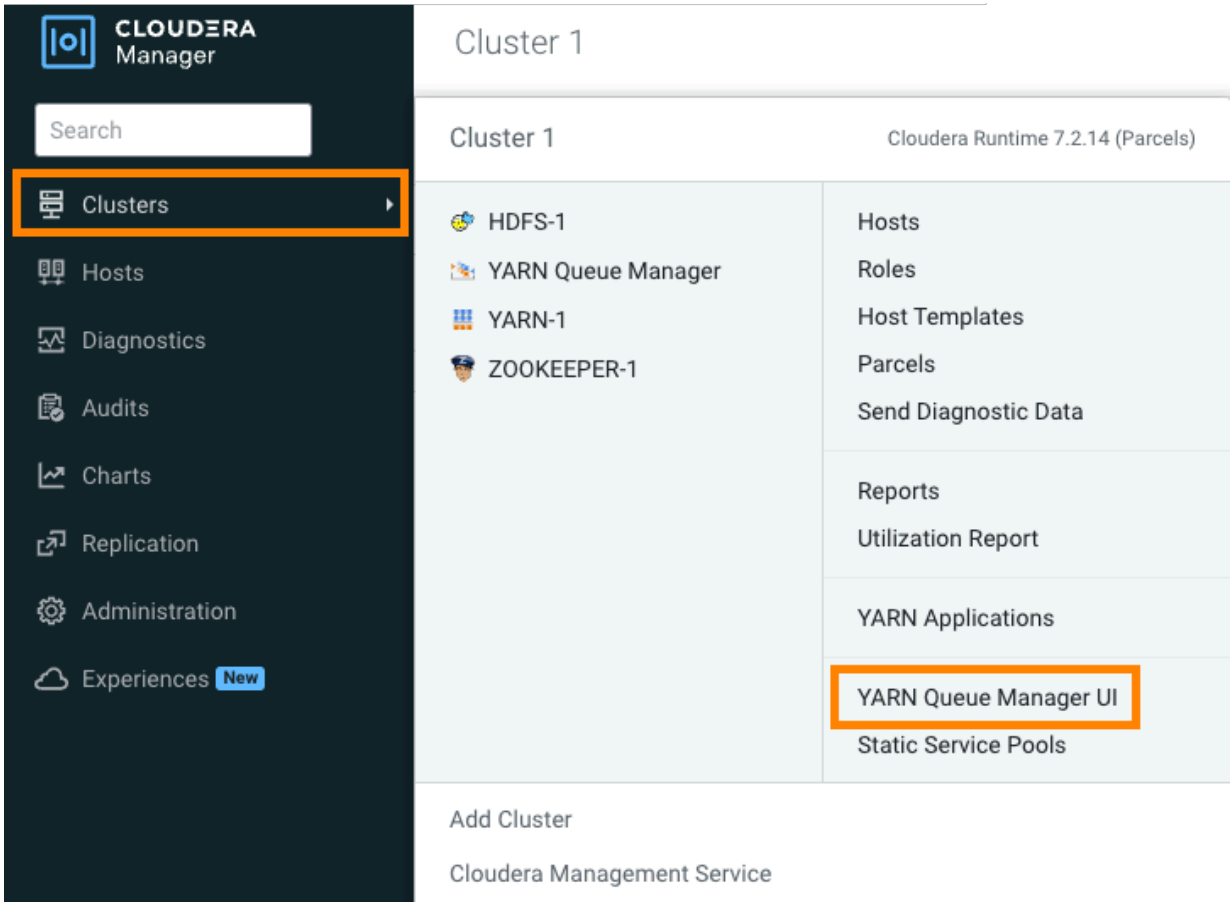

2. Click on the three vertical dots on the root and select the Add Child Queue option.

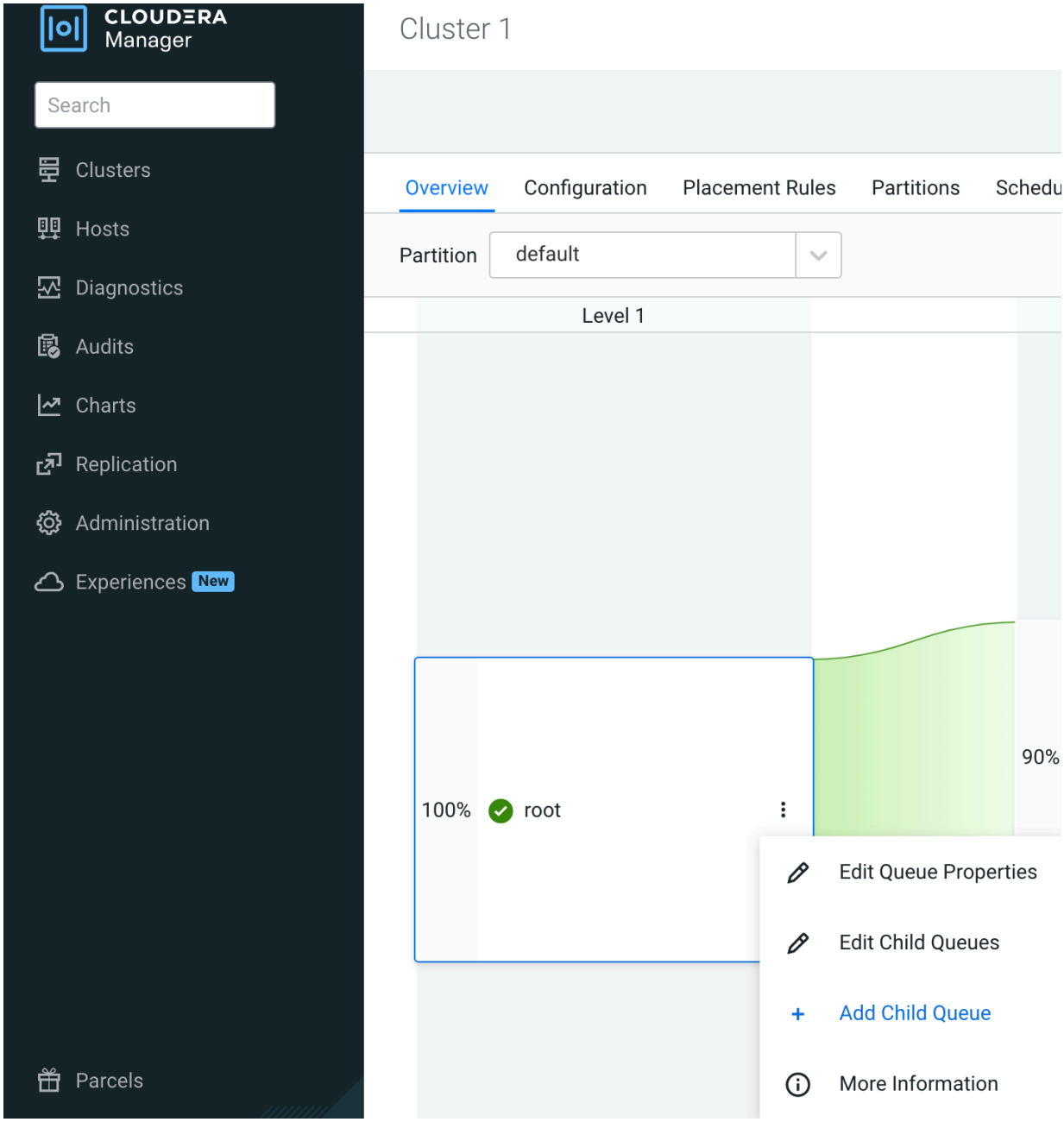

3. Enter the information based on the Relative or Absolute or Weight allocation mode.
   - Absolute allocation mode: Enter the name of the queue and units of memory in MiB in the Memory tab. Enter the number of cores in the vCores tab.

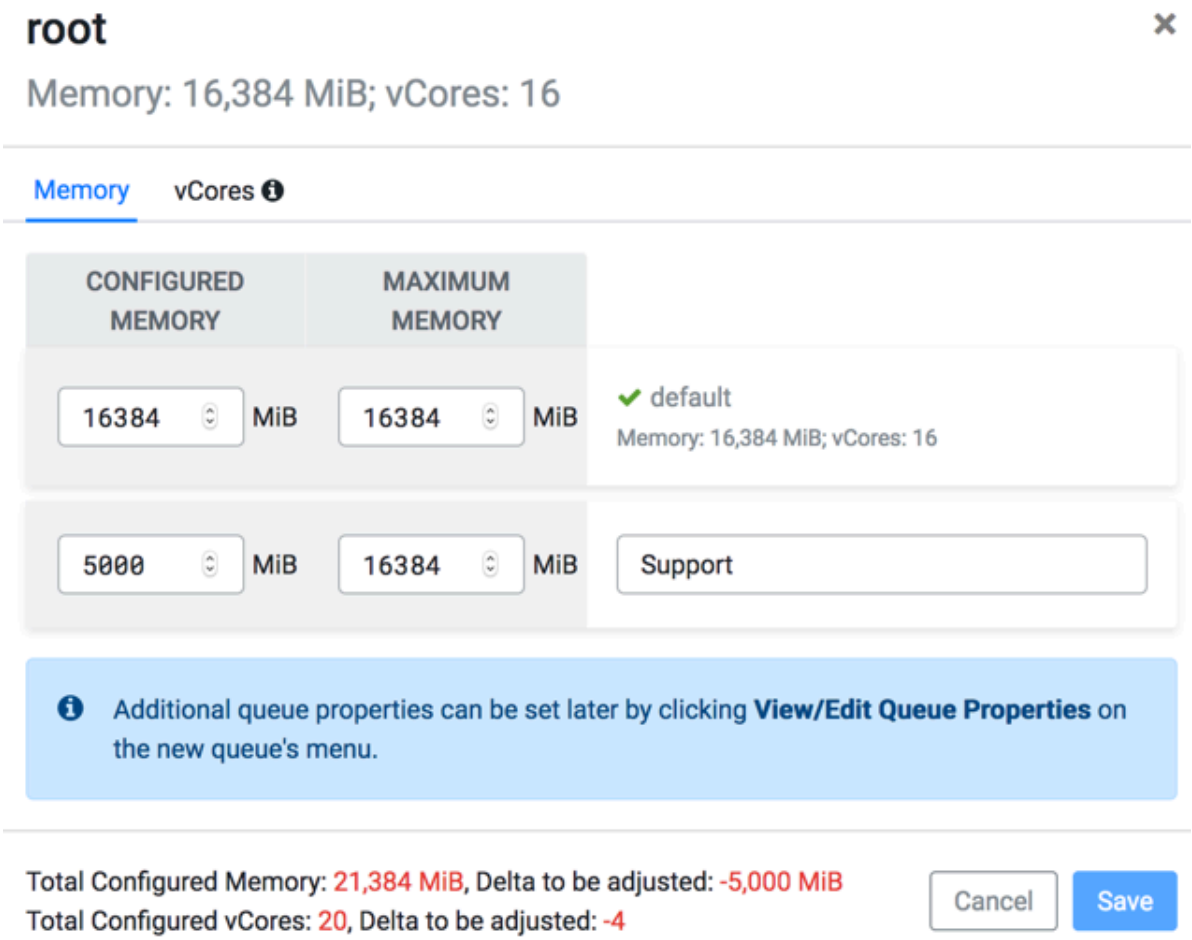

   Note: When decreasing a queue's configured resource capacity to allocate resources for the new sibling queue you must update resource capacities starting from the lowest queue level. This is because a parent queue's capacity cannot be decreased until the total resource capacity of all of its child queues is reduced. You might have to change both the minimum and maximum values.
   - Relative allocation mode: Enter the name of the queue, Configured Capacity, and Maximum Capacity values for the queue.

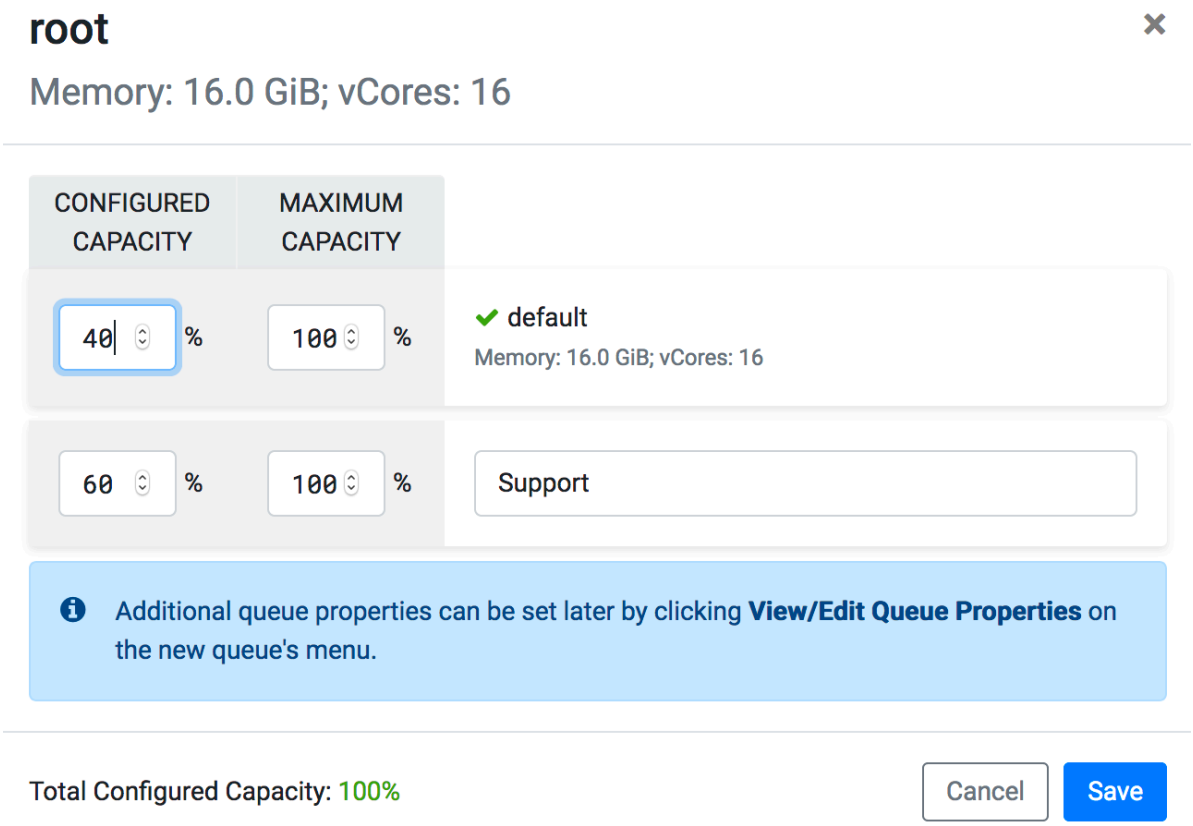

- Weight allocation mode: Enter the name of the queue and the fraction of the resource of the resource in the Configured Weight for the queue.

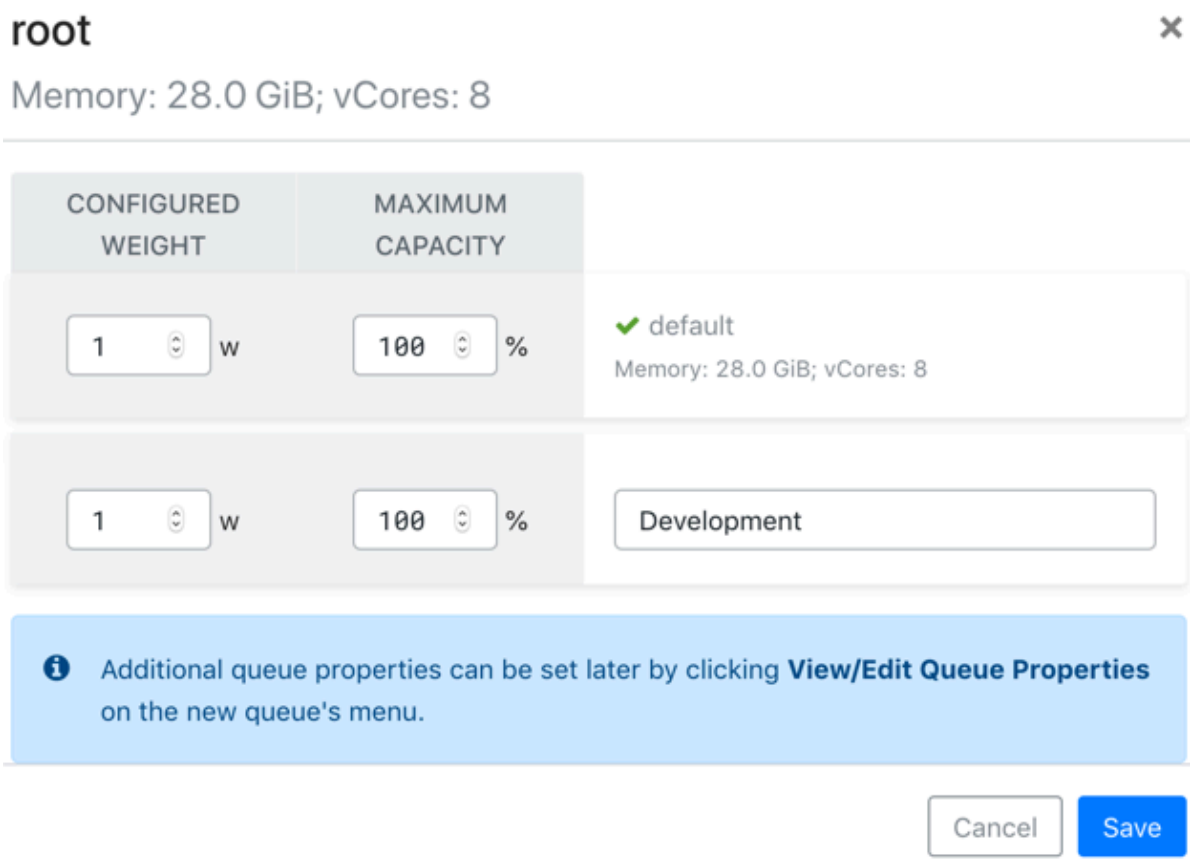

4. Click Save.

   You can continue to add more parent and child queues by following the same steps.

Related Information

Resource allocation overview

Changing resource allocation mode

Configuring cluster capacity with queues

You can manage your cluster capacity using queues to balance resource requirements of multiple applications from various users.

You can use the Capacity Scheduler to share cluster resources using FIFO (first in, first out) queues.

You can configure queues either by specifying the percentage of the capacity using the Relative mode or the actual units of vCores and memory using the Absolute mode or the fraction of the total capacity using the Weight mode. If you are upgrading your cluster, Weight mode is the default mode. If you are installing and configuring the cluster freshly, Relative mode is the default mode. You can change the allocation mode to the Absolute mode.

You can submit applications to different queues at multiple levels in the queue hierarchy if the capacity is available on the nodes in the cluster. Because total cluster capacity can vary, capacity configuration values are expressed using percentages, units, or fractions.

Example – Configuring capacity using the Relative mode

You can specify the capacity property to allocate a floating-point percentage value of cluster capacity to a queue. The following configurations divide the cluster resources between the "engineering", "support", and "marketing" organizations in a 6:1:3 ratio (60%, 10%, and 30%).

To specify the capacity property based on the above example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the root and select Edit Child Queues option.
3. Enter the Configured Capacity of "engineering" to 60, "support" to 10, and "marketing" to 30.
4. Click Save.

Example – Configuring capacity using the Absolute mode

You can specify the capacity in units of memory in MiB and the number of cores to a queue. If the total units of memory is 16384 MiB and 16 cores, and the the cluster resources between the "engineering", "support", and "marketing" organizations are allocated as follows:

Organization	Memory	vCores
engineering	9830	10
support	1638	2
marketing	4916	4

To specify the capacity property based on the above example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the root and select Edit Child Queues option.
3. Enter the Configured Memory of "engineering" to 9830, "support" to 1638, and "marketing" to 4916.
4. Enter the Configured vCores of "engineering" to 10, "support" to 2, and "marketing" to 4.
5. Click Save.

Example – Configuring capacity using the Weight mode

You can specify the capacity in fractions of total resources. The resources are divided based on how the queue's weight relates to the sum of configured weights under the parent. The following configurations divide the cluster resources between the "engineering", "support", and "marketing" organizations in 6:1:3 (6/10, 1/10, and 3/10) fraction of total resources.

To specify the capacity property based on the above example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the root and select Edit Child Queues option.
3. Enter the Configured Weight of "engineering" to 6, "support" to 1, and "marketing" to 3.
4. Click Save.

Related Information

Resource allocation overview

Configuring the resource capacity of root queue in absolute mode

In absolute resource allocation mode when you remove or add new resources to your cluster, you can update the resource capacity for the root queue accordingly. You can configure the capacity of the root queue using the YARN Queue Manager UI.

About this task

In absolute resource allocation mode when the available resource capacity changes permanently in the cluster or in a partition, reconfiguring the resource capacity of the root queue might be needed.

In case of a resource capacity increase, you can update the root queue directly.

However, if there was a decrease in the available resource capacity, you cannot update the root queue until child queues have been updated. In a case like this, you must update resource capacities starting from the lowest queue level. This is because a parent queue's capacity cannot be decreased until the total resource capacity of all of its child queue is reduced.

Procedure

1. In Cloudera Manager navigate to Clusters YARN Queue Manager UI .
   The **Overview** tab is opened, displaying a hierarchical view of the queues.
2. If there was a decrease in the available resources, configure the resource capacity of the child queues.

   You must update resource capacities starting from the lowest queue level. This is because a parent queue's capacity cannot be decreased until the total resource capacity of all of its child queue is reduced. You might have to change both the minimum and maximum values.
3. Click the three vertical dots next to the root queue.
4. Select Edit Root Queue.
   The **Accessible Cluster Capacity** dialog box is displayed.

5. Configure resource capacity for the root queue.

   The **Accessible Cluster Capacity** dialog box provides information about the available resources:

   - Total Configured Memory and the delta to be adjusted
   - Total Configured vCores and the delta to be adjusted

   If there was an increase in the resource capacity, the **Accessible Cluster Capacity** dialog box looks similar to the following:

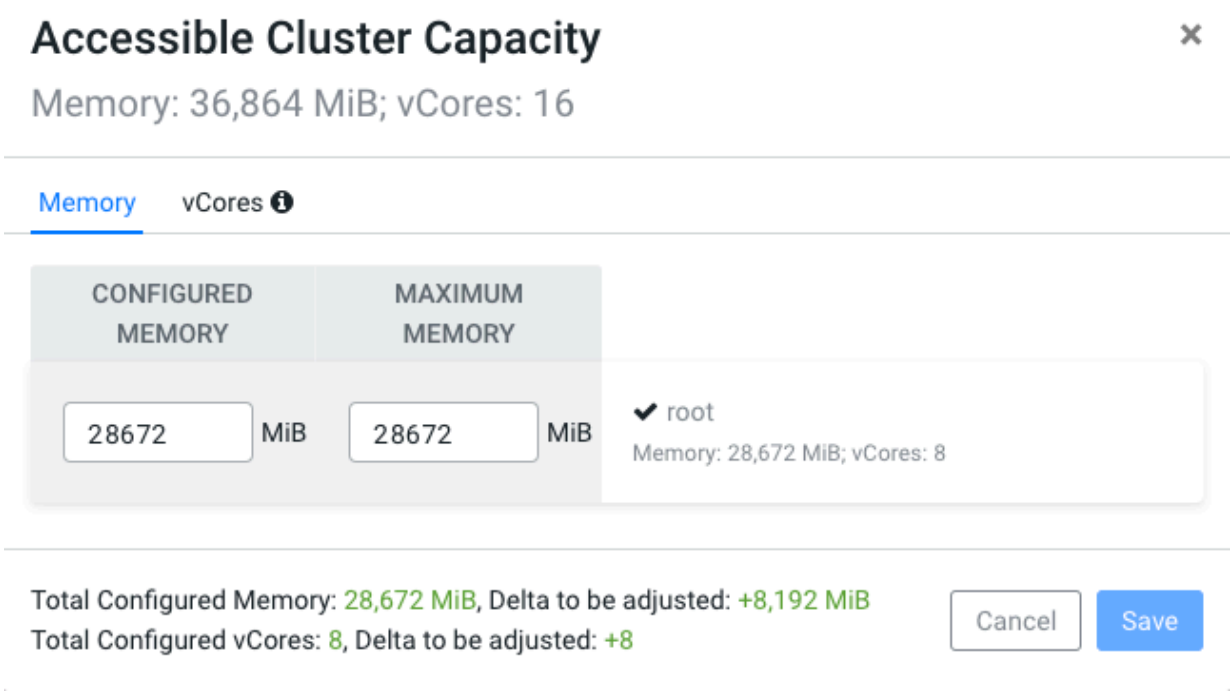

   If there was a decrease in the resource capacity, the **Accessible Cluster Capacity** dialog box looks similar to the following:

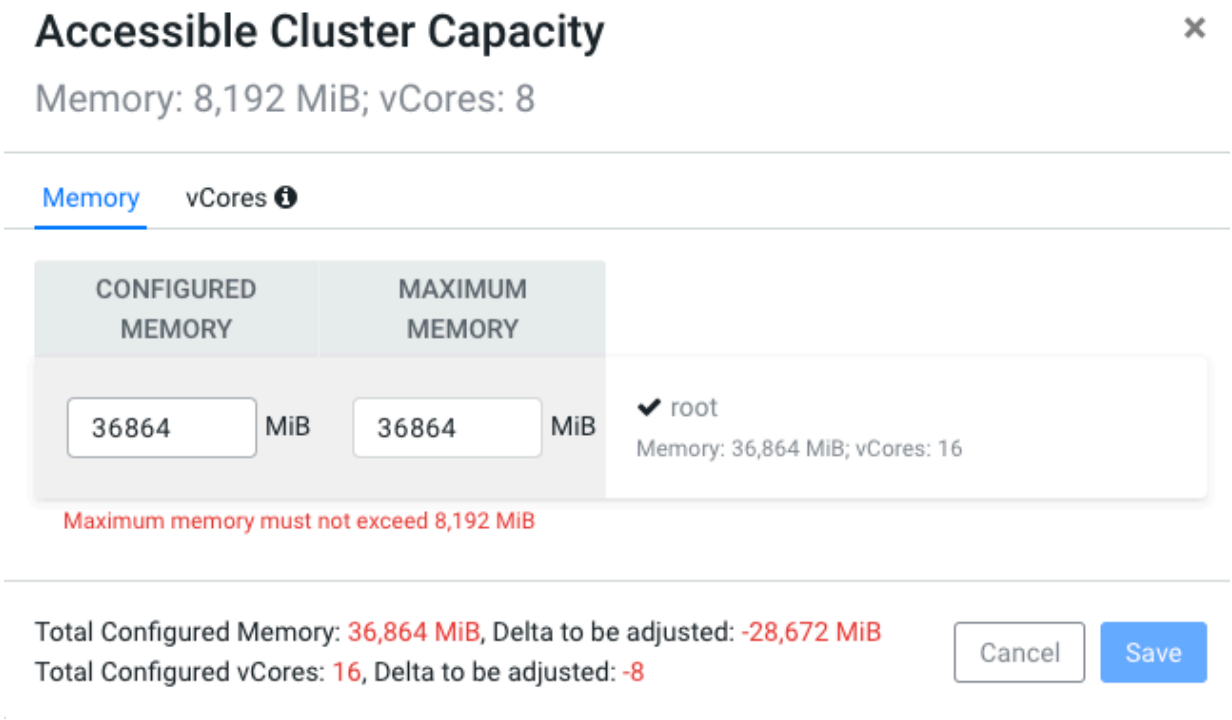

6. Click Save.

   If the adjustable delta is less than zero, you cannot save the updated configuration.

After saving the resource capacity of the root queue, the **Failed to update queue configuration** error message is displayed with the following details: Validation failed for modify queue operation. Error message: CapacityScheduler configuration validation failed:java.io.IOException: Failed to re-init queues : Parent Queues capacity: <memory:*[***CONFIGURED MEMORY OF THE ROOT QUEUE***]*, vCores:*[***CONFIGURED VCORES OF THE ROOT QUEUE***]*> is less than to its children:<memory:*[***CONFIGURED MEMORY OF THE CHILD QUEUES***]*, vCores:*[***CONFIGURED VCORES OF THE CHILD QUEUES***]*> for queue:root

This error message appears when there was a decrease in the available resources and you updated the resource capacity of the root queue before updating the child queues.

To resolve this issue, update the resource capacity of the child queues first, and then update the root queue.

Mixed resource allocation mode (Technical Preview)

In this mode, you can specify the resources in mixed types. You can specify the actual units of vcores and memory resources for each queue or specify the percentage of the total resources used by each queue or specify a weight for each queue. You can use a combination of these allocation modes. The queues under one parent can also mix their modes.

Technical Preview: This is a technical preview feature and is considered under development. Do not use this in your production systems. To share your feedback, contact Support by logging a case on our Cloudera Support Portal. Technical preview features are not guaranteed troubleshooting guidance and fixes.

Note: YARN Queue Manager does not support mixed resource allocation mode. This feature is only available through the safety valve.

Example of mixed mode resource allocation calculation

Capacity types are evaluated in the following order (precedence):

1. Absolute value

2. Percentage value
3. Weight value

An example of a calculation of queues *A*, *B*, and *C* with the following configured capacity vectors all resources: [memory=16284, vcore=16, custom=100]

- a: [memory=4096, vcore=50%, custom=3w]
- b: [memory=30%, vcore=10, custom=5w]
- c: [memory=70%, vcore=1w, custom=50]

The calculation of the effective resource is evaluated as the following:

1. Memory: *A* has an absolute resource, and *B* and *C* have a percentage, therefore the percentage is calculated from the remaining resource after allocating queue *A*. In this example, the memory of the a queue is 4096. In the remaining resources (16284 - 4096 = 12188), *B* queue gets 8532 (70% of 12188) and *C* queue gets 3656 (30% of 12188).
2. Vcore: *B* has an absolute resource, *A* has a percentage, then *C* has weight, which gets the remaining vcore. In this example, the vcore of the *B* queue is 10. In the remaining vcores (16 - 10 = 6), *C* queue gets 3 (1W of 6) and *A* queue gets 3 (50% of 6).
3. Custom: *C* has an absolute resource, and *B* and *C* have weight. In this example, the custom of the *C* queue is 50. In the remaining vcores (100 - 50 = 50), *B* queue gets 15 (5w of 50) and *A* queue gets 35 (3w of 50).

Related Information

Cloudera Support Portal

Setting capacity using mixed resource allocation mode (Technical Preview)

Set the resource allocation in the mixed allocation mode using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Go to the Configuration tab.
3. Set the following property in the Capacity Scheduler Configuration Advanced Configuration Snippet (Safety Valve) field to use the mixed allocation mode:

```
<property>
<name>yarn.scheduler.capacity.legacy-queue-mode.enabled</name>
<value>false</value>
</property>
```

4. Specify the capacity for each queue in the preferred allocation mode. Example:

```
<property>
<name>yarn.scheduler.capacity.root.queues</name>
<value>default,test_1,test_2</value>
</property>
<property>
<name>yarn.scheduler.capacity.root.capacity</name>
<value>100</value>
</property>
<property>
<name>yarn.scheduler.capacity.root.default.capacity</name>
<value>[memory=25%, vcores=1w]</value>
</property>
<property>
<name>yarn.scheduler.capacity.root.test_1.capacity</name>
<value>[memory=1w, vcores=10%]</value>
</property>
<property>
<name>yarn.scheduler.capacity.root.test_2.capacity</name>
```

```
<value>[memory=2048, vcores=2w]</value>
</property>
```

5. Click Save Changes.
6. Click Actions > Restart to restart the service.

Related Information

Resource allocation overview

Changing resource allocation mode

There are three supported resource allocation modes: absolute, relative, and weight resource allocation mode. When creating a new cluster, the default allocation mode is the relative resource allocation mode. You can change the resource allocation mode from the root queue by editing the queue properties using the Yarn Queue Manager UI.

About this task

The way you specify how resources are allocated is done differently in each of the resource allocation modes:

- Absolute resource allocation mode: In this mode, you specify the actual units of vcores and memory resources. It allows you to configure minimum resources independently for each queue.
- Relative resource allocation mode: In this mode, you specify the percentage of the total resources used by each queue.
- Weight resource allocation mode: In this mode, you specify a weight to each queue. For example, a queue with weight 2 should receive approximately twice as many resources as a queue with weight 1.
- Mixed resource allocation mode (Technical Preview): In this mode, you can specify the resources in mixed types by updating the XML configuration using Cloudera Manager. You can specify the actual units of vcores and memory resources for each queue or specify the percentage of the total resources used by each queue or specify a weight for each queue. You can mix the allocation modes. The queues under one parent can also mix their modes. For more information, see *Setting capacity using mixed resource allocation mode*.

Important: This is a technical preview feature and is considered under development. Do not use this in your production systems. If you have feedback, contact Support by logging a case on our Cloudera Support Portal. Technical preview features are not guaranteed troubleshooting and fixes.

Note: When migrating from a CDH cluster and performing the fs2cs scheduler conversion, the default resource allocation mode is weight mode. For more information see, *Migrating Fair Scheduler to Capacity Scheduler*.

Before you begin

- If you are changing to weight resource allocation mode, delete any managed parent queues.
- If you are changing from weight resource allocation mode, delete any dynamic parent queues.

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service.

   A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the root and select View Edit Queue Properties .

   The Queue Properties dialog box is displayed.
3. Select the resource allocation mode.
4. Click Save.
5. Enter yes in the Switch Allocation Mode dialog box and click OK.

   As a result, Queue Manager calculates and updates the resource allocations for all the existing queues. If required, you can further modify the resource allocation.
6. Click Save.

7. Restart YARN and YARN Queue Manager services.

Related Information

Resource allocation overview
Adding queues using YARN Queue Manager UI
Migrating Fair Scheduler to Capacity Scheduler
Migrating Fair Scheduler to Capacity Scheduler
Setting capacity using mixed resource allocation mode (Technical Preview)

Starting and stopping queues

Queues in YARN can be in two states: RUNNING or STOPPED. A RUNNING state indicates that a queue can accept application submissions, and a STOPPED queue does not accept application submissions. The default state of any configured queue is RUNNING.

In Capacity Scheduler, parent queues and leaf queues can be stopped. For an application to be accepted at any leaf queue, all the queues in the hierarchy all the way up to the root queue must be running. This means that if a parent queue is stopped, all of the descendant queues in that hierarchy are inactive, even if their own state is RUNNING.

To stop a queue:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select Stop Queue .
3. You will be prompted for a confirmation. Click OK to stop the queue.

To start a queue:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select Start Queue .
3. You will be prompted for a confirmation. Click OK to start the queue.

Administrators can use the ability to stop and drain applications in a queue for a number of reasons, such as when decommissioning a queue and migrating its users to other queues. Administrators can stop queues at run-time, so that while current applications run to completion, no new applications are accepted. Existing applications can continue until they finish running, and thus the queue can be drained gracefully without any end-user impact.

Deleting queues

You must first stop the queue before deleting the queue. You can delete a single queue and also parent queue and its children if all the queues in the hierarchy are stopped.

In Capacity Scheduler, parent queues, child queues, and the root queue can all be stopped. For an application to be accepted at any child queue, all the queues in the hierarchy all the way up to the root queue must be running. This means that if a parent queue is stopped, all of the descendant queues in that hierarchy are inactive, even if their own state is RUNNING.

Note: If the queue is associated with one or more partitions, you must first set the partition capacity to zero using Edit Child Queue for that queue for all the partitions before you delete the queue.

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select Delete Queue .

   You can use the Delete Queues and its Children option to delete both parent queue and its children queues.

3. You will be prompted for a confirmation. Click OK to stop the queue.

Setting queue priorities

By setting queue priorities you can ensure that applications can access cluster resources.

About this task

Although the preemption feature, which is enabled by default, improves cluster resources use, there are some use cases where applications might not have access to cluster resources without setting priorities:

Long-running applications:

Without setting queue priorities, long-running applications in queue that are under capacity and with lower relative resource use may not release cluster resources until they finish running.

Applications that require large containers:

The issue with long-running applications is exacerbated for applications that require large containers. With short-running applications, previous containers may eventually finish running and free the cluster resources for applications with large containers. But with long-running services in the cluster, the large containers may never get sufficiently large resources on any nodes.

Hive LLAP

Hive LLAP (Low-Latency Analytical Processing) enables you to run Hive queries with low-latency in near real-time. To ensure low-latency, you should set the priority of the queue used for LLAP to a higher priority, especially if your cluster includes long-running applications.

For more information, see *Configuring preemption* and *Enabling Intra-Queue preemption*

Before you begin

Configure preemption as described in *Configuring preemption*.

Procedure

1. In Cloudera Manager navigate to Cluster YARN Queue Manager UI .
   The Overview tab is opened, displaying a hierarchical view of the queues.
2. Find the queue for which you want to set queue priority and click the three vertical dots ⋮ next to its name.
3. Select Edit Queue Properties.
4. In the Resource Allocation section find the Queue Priority property.
5. Set the queue priority.
   By default all queues are set to a priority of 0. Higher numbers indicate higher priority.
6. Click Save.

Related Information

Configuring preemption

Enabling Intra-Queue preemption

Configuring scheduler properties at the global level

You can configure the scheduler properties to define the behaviour of all the queues. All parent and children queues inherit the properties set using the Scheduler Properties.

About this task

In Cloudera Manager, you can use the Configuration tab to configure the scheduler properties.

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service.

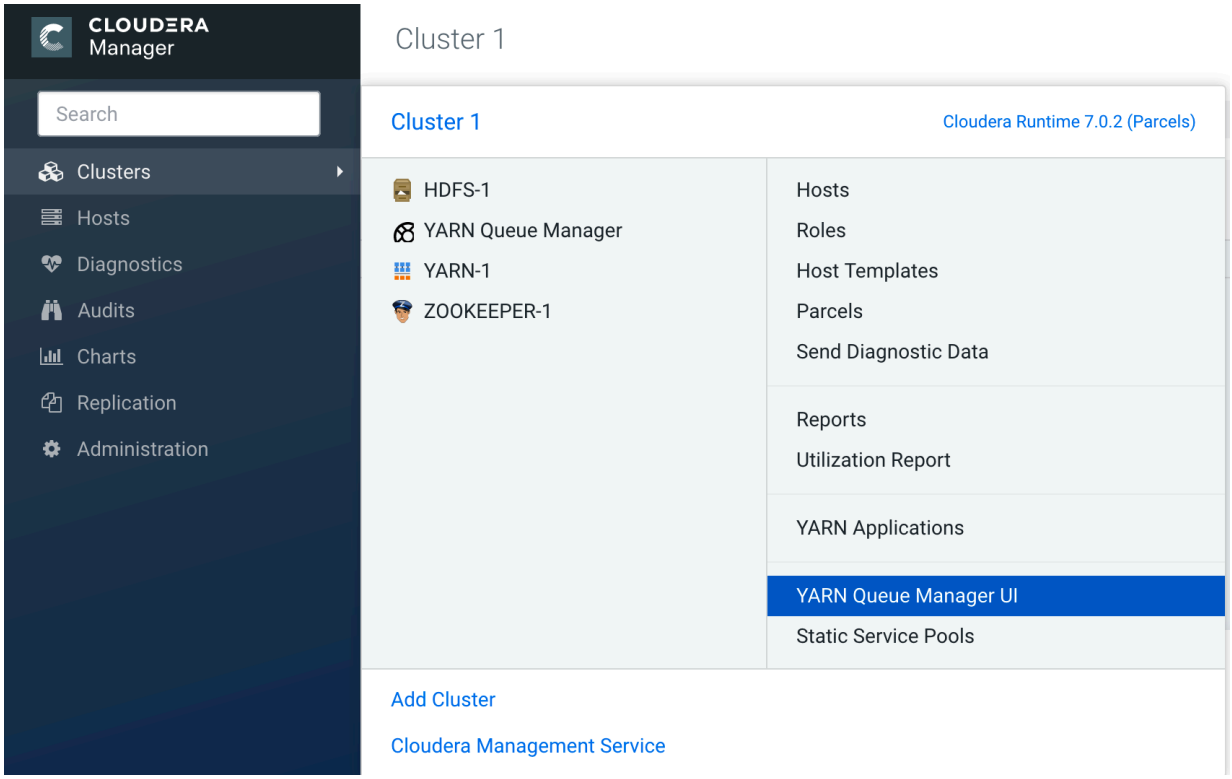

2. In the YARN Queue Manager window, click on the Configuration tab.

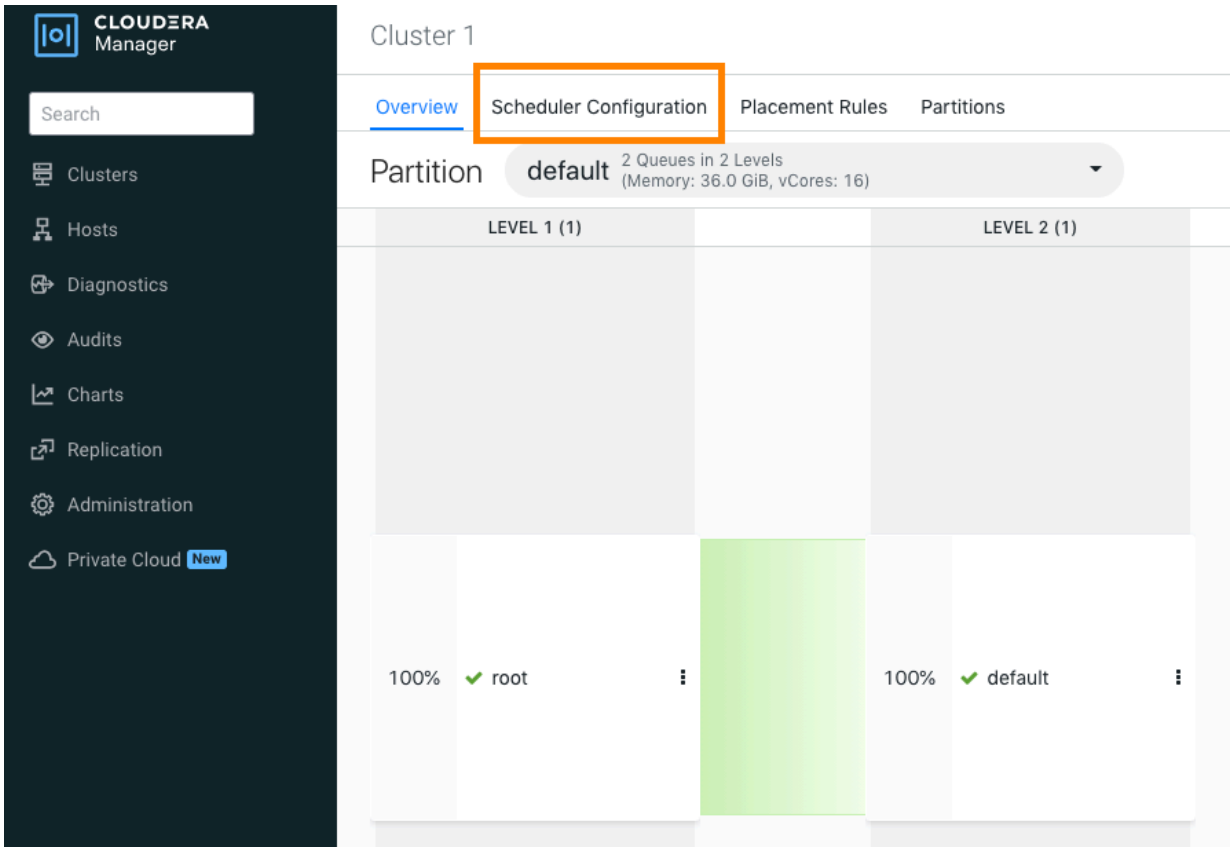

3. In the Scheduler Configuration window, enter the value of the property and click Save.

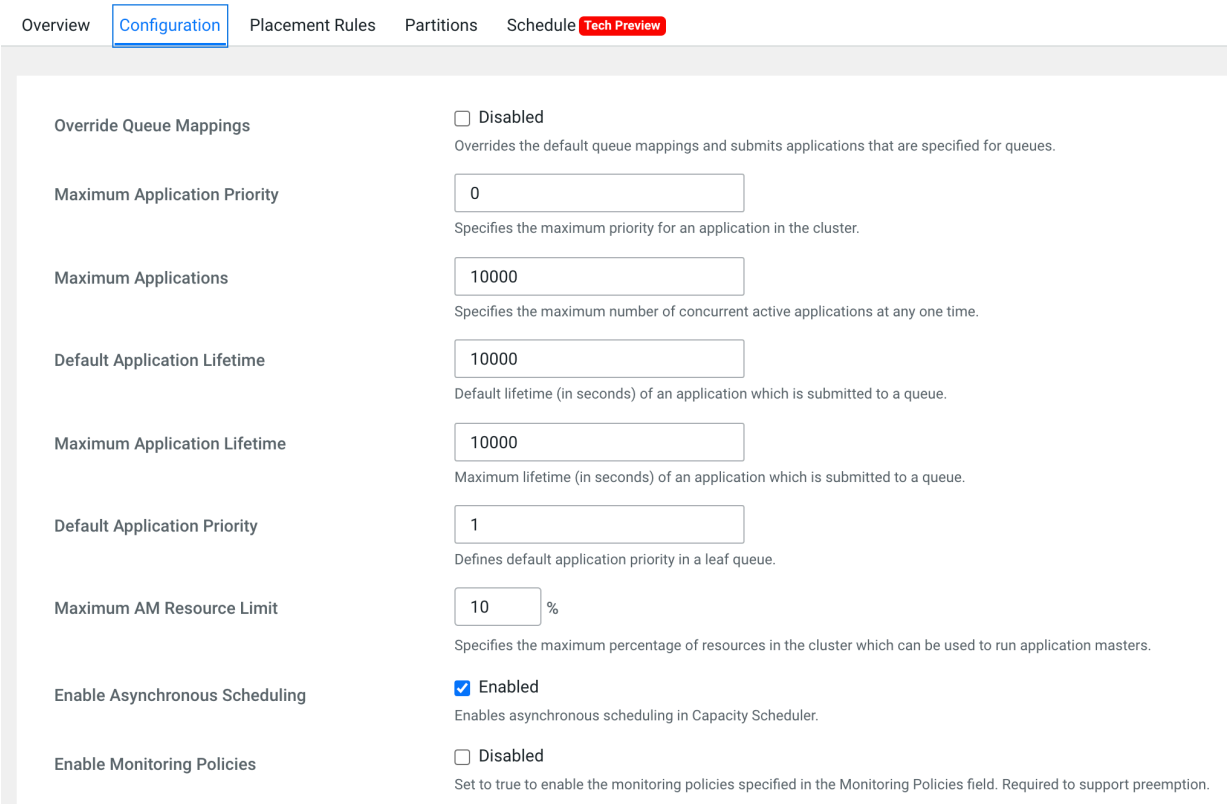

Setting global maximum application priority

You can use Priority Scheduling to run YARN applications at higher priority, regardless of other applications that are already running in the cluster. YARN allocates more resources to applications running at a higher priority over those running at a lower priority. Priority Scheduling enables you to set an application's priority both at the time of submission and dynamically at run time.

About this task

Priority Scheduling works only with the FIFO (First In, First Out) ordering policy. FIFO is the default Capacity Scheduler ordering policy.

To set appliation priority (yarn.cluster.max-application-priority), perform the following steps:

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click Configuration.
3. Search for yarn-site.xml.
4. Under YARN Service Advanced Configuration Snippet (Safety Valve) for yarn-site.xml, add the yarn.cluster.max-application-priority parameter and the value you need.
5. Click Save Changes.
6. Restart the YARN service.

Configuring preemption

Preemption allows applications of higher priority to preempt applications of lower priority.

About this task

A scenario can occur where a queue has a guaranteed level of cluster resources, but must wait to run applications because other queues are utilizing all of the available resources. If preemption is enabled, applications of higher priority do not have to wait because applications of lower priority have taken up the available capacity. When you enable Preemption (yarn.resourcemanager.scheduler.monitor.enable) underserved queues can begin to claim their allocated cluster resources almost immediately, without having to wait for other queues' applications to finish running.

You can use Priority Scheduling to run YARN applications at a higher priority, regardless of other applications that are already running in the cluster . For more information, see Setting global maximum application priority on page 41.

To configure the Queue Preemption options on all queues, perform the following:

Procedure

1. In Cloudera Manager, navigate to YARN Configuration .
2. Search for preemption.
3. Find the Capacity Scheduler Preemption property.
4. Ensure that it is checked, meaning that Preemption is enabled.
5. Under Configuration, search for ResourceManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml.

6. Configure the required preemption properties:
   - Preemption: Observe Only - Add the yarn.resourcemanager.monitor.capacity.preemption.observe_only parameter and the value you need.

     It runs the policy, but does not affect the cluster with preemption and stop events.
   - Preemption: Monitoring Interval (ms) - Add the yarn.resourcemanager.monitor.capacity.preemption.monitoring_interval parameter and the value you need.

     The time in milliseconds between invocations of this policy. Setting this value to a longer time interval causes the Capacity Monitor to run less frequently.
   - Preemption: Maximum Wait Before Kill (ms) - Add the yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill parameter and the value you need.

     The time in milliseconds between requesting a preemption from an application and stopping the container. Setting this to a higher value gives applications more time to respond to preemption requests and gracefully release containers.
   - Preemption: Total Resources Per Round - Add the yarn.resourcemanager.monitor.capacity.preemption.total_preemption_per_round parameter and the value you need.

     The maximum percentage of resources preempted in a single round. You can use this value to restrict the pace at which containers are reclaimed from the cluster. After computing the total desired preemption, the policy scales it back to this limit.
   - Preemption: Over Capacity Tolerance - Add the yarn.resourcemanager.monitor.capacity.preemption.max_ignored_over_capacity parameter and the value you need.

     The maximum amount of resources above the target capacity ignored for preemption. The default value is 0.1, which means that the Resource Manager starts preemption for a queue only when it goes 10% above its guaranteed capacity. This avoids resource-rotation and aggressive preemption.
   - Preemption: Maximum Termination Factor - Add the yarn.resourcemanager.monitor.capacity.preemption.natural_termination_factor parameter and the value you need.

     The maximum percentage of each queue's preemption target capacity that is preempted per cycle. You can increase this value to speed up resource reclamation.
7. Click Save Changes.
8. Restart the YARN services.

   For information about configuring preemption for a specific queue, see Configure preemption for a specific queue.

Enabling Intra-Queue preemption

Intra-queue preemption prevents resource imbalances in a queue.

Before you begin

Enable preemption by setting the yarn.resourcemanager.scheduler.monitor.enable property to true.

About this task

Intra-queue preemption helps in effective distribution of resources within a queue based on configured user limits or application priorities.

To configure Intra-Queue Preemption (yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled) on all queues, perform the following steps:

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click Configuration.
3. Search for yarn-site.xml.
4. Under YARN Service Advanced Configuration Snippet (Safety Valve) for yarn-site.xml, add the yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled parameter and enter the value true.
5. Click Save Changes.
6. Restart the YARN service.

   For information about configuring intra-queue preemption for a specific queue, see Configure Intra-Queue Preemption for a specific queue.

Enabling LazyPreemption

By enabling the LazyPreemption feature you can optimize the selection of containers from queues for preemption. Containers are preempted only if the resources can be used by another application.

About this task

Normally when preemption is performed there is no guarantee that preempted resources can be used by pending resource requests from under-satisfied queues. That can lead to excessive preemption as the preemption monitor keeps preempting resources from queue-A, but queue-B cannot use the preempted resources. As a result, such sources are allocated to queue-A again, but then they are preempted again.

The LazyPreemption feature makes resource preemption more efficient. It prevents excessive preemption and ensures that container selection is continuously optimized. It is diabled by default.

Enabling the LazyPreemption feature has the following advantages:

- Containers are killed only if the demanding application can use that capacity on the specific nodes.
- Application requests that cannot be satisfied at a moment for some reason will not lead to any container kill.
- If the application request gets canceled, no containers are killed.

LazyPreemption avoids scenarios when preempted resources cannot be used by the application that needs the resource, and then such preempted resources go back to the over-utilized queue again.

Procedure

1. In Cloudera Manager, navigate to YARN Configuration .
2. Search for safety valves.
3. Find the Capacity Scheduler Configuration Advanced Configuration Snippet (Safety Valve) property.
4. Set the yarn.scheduler.capacity.lazy-preemption-enabled property to true by adding the following code snipper:

```
<property>
   <name>yarn.scheduler.capacity.lazy-preemption-enabled</name>
   <value>true</value>
 </property>
 <property>
```

5. Click Save Changes.

Results

LazyPreemption is enabled.

Setting global application limits

To avoid system-thrash due to an unmanageable load -- caused either by malicious users, or by accident -- the Capacity Scheduler enables you to place a static, configurable limit on the total number of concurrently active (both running and pending) applications at any one time.

You can set the maximum applications limit (yarn.scheduler.capacity.maximum-applications) using this configuration property. The default value is 10,000.

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the Scheduler Configuration tab.
3. Enter the maximum application limit in the Maximum Applications text box.
4. Click Save.

For information about overriding this limit on a per-queue basis, see Set Application limit for a specific queue.

Related Information

Setting maximum parallel application limits

Setting maximum parallel application limits for a specific queue

Setting default Application Master resource limit

The Application Master (AM) resource limit that can be used to set a maximum percentage of cluster resources allocated specifically to Application Masters. This property has a default value of 10%, and exists to avoid cross-application deadlocks where significant resources in the cluster are occupied entirely by the Containers running ApplicationMasters.

About this task

The Maximum AM Resource Limit (yarn.scheduler.capacity.maximum-am-resource-percent) property also indirectly controls the number of concurrent running applications in the cluster, with each queue limited to a number of running applications proportional to its capacity.

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Configuration tab.
3. Enter the maximum AM resource limit in the Maximum AM Resource Limit text box.
4. Click Save.

   For information about overriding this limit on a per-queue basis, see Configure Application Master resource limit for a specific queue.

Related Information

Setting maximum parallel application limits

Setting maximum parallel application limits for a specific queue

Enabling asynchronous scheduler

Asynchronous scheduler decouples the CapacityScheduler scheduling from Node Heartbeats. This improves the latency significantly.

About this task

To enables asynchronous scheduling (yarn.scheduler.capacity.schedule-asynchronously.enable), perform the following:

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Configuration tab.
3. Select the Enable Asynchronous Scheduler check-box.
4. Click Save.

Configuring queue mapping to use the user name from the application tag using Cloudera Manager

You learn how to add service users to the YARN queue by following a mapping procedure.

You can configure queue mapping to use the user name from the application tag instead of the proxy user who submitted the job. You can add only service users like hive using the `yarn.resourcemanager.application-tag-based-placement.username.whitelist` property and not normal users.

When a user runs Hive queries, HiveServer2 submits the query in the queue mapped from an end user instead of a hive user. For example, when user *ALICE* submits a Hive query with doAs=false mode, job will run in YARN as hive user. If application-tag based scheduling is enabled, then the job will be placed to a target queue based on the queue mapping configuration of user *ALICE*.

For more information about queue mapping configurations, see Manage placement rulesManage placement rules. For information about Hive access, see Apache Hive documentation.

How to configure queue mapping

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for ResourceManager. In the Filters pane, under Scope, select ResourceManager.
4. In ResourceManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml add the following:
   a. Enable the application-tag-based-placement property to enable application placement based on the user ID passed using the application tags.

   ```
   Name: yarn.resourcemanager.application-tag-based-placement.enable
   Value: true
   Description: Set to "true" to enable application placement based on the
    user ID passed using the application tags. When it is enabled, it chec
   ks for the userid=<userId> pattern and if found, the application will be
    placed onto the found user's queue, if the original user has the requir
   ed rights on the passed user's queue.
   ```

   b. Add the list of users in the allowlist who can use application tag based placement. The applications when the submitting user is included in the allowlist, will be placed onto the queue defined in the yarn.scheduler.c apacity.queue-mappings property defined for the user from the application tag. If there is no user defined, the submitting user will be used.

   ```
   Name: yarn.resourcemanager.application-tag-based-placement.username.whit
   elist
   Value: <Hive process user(s)>
   ```

```
Description: Comma separated list of users who can use the application t
ag based placement, if "yarn.resourcemanager.application-tag-based-place
ment.enable" is enabled.
```

Note: Check the Hive system user value(s) and add the value(s) to the allowlist:

1. In Cloudera Manager, navigate to Hive Configuration .
2. Search for System User.
3. Note down the value(s) set as process username(s), and add the value(s) to the allowlist.
4. When there are no rules mapping for the tag user, it will fallback to the proxy user's (Hive) placement rule.

5. Restart the ResourceManager service for the changes to apply.

Related Information

Managing placement rules

Configuring NodeManager heartbeat

You can control how many containers can be allocated in each NodeManager heartbeat.

To configure NodeManager heartbeat for all queues, perform the following:

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Scheduler Configuration tab.
3. Select the Enable Multiple Assignments Per Heartbeat checkbox to allow multiple container assignments in one NodeManager heartbeat.
4. Configure the following NodeManager heartbeat properties:
   - Maximum Container Assignments Per Heartbeat - The maximum number of containers that can be assigned in one NodeManager heartbeat. Setting this value to -1 disables this limit.
   - Maximum Off-Switch Assignments Per Heartbeat - The maximum number of off-switch containers that can be assigned in one NodeManager heartbeat.
5. Click Save.

Configuring data locality

Capacity Scheduler leverages Delay Scheduling to honor task locality constraints. There are three levels of locality constraint: node-local, rack-local, and off-switch. The scheduler counts the number of missed opportunities when the locality cannot be satisfied and waits for this count to reach a threshold before relaxing the locality constraint to the next level. You can configure this threshold using the Node Locality Delay (yarn.scheduler.capacity.node-locality-delay) and Rack Locality Additional Delay (yarn.scheduler.capacity.rack-locality-additional-delay) fields.

To configure data locality, perform the following:

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Scheduler Configuration tab.
3. In the Node Locality Delay textbox, enter the number of scheduling opportunities that can be missed.

   Capacity Scheduler attempts to schedule rack-local containers only after missing this number of opportunities. You must ensure this number is same as the number of nodes in the cluster.
4. In the Rack Locality Additional Delay textbox, enter the number of missed scheduling opportunities, over the Node Locality Delay ones, after which Capacity Scheduler should attempt to schedule off-switch containers.

   A value of -1 means that the value is calculated based on the formula L * C / N, where L is the number of locations (nodes or racks) specified in the resource request, C is the number of requested containers, and N is the size of the cluster.

5. Click Save.

Setting Maximum Parallel Application

You can set the maximum number of applications that can run at the same time in a cluster. Ensure you understand how the parallel application limits are inherited from the parent queue in the queue hierarchy before using this option.

You can set parallel application limits for all queues, all users, and at the user level. When the maximum parallel applications limit is reached, application submissions are not rejected and instead they stay in the *ACCEPTED* state until they are eligible to run.

The evaluation of the limits happens in the following order:

1. Maximum Applications - if the limit is exceeded, the submission is rejected immediately.
2. Maximum Parallel Applications - if the limit is exceeded, the submission is accepted, but the application does not transition to the RUNNING state. It stays in the ACCEPTED state until the queue or user limits are satisfied.
3. Maximum AM Resource Limit - if there are too many Application Masters running, the application stays in the ACCEPTED state until the resources become available.

Example - Application Limits

The maximum parallel application limit is an inherited property in the queue hierarchy. You can lower the limit down the hierarchy, but setting a higher value than its higher queue in the queue structure, the limit will not be effective. The following example provides information about how the parallel application limit is applied in a queue hierarchy. The Maximum Parallel Applications: All Queues value is set to 250 at the scheduler level and 500 is set at the root level.

Example queue hierarchy:

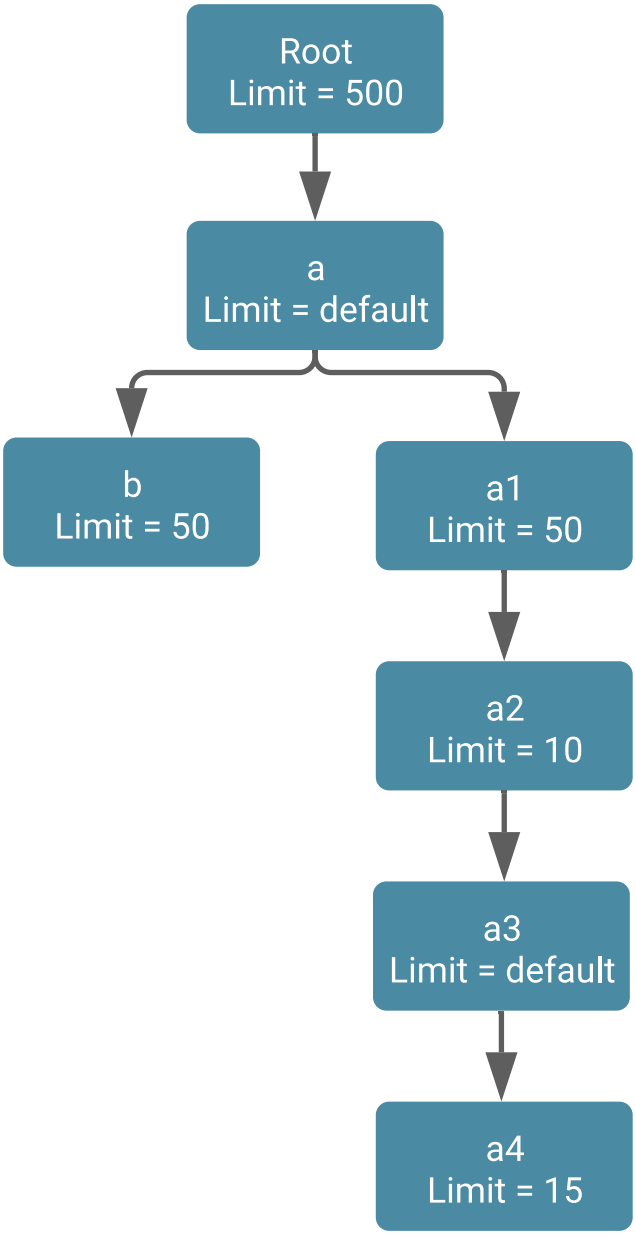

- Maximum 250 applications can run parallel under the root.a queues.
- Maximum 50 applications can run parallel under the root.a.a1 queues.
- Maximum 10 applications can run parallel under the root.a.a1.a2 queues.
- Maximum 10 applications can run parallel under the root.a1.a2.a3 queues. Even though the Maximum Parallel Applications limit is not set for .a3, the default value of 250 applies for that queue; however, the parent queue (root.a.a1.a2) has a lower value of 10, and the child .a3 queue cannot exceed that limit.
- Maximum 10 applications can run parallel under the root.a1.a2.a3.a4 queue. Even though the queue level limit is configured for 15 applications, the parents restrict this limit to 10.
- Maximum 50 applications can run parallel under the root.a.b queue.

Example - User Limits

- Scenario 1 - In this scenario, there are two users, *USER1* and *USER2*. When you set the Maximum Parallel Applications: All Queues to 100 and Maximum Parallel Applications: All Users to 50, both *USER1* and *USER2* can have 50 applications running in parallel.

- Scenario 2 - In this scenario, there are four users, *USER1*, *USER2*, *USER3*, and *USER4*. You can set the limits as follows:
  - Maximum Parallel Applications: All Queues - 100
  - Maximum Parallel Applications: All Users - 15
  - Maximum Parallel Applications: Per Users
    - User1 - 10
    - User 2 - 25

  USER1 can have 10 applications running in parallel. *USER2* can have 25 applications running in parallel. *USER3* and *USER4* can have 15 applications running in parallel.
- Scenario 3: In this scenario, the Maximum Parallel Applications limit for root.a is set to 5. The Maximum Parallel Applications: Per Users for *USER1* is set to 10. Even though *USER1* can run 10 parallel applications, in the root.a queue, *USER1* can run only 5 maximum applications. *USER1* can also run 5 more applications on some other queue.
- Scenario 4: In this scenario, the Maximum Parallel Applications limit for root.b is set to 20. The Maximum Parallel Applications: Per Users limit for *USER2* is set to 10. *USER2* can run 10 parallel applications, in the root.b queue. Some other users can run 10 more applications in the root.b queue.

Related Information

Setting maximum parallel application limits

Setting maximum parallel application limits

You can set the maximum number of applications limits for all queues, all users, and at the user level. The maximum parallel application limit is inherited from the “root” queue level and is lowered down in the queue hierarchy. The limit is checked in the queue hierarchy and the lowest value is applied as the limit.

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Configuration tab.
3. Specify the application limits:
   - Maximum Parallel Applications: All Queues - specifies the maximum number of applications that can run at the same time for any queue that does not override it with the per queue level setting.
   - Maximum Parallel Applications: All Users - specifies the maximum number of applications that can run at the same time for all the users. This limit is overridden based on the per user limit.
   - Maximum Parallel Applications: Per User - specifies parallel running application limit for the specified user. Click + to add more users and their application limits.
4. Click Save.

Related Information

Setting Maximum Parallel Application

Setting maximum parallel application limits for a specific queue

Setting default Application Master resource limit

Setting global application limits

Configuring per queue properties

Queue properties contain the settings that define the behavior of the queue. Using queue properties, you can define the setting that need not inherit properties directly from the parent queue and define the setting specific to the queue.

About this task

In Cloudera Manager, you can use the Queue Properties to view and configure the queue properties.

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service.

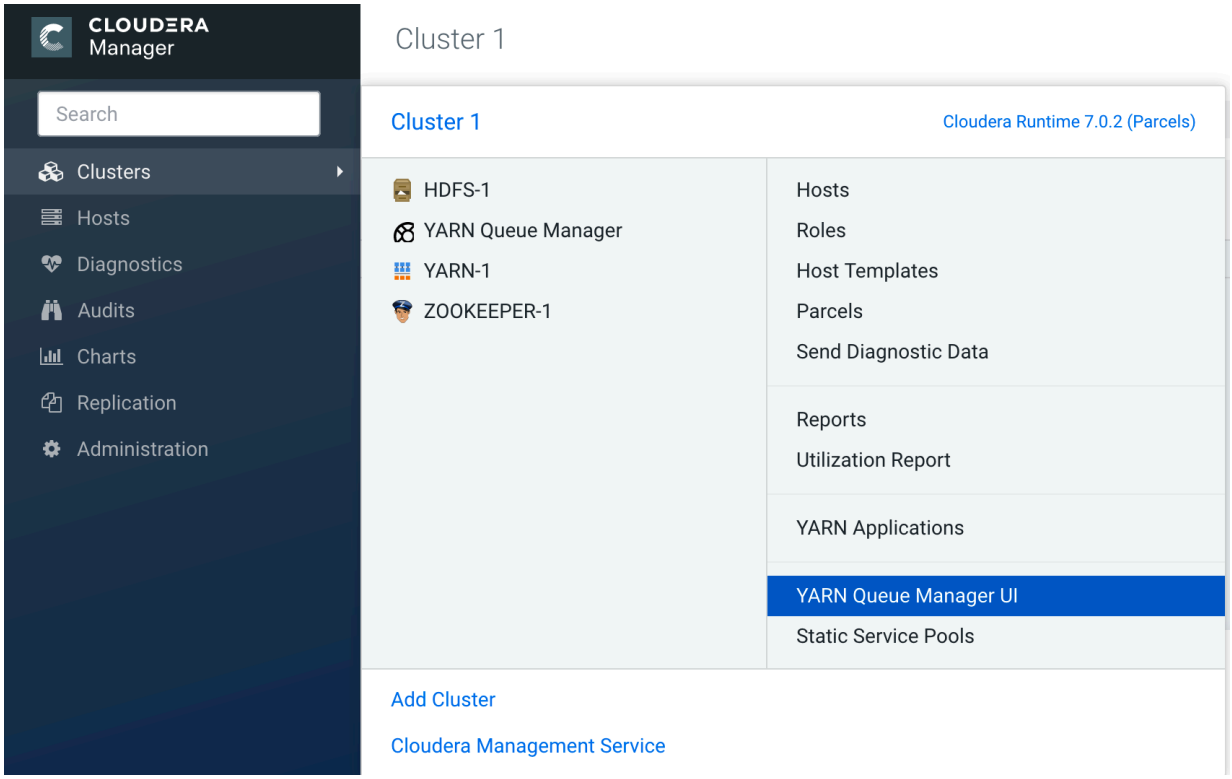

2. Click on the three vertical dots on the queue and select the Edit Queue Properties option.

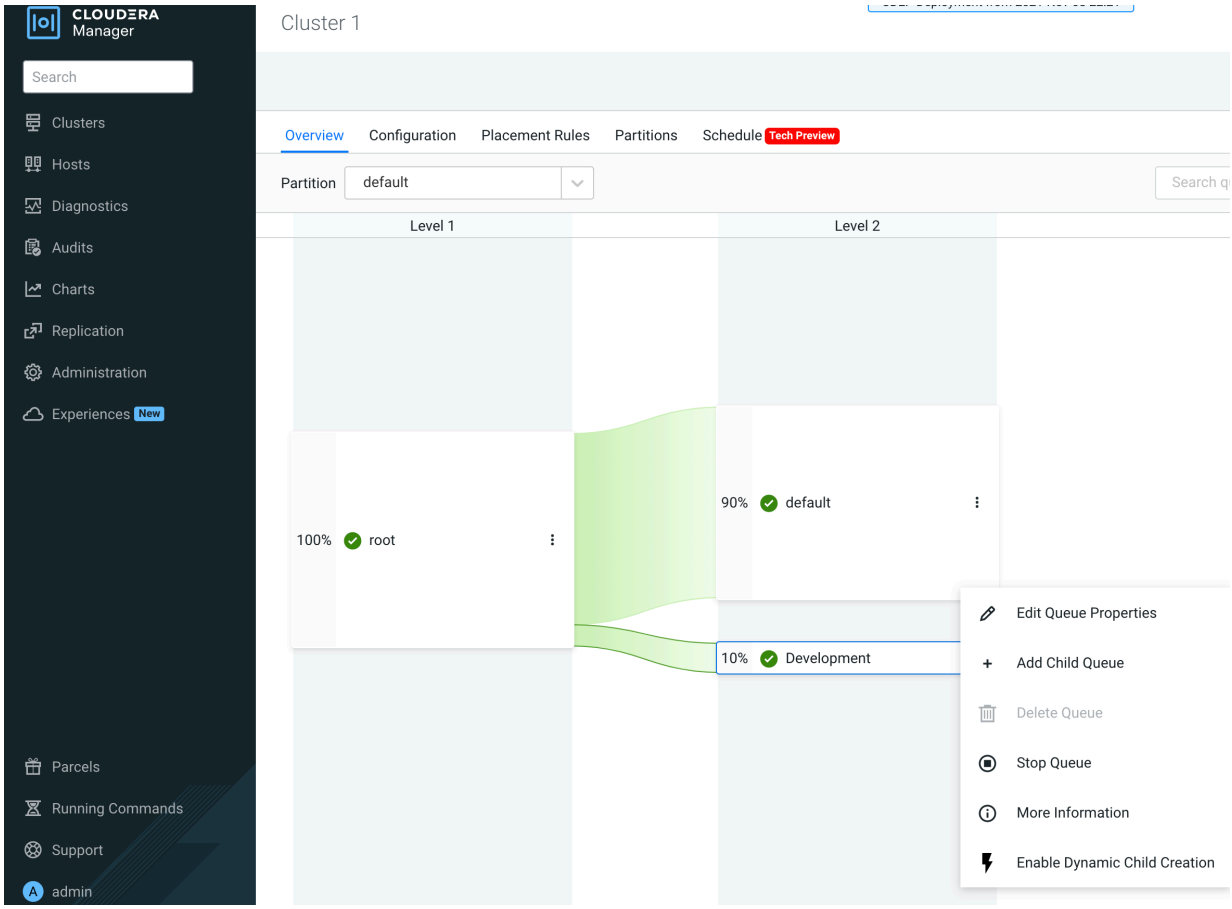

3. In the Queue Properties window, enter the value of the property and click Save.

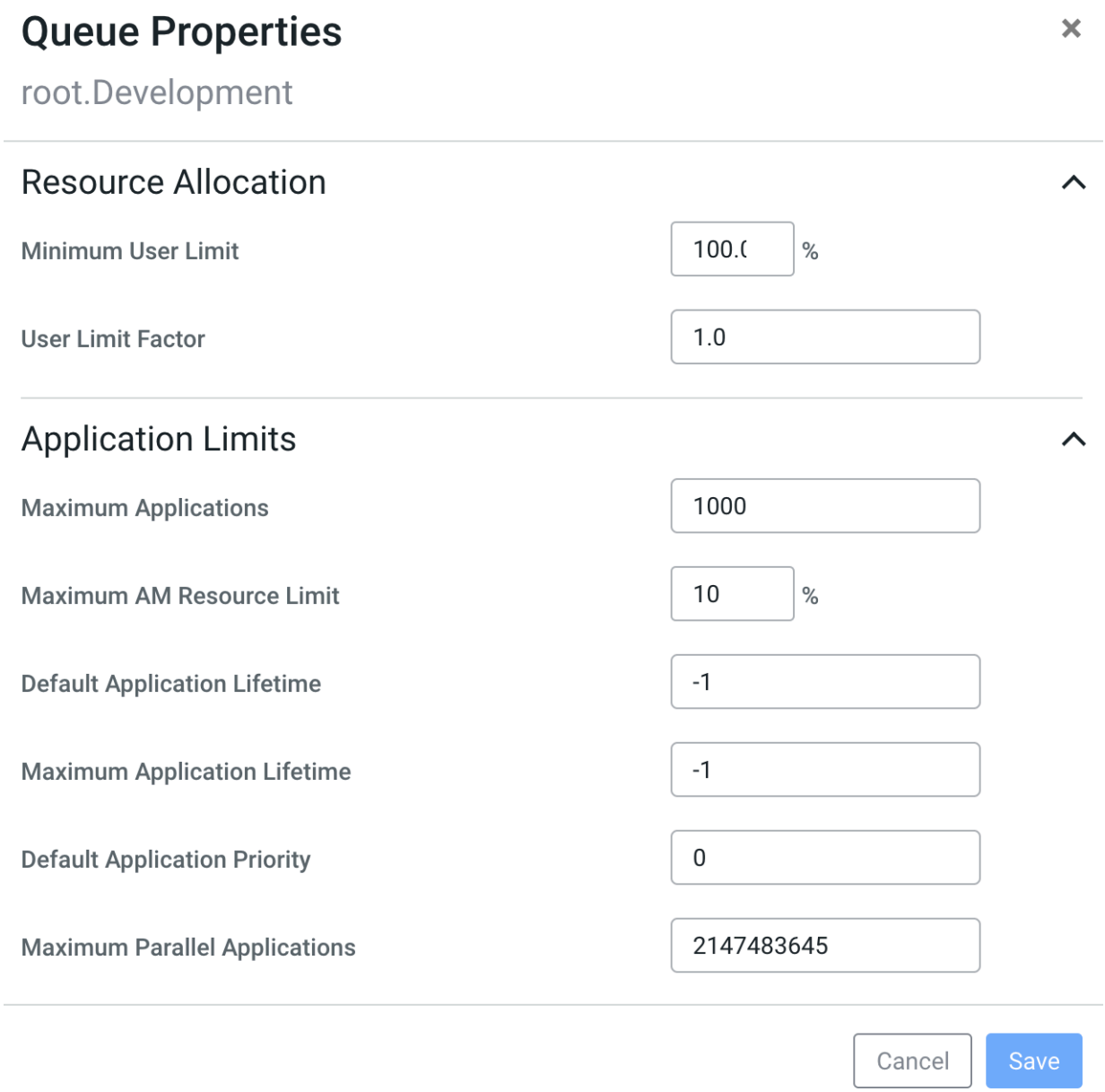

Setting user limits within a queue

Set a minimum percentage of resources allocated to each leaf queue user.

Minimum User Limit

The Minimum User Limit (minimum-user-limit-percent) property can be used to set the minimum percentage of resources allocated to each leaf queue user. The Minimum User Percentage is a soft limit on the smallest amount of resources a single user should get access to if they are requesting it. For example, if you set the Minimum User Percentage property to 10%, 10 users get 10% each, assuming they are all requesting it; this value is a soft limit because if one of the users asks for less capacity, you can place more users in the queue.

For example, to enable equal sharing of the *SERVICES* leaf queue capacity among five users, you must set the Minimum User Limit value to 20%.

To configure user limits based on this example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the Services queue and select Edit Queue Properties option.
3. In the Queue Properties dialog-box, enter 20 in the Minimum User Limit text box.
4. Click Save.

User Limit Factor

You can use the User Limit Factor (user-limit-factor) to control the maximum amount of resources that a single user can consume. User Limit Factor is set as a multiple of the queues minimum capacity where a user limit factor of one means the user can consume the entire minimum capacity of the queue. If the user limit factor is greater than 1, it is possible for the user consumption to grow into maximum capacity and if the value is set to less than 1, such as 0.5, a user can only obtain half of the minimum capacity of the queue. If you want a user to obtain the maximum capacity of a queue, set a value greater than 1 to allow the minimum capacity to be overtaken by a user that many times.

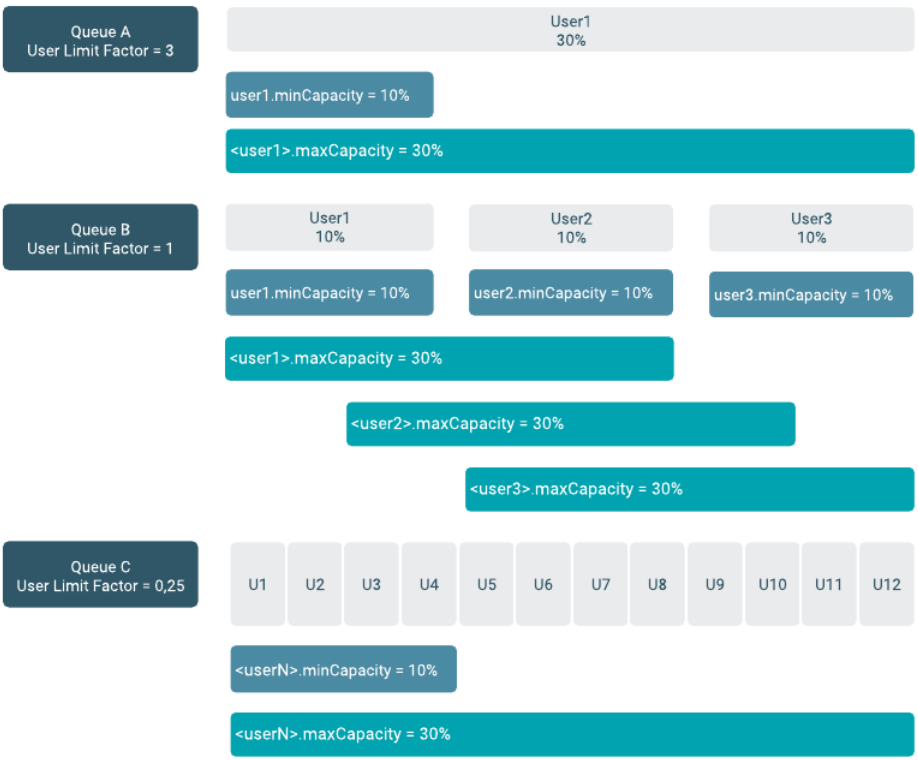

The following table shows how the queue resources are adjusted as users submit jobs to a queue with a minimum user limit percentage is set to 20%:

Table 1: Minimum User Limit

Minimum user limit percent = 20	
1st user submits job	Sole user gets 100% of the queue capacity
2nd user submits job	Each user equally shares 50% of the queue capacity
3rd user submits job	Each user equally shares 30.33% of the queue capacity
4th user submits job	Each user equally shares 25% of the queue capacity
5th user submits job	Each user equally shares 20% of the queue capacity
6th user submits job	6th user must wait for queue capacity to free up

- Queue resources are adjusted in the same manner for a single user submitting multiple jobs in succession. If no other users are requesting queue resources, the first job receives 100% of the queue capacity. When the user submits a second job, each job receives 50% of queue capacity. When the user submits a third job, each job receives 33% of queue capacity. If a fourth user then submits a job, each job would receive 25% of queue capacity. When the number of jobs submitted by all users reaches a total of five, each job will receive 20% of queue capacity, and subsequent users must wait for queue capacity to free up (assuming preemption is not enabled).
- The Capacity Scheduler also manages resources for decreasing numbers of users. As users' applications finish running, other existing users with outstanding requirements begin to reclaim that share.
- Note that despite this sharing among users, the FIFO application scheduling order of Capacity Scheduler does not change. This guarantees that users cannot monopolize queues by submitting new applications continuously. Applications (and thus the corresponding users) that are submitted first always get a higher priority than applications that are submitted later.

Capacity Scheduler's leaf queues can also use the user-limit-factor property to control user resource allocations. This property denotes the fraction of queue capacity that any single user can consume up to a maximum value, regardless of whether or not there are idle resources in the cluster.

To configure the maximum limit (user-limit-factor) based on this example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue you want to set limit and select Edit Queue Properties option.
3. In the Queue Properties dialog-box, enter 1 in the User Limit Factor text box.
4. Click Save.

The default value of 1 means that any single user in the queue can, at the most, only occupy only the queue's configured capacity. This prevents users in a single queue from monopolizing resources across all queues in a cluster. Setting the value to 2 restricts the queue's users to twice the queue's configured capacity. Setting it to a value of 0.5 restricts any user from using resources beyond half of the queue capacity.

Setting Maximum Application limit for a specific queue

You can set the maximum number of applications that can run in a specific queue. This limit overrides it's parent queue limit if this specific queue level value is lesser than what is set for the parent queue.

About this task

You can set the maximum-application-limit property using the Maximum Application queue property. The limit for running applications in any specific queue is a fraction of this total limit, proportional to its capacity. This is a hard limit, which means that once this limit is reached for a queue, any new applications to that queue will be rejected, and clients will have to wait and retry later.

To set the application limit (yarn.scheduler.capacity.<queue-path>.maximum-applications) on a specific queue, perform the following:

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, enter the maximum application limit in the Maximum Application text box.
4. Click Save.

   For information about setting application limits on all the queues, see Set Application Limits.

Setting Application-Master resource-limit for a specific queue

The Application Master (AM) resource limit can be used to set a maximum percentage of cluster resources allocated specifically to Application Masters. The default value is 10% and exists to avoid cross-application deadlocks where significant resources in the cluster are occupied entirely by the Containers running Application Masters.

About this task

This property also indirectly controls the number of concurrent running applications in the cluster, with each queue limited to a number of running applications proportional to its capacity.

To set the maximum Application Masters resource limit (yarn.scheduler.capacity.maximum-am-resource-percent) for a specific queue:

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, enter the limit in the Maximum AM Resource Limit text box.
4. Click Save.

   For information about setting Application Master resource limits on all the queues, see Set Application Master Resource Limit.

Setting maximum parallel application limits for a specific queue

You can set the maximum number of applications that can run in a specific queue. This limit overrides it's parent queue limit if this specific queue level value is lesser than what is set for the parent queue.

Before you begin

Learn how the parallel application limits are inherited from the parent queue in the queue hierarchy and its evaluation. For more information, see Maximum Parallel Application settings .

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select Edit Queue Properties option.
3. In the Queue Properties dialog-box, enter the maximum parallel application limit in the Maximum Parallel Applications text box.
4. Click Save.

Related Information

Setting maximum parallel application limits

Setting default Application Master resource limit

Setting global application limits

Controlling access to queues using ACLs

Use Access-control lists (ACLs) to control access rights to users and administrators to the Capacity Scheduler queues.

Application submission can really only happen at the leaf queue level, but an ACL restriction set on a parent queue will be applied to all of its descendant queues.

In the Capacity Scheduler, ACLs are configured by granting queue access to a list of users and groups with the Submit Application ACL parameter. The format of the list is "user1,user2 group1,group2" -- a comma-separated list of users, followed by a space, followed by a comma-separated list of groups.

Note: The default value of Submit Application ACL for a root queue is yarn, which means that only the default yarn user can submit applications to that queue. Therefore, to provide specific users and groups with access to the queue, you must explicitly set the value of Submit Application ACL to those users and groups.

The Submit Application ACL (acl_submit_applications) parameter can also be set to the following values:

asterisk ("*")

Allow access to all users and group

space character (" ")

Block access to all users and groups

empty string value ("")

Block access to all users and groups

As mentioned previously, ACL settings on a parent queue are applied to all of its descendant queues. Therefore, if the parent queue uses the "*" (asterisk) value (or is not specified) to allow access to all users and groups, its child queues cannot restrict access. Similarly, before you can restrict access to a child queue, you must first set the parent queue to " " (space character) or "" (empty string value) to block access to all users and groups.

For example, the following properties would set the root Submit Application ACL value to " " (space character) to block access to all users and groups, and also restrict access to its child "support" queue to the users "sherlock" and "john" and the members of the "cfo-group" group:

Each child queue is tied to its parent queue with the configuration property. The top-level "support", "engineering", and "marketing" queues would be tied to the "root" queue.

To set the ACLs based on this example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue you want to set ACL and select Edit Queue Properties option.
3. In the Queue Properties dialog-box, add sherlock,john cfo-group in the Submit Application ACL text box.
4. Click Save.

A separate ACL can be used to control the administration of queues at various levels. Queue administrators can submit applications to the queue, stop applications in the queue, and obtain information about any application in the queue (whereas normal users are restricted from viewing all of the details of other users' applications).

If the Queue Administer ACL value is set to " " (space character) or "" (empty string value), it blocks access to all users and groups. If the ACL is set to sherlock,john cfo-group , it allows access to the users "sherlock" and "john" and the members of the "cfo-group" group.

Enabling preemption for a specific queue

Capacity Scheduler Preemption allows higher-priority applications to preempt lower-priority applications.

About this task

A scenario can occur in which a queue has a guaranteed level of cluster resources, but must wait to run applications because other queues are utilizing all of the available resources. If Preemption is enabled, higher-priority applications do not have to wait because lower priority applications have taken up the available capacity. With Preemption enabled, under-served queues can begin to claim their allocated cluster resources almost immediately, without having to wait for other queues' applications to finish running.

Note: If the preemption policy is disabled in the scheduler configurations, you cannot enable preemption for a specific queue. For information about setting scheduler level preemption, see Configuring preemption on page 41.

You can disable the queue preemption (yarn.resourcemanager.scheduler.monitor.enable) for a specific queue.

Before you begin

To enable or disable preemption at the queue level in the Queue Manager UI, you must enable monitoring policies in the global configuration level and define the monitoring policies' class.

- In the YARN Queue Manager WebUI Configuration , check Enable Monitoring Policies and set Monitoring Policies to org.apache.hadoop.yarn.server.resourcemanager.monitor.capacity.ProportionalCapacityPreemptionPolicy and save to enable/disable preemption for specific queues.

With Enable Monitoring Policies disabled the queue-level preemption option is greyed out and indicates the inherited configuration (enabled by default).

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, uncheck the Enable Preemption checkbox.
4. Click Save.

Enabling Intra-Queue Preemption for a specific queue

Intra-queue preemption prevents resource imbalances in a queue.

About this task

Intra-queue preemption helps in effective distribution of resources within a queue based on configured user limits or application priorities.

Note: If the intra-queue preemption policy is disabled in the scheduler configurations, you cannot enable intra-queue preemption for a specific queue. For information about setting scheduler level intra-queue preemption, see Configure Intra-Queue Preemption.

To disable Intra-Queue Preemption (yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled) for a specific queue.

Before you begin

You must enable inter-queue preemption to enable intra-queue preemption at the queue level.

Procedure

1. In Cloudera Manager select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, uncheck the Enable Intra Queue Preemption checkbox.
4. Click Save.

Setting ordering policies within a specific queue

Set FIFO (First In, First Out) or Fair scheduling policies in Capacity Scheduler depending on your requirements.

The default ordering policy in Capacity Scheduler is FIFO (First In, First Out). FIFO generally works well for predictable, recurring batch jobs, but sometimes not as well for on-demand or exploratory workloads. For these types of jobs, Fair Scheduling is often a better choice. Flexible scheduling policies enable you to assign FIFO or Fair ordering policies for different types of workloads on a per-queue basis.

Examples FIFO and Fair Sharing Policies

Both FIFO (First In, First Out) and Fair scheduling policies work differently in batch jobs and ad hoc jobs.

Batch Example

In the below example, two queues have the same resources available. One uses the FIFO ordering policy, and the other uses the Fair Sharing policy. A user submits three jobs to each queue one after another, waiting just long enough for each job to start. The first job uses 6x the resource limit in the queue, the second 4x, and last 2x.

- In the FIFO queue, the 6x job would start and run to completion, then the 4x job would start and run to completion, and then the 2x job. They would start and finish in the order 6x, 4x, 2x.
- In the Fair queue, the 6x job would start, then the 4x job, and then the 2x job. All three would run concurrently, with each using 1/3 of the available application resources. They would typically finish in the following order: 2x, 4x, 6x.

Ad Hoc Plus Batch Example

In this example, a job using 10x the queue resources is running. After the job is halfway complete, the same user starts a second job needing 1x the queue resources.

- In the FIFO queue, the 10x job will run until it no longer uses all queue resources (map phase complete, for example), and then the 1x job will start.
- In the Fair queue, the 1x job will start, run, and complete as soon as possible – picking up resources from the 10x job by attrition.

Best Practices for Ordering Policies

- Ordering policies are configured on a per-queue basis, with the default ordering policy set to FIFO. Fairness is usually best for on-demand, interactive, or exploratory workloads, while FIFO can be more efficient for predictable, recurring batch processing. You should segregate these different types of workloads into queues configured with the appropriate ordering policy.
- · In queues supporting both large and small applications, large applications can potentially "starve" (not receive sufficient resources). To avoid this scenario, use different queues for large and small jobs, or use size-based weighting to reduce the natural tendency of the ordering logic to favor smaller applications.
- · Use the Maximum AM Resource Limit scheduler property to restrict the number of concurrent applications running in the queue to avoid a scenario in which too many applications are running simultaneously. Limits on each queue are directly proportional to their queue capacities and user limits. This property is specified as a float, for example: 0.5 = 50%. The default setting is 0.1=10%. This property can be set for all queues by setting the Maximum AM Resource Limit property at the root level, and can also be overridden on a per-queue basis by setting the Maximum AM Resource LimitSetting default Application Master resource limit on page 44property at the queue level.

Configure queue ordering policies

You can configure the property for queue ordering policies (yarn.scheduler.capacity.<queue-path>.ordering-policy) to FIFO or Fair. The default setting is FIFO.

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue you want to configure queue ordering policies and select the Edit Queue Properties option.
3. In the Queue Properties dialog-box, select the ordering policy as FIFO or Fair using the Ordering Policy drop-down box .
4. Click Save.

Autoscaling clusters

Autoscaling is a feature that adjusts the capacity of cluster nodes running YARN by automatically increasing or decreasing, or suspending and resuming, the nodes in a host group. You can enable autoscaling based either on a schedule that you define, or the real-time demands of your workloads.

Load-based autoscaling

Attention:

Load-based autoscaling is available for clusters with the following Runtime versions:

- 7.2.15 and higher

Load-based auto-scaling is supported in AWS and Azure. In GCP, load-based autoscaling is available as technical preview and is under entitlement. To obtain the required entitlement, contact your Cloudera Account Representative.

Load-based autoscaling suspends and resumes (stops and starts) instances on the cloud provider to increase or decrease capacity for nodes running NodeManagers (for example, the compute host group), based upon YARN's assessment of pending demand and available capacity. Load-based autoscaling can help control costs while providing quick, on-demand cluster capacity when you need it (within a few minutes).

When you configure a load-based autoscaling policy, you choose a minimum and maximum number of nodes for the host group. The maximum number of nodes determines how many instances are provisioned, but these instances are suspended and resumed as the workload demand requires. The policy will not provision instances beyond the maximum range of nodes that you define, regardless of the demand on the cluster. You also define a cooldown period, which is the amount of time in minutes to wait before another autoscaling operation is performed.

Schedule-based autoscaling

Attention:

Schedule-based autoscaling is available for clusters with the following Runtime versions:

- 7.2.15 and higher

Schedule-based autoscaling scales the nodes in a host group up or down based upon a schedule that you define. Schedule-based autoscaling is useful if workload demands tend to be high or low on a fairly regular, consistent basis.

When you configure a schedule-based autoscaling policy, you define a target node count, which is the number of nodes that you want to scale up or down to at a particular time. You also select whether or not to repeat the schedule, and finalize the schedule by entering a CRON expression and selecting the desired timezone. When the particular time/date that you define in the CRON expression occurs, the cluster is upscaled or downscaled to your target node count. The time taken to add nodes to a cluster varies by cloud provider and the configuration of the nodes (for example, recipes can have an impact on the time it takes to add a node).

Note: Typically, schedules (especially those configured to scale-up) should be defined in a manner to factor in the time it takes to add new nodes. For example, workloads starting at 9pm will typically set a scale-up schedule for 8:45 pm, assuming it takes around 15 minutes to add a node.

General considerations

Currently, YARN NodeManager is the only service compatible with autoscaling. Because of this, autoscaling is only available by default in the Data Engineering and Data Engineering HA cluster definitions. Autoscaling can also be configured for clusters with custom templates that include YARN NodeManager and the necessary gateway components.

Known Limitations

- Load-based autoscaling does not work with YARN dynamic queues.

Autoscaling behavior

Before you define an autoscaling policy, note the following autoscaling behaviors.

General behaviors

- Clusters can perform one upscale or downscale operation at a time.
- A cluster will continue to accept jobs while it is running, regardless of any in-progress upscale or downscale operations.
- Only one autoscale policy type (either load-based or schedule-based) can be configured for a single host group, in a single cluster, at a time.
- Autoscaling is available for host groups with nodes running YARN NodeManager only (and optionally client/GATEWAY components, but not any other service components).
- If there are not enough nodes available to match the requested scale operation, the operation will proceed on however many nodes are available (for example, during a request for a 10 node scale-up, if the cluster loses 1 node, the operation will proceed with scaling-up 9 nodes instead of 10).
- Autoscaling will be disabled if the cluster has any node failures on instances running YARN ResourceManager, or the ClouderaManager node.
- After scaling down, nodes will show up as UNHEALTHY in Cloudera Manager. This is expected. While scaling down, stopped nodes are put into maintenance mode, to suppress alerts.

Attention: Autoscaling is not available for Data Hubs created with the default Streaming Analytics Light Duty cluster template.

Note: Make sure to create clusters with at least one node in each host group running YARN NodeManagers, for automatic resource configuration to take effect (unless explicitly configuring YARN node-level resources in the cluster template). Once a cluster has been created, the node count for the same can go to 0 (or the minimum can be set to 0 via autoscaling configuration).

Warning: Before performing the FreeIPA upgrade for Data Hub clusters where autoscaling is enabled, you must disable autoscaling and start all compute nodes to ensure the healthy state of Data Hub clusters. After upgrading FreeIPA, autoscaling can be enabled again. For more information, see the Autoscaling must be stopped before performing FreeIPA upgrade description in the Known issue.

Cluster management operations with load-based autoscaling enabled

Specific operations are affected as follows:

- Node repair: If the node on which Cloudera Manager is running is selected for repair, then all the stopped nodes need to be selected for repair.
- Cluster stop/start: Cluster stop is supported. During a cluster start, all nodes in the host group will be started and recommissioned.
- Node delete: Node delete (non-forced) is supported, including for STOPPED instances. Forced node delete may require you to End Maintenance (Enable Alerts/Recommission) in Cloudera Manager after performing another upscale operation.
- Retry: Do not attempt retry on an autoscale operation. Instead, see *Manually recovering from load-based scaling failures*. The nature of load-based autoscaling is to automatically try again after some time, based on the load on the cluster. There are a few scenarios where this retry will be adequate to continue cluster operations.

Only a single management operation (including scaling) can be performed on a cluster at a time. For example, if an upgrade is in progress, scaling cannot be performed until the upgrade completes.

EBS versus ephemeral storage

Caution: When you configure a load-based policy, you may receive a warning about using attached EBS volumes over a certain size. While instances are in a STOPPED state (scaled-down), EBS volumes continue to be charged. In order to reduce costs, Cloudera recommends using instances with ephemeral storage (for example, the r5d or m5d instance families) for compute nodes. Ephemeral volumes will also result in better performance of the workloads–they are typically faster, and don't utilize network throughput when writing data.

YARN Decommission Timeout

- Schedule-based downscaling will attempt to remove nodes to reach the target count, regardless of whether the node has running tasks, or data for a running job. You can use YARN DecommissionTimeout to control the leeway available to jobs beyond the scheduled decommission time. For example, with YARN decommission timeout set to one hour, a scheduled downscale will result in the following behavior:
  - Vacant nodes (no running tasks, no tasks previously run for running jobs) will be removed immediately.
  - Other nodes will be marked as decommissioning, and will not accept new work.
  - The remaining nodes will be terminated once any jobs complete, or at the one hour (YARN decommission timeout) mark.

For load-based autoscaling, the YARN DecommissionTimeout should be set to 30 seconds. See Configuring autoscaling for more details.

Configuring autoscaling

To configure autoscaling, add a load-based or schedule-based policy to a cluster and define the policy parameters.

Before you begin

If you are configuring a load-based autoscaling policy, you must set the YARN Node Decommission Timeout property to 30 seconds. Configure the following property in Cloudera Manager: yarn_resourcemanager_nodemanager _graceful_decommission_timeout_secs

Procedure

1. From the CDP Management Console, click Data Hub Clusters and then select the cluster that you want to add an autoscaling policy to. You can add autoscaling policies to clusters after they have been created, not during the cluster creation process.
2. From the cluster details page, select the Autoscale tab and then click the slider button to enable autoscaling. Autoscaling is disabled by default.
3. Click Add Autoscale Policy and select Load-Based or Schedule-Based.

4. Define the policy parameters.
   - For load-based scaling:

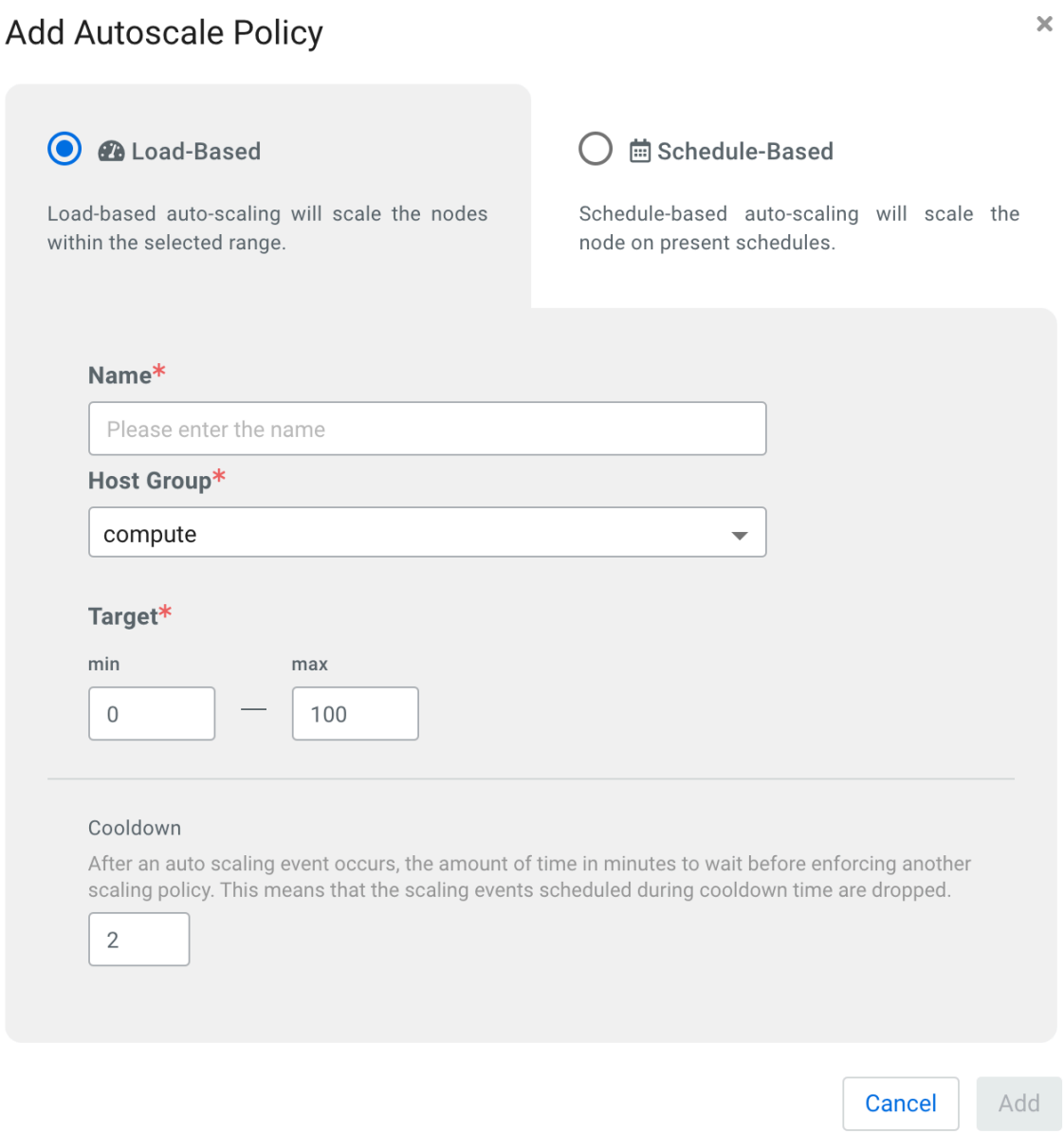

Parameter	Description
Name	Enter a unique name for the policy.
Hostgroup	Select the host group that you want to scale. The list of available host groups is determined by which host groups include services that can be scaled.
Target	Enter a minimum and maximum number of nodes for the policy. The maximum number of nodes determines how many instances are provisioned, but these instances are suspended and resumed as the workload demand requires. The policy will not provision instances beyond the maximum range of

Parameter	Description
	nodes that you define, regardless of the demand on the cluster.

- For schedule-based scaling:

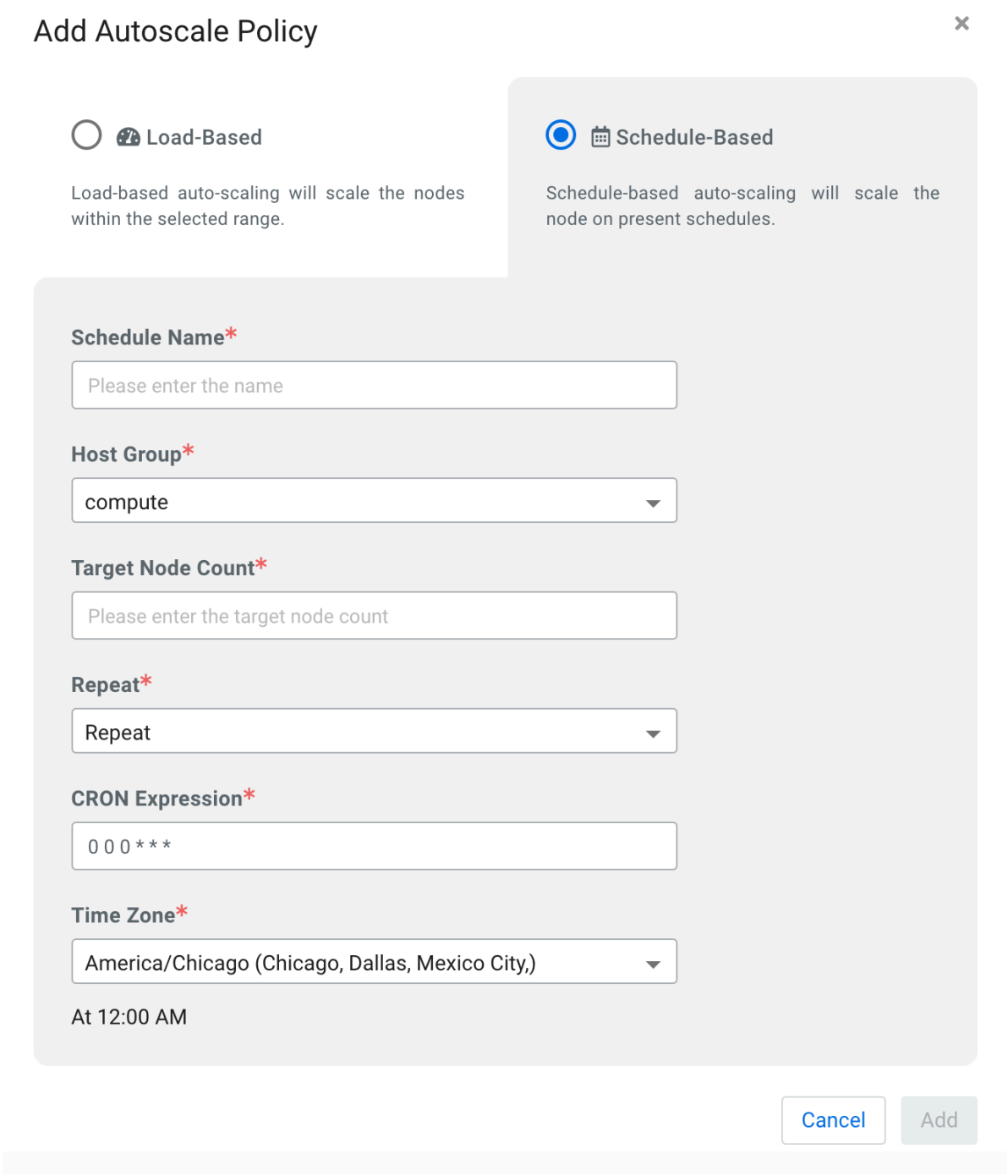

Parameter	Description
Schedule Name	Enter a unique name for the schedule.

Parameter	Description
Hostgroup	Select the host group that you want to scale. The list of available host groups is determined by which host groups include services that can be scaled.
Target Node Count	Enter the number of nodes that you want to upscale or downscale the host group to.
Repeat	Currently, only repeating schedules are supported.
CRON Expression	Enter a CRON expression string to define the details of the schedule that you want to create.
Time Zone	Select the appropriate timezone on which to base the CRON expression.

5. Click Add.

Dynamic Queue Scheduling

The Dynamic Queue Scheduling feature enables you to dynamically change the system state by reconfiguring the values of properties at a predefined time.

The Dynamic Queue Scheduling feature supports queue-level resource allocation configurations only. Additionally, the feature is only supported in relative and absolute resource allocation mode, and does not include dynamic queues.

The Dynamic Queue Scheduling feature enables you to set a time when a predefined configuration is applied to the YARN Queue Manager system.

Review the following scenario: There are two system states, State-A and State-B, that have to be scheduled for the system. State-A should be used from 8 AM to 8 PM, and State-B should be used from 8 PM to 8 AM. In this case two scheduling rules have to be created:

- Scheduling rule A takes effect at 8 AM and applies Dynamic Configuration State-A to the system.
- Scheduling rule B takes effect at 8 PM and applies Dynamic Configuration State-B to the system.

When a new Dynamic Configuration is created, it is based on the current configuration and enabled by default. Dynamic Configurations can be disabled at any time, either manually or by the system. If a Dynamic Configuration is an invalid configuration, it is disabled automatically by the system and it cannot be enabled manually until it is brought back into a valid state.

Dynamic Configurations do not contain any structural changes such as adding or deleting queues. Moreover, some manual structural changes can invalidate already existing Dynamic Configurations. As a result, existing Dynamic Configurations are validated when any structural changes occur. The following table explains how and when the system validates a Dynamic Configuration and the results of validation.

Table 2: Dynamic Configuration validation

Validation time	Triggering event	What is checked?	Action if Dynamic Configuration is invalid	Action if Dynamic Configuration is valid
Design time	When saving a Dynamic Configuration.	The system checks if the Dynamic Configuration is valid and can be applied.	The Dynamic Configuration is saved in a disabled state and marked as invalid.	Dynamic Configuration is saved in an enabled state.
Run time	When the system begins to apply a Dynamic Configuration.	The system checks if the Dynamic Configuration is valid and can be applied.	Dynamic Configuration is not applied and is disabled.	Dynamic Configuration is applied to the system.

In addition to the listed automatic validation, manual validation is required after queue configuration changes. This validation is not triggered automatically, but the UI will prompt you to perform Dynamic Configuration revalidation when you make any queue configuration changes. For more information, see *Revalidate Dynamic Configurations.*

Creating a new Dynamic Configuration

In Cloudera Manager, using the YARN Queue Manager UI you can create new Dynamic Configurations that will be used by the Dynamic Queue Scheduling feature.

Procedure

1. In Cloudera Manager, navigate to Clusters YARN Queue Manager UI .
2. Click Schedule.
3. Click +Add.
   The **Add Configuration** window is displayed with the current configuration setting.

4. Configure Queues.

When configuring queues for the Dynamic Configuration, you can either change a queue's configuration or include its current values.

a) Changing queue configuration:

Edit child queues by clicking the three vertical dots next to the parent queue's name and select Edit Child Queue.

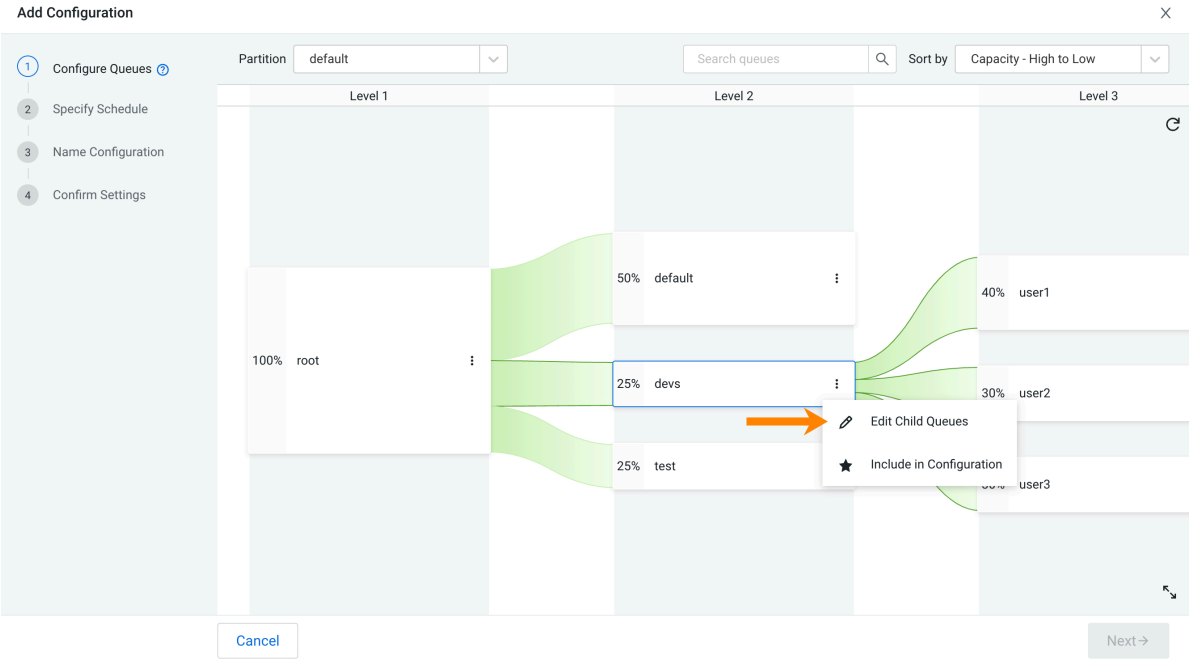

After you edit the child queues, they are included in the Dynamic Configuration which is indicated by a blue background. In relative resource allocation mode if one queue is included in the configuration at a certain level, all other queues on that level (siblings) are also included.

A blue asterisk next to the queue’s name indicates that you have changed the properties of that queue to create the Dynamic Configuration.

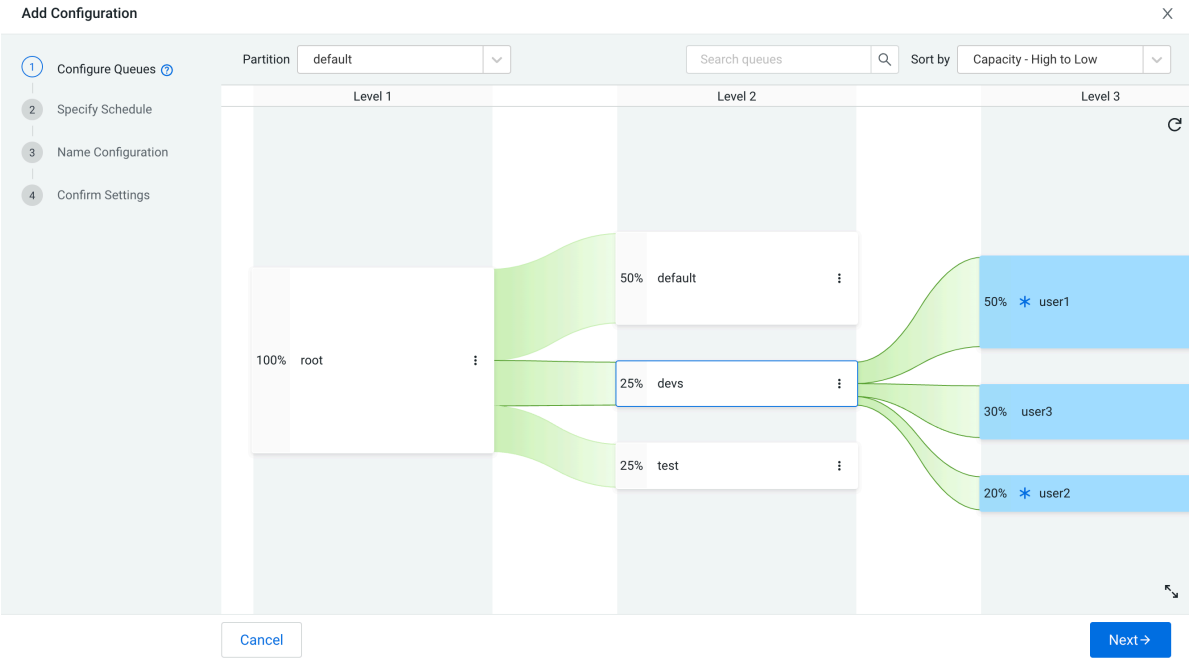

In this example the user1 and user2 queue's capacity properties were changed from 40% to 50% and from 30% to 20% respectively. The user3 queue has the same setting as it had in the configuration that is the base of the Dynamic Configuration that is currently being created.

b) Including the current values in the dynamic configuration:

Click the three vertical dots next to the queue which you want to include in the configuration and click Include in Configuration.

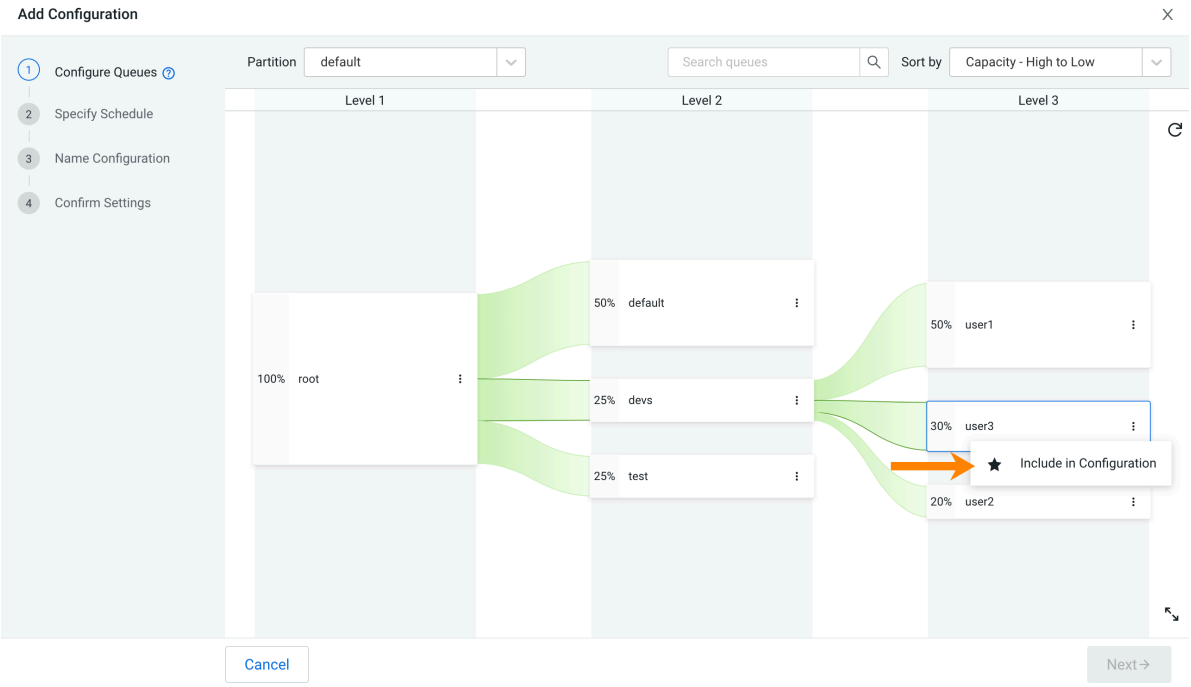

As a result, in relative resource allocation mode the selected queue and its sibling queues are included in the configuration with their current resource configuration. In absolute resource allocation mode, only the selected queue is added to the configuration.

As there was no change in the queue's resource configuration, no asterisk was added next to the queue's name.

5. Revert Changes.

If you want to revert changes you made to a queue, click the three vertical dots next to the name of the queue and select Revert Changes.

As a result, in relative resource allocation mode all changes on the selected queue and its sibling queues are reverted. In absolute resource allocation mode, only the changes of the selected queue are reverted.

6. Remove from Configuration.

If you want to remove a queue with its current resource configuration from the Dynamic Configuration, click the three vertical dots next to the queue's name and click Remove from Configuration.

As a result, in relative resource allocation mode the selected queue and its sibling queues are removed from the configuration. In absolute resource allocation mode, only the selected queue is removed to the configuration

7. Click Next.

8. Specify Schedule.

Important: When specifying schedule the date and time that is taken into account is the server date and time, not the local date and time of the user.

Set up a schedule to control how frequently the Dynamic Configuration is applied.

The time and frequency can be configured using the various options available on the page. Once an option is selected a note appears on the top of the window that tells you the exact time and frequency when the schedule is applied.

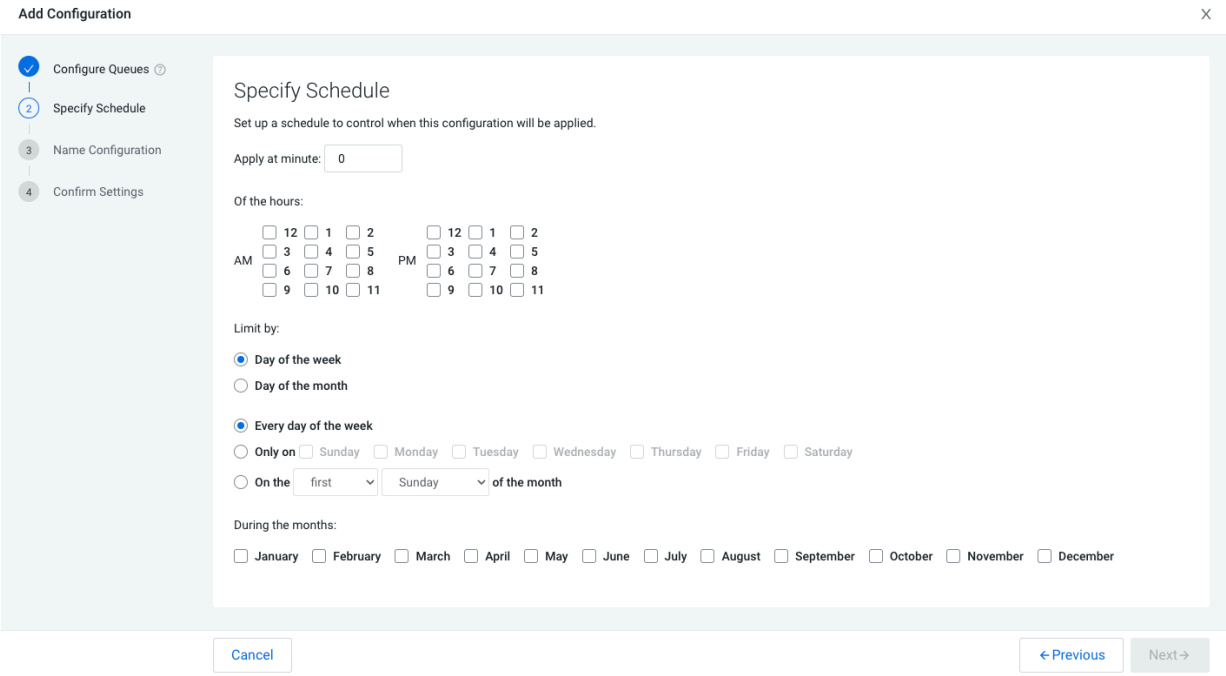

Using the Apply at minute: and Of the hours: options, you can set the exact time when a configuration should be applied. The Of the hours: section allows multiple values to be selected simultaneously. The Apply at minute: option however only accepts a single value. This means that while you can specify a schedule to apply at 6:00 AM and at 8:00 AM, you can't specify a schedule to apply for example at 6:00 AM and 8:15 AM. If you use the

default 0 value for the Apply at minute: option, and you do not select any hours, then the configuration will be applied at every hour.

You can configure the schedule on a weekly or a monthly basis, by selecting either Limit by: Day of the week or Limit by: Day of the month.

If you select Day of the week, you can choose from the following options:

- Every day of the week
- Only on: The configuration is applied on selected day or days of the week.
- On the *[***FIRST-LAST***] [***MON-SUN****]* of the month: The configuration is applied on the selected day.

If you select Day of the month, you can choose from the following options:

- Every day of the month
- Every # days starting on day # of the month: The configuration is applied on the specified days based on the configured starting day and frequency.
- On day(s): The configuration is applied on the specified days. This option accepts integers 1-31. You can specify a single integer to select a single day, specify a range to select multiple days, or you can also add a combination of the two. For example, 1-5, 8, 11-13.
- # day before the last day of the month
- On the weekday nearest day # of the month
- On the last weekday of the month

In addition, you can specify month or months when the schedule is applied. If you do not specify any month, the schedule is applied in every month.

Consider the following example:

- Apply at minute: 15
- Of the hours: AM 2, 6 and PM 2, 6
- Limit by: Day of the week
- Only on: Monday, Wednesday
- During the months: February, March

Then the configuration will be applied at 02:15 AM, 06:15 AM, 02:15 PM and 06:15 PM, only on Monday and Wednesday, only in February and May. The UI will display the schedule:

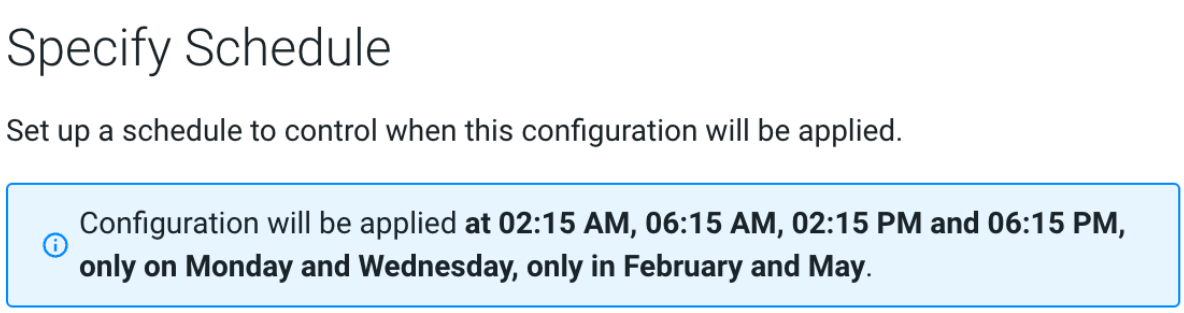

9. Click Next.
10. Name Configuration.

    Provide a unique name for the Dynamic Configuration and click Next.
11. Confirm Settings.

    Review and confirm the scheduling rule and click Finish.

Managing Dynamic Configurations

In Cloudera Manager using the Queue Manager UI you can manage the Dynamic Configuration you have created for the Dynamic Queue Scheduling feature.

Procedure

1. In Cloudera Manager, click Clusters YARN Queue Manager UI .
2. Select Schedule.
3. Find the Dynamic Configuration you want to manage from the list.
4. Click the three vertical dots for the scheduling rule.
5. Select an action:
   - View: Displays the configuration details of the Dynamic Configuration.You can review the name, schedule, and queue configuration of the Dynamic Configuration.
   - Disable/Enable: Allows you to enable or disable the Dynamic Configuration. If you disable a Dynamic Configuration it will not be applied to your system at its scheduled time. Later, you can enable it again.

     Note: Invalid Dynamic Configurations are disabled automatically and cannot be enabled manually until they are brought back into a valid state.
   - View conflict: Available if the Dynamic Configuration is conflicted with any previously created Dynamic Configuration. Use this action to review the conflicts or to accept the Dynamic Configuration with conflicts.
   - Edit: Allows you to edit details of the Dynamic Configuration, such as queue configuration, schedule, or name.
   - Delete: Allows you to delete the Dynamic Configuration.
   - Activate Now: Allows you to activate the queue configuration of this Dynamic Configuration immediately.

How to read the Configurations table

In the Queue Manager UI you can view all of your Dynamic Configurations on one page. Learning about this page can help you manage the Dynamic Queue Scheduling feature.

The schedule overview page is displayed in Cloudera Manager if you select the Queue Manager UI and then go to the Schedule tab. It displays a table with the following columns:

Table 3: Schedule table

Column name	Description
Status	The status of the Dynamic Configuration: • Valid and enabled • Valid and disabled • Invalid • Disabled due to scheduling conflict • Enabled with scheduling conflict • The resource allocation mode of the Dynamic Configuration differs from the currently used resource allocation mode. As a result the Dynamic Configuration will fail when executed.
Name	The name of the Dynamic Configuration.
Scheduled At	Information about the time at which the Dynamic Configuration is scheduled to be applied to the system.

Handling Dynamic Configuration conflicts

A dynamic queue configuration related scheduling conflict occurs if two or more Dynamic Configurations are scheduled to be applied at the same time. You can either enable Dynamic Configuration with conflicts or modify the conflicting Dynamic Configuration to resolve conflicts.

About this task

When a Dynamic Configuration is created with a schedule that conflicts with one or more other Dynamic Configuration, the new Dynamic Configuration will be saved in a disabled with conflict state.

Note: Although recurrence patterns of Dynamic Configurations can be open ended, scheduling conflicts are detected only one year ahead of time.

Procedure

1. Find the Dynamic Configuration that is conflicted.
2. Click the three vertical dots in the line of the conflicted Dynamic Configuration.
3. Select View Conflicts.

   All conflicting Dynamic Configurations and conflicting times are displayed.

4. Review the conflicts and perform one of the following:
   - Accept the existing conflicts:
     a. Select Accept all conflicts.
     b. Click Enable.

     The system will execute the conflicted Dynamic Configurations one after another without any predetermined order.
   - Resolve conflicts by modifying the conflicting schedule:
     a. Click Cancel.
     b. Click the three vertical dots in the line of the conflicted Dynamic Configuration.
     c. Select Edit.
     d. Modify the Dynamic Configuration to resolve the conflict.
     e. Click Save.

Revalidating Dynamic Configurations

Dynamic Configurations need to be revalidated after queue configuration changes. You can revalidate Dynamic Configurations using the YARN Queue Manager UI.

About this task

When changing the queue configuration, on the bottom of the queue's configuration dialog box the following message is displayed to remind you that Dynamic Configuration revalidation needs to be performed:

```
Changes you make here may invalidate dynamic configurations, so you will nee
d to revalidate them. This can be performed on the Schedule tab.
```

In addition a red dot appears next to the Scheduler tab, indication that some action needs to be taken related to Dynamic Queue Scheduling.

Note: The revalidation of Dynamic Configurations can take a significant amount of time if you have a higher number of queues. Until the revalidation is done you cannot use the Schedule tab of the YARN Queue Manager UI.

Procedure

1. In Cloudera Manager nevigate to the YARN Queue Manager UI.
2. Click Schedule.

   After any queue change the following message is displayed on the Schedule page:

   ```
   Revalidation Required: All dynamic configuration must be revalidated due
    to queue changes. Please click Revalidate to proceed.
   ```
3. Click Revalidate.

   As a result an orange dot next to the Schedule tab, and Revalidation in progress status text appear.
4. Wait until revalidation is finished.

   You do not have to stay on the Schedule page until the Revalidation is done. Once the orange dot next to the Schedule tab disappears , the revalidation is finished.
5. Check if any of the Dynamic Configuration became disabled.

6. Perform one of the following actions for the disabled Dynamic Configurations:
   - Update the queue configuration to make the Dynamic Configuration valid
   - Update the Dynamic Configuration to make it valid
   - Delete the invalid Dynamic Configuration

Dynamic Configurations execution log

The execution log page displays all Dynamic Configurations that are executed, either according to their schedule or manually. The log provides information if those Dynamic Configurations were applied successfully or failed.

To view the execution log navigate to YARN Queue Manager UI Schedule Execution Log:

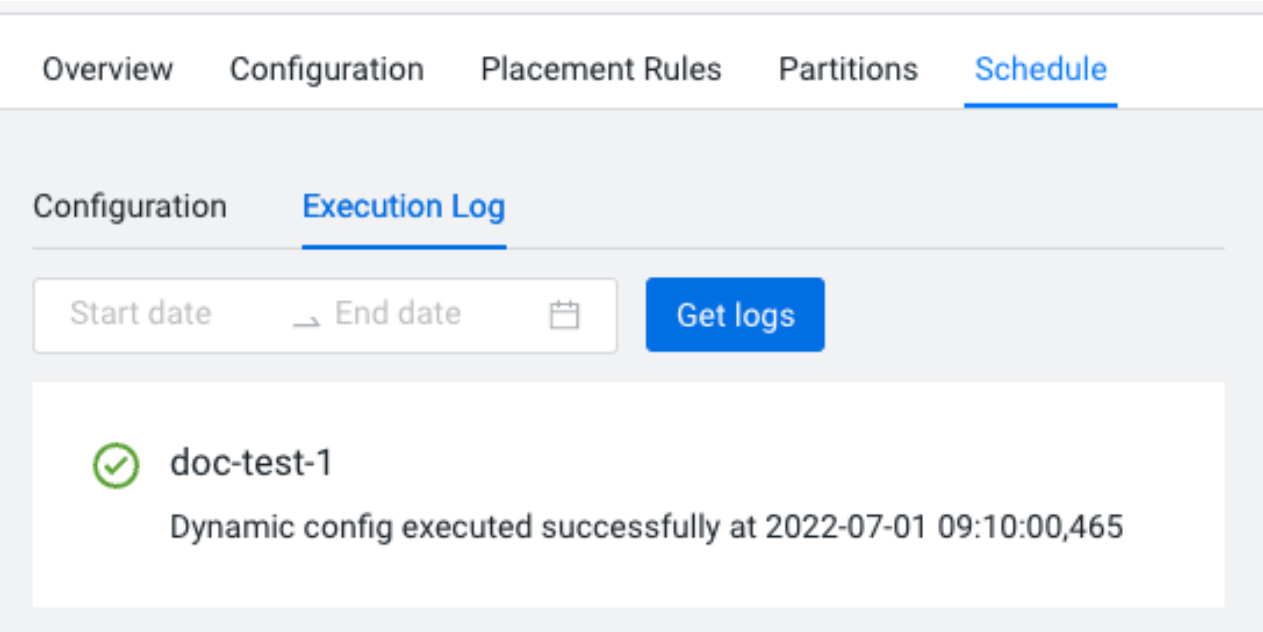

The execution log will display all executed Dynamic Configuration providing the following information:

- Name of the Dynamic Configuration
- If the Dynamic Configuration was applied successfully or failed
- The exact date and time of the execution

Note: The Dynamic Configuration name is not displayed correctly when it is manually activated.

You can also set a Start date and End date to view the logs of a specific time period. When they are not specified the default time period is 7 days, meaning that the execution logs for the last 7 days are displayed.

Managing placement rules

Placement rules can define the logic that is taken into account when specifying which queue should be used for a submitted job. These predefined rules enable you to submit jobs without specifying the queue name at the time of job submission.

There are two kinds of queue to which jobs can be submitted:

- Static queues: Queues that always exist and were defined by the user using the Queue Manager UI (or configuration files).

- Dynamic queues: Queues that are created dynamically when jobs are submitted to them. If the YARN service is restarted they are automatically deleted. To learn more about dynamic queues, see *Manage dynamic queues*.

Placement rules enable you to define the logic that applies when a job is submitted to specify which queue should be used for the submitted job. That enables you to submit a job without defining a target queue or to even override the target queue that was specified by the submitter during the job submission.

By default, placement rules are taken into account only if no target queue is specified during a job submission or if the specified target queue is provided as “default”. To change this behavior, see *Enable override of default queue mappings*.

Placement rules are evaluated in the order in which they appear in the placement rule list. When a job is submitted and placement rules have to be taken into account, the rules are evaluated, and the first matching rule is used to determine the queue in which the job runs.

If there is no placement rule and no target queue was specified during the job submission, then the job is submitted to the default queue of the scheduler.

If the target queue of a placement rule does not exist or it cannot be created, the configured fallback action is performed. You can configure the fallback action for each placement rule, and it can have the following values:

- Skip: Ignore the current rule and proceed to the next.
- PlaceDefault: Place the application to the default queue root.default (unless it is overridden to something else).
- Reject: Reject the submission.

If no target queue was specified during the job submission and no placement rule matches the job, then the job is submitted to the default queue of the scheduler.

By default, if an invalid queue was specified during a job submission, the submission is rejected. To change this behavior, see *Enable override of default queue mappings*.

Related Information

Managing dynamic queues

Placement rule policies

When creating a placement rule you have to set its policy to define the logic that is taken into account when specifying which queue should be used for a submitted job. There are many predefined policies you can choose from, or you can create a custom policy.

Important: When referring to a queue, Cloudera recommends to always provide the parent queue as well. Although, in Capacity Scheduler you can refer to a queue only by its leaf queue name, it can lead to issues if there are more leaf queues with the same name. Providing the parent queue ensures that the reference is translated to a fully qualified path, that is, there will be no ambiguity.

The following table lists the names of the policies, the options displayed in the Placement Rule creation dialog box in the Queue Manager UI, and their detailed description:

Table 4: Placement rule policies

Policy	Queue Manager UI	Description
user	Place the application into a queue named for the user.	Places the application into a queue which matches the username of the submitter.
primaryGroup	Place the application into a queue named for the user’s primary group.	Places the application into a queue which matches the primary group of the submitter.
primaryGroupUser	Place the application into a queue named for the user that is the child of a queue named for the user’s primary group.	Places the application into the queue hierarchy [parentQueue].<primaryGroup>.<userName>. The parentQueue is optional.

Policy	Queue Manager UI	Description
secondaryGroup	Place the application into a queue named for the user's secondary group.	Places the application into a queue which matches the secondary group of the submitter.
secondaryGroupUser	Place the application into a queue named for the user that is the child of a queue named for the user's secondary group.	Places the application into the queue hierarchy [parentQueue].<secondaryGroup>.<user Name>. The parentQueue is optional.
applicationName	Place the application into a queue named for the application.	Places the application into a queue which matches the name of the application. **Important:** It is case-sensitive, white spaces are not removed.
specified	Place the application in the queue specified at the run time.	Places the application to the queue that was defined during submission.
reject	Reject the application.	Rejects the submission.
defaultQueue	Place the application into the default queue.	Places the application into the default queue root.default or to its overwritten value.
setDefaultQueue	Set the default queue to:	Changes the default queue from root.default. This policy does not permanently change the default queue. It is always root.default when the evaluation begins when an application is submitted. However, the adjusted default queue remains in effect until the placement rules evaluation is completed. **Important:** • This policy rule does not place the application in the specified queue. • Fallback will not be triggered, no matter what the selection is.
custom	Use the following custom policy:	Enables the user to use custom placement strings.

Using the custom placement policy

The custom placement policy can describe other policies with the appropriate variable placeholders. For example, primaryGroupUser with the parent queue root.groups can be expressed as: root.groups.%primary_group.%user

The *Custom policy variables* table described what variables are available if you intend to use the custom policy.

Internally, the tool populates certain variables with appropriate values. These can be used if custom mapping policy is selected. The placement rules evaluation engine does only minimal verification when it comes to replacing them. Therefore it is your responsibility to provide the correct string.

Table 5: Custom policy variables

Variable	Meaning
%application	The name of the submitted application.
%default	The default queue of the scheduler.
%primary_group	Primary group of the submitter.

Variable	Meaning
%secondary_group	Secondary (supplementary) group of the submitter.
%specified	Contains the queue that the submitter defined. It is a standalone variable, do not combine it with other custom variables or paths. If the specified target queue is default this variable will not be set. If the target queue is the default queue, it should be specified with the root.default parent path.
%user	The user who submitted the application.

If you submit a MapReduce application to a queue root.users.mrjobs, the value of %specified will be set to root.users.mrjobs.

Few predefined policies can be achieved with the custom policy. For example, instead of using the specified policy, you can use the custom policy and set the customPlacement field to %specified.

However, you have more control over the policy if you use custom instead of the predefined one. That is because you can also append or prepend an extra string to these variables. As a result, for example, the following setting is possible:root.groups.%primary_group.admins.%user.longrunningjobs

The string must be resolved to a valid queue path to have a proper placement.

Differences between legacy modes and weight mode

There are some cases when the legacy resource allocation modes (absolute and relative mode) behave differently than weight mode.

The create flag

- Legacy modes: It has no effect if the parent is not managed.
- Weight mode: It applies to all parent queues. However, if yarn.scheduler.capacity.<queue-path>.auto-queue-creation-v2.enabled is set to false, then the queue will not be created.

The nested rules primaryGroupUser and secondaryGroupUser

- Legacy modes: They expect the parent queues to exist, meaning they cannot be created automatically. More specifically, when you use primaryGroupUser it will result in a queue path similar to root.<primaryGroup>.<userName> and the root.<primaryGroup> parent queue must exist. It can be a managed parent in order to have the userName leaf created automatically, but the parent still has to be created manually.
- Weight mode: As long as the parent allows dynamic queues to be created, there is no limitations. The requested queues will be created.

How to read the Placement Rules table

In the Queue Manager UI you can view all of your placement rules on one page. Learning about this page can help you manage your placement rules according to your needs.

The placement rules overview page is displayed in Cloudera Manager if you select the Queue Manager UI and then go to the Placement Rules tab. It displays a table with the following columns:

Table 6: Placement rules overview page

Column Name	Description
Order	Placement rules are evaluated from top to bottom. This column provides the order of the placement rules.

Column Name	Description
Type	Indicates to the engine what the current rule should be matched against: application, user, or group. Values: Application, Users, Group
Match	The string to match, or an asterisk "*" which means "all". For example, if the type is User and this string is "hadoop" then the rule will only be evaluated if the submitter user is "hadoop". The "*" does not work with groups.
Policy	Pre-defined or custom policies which defines where the application should be placed. Values: User, PrimaryGroup, PrimaryGroupUser, SecondaryGroup, SecondaryGroupUser, ApplicationName, Specified, Reject, DefaultQueue, SetDefaultQueue, Custom
Parent Queue	In the case of User, PrimaryGroup, PrimaryGroupUser, SecondaryGroup, and SecondaryGroupUser policies, this tells the engine where the matching queue should be looked for. For example, if the policy is PrimaryGroup, parent is root.groups and the submitted's group is "admin", then the resulting queue will be: root.groups.admin
Custom Value	It can have two types of values: • In the case of the SetDefaultQueue policy: The default queue will change from root.default to the value of this field. • In the case of the Custom policy: The value of this field will be evaluated directly by the placement rules evaluation engine, which means that various placeholders such as %application or %primary_group will be replaced with their respective values.
Create Queue?	It sets the create flag and it works differently in weight and in legacy mode. If it is set to No, the target queue determined by the placement policy will not be created if it does not exist. That means that auto dynamic child creation will not happen. However, even if it is set to Yes it is still not guaranteed that the queue will be created. You also have to ensure that for the specified parent queue the auto dynamic child creation feature is enabled. The following policies do not support the create flag: • reject • setDefaultQueue • defaultQueue
Fallback	If the target queue does not exist or it cannot be created, it defines a fallback action. Values: Skip, PlaceDefault, Reject
Actions	Click on the applicable icon to perform actions on the placement rules: • View Rule • Edit Rule • Delete Rule

Creating placement rules

You can create placement rules using the YARN Queue Manager UI in Cloudera Manager.

Before you begin

Placement rule that uses static queue

Create the target leaf queue before creating the placement rule that uses it. When creating the placement rule the UI will display all the existing leaf queues.

Placement rule that uses dynamically created queues

Enable the dynamic child creation feature for the target parent queue before creating the placement rule that uses it. When creating the rule the UI will display all the existing queues as target parent queue options but if the dynamic child creation feature is not enabled for the selected queue, a warning message will be displayed and you cannot create the placement rule. For more information, see *Manage dynamic queues*.

Procedure

1. In Cloudera Manager, select the YARN Queue Manager UI.
   A graphical queue hierarchy is displayed in the Overview tab.
2. Navigate to Placement Rules.

3. Click +Add.
   The Add Placement Rules dialog box is displayed with the default settings or with settings that are the exact replica of the last placement rule that you have created.

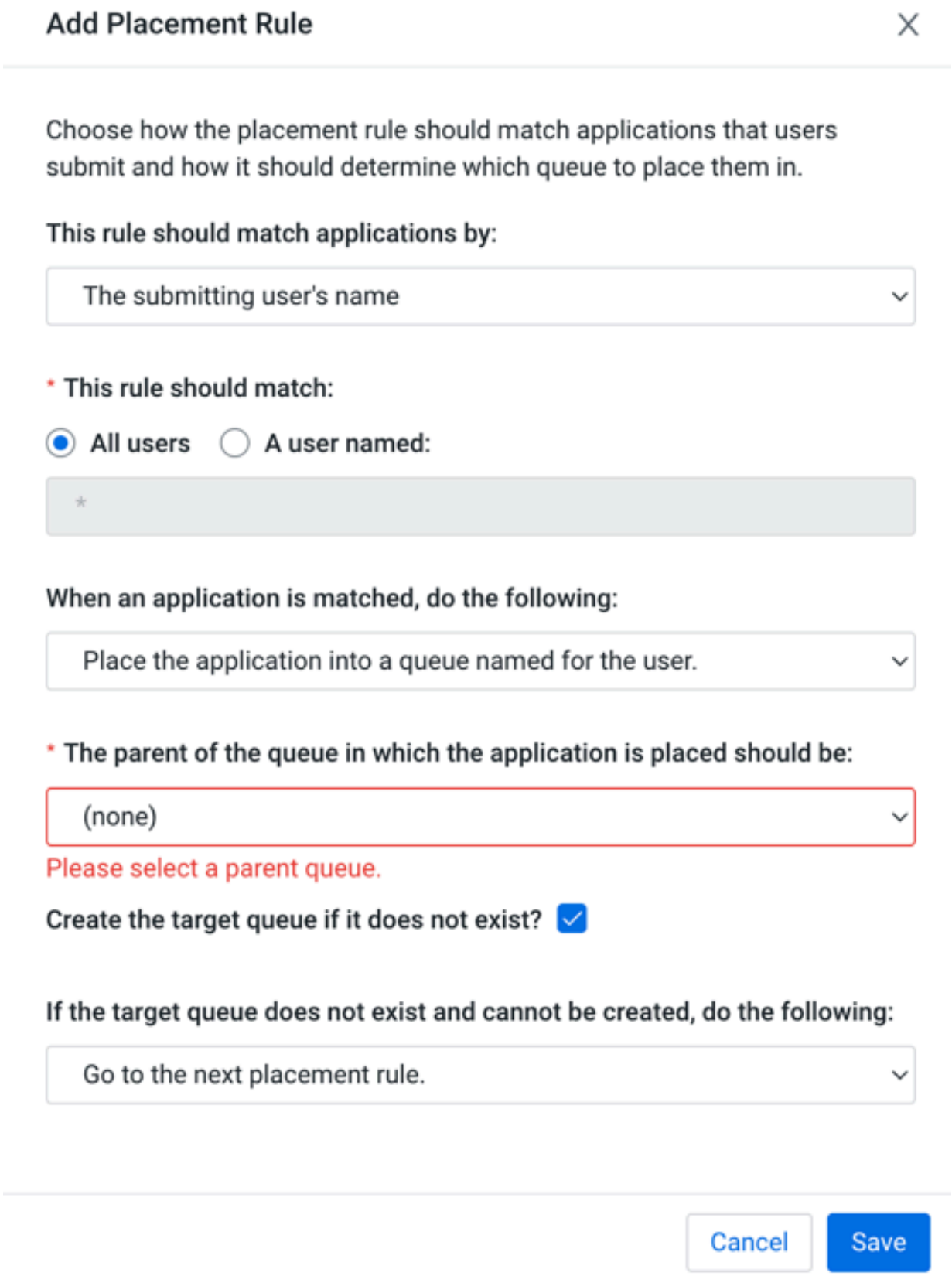

4. Use the The rule should match application by property to set the application parameter that should match with the rule.
5. Use the This rule should match field to set the value of the matching application parameter that should match with the rule.
6. Set what the rule should do when an application is matched.

   When an application is matched, do the following: Sets the placement rule policy. The following options are available:

   - Place the application into a queue named for the user.
   - Place the application into a queue named for the user's primary group.
   - Place the application into a queue named for the user that is the child of a queue named for the user's primary group.
   - Place the application into a queue named for the user's secondary group.
   - Place the application into a queue named for the user that is the child of a queue named for the user's secondary group.
   - Place the application into a queue named for the application.
   - Place the application in the queue specified at run time.
   - Place the application into the default queue.
   - Reject the application.
   - Set the default queue to:
   - Use the following custom policy:
7. Set additional configuration for the placement rule.

   Depending on the selected policy, different options become available. These are as follows:

   Table 7: Policy specification

Selected policy	Additional configuration
• Place the application into a queue named for the user. • Place the application into a queue named for the user's primary group. • Place the application into a queue named for the user that is the child of a queue named for the user's primary group. • Place the application into a queue named for the user's secondary group. • Place the application into a queue named for the user that is the child of a queue named for the user's secondary group. • Place the application into a queue named for the application.	The The parent of the queue in which the application is placed should be dropdown list appears. Select the parent of the queue to which the job should be submitted. **Important:** Cloudera recommends to always set the parent queue when it is an available property, even if it is only optional. In this way issues caused by leaf queues with the same name can be avoided.
• Place the application in the queue specified at run time. • Place the application into the default queue. • Reject the application.	No additional configuration required.
Set the default queue to:	A dropdown list appears. Select which available queue you want to make the default queue.
Use the following custom policy:	A text box appears in which you can provide the custom policy you want to use.

8. If you want the target queue to be created, if it does not exist at the time of the job submission, select the Create the target queue if it does not exist? checkbox. To enable this feature you have to set a parent queue in Step 7.

   Note: You have to enable the dynamic child creation feature for the parent queue you select if you want the target queue to be created if it does not exist at the time of the job submission.

9. Use the If the target queue does not exist and cannot be created, do the following property to set the fallback action.

   The following fallback actions are available:

   - Go to the next placement rule.
   - Place the application into the default queue.
   - Reject the application.
10. Check you placement rule settings.
11. Click OK.
12. Provide a description of the change and then click OK.

Results

The rule is added to the bottom of the placement rule list and becomes the last rule to be evaluated.

Related Information

Managing dynamic queues

Example - Placement rules creation

You have to set up your placement rules to fulfill your specific need. In this example a cluster is shared among developers, QA engineers and test developers and there are nine placement logics that you want to set up.

The following is the placement logic that has to be achieved when creating placement rules:

1. If the user belongs to the devs primary group, the application should be placed to root.users.devs. This is reserved for developers.
2. If the user belongs to the qa primary group, then the application should go to root.users.lowpriogroups.<primaryGroup>. These queues have lower capacities and are intended for testers.
3. If the user belongs to the qa-dev primary group, then the application should go to root.users.highpriogroups.<primaryGroup>. These queues have higher capacities and are intended for test developers.
4. Place the application into the queue which matches the user name.
5. If there is no such queue, take the queue from the application submission context, but the queue should not be created if it does not exist and the parent is managed.
6. If none of the above matches, then the application should be placed into the root.default queue.
7. If the default placement fails , change the default queue to root.users.default.
8. Try a placement to the default queue again.
9. If that fails, reject the submission altogether.

Using the Queue Manager UI, this logic can be achieved in the following way:

Queue hierarchy

Queue with bolt signs next to their names are auto dynamic child creation enabled parents.

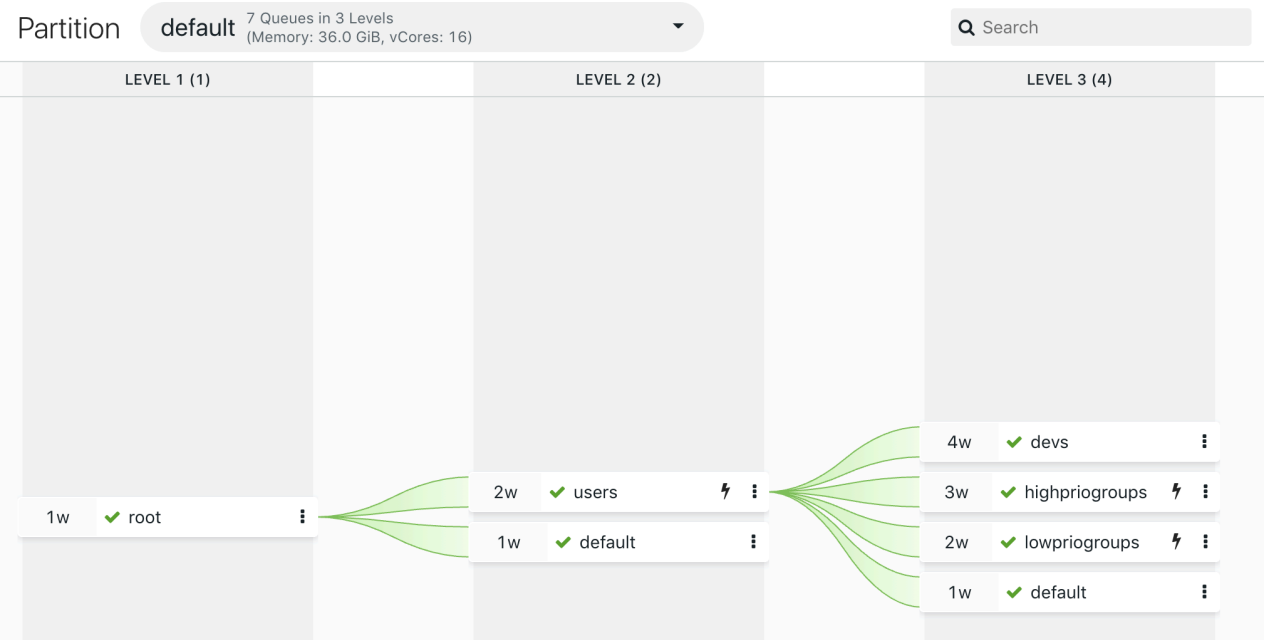

Placement rules overview

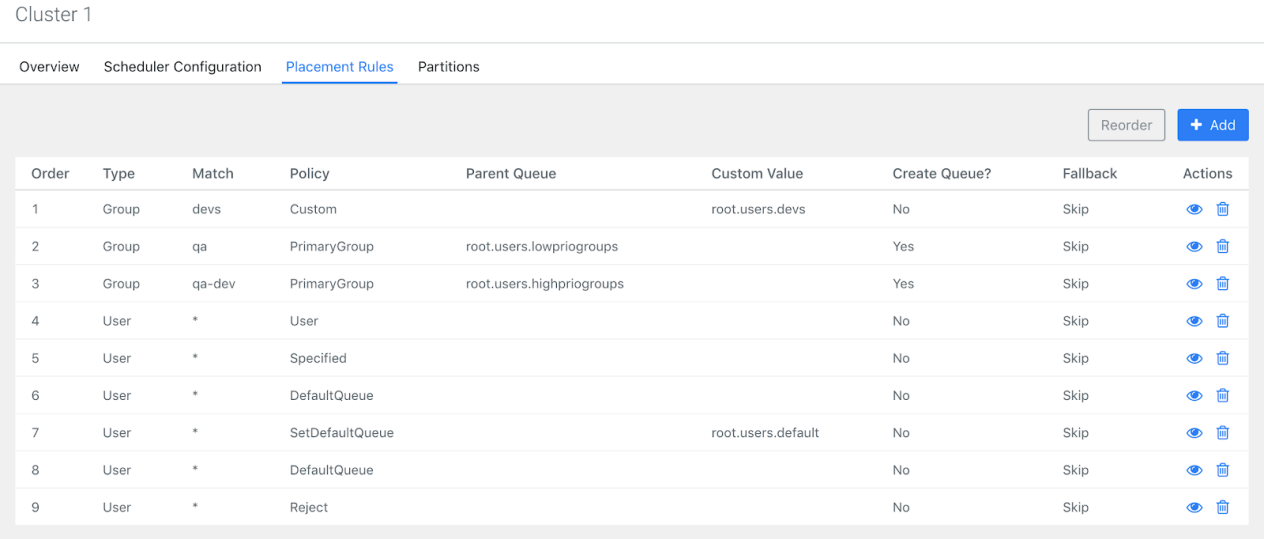

Order	Type	Match	Policy	Parent Queue	Custom Value	Create Queue?	Fallback	Actions
1	Group	devs	Custom		root.users.devs	No	Skip	
2	Group	qa	PrimaryGroup	root.users.lowpriogroups		Yes	Skip	
3	Group	qa-dev	PrimaryGroup	root.users.highpriogroups		Yes	Skip	
4	User	*	User			No	Skip	
5	User	*	Specified			No	Skip	
6	User	*	DefaultQueue			No	Skip	
7	User	*	SetDefaultQueue		root.users.default	No	Skip	
8	User	*	DefaultQueue			No	Skip	
9	User	*	Reject			No	Skip	

Reordering placement rules

Placement rules are evaluated in the order in which they appear in the placement rule list. When a job is submitted, the rules are evaluated, and the first matching rule is used to determine the queue in which the job is run.

About this task

When a job is submitted, the rules are evaluated from top to bottom, and the first matching rule is used to determine the queue in which the job is run.

If a rule is always satisfied, subsequent rules are not evaluated. By default, placement rules are ordered in the order they were added; the rule added first appears first. You can easily reorder rules.

Procedure

1. In Cloudera Manager, select YARN Queue Manager UI.
   A graphical queue hierarchy is displayed in the Overview tab.
2. Go to the Placement Rules tab.
   A list of placement rules is displayed.
3. Click Reorder.

   The Reorder option is only available if you have at least two placement rules.
4. Click Move Up and Move Down arrow buttons in a rule row.
5. Click Save Reorder.

Editing placement rules

The Queue Manager UI enables you to edit previously created placement rules.

Procedure

1. In Cloudera Manager, select YARN Queue Manager UI.
   A graphical queue hierarchy is displayed in the Overview tab.
2. Go to the Placement Rules tab.
   A list of placement rules is displayed.
3. In the Action column click on the Edit rule icon in the row of the placement rule you want to edit.
   The Edit Placement Rule dialog box is displayed.
4. Edit the placement rule.
5. Click Save.

Deleting placement rules

The YARN Queue Manager UI enables you to delete previously created placement rules. If you want to delete queues associated with placement rules, you first have to delete its associated placement rules.

Procedure

1. In Cloudera Manager, select YARN Queue Manager UI.
   A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Placement Rules tab.
   A list of placement rules is displayed.
3. In the Action column click on the Bin icon in the row of the placement rule you want to delete.
4. Click Save.

Enabling override of default queue mappings

By default, placement rules are considered only if a target queue is not specified during a job submission.You can change that behaviour to take placement rules into account whether a target queue is specified at job submission or not.

About this task

The yarn.scheduler.capacity.queue-mappings-override.enable property controls when placement rules are considered. In the YARN Queue Manager UI, this property is called Override Queue Mapping. By default, the property is set to false, which means the feature is disabled and placement rules cannot override the target queue that was specified at job submission.

The following table shows how it is specified which queue should be used for a job in different scenarios:

Table 8: Target queue speficiation scenarios

Override Queue Mapping	Target queue is specified at job submission?	Placement rules exist?	End result
Disabled (set to false)	Yes	Yes	Job is submitted to the queue that was specified by the submitter.
Disabled (set to false)	Yes	No	Job is submitted to the queue that was specified by the submitter.
Disabled (set to false)	No	Yes	Placement rules specify the target queue.
Disabled (set to false)	No	No	Job is submitted to the default queue of the scheduler (root.default).
Enabled (set to true)	Yes	Yes	Placement rules specify the target queue.
Enabled (set to true)	Yes	No	Job is submitted to the queue that was specified by the submitter.
Enabled (set to true)	No	Yes	Placement rules specify the target queue.
Enabled (set to true)	No	No	Job is submitted to the default queue of the scheduler (root.default).

When placement rules override the target queue defined at job submission, the specified queue can still be taken into account if the specified placement rule policy is used. For more information, see *Placement rule policies*.

Procedure

1. In Cloudera Manager, select YARN Queue Manager UI.
   A graphical queue hierarchy is displayed in the Overview tab.
2. Go to the Scheduler Configuration tab.
3. Find the Override Queue Mappings property. It is disabled by default.

4. Check the box to enable this feature.

5. Click Save.
6. Provide a description of the change and then click OK.

Results

Placement rules are considered even if a target queue is specified during a job submission.

Managing dynamic queues

Dynamic queues are created automatically during application runtime. They are deleted when the YARN service is restarted.

Dynamic queues are created during runtime automatically. Dynamic queues are not defined in the capacity-sch eduler.xml configuration file. Dynamic queues can be created under those static parents where auto dynamic child creation is allowed. Auto dynamic child creation has to be set explicitly using the YARN Queue Manager UI.

You can create dynamic queues in two ways:

- A dynamic queue path is specified by the submitter at the time of the job submission. Dynamic queues will be created if the auto dynamic child creation feature is enabled for the parent queue that was provided in the queue path.
- A placement rule applies to the submitted job that could place it into a dynamic queue. Dynamic queue is created based on the predefined expression of the dynamic placement rule.

Based on your resource allocation mode, dynamic queues are managed differently:

In absolute and relative modes, when you enable the auto dynamic child creation feature for a queue, it becomes a Managed Parent Queue. It cannot have static child queues, queues under it can be created only dynamically. It allows 1-level dynamic queue nesting.

In the weight mode, there is no Managed Parent Queue. When you enable the auto dynamic child creation feature for a queue, it becomes a parent queue that can have both static and dynamic child queues. It allows 2-level dynamic queue nesting.

Note: Although it is possible to use safety valve configuration snippets to configure dynamic queues, Cloudera recommends to use the YARN Queue Manager UI for dynamic queue configuration even if that leads to some limitations.

Managed Parent Queues

Managed Parent Queues are auto dynamic child creation enabled queues in absolute and relative resource allocation mode.

In absolute and relative modes, when you enable auto dynamic child creation for a queue it becomes a Managed Parent Queue. It cannot have static child queues, and queues under it can be created only dynamically.

In absolute and relative modes, dynamically created queues always fall under a predefined (static) queue, which is the Managed Parent Queue. This limits the nesting to only one level. In addition, the queue properties set for the Managed Parent Queue will be applied to all of its dynamically created child queues. To change the queue properties of all its dynamic child queues, you have to change the configuration at the Managed Parent Queue level.

It is possible to dynamically create queues with zero capacity by incorrectly setting the Managed Parent Queue.

For example, you can create a Managed Parent Queue and assign a percentage based minimum capacity limit of 5%, to its dynamically created child queues. In this case, at most 20 queues can function with the 5% capacity limit. This forces all the following queues to wait until a queue is released (if no application is running in a queue, its capacity is set to zero). Therefore, it is vital to design the minimum capacity limit of child queues that belong to a Managed Parent Queue in a way that takes into account the number of queues that should be running in parallel.

Converting a queue to a Managed Parent Queue

In absolute and relative modes, you have to create Managed Parent Queues to enable dynamic queue creation. You can do that through the YARN Queue Manager UI.

About this task

Note that the queue properties that is set at the Managed Parent Queue level is applied to all of its leaf queues.

Important: Once dynamic child creation is enabled for a queue you cannot disable it. If you enable it by mistake you have to delete the queue and its child queues and then recreate them.

Procedure

1. In Cloudera Manager, select the YARN Queue Manager UI.
   A graphical queue hierarchy is displayed in the Overview tab.
2. Find the queue for which you want to enable the dynamic child creation feature.
3. Select the More Options menu and select Enable Dynamic Child Creation.

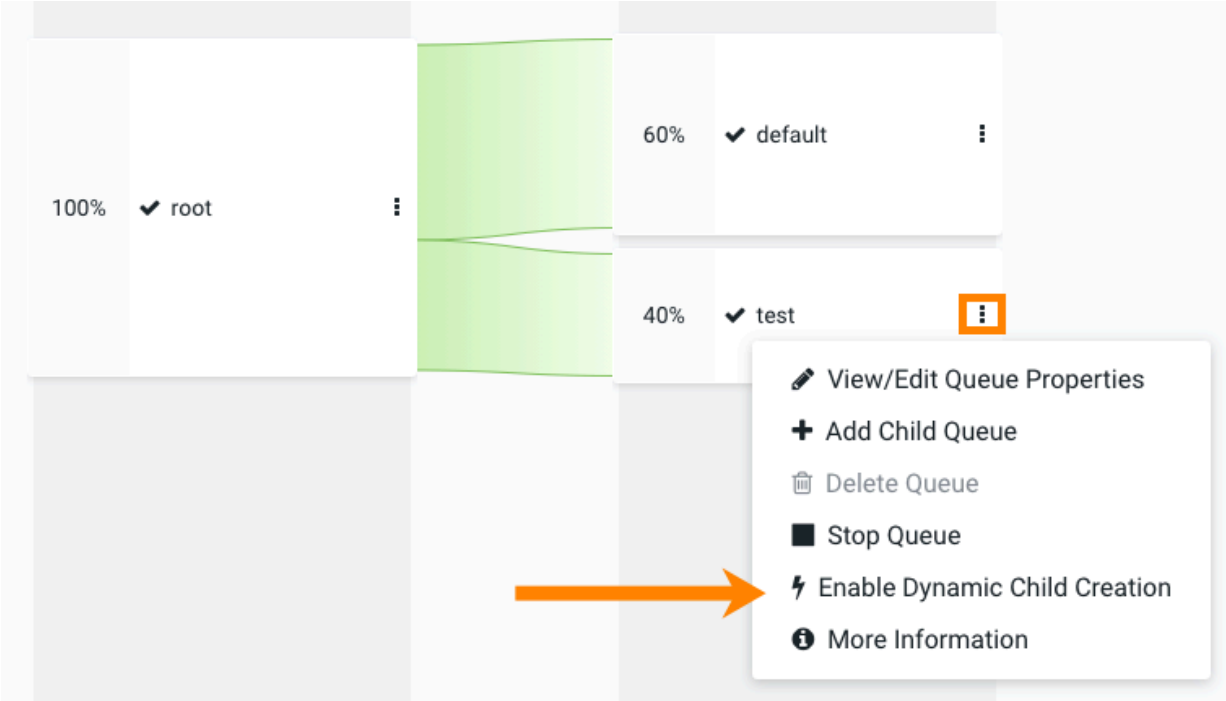

4. Set the Minimum and Maximum capacities that will be applied for every dynamic child queue under that particular parent queue.

   Save the minimum and maximum capacity for the queue with the default values that copy the parent queue, or change them using the Auto Queue Creation template (Technical Preview) feature.

   Technical Preview: This is a technical preview feature and considered under development. Do not use this in your production systems. To share your feedback, contact Support by logging a case on our Cloudera Support Portal. Technical preview features are not guaranteed troubleshooting guidance and fixes.
5. Click Save.

What to do next

To define a placement rule that could lead to dynamically created child queues, ensure that during placement rule creation you select the Create the target queue if it does not exist? option and provide a Managed Parent Queue as the parent queue. For more information, see *Manage placement rules*.

Related Information

Managing placement rules

Enabling dynamic child creation in weight mode

In the weight resource allocation mode, when you enable dynamic child creation for a queue, it becomes a parent queue that can have both static and dynamic child queues. You can enable that feature through the YARN Queue Manager UI.

About this task

In the weight mode, there is no Managed Parent Queue. When you enable the dynamic child creation feature for a queue, it becomes a parent queue that can have both static and dynamic child queues simultaneously. If this feature is not enabled, the queue can only have static child queues.

In contrast to Manage Parent Queue where the dynamic queue nesting level is limited to one, in weight mode, auto dynamic child creation allows you to create 2-level dynamic queues.

By default dynamic child queues are deleted automatically 5 minutes after the last job finished on them. You can disable this feature using the Capacity Scheduler Auto Queue Deletion YARN property. For more information, see *Disabling Auto Queue Deletion*.

Procedure

1. In Cloudera Manager, select the YARN Queue Manager UI.
   A graphical queue hierarchy is displayed in the Overview tab.
2. Find the queue for which you want to enable the auto dynamic child creation feature.
3. Select the More Options menu and select Enable Dynamic Child Creation.

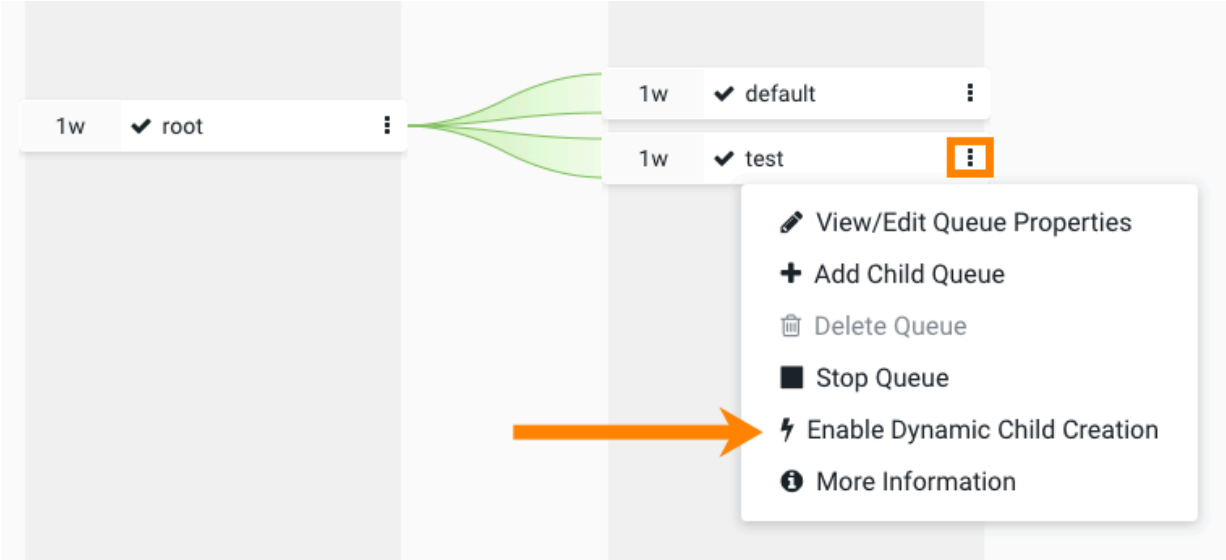

   Dynamic Child Queue Capacities window is displayed.
4. Set the Minimum and Maximum capacities that will be applied for every dynamic child queue under that particular parent queue.

   Save the minimum and maximum capacity with the default values that copy the parent queue, or change them using the Auto Queue Creation template (Technical Preview) feature.

   Technical Preview: This is a technical preview feature and considered under development. Do not use this in your production systems. To share your feedback, contact Support by logging a case on our Cloudera Support Portal. Technical preview features are not guaranteed troubleshooting guidance and fixes.

Results

Dynamic child creation is enabled for the queue and a bolt icon is visible next to the queue's name.

What to do next

If you want to define a placement rule that could lead to dynamically created child queues, ensure that during placement rule creation you check the Create the target queue if it does not exist? property and provide a parent queue for which dynamic child creation is enabled. For more information, see *Manage placement rules*.

Related Information

Managing placement rules
Placement rule policies
Disabling auto queue deletion globally

Disabling dynamic child creation in weight mode

In the weight resource allocation a dynamic child creation enabled parent queue can have both static and dynamic child queues. If you disable the dynamic child creation feature for a parent queue its dynamic child queues are deleted, and it can have only static queues.

Before you begin

Ensure that no placement rule uses the dynamic child creation enabled parent queue for which you want to disable the dynamic child creation feature. The YARN Queue Manager UI will not let you disable the feature for a parent queue that is used by a placement rule.

Procedure

1. In Cloudera Manager navigate to Clusters YARN Queue Manager UI
2. Find the dynamic child creation enabled queue for which you want to disable the dynamic child creation feature.
3. Click the three vertical dots next to the queue name.
4. Select Disable Dynamic Child Creation.
5. Click Yes to confirm to disable.

   On the YARN Queue Manager UI, and also on the YARN Web UI, the dynamic child queues of the selected parent queue are removed. However, if there were any running applications on such child queues they will continue running in their original queue.
6. Restart the YARN service.

Results

All dynamically created child queues of the selected parent queue are deleted. After the YARN service restart placement rules are reevaluated and applications that were running on the deleted dynamic child queues are either moved to a different queue or are rejected according to the placement rules.

Managing dynamic child creation enabled parent queues

The YARN Queue Manager UI in Cloudera Manager provides an overview of your queue hierarchy where you can view and manage the parent queues for which dynamic child creation is enabled.

Procedure

1. In Cloudera Manager, select the YARN Queue Manager UI.
   A graphical queue hierarchy is displayed in the Overview tab. A bolt icon is displayed next to the queue name of parent queues for which dynamic child creation is enabled.

2. Select the three vertical dots and perform one of the following action for a dynamic child creation enabled parent queue:
   - Edit Queue Properties: In relative and absolute resource allocation modes, you can view and edit the Dynamic Auto-Creation of Queue section of the Queue Properties. In weight resource allocation mode, you can only view that section.
   - Add Child Queue: Supported only in weight resource allocation mode where a dynamic parent queue can have both static and dynamic child queues simultaneously.
   - Delete Queue: You have to stop the queue first before you can delete it.

     Note: You cannot delete the root and the root.default queues.

   - Stop Queue / Stop Queue and Its Children: Stop the queue and its child queues, if there is any.

     Note: In weight resource allocation mode, if the queue has dynamically created child queues, you cannot restart the dynamic child queues once you have stopped them.

   - Edit Dynamic Child Queue Capacities: Edit the capacities of dynamic child queues created under the parent queue.
   - More Information: You are redirected to the Resource Manager UIv2 Queues page.

Managing dynamically created child queues

YARN Queue Manager UI provides an overview of your queue hierarchy where you can view and manage your dynamically created child queues.

About this task

You cannot directly delete dynamically created child queues. For more information about dynamic queue deletion, see *Delete dynamically created child queues*.

Procedure

1. In Cloudera Manager, select the YARN Queue Manager UI.

   A graphical queue hierarchy is displayed in the Overview tab. A leaf is displayed next to the queue name of parent queues for which dynamic child creation is enabled.
2. Select the More Options menu and perform one of the following actions for a dynamically created queue:
   - Edit Queue Properties: Available in relative and absolute resource allocation mode.
   - More Information: You are redirected to the Resource Manager UIv2 Queue page.

Related Information

Deleting dynamically created child queues

Deleting dynamically created child queues

Dynamically created child queues can be deleted automatically once they become inactive, or manually with some limitations.

Automatic child queue deletion

The automatic queue deletion feature is enabled by default. As a result, dynamically created child queues are deleted 300 seconds after their last job is finished. You can disable this feature globally for all dynamic child creation enabled parent queues or for specific parent queues. Additionally, configuring the queue removal expiration time is also possible.

Note: This feature is called auto queue deletion on the global level and queue auto removal on the queue level.

Manual child queue deletion

Unlike parent queues, dynamically created child queues cannot be deleted manually. No action or button exists that deletes a specific dynamically created child queue. Instead, you can delete them by either restarting the YARN service or deleting the parent queue.

Disabling auto queue deletion globally

Dynamically created child queues are automatically deleted after they expire. If you want to keep all dynamically created child queues running after their last job is finished, you can disable automatic queue deletion on a global level.

About this task

By default automatic queue removal is enabled on two levels: globally and on the level of dynamic child creation enabled parent queues. The following steps describe how to disable the auto queue deletion feature on a global level, that is, dynamically created child queues are not deleted automatically once they become inactive.

Note: If you want to disable the automatic queue removal for specific dynamic child creation enabled parent queues, use the Queue Auto Removal (YARN restart required) Queue Manager property. For more information, see Disabling queue auto removal on a queue level.

Procedure

1. In Cloudera Manager, navigate to YARN Configuration .
2. Search for queue deletion.
3. Find and clear the Capacity Scheduler Auto Queue Deletion property to disable the auto queue deletion feature.
4. Click Save.
5. Restart the YARN service.

Results

Auto queue deletion is disabled. Dynamically created queues do not get deleted once they became inactive.

Related Information

Disabling queue auto removal on a queue level

Disabling queue auto removal on a queue level

Dynamically created child queues are automatically deleted after they expire. If you want to keep dynamically created child queues of a specific parent queue running after their last job is finished, you can disable the queue auto removal feature for that parent queue.

About this task

By default automatic queue removal is enabled on two levels - global level and on the level of dynamic child creation enabled parent queues. The following steps describe how to disable the queue auto removal feature for a dynamic child creation enabled parent queue; meaning, its dynamically created child queues are not deleted automatically once they become inactive.

Note: If you want to disable the automatic queue removal for all dynamically created child queues, you can use the global Capacity Scheduler Auto Queue Deletion YARN property. For more information, see *Disabling auto queue deletion globally*.

Before you begin

Ensure that the global level auto queue deletion feature is enabled. The feature is enabled if the YARN Configuration Capacity Scheduler Auto Queue Deletion checkbox is selected.

Procedure

1. In Cloudera Manager navigate to Clusters YARN Queue Manager UI .
   The Overview tab is opened, displaying a hierarchical view of the queues.
2. Disable the queue auto removal feature for a specific queue.

   Repeat the following substeps for all applicable dynamic child creation enabled queues:

   a) Find the queue for which you want to disable the queue auto removal feature, and click the three vertical dots ⋮ next to its name.
   b) Select Edit Queue Properties.
   c) In the Dynamic Auto-Creation of Queue section find and clear the Queue Auto Removal (YARN restart required) checkbox.
   d) Click Save.

Results

Auto queue deletion is enabled on the global level and disable for the selected dynamic child creation enabled queues.

Related Information

Disabling auto queue deletion globally

Configuring the queue auto removal expiration time

The expiration time for automatic queue deletion is set to 300 seconds by default. You can change this value using the YARN Queue Manager UI.

Procedure

1. In Cloudera Manager navigate to Clusters YARN Queue Manager UI .
   The Overview tab is opened, displaying a hierarchical view of the queues.
2. Click Configuration.
3. Find the Queue Auto Removal Expiration Time property and set the expiration time in seconds.

   The default expiration time is 300, meaning that if auto queue deletion is enabled for a queue, its dynamically created child queues are deleted 300 seconds after the last job finished on them.
4. Click Save.
5. Restart the YARN service.

Deleting dynamically created child queues manually

Deleting dynamically created child queues manually is not supported by the YARN Queue Manager UI, but there are some workarounds to remove them. It can be useful, for example, when the applications in that queue are terminated.

There are two ways to remove dynamically created child queues manually:

- Restart the YARN service: That stops and deletes all dynamically created queues.
- Stop and then delete the parent queue of the dynamically created child queues: This will delete both the dynamic parent and all of its child queues - both static and dynamic ones.

Partition configuration

You can partition a cluster into sub-clusters so that jobs run on nodes with specific characteristics. You can configure these partitions, so that you run YARN applications on cluster nodes of the specified partition.

Note: The term *Partitions* is used instead of *Node Labels* to be consistent with the YARN terminology.

Partition can be specified as exclusive or non-exclusive/shareable

- exclusive - Access is restricted to applications running in queues associated with the partition.
- non-exclusive - If idle capacity is available on the partition, resources are shared with all applications in the cluster.

The fundamental unit of scheduling in YARN is the queue. The capacity of each queue specifies the percentage of cluster resources that are available for applications submitted to the queue. Queues can be set up in a hierarchy that reflects the resource requirements and access restrictions required by the various organizations, groups, and users that utilize cluster resources.

Using partition, you can divide a cluster into sub-clusters so that jobs can be run on partitions with specific characteristics. For example, you can use a partition to run memory-intensive jobs only on nodes with a larger amount of RAM. Partitions can be assigned to cluster nodes, and specified as exclusive or non-exclusive. You can then associate partition with capacity scheduler queues. Each node can be associated with only one partition.

Partition Type: Exclusive

When a queue is associated with one or more exclusive partitions, all applications submitted by the queue have exclusive access to the nodes in those partitions.

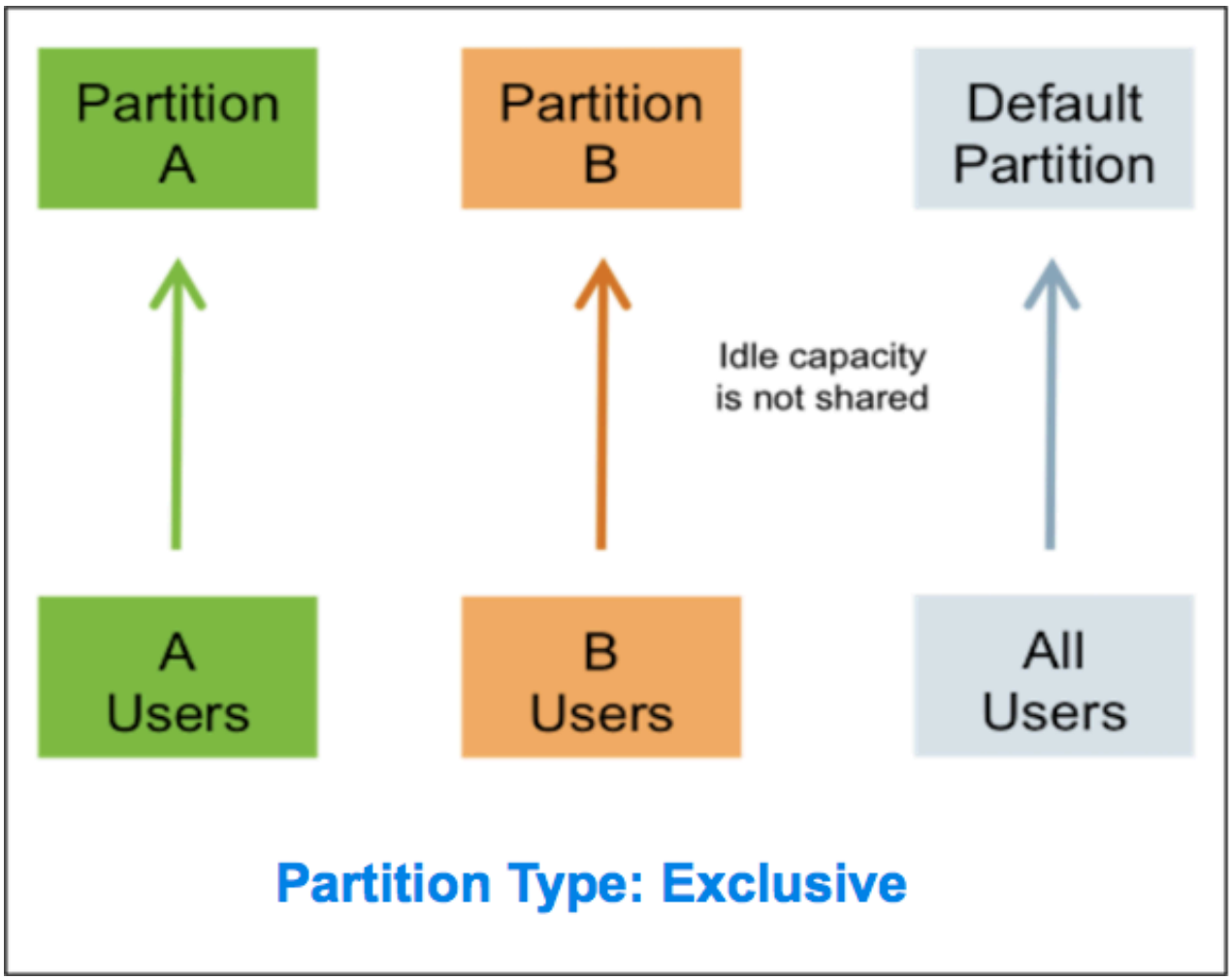

Partition Type: Non-exclusive

When a queue is associated with one or more non-exclusive partitions, all applications submitted by the queue get first priority on nodes in those partitions. If idle capacity is available on those partition nodes, resources are shared with other applications in the cluster. Non-labeled applications are preempted if labeled applications request new resources on the labeled nodes.

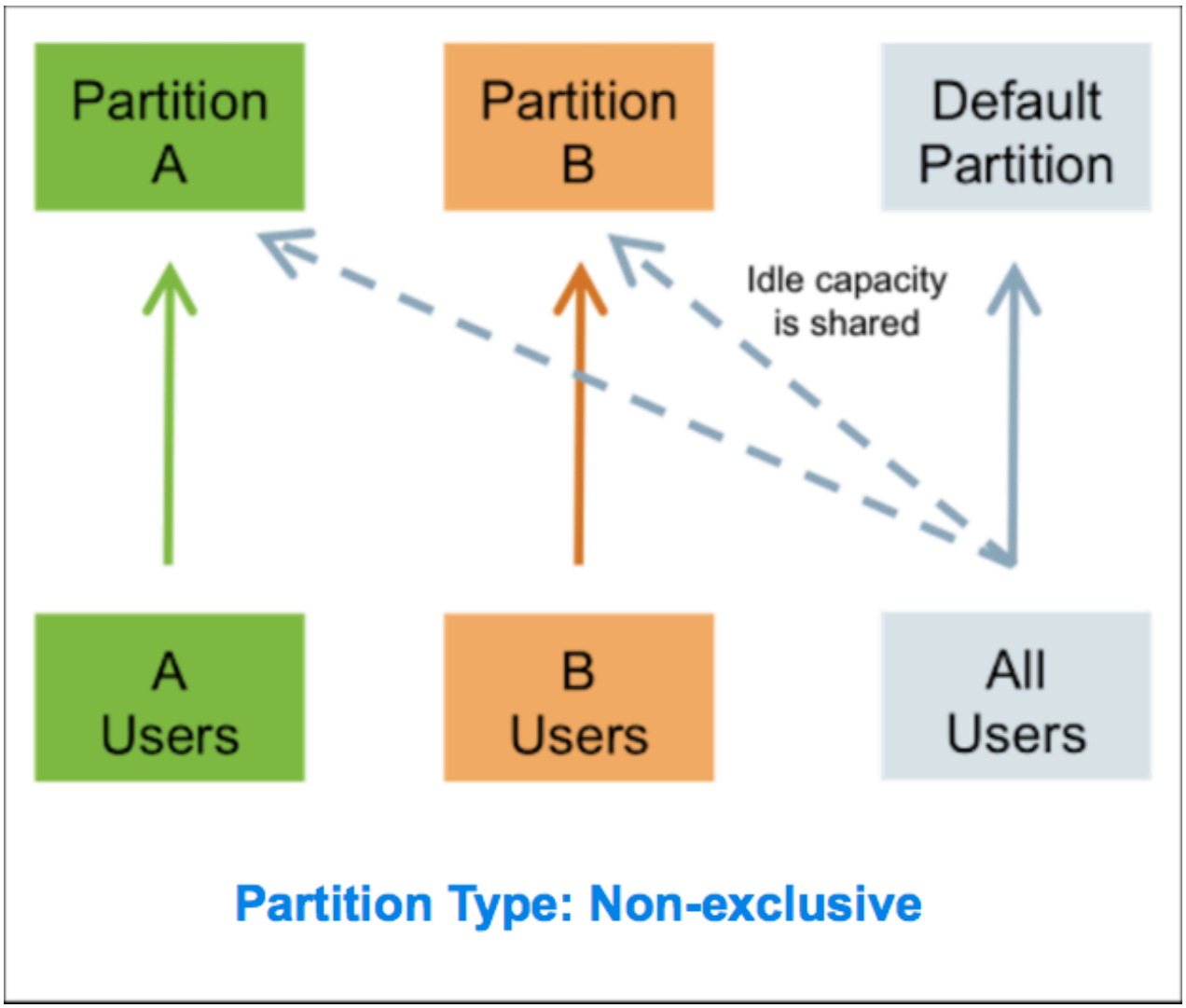

Queues without associate partition

If no partition is assigned to a queue, the applications submitted by the queue can run on any node without a partition, and on nodes with non-exclusive partitions if idle resources are available.

Preemption

Labeled applications that request labeled resources preempt non-labeled applications on labeled nodes. If a labeled resource is not explicitly requested, the normal rules of preemption apply. Non-labeled applications cannot preempt labeled applications running on labeled nodes.

Enabling node labels on a cluster to configure partition

You can configure partitions on a cluster by making configuration changes on the YARN ResourceManager host.

Procedure

1. Create a Label Directory in HDFS.
   a) Use the following commands to create a node-labels directory in which to store the Node Labels in HDFS:

```
sudo su hdfs
```

```
hadoop fs -mkdir -p /yarn/node-labels
hadoop fs -chown -R yarn:yarn /yarn
hadoop fs -chmod -R 700 /yarn
```

Where -chmod -R 700 specifies that only the yarn user can access the node-labels directory.

b) Use the following command to confirm that the directory was created in HDFS:

```
hadoop fs -ls /yarn
```

The new node-labels directory should appear. The owner should be yarn, and the permission should be drwx:

```
Found 1 items
drwx------ - yarn yarn 0 2014-11-24 13:09 /yarn/node-labels
```

2. In Cloudera Manager, navigate to YARN Configuration .
3. Search for node label.
4. Find the Node Labels property and ensure that it is selected (enabled).
5. Find the Node Labels Directory property and set its value to reference the HDFS node label directory.

   For example: hdfs://node-1.example.com:8020/yarn/node-labels/.

   HDFS paths are automatically translated, however, any other file system requires a full path to the Node Labels Directory.

   The default value of the Node Labels Directory property is /yarn/node-labels.
6. Start or Restart the YARN ResourceManager.

Creating partitions

You must first create partitions to assign them to nodes and associate it with queues.

Before you begin

You must enable node labels on a cluster before you create a partition. For more information, see Enable node labels on a cluster.

Procedure

1. In Cloudera Manager, navigate to Clusters YARN Queue Manager UI service.
   A graphical queue hierarchy is displayed in the Overview tab.
2. Click Partitions.
3. Click +Create.
   The Create Partition dialog box is displayed.
4. Add a name for the partition in Partition Name.
5. Select Exclusive or Non-Exclusive node label type under Partition Type.

   For information about Exclusive or Non-Exclusive partition type, see Configure Partitions.
6. Select one or more unassigned nodes listed under Unassigned Nodes and click the < arrow to move it under Assigned Nodes to assign it to the partition. You can also search or filter the nodes using the regular expression.
7. Click Save.

Assigning or unassigning a node to a partition

You can assign or unassign a node to an existing partition.

Procedure

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Partition tab. A list of partitions is displayed.
3. Click the Edit icon at the right of a partition.
4. Assign or unassign the nodes in the Edit Partition dialog box.
   a. Assign Nodes: Select an unassigned node listed under Unassigned Nodes and click the < arrow button to move it to under Assigned Nodes.
   b. Unassign Nodes: Select an assigned node listed under Assigned Nodes and click the > arrow button to move it to under Unassigned Nodes.
5. Click Save.

Viewing partitions

You can view the list of available partitions in the cluster. For each partition, it lists the number of associated nodes under the Hosts column along with the partition type and capacity.

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Partitions tab. A list of existing partitions is displayed.
3. Optionally, you can click on the number listed in the Host column to view the associated nodes.

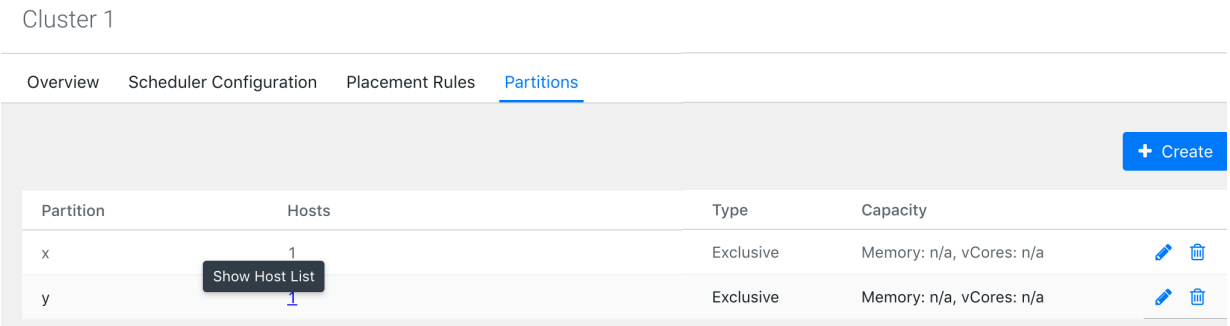

View Node Label Assignments

You can use the following commands to view information about partitions.

- List all running nodes in the cluster: yarn node -list

  Example:

```
[root@node-1 /]# yarn node -list
14/11/21 12:14:06 INFO impl.TimelineClientImpl: Timeline service address:
 http://node-1.example.com:8188/ws/v1/timeline/
14/11/21 12:14:07 INFO client.RMProxy: Connecting to ResourceManager at no
de-1.example.com/240.0.0.10:8032
Total Nodes:3
 Node-Id Node-State Node-Http-Address Number-of-Running-Containers
node-3.example.com:45454 RUNNING node-3.example.com:50060 0
node-1.example.com:45454 RUNNING node-1.example.com:50060 0
node-2.example.com:45454 RUNNING node-2.example.com:50060 0
```

- List the status of a node (includes partition): yarn node -status <Node_ID>

  Example:

```
[root@node-1 /]# yarn node -status node-1.example.com:45454
14/11/21 06:32:35 INFO impl.TimelineClientImpl: Timeline service address:
 http://node-1.example.com:8188/ws/v1/timeline/
14/11/21 06:32:35 INFO client.RMProxy: Connecting to ResourceManager at
 node-1.example.com/240.0.0.10:8032
Node Report :
 Node-Id : node-1.example.com:45454
 Rack : /default-rack
 Node-State : RUNNING
 Node-Http-Address : node-1.example.com:50060
 Last-Health-Update : Fri 21/Nov/14 06:32:09:473PST
 Health-Report :
 Containers : 0
 Memory-Used : 0MB
 Memory-Capacity : 1408MB
 CPU-Used : 0 vcores
 CPU-Capacity : 8 vcores
 Node-Labels : x
```

Partitions are also displayed in the ResourceManager UI on the Nodes and Scheduler pages.

Associating partitions with queues

You can use partitions to run YARN applications on cluster nodes that have a specified partition.

Before you begin

Before associating partitions, you must create partitions and assign partitions to cluster nodes. For more information about creating partitions see, Creating partitions on page 95.

Note: After you associate a partition with one or more queues, in the YARN Queue Manager UI, click Overview > <*PARTITION NAME*> from the drop-down list and distribute capacity to the queues before switching allocation mode or creating placement rules.

Note: The term *Partitions* is used instead of *Node Labels* to be consistent with the YARN terminology.

About this task

Use Queue Manager to create and assign partitions to cluster nodes, associate the partitions (yarn.scheduler.capacity.<queue-path>.accessible-node-labels) to queues and configure capacity to that queue for the specified partition. Queue Manager evenly distributes the available capacity among all the queues in the partition. You can manually modify the capacity on each partition of each queue, and also ensure that the sum of capacities of each partition of direct children of a parent queue at every level is equal to 100%. partitions that a queue can access (accessible partitions of a queue) must be the same as, or a subset of, the accessible partitions of its parent queue.

Example

Assume that a cluster has a total of 8 nodes. The first 3 nodes (n1-n3) have partition = x, the next 3 nodes (n4-n6) have partition = y, and the final 2 nodes (n7, n8) do not have any partitions. Each node can run 10 containers.

The queue hierarchy is as follows:

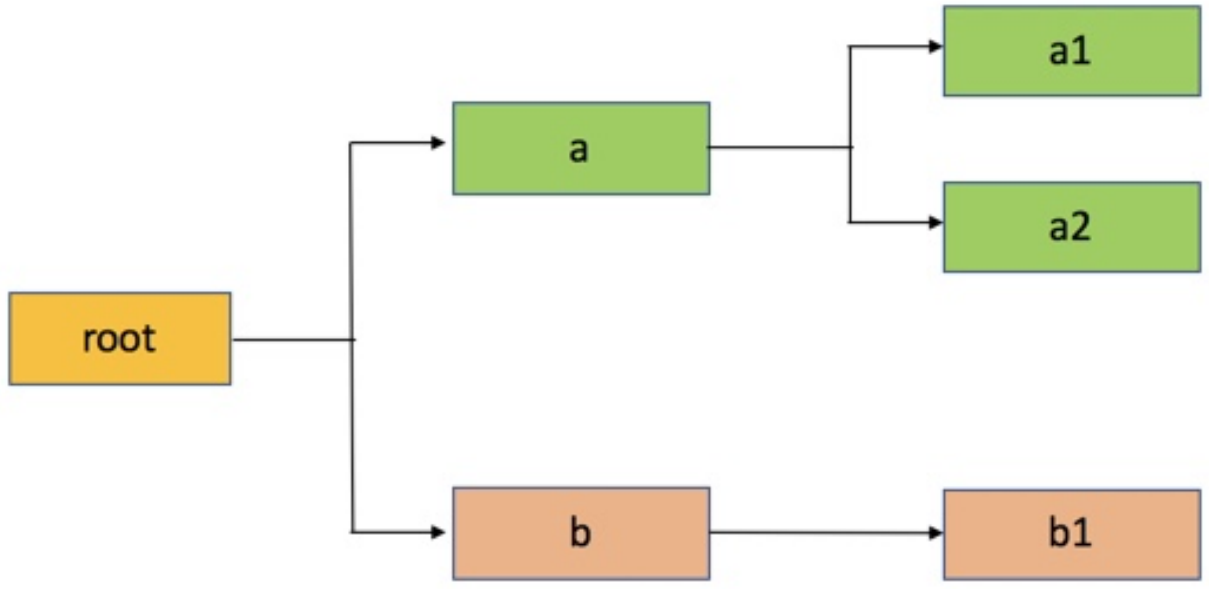

Assume that queue “a” can access partitions “x” and “y”, and queue “b” can only access partition “y”. By definition, nodes without labels can be accessed by all queues.

Consider the following example label configuration for the queues, in the Relative resource allocation mode:

capacity(a) = 40, capacity(a, label=x) = 100, capacity(a, label=y) = 50; capacity(b) = 60, capacity(b, label=y) = 50

This means that:

- Queue “a” can access 40% of the resources on nodes without any labels, 100% of the resources on nodes with label=x, and 50% of the resources on nodes with label=y.
- Queue “b” can access 60% of the resources on nodes without any labels, and 50% of the resources on nodes with label=y.

You can also see that for this configuration:

capacity(a) + capacity(b) = 100

capacity(a, label=x) + capacity(b, label=x) (b cannot access label=x, it is 0) = 100

capacity(a, label=y) + capacity(b, label=y) = 100

For child queues under the same parent queue, the sum of the capacity for each label should equal 100%.

Similarly, you can set the capacities of the child queues a1, a2, and b1:

a1 and a2: capacity(a.a1) = 40, capacity(a.a1, label=x) =30, capacity(a.a1, label=y) =50 capacity(a.a2) = 60, capacity(a.a2, label=x) =70, capacity(a.a2, label=y) =50;

b1: capacity(b.b1) = 100, capacity(b.b1, label=y) = 100

You can see that for the a1 and a2 configuration:

capacity(a.a1) + capacity(a.a2) = 100

capacity(a.a1, label=x) + capacity(a.a2, label=x) = 100

capacity(a.a1, label=y) + capacity(a.a2, label=y) = 100

How many resources can queue a1 access?

Resources on nodes without any labels: Resource = 20 (total containers that can be allocated on nodes without a label, in this case n7, n8) * 40% (a.capacity) * 40% (a.a1.capacity) = 3.2 (containers)

Resources on nodes with label=x

Resource = 30 (total containers that can be allocated on nodes with label=x, in this case n1-n3) * 100% (a.labelx.capacity) * 30% = 9 (containers)

To implement this example configuration, perform the following

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on a queue and select the View/Edit Queue Properties option.
3. In the Queue Properties dialog-box , select the x label from the Accessible Partitions drop-down box, click +, again select the y label from the Accessible Partitions drop-down box, and click Save.

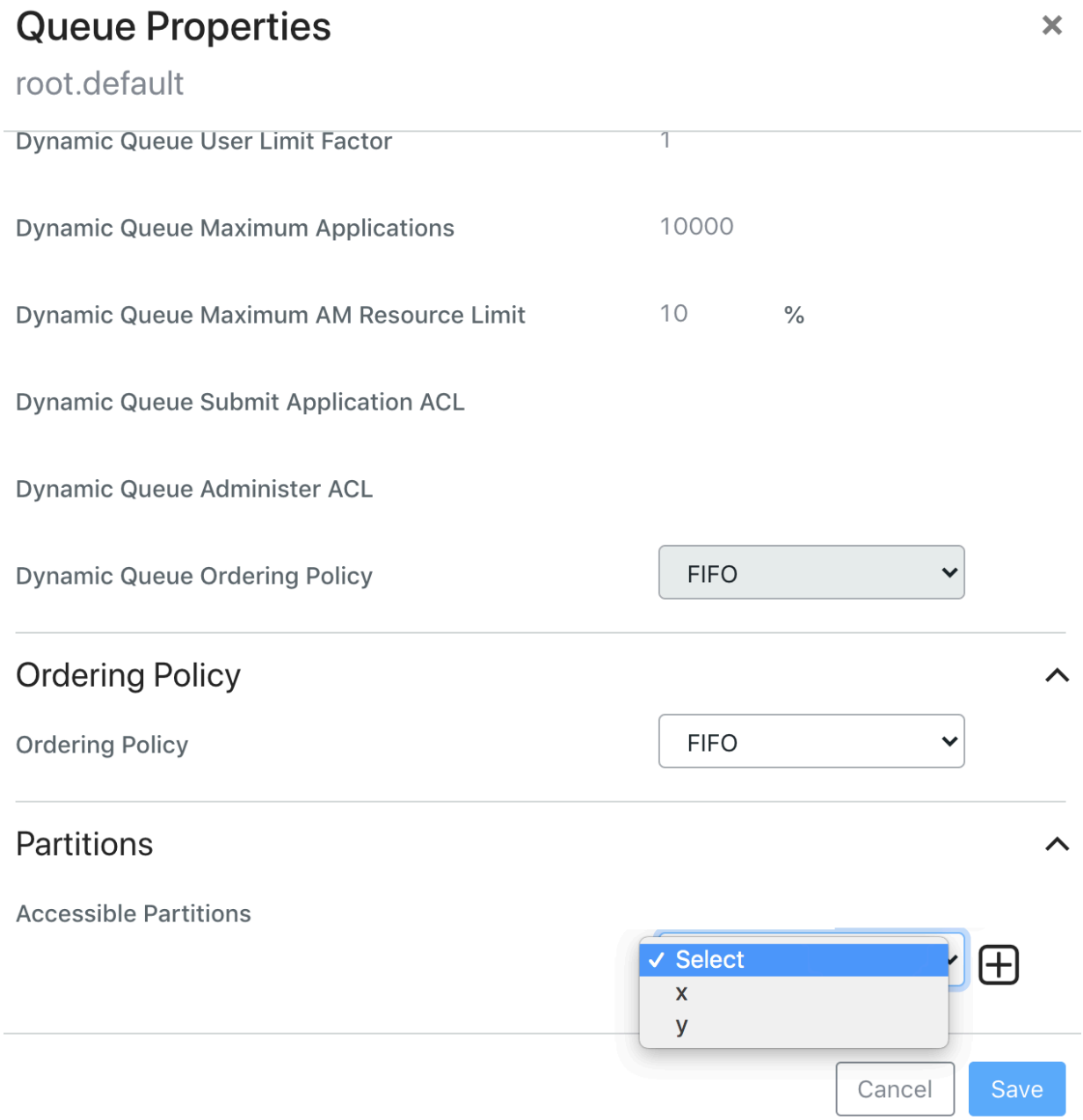

4. Repeat the above steps to assign x label for a1 and a2 queues.
5. Click on the three vertical dots on b queue and select the Edit Queue Properties option.
6. In the Queue Properties dialog-box, select the y label from the Accessible Partitions drop-down box, click +, and click Save.

7. Repeat the above steps to assign y label for b1, a, a1, and a2 queue.

   Queue Manager automatically distributes the available capacity among all the queues in the partition. If you want to modify the capacity of the queues, click on the Partition drop-down box in the Overview tab, select the label and modify the queue capacity.
8. In the Overview tab, click on the Partition drop-down box and select label y.
9. Click on the three vertical dots on the a queue and select the Edit Child Queues option.
10. Enter the Configured Capacity of a1 to 50 and a2 to 50 and Click Save.
11. Click on the three vertical dots on the b queue and select the Edit Child Queues option.
12. Enter the Configured Capacity of b1 to 100 and click Save.
13. Click on the three vertical dots on the root queue and select the Edit Child Queues option.
14. Enter the Configured Capacity of a to 50 and b to 50 and click Save.

Disassociating partitions from queues

You can disassociate a partition from the queue. You should disassociate a partition before you delete the queue. Before disassociating a partition from a queue, you should remove the partition capacity for that queue by setting it to zero.

Procedure

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on a queue and select the View/Edit Queue Properties option.

3. In the Queue Properties dialog box, from the Accessible Partitions, click X next to the name of the partition.

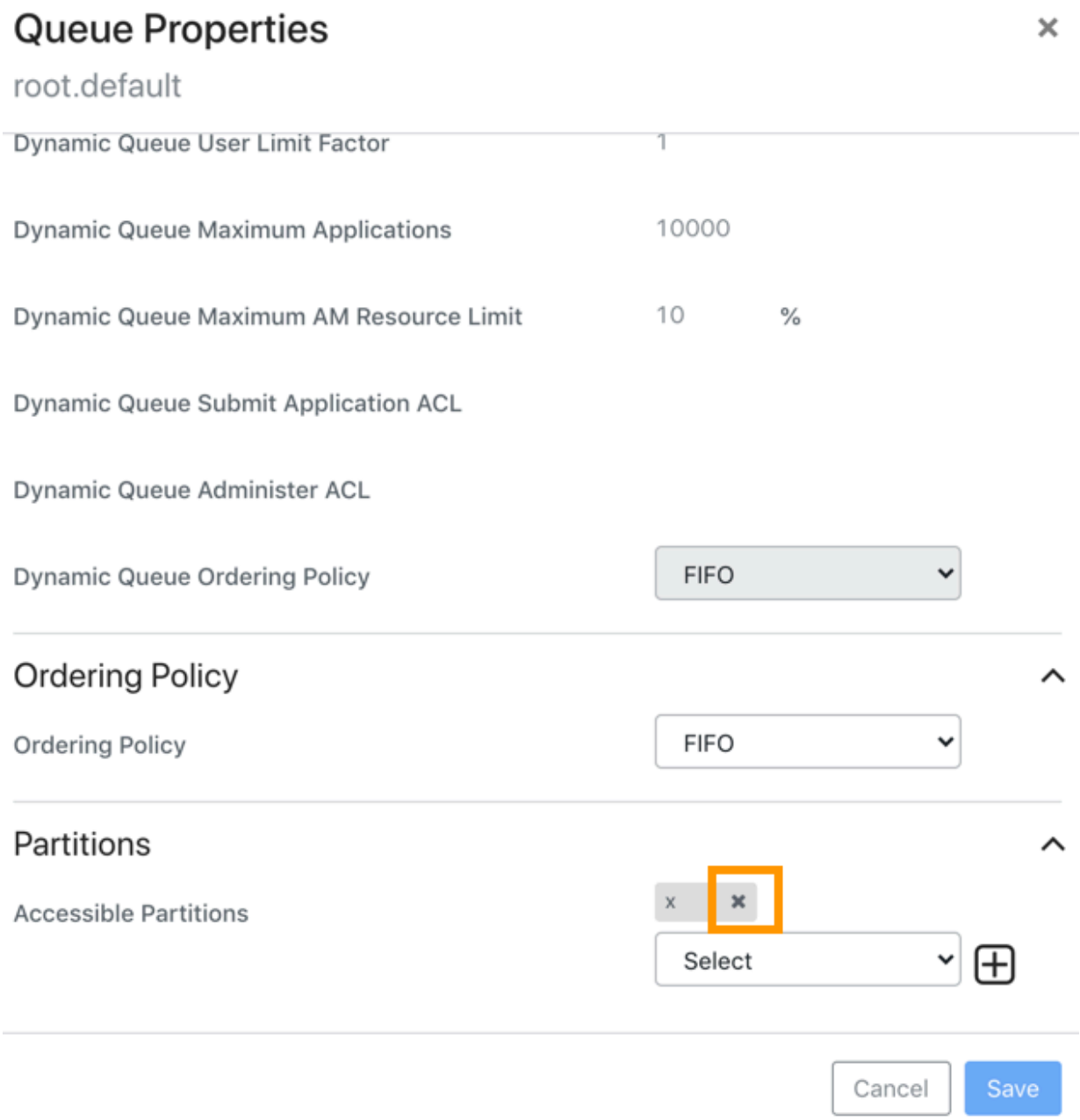

4. Click Save.

Deleting partitions

Due to a known issue, COMPX-5106, it is not recommended to delete a partition if it is associated with queues and the queues have capacities configured for that partition.

Before you begin

You must have created a partition in the cluster using YARN /add-node-labels API, and assigned this partition to some queues and set the partition capacities to queues.

About this task

Use Queue Manager to create a partition in the cluster using YARN /add-node-labels API and assign this partition to some queues and set the partition capacities to queues as follows:

1. Create the partition demo
2. Assign the partition to queues:
   - yarn.scheduler.capacity.root.accessible-node-labels=”*”
   - yarn.scheduler.capacity.root.default.accessible-node-labels=”demo”
3. Set the partition capacities to queues:
   - yarn.scheduler.capacity.root.accessible-node-labels.demo.capacity=100
   - yarn.scheduler.capacity.root.accessible-node-labels.demo.maximum-capacity=100
   - yarn.scheduler.capacity.root.default.accessible-node-labels.demo.capacity=100
   - yarn.scheduler.capacity.root.default.accessible-node-labels.demo.maximum-capacity=100

Now, to delete the partition and its related configs that are created above perform the following:

1. Unassign node labels from the queues set in step-2 above:
   - yarn.scheduler.capacity.root.accessible-node-labels=”*”
   - yarn.scheduler.capacity.root.default.accessible-node-labels=””
2. Delete the partition created in step-1 using the YARN /remove-node-labels API.

Now the system is left with residue configs that was set in step-3. And there is no way to remove these and causes various issues when trying to remove the residue configs.

Solution to remove the configs

YARN’s mutation API is designed to be simple and very generic, it only handles key-value pairs. The unwanted capacities can be removed by using an existing mutation API in YARN. This is performed by setting values to an empty (“”) string (or null) and then those configs will be removed from the in-memory config.

For example:

```
{
    "update-queue":
    [
        {
            "queue-name": "root",
            "params":
            {
                "entry":
                [
                    {
                        "key": "accessible-node-labels.label1.maximum-capac
ity",
                        "value": ""
                    },
                    {
                        "key": "accessible-node-labels",
                        "value": "*"
                    },
                    {
                        "key": "accessible-node-labels.label1.capacity",
                        "value": ""
                    }
                ]
            }
        },
{
            "queue-name": "root.a",
            "params":
```

```
            {
                "entry":
                [
                    {
                        "key": "accessible-node-labels.label1.maximum-cap
acity",
                        "value": ""
                    },
                    {
                        "key": "accessible-node-labels",
                        "value": ""
                    },
                    {
                        "key": "accessible-node-labels.label1.capacity",
                        "value": ""
                    }
                ]
            }
        },
        {
            "queue-name": "root.a.a1",
            "params":
            {
                "entry":
                [
                    {
                        "key": "accessible-node-labels.label1.maximum-cap
acity",
                        "value": ""
                    },
                    {
                        "key": "accessible-node-labels",
                        "value": ""
                    },
                    {
                        "key": "accessible-node-labels.label1.capacity",
                        "value": ""
                    }
                ]
            }
        },
        {
            "queue-name": "root.a.a2",
            "params":
            {
"entry":
                [
                    {
                        "key": "accessible-node-labels.label1.maximum-cap
acity",
                        "value": ""
                    },
                    {
                        "key": "accessible-node-labels",
                        "value": ""
                    },
                    {
                        "key": "accessible-node-labels.label1.capacity",
                        "value": ""
                    }
                ]
            }
        }
    ],
```

```
    "global": null,
    "global-updates": null
}
```

```
{
    "key": "accessible-node-labels.label1.capacity",
    "value": ""
}
```

and

```
{
    "key": "accessible-node-labels.label1.maximum-capacity",
    "value": ""
}
```

As seen in the above JSON, add null or empty string values for all the queues where you want to clear values from the config.

Setting a default partition expression

You can set a default partition on a queue. The default partition is used if no partition is specified when the application is submitted to the queue. By default, this field is empty, so applications get containers from nodes without a partition or label.

Procedure

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select the View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, select the default partition from the Default Partition Expression drop-down list.
4. Click Save.

Using partitions when submitting a job

You can use various methods to specify partitions when submitting jobs.

Procedure

- Set partitions when Submitting Jobs

  You can use the following methods to specify partitions when submitting jobs:

  - ApplicationSubmissionContext.setNodeLabelExpression(<node_label_expression>) - sets the partition expression for all containers of the application.
  - ResourceRequest.setNodeLabelExpression(<node_label_expression>) - sets the partition expression for individual resource requests. This overrides the partitions expression set in ApplicationSubmissionContext.set NodeLabelExpression(<node_label_expression>).
  - Specify setAMContainerResourceRequest.setNodeLabelExpression in ApplicationSubmissionContext to indicate the expected partition for the ApplicationMaster container.

  You can use one of these methods to specify a partition expression, and -queue to specify a queue, when you submit YARN jobs using the distributed shell client. If the queue has a label that satisfies the label expression, it will run the job on the partition(s). If the label expression does not reference a label associated with the specified

queue, the job does not run and an error is returned. If no partitionn is specified, the job runs only on nodes without a partition, and on nodes with non-exclusive partitions if idle resources are available.

Note:

You can only specify one partition in the .setNodeLabelExpression methods.

For example, the following commands run a simple YARN distributed shell "sleep for a long time" job. In this example you are asking for more containers than the cluster can run so you can see which node the job runs on. We are specifying that the job should run on queue "a1", which our user has permission to run jobs on. We are also using the -node_label_expression parameter to specify that the job will run on all nodes with label "x".

```
sudo su yarn
hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-yarn/hadoop-yarn-applic
ations-distributedshell.jar
 -shell_command "sleep 100" -jar /opt/cloudera/parcels/CDH/lib/hadoop-ya
rn/hadoop-yarn-applications-distributedshell.jar
 -num_containers 30 -queue a1 -node_label_expression x
```

If you run this job on the example cluster we configured previously, containers are allocated on node-1, as this node has been assigned partition "x", and queue "a1" also has partition "x":

The following commands run the same job that you specified for partition "x", but this time you will specify queue "b1" rather than queue "a1".

```
sudo su yarn
hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-yarn/hadoop-yarn-applic
ations-distributedshell.jar
 -shell_command "sleep 100000" -jar /opt/cloudera/parcels/CDH/lib/hadoop-
yarn/hadoop-yarn-applications-distributedshell.jar
 -num_containers 30 -queue b1 -node_label_expression x
```

When you attempt to run this job on our example cluster, the job will fail with the following error message because label "x" is not associated with queue "b1".

```
14/11/24 13:42:21 INFO distributedshell.Client: Submitting application to
 ASM
14/11/24 13:42:21 FATAL distributedshell.Client: Error running Client
org.apache.hadoop.yarn.exceptions.InvalidResourceRequestException: Invalid
 resource request, queue=b1 doesn't
have permission to access all labels in resource request. labelExpression
 of resource request=x. Queue labels=y
```

- MapReduce Jobs and Partitions

  Currently you cannot specify a partition when submitting a MapReduce job. However, if you submit a MapReduce job to a queue that has a default partition expression, the default partition is applied to the MapReduce job.

  Using default partition expressions tends to constrain larger portions of the cluster, which at some point starts to become counter-productive for jobs - such as MapReduce jobs - that benefit from the advantages offered by distributed parallel processing.

Cloudera Runtime 7.3.1

Managing Apache Hadoop YARN Services

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Configuring YARN Services API to manage long-running applications..... 4
MapReduce Job History Server..... 4

Configuring YARN Services using Cloudera Manager..... 5

Configuring node attribute for application master placement..... 5

Migrating database configuration to a new location..... 6

Running YARN Services.....9
Deploying and managing services on YARN.....10
Launching a YARN service..... 10
Saving a YARN service definition..... 11
Creating new YARN services using UI.....11
Creating a standard YARN service.....11
Creating a custom YARN service.....11
Managing the YARN service life cycle through the REST API.....12
YARN services API examples.....13

Configuring YARN Services API to manage long-running applications

You can use the YARN Services API to manage long-running YARN applications.

About this task

You can use the YARN Services API to deploy and manage the YARN services.

Procedure

1. Use the YARN Services API to run a POST operation on your application, specifying a long or unlimited lifetime in the POST attributes.
2. Use the YARN Services API to manage your application.
   - Increase or decrease the number of application instances.
   - Perform other application life cycle tasks.

Related Information

Configure YARN for long-running applications

Configure YARN for long-running applications

MapReduce Job History Server

The MapReduce Job History Server in Apache Hadoop is a web-based service that provides detailed information about the execution of past MapReduce jobs. This server is particularly useful for tracking, debugging, and monitoring historical job data after the job has completed.

Key Components:

The following section describes the key components of the MapReduce Job History Server:

- Job History Files:
  - The Job History Server retrieves completed job information from the job history logs, which are stored in a designated directory in HDFS.
  - These files include details such as job configuration, task-level metrics, counters, logs, and other metadata.
- Server UI
  - The Job History Server provides a web interface (often available on port 19888 by default) where users can view job-specific details.
  - The UI includes a dashboard with lists of completed jobs, allowing users to drill down into each job to view individual tasks, map/reduce counters, and logs.
- Job Summary and Details:
  - Each completed job in the UI shows a summary that includes, Job start and end times, Duration and the Final status (successful or failed)
  - Number of tasks, task types (map/reduce), and completion statistics
  - Detailed task views are available, including information on data read/write, task attempts, execution time, and error messages if applicable.
- Error Analysis and Logs:
  - The Job History Server enables users to review logs and error messages for failed tasks.
  - Logs help with troubleshooting and identifying performance bottlenecks.

- Counters and Metrics:
  - Job and task-level counters offer insights into resource usage, such as data processed, time spent, and custom counters defined in the job.

Configuring YARN Services using Cloudera Manager

You can enable and configure the YARN Services feature using Cloudera Manager.

About this task

YARN Services is enabled by default to ensure that any program that is dependent on it, for example Hive LLAP, can be installed. However you can disable it using Cloudera Manager.

Before you begin

If you want to actively use the YARN Services feature, Cloudera recommends to use Capacity Scheduler, which is the default scheduler, as only that scheduler type can fully support this feature.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Select the YARN Services Management filter.
4. Ensure that Enable YARN Services is checked.
5. Configure the YARN Services Dependencies Path to specify the path where the YARN services dependencies tarball is uploaded.

   Cloudera recommends using the default path:

   ```
   /user/yarn/services/service-framework/${cdhVersion}/service-dep.tar.gz
   ```

6. Click Save Changes.

If you changed the YARN Services Dependencies path, do the following:

7. Click the Actions button.
8. Select Install YARN Services Dependencies.
9. Confirm that you want to run the Install YARN Servioces Dependencies comment by clicking the Install YARN Services Dependencies button.
10. Once the command is run successfully, close the status window.
11. Click the *Stale Service Restart* icon that is next to the service to invoke the cluster restart wizard.
12. Click Restart Stale Services.
13. Select Re-deploy client configuration.
14. Click Restart Now.

Configuring node attribute for application master placement

Use the Node Attribute property to describe the attributes of a Node. The placement preference assigns nodes as worker nodes or compute nodes using the Node Attribute property. Application Master (AM) container is placed to run on worker nodes instead of compute nodes. The worker group is more stable because YARN ResourceManager and HDFS NameNode run in it. Also, the worker group nodes are less likely to be shut down due to autoscaling.

By default, a script runs to tag nodes as worker nodes or compute nodes using the node attribute mapping rule. AM containers are placed to run on worker nodes instead of compute nodes. You can disable the script to not make a placement preference for running AM containers.

1. In Cloudera Manager, click Clusters > YARN.
2. Go to the Configuration tab.
3. Search for *NM YARN-SITE.XML*.
4. Set the below property using advanced configuration snippet (safety-valve).

```
Name: yarn.nodemanager.node-attributes.provider
Value: config
```

5. Click Save.
6. Click Actions and select Restart to restart the service.

Note: To enable the node attribute placement preference again, follow the above steps and delete the `yarn.nodemanager.node-attributes.provider` and `config` configuration. Restart the YARN service after updating the configuration.

Migrating database configuration to a new location

The operations performed on queues in Queue Manager UI are stored as Queue Manager versions. You can either store these versions in the default database location on the host or configure a new location using Cloudera Manager UI.

By default, Queue Manager stores the operations performed on queues as versions in the config-service.mv.db database file on the host. Alternatively, you can configure a new database file in a new location and migrate the version information to the new file.

For security reasons, if you do not want to allow users to access the default database, you can move the database file to an alternative location. During an upgrade, you can move the database file to some other location and then restore this file to the default location after the upgrade.

1. (Optional) You can view the version information in the Queue Manager UI that needs to be migrated to a new location.

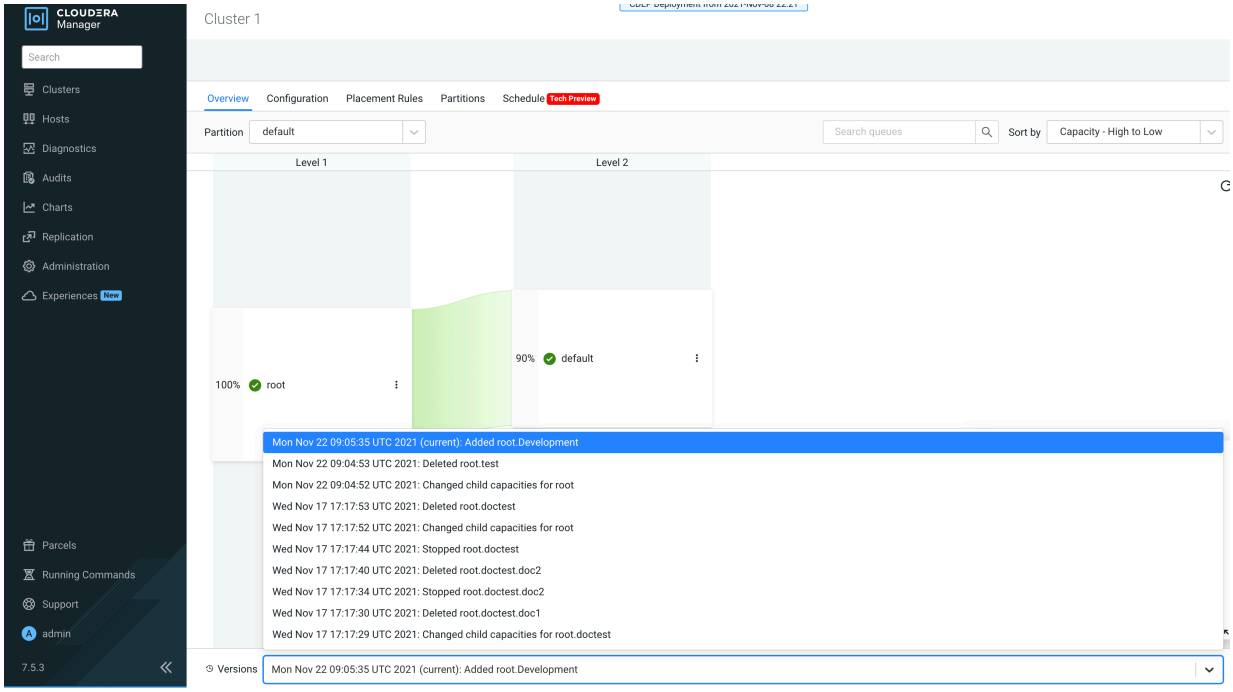

Note: The following steps explain how to change the database to a new location and copy the existing Queue Manager version information to the new database file. If you want to backup the currently used database file to another location and not migrate, perform only Steps 7, 8, 9 to backup the file.

2. In Cloudera Manager, click Clusters > queuemanager

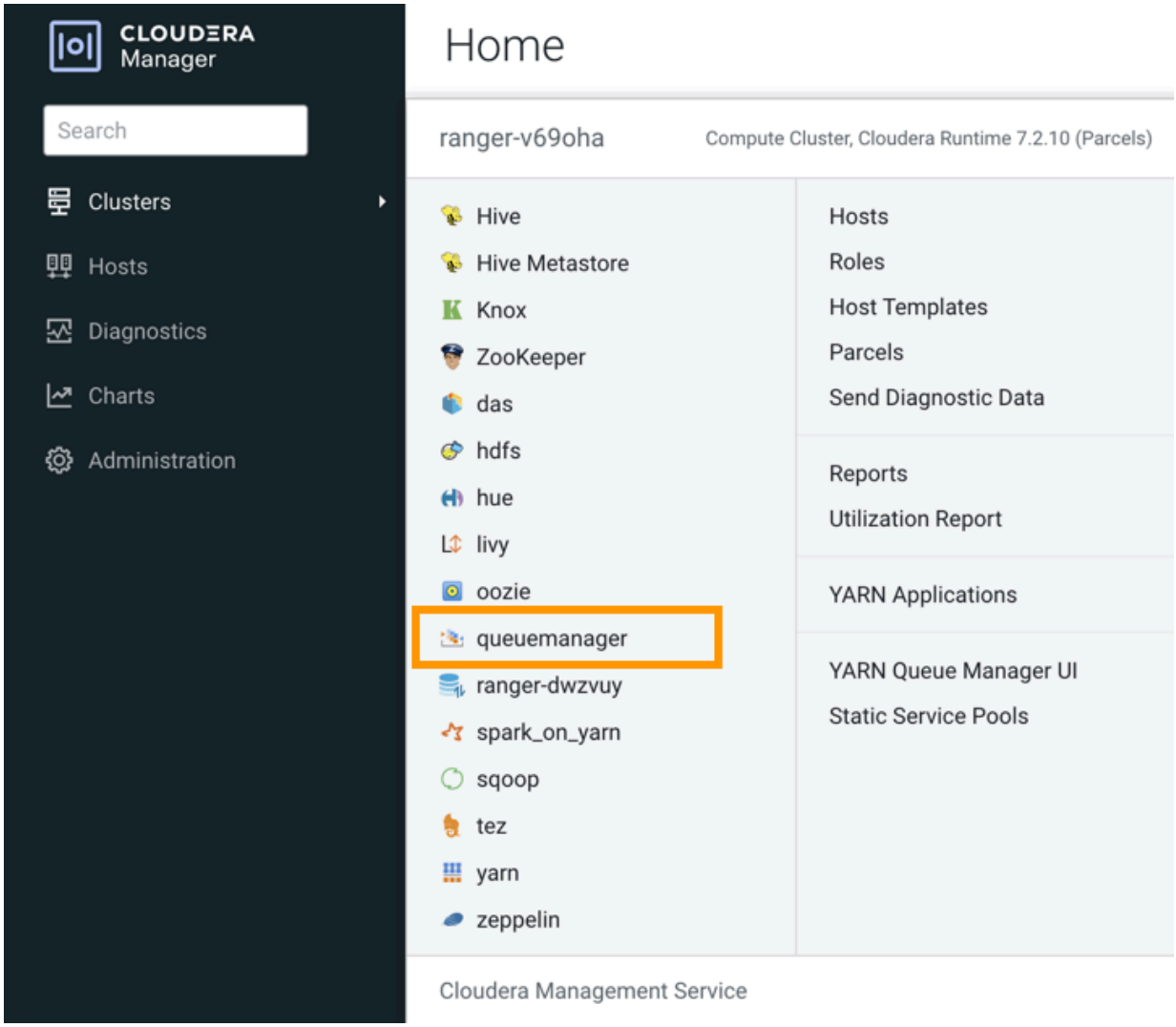

.

3. Go to the Configuration tab.
4. Specify the new file location /var/lib/hadoop-yarn/config-service in Location for config-service DB.

> **Note:** The config-service database file name should not contain only special characters and dot (.) characters. Even though the file name is allowed during the creation of database from the UI, in the backend the .mv.db file is not created. You must have the required permissions to the specified new file location.

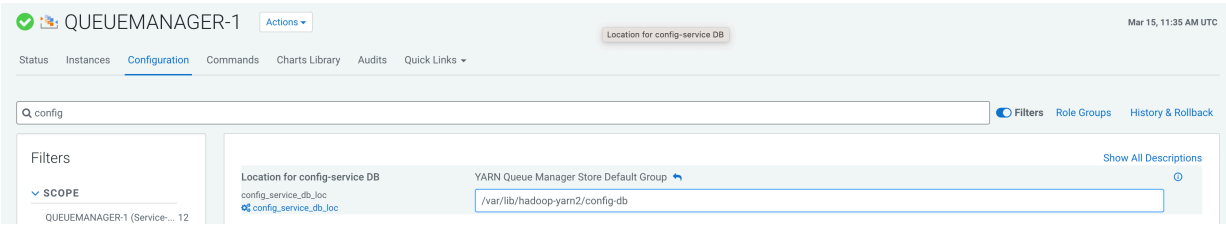

5. Click Save Changes.

6. Click Actions and select Restart.

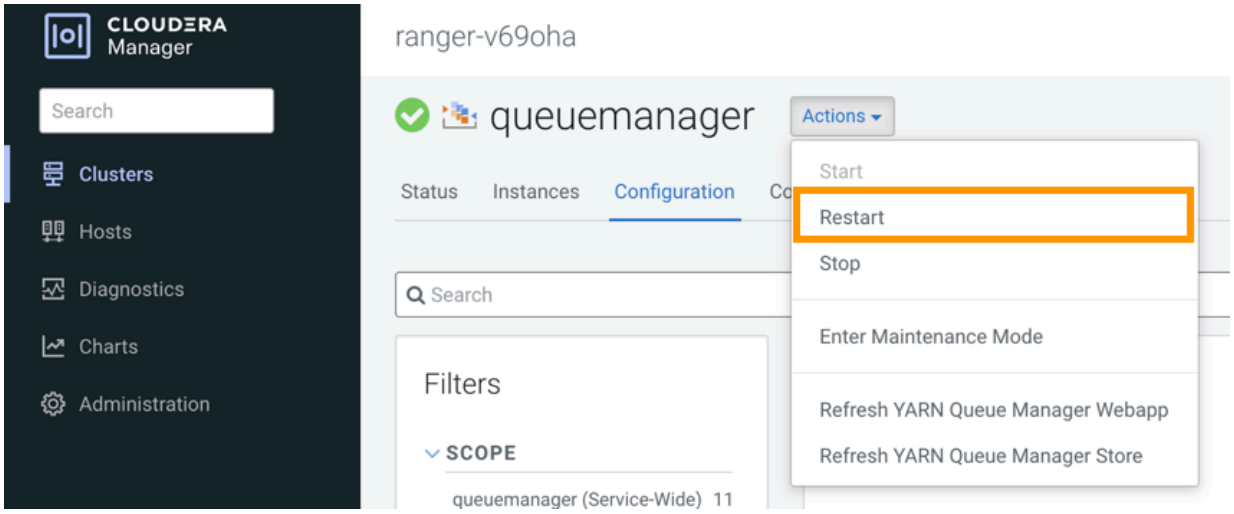

After you add the new database location in the Queue Manager service configuration, there will be a similar .mv.db file and .trace.db file in the new location.

7. SSH into the host machine running the YARN Queue Manager service.

```
ssh [***your_username***]@[***queue_manager_host_ip_address***]
```

8. Navigate to the directory where the configuration database file is stored.

```
cd /var/lib/hadoop-yarn/
```

9. Find the default location of the initial database file.

```
find / -name  "config-service.mv.db"
/var/lib/hadoop-yarn/config-service.mv.db
```

The complete path to the .db file is displayed.

10. Replicate the permission at the directory level for the new database location and verify if the database directory level permissions are the same.
11. Copy the contents of the old file .mv.db file to the new .mv.db file.

```
# cp config-service.mv.db new-location-config-service.mv.db
```

12. Type y to overwrite the file.
13. In Cloudera Manager, click Clusters > queue manager. Click Actions and select Restart to restart the service.

You have now changed the location of the database file. All queue information is now available in the new file and can be viewed from the Queue Manager UI.

Running YARN Services

You can use YARN Services API to manage YARN services. You can use the YARN CLI to view the logs for running applications. In addition, you can use YARN distributed cache to deploy multiple versions of Mapreduce.

Previously, deploying a new service on YARN was not a simple experience. The APIs of existing frameworks were either too low level (native YARN), required writing new code (for frameworks with programmatic APIs), or required writing a complex specification (for declarative frameworks). Apache Slider was developed to run services such as HBase, Storm, Accumulo, and Solr on YARN by exposing higher-level APIs that supported running these services on YARN without modification.

The new YARN Services API greatly simplifies the deployment and management of YARN services.

Deploying and managing services on YARN

Using the YARN Services API, you can run simple and complex template-based apps on containers.

Without having the need to write new code or modify your apps, you can create and manage the life cycle of these YARN services.

```
{
  "name": "sleeper-service",
  "version": "1.0.0",
  "components" : [
    {
      "name": "sleeper",
      "number_of_containers": 2,
      "launch_command": "sleep 900000",
      "resource": {
      "cpus": 1,
      "memory": "256"
     }
    }
   ]
  }
```

Each service file contains, at a minimum, a name, version, and list of components. Each component of a service has a name, a number of containers to be launched (also referred to as component instances), a launch command, and an amount of resources to be requested for each container.

Components optionally also include an artifact specification. The artifact can be the default type (with no artifact specified, like the sleeper-service example above) or can have other artifact types such as TARBALL, or SERVICE.

Launching a YARN service

Launching a service saves the service file to HDFS and starts the service.

Run the following command to launch the sleeper service example. This sleeper example is provided with YARN, so it can be referred to by the name *;sleeper;* or the path to the JSON file can be specified:

```
yarn app -launch sleeper-service <sleeper OR /path/to/sleeper.json>
```

This command saves and starts the sleeper service with two instances of the sleeper component. The service could also be launched by making calls to the REST API instead of using the command line. The service can be stopped and destroyed as follows. The stop command stops the service from running, but leaves the service JSON file stored in HDFS so that the service could be started again using a start command. The destroy command stops the service if it is running and removes the service information entirely.

```
yarn app -stop sleeper-service
```

```
yarn app -destroy sleeper-service
```

Saving a YARN service definition

You can save a service YARN file initially without starting the service and later refer to this service YARN file while launching other services.

Run the following command to save the simple-httpd-service YARN file:

```
yarn app -save simple-httpd-service /path/to/simple-httpd-service.json
```

Saving or launching the service from a YARN file stores a modified version of the YARN file in HDFS. This service specification can then be referenced by other services, allowing users to assemble more complex services.

Creating new YARN services using UI

The YARN Web User Interface enables you to define new services. You can either create standard services by providing their details or custom services by using JSON files containing the required definitions.

Creating a standard YARN service

You can create a standard service as a template and deploy it based on your requirements.

Procedure

1. On the Services page of the YARN Web User Interface, click New Service.
2. In the User name for service field, specify the name of the user who launches the service.
3. Enter the service definition details.
   - Service Name: Enter a unique name for the application.
   - Queue Name: Enter the name of the YARN queue to which this application should belong.
   - Service Lifetime: Life time of the application from the time it is in STARTED state till the time it is automatically destroyed by YARN. If you want to have unlimited life time, do not enter any value.
   - Service Components: Enter the details of the service components such as component name, CPU required, memory, number of containers, artifact ID, and launch command. If it is an application like HBase, the components can be a simple role like master or RegionServer. For a complex application, the components can be other nested applications with their own details.
   - Service Configurations: Set of configuration properties that can be ingested into the application components through environments, files, and custom pluggable helper docker containers. You can upload files of several formats such as properties files, JSON files, XML files, YAML files, and template files.
   - File Configurations: Set of file configurations that needs to be created and made available as a volume in an application component container. You can upload JSON file configurations to add to the service.
4. Click Save.
5. Specify a name for the new service and click Add.

   The newly created service is added to the list of saved templates.

   Note: Click Reset if you do not want to save your changes and specify the service details again.

6. Select the service and click Deploy to deploy it.

Creating a custom YARN service

You can define a service in JSON format and save it as a template.

Procedure

1. On the Services page of the YARN Web User Interface, click New Service.

2. Click the Custom tab.
3. In the User name for service field, specify the name of the user who launches the service.
4. In the Service Definition field, specify the service definition in JSON format.

   The following example shows the sleeper service template definition.

```
{
  "name": "sleeper-service",
  "version": "1.0.0",
  "components" :
  [
    {
      "name": "sleeper",
      "number_of_containers": 2,
      "launch_command": "sleep 900000",
      "resource": {
        "cpus": 1,
        "memory": "256"
      }
    }
  ]
}
```

5. Click Save.
6. Specify a name for the new service and click Add.

   The newly created service is added to the list of saved templates.

   Note: Click Reset if you do not want to save your changes and specify the service details again.

7. Select the service and click Deploy to deploy it.

Managing the YARN service life cycle through the REST API

You can perform various operations to manage the life cycle of a YARN service through the REST API.

Create a service

Use the following endpoint to create a service:

```
POST /app/v1/services
```

The execution of this command confirms the success of the service creation request. You cannot be sure if the service will reach a running state. Resource availability and other factors will determine if the service will be deployed in the cluster. You have to call the GET API to get the details of the service and determine its state.

Update a service or upgrade the binary version of a service

You can update the runtime properties of a service. You can update the lifetime, and start or stop a service. You can also upgrade the service containers to a newer version of their artifacts.

Use the following endpoint to update the service:

```
PUT /app/v1/services/{service_name}
```

Destroy a service

Use the following endpoint to destroy a service and release all its resources.

```
DELETE /app/v1/services/{service_name}
```

Get the details of a service

Use the following endpoint to view the details (including containers) of a running service.

```
GET /app/v1/services/{service_name}
```

Set the number of instances of a component

Use the following endpoint to set a component's desired number of instances:

```
PUT /app/v1/services/{service_name}/components/{component_name}
```

YARN services API examples

You can use the YARN Services API for situations such as creating a single-component service and performing various operations on the service.

- Create a simple single-component service with most attribute values as defaults

  POST URL – http://localhost:8088/app/v1/services

  POST Request JSON

```
{
  "name": "hello-world",
  "version": "1",
  "components": [
    {
      "name": "hello",
      "number_of_containers": 1,
      "artifact": {
        "id": "nginx:latest",
        "type": "DOCKER"
      },
      "launch_command": "./start_nginx.sh",
      "resource": {
        "cpus": 1,
        "memory": "256"
      }
    }
  ]
}
```

  GET Response JSON

  GET URL – http://localhost:8088/app/v1/services/hello-world

  Note that a lifetime value of -1 means unlimited lifetime.

```
{
    "name": "hello-world",
    "version": "1",
```

```
    "id": "application_1503963985568_0002",
    "lifetime": -1,
    "components": [
        {
            "name": "hello",
            "dependencies": [],
            "resource": {
                "cpus": 1,
                "memory": "256"
            },
            "configuration": {
                "properties": {},
                "env": {},
                "files": []
            },
            "quicklinks": [],
            "containers": [
                {
                    "id": "container_e03_1503963985568_0002_01_000001",
                    "ip": "10.22.8.143",
                    "hostname": "myhost.local",
                    "state": "READY",
                    "launch_time": 1504051512412,
                    "bare_host": "10.22.8.143",
                    "component_name": "hello-0"
                },
                {
                    "id": "container_e03_1503963985568_0002_01_000002",
                    "ip": "10.22.8.143",
                    "hostname": "myhost.local",
                    "state": "READY",
                    "launch_time": 1504051536450,
                    "bare_host": "10.22.8.143",
                    "component_name": "hello-1"
                }
            ],
            "launch_command": "./start_nginx.sh",
            "number_of_containers": 1,
            "run_privileged_container": false
        }
    ],
    "configuration": {
        "properties": {},
        "env": {},
        "files": []
    },
    "quicklinks": {}
}
```

- Update the lifetime of a service

  PUT URL – http://localhost:8088/app/v1/services/hello-world

  PUT Request JSON

  Note that irrespective of what the current lifetime value is, this update request will set the lifetime of the service to 3600 seconds (1 hour) from the time the request is submitted. Therefore, if a service has remaining lifetime of 5 minutes (for example) and would like to extend it to an hour, OR if an application has remaining lifetime of 5 hours (for example) and would like to reduce it down to one hour, then for both scenarios you would need to submit the following request.

```
{
  "lifetime": 3600
```

```
}
```

- Stop a service

  PUT URL – http://localhost:8088/app/v1/services/hello-world

  PUT Request JSON

```
{
    "state": "STOPPED"
}
```

- Start a service

  PUT URL – http://localhost:8088/app/v1/services/hello-world

  PUT Request JSON

```
{
    "state": "STARTED"
}
```

- Increase or decrease the number of containers (instances) of a component of a service

  PUT URL – http://localhost:8088/app/v1/services/hello-world/components/hello

  PUT Request JSON

```
{
    "number_of_containers": 3
}
```

- Destroy a service

  DELETE URL – http://localhost:8088/app/v1/services/hello-world

- Create a complicated service – HBase

  POST URL - http://localhost:8088/app/v1/services/hbase-app-1

```
{
  "name": "hbase-app-1",
  "lifetime": "3600",
  "version": "2.0.0.3.0.0.0",
  "artifact": {
    "id": "hbase:2.0.0.3.0.0.0",
    "type": "DOCKER"
  },
  "configuration": {
    "env": {
      "HBASE_LOG_DIR": "<LOG_DIR>"
    },
    "files": [
      {
        "type": "TEMPLATE",
        "dest_file": "/etc/hadoop/conf/core-site.xml",
        "src_file": "core-site.xml"
      },
      {
        "type": "TEMPLATE",
        "dest_file": "/etc/hadoop/conf/hdfs-site.xml",
        "src_file": "hdfs-site.xml"
      },
      {
        "type": "XML",
        "dest_file": "/etc/hbase/conf/hbase-site.xml",
```

```
        "properties": {
          "hbase.cluster.distributed": "true",
          "hbase.zookeeper.quorum": "${CLUSTER_ZK_QUORUM}",
          "hbase.rootdir": "${SERVICE_HDFS_DIR}/hbase",
          "zookeeper.znode.parent": "${SERVICE_ZK_PATH}",
          "hbase.master.hostname": "hbasemaster-0.${SERVICE_NAME}.${USER}.
${DOMAIN}",
          "hbase.master.info.port": "16010"
        }
      }
    ]
  },
  "components": [
    {
      "name": "hbasemaster",
      "number_of_containers": 1,
      "launch_command": "sleep 15;/opt/cloudera/parcels/CDH-<version>/bin/
hbase master start",
      "resource": {
        "cpus": 1,
        "memory": "2048"
      },
      "configuration": {
        "env": {
          "HBASE_MASTER_OPTS": "-Xmx2048m -Xms1024m"
        }
      }
    },
    {
      "name": "regionserver",
      "number_of_containers": 1,
      "launch_command": "sleep 15; /opt/cloudera/parcels/CDH-<version>/bi
n/hbase master start",
      "dependencies": [
        "hbasemaster"
      ],
      "resource": {
        "cpus": 1,
        "memory": "2048"
      },
      "configuration": {
        "files": [
          {
            "type": "XML",
            "dest_file": "/etc/hbase/conf/hbase-site.xml",
            "properties": {
              "hbase.regionserver.hostname": "${COMPONENT_INSTANCE_NAME}.
${SERVICE_NAME}.${USER}.${DOMAIN}"
            }
          }
        ],
        "env": {
          "HBASE_REGIONSERVER_OPTS": "-XX:CMSInitiatingOccupancyFraction=
70 -Xmx2048m -Xms1024m"
        }
      }
    },
    {
      "name": "hbaseclient",
      "number_of_containers": 1,
      "launch_command": "sleep infinity",
      "resource": {
        "cpus": 1,
        "memory": "1024"
```

```
      }
    }
  ],
  "quicklinks": {
    "HBase Master Status UI": "http://hbasemaster-0.${SERVICE_NAME}.${US
ER}.${DOMAIN}:16010/master-status"
  }
}
```

Cloudera Runtime 7.1.9

Configuring Apache Hadoop YARN High Availability

Date published: 2020-02-11
Date modified: 2023-08-29

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

YARN ResourceManager high availability.....4
YARN ResourceManager high availability architecture..... 4
Configuring YARN ResourceManager high availability.....5
Using the yarn rmadmin tool to administer ResourceManager high availability..... 6
Migrating ResourceManager to another host.....6

Work preserving recovery for YARN components.....7
Configuring work preserving recovery on ResourceManager..... 7
Configuring work preserving recovery on NodeManager..... 7
Example: Configuration for work preserving recovery..... 8

YARN ResourceManager high availability

The ResourceManager high availability (HA) feature adds redundancy in the form of an active-standby ResourceManager pair.

The YARN ResourceManager is responsible for tracking the resources in a cluster and scheduling applications (for example, MapReduce jobs). The ResourceManager high availability (HA) feature adds redundancy in the form of an active-standby ResourceManager pair to remove this single point of failure. Furthermore, upon failover from the active ResourceManager to the standby, the applications can resume from the last state saved to the state store; for example, map tasks in a MapReduce job are not run again if a failover to a new active ResourceManager occurs after the completion of the map phase. This allows events such the following to be handled without any significant performance effect on running applications:

- Unplanned events such as machine crashes
- Planned maintenance events such as software or hardware upgrades on the machine running the ResourceManager

ResourceManager HA requires ZooKeeper and HDFS services to be running.

YARN ResourceManager high availability architecture

Learn about the architecture of YARN ResourceManager High Availability to use it efficiently.

ResourceManager HA is implemented by means of an active-standby pair of ResourceManagers. On start-up, each ResourceManager is in the standby state; the process is started, but the state is not loaded. When one of the ResourceManagers is transitioning to the active state, the ResourceManager loads the internal state from the designated state store and starts all the internal services. The stimulus to transition to active comes from either the administrator (through the CLI) or through the integrated failover controller when automatic failover is enabled.

ResourceManager Restart

Restarting the ResourceManager allows for the recovery of in-flight applications if recovery is enabled. To achieve this, the ResourceManager stores its internal state, primarily application-related data and tokens, to the RMStateS tore; the cluster resources are re-constructed when the NodeManagers connect. The available alternatives for the state store are MemoryRMStateStore (a memory-based implementation) and ZKRMStateStore (ZooKeeper-based implementation). Note that MemoryRMStateStore will not work for HA.

Fencing

When running two ResourceManagers, a split-brain situation can arise where both ResourceManagers assume they are active. To avoid this, only a single ResourceManager should be able to perform active operations and the other ResourceManager should be "fenced". The ZooKeeper-based state store (ZKRMStateStore) allows only a single ResourceManager to make changes to the stored state, implicitly fencing the other ResourceManager. This is accomplished by the ResourceManager claiming exclusive create-delete permissions on the root znode. The ACLs on the root znode are automatically created based on the ACLs configured for the store; in case of secure clusters, Cloudera recommends that you set ACLs for the root host such that both ResourceManagers share read-write-admin access, but have exclusive create-delete access. The fencing is implicit and does not require explicit configuration (as fencing in HDFS does). You can plug in a custom "Fencer" if you choose to – for example, to use a different implementation of the state store.

Configuration and FailoverProxy

In an HA setting, you should configure two ResourceManagers to use different ports (for example, ports on different hosts). To facilitate this, YARN uses the notion of an ResourceManager Identifier (rm-id). Each ResourceManager has a unique rm-id, and all the RPC configurations (<*RPC-ADDRESS*>; for example yarn.resourcemanager.add ress) for that ResourceManager can be configured via <rpc-address>.<rm-id>. Clients, ApplicationMasters, and NodeManagers use these RPC addresses to talk to the active ResourceManager automatically, even after a failover.

To achieve this, they cycle through the list of ResourceManagers in the configuration. This is done automatically and does not require any configuration (as it does in HDFS).

Automatic Failover

By default, ResourceManager HA uses ZKFC (ZooKeeper-based failover controller) for automatic failover in case the active ResourceManager is unreachable or goes down. Internally, the *StandbyElector* is used to elect the active ResourceManager. The failover controller runs as part of the ResourceManager.

You can plug in a custom failover controller if you prefer.

Manual Transitions and Failover

You can use the command-line tool yarn rmadmin to transition a particular ResourceManager to active or standby state, to fail over from one ResourceManager to the other, to get the HA state of an ResourceManager, and to monitor an ResourceManager's health.

Configuring YARN ResourceManager high availability

You can use Cloudera Manager to configure YARN ResourceManager high availability (HA).

Cloudera Manager supports automatic failover of the ResourceManager. It does not provide a mechanism to manually force a failover through the Cloudera Manager user interface.

Important: Enabling or disabling High Availability will cause the previous monitoring history to become unavailable.

Enable high availability

You can enable YARN ResourceManager High Availability using Cloudera Manager. When you enable ResourceManager HA in Cloudera Manager, work preserving recovery is also enabled for the ResourceManager by default.

1. In Cloudera Manager, select the YARN service.
2. Click Actions.
3. Select Enable High Availability.

   A screen showing the hosts that are eligible to run a standby ResourceManager displays. The host where the current ResourceManager is running is not available as a choice.
4. Select the host where you want the standby ResourceManager to be installed.
5. Click Continue.

   Cloudera Manager proceeds to run a set of commands that stop the YARN service, add a standby ResourceManager, initialize the ResourceManager high availability state in ZooKeeper, restart YARN, and redeploy the relevant client configurations.

   ResourceManager HA does not affect the JobHistory Server (JHS). JHS does not maintain any state, so if the host fails you can simply assign it to a new host. If you want to enable process auto-restart, do the following:
6. In Cloudera Manager, select the YARN service.
7. Click the Configuration tab.
8. Search for restart.
9. Find the Automatically Restart Process property.
10. Click Edit Individual Values.
11. Select the JobHistory Server Default Group option.
12. Click Save Changes.
13. Restart the JobHistory Server role.

Disable high availability

You can disable YARN ResourceManager High Availability using Cloudera Manager.

1. In Cloudera Manager, select the YARN service.
2. Click Actions.
3. Select Disable High Availability.

   A screen showing the hosts running the ResourceManagers displays.
4. Select the ResourceManager (host) you want to remain as the single ResourceManager.
5. Click Continue.

   Cloudera Manager runs a set of commands that stop the YARN service, remove the standby ResourceManager and the Failover Controller, restart the YARN service, and redeploy client configurations.

Using the yarn rmadmin tool to administer ResourceManager high availability

You can use yarn rmadmin in the command line to manage your ResourceManager HA deployment.

Note: Manually failing over the ResourceManager is not possible when high availability is enabled.

yarn rmadmin has the following options related to ResourceManager high availability:

```
[-getServiceState serviceId]
[-checkHealth <serviceId]
[-help <command>]
```

when *serviceID* is rm-id.

Note: Even though -help lists the -failover option, it is not supported by yarn rmadmin.

Migrating ResourceManager to another host

You can move the YARN ResourceManager role from one host to another using Cloudera Manager.

Before you begin

Cloudera recommends to enable RecourceManager High Availability.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click Instance.
3. Find the ResourceManager role that you want to migrate to a new host and select it.
4. Select Actions for Selected Stop .
5. Select Actions for Selected Delete .
6. Click Add Role Instances.
7. Use the dropdown menu under ResourceManager to select a hosts for the new ResourceManager role.
8. Click Continue and then click Finish.
9. Select the newly created ResourceManager role.
10. Select Actions for Selected Start and click Start.

11. Restart the YARN service.

Results

The ResourceManager role now run on a new host.

Work preserving recovery for YARN components

With work preserving recovery enabled, if a ResourceManager or NodeManager restarts, no in-flight work is lost.

You can configure work preserving recovery separately for a ResourceManager or NodeManager. You can use work preserving recovery whether or not you use ResourceManager High Availability.

Note: YARN does not support high availability for the JobHistory Server (JHS). If the JHS goes down, Cloudera Manager restarts it automatically.

Note:

After moving the JobHistory Server to a new host, the URLs listed for the JobHistory Server on the ResourceManager web UI still point to the old JobHistory Server. This affects existing jobs only. New jobs started after the move are not affected. For any existing jobs that have the incorrect JobHistory Server URL, there is no option other than to allow the jobs to roll off the history over time. For new jobs, make sure that all clients have the updated mapred-site.xml that references the correct JobHistory Server.

Configuring work preserving recovery on ResourceManager

Work preserving recovery is enabled by default for the ResourceManager. You can disable this feature using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for Enable ResourceManager Recovery.
4. Clear the ResourceManager Default Group checkbox in the Enable ResourceManager Recovery field.
5. Click Save Changes.

Configuring work preserving recovery on NodeManager

Work preserving recovery is enabled by default for the NodeManager. You can disable this feature using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for yarn-site.xml.
4. Find the NodeManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml.
5. Select the NodeManager and click on the plus icon.
6. Add the following:
   - Name: yarn.nodemanager.recovery.enabled
   - Value: false
7. Click Save Changes.

Example: Configuration for work preserving recovery

Use work preserving recovery properties to enable and configure this feature.

```
<property>
<name>yarn.resourcemanager.work-preserving-recovery.enabled</name>
<value>true</value>
<description>Whether to enable work preserving recovery for the Resource Ma
nager.</description>
</property>

<property>
<name>yarn.nodemanager.recovery.enabled</name>
<value>true</value>
<description>Whether to enable work preserving recovery for the Node Mana
ger.</description>
</property>

<property>
<name>yarn.nodemanager.recovery.dir</name>
<value>/home/cloudera/recovery</value>
<description>The location for stored state on the Node Manager, if work pres
erving recovery is enabled.</description>
</property>

<property>
<name>yarn.nodemanager.address</name>
<value>0.0.0.0:45454</value>
<description>The address of the container manager in the Node Manager.</desc
ription>
</property>

<property>
<name>yarn.resourcemanager.store.class</name>
<value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateSt
ore</value>
 <description>The class name of the state-store to be used for saving app
lication/attempt state and the credentials. </description>
</property>

<property>
<name>hadoop.zk.address</name>
<value>0.0.0.0:3001,0.0.0.0:3002,0.0.0.0:3003</value>
<description>Comma separated list of Host:Port pairs. Each corresponds to a
 ZooKeeper to be used by the Resource Manager for storing Resource Manager s
tate.</description>
</property>

<property>
<name>yarn.resourcemanager.zk-state-store.parent-path</name>
<value>/rmstore</value>
<description>The full path of the root znode where Resource Manager state
will be stored.</description>
</property>
<property>
<name>hadoop.zk.num-retries</name>
<value>500</value>
<description>Number of times Resource Manager tries to connect to ZooKeeper
 server if the connection is lost.</description>
</property>

<property>
```

```
<name>hadoop.zk.retry-interval-ms</name>
<value>120</value>
<description>The interval in milliseconds between retries when connecting to
 a ZooKeeper server.</description>
</property>

<property>
<name>hadoop.zk.timeout-ms</name>
<value>10</value>
<description> ZooKeeper session timeout in milliseconds. Session expiration
 is managed by the ZooKeeper cluster itself and not by the client. This val
ue is used by the cluster to determine when the client's session expires. Ex
pirations happens when the cluster does not hear from the client within the
specified session timeout period (that is, no heartbeat). </description>
</property>

<property>
<name>hadoop.zk.acl</name>
<value>world:anyone:rwcda</value>
<description>ACLs to be used for setting permissions on ZooKeeper znodes.</
description>
</property>
```

Cloudera Runtime 7.3.1

Configuring YARN Log Aggregation

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

YARN log aggregation overview....4

Log aggregation file controllers....4

Configuring log aggregation....4

Log aggregation properties....5

Configuring debug delay....7

YARN log aggregation overview

The YARN Log Aggregation feature enables you to move local log files of any application onto HDFS or a cloud-based storage depending on your cluster configuration.

Application logs has great significance: meaningful information can be extracted from them, can be used to debug issues or can be kept for historical analysis. YARN can move local logs securely onto HDFS or a cloud-based storage, such as AWS. This allows the logs to be stored for a much longer time than they could be on a local disk, allows faster search for a particular log file and optionally can handle compression.

Log aggregation file controllers

By default, log aggregation supports two file controllers, TFile and IFile. You can also add your own custom file controller.

By default IFile is used to write the aggregated logs.

Note: TFile and IFile are commonly referred as file formats. However, they do not only define the format of the file, but also how the log aggregation is managed. Therefore referring to them as file controllers specifies them more clearly.

TFile

TFile is the legacy file controller in YARN. It is reliable and well tested. Its buffer and chunk sizes are configurable.

TFile provides the following features:

- Block compression
- Named metadata blocks
- Sorted or unsorted keys
- Seek by key or by file offset

IFile

IFile is a newer file controller than TFile. It also uses TFile internally so it provides the same features as TFile.

In an IFile the files are indexed so it is faster to search in the aggregated log file than in a regular TFile. It uses checksums and temporary files which help to prevent failures. Its buffer sizes and rollover file size are configurable on top of the configuration options of TFile.

Configuring log aggregation

Log aggregation is enabled by default. You can configure it using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Select the Log Aggregation filter under Category.
   Every log aggregation related property is displayed.
4. Find the Enable Log Aggregation property and ensure that it is selected, meaning that log aggregation is enabled.

5. Configure the log aggregation properties as applicable.

Important:

The remote-app-log-dir and remote-app-log-dir-suffix configuration properties are not backward compatible. If you change their value later, the logs of applications that are already finished cannot be read.

6. Click Save Changes.

Related Information

Log aggregation properties

Log aggregation properties

You can configure YARN log aggregation for your cluster by specifying values of specific properties depending on your requirements.

Name	Property	Default value	Description
Enable Log Aggregation	yarn.log-aggregation-enable	Enabled	Specifies if log aggregation is enabled. If log aggregation is disabled, YARN container logs are not aggregated.
Log Aggregation Retention Period	yarn.log-aggregation.retain-seconds	7 days	Specifies how long aggregation logs are kept before they are deleted.
Log Aggregation Status Timeout	yarn.log-aggregation-status.time-out.ms	10 minutes	Specifies the maximum amount of time that the NodeManager has for reporting a container's log aggregation status. If no log aggregation status is sent by the NodeManager within the allotted time, the ResourceManager reports a TIME_OUT log aggregation status for that container.
Supported Log Aggregation File Formats	yarn.log-aggregation.file-formats	IFile and TFile	Specifies which log file formats are supported. The first file format in the list is used to write the aggregated logs. TFile format is always added to the end of the list, to support backward compatibility.
Remote App Log Directory for TFile Format	yarn.log-aggregation.TFile.remote-app-log-dir	empty	Specifies the path of the directory where application logs are stored after an application is completed if TFile format is selected for writing. This configuration overwrites the one given in NodeManager level (yarn.nodemanager.remote-app-log-dir). This property is not backward compatible. If you change its value later, the logs of applications that are already finished cannot be read.

Name	Property	Default value	Description
Remote App Log Directory Suffix for TFile Format	yarn.log-aggregation.TFile.remote-app-log-dir-suffix	empty	The remote log directory is created at {remote-app-log-dir}/${user}/{thisParam} if TFile format is selected for writing. This configuration overwrites the one given in NodeManager level (yarn.nodemanager.remote-app-log-dir-suffix). This property is not backward compatible. If you change its value later, the logs of applications that are already finished cannot be read.
Remote App Log Directory for IFile Format	yarn.log-aggregation.IFile.remote-app-log-dor	empty	Specifies the path of the directory where application logs are stored after an application is completed if IFile format is given as the file format for writing. This configuration overwrites the one given in NodeManager level (yarn.nodemanager.remote-app-log-dir). This property is not backward compatible. If you change its value later, the logs of applications that are already finished cannot be read.
Remote App Log Directory Suffix for IFile Format	yarn.log-aggregation.IFile.remote-app-log-dir-suffix	empty	The remote log directory is created at {remote-app-log-dir}/${user}/{thisParam} if IFile format is selected for writing. This configuration overwrites the one given in NodeManager level (yarn.nodemanager.remote-app-log-dir-suffix). This property is not backward compatible. If you change its value later, the logs of applications that are already finished cannot be read.
Log Aggregation Compression Type	yarn.nodemanager.log-aggregation.compression-type	none	Specifies the TFile compression type used to compress aggregated logs. Its value can be: gz, lzo or none.
Remote App Log Directory	yarn.nodemanager.remote-app-log-dir	/tmp/logs	Specifies the path of the directory where application logs are stored after an application is completed. This property is not backward compatible. If you change its value later, the logs of applications that are already finished cannot be read.

Name	Property	Default value	Description
Remote App Log Directory Suffix	yarn.nodemanager.remote-app-log-dir-suffix	logs	The remote log dir will be created at {yarn.nodemanager.remote-app-log-dir}/${user}/{thisParam}. This property is not backward compatible. If you change its value later, the logs of applications that are already finished cannot be read.
Log Aggregation Policy	yarn.nodemanager.log-aggregation.policy.class	All	Specifies the types of container logs that are uploaded during the log aggregation. Valid values are: Only AM, AM or Failed, All, Only Failed, Failed or Killed, None, or Sample.

Related Information

Configuring log aggregation

Configuring debug delay

You can keep the application's localized files and log directory on the local disk for a predefined time before the NodeManager deletes them.

About this task

This feature can be useful for example when there is a problem with the log aggregation feature or if you want to diagnose YARN application issues.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for deletion delay.
4. Find the Localized Dir Deletion Delay property.
5. Define the delay time in seconds.

   If you want to diagnose YARN application issues, set this property to a large value, for example to 600 seconds, to permit examination of these directories.
6. Click Save Changes.

Cloudera Runtime 7.3.1

Apache Hadoop YARN Overview

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction 4

YARN Features 4

Understanding YARN architecture 4

Introduction

Apache Hadoop YARN is the processing layer for managing distributed applications that run on multiple machines in a network.

YARN Features

YARN enables you to manage resources and schedule jobs in Hadoop.

YARN provides the following features:

Multi-tenancy

You can use multiple open-source and proprietary data access engines for batch, interactive, and real-time access to the same dataset. Multi-tenant data processing improves an enterprise's return on its Hadoop investments.

Cluster utilization

You can dynamically allocate cluster resources to improve resource utilization.

Multiple resource types

You can use multiple resource types such as memory, CPU, and GPU.

Scalability

Significantly improved data center processing power. YARN's ResourceManager focuses exclusively on scheduling and keeps pace as clusters expand to thousands of nodes managing petabytes of data.

Compatibility

MapReduce applications developed for Hadoop 1 runs on YARN without any disruption to existing processes. YARN maintains API compatability with the previous stable release of Hadoop.

Understanding YARN architecture

YARN allows you to use various data processing engines for batch, interactive, and real-time stream processing of data stored in HDFS or cloud storage like S3 and ADLS. You can use different processing frameworks for different use-cases, for example, you can run Hive for SQL applications, Spark for in-memory applications, and Storm for streaming applications, all on the same Hadoop cluster.

YARN extends the power of Hadoop to new technologies found within the data center so that you can take advantage of cost-effective linear-scale storage and processing. It provides independent software vendors and developers a consistent framework for writing data access applications that run in Hadoop.

YARN architecture and workflow

YARN has three main components:

- ResourceManager: Allocates cluster resources using a Scheduler and ApplicationManager.
- ApplicationMaster: Manages the life-cycle of a job by directing the NodeManager to create or destroy a container for a job. There is only one ApplicationMaster for a job.
- NodeManager: Manages jobs or workflow in a specific node by creating and destroying containers in a cluster node.

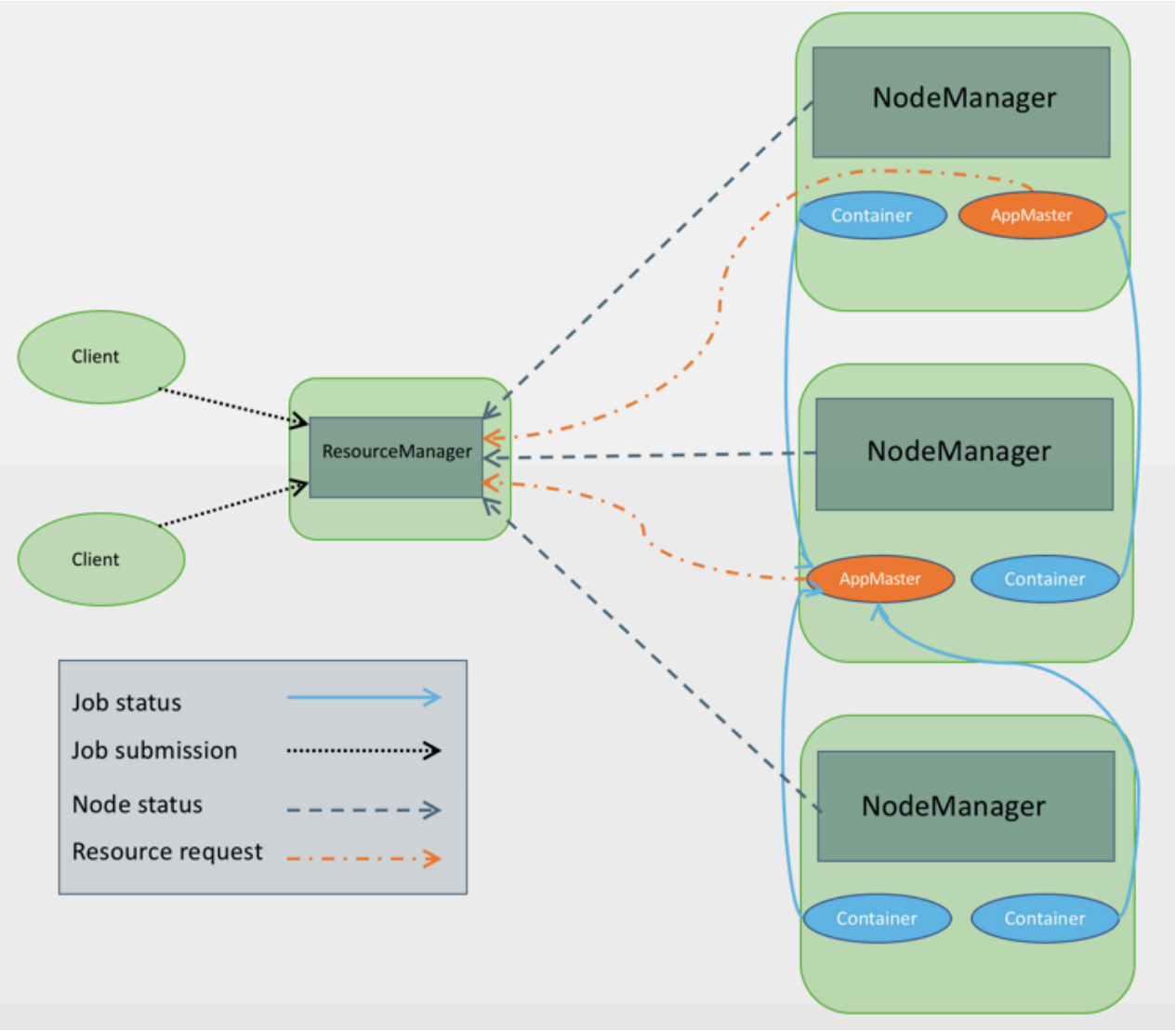
NodeManager
Container
AppMaster
Client
ResourceManager
NodeManager
Client
AppMaster
Container
Job status
Job submission
Node status
Resource request
NodeManager
Container
Container

Cloudera Runtime 7.3.1

Apache Hadoop YARN Reference

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Tuning Apache Hadoop YARN..... 4
YARN tuning overview.....4
Step 1: Worker host configuration.....8
Step 2: Worker host planning..... 8
Step 3: Cluster size.....9
Steps 4 and 5: Verify settings.....9
Step 6: Verify container settings on cluster.....9
Step 6A: Cluster container capacity.....10
Step 6B: Container parameters checking.....10
Step 7: MapReduce configuration.....10
Step 7A: MapReduce settings checking.....11
Set properties in Cloudera Manager..... 11
Configure memory settings..... 12

YARN Configuration Properties.....13

Use the YARN REST APIs to manage applications..... 15

Comparison of Fair Scheduler with Capacity Scheduler..... 18
Why one scheduler?..... 18
Scheduler performance improvements..... 19
Feature comparison.....20
Migration from Fair Scheduler to Capacity Scheduler.....21

Configuring and using Queue Manager REST API..... 21
Limitations..... 22
Using the REST API.....22
Prerequisites..... 22
Start Queue..... 23
Stop Queue.....24
Add Queue..... 25
Change Queue Capacities.....26
Change Queue Properties..... 27
Delete Queue.....28

Tuning Apache Hadoop YARN

YARN tuning overview

Abstract description of a YARN cluster and the goals of YARN tuning.

This topic applies to YARN clusters only, and describes how to tune and optimize YARN for your cluster.

Note: Download the Cloudera YARN tuning spreadsheet to help calculate YARN configurations. For a short video overview, see Tuning YARN Applications.

This overview provides an abstract description of a YARN cluster and the goals of YARN tuning.

A YARN cluster is composed of host machines. Hosts provide memory and CPU resources. A vcore, or virtual core, is a usage share of a host CPU.	

Tuning YARN consists primarily of optimally defining containers on your worker hosts. You can think of a container as a rectangular graph consisting of memory and vcores. Containers perform tasks.	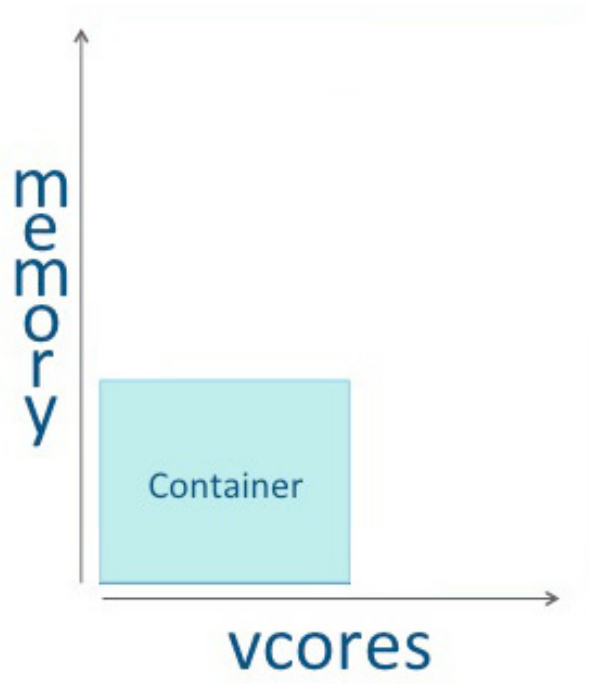
Some tasks use a great deal of memory, with minimal processing on a large volume of data.	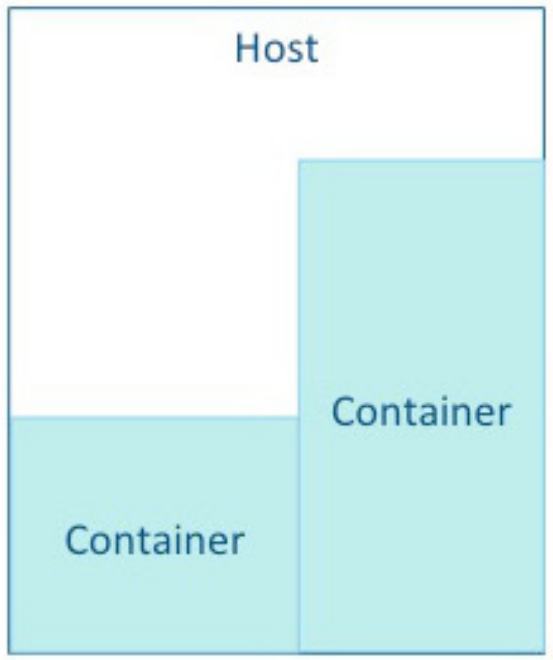

Other tasks require a great deal of processing power, but use less memory. For example, a Monte Carlo Simulation that evaluates many possible "what if?" scenarios uses a great deal of processing power on a relatively small dataset.	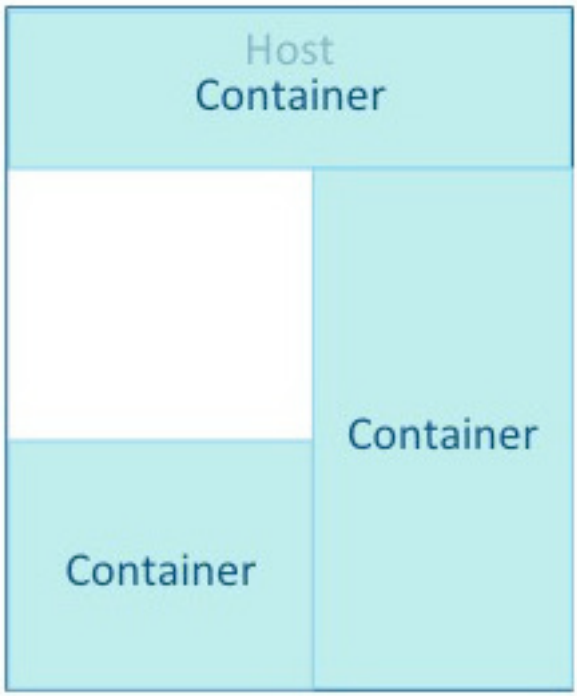
The YARN ResourceManager allocates memory and vcores to use all available resources in the most efficient way possible. Ideally, few or no resources are left idle.	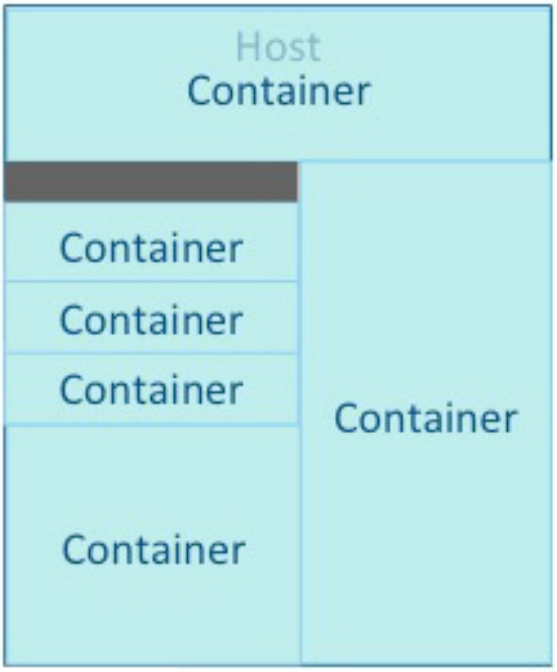

An application is a YARN client program consisting of one or more tasks. Typically, a task uses all of the available resources in the container. A task cannot consume more than its designated allocation, ensuring that it cannot use all of the host CPU cycles or exceed its memory allotment.	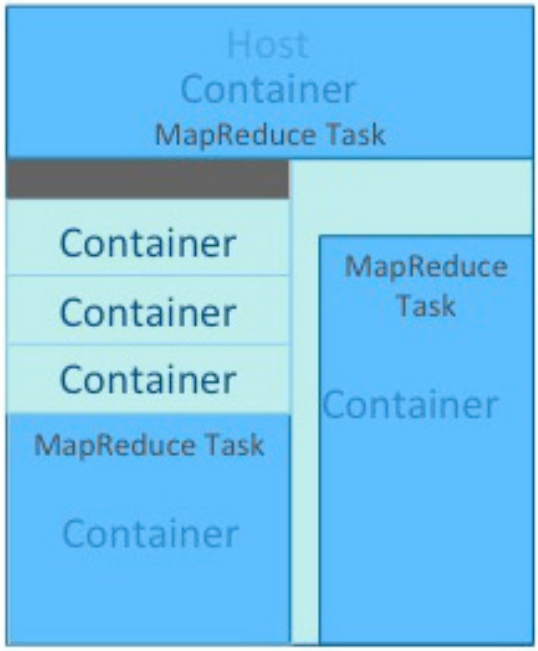
Tune your YARN hosts to optimize the use of vcores and memory by configuring your containers to use all available resources beyond those required for overhead and other services.	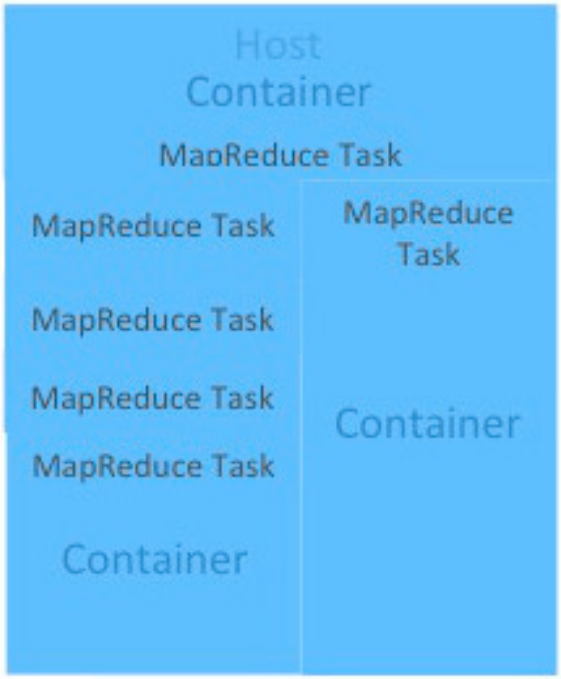

YARN tuning has three phases. The phases correspond to the tabs in the YARN tuning spreadsheet.

1. Cluster configuration, where you configure your hosts.
2. YARN configuration, where you quantify memory and vcores.
3. MapReduce configuration, where you allocate minimum and maximum resources for specific map and reduce tasks.

YARN and MapReduce have many configurable properties. The YARN tuning spreadsheet lists the essential subset of these properties that are most likely to improve performance for common MapReduce applications.

Step 1: Worker host configuration

Define the configuration for a single worker host computer in your cluster

Step 1 is to define the configuration for a single worker host computer in your cluster.

STEP 1: Worker Host Configuration

Enter your likely machine configuration in the input boxes below. If you are uncertain what machines you plan on buying, put in some minimum values that will suit what you expect to buy.

Host Components	Quantity	Size	Total	Description / Notes
RAM	256G		256G	Node memory in Gigabytes
CPU	4	6	48	Number of CPU's and the number of HW cores per CPU. The calculation of vcores below includes HyperThreading support.
HyperThreading CPU	yes			Does the CPU support HyperThreading?
HDD (Hard Disk Drive)	24	3T	72G	Number of Hard Drives and size per drive in JBOD Configuration
Ethernet	2	1G	2G	Number of Ethernet connections and the transfer speed

As with any system, the more memory and CPU resources available, the faster the cluster can process large amounts of data. A machine with 4 CPUs with HyperThreading, each with 6 cores, provides 48 vcores per host.

3 TB hard drives in a 2-unit server installation with 12 available slots in JBOD (Just a Bunch Of Disks) configuration is a reasonable balance of performance and pricing at the time the spreadsheet was created. The cost of storage decreases over time, so you might consider 4 TB disks. Larger disks are expensive and not required for all use cases.

Two 1-Gigabit Ethernet ports provide sufficient throughput at the time the spreadsheet was published, but 10-Gigabit Ethernet ports are an option where price is of less concern than speed.

Step 2: Worker host planning

Allocate resources on each worker machine,

STEP 2: Worker Host Planning

Now that you have your base Host configuration from Step 1, use the table below to allocate resources, mainly CPU and memory, to the various software components that run on the host.

Service	Category	CPU (cores)	Memory (MB)	Notes
Operating System	Overhead	1	8192	Most operating systems use 4-8GB minimum.
Other services	Overhead	0	0	Enter the required cores or memory for non CDH services not part of the OS.
Cloudera Manager agent	Overhead	1	1024	Allocate 1GB and 1 vcore for Cloudera Manager agents, which track resource usage on a host.
HDFS DataNode	CDH	1	1024	Allocation for the HDFS DataNode heap: default 1GB and 1 vcore.
YARN NodeManager	CDH	1	1024	Allocation for the YARN NodeManager heap: default 1GB and 1 vcore
Impala daemon	CDH	0	0	(Optional Service) Suggestion: Allocate at least 16GB memory when using Impala.
Hbase RegionServer	CDH	0	0	(Optional Service) Suggestion: Allocate no more than 12-16GB memory when using HBase Region Servers.
Solr Server	CDH	0	0	(Optional Service) Suggestion: Minimum 1GB for Solr server. More might be necessary depending on index sizes.
Kudu Server	CDH	0	0	(Optional Service) Suggestion: Minimum 1GB for Kudu Tablet server. More might be necessary depending on data sizes.
Available Container Resources		44	250880	

Container resources				
Physical Cores to Vcores Multiplier		1		Set this ratio based on the expected number of concurrent threads in a container per thread core. Default is 1.
YARN Available Vcores		44		This value will be used in STEP 4 for YARN Configuration
YARN Available Memory			250880	This value will be used in STEP 4 for YARN Configuration

Start with at least 8 GB for your operating system, and 1 GB for Cloudera Manager. If services outside of Cloudera Runtime require additional resources, add those numbers under Other Services.

The HDFS DataNode uses a minimum of 1 core and about 1 GB of memory. The same requirements apply to the YARN NodeManager.

The spreadsheet lists several optional services:

- Impala daemon requires at least 16 GB for the daemon.
- HBase Region Servers requires 12-16 GB of memory.
- Solr server requires a minimum of 1 GB of memory.
- Kudu Tablet server requires a minimum of 1 GB of memory.

Any remaining resources are available for YARN applications (Spark and MapReduce). In this example, 44 CPU cores are available. Set the multiplier for vcores you want on each physical core to calculate the total available vcores.

Step 3: Cluster size

Having defined the specifications for each host in your cluster, enter the number of worker hosts needed to support your business case.

To see the benefits of parallel computing, set the number of hosts to a minimum of 10.

STEP 3: Cluster Size						
Enter the number of nodes you have (or expect to have) in the cluster						
			Quantity			
Number of Worker Hosts in the cluster			10			

Steps 4 and 5: Verify settings

Verify the memory and vcore settings.

Step 4 pulls forward the memory and vcore numbers from step 2. Step 5 shows the total memory and vcores for the cluster.

STEP 4: YARN Configuration on Cluster		
These are the first set of configuration values for your cluster. You can set these values in YARN->Configuration		
YARN NodeManager Configuration Properties	Value	Note
yarn.nodemanager.resource.cpu-vcores	44	Copied from STEP 2 "Available Resources"
yarn.nodemanager.resource.memory-mb	250880	Copied from STEP 2 "Available Resources"

STEP 5: Verify YARN Settings on Cluster		
Go to the Resource Manager Web UI (usually http://<ResourceManagerIP>:8088/ and verify the "Memory Total" and "Vcores Total" matches the values above. If your machine has no bad nodes, then the numbers should match exactly.		
Resource Manager Property to Check	Value	Note
Expected Value for "Vcores Total"	440	Calculated from STEP 2 "YARN Available Vcores" and STEP 3
Expected Value for "Memory Total" (in GB)	2450	Calculated from STEP 2 "YARN Available Memory" and STEP 3

Step 6: Verify container settings on cluster

You can change the values that impact the size of your containers.

The minimum number of vcores should be 1. When additional vcores are required, adding 1 at a time should result in the most efficient allocation. Set the maximum number of vcore reservations to the size of the node.

Set the minimum and maximum reservations for memory. The increment should be the smallest amount that can impact performance. Here, the minimum is approximately 1 GB, the maximum is approximately 8 GB, and the increment is 512 MB.

STEP 6: Verify Container Settings on Cluster		
In order to have YARN jobs run cleanly, you need to configure the container properties.		
YARN Container Configuration Properties (Vcores)	Value	Description
yarn.scheduler.minimum-allocation-vcores	1	Minimum vcore reservation for a container
yarn.scheduler.maximum-allocation-vcores	44	Maximum vcore reservation for a container
yarn.scheduler.increment-allocation-vcores	1	Vcore allocations must be a multiple of this value
YARN Container Configuration Properties (Memory)	Value	Description
yarn.scheduler.minimum-allocation-mb	1024	Minimum memory reservation for a container in MegaByte
yarn.scheduler.maximum-allocation-mb	250880	Maximum memory reservation for a container in MegaByte
yarn.scheduler.increment-allocation-mb	512	Memory allocations must be a multiple of this value in MegaByte

Step 6A: Cluster container capacity

Validate the minimum and maximum number of containers in your cluster, based on the numbers you entered

Step 6A: Cluster Container Capacity		
This section will tell you the capacity of your cluster (in terms of containers).		
Cluster Container Estimates	Minimum	Maximum
Max possible number of containers, based on memory configuration		2450
Max possible number of containers, based on vcore configuration		440
Container number based on 2 containers per disk spindles		480
Min possible number of containers, based on memory configuration	10	
Min possible number of containers, based on vcore configuration	10	

Step 6B: Container parameters checking

See whether you have over-allocated resources.

STEP 6B: Container Sanity Checking		
This section will do some basic checking of your container parameters in STEP 6 against the hosts.		
	Check	
Sanity Check	Status	Description
Scheduler maximum vcores must be larger than minimum	GOOD	yarn.scheduler.maximum-allocation-vcores >= yarn.scheduler.minimum-allocation-vcores
Scheduler maximum allocation MB must be larger than minimum	GOOD	yarn.scheduler.maximum-allocation-mb >= yarn.scheduler.minimum-allocation-mb
Scheduler minimum vcores must be greater than or equal to 0	GOOD	yarn.scheduler.minimum-allocation-vcores >= 0
Scheduler maximum vcores must be greater than or equal to 1	GOOD	yarn.scheduler.maximum-allocation-vcores >= 1
Host vcores must be larger than scheduler minimum vcores	GOOD	yarn.nodemanager.resource.cpu-vcores >= yarn.scheduler.minimum-allocation-vcores
Host vcores must be larger than scheduler maximum vcores	GOOD	yarn.nodemanager.resource.cpu-vcores >= yarn.scheduler.maximum-allocation-vcores
Host allocation MB must be larger than scheduler minimum	GOOD	yarn.nodemanager.resource.memory-mb >= yarn.scheduler.maximum-allocation-mb
Host allocation MB must be larger than scheduler maximum vcores	GOOD	yarn.nodemanager.resource.memory-mb >= yarn.scheduler.minimum-allocation-mb
Small container limit	GOOD	If yarn.scheduler.minimum-allocation-mb is less than 1GB, containers will likely get killed by YARN due to OutOfMemory issues

Step 7: MapReduce configuration

You can increase the memory allocation for the ApplicationMaster, map tasks, and reduce tasks.

The minimum vcore allocation for any task is always 1. The Spill/Sort memory allocation of 400 should be sufficient, and should be (rarely) increased if you determine that frequent spills to disk are hurting job performance.

The common MapReduce parameters mapreduce.map.java.opts, mapreduce.reduce.java.opts, and yarn.app.mapreduce.am.command-opts are configured for you automatically based on the *HEAP TO CONTAINER SIZE RATIO*.

STEP 7: MapReduce Configuration

For CDH 5.5 and later we recommend that only the heap or the container size is specified for map and reduce tasks. The value that is not specified will be calculated based on the setting mapreduce.job.heap.memory-mb.ratio. This calculation follows Cloudera Manager and calculates the heap size based on the ratio and the container size.

Application Master Configuration properties		Value	Description
yarn.app.mapreduce.am.resource.cpu-vcores		1	AM container vcore reservation
yarn.app.mapreduce.am.resource.mb		1024	AM container memory reservation in MegaByte
yarn.app.mapreduce.am.command-opts	-Xmx	800	AM Java heap size in MegaByte
Task auto heap sizing			
Use task auto heap sizing		yes	
mapreduce.job.heap.memory-mb.ratio		0.8	Ratio between the container size and task heap size.
Map Task Configuration properties			
mapreduce.map.cpu.vcores		1	Map task vcore reservation
mapreduce.map.memory.mb		1024	Map task memory reservation in MegaByte
mapreduce.map.java.opts	ignored	800	Map task Java heap size in MegaByte
mapreduce.task.io.sort.mb		400	Spill/Sort memory reservation
ReduceTask Configuration properties			
mapreduce.reduce.cpu.vcores		1	Reduce task vcore reservation
mapreduce.reduce.memory.mb		1024	Reduce task memory reservation in MegaByte
mapreduce.reduce.java.opts	ignored	800	Reduce Task Java heap size in MegaByte

Step 7A: MapReduce settings checking

Verify at a glance that all of your minimum and maximum resource allocations are within the parameters you set.

STEP 7A: MapReduce Sanity Checking

Sanity check MapReduce settings against container minimum/maximum properties.

Application Master Sanity Checks	Value	Description
AM vcore request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-vcores <= **yarn.app.mapreduce.am.resource.cpu-vcores** <= yarn-scheduler.maximum-allocation-vcores
AM memory request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-mb <= **yarn.app.mapreduce.am.resource.cpu-vcores** <= yarn.scheduler.maximum-allocation-mb
Container size must large enough for java heap and overhead	GOOD	Java Heap should be between 75% and 90% of the container size: too low wastes resources, to high could lead to OOM
Ratio should be between 0.75 and 0.9	GOOD	Java Heap should be between 75% and 90% of the container size: too low wastes resources, to high could lead to OOM

Map Task Sanity Checks	Value	Description
Map task vcore request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-vcores <= **mapreduce.map.cpu.vcores** <= yarn-scheduler.maximum-allocation-vcores
Map task memory request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-mb <= **mapreduce.map.memory.mb** <= yarn.scheduler.maximum-allocation-mb
Container size must large enough for java heap and overhead	N/A	Java Heap should be between 75% and 90% of the container size: too low wastes resources, to high could lead to OOM
Spill/Sort memory should not use whole map task heap	GOOD	Make sure that Spill/Sort memory reservation uses between 40% and 60% of the heap of a map task.

Reduce Task Sanity Checks	Value	Description
Reduce task vcore request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-vcores <= **mapreduce.reduce.cpu.vcores** <= yarn-scheduler.maximum-allocation-vcores
Reduce task memory request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-mb <= **mapreduce.reduce.memory.mb** <= yarn.scheduler.maximum-allocation-mb
Container size must large enough for java heap and overhead	N/A	Java Heap should be between 75% and 90% of the container size: too low wastes resources, to high could lead to OOM

Set properties in Cloudera Manager

When you are satisfied with the cluster configuration estimates, use the values in the spreadsheet to set the corresponding properties in Cloudera Manager

Table 1: Cloudera Manager Property Correspondence

Step	YARN/MapReduce Property	Cloudera Manager Equivalent
4	yarn.nodemanager.resource.cpu-vcores	Container Virtual CPU Cores
4	yarn.nodemanager.resource.memory-mb	Container Memory
6	yarn.scheduler.minimum-allocation-vcores	Container Virtual CPU Cores Minimum
6	yarn.scheduler.maximum-allocation-vcores	Container Virtual CPU Cores Maximum
6	yarn.scheduler.increment-allocation-vcores	Container Virtual CPU Cores Increment
6	yarn.scheduler.minimum-allocation-mb	Container Memory Minimum
6	yarn.scheduler.maximum-allocation-mb	Container Memory Maximum
6	yarn.scheduler.increment-allocation-mb	Container Memory Increment
7	yarn.app.mapreduce.am.resource.cpu-vcores	ApplicationMaster Virtual CPU Cores
7	yarn.app.mapreduce.am.resource.mb	ApplicationMaster Memory
7	mapreduce.map.cpu.vcores	Map Task CPU Virtual Cores
7	mapreduce.map.memory.mb	Map Task Memory
7	mapreduce.reduce.cpu.vcores	Reduce Task CPU Virtual Cores
7	mapreduce.reduce.memory.mb	Reduce Task Memory
7	mapreduce.task.io.sort.mb	I/O Sort Memory

Configure memory settings

The memory configuration for YARN and MapReduce memory is important to get the best performance from your cluster.

Several different settings are involved. The table below shows the default settings, as well as the settings that Cloudera recommends, for each configuration option.

Table 2: YARN and MapReduce Memory Configuration

Cloudera Manager Property Name	Cloudera Runtime Property Name	Default Configuration	Cloudera Tuning Guidelines
Container Memory Minimum	yarn.scheduler.minimum-allocation-mb	1 GB	0
Container Memory Maximum	yarn.scheduler.maximum-allocation-mb	64 GB	amount of memory on largest host
Container Memory Increment	yarn.scheduler.increment-allocation-mb	512 MB	Use a fairly large value, such as 128 MB
Container Memory	yarn.nodemanager.resource.memory-mb	8 GB	8 GB
Map Task Memory	mapreduce.map.memory.mb	1 GB	1 GB
Reduce Task Memory	mapreduce.reduce.memory.mb	1 GB	1 GB
Map Task Java Opts Base	mapreduce.map.java.opts	-Djava.net.preferIPv4Stack=true	-Djava.net.preferIPv4Stack=true -Xmx768m
Reduce Task Java Opts Base	mapreduce.reduce.java.opts	-Djava.net.preferIPv4Stack=true	-Djava.net.preferIPv4Stack=true -Xmx768m
ApplicationMaster Memory	yarn.app.mapreduce.am.resource.mb	1 GB	1 GB

Cloudera Manager Property Name	Cloudera Runtime Property Name	Default Configuration	Cloudera Tuning Guidelines
ApplicationMaster Java Opts Base	yarn.app.mapreduce.am.command-opt	-Djava.net.preferIPv4Stack=true	-Djava.net.preferIPv4Stack=true -Xmx768m

YARN Configuration Properties

This table provides information about the parameters listed in the yarn-site.xml file.

Parameter	Value
hadoop.registry.zk.quorum	c2185-node3.coelab.root.hwx.site:2181,c2185-node1.coelab.root.hwx.site:2181,c2185-node2.coelab.root.hwx.site:2181
yarn.acl.enable	true
yarn.admin.acl	yarn,hive,hdfs,mapred
yarn.am.liveness-monitor.expiry-interval-ms	600000
yarn.application.classpath	$HADOOP_CLIENT_CONF_DIR,$HADOOP_COMMON_HOME/*,$HADOOP_COMMON_HOME/lib/*,$HADOOP_HDFS_HOME/*,$HADOOP_HDFS_HOME/lib/*,$HADOOP_YARN_HOME/*,$HADOOP_YARN_HOME/lib/*
yarn.authorization-provider	org.apache.ranger.authorization.yarn.authorizer.RangerYarnAuthorizer
yarn.client.failover-sleep-base-ms	100
yarn.client.failover-sleep-max-ms	2000
yarn.cluster.scaling.recommendation.enable	true
yarn.http.policy	HTTPS_ONLY
yarn.http.rmwebapp.custom.unwrapped.dao.classes	com.cloudera.cloud.yarn.resourcemanager.webapp.dao.ClusterScalingInfo,com.cloudera.c
yarn.log-aggregation-enable	true
yarn.log-aggregation-status.time-out.ms	600000
yarn.log-aggregation.TFile.remote-app-log-dir-suffix	
yarn.log-aggregation.file-controller.IFile.class	org.apache.hadoop.yarn.logaggregation.filecontroller.ifile.LogAggregationIndexedFileCon
yarn.log-aggregation.file-controller.TFile.class	org.apache.hadoop.yarn.logaggregation.filecontroller.tfile.LogAggregationTFileController
yarn.log-aggregation.file-formats	IFile,TFile
yarn.log-aggregation.retain-seconds	604800
yarn.nm.liveness-monitor.expiry-interval-ms	600000
yarn.node-labels.enabled	true
yarn.node-labels.fs-store.root-dir	/yarn/node-labels
yarn.resourcemanager.address.rm1546335021	coelab.int.cldr.work:8032
yarn.resourcemanager.address.rm1546335074	coelab.int.cldr.work:8032
yarn.resourcemanager.admin.address.rm1546335021	coelab.int.cldr.work:8033
yarn.resourcemanager.admin.address.rm1546335074	coelab.int.cldr.work:8033
yarn.resourcemanager.admin.client.thread-count	1
yarn.resourcemanager.am.max-attempts	2
yarn.resourcemanager.am.placement-preference-with-node-attributes	ORDER NODES IN NodeInstanceType WITH worker > compute
yarn.resourcemanager.amliveliness-monitor.interval-ms	1000

Parameter	Value
yarn.resourcemanager.autoscaling.plugin-type	coarse
yarn.resourcemanager.client.thread-count	50
yarn.resourcemanager.cluster-id	yarnRM
yarn.resourcemanager.container.liveness-monitor.interval-ms	600000
yarn.resourcemanager.ha.automatic-failover.embedded	true
yarn.resourcemanager.ha.automatic-failover.enabled	true
yarn.resourcemanager.ha.enabled	true
yarn.resourcemanager.ha.id	rm1546335074
yarn.resourcemanager.ha.rm-ids	rm1546335074,rm1546335021
yarn.resourcemanager.keytab	yarn.keytab
yarn.resourcemanager.max-completed-applications	10000
yarn.resourcemanager.nm.liveness-monitor.interval-ms	1000
yarn.resourcemanager.nodes.exclude-path	{{CMF_CONF_DIR}}/nodes_exclude.txt
yarn.resourcemanager.nodes.include-path	{{CMF_CONF_DIR}}/nodes_allow.txt
yarn.resourcemanager.non-am.placement-preference-with-node-attributes	ORDER NODES IN NodeInstanceType WITH compute > worker
yarn.resourcemanager.placement-constraints.handler	scheduler
yarn.resourcemanager.principal	yarn/_HOST@YARN-Y4C.L2OV-M7VS.INT.CLDR.WORK
yarn.resourcemanager.proxy-user-privileges.enabled	true
yarn.resourcemanager.recovery.enabled	true
yarn.resourcemanager.resource-tracker.address.rm1546335021	coelab.cldr.work:8031
yarn.resourcemanager.resource-tracker.address.rm1546335074	coelab.int.cldr.work:8031
yarn.resourcemanager.resource-tracker.client.thread-count	50
yarn.resourcemanager.scheduler.address.rm1546335021	coelab.cldr.work:8030
yarn.resourcemanager.scheduler.address.rm1546335074	coelab.int.cldr.work:8030
yarn.resourcemanager.scheduler.class	org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler
yarn.resourcemanager.scheduler.client.thread-count	50
yarn.resourcemanager.scheduler.monitor.enable	true
yarn.resourcemanager.scheduler.monitor.policies	org.apache.hadoop.yarn.server.resourcemanager.monitor.capacity.ProportionalCapacityPre
yarn.resourcemanager.store.class	org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore
yarn.resourcemanager.webapp.address.rm1546335021	yarn-zfv9wy-master1.yarn-y4c.l2ov-m7vs.int.cldr.work:8088
yarn.resourcemanager.webapp.address.rm1546335074	yarn-zfv9wy-master0.yarn-y4c.l2ov-m7vs.int.cldr.work:8088
yarn.resourcemanager.webapp.cross-origin.enabled	true
yarn.resourcemanager.webapp.https.address.rm1546335021	coelab.cldr.work:8090
yarn.resourcemanager.webapp.https.address.rm1546335074	coelab.int.cldr.work:8090
yarn.resourcemanager.webapp.spnego-keytab-file	yarn.keytab
yarn.resourcemanager.webapp.spnego-principal	HTTP/_HOST@YARN-Y4C.L2OV-M7VS.INT.CLDR.WORK
yarn.resourcemanager.work-preserving-recovery.enabled	true
yarn.resourcemanager.zk-acl	sasl:yarn:cdrwa

Parameter	Value
yarn.resourcemanager.zk-address	coelab.int.cldr.work:2181,coelab.int.cldr.work:2181,coelab-master0.yarn-y4c.l2cldr.work:2181
yarn.resourcemanager.zk-timeout-ms	60000
yarn.scheduler.capacity.multi-node-placement-enabled	true
yarn.scheduler.capacity.multi-node-sorting.policy	instancetype-based
yarn.scheduler.capacity.multi-node-sorting.policy.instancetype-based.class	com.cloudera.cloud.yarn.resourcemanager.scheduler.NodeInstanceTypeBinPackingPolicy
yarn.scheduler.capacity.multi-node-sorting.policy.names	instancetype-based
yarn.scheduler.capacity.resource-calculator	org.apache.hadoop.yarn.util.resource.DefaultResourceCalculator
yarn.scheduler.configuration.store.class	zk
yarn.scheduler.increment-allocation-mb	512
yarn.scheduler.increment-allocation-vcores	1
yarn.scheduler.maximum-allocation-mb	450560
yarn.scheduler.maximum-allocation-vcores	32
yarn.scheduler.minimum-allocation-mb	11264
yarn.scheduler.minimum-allocation-vcores	1
yarn.service.classpath	$HADOOP_CLIENT_CONF_DIR
yarn.service.framework.path	/user/yarn/services/service-framework/7.2.13/service-dep.tar.gz
yarn.webapp.api-service.enable	true
yarn.webapp.custom.webservice.class	com.cloudera.cloud.yarn.resourcemanager.webapp.YarnOnCloudWebServices
yarn.webapp.filter-entity-list-by-user	true
yarn.webapp.ui2.enable	true

For information about the YARN configuration properties supported by Cloudera Manager, see *Cloudera Manager* documentation.

Use the YARN REST APIs to manage applications

You can use the YARN REST APIs to submit, monitor, and stop applications.

Important: In a non-secure cluster, you must append a request with ?user.name=<user>.

Example: Get application data

- Without ?user.name=<user>:

```
curl http://localhost:19888/jobhistory/job/job_1516861688424_0001
Access denied: User null does not have permission to view job job_
1516861688424_0001
```

- With ?user.name=<user>:

```
curl http://localhost:19888/jobhistory/job/job_1516861688424_0001?user.name=hrt_1
{"job":{"submitTime":1516863297896,"startTime":1516863310110,"finishTime":1516863330610,
"id":"job_1516861688424_0001","name":"Sleepjob","queue":"default","user":"hrt_1",
"state":"SUCCEEDED","mapsTotal":1,"mapsCompleted":1,"reducesTotal":1,"reducesCompleted":1,
"uberized":false,"diagnostics":"","avgMapTime":10387,"avgReduceTime":536,"avgShuffleTime":4727,
"avgMergeTime":27,"failedReduceAttempts":0,"killedReduceAttempts":0,"successfulReduceAttempts":1,
"failedMapAttempts":0,"killedMapAttempts":0,"successfulMapAttempts":1,"acls":[{"name":"mapreduce.j
ob.acl-
view-job","value":" "},{"name":"mapreduce.job.acl-modify-job","value":" "}]}}
```

Get an Application ID

You can use the New Application API to get an application ID, which can then be used to submit an application. For example:

```
curl -v -X POST 'http://localhost:8088/ws/v1/cluster/apps/new-application'
```

The response returns the application ID, and also includes the maximum resource capabilities available on the cluster. For example:

```
    {
    application-id: application_1409421698529_0012",
    "maximum-resource-capability":{"memory":"8192","vCores":"32"}
    }
```

Set Up an Application .json File

Before you submit an application, you must set up a .json file with the parameters required by the application. This is analogous to creating your own ApplicationMaster. The application .json file contains all of the fields you are required to submit in order to launch the application.

The following is an example of an application .json file:

```
 {
    "application-id":"application_1404203615263_0001",
    "application-name":"test",
    "am-container-spec":
    {
       "local-resources":
       {
          "entry":
         [
            {
               "key":"AppMaster.jar",
               "value":
               {
                  "resource":"hdfs://hdfs-namenode:9000/user/testuser/Dis
tributedShell/demo-app/AppMaster.jar",
                  "type":"FILE",
                  "visibility":"APPLICATION",
                  "size": "43004",
                  "timestamp": "1405452071209"
               }
            }
         ]
       },
       "commands":
       {
          "command":"{{JAVA_HOME}}/bin/java -Xmx10m org.apache.hadoop.yar
n.applications.distributedshell.ApplicationMaster --container_memory 10 --co
ntainer_vcores 1 --num_containers 1 --priority 0 1><LOG_DIR>/AppMaster.stdou
t 2><LOG_DIR>/AppMaster.stderr"
       },
       "environment":
       {
          "entry":
          [
             {
                "key": "DISTRIBUTEDSHELLSCRIPTTIMESTAMP",
                "value": "1405459400754"
```

```
             },
             {
                "key": "CLASSPATH",
                "value": "{{CLASSPATH}}<CPS>./*<CPS>{{HADOOP_CONF_DIR}}<C
PS>{{HADOOP_COMMON_HOME}}/share/hadoop/common/*<CPS>{{HADOOP_COMMON_HOME}}/s
hare/hadoop/common/lib/*<CPS>{{HADOOP_HDFS_HOME}}/share/hadoop/hdfs/*<CPS>{{
HADOOP_HDFS_HOME}}/share/hadoop/hdfs/lib/*<CPS>{{HADOOP_YARN_HOME}}/share/ha
doop/yarn/*<CPS>{{HADOOP_YARN_HOME}}/share/hadoop/yarn/lib/*<CPS>./log4j.pro
perties"
             },
             {
                "key": "DISTRIBUTEDSHELLSCRIPTLEN",
                "value": "6"
             },
             {
                "key": "DISTRIBUTEDSHELLSCRIPTLOCATION",
                "value": "hdfs://hdfs-namenode:9000/user/testuser/demo-app/
shellCommands"
             }
          ]
       }
    },
    "unmanaged-AM":"false",
    "max-app-attempts":"2",
    "resource":
    {
       "memory":"1024",
       "vCores":"1"
    },
    "application-type":"YARN",
    "keep-containers-across-application-attempts":"false"
  }
```

Submit an Application

You can use the Submit Application API to submit applications. For example:

```
curl -v -X POST -d @example-submit-app.json -H "Content-type: application/js
on"'http://localhost:8088/ws/v1/cluster/apps'
```

After you submit an application the response includes the following field:

```
HTTP/1.1 202 Accepted
```

The response also includes the Location field, which you can use to get the status of the application (app ID). The following is an example of a returned Location code:

```
Location: http://localhost:8088/ws/v1/cluster/apps/application_1409421698529
_0012
```

Monitor an Application

You can use the Application State API to query the application state. To return only the state of a running application, use the following command format:

```
curl 'http://localhost:8088/ws/v1/cluster/apps/application_1409421698529_001
2/state'
```

You can also use the value of the Location field (returned in the application submission response) to check the application status. For example:

```
curl -v 'http://localhost:8088/ws/v1/cluster/apps/application_1409421698529_
0012'
```

You can use the following command format to check the logs:

```
yarn logs -appOwner 'dr.who' -applicationId application_1409421698529_0012 |
 less
```

Kill an Application

You can also use the Application State API to end an application by using a PUT operation to set the application state to KILLED. For example:

```
curl -v -X PUT -H 'Accept: application/json' -H 'Content-Type: application/j
son' -d '{"state": "KILLED"}' 'http://localhost:8088/ws/v1/cluster/apps/appl
ication_1409421698529_0012/state'
```

Comparison of Fair Scheduler with Capacity Scheduler

This section provides information about choosing Capacity Scheduler, its benefits, and performance improvements along with features comparison between Fair Scheduler and Capacity Scheduler.

Why one scheduler?

Cloudera only supports the Capacity Scheduler in the YARN clusters.

Prior to its launch, Cloudera customers used one of the two schedulers (Fair Scheduler and Capacity Scheduler) depending on which product they were using (CDH or HDP respectively).

The choice to converge to one scheduler in Cloudera was a hard one but ultimately rooted in our intention to reduce complexity for our customers and at the same time help focus our future investments. Over the years, both the schedulers have evolved greatly, to the point that Fair Scheduler borrowed almost all of the features from Capacity Scheduler and vice-versa. Given this, we ultimately decided to put our weight behind Capacity Scheduler for all your YARN clusters.

Those clusters that currently use the Fair scheduler must migrate to the Capacity Scheduler when moving to Cloudera. Cloudera provides tools, documentation, and related help for such migrations.

Benefits of Using Capacity Scheduler

The following are some of the benefits when using Capacity Scheduler:

- Integration with Ranger
- Node partitioning/labeling
- Improvements to schedule on cloud-native environments, such as better bin-packing, autoscaling support, and so on.
- Scheduling throughput improvements
  - Global Scheduling Framework
  - Lookup of multiple nodes at one time

Fore more details about Scheduling throughput improvements, see Scheduler Performance Improvements.

- Affinity/anti-affinity: run application X only on those nodes which run application Y and the other way around. Do not run application X and application Y on the same node.

For information about the currently supported features, see Supported Features.

Scheduler performance improvements

Provides information about Global scheduling feature and its test results.

Improvements brought by Global Scheduling Improvements (YARN-5139)

Before the changes of global scheduling, the YARN scheduler was under a monolithic lock, which was underperforming. Global scheduling largely improved the internal locking structure and the thread-model of how the YARN scheduler works. The scheduler can now decouple placement decisions and change the internal data structure. This can also enable to lookup multiple nodes at a time, which is used by auto-scaling and bin-packing policies on cloud. For more information, see the design and implementation notes.

Based on the simulation, the test result of using Global Scheduling feature shows:

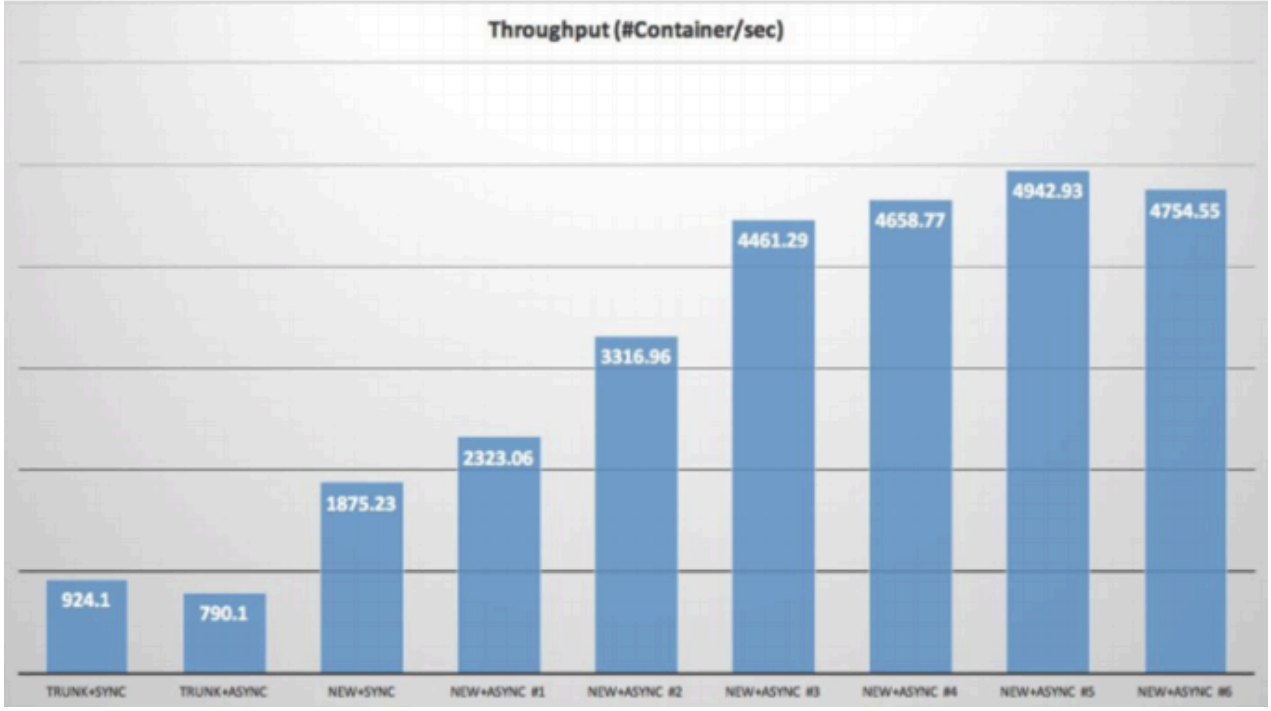

This is a simulated environment which has 20000 nodes and 47000 running applications. For more information about these tests, see the performance report.

Performance test from YARN community

Microsoft published Hydra: a federated resource manager for data-center scale analytics (Carlo, et al) report which highlights the scalability (Deployed YARN to more than 250k nodes, which includes five large federated cluster, each of them having 50k nodes) and scheduling performance (each cluster's scheduler can make more than 40k container allocation per second) by using Capacity Scheduler. This is the largest known YARN deployment in the world.

We also saw performance numbers from other companies in the community in line with what we have tested using simulators (thousands of container allocations per second for a cluster that has thousands of nodes).

Disclaimer: The performance number discussed above is related to the size of the cluster, workloads running on the cluster, queue structure, healthiness (such as node manager, disk, and network), container churns, and so on. This typically needs fine-tuning efforts for the scheduler and other cluster parameters to reach the ideal performance. This is NOT a guaranteed number which can be achieved just by using Cloudera.

Feature comparison

The features of both schedulers have become similar over time. The current feature list and differences between the two schedulers is listed in the tables.

Supported Features

Feature List		Capacity Scheduler	Fair Scheduler	Comments
Queues	Hierarchical Queues	yes	yes	
	Elastic Queue Capacity for better resource sharing	yes	yes	
	Percentage Based Resource Configuration in Queues	yes	yes	Percentages and absolute resources settings cannot be used simultaneous.
	Auto Queue Creation	yes	yes	
	User Mapping (user/group to queue mapping)	yes	yes	
	CLI/REST API support to manage queues	yes	yes	
	Move applications between queues	yes	yes	
	Dynamic Queue creation/ deletion/modification	yes	yes	
	Reservation support in queues	yes	yes	
Authorization	Authorization control (ACLs in Queues for submit/manage/admin)	yes	yes	
	Third party ACL control (Ranger)	yes	yes	
Application Placement	Node Labels support	yes	no	
	Hive placement integration	yes	yes	
	Node Attributes support	yes	no	
	Placement constraints support	yes	no	Supported constraints are limited in the current implementation.
	Node Locality	yes	yes	
	Locality Delay control	yes	yes	
	User limit quota management	yes	yes	
	AM resource quota management	yes	yes	
	Queue Priority	yes	no	Indirectly managed through queue weights.
	Maximum and Minimum allocation limit per container unit	yes	yes	
Scheduling	Asynchronous scheduling support	yes	yes	Implementation differs between the schedulers and should not be treated as equivalent.

	Multiple resource types support (CPU, Memory, GPU, and so on)	yes	yes	
	Queue Ordering Policies (Fair, FIFO, and so on)	yes	yes	
	Multiple container assignments per heartbeat	yes	yes	
Preemption	Inter Queue preemption support	yes	yes	
	Intra Queue preemption support	yes	yes	
	Reservation based preemption	yes	yes	
	Queue Priority based preemption	yes	no	Queue weights are taken into account when preempting decisions are made.
Application Support	First class Concept of application	yes	yes	
	Application priority	yes	yes	
	Application timeout	yes	yes	
	Moving Application across queues	yes	yes	
	High Availability stateful application recovery	yes	yes	

Roadmap Features

Feature List		Capacity Scheduler	Fair Scheduler	Comments
Queues	Absolute Resource Configuration in Queues	yes	yes	Percentages and absolute resources settings cannot be usedsimultaneously.
Application Placement	Maximum number of applications	no	yes	Indirectly managed through AM resource quotas.
Scheduling	Application Size Based Fairness	no	yes	

Migration from Fair Scheduler to Capacity Scheduler

Starting from the Cloudera Cloudera Private Cloud Base 7.1 release, Cloudera provides the `fs2cs` conversion utility, which is a CLI application and is a part of the YARN CLI command. This utility helps to migrate from Fair Scheduler to Capacity Scheduler.

For information about using the `fs2cs` conversion utility, the scheduler conversion process, and manual configurations, see Migrating Fair Scheduler to Capacity Scheduler.

Configuring and using Queue Manager REST API

You can use the Queue Manager REST APIs to automate the Queue Manager operations using the HTTP endpoints.

Limitations

These Queue Manager REST APIs are recommended to be used only for administrative purposes, and are not supported in heavily concurrent use cases.

Using the REST API

The Queue Manager REST server exposes endpoints that provide the start, stop, add, delete, change queue capacity, and change queue properties operations. For a given endpoint, the HTTP verb controls the type of operation.

Prerequisites

You must first authenticate and obtain the cookies to send a request.

- Authenticate and obtain cookie
  - Log in to Cloudera Manager and get the cookie to send a request. This generates a session cookie that must be provided with every request. If you are using an HTTP client such as Postman, you can copy the cookie value from your browser and add it as a header to your requests. You can also authenticate and obtain the cookie using the command line using cURL, as follows:

    ```
    curl -i -k -v -c cookies.txt -u admin:admin
    http://<cm_host>:7180/api/v41/clusters
    ```

  - This command saves the resulting cookie in cookies.txt. You can use the cookie to make the requests. Run the following command to make the initial validation request.

    ```
    curl -v -k -b cookies.txt -u <username>:<password> -H 'Content-Type: app
    lication/json' -X POST -H 'referer:
    http://<cm_host>:7180/cmf/clusters/Cluster%201/queue-manager/'
    http://<cm_host>:7180/cmf/clusters/Cluster%201/queue-manager-api/api/
    v1/environments/dev/clusters/Cluster%201/resources/scheduler/partitions/
    default/queues/root.default -d '<request_body>'
    ```

- Secure cluster (SSL and Kerberos enabled)

  If you are using a cluster secured with SSL and Kerberos, obtain the required certificates.

  1. Create a directory on your machine to copy the certificates from the Queue Manager node. The following command creates the apiCerts directory.

     ```
     mkdir apiCerts
     ```

  2. SSH to your Queue Manager node to obtain the WebApp folder information.

     ```
     cd /var/run/cloudera-scm-agent/process/
     ```

  3. Copy the certificates from the Queue Manager node to the “apiCerts” directory.

     ```
     scp -i <your_keypair>
     root@<qm_host>:/var/run/cloudera-scm-agent/process/{12}-queuemanager-Q
     UEUEMANAGER_WEBAPP/cm-auto-host_cert_chain.pem apiCerts/
     ```

  4. Copy the following certificates from the Queue Manager node.

     ```
     scp -i <your_keypair>
     root@<qm_host>:/var/run/cloudera-scm-agent/process/12-queuemanager-QU
     EUEMANAGER_WEBAPP/cm-auto-global_cacerts.pem apiCerts/
     ```

```
scp -i <your_keypair>
root@<qm_host>:/var/run/cloudera-scm-agent/process/12-queuemanager-QU
EUEMANAGER_WEBAPP/cm-auto-host_cert_chain.pem apiCerts/
scp -i <your_keypair>
root@<qm_host>:/var/run/cloudera-scm-agent/process/12-queuemanager-QUEUE
MANAGER_WEBAPP/cm-auto-host_key.pem apiCerts/

scp -i <your_keypair>
root@<qm_host>:/var/run/cloudera-scm-agent/process/12-queuemanager-QUE
UEMANAGER_WEBAPP/cm-auto-host_key.pw apiCerts/
```

5. Decrypt the certificate so that you can use it later in the cURL call.

```
openssl rsa -passin file:cm-auto-host_key.pw -in cm-auto-host_key.pem -o
ut
cm-auto-host_key_decrypted.pem
chmod 644 cm-auto-host_key_decrypted.pem
```

6. Save the cookie for the API calls.

```
curl --key cm-auto-host_key_decrypted.pem --cert cm-auto-host_cert_chain
.pem --cacert cm-auto-global_cacerts.pem -v -c cookies_ssl.txt -u admin:
admin https://<cm_host>:7183/api/v41/clusters
```

7. Make the initial validation request.

```
curl --key cm-auto-host_key_decrypted.pem --cert cm-auto-host_cert_chain
.pem --cacert cm-auto-global_cacerts.pem -v -b cookies_ssl.txt -u admin:
admin -H 'Content-Type: application/json' -X POST -H 'referer: https://<
cm_host>:7183/cmf/clusters/Cluster%201/queue-manager/' https://<cm_host>
:7183/cmf/clusters/Cluster%201/queue-manager-api/api/v1/environments/dev
/clusters/Cluster%201/resources/scheduler/partitions/default/queues/root
.default\?validate=true\&startsafemode=60 -d '<request_body>
```

Start Queue

Run the following to start a queue.

Request URL

```
PUT
http://<cm_host>:7180/cmf/clusters/<Cluster-id>/queue-manager-api/api/v1/env
ironments/support/clusters/<Cluster-id>/resources/scheduler/partitions/defau
lt/queues/<path-to-the-queue>
```

Request Body

```
{
  "properties": [{ "name": "state", "value": "RUNNING" }],
  "message": "Started <path-to-the-queue>"
}
```

Note: In the Queue Manager UI, hover on the queue name to know the path of the queue.

Example Request

```
curl -v -X PUT -b cookie.txt --key cm-auto-host_key_decrypted.pem --cert cm-
auto-host_cert_chain.pem --cacert cm-auto-global_cacerts.pem -H "Referer: ht
tps://<cm_host>:7183/cmf/clusters/Cluster%201/queue-manager/"
```

```
my-cluster-1.my-cluster.root.hwx.site:7180/cmf/clusters/Cluster201/queue-man
ager-api/api/v1/environments/support/clusters/Cluster201/resources/scheduler
/partitions/default/queues/root.samplequeue -d @request.json
```

The above example starts the samplequeue queue placed under the root queue.

Note: You must save the Request Body content as a .json file before using this command. In the above example, the Request Body content is saved as the *REQUEST.JSON* file.

Parameter	Description
cm_host	Cloudera Manager host address
Cluster-id	Name of the cluster
path-to-the-queue	Path to the queue with the queue name

Stop Queue

Run the following to stop a queue.

Request URL

```
PUT
http://<CM_HOST>:7180/cmf/clusters/<CLUSTER-ID>/queue-manager-api/api/v1/en
vironments/support/clusters/<CLUSTER-ID>/resources/scheduler/partitions/defa
ult/queues/<PATH-TO-THE-QUEUE>
```

Request Body

```
{
  "properties": [{ "name": "state", "value": "STOPPED" }],
  "message": "Stopped <PATH-TO-THE-QUEUE>"
}
```

Note: In the Queue Manager UI, hover on the queue name to know the path of the queue.

Example Request

```
curl -v -X PUT -b cookie.txt --key cm-auto-host_key_decrypted.pem --cert cm-
auto-host_cert_chain.pem --cacert cm-auto-global_cacerts.pem -H "Referer: ht
tps://<cm_host>:7183/cmf/clusters/Cluster%201/queue-manager/"

my-cluster-1.my-cluster.root.hwx.site:7180/cmf/clusters/Cluster201/queue-man
ager-api/api/v1/environments/support/clusters/Cluster201/resources/scheduler
/partitions/default/queues/root.samplequeue -d @request.json
```

The above example stops the `samplequeue` queue placed under the root queue.

Note: You must save the Request Body content as a .json file before using this command. In the above example, the Request Body content is saved as the *REQUEST.JSON* file.

Parameter	Description
cm_host	Cloudera Manager host address
Cluster-id	Name of the cluster
path-to-the-queue	Path to the queue with the queue name

Add Queue

Run the following to add a queue.

Request URL

```
POST
http://<CM_HOST>:7180/cmf/clusters/<CLUSTER-ID>/queue-manager-api/api/v1/en
vironments/support/clusters/<CLUSTER-ID>/resources/scheduler/partitions/defa
ult/queues/<PATH-TO-THE-QUEUE>
```

Request Body

```
{
  "queuePath": "{queue-path}",
  "queueName": "{queue-name}",
  "capacity": "{queue-capacity}",
  "maximum-capacity": "{queue-maximum-capacity}",
  "siblingCapacities": [
    {
      "queuePath": "{sibling-queue-path}",
      "queueName": "{sibling-queue-name}",
      "capacity": "{sibling-queue-capacity}",
      "maximum-capacity": "{sibling-queue-maximum-capacity}",
    }
  ],
  "message": "Added {queue-name}"
}
```

Note: The siblingCapacities parameter is required only if you are adding a sibling to an existing leaf queue. You must ensure that the total capacity of the sibling queues and the new queue must be equal to 100% of the parent queue in the Relative mode. The total allocated capacity of the sibling queues and the new queue must be less than or equal to the capacity of the parent in the Absolute mode. For information about resource allocation, see "change link" Resource allocation overview.

Parameter	Description
cm_host	Cloudera Manager host address
Cluster-id	Name of the cluster
queuePath	Path to the queue
queueName	Name of the queue
capacity	Capacity of the queue
maximum-capacity	Maximum capacity of the queue
siblingCapacities	Capacity of the sibling queue

Example Request

```
curl -v -X POST -b cookie.txt --key cm-auto-host_key_decrypted.pem --cert cm
-auto-host_cert_chain.pem --cacert cm-auto-global_cacerts.pem -H "Referer: h
ttps://<cm_host>:7183/cmf/clusters/Cluster%201/queue-manager/"
my-cluster-1.my-cluster.root.hwx.site:7180/cmf/clusters/Cluster201/queue-ma
nager-api/api/v1/environments/support/clusters/Cluster201/resources/schedule
r/partitions/default/queues/root.samplequeue -d @request.json
```

Note: You must save the Request Body content as a .json file before using this command. In the above example, the Request Body content is saved as the *REQUEST.JSON* file.

Change Queue Capacities

Run the following to change the queue properties.

Request URL

```
PUT
http://<CM_HOST>:7180/cmf/clusters/<CLUSTER-ID>/queue-manager-api/api/v1/en
vironments/support/clusters/<CLUSTER-ID>/resources/scheduler/partitions/defa
ult/queues/<PATH-TO-THE-QUEUE>
```

Request Body

```
{
  "properties": [
    { "name": "capacity", "value": "0" },
    { "name": "maximum-capacity", "value": "100" }
      "accessible-node-labels.<NAME-OF-THE-PARTITION>.capacity":"<QUEUE-
CAPACITY>",
      "accessible-node-labels.<NAME-OF-THE-PARTITION>.maximum-capacity":"
100",
      "accessible-node-labels.<NAME-OF-THE-PARTITION>.capacity":"<QUEUE-
CAPACITY>",
      "accessible-node-labels.<NAME-OF-THE-PARTITION>.maximum-capacity":"
100"
  ],
  "siblingCapacities": [
    {
      "queuePath": "root.default",
      "queueName": "default",
      "capacity": "100",
      "maximum-capacity": "100"
      "accessible-node-labels.<NAME-OF-THE-PARTITION>.capacity":"<QUEUE-
CAPACITY>",
      "accessible-node-labels.<NAME-OF-THE-PARTITION>.maximum-capacity":"
100",
      "accessible-node-labels.<NAME-OF-THE-PARTITION>.capacity":"<QUEUE-
CAPACITY>",
      "accessible-node-labels.<NAME-OF-THE-PARTITION>.maximum-capacity":"
100"
    }
  ],
  "message": "Changed child capacities for root"
}
```

Important:

- You can specify the capacity for a particular partition using the accessible-node-labels.<name-of-the-partition>.capacity option. The total capacity of all the queues in a given partition must be equal to 100% of the parent queue in the Relative mode. The total allocated capacity of all the queues and the must be less than or equal to the capacity of the parent in the Absolute mode.
- YARN Queue Manager does not support mixed allocation of resources. That is, some queues allocated using percentages and some queues using absolute units. Hence, you must not update the capacities in mixed mode at different levels. You must set the capacities at all levels in the same mode and that mode must match with the current Queue Manager mode.

For information about Partitions, see https://docs.cloudera.com/runtime/7.3.1/yarn-allocate-resources/topics/yarn-configure-partition-a-cluster.html. For information about changing queue capacities, see https://docs.cloudera.com/runtime/7.3.1/yarn-allocate-resources/topics/yarn-configuring-cluster-capacity-with-queues.html.

Parameter	Description
cm_host	Cloudera Manager host address
Cluster-id	Name of the cluster
queuePath	Path to the queue
queueName	Name of the queue
capacity	Capacity of the queue
maximum-capacity	Maximum capacity of the queue
name-of-the-partition	Name of the partition

Example Request

```
curl -v -X PUT -b cookie.txt --key cm-auto-host_key_decrypted.pem --cert cm-
auto-host_cert_chain.pem --cacert cm-auto-global_cacerts.pem -H "Referer: ht
tps://<cm_host>:7183/cmf/clusters/Cluster%201/queue-manager/"

my-cluster-1.my-cluster.root.hwx.site:7180/cmf/clusters/Cluster201/queue-man
ager-api/api/v1/environments/support/clusters/Cluster201/resources/scheduler
/partitions/default/queues/root.samplequeue -d @request.json
```

Note: You must save the Request Body content as a .json file before using this command. In the above example, the Request Body content is saved as the *REQUEST.JSON* file.

Change Queue Properties

You can configure numerous queue properties that can be included in the properties array, each with its own range of valid values.

For information about the YARN properties, see Managing and Allocating Cluster Resources using Capacity Scheduler.

Request URL

```
PUT
http://<CM_HOST>:7180/cmf/clusters/<CLUSTER-ID>/queue-manager-api/api/v1/en
vironments/support/clusters/<CLUSTER-ID>/resources/scheduler/partitions/defa
ult/queues/<PATH-TO-THE-QUEUE>
```

Request Body

```
{
  "properties": [{ "name": "{name-of-the-property}", "value": "{value}" }],
  "message": "Changed properties of {path-of-the-queue}"
}
```

Parameter	Description
cm_host	Cloudera Manager host address
Cluster-id	Name of the cluster
path-of-the-queue	Path to the queue
name-of-the-property	Name of the property
value	Value of the YARN property

Example Request Body

```
{
  "properties": [{ "name": "maximum-applications", "value": "1" }],
  "message": "Changed properties of root.samplequeue"
}
```

Note: You can change the queue capacity and configure queue capacity using the same request body.

Example Request

```
curl -v -X PUT -b cookie.txt --key cm-auto-host_key_decrypted.pem --cert cm-
auto-host_cert_chain.pem --cacert cm-auto-global_cacerts.pem -H "Referer: ht
tps://<cm_host>:7183/cmf/clusters/Cluster%201/queue-manager/"

my-cluster-1.my-cluster.root.hwx.site:7180/cmf/clusters/Cluster201/queue-man
ager-api/api/v1/environments/support/clusters/Cluster201/resources/scheduler
/partitions/default/queues/root.samplequeue -d @request.json
```

Note: You must save the Request Body content as a .json file before using this command. In the above example, the Request Body content is saved as the *REQUEST.JSON* file.

Delete Queue

You must set the capacity of the queue to zero before you delete a queue.

For information about changing queue capacity using the API, see Change Queue Capacities.

The delete queue requests cannot have a body, but the version message can be passed as a query string parameter as shown in the Request URL.

Request URL

```
DELETE
http://<CM_HOST>:7180/cmf/clusters/<CLUSTER-ID>/queue-manager-api/api/v1/en
vironments/support/clusters/<CLUSTER-ID>/resources/scheduler/partitions/defa
ult/queues/<PATH-TO-THE-QUEUE>
```

Parameter	Description
cm_host	Cloudera Manager host address
Cluster-id	Name of the cluster
path-of-the-queue	Path to the queue

Example Request

```
curl -v -X DELETE -b cookie.txt --key cm-auto-host_key_decrypted.pem --cert
cm-auto-host_cert_chain.pem --cacert cm-auto-global_cacerts.pem -H "Referer:
 https://<cm_host>:7183/cmf/clusters/Cluster%201/queue-manager/"

my-cluster-1.my-cluster.root.hwx.site:7180/cmf/clusters/Cluster201/queue-
manager-api/api/v1/environments/support/clusters/Cluster201/resources/schedu
ler/partitions/default/queues/root.a?message=Deleted%20root.a
```

Cloudera Runtime 7.3.1

Configuring Apache Hadoop YARN Security

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Linux Container Executor **4**

Managing Access Control Lists **4**
- YARN ACL rules 4
- YARN ACL syntax 5
- YARN ACL types 5
  - Admin ACLs 6
  - Queue ACLs 6
  - Application ACLs 6

Configuring TLS/SSL for Core Hadoop Services **10**
- Configuring TLS/SSL for HDFS 11
- Configuring TLS/SSL for YARN 12

Enabling HTTPS between ResourceManager and Application Master **13**

Configuring Cross-Origin Support for YARN UIs and REST APIs **14**

Configuring YARN Security for Long-Running Applications **15**

YARN Ranger authorization support **16**
- YARN Ranger authorization support compatibility matrix 17
- Enabling YARN Ranger authorization support 17
- Disabling YARN Ranger authorization support 17

Enabling custom Kerberos principal support in YARN **18**

Enabling custom Kerberos principal support in a Queue Manager cluster **20**

Linux Container Executor

A setuid binary called container-executor is part of the hadoop-yarn package and is installed in /var/lib/hadoop-yarn/bin/container-executor.

This container-executor program, which is used on YARN only and supported on GNU/Linux only, runs the containers as the user who submitted the application. It requires all user accounts to be created on the cluster hosts where the containers are launched. It uses a setuid executable that is included in the Hadoop distribution. The NodeManager uses this executable to launch and stop containers. The setuid executable switches to the user who has submitted the application and launches or stops the containers. For maximum security, this executor sets up restricted permissions and user/group ownership of local files and directories used by the containers such as the shared objects, jars, intermediate files, and log files. As a result, only the application owner and NodeManager can access any of the local files/directories including those localized as part of the distributed cache.

Important: Configuration changes to the Linux container executor could result in local NodeManager directories (such as usercache) being left with incorrect permissions. To avoid this, when making changes using either Cloudera Manager or the command line, first manually remove the existing NodeManager local directories from all configured local directories (yarn.nodemanager.local-dirs), and let the NodeManager recreate the directory structure.

Managing Access Control Lists

An Access Control List (ACL) is a list of specific permissions or controls that allow individual users or groups to perform specific actions upon specific objects, as well as defining what operations are allowed on a given object.

YARN ACLs do not deny access; rather, they identify a user, list of users, group, or list of groups who can access a particular object.

Like HDFS ACLs, YARN ACLs provide a way to set different permissions for specific named users or named groups. ACLs enhance the traditional permissions model by defining access control for arbitrary combinations of users and groups instead of a single owner/user or a single group.

YARN ACL rules

Learn about the YARN ACL rules to which all YARN ACLs must adhere.

All YARN ACLs must adhere to the following rules:

- Special Values:
  - The wildcard character (*) indicates that everyone has access.

    Note: You cannot use the wildcard (*) character along with a list of users and/or groups in the same ACL. If you use the wildcard character it must be the only item in the ACL.
  - A single space entry indicates that no one has access.
- If there are no spaces in an ACL, then all entries (the listed users and/or groups) are considered authorized users.
- Group names in YARN Resource Manager ACLs are case sensitive. So, if you specify an uppercase group name in the ACL, it will not match the group name resolved from the Active Directory because Active Directory group names are resolved in lowercase.
- If an ACL starts with a single space, then it must consist of groups only.
- All entries after the occurrence of a second single space in an ACL are ignored.
- There are no ACLs that deny access to a user or group. However, if you wish to block access to an operation entirely, enter a value for a non-existent user or group (for example,'NOUSERS NOGROUPS'), or simply enter a single space. By doing so, you ensure that no user or group maps to a particular operation by default.

- If you wish to deny only a certain set of users and/or groups, specify every single user and/or group that requires access. Users and/or groups that are not included are "implicitly" denied access.

YARN ACL syntax

Example of YARN ACL syntax.

Note: In all cases where a single space is required, you will see: <single space>.

- Users only

```
user1,user2,userN
```

  Use a comma-separated list of user names. Do not place spaces after the commas separating the users in the list.
- Groups only

```
<single space>HR,marketing,support
```

  You must begin group-only ACLs with a single space. Group-only ACLs use the same syntax as users, except each entry is a group name rather than user name.
- Users and Groups

```
fred,alice,haley<single space>datascience,marketing,support
```

  A comma-separated list of user names, followed by a single space, followed by a comma-separated list of group names. This sample ACL authorizes access to users “fred”, “alice”, and “haley”, and to those users in the groups “datascience”, “marketing”, and “support”.

Examples

The following ACL entry authorizes access only to the members of “my_group”:

```
<single space>my_group
```

The following ACL entry authorizes access to anyone:

```
*
```

The following ACL authorizes access to the users “john”, “jane”, and the group “HR”:

```
john,jane<single space>HR
```

In this example, six groups (“group_1” through “group_6”) are defined in the system. The following ACL authorizes access to a subset of the defined groups, allowing access to all members of groups 1 through 5 (and implicitly denies access to members of the group “group_6”):

```
<single space>group_1,group_2,group_3,group_4,group_5
```

YARN ACL types

There are three different kinds of YARN ACL types: YARN Admin ACLs, YARN Queue ACLs and YARN Application ACLs.

Admin ACLs

Use the YARN Admin ACLs to allow users to run YARN administrator sub-commands, which are executed via the yarn rmadmin <command>.

Important: The YARN Admin ACL is triggered and applied only when you run YARN sub-commands via yarn rmadmin <cmd>. If you run other YARN commands via the YARN command line (for example, starting the Resource or Node Manager), it does not trigger the YARN Admin ACL check or provide the same level of security.

The default YARN Admin ACL is set to the wildcard character (*), meaning all users and groups have YARN Administrator access and privileges. So after YARN ACL enforcement is enabled, (via the yarn.acl.enable property) every user has YARN ACL Administrator access. Unless you wish for all users to have YARN Admin ACL access, edit the yarn.admin.acl setting upon initial YARN configuration, and before enabling YARN ACLs.

A typical YARN Admin ACL looks like the following, where the system's Hadoop administrator and multiple groups are granted access:

```
hadoopadmin<space>yarnadmgroup,hadoopadmgroup
```

Queue ACLs

Use Queue ACLs to control access rights to users and administrators to the Capacity Scheduler queues. Application submission can really only happen at the leaf queue level, but an ACL restriction set on a parent queue will be applied to all of its descendant queues.

In the Capacity Scheduler, ACLs are configured by granting queue access to a list of users and groups with the Submit Application ACL parameter. Use Queue Administer ACL control the administration of queues at various levels. Queue administrators can submit applications to the queue, stop applications in the queue, and obtain information about any application in the queue.

For information about setting queue ACLs, see Control access to queues using ACLs

Related Information

Control access to queues using ACLs

Application ACLs

Use Application ACLs to provide a user or group access to an application.

Use Application ACLs to provide a user and/or group–neither of whom is the owner–access to an application. The most common use case for Application ACLs occurs when you have a team of users collaborating on or managing a set of applications, and you need to provide read access to logs and job statistics, or access to allow for the modification of a job (killing the job) and/or application. Application ACLs are set per application and are managed by the application owner.

Users who start an application (the owners) always have access to the application they start, which includes the application logs, job statistics, and ACLs. No other user can remove or change owner access. By default, no other users have access to the application data because the Application ACL defaults to “ “ (single space), which means no one has access.

Application ACL evaluation

The better you understand how YARN ACLs are evaluated, the more prepared you will be to define and configure the various YARN ACLs available to you.

For example, if you enable user access in Administrator ACLs, then you must be aware that user may have access to/see sensitive data, and should plan accordingly. So if you are the administrator for an entire cluster, you also have access to the logs for running applications, which means you can view sensitive information in those logs associated with running the application.

Best Practice: A best practice for securing an environment is to set the YARN Admin ACL to include a limited set of users and or groups.

The following diagram shows the evaluation flow for Application ACLs:

Figure 1: Application ACL Evaluation Flow

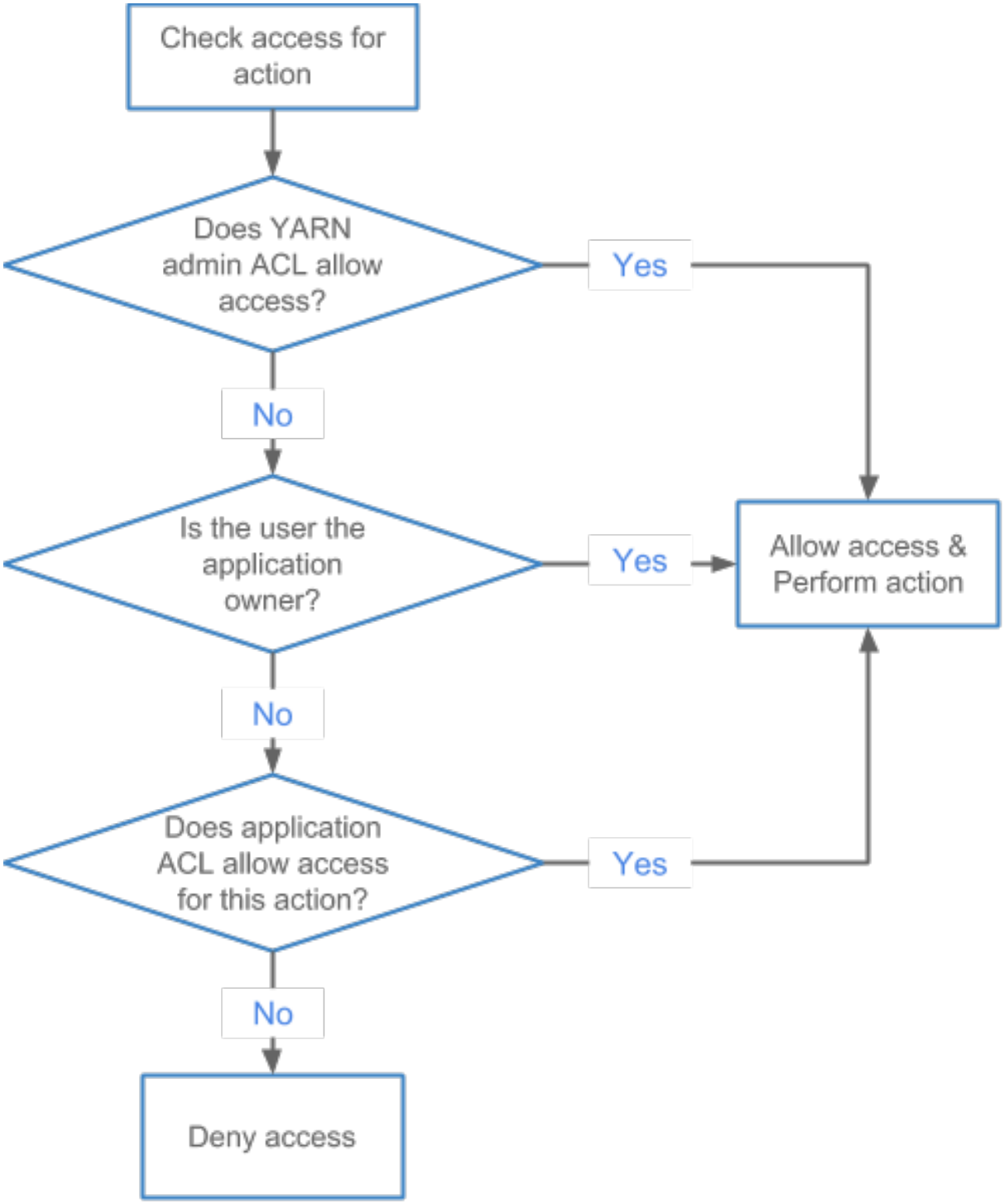

The following diagram shows a sample queue structure, starting with leaf queues on the bottom, up to root queue at the top:

Figure 2: Queue Structure

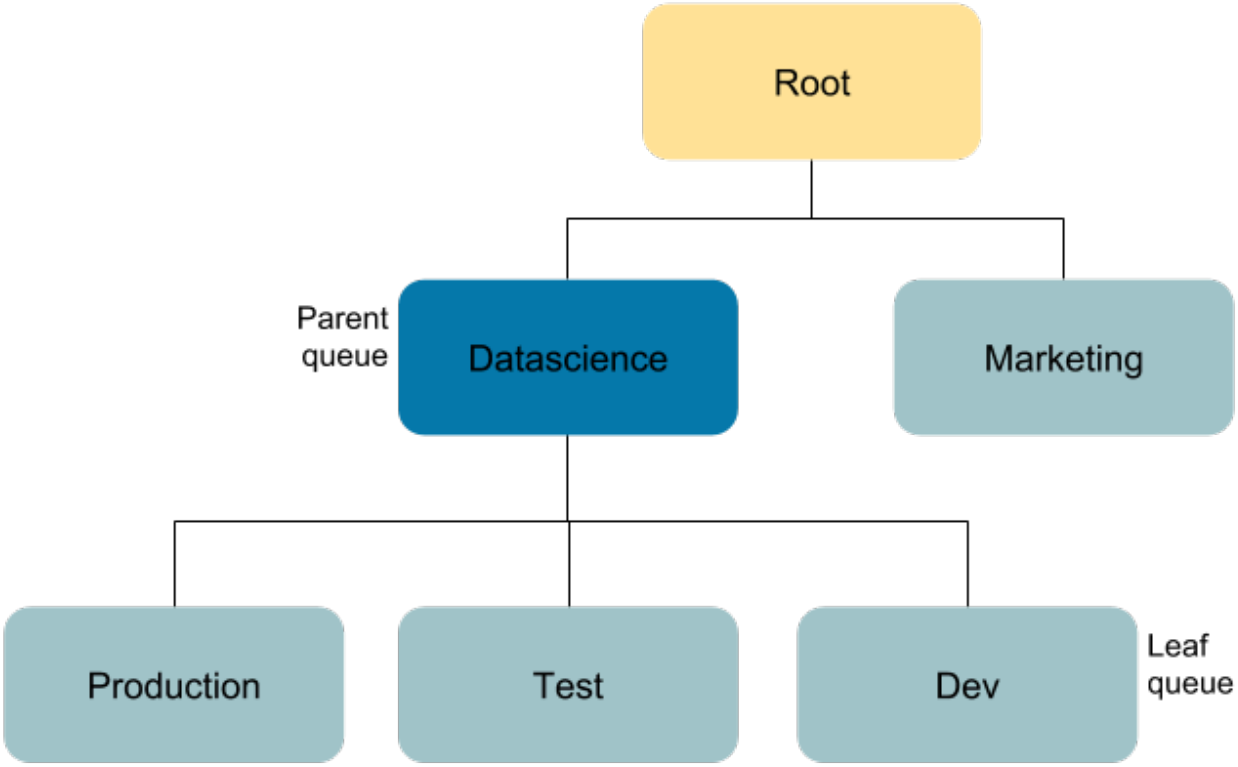

Example: Stopping an application in the "Production" queue

Provide the privilege to a user to stop an application in a specific queue.

For this Application ACL evaluation flow example, assume the following for application_1536220066338_0001 running in the queue "Production":

- Application owner: John
- "Datascience" queue administrator: Jane
- YARN cluster administrator: Bob

In this use case, John attempts to stop the application, which is allowed because he is the application owner.

Working as the queue administrator, Jane attempts to stop a job in the queue "Production", which she can do as the queue administrator of the parent queue.

Bob is the YARN cluster administrator and he is also listed as a user in the Admin ACL. He attempts to stop the job for which he is not the owner, but because he is the YARN cluster administrator, he can stop the job.

Example: Moving the application and viewing the log in the "Test" queue

Provide the privileges to a user to move application between queues and to view a log in a specific queue.

For this Application ACL evaluation flow example, assume the following for application_1536220066338_0002 running in the queue "Test":

- Application owner: John
- "Marketing" and "Dev" queue administrator: Jane
- Jane has log view rights via the mapreduce.job.acl-view-job ACL
- YARN cluster administrator: Bob

In this use case, John attempts to view the logs for his job, which is allowed because he is the application owner.

Jane attempts to access application_1536220066338_0002 in the queue "Test" to move the application to the "Marketing" queue. She is denied access to the "Test" queue via the queue ACLs–so she cannot submit to or administer the queue "Test". She is also unable to stop a job running in queue "Test". She then attempts to access the logs for application_1536220066338_0002 and is allowed access via the mapreduce.job.acl-view-job ACL.

Bob attempts to access application_1536220066338_0002 in the queue "Test" to move the application to the "Marketing" queue. As the YARN cluster administrator, he has access to all queues and can move the application.

Note: Permissions on the log files are also set at the filesystem level and are enforced by the filesystem: the filesystem can block you from accessing the file, which means that you can not open/read the file to check the ACLs that are contained in the file.

MapReduce Job ACLs

Create and use the following MapReduce Application ACLs to view YARN logs.

- mapreduce.job.acl-view-job

  Provides read access to the MapReduce history and the YARN logs.
- mapreduce.job.acl-modify-job

  Provides the same access as mapreduce.job.acl-view-job, and also allows the user to modify a running job.

  Note: Job modification is currently limited to killing the job. No other YARN system modifications are supported.

During a search or other activities, you may come across the following two legacy settings from MapReduce; they are not supported by YARN. Do not use them:

- mapreduce.cluster.acls.enabled
- mapreduce.cluster.administrators

Stopping an application

The Application ACL mapreduce.job.acl-modify-job determines whether or not a user can modify a job, but in the context of YARN, this only allows the user to stop an application.

The Application ACL mapreduce.job.acl-modify-job determines whether or not a user can modify a job, but in the context of YARN, this only allows the user to stop an application. The kill action is application agnostic and part of the YARN framework. Other application types, like MapReduce or Spark, implement their own kill action independent of the YARN framework. MapReduce provides the kill actions via the mapred command.

For YARN, the following three groups of users are allowed to stop a running application:

- The application owner
- A cluster administrator defined in yarn.admin.acl
- A queue administrator defined in aclAdministerApps for the queue in which the application is running

Note that for the queue administrators, ACL inheritance applies, as described earlier.

Related Information

Application logs' ACLs

Spark Job ACLs

Spark ACLs using a separate property for users and groups.

Both user and group lists use a comma-separated list of entries. The wildcard character “*” allows access to anyone, and the single space “ “ allows access to no one. Enable Spark ACLs using the property spark.acls.enable, which is set to false by default (not enabled) and must be changed to true to enforce ACLs at the Spark level.

Create and use the following Application ACLs for the Spark application:

- Set spark.acls.enable to true (default is false).
- Set spark.admin.acls and spark.admin.acls.groups for administrative access to all Spark applications.

- Set spark.ui.view.acls and spark.ui.view.acls.groups for view access to the specific Spark application.
- Set spark.modify.acls and spark.modify.acls.groups for administrative access to the specific Spark application.

Application logs' ACLs

The MapReduce Application ACL mapreduce.job.acl-view-job determines whether or not you can view an application log.

Aaccess is evaluated through the following ACLs:

- YARN Admin and Queue ACLs
- Application ACLs

After an application is in the “finished” state, logs are aggregated, depending on your cluster setup. You can access the aggregated logs via the MapReduce History server web interface. Aggregated logs are stored on shared cluster storage, which in most cases is HDFS. You can also share log aggregation via storage options like S3 or Azure by modifying the yarn.nodemanager.remote-app-log-dir setting in Cloudera Manager to point to either S3 or Azure, which should already be configured.

The shared storage on which the logs are aggregated helps to prevent access to the log files via file level permissions. Permissions on the log files are also set at the file system level, and are enforced by the file system: the file system can block any user from accessing the file, which means that the user cannot open/read the file to check the ACLs that are contained within.

In the cluster storage use case of HDFS, you can only access logs that are aggregated via the:

- Application owner
- Group defined for the MapReduce History server

When an application runs, generates logs, and then places the logs into HDFS, a path/structure is generated (for example: /tmp/logs/john/logs/application_1536220066338_0001). So access for the application owner "john" might be set to 700, which means read, write, execute; no one else can view files underneath this directory. If you don’t have HDFS access, you will be denied access. Command line users identified in mapreduce.job.acl-view-job are also denied access at the file level. In such a use case, the Application ACLs stored inside the aggregated logs will never be evaluated because the Application ACLs do not have file access.

For clusters that do not have log aggregation, logs for running applications are kept on the node where the container runs. You can access these logs via the Resource Manager and Node Manager web interface, which performs the ACL checks.

Configuring TLS/SSL for Core Hadoop Services

TLS/SSL for the core Hadoop services (HDFS and YARN) must be enabled as a group.

TLS/SSL for the core Hadoop services (HDFS and YARN) must be enabled as a group. Enabling TLS/SSL on HDFS is required before it can be enabled on YARN.

Note: If you enable TLS/SSL for HDFS, you must also enable it for YARN.

The steps in the following topics include enabling Kerberos authentication for HTTP Web-Consoles. Enabling TLS/SSL for the core Hadoop services on a cluster without enabling authentication displays a warning.

Before You Begin

- Before enabling TLS/SSL, keystores containing certificates bound to the appropriate domain names will need to be accessible on all hosts on which at least one HDFS or YARN daemon role is running.
- Since HDFS and YARN daemons act as TLS/SSL clients as well as TLS/SSL servers, they must have access to truststores. In many cases, the most practical approach is to deploy truststores to all hosts in the cluster, as it may not be desirable to determine in advance the set of hosts on which clients will run.

- Keystores for HDFS and YARN must be owned by the hadoop group, and have permissions 0440 (that is, readable by owner and group). Truststores must have permissions 0444 (that is, readable by all)
- Cloudera Manager supports TLS/SSL configuration for HDFS and YARN at the service level. For each of these services, you must specify absolute paths to the keystore and truststore files. These settings apply to all hosts on which daemon roles of the service in question run. Therefore, the paths you choose must be valid on all hosts.

  An implication of this is that the keystore file names for a given service must be the same on all hosts. If, for example, you have obtained separate certificates for HDFS daemons on hosts node1.example.com and node2.example.com, you might have chosen to store these certificates in files called hdfs-node1.keystore and hdfs-node2.keystore (respectively). When deploying these keystores, you must give them both the same name on the target host — for example, hdfs.keystore.
- Multiple daemons running on a host can share a certificate. For example, in case there is a DataNode and an Oozie server running on the same host, they can use the same certificate.

Configuring TLS/SSL for HDFS

Enabling TLS/SSL on HDFS is required before it can be enabled on YARN.

About this task

Enabling TLS/SSL on HDFS is required before it can be enabled on YARN.

Cloudera recommends you enable web UI authentication for the HDFS service. Web UI authentication uses SPNEGO. After enabling this, you cannot access the Hadoop web consoles without a valid Kerberos ticket and proper client-side configuration.

Procedure

1. In Cloudera Manager, select the HDFS service.
2. Click the Configuration tab.
3. Search for TLS/SSL.
4. Find and edit the following properties according to you cluster configuration:

Property	Description
Hadoop TLS/SSL Server Keystore File Location	Path to the keystore file containing the server certificate and private key.
Hadoop TLS/SSL Server Keystore File Password	Password for the server keystore file.
Hadoop TLS/SSL Server Keystore Key Password	Password that protects the private key contained in the server keystore.

If you are not using the default trustore, do the following:

5. Configure TLS/SSL client trustore properties.

Important: The HDFS properties below define a cluster-wide default truststore that can be overridden by YARN.

Property	Description
Cluster-Wide Default TLS/SSL Client Truststore Location	Path to the client truststore file. This truststore contains certificates of trusted servers, or of Certificate Authorities trusted to identify servers.
Cluster-Wide Default TLS/SSL Client Truststore Password	Password for the client truststore file.

If you want to enable Web UI authentication for the HDFS service, do the following:

6. Search for web consoles.
7. Find the Enable Authentication for HTTP Web-Consoles property.

8. Check the property to enable web UI authentication.

   Note: This is effective only if security is enabled for the HDFS service.

9. Click Save Changes.
10. Configure TLS/SSL for YARN.

Related Information

Configuring TLS/SSL for YARN

Configuring TLS/SSL for YARN

If you enabled TLS/SSL for HDFS, you must also enable it for YARN.

About this task

If you enable TLS/SSL for HDFS, you must also enable it for YARN.

Cloudera recommends to enable Web UI authentication for YARN.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for TLS/SSL.
4. Find and edit the following properties according to your cluster configuration:

Property	Description
Hadoop TLS/SSL Server Keystore File Location	Path to the keystore file containing the server certificate and private key.
Hadoop TLS/SSL Server Keystore File Password	Password for the server keystore file.
Hadoop TLS/SSL Server Keystore Key Password	Password that protects the private key contained in the server keystore.

If you want to override the cluster-wide defaults set by the HDFS properties, do the following:

5. Configure the following TLS/SSL client truststore properties for YARN.

Property	Description
TLS/SSL Client Truststore File Location	Path to the client truststore file. This truststore contains certificates of trusted servers, or of Certificate Authorities trusted to identify servers.
TLS/SSL Client Truststore File Password	Password for the client truststore file.

If you want to enable the TSL/SSL for YARN Queue Manager, do the following:

Note: If Cloudera Manager is TLS/SSL enabled, then ensure that YARN Queue Manager is also TLS/SSL enabled. Disabling Queue Manager TLS/SSL where, Cloudera Manager has TLS/SSL causes error.

6. In Cloudera Manager, select the QUEUE MANAGER service.
7. Click the Configuration tab.
8. Search for TSL/SSL.
9. Select the Enable TLS/SSL for YARN Queue Manager Store checkbox to encrypt communication between clients and YARN Queue Manager Store .

10. Configure the following properties according to your cluster configuration:

Property	Description
TLS/SSL Server JKS Keystore File Location	Path to the JKS keystore file containing the server and private key.
TLS/SSL Server JKS Keystore File Password	Password for the JKS keystore file.

11. Select the Enable TLS/SSL for YARN Queue Manager Webapp checkbox to encrypt communication between clients and YARN Queue Manager Webapp.
12. Configure the following properties according to your cluster configuration:

Property	Description
Webapp TLS/SSL Server JKS Keystore File Location	Path to the JKS keystore file containing the server and private key.
Webapp TLS/SSL Server JKS Keystore File Password	Password for the Webapp JKS keystore file.
Webapp TLS/SSL Client Trust Store File	Path to the client JKS truststore file. This truststore contains certificates of trusted servers, or of Certificate Authorities trusted to identify servers.
Webapp TLS/SSL Client Trust Store Password	Password for the Webapp truststore file.

If you want to enable Web UI authentication for YARN, do the following:

13. Search for web consoles.
14. Find the Enable Authentication for HTTP Web-Consoles property.
15. Check the property to enable web UI authentication.

Note: This is effective only if security is enabled for the HDFS service.

16. Click Save Changes.
17. Go back to the home page, by clicking the Cloudera Manager logo.
18. Select the HDFS service.
19. Click the Configuration tab.
20. Search for Hadoop SSL Enabled.
21. Find and select the Hadoop SSL Enabled property.
    The SSL communication for HDFS and YARN is enabled.
22. Click Save Changes.
23. Click the *Stale Service Restart* icon that is next to the service to invoke the cluster restart wizard.
24. Click Restart Stale Services.
25. Select Re-deploy client configuration.
26. Click Restart Now.

Related Information

Configuring TLS/SSL for HDFS

Enabling HTTPS between ResourceManager and Application Master

If you want to enable HTTPS communication between the ResourceManager and the Application Master, you have to use a ResourceManager-level and two Application Master-level configuration properties.

Procedure

1. Enable HTTPS communication on the ResourceManager's side:
   a) Navigate to Cloudera Manager.
   b) Select the YARN service.
   c) Search for ResourceManager and find the ResourceManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml safety valve.
   d) Add the following configuration:

   ```
   <property>
   <name>yarn.resourcemanager.application-https.policy</name>
   <value>STRICT</value>
   </property>
   ```

   e) Restart the ResourceManager to apply the configuration change.

   HTTPS communication is enabled on the ResourceManager's side. Keystore and Truststore are available.

   There are two supported Keystore and Truststore formats. The format is automatically set by the ResourceManager:

   - JKS: when FIPS is not enabled
   - BCFKS: when FIPS is enabled

   The NodeManager collects the Keystore and Truststore related information (location on the file system, password, and format) and uses environment variables to provide the Application Master with the information.
2. Enable the HTTPS communication on the ApplicationMaster's side:

   When submitting a job set the yarn.app.mapreduce.am.webapp.https.enabled and the yarn.app.mapreduce.am.webapp.https.client.auth properties to true.

   The following is an example of a simple 'pi job' that can be considered as a test job. Note that the jar file is depending on the deployed CDP version:

   ```
   yarn jar /opt/cloudera/parcels/CDH-7.1.5-1.cdh7.1.5.p0.6683980/lib/hadoop-
   mapreduce/hadoop-mapreduce-examples-3.1.1.7.1.5.0-149.jar pi -Dyarn.app.ma
   preduce.am.webapp.https.enabled=true -Dyarn.app.mapreduce.am.webapp.http
   s.client.auth=true 1 1000
   ```

   HTTPS communication between ResourceManager and Application Master is enabled.

What to do next

For more information, see the design document in YARN-6586

Configuring Cross-Origin Support for YARN UIs and REST APIs

Cross-Origin Resource Sharing (CORS) is enabled by default on YARN so that the corresponding services accept cross-origin requests from only selected domains. Enabling CORS also helps the YARN UI fetch data endpoints from the browser.

About this task

CORS is enabled at cluster level and then on individual components such as YARN. If you want to set other values than the default ones, use safety valves in Cloudera Manager to configure the CORS configuration properties.

Procedure

1. In the Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for core-site.xml.
4. Find YARN Service Advanced Configuration Snippet (Safety Valve) for core-site.xml.
5. Click on the plus icon to add and configure the following properties :

Property	Default value	Description
hadoop.http.cross-origin.allowed-origins	regex:.*[.]subdomain[.]*EXAMPLE*[.]com(;\d+)?.*	Comma-separated list of origins allowed for cross-origin support. The default value is *. Mention only specific origins so that the services do not accept all the cross-origin requests.
hadoop.http.cross-origin.allowed-methods	GET, PUT, POST, OPTIONS, HEAD, DELETE	Comma-separated list of methods allowed for cross-origin support.
hadoop.http.cross-origin.allowed-headers	X-Requested-With, Content-Type, Accept, Origin, WWW-Authenticate, Accept-Encoding, Transfer-Encoding	Comma-separated list of headers allowed for cross-origin support.
hadoop.http.cross-origin.max-age	180	Number of seconds until when a preflight request can be cached.

6. Click Save Changes.
7. In the Cloudera Manager, select the YARN service.
8. Click the Configuration tab.
9. Search for yarn-site.xml.
10. Find the YARN Service Advanced Configuration Snippet (Safety Valve) for yarn-site.xml.
11. Click on the plus icon to add and configure the cross-origin.enabled properties:

Property	Default value	Description
yarn.nodemanager.webapp.cross-origin.enabled	true	Enable cross-origin support for the NodeManager.
yarn.resourcemanager.webapp.cross-origin.enabled	true	Enable cross-origin support for the ResourceManager.
yarn.timeline-service.http-cross-origin.enabled	true	Enable cross-origin support for the timeline service.

Note: The value of the hadoop.http.cross-origin.allowed-origins property in yarn-site.xml overrides the value of the same property in core-site.xml. You can configure the value of this property to allow access to specific domains; for example, regex:.*[.]hwx[.]site(:\d*)?.

12. Click Save Changes.
13. Restart the HDFS and YARN services.

Configuring YARN Security for Long-Running Applications

Long-running applications need additional configuration since the default settings only allow the hdfs user's delegation tokens a maximum lifetime of 7 days which is not always sufficient.

About this task

Long-running applications such as Spark Streaming jobs will need additional configuration since the default settings only allow the hdfs user's delegation tokens a maximum lifetime of 7 days which is not always sufficient.

You can work around this by configuring the ResourceManager as a proxy user for the corresponding HDFS NameNode so that the ResourceManager can request new tokens when the existing ones are past their maximum lifetime. YARN will then be able to continue performing localization and log-aggregation on behalf of the hdfs user.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for yarn-site.xml.
4. Find the ResourceManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml.
5. Click the plus icon and add the followoing:
   - Name: yarn.resourcemanager.proxy-user-privileges.enabled
   - Value: true
6. Click Save Changes.
7. Go back to the home page by clicking the Cloudera Manager logo.
8. Select the HDFS service.
9. Click the Configuration tab.
10. Search for core-site.xml.
11. Find the Cluster-wide Advances Configuration Snippet (Safety Valve) for core-site.xml.
12. Click the plus icon and add the following:
    - Name: hadoop.proxyuser.yarn.hosts
    - Value: *
13. Click the plus icon again and add the following:
    - Name: hadoop.proxyuser.yarn.groups
    - Value: *
14. Click Save Changes.
15. Click the *Stale Service Restart* icon that is next to the service to invoke the cluster restart wizard.
16. Click Restart Stale Services.
17. Select Re-deploy client configuration.
18. Click Restart Now.

YARN Ranger authorization support

The YARN Ranger authorization support feature enables each Cloudera Data Hub cluster’s YARN cluster to have a dedicated Ranger YARN repository. As a result admins can set different YARN policies for different Cloudera Data Hub clusters.

The YARN Ranger authorization support feature enables admins to create separate YARN queues for users sharing the same cluster and to control who can submit jobs to each queue, and who can see or kill jobs within the queue. In addition, users without admin privileges can only see their own jobs in a given queue.

This feature is enabled by default in a new Cloudera Data Hub cluster installation. Cloudera Manager automatically creates a Ranger YARN repository for each cluster.

The YARN Ranger authozitation support is not enabled by default after migrating to Cloudera Public Cloud 7.2.10 or higher. That means no YARN Ranger repository is created after migration, and the cm_yarn repository is used until the cluster is deleted and a new Cloudera Data Hub cluster with Cloudera Manager 7.4.2 or higher is created.

Alternatively, you can enable this feature manually using Cloudera Manager.

YARN Ranger authorization support compatibility matrix

Check the compatibility matrix to see what Cloudera Runtime versions support the YARN Ranger authorization support feature.

Table 1: YARN Ranger authorization support: Compatibility Matrix

SDX Cloudera Runtime Version	Cloudera Data Hub Cloudera Runtime version	Fresh install or Upgraded cluster?	YARN Ranger plugin enabled?	YARN Ranger plugin default service name
7.2.10.0 and higher	7.2.10 and higher	Fresh install	Yes	<DE_Cluster_Name>_yarn
7.2.9 and lower	7.2.9 and lower	Fresh isntall	Yes	cm_yarn
Source: 7.2.9 and lower Target: 7.2.10 and higher	Source: 7.2.9 and lower Target: 7.2.10 and higher	*Upgrade [Technical Preview]	Yes	cm_yarn
Source: 7.2.10 Target: 7.2.10 and higher	Source: 7.2.10 Target: 7.2.10 and higher	*Upgrade [Technical Preview]	Yes	<DE_Cluster_Name>_yarn

Enabling YARN Ranger authorization support

Enable the YARN Ranger authorization support feature if you want to set different YARN policies for different Cloudera Data Hub clusters.

Procedure

1. In Cloudera Manager, navigate to YARN Configuration .
2. Search for ranger service.
3. Find the Ranger Service Name property.
4. Set the Ranger service using the following pattern: <ClusterName>_yarn
5. Click Save Changes and restart the YARN service.

Disabling YARN Ranger authorization support

You can disable any YARN Ranger repository through the Ranger Admin UI.

Procedure

1. In Cloudera Manager, selec the Ranger service.
2. Click Ranger Admin Web UI.
   Takes you to the Ranger Admin Web UI site.
3. Log in to the Ranger Admin Web UI.
4. Find the applicable YARN Ranger repository.
5. Click the edit icon.
6. Find the Active Status field and select Disabled.

7. Click Save.

Enabling custom Kerberos principal support in YARN

By default Cloudera Manager sets the Kerberos principal in YARN to yarn. This value can be easily changed using Cloudera Manager.

Procedure

1. Disable the ZooKeeper ACL check temporarily.
   a) In Cloudera Manager, navigate to ZooKeeper Configuration .
   b) Find the Java Configuration Options for ZooKeeper Server property.
   c) Add the following value:

   ```
   -Dzookeeper.skipACL=yes
   ```

   d) Click Save Changes.
   e) Restart the ZooKeeper service.
2. In Cloudera Manager, navigate to YARN Configuration .
3. Find the Kerberos principal property and set it to the required name.
4. Click Save Changes.
5. Find the Admin ACL property and ensure that it contains either the ${yarn_user} placeholder or the updated principal name value.
6. Ensure that YARN HDFS folders have the correct access rights and owners.

   For example, if you use the Capacity Scheduler Node labels feature, the yarn.node-labels.fs-store.root-dir folder could be inaccessible by YARN with the new Kerberos principal name. To resolve this issue, make it accessible or use the hdfs chown command. The following is the minimum list of folder paths for which ownership needs to be changed if Capacity Scheduler Node labels are enabled:

   - /yarn/node-labels
   - /user/yarn/mapreduce/mr-framework
   - /user/yarn/services/service-framework/*
   - /tmp/logs
7. Restart all services with stale configuration.
   a) Click Stale configuration: Restart needed.
   b) Ensure that Deploy client configuration is selected.
   c) Click Restart.
8. Select the YARN service.

9. Click Actions Reset ACLs on ZooKeeper nodes .

This step can take a long time if a lot of application data is stored in ZooKeeper.

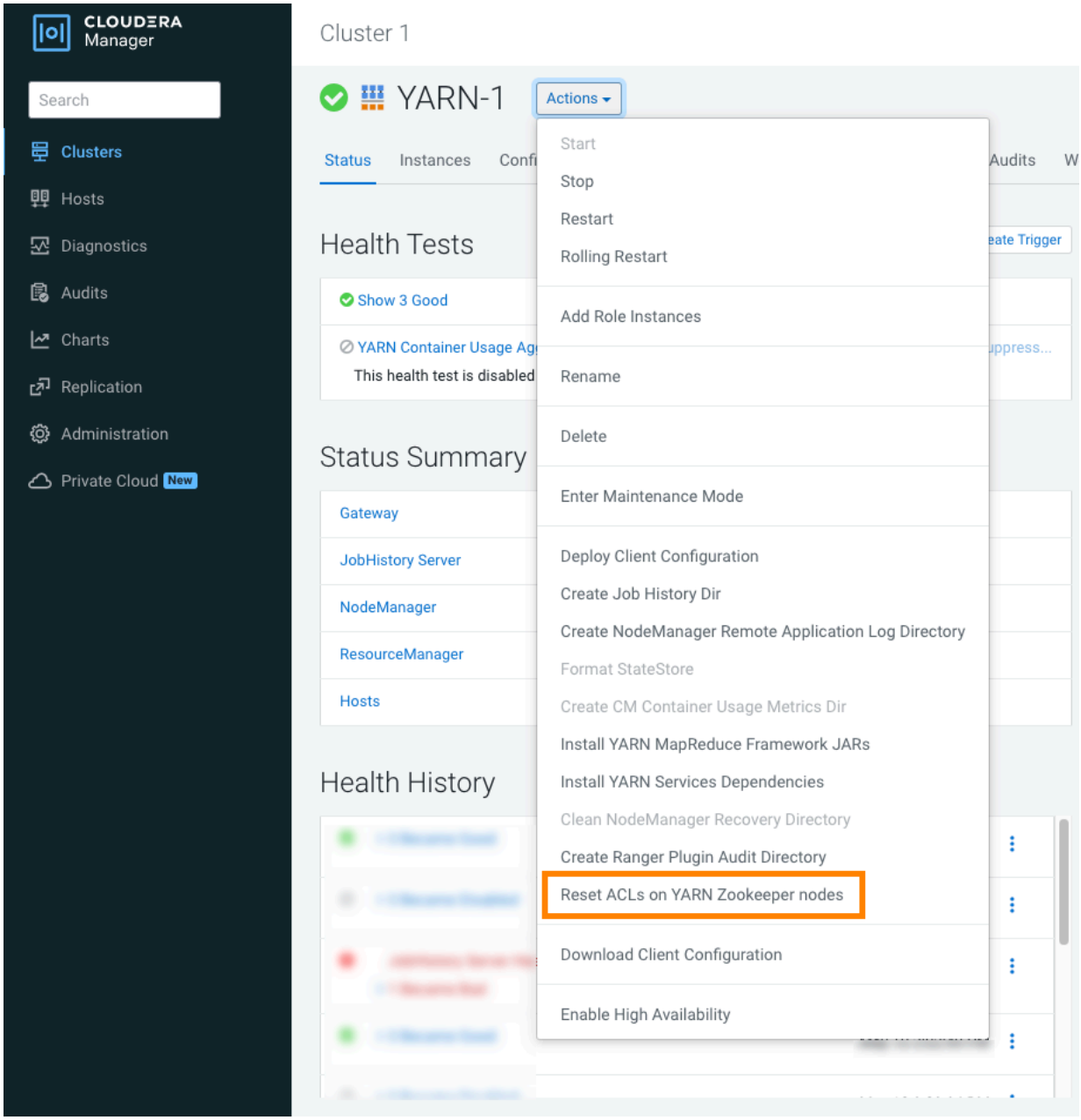

10. Enable the ZooKeeper ACL check.

a) Navigate to ZooKeeper Configuration .
b) Find Jave Configuration Options for ZooKeeper Server.
c) Remove the -Dzookeeper.skipACL=yes value.
d) Click Save Changes.
e) Restart the ZooKeeper service.

Enabling custom Kerberos principal support in a Queue Manager cluster

In a secure Queue Manager cluster you can enable custom Kerberos principal using Cloudera Manager.

Before you begin

Ensure your Queue Manager cluster is secure. For that check if it runs on port 7183.

Procedure

1. In Cloudera Manager, navigate to YARN Configuration .
2. Search for admin acl.
3. Find the Admin ACL (yarn_admin_acls) property.
4. Add the user you want to use for the Kerberos principal to the list of Admin ACL property.
5. Click Save Changes.
6. Restart the YARN service.
7. Navigate to Queue Manager Configuration .
8. Search for kerberos principal.
9. Find the Kerberos Principal (kerberos_princ_name) property.
10. Add the same user you added to the Admin ACL property.
11. Click Save Changes.
12. Restart the YARN Queue Manager service to avoid staleness.

If you get the Exception: Only admins can carry out this operation error message while doing Queue Manager operations, ensure that the principal name is added to the YARN Admin ACL list.

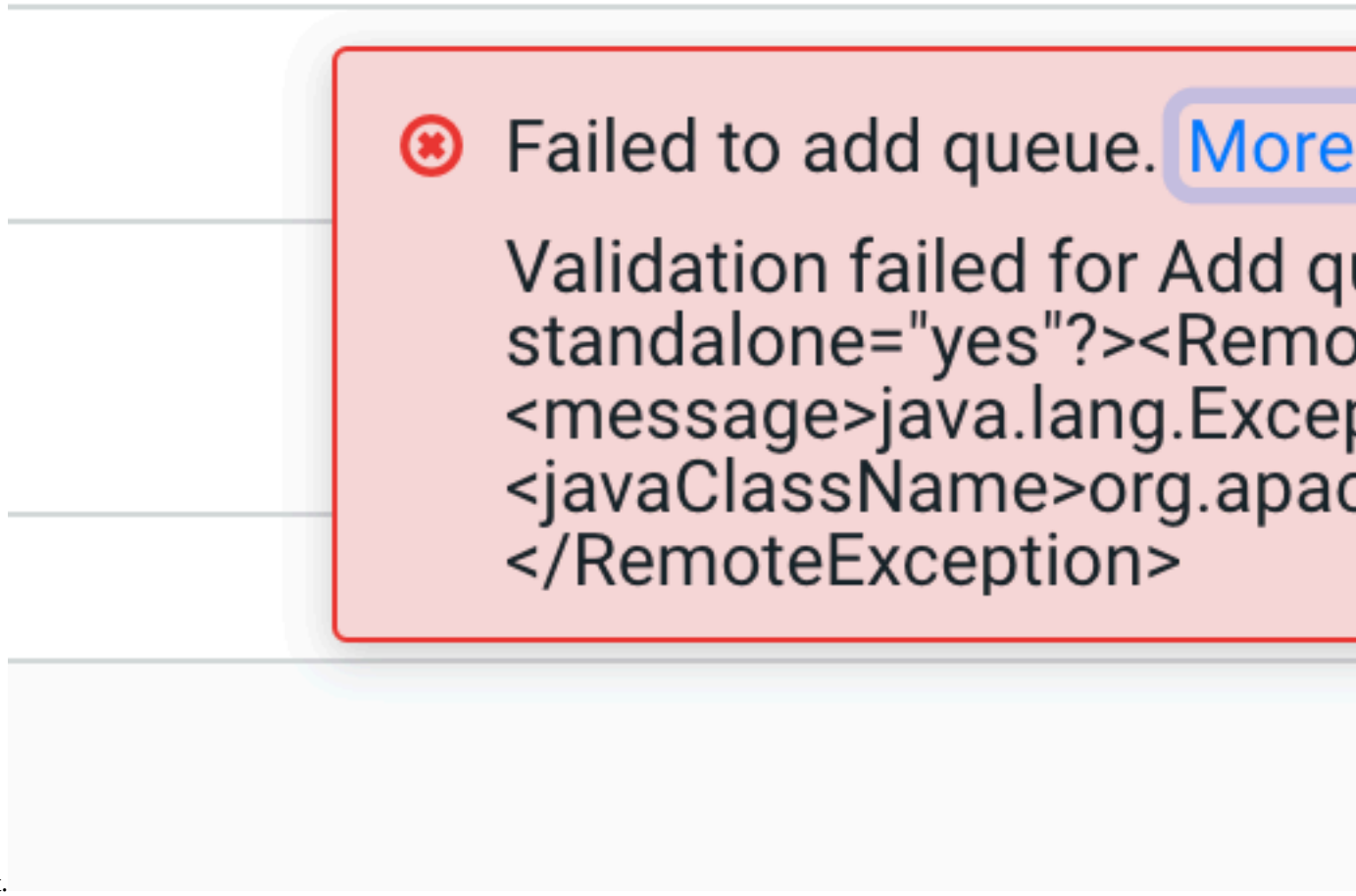

Cloudera Runtime 7.3.1

Troubleshooting Apache Hadoop YARN

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Troubleshooting Docker on YARN.....4

Troubleshooting on YARN.....11

YARN Queue Manager UI behavior in mixed resource allocation mode.....11

Troubleshooting for mixed resource allocation mode in YARN Queue Manager..... 13

Troubleshooting Linux Container Executor.....13

Troubleshooting Docker on YARN

A list of common Docker on YARN related problem and how to resolve them.

Docker is not enabled

Problem statement

Started an application on Docker, but the containers are running as regular containers.

Root cause

Docker is not enabled.

Resolution

Enable Docker in Cloudera Manager.

YARN_CONTAINER_RUNTIME_TYPE runtime environment variable is not provided during Application submission

Problem statement

Started an application on Docker, but the containers are running as regular containers.

Root cause

YARN_CONTAINER_RUNTIME_TYPE runtime environment variable is not provided during Application submission.

Resolution

Provide the environment variable when submitting the application.

LCE enforces running user to be nobody in an unsecure cluster

Problem statement

On an unsecure cluster, Appattempt exited with exitCode -1000 with diagnostic message:

```
[...]
main : run as user is nobody
main : requested yarn user is yarn
Can't create directory /yarn/nm/usercache/yarn/appcache/applic
ation_1570626013274_0001 - Permission denied
```

Root cause

LCE enforces running user to be nobody in an unsecure cluster if yarn.nodemanager.linux-container-executor.nonsecure-mode.limit-users is set.

Resolution

In Cloudera Manager, add the following configuration to the YARN Service Advanced Configuration Snippet (Safety Valve) for yarn-site.xml safety-valve by clicking the plus icon:

- Key: yarn.nodemanager.linux-container-executor.nonsecure-mode.limit-users
- Value: false

Then use a user who has the correct permissions or add more permissive access to these folders for the nobody user. For more information, see YARN force nobody user on all jobs.

The Docker binary is not found

Problem Statement

Container launch fails with the following message:

```
Container launch fails
Exit code: 29
Exception message: Launch container failed
Shell error output: sh: <docker binary path, /usr/bin/docker by
default>: No such file or directory
Could not inspect docker network to get type /usr/bin/docker net
work inspect host --format='{{.Driver}}'.
Error constructing docker command, docker error code=-1, error
 message='Unknown error'
```

Root cause

The Docker binary is not found.

Resolution

The Docker binary is either not installed or installed to a different folder. Install Docker binary and provide the path to the binaries by specifying it using the Docker Binary Path (docker.binary) property in Cloudera Manager.

The Docker daemon is not running or does not respond

Problem statement

Container launch fails with the following message:

```
[timestamp] Exception from container-launch.
Container id: container_e06_1570629976081_0004_01_000003
Exit code: 29
Exception message: Launch container failed
Shell error output: Cannot connect to the Docker daemon at unix:/
//var/run/docker.sock. Is the docker daemon running?
Could not inspect docker network to get type /usr/bin/docker ne
twork inspect host --format='{{.Driver}}'.
Error constructing docker command, docker error code=-1, error me
ssage='Unknown error'
```

Root cause

The Docker daemon is not running or does not respond.

Resolution

Start or restart the Docker daemon with the dockerd command.

Docker rpm misses some symbolic link

Problem statement

On Centos 7.5 container launch fails with the following message:

```
[...]
[layer hash]: Pull complete
[layer hash]: Pull complete
Digest: sha256:[sha]
Status: Downloaded newer image for [image]
/usr/bin/docker-current: Error response from daemon: shim error:
 docker-runc not installed on system.
```

Root cause

Docker rpm misses some symbolic link.

Resolution

Create the missing symbolic link using the following command in a terminal:sudo ln -s /usr/libexec/docker/docker-runc-current /usr/bin/docker-runc

YARN_CONTAINER_RUNTIME_DOCKER_IMAGE is not set

Problem statement

Container launch fails with the following message:

```
[timestamp]Exception from container-launch.
Container id: container_e06_1570629976081_0004_01_000003
Exit code: -1
Exception message: YARN_CONTAINER_RUNTIME_DOCKER_IMAGE not set!
Shell error output: <unknown>
Shell output: <unknown>
```

Root cause

YARN_CONTAINER_RUNTIME_DOCKER_IMAGE is not set.

Resolution

Set the YARN_CONTAINER_RUNTIME_DOCKER_IMAGE environment variable when submitting the application.

Image is not trusted

Problem statement

Container launch fails with the following message:

```
[timestamp] Exception from container-launch.
Container id: container_e06_1570629976081_0004_01_000003
Exit code: 127
Exception message: Launch container failed
Shell error output: image: [image] is not trusted.
Disable mount volume for untrusted image
image: library/ibmjava:8 is not trusted.
Disable cap-add for untrusted image
Docker capability disabled for untrusted image
[...]
```

Root cause

The image is not trusted.

Resolution

Add the image's registry to the list of trusted registries (docker.trusted.registries). For example in case of library/ubuntu:latest, add the "library" registry to that list.

Docker image does not include the Snappy library

Problem statement

Running the hadoop-mapreduce-examples pi job fails with the following error:

```
[...]
[timestamp] INFO mapreduce.Job:  map 0% reduce 0%
[timestamp] INFO mapreduce.Job: Task Id : attempt_1570629976081_
0001_m_000000_0, Status : FAILED
Error: org/apache/hadoop/util/NativeCodeLoader.buildSupportsSna
ppy()Z
```

Root cause

The provided Docker image does not include the Snappy library. MapReduce needs this if compression is used and the Snappy codec is chosen for compression.

Resolution

Either add the Snappy library to the image or change the “Compression Codec of MapReduce Map Output” to some other codec

Hadoop UserGroupInformation class does not have access to the user permissions in the host system

Problem statement

Container fails shortly after start with the following exception:

```
Exception in thread "main" org.apache.hadoop.security.KerberosAu
thException: failure to login: javax.security.auth.login.LoginEx
ception: java.lang.NullPointerException: invalid null input: name
 At com.sun.security.auth.UnixPrincipal.<init>(UnixPrincipal.j
ava:71)
 at com.sun.security.auth.module.UnixLoginModule.login(UnixLogin
Module.java:133)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAcc
essorImpl. java:62)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke( DelegatingM
ethodAccessorImpl.java:43)
```

Root cause

The Hadoop UserGroupInformation class does not have access to the user permissions in the host system.

Resolution

Mount the /etc/passwd to the image. More configuration issues can be found in upstream Hadoop 3.2 documentation: Launching Applications Using Docker Containers upstream documentation.

Kerberos configuration is not mounted for Docker containers

Problem Statement

MapReduce and Spark jobs fail with Docker on a secure cluster. It cannot get Kerberos realm.

```
user@<HOSTNAME> /]$ cd /yarn/container-logs/application_157376
4921308_0002/container_e147_1573764921308_0002_01_000005
[user@<HOSTNAME> container_e147_1573764921308_0002_01_000005]$ ll
total 8
-rw-r--r-- 1 systest yarn   0 Nov 14 12:57 prelaunch.err
-rw-r--r-- 1 systest yarn  70 Nov 14 12:57 prelaunch.out
-rw-r--r-- 1 systest yarn   0 Nov 14 12:57 stderr
-rw-r----- 1 systest yarn   0 Nov 14 12:57 stderr.txt
-rw-r--r-- 1 systest yarn   0 Nov 14 12:57 stdout
-rw-r----- 1 systest yarn   0 Nov 14 12:57 stdout.txt
-rw-r--r-- 1 systest yarn 892 Nov 14 12:57 syslog
[user@<HOSTNAME> container_e147_1573764921308_0002_01_000005]$
cat syslog
2019-11-14 20:57:41,765 ERROR [main] org.apache.hadoop.yarn.YarnU
ncaughtExceptionHandler: Thread Thread[main,5,main] threw an Exc
eption.
java.lang.IllegalArgumentException: Can't get Kerberos realm
    at org.apache.hadoop.security.HadoopKerberosName.setConfigu
ration(HadoopKerberosName.java:71)
    at org.apache.hadoop.security.UserGroupInformation.initialize
(UserGroupInformation.java:330)
```

```
    at org.apache.hadoop.security.UserGroupInformation.setConfig
uration(UserGroupInformation.java:381)
    at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:80)
Caused by: java.lang.IllegalArgumentException
    at javax.security.auth.kerberos.KerberosPrincipal.<init>(Kerb
erosPrincipal.java:136)
    at org.apache.hadoop.security.authentication.util.KerberosUt
il.getDefaultRealm(KerberosUtil.java:108)
    at org.apache.hadoop.security.HadoopKerberosName.setConfig
uration(HadoopKerberosName.java:69)
    ... 3 more
[user@<HOSTNAME> container_e147_1573764921308_0002_01_000005]$
```

Root cause

Kerberos configuration is not mounted for Docker containers.

Resolution

In case of MapReduce job, add the following environment variable when running the job: -Dmapreduce.reduce.env=YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS=/etc/krb5.conf:/etc/krb5.conf:ro

Ensure to add /etc/krb5.conf to the Allowed Read-Only Mounts in Cloudera Manager configuration.

Example:

```
yarn jar /opt/cloudera/parcels/CDH-7.0.3-1.cdh7.0.3.p0.1616399/l
ib/hadoop-mapreduce/hadoop-mapreduce-examples.jar pi -Dmapreduce
.map.env="YARN_CONTAINER_RUNTIME_TYPE=docker,YARN_CONTAINER_RUNT
IME_DOCKER_IMAGE=library/ibmjava:8,YARN_CONTAINER_RUNTIME_DOCKER
_DELAYED_REMOVAL=true,YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS=/etc/
krb5.conf:/etc/krb5.conf:ro" -Dmapreduce.reduce.env="YARN_CONTAI
NER_RUNTIME_TYPE=docker,YARN_CONTAINER_RUNTIME_DOCKER_IMAGE=libr
ary/ibmjava:8,YARN_CONTAINER_RUNTIME_DOCKER_DELAYED_REMOVAL=true
,YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS=/etc/krb5.conf:/etc/krb5.c
onf:ro"  1 40000
```

In case of Spark job, ensure that mount is added as read-only for /etc/krb5.conf as spark.appMasterEnv and spark.executorEnv:

```
 --conf spark.yarn.appMasterEnv.YARN_CONTAINER_RUNTIME_DOCKER_MO
UNTS=/etc/passwd:/etc/passwd:ro,/opt/cloudera/parcels:/opt/cloud
era/parcels:ro,/etc/krb5.conf:/etc/krb5.conf:ro \
```

```
 --conf spark.executorEnv.YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS="
/etc/passwd:/etc/passwd:ro,/opt/cloudera/parcels/:/opt/cloudera/
parcels/:ro,/etc/krb5.conf:/etc/krb5.conf:ro"
```

The ssl-client.xml file and the truststore file is not mounted for Docker containers using MapReduce

Problem statement

Reducer cannot connect to the shuffle service due to SSL handshake issues.

CLI logs:

```
19/11/15 03:26:02 INFO impl.YarnClientImpl: Submitted application
 application_1573810028869_0004
19/11/15 03:26:02 INFO mapreduce.Job: The url to track the job:
 <URL>
19/11/15 03:26:02 INFO mapreduce.Job: Running job: job_1573810
028869_0004
```

```
19/11/15 03:26:12 INFO mapreduce.Job: Job job_1573810028869_0004
 running in uber mode : false
19/11/15 03:26:12 INFO mapreduce.Job:  map 0% reduce 0%
19/11/15 03:26:23 INFO mapreduce.Job:  map 100% reduce 0%
19/11/15 03:27:30 INFO mapreduce.Job: Task Id : attempt_15738100
28869_0004_r_000000_0, Status : FAILED
Error: org.apache.hadoop.mapreduce.task.reduce.Shuffle$ShuffleErr
or: error in shuffle in fetcher#2
    at org.apache.hadoop.mapreduce.task.reduce.Shuffle.run(Shu
ffle.java:136)
    at org.apache.hadoop.mapred.ReduceTask.run(ReduceTask.java:37
7)
    at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:17
4)
    at java.security.AccessController.doPrivileged(AccessControll
er.java:770)
    at javax.security.auth.Subject.doAs(Subject.java:570)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserG
roupInformation.java:1876)
    at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:168
)
Caused by: java.io.IOException: Exceeded MAX_FAILED_UNIQUE_FET
CHES; bailing-out.
    at org.apache.hadoop.mapreduce.task.reduce.ShuffleSchedulerIm
pl.checkReducerHealth(ShuffleSchedulerImpl.java:396)
    at org.apache.hadoop.mapreduce.task.reduce.ShuffleScheduler
Impl.copyFailed(ShuffleSchedulerImpl.java:311)
    at org.apache.hadoop.mapreduce.task.reduce.Fetcher.openShuffl
eUrl(Fetcher.java:291)
    at org.apache.hadoop.mapreduce.task.reduce.Fetcher.copyFromHo
st(Fetcher.java:330)
    at org.apache.hadoop.mapreduce.task.reduce.Fetcher.run(Fetc
her.java:198)
```

NodeManager logs:

```
2019-11-15 03:30:16,323 INFO org.apache.hadoop.yarn.server.nodem
anager.NodeStatusUpdaterImpl: Removed completed containers from
NM context: [container_e149_1573810028869_0004_01_000005]
2019-11-15 03:30:50,812 ERROR org.apache.hadoop.mapred.Shuffle
Handler: Shuffle error:
javax.net.ssl.SSLException: Received fatal alert: certificate_un
known
        at sun.security.ssl.Alerts.getSSLException(Alerts.java
:208)
        at sun.security.ssl.SSLEngineImpl.fatal(SSLEngineImpl.
java:1666)
        at sun.security.ssl.SSLEngineImpl.fatal(SSLEngineImpl.jav
a:1634)
        at sun.security.ssl.SSLEngineImpl.recvAlert(SSLEngineImp
l.java:1800)
        at sun.security.ssl.SSLEngineImpl.readRecord(SSLEngineI
mpl.java:1083)
        at sun.security.ssl.SSLEngineImpl.readNetRecord(SSLEngine
Impl.java:907)
        at sun.security.ssl.SSLEngineImpl.unwrap(SSLEngineImpl.ja
va:781)
        at javax.net.ssl.SSLEngine.unwrap(SSLEngine.java:624)
        at org.jboss.netty.handler.ssl.SslHandler.unwrap(SslHa
ndler.java:1218)
        at org.jboss.netty.handler.ssl.SslHandler.decode(SslHan
dler.java:852)
```

```
        at org.jboss.netty.handler.codec.frame.FrameDecoder.callD
ecode(FrameDecoder.java:425)
        at org.jboss.netty.handler.codec.frame.FrameDecoder.mes
sageReceived(FrameDecoder.java:303)
        at org.jboss.netty.channel.SimpleChannelUpstreamHandler.
handleUpstream(SimpleChannelUpstreamHandler.java:70)
        at org.jboss.netty.channel.DefaultChannelPipeline.sendU
pstream(DefaultChannelPipeline.java:564)
        at org.jboss.netty.channel.DefaultChannelPipeline.sendU
pstream(DefaultChannelPipeline.java:559)
        at org.jboss.netty.channel.Channels.fireMessageReceived
(Channels.java:268)
        at org.jboss.netty.channel.Channels.fireMessageReceived(
Channels.java:255)
        at org.jboss.netty.channel.socket.nio.NioWorker.read(NioW
orker.java:88)
        at org.jboss.netty.channel.socket.nio.AbstractNioWorker.p
rocess(AbstractNioWorker.java:108)
        at org.jboss.netty.channel.socket.nio.AbstractNioSelector
.run(AbstractNioSelector.java:337)
        at org.jboss.netty.channel.socket.nio.AbstractNioWorker.r
un(AbstractNioWorker.java:89)
        at org.jboss.netty.channel.socket.nio.NioWorker.run(Ni
oWorker.java:178)
        at org.jboss.netty.util.ThreadRenamingRunnable.run(Thr
eadRenamingRunnable.java:108)
        at org.jboss.netty.util.internal.DeadLockProofWorker$1
.run(DeadLockProofWorker.java:42)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(T
hreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(T
hreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
2019-11-15 03:30:50,812 ERROR org.apache.hadoop.mapred.ShuffleH
andler: Shuffle error [id: 0xf95ad8ab, /10.65.53.21:44366 => /10
.65.53.21:13562] EXCEPTION: javax.net.ssl.SSLException: Received
 fatal alert: certificate_unknown
2019-11-15 03:30:51,156 INFO org.apache.hadoop.yarn.server.nod
emanager.containermanager.ContainerManagerImpl: Stopping contain
er with container Id: container_e149_1573810028869_0004_01_000006
```

NodeManager logs (Exception):

```
2019-11-15 03:30:50,812 ERROR org.apache.hadoop.mapred.ShuffleHa
ndler: Shuffle error:
javax.net.ssl.SSLException: Received fatal alert: certificate_
unknown
        [...]
2019-11-15 03:30:50,812 ERROR org.apache.hadoop.mapred.Shuffle
Handler: Shuffle error [id: 0xf95ad8ab, /10.65.53.21:44366 => /1
0.65.53.21:13562] EXCEPTION: javax.net.ssl.SSLException: Received
 fatal alert: certificate_unknown
2019-11-15 03:30:51,156 INFO org.apache.hadoop.yarn.server.nodema
nager.containermanager.ContainerManagerImpl: Stopping container
with container Id: container_e149_1573810028869_0004_01_000006
```

Root cause

For normal containers, the file ssl-client.xml defines the SSL settings and it is on the classpath (normally under directory: /etc/hadoop/conf.cloudera.YARN-1/ssl-client.xml). Therefore, it has to be mounted for Docker containers using MapReduce. Since the ssl-client.xml file refers to the truststore file as well, that also had to be mounted.

Resolution

Add the following when running the job:

```
-Dmapreduce.reduce.env=YARN_CONTAINER_RUNTIME_DOCKER_MOUNTS="/et
c/hadoop/conf.cloudera.YARN-1/ssl-client.xml:/etc/hadoop/conf.cl
oudera.YARN-1/ssl-client.xml:ro,/var/lib/cloudera-scm-agent/agen
t-cert/cm-auto-global_truststore.jks:/var/lib/cloudera-scm-agent/
agent-cert/cm-auto-global_truststore.jks:ro"
```

Ensure to add /etc/hadoop/conf.cloudera.YARN-1/ssl-client.xml and /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks to the Allowed Read-Only Mounts in Cloudera Manager.

Note, that the location of the truststore can vary, so verify its location from the ssl-client.xml file. You can access that file in Clouder Manager through the Processes view for NodeManager.

Troubleshooting on YARN

General troubleshooting procedures to diagnose some of the commonly encountered issues in YARN.

The Kill application button does not display in the YARN UI

Problem statement

The YARN UI does not display the Kill application button.

Root cause

Kerberos is not enabled.

Resolution

Enable Kerberos in order to view the Kill application button.

With the application state API, you can query the state of a submitted app as well as kill a running app by modifying the state of a running app using a PUT request with the state set to “KILLED”. To perform the PUT operation, authentication has to be set up for the Resource Manager (RM) web services. See Enabling Kerberos Authentication for Cloudera linked below.

YARN Queue Manager UI behavior in mixed resource allocation mode

The mixed resource allocation mode in YARN is only supported through safety valves. If you open the Queue Manager UI or try to access Queue Manager APIs when mixed calculation mode is turned on, Queue Manager blocks the UI or APIs and informs you that mixed calculation mode is turned on, Queue Manager is inaccessible until this feature is fully compatible.

If the flag `yarn.scheduler.capacity.legacy-queue-mode.enabled` is not set or the property is missing, mixed resource allocation mode is not enabled in YARN. The detection of mixed allocation mode is determined by this flag in the capacity scheduler safety valve XML.

For more details on Mixed resource allocation, see the links provided below.

Scenarios

The following scenarios describe how the YARN Queue Manager UI behaves in mixed resource allocation mode in YARN.

- If YARN mixed resource allocation mode is activated or detected while the YARN Queue Manager is already running, it causes YARN Queue Manager UI to fail to load, resulting in an error message being displayed in the UI.

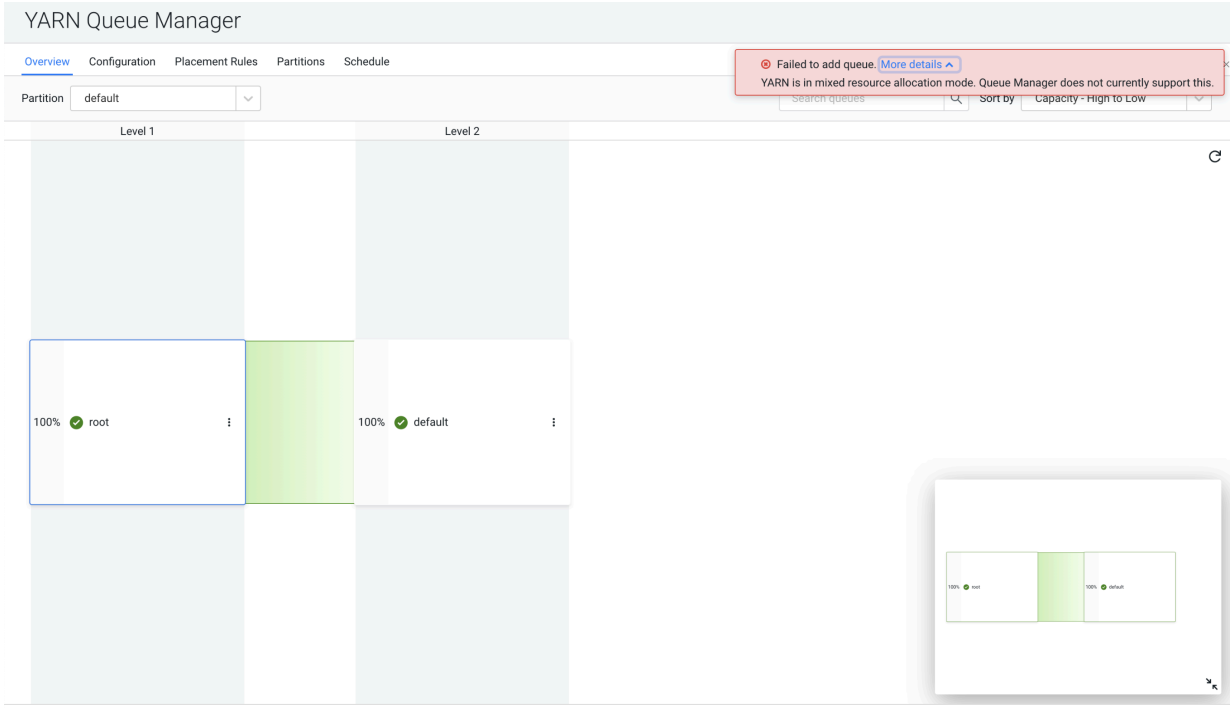

- If you start the YARN Queue Manager while YARN is already in mixed resource calculation mode, Queue Manager UI and APIs are blocked displaying the following error message.

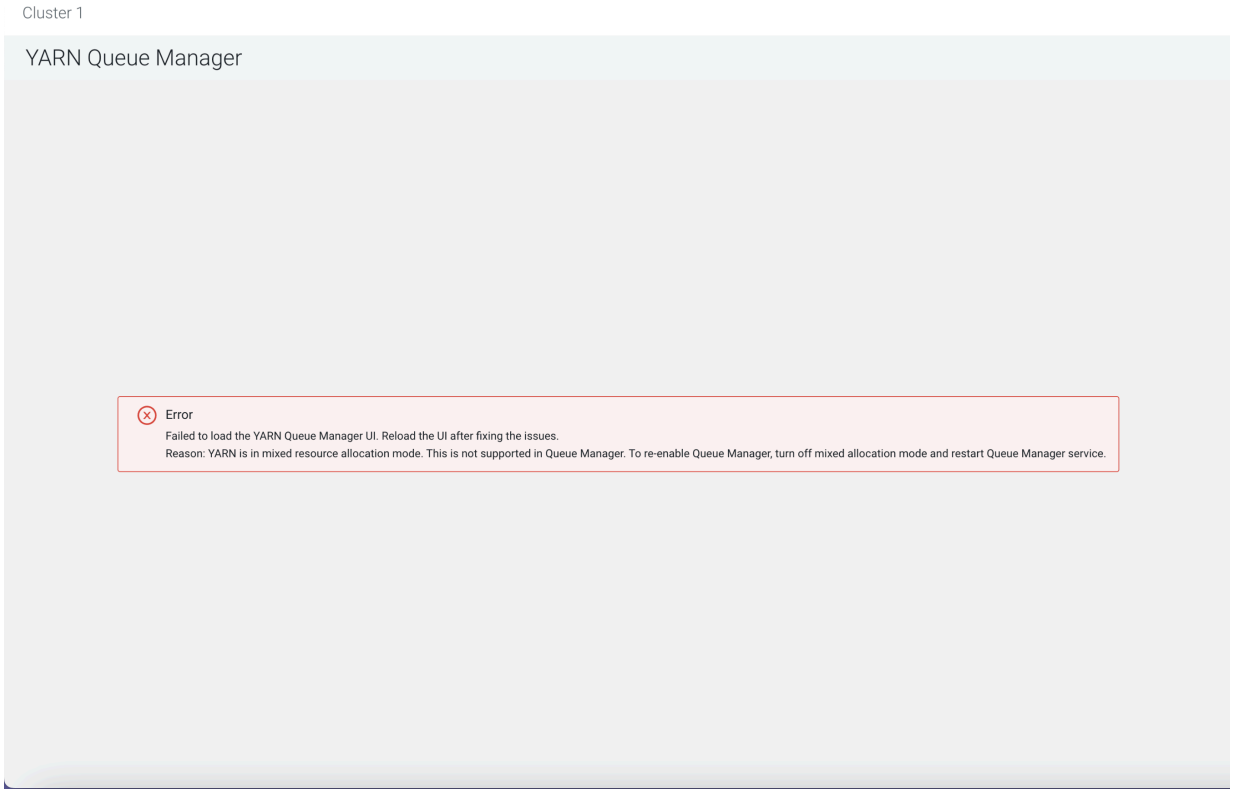

Related Information

Mixed resource allocation mode (Technical Preview)

Troubleshooting for mixed resource allocation mode in YARN Queue Manager

You can reset YARN Queue Manager UI and restore its functionality to troubleshoot YARN Queue Manager UI behavior in mixed resource allocation mode from Cloudera Manager.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Go to the Configuration tab.
3. If the flag is not set or the property is missing, mixed resource allocation mode will not be enabled in YARN. Set the following property yarn.scheduler.capacity.legacy-queue-mode.enabled value to 'true' in the Capacity Scheduler Configuration Advanced Configuration Snippet (Safety Valve) field to use the mixed allocation mode:

```
<property>
<name>yarn.scheduler.capacity.legacy-queue-mode.enabled</name>
<value>false</value>
</property>
```

4. Click Save Changes.
5. Click Actions Restart to restart YARN.
6. Click Actions Restart to restart YARN Queue Manager.

Troubleshooting Linux Container Executor

A list of numeric error codes communicated by the container-executor to the NodeManager that appear in the /var/log/hadoop-yarn NodeManager log.

Table 1: Numeric error codes that are applicable to the container-executor in YARN, but are used by the LinuxContainerExecutor only.

Numeric Code	Name	Description
1	INVALID_ARGUMENT_NUMBER	• Incorrect number of arguments provided for the given container-executor command • Failure to initialize the container localizer
2	INVALID_USER_NAME	The user passed to the container-executor does not exist.
3	INVALID_COMMAND_PROVIDED	The container-executor does not recognize the command it was asked to run.
5	INVALID_NM_ROOT	The passed NodeManager root does not match the configured NodeManager root (yarn.nodemanager.local-dirs), or does not exist.
6	SETUID_OPER_FAILED	Either could not read the local groups database, or could not set UID or GID
7	UNABLE_TO_EXECUTE_CONTAINER_SCRIPT	The container-executor could not run the container launcher script.
8	UNABLE_TO_SIGNAL_CONTAINER	The container-executor could not signal the container it was passed.
9	INVALID_CONTAINER_PID	The PID passed to the container-executor was negative or 0.

Numeric Code	Name	Description
18	OUT_OF_MEMORY	The container-executor couldn't allocate enough memory while reading the container-executor.cfg file, or while getting the paths for the container launcher script or credentials files.
20	INITIALIZE_USER_FAILED	Couldn't get, stat, or secure the per-user NodeManager directory.
21	UNABLE_TO_BUILD_PATH	The container-executor couldn't concatenate two paths, most likely because it ran out of memory.
22	INVALID_CONTAINER_EXEC_PERMISSIONS	The container-executor binary does not have the correct permissions set.
24	INVALID_CONFIG_FILE	The container-executor.cfg file is missing, malformed, or has incorrect permissions.
24	Error starting NodeManager	NodeManager can fail to start up if the nosuid option is set on the file system where the container-executor binary resides. nosuid prevents the setuid bit on executable from taking effect. This means that the container-executor binary that has the setuid bit set with "root" privileges, is unable to access the container-executor.cfg configuration file owned by "root" and results in error.
25	SETSID_OPER_FAILED	Could not set the session ID of the forked container.
26	WRITE_PIDFILE_FAILED	Failed to write the value of the PID of the launched container to the PID file of the container.
255	Unknown Error	This error has several possible causes. Some common causes are: • User accounts on your cluster have a user ID less than the value specified for the min.user.id property in the cont ainer-executor.cfg file. The default value is 1000 which is appropriate on Ubuntu systems, but may not be valid for your operating system. For information about setting min. user.id in the container-executor.cfg file. • This error is often caused by previous errors; look earlier in the log file for possible causes.

Table 2: Exit status codes apply to all containers in YARN. These exit status codes are part of the YARN framework and are in addition to application specific exit codes that can be set.

Numeric Code	Name	Description
0	SUCCESS	Container has finished succesfully.
-1000	INVALID	Initial value of the container exit code. A container that does not have a COMPLETED state will always return this status.
-100	ABORTED	Containers killed by the framework, either due to being released by the application or being 'lost' due to node failures, for example.
-101	DISKS_FAILED	Container exited due to local disks issues in the NodeManager node. This occurs when the number of good nodemanager-local-directories or nodemanager-log-directories drops below the health threshold.
-102	PREEMPTED	Containers preempted by the framework. This does not count towards a container failure in most applications.

Numeric Code	Name	Description
-103	KILLED_EXCEEDED_VMEM	Container terminated because of exceeding allocated virtual memory limit.
-104	KILLED_EXCEEDED_PMEM	Container terminated because of exceeding allocated physical memory limit.
-105	KILLED_BY_APPMASTER	Container was terminated on request of the application master.
-106	KILLED_BY_RESOURCEMANAGER	Container was terminated by the resource manager.
-107	KILLED_AFTER_APP_COMPLETION	Container was terminated after the application finished.

Cloudera Runtime 7.3.1

Using YARN Web UI and CLI

Date published: 2020-07-28
Date modified: 2024-12-10

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Accessing the YARN Web User Interface....4

Viewing the Cluster Overview.... 4

Viewing nodes and node details....5

Viewing queues and queue details....6

Viewing all applications.... 8

Searching applications....9

Viewing application details....10

UI Tools....11

Using the YARN CLI to viewlogs for applications.... 11

Accessing the YARN Web User Interface

Access the YARN Web User Interface to monitor clusters, queues, applications, services, and flow activities.

About this task

You can access the YARN Web User Interface from Cloudera Manager to monitor the cluster.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Web UI tab.
3. Select the ResourceManager Web UI.
   The ResourceManager Web UI opens in a new browser window.

   Note, that in a ResourceManager High Availability configuration, if you select the non-active ResourceManager the link directs to the web UI of the active ResourceManager.

Viewing the Cluster Overview

The Cluster Overview page shows cluster resource usage by applications and queues, information about finished and running applications, and usage of memory and vCores in the cluster.

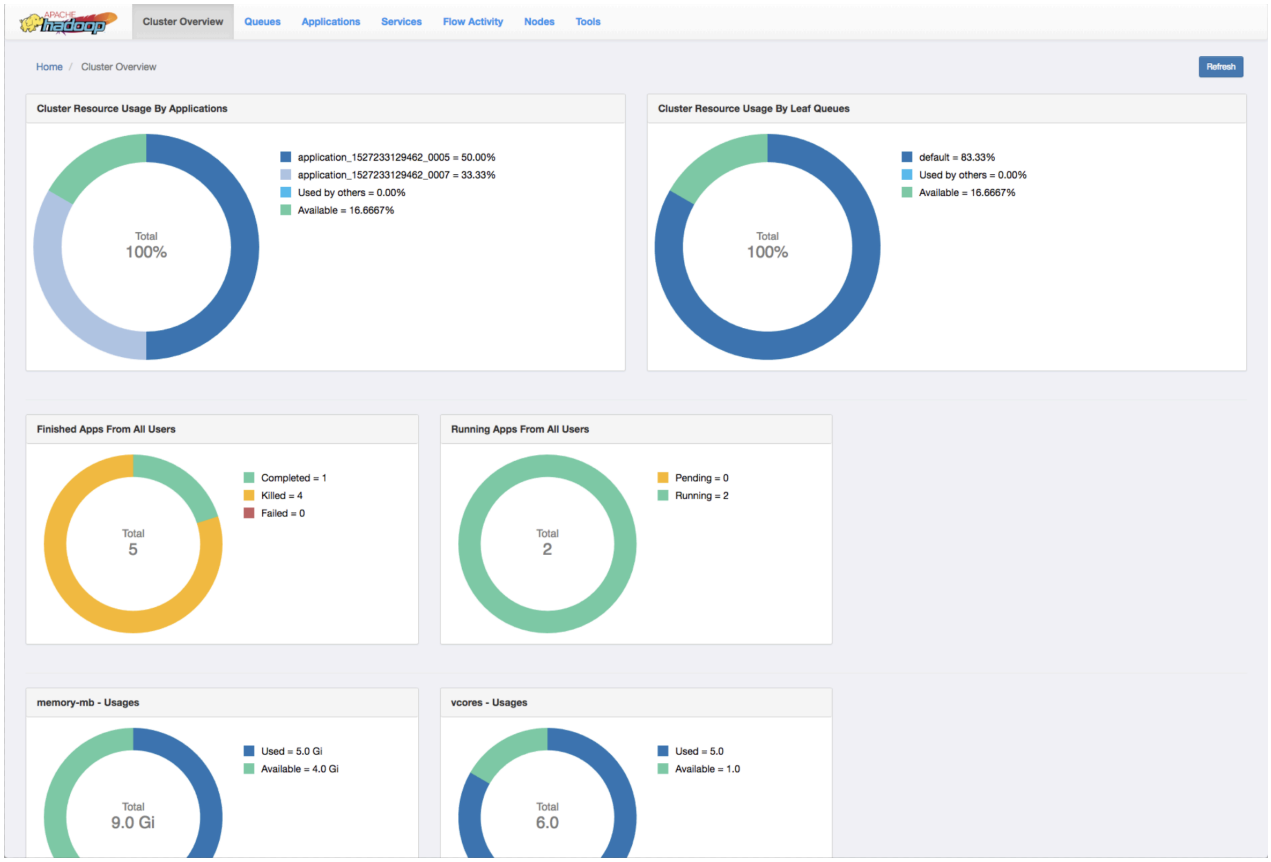

Cluster Resource Usage by Applications

Displays the percentage of cluster resources in use by applications and the percentage available for usage.

Cluster Resource Usage by Leaf Queues

Displays the percentage of cluster resources in use by leaf queues and the percentage available for usage.

Finished Apps From All Users

Displays the number of completed, killed, and failed applications.

Monitor Running Apps

Displays the number of pending and running applications.

memory-mb - Usages

Displays the amount of used and available memory.

vcores - Usages

Displays the number of used and available virtual cores.

Monitor Node Managers

Displays the status of the Node Managers under the following categories:

- Active
- Unhealthy
- Decommissioning
- Decommissioned

Viewing nodes and node details

The Nodes page on the YARN Web User Interface enables you to view information about the cluster nodes on which the NodeManagers are running.

The Nodes page displays details under the following headers: Information, Node Status and Nodes Heatmap Chart.

Information

The Information tab displays the node details as shown in the following figure:

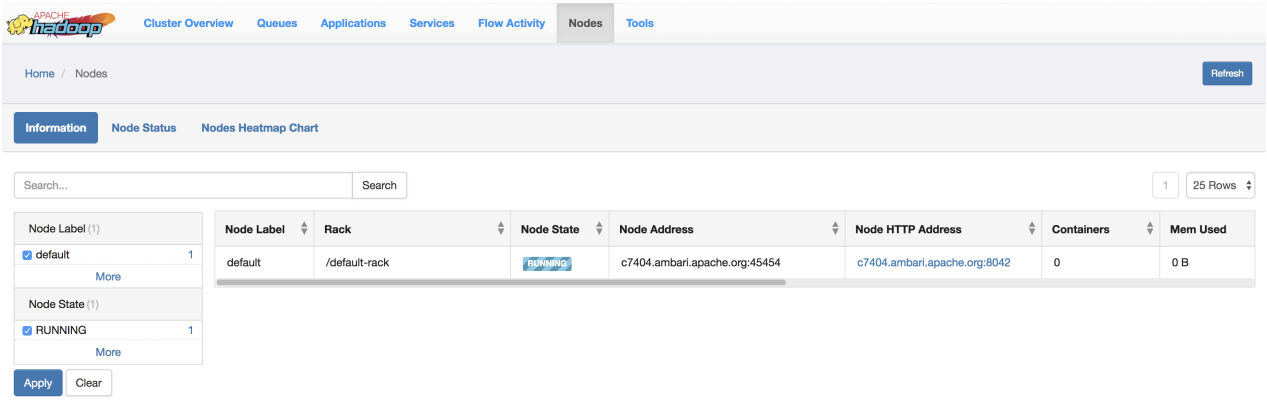

You can sort through any of the columns to view the details of the required nodes. You can also search to find the specific node labels from the entire list.

Node Manager page

To view the Node Manager page of any node label, click the corresponding Node HTTP address. The Node Manager page displays details of a node as shown in the following figure:

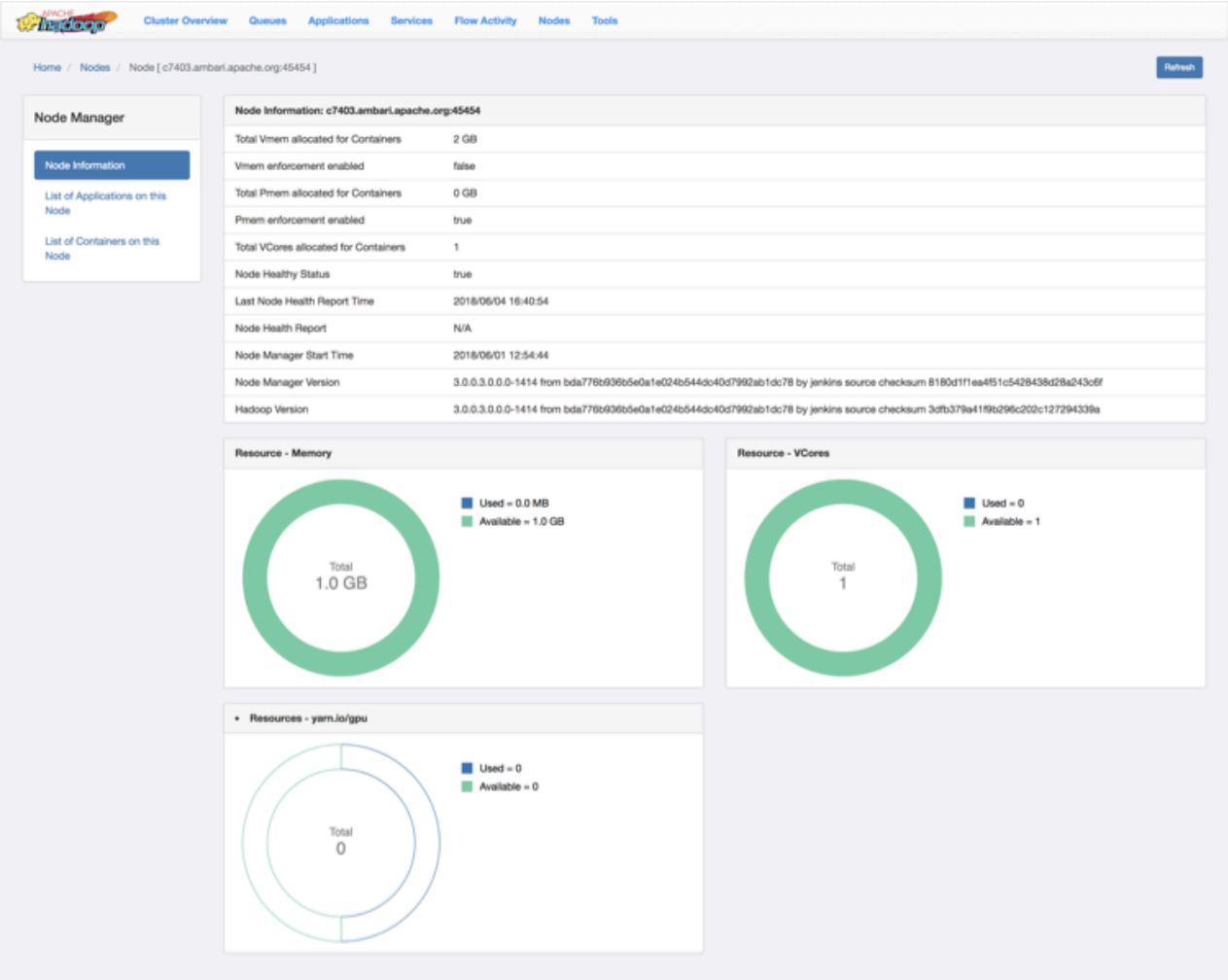

You can also view the resource usage in the following categories in a pie-chart representation:

1. Memory
2. VCores
3. yarn-io/GPU

Node status

This tab displays the node managers in a pictorial representation. It displays details such as the number of active nodes, number of unhealthy nodes, decommissioning nodes, and the number of decommissioned node managers.

Nodes heatmap chart

This tab graphically displays nodes on the basis of their usage of memory. You can enter host or rack details in the search bar to filter nodes.

Viewing queues and queue details

The Queues page is scheduler dependent and displays details of YARN queues. You can either view queues from all the partitions or filter to view queues of a partition.

Capacity Scheduler

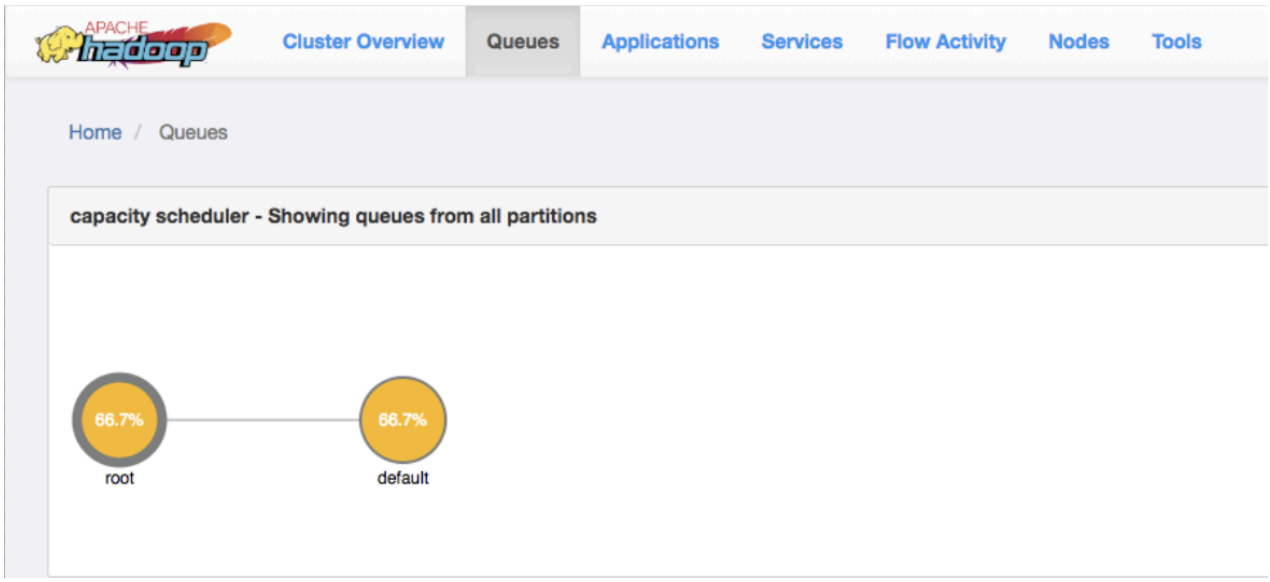

View queue details

In the capacity scheduler view, click the circle that represents a particular queue. The right column of the page gets updated with details of that queue.

The following example shows the details of a queue:

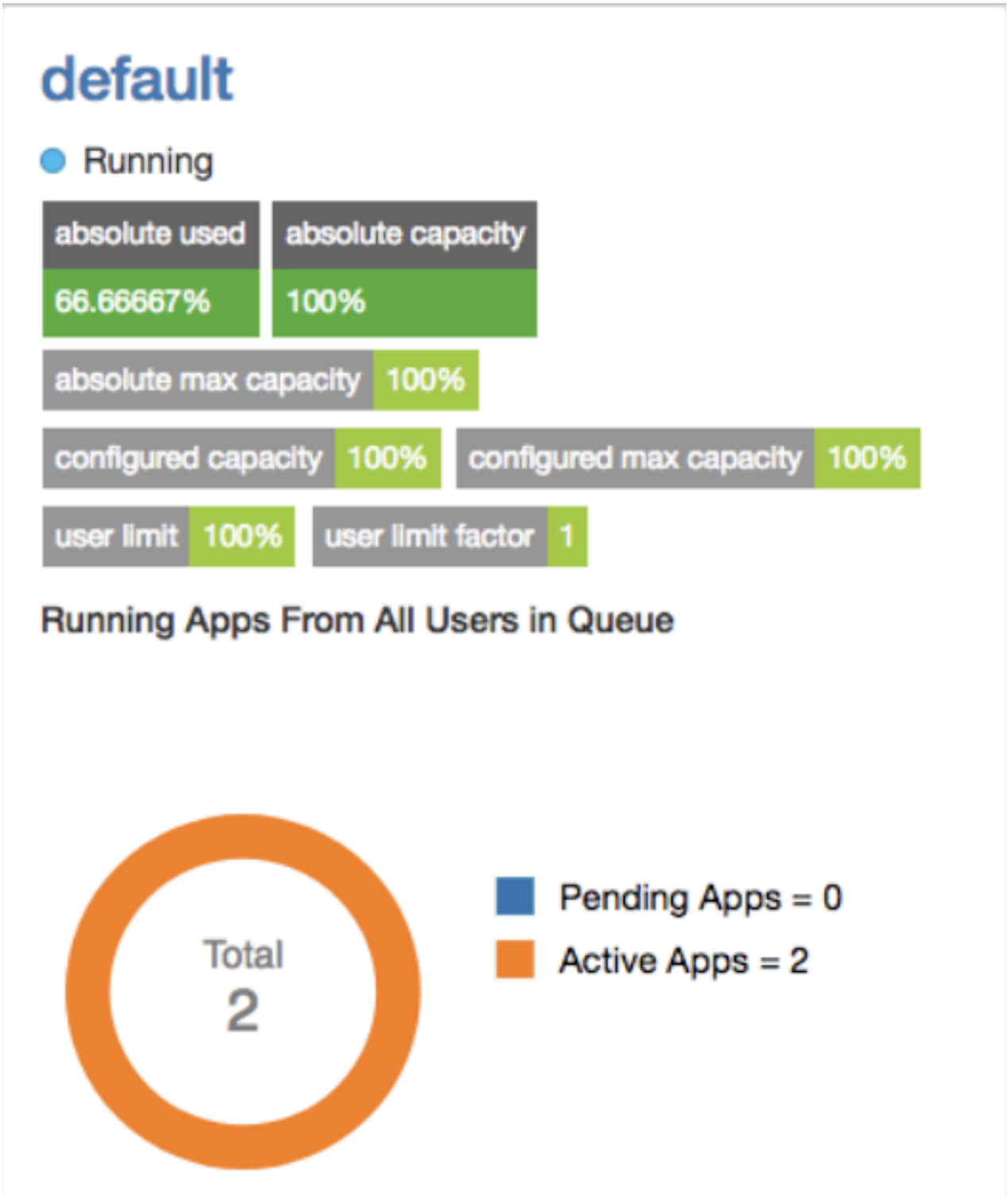

You can double-click the queue to view its details on a separate page. You can also view details of any application submitted to that queue by clicking its corresponding Application ID.

Viewing all applications

You can search for applications and view their details using the YARN Web User Interface.

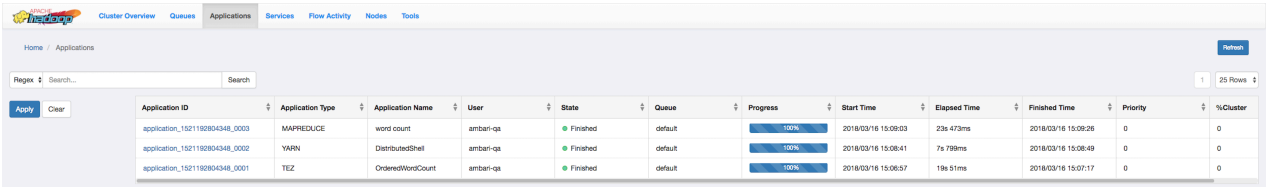

The Applications page displays details of the YARN applications in a tabular form.

- Application ID: The identifier for the application.
- Application Type: Specifies the application type for Mapreduce, YARN, TEZ, or other applications.
- Application Name: Provides the name of the application
- User: The name of the user who is the owner of the application.
- State: The running state of the application.
- Queue: Specifies the name of the queue to which the application belongs.
- Progress: The progress of the application in a percentage display.
- Start Time: The time when an application run started.
- Elapsed Time: The time taken for the application run.
- Finished Time: The time of completion of the application run.
- Priority: The priority of running the application.
- %Cluster: The percentage of cluster resources used by the application run.

Searching applications

The Applications page displays the list of YARN applications in a tabular form. You can apply search filters on this list to show only those applications that match the search criteria.

About this task

You can specify the search criteria either as regular expressions or SQL statements.

Procedure

1. On the Applications page, select either Regex or SQL from the drop-down list depending on the type of search you want to perform.
2. In the Search box, specify the search criteria.

Search Criteria	Description
Regex	Lists the applications whose details match the search criterion specified as a regular expression. For example, if you want to view application runs longer than an hour, mention the regular expression ^hso that the YARN UI shows only those applications that mention the Elapsed Time value in terms of hours, minutes, and seconds.
SQL	Lists the applications whose details match the search criterion specified as a SQL statement. For example, if you want to view all the applications submitted by the user yarn, specify "User"='yarn' as the search criterion.

3. Click Search to view details of the applications matching the search criteria.

Note:

- Apart from specifying search criteria to filter the list of applications, you can also select or clear the State and Queue check-boxes to view a specific set of applications depending on your requirements.
- You can sort the application entries in ascending or descending order by clicking the corresponding arrow next to any column header in the table.

Viewing application details

Clicking a YARN application on the Applications page displays its details.

You can view the following details for the selected application:

- Application Attempts
- Resource Usage
- Diagnostics
- Logs

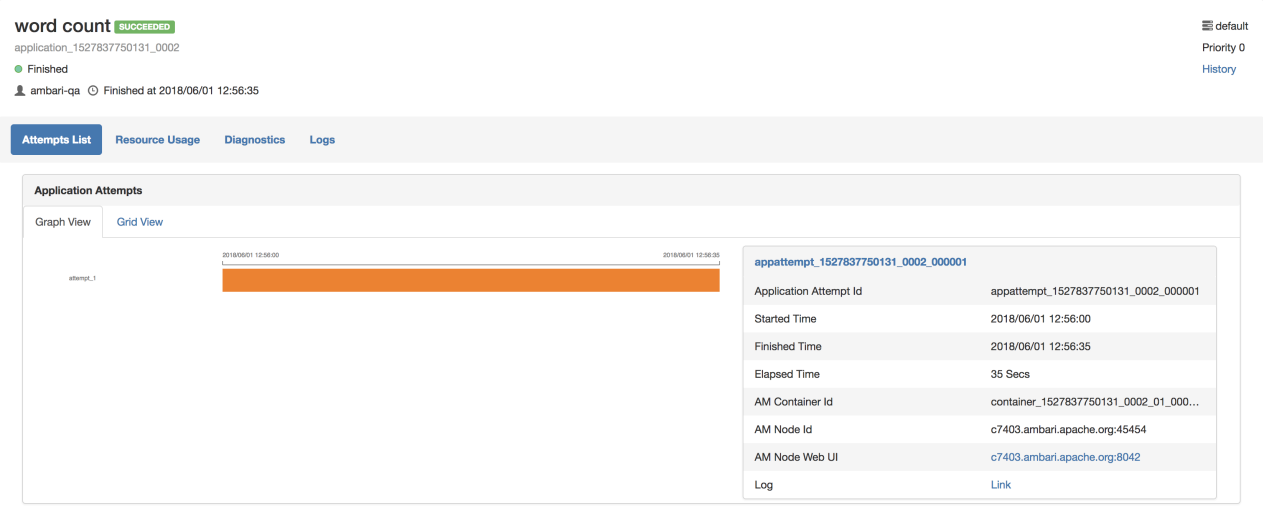

Application Attempts

You can view the attempts in Graph View and Grid View.

Graph View

A graph displays the start time and finish time of the attempt. You can also find the details of the attempt such as application attempt ID, started time, finished time, elapsed time, AM Container ID, and AM Node ID in the form of a table. You can access the Node UI using the AM Node Web UI. You can also view the log by clicking on the log link.

Grid View

A table displays the details of the application attempts. You can find the details of the attempt such as application attempt ID, started time, finished time, elapsed time, AM Container ID, and AM Node ID. You can access the Node UI using the AM Node Web UI. You can also view the log by clicking on the log link.

Resource Usage

This tab displays the resources used by the application attempts.

Diagnostics

Use this tab to view the diagnostics details of the application attempts. You can view any outstanding resource requests, and scheduling information.

Logs

Use this tab to view logs specific to containers. Select an attempt from the dropdown list and select the specific container to view the desired logs.

UI Tools

You can view the YARN configuration and YARN Daemon logs on the Tools page.

YARN Configuration

You can see the values of the properties defined in the following configuration files:

1. Core Configuration: Details of properties defined in the core-default.xml file.
2. YARN Configuration: Details of properties defined in the yarn-default.xml file.
3. MapReduce Configuration: Details of the properties defined in the mapred-default.xml file.

YARN Daemon Logs

You can view the list of log files. Click on a log file to view its contents in another tab of your browser.

Using the YARN CLI to viewlogs for applications

Use the YARN CLI to view logs for running application. Configure the log aggregation to aggregate and write out logs for all containers belonging to a single Application grouped by NodeManagers to single log files at a configured location in the file system.

You can access container log files using the YARN ResourceManager web UI, but more options are available when you use the yarn logs CLI command.

View all log files for an application

Use the following command format to view all logs for an application:

```
yarn logs -applicationId <Application ID>
```

View a specific log type for an application

Use the following command format to view all logs of a particular type for an application:

```
yarn logs -applicationId <Application ID> -log_files <log_file_type>
```

For example, to view only the stderr error logs:

```
yarn logs -applicationId <Application ID> -log_files stderr
```

The -logFiles option also supports Java regular expressions, so the following format would return all types of log files:

```
yarn logs -applicationId <Application ID> -log_files .*
```

View ApplicationMaster log files

Use the following command format to view all ApplicationMaster container log files for an application:

```
yarn logs -applicationId <Application ID> -am ALL
```

Use the following command format to view only the first ApplicationMaster container log files:

```
yarn logs -applicationId <Application ID> -am 1
```

List container IDs

Use the following command format to list all container IDs for an application:

```
yarn logs -applicationId <Application ID> -show_application_log_info
```

View log files for one container

Once you have the container IDs, you can use the following command format to list the log files for a particular container:

```
yarn logs -applicationId <Application ID> -containerId <Container ID>
```

Show container log file information

Use the following command format to list all of the container log file names (types) for an application:

```
yarn logs -applicationId <Application ID> -show_container_log_info
```

You can then use the -logFiles option to view a particular log type.

View a portion of the log files for one container

For large container log files, you can use the following command format to list only a portion of the log files for a particular container:

```
yarn logs -applicationId <Application ID> -containerId <Container ID> -size
<bytes>
```

To view the first 1000 bytes:

```
yarn logs -applicationId <Application ID> -containerId <Container ID> -size
1000
```

To view the last 1000 bytes:

```
yarn logs -applicationId <Application ID> -containerId <Container ID> -size
-1000
```

Download logs for an application

Use the following command format to download logs to a local folder:

```
yarn logs -applicationId <Application ID> -out <path_to_local_folder>
```

The container log files are organized in parent folders labeled with the applicable node ID.

Display help for YARN logs

To display Help for yarn logs, run the following command:

```
yarn logs -help
```

Cloudera Runtime 7.2.11

Configuring Apache ZooKeeper

Date published: 2019-10-28
Date modified: 2021-09-09

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enable the AdminServer.....4

Configure four-letter-word commands in ZooKeeper.....4

Enable the AdminServer

AdminServer is a diagnostics REST interface that provides simple diagnostics and metrics through HTTP. It is disabled by default, but you can enable it and change the server port using Cloudera Manager.

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Search for admin.
4. Find Enable ZooKeeper Admin Server and select the role configuration groups on which you want to allow the AdminServer.
5. Find Admin Server Portal and change the port number if necessary.

   Its default value is 5181.
6. Click Save Changes.

Configure four-letter-word commands in ZooKeeper

Four-letter-words commands are simple diagnostics commands that can be used over telnet for ZooKeeper. These commands can be configured with Cloudera Manager.

About this task

You can configure which four-letter-words commands you want to enable in ZooKeeper. By default the wchp and the wchc commands are disabled.

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Search for 4lw.
4. Find the ZooKeeper 'Four Letter Word' Command Whitelist property.
5. Use the minus and plus icon to remove or add four-letter-word commands to the allowlist.

   By default the following commands are on the allowlist: conf, cons, crst, dirs, dump, envi, gtmk, ruok, stmk, srst, stat, wchs, mntr, and isro.

   The wchp and wchc commands are not enabled by default because of their known DOS vulnerability. For more information, see ZOOKEEPER-2693.

   If * is set, all commands are enabled.
6. Click Save Changes.

Cloudera Runtime 7.2.11

Managing Apache ZooKeeper Security

Date published: 2019-09-23
Date modified: 2021-09-09

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

ZooKeeper Authentication....4
Configure ZooKeeper server for Kerberos authentication....4
Configure ZooKeeper client shell for Kerberos authentication....4
Verify the ZooKeeper authentication....4
Enable server-server mutual authentication....5
Use Digest Authentication Provider in ZooKeeper....6

Configure ZooKeeper TLS/SSL using Cloudera Manager....6

ZooKeeper ACLs Best Practices....7
ZooKeeper ACLs Best Practices: Atlas....7
ZooKeeper ACLs Best Practices: Cruise Control....8
ZooKeeper ACLs Best Practices: HBase....8
ZooKeeper ACLs Best Practices: HDFS....9
ZooKeeper ACLs Best Practices: Kafka....9
ZooKeeper ACLs Best Practices: Oozie....11
ZooKeeper ACLs Best Practices: Ranger....11
ZooKeeper ACLs best practices: Search....12
ZooKeeper ACLs Best Practices: YARN....15
ZooKeeper ACLs Best Practices: ZooKeeper....15

ZooKeeper Authentication

ZooKeeper supports mutual server-to-server (quorum peer) authentication using SASL (Simple Authentication and Security Layer), which provides a layer around Kerberos authentication.

Server to server authentication among ZooKeeper servers in an ensemble mitigates the risk of spoofing by a rogue server on an unsecured network. For more information about quorum peer authentication and how the feature leverages ZooKeeper's SASL support, see the Cloudera Engineering Blog post, Hardening Apache ZooKeeper Security.

Configuring ZooKeeper to use Kerberos for client-server or server-server authentication requires that your organization's Kerberos instance (MIT Kerberos, Microsoft Active Directory) be up and running, and reachable by the ZooKeeper server or client during the configuration processes detailed below.

Before enabling mutual authentication, the ZooKeeper servers in the cluster must be configured to authenticate using Kerberos.

Cloudera recommends that you ensure your ZooKeeper ensemble is working properly, before you attempt to integrate Kerberos authentication.

Configure ZooKeeper server for Kerberos authentication

You can configure the ZooKeeper server for Kerberos authentication in Cloudera Manager.

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Search for kerberos.
4. Find the Enable Kerberos Authentication property.
5. Select the ZooKeeper services that you want to configure for Kerberos authentication.
6. Click Save Changes.

Configure ZooKeeper client shell for Kerberos authentication

The zookeeper-client shell script is automatically configured for Kerberos authentication if Kerberos is enabled in Cloudera Manager. You do not need to do any manual configurations.

Verify the ZooKeeper authentication

After enabling Kerberos authentication and restarting the ZooKeeper cluster, you can verify that the ZooKeeper authentication is working correctly.

Procedure

1. Start the ZooKeeper client, passing to it the name of a ZooKeeper server:

```
zookeeper-client -server FQDN.EXAMPLE.COM:port
```

2. From the ZooKeeper CLI, create a protected znode using your ZooKeeper client principal.

```
create /znode1 znode1data sasl:zkcli:cdwra
```

Cloudera Manager by default maps the Kerberos principal to its short name by setting two parameters in ZooKeeper's service configuration file zoo.cfg:

```
kerberos.removeHostFromPrincipal=true
kerberos.removeRealmFromPrincipal=true
```

With the help of above settings, for example, the client principal zkcli/myhost@EXAMPLE.COM will be authenticated in ZooKeeper as zkcli.

3. Verify the znode created and the ACL is set correctly:

```
getAcl/znode1
```

The getAcl command returns the znode's scheme and permission values.

4. Verify that the znode's scheme and permissions values are as expected.

Enable server-server mutual authentication

You can enable mutual authentication between multiple ZooKeeper Servers.

About this task

Support for mutual authentication between ZooKeeper Servers can be enabled through the Cloudera Manager Admin Console. For secured networks, server-to-server authentication is considered an optional security enhancement, so the capability is disabled by default:

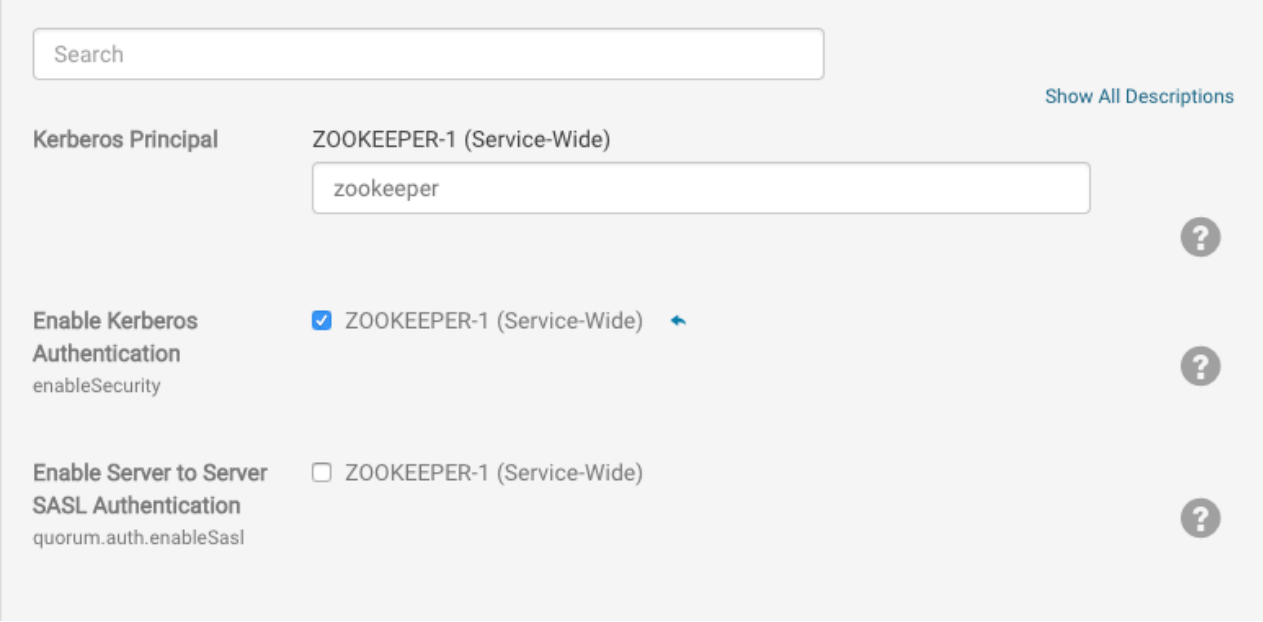

Server-to-server SASL authentication requires all servers in the ZooKeeper ensemble to authenticate using Kerberos.

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Search for sasl.

4. Find the Enable Server to Server SASL Authentication and select it.
5. Click Save Changes.
6. Click the Actions button.
7. Select Restart.

Use Digest Authentication Provider in ZooKeeper

Cloudera recommends to override the default SHA1 algorithm and use a more secure one when using the Digest Authentication Provider in ZooKeeper.

About this task

When using the Digest Authentication Provider in ZooKeeper, for example for specifying SuperUser, it is recommended to override the default SHA1 algorithm and use a more secure one, such as SHA3-384 or SHA3-512. This is mandatory for FIPS compliant clusters.

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Find the Java Configuration Options for ZooKeeper Server.
4. Add the following configuration:

   ```
   -Dzookeeper.DigestAuthenticationProvider.digestAlg=[***ALGORITHM***]
   ```

   For example:

   ```
   -Dzookeeper.DigestAuthenticationProvider.digestAlg=SHA3-512
   ```

Configure ZooKeeper TLS/SSL using Cloudera Manager

TLS/SSL encryption between the ZooKeeper client and the ZooKeeper server and within the ZooKeeper Quorum is supported.

About this task

The ZooKeeper TLS/SSL feature has the following limitations:

- In each ZooKeeper server process the same ZooKeeper Server KeyStore / TrustStore configuration is used for both QuorumSSL and ClientSSL.
- HTTPS for the ZooKeeper REST Admin Server is still not supported. Even if you enable SSL for ZooKeeper, the AdminServer will still use HTTP only.

TLS/SSL encryption is automatically enabled when AutoTLS is enabled. As a result it is enabled by default in Data Hub cluster templates.

You can disable, enable and configure ZooKeeper TLS/SSL manually using Cloudera Manager:

Procedure

1. In Cloudera Manager, select the ZooKeeper service.
2. Click the Configuration tab.
3. Search for SSL.

4. Find the Enable TLS/SSL for ZooKeeper property and select it to enable TLS/SSL for ZooKeeper.

   When TLS/SSL for ZooKeeper is enabled, two ZooKeeper features get enabled:

   - QuorumSSL: the ZooKeeper servers are talking to each other using secure connection.
   - ClientSSL: the ZooKeeper clients are using secure connection when talking to the ZooKeeper server.
5. Find the Secure Client Port property and change the port number if necessary.

   When ClientSSL is enabled, a new secure port is opened on the ZooKeeper server which handles the SSL connections. Its default value is 2182.

   The old port (default: 2181) will still be available for unsecured connections. Unsecure port cannot be disabled in Cloudera Manager, because most components cannot support ZooKeeper TLS/SSL yet.
6. Click Save Changes.
7. Restart.

What to do next

Enable and configure ClientSSL by other components that support this feature.

The following components support ZooKeeper TLS/SSL:

- HBase Indexer (KS Indexer)
- Kafka
- Oozie
- Solr

ZooKeeper ACLs Best Practices

As more and more components begin to rely on ZooKeeper within a Hadoop cluster, there are various permissions that need to be maintained to ensure the integrity and security of the znodes. These permissions are different from component to component. You must follow the required steps for tightening the ZooKeeper ACLs or permissions when provisioning a secure cluster as a best practices guideline.

Some components only use ZooKeeper when they are running in their component specific HA mode. Others have separate secure and unsecure ACLs defined and switch between which to enforce based on the component knowledge of whether the cluster is secured or not.

In general, ACLs are pretty open and assume an unsecure cluster by default. These permissions need to be hardened for secure clusters in order to avoid inappropriate access or modification of this critical platform state.

Unaffected Components

The following components require no action:

- Calcite
- Knox
- MapReduce
- Spark
- Tez
- Zeppelin

ZooKeeper ACLs Best Practices: Atlas

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for Atlas when provisioning a secure cluster.

- ZooKeeper Usage:
  - /apache_atlas - Root zookeeper node which is configured for curator, under which nodes for leader election are created.
  - /apache_atlas/active_server_info - Znode used in HA environments for storing active server information.
  - /apache_atlas/setup_in_progress - Transient Znode used to ensure some setup steps are executed only from one instance. This gets deleted after use and should normally not be seen.
  - /leader_elector_path and /setup_lock - Transient Znodes.
  - /apache_atlas is the root node which is configuration driven. The default value is /apache_atlas but can be overwritten by setting the value for this property atlas.server.ha.zookeeper.zkroot
- Default ACLs:
  - All znodes have world:anyone:cdrwa by default.
- Security Best Practice ACLs/Permissions and Required Steps:
  - No user intervention is required for creating/using the Znodes. They are all managed internally by Atlas. Atlas exposes two configuration properties that define the auth and ACL - to use while creating these Znodes.

    Cloudera Manager should configure these correctly for a secure cluster. The recommended configuration is atlas.server.ha.zookeeper.auth=sasl:atlas@<domain.com> andatlas.server.ha.zookeeper.acl=sasl:atlas@<domain.com>, where <domain.com> should be replaced with the right value of the Atlas service user principal. (Assuming atlas is the service user name). When set this way, the ACLs for all Znodes will be atlas.server.ha.zookeeper.acl=sasl:atlas@<domain.com>:cdrwa.

    Note: Configuring permissions from Cloudera Manager is not allowed.

    .

ZooKeeper ACLs Best Practices: Cruise Control

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for Cruise Control when provisioning a secure cluster.

Cruise Control currently logs into Zookeeper with Kafka credentials. This configuration should not be changed as it will render Cruise Control unable to operate.

A Custom ZooKeeper node, /CruiseControlBrokerList, is created under the root node of Kafka to store the failed broker information in case of Kafka broker failure.

For more information about configuring ZooKeeper ACLs for Kafka, see the ZooKeeper ACLs Best Practices: Kafka section.

Related Information

ZooKeeper ACLs Best Practices: Kafka

ZooKeeper ACLs Best Practices: HBase

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for HBase when provisioning a secure cluster.

- ZooKeeper Usage:
  - /hbase - Default ZNode for unsecured and secured clusters

    Note: The ZNodes do not contain any sensitive information, only metadata.

- Default ACLs:
  - In unsecured setup /hbase - world:anyone:cdrwa
    - All children ZNodes are also world cdrwa
  - Open for global read, write protected: world:anyone:r, sasl:hbase:cdrwa
    - /hbase
    - /hbase/master
    - /hbase/meta-region-server
    - /hbase/hbaseid
    - /hbase/table
    - /hbase/rs
  - No global read, r/w protected: sasl:hbase:cdrwa:
    - /hbase/acl
    - /hbase/namespace
    - /hbase/backup-masters
    - /hbase/online-snapshot
    - /hbase/draining
    - /hbase/replication
    - /hbase/region-in-transition
    - /hbase/splitWAL
    - /hbase/table-lock
    - /hbase/recovering-regions
    - /hbase/running
    - /hbase/tokenauth
  - Security Best Practice ACLs/Permissions and Required Steps:
    - HBase code determines which ACL to enforce based on the configured security mode of the cluster/hbase. Users are not expected to perform any modification of ZooKeeper ACLs on ZNodes and users should not alter any ACLs by hand.

ZooKeeper ACLs Best Practices: HDFS

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for HDFS when provisioning a secure cluster.

- ZooKeeper Usage:
  - hadoop-ha: Default ZNode for unsecured and secured clusters.
- Default ACLs:
  - In an unsecured deployment, the default ACL is world: anyone: cdrwa
  - In a secured deployment, the default ACL is digest: hdfs-fcs: cdrwa
- Security Best Practice ACLs/Permissions and Required Steps:
  - HDFS ZNodes are protected with digest authentication by default in a secure CDP cluster. You need not modify Zookeeper ACLs on HDFS ZNodes or alter any ACLs by hand.

ZooKeeper ACLs Best Practices: Kafka

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for Kafka when provisioning a secure cluster.

- ZooKeeper Usage:
  - /controller - Kafka Znode for controller leader election
  - /cluster - Contains a unique id for the Kafka cluster
  - /brokers - Kafka Znode for broker metadata
  - /kafka-acl - Kafka ZNode for SimpleAclAuthorizer ACL storage
  - /admin - Kafka admin tool metadata
  - /isr_change_notification - Track changes to In Sync Replication
  - /log_dir_event_notification - Node for the broker to notify the controller about log dir events like disk failure
  - /delegation_token - Znode for storing delegation tokens
  - /controller_epoch - Track movement of controller
  - /consumers - Kafka Consumer list
  - /config - Entity configuration

  Note: The ZNodes do not contain customer data or PII information.

- Default ACLs:
  - /controller - world:anyone:cdrwa
  - /cluster - world:anyone:cdrwa
  - /brokers - world:anyone:cdrwa
  - /kafka-acl - world:anyone:cdrwa
  - /admin - world:anyone:cdrwa
  - /isr_change_notification - world:anyone:cdrwa
  - /log_dir_event_notification -world:anyone:cdrwa
  - /delegation_token - world:anyone:cdrwa
  - /controller_epoch - world:anyone:cdrwa
  - /consumers - world:anyone:cdrwa
  - /config - world:anyone:cdrwa
- Security Best Practice ACLs/Permissions and Required Steps:
  - /controller - sasl:kafka:cdrwa
  - /cluster - sasl:kafka:cdrwa
  - /brokers - sasl:kafka:cdrwa
  - /kafka-acl - sasl:kafka:cdrwa
  - /admin - sasl:kafka:cdrwa
  - /isr_change_notification - sasl:kafka:cdrwa
  - /log_dir_event_notification -sasl:kafka:cdrwa
  - /delegation_token - sasl:kafka:cdrwa
  - /controller_epoch - sasl:kafka:cdrwa
  - /consumers - sasl:kafka:cdrwa
  - /config - sasl:kafka:cdrwa

In a secure Kafka cluster Cloudera recommends that the Enable Zookeeper ACL (zookeeper.set.acl) property is set to true. You can configure this property in Cloudera Manager by going to Kafka Configuration . Once the property is set to true, run the zookeeper-security-migration tool with the zookeeper.acl option set to secure. Finally, reset the ACLs on the root node to allow full access.

For more information, see *Kafka security hardening with Zookeeper ACLs*.

Related Information

Kafka security hardening with ZooKeeper ACLs

ZooKeeper ACLs Best Practices: Oozie

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for Oozie when provisioning a secure cluster.

- ZooKeeper Usage:
  - Used to coordinate multiple Oozie servers.
- Default ACLs:

  In a secure cluster, Oozie restricts the access to Oozie Znodes to the oozie principals only using Kerberos backed ACLs.

  - /oozie - node that stores oozie server information in HA mode

  Default ACLs:

  - /oozie - world:anyone:cdrwa
  - /zkdtsm-oozie - node used for handling Oozie delegation tokens when the callback URL authentication is enabled

    ACLs:

    /zkdtsm-oozie - world:anyone:cdrwa
- Security Best Practice ACLs/Permissions and Required Steps:
  - If security is enabled in ZooKeeper, then Oozie connects to ZooKeeper using Kerberos, by default.

ZooKeeper ACLs Best Practices: Ranger

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for Ranger when provisioning a secure cluster.

- ZooKeeper Usage:
  - Ranger does not use ZooKeeper directly. Only if Audit to Solr is enabled and Solr is configured in SolrCloud mode, Solr nodes will need to access zookeeper node /ranger_audits.

    /ranger_audits
- Default ACLs:
  - /ranger_audits - world:anyone:cdrwa

- Security Best Practice ACLs/Permissions and Required Steps:
  - Only Solr needs access to this Znode:

    /ranger_audits - sasl:solr:cdrwa
  - After enabling SolrCloud, edit the Ranger collection path permission on Znode:

    1. SSH to the cluster.

       For example, any host where the ZooKeeper server is running.
    2. Start the ZooKeeper CLI by running the following command:

       ```
       zookeeper-client -server [***ZOOKEEPER SERVER HOST***]:[***ZOOKEEPER
        CLIENT PORT***]
       ```

       The default port of zookeeper client port is 2181.
    3. After it connects, run: ls /
    4. Verify there is a folder for the Ranger Solr collection.
    5. Run getAcl /ranger_audits and if the permission is for world,anyone: cdrwa, restrict the permission to “sas l:solr:cdrwa” using this command: setAcl /ranger_audits sasl:solr:cdrwa.
    6. Repeat the above step for all clusters where SolrCloud is installed.

    ```
    [zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 0] ls /
              [zookeeper, rmstore, ranger_audits]
              [zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 1] getAcl /ran
    ger_audits
              'world,'anyone
              : cdrwa
              [zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 2] setAcl /ra
    nger_audits sasl:solr:cdrwa
              cZxid = 0x200000037
              ctime = Wed Jun 29 10:40:24 UTC 2016
              mZxid = 0x200000037
              mtime = Wed Jun 29 10:40:24 UTC 2016
              pZxid = 0x200000056
              cversion = 7
              dataVersion = 0
              aclVersion = 1
              ephemeralOwner = 0x0
              dataLength = 0
              numChildren = 7
              [zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 3] getAcl /rang
    er_audits
              'sasl,'solr
              : cdrwa
              [zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 4]
    ```

ZooKeeper ACLs best practices: Search

You must follow certain best practices for effective control of client access to ZNodes using the ZooKeeper access-control lists (ACLs) or permissions for Search when provisioning a secure cluster.

- ZooKeeper Usage
  - Solr uses ZooKeeper to store cluster configuration (document collections, shard and replica configuration / location), cluster status (live servers, overseer metadata), collection configuration (schemas, settings, etc.), and

security-related information (delegation tokens). The root ZNode is configurable inCloudera Manager for each Solr service.

Note: Since users or administrators might upload data into ZNodes, remember that they should not upload sensitive data including PII or passwords in files such as collection configurations. Search content is not stored in ZooKeeper.

Note: For infrastructure services such as Ranger audits, a dedicated Solr service called Infra Solr is used, where the default ZNode is called /solr-infra.

- Default ACLs

  /solr

  - In a non-kerberized cluster: world:anyone:cdrwa
  - With Kerberos enabled: sasl:solr:cdrwa, world:anyone:r
- Security Best Practice ACLs/Permissions and Required Steps
  - ACLs are initialized during deployment, depending on whether Kerberos for Solr is enabled or not. Operators do not need to modify them by default.
  - When creating a new Solr collection, a set of configuration files need to be uploaded to ZooKeeper. Although it is possible to directly upload files (solrctl instancedir commands), because of ACL settings only the solr user is allowed to do this. Cloudera recommends using the Config and ConfigSets APIs (solrctl config commands) instead, which do not require direct ZooKeeper communication. As solrctl config commands communicate directly with Solr where you can configure role-based authorization using Ranger, this method is considered more secure.

- If you disable or enable Kerberos authentication for an existing Solr service in Cloudera Manager, the ACLs in ZooKeeper are not adjusted automatically. In case of disabling Kerberos this can prevent the recently unkerberized Solr service from writing data to ZooKeeper.

  To update ZooKeeper ACLs, you need to perform the following steps:

1. Shut down Solr.

   If you want to disable Kerberos, shut down Solr while Kerberos is still enabled. This allows you to use a valid Solr keytab.
2. Navigate to the /var/run/cloudera-scm-agent/process directory:

   ```
   cd /var/run/cloudera-scm-agent/process
   ```
3. Locate the current or most recent [***PROCESS_DIR_ID***]-solr-SOLR_SERVER directory, with the solr.keytab and jaas.conf files. The [***PROCESS_DIR_ID***] part changes every time the Solr server is restarted.

   To identify the current or most recent directory, list the /var/run/cloudera-scm-agent/process/*solr* directories sorted by time in reverse as follows:

   ```
   ls -ltrd /var/run/cloudera-scm-agent/process/*solr*
   ```

   The entry at the bottom is the current or most recent process directory.
4. Go to the Solr directory:

   ```
   cd [***PROCESS_DIR_ID***]-solr-SOLR_SERVER
   ```
5. Copy log4j2.properties temporarily to /opt/cloudera/parcels/CDH/lib/solr/bin, to get proper logging for the zkcli.sh script.

   ```
   cp log4j2.properties /opt/cloudera/parcels/CDH/lib/solr/bin/
   ```
6. Export the JVM flags for ZooKeeper client to configure ACLs.

   a. To enable Kerberos-specific ACLs:

   ```
   export ZKCLI_JVM_FLAGS="-Djava.security.auth.login.config=jaas.conf
   -DzkACLProvider=org.apache.solr.common.cloud.SaslZkACLProvider -Droo
   t.logger=INFO,console"
   ```

   b. To enable unkerberized ACLs:

   ```
   export ZKCLI_JVM_FLAGS="-Djava.security.auth.login.config=jaas.conf
   -DzkACLProvider=org.apache.solr.common.cloud.DefaultZkACLProvider -D
   root.logger=INFO,console"
   ```
7. Run the following command to update ACLs:

   ```
   /opt/cloudera/parcels/CDH/lib/solr/bin/zkcli.sh -zkhost [***ZOOKEEPER SE
   RVER HOST***]:[***ZOOKEEPER CLIENT PORT***] -cmd updateacls /solr
   ```

   The default [***ZOOKEEPER CLIENT PORT***] is 2181.
8. To verify that the ZooKeeper client uses jaas.conf and ACLs were updated, check the output of the following command:

   ```
   zookeeper-client -server [***ZOOKEEPER SERVER HOST***]:[***ZOOKEEPER CLI
   ENT PORT***] getAcl /solr
   ```

   The default [***ZOOKEEPER CLIENT PORT***] is 2181.

For a non-kerberized ACL setup, the command output looks like this:

```
'world,'anyone
: cdrwa
```

For a kerberized ACL setup, the command output looks like this:

```
'sasl,'solr
: cdrwa
'world,'anyone
: r
```

9. Remove log4j2.properties from /opt/cloudera/parcels/CDH/lib/solr/bin:

```
rm /opt/cloudera/parcels/CDH/lib/solr/bin/log4j2.properties
```

ZooKeeper ACLs Best Practices: YARN

YARN related ZooKeeper ACLs are automatically created using Cloudera Manager. Review the list of default ACLs to ensure they are set as recommended for YARN.

Cloudera Manager automatically sets some ZooKeeper ACLs related YARN properties that are used by the YARN service to set up the default ZooKeeper ACLs. That means no manual configuration step is needed. However, customized principles are set by Cloudera Manager only at first launch. For any later launches Clouder Manager checks only the root znodes.

- ZooKeeper Usage:
  - /yarn-leader-election - used for RM leader election
  - /rmstore - used for storing RM application state
- Default ACLs:
  - /yarn-leader-election - sasl:*[***CUSTOMIZED PRINCIPLE UPON FIRST LAUNCH***]*:cdrwa
  - /rmstore - sasl:*[***CUSTOMIZED PRINCIPLE UPON FIRST LAUNCH***]*:cdrwa

If default ACLs are set incorrectly, perform one of the following workarounds:

- Delete the znode and restart the YARN service.
- Use the reset ZK ACLs command. This also sets the znodes below /rmstore/ZKRMStateRoot to world:anyone:cdrwa which is less secure.

ZooKeeper ACLs Best Practices: ZooKeeper

You must follow the best practices for tightening the ZooKeeper ACLs or permissions for ZooKeeper when provisioning a secure cluster.

- ZooKeeper Usage:
  - /zookeeper - Node stores metadata of ZooKeeper itself.
  - /zookeeper/quota - Stores quota information.
  - /zookeeper/config - Stores the current configuration, if the dynamic reconfiguration feature is enabled.
- Default ACLs:
  - /zookeeper - world:anyone:cdrwa
  - /zookeeper/quota - world:anyone:cdrwa
  - /zookeeper/config - world:anyone:r

- Security Best Practice ACLs/Permissions and Required Steps:

  The following steps must be manually performed by users who are using the ZooKeeper quota or dynamic reconfig feature. Components in CDP do not use these features, so these ZNodes are acutally empty by default -- most users do not need to run the following commands:

  - setAcl /zookeeper sasl:zookeeper:rwcda
  - setAcl /zookeeper/quota sasl:zookeeper:cdrwa,world:anyone:r